Lab Answer Key: Module 13: Using Window Ranking, Offset, and Aggregate Functions

Lab: Using Window Ranking, Offset, and Aggregate Functions

Exercise 1: Writing Queries That Use Ranking Functions

Task 1: Prepare the Lab Environment

1. Ensure that the **20761C-MIA-DC** and **20761C-MIA-SQL** virtual machines are both running, and then log on to **20761C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab13\Starter** folder, right-click **Setup.cmd** and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.

Task 2: Write a SELECT Statement That Uses the ROW_NUMBER Function to Create a Calculated Column

1. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
2. On the **File** menu, click **Open** and click **Project/Solution**.
3. In the **Open Project** window, open the project **D:\Labfiles\Lab13\Starter\Project\Project.ssmssln**.
4. In Solution Explorer, double-click the query **51 - Lab Exercise 1.sql**.
5. When the query window opens, highlight the statement **USE TSQL**; and click **Execute**.
6. In the query pane, type the following query after the **Task 1** description:

```
SELECT
orderid,
orderdate,
val,
ROW_NUMBER() OVER (ORDER BY orderdate) AS rowno
FROM Sales.OrderValues;
```

7. Highlight the written query and click **Execute**.

Task 3: Add an Additional Column Using the RANK Function

1. Highlight the previous query. On the toolbar, click **Edit** and then **Copy**.
2. In the query window, click the line after the **Task 2** description. On the toolbar, click **Edit** and then **Paste.**
3. Modify the T-SQL statement by adding an additional calculated column. The query should look like this:

```
SELECT
orderid,
orderdate,
val,
ROW_NUMBER() OVER (ORDER BY orderdate) AS rowno,
RANK() OVER (ORDER BY orderdate) AS rankno
FROM Sales.OrderValues;
```

4. Highlight the written query and click **Execute**.
5. Observe the results. What is the difference between the RANK and ROW_NUMBER functions? The ROW_NUMBER function provides unique sequential integer values within the partition. The RANK function assigns the same ranking value to rows with the same values in the specified sort columns when the ORDER BY list is not unique. Also, the RANK function skips the next number if there is a tie in the ranking value.

Task 4: Write A SELECT Statement to Calculate a Rank, Partitioned by Customer and Ordered by the Order Value

1. In the query pane, type the following query after the **Task 3** description:

```
SELECT
orderid,
orderdate,
custid,
val,
RANK() OVER (PARTITION BY custid ORDER BY val DESC) AS orderrankno FROM
Sales.OrderValues;
```

2. Highlight the written query and click **Execute**.

Task 5: Write a SELECT Statement to Rank Orders, Partitioned by Customer and Order Year, and Ordered by the Order Value

1. In the query pane, type the following query after the **Task 4** description:

```
SELECT
custid,
val,
YEAR(orderdate) as orderyear,
RANK() OVER (PARTITION BY custid, YEAR(orderdate) ORDER BY val DESC) AS orderrankno
FROM Sales.OrderValues;
```

2. Highlight the written query and click **Execute**.

Task 6: Filter Only Orders with the Top Two Ranks

1. Highlight the previous query. On the toolbar, click **Edit** and then **Copy**.
2. In the query window, click the line after the **Task 5** description. On the toolbar, click **Edit** and then **Paste**.
3. Modify the T-SQL statement to look like this:

```
SELECT
s.custid,
s.orderyear,
s.orderrankno,
s.val
FROM
(
SELECT
custid,
val,
YEAR(orderdate) as orderyear,
RANK() OVER (PARTITION BY custid, YEAR(orderdate) ORDER BY val DESC) AS orderrankno
FROM Sales.OrderValues
) AS s
WHERE s.orderrankno <= 2;
```

4. Highlight the written query and click **Execute**.

Result: After this exercise, you should know how to use ranking functions in T-SQL statements.

Exercise 2: Writing Queries That Use Offset Functions

Task 1: Write a SELECT Statement to Retrieve the Next Row Using a Common Table Expression (CTE)

1. In Solution Explorer, double-click the query **61 - Lab Exercise 2.sql**.
2. When the query window opens, highlight the statement **USE TSQL;** and click **Execute**.
3. In the query pane, type the following query after the **Task 1** description:

```
WITH OrderRows AS
(
SELECT
orderid,
orderdate,
ROW_NUMBER() OVER (ORDER BY orderdate, orderid) AS rowno,
val
FROM Sales.OrderValues
)
SELECT
o.orderid,
o.orderdate,
o.val,
o2.val as prevval,
o.val - o2.val as diffprev
FROM OrderRows AS o
LEFT OUTER JOIN OrderRows AS o2 ON o.rowno = o2.rowno + 1;
```

4. Highlight the written query and click **Execute**.

Task 2: Add a Column to Display the Running Sales Total

1. In the query pane, type the following query after the **Task 2** description:

```
SELECT
orderid,
orderdate,
val,
LAG(val) OVER (ORDER BY orderdate, orderid) AS prevval,
val - LAG(val) OVER (ORDER BY orderdate, orderid) AS diffprev
FROM Sales.OrderValues;
```

2. Highlight the written query and click **Execute**.

Task 3: Analyze the Sales Information for the Year 2007

1. In the query pane, type the following query after the **Task 3** description:

```
WITH SalesMonth2007 AS
(
SELECT
MONTH(orderdate) AS monthno,
SUM(val) AS val
FROM Sales.OrderValues
WHERE orderdate >= '20070101' AND orderdate < '20080101'
GROUP BY MONTH(orderdate)
)
SELECT
monthno,
val,
(LAG(val, 1, 0) OVER (ORDER BY monthno) + LAG(val, 2, 0) OVER (ORDER BY monthno) +
LAG(val, 3, 0) OVER (ORDER BY monthno)) / 3 AS avglast3months,
val - FIRST_VALUE(val) OVER (ORDER BY monthno ROWS UNBOUNDED PRECEDING) AS diffjanuary,
LEAD(val) OVER (ORDER BY monthno) AS nextval
FROM SalesMonth2007;
```

2. Highlight the written query and click **Execute**.

Result: After this exercise, you should be able to use the offset functions in your T-SQL statements.

Exercise 3: Writing Queries That Use Window Aggregate Functions

Task 1: Write a SELECT Statement to Display the Contribution of Each Customer's Order Compared to That Customer's Total Purchase

1. In Solution Explorer, double-click the query **71 - Lab Exercise 3.sql**.
2. When the query window opens, highlight the statement **USE TSQL;** and click **Execute**.
3. In the query pane, type the following query after the **Task 1** description:

```
SELECT
custid,
orderid,
orderdate,
val,
100. * val / SUM(val) OVER (PARTITION BY custid) AS percoftotalcust
FROM Sales.OrderValues
ORDER BY custid, percoftotalcust DESC;
```

4. Highlight the written query and click **Execute**.

Task 2: Add a Column to Display the Running Sales Total

1. Highlight the previous query. On the toolbar, click **Edit** and then **Copy**.
2. In the query window, click the line after the **Task 2** description. On the toolbar, click **Edit** and then **Paste**.
3. Modify the T-SQL statement by adding an additional calculated column. The query should look like this:

```
SELECT
custid,
orderid,
orderdate,
val,
100. * val / SUM(val) OVER (PARTITION BY custid) AS percoftotalcust,
SUM(val) OVER (PARTITION BY custid
ORDER BY orderdate, orderid
ROWS BETWEEN UNBOUNDED PRECEDING
AND CURRENT ROW) AS runval
FROM Sales.OrderValues;
```

4. Highlight the written query and click **Execute**.

Task 3: Analyze the Year-to-Date Sales Amount and Average Sales Amount for the Last Three Months

1. In the query pane, type the following query after the **Task 3** description:

```
WITH SalesMonth2007 AS
(
SELECT
MONTH(orderdate) AS monthno,
SUM(val) AS val
FROM Sales.OrderValues
WHERE orderdate >= '20070101' AND orderdate < '20080101'
GROUP BY MONTH(orderdate)
)
SELECT
monthno,
val,
AVG(val) OVER (ORDER BY monthno ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS
```

```
avglast3months,
SUM(val) OVER (ORDER BY monthno ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS
ytdval
FROM SalesMonth2007;
```

2. Highlight the written query and click **Execute**.

3. Close SQL Server Management Studio without saving any changes.

Result: After this exercise, you should have a basic understanding of how to use window aggregate functions in T-SQL statements.