

LAPORAN HASIL PRAKTIKUM

BASIS DATA JOBSHEET 10



OLEH:

Bagas Satria YN 04/2341760108

SIB 2B

D-IV SISTEM INFORMASI BISNIS JURUSAN
TEKNOLOGI INFORMASI POLITEKNIK NEGERI
MALANG



Praktikum – Bagian 1: UNION & UNION ALL

Langkah	Keterangan
1	<p>Berikut ini adalah sebuah SQL kueri ke tabel 'Production.Products' yang akan menampilkan 'productid' dan 'productname', khusus bagi product yang memiliki 'categoryid' bernilai 4!</p> <pre>SELECT productid, productname FROM Production.Products WHERE categoryid = 4;</pre> <p>Ketik dan eksekusi SQL tersebut dan pastikan hasilnya sesuai dengan gambar berikut:</p>

Results Messages

	productid	productname
1	11	Product QMVUN
2	12	Product OSFNS
3	31	Product XWOXC
4	32	Product NUNAW
5	33	Product ASTMN
6	59	Product UKXRI
7	60	Product WHBYK
8	69	Product COAXA
9	71	Product MYMOI
10	72	Product GEEEO

```
--1 union & union all  
SELECT  
    productid,  
    productname  
FROM  
    Production.Products  
WHERE  
    categoryid = 4;
```

184 %

Results Messages

	productid	productname
1	11	Product QMVUN
2	12	Product OSFNS
3	31	Product XWOXC
4	32	Product NUNAW
5	33	Product ASTMN
6	59	Product UKXRI
7	60	Product WHBYK
8	69	Product COAXA
9	71	Product MYMOI
10	72	Product GEEEO

2	SQL berikut ini adalah SQL yang menampilkan 'productid' dan 'productname' dari tabel 'Production.Products'. Hasil dari query ini di- <i>filter</i> sedemikian rupa sehingga yang tampil hanyalah product yang telah memiliki nilai jual total lebih dari \$50.000.

```
SELECT
    P.productid,
    P.productname
FROM
    Production.Products P INNER JOIN Sales.OrderDetails OD
ON
    P.productid = OD.productid
GROUP BY
    P.productid, P.productname
HAVING
    SUM(OD.qty * OD.unitprice) > 50000;
```

Keterangan: Untuk mendapatkan nilai jual total, SQL diatas bekerja dengan cara sebagai berikut:

1. Meng-Inner-joinkan tabel 'Production.Products' dengan tabel 'Sales.OrderDetails' karena data penjualan ada di tabel yang terakhir.
2. Melakukan GROUP BY, berdasarkan 'productid' dan 'productname'-nya
3. Dan yang terakhir, mem-*filter* grup menggunakan HAVING dengan kondisi data yang **totalNilaiPenjualannya > 50000**
4. Dimana **totalNilaiPenjualan** = ('unitprice' × 'qty')

Eksekusilah SQL diatas tadi dan pastikan hasilnya sesuai dengan gambar berikut:



	productid	productname
1	29	Product VJXYN
2	38	Product QDOMO
3	59	Product UKXRI
4	60	Product WHBYK

```

SELECT
    P.productid,
    P.productname
FROM
    Production.Products P INNER JOIN Sales.OrderDetail
ON
    P.productid = OD.productid
GROUP BY
    P.productid, P.productname
HAVING
    SUM(OD.qty * OD.unitprice) > 50000;

```

114 %

Results Messages

	productid	productname
1	29	Product VJXYN
2	38	Product QDOMO
3	59	Product UKXRI
4	60	Product WHBYK

3

[Soal-1] Tulis sebuah SQL yang menampilkan hasil pada praktikum-1 langkah-1 & 2 secara sekaligus (gabungan) dengan menggunakan **UNION**!

Petunjuk: Letakkan UNION diantara kedua SQL tersebut.

Pastikan hasilnya sesuai dengan gambar berikut:

```

SELECT
    productid,
    productname
FROM
    Production.Products
WHERE
    categoryid = 4

UNION

SELECT
    P.productid,
    P.productname
FROM
    Production.Products P
INNER JOIN
    Sales.OrderDetails OD ON P.productid = OD.productid
GROUP BY
    P.productid,
    P.productname
HAVING
    SUM(OD.qty * OD.unitprice) > 50000;

--soal 2

```

64 %

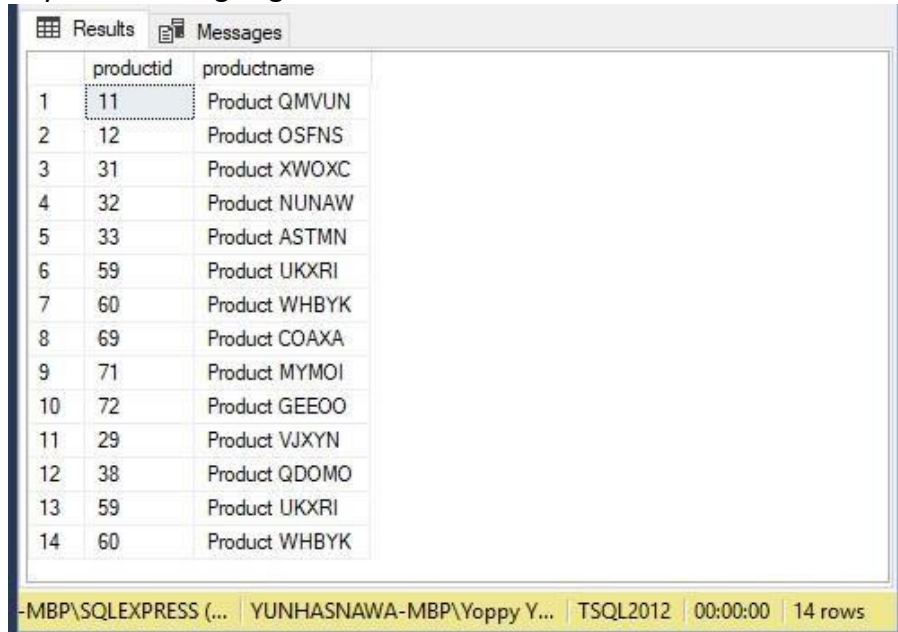
Results Messages

	productid	productname
1	11	Product QMVUN
2	12	Product OSFNS
3	29	Product VJXYN
4	31	Product XWOXC
5	32	Product NUNAW
6	33	Product ASTMN
7	38	Product QDOMO
8	59	Product UKXRI
9	60	Product WHBYK
10	69	Product COAXA
11	71	Product MYMOI

[Soal-2] Serupa dengan langkah sebelumnya, kali ini tulislah sebuah SQL yang menampilkan hasil pada praktikum-1 langkah-1 & 2 secara sekaligus (gabungan) dengan menggunakan **UNION ALL**!

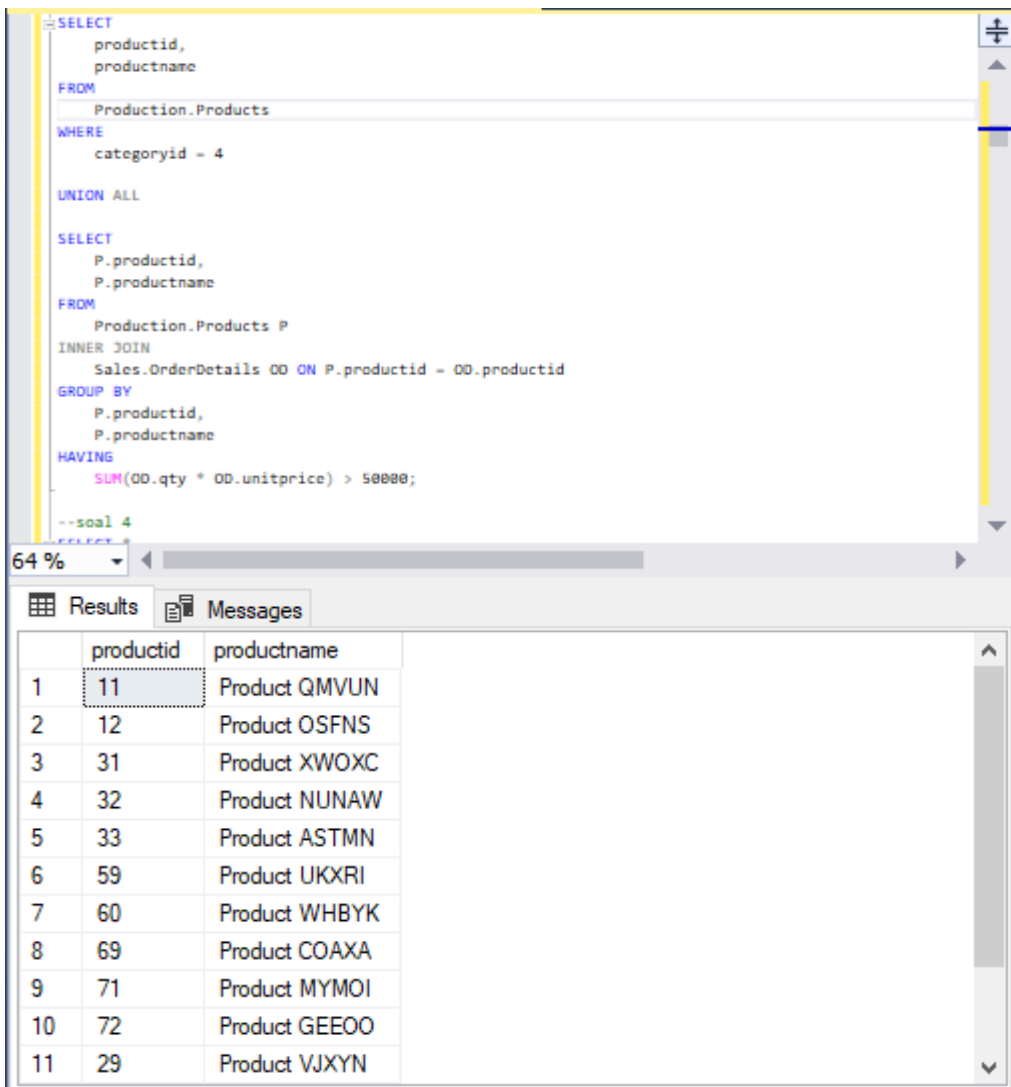
Pastikan hasilnya sesuai dengan gambar berikut:

4



	productid	productname
1	11	Product QMVUN
2	12	Product OSFNS
3	31	Product XWOXC
4	32	Product NUNAW
5	33	Product ASTMN
6	59	Product UKXRI
7	60	Product WHBYK
8	69	Product COAXA
9	71	Product MYMOI
10	72	Product GEEEO
11	29	Product VJXYN
12	38	Product QDOMO
13	59	Product UKXRI
14	60	Product WHBYK

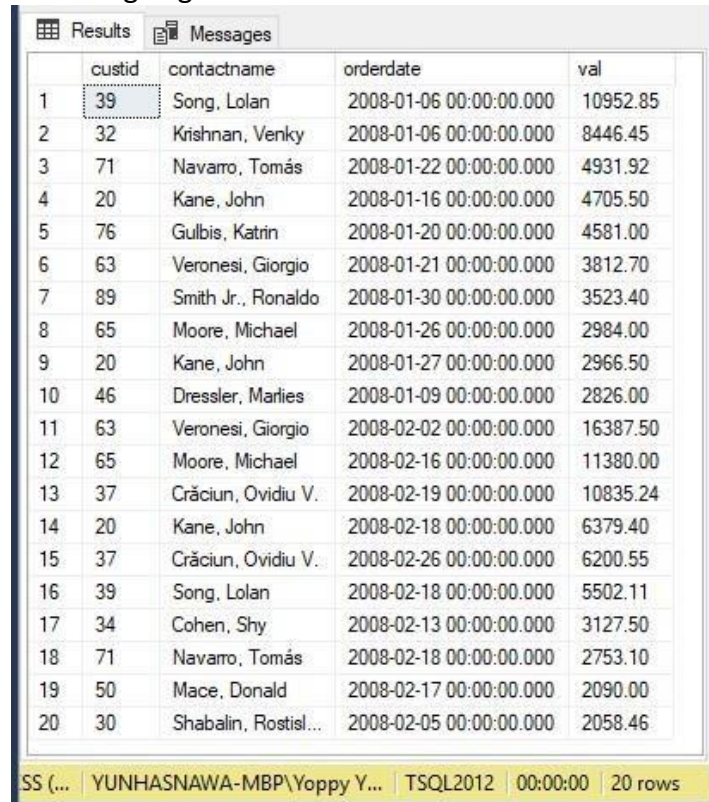
-MBP\SQLEXPRESS (... | YUNHASNAWA-MBP\Yoppy Y... | TSQL2012 | 00:00:00 | 14 rows

	 <pre>SELECT productid, productname FROM Production.Products WHERE categoryid = 4 UNION ALL SELECT P.productid, P.productname FROM Production.Products P INNER JOIN Sales.OrderDetails OD ON P.productid = OD.productid GROUP BY P.productid, P.productname HAVING SUM(OD.qty * OD.unitprice) > 50000; --soal 4 SELECT</pre> <table><thead><tr><th></th><th>productid</th><th>productname</th></tr></thead><tbody><tr><td>1</td><td>11</td><td>Product QMVUN</td></tr><tr><td>2</td><td>12</td><td>Product OSFNS</td></tr><tr><td>3</td><td>31</td><td>Product XWOXC</td></tr><tr><td>4</td><td>32</td><td>Product NUNAW</td></tr><tr><td>5</td><td>33</td><td>Product ASTMN</td></tr><tr><td>6</td><td>59</td><td>Product UKXRI</td></tr><tr><td>7</td><td>60</td><td>Product WHBYK</td></tr><tr><td>8</td><td>69</td><td>Product COAXA</td></tr><tr><td>9</td><td>71</td><td>Product MYMOI</td></tr><tr><td>10</td><td>72</td><td>Product GEEEO</td></tr><tr><td>11</td><td>29</td><td>Product VJXYN</td></tr></tbody></table>		productid	productname	1	11	Product QMVUN	2	12	Product OSFNS	3	31	Product XWOXC	4	32	Product NUNAW	5	33	Product ASTMN	6	59	Product UKXRI	7	60	Product WHBYK	8	69	Product COAXA	9	71	Product MYMOI	10	72	Product GEEEO	11	29	Product VJXYN
	productid	productname																																			
1	11	Product QMVUN																																			
2	12	Product OSFNS																																			
3	31	Product XWOXC																																			
4	32	Product NUNAW																																			
5	33	Product ASTMN																																			
6	59	Product UKXRI																																			
7	60	Product WHBYK																																			
8	69	Product COAXA																																			
9	71	Product MYMOI																																			
10	72	Product GEEEO																																			
11	29	Product VJXYN																																			
5	<p>[Soal-3] Apa bedanya UNION & UNION ALL?</p> <p>UNION menghapus data duplikat, lebih lambat.</p> <p>UNION ALL menyertakan semua data (termasuk duplikat), lebih cepat.</p>																																				
6	<p>[Soal-4] Tuliskan SQL untuk menampilkan 10 pelanggan dengan nilai pembelian tertinggi pada bulan Januari 2008 serta 10 tertinggi pada bulan Februari 2008.</p>																																				

Petunjuk:

1. Buat dahulu query untuk menampilkan data yang bulan-nya Januari lalu UNION-kan dengan bulan Februari.
2. Pada tiap-tiap bulan lakukan INNER JOIN antara tabel 'Sales.Customers' & 'Sales.OrderValue'

Pastikan hasilnya sesuai dengan gambar berikut:



	custid	contactname	orderdate	val
1	39	Song, Lolan	2008-01-06 00:00:00.000	10952.85
2	32	Krishnan, Venky	2008-01-06 00:00:00.000	8446.45
3	71	Navarro, Tomás	2008-01-22 00:00:00.000	4931.92
4	20	Kane, John	2008-01-16 00:00:00.000	4705.50
5	76	Gulbis, Katrin	2008-01-20 00:00:00.000	4581.00
6	63	Veronesi, Giorgio	2008-01-21 00:00:00.000	3812.70
7	89	Smith Jr., Ronaldo	2008-01-30 00:00:00.000	3523.40
8	65	Moore, Michael	2008-01-26 00:00:00.000	2984.00
9	20	Kane, John	2008-01-27 00:00:00.000	2966.50
10	46	Dressler, Marlies	2008-01-09 00:00:00.000	2826.00
11	63	Veronesi, Giorgio	2008-02-02 00:00:00.000	16387.50
12	65	Moore, Michael	2008-02-16 00:00:00.000	11380.00
13	37	Crăciun, Ovidiu V.	2008-02-19 00:00:00.000	10835.24
14	20	Kane, John	2008-02-18 00:00:00.000	6379.40
15	37	Crăciun, Ovidiu V.	2008-02-26 00:00:00.000	6200.55
16	39	Song, Lolan	2008-02-18 00:00:00.000	5502.11
17	34	Cohen, Shy	2008-02-13 00:00:00.000	3127.50
18	71	Navarro, Tomás	2008-02-18 00:00:00.000	2753.10
19	50	Mace, Donald	2008-02-17 00:00:00.000	2090.00
20	30	Shabalín, Rostislav	2008-02-05 00:00:00.000	2058.46

SS (... | YUNHASNAWA-MBP\Yoppy Y... | TSQL2012 | 00:00:00 | 20 rows

--soal 4

```

SELECT *
FROM
  (SELECT TOP 10
    c.custid, c.contactname, o.orderdate, o.val
  FROM
    Sales.Customers AS c
  INNER JOIN
    Sales.OrderValues AS o
  ON
    c.custid = o.custid
  WHERE
    MONTH (orderdate) = 01 AND YEAR (orderdate) = 2008
  ORDER BY
    o.val DESC)
  AS a
UNION ALL
SELECT *
FROM
  (SELECT TOP 10
    c.custid, c.contactname, o.orderdate, o.val
  FROM
    Sales.Customers AS c
  INNER JOIN
    Sales.OrderValues AS o
  ON
    c.custid = o.custid
  WHERE
    MONTH (orderdate) = 02 AND YEAR (orderdate) = 2008
  ORDER BY
    o.val DESC)
  AS b

```

64 %

Results Messages

	custid	contactname	orderdate	val
1	39	Song, Lolan	2008-01-06 00:00:00.000	10952.85
2	32	Krishnan, Venky	2008-01-06 00:00:00.000	8446.45
3	71	Navarro, Tomás	2008-01-22 00:00:00.000	4931.92
4	20	Kane, John	2008-01-16 00:00:00.000	4705.50
5	76	Gulbis, Katrin	2008-01-20 00:00:00.000	4581.00
6	63	Veronesi, Giorgio	2008-01-21 00:00:00.000	3812.70
7	89	Smith Jr., Ronaldo	2008-01-30 00:00:00.000	3523.40
8	65	Moore, Michael	2008-01-26 00:00:00.000	2984.00
9	20	Kane, John	2008-01-27 00:00:00.000	2966.50
10	46	Dressler, Marlies	2008-01-09 00:00:00.000	2826.00
11	63	Veronesi, Giorgio	2008-02-02 00:00:00.000	16387.50

Praktikum – Bagian 2: CROSS APPLY & OUTER APPLY

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

APPLY: Adalah fasilitas yang memungkinkan kita untuk menerapkan inner-query maupun TVF (Table-Valued Function) ke setiap hasil yang didapat oleh outer-querynya.

APPLY ada 2:

1. CROSS APPLY: Hanya mengembalikan baris yang inner-query atau TVF-nya ada nilainya
2. OUTER APPLY: Mengembalikan baris baik yang inner-query atau TVF-nya ada nilainya maupun tidak. Kalau tidak ada hasilnya, maka digandeng-kan dengan NULL.

Supaya Anda paham, ketik dan eksekusilah SQL berikut ini lalu perhatikan hasilnya!\

1

```
SELECT p.productid, p.productname, o.orderid
FROM Production.Products AS p
CROSS APPLY
(
    SELECT TOP(2)
    d.orderid
    FROM Sales.OrderDetails AS d
    WHERE d.productid = p.productid
    ORDER BY d.orderid DESC
) AS o
ORDER BY p.productid;
```

Pastikan hasilnya sesuai dengan gambar berikut:



```
--Bagian 2 cross apply & outer apply
--1
SELECT p.productid, p.productname, o.orderid
FROM Production.Products AS p
CROSS APPLY
(
    SELECT TOP(2)
        d.orderid
    FROM Sales.OrderDetails AS d
    WHERE d.productid = p.productid
    ORDER BY d.orderid DESC
) AS o
ORDER BY p.productid;

--2
IF OBJECT_ID('dbo.fnGetTop3ProductsForCustomer') IS NOT NULL
```

94 %

Results Messages

	productid	productname	orderid
1	1	Product HHYDP	11070
2	1	Product HHYDP	11047
3	2	Product RECZE	11077
4	2	Product RECZE	11075
5	3	Product IMEHJ	11077
6	3	Product IMEHJ	11017
7	4	Product KSBRM	11077
8	4	Product KSBRM	11000
9	5	Product EPEIM	11047
10	5	Product EPEIM	11030
11	6	Product VAIIV	11077

Results		Messages	
	productid	productname	orderid
1	1	Product HHYDP	11070
2	1	Product HHYDP	11047
3	2	Product RECZE	11077
4	2	Product RECZE	11075
5	3	Product IMEHJ	11077
6	3	Product IMEHJ	11017
7	4	Product KSBRM	11077
8	4	Product KSBRM	11000
9	5	Product EPEIM	11047
10	5	Product EPEIM	11030
11	6	Product VAIIV	11077
12	6	Product VAIIV	11076
13	7	Product HMLNI	11077
14	7	Product HMLNI	11071
15	8	Product WVJFP	11077
16	8	Product WVJFP	11007
17	9	Product AOZB...	10848
18	9	Product AOZB...	10693
19	10	Product YHXGE	11077
20	10	Product YHXGE	11020
21	11	Product QMV...	11073
22	11	Product QMV...	11043

QLEXPRESS (... | YUNHASNAWA-MBP\Yoppy Y... | TSQL2012 | 00:00:00 | 154 rows

```
--Bagian 2 cross apply & outer apply
--1
SELECT p.productid, p.productname, o.orderid
FROM Production.Products AS p
CROSS APPLY
(
    SELECT TOP(2)
        d.orderid
    FROM Sales.OrderDetails AS d
    WHERE d.productid = p.productid
    ORDER BY d.orderid DESC
) AS o
ORDER BY p.productid;
--2
-- IF OBJECT_ID('dbo.fnGetTop3ProductsForCustomer') IS NOT NULL
```

	productid	productname	orderid
1	1	Product HHYDP	11070
2	1	Product HHYDP	11047
3	2	Product RECZE	11077
4	2	Product RECZE	11075
5	3	Product IMEHJ	11077
6	3	Product IMEHJ	11017
7	4	Product KSBRM	11077
8	4	Product KSBRM	11000
9	5	Product EPEIM	11047
10	5	Product EPEIM	11030
11	6	Product VALIV	11077

Keterangan: SQL diatas menampilkan 'orderid' dan 'productname' dari produk-produk yang ada di tabel 'Production.Products' dengan mencantumkan 'orderid' dari 2 pemesan terakhir yang melibatkan masing-masing produk.

Untuk memahami penggunaan APPLY dengan TVF, tulis dan eksekusilah SQL berikut ini:

```
IF OBJECT_ID('dbo.fnGetTop3ProductsForCustomer') IS NOT NULL
    DROP FUNCTION dbo.fnGetTop3ProductsForCustomer;
GO
CREATE FUNCTION dbo.fnGetTop3ProductsForCustomer (@custid AS INT)
RETURNS TABLE
AS
    RETURN
        SELECT TOP(3)
            d.productid,
            p.productname,
            SUM(d.qty * d.unitprice) AS totalsalesamount
        FROM Sales.Orders AS o
        INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid
        INNER JOIN Production.Products AS p ON p.productid = d.productid
        WHERE custid = @custid
        GROUP BY d.productid, p.productname
    ORDER BY totalsalesamount DESC;
GO
```

2



```
--2
IF OBJECT_ID('dbo.fnGetTop3ProductsForCustomer') IS NOT NULL
    DROP FUNCTION dbo.fnGetTop3ProductsForCustomer;
GO
CREATE FUNCTION dbo.fnGetTop3ProductsForCustomer (@custid AS INT)
RETURNS TABLE
AS
RETURN
    SELECT TOP(3)
        d.productid,
        p.productname,
        SUM(d.qty * d.unitprice) AS totalsalesamount
    FROM Sales.Orders AS o
    INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid
    INNER JOIN Production.Products AS p ON p.productid = d.productid
    WHERE custid = @custid
    GROUP BY d.productid, p.productname
    ORDER BY totalsalesamount DESC;
GO
```

78 %

Messages

Commands completed successfully.

Completion time: 2024-11-13T09:29:11.7783381+07:00

78 %

[Soal-5] Ubahlah CROSS APPLY pada Praktikum bagian-2.2 menjadi **OUTER APPLY**, sehingga hasilnya menjadi seperti gambar berikut:

```
--soal 5
SELECT
    c.custid,
    c.contactname,
    p.productid,
    p.productname,
    p.totalsalesamount
FROM
    Sales.Customers AS c
OUTER APPLY
    dbo.fnGetTop3ProductsForCustomer(c.custid) AS p
ORDER BY
    c.custid;
```

	custid	contactname	productid	productname	totalsalesamount
1	1	Allen, Michael	63	Product ICKNK	878,00
2	1	Allen, Michael	59	Product UKXRI	825,00
3	1	Allen, Michael	28	Product OFBNT	775,20
4	2	Hassall, Mark	72	Product GEEOO	348,00
5	2	Hassall, Mark	60	Product WHBYK	340,00
6	2	Hassall, Mark	32	Product NUNAW	320,00
7	3	Peoples, John	11	Product QMVUN	1453,20
8	3	Peoples, John	26	Product HLGZA	936,90
9	3	Peoples, John	59	Product UKXRI	825,00
10	4	Amdt, Torsten	20	Product QHFFP	4050,00
11	4	Amdt, Torsten	31	Product XWOXC	1437,50

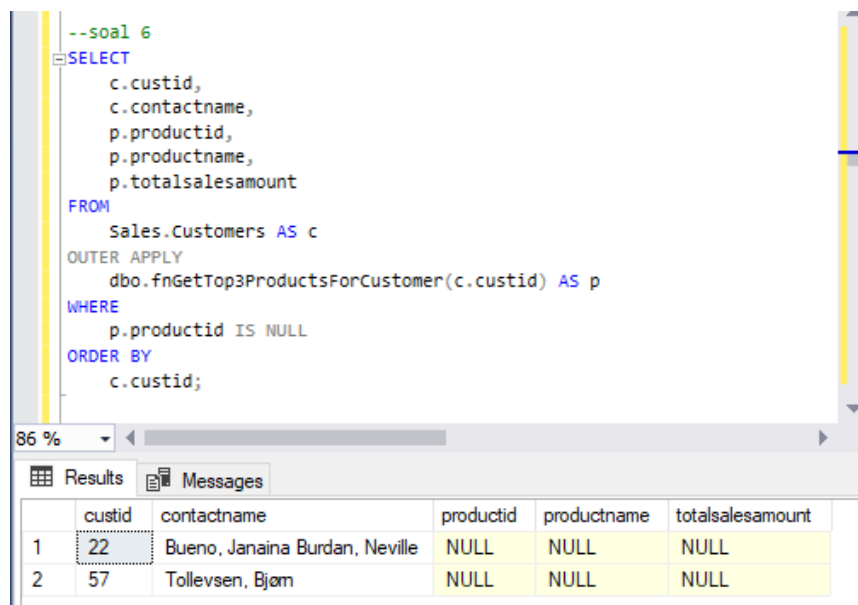
3

[Soal-6] Modifikasilah SQL yang telah Anda buat dari bagian sebelumnya sehingga SQL tersebut HANYA menampilkan customer yang tidak pernah membeli produk.

Petunjuk: Tambahkan WHERE <?> IS NULL. Dimana <?> adalah sebuah nama kolom.

Pastikan hasilnya seperti gambar berikut:

4


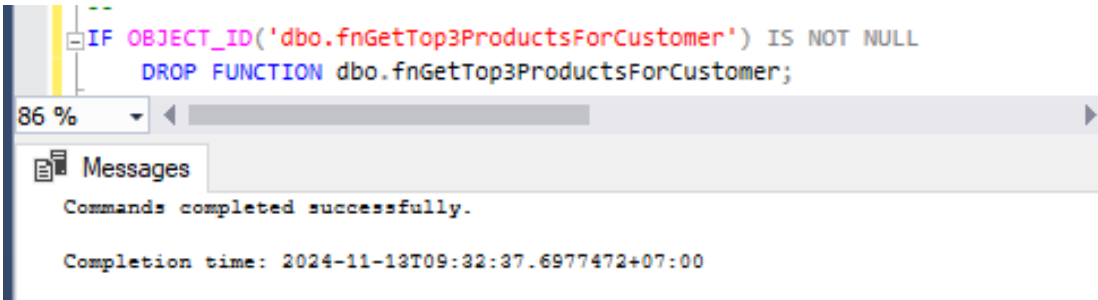


```
--soal 6
SELECT
    c.custid,
    c.contactname,
    p.productid,
    p.productname,
    p.totalsalesamount
FROM
    Sales.Customers AS c
OUTER APPLY
    dbo.fnGetTop3ProductsForCustomer(c.custid) AS p
WHERE
    p.productid IS NULL
ORDER BY
    c.custid;
```

86 %

Results Messages

	custid	contactname	productid	productname	totalsalesamount
1	22	Bueno, Janaina Burdan, Neville	NULL	NULL	NULL
2	57	Tollefsen, Bjørn	NULL	NULL	NULL

	
5	<p>Hapus fungsi yang dibuat pada praktikum bagian 2.2 dengan mengeksekusi SQL berikut</p> <pre>IF OBJECT_ID('dbo.fnGetTop3ProductsForCustomer') IS NOT NULL DROP FUNCTION dbo.fnGetTop3ProductsForCustomer;</pre> 

Praktikum – Bagian 3: EXCEPT & INTERSECT

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

Berikut ini adalah sebuah statement SELECT yang menampilkan kolom 'custid' dari tabel Sales.Orders. Dimana hasilnya difilter lagi sehingga yang tampil hanyalah pelanggan yang sudah membeli lebih dari 20 produk yang berbeda (berdasarkan kolom 'productid' dari tabel 'Sales.OrderDetails').

```
SELECT o.custid
FROM Sales.Orders AS o
INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid
GROUP BY o.custid
HAVING COUNT(DISTINCT d.productid) > 20;
```

Ketik dan eksekusilah SQL tersebut sehingga hasilnya serupa dengan gambar berikut:

1

Results		Messages	
	custid		
1	4		
2	5		
3	7		
4	9		
5	10		
6	20		
7	24		
8	25		
9	30		
10	34		
11	35		
12	37		
13	39		
14	41		
15	44		
16	46		
17	47		
18	51		
19	56		
20	62		
21	63		
22	65		

-MBP\SQLEXPRESS (... | YUNHASNAWA-MBP\Yoppy Y... | TSQL2012 | 00:00:00 | 33 rows

```
--Bagian 3 EXCEPT & INTERSECT
SELECT o.custid
FROM Sales.Orders AS o
INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid
GROUP BY o.custid
HAVING COUNT(DISTINCT d.productid) > 20;
```

94 %

Results		Messages	
	custid		
1	4		
2	5		
3	7		
4	9		
5	10		
6	20		
7	24		
8	25		
9	30		
10	34		
11	35		

2	<p>[Soal-7] Buatlah sebuah statement SELECT yang menampilkan kolom 'custid' dari tabel 'Sales.Orders'. Saring hasilnya sehingga yang tampil hanyalah pelanggan yang berasal dari USA kecuali SEMUA pelanggan yang muncul pada hasil query pada praktikum bagian 3.1.</p> <p>Petunjuk: Tambahkan sebuah query untuk mendapatkan customer dari USA dan tambahkan operator EXCEPT didepan query praktikum-3 langkah-1.</p> <p>Pastikan hasilnya seperti pada gambar berikut:</p>
---	---

```
--soal 7
-- Query untuk menampilkan pelanggan dari USA
SELECT o.custid
FROM Sales.Orders AS o
INNER JOIN Sales.Customers AS c ON o.custid = c.custid
WHERE c.country = 'USA'
```

104 %

Results Messages

	custid
1	32
2	32
3	32
4	32
5	32
6	32
7	32
8	32
9	32
10	32
11	32

3	<p>Berikut ini adalah sebuah statement SELECT yang menampilkan kolom 'custid' dari tabel 'Sales.Orders'. Hasilnya kemudian di-filter sedemikian rupa sehingga hanya customer yang telah berbelanja lebih dari \$10.000 yang tampil. Nilai belanja customer-customer tersebut didapatkan dari perkalian kolom 'qty' dan 'unitprice' yang ada di tabel 'Sales.OrderDetails'.</p> <pre>SELECT o.custid FROM Sales.Orders AS o INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid GROUP BY o.custid HAVING SUM(d.qty * d.unitprice) > 10000;</pre> <p>Ketik dan eksekusi SQL diatas lalu pastikan hasilnya seperti pada gambar berikut:</p>
---	--

Results		Messages
	custid	
1	23	
2	46	
3	75	
4	9	
5	89	
6	72	
7	32	
8	35	
9	86	
10	63	
11	55	
12	67	
13	87	
14	7	
15	44	
16	50	
17	24	
18	47	
19	30	
20	10	
21	41	
22	4	

-MBP\SQLEXPRESS (... | YUNHASNAWA-MBP\Yoppy Y... | TSQL2012 | 00:00:00 | 41 rows

```
--  
SELECT o.custid  
FROM Sales.Orders AS o  
INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid  
GROUP BY o.custid  
HAVING SUM(d.qty * d.unitprice) > 10000;
```

104 %

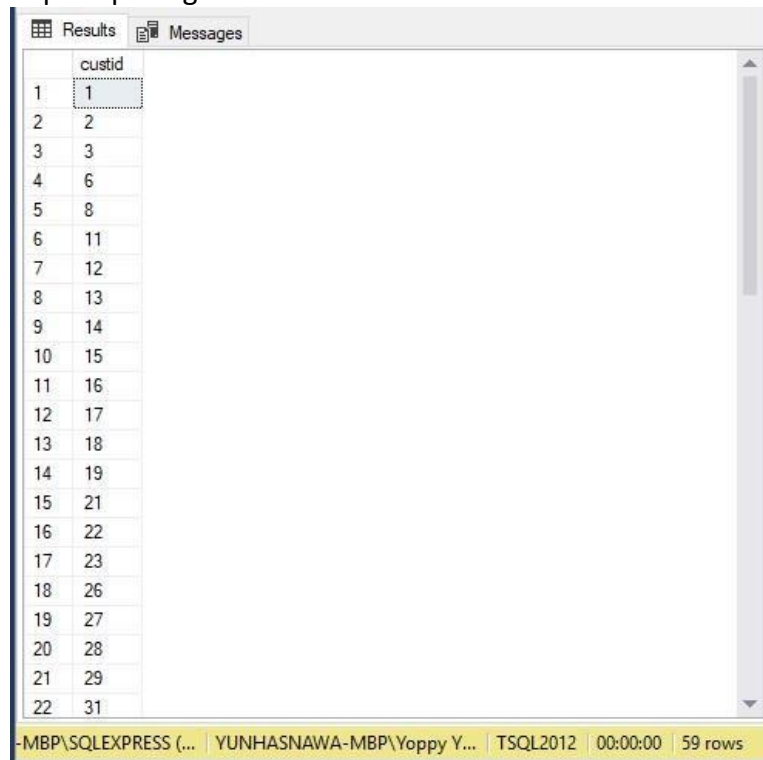
Results Messages

	custid
1	23
2	46
3	75
4	9
5	89
6	72
7	32
8	35
9	86
10	63
11	55

4

[Soal-8] Salin SQL pada bagian 3.1 tambahkan operator INTERSECT dibelakangnya, kemudian salin-tempel SQL pada bagian 3.3 dibelakang operator INTERSECT tadi. Jalankan, dan perhatikan hasilnya.

Pastikan hasilnya seperti pada gambar berikut:



	custid
1	1
2	2
3	3
4	6
5	8
6	11
7	12
8	13
9	14
10	15
11	16
12	17
13	18
14	19
15	21
16	22
17	23
18	26
19	27
20	28
21	29
22	31

Results Messages

-MBP\SQLEXPRESS (... YUNHASNAWA-MBP\Yoppy Y... TSQL2012 00:00:00 59 rows

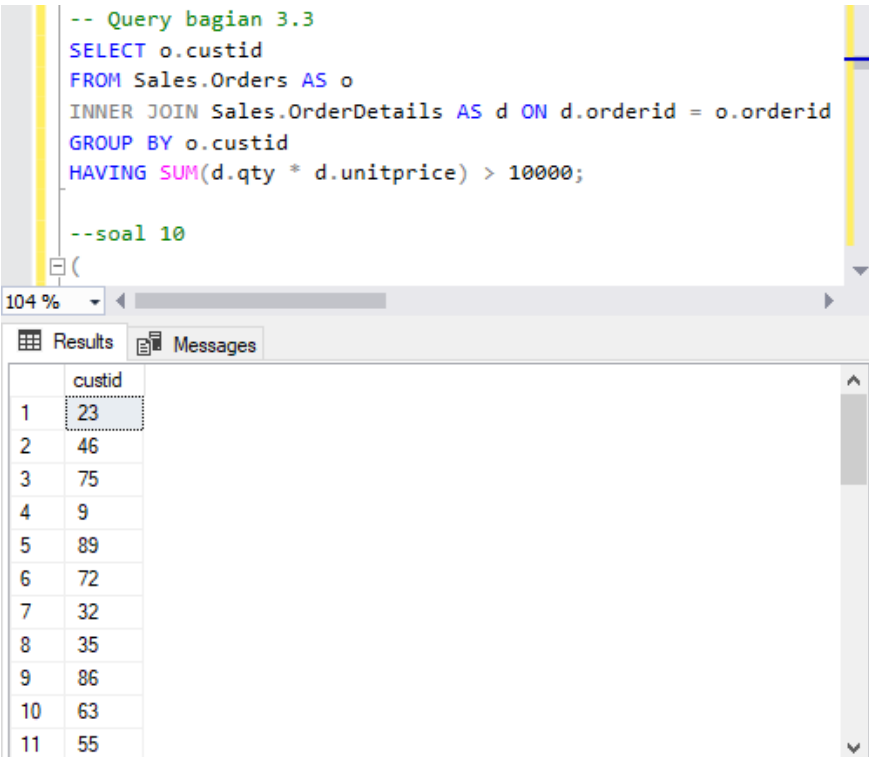
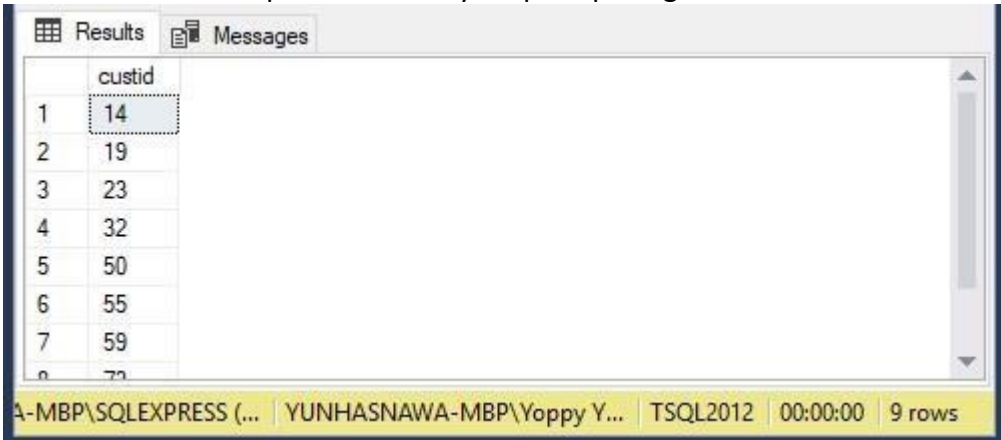
```
--soal 8
SELECT c.custid
FROM Sales.Customers AS c
EXCEPT
SELECT o.custid
FROM Sales.Orders AS o
INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid
GROUP BY o.custid
HAVING COUNT(DISTINCT d.productid)>20
-- Menambahkan INTERSECT
INTERSECT
```

104 %

Results Messages

	custid
1	1
2	2
3	3
4	6
5	8
6	11
7	12
8	13
9	14
10	15
11	16

[Soal-9] Dapatkah Anda menyimpulkan, customer yang bagaimana yang tampil pada hasil query bagian 3.4 ini?

	<pre>-- Query bagian 3.3 SELECT o.custid FROM Sales.Orders AS o INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid GROUP BY o.custid HAVING SUM(d.qty * d.unitprice) > 10000; --soal 10 (</pre> 
5	<p>[Soal-10] Salin keseluruhan query pada bagian-3.4, modifikasi SQL tersebut dengan cara mengagipit statement SELECT sebelum operator INTERSECT dengan tanda kurung '(' dan ')'. Eksekusilah SQL tersebut dan pastikan hasilnya seperti pada gambar berikut:</p> 


```
--soal 10
(
    SELECT c.custid
    FROM Sales.Customers AS c
    EXCEPT
    SELECT o.custid
    FROM Sales.Orders AS o
    INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid
    GROUP BY o.custid
    HAVING COUNT(DISTINCT d.productid) > 20
)
INTERSECT
(
    SELECT o.custid
    FROM Sales.Orders AS o
    INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid
    GROUP BY o.custid
    HAVING SUM(d.qty * d.unitprice) > 10000
);
```

78 %

Results Messages

	custid
1	14
2	19
3	23
4	32
5	50
6	55
7	59
8	73
9	75

[Soal-11] Apakah hasilnya berbeda dengan SQL pada bagian-3.4? Mengapa demikian? Dapatkah Anda menjelaskan tentang urutan prioritas (*precedence*) operator yang digunakan pada SQL bagian ini?

```
--Bagian 4 TRIGGER(AFTER)
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 called!';

    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
```

71 %

Messages

Commands completed successfully.

Completion time: 2024-11-13T09:42:18.8772288+07:00

Praktikum – Bagian 4: TRIGGER (AFTER)

Langkah	Keterangan
1	<p>TRIGGER: Trigger adalah semacam stored procedure (fungsi yang tidak mengembalikan nilai) spesial yang akan dieksekusi ketika ada sebuah event yang terjadi pada suatu tabel.</p> <p>Trigger ada 2:</p> <ul style="list-style-type: none"> - TRIGGER AFTER: Trigger yang MENAMBAHKAN suatu aksi - TRIGGER INSTEAD OF: Trigger yang MENCEGAH suatu aksi <p>Trigger AFTER INSERT: Adalah trigger yang akan dieksekusi ketika ada operasi INSERT berhasil (selesai, after) dilakukan pada tabel yang dipasang trigger tersebut.</p> <p>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.</p> <p>Ketikkan SQL berikut pada SSMS dan eksekusilah!</p> <pre>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;</pre> <p>Jalankan SQL berikut untuk menambahkan data baru ke tabel Sales.Orders sehingga memicu ter-eksekusinya Trigger yang kita buat diatas tadi.</p>

```
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,
akan

maka

Results Messages

TRIGGER trgAutoAddOrderDetailsForOrder dipanggil!

(1 row affected)

Data kosong ditambahkan secara otomatis ke tabel Sales.OrderDetails

(1 row affected)

(2156 rows affected)

menampilkan pesan berikut:

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

```

--
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-Hatta', 'Malang', 'Indonesia');
--

SELECT orderid, productid, unitprice, qty, discount
FROM Sales.OrderDetails
ORDER BY orderid;

```

71 %

Results Messages

	orderid	productid	unitprice	qty	discount
1	10248	11	14,00	12	0.000
2	10248	42	9,80	10	0.000
3	10248	72	34,80	5	0.000
4	10249	14	18,60	9	0.000
5	10249	51	42,40	40	0.000
6	10250	41	7,70	10	0.000
7	10250	51	42,40	35	0.150
8	10250	65	16,80	15	0.150
9	10251	22	16,80	6	0.050
10	10251	57	15,60	15	0.050
11	10251	65	16,80	20	0.000

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);

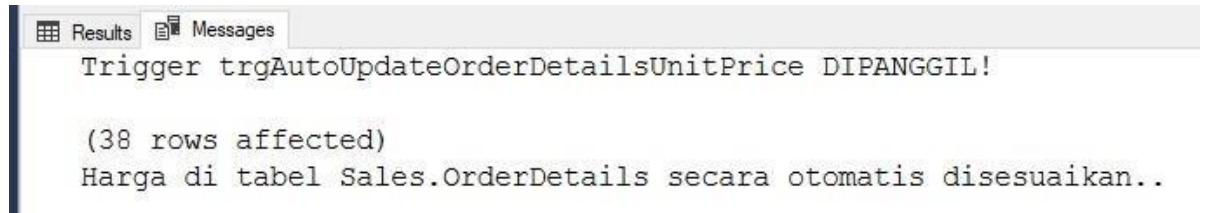
    UPDATE Sales.OrderDetails SET unitprice = @unitprice
    WHERE productid = @productid;

    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:

```
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


```

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

--soal 12

```

86 %

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	14,00	12	0.000
2	10296	11	16,80	12	0.000
3	10327	11	16,80	50	0.200
4	10353	11	16,80	12	0.200
5	10365	11	16,80	24	0.000
6	10407	11	16,80	30	0.000
7	10434	11	16,80	6	0.000
8	10442	11	16,80	30	0.000

3

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.

[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:

Results

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

```

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

```

Results

	productid	productname	supplierid	categoryid	unitprice	discontinued
1	10	Product YHXGE	4	8	31,00	0

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;
```

2

```

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

```

71 %

Messages

Commands completed successfully.

Completion time: 2024-11-13T10:17:49.1146849+07:00

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.

```
--
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
```

86 %

Results

empid	lastname	firstname	title	titleofcourtesy	birthdate	hiredate	address	city	region	postalcode	country	pho
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) 556-1111
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	USA	(206) 561-7222
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	USA	(206) 835-4545
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	USA	(206) 751-9876
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	UK	(7) 534-5678
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	UK	(7) 534-5678
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	UK	(7) 534-5678
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	USA	(206) 751-9876
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	UK	(7) 534-5678
10	santoso	Adi	Staff	Mr.	1983-01-01 00:00:00.000	2017-01-01 00:00:00.000	Jl. Soekarno-Hatta	Malang	Jawa Timur	65150	Indonesia	(085) 123-456

Query executed successfully.

BEBI\DBMS22 (16.0 RTM) | BEBI\bebi findia (69) | TSQL | 00:00:00 | 10 rows

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';
```

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

3

```
-- Sample UPDATE statement that triggers the trigger
UPDATE HR.Employees
SET firstname = 'DEPAN', lastname = 'BELAKANG'
WHERE firstname = 'Adi';
```

86 %

Messages

(0 rows affected)

Completion time: 2024-11-13T10:22:33.5871846+07:00

	<div data-bbox="316 293 1522 555"> <div>Messages</div> <p>TRIGGER trgDivertUpdateEmployeeToBackup DIPANGGIL!</p> <p>(1 row affected)</p> <p>Karyawan dengan empid: 10 yang ada di HR.EmployeesBackup yang diupdate.</p> <p>(1 row affected)</p> </div>
4	<p>Trigger INSTEAD OF DELETE: Mencegah user melakukan DELETE pada suatu tabel.</p> <p>[Soal-14] Buatlah TRIGGER yang mencegah user melakukan DELETE ke table 'HR.Employee'. Ketika ada DELETE yang terjadi, jangan biarkan ada data pada tabel tersebut yang hilang! Hapus data yang sama 'HR.EmployeesBackup'!</p> <p>Lalu jalankan SQL berikut agar TRIGGER yang Anda buat tereksekusi:</p> <div data-bbox="316 1442 1161 1957"> <pre>--soal 14 IF OBJECT_ID('HR.trgDivertDeleteEmployeeToBackup') IS NOT NULL DROP TRIGGER HR.trgDivertDeleteEmployeeToBackup; GO CREATE TRIGGER trgDivertDeleteEmployeeToBackup ON HR.Employees INSTEAD OF DELETE AS PRINT 'TRIGGER trgDivertDeleteEmployeeToBackup DIPANGGIL!'; PRINT 'Karyawan dengan nama : Maria Cameron dihapus di HR.EmployeesBac'; GO; DELETE FROM HR.Employees WHERE firstname = 'Maria'</pre> <div>86 %</div> <div>Messages</div> <p>Commands completed successfully.</p> <p>Completion time: 2024-11-13T10:23:25.9581043+07:00</p> </div>

```
DELETE FROM HR.Employees WHERE firstname = 'Maria'
SELECT * FROM HR.EmployeesBackup;
```

Messages

TRIGGER trgDivertDeleteEmployeeToBackup DIPANGGIL!

(1 row affected)

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

(1 row affected)

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

```
DELETE FROM HR.Employees WHERE firstname = 'Maria'
SELECT * FROM HR.EmployeesBackup;
```

6 %

Results

Messages

Msg 547, Level 16, State 0, Line 441

The DELETE statement conflicted with the REFERENCE constraint "FK_Orders_Employee".
The statement has been terminated.

(10 rows affected)

Completion time: 2024-11-13T10:24:37.5331470+07:00

Dan akan menghapus 1 baris pada table *backup*, sementara di tabel aslinya datanya tetap ada.

--- Selamat Mengerjakan ---