Advanced Database
Table Expressions

Name
Dicha Zelianivan Arkana

NIM
2241720002

Class
2i

Department
Information Technology

Study Program
D4 Informatics Engineering

1 Practicum

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

2.
   ```
   CREATE VIEW
       Production.ProductsBeverages
   AS SELECT
       productid,
       productname,
       supplierid,
       unitprice,
       discontinued
   FROM
       Production.Products
   WHERE
       categoryid = 1;
   ```

3.
   ```
   SELECT
       productid,
       productname
   FROM
       Production.ProductsBeverages
   WHERE
       supplierid = 1;
   ```

4. Muncul pesan error seperti dibawah ini

   ```
   The ORDER BY clause is invalid in views, inline functions, derived tables, subqueries, and comm
   unless TOP, OFFSET or FOR XML is also specified.
   ```

   Hal ini dikarenakan kita tidak dapaet menggunakan klausa `ORDER BY` tanpa menggunakan klausa `TOP`, `OFFSET`, atau `FOR XML`.

   ```
   ALTER VIEW
       Production.ProductsBeverages
   AS SELECT TOP(100) PERCENT
       productid,
       productname,
       supplierid,
       unitprice,
       discontinued
   FROM Production.Products
   WHERE categoryid = 1
   ORDER BY productname;
   ```

5. Tidak, data tidak akan urut apabila kita tidak menggunakan klausa `ORDER BY`

6. Muncul pesan error

   ```
   Create View or Function failed because no column name was specified for column 6.
   ```

   Hal ini dikarenakan pada penggunaan klausa `CASE` kita tidak memberikan *alias* pada hasilnya.

7.
   ```
   ALTER VIEW Production.ProductsBeverages AS
   SELECT
       productid,
       productname,
       supplierid,
       unitprice,
       discontinued,
       CASE
           WHEN unitprice > 100. THEN N'high'
           ELSE N'normal'
       END AS pricetype
   FROM Production.Products
   WHERE categoryid = 1;
   ```

8.
```
SELECT
    p.productid, p.productname
FROM
    (
        SELECT
            productid, productname, supplierid, unitprice, discontinued,
            CASE
                WHEN unitprice > 100. THEN N'high'
                ELSE N'normal'
            END AS pricetype
        FROM Production.Products
        WHERE categoryid = 1
    ) AS p
WHERE p.pricetype = N'high';
```

9.
```
SELECT
    c.custid,
    SUM(c.totalsalesamountperorder) AS totalsalesamount,
    AVG(c.totalsalesamountperorder) AS avgsalesamount
FROM
    (
        SELECT
            o.custid,
            o.orderid,
            SUM(d.unitprice * d.qty) AS totalsalesamountperorder
        FROM
            Sales.Orders AS o
        INNER JOIN
            Sales.OrderDetails d ON d.orderid = o.orderid
        GROUP BY
            o.custid, o.orderid
    ) AS c
GROUP BY c.custid;
```

10.
```
SELECT
    cy.orderyear,
    cy.totalsalesamount AS curtotalsales,
    py.totalsalesamount AS prevtotalsales,
    ((cy.totalsalesamount - py.totalsalesamount)
        / py.totalsalesamount
        * 100.) AS percentgrowth
FROM
    (
        SELECT
            YEAR(orderdate) AS orderyear,
            SUM(val) AS totalsalesamount
        FROM Sales.OrderValues
        GROUP BY YEAR(orderdate)
    ) AS cy
LEFT OUTER JOIN
    (
        SELECT
            YEAR(orderdate) AS orderyear,
            SUM(val) AS totalsalesamount
        FROM Sales.OrderValues
        GROUP BY YEAR(orderdate)
    ) AS py
    ON cy.orderyear = py.orderyear + 1
ORDER BY cy.orderyear;
```

11.
```
WITH ProductsBeverages AS
    (
        SELECT
            productid,
            productname,
            supplierid,
            unitprice,
            discontinued,
            CASE
                WHEN unitprice > 100. THEN N'high'
                ELSE N'normal'
            END AS pricetype
        FROM Production.Products
        WHERE categoryid = 1
    )
SELECT
    productid,
    productname
FROM ProductsBeverages
```

```
    WHERE pricetype = N'high';
```

12.
```
    WITH c2008 (custid, salesamt2008) AS
        (
            SELECT
                custid,
                SUM(val)
            FROM Sales.OrderValues
            WHERE YEAR(orderdate) = 2008
            GROUP BY custid
        )
    SELECT
        c.custid,
        c.contactname,
        c2008.salesamt2008
    FROM Sales.Customers AS c
    LEFT OUTER JOIN
        c2008 ON c.custid = c2008.custid;
```

13.
```
    WITH
        c2008 (custid, salesamt2008) AS
            (
                SELECT
                custid, SUM(val)
                FROM Sales.OrderValues
                WHERE YEAR(orderdate) = 2008
                GROUP BY custid
            ),
        c2007 (custid, salesamt2007) AS
            (
                SELECT
                custid, SUM(val)
                FROM Sales.OrderValues
                WHERE YEAR(orderdate) = 2007
                GROUP BY custid
            )
    SELECT
        c.custid,
        c.contactname,
        c2008.salesamt2008,
        c2007.salesamt2007,
        COALESCE(
            (c2008.salesamt2008 - c2007.salesamt2007) / c2007.salesamt2007 * 100.,
            0
        )
```

```
AS percentgrowth
FROM Sales.Customers AS c
LEFT OUTER JOIN
    c2008 ON c.custid = c2008.custid
LEFT OUTER JOIN
    c2007 ON c.custid = c2007.custid
ORDER BY percentgrowth DESC;
```

14.
```
SELECT
    custid,
    SUM(val) AS totalsalesamount
FROM
    Sales.OrderValues
WHERE
    YEAR(orderdate) = 2007
GROUP BY
    custid;
```

15.
```
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;
```

16.
```
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;
```

17.
```
SELECT
    custid,
    totalsalesamount
FROM
    dbo.fnGetSalesByCustomer(2007);
```

18.
```
SELECT TOP(3)
    d.productid,
    MAX(p.productname) AS 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 = 1
GROUP BY d.productid
ORDER BY totalsalesamount DESC;
```

19.
```
SELECT TOP(3)
    d.productid,
    MAX(p.productname) AS 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 = 1
GROUP BY d.productid
ORDER BY totalsalesamount DESC;
```

20.
```
CREATE FUNCTION dbo.fnGetTop3ProductsForCustomer
    (@custid AS INT) RETURNS TABLE
AS RETURN
SELECT TOP(3)
    d.productid,
    MAX(p.productname) AS 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
ORDER BY totalsalesamount DESC;


SELECT
    p.productid,
    p.productname,
    p.totalsalesamount
FROM
    dbo.fnGetTop3ProductsForCustomer(1) AS p;
```