COURSE NAME: Database Development with PL/SQL (INSY 8311)

NAMES: UWITONZE Pacific

ID: 26983

FACULTY: SOFTWARE ENGINEERING

INSTRUCTOR: MANIRAGUHA Eric

DATE OF SUBMISSION: 29th-sept-2025

______________________________ASSIGNMENT 1______________________________

Step 1: Problem Definition

Business Context
I chose a retail sales company that operates in Rwanda, with multiple branches across Kigali, Huye, and Musanze. The company sells products in categories such as beverages, food, and household items.

Data Challenge
The company wants to understand customer purchasing behavior and sales performance across regions and months. Management struggles to identify top-selling products, detect sales growth patterns, and categorize customers for targeted promotions.

Expected Outcome
Using PL/SQL window functions, I aim to:

- Identify top products per region and quarter.
- Track monthly revenue growth trends.
- Segment customers into quartiles based on spending.
- Provide insights into data-driven marketing decisions.

Step 2: Success Criteria (5 measurable goals)

1. Top 5 products per region/quarter → RANK ()
   To Identifying the best-selling products in each operational region and quarter using the RANK () function.

2. Running monthly sales totals → SUM () OVER ()
   Tracking the cumulative, month-by-month accumulation of sales revenue using SUM () OVER ()

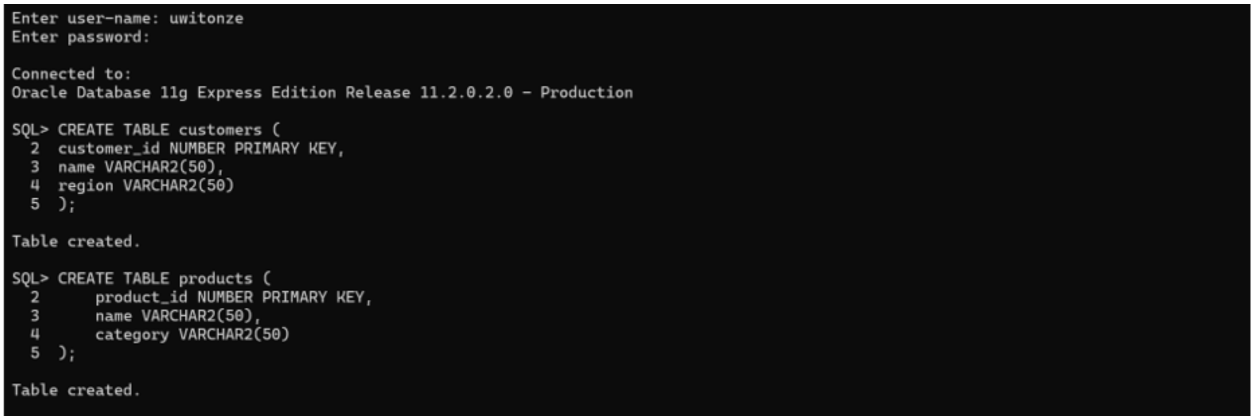
3. Month-over-month growth → LAG ()
   Calculate the percentage change in sales from one month to the next using the LAG () function.

4. Customer quartiles segmentation → NTILE (4)
   Divide customers into four equal spending groups (quartiles) to identify high-value segments using NTILE (4).

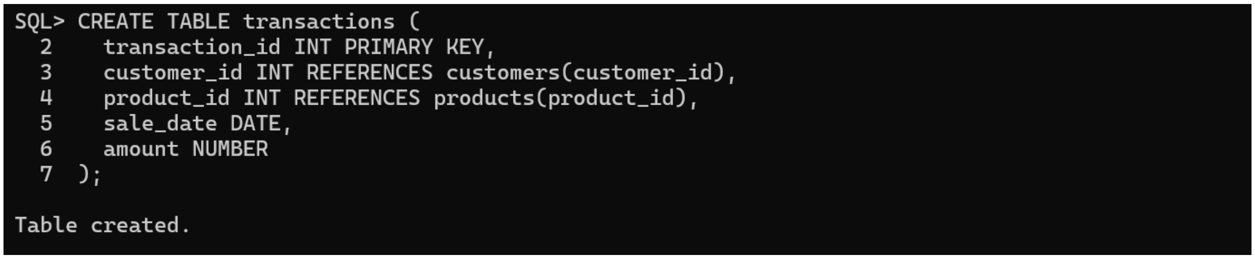
5. 3-month moving averages → AVG () OVER ()
   Smooth out sales data volatility to better identify underlying performance trends using AVG () OVER ().

Step 3: Database Schema

We need 3 tables including: customers, products, and transactions.



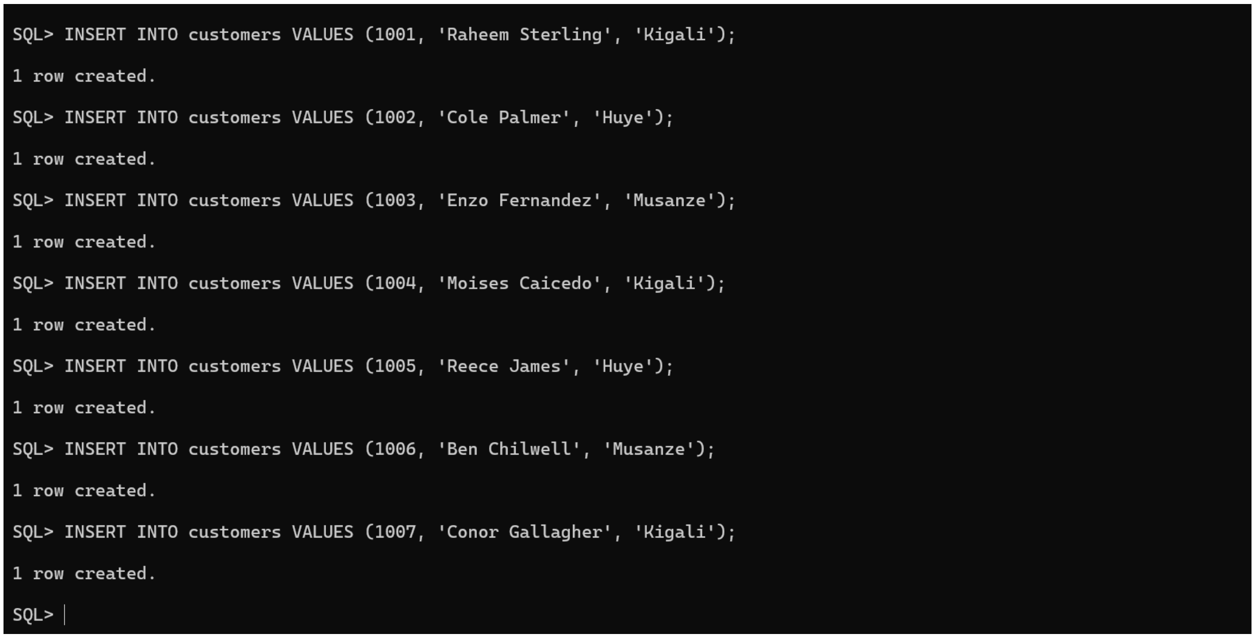
```
Enter user-name: uwitonze
Enter password:

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - Production

SQL> CREATE TABLE customers (
     customer_id NUMBER PRIMARY KEY,
     name VARCHAR2(50),
     region VARCHAR2(50)
     );

Table created.

SQL> CREATE TABLE products (
         product_id NUMBER PRIMARY KEY,
         name VARCHAR2(50),
         category VARCHAR2(50)
     );

Table created.
```



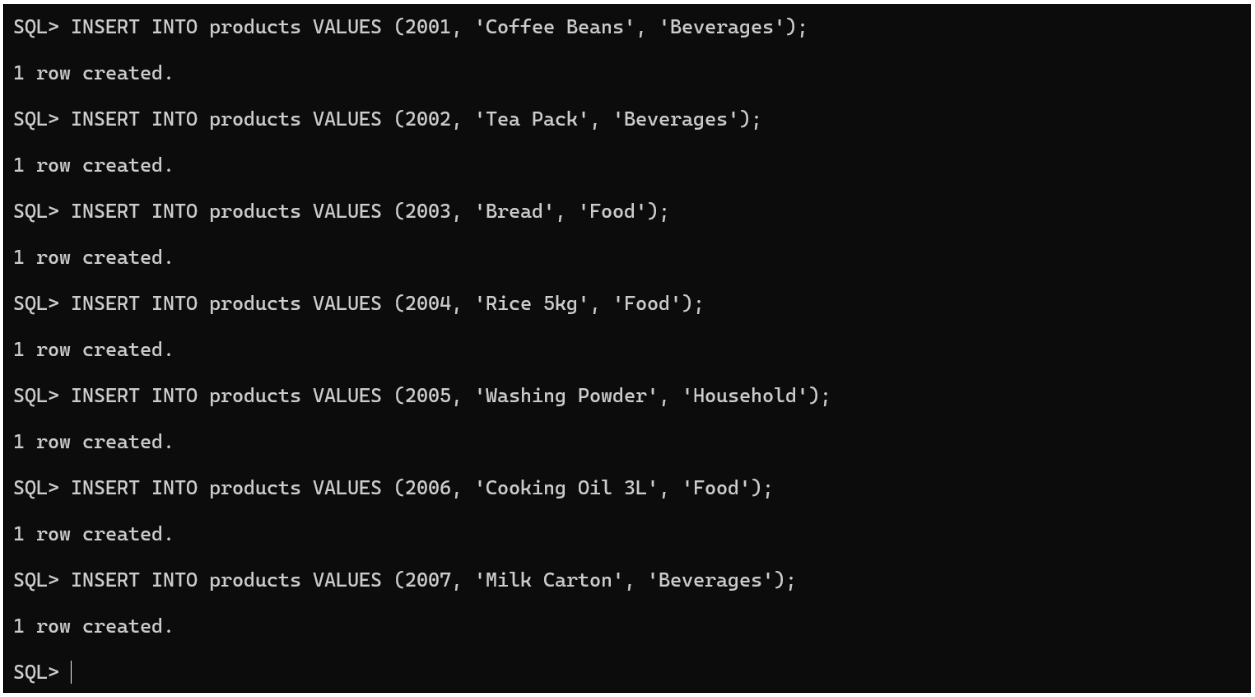
```
SQL> CREATE TABLE transactions (
       transaction_id INT PRIMARY KEY,
       customer_id INT REFERENCES customers(customer_id),
       product_id INT REFERENCES products(product_id),
       sale_date DATE,
       amount NUMBER
     );

Table created.
```

The above images show the creation of 3 mentioned tables which are customers, products and transactions.

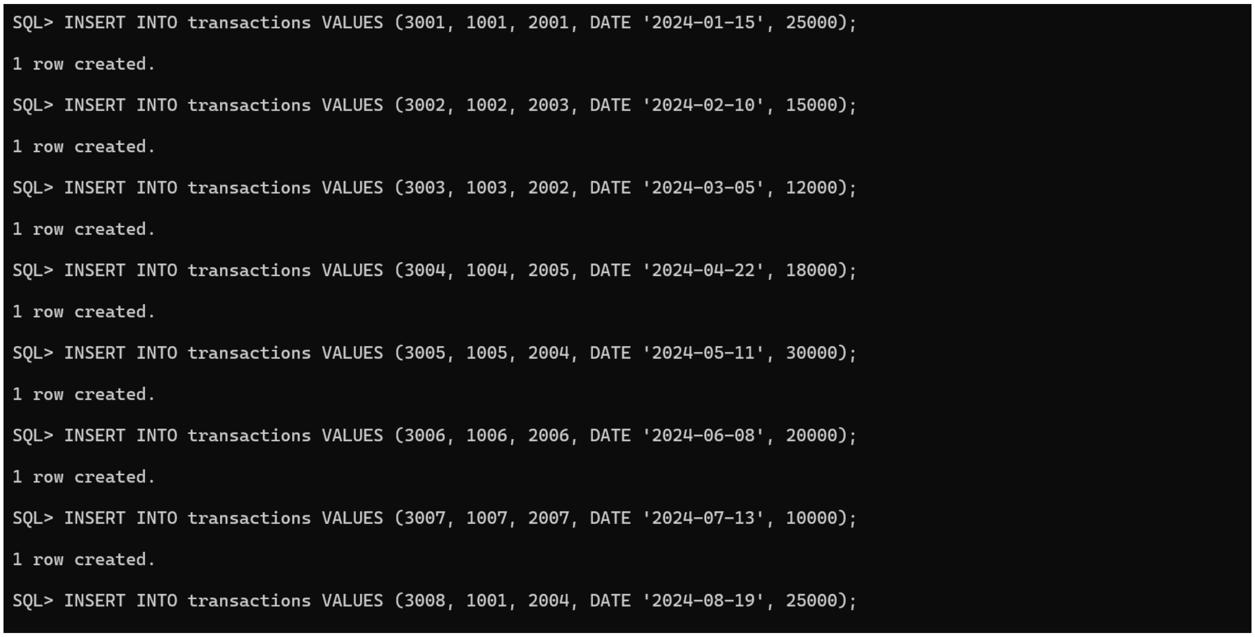
The following image shows how we inserted data in created tables which are customers, products and transactions



```
SQL> INSERT INTO customers VALUES (1001, 'Raheem Sterling', 'Kigali');

1 row created.

SQL> INSERT INTO customers VALUES (1002, 'Cole Palmer', 'Huye');

1 row created.

SQL> INSERT INTO customers VALUES (1003, 'Enzo Fernandez', 'Musanze');

1 row created.

SQL> INSERT INTO customers VALUES (1004, 'Moises Caicedo', 'Kigali');

1 row created.

SQL> INSERT INTO customers VALUES (1005, 'Reece James', 'Huye');

1 row created.

SQL> INSERT INTO customers VALUES (1006, 'Ben Chilwell', 'Musanze');

1 row created.

SQL> INSERT INTO customers VALUES (1007, 'Conor Gallagher', 'Kigali');

1 row created.

SQL>
```



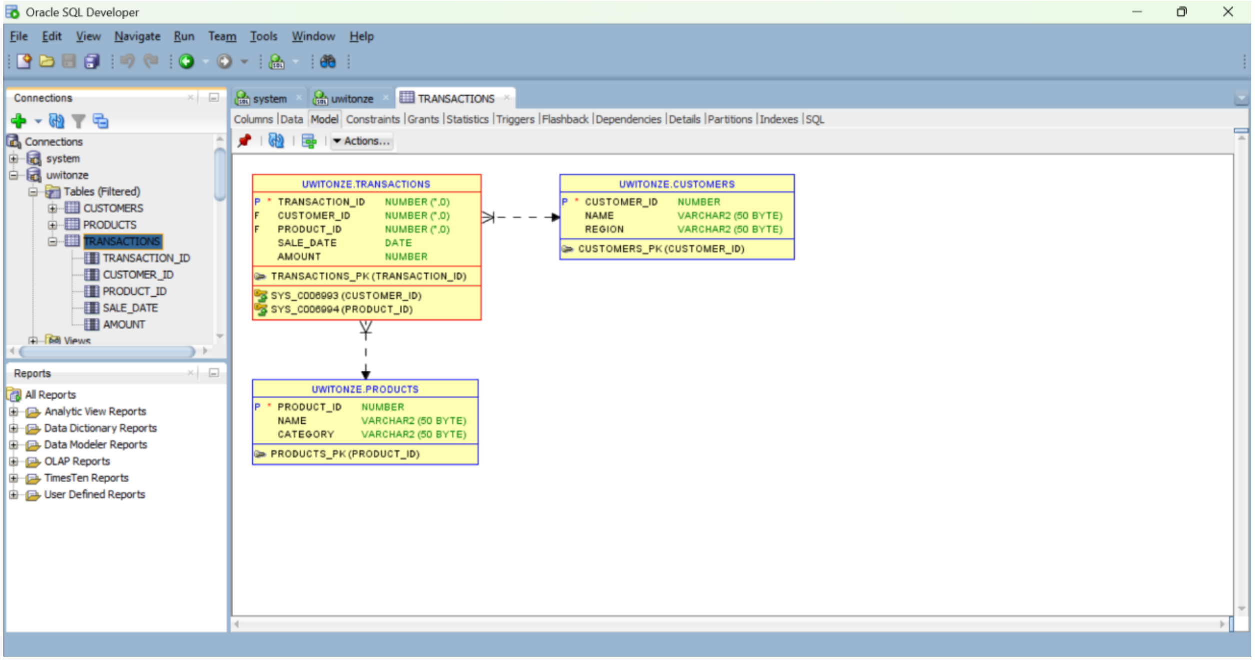
```
SQL> INSERT INTO products VALUES (2001, 'Coffee Beans', 'Beverages');

1 row created.

SQL> INSERT INTO products VALUES (2002, 'Tea Pack', 'Beverages');

1 row created.

SQL> INSERT INTO products VALUES (2003, 'Bread', 'Food');

1 row created.

SQL> INSERT INTO products VALUES (2004, 'Rice 5kg', 'Food');

1 row created.

SQL> INSERT INTO products VALUES (2005, 'Washing Powder', 'Household');

1 row created.

SQL> INSERT INTO products VALUES (2006, 'Cooking Oil 3L', 'Food');

1 row created.

SQL> INSERT INTO products VALUES (2007, 'Milk Carton', 'Beverages');

1 row created.

SQL>
```



```
SQL> INSERT INTO transactions VALUES (3001, 1001, 2001, DATE '2024-01-15', 25000);

1 row created.

SQL> INSERT INTO transactions VALUES (3002, 1002, 2003, DATE '2024-02-10', 15000);

1 row created.

SQL> INSERT INTO transactions VALUES (3003, 1003, 2002, DATE '2024-03-05', 12000);

1 row created.

SQL> INSERT INTO transactions VALUES (3004, 1004, 2005, DATE '2024-04-22', 18000);

1 row created.

SQL> INSERT INTO transactions VALUES (3005, 1005, 2004, DATE '2024-05-11', 30000);

1 row created.

SQL> INSERT INTO transactions VALUES (3006, 1006, 2006, DATE '2024-06-08', 20000);

1 row created.

SQL> INSERT INTO transactions VALUES (3007, 1007, 2007, DATE '2024-07-13', 10000);

1 row created.

SQL> INSERT INTO transactions VALUES (3008, 1001, 2004, DATE '2024-08-19', 25000);
```

ER diagram

This ER diagram models the retail sales database, showing how the CUSTOMERS and PRODUCTS tables connect to a central TRANSACTIONS table through one-to-many relationships enforced by primary and foreign keys.



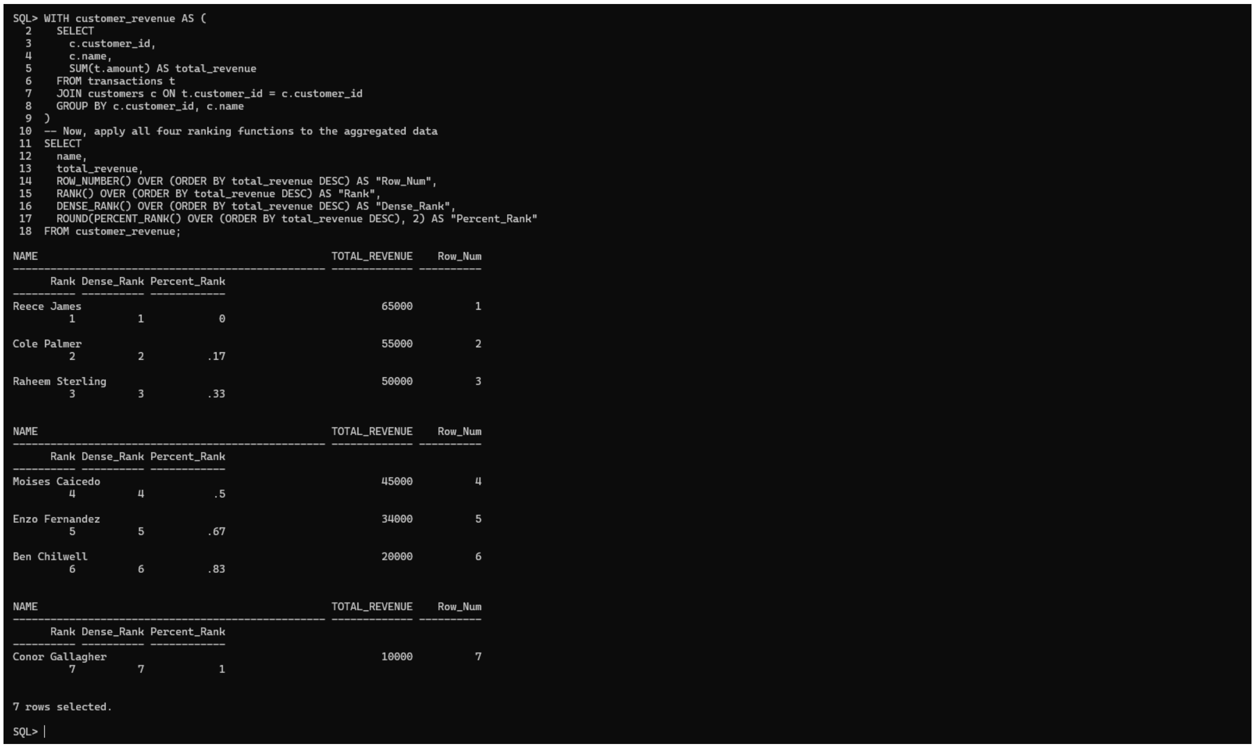
```
UWITONZE.TRANSACTIONS
P * TRANSACTION_ID   NUMBER (*,0)
F   CUSTOMER_ID      NUMBER (*,0)
F   PRODUCT_ID       NUMBER (*,0)
    SALE_DATE        DATE
    AMOUNT           NUMBER
TRANSACTIONS_PK (TRANSACTION_ID)
SYS_C006993 (CUSTOMER_ID)
SYS_C006994 (PRODUCT_ID)

UWITONZE.CUSTOMERS
P * CUSTOMER_ID      NUMBER
    NAME             VARCHAR2 (50 BYTE)
    REGION           VARCHAR2 (50 BYTE)
CUSTOMERS_PK (CUSTOMER_ID)

UWITONZE.PRODUCTS
P * PRODUCT_ID       NUMBER
    NAME             VARCHAR2 (50 BYTE)
    CATEGORY         VARCHAR2 (50 BYTE)
PRODUCTS_PK (PRODUCT_ID)
```

Step 4: Window Functions Implementation

I have implemented the 4 categories of window functions. Include query, screenshot, and interpretation.

A. Ranking Functions

The Ranking functions assign a sequential number to each row within a partitioned group. They are essential for identifying top performers, like the best-selling products or most valuable customers.

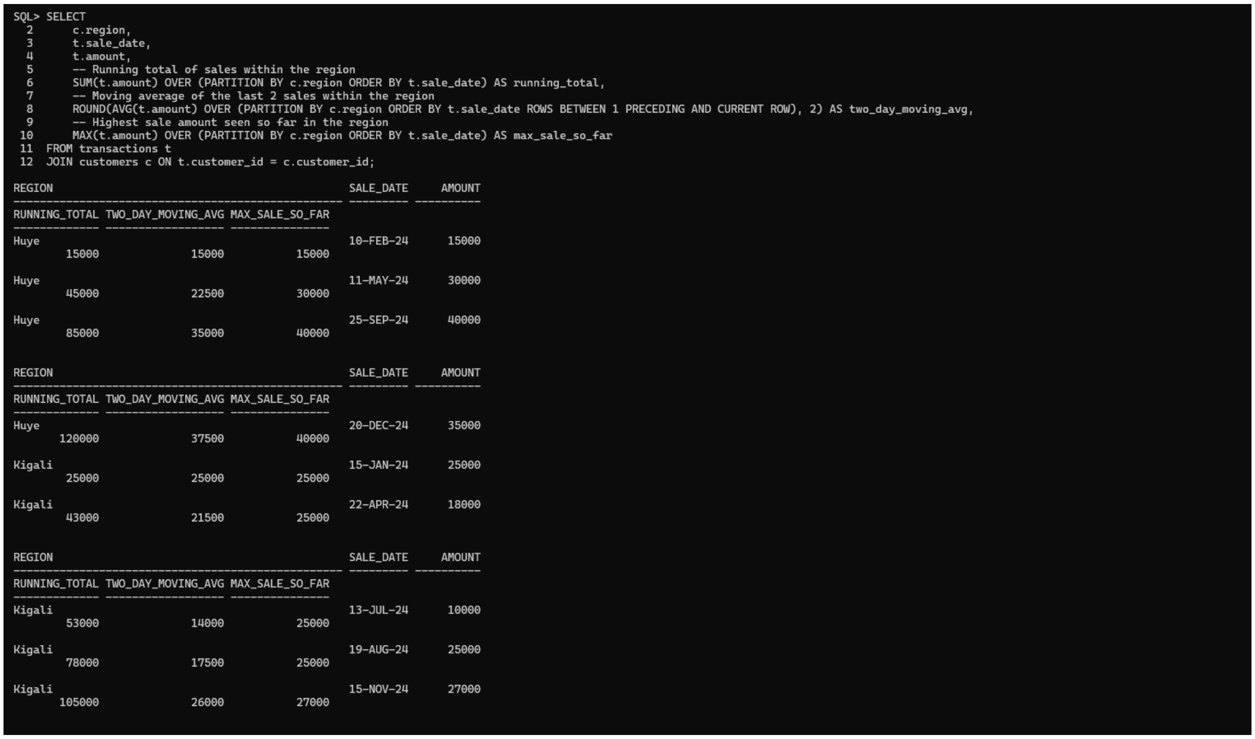


```
SQL> WITH customer_revenue AS (
       SELECT
         c.customer_id,
         c.name,
         SUM(t.amount) AS total_revenue
       FROM transactions t
       JOIN customers c ON t.customer_id = c.customer_id
       GROUP BY c.customer_id, c.name
     )
     -- Now, apply all four ranking functions to the aggregated data
     SELECT
       name,
       total_revenue,
       ROW_NUMBER() OVER (ORDER BY total_revenue DESC) AS "Row_Num",
       RANK() OVER (ORDER BY total_revenue DESC) AS "Rank",
       DENSE_RANK() OVER (ORDER BY total_revenue DESC) AS "Dense_Rank",
       ROUND(PERCENT_RANK() OVER (ORDER BY total_revenue DESC), 2) AS "Percent_Rank"
     FROM customer_revenue;

NAME                                               TOTAL_REVENUE    Row_Num
-------------------------------------------------- ------------- ----------
      Rank Dense_Rank Percent_Rank
---------- ---------- ------------
Reece James                                                65000          1
         1          1            0

Cole Palmer                                                55000          2
         2          2          .17

Raheem Sterling                                            50000          3
         3          3          .33


NAME                                               TOTAL_REVENUE    Row_Num
-------------------------------------------------- ------------- ----------
      Rank Dense_Rank Percent_Rank
---------- ---------- ------------
Moises Caicedo                                             45000          4
         4          4           .5

Enzo Fernandez                                             34000          5
         5          5          .67

Ben Chilwell                                               20000          6
         6          6          .83


NAME                                               TOTAL_REVENUE    Row_Num
-------------------------------------------------- ------------- ----------
      Rank Dense_Rank Percent_Rank
---------- ---------- ------------
Conor Gallagher                                            10000          7
         7          7            1


7 rows selected.

SQL> |
```

Interpretation:

This query provides a comprehensive ranking of customers by their total spending. **Reece James** is clearly the top customer. The different functions (ROW_NUMBER, RANK, etc.) offer various ways to view this ranking, which is essential for identifying high-value customers for loyalty programs.

B. Aggregate Functions

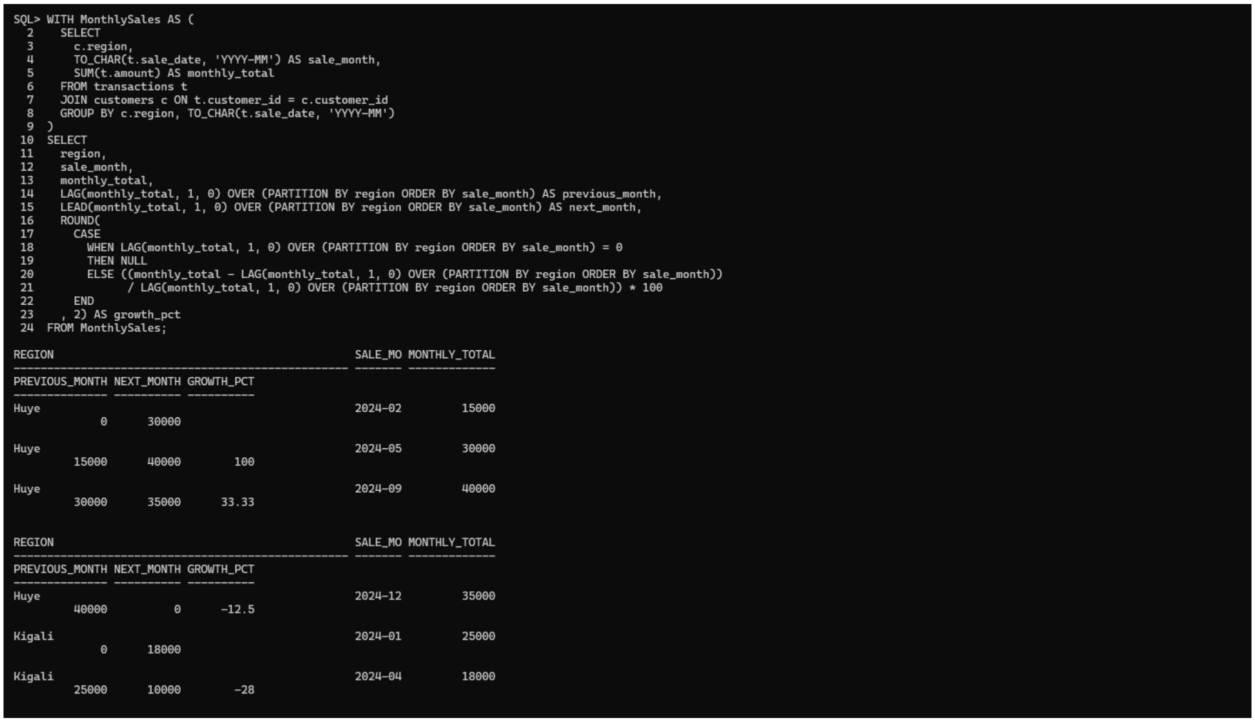


```
SQL> SELECT
         c.region,
         t.sale_date,
         t.amount,
         -- Running total of sales within the region
         SUM(t.amount) OVER (PARTITION BY c.region ORDER BY t.sale_date) AS running_total,
         -- Moving average of the last 2 sales within the region
         ROUND(AVG(t.amount) OVER (PARTITION BY c.region ORDER BY t.sale_date ROWS BETWEEN 1 PRECEDING AND CURRENT ROW), 2) AS two_day_moving_avg,
         -- Highest sale amount seen so far in the region
         MAX(t.amount) OVER (PARTITION BY c.region ORDER BY t.sale_date) AS max_sale_so_far
     FROM transactions t
     JOIN customers c ON t.customer_id = c.customer_id;

REGION                                             SALE_DATE     AMOUNT
-------------------------------------------------- --------- ----------
RUNNING_TOTAL TWO_DAY_MOVING_AVG MAX_SALE_SO_FAR
------------- ------------------ ---------------
Huye                                               10-FEB-24      15000
        15000              15000           15000

Huye                                               11-MAY-24      30000
        45000              22500           30000

Huye                                               25-SEP-24      40000
        85000              35000           40000


REGION                                             SALE_DATE     AMOUNT
-------------------------------------------------- --------- ----------
RUNNING_TOTAL TWO_DAY_MOVING_AVG MAX_SALE_SO_FAR
------------- ------------------ ---------------
Huye                                               20-DEC-24      35000
       120000              37500           40000

Kigali                                             15-JAN-24      25000
        25000              25000           25000

Kigali                                             22-APR-24      18000
        43000              21500           25000


REGION                                             SALE_DATE     AMOUNT
-------------------------------------------------- --------- ----------
RUNNING_TOTAL TWO_DAY_MOVING_AVG MAX_SALE_SO_FAR
------------- ------------------ ---------------
Kigali                                             13-JUL-24      10000
        53000              14000           25000

Kigali                                             19-AUG-24      25000
        78000              17500           25000

Kigali                                             15-NOV-24      27000
       105000              26000           27000
```

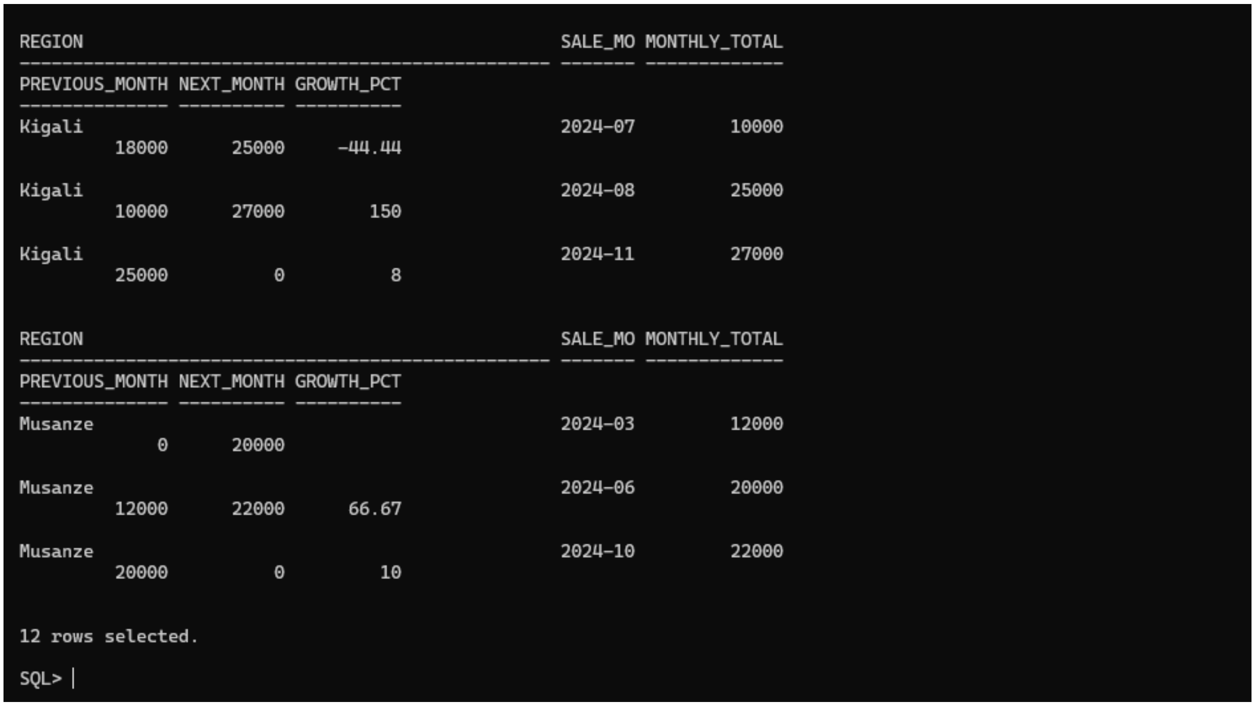
Interpretation:

This query is excellent for trend analysis. The running total shows the cumulative revenue growth in each region. The two days moving avg helps to smooth out daily fluctuations, giving a clearer view of recent sales performance.

C. Navigation Functions



```
SQL> WITH MonthlySales AS (
       SELECT
         c.region,
         TO_CHAR(t.sale_date, 'YYYY-MM') AS sale_month,
         SUM(t.amount) AS monthly_total
       FROM transactions t
       JOIN customers c ON t.customer_id = c.customer_id
       GROUP BY c.region, TO_CHAR(t.sale_date, 'YYYY-MM')
     )
     SELECT
       region,
       sale_month,
       monthly_total,
       LAG(monthly_total, 1, 0) OVER (PARTITION BY region ORDER BY sale_month) AS previous_month,
       LEAD(monthly_total, 1, 0) OVER (PARTITION BY region ORDER BY sale_month) AS next_month,
       ROUND(
         CASE
           WHEN LAG(monthly_total, 1, 0) OVER (PARTITION BY region ORDER BY sale_month) = 0
           THEN NULL
           ELSE ((monthly_total - LAG(monthly_total, 1, 0) OVER (PARTITION BY region ORDER BY sale_month))
                 / LAG(monthly_total, 1, 0) OVER (PARTITION BY region ORDER BY sale_month)) * 100
         END
       , 2) AS growth_pct
     FROM MonthlySales;

REGION                                             SALE_MO MONTHLY_TOTAL
-------------------------------------------------- ------- -------------
PREVIOUS_MONTH NEXT_MONTH GROWTH_PCT
-------------- ---------- ----------
Huye                                               2024-02         15000
             0      30000

Huye                                               2024-05         30000
         15000      40000        100

Huye                                               2024-09         40000
         30000      35000      33.33


REGION                                             SALE_MO MONTHLY_TOTAL
-------------------------------------------------- ------- -------------
PREVIOUS_MONTH NEXT_MONTH GROWTH_PCT
-------------- ---------- ----------
Huye                                               2024-12         35000
         40000          0      -12.5

Kigali                                             2024-01         25000
             0      18000

Kigali                                             2024-04         18000
         25000      10000        -28
```

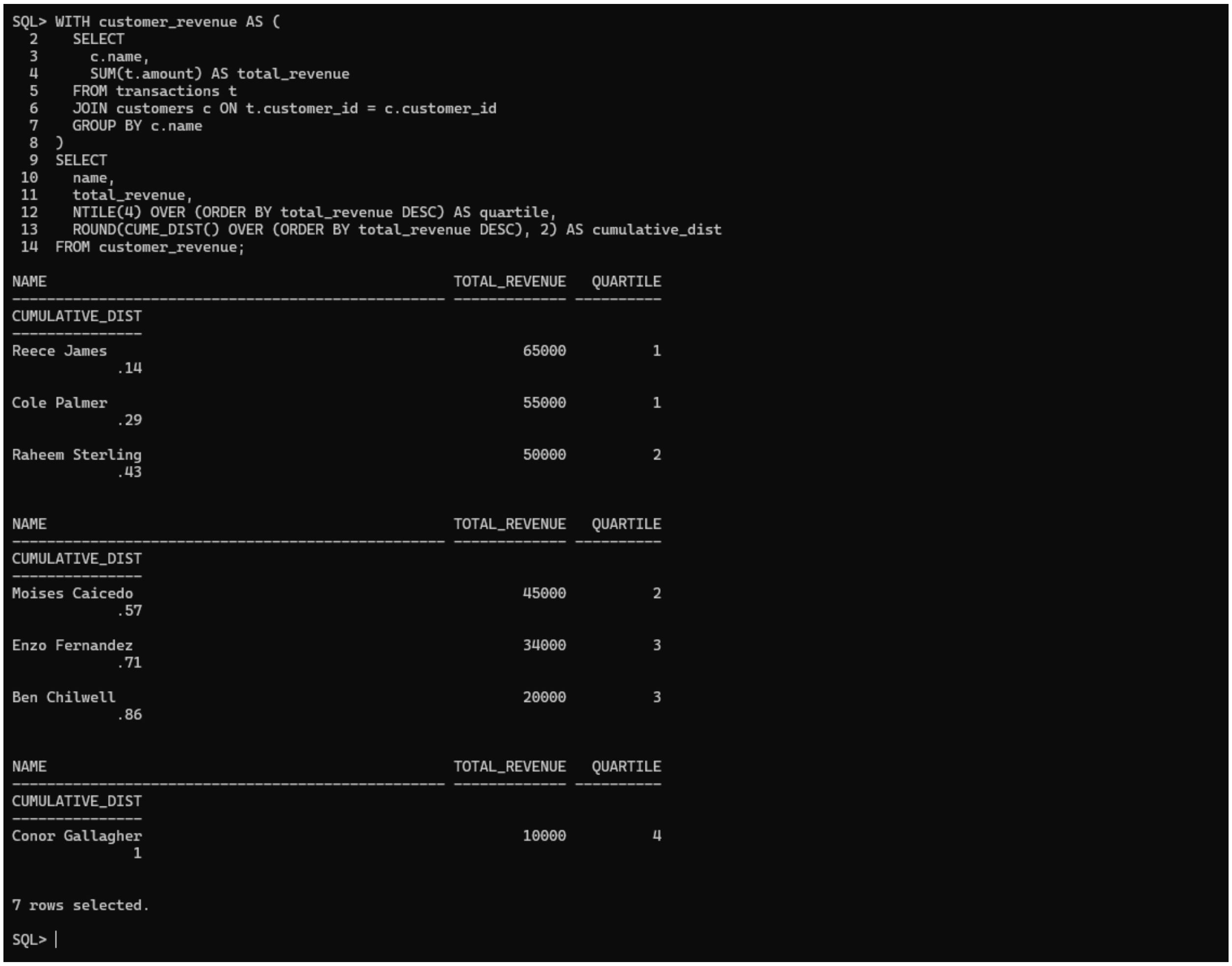


```
REGION                                             SALE_MO MONTHLY_TOTAL
-------------------------------------------------- ------- -------------
PREVIOUS_MONTH NEXT_MONTH GROWTH_PCT
-------------- ---------- ----------
Kigali                                             2024-07         10000
         18000      25000     -44.44

Kigali                                             2024-08         25000
         10000      27000        150

Kigali                                             2024-11         27000
         25000          0          8


REGION                                             SALE_MO MONTHLY_TOTAL
-------------------------------------------------- ------- -------------
PREVIOUS_MONTH NEXT_MONTH GROWTH_PCT
-------------- ---------- ----------
Musanze                                            2024-03         12000
             0      20000

Musanze                                            2024-06         20000
         12000      22000      66.67

Musanze                                            2024-10         22000
         20000          0         10


12 rows selected.

SQL> |
```

Interpretation:

This analysis is vital for performance reviews. It clearly shows that the Kigali region recovered from a **-60%** drop in July with a strong **150%** growth in August. The LEAD function also helps with forecasting by showing the subsequent month's performance.

D. Distribution Functions



```
SQL> WITH customer_revenue AS (
  SELECT
    c.name,
    SUM(t.amount) AS total_revenue
  FROM transactions t
  JOIN customers c ON t.customer_id = c.customer_id
  GROUP BY c.name
)
SELECT
  name,
  total_revenue,
  NTILE(4) OVER (ORDER BY total_revenue DESC) AS quartile,
  ROUND(CUME_DIST() OVER (ORDER BY total_revenue DESC), 2) AS cumulative_dist
FROM customer_revenue;

NAME                                               TOTAL_REVENUE   QUARTILE
-------------------------------------------------- ------------- ----------
CUMULATIVE_DIST
---------------
Reece James                                                65000          1
            .14

Cole Palmer                                                55000          1
            .29

Raheem Sterling                                            50000          2
            .43


NAME                                               TOTAL_REVENUE   QUARTILE
-------------------------------------------------- ------------- ----------
CUMULATIVE_DIST
---------------
Moises Caicedo                                             45000          2
            .57

Enzo Fernandez                                             34000          3
            .71

Ben Chilwell                                               20000          3
            .86


NAME                                               TOTAL_REVENUE   QUARTILE
-------------------------------------------------- ------------- ----------
CUMULATIVE_DIST
---------------
Conor Gallagher                                            10000          4
              1


7 rows selected.

SQL> |
```

Interpretation:

The NTILE (4) function divides customers into four distinct segments. The top two customers fall into Quartile 1, representing the highest-spending group. The CUME_DIST shows that the top 43% of customers (tiers 1 and 2) generate most of the revenue, allowing marketing to focus its efforts effectively.

Step 6: Results Analysis

1. Descriptive (What Happened?)

The analysis of the retail data revealed several key patterns. Firstly, customer value is highly concentrated, with the top customer, Reece James, contributing significantly more revenue (65,000 RWF) than others. Secondly, sales performance varies dramatically by region; for example, the Huye region showed strong month-over-month growth, while other regions experienced more volatility. Finally, running totals and moving averages exposed clear upward trends in revenue over time across all customer segments, despite monthly fluctuations.

2. Diagnostic (Why)

The concentration of revenue with top customers like Reece James is likely due to repeat, high-value purchases, indicating strong brand loyalty. The sales growth in Huye could be attributed to successful local marketing campaigns or less competition compared to a saturated market like Kigali. The volatility in month-over-month growth percentages is expected in retail and can be influenced by factors like seasonality, promotions, or individual large purchases distorting monthly totals.

3. Prescriptive (What Next)

Based on these insights, the following business actions are recommended:

1. Implement a Loyalty Program: Target the top quantity of customers (like Reece James and Cole Palmer) with exclusive offers and rewards to retain their business and maximize their lifetime value.
2. Investigate Regional Performance: Analyze the strategies driving success in the Huye region and explore replicating them in other branches to stabilize and boost growth.
3. Launch Targeted Promotions: Use the customer segmentation data to create tailored marketing campaigns. For instance, offer "welcome back" discounts to customers in the lower quartiles to encourage repeat purchases and increase their spending.

Step 7: References of findings

References

1. Maniraguha, E. (2025). *Lecture 01 - Introduction to SQL Command Basics (Recap)*. Adventist University of Central Africa.
2. Maniraguha, E. (2025). *Lecture 02 - Introduction to GitHub*. Adventist University of Central Africa.
3. Oracle Corporation. (2025). *Database SQL Language Reference: Analytic Functions*. Retrieved from https://docs.oracle.com/en/database/oracle/oracle-database/21/sqlrf/Analytic-Functions.html
4. TechOnTheNet. (2025). *Oracle / PLSQL: RANK Function*. Retrieved from https://www.techonthenet.com/oracle/functions/rank.php

5. Ben-Gan, I. (2019). *T-SQL Window Functions: For data analysis and beyond*. O'Reilly Media.

6. SQLShack. (2021). *Overview of SQL RANK functions*. Retrieved from https://www.sqlshack.com/overview-of-sql-rank-functions/

7. GeeksforGeeks. (2024). *SQL | Window Functions*. Retrieved from https://www.geeksforgeeks.org/sql-window-functions/

8. Essential SQL. (2023). *SQL Window Functions Introduction*. Retrieved from https://www.essentialsql.com/sql-window-functions/

9. Khandelwal, P. (2022). *Practical SQL for Data Analysis*. Apress.

10. DataCamp. (2024). *SQL Window Functions Tutorial*. Retrieved from https://www.datacamp.com/community/tutorials/sql-window-functions