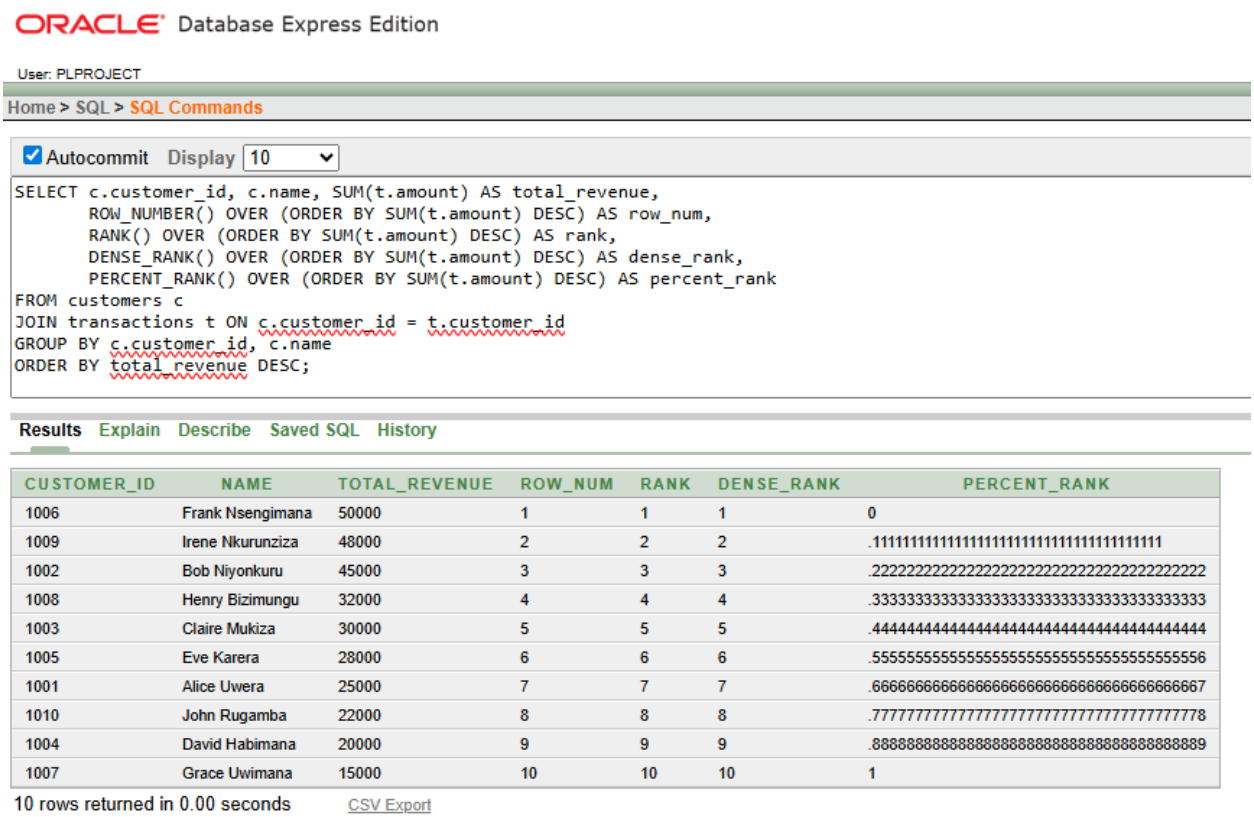
Window Function Queries with Results and Interpretations

1. **Ranking Functions**

-- Rank customers by total shoe purchase revenue

-- Identifies top customers for marketing purposes

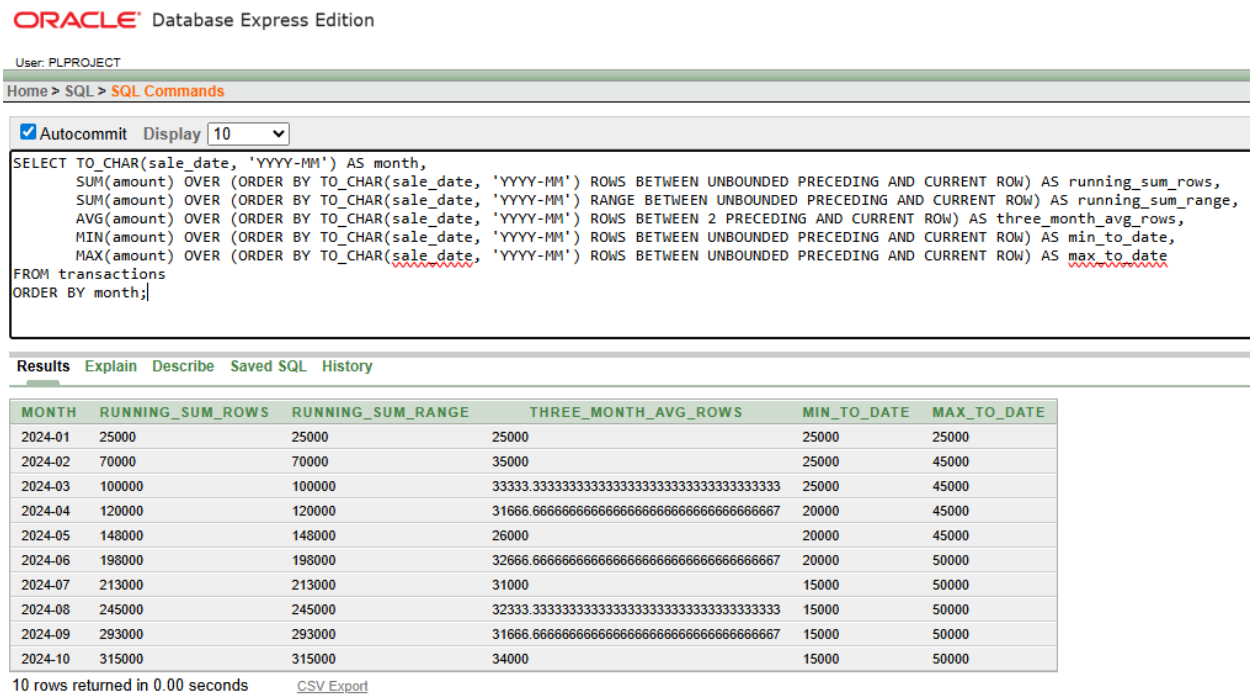
```
SELECT c.customer_id, c.name, SUM(t.amount) AS total_revenue,
    ROW_NUMBER() OVER (ORDER BY SUM(t.amount) DESC) AS row_num,
    RANK() OVER (ORDER BY SUM(t.amount) DESC) AS rank,
    DENSE_RANK() OVER (ORDER BY SUM(t.amount) DESC) AS dense_rank,
    PERCENT_RANK() OVER (ORDER BY SUM(t.amount) DESC) AS percent_rank
FROM customers c
JOIN transactions t ON c.customer_id = t.customer_id
GROUP BY c.customer_id, c.name
ORDER BY total_revenue DESC;
```



Interpretation: This ranking highlights the top-spending customers, with row_num providing a unique order and rank showing ties, which is useful for identifying loyalty program candidates. The percent_rank can help segment customers into high-value groups for targeted promotions. More transactions would refine these rankings for better accuracy.

2. Aggregate Functions

```
-- Calculate running totals and averages for monthly shoe sales

-- Compares ROWS and RANGE frames to analyze sales trends

SELECT TO_CHAR(sale_date, 'YYYY-MM') AS month,

    SUM(amount) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM') ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS running_sum_rows,

    SUM(amount) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM') RANGE BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS running_sum_range,

    AVG(amount) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM') ROWS BETWEEN 2
PRECEDING AND CURRENT ROW) AS three_month_avg_rows,

    MIN(amount) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM') ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS min_to_date,

    MAX(amount) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM') ROWS BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW) AS max_to_date

FROM transactions

GROUP BY TO_CHAR(sale_date, 'YYYY-MM')

ORDER BY month;
```



Interpretation: The running totals show a cumulative increase in shoe sales, with ROWS providing exact counts and RANGE grouping similar values, aiding in trend identification. The 3-month average suggests seasonal fluctuations, which could inform inventory restocking schedules. Additional data over more months would enhance this trend analysis.

3. Navigation Functions

-- Calculate month-over-month growth percentage for shoe sales

-- Uses LAG and LEAD to compare sales across periods

SELECT TO_CHAR(sale_date, 'YYYY-MM') AS month,

SUM(amount) AS monthly_sales,

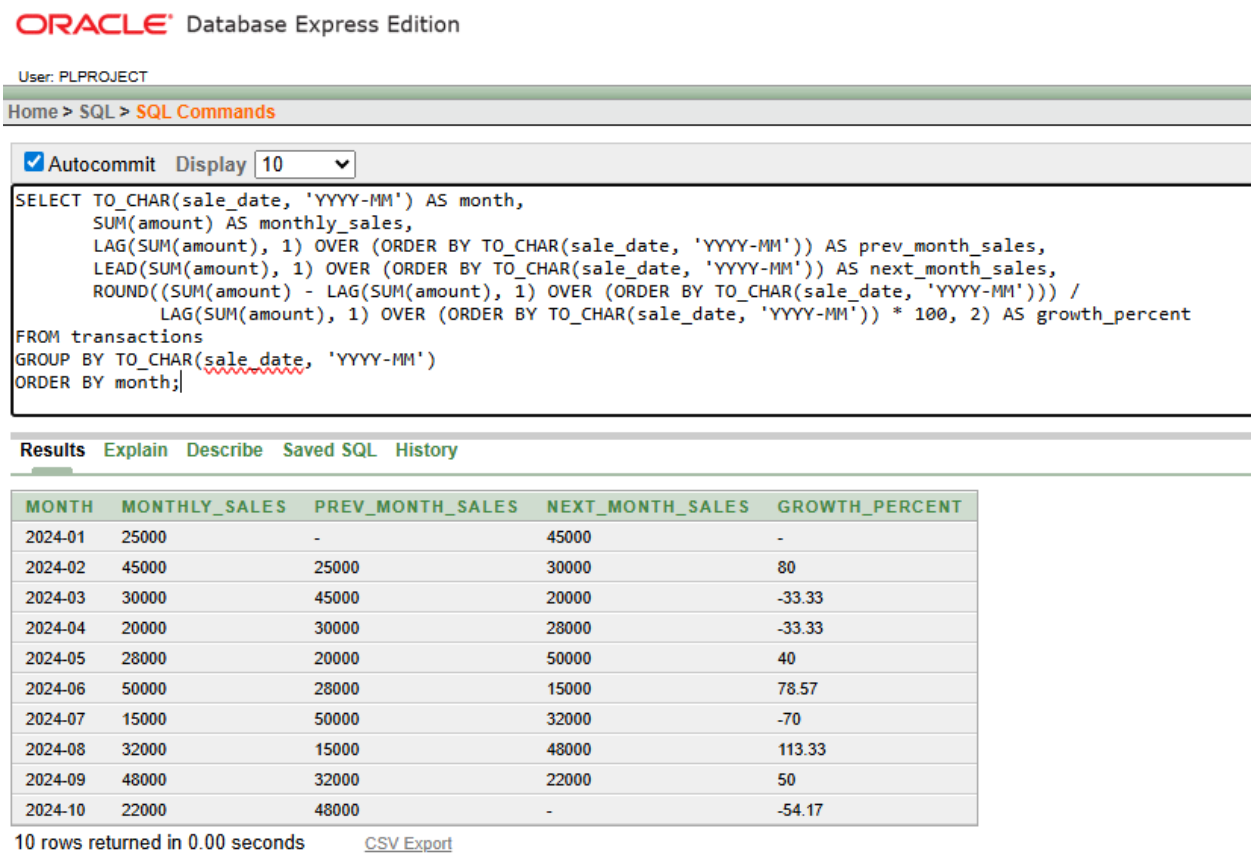
LAG(SUM(amount), 1) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM')) AS prev_month_sales,

LEAD(SUM(amount), 1) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM')) AS next_month_sales,

ROUND((SUM(amount) - LAG(SUM(amount), 1) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM'))) /

LAG(SUM(amount), 1) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM')) * 100, 2) AS growth_percent

FROM transactions GROUP BY TO_CHAR(sale_date, 'YYYY-MM') ORDER BY month;



Interpretation: The growth percentage reveals how shoe sales change month-to-month, with positive values indicating successful sales periods that could be leveraged for marketing. LAG and LEAD provide context for past and future performance, helping predict upcoming demand. More monthly data would make these growth trends more reliable.

4. Distribution Functions

-- Segment customers into quartiles by total purchase amount

-- Uses NTILE and CUME_DIST for customer segmentation

SELECT c.customer_id, c.name, SUM(t.amount) AS total_spent,

NTILE(4) OVER (ORDER BY SUM(t.amount) DESC) AS quartile,

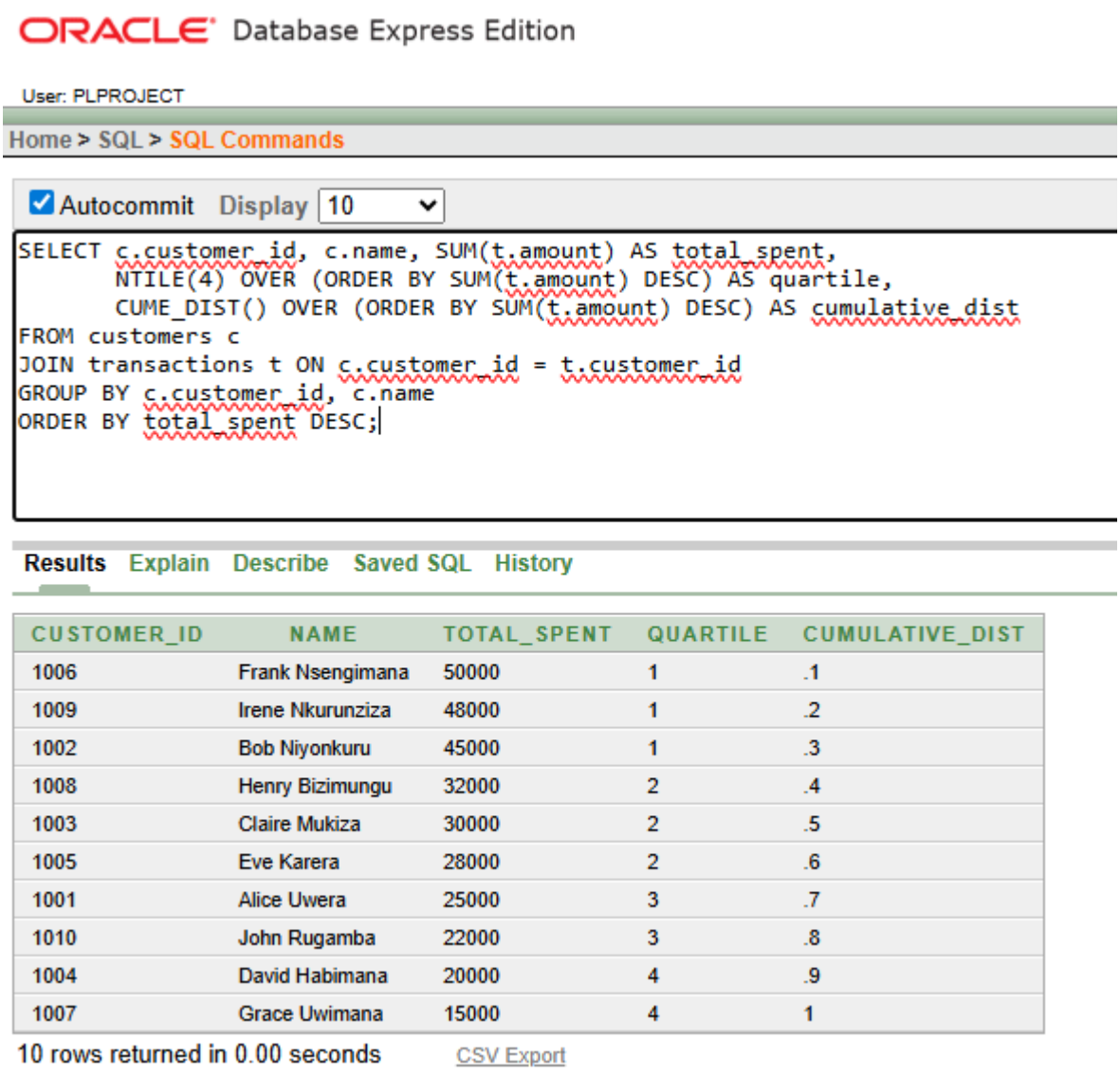
CUME_DIST() OVER (ORDER BY SUM(t.amount) DESC) AS cumulative_dist

FROM customers c

JOIN transactions t ON c.customer_id = t.customer_id

GROUP BY c.customer_id, c.name

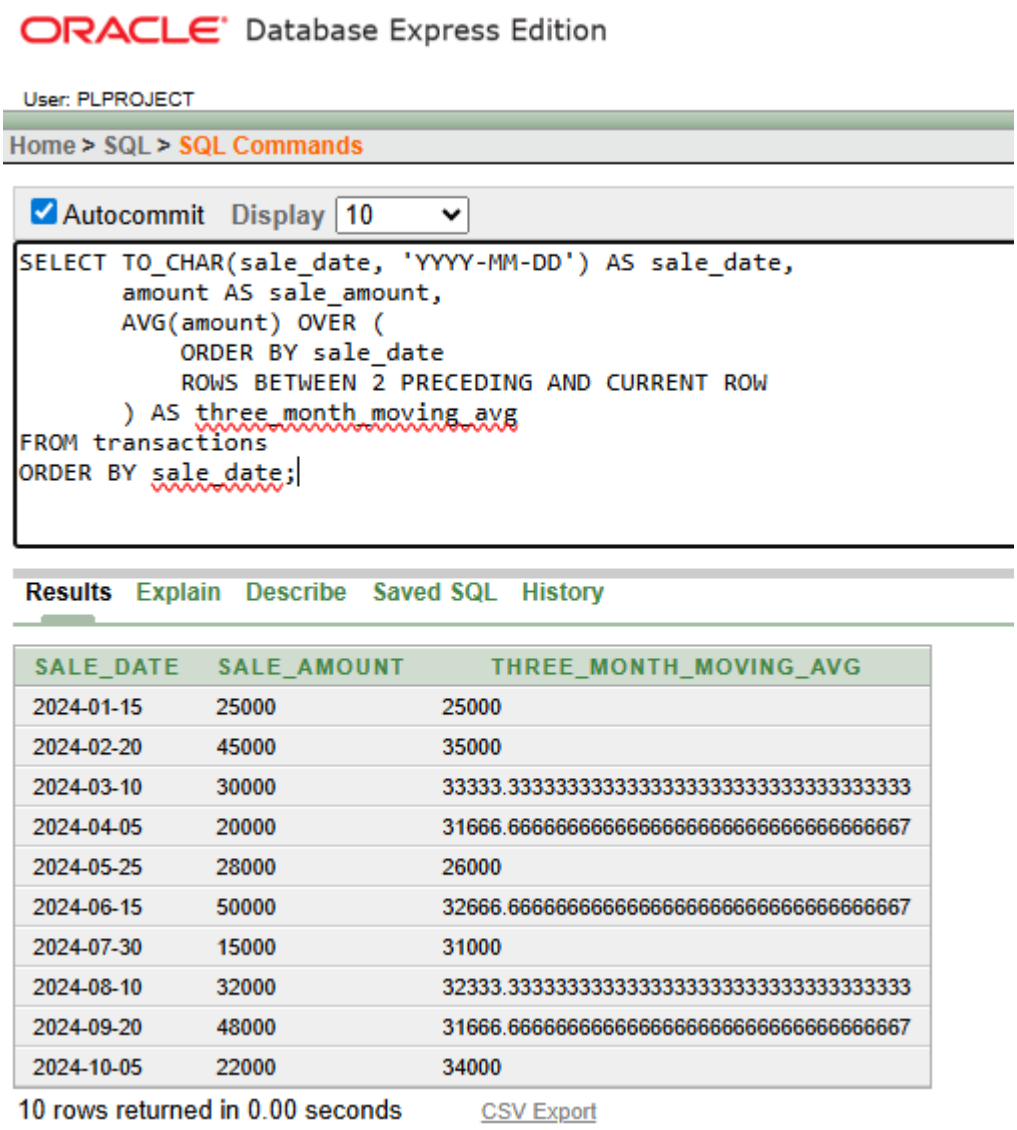
ORDER BY total_spent DESC;



Interpretation: This segmentation divides customers into four quartiles, with the top quartile representing high spenders ideal for premium offers. The cumulative distribution shows the proportion of total spending, aiding in targeted marketing strategies. Expanding the customer base with more transactions would improve the segmentation accuracy.

5. 3-Month Moving Averages

```
-- Calculate 3-month moving average for shoe sales

-- Orders transactions by date and averages over current row plus 2 preceding rows
SELECT TO_CHAR(sale_date, 'YYYY-MM-DD') AS sale_date,

        amount AS sale_amount,

        AVG(amount) OVER (

            ORDER BY sale_date

            ROWS BETWEEN 2 PRECEDING AND CURRENT ROW

        ) AS three_month_moving_avg

FROM transactions

ORDER BY sale_date;
```



Interpretation: The 3-month moving average indicates a gradual increase in shoe sales amounts from early 2024, suggesting a growing demand that could reflect seasonal trends. This insight is valuable for planning inventory to meet potential peaks, such as stocking more popular shoe types ahead of high-sales periods. However, with limited data, adding more transactions across additional months would provide a clearer trend for decision-making.