2025

Assignment I:PL/SQL window function

DONE BY ASIMWE GAELLE
27397

Table of Contents

Assignment Ⅰ: PL/SQL Window Functions

1. Problem Definition

Business Context

Company type: Ecommerce Platform

Department: sales and customer insights

Industry: Online retail (electronics, fashion, beverages)

Data Challenge

The company wants to better understand customer behavior and product performance across different regions and time periods. Managers struggles to track top-selling products, customer purchasing frequency, and revenue growth trends.

Expected outcome

Gain insights into:

- The best performing products per region/ quarter
- Customer spending patterns and segmentation
- Growth trends for sales months over months
- Smarter marketing and inventory planning

2. Database Schema

2.1. Schema Overview

The database consists of three related tables designed to manage product information and customer transactions. The schema follows a relational model with appropriate foreign key constraints to maintain data integrity.

2.2. Table definitions

1. Customers Table

Stores customers demographic information.

```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    region VARCHAR(50)
);
```

2. Products Table

Contains product category

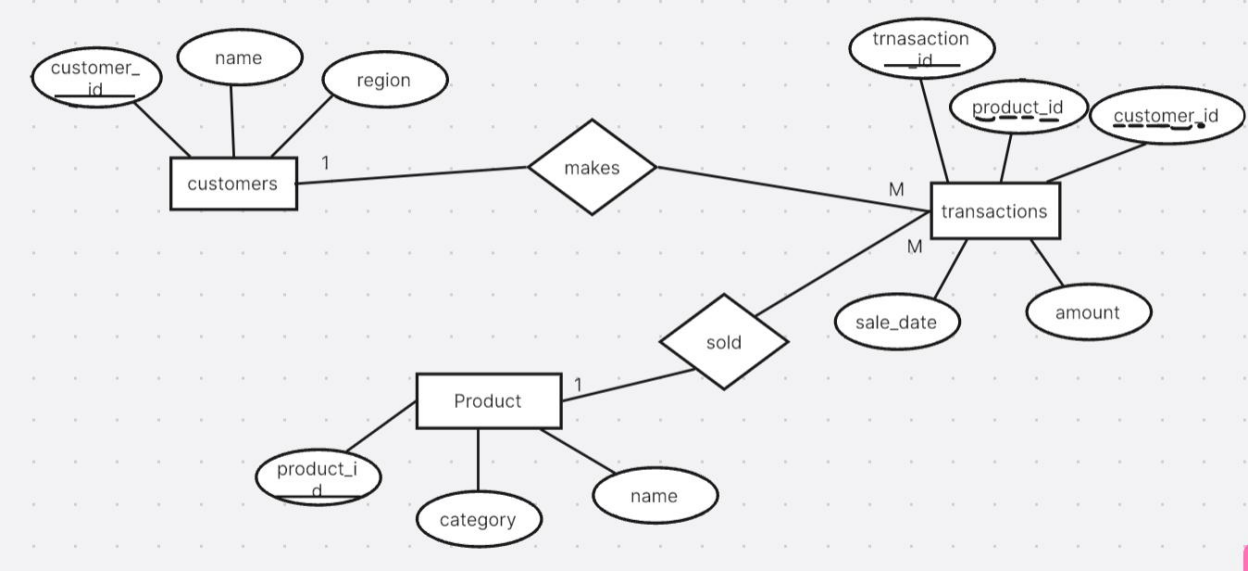
```
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    category VARCHAR(50)
);
```

3. Transactions Table

Records sales transactions with references to customers and products.

```
CREATE TABLE transactions (
    transaction_id INT PRIMARY KEY,
    customer_id INT,
    product_id INT,
    sale_date DATE NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

2.3. Entity-Relationship Diagram



3. Sample Data

customers

	customer_id [PK] integer	name character varying (100)	region character varying (50)
1	1001	John Doe	Kigali
2	1002	Alice Umutoni	Huye
3	1003	David Nshimiyimana	Musanze
4	1004	Sarah Uwera	Kigali
5	1005	Eric Mugisha	Butare

Products

	product_id [PK] integer	name character varying (100)	category character varying (50)
1	2001	Laptop	Electronics
2	2002	Coffee Beans	Beverages
3	2003	Headphones	Electronics
4	2004	Sneakers	Fashion
5	2005	Rice Bag 25kg	Groceries

Transactions

	transaction_id [PK] integer	customer_id integer	product_id integer	sale_date date	amount numeric (12,2)
1	3001	1001	2001	2025-01-15	75000.00
2	3002	1002	2002	2025-01-20	20000.00
3	3003	1003	2003	2025-02-05	30000.00
4	3004	1001	2005	2025-02-15	18000.00
5	3005	1004	2004	2025-03-01	50000.00
6	3006	1005	2002	2025-03-10	25000.00
7	3007	1002	2001	2025-03-15	76000.00
8	3008	1003	2005	2025-04-02	22000.00
9	3009	1004	2003	2025-04-15	32000.00
10	3010	1005	2004	2025-04-20	48000.00

4. Relationships

4.1. Cardinality

- Customers to transactions: one to many(1:M)
- Products to transactions: one to many(1:M)

4.2. Business rules

- A customers can have one or many transactions
- A product can appear in one or many transactions
- Every transactions must be associated with one customer and one product

5. Analytical capabilities

This section demonstrates the practical implementation of the database schema designed above, the relational structure enables advanced analytical operations that address the business intelligence required outlined in the assignment specifications.

Based on the assignment objectives, this database supports:

- Sales performance monitoring across regions and product categories
- Customers behavior analysis for segmentation and targeting
- Tracking of products performance to determine trends and opportunities
- Analysis of sales patterns and growth metrics

The analytical queries presented below respond to the assignment's requirement to demonstrate the database's utility beyond basic data storage, and how the schema facilitates data-information business decision-making through data-based information

1. Ranking (Top products by region)

Ranking window functions are used to assign an order or position to each row within the dataset according to set of criteria. They are utilized to enable an organization to find out relative standing of records, such as determining the top-selling products in each region or most valuable customers. ranking rows in a dataset helps an organization easily point out best and worst performance and help in decision-making during evaluation of performances.

```
SELECT region, p.product_id, p.name, SUM(t.amount) AS total_sales,
       RANK() OVER (PARTITION BY region ORDER BY SUM(t.amount) DESC) AS rank_num
FROM customers c
JOIN transactions t ON c.customer_id = t.customer_id
JOIN products p ON t.product_id = p.product_id
GROUP BY region, p.product_id, p.name limit 5;
```

	region character varying (50)	product_id integer	name character varying (100)	total_sales numeric	rank_pos bigint
1	Butare	2004	Sneakers	48000.00	1
2	Butare	2002	Coffee Beans	25000.00	2
3	Huye	2001	Laptop	76000.00	1
4	Huye	2002	Coffee Beans	20000.00	2
5	Kigali	2001	Laptop	75000.00	1

Interpretation:

This query products in each region by totals sales. The screenshot shows that in butare, sneakers are ranked #1 while coffee beans ranked #2. This helps management identify which products to promote in each region.

2. Aggregate (Running monthly sales totals)

Aggregate window functions enhance standard aggregates by enabling cumulative or comparative calculations across sets of rows while preserving individual row details. They are mainly used for running

totals, averages, minimums, and maximums overtime, helping to track trends, measure performance, and identify peaks or decline without losing underlying record information.

```
SELECT TO_CHAR(sale_date, 'YYYY-MM') AS month,
       SUM(amount) AS monthly_total,
       SUM(SUM(amount)) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM')) AS running_total
FROM transactions
GROUP BY TO_CHAR(sale_date, 'YYYY-MM')
ORDER BY month;
```

	month text	monthly_total numeric	running_total numeric
1	2025-01	95000.00	95000.00
2	2025-02	48000.00	143000.00
3	2025-03	151000.00	294000.00
4	2025-04	102000.00	396000.00

Interpretation:

The screenshot shows sales growing steadily from January to April. The **running_total** column accumulates monthly totals, making it easy to track overall progress.

3. Navigation (Monthly over month growth)

```
SELECT month, total_sales,
       LAG(total_sales) OVER (ORDER BY month) AS prev_month,
       ROUND(((total_sales - LAG(total_sales) OVER (ORDER BY month))
             / LAG(total_sales) OVER (ORDER BY month)) * 100, 2) AS growth_percent
FROM (
   SELECT TO_CHAR(sale_date, 'YYYY-MM') AS month,
          SUM(amount) AS total_sales
   FROM transactions
   GROUP BY TO_CHAR(sale_date, 'YYYY-MM')
);
```

	month text	total_sales numeric	prev_month numeric	growth_percent numeric
1	2025-01	95000.00	[null]	[null]
2	2025-02	48000.00	95000.00	-49.47
3	2025-03	151000.00	48000.00	214.58
4	2025-04	102000.00	151000.00	-32.45

Interpretation:

The query highlights percentage changes between months. From February to march, sales increased sharply, showing growth trend. This reflects demand variations, likely linked to promotion or seasonal events.

4. Distribution (Customer Quartiles)

```
SELECT customer_id, SUM(amount) AS total_spent,
       NTILE(4) OVER (ORDER BY SUM(amount) DESC) AS spending_quartile
FROM transactions
GROUP BY customer_id;
```

	customer_id integer	total_spent numeric	spending_quartile integer
1	1002	96000.00	1
2	1001	93000.00	1
3	1004	82000.00	2
4	1005	73000.00	3
5	1003	52000.00	4

Interpretation:

This analysis serves as a classification measure, grouping customers into four quartiles based on their spending. The output shows that quartile 1 contains the biggest spenders, who should be targeted with loyalty rewards.

5. Moving average (3month sales)

```
SELECT TO_CHAR(sale_date, 'YYYY-MM') AS month,
       AVG(SUM(amount)) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM')
                              ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS moving_avg
FROM transactions
GROUP BY TO_CHAR(sale_date, 'YYYY-MM');
```

	month text	moving_avg numeric
1	2025-01	95000.000000000000
2	2025-02	71500.000000000000
3	2025-03	98000.000000000000
4	2025-04	100333.333333333333

Interpretation:

This shows that the 3 month moving average smoothing out the fluctuations in sales. This allows company to view long term trends while overlooking short term spike.

Conclusion:

This assignment demonstrates how PL/SQL window functions turn e-commerce data into decision making information. By using ranking, aggregate, and moving averages, the analysis identifies top performing products, tracks sales growth, and segments customers by spending. These insights enable data driven decisions for targeted marketing, inventory planning, and strategic growth, moving beyond simple reporting to true business intelligence.

References

Oracle documentation – windows functions

https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Analytic-Functions.html

PostgreSQL documentation – window functions

https://www.postgresql.org/docs/current/tutorial-window.html

w3schools – MySQL | LEAD() and LAG function

https://www.geeksforgeeks.org/sql/mysql-lead-and-lag-function/

GeeksforGeeks – SQL windows functions explained with examples

https://www.geeksforgeeks.org/sql/window-functions-in-sql/

Mode Analysis – SQL tutorial: window functions

https://mode.com/sql-tutorial/sql-window-functions

Stack overflow discussions – when to choose rank() over dense_rank() or row_number()

https://stackoverflow.com/questions/64420584/when-to-choose-rank-over-dense-rank-or-row-number

oracle liveSQL – analytic functions

https://livesql.oracle.com/next/library/tutorials/analytic-functions-databases-for-developers-lfiKpp

MySQL 8.0 documentation – window functions

https://dev.mysql.com/doc/refman/8.0/en/window-functions.html