```sql
/*
============================================================
WINDOW FUNCTIONS PRACTICE - 30 TASKS WITH SOLUTIONS
ORDERS TABLE
============================================================
*/

USE SalesDB;

/*
------------------------------------------------------------
TASK 1) TOTAL NUMBER OF ORDERS
------------------------------------------------------------
Show total count of all orders for each row
*/
SELECT
    OrderID,
    COUNT(*) OVER() AS Total_Orders
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 2) TOTAL SALES PER CUSTOMER
------------------------------------------------------------
Calculate total sales for each customer
*/
SELECT
    OrderID,
    CustomerID,
    Sales,
    SUM(Sales) OVER(PARTITION BY CustomerID) AS Customer_Total_Sales
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 3) AVERAGE SALES PER PRODUCT
------------------------------------------------------------
Find average sales inside each product
*/
SELECT
    OrderID,
    ProductID,
    Sales,
    AVG(Sales) OVER(PARTITION BY ProductID) AS Avg_Product_Sales
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 4) MAX SALES PER PRODUCT
------------------------------------------------------------
Find highest sale inside each product
*/
SELECT
    OrderID,
    ProductID,
    Sales,
    MAX(Sales) OVER(PARTITION BY ProductID) AS Max_Product_Sales
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 5) MIN SALES PER CUSTOMER
------------------------------------------------------------
Find lowest sale per customer
*/
SELECT
    OrderID,
    CustomerID,
    Sales,
    MIN(Sales) OVER(PARTITION BY CustomerID) AS Min_Customer_Sales
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 6) RUNNING TOTAL OF SALES
------------------------------------------------------------
Calculate cumulative sales based on order date
*/
SELECT
    OrderID,
    OrderDate,
    Sales,
    SUM(Sales) OVER(ORDER BY OrderDate) AS Running_Total
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 7) RUNNING TOTAL PER CUSTOMER
------------------------------------------------------------
Calculate cumulative sales per customer
*/
SELECT
    CustomerID,
    OrderDate,
    Sales,
    SUM(Sales) OVER(PARTITION BY CustomerID ORDER BY OrderDate)
        AS Running_Total_Per_Customer
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 8) ROW NUMBER BY SALES
------------------------------------------------------------
Assign sequence based on highest sales
*/
SELECT
    OrderID,
    Sales,
    ROW_NUMBER() OVER(ORDER BY Sales DESC) AS Sales_Row_Number
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 9) RANK SALES PER PRODUCT
------------------------------------------------------------
Rank sales inside each product
*/
SELECT
    ProductID,
    OrderID,
    Sales,
    RANK() OVER(PARTITION BY ProductID ORDER BY Sales DESC) AS Product_Rank
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 10) DENSE RANK SALES PER PRODUCT
------------------------------------------------------------
Rank without skipping numbers
*/
SELECT
    ProductID,
    OrderID,
    Sales,
    DENSE_RANK() OVER(PARTITION BY ProductID ORDER BY Sales DESC) AS Dense_Rank
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 11) FIRST ORDER DATE PER CUSTOMER
------------------------------------------------------------
Find earliest order for each customer
*/
SELECT
    CustomerID,
    OrderID,
    OrderDate,
    MIN(OrderDate) OVER(PARTITION BY CustomerID) AS First_Order_Date
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 12) LAST ORDER DATE PER CUSTOMER
------------------------------------------------------------
Find most recent order per customer
*/
SELECT
    CustomerID,
    OrderID,
    OrderDate,
    MAX(OrderDate) OVER(PARTITION BY CustomerID) AS Last_Order_Date
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 13) PREVIOUS SALES
------------------------------------------------------------
Show previous sales value
*/
SELECT
    OrderID,
    Sales,
    LAG(Sales) OVER(ORDER BY OrderDate) AS Previous_Sales
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 14) NEXT SALES
------------------------------------------------------------
Show next sales value
*/
SELECT
    OrderID,
    Sales,
    LEAD(Sales) OVER(ORDER BY OrderDate) AS Next_Sales
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 15) SALES DIFFERENCE FROM PREVIOUS
------------------------------------------------------------
Calculate difference between current and previous sale
*/
SELECT
    OrderID,
    Sales,
    Sales - LAG(Sales) OVER(ORDER BY OrderDate) AS Sales_Difference
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 16) PERCENTAGE OF TOTAL SALES
------------------------------------------------------------
Calculate sales contribution percentage
*/
SELECT
    OrderID,
    Sales,
    100.0 * Sales / SUM(Sales) OVER() AS Sales_Percentage
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 17) COUNT ORDERS PER CUSTOMER
------------------------------------------------------------
Count number of orders for each customer
*/
SELECT
    OrderID,
    CustomerID,
    COUNT(*) OVER(PARTITION BY CustomerID) AS Orders_Per_Customer
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 18) SALES ABOVE CUSTOMER AVERAGE
------------------------------------------------------------
Compare sales with customer's average
*/
SELECT
    OrderID,
    CustomerID,
    Sales,
    AVG(Sales) OVER(PARTITION BY CustomerID) AS Customer_Avg
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 12) LAST ORDER DATE PER CUSTOMER
------------------------------------------------------------
Find most recent order per customer
*/
SELECT
    CustomerID,
    OrderID,
    OrderDate,
    MAX(OrderDate) OVER(PARTITION BY CustomerID) AS Last_Order_Date
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 13) PREVIOUS SALES
------------------------------------------------------------
Show previous sales value
*/
SELECT
    OrderID,
    Sales,
    LAG(Sales) OVER(ORDER BY OrderDate) AS Previous_Sales
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 14) NEXT SALES
------------------------------------------------------------
Show next sales value
*/
SELECT
    OrderID,
    Sales,
    LEAD(Sales) OVER(ORDER BY OrderDate) AS Next_Sales
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 15) SALES DIFFERENCE FROM PREVIOUS
------------------------------------------------------------
Calculate difference between current and previous sale
*/
SELECT
    OrderID,
    Sales,
    Sales - LAG(Sales) OVER(ORDER BY OrderDate) AS Sales_Difference
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 16) PERCENTAGE OF TOTAL SALES
------------------------------------------------------------
Calculate sales contribution percentage
*/
SELECT
    OrderID,
    Sales,
    100.0 * Sales / SUM(Sales) OVER() AS Sales_Percentage
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 17) COUNT ORDERS PER CUSTOMER
------------------------------------------------------------
Count number of orders for each customer
*/
SELECT
    OrderID,
    CustomerID,
    COUNT(*) OVER(PARTITION BY CustomerID) AS Orders_Per_Customer
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 18) SALES ABOVE CUSTOMER AVERAGE
------------------------------------------------------------
Compare sales with customer's average
*/
SELECT
    OrderID,
    CustomerID,
    Sales,
    AVG(Sales) OVER(PARTITION BY CustomerID) AS Customer_Avg
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 19) DAYS BETWEEN ORDERS
------------------------------------------------------------
Find gap between consecutive orders
*/
SELECT
    OrderID,
    OrderDate,
    DATEDIFF(DAY,
        LAG(OrderDate) OVER(ORDER BY OrderDate),
        OrderDate) AS Days_Between
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 20) TOP SALE PER PRODUCT
------------------------------------------------------------
Find highest sales order in each product
*/
SELECT *
FROM (
    SELECT *,
           ROW_NUMBER() OVER(PARTITION BY ProductID ORDER BY Sales DESC) AS rn
    FROM Sales.Orders
) t
WHERE rn = 1;
```

```sql
/*
------------------------------------------------------------
TASK 21) TOP 2 SALES PER PRODUCT
------------------------------------------------------------
Find top two sales in each product
*/
SELECT *
FROM (
    SELECT *,
           ROW_NUMBER() OVER(PARTITION BY ProductID ORDER BY Sales DESC) AS rn
    FROM Sales.Orders
) t
WHERE rn <= 2;



/*
------------------------------------------------------------
TASK 22) SALES CUMULATIVE PER PRODUCT
------------------------------------------------------------
Running total inside each product
*/
SELECT
    ProductID,
    OrderDate,
    Sales,
    SUM(Sales) OVER(PARTITION BY ProductID ORDER BY OrderDate)
        AS Product_Running_Total
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 23) ORDER COUNT RUNNING
------------------------------------------------------------
Running count of orders over time
*/
SELECT
    OrderID,
    COUNT(*) OVER(ORDER BY OrderDate) AS Running_Count
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 24) SALES VS MAX SALES
------------------------------------------------------------
Compare each sale with max sale
*/
SELECT
    OrderID,
    Sales,
    MAX(Sales) OVER() AS Max_Sale
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 25) SALES GAP FROM MAX SALE
------------------------------------------------------------
Difference from highest sale
*/
SELECT
    OrderID,
    Sales,
    MAX(Sales) OVER() - Sales AS Gap_From_Max
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 26) CUSTOMER FIRST ORDER FLAG
------------------------------------------------------------
Mark first order per customer
*/
SELECT
    OrderID,
    CustomerID,
    OrderDate,
    CASE WHEN ROW_NUMBER() OVER(PARTITION BY CustomerID ORDER BY OrderDate) = 1
         THEN 'First Order'
         ELSE 'Repeat'
    END AS Order_Type
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 27) CUSTOMER LAST ORDER FLAG
------------------------------------------------------------
Mark last order per customer
*/
SELECT
    OrderID,
    CustomerID,
    OrderDate,
    CASE WHEN ROW_NUMBER() OVER(PARTITION BY CustomerID ORDER BY OrderDate DESC) = 1
         THEN 'Last Order'
         ELSE 'Old'
    END AS Order_Status
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 28) SALES MOVING AVERAGE
------------------------------------------------------------
Calculate 3-row moving average
*/
SELECT
    OrderID,
    Sales,
    AVG(Sales) OVER(
        ORDER BY OrderDate
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
    ) AS Moving_Avg
FROM Sales.Orders;


/*
------------------------------------------------------------
TASK 29) SALES MOVING SUM
------------------------------------------------------------
Calculate rolling sum of last 3 orders
*/
SELECT
    OrderID,
    Sales,
    SUM(Sales) OVER(
        ORDER BY OrderDate
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
    ) AS Moving_Sum
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 30) CUSTOMER SALES PERCENTAGE SHARE
------------------------------------------------------------
Find percentage of each customer's sales
*/
SELECT
    CustomerID,
    OrderID,
    Sales,
    100.0 * Sales / SUM(Sales) OVER(PARTITION BY CustomerID)
        AS Customer_Share_Percentage
FROM Sales.Orders;
```

```sql
/*
    ========================================================
    KEY RULES TO REMEMBER
    ========================================================
    1) OVER() is REQUIRED in all window functions
    2) PARTITION BY = like GROUP BY but keeps rows
    3) ORDER BY inside OVER controls calculation order
    4) Window functions DO NOT reduce rows
    5) Execution order:
    FROM → WHERE → GROUP BY → WINDOW → ORDER BY

========================================================
*/


/*
      ============================================================
      == HARD WINDOW FUNCTION INTERVIEW TASKS – WITH SOLUTIONS ==
      ============================================================
*/


/*
------------------------------------------------------------
TASK 1) TOP 3 HIGHEST SALES ORDERS
------------------------------------------------------------
*/
SELECT *
FROM (
    SELECT *,
           ROW_NUMBER() OVER(ORDER BY Sales DESC) AS rn
    FROM Sales.Orders
) t
WHERE rn <= 3;



/*
------------------------------------------------------------
TASK 2) TOP 2 SALES PER PRODUCT
------------------------------------------------------------
*/
SELECT *
FROM (
    SELECT *,
           ROW_NUMBER() OVER(PARTITION BY ProductID ORDER BY Sales DESC) AS rn
    FROM Sales.Orders
) t
WHERE rn <= 2;
```

```sql
/*
-------------------------------------------------------------
TASK 3) LOWEST SALE PER CUSTOMER
-------------------------------------------------------------
*/
SELECT *
FROM (
    SELECT *,
           ROW_NUMBER() OVER(PARTITION BY CustomerID ORDER BY Sales) AS rn
    FROM Sales.Orders
) t
WHERE rn = 1;



/*
-------------------------------------------------------------
TASK 4) RANK CUSTOMERS BY TOTAL SALES
-------------------------------------------------------------
*/
SELECT
    CustomerID,
    SUM(Sales) AS Total_Sales,
    RANK() OVER(ORDER BY SUM(Sales) DESC) AS Rank_Customer
FROM Sales.Orders
GROUP BY CustomerID;



/*
-------------------------------------------------------------
TASK 5) TOP 10% SALES ORDERS
-------------------------------------------------------------
*/
SELECT *
FROM (
    SELECT *,
           NTILE(10) OVER(ORDER BY Sales DESC) AS Bucket
    FROM Sales.Orders
) t
WHERE Bucket = 1;



/*
-------------------------------------------------------------
TASK 6) SALES DIFFERENCE FROM PRODUCT MAX
-------------------------------------------------------------
*/
SELECT
    OrderID,
    ProductID,
    Sales,
    MAX(Sales) OVER(PARTITION BY ProductID) - Sales AS Difference
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 7) SALES ABOVE PRODUCT AVERAGE
------------------------------------------------------------
*/
SELECT *
FROM (
    SELECT *,
           AVG(Sales) OVER(PARTITION BY ProductID) AS Avg_Sales
    FROM Sales.Orders
) t
WHERE Sales > Avg_Sales;



/*
------------------------------------------------------------
TASK 8) CUSTOMER RUNNING TOTAL SALES
------------------------------------------------------------
*/
SELECT
    CustomerID,
    OrderDate,
    Sales,
    SUM(Sales) OVER(PARTITION BY CustomerID ORDER BY OrderDate)
        AS Running_Total
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 9) CUSTOMER SALES GROWTH FROM PREVIOUS
------------------------------------------------------------
*/
SELECT
    CustomerID,
    OrderDate,
    Sales,
    Sales - LAG(Sales) OVER(PARTITION BY CustomerID ORDER BY OrderDate)
        AS Growth
FROM Sales.Orders;


/*
------------------------------------------------------------
TASK 10) DAYS BETWEEN CUSTOMER ORDERS
------------------------------------------------------------
*/
SELECT
    CustomerID,
    OrderDate,
    DATEDIFF(DAY,
        LAG(OrderDate) OVER(PARTITION BY CustomerID ORDER BY OrderDate),
        OrderDate) AS Gap_Days
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 11) FIRST ORDER PER CUSTOMER
------------------------------------------------------------
*/
SELECT *
FROM (
    SELECT *,
           ROW_NUMBER() OVER(PARTITION BY CustomerID ORDER BY OrderDate) rn
    FROM Sales.Orders
) t
WHERE rn = 1;



/*
------------------------------------------------------------
TASK 12) LAST ORDER PER CUSTOMER
------------------------------------------------------------
*/
SELECT *
FROM (
    SELECT *,
           ROW_NUMBER() OVER(PARTITION BY CustomerID ORDER BY OrderDate DESC) rn
    FROM Sales.Orders
) t
WHERE rn = 1;



/*
------------------------------------------------------------
TASK 13) SECOND HIGHEST SALE PER PRODUCT
------------------------------------------------------------
*/
SELECT *
FROM (
    SELECT *,
           DENSE_RANK() OVER(PARTITION BY ProductID ORDER BY Sales DESC) rnk
    FROM Sales.Orders
) t
WHERE rnk = 2;



/*
------------------------------------------------------------
TASK 14) MOVING AVERAGE (LAST 3 ORDERS)
------------------------------------------------------------
*/
SELECT
    OrderID,
    Sales,
    AVG(Sales) OVER(
        ORDER BY OrderDate
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
    ) AS Moving_Avg
FROM Sales.Orders;
```

```sql
/*
-------------------------------------------------------------
TASK 15) MOVING SUM LAST 5 ORDERS
-------------------------------------------------------------
*/
SELECT
    OrderID,
    Sales,
    SUM(Sales) OVER(
        ORDER BY OrderDate
        ROWS BETWEEN 4 PRECEDING AND CURRENT ROW
    ) AS Moving_Sum
FROM Sales.Orders;



/*
-------------------------------------------------------------
TASK 16) PERCENTAGE SHARE PER CUSTOMER
-------------------------------------------------------------
*/
SELECT
    CustomerID,
    OrderID,
    Sales,
    100.0 * Sales / SUM(Sales) OVER(PARTITION BY CustomerID)
        AS Share_Percentage
FROM Sales.Orders;



/*
-------------------------------------------------------------
TASK 17) PERCENTAGE SHARE OF TOTAL SALES
-------------------------------------------------------------
*/
SELECT
    OrderID,
    Sales,
    100.0 * Sales / SUM(Sales) OVER() AS Company_Share
FROM Sales.Orders;



/*
-------------------------------------------------------------
TASK 18) FLAG ABOVE AVERAGE SALES
-------------------------------------------------------------
*/
SELECT
    OrderID,
    Sales,
    CASE WHEN Sales > AVG(Sales) OVER()
         THEN 'Above Avg'
         ELSE 'Below Avg'
    END AS Status
FROM Sales.Orders;
```

```sql
/*
-------------------------------------------------------------
TASK 19) PRODUCT SALES RANK PERCENTILE
-------------------------------------------------------------
*/
SELECT
    ProductID,
    OrderID,
    Sales,
    PERCENT_RANK() OVER(PARTITION BY ProductID ORDER BY Sales) AS Percentile
FROM Sales.Orders;



/*
-------------------------------------------------------------
TASK 20) CUSTOMER ORDER SEQUENCE
-------------------------------------------------------------
*/
SELECT
    CustomerID,
    OrderID,
    ROW_NUMBER() OVER(PARTITION BY CustomerID ORDER BY OrderDate)
        AS Order_Number
FROM Sales.Orders;



/*
-------------------------------------------------------------
TASK 21) SALES GAP FROM PREVIOUS ORDER
-------------------------------------------------------------
*/
SELECT
    OrderID,
    Sales,
    Sales - LAG(Sales) OVER(ORDER BY OrderDate) AS Gap
FROM Sales.Orders;



/*
-------------------------------------------------------------
TASK 22) SALES GAP FROM NEXT ORDER
-------------------------------------------------------------
*/
SELECT
    OrderID,
    Sales,
    LEAD(Sales) OVER(ORDER BY OrderDate) - Sales AS Next_Gap
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 23) CUSTOMER LIFETIME VALUE RUNNING
------------------------------------------------------------
*/
SELECT
    CustomerID,
    OrderDate,
    SUM(Sales) OVER(PARTITION BY CustomerID ORDER BY OrderDate)
        AS Lifetime_Value
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 24) PRODUCT SALES CUMULATIVE
------------------------------------------------------------
*/
SELECT
    ProductID,
    OrderDate,
    SUM(Sales) OVER(PARTITION BY ProductID ORDER BY OrderDate)
        AS Product_Total
FROM Sales.Orders;



/*
------------------------------------------------------------
TASK 25) FIND DUPLICATE SALES VALUES
------------------------------------------------------------
*/
SELECT *
FROM (
    SELECT *,
           COUNT(*) OVER(PARTITION BY Sales) cnt
    FROM Sales.Orders
) t
WHERE cnt > 1;



/*
------------------------------------------------------------
TASK 26) SALES RANK WITHOUT GAPS
------------------------------------------------------------
*/
SELECT
    OrderID,
    Sales,
    DENSE_RANK() OVER(ORDER BY Sales DESC) AS DenseRank
FROM Sales.Orders;
```

```sql
/*
------------------------------------------------------------
TASK 27) FIND MIDDLE SALE VALUE (MEDIAN APPROX)
------------------------------------------------------------
*/
SELECT *
FROM (
    SELECT *,
           NTILE(2) OVER(ORDER BY Sales) grp
    FROM Sales.Orders
) t
WHERE grp = 1;


/*
------------------------------------------------------------
TASK 28) CUSTOMER ORDER GAP FLAG
------------------------------------------------------------
*/
SELECT
    CustomerID,
    OrderDate,
    CASE WHEN DATEDIFF(DAY,
           LAG(OrderDate) OVER(PARTITION BY CustomerID ORDER BY OrderDate),
           OrderDate) > 30
         THEN 'Long Gap'
         ELSE 'Normal'
    END AS Gap_Status
FROM Sales.Orders;


/*
------------------------------------------------------------
TASK 29) SALES TREND FLAG
------------------------------------------------------------
*/
SELECT
    OrderID,
    Sales,
    CASE WHEN Sales > LAG(Sales) OVER(ORDER BY OrderDate)
         THEN 'Increasing'
         ELSE 'Decreasing'
    END AS Trend
FROM Sales.Orders;


/*
------------------------------------------------------------
TASK 30) IDENTIFY PEAK SALES PER PRODUCT
------------------------------------------------------------
*/
SELECT *
FROM (
    SELECT *,
           RANK() OVER(PARTITION BY ProductID ORDER BY Sales DESC) rnk
    FROM Sales.Orders
) t
```

```
WHERE rnk = 1;
```