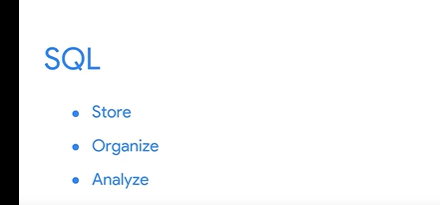
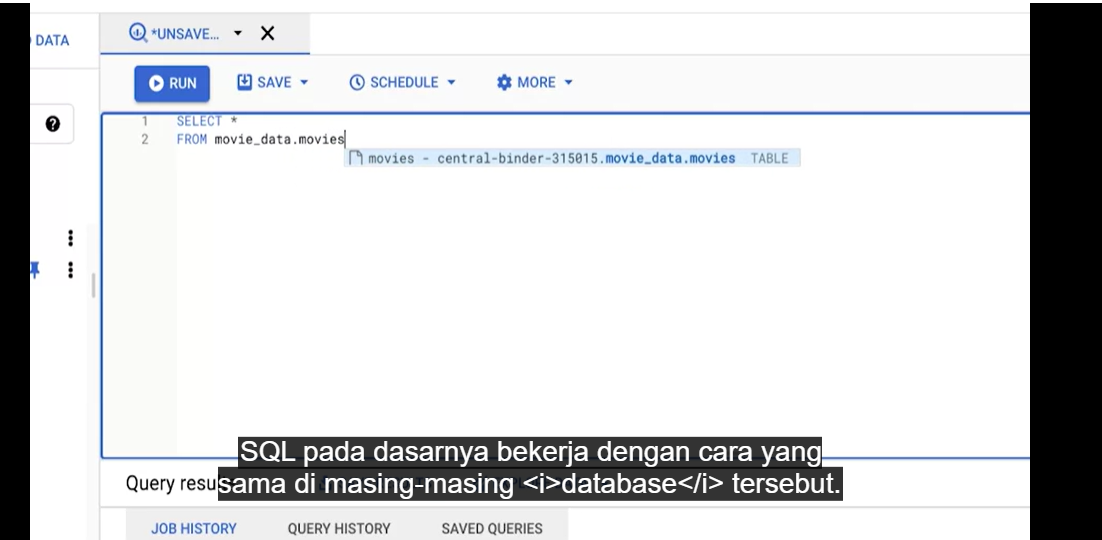
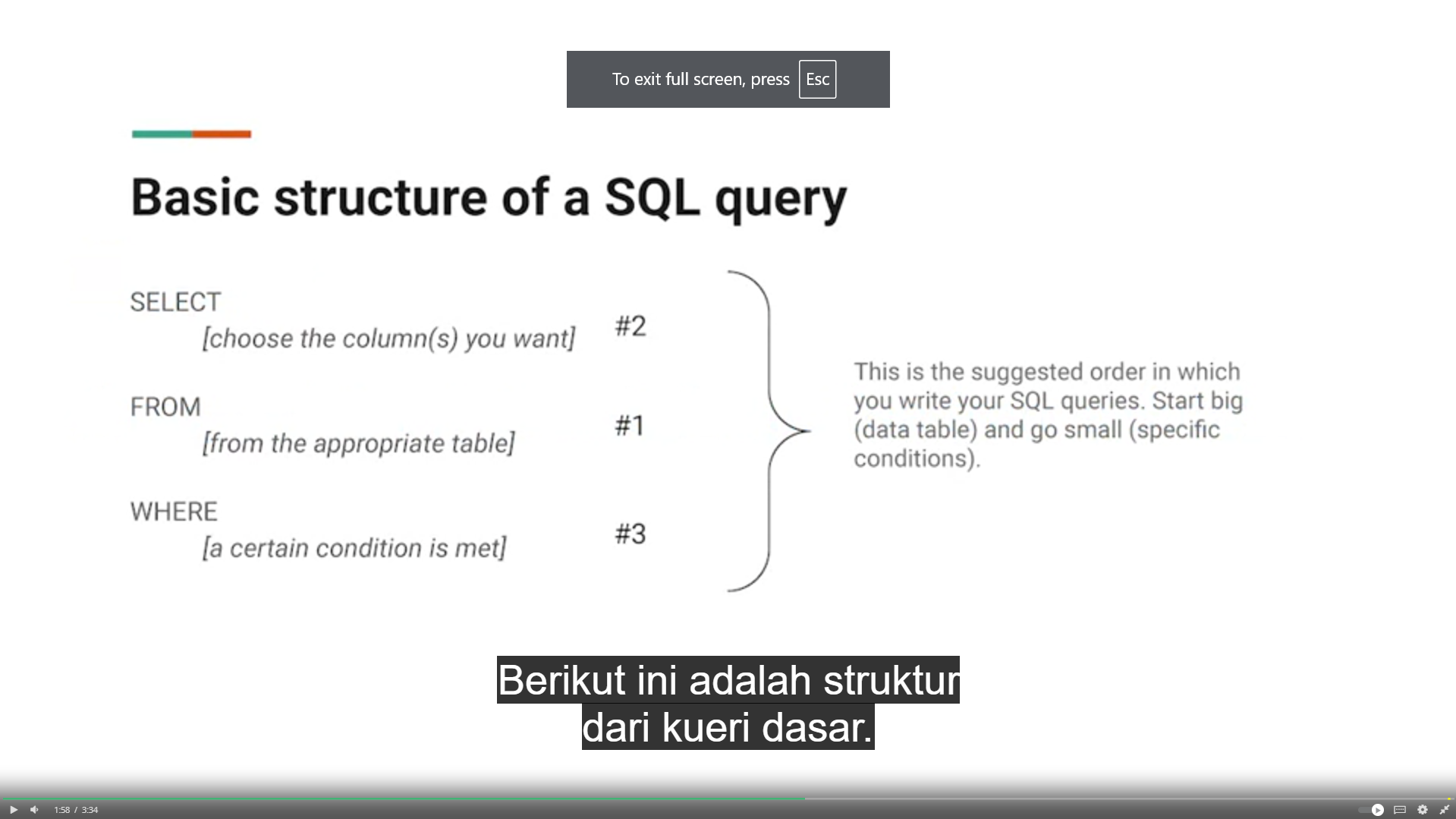
**Penggunaan SQL**

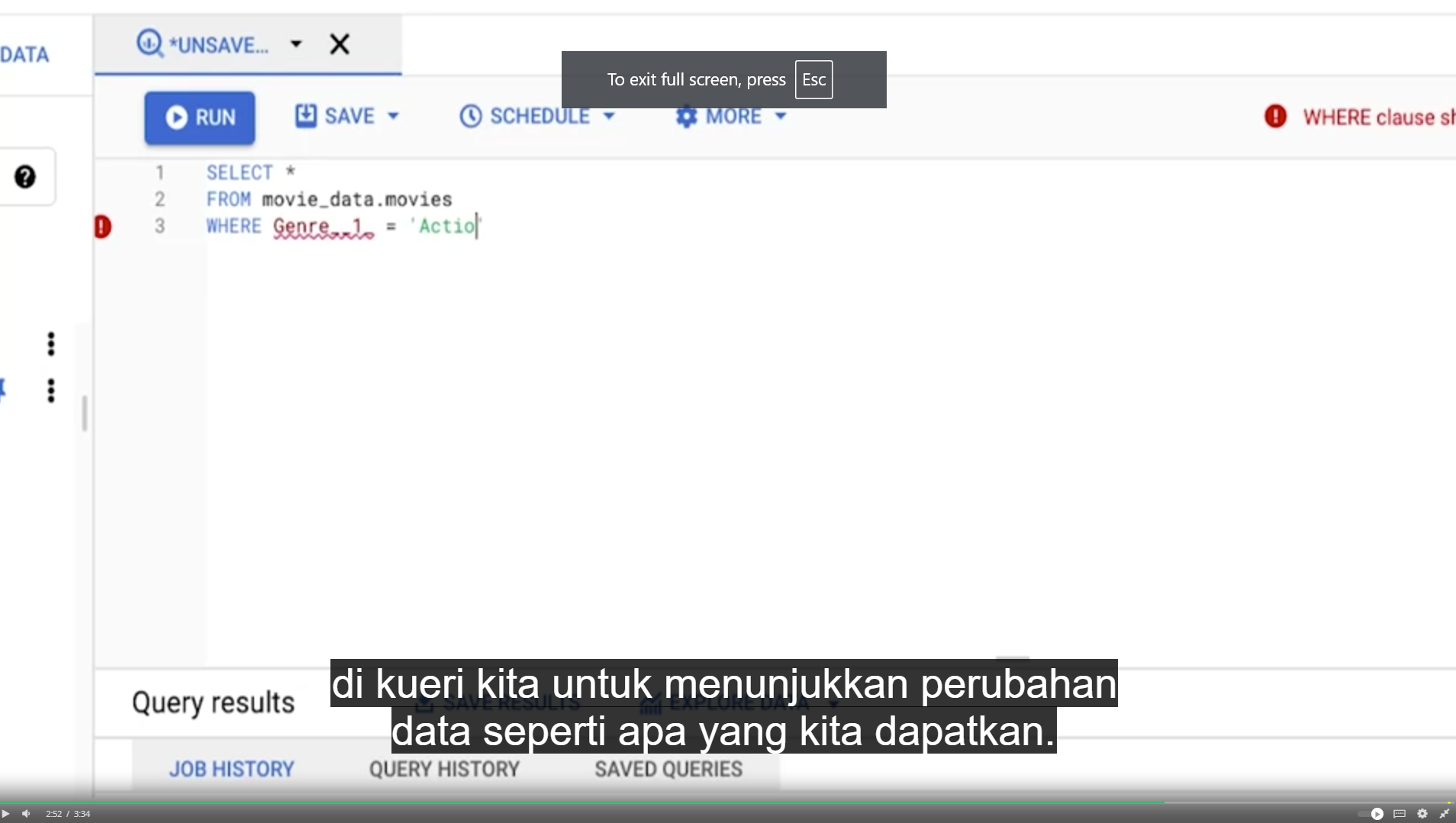
Save note





A query : A request for data or information from a database





Seperti yang mungkin Anda ingat sebelumnya, Kita menyinggung mengenai bahasa kueri SQL. Di video ini, Anda akan melihat penggunaan SQL dan mempelajari apa yang Anda dapat lakukan dengannya. Melalui beberapa contoh dari pertanyaan-pertanyaan tertentu. Saya rasa Anda bisa menyebut bahwa saat ini adalah lanjutan dari SQL. Kita akan mencoba membuat kali ini bahkan lebih baik dari aslinya. Ingat, SQL dapat melakukan banyak hal dengan data sama halnya seperti yang dilakukan oleh &lt;i&gt;spreadsheet.&lt;/i&gt; Di antaranya adalah Anda dapat menggunakannya untuk menyimpan, mengatur dan menganalisis data. Tetapi SQL dapat melakukannya dalam skala yang lebih besar dan lebih banyak penggunaan di dalamnya. Anggap saja sebagai &lt;i&gt;spreadsheet&lt;/i&gt; dengan ukuran super. Misalnya, Anda mungkin ingin mempertimbangkan sebuah &lt;i&gt;spreadsheet&lt;/i&gt; ketika Anda memiliki &lt;i&gt;dataset&lt;/i&gt; yang lebih kecil, contohnya satu dataset yang hanya terdiri 100 baris. Tetapi jika kumpulan data Anda tampaknya akan dipakai bertambah terus, dan &lt;i&gt;spreadsheet&lt;/i&gt; Anda mulai kewalahan, SQL akan menjadi cara yang masuk akal. Ketika Anda menggunakan SQL, Anda perlu tempat di mana bahasa SQL dapat dipahami. Jika Anda pernah pergi ke suatu tempat dan tidak mengenal bahasanya, hal ini dapat menjadi tantangan untuk berkomunikasi. Anda mungkin meminta sesuatu hal dan mendapatkan sesuatu hal lain yang benar-benar berbeda. Jadi, SQL memahami kondisi tersebut. SQL membutuhkan &lt;i&gt;database&lt;/i&gt; yang dapat memahami bahasanya. Mari kita lanjutkan. Ada beberapa &lt;i&gt;database&lt;/i&gt; di luar sana yang menggunakan SQL. Anda mungkin dapat menggunakan beberapa database tersebut selama Anda menjadi analis data. Tapi inilah masalahnya, tidak peduli &lt;i&gt;database&lt;/i&gt; mana yang Anda gunakan, SQL pada dasarnya bekerja dengan cara yang sama di masing-masing &lt;i&gt;database&lt;/i&gt; tersebut. Contohnya, pada SQL, kueri (query) bersifat universal. Kita telah berbicara mengenai kueri sebelumnya, tetapi hal itu tak pernah membosankan untuk dibicarakan kembali. Kueri (query) adalah permintaan untuk data atau informasi dari &lt;i&gt;database&lt;/i&gt;. Berikut ini adalah struktur dari kueri dasar. Anda dapat mengamati bahwa dengan kueri ini kita dapat memilih data tertentu dari tabel dengan menambahkan di mana kita bisa menyaring data sesuai dengan kondisi tertentu. Mari kita mulai. Kita akan membuka &lt;i&gt;database&lt;/i&gt; kita dan mengamati bagaimana SQL dapat berkomunikasi dengan data untuk melakukan beberapa tugas sederhana. Pertama, mari kita pilih kumpulan data kita. Kita akan gunakan tanda bintang (asterisk) untuk memilih semua data dari tabel. Dengan kueri sederhana tersebut, &lt;i&gt;database&lt;/i&gt; memanggil tabel yang kita butuhkan.

Play video starting at :2:40 and follow transcript2:40

Menakjubkan. Mari kita tambahkan perintah &lt;i&gt;Where&lt;/i&gt; di kueri kita untuk menunjukkan perubahan data seperti apa yang kita dapatkan.

Play video starting at :2:56 and follow transcript2:56

Anda dapat amati bahwa saat ini data hanya sekedar menampilkan, film-film yang ada di genre aksi. Itu saja, kueri dasar di SQL. Cukup menarik. Sebentar lagi Anda akan mempelajari bagaimana membangun kueri yang lebih kompleks. Untuk saat ini, kita senang bisa mempelajari mengenai struktur dari kueri SQL dasar, seperti perintah &lt;i&gt;select&lt;/i&gt;, &lt;i&gt;from&lt;/i&gt;, dan &lt;i&gt;where&lt;/i&gt;. Ketika Anda melanjutkan program ini, Anda memiliki kesempatan menggunakan SQL untuk diri sendiri. Semoga video ini menjadi pemanis yang berguna untuk materi yang akan datang.

# Panduan SQL: Langkah awal

Komputer sama seperti manusia yang menggunakan bahasa berbeda untuk berkomunikasi dengan orang lain. ***Structured Query Language*** (atau **SQL,** sering diucapkan "sequel") memungkinkan analis data untuk berkomunikasi dengan database. SQL adalah salah satu peralatan analis data yang paling berguna, terutama ketika mengolah dataset yang besar di tabel. SQL dapat membantu Anda memeriksa database yang besar, melacak teks (disebut sebagai string) dan angka, selain itu juga memfilter secara tepat jenis data yang dibutuhkan — jauh lebih cepat daripada spreadsheet.

Jika Anda belum pernah menggunakan SQL sebelumnya, bacaan ini akan membantu Anda mempelajari penggunaan dasarnya sehingga Anda memahami manfaat penggunaan SQL beserta kuerinya. Anda akan menulis kueri SQL dalam waktu singkat.

## **Apa yang dimaksud dengan kueri?**

**Kueri** adalah permintaan untuk mengakses data atau informasi di database. Saat membuat kueri di database, Anda menggunakan SQL untuk berkomunikasi mengenai pertanyaan atau permintaan Anda. Anda dan database selalu dapat bertukar informasi selama Anda berbicara dengan bahasa yang sama.

Setiap bahasa pemrograman, termasuk SQL, memiliki pedoman unik yang disebut ***syntax*.*Syntax*** adalah struktur bahasa yang telah ditentukan sebelumnya, mencakup semua kata, simbol, dan tanda baca yang diperlukan, serta ditempatkan secara tepat. Setelah Anda memasukkan kriteria pencarian dengan syntax yang benar, kueri mulai menarik data yang diminta dari database tujuan.

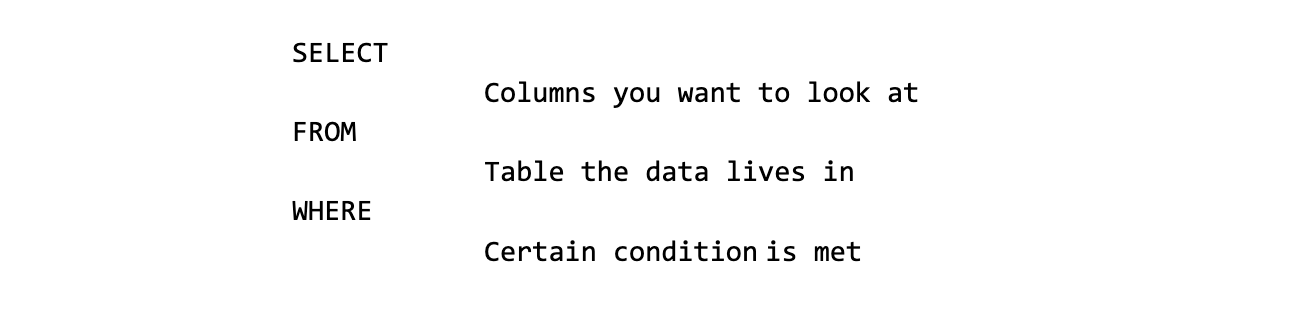
Syntax dari setiap kueri SQL adalah sama:

* Gunakan **SELECT** untuk memilih kolom yang akan dikembalikan.
* Gunakan **FROM** untuk memilih tabel tempat kolom yang diinginkan berada.
* Gunakan **WHERE** untuk memfilter informasi tertentu.

Kueri SQL seperti mengisi template. Jika Anda menulis kueri SQL dari awal, akan sangat membantu jika dimulai dengan menulis kata kunci SELECT, FROM, dan WHERE dalam format berikut:



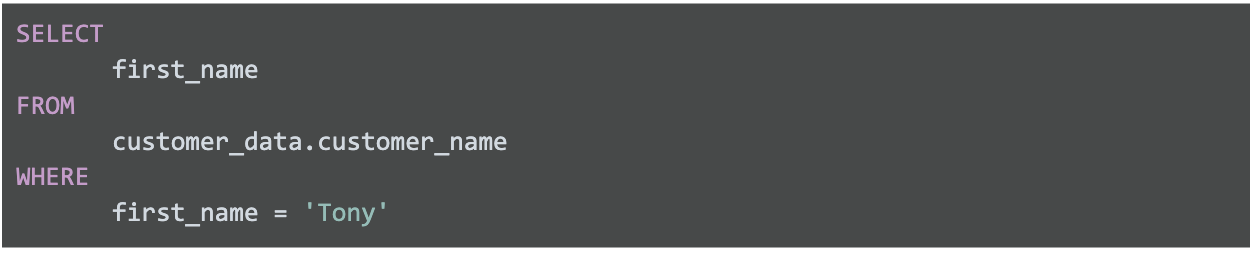
Selanjutnya, masukkan nama tabel setelah FROM; kolom tabel yang diinginkan setelah SELECT; dan, terakhir, kondisi yang akan ditempatkan pada kueri setelah WHERE. Pastikan untuk menambahkan baris dan inden baru saat menambahkan ini, seperti yang ditunjukkan di bawah ini:



Akan lebih mudah jika mengikuti metode ini setiap kali menulis kueri SQL. Hal ini juga membantu Anda membuat lebih sedikit kesalahan pada syntax.

### **Contoh kueri**

Berikut ini adalah kueri sederhana yang akan muncul di BigQuery, suatu data warehouse di Google Cloud Platform.



Kueri di atas menggunakan tiga perintah untuk menemukan pelanggan dengan nama depan Tony:

1. **SELECT** kolom **first\_name**
2. **FROM** tabel **customer\_name** (di dataset **customer\_data**) (Nama dataset selalu diikuti oleh titik, kemudian nama tabel.)
3. Hanya kembalikan data **WHERE Tony** ada di first\_name

Hasil kueri mungkin mirip dengan kueri berikut ini:

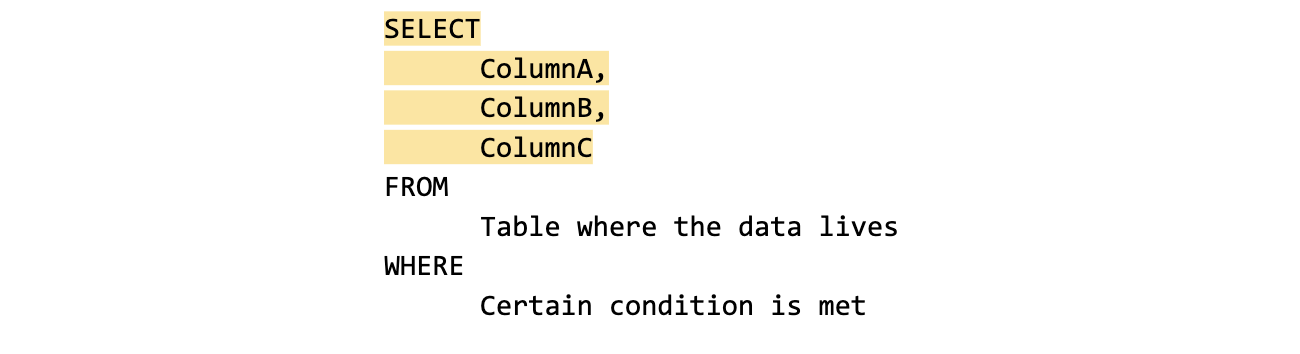
| **first\_name** |
| --- |
| Tony |
| Tony |
| Tony |

Bisa disimpulkan, kueri ini adalah syntax yang benar, tetapi tidak terlalu berguna setelah data dikembalikan.

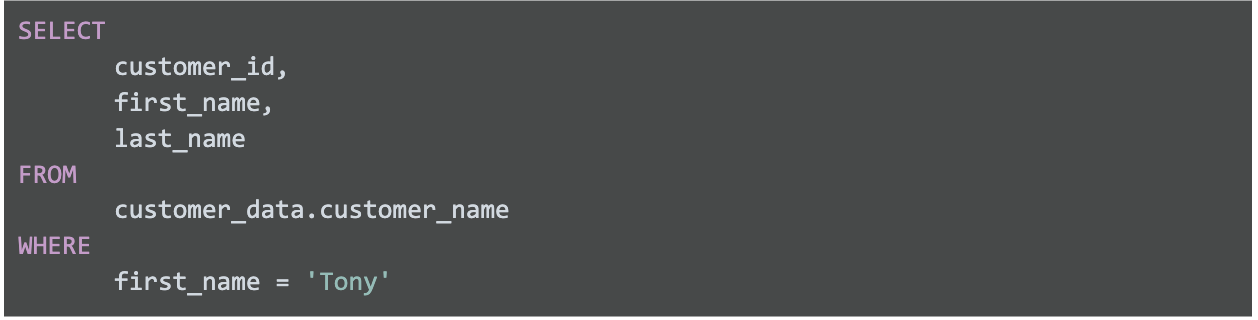
## **Beberapa kolom dalam sebuah kueri**

Di kehidupan nyata, Anda perlu mengolah lebih banyak data pelanggan yang sekadar bernama Tony. Beberapa kolom yang sama dipilih dengan perintah SELECT dapat di masukkan ke dalam kelompok yang sama.

Jika Anda meminta beberapa data field dari tabel, Anda perlu menyertakan kolom tersebut dengan perintah SELECT. Setiap kolom dipisahkan dengan koma seperti yang ditunjukkan di bawah ini:



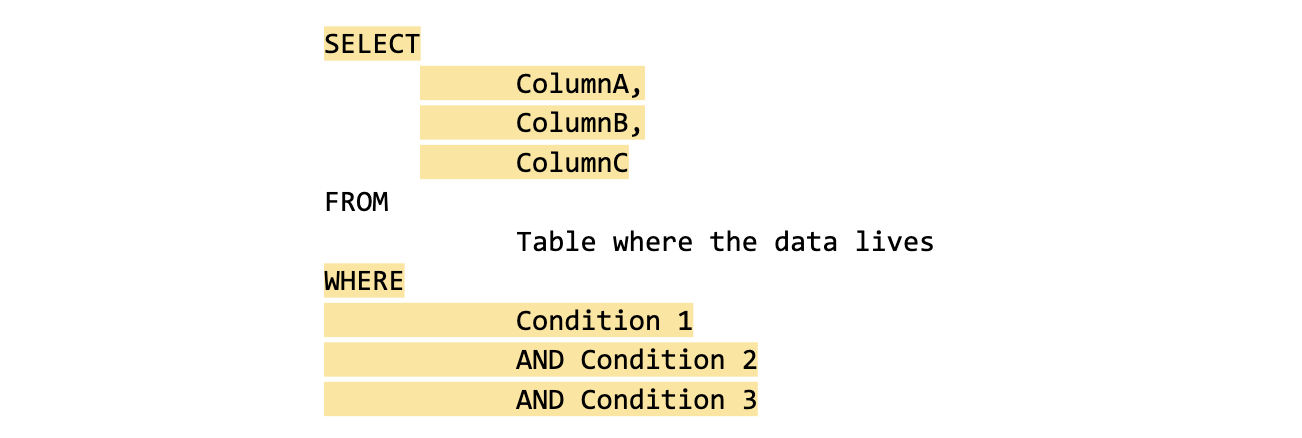
Berikut adalah contoh tampilan yang tampak di BigQuery:



Kueri di atas menggunakan tiga perintah untuk menemukan pelanggan dengan nama depan Tony.

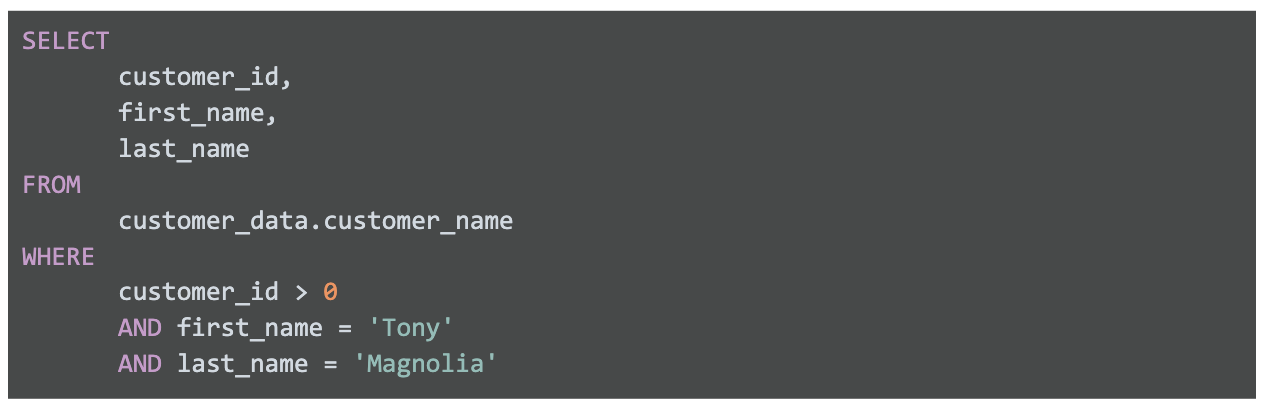
1. **SELECT** kolom **customer\_id, first\_name**, dan **last\_name**
2. **FROM** tabel **customer\_name** (di dataset **customer\_data**) (Nama dataset selalu diikuti oleh titik, kemudian nama tabel.)
3. Hanya kembalikan data **WHERE Tony** ada di first\_name

Satu-satunya perbedaan antara kueri ini dengan sebelumnya adalah ada lebih banyak kolom data yang dipilih. Kueri sebelumnya yang dipilih hanya first\_name sementara pada saat kueri ini memilih customer\_id dan last\_name selain first\_name. Secara umum, ini adalah penggunaan sumber daya yang lebih efisien untuk memilih kolom yang dibutuhkan. Misalnya, lebih logis untuk memilih lebih banyak kolom jika Anda akan menggunakan field tambahan menggunakan klausa WHERE. Jika Anda memiliki beberapa kondisi dengan klausa WHERE, dapat ditulis sebagai berikut ini:



Perhatikan tidak seperti perintah SELECT dengan menggunakan koma untuk memisahkan field/variabel/parameter, perintah WHERE menggunakan pernyataan AND untuk menghubungkan beberapa kondisi. Setelah Anda lebih mahir menullis kueri, Anda akan menggunakan konektor/operator lain seperti OR dan NOT.

Berikut ini adalah contoh BigQuery dengan beberapa field yang digunakan di klausa WHERE:



Kueri di atas menggunakan tiga perintah untuk menemukan pelanggan dengan ID pelanggan yang valid (lebih besar dari 0) dengan nama depan Tony dan nama belakangnya Magnolia.

1. **SELECT** kolom **customer\_id**, **first\_name**, dan **last\_name**
2. **FROM** tabel **customer\_name** (di dataset **customer\_data**) (Nama dataset selalu diikuti oleh titik, kemudian nama tabel.)
3. Tetapi, hanya kembalikan data **WHERE** customer\_id lebih besar dari **0**, first\_name adalah **Tony**, dan last\_name adalah **Magnolia.**

Perhatikan bahwa salah satu kondisi adalah kondisi logis yang memeriksa apakah customer\_id lebih besar dari nol.

Jika hanya satu pelanggan bernama Tony Magnolia, hasil kueri adalah sebagai berikut:

| **customer\_id** | **first\_name** | **last\_name** |
| --- | --- | --- |
| 1967 | Tony | Magnolia |

Jika lebih dari satu pelanggan yang memiliki nama yang sama, hasil kueri adalah sebagai berikut:

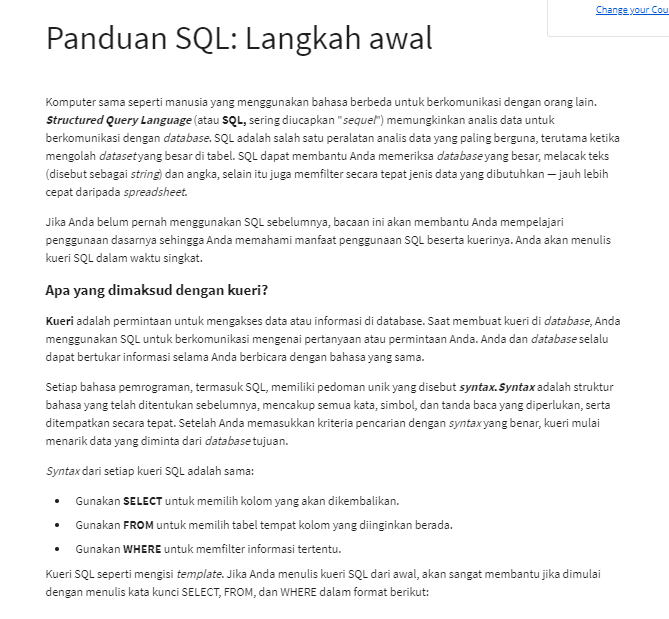
| **customer\_id** | **first\_name** | **last\_name** |
| --- | --- | --- |
| 1967 | Tony | Magnolia |
| 7689 | Tony | Magnolia |

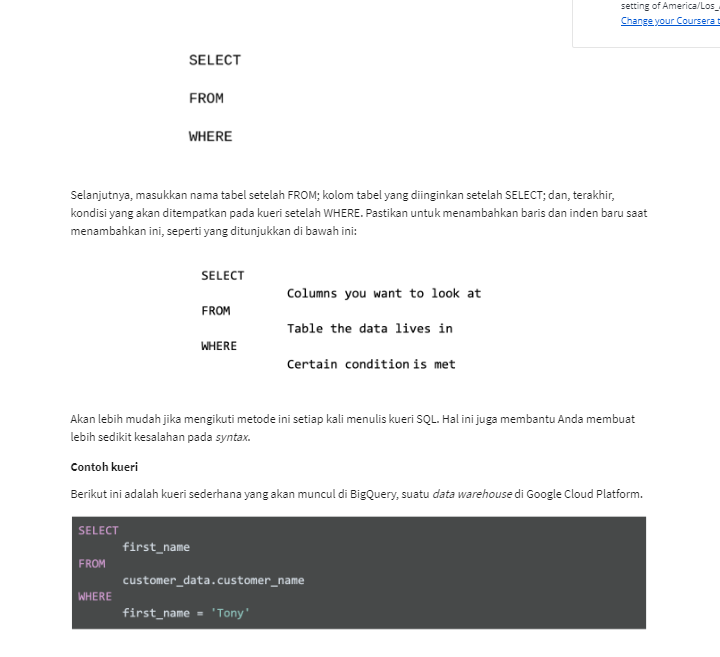
## **Hal penting yang bisa dipelajari**

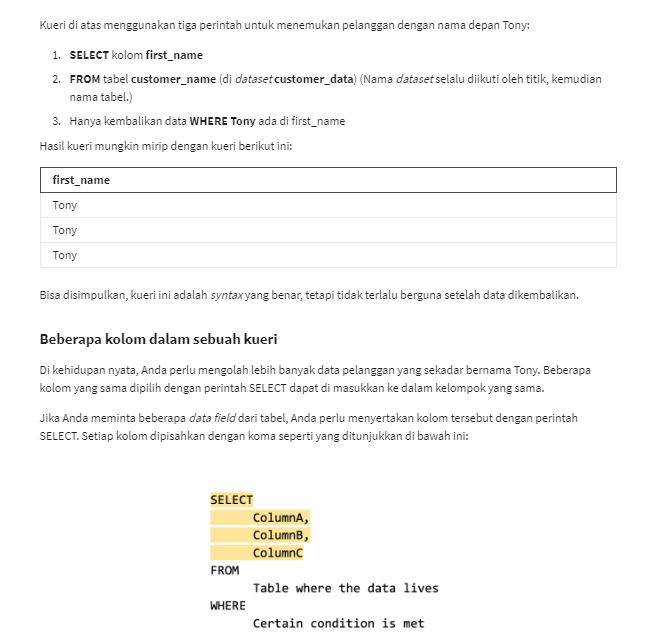
Hal terpenting untuk diingat adalah penggunaan SELECT, FROM, dan WHERE di dalam kueri. Kueri dengan beberapa field akan menjadi lebih sederhana setelah Anda berlatih menulis sendiri kueri SQL selanjutnya di program.

Go to next item

### Completed

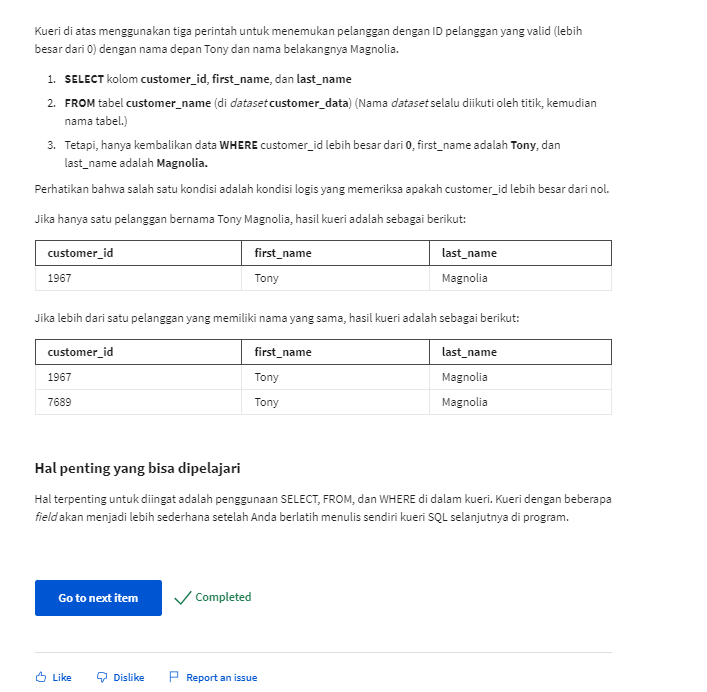












# Angie: Setiap hari giat berlatih mempelajari keterampilan baru

Saya adalah Angie, seorang manajer program

di bidang rekayasa di Google.

Saat ini saya sedang mengerjakan sertifikasi Analitik Data.

Sebelumnya, saya adalah seorang peneliti untuk analitik manusia.

Saya juga bagaikan seorang "pasukan analitik sewaan"

di banyak perusahaan berbeda

untuk bantu mereka memahami data perusahaan.

Setiap kali saya belajar keterampilan baru,

saya merasa seperti saya sedang belajar bagaimana saya mulai bicara.

Saya ingat kali pertama saya belajar SQL,

saya sangat frustrasi karena semua orang di sekitar saya,

lancar menggunakan SQL,

mereka sangat paham dengan apa yang mereka lakukan.

Saya ingat bahwa saya berjuang dengan hal-hal yang paling mendasar,

sama seperti mengeluarkan data dari tabel atau seingat saya

saat ada yang bertanya mengenai bagaimana cara membuat rata-rata

dari suatu data dan saya terus melakukan kesalahan.

Hal ini terasa seperti Anda

belajar bahasa baru dan Anda

masih berada di tingkat balita ketika

semua orang di sekitar Anda sudah fasih berbicara.

Orang tua saya berimigrasi ke

negara ini ketika mereka berusia sekitar 30 tahun.

Mereka harus belajar

bahasa lain, memulai

kehidupan baru, dan belajar bahasa Inggris.

Saya ingat sebagai seorang anak yang mengamati mereka

berjuang setiap hari untuk memahami bahasa baru,

untuk bisa berbicara hal-hal sederhana,

seperti meminta bantuan di toko kelontong.

Saya ingat ketika memanggil perusahaan kabel di saat saya masih berusia enam tahun,

saya bertanya mengenai tagihan

orangtua saya karena mereka tak bisa menanyakannya.

Saya ingat betapa sulitnya mereka memahami

bahasa baru dan berbicara fasih

dan setiap kali saya mempelajari

bahasa data yang baru seperti

SQL saya berpikir bagaimana sulitnya menggunakan bahasa tersebut.

Saya berpikir orang tua saya bisa melakukannya, maka saya juga bisa mempelajari SQL.

Jika mereka meminta bantuan untuk hal yang paling sederhana,

saya dapat menanyakannya ke Analis Data di sebelah saya bagaimana

cara menulis pernyataan SQL dan

bagaimana cara mendapatkan data dari tabel.

Hal tersebut benar-benar membantu saya,

dengan pola pikir tersebut

dan mengetahui bahwa saya dapat meminta bantuan.

# Kemungkinan SQL yang tak terbatas

Information:

This item includes content that hasn’t been translated into your preferred language yet.

Anda telah mempelajari bahwa kueri SQL menggunakan **SELECT**, **FROM**, dan **WHERE** untuk menentukan data mana yang akan dikembalikan. Bacaan ini memberikan informasi terperinci bagaimana memformat kueri, menggunakan kondisi WHERE, memilih semua kolom di tabel, menambahkan komentar, dan menggunakan alias. Semua ini memudahkan Anda untuk memahami (dan menulis) kueri ketika digunakan di SQL. Bab terakhir dari bacaan ini adalah memberikan contoh pengambilan data karyawan di sebuah proyek yang akan dikerjakan analis data.

## **Penulisan huruf besar, penempatan *indent*, dan penulisan titik koma**

Anda dapat menulis kueri SQL Anda dengan menggunakan huruf kecil dan tidak perlu khawatir mengenai ruang ekstra di antara kata. Tetapi, penggunaan huruf besar dan penempatan indent dapat membantu Anda membaca informasi dengan lebih mudah. Pastikan kueri Anda tetap rapi, dan lebih mudah ditinjau atau diperbaiki jika nanti Anda memeriksanya.



Perhatikan bahwa pernyataan SQL yang ditunjukkan di atas diakhiri dengan titik koma (semicolon). Titik koma merupakan penentu/pengakhir pernyataan dan bagian dari standar SQL-92 American National Standards Institute (ANSI), yang merupakan rekomendasi syntax umum yang diadopsi oleh semua database SQL. Tetapi, tidak semua database SQL telah mengadopsi atau menggunakan titik koma, jadi mungkin saja akan ditemukan beberapa pernyataan SQL yang tidak diakhiri dengan titik koma. Tidak mengapa, jika sebuah pernyataan kemudian berfungsi walau tidak diakhiri titik koma,

## **Kondisi WHERE**

Di kueri yang ditunjukkan di atas, klausa **SELECT** mengidentifikasi kolom mana yang ingin Anda ambil datanya berdasarkan nama, **field1**, dan klausa **FROM** mengidentifikasi tabel tempat kolom berada berdasarkan nama, **tabel.** Jadi, klausa **WHERE** mempersempit kueri Anda sehingga database hanya mengembalikan data dengan kecocokan nilai yang tepat atau data yang sesuai dengan kondisi tertentu yang akan dipenuhi.

Misalnya, jika Anda mencari pelanggan tertentu dengan nama belakang Chavez, klausa WHERE menjadi sebagai berikut:

**WHERE field1 = 'Chavez'**

Tetapi, jika Anda mencari semua pelanggan dengan nama belakang huruf awal "Ch," klausa WHERE akan menjadi sebagai berikut:

**WHERE field1 LIKE 'Ch%'**

Dapat disimpulkan bahwa klausa LIKE sangat berfungsi karena memungkinkan Anda untuk berkomunikasi dengan database dalam pencarian pola spesifik! Tanda persen (%) digunakan sebagai pengganti untuk mencocokkan karakter berjumlah satu atau lebih. Contoh di atas, data **Chavez** dan **Chen** akan dikembalikan. Perhatikan bahwa pada beberapa database, tanda bintang (\*) digunakan sebagai pengganti selain tanda persen (%).

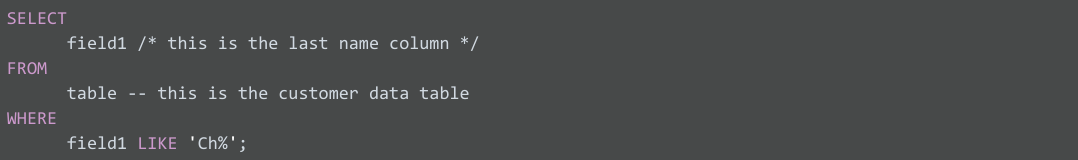
## **SELECT semua kolom**

Bisakah Anda menggunakan  **SELECT \*?**

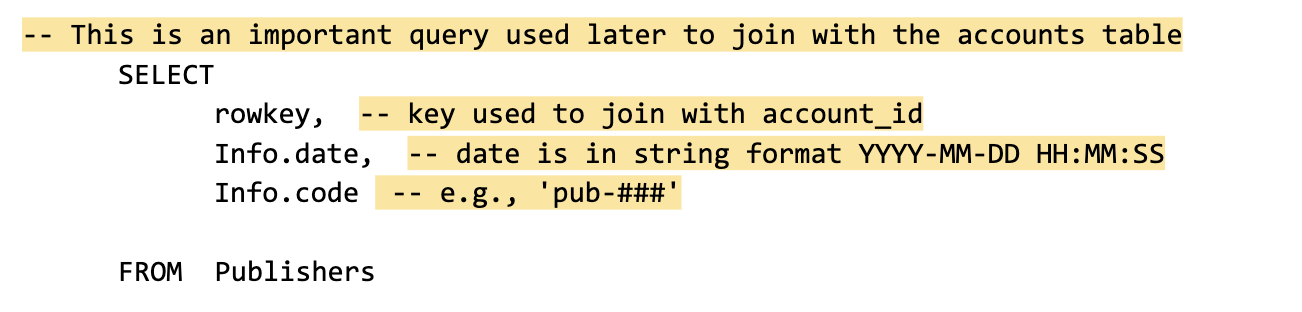
Di contoh, jika Anda mengganti **SELECT field1** dengan **SELECT \***, maka Anda akan memilih semua kolom di tabel selain kolom field1 saja. Dari sudut pandang syntax, pernyataan SQL ini benar, tetapi Anda harus menggunakan tanda bintang (\*) dengan hemat dan hati-hati. Tergantung dari berapa banyak kolom yang ada di sebuah tabel, bisa saja Anda memilih sejumlah data yang banyak sekali. Memilih terlalu banyak data dapat menyebabkan kueri berjalan lambat.

## **Komentar**

Beberapa tabel tidak dirancang dengan deskripsi konvensi penamaan. Di contoh, **field1** adalah kolom untuk nama belakang pelanggan, tetapi namanya tidak diketahui. **last\_name** adalah pemilihan nama yang lebih baik**.** Di kasus ini, Anda dapat menempatkan komentar di sepanjang SQL supaya Anda mengingat apa yang diwakili oleh nama tersebut. Komentar adalah teks yang ditempatkan di antara karakter tertentu, /\* dan **\*/**, atau setelah dua tanda hubung (--) seperti yang ditunjukkan di bawah ini.



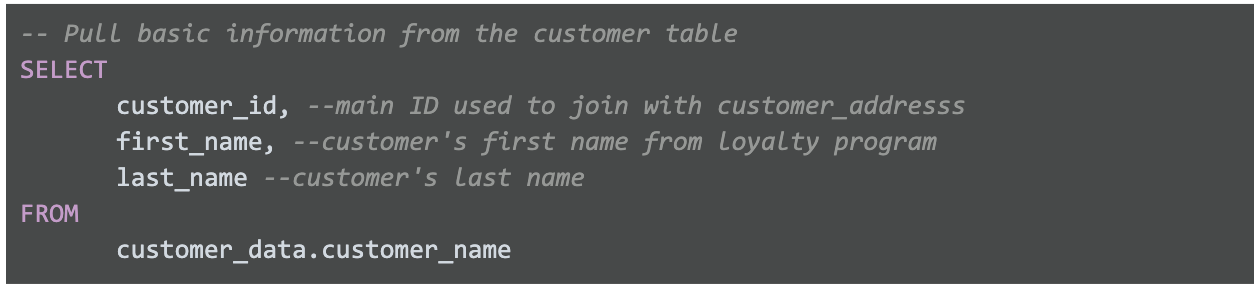
Komentar juga dapat ditambahkan di luar atau di dalam pernyataan. Anda dapat menggunakan fleksibilitas ini untuk memberikan deskripsi keseluruhan yang akan dilakukan, catatan mengenai langkah demi langkah bagaimana Anda mencapainya, dan mengapa Anda menetapkan parameter/kondisi yang berbeda.



Semakin Anda terbiasa dengan SQL, semakin mudah untuk membaca dan memahami kueri dalam sekilas. Tetapi, tidak ada salahnya untuk berkomentar di kueri untuk mengingatkan apa yang akan Anda lakukan nanti. Hal ini juga memudahkan orang lain untuk memahami kueri Anda jika membagikan kueri Anda tersebut ke orang lain. Ketika kueri Anda menjadi semakin kompleks, cara ini akan menghemat banyak waktu dan energi untuk memahami kueri kompleks yang telah Anda tulis berbulan-bulan atau bahkan bertahun-tahun yang lalu.

### **Contoh kueri dengan komentar**

Berikut ini adalah contoh komentar yang ditulis di BigQuery:



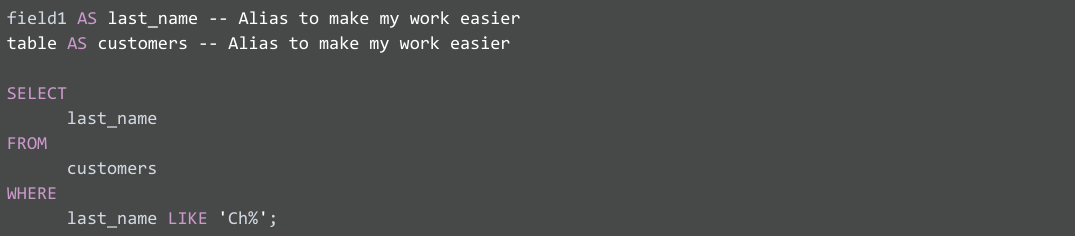
Di contoh di atas, suatu komentar telah ditambahkan sebelum pernyataan SQL untuk menjelaskan apa yang dilakukan kueri. Selain itu, komentar yang telah ditambahkan di samping masing-masing nama kolom berfungsi untuk mendeskripsikan kolom serta penggunaannya. Dua tanda hubung (--) umumnya didukung. Penggunaan terbaik adalah menggunakan -- dan secara konsisten menggunakannya. Anda dapat menggunakan # sebagai pengganti -- pada kueri di atas, tetapi # tidak dikenali di semua versi SQL; misalnya, MySQL tidak mengenali #.  Anda juga dapat menempatkan komentar di antara /\* dan \*/ jika database yang digunakan juga mendukungnya.

Ketika Anda mengembangkan keterampilan Anda secara profesional, tergantung pada database SQL yang digunakan, Anda dapat memilih simbol pembatasan komentar yang Anda sukai dan tetaplah konsisten menggunakannya. Ketika kueri Anda menjadi semakin kompleks, cara penambahan komentar yang bermanfaat akan menghemat banyak waktu dan energi untuk memahami kueri yang mungkin telah Anda tulis berbulan-bulan atau bahkan bertahun-tahun sebelumnya.

## **Alias**

Anda juga dapat mempermudah dengan menetapkan nama atau **alias** baru ke dalam kolom atau nama tabel untuk membuatnya lebih mudah digunakan (dan menghindari penggunaan komentar). Hal ini dilakukan dengan klausa SQL AS. Di contoh di bawah ini, alias **last\_name** penanda untuk **field1** dan alias **pelanggan** adalah penanda **tabel.** Alias ini hanya berlaku selama durasi kueri tersebut saja. Alias tidak mengubah nama sebenarnya dari kolom atau tabel dalam database.

## **Contoh kueri dengan alias**



## **Menggunakan SQL sebagai analis data**

Bayangkan jika Anda sebagai analis data di bisnis kecil dan manajer Anda meminta beberapa data karyawan. Anda memutuskan untuk menulis kueri dengan menggunakan SQL untuk mendapatkan data apa yang Anda butuhkan dari database.

Anda ingin mendapatkan semua kolom: **empID**, **firstName**, **lastName**, **jobCode**, dan **salary.** Karena Anda mengetahui jika database tidak sebesar itu, daripada memasukkan setiap nama kolom dengan menggunakan klausa **SELECT,** Anda menggunakan **SELECT \*.**  Hal ini akan memilih semua kolom yang ada di tabel Karyawan pada klausa **FROM**.



Sekarang, Anda bisa mendapatkan data lebih spesifik yang diinginkan dari tabel Karyawan. Jika Anda ingin semua data tentang karyawan yang bekerja dengan kode pekerjaan **SFI,** Anda dapat menggunakan klausa **WHERE** untuk memfilter data berdasarkan persyaratan tambahan ini.

Anda dapat menggunakan:



Sebagian data yang dihasilkan yang dikembalikan oleh kueri SQL mungkin terlihat sebagai berikut:

| **empID** | **firstName** | **lastName** | **jobCode** | **salary** |
| --- | --- | --- | --- | --- |
| 0002 | Homer | Simpson | SFI | 15000 |
| 0003 | Marge | Simpson | SFI | 30000 |
| 0034 | Bart | Simpson | SFI | 25000 |
| 0067 | Lisa | Simpson | SFI | 38000 |
| 0088 | Ned | Flanders | SFI | 42000 |
| 0076 | Barney | Gumble | SFI | 32000 |

Misalkan Anda melihat kisaran gaji yang besar untuk kode pekerjaan **SFI.** Anda mungkin ingin menandai semua karyawan yang memiliki gaji lebih rendah yang ada di semua departemen sebagai data untuk manajer Anda. Karena magang juga termasuk dalam tabel dan mereka memiliki gaji kurang dari $30.000, maka Anda ingin hasil pencarian data yang dilakukan memastikan bahwa hanya karyawan yang bekerja penuh dengan gaji $30.000 atau kurang akan termasuk. Dengan kata lain, Anda ingin mengecualikan magang dengan kode pekerjaan **INT** yang juga memiliki penghasilan kurang dari $ 30.000. Klausa **AND** memungkinkan Anda untuk menguji kedua kondisi tersebut.

Anda membuat kueri SQL yang mirip di bawah ini, di mana <> berarti "tidak sama":



Data yang dihasilkan dari kueri SQL terlihat sebagai berikut (magang dengan kode pekerjaan **INT** tidak dikembalikan):

| **empID** | **firstName** | **lastName** | **jobCode** | **salary** |
| --- | --- | --- | --- | --- |
| 0002 | Homer | Simpson | SFI | 15000 |
| 0003 | Marge | Simpson | SFI | 30000 |
| 0034 | Bart | Simpson | SFI | 25000 |
| 0108 | Edna | Krabappel | TUL | 18000 |
| 0099 | Moe | Szyslak | ANA | 28000 |

Dengan menggunakan SQL maka diperoleh akses data yang cepat, Anda dapat menyediakan data karyawan dari banyak sudut pandang yang berbeda ke manajer Anda, termasuk apakah gaji karyawan di perusahaan sudah sesuai dan merata. Bagusnya, kueri hanya menunjukkan dua karyawan tambahan yang mungkin memerlukan penyesuaian gaji dan Anda dapat membagikan data tersebut ke manajer Anda.

Mendapatkan data, menganalisisnya, dan menerapkan solusi pada akhirnya dapat membantu untuk meningkatkan kepuasan dan loyalitas karyawan. Hal ini membuat SQL menjadi alat yang cukup handal.

## **Sumber untuk mempelajari lebih lanjut**

Bukan pelanggan dapat mengakses sumber ini secara gratis, tetapi jika sebuah situs membatasi jumlah artikel gratis per bulannya dan apabila jumlah sumber yang Anda akses sudah mencapai batas, tandai sumber tersebut dan kembali lagi ke sana di kemudian hari.

* [Tutorial W3Schools SQL](https://www.w3schools.com/sql/default.asp): Jika Anda ingin mengeksplorasi tutorial SQL secara rinci, maka di sinilah tempat yang sempurna untuk mulai berlatih. Tutorial ini mencakup contoh interaktif yang dapat diedit, diuji, dan dibuat ulang. Gunakanlah sebagai referensi atau pelajari seluruh tutorial untuk berlatih menggunakan SQL. Klik tombol **Start learning SQL now** atau tombol **Next** untuk memulai tutorial.
* [Contekan SQL](https://www.sqltutorial.org/sql-cheat-sheet/): Untuk pelajar yang lebih mahir, buka artikel ini untuk menggunakan standar syntax SQL di PostgreSQL. Pada saat Anda selesai, Anda akan memahami lebih banyak tentang SQL dan siap menggunakannya sebagai analisis bisnis ataupun pekerjaan lainnya.

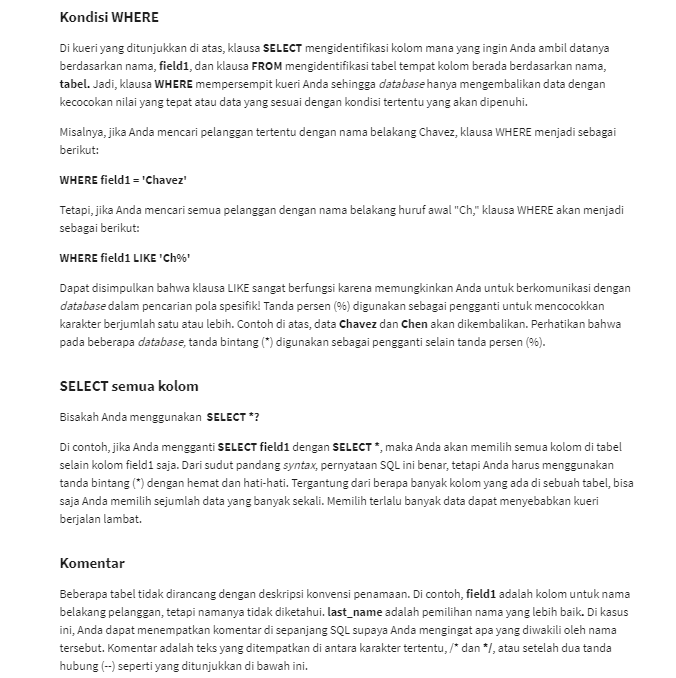
Go to next item

<https://www.w3schools.com/sql/default.asp>

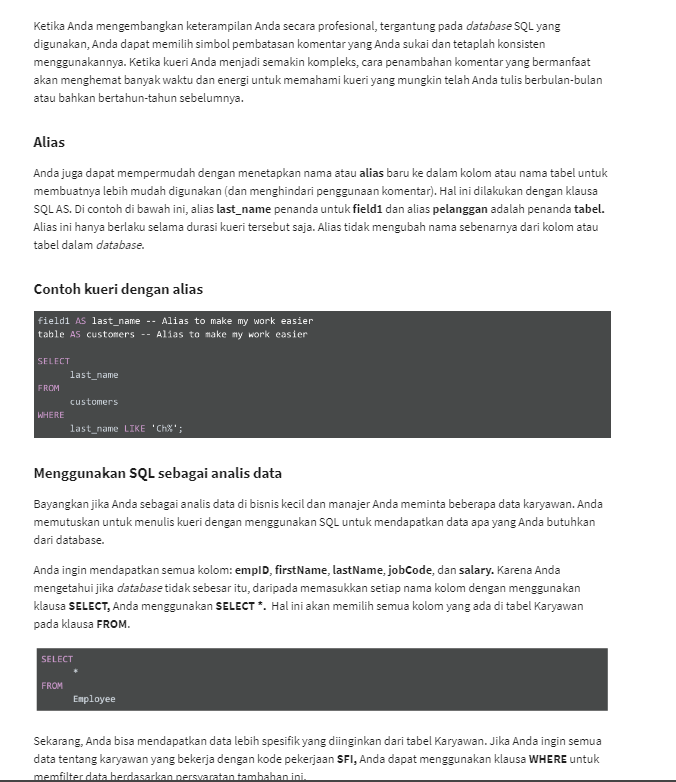
<https://www.sqltutorial.org/sql-cheat-sheet/>

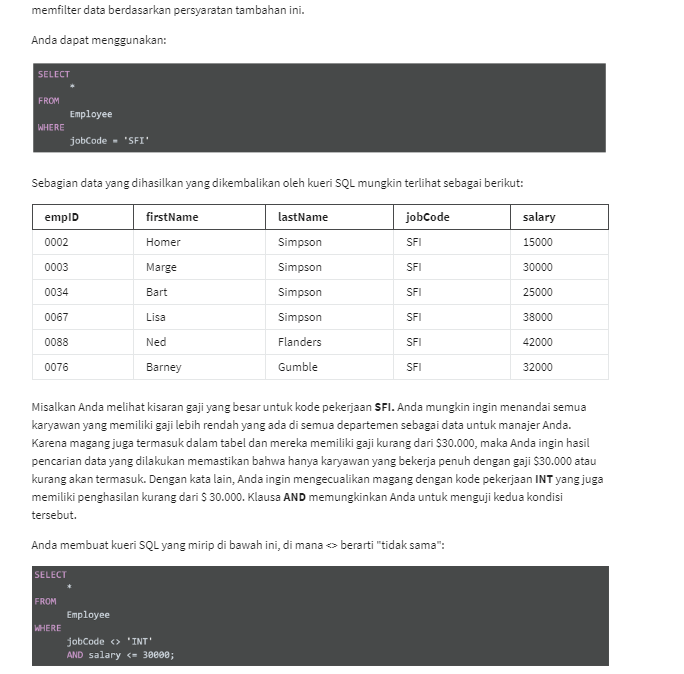
### Completed

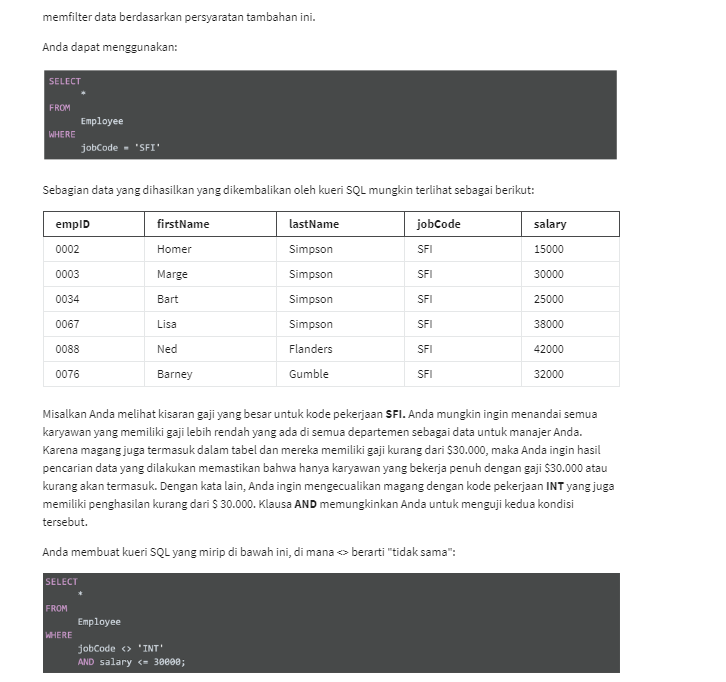


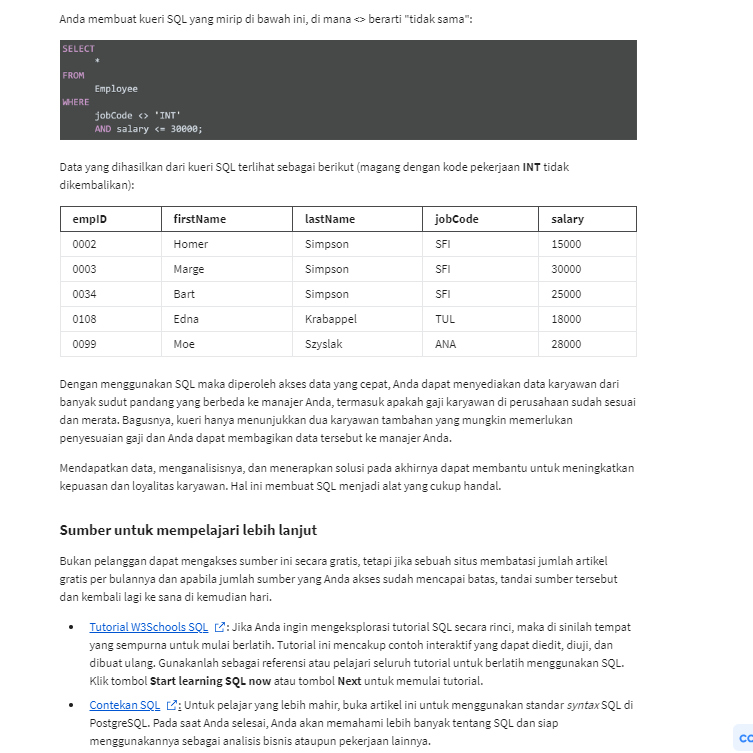


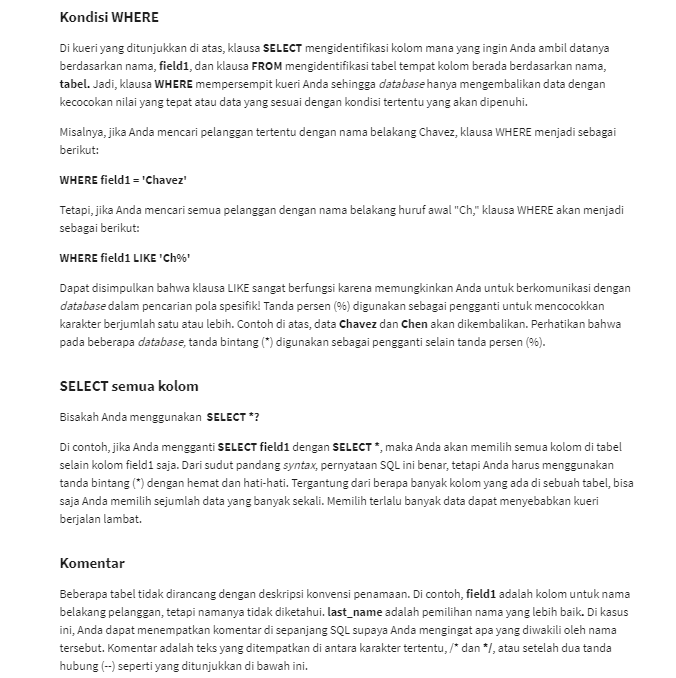
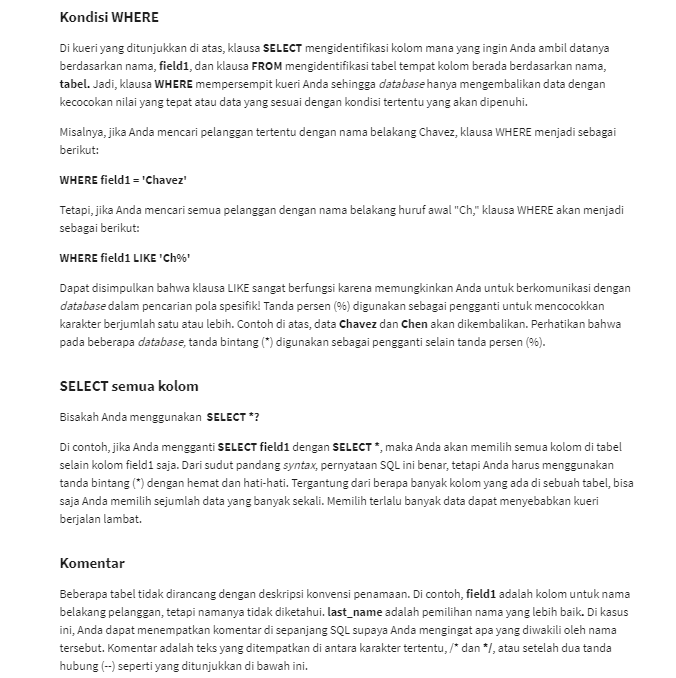






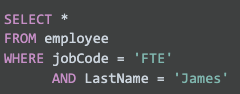






### 1.

Question 1



Apa yang diminta oleh tanda bintang (\*) setelah SELECT di kueri tersebut ke database?

1 / 1 point

Memilih kolom LastName dari tabel employee

Memilih semua kolom dari tabel employee

Memilih semua data yang memenuhi kriteria yang tertulis di kueri

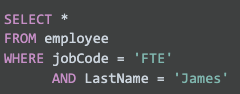
Memilih semua data yang memenuhi kriteria yang dituliskan dalam kueri, kemudian menggandakan data tersebut

Correct

SELECT \* meminta database untuk memilih semua kolom dari tabel employee. Kriteria pada klausa WHERE berfungsi memberitahu database data mana di kolom-kolom tersebut yang perlu ditampilkan.

### 2.

Question 2



Dengan kueri ini, dari tabel yang mana analis data akan mengambil data ?

1 / 1 point

LastName

jobCode

James

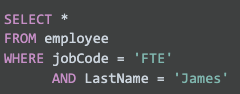
employee

Correct

Analis data akan mengambil data dari tabel employee.

### 3.

Question 3



Data apa yang akan diambil dari databasedengan menggunakan kueri ini?

1 / 1 point

Semua data dari tabel FTE di mana LastName dari karyawan adalah James.

Semua data dari tabel employee, dengan jobCode adalah FTE dan nama belakang adalah James.

Semua data dari tabel employee, di mana jobCode adalah FTE dan karyawan memiliki nama belakang selain James.

Semua data dari tabel jobCode, di mana jobCode adalah FTE dan karyawan memiliki nama belakang selain James.

Correct

Kueri ini akan memilih semua data dari tabel employee di mana jobCode adalah FTE dan nama belakangnya adalah James.

### 4.

Question 4

Anda sedang mengerjakan tabel databaseyang berisi data tentang artis musik. Tabel tersebut diberi nama artist. Anda ingin meninjau semua kolom dalam tabel ini.

Anda menuliskan kueri SQL seperti di bawah ini. Tambahkan klausa FROM yang akan mengambil data dari tabel artist.

1

2

3

SELECT

\* from artist musik

RunReset

Berapa jumlah kolom di tabel artist ?

1 / 1 point

8

5

9

2

Correct

Klausa FROM artist akan mengambil data dari tabel artist . Kueri lengkapnya adalah SELECT \* FROM artist. Klausa FROM menentukan tabel database mana yang akan dipilih datanya. Ada dua kolom di tabel artist.

### 5.

Question 5

Anda sedang mengerjakan tabel databaseyang berisi data tentang album musik. Anda hanya tertarik pada data sebuah album dengan nomor ID 277. ID album tercantum di kolom album\_id dari tabel album.

Anda menuliskan kueri SQL seperti di bawah ini. Tambahkan klausa WHERE yang hanya menampilkan data tentang album dengan nomor ID 277.

1

2

3

4

SELECT

\*

FROM

 album WHERE album\_id = 277

RunReset

Apa nama album dengan nomor ID 277?

1 / 1 point

Beethoven: Piano Sonatas

Vivaldi: The Four Seasons

Mozart: Chamber Music

Bach: Goldberg Variations

Correct

Klausa WHERE genre\_id = 277hanya menampilkan data tentang album dengan nomor ID 277. Kueri lengkapnya adalah SELECT \* FROM album WHERE album\_id = 277. Klausa WHERE menyaring hasil yang memenuhi kondisi tertentu. Klausa WHERE meliputi nama kolom, tanda sama dengan, dan nilai pada kolom yang akan disertakan. Nama album dengan nomor ID 277 adalah Bach: Goldberg Variations.

