PL/SQL Window Functions Mastery Project

Akariza GASANA Leslie

9/27/2025

Problem Definition

A large financial services company, finance department operating in the auditing and compliance industry' s finance team manages thousands of digital transaction records but struggles to analyze them well. Managers find it hard to track period to period variations and detect reoccurring errors leading to long audit processes and overlooked discrepancies.

By applying PL/SQL window functions, the company will identify its highest spending categories, monitor monthly expense growth and segment transactions by risk level. These insights will help shorten the audit process, improve accuracy and help in better financial decision making.

Success Criteria

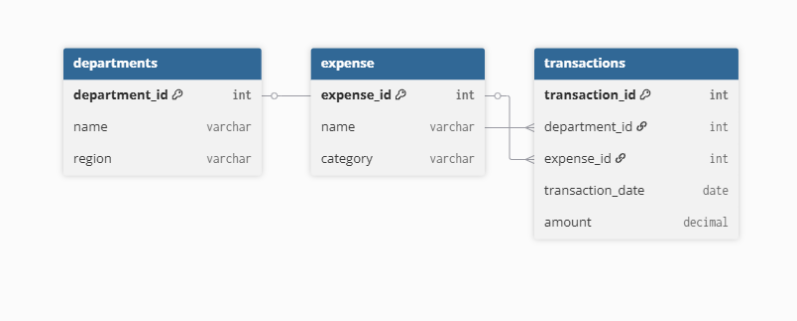
1. **Top spending categories per month**
   RANK( ) is used to identify the highest spending categories each month to prioritize auditing focus.
2. **Running monthly expense totals**
   SUM( ) OVER( ) is used to track monthly expenses to monitor trends and detect anomalies.
3. **Monthly expense growth**
   LAG( )/LEAD( ) is used to calculate changes in spending compared to previous months to spot unusual increases or decreases.
4. **Transaction risk quartiles**
   NTILE( ) is used to divide transactions into quartiles based on risk scores, highlighting high risk transactions for review.
5. **3 month moving average of expenses**
   AVG( ) OVER( ) is used to smooth out fluctuations in monthly expenses to see the underlying trends more clearly and identify long term trends in departmental spending.

Database Schema

Table	Purpose	Key Columns	Example Row
departments	Departments responsible for spending	department_id (PK), name, region	1001, Finance Dept, Kigali
expense	Types of spending	expense_id (PK), name, category	2001, Travel, Transportation

transactions	Individual financial transactions	transaction_id (PK), department_id (FK), expense_id (FK), transaction_date, amount	3001, 1001, 2001, 2024-01-15, 25000

ER diagram



Window Functions Implementation

1. Ranking

```
mysql> WITH department_revenue AS ( SELECT d.department_id, d.name AS department_name, SUM(t.amount) AS total_revenue, R
OW_NUMBER() OVER (ORDER BY SUM(t.amount) DESC) AS row_num, RANK() OVER (ORDER BY SUM(t.amount) DESC) AS rank_num, DENSE_
RANK() OVER (ORDER BY SUM(t.amount) DESC) AS dense_rank_num FROM transactions t JOIN departments d ON t.department_id =
d.department_id GROUP BY d.department_id, d.name ) SELECT department_name, total_revenue, row_num, rank_num, dense_rank_
num FROM department_revenue WHERE row_num <= 10 ORDER BY total_revenue DESC;
+-----------------+---------------+---------+----------+----------------+
| department_name | total_revenue | row_num | rank_num | dense_rank_num |
+-----------------+---------------+---------+----------+----------------+
| Finance Dept    |      86000.00 |       1 |        1 |              1 |
| Marketing Dept  |      38000.00 |       2 |        2 |              2 |
| HR Dept         |      22000.00 |       3 |        3 |              3 |
| IT Dept         |      11000.00 |       4 |        4 |              4 |
+-----------------+---------------+---------+----------+----------------+
4 rows in set (0.22 sec)
```

Interpretation: The ranking functions show finance as the top revenue generator followed by marketing, HR, and IT department, All ranking methods show consistent results since there are no tied revenues, making RANK and DENSE_RANK identical in this case.

2. Aggregate

```
mysql> SELECT YEAR(transaction_date) AS year, MONTH(transaction_date) AS month, SUM(amount) AS monthly_amount, SUM(SUM(a
mount)) OVER ( ORDER BY YEAR(transaction_date), MONTH(transaction_date) ROWS UNBOUNDED PRECEDING ) AS running_total_rows
, ROUND(AVG(SUM(amount)) OVER ( ORDER BY YEAR(transaction_date), MONTH(transaction_date) ROWS BETWEEN 2 PRECEDING AND CU
RRENT ROW ), 2) AS moving_avg_3month, MIN(SUM(amount)) OVER ( PARTITION BY YEAR(transaction_date) ) AS min_monthly_year,
 MAX(SUM(amount)) OVER ( PARTITION BY YEAR(transaction_date) ) AS max_monthly_year FROM transactions GROUP BY YEAR(trans
action_date), MONTH(transaction_date) ORDER BY year, month;
+------+-------+----------------+--------------------+-------------------+-----------------+-----------------+
| year | month | monthly_amount | running_total_rows | moving_avg_3month | min_monthly_year | max_monthly_year |
+------+-------+----------------+--------------------+-------------------+-----------------+-----------------+
| 2024 |     1 |       75000.00 |           75000.00 |          75000.00 |        75000.00 |        82000.00 |
| 2024 |     2 |       82000.00 |          157000.00 |          78500.00 |        75000.00 |        82000.00 |
+------+-------+----------------+--------------------+-------------------+-----------------+-----------------+
2 rows in set (0.04 sec)
```

Interpretation: The data shows a positive trend with february 2024 outperforming january. The 3-month moving avg is calculated using available data, and year-to-date mix/max values help identify monthly performance within the annual context.

3. Navigation

```
mysql> WITH monthly_totals AS ( SELECT YEAR(transaction_date) AS year, MONTH(transaction_date) AS month, SUM(amount) AS
current_month_amount, LAG(SUM(amount)) OVER (ORDER BY YEAR(transaction_date), MONTH(transaction_date)) AS previous_month
_amount, LEAD(SUM(amount)) OVER (ORDER BY YEAR(transaction_date), MONTH(transaction_date)) AS next_month_amount FROM tra
nsactions GROUP BY YEAR(transaction_date), MONTH(transaction_date) ) SELECT year, month, current_month_amount, previous_
month_amount, next_month_amount, ROUND(((current_month_amount - previous_month_amount) / previous_month_amount) * 100, 2
) AS mom_growth_percent, CASE WHEN current_month_amount > previous_month_amount THEN 'Increase' WHEN current_month_amoun
t < previous_month_amount THEN 'Decrease' ELSE 'No Change' END AS mom_trend FROM monthly_totals ORDER BY year, month;
+------+-------+----------------------+-----------------------+-------------------+--------------------+-----------+
| year | month | current_month_amount | previous_month_amount | next_month_amount | mom_growth_percent | mom_trend |
+------+-------+----------------------+-----------------------+-------------------+--------------------+-----------+
| 2024 |     1 |             75000.00 |                  NULL |          82000.00 |               NULL | No Change |
| 2024 |     2 |             82000.00 |              75000.00 |              NULL |               9.33 | Increase  |
+------+-------+----------------------+-----------------------+-------------------+--------------------+-----------+
2 rows in set (0.01 sec)
```

Interpretation: February 2024 shows a healthy month-over-month growth from January. The navigation functions effectively track the sequential performance changes, providing clear trend indicators for financial analysis.

4. Distribution

```
    SELECT t.transaction_id, d.name AS department_name, e.category, t.transaction_date, t.amount, NTILE(4) OVER (ORDE
R BY t.amount DESC) AS risk_quartile, CASE NTILE(4) OVER (ORDER BY t.amount DESC) WHEN 1 THEN 'High Risk' WHEN 2 THEN 'M
edium-High Risk' WHEN 3 THEN 'Medium-Low Risk' WHEN 4 THEN 'Low Risk' END AS risk_segment FROM transactions t JOIN depar
tments d ON t.department_id = d.department_id JOIN expense e ON t.expense_id = e.expense_id ORDER BY amount DESC;
+----------------+-----------------+---------------+------------------+----------+---------------+------------------+
| transaction_id | department_name | category      | transaction_date | amount   | risk_quartile | risk_segment     |
+----------------+-----------------+---------------+------------------+----------+---------------+------------------+
|           3006 | Finance Dept    | Transportation | 2024-02-10       | 30000.00 |             1 | High Risk        |
|           3001 | Finance Dept    | Transportation | 2024-01-15       | 25000.00 |             1 | High Risk        |
|           3005 | Marketing Dept  | Transportation | 2024-01-30       | 20000.00 |             1 | High Risk        |
|           3009 | Marketing Dept  | Operations     | 2024-02-20       | 18000.00 |             2 | Medium-High Risk |
|           3010 | Finance Dept    | Operations     | 2024-02-22       | 16000.00 |             2 | Medium-High Risk |
|           3002 | Finance Dept    | Operations     | 2024-01-20       | 15000.00 |             2 | Medium-High Risk |
|           3007 | HR Dept         | HR             | 2024-02-12       | 12000.00 |             3 | Medium-Low Risk  |
|           3003 | HR Dept         | HR             | 2024-01-18       | 10000.00 |             3 | Medium-Low Risk  |
|           3008 | IT Dept         | Operations     | 2024-02-15       |  6000.00 |             4 | Low Risk         |
|           3004 | IT Dept         | Operations     | 2024-01-25       |  5000.00 |             4 | Low Risk         |
+----------------+-----------------+---------------+------------------+----------+---------------+------------------+
10 rows in set (0.12 sec)
```

Interpretation: The NTILE function effectively divides transactions into risk quartiles, with the top 25% (amount >= $20,000) classified as high risk. Transportation expenses in finance and marketing departments dominate the high-risk category indicating areas requiring closer monitoring.

SOURCES

DBDiagram. (2024). *Database Relationship Diagrams Tool*. DBDiagram.io. Retrieved from https://dbdiagram.io/d

GeeksforGeeks. (2024). *Window functions in PL/SQL*. GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/plsql/window-functions-in-plsql/

Upmetrics. (2023). *How to write a business problem statement*. Upmetrics. Retrieved from https://upmetrics.co/blog/business-problem-statement