Database Advance Job Sheet-6: Table Expressions

From:

AL AZHAR RIZQI RIFA'I FIRDAUS

Class:

2 I

Absence:

01

Student Number Identity:

2241720263

Department:

Information Technology

Study Program:

Informatics Engineering

Practicum 1

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 that are in the category of Beverages only (categoryid = 1).

num1.sql

minggu6 > query > num1.sql

```sql
SELECT
    productid, productname, supplierid, unitprice
FROM Production.Products
WHERE categoryid = 1;
```

num1.sql

RESULTS

	productid	productname	supplierid	unitprice
1	1	Product HHYDP	1	18.00
2	2	Product RECZE	1	19.00
3	24	Product QOG...	10	4.50
4	34	Product SWNJY	16	14.00
5	35	Product NEVTJ	16	18.00
6	38	Product QDO...	18	263.50
7	39	Product LSOFL	18	18.00
8	43	Product ZZZHR	20	46.00
9	67	Product XLXQF	16	14.00
10	70	Product TOONT	7	15.00
11	75	Product BWRLG	12	7.75
12	76	Product JYGFE	23	18.00

2. Modify the T-SQL code from number 2 above by adding the following T-SQL code (place it before the SELECT query)

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

num1.sql

minggu6 > query > num1.sql

```sql
CREATE VIEW Production.ProductsBeverages AS
SELECT
    productid, productname, supplierid, unitprice
FROM Production.Products
WHERE categoryid = 1;
```

num1.sql

MESSAGES

Timestamp	Message
[2:57:40 PM]	Started executing query at Line 1
	Commands completed successfully.
	Total execution time: 00:00:00.005

Views

System Views

Production.ProductsBeverages

Practicum 2

3. Create a SELECT query consisting of productid and productname columns from the VIEW Production.ProductsBeverages. Then filter the results to only display products with supplierid = 1.

num1.sql | num3.sql | num2.sql

minggu6 > query > num3.sql

```sql
SELECT
    productid, productname
FROM Production.ProductsBeverages
WHERE supplierid = 1;
```

num3.sql

RESULTS

	productid	productname
1	1	Product HHYDP
2	2	Product RECZE

Practicum 3

4. After executing the T-SQL above, what happens? Write down the error message and explain why the error occurred!

num1.sql | num3.sql | num4.sql | num2.sql

minggu6 > query > num4.sql

```sql
ALTER VIEW Production.ProductsBeverages AS
SELECT
    productid, productname, supplierid, unitprice, discountinued
FROM Production.Products
WHERE categoryid = 1
ORDER BY productname;
```

num4.sql

MESSAGES CTRL+ALT+Y

Timestamp	Message
[3:08:20 PM]	Started executing query at Line 1
	Msg 1033, Level 15, State 1, Procedure ProductsBeverages, Line 6
	The ORDER BY clause is invalid in views, inline functions, derived tables, subqueries, and common table expressions, unless TOP, OFFSET or FOR XML is also specified.
	Total execution time: 00:00:00.005

- According query above, it can error because ORDER BY can't be definition in view without top.

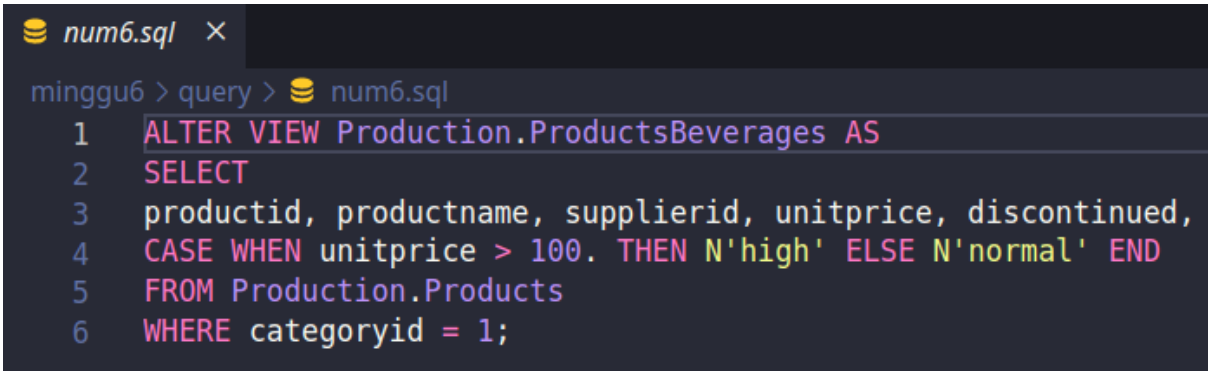
5. If a query is run against the Production.ProductsBeverages VIEW that has been modified VIEW, will the rows produced from the VIEW always be sorted by productname? Explain!

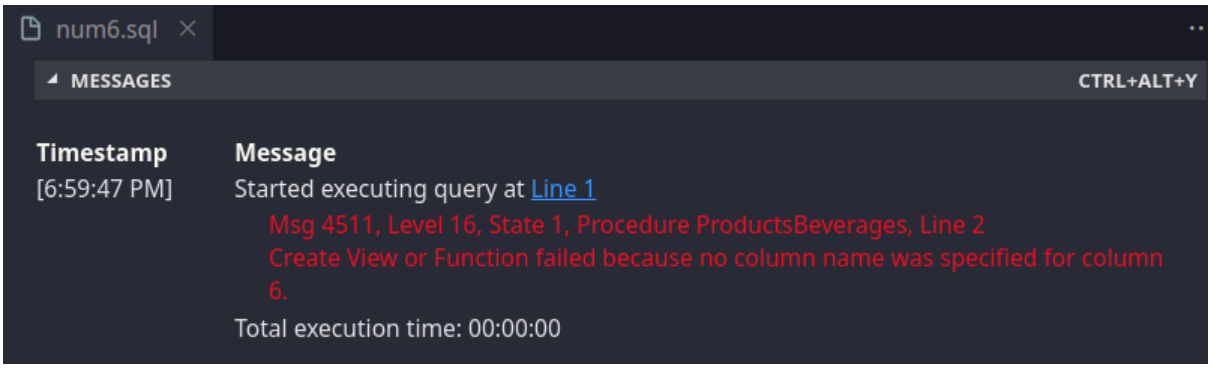
- Yes, because rows from Production.ProductsBeverages views be sorted based on productname that used ORDER BY.

Practicum 4

```
ALTER VIEW Production.ProductsBeverages AS
SELECT
      productid, productname, supplierid, unitprice, discontinued,
      CASE WHEN unitprice > 100. THEN N'high' ELSE N'normal' END
FROM Production.Products
WHERE categoryid = 1;
```

6. After executing the T-SQL above, what happens? Write down the error message and explain why the error occurred!





- According query above, it happened because there is no column that used to display the result from the CASE.

7. Correct the above T-SQL script so that it runs correctly.

num6.sql

minggu6 > query > num6.sql

```sql
ALTER VIEW Production.ProductsBeverages AS
SELECT
productid, productname, supplierid, unitprice, discontinued,
CASE WHEN unitprice > 100. THEN N'high' ELSE N'normal' END AS price
FROM Production.Products
WHERE categoryid = 1;
```

num6.sql

RESULTS

	productid	productname	supplierid	unitprice	discontinued	price
1	1	Product HHYDP	1	18.00	0	normal
2	2	Product RECZE	1	19.00	0	normal
3	24	Product QOG...	10	4.50	1	normal
4	34	Product SWNJY	16	14.00	0	normal
5	35	Product NEVTJ	16	18.00	0	normal
6	38	Product QDO...	18	263.50	0	high
7	39	Product LSOFL	18	18.00	0	normal
8	43	Product ZZZHR	20	46.00	0	normal
9	67	Product XLXQF	16	14.00	0	normal
10	70	Product TOONT	7	15.00	0	normal
11	75	Product BWRLG	12	7.75	0	normal
12	76	Product JYGFE	23	18.00	0	normal

Practicum 5

Practicum 6

8. Using the TSQL2012 database, create a SELECT query against a derived table that contains the productid and productname columns, with a filter that only display only data whose 'pricetype' is 'high'. Use the SELECT query from Practicum - Part 4 - Step 1 as the derived table. Give an alias name p to the derived table.

num8.sql

minggu6 > query > num8.sql

```sql
SELECT p.productid, p.productname
FROM(
    SELECT
        productid, productname, supplierid, unitprice, discontinued,
        CASE WHEN unitprice > 100. THEN N'high' ELSE N'normal'
        END AS price
    FROM Production.Products
    WHERE categoryid = 1
) AS p
WHERE p.price = N'high';
```

num8.sql

RESULTS

	productid	productname
1	38	Product QDO...

Practicum 7

9. Create a SELECT query to get the custid column and 2 (two) calculation columns calculation columns, namely totalsalesamount (total nominal amount of orders per customer) and avgsalesamount (average order amount per customer). To find out the average nominal amount of orders per customer, we must first find the total amount of nominal amount per order. The trick is to create a derived table that contains a query JOIN between the Sales.Orders and Sales.OrderDetails tables. After that, you can use the custid and orderid columns from the Sales.Orders table, and the qty and unitprice columns from the Sales.OrderDetails table.

num9.sql

minggu6 > query > num9.sql

```sql
SELECT
    custid, SUM(totalsalesamount) AS total_sales_amount, AVG
    (totalsalesamount) AS avg_sales_amount
FROM (
    SELECT
    o.custid, o.orderid, SUM(qty * unitprice) AS
    totalsalesamount
    FROM Sales.Orders AS o
    INNER JOIN Sales.OrderDetails od ON o.orderid = od.orderid
    GROUP BY custid, o.orderid
) AS p
GROUP BY custid;
```

num9.sql

RESULTS

	custid	total_sales_a...	avg_sales_am...
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
8	8	5297.80	1765.9333
9	9	23850.95	1402.997
10	10	22607.70	1614.8357

Practicum 8

10. Write a SELECT query that contains the following columns:

- orderyear: the year of the order date

- curtotalsales: total sales for the year

- prevtotalsales: total sales in the previous year

- percentgrowth: the percentage of sales growth of the current year compared to the previous year.

num10.sql

minggu6 > query > num10.sql

```sql
SELECT
    a.orderyear, curtotalsales, prevtotalsales, (curtotalsales - prevtotalsales) / prevtotalsales * 100 AS percentgrowth
FROM (
    Select YEAR(orderdate) AS orderyear, SUM(val) AS curtotalsales
    FROM Sales.OrderValues
    GROUP BY YEAR(orderdate)
) AS a
LEFT JOIN (
    Select YEAR(orderdate) AS orderyear, SUM(val) AS prevtotalsales
    FROM Sales.OrderValues
    GROUP BY YEAR(orderdate)
) AS b
ON a.orderyear = b.orderyear+1
ORDER BY orderyear ASC;
```

num10.sql

RESULTS

	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

Practicum 9

11. Continuing to use the TSQL2012 database, create a SELECT query as in Practicum - Part 6, but using Common Table Expressions (CTE). Give the CTE query alias as ProductBeverages.

num10.sql num11.sql

minggu6 > query > num11.sql

```sql
WITH ProductBeverages AS (
    SELECT
        productid, productname, supplierid, unitprice,
        discontinued,
        CASE WHEN unitprice > 100 THEN N'high' ELSE N'normal'
        END AS price
    FROM Production.Products
    WHERE categoryid = 1
)
SELECT
    productid, productname
FROM ProductBeverages
WHERE price = N'high';
```

num11.sql

RESULTS

	productid	productname
1	38	Product QDO...

Practicum 10

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 custid and salesamt2008 columns. Then, perform a JOIN operation between the Sales.Customers table and CTE c2008, resulting in the following resulting in the custid and contactname columns from the Sales.Customer table and the salesamt2008 column from CTE c2008.

num12.sql

minggu6 > query > num12.sql

```sql
WITH c2008 AS (
    SELECT
        custid, SUM(val) AS salesamt2008
    FROM Sales.OrderValues
    WHERE year(orderdate) = 2008
    GROUP BY custid
)
SELECT c2.custid, c1.contactname, c2.salesamt2008
FROM Sales.Customers AS c1
JOIN c2008 AS c2 ON c1.custid = c2.custid;
```

num12.sql

RESULTS

	custid	contactname	salesamt2008
1	1	Allen, Michael	2250.50
2	2	Hassall, Mark	514.40
3	3	Peoples, John	660.00
4	4	Arndt, Torsten	5604.75
5	5	Higginbotham...	6754.16
6	6	Poland, Carole	2160.00
7	7	Bansal, Dushy...	730.00
8	8	Ilyina, Julia	224.00
9	9	Raghav, Amrit...	6680.61
10	10	Bassols, Pilar ...	11338.56

Practicum 11

13. Create a SELECT query that contains the custid and contactname columns against the table Sales.Customers. Also, get the values for the following columns:

- salesamt2008: total sales in 2008

- salesamt2007: total sales in 2007

- percentgrowth: the percentage of sales growth between 2007 and 2008

If percentgrowth returns NULL, display it as 0. You can use the CTE from Practicum Section 10 and create another one for 2007. Then, perform a JOIN operation between the two CTEs and the Sales.Customers table. Sort the results based on the percentgrowth column.

num13.sql

minggu6 > query > num13.sql

```sql
WITH c2008 AS (
    SELECT
        custid, SUM(val) AS salesamt2008
    FROM Sales.OrderValues
    WHERE year(orderdate) = 2008
    GROUP BY custid
), c2007 AS (
    SELECT
        custid, SUM(val) AS salesamt2007
    FROM Sales.OrderValues
    WHERE year(orderdate) = 2007
    GROUP BY custid
)
SELECT
    c7.custid, c.contactname, c8.salesamt2008, c7.salesamt2007,
    CASE
        WHEN c7.salesamt2007 IS NULL THEN 0
        WHEN c8.salesamt2008 IS NULL THEN 0
        ELSE (c8.salesamt2008 - c7.salesamt2007) / c7.
        salesamt2007 * 100
    END AS percentgrowth
FROM Sales.Customers as c
LEFT JOIN c2008 AS c8
ON c8.custid = c.custid
LEFT JOIN c2007 AS c7
ON c7.custid = c.custid
ORDER BY percentgrowth DESC;
```

num13.sql

RESULTS

	custid	contactname	salesamt2008	salesamt2007	percentgrowt
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, Kaspa...	3976.75	700.00	468.107100
6	27	Schm[?]llerl, M...	1296.00	249.70	419.022800
7	34	Cohen, Shy	23821.20	6022.77	295.519000
8	81	Nagel, Jean-P...	4234.26	1320.40	220.680000
9	26	Koch, Paul	2252.06	920.10	144.762500
10	19	Boseman, Ran...	9296.69	4514.35	105.936400

Practicum 12

14. Dengan menggunakan database TSQL2012, buatlah query SELECT terhadap view Sales.OrderValues yang berisi kolom custid dan kolom totalsalesamount (total dari kolom val). Filter hasilnya untuk menampilkan hanya pesanan pada tahun 2017.

num13.sql num14.sql

minggu6 > query > num14.sql

```sql
SELECT
    custid, SUM(val) AS totalsalesamount
FROM Sales.OrderValues
WHERE YEAR(orderdate) = 2007
GROUP BY custid;
```

num14.sql

RESULTS

	custid	totalsalesamo...
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	7630.25

15. Create an inline TVF / Table-Valued Function by adding the following line and put it before the SELECT query in Step 1 above

num15.sql × num14.sql

minggu6 > query > num15.sql

```sql
CREATE FUNCTION dbo.fnGetSalesByCustomer
(@orderyear AS INT) RETURNS TABLE
AS
RETURN
SELECT
    custid, SUM(val) AS totalsalesamount
FROM Sales.OrderValues
WHERE YEAR(orderdate) = 2007
GROUP BY custid;
```

num15.sql ×

MESSAGES

Timestamp	Message
[8:55:52 PM]	Started executing query at Line 1
	Commands completed successfully.
	Total execution time: 00:00:00.010

16. Modify the query by replacing the constant value of 2007 in the WHERE clause, with the @orderyear parameter.

num14.sql num15.sql

minggu6 > query > num15.sql

```sql
CREATE FUNCTION dbo.fnGetSalesByCustomer
(@orderyear AS INT) RETURNS TABLE
AS
RETURN
SELECT
    custid, SUM(val) AS totalsalesamount
FROM Sales.OrderValues
WHERE YEAR(orderdate) = @orderyear
GROUP BY custid;
```

num15.sql

MESSAGES

Timestamp	Message
[9:01:33 PM]	Started executing query at Line 1
	Commands completed successfully.
	Total execution time: 00:00:00.003

- Programmability
  - Stored Procedures
  - Functions
    - System Functions
    - Table-valued Functions
      - dbo.fnGetSalesByCustomer

Practicum 12.1

17. Create a SELECT query that contains custid and totalsalesamount columns against the dbo.fnGetSalesByCustomer inline TVF. Enter the value 2007 as the parameter.

num14.sql num15.sql num17.sql

minggu6 > query > num17.sql

```sql
SELECT
    custid, totalsalesamount
FROM dbo.fnGetSalesByCustomer(2007);
```

num17.sql

RESULTS

	custid	totalsalesamo...
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	7630.25

Practicum 13

18. Create a SELECT query that displays the 3 best-selling products for a customer with ID = Get the productid and productname columns from the Production.Products table. Use qty and unitprice columns from the Sales.OrderDetails table to calculate the amount for each row of the order, which is then summed up for each product to produce the totalsalesamount column. produce the totalsalesamount column. Filter the result to only display data with the value custid = 1.

num18.sql

minggu6 > query > num18.sql

```sql
SELECT TOP (3)
    p.productid, MAX(productname) AS productname, SUM(od.qty *
    od.unitprice) AS totalsalesamount
FROM Sales.OrderDetails AS od
INNER JOIN Sales.Orders AS o ON od.orderid = o.orderid
INNER JOIN Production.Products AS p ON od.productid = p.
productid
WHERE custid = 1
GROUP BY p.productid
ORDER BY totalsalesamount DESC;
```

num18.sql

RESULTS

	productid	productname	totalsalesamo...
1	63	Product ICKNK	878.00
2	59	Product UKXRI	825.00
3	28	Product OFBNT	775.20

19. Using the SELECT query in step 1 above, create an inline TVF by adding a few function lines before the SELECT query and set the value of the constant in the query with the @custid parameter, as follows:

```sql
CREATE FUNCTION dbo.fnGetTop3ProductsForCustomer
(@custid AS INT) RETURNS TABLE
AS
RETURN
```

num19.sql × num18.sql

minggu6 > query > num19.sql

```sql
CREATE FUNCTION dbo.fnGetTop3ProductsForCustomer
(@custid AS INT) RETURNS TABLE
AS
RETURN
SELECT TOP (3)
    p.productid, MAX(productname) AS productname, SUM(od.qty *
    od.unitprice) AS totalsalesamount
FROM Sales.OrderDetails AS od
INNER JOIN Sales.Orders AS o ON od.orderid = o.orderid
INNER JOIN Production.Products AS p ON od.productid = p.
productid
WHERE custid = 1
GROUP BY p.productid
ORDER BY totalsalesamount DESC;
```

num19.sql ×

MESSAGES

Timestamp	Message
[9:10:28 PM]	Started executing query at Line 1
	Commands completed successfully.
	Total execution time: 00:00:00.006

- Programmability
  - Stored Procedures
  - Functions
    - System Functions
    - Table-valued Functions
      - dbo.fnGetSalesByCustomer
      - dbo.fnGetTop3ProductsForCustomer

20. Test it by creating a SELECT query on the inline TVF and enter the value 1 as the customer ID parameter. Display the productid, productname, totalsalesamount, and give the alias p to the inline TVF.

num20.sql

minggu6 > query > num20.sql

```sql
SELECT
    p.productid, p.productname, p.totalsalesamount
FROM dbo.fnGetTop3ProductsFOrCustomer(1) AS p;
```

num20.sql

RESULTS

	productid	productname	totalsalesamo...
1	63	Product ICKNK	878.00
2	59	Product UKXRI	825.00
3	28	Product OFBNT	775.20