# [Fundamental SQL Using INNER JOIN and UNION](https://academy.dqlab.id/main/package/practice/244)

## CHAPTER 1

### Pendahuluan

Aku terlonjak kaget ketika merasakan sentuhan tangan di balik pundakku. Ternyata Senja. Dahinya mengernyit, menatapku penuh tanya. “Kamu tampak kebingungan, Aksara. Ada masalah?”

Hening sejenak. Aku menimbang-nimbang apakah akan menyampaikan kegelisahanku pada Senja. Kutengok jam dinding di seberang meja, sudah setengah jam aku berkutat sendiri dalam ketidaktahuan. Lebih baik bertanya.

“Ini, Nja. Aku dapet laporan mengenai transaksi penjualan produk kotak pensil dan flashdisk 32 GB cabang X. Sudah aku coba utak-atik dan lihat, aku nemuin kalau tabel data penjualannya enggak berisi kolom nama produk, jadi aku benar-benar bingung untuk mengidentifikasi tiap kategori produknya,” curhatku akhirnya.

Aku melihat Senja tersenyum dan mendorong bangku yang teronggok di pojok ke sebelahku. “Kalau kamu sudah mahir SQL ini akan mudah. Untuk mendapatkan kolom nama produk, kamu hanya perlu menggabungkan tabel data penjualan dengan tabel lain di database yang berisi detail produk termasuk nama\_produk, mekanisme penggabungan ini disebut JOIN,” jelas Senja sembari mengarahkanku.

“Dengan JOIN, beberapa tabel pada suatu database bisa digabungkan satu sama lain. Kemampuan inilah sebenarnya ciri khas dan kekuatan yang membedakan RDMS dengan sistem database lain. Sudah dapat gambarannya, Aksara?” lanjut Senja.

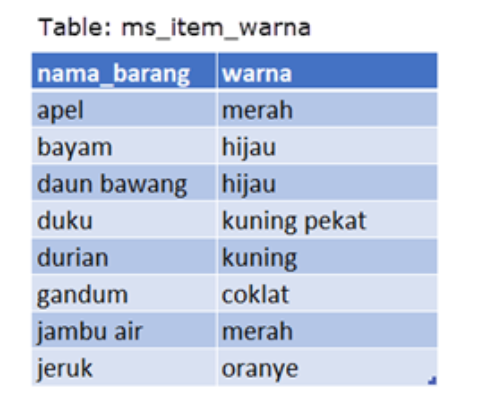
Kepala mengangguk sedikit. Aku perlu mempraktikkan sendiri agar lebih jelas. Seperti tahu keinginan dan maksudku, Senja mengajakku untuk menyelesaikan studi kasus lebih sederhana.

“Oke, mari kita coba praktikkan penggunaannya lebih konkrit dengan dua contoh tabel sederhana, yaitu ms\_item\_warna dan ms\_item\_category, sehingga kita dapat dengan mudah memahami penggunaan JOIN. Jadi, kamu juga lebih mudah menyelesaikan kasus tadi.”

“Siap!” Mataku berbinar.

### Tabel pertama: ms\_item\_warna

“Oke, contoh tabel pertama yang akan kita gunakan sebagai bahan latihan adalah tabel **ms\_item\_warna** seperti terlihat berikut ini,” bimbing Senja sambil memperlihatkan layar laptopnya padaku:



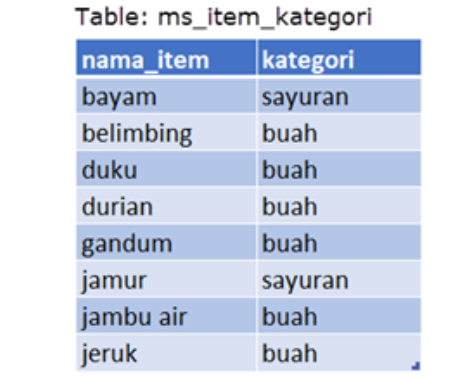
“Tabel ini terdiri dari dua kolom yang berisi informasi nama buah atau sayuran dengan warnanya, dan memiliki delapan baris data,” ujar Senja.

“Lalu, bagaimana langkah selanjutnya?” tanyaku menunggu.

“Selanjutnya, jika permasalahan kita adalah ingin mengetahui informasi tidak hanya mengenai warna tetapi juga jenis/kategori dari setiap nama barang apakah tergolong sayuran atau buah, maka kita tidak dapat memperoleh informasi tersebut jika hanya berdasarkan tablel ms\_item\_warna. Oleh karena itu, kita perlu menghubungkan tabel tersebut dengan tabel lain yang berisi informasi tersebut, yaitu tabel **ms\_item\_kategori**.”

### Tabel kedua: ms\_item\_kategori

“Perhatikan kembali ya, Aksara. Kalau ini contoh tabel kedua, yakni tabel **ms\_item\_kategori**seperti terlihat berikut ini.”



Tabel ini terdiri dari dua kolom yang berisi informasi nama tumbuhan dengan kategorinya, dan memiliki delapan baris data.

“Nah, sekarang kita telah memiliki tabel kedua yaitu tabel ms\_item\_kategori yang memiliki delapan baris data dan berisi informasi mengenai nama tumbuhan dengan kategorinya. Sama seperti permasalahan di tabel sebelumnya, jika kita hanya memiliki tabel ms\_item\_kategori maka kita tidak dapat memperoleh informasi mengenai nama dan warna tumbuhan. Oleh karena itu, kita bisa menggabungkan kedua tabel tersebut menjadi satu tabel baru yang berisi informasi lengkap mengenai nama, warna dan kategori setiap barang/item, “ tambah Senja.

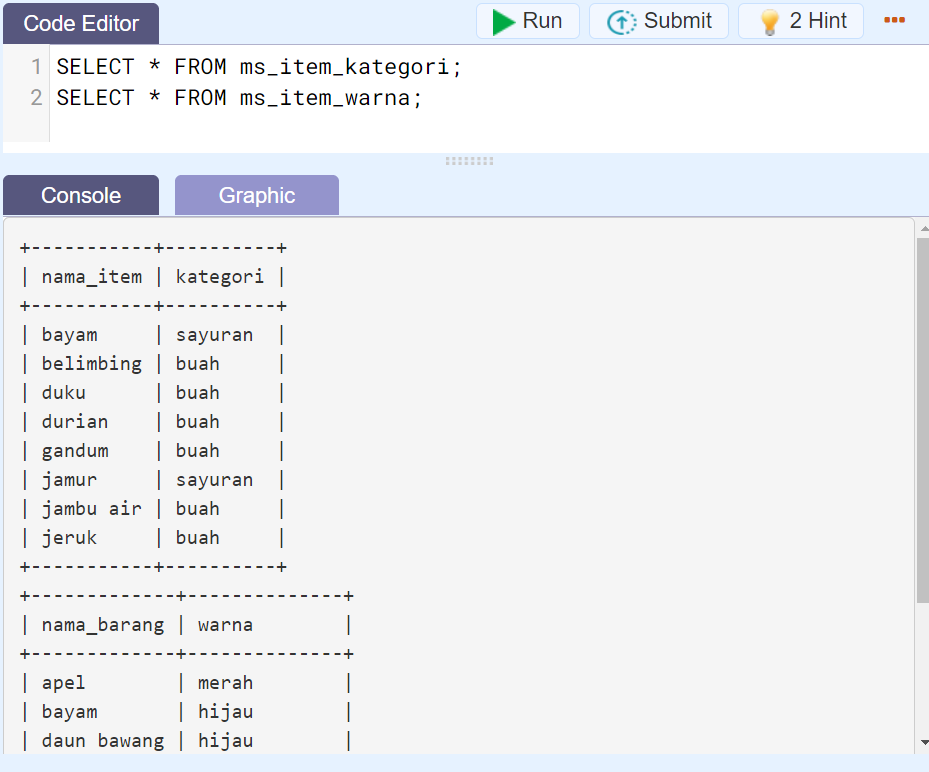
Aku mengangguk paham.

“Kalau sudah cukup mengerti contoh dua tabel tadi, kamu bisa kerjakan tugas praktik dan kuisnya. Saya akan memperhatikan.”

Aku pun kembali mengambil alih laptop.

### Tugas Praktek

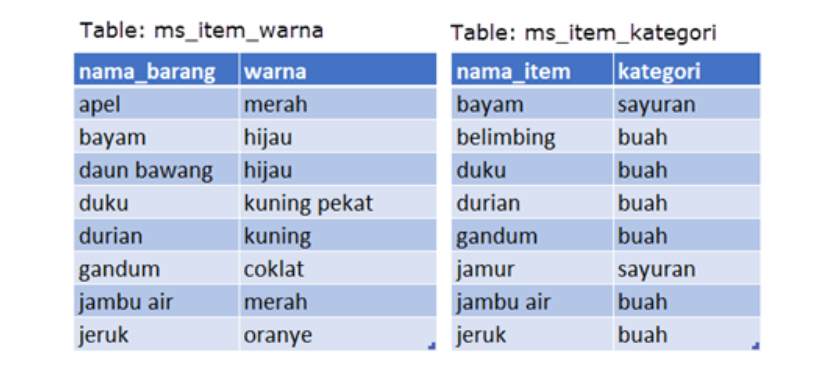
Cobalah ketik query pada code editor untuk melihat keseluruhan isi dari kolom tabel **ms\_item\_kategori** dan **ms\_item\_warna.**



### Key Columns

Penjelasan selanjutnya adalah mengenai key columns dan bagaimana penggunaannya dalam menggabungkan kedua tabel.

Perhatikan kembali tabel **ms\_item\_warna** dan **ms\_item\_kategori** yang telah ditemui pada pembahasan sebelumnya.



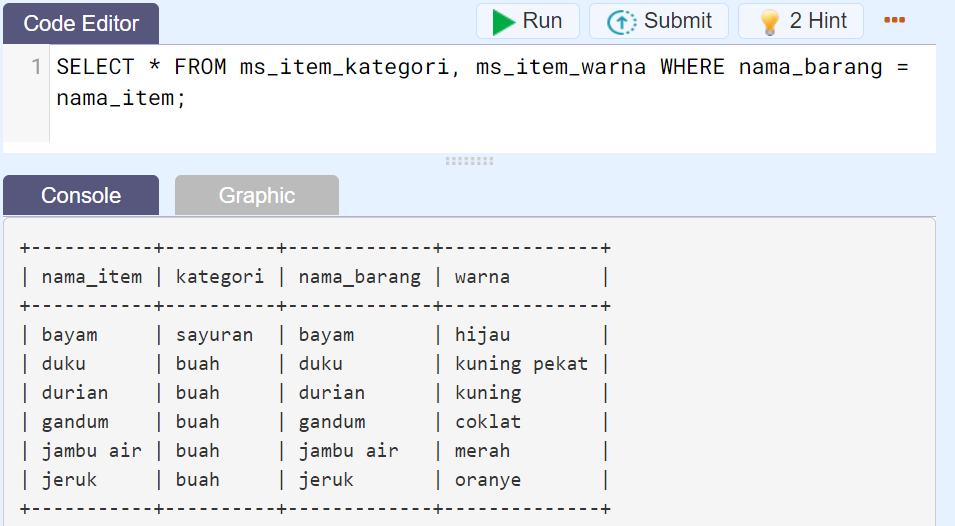
Dari rangkaian pembahasan sebelumnya, dapat disimpulkan bahwa kedua tabel tersebut memiliki hubungan/relasi yaitu pada kolom **nama\_barang** (tabel ms\_item\_warna) dan kolom **nama\_item** (tabel ms\_item\_kategori) karena banyak kecocokan dari isi data.

Kedua kolom ini untuk selanjutnya disebut sebagai **kolom kunci** atau ***key columns***.

### Menggabungkan Tabel dengan Key Columns

Saatnya  mempraktekkan penggabungan tabel **ms\_item\_kategori** dan **ms\_item\_warna** menggunakan *key columns*.

Cobalah tuliskan query berikut di code editor:



Dapat dilihat **hasil dari penggabungan dua tabel** tersebut yaitu berupa **tabel baru dengan empat kolom dan enam baris data**. Perlu diketahui bahwa penggabungan ini bersifat sementara artinya tabel asli di database tidak mengalami perubahan, dan tabel baru hasil penggabungan ini juga tidak serta merta tersimpan di database.

### Penjelasan Query Penggabungan Tabel

Penggabungan dua tabel menjadi satu tabel baru menggunakan query **SELECT** pada subbab sebelumnya dilakukan dengan cara berikut:

menuliskan dua nama tabel yang akan digunakan dengan dipisahkan operator koma (,).

menuliskan pasangan key columns dengan penghubung operator sama dengan (=) di bagian filter atau kondisi.

Berikut adalah gambaran detil syntax penggunaan cara di atas:



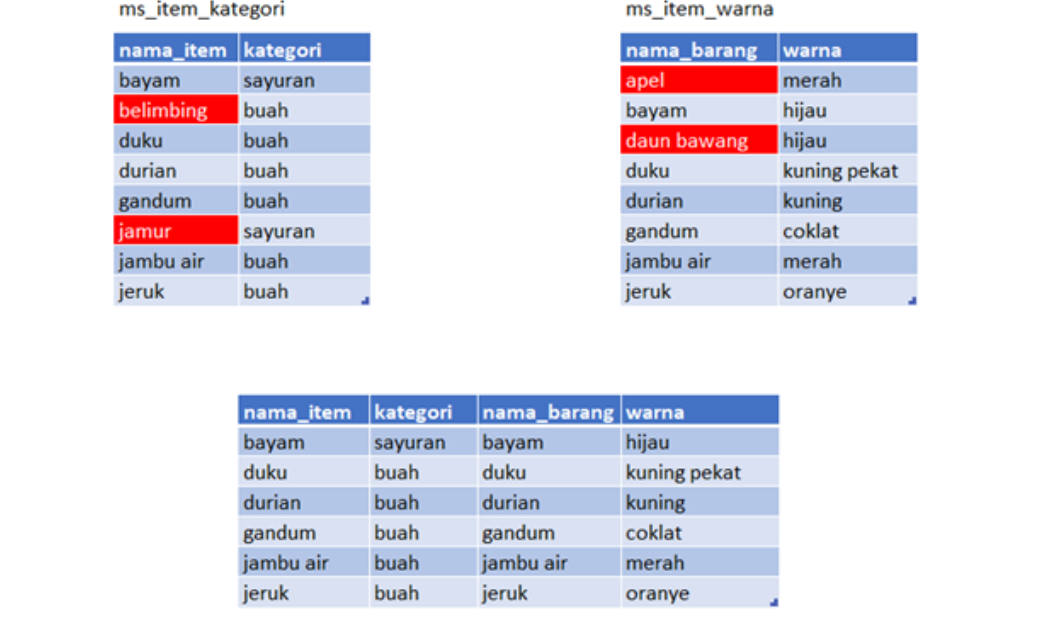
Cara join seperti ini disebut dengan cara join menggunakan **operator koma**.

### Penjelasan Hasil Penggabungan Tabel

Sejauh ini aku tidak menemukan masalah yang signifikan, tapi ketika berusaha menggabungkan tabel aku masih kesulitan. Untungnya Senja masih di sampingku.

“Nja, aku mau nanya. Jika aku amati, penggabungan tabel menghasilkan enam baris data. Sedangkan di kedua tabel asal masing-masing memiliki delapan baris data. Lalu, kenapa bisa berkurang dua baris data?” tanyaku penasaran.

“Untuk menjawab pertanyaan tersebut, cobalah perhatikan gambar berikut,” jawab Senja lugas langsung memberi penjelasan lengkap bersama visualnya

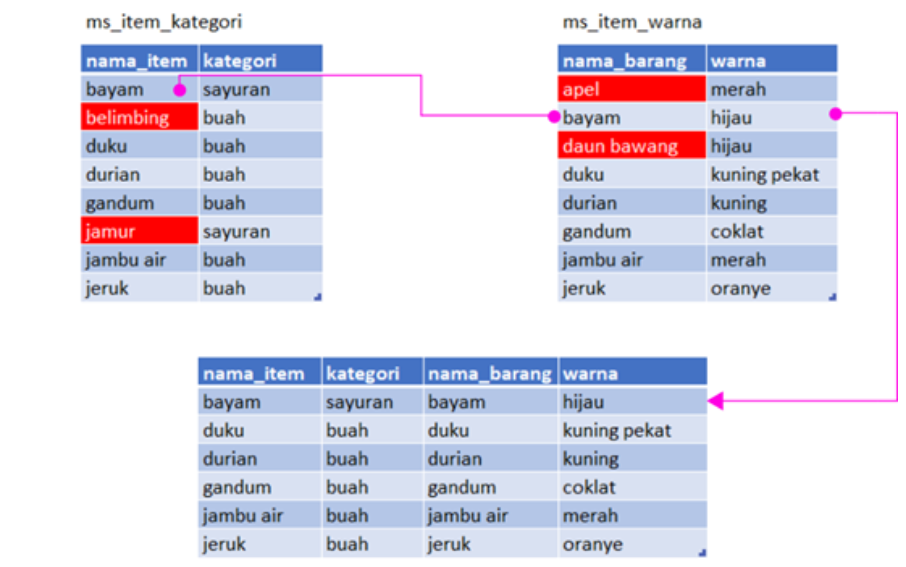


Dua tabel di bagian atas yaitu tabel **ms\_item\_kategori** dan tabel **ms\_item\_warna** merupakan tabel sumber data, dan bagian bawah adalah tabel hasil penggabungan dari kedua tabel tersebut.

Dari tabel hasil penggabungan terlihat bahwa baris data yang muncul hanyalah baris data yang isi datanya terdapat di kedua *key columns,*dan data yang isinya cocok untuk kedua tabel jumlahnya hanya enam dari total delapan baris data yang ada. Sedangkan baris data yang berisi belimbing, jamur, apel dan daun bawang, tidak terdapat di kedua tabel, sehingga baris data ini akan di-*exclude* dan tidak akan muncul di tabel hasil penggabungan.

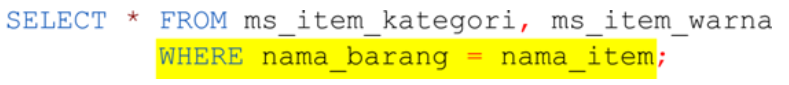
### Penjelasan Proses Penggabungan Tabel

Mari diperhatikan kembali gambar di bawah ini untuk penjelasan proses penggabungan.



Panah berwarna magenta menunjukkan proses pencocokan dan penggabungan data. Sebagai contoh: Baris data pertama dengan isi “**bayam**” pada key column **nama\_item**(tabel **ms\_item\_kategori)**, akan mencari isi yang sama di key column **nama\_barang** (tabel **ms\_item\_warna**), dan ditemukan pada baris kedua kemudian dihubungkan menjadi satu baris di tabel baru hasil penggabungan.

Pencocokan dengan pencarian ini berdasarkan bagian kondisi (*conditional clause*) pada query yang ditandai dengan warna kuning berikut.



Setelah seluruh data ditemukan dengan kondisi ini, dan jika tidak menentukan spesifik kolom maka seluruh kolom data dari kedua tabel akan dimunculkan dengan isi/baris data yang sama dan terdapat di kedua *key column* digabungkan menjadi satu baris pada tabel baru.

Isi data yang ditandai dengan warna merah adalah data yang tidak memiliki pasangan sehingga tidak bisa digabungkan, dan dengan demikian tidak memiliki data gabungan pada hasil akhir.

Cara pencocokan dan penggabungan disebut dengan INNER JOIN - dimana isi dari key column kedua tabel harus cocok satu sama lain baru dapat digabungkan.

Penjelasan Senja selalu lengkap seperti biasa. Aku mengintip buku catatan dari sudut mataku, tak terasa sudah sekitar tiga halaman habis oleh coretan tanganku. Malam ini harus kurapikan nih!

“Satu lagi, Aksara. Saya perhatikan kamu suka sekali mencatat, jangan kelewat soal ini: jika kamu biasa bekerja dengan Excel, proses ini hampir menyerupai penggunaan fungsi **vlookup**. Jadi harusnya sudah terbiasa.”

Aku terkekeh, “Hehehe, iya. Biar rapi. Biasanya aku taruh di buku diari juga.”

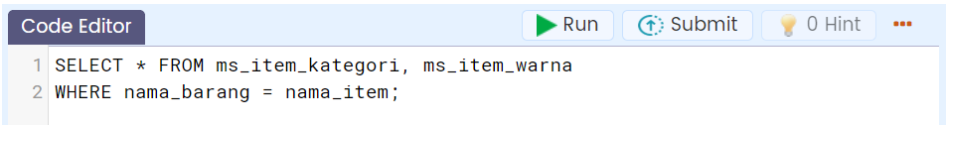
“Metode belajarmu unik juga, Aksara. Kuisnya jangan lupa dikerjakan.”

Aku mengangguk sambil mengangkat jari membentuk kata ‘peace’. Setelah kutelusuri lagi lebih dalam, tak hanya ada kuis saja tapi juga banyak sekali materi-materi baru lainnya! Mumpung Senja masih di sini, aku harus belajar ‘ngebut’ agar bisa bertanya kalau ada yang bingung.

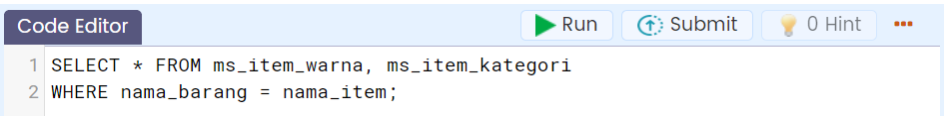
### Bagaimana jika urutan Tabel diubah?

Jika akan mengubah urutan tabel di bagian FROM pada query inner join, maka hanya urutan kolom saja yang berubah tetapi isi data dan jumlah data tidak berubah.

Mari lihat contohnya dengan mengganti query yang sebelumnya, yaitu:



Menjadi query seperti ini

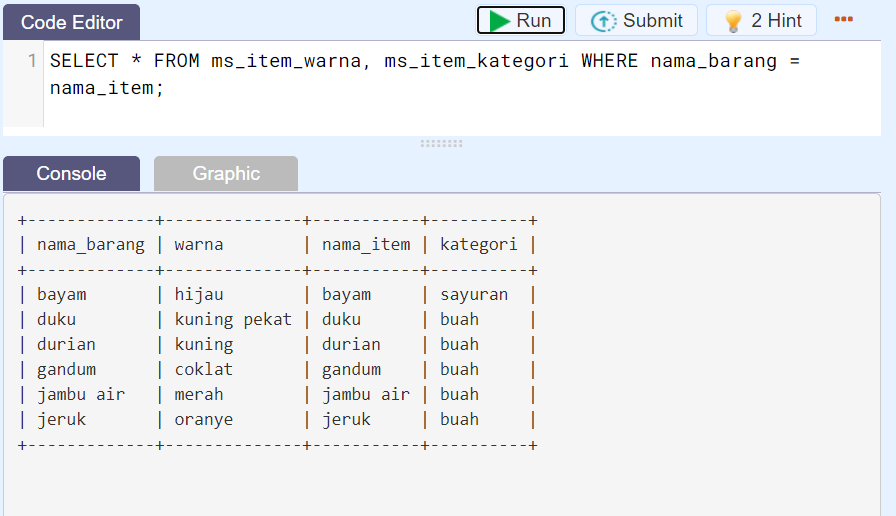


Ketikkan query yang terakhir ini pada code editor dan jalankan dengan menekan tombol . Jika berhasil maka akan menampilkan hasil berikut.



Terlihat jumlah data yang dihasilkan tetap 6 baris data, namun dengan **urutan kolom yang berbeda**.

Dimana dua kolom pertama adalah dari tabel **ms\_item\_warna,** dan dua kolom berikutnya dari tabel **ms\_item\_barang**. Hal ini sesuai dengan urutan nama tabel yang diketikkan setelah FROM.

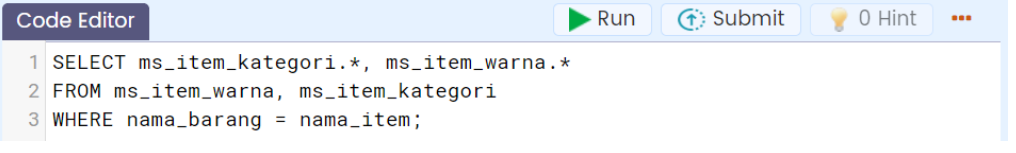


### Menggunakan Prefix Nama Tabel

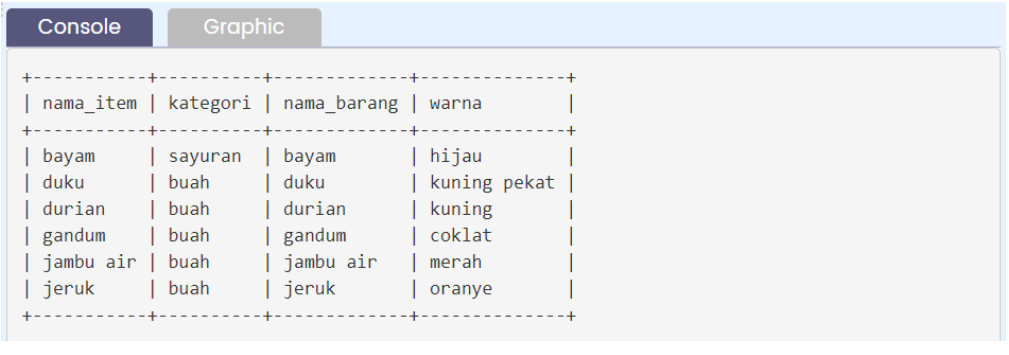
Seperti yang dijelaskan sebelumnya, jika akan menggunakan *wildcard* (\*), dan tidak menentukan spesifik nama kolom yang akan dimunculkan di bagian SELECT, maka secara default urutan kolom dimulai dengan kolom dari tabel yang dinyatakan pertama di bagian FROM. Akan tetapi, bisa juga memanfaatkan wildcard dengan menambahkan prefix nama tabel, dimana dengan merinci prefix nama tabel ini, dimungkinkan untuk menentukan urutan kolom dari tabel mana yang muncul duluan.

Jika lupa dapat mengacu kembali ke **Modul: Fundamental SQL Using SELECT Statement, Chapter: Prefix dan Alias.**

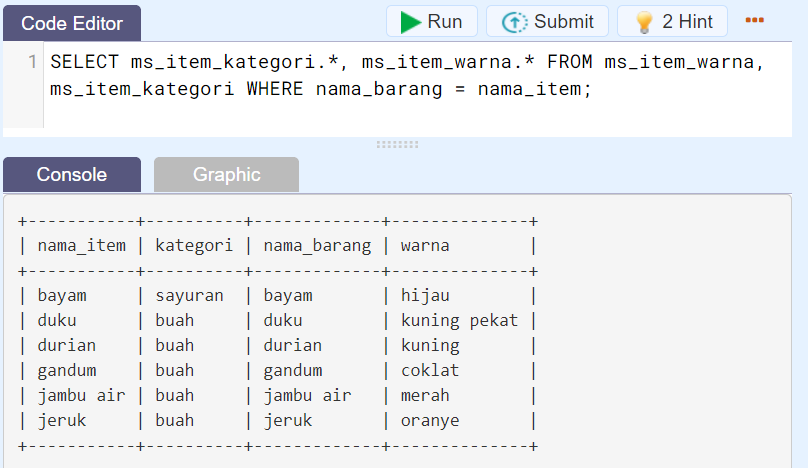
Sebagai contoh, jika akan menggabungkan kedua tabel, dengan menyatakan tabel **ms\_item\_warna** terlebih dahulu di bagian FROM, tetapi yang ingin dimunculkan kolomnya di awal hasil adalah dari yang **ms\_item\_kategori,** maka querynya akan menjadi sebagai berikut.



Jika query yang diketikkan sesuai dengan yang ditunjukkan dan berhasil dijalankan maka akan diperoleh hasil berikut ini:



Terlihat urutan kolom yang muncul adalah dari tabel **ms\_item\_kategori** terlebih dahulu.



### Penggabungan Tanpa Kondisi

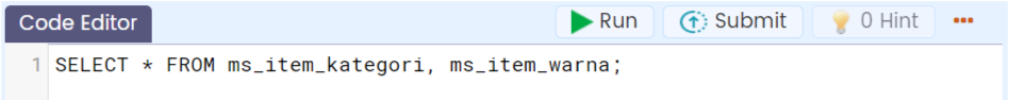
Sesuai dugaanku. Aku kembali menemukan hal yang membingungkan. Dan, tampaknya hari ini Senja memang sudah mendedikasikan waktunya untuk mengajariku karena (beruntungnya!) Senja masih tepat di sebelahku.

“Nja, hehehe. Sorry mau ganggu lagi. Bagaimana kalau aku ingin menggabungkan tabel tanpa ada kondisi? Apakah tetap bisa memakai metode ini?”

Senja memicingkan matanya pada layar laptop yang kutunjuk sebelum menjawab. “Pertanyaan bagus. Penjelasan dan praktek yang kita dilakukan pada materi sebelumnya adalah penggabungan dua tabel dengan menggunakan kondisi, yaitu terdapat data yang sama pada *key kolom* dari kedua tabel. Akan tetapi, memang benar, dalam beberapa case di real problem, sering kali terdapat permasalahan tertentu dimana kita ingin menggabungkan tabel tanpa ada kondisi. Proses penggabungan ini juga dapat dilakukan dengan metode koma dan tanpa menggunakan kondisi relasi antar kolom.”

Aku mencoba mencerna penjelasan Senja. Sebelum aku sempat memahami semuanya. Senja sudah mengajakku untuk melihat contoh.

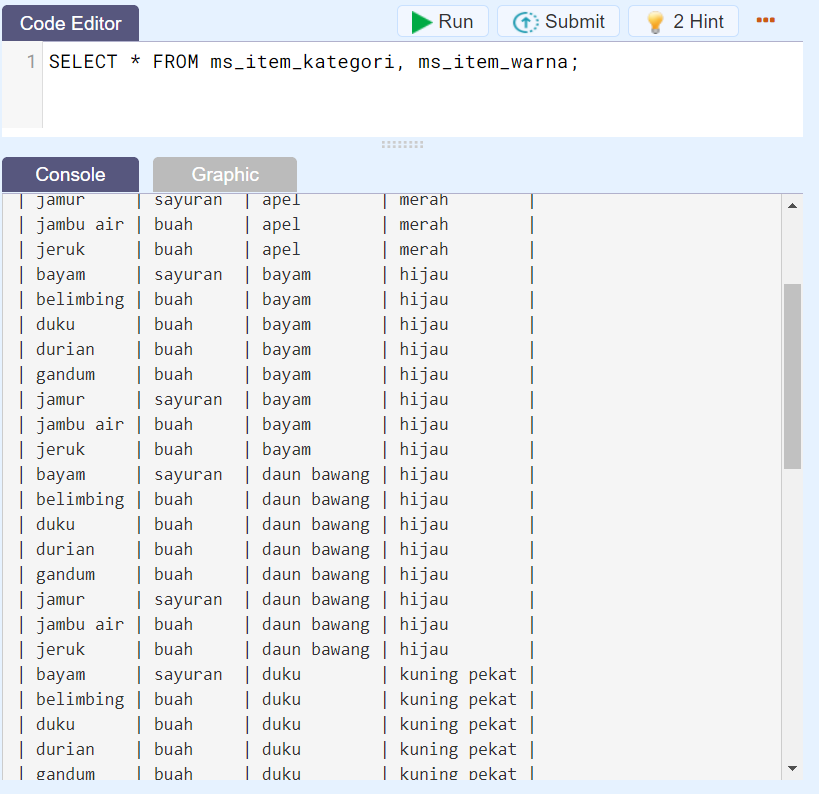
“Berikut adalah contoh query dari penggabungan seperti itu.”



“Terlihat pada query kita hanya menyertakan nama dua tabel yang akan diambil datanya, tapi tidak ada informasi kondisi bagaimana kedua tabel tersebut berelasi satu dengan yang lainnya melalui key column. Lalu apa hasilnya?” tanya Senja retoris.

“Kamu hanya butuh untuk mengetik  query ini pada code editor dan jalankan, jika berhasil dengan baik maka akan muncul hasil dengan enam puluh empat baris data seperti berikut.”

Aku memperhatikan dengan saksama. Jangan sampai ada yang terlewat!



Aku menggumam mengerti. Terlihat banyak sekali hasil yang keluar, ini dikarenakan setiap baris data pada kedua tabel akan dihubungkan satu sama lain - tanpa ada hubungan.

Jumlah enam puluh empat baris data ini adalah hasil perkalian dari jumlah data dari kedua tabel, dimana masing-masing memiliki delapan baris data. Cara menggabungkan kedua tabelseperti ini disebut dengan mekanisme **cross join**.

Aku sudah sampai di penghujung materi dan sejauh ini cukup paham! Hanya butuh satu kuis lagi untuk menuntaskannya. Belajar memang jadi lebih mudah kalau ada Senja, eh maksudku mentor, hehehe.

### Kesimpulan

Bahasa SQL diciptakan untuk sistem database relasional atau RDBMS. Dan aspek **relasional** inilah yang menjadi fitur paling penting dan unggul, dimana beberapa tabel dapat digabungkan menjadi satu sumber data baru.

Cara menggabungkan antar tabel disebut sebagai mekanisme join. Pada bab ini telah dicontohkan bagaimana dua tipe join, yaitu **inner join** dan **cross join** diterapkan dengan dua contoh tabel sederhana.

**Inner join** terjadi jika kedua tabel digabungkan melalui kolom kunci atau key column. Syarat penggabungan adalah dimana Isi data dari key column tabel yang satu harus dapat dicocokkan dengan isi data dari key column tabel yang lain.

Sedangkan **cross join** terjadi dari penggabungan tabel tanpa kondisi, dan menghasilkan seluruh penggabungan data seperti proses perkalian.

Untuk bab ini, inner join maupun cross join dilakukan dengan cara menggunakan operator koma dan pengkondisian where. Pada bab selanjutnya, akan terlihat penggunaan inner join dengan menggunakan keyword **INNER JOIN** … **ON**.

## CHAPTER 2

### Pendahuluan

Pagi ini ada yang membuatku menyunggingkan senyum: menemukan laptop dan beberapa berkas kerja Senja sudah rapi di sebelah mejaku. Ini artinya Senja akan mendampingiku lagi hari ini!

“Eits, jangan girang dulu! Karena bakal ada materi baru yang enggak kalah ribet dibanding kasus kamu kemarin lho,” bisik Senja dari belakang pundakku. Aku sontak membalikkan badan karena terkejut oleh kemunculan Senja yang tiba-tiba.

“Senja! Bikin kaget saja.”

“Sorry, hahaha. Soalnya kamu senyum-senyum sendiri begitu. Oh ya, kemarin kamu sudah mempelajari mengenai konsep dan penggunaan inner join menggunakan operator koma dengan klausa **FROM…WHERE…** pada perintah **SELECT** untuk menggabungkan dua tabel menjadi satu sumber data baru. Hari ini kita naik level ya, sudah siap?”

“Pastinya,” tantangku. Senja hanya tersneyum melihat kepercayaan diriku, padahal lubuk hati terdalamku tegang juga rasanya!

“Selanjutnya saya akan menjelaskan bagaimana menggabungkan dua tabel dengan metode INNER JOIN lain, yaitu dengan keyword INNER JOIN, yang memberikan hasil yang sama, tetapi dengan konstruksi atau syntax yang sedikit berbeda. Konstruksi baru ini akan memberi manfaat lebih ketika pada suatu query melibatkan beberapa join sehingga penggunaan operator koma menjadi sangat terbatas. Sampai sini sudah cukup pusing?”

Lagi-lagi Senja menggodaku. Aku menggeleng mantap, walau sebenarnya ada beberapa hal yang masih asing di telingaku. Tapi, kuyakin pasti akan paham akhirnya. Optimis itu harus!

“Untuk penjelasan mengenai materi JOIN dengan penggunaan keyword INNER JOIN, kita akan kembali menggunakan dua contoh tabel sederhana, yaitu ms\_item\_warna dan ms\_item\_category, sehingga kita dapat dengan mudah memahami penggunaan keyword **INNER JOIN** untuk menggabungkan dua tabel tersebut. Here we go, Aksara.”

### Syntax INNER JOIN

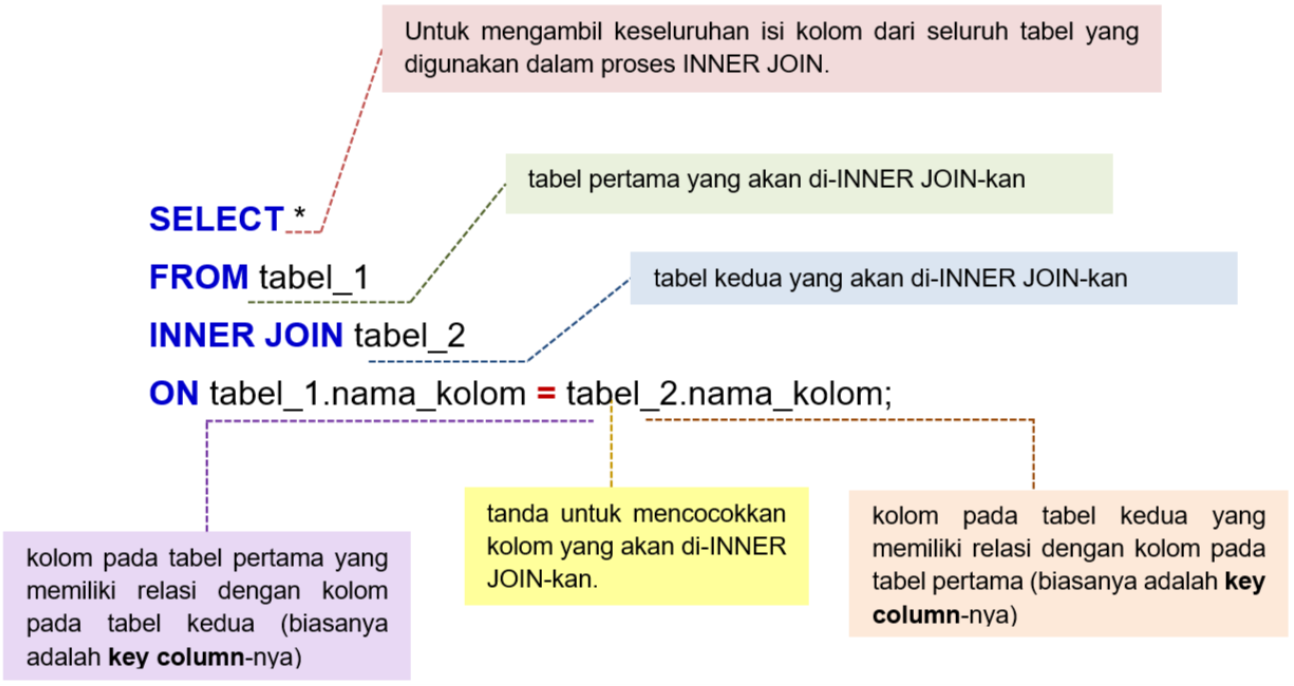
Aku menerima modul baru dari Senja. Saat kubuka, bab baru yang menyapaku adalah “Syntax INNER JOIN”.

“Sudah tahu kan bahwa tujuan JOIN adalah menggabungkan dua atau lebih tabel yang berbeda di dalam database. Nah, JOIN ini hanya bisa dilakukan apabila tabel - tabel tersebut memiliki key kolom yang sama. Di SQL, terdapat beberapa tipe JOIN yaitu : INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, dan SELF JOIN. Setiap tipe join memiliki tujuan dan hasil query yang berbeda. Pada metode INNER JOIN, baris dari kedua atau lebih tabel akan dibandingkan untuk mengecek baris - baris mana saja yang cocok satu sama lain berdasarkan kondisi JOIN yang ditentukan,” jelas Senja fasih.

“Memang JOIN itu sendiri ada berapa tipe sih, Nja?”“Untuk materi ini, kita akan ke tipe pertama, yaitu **INNER JOIN**. Namun demikian, pembahasan akan makin mendalam dengan memberikan konsep pengantar terlebih dahulu agar memudahkan dalam memahami tipe-tipe JOIN lainnya.”

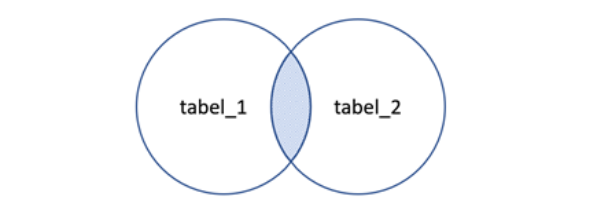
“Oh oke, Nja.”

Syntax INNER JOIN:



Penggunaan prefix pada tabel\_1.nama\_kolom dan tabel\_2.nama\_kolom ditujukan agar terdapat kejelasan (*clarity*) kolom pada tabel mana yang dijadikan acuan dalam proses INNER JOIN. Tentunya, penggunaan prefix ini ini sangat bermanfaat jika ingin menggabungkan beberapa tabel. Untuk penjelasan lebih detail, coba ingat kembali penggunaan prefix dan alias, yang telah dipelajari pada modul "Fundamental SQL with SELECT statement chapter Prefix dan Alias".

Jika masing-masing tabel tersebut dianalogikan sebagai dua himpunan maka proses INNER JOIN ekivalen dengan INTERSECTION (IRISAN) antara dua himpunan.



Proses INNER JOIN memiliki konsep penggabungan untuk records pada kedua tabel yang bernilai sama.

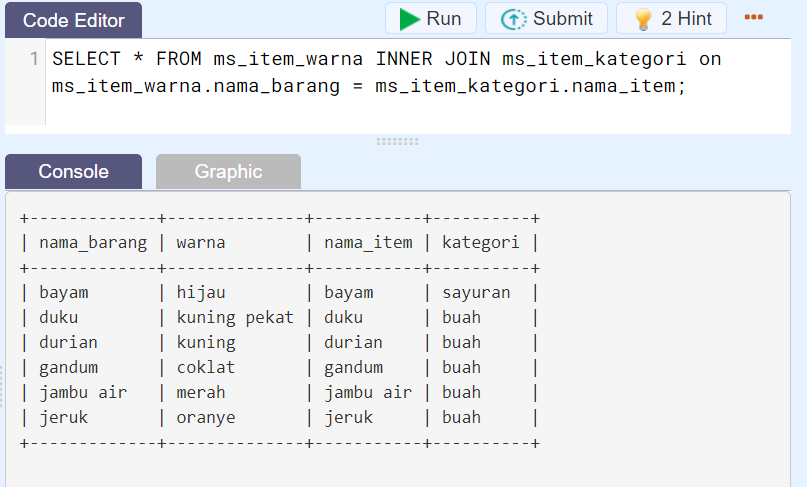
### Tugas Praktek: Menggunakan INNER JOIN (1/3)

Sejauh ini aku belum memiliki pertanyaan berarti dan semoga saja tidak ada. Dalam hati kuberdoa kelancaran pembelajaran ini juga sampai ke kuis dan tugasnya, hehehe.

“Kita lanjut penjelasannya ya, Aksara,” ujar Senja menarikku kembali ke materi.

“Nah, sebelumnya kan kita sudah menerapkan penggunaan … **WHERE** …; dari perintah **SELECT** … dan operator koma, untuk penggabungan tabel ms\_item\_warna dan ms\_item\_kategori. Selanjutnya, ini kita akan menggunakan klausa **INNER JOIN** … **ON** …; untuk menggabungkan kedua ms\_item\_warna dan ms\_item\_kategori berdasarkan sintaks INNER JOIN yang telah dijelaskan sebelumnya.”

Silakan ketikkan query berikut pada code editor



Hasil yang diperoleh dengan penggunaan

**SELECT** … **FROM** … **INNER** **JOIN** … **ON** …; adalah sama dengan penerapan

**SELECT** … **FROM** … **WHERE** …;.

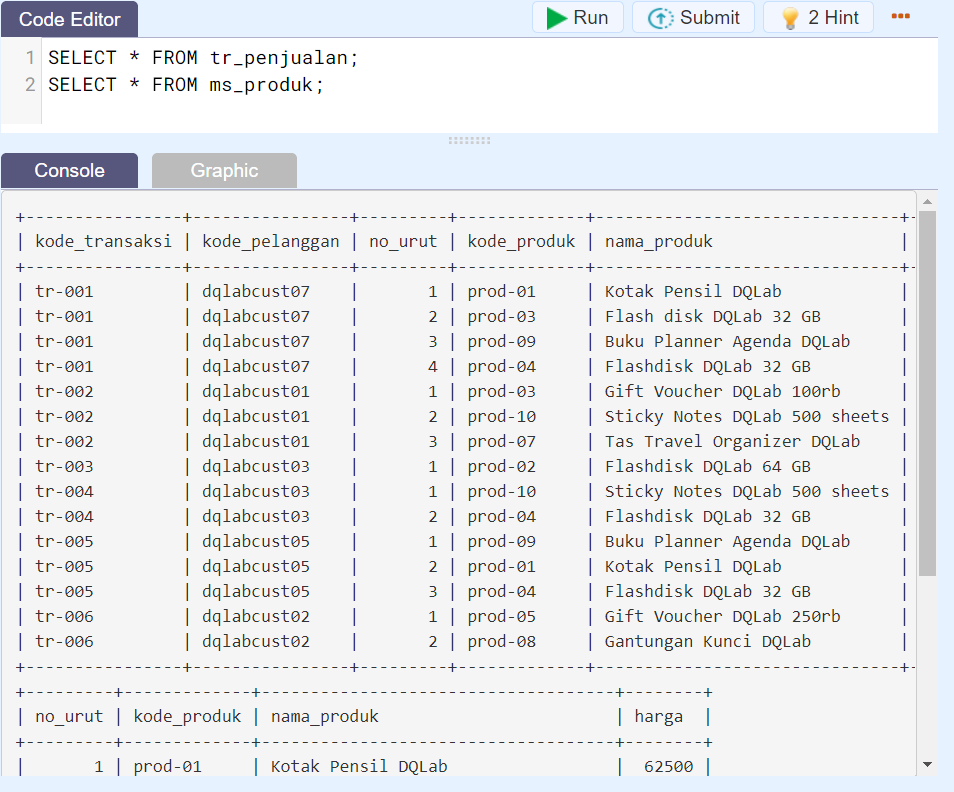
### Tabel tr\_penjualan dan tabel ms\_produk

Tak kusangka penggunaan INNER JOIN ini banyak sekali langkah per langkahnya. Untungnya aku sudah diberi modul oleh Senja, kalau tidak, buku catatanku bisa habis diisi oleh materi ini saja.

“Setelah kita menggunakan tabel ms\_item\_warna dan ms\_item\_kategori, sekarang mari kita gunakan tabel **tr\_penjualan** dan tabel **ms\_produk** yang ada di-database, sehingga kita bisa lebih paham mengenai penerapan konsep JOIN di real case,” pinta Senja.

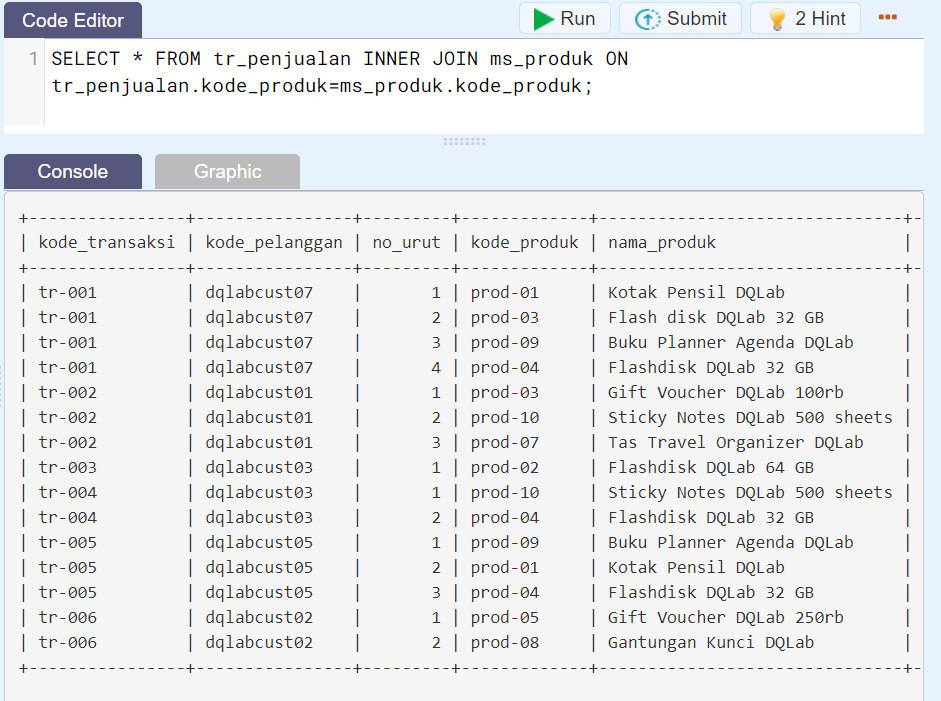
“Oke, Nja.”

”Mari kita lihat isi dari tabel tr\_penjualan. Silakan ketikkan query berikut pada code editor.”



### Tugas Praktek: Menggunakan INNER JOIN (2/3)

“Oke, Aksara. Silakan dipraktikkan pada code editor untuk menggabungkan tabel **tr\_penjualan** dan **ms\_produk** dan menampilkan seluruh kolom dari kedua tabel.”

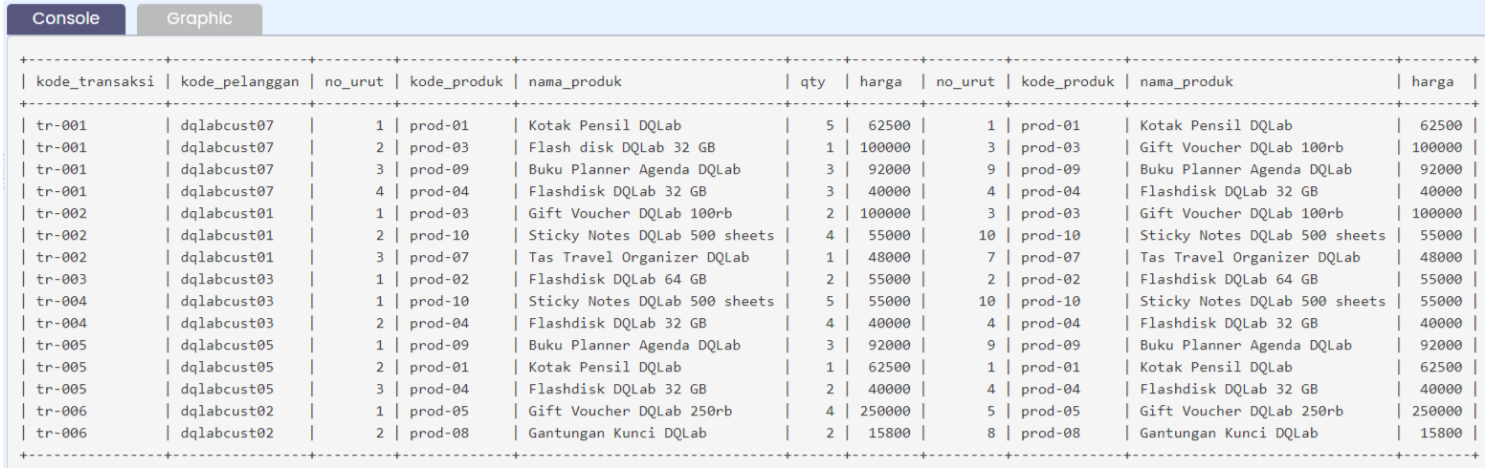


### Memahami Proses dan Hasil Query dengan INNER JOIN

“Lanjut ya, Aksara. Kita ngejar supaya materi ini bisa kamu kuasai hari ini juga,” ujar Senja.

Aku pun kembali serius menekuri layar laptop dan isi modul.

“Perhatikan tabel hasil query pada tugas praktek sebelumnya, apakah perbedaan antara tabel sebelum penggabungan dengan INNER JOIN dan setelah penggabungan?”



“Sekarang seluruh kolom dari kedua tabel berada di satu tabel dan berisi seluruh kolom - kolom dari kedua tabel. Jadi, Aku juga dapat dengan mudah mengidentifikasi nama produk dari setiap kode produk yang dibeli oleh customer.”

“Iya, benar sekali, Aksara. Sekarang perhatikan lebih lanjut dan bandingkan antara kolom **kode\_produk** pada hasil query dengan INNER JOIN di atas, dengan kolom **kode\_produk** pada tabel **ms\_produk**. Menurutmu apa yang berbeda?”

“Hm, aku tidak menemukan **prod-06** di tabel hasil query, padahal di tabel **ms\_produk** terdapat data **prod-06** yaitu pulpen multifunction + laser DQLAB.”

“Tepat sekali, selanjutnya bandingkan dengan tabel **tr\_penjualan**, apakah kamu menemukan data **prod-06** di tabel **tr\_penjualan**?” tanya Senja kembali menguji kepemahamanku.Dan, untungnya aku tak pernah mengantuk saat mendengar penjelasan Senja sepanjang materi. Kalau iya, habislah sudah!

“Tidak ada transaksi dengan **kode\_produksi** **prod-06** di tabel **tr\_penjualan**.   Sepertinya aku mulai paham. INNER JOIN akan mencocokkan key kolom tabel **tr\_penjualan** dengan key kolom **ms\_produk**, jika value dari key kolom sama - sama ada di kedua tabel , maka baris itu akan dikembalikan sebagai hasil query dan membentuk satu tabel yang berisi seluruh kolom dari kedua tabel, sedangkan jika value key kolom hanya terdapat di satu kolom, maka baris ini tidak akan ditampilkan. **prod-06** ada di tabel di **ms\_produk** tetapi tidak ada di data penjualan sehingga saat penggabungan tabel dengan INNER JOIN, data **prod-06** dari tabel **ms\_produk** tidak akan muncul di hasil query.”

Bibir Senja melengkung. Aku bisa mendapati senyum puas Senja atas jawabanku. Senyum apresiasi dari Senja selalu mampu menghangatkan hatiku. Senangnya! Walau sederhana tapi bermakna.

“Tepat Sekali, Aksara. Sepertinya kamu sudah paham dengan penggunaan INNER JOIN. Silahkan lanjutkan  materinya di modul. Oh iya, pada bagian SELECT, kamu boleh memilih kolom mana saja yang ingin kamu tampilkan setelah kedua tabel di-JOIN.”

### Syntax INNER JOIN – Memilih Beberapa Kolom Untuk Ditampilkan

“Satu lagi, Aksara. Sementara itu, jika kita ingin menampilkan beberapa kolom saja dari tabel hasil penggabungan, maka syntax SQL dari INNER JOIN dinyatakan sebagai berikut ini.”



Bagian query

**SELECT** tabel1.nama\_kolom1, tabel1.nama\_kolom2, ..., tabel2.nama\_kolom2, ....   
**FROM** tabel1

menghendaki pemilihan kolom mana saja dari kedua tabel yang akan digabungkan. Disini diperlukan penggunaan prefix nama tabelnya.

Selanjutnya, bagian query

INNER JOIN tabel2

digunakan untuk menggabungkan **tabel1** dengan **tabel2.**

Akhirnya, bagian query

**ON** tabel\_1.nama\_kolom1 = tabel2.nama\_kolom1;

adalah acuan penggabungan **tabel1** dan **tabel2** berdasarkan kolom yang memiliki tingkat kecocokan yang tinggi. Kedua kolom dengan tingkat kecocokan yang tinggi pada masing-masing tabel disebut juga dengan **key column**.

Catatan:

Perlu diperhatikan jika menampilkan kolom dengan nama yang sama di kedua tabel, maka pada bagian Select, tidak bisa hanya mengetikkan nama kolom saja, tetapi juga harus didahului oleh prefix nama tabel dimana kolom itu berasal untuk menghindari error karena ambiguitas," Senja mengingatkanku sekali lagi.

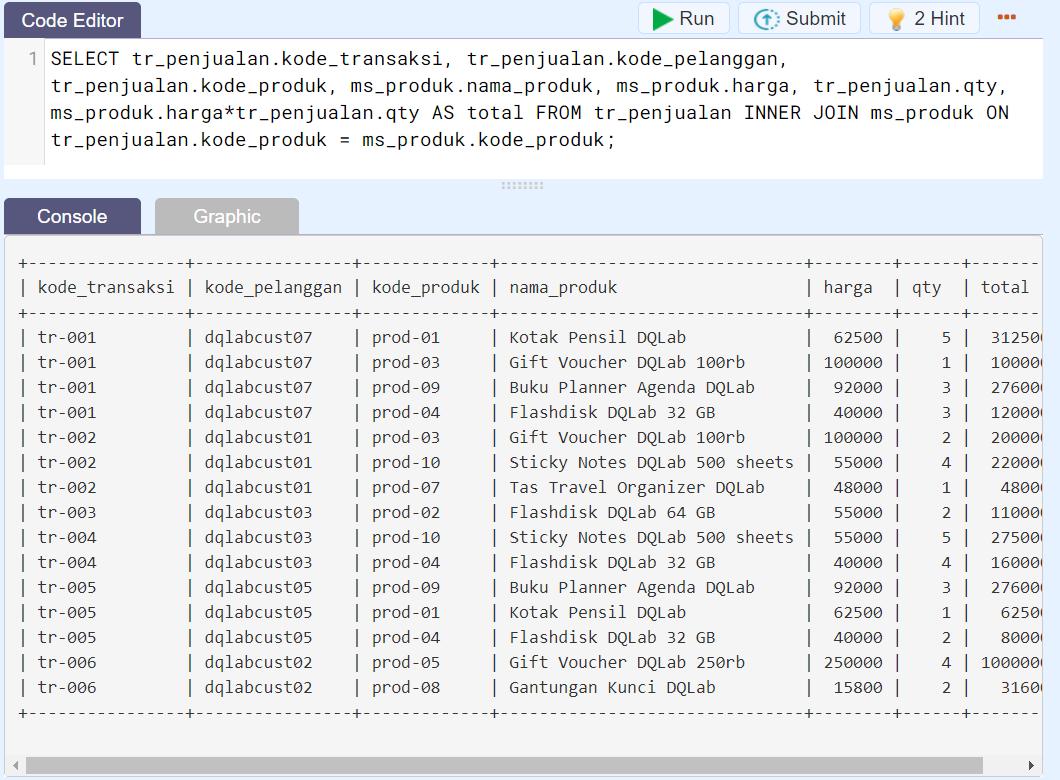
### Tugas Praktek: Menggunakan INNER JOIN (3/3)

“Oke, semua dariku sudah. Sekarang silakan dipraktikkan pada code editor untuk menggabungkan tabel **tr\_penjualan** dan **ms\_produk** dengan kolom yang ditampilkan dari tabel **tr\_penjualan** adalah **kode\_transaksi, kode\_pelanggan, kode\_produk, qty.** Untuk tabel **ms\_produk** tampilkan kolom **nama\_produk** dan **harga**."

Aku mengangguk mantap. Senja sudah membekaliku sampai di sini. Ini saatnya aku mandiri.

Kemudian aku membentuk kolom **total** yang merupakan hasil perkalian setiap baris pada kolom **harga** di tabel **ms\_produk** dengan kolom **qty** di tabel **tr\_penjualan**.

"Tabel hasil penggabungan haruslah membentuk kolom-kolom dengan urutannya adalah **kode\_transaksi, kode\_pelanggan, kode\_produk, nama\_produk, harga, qty,** dan **total,"**lanjut Senja lagi mengingatkan.



### Kesimpulan

Pada *chapter* **INNER JOIN** ini kita telah mempelajari bagaimana menggabungkan dua tabel berdasarkan key column-nya.

Pada tahap awal kita menggunakan INNER JOIN untuk menggabungkan keseluruhan kolom yang dimiliki oleh kedua tabel tersebut.

Selanjutnya, kita menerapkan INNER JOIN dengan menggunakan prefix nama tabel untuk memilih kolom-kolom mana saja yang akan ditampilkan pada tabel hasil penggabungan.

Untuk *chapter* selanjutnya kita akan menggabungkan tabel dengan menggunakan **UNION**.

## CHAPTER 3

### Pendahuluan

Aku sedang mengerjakan kuis-kuis kecil ketika Senja datang membawa dua gelas minuman hangat. Dari aromanya bisa kutebak kalau itu kopi.

“Kopi saya. Ini teh susu buatmu.” Pagi di ruangan kantor yang masih sejuk-sejuk dingin memang paling pas minum yang anget! Dan, Senja masih ingat kalau aku enggak ngopi.

“Hari ini masih lanjut belajarnya, Aksara. Masih semangat kan?” tanya Senja sambil menyeruput kopinya lagi.

“Tentang apa materinya?”

“Saya melihat prosesmu bagus. Jadi kita sudah bisa lanjut ke bagian yang lebih kompleks, yaitu mengakses data dalam database dengan SQL (*Structure Query Language).*” Senja mengatakannya dengan begitu santai, padahal aku yang mendengarnya pun sudah mengerutkan alis.

“Sip, aku tak mungkin menolak bukan? Hehehe,” jawabku terkekeh.

“Pembelajaran kita hari ini akan dimulai dengan studi kasus praktik jadi learning by doing. Ini ada data berupa analisis hasil penjualan suatu store. Akan tetapi ketika kita coba cek di database, terdapat 2 tabel penjualan, yaitu :

Tabel A berisi data transaksi untuk kode transaksi ‘tr-001’ sampai ‘tr-003’, dan

Tabel B berisi data data transaksi untuk kode transaksi ‘tr-004’ sampai ‘tr-006’," buka Senja.

“Lalu, bagaimana cara yang cepat dan efektif untuk membuat analisis dari kedua tabel tersebut menggunakan SQL? Apakah bisa menggunakan metode JOIN seperti yang kupelajari sebelumnya?”

“Mari kita selesaikan bersama.”

### Penggabungan hasil SELECT secara “Vertikal”

“Untuk kasus seperti ini kita perlu menggunakan metode UNION agar kedua tabel tadi menjadi satu tabel,” jelas Senja.

“UNION itu apa dan bagaimana mengoperasikannya?” tanyaku penuh rasa ingin tahu.

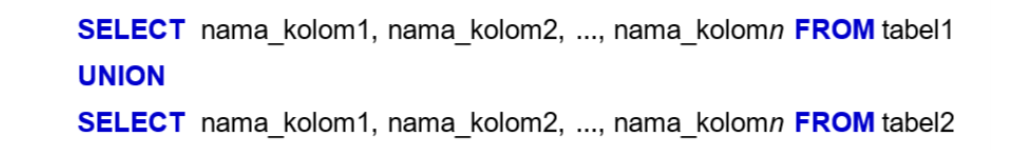
“**UNION** adalah operator SQL yang digunakan untuk menggabungkan hasil dari 2 atau lebih **SELECT** - statement secara “Vertikal”, dengan **catatan**,” Senja membuka buku catatannya dan menggesernya padaku untuk dicerna.

Setiap hasil dari **SELECT** statement yang akan digabungkan **(UNION)**memiliki jumlah kolom yang sama

Kolom tersebut juga harus memiliki tipe data yang sama, dan

Kolom tersebut memiliki urutan posisi yang sama.

Berikut format syntax-nya:

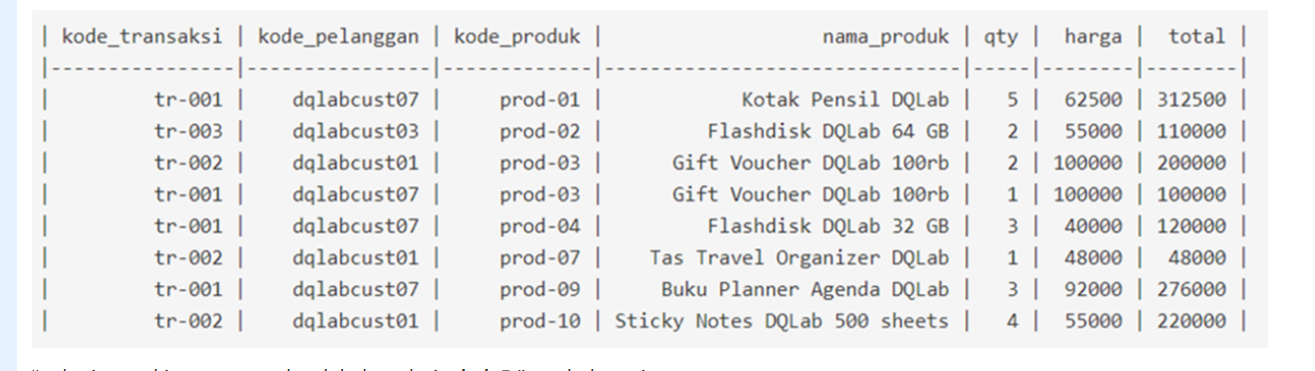


“Biar lebih mudah dipahami, mari kita praktekkan dengan tabel yang ingin dianalisis ini. Kamu bisa perhatikan ya langkah per langkahnya, Aksara.”

Aku menekuri layar laptop menunggu Senja beraksi memperlihatkan contoh.

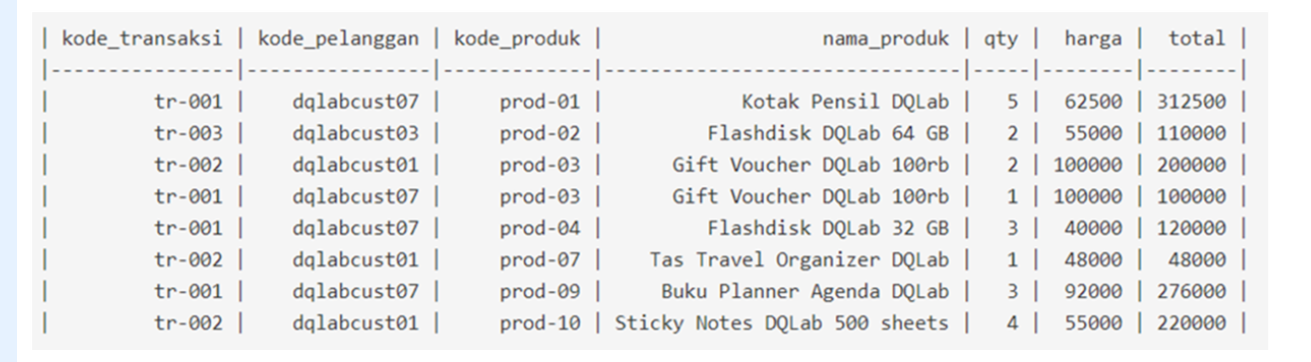
### Tabel yang Akan Digabungkan

“Oke, pertama - tama mari kita SELECT seluruh kolom dari **tabel\_A**.” Aku memperhatikan intruksi pertama Senja.



“Selanjutnya kita SELECT seluruh kolom dari **tabel\_B**,” tambah Senja.

Dan, hasilnya seperti ini :

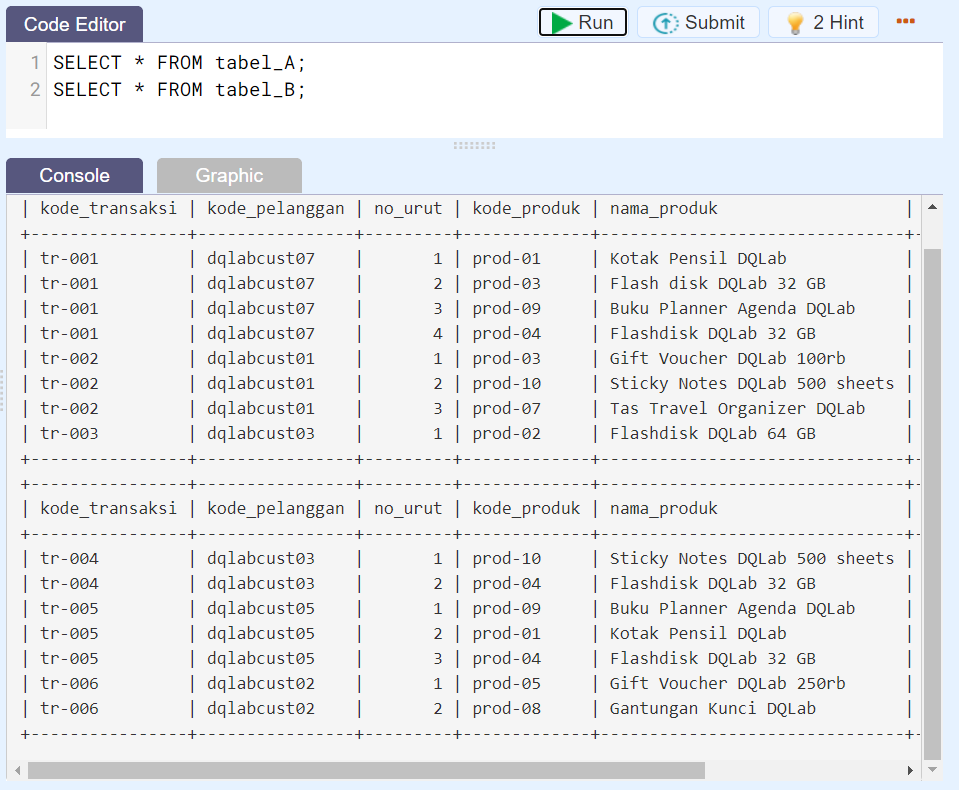


Sesuai dengan syarat untuk penggabungan dengan UNION yang telah dijelaskan tadi bahwa:

jumlah kolom **tabel\_A** dan **tabel\_B** adalah sama

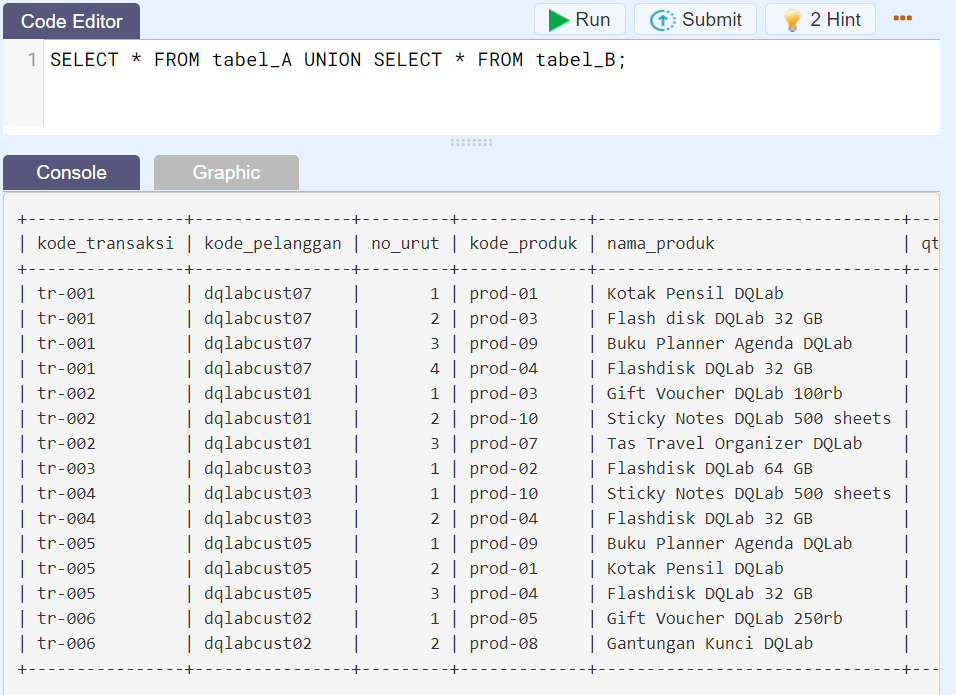
kolom-kolom pada **tabel\_A** dan **tabel\_B** memiliki tipe data yang sama, dan

kolom-kolom pada **tabel\_A** dan **tabel\_B** memiliki urutan posisi yang sama.



### Menggunakan UNION

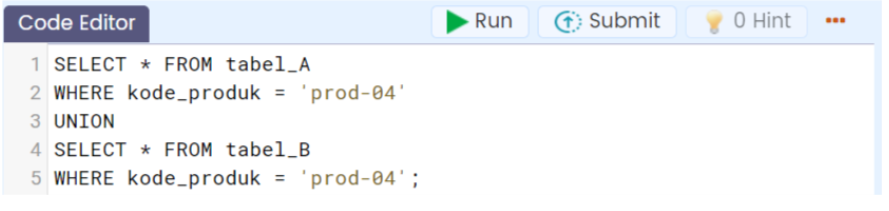
Kedua **tabel\_A** dan **tabel\_B** sudah memiliki jumlah kolom yang sama, dan juga urutan posisi kolom juga sama, jadi bisa langsung menggabungkan kedua kolom tersebut dengan menambahkan **UNION**.



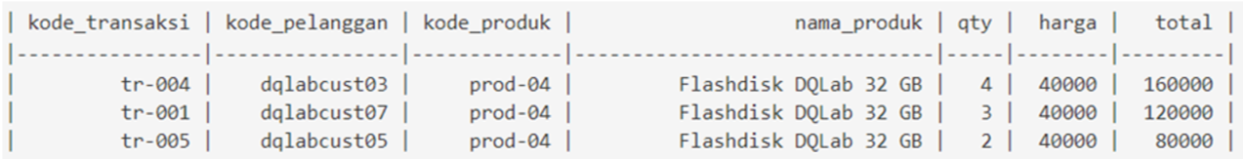
### Menggunakan UNION dengan Klausa WHERE

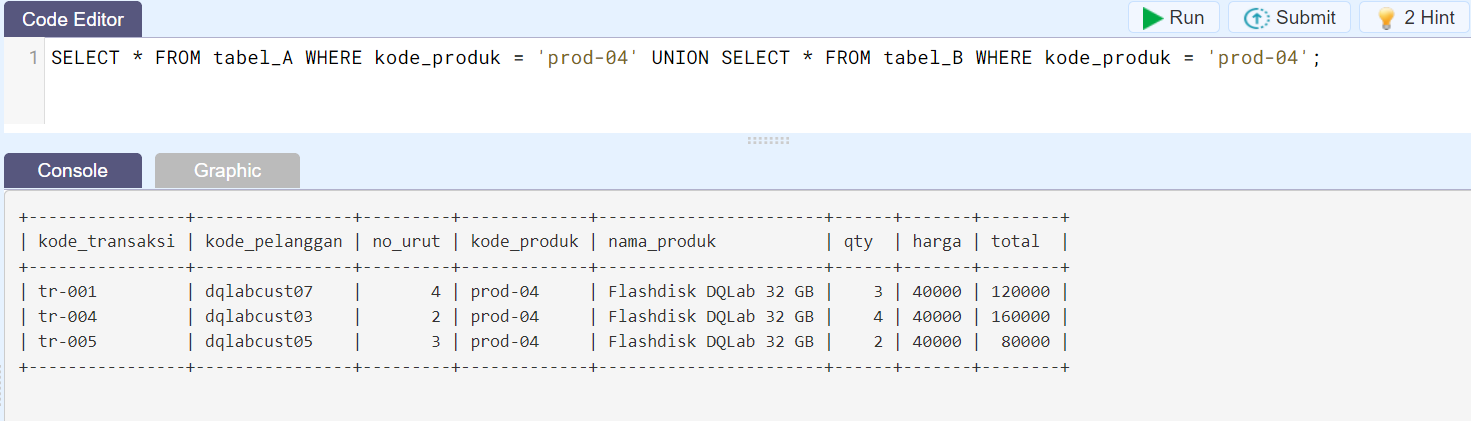
Aku bertanya pada Senja, “Terus, kalo ada kondisi WHERE, syntaxnya bagaimana? Misalnya aku hanya ingin menggabungkan tabel yang isinya data penjualan untuk kode produk **prod-04** saja?”

”Mudah saja, tinggal tambahkan **WHERE** di kedua **SELECT**-statement, seperti berikut ini,”



Jika query-nya dengan benar dituliskan dan kemudian dijalankan maka akan diperoleh :

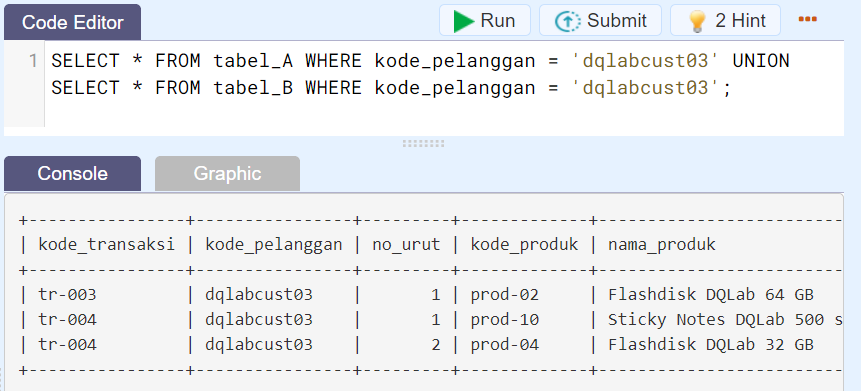




#### **Tugas Praktek:**

Lakukanlah hal yang sama dengan yang dicontohkan, akan dipilih kode\_pelanggan = 'dqlabcust03' sebagai kondisinya.

Jika query-nya diketikkan dengan benar maka tabel penggabungan yang tampil dengan kondisi kode\_pelanggan = 'dqlabcust03' adalah:



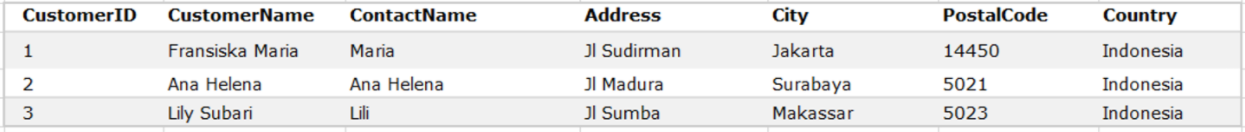
### Menyelaraskan (Conforming) Kolom

Aku diam sebentar untuk menyimak. Tapi, masih ada pertanyaan yang mengganjal di benakku.

“Hmm, aku masih bingung, Nja. Kebetulan data penjualan ini berada di kedua tabel A & B jumlah kolom dan posisinya sama serta nama kolomnya sama. Bagaimana kalau posisi kolom dari kedua tabelnya tidak sama? Apa tidak bisa di-**UNION**-kan?”

“Tentu saja bisa, kamu bisa menyelaraskan kolom dari kedua tabel di SELECT-statement. Mari kita contohkan dengan data dari tabel berikut ini.”

tabel Customers



dan tabel **Supplier**



Jumlah kolom dari kedua tabel tersebut sama - sama 7 kolom, tetapi kolom posisi kolom **ContactName** dari kedua tabel tidak sama. Di tabel **Customer,** posisi kolom **ContactName** berada di Kolom ke - 3 sedangkan di tabel supplier berada di kolom ke-2.

Jika langsung menggabungkan keduanya, tanpa menyelaraskan kolom hasilnya akan sebagai berikut:



Tentunya, ini hasil UNION yang tidak diinginkan, oleh karena itu, urutkan posisi kolom tersebut di SELECT-Statement dan juga pilih kolom yang ingin digabungkan, sehingga tidak perlu semua kolom dari kedua tabel di-UNION-kan, seperti berikut ini :

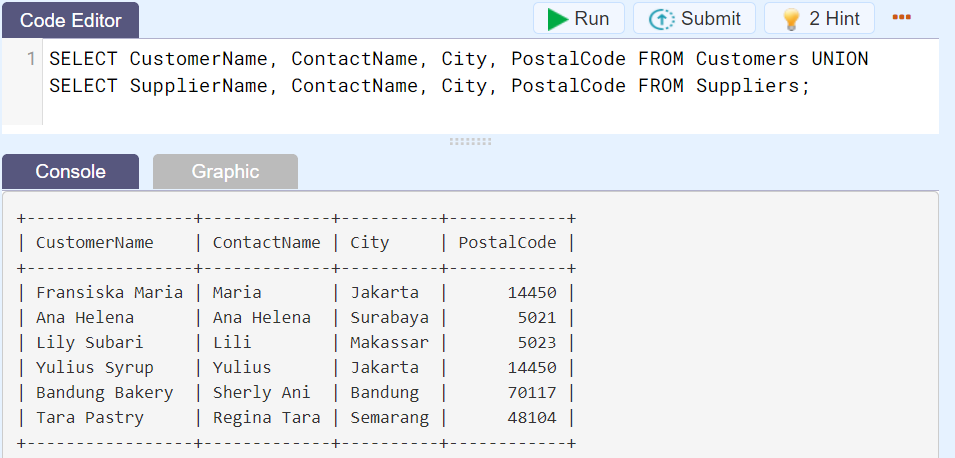
**SELECT** CustomerName, ContactName, City, PostalCode   
**FROM** Customers   
**UNION**   
**SELECT** SupplierName, ContactName, City, PostalCode   
**FROM** Suppliers;

Jika terdapat perbedaan nama kolom antara SELECT-statement pertama dan SELECT-statement kedua, maka secara default akan digunakan nama kolom dari SELECT-statement yang pertama.

### Menggunakan UNION dan Menyelaraskan Kolom-Kolomnya.

Senja menyerahkan tugas praktik sederhana untuk menguji materi ini. Sekilas kubaca pertanyaan dan perintahnya tidak sulit. Baiklah, mari langsung terapkan ilmunya!

Ketikkan syntax query di atas pada Code Editor, untuk melihat hasil dari penggabungan tabel (UNION) dengan menyelaraskan kolom.

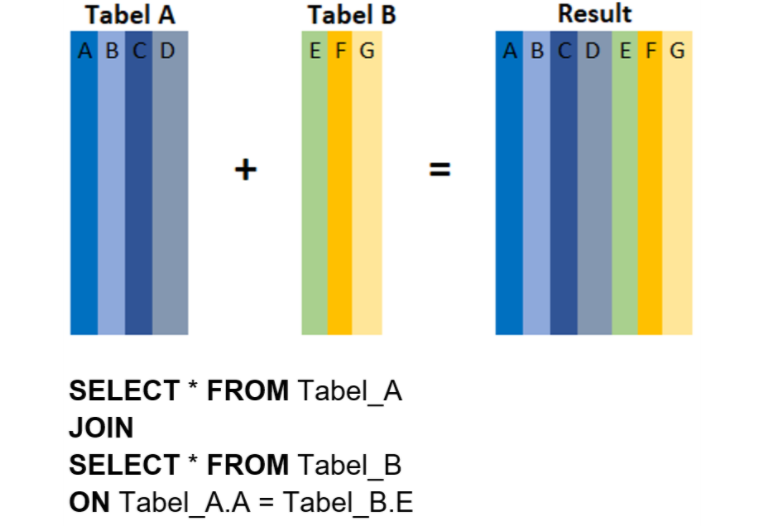


### Perbedaan antara UNION dan JOIN

Setelah mengerjakan tugas, aku jadi teringat materiku sebelumnya mengenai JOIN. Sebenarnya kalau dipikir-pikir fungsi keduanya tampak mirip. Jadi, kapan waktu yang tepat untuk memaki salah satu darinya? Aku memutuskan menanyakan hal ini pada Senja.

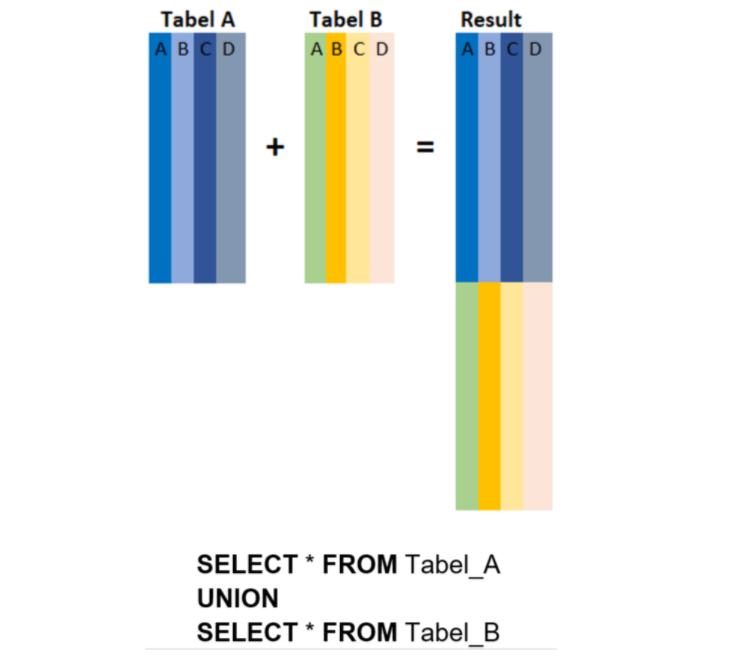
“ Nja, aku sudah paham bagaimana menggunakan UNION tetapi aku masih belum mengerti bedanya dengan metode JOIN, bukankah keduanya sama – sama untuk menggabungkan data dari 2 tabel? Lalu, kapan aku perlu pakai JOIN dan kapan aku perlu pakai UNION?”

“Memang benar UNION dan JOIN digunakan untuk menggabungkan data dari dua atau lebih tabel. Tapi yang membedakan adalah bagaimana tabel - tabel itu digabungkan. Kita menggunakan JOIN ketika akan menggabungkan tabel secara horizontal, sehingga hasil join akan memuat kolom - kolom dari kedua atau lebih tabel yang digabungkan. Berikut gambaran penggabungan tabel dengan metode JOIN,” Senja menampilkan contoh tabel di layar laptop.



Pada metode JOIN, penggabungan dilakukan berdasarkan key/kolom tertentu yang terdapat di tabel-tabel yang akan digabungkan dan key/kolom ini memiliki nilai yang saling terkait. Seperti yang terlihat pada gambar, Kolom A dan Kolom E merupakan key/kolom yang saling terkait sehingga kedua tabel dapat digabungkan dengan mencocokan nilai dari kedua kolom ini. Proses JOIN tidak dapat dilakukan jika tidak terdapat key/kolom yang saling terkait di kedua atau lebih tabel yang akan digabungkan.

Untuk UNION seperti yang sudah dijelaskan, digunakan ketika ingin menggabungkan tabel secara secara vertikal yaitu menggabungkan baris/row dari dua atau lebih tabel. Tidak seperti JOIN, untuk penggabungan dengan UNION, tidak diperlukan key/kolom yang saling terkait tetapi UNION mensyaratkan bahwa jumlah kolom dari tabel - tabel yang akan digabungkankan adalah sama dan berada diposisi yang sama pula. Berikut ilustrasi penggabungan dengan UNION:



Pada proses penggabungan UNION, tidak terdapat penambahan kolom tetapi jumlah baris/rows yang akan bertambah.

### Kesimpulan

Pada *chapter* **UNION** ini telah dipelajari bagaimana menggabungkan dua tabel secara vertikal (bertambah barisnya). Tentunya ada syarat yang harus dipenuhi oleh kedua tabel yang digabungkan dengan UNION, yaitu:

* Setiap hasil dari **SELECT** statement yang akan digabungkan **(UNION)**memiliki jumlah kolom yang sama
* Kolom tersebut juga harus memiliki tipe data yang sama, dan
* Kolom tersebut memiliki urutan posisi yang sama.

Selain itu, mempelajari bagaimana penyelerasan kolom sehingga record/baris yang ditampilkan pada tabel hasil penggabungan memiliki arti.

Perbedaan mendasar dari JOIN dan UNION adalah JOIN menggabungkan 2 tabel atau lebih berdasarkan baris yang saling berelasi/terkait sedangkan UNION menggabungkan 2 tabel secara vertikal.

## CHAPTER 4

### Pendahuluan

“Sudah bisa menjawab kuis-kuisnya, Aksara?”

Aku mengangguk. Syukurlah materi baru hari ini berjalan lancar dan aku belum menemukan persoalan berarti. “Ini juga berkat kamu,” pujiku tulus.

“Baiklah, kamu tentu tahu kenapa saya cukup ‘ngebut’ mengajarimu banyak hal minggu ini. Soalnya dari atasan sudah menyiapkan mini proyek untuk kamu. Untuk mengerjakan proyek ini kamu perlu menguasai kompetensi dari materi yang diajarkan sepanjang minggu.”

Ini bukan kali pertama aku diberi proyek setelah pembelajaran berakhir. Tapi, tetap saja tiap kali mendengarnya berhasil bikin aku tegang. Seperti biasa, aku tetap menyanggupi.

“Sekarang istirahat saja dulu, Aksara. Sebentar lagi jam makan siang. Aku akan mengirim detail proyeknya melalui email. Nanti bisa langsung dikerjakan ya, tenggat waktunya besok siang,” tambah Senja sembari mengedipkan satu matanya padaku saat mengatakan tenggat waktu.

Dan, email berisi mini proyek itu pun mampir ke kotak masuk kurang dari setengah jam sesudahnya:

1. Project INNER JOIN, dan
2. Project UNION.

Akupun membuka project pertama (INNER JOIN).

### Project INNER JOIN

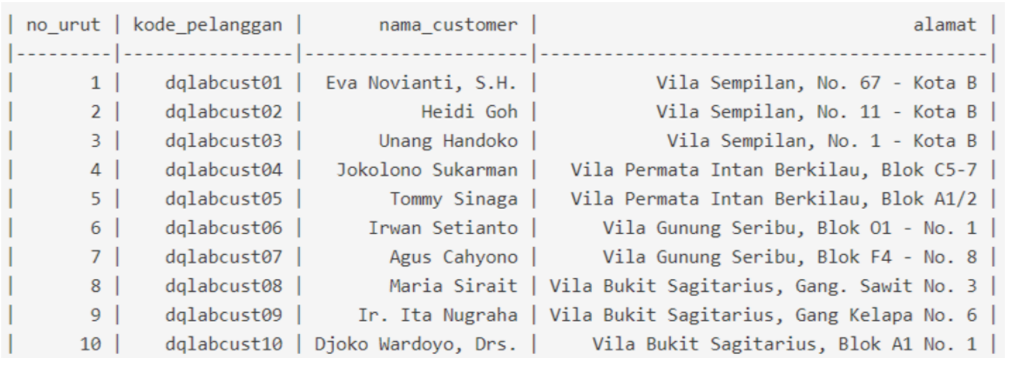
Dalam database, terdapat tabel **ms\_pelanggan** yang berisi data - data pelanggan yang membeli produk dan tabel **tr\_penjualan** yang berisi data transaksi pembelian di suatu store.

Suatu hari, departemen marketing & promotion meminta bantuan untuk meng-query data-data pelanggan yang membeli produk Kotak Pensil DQLab, Flashdisk DQLab 32 GB, dan Sticky Notes DQLab 500 sheets.

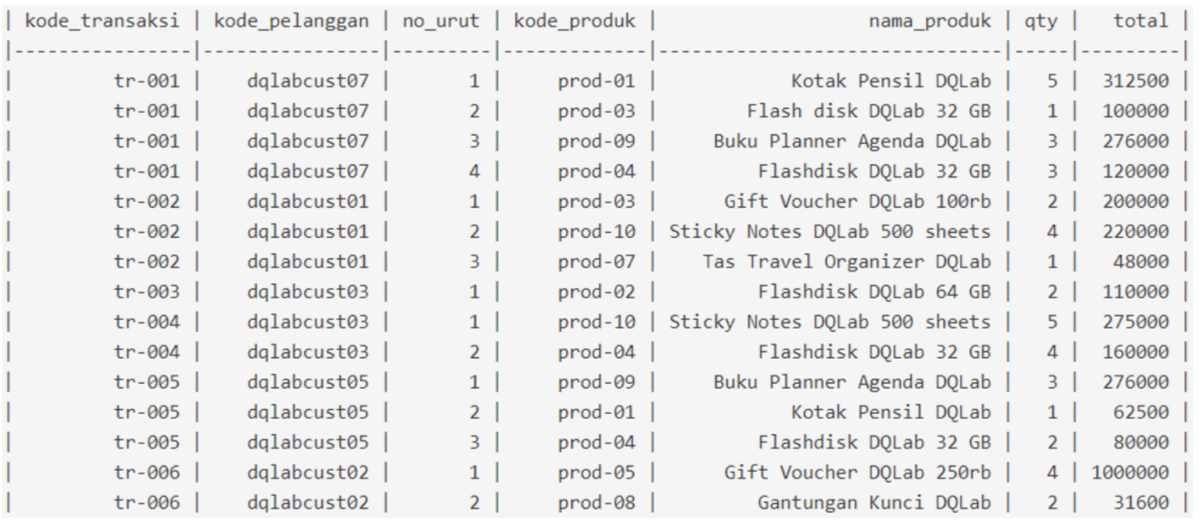
Buatlah query menggunakan tabel **ms\_pelanggan** dan **tr\_penjualan** untuk mendapatkan data - data yang diminta oleh marketing yaitu **kode\_pelanggan, nama\_customer, alamat.**

**NB:** Gunakan SELECT DISTINCT untuk menghilangkan duplikasi, jika diperlukan.

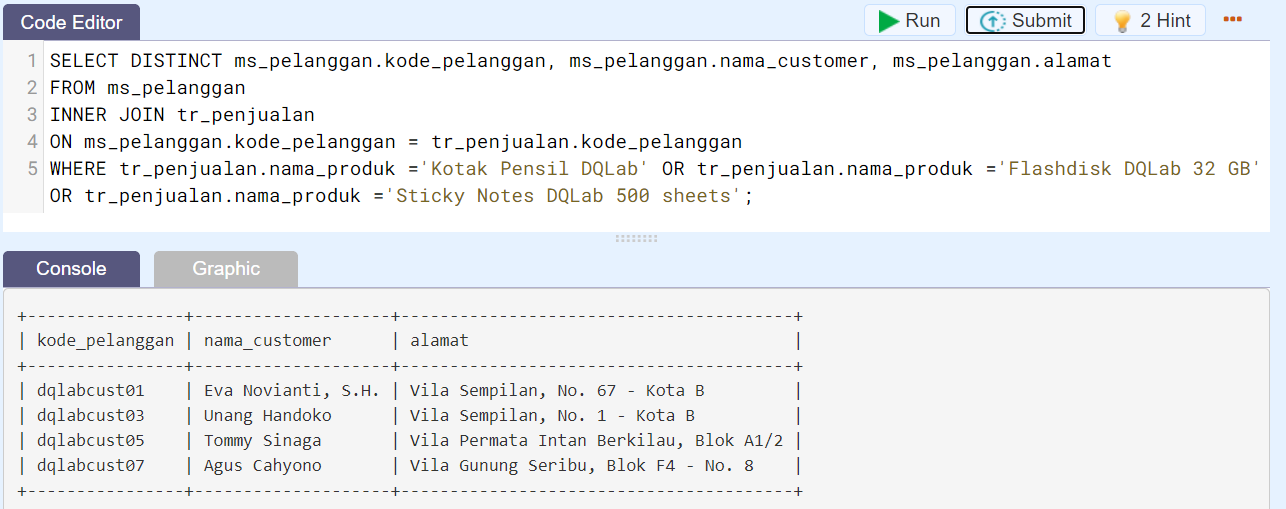
Tabel**ms\_pelanggan**



Tabel**tr\_penjualan**



Hasil :



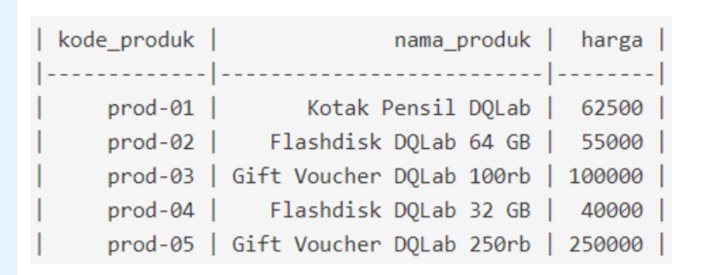
### Project UNION

Siapkan data katalog mengenai mengenai nama - nama produk yang dijual di suatu store. Data tersebut akan digunakan dalam meeting untuk mereview produk mana saja yang akan dilanjutkan penjualannya dan mana yang akan di diskontinu.

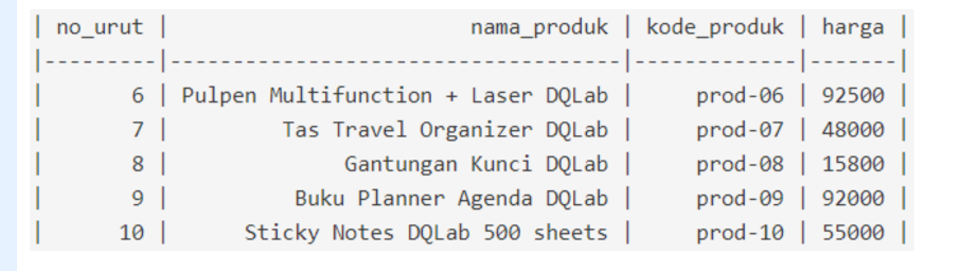
Siapkan hanya data produk dengan harga di bawah 100K untuk kode produk prod-1 sampai prod-5; dan dibawah 50K untuk kode produk prod-6 sampai prod-10.

Saat mengecek data produk di database, terdapat 2 tabel yang sama - sama berisi data katalog, yaitu :

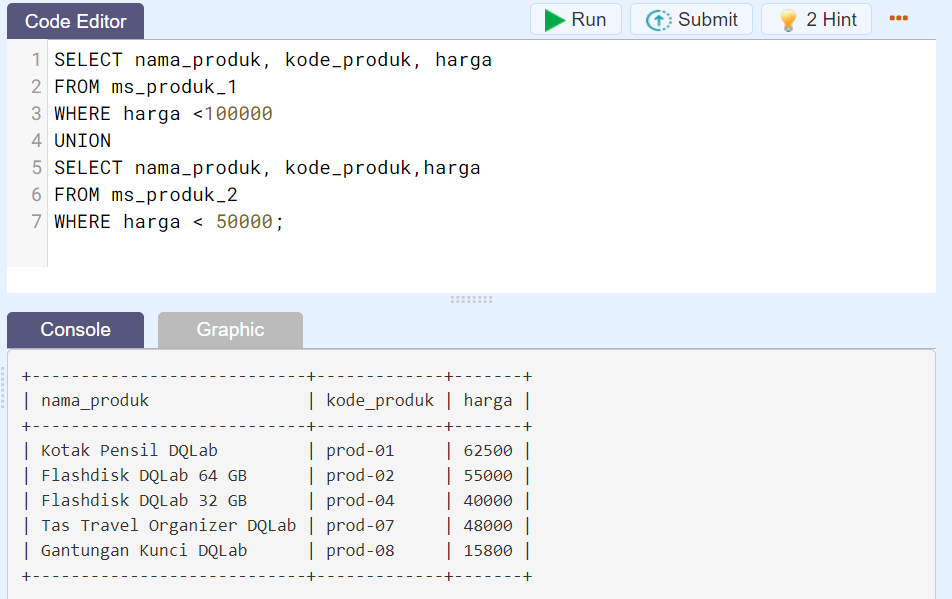
Tabel**ms\_produk\_1**



Tabel **ms\_produk\_2**



HASIL :



### Hasil Belajarku

Wah, tidak terasa aku telah menyelesaikan modul **Fundamental SQL Using INNER JOIN and UNION**. Selama belajar dengan modul ini, aku sudah dapat memahami dan mampu mempraktikkan:

* Penggabungan dua tabel dengan menggunakan WHERE clause dan teknik cross join.
* Penggabungan dua tabel relasi dengan menggunakan INNER JOIN.
* Penggabungan dua tabel secara vertikal dengan menggunakan UNION.
* Mengerjakan mini project yang merupakan integrasi keseluruhan materi dan tentunya materi-materi pada modul-modul sebelumnya untuk menyelesaikan persoalan bisnis.

Dengan kemampuan ini, aku lebih pede untuk mengolah data dengan SQL. Keterampilan ini sendiri adalah 60% aktivitas awal yang akan dilakukan seorang analis. **Keep Fighting!**