RANKING FUNCTIONS

ROW_NUMBER (): Assigns a unique number to each row based on the specified order.

RANK (): Assigns the same rank to rows with the same value and skips the next rank.

DENSE_RANK (): Assigns the same rank to rows with the same value but doesn't skip the rank.

PERCENT_RANK (): Calculates the relative rank of the row within the ordered set.

AGGREGATE WINDOW FUNCTIONS

SUM (), AVG (), MIN (), MAX (): Are used to calculate running totals and trends.

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW: Defines the frame for the window function, including all rows up to the current row.

NAVIGATION FUNCTIONS

LAG (): Retrieves the value of the column from the previous row in the ordered set.

LEAD (): Retrieves the value of the column from the next row in the ordered set.

These functions are useful for period-to-period comparisons and growth analysis.

DISTRIBUTION FUNCTIONS

NTLIE (4): Divides the data into 4 equal parts for the customer segmentations.

CUM_DIST (): Calculates the cumulative distribution of a row within the ordered set, useful for understanding relative standings.

1. Ranking Functions

ROW_NUMBER (), RANK(), DENSE_RANK(), PERCENT_RANK()

SQL Query:

```
-- Ranking Functions: Assign ranks based on revenue
SELECT
   product_id,
   product_name,
   price,
   ROW_NUMBER () OVER (ORDER BY revenue DESC) AS row_num,  -- Unique rank based
on revenue
   RANK () OVER (ORDER BY price DESC) AS rank_pose,           -- Rank with ties
   DENSE_RANK () OVER (ORDER BY price DESC) AS dense_rank,  -- Rank without
skipping
   PERCENT_RANK() OVER (ORDER BY price DESC) AS percent_rank  -- Relative rank
FROM
```

```
    products;
```

---

2. Aggregate Window Functions

SUM(), AVG(), MIN(), MAX()

SQL Query:

```
-- Aggregate Window Functions: Calculate running totals and trends
SELECT
    transaction_date,
    SUM(total_amount) OVER (ORDER BY transaction_date ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS running_total,  -- Running total of
revenue
    AVG(total_amount) OVER (ORDER BY transaction_date ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS average _total ,  -- Average Total
    MIN(total_amount) OVER (ORDER BY transaction_date ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS minimum _total ,   -- Minimum
Total
    MAX(total_amount) OVER (ORDER BY transaction_date ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS Maximum_ Total   -- Maximum
Total
FROM
    Transactions;
            .
```

---

3. Navigation Functions

LAG (), LEAD ()

SQL Query:

```
-- Navigation Functions: Compare current row with previous and next rows
SELECT
    product_ id,
    product_   name,
    price,
    LAG (price, 1) OVER (ORDER BY transaction_date) AS previous_ date,   -- Previous_ date
from the previous row
    LEAD (price, 1) OVER (ORDER BY transaction_date) AS next_ date   -- Next_ date from the
next row
FROM
    products;
```

DISTRIBUTION FUNCTIONS

NTILE(4), CUME_DIST()

SQL QUERY:

```
SELECT
    customer_ id,
   Firstname,
    price,
    NTILE (4) OVER (ORDER BY price DESC) AS customer_ segment,   -- Divide into 4 equal
parts
    CUME_DIST () OVER (ORDER BY price DESC) AS cumulative_distribution   --
Cumulative distribution
FROM
    customers;
```