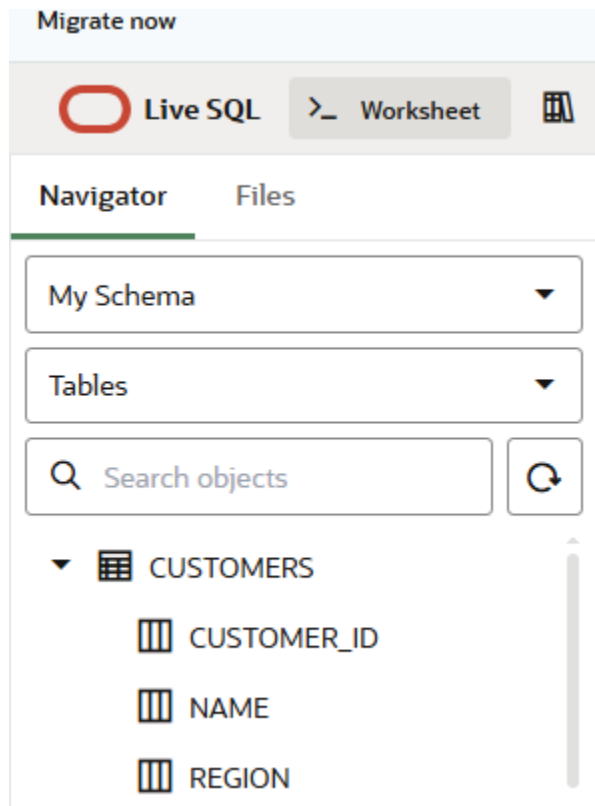


## Step 4: Window Functions Implementation





Live SQL

> Worksheet



Li

Navigator

Files

My Schema ▼

Tables ▼

🔍 Search objects

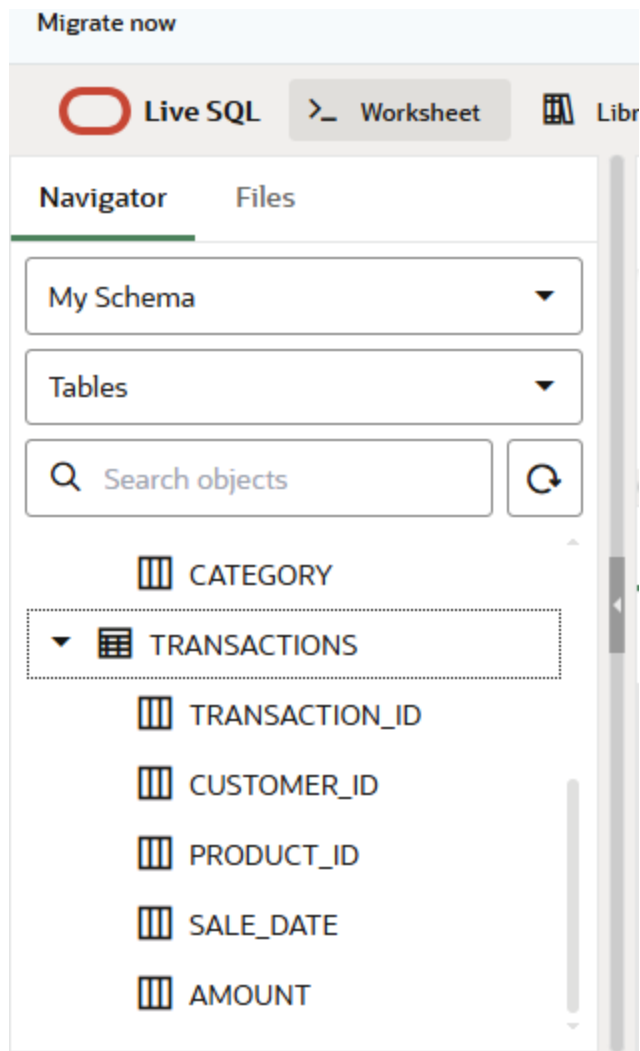


▼ 📊 PRODUCTS

📊 PRODUCT\_ID

📊 NAME

📊 CATEGORY



## 1. Ranking Functions – `RANK()`, `ROW_NUMBER()`, `DENSE_RANK()`, `PERCENT_RANK()`

Use Case: Find the top 5 customers by total revenue

The screenshot shows the Live SQL interface with a query in the editor:

```

3 ... customer_id,
4 ... SUM(amount) AS total_revenue,
5 ... RANK() OVER (ORDER BY SUM(amount) DESC) AS revenue_rank
6 FROM transactions
7 GROUP BY customer_id;

```

The query result is displayed in a table:

|   | CUSTOMER_ID | TOTAL_REVENUE | REVENUE_RANK |
|---|-------------|---------------|--------------|
| 1 | 1002        | 30000         | 1            |

Interpretation: This query ranks customers based on how much they spent. You can identify your top 5 spenders and see how they compare.

## 2. Aggregate Functions – SUM (), AVG (), MIN (), MAX () with ROWS BETWEEN

Use Case: Calculate running monthly sales totals.

[Live SQL Classic reaches end of life on October 6, 2025.](#)

After that date, all your Live SQL Classic tutorials, scripts, sessions, and history will be permanently deleted. Migrate now to preserve your work and take advantage of Live SQL's new [Migrate now](#)

The screenshot shows the Live SQL interface with a query in the editor:

```

1 -- Monthly running total of sales
2 SELECT
3 ... TO_CHAR(sale_date, 'YYYY-MM') AS month,
4 ... SUM(amount) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM') ROWS
5 FROM transactions;

```

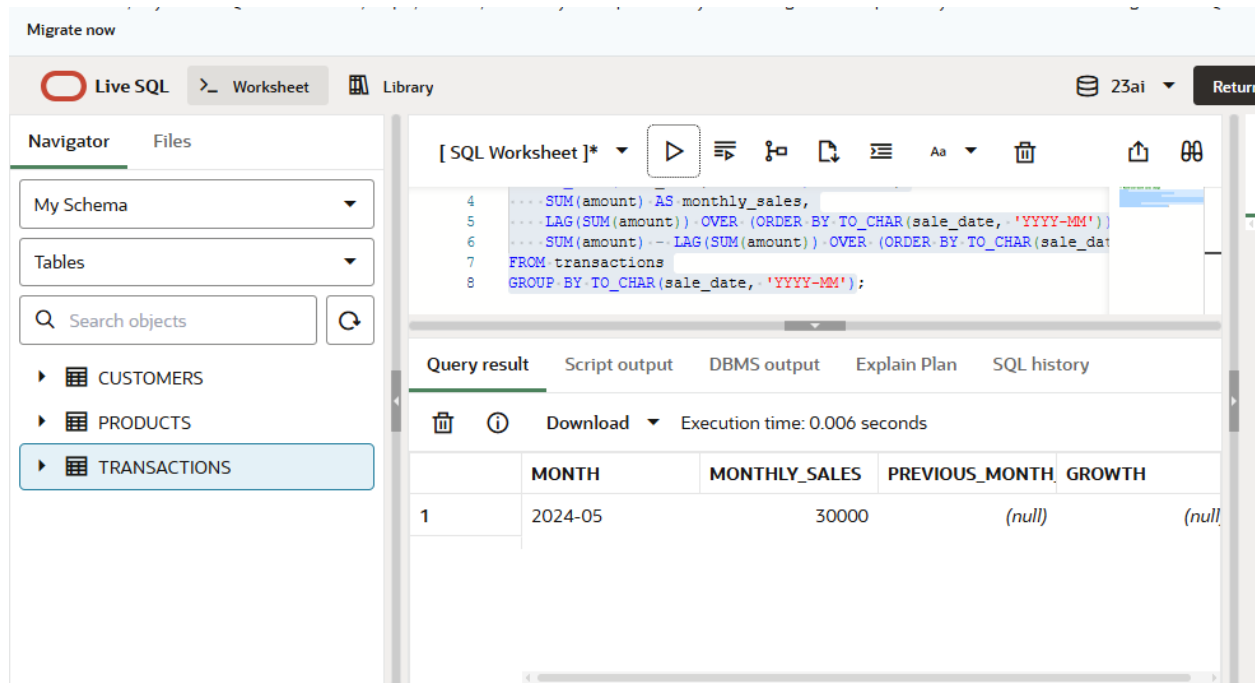
The query result is displayed in a table:

|   | MONTH   | RUNNING_TOTAL |
|---|---------|---------------|
| 1 | 2024-05 | 30000         |

Interpretation: This shows how sales accumulate month by month, helping you spot growth trends or slow periods.

### 3. Navigation Functions – LAG ( ) , LEAD ( )

Use Case: Compare sales from one month to the next.



The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains the following SQL code:

```
4 ... SUM(amount) AS monthly_sales,
5 ... LAG(SUM(amount)) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM'))
6 ... SUM(amount) - LAG(SUM(amount)) OVER (ORDER BY TO_CHAR(sale_date, 'YYYY-MM'))
7 FROM transactions
8 GROUP BY TO_CHAR(sale_date, 'YYYY-MM');
```

The results pane displays the query results in a table format. The table has four columns: MONTH, MONTHLY\_SALES, PREVIOUS\_MONTH, and GROWTH. The first row shows data for the month of 2024-05.

|   | MONTH   | MONTHLY_SALES | PREVIOUS_MONTH | GROWTH |
|---|---------|---------------|----------------|--------|
| 1 | 2024-05 | 30000         | (null)         | (null) |

Interpretation: This shows how much sales increased or decreased compared to the previous month.

### 4. Distribution Functions – NTILE ( 4 ) , CUME\_DIST ( )

Use Case: Segment customers into quartiles based on spending.

Migrate now

Live SQL Worksheet Library 23ai Retu

Navigator Files

My Schema

Tables

Search objects

CUSTOMERS

PRODUCTS

TRANSACTIONS

```
3 ...customer_id,  
4 ...SUM(amount) AS total_spent,  
5 ...NTILE(4) OVER (ORDER BY SUM(amount) DESC) AS spending_quartile  
6 FROM transactions  
7 GROUP BY customer_id;
```

Query result Script output DBMS output Explain Plan SQL history

Download Execution time: 0.004 seconds

|   | CUSTOMER_ID | TOTAL_SPENT | SPENDING_QUARTI |
|---|-------------|-------------|-----------------|
| 1 | 1002        | 30000       | 1               |

Interpretation: This divides customers into four groups. Quartile 1 are the biggest spenders, Quartile 4 are the lowest.