JOBSHEET

PRAKTIKUM BASIS DATA LANJUT

Jurusan Teknologi Informasi

POLITEKNIK NEGERI MALANG

Week 6

SQL SERVER – TABLE EXPRESSION

Information Technology Department, Malang State Polytechnic

Jobsheet 6: Table Expression

Supervisor: Advanced Database Teaching Team

September 2024

Topics

1. Table Expressions

Objective

1. Students understand how to use VIEWS
2. Students understand how to use derived tables
3. Students understand how to use common table-expression (CTE)
4. Students understand how to use inline table-valued functions (TVF)

General Instructions

1. Follow the steps in the practical sections in the order given.
2. You can use SQL Server 2012 Standard Edition to try the practicum on this jobsheet. Adjust it to your computer's condition.
3. Answer all questions marked [Question-X] that are found in certain steps in each part of the practicum.
4. In each step of the practicum, there is an explanation that will help you answer the questions in instruction number 3, so read and do all the practicum parts in this jobsheet.
5. Write the answers to the questions in the instructions number 3 in a report that is done using a word processing application (Word, OpenOffice, or other similar). Export as a **PDF file** with the following name format:
   - **BDL_Task 6 _Class_2X_AbsenteeNumberDigit_YourFullName** .pdf
   - Example:
     - **BDL_Assignment 6 _SIB2Q_99_DonaldDuck** .pdf
   - Pay close attention to the naming format.
   - Collect the PDF files as a practical report to the supervising lecturer.
   - In addition to the file name, also include your identity on the first page of the report.

Lab – Part 1: View - Write a SELECT query to get all products in a particular category

Step	Information
1	Make sure your database is connected to 'TSQL'
2	[Question- 1] Write a SELECT query to display the *productid* , *productname* , *supplierid* , *unitprice* and *discontinued columns* from the **Productions.Product table** . Then filter the results to only display products in the Beverages category (`categoryid = 1`)

3

Execute the query in step 2 above and compare it with the results shown in the following display:

Results | Messages

	productid	productname	supplierid	unitprice	discontinued
1	1	Product HHYDP	1	18,00	0
2	2	Product RECZE	1	19,00	0
3	24	Product QOGNU	10	4,50	1
4	34	Product SWNJY	16	14,00	0
5	35	Product NEVTJ	16	18,00	0
6	38	Product QDOMO	18	263,50	0
7	39	Product LSOFL	18	18,00	0
8	43	Product ZZZHR	20	46,00	0
9	67	Product XLXQF	16	14,00	0
10	70	Product TOONT	7	15,00	0
11	75	Product BWRLG	12	7,75	0
12	76	Product JYGFE	23	18,00	0

Query... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (63) | TSQL2012 | 00:00:00 | 12 rows

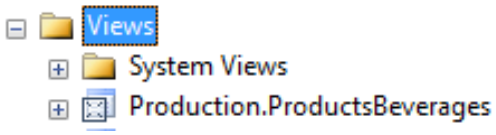
4

[Question- 2] Modify the T-SQL code from no. 2 above by adding the following T-SQL code (place it before the SELECT query)

```
CREATE VIEW Production . ProductsBeverages AS
```

5

Execute the query in step 4 above, which will produce a *VIEW object* named **ProductsBeverages** in the **Production schema.**



Practical – Part 2: View - Writing a SELECT query against the VIEW that has been created

Step	Information
1	[Question- 3] Create a SELECT query consisting of the *productid* and *productname columns* from *VIEW* **Production.ProductsBeverages** . Then filter the results to only display products with `supplierid = 1` .

2	Execute the query in step 1 above and compare it with the results shown in the following display:

Results | Messages

	productid	productname
1	1	Product HHYDP
2	2	Product RECZE

CRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (63) | TSQL2012 | 00:00:00 | 2 rows

Lab – Part 3: View - Adding an ORDER BY clause to *a VIEW*

Step	Information

1	Consider the following T-SQL script:
	
	`ALTER VIEW Production . ProductsBeverages AS`
	`SELECT`
	`productid , product name , supplierid , unit price , discontinued`
	`FROM Production . Products`
	`WHERE Category ID = 1`
	`ORDER BY product name ;`

2	[Question- 4] After executing the T-SQL above, what happens? Write down the error message and explain the cause of the error!

3	Modify the T-SQL in step 1 above by adding `TOP(100) PERCENT` so that now the query becomes:
	
	`ALTER VIEW Production . ProductsBeverages AS`
	`SELECT TOP ( 100 ) PERCENT`
	`productid , product name , supplierid , unit price , discontinued`
	`FROM Production . Products`
	`WHERE Category ID = 1`
	`ORDER BY product name ;`

4	Execute the T-SQL in step 3 above and notice that the query has successfully changed the VIEW **Production.ProductsBeverages** even though there is still an ORDER BY clause in the query.

5	[Question- 5] If a query is run against a modified VIEW **Production.ProductsBeverages , will the rows generated from the VIEW always be sorted by** *productname* ? Explain!

Lab – Part 4: View - Adding columns to *a VIEW*

Step	Information

1	Consider the following T-SQL statement that adds an additional column to the VIEW **Production.ProductsBeverages** that was created in the Practical - Part 1 with the ALTER VIEW command.
	
	`ALTER VIEW Production . ProductsBeverages AS`
	`SELECT`
	`productid , product name , supplierid , unit price , discontinued ,`
	`CASE WHEN unit price > 100. THEN N'high' ELSE N'normal' END`
	`FROM Production . Products`
	`WHERE Category ID = 1 ;`

2	[Question- 6] After executing the T-SQL above, what happens? Write down the error message and explain the cause of the error!
3	[Question- 7] Fix the T-SQL script above so that it runs correctly.

Lab – Part 5: View - Deleting a VIEW

Step	Information

1	To delete the VIEW **Production.ProductsBeverages** , execute the following T-SQL command:
	
	`IF OBJECT_ID ( N'Production.ProductsBeverages' , N'V' ) IS NOT NULL`
	`DROP VIEW Production . ProductsBeverages ;`

Practical – Part 6: Derived Table - Creating a SELECT query in a derived table

Step	Information
1	[Question-8] Using the TSQL database, create a SELECT query against *the derived table* containing the *productid* and *productname columns* , with a filter to only display data whose *'pricetype'* is 'high'. Use the SELECT query in the Practical - Part 4 - Step 1 as *the derived table* . Give the alias name *p* to the *derived table* .

2	Execute the query in step 1 above and compare it with the results shown in the following display:

Results | Messages

	productid	productname
1	38	Product QDOMO

CRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (65) | TSQL2012 | 00:00:00 | 1 rows

Practical – Part 7: Derived Table - Create a SELECT query to find out the total and average number of orders (nominal)

Step	Information
1	[Question- 9] Create a SELECT query to get the *custid column* and 2 (two) calculation columns, namely *totalsalesamount* (total nominal amount of orders per customer) and *avgsalesamount* (average nominal amount of orders per customer). To find out the average nominal order per customer, you must first find the total nominal amount per order. The way to do this is by creating a *derived table* that contains a JOIN query between the **Sales.Orders** and **Sales.OrderDetails tables. After that, you can use the** *custid* and *orderid* columns from the **Sales.Orders table** , as well as the *qty* and *unitprice columns* from the **Sales.OrderDetails table** .
2	Execute the query in step 1 above and compare it with the results shown in the following display:

Results | Messages

	custid	totalsalesamount	avgsalesamount
1	1	4596,20	766,0333
2	2	1402,95	350,7375
3	3	7515,35	1073,6214
4	4	13806,50	1062,0384
5	5	26968,15	1498,2305
6	6	3239,80	462,8285
7	7	19088,00	1735,2727

RURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (65) | TSQL2012 | 00:00:00 | 89 rows

Practical – Part 8: Derived Table - Create a SELECT query to get the sales growth percentage

Step	Information
1	[Question- 10] Write a SELECT query that contains the following columns: - *orderyear* : year from order date - *curtotalsales* : total amount of sales in the year - *prevtotalsales* : total sales amount in the previous year - *percentgrowth* : percentage of sales growth from the current year compared to the previous year
2	You need to create a T-SQL query using 2 (two) *derived tables* . To get the year and total sales for each SELECT query, you can use the existing VIEW named **Sales.OrderValues** . In that view, the *val column* represents the sales amount.
3	It should be noted that in the TSQL database, 2006 is the earliest order year (there are no previous years), but the query can still be executed.
4	Execute the query in step 1 above and compare it with the results shown in the following display:

	Results | Messages orderyear curtotalsales prevtotalsales percentgrowth 1 2006 208083.99 NULL NULL 2 2007 617085.30 208083.99 196.555800 3 2008 440623.93 617085.30 -28.595900 CRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (65) | TSQL2012 | 00:00:00 | 3 rows

Practical – Part 9 : CTE - Creating a SELECT query using CTE

Step	Information
1	[Question-11] While still using the TSQL database, create a SELECT query like in the Practical - Part 6 , but using Common Table Expressions (CTE). Name the CTE query alias as **ProductBeverages** .
2	Execute the query in step 1 and compare it with the results shown in the following display: Results | Messages productid productname 1 38 Product QDOMO Query e... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (72) | TSQL2012 | 00:00:00 | 1 rows

Practical – Part 10 : CTE - Create a SELECT query to get the total sales amount (nominal) for each customer.

Step	Information
1	[Question-12] Create a SELECT query against the **Sales.OrderValues view** to get the customer ID and total sales amount in 2008. Name this CTE as **c2008** , which consists of the *custid* and *salesamt2008 columns.*

	Then, perform a JOIN operation between the **Sales.Customers table** and the CTE c2008, resulting in the *custid* and *contactname columns* from the **Sales.Customer** table and *the salesamt2008 column* from the CTE **c2008** .
2	Execute the query in step 1 above and compare it with the results shown in the following display:

Results | Messages

	custid	contactname	salesamt2008
1	1	Allen, Michael	2250.50
2	2	Hassall, Mark	514.40
3	3	Peoples, John	660.00
4	4	Amdt, Torsten	5604.75
5	5	Higginbotham, Tom	6754.16
6	6	Poland, Carole	2160.00
7	7	Bansal, Dushyant	730.00
8	8	Ilyina, Julia	224.00

Query... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (72) | TSQL2012 | 00:00:00 | 91 rows

<u>Practical – Part 11 : CTE - Create a SELECT query to compare the total sales amount for each customer with the previous year.</u>

Step	Information
1	[Question- 13] Create a SELECT query containing the *custid* and *contactname columns* against the **Sales.Customers table** . Also, get the values for the following columns: - *salesamt2008* : total sales amount in 2008 - *salesamt2007* : total sales amount in 2007 - *percentgrowth* : percentage growth in sales between 2007 and 2008 If *percentgrowth* returns NULL, display it as 0. You can use the CTE from <u>Lab Part 10</u> and create another CTE for the year 2007. Then, perform a JOIN operation between the two CTEs with the **Sales.Customers table** . Sort the results by the *percentgrowth column.*
2	Execute the query in step 1 above and compare it with the results shown in the following display:

Results | Messages

	custid	contactname	salesamt2008	salesamt2007	percentgrowth
1	74	O'Brien, Dave	2371.00	52.35	4429.130800
2	54	Tiano, Mike	3031.00	429.20	606.197500
3	17	Jones, TiAnna	2809.61	420.00	568.954700
4	12	Ray, Mike	1576.80	238.00	562.521000
5	70	Ginters, Kaspars	3976.75	700.00	468.107100
6	27	Schmöllerl, Martin	1296.00	249.70	419.022800
7	34	Cohen, Shy	23821.20	6022.77	295.519000
8	81	Nagel, Jean-Philippe	4234.26	1320.40	220.680000

Query... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (72) | TSQL2012 | 00:00:00 | 91 rows

Practical – Part 12: Inline TVF - Create a SELECT query to get the total sales amount (nominal) for each customer.

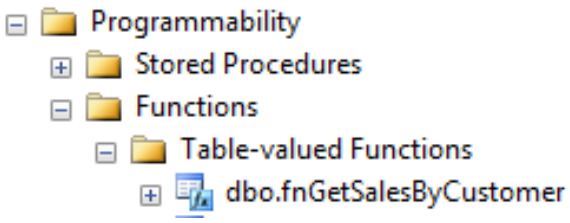
Step	Information
1	[Question- 14] Using a TSQL database, create a SELECT query against the **Sales.OrderValues view** that contains the *custid column* and the *totalsalesamount column* (the total of the *val column*). Filter the results to only display orders in 2007.
2	Execute the query in step 1 above and compare it with the results shown in the following display:
3	[Question- 15] Create an inline TVF/Table-Valued Function by adding the following line and placing it before the SELECT query in Step 1 above.

Results | Messages

	custid	totalsalesamount
1	1	2022.50
2	2	799.75
3	3	5960.78
4	4	6406.90
5	5	13849.02
6	6	1079.80
7	7	7817.88
8	8	3026.85
9	9	11208.36
10	10	7620.25

Query... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (51) | TSQL2012 | 00:00:00 | 86 rows

```
CREATE FUNCTION dbo . fnGetSalesByCustomer
( @orderyear US INT ) RETURNS TABLE
AS
```

	RETURN
4	[Question- 16] Modify the query by replacing the constant value of 2007 in the WHERE clause, with the **@orderyear parameter** .
5	Run the script in step 4 above so that an inline TVF named **dbo.fnGetSalesByCustomer will be created.** 

Practical – Part 12 : Inline ITF - Creating a SELECT query that operates on an inline table-valued function

Step	Information
1	[Question- 17] Create a SELECT query containing the *custid* and *totalsalesamount columns* against the inline TVF **dbo.fnGetSalesByCustomer** . Enter the value 2007 as the parameter.
2	Execute the query in step 1 above and compare it with the results shown in the following display:

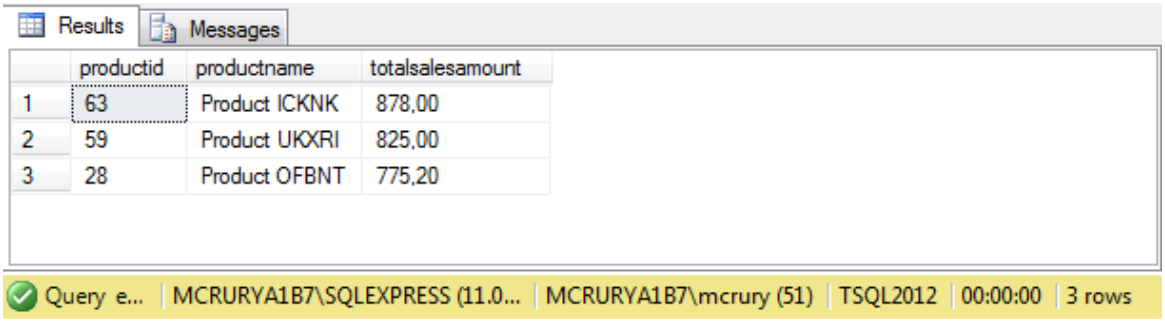
Results | Messages

	custid	totalsalesamount
1	1	2022.50
2	2	799.75
3	3	5960.78
4	4	6406.90
5	5	13849.02
6	6	1079.80
7	7	7817.88
8	8	3026.85
9	9	11208.36
10	10	7620.25

Query... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (51) | TSQL2012 | 00:00:00 | 86 rows

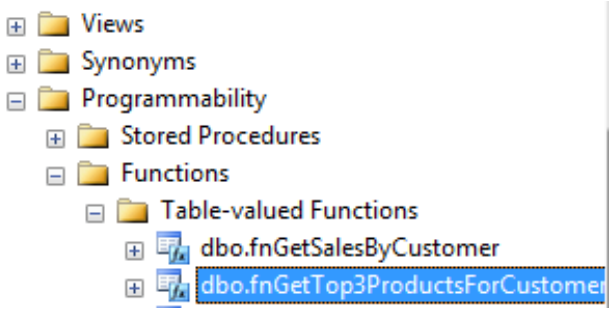
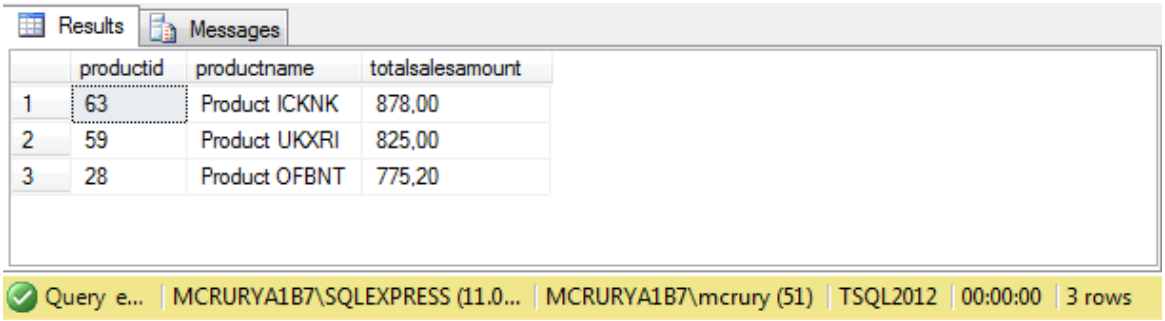
Practical – Part 13 : Inline ITF - Creating a SELECT query to get the 3 best-selling products for a particular customer

Step	Information
1	[Question-1 8] Create a SELECT query that displays the top 3 best-selling products for a customer with ID = 1. Get the *productid* and *productname columns* from the **Production.Products table** . Use the *qty* and *unitprice columns from the Sales.OrderDetails* table to calculate the nominal value for each order row, which is then added up for each product to produce the *totalsalesamount column* . Filter the results to only display data with a custid value = 1.

2	Execute the query in step 1 above and compare it with the results shown in the following display:  Results | Messages productid productname totalsalesamount 1 63 Product ICKNK 878,00 2 59 Product UKXRI 825,00 3 28 Product OFBNT 775,20 Query e... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (51) | TSQL2012 | 00:00:00 | 3 rows

3	[Question-1 9] Using the SELECT query in step 1 above, create an inline TVF by adding a few lines of function before the SELECT query and set the value of *the custid constant* in the query with the **@custid parameter** , as follows:
	
	`CREATE FUNCTION dbo . fnGetTop3ProductsForCustomer`
	`( @custid US INT ) RETURNS TABLE`
	`AS`
	`RETURN`

4	Run the script so that an inline TVF named **dbo.fnGetTop3ProductsForCustomer will be created** which has a customer ID parameter.

	
5	[Question-20] Test it by creating a SELECT query on the inline TVF and insert the value 1 as the customer ID parameter. Display the *productid* , *productname* , *totalsalesamount columns* , and give the alias name *p* for the inline TVF.
6	Execute the query in step 1 above and compare it with the results shown in the following display: 

Results | Messages

	productid	productname	totalsalesamount
1	63	Product ICKNK	878,00
2	59	Product UKXRI	825,00
3	28	Product OFBNT	775,20

Query e... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (51) | TSQL2012 | 00:00:00 | 3 rows

Lab – Part 14 : Inline TVF - Deleting inline Table-valued function

Step	Information

1	Delete the inline TVF that has been created by running the following script:
	
	`IF OBJECT_ID ( 'dbo.fnGetSalesByCustomer' ) IS NOT NULL DROP FUNCTION dbo . fnGetSalesByCustomer ;`
	`IF OBJECT_ID ( 'dbo.fnGetTop3ProductsForCustomer' ) IS NOT NULL DROP FUNCTION dbo . fnGetTop3ProductsForCustomer ;`

--- Have a great time doing it ----