

BASIS DATA LANJUT
Subquery, Grouping, dan Aggregating



Lavina/2341760062

SIB 2D

PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS

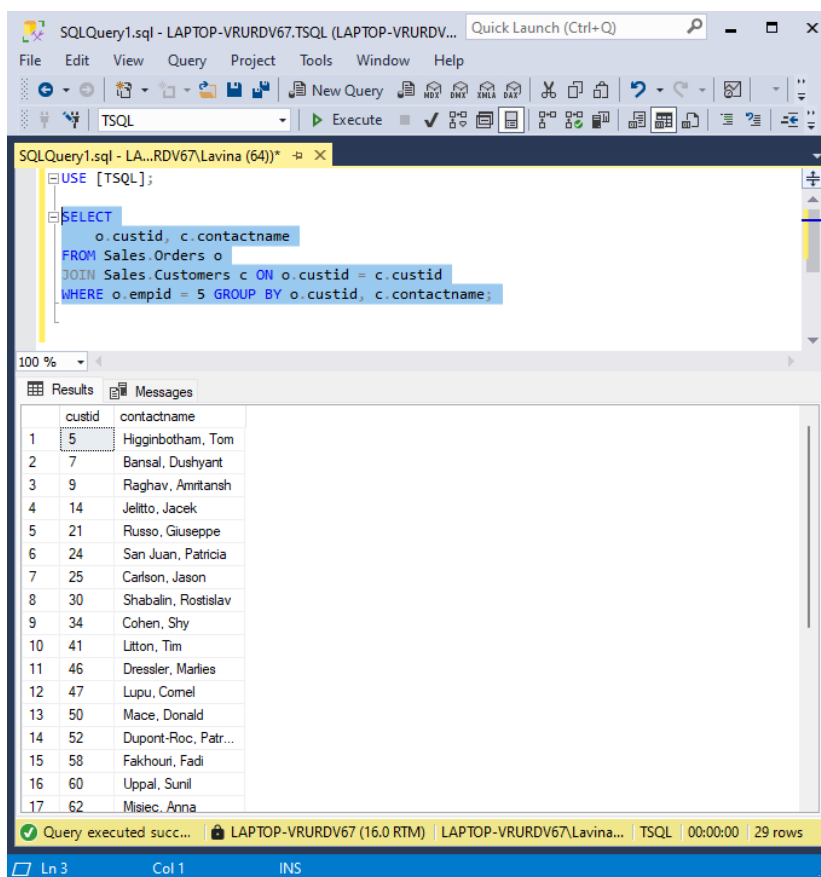
POLITEKNIK NEGERI MALANG

2024

Praktikum – Bagian 1: Menulis Query Menggunakan Klausa GROUP BY

Skenario : Departemen penjualan suatu perusahaan ingin menciptakan peluang up-sell tambahan dari para pelanggan. Untuk itu karyawan butuh melakukan analisis mengenai berbagai kelompok pelanggan dan kategori produk berdasar pada beberapa peraturan bisnis

[Soal-1] Tuliskan T-SQL SELECT yang akan menampilkan kelompok pelanggan yang melakukan pembelian. Klausa SELECT harus mencakup kolom custid dari tabel Sales.Orders dan kolom contactname dari tabel Sales.Customers. Kelompokkan kedua kolom tersebut, dan filter hanya pesanan dari sales employee yang memiliki empid sama dengan 5!



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query editor contains the following T-SQL code:

```
USE [TSQL];  
SELECT  
    o.custid, c.contactname  
FROM Sales.Orders o  
JOIN Sales.Customers c ON o.custid = c.custid  
WHERE o.empid = 5 GROUP BY o.custid, c.contactname;
```

The results pane displays a table with two columns: custid and contactname. The table contains 17 rows of data, with the first row highlighted.

	custid	contactname
1	5	Higginbotham, Tom
2	7	Bansal, Dushyant
3	9	Raghav, Amritansh
4	14	Jelitto, Jacek
5	21	Russo, Giuseppe
6	24	San Juan, Patricia
7	25	Carlson, Jason
8	30	Shabalin, Rostislav
9	34	Cohen, Shy
10	41	Litton, Tim
11	46	Dressler, Marlies
12	47	Lupu, Cornel
13	50	Mace, Donald
14	52	Dupont-Roc, Patr...
15	58	Fakhouri, Fadi
16	60	Uppal, Sunil
17	62	Misiec, Anna

The status bar at the bottom indicates that the query was executed successfully, returning 29 rows.

Penjelasan:

- JOIN digunakan untuk menggabungkan tabel Sales.Orders dan Sales.Customers berdasarkan kolom custid.
- WHERE o.empid = 5 digunakan untuk memfilter hanya pesanan dari sales employee dengan empid yang sama dengan 5.
- GROUP BY o.custid, c.contactname digunakan untuk mengelompokkan hasil berdasarkan custid dan contactname.

Query diatas menghasilkan daftar pelanggan yang melakukan pembelian, difilter berdasarkan karyawan yang melayani penjualan dalam kasus ini karyawan dengan empid = 5.

[Soal-2] Salin T-SQL jawaban soal-1. Kemudian modifikasi untuk menampilkan informasi tambahan kolom city dari tabel Sales.Customers pada klausa SELECT!

[Soal-3] Apakah terdapat pesan error pada jawaban soal-2? Apakah pesan errornya? Kenapa pesan itu bisa terjadi?

Messages
Msg 8120, Level 16, State 1, Line 4
Column 'Sales.Customers.city' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.
Completion time: 2024-09-26T18:12:04.9906156+07:00

Penjelasan: Terjadi error karena kolom c.city tidak ada dalam klausa GROUP BY. Oleh karena itu, SQL Server tidak tahu bagaimana mengelompokkan atau menangani nilai-nilai city ketika datanya hanya dikelompokkan berdasarkan custid dan contactname.

[Soal-4] Perbaiki error yang terjadi pada jawaban soal-2!

The screenshot shows a SQL Server Enterprise Manager window with a query executed successfully. The query is as follows:

```
SELECT
    o.custid, c.contactname, c.city
FROM Sales.Orders o
JOIN Sales.Customers c ON o.custid = c.custid
WHERE o.empid = 5 GROUP BY o.custid, c.contactname, c.city;
```

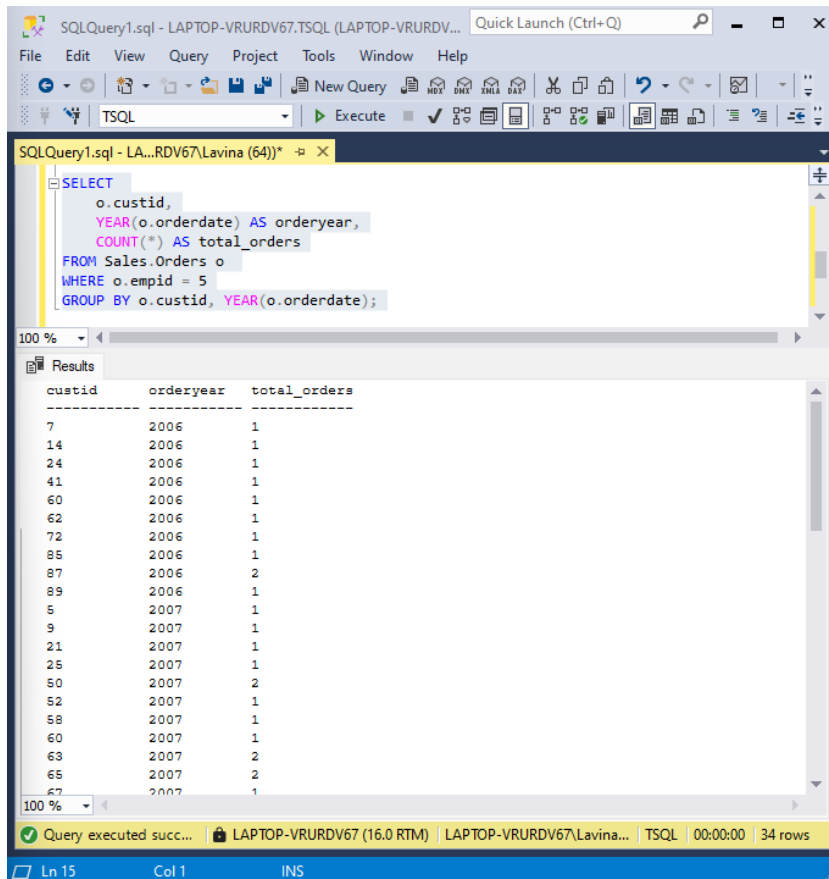
The results are displayed in a table with the following columns: custid, contactname, and city. The data is grouped by custid and contactname, with city listed for each group.

custid	contactname	city
5	Higginbotham, Tom	Luleå
7	Bansal, Dushyant	Strasbourg
9	Raghav, Amritansh	Marseille
14	Jelitto, Jacek	Bern
21	Russo, Giuseppe	Sao Paulo
24	San Juan, Patricia	Bräcke
25	Carlson, Jason	München
30	Shabalin, Rostislav	Sevilla
34	Cohen, Shy	Rio de Janeiro
41	Litton, Tim	Toulouse
46	Dressler, Marlies	Barquisimeto
47	Lupu, Cornel	I. de Margarita
50	Mace, Donald	Bruxelles
52	Dupont-Roc, Patrice	Leipzig
58	Fakhouri, Fadi	México D.F.
60	Uppal, Sunil	Lisboa
62	Misieć, Anna	Sao Paulo
63	Veronesi, Giorgio	Cunevalde
65	Moore, Michael	Albuquerque
66	Voss, Florian	Reggio Emilia
67	Garden, Euan	Rio de Janeiro
71	Navarro, Tomás	Boise

The status bar at the bottom indicates: Query executed succ... | LAPTOP-VRURDV67 (16.0 RTM) | LAPTOP-VRURDV67\Lavina... | TSQL | 00:00:00 | 29 rows

Untuk memperbaikinya perlu menambahkan c.city dalam klausa GROUP BY nya juga, agar data city dikelompokkan juga.

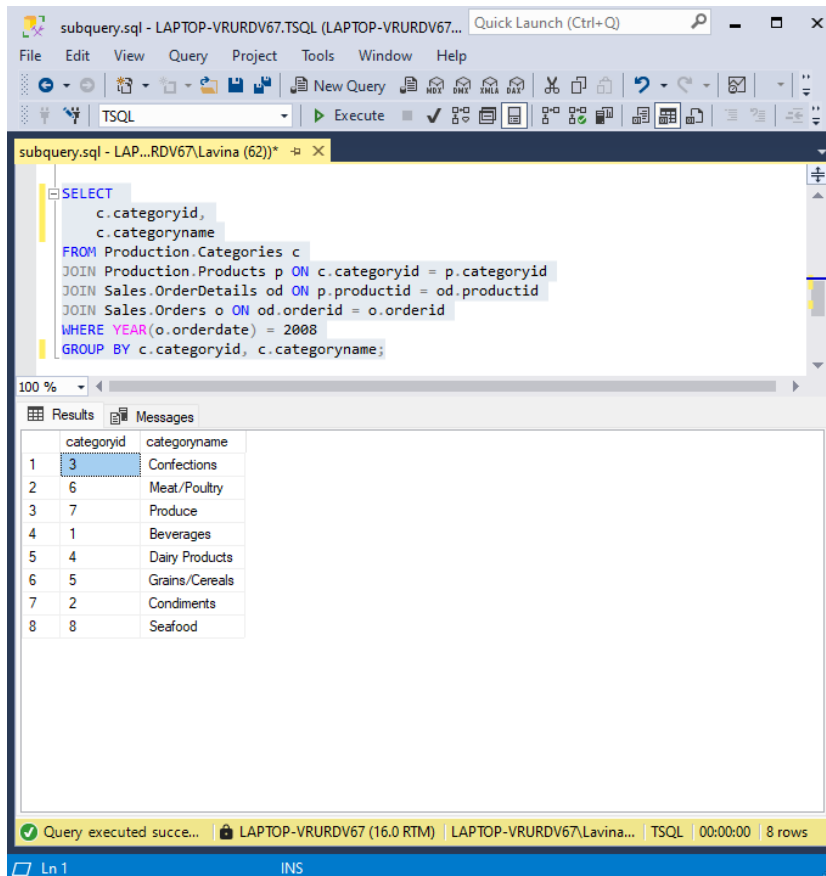
[Soal-5] Tuliskan pernyataan SELECT yang akan menampilkan kelompok baris berdasarkan kolom custid dan akan dihitung oleh kolom orderyear mewakili tahun pesanan berdasarkan kolom orderdate dari tabel Sales.Orders. Kemudian filter hasilnya untuk memasukkan hanya pesanan dari karyawan penjualan yang empid nya sama dengan 5!



Kode diatas menampilkan data customer, tahun pesanan dan total pesanan dari customer yang dilayani oleh karyawan dengan empid = 5. Penjelasan sintaks :

- YEAR(o.orderdate) digunakan untuk mengambil tahun dari kolom orderdate.
- COUNT(*) AS total_orders menghitung jumlah total pesanan untuk setiap custid dan orderyear.
- WHERE o.empid = 5 memfilter hanya pesanan dari karyawan penjualan dengan empid = 5.
- GROUP BY o.custid, YEAR(o.orderdate) mengelompokkan hasil berdasarkan custid dan orderyear.

[Soal-6] Tuliskan pernyataan SELECT yang akan mengembalikan kelompok baris berdasarkan kolom categoryname di tabel Production.Categories. Kemudian filter hasilnya hanya untuk product categories yang dipesan pada tahun 2008!



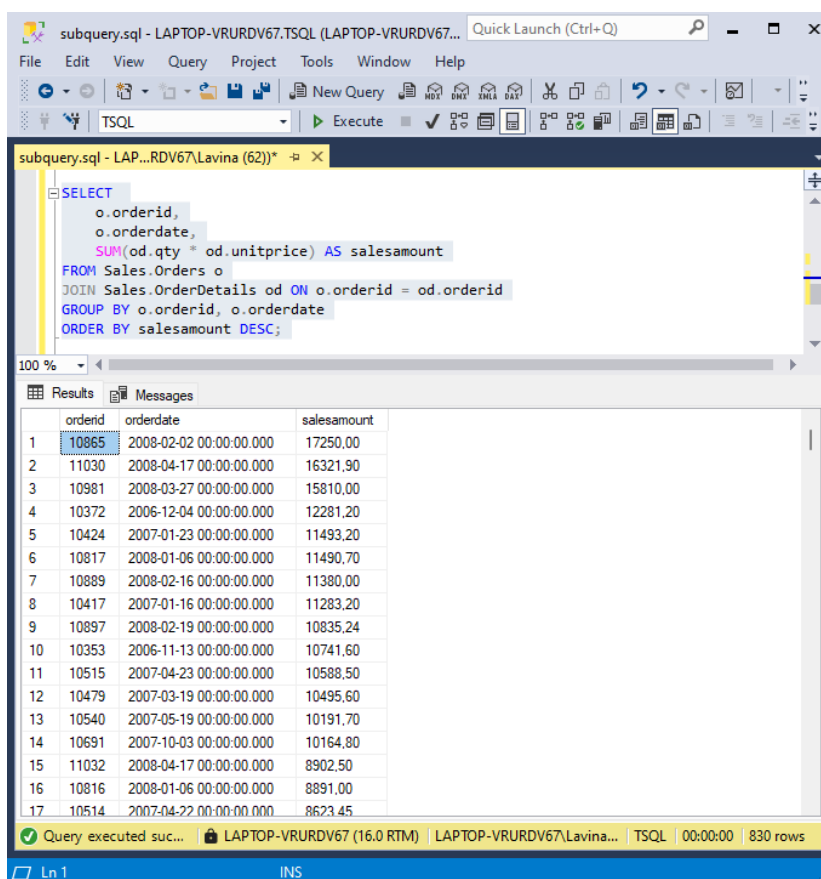
Penjelasan: Query diatas menampilkan data kategori dari produk-produk yang terjual pada tahun 2008. Pada query tersebut terdapat alur JOIN yang dimulai dari table Production.Products yang memiliki categoryid lalu di-JOIN dengan tabel Production.Categories. Untuk mendapatkan informasi terkait produk yang dipesan tabel Production.Products JOIN dengan tabel Sales.OrdersDetail melalui productid, terakhir untuk mendapatkan informasi pemesanannya dilakukan JOIN antara tabel Sales.OrderDetails dengan tabel Sales.Orders melalui orderid. Berikut penjelasan lebih lanjut pada tiap sintaksnya :

- WHERE YEAR(o.orderdate) = 2008 memfilter untuk menampilkan data pesanan pada tahun 2008 saja.
- GROUP BY c.categoryid, c.categoryname mengelompokkan hasil berdasarkan kolom id kategori dan nama kategori.

Praktikum – Bagian 2: Menulis Query Menggunakan Fungsi Agregasi

Skenario : Bagian pemasaran ingin meluncurkan kampanye baru, sehingga karyawan perlu mendapatkan wawasan yang lebih baik mengenai perilaku pembelian para pelanggan. Oleh karena itu, harus dibuat laporan penjualan yang berbeda yang didasarkan pada jumlah penjualan rata-rata per tahun per pelanggan.

[Soal-7] Tuliskan pernyataan SELECT yang akan mengembalikan kolom orderid, orderdate dari tabel Sales.Orders dan total sales amount per orderid (Petunjuk : Kalikan kolom qty dan unitprice dari tabel Sales.OderDetails) Gunakan alias salesamount untuk kolom yang dihitung. Kemudian urutkan hasilnya dengan total sales amount dalam urutan menurun!



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results grid. The query is as follows:

```
SELECT
    o.orderid,
    o.orderdate,
    SUM(od.qty * od.unitprice) AS salesamount
FROM Sales.Orders o
JOIN Sales.OrderDetails od ON o.orderid = od.orderid
GROUP BY o.orderid, o.orderdate
ORDER BY salesamount DESC;
```

The results grid displays the following data:

	orderid	orderdate	salesamount
1	10865	2008-02-02 00:00:00.000	17250,00
2	11030	2008-04-17 00:00:00.000	16321,90
3	10981	2008-03-27 00:00:00.000	15810,00
4	10372	2006-12-04 00:00:00.000	12281,20
5	10424	2007-01-23 00:00:00.000	11493,20
6	10817	2008-01-06 00:00:00.000	11490,70
7	10889	2008-02-16 00:00:00.000	11380,00
8	10417	2007-01-16 00:00:00.000	11283,20
9	10897	2008-02-19 00:00:00.000	10835,24
10	10353	2006-11-13 00:00:00.000	10741,60
11	10515	2007-04-23 00:00:00.000	10588,50
12	10479	2007-03-19 00:00:00.000	10495,60
13	10540	2007-05-19 00:00:00.000	10191,70
14	10691	2007-10-03 00:00:00.000	10164,80
15	11032	2008-04-17 00:00:00.000	8902,50
16	10816	2008-01-06 00:00:00.000	8891,00
17	10514	2007-04-22 00:00:00.000	8623,45

The status bar at the bottom indicates: Query executed suc... | LAPTOP-VRURDV67 (16.0 RTM) | LAPTOP-VRURDV67\Lavina... | TSQL | 00:00:00 | 830 rows

Penjelasan: Query bertujuan untuk mendapatkan daftar pesanan (orderid, orderdate) dengan total penjualan (salesamount) untuk setiap pesanannya, lalu diurutkan berdasarkan total penjualan terbesar. Berikut penjelasan lebih detail untuk setiap sintaksnya :

- SUM(od.qty * od.unitprice): Berfungsi untuk menjumlahkan hasil dari qty * unitprice untuk setiap baris dalam kelompok (yaitu setiap orderid dan orderdate).

- GROUP BY o.orderid, o.orderdate: Mengelompokkan pesanan berdasarkan orderid dan orderdate agar setiap pesanan dihitung secara terpisah.
- ORDER BY salesamount DESC: Mengurutkan hasil berdasarkan total penjualan (salesamount) secara menurun, sehingga pesanan dengan penjualan tertinggi akan muncul terlebih dahulu.

[Soal-8] Salin pernyataan T-SQL pada jawaban soal-7 dan modifikasi dengan memasukkan jumlah order lines untuk setiap order dan nilai rata-rata sales amount per orderid sesuai pesanan. Gunakan nama alias masing-masing nooforderlines dan avgsalesamountperorderlines!

The screenshot shows a SQL query window with the following T-SQL code:

```

SELECT
    o.orderid,
    o.orderdate,
    SUM(od.qty * od.unitprice) AS salesamount,
    COUNT(od.orderid) AS nooforderlines,
    AVG(od.qty * od.unitprice) AS avgsalesamountperorderlines
FROM Sales.Orders o
JOIN Sales.OrderDetails od ON o.orderid = od.orderid
GROUP BY o.orderid, o.orderdate
ORDER BY salesamount DESC;

```

Below the query, the 'Results' tab displays the following data:

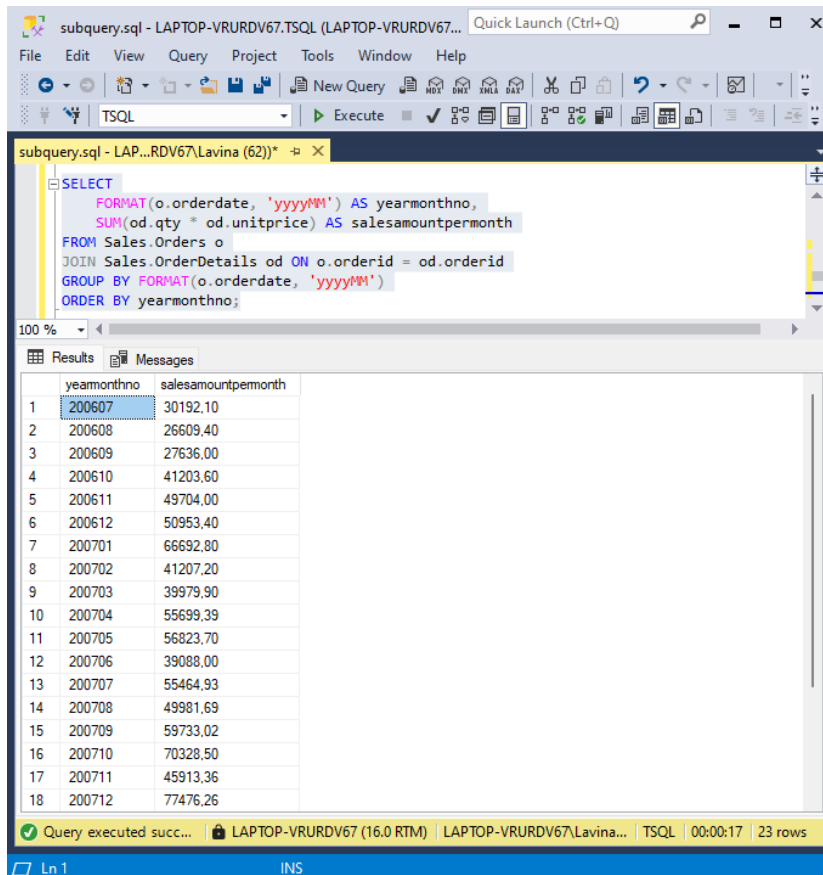
orderid	orderdate	salesamount	nooforderlines	avgsalesamountperorderlines
10865	2008-02-02 00:00:00.000	17250,00	2	8625,00
11030	2008-04-17 00:00:00.000	16321,90	4	4080,475
10981	2008-03-27 00:00:00.000	15810,00	1	15810,00
10372	2006-12-04 00:00:00.000	12281,20	4	3070,30
10424	2007-01-23 00:00:00.000	11493,20	3	3831,0666
10817	2008-01-06 00:00:00.000	11490,70	4	2872,675
10889	2008-02-16 00:00:00.000	11380,00	2	5690,00
10417	2007-01-16 00:00:00.000	11283,20	4	2820,80
10897	2008-02-19 00:00:00.000	10835,24	2	5417,62
10353	2006-11-13 00:00:00.000	10741,60	2	5370,80
10515	2007-04-23 00:00:00.000	10588,50	5	2117,70
10479	2007-03-19 00:00:00.000	10495,60	4	2623,90
10540	2007-05-19 00:00:00.000	10191,70	4	2547,925
10691	2007-10-03 00:00:00.000	10164,80	5	2032,96
11032	2008-04-17 00:00:00.000	8902,50	3	2967,50
10816	2008-01-06 00:00:00.000	8881,00	2	4440,50

The status bar at the bottom indicates: Query executed suc... LAPTOP-VRURDV67 (16.0 RTM) | LAPTOP-VRURDV67\Lavina... | TSQL | 00:00:00 | 830 rows

Penjelasan: Query diatas menggunakan beberapa fungsi aggregate, yaitu :

- SUM(od.qty * od.unitprice): Menjumlahkan total sales amount untuk setiap orderid.
- COUNT(od.orderid) akan memberikan jumlah order lines untuk setiap orderid. Ini akan dihitung berdasarkan jumlah baris detail pesanan (Sales.OrderDetails).
- AVG(od.qty * od.unitprice) akan memberikan nilai rata-rata sales amount per order line.

[Soal-9] Tuliskan pernyataan SELECT untuk mengambil jumlah penjualan total untuk setiap bulannya! Penggunaan klausa SELECT seharusnya menyertakan perhitungan kolom yearmonthno (notasi YYYYMM) berdasarkan kolom orderdate pada tabel Sales.Orders dan total jumlah penjualan (Perkalian kolom qty dengan unitprice dari tabel Sales.OrderDetails) yang diberi alias salesamountpermonth. Urutan hasilnya didasarkan pada perhitungan kolom yearmonthno.



```
SELECT
    FORMAT(o.orderdate, 'yyyyMM') AS yearmonthno,
    SUM(od.qty * od.unitprice) AS salesamountpermonth
FROM Sales.Orders o
JOIN Sales.OrderDetails od ON o.orderid = od.orderid
GROUP BY FORMAT(o.orderdate, 'yyyyMM')
ORDER BY yearmonthno;
```

	yearmonthno	salesamountpermonth
1	200607	30192.10
2	200608	26609.40
3	200609	27636.00
4	200610	41203.60
5	200611	49704.00
6	200612	50953.40
7	200701	66692.80
8	200702	41207.20
9	200703	39979.90
10	200704	55699.39
11	200705	56823.70
12	200706	39088.00
13	200707	55464.93
14	200708	49981.69
15	200709	59733.02
16	200710	70328.50
17	200711	45913.36
18	200712	77476.26

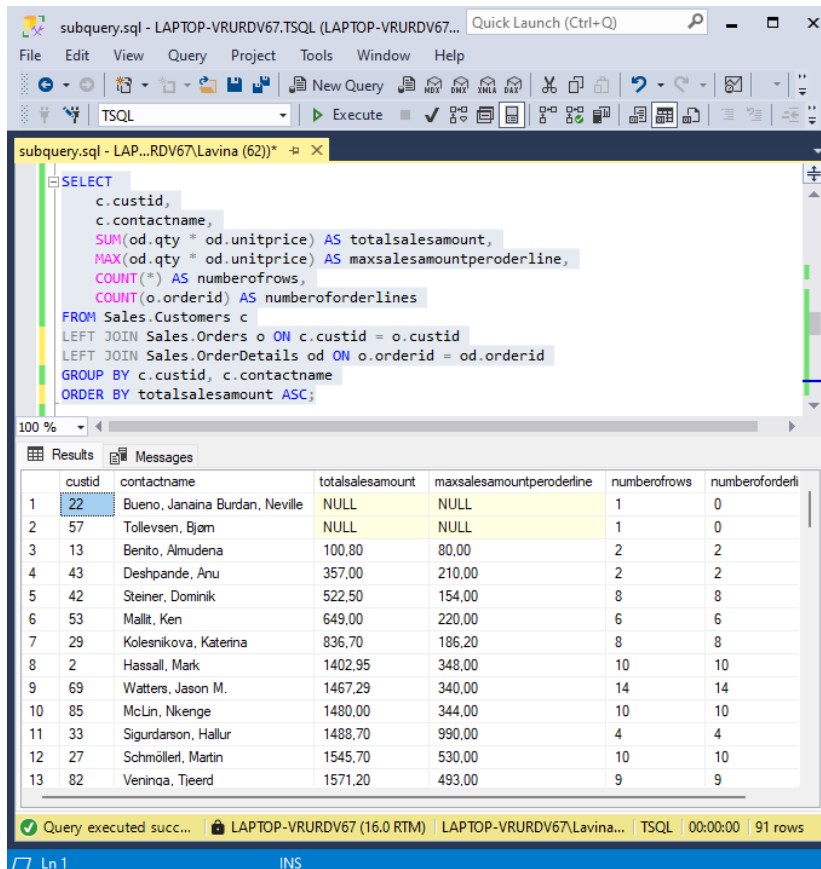
Query executed succ... | LAPTOP-VRURDV67 (16.0 RTM) | LAPTOP-VRURDV67\Lavina... | TSQL | 00:00:17 | 23 rows

Penjelasan:

- `FORMAT(o.orderdate, 'yyyyMM')`: Menghasilkan nilai YYYYMM dari kolom orderdate. Fungsi FORMAT digunakan untuk mengonversi tanggal ke dalam format tahun-bulan.
- `SUM(od.qty * od.unitprice)`: Menjumlahkan total nilai penjualan (`qty * unitprice`) dari tabel Sales.OrderDetails untuk setiap bulan.
- `GROUP BY FORMAT(o.orderdate, 'yyyyMM')`: Mengelompokkan hasil berdasarkan tahun dan bulan (`yearmonthno`).
- `ORDER BY yearmonthno`: Mengurutkan hasil berdasarkan kolom `yearmonthno` dalam urutan naik.

[Soal-10] Tulislah perintah SELECT yang akan mengambil semua pelanggan/customer (termasuk yang tidak memiliki pesanan) dan jumlah penjualan, jumlah pesanan maksimum per baris, dan jumlah pesanan! Klausa SELECT harus memasukkan kolom custid dan contactname dari tabel Sales.Customers dan 4 (empat) kolom yang dikalkulasi berdasarkan fungsi agregasi sebagai berikut :

1. totalsalesamount, adalah alias untuk jumlah penjualan total per pesanan
2. maxsalesamountperorderline, adalah alias untuk jumlah penjualan maksimum per baris pesanan
3. numberofrows, adalah alias untuk jumlah baris (gunakan * dalam fungsi COUNT)
4. numberoforderlines, adalah alias untuk jumlah baris pesanan (gunakan kolom orderid di kolom fungsi COUNT) Urutkan hasilnya berdasarkan kolom totalsalesamount.



```

SELECT
    c.custid,
    c.contactname,
    SUM(od.qty * od.unitprice) AS totalsalesamount,
    MAX(od.qty * od.unitprice) AS maxsalesamountperorderline,
    COUNT(*) AS numberofrows,
    COUNT(o.orderid) AS numberoforderlines
FROM Sales.Customers c
LEFT JOIN Sales.Orders o ON c.custid = o.custid
LEFT JOIN Sales.OrderDetails od ON o.orderid = od.orderid
GROUP BY c.custid, c.contactname
ORDER BY totalsalesamount ASC;

```

	custid	contactname	totalsalesamount	maxsalesamountperorderline	numberofrows	numberoforderlines
1	22	Bueno, Janaina Burdan, Neville	NULL	NULL	1	0
2	57	Tollefsen, Bjørn	NULL	NULL	1	0
3	13	Benito, Almudena	100,80	80,00	2	2
4	43	Deshpande, Anu	357,00	210,00	2	2
5	42	Steiner, Dominik	522,50	154,00	8	8
6	53	Mallit, Ken	649,00	220,00	6	6
7	29	Kolesnikova, Katerina	836,70	186,20	8	8
8	2	Hassall, Mark	1402,95	348,00	10	10
9	69	Watters, Jason M.	1467,29	340,00	14	14
10	85	McLin, Nkenge	1480,00	344,00	10	10
11	33	Sigurdson, Hallur	1488,70	990,00	4	4
12	27	Schmöller, Martin	1545,70	530,00	10	10
13	82	Veninqa, Tjeerd	1571,20	493,00	9	9

Query executed succ... | LAPTOP-VRURDV67 (16.0 RTM) | LAPTOP-VRURDV67\Lavina... | TSQL | 00:00:00 | 91 rows

Penjelasan: Pada query diatas menggunakan **LEFT JOIN** yang berguna untuk memastikan semua pelanggan ditampilkan, termasuk yang tidak memiliki pesanan.

- Sales.Customers dihubungkan dengan Sales.Orders berdasarkan custid.
- Sales.Orders dihubungkan dengan Sales.OrderDetails berdasarkan orderid.

Berikut penjelasan lebih detail untuk tiap sintaksnya :

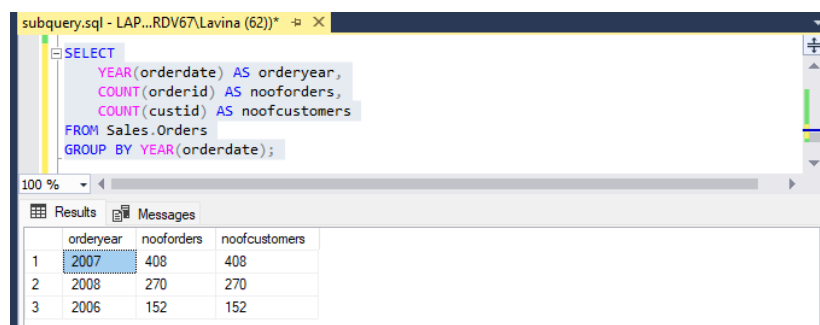
- SUM(od.qty * od.unitprice) AS totalsalesamount: Menghitung total penjualan dari setiap pelanggan, dengan menghitung jumlah dari kuantitas dan harga unit (qty * unitprice) untuk semua pesanan pelanggan tersebut.
- MAX(od.qty * od.unitprice) AS maxsalesamountperorderline: Mengambil jumlah penjualan maksimum untuk satu baris pesanan (qty * unitprice).
- COUNT(*) AS numberofrows: Menghitung jumlah total baris pesanan, baik yang memiliki pesanan maupun tidak.
- COUNT(o.orderid) AS numberoforderlines: Menghitung jumlah baris pesanan berdasarkan kolom orderid.
- ORDER BY totalsalesamount ASC: Mengurutkan hasil berdasarkan total penjualan (totalsalesamount), dari yang terendah.

Praktikum – Bagian 3: Menulis Query Menggunakan Fungsi Agregasi Distinct

Skenario : Departemen pemasaran ingin memiliki beberapa laporan tambahan yang menunjukkan jumlah pelanggan yang memiliki pemesanan dalam jangka waktu tertentu dan jumlah pelanggan berdasarkan huruf pertama dan contact name.

[Soal-11] Berdasarkan hasil eksekusi T-SQL di bawah ini, Kenapa jumlah pesanan (nooforders) sama dengan jumlah pelanggan (noofcustomers)?

```
SELECT
YEAR(orderdate) AS orderyear,
COUNT(orderid) AS nooforders,
COUNT(custid) AS noofcustomers
FROM Sales.Orders
GROUP BY YEAR(orderdate);
```

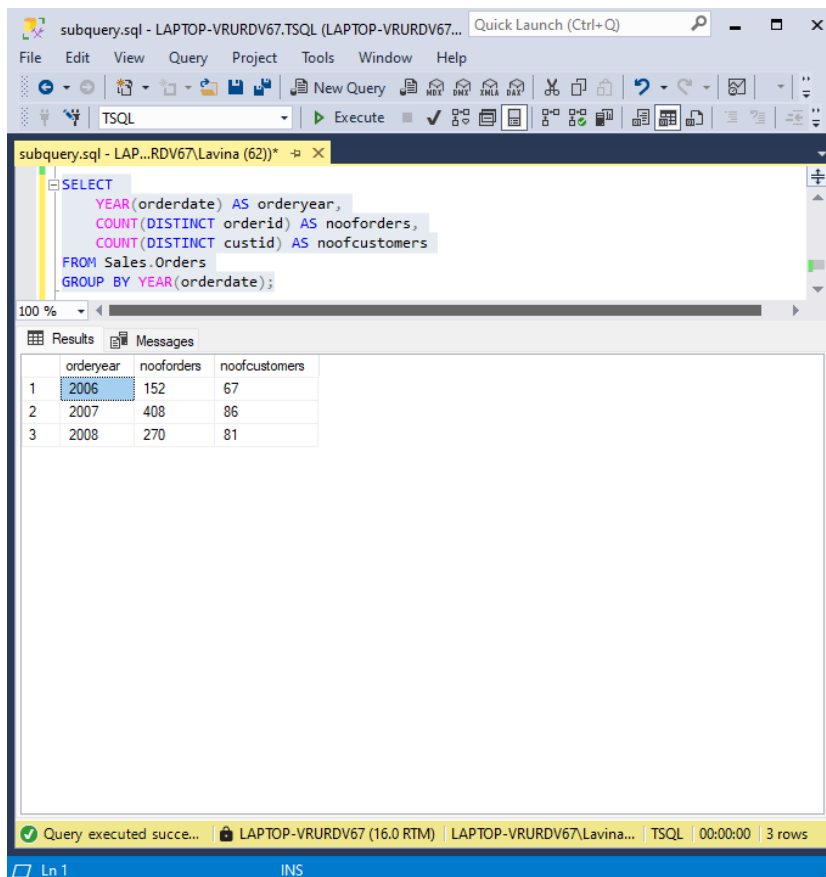


	orderyear	nooforders	noofcustomers
1	2007	408	408
2	2008	270	270
3	2006	152	152

Penjelasan: Jumlah pesanan (nooforders) dan jumlah pelanggan (noofcustomers) sama dalam query yang ditampilkan karena menggunakan COUNT(custid) untuk menghitung jumlah

pelanggan. Namun, COUNT() hanya menghitung jumlah nilai yang tidak NULL dalam kolom yang diberikan. Dalam konteks ini, kolom custid hampir selalu diisi dalam tabel Sales.Orders, sehingga jumlah pesanan dan jumlah pelanggan yang dihitung oleh query tersebut akan sama jika setiap pesanan memiliki custid.

[Soal-12] Perbaiki T-SQL pada soal-12 untuk menunjukkan jumlah pelanggan yang benar yang melakukan pemesanan setiap tahun!



Untuk memperbaikinya bisa menggunakan DISTINCT pada fungsi COUNT untuk meng-agregasikan nilai yang unik saja. Melakukan agregasi dengan DISTINCT hanya menghilangkan nilai yang sama, BUKAN baris yang sama (tidak seperti SELECT DISTINCT).

[Soal-13] Tuliskan pernyataan SELECT untuk mengambil jumlah pelanggan berdasarkan huruf pertama dari nilai pada kolom contactname dari tabel Sales.Customers. Tambahkan kolom yang menunjukkan jumlah pesanan yang dilakukan setiap grup pelanggan/customer. Gunakan alias masing-masing firstletter, noofcustomers dan nooforders. Urutkan hasilnya berdasarkan kolom firstletter!

The screenshot shows a SQL Server Enterprise Manager window with a query executed. The query is as follows:

```

SELECT
    LEFT(c.contactname, 1) AS firstletter,
    COUNT(DISTINCT c.custid) AS noofcustomers,
    COUNT(DISTINCT o.orderid) AS nooforders
FROM Sales.Customers c
LEFT JOIN Sales.Orders o ON c.custid = o.custid
GROUP BY LEFT(c.contactname, 1)
ORDER BY firstletter;

```

The results are displayed in a table with the following data:

	firstletter	noofcustomers	nooforders
1	A	2	19
2	B	6	37
3	C	6	72
4	D	5	35
5	E	1	10
6	F	3	26
7	G	6	51
8	H	2	22
9	I	1	3
10	J	3	24
11	K	5	54
12	L	8	100
13	M	8	76
14	N	2	37
15	O	2	8
16	P	2	14
17	R	5	39

The status bar at the bottom indicates: Query executed succ..., LAPTOP-VRURDV67 (16.0 RTM), LAPTOP-VRURDV67\Lavina..., TSQL, 00:00:00, 23 rows.

Penjelasan:

- LEFT(c.contactname, 1) AS firstletter: Mengambil huruf pertama dari kolom contactname dan memberikan alias firstletter.
- COUNT(c.custid) AS noofcustomers: Menghitung jumlah pelanggan (custid) untuk setiap huruf pertama (grup pelanggan).
- COUNT(o.orderid) AS nooforders: Menghitung jumlah pesanan yang dilakukan oleh setiap grup pelanggan berdasarkan huruf pertama dari contactname. Menggunakan LEFT JOIN agar pelanggan yang tidak memiliki pesanan juga ditampilkan, dan jumlah pesanan mereka akan bernilai 0.
- GROUP BY LEFT(c.contactname, 1): Mengelompokkan hasil berdasarkan huruf pertama dari contactname.
- ORDER BY firstletter: Mengurutkan hasil berdasarkan huruf pertama (firstletter) dari contactname

[Soal-14] Salin T-SQL pada jawaban soal-6 kemudian modifikasi dengan memasukkan informasi tentang setiap category produk : jumlah penjualan, jumlah pesanan, dan jumlah penjualan rata-rata setiap pemesanan. Gunakan nama alias masing-masing, nooforders, dan avgsalesamountperorder.

The screenshot shows a SQL Server Enterprise Manager window with a T-SQL query and its results. The query is as follows:

```

SELECT
    c.categoryid,
    c.categoryname,
    SUM(od.qty * od.unitprice) AS totalsalesamount,
    COUNT(DISTINCT o.orderid) AS nooforders,
    SUM(od.qty * od.unitprice) / COUNT(DISTINCT o.orderid) AS avgsalesamountperorder
FROM Production.Categories c
JOIN Production.Products p ON c.categoryid = p.categoryid
JOIN Sales.OrderDetails od ON p.productid = od.productid
JOIN Sales.Orders o ON od.orderid = o.orderid
WHERE YEAR(o.orderdate) = 2008
GROUP BY c.categoryid, c.categoryname
ORDER BY c.categoryid;

```

The results are displayed in a table with the following columns: categoryid, categoryname, totalsalesamount, nooforders, and avgsalesamountperorder. The table contains 8 rows of data.

categoryid	categoryname	totalsalesamount	nooforders	avgsalesamountperorder
1	Beverages	122223,75	128	954,873
2	Condiments	34557,45	62	557,3782
3	Confections	58359,73	89	655,7273
4	Dairy Products	82803,90	90	920,0433
5	Grains/Cereals	30422,25	55	553,1318
6	Meat/Poultry	60275,57	43	1401,7574
7	Produce	32415,85	42	771,8059
8	Seafood	48712,84	101	482,3053

Penjelasan:

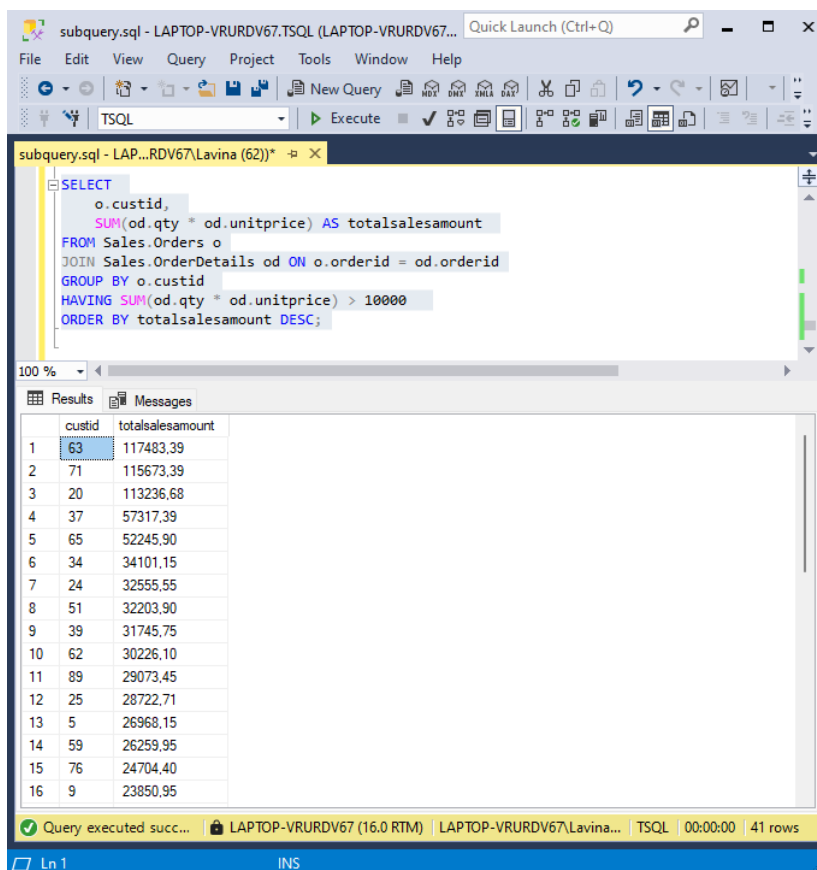
- **COUNT(DISTINCT o.orderid):** Ini menghitung jumlah pesanan unik dalam satu kategori (bukan per baris pesanan).
- **SUM(od.qty * od.unitprice) / COUNT(DISTINCT o.orderid):** Ini membagi total penjualan dalam satu kategori dengan jumlah pesanan unik untuk mendapatkan rata-rata total penjualan per pesanan.

Dengan query tersebut bisa menghitung total penjualan dan jumlah pesanan terlebih dahulu, lalu mendapatkan rata-rata dengan cara membagi dua nilai tersebut.

Praktikum – Bagian 4: Menulis Query Yang Melakukan Filter Group Dengan Klausu HAVING

Skenario : Laporan tentang analisis tingkah laku pelanggan yang telah dibuat pada percobaan sebelumnya, telah memenuhi kebutuhan departemen penjualan dan pemasaran. Sekarang departemen tersebut membutuhkan laporan tersebut difilter berdasarkan total jumlah penjualan dan jumlah pesanan. Jadi skenario bagian ini akan membahas tata cara filter hasil uji coba sebelumnya berdasarkan fungsi agregasi dan mempelajari penggunaan klausa WHERE dan HAVING.

[Soal-15] Tuliskan perintah T-SQL dengan klausa SELECT untuk mengambil 5 pelanggan teratas dengan penjualan total lebih dari \$10.000. Tampilkan kolom custid dari tabel order dan hitung kolom yang berisi jumlah penjualan berdasarkan kolom qty dan unitprice dari tabel Sales.OrderDetails. Gunakan alias totalsalesamount.



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a T-SQL query in the 'subquery.sql' file. The query is as follows:

```
SELECT
    o.custid,
    SUM(od.qty * od.unitprice) AS totalsalesamount
FROM Sales.Orders o
JOIN Sales.OrderDetails od ON o.orderid = od.orderid
GROUP BY o.custid
HAVING SUM(od.qty * od.unitprice) > 10000
ORDER BY totalsalesamount DESC;
```

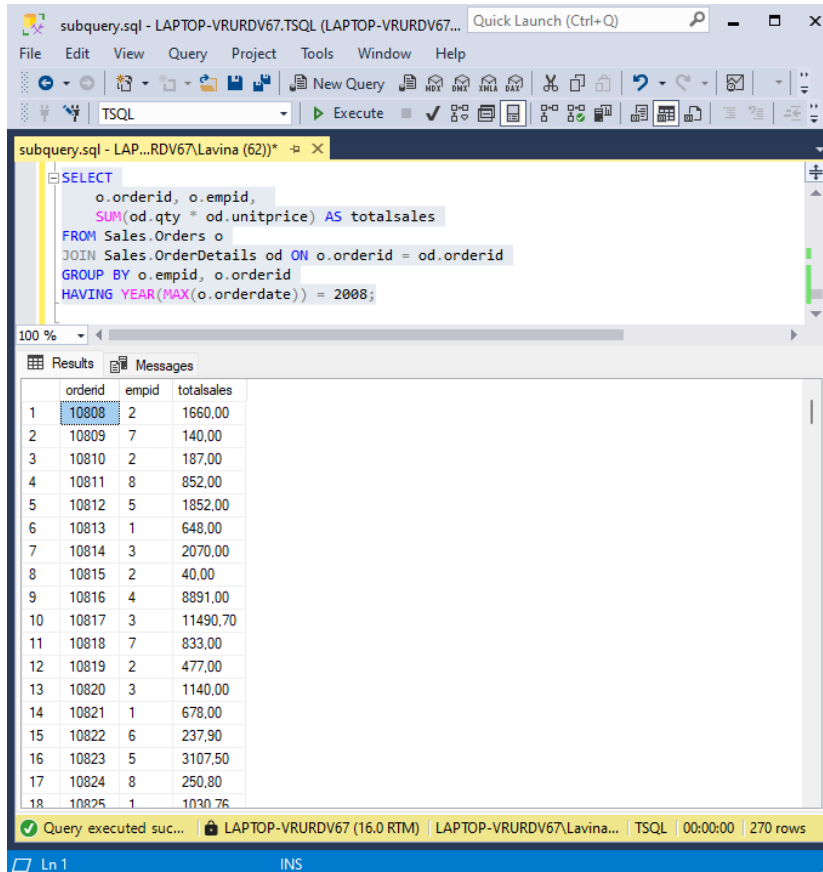
The bottom pane shows the 'Results' tab with a table containing 16 rows of data. The columns are 'custid' and 'totalsalesamount'. The data is sorted in descending order of 'totalsalesamount'.

	custid	totalsalesamount
1	63	117483,39
2	71	115673,39
3	20	113236,68
4	37	57317,39
5	65	52245,90
6	34	34101,15
7	24	32555,55
8	51	32203,90
9	39	31745,75
10	62	30226,10
11	89	29073,45
12	25	28722,71
13	5	26968,15
14	59	26259,95
15	76	24704,40
16	9	23850,95

The status bar at the bottom indicates 'Query executed succ...' and '41 rows'.

Penjelasan: Query diatas menggunakan fungsi HAVING SUM(od.qty * od.unitprice) > 10000 yang berfungsi untuk memfilter hasil untuk hanya menampilkan pelanggan dengan total penjualan lebih dari \$10.000.

[Soal-16] Tuliskan perintah T-SQL dengan klausa SELECT untuk mengambil kolom empid,orderid dan kolom yang mempresentasikan perhitungan total penjualan (total sales amount) berdasarkan tabel Sales.Orders dan Sales.OrderDetails. Filter hasilnya menjadi grup baris data hanya untuk pesanan di tahun 2008!



The screenshot shows a SQL Server Enterprise Manager window with a T-SQL query editor and a results grid. The query is as follows:

```
SELECT
    o.orderid, o.empid,
    SUM(od.qty * od.unitprice) AS totalsales
FROM Sales.Orders o
JOIN Sales.OrderDetails od ON o.orderid = od.orderid
GROUP BY o.empid, o.orderid
HAVING YEAR(MAX(o.orderdate)) = 2008;
```

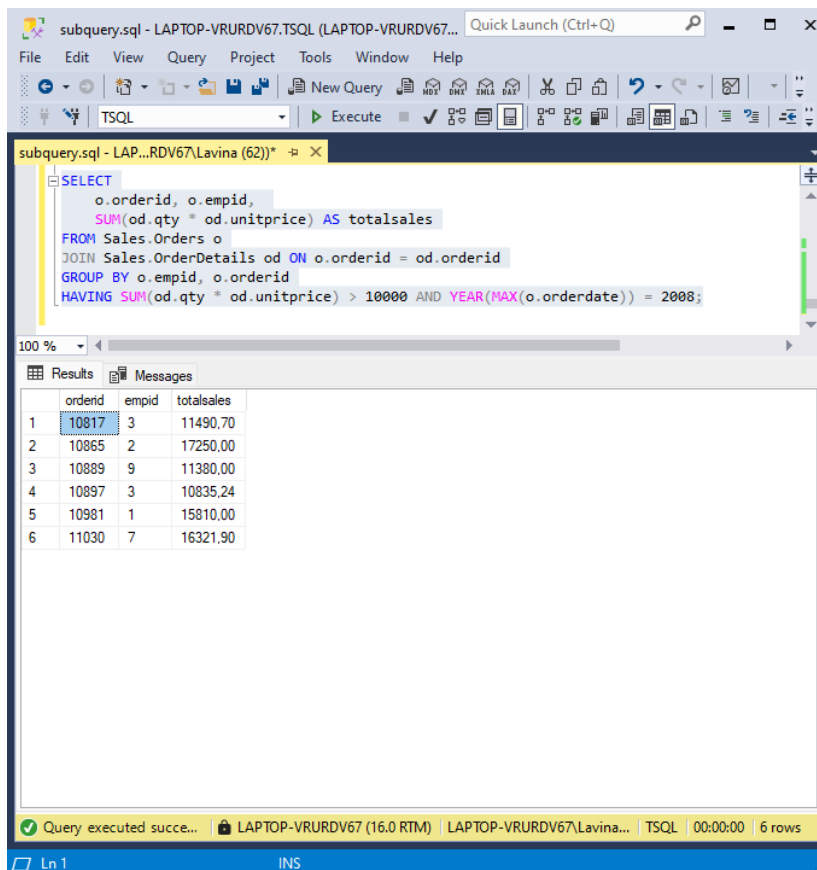
The results grid displays 18 rows of data with columns orderid, empid, and totalsales. The status bar at the bottom indicates the query was executed successfully, returning 270 rows.

	orderid	empid	totalsales
1	10808	2	1660,00
2	10809	7	140,00
3	10810	2	187,00
4	10811	8	852,00
5	10812	5	1852,00
6	10813	1	648,00
7	10814	3	2070,00
8	10815	2	40,00
9	10816	4	8891,00
10	10817	3	11490,70
11	10818	7	833,00
12	10819	2	477,00
13	10820	3	1140,00
14	10821	1	678,00
15	10822	6	237,90
16	10823	5	3107,50
17	10824	8	250,80
18	10825	1	1030,76

Penjelasan:

- GROUP BY o.empid, o.orderid digunakan untuk mengelompokkan hasil berdasarkan empid dan orderid.
- HAVING YEAR(MAX(o.orderdate)) = 2008 digunakan untuk memfilter grup yang memiliki tanggal pemesanan dalam tahun 2008. Di sini, MAX(o.orderdate) digunakan untuk memastikan kita mendapatkan tahun dari tanggal pesanan yang relevan dalam setiap grup.

[Soal-17] Salin perintah T-SQL jawaban soal-16 dan modifikasi untuk menambahkan filter yang hanya mengambil baris yang memiliki jumlah penjualan lebih dari \$10.000!



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query is as follows:

```
SELECT
    o.orderid, o.empid,
    SUM(od.qty * od.unitprice) AS totalsales
FROM Sales.Orders o
JOIN Sales.OrderDetails od ON o.orderid = od.orderid
GROUP BY o.empid, o.orderid
HAVING SUM(od.qty * od.unitprice) > 10000 AND YEAR(MAX(o.orderdate)) = 2008;
```

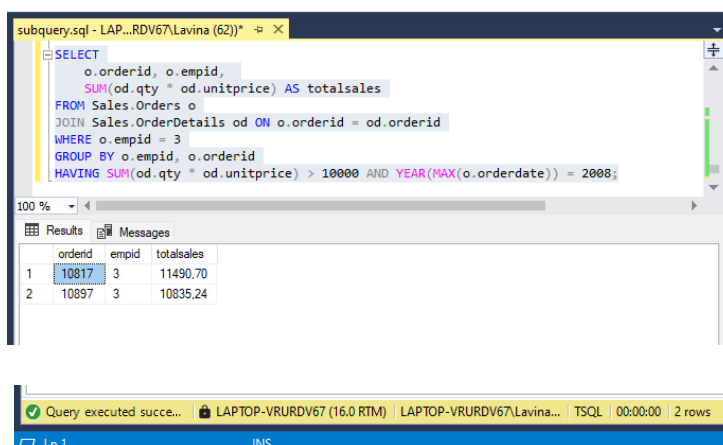
The results pane shows 6 rows of data:

	orderid	empid	totalsales
1	10817	3	11490,70
2	10865	2	17250,00
3	10889	9	11380,00
4	10897	3	10835,24
5	10981	1	15810,00
6	11030	7	16321,90

The status bar at the bottom indicates "Query executed succe...", "LAPTOP-VRURDV67 (16.0 RTM)", "LAPTOP-VRURDV67\Lavina...", "TSQL", "00:00:00", and "6 rows".

Untuk memfilter penjualan diatas \$10.000 hanya perlu menggunakan operator AND dimana kondisi pertama untuk memfilter penjualan diatas \$10.000 dan kondisi kedua untuk memfilter pesanan pada tahun 2008.

[Soal-18] Salin perintah T-SQL jawaban soal-17 dan modifikasi untuk menambahkan filter yang hanya menampilkan pegawai dengan empid sama dengan 3(tiga)!



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query is as follows:

```
SELECT
    o.orderid, o.empid,
    SUM(od.qty * od.unitprice) AS totalsales
FROM Sales.Orders o
JOIN Sales.OrderDetails od ON o.orderid = od.orderid
WHERE o.empid = 3
GROUP BY o.empid, o.orderid
HAVING SUM(od.qty * od.unitprice) > 10000 AND YEAR(MAX(o.orderdate)) = 2008;
```

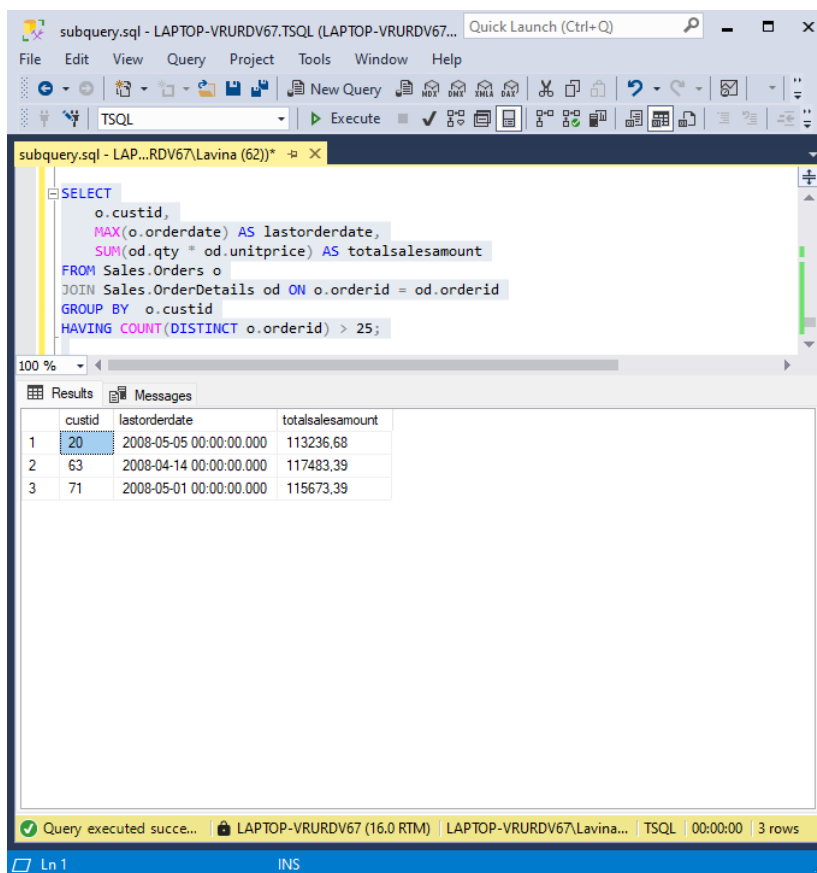
The results pane shows 2 rows of data:

	orderid	empid	totalsales
1	10817	3	11490,70
2	10897	3	10835,24

The status bar at the bottom indicates "Query executed succe...", "LAPTOP-VRURDV67 (16.0 RTM)", "LAPTOP-VRURDV67\Lavina...", "TSQL", "00:00:00", and "2 rows".

Untuk menampilkan data dengan pegawai empid = 3 hanya perlu menggunakan klausa WHERE o.empid = 3.

[Soal-19] Tuliskan perintah T-SQL dengan klausa SELECT untuk mengambil semua pelanggan yang memiliki lebih dari 25 order, dan tambahkan informasi mengenai tanggal pesanan terakhir dan jumlah penjualan. Tampilkan kolom custid dari tabel Sales.Orders table dan dua kolom perhitungan (lastorderdate berdasarkan kolom orderdate dan totalsalesamount berdasarkan kolom qty dan unitprice dari tabel Sales.OrderDetails!



The screenshot shows a SQL Server Enterprise Manager window with a T-SQL query editor and a results grid. The query is as follows:

```
SELECT
    o.custid,
    MAX(o.orderdate) AS lastorderdate,
    SUM(od.qty * od.unitprice) AS totalsalesamount
FROM Sales.Orders o
JOIN Sales.OrderDetails od ON o.orderid = od.orderid
GROUP BY o.custid
HAVING COUNT(DISTINCT o.orderid) > 25;
```

The results grid displays the following data:

	custid	lastorderdate	totalsalesamount
1	20	2008-05-05 00:00:00.000	113236.68
2	63	2008-04-14 00:00:00.000	117483.39
3	71	2008-05-01 00:00:00.000	115673.39

The status bar at the bottom indicates: Query executed succe... | LAPTOP-VRURDV67 (16.0 RTM) | LAPTOP-VRURDV67\Lavina... | TSQL | 00:00:00 | 3 rows

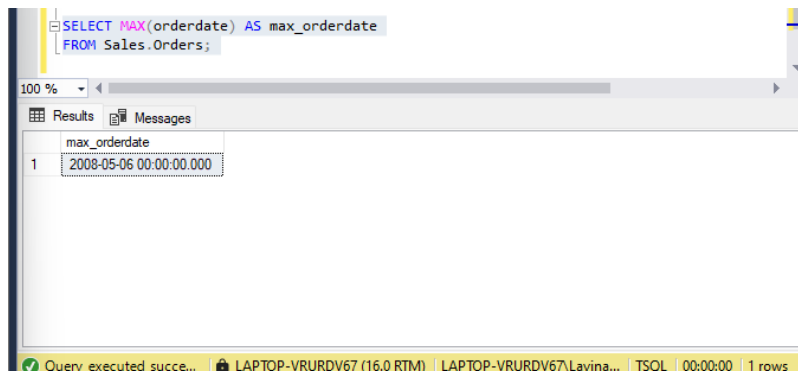
Penjelasan:

- MAX(o.orderdate): Mengambil tanggal pesanan terakhir untuk setiap pelanggan.
- HAVING COUNT(DISTINCT o.orderid) > 25: Memfilter hasil untuk hanya menampilkan pelanggan yang memiliki lebih dari 25 order.

Praktikum – Bagian 5: Menulis Query Menggunakan Self-Contained Sub-query

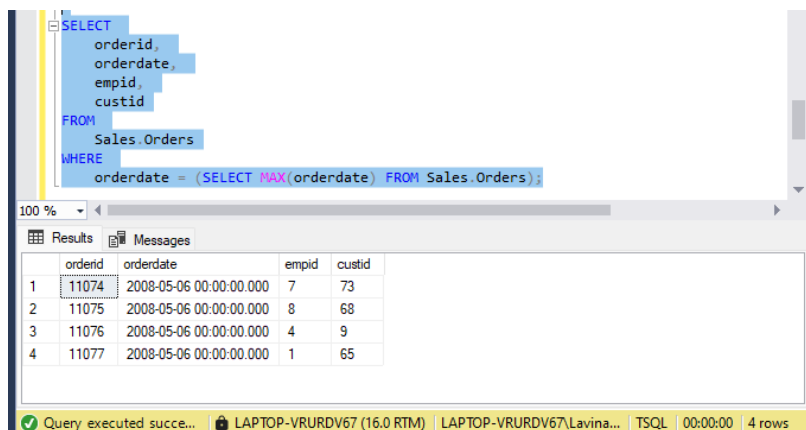
Skenario : Departemen penjualan memerlukan beberapa laporan lanjutan untuk menganalisis pesanan penjualan. Untuk itu dibutuhkan pernyataan SELECT yang menggunakan self-contained subquery.

[Soal-20] Tulislah pernyataan SELECT untuk menampilkan pemesanan orderdate maksimum dari tabel Sales.Orders.



Penjelasan: Query MAX(orderdate) digunakan untuk mendapatkan tanggal pemesanan terbesar (terakhir) dari kolom orderdate.

[Soal-21] Tulislah pernyataan SELECT untuk menampilkan kolomorderid, orderdate, empid, dan custid dari tabel Sales.Orders. Kemudian saring hasilnya dengan menyertakan hanya pesanan yang sesuai dengan waktu pesan paling akhir (Gunakan query pada jawaban soal-20 sebagai sub-query self-contained subquery)!

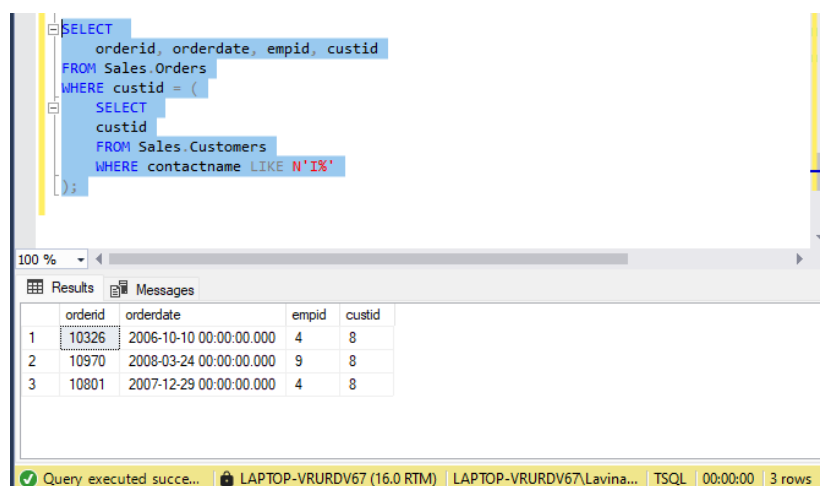


Penjelasan: WHERE orderdate = (SELECT MAX(orderdate) FROM Sales.Orders): Menggunakan subquery untuk mendapatkan tanggal pemesanan maksimum (MAX(orderdate)), dan

kemudian menyaring hasil utama dengan hanya menampilkan pesanan yang memiliki orderdate yang sama dengan nilai maksimum tersebut.

[Soal-22] Eksekusi T-SQL di bawah ini, kemudian modifikasi dengan filter pelanggan berdasarkan contact name yang diawali dengan huruf B!

```
SELECT
orderid, orderdate, empid, custid
FROM Sales.Orders
WHERE
custid =
(
    SELECT custid
    FROM Sales.Customers
    WHERE contactname LIKE N'I%'
);
```



The screenshot displays a SQL Server query window with the following T-SQL code:

```
SELECT
orderid, orderdate, empid, custid
FROM Sales.Orders
WHERE custid = (
    SELECT custid
    FROM Sales.Customers
    WHERE contactname LIKE N'I%'
);
```

Below the query, the 'Results' tab shows the output of the query, which consists of 3 rows:

	orderid	orderdate	empid	custid
1	10326	2006-10-10 00:00:00.000	4	8
2	10970	2008-03-24 00:00:00.000	9	8
3	10801	2007-12-29 00:00:00.000	4	8

The status bar at the bottom indicates: Query executed successfully, LAPTOP-VRURDV67 (16.0 RTM), LAPTOP-VRURDV67\Lavina..., TSQL, 00:00:00, 3 rows.

[Soal-23] Apakah terjadi error pada hasil eksekusi soal-22? Mengapa?

Subquery Harus Mengembalikan Nilai Tunggal: Subquery di dalam klausa WHERE mengharapkan hasil yang tunggal (satu nilai). Jika subquery tersebut mengembalikan lebih dari satu custid, maka akan muncul error. Jika ada kemungkinan subquery mengembalikan lebih dari satu baris, Anda harus menggunakan klausa seperti IN sebagai pengganti =.

[Soal-25] Tulislah pernyataan SELECT untuk mengambil kolom orderid dari tabel Sales.Orders dan juga kolom hasil perhitungan : 1) totalsalesamount (berdasarkan kolom qty dan unitprice dari tabel Sales.OrderDetails) 2) salespctoftotal (presentase total jumlah penjualan setiap pesanan dibagi jumlah total penjualan untuk semua pesanan dalam periode tertentu Filter hasilnya hanya untuk pemesanan pada bulan mei 2008.

The screenshot displays a SQL Server query window with the following T-SQL code:

```

SELECT
    o.orderid,
    SUM(od.qty * od.unitprice) AS totalsalesamount,
    (SUM(od.qty * od.unitprice) / (
        SELECT SUM(od2.qty * od2.unitprice)
        FROM Sales.Orders o2
        JOIN Sales.OrderDetails od2 ON o2.orderid = od2.orderid
        WHERE YEAR(o2.orderdate) = 2008 AND MONTH(o2.orderdate) = 5
    )) * 100 AS salespctoftotal
FROM
    Sales.Orders o
JOIN
    Sales.OrderDetails od ON o.orderid = od.orderid
WHERE
    YEAR(o.orderdate) = 2008
    AND MONTH(o.orderdate) = 5
GROUP BY
    o.orderid;

```

The results pane shows the following data:

orderid	totalsalesamount	salespctoftotal
11064	4722.30	23.73
11065	252.56	1.26
11066	928.75	4.66
11067	86.85	0.43
11068	2384.80	11.98
11069	360.00	1.80
11070	1873.50	9.41
11071	510.00	2.56
11072	5218.00	26.22

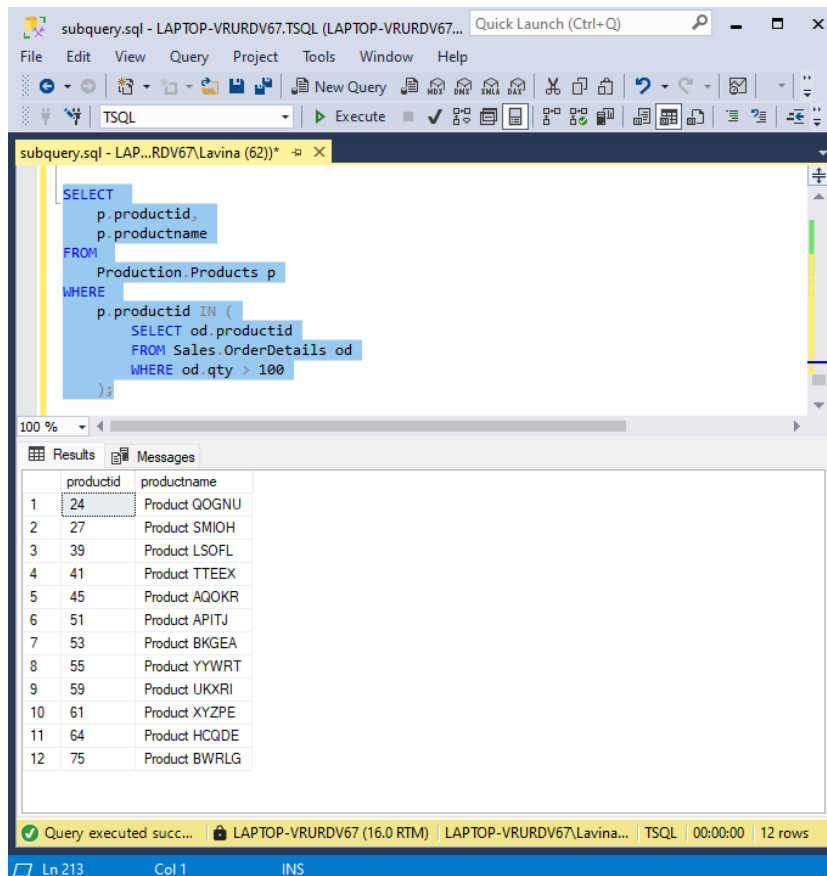
Query executed successfully. 14 rows.

Penjelasan:

- **o.orderid**: Menampilkan kolom orderid dari tabel Sales.Orders.
- **SUM(od.qty * od.unitprice) AS totalsalesamount**: Menghitung total penjualan untuk setiap pesanan berdasarkan kuantitas (qty) dan harga satuan (unitprice) dari tabel Sales.OrderDetails.
- **Subquery di salespctoftotal**: Subquery ini menghitung total penjualan untuk semua pesanan di bulan Mei 2008, digunakan untuk menghitung persentase penjualan dari setiap pesanan terhadap total penjualan di periode tersebut.
- **WHERE YEAR(o.orderdate) = 2008 AND MONTH(o.orderdate) = 5**: Memfilter hasil hanya untuk pesanan di bulan Mei tahun 2008.
- **GROUP BY o.orderid**: Mengelompokkan hasil berdasarkan orderid, sehingga perhitungan agregat (seperti SUM) dilakukan untuk setiap pesanan.

Praktikum – Bagian 6: Menulis Query Yang Menggunakan Sub-Query Skalar Dan Multi Nilai

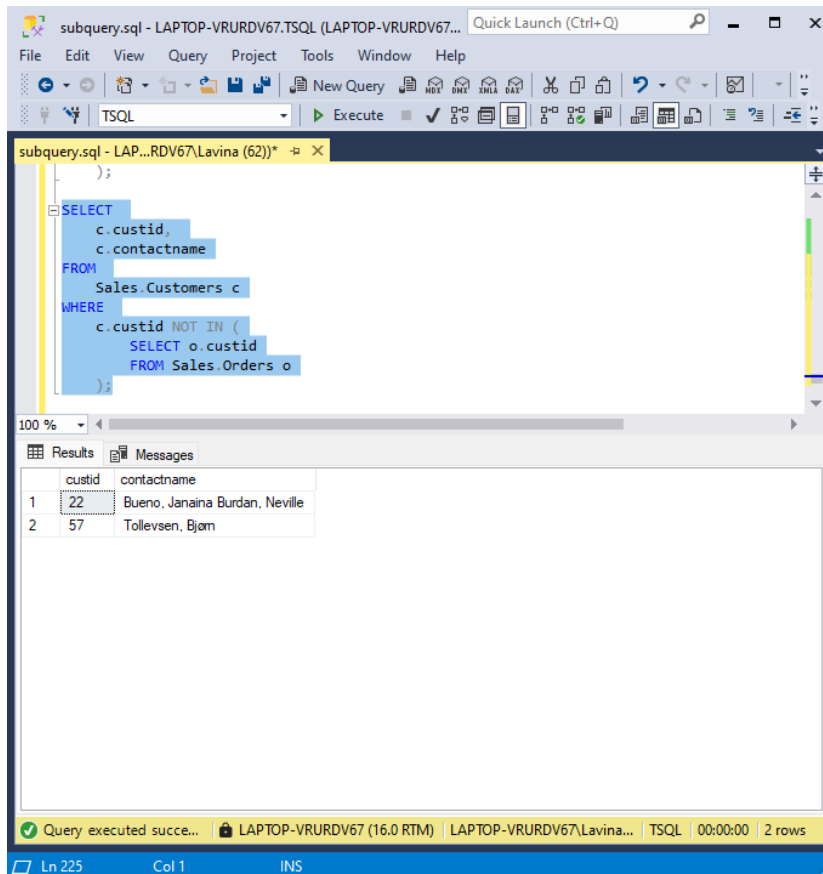
[Soal-26] Tulislah pernyataan SELECT untuk mengambil kolom productid dan productname dari tabel Production.Products. Kemudian filter hasilnya untuk menampilkan produk yang terjual dalam jumlah yang banyak (lebih dari 100 produk) untuk baris pesana tertentu!



Penjelasan:

- **p.productid, p.productname:** Memilih kolom productid dan productname dari tabel Production.Products dengan alias p.
- **Subquery:** Subquery di dalam klausa WHERE memilih productid dari tabel Sales.OrderDetails di mana jumlah produk (qty) yang terjual lebih dari 100.
- **p.productid IN (...):** Filter pada tabel Production.Products untuk hanya menampilkan produk yang productid-nya terdapat dalam hasil subquery, yaitu produk yang terjual lebih dari 100 unit.

[Soal-27] Tulislah pernyataan SELECT untuk mengambil kolom custid dan contactname dari tabel Sales.Customers. Kemudian lakukan filter hanya untuk pelanggan/customer yang tidak memiliki pesanan apapun!



Penjelasan:

- **c.custid, c.contactname:** Memilih kolom custid dan contactname dari tabel Sales.Customers dengan alias c.
- **Subquery:** Subquery di dalam klausa NOT IN memilih semua custid dari tabel Sales.Orders yang mewakili pelanggan yang telah melakukan pesanan.
- **c.custid NOT IN (...):** Memfilter hasil untuk hanya menampilkan pelanggan yang custid-nya tidak ada dalam hasil subquery, artinya mereka belum pernah melakukan pesanan.

[Soal-28] Terdapat tambahan satu baris data pada tabel Sales.Orders dengan T-SQL sebagai berikut :

```
INSERT INTO Sales.Orders (
    custid, empid, orderdate, requireddate, shippeddate, shipperid, freight,
    shipname, shipaddress, shipcity, shipregion, shippostalcode, shipcountry)
VALUES
    (NULL, 1, '20111231', '20111231', '20111231', 1, 0,
    'ShipOne', 'ShipAddress', 'ShipCity', 'RA', '1000', 'USA');
```

100 %

Messages

(1 row affected)

Completion time: 2024-09-26T11:27:30.3825760+07:00

[Soal-29] Modifikasi jawaban soal-27 (cara yang berbeda dengan output yang sama), dengan cara menghapus baris dengan nilai yang tidak diketahui pada kolom custid!

```
SELECT
    c.custid,
    c.contactname
FROM
    Sales.Customers c
LEFT JOIN
    Sales.Orders o ON c.custid = o.custid
WHERE
    o.custid IS NULL;
```

100 %

Results Messages

	custid	contactname
1	22	Bueno, Janaina Burdan, Neville
2	57	Tollefsen, Bjørn

Query executed succe... LAPTOP-VRURDV67 (16.0 RTM) LAPTOP-VRURDV67\Lavina... TSQL 00:00:00 2 rows

Cara lain bisa menggunakan **WHERE o.custid IS NULL** yang bisa memfilter hasil untuk hanya menampilkan pelanggan yang custid-nya tidak ditemukan di tabel Sales.Orders, artinya mereka belum pernah melakukan pesanan.

Praktikum – Bagian 7: Menulis Query Yang Menggunakan Sub-Query Yang Berkorelasi Dan Predikat EXISTS

[Soal-30] Tulislah pernyataan SELECT untuk mengambil kolom custid dan contactname dari tabel Sales.Customers. Tambahkan kolom lastorderdate yang berisi tanggal terakhir dari tabel Sales.Orders untuk setiap pelanggan (Gunakan sub-query yang berkorelasi).

The screenshot shows a SQL query window with the following query:

```
SELECT
  c.custid,
  c.contactname,
  (SELECT MAX(o.orderdate)
   FROM Sales.Orders o
   WHERE o.custid = c.custid
  ) AS lastorderdate
FROM
  Sales.Customers c
```

Below the query, the results are displayed in a table with 15 rows. The columns are custid, contactname, and lastorderdate.

	custid	contactname	lastorderdate
1	1	Allen, Michael	2008-04-09 00:00:00.000
2	2	Hassall, Mark	2008-03-04 00:00:00.000
3	3	Peoples, John	2008-01-28 00:00:00.000
4	4	Amdt, Torsten	2008-04-10 00:00:00.000
5	5	Higginbotham, Tom	2008-03-04 00:00:00.000
6	6	Poland, Carole	2008-04-29 00:00:00.000
7	7	Bansal, Dushyant	2008-01-12 00:00:00.000
8	8	Ilyina, Julia	2008-03-24 00:00:00.000
9	9	Raghav, Amritansh	2008-05-06 00:00:00.000
10	10	Bassols, Pilar Col...	2008-04-24 00:00:00.000
11	11	Jaffe, David	2008-04-14 00:00:00.000
12	12	Ray, Mike	2008-04-28 00:00:00.000
13	13	Benito, Almudena	2006-07-18 00:00:00.000
14	14	Jelitto, Jacek	2008-04-22 00:00:00.000
15	15	Richardson, Shawn	2008-04-22 00:00:00.000

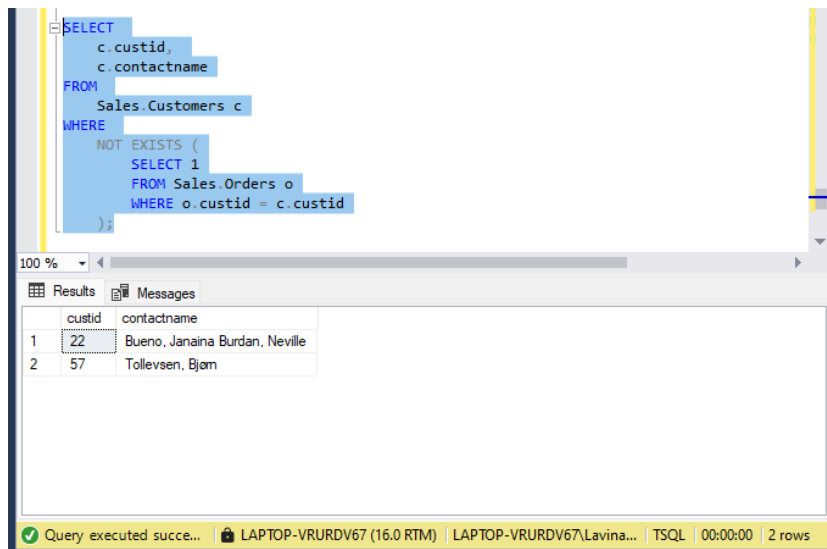
The status bar at the bottom indicates: Query executed succ..., LAPTOP-VRURDV67 (16.0 RTM), LAPTOP-VRURDV67\Lavina..., TSQL, 00:00:00, 91 rows.

Penjelasan :

- **c.custid, c.contactname:** Memilih kolom custid dan contactname dari tabel Sales.Customers dengan alias c.
- **Subquery yang Berkorelasi: (SELECT MAX(o.orderdate) FROM Sales.Orders o WHERE o.custid = c.custid):** Subquery ini mencari tanggal pesanan terakhir (MAX(o.orderdate)) dari tabel Sales.Orders untuk setiap custid yang sama dengan custid dari tabel Sales.Customers. Ini adalah subquery yang berkorelasi karena ia bergantung pada nilai custid dari kueri utama.

- **AS lastorderdate:** Kolom ini menampilkan hasil dari subquery sebagai lastorderdate yang berisi tanggal terakhir pesanan untuk setiap pelanggan.

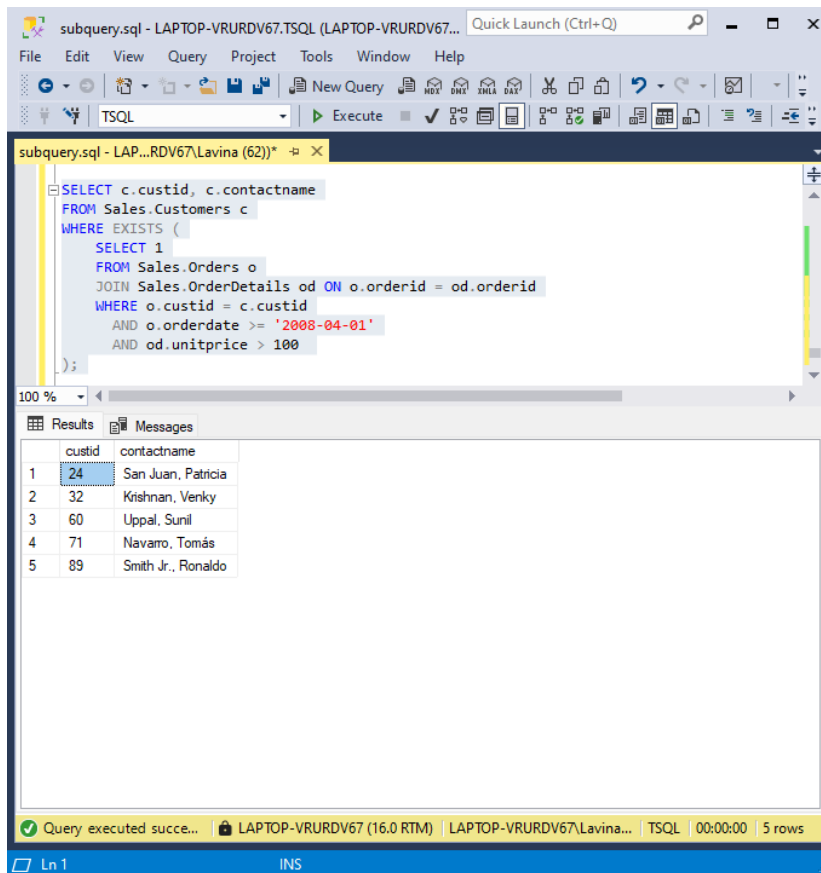
[Soal-31] Tuliskan pernyataan SELECT untuk mengambil semua pelanggan yang tidak memiliki pesanan di tabel Sales.Orders. Gunakan predikat EXISTS untuk melakukan filter yang mencakup pelanggan yang tidak memiliki pesanan! (Tidak diperlukan pemeriksaan eksplisit kolom custid dari tabel Sales.Orders table berstatus not NULL)



Penjelasan:

- **c.custid, c.contactname:** Memilih kolom custid dan contactname dari tabel Sales.Customers dengan alias c.
- **Subquery:** Subquery dengan **EXISTS** memeriksa apakah ada baris dalam tabel Sales.Orders yang memiliki custid yang sama dengan pelanggan di tabel Sales.Customers.
- **NOT EXISTS:** Kondisi ini akan memfilter hasil untuk hanya menampilkan pelanggan yang **tidak memiliki pesanan** (yaitu, custid pelanggan tersebut tidak ditemukan di tabel Sales.Orders).

[Soal-32] Tuliskan pernyataan SELECT untuk mengambil kolom custid dan contactname dari tabel Sales.Customers. Kemudian filter hasilnya hanya untuk pelanggan yang memesan pada atau setelah 1 april 2008, dan melakukan pemesanan dengan harga tinggi di atas \$100!



Penjelasan: Dalam query ini, subquery memeriksa apakah ada pemesanan untuk setiap pelanggan yang memenuhi syarat berdasarkan orderdate dan unitprice.

[Soal-33] Tulislah pernyataan SELECT yang akan mengambil informasi setiap tahun sebagai berikut :

- 1) Tahun pesanan
- 2) Jumlah total penjualan
- 3) Jumlah total penjualan yang terjual selama bertahun-tahun (setiap tahun dikembalikan jumlah total penjualan sampai tahun tertentu, misalkan awal tahun 2006 mengembalikan jumlah total penjualan untuk tahun selanjutnya 2007)
- 4) Pernyataan SELECT harus memiliki 3 kolom :
 - orderyear, berasal dari kolom orderyear dari tabel Sales.Orders
 - totalsales, berasal dari kolom qty dan unitprice dari tabel Sales.OrderDetails
 - runsales, mewakili jumlah penjualan yang sedang terjadi. Kolom ini menggunakan sub-query yang berkorelasi

The screenshot displays a SQL Server query window with the following T-SQL code:

```

SELECT
    YEAR(o.orderdate) AS orderyear,
    SUM(od.qty * od.unitprice) AS totalsales,
    (SELECT SUM(od2.qty * od2.unitprice)
     FROM Sales.Orders o2
     JOIN Sales.OrderDetails od2 ON o2.orderid = od2.orderid
     WHERE YEAR(o2.orderdate) <= YEAR(o.orderdate)) AS runsales
FROM
    Sales.Orders o
JOIN
    Sales.OrderDetails od ON o.orderid = od.orderid
GROUP BY
    YEAR(o.orderdate)
ORDER BY
    YEAR(o.orderdate);

```

The results pane shows the following data:

	orderyear	totalsales	runsales
1	2006	226298,50	226298,50
2	2007	658388,75	884687,25
3	2008	469771,34	1354458,59

The status bar at the bottom indicates: Query executed succe... | LAPTOP-VRURDV67 (16.0 RTM) | LAPTOP-VRURDV67\Lavina... | TSQL | 00:00:00 | 3 rows

Penjelasan: Query SQL ini bertujuan untuk mengambil informasi tentang total penjualan berdasarkan tahun dari tabel `Sales.Orders` dan `Sales.OrderDetails`. Pertama, kueri menggunakan fungsi `YEAR()` untuk mengekstrak tahun dari kolom `orderdate` dan menyimpannya sebagai `orderyear`. Kemudian, kueri menghitung total penjualan (`totalsales`) dengan menjumlahkan hasil kali antara jumlah (`qty`) dan harga unit (`unitprice`) untuk setiap tahun yang ditentukan. Selain itu, query mencakup subquery yang menghitung total penjualan kumulatif (`runsales`) hingga tahun tertentu, dengan memeriksa semua pemesanan sampai tahun yang sama dengan tahun saat ini dalam grup. Hasil akhirnya dikelompokkan berdasarkan tahun (`orderyear`) dan diurutkan berdasarkan tahun tersebut, sehingga pengguna dapat melihat total penjualan serta kumulatif penjualan per tahun dengan cara yang terstruktur.