Names: Nankunda Winny
Id:27959

SCREEN SHOOTS

INNER JOIN

```
online_db=# -- INNER JOIN - Orders with valid customers and products
online_db=# SELECT
online_db-#     o.OrderID,
online_db-#     c.FirstName,
online_db-#     c.LastName,
online_db-#     p.ProductName,
online_db-#     o.Quantity,
online_db-#     o.OrderDate
online_db-# FROM Orders o
online_db-# INNER JOIN Customers c ON o.CustomerID = c.CustomerID
online_db-# INNER JOIN Products p ON o.ProductID = p.ProductID;
 orderid | firstname | lastname | productname  | quantity | orderdate
---------+-----------+----------+--------------+----------+------------
    1001 | Alice     | teta     | Laptop       |        1 | 2026-01-05
    1002 | Bob       | kwizera  | T-Shirt      |        2 | 2026-01-07
    1003 | Alice     | teta     | Headphones   |        1 | 2026-01-10
    1004 | Clara     | kaneza   | Sneakers     |        1 | 2026-01-12
    1005 | Bob       | kwizera  | Coffee Maker |        1 | 2026-01-15
(5 rows)
```

Interpretation :Returns only the rows where there is matching information in both tables. Any data without a match is ignored.

LEFT JOIN:

```
online_db=# --  LEFT JOIN - Customers who have never made an order
online_db=# SELECT
online_db-#     c.CustomerID,
online_db-#     c.FirstName,
online_db-#     c.LastName,
online_db-#     o.OrderID
online_db-# FROM Customers c
online_db-# LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
online_db-# WHERE o.OrderID IS NULL;
 customerid | firstname | lastname | orderid
------------+-----------+----------+---------
          5 | Eva       | uwase    |
          4 | David     | habakuki |
(2 rows)
```

Interpretation: Shows all records from the first table and matching ones from the second. Missing matches appear as NULL.

RIGHT JOIN:

```
online_db=# -- RIGHT JOIN - Products never ordered
online_db=# SELECT
online_db-#     p.ProductID,
online_db-#     p.ProductName,
online_db-#     o.OrderID
online_db-# FROM Products p
online_db-# RIGHT JOIN Orders o ON p.ProductID = o.ProductID
online_db-# WHERE o.OrderID IS NULL;
 productid | productname | orderid
-----------+-------------+---------
(0 rows)
```

Interpretation:Shows all records from the second table and matching ones from the first. Missing matches appear as NULL.

FULL OUTER JOIN:

```
online_db=# -- FULL OUTER JOIN - Show all customers and products
online_db=# SELECT
online_db-#     c.CustomerID,
online_db-#     c.FirstName,
online_db-#     c.LastName,
online_db-#     p.ProductID,
online_db-#     p.ProductName,
online_db-#     o.OrderID
online_db-# FROM Customers c
online_db-# FULL OUTER JOIN Orders o ON c.CustomerID = o.CustomerID
online_db-# FULL OUTER JOIN Products p ON o.ProductID = p.ProductID;
 customerid | firstname | lastname | productid | productname  | orderid
------------+-----------+----------+-----------+--------------+---------
          1 | Alice     | teta     |       101 | Laptop       |    1001
          2 | Bob       | kwizera  |       103 | T-Shirt      |    1002
          1 | Alice     | teta     |       102 | Headphones   |    1003
          3 | Clara     | kaneza   |       105 | Sneakers     |    1004
          2 | Bob       | kwizera  |       104 | Coffee Maker |    1005
          5 | Eva       | uwase    |           |              |
          4 | David     | habakuki |           |              |
(7 rows)
```

Interpretation:Combines all records from both tables and keeps them in the result. Where matches do not exist, the missing side shows empty values.

SELF JOIN

```
online_db=#
online_db=# --  SELF JOIN - Compare customers in the same city
online_db=# SELECT
online_db-#     c1.CustomerID AS Customer1,
online_db-#     c1.FirstName AS FirstName1,
online_db-#     c2.CustomerID AS Customer2,
online_db-#     c2.FirstName AS FirstName2,
online_db-#     c1.City
online_db-# FROM Customers c1
online_db-# INNER JOIN Customers c2 ON c1.City = c2.City AND c1.CustomerID < c2.CustomerID;
 customer1 | firstname1 | customer2 | firstname2 |  city
-----------+------------+-----------+------------+--------
         1 | Alice      |         5 | Eva        | Kigali
         1 | Alice      |         3 | Clara      | Kigali
         3 | Clara      |         5 | Eva        | Kigali
(3 rows)
```

Interpretation: Compares records within the same table. It helps to find similarities such as customers from the same region.

AGGREGATE FUNCTION:

```
online_db=# -- Goal: See cumulative sales month by month
online_db=# WITH MonthlySales AS (
online_db(#     SELECT
online_db(#         DATE_TRUNC('month', OrderDate) AS Month,
online_db(#         SUM(Quantity * Price) AS TotalSales
online_db(#     FROM Orders o
online_db(#     JOIN Products p ON o.ProductID = p.ProductID
online_db(#     GROUP BY DATE_TRUNC('month', OrderDate)
online_db(# )
online_db-# SELECT
online_db-#     Month,
online_db-#     TotalSales,
online_db-#     SUM(TotalSales) OVER (ORDER BY Month) AS RunningTotal
online_db-# FROM MonthlySales
online_db-# ORDER BY Month;
         month          | totalsales | runningtotal
------------------------+------------+--------------
 2026-01-01 00:00:00-10 |    1530.00 |      1530.00
(1 row)
```

Interpretation: Used to calculate totals or averages while keeping individual records. It helps to track sales trends and running totals.

DISTRIBUTION FUNCTION:

```
online_db=# -- Goal: Divide customers into 4 groups based on total spending
online_db=# WITH CustomerSpending AS (
online_db(#     SELECT
online_db(#         c.CustomerID,
online_db(#         c.FirstName,
online_db(#         c.LastName,
online_db(#         COALESCE(SUM(o.Quantity * p.Price), 0) AS TotalSpent
online_db(#     FROM Customers c
online_db(#     LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
online_db(#     LEFT JOIN Products p ON o.ProductID = p.ProductID
online_db(#     GROUP BY c.CustomerID, c.FirstName, c.LastName
online_db(# )
online_db-# SELECT
online_db-#     CustomerID,
online_db-#     FirstName,
online_db-#     LastName,
online_db-#     TotalSpent,
online_db-#     NTILE(4) OVER (ORDER BY TotalSpent DESC) AS CustomerQuartile
online_db-# FROM CustomerSpending
online_db-# ORDER BY CustomerQuartile, TotalSpent DESC;
 customerid | firstname | lastname | totalspent | customerquartile
------------+-----------+----------+------------+------------------
          1 | Alice     | teta     |    1350.00 |                1
          2 | Bob       | kwizera  |     120.00 |                1
          3 | Clara     | kaneza   |      60.00 |                2
          5 | Eva       | uwase    |          0 |                3
          4 | David     | habakuki |          0 |                4
(5 rows)
```

Interpretation: Used to divide customers or products into groups based on their sales performance. It helps the business understand different customer levels and buying patterns.

RANKING FUNCTION:

```
online_db=# --  Top 5 products per category using RANK()
online_db=# -- Goal: Find best-selling products in each category
online_db=# SELECT
online_db-#     Category,
online_db-#     ProductName,
online_db-#     SUM(Quantity) AS TotalSold,  -- total quantity sold per product
online_db-#     RANK() OVER (PARTITION BY Category ORDER BY SUM(Quantity) DESC) AS ProductRank
online_db-# FROM Orders o
online_db-# JOIN Products p ON o.ProductID = p.ProductID  -- join to get product info
online_db-# GROUP BY Category, ProductName
online_db-# ORDER BY Category, ProductRank;
  category   | productname  | totalsold | productrank
-------------+--------------+-----------+-------------
 Clothes     | T-Shirt      |         2 |           1
 Clothes     | Sneakers     |         1 |           2
 Electronics | Headphones   |         1 |           1
 Electronics | Laptop       |         1 |           1
 Home        | Coffee Maker |         1 |           1
(5 rows)
```

Interpretation:Used to arrange products or customers based on sales or performance. It helps to identify top-selling products or best customers.

Navigation Function

```
online_db=# --  Month-over-month growth using LAG()
online_db=# -- Goal: Compare sales this month to previous month
online_db=# SELECT
online_db-#     Month,
online_db-#     TotalSales,
online_db-#     TotalSales - LAG(TotalSales) OVER (ORDER BY Month) AS Growth  -- calculate growth
online_db-# FROM (
online_db(#     -- calculate total sales per month first
online_db(#     SELECT
online_db(#         DATE_TRUNC('month', OrderDate) AS Month,
online_db(#         SUM(Quantity * Price) AS TotalSales
online_db(#     FROM Orders o
online_db(#     JOIN Products p ON o.ProductID = p.ProductID
online_db(#     GROUP BY DATE_TRUNC('month', OrderDate)
online_db(# ) AS MonthlySales
online_db-# ORDER BY Month;
         month          | totalsales | growth
------------------------+------------+--------
 2026-01-01 00:00:00-10 |    1530.00 |
(1 row)
```

Interpretation:Navigation functions compare values between rows such as previous or next sales records. They help to analyze changes in sales performance over time.