**Menganalisis Data untuk Menjawab Pertanyaan**

**VLOOKUP untuk agregasi data**

Agregasi adalah mengumpulkan atau menyatukan banyak potongan yang terpisah menjadi satu. Jadi agregasi data adalah proses pengumpulan data dari berbagai sumber yang digabungkan menjadi satu koleksi yang ringkas. Di analisis data, ringkasan koleksi, ataupun ringkasan, menjelaskan pengidentifikasian data yang Anda butuhkan, kemudian mengumpulkannya di satu tempat. Agregasi data membantu analis data untuk mengidentifikasi tren, membuat perbandingan, dan mendapatkan wawasan di mana di tiap elemen datanya tidak memungkinkan untuk dianalisis secara terpisah. Kueri adalah permintaan informasi dari database. Jadi subkueri, disebut juga inner atau nested query, yaitu kueri yang ada di dalam kueri lainnya.



VLOOKUP adalah singkatan dari vertical lookup. Sederhananya, ini adalah fungsi untuk mencari nilai tertentu di kolom untuk mengembalikan informasi yang sesuai. Sebelumnya, kita telah menggunakan VLOOKUP untuk mengambil nilai di dalam satu sel dan mencari kecocokan di sel lainnya.



**Konsep inti VLOOKUP**

Fungsi dapat digunakan untuk menemukan informasi secara cepat dan melakukan perhitungan berdasarkan nilai tertentu. Dalam bacaan ini, Anda akan mempelajari tentang pentingnya sebuah fungsi, VLOOKUP, atau Vertical Lookup, yang mencari sebuah nilai tertentu di kolom spreadsheet dan menghasilkan sepotong informasi yang sesuai dengan nilai yang ada di baris di mana nilai yang dicari tersebut ditemukan.

**Kapan Anda perlu menggunakan VLOOKUP?**

Dua alasan umum untuk menggunakan VLOOKUP adalah:

* Menyimpan data dalam spreadsheet
* Menggabungkan data dari satu spreadsheet dengan data yang lain

**Sintaks VLOOKUP**

Fungsi VLOOKUP tersedia di Microsoft Excel dan Google Sheets. Anda akan diperkenalkan dengan sintaks umum di Google Sheets. (Anda dapat merujuk ke beberapa referensi di akhir dokumen ini untuk informasi lebih lanjut tentang VLOOKUP di Microsoft Excel.)





**search\_key**

* Nilai yang harus dicari.
* Misalnya, 42, "Cats", atau I24.

**range (Rentang sel)**

* Rentang sel yang perlu dipertimbangkan untuk pencarian.
* Kolom pertama dalam rentang yang dicari dipergunakan untuk menemukan data yang cocok dengan nilai yang ditentukan oleh search\_key.

**index**

* Indeks kolom dari nilai yang dihasilkan, di mana kolom pertama dalam rentang diberi nomor 1.
* Jika indeks tidak berada di antara 1 dan jumlah kolom dalam rentang, keterangan #VALUE! dimunculkan.

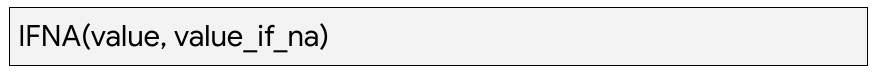
**is\_sorted**

* Menunjukkan apakah kolom yang akan dicari (kolom pertama dalam rentang yang ditentukan) telah diurutkan. TRUE secara default.
* Disarankan untuk mengisi is\_sorted dengan FALSE. Jika diatur ke FALSE, padanan yang benar-benar sama persis dimunculkan. Jika ada beberapa nilai yang cocok, isi sel yang sesuai dengan nilai yang ditemukan pertama kali akan dimunculkan, sebaliknya, #N/A akan dimunculkan jika tidak ada satu nilai pun yang ditemukan.
* Jika is\_sorted diisi TRUE atau diabaikan, padanan yang paling mendekati (kurang dari atau sama dengan search key) dimunculkan. Jika semua nilai di kolom pencarian lebih besar dari search key, #N/A yang dimunculkan.

**Bagaimana jika Anda menemui #N/A?**

Seperti yang telah Anda pelajari, #N/A menunjukkan bahwa nilai yang sesuai tidak dapat diperoleh sebagai hasil VLOOKUP. Kesalahan tersebut tidak berarti menandakan adanya kesalahan pada data, melainkan hanya sebagai sebuah tanda bahwa telah terjadi sebuah kesalahan. Anda dapat menggunakan fungsi IFNA untuk menggantikan kesalahan #N/A dengan sesuatu yang lebih deskriptif, seperti “Does not exist”.





**value**

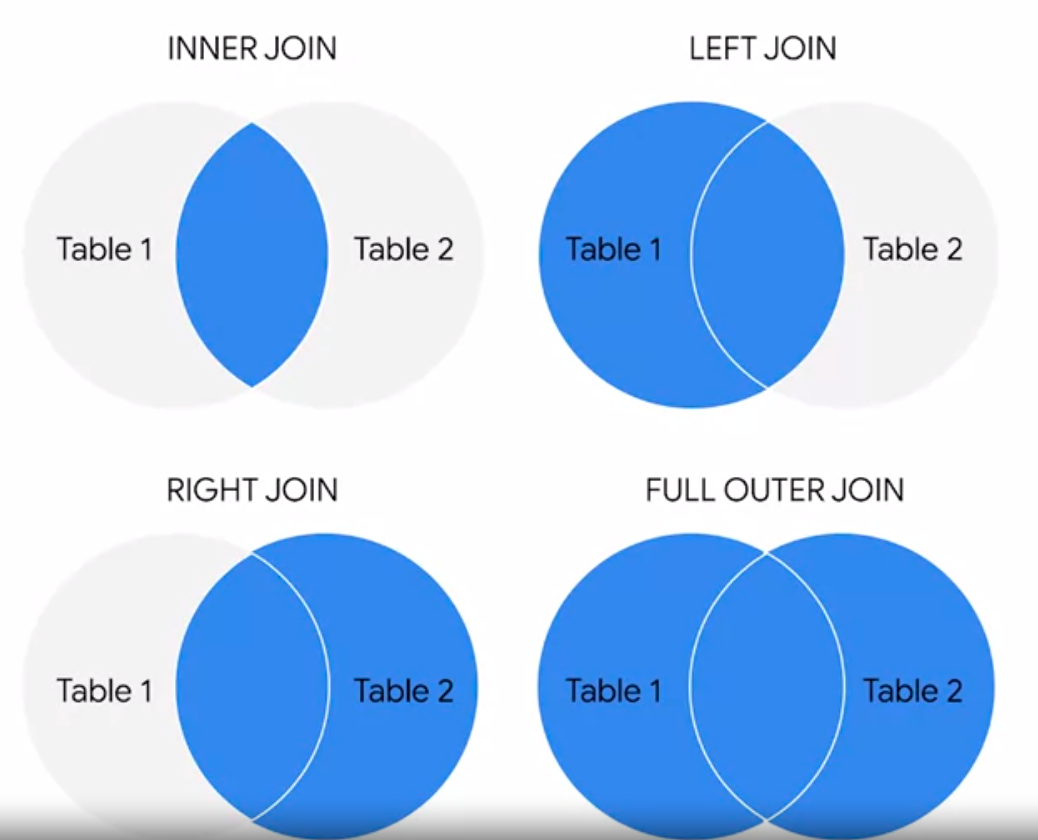
* Nilai yang diperlukan.
* Fungsi tersebut memeriksa nilai sel sesuai dengan value; seperti #N/A.

**value\_if\_na**

* Nilai yang diperlukan.
* Fungsi tersebut menampilkan nilai ini jika nilai sel sesuai dengan nilai pada isian pertama; fungsi ini menghasilkan nilai yang dimaksud ketika nilai sel adalah #N/A.

**Pengingat VLOOKUP yang bermanfaat**

* TRUE menunjukkan padanan yang tidak harus sama persis, FALSE berarti padanan yang benar-benar sama persis dengan search key. Jika data yang digunakan pada search key telah diurutkan, TRUE dapat digunakan.
* Anda menginginkan kolom yang sesuai dengan search key dalam rumus VLOOKUP adalah yang berada di sebelah kiri data. VLOOKUP hanya mencari data dari kolom paling kiri ke kanan setelah kecocokan ditemukan. Dengan kata lain, indeks VLOOKUP hanya menggunakan kolom dari kiri ke kanan saja. Hal ini mengharuskan Anda menentukan kolom yang sesuai sebelum menggunakan VLOOKUP.
* Setelah menentukan data pada rumus VLOOKUP, Anda dapat menyalin dan menempelkan data tersebut hanya sebagai nilai untuk menghilangkan formula-formula sehingga Anda dapat menggunakan kembali data tersebut.



**Pentingnya penggunaan nama alias dalam SQL**

Nama alias (Aliases) digunakan dalam kueri SQL untuk membuat nama sementara dari kolom atau tabel. Nama alias menyederhanakan perujukan tabel dan kolom dalam kueri SQL Anda jika Anda memiliki nama tabel atau kolom yang terlalu panjang atau rumit untuk digunakan dalam kueri. Bayangkan nama sebuah tabel seperti *special\_projects\_customer\_negotiation\_mileages*. Itu akan menyulitkan setiap kali Anda akan menggunakan tabel itu. Dengan sebuah nama alias, Anda bisa membuat sebuah nama singkat bermakna yang dapat digunakan untuk analisis Anda. Dalam hal ini, “*special\_projects\_customer\_negotiation\_mileages*” dapat disederhanakan menjadi “*mileage.”* Daripada harus menulis nama tabel yang panjang, Anda dapat menentukan sebuah nama singkat bermakna yang bisa digunakan.

**Sintaks dasar untuk aliasing**

Aliasing adalah proses penggunaan nama alias. Dalam kueri SQL, nama alias diimplementasikan dengan menggunakan perintah AS. Sintaks dasar untuk perintah AS dapat dilihat dalam kueri untuk aliasing sebuah tabel berikut:



Perhatikan bahwa AS didahului oleh nama tabel dan diikuti oleh nama singkat baru. Hal yang sama dilakukan pada aliasing sebuah kolom:



Dari kedua kasus tersebut, Anda sekarang memiliki sebuah nama baru yang digunakan untuk merujuk ke kolom atau tabel yang diberi nama alias.

**Sintaks alternatif untuk nama alias**

Jika penggunaan AS menimbulkan kesalahan ketika menjalankan sebuah kueri karena basis data SQL yang Anda gunakan tidak mendukung, tinggalkan saja. Pada contoh sebelumnya, sintaks alternatif untuk aliasing sebuah tabel atau kolom adalah:

* FROM table\_name alias\_name
* SELECT column\_name alias\_name

Hal penting yang dapat dipelajari adalah kueri dapat dijalankan dengan atau tanpa menggunakan AS untuk aliasing, tetapi menggunakan AS membuat kueri lebih mudah dibaca. Ini membuat nama alias terlihat lebih jelas.

**Penggunaan aliasing**

Mari kita lihat sebuah contoh kueri SQL yang menggunakan aliasing. Anggap Anda sedang bekerja menggunakan dua tabel: salah satunya berisi data karyawan dan lainnya berisi data departemen. Rumusan FROM untuk nama alias tabel tersebut adalah:

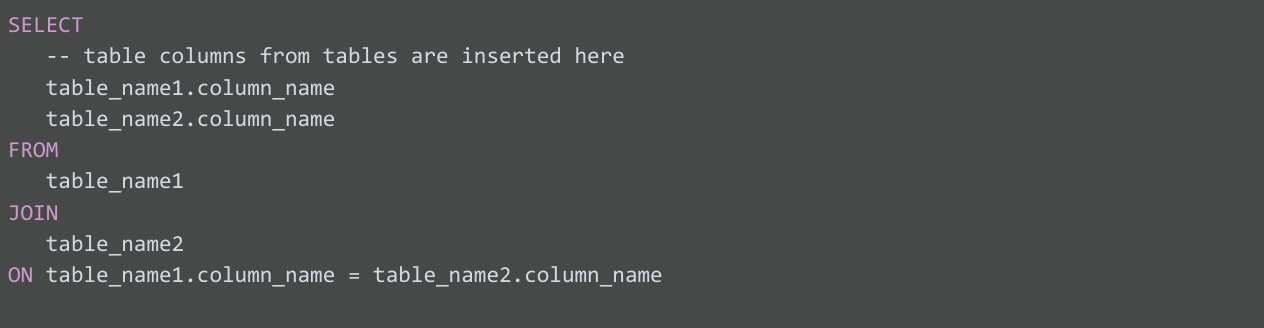


Nama alias ini membuat Anda mengetahui isi tabel yang sebenarnya, tanpa Anda perlu mengetik nama tabel yang panjang. Nama alias sangat berguna untuk mengatasi kueri yang panjang dan rumit. Akan lebih mudah membaca dan mengetik kueri ketika Anda memiliki nama alias yang mencakup data yang ada dalam tabel.

**Penggunaan JOIN secara efektif**

JOIN menggabungkan beberapa tabel menggunakan primary atau foreign key untuk menyelaraskan informasi dari kedua tabel dalam proses penyatuannya. JOIN menggunakan primary dan foreign key untuk mengidentifikasi hubungan dan nilai yang sesuai di seluruh tabel.

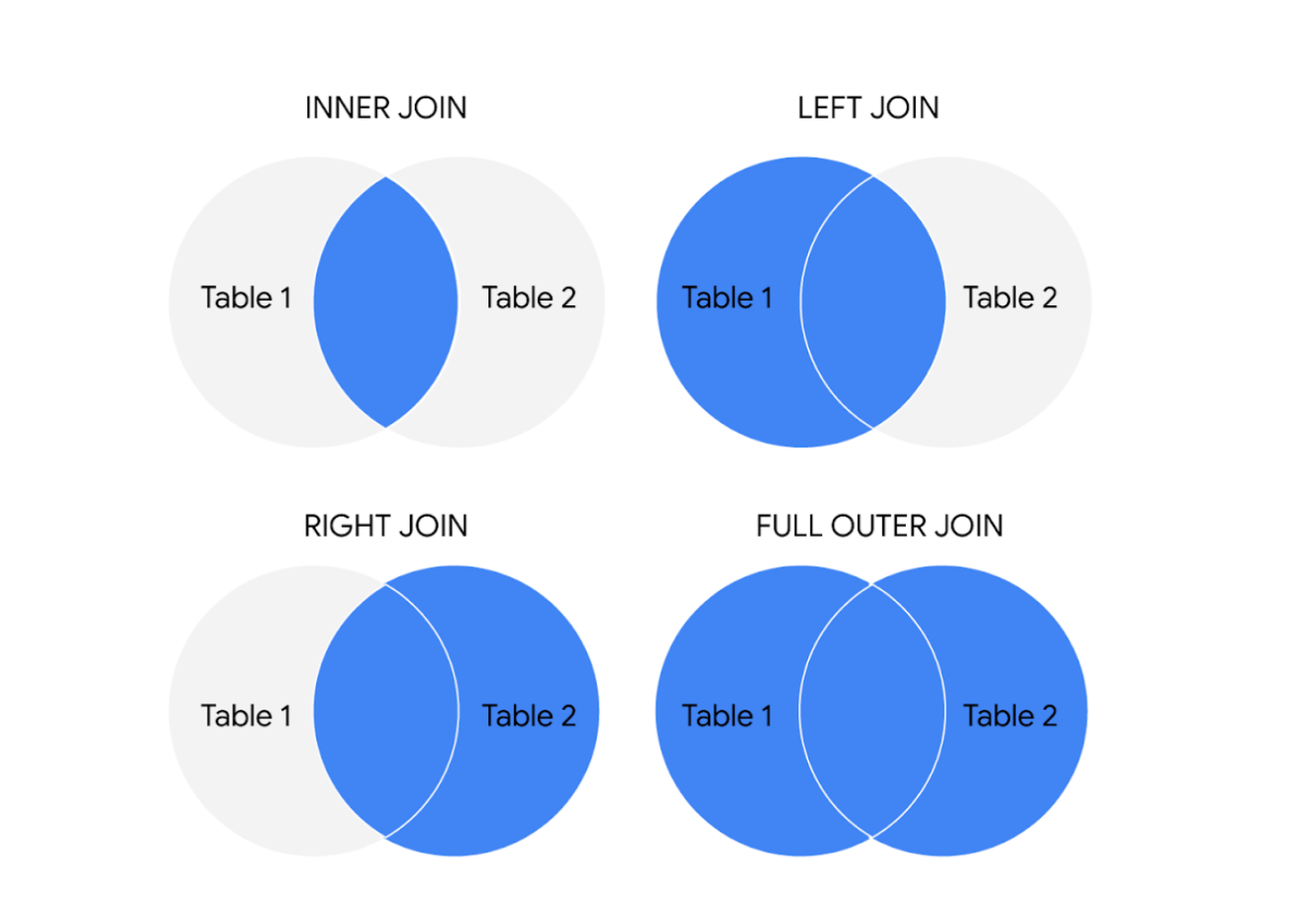
**Sintaks umum JOIN**



Seperti yang Anda lihat pada sintaks, rumusan JOIN adalah bagian dari klausa FROM kueri. JOIN dalam SQL menunjukkan bahwa Anda akan menggabungkan data dari dua tabel. ON dalam SQL menunjukkan bagaimana menyesuaikan informasi yang benar dari beberapa tabel yang akan digabungkan.

**Jenis-jenis JOIN**

Ada empat cara umum untuk menggunakan JOIN dalam kueri SQL: INNER, LEFT, RIGHT, dan FULL OUTER.

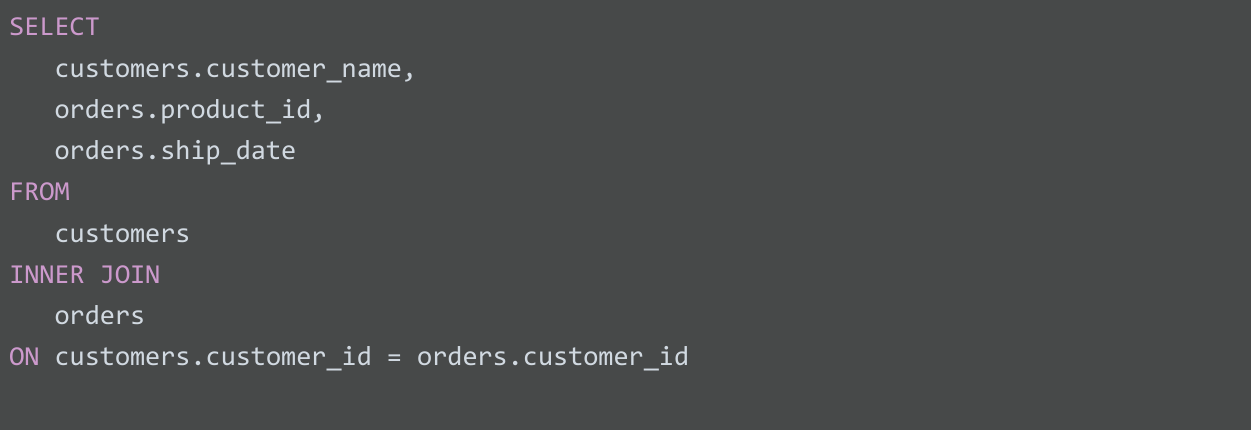


Lingkaran-lingkaran ini mewakili tabel kiri dan kanan, serta irisan di antara keduanya diwarnai biru

Berikut adalah yang dilakukan oleh berbagai kueri JOIN ini.

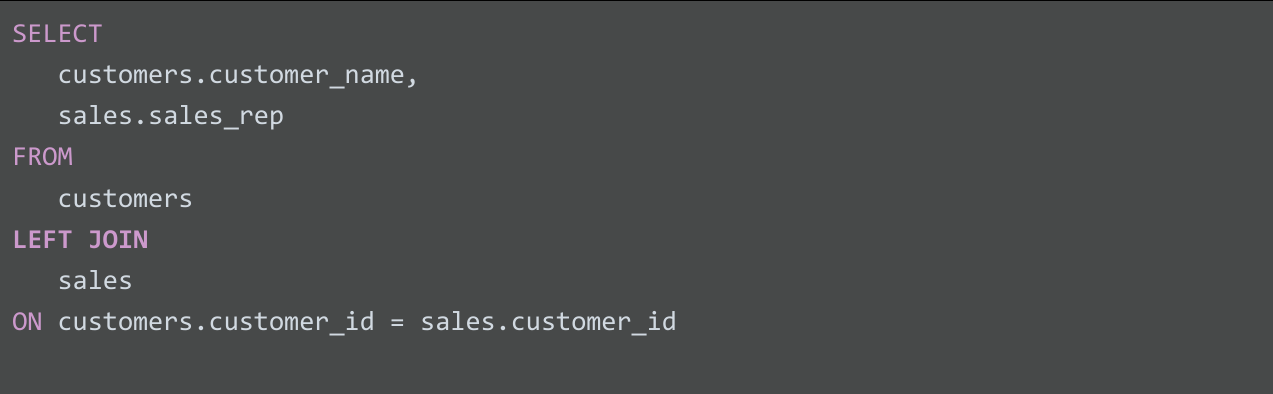
**INNER JOIN**

INNER adalah opsional dalam kueri SQL ini karena merupakan default seperti halnya penggunaan JOIN pada umumnya. Anda dapat menyebutnya sebagai JOIN saja. INNER JOIN menampilkan data yang ada di kedua tabel tersebut. Misalnya, jika Anda menggunakan INNER JOIN pada tabel ‘customers’ dan ‘orders’ dan mencocokkan datanya menggunakan kunci customer\_id, Anda akan menggabungkan data setiap customer\_id yang ada di kedua tabel. Jika sebuah customer\_id ada di tabel customers tetapi tidak ada di tabel orders, data untuk customer\_id tidak digabungkan atau dimunculkan oleh kueri.



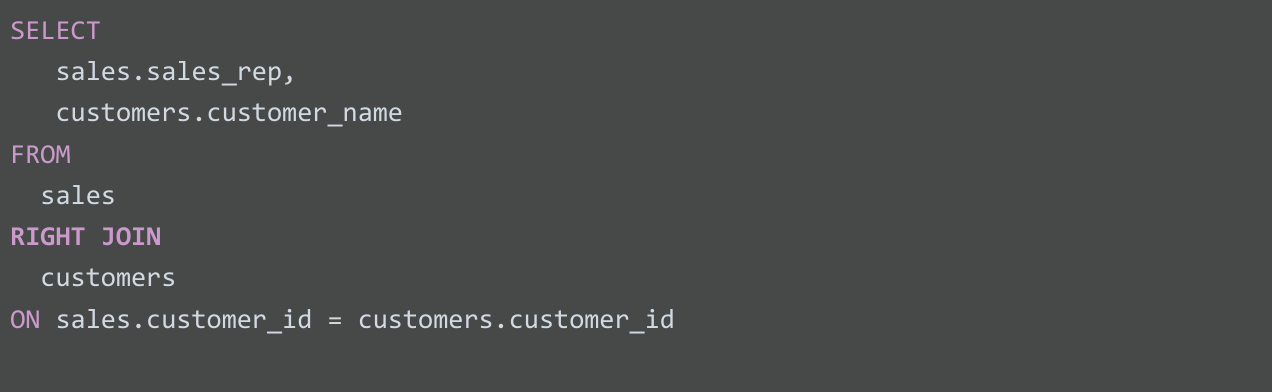
**LEFT JOIN**

LEFT JOIN menampilkan semua data dari tabel sebelah kiri dan hanya data yang sesuai dari tabel sebelah kanan. Gunakan LEFT JOIN setiap kali Anda memerlukan semua data dari tabel pertama dan beberapa nilai dari tabel kedua, jika ada. Misalnya, pada kueri di bawah ini, LEFT JOIN akan menampilkan customer\_name dengan sales\_rep yang sesuai, jika ada. Jika ada seorang pelanggan yang tidak berinteraksi dengan perwakilan sales, maka pelanggan tersebut akan tetap muncul di hasil kueri tetapi dengan nilai NULL untuk sales\_rep.



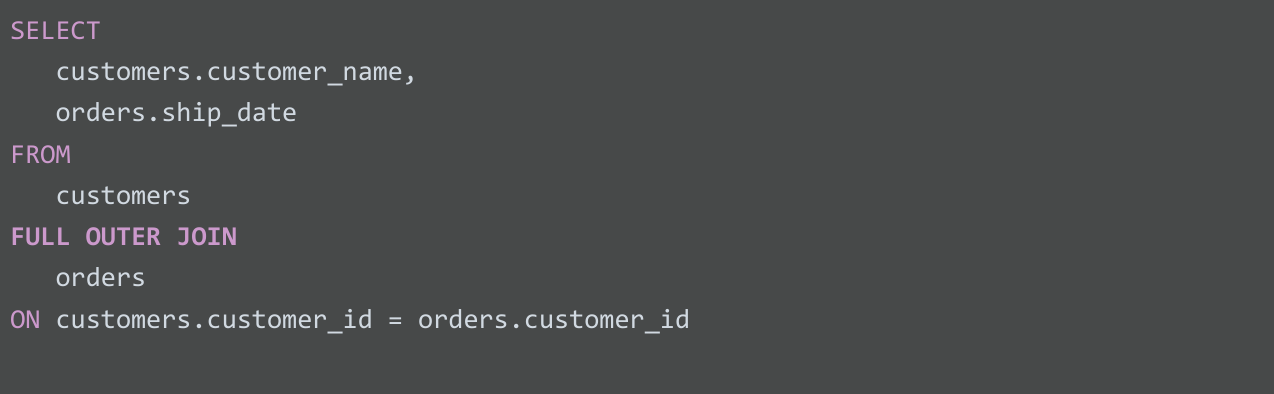
**RIGHT JOIN**

RIGHT JOIN menampilkan semua data dari tabel sebelah kanan dan beberapa data yang sesuai dari tabel sebelah kiri. Secara praktis, RIGHT JOIN jarang digunakan. Kebanyakan orang hanya mengganti tabel dan tetap menggunakan LEFT JOIN. Tetapi dengan menggunakan contoh sebelumnya untuk LEFT JOIN, kueri yang menggunakan RIGHT JOIN akan terlihat seperti berikut:



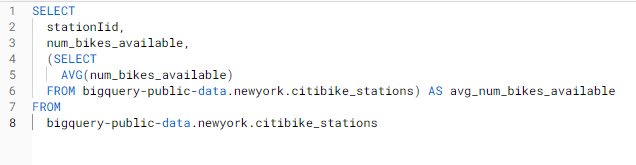
**FULL OUTER JOIN**

Terkadang disebut sebagai FULL JOIN. FULL OUTER JOIN menampilkan semua data dari tabel yang ditentukan. Anda dapat menggabungkan tabel dengan cara ini, tetapi ingat bahwa ini berpotensi menampilkan data yang besar sebagai hasilnya. FULL OUTER JOIN menampilkan semua data dari kedua tabel sekalipun datanya tidak tercantum di salah satu tabel tersebut. Misalnya, pada kueri di bawah ini, Anda akan mendapatkan data semua pelanggan dan tanggal pengiriman produk mereka. Karena menggunakan FULL OUTER JOIN, Anda mungkin mendapatkan data pelanggan dimunculkan tanpa kesesuaian dengan tanggal pengiriman atau sebaliknya. Sebuah nilai NULL dimunculkan jika tidak ada kesesuaian data di kedua tabel.

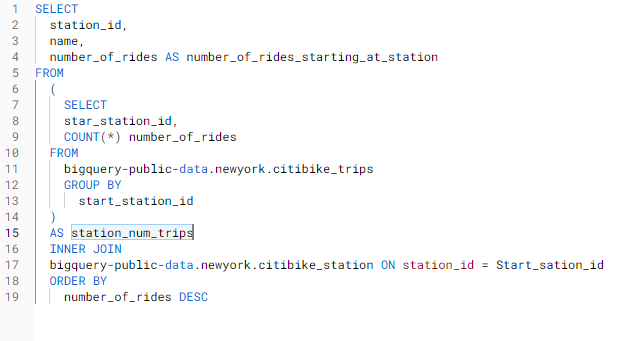


**Sub-Query**

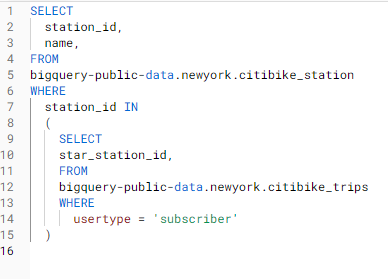
Sub kueri adalah kueri SQL yang bersarang di dalam kueri yang lebih besar. Kueri Anda yang lebih besar dapat memiliki subkueri di dalamnya lalu subkueri tersebut bisa memiliki subkueri, lalu subkueri itu juga dapat memiliki subkueri lainnya juga. Tapi ketika Anda menumpuk mereka secara bersamaan, mereka membentuk satu kueri. Dengan berbagai subkueri, Anda dapat menyatukan potongan logika yang berbeda secara bersamaan. Karena logika dari outer kueri bergantung pada inner kueri, Anda bisa melakukannya dengan satu kueri. Ini berarti semua logika terletak di satu tempat, yang bisa membuat kueri lebih efisien dan lebih mudah dibaca. Pernyataan yang berisi subkueri juga bisa disebut sebagai outer kueri atau outer select. Hal ini menjadikan subkuerinya menjadi inner kueri atau inner select. Inner kueri dijalankan terlebih dahulu sehingga hasilnya dapat digunakan oleh outer kueri.



Contoh umum sub-query



Contoh sub query kompleks





Anda akan menemukan bahwa, dalam klausa SELECT pertama terdapat klausa SELECT lain. Klausul SELECT kedua menandai awal subkueri dalam pernyataan ini. Ada beberapa pilihan cara di mana Anda dapat menggunakan subkueri, dan referensi yang diberikan akan memberikan panduan tambahan saat Anda mempelajarinya. Tapi pertama-tama, mari kita rangkum beberapa aturan subkueri.

Ada beberapa aturan yang harus diikuti subkueri:

* Subkueri harus diapit oleh tanda kurung.
* Subkueri hanya dapat memiliki satu kolom yang ditentukan dalam klausa SELECT. Jika Anda ingin subkueri membandingkan beberapa kolom, maka kolom-kolom tersebut harus dipilih dalam kueri utama.
* Subkueri yang menghasilkan lebih dari satu baris hanya dapat digunakan dengan beberapa operator nilai, seperti operator IN yang memungkinkan Anda menentukan beberapa nilai dalam klausa WHERE.
* Subkueri tidak dapat disarangkan dalam perintah SET. Perintah SET digunakan bersamaan dengan UPDATE untuk menentukan kolom (dan nilai-nilai) mana yang akan diperbarui dalam tabel.

**Agregat Data dengan sub-Query**

HAVING pada dasarnya bisa digunakan untuk menambahkan filter ke kueri selain dari tabelnya ketika Anda bekerja dengan fungsi agregat. Cara tersebut hanya mengembalikan catatan yang memenuhi kondisi spesifik. Demikian pula, CASE yang mengembalikan catatan dengan ketentuan Anda yang memungkinkan Anda menyertakan pernyataan if/then di kueri Anda.

