PANDUAN DEFINITIF UNTUK POWER QUERY (M)

Dr. Phil. Dony Novaliendry, S. Kom., M. Kom

Buku "Panduan Definitif untuk Power Query (M) – Jilid 1" hadir sebagai sumber belajar komprehensif dalam memahami dan menguasai transformasi data kompleks menggunakan Power Query. Disusun secara sistematis, buku ini menguraikan konsep dasar hingga penerapan praktis bahasa M, yang menjadi inti dari Power Query.

Pembahasan dimulai dari pengenalan M, sejarah, karakteristik, hingga alasan mengapa M penting untuk dikuasai. Selanjutnya, buku ini memandu pembaca bekerja langsung dengan Power Query/M, mulai dari eksplorasi antarmuka, pengolahan data, pembuatan kolom khusus, hingga penggunaan editor tingkat lanjut.

Pada bagian berikutnya, pembaca diperkenalkan dengan cara mengakses, menggabungkan, dan menyatukan berbagai sumber data—mulai dari file, folder, konten web, hingga database—dilengkapi dengan teknik optimalisasi penggunaan fungsi bawaan M. Bab terakhir dalam jilid ini membahas nilai dan ekspresi, mencakup tipe data, operator, ekspresi bersarang, serta praktik terbaik dalam pengkodean.

D ilengkapi dengan kasus pemantik berpikir kritis, tes formatif, glosarium, dan lampiran, buku ini dirancang untuk memudahkan pemahaman sekaligus memperkuat keterampilan analisis data. Dengan pendekatan praktis, buku ini dapat menjadi referensi penting bagi peserta didik, dosen, praktisi data, maupun siapa saja yang ingin memperdalam pemanfaatan Power Query dalam dunia nyata.

Appti
IKAPI
PENERBITAN & PERCETAKAN UNP PRESS
Jln. Prof. Dr. Hamka Air Tawar Padang
Sumatera Barat
UNP PRESS

PANDUAN DEFINITIF UNTUK POWER QUERY (M)
Dr. Phil. Dony Novaliendry, S. Kom., M. Kom

PANDUAN DEFINITIF UNTUK POWER QUERY (M)

JILID 1

Penerbitan & Percetakan
UNP PRESS

Dr. Phil. Dony Novaliendry, S. Kom., M. Kom

PANDUAN DEFINITIF UNTUK POWER QUERY (M) JILID 1

Dr. Phil. Dony Novaliendry, S. Kom., M. Kom

DUMMY
Penerbitan & Percetakan
UNP PRESS

UNDANG-UNDANG REPUBLIK INDONESIA
NO 19 TAHUN 2002
TENTANG HAK CIPTA
PASAL 72
KETENTUAN PIDANA SANGSI PELANGGARAN

1. Barang siapa dengan sengaja dan tanpa hak mengumumkan atau memperbanyak suatu Ciptaan atau memberi izin untuk itu, dipidana dengan pidana penjara paling singkat 1 (satu) bulan dan denda paling sedikit Rp 1.000.000, 00 (satu juta rupiah), atau pidana penjara paling lama 7 (tujuh) tahun dan denda paling banyak Rp 5.000.000.000, 00 (lima milyar rupiah)
2. Barang siapa dengan sengaja menyerahkan, menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu Ciptaan atau barang hasil pelanggaran Hak Cipta atau Hak Terkait sebagaimana dimaksud dalam ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan denda paling banyak Rp 500.000.000, 00 (lima ratus juta rupiah).

PANDUAN DEFINITIF UNTUK POWER QUERY (M) JILID 1

Dr. Phil. Dony Novaliendry, S. Kom., M. Kom

DUMMY
Penerbitan & Percetakan
UNP PRESS

UNP PRESS

2025

PANDUAN DEFINITIF UNTUK POWER QUERY (M) JILID 1

editor, Tim editor UNP Press
Penerbit UNP Press, Padang, 2025
1 (satu) jilid; 17.6 x 25 cm (B5)
Jumlah Halaman ix + 226 Halaman Buku

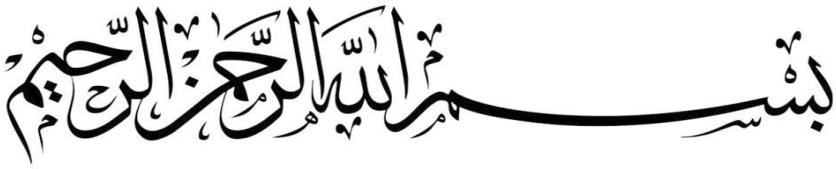
ISBN:

DUMMY
Penerbitan & Percetakan
UNP PRESS

PANDUAN DEFINITIF UNTUK POWER QUERY (M)

Penyusun: Dr. Phil. Dony Novaliendry, S. Kom., M. Kom
Editor Substansi: Fadhillah Majid Saragih, S.Pd., M.Pd.T
Editor Bahasa: Prof. Dr. Harris Effendi Thahar, M.Pd.
Desain Sampul & Layout: Mukhlis Zaki Insani

KATA PENGANTAR



Puji syukur kehadirat Allah SWT atas berkat limpahan rahmat dan karunia-Nya sehingga Buku ajar Panduan Definitif untuk Power Query (M) dapat di buat. Buku ajar Panduan Definitif untuk Power Query (M) adalah buku ajar yang membahas tentang Menguasai Transformasi Data Kompleks dengan Power Query yang bisa di manfaatkan oleh peserta didik khususnya dan bagi semua pihak dari segala lapisan yang membutuhkan sebagai referensi untuk belajar mengunakan Buku ajar Panduan Definitif untuk Power Query (M).

Kami mengucapkan terima kasih kepada semua pihak yang telah membantu dalam proses penyelesain buku ajar ini.

Kami menyadari masih terdapat banyak kekurangan dalam buku ajar ini untuk itu kritik dan saran yang membangun demi penyempurnaan buku ajar ini sangat diharapkan. Dan semoga buku ini dapat memberikan maanfaat bagi para peserta didik khususnya dan bagi semua pihak dari segala lapisan yang membutuhkan.

Padang, Juni 2025

Penulis

DAFTAR ISI

DUMMY
Penerbitan & Percetakan
UNP PRESS

DAFTAR GAMBAR

DUMMY
Penerbitan & Percetakan
UNP PRESS

DAFTAR TABEL

Pendahuluan

Dalam era digital yang semakin berkembang, data telah menjadi salah satu aset terpenting bagi organisasi dan individu. Kemampuan untuk mengolah, menganalisis, dan memvisualisasikan data dengan efektif menjadi kunci untuk mengambil keputusan yang tepat dan strategis. Salah satu alat yang sangat berguna dalam proses ini adalah Power Query, sebuah fitur yang tersedia dalam Microsoft Excel dan Power BI. Power Query memungkinkan pengguna untuk mengimpor, membersihkan, dan mengubah data dari berbagai sumber dengan cara yang intuitif dan efisien.

Buku ajar ini hadir sebagai panduan definitif untuk memahami dan memanfaatkan Power Query secara maksimal. Dengan pendekatan yang sistematis, buku ini dirancang untuk membantu pembaca dari berbagai latar belakang, baik pemula maupun yang sudah berpengalaman, dalam menguasai bahasa pemrograman M yang menjadi dasar dari Power Query. Melalui penjelasan yang jelas dan contoh-contoh praktis, pembaca akan diajak untuk menjelajahi berbagai fitur dan kemampuan yang ditawarkan oleh Power Query.

Dalam setiap bab, pembaca akan menemukan langkah-langkah yang terperinci untuk melakukan berbagai tugas, mulai dari pengambilan data hingga transformasi yang kompleks. Buku ini juga akan membahas berbagai teknik dan trik yang dapat meningkatkan efisiensi kerja, serta cara mengatasi tantangan yang sering dihadapi saat bekerja dengan data. Dengan demikian, pembaca tidak hanya akan belajar cara menggunakan Power Query, tetapi juga memahami prinsip-prinsip dasar yang mendasari pengolahan data.

Akhirnya, kita berharap buku ini dapat menjadi sumber referensi yang berguna dan inspiratif bagi siapa saja yang ingin meningkatkan keterampilan analisis data mereka. Dengan menguasai Power Query, pembaca akan memiliki

alat yang kuat untuk mengubah data menjadi informasi yang berharga, mendukung pengambilan keputusan yang lebih baik, dan pada akhirnya, mencapai tujuan yang diinginkan dalam dunia yang semakin didorong oleh data. Selamat membaca dan selamat berpetualang dalam dunia Power Query!

1. **Unduh File Kode Contoh**

   Kumpulan kode untuk buku ini dihosting di GitHub di https://github.com/PacktPublishing/The- Definitive-Guide-to-Power-Query-M-/. Kita juga memiliki kumpulan kode lain dari katalog buku dan video kita yang lengkap yang tersedia di https://github.com/PacktPublishing/. Lihatlah!

2. **Unduh Gambar Berwarna**

   Kita juga menyediakan berkas PDF yang berisi gambar berwarna dari cuplikan layar/diagram yang digunakan dalam buku ini. Anda dapat mengunduhnya di sini: https://packt.link/gbp/9781835089729.

3. **Konvensi yang Digunakan**

   Ada sejumlah konvensi teks yang digunakan di seluruh buku ini. *CodeInText*: Menunjukkan kata kode dalam teks, nama tabel basis data, nama folder, nama file, ekstensi file, nama jalur, URL tiruan, input pengguna, dan akun Twitter. Misalnya: “Arahkan ke folder /ClientApp/src/app/cities.” Blok kode ditetapkan sebagai berikut:

```
#date(
  year as number,
  month as number,
  day as number,
) as date
```

   Bila kita ingin menarik perhatian Anda ke bagian tertentu dari blok kode, baris atau item yang relevan disorot:

```
#date(
  year as number,
  month as number,
  day as number,
) as date
```

Tebal: Menunjukkan istilah baru, kata penting, atau kata-kata yang Anda lihat di layar. Misalnya, kata-kata dalam menu atau kotak dialog muncul dalam teks seperti ini. Misalnya: “Arahkan ke tab Beranda pada pita, klik menu tarik-turun di bawah tombol Ubah data, dan pilih Edit parameter.”

DUMMY
Penerbitan & Percetakan
UNP PRESS

BAB 1

Memperkenalkan M

Topik Bab

Pada bab ini, topik yang akan dibahas adalah:

- Sejarah M
- Siapa yang harus mempelajari M?
- Di mana dan bagaimana M digunakan?
- Mengapa belajar M?
- Dasar-dasar bahasa M
- Karakteristik M

A. Pendahuluan

M adalah bahasa rumus yang kuat dan serbaguna yang dirancang khusus untuk manipulasi dan transformasi data. Istilah M adalah sebutan informal. Nama resmi M adalah Bahasa Rumus Power Query. Untuk penjelasan tentang sebutan ini, lihat bagian Sejarah M di bagian akhir bab ini. M adalah bahasa inti Power Query, yang digunakan dalam berbagai aplikasi seperti Microsoft Excel, Power BI, Power Platform, dan Microsoft Fabric untuk transformasi dan persiapan data.

Popularitas bahasa M terus tumbuh selama dekade terakhir, dan bahasa tersebut telah diintegrasikan ke dalam serangkaian alat dan platform Microsoft yang mengesankan. Saat ini, M dan Power Query adalah alat yang sangat diperlukan bagi para profesional data modern seperti analis bisnis, ilmuwan data, dan penggemar data.

Bab ini adalah awal dari perjalanan menarik Anda yang berpuncak pada penguasaan bahasa M. Kita mulai dengan sejarah singkat M dan kemudian membahas dasar-dasar tentang siapa, di mana, mengapa, dan bagaimana. Selanjutnya, kita memperkenalkan dasar-dasar mutlak bahasa M dan diakhiri dengan karakteristik formal dan informal M (sebenarnya, apa itu M?). Secara keseluruhan, bab ini menyediakan landasan yang kuat untuk penjelajahan bahasa M yang lebih mendalam yang ditemukan di seluruh bagian buku ini. Secara khusus, bab ini mencakup topik-topik berikut:

- Sejarah M
- Siapa yang harus mempelajari M?
- Di mana dan bagaimana M digunakan?
- Mengapa belajar M?
- Dasar-dasar bahasa M
- Karakteristik M

1. Kasus Pemantik Berpikir Kritis: Analisis Data Penjualan

Sebuah perusahaan retail ingin menganalisis data penjualan mereka untuk meningkatkan strategi pemasaran dan pengelolaan inventaris. Mereka memiliki data penjualan yang tersebar di beberapa file Excel dan database yang berbeda. Manajer pemasaran meminta tim analisis data untuk menggunakan Power Query untuk mengumpulkan dan menganalisis data tersebut.

1) Identifikasi Sumber Data:
   Apa saja sumber data yang mungkin perlu diintegrasikan untuk analisis penjualan ini? Bagaimana Anda menentukan sumber mana yang paling relevan?
2) Proses Pengumpulan Data:
   Apa langkah-langkah yang harus diambil untuk mengimpor data dari berbagai sumber ke dalam Power Query? Apa tantangan yang mungkin dihadapi selama proses ini?
3) Transformasi Data:
   Data yang diimpor mungkin tidak dalam format yang siap untuk analisis. Apa jenis transformasi yang perlu dilakukan untuk memastikan data tersebut dapat digunakan secara efektif?
4) Analisis Kualitas Data:
   Bagaimana Anda akan mengevaluasi kualitas data yang telah diimpor? Apa indikator yang dapat digunakan untuk menilai apakah data tersebut dapat diandalkan?
5) Penggunaan Filter dan Kriteria:
   Dalam analisis penjualan, bagaimana Anda akan menggunakan fitur filter di Power Query untuk mendapatkan wawasan yang lebih mendalam? Berikan contoh kriteria yang mungkin digunakan.
6) Visualisasi Hasil:

7) Setelah data dianalisis, bagaimana Anda akan menyajikan hasil analisis kepada manajemen? Apa jenis visualisasi yang paling efektif untuk menyampaikan informasi tersebut?
8) Pengambilan Keputusan:

   Berdasarkan analisis yang dilakukan, keputusan apa yang dapat diambil oleh manajemen untuk meningkatkan penjualan? Apa langkah-langkah selanjutnya yang harus diambil?

B. Sejarah M

Proses mengekstraksi, mengubah, dan memuat data merupakan tantangan setua teknologi informasi itu sendiri. Baik pengguna bisnis maupun profesional TI secara historis telah berjuang menghadapi tantangan tersebut, dan banyak perangkat lunak telah dikembangkan selama bertahun-tahun untuk membantu mengatasi tantangan tersebut seperti SQL **Server Integration Services (SSIS)** dan **Alteryx**.

Akan tetapi, banyak dari alat-alat ini yang rumit dan tidak mudah dibawa. Bahasa M dan Power Query diciptakan untuk membantu memecahkan masalah-masalah ini.

Meskipun mungkin ada beberapa sejarah yang lebih spekulatif mengenai asal-usul M, setidaknya kita dapat melacak M secara definitif kembali ke sebuah proyek yang awalnya diberi nama kode **Data Explorer**. Data Explorer adalah proyek Azure SQL Labs sekitar tahun 2011 yang bertujuan untuk menyederhanakan proses mengakses, membersihkan, dan menyiapkan data dari berbagai sumber. Bahasa kueri dianggap sebagai bahasa mashup (maka dari itu M untuk mashup).

Pada tahun 2013, Microsoft merilis Power Query sebagai add-in untuk Excel. Power Query memperkenalkan antarmuka yang mudah digunakan, yang memungkinkan pengguna bisnis untuk melakukan transformasi data melalui editor visual. Di balik layar, Power Query menggunakan bahasa M sebagai

bahasa rumus yang mendasarinya untuk mendorong transformasi data, dan dengan demikian, transformasi data ini menjadi dapat diulang. Alih-alih, misalnya, pengguna bisnis terus-menerus melakukan transformasi data manual yang sama pada data sumber yang diterima sebagai file yang dibatasi koma, proses tersebut sekarang dapat diotomatisasi secara efektif.

Setelah keberhasilan Power Query di Excel, Microsoft memasukkan Power Query sebagai bagian dari produk barunya, Power BI Designer, yang akhirnya menjadi Power BI Desktop. Seiring dengan semakin populernya Power Query, muncul kebutuhan untuk menstandardisasi bahasa rumus yang mendasarinya. Pada tahun 2016, Microsoft mengajukan spesifikasi Bahasa Rumus Power Query ke **European Computer Manufacturers Association (ECMA)**, sebuah organisasi standar internasional. Upaya ini menetapkan spesifikasi formal untuk bahasa tersebut, yang memastikan kompatibilitas dan interoperabilitas antara implementasi yang berbeda.

Meskipun bahasa tersebut secara formal disebut sebagai Bahasa Rumus Power Query, bahasa tersebut kemudian dikenal sebagai M di antara komunitas pengguna. Nama informal M diterima secara luas dan sekarang digunakan secara luas untuk merujuk ke bahasa tersebut.

Microsoft terus meningkatkan dan menyempurnakan bahasa M sebagai bagian dari investasi berkelanjutannya dalam teknologi integrasi dan transformasi data. Fungsi, fitur, dan penyempurnaan baru diperkenalkan secara berkala untuk menyediakan cara yang lebih canggih dan efisien bagi pengguna untuk memanipulasi dan menyiapkan data mereka. Selain itu, Microsoft terus memperkenalkan M dalam perangkat lunak dan platform tambahan, seperti integrasi data dalam Microsoft Power Platform dan aliran data dalam Power BI dan Fabric.

Saat ini, bahasa **M** merupakan komponen utama dari perangkat transformasi dan integrasi data Microsoft. Perkembangan **M**, serta fleksibilitas dan

ekstensibilitasnya, menjadikannya bahasa yang sangat berharga bagi para profesional data modern saat ini.

Sekarang mari kita alihkan perhatian kita kepada siapa yang harus mempelajari M.

C. Siapa yang Harus Belajar M?

M adalah alat yang ampuh bagi para profesional data dan individu yang bekerja dengan data secara rutin. Fleksibilitas dan kemampuan M menjadikannya bahasa yang berharga untuk dipelajari untuk berbagai peran, termasuk yang berikut:

- **Analis data**: Analis data yang menangani tugas ekstraksi, transformasi, dan persiapan data dapat memperoleh manfaat besar dari mempelajari M. M menyediakan serangkaian fungsi dan operator komprehensif yang memungkinkan analis data untuk membentuk dan memanipulasi data dari berbagai sumber secara efisien. Dengan menguasai M, analis data dapat mengotomatiskan tugas berulang, menangani transformasi data yang kompleks, dan memastikan kualitas data, yang mengarah pada analisis data yang lebih akurat dan andal.
- **Profesional intelijen bisnis**: Profesional di bidang **business intelligence (BI)** dapat meningkatkan keterampilan mereka secara signifikan dengan mempelajari M. M merupakan komponen inti dari Power BI. Dengan memahami M, profesional BI memperoleh kemampuan untuk terhubung ke berbagai sumber data, melakukan transformasi data yang kompleks, dan membuat alur kerja persiapan data yang dapat digunakan kembali, yang memungkinkan mereka untuk memberikan wawasan yang dapat ditindaklanjuti dan mendorong pengambilan keputusan yang tepat.
- **Insinyur data**: Insinyur data yang terlibat dalam desain dan implementasi jalur data dan proses integrasi data benar-benar dapat memperoleh manfaat dari mempelajari M. M memungkinkan insinyur data untuk **extract,**

transform, and **load (ETL)** data secara efisien dari berbagai sumber ke dalam gudang data atau danau data, khususnya dalam Power BI dan Microsoft Fabric. M juga menyediakan fleksibilitas dan kekuatan untuk menangani format data yang kompleks, menentukan transformasi khusus, dan menciptakan alur kerja pemrosesan data yang efisien. Dengan menguasai M, teknisi data dapat menyederhanakan proses integrasi data dan memastikan konsistensi dan kualitas data.

- **Ilmuwan data**: Ilmuwan data yang melakukan analisis data eksploratif, pengembangan model, dan analitik tingkat lanjut dapat memanfaatkan kemampuan M untuk menyiapkan data mereka secara efisien. M menyediakan serangkaian fungsi yang kuat untuk membersihkan, membentuk, dan menggabungkan data, yang memungkinkan ilmuwan data untuk fokus pada aspek analitis pekerjaan mereka. Dengan menggabungkan M ke dalam alur kerja persiapan data mereka, ilmuwan data dapat menyederhanakan alur kerja untuk mengubah data mentah menjadi wawasan, menghabiskan lebih sedikit waktu untuk pembersihan dan persiapan data, dan lebih banyak waktu untuk pemodelan dan analisis data.
- **Pengguna ahli**: Pengguna ahli di Excel dan Power Platform Microsoft yang bekerja secara ekstensif dengan data dan melakukan manipulasi data yang kompleks dapat memperoleh manfaat dari mempelajari M. M diintegrasikan ke dalam Excel melalui Power Query, yang memberdayakan pengguna untuk melakukan transformasi data tingkat lanjut dalam antarmuka Excel yang familier. Selain itu, M diintegrasikan ke dalam Microsoft Power Platform melalui integrasi data, yang memungkinkan data untuk diubah dan dipetakan antar sistem. Dengan menguasai M, pengguna ahli dapat memperluas kemampuan manipulasi data mereka, mengotomatiskan tugas-tugas yang berulang, dan meningkatkan akurasi dan keandalan analisis mereka. • Individu dalam peran yang digerakkan oleh data: Di luar peran-peran khusus yang disebutkan di atas, individu dalam berbagai peran yang

digerakkan oleh data, seperti manajer proyek, konsultan, peneliti, dan pakar domain, dapat memperoleh manfaat dari mempelajari M. Penguasaan M memberikan kemampuan bagi individu dalam peran-peran ini untuk secara independen menangani tugas-tugas yang terkait dengan data, mengekstrak wawasan yang berarti, dan membuat keputusan yang tepat berdasarkan data yang dapat diandalkan.

M adalah bahasa yang berharga bagi berbagai profesional data dan individu lain yang bekerja dengan data. Apakah Anda seorang analis data, profesional BI, insinyur data, ilmuwan data, pengguna berpengalaman, atau seseorang yang berkecimpung dalam peran yang digerakkan oleh data, mempelajari M memberdayakan Anda untuk menyerap, mengubah, dan menyiapkan data secara efisien untuk analisis.

Sekarang setelah kita memahami tipe individu seperti apa yang ingin mempelajari M, selanjutnya mari kita telusuri di mana individu tersebut dapat memanfaatkan penguasaan mereka terhadap M.

D. Di mana dan Bagaimana M Digunakan?

M adalah bahasa serbaguna yang disertakan dalam berbagai alat dan platform yang sangat membutuhkan transformasi dan manipulasi data. Integrasinya dalam ekosistem Power Query memungkinkan pengguna memanfaatkan kemampuan M di berbagai lingkungan. Di bagian ini, kita akan membahas beberapa area utama tempat M digunakan secara luas.

1. Pengalaman

Sebelum membahas produk tertentu yang menggunakan M, penting untuk memahami berbagai pengalaman yang tersedia untuk membuat M. Ada dua pengalaman yang tersedia untuk membuat M, satu ditujukan untuk penggunaan di tempat dan yang lainnya untuk aplikasi berbasis cloud. Pengalaman-pengalaman ini adalah sebagai berikut:

- **Power Query Desktop**: Power Query Desktop adalah pengalaman untuk Power Query yang ditemukan dalam aplikasi desktop seperti Power BI Desktop dan Microsoft Excel. Meskipun pengalamannya serupa, ada perbedaan. Misalnya, integrasi **artificial intelligence (AI)** dan **machine learning (ML)** serta integrasi dengan R dan Python yang ada di Power BI Desktop tidak ada di Microsoft Excel. Sebaliknya, opsi **Kolom Terstruktur** yang tersedia di Excel tidak tersedia di Power BI Desktop.
- **Power Query Online**: Power Query Online, layanan berbasis cloud, memungkinkan pengguna untuk membuat dan mengelola transformasi data dalam browser web. M digunakan secara luas di Power Query Online untuk menentukan transformasi data, menghubungkan ke sumber data, dan melakukan manipulasi data yang kompleks. Pengguna dapat mengakses dan mengedit kueri M secara langsung dalam antarmuka browser, sehingga memudahkan kolaborasi dan mengerjakan tugas transformasi data dari mana saja dengan koneksi internet. Power Query Online terintegrasi ke dalam berbagai produk Microsoft, termasuk layanan Power BI, Power Apps, Power Automate, dll.

Penting untuk dicatat bahwa meskipun ada dua pengalaman berbeda dalam penulisan M, keduanya memberikan pengalaman pengguna yang hampir sama persis. Yang lebih baik lagi, keduanya memberikan kemampuan untuk mengedit kode M yang mendasarinya, yang merupakan fokus utama buku ini. Dengan demikian, keterampilan yang dipelajari di sini berlaku sama untuk kedua pengalaman yang digunakan dalam produk atau layanan apa pun.

2. Produk dan layanan

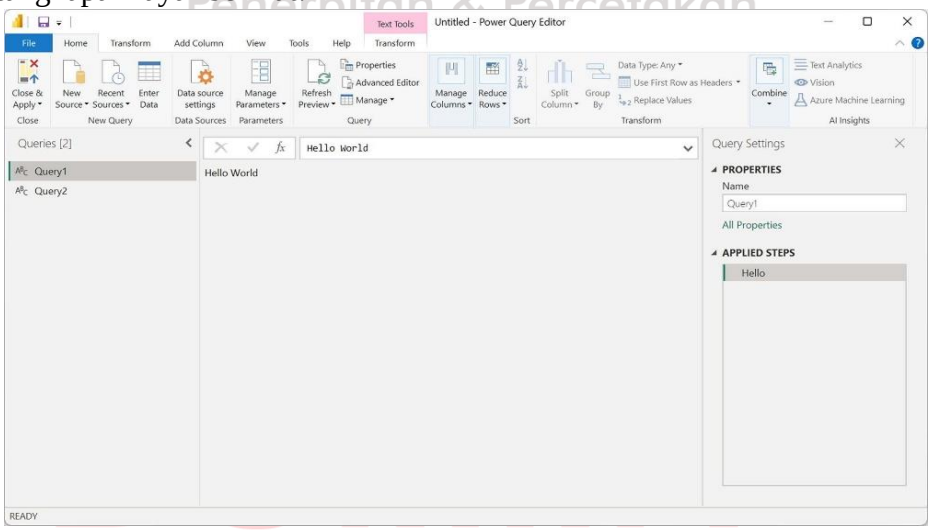
M ada di mana-mana dalam ekosistem Microsoft, termasuk perangkat lunak dan layanan berikut:

- **Aliran Data**: Aliran data adalah kueri M berbasis cloud yang tidak bergantung pada produk yang dapat digunakan kembali di berbagai produk yang berbeda. Aliran data memungkinkan pengguna untuk membangun dan mengelola proses penyiapan dan transformasi data yang dapat digunakan kembali. Aliran data memanfaatkan pengalaman Power Query Online.
- **Power BI Desktop**: M adalah komponen dasar Power BI Desktop, alat BI terkemuka. M memungkinkan pengguna untuk terhubung ke berbagai

sumber data, melakukan transformasi data, dan membuat visualisasi serta laporan interaktif.

M memungkinkan pengguna untuk mengekstrak, membersihkan, dan membentuk data dari berbagai sumber, seperti basis data, file Excel, layanan web, dan banyak lagi. Dengan M, pengguna dapat menentukan langkah-langkah transformasi data dan membuat kueri yang dapat digunakan kembali yang menyegarkan dan memperbarui data secara otomatis saat sumber yang mendasarinya berubah.

Di dalam Power BI Desktop, M digunakan dalam editor Power Query, subprogram yang diluncurkan dari dalam Power BI Desktop. Editor Power Query menyediakan **graphical user interface (GUI)** yang canggih untuk bekerja dengan bahasa rumus M, seperti yang ditunjukkan dalam tangkapan layar berikut:

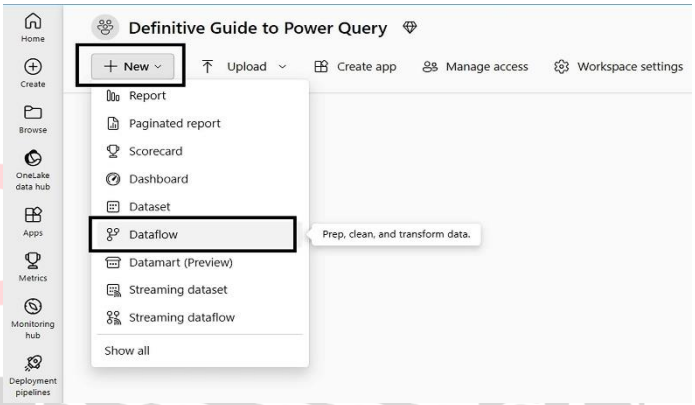


Gambar 1. 1 Editor Power Query di Power BI Desktop

Editor Power Query dibahas lebih rinci di Bab 2, Bekerja dengan Power Query/M. Power BI Desktop juga mendukung penggunaan aliran data.

- **Layanan Power BI/Fabric**: Layanan Power BI/Fabric (powerbi.com) adalah komponen Power BI berbasis awan yang memungkinkan Anda berbagi laporan, dasbor, dan konten lainnya. Layanan ini mendukung penggunaan kode M melalui pembuatan aliran data, menggunakan pengalaman Power Query Online.

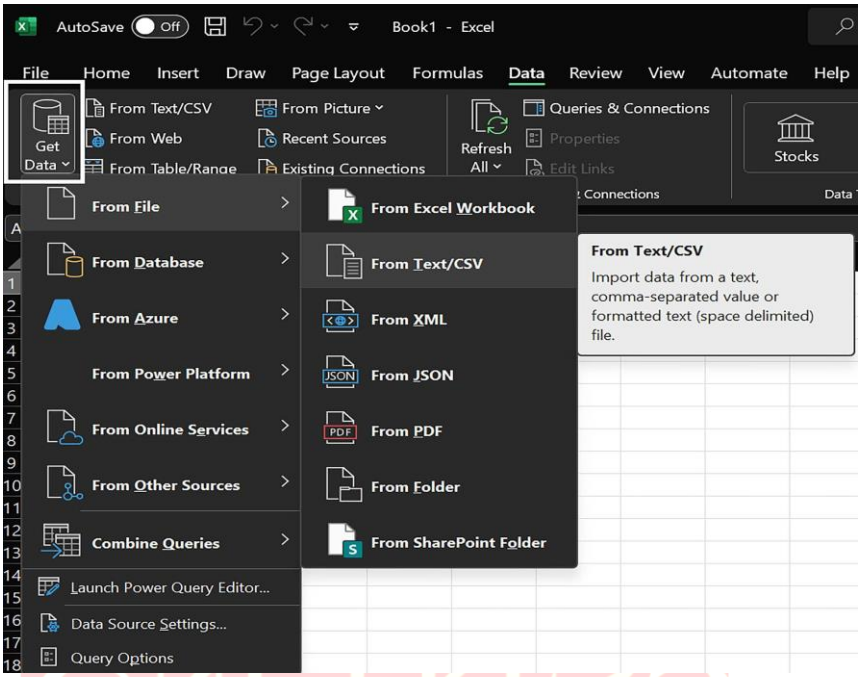
Untuk membuat aliran data di layanan Power BI, navigasikan ke ruang kerja mana pun selain **My Work-space** dan pilih **New** lalu **Datafiow**, seperti yang ditunjukkan pada Gambar 1.2:



Gambar 1. 2 Membuat aliran data di layanan Power BI

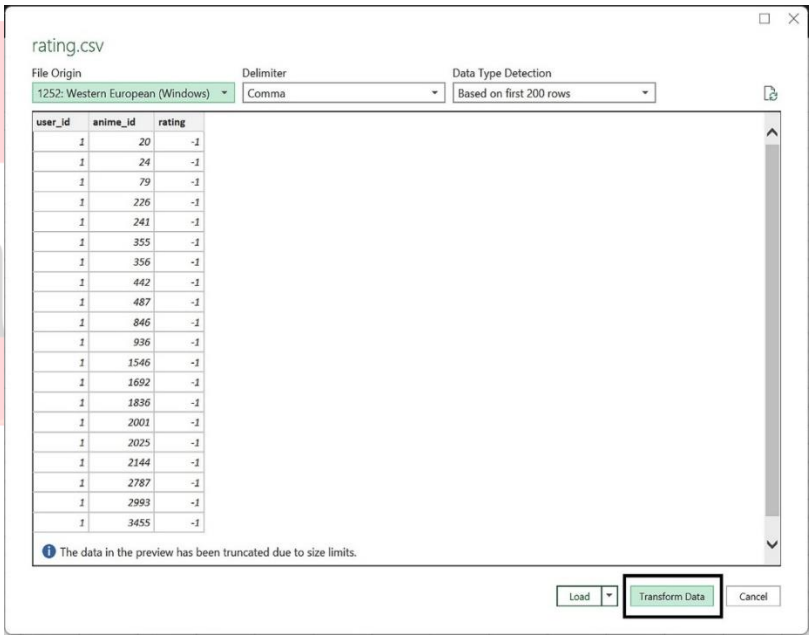
- **Power BI Report Server: Power BI Report Server** (PBRS) mendukung pengalaman Power Query Desktop, yang memungkinkan pengguna membuat transformasi data yang kaya melalui M.
- **Excel (Windows dan Macintosh)**: M terintegrasi dengan mulus ke dalam Excel, memberdayakan pengguna untuk melakukan transformasi data tingkat lanjut dalam antarmuka Excel yang familier. Power Query, mesin di balik kemampuan transformasi data Excel, didukung oleh M. Pengguna dapat mengakses editor Power Query di Excel untuk menerapkan transformasi M, memfilter dan mengurutkan data, menghapus duplikat, menggabungkan dan menambahkan tabel, dan melakukan tugas persiapan data lainnya. M memungkinkan pengguna untuk membersihkan, membentuk ulang, dan memperkaya data di Excel, yang meningkatkan akurasi dan keandalan analisis mereka.

  Di Excel, antarmuka editor Power Query dapat diakses dengan menggunakan tab **Data** pada pita dan memilih **Get Data**:



Gambar 1. 3 Mendapatkan Data di Microsok Excel

Setelah sumber data dipilih, antarmuka Editor Power Query dapat diakses dengan memilih tombol **Transform Data**:



Gambar 1. 4 Opsi Transformasi Data di Microsok Excel

Baik versi Windows maupun Macintosh Excel juga mendukung akses dan penggunaan alur data.

- **Power Apps**: Power Apps adalah platform low-code Microsoft untuk membuat aplikasi. Baik pengalaman Power Query Online maupun penggunaan alur data didukung. Kasus penggunaan umum adalah memanfaatkan M baik melalui pengalaman Power Query Online maupun alur data, yang memungkinkan pengguna untuk membawa data mereka ke Dataverse (sebelumnya Common Data Service) dengan lancar.
- **Power Automate**: Power Automate adalah platform low-code Microsoft untuk mengotomatiskan alur kerja. Power Automate memungkinkan pengguna untuk mengotomatiskan alur kerja dan proses berulang yang mungkin melibatkan manipulasi data dan tugas integrasi. M dapat digunakan dalam Power Automate untuk melakukan transformasi data dan menangani skenario data yang kompleks sebagai bagian dari alur kerja otomatis, melalui pengalaman Power Query Online. Dengan menggabungkan M ke dalam Power Automate, pengguna dapat membangun solusi integrasi dan otomatisasi data canggih yang menyederhanakan proses bisnis mereka. Selain itu, alur data dapat dimanfaatkan dalam Power Automate melalui konektor Power Query Dataflows. Hal ini memungkinkan tindakan terjadi setelah aliran data selesai dan juga menyediakan kemampuan bagi aliran data untuk dimulai sebagai tindakan dalam aliran Power Automate.
- **Data Factory**: Data Factory adalah layanan cloud terkelola yang secara khusus dibuat untuk proyek integrasi **extract-transform-load (ETL)** dan **extract-load-transform (ELT)** yang kompleks. Data Factory memungkinkan pembuatan dan orkestrasi alur kerja berbasis data, pergerakan data, dan transformasi dalam skala besar. Baik Azure Data Factory maupun Data Factory di Microsoft Fabric mendukung kode M, baik melalui pengalaman Power Query Online maupun aliran data.

- **SQL Server**: SSIS mendukung mesin inti M sementara **SQL Server Analysis Services (SSAS)** mendukung pengalaman Power Query Desktop.
- **Dynamics 365 Customer Insights**: Customer Insights dalam Dynamics 365 adalah **customer data platform (CDP)** Microsoft yang menyediakan tampilan holistik pelanggan, yang memungkinkan pengalaman pelanggan yang dipersonalisasi. Customer Insights mendukung aliran data maupun pengalaman Power Query Online.
- **Visual Studio**: Visual Studio memungkinkan M untuk diintegrasikan sebagai bahasa. Hal ini dilakukan melalui Layanan Bahasa Power Query untuk Visual Studio Code dan tersedia di Visual Studio Code Marketplace. Layanan bahasa ini menyediakan pelengkapan otomatis fuzzy, hover, petunjuk fungsi, dan fungsionalitas lain untuk menulis kode M dalam Visual Studio.

  Ada juga **Software Development Kit (SDK)** Visual Studio Power Query. SDK ini terdiri dari serangkaian alat yang dirancang untuk membantu membuat konektor sumber data Power Query kustom. SDK Visual Studio Power Query dibahas lebih rinci dalam GBAPTER 16, Mengaktifkan Ekstensi.
- **Skenario integrasi data lainnya**: M tidak terbatas pada perangkat lunak dan layanan yang disebutkan di atas. M juga dapat dimanfaatkan dalam aplikasi kustom dan lingkungan pemrograman yang menggunakan pustaka Power Query.

Seperti yang dapat Anda lihat, M digunakan secara luas di berbagai alat dan platform dalam ekosistem Microsoft, seperti Power BI Desktop, Excel, layanan Power BI dan Fabric, Power Platform, SQL Server, dan Dynamics. M memungkinkan pengguna untuk terhubung ke berbagai sumber data, melakukan transformasi data tingkat lanjut, dan mengotomatiskan alur kerja integrasi data. Keterampilan yang dipelajari dalam buku ini berhubungan

dengan bahasa M itu sendiri dan, dengan demikian, melampaui pengalaman serta produk atau layanan tertentu. Dengan demikian, dengan menguasai M, pengguna memperoleh kemampuan untuk membuat proses transformasi data yang dapat digunakan kembali dan meningkatkan kemampuan manipulasi data mereka di berbagai skenario terkait data, serta di semua pengalaman, produk, atau layanan yang menggunakan M sebagai lapisan transformasi data yang mendasarinya.

Sekarang mari kita alihkan perhatian kita ke alasan mengapa profesional data dan individu lain mungkin ingin menambahkan M ke dalam repertoar bahasa mereka.

E. Mengapa Belajar M?

Dalam dunia yang digerakkan oleh data saat ini, mengubah dan menganalisis data secara efisien dan efektif merupakan keterampilan yang berharga. Power Query, alat transformasi dan penyiapan data yang canggih, memperoleh popularitas luar biasa karena integrasinya yang lancar dengan banyak sistem perangkat lunak populer serta kemudahan penggunaannya. Inti dari Power Query adalah M, Bahasa Rumus Power Query. Namun, Anda mungkin bertanya pada diri sendiri, mengapa Anda harus meluangkan waktu untuk mempelajari M?

Berikut adalah tujuh alasan mengapa kita yakin para profesional data dan individu lain harus mempelajari M:

- **Memanfaatkan sepenuhnya kekuatan Power Query**: Dalam buku Gil Raviv, Gollect, Gombine, and Transform Data Using Power Query in Excel and Power BI, Tn. Raviv memperkirakan bahwa GUI untuk membuat kueri M (lihat Di mana dan kapan M digunakan? dalam bab ini) memungkinkan Anda untuk memecahkan hanya 40% tantangan yang terkait dengan transformasi data, tetapi penguasaan M memungkinkan Anda untuk membawa angka tersebut mendekati 99,99%. Bab-bab selanjutnya

menunjukkan contoh-contoh spesifik dari pemecahan tantangan transformasi data yang tidak dapat dilakukan di GUI. Karena M berfungsi sebagai tulang punggung kemampuan transformasi data Power Query, dengan menguasai M, Anda memperoleh kendali penuh atas proses transformasi data, yang memungkinkan Anda untuk mengekstrak, membersihkan, mengubah, dan membentuk ulang data dari berbagai sumber.

- **Otomatisasi tugas-tugas berulang**: Salah satu alasan utama Anda harus mempelajari M adalah untuk mengotomatisasi tugas-tugas transformasi data yang berulang. Profesional bisnis dan TI sering kali ditugaskan untuk menerima data secara berulang dan kemudian membuat laporan berdasarkan data ini. Daripada mentransformasi data ini secara manual setiap kali (sering kali di Excel) untuk menyiapkannya untuk tujuan pelaporan, memanfaatkan M untuk transformasi data ini memungkinkan logika transformasi data diimplementasikan sekali dan kemudian berjalan secara otomatis setiap kali data baru diterima.
- **Fleksibilitas dan kustomisasi**: Meskipun Power Query menyediakan antarmuka yang mudah digunakan untuk tugas transformasi data, namun ada keterbatasannya. Dengan mempelajari M, Anda dapat memperluas kapabilitas Power Query dan mengatasi keterbatasan ini. M memungkinkan Anda menulis fungsi kustom, melakukan transformasi tingkat lanjut, dan menerapkan logika kompleks yang melampaui kapabilitas bawaan antarmuka Power Query. Fleksibilitas ini memberdayakan Anda untuk menyesuaikan transformasi data secara tepat untuk memenuhi persyaratan unik sumber data dan analisis Anda.
- **Optimalisasi efisiensi dan kinerja**: M adalah bahasa yang sangat efisien dan optimal untuk transformasi data. Mesin Power Query memproses ekspresi M secara cerdas, mengoptimalkan kinerja dengan mengurangi beban dan transformasi data yang tidak perlu. Saat bekerja dengan

kumpulan data besar atau transformasi kompleks, mengetahui M memungkinkan Anda menulis kode efisien yang secara signifikan mempercepat pemrosesan data Anda, seperti yang ditunjukkan dalam Bab 15, Mengoptimalkan Kinerja. Dengan memahami prinsip dasar M dan pertimbangan kinerjanya, Anda dapat mengoptimalkan alur kerja data dan menghemat waktu yang berharga. Terakhir, memanfaatkan M dapat sangat mengurangi dan menyederhanakan rumus dan kode Data Analysis Expressions (DAX) dalam aplikasi hilir seperti Excel dan Power BI Desktop.

- **Pembersihan dan transformasi data tingkat lanjut**: M menyediakan serangkaian fungsi pembersihan dan transformasi data yang komprehensif yang jauh melampaui operasi dasar yang tersedia dalam aplikasi spreadsheet tradisional. Dengan M, Anda dapat dengan mudah menangani masalah kualitas data, seperti menghapus duplikat, menangani nilai yang hilang, membagi kolom, menggabungkan set data, dan melakukan perhitungan tingkat lanjut. Mempelajari M memungkinkan Anda untuk menangani tugas pembersihan dan transformasi data yang rumit secara efisien, yang mengarah pada analisis data yang akurat dan andal.
- **Integrasi dengan bahasa pemrograman lain**: M bukan hanya bahasa yang berdiri sendiri tetapi juga terintegrasi dengan baik dengan bahasa pemrograman lain seperti SQL, R, dan Python. Integrasi ini memungkinkan Anda untuk memanfaatkan kemampuan bahasa-bahasa ini dalam alur kerja Power Query Anda. Anda dapat menggabungkan kode M dengan kueri SQL asli, memanggil skrip R atau Python, dan dengan mudah memasukkan pustaka dan fungsi eksternal perusahaan ke dalam proses transformasi data Anda. Dengan memperluas pengetahuan Anda untuk menyertakan M, Anda membuka potensi untuk memanfaatkan fitur-fitur terbaik dari berbagai bahasa pemrograman untuk manipulasi data.

- **Kemajuan karier**: Kemahiran dalam M dan Power Query telah menjadi keterampilan yang dicari dalam industri data. Karena organisasi semakin bergantung pada data untuk pengambilan keputusan, individu yang memiliki kemampuan untuk mengubah, membersihkan, dan menganalisis data secara efisien sangat dibutuhkan. Dengan menginvestasikan waktu dan upaya dalam mempelajari M, Anda memposisikan diri Anda sebagai aset berharga bagi organisasi yang bergantung pada wawasan berbasis data. Pengetahuan tentang M dapat membuka peluang karier baru, meningkatkan prospek pekerjaan Anda, dan memungkinkan Anda untuk mengerjakan proyek terkait data yang menantang.

Singkatnya, mempelajari M memungkinkan Anda untuk mencerna, mengubah, dan menganalisis data dari berbagai sumber secara efisien. Ini memberikan fleksibilitas, kustomisasi, dan kemampuan pengoptimalan kinerja yang memperluas fungsionalitas Power Query itu sendiri. Dengan menguasai M, Anda memperoleh keunggulan kompetitif dalam industri data dan membuka pintu menuju kemungkinan karier baru.

Kita harap Anda sekarang bersemangat untuk mempelajari M! Mari kita alihkan perhatian kita ke dasar-dasar bahasa M.

F. Dasar-Dasar Dahasa M

Seperti yang telah disebutkan sebelumnya, M adalah bahasa yang kuat yang dirancang untuk penyerapan dan transformasi data dalam berbagai perangkat lunak dan layanan Microsoft. Memahami dasar-dasar bahasa M sangat penting untuk memanfaatkan kemampuannya secara efektif. Berikut ini adalah beberapa dasar penting mengenai bahasa M:

- Ekspresi dan fungsi: Dalam M, ekspresi membentuk blok penyusun transformasi data. Ekspresi merepresentasikan komputasi atau operasi yang dievaluasi menjadi suatu nilai. M menyediakan berbagai fungsi bawaan yang dapat digunakan untuk melakukan operasi pada data. Fungsi dalam M

dipanggil menggunakan sintaksis di mana nama fungsi diikuti oleh argumen dalam tanda kurung. Misalnya, fungsi Text.Start("Hello, World!", 5) mengembalikan substring Hello dari teks input. Informasi lebih lanjut tentang ekspresi dan fungsi dibahas dalam BAB 4, Memahami Nilai dan Ekspresi, serta BAB 9, Parameter dan Fungsi Khusus.

- **Tipe data**: M mendukung berbagai tipe data, termasuk teks, angka, tanggal, waktu, list, tabel, dan record. Memahami tipe data dalam M sangat penting untuk melakukan transformasi yang akurat. M menyediakan fungsi untuk mengonversi antara berbagai tipe data dan memanipulasi data, berdasarkan karakteristik inherennya. Misalnya, fungsi Text.From mengonversi nilai menjadi teks, sedangkan fungsi Date.Year mengekstrak komponen tahun dari nilai tanggal atau datetime. Tipe data dibahas dalam Bab 5, Memahami Tipe Data.
- **Variabel dan konstanta**: M memungkinkan Anda untuk menentukan variabel dan konstanta untuk menyimpan dan menggunakan kembali nilai selama transformasi data. Variabel dibuat dalam ekspresi let, diikuti oleh list penugasan variabel yang dipisahkan koma. Di sisi lain, konstanta adalah nilai tetap yang tetap konstan selama eksekusi. Variabel dan konstanta membantu meningkatkan keterbacaan kode, memungkinkan penggunaan kembali, dan membuat transformasi kompleks lebih mudah dikelola. Informasi lebih lanjut tentang variabel dan konstanta dapat ditemukan di seluruh buku ini.
- **Operator**: M mendukung berbagai operator untuk melakukan kalkulasi matematika, perbandingan logis, dan manipulasi teks. Operator aritmatika (+, -, *, /, dan seterusnya) digunakan untuk kalkulasi numerik, sementara operator perbandingan (>, <, =, dan seterusnya) mengevaluasi kondisi logis. Operator kombinasi ‘&' digunakan untuk menggabungkan nilai teks, menambahkan list dan tabel, atau menggabungkan record. Operator dibahas dalam Bab 4, Memahami Nilai dan Ekspresi.

- **Proses transformasi langkah demi langkah**: M mengikuti proses transformasi langkah demi langkah, di mana setiap langkah mendefinisikan operasi transformasi data. Editor Power Query menyediakan antarmuka visual untuk mendefinisikan langkah-langkah ini dan menghasilkan kode M yang sesuai. Langkah-langkah dapat mencakup operasi seperti memfilter baris, menghapus duplikat, membagi kolom, menggabungkan tabel, dan menggabungkan data. Bab 2, Bekerja dengan Power Query/M, membahas topik ini secara lebih terperinci.
- **Pelipatan kueri**: Pelipatan kueri adalah teknik pengoptimalan dalam Power Query yang mendorong transformasi data ke sumber data bila memungkinkan. Saat menggunakan M, penting untuk menyadari pelipatan kueri guna memastikan pemrosesan data yang efisien. Pelipatan kueri dapat meningkatkan kinerja dengan mengurangi transfer data antara sumber data dan Power Query. Namun, tidak semua transformasi dapat dilipat, jadi penting untuk memahami operasi mana yang dapat dilipat dan yang tidak. Misalnya, saat menggunakan mode Direct Query atau penyimpanan Ganda untuk tabel, semua kueri M harus dilipat, yang dapat membatasi operasi transformasi tertentu. Pelipatan kueri dibahas dalam Bab 7, Gonceptualizing M, dan dalam Bab 15, Optimizing Performance.
- **Penanganan dan debugging kesalahan**: M menyediakan mekanisme penanganan kesalahan untuk menangkap dan menangani pengecualian selama transformasi data. Dengan menggunakan fungsi seperti try, otherwise, dan error, Anda dapat mengendalikan aliran eksekusi dan menangani potensi kesalahan dengan baik. Selain itu, M mendukung kemampuan debugging, seperti kemampuan untuk menelusuri kode guna mengidentifikasi dan menyelesaikan masalah dalam transformasi yang kompleks. Penanganan dan debugging kesalahan dibahas dalam Bab 12, Penanganan Kesalahan dan Debugging.

- **Sensitivitas huruf besar/kecil**: M peka huruf besar/kecil. Ini berlaku untuk semua fungsi, ekspresi, variabel, konstanta, dan aspek lain dari bahasa M.
- **Komentar**: Komentar dalam M mengikuti gaya komentar bahasa C. Komentar sebaris diawali dengan garis miring ganda (//) sementara komentar blok menggunakan pola garis miring-tanda bintang/tanda bintang-garis miring (/* dan */).

Sekarang setelah kita memiliki pemahaman yang baik tentang komponen inti bahasa M, selanjutnya mari kita jelajahi komponen paling mendasar dari bahasa M, ekspresi let.

1. Ekspresi *Let*

Inti dari bahasa M adalah ekspresi *let*, yang harus dipasangkan dengan ekspresi *in*. Secara sederhana, ekspresi *let* berisi input dan transformasi, sedangkan ekspresi *in* berisi output. *Hello World* sederhana untuk M terlihat seperti berikut:

```
let
    Hello = "Hello World"
in
    Hello
```

Kode ini akan mengembalikan teks Hello World yang ada di mana-mana. Penting untuk dicatat bahwa setiap ekspresi dalam pernyataan let harus diikuti oleh koma (,) kecuali ekspresi terakhir sebelum ekspresi in. Jadi, jika ekspresi let terdiri dari beberapa sub-ekspresi, maka kodenya mungkin terlihat seperti berikut:

```
let

    Hello = "Hello",
    World = "World",

    Return = Hello & " " & World

in

    Return
```

Kode ini juga menampilkan Hello World sebagai output. Memahami dasar-dasar M, termasuk ekspresi, fungsi, tipe data, variabel, operator, dan proses transformasi langkah demi langkah, sangat penting untuk memanipulasi dan menyiapkan data secara efektif. Dengan menguasai konsep-konsep dasar ini, Anda memperoleh kemampuan untuk melakukan transformasi kompleks, mengoptimalkan alur kerja data, dan membuka potensi penuh bahasa M. Sisa buku ini ditujukan untuk membantu Anda menguasai semua konsep dasar ini dan cara menerapkannya pada transformasi data yang kompleks.

G. Ciri-ciri M

M adalah bahasa pemrograman yang berfungsi sebagai tulang punggung Power Query, yang memungkinkan pengguna untuk mengekstrak, membersihkan, dan membentuk ulang data dari berbagai sumber, seperti basis data, lembar kerja, halaman web, dan banyak lagi. Namun, tidak seperti bahasa pemrograman tujuan umum seperti C, C#, Java, dan Python, yang dirancang untuk berbagai macam aplikasi, M adalah bahasa khusus domain, yang secara khusus dirancang untuk penyerapan dan manipulasi data. Dengan demikian, M menyediakan serangkaian fungsi, operator, dan ekspresi yang memungkinkan Anda untuk melakukan transformasi, kalkulasi, dan agregasi data yang kompleks. Mari kita pahami hal ini dengan lebih baik dengan melihat karakteristik M dari perspektif formal dan informal.

1. Klasifikasi Formal

Bahasa pemrograman diklasifikasikan menurut sejumlah properti, seperti murni/tidak murni, orde rendah/orde tinggi, diketik secara statis/dinamis, diketik dengan kuat/lemah, evaluasi bersemangat/malas, dan imperatif/fungsional. Microsoft telah mendeskripsikan M sebagai:

- **Sebagian besar murni**: Bahasa pemrograman dikatakan murni jika menyediakan integritas referensial. Dengan kata lain, ekspresi apa pun

dapat diganti dengan nilai ekspresi tersebut tanpa mengubah perilaku atau makna program.

Bahasa pemrograman yang tidak murni memungkinkan efek samping, yang merupakan tindakan yang menyebabkan perubahan di luar cakupan nilai pengembalian fungsi. Dalam kasus M, ini umumnya digunakan untuk tugas transformasi dan pengambilan data, yang sering kali melibatkan interaksi dengan sumber data eksternal, melakukan operasi pada data, dan menghasilkan output. Tindakan ini merupakan efek samping karena memengaruhi status sumber data atau menghasilkan output di luar nilai pengembalian fungsi.

Sementara M menyediakan konstruksi pemrograman fungsional dan mendukung kekekalan, yang memungkinkan terciptanya fungsi murni, bahasa tersebut tidak sepenuhnya fungsional karena sifatnya yang tidak murni. Bahasa tersebut mencakup kombinasi paradigma pemrograman fungsional dan imperatif untuk memfasilitasi manipulasi dan pengambilan data yang efisien dan praktis.

- **Tingkat tinggi**: Untuk bahasa tingkat rendah, seperti kode mesin atau bahasa assembly, setiap pernyataan pemrograman sesuai dengan satu instruksi untuk komputer, sedangkan dalam bahasa tingkat tinggi, setiap pernyataan sesuai dengan beberapa instruksi untuk komputer.

  Bahasa tingkat tinggi biasanya memungkinkan hal-hal seperti fungsi, objek, dan modul untuk digunakan sebagai nilai dalam suatu program. Bahasa tingkat tinggi sering memperlakukan fungsi sebagai warga negara kelas satu. Secara khusus, ini berarti bahwa fungsi dapat ditetapkan ke variabel, diteruskan sebagai argumen ke fungsi lain, dan dikembalikan sebagai nilai dari fungsi.

  Power Query M mendukung pemrograman tingkat tinggi dengan memungkinkan Anda untuk menentukan dan memanipulasi fungsi sebagai nilai. Anda dapat menetapkan fungsi ke variabel, meneruskan

fungsi sebagai argumen ke fungsi lain, dan mengembalikan fungsi sebagai hasil. Hal ini memungkinkan Anda membuat kode yang lebih modular dan fleksibel dengan mengabstraksi dan menggunakan kembali logika fungsi. Dengan kemampuan pemrograman tingkat tinggi, M memungkinkan Anda menerapkan transformasi dan perhitungan secara dinamis berdasarkan parameter input, aliran kontrol, dan karakteristik data. Anda dapat menulis fungsi yang beroperasi pada fungsi lain, yang memungkinkan manipulasi data dan skenario transformasi yang canggih. Misalnya, Anda dapat menggunakan fungsi tingkat tinggi di M untuk menerapkan serangkaian transformasi secara dinamis ke kumpulan data, berdasarkan kriteria yang ditentukan pengguna, atau untuk membuat alur fungsi yang dapat digunakan kembali untuk pemrosesan data.

Dengan menyediakan fitur pemrograman tingkat tinggi, M memberdayakan pengembang untuk menulis kode ekspresif dan modular, sehingga memudahkan untuk bekerja dengan transformasi data yang kompleks dan menyesuaikan perilaku fungsi agar sesuai dengan persyaratan tertentu.

- **Diketik secara dinamis**: Bahasa yang diketik secara dinamis melakukan pemeriksaan tipe pada waktu proses alih-alih pada waktu kompilasi, seperti halnya dengan bahasa yang diketik secara statis. Pemeriksaan tipe hanyalah proses untuk memastikan bahwa hal-hal seperti parameter yang diteruskan ke fungsi memiliki tipe yang benar, seperti teks, angka, atau tanggal.
- **Diketik secara lemah**: Meskipun terkait erat dengan properti yang diketik secara statis/dinamis, diketik secara kuat dan diketik secara lemah merujuk pada sesuatu yang sangat berbeda. Bahasa yang diketik secara kuat sangat sensitif terhadap kompatibilitas tipe dan memerlukan definisi tipe eksplisit untuk variabel sebelum digunakan. Sebaliknya, bahasa

dengan tipe lemah seperti M tidak memerlukan definisi tipe eksplisit, dan beberapa bahkan melakukan konversi tipe otomatis.

M tidak memiliki tipe lemah seperti bahasa pemrograman seperti Python, karena variabel tidak dapat diubah setelah dihitung. Jadi, definisi tipe lemah untuk M umumnya mengacu pada kemampuan untuk menggunakan variabel yang tipe datanya belum ditentukan secara eksplisit.

Pertimbangkan kode Python berikut:

```
a = 42

a = "Hello World"
print(a)
```

Kode ini tidak akan menghasilkan kesalahan dalam Python, meskipun dua tipe data berbeda ditetapkan ke variabel a. Namun, kode serupa tidak mungkin dalam M, karena variabel, setelah dihitung, tidak dapat diubah (yaitu, tidak dapat diubah).

Fleksibilitas dalam pengetikan data memungkinkan Power Query M untuk menangani berbagai sumber data dan melakukan berbagai transformasi data secara efektif. Ini menyederhanakan proses bekerja dengan kumpulan data heterogen yang mungkin berisi tipe dan struktur data yang berbeda.

Penting untuk dicatat bahwa meskipun Power Query M diketik dengan lemah, ia masih melakukan pemeriksaan tipe selama eksekusi (waktu proses) untuk memastikan konsistensi operasi. Jika operasi tertentu tidak kompatibel dengan tipe nilai yang disimpulkan, kesalahan dapat terjadi saat waktu proses.

Secara keseluruhan, sifat Power Query M yang diketik dengan lemah mencapai keseimbangan antara fleksibilitas dan integritas data, menyediakan pengguna dengan bahasa yang serbaguna untuk tugas transformasi data. • Sebagian malas: Secara umum, M mengikuti strategi evaluasi yang bersemangat, yang berarti bahwa saat Anda menentukan transformasi atau perhitungan, transformasi dan perhitungan tersebut

dilakukan segera setelah Anda menerapkannya pada data. Pendekatan evaluasi yang bersemangat ini memastikan bahwa transformasi data terjadi dengan segera dan hasilnya tersedia untuk pemrosesan atau analisis lebih lanjut. Power Query M dirancang untuk menangani tugas manipulasi dan pengambilan data secara efisien, dengan fokus pada evaluasi langsung untuk memberikan umpan balik waktu nyata pada transformasi.

Istilah lazy parsial merujuk pada fitur tertentu dalam M yang disebut evaluasi lazy, yang berbeda dari strategi evaluasi bahasa secara keseluruhan. Dalam M, evaluasi lazy diterapkan pada ekspresi dalam beberapa konstruksi tertentu, khususnya ekspresi List, Record, dan Table serta ekspresi let. Konstruksi ini memungkinkan Anda untuk mendefinisikan ekspresi yang dievaluasi hanya saat dibutuhkan, menyediakan bentuk evaluasi sesuai permintaan atau lazy dalam konteks tersebut.

M juga memungkinkan Anda untuk mendefinisikan argumen opsional untuk fungsi. Argumen opsional ini dievaluasi secara lazy, artinya argumen tersebut tidak dihitung kecuali digunakan secara eksplisit dalam badan fungsi. Evaluasi lazy argumen opsional membantu mengoptimalkan kinerja dengan menghindari perhitungan yang tidak perlu untuk nilai opsional yang sebenarnya tidak digunakan dalam fungsi. Evaluasi lazy memastikan bahwa perhitungan untuk argumen opsional ditangguhkan hingga nilainya diperlukan dalam eksekusi fungsi.

M juga mendukung konstruksi percabangan bersyarat seperti pernyataan if-then-else. Hanya cabang yang cocok dengan suatu kondisi yang dievaluasi, sedangkan cabang lainnya tidak dihitung, sehingga menghasilkan evaluasi lazy. Hal ini berbeda dengan strategi evaluasi yang bersemangat, di mana kedua cabang dievaluasi tanpa mempedulikan hasil kondisi. Bentuk evaluasi malas dalam konstruksi if-then-else ini

memungkinkan komputasi yang efisien dengan menghindari evaluasi ekspresi yang tidak perlu di cabang yang tidak cocok.

Penting untuk dicatat bahwa meskipun M memiliki fitur-fitur yang sebagian malas ini, strategi evaluasi bahasa secara keseluruhan tetap dominan bersemangat. Sebagian besar ekspresi dalam Power Query M dievaluasi dengan bersemangat, memastikan bahwa transformasi data terjadi dengan segera, dan hasilnya segera tersedia untuk pemrosesan lebih lanjut. Dengan demikian, M pada dasarnya adalah bahasa pemrograman yang bersemangat, tetapi menggabungkan evaluasi malas sebagian dalam konstruksi tertentu, seperti untuk ekspresi List, Record, Table, dan let, serta argumen fungsi opsional dan percabangan bersyarat. Evaluasi yang sebagian malas ini menawarkan fleksibilitas dan mengoptimalkan kinerja dalam konteks tersebut.

- Fungsional: M menggabungkan banyak konsep dan fitur pemrograman fungsional, menjadikannya bahasa pemrograman fungsional. Konsep dan fitur fungsional ini meliputi kekekalan, fungsi tingkat tinggi, komposisi fungsi, fungsi murni, dan rekursi.

  M mendorong kekekalan, yang berarti bahwa nilai data tidak dimodifikasi di tempat tetapi diubah menjadi nilai baru. Ini mempromosikan prinsip pemrograman fungsional untuk menghindari efek samping.

  M mendukung fungsi tingkat tinggi, yang memungkinkan fungsi diperlakukan sebagai nilai kelas satu. Anda dapat meneruskan fungsi sebagai argumen ke fungsi lain, mengembalikan fungsi dari fungsi, dan menyimpan fungsi dalam variabel.

  M memfasilitasi komposisi fungsi, memungkinkan Anda menggabungkan beberapa fungsi untuk membuat transformasi yang lebih kompleks. Komposabilitas ini merupakan karakteristik pemrograman fungsional.

  M mendukung penggunaan fungsi murni, yang tidak memiliki efek samping dan menghasilkan keluaran yang sama untuk masukan yang

sama. Fungsi murni membuat kode lebih mudah diprediksi dan lebih mudah dipecahkan masalahnya.

M mendukung rekursi. Meskipun rekursi tidak didukung secara luas seperti dalam beberapa bahasa fungsional lainnya, M menawarkan dukungan terbatas untuk fungsi rekursif, yang memungkinkan pengembang untuk memecahkan masalah melalui teknik rekursif.

Dalam hal perbandingan dengan bahasa lain, M mungkin paling mirip dengan *F#*, bahasa pemrograman yang dikembangkan dan diimplementasikan oleh Don Syme dari *Microsoft Research*, Cambridge, Inggris.

Sekarang setelah kita membahas klasifikasi formal bahasa M, selanjutnya mari kita lihat beberapa karakteristik informal M.

2. Karakteristik Informal M

Secara informal, berikut ini adalah beberapa karakteristik dan fitur utama M:

- **Bahasa fungsional**: M adalah bahasa fungsional, artinya bahasa ini didasarkan pada konsep fungsi sebagai blok penyusun utama untuk transformasi data. Fungsi dalam M dapat digabungkan, disusun, dan dikomposisi untuk melakukan manipulasi data yang rumit. M menyediakan lebih dari 700 fungsi bawaan untuk operasi umum, serta kemampuan untuk membuat fungsi khusus yang disesuaikan dengan kebutuhan spesifik Anda. Banyak dari fungsi ini, serta fungsi khusus, dibahas dalam bab-bab selanjutnya.

  Kumpulan fungsi bawaan yang luas dalam M menyediakan alat yang hebat bagi pengguna untuk menangani berbagai skenario transformasi data. Fungsi-fungsi ini dirancang untuk menyederhanakan tugas-tugas manipulasi data umum dan memungkinkan pengguna untuk mengubah dan membentuk data mereka secara efisien.

- **Sintaks yang ekspresif dan mudah dibaca**: Sintaks M dirancang agar intuitif dan mudah dibaca, sehingga dapat diakses oleh programmer pemula dan berpengalaman. Ekspresi M ditulis dengan cara yang jelas dan ringkas, sehingga memudahkan pembuatan transformasi data yang kompleks tanpa mengorbankan keterbacaan. Sintaks mengikuti pendekatan langkah demi langkah, yang memungkinkan Anda untuk menentukan serangkaian transformasi berurutan yang akan diterapkan pada data Anda. Bab-bab selanjutnya menunjukkan sintaksis M yang ekspresif dan mudah dibaca dengan contoh-contoh spesifik.
- **Tipe dan nilai data**: M mendukung berbagai tipe data, termasuk tipe data primitif seperti teks, angka, tanggal, dan durasi, serta tipe data terstruktur seperti list, tabel, dan record. M menyediakan fungsi-fungsi hebat untuk bekerja dengan tipe-tipe data ini, membantu Anda untuk memanipulasi dan mengubah data pada tingkat yang terperinci. M juga memungkinkan Anda untuk menentukan dan bekerja dengan variabel, konstanta, dan parameter untuk menyimpan dan menggunakan kembali hasil sementara selama proses transformasi data. Tipe data dan nilai dibahas secara mendalam di Bab 4, Memahami Nilai dan Ekspresi, dan Bab 5, Memahami Tipe Data.
- **Integrasi dengan Power Query Editor**: M terintegrasi dengan lancar dengan Power Query Editor, menyediakan antarmuka yang mudah digunakan untuk berinteraksi dengan dan mengembangkan kode M. Power Query Editor memungkinkan Anda untuk membuat langkah-langkah transformasi data secara visual, melihat pratinjau hasilnya, dan membuat kode M secara otomatis. Ini menyediakan lingkungan pengembangan yang tangguh tempat Anda dapat menulis, men-debug, dan menyempurnakan ekspresi M Anda. Lihat Bab 2, Bekerja dengan Power Query/M, untuk informasi lebih lanjut tentang subjek ini.

- **Ekstensibilitas dan kustomisasi**: Salah satu fitur menonjol dari M adalah ekstensibilitasnya. Sementara M menawarkan berbagai transformasi bawaan, M juga memungkinkan Anda untuk melampaui kemampuan ini dan membuat transformasi kustom yang sesuai dengan kebutuhan spesifik Anda sendiri. Anda dapat menentukan fungsi Anda sendiri, menulis potongan kode yang dapat digunakan kembali, dan membuat logika manipulasi data tingkat lanjut menggunakan M. Tingkat kustomisasi ini memberdayakan Anda untuk menangani skenario data kompleks yang tidak tercakup oleh transformasi standar yang dapat diakses melalui antarmuka pengguna. Gbapter 16, Mengaktifkan Ekstensi, menunjukkan fleksibilitas dan ekstensibilitas M.
- **Optimalisasi kinerja**: M dioptimalkan untuk kinerja, yang memungkinkan pemrosesan data yang efisien, bahkan dengan kumpulan data yang besar. Mesin Power Query secara cerdas mengevaluasi dan mengoptimalkan ekspresi M untuk meminimalkan pemuatan dan transformasi data, sehingga menghasilkan pemrosesan data yang lebih cepat dan lebih efisien.

  Satu teknik pengoptimalan kinerja yang spesifik disebut streaming secara semantik dan merupakan properti dari ekspresi List dan Tabel. Streaming secara semantik melibatkan penghitungan ulang baris tabel atau item list. Alih-alih mengulangi tabel atau list untuk setiap transformasi data, streaming secara semantik setiap baris tabel atau item dalam list dievaluasi untuk semua transformasi data, dan hasilnya dikumpulkan sebagai bagian dari keluaran untuk ekspresi tersebut. Streaming secara semantik memungkinkan transformasi kumpulan data yang tidak sesuai dengan memori yang tersedia.

  Teknik pengoptimalan kinerja lainnya disebut pelipatan kueri. Akan tetapi, pelipatan kueri bukanlah properti dari bahasa M itu sendiri. Sebaliknya, pelipatan kueri digunakan dalam Power Query untuk **push**

atau **fold** transformasi data kembali ke sistem data sumber. Intinya, ekspresi transformasi dalam M diterjemahkan ke pernyataan transformasi ekuivalen yang tersedia dalam sistem sumber, seperti SQL Server. Ini mendorong pemrosesan transformasi kembali ke sistem sumber, bukan sistem klien yang mengeksekusi kueri M. Ini dapat meningkatkan kinerja dan efisiensi dengan meminimalkan transfer data dan mengurangi jumlah data yang diproses oleh Power Query.

Dengan memahami prinsip dasar M dan pertimbangan kinerjanya, Anda dapat menulis kode yang dioptimalkan dan meningkatkan kinerja keseluruhan alur kerja data Anda. Bab 15, Mengoptimalkan Kinerja, menyediakan informasi lebih lanjut tentang pengoptimalan kinerja, dengan contoh-contoh spesifik.

Singkatnya, M adalah bahasa yang serbaguna dan ekspresif yang dirancang khusus untuk transformasi dan manipulasi data dalam Power Query. Sifat fungsionalnya, rangkaian fungsi yang luas, dan integrasi dengan editor Power Query menjadikannya alat yang hebat untuk mengekstraksi, membersihkan, dan membentuk ulang data dari berbagai sumber.

H. Ringkasan

Bahasa M adalah tulang punggung kemampuan transformasi data Microsoft Power Query. M digunakan secara luas dalam ekosistem Microsoft yang lebih besar, termasuk berbagai aplikasi perangkat lunak dan layanan berbasis web. Banyak jenis profesional data yang dapat memperoleh manfaat dari mempelajari M untuk menyerap dan mengubah data secara efisien dan efektif dalam berbagai skenario. Dengan menguasai M, Anda memperoleh kemampuan untuk menangani transformasi data yang kompleks secara efisien, menyesuaikan alur kerja data Anda, dan mengoptimalkan kinerja, sehingga memanfaatkan potensi penuh yang ditawarkan Power Query untuk kebutuhan manipulasi data Anda.

Dalam bab ini, kita telah membahas pertanyaan penting tentang siapa, apa, di mana, mengapa, dan bagaimana M digunakan, serta memberikan pengantar singkat tentang aspek fundamental bahasa M. Dalam bab berikutnya, kita memperkenalkan Anda pada antarmuka utama tempat Anda akan menulis dan bekerja dengan M.

I. Daftar Pustaka

1. Kasun Bandara and Rob J Hyndman and Christoph Bergmeir. (2021). MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns. arXiv:2107.13462 [stat.AP]. https://arxiv.org/abs/2107.13462.
2. Hochenbaum, J., Vallis, O., & Kejariwal, A. (2017). Automatic Anomaly Detection in the Cloud Via Statistical Learning. ArXiv, abs/1704.07706. https://arxiv.org/abs/1704.07706.

J. Penutup

1. Tes Formatif

Tabel 1. 1 Tes Formatif Bab 1

No	Soal	Bobot
1.	Jelaskan apa yang dimaksud dengan Power Query dan sebutkan fungsinya dalam pengolahan data.	10
2.	Apa saja kelebihan yang ditawarkan oleh Power Query dibandingkan dengan alat pengolahan data lainnya? Berikan contoh konkret.	10
3.	Deskripsikan langkah-langkah yang terlibat dalam proses pengambilan data menggunakan Power Query.	10
4.	Apa yang dimaksud dengan transformasi data dalam konteks Power Query? Berikan contoh jenis transformasi yang dapat dilakukan.	10

5.	Sebutkan berbagai sumber data yang dapat diakses melalui Power Query dan jelaskan bagaimana cara menghubungkannya.	10
6.	Uraikan antarmuka pengguna Power Query. Apa saja komponen utama yang ada di dalamnya?	10
7.	Apa itu M Language dalam Power Query? Jelaskan perannya dalam pengolahan data.	10
8.	Diskusikan bagaimana Power Query dapat diterapkan dalam konteks bisnis untuk meningkatkan efisiensi pengolahan data.	10
9.	Identifikasi beberapa kesalahan umum yang sering dilakukan pengguna saat menggunakan Power Query dan cara menghindarinya.	10
10.	Berikan contoh studi kasus di mana Power Query berhasil digunakan untuk menyelesaikan masalah pengolahan data. Apa hasil yang diperoleh?	10

BAB 2

Bekerja dengan Power Query/M

Topik Bab

Pada bab ini, topik yang akan dibahas adalah:

- Menjelajahi pengalaman Power Query Desktop
- Mengubah kode yang dihasilkan berdasarkan pengalaman
- Membuat kolom khusus
- Menggunakan Editor Lanjutan

A. Pendahuluan

Dalam Bab 1, Memperkenalkan M, kita memperkenalkan aspek fundamental bahasa M. Di bagian Di mana M digunakan?, kita memperkenalkan konsep pengalaman Power Query secara singkat, termasuk Power Query Desktop dan Power Query Online.

Bab ini memberikan detail lebih lanjut mengenai cara bekerja dengan M dalam pengalaman Power Query Desktop menggunakan Power BI Desktop. Pengetahuan yang dibahas di sini akan memungkinkan Anda untuk mulai menulis kode M melalui **Graphical User Interface (GUI)** yang mudah digunakan yang disediakan oleh Power Query Editor dan dapat langsung diterjemahkan ke pengalaman desktop lainnya, seperti yang ditemukan di Excel serta pengalaman Power Query Online dalam situs web seperti powerbi.com. Sebenarnya, pengalaman Power Query Desktop memungkinkan Anda membuat kode M tanpa benar-benar menulis kode itu sendiri. Dengan demikian, Anda dapat segera mulai menggunakan M untuk membersihkan dan mengubah data bisnis Anda tanpa mengetahui bahasanya.

Karena buku ini terutama difokuskan pada bahasa M, bab ini tidak memberikan cakupan yang komprehensif tentang pengalaman Power Query. Namun, memahami dasar-dasar di mana kode M ditulis adalah penting dan dengan demikian bab ini mencakup topik-topik berikut:

- Menjelajahi pengalaman Power Query Desktop
- Mengubah kode yang dihasilkan berdasarkan pengalaman
- Membuat kolom khusus
- Menggunakan Editor Lanjutan

1. Kasus Pemantik Berpikir Kritis: Pengolahan Data Penjualan Bulanan

Sebuah perusahaan e-commerce mengumpulkan data penjualan bulanan dari berbagai kategori produk. Data ini disimpan dalam beberapa file Excel yang berbeda, masing-masing berisi informasi tentang produk, jumlah terjual, dan pendapatan. Tim analisis data ditugaskan untuk menggunakan Power

Query untuk mengolah dan menganalisis data ini agar dapat memberikan wawasan yang berguna bagi manajemen.

1) Identifikasi Jenis Data:

   Apa saja jenis data yang perlu diambil dari file Excel untuk analisis penjualan? Bagaimana Anda akan memastikan bahwa semua data yang relevan telah diidentifikasi?

2) Proses Pengimporan Data:

   Apa langkah-langkah yang harus diambil untuk mengimpor data dari beberapa file Excel ke dalam Power Query? Apa tantangan yang mungkin Anda hadapi dalam proses ini?

3) Transformasi Data:

   Data yang diimpor mungkin memerlukan transformasi. Sebutkan beberapa transformasi yang mungkin diperlukan dan jelaskan mengapa transformasi tersebut penting untuk analisis.

4) Penggunaan Filter dan Sorting:

   Bagaimana Anda akan menggunakan fitur filter dan sorting di Power Query untuk mendapatkan informasi yang lebih spesifik dari data penjualan? Berikan contoh situasi di mana ini akan sangat berguna.

5) Penggabungan Data:

   Jika data penjualan dari beberapa kategori produk perlu digabungkan, apa langkah-langkah yang harus diambil untuk melakukan penggabungan ini? Apa yang perlu diperhatikan agar penggabungan data berjalan lancar?

6) Analisis Kualitas Data:

   Bagaimana Anda akan mengevaluasi kualitas data yang telah diimpor dan ditransformasi? Apa indikator yang dapat digunakan untuk menilai keakuratan dan konsistensi data?

7) Visualisasi Hasil Analisis:

Setelah data dianalisis, bagaimana Anda akan menyajikan hasil analisis kepada tim manajemen? Apa jenis visualisasi yang paling efektif untuk menyampaikan informasi tentang penjualan?

8) Pengambilan Keputusan Berdasarkan Data:

Berdasarkan hasil analisis yang dilakukan, keputusan apa yang dapat diambil oleh manajemen untuk meningkatkan penjualan? Apa langkah-langkah yang harus diambil untuk menerapkan keputusan tersebut?

B. Persyaratan Teknis

Untuk menyelesaikan tugas dalam bab ini, Anda memerlukan hal berikut:

- Power BI Desktop
- File untuk bab ini dapat diunduh dari repositori GitHub berikut: https://github.com/PacktPublishing/The-Definitive-Guide-to-Power-Query-M-

Power BI Desktop adalah tempat Anda akan diperkenalkan dengan pengalaman Power Query Editor yang memungkinkan Anda mulai mengubah dan membentuk data Anda.

C. Menjelajahi Pengalaman Power Query Desktop

Power BI Desktop dilengkapi dengan subprogram canggih yang disebut editor Power Query. Editor Power Query adalah alat canggih untuk persiapan dan transformasi data, yang memungkinkan pengguna untuk membersihkan, membentuk, dan menggabungkan jutaan atau bahkan miliaran baris data dari berbagai sumber sebelum menggunakannya dalam alat analitik, pelaporan, atau visualisasi. Pembersihan dan transformasi data memainkan peran utama dalam menciptakan model semantik yang kuat dan terstruktur dengan baik untuk analisis dan pelaporan data yang efektif.

Untuk mengakses Editor Power Query, pertama-tama Anda perlu mengunduh dan menginstal Power BI Desktop. Power BI Desktop dapat diinstal dari Microsoft Store. Ini adalah pendekatan yang direkomendasikan, meskipun

Power BI Desktop juga dapat diinstal dari tautan berikut: https://www.microsoft.com/en-us/download/details.aspx?id=58494.

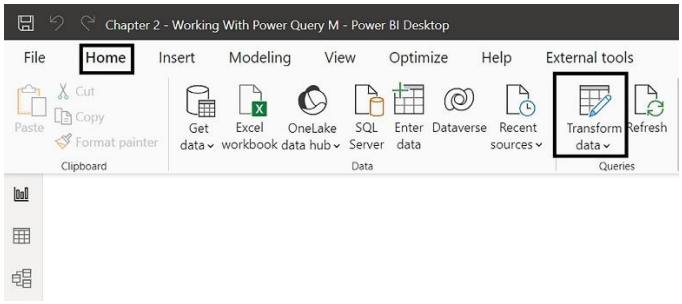
Untuk menginstal Power BI Desktop dari Microsoft Store, lakukan langkah-langkah berikut:

1) Buka aplikasi Microsoft Store di PC Windows Anda dan cari Power BI Desktop, seperti yang ditunjukkan pada Gambar 2.1:

Gambar 2. 1 Aplikasi Power BI Desktop di Microsok Store

2) Pastikan nama aplikasinya adalah Power BI Desktop dan bukan Power BI. Keduanya adalah aplikasi yang berbeda.
3) Jika belum terpasang, tombol **Open** pada Gambar 2.1 akan bertuliskan **Install**. Jika tombolnya sudah bertuliskan **Open**, lanjutkan ke langkah 5.
4) Klik tombol **Install**, lalu setelah terpasang, tombol **Install** akan berubah menjadi **Open**.
5) Klik tombol **Open** untuk meluncurkan Power BI Desktop.

Untuk mengakses Editor Power Query dari Power BI Desktop, pastikan tab **Home** dipilih di pita, lalu klik ikon **Transform data** di pita, seperti yang diperlihatkan pada Gambar 2.2:

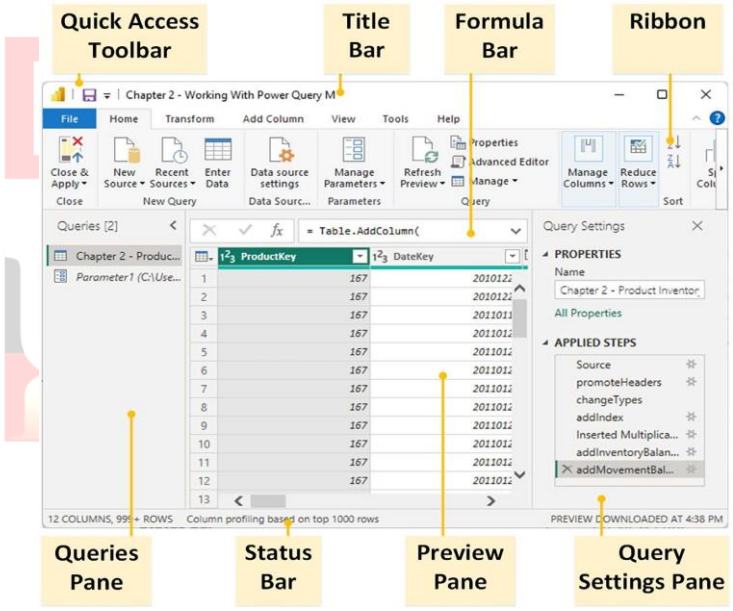


Gambar 2. 2 Mengubah data di Power BI Desktop

Sekarang setelah Power BI Desktop terinstal dan editor Power Query diluncurkan, selanjutnya mari kita lihat sekilas GUI Editor Power Query dan pengaturan terkait.

1. Tur Singkat

Antarmuka Power Query Editor memiliki karakteristik yang serupa dan memiliki elemen yang sama dengan Power BI Desktop. Antarmuka pengguna Power Query Editor terdiri dari tujuh area utama. Lihat Gambar 2.3 saat membaca tentang ketujuh area ini di bagian selanjutnya.



Gambar 2. 3 Pengalaman Editor Power Query di Power BI Desktop

Mari kita bahas secara singkat setiap area utama antarmuka editor Power Query dimulai dengan area header, yang mencakup Title Bar dan Quick Access Toolbar.

Header

Hampir identik dengan Power BI Desktop, header adalah strip kecil di bagian atas jendela Power Query Editor. Area ini standar untuk aplikasi Windows. Klik kiri ikon aplikasi di sudut kiri akan memberikan ukuran standar dan perintah keluar umum, seperti meminimalkan, memaksimalkan, dan menutup.

Di sebelah ikon ini terdapat **Quick Access Toolbar**. Toolbar ini dapat ditampilkan di atas atau di bawah pita, dan perintah di dalam pita dapat ditambahkan ke toolbar ini dengan mengklik kanan ikon di pita dan memilih **Add to Quick Access Toolbar**. Secara default, hanya ikon **Save** yang ditampilkan.

Di sebelah kanan **Quick Access Toolbar** terdapat nama file yang sedang dibuka. Terakhir, di paling kanan terdapat ikon standar meminimalkan, mengembalikan ke bawah/memaksimalkan, dan menutup.

Formula Bar

Bilah Rumus memungkinkan Anda untuk melihat, memasukkan, dan mengubah kode M untuk langkah kueri tertentu. Seperti yang Anda ketahui dari Bab 1, Memperkenalkan M, M adalah bahasa pemrograman fungsional yang terdiri dari fungsi, operator, dan nilai yang digunakan untuk menghubungkan dan mengubah data.

M adalah bahasa di balik kueri di Power BI Desktop. Saat Anda membuat kueri di Editor Power Query menggunakan GUI, di balik layar, ini benar-benar membangun skrip M yang dijalankan untuk menghubungkan, mengubah, dan mengimpor data Anda. Setiap Langkah Terapan (lihat panel **Query Settings** di bawah) dalam kueri sebenarnya adalah baris kode bahasa M. Anda tidak

perlu khawatir tentang itu dulu; kita akan membahasnya sedikit nanti di bab ini.

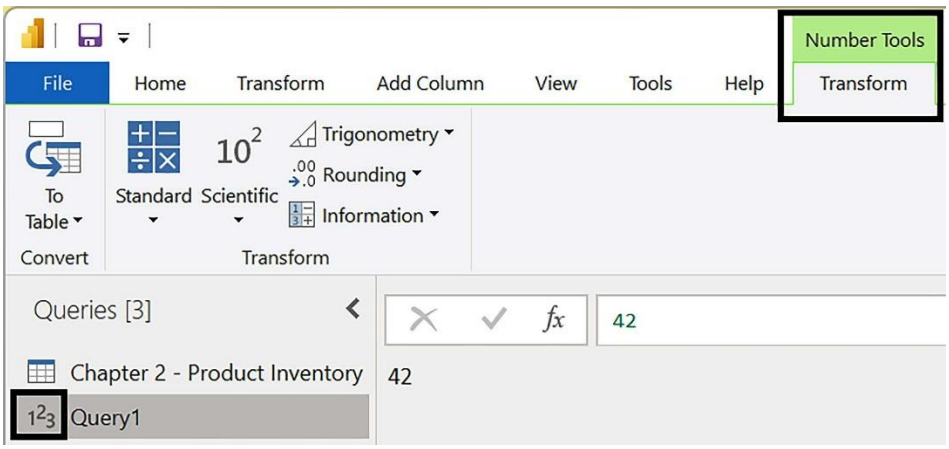
Ribbon

Di bawah Bilah Alat Akses Cepat dan Bilah Judul di header terdapat pita. Jika Anda familier dengan versi Microsoft Office modern, Anda akan mengenali fungsi area ini, meskipun kontrol yang ditampilkan agak berbeda. Pita terdiri dari tujuh tab utama ditambah satu tab kondisional:

1) **File**: Tab File menampilkan menu fly-out saat diklik yang memungkinkan file Power BI Desktop disimpan, serta Editor Power Query ditutup, dan perubahan yang dibuat dalam Editor Power Query diterapkan. Selain itu, menu **File** menyediakan akses ke **Option** dan **Data Source Settings**.
2) **Home**: Tab **Home** menyediakan berbagai operasi yang paling umum, seperti menghubungkan ke sumber, mengelola parameter, dan fungsi transformasi umum seperti menghapus baris dan kolom, membagi kolom, dan mengganti nilai.
3) **Transform**: Tab **Transform** menyediakan fungsi manipulasi data yang memungkinkan transposisi baris dan kolom, pivoting dan unpivoting kolom, penggabungan kolom, penambahan skrip R dan Python, dan banyak kalkulasi ilmiah, terkait statistik, trigonometri, tanggal, dan waktu. Penting untuk dipahami bahwa fungsi pada tab **Transform** memodifikasi kolom yang sudah ada, berbeda dengan fungsi serupa yang ditemukan pada tab **Add Column**, yang malah menambahkan kolom baru dengan transformasi data yang ditentukan.
4) **Add Column**: Tab **Add Column** menyediakan operasi yang difokuskan pada penambahan kolom terhitung, bersyarat, dan indeks. Selain itu, banyak fungsi yang tersedia pada tab **Transform**, seperti perhitungan ilmiah, statistik, trigonometri, tanggal, dan waktu juga tersedia pada tab **Add Column**; namun, penting untuk dipahami bahwa fungsi-fungsi tersebut bekerja sangat berbeda antara dua tab pita. Pada tab Transformasi, fungsi-

fungsi ini mengubah kolom yang saat ini dipilih, sementara pada tab Tambahkan Kolom, fungsi-fungsi ini menambahkan kolom terhitung tambahan ke data.

5) **View**: Tab **View** menyertakan kontrol untuk mengendalikan tata letak antarmuka Power Query, seperti apakah panel **Query Settings** ditampilkan atau tidak, dan apakah **Formula Bar** ditampilkan.
6) **Tools**: Tab **Tools** menyediakan akses ke alat dan opsi diagnostik.
7) **Help**: Tab **Help** menyertakan tautan yang berguna untuk mendapatkan bantuan dengan Power BI, termasuk tautan ke situs Komunitas Power BI, dokumentasi, pembelajaran terpandu, video pelatihan, dan banyak lagi.
8) **Tools | Transform**: **Tools |** Tab **Transform** ditampilkan secara kondisional saat tipe pengembalian kueri bukan tabel. Tab ini menampilkan opsi untuk mengonversi nilai ke tabel, serta opsi kontekstual tergantung pada tipe data yang dikembalikan. Misalnya, ekspresi yang mengembalikan teks akan memiliki opsi yang berbeda dengan ekspresi yang mengembalikan angka. Selain itu, nama tab yang sebenarnya berubah berdasarkan tipe data yang dikembalikan.
9) Tab **Tools | Transform** untuk query yang mengembalikan angka ditampilkan pada Gambar 2.4:



Gambar 2. 4 Alat Kontekstual, tab Transformasi

Sekarang setelah kita meninjau pita, mari beralih ke panel Kueri.

Panel Queries

Panel Kueri menampilkan list berbagai kueri yang terkait dengan file Power BI saat ini. Saat kueri dibuat, kueri tersebut ditampilkan di sini. Selain itu, jika kesalahan muncul selama eksekusi kueri, area ini menampilkan rangkaian baris kesalahan yang dihasilkan oleh kueri.

Mengklik kanan kueri akan menyediakan berbagai opsi, seperti menghapus, menyalin, menduplikasi, merujuk, dan mengelompokkan kueri. Beberapa opsi ini memerlukan penjelasan tambahan:

- **Copy**: Ini sama dengan memilih kueri dan menekan Gtrl + G lalu Gtrl + V. Menyalin kueri akan membuat salinan kueri serta semua kueri dan parameter yang direferensikan/dikaitkan. Misalnya, jika kueri menggunakan parameter untuk menentukan sumber data, seperti nama server SQL, basis data, atau jalur file, dan/atau merupakan kueri yang menggabungkan atau menambahkan beberapa kueri lain, maka salinan semua kueri dan parameter terkait ini juga akan dibuat.
- **Duplicate**: Membuat duplikat kueri akan membuat salinan kueri itu sendiri. Pada dasarnya, semua kode M untuk kueri disalin dan ditempel ke kueri baru. Kueri dan parameter terkait tidak diduplikasi, seperti pada opsi **Copy**.
- **Refence**: Memilih untuk membuat referensi akan membuat kueri baru yang sumber datanya adalah kueri asli. Referensi sering digunakan pada kueri dasar yang melakukan operasi transformasi data umum yang diperlukan oleh beberapa kueri/tabel.
- **Enabled Load**: Kueri dapat diaktifkan atau dinonaktifkan untuk pemuatan dalam tujuan klien akhir seperti Microsoft Excel atau Power BI Desktop. Menonaktifkan pemuatan akan mempertahankan kueri dalam pengalaman Power Query, tetapi kueri tersebut tidak lagi membuat tabel dalam aplikasi klien. Sering kali, kueri yang direferensikan dinonaktifkan

dari pemuatan karena kueri tersebut melakukan langkah transformasi data antara.

Mari kita jelajahi aplikasi praktis mengenai cara menggunakan fitur-fitur ini. Bayangkan mengumpulkan aktivitas dari sistem manajemen hubungan pelanggan (CRM), seperti Dynamics 365. Anda dapat membuat parameter yang menentukan titik akhir sistem data sumber. Parameter ini kemudian akan digunakan dalam kueri Email yang terhubung ke tabel/entitas Email yang mendasarinya untuk mengimpor data. Anda kemudian memilih kolom Pengguna dan Dibuat Pada dan menambahkan langkah untuk menghapus semua kolom lainnya. Selain itu, Anda menambahkan kolom Jenis kustom yang hanya mengembalikan Email untuk setiap baris.

Sekarang, jika Anda ingin mengimpor janji temu juga, pilihan yang baik di sini adalah menduplikasi kueri Email Anda dan mengganti nama kueri duplikat menjadi Janji Temu. Anda tidak ingin menyalin kueri karena ini akan membuat duplikat parameter Anda juga. Anda kemudian cukup mengedit kueri untuk mengembalikan Janji Temu untuk jenis tersebut sementara semua transformasi data lainnya tetap sama.

Terakhir, Anda ingin mendapatkan tabel tunggal terakhir yang disebut Email dan Janji Temu. Kueri ini hanya menambahkan kueri Email dan Janji Temu. Kueri dasar ini direferensikan oleh kueri penambahan baru ini. Anda kemudian klik kanan kueri Email dan Janji Temu dan nonaktifkan pemuatan karena tabel akhir Anda berisi semua baris dari kedua kueri dasar.

Panel Pengaturan Queries (Panel Pengaturan Kueri)

Panel Pengaturan Kueri menyediakan akses ke properti untuk kueri, seperti nama kueri. Secara default, nama kueri menjadi nama tabel dalam model semantik.

Yang lebih penting, panel **Pengaturan Kueri** menyertakan list **Langkah yang Diterapkan** untuk kueri. Kueri sebenarnya hanyalah serangkaian langkah yang diterapkan, atau transformasi, dari data. Saat Anda mengubah

data yang diimpor oleh kueri, langkah dibuat untuk setiap transformasi. Jadi, menjalankan kueri untuk menyegarkan data dari sumber data hanyalah masalah menjalankan kembali langkah-langkah atau operasi transformasi ini.

Panel Pratinjau (Preview Pane)

Panel **Pratinjau** menyediakan pratinjau data yang sedang dimuat dan diubah. Area ini bersifat kontekstual, menampilkan tabel data untuk langkah kueri yang sedang dipilih. Header kolom dalam area panel **Pratinjau** dapat digunakan untuk mengganti nama kolom serta mengakses berbagai operasi transformasi dan pembersihan.

Status Bar (Bilah Status)

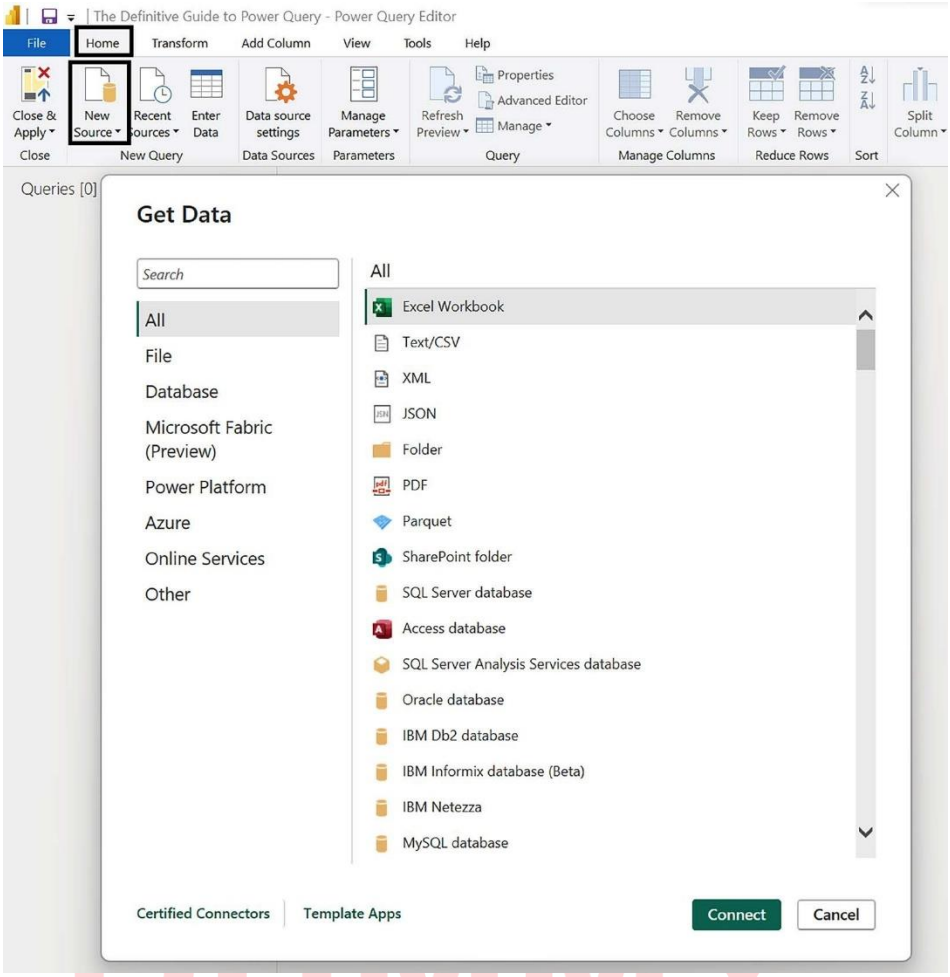
Area **Bilah Status** bersifat kontekstual berdasarkan kueri yang dipilih dalam Editor Power Query. Informasi bermanfaat, seperti jumlah baris dan kolom dalam tabel dan kapan pratinjau terakhir data dimuat, ditampilkan di sini.

Sekarang setelah kita menyelesaikan tur singkat Editor Power Query, mari kita mulai menulis beberapa kode M dengan membuat kueri pertama kita.

2. Query Pertama Anda

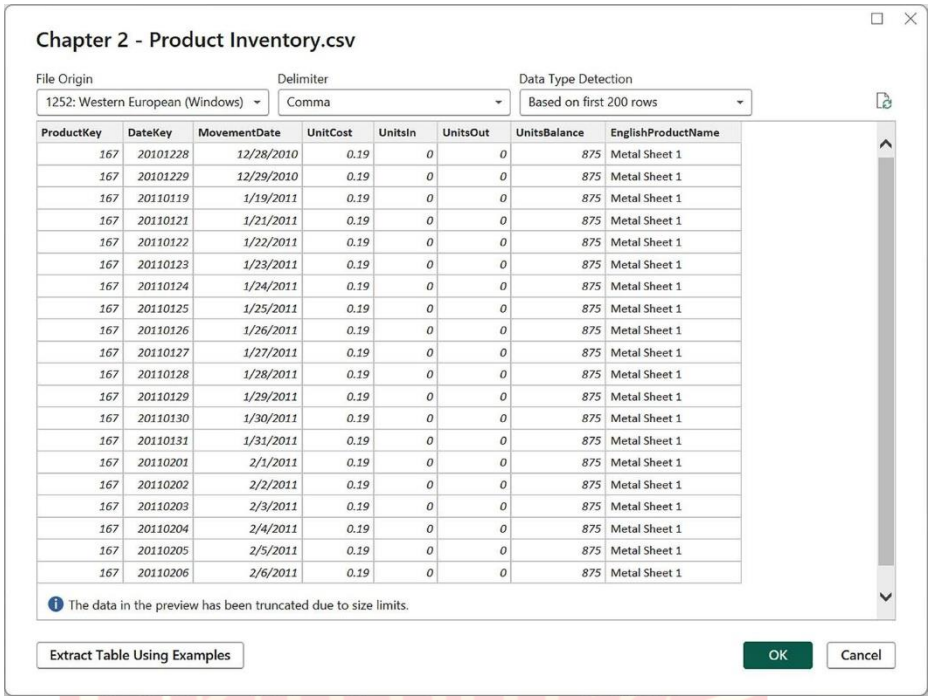
Untuk membuat kueri pertama Anda, Anda memerlukan file *Bab 2, Bekerja dengan Power Query/M* yang tersedia dengan konten tambahan untuk buku ini. File ini berisi ekspor dari database sampel AdventureWorks DW yang terkenal dan banyak digunakan dan dapat diunduh dari repositori Packt GitHub untuk buku ini. Lihat bagian Persyaratan Teknis di awal bab ini. Untuk membuat kueri, lakukan langkah-langkah berikut:

1) Di Editor Power Query, pilih tab **Home**, lalu klik ikon **New Source** untuk menampilkan jendela **Get Data**, seperti yang diperlihatkan pada Gambar 2.5:



Gambar 2. 5 Mendapatkan Data di Editor Power Query

2) Pilih **Teks/CSV** dari list **All** sumber data, lalu klik tombol **Connect**.
3) Arahkan ke lokasi file *Bab 2, Bekerja dengan Power Query/M*, pilih file, lalu klik tombol **Open** untuk menampilkan pratinjau data, seperti yang ditunjukkan pada Gambar 2.6:

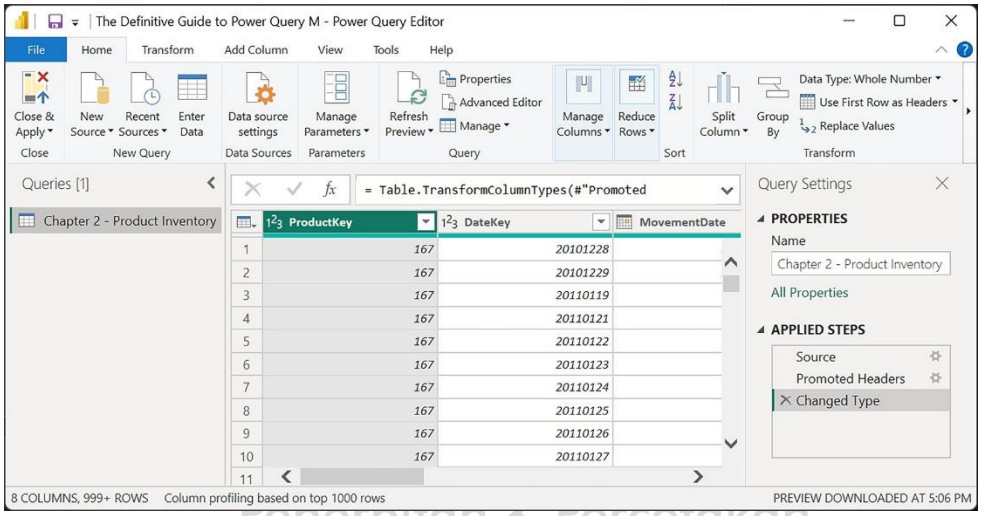


Gambar 2. 6 Pratinjau data

Bergantung pada sumber data yang dipilih, antarmuka ini berubah dan menyediakan berbagai opsi. Untuk file teks/CSV, seseorang dapat memilih asal file (kumpulan karakter), pemisah antara kolom, serta apakah akan mendeteksi tipe data atau tidak dan apakah deteksi tipe data tersebut terjadi pada seluruh model semantik atau sebagian dari model semantik. Seperti yang dicatat melalui ikon informasi yang ditunjukkan oleh (i), pratinjau umumnya dipotong menjadi hanya sebagian dari baris dalam model semantik.

4) Klik tombol **OK** untuk memuat pratinjau yang lebih besar (hingga 1.000 baris) ke dalam Editor Power Query.

Sekarang, jendela Editor Power Query akan terlihat mirip dengan Gambar 2.7:



Gambar 2. 7 Editor Power Query dengan pratinjau data

Perhatikan bahwa panel Kueri menampilkan nama kueri, yang dalam kasus ini, didasarkan pada nama file sumber. Panel Pratinjau menampilkan kolom dan data yang diimpor dari file. Panel Pengaturan Kueri juga menampilkan nama kueri serta tiga Langkah Terapan:

- **Source**: Ini adalah langkah dalam kueri yang mengakses file CSV
- **Tajuk yang Dipromosikan (Promoted Headers)**: Langkah ini mempromosikan baris pertama nilai sebagai nama kolom
- **Jenis yang Diubah (Changed Type)**: Langkah ini mengubah jenis data untuk kolom menggunakan deteksi otomatis berdasarkan 200 baris data pertama

Bilah status menunjukkan bahwa ada total 8 kolom dan lebih dari 999 baris data dan bahwa pembuatan profil kolom (dibahas nanti dalam bab ini) didasarkan pada 1.000 baris data pertama. Selain itu, di sisi kanan bilah status terdapat informasi tentang kapan pratinjau dimuat.

Terakhir, bilah rumus mencantumkan kode M aktual di balik langkah kueri terakhir, **Jenis yang Diubah**. Kita dapat melihat bahwa fungsi M yang digunakan untuk mengubah tipe data kolom adalah fungsi *Table.TransformColumnTypes*. Tipe data yang dipilih untuk setiap kolom

tercermin dalam tajuk kolom dalam panel **Pratinjau**, dengan bilangan bulat (*Int64.Type*) memiliki ikon **123**, tanggal (*type data*) memiliki ikon kalender, dan mata uang (*Currency.Type*) memiliki ikon **$.**

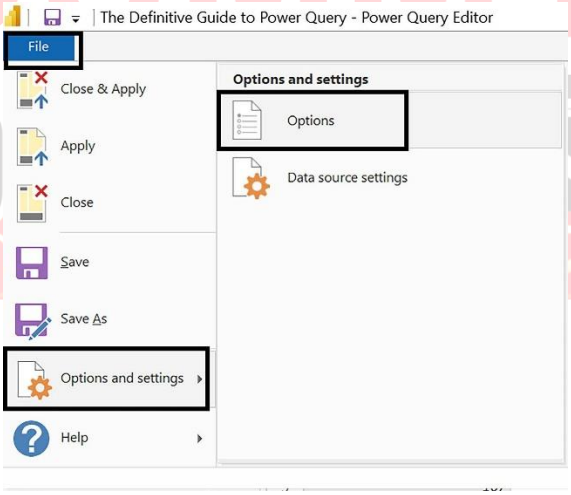
Nah, itu dia! Anda telah menulis kueri pertama dan kode M pertama Anda. Sekarang setelah kita memiliki kueri untuk digunakan di Editor Power Query, selanjutnya mari kita lihat berbagai opsi yang tersedia saat bekerja dengan kueri dan pengaturan sumber data.

3. Opsi dan Pengaturan Sumber Data

Ada sejumlah opsi yang dapat mengontrol tampilan dan perilaku Editor Power Query serta cara data dimuat ke dalam sistem. Selain itu, membuat kueri yang terhubung ke sumber data baru juga membuat sumber data di komputer lokal. Sumber data ini dapat dikelola melalui Editor Power Query. Pertama-tama, mari kita lihat Opsi.

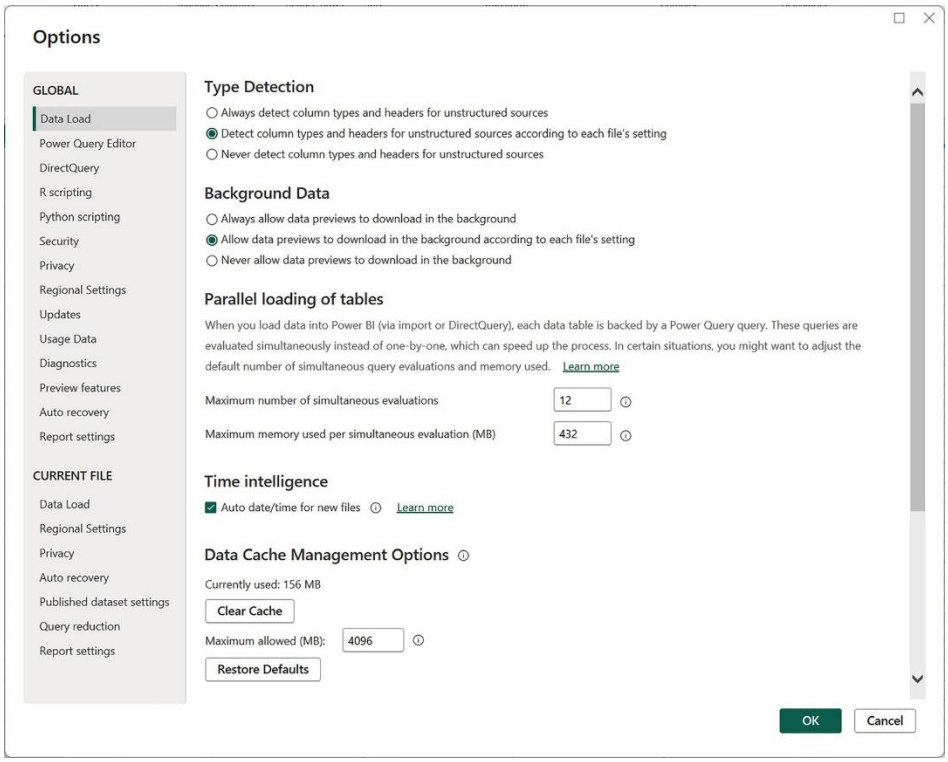
Option (Pilihan)

Untuk mengakses opsi yang mengontrol tampilan dan perilaku Editor Power Query, klik **File** di pita, lalu klik **Options and settings**, dan terakhir klik **Options**, seperti yang diperlihatkan dalam Gambar 2.8:



Gambar 2. 8 Mengakses Opsi dalam Editor Power Query

Melakukan rangkaian tindakan ini akan menampilkan jendela **Option**, seperti yang ditunjukkan pada Gambar 2.9:



Gambar 2. 9 Opsi di Power BI Desktop/Editor Power Query

Jendela **Options** pada Gambar 2.9 mencantumkan semua opsi yang tersedia untuk Power BI Desktop. Hanya sebagian kecil dari opsi ini yang memengaruhi Editor Power Query dan kueri transformasi data, khususnya kategori **Data Load** dan **Power Query Editor** yang tercantum di panel navigasi kiri.

Penting juga untuk dicatat bahwa opsi dalam pengalaman Power Query Desktop di alat lain seperti Microsoft Excel mungkin sedikit berbeda dari opsi yang tercantum di sini. Namun, secara keseluruhan, opsi dalam pengalaman Power Query Desktop lainnya sangat mirip dan umumnya merupakan sebagian kecil dari opsi yang tercantum di sini. Selain itu, pengalaman Power Query Online umumnya tidak menawarkan jenis opsi yang dibahas di bagian ini karena pengalaman Power Query Online disediakan sebagai layanan.

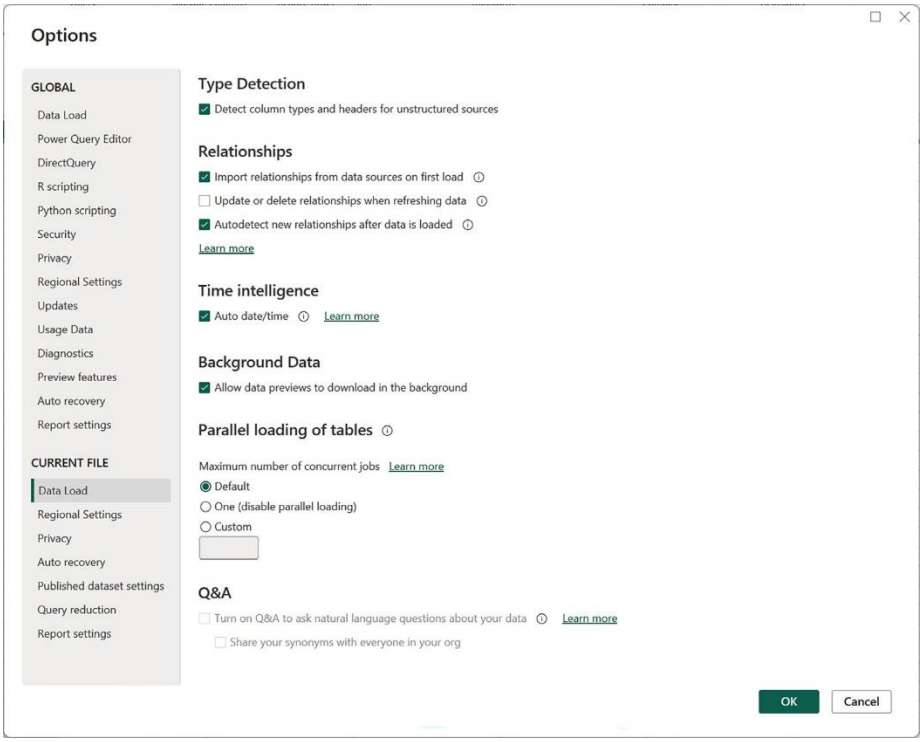
Gambar 2.9 menampilkan opsi **Pemuatan Data GLOBAL**. Opsi global memengaruhi semua file, meskipun pengaturan tertentu dapat ditimpa pada tingkat laporan (**CURRENT FILE**). Opsi **GLOBAL** ini mencakup yang berikut:

- **Deteksi Tipe (Type Detection)**: Ini mengontrol apakah tipe data dideteksi secara otomatis untuk sumber data tak terstruktur (seperti file teks atau Excel). Untuk sumber data terstruktur, tipe data diwarisi dari sumbernya. Defaultnya adalah mendeteksi tipe data untuk kolom (langkah **Tipe yang Diubah** dari Gambar 2.7) serta mendeteksi header secara otomatis (langkah **Header yang Dipromosikan** dari Gambar 2.7) dalam sumber data tak terstruktur menurut pengaturan untuk setiap file. Ingat kembali dari Gambar 2.6 bagaimana pengaturan **Deteksi Tipe Data** memungkinkan kita untuk mengontrol apakah dan bagaimana deteksi tipe data terjadi untuk sumber data file CSV tak terstruktur Anda. Banyak profesional data menyarankan untuk mengubah pengaturan ini agar tidak pernah mendeteksi tipe data secara otomatis. Namun, tentu saja ada faktor kemudahan bagi pengguna bisnis biasa karena deteksi tipe otomatis Power Query cukup baik.
- **Data Latar Belakang (Background Data)**: Pengaturan ini mengontrol apakah pratinjau data di Editor Power Query dan diunduh di latar belakang. Mengunduh pratinjau di latar belakang memungkinkan Anda untuk terus bekerja di Editor Power Query tanpa harus menunggu pratinjau data dimuat sepenuhnya. Biasanya, sebaiknya biarkan pengaturan ini pada pengaturan default yang mengizinkan pemuatan pratinjau data di latar belakang sesuai dengan pengaturan setiap file.
- **Pemuatan tabel secara parallel (Parallel loading of tables)**: Untuk file yang berisi beberapa kueri, Power BI memuat kueri secara paralel untuk mengoptimalkan kinerja. Namun, dalam situasi tertentu, Anda mungkin ingin menyesuaikan parameter ini, termasuk **Jumlah maksimum**

evaluasi simultan, dan Memori maksimum yang digunakan per evaluasi simultan (MB). Untuk informasi selengkapnya tentang pengaturan ini, lihat dokumentasi Microsoft: https://learn.microsoft.com/en-us/power-bi/create-reports/desktop-evaluation-configuration Pengaturan konfigurasi evaluasi untuk Desktop - Power BI | Microsoß Learn.

- **Kecerdasan waktu (Time intelligence)**: Secara default, Power BI Desktop secara otomatis membuat tabel tanggal tersembunyi untuk setiap tanggal atau kolom tanggal/waktu. Meskipun ini dapat memudahkan pengguna biasa, sebagian besar profesional data sangat menyarankan untuk menonaktifkan fitur ini. Mengaktifkan fitur ini sering kali dapat menyebabkan model semantik yang membengkak dan jauh lebih besar dari yang diperlukan.
- **Manajemen Cache Data**: Editor Power Query menyimpan hasil pratinjau untuk kueri dalam cache untuk mengoptimalkan kinerja agar tampilan lebih cepat. Pengaturan ini memperlihatkan seberapa banyak ruang disk yang saat ini dipakai oleh cache ini, sekaligus memungkinkan Anda menetapkan jumlah maksimum ruang disk yang digunakan, dan akhirnya menghapus cache jika perlu.

  Seperti yang disebutkan, pengaturan **GLOBAL** dapat diganti per file, menggunakan pengaturan **CURRENT FILE** untuk opsi **Pemuatan Data**. Opsi **Pemuatan Data** untuk **CURRENT FILE** ditunjukkan pada Gambar 2.10:



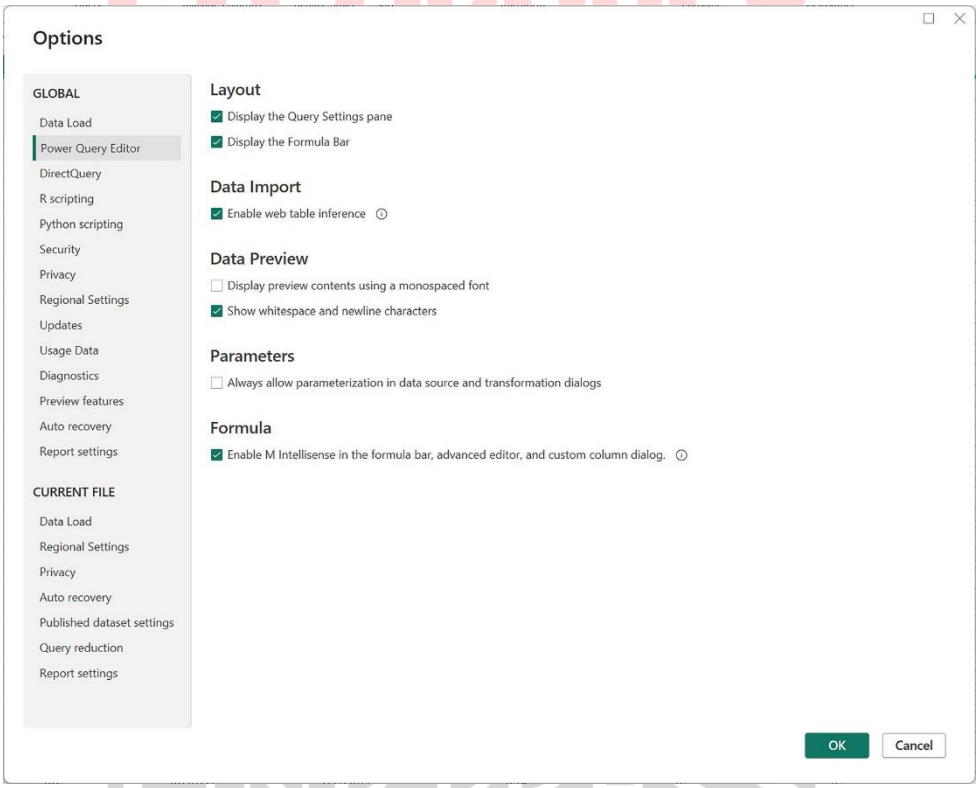
Gambar 2. 10 Opsi Pemuatan Data untuk CURRENT FILE

Opsi **Pemuatan Data** untuk **CURRENT FILE** mencakup kemampuan untuk mengganti pengaturan **GLOBAL** untuk **Type Detection, Time intelligence, Background Data,** dan **Parallel loading of tables**. Selain itu, ada dua pengaturan tambahan:

- **Hubungan (Relatiomship)**: Mengontrol impor dan deteksi otomatis hubungan antara tabel yang dimuat melalui kueri. Default untuk pengaturan ini ditunjukkan pada Gambar 2.10. Banyak profesional data menyarankan untuk menonaktifkan deteksi otomatis hubungan. Untuk informasi selengkapnya tentang pengaturan ini, rujuk ke Greate dan kelola hubungan di Power BI Desktop - Power BI | Microsoß Learn (https://learn.microsoft.com/en-us/power-bi/transform-model/desktop-create-and- manage-relationships#automatic-relationship-updates).
- **Q&A**: Mengaktifkan atau menonaktifkan fitur Tanya Jawab bahasa alami Power BI. Pengaturan ini dinonaktifkan pada Gambar 2.10 karena tidak

ada kueri yang benar-benar dimuat ke dalam model **semantik**. Untuk informasi selengkapnya tentang pengaturan ini, rujuk Sumber data untuk Tanya Jawab bahasa alami (https://learn.microsoft.com/en-us/power-bi/natural-language/q-and-a-data-sources).

Selain opsi **Data Load** untuk **GLOBAL** dan **CURRENT FILE**, ada juga opsi untuk mengendalikan tampilan dan perilaku **Editor Power Query**. Opsi ini diperlihatkan pada Gambar 2.11:



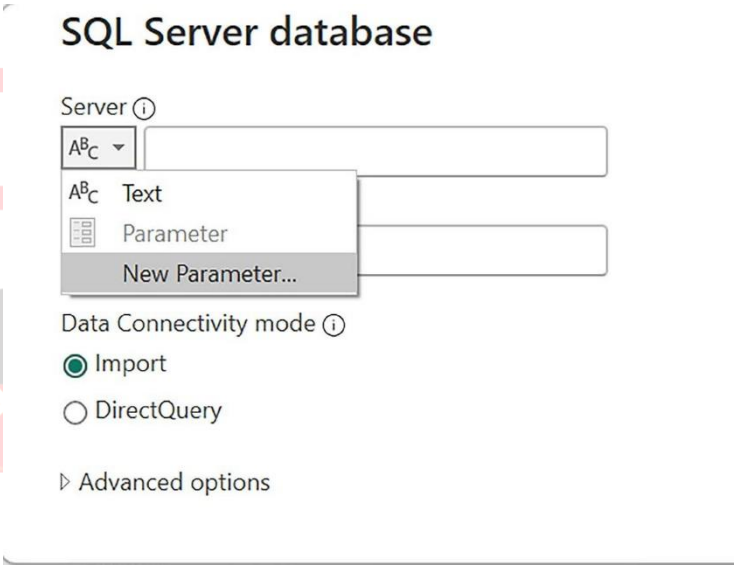
Gambar 2. 11 Opsi Editor Power Query di Power BI Desktop

Opsi yang tercantum dalam Gambar 2.11 mencakup yang berikut:

- **Tata Letak (Layout)**: Mengontrol apakah panel **Pengaturan Query** dan bilah rumus ditampilkan di Editor Power Query atau tidak.
- **Data Import**: Mengaktifkan opsi ini memanfaatkan fungsi *Web.BrowserContents* saat mengambil data dari halaman web. Fungsi ini dapat mendeteksi pola berulang dalam konten yang melampaui deteksi sederhana tabel HTML.

- **Data Preview**: Mengontrol tampilan dan nuansa konten pratinjau.
- **Parameters**: Pengaturan ini dapat sangat berguna dalam lingkungan tempat laporan bergerak melalui fase pengembangan, pengujian, dan produksi. Tanpa mengaktifkan pengaturan ini, pengembang hanya memiliki dua pilihan. Pertama, mengedit pengaturan sumber data (lihat bagian Pengaturan sumber data dalam bab ini), atau membuat parameter kueri secara manual (lihat Bab 9, Parameter dan Fungsi Kustom) untuk hal-hal seperti nama server dan basis data, lalu mengedit kueri secara manual untuk menggunakan parameter ini. Dengan mengaktifkan pengaturan ini, pembuatan parameter menjadi bagian dari dialog sumber data dan transformasi.

  Misalnya, saat menghubungkan ke basis data SQL Server, mengaktifkan fitur ini akan menyediakan opsi dialog baru untuk menentukan parameter baru bagi **Server** dan **Basis Data**, seperti yang ditunjukkan pada Gambar 2.12:



Gambar 2. 12 Opsi dialog Parameter Baru…

- **Formula**: Mengaktifkan atau menonaktifkan M IntelliSense (teknologi pengetikan cepat milik Microsoft).

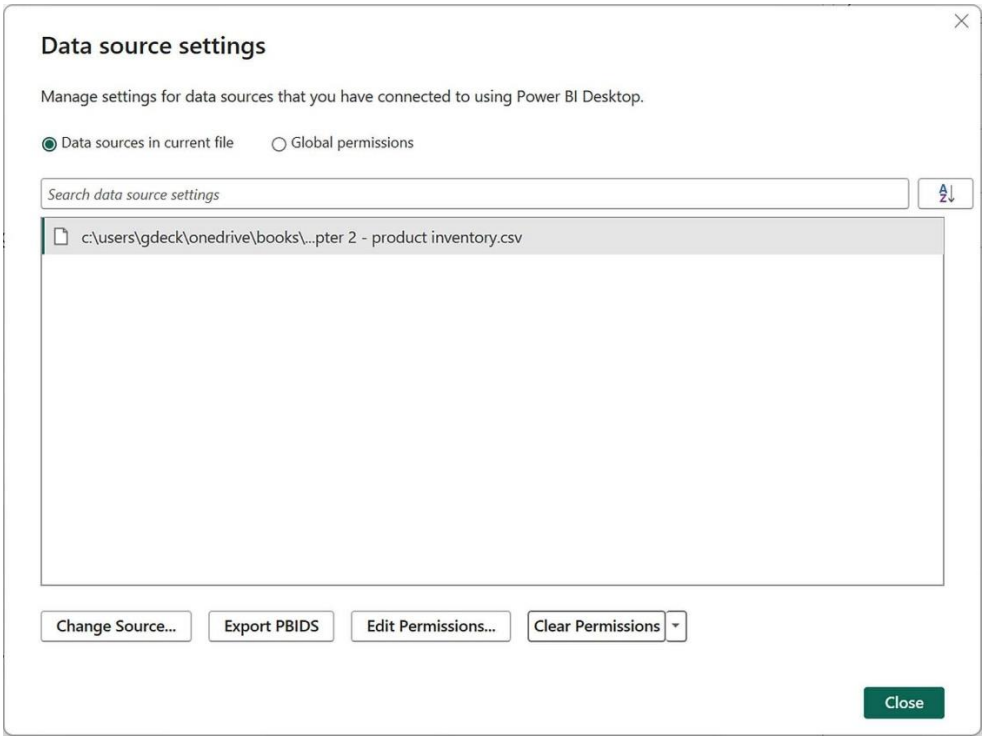
Sekarang setelah kita menjelajahi opsi untuk mengendalikan tampilan dan perilaku Editor Power Query, mari kita bahas secara singkat pengaturan Sumber data berikutnya.

Setelan sumber data (Data Source Settings)

Sama seperti kebanyakan tabel dalam model data yang didukung oleh kueri bahasa M, demikian pula kebanyakan kueri didukung oleh sumber data. Definisi sumber data ini disimpan sebagai bagian dari file desktop (baik Power BI Desktop maupun Excel).

Selain itu, kredensial yang digunakan untuk mengautentikasi ke sumber data ini di-cache secara lokal di komputer tempat pengalaman Power Query Desktop berjalan. Yang terpenting, kredensial sumber data ini tidak disimpan dalam file desktop itu sendiri. Ini berarti bahwa jika membuka file desktop di komputer lain, kredensial yang digunakan untuk mengautentikasi ke sumber data harus dimasukkan kembali.

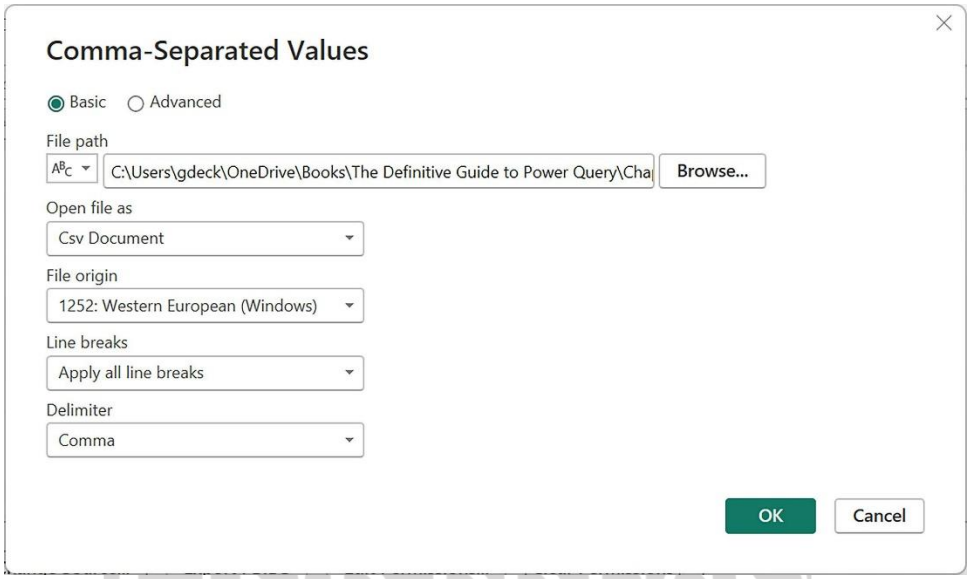
Setelah satu atau beberapa sumber data dibuat, umumnya melalui pembuatan kueri, pengaturan untuk sumber data ini dapat dilihat dan diedit dengan mengeklik **File** di pita, lalu pada **Options and settings**, dan terakhir pada Pengaturan sumber data, seperti yang ditunjukkan pada Gambar 2.8. Melakukan rangkaian tindakan ini akan menampilkan jendela **Data source settings** yang ditunjukkan pada Gambar 2.13:



Gambar 2. 13 Jendela pengaturan sumber data

Tombol radio **Data sources in current file** hanya menampilkan sumber data yang terdapat dalam berkas desktop saat ini, sementara **Opsi Global** permissions menampilkan sumber data yang digunakan di seluruh berkas desktop Anda. **Data sources in current file** menampilkan tombol untuk **Change Source…, Export PBIDS, Edit Permissions…,** dan **Clear Permissions** sementara opsi **Global permissions** hanya menampilkan tiga tombol terakhir.

Dengan mengklik tombol **Change Source…** yang ditunjukkan pada Gambar 2.13 untuk kueri yang dibuat dalam bab ini, Anda dapat melihat dan mengubah opsi konfigurasi untuk sumber data seperti yang ditunjukkan pada Gambar 2.14:



Gambar 2. 14 Jendela pengaturan sumber data Nilai yang Dipisahkan Koma

Tombol **Export PBIDS** menyimpan sumber data sebagai dokumen JSON. Perhatikan, jika Anda menggunakan file PBIX bab yang diunduh dari GitHub, Anda perlu mengedit parameter *Parameter1* untuk menunjuk ke lokasi yang valid untuk sistem Anda. Misalnya, sumber data yang dibuat dalam bab ini di bagian Kueri pertama Anda tampak mirip dengan berikut ini saat diekspor:

```
{
  "version": "0.1",
  "connections": [
    {
      "details": {
        "protocol": "file",
        "address": {
          "path": "c:\\temp\\Product inventory.csv"
        },
        "authentication":
        null, "query": null
      },
      "options": {},
      "mode": null
    }
     ]
    }
```

Bergantung pada jenis sumber data, tombol **Edit Permissions…** memungkinkan Anda untuk melihat dan mengedit hal-hal seperti kredensial autentikasi, tingkat privasi, enkripsi, dan kueri basis data asli yang disetujui untuk sumber data seperti SQL Server.

Ini menyimpulkan ikhtisar kita tentang pengalaman Power Query Desktop. Sekarang mari beralih ke cara mengedit kode M yang dihasilkan dengan menggunakan Editor Power Query.

D. Mengedit Kode yang Dihasilkan Berdasarkan Pengalaman

Seperti yang ditunjukkan di bagian Kueri pertama Anda di bab ini, penggunaan **Graphical User Interface (GUI)** pengalaman Power Query Desktop untuk menghubungkan dan mengubah data menghasilkan kode bahasa M. Kode yang dihasilkan ini dapat diubah atau diedit menggunakan bilah rumus.

Misalnya, di area **Langkah yang Diterapkan** pada panel **Query Settings**, dengan mengeklik langkah **Sumber** untuk kueri lalu mengeklik panah bawah di ujung kanan bilah rumus, Anda dapat melihat kode bahasa M lengkap untuk langkah **Sumber**, seperti berikut:

```
= Csv.Document(File.Contents("C:\Users\gdeck\OneDrive\Books\The Definitive Guide
to Power Query\Chapter 2\Chapter 2 - Product Inventory.

csv"),[Delimiter=",", Columns=8, Encoding=1252, QuoteStyle=QuoteStyle.None])
```

Seperti yang dapat Anda lihat dalam kode ini, terdapat dua fungsi bertingkat yang digunakan untuk menghubungkan ke berkas CSV, *Csv.Document* dan *File.Contents*.

- Fungsi *File.Contents* memiliki satu parameter yang diteruskan ke fungsi tersebut, jalur berkas, dan nama berkas CSV.
- *Csv.Document* memiliki dua parameter yang diteruskan kepadanya, yang pertama adalah hasil dari fungsi File.Content dan yang kedua adalah record yang terdiri dari tiga pasangan kunci/nilai. Record dibahas lebih lanjut dalam Bab 4, Memahami Nilai dan Ekspresi.

Anda dapat mengedit kode bahasa M dalam bilah rumus untuk membuat perubahan yang diperlukan pada kode tersebut. Misalnya, perubahan umum untuk berkas CSV adalah menghapus sepenuhnya pasangan kunci/nilai kedua dari record, seperti berikut ini:

```
= Csv.Document(File.Contents("C:\Users\gdeck\OneDrive\Books\The
Definitive Guide to Power Query\Chapter 2\Chapter 2 - Product Inventory.
csv"),[Delimiter=",",Encoding=1252, QuoteStyle=QuoteStyle.None])
```

Untuk melakukan perubahan ini, cukup klik di dalam bilah rumus, buat perubahan yang diperlukan, lalu klik di luar bilah rumus atau tekan tombol Enter.

Pengeditan ini sering dilakukan untuk membantu kueri yang mengakses file CSV agar siap digunakan di masa mendatang. Misalnya, jika kolom kesembilan ditambahkan ke sumber data, versi asli kode M akan mengabaikan kolom tambahan tersebut karena hanya delapan kolom yang ditentukan. Ini adalah skenario umum dengan lembar kerja Excel di mana setiap bulan ditambahkan sebagai kolom baru, misalnya.

Sebaliknya, dengan pasangan kunci/nilai *Columns=8* dihapus dari record yang memuat parameter kedua dalam fungsi *Csv.Document*, kolom tambahan akan secara otomatis dikenali dan disertakan dalam data saat kueri diperbarui.

Perubahan lain yang dapat dilakukan melibatkan fungsi **Changed Type**. Mengklik langkah **Changed Type** dalam bagian Applied Steps dari panel **Query Settings** akan menampilkan kode bahasa M berikut dalam bilah rumus:

```
= Table.TransformColumnTypes(#"Promoted Headers",{{"ProductKey", Int64.Type},

{"DateKey", Int64.Type}, {"MovementDate", type date}, {"UnitCost", Currency.
Type}, {"UnitsIn", Int64.Type}, {"UnitsOut", Int64.Type}, {"UnitsBalance",
Int64.Type}, {"EnglishProductName", type text}})
```

Anda akan mempelajari lebih lanjut tentang fungsi ini dan sintaksisnya di bab-bab selanjutnya dari buku ini, tetapi secara sederhana, kode ini mengubah setiap kolom menjadi tipe data tertentu (misalnya numerik, teks, logika, dll.).

Kolom *MovementDate* memiliki tipe data tanggal. Namun, Anda mungkin ingin menyertakan stempel waktu juga. Anda dapat melakukannya dengan mengedit kode di bilah rumus menjadi berikut:

```
= Table.TransformColumnTypes(#"Promoted Headers",{{"ProductKey", Int64.
Type}, {"DateKey", Int64.Type}, {"MovementDate", type datetime}, {"UnitCost",
Currency.Type}, {"UnitsIn", Int64.Type}, {"UnitsOut", Int64.Type},

{"UnitsBalance", Int64.Type}, {"EnglishProductName", type text}})
```

Melakukan tindakan ini akan menambahkan stempel waktu 12:00:00 AM ke semua tanggal dalam kolom *MovementDate*.

Seperti yang Anda lihat, bilah rumus adalah tempat pertama dan paling jelas di mana Anda dapat menulis kode M kustom Anda sendiri. Selanjutnya, kita akan menjelajahi opsi lain untuk memasukkan kode M kustom, yaitu menambahkan kolom kustom.

E. Membuat Kolom Khusus

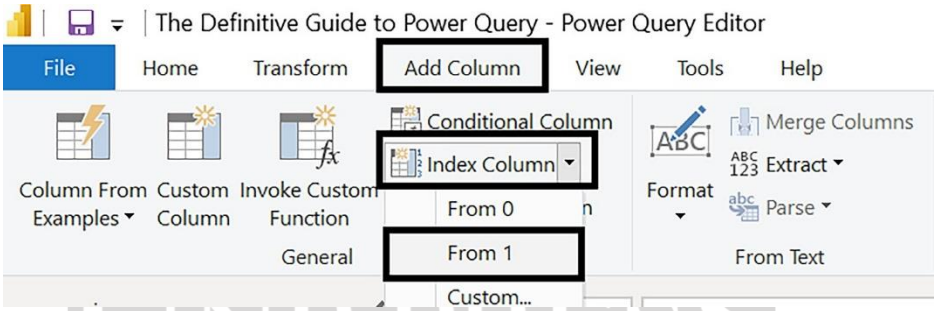
Membuat kolom kustom merupakan aktivitas transformasi data yang umum saat bekerja dengan Power Query dan bahasa M. Meskipun jumlah skenario untuk menambahkan kolom kustom pada dasarnya tidak terbatas, contoh umum mungkin menggabungkan kolom harga satuan dan kolom kuantitas menjadi satu kolom total penjualan. Di bagian ini, kita akan menjelajahi beberapa cara untuk membuat kolom kustom baik menggunakan GUI Editor Power Query maupun menulis kode M kustom.

1. Menambahkan Kolom Indeks

Kolom umum yang ditambahkan ke kueri M adalah kolom indeks, yang memberi nomor baris secara berurutan. Kolom indeks sangat bermanfaat dalam skenario tertentu, seperti **Mean Time Between Failure (MTBF)**, di mana perlu membandingkan perbedaan antara dua baris data.

Untuk menambahkan kolom indeks ke kueri yang sudah ada yang dibuat di bagian Kueri Pertama Anda di bab ini, lakukan hal berikut:

1) Klik tab **Add Column** di pita, lalu klik panah tarik-turun di sebelah kanan opsi **Index Column**, dan terakhir pilih opsi, seperti yang ditunjukkan pada Gambar 2.15:



Gambar 2. 15 Menambahkan kolom indeks mulai dari 1

2) Melakukan tindakan ini akan menambahkan **Added Index** Tambahan di area **Applied Steps** pada panel **Power Query Settings**. Bilah rumus menampilkan kode M berikut dengan langkah **Added Index** yang dipilih:

```
= Table.AddIndexColumn(#"Changed Type", "Index", 1, 1, Int64.Type)
```

Dalam kasus ini, fungsi *Table.AddIndexColumn* digunakan untuk menambahkan kolom bernama Index (parameter kedua) ke tabel yang dikembalikan oleh langkah #"*Changed Type*" (parameter pertama). Kolom Index ini dimulai pada angka *1* (parameter ketiga), bertambah *1* (parameter keempat), dan memiliki tipe data bilangan bulat (*Int64.Type*). Sekarang mari kita lihat cara menambahkan kolom menggunakan contoh.

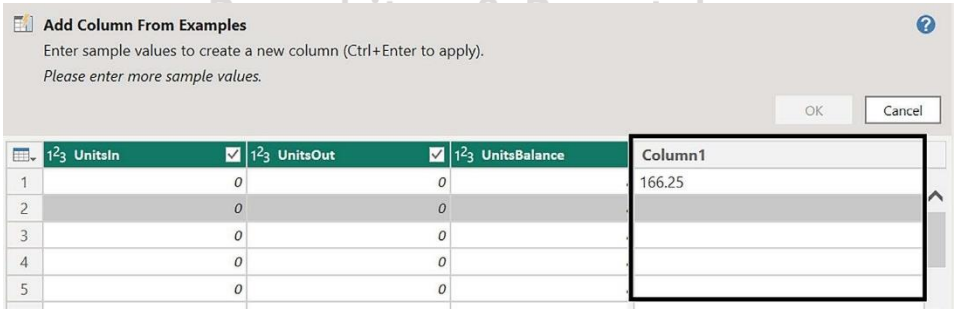
2. Menambahkan Kolom Dengan Contoh

Menambahkan kolom melalui contoh menerapkan pencocokan pola **Machine Learning (ML)** tingkat lanjut untuk mencari tahu makna di balik contoh data yang dimasukkan ke baris baru oleh pengguna dan secara otomatis menulis kode M yang sesuai untuk melakukan transformasi data yang sesuai. Contoh klasiknya adalah memiliki kolom nama depan dan kolom nama

belakang. Dengan menambahkan kolom melalui contoh, seseorang dapat memasukkan nama depan dan belakang ke kolom baru yang dipisahkan oleh spasi dan Power Query akan secara otomatis membuat kode M untuk menggabungkan nilai-nilai di dua kolom asli.

Kita dapat menguji penambahan kolom dengan contoh dengan mengikuti langkah-langkah berikut:

1) Klik pada tab **Add Column** pada pita, lalu klik fungsi **Column From Examples** di area **General** pada pita. Ini akan mengubah panel **Preview** menjadi seperti Gambar 2.16:



Gambar 2. 16 Tambahkan Kolom Dari Contoh

2) Seperti yang ditunjukkan pada Gambar 2.16, ketik 166,25 pada beberapa baris pertama. Power Query secara otomatis menentukan bahwa angka ini adalah perkalian kolom **UnitsCost** dan **UnitsBalance**.
3) Klik tombol **OK**.

Kolom baru dibuat bernama **Multiplication** dengan rumus berikut:

```
= Table.AddColumn(addIndex, "Multiplication", each [UnitCost] * [UnitsBalance],
type number)
```

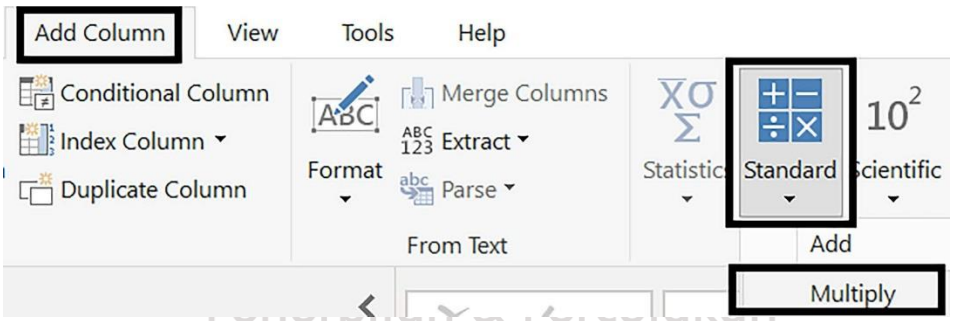
Lanjutkan dan hapus kolom **Multiplication** ini dengan mengklik kanan tajuk kolom dan memilih **Remove**, atau cukup klik ikon **X** di sebelah kiri langkah **Inserted Multiplication** yang ditambahkan ke area **Applied Steps** di panel **Query Settings**.

Sekarang mari beralih ke pembuatan/pengubahan kolom menggunakan operasi matematika.

3. Operasi Matematika

Data untuk kueri yang dibuat di bagian Kueri pertama Anda dalam bab ini mencakup kolom UnitCost dan kolom UnitsBalance. Daripada menyimpan kedua kolom ini dalam data, Anda mungkin ingin menggabungkan kolom-kolom ini menjadi satu kolom yang berisi kedua kolom yang dikalikan bersama. Dalam beberapa skenario, hal ini dapat menghasilkan ukuran model semantik yang lebih kecil. Hal ini dapat dilakukan dengan mengikuti langkah-langkah berikut:

1) Pada panel **Preview**, klik tajuk kolom untuk kolom **UnitCost**.
2) Sambil menahan tombol Gtrl pada keyboard, selanjutnya pilih tajuk kolom untuk kolom **Units-Balance**.
3) Klik pada tab **Add Column** pada pita, lalu klik opsi **Standard** dalam bagian **From Number** pada pita, dan terakhir, pilih **Multiply**, seperti yang ditunjukkan pada Gambar 2.17:



Gambar 2. 17 Mengalikan dua kolom

Melakukan tindakan ini akan menambahkan langkah **Inserted Multiplication** di area **Applied Steps** pada panel **Power Query Settings**. Bilah rumus menampilkan kode M berikut dengan langkah **Inserted Multiplication** dipilih:

```
= Table.AddColumn(#"Added Index", "Multiplication", each [UnitCost] *
[UnitsBalance], Currency.Type)
```

Di sini, fungsi *Table.AddColumn* digunakan untuk menambahkan kolom bernama **Multiplication** (parameter kedua) ke tabel yang dikembalikan oleh

langkah #"*Added Index*" (parameter pertama), di mana untuk setiap baris, kolom **UnitCost** dikalikan dengan kolom **UnitsBalance** (parameter ketiga) dan tipe data untuk kolom tersebut adalah *Currency.Type* (parameter keempat).

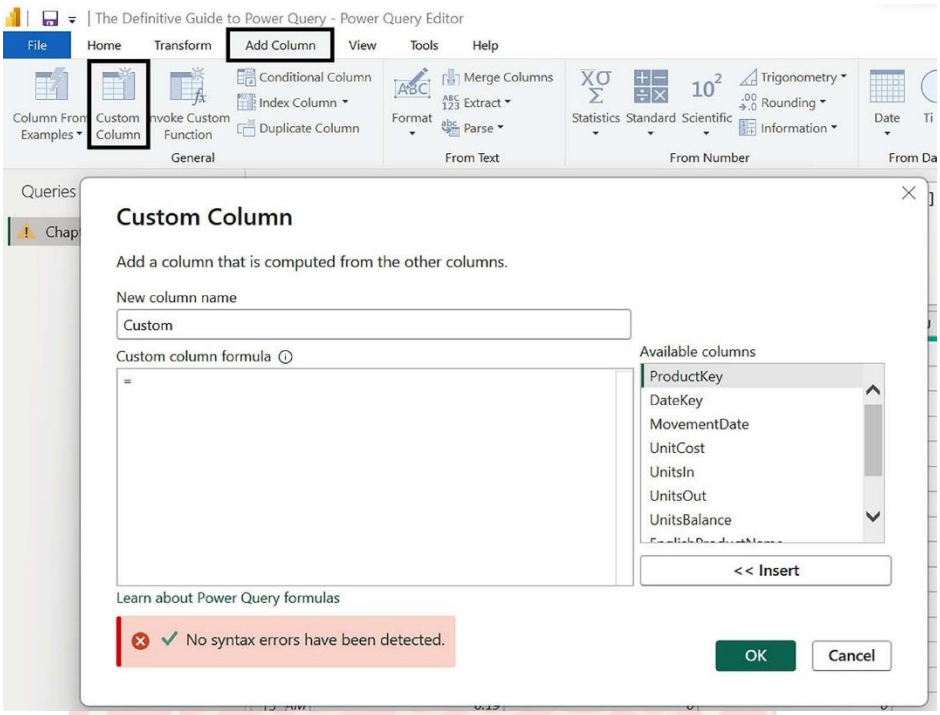
Jangan khawatir jika beberapa sintaksis, seperti penggunaan kata kunci *each*, tidak sepenuhnya masuk akal saat ini. Sintaksis ini telah dijelaskan dengan baik di bab-bab berikutnya.

Kita sekarang telah menjelajahi dua metode yang dapat digunakan untuk membuat kolom kustom menggunakan GUI dari Power Query Editor. Namun, tidak ada skenario yang mengharuskan kita untuk menulis kode M apa pun. Dalam kedua kasus, kode M dibuat secara otomatis untuk kita. Oleh karena itu, sekarang mari kita lihat cara menambahkan kolom kustom di mana kita benar-benar menulis sedikit kode M sendiri.

4. Menambahkan Kolom Kode M Khusus

Dalam contoh berikut ini, kita akan memasukkan sendiri beberapa kode M sederhana. Di sini, sekali lagi, kita akan tetap memanfaatkan Editor Power Query untuk menulis sebagian besar kode M. Untuk membuat kolom kode M kustom, lakukan langkah-langkah berikut:

1) Klik tab **Add Column** pada pita, lalu klik opsi **Custom Column** di bagian General pada pita untuk menampilkan dialog **Custom Column**, seperti yang ditunjukkan pada Gambar 2.18:



Gambar 2. 18 Dialog Kolom Kustom

2) Ganti kata **Custom** di bawah **New column name** dengan teks *MovementBalance*.
3) Di **Custom column formula**, masukkan kode berikut:

```
( [UnitsIn] - [UnitsOut] ) * [UnitCost]
```

4) Klik tombol **OK**.

Seperti kolom **Multiplication** sebelumnya, kolom ini juga melakukan beberapa operasi matematika sederhana pada beberapa kolom. Dalam kasus ini, untuk setiap baris data, kolom **UnitsOut** dikurangi dari kolom **UnitsIn** dan hasilnya kemudian dikalikan dengan kolom **UnitCost**. Kolom yang dihasilkan diberi nama **MovementBalance** dan berisi biaya positif atau negatif dari selisih unit yang dipindahkan masuk dan keluar pada setiap baris.

Melakukan tindakan ini akan menambahkan langkah **Added Custom** di area **Applied Steps** pada panel **Power Query Settings**. Bilah rumus menampilkan kode M berikut dengan langkah **Added Custom** dipilih:

```
= Table.AddColumn(#"Inserted Multiplication", "MovementBalance", each (
[UnitsIn] - [UnitsOut] ) * [UnitCost])
```

Rumus ini sangat mirip dengan kolom **Multiplication** kita sebelumnya. Perbedaan yang mencolok adalah tidak adanya parameter keempat yang menentukan tipe data untuk kolom baru. Jika kita perhatikan dengan saksama tajuk kolom untuk kolom baru kita, kita dapat melihat ikon tipe data di sebelah kiri menampilkan **ABC123**, yang menunjukkan bahwa kolom ini tidak memiliki tipe data tertentu. Untuk memperbaiki situasi ini, gunakan bilah rumus untuk mengubah biaya menjadi berikut:

```
= Table.AddColumn(#"Inserted Multiplication", "MovementBalance", each (
[UnitsIn] - [UnitsOut] ) * [UnitCost], Currency.Type)
```

Kita kini telah menetapkan bahwa tipe data untuk kolom **MovementBalance** kita adalah tipe *Currency*. Type dan ikon tipe data untuk kolom kita kini menampilkan $.

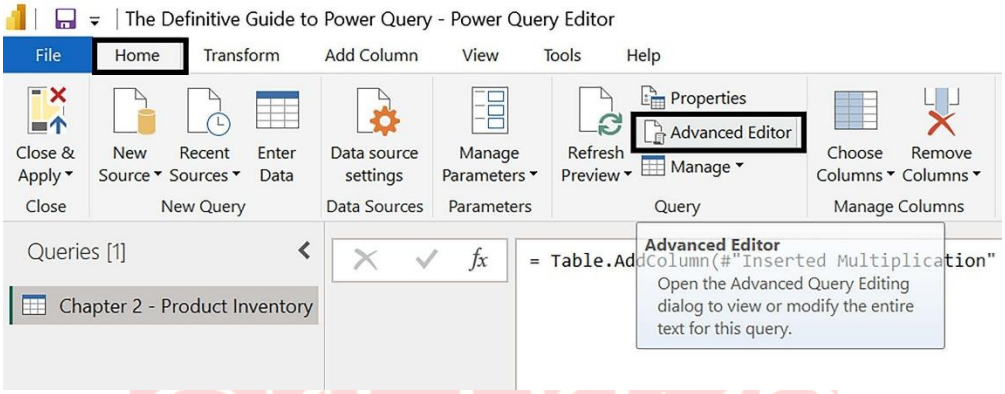
Sejauh ini, kita telah perlahan-lahan memperkenalkan cara menulis kode M kustom sambil tetap sangat bergantung pada Power Query Editor untuk menulis sebagian besar kode bagi kita. Selanjutnya, mari kita jelajahi bagaimana Anda dapat mengakses kode M sepenuhnya untuk kueri menggunakan Advanced Editor dan membuat modifikasi tanpa bergantung pada Power Query Editor untuk melakukan pekerjaan berat.

F. Menggunakan Editor Tingkat Lanjut

Sementara sebagian besar pemula akan sangat bergantung pada GUI dan Power Query Editor untuk menulis semua atau sebagian besar kode M mereka, pengguna tingkat lanjut pada akhirnya akan bermigrasi ke keinginan untuk mengakses langsung kode M, mirip dengan cara kode sumber dibuat dan diedit dalam sebagian besar bahasa pemrograman lainnya. Untungnya, Power Query Editor menyediakan antarmuka seperti itu, **Advanced Editor**. Dengan memanfaatkan Advanced Editor, kekuatan penuh bahasa M dapat digunakan

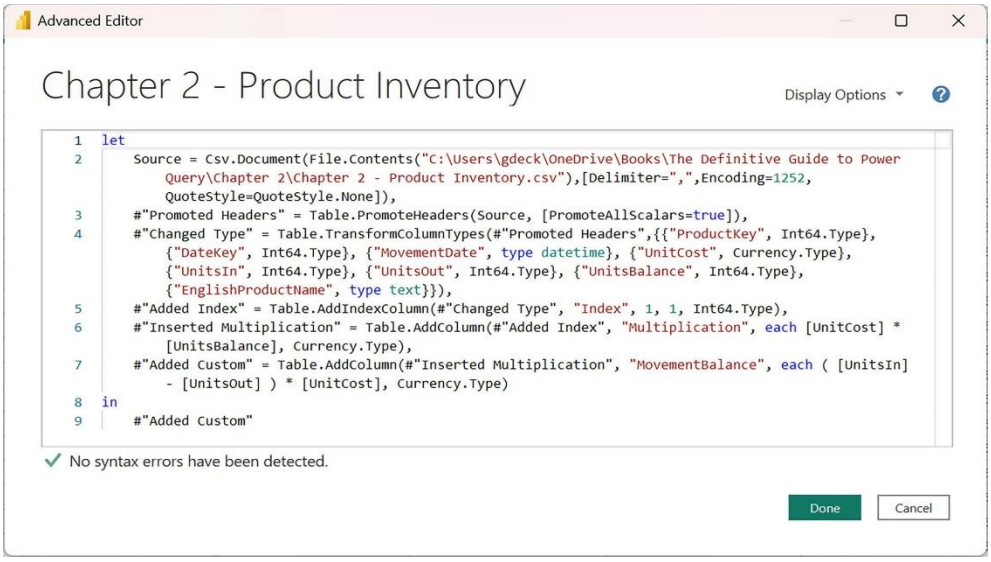
selama transformasi data dibandingkan hanya sebagian kecil bahasa yang dapat diakses menggunakan GUI.

Untuk mengakses **Advanced Editor**, klik pada tab **Home** pada pita lalu pilih **Advanced Editor** dari bagian Query, seperti yang ditunjukkan pada Gambar 2.19:



Gambar 2. 19 Mengakses Editor Lanjutan

Ini meluncurkan dialog **Advanced Editor**, seperti yang ditunjukkan pada Gambar 2.20:



Gambar 2. 20 Editor Tingkat Lanjut

Pada Gambar 2.20, seluruh kode M yang terdiri dari Gbapter 2, Bekerja dengan kueri Power Query/M ditampilkan. Semua langkah kueri terdapat dalam

ekspresi let yang diperkenalkan di Bab 1, Memperkenalkan M. Selain itu, menu tarik-turun **Display Options** telah digunakan untuk **Display line numbers** dan **Enable word wrap**.

Mengklik ikon tanda tanya (?) akan membuka peramban web ke halaman web referensi fungsi Power Query M. Namun, sumber yang sangat baik yang mungkin menyediakan informasi lebih banyak lagi adalah situs web *powerquery.how* yang dikelola oleh Rick de Groot. Misalnya, *powerquery.how* berisi list lengkap enumerasi kode M (kumpulan nilai yang mungkin dan ditetapkan untuk argumen fungsi), sedangkan informasi tersebut hampir sepenuhnya tidak ada dalam dokumentasi bahasa M resmi Microsoft.

Dalam hal situs web yang berguna di luar dokumentasi bahasa M resmi Microsoft, situs web powerqueryformatter.com adalah situs web yang berguna untuk memformat kode M secara otomatis sesuai dengan praktik terbaik. Walaupun Editor Lanjutan menyertakan fungsi pengetikan awal dan pemformatan warna, ia tidak memiliki fitur pemformatan otomatis apa pun.

Tombol pintas berikut tersedia untuk memperbesar dan memperkecil tampilan:

- **Zoom in**: *Gtrl* + *Sbiß* + =
- **Zoom out**: *Gtrl* + *Sbiß* + -

Perhatikan bahwa tombol pintas untuk memperbesar tampilan ini juga berfungsi di dalam Editor Power Query itu sendiri dan bukan hanya Editor Lanjutan.

Dengan menggunakan **Advanced Editor**, seluruh konten kueri dapat diedit untuk memungkinkan indentasi kode yang tepat demi tujuan keterbacaan, serta membersihkan nama langkah, seperti menerapkan camel case.

Seperti ditunjukkan pada Gambar 2.20, nama langkah yang mengandung spasi harus diawali dengan karakter hashtag (#) dan kemudian nama langkah diapit tanda kutip ganda.

Misalnya, seluruh isi kode **Advanced Editor** dapat diganti dengan kode berikut. Perhatikan bahwa Anda perlu memastikan bahwa jalur file sesuai dengan file di sistem lokal Anda, bukan jalur file contoh yang diberikan:

```
let
    Source =
        Csv.Document(
            File.Contents("C:\temp\Product Inventory.csv"),
            [Delimiter=",",Encoding=1252, QuoteStyle=QuoteStyle.None]
        ),
    promoteHeaders = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
    changeTypes =
        Table.TransformColumnTypes(
            promoteHeaders,
            {
                {"ProductKey", Int64.Type},
                {"DateKey", Int64.Type},
                {"MovementDate", type datetime},
                {"UnitCost", Currency.Type},
                {"UnitsIn", Int64.Type},
                {"UnitsOut", Int64.Type},
                {"UnitsBalance", Int64.Type},
                {"EnglishProductName", type text}
            }
        ),
    addIndex = Table.AddIndexColumn(changeTypes, "Index", 1, 1, Int64.Type),
    addInventoryBalanceColumn =
        Table.AddColumn(
            addIndex,
            "InventoryBalance",
            each [UnitCost] * [UnitsBalance],
            Currency.Type
        ),
    addMovementBalanceColumn =
        Table.AddColumn(
            addInventoryBalanceColumn,
            "MovementBalance",
            each ( [UnitsIn] - [UnitsOut] ) * [UnitCost],
            Currency.Type
        )
in
    addMovementBalanceColumn
```

Dengan perubahan ini, kode menjadi jauh lebih bersih dan mudah dibaca, seperti yang ditunjukkan pada Gambar 2.21:

Gambar 2. 21 Editor Lanjutan dengan perubahan kode khusus

Penting untuk dicatat bahwa jika Anda menggunakan **Advanced Editor** untuk mengganti nama langkah, Anda perlu menyesuaikan ekspresi lain yang merujuk ke langkah tersebut secara manual. Karena alasan ini, mungkin lebih baik menggunakan area **Applied Steps** dari panel **Query Settings** untuk mengganti nama langkah dengan mengklik kanan nama langkah dan memilih Ubah Nama.

Setelah Anda selesai mengedit kode M dalam Editor Lanjutan, Anda dapat menerapkan perubahan dengan mengeklik tombol **Done**. Untuk benar-benar memuat data ke dalam model Power BI atau Excel, klik tombol **Close & Apply** di paling kiri tab **Home** pada pita, seperti yang diperlihatkan pada Gambar 2.19.

Ini melengkapi penjelajahan kita terhadap Editor Lanjutan dalam pengalaman Power Query Desktop.

G. Ringkasan

Pengalaman Power Query Desktop dan Online menyediakan antarmuka utama untuk menulis kode M. Pengalaman ini sebagian besar melindungi Anda dari keharusan menulis langsung semua atau sebagian besar kode M yang terdiri dari kueri. Namun, seiring bertambahnya keahlian Anda dengan bahasa M, Anda akan mendapati diri Anda lebih sering menulis kode M secara langsung.

Dalam bab ini, kita menjelajahi Power Query menggunakan pengalaman Power Query Desktop di Power BI Desktop sebagai panduan kita. Ini termasuk ikhtisar komponen utama antarmuka Editor Power Query serta eksplorasi opsi yang mengontrol tampilan dan perilaku antarmuka ini dan cara melihat dan mengubah pengaturan sumber data. Kita juga membahas beberapa cara untuk mengubah atau menulis kode M, termasuk menggunakan bilah rumus serta saat menambahkan kolom kustom. Terakhir, kita menjelajahi Editor Lanjutan untuk melakukan pengeditan massal dan menulis kode M yang lebih canggih.

Dalam bab berikutnya dan selanjutnya, kita mempelajari pembahasan yang lebih mendalam tentang bahasa M, dimulai dengan eksplorasi tentang cara mengakses dan menggabungkan data dalam M.

H. Daftar Pustaka

1. Kasun Bandara and Rob J Hyndman and Christoph Bergmeir. (2021). MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns. arXiv:2107.13462 [stat.AP]. https://arxiv.org/abs/2107.13462.
2. Hochenbaum, J., Vallis, O., & Kejariwal, A. (2017). Automatic Anomaly Detection in the Cloud Via Statistical Learning. ArXiv, abs/1704.07706. https://arxiv.org/abs/1704.07706.

I. Penutup

1. Tes Formatif

Tabel 2. 1 Tes Formatif Bab 2

No	Soal	Bobot
1.	Apa itu Power Query dan bagaimana fungsinya dalam pengolahan data?	10
2.	Deskripsikan langkah-langkah yang diperlukan untuk mengimpor data menggunakan Power Query?	10
3.	Sebutkan dan jelaskan tiga jenis transformasi data yang dapat dilakukan dengan Power Query?	10
4.	Apa itu Query Editor dalam Power Query? Bagaimana cara menggunakannya untuk memodifikasi data?	10
5.	Bagaimana cara menerapkan filter pada data di Power Query? Berikan contoh situasi di mana filter diperlukan?	10
6.	Diskusikan metode yang dapat digunakan untuk menggabungkan beberapa sumber data dalam Power Query?	10
7.	Sebutkan beberapa fungsi yang umum digunakan dalam Power Query dan jelaskan fungsinya?	10
8.	Bagaimana cara menyimpan dan memperbarui data yang telah diproses di Power Query?	10
9.	Jelaskan bagaimana Power Query dapat diintegrasikan dengan Excel dan manfaatnya bagi pengguna?	10
10.	Berikan contoh studi kasus di mana Power Query digunakan untuk menyelesaikan masalah pengolahan data. Apa hasil yang diperoleh.	10

BAB 3

Mengakses dan Menggabungkan Data

Topik Bab

Pada bab ini, topik yang akan dibahas adalah:

- Mengakses file dan folder
- Mengambil konten web
- Menyelidiki fungsi biner
- Mengakses database dan kubus
- Bekerja dengan protokol data standar
- Mengatasi konektor tambahan
- Menggabungkan dan menggabungkan data

A. Pendahuluan

Mengingat bahwa M adalah bahasa yang digunakan untuk mengambil dan mengubah data, wajar saja jika bahasa M memiliki sejumlah besar fungsi bawaan yang ditujukan untuk mengambil data dari berbagai sistem seperti file, folder, basis data, halaman web, dan struktur data serta protokol standar seperti eXtensible Markup Language (XML), JavaScript Object Notation (JSON), dan Open Data Protocol (OData).

Faktanya, M mencakup lebih dari 100 fungsi pengambilan data standar dan menyediakan kemampuan untuk menulis ekstensi guna mengambil data dari sumber tambahan seperti sistem bisnis kustom seperti sistem manajemen hubungan pelanggan (CRM) yang dikembangkan sendiri atau sistem manajemen gudang (WMS). Pembuatan ekstensi tersebut dibahas dalam GBAPTER 16, Mengaktifkan Ekstensi. Dengan mempertimbangkan ekstensi ini, ada lebih dari 350 fungsi data yang tersedia untuk mengakses data dalam editor Power Query. Jelas, tidak ada buku yang dapat membahas fungsi individual sebanyak itu. Oleh karena itu, bab ini berupaya memberikan pembahasan luas tentang banyak fungsi pengaksesan data umum dengan contoh-contoh spesifik yang berlaku untuk sebagian besar fungsi lainnya.

Selain pengambilan data, M juga menyediakan kemampuan yang hebat dalam hal menggabungkan data dari berbagai sumber, termasuk kemampuan untuk menggabungkan (menambahkan) data bersama-sama dan menggabungkan (menggabungkan) data. Kemampuan penggabungan ini bahkan meluas ke kemampuan pencocokan fuzzy di mana nilai data dalam kolom-kolom kunci yang digabungkan serupa tetapi tidak harus identik.

Sepanjang bab ini, kita akan menjelajahi kemampuan penanganan data unik bahasa M, mengungkap seluk-beluk bekerja dengan struktur data hierarkis, basis data, dan sistem berkas. Selain itu, kita akan mempelajari dua teknik paling umum untuk transformasi data; agregasi, dan penggabungan, yang memungkinkan Anda memanfaatkan potensi penuh M untuk analisis dan

pelaporan data. Untuk informasi tambahan tentang pengoptimalan kinerja selama pengambilan dan transformasi data, lihat Bab 15, Mengoptimalkan Kinerja.

Seiring dengan kemajuan kita, Anda akan mempelajari cara mengekstrak data dari berbagai sumber dan menggabungkan data tersebut, semuanya menggunakan kemampuan serbaguna M. Secara khusus, bab ini mencakup hal-hal berikut:

- Mengakses file dan folder
- Mengambil konten web
- Menyelidiki fungsi biner
- Mengakses database dan kubus
- Bekerja dengan protokol data standar
- Mengatasi konektor tambahan
- Menggabungkan dan menggabungkan data

1. Kasus Pemantik Berfikir Kritis: Pengolahan Data Siswa

Sebuah sekolah ingin menganalisis data nilai siswa untuk mengetahui performa akademik mereka. Data nilai siswa disimpan dalam file Excel yang berisi informasi seperti nama siswa, kelas, dan nilai ujian. Tim pengajar menggunakan Power Query untuk mengolah data ini. Setelah mengimpor data nilai siswa ke dalam Power Query, tim menemukan beberapa masalah, seperti:

- Beberapa nama siswa ditulis dengan ejaan yang berbeda (misalnya, "Ahmad" dan "Ahmad S").
- Ada kolom yang berisi nilai kosong untuk beberapa siswa.
- Format nilai yang tidak konsisten (misalnya, ada yang menggunakan angka desimal dan ada yang menggunakan angka bulat).

1) Apa saja masalah yang dihadapi tim pengajar dalam pengolahan data nilai siswa? Mengapa masalah ini penting untuk diatasi?

2) Bagaimana cara tim dapat mengatasi masalah ejaan yang berbeda untuk nama siswa? Apa langkah-langkah yang dapat diambil untuk memastikan konsistensi nama?
3) Apa yang dapat dilakukan tim untuk menangani kolom yang berisi nilai kosong? Diskusikan beberapa opsi yang mungkin, seperti pengisian nilai atau penghapusan data.
4) Bagaimana cara tim dapat menyamakan format nilai yang tidak konsisten? Mengapa penting untuk memiliki format yang seragam dalam analisis data?
5) Setelah data dibersihkan, bagaimana tim dapat menggunakan hasil analisis untuk meningkatkan performa akademik siswa? Berikan contoh tindakan yang dapat diambil berdasarkan analisis tersebut.
6) Apa langkah-langkah yang dapat diambil untuk memastikan bahwa data nilai siswa selalu terjaga kualitasnya di masa depan?

B. Persyaratan Teknis

Untuk menyelesaikan tugas dalam bab ini, Anda memerlukan hal berikut:

- Power BI Desktop
- File sumber yang digunakan dalam bab ini disertakan dalam repositori GitHub untuk buku ini (https://github.com/PacktPublishing/The-Definitive-Guide-to-Power-Query-M-/tree/main/Chapter%2003)

C. Mengakses File dan Folder

Banyak pengguna bekerja dengan M dengan mengakses file dan folder di komputer lokal, drive jaringan, atau penyimpanan cloud. Sementara sebagian besar pengguna bisnis kemungkinan akan mengambil data dari file Excel atau mungkin folder file **Comma-Separated Values (CSV)**, ada banyak format file lain yang didukung oleh M termasuk yang tercantum dalam Tabel 3.1 beserta fungsi pengambilan data M yang sesuai:

Tabel 3. 1 Fungsi M untuk berbagai format file

Format File	**Fungsi M**
Azure Storage	*AzureStorage.BlobContents, AzureStorage. Blobs, AzureStorage.DataLake, AzureStorage. DataLakeContents, AzureStorage.Tables*
Binary	*File.Contents*
Excel	*Excel.Workbook, Excel.CurrentWorkbook*
Folder	*Folder.Contents, Folder.Files*
HDFS	*Hdfs.Contents,* ***Hdfs.Files***
HDInsight	*HdInsights.Containers, HdInsight.Contents, HdInsight.Files*
JSON	*Json.Document*
PDF	*Pdf.Tables*
RData	*RData.FromBinary*
Text/CSV	*Csv.Document*
XML	*Xml.Document, Xml.Tables*

Format data bisa sangat beragam dan format yang disukai telah berubah seiring waktu. Selain itu, ada format data khusus yang digunakan oleh segmen tertentu di dunia teknologi, seperti file **RData** yang digunakan saat menggunakan bahasa R untuk analisis statistik. Meningkatnya big data memperkenalkan **Hadoop Distributed File System (HDFS)**. Terakhir, meningkatnya data lake dan lakehouse telah membuat format file seperti Parquet menjadi populer.

Mari kita lihat lebih dekat berbagai format file ini, hanya saja daripada mengerjakannya secara alfabetis seperti dalam tabel, pertama-tama kita akan melihat apa yang bisa dibilang sebagai fungsi inti akses file, *File.Contents*.

1. File.Contents

Fungsi *File.Contents* merupakan fungsi pengaksesan data dasar untuk file dan folder. Jarang digunakan sendiri, output dari fungsi *File.Contents* umumnya berfungsi sebagai input untuk parameter pertama dari banyak fungsi

pengaksesan file lainnya seperti *Csv.Document, Excel.Workbook, Json.Document, Xml.Document,* dan *Xml.Tables*.

Fungsi *File.Contents* mengambil dua parameter, parameter konten yang diperlukan sebagai parameter pertamanya dan record opsi opsional sebagai parameter keduanya. Jika Anda familier dengan bahasa pemrograman lain, konsep parameter yang digunakan dengan fungsi sama persis. Jika Anda tidak familier dengan fungsi dan parameter, rujuk Bab 9, Parameter dan Fungsi Kustom. Selain itu, record dibahas dalam Bab 6, Nilai Terstruktur.

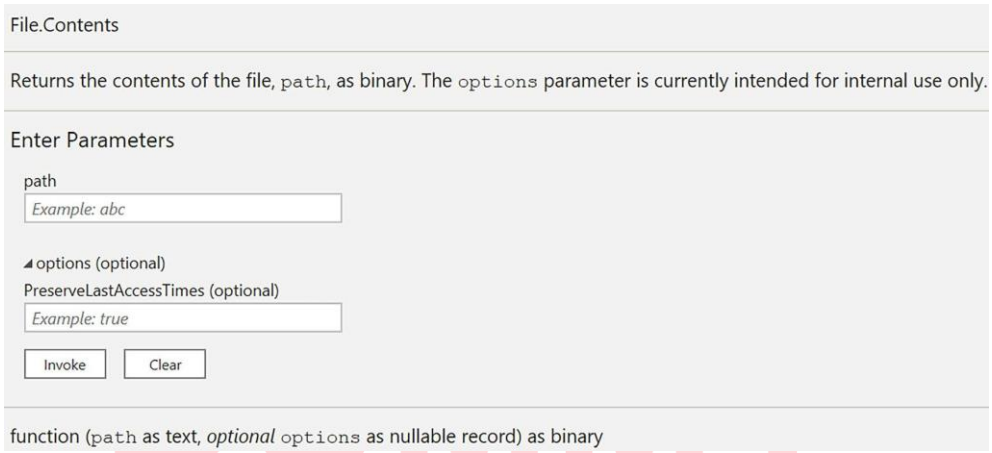
Output fungsi adalah konten file yang ditentukan dalam parameter *content* yang diformat sebagai biner (non-teks). Parameter konten menentukan lokasi file, seperti *C:\accounting\invoices.csv*. Parameter kedua adalah record yang berisi opsi untuk memproses file.

Parameter opsi kedua umum untuk banyak fungsi pengaksesan data. Namun, opsi yang tersedia terkenal tidak terdokumentasi dengan baik dalam dokumentasi M resmi yang tersedia daring, dan opsi yang valid bervariasi bergantung pada fungsi pengaksesan data tertentu yang digunakan. Dalam kasus *File*. Contents, ada opsi valid yang diketahui, *PreserveLastAccessTimes*, yang bisa benar atau salah.

Ada cara untuk mengungkap opsi yang ditawarkan ini untuk fungsi pengaksesan data. Di dalam editor Power Query, cukup buat kueri seperti berikut:

```
let
    Source = File.Contents
in
    Source
```

Seperti yang ditunjukkan, cukup tentukan fungsi sebagai sumber data. Di dalam editor Power Query, dokumentasi dikembalikan terkait fungsi tersebut:



Gambar 3. 1 Dokumentasi File.Contents

Mekanisme di balik penyediaan dokumentasi ini dibahas dalam Bab 7, Konseptualisasi M. Perlu dicatat bahwa saat menggunakan antarmuka Power Query untuk fungsi yang dibahas dalam bagian ini, Mengakses file dan folder, antarmuka mengasumsikan file lokal atau jaringan sedang diakses dan dengan demikian menggunakan *File.Contents* sebagai fungsi untuk mengembalikan konten biner file. Namun, *File.Contents* tidak diperlukan sebagai fungsi dasar untuk mengembalikan konten biner ini. Faktanya, fungsi apa pun yang mengembalikan konten biner file dapat digunakan sebagai gantinya. Misalnya, jika file tersebut ada di situs web, fungsi *Web.Contents* dapat digunakan sebagai ganti *File.Contents* karena *Web.Contents* mengembalikan konten biner file dari URI. Untuk informasi selengkapnya tentang *Web.Contents*, lihat bagian Mengambil Konten Web dalam bab ini.

Sekarang mari kita lihat penggunaan File.Contents secara praktis pada file **Comma-Separated Values (CSV)**.

2. Text/CSV

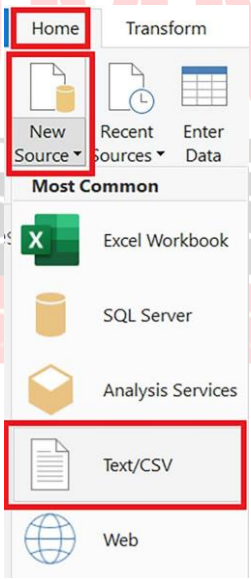
File teks dan CSV merupakan jenis file umum yang digunakan dengan data dan memiliki dukungan yang hampir universal sebagai opsi ekspor dalam sistem perangkat lunak yang menangani data. File tersebut sering kali

merupakan output dari pengeksporan data dari gudang data atau penyimpanan data terpusat lainnya.

Sesuai namanya, file teks hanyalah teks dengan kolom atau bidang yang paling sering dipisahkan oleh tab atau koma. File yang dibatasi tab dan dibatasi koma merupakan opsi ekspor data umum untuk banyak sistem. Untuk mempelajari file teks/CSV secara lebih rinci, kita akan menggunakan file Avocado Prices.csv. File ini disertakan dalam repositori GitHub untuk buku ini. Unduh file ini dari repositori GitHub dan simpan file ini dalam folder di komputer lokal Anda.

Data Harga Alpukat merupakan sumber data publik umum yang digunakan untuk menunjukkan variasi harga dan volume penjualan dari waktu ke waktu (analisis deret waktu). File tersebut berisi harga rata-rata dan total volume alpukat yang dijual menurut kota, negara bagian, dan wilayah selama periode sekitar 3 tahun. Untuk membuat kueri file CSV, lakukan langkah-langkah berikut:

1) Di editor Power Query, klik tombol **New Source** pada tab **Home** di pita dan pilih **Teks/CSV**:



Gambar 3. 2 Membuat kueri Teks/CSV

2) Gunakan jendela File Explorer untuk menavigasi dan memilih file, lalu tekan tombol Buka.
3) Dalam dialog Avocado Prices.csv, tekan tombol OK.

Setelah melakukan langkah-langkah ini, langkah Sumber dapat dilihat di **Advanced Editor** dan tampak seperti berikut:

```
    Source = Csv.Document(File.Contents("C:\data\Avocado Prices.csv"),
[Delimiter=",", Columns=14, Encoding=1252, QuoteStyle=QuoteStyle.None]),
```

Seperti yang dapat dilihat dalam kode, fungsi *File.Contents* digunakan untuk mengambil konten file *C:\data\Avocado Prices.csv* sebagai data biner. Mengakses dan mengambil data sumber merupakan langkah yang jelas dan penting dalam analisis data. Terlepas dari analisis yang akan dilakukan, jika seseorang tidak dapat mencerna atau mengakses data yang diperlukan, analisis tidak dapat dilanjutkan.

Output dari fungsi *File.Contents* digunakan sebagai parameter pertama untuk fungsi *Csv.Document*. Parameter kedua untuk fungsi *Csv.Document* adalah record opsi dengan opsi yang ditetapkan untuk hal berikut:

- *Delimiter*: Menentukan bahwa kolom dalam data dipisahkan oleh koma (,).
- *Columns*: Menentukan bahwa ada 14 kolom dalam data.
- *Encoding*: Menentukan halaman kode 1252 (pengodean karakter Windows). Halaman kode hanyalah spesifikasi tentang bagaimana karakter yang dapat dicetak dan karakter yang tidak dapat dicetak (seperti karakter kontrol seperti carriage return dan line feed) dikaitkan dengan nomor unik.
- *QuoteStyle*: Menentukan bahwa jeda baris diperlakukan sebagai akhir baris saat ini terlepas dari apakah jeda baris terjadi dalam nilai yang dikutip.

Parameter kedua dari fungsi *Csv.Document* sebenarnya dapat berupa null, tipe tabel, list nama kolom, jumlah kolom, atau, seperti pada kode yang

diberikan, record opsi. Fungsi *Csv.Document* menentukan cara memperlakukan parameter kedua berdasarkan tipe nilai, seperti *number*, *list*, atau *record*. Kita akan melihat record opsi yang disediakan oleh parameter kedua ini sebentar lagi, tetapi pertama-tama, mari kita hitung parameter opsional tambahan yang tersedia. Ada tiga parameter tambahan tersebut.

Parameter ketiga adalah parameter *delimiter*. Defaultnya adalah karakter koma (,). Nilai delimiter dari string teks kosong ("") menunjukkan bahwa baris harus dibagi menjadi kolom berdasarkan karakter spasi berurutan.

Untuk menggunakan karakter khusus, seperti karakter tab, Anda harus menggunakan M escape sequence. Escape sequence karakter diformat sebagai #(<*character*>). Beberapa karakter memiliki nama khusus seperti tab (karakter tab), *cr* (carriage return), dan *lf* (line feed). Misalnya, untuk file yang dibatasi tab, tentukan #(*tab*) sebagai delimiter. Namun, karakter apa pun dapat digunakan sebagai delimiter dengan menentukan kode hex Unicode empat digit karakter seperti #(2605) untuk karakter bintang hitam (U+2605).

Dokumentasi resmi untuk fungsi ini mengklaim bahwa Anda juga dapat memberikan list karakter. Namun, menentukan { ",", "|" } akan mengembalikan kesalahan bahwa nilai bertipe List tidak dapat diubah menjadi tipe Text. Jadi, artinya, alih-alih pemisah satu karakter seperti koma (,), beberapa karakter dapat digunakan sebagai pemisah, seperti #(*tab*)#(2605) dalam kasus tab dan karakter bintang hitam digunakan sebagai pemisah. Ini memungkinkan fleksibilitas yang cukup untuk mendukung format data yang kompleks.

Parameter keempat adalah parameter *extraValues*. Parameter ini menentukan bagaimana fungsi *Csv.Document* harus menangani kolom tambahan yang tidak diharapkan. Parameter ini menerima nilai sesuai dengan enumerasi *ExtraValues.Type*. Secara khusus, nilai yang diizinkan adalah sebagai berikut:

Tabel 3. 2 Nilai yang diizinkan untuk parameter extraValues

Friendly Name	Value	Comments
extraValues.List	0	Mengembalikan kolom tambahan sebagai list
extraValues.Error	1	Menimbulkan kesalahan
extraValues.Ignore	2	Mengabaikan kolom tambahan

Nilai default adalah *ExtraValues.Ignore*. Ini berarti bahwa jika kolom kelima belas ditambahkan ke file *Avocado Prices.csv*, karena catatan opsi kita menyertakan penetapan 14 untuk opsi **column**, kolom kelima belas tambahan ini akan diabaikan. Ini dapat membantu mencegah kesalahan selama penyerapan data jika terjadi kesalahan saat membuat data. Sebaliknya, seseorang mungkin ingin diberi tahu tentang perubahan apa pun pada format data yang mendasarinya dan dengan demikian menggunakan *ExtraValues.Error* untuk memastikan bahwa setiap perubahan format data harus ditangani secara khusus.

Atau, mungkin disarankan untuk menghilangkan opsi **column** sama sekali untuk membantu mengamankan kueri Anda di masa mendatang. Dengan tidak menetapkan jumlah kolom, jika kolom kelima belas muncul dalam data, kueri akan memproses file CSV dan menyertakan kolom kelima belas ini.

Parameter kelima dan terakhir adalah parameter pengodean. Parameter ini menetapkan jenis pengodean file. Jenis pengodean teks umum disediakan oleh enumerasi *TextEncoding.Type*, termasuk:

Tabel 3. 3 Jenis pengkodean teks

Friendly Name	Value	Comments
TextEncoding.Utf16, TextEncoding.Unicode	1200	Little endian binary form (UTF16)
TextEncoding.Unicode	1200	Little endian binary form (UTF16)
TextEncoding.BigEndianUnicode	1201	Big endian binary form (UTF16)
TextEncoding.Windows	1252	Windows binary form
TextEncoding.Ascii	20127	ASCII binary form
TextEncoding.Utf8	65001	UTF8 binary form

Akan tetapi, ada banyak nilai penyandian tambahan yang dapat diberikan. Faktanya, dialog Avocado Prices.csv menyediakan opsi **File Origin** sebagai bagian dari dialog. Dengan menggunakan menu tarik-turun, seseorang dapat melihat nilai penyandian tambahan dan deskripsi untuk penyandian tersebut. Misalnya, nilai 10017 adalah nilai penyandian untuk komputer Macintosh Ukraina.

Sekarang mari kita kembali ke record yang digunakan untuk parameter kedua dalam contoh kita. Untuk menggunakan record opsi sebagai parameter kedua, parameter ketiga, keempat, dan kelima harus null atau tidak disertakan dalam pemanggilan fungsi. Jika demikian, opsi berikut dapat disertakan dalam record:

- *Delimiter*: Berfungsi persis seperti parameter Delimiter ketiga untuk *Csv.Document*.
- *Columns*: Berfungsi persis seperti parameter Columns kedua untuk *Csv.Document* selain itu tidak dapat menjadi record.
- *Encoding*: Berfungsi persis seperti parameter kelima untuk *Csv.Document*.
- *CsvStyle*: Mempengaruhi apakah tanda kutip dalam kolom hanya signifikan segera setelah pembatas, atau selalu signifikan. Menerima enumerasi *CsvStyle.Type*. Enumerasi *CsvStyle.Type* mencakup yang berikut:

Tabel 3. 4 Enumerasi CsvStyle

Friendly Name	Value	Comments
`CsvStyle.QuoteAfterDelimiter`	`0`	Kutipan hanya penting jika diawali langsung oleh pemisah. Ini adalah aturan baku.
`CsvStyle.QuoteAlways`	`1`	Kutipan selalu penting.

- *QuoteStyle*: Menentukan bagaimana kutipan diperlakukan dalam string (sedangkan *CsvStyle* hanya memengaruhi kutipan yang langsung

mengikuti pembatas). Menerima enumerasi *QuoteStyle.Type*. Enumerasi *QuoteStyle.Type* mencakup yang berikut:

Tabel 3. 5 Pencacahan QuoteStyle

Friendly Name	Value	Comments
`QuoteStyle.None`	`0`	Kutipan diabaikan. Ini adalah default.
`QuoteStyle.Csv`	`1`	Tanda kutip adalah awal dari rangkaian tanda kutip. Dua tanda kutip mewakili tanda kutip bertingkat.

Untuk contoh spesifik, lihat tautan berikut: https://powerquery.how/quotestyle-none/.

Mempertimbangkan semua yang telah dibahas sejauh ini, ini berarti bahwa alih-alih menggunakan record opsi untuk baris *Source* dalam contoh file *Avocado Prices.csv* kita, kita dapat menggunakan yang berikut ini dan memperoleh hasil yang serupa:

```
    Source = Csv.Document(File.Contents("C:\data\Avocado Prices.csv"),
14, ",", null, TextEncoding.Windows),
```

Namun, ekspresi Sumber alternatif ini hanya berfungsi karena tidak ada koma dalam nilai kolom kita. Jika ada, baris tersebut akan diurai secara tidak benar. Jadi, sebaiknya selalu gunakan record opsi untuk file teks/CSV sebagai opsi. Mari beralih ke file Excel.

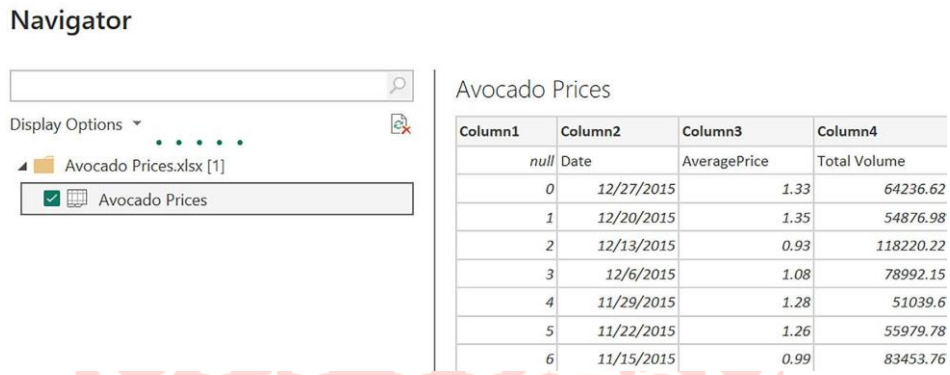
3. *Excel*

Bahasa M memiliki dua fungsi akses data standar untuk Excel, *Excel.Workbook* dan *Excel.CurrentWorkbook*. Fungsi-fungsi ini penting untuk mengotomatiskan ekstraksi data keuangan, metrik penjualan, atau data lainnya dari file Excel, yang merupakan salah satu format file paling populer yang digunakan oleh pengguna bisnis.

Mari kita lihat fungsi Excel.Workbook terlebih dahulu. Untuk menyelidiki fungsi ini, lakukan langkah-langkah berikut:

1) Buka file *Avocado Prices.csv* di Microsoft Excel dan simpan file tersebut sebagai file *.xlsx*, *Avocado Prices.xlsx*.

2) Di editor Power Query, klik tombol **New Source** pada tab **Home** di pita dan pilih **Excel Workbook**.
3) Gunakan jendela **File Explorer** untuk menavigasi dan memilih file yang dibuat pada langkah 1, lalu tekan tombol **Open**.
4) Di dialog **Navigator**, pilih lembar Avocado Prices, lalu tekan tombol **OK**:



Gambar 3. 3 Mengimpor lembar dari file Excel

Setelah melakukan langkah-langkah ini, kode M berikut dibuat untuk dua ekspresi pertama dalam pernyataan let, yang dapat dilihat di **Advanced Editor**:

```
Source = Excel.Workbook(File.Contents("C:\Users\gdeck\OneDrive\Books\The Definitive Guide to Power Query\Chapter 7\Avocado Prices.xlsx"), null, true),

#"Avocado Prices_Sheet" = Source{[Item="Avocado Prices",Kind="Sheet"]} [Data],
```

Mari kita lihat baris Source terlebih dahulu. Seperti yang ditunjukkan, fungsi *Excel.Workbook* memiliki tiga parameter. Mirip dengan *Csv.Document*, parameter pertama adalah output dari fungsi *File.Contents*, yang mengakses file Excel dan mengembalikan data biner.

Parameter kedua adalah parameter useHeaders. Parameter ini dapat berupa *null*, *true*, atau *false* dan defaultnya adalah false. Atau, parameter ini juga dapat berupa record opsi selama parameter ketiga adalah *null*. Menentukan true menyebabkan fungsi *Excel.Workbook* menggunakan baris

pertama tabel yang dikembalikan sebagai nama tajuk kolom. Dalam contoh kita, Power Query tidak secara otomatis mengenali bahwa baris pertama data terdiri dari nama kolom (karena nilai *null* di kolom pertama baris pertama). Jadi, kita dapat mengubahnya menjadi true atau kita dapat mempromosikan tajuk dalam langkah kueri berikutnya (*Table.PromoteHeaders*).

Parameter ketiga adalah parameter *delayTypes*. Parameter ini juga dapat berupa null, true, atau false dan defaultnya adalah *false*. Menetapkan true menyebabkan parameter *Excel.Workbook* tidak menetapkan tipe ke kolom. Banyak analis data mempromosikan praktik terbaik untuk menetapkan tipe data secara manual dan tidak otomatis. Namun, penetapan tipe data secara otomatis dapat menghemat waktu bagi pengguna bisnis.

Seperti yang disebutkan, parameter kedua, *useHeaders*, dapat juga berupa record opsi selama parameter ketiga (*delayTypes*) ditetapkan ke null. Untuk tinjauan record opsi, lihat subbagian *File.Gontents* dan *Text/GSV* dalam bagian ini. Dalam kasus ini, record opsi mendukung opsi berikut:

- *UseHeaders*: Beroperasi persis seperti parameter *useHeaders* selain tidak dapat menjadi record opsi.
- *DelayTypes*: Beroperasi persis seperti parameter *delayTypes*.
- *InferSheetDimensions*: Nilai ini dapat berupa *null, true*, atau *false* dan nilai default-nya adalah *false*. Opsi ini hanya didukung oleh format file Open XML modern dan bukan format file Excel lama. Menentukan true menyebabkan fungsi *Excel.Workbook* mengabaikan metadata dimensi yang disertakan dalam file Excel dan menyimpulkan area lembar kerja dengan membaca lembar kerja tersebut.

Sekarang mari kita lihat baris kedua, #"*Avocado Prices_Sheet*". Ini adalah langkah Navigator dalam kueri kita yang ditunjukkan pada Gambar 3.3. Ekspresi ini merujuk pada ekspresi *Source*, mengakses tabel yang dikembalikan sebagai list record dan mengambil record dengan item *Avocado*

Prices dan sejenis *Sheet*. Terakhir, bidang Data dari record ini diambil sebagai keluaran tabel.

Sebagai percobaan, ubah ekspresi setelah kata kunci in dalam *query* yang dihasilkan menjadi *Source*. Tabel satu baris dikembalikan dengan tajuk kolom *Name*, *Data*, *Item*, *Kind*, dan *Hidden*. Ini adalah tabel sebenarnya yang dikembalikan oleh fungsi *Excel.Workbook*. Jika beberapa lembar, tabel, rentang bernama, dan/atau array dinamis disertakan dalam file Excel, maka beberapa baris untuk setiap item akan muncul dalam tabel ini. Informasi ini digunakan oleh editor Power Query untuk membangun jendela **Navigator** yang terlihat pada Gambar 3.3 untuk menunjukkan bagaimana eksperimen ini berhubungan dengan apa yang terjadi dalam editor Power Query, yang merupakan cara Anda memilih data secara visual.

Karena berkas Excel kita hanya memuat satu lembar dan tidak ada tabel atau rentang bernama, selanjutnya coba alihkan ekspresi setelah kata kunci in ke *Source{[]}*. Di sini, baris tunggal dalam tabel dikembalikan sebagai record. Jika ada beberapa baris, notasi ini akan mengembalikan kesalahan. Untuk informasi lebih lanjut tentang notasi ini dan pemilihan item dalam tabel, lihat Bab 6, Nilai Terstruktur.

Selanjutnya mari kita lihat fungsi *Excel.CurrentWorkbook*. Fungsi ini digunakan dalam Power Query untuk Excel dan merujuk ke file Excel yang sedang dibuka. Dengan demikian, fungsi ini tidak mengambil parameter apa pun. Tidak seperti fungsi *Excel.Workbook*, fungsi ini tidak mengembalikan lembar kerja, melainkan hanya tabel, rentang bernama, dan array dinamis. Selain itu, untuk setiap item, dua kolom dikembalikan, Konten dan Nama. Item diakses dengan cara yang sama seperti fungsi *Excel.Workbook*, seperti:

```
Source = Excel.CurrentWorkbook(){[Name="Table2"]}[Content])
```

Setelah membahas file Excel, selanjutnya mari kita bahas cara mengimpor beberapa file dari satu folder, situasi yang sering muncul saat data diekspor secara berkala dari sistem sumber setiap hari, minggu, atau bulan.

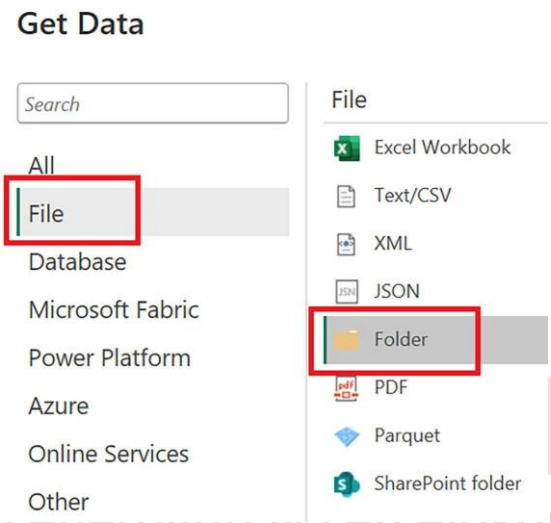
4. *Folder*

Selain mengakses file tunggal, dua fungsi disertakan untuk menangani folder sistem file yang berisi sejumlah file, *Folder.Contents* dan *Folder.Files*. Kedua fungsi tersebut sangat mirip, dengan mengambil jalur file teks sebagai parameter pertama dan catatan opsi sebagai parameter kedua. Mirip dengan *File.Contents*, catatan opsi untuk kedua fungsi mendukung satu opsi, *PreserveLastAccessTimes*, yang dapat berupa true atau false.

Kedua fungsi tersebut hanya berbeda dalam output-nya. Fungsi *Folder.Contents* mengembalikan tabel yang terdiri dari baris untuk setiap folder dan file dalam jalur folder yang ditentukan sementara *File.Contents* hanya mengembalikan baris untuk setiap file. Dengan demikian, keduanya akan mengembalikan tabel informasi yang sama persis jika tidak ada sub-folder dalam jalur folder yang ditentukan.

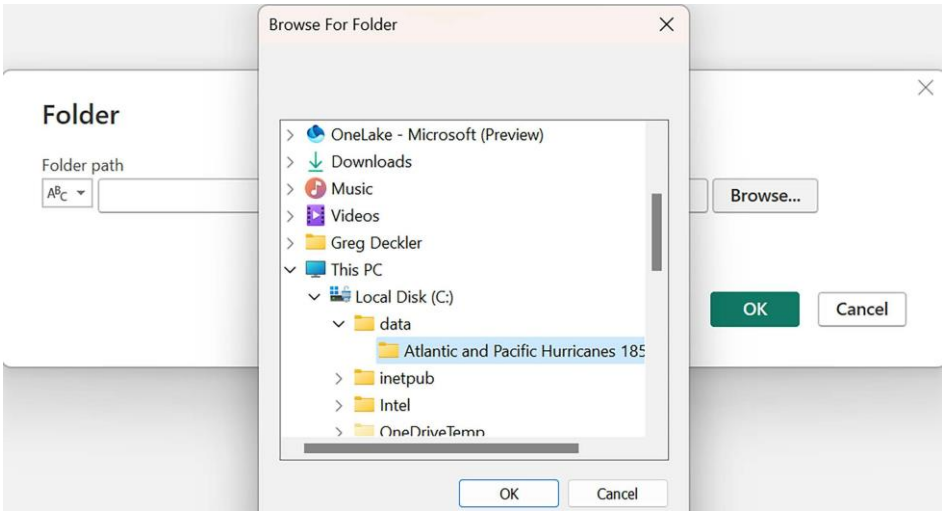
Untuk mengeksplorasi fungsi-fungsi ini secara lebih mendalam, kita akan menggunakan file Atlantic and Pacific Hurricanes 1851-2014.zip yang disertakan dalam repositori GitHub untuk buku ini. Ikuti langkah-langkah berikut:

1) Unduh file Atlantic and Pacific Hurricanes 1851-2014.zip dan ekstrak keempat file yang disertakan dalam file .zip ke satu folder di komputer lokal Anda. Ada dua file CSV, atlantic.csv dan pacific.csv, beserta dua file PDF, atlantic.pdf dan pacific.pdf.
2) Di editor Power Query, klik tombol **New Source** pada tab **Home** pada pita dan pilih **More**….
3) Dalam dialog **Get Data**, pilih **File** di panel sebelah kiri, pilih **Folder** di list sebelah kanan, lalu tekan tombol **Connect**.



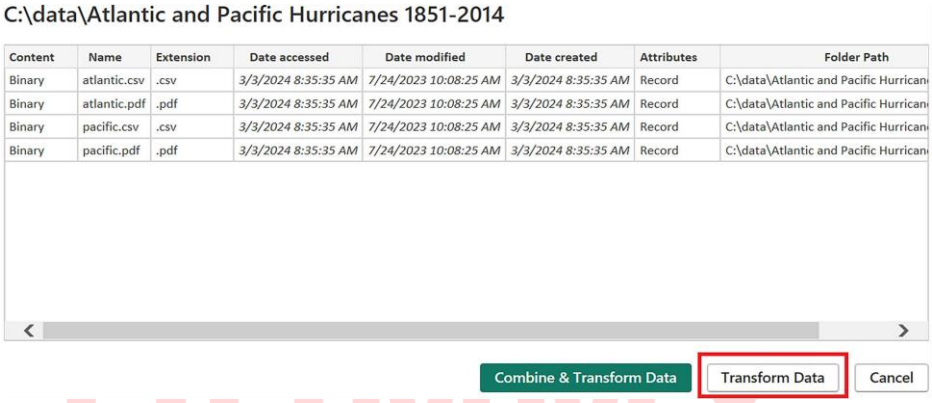
Gambar 3. 4 Dialog Dapatkan Data

4) Pada kotak dialog **Folder**, klik tombol **Browse**…
5) Menggunakan dialog **Browse For Folder**, navigasikan dan pilih folder tempat Anda mengekstrak file pada langkah 1, lalu tekan tombol **OK**.



Gambar 3. 5 Browse for folder

6) Kembali ke dialog **Folder**, tekan tombol **OK**.
7) Pada dialog berikutnya, pratinjau keempat file akan ditampilkan. Klik tombol **Transform Data**.



Gambar 3. 6 Files preview

Melakukan langkah-langkah ini menghasilkan kode M yang mirip dengan berikut ini yang dapat dilihat di **Advanced Editor**:

```
let
    Source = Folder.Files("C:\data\Atlantic and Pacific Hurricanes 1851-2014")
in
    Source
```

Kode ini cukup mudah dipahami dan menghasilkan tabel dengan kolom-kolom berikut: *Content*, *Name*, *Extension*, *Date accessed*, *Date modified*, *Date created*, *Attributes*, dan *Folder Path*. Kolom Konten menyimpan konten setiap file sebagai data biner dan kolom Atribut menyimpan record. Kolom lainnya berupa teks kecuali tiga kolom Tanggal, yang merupakan nilai datetime.

Sekarang buat folder baru di folder tempat Anda mengekstrak keempat file tersebut, lalu segarkan data Anda. Perhatikan bahwa tidak ada baris tambahan yang ditampilkan. Selanjutnya, edit baris *Source* kueri untuk menggunakan fungsi *Folder.Contents* alih-alih fungsi *Folder.Files*. Perhatikan bahwa baris tambahan ditambahkan ke tabel yang dikembalikan. Baris ini untuk folder baru yang dibuat dan memiliki nilai Tabel di kolom Konten dan tidak ada nilai di kolom *Extension*.

Di baris untuk file *atlantic.csv*, klik tautan **Binary** di kolom *Content*. Banyak langkah tambahan ditambahkan ke kueri dan hasilnya adalah tabel yang diimpor dari file CSV. Jika kita perhatikan lebih teliti kedua ekspresi

tersebut tepat setelah ekspresi Sumber, kita melihat kode yang mirip dengan kode berikut di *Advanced Editor* (beberapa langkah mungkin memiliki nama yang berbeda, tergantung pada lokasi pasti folder di komputer Anda):

```
    #"atlantic csv" = Source{[Name="atlantic.csv"]}[Content],
    #"Imported CSV" = Csv.Document(#"atlantic csv",[Delimiter=",", Columns=22,
Encoding=1252, QuoteStyle=QuoteStyle.None]),
```

Dalam ekspresi #"*atlantic.csv*", kita melihat bahwa data yang diinginkan diakses dengan cara yang sama seperti bagaimana lembar data tertentu diakses dari output yang dihasilkan oleh fungsi *Excel.Workbook* (lihat bagian Excel). Tabel diakses sebagai list record yang menentukan record tertentu menggunakan kolom Nama. Dari pilihan ini, kolom Konten dikembalikan. Ekspresi berikutnya, #"*Imported CSV*" menggunakan output dari record #"*atlantic.csv*" sebagai parameter pertama fungsi *Csv.Document* (lihat bagian *Teks/GSV*).

Seperti yang Anda lihat, mengakses data file individual menggunakan fungsi *Folder.Files* atau *Folder.Contents* cukup mudah. Namun, skenario umum adalah memproses dan menggabungkan beberapa *file* dalam *folder* yang semuanya memiliki format yang sama seperti laporan keuangan bulanan atau triwulanan atau laporan harian tentang status inventaris. Untuk menjelajahi opsi ini, buat kueri baru dengan melakukan langkah yang sama seperti sebelumnya, kecuali kali ini, untuk langkah 7 klik tombol **Combine & Transform Data** alih-alih **Transform Data**.

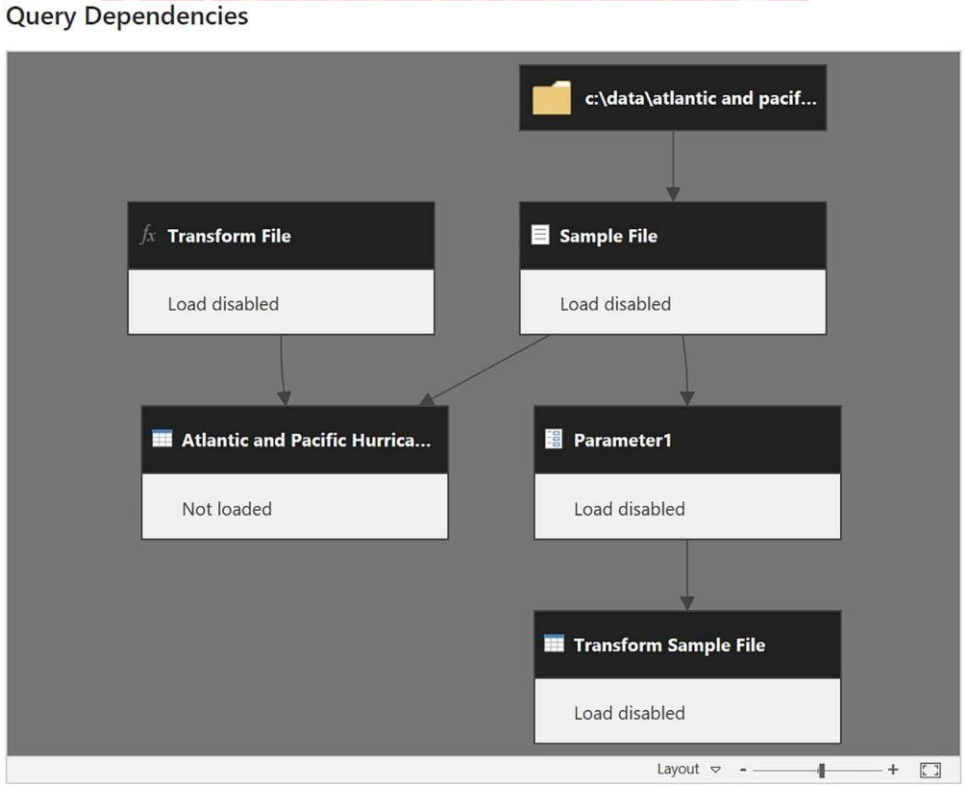
Dialog Gabungkan File ditampilkan. Menu dropdown **Sample File** secara default menggunakan file pertama, tetapi menggunakan menu tarik-turun tersebut, file tertentu dapat ditentukan. Perhatikan juga kotak centang **Skip files with errors**. Namun, untuk saat ini, cukup tekan tombol **OK**.

Melakukan langkah-langkah ini menghasilkan lebih dari sekadar satu kueri. Bahkan, selain kueri utama, grup kueri dibuat dengan nama sesuai nama folder sumber. Di dalam grup kueri ini terdapat grup kueri yang disebut **Helper Queries** dan kueri **Transform Sample File**. Dengan memperluas grup **Helper**

Queries, kita melihat parameter, **Parameter1**, kueri lain, **Sample File**, dan fungsi, **Transform File**:

Gambar 3. 7 Kueri, fungsi, dan parameter yang dibuat oleh Combine & Transform Data

Perhatikan bahwa semua kueri dan parameter ini dicetak miring dalam editor Power Query. Ini berarti tidak ada satu pun yang dimuat ke dalam model data. Ketergantungan antara berbagai kueri ini adalah sebagai berikut:



Gambar 3. 8 Ketergantungan kueri untuk kueri Folder

Mari kita telusuri kueri yang dibuat lebih lanjut dengan memulai dengan kueri *Sample File*. Kueri *Sample File* berisi kode berikut:

```
let
    Source = Folder.Files("C:\data\Atlantic and Pacific Hurricanes 1851-2014"),
    Navigation1 = Source{0}[Content]
in
    Navigation1
```

Seperti yang dapat Anda lihat, kueri ini memiliki kode yang mirip dengan yang telah kita lihat sebelumnya. Ekspresi Source menggunakan *Folder.Files* untuk mengembalikan tabel file sebagai baris. Ekspresi Navigation1 mengakses konten untuk salah satu file ini, file pertama dalam tabel diakses sebagai *list {0}*. Mengubah *0* menjadi *1* akan mengakses *file* kedua dan seterusnya. Atau, jika kita telah memilih file tertentu sebagai bagian dari dialog **Combine Files** menggunakan menu dropdown **Sample File**, ekspresi tersebut akan terlihat mirip dengan ekspresi #"*atlantic csv*" dalam contoh kode sebelumnya.

Selanjutnya mari kita lihat parameter *Parameter1* (*Sample File*). Parameter ini hanya ditetapkan ke *Sample File*, referensi ke kueri *Sample File*.

Fungsi *Transform File* ditautkan ke kueri *Transform Sample File*. Ini berarti bahwa setiap perubahan pada kueri *Transform Sample File* secara otomatis tercermin dalam fungsi *Transform File*. Bahkan, saat membuka fungsi *Transform File* di **Advanced Editor**, Anda akan menerima peringatan tentang hal ini. Fungsi *Transform File* berisi kode M berikut:

```
let
    Source = (Parameter1) => let
        Source = Csv.Document(Parameter1,[Delimiter=",", Columns=22,
Encoding=1252, QuoteStyle=QuoteStyle.None]),
        #"Promoted Headers" = Table.PromoteHeaders(Source,
[PromoteAllScalars=true])
    in
        #"Promoted Headers"
in
    Source
```

Definisi fungsi ini menggunakan satu parameter, dengan nilai default adalah *Parameter1*. *Parameter1* adalah referensi ke parameter Power Query *Parameter1*, yang saat ini ditetapkan ke *File Sampel*. Dengan menentukan nilai default, ini memungkinkan fungsi untuk beroperasi dan mengembalikan tabel nilai untuk tujuan pratinjau tetapi memungkinkan fungsi untuk juga beroperasi pada data yang diteruskan ke fungsi melalui argumen.

Kode M dalam ekspresi let kedua sebenarnya mencerminkan kode M dalam kueri *Transform Sample File* seperti yang dapat Anda lihat jika Anda membuka kueri *Transform Sample File* di **Advanced Editor**:

```
let
    Source = Csv.Document(Parameter1,[Delimiter=",", Columns=22, Encoding=1252,
QuoteStyle=QuoteStyle.None]),
    #"Promoted Headers" = Table.PromoteHeaders(Source,
[PromoteAllScalars=true])
in
    #"Promoted Headers"
```

Baris Sumber merujuk ke Parameter1, yang saat ini ditetapkan untuk merujuk ke kueri *Sample File* dan menggunakan fungsi *Csv.Document* untuk mengambil data. Ini karena berkas pertama dalam folder tersebut adalah berkas CSV. Jika kita menentukan salah satu berkas PDF, fungsi akses data akan berbeda. Kueri tersebut juga berisi satu langkah transformasi menggunakan fungsi *Table.PromoteHeaders*. Efek bersihnya adalah Anda dapat mengubah kueri Transform Berkas Contoh untuk melakukan transformasi data tambahan, seperti menghapus dan mengganti nama kolom. Transformasi data ini tercermin dalam fungsi *Transform* Berkas yang diperbarui.

Sekarang mari kita lihat kueri utama yang dibuat. Di **Advanced Editor**, kueri ini menyertakan tiga ekspresi berikut tepat setelah pernyataan *let*:

```
    Source = Folder.Files("C:\data\Atlantic and Pacific Hurricanes 1851-2014"),
    #"Filtered Hidden Files1" = Table.SelectRows(Source, each
[Attributes]?[Hidden]? <> true),
    #"Invoke Custom Function1" = Table.AddColumn(#"Filtered Hidden Files1",
"Transform File", each #"Transform File"([Content])),
```

Ekspresi #"*Invoke Custom Function1*" menggunakan fungsi *Table.AddColumn* untuk menambahkan kolom bernama *Transform File* yang isinya adalah output dari pemanggilan fungsi *Transform File* dengan kolom Konten untuk setiap baris. Akibatnya, ini mengembalikan baris dengan kolom, *Transform File*, yang isinya adalah output dari fungsi *Transform File* dengan kolom *Content* untuk baris yang diteruskan sebagai parameter, yang berarti bahwa pada akhirnya, transformasi dari kueri *Transform Sample File* diterapkan ke konten setiap file yang dikembalikan dari ekspresi *Source*, yang menggunakan fungsi *Folder.Files* untuk menghitung file dalam folder.

Dua ekspresi berikutnya (tidak ditampilkan dalam contoh kode) dalam kueri utama mengganti nama dan menghapus kolom sehingga hanya kolom *Source.Name* dan kolom *Transform File* yang tersisa. Ekspresi ini masing-masing menggunakan *Table.RenameColumns* dan *Table.SelectColumns*. Ekspresi berikutnya kemudian memperluas semua baris di kolom *Transform File* menggunakan fungsi *Table.ExpandTableColumns*.

Sekarang kita memiliki konten setiap berkas yang diubah oleh fungsi *Transform Files* yang dimuat ke dalam satu tabel beserta kolom *Source.Name* yang berisi nama berkas. Langkah terakhir dalam kueri tersebut cukup mengubah tipe kolom menggunakan fungsi *Table.TransformColumnTypes*.

Jika Anda memuat kueri ini ke dalam data, baris model untuk file PDF akan disertakan. Namun, baris ini akan kosong atau null untuk setiap kolom selain kolom *Source.Name*. Hal ini karena langkah transformasi tidak memproses file PDF dengan benar karena transformasi tersebut untuk file CSV. Untuk memperbaikinya, klik langkah Source di area **Applied Steps** dari editor Power Query untuk kueri utama yang dibuat.

Gunakan menu tarik-turun untuk kolom Extensions untuk memfilter baris ke baris *csv* saja. Dua ekspresi pertama dalam kueri utama sekarang terlihat seperti berikut:

```
    Source = Folder.Files("C:\data\Atlantic and Pacific Hurricanes 1851-2014"),
    #"Filtered Rows" = Table.SelectRows(Source, each ([Extension] = ".csv")),
```

Di sini, fungsi *Table.SelectRows* memfilter file yang dikembalikan oleh fungsi *Folder.Files* hanya ke file dengan ekstensi *.csv*.

Ini melengkapi penjelajahan kita terhadap fungsi folder di M. Selanjutnya, mari kita lihat pemrosesan file PDF.

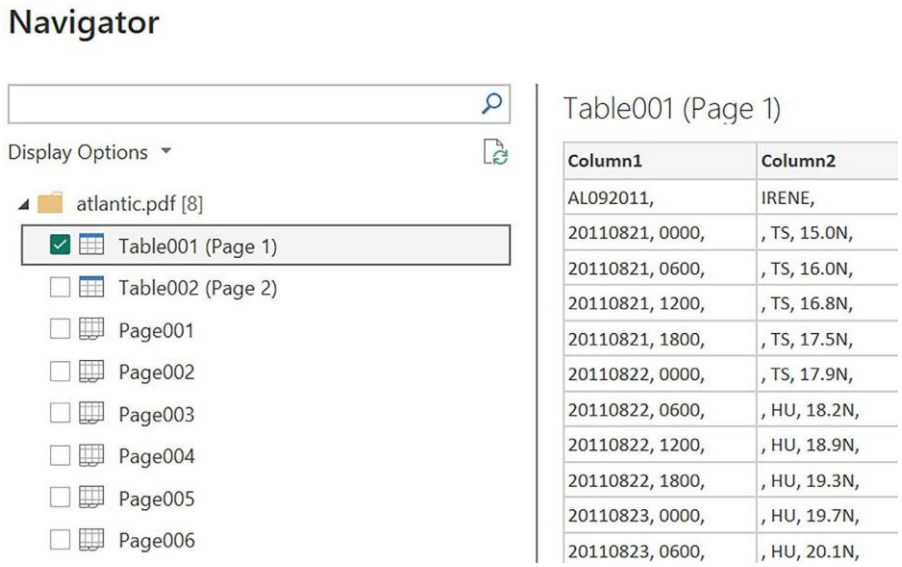
5. *PDF*

Dikembangkan pada tahun 1992 dan distandarkan sebagai ISO 32000, file **Portable Document Format (PDF)** telah lama menjadi format file standar yang digunakan untuk bertukar dokumen berformat yang mencakup teks berformat, tabel, gambar, anotasi, bidang formulir, media kaya seperti konten video, lapisan, dll. Mengingat sejarahnya yang panjang, lebih dari 30 tahun sebagai standar bebas dan terbuka untuk pertukaran dokumen yang independen dari perangkat keras, sistem operasi, dan perangkat lunak aplikasi, ada banyak data yang berpotensi berguna yang tersimpan dalam dokumen PDF. Misalnya, dokumen PDF sering digunakan untuk laporan keuangan bulanan atau triwulanan. Dengan demikian, kemampuan untuk mengimpor file PDF tersebut ke dalam satu model semantik memberikan peluang untuk melacak perubahan dari waktu ke waktu serta analisis tren. Untungnya, bahasa M menyertakan fungsi akses data untuk mengekstrak data dari file PDF, fungsi *PDF.Tables*.

Untuk mengeksplorasi fungsi *PDF.Tables* secara lebih rinci, kita akan menggunakan file PDF yang disertakan dalam file Atlantic and Pacific Hurricanes 1851-2014.zip yang disertakan dalam repositori GitHub untuk buku ini. Jika Anda belum melakukannya, unduh berkas Atlantic and Pacific Hurricanes 1851-2014.zip dan ekstrak keempat berkas yang disertakan dalam berkas ZIP ke satu folder di komputer lokal Anda. Ada dua berkas CSV, *atlantic.csv* dan *pacific.csv*, beserta dua berkas PDF, *atlantic.pdf* dan *pacific.pdf*.

Setelah Anda mengunduh dan mengekstrak berkas PDF, ikuti langkah-langkah berikut:

1) Di editor Power Query, klik tombol **New Source** pada tab Home di pita dan pilih **More**….
2) Di dialog **Get Data**, pilih **File** di navigasi kiri, pilih **PDF** di list sebelah kanan, lalu tekan tombol **Connect**.
3) Gunakan jendela **File Explorer** untuk menavigasi ke folder tempat Anda mengekstrak file yang terdapat dalam *Atlantic* and *Pacific Hurricanes 1851-2014.zip*.
4) Pilih file *atlantic.pdf* dan tekan tombol **OK**.
5) Dalam dialog **Navigator**, perhatikan bahwa beberapa elemen **Table** dan **Page** muncul. Fungsi Pdf.Tables mencoba mengidentifikasi tabel secara otomatis dalam dokumen PDF dan menyertakannya sebagai pilihan data potensial bersama dengan setiap halaman file PDF. Pilih item **Table001** (**Page 1)** lalu tekan tombol **OK**:



Gambar 3. 9 Navigator untuk file PDF

Setelah melakukan langkah-langkah ini, kode M berikut dibuat untuk dua ekspresi pertama dalam pernyataan *let* yang dapat dilihat di **Advanced Editor**:

```
    Source = Pdf.Tables(File.Contents("C:\data\Atlantic and Pacific Hurricanes
1851-2014\atlantic.pdf"), [Implementation="1.3"]),
    Table001 = Source{[Id="Table001"]}[Data],
```

Mari kita lihat baris *Source* terlebih dahulu. Seperti yang ditunjukkan, fungsi *Pdf.Tables* memiliki dua parameter. Seperti fungsi pengaksesan berkas lain yang telah kita bahas, parameter pertama adalah output dari fungsi *File.Contents*, yang mengakses berkas PDF dan mengembalikan data biner.

Parameter kedua adalah record opsi opsional. Record opsi ini berfungsi sama dengan record opsi untuk fungsi pengaksesan berkas lain yang telah kita bahas dan menyertakan opsi berikut:

- *Implementation*: Microsoft telah mengimplementasikan sejumlah algoritme berbeda untuk mengidentifikasi tabel dalam berkas PDF. Opsi ini menentukan algoritme mana yang digunakan dengan nilai yang valid yaitu *1.3*, *1.2*, dan *1.1*. Anda harus selalu menggunakan nilai algoritme terbaru karena nilai lainnya disediakan semata-mata untuk tujuan kompatibilitas mundur.
- *StartPage*: Defaultnya adalah 1 dan menentukan halaman pertama yang akan disertakan dalam pengambilan data.
- *EndPage*: Defaultnya adalah halaman terakhir yang tersedia dan menentukan halaman terakhir yang akan disertakan dalam pengambilan data.
- *MultiPageTables*: Menentukan apakah akan memperlakukan tabel yang mencakup beberapa halaman sebagai satu tabel atau beberapa tabel. Nilai ini dapat berupa *true* atau *false* dan defaultnya adalah true. Perhatikan bahwa identifikasi tabel yang mencakup beberapa halaman tidak selalu berhasil.
- *EnforceBorderLines*: Menentukan apakah garis batas tabel diberlakukan sebagai batas sel dalam tabel. Nilai ini dapat berupa *true* atau false dan defaultnya adalah *false*.

Ekspresi berikutnya, *Table001*, beroperasi persis seperti langkah navigasi yang kita jelaskan di bagian Excel dalam bab ini. Sekarang mari beralih ke file XML.

6. *XML*

XML, atau eXtensible Markup Language, adalah format file yang banyak digunakan untuk menyimpan data acak. Fleksibilitas dan standarisasinya membuatnya sangat cocok untuk pertukaran data antar organisasi. Misalnya, **Health Level 7 (HL7)** adalah standar global berbasis XML untuk pertukaran data kesehatan antar aplikasi. Contoh lain adalah OpenTravel Alliance, sebuah konsorsium perusahaan perjalanan dan perhotelan yang menggunakan file XML standar untuk bertukar data antar sistem.

Keluarga fungsi XML mencakup *Xml.Tables* dan *Xml.Document*. Argumen pertama dari kedua fungsi tersebut diperlukan dan menentukan isi file. Argumen terakhir dari kedua fungsi tersebut adalah nomor enkode opsional *(TextEncoding.Type*). Fungsi *Xml.Tables* juga memiliki parameter kedua opsional untuk menentukan opsi.

Meskipun kedua fungsi tersebut mengurai file XML, implementasi internalnya berbeda. *Xml.Tables* mengurai XML dan mengembalikan elemen sebagai tajuk kolom sementara *Xml.Document* mengembalikan struktur tabel hierarkis dengan tajuk kolom *Name, Namespace, Value*, dan *Attribute*.

Untuk memeriksa perbedaan dalam kedua fungsi ini, kita akan menggunakan contoh berkas *books.xml* yang ditemukan di sini: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ms762271(v=vs.85)

Berkas ini juga disertakan sebagai bagian dari repositori GitHub untuk bab ini. Untuk memudahkan referensi, berikut adalah contoh isi berkas:

```
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications
      with XML.</description>
   </book>
   <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-12-16</publish_date>
      <description>A former architect battles corporate zombies,
      an evil sorceress, and her own childhood to become queen
      of the world.</description>
   </book>
</catalog>
```

Berkas tersebut berisi simpul akar yang disebut katalog. Di dalam katalog tersebut terdapat beberapa simpul buku. Setiap simpul buku berisi banyak properti seperti *author*, *title*, *genre*, *price*, dll.

Xml.Tables

Menggunakan GUI editor Power Query untuk terhubung ke berkas ini akan menampilkan tabel berlabel buku di **Navigator**. Memilih tabel ini untuk pengambilan data akan menghasilkan kueri berikut:

```
let
    Source = Xml.Tables(File.Contents("C:\data\Books.xml")),
    Table0 = Source{0}[Table],
    #"Changed Type" = Table.TransformColumnTypes(Table0,{{"author", type text},
{"title", type text}, {"genre", type text}, {"price", type number}, {"publish_
date", type date}, {"description", type text}, {"Attribute:id", type text}})
in
    #"Changed Type"
```

Seperti yang dapat dilihat dalam kode, langkah Source mengambil konten file menggunakan *File.Contents* dan kemudian memproses konten file lebih

lanjut menggunakan *Xml.Tables*. Elemen di bawah simpul akar dikembalikan sebagai list. Dengan demikian, langkah Table0 mengakses item pertama dalam *list {0}* dan kemudian bagian [*Table*] menggunakan pemilihan bidang dan mengembalikan konten bidang record bernama *Table*. Langkah *#Changed Type* secara otomatis mencoba mengidentifikasi tipe data kolom yang dikembalikan. Perhatikan bahwa upaya ini tidak sepenuhnya berhasil karena kolom harga diidentifikasi sebagai angka desimal, bukan angka desimal tetap (mata uang).

Efek bersih dari kode tersebut adalah bahwa dokumen XML dikembalikan dengan tajuk kolom *author, title, genre, price, publish_date, description,* dan *Attribute*. Data untuk setiap buku disajikan sebagai baris dalam tabel ini. Perhatikan bahwa kolom Attribute.id berisi atribut ID untuk setiap simpul buku.

Sebelum beralih ke fungsi kedua, *Xml.Document*, mari kita jelajahi terlebih dahulu parameter kedua dari fungsi *Xml.Tables*, options. Parameter opsi fungsi *Xml.Tables* tidak terdokumentasi dengan baik. Penelitian terkini tentang subjek tersebut menunjukkan bahwa hanya satu bidang record yang didukung, *NavigationTables*, yang secara default bernilai *true*. Parameter ketiga opsional menentukan jenis penyandian (enumerasi *TextEncoding.Type*) seperti *TextEncoding*.Windows, yang berupa angka 1252.

Seperti yang dapat dilihat, penggunaan *Xml.Tables* memungkinkan hanya beberapa baris kode M untuk membuat tabel buku dengan properti buku tersebut yang ditampilkan sebagai kolom. Hal yang sama tidak berlaku untuk penggunaan *Xml.Document*.

Xml.Tables

Untuk berkas yang sama, penggunaan *Xml.Document* dan bukan *Xml.Tables* memerlukan kode berikut untuk mengembalikan tabel data yang sama persis:

```
let
    Source = Xml.Document(File.Contents("C:\data\Books.xml")),
    #"Removed Columns" = Table.RemoveColumns(Source,{"Namespace", "Attributes",
"Name"}),
    #"Expanded Value" = Table.ExpandTableColumn(#"Removed Columns", "Value",
{"Value", "Attributes"}, {"Value.1", "Attributes"}),
    #"Expanded Attributes" = Table.ExpandTableColumn(#"Expanded Value",
"Attributes", {"Value"}, {"Value"}),
    #"Expanded Value.1" = Table.ExpandTableColumn(#"Expanded Attributes",
"Value.1", {"Name", "Value"}, {"Name.1", "Value.2"}),
    PivotedColumn = Table.Pivot(#"Expanded Value.1", List.Distinct(#"Expanded
Value.1"[Name.1]), "Name.1", "Value.2"),
    #"Renamed Columns" = Table.RenameColumns(PivotedColumn,{{"Value",
"Attribute.id"}}),
    #"Changed Type" = Table.TransformColumnTypes(#"Renamed
Columns",{{"Attribute.id", type text}, {"author", type text}, {"title", type
text}, {"genre", type text}, {"description", type text}, {"price", type
number}, {"publish_date", type date}})
in
    #"Changed Type"
```

Seperti yang dapat Anda lihat, langkah *Source* hampir identik, kecuali penggunaan fungsi *Xml.Document* sebagai ganti fungsi *Xml.Tables*. Langkah source ini mengembalikan tabel baris tunggal sebagai berikut:

Name	Namespace	Value	Attributes
catalog		Table	Table

Baris ini mewakili elemen akar dari dokumen XML, *catalog*. Untuk mengembalikan elemen *books*, pertama-tama kita hapus kolom *Name*, *Namespace*, and *Attributes* menggunakan fungsi *Table.RemoveColumns*. Kemudian kita perluas kolom Nilai untuk Nilai dan Atribut menggunakan fungsi *Table.ExpandTableColumn*. Langkah ini mengembalikan baris untuk setiap elemen buku dengan tajuk kolom *Value.1* dan *Attributes*. Kedua kolom berisi tabel untuk setiap baris.

Dalam dua langkah berikutnya, kita kembali menggunakan *Table.ExpandTableColumn* untuk terlebih dahulu memperluas kolom *Attributes*, hanya untuk kolom *Value*. Ini adalah langkah #Atribut yang Diperluas. Kemudian kita perluas kolom *Value.1* hanya untuk kolom *Name*

dan *Value*. Perhatikan bahwa urutan kedua langkah ini tidak relevan. Satu-satunya perbedaan adalah sedikit variasi dalam nama kolom yang dihasilkan.

Pada titik ini, kita memiliki tabel yang beberapa baris pertamanya terlihat seperti berikut:

Tabel 3. 6 Tabel yang diperluas

Name.1	Value.2	Value
author	Gambardella, Matthew	bk101
title	XML Developer's Guide	bk101
Genre	Computer	bk101

Sekarang kita perlu memutar tabel sehingga kolom pertama, *Name.1*, menjadi tajuk kolom. Tabel kita sekarang berada dalam format dasar yang sama seperti yang dikembalikan oleh fungsi *Xml.Tables* yang dijelaskan sebelumnya di bagian ini. Yang diperlukan hanyalah mengganti nama kolom Value menjadi *Attribute.id* dan mengubah jenis kolom.

Seperti yang Anda lihat, fungsi *Xml.Tables* dan *Xml.Document* berperilaku sangat berbeda. Fungsi *Xml.Tables* mengembalikan elemen anak sebagai tabel sementara fungsi *Xml.Document* memungkinkan Anda menelusuri hierarki dokumen XML menggunakan empat kolom yang sama, *Name, Namespece, Value*, dan *Attributes*. Dalam contoh kita, fungsi *Xml.Document* memerlukan lebih banyak transformasi data untuk mengembalikan tabel yang sesuai dengan informasi yang diinginkan. Namun, jumlah transformasi data untuk setiap fungsi akan bervariasi tergantung pada struktur file XML yang tepat.

7. Penyimpanan Azure

Meskipun pembahasan lengkap tentang konsep Azure Storage berada di luar cakupan buku ini, sebagai ikhtisar singkat, pertimbangkan bahwa Microsoft Azure menyediakan beberapa mekanisme berbeda untuk menyimpan file dan informasi. Inti dari penyimpanan ini adalah apa yang dikenal sebagai akun Azure Storage. Akun Azure Storage dapat menyimpan

berbagai objek data seperti blob, file, antrean, tabel, dan penyimpanan data lake. Setiap akun penyimpanan ini memiliki namespace yang unik dan sering diakses menggunakan kunci akun yang dibuat secara otomatis.

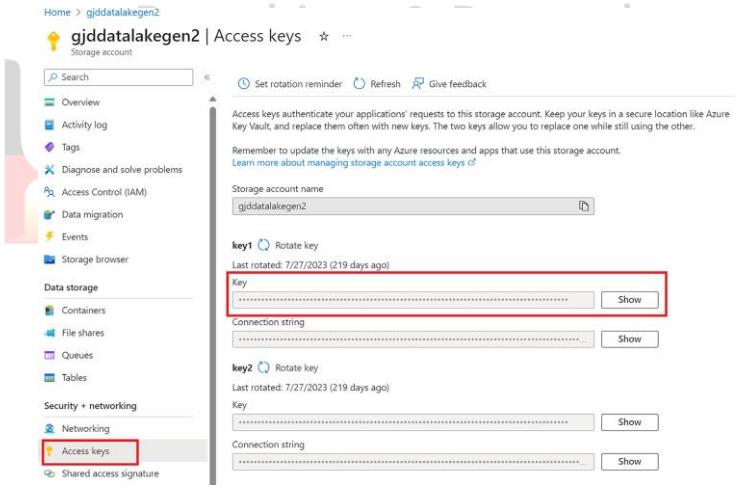
Pustaka inti M berisi beberapa fungsi untuk mengakses objek penyimpanan ini, termasuk:

- AzureStorage.Blobs
- AzureStorage.Tables
- AzureStorage.BlobContents
- AzureStorage.DataLake
- AzureStorage.DataLakeContents

Mari kita lihat fungsi *AzureStorage.Blobs*. Contoh kueri untuk menggunakan fungsi ini mungkin adalah:

```
let
    Source = AzureStorage.Blobs("mystorageaccountname")
in
    Source
```

Di sini, mystorageaccountname akan diganti dengan nama akun Azure Storage Anda. Saat mengautentikasi, Anda dapat memilih opsi **Account key**. Kunci akun ini dapat diambil menggunakan opsi **Access keys** saat melihat properti akun Azure Storage Anda di portal.azure.com:



Gambar 3. 10 Kunci akses akun penyimpanan di portal.azure.com

Fungsi ini mengembalikan tabel dua kolom yang terdiri dari semua kontainer blob yang ada dalam akun penyimpanan. *Tabel* ini terdiri dari kolom Nama untuk nama kontainer blob dan kolom Data yang mengembalikan objek *Tabel*. Anda dapat menavigasi ke objek Tabel ini untuk melihat folder dan file dalam kontainer blob. Pengalamannya sangat mirip dengan menggunakan fungsi *Folder.Contents* atau *Folder.Files*. Bahkan, setelah berada di level blob (file), kolom yang dikembalikan identik.

Fungsi *AzureStorage.Blobs* juga mendukung parameter opsional kedua. Parameter ini adalah record opsi yang dapat berisi opsi *BlockSize*, *RequestSize*, dan *ConcurrentRequests*. Nilai default untuk opsi ini masing-masing adalah *4MB*, *4MB*, dan *16*. Parameter ini dapat disesuaikan untuk mempercepat pengunduhan dengan mengorbankan penggunaan memori yang lebih besar. Misalnya, parameter *ConcurrentRequests* dapat disesuaikan menjadi 32, yang berarti akan ada 32 permintaan data secara bersamaan, masing-masing meminta *4MB* (jika menggunakan *RequestSize* default). Jadi, memori yang digunakan adalah *32 * 4MB* atau *128MB*.

Fungsi AzureStorage.Tables beroperasi secara identik dengan fungsi AzureStorage.Blobs kecuali bahwa fungsi ini mengembalikan tabel, bukan kontainer blob dari akun Azure Storage. Perbedaan lainnya adalah bahwa parameter kedua fungsi tersebut, record opsi, hanya mendukung opsi Timeout yang ditetapkan sebagai nilai durasi. Nilai default untuk opsi ini disediakan oleh sumber.

Fungsi *AzureStorage.BlobContents* mengambil URI sebagai parameter pertamanya dan record opsi opsional dengan opsi yang sama seperti fungsi *AzureStorage.Blobs*. URI harus berupa URI lengkap ke blob dalam akun Azure Storage.

Fungsi *Azure.DataLake* mengambil titik akhir data lake untuk akun Azure Storage sebagai parameter pertamanya. Ini adalah URI dalam bentuk https://mystorageaccountname.dfs.core.windows.net/.

Fungsi ini beroperasi seperti keluarga fungsi Folder, bahkan menyediakan opsi **Combine & Transform Data** yang sama dalam editor Power Query. Fungsi ini juga mengambil parameter kedua opsional, yaitu record opsi dengan opsi *BlockSize*, *RequestSize*, dan *ConcurrentRequests* yang sama serta nilai default seperti fungsi AzureStorage.Blobs serta opsi untuk *HierarchicalNavigation*. Opsi terakhir ini dapat berupa *null*, *true*, atau *false* dan default-nya adalah *false*.

Seperti yang Anda duga, *Azure.DataLakeContents* mengembalikan konten file individual yang disimpan dalam titik akhir data lake akun Azure Storage. Parameter pertama adalah URI lengkap ke file tersebut. Record opsi dapat diberikan sebagai parameter kedua dan memiliki opsi serta nilai default yang sama seperti fungsi *AzureStorage.Blobs*.

8. Format File Tambahan

Pustaka M standar berisi fungsi tambahan untuk mengakses berkas. Fungsi-fungsi ini beroperasi serupa dengan fungsi yang dibahas di bagian ini. Fungsi-fungsi ini meliputi yang berikut:

- *Hdfs.Contents*: Mengambil tabel folder dan berkas dari sistem berkas Hadoop. Mengambil satu parameter, *url*, sebagai teks. Ini adalah URI untuk folder Hadoop. Mirip dengan Folder.Contents.
- *Hdfs.Files*: Mengambil tabel berkas dari sistem berkas Hadoop. Mengambil satu parameter, *url*, sebagai teks. Ini adalah URI untuk folder Hadoop. Mirip dengan *Folder.Files*.
- *HdInsights.Containers*: Mengambil tabel kontainer dari brankas penyimpanan Azure HDInsights. Azure HDInsights adalah layanan Azure terkelola yang menyediakan kemampuan untuk menggunakan kerangka kerja sumber terbuka seperti Hadoop, LLAP, Apache Hive, Apache Spark, dan Apache Kafka. Mengambil satu parameter, akun, sebagai teks. Ini adalah URI akun untuk brankas penyimpanan Azure HDInsights. Setiap baris mengembalikan tautan ke blob kontainer.

- *HdInsights.Contents*: Juga mengambil tabel kontainer dari brankas penyimpanan Azure HDInsights. Mengambil satu parameter, akun, sebagai teks. Ini adalah URI akun untuk brankas penyimpanan Azure HDInsights. Setiap baris mengembalikan tautan ke blob kontainer.
- *HdInsights.Files*: Mengambil tabel file blob dari brankas penyimpanan Azure HDInsights. Mengambil satu parameter, akun, sebagai teks. Ini adalah URI akun untuk brankas penyimpanan Azure HDInsights. Setiap baris yang dikembalikan mencakup properti file dan tautan ke konten file.
- *Json.Document*: Mengembalikan konten dokumen JSON. Memiliki dua parameter, *jsonText* (sering kali merupakan output dari *File.Contents* atau *Web.Contents*) dan encoding, yang mengambil *TextEncoding*. Enumerasi *Type* (lihat bagian Text/GSV).
- *RData.FromBinary*: Digunakan untuk mengembalikan record bingkai data dari file RData. Mengambil satu parameter, *stream*, sebagai konten biner file.
- *Parquet.Document*: Bukan bagian dari pustaka inti M, tetapi diimplementasikan sebagai ekstensi (konektor). Untuk ikhtisar ekstensi dan konektor, lihat Bab 16, Mengaktifkan Ekstensi. Mengambil tabel dari konten dokumen Parquet. Memiliki dua parameter, biner, konten biner file (sering kali berupa keluaran *File.Contents* atau *Web.Contents*) dan record opsi opsional. Record opsi ini dapat berisi opsi atau *MaxDepth*, *Compression*, *PreserveOrder*, *LegacyColumnNameEncoding*, dan *TypeMapping*.

Ini melengkapi eksplorasi kita tentang format file dan folder. Selanjutnya, mari kita alihkan perhatian kita ke basis data dan kubus.

D. Mengambil Konten Web

Pada bagian terakhir, kita membahas akses data dalam file yang disimpan di komputer lokal atau sistem file jaringan. Namun, sumber data umum lainnya

adalah internet. Pustaka M standar mencakup sejumlah fungsi untuk mengambil data dari internet, termasuk:

- Web.BrowserContents
- Html.Table
- Web.Page
- Web.Contents
- Web.Headers
- WebAction.Request

Untuk melihat fungsi-fungsi ini beraksi, pertama-tama lakukan langkah-langkah berikut:

1) Pilih **Get Data** dari tab **Home** editor Power Query, lalu pilih **Web**.
2) Dalam dialog **From Web**, gunakan yang berikut untuk URL, https://subscription.packtpub.com/search.
3) Tekan tombol **OK**.
4) Pilih Autentikasi **Anonymous**, lalu tekan tombol **Connect**.
5) Dalam dialog **Navigator**, pilih **HTML Code** di bawah folder **Text**.

Mengikuti langkah-langkah ini akan menghasilkan kode M berikut seperti yang terlihat di **Advanced Editor**:

```
let
    #"HTML Code" = Web.BrowserContents("https://subscription.packtpub.com/
search")
in
    #"HTML Code"
```

Output dari kueri ini hanya mengembalikan **HyperText Markup Language (HTML)** mentah untuk halaman web. Fungsi *Web.BrowserContents* hanya mengembalikan HTML mentah untuk URI yang ditetapkan sebagai parameter pertama. Record opsi dapat diberikan sebagai parameter kedua opsional.

Record opsi ini dapat berisi kolom *ApiKeyName* dan *WaitFor*. Anda dapat mempelajari lebih lanjut tentang kolom opsi ini di URL berikut: https://powerquery.how/web-browsercontents/.

Sekarang, buat kueri yang sama seperti sebelumnya, tetapi kali ini pilih **Displayed Text** alih-alih **HTML Code** pada langkah 5. Kali ini, kode M berikut dibuat:

```
let
    Source = Web.BrowserContents("https://subscription.packtpub.com/search"),
    #"Extracted Table From Html" = Html.Table(Source, {{"Column1", "BODY"}}),
    Column1 = #"Extracted Table From Html"[Column1]{0}
in
    Column1
```

Dalam kueri ini, *Web.BrowserContents* masih digunakan untuk mendapatkan HTML mentah untuk halaman web. Namun, fungsi *Html.Table* kemudian digunakan untuk memilih BODY halaman web. Seharusnya jelas bahwa parameter pertama untuk fungsi *Html.Table* adalah konten HTML sebagai teks.

Parameter kedua untuk fungsi *Html.Table* sebenarnya adalah list pasangan pemilih **Cascading Style Sheets (CSS)** yang digunakan untuk mengekstrak informasi dari HTML yang disediakan. Pembahasan lengkap mengenai CSS berada di luar cakupan buku ini, tetapi perlu dicatat bahwa semua halaman web memiliki struktur dasar berikut:

```
<html>
    <head>
    </head>
    <body>
    </body>
</html>
```

Kode HTML ini, ketika dilihat sebagai tabel pada level akar (html), merupakan tabel satu kolom yang terdiri dari head dan body. Jadi, pemilih *BODY* masuk akal karena ini memilih bagian body dari halaman web.

Perhatikan bahwa *Html.Table* tidak mengembalikan HTML mentah, melainkan apa yang sebenarnya terlihat di halaman sebagai teks.

Parameter ketiga juga tersedia, record opsi. Record opsi ini mendukung satu bidang, *RowSelector*. *RowSelector* ini menggunakan pemilih CSS untuk hanya mengembalikan baris tertentu dari tabel HTML.

Catatan terakhir tentang *Html.Table* adalah bahwa fungsi Html.Table digunakan setiap kali Anda memilih tabel yang diidentifikasi atau menggunakan fitur Add **Table Using Examples** dalam dialog **Navigator** saat menggunakan konektor **Web**. Dalam skenario ini, editor Power Query menggunakan pemilih CSS dalam dokumen HTML untuk mengekstrak data tabel. Selanjutnya mari kita lihat fungsi Web.Page. Buat kueri berikut:

```
let
    Source =
        Web.Page(
            Web.BrowserContents("https://subscription.packtpub.com/search")
        ),
    Data = Source{0}[Data],
    Children = Data{0}[Children]
in
    Children
```

Fungsi *Web.Page* menggunakan satu parameter, html. Dalam kasus kita, kita menggunakan output dari fungsi *Web.BrowserContents* untuk parameter ini karena fungsi *Web.BrowserContents* mengembalikan HTML mentah sebagai teks.

Output dari fungsi *Web.Page* sangat mirip dengan fungsi *Xml.Document* yang dibahas sebelumnya dalam bab ini, seperti halnya interaksi pengguna saat menavigasi dokumen.

Output dari kueri adalah tabel dua baris, empat kolom. Kolom yang dikembalikan adalah *Kind*, *Name*, *Children*, dan *Text*, dan baris memiliki nilai *HEAD* dan *BODY* untuk kolom *Name*. Anda dapat terus menavigasi struktur

pohon hierarkis yang dikembalikan oleh kueri ini dengan mengeklik sel mana pun yang berisi tabel. Lihat kembali bagian XML dari bab ini dan khususnya bagian pada *Xml.Document* untuk penjelasan lebih lanjut.

Fungsi pengaksesan data terkait web terakhir yang akan kita bahas adalah fungsi *Web.Contents*. Fungsi *Web.Contents* mengembalikan konten permintaan web sebagai data biner, mirip dengan cara fungsi *File.Contents* mengembalikan konten file sebagai data biner. Ini berarti bahwa di mana pun kita menggunakan *File.Contents* di bagian Membuka file dan folder bab ini, kita dapat menggunakan *Web.Contents*.

Seperti fungsi *File.Contents*, fungsi *Web.Contents* memiliki dua parameter. Parameter pertama adalah URI teks (URL). Parameter kedua adalah record opsi. Record opsi ini dapat memiliki bidang berikut:

- *Query*: Menambahkan parameter string kueri ke URL secara terprogram.
- *ApiKeyName*: Digunakan untuk menentukan nama parameter kunci yang digunakan dalam URL.
- *Headers*: Menyediakan header HTTP tambahan yang digunakan dalam permintaan.
- *Timeout*: Nilai durasi yang menentukan berapa lama menunggu respons sebelum batas waktu habis. Batas waktu default adalah 100 detik.
- *ExcludedFromCacheKey* – List kunci header HTTP yang akan dikecualikan dari perhitungan untuk penyimpanan sementara.
- *IsRetry*: Bisa *true* atau *false*. Menentukan benar akan mengabaikan respons yang ada dalam cache.
- *ManualStatusHandling*: List kode status HTTP. Respons dengan kode ini tidak akan ditangani secara otomatis oleh fungsi *Web.Contents*.
- *RelativePath*: Menambahkan nilai yang ditentukan ke URL dasar saat membuat permintaan.
- *Content*: Nilai biner. Mengubah permintaan dari *GET* ke *POST* dengan nilai ini sebagai konten.

Untuk mengamati penggunaan opsi ini, buat kueri berikut:

```
let
    Source =
        Web.Contents(
            "https://subscription.packtpub.com",
            [
                RelativePath = "search",
                Query = [ query = "power+query",
                          products = "Book"
                        ],
                Timeout = #duration(0,0,0,30)
            ]
        )
in
    Source
```

Kueri ini menetapkan batas waktu 30 detik dan mengembalikan konten URI berikut sebagai biner: https://subscription.packtpub.com/search?query=power+query&products=Book

Nilai untuk RelativePath ditambahkan ke URL dasar yang ditetapkan. String kueri (bagian setelah?) dibuat secara otomatis dari nilai record parameter Kueri.

Ini melengkapi eksplorasi kita seputar pengambilan konten web. Sekarang kita akan beralih ke penyelidikan fungsi biner.

E. Menyelidiki Fungsi Biner

Seperti yang telah kita bahas, *File.Contents* dan *Web.Contents* berfungsi sebagai fungsi pengaksesan data inti. Yang satu menerima jalur file sementara yang lain menerima URI tetapi keduanya mengembalikan file atau halaman web yang diakses sebagai data biner. Maka masuk akal jika ada sekelompok fungsi yang didedikasikan untuk memproses dan menangani data biner yang dibangun ke dalam pustaka M inti. Ini adalah fungsi yang dimulai dengan kata Binary.

Faktanya, ada lebih dari 40 fungsi dalam keluarga Binary. Banyak dari fungsi ini agak esoteris. Namun, penggunaan umum untuk fungsi ini dapat diamati dalam kueri *Enter Data*.

Untuk mengamati fungsi biner ini dalam tindakan, buat kueri *Enter* Data di editor Power Query dengan data berikut:

Tabel 3. 7 Contoh data

Column1	Column2
One	1
Two	2
Three	3

Ini menghasilkan kode M berikut setelah pemformatan yang tepat:

```
let
    Source =
        Table.FromRows(
            Json.Document(
                Binary.Decompress(
                    Binary.FromText(
                        "i45W8s9LVdJRMlSK1YlWCinPB7KNIOyMolSQjLFSbCwA",
                        BinaryEncoding.Base64
                    ),
                    Compression.Deflate
                )
            ),
            let _t = ((type nullable text) meta [Serialized.Text = true]) in
type table [Column1 = _t, Column2 = _t]
        ),
    #"Changed Type" = Table.TransformColumnTypes(Source,{{"Column2", Int64.
Type}})
in
    #"Changed Type"
```

Di sini kita dapat melihat bahwa data yang kita masukkan diubah menjadi teks terkompresi yang dikodekan dalam bentuk biner. Ini adalah rangkaian teks dan angka yang panjang yang dimulai dengan *i45w8* dan diakhiri dengan *FSbCWA*. Rangkaian ini diubah menjadi data biner menggunakan fungsi *Binary.FromText*.

Pengodean dan kompresi biner adalah subjek yang rumit. Akan tetapi, konsep dasarnya cukup mudah dijelaskan. Bayangkan bahwa dalam sekumpulan data tertentu, kata the muncul puluhan atau bahkan ratusan kali. Alih-alih menyimpan kata *the* untuk setiap contoh, kita dapat memilih untuk merepresentasikan the dengan angka 1. Jadi, kita hanya perlu menyimpan satu digit, 1, untuk setiap contoh kata *the*, beserta tabel translasi yang memetakan angka 1 ke kata *the*.

Fungsi *Binary.FromText* memiliki dua parameter. Parameter pertama adalah teks yang akan diubah menjadi data biner dan parameter kedua adalah pengodean untuk teks tersebut. Parameter kedua ini adalah BinaryEncoding. Jenis enumerasi dan dapat berupa *BinaryEncoding.Base64* atau *BinaryEncoding.Hex*.

Setelah dikonversi ke data biner, data biner tersebut didekompresi menggunakan fungsi *Binary.Decompress*. Fungsi *Binary.Decompress* memiliki dua parameter. Parameter pertama adalah data biner yang akan didekompresi. Parameter kedua adalah tipe kompresi. Parameter kedua ini adalah enumerasi *Compression*.Type dan dapat memiliki nilai berikut:

- Kompresi.None
- Kompresi.Gzip
- Kompresi.Deflate
- Kompresi.Snappy
- Compression.Brotli
- Compression.LZ4
- Compression.Zstandard

Meskipun pembahasan yang mendalam tentang perbedaan jenis kompresi ini berada di luar cakupan buku ini, singkatnya, penggunaan satu format kompresi atau yang lain benar-benar bergantung pada apakah seseorang menilai rasio kompresi versus kecepatan. Misalnya, kompresi LZ4 dan Snappy jauh lebih cepat daripada Gzip tetapi menghasilkan file yang lebih besar. Di sisi lain, kompresi Brotli dirancang untuk mengompresi aliran video dengan cepat dan sebenarnya berkinerja lebih baik daripada Gzip dan menghasilkan file yang lebih kecil.

Setelah teks didekompresi, fungsi *Json.Document* dan fungsi *Table.FromRows* digunakan untuk mengubah data menjadi tabel kolom dan baris.

Orang mungkin ingin tahu dari mana asal rangkaian angka dan huruf yang panjang itu. Untuk mengungkapnya, pertama-tama kita perlu mengetahui dengan tepat teks apa yang dikompresi dan dikodekan. Kita dapat melakukannya menggunakan fungsi *Text.FromBinary* sebagai berikut:

```
let
    Source =
        Text.FromBinary(
            Binary.Decompress(
                Binary.FromText(
                    "i45W8s9LVdJRMlSK1YlWCinPB7KNIOyMolSQjLFSbCwA",
                    BinaryEncoding.Base64
                ),
                Compression.Deflate
            )
        )
in
    Source
```

Dalam kode ini, alih-alih memproses data biner yang didekompresi menggunakan fungsi *Json.Document* dan *Table.FromRows*, kita malah menggunakan fungsi *Text.FromBinary* untuk mengembalikan teks berikut:

```
[["One","1"],["Two","2"],["Three","3"]]
```

Fungsi *Text.FromBinary* memiliki dua parameter. Parameter pertama adalah data biner. Parameter kedua adalah enumerasi *TextEncoding.Type*. Enumerasi *TextEncoding.Type* dibahas di bagian *Text/GSV* bab ini.

Seperti yang terlihat pada output, dalam kasus ini, setiap baris diapit oleh tanda kurung siku dengan nilai kolom pada baris disimpan sebagai string yang diapit oleh tanda kutip ganda. Nilai kolom dipisahkan dengan koma. Terakhir, seluruh konten diapit oleh tanda kurung siku. Kita dapat menggunakan output ini untuk membuat ulang teks terkompresi dan terkode yang terlihat pada kueri Enter Data asli menggunakan kode berikut:

```
let
    Source =
        Binary.ToText(
            Binary.Compress(
                Text.ToBinary(
                    "[[""One"",""1""],[""Two"",""2""],[""Three"",""3""]]",
                    BinaryEncoding.Base64,
                    false
                ),
                Compression.Deflate
            )
        )
in
    Source
```

Perhatikan bahwa karena karakter tanda kutip ganda hadir dalam string, kita harus menggunakan dua karakter tanda kutip ganda sebagai urutan escape. Kode ini merekayasa balik bagaimana string asli yang dikompresi dan dikodekan dibuat.

Pertama, fungsi *Text.ToBinary* mengubah string teks menjadi biner. Fungsi *Text.ToBinary* mengambil tiga parameter. Parameter pertama adalah teks yang akan diubah menjadi data biner. Parameter kedua adalah enumerasi *TextEncoding.Type* sebagaimana dibahas dalam bagian *Text/GSV* bab ini. Dalam kode yang dirujuk, tipe enkode *BinaryEncoding.Base64* digunakan. Parameter ketiga adalah parameter *includeByteOrderMark* dan dapat bernilai benar atau salah.

Setelah teks diubah menjadi biner, fungsi *Binary.Compress* digunakan untuk mengompresi data. Fungsi *Binary.Compress* memiliki dua parameter, yang pertama adalah data biner yang akan dikompresi. Parameter kedua adalah enumerasi *Compression.Type* yang dibahas sebelumnya di bagian ini. Kita sekarang dapat menggunakan fungsi *Binary.ToText* untuk mengembalikan teks terkompresi dan berkode biner sebagai teks, yang menghasilkan keluaran berikut:

```
i45W8s9LVdJRMlSK1YlWCinPB7KNIOyMolSQjLFSbCwA
```

Di sini kita dapat melihat bahwa ini adalah teks yang sama persis dengan yang muncul dalam kueri *Enter Data* asli kita. Fungsi *Binary.ToText* memiliki dua parameter, data biner dan enumerasi *BinaryEncoding.Type*. Sekarang mari kita lihat kelas fungsi tambahan dan bagaimana fungsi tersebut dapat digunakan untuk membantu kita mengakses dan mengambil data.

1. Fungsi Garis

Selain fungsi keluarga Binary, ada fungsi tambahan untuk menangani data biner dan konten file seperti berikut:

- Lines.FromBinary
- Lines.FromText
- Lines.ToBinary
- Lines.ToText

Keluarga fungsi Lines mengonversi list teks ke dan dari nilai teks tunggal atau ke dan dari data biner. Secara default, fungsi-fungsi ini menggunakan karakter carriage return dan line-feed sebagai pemisah.

Untuk melihat bagaimana fungsi-fungsi ini dapat digunakan, pertimbangkan file yang berisi beberapa dokumen JSON dengan satu dokumen per baris, seperti:

```
{"Column1": "One", "Column2": "1"}
{"Column1": "Two", "Column2": "2"}
{"Column1": "Three", "Column2": "3"}
```

Meskipun setiap baris adalah dokumen JSON, upaya untuk mengurai berkas ini menggunakan fungsi Json.Document akan gagal karena berkas tersebut secara keseluruhan bukanlah JSON yang valid. Untungnya, kita dapat menggunakan fungsi *Lines.FromBinary* untuk membantu kita mengonversi data dalam berkas ini menjadi satu tabel menggunakan kode berikut:

```
let
    Source =
        Table.FromColumns(
            {
            Lines.FromBinary(
                File.Contents(
"C:\data\MultipleJSONDocs.json"), null, null
            )
            }
        ),
    #"Transformed Column" = Table.TransformColumns(Source, {"Column1", Json.
Document}),
    #"Expanded Column1" = Table.ExpandRecordColumn(#"Transformed Column",
"Column1", {"Column1", "Column2"}, {"Column1", "Column2"}),
    #"Changed Type" = Table.TransformColumnTypes(#"Expanded
Column1",{{"Column1", type text}, {"Column2", Int64.Type}})
in
    #"Changed Type"
```

Dalam kode ini, fungsi *Lines.FromBinary* digunakan untuk membaca setiap baris file dan mengembalikan baris-baris ini sebagai list:

	List
1	{"Column1": "One", "Column2": "1"}
2	{"Column1": "Two", "Column2": "2"}
3	{"Column1": "Three", "Column2": "3"}

Gambar 3. 11 Output dari fungsi Lines.FromBinary

List ini diubah menjadi tabel satu kolom melalui fungsi *Table.FromColumns* dan kemudian diubah lebih lanjut menggunakan fungsi *Json.Document* untuk memproses setiap baris sebagai dokumen JSON. Ini dilakukan dengan bantuan fungsi *Table.TransformColumns*. Dalam kasus ini, fungsi *Json.Document* mengembalikan record untuk setiap baris, yang dapat kita perluas menjadi kolom menggunakan fungsi *Table.ExpandRecordColumn*.

Ini melengkapi eksplorasi kita tentang fungsi Biner. Informasi tambahan tentang fungsi Biner lainnya dapat ditemukan di sini:

https://powerquery.how/binary-functions/. Sekarang mari kita lanjutkan untuk mempelajari cara mengakses basis data dan kubus.

F. Mengakses Database dan Kubus

Sama seperti pustaka fungsi M standar yang mendukung banyak format file standar yang berbeda, terdapat pula fungsi inti untuk mengakses berbagai format basis data/kubus standar industri yang berbeda. Di sini, kita menggunakan istilah kubus untuk merujuk pada sistem basis data yang diklasifikasikan sebagai sistem **Online Analytics Processing (OLAP)**. Istilah basis data merujuk pada basis data relasional yang diklasifikasikan sebagai sistem **Online Transactional Processing (OLTP)**. Sistem OLAP cocok untuk analisis data multidimensi sementara sistem OLTP cocok untuk operasi transaksional.

Sistem OLTP cenderung menggunakan struktur data yang sangat dinormalisasi yang menghargai efisiensi penyimpanan data dan kecepatan operasi penulisan di atas masalah lain. Pikirkan skenario transaksi tinggi seperti manajemen inventaris dalam gudang atau penjualan eceran. Sebaliknya, sistem OLAP menggabungkan (mendenormalisasi) informasi historis dan menghargai kecepatan operasi baca serta efisiensi analisis dan pelaporan.

Pustaka inti M mencakup fungsi untuk mengakses jenis basis data dan kubus berikut:

Tabel 3. 8 Fungsi M untuk mengakses database dan kubus

Sistem	Deskripsi	Fungsi M
Microsoft SQL Server	Sistem basis data relasional Microsoft.	`Sql.Database` `Sql.Databases`
Microsoft Analysis Services	Merujuk pada **SQL Server Analysis Services (SSAS)** dan **Azure Analysis Services (AAS)**. Mendukung kubus tabular dan multidimensi. Kubus tabular menggunakan **Data Analysis Expressions (DAX)** untuk kueri dan kalkulasi, sedangkan kubus multidimensi menggunakan **Multidimensional Expressions (MDX)** untuk hal yang sama.	`AnalysisServices.Database` `AnalysisServices.Databases`

Microsoft Access	Mesin Basis **Access Database Engine (ACE)**, sebelumnya mesin basis data Jet.	`Access.Database`
Adobe Analytics cubes	Layanan analitik kubus Adobe Experience Cloud, sistem terkemuka untuk analitik web. Adobe Experience Cloud sebelumnya dikenal sebagai Adobe Marketing Cloud. Adobe Systems mengakuisisi komponen analitik Adobe Experience Cloud dari Omniture.	`AdobeAnalytics.Cubes`
IBM DB2	Sistem basis data relasional yang dikembangkan oleh IBM.	`DB2.Database`
IBM Informix	Sistem basis data relasional yang awalnya dikembangkan oleh Informix, yang diakuisisi oleh IBM pada tahun 2001.	`Informix.Database`
Oracle Essbase	Sistem kubus multidimensi yang awalnya dikembangkan oleh Arbor Software Corporation, yang bergabung dengan Hyperion Software pada tahun 1998. Oracle kemudian mengakuisisi Hyperion Solutions Corporation dan awalnya memasarkan Essbase sebagai DB2 OLAP Server.	`Essbase.Cubes`
Oracle MySQL	MySQL adalah database relasional gratis dan open-source yang dirilis di bawah Lisensi Publik Umum GNU pada tahun 1995. Awalnya dimiliki dan disponsori oleh perusahaan MySQL AB, yang diakuisisi oleh Sun Microsystems, yang diakuisisi oleh Oracle pada tahun 2010.	`MySQL.Database`
Oracle Database	Umumnya disebut sebagai Oracle, Oracle Database adalah database relasional yang mendukung beban kerja OLTP dan gudang data.	`Oracle.Database`
PostgreSQL	Juga dikenal sebagai Postgres, PostgresSQL adalah database relasional gratis dan sumber terbuka yang awalnya dirilis pada tahun 1996.	`PostgreSQL.Database`
SAP HANA	Sistem basis data relasional berorientasi kolom, dalam memori, yang dikembangkan oleh SAP.	`SapHana.Database`
SAP Business Warehouse	Awalnya merupakan database relasional, SAP Business Warehouse kemudian berkembang untuk memanfaatkan database dalam memori SAP HANA dan menyediakan fungsionalitas OLAP tingkat lanjut.	`SapBusinessWarehouse. Cubes`
SAP Sybase	Sistem basis data relasional yang awalnya dibuat oleh Sybase dan kemudian diakuisisi oleh SAP.	`Sybase.Database`
Teradata	Sistem basis data relasional Teradata.	`Teradata.Database`

Seperti yang Anda duga, fungsi yang diakhiri dengan .*Database* umumnya dirancang untuk bekerja dengan database relasional (OLTP) sementara fungsi yang diakhiri dengan .*Cubes* dirancang untuk mengakses sistem analitik (OLAP). Pengecualiannya adalah keluarga fungsi *AnalysisServices*.

Meskipun ada pengecualian, sebagian besar fungsi beroperasi dengan cara yang sama dan mengambil dua atau tiga parameter. Parameter pertama adalah parameter *server*, yang menentukan string koneksi untuk koneksi ke server basis data. Jika ada tiga parameter untuk fungsi tersebut, parameter kedua adalah parameter basis data, yang menentukan nama basis data yang akan diakses. Parameter terakhir adalah record opsi.

Opsi yang tersedia untuk record opsi bervariasi tergantung pada fungsinya. Namun, opsi umum mencakup yang berikut ini:

- *Query*: SQL, MDX, DAX, atau kueri asli lainnya yang didukung oleh sistem sumber.
- *CommandTimeout*: Menentukan durasi berapa lama kueri diizinkan untuk berjalan sebelum dibatalkan. Ini adalah nilai durasi dan defaultnya adalah sepuluh menit.
- *ConnectionTimeout*: Menentukan durasi berapa lama untuk mencoba koneksi sebelum waktu habis. Ini adalah nilai durasi dan defaultnya tergantung pada driver yang digunakan.
- *CreateNavigationProperties*: Menentukan apakah akan membuat properti navigasi. Dapat berupa benar atau salah dan defaultnya adalah benar.
- *NavigationPropertyNameGenerator*: Menentukan fungsi yang digunakan untuk membuat nama properti navigasi.
- *HierarchicalNavigation*: Menentukan apakah objek dikelompokkan berdasarkan nama skemanya. Bisa benar atau salah dan defaultnya salah.

Sebagai percobaan, buat dua kueri berikut:

```
let
    Source = Sql.Database("ServerName", "DatabaseName", [HierarchicalNavigation
= false])
in
    Source
```

dan

```
let
    Source = Sql.Database("ServerName", "DatabaseName", [HierarchicalNavigation
= true])
in
    Source
```

Ganti *ServerName* dan DatabaseName dalam kode dengan nama sebenarnya dari SQL Server dan database SQL. Kueri pertama mengembalikan tabel lima kolom yang mencantumkan tabel, tampilan, dan fungsi dalam database. Kolom yang dikembalikan adalah *Name, Data, Schema, Item*, dan *Kind*. Kueri kedua mengembalikan tabel dua kolom yang mencantumkan skema dalam database. Kolom yang dikembalikan adalah Skema dan Data.

Dalam contoh berikutnya, kita menggunakan record opsi untuk menjalankan kueri SQL asli guna mengembalikan hanya beberapa kolom dan 1.000 baris teratas dari tabel tertentu:

```
let
    Source =
        Sql.Database(
            "localhost",
            "AdventureWorksDW2019",
            [Query="SELECT TOP (1000) [CustomerKey],[FirstName],[MiddleName]
FROM [AdventureWorksDW2019].[dbo].[DimCustomer]"])
in
    Source
```

Di sini, kita mengambil data dari tabel *dbo.DimCustomer* dari database bernama *AdventureWorksDW2019* yang tersedia di instans SQL Server lokal (localhost). Kolom Query digunakan untuk menjalankan pernyataan *SQL SELECT*.

Contoh database AdventureWorks dapat diunduh di sini: https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure.

1. Fungsi Kubus

Fungsi yang mengakses kubus memiliki keluarga fungsi khusus yang dapat digunakan untuk mengubah data. Fungsi-fungsi ini dimulai dengan *Cube*. Ada 16 fungsi seperti berikut:

- Cube.AddAndExpandDimensionColumn
- Cube.AddMeasureColumn
- Cube.ApplyParameter
- Cube.AttributeMemberId
- Cube.AttributeMemberProperty
- Cube.CollapseAndRemoveColumns
- Cube.Dimensions
- Cube.DisplayFolders
- Cube.MeasureProperties
- Cube.MeasureProperty
- Cube.Measures
- Cube.Parameters
- Cube.Properties
- Cube.PropertyKey
- Cube.ReplaceDimensions
- Cube.Transform

Saat bekerja dengan kubus di editor Power Query, tab **Cube Tools** | **Manage** khusus muncul di pita. Pita ini memaparkan opsi **Add Items** dan opsi **Collapse Columns**. Opsi Add Items memanggil fungsi **Cube.AddAndExpandDimensionColumn** sementara opsi **Collapse Columns** memanggil fungsi *Cube.CollapseAndRemoveColumns*.

Jika menggunakan editor Power Query untuk mengakses kumpulan data Power BI yang dipublikasikan di cloud melalui Analysis Services, kode M yang dihasilkan terlihat seperti ini:

```
let
    Source =
        AnalysisServices.Databases(
            "powerbi://api.powerbi.com/v1.0/<tenant>/<workspace>",
            [TypedMeasureColumns=true, Implementation="2.0"]
        ),
    #"MyCube" = Source{[Name="MyCube"]}[Data],
    Model1 = #"MyCube"{[Id="Model"]}[Data],
    Model2 = Model1{[Id="Model"]}[Data],
    #"Added Items" = Cube.Transform(Model2,
        {
            {
                Cube.AddAndExpandDimensionColumn,
                "[FactInternetSales]",
                {
                    "[FactInternetSales].[CarrierTrackingNumber].
[CarrierTrackingNumber]",
                    "[FactInternetSales].[CurrencyKey].[CurrencyKey]"
                }
            }
        })
in
    #"Added Items"
```

Seperti yang dapat Anda lihat di baris Sumber, fungsi *AnalysisServices.Database* digunakan untuk terhubung ke ruang kerja. Format string koneksi untuk mengakses titik akhir XMLA untuk ruang kerja menggunakan format berikut: power-bi://api.powerbi.com/v1.0/<tenant>/<workspace> Di sini, <*tenant*> adalah nama penyewa Microsoft 365 dan <workspace> adalah nama ruang kerja Power BI.

Catatan opsi untuk fungsi AnalysisServices.Database dapat menyertakan opsi untuk CommandTimeout

dan *ConnectionTimeout* yang dibahas sebelumnya, tetapi juga dapat menyertakan opsi berikut:

- *TypeMeasureColumn*: Menentukan apakah tipe dalam kubus digunakan untuk tipe kolom pengukuran yang ditambahkan. Dapat berupa *true* atau *false*. Nilai default adalah *false* dan dengan demikian semua kolom pengukuran yang ditambahkan bertipe number.

- *Culture*: Menentukan budaya yang digunakan untuk data. Ini serupa dengan properti *Local Identifier* yang tersedia dalam string koneksi Analysis Services.
- *SubQueries*: Mengontrol perilaku anggota terhitung pada subkubus atau subpilihan. Dapat berupa 0, 1, atau 2 dan default ke 2. Ini serupa dengan properti SubQueries yang tersedia dalam string koneksi Analysis Services.
- Implementation: Menentukan versi implementasi.

Tiga ekspresi berikutnya setelah ekspresi Sumber cukup menavigasi ke kubus yang diinginkan menggunakan sintaks pengaksesan data standar yang dijelaskan sebelumnya. Terakhir, fungsi *Cube.Transform* digunakan bersama dengan fungsi *Cube.AddAndExpandDimensionColumn* untuk mengembalikan informasi yang diinginkan dari kubus.

Mirip dengan bagaimana *File.Contents* dan *Web.Contents* masing-masing merupakan fungsi pengaksesan data dasar untuk file dan situs web, fungsi *Cube.Transform* memiliki peran yang serupa sehubungan dengan pengaksesan data dari kubus. Fungsi *Cube.Transform* menggunakan dua parameter, kubus yang akan ditransformasi dan list transformasi yang akan diterapkan pada kubus. Dalam kode contoh, satu transformasi ditetapkan melalui fungsi *Cube.AddAndExpandDimensionColumn*. Informasi selengkapnya tentang keluarga fungsi Cube dapat ditemukan di *powerquery.how*.

Ini melengkapi penyelidikan kita tentang akses data untuk basis data dan kubus. Selanjutnya, kita akan mengalihkan perhatian kita ke protokol data standar.

G. Bekerja Dengan Protokol Data Standar

Selama bertahun-tahun, sejumlah standar telah dikembangkan untuk memfasilitasi komunikasi dan pertukaran data yang efisien dan lancar antara berbagai platform dan sistem. Pustaka M standar menyediakan fungsi untuk banyak standar pengaksesan data ini, termasuk:

Tabel 3. 9 Fungsi M untuk standar akses data

Standard	Description	M Functions
ADO.NET	**ADO.NET** (**ActiveX Data Objects .NET**) adalah teknologi akses data dalam kerangka kerja Microsoft .NET yang menyediakan serangkaian pustaka dan kelas untuk mengakses dan memanipulasi data dari berbagai sumber data, seperti basis data, file XML, dan lainnya. ADO.NET dirancang secara khusus untuk membangun aplikasi berbasis data dan merupakan komponen mendasar dalam kerangka kerja .NET untuk bekerja dengan basis data relasional.	`AdoDotNet.DataSource` `AdoDotNet.Query`
Delta Lake	Delta Lake adalah lapisan penyimpanan data sumber terbuka yang memperluas mesin komputasi seperti Apache Spark, PrestoDB, dan Hive untuk membangun data lake yang dapat diskalakan, andal, dan berkinerja tinggi. Awalnya dikembangkan oleh Databricks, Delta Lake menyediakan beberapa fitur dan pengoptimalan tingkat lanjut untuk mengatasi tantangan umum data lake dan meningkatkan manajemen data.	`DeltaLake.Table`
OData	OData, kependekan dari Open Data Protocol, adalah standar terbuka untuk meminta dan memperbarui data melalui web. Ini adalah protokol RESTful yang memungkinkan klien berinteraksi dengan sumber daya data dengan menggunakan metode HTTP standar, seperti GET, POST, PUT, PATCH, dan DELETE. OData dirancang untuk menyediakan cara standar untuk mengekspos dan mengakses data dari berbagai sumber, sehingga memudahkan pembuatan aplikasi berbasis data yang dapat menggunakan dan berinteraksi dengan data di berbagai platform dan layanan.	`OData.Feed`

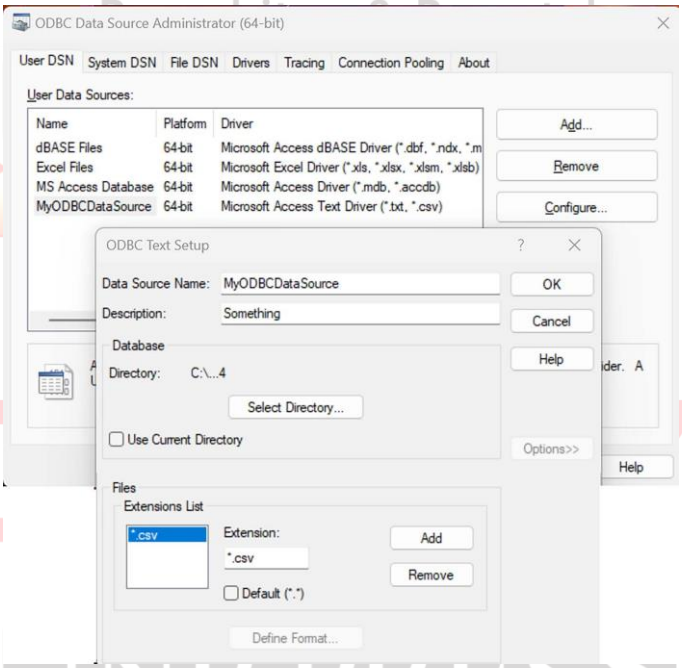
ODBC	ODBC adalah singkatan dari Open Database Connectivity. ODBC merupakan antarmuka pemrograman aplikasi (API) standar industri yang memungkinkan aplikasi perangkat lunak untuk mengakses dan berinteraksi dengan basis data menggunakan antarmuka umum, apa pun sistem manajemen basis data (DBMS) yang digunakan. ODBC dikembangkan oleh Microsoft pada awal tahun 1990-an dan sejak saat itu telah	`Odbc.DataSource`
		`Odbc.InferOptions`
		`Odbc.Query`

	menjadi standar yang diadopsi secara luas untuk konektivitas basis data.	
OLE DB	OLE DB (singkatan dari Object Linking and Embedding Database) adalah teknologi Microsoft yang menyediakan antarmuka akses data untuk mengakses dan memanipulasi data dari berbagai sumber data. Ini adalah serangkaian antarmuka dan protokol **Component Object Model (COM)** yang memungkinkan aplikasi berinteraksi dengan berbagai sumber data, termasuk basis data relasional, spreadsheet, file teks, dan banyak lagi. OLE DB diperkenalkan oleh Microsoft sebagai evolusi dari teknologi ODBC (Open Database Connectivity) sebelumnya dan dianggap sebagai penerus ODBC.	`OleDb.DataSource` `OleDb.Query`
SODA	Socrata Open Data API (SODA) adalah API yang disediakan oleh Socrata (sekarang Tyler Technologies), sebuah perusahaan yang mengkhususkan diri dalam manajemen data dan solusi data terbuka. API SODA memungkinkan pengembang untuk mengakses dan berinteraksi dengan kumpulan data yang dipublikasikan di portal data terbuka Socrata. Portal ini digunakan oleh berbagai lembaga pemerintah, organisasi nirlaba, dan bisnis untuk berbagi dan membuat data mereka dapat diakses publik.	`Soda.Feed`

Setiap komputer Windows dilengkapi dengan sejumlah driver ODBC. Driver adalah komponen perangkat lunak yang bertindak sebagai perantara ke berbagai format basis data dan file. Windows dilengkapi dengan driver ODBC untuk file dBASE, file Excel, file Microsoft Access, file CSV, SQL Server, dan sistem basis data lainnya. Driver ODBC tambahan juga tersedia dari berbagai pihak ketiga.

Ada driver ODBC 32-bit dan 64-bit. Anda harus menggunakan driver yang sesuai yang kompatibel dengan perangkat lunak aplikasi Anda (64-bit untuk Power BI Desktop).

Untuk menggunakan ODBC, Anda harus menginstal driver yang sesuai pada sistem Anda terlebih dahulu. Kemudian, Anda membuat **Data Source Name (DSN)** ODBC yang menentukan detail dan pengaturan koneksi untuk sumber data tertentu. Ini dapat dilakukan dengan mengetik Sumber Data ODBC di bilah pencarian Windows. DSN ini kemudian digunakan dalam string koneksi untuk koneksi ODBC:



Gambar 3. 12 ODBC DSN

Misalnya, setelah menyiapkan DSN 64-bit yang disebut MyODBCDataSource menggunakan Microsoft Access Text Driver untuk menunjuk ke folder untuk kumpulan data Badai Atlantic dan *Pacific Hurricanes 1851-2014*, kita dapat menggunakan kueri berikut untuk mengambil konten file *atlantic.csv* menggunakan ODBC:

```
let
    Source = Odbc.DataSource("dsn=MyODBCDataSource",
[HierarchicalNavigation=true]),
    #"C:\DATA\ATLANTIC AND PACIFIC HURRICANES 1851-2014_
Database" = Source{[Name="C:\DATA\ATLANTIC AND PACIFIC HURRICANES
1851-2014",Kind="Database"]}[Data],
    atlantic.csv_Table = #"C:\DATA\ATLANTIC AND PACIFIC HURRICANES 1851-2014_
Database"{[Name="atlantic.csv",Kind="Table"]}[Data]
in
    atlantic.csv_Table
```

Struktur pengaksesan data serupa dengan apa yang telah kita lihat pada contoh lain dalam bab ini. Fungsi Odbc.DataSource mendukung dua parameter, yang pertama adalah string koneksi dan yang kedua adalah record opsi. Kolom record opsi yang tersedia mencakup banyak opsi fungsi pengaksesan data kubus dan basis data umum yang dibahas dalam bagian Mengakses Basis Data dan Gubes dalam bab ini, termasuk *CreateNavigationProperties*, *HierarchicalNavigation*, *ConnectionTimeout*, dan *CommandTimeout*. Selain itu, ada kolom true/false logis, SqlCompatibleWindowsAuth, yang menentukan apakah string koneksi kompatibel dengan autentikasi Windows. Nilai default adalah true.

OData adalah standar protokol data umum dan populer lainnya. Kita dapat mengambil data dari titik akhir OData menggunakan kueri berikut:

```
let
    Source = OData.Feed("https://services.odata.org/TripPinRESTierService/
People", null, [Implementation="2.0"])
in
    Source
```

Di sini, fungsi *OData.Feed* digunakan untuk mengambil data dari URI yang dikonfigurasi untuk OData. Dalam kasus kita, kita menggunakan salah satu layanan referensi yang disediakan oleh *odata.org*. Fungsi *Odata.Feed* memiliki tiga parameter. Parameter pertama adalah URI koneksi. Parameter kedua adalah record header (biasanya *null*). Parameter ketiga adalah record opsi. Record opsi ini mendukung banyak bidang yang sama seperti fungsi *Web.Contents*. Namun, ada banyak opsi yang tersedia yang khusus untuk OData. Anda dapat

mengetahui lebih lanjut tentang opsi ini di sini: https://learn.microsoft.com/en-us/powerquery-m/odata-feed

Ini melengkapi eksplorasi kita tentang protokol data standar. Sekarang mari kita lihat fungsi akses data standar yang mungkin terlewatkan.

H. Mengatasi Konektor Tambahan

Sejauh ini kita telah membahas sebagian besar, tetapi tidak semua, fungsi pengaksesan data yang tersedia dalam pustaka M standar. Perhatikan bahwa lebih banyak fungsi pengaksesan data tersedia dalam lingkungan M global dalam produk seperti Power BI Desktop. Fungsi pengaksesan data tambahan ini menyediakan kemampuan untuk terhubung ke berbagai sistem bisnis khusus seperti sistem **Customer Relationship Management (CRM)**, sistem manajemen **Enterprise Resource Management (ERP)**, dan lainnya. Fungsi pengaksesan data tambahan ini berasal dari konektor eksternal yang disertakan dengan Power BI Desktop.

Konektor eksternal dibahas dalam Bab 16, Mengaktifkan Ekstensi. List lengkap fungsi akses data yang tersedia di lingkungan global untuk Power BI Desktop dapat ditemukan di sini: https://powerquery.how/accessing-data-functions/

Mari kita bahas secara singkat fungsi akses data yang tersisa yang tersedia di pustaka M standar. Fungsi ini umumnya terbagi dalam dua kategori, sistem perangkat lunak populer dan fungsi identitas.

1. Sistem Perangkat Lunak Populer

Pustaka fungsi M standar mencakup beberapa fungsi untuk menghubungkan ke sistem perangkat lunak populer seperti berikut ini:

Tabel 3. 10 Fungsi M untuk menghubungkan ke sistem perangkat lunak

Sistem	Deskripsi	Fungsi M
Microsoft Active Directory	**Active Directory** (**AD**) adalah layanan direktori yang dikembangkan oleh Microsoft yang digunakan terutama dalam jaringan berbasis Windows untuk mengelola dan mengatur sumber daya seperti pengguna, komputer, printer, dan objek jaringan lainnya. Layanan ini menyediakan basis data terpusat yang menyimpan informasi tentang sumber daya jaringan dan memungkinkan administrator untuk mengontrol dan mengelola akses ke sumber daya ini dalam lingkungan domain.	`ActiveDirectory.Domains`
Microsoft Exchange	Microsoft Exchange adalah server email dan perangkat lunak kalender yang populer dan banyak digunakan yang dikembangkan oleh Microsoft. Ini adalah bagian dari keluarga produk Microsoft Office dan dirancang untuk menyediakan komunikasi email, penjadwalan kalender, dan kemampuan kolaborasi untuk bisnis dan organisasi.	`Exchange.Contents`
Microsoft SharePoint	Microsoft SharePoint adalah platform kolaboratif berbasis web dan sistem manajemen konten yang dikembangkan oleh Microsoft. SharePoint merupakan bagian dari rangkaian alat produktivitas Microsoft 365 (sebelumnya dikenal sebagai Office 365) dan dirancang untuk memfasilitasi kerja tim, berbagi konten, dan manajemen dokumen dalam organisasi.	`SharePoint.Contents`, `SharePoint.Files`, `SharePoint.Tables`
Salesforce	Salesforce adalah platform CRM berbasis cloud terkemuka yang dikembangkan oleh Salesforce.com, Inc. Platform ini dirancang untuk membantu bisnis dari semua ukuran mengelola hubungan pelanggan, proses penjualan, upaya pemasaran, dan interaksi layanan pelanggan secara lebih efisien dan efektif..	`Salesforce.Data`, `Salesforce. Reports`

Penggunaan fungsi pengaksesan data ini pada umumnya cukup mudah. Misalnya, kueri berikut akan mengembalikan semua kontak untuk kotak surat dari Exchange Online:

```
let
    Source = Exchange.Contents("user@company.com"),
    People1 = Source{[Name="People"]}[Data]
in
    People1
```

Rangkaian fungsi SharePoint perlu diklarifikasi. Pertimbangkan pertanyaan berikut:

```
let
    Source = SharePoint.Files("https://<tenant>.sharepoint.com/<site>/",
[ApiVersion = 15])
in
    Source
```

Kueri ini menggunakan fungsi *SharePoint.Files* dan mengembalikan tabel dalam format yang sama dengan fungsi Folder.Contents. Setiap file dalam situs yang ditentukan, apa pun pustaka dokumennya, disertakan dalam tabel. Sebaliknya, pertimbangkan kueri berikut:

```
let
    Source = SharePoint.Contents("https://<tenant>.sharepoint.com/<site>/",
[ApiVersion = 15])
in
    Source
```

Kueri ini menggunakan fungsi SharePoint.Contents dan juga mengembalikan tabel dalam format yang sama dengan fungsi Folder.Contents. Namun, pustaka dokumen individual dikembalikan sebagai baris. Dengan demikian, Anda dapat menavigasi ke folder dan file yang terdapat dalam pustaka dokumen tertentu dengan mengeklik tautan Tabel di kolom Konten. Baik folder maupun file dikembalikan sebagai baris. Terakhir, pertimbangkan kueri ini:

```
let
    Source = SharePoint.Tables("https://<tenant>.sharepoint.com/<site>/",
[ApiVersion = 15])
in
    Source
```

Kueri ini menggunakan fungsi SharePoint.Tables. Fungsi ini mengembalikan tabel yang setiap barisnya berupa list atau pustaka dokumen dalam situs SharePoint yang ditentukan. Tabel tersebut memiliki tiga kolom, Id, Judul, dan Item. Kolom Id merupakan pengenal unik, kolom Judul berisi nama list atau pustaka dokumen, dan kolom Item berisi objek Tabel yang dapat dinavigasi untuk mencantumkan item individual dalam list atau pustaka dokumen.

2. Fungsi Identitas

Pustaka M standar mencakup beberapa fungsi yang terkait dengan penanganan identitas. Fungsi-fungsi ini umumnya tidak digunakan oleh pengguna, tetapi diklasifikasikan sebagai fungsi pembantu untuk implementasi internal dalam mesin M. Fungsi-fungsi ini mencakup yang berikut:

- *GoogleAnalytics.Accounts*: Mengembalikan akun Google Analytics yang tersedia dengan kredensial saat ini.
- *Identity.From*: Membuat identitas yang diberikan penyedia identitas sebagai fungsi dan nilai.
- *Identity.IsMemberOf*: Mengembalikan *true* atau *false* tergantung pada apakah identitas yang ditentukan merupakan anggota dari kumpulan identitas yang ditentukan sebagai record.
- *IdentityProvider.Default*: Mengembalikan penyedia identitas default untuk host saat ini.

Bahkan pengguna M tingkat lanjut jarang akan menemukan kasus penggunaan untuk fungsi-fungsi ini. Informasi lebih lanjut tentang fungsi-fungsi ini dapat ditemukan di *powerquery.how*.

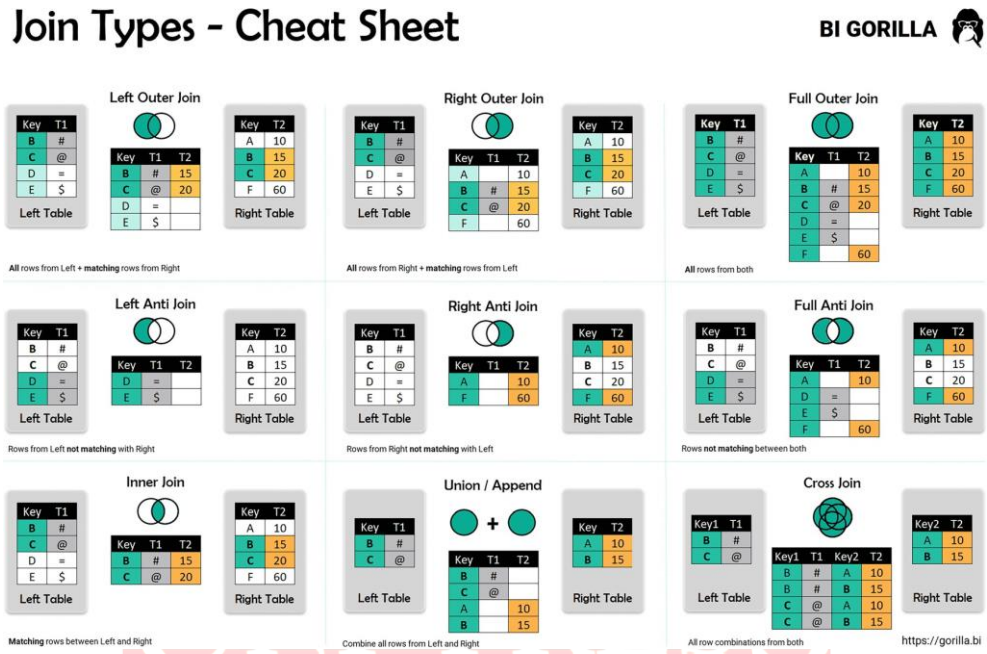
Kita sekarang telah membahas semua fungsi pengaksesan data yang tersedia di pustaka fungsi M standar. Sekarang kita akan mengalihkan perhatian kita ke beberapa fungsi transformasi data dasar yang memungkinkan kita untuk menggabungkan dan menyatukan data dari berbagai sumber.

I. Menggabungkan dan Menyatukan Data

Sejauh ini, kita telah berfokus pada akses dan pengambilan data dari sumber data individual dengan transformasi data minimal. Namun, selain mengakses data, salah satu kekuatan bahasa M adalah kemampuan untuk mengubah dan menggabungkan data tersebut. Ini memberikan kemampuan untuk menggabungkan data dari berbagai sumber bersama-sama seperti menggabungkan data inventaris dengan data penjualan atau menggabungkan data pelanggan dari berbagai sistem CRM. Oleh karena itu, kita akan membahas secara singkat lima fungsi dasar yang memungkinkan kita untuk menggabungkan dan menyatukan data kita. Kelima fungsi ini adalah:

- *Table.Combine*
- *Table.NestedJoin*
- *Table.Join*
- *Table.FuzzyNestedJoin*
- *Table.FuzzyJoin*

Saat Anda membaca bagian ini, akan sangat membantu untuk merujuk pada Join Types Gbeat Sbeet karya Rick de Groot, yang dapat ditemukan di sini: https://gorilla.bi/power-query/join-types/. Untuk memudahkan referensi, lembar contekan disertakan di sini:



Gambar 3. 13 Keluaran dari fungsi Table.NestedJoin

Mari kita mulai dengan fungsi *Table.Combine*.

1. Table.Combine

Fungsi *Table.Combine* menambahkan atau menggabungkan dua atau lebih tabel. Jelajahi berkas *Atlantic.pdf* yang disertakan dalam berkas *Atlantic* and *Pacific Hurricanes 1851-2014.zip* yang disertakan dalam repositori GitHub untuk buku ini. Jika Anda belum melakukannya, unduh berkas *Atlantic* and *Pacific Hurricanes 1851-2014.zip* dan ekstrak keempat berkas yang disertakan dalam berkas *.zip* ke satu folder di komputer lokal Anda. Ada dua berkas *CSV*, *atlantic.csv* dan *pacific.csv*, beserta dua berkas *PDF*, *atlantic.pdf* dan *pacific.pdf*.

Setelah Anda mengunduh dan mengekstrak berkas PDF, ikuti langkah-langkah berikut:

1) Di editor Power Query, klik tombol **New Source** pada tab **Home** pada pita dan pilih **More**….
2) Dalam dialog **Get Data**, pilih PDF dan tekan tombol **Connect**.

3) Gunakan jendela **File Explorer** untuk menavigasi ke folder tempat Anda mengekstrak file yang terdapat dalam Atlantic and *Pacific Hurricanes 1851-2014.zip*.
4) Pilih berkas **atlantic.pdf** dan tekan tombol **OK**.
5) Dalam dialog **Navigator**, pilih item **Table001** (**Page 1**) dan **Table002** (**Page 2**) lalu tekan tombol **OK**.
6) Ubah nama kueri ini menjadi *atlanticPDFTable1* dan *atlanticPDFTable2*.

Karena fungsi *PDF.Tables* gagal mengenali bahwa tabel yang dimaksud mencakup dua halaman, kita dapat menggunakan fungsi *Table.Combine* untuk memperbaiki masalah ini. Buat kueri baru dengan kode berikut:

```
let
    Source = Table.Combine( {atlanticPDFTable1, atlanticPDFTable2})
in
    Source
```

Seperti yang ditunjukkan, fungsi Table.Combine mengambil list tabel sebagai parameter pertamanya. Parameter opsional kedua tersedia untuk menentukan kolom yang akan dikembalikan. Misalnya, versi kueri ini hanya mengembalikan satu kolom, Kolom1, bukan 17 kolom seperti pada versi asli:

```
let
    Source = Table.Combine( {atlanticPDFTable1, atlanticPDFTable2},
{"Column1"})
in
    Source
```

Ubah nama kueri ini menjadi *atlanticPDFTableCombined*. Sekarang mari kita lihat penggabungan tabel.

2. *Table.NestedJoin* dan *Table.Join*

Fungsi *Table.NestedJoin* dan *Table.Join* menggabungkan atau menyatukan tabel. Kedua fungsi ini mirip satu sama lain tetapi memiliki nuansa tertentu. Menggabungkan tabel dapat membantu dalam analisis data, seperti memperkaya tabel pelanggan dengan informasi geografis dari tabel

geografi atau memperkaya tabel penjualan dengan informasi demografi pelanggan.

Melanjutkan contoh terakhir kita, dari bagian *Table.Gombine*, pertama-tama ganti nama Kolom1 dalam output kueri *atlanticPDFTableCombined* menjadi *Column0*. Selanjutnya, buat dua kueri berikut:

```
let
    Source = Table.NestedJoin(atlanticPDFTableCombined, {"Column0"},
atlanticPDFTable1, {"Column1"}, "atlanticPDFTable1", JoinKind.LeftOuter)
in
    Source
```

dan

```
let
    Source = Table.Join(atlanticPDFTableCombined, {"Column0"},
atlanticPDFTable1, {"Column1"}, JoinKind.LeftOuter)
in
    Source
```

Kita dapat langsung melihat perbedaan utama antara fungsi-fungsi ini dalam output dari kedua kueri ini. Kueri pertama menghasilkan tabel dua kolom dengan tajuk kolom *Column0* dan *atlanticPDFTable1* seperti yang ditunjukkan pada Gambar 3.14:

	Column0	atlanticPDFTable1
1	AL092011,	Table
2	20110821, 0000,	Table
3	20110821, 0600,	Table
4	20110821, 1200,	Table
5	20110821, 1800,	Table
6	20110822, 0000,	Table
7	20110822, 0600,	Table
8	20110822, 1200,	Table
9	20110822, 1800,	Table
10	20110823, 0000,	Table

Gambar 3. 14 Keluaran dari fungsi *Table.NestedJoin*

Kolom *atlanticPDFTable1* berisi nilai-nilai *Table*. Kolom ini dapat diperluas hanya ke kolom-kolom tertentu yang diinginkan menggunakan panah-panah divergen di tajuk kolom.

Sebaliknya, kueri kedua mengembalikan tabel dengan *Column0* dari kueri *atlanticPDFTableCombined* serta semua kolom dari kueri *atlanticPDFTable1* seperti yang ditunjukkan pada Gambar 3.15:

	Column0	Column1	Column2	Column3	Column4	Co
1	AL092011,	AL092011,	IRENE,	39,	*null*	
2	20110821, 0000,	20110821, 0000,	, TS, 15.0N,	59.0W,	*45*	
3	20110821, 0600,	20110821, 0600,	, TS, 16.0N,	60.6W,	*45*	
4	20110821, 1200,	20110821, 1200,	, TS, 16.8N,	62.2W,	*45*	
5	20110821, 1800,	20110821, 1800,	, TS, 17.5N,	63.7W,	*50*	
6	20110822, 0000,	20110822, 0000,	, TS, 17.9N,	65.0W,	*60*	
7	20110822, 0600,	20110822, 0600,	, HU, 18.2N,	65.9W,	*65*	
8	20110822, 1200,	20110822, 1200,	, HU, 18.9N,	67.0W,	*70*	
9	20110822, 1800,	20110822, 1800,	, HU, 19.3N,	68.0W,	*75*	
10	20110823, 0000,	20110823, 0000,	, HU, 19.7N,	68.8W,	*80*	

Gambar 3. 15 Output dari fungsi *Table.Join*

Kedua fungsi tersebut melakukan apa yang dikenal sebagai penggabungan, di mana kolom kunci dari kedua tabel dipilih dan kolom-kolom kunci ini dibandingkan untuk menemukan baris yang cocok. Tabel dan kolom kunci adalah empat parameter pertama untuk fungsi *Table.NestedJoin* dan *Table.Join*. Parameter pertama adalah tabel "kiri" dan parameter ketiga adalah tabel "kanan". Perhatikan bahwa kolom kunci (parameter kedua dan keempat) ditetapkan sebagai list.

Ini berarti penggabungan dapat terjadi pada beberapa kolom, bukan hanya pada satu kolom. Perlu dicatat juga bahwa kolom gabungan harus memiliki tipe data yang sama, misalnya teks, angka, atau tanggal.

Parameter kelima fungsi *Table.NestedJoin* adalah nama kolom untuk menyimpan nilai Tabel (baris yang cocok dari operasi penggabungan). Parameter kelima fungsi *Table.Join* adalah tipe penggabungan yang akan dilakukan, *joinKind*. Ini juga merupakan parameter keenam fungsi *Table.NestedJoin*.

Parameter *JoinKind* menggunakan enumerasi *JoinKind.Type*. Enumerasi ini dapat memiliki nilai berikut:

Tabel 3. 11 bergabung dengan enumerasi Kind

Friendly Name	Value	Comments
JoinKind.Inner	0	Hanya baris yang cocok yang dikembalikan.
JoinKind.LeftOuter	1	Semua baris dari tabel kiri, hanya baris yang cocok dari tabel kanan. Ini adalah default.
JoinKind.RightOuter	2	Semua baris dari tabel kanan, hanya baris yang cocok dari tabel kiri.
JoinKind.FullOuter	3	Semua baris dari kedua tabel dikembalikan. Baris yang tidak cocok memiliki nilai null untuk kolom.
JoinKind.LeftAnti	4	Semua baris dari tabel kiri yang tidak cocok dengan baris mana pun di tabel kanan.
JoinKind.RightAnti	5	Semua baris dari tabel kanan yang tidak cocok dengan baris mana pun di tabel kiri.

Siapa pun yang familier dengan SQL pasti familier dengan tipe gabungan ini.

Parameter keenam dari fungsi *Table.Join* adalah parameter *JoinAlgorithm*. Parameter ini memungkinkan Anda menentukan algoritma yang digunakan selama proses penggabungan. Parameter ini mengambil enumerasi JoinAlgorithm.Type yang dapat memiliki nilai berikut:

Tabel 3. 12 Enumerasi JoinKind

Friendly Name	Value	Comments
JoinAlgorithm.Dynamic	0	Secara otomatis menentukan algoritma gabungan yang akan digunakan. Ini adalah default.
JoinAlgorithm.PairwiseHash	1	Disarankan hanya untuk tabel kecil. Buffer kedua tabel hingga salah satu tabel terisi penuh. Bergantung pada tabel mana yang di-buffer, ia melakukan *LeftHash* atau *RightHash*.
JoinAlgorithm.SortMerge	2	Mengasumsikan kedua tabel diurutkan berdasarkan kolom kunci gabungannya. Melakukan penggabungan streaming, yang sangat cepat tetapi mengembalikan hasil yang salah jika tabel tidak diurutkan seperti yang diharapkan.
JoinAlgorithm.LeftHash	3	Menyangga baris tabel kiri. Melakukan penggabungan streaming untuk baris tabel kanan. Direkomendasikan saat tabel kiri

		berukuran kecil dan sebagian besar baris tabel kanan cocok. Pertimbangkan algoritme ini untuk menggabungkan tabel fakta dan tabel dimensi saat tabel dimensi berada di tabel kiri.
`JoinAlgorithm.RightHash`	`4`	Menyangga baris tabel kanan. Melakukan penggabungan streaming untuk baris tabel kiri. Direkomendasikan saat tabel kanan berukuran kecil dan sebagian besar baris tabel kiri cocok. Pertimbangkan algoritme ini untuk menggabungkan tabel fakta dan tabel dimensi saat tabel dimensi adalah tabel kanan.
`JoinAlgorithm.LeftIndex`	`5`	Beroperasi secara berkelompok menggunakan kunci dari tabel kiri untuk mengkueri tabel kanan. Direkomendasikan ketika tabel kanan berukuran besar dengan sedikit kecocokan yang diharapkan.
`JoinAlgorithm.RightIndex`	`6`	Beroperasi secara berkelompok menggunakan kunci dari tabel kanan untuk mengkueri tabel kiri. Direkomendasikan ketika tabel kiri berukuran besar dengan sedikit kecocokan yang diharapkan.

Menggunakan algoritma *JoinAlgorithm.SortMerge* dapat mempercepat operasi penggabungan secara drastis (berdasarkan urutan besarnya). Namun, Anda harus berhati-hati agar tabel Anda benar-benar diurutkan dengan benar atau Anda akan mendapatkan hasil yang salah. Dalam kasus kita, kita dapat menggunakan SortMerge dengan contoh kita sebagai berikut:

```
let
    Source = Table.Join(atlanticPDFTableCombined, {"Column0"},
atlanticPDFTable1, {"Column1"}, JoinKind.LeftOuter, JoinAlgorithm.SortMerge)
in
    Source
```

Terakhir, kedua fungsi tersebut memiliki parameter ketujuh, *keyEqualityComparers*, yang ditujukan hanya untuk penggunaan internal.

Penting untuk diketahui bahwa saat melakukan penggabungan atau gabung, kedua fungsi tersebut mengharuskan kolom kunci sama persis satu sama lain. Namun, M menyediakan fungsi penggabungan yang belum tentu demikian, jadi mari kita bahas hal tersebut selanjutnya.

3. *Table.FuzzyNestedJoin* dan *Table.FuzzyJoin*

Seperti yang ditunjukkan, fungsi *Table.FuzzyNestedJoin* dan *Table.FuzzyJoin* memungkinkan penggabungan atau penyambungan tabel bahkan ketika nilai kunci di kedua tabel tidak sama persis. Ini dikenal sebagai pencocokan fuzzy.

Parameter untuk kedua fungsi ini analog dengan padanan non-fuzzy mereka, *Table.NestedJoin* dan *Table.Join*, dengan pengecualian bahwa *Table.FuzzyJoin* tidak mendukung parameter joinAlgorithm.

Algoritme yang digunakan oleh Power Query untuk mengukur kesamaan antara pasangan disebut algoritme kesamaan Jaccard. Algoritme kesamaan Jaccard hanya mengambil jumlah item yang muncul di kedua pasangan (interseksi atau gabungan dalam) dan membaginya dengan jumlah total item di kedua pasangan (gabungan). Ini menghasilkan angka antara 0 dan 1 dengan 0 menunjukkan tidak ada item yang sama (kurang mirip) dan 1 menunjukkan bahwa kedua pasangan memiliki item yang sama (lebih mirip).

Parameter terakhir untuk setiap fungsi adalah parameter *joinOptions*. Parameter *joinOptions* adalah record opsi untuk mengendalikan kriteria pencocokan fuzzy yang digunakan untuk menggabungkan tabel. Record ini dapat berisi bidang-bidang berikut:

- *ConcurrentRequests*: Defaultnya adalah 1. Menentukan jumlah utas paralel yang akan digunakan untuk pencocokan fuzzy. Dapat berupa angka antara 1 dan 8.
- *Culture*: Defaultnya adalah "", yang merupakan budaya Inggris yang tidak berubah. Jika ditentukan, dapat mencocokkan baris berdasarkan aturan khusus budaya. Budaya ditentukan melalui standar ISO 639 seperti *ja-JP*, *en-US, en-UK*.
- *IgnoreCase*: Dapat bernilai *true* atau *false* dan defaultnya adalah benar. Mengabaikan huruf besar-kecil saat mencocokkan, jadi *grapes*, "*Grapes*", "*gRapes*", "*GrApEs*", dan "*grapeS*" semuanya akan cocok.

- *IgnoreSpace*: Dapat bernilai *true* atau *false* dan defaultnya adalah benar. Mengizinkan penggabungan bagian teks untuk menemukan kecocokan. Misalnya, "*grapes*", "*gra pes*", dan "*grape s*" semuanya akan cocok.
- *NumberOfMatches*: Nilai default untuk semua baris yang cocok. Menentukan bilangan bulat yang mewakili jumlah maksimum baris yang cocok yang dikembalikan.
- *SimilarityColumnName*: Menentukan nama untuk kolom yang mengembalikan kesamaan yang dihitung dari baris yang cocok. Kesamaan baris yang cocok dihitung sebagai nilai antara 0,00 dan 1,00. Nilai default untuk *null*, artinya tidak ada kolom tersebut yang dikembalikan. Baris dengan kesamaan di atas opsi *Threshold* yang ditentukan dianggap sebagai baris yang cocok.
- *Threshold*: Angka desimal antara nilai 0,00 dan 1,00. Nilai default untuk 0,80. Menentukan ambang batas untuk baris yang cocok berdasarkan algoritma pencocokan fuzzy.
- *TransformationTable*: Tabel dengan kolom *From* dan kolom To. Memungkinkan baris yang cocok ditentukan berdasarkan tabel pemetaan kustom ini. Misalnya, tabel transformasi yang menyertakan baris dengan *grapes* dan *kismis* akan cocok dengan baris yang kolom kuncinya berisi nilai *grapes* dan *kismis* atau *grapes* itu enak dan *kismis* itu enak.

Ini melengkapi eksplorasi kita tentang fungsi pengaksesan data serta menggabungkan dan menggabungkan data.

J. Ringkasan

Pustaka fungsi M standar berisi banyak fungsi yang dirancang untuk memfasilitasi akses dan pengambilan data dari beragam sistem dan format penyimpanan data. Selain itu, bahasa M juga menyediakan fungsi untuk menggabungkan dan menyatukan data. Contoh-contoh praktis disertakan di

seluruh bab ini untuk memungkinkan pembaca bereksperimen dan memvisualisasikan keluaran dari fungsi-fungsi ini.

Dalam bab ini, kita mengeksplorasi fungsi-fungsi pengaksesan data untuk berbagai format file, folder, konten web, basis data, kubus, dan protokol data standar seperti OData dan ODBC. Kita juga membahas berbagai fungsi biner, fungsi identitas, dan fungsi khusus untuk mengakses sistem perangkat lunak populer seperti Microsoft Exchange dan Microsoft SharePoint. Terakhir, kita mengeksplorasi lima fungsi untuk menggabungkan dan menyatukan data, termasuk kemampuan untuk melakukan pencocokan fuzzy saat menggabungkan tabel.

Dalam bab berikutnya, kita terus mengeksplorasi bahasa M dengan berusaha memahami nilai dan ekspresi secara lebih rinci.

K. Daftar Pustaka

1. Kasun Bandara and Rob J Hyndman and Christoph Bergmeir. (2021). MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns. arXiv:2107.13462 [stat.AP]. https://arxiv.org/abs/2107.13462.
2. Hochenbaum, J., Vallis, O., & Kejariwal, A. (2017). Automatic Anomaly Detection in the Cloud Via Statistical Learning. ArXiv, abs/1704.07706. https://arxiv.org/abs/1704.07706.

L. Penutup

1. Tes Formatif

Tabel 3. 13 Tes Formatif Bab 3

No	Soal	Bobot
1.	Apa itu Power Query dan bagaimana fungsinya dalam pengolahan data? Berikan contoh penggunaannya dalam konteks analisis data.	10

2.	Deskripsikan langkah-langkah yang diperlukan untuk mengambil data dari sumber eksternal menggunakan Power Query. Apa saja tantangan yang mungkin dihadapi?	10
3.	Diskusikan berbagai metode transformasi data yang tersedia di Power Query. Bagaimana transformasi ini dapat meningkatkan kualitas data?	10
4.	Bagaimana cara menggunakan filter dalam Power Query? Berikan contoh situasi di mana penggunaan filter sangat penting.	10
5.	Jelaskan proses penggabungan data dari beberapa sumber dalam Power Query. Apa manfaat dari penggabungan data ini?	10
6.	Sebutkan beberapa fungsi yang sering digunakan dalam Power Query dan jelaskan bagaimana fungsi-fungsi tersebut dapat membantu dalam analisis data.	10
7.	Bagaimana cara menyimpan dan memperbarui data yang telah diproses di Power Query? Diskusikan pentingnya pembaruan data secara berkala.	10
8.	Identifikasi beberapa kesalahan umum yang sering terjadi saat menggunakan Power Query. Bagaimana cara menghindari kesalahan tersebut?	10
9.	Berikan contoh studi kasus di mana Power Query digunakan untuk menyelesaikan masalah analisis data. Apa hasil yang diperoleh dari penggunaan Power Query dalam kasus tersebut?	10
10.	Bandingkan Power Query dengan alat pengolahan data lainnya. Apa kelebihan dan kekurangan Power Query dibandingkan dengan alat-alat tersebut?	10

BAB 4

Memahami Nilai dan Ekspresi

Topik Bab

Pada bab ini, topik yang akan dibahas adalah:

- Memperkenalkan jenis-jenis nilai
- Ekspresi
- Operator
- Struktur control
- Petugas Pencacah

A. Pendahuluan

Sekarang setelah kita membahas ikhtisar M, meninjau elemen **User Interface (UI)** Power Query, dan membahas cara mengakses dan menggabungkan data kita, kita siap untuk menyelami komponen spesifik bahasa M. Nilai terdiri dari komponen pertama, yang secara alami mengikuti dari bab sebelumnya karena setiap kali Anda memasukkan file ke Power Query, masing-masing elemen data individual adalah nilai.

Jika Anda menganggap Power Query dan M sebagai lab tempat Anda membersihkan dan membentuk data Anda sehingga siap untuk pemodelan dan pelaporan data, maka nilai adalah atom kode M – unit tunggal terkecil. Nilai-nilai ini digabungkan untuk membentuk ekspresi, menjadikannya molekul lab kita. Satu atau beberapa ekspresi dapat diintegrasikan untuk membentuk senyawa yang sangat kuat (yaitu, kueri) yang memungkinkan Anda mengubah data Anda dalam jumlah cara yang tak terbatas.

Dalam bab ini, kita memeriksa nilai-nilai ini secara saksama, melihat semua jenis yang akan Anda temui, dan bagaimana jenis nilai tersebut secara langsung memengaruhi apa yang dapat (dan tidak dapat) Anda lakukan dengannya. Kita juga melihat struktur dasar ekspresi dan kueri, serta ikatan yang menghubungkan nilai-nilai dalam ekspresi – operator dan struktur kontrol.

Bab ini membahas semua jenis nilai yang berbeda, termasuk isu-isu utama yang perlu diperhatikan saat bekerja dengan setiap jenis nilai. Kita juga mempelajari bagaimana operator dan struktur kontrol digunakan untuk menggabungkan nilai-nilai ke dalam ekspresi dan kueri, dan, akhirnya, bagaimana enumerator digunakan untuk menentukan bagaimana nilai-nilai fungsi tertentu beroperasi. Bab ini mencakup topik-topik berikut:

- Memperkenalkan jenis-jenis nilai
- Ekspresi
- Operator
- Struktur control
- Petugas Pencacah

1. Kasus Pemantik Berfikir Kritis: Analisis Data Penjualan

Sebuah perusahaan retail ingin menganalisis data penjualan mereka untuk memahami tren dan performa produk. Data penjualan disimpan dalam format CSV dan mencakup kolom seperti tanggal, nama produk, jumlah terjual, dan total pendapatan.

1) Identifikasi Nilai Tanggal:

   Bagaimana cara Anda mengelola nilai tanggal dalam data penjualan? Apa pentingnya format tanggal yang konsisten?

2) Nilai Angka:

   Bagaimana Anda akan memastikan bahwa kolom jumlah terjual dan total pendapatan terkonversi dengan benar menjadi nilai numerik? Apa saja langkah-langkah yang perlu diambil untuk menangani nilai yang hilang atau tidak valid?

3) Penggunaan Fungsi:

   Fungsi apa yang bisa Anda gunakan untuk menghitung total pendapatan per produk? Bagaimana cara Anda menggunakan fungsi tersebut dalam Power Query?

4) Penerapan Struktur Kontrol:

   Jika Anda ingin menandai produk yang memiliki penjualan di bawah rata-rata, bagaimana Anda akan menggunakan struktur kontrol if-then-else dalam ekspresi M?

5) Pencocokan Fuzzy:

   Jika terdapat variasi dalam nama produk (misalnya, "Kopi" dan "Kopi A"), bagaimana Anda dapat menggunakan pencocokan fuzzy untuk menggabungkan data penjualan dari kedua nama tersebut?

B. Memperkenalkan Jenis-Jenis Nilai

Bahasa M mengenali 15 jenis nilai yang berbeda, yang tersebar di beberapa kategori berikut:

- **Nilai primitif**: Elemen data individual yang diperlakukan M sebagai satu unit untuk tujuan pemrosesan dan operasi. Dalam konteks DAX, ini disebut sebagai nilai skalar.
- **Nilai terstruktur**: Ini dapat menampung beberapa nilai dengan struktur tertentu, meskipun mungkin saja nilai terstruktur tertentu hanya menampung satu nilai atau tidak memiliki nilai sama sekali. Tiga jenis nilai terstruktur adalah list, record, dan tabel, yang akan dibahas secara terperinci di Bab 6, Nilai Terstruktur. Nilai terstruktur yang menampung nilai terstruktur lainnya (juga disebut sebagai struktur bersarang) akan dibahas secara mendalam di Bab 8, Bekerja dengan Struktur Bersarang.
- **Nilai fungsi**: Fungsi adalah nilai yang mewakili pemetaan dari sekumpulan nilai argumen ke satu nilai. Bahasa M mencakup lebih dari 700 fungsi yang ditentukan, dan pengguna memiliki kemampuan untuk mengembangkan fungsi kustom mereka sendiri yang dapat digunakan kembali dan diterapkan di beberapa kueri. Pengembangan fungsi kustom akan dibahas dalam Bab 9, Parameter dan Fungsi Kustom.
- **Nilai tipe**: Ini dapat dianggap sebagai nilai meta yang memberikan informasi tentang tipe data dari nilai lain. Klasifikasi nilai ke dalam tipe data yang sesuai adalah proses yang dibahas secara luas dalam Bab 5, Memahami Tipe Data.

Tabel berikut memberikan ringkasan singkat dari masing-masing dari 15 nilai yang dikenali di keempat kelompok ini. Informasi tambahan dapat ditemukan di seluruh bab ini serta di Bab 5, 6, 8, dan 9. Perhatikan bahwa istilah **literal** mengacu pada representasi tekstual dari suatu nilai sebagaimana ditulis dalam bahasa pemrograman.

Tabel 4. 1 Jenis nilai dalam M

Kind	Literal	Deskripsi
Primitive		

Date	*#date(2023, 12, 29)*	Nilai tanggal ditentukan dalam format: *#date(year, month, day).*
Binary	*#binary("QRST")*	Data biner direpresentasikan sebagai serangkaian nilai byte, yang paling sering digunakan dalam penyerapan konten web, konten multimedia, dan untuk konektor khusus.
DateTime	*#datetime(2023, 11, 26, 24, 59, 01)*	Nilai tanggal ditentukan dalam format: *#datetime(year, month, day, hours, minutes, seconds).*
DateTimeZone	*#datetimezone(2023, 11, 26, 24, 59, 1, 08, 00)*	Nilai tanggal ditentukan dalam format: *#datetime(year, month, day, hours, minutes, seconds, offset hours, offset minutes).*
Duration	*#duration(1, 5, 30, 0)*	Nilai durasi mewakili periode tertentu, yang didefinisikan sebagai *#duration(days, hours, minutes, seconds).*
Logical	*true false*	Salah satu dari dua nilai kebenaran, true atau false, biasanya digunakan dalam perbandingan dan kondisi.
Null	*null*	Nilai yang tidak diketahui atau hilang dilambangkan dengan null.
Number	*-1 0 1 1.5 2.3e-5*	Nilai numerik bisa berupa bilangan bulat atau desimal.
Text	*"Canada"*	Data tekstual direpresentasikan dalam tanda kutip ganda.
Time	*#time(23, 59, 15)*	Nilai waktu ditentukan dalam format: *#time(hours, minutes, seconds).*

Nilai-nilai terstruktur		
`List`	`{ 1, 2, 3 }`	Urutan nilai yang didefinisikan dalam tanda kurung kurawal.
`Record`	`[ A = 2023, B = "Argentina" ]`	Kumpulan nilai bernama yang dibundel dalam tanda kurung siku.

Table	`#table( { "a", "b"}, {{1,2},{3,4}})`	Kumpulan baris, dimana setiap baris adalah record.
Nilai Fungsi		
Function	`(x) => x * 2`	Ekspresi yang mengambil satu atau lebih nilai input dan mengembalikan nilai.
Ketik nilai		
Type	`type date, type table`	Tipe digunakan untuk mengklasifikasikan nilai berdasarkan jenisnya.

Masing-masing dari 15 jenis nilai tersebut terstruktur secara unik, memiliki kegunaan yang berbeda, dan bekerja dengan fungsi kode M yang berbeda. Selain membahas karakteristik ini untuk setiap jenis nilai, bagian ini juga akan menyoroti masalah-masalah kecil yang terkait dengan setiap jenis nilai yang dapat menggagalkan Anda jika Anda tidak mengetahuinya.

Sekarang setelah Anda diperkenalkan dengan jenis-jenis nilai yang akan Anda temui di M, kita akan membahasnya masing-masing secara lebih rinci.

1. Nilai Biner

Nilai biner adalah nilai yang disimpan sebagai byte (urutan angka nol dan satu). Nilai ini biasanya berupa file khusus yang dapat dibaca oleh program, seperti gambar. Power Query memproses data biner dalam tiga kasus penggunaan utama:

- Penyerapan data lokal (melalui fungsi *File.Contents*)
- Penyerapan data dari web (melalui fungsi *Web.Contents*)
- Penyerapan data melalui konektor kustom

Selain itu, saat nilai dimasukkan oleh pengguna melalui tombol **Enter Data**, nilai tersebut diubah menjadi biner dalam proses penyimpanan nilai tersebut. Misalnya, jika kita memasukkan data berikut secara manual menggunakan opsi **Enter Data**, data tersebut diubah menjadi biner lalu dikodekan menjadi teks:

	Author	Country
1	Brian	USA
2	Greg	USA
3	Melissa	NED
4	Rick	NED

```
let
    Source = Table.FromRows(Json.Document(Binary.Decompress(Binary.FromText
        ("i45WcirKTMxT0lEKDXZUitWJVnIvSk1H4vqm5mQWFycCRfxcXcAiQZnJ2TBuLAA=", BinaryEncoding.Base64),
        Compression.Deflate)), let _t = ((type nullable text) meta [Serialized.Text = true]) in type table [Author
        = _t, Country = _t])
in
    Source
```

Gambar 4. 1 Pembuatan nilai biner melalui opsi Enter Data

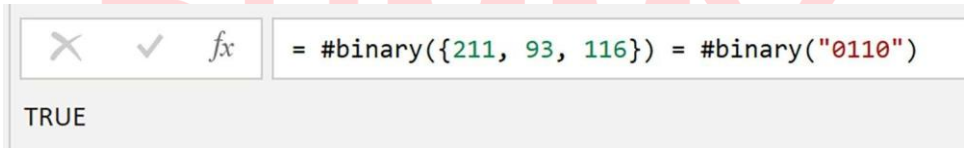
Rincian tambahan mengenai proses ini dapat ditemukan di Bab 3, Mengakses dan Menggabungkan Data.

Struktur

Kode berikut membuat nilai biner dari list angka atau nilai teks berkode Base64:

```
#binary( value as any ) as any
```

Kedua contoh yang ditunjukkan dalam gambar tangkapan layar berikut akan menghasilkan nilai biner yang sama:



Gambar 4. 2 Pembuatan langsung nilai biner dari teks dan list angka

Fungsi Terkait

Ada 40 fungsi biner terkait, yang mencakup tujuh kategori berikut:

- **Buffering dan kompresi**: Fungsi ini menyangga nilai biner ke dalam memori dan mengompresi serta mendekompresi nilai biner.
- **Pengurutan byte**: Fungsi ini menentukan pengurutan byte dari nilai biner.
- **Pembuatan dan konversi**: Fungsi ini membuat dan mengubah nilai biner ke dan dari jenis nilai lainnya.

- **Informasional**: Fungsi ini mengembalikan informasi tentang panjang dan jenis konten yang disimpulkan dari nilai biner.
- **Reading**: Kategori ini mewakili lebih dari setengah fungsi biner terkait secara keseluruhan dan menentukan bagaimana nilai biner yang berbeda harus dibaca
- **Transformasi**: Fungsi ini menggabungkan, membagi, dan mengekstrak bagian tertentu dari nilai biner
- **Viewing**: Fungsi ini digunakan untuk membuat dan memanipulasi tampilan data biner tanpa mengubah konten biner yang mendasarinya.

Pertimbangan Khusus

Pertimbangan khusus saat membuat nilai biner adalah struktur data yang unik. Gambar, teks, data numerik, dan banyak jenis data lainnya dapat ditangani sebagai data biner dalam M. Memahami bagaimana masing-masing jenis nilai yang berbeda ini disusun, dikodekan, dan dienkode sangat penting untuk bekerja dengan nilai biner dengan sukses.

Penggunaan nilai biner dalam konektor kustom dibahas dalam Bab 16, Mengaktifkan Ekstensi, sementara peran nilai dan fungsi biner dalam penyerapan data lokal dan web dijelaskan lebih lanjut dalam Bab 3, Mengakses dan Menggabungkan Data.

2. Keluarga nilai Tanggal/Waktu

Banyak nilai data yang akan Anda temukan terhubung ke momen tertentu dalam waktu. Baik Anda menganalisis pola dalam penjualan produk, peningkatan kemampuan model AI, perubahan intensitas badai, atau hasil lari cepat 100 meter Olimpiade, mengetahui kapan peristiwa ini terjadi sangatlah relevan. Oleh karena itu, kelompok nilai ini, yang membantu kita memahami kapan sesuatu terjadi, adalah salah satu jenis data yang paling signifikan dan umum ditemui dalam M.

Selain itu, penanganan nilai-nilai ini secara terampil merupakan keterampilan penting bagi pengembang Power BI, karena analisis apa pun dengan elemen temporal memerlukan tabel *Date/Calendar* dalam model data Power BI Anda. Jika Anda tidak mendapatkan tabel ini dari gudang data atau sumber lain, membuatnya di Power Query menggunakan M sangat disarankan. Pendekatan ini dianggap sebagai praktik terbaik. Untuk mempelajari lebih dalam tentang cara bekerja dengan tanggal, waktu, dan durasi, termasuk topik lanjutan seperti analisis data temporal, pastikan untuk memeriksa Bab 10, Berurusan dengan Tanggal, Waktu, dan Durasi.

Karena semua fungsi ini membantu mengkarakterisasi aspek temporal data Anda, kita membahas nilai Tanggal, Waktu, *DateTime*, *DateTimeZone*, dan Durasi secara bersamaan di bagian ini. Namun, pembahasan tentang perhitungan yang melibatkan nilai Tanggal, Waktu, DateTime, DateTimeZone, dan Durasi dapat ditemukan di bagian selanjutnya yang membahas operasi.

Nilai Tanggal

Tanggal sering kali merupakan level paling terperinci tempat data dikumpulkan. Oleh karena itu, jika Anda memahami cara Power Query menangani tanggal dan mempelajari cara menyesuaikannya menggunakan M, Anda akan merasa lebih mudah menangani jenis nilai lain dalam kategori ini. Hal ini karena banyak konsep dan pola umum yang sama diterapkan ke yang lain.

Struktur

Nilai tanggal dapat dibuat menggunakan struktur berikut:

```
#date(
  year as number,
  month as number,
  day as number,
) as date
```

Input memiliki batasan tambahan berikut atau ekspresi akan mengembalikan kesalahan:

```
1 ≤ year ≤ 9999
1 ≤ month ≤ 12
1 ≤ day ≤ 31
```

Ekspresi literal #date peka huruf besar-kecil tanpa spesifikasi alternatif – dengan demikian, *#Date* dan *#DATE* akan mengembalikan kesalahan.

Contohnya adalah *#date(2023, 11, 1)*, yang mengembalikan nilai yang mewakili 1 November 2023.

Fungsi Terkait

Ada hampir 60 fungsi yang membantu dalam memanipulasi nilai Tanggal. Secara umum, fungsi-fungsi ini terbagi dalam kategori berikut:

- Fungsi *Date.Add* (misalnya, *Date.AddMonths*): Fungsi-fungsi ini menerima nilai Tanggal, DateTime, atau DateTimeZone, dan angka, dan menambahkan jumlah periode waktu yang ditentukan ke nilai asli. Perhatikan bahwa angka tersebut dapat negatif, untuk mundur ke masa lalu.
- Fungsi *Date.Component*: Fungsi-fungsi ini menerima semua jenis nilai tanggal dan mengembalikan komponen yang ditentukan sebagai angka (misalnya, *Date.Month*) atau nilai teks (misalnya, *Date.MonthName*).
- Fungsi *Date.Start* dan *Date.End* (misalnya, *Date.EndOfMonth*): Fungsi-fungsi ini menerima nilai Tanggal, *DateTime*, atau *DateTimeZone* dan mengembalikan jenis nilai yang sama untuk tanggal yang memenuhi fungsi tersebut. Misalnya, menggunakan contoh sebelumnya,

Date.EndOfMonth(#date(2023, 11, 1)) mengembalikan tanggal 30 November 2024.

- Fungsi *Date.IsIn* (misalnya, *Date.IsInPreviousYear*): Fungsi ini menerima nilai Tanggal, *DateTime*, atau *DateTimeZon*e dan memberikan hasil benar atau salah. Komponen *DateTime* yang relevan, yang digunakan untuk mengevaluasi nilai logika ekspresi, didasarkan pada nilai sistem pada tanggal/waktu evaluasi. Misalnya, pertimbangkan untuk mengevaluasi fungsi *Date.IsInPreviousYear(# date(2023, 11, 1))* pada tanggal 1 Agustus 2023. Fungsi ini memeriksa apakah tanggal yang diberikan, 1 November 2023, jatuh pada tahun sebelumnya, yaitu 2022. Karena tanggal tersebut jatuh pada tahun 2023 dan bukan 2022, fungsi tersebut mengembalikan nilai salah.
- Fungsi *Date.To* dan *Date.From*: Fungsi-fungsi ini mengubah nilai Tanggal menjadi jenis nilai lain atau mengekstrak nilai tanggal dari jenis nilai yang berbeda. Misalnya, fungsi *Date.ToText(#date(2023, 11, 1))* mengembalikan 11/1/2023 di komputer saya, yang mencerminkan pengaturan lokal default untuk format tanggal. Pengaruh pengaturan lokal, termasuk bagaimana parameter budaya memengaruhi konversi tersebut, dibahas secara menyeluruh di Bab 5, Memahami Jenis Data.

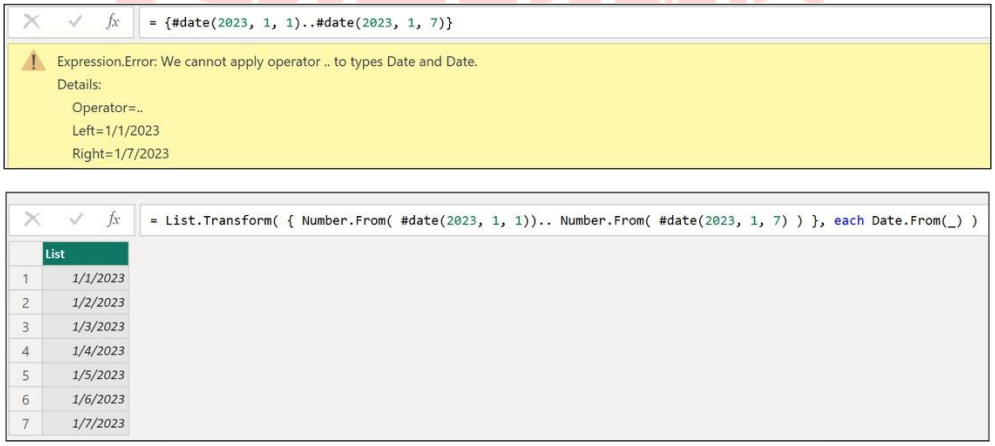
  Fungsi *Date.ToText* juga menyertakan parameter format opsional yang memungkinkan Anda menentukan format output. Jadi, sementara *Date.ToText(# date(2023, 11, 1))* mengembalikan 11/1/2023, *Date.ToText(# date(2023, 11, 1), [Format="dd MMM yyyy"])* mengembalikan 1 Nov 2023. Informasi tambahan tentang fungsi ini dan fungsi tanggal lainnya dapat ditemukan di Bab 10, Berurusan dengan Tanggal, Waktu, dan Durasi.

Pertimbangan Khusus

Pertimbangan khusus saat menggunakan nilai Tanggal adalah sistem tanggal di Power Query. Meskipun struktur *#date(year, month, day)* adalah

cara utama untuk menentukan tanggal di Power Query, akan sangat membantu jika mengetahui sistem tanggal dasar yang digunakan Power Query (baik di Excel maupun Power BI Desktop). Tanggal dihitung sebagai bilangan bulat yang mewakili jumlah total hari yang telah berlalu sejak 30 Desember 1899.

Hal ini berguna untuk fungsi yang tidak menerima tanggal sebagai input tetapi menerima angka. Misalnya, ekspresi pembuatan list menggunakan formulir A..B, yang akan dibahas secara terperinci di Bab 6, akan mengembalikan kesalahan jika Anda memasukkan input sebagai tanggal secara langsung. Namun, seperti yang diperlihatkan dalam tangkapan layar berikut, jika Anda terlebih dahulu mengubah input tanggal menjadi nomor seri dasar, mengevaluasi ekspresi menggunakan bilangan bulat, lalu mengubah list yang dihasilkan kembali menjadi tanggal, hasilnya akan sempurna:



Gambar 4. 3 Mengubah tanggal menjadi nomor seri yang mendasarinya sebelum membuat list

Ini menyimpulkan ikhtisar kita mengenai nilai Tanggal, tetapi konsep dan aplikasi lanjutan yang melibatkan keluarga nilai Tanggal/Waktu akan dibahas secara lebih terperinci di Bab 10, Mengenai Tanggal, Waktu, dan Durasi.

Nilai Waktu

Nilai waktu dalam M mewakili titik tertentu dalam sehari, yang dinyatakan menggunakan siklus 24 jam. Di sini, kita menyajikan beberapa

dasar nilai waktu. Informasi lebih lanjut dapat ditemukan di Bab 10, Berkaitan dengan Tanggal, Waktu, dan Durasi.

Struktur

Nilai waktu dapat dibuat menggunakan struktur berikut:

```
#time(
  hour as number,
  minute as number,
  second as number,
) as date
```

Input memiliki batasan berikut atau ekspresi akan menghasilkan kesalahan:

- $0 \leq$ jam ≤ 24
- $0 \leq$ menit ≤ 59
- $0 \leq$ detik < 60

Jika nilai jam adalah 24, maka nilai menit dan detik harus 0. Selain itu, seperti halnya ekspresi literal #date, ekspresi #time peka huruf besar-kecil. Beberapa contohnya adalah sebagai berikut:

```
#time( 7, 23, 30)  // returns 7:23:30 AM
#time(23, 59, 59)  // returns 11:59:59 PM
#time(24, 0, 0)    // returns 12:00:00 AM
```

Fungsi Terkait

Sementara keluarga tanggal memiliki hampir 60 fungsi terkait, M hanya memiliki 9 fungsi waktu, yang terbagi dalam tiga kategori:

- *Fungsi Time.To* dan Time.From: Fungsi-fungsi ini memungkinkan Anda mengonversi nilai waktu ke nilai record atau teks, dan mengekstrak nilai waktu dari jenis nilai lain yang sesuai.

- *Time.EndOfHour* dan *TimeStartOfHour*: Fungsi-fungsi ini memungkinkan Anda menentukan waktu mulai dan berakhirnya komponen jam dari nilai waktu tertentu.
- *Time.Hour*, *Time.Minute*, dan *Time.Second*: Fungsi-fungsi ini memungkinkan Anda mengekstrak komponen jam, menit, dan detik sebagai nilai angka dari nilai waktu.

Pertimbangan Khusus

Berikut ini adalah beberapa pertimbangan khusus saat bekerja dengan nilai Waktu:

- **Kesadaran lokasi**: #time tidak menyertakan informasi apa pun tentang zona waktu. Jika Anda perlu menyatakan waktu relatif terhadap lokasi, *#datetimezone* harus digunakan sebagai gantinya, dan, jika perlu, komponen waktu dapat diekstraksi dari nilai tersebut.
- **Presisi**: Power Query mendukung pelacakan dan penghitungan nilai waktu hingga ke level mikrodetik (sepersejuta detik), tetapi kecil kemungkinan pengguna akan memerlukan level presisi yang lebih tinggi dari level milidetik.

Nilai *DateTime*

Nilai *DateTime* merupakan gabungan dari nilai Tanggal dan Waktu. Informasi dasar tentang nilai *DateTime* disajikan di sini, dengan informasi lebih lanjut di Bab 10, Berkaitan dengan Tanggal, Waktu, dan Durasi.

Struktur

Nilai *DateTime* dapat dibuat menggunakan struktur berikut:

```
#datetime(
  year as number,
  month as number,
  day as number,
  hour as number,
  minute as number,
  second as number
) as datetime
```

Input memiliki batasan tambahan berikut atau ekspresi akan menghasilkan kesalahan:

- $1 \leq year \leq 9999$
- $1 \leq month \leq 12$
- $1 \leq day \leq 31$
- $0 \leq hour \leq 23$
- $0 \leq minute \leq 59$
- $0 \leq second < 60$

Selain itu, seperti halnya ekspresi literal *#date*, ekspresi *#datetime* peka huruf besar-kecil. Contoh berikut menghasilkan nilai 11/1/23 7:15:00 AM:

```
#datetime( 2023, 11, 1, 7, 15, 0 )
```

Fungsi Terkait

Ada 25 fungsi yang terkait dengan *DateTime*, yang sebagian besar mencerminkan kategori *Component*, *IsIn*, dan *To/From* dari fungsi *#date*. Namun, ada juga tiga fungsi DateTime yang tidak memiliki analog dalam fungsi terkait *#date*.

- *DateTime.LocalNow* dan *DateTime.FixedLocalNow*: Kedua fungsi mengembalikan nilai datetime yang ditetapkan ke tanggal dan waktu saat ini pada sistem. Perbedaannya adalah yang pertama dapat berubah selama eksekusi ekspresi, sedangkan yang terakhir tetap konstan.

Hal penting yang perlu diperhatikan adalah bahwa fungsi-fungsi ini mengembalikan nilai datetime yang terkait dengan sistem tempat kueri dieksekusi. Ini berarti jika kumpulan data Anda diperbarui di Australia tetapi Anda berada di Kanada, fungsi tersebut akan menghasilkan waktu di Australia. Jika Anda ingin mengembalikan waktu di Kanada, Anda perlu menambahkan zona waktu untuk menyesuaikan waktu guna memperhitungkan perbedaan zona waktu.

- *DateTime.AddZone*: Fungsi ini mengambil nilai *#datetime* yang valid, nilai jam, dan nilai menit yang dapat berupa null sebagai input dan mengembalikan nilai *#datetimezone*, untuk menambahkan elemen lokasi ke nilai *#datetime* guna menyesuaikan waktu setempat dan/atau waktu musim panas.

Pertimbangan Khusus

Nilai *DateTime* harus digunakan dengan hati-hati dan hanya jika benar-benar diperlukan karena nilai tersebut memiliki dua kelemahan utama yang relatif terhadap nilai Tanggal dan Waktu:

- Penggunaan nilai *datetime* dapat secara drastis meningkatkan kardinalitas kolom yang memuatnya, yang dapat memperbesar ukuran file yang diperlukan untuk menyimpan semua nilai yang berbeda tersebut, serta berdampak negatif pada kinerja dan penggunaan memori. Dalam kebanyakan kasus, praktik yang lebih baik adalah membagi datetime menjadi bidang tanggal dan waktu yang terpisah, dengan masing-masing digabungkan dalam hubungan satu-ke-banyak dengan tabel Tanggal dan Waktu yang terpisah dalam model data Power BI (atau Excel).
- Nilai *DateTime* tidak berisi informasi mengenai zona waktu. Ini berarti bahwa jika Anda membandingkan nilai datetime di berbagai lokasi, Anda perlu memperhitungkan perbedaan zona waktu.

Nilai *DateTimeZone*

Nilai *DateTimeZone* sangat mirip dengan *DateTime*, dengan satu-satunya perbedaan adalah bahwa keduanya menyertakan nilai offset zona waktu yang dinyatakan dalam jam dan menit opsional, yang mewakili perbedaan antara waktu lokal dan **Coordinated Universal Time (UTC)**. UTC adalah standar dunia, berdasarkan waktu atom dan digunakan untuk navigasi dan pengukuran waktu ilmiah.

UTC menggantikan standar sebelumnya, Greenwich Mean Time, dan semua zona waktu dapat dinyatakan sebagai offset dari UTC. Bab 10, Berkaitan dengan Tanggal, Waktu, dan Durasi, berisi informasi tambahan tentang nilai *DateTimeZone*.

Struktur

Nilai *DateTimeZone* dapat dibuat menggunakan struktur berikut:

```
#datetimezone(
  year as number,
  month as number,
  day as number,
  hour as number,
  minute as number,
  second as number,

  offsetHours as number,
  offsetMinutes as nullable number,

) as datetimezone
```

Input memiliki batasan tambahan berikut atau ekspresi akan menghasilkan kesalahan:

- $1 \leq year \leq 9999$
- $1 \leq month \leq 12$
- $1 \leq day \leq 31$
- $0 \leq hour \leq 23$
- $0 \leq minute \leq 59$

- $0 \leq$ second < 60
- $-14 \leq$ offset-hours + offset-minutes / $60 \leq 14$

Selain itu, seperti halnya ekspresi literal *#datetime*, ekspresi *#datetimezone* peka huruf besar-kecil.

Kode berikut mengembalikan nilai 11/1/2023 7:15:00 AM -05:00: *#datetimezone(2023, 11, 1, 7, 15, 0, -5, 0).*

Fungsi Terkait

Power Query memiliki 15 fungsi *DateTimeZone*, yang sebagian besar mencerminkan komponen, *LocalNow/FixedLocalNow*, dan kategori *To/From* dari fungsi *#datetime*. Namun, ada juga empat fungsi *DateTimeZone* yang tidak memiliki kecocokan yang sesuai dalam fungsi terkait *DateTime*:

- **DateTimeZone.UtcNow dan DateTimeZone.FixedUtcNow**: Kedua fungsi mengembalikan nilai *DateTimeZone* yang ditetapkan ke tanggal dan waktu saat ini pada sistem. Perbedaannya adalah bahwa yang pertama dapat berubah selama eksekusi ekspresi, sedangkan yang terakhir tetap konstan.
- **DateTimeZone.RemoveZone** dan *DateTimeZone*.*SwitchZone*: Fungsi-fungsi ini memungkinkan manipulasi nilai offset waktu, masing-masing memungkinkan pengguna untuk menghapus komponen zona waktu, dengan demikian mengubah nilai menjadi nilai datetime, atau mengganti nilai offset waktu agar sesuai dengan zona yang berbeda.

Pertimbangan Khusus

Berikut pertimbangan utama untuk bekerja dengan nilai *DateTimeZone*:

- **Waktu musim panas**: Zona waktu yang berubah dalam nilai DateTimeZone tidak secara otomatis menyesuaikan dengan waktu musim panas, dan berbagai negara yang menerapkan waktu musim panas dapat menyesuaikan secara musiman pada tanggal yang berbeda. Kita akan

membahas pendekatan komprehensif untuk mengatasi masalah ini di Bab 10.

- **Konversi**: Saat mengonversi nilai *DateTimeZone*, penting untuk mempertimbangkan bagaimana M menangani konversi zona waktu tersebut. Misalnya, saat nilai *DateTimeZone* dikonversi ke nilai *DateTime*, zona waktu yang berubah akan dibuang, yang secara efektif menampilkan waktu sebagai UTC.

Nilai Durasi

Nilai durasi dalam M digunakan untuk mewakili lamanya waktu yang dinyatakan dalam hari, jam, menit, dan detik (pecahan). Dasar-dasar nilai Durasi dibahas di sini dengan informasi tambahan yang tersedia di Bab 10, Berurusan dengan Tanggal, Waktu, dan Durasi.

Struktur

Berikut ini adalah struktur yang dapat digunakan untuk membuat *#duration*:

```
#duration(
  days as number,
  hours as number,
  minutes as number,
  seconds as number,
) as duration
```

Beberapa contohnya adalah sebagai berikut:

```
#duration(0,1,2,3)      // returns 0.01:02:03

#duration(9,8,7,6.543210) // returns 9.08:07:06.5432100
#duration(3,32,74,82)      // returns 4.09:15:22
```

Fungsi Terkait

Ada 12 fungsi yang termasuk dalam kategori durasi, yang tersebar merata dan cukup dapat diprediksi di tiga kategori:

- **Ekstraksi komponen**: Fungsi-fungsi ini memungkinkan Anda mengekstrak nilai hari, jam, menit, atau detik dari nilai durasi.
- **Perhitungan**: Fungsi-fungsi ini menghitung hari, jam, menit, atau detik dalam nilai durasi tertentu.
- **Pembuatan dan konversi**: Fungsi-fungsi ini menyediakan cara langsung untuk mengonversi nilai durasi ke atau dari jenis nilai lain.

Pertimbangan Khusus

Beberapa pertimbangan penting untuk nilai Durasi adalah:

- **Nilai negatif**: Durasi dapat memiliki nilai negatif, baik secara langsung saat dibuat atau saat nilai durasi yang lebih besar dikurangi dari nilai yang lebih kecil.
- **Perhitungan aritmatika**: Karena durasi mewakili lamanya waktu, bukan titik waktu tertentu, durasi memiliki rentang operasi aritmatika yang lebih luas yang dapat diterapkan padanya. Topik ini dibahas secara terperinci di bagian operator berikut.

3. Nilai-nilai Logis

Nilai logika sangat penting untuk melakukan operasi logika pada data Anda, dan juga untuk mengendalikan aliran data melalui ekspresi melalui struktur kontrol, seperti yang akan kita bahas panjang lebar nanti dalam bab ini.

Struktur

Nilai logika sangat sederhana, hanya mengacu pada nilai benar atau salah.

Fungsi Terkait

Nilai logika memiliki tiga fungsi terkait, yang semuanya berkaitan dengan konversi:

- *Logical.From*: Ini mengonversi nilai apa pun yang dapat dievaluasi sebagai *true* atau *false* ke nilai logika yang sesuai
- *Logical.FromText*: Ini membuat nilai logika dari nilai teks *true* dan *false*
- *Logical.ToText*: Mengembalikan teks *true* atau *false* jika diberikan nilai logika

Pertimbangan Khusus

Berikut ini adalah pertimbangan khusus yang perlu diingat:

- **Konversi**: Nilai logika memiliki beberapa pola konversi unik yang penting untuk diketahui agar dapat memanfaatkan jenis nilai ini secara optimal:
  - false sama dengan 0, *true* sama dengan 1. Hal ini dapat dibuktikan dengan:

```
Number.From( true )  // returns 1
Number.From( false ) // returns 0
```

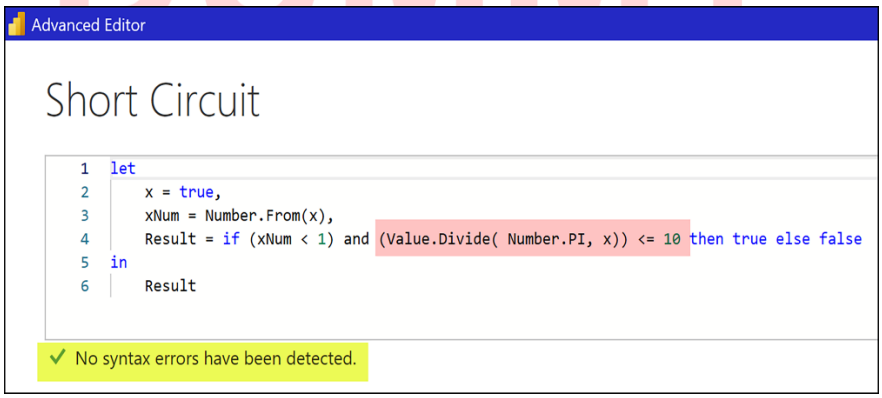
  - Dalam mengonversi angka ke nilai logika, 0 akan berubah menjadi *false*, dan angka lainnya akan berubah menjadi benar – hal ini dapat dibuktikan dengan:

```
Logical.From( 0 )      // returns false
Logical.From( -2 )     // returns true
Logical.From( "csv")   // returns expression.Error:
                       // We couldn't convert to Logical
```

- **Evaluasi short-circuit**: Seperti banyak bahasa lainnya, M menggunakan metode evaluasi ekspresi logika ini, di mana argumen kedua hanya dieksekusi atau dievaluasi jika argumen pertama tidak cukup untuk menentukan nilai ekspresi. Ini berarti bahwa dalam ekspresi logika yang

menggunakan and, jika kondisi pertama *false*, kondisi kedua tidak akan dievaluasi. Untuk ekspresi or, jika kondisi pertama *true*, struktur akan keluar sebelum mengevaluasi kondisi kedua.

Hal ini dapat menyebabkan masalah di mana kondisi kedua ditetapkan dengan cara yang akan memicu kesalahan jika tercapai, tetapi dapat tetap tidak terdeteksi hingga titik tersebut. Mari kita lihat contoh spesifik (perhatikan bahwa *Number.PI* hanyalah konstanta yang mengembalikan angka pi ke 16 tempat desimal):



Gambar 4. 4 Potensi jebakan terkait evaluasi hubung singkat

Dalam ekspresi ini, kita tahu dari pembahasan kita tentang konversi ekspresi logika di atas bahwa *Number.From(true)* akan bernilai 1, dan dengan demikian kondisi pertama dari ekspresi Result akan bernilai *false*. Mengingat bahwa Result mengharuskan kedua kondisi dalam ekspresi bernilai *true* agar bernilai *true*, struktur kontrol keluar segera setelah mengevaluasi kondisi pertama dan mengembalikan nilai *false* untuk seluruh ekspresi.

Namun, kita dapat melihat bahwa ketika x berjenis nilai logika, ekspresi kedua ini akan selalu menghasilkan error, karena ia mencoba membagi angka (pi) dengan nilai logika (*x*), bukan dengan padanan numerik x (didefinisikan dalam ekspresi ini sebagai *xNum*). Namun, M akan memungkinkan kita untuk mencoba melakukan ini karena ekspresi mengambil sintaksis yang benar, dan jika nilai yang kita evaluasi untuk x

sedemikian rupa sehingga *xNum* akan menjadi *>= 1* untuk semua nilai, maka kita tidak akan memicu error dan itu bisa tidak terdeteksi.

Kita akan mengeksplorasi teknik-teknik di Bab 5 tentang tipe data yang akan membantu mencegah jenis masalah ini.

4. Nilai Null

Nilai nol digunakan untuk menyatakan tidak adanya nilai, atau nilai yang keadaannya tidak pasti atau tidak diketahui (misalnya dalam kasus nilai yang hilang).

Struktur

Nilai null ditulis menggunakan null literal. Penting untuk dicatat bahwa null berbeda dari 0. Nilai null adalah tidak adanya nilai, sedangkan 0 adalah nilai numerik tertentu, seperti 1 atau 732. Contoh penggunaan ini adalah:

```
if [Value] = null then "NA" else [Value]
```

Fungsi Terkait

Tidak ada fungsi yang terkait langsung dengan nilai null, tetapi fungsi terkait untuk banyak nilai lainnya secara eksplisit memperhitungkan nilai null.

Pertimbangan Khusus

Pertimbangan utama meliputi:

- **Perbandingan kesetaraan**: *null* memiliki beberapa properti unik terkait cara mengevaluasi ekspresi yang melibatkan operator kualitas. Misalnya, *null* hanya sama persis dengan dirinya sendiri:

```
null = null  // returns true
null = true  // returns false
null = false // returns false
null <= null // returns null
null >= null // returns null
null <> null // returns null
null < 0     // returns null
```

- **Penggabungan dengan null**: Penggabungan nilai dengan null akan menghasilkan null. Misalnya:

```
null & " orange "    //   returns null
```

- **Penyebaran null**: Istilah dalam M ini merujuk pada fakta bahwa nilai null ditambah nilai yang kompatibel dengan operator + akan menghasilkan nilai null. Misalnya:

```
null + number type  // returns null
null + datetime    // returns null
null + duration type // returns null
```

- ***Nullability***: Semua tipe data dalam M dapat dibuat nullable; dengan kata lain, semua tipe data tersebut kompatibel dengan nilai null. Kita akan membahas konsep ini lebih mendalam di Bab 5 tentang tipe data.
- ***Coalesce***: Bahasa M menyertakan operator coalesce khusus untuk mempermudah penanganan *null*, yang dilambangkan dengan tanda tanya ganda (??). Coalesce memungkinkan Anda mengembalikan nilai yang berbeda, bukan null. Jika nilai di sisi kiri operator adalah null, maka nilai di sisi kanan operator akan dikembalikan.

  Sebagai contoh, mari kita lihat pernyataan if berikut dalam M:

```
if [FirstLetter] <> null then [FirstLetter] else
if [SecondLetter] <> null then [SecondLetter] else

   "ZZ"
```

  Operator Coalesce memungkinkan kita untuk menulis ekspresi ini dengan lebih sederhana sebagai:

```
[FirstLetter] ?? [SecondLetter] ?? "ZZ"
```

Bab 12, Penanganan Kesalahan dan Debugging, membahas lebih dalam tentang cara menangani nilai *null* dengan tepat.

5. Nilai Angka

Nilai angka dalam M digunakan untuk menyatakan jumlah numerik dan untuk perhitungan matematika. Meskipun konsep "angka" sudah tidak asing

lagi bagi semua orang, nilai angka dalam M memiliki bentuk literal yang paling beragam dari semua jenis nilai.

Struktur dan Contoh

Semua ekspresi berikut mewakili tipe nilai angka yang valid dalam M:

Expression	Description	Result
2.998	Fractional number	*2.998*
-3.2	Fractional number	*-3.2*
1.00e+3	Fractional number with exponent	*1000*
1.0e-3	Fractional number wiht exponent	*0.001*
36	Whole number	*36*
2e4	Whole number with exponent	*20000*
0x62	Whole number in hex	*98*

Gambar 4. 5 Nilai angka dalam M

Fungsi Terkait

Karena berbagai macam operasi matematika yang dapat dilakukan pada nilai angka, terdapat 52 fungsi angka terkait yang berbeda di M, yang tersebar di enam kategori berikut:

- **Bitwise**: Fungsi-fungsi ini secara langsung memanipulasi bit dari sebuah angka. Akan tetapi, karena M adalah bahasa fungsional dan biasanya tidak melibatkan transformasi biner tingkat rendah, operasi bitwise tidak terdokumentasi dengan baik atau tidak umum digunakan di M.
- **Creation and conversion**: Fungsi-fungsi ini digunakan untuk membuat nilai angka atau mengonversinya ke atau dari jenis nilai yang kompatibel.
- **Informational**: Fungsi-fungsi ini menentukan apakah nilai angka genap, ganjil, atau **Not a Number (NaN)**.
- **Mathematical**: Fungsi-fungsi ini melakukan kalkulasi matematika pada nilai angka. Hampir setengah dari fungsi matematika berhubungan dengan kalkulasi trigonometri.

- **Random**: Fungsi-fungsi ini digunakan untuk membuat angka acak dalam rentang yang ditentukan.
- **Rounding**: Fungsi-fungsi ini mengontrol jumlah tempat desimal yang akan digunakan untuk mengekspresikan nilai angka.

Pertimbangan Khusus

Berikut ini adalah beberapa pertimbangan penting yang relevan dengan nilai Angka:

- **Tipe data versus format**: Salah satu konsep terpenting yang perlu dipahami terkait angka dalam M adalah bahwa meskipun tipe data angka ditetapkan dalam Power Query/M, formatnya ditetapkan dalam lapisan visual Power BI (atau Excel), dan bukan dalam Power Query. Implikasi utama dari perbedaan ini adalah bahwa tipe tersebut mengontrol cara angka disimpan, yang pada gilirannya dapat memengaruhi tingkat presisi dan kebutuhan memori angka. Di sisi lain, format hanya mengontrol bagaimana angka tersebut muncul saat digunakan dalam visual. Masalah ini akan dibahas secara terperinci dalam Gbapter 5 tentang tipe data.
- **Sisi (Facets)**: Anda dapat menentukan informasi yang lebih terperinci tentang tipe angka di M dengan menggunakan serangkaian subtipe yang disebut klaim tipe. Subtipe ini termasuk dalam kategori sisi. Sisi memengaruhi berbagai opsi penyimpanan, seperti mata uang, desimal, dan bilangan bulat, sehingga memengaruhi ketepatan suatu nilai. Nilai yang berada di luar rentang yang ditentukan akan menghasilkan kesalahan. Penggunaan dan implikasi sisi angka yang tepat akan dibahas di Gbapter 5 tentang tipe data.
- **Pembagian dengan nol**: Di M, pembagian dengan nol akan menghasilkan hasil yang berbeda tergantung pada pembilangnya:
    - Membagi angka positif dengan 0 akan menghasilkan positif tak terhingga (∞)

- Membagi angka negatif dengan tak terhingga akan menghasilkan negatif tak terhingga (-∞)
- Membagi 0 dengan 0 menghasilkan NaN

Dalam banyak perhitungan di M, pembagian dengan nol akan menyebabkan kesalahan waktu proses, jadi langkah-langkah harus diambil untuk mencegah skenario ini. Penanganan kesalahan akan dibahas secara terperinci di Bab 12.

- **Konstanta (Constats)**: Ada enam konstanta numerik yang didefinisikan dalam bahasa M:
  - *Number.E*: Mengembalikan e, atau bilangan Euler, dengan akurasi 16 tempat desimal.
  - *Number.Epsilon*: Mengembalikan bilangan positif terkecil yang mungkin untuk nilai floating point.
  - *Number.NaN*: Mengembalikan nilai konstanta yang mewakili 0 dibagi 0.
  - *Number.NegativeInfinity*: Mengembalikan nilai konstanta yang mewakili 1 dibagi 0.
  - *Number.PI*: Mengembalikan bilangan pi dengan akurasi 16 tempat desimal.
  - *Number.PositiveInfinity*: Mengembalikan nilai konstanta yang mewakili 1 dibagi 0.

6. Nilai Teks

Dalam M, nilai teks adalah rangkaian karakter yang digunakan untuk merepresentasikan data teks. Informasi lebih lanjut tentang nilai teks dapat ditemukan di Bab 5, Memahami Tipe Data.

Struktur

Nilai teks dalam M ditunjukkan dengan " ". Berikut ini adalah beberapa contohnya:

```
" Basketball "
"23 "

"November 9, 1993"
```

Fungsi Terkait

Karena semua jenis nilai non-abstrak dan non-terstruktur kecuali null dapat dikonversi ke dan dari nilai teks, tidak mengherankan bahwa ada 45 fungsi terkait teks, yang tersebar di kategori berikut:

- **Conversion and creation**: Fungsi yang membuat karakter dan nilai teks dan mengubahnya menjadi jenis nilai lain
- **Extraction and position**: Fungsi yang menentukan posisi karakter dalam nilai teks dan mengekstrak berbagai subset dari dalam nilai teks
- **Formatting**: Fungsi yang membersihkan karakter yang tidak diinginkan dan mengontrol huruf besar-kecil pada teks dan karakteristik pemformatan lainnya
- **Information**: Fungsi yang menentukan panjang dan isi nilai teks
- **Transformation**: Fungsi yang menggabungkan, membagi, menyisipkan, atau menghapus teks atau subset nilai

Pertimbangan Khusus

Berikut ini adalah fitur-fitur utamanya:

- **Penggabungan (Concatenation)**: Anda dapat menggabungkan nilai teks dalam M menggunakan operator ampersand (&). Misalnya:

```
Sentence = "M code is not that hard " & "if you know the basic
principles."
```

Kode M ini mengembalikan:

```
"M code is not that hard if you know the basic principles."
```

- **Karakter escape**: Karakter dan kode tertentu yang disediakan untuk tujuan khusus dalam bahasa kode M, seperti line feed (*lf*) dan carriage

return (*cr)*, memerlukan urutan "karakter escape" sebelum karakter tersebut untuk memerintahkan program agar membacanya sebagai karakter teks literal. Misalnya, karakter tersebut dapat ditulis sebagai *#(lf)* dan *#(cr)*.

- **Evaluasi operator perbandingan**: Nilai teks mendukung semua operator perbandingan umum, yang secara default peka terhadap huruf besar/kecil. Namun, huruf besar/kecil tidak dievaluasi seperti yang mungkin dipikirkan orang – huruf kecil menerima nilai yang lebih tinggi daripada padanannya dalam huruf besar:

```
"gza" < "rza"      // returns true
"rza" = "Rza"      // returns false
"rza" > "Rza"      // returns true
```

Jika ragu tentang urutan evaluasi nilai teks, Anda dapat menggunakan fungsi *Character.ToNumber* untuk mengembalikan nilai numerik (yang menjadi dasar evaluasi) dari karakter teks apa pun. Untuk membandingkan nilai teks tanpa mempertimbangkan huruf besar/kecil, kita dapat menggunakan opsi *Comparer.OrdinalIgnoreCase*. Misalnya, contoh berikut mengembalikan *true*:

```
Comparer.Equals(Comparer.OrdinalIgnoreCase, "rza", "RZa ")
```

Meskipun nilai teks mungkin tampak sederhana pada pandangan pertama, nilai tersebut sering kali menghadirkan tantangan pembersihan data yang rumit. Masalah ini dibahas dalam Bab 14, Troublesome Data Patterns, dan Bab 11, Gomparers, Replacers, Gombiners, dan Splitters. Selain itu, untuk pemegang lisensi Power BI Premium dan Premium per pengguna, Power Query menyertakan analitik teks AI, yang menyediakan kemampuan tingkat lanjut termasuk pemrosesan teks dan analisis sentimen.

7. List Nilai (*List Values*)

List adalah serangkaian nilai. Dalam M, satu kolom nilai adalah list, dan list dapat berisi semua jenis nilai primitif atau terstruktur. List dibahas secara mendalam di Bab 6, Nilai Terstruktur.

Struktur

List diawali dengan tanda kurung kurawal, { }. Misalnya:

```
{ 1, 2, 3 }
{ "Cat", "Dog", "Monkey" }
{ { 1, 2, 3 }, { "Cat", "Dog", "Monkey" } }
```


Fungsi Terkait

List adalah salah satu jenis nilai yang paling fleksibel dan kuat dalam M, yang menjelaskan mengapa ada berbagai macam fungsi yang terkait dengannya. Ada total 71 fungsi terkait, yang dikategorikan ke dalam lima kelompok berbeda:

- **Pembuatan (Generation)**: Fungsi yang membuat list dengan karakteristik tertentu yang berbeda, misalnya, list tanggal, angka, dan angka acak, serta fungsi yang lebih kompleks yang memungkinkan Anda membuat list secara rekursif
- **Informational**: Terutama fungsi yang menghasilkan nilai logika tergantung pada konten list yang diberikan
- **Selection**: Fungsi yang memilih item atau kelompok item dari list berdasarkan kriteria yang ditentukan
- **Statical**: Fungsi yang mengembalikan hasil perhitungan pada list yang diberikan (misalnya, *sum, median, max, etc*, dll.)
- **Transformational**: Fungsi yang mengambil list sebagai input dan membuat list baru dengan memodifikasi struktur dan/atau item dalam list asli

Pertimbangan Khusus

Karakteristik yang sangat penting untuk diperhatikan mengenai list dalam M adalah bahwa list tersebut bekerja dengan indeks berbasis nol. Ini berarti bahwa indeks yang menentukan posisi item dalam list dimulai dengan Perhatikan bahwa ini berbeda dari **Data Analysis Expressions (DAX)**, yang menggunakan indeks berbasis satu, di mana item pertama dalam list ditemukan pada posisi 1.

Misalnya, untuk menemukan huruf ketiga dalam kata *Method*, dalam M, Anda dapat menggunakan yang berikut:

```
Text.At("Method", 2)
```

Ini mengembalikan t, karakter pada posisi ketiga (0, 1, 2) dari string teks. Sebaliknya, untuk memperoleh hasil yang sama dalam DAX, Anda dapat menggunakan:

```
MID("Method", 3, 1)
```

Ini mengembalikan 1 karakter, dimulai pada posisi ketiga (1, 2, 3). List akan dibahas secara rinci dalam bab-bab berikutnya, dimulai dengan Bab 6 tentang nilai terstruktur.

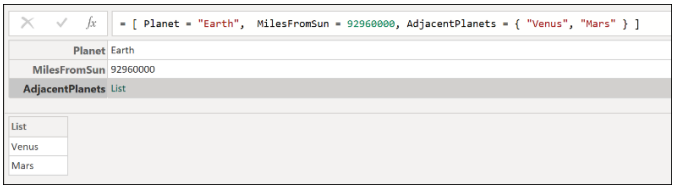
8. Nilai Record

Catatan adalah list nilai yang diberi nama. Anda juga dapat menganggap catatan sebagai satu baris dalam tabel, yang setiap kolomnya memiliki nama kolom dan nilai. Seperti list, catatan dapat berisi nilai primitif atau terstruktur. Informasi tambahan tentang nilai catatan dapat ditemukan di Bab 6, Nilai Terstruktur.

Struktur

Catatan diawali dengan tanda kurung siku, [ ]. Berikut adalah contoh dan catatan terkait yang dibuat:

```
[Planet = "Earth",  MilesFromSun = 92960000, AdjacentPlanets = { "Venus",
"Mars" }]
```



Gambar 4. 6 Pembuatan record di M, juga menampilkan nilai dalam list bersarang

Fungsi Terkait

Ke-23 fungsi yang terkait dengan catatan tersebar di lima kategori yang sebagian besar sejajar dengan kategorisasi fungsi list. Namun, tidak ada fungsi catatan statistik dan Anda akan menemukan bahwa kategori baru ditambahkan:

- **Konversion**: Fungsi yang membuat catatan dari list atau tabel, atau tabel atau list dari catatan
- **Geospatial**: Fungsi yang menangani data geospasial dalam bentuk **well-known text (WKT)**
- **Informational**: Fungsi yang menghasilkan nilai tergantung pada konten list yang diberikan
- **Pemilihan (Selection)**: Fungsi yang memilih bidang atau nilai bidang tertentu dari catatan berdasarkan kriteria tertentu
- **Transformational**: Fungsi yang mengubah struktur atau konten catatan

Pertimbangan Khusus

Mirip dengan bagaimana {} memungkinkan akses ke item dalam list menggunakan indeks posisi, [] dengan nama bidang di dalamnya memberikan akses ke bidang tertentu dalam record. Record dibahas secara lebih rinci di Bab 6, Nilai Terstruktur.

9. Nilai Tabel

Tabel terdiri dari tiga elemen kunci:

- **Rows**, di mana setiap baris mewakili satu record individual
- **Column**, di mana setiap kolom mendefinisikan bidang dalam record dan diberi tipe data tertentu

- **Headers**, yang mengidentifikasi nama setiap kolom

Dalam M, tabel dapat dibuat menggunakan berbagai metode, termasuk menggunakan konstruktor #table, atau dengan mengubah nilai, list, record, kolom, atau baris dengan fungsi M yang sesuai.

Informasi tambahan tentang nilai tabel dapat ditemukan di Bab 6, Nilai Terstruktur, serta berbagai bab lainnya.

Struktur

Berikut ini adalah struktur yang dapat digunakan untuk membuat tabel:

```
#table(

  columns as any,
  rows as any,

) as any
```

Mari kita lihat contohnya:

```
let
  Source = #table(
    type table [ProductID = text, ProductQuantity = number,
                Column3 = date],
      {
        {"P001", 10, #date(2023, 8, 13)},
        {"P002", 25, #date(2023, 8, 14)},
        {"P003", 30, #date(2023, 8, 15)}
         }
    )
in
  Source

#table({}, {}) // empty table
```

Ada banyak cara untuk membuat tabel menggunakan *#table*, serta fungsi lain yang membuat tabel dari jenis nilai primitif dan terstruktur lainnya. Metode ini akan dibahas secara rinci di Bab 6, Nilai Terstruktur.

Fungsi Terkait

Mengingat tabel merupakan jenis nilai terstruktur yang paling sering digunakan, tidak mengherankan jika tabel memiliki jumlah fungsi terkait tertinggi (116) di antara semua jenis nilai. Fungsi-fungsi ini tersebar di lima kategori utama yang menentukan kegunaannya:

- **Creation and conversion**: Fungsi-fungsi ini membuat tabel dan mengubahnya dari dan ke jenis nilai lainnya.
- **Informational**: Tabel dicirikan oleh banyak atribut yang berbeda, seperti dimensi, skema, hubungan, dan konten. Fungsi-fungsi ini dapat mengekstrak informasi tentang atribut spesifik apa pun dari tabel tertentu.
- **Column specific**: Fungsi-fungsi ini memungkinkan manipulasi kolom dan tajuk tabel.
- **Row specific**: Fungsi-fungsi ini memungkinkan manipulasi record individual atau grup dalam tabel.
- **Other functions**: Fungsi-fungsi ini mengontrol buffering, pelipatan kueri, dan pengendali yang ditentukan pengguna untuk operasi kueri. Kecuali **Table.Buffer**, fungsi-fungsi ini jarang digunakan.

Pertimbangan Khusus

Berikut ini adalah beberapa fitur utama yang perlu diperhatikan saat bekerja dengan tabel:

- **Pemilihan dan proyeksi**: Aspek penting dalam bekerja dengan tabel adalah kemampuan untuk memfilter tabel hingga hanya baris (pemilihan) dan kolom (proyeksi) yang diperlukan dalam transformasi tertentu. Cara Anda melakukan pemilihan dan proyeksi memiliki implikasi untuk kinerja dan keterbacaan kode M Anda. Terutama saat Anda mulai menggunakan struktur bersarang di M, pemilihan dan proyeksi dapat menjadi sangat rumit. Masalah ini dibahas secara mendalam di Bab 8, Bekerja dengan Struktur Bersarang.
- **Kinerja**: Karena sebagian besar langkah dalam kueri mengembalikan nilai tabel, sebagian besar fokus pada pengoptimalan kinerja dalam Power

Query berpusat pada pengoptimalan pembuatan dan transformasi tabel. Pengoptimalan kinerja dibahas di Bab 15, Mengoptimalkan Kinerja.

Tabel akan dibahas secara sangat rinci di bab-bab berikutnya, dimulai dengan Bab 6 tentang nilai terstruktur.

10. Nilai Fungsi

Nilai fungsi adalah nilai yang jika dipanggil dengan serangkaian nilai input (misalnya argumen) akan menghasilkan nilai baru.

Pertimbangan Khusus

Dalam M, fungsi ditentukan dengan mencantumkan parameter input fungsi dalam tanda kurung, lalu operator "goes to" (=>), lalu ekspresi yang mendefinisikan fungsi tersebut. Berikut ini adalah contohnya:

```
Concatenator =
  (parameter1, parameter2) => (parameter1 & parameter2)
```

```
DifferentDistinct =
  (x) => List.Difference(x,List.Distinct(x))
```

Fungsi Terkait

Nilai fungsi memiliki lima fungsi terkait – dua untuk membuat fungsi, dua untuk memanggil fungsi kustom, dan satu fungsi informasi untuk memeriksa apakah fungsi tertentu bertindak sebagai sumber data.

Pertimbangan Khusus

Berikut ini adalah beberapa pertimbangan utama:

- **Fleksibilitas dan kekuatan**: Fungsi dapat ditetapkan ke variabel, diteruskan sebagai argumen, dan dikembalikan dari fungsi lain, yang menjadikannya beberapa jenis nilai yang paling serbaguna dan kuat di M.
- **Pentingnya tipe data**: Untuk menghindari kesalahan di M, penting untuk menetapkan tipe data yang tepat ke parameter fungsi. Kita akan membahas

masalah penerapan tipe data yang tepat di bab berikutnya, serta di Bab 9, yang secara khusus didedikasikan untuk fungsi kustom.

- **Kinerja**: Saat memanggil fungsi kompleks, khususnya fungsi rekursif, dan fungsi tingkat tinggi yang mengambil satu atau beberapa fungsi sebagai argumen dan/atau mengembalikan fungsi, perhatikan potensi dampak buruk pada kinerja yang dapat ditimbulkan oleh fungsi tersebut.

Fungsi yang ditentukan pengguna adalah salah satu elemen M yang paling kuat dan dibahas secara terperinci di Bab 9.

11. Tipe Nilai

Nilai tipe dapat dianggap sebagai nilai meta yang menyediakan informasi tentang tipe data nilai lain.

Struktur

Karena mereka mengklasifikasikan tipe nilai, mereka dapat mengambil deskriptor apa pun dari nilai-nilai tersebut (misalnya, *Date, binary, DateTime, DateTimeZone, Duration, logical, null, number, text, time, list, record, table, function,* dan *type*).

Dalam M, fungsi *Value.Type* mengembalikan nilai yang mencerminkan tipe data dari nilai yang dimasukkan ke dalam fungsi:

```
Value.Type( 400 ) = number
Value.Type( "Book" ) = text

Value.Type( { 2, 3, 5, 7, 11, 13 } ) = list
```

Fungsi Terkait

Sebanyak 22 fungsi dikaitkan dengan nilai tipe. Karena nilai tipe menyampaikan informasi tentang nilai lain alih-alih menjadi struktur tersendiri, fungsi-fungsi ini terutama memberikan detail lebih lanjut tentang tipe suatu nilai. Misalnya, *Type.TableRow* dan *Type.TableColumn*

mengembalikan nilai tipe untuk baris atau kolom tertentu, masing-masing, dalam suatu tabel.

Pertimbangan Khusus

Aspek-aspek utama meliputi yang berikut:

- **Integritas data**: Nilai tipe penting untuk memastikan integritas input dan output data fungsi. Selain itu, nilai tersebut digunakan untuk menetapkan tipe data ke data tak terstruktur dan untuk mengonversi tipe data dari satu ke yang lain saat transformasi dilakukan.
- **Nilai tipe abstrak**: Ada sejumlah tipe abstrak yang tidak mengklasifikasikan nilai apa pun secara unik. Ini termasuk tipe data *any*, *anynonnull*, dan *none*, yang masing-masing mencakup semua nilai, semua nilai bukan null, dan tidak ada nilai. Dua nilai pertama digunakan untuk mendukung nilai input dari beberapa tipe, sedangkan tipe none tidak mengklasifikasikan nilai apa pun dan umumnya tidak digunakan.
- **Nilai tipe kustom**: Semua tipe data di luar gabungan tipe data primitif dan abstrak dianggap sebagai tipe data kustom, yang relevan terutama untuk mendeskripsikan nilai terstruktur dan untuk pembuatan konektor Power Query kustom. Tipe data kustom dan nilai tipe kustom terkaitnya akan dibahas lebih lanjut di Bab 5.

Kita sekarang telah melihat berbagai tipe nilai yang tersedia dalam M. Selanjutnya, kita akan menjelajahi operator – "ikatan" yang mengikat nilai bersama-sama dalam ekspresi.

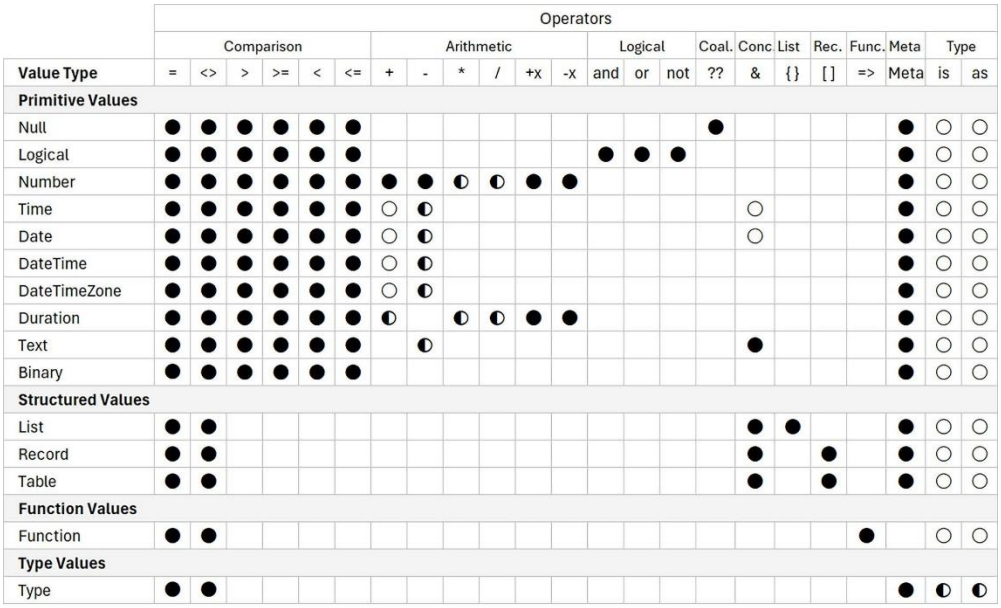
C. Operator

Sama seperti ada berbagai jenis ikatan yang menghubungkan atom, ada juga berbagai kategori operator dalam M yang menghubungkan nilai. Tabel berikut memaparkan kategori yang berbeda dan operator spesifik yang termasuk dalam setiap kategori:

Tabel 4. 2 Kategori operator di M

Kategori Operator	Termasuk Operator	Deskripsi
Aritmatika	Addition (+) Subtraction (-) Multiplication (*) Division (/)	Melakukan operasi matematika
Comparison (juga dikenal sebagai operator umum, karena digunakan oleh semua operator nilai primitif)	Equal to (=) Not equal to (<>) Less than (<) Less than or equal to (<=) Greater than (>) Greater than or equal to (>=)	Membandingkan dua nilai (mengembalikan hasil Boolean)
Logical	and or not	Melakukan operasi logika (Boolean)
Text	Concatenation (&)	Joins text strings
List	Equal to (=) Not equal to (<>) Concatenation (&) List indexer ({})	Digunakan untuk membuat, membandingkan, menggabungkan, dan mengakses list
Record	Equal to (=) Not equal to (<>) Concatenation (&) Record lookup ([])	Digunakan untuk membuat, membandingkan, menggabungkan, dan mengakses catatan.
Table	Equal to (=) Not equal to (<>) Concatenation (&)	Digunakan untuk membandingkan dan menggabungkan tabel
Function	Goes to (=>)	Digunakan untuk memetakan input ke output dalam definisi fungsi
Kompatibilitas dan pernyataan tipe	is as	Digunakan untuk memeriksa tipe data dan menegaskan tipe suatu nilai

Sama seperti jenis nilai menentukan fungsi apa yang akan dan tidak akan bekerja dengan nilai tertentu, jenis nilai juga menentukan serangkaian operator yang kompatibel. Gambar berikut memetakan jenis nilai yang dibahas sebelumnya dalam bab ini, dengan kategori operator dan operator spesifik yang berlaku untuknya:



	Operators																					
	Comparison						Arithmetic						Logical			Coal.	Conc	List	Rec.	Func.	Meta	Type	
Value Type	=	<>	>	>=	<	<=	+	-	*	/	+x	-x	and	or	not	??	&	{}	[]	=>	Meta	is	as
Primitive Values																							
Null	●	●	●	●	●	●										●					●	○	○
Logical	●	●	●	●	●	●							●	●	●						●	○	○
Number	●	●	●	●	●	●	●	●	◐	◐	●	●									●	○	○
Time	●	●	●	●	●	●	○	◐									○				●	○	○
Date	●	●	●	●	●	●	○	◐									○				●	○	○
DateTime	●	●	●	●	●	●	○	◐													●	○	○
DateTimeZone	●	●	●	●	●	●	○	◐													●	○	○
Duration	●	●	●	●	●	●	◐		◐	◐	●	●									●	○	○
Text	●	●	●	●	●	●		◐									●				●	○	○
Binary	●	●	●	●	●	●															●	○	○
Structured Values																							
List	●	●															●	●			●	○	○
Record	●	●															●		●		●	○	○
Table	●	●															●		●		●	○	○
Function Values																							
Function	●	●																		●		○	○
Type Values																							
Type	●	●																			●	◐	◐

● **Self-Only** Operates exclusively with its own value type.
◐ **Self+Other** Operates with its own and some other value types.
○ **Other-Only** Operates exclusively with other value types.

Gambar 4. 7 Pemetaan jenis nilai ke kategori operator dan operator

Menerapkan operator aritmatika dan penggabungan pada nilai *Date*, *DateTime*, *DateTimeZone*, *Duration*, dan *Time* menyajikan sejumlah kasus khusus, seperti yang dirangkum di sini:

Operand LHS => RHS	Date	DateTime	DateTimeZone	Duration	Time	Number
Date	All comparison - = duration			+, - = Date	& = DateTime	
DateTime		All comparison - = duration		+, - = DateTime		
DateTimeZone			All comparison - = duration	+, - = DateTimeZone		
Duration	+, - = Date	+, - = DateTime	+, - = DateTimeZone	All comparison +, - = duration	- = Time	*, / = Duration
Time	& = DateTime			+, - = Time	All comparison - = duration	
Number				* = duration		All comparison +, -, *, / = Number

Gambar 4. 8 List operasi aritmatika dan penggabungan yang valid dan tipe nilai yang dihasilkan untuk nilai Tanggal, WaktuTanggal, ZonaWaktuTanggal, Durasi, dan Waktu

Secara umum, aturan berikut berlaku untuk operasi aritmatika yang melibatkan nilai *Date*, *Time*, *DateTime*, dan *DateTimeZone*:

- Saat Anda mengurangi dua nilai dengan jenis yang sama, hasilnya adalah *Duration* positif atau negatif

- Saat Anda menambahkan atau mengurangi *Duration* dari salah satu jenis nilai ini, hasilnya adalah jenis nilai yang sama
- Menambahkan atau mengurangi dua *Durations* akan menghasilkan *Duration*
- Mengalikan atau membagi *Duration* dengan nilai angka akan menghasilkan *Duration*
- Menggabungkan nilai *Date* dan nilai *Time* akan menghasilkan nilai DateTime

Satu kasus khusus terakhir yang perlu diperhatikan berkaitan dengan operasi aritmatika. Operasi numerik dalam M mengikuti urutan operasi matematika standar yang disingkat sebagai PEMDAS (*parentheses, exponents, multi- plication, division, addition,* dan *subtraction*). Namun, tidak seperti banyak bahasa yang menggunakan tanda sisipan (^) sebagai operator untuk eksponensial, M mengharuskan penggunaan fungsi *Number.Power* untuk menaikkan nilai angka ke pangkat. List lengkap prioritas operator dapat ditemukan di sini: https://learn.microsoft.com/en-us/powerquery-m/m-spec-operators.

Ada beberapa operator tambahan yang kurang umum dalam bahasa M, termasuk:

- **Coalesce (??)**: Mengembalikan hasil operan kiri jika bukan null; jika tidak, mengembalikan hasil operan kanan. Contoh: x ?? y. Operator ini dibahas di bagian Nilai null dalam bab ini.
- **Metadata (meta)**: Menambahkan metadata ke suatu nilai. Contoh: x meta y. Operator ini dibahas di Bab 7, Gonceptualizing M, dan Bab 16, Enabling Extensions.
- **Unary Plus (+x)**: Operator identitas untuk nilai angka. Misalnya, + 12 menghasilkan 12.
- **Negation (-x)**: Operator negasi untuk nilai angka. Misalnya, -12 menghasilkan -12.

Di bagian berikutnya, kita akan melihat cara menggabungkan operator dan nilai (yang telah kita bahas sebelumnya dalam bab ini) menjadi ekspresi.

D. Ekspresi

Ekspresi dalam M seperti rumus – dengan menggunakan operator dan nilai sebagai input, ekspresi memungkinkan Anda untuk menghasilkan nilai baru. Ekspresi ini dievaluasi atau dihitung untuk selalu mengembalikan satu nilai sebagai hasilnya. Pada akhirnya, setiap ekspresi dalam M harus menghasilkan nilai atau kesalahan. Melalui ekspresi, M melampaui representasi data belaka dan memungkinkan Anda untuk mengubah data Anda.

Jadi, apa perbedaan antara nilai dan ekspresi? Sementara nilai merepresentasikan titik data dalam bentuk yang paling sederhana, ekspresi adalah rumus yang dapat digunakan untuk memanipulasi atau menghasilkan nilai-nilai ini. Ekspresi ini dapat sesederhana mengembalikan satu konstanta atau serumit pernyataan yang berisi ratusan ekspresi perantara. Misalnya, contoh berikut adalah ekspresi nilai tunggal:

```
1                               // returns a number value
"We love Power Query"           // returns a text value

[ Fruit = "Apple", Fruit = "Plum"] // returns a record value
```

Namun, Power Query juga memungkinkan Anda menggabungkan beberapa nilai atau memungkinkan Anda membuat logika untuk mengembalikan nilai lain. Misalnya, baris kode berikut juga dianggap sebagai ekspresi:

```
1 + 5                   // the sum of two numbers
List.Count({1,2,3})     // an expression using a function
( x, y ) => x - y       // a function that subtracts y from x
let age = 2 in age * 2 // a let expression
```

Dalam contoh ini, ekspresi adalah resep masukan untuk menghasilkan hasil, sedangkan nilai adalah hasil dari resep tersebut.

Contoh sebelumnya relatif mudah. Namun, sangat valid untuk membuat ekspresi yang lebih kompleks dengan menggabungkan beberapa ekspresi. Misalnya:

```
Date.AddMonths( #date( 2024,1,1), 1 )
```

Contoh ini terdiri dari 6 nilai. Anda dapat menemukan yang berikut:

- **Nilai fungsi**: Baik *Date.AddMonths* maupun #date adalah fungsi yang menjalankan suatu operasi.
- **Nilai angka**: Anda dapat menemukan tiga nilai angka yang digunakan dalam *Date.AddMonths* dan satu nilai angka digunakan dalam fungsi #date.

Mengerjakan nilai melalui ekspresi merupakan elemen penting dalam menjalankan transformasi. Namun, hal-hal dapat menjadi rumit dengan cepat setelah Anda merasa lebih nyaman dalam membuat ekspresi. Bayangkan Anda menumpuk 20 nilai yang berbeda bersama-sama. Dalam situasi seperti itu, Anda dapat mempertimbangkan untuk menyimpan logika Anda dalam pernyataan let. Pernyataan let memungkinkan Anda untuk menentukan serangkaian variabel, membuat referensi ke variabel tersebut, dan mengeluarkan hasil. Namun, alih-alih melihat lautan kode, nama variabel yang disertakan membantu dalam membagi kode menjadi potongan-potongan yang mudah dipahami.

Misalnya, jika kita melanjutkan contoh tanggal sebelumnya, tetapi sekarang juga memformatnya dengan cara tertentu, kita dapat menggunakan kode ini:

```
Date.ToText( Date.AddMonths( #date( 2024,1,1), 1 ), [Format = "yyyy-MM-dd"] )
```

Ekspresi ini mengembalikan tanggal 1 Februari 2024. Namun, semakin banyak nilai yang kita sertakan dalam ekspresi kita, semakin sulit dibaca. Untuk memperbaiki situasi ini, kita dapat memformat kode sebagai berikut:

```
Date.ToText(
  Date.AddMonths(
    #date( 2024,1,1 ),
    1 ),
  [Format = "yyyy-MM-dd"] )
```

Kodenya kini lebih mudah dibaca tetapi masih memerlukan pemrosesan mental. Untuk mempermudah, mari sertakan variabel dengan menggunakan pernyataan let:

```
let
  myDate = #date( 2024,1,1 ),
  addMonth = Date.AddMonths( myDate, 1 ),
  formatDate = Date.ToText( addMonth, [Format = "yyyy-M-d"] )
```

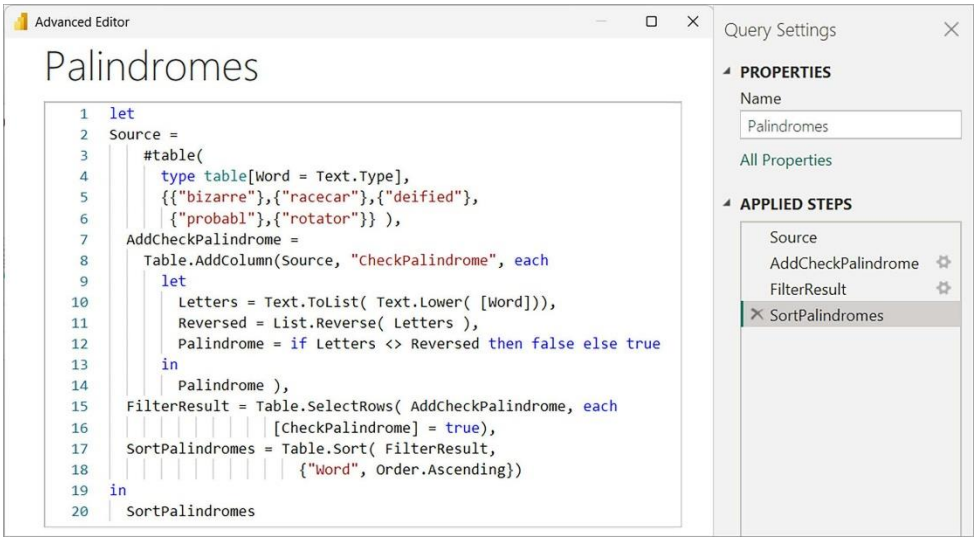
Di sini, kata kunci let diikuti oleh tiga variabel, dan hasilnya ditunjukkan oleh nama variabel *formatDate* yang muncul setelah in. Hasil kode ini identik untuk ketiga cuplikan, tetapi dengan memberikan nama variabel yang jelas, ekspresinya jauh lebih mudah dipahami.

Anda sekarang telah melihat contoh mudah penggunaan pernyataan let. Tempat paling umum tempat Anda akan menemukan pernyataan tersebut adalah di **Editor Lanjutan**. Saat melakukan operasi, Power Query secara otomatis membuat pernyataan let di **Advanced Editor**. Meskipun pernyataan let khusus ini dibuat secara otomatis, Anda dapat secara manual menyertakan pernyataan *let* tambahan dalam setiap langkah yang diterapkan. Mari kita bahas seperti apa bentuknya.

1. Ekspresi *Let* Bersarang

Power Query menggunakan ekspresi let untuk mewakili langkah-langkah yang diterapkan dalam kueri Anda. Untuk membuat logika transformasi Anda untuk suatu langkah lebih mudah dicerna, Anda dapat menumpuk beberapa ekspresi *let*.

Di bagian ini, kita akan menjelajahi kueri Palindrome, yang menggunakan ekspresi let bertumpuk. Kueri ini bertujuan untuk menganalisis kolom kata dan menentukan kata mana yang merupakan palindrom, yaitu kata-kata yang dibaca sama baik dari belakang maupun depan. Kita kemudian akan mengembalikan tabel kata-kata ini, diurutkan dalam urutan abjad menaik. Berikut tampilan kueri di **Advanced Editor**:



Gambar 4. 9 Kueri yang lebih kompleks yang berisi ekspresi, nilai, operator, struktur kontrol, dan enumerasi.

Untuk memudahkan referensi, berikut adalah kueri Palindrome dalam bentuk teks:

```
let
  Source =
    #table(
      type table[Word = Text.Type],
      {{"bizarre"},{"racecar"},{"deified"},
       {"probabl"},{"rotator"}} ),
  AddCheckPalindrome =
    Table.AddColumn(Source, "CheckPalindrome", each
      let
        Letters = Text.ToList( Text.Lower( [Word])),
        Reversed = List.Reverse( Letters ),
        Palindrome = if Letters <> Reversed then false else true
      in
        Palindrome ),
  FilterResult = Table.SelectRows( AddCheckPalindrome, each
                   [CheckPalindrome] = true),
  SortPalindromes = Table.Sort( FilterResult,
                      {"Word", Order.Ascending})
in
  SortPalindromes
```

Kueri tersebut menggunakan pernyataan let dengan empat variabel dan mengembalikan tabel dalam langkah yang disebut *SortPalindromes*. Ini merupakan empat langkah yang diterapkan dalam kueri kita.

Di dalam pernyataan let tersebut terdapat langkah yang disebut *AddCheckPalindrome*. Langkah ini berisi pernyataan let lain yang terdiri dari tiga ekspresi yang menghasilkan nilai untuk setiap kata, yang menunjukkan apakah itu palindrom (*true*) atau bukan palindrom (*false*). Meskipun ini mungkin terdengar membingungkan pada awalnya, jangan khawatir. Anda akan terbiasa bekerja dengan pernyataan *let*. Pernyataan ini hanyalah cara untuk membagi kode Anda menjadi variabel dan mengembalikan hasil.

Mari kita uraikan apa yang dilakukan kueri ini dengan memeriksanya langkah demi langkah:

- *Source*: Ekspresi ini menyediakan data sumber untuk ekspresi tersebut – tabel kolom tunggal yang diidentifikasi sebagai *Source*. Seperti yang dibahas dalam Bab 3, kita biasanya menyerap data dari sumber data terstruktur atau tidak terstruktur eksternal. Namun, untuk tujuan kejelasan, kita menghasilkan data sumber kita dalam ekspresi let.
- *AddCheckPalindrome*: Langkah ini menambahkan kolom baru bernama *CheckPalindrome* ke tabel Sumber menggunakan pernyataan *let* bersarang. Pernyataan ini dijabarkan menjadi tiga ekspresi sementara:
    - Yang pertama, *Letters*, mengubah setiap kata menjadi huruf kecil dan membagi kata tersebut menjadi list huruf yang menyusunnya.
    - Langkah berikutnya membalikkan list *Letters* dalam langkah yang disebut *Reversed*.
    - Ekspresi terakhir, Palindrome, menggunakan logika *if then else* untuk menguji apakah list huruf asli identik dengan list yang sama dalam urutan terbalik. Ini mengisi kolom CheckPalindrome dengan nilai *true*/*false* untuk setiap baris.

- *FilterResult*: Langkah ini menerapkan filter ke kolom *CheckPalindrome* untuk hanya menyimpan baris yang memiliki nilai benar (yaitu, yang dianggap sebagai palindrom).
- *SortResult*: Langkah terakhir mengurutkan palindrom yang tersisa di kolom *Words* dalam urutan menaik. *Order.Ascending* dalam langkah ini juga menandai kemunculan pertama enumerasi – kita akan membahas nilai-nilai ini secara panjang lebar nanti di bab ini.

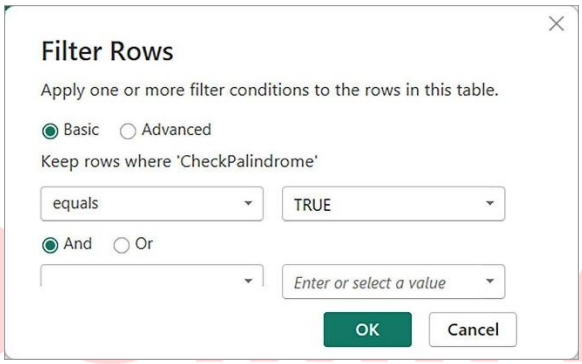
Perhatikan bahwa di **Applied Steps** Terapan di sebelah kanan kode Editor Lanjutan, setiap nama variabel dalam pernyataan *let* terluar akan muncul tercantum sebagai langkah terpisah, tetapi ekspresi peralihan dalam pernyataan *let* bersarang kita tidak.

Jika kita tidak menumpuk ekspresi *Letters*, *Reversed*, dan Palindrome dalam pernyataan *let* kedua tersebut, tetapi menambahkannya sebagai langkah terpisah dalam pernyataan *let* terluar, ekspresi tersebut akan muncul dalam **Applied Steps** (Langkah yang Diterapkan). Karena ekspresi tersebut merupakan sub-langkah dari proses yang diperlukan untuk menghasilkan hasil **CheckPalindrome** dan tidak diperlukan untuk tujuan lain dalam ekspresi tersebut, kita memilih untuk menumpuknya dalam pernyataan *let* kedua tersebut. Hal ini memastikan logika kita diwakili oleh satu langkah terapan, bukan tiga langkah terpisah.

Perhatikan juga bahwa di samping langkah *AddCheckPalindrome* dan *FilterResult* muncul ikon roda gigi kecil. Kehadiran ikon roda gigi tersebut menunjukkan bahwa Anda dapat mengekliknya (atau mengeklik dua kali pada nama langkah) lalu mengubah langkah tersebut menggunakan UI Power Query.

Misalnya, jika kita mengklik roda gigi di samping langkah *FilterResult*, layar berikut akan muncul, dan kita dapat merevisi kondisi filter menggunakan

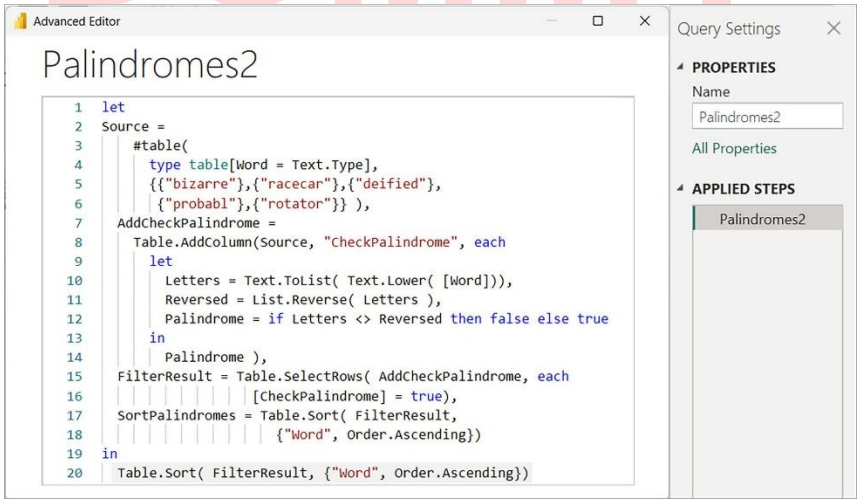
UI, dan Power Query akan menuliskan perubahan kembali ke ekspresi kode M kita:



Gambar 4. 10 Contoh pilihan yang muncul saat mengklik ikon roda gigi di samping langkah tertentu

Setelah Anda merasa nyaman menulis kode M langsung di **Advanced Editor**, Anda mungkin menemukan bahwa membuat jenis perubahan ini dapat dilakukan lebih cepat dan lebih mudah melalui pengodean daripada menggunakan UI. Namun, ikon roda gigi memungkinkan Anda menggunakan UI jika diinginkan.

Penting juga untuk dicatat bahwa kita dapat menempatkan operasi akhir kita setelah kata kunci in, seperti yang ditunjukkan pada tangkapan layar berikut:



Gambar 4. 11 Mengubah ekspresi akhir akan meruntuhkan list Langkah yang Diterapkan

Alih-alih merujuk ke variabel, bagian setelah **in** sekarang menunjukkan ekspresi. Perubahan ini memang mengarah ke hasil yang diinginkan, namun perhatikan bagaimana hal itu memengaruhi list **Applied Steps**. Semua operasi dikonsolidasikan menjadi satu langkah, yang diberi label dengan nama kueri, *Palindromes2*. Kekuatan utama Power Query adalah kemampuan untuk menavigasi setiap ekspresi dalam Applied Steps untuk memeriksa hasilnya. Oleh karena itu, menyusun kueri Anda untuk mengembalikan satu nama variabel setelah in terakhir, dan dengan itu, menunjukkan setiap langkah transformasi, dianggap sebagai praktik terbaik.

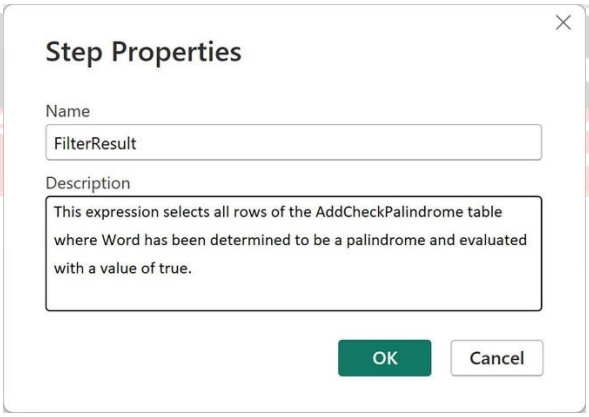
2. Praktik Terbaik Pengkodean Untuk Ekspresi

Selain menulis ekspresi let terluar Anda dengan satu pengenal setelah operator in agar list Applied Steps tidak runtuh, berikut ini beberapa rekomendasi lain untuk menulis ekspresi yang kita anggap sebagai praktik terbaik:

- **Gunakan pengenal deskriptif**: Mungkin Anda tergoda untuk menggunakan huruf tunggal, akronim, atau singkatan untuk pengenal ekspresi Anda. Namun, melakukannya kemungkinan akan membuat kode Anda lebih sulit dipahami orang lain (atau bahkan Anda sendiri satu atau dua bulan dari sekarang).
- **Hindari penggunaan spasi dalam pengenal Anda**: Ada dua alasan berbeda untuk menghindari spasi:
    - Jika pengenal Anda menyertakan spasi, Anda harus merujuknya dalam ekspresi berikutnya dengan tagar dan tanda kutip (misalnya, *#"Filter Result"*). Ini memberi sinyal kepada Power Query bahwa itu adalah pengenal tunggal, bukan variabel terpisah. Simbol tambahan ini dapat membuat kode Anda tampak berantakan dan sulit dibaca, dan juga merupakan sumber kesalahan potensial yang tidak perlu.
    - Power Query memiliki kemampuan untuk memanggil skrip Python dan R secara langsung dari dalam ekspresi kode M. Kemampuan ini

memberi M kekuatan super – kemampuan untuk melakukan analisis statistik tingkat lanjut, pembelajaran mesin yang kompleks, dan banyak lagi. Namun, baik Python maupun R lebih menyukai pengenal tanpa spasi. Jadi, jika Anda ingin memanfaatkan fitur ini, sebaiknya gunakan pengenal yang tidak mengandung spasi.

- **Gunakan huruf besar yang konsisten untuk pengenal**: Jika Anda memilih untuk mematuhi rekomendasi sebelumnya, sebaiknya pilih huruf besar tertentu untuk pengenal Anda dan patuhi itu demi konsistensi dan keterbacaan. Huruf besar tertentu yang Anda pilih untuk digunakan (misalnya, PascalCase, camelCase, snake_case, dll.) sebagian besar merupakan masalah preferensi pribadi.
- **Comment**: Ada dua cara untuk mengomentari kode M Anda pada tingkat ekspresi individual:
  - Anda dapat menggunakan // di awal baris untuk mengomentari baris tersebut dan menggunakannya untuk memberikan komentar deskriptif tentang ekspresi terkait. Atau, /* dan */ dapat digunakan untuk membuka dan menutup blok komentar multi-baris.
- Jika Anda klik kanan pada ekspresi dalam list **Applied Steps**, Anda dapat memilih **Properties** dan akan muncul kotak dialog berikut untuk Anda memasukkan komentar terkait ekspresi tersebut:



Gambar 4. 12 Menggunakan Properti Langkah untuk mendokumentasikan ekspresi

Dalam kedua kasus, komentar Anda akan muncul dalam warna hijau di **Advanced Editor**, dan ikon informasi kecil yang dilingkari sekarang akan muncul di sisi kanan langkah yang relevan dalam list **Applied Steps**.

Sekarang setelah kita membahas struktur dasar ekspresi, selanjutnya mari kita alihkan perhatian kita ke struktur kontrol, yang membantu Anda mengendalikan alur logika program dalam kueri Anda.

E. Struktur Kendali

Saat kita terus menguraikan komponen kueri dan ekspresi yang memungkinkan kita menggunakan nilai dengan cara yang berbeda, kita akan melihat sekilas peran struktur kontrol dalam M. Struktur kontrol adalah konstruksi yang memanipulasi aliran eksekusi, yang memungkinkan program untuk bercabang ke arah yang berbeda dan mengulang bagian kode berdasarkan evaluasi kondisi.

Bagian ini singkat, bukan karena struktur kontrol tidak penting, tetapi karena (seperti yang kita bahas di Bab 1) M adalah bahasa fungsional dan karenanya memiliki struktur kontrol yang jauh lebih sedikit daripada bahasa prosedural seperti C, bahasa berorientasi objek seperti Python atau C++, atau bahasa berbasis peristiwa seperti JavaScript.

Faktanya, M hanya memiliki satu struktur kontrol formal – pernyataan *if-then-else*, atau dikenal sebagai kondisional. Ini berfungsi persis seperti namanya. Jika suatu kondisi bernilai benar, maka kembalikan ekspresi benar (klausa *then*). Jika kondisi bernilai salah, maka kembalikan ekspresi salah (klausa *else*). Jika kondisi menghasilkan nilai non-logis (baik benar maupun salah), maka kesalahan akan dikembalikan. Misalnya, ekspresi sederhana berikut mengembalikan 1:

```
if true then 1 else 0
```

Demikian pula, ekspresi sederhana ini mengembalikan 0:

```
if false then 1 else 0
```

Contoh berikut mengembalikan kesalahan karena kondisinya adalah nilai teks, bukan nilai logika:

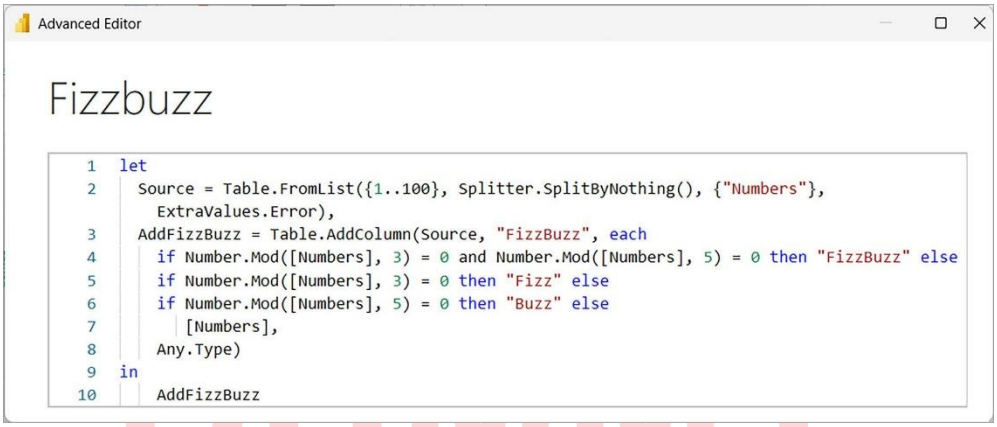
```
if "12345" then 1 else 0
```

Terakhir, penting untuk dicatat bahwa meskipun logika benar sering dilambangkan dengan angka 1 dan logika salah dilambangkan dengan angka 0, ekspresi berikut juga menghasilkan kesalahan:

```
if 1 then 1 else 0
```

Hal ini karena angka satu bukanlah nilai logika melainkan nilai angka. Anda dapat menumpuk pernyataan *if-then-else* sedemikian rupa sehingga jika kondisinya bernilai benar, maka ekspresi benar dapat berupa pernyataan *if-then-else* kedua, dan seterusnya. Sayangnya, M tidak memiliki struktur kontrol switch atau case seperti DAX atau SQL yang memungkinkan Anda memperhitungkan lebih dari dua hasil untuk kondisi tertentu. Dengan demikian, skenario tersebut biasanya ditangani menggunakan beberapa pernyataan *if-then-else* yang bertumpuk.

Contoh berikut yang menunjukkan penggunaan struktur kontrol *if-then-else* bertumpuk didasarkan pada tantangan pemrograman klasik yang disebut **FizzBuzz**, yang umumnya digunakan untuk mengevaluasi kandidat untuk posisi pengembang. Tugasnya adalah membuat list bilangan bulat dari 1 hingga 100, lalu untuk setiap angka, jika habis dibagi 3, nilainya adalah *Fizz*; jika habis dibagi 5, nilainya adalah *Buzz*; dan jika habis dibagi kedua angka, nilainya adalah *FizzBuzz*. Baris yang tidak memenuhi salah satu kondisi tersebut hanya akan mengembalikan angka pada baris tersebut. Untuk melakukannya, kita menggunakan kode berikut:



Gambar 4. 13 Ilustrasi pernyataan if-then-else memasukkan tantangan FizzBuzz

Demi kenyamanan, berikut adalah kueri FizzBuzz dalam bentuk teks:

```
let
  Source = Table.FromList({1..100}, Splitter.SplitByNothing(), {"Numbers"},
ExtraValues.Error),
  AddFizzBuzz = Table.AddColumn(Source, "FizzBuzz", each
    if Number.Mod([Numbers], 3) = 0
       and Number.Mod([Numbers], 5) = 0 then "FizzBuzz" else
    if Number.Mod([Numbers], 3) = 0 then "Fizz" else
    if Number.Mod([Numbers], 5) = 0 then "Buzz" else
       [Numbers],
    Any.Type)
in
    AddFizzBuzz
```

Cuplikan layar berikut menunjukkan 15 baris pertama yang dikembalikan oleh kueri:

	Numbers	FizzBuzz
1	1	1
2	2	2
3	3	Fizz
4	4	4
5	5	Buzz
6	6	Fizz
7	7	7
8	8	8
9	9	Fizz
10	10	Buzz
11	11	11
12	12	Fizz
13	13	13
14	14	14
15	15	FizzBuzz

Gambar 4. 14 Contoh data dari kueri FizzBuzz

Jadi apa yang terjadi dengan kueri ini? *Number.Mod* adalah fungsi yang mengambil dua nilai jenis angka, membagi yang pertama dengan yang kedua, dan mengembalikan sisanya. Jadi, dengan menggunakan fungsi tersebut dalam struktur kontrol bersarang, masalah ini menjadi cukup mudah:

- **Baris 1**: Ini memulai ekspresi kita dengan *let*.
- **Baris 2**: Ini membuat list bilangan bulat menggunakan operator *list* dan mengubah list tersebut menjadi tabel.
- **Baris 3**: Ini menambahkan kolom bernama *FizzBuzz* ke tabel untuk menampung hasil evaluasi kita.
- **Baris 4**: Ini adalah awal dari struktur kontrol *if-then-else* kita. Pertama, kita mengevaluasi kondisi yang paling ketat (*FizzBuzz*) sehingga struktur kontrol tidak keluar sebelum waktunya ketika kedua kondisi bernilai benar. Jika kedua kondisi bernilai benar, nilai *FizzBuzz* dikembalikan dan program

keluar dari struktur kontrol. Jika tidak bernilai *true*, program akan melanjutkan ke baris berikutnya.

- **Baris 5**: Ketika baris ini dijalankan, kita tahu bahwa kedua kondisi tidak bernilai *true*, jadi kita menguji setiap kondisi secara individual. Pertama, kita menguji *Number.Mod* dengan pembagi 3. Jika itu membagi nilai kita secara merata, kita mengembalikan nilai *Fizz*. Jika tidak, lanjutkan ke Baris 6.
- **Baris 6**: Di sini, kita menguji *Number.Mod* dengan pembagi 5. Jika pembagi tersebut membagi nilai kita secara merata, kita akan mengembalikan nilai *Buzz*. Jika tidak, lanjutkan ke **Baris 7**.
- **Baris 7**: Ini adalah pernyataan umum kita jika tidak ada pernyataan sebelumnya yang bernilai *true*. Jika struktur kontrol mencapai titik ini, maka kita cukup mengembalikan nilai angka asli.
- **Baris 8**: Ini menetapkan tipe data kolom *FizzBuzz* yang kita tambahkan ke tipe teks karena berisi campuran nilai teks dan angka. Kita akan membahas proses ini, yang disebut sebagai "atribusi tipe data," secara panjang lebar di bab berikutnya.
- **Baris 9–10**: Ini hanya menutup ekspresi let awal kita, dan mengembalikan nilai akhir ekspresi kita.

Masalah ini merupakan skenario yang sangat umum pada banyak aplikasi dunia nyata, itulah sebabnya masalah ini sering digunakan sebagai pertanyaan pengujian. Perhatikan bahwa kondisi awal dalam struktur kontrol kita dapat berupa sesuatu yang sangat sederhana seperti [warna] = "merah", atau ekspresi let bersarang ganda yang sangat kompleks. Yang diperlukan hanyalah kondisi awal mengembalikan nilai logika (benar atau salah).

Meskipun M hanya memiliki satu struktur kontrol formal, bukan berarti bahasa ini bukan bahasa yang canggih. Bahasa ini memiliki fungsi dan kapabilitas lain yang dapat mereplikasi tindakan struktur kontrol dalam bahasa lain:

- Fungsi *List.Accumulate* dapat mengeksekusi ekspresi beberapa kali yang ditentukan, dengan cara yang sama seperti struktur kontrol loop for yang umum
- Fungsi *List.Generate* dapat mengeksekusi ekspresi berulang kali dalam jumlah yang tidak ditentukan selama kondisi evaluasi tetap benar dengan cara yang sama seperti struktur kontrol loop while yang umum
- M juga menyediakan simbol @ untuk digunakan dalam membuat fungsi rekursif khusus yang dapat mereplikasi perilaku kontrol loop for atau while.

Terakhir, banyak program, seperti R, menyertakan fungsi peta yang bertindak sebagai jenis struktur kontrol, yang menerapkan fungsi tertentu ke semua baris kolom. M memiliki beberapa cara untuk melakukan ini, termasuk fungsi *Table.TransformColumn* dan *List.Transform*.

Struktur perulangan dan rekursif yang lebih kompleks ini akan menjadi fokus utama Bab 13, Iterasi dan Rekursi, karena keduanya sangat fleksibel dan canggih, dengan beragam aplikasi umum dan praktis.

Sekarang kita sampai pada komponen terakhir yang terkait dengan nilai, yang dikenal sebagai enumerasi.

F. Pencacahan

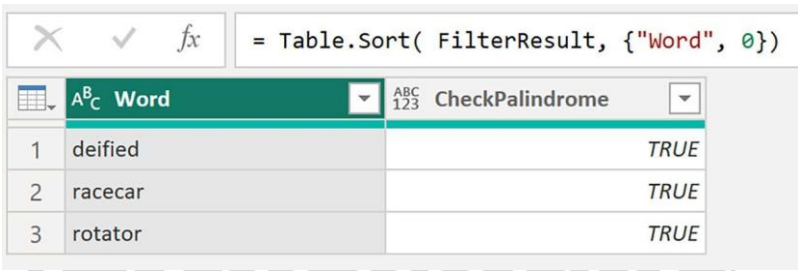
Enumerasi merupakan konsep dasar dalam pemrograman. Enumerasi merupakan kumpulan nilai bernama yang menentukan perilaku fungsi. Nama-nama ini memudahkan kita untuk memilih opsi dalam kode, sehingga maksud kita menjadi jelas. Pengguna dapat menentukan enumerasi dengan menggunakan nomor indeks yang sesuai atau representasi tekstualnya. Representasi tekstual sering kali membuat kode lebih mudah dipahami.

Ambil fungsi *Table.Sort* sebagai contoh. Fungsi ini mengurutkan data tabel menggunakan enumerasi untuk menentukan urutannya. Ingat kembali bahwa kita membahas pengurutan kata palindrom dalam urutan menaik. Untuk melakukannya, kita membuat ekspresi berikut:

```
SortPalindromes =

  Table.Sort(FilterResult, {"Word", Order.Ascending})
```

Di sini, *Order.Ascending* memberi tahu cara mengurutkan tabel *FilterResult*. Ada cara lain untuk menuliskannya yang menghasilkan hasil yang sama, yang ditunjukkan pada gambar berikut:



Gambar 4. 15 Metode alternatif untuk menentukan nilai enumerasi menggunakan angka

Kode ini berfungsi karena, dalam M, setiap opsi enumerasi memiliki padanan integer yang dapat kita gunakan sebagai ganti nilai teks terkait enumerasi tersebut. Sekilas pandang pada dokumentasi daring Microsoft untuk enumerasi ini menunjukkan Anda memiliki opsi untuk menggunakan 0 atau *Order.Ascending* untuk urutan menaik, dan 1 atau *Order.Descending* untuk urutan menurun.

Name	Value	Description
Order.Ascending	0	Sorts the values in ascending order.
Order.Descending	1	Sorts the values in descending order.

Gambar 4. 16 Nilai yang diizinkan untuk enumerasi Order.Type

Hal menarik tentang enumerasi adalah enumerasi dapat ditukar berdasarkan nomor indeks yang mendasarinya. Misalnya, *Day.Sunday* dan *Order.Ascending* keduanya memiliki nilai angka yang sesuai, yaitu 0. Jadi, jika *Day.Sunday* digunakan dalam *Table.Sort*, ia akan membacanya sebagai 0 dan mengurutkannya dalam urutan menaik. Ini menunjukkan bagaimana angka di belakang nama dapat mengarah ke penggunaan yang berbeda, di mana nilai

indeks yang sama dapat mengarah ke perilaku yang berbeda secara kontekstual di seluruh fungsi. Misalnya, cuplikan berikut mengembalikan hasil yang sama:

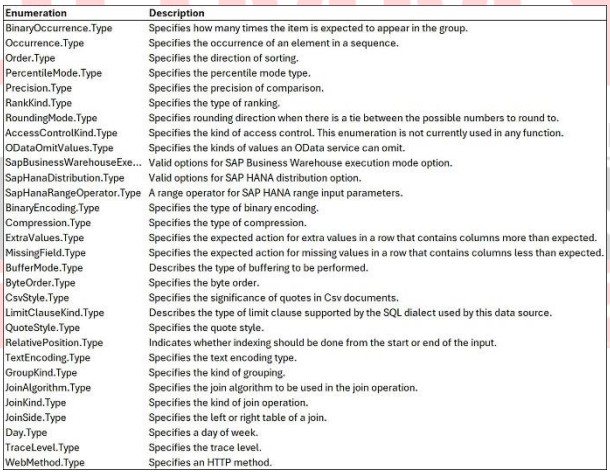
```
List.Sort( { 4, 3, 2, 1 }, Order.Ascending )
List.Sort( { 4, 3, 2, 1 }, Day.Sunday )
List.Sort( { 4, 3, 2, 1 }, 0 )
```

Ini mengilustrasikan bahwa enumerasi adalah sintaks gula untuk nilai integer. Fungsi yang menyertakan enumerasi telah dirancang untuk berperilaku dengan cara tertentu berdasarkan integer ini. Selain *Order.Type*, enumerasi lain yang umum digunakan adalah:

- *RoundingMode.Type*: Ini memberi tahu fungsi seperti *Number.Round*, *Currency.From*, dan *Int64.From* cara membulatkan angka. Apakah Anda perlu membulatkan ke atas, ke bawah, atau ke angka terdekat, enumerasi ini menjelaskannya dengan jelas.
- *MissingField.Type*: Tidak jarang menemukan kolom yang hilang dalam data. Enumerasi ini memungkinkan Anda memutuskan tindakan apa yang harus diambil—mengabaikan, mengembalikan kesalahan, dan sebagainya—ketika skenario seperti itu muncul. Ini berguna untuk fungsi seperti *Record.RemoveFields*, *Table.SelectColumns*, dan *Table.TransformColumns*.
- *GroupKind.Type*: Anda mungkin memiliki persyaratan yang berbeda saat mengelompokkan data. Enumerasi ini memandu bagaimana fungsi *Table.Group* memasukkan data ke dalam kelompok. Bergantung pada enumerasi yang digunakan, fungsi ini mematuhi urutan data saat mengelompokkan.
- *JoinKind.Type*: Menentukan sifat gabungan sangat penting saat menggabungkan set data dengan *Table.Join*, *Table.NestedJoin*, atau padanan fuzzy-nya. Enumerasi ini memandu jenis gabungan—bagian dalam, bagian luar, kiri, kanan, dan seterusnya.

- *Day.Type*: Untuk perhitungan terkait tanggal, menentukan hari mana yang memulai minggu dapat memengaruhi hasil kita. Enumerasi ini memungkinkan kita untuk menetapkan hari awal tersebut untuk fungsi seperti *Date.StartOfWeek*, *Date.EndOfWeek*, dan *Date.WeekOfMonth*.

Pada saat artikel ini ditulis, terdapat total 30 enumerasi berbeda dalam M, dengan banyak di antaranya digunakan dalam beberapa fungsi. Anda dapat melihat ikhtisar berikut:



Enumeration	Description
BinaryOccurrence.Type	Specifies how many times the item is expected to appear in the group.
Occurrence.Type	Specifies the occurrence of an element in a sequence.
Order.Type	Specifies the direction of sorting.
PercentileMode.Type	Specifies the percentile mode type.
Precision.Type	Specifies the precision of comparison.
RankKind.Type	Specifies the type of ranking.
RoundingMode.Type	Specifies rounding direction when there is a tie between the possible numbers to round to.
AccessControlKind.Type	Specifies the kind of access control. This enumeration is not currently used in any function.
ODataOmitValues.Type	Specifies the kinds of values an OData service can omit.
SapBusinessWarehouseExe...	Valid options for SAP Business Warehouse execution mode option.
SapHanaDistribution.Type	Valid options for SAP HANA distribution option.
SapHanaRangeOperator.Type	A range operator for SAP HANA range input parameters.
BinaryEncoding.Type	Specifies the type of binary encoding.
Compression.Type	Specifies the type of compression.
ExtraValues.Type	Specifies the expected action for extra values in a row that contains columns more than expected.
MissingField.Type	Specifies the expected action for missing values in a row that contains columns less than expected.
BufferMode.Type	Describes the type of buffering to be performed.
ByteOrder.Type	Specifies the byte order.
CsvStyle.Type	Specifies the significance of quotes in Csv documents.
LimitClauseKind.Type	Describes the type of limit clause supported by the SQL dialect used by this data source.
QuoteStyle.Type	Specifies the quote style.
RelativePosition.Type	Indicates whether indexing should be done from the start or end of the input.
TextEncoding.Type	Specifies the text encoding type.
GroupKind.Type	Specifies the kind of grouping.
JoinAlgorithm.Type	Specifies the join algorithm to be used in the join operation.
JoinKind.Type	Specifies the kind of join operation.
JoinSide.Type	Specifies the left or right table of a join.
Day.Type	Specifies a day of week.
TraceLevel.Type	Specifies the trace level.
WebMethod.Type	Specifies an HTTP method.

Gambar 4. 17 Enumerasi dalam M (sumber: https://learn.microsok.com/en-us/powerquery-m/enumer- ations)

Anda bisa mendapatkan informasi lebih lanjut tentang masing-masing enumerasi ini, termasuk fungsi mana yang mendukungnya dan bagaimana Anda dapat menggunakannya, di https://powerquery.how/enumerations/. Demikian pula, saat membaca dokumentasi tentang fungsi, Anda akan menemukan referensi ke dokumentasi enumerasi jika berlaku.

Singkatnya, enumerasi memainkan peran penting dalam membuat kode Anda lebih ekspresif dan lebih mudah dibaca. Dengan memungkinkan representasi numerik dan tekstual, pengguna dapat memilih antara keringkasan kode atau keterbacaan.

G. Ringkasan

Dalam bab ini, Anda telah memperoleh dasar yang kuat dari nilai-nilai dasar yang diperlukan untuk mulai membuat kueri M Anda sendiri, yang memungkinkan Anda untuk membersihkan dan mengubah data Anda. Kita telah membahas apa itu nilai dan ekspresi dan mengapa keduanya begitu penting dalam M. Kita telah menjelajahi empat kategori nilai dalam M dan 15 jenis nilai yang berbeda dalam kategori tersebut. Kita memperoleh pemahaman terperinci tentang setiap jenis nilai, termasuk strukturnya, kategori fungsi terkait untuk masing-masing, dan berapa banyak di antaranya yang saling membangun. Kita juga membahas beberapa pertimbangan khusus yang harus diperhitungkan saat menggunakan setiap jenis nilai untuk menghindari kesalahan kecil dan potensi ketidakakuratan. Terakhir, kita membahas "ikatan penghubung" yang digunakan untuk menghubungkan nilai-nilai bersama dalam ekspresi, termasuk operator, struktur kontrol, dan enumerator

Dalam bab-bab berikutnya, kita akan mengumpulkan dua elemen terakhir yang tersisa, yaitu, tipe data (Bab 5) dan nilai terstruktur (Bab 6). Berbekal pengetahuan ini, kita akan siap untuk tahap lanjutan dari perjalanan kita. Di sana, kita akan mempelajari cara merakit komponen-komponen dasar ini menjadi struktur yang semakin kompleks yang dapat membantu mengatasi hampir semua tantangan data yang mungkin kita hadapi.

H. Daftar Pustaka

1. Assimakopoulos, Vassilis and Nikolopoulos, K.. (2000). The theta model: A decomposition approach to forecasting. International Journal of Forecasting. 16. 521-530. https://www.researchgate.net/publication/223049702_The_theta_model_A_decomposition_approach_to_forecasting.
2. Rob J. Hyndman, Baki Billah. (2003). Unmasking the Theta method. International Journal of Forecasting. 19. 287-290. https://robjhyndman.com/papers/Theta.pdf.
3. Shannon, C.E. (1948), A Mathematical Theory of Communication. Bell System Technical Journal, 27: 379-423.

https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf.
4. Kaboudan, M. (1999). A measure of time series' predictability using genetic programming applied to stock returns. Journal of Forecasting, 18, 345-357: http://www.aiecon.org/ conference/efmaci2004/pdf/GP_Basics_paper.pdf.
5. Duan, M. (2002). TIME SERIES PREDICTABILITY: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.1898&rep=rep1&type=pdf.

I. Penutup

1. Tes Formatif

Tabel 4. 3 Tes Formatif Bab 4

No	Soal	Bobot
1.	Jelaskan apa yang dimaksud dengan nilai dalam konteks bahasa M. Mengapa pemahaman tentang nilai sangat penting dalam pengolahan data?	10
2.	Diskusikan perbedaan antara nilai primitif dan nilai terstruktur dalam M. Berikan contoh untuk masing-masing.	10
3.	Apa yang dimaksud dengan ekspresi dalam M? Berikan contoh ekspresi sederhana dan jelaskan bagaimana ekspresi tersebut berfungsi.	10
4.	Deskripsikan beberapa fungsi yang berhubungan dengan nilai tanggal. Bagaimana fungsi-fungsi ini dapat membantu dalam analisis data?	10
5.	Jelaskan struktur dasar dari nilai waktu dalam M. Apa saja fungsi yang tersedia untuk memanipulasi nilai waktu?	10
6.	Apa itu nilai DateTimeZone? Diskusikan perbedaan antara nilai DateTime dan DateTimeZone serta situasi di mana masing-masing lebih tepat digunakan.	10
7.	Diskusikan pentingnya tipe data dalam M. Bagaimana tipe data mempengaruhi proses transformasi dan analisis data?	10
8.	Jelaskan bagaimana operator digunakan untuk menggabungkan nilai dalam ekspresi M. Berikan contoh penggunaan operator aritmatika dan logika.	10
9.	Apa itu enumerasi dalam bahasa M? Bagaimana enumerasi dapat meningkatkan keterbacaan dan pemeliharaan kode?	10
10.	Diskusikan tantangan yang mungkin dihadapi saat bekerja dengan nilai null dalam M. Bagaimana cara mengatasi masalah tersebut dalam ekspresi?	10

GLOSARIUM

AAS	**Azure Analysis Services:** Layanan Platform-as-a-Service (PaaS) yang disediakan oleh Microsoft Azure untuk membuat solusi Business Intelligence (BI) tingkat perusahaan berbasis model data analitik
ACE	**Access Database Engine:** Komponen inti Microsoft Access yang bertanggung jawab untuk menyimpan dan mengambil data dalam format basis data Access (.accdb dan .mdb).
AD	**Active Directory:** Layanan direktori yang dikembangkan oleh Microsoft untuk mengelola dan mengatur sumber daya dalam jaringan komputer berbasis Windows Server.
ADO.NET	**ActiveX Data Objects .NET:** Seperangkat kelas dalam .NET Framework yang menyediakan akses terpadu ke berbagai sumber data.
AI	**Artificial Intelligence:** Bidang ilmu komputer yang berfokus pada pengembangan sistem komputer yang dapat melakukan tugas-tugas yang biasanya membutuhkan kecerdasan manusia.
API	**Application Programming Interface:** Sekumpulan aturan, protokol, dan alat yang memungkinkan berbagai aplikasi perangkat lunak untuk saling berkomunikasi dan bertukar data.
BI	**Business Intelligence:** Proses pengumpulan, pengorganisasian, analisis, interpretasi, dan penyajian data bisnis untuk membantu para pengambil keputusan membuat keputusan yang lebih baik dan lebih terinformasi.
COM	**Component Object Model:** Kerangka kerja biner yang dikembangkan oleh Microsoft yang memungkinkan aplikasi untuk berinteraksi dengan objek perangkat lunak tanpa perlu mengetahui detail implementasi internal objek tersebut.
CRM	**Customer Relationship Management:** Strategi bisnis yang berfokus pada membangun dan memelihara hubungan yang kuat dan positif dengan pelanggan.
CSS	**Cascading Style Sheets:** Bahasa stylesheet yang digunakan untuk menggambarkan tampilan (presentasi) dokumen yang ditulis dengan bahasa markup seperti HTML atau XML.
CSV	**Comma-Separated Value:** Format file teks sederhana yang digunakan untuk menyimpan data tabular (seperti spreadsheet atau database) di mana setiap baris mewakili satu catatan dan setiap kolom dalam catatan tersebut dipisahkan oleh tanda koma.
DBMS	**Database Management System:** Perangkat lunak yang memungkinkan pengguna untuk membuat, memelihara, dan berinteraksi dengan database secara efisien dan aman.

DOM	**Document Object Model:** Representasi terstruktur (berbentuk pohon) dari dokumen HTML atau XML yang memungkinkan program (seperti JavaScript) untuk mengakses dan memanipulasi konten, struktur, dan gaya dokumen tersebut.
DSN	**Data Source Name:** String koneksi yang berisi informasi yang dibutuhkan oleh aplikasi untuk terhubung ke sumber data tertentu.
ECMA	**European Computer Manufacturers Association:** Organisasi standar internasional untuk sistem informasi dan komunikasi.
ELT	**Extract-Load-Transform:** Proses integrasi data yang terdiri dari tiga tahap utama: Ekstraksi (Extract), Pemuatan (Load), dan Transformasi (Transform).
ERP	**Enterprise Resource Management:** Sebuah sistem yang mengintegrasikan proses bisnis inti perusahaan
EST	**Eastern Standard Time:** Zona waktu yang digunakan di sebagian Amerika Utara dan beberapa wilayah Karibia.
ETL	**Extract, Transform, And Load:** Proses integrasi data yang umum digunakan untuk memindahkan data dari berbagai sumber ke sistem target, seperti data warehouse, untuk tujuan analisis dan pelaporan.
GUI	**Graphical User Interface:** Jenis antarmuka pengguna yang memungkinkan pengguna berinteraksi dengan perangkat elektronik melalui elemen visual seperti ikon, menu, jendela, tombol, dan pointer (biasanya dikendalikan oleh mouse, trackpad, atau layar sentuh).
HDFS	**Hadoop Distributed File System:** Sistem file terdistribusi yang dirancang untuk menyimpan dan mengelola dataset yang sangat besar yang berjalan di atas perangkat keras komoditas (perangkat keras standar yang tidak terlalu mahal).
HL7	**Health Level 7:** Seperangkat standar internasional yang digunakan untuk pertukaran, integrasi, berbagi, dan pengambilan informasi kesehatan elektronik antara berbagai sistem perangkat lunak yang digunakan oleh penyedia layanan kesehatan.
HTML	**HyperText Markup Language:** Bahasa markup standar untuk membuat halaman web.
IT	**Information Technology:** Penerapan komputer dan sistem telekomunikasi untuk menyimpan, mengambil, mengirim, dan memanipulasi data atau informasi.
JDN	**Julian Day Number:** Jumlah hari yang telah berlalu sejak epoch Julian tengah siang di Greenwich pada tanggal 1 Januari 4713 SM (kalender Julian proleptik).
JSON	**JavaScript Object Notation:** Format data ringan yang digunakan untuk mentransmisikan data antara server dan aplikasi web, serta untuk menyimpan data terstruktur.

ODBC	**Open Database Connectivity:** Antarmuka pemrograman aplikasi (API) standar yang memungkinkan aplikasi untuk mengakses berbagai sistem manajemen basis data (DBMS) secara independen dari DBMS yang mendasarinya.
OLAP	**Online Analytics Processing:** Pendekatan untuk menganalisis data multidimensi dalam jumlah besar dengan cepat dari berbagai perspektif.
OLE DB	**Object Linking and Embedding Database:** Sekumpulan antarmuka yang dikembangkan oleh Microsoft yang menyediakan akses terpadu ke berbagai sumber data.
OLTP	**Online Transactional Processing:** Jenis pemrosesan data yang berfokus pada eksekusi dan pencatatan sejumlah besar transaksi kecil dan bersamaan secara real-time.
OPC	**Open Packaging Conventions:** Sebuah teknologi format file kontainer yang awalnya dibuat oleh Microsoft untuk menyimpan kombinasi file XML dan non-XML yang bersama-sama membentuk satu entitas logis.
PBRS	**Power BI Report Server:** Solusi pelaporan on-premises (di tempat) yang memungkinkan Anda untuk membuat, menerbitkan, dan mengelola laporan serta KPI (Key Performance Indicators) di dalam infrastruktur organisasi Anda sendiri.
PDF	**Portable Document Format:** Format file yang dikembangkan oleh Adobe untuk menyajikan dokumen, termasuk teks, gambar, hyperlink, font yang disematkan, dan lainnya, dengan cara yang independen dari perangkat lunak aplikasi, perangkat keras, dan sistem operasi.
SSAS	**SQL Server Analysis Services:** Komponen dalam Microsoft SQL Server yang menyediakan kemampuan pemrosesan analitik daring (OLAP) dan data mining.
SSIS	**SQL Server Integration Services:** Platform untuk membangun solusi integrasi data dan transformasi data berkinerja tinggi.
UI	**User Interface:** Bagian dari sistem (perangkat lunak atau perangkat keras) yang memungkinkan manusia berinteraksi dengannya.
URI	**Uniform Resource Identifier:** String karakter yang mengidentifikasi sumber daya fisik atau abstrak.
UTC	**Coordinated Universal Time:** Standar waktu utama dunia yang digunakan sebagai dasar untuk semua zona waktu sipil di seluruh dunia.
VoIP	**Voice over Internet Protocol:** Sekelompok teknologi yang memungkinkan Anda melakukan panggilan suara menggunakan koneksi internet broadband, bukan saluran telepon analog biasa.

WMS	**Warehouse Management System:** Sistem perangkat lunak yang dirancang untuk mengelola dan mengoptimalkan operasi gudang.
XML	**eXtensible Markup Language:** Bahasa markup yang dirancang untuk membawa data dan mendeskripsikan struktur data.

DUMMY
Penerbitan & Percetakan
UNP PRESS

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

TENTANG PENULIS

Dr. Phil. Dony Novaliendry, S.Kom., M.Kom., Lahir dan besar di Padang tanggal 4 November 1975. Merupakan anak dari Prof. Dr. H. Aljufri B. Syarif, M.Sc (Alm) dengan Endang Ratna Sulistri. Anak ke-4 dari 4 orang bersaudara. Menamatkan S1 di Universitas Gunadarma Jurusan Sistem Informasi dan di lanjutkan S2 di Universitas Gadjah Mada Jurusan Ilmu Komputer dan melanjutkan S3 di National Kaohsiung University of Science and Technology (NKUST) di Taiwan dengan bidang Bio-Informatics. Saat ini tertarik untuk mengembangkan diri di bidang bio-informatics, bio-medics, Artificial Intelligence, Decision Support System, Multimedia, Big Data dan Data Mining. Ini adalah buku yang ke enam yang diterbitkan. Semoga ditahun mendatang masih bisa terus berkarya menciptakan beberapa buku lagi.

Quote:Life Must Go On.

Saran, kritik dan kerjasama: dony.novaliendry@ft.unp.ac.id

RINGKASAN ISI BUKU

Buku "Panduan Definitif untuk Power Query (M) – Jilid 1" hadir sebagai sumber belajar komprehensif dalam memahami dan menguasai transformasi data kompleks menggunakan Power Query. Disusun secara sistematis, buku ini menguraikan konsep dasar hingga penerapan praktis bahasa M, yang menjadi inti dari Power Query.

Pembahasan dimulai dari pengenalan M, sejarah, karakteristik, hingga alasan mengapa M penting untuk dikuasai. Selanjutnya, buku ini memandu pembaca bekerja langsung dengan Power Query/M, mulai dari eksplorasi antarmuka, pengolahan data, pembuatan kolom khusus, hingga penggunaan editor tingkat lanjut.

Pada bagian berikutnya, pembaca diperkenalkan dengan cara mengakses, menggabungkan, dan menyatukan berbagai sumber data—mulai dari file, folder, konten web, hingga database—dilengkapi dengan teknik optimalisasi penggunaan fungsi bawaan M. Bab terakhir dalam jilid ini membahas nilai dan ekspresi, mencakup tipe data, operator, ekspresi bersarang, serta praktik terbaik dalam pengkodean.

Dilengkapi dengan kasus pemantik berpikir kritis, tes formatif, glosarium, dan lampiran, buku ini dirancang untuk memudahkan pemahaman sekaligus memperkuat keterampilan analisis data. Dengan pendekatan praktis, buku ini dapat menjadi referensi penting bagi peserta didik, dosen, praktisi data, maupun siapa saja yang ingin memperdalam pemanfaatan Power Query dalam dunia nyata.

DUMMY
Penerbitan & Percetakan
UNP PRESS

LAMPIRAN

1. Kartu Rencana Studi (KRS)

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI UNIVERSITAS NEGERI PADANG – DIREKTORAT AKADEMIK – SUBDIT. INOVASI PEMBELAJARAN DAN MBKM						
RENCANA PEMBELAJARAN SEMESTER						
MATA KULIAH (MK)	**KODE**	**Rumpun MK**	**BOBOT (sks)**		**SEMESTER**	**Tgl Penyusunan**
Praktikum Basis Data	INF1.62.4008	Mata kuliah Wajib Program Studi	1 SKS	Praktek	4	30 Januari 2024
OTORISASI / PENGESAHAN	**Dosen Pengembang RPS**		**Koordinator RMK**		**Koordinator PRODI**	
Direktorat Akademik, UNP **Subdit. Inovasi Pembelajaran dan MBKM (Dr. Nofrion, M. Pd)**	**1. Dr. Phil. Dony Novaliendry, S. Kom., M. Kom**		**Dr. Phil. Dony Novaliendry, S. Kom., M. Kom** **NIP. 197511042006041002**		**Dr. Yeka Hendriyani, S.Kom., M.Kom** **NIP. 198405202010122003**	

Capaian Pembelajaran	**CPL-PRODI yang dibebankan pada MK**	
	S5	Bertaqwa kepada Tuhan Yang Maha Esa dan mampu menunjukkan sikap religious dan Menunjukkan sikap bertanggungjawab atas pekerjaan di bidang keahliannya secara mandiri
	P2	Memahami konsep dasar matematika, ilmu listrik dan elektronika dalam bidang komputer
	KU5	mampu mengambil keputusan secara tepat dalam konteks penyelesaian masalah di bidang keahliannya, berdasarkan hasil analisis informasi dan data.
	KK 2	Kemampuan menguasai dasar pemograman phyton,metode komputasi Gauss dan komputasi metode LU Decomposition
	Capaian Pembelajaran Mata Kuliah (CPMK)	
	CPMK 1	Menguasai konsep bahasa pemrograman.
	CPMK 2	Mengidentikasi model- model bahasa pemrograman.
	CPMK 3	Membandingkan berbagai solusi.
	CPMK 4	Menguasai konsep-konsep basis data dan mampu membangun basis data untuk pngembangan sistem berbasis komputer
	Kemampuan akhir tiap tahapan belajar (Sub-CPMK)	
	SUB CPMK 1	Mahasiswa mampu memahami dan menganalisa konsep Bahasa pemrograman
	SUB CPMK 2	Mahasiswa mampu merumuskan konsep model-model Bahasa pemrograman
	SUB CPMK 3	Mahasiswa mampu merumuskan latar belakang konsep membandingkan berbagai solusi terkait Bahasa pemrograman.
	SUB CPMK 4	Mahasiswa mampu memahami, menganalisa, dan membangun konsep basis data, perancangan basis data, perancangan tabel-tabel, entry data pada tabel, relasi antar tabel, dan pembuatan report.

Peta CPL – CP MK

	Sub-CPMK1	Sub-CPMK2	Sub-CPMK3	Sub-CPMK4
CPMK 1	√			
CPMK 2		√		
CPMK 3			√	
CPMK 4				√

Deskripsi Singkat Mata Kuliah	Mata kuliah ini mempelajari dan menguasai konsep dan mengimplementasi pembuatan aplikasi basis data menggunakan salah satu DBMS dan bahasa pemrograaman basis data dengan urutan: perancangan tabel-tabel, entry data pada tabel, pembuatan form (untuk entry data dan untuk tampilan menu), pembuatan query, pembuatan report.
Bahan Kajian: Materi pembelajaran	1. Aplikasi DBMS, MariaDB, MySQL, PostgreSQL, phpMyAdmin 2. Bahasa pemograman sql, DDL 3. DML (Data Manipulation Language) 4. Arithmetic Operator, Agregate function, String function, Numeric function, Date/Time Function 5. Group by, group by with order, having 6. Fungsi agregat 7. Database Relation 8. Database Relation Join 9. Union, Intersect, Except 10. View dan control flow function 11. Create Procedure

Pustaka Buku 10 tahun terakhir Artikel 5 tahun terakhir Kecuali buku atau artikel yang memuat grand theory.	**Utama:** 1. Modul Praktikum Sistem Basis Data: Tim Dosen Program Studi Informatika 2. C. J. Date. 2006. An Introduction to Database Systems 8th. Pearson Education
	Pendukung: 1. Churcher, C., 2007, Beginning Database Design: From Novice to Professional (ebook available) 2. Oppel, A. & Sheldon, R., 2009, SQL: A Beginner's Guide- 3. Taylor, A.G., 2011, SQL Essential-All in One for Dummies
Dosen Pengampu	Dr. Phil. Dony Novaliendry, S. Kom., M. Kom NIP. 197511042006041002
Mata kuliah syarat	-

Mg Ke-	SUB-CPMK (Kemampuan Akhir Yang Diharapkan)	Penilaian		Bentuk Pembelajaran, Metode Pembelajaran, Penugasan Mahasiswa [Estimasi Waktu]		Materi Pembelajaran [Rujukan]	Bobot Penilaian (%)
		Indikator	Bentuk & Kriteria	Luring (Tatap Muka)	Daring (Online)		
1	Mahasiswa mengenal dan dapat menggunakan Aplikasi DBMS	1. Ketepatan menjelaskan aplikasi Xampp (MariaDB) 2. Ketepatan menjelaskan aplikasi MySQL 3. Ketepatan menjelaskan aplikasi PostgreSQL 4. Ketepatan menjelaskan aplikasi Oracle	Menggunakan Rubrik Penilaian	1. Presentasi 2. Praktek **TM : 1x (1 x 100 Menit)** 3. Penugasan Terstruktur **BM+BT : 1x(1x70 Menit)**	Bahan ajar materi dapat diunduh melalui LMS UNP pada tautan : https://elearning.unp.ac.id/my/	Mengenal Aplikasi DBMS - MariaDB - MySQL - PostgreSQL - Oracle	**10%**
2	Mahasiswa mampu menjelaskan perintah-perintah dasar SQL dan kelompok pernyataan SQL untuk pendefinisian basis data	1. Ketepatan menjelaskan DDL (Data Defenition Language)	Menggunakan Rubrik Penilaian	1. Presentasi 2. Praktek **TM : 1x (1 x 100 Menit)** 3. Penugasan Terstruktur **BM+BT : 1x(1x70 Menit)**	Bahan ajar materi dapat diunduh melalui LMS UNP pada tautan : https://elearning.unp.ac.id/my/	SQL: - Pengenalan SQL - DDL (Data Definition Language)	**5%**
3	Mahasiswa mampu menggunakan perintah-perintah DML	1. Ketepatan menjelaskan Data Manipulation Language	Menggunakan Rubrik Penilaian	1. Presentasi 2. Praktek **TM : 1x (1 x 100 Menit)** 3. Penugasan Terstruktur **BM+BT : 1x(1x70 Menit)**	Bahan ajar materi dapat diunduh melalui LMS UNP pada tautan : https://elearning.unp.ac.id/my/	DML (Data Manipulation Language)	**5%**
4	Mahasiswa Memahami perintah-perintah SQL untuk mengambil data dan kemudian melakukan perhitungan-perhitungan aritmatika dari datatersebut	1. Ketepatan menjelaskan Arithmetic Operators 2. Ketepatan menjelaskan agregate function 3. Ketepatan menjelaskan aString function 4. Ketepatan menjelaskan	Menggunakan Rubrik Penilaian	1. Presentasi 2. Praktek **TM : 1x (1 x 100 Menit)** 3. Penugasan Terstruktur **BM+BT : 1x(1x70 Menit)**	Bahan ajar materi dapat diunduh melalui LMS UNP pada tautan : https://elearning.unp.ac.id/my/	Operator, Agregate Function, String Function, Numeric Function, Date/Time Function.	**10%**

		5. Ketepatan menjelaskan Numeric function Date Function					
5	Mahasiswa mampu menggunakan perintah SQL untuk menyatukan dua atau lebih grup data kedalam suatu fungsi data tunggal	1. Ketepatan menjelaskan Group By 2. Ketepatan menjelaskan group by with order 3. Ketepatan menjelaskan having	Menggunakan Rubrik Penilaian	1. Presentasi 2. Praktek **TM : 1x (1 x 100 Menit)** 3. Penugasan Terstruktur **BM+BT : 1x(1x70 Menit)**	Bahan ajar materi dapat diunduh melalui LMS UNP pada tautan : https://elearning.unp.ac.id/my/	Group By, Group By with Order By, Having.	**10%**
6	Mahasiswa mampu menjalankan perintah-perintah fungsi Agregat pada tabel	1. Ketepatan menjelaskan fungsi agreagat	Menggunakan Rubrik Penilaian	1. Presentasi 2. Praktek **TM : 1x (1 x 100 Menit)** 3. Penugasan Terstruktur **BM+BT : 1x(1x70 Menit)**	Bahan ajar materi dapat diunduh melalui LMS UNP pada tautan : https://elearning.unp.ac.id/my/	Fungsi Agregat	**10%**
7	Mahasiswa mampu merelasikan tabel	1. Ketepatan menjelaskan relasi 2 tabel 2. Ketepatan menjelaskan relasi 3 tabel 3. Ketepatan menjelaskan lebih dari 3 tabel	Menggunakan Rubrik Penilaian	1. Presentasi 2. Praktek **TM : 1x (1 x 100 Menit)** 3. Penugasan Terstruktur **BM+BT : 1x(1x70 Menit)**	Bahan ajar materi dapat diunduh melalui LMS UNP pada tautan : https://elearning.unp.ac.id/my/	Database Relation - Relasi 2 tabel - Relasi 3 tabel - Relasi Lebih dari 3 tabel	**10%**
8	**Ujian Tengah Semester (UTS) = 10%**						
9	Mahasiswa mampu merelasikan tabel dengan menggunakan perintah Join	1. Ketepatan menjelaskan relasi antar tabel menggunakan join	Menggunakan Rubrik Penilaian	1. Presentasi 2. Praktek **TM : 1x (1 x 100 Menit)** 3. Penugasan Terstruktur **BM+BT : 1x(1x70 Menit)**	Bahan ajar materi dapat diunduh melalui LMS UNP pada tautan : https://elearning.unp.ac.id/my/	Relasi Antar Tabel - Join	**10%**
10	Mahasiswa mampu menjalankan perintah-perintah Union, Intersect, Except.	1. Ketepatan menjelaskan Union 2. Ketepatan menjelaskan Intersect 3. Ketepatan menjelaskan except	Menggunakan Rubrik Penilaian	1. Presentasi 2. Praktek **TM : 1x (1 x 100 Menit)** 3. Penugasan Terstruktur **BM+BT : 1x(1x70 Menit)**	Bahan ajar materi dapat diunduh melalui LMS UNP pada tautan : https://elearning.unp.ac.id/my/	- Union - Intersect - Except	**10%**
11-12	Mahasiswa Memahami dan mampu menggunakan View dan control flow function	1. Ketepatan menjelaskan view di database 2. Ketepatan menjelaskan control flow function	Menggunakan Rubrik Penilaian	1. Presentasi 2. Praktek **TM : 1x (1 x 100 Menit)** 3. Penugasan Terstruktur **BM+BT : 1x(1x70 Menit)**	Bahan ajar materi dapat diunduh melalui LMS UNP pada tautan : https://elearning.unp.ac.id/my/	View dan control flow function.	**10%**
13-15	Mahasiswa memahami dan mampu menggunakan Store Procedure dan Function yang merupakan perintah-perintah SQL yang diletakkan di dalam server database.	1. Ketepatan menjelaskan create procedure 2. Ketepatan menjelaskan create function	Menggunakan Rubrik Penilaian	1. Presentasi 2. Praktek **TM : 1x (1 x 100 Menit)** 3. Penugasan Terstruktur **BM+BT : 1x(1x70 Menit)**	Bahan ajar materi dapat diunduh melalui LMS UNP pada tautan : https://elearning.unp.ac.id/my/	Create Procedure dan Create Function.	**15%**
16	**Ujian Akhir Semester (UAS) = 10%**						