UJIAN TENGAH SEMESTER

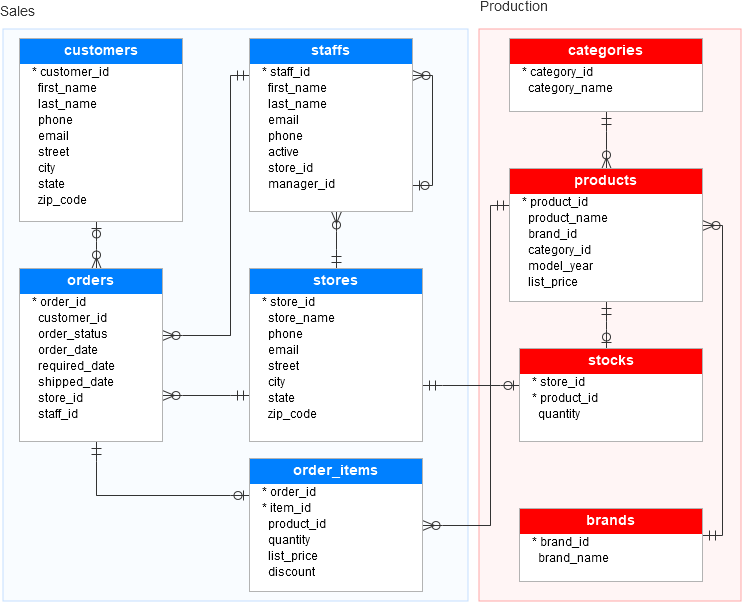
Nama : Muhammad Abhinaya Zurfa

Kelas : SIB 2 F/ 18

Mata Kuliah : Basis Data Lanjut

# Studi Kasus Toko Sepeda

Toko Fast merupakan toko yang menjual segala jenis sepeda. Gambar dibawah ini merupakan gambar desain database diagram Toko Fast



**DETAIL DATABASE**

**Tabel sales.stores**

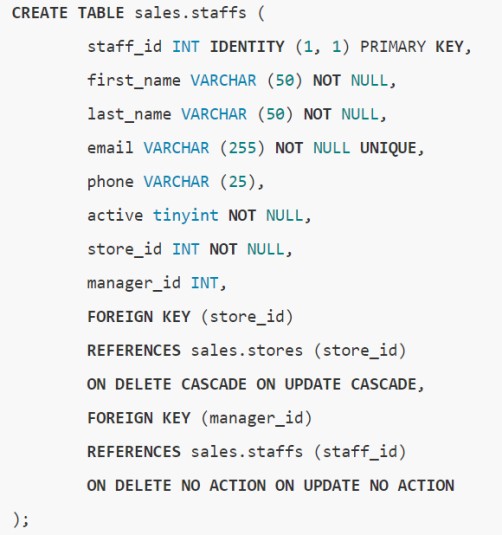
**Tabel sales.stores** berisi informasi toko. Setiap toko memiliki nama toko, informasi kontak seperti telepon dan email, dan alamat termasuk jalan, kota, negara bagian, dan kode pos. Berikut ini detail source code untuk mebuat tabel sales.stores



# Tabel sales.staffs

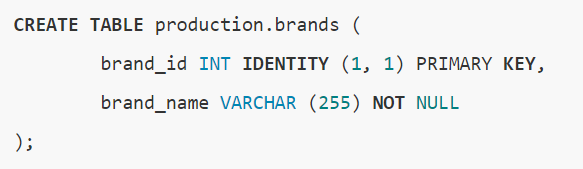
**Tabel sales.staffs** menyimpan informasi penting dari staf termasuk nama depan, nama belakang. Ini juga berisi informasi komunikasi seperti email dan telepon. Seorang staf bekerja di toko yang ditentukan oleh nilai di kolom store\_id. Sebuah toko dapat memiliki satu atau lebih staf. Seorang staf melapor kepada manajer toko yang ditentukan oleh nilai di kolom manager\_id. Jika nilai di manager\_id adalah null, maka staf adalah berperan sebagai top manajer. Jika staf tidak lagi bekerja untuk penyimpanan mana pun, nilai di kolom aktif di set ke nilai zero/nol.

Berikut ini detail source code untuk mebuat tabel sales.staffs



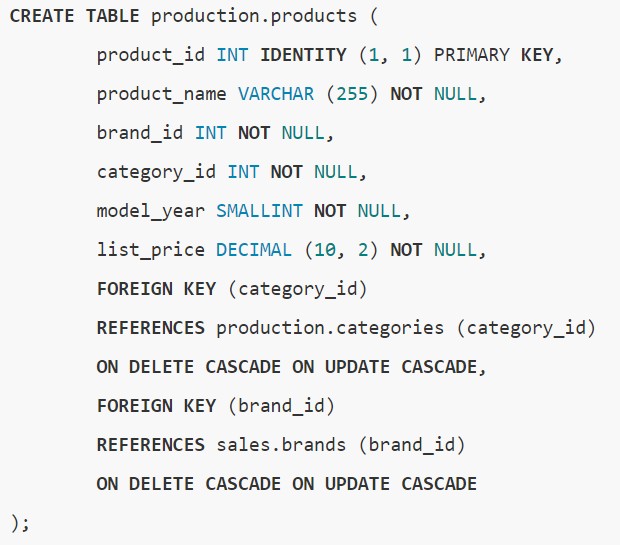
# Tabel production.brands

**Tabel production.brands** menyimpan data tentang informasi merk sepeda contohnya Electra, Haro, and Heller. Berikut ini detail source code untuk mebuat tabel production.brands



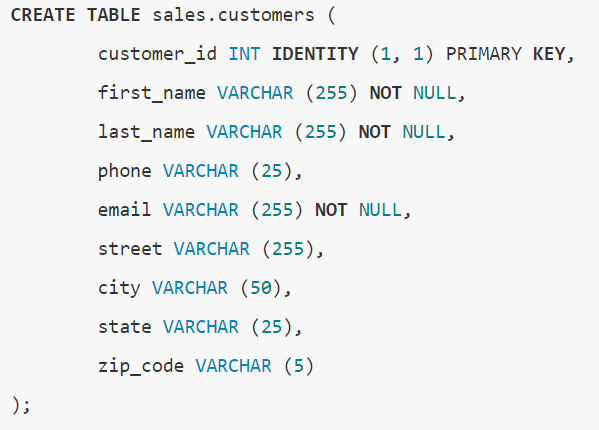
# Tabel production.products

**Tabel production.products** menyimpan informasi produk seperti nama, merek, kategori, model tahun, dan daftar harga. Setiap produk milik merek yang ditentukan oleh kolom brand\_id. Karenanya, sebuah merek mungkin memiliki nol atau banyak produk. Setiap produk juga termasuk dalam kategori yang ditentukan oleh kolom category\_id. Selain itu, setiap kategori mungkin memiliki nol atau banyak produk. Berikut ini detail source code untuk mebuat tabel production.products



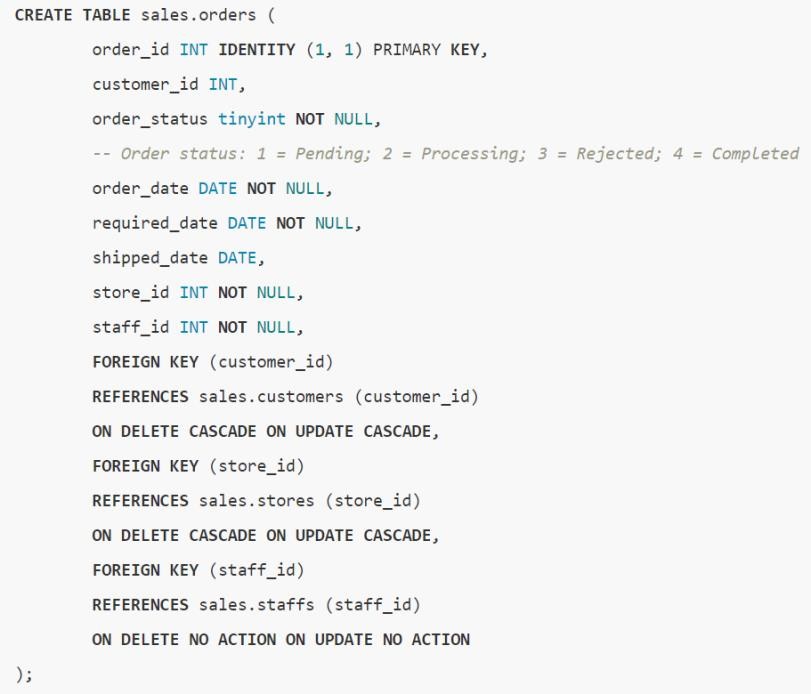
# Tabel sales.customers

**Tabel sales.customers** menyimpan informasi tabel pelanggan termasuk nama depan, nama belakang, telepon, email, jalan, kota, negara dan kode pos. Berikut ini detail source code untuk mebuat tabel sales.customers



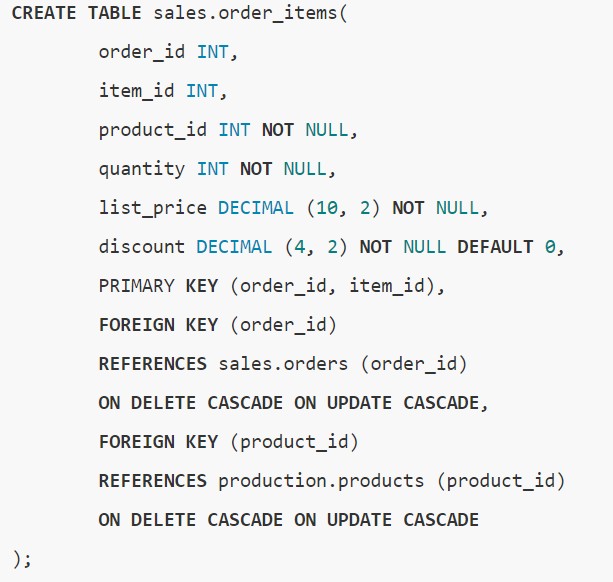
# Tabel sales.orders

Tabel sales.orders menyimpan informasi header order penjualan ini termasuk pelanggan, status order, tanggal order, tanggal yang dibutuhkan, tanggal dikirim. Ini juga menyimpan informasi di mana transaksi penjualan dibuat (toko) dan siapa yang membuatnya (staf). Setiap pesanan penjualan memiliki baris di tabel sales\_orders. Pesanan penjualan memiliki satu atau banyak item baris yang disimpan di tabel sales.order\_items. Berikut ini detail source code untuk mebuat tabel sales.orders



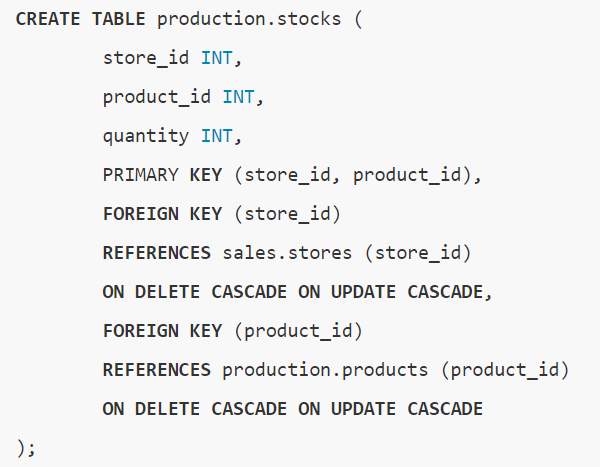
# Tabel sales.orders\_items

Tabel sales.order\_items menyimpan item baris dari order penjualan. Setiap item baris milik pesanan penjualan yang ditentukan oleh kolom order\_id. Item baris pesanan penjualan mencakup produk, kuantitas pesanan, harga jual, dan diskon. Berikut ini detail source code untuk mebuat tabel sales.orders\_items



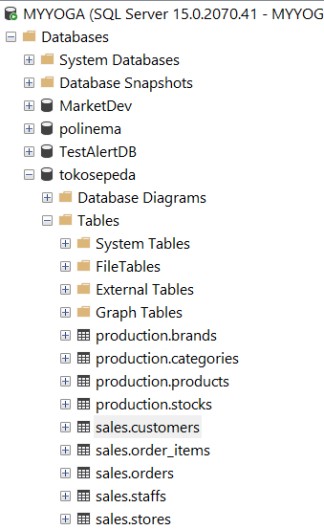
# Tabel production.stocks

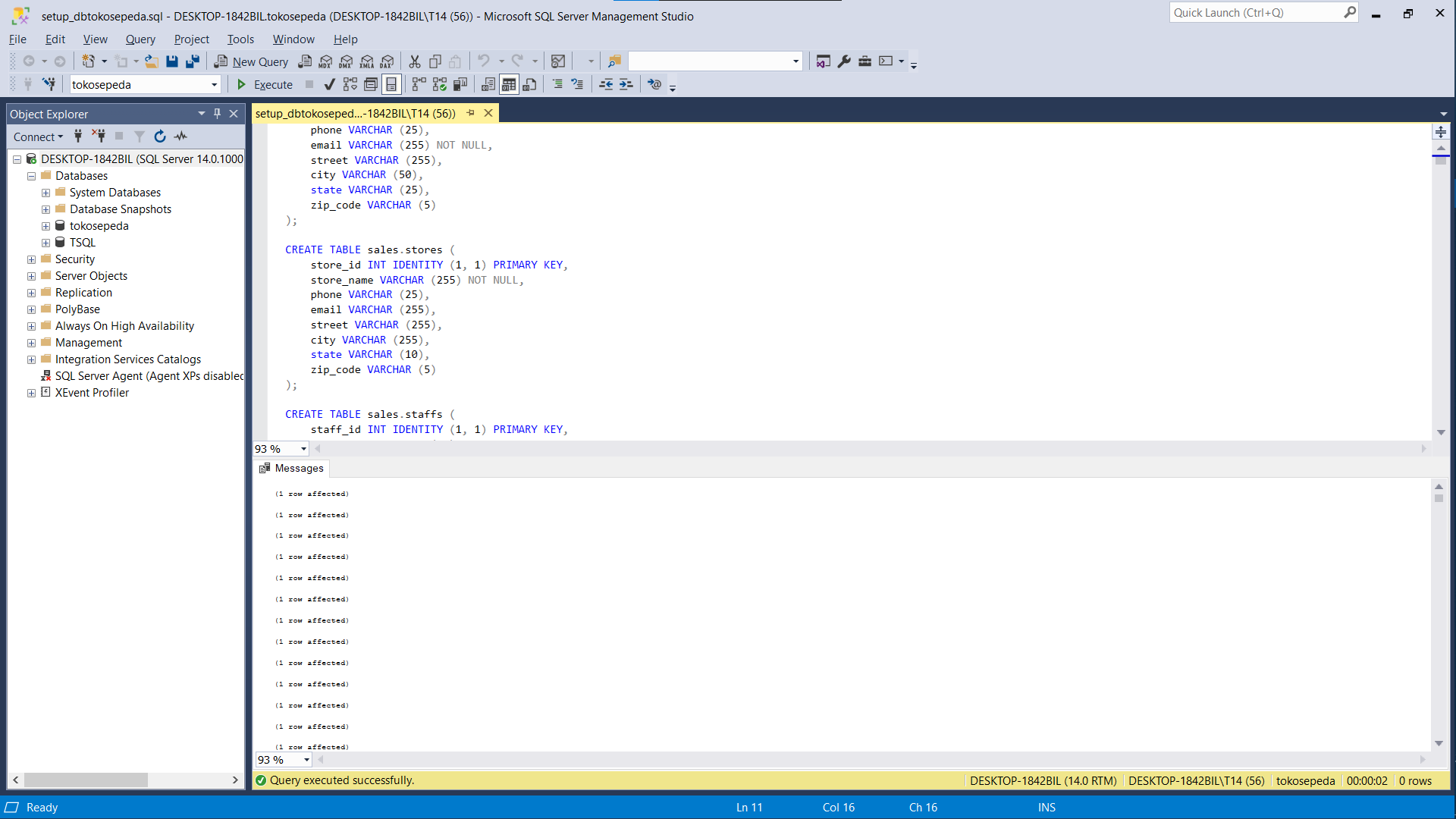
production.stocks menyimpan informasi persediaan yaitu kuantitas produk tertentu di toko tertentu. Berikut ini detail source code untuk mebuat tabel production.stocks



# SOAL

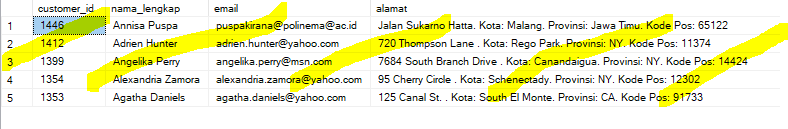
1. Silahkan jalankan file setup.sql di SQL Server Anda. Source code ini berfungsi untuk membuat database toko sepeda dengan detail yang sudah dijelaskan di penjelasan sebelumnya



**

1. Tampilkan informasi dari **tabel sales.customers** dengan ketentuan huruf awal di nama depan mereka sama dengan nama depan Anda kemudian urutkan berdasarkan customer\_id yang paling besar terlebih dahulu dan tampilkan hanya 3 data teratas. Data yang ditampilkan meliputi
   * customer\_id
   * nama\_lengkap : data merupakan gabungan dari kolom first\_name dan last name
   * email
   * alamat : data merupakan gabungan dari kolom street, city, dan zip\_code. Perhatikan penulisan data alamat seperti di contoh hasil

Capture source code dan hasil SELECT Anda Contoh hasil :

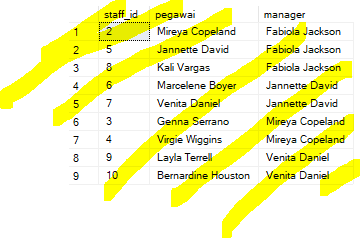


|  |
| --- |
| SELECT TOP 3  customer\_id,  CONCAT(first\_name, ' ', last\_name) AS nama\_lengkap,  email,  CONCAT(street, ', Kota: ', city, ', Provinsi: ', state, ', Kode Pos: ', zip\_code) AS alamat  FROM  sales.customers  WHERE  first\_name LIKE 'M%'  ORDER BY  customer\_id DESC; |

|  |
| --- |
|  |
|  |

**3a**. Tuliskan perintah SQL untuk melakukan **self join** pada tabel sales.staffs.

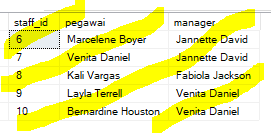
Lakukan seleksi dengan ketentuan sebagai berikut

* + - staff\_id
    - pegawai: data merupakan gabungan dari kolom first\_name dan last name employee
    - manager: data merupakan gabungan dari kolom first\_name dan last name employe

|  |
| --- |
| SELECT  ss1.staff\_id,  CONCAT(ss1.first\_name, ' ', ss1.last\_name) AS pegawai,  CONCAT(ss2.first\_name, ' ', ss2.last\_name) AS manager  FROM  sales.staffs ss1  JOIN  sales.staffs ss2 ON ss1.manager\_id = ss2.staff\_id  WHERE  ss2.staff\_id IS NOT NULL  ORDER BY  manager |
|  |

**3b.** Setelah itu, gunakan klausa **OFFSET-FETCH** sehingga data yang ditampilkan adalah data yang telah diurutkan berdasarkan staff\_id (kecil-besar) kemudian tampilkan data setelah 4 baris pertama dan tampilkan hanya 5 data saja

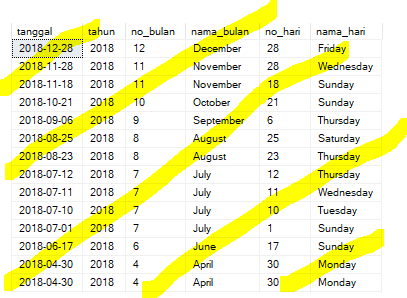
Contoh hasil :



|  |
| --- |
| SELECT  ss1.staff\_id,  CONCAT(ss1.first\_name, ' ', ss1.last\_name) AS pegawai,  CONCAT(ss2.first\_name, ' ', ss2.last\_name) AS manager  FROM  sales.staffs ss1  JOIN  sales.staffs ss2 ON ss1.manager\_id = ss2.staff\_id  ORDER BY  ss1.staff\_id ASC  OFFSET 4 ROWS  FETCH NEXT 5 ROWS ONLY; |
|  |

1. Pilih salah satu tabel yang memiliki kolom bertipe date time, kemudian ubah format tanggalnya menjadi bentuk tahun, bulan (format nomor dan nama), dan hari (format nomor dan nama)

Contoh hasil (diambil dari tabel sales.orders) :

 …dst

|  |
| --- |
| SELECT  FORMAT(order\_date, 'yyyy-dd-MM') AS tanggal,  FORMAT(order\_date, 'yyyy') AS tahun,  FORMAT(order\_date, 'MM') AS nomor\_bulan,  FORMAT(order\_date, 'MMMM') AS nama\_bulan,  FORMAT(order\_date, 'dd') AS nomor\_hari,  FORMAT(order\_date, 'dddd') AS nama\_hari  FROM  sales.orders  ORDER BY  tahun DESC; |
|  |

1. Buatlah perintah SQL untuk mengeluarkan jumlah total transaksi yang dikelompokkan berdasarkan rentang waktu tertentu (misal harian/ mingguan/bulanan/tahunan) dengan memanfaatkan klausa **Group By dan Fungsi aggregate.**

|  |
| --- |
| *Penjelasan tentang nama-nama tabel dan kolom yang di seleksi*  *Pada nomor 5 mencoba menghitung total transaksi bulanan dengan menggunakan table sales.orders dan sales.order\_item dengan kolom order\_date yang digunakan untuk mencari tahun dan bulan lalu untuk perhitungan total transaksi yang terjadi menggunakan count dari kolom order id*  *Dan yang di joinkan adalah pada kolom order\_id* |
| SELECT  DATEPART(YEAR, so.order\_date) AS tahun,  DATENAME(MONTH, so.order\_date) AS nama\_bulan,  COUNT(soi.order\_id) AS total\_transaksi  FROM  sales.orders so  JOIN  sales.order\_items soi ON so.order\_id = soi.order\_id  GROUP BY  DATEPART(YEAR, so.order\_date),  DATENAME(MONTH, so.order\_date),  DATEPART(MONTH, so.order\_date)  ORDER BY  tahun,  DATEPART(MONTH, so.order\_date); |
|  |

1. Buatlah perintah SQL untuk mengeluarkan jumlah total transaksi yang dikelompokkan berdasarkan rentang waktu tertentu (misal harian/ mingguan/bulanan/tahunan) dengan memanfaatkan **DERIVED TABLE**

|  |
| --- |
| *Penjelasan tentang nama-nama tabel dan kolom yang di seleksi*  *Perintah sama seperti sebelumnya tetapi pada hal ini menggunakan derived table yang dimana membuat seperti table sementara yang bertujuan untuk menyederhanakan dan memperjelas, pada derived table menggunakan table sales.orders dan sales.order\_items yang di joinkan melalui kolom order\_id lalu di select lagi querynya berdasarkan derived table tadi dengan select tahun, nama\_bulan dan total\_transaksi* |
| SELECT  tahun,  nama\_bulan,  COUNT(total\_transaksi) AS total\_transaksi  FROM (  SELECT  DATEPART(YEAR, so.order\_date) AS tahun,  DATENAME(MONTH, so.order\_date) AS nama\_bulan,  DATEPART(MONTH, so.order\_date) AS no\_bulan,  soi.quantity \* soi.list\_price \* (1 - soi.discount) AS total\_transaksi  FROM  sales.orders so  JOIN  sales.order\_items soi ON so.order\_id = soi.order\_id  ) AS derived\_table  GROUP BY  tahun,  nama\_bulan,  no\_bulan  ORDER BY  tahun,  no\_bulan; |
|  |

1. Buatlah perintah SQL untuk mengeluarkan jumlah total transaksi yang dikelompokkan berdasarkan rentang waktu tertentu (misal harian/ mingguan/bulanan/tahunan) dengan memanfaatkan **Common Table Expression**

|  |
| --- |
| *Penjelasan tentang nama-nama tabel dan kolom yang di seleksi*  *Perintah sama seperti sebelumnya juga untuk mencari transaksi bulanan tetapi pada soal ini menggunakan common table expression yang dimana mendefinisikan table sementara yang bise digunakan lagi dalam query selanjutnya, pada with TransaksiBulanan menggunakan kolom order\_date dari table sales.orders untuk mencari datepart, lalu untuk jumlah transaksi dihitung berdasarkan order\_id dari table sales order, lalu dipilih lagi / select tahun, nama\_bulan, total\_transaksi dari yang di CTE tadi (TransaksiBulanan) dan diurutkan berdasarkan tahun dan bulan sehingga hasilnya seperti dibawah* |
| WITH TransaksiBulanan AS (  SELECT  DATEPART(YEAR, so.order\_date) AS tahun,  MONTH(so.order\_date) AS bulan,  DATENAME(MONTH, so.order\_date) AS nama\_bulan,  COUNT(so.order\_id) AS total\_transaksi  FROM  sales.orders so  JOIN  sales.order\_items soi ON so.order\_id = soi.order\_id  GROUP BY  DATEPART(YEAR, so.order\_date),  MONTH(so.order\_date),  DATENAME(MONTH, so.order\_date)  )  SELECT  tahun,  nama\_bulan,  total\_transaksi  FROM  TransaksiBulanan  ORDER BY  tahun,  bulan; |
|  |