

JOBSHEET

PRAKTIKUM BASIS DATA LANJUT

Jurusan Teknologi Informasi
POLITEKNIK NEGERI MALANG



PERTEMUAN 10

SQL SERVER - OPERASI HIMPUNAN & TRIGGER



Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet 10: Operasi Himpunan & Trigger

Mata Kuliah Basis Data Lanjut

Pengampu: Tim Ajar Basis Data Lanjut

Tujuan

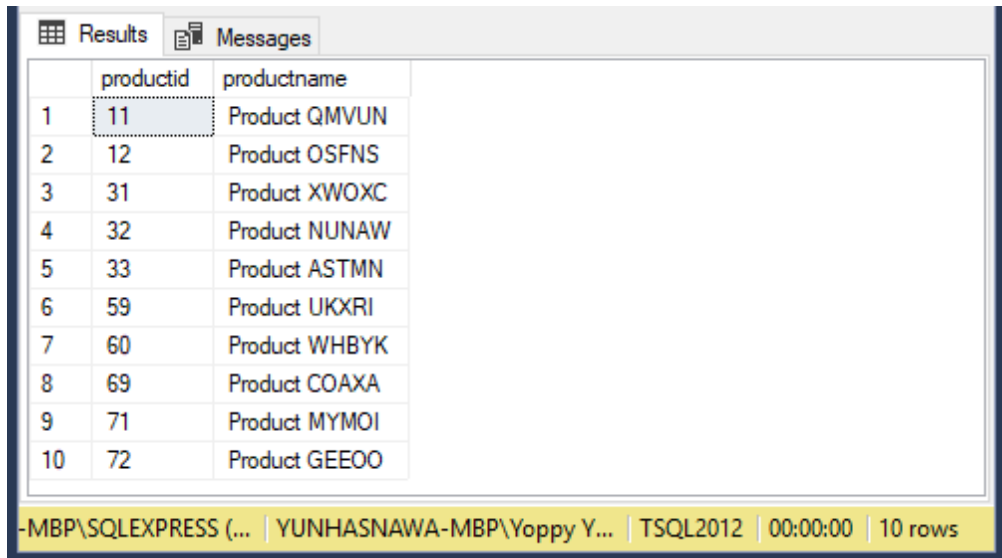
Mahasiswa diharapkan dapat:

1. Menerapkan query UNION dan UNION ALL
2. Menerapkan query CROSS APPLY dan OUTER APPLY
3. Menerapkan query EXCEPT dan INTERSECT
4. Menerapkan query TRIGGER (AFTER)
5. Menerapkan query TRIGGER (INSTEAD OF)

Petunjuk Umum

1. Ikuti langkah-langkah pada bagian-bagian praktikum sesuai dengan urutan yang diberikan.
2. Jawablah semua pertanyaan bertanda **[Soal-X]** yang terdapat pada langkah-langkah tertentu di setiap bagian praktikum.
3. 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.
4. 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:
 - **10_Prak.BDL_Kelas_NamaLengkapAnda.pdf**
 - Kumpulkan file PDF tersebut sebagai laporan praktikum kepada dosen pengampu.
 - Selain pada nama file, cantumkan juga identitas Anda pada tiap footer halaman laporan tersebut.

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>  <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></tbody></table> <p>-MBP\SQLEXPRESS (... YUNHASNAWA-MBP\Yoppy Y... TSQL2012 00:00:00 10 rows</p> | | 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 |
| | 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 | <p>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.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

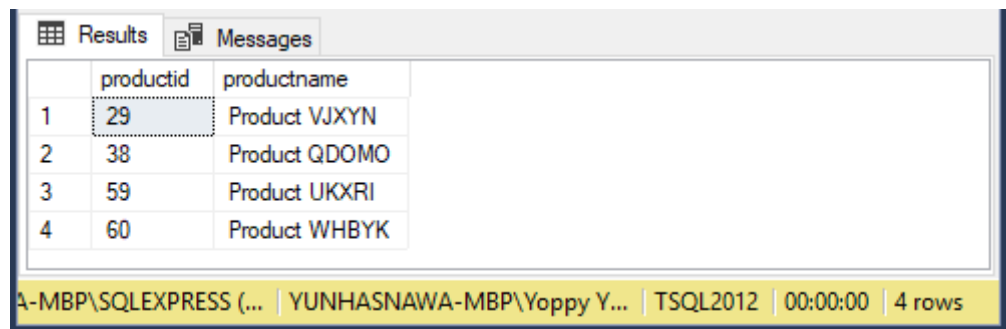


```
SELECT
    P.productid,
    P.productname
FROM
    Production.Products P INNER JOIN Sales.OrderDetails OD
ON
    P.productid = OD.productid
GROUP BY
    P.productid, P.productname
HAVING
    SUM(OD.qty * OD.unitprice) > 50000;
```

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 |

A-MBP\SQLEXPRESS (... | YUNHASNAWA-MBP\Yoppy Y... | TSQL2012 | 00:00:00 | 4 rows

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:

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

-MBP\SQLEXPRESS (... | YUNHASNAWA-MBP\Yoppy Y... | TSQL2012 | 00:00:00 | 12 rows

Jawab :

SQLQuery1.sql - L...4MO8C\Lenovo (71))*

```

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;
  
```

86 %

| 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 |
| 12 | 72 | Product GEEEO |

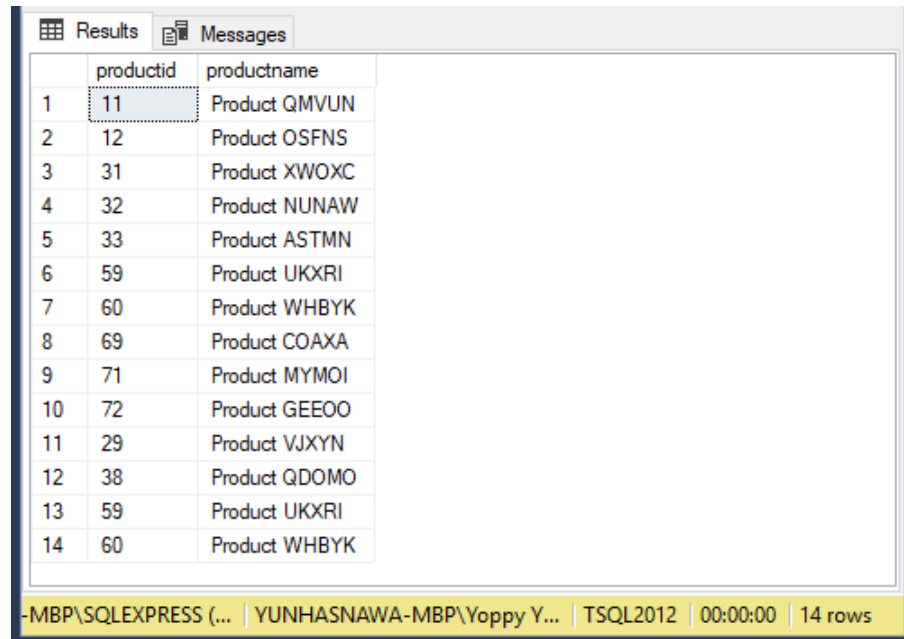
Qu LAPTOP-95A4MO8C (16.0 RTM) | LAPTOP-95A4MO8C\Lenovo... | TSQL | 00:00:00 | 12 rows

4

[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:



| | 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 GEE00 |
| 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

Jawab :



SQLQuery1.sql - L...4MO8C\Lenovo (71))*

```
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;
```

86 %

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 GEEOO |
| 11 | 29 | Product VJXYN |
| 12 | 38 | Product QDOMO |
| 13 | 59 | Product UKXRI |
| 14 | 60 | Product WHBYK |

Qu | LAPTOP-95A4MO8C (16.0 RTM) | LAPTOP-95A4MO8C\Lenovo... | TSQL | 00:00:00 | 14 rows

5

[Soal-3] Apa bedanya UNION & UNION ALL?

Jawab :

Pada UNION menghilangkan baris duplikat dalam hasil gabungan. Hanya baris unik yang akan ditampilkan, sedangkan UNION ALL menampilkan semua hasil dari pernyataan SELECT, termasuk yang duplikat. Dengan kata lain, semua baris dari masing-masing hasil query akan digabungkan tanpa penghilangan duplikat.

6

[Soal-4] Tuliskan SQL untuk menampilkan 10 pelanggan dengan nilai pembelian tertinggi pada bulan Januari 2008 serta 10 tertinggi pada bulan Februari 2008.

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:

| 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 |
| 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

Jawab :



```
SELECT TOP 10
    c.custid,
    c.contactname,
    o.orderdate,
    o.val
FROM
    Sales.Customers c INNER JOIN Sales.OrderValues o
ON
    c.custid = o.custid
WHERE
    MONTH(o.orderdate) = 1 AND YEAR(o.orderdate) = 2008
UNION
SELECT TOP 10
    c.custid,
    c.contactname,
    o.orderdate,
    o.val
FROM
    Sales.Customers c INNER JOIN Sales.OrderValues o
ON
    c.custid = o.custid
WHERE
    MONTH(o.orderdate) = 2 AND YEAR(o.orderdate) = 2008
ORDER BY
    o.val DESC;
```

86 %

Results Messages

| | custid | contactname | orderdate | val |
|---|--------|---------------------------|-------------------------|----------|
| 1 | 63 | Veronesi, Giorgio | 2008-02-02 00:00:00.000 | 16387.50 |
| 2 | 39 | Song, Lolan | 2008-01-06 00:00:00.000 | 10952.85 |
| 3 | 32 | Krishnan, Venky | 2008-01-06 00:00:00.000 | 8446.45 |
| 4 | 30 | Shabalin, Rostislav | 2008-02-05 00:00:00.000 | 2058.46 |
| 5 | 9 | Raghav, Amritansh | 2008-02-05 00:00:00.000 | 1979.23 |
| 6 | 62 | Misiec, Anna | 2008-02-04 00:00:00.000 | 1920.60 |
| 7 | 84 | Tuntisangaroon, Sittichai | 2008-01-05 00:00:00.000 | 1788.45 |

Qu LAPTOP-95A4MO8C (16.0 RTM) LAPTOP-95A4MO8C\Lenovo... TSQL 00:00:00 20 rows

Praktikum – Bagian 2: CROSS APPLY & OUTER APPLY

| Langkah | Keterangan |
|---------|--|
| 1 | <p>APPLY: Adalah fasilitas yang memungkinkan kita untuk menerapkan inner-query maupun TVF (Table-Valued Function) ke setiap hasil yang didapat oleh outer-querynya.</p> <p>APPLY ada 2:</p> <ol style="list-style-type: none"> 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. <p>Supaya Anda paham, ketik dan eksekusilah SQL berikut ini lalu perhatikan hasilnya!\</p> <pre> 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; </pre> <p>Pastikan hasilnya sesuai dengan gambar berikut:</p> |



| 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 KSB RM | 11077 |
| 8 | 4 | Product KSB RM | 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

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

2

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
```

Eksekusilah SQL diatas terlebih dahulu supaya fungsinya tersimpan di database.

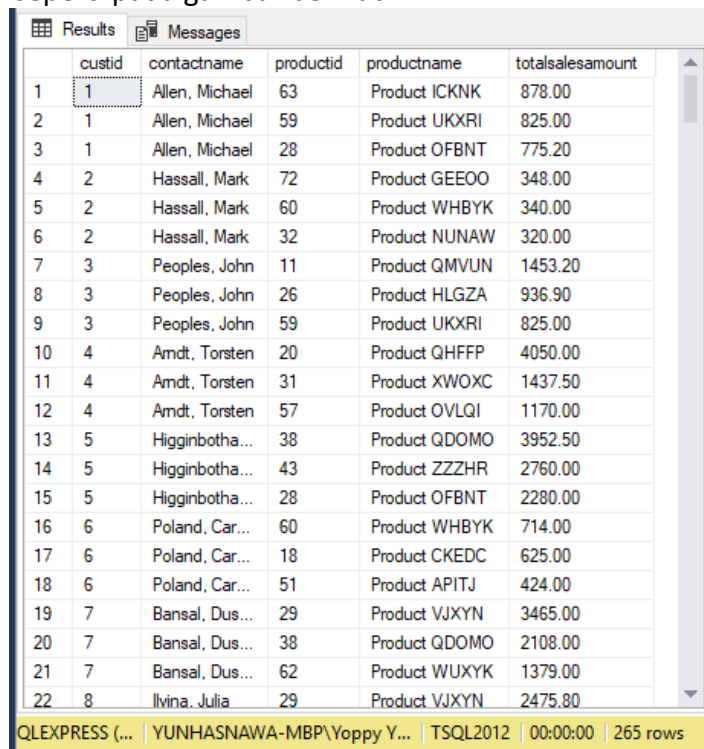
Kemudian tulis dan jalankan SQL dibawah ini untuk menampilkan data 3 pembelian teratas yang dilakukan oleh customer pada tabel 'Sales.Customers' dimana data tersebut meliputi 'productid', 'productname', & 'totalsalesamount'.

```
SELECT
    c.custid, c.contactname, p.productid, p.productname, p.totalsalesamou
FROM
    Sales.Customers AS c
CROSS APPLY
    dbo.fnGetTop3ProductsForCustomer (c.custid) AS p
ORDER BY
    c.custid;
```

Keterangan:

1. Pada SQL tersebut digunakan fungsi '**fnGetTop3ProductsForCustomer**' yang dibuat sebelumnya untuk mendapatkan 3 baris 'productid', 'productname', 'totalsalesamount' milik masing-masing customer.
2. Pada SQL tersebut digunakan **CROSS APPLY**, sehingga setiap 'custid' yang dihasilkan oleh outer-querynya, akan mendapatkan 3 hasil karena fungsi '**fnGetTop3ProductsForCustomer**' mengembalikan 3 hasil.

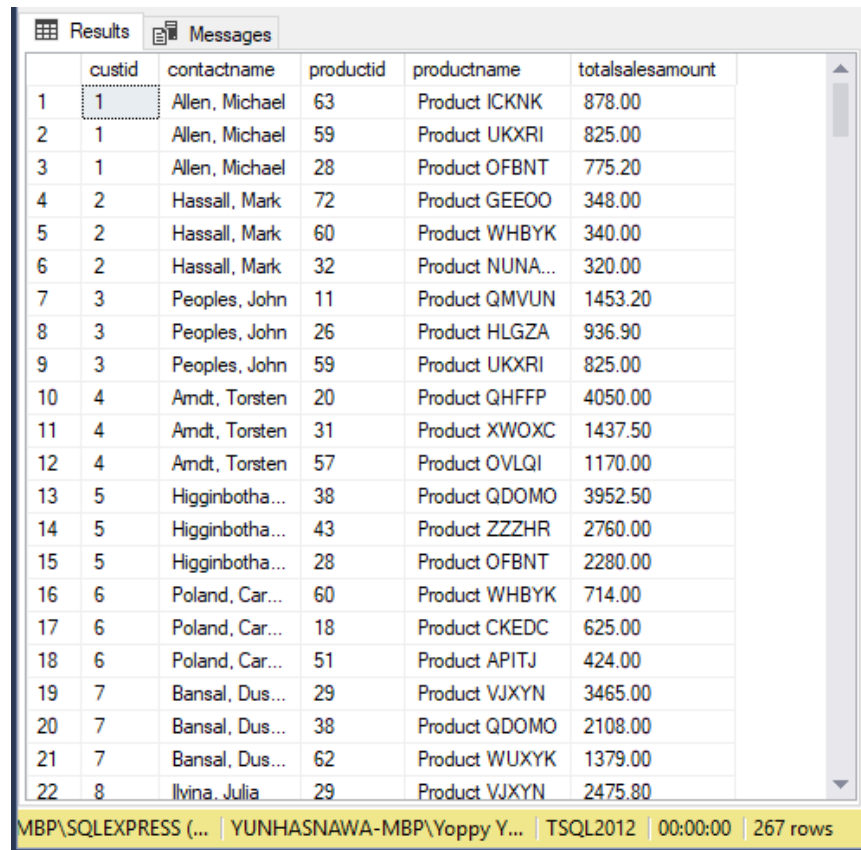
Pastikan hasilnya seperti pada gambar berikut:



| | 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 GEEEO | 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 |
| 12 | 4 | Amdt, Torsten | 57 | Product OVLQI | 1170.00 |
| 13 | 5 | Higginbotha... | 38 | Product QDOMO | 3952.50 |
| 14 | 5 | Higginbotha... | 43 | Product ZZZHR | 2760.00 |
| 15 | 5 | Higginbotha... | 28 | Product OFBNT | 2280.00 |
| 16 | 6 | Poland, Car... | 60 | Product WHBYK | 714.00 |
| 17 | 6 | Poland, Car... | 18 | Product CKEDC | 625.00 |
| 18 | 6 | Poland, Car... | 51 | Product APITJ | 424.00 |
| 19 | 7 | Bansal, Dus... | 29 | Product VJXYN | 3465.00 |
| 20 | 7 | Bansal, Dus... | 38 | Product QDOMO | 2108.00 |
| 21 | 7 | Bansal, Dus... | 62 | Product WUXYK | 1379.00 |
| 22 | 8 | Ilvina, Julia | 29 | Product VJXYN | 2475.80 |

QLEXPRESS (... | YUNHASNAWA-MBP\Yoppy Y... | TSQL2012 | 00:00:00 | 265 rows

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

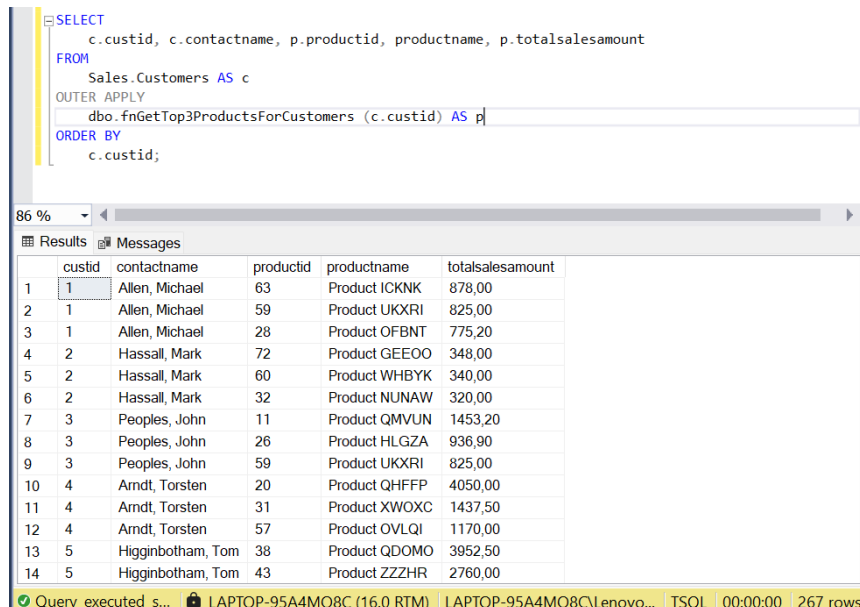


| | custid | contactname | productid | productname | totalsalesamount |
|----|--------|----------------|-----------|-----------------|------------------|
| 1 | 1 | Allen, Michael | 63 | Product ICKNK | 878.00 |
| 2 | 1 | Allen, Michael | 59 | Product UKXRI | 825.00 |
| 3 | 1 | Allen, Michael | 28 | Product OFBNT | 775.20 |
| 4 | 2 | Hassall, Mark | 72 | Product GEEEO | 348.00 |
| 5 | 2 | Hassall, Mark | 60 | Product WHBYK | 340.00 |
| 6 | 2 | Hassall, Mark | 32 | Product NUNA... | 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 |
| 12 | 4 | Amdt, Torsten | 57 | Product OVLQI | 1170.00 |
| 13 | 5 | Higginbotha... | 38 | Product QDOMO | 3952.50 |
| 14 | 5 | Higginbotha... | 43 | Product ZZZHR | 2760.00 |
| 15 | 5 | Higginbotha... | 28 | Product OFBNT | 2280.00 |
| 16 | 6 | Poland, Car... | 60 | Product WHBYK | 714.00 |
| 17 | 6 | Poland, Car... | 18 | Product CKEDC | 625.00 |
| 18 | 6 | Poland, Car... | 51 | Product APITJ | 424.00 |
| 19 | 7 | Bansal, Dus... | 29 | Product VJXYN | 3465.00 |
| 20 | 7 | Bansal, Dus... | 38 | Product QDOMO | 2108.00 |
| 21 | 7 | Bansal, Dus... | 62 | Product WUXYK | 1379.00 |
| 22 | 8 | Ilvina, Julia | 29 | Product VJXYN | 2475.80 |

MBP\SQLEXPRESS (... | YUNHASNAWA-MBP\Yoppy Y... | TSQL2012 | 00:00:00 | 267 rows

Perhatikan hasilnya, sekarang ada 267 baris. 2 baris tambahan yang muncul tersebut adalah pelanggan-pelanggan yang belum pernah melakukan pembelian, sehingga fungsi 'fnGetTop3ProductsForCustomer()' mengembalikan nilai **NULL**.

Jawab :



```

SELECT
    c.custid, c.contactname, p.productid, productname, p.totalsalesamount
FROM
    Sales.Customers AS c
OUTER APPLY
    dbo.fnGetTop3ProductsForCustomers (c.custid) AS p
ORDER BY
    c.custid;

```

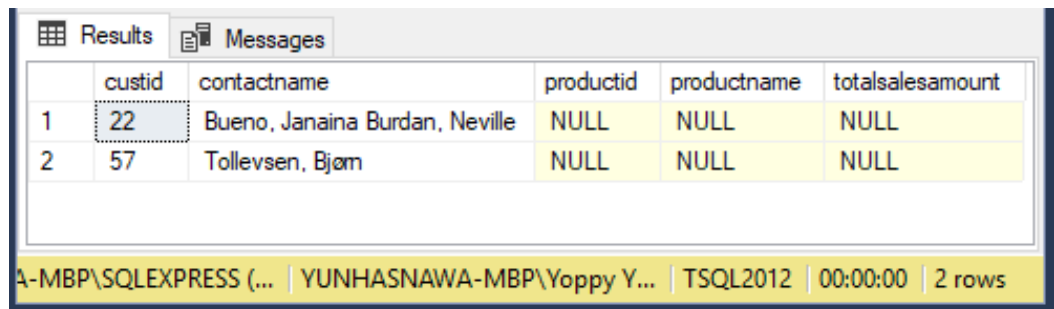
| | custid | contactname | productid | productname | totalsalesamount |
|----|--------|-------------------|-----------|---------------|------------------|
| 1 | 1 | Allen, Michael | 63 | Product ICKNK | 878,00 |
| 2 | 1 | Allen, Michael | 59 | Product UKXRI | 825,00 |
| 3 | 1 | Allen, Michael | 28 | Product OFBNT | 775,20 |
| 4 | 2 | Hassall, Mark | 72 | Product GEEEO | 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 |
| 12 | 4 | Amdt, Torsten | 57 | Product OVLQI | 1170,00 |
| 13 | 5 | Higginbotham, Tom | 38 | Product QDOMO | 3952,50 |
| 14 | 5 | Higginbotham, Tom | 43 | Product ZZZHR | 2760,00 |

Query executed s... LAPTOP-95A4MO8C (16.0 RTM) LAPTOP-95A4MO8C\Lenovo... TSQL 00:00:00 | 267 rows

[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:



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

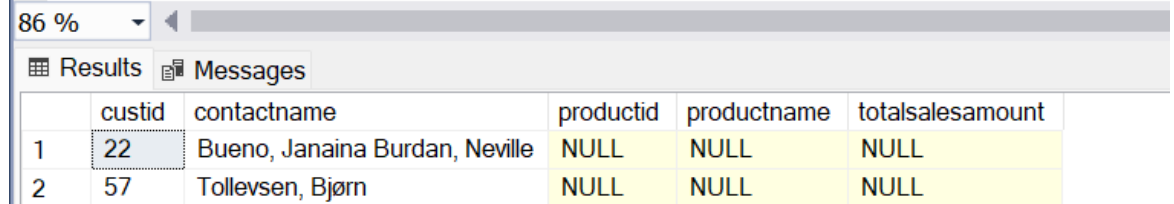
A-MBP\SQLEXPRESS (... | YUNHASNAWA-MBP\Yoppy Y... | TSQL2012 | 00:00:00 | 2 rows

4

Jawab :

```
SELECT
    c.custid, c.contactname, p.productid, productname, p.totalsalesamount
FROM
    Sales.Customers AS c
OUTER APPLY
    dbo.fnGetTop3ProductsForCustomers (c.custid) AS p
WHERE productid IS NULL AND productname IS NULL AND totalsalesamount IS NULL
ORDER BY
    c.custid;
```

86 %



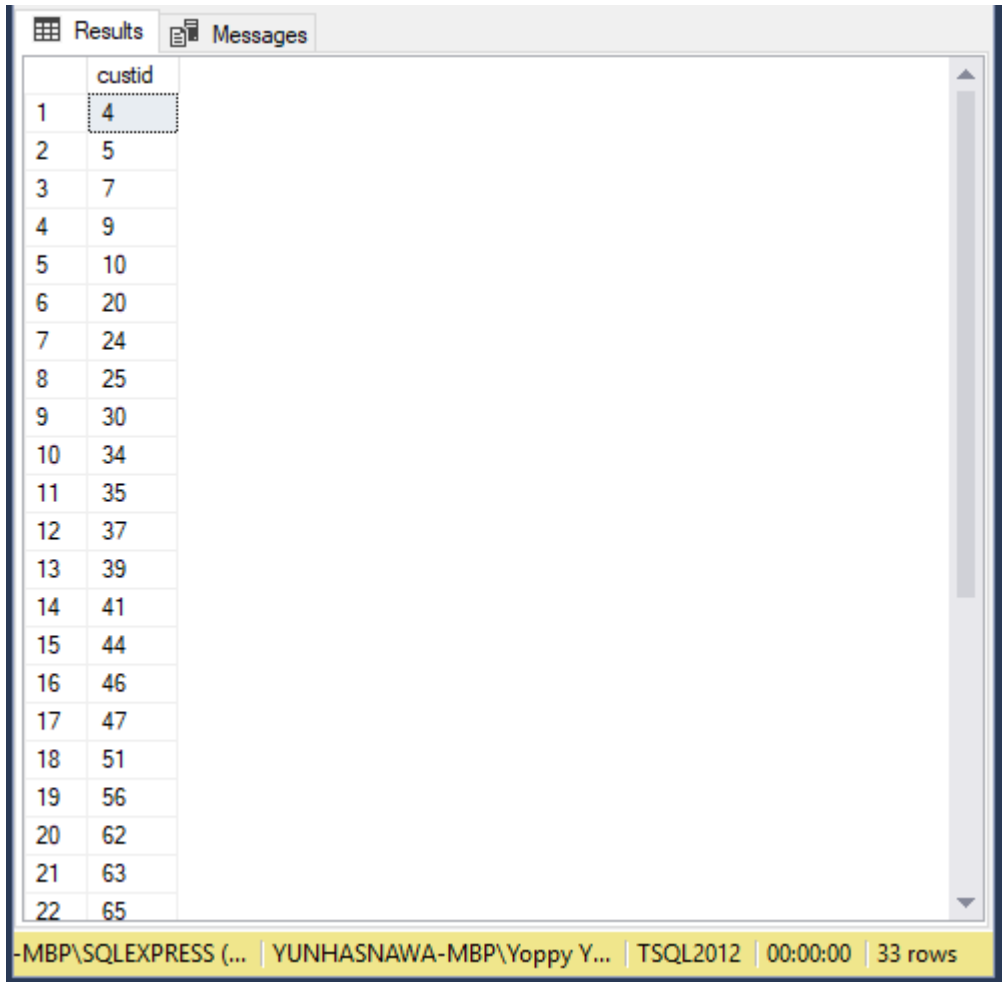
| | custid | contactname | productid | productname | totalsalesamount |
|---|--------|--------------------------------|-----------|-------------|------------------|
| 1 | 22 | Bueno, Janaina Burdan, Neville | NULL | NULL | NULL |
| 2 | 57 | Tollefsen, Bjørn | NULL | NULL | NULL |

5

Hapus fungsi yang dibuat pada praktikum bagian 2.2 dengan mengeksekusi SQL berikut:

```
IF OBJECT_ID('dbo.fnGetTop3ProductsForCustomer') IS NOT NULL
DROP FUNCTION dbo.fnGetTop3ProductsForCustomer;
```

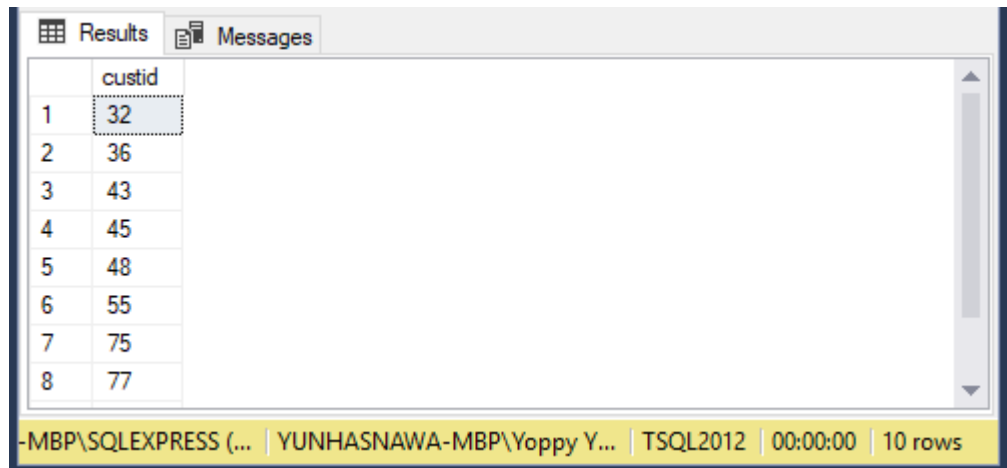
Praktikum – Bagian 3: EXCEPT & INTERSECT

| Langkah | Keterangan |
|---------|--|
| 1 | <p>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').</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 COUNT(DISTINCT d.productid) > 20; </pre> <p>Ketik dan eksekusilah SQL tersebut sehingga hasilnya serupa dengan gambar berikut:</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.

Petunjuk: Tambahkan sebuah query untuk mendapatkan customer dari USA dan tambahkan operator EXCEPT didepan query praktikum-3 langkah-1.

Pastikan hasilnya seperti pada gambar berikut:

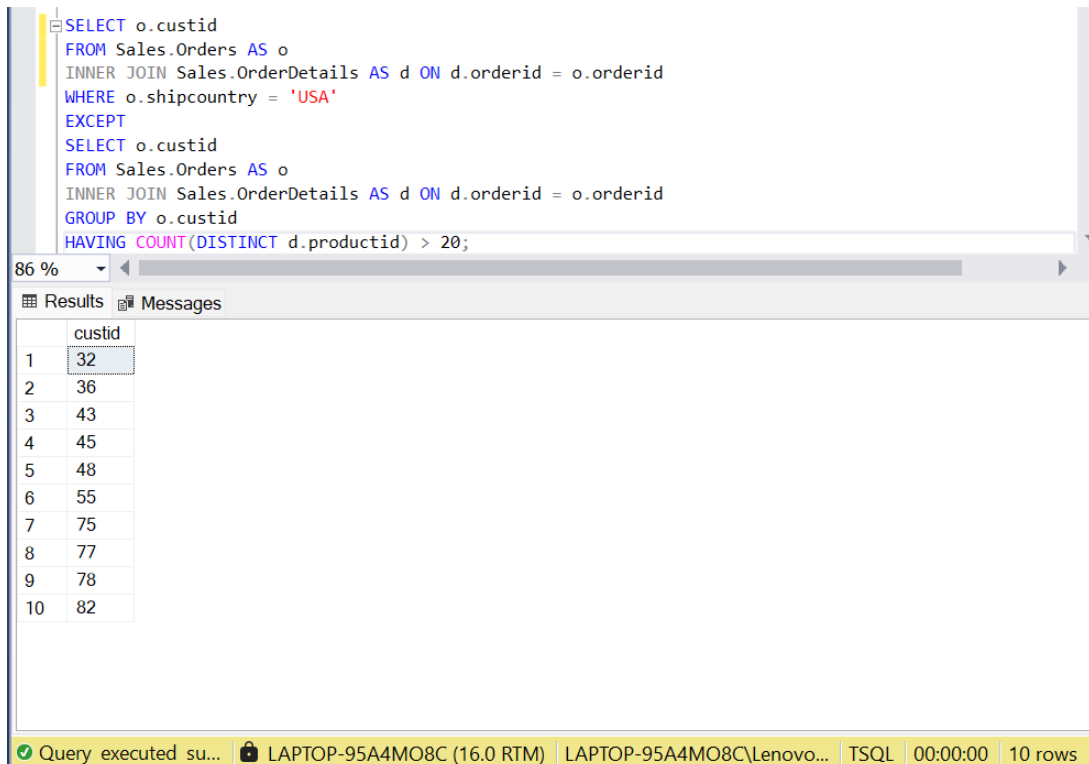


| | custid |
|---|--------|
| 1 | 32 |
| 2 | 36 |
| 3 | 43 |
| 4 | 45 |
| 5 | 48 |
| 6 | 55 |
| 7 | 75 |
| 8 | 77 |

MBP\SQLEXPRESS (... | YUNHASNAWA-MBP\Yoppy Y... | TSQL2012 | 00:00:00 | 10 rows

2

Jawab :



```

SELECT o.custid
FROM Sales.Orders AS o
INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid
WHERE o.shipcountry = 'USA'
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;

```

| | custid |
|----|--------|
| 1 | 32 |
| 2 | 36 |
| 3 | 43 |
| 4 | 45 |
| 5 | 48 |
| 6 | 55 |
| 7 | 75 |
| 8 | 77 |
| 9 | 78 |
| 10 | 82 |

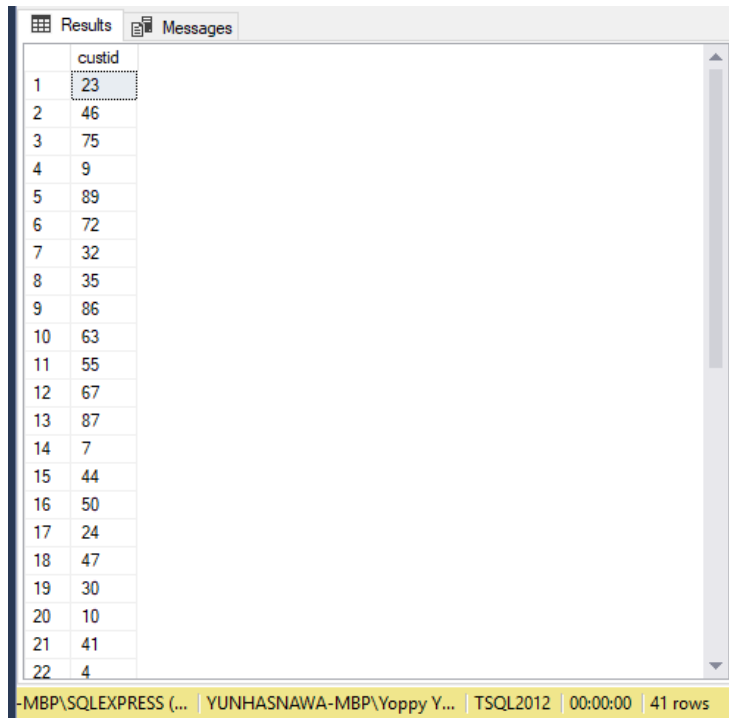
Query executed su... | LAPTOP-95A4MO8C (16.0 RTM) | LAPTOP-95A4MO8C\Lenovo... | TSQL | 00:00:00 | 10 rows

3

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'.


```
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;
```

Ketik dan eksekusi SQL diatas lalu pastikan hasilnya seperti pada gambar berikut:



| | 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 |

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:



| 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 | |
| 12 | 17 | |
| 13 | 18 | |
| 14 | 19 | |
| 15 | 21 | |
| 16 | 22 | |
| 17 | 23 | |
| 18 | 26 | |
| 19 | 27 | |
| 20 | 28 | |
| 21 | 29 | |
| 22 | 31 | |

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

Jawab :

```
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;

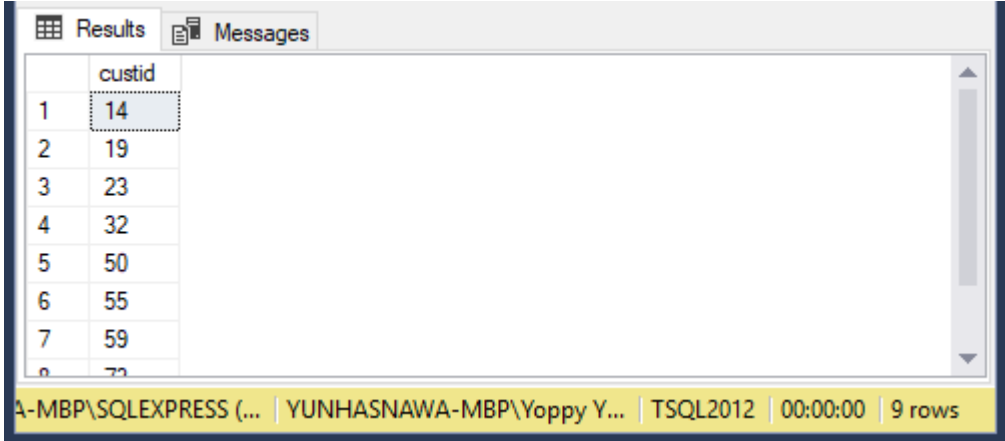
--soal no 10
```

81 %

| 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 | |
| 12 | 17 | |
| 13 | 18 | |
| 14 | 19 | |

LAPTOP-OGTLJMKB (16.0 RTM) | LAPTOP-OGTLJMKB\ASUS (57) | TSQL | 00:00:00 | 59 rows



| | <p>[Soal-9] Dapatkan Anda menyimpulkan, customer yang bagaimana yang tampil pada hasil query bagian 3.4 ini?</p> <p>Jawab :</p> <p>Pada hasil query dapat disimpulkan bahwa yang ditampilkan adalah customer yang membeli product lebih dari 20 beririsan dengan customer dengan total harga yang dibayar lebih dari 10000.</p> | | | | | | | | | | | | | | | | | | | | |
|---|---|--|--------|---|----|---|----|---|----|---|----|---|----|---|----|---|----|---|----|---|----|
| 5 | <p>[Soal-10] Salin keseluruhan query pada bagian-3.4, modifikasi SQL tersebut dengan cara mengagit statement SELECT sebelum operator INTERSECT dengan tanda kurung '(' dan ')'. Eksekusilah SQL tersebut dan pastikan hasilnya seperti pada gambar berikut:</p>  <p>The screenshot shows a SQL Server Results window with a table containing 9 rows of customer IDs. The status bar at the bottom indicates '9 rows'.</p> <table border="1"><thead><tr><th></th><th>custid</th></tr></thead><tbody><tr><td>1</td><td>14</td></tr><tr><td>2</td><td>19</td></tr><tr><td>3</td><td>23</td></tr><tr><td>4</td><td>32</td></tr><tr><td>5</td><td>50</td></tr><tr><td>6</td><td>55</td></tr><tr><td>7</td><td>59</td></tr><tr><td>8</td><td>72</td></tr><tr><td>9</td><td>73</td></tr></tbody></table> <p>Jawab :</p> | | custid | 1 | 14 | 2 | 19 | 3 | 23 | 4 | 32 | 5 | 50 | 6 | 55 | 7 | 59 | 8 | 72 | 9 | 73 |
| | custid | | | | | | | | | | | | | | | | | | | | |
| 1 | 14 | | | | | | | | | | | | | | | | | | | | |
| 2 | 19 | | | | | | | | | | | | | | | | | | | | |
| 3 | 23 | | | | | | | | | | | | | | | | | | | | |
| 4 | 32 | | | | | | | | | | | | | | | | | | | | |
| 5 | 50 | | | | | | | | | | | | | | | | | | | | |
| 6 | 55 | | | | | | | | | | | | | | | | | | | | |
| 7 | 59 | | | | | | | | | | | | | | | | | | | | |
| 8 | 72 | | | | | | | | | | | | | | | | | | | | |
| 9 | 73 | | | | | | | | | | | | | | | | | | | | |

```
(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;
```

74 %

Results Messages

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

LAPTOP-OGTLJMKB (16.0 RTM) | LAPTOP-OGTLJMKB\ASUS (57) | TSQL | 00:00:00 | 9 rows

[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?

Jawab :

Iya beda, karena tanda kurung mengelompokkan operasi EXCEPT terlebih dahulu, sehingga query tersebut bekerja dalam dua tahap:

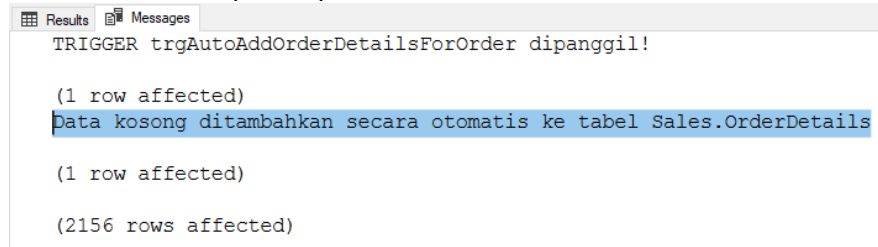
1. Operasi EXCEPT: Mengambil custid dari Sales.Customers yang tidak memiliki pesanan dengan lebih dari 20 produk unik. Ini akan menghasilkan daftar custid yang memenuhi kondisi tersebut.
2. Operasi INTERSECT: Kemudian, hasil dari operasi EXCEPT akan dicocokkan dengan custid yang memiliki total pembelian lebih dari 10.000. Hanya custid yang muncul di kedua hasil inilah yang akan tampil. Jadi, query pertama ini mencari pelanggan yang:
 - Tidak pernah memesan lebih dari 20 produk unik dan
 - Memiliki total pembelian lebih dari 10.000.

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, maka akan menampilkan pesan berikut:



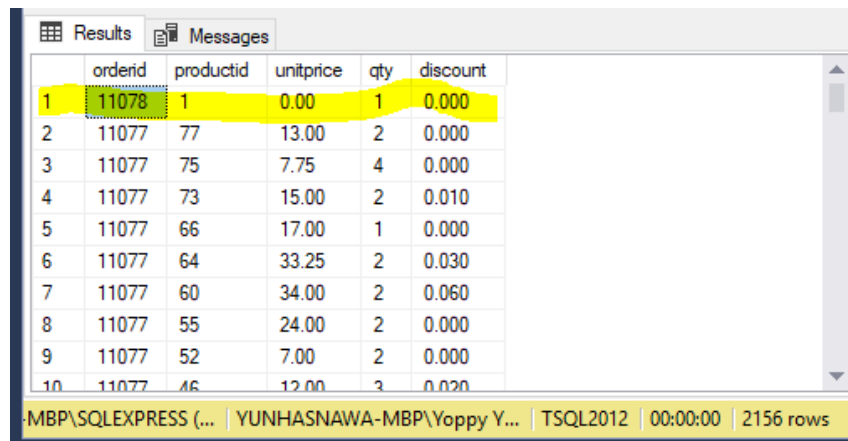
```
Results Messages
TRIGGER trgAutoAddOrderDetailsForOrder dipanggil!

(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

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

2

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);

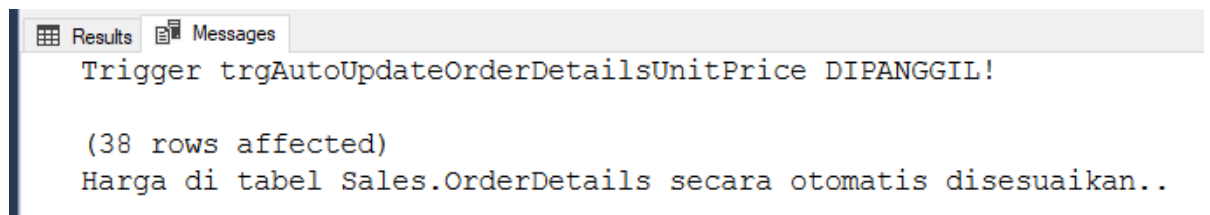
    UPDATE Sales.OrderDetails SET unitprice = @unitprice
    WHERE productid = @productid;

    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:



```
Results  Messages
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

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!

Jawab :


```

CREATE TRIGGER trgUpdateDiscontinuedAfterDelete
ON Sales.OrderDetails
AFTER DELETE
AS
BEGIN
    PRINT 'TRIGGER trgUpdateDiscontinuedAfterDelete dipanggil!';

    DECLARE @deletedProductId INT;

    -- Ambil productid yang dihapus
    SELECT @deletedProductId = d.productid
    FROM deleted d;

    -- Periksa apakah semua entri productid dihapus dari Sales.OrderDetails
    IF NOT EXISTS (
        SELECT 1
        FROM Sales.OrderDetails od
        WHERE od.productid = @deletedProductId
    )
    BEGIN
        -- Jika semua entri productid dihapus, update kolom discontinued di tabel Production.Products
        UPDATE Production.Products
        SET discontinued = 1
        WHERE productid = @deletedProductId;

        PRINT 'Kolom discontinued di tabel Production.Products diubah menjadi 1';
    END
END

```

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 | | Messages | | | | |
|---------|-----------|---------------|------------|------------|-----------|--------------|
| | 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 'SQLQuery-EmployeesBackup.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> |
| 2 | <p>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.</p> <p>Buatlah TRIGGER yang menyelesaikan permasalahan diatas dengan mengeksekusi SQL berikut:</p> <pre> IF OBJECT_ID('HR.trgDivertInsertEmployeeToBackup') IS NOT NULL DROP TRIGGER HR.trgDivertInsertEmployeeToBackup GO; CREATE TRIGGER <u>trgDivertInsertEmployeeToBackup</u> 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;</pre> |



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';
```

3

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

```
Messages
TRIGGER trgDivertUpdateEmployeeToBackup DIPANGGIL!

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

(1 row affected)
```

Jawab :



```

CREATE TRIGGER trgPreventUpdateAndBackup
ON HR.Employees
INSTEAD OF UPDATE
AS
BEGIN
    PRINT 'TRIGGER trgPreventUpdateAndBackup dipanggil!';

    DECLARE @ID VARCHAR(2) = (SELECT empid FROM inserted);
    DECLARE @firstname varchar(10) = (SELECT firstname FROM inserted);
    DECLARE @lastname varchar(10) = (SELECT lastname FROM inserted);

    UPDATE HR.EmployeesBackup SET
    firstname = @firstname,
    lastname = @lastname
    WHERE empid = @ID;

    SELECT @ID = empid FROM inserted;
    PRINT 'data karyawan empid= ' + @ID + ' yang ada di
    HR.EmployeesBackup yang diupdate';

    DISABLE TRIGGER trgDivertInsertEmployeeToBackup ON HR.Employees
GO

INSERT INTO HR.Employees
VALUES (
    'Santoso', 'Adi', 'Staff', 'Mr.', '19830101', '20170101',
    'Jl. Soekarno-Hatta', 'Malang', 'Jawa Timur', '65151', 'Indonesia',
    '(085) 123-456', 1
);

```

Messages

TRIGGER trgDivertUpdateEmployeeToBackup DIPANGGIL!

(1 row affected)

Karyawan dengan empid: 10 yang ada di HR.EmployeesBackup yang diupdate.

(1 row affected)

4

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

[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'!

Lalu jalankan SQL berikut agar TRIGGER yang Anda buat tereksekusi:

```

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

Jawab :

```
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!';

    DECLARE @firstname varchar(10) = (SELECT firstname FROM deleted);
    DECLARE @lastname varchar(10) = (SELECT lastname FROM deleted);

    DELETE FROM HR.EmployeesBackup
    WHERE firstname = @firstname;

    SELECT @firstname = firstname from deleted;

    PRINT 'Karyawan dengan nama : ' + @firstname + ' ' + @lastname +
    ' dihapus di HR.EmployeesBackup saja. Di tabel aslinya tetap.';
GO
```

67 %

Messages

Commands completed successfully.

Completion time: 2024-11-09T21:43:29.9496627+07:00

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

74 %

Messages

TRIGGER trgDivertDeleteEmployeeToBackup DIPANGGIL!

(0 rows affected)

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

(1 row affected)

Completion time: 2024-11-09T21:44:31.6835683+07:00

--- Selamat Mengerjakan ---