



Nama : Yusriyah Firjatullah

Kelas : TI 2D

NIM : 2241720178

Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet Minggu-10: Stored Procedure, SQL Dinamis dan Trigger Mata Kuliah Basis Data Lanjut (BDL)

Pengampu: Tim Ajar Basis Data Lanjut *Oktober*
2023

Topik

Membuat Stored Procedure, SQL Dinamis dan Trigger

Tujuan

1. Mengembalikan hasil dengan Stored Procedure.
2. Melewatkan parameter ke Procedure.
3. Membuat stored procedure sederhana, yang menyertakan pernyataan SELECT.
4. Membangun dan menjalankan SQL dinamis dengan EXEC dan sp_executesql.
5. Membangun Trigger

Petunjuk Umum

Langkah	Keterangan
1	<p>Skenario :</p> <p>Departemen TI telah menyediakan kode T-SQL untuk membuat stored procedure untuk mengambil 10 pelanggan teratas berdasarkan jumlah penjualan. Percobaan bagian ini berisi tentang latihan stored procedure.</p> <p>Untuk melakukan percobaan pada praktikum bagian 1 ini terlebih dahulu lakukan login pada SQL Server Management Studio (SSMS). Pastikan database terhubung dengan "TSQL2012".</p>

2	<p>Jalankan kode T-SQL untuk membuat stored procedur Sales.GetTopCustomers berikut:</p> <pre>CREATE PROCEDURE Sales.GetTopCustomers AS</pre>
---	--

1. Ikuti langkah-langkah pada bagian-bagian praktikum sesuai dengan urutan yang diberikan.
2. Anda dapat menggunakan SQL Server untuk mencoba praktikum pada jobsheet ini. Sesuaikan dengan kondisi komputer Anda.
3. Jawablah semua pertanyaan bertanda **[Soal-X]** yang terdapat pada langkah-langkah tertentu di setiap bagian praktikum.
4. Dalam setiap langkah pada praktikum terdapat penjelasan yang akan membantu Anda dalam menjawab pertanyaan-pertanyaan pada petunjuk nomor 3, maka baca dan kerjakanlah semua bagian praktikum dalam jobsheet ini.
5. Tulis jawaban dari soal-soal pada petunjuk nomor 3 pada sebuah laporan yang dikerjakan menggunakan aplikasi word processing (Word, OpenOffice, atau yang lain yang sejenis). Ekspor sebagai file **PDF** dengan format nama sebagai berikut:
 - **BDL_Tugas14_Kelas_2DigitNomorAbsen_NamaLengkapAnda.pdf** - Contoh:
 - o **BDL_Tugas14_TI2Q_99_JoeBiden.pdf** -
Perhatikan baik-baik format penamaanya.
 - Kumpulkan file PDF tersebut sebagai laporan praktikum kepada dosen pengampu.
 - Selain pada nama file, cantumkan juga identitas Anda pada halaman pertama laporan tersebut.

Praktikum – Bagian 1: Menggunakan pernyataan EXECUTE Untuk Memanggil Stored Procedure

```

SELECT TOP(10) c.custid, c.contactname,
SUM(o.val) AS salesvalue
FROM Sales.OrderValues AS o
INNER JOIN Sales.Customers AS c
ON c.custid = o.custid
GROUP BY c.custid, c.contactname
ORDER BY salesvalue DESC;

CREATE PROCEDURE Sales.GetTopCustomers
AS
SELECT TOP(10) c.custid, c.contactname,
SUM(o.val) AS salesvalue
FROM Sales.OrderValues AS o
INNER JOIN Sales.Customers AS c
ON c.custid = o.custid
GROUP BY c.custid, c.contactname
ORDER BY salesvalue DESC;

```

100 %



Messages

Commands completed successfully.

3

[Soal-1] Tuliskan pernyataan T-SQL untuk menjalankan prosedur yang telah dibuat sebelumnya!

```
EXEC Sales.GetTopCustomers;
```

4

[Soal-2] Letakkan hasil eksekusi soal no. 1 pada bagian ini dan pastikan hasilnya sama dengan gambar di bawah ini.

custid	contactname	salesvalue
63	Veronesi, Giorgio	110277.32
20	Kane, John	104874.99
71	Navarro, Tomás	104361.96
...		
...		
...		
24	San Juan, Patricia	29567.57
51	Taylor, Maurice	28872.19
89	Smith Jr., Ronaldo	27363.61
(10 row(s) affected)		

	custid	contactname	salesvalue
1	63	Veronesi, Giorgio	110277.32
2	20	Kane, John	104874.99
3	71	Navarro, Tomás	104361.96
4	65	Moore, Michael	51097.80
5	37	Cr?ciun, Ovidiu V.	49979.91
6	34	Cohen, Shy	32841.37
7	39	Song, Lolan	30908.39
8	24	San Juan, Patricia	29567.57
9	51	Taylor, Maurice	28872.19
10	89	Smith Jr., Ronaldo	27363.61

Departemen TI telah merubah stored procedure dari percobaan 1 langkah 2 dengan kode T-SQL di bawah ini :

```
ALTER PROCEDURE Sales.GetTopCustomers
AS
SELECT  c.custid, c.contactname,
SUM(o.val) AS salesvalue
FROM Sales.OrderValues AS o
INNER JOIN Sales.Customers AS c
ON c.custid = o.custid
GROUP BY c.custid, c.contactname
ORDER BY salesvalue DESC
OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY;

ALTER PROCEDURE Sales.GetTopCustomers
AS
SELECT  c.custid, c.contactname,
SUM(o.val) AS salesvalue
FROM Sales.OrderValues AS o
INNER JOIN Sales.Customers AS c
ON c.custid = o.custid
GROUP BY c.custid, c.contactname
ORDER BY salesvalue DESC
OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY;
```

Messages
Commands completed successfully.

[Soal-3] Tuliskan pernyataan T-SQL yang akan menjalankan prosedur pada percobaan 1 langkah 6 di atas!

```
EXEC Sales.GetTopCustomers;
```

[Soal-4] Letakkan hasil eksekusi soal no. 3 pada bagian ini dan pastikan hasilnya sama dengan gambar di bawah ini.

```

custid    contactname    salesvalue
-----
63        Veronesi, Giorgio    110277.32
20        Kane, John           104874.99
71        Navarro, Tomás       104361.96
...
...
...
24        San Juan, Patricia    29567.57
51        Taylor, Maurice       28872.19
89        Smith Jr., Ronaldo    27363.61

(10 row(s) affected)

```

7

	custid	contactname	salesvalue
1	63	Veronesi, Giorgio	110277.32
2	20	Kane, John	104874.99
3	71	Navarro, Tomás	104361.96
4	65	Moore, Michael	51097.80
5	37	Cr?ciun, Ovidiu V.	49979.91
6	34	Cohen, Shy	32841.37
7	39	Song, Lolan	30908.39
8	24	San Juan, Patricia	29567.57
9	51	Taylor, Maurice	28872.19
10	89	Smith Jr., Ronaldo	27363.61

8

[Soal-5] Apa perbedaan kode T-SQL untuk prosedur pada percobaan 1 langkah 2 dan langkah 5?

- kode T-SQL untuk prosedur langkah 2 menggunakan keyword top untuk mengambil 10 nilai teratas dari query. Sedangkan pada langkah ke 5 menggunakan offset fetch. Dengan nilai rows offsetnya 0 dan fetchnya 10 yang artinya mengambil 10 kolom mulai dari kolom pertama

9	<p>[Soal-6] Jika beberapa aplikasi menggunakan prosedur yang ada pada percobaan 1 langkah 2, apakah aplikasi-aplikasi tersebut masih bisa bekerja dengan baik setelah terjadi perubahan sesuai dengan percobaan 1 langkah 5?</p> <p>- Jika aplikasi menggunakan prosedur yang mengandalkan kode T-SQL pada langkah 2 yang menggunakan TOP untuk mengambil 10 nilai teratas dari query, perubahan pada langkah 5 dengan penggunaan OFFSET FETCH bisa menyebabkan dampak terhadap cara aplikasi tersebut mengambil dan memproses data.</p>
10	<p>Kesimpulan : Setelah menjalankan praktikum bagian ini, mahasiswa mengetahui bagaimana memanggil stored procedure menggunakan pernyataan EXECUTE.</p>

Praktikum – Bagian 2: Melewatkan Parameter ke Stored Procedure

Langkah	Keterangan
1	<p>Skenario :</p> <p>Departemen TI memberikan tambahan modifikasi dari stored procedure di praktikum bagian 1. Stored procedure yang dimodifikasi akan melewati parameter yang menentukan tahun pesanan dan jumlah pelanggan yang akan diambil.</p> <p>Untuk melakukan percobaan pada praktikum bagian 2 ini, pastikan database terhubung dengan "TSQL2012".</p>
2	<p>Jalankan kode T-SQL berikut, untuk memodifikasi stored procedure Sales.GetTopCustomers untuk memasukkan parameter tahun pesanan (@orderyear) :</p> <pre> ALTER PROCEDURE Sales.GetTopCustomers @orderyear int AS SELECT c.custid, c.contactname, SUM(o.val) AS salesvalue FROM Sales.OrderValues AS o INNER JOIN Sales.Customers AS c ON c.custid = o.custid WHERE YEAR(o.orderdate) = @orderyear GROUP BY c.custid, c.contactname ORDER BY salesvalue DESC OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY; </pre>

```

ALTER PROCEDURE Sales.GetTopCustomers
@orderyear int
AS
SELECT c.custid, c.contactname,
SUM(o.val) AS salesvalue
FROM Sales.OrderValues AS o
INNER JOIN Sales.Customers AS c
ON c.custid = o.custid
WHERE YEAR(o.orderdate) = @orderyear
GROUP BY c.custid, c.contactname
ORDER BY salesvalue DESC
OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY;

```

100 %

Messages

Commands completed successfully.

3

[Soal-7] Tuliskan pernyataan EXECUTE untuk memanggil stored procedure Sales.GetTopCustomers untuk tahun 2007!

```
EXECUTE Sales.GetTopCustomers @orderyear = 2007;
```

4

[Soal-8] Letakkan hasil eksekusi soal no. 7 pada bagian ini dan pastikan hasilnya sama dengan gambar di bawah ini.

custid	contactname	salesvalue
63	Veronesi, Giorgio	61109.92
71	Navarro, Tomás	57713.58
20	Kane, John	48096.27
...		
...		
...		
5	Higginbotham, Tom	13849.02
35	Langohr, Kris	13482.74
24	San Juan, Patricia	13314.67

(10 row(s) affected)

	custid	contactname	salesvalue
1	63	Veronesi, Giorgio	61109.92
2	71	Navarro, Tomás	57713.58
3	20	Kane, John	48096.27
4	51	Taylor, Maurice	23332.31
5	37	Cr?ciun, Ovidiu V.	20454.41
6	65	Moore, Michael	19383.75
7	73	Gonzalez, Nuria	16232.41
8	5	Higginbotham, Tom	13849.02
9	35	Langohr, Kris	13482.74
10	24	San Juan, Patricia	13314.67

[Soal-9] Tuliskan pernyataan EXECUTE untuk memanggil stored procedure Sales.GetTopCustomers untuk tahun 2008! Pastikan hasilnya sesuai dengan gambar berikut :

custid	contactname	salesvalue
20	Kane, John	41210.65
63	Veronesi, Giorgio	37217.32
71	Navarro, Tomás	36310.11
...		
...		
...		
89	Smith Jr., Ronaldo	15278.90
24	San Juan, Patricia	13644.07
76	Gulbis, Katrin	11644.60

(10 row(s) affected)

```
EXECUTE Sales.GetTopCustomers @orderyear = 2008;
```

00 %

Results Messages

	custid	contactname	salesvalue
1	20	Kane, John	41210.65
2	63	Veronesi, Giorgio	37217.32
3	71	Navarro, Tomás	36310.11
4	34	Cohen, Shy	23821.20
5	65	Moore, Michael	21238.27
6	37	Cr?ciun, Ovidiu V.	20402.12
7	39	Song, Lolan	19582.78
8	89	Smith Jr., Ronaldo	15278.90
9	24	San Juan, Patricia	13644.07
10	76	Gulbis, Katrin	11644.60

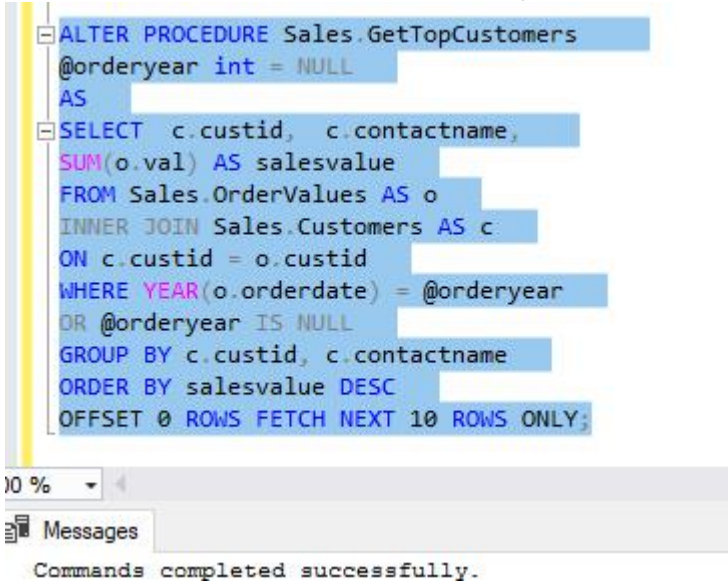
[Soal-10] Tuliskan pernyataan EXECUTE untuk memanggil stored procedure Sales.GetTopCustomers tanpa parameter!

```
EXECUTE Sales.GetTopCustomers;
```

0 %

Messages

Msg 201, Level 16, State 4, Procedure Sales.GetTopCustomers, Line 0 [Batch Start Line 19]
Procedure or function 'GetTopCustomers' expects parameter '@orderyear', which was not supplied.

7	<p>[Soal-11] Apakah hasil yang didapat dari soal no.10? Jika terdapat error, tuliskan pesan errornya!</p> <ul style="list-style-type: none"> - Procedure or function 'GetTopCustomers' expects parameter '@orderyear', which was not supplied --> menunjukkan bahwa procedure membutuhkan parameter untuk variabel orderyear
8	<p>[Soal-12] Jika aplikasi dirancang untuk menggunakan stored procedure pada praktikum bagian 1, apakah stored procedure yang telah dimodifikasi mempengaruhi kegunaan aplikasi tersebut?Jelaskan!</p> <ul style="list-style-type: none"> - Prosedur yang telah dimodifikasi dengan menggunakan OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY untuk mengambil 10 baris teratas dari hasil kueri akan mempengaruhi aplikasi yang menggunakannya. Dampaknya tergantung pada bagaimana aplikasi tersebut menggunakan hasil yang diterima dari prosedur ini.
9	<p>Jalankan kode T-SQL berikut untuk memodifikasi stored procedure Sales.GetTopCustomers:</p> <pre> ALTER PROCEDURE Sales.GetTopCustomers @orderyear int = NULL AS SELECT c.custid, c.contactname, SUM(o.val) AS salesvalue FROM Sales.OrderValues AS o INNER JOIN Sales.Customers AS c ON c.custid = o.custid WHERE YEAR(o.orderdate) = @orderyear OR @orderyear IS NULL GROUP BY c.custid, c.contactname ORDER BY salesvalue DESC OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY; </pre>  <p>100 %</p> <p>Messages</p> <p>Commands completed successfully.</p>

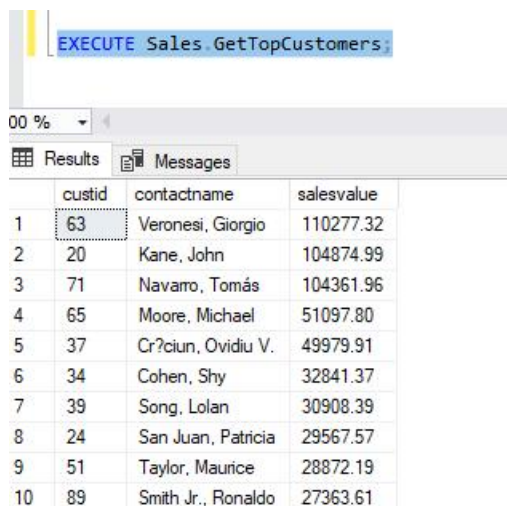
[Soal-13] Tulis pernyataan EXECUTE untuk memanggil stored procedure

```
custid    contactname    salesvalue
-----
63        Veronesi, Giorgio    110277.32
20        Kane, John          104874.99
71        Navarro, Tomás      104361.96
...
...
...
24        San Juan, Patricia    29567.57
51        Taylor, Maurice      28872.19
89        Smith Jr., Ronaldo    27363.61

(10 row(s) affected)
```

Sales.GetTopCustomers tanpa parameter! Pastikan hasilnya sesuai dengan gambar berikut :

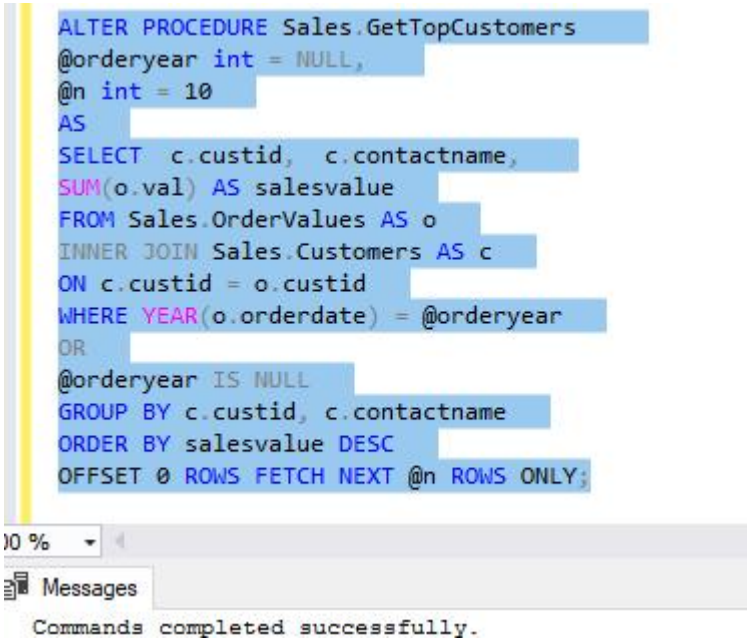
10



```
EXECUTE Sales.GetTopCustomers;
```

	custid	contactname	salesvalue
1	63	Veronesi, Giorgio	110277.32
2	20	Kane, John	104874.99
3	71	Navarro, Tomás	104361.96
4	65	Moore, Michael	51097.80
5	37	Cr?ciun, Ovidiu V.	49979.91
6	34	Cohen, Shy	32841.37
7	39	Song, Lolan	30908.39
8	24	San Juan, Patricia	29567.57
9	51	Taylor, Maurice	28872.19
10	89	Smith Jr., Ronaldo	27363.61

11	<p>[Soal-14] Jika terdapat modifikasi pada stored procedure, apakah mempengaruhi aplikasi atau memerlukan perubahan pada desain aplikasi?</p> <ul style="list-style-type: none"> - Jika terdapat modifikasi pada stored procedure, dampaknya pada aplikasi tergantung pada jenis perubahan yang dilakukan. Perubahan pada stored procedure dapat mempengaruhi aplikasi dan terkadang memerlukan perubahan pada desain aplikasi. Contoh jika perubahan terjadi pada parameter stored procedure, aplikasi yang memanggil stored procedure tersebut mungkin perlu diperbarui agar sesuai dengan perubahan.
----	---

12	<p>Jalankan kode T-SQL yang disediakan untuk menambahkan parameter @n ke stored procedure Sales.GetTopCustomers. Parameter tersebut digunakan untuk menentukan berapa banyak pelanggan yang akan diambil. Nilai defaultnya adalah 10.</p> <pre> ALTER PROCEDURE Sales.GetTopCustomers @orderyear int = NULL, @n int = 10 AS SELECT c.custid, c.contactname, SUM(o.val) AS salesvalue FROM Sales.OrderValues AS o INNER JOIN Sales.Customers AS c ON c.custid = o.custid WHERE YEAR(o.orderdate) = @orderyear OR @orderyear IS NULL GROUP BY c.custid, c.contactname ORDER BY salesvalue DESC OFFSET 0 ROWS FETCH NEXT @n ROWS ONLY; </pre>  <p>10 %</p> <p>Messages</p> <p>Commands completed successfully.</p>
----	---

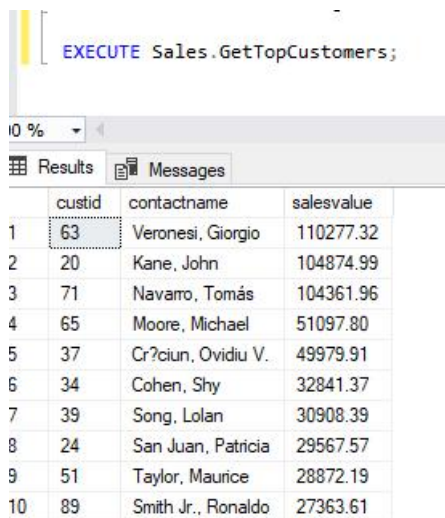
[Soal-15] Tulis pernyataan EXECUTE untuk memanggil stored procedure

```
custid    contactname    salesvalue
-----
63        Veronesi, Giorgio    110277.32
20        Kane, John           104874.99
71        Navarro, Tomás       104361.96
...
...
...
24        San Juan, Patricia   29567.57
51        Taylor, Maurice       28872.19
89        Smith Jr., Ronaldo    27363.61

(10 row(s) affected)
```

Sales.GetTopCustomers tanpa parameter! Pastikan hasilnya sesuai dengan gambar berikut :

13



```
EXECUTE Sales.GetTopCustomers;
```

	custid	contactname	salesvalue
1	63	Veronesi, Giorgio	110277.32
2	20	Kane, John	104874.99
3	71	Navarro, Tomás	104361.96
4	65	Moore, Michael	51097.80
5	37	Cr?ciun, Ovidiu V.	49979.91
6	34	Cohen, Shy	32841.37
7	39	Song, Lolan	30908.39
8	24	San Juan, Patricia	29567.57
9	51	Taylor, Maurice	28872.19
10	89	Smith Jr., Ronaldo	27363.61

[Soal-16] Tuliskan pernyataan EXECUTE untuk memanggil stored procedure Sales.GetTopCustomers untuk tahun 2008 dan 5 orang pelanggan! Pastikan hasilnya sesuai dengan gambar berikut :

custid	contactname	salesvalue
20	Kane, John	41210.65
63	Veronesi, Giorgio	37217.32
71	Navarro, Tomás	36310.11
34	Cohen, Shy	23821.20
65	Moore, Michael	21238.27

(5 row(s) affected)

14

```
EXECUTE Sales.GetTopCustomers @orderyear = 2008, @n = 5;
```

%

Results

Messages

custid	contactname	salesvalue
20	Kane, John	41210.65
63	Veronesi, Giorgio	37217.32
71	Navarro, Tomás	36310.11
34	Cohen, Shy	23821.20
65	Moore, Michael	21238.27

15

[Soal-17] Tuliskan pernyataan EXECUTE untuk memanggil stored procedure Sales.GetTopCustomers untuk tahun 2007! Pastikan hasilnya sesuai dengan gambar berikut :

custid	contactname	salesvalue
63	Veronesi, Giorgio	61109.92
71	Navarro, Tomás	57713.58
20	Kane, John	48096.27
...		
...		
...		
5	Higginbotham, Tom	13849.02
35	Langohr, Kris	13482.74
24	San Juan, Patricia	13314.67

(10 row(s) affected)

EXECUTE Sales.GetTopCustomers @orderyear = 2007, @n = 10;			
% ▾			
Results Messages			
custid	contactname	salesvalue	
63	Veronesi, Giorgio	61109.92	
71	Navarro, Tomás	57713.58	
20	Kane, John	48096.27	
51	Taylor, Maurice	23332.31	
37	Cr?ciun, Ovidiu V.	20454.41	
65	Moore, Michael	19383.75	
73	Gonzalez, Nuria	16232.41	
5	Higginbotham, Tom	13849.02	
35	Langohr, Kris	13482.74	
24	San Juan, Patricia	13314.67	

[Soal-18]. Tuliskan pernyataan EXECUTE untuk memanggil stored procedure Sales.GetTopCustomers untuk mengambil 20 orang pelanggan! Pastikan hasilnya sesuai dengan gambar berikut :

custid	contactname	salesvalue
63	Veronesi, Giorgio	110277.32
20	Kane, John	104874.99
71	Navarro, Tomás	104361.96
...		
...		
...		
10	Bassols, Pilar Colome	20801.61
68	Myrcha, Jacek	19343.78
44	Louverdis, George	19261.42
(20 row(s) affected)		

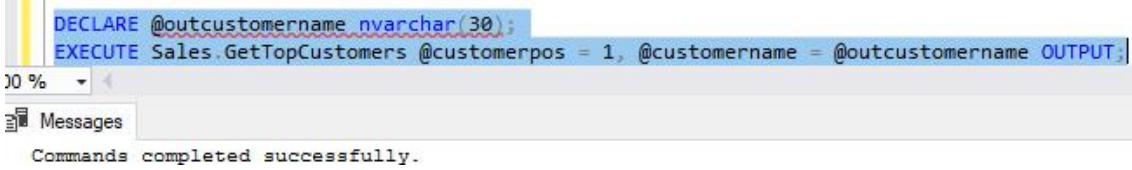
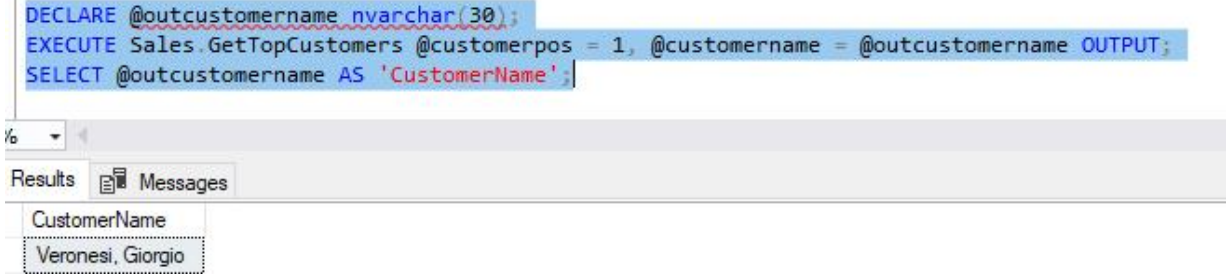
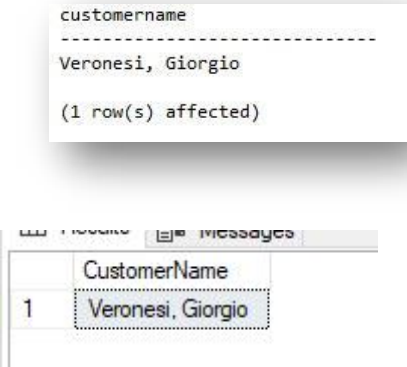
16

EXECUTE Sales.GetTopCustomers @n = 20;			
10 % ▾			
Results Messages			
	custid	contactname	salesvalue
1	63	Veronesi, Giorgio	110277.32
2	20	Kane, John	104874.99
3	71	Navarro, Tomás	104361.96
4	65	Moore, Michael	51097.80
5	37	Cr?ciun, Ovidiu V.	49979.91
6	34	Cohen, Shy	32841.37
7	39	Song, Lolan	30908.39
8	24	San Juan, Patricia	29567.57
9	51	Taylor, Maurice	28872.19
10	89	Smith Jr., Ronaldo	27363.61
11	25	Carlson, Jason	26656.57

[Soal-19] Apakah aplikasi yang menggunakan stored procedure perlu diubah jika parameter lain ditambahkan?
 - Ya, aplikasi yang menggunakan stored procedure perlu diubah jika parameter lain ditambahkan ke stored procedure tersebut. Tergantung apakah parameter

17

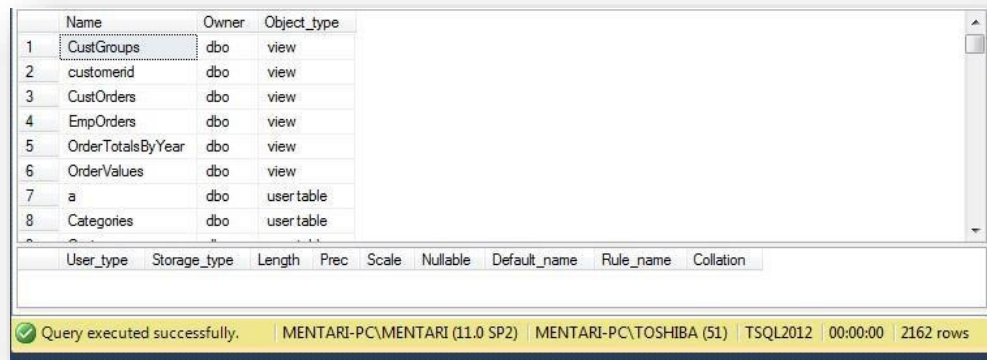
	<p>yang ditambahkan bersifat opsional atau wajib, dan bagaimana parameter tersebut digunakan dalam logika bisnis stored procedure. Jika parameter yang ditambahkan telah diberikan nilai defaultnya maka bersifat opsional sehingga pengaplikasian boleh tidak diubah</p>
18	<p>Jalankan kode T-SQL berikut yang memodifikasi stored procedure Sales.GetTopCustomers untuk mengembalikan nama pelanggan/konsumen berdasarkan posisi tertentu dalam peringkat total penjualan yang disediakan oleh parameter @customerpos. Stored procedure ini juga menyediakan parameter baru bernama @customername yang memiliki opsi OUTPUT.</p> <pre> ALTER PROCEDURE Sales.GetTopCustomers @customerpos int = 1, @customername nvarchar(30) OUTPUT AS SET @customername = (SELECT c.contactname FROM Sales.OrderValues AS o INNER JOIN Sales.Customers AS c ON c.custid = o.custid GROUP BY c.custid, c.contactname ORDER BY SUM(o.val) DESC OFFSET @customerpos - 1 ROWS FETCH NEXT 1 ROW ONLY); </pre>  <p>Commands completed successfully.</p>
19	<p>Departemen TI juga memberikan kode T-SQL untuk mendeklarasikan variabel baru @outcustomername. Variabel ini digunakan sebagai output parameter stored procedure yang dibuat.</p> <pre> DECLARE @outcustomername nvarchar(30); </pre>  <p>Commands completed successfully.</p>

20	<p>[Soal-20] Tuliskan pernyataan EXECUTE untuk memanggil stored procedure Sales.GetTopCustomers, dan ambil pelanggan pertama!</p> 
21	<p>[Soal-21] Tuliskan pernyataan SELECT untuk mengambil nilai dari paramater output @outcustomername!</p> 
22	<p>[Soal-22] Jalankan sekumpulan kode T-SQL yang terdiri dari pernyataan DECLARE, EXECUTE, dan SELECT! Pastikan hasilnya sesuai dengan gambar berikut :</p> 
23	<p>Kesimpulan: Setelah menjalankan praktikum bagian ini, mahasiswa mengetahui bagaimana memanggil stored procedure yang memiliki parameter.</p>

Praktikum – Bagian 3: Executing System Stored Procedures

Langkah	Keterangan
1	<p>Skenario :</p> <p>Praktikum pada bagian ini akan mempelajari bagaimana menjalankan beberapa sistem yang paling sering digunakan untuk menyimpan informasi tentang tabel dan kolom.</p> <p>Untuk melakukan percobaan pada praktikum bagian 3 ini, pastikan database terhubung dengan “TSQL2012”.</p>

[Soal-23] Tuliskan pernyataan EXECUTE untuk memanggil stored procedure tersimpan

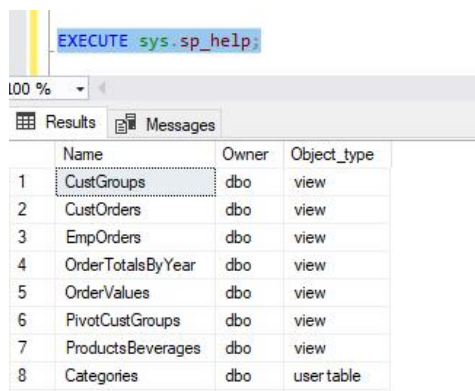


	Name	Owner	Object_type
1	CustGroups	dbo	view
2	customerid	dbo	view
3	CustOrders	dbo	view
4	EmpOrders	dbo	view
5	OrderTotalsByYear	dbo	view
6	OrderValues	dbo	view
7	a	dbo	user table
8	Categories	dbo	user table

Query executed successfully. | MENTARI-PC\MENTARI (11.0 SP2) | MENTARI-PC\TOSHIBA (51) | TSQL2012 | 00:00:00 | 2162 rows

sys.sp_help tanpa parameter, dengan hasil yang sesuai tampilan di bawah ini!

2



```
EXECUTE sys.sp_help;
```

100 %

Results Messages

	Name	Owner	Object_type
1	CustGroups	dbo	view
2	CustOrders	dbo	view
3	EmpOrders	dbo	view
4	OrderTotalsByYear	dbo	view
5	OrderValues	dbo	view
6	PivotCustGroups	dbo	view
7	ProductsBeverages	dbo	view
8	Categories	dbo	user table

3

[Soal-24] Tuliskan pernyataan EXECUTE untuk memanggil store prosedur sys.sp_help untuk tabel tertentu dengan melewati parameter Sales.Customer, dengan hasil yang sesuai tampilan di bawah ini!

```
EXECUTE sys.sp_help @objname = 'Sales.Customers';
```

Name	Owner	Type	Created_datetime
Customers	dbo	user table	2017-11-26 08:22:16.967

Column_name	Type	Computed	Length	Prec	Scale	Nullable	Trim	TrailingBlanks	FixedLenNullInSource	Collation
1 custid	int	no	4	10	0	no	(n/a)	(n/a)	(n/a)	NULL
2 companyname	nvarchar	no	80			no	(n/a)	(n/a)	(n/a)	Latin1_General_CI_AS
3 contactname	nvarchar	no	60			no	(n/a)	(n/a)	(n/a)	Latin1_General_CI_AS
4 contacttitle	nvarchar	no	60			no	(n/a)	(n/a)	(n/a)	Latin1_General_CI_AS
5 address	nvarchar	no	120			no	(n/a)	(n/a)	(n/a)	Latin1_General_CI_AS
6 city	nvarchar	no	30			no	(n/a)	(n/a)	(n/a)	Latin1_General_CI_AS
7 region	nvarchar	no	30			yes	(n/a)	(n/a)	(n/a)	Latin1_General_CI_AS
8 postalcode	nvarchar	no	20			yes	(n/a)	(n/a)	(n/a)	Latin1_General_CI_AS
9 country	nvarchar	no	30			no	(n/a)	(n/a)	(n/a)	Latin1_General_CI_AS

Identity	Seed	Increment	Not For Replication
1 custid	1	1	0

RowGuidCol
1 No rowguidcol column defined.

Date_located_on_filegroup
1 PRIMARY

index_name	index_description	index_keys
1 idx_nc_city	nonclustered located on PRIMARY	city
2 idx_nc_companyname	nonclustered located on PRIMARY	companyname
3 idx_nc_postalcode	nonclustered located on PRIMARY	postalcode
4 idx_nc_region	nonclustered located on PRIMARY	region
5 PK_Customers	clustered, unique, primary key loc...	custid

constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_keys
1 PRIMARY KEY (clustered)	FK_Customers	(n/a)	(n/a)	(n/a)	(n/a)	custid

Table is referenced by foreign key
1 TSQL2012.Sales.Orders: FK_Orders_Customers

Query executed successfully. MENTARI-PC\MENTARI (11.0 SP2) MENTARI-PC\TOSHIBA (51) TSQL2012 00:00:00 11 row

Name	Owner	Type	Created_datetime
Customers	dbo	user table	2023-09-16 15:09:50.697

Column_name	Type	Computed	Length	Prec	Scale	Nullable	Trim	TrailingBlanks	FixedLenNullInSource	Collation
1 custid	int	no	4	10	0	no	(n/a)	(n/a)	(n/a)	NULL
2 companyname	nvarchar	no	80			no	(n/a)	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
3 contactname	nvarchar	no	60			no	(n/a)	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
4 contacttitle	nvarchar	no	60			no	(n/a)	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
5 address	nvarchar	no	120			no	(n/a)	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
6 city	nvarchar	no	30			no	(n/a)	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
7 region	nvarchar	no	30			yes	(n/a)	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS
8 postalcode	nvarchar	no	20			yes	(n/a)	(n/a)	(n/a)	SQL_Latin1_General_CP1_CI_AS

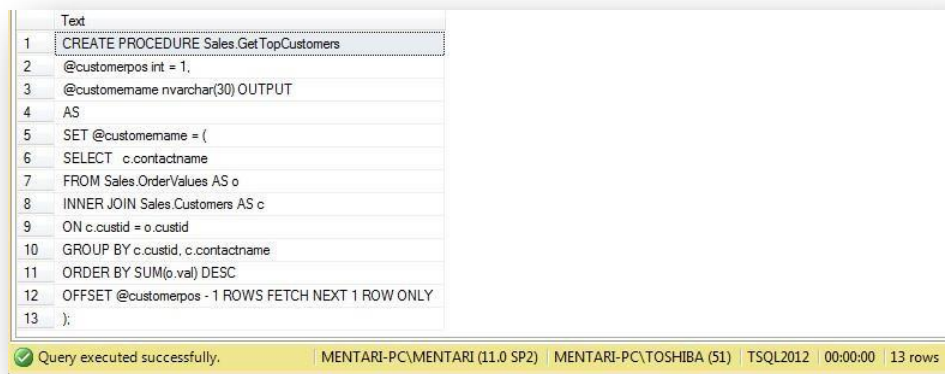
Identity	Seed	Increment	Not For Replication
1 custid	1	1	0

RowGuidCol
1 No rowguidcol column defined.

Date_located_on_filegroup
1 PRIMARY

index_name	index_description	index_keys
1 idx_nc_city	nonclustered located on PRIMARY	city
2 idx_nc_companyname	nonclustered located on PRIMARY	companyname
3 idx_nc_postalcode	nonclustered located on PRIMARY	postalcode

[Soal-25] Tuliskan pernyataan EXECUTE untuk memanggil stored prosedur sys.sp_helptext, kemudian lewatkan stored prosedur Sales.GetTopCustomers yang disimpan sebagai



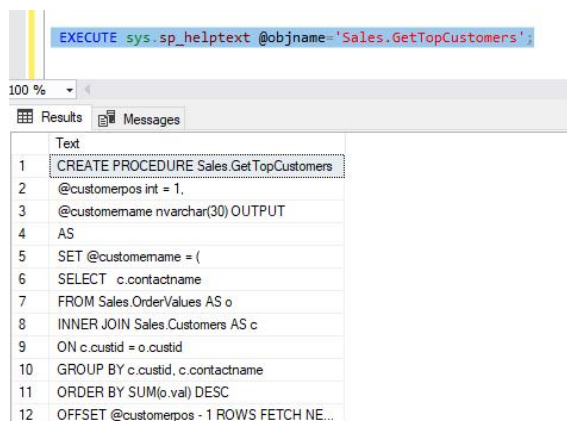
The screenshot shows a SQL Server Enterprise Manager window with the 'Text' tab selected. It displays the definition of the stored procedure Sales.GetTopCustomers. The code is as follows:

```
1 CREATE PROCEDURE Sales.GetTopCustomers
2 @customerpos int = 1,
3 @customername nvarchar(30) OUTPUT
4 AS
5 SET @customername = (
6 SELECT c.contactname
7 FROM Sales.OrderValues AS o
8 INNER JOIN Sales.Customers AS c
9 ON c.custid = o.custid
10 GROUP BY c.custid, c.contactname
11 ORDER BY SUM(o.val) DESC
12 OFFSET @customerpos - 1 ROWS FETCH NEXT 1 ROW ONLY
13 );
```

At the bottom of the window, a status bar indicates: 'Query executed successfully. MENTARI-PC\MENTARI (11.0 SP2) MENTARI-PC\TOSHIBA (51) TSQL2012 00:00:00 13 rows'.

parameter, dengan hasil yang sesuai tampilan di bawah ini!

4



The screenshot shows a SQL Server Enterprise Manager window with the 'Text' tab selected. It displays the execution of the sys.sp_helptext stored procedure for the Sales.GetTopCustomers procedure. The code is as follows:

```
EXECUTE sys.sp_helptext @objname='Sales.GetTopCustomers';
```

Below the code, there are tabs for 'Results' and 'Messages'. The 'Results' tab is selected, and it shows the definition of the stored procedure Sales.GetTopCustomers, which is identical to the one shown in the previous screenshot.

[Soal-26] Tulislah pernyataan EXECUTE untuk memanggil stored procedure sys.sp_columns untuk tabel Sales.Customers. Terdapat dua parameter yang harus dilewati, yaitu:

	TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME	PRECISION	LENGTH	SCALE
1	TSQL2012	Sales	Customers	custid	4	int identity	10	4	0
2	TSQL2012	Sales	Customers	companyname	-9	nvarchar	40	80	NULL
3	TSQL2012	Sales	Customers	contactname	-9	nvarchar	30	60	NULL
4	TSQL2012	Sales	Customers	contacttitle	-9	nvarchar	30	60	NULL
5	TSQL2012	Sales	Customers	address	-9	nvarchar	60	120	NULL
6	TSQL2012	Sales	Customers	city	-9	nvarchar	15	30	NULL
7	TSQL2012	Sales	Customers	region	-9	nvarchar	15	30	NULL
8	TSQL2012	Sales	Customers	postalcode	-9	nvarchar	10	20	NULL
9	TSQL2012	Sales	Customers	country	-9	nvarchar	15	30	NULL
10	TSQL2012	Sales	Customers	phone	-9	nvarchar	24	48	NULL
11	TSQL2012	Sales	Customers	fax	-9	nvarchar	24	48	NULL

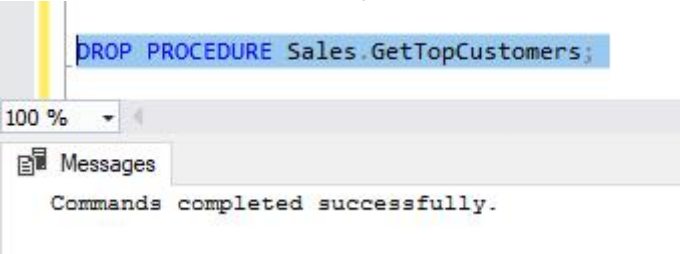
Query executed successfully. MENTARI-PC\MENTARI (11.0 SP2) MENTARI-PC\TOSHIBA (51) TSQL2012 00:00:00 11 rows

@table_name dan @table_owner, dengan hasil yang sesuai tampilan di bawah ini!

5

```
EXECUTE sys.sp_columns @table_name = 'Customers', @table_owner = 'Sales';
```

	TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME	PRECISION	LENGTH	SCALE	RADIX	NULLABLE	REMARKS	COLUMN_DEF	SQL
1	TSQL	Sales	Customers	custid	4	int identity	10	4	0	10	0	NULL	NULL	4
2	TSQL	Sales	Customers	companyname	-9	nvarchar	40	80	NULL	NULL	0	NULL	NULL	-9
3	TSQL	Sales	Customers	contactname	-9	nvarchar	30	60	NULL	NULL	0	NULL	NULL	-9
4	TSQL	Sales	Customers	contacttitle	-9	nvarchar	30	60	NULL	NULL	0	NULL	NULL	-9
5	TSQL	Sales	Customers	address	-9	nvarchar	60	120	NULL	NULL	0	NULL	NULL	-9
6	TSQL	Sales	Customers	city	-9	nvarchar	15	30	NULL	NULL	0	NULL	NULL	-9
7	TSQL	Sales	Customers	region	-9	nvarchar	15	30	NULL	NULL	1	NULL	NULL	-9
8	TSQL	Sales	Customers	postalcode	-9	nvarchar	10	20	NULL	NULL	1	NULL	NULL	-9
9	TSQL	Sales	Customers	country	-9	nvarchar	15	30	NULL	NULL	0	NULL	NULL	-9
10	TSQL	Sales	Customers	phone	-9	nvarchar	24	48	NULL	NULL	0	NULL	NULL	-9
11	TSQL	Sales	Customers	fax	-9	nvarchar	24	48	NULL	NULL	1	NULL	NULL	-9

6	<p>Eksekusi pernyataan T-SQL berikut untuk menghapus stored procedure Sales.GetTopCustomers:</p> <pre> DROP PROCEDURE Sales.GetTopCustomers; </pre> 
7	<p>Kesimpulan: Setelah melakukan praktikum bagian ini, mahasiswa memiliki pengetahuan dasar cara memanggil berbagai sistem stored procedure yang berbeda-beda.</p>

Praktikum – Bagian 4: TRIGGER (AFTER)

Langkah	Keterangan
---------	------------

TRIGGER: Trigger adalah semacam stored procedure (fungsi yang tidak mengembalikan nilai) spesial yang akan dieksekusi ketika ada sebuah event yang terjadi pada suatu tabel.

Trigger ada 2:

- TRIGGER **AFTER**: Trigger yang MENAMBAHKAN suatu aksi
- TRIGGER **INSTEAD OF**: Trigger yang MENCEGAH suatu aksi

Trigger **AFTER INSERT**: Adalah trigger yang akan dieksekusi ketika ada operasi INSERT berhasil (selesai, after) dilakukan pada tabel yang dipasang trigger tersebut.

Misalkan kita ingin membuat, ketika tabel pemesanan (Sales.Orders) diisi, maka secara otomatis tabel detailnya diisi dengan data default, maka kita bisa menggunakan TRIGGER AFTER INSERT.

Ketikkan SQL berikut pada SSMS dan eksekusilah!

1

```
IF OBJECT_ID('Sales.trgAutoAddOrderDetailsForOrder') IS NOT NULL
    DROP TRIGGER Sales.trgAutoAddOrderDetailsForOrder;
GO;

CREATE TRIGGER trgAutoAddOrderDetailsForOrder ON Sales.Orders
AFTER INSERT
AS
    PRINT 'TRIGGER trgAutoAddOrderDetailsForOrder dipanggil!';

    DECLARE @orderid INT = (SELECT orderid FROM inserted);
    DECLARE @productid INT = 1;
    DECLARE @unitprice MONEY = 0;
    DECLARE @qty SMALLINT = 1;
    DECLARE @discount NUMERIC(4,3) = 0;

    INSERT INTO Sales.OrderDetails VALUES
        (@orderid, @productid, @unitprice, @qty, @discount);

    PRINT 'Data kosong ditambahkan secara otomatis ke tabel Sales.OrderDetails';
GO;
```

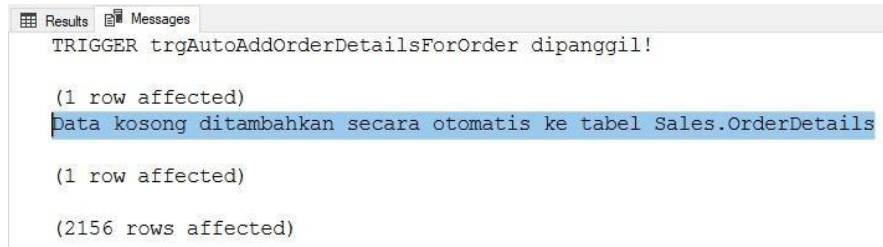
Jalankan SQL berikut untuk menambahkan data baru ke tabel Sales.Orders sehingga memicu ter-eksekusinya Trigger yang kita buat diatas tadi.


```

INSERT INTO Sales.Orders(
    custid, empid, orderdate, requireddate, shipperid, freight, shipname,
    shipaddress, shipcity, shipcountry)
VALUES (
    85, 5, GETDATE(), GETDATE(), 3, 100, 'Kapal Api',
    'Jl. Soekarno-Hata', 'Malang', 'Indonesia');

```

Jika benar, maka akan menampilkan pesan berikut:



Results Messages

TRIGGER trgAutoAddOrderDetailsForOrder dipanggil!

(1 row affected)

Data kosong ditambahkan secara otomatis ke tabel Sales.OrderDetails

(1 row affected)

(2156 rows affected)

Serta hasil seperti dibawah:



	orderid	productid	unitprice	qty	discount
1	11078	1	0.00	1	0.000
2	11077	77	13.00	2	0.000
3	11077	75	7.75	4	0.000
4	11077	73	15.00	2	0.010
5	11077	66	17.00	1	0.000
6	11077	64	33.25	2	0.030
7	11077	60	34.00	2	0.060
8	11077	55	24.00	2	0.000
9	11077	52	7.00	2	0.000
10	11077	46	12.00	3	0.020

MBP\SQLEXPRESS (... YUNHASNAWA-MBP\Yoppy Y... TSQL2012 00:00:00 2156 rows

2

Trigger **AFTER UPDATE**: Adalah trigger yang akan dieksekusi ketika ada operasi UPDATE berhasil (selesai, after) dilakukan pada tabel yang dipasang trigger tersebut.

Contoh kasus: Misalkan pada tabel 'Sales.OrderDetails' terdapat kolom 'unitprice' dimana kolom ini mengacu pada kolom yang sama pada 'Production.Product'. Akan tetapi, jika pada tabel 'Production.Products' kita ubah 'unitprice' sebuah produk, 'unitprice' yang ada di 'Sales.OrderDetails' tidak otomatis berubah. Agar harga di tabel 'OrderDetails' otomatis berubah ketika tabel 'Products' diupdate kita dapat menggunakan TRIGGER AFTER UPDATE.

Jalankan SQL berikut untuk membuat TRIGGER yang menyelesaikan contoh kasus diatas:


```

IF OBJECT_ID('Production.trgAutoUpdateOrderDetailsUnitPrice') IS NOT NULL
    DROP TRIGGER Production.trgAutoUpdateOrderDetailsUnitPrice;
GO;

CREATE TRIGGER trgAutoUpdateOrderDetailsUnitPrice ON Production.Products
AFTER UPDATE
AS
    PRINT 'Trigger trgAutoUpdateOrderDetailsUnitPrice DIPANGGIL!';

    DECLARE @productid INT = (SELECT productid FROM inserted);
    DECLARE @unitprice MONEY =
        COALESCE((SELECT unitprice FROM inserted), 0.0);

    UPDATE Sales.OrderDetails SET unitprice = @unitprice
    WHERE productid = @productid;

    PRINT 'Harga di tabel Sales.OrderDetails secara otomatis disesuaikan..';
GO;

```

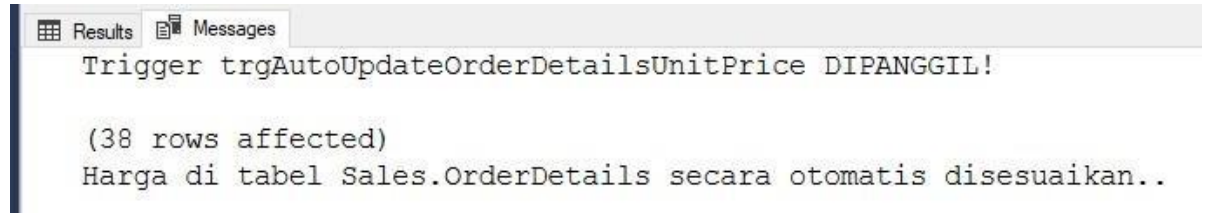
Eksekusilah SQL berikut, untuk mengetes TRIGGER yang telah Anda buat tadi:

```

UPDATE Production.Products SET unitprice = 100 WHERE productid = 11;
SELECT * FROM Production.Products WHERE productid = 11;
SELECT * FROM Sales.OrderDetails WHERE productid = 11;

```

Sehingga menghasilkan pesan seperti dibawah ini:



The screenshot shows the 'Messages' tab in SQL Server Enterprise Manager. It displays the following output from the trigger:

```

Trigger trgAutoUpdateOrderDetailsUnitPrice DIPANGGIL!

(38 rows affected)
Harga di tabel Sales.OrderDetails secara otomatis disesuaikan..

```

Dan hasil seperti berikut:

Results		Messages				
	productid	productname	supplierid	categoryid	unitprice	discontinued
1	11	Product QMVUN	5	4	100.00	0

	orderid	productid	unitprice	qty	discount
1	10248	11	100.00	12	0.000
2	10296	11	100.00	12	0.000
3	10327	11	100.00	50	0.200
4	10353	11	100.00	12	0.200
5	10365	11	100.00	24	0.000
6	10407	11	100.00	30	0.000
7	10434	11	100.00	6	0.000
8	10442	11	100.00	30	0.000
9	10443	11	100.00	6	0.200
10	10466	11	100.00	10	0.000
11	10486	11	100.00	5	0.000
12	10489	11	100.00	15	0.250
13	10528	11	100.00	3	0.000
14	10535	11	100.00	50	0.100
15	10542	11	100.00	15	0.050
16	10545	11	100.00	10	0.000
17	10553	11	100.00	15	0.000
18	10566	11	100.00	35	0.150
19	10570	11	100.00	15	0.050
20	10614	11	100.00	14	0.000
21	10637	11	100.00	10	0.000
22	10698	11	100.00	15	0.000
23	10726	11	100.00	5	0.000
24	10770	11	100.00	15	0.250

A-MBP\SQLEXPRESS (...)	YUNHASNAWA-MBP\Yoppy Y...	TSQL2012	00:00:00	38 rows
------------------------	---------------------------	----------	----------	---------

Trigger **AFTER DELETE**: Adalah TRIGGER yang dieksekusi ketika sebuah operasi DELETE dilakukan pada suatu tabel.

Contoh kasus: Perhatikan tabel 'Sales.OrderDetails', pada tabel tersebut terdapat kolom 'productid' yang merupakan Foreign Key yang mengacu pada tabel 'Production.Products'. Misalkan kita ingin supaya: ketika sebuah 'productid' dihapus semuanya dari tabel 'OrderDetails' maka kolom 'discontinued' diubah nilainya menjadi '1', kita dapat menggunakan TRIGGER AFTER DELETE.

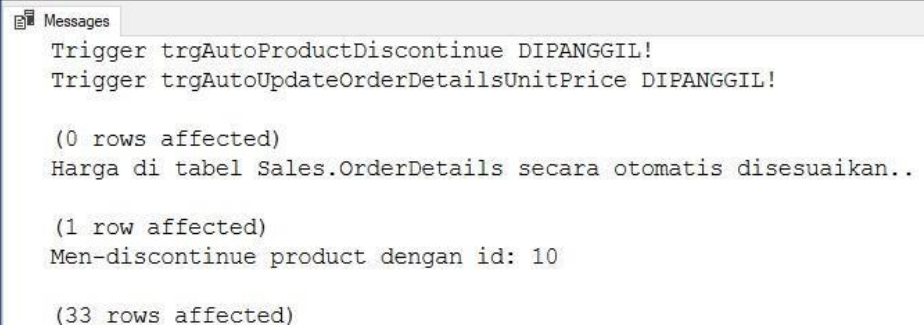
3

[Soal-12] Buatlah TRIGGER yang dapat menyelesaikan permasalahan pada contoh kasus diatas!

Lalu jalankan SQL berikut agar TRIGGER yang Anda buat tereksekusi:

```
DELETE FROM Sales.OrderDetails WHERE productid = 10;  
SELECT * FROM Production.Products WHERE productid = 10;
```

Pastikan **message**-nya seperti berikut:



Messages

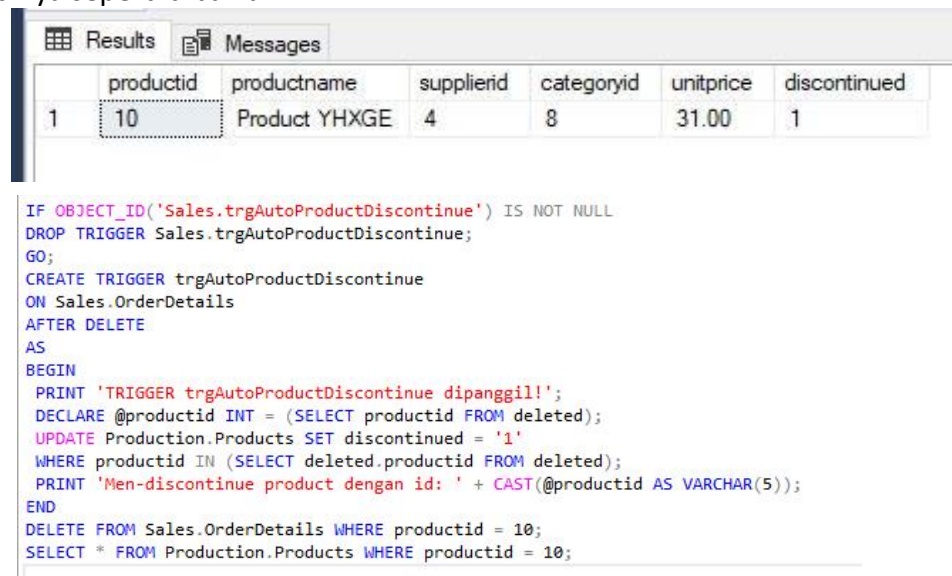
Trigger trgAutoProductDiscontinue DIPANGGIL!
Trigger trgAutoUpdateOrderDetailsUnitPrice DIPANGGIL!

(0 rows affected)
Harga di tabel Sales.OrderDetails secara otomatis disesuaikan..

(1 row affected)
Men-discontinue product dengan id: 10

(33 rows affected)

Dan **result**-nya seperti dibawah:



	productid	productname	supplierid	categoryid	unitprice	discontinued
1	10	Product YHXGE	4	8	31.00	1

```
IF OBJECT_ID('Sales.trgAutoProductDiscontinue') IS NOT NULL  
DROP TRIGGER Sales.trgAutoProductDiscontinue;  
GO;  
CREATE TRIGGER trgAutoProductDiscontinue  
ON Sales.OrderDetails  
AFTER DELETE  
AS  
BEGIN  
    PRINT 'TRIGGER trgAutoProductDiscontinue dipanggil!';  
    DECLARE @productid INT = (SELECT productid FROM deleted);  
    UPDATE Production.Products SET discontinued = '1'  
    WHERE productid IN (SELECT deleted.productid FROM deleted);  
    PRINT 'Men-discontinue product dengan id: ' + CAST(@productid AS VARCHAR(5));  
END  
DELETE FROM Sales.OrderDetails WHERE productid = 10;  
SELECT * FROM Production.Products WHERE productid = 10;
```

```
TRIGGER trgAutoProductDiscontinue dipanggil!  
TRIGGER trgAutoUpdateOrderDetailsUnitPrice dipanggil!
```

(0 rows affected)

Harga di tabel Sales.Orderdetails secara otomatis disesuaikan

(1 row affected)

Men-discontinue product dengan id: 10

(33 rows affected)

	productid	productname	supplierid	categoryid	unitprice	discontinued
1	10	Product YHXGE	4	8	31.00	1

Praktikum – Bagian 5: TRIGGER (INSTEAD OF)

Langkah	Keterangan
1	<p>Buat dulu tabel backup dengan cara membuka dan mengeksekusi file 'SQLQueryEmployeesBackup.sql' yang disertakan bersama jobsheet ini.</p> <p>Lokasi: <Folder Jobsheet>\Resources\SQLQuery-EmployeesBackup.sql</p> <p>Isi tabel HR.EmployeesBackup dengan isi yang sama persis dari tabel HR.Employees dengan cara mengeksekusi SQL berikut</p> <pre>INSERT INTO HR.EmployeesBackup (lastname, firstname, title, titleofcourtesy, birthdate, hiredate, [address], city, region,postalcode, country, phone, mgrid) SELECT lastname, firstname, title, titleofcourtesy, birthdate, hiredate, [address], city, region,postalcode, country, phone, mgrid FROM HR.Employees;</pre>

Trigger **INSTEAD OF INSERT**: Trigger ini akan mencegah user melakukan insert pada tabel 'HR.Employee', alih-alih membiarkan INSERT terjadi pada tabel tersebut, trigger berikut ini akan 'membelokkan' data yang diinsert ke tabel 'HR.EmployeesBackup' yang kita buat sebelumnya.

Buatlah TRIGGER yang menyelesaikan permasalahan diatas dengan mengeksekusi SQL berikut:

```
IF OBJECT_ID('HR.trgDivertInsertEmployeeToBackup') IS NOT NULL
    DROP TRIGGER HR.trgDivertInsertEmployeeToBackup
GO;

CREATE TRIGGER trgDivertInsertEmployeeToBackup ON HR.Employees
INSTEAD OF INSERT
AS
    PRINT 'TRIGGER trgDivertInsertEmployeeToBackup DIPANGGIL!';

    INSERT INTO HR.EmployeesBackup(
        lastname, firstname, title, titleofcourtesy, birthdate, hiredate,
        [address], city, region, postalcode, country, phone, mgrid)
    SELECT
        lastname, firstname, title, titleofcourtesy, birthdate, hiredate,
        [address], city, region, postalcode, country, phone, mgrid
    FROM inserted;

    PRINT 'Employee baru disimpan di tabel HR.EmployeesBackup..';
GO;
```

Lalu tes TRIGGER tadi dengan mengeksekusi SQL INSERT berikut:

```
INSERT INTO HR.Employees
VALUES
    ('Santoso', 'Adi', 'Staff', 'Mr. ', '19830101', '20170101',
    'Jl. Soekarno-Hatta', 'Malang', 'Jawa Timur', '65150', 'Indonesia',
    '(085) 123-456', 1)

SELECT * FROM HR.EmployeesBackup
```

Akan menghasilkan baris baru pada tabel 'EmployeesBackup' dan tabel 'Employees' tidak akan ada perubahan.

10	Santoso	Adi	Staff	Mr.	1983-01-01 00:00:00.000	2017-01-01 00:00:00.000	Jl. Soekarno-Hatta
----	---------	-----	-------	-----	-------------------------	-------------------------	--------------------

Trigger **INSTEAD OF UPDATE**: Mencegah user melakukan UPDATE pada suatu tabel.

[Soal-13] Dengan cara yang serupa dengan langkah sebelumnya, buatlah TRIGGER yang mencegah user melakukan UPDATE ke table 'HR.Employee'. Ketika ada UPDATE yang terjadi, terapkan hasilnya ke tabel 'HR.EmployeesBackup'!

Lalu jalankan SQL berikut agar TRIGGER yang Anda buat tereksekusi:

```
UPDATE HR.Employees SET firstname = 'DEPAN', lastname = 'BELAKANG'
WHERE firstname = 'Adi';
```

```
Messages
    TRIGGER trgDivertUpdateEmployeeToBackup DIPANGGIL!

    (1 row affected)
    Karyawan dengan empid: 10 yang ada di HR.EmployeesBackup yang diupdate.

    (1 row affected)
```

Apabila TRIGGER yang Anda buat benar maka *message*-nya akan tampil seperti berikut:

3

```
IF OBJECT_ID('HR.trgDivertUpdateEmployeeToBackup') IS NOT NULL
    DROP TRIGGER HR.trgDivertUpdateEmployeeToBackup;
GO;

CREATE TRIGGER trgDivertUpdateEmployeeToBackup ON HR.Employees
INSTEAD OF UPDATE
AS BEGIN
    PRINT 'TRIGGER trgDivertUpdateEmployeeToBackup DIPANGGIL!';
    DECLARE @empid INT = (SELECT empid FROM inserted);
    UPDATE HR.EmployeesBackup
    SET
        lastname = i.lastname,
        firstname = i.firstname,
        title = i.title,
        titleofcourtesy = i.titleofcourtesy,
        birthdate = i.birthdate,
        hiredate = i.hiredate,
        [address] = i.[address],
        city = i.city,
        region = i.region,
        postalcode = i.postalcode,
        country = i.country,
        phone = i.phone,
        mgrid = i.mgrid
    FROM inserted AS i
    WHERE HR.EmployeesBackup.empid = @empid;
    PRINT 'Karyawan dengan empid : ' + CAST(@empid AS VARCHAR(10)) +
        ' yang ada di HR.EmployeesBackup telah diupdate.';
END;
UPDATE HR.Employees SET firstname = 'DEPAN', lastname = 'BELAKANG'
WHERE firstname = 'Adi';

Commands completed successfully.

Completion time: 2023-11-02T19:35:28.7890845+07:00
```

<

TRIGGER tgrDivertDeleteEmployeeToBackup DIPANGGIL!

(1 row affected)

Karyawan dengan nama Maria Cameron dihapus di HR.EmployeesBackup saja. Di tabel aslinya tetap

(1 row affected)

Completion time: 2023-11-02T21:03:11.3974139+07:00

```
SELECT * FROM HR.Employees
SELECT * FROM HR.EmployeesBackup;
```

	empid	lastname	firstname	title	titleofcourtesy	birthdate	hiredate	address	city	region	postalcode
1	1	Davis	Sara	CEO	Ms.	1958-12-08 00:00:00.000	2002-05-01 00:00:00.000	7890 - 20th Ave. E., Apt. 2A	Seattle	WA	10003
2	2	Funk	Don	Vice President, Sales	Dr.	1962-02-19 00:00:00.000	2002-08-14 00:00:00.000	9012 W. Capital Way	Tacoma	WA	10001
3	3	Lew	Judy	Sales Manager	Ms.	1973-08-30 00:00:00.000	2002-04-01 00:00:00.000	2345 Moss Bay Blvd.	Kirkland	WA	10007
4	4	Peled	Yael	Sales Representative	Mrs.	1947-09-19 00:00:00.000	2003-05-03 00:00:00.000	5678 Old Redmond Rd.	Redmond	WA	10009
5	5	Buck	Sven	Sales Manager	Mr.	1965-03-04 00:00:00.000	2003-10-17 00:00:00.000	8901 Garrett Hill	London	NULL	10004
6	6	Suurs	Paul	Sales Representative	Mr.	1973-07-02 00:00:00.000	2003-10-17 00:00:00.000	3456 Coventry House, Miner Rd.	London	NULL	10005
7	7	King	Russell	Sales Representative	Mr.	1970-05-29 00:00:00.000	2004-01-02 00:00:00.000	6789 Edgeham Hollow, Winchester Way	London	NULL	10002
8	8	Cameron	Maria	Sales Representative	Ms.	1968-01-09 00:00:00.000	2004-03-05 00:00:00.000	4567 - 11th Ave. N.E.	Seattle	WA	10006
9	9	Dolgopyatova	Zoya	Sales Representative	Ms.	1976-01-27 00:00:00.000	2004-11-15 00:00:00.000	1234 Houndstooth Rd.	London	NULL	10008

	empid	lastname	firstname	title	titleofcourtesy	birthdate	hiredate	address	city	region	postalcode	country	phone	mgrid
1	1	Davis	Sara	CEO	Ms.	1958-12-08 00:00:00.000	2002-05-01 00:00:00.000	7890 - 20th Ave. E., Apt. 2A	Seattle	WA	10003	USA	(206) 555-0101	NULL
2	2	Funk	Don	Vic...	Dr.	1962-02-19 00:00:00.000	2002-08-14 00:00:00.000	9012 W. Capital Way	Taco...	WA	10001	USA	(206) 555-0100	1
3	3	Lew	Judy	Sal...	Ms.	1973-08-30 00:00:00.000	2002-04-01 00:00:00.000	2345 Moss Bay Blvd.	Kirkla...	WA	10007	USA	(206) 555-0103	2
4	4	Peled	Yael	Sal...	Mrs.	1947-09-19 00:00:00.000	2003-05-03 00:00:00.000	5678 Old Redmond Rd.	Red...	WA	10009	USA	(206) 555-0104	3
5	5	Buck	Sven	Sal...	Mr.	1965-03-04 00:00:00.000	2003-10-17 00:00:00.000	8901 Garrett Hill	Lond...	NULL	10004	UK	(71) 234-5678	2
6	6	Suurs	Paul	Sal...	Mr.	1973-07-02 00:00:00.000	2003-10-17 00:00:00.000	3456 Coventry House, Mi...	Lond...	NULL	10005	UK	(71) 345-6789	5
7	7	King	Russell	Sal...	Mr.	1970-05-29 00:00:00.000	2004-01-02 00:00:00.000	6789 Edgeham Hollow, W...	Lond...	NULL	10002	UK	(71) 123-4567	5
8	9	Dolgop...	Zoya	Sal...	Ms.	1976-01-27 00:00:00.000	2004-11-15 00:00:00.000	1234 Houndstooth Rd.	Lond...	NULL	10008	UK	(71) 456-7890	5

--- Selamat Mengerjakan ---