Database Advance Job Sheet-7: Window Ranking, Offset, Aggregate Function

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 statement to retrieve the orderid, orderdate, and val columns as well as the calculation result column named rowno from the Sales.OrderValues view! Use the ROW_NUMBER function to return rowno, sorting the row number by the orderdate column!

num1.sql

minggu7 > query > num1.sql

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

num1.sql

RESULTS

	orderid	orderdate	val	rowno
1	10248	2006-07-04 00...	440.00	1
2	10249	2006-07-05 00...	1863.40	2
3	10250	2006-07-08 00...	1552.60	3
4	10251	2006-07-08 00...	654.06	4
5	10252	2006-07-09 00...	3597.90	5
6	10253	2006-07-10 00...	1444.80	6
7	10254	2006-07-11 00...	556.62	7
8	10255	2006-07-12 00...	2490.50	8
9	10256	2006-07-15 00...	517.80	9
10	10257	2006-07-16 00...	1119.90	10

2. Copy the T-SQL in question no 1. Then modify it by inserting an additional column named rankno. To create rankno use the RANK function with the rank order based on the orderdate column!

num1.sql num2.sql

minggu7 > query > num2.sql

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

num2.sql

RESULTS CTRL

	orderid	orderdate	val	rowno	rankno
1	10248	2006-07-04 00...	440.00	1	1
2	10249	2006-07-05 00...	1863.40	2	2
3	10250	2006-07-08 00...	1552.60	3	3
4	10251	2006-07-08 00...	654.06	4	3
5	10252	2006-07-09 00...	3597.90	5	5
6	10253	2006-07-10 00...	1444.80	6	6
7	10254	2006-07-11 00...	556.62	7	7
8	10255	2006-07-12 00...	2490.50	8	8
9	10256	2006-07-15 00...	517.80	9	9
10	10257	2006-07-16 00...	1119.90	10	10

3. What is the difference between the RANK function and the ROW_NUMBER function?

- ROW_NUMBER and RANK are similar. ROW_NUMBER numbers all rows sequentially (for example 1, 2, 3, 4, 5). RANK provides the same numeric value for ties (for example 1, 2, 2, 4, 5).

4. Write a SELECT statement to retrieve the columns orderid, orderdate, custid, and val and calculate the column named orderrankno from the Sales.OrderValues view. The orderrankno column should display the ranking per customer independently, based on the order val in descending order!

num1.sql num2.sql num4.sql

minggu7 > query > num4.sql

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

num4.sql

RESULTS

	orderid	orderdate	custid	val	orderrankno
1	11011	2008-04-09 00...	1	933.50	1
2	10692	2007-10-03 00...	1	878.00	2
3	10835	2008-01-15 00...	1	845.80	3
4	10643	2007-08-25 00...	1	814.50	4
5	10952	2008-03-16 00...	1	471.20	5
6	10702	2007-10-13 00...	1	330.00	6
7	10926	2008-03-04 00...	2	514.40	1
8	10625	2007-08-08 00...	2	479.75	2
9	10759	2007-11-28 00...	2	320.00	3
10	10308	2006-09-18 00...	2	88.80	4

5. Write a SELECT statement to retrieve the custid and val columns from view

Sales.OrderValues view. Add the following two columns:

1) orderyear as the year of the orderdate column

2) orderrankno as the order number, partitioned by customer and order year, and sorted by order value in descending order!

num5.sql

minggu7 > query > num5.sql

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

num5.sql

RESULTS

	custid	val	orderyear	orderrankno
1	1	878.00	2007	1
2	1	814.50	2007	2
3	1	330.00	2007	3
4	1	933.50	2008	1
5	1	845.80	2008	2
6	1	471.20	2008	3
7	2	88.80	2006	1
8	2	479.75	2007	1
9	2	320.00	2007	2
10	2	514.40	2008	1

6. Copy the query answer to question 6 and modify it to filter only the orders with the first two ranks based on the orderrankno column!

num5.sql num6.sql

minggu7 > query > num6.sql

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

num6.sql

RESULTS

	custid	orderyear	orderrankno	val
1	1	2007	1	878.00
2	1	2007	2	814.50
3	1	2008	1	933.50
4	1	2008	2	845.80
5	2	2006	1	88.80
6	2	2007	1	479.75
7	2	2007	2	320.00
8	2	2008	1	514.40
9	3	2006	1	403.20
10	3	2007	1	2082.00

Practicum 2

7. Create a CTE (common table expression) with the name OrderRows based on the query that query that retrieves the orderid, orderdate, and val columns from the Sales.OrderValues view. Add a columnwith the name rowno using the ROW_NUMBER function that is sorted by the orderdate and orderid columns!

num7.sql

minggu7 > query > num7.sql

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

num7.sql

RESULTS

	orderid	orderdate	val	rowno
1	10248	2006-07-04 00...	440.00	1
2	10249	2006-07-05 00...	1863.40	2
3	10250	2006-07-08 00...	1552.60	3
4	10251	2006-07-08 00...	654.06	4
5	10252	2006-07-09 00...	3597.90	5
6	10253	2006-07-10 00...	1444.80	6
7	10254	2006-07-11 00...	556.62	7
8	10255	2006-07-12 00...	2490.50	8
9	10256	2006-07-15 00...	517.80	9
10	10257	2006-07-16 00...	1119.90	10

8. Write SELECT statement against CTE and use LEFT JOIN with the same CTE to retrieve the current row and previous row based on the rowno column. Return the orderid, orderdate, and val columns for the current row and the val column for the previous row as prevval. column for the previous row as prevval. Add a calculation result column with the name diffprev that shows the difference between current val and previous val!

num8.sql

minggu7 > query > num8.sql

```sql
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, r.val AS prevval, o.val - r.
    val AS diffprev
FROM OrderRows AS o
LEFT OUTER JOIN OrderRows AS r ON o.rowno = r.rowno + 1;
```

num8.sql

RESULTS CTRL+

	orderid	orderdate	val	prevval	diffprev
1	10248	2006-07-04 00...	440.00	*NULL*	*NULL*
2	10249	2006-07-05 00...	1863.40	440.00	1423.40
3	10250	2006-07-08 00...	1552.60	1863.40	-310.80
4	10251	2006-07-08 00...	654.06	1552.60	-898.54
5	10252	2006-07-09 00...	3597.90	654.06	2943.84
6	10253	2006-07-10 00...	1444.80	3597.90	-2153.10
7	10254	2006-07-11 00...	556.62	1444.80	-888.18
8	10255	2006-07-12 00...	2490.50	556.62	1933.88
9	10256	2006-07-15 00...	517.80	2490.50	-1972.70
10	10257	2006-07-16 00...	1119.90	517.80	602.10

9. Write a SELECT statement using the LAG function to get the same results as with the query in question no.2! The query in this question does not use CTE.

num8.sql num9.sql

minggu7 > query > num9.sql

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

num9.sql

RESULTS

	orderid	orderdate	val	prevval	diffprev
1	10248	2006-07-04 00...	440.00	*NULL*	*NULL*
2	10249	2006-07-05 00...	1863.40	440.00	1423.40
3	10250	2006-07-08 00...	1552.60	1863.40	-310.80
4	10251	2006-07-08 00...	654.06	1552.60	-898.54
5	10252	2006-07-09 00...	3597.90	654.06	2943.84
6	10253	2006-07-10 00...	1444.80	3597.90	-2153.10
7	10254	2006-07-11 00...	556.62	1444.80	-888.18
8	10255	2006-07-12 00...	2490.50	556.62	1933.88
9	10256	2006-07-15 00...	517.80	2490.50	-1972.70
10	10257	2006-07-16 00...	1119.90	517.80	602.10

10. Create a CTE named SalesMonth2007 that creates two columns viz, monthno (number of months from orderdate column) and val (aggregate from val column)! Then filter the results only for order year 2007 and grouped by monthno!

num9.sql num10.sql ×

minggu7 > query > num10.sql

```sql
WITH SalesMonth2007 AS (
    SELECT
        MONTH(orderdate) AS monthno, SUM(val) AS val
    FROM Sales.OrderValues
    WHERE YEAR(orderdate) = 2007
    GROUP BY MONTH(orderdate)
)
SELECT monthno, val
FROM SalesMonth2007;
```

num10.sql ×

RESULTS

	monthno	val
1	1	61258.08
2	2	38483.64
3	3	38547.23
4	4	53032.95
5	5	53781.30
6	6	36362.82
7	7	51020.86
8	8	47287.68
9	9	55629.27
10	10	66749.23
11	11	43533.80
12	12	71398.44

11. Write a SELECT statement that will retrieve the montohno and val columns from CTE and add 3 columns to display, namely:

1) avglast3months (average sales amount of the last three months)

2) diffjanuary (the difference between current val and val in January, use the function FIRST_VALUE)

3) nextval (the value of the val column in the next month) Information: The average amount for the last three months is not calculated correctly because the sum of the total of the first 2 months is divided by 3.

num10.sql num11.sql ×

minggu7 > query > num11.sql

```sql
WITH SalesMonth2007 AS (
    SELECT
        MONTH(orderdate) AS monthno,
        SUM(val) AS val
    FROM Sales.OrderValues
    WHERE YEAR(orderdate) = 2007
    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;
```

num11.sql ×

RESULTS CTRL+AL

	monthno	val	avglast3months	diffjanuary	nextval
1	1	61258.08	0.000000	0.00	38483.64
2	2	38483.64	20419.360000	-22774.44	38547.23
3	3	38547.23	33247.240000	-22710.85	53032.95
4	4	53032.95	46096.316666	-8225.13	53781.30
5	5	53781.30	43354.606666	-7476.78	36362.82
6	6	36362.82	48453.826666	-24895.26	51020.86
7	7	51020.86	47725.690000	-10237.22	47287.68
8	8	47287.68	47054.993333	-13970.40	55629.27
9	9	55629.27	44890.453333	-5628.81	66749.23
10	10	66749.23	51312.603333	5491.15	43533.80

Practicum 3

12. Write a SELECT statement to retrieve the custid, orderid, orderdate, and val from the Sales.OrderValues view. Add a column named percoftotalcust that contains the percentage value of each sales order amount compared to the total sales for that customer!

num11.sql num12.sql

minggu7 > query > num12.sql

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

num12.sql

RESULTS

	custid	orderid	orderdate	val	percoftotalcust
1	1	11011	2008-04-09 00...	933.50	21.846477884...
2	1	10952	2008-03-16 00...	471.20	11.027381230...
3	1	10835	2008-01-15 00...	845.80	19.794055698...
4	1	10702	2007-10-13 00...	330.00	7.7229113035...
5	1	10692	2007-10-03 00...	878.00	20.547624619...
6	1	10643	2007-08-25 00...	814.50	19.061549262...
7	2	10926	2008-03-04 00...	514.40	36.665597491...
8	2	10759	2007-11-28 00...	320.00	22.809080865...
9	2	10625	2007-08-08 00...	479.75	34.195801703...
10	2	10308	2006-09-18 00...	88.80	6.3295199401...

13. Copy the previous SELECT statement and modify it by adding a column named runval! This column should contain the total sales that are occurring for each customer by order date, using orderid as the tiebreaker.

num12.sql num13.sql

minggu7 > query > num13.sql

```
1  SELECT
2      custid,
3      orderid,
4      orderdate,
5      val,
6      val / SUM(val) OVER (PARTITION BY custid) * 100 AS
       percoftotalcust,
7      SUM(val) OVER (PARTITION BY custid ORDER BY orderdate,
       orderid) AS runval
8  FROM Sales.OrderValues
9  ORDER BY custid, orderdate, orderid;
```

num13.sql

RESULTS CTRL

	custid	orderid	orderdate	val	percoftotalcust	runval
1	1	10643	2007-08-25 00...	814.50	19.061549262...	814.50
2	1	10692	2007-10-03 00...	878.00	20.547624619...	1692.50
3	1	10702	2007-10-13 00...	330.00	7.7229113035...	2022.50
4	1	10835	2008-01-15 00...	845.80	19.794055698...	2868.30
5	1	10952	2008-03-16 00...	471.20	11.027381230...	3339.50
6	1	11011	2008-04-09 00...	933.50	21.846477884...	4273.00
7	2	10308	2006-09-18 00...	88.80	6.3295199401...	88.80
8	2	10625	2007-08-08 00...	479.75	34.195801703...	568.55
9	2	10759	2007-11-28 00...	320.00	22.809080865...	888.55
10	2	10926	2008-03-04 00...	514.40	36.665597491...	1402.95

14. Copy the CTE SalesMonth2007 in experiment 2. Write a SELECT statement to retrieve the monthno and val columns. Add two calculated columns:

1) avglast3months. This column should contain the average sales amount for the three months prior to the current month using the aggregate window function. Assume that there are there are no missing months.

2) ytdval This column should contain the cumulative sales value up to the current month.

num12.sql num13.sql num14.sql ×

minggu7 > query > num14.sql

```sql
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;
```

num14.sql ×

RESULTS

	monthno	val	avglast3months	ytdval
1	1	61258.08	61258.080000	61258.08
2	2	38483.64	49870.860000	99741.72
3	3	38547.23	46096.316666	138288.95
4	4	53032.95	47830.475000	191321.90
5	5	53781.30	45961.280000	245103.20
6	6	36362.82	45431.075000	281466.02
7	7	51020.86	48549.482500	332486.88
8	8	47287.68	47113.165000	379774.56
9	9	55629.27	47575.157500	435403.83
10	10	66749.23	55171.760000	502153.06
11	11	43533.80	53299.995000	545686.86
12	12	71398.44	59327.685000	617085.30