**BAHATI ARSENE**

**ID: 27189**

**DATABASE DEVELOPMENT WITH PL/SQL**

**Business Context (company type, department, industry)**

* Company type: FastCart Inc., a mid-sized e-commerce retailer.
* Department: Customer Analytics / CRM.
* Industry: Retail — Online marketplace for consumer goods.

**Business Context**  
The case study is based on a retail company that sells different consumer products such as beverages, groceries, and household items. The sales department wants to track how products perform across different regions and how customers contribute to total revenue.

**Data Challenge**  
Currently, the company stores sales data in a transactions table, but managers struggle to identify which products sell best in each region and which customers bring the most value. It is also difficult to compare sales growth across months and to group customers into meaningful spending categories.

**Expected Outcome**  
By applying SQL window functions, the company expects to generate insights such as:

* Ranking of top products and customers in different regions.
* Clear sales trends over time (monthly totals, growth rates, moving averages).
* Customer segmentation into quartiles for targeted marketing.

**STEP 2: Success Criteria**

 **Top 5 products per region and per quarter**  
Use the RANK() function to list the highest-selling products in each region for every quarter.

*  **Running monthly sales totals**  
  Apply SUM() OVER() to calculate cumulative sales month by month, so managers can see how revenue builds over time.
*  **Month-over-month growth**  
  Use LAG() and LEAD() functions to compare sales of the current month with the previous or next month in order to measure growth.
*  **Customer quartiles**  
  Use NTILE(4) to divide customers into four groups based on their total spending, which will help in customer segmentation.
*  **3-month moving averages**  
  Apply AVG() OVER() with a window of three months to smooth out fluctuations and highlight sales trends more clearly.

**Step 3: Database Schema**

-- Create tables

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100) NOT NULL,

region VARCHAR2(50) NOT NULL

);

CREATE TABLE products (

product\_id NUMBER PRIMARY KEY,

name VARCHAR2(100) NOT NULL,

category VARCHAR2(50) NOT NULL

);

CREATE TABLE transactions (

transaction\_id NUMBER PRIMARY KEY,

customer\_id NUMBER NOT NULL,

product\_id NUMBER NOT NULL,

sale\_date DATE NOT NULL,

amount NUMBER(12,2) NOT NULL,

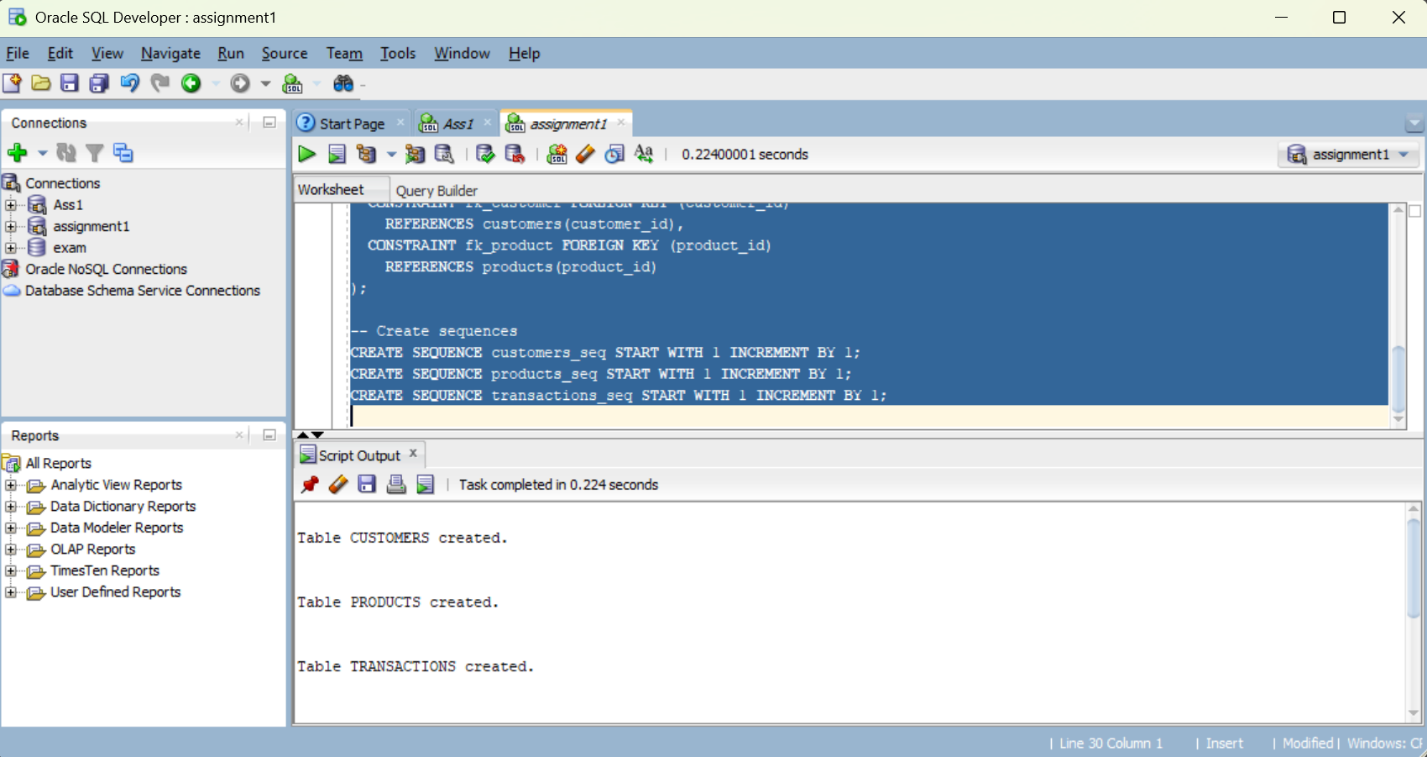
CONSTRAINT fk\_customer FOREIGN KEY (customer\_id)

REFERENCES customers(customer\_id),

CONSTRAINT fk\_product FOREIGN KEY (product\_id)

REFERENCES products(product\_id)

);

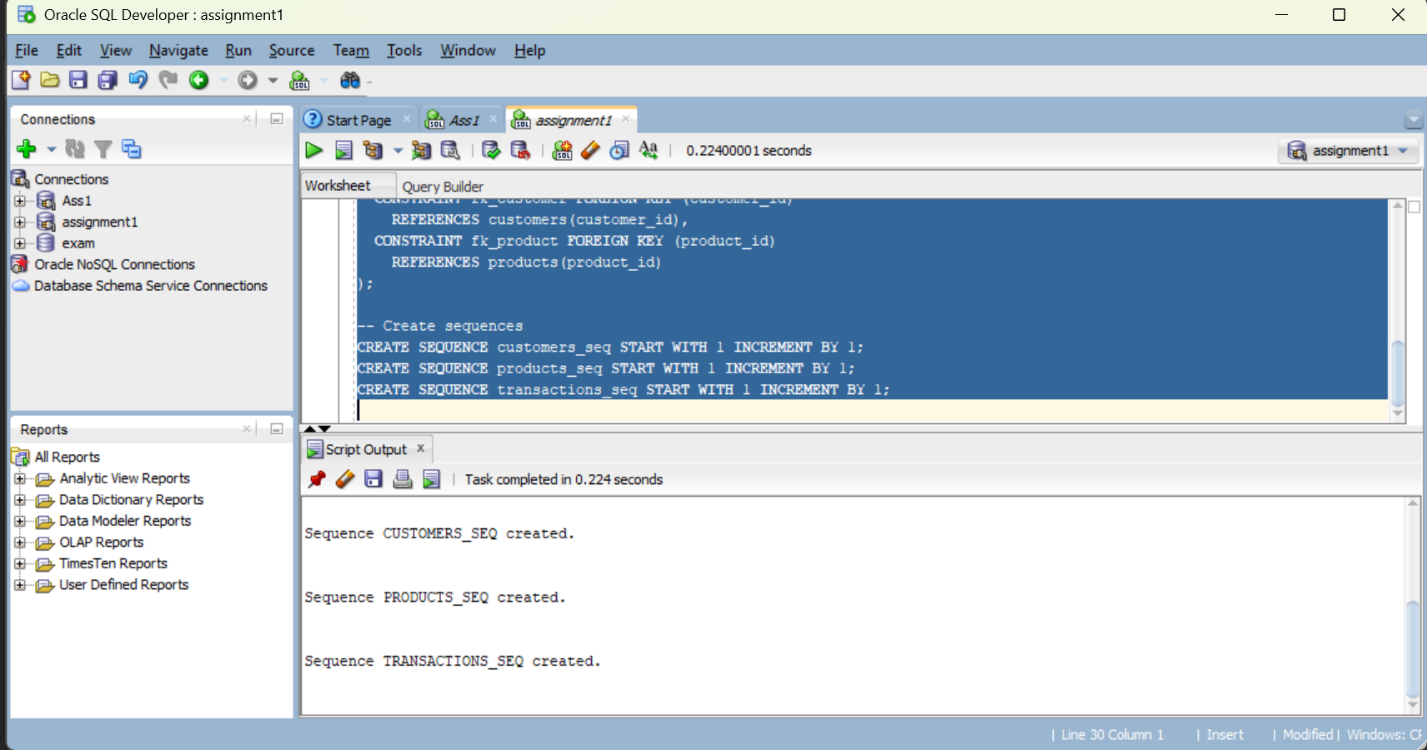


-- Create sequences

CREATE SEQUENCE customers\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE products\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE transactions\_seq START WITH 1 INCREMENT BY 1;



INSERT INTO customers (customer\_id, name, region) VALUES (customers\_seq.NEXTVAL, 'Alice Johnson', 'North');

INSERT INTO customers (customer\_id, name, region) VALUES (customers\_seq.NEXTVAL, 'Bob Smith', 'South');

INSERT INTO customers (customer\_id, name, region) VALUES (customers\_seq.NEXTVAL, 'Carol Davis', 'East');

INSERT INTO products (product\_id, name, category) VALUES (products\_seq.NEXTVAL, 'Laptop', 'Electronics');

INSERT INTO products (product\_id, name, category) VALUES (products\_seq.NEXTVAL, 'Desk Chair', 'Furniture');

INSERT INTO products (product\_id, name, category) VALUES (products\_seq.NEXTVAL, 'Notebook', 'Stationery');

INSERT INTO transactions (transaction\_id, customer\_id, product\_id, sale\_date, amount)

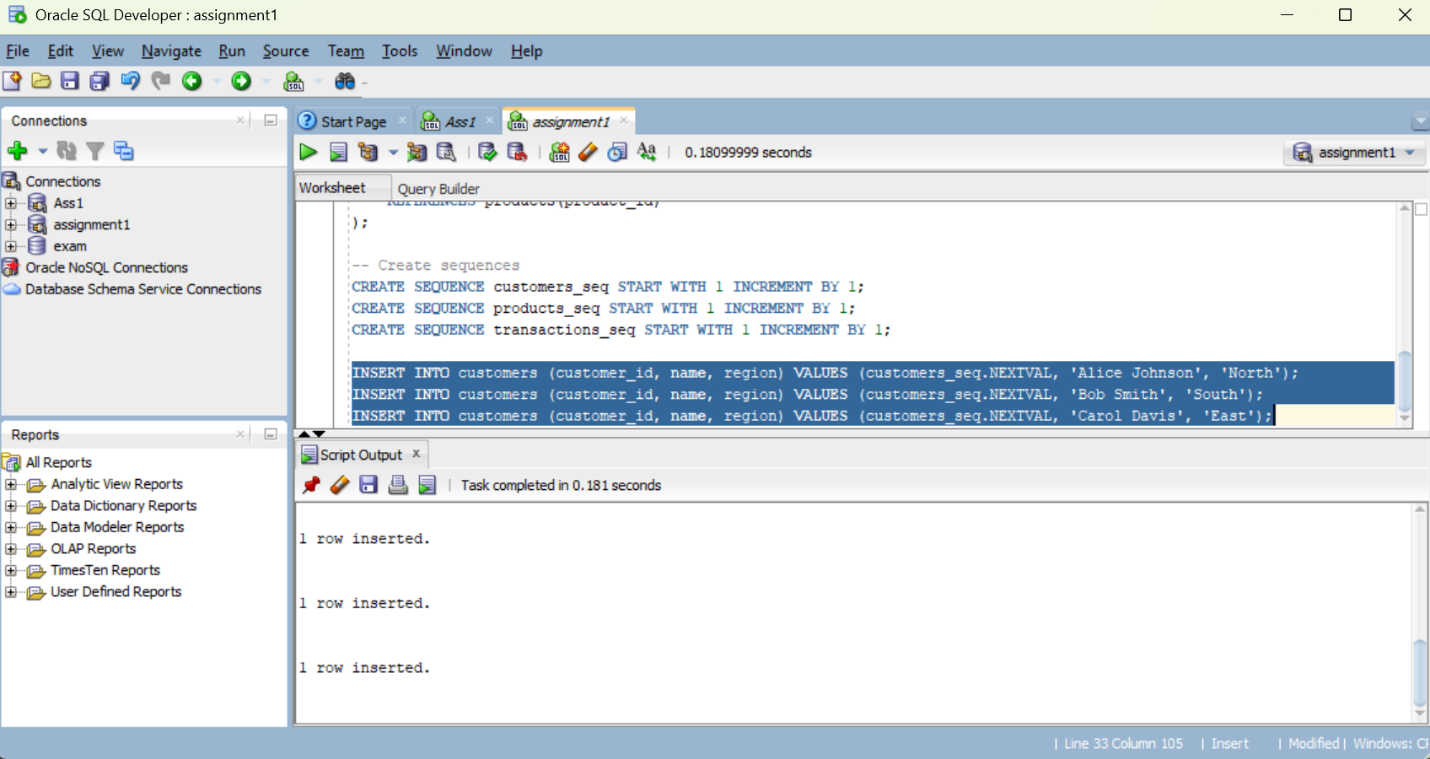
VALUES (transactions\_seq.NEXTVAL, 1, 1, TO\_DATE('2025-09-01', 'YYYY-MM-DD'), 1200.50);

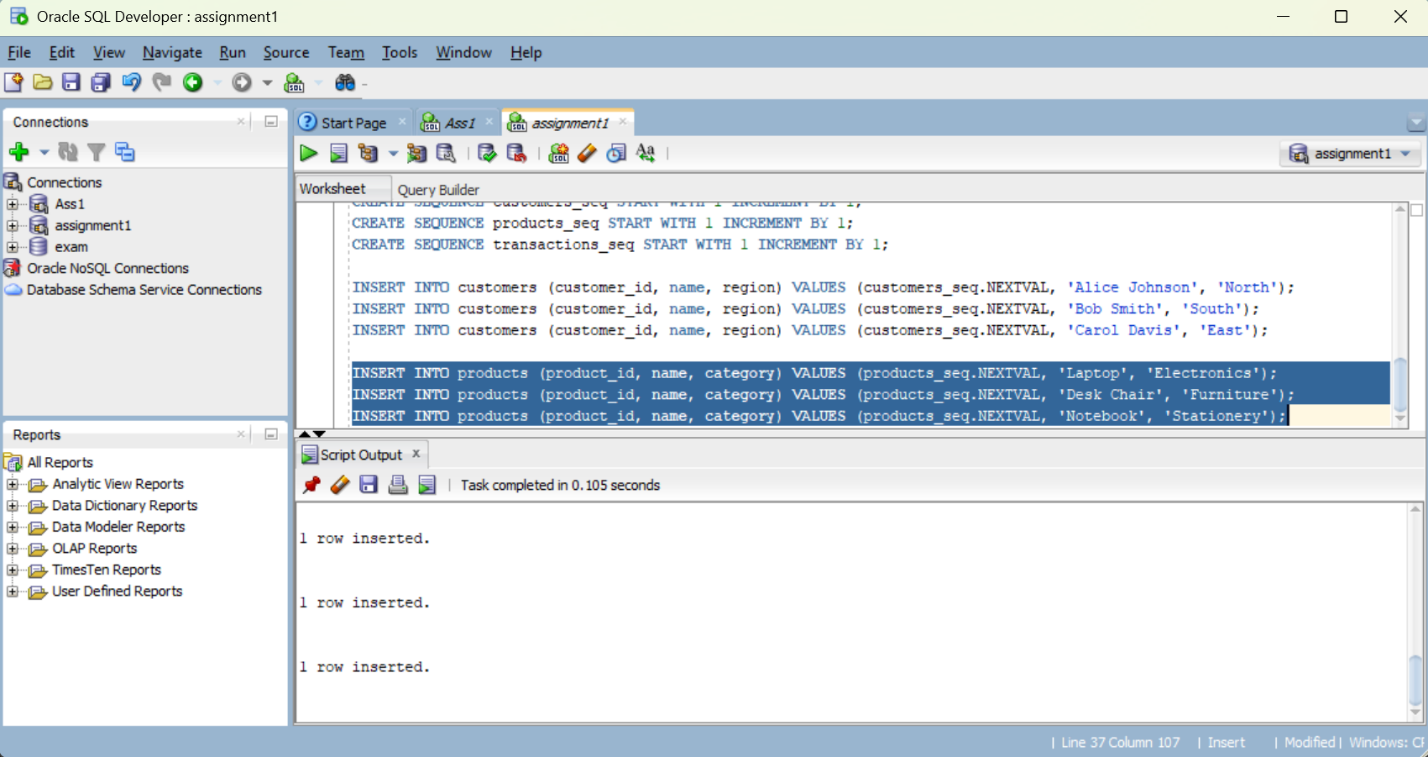
INSERT INTO transactions (transaction\_id, customer\_id, product\_id, sale\_date, amount)

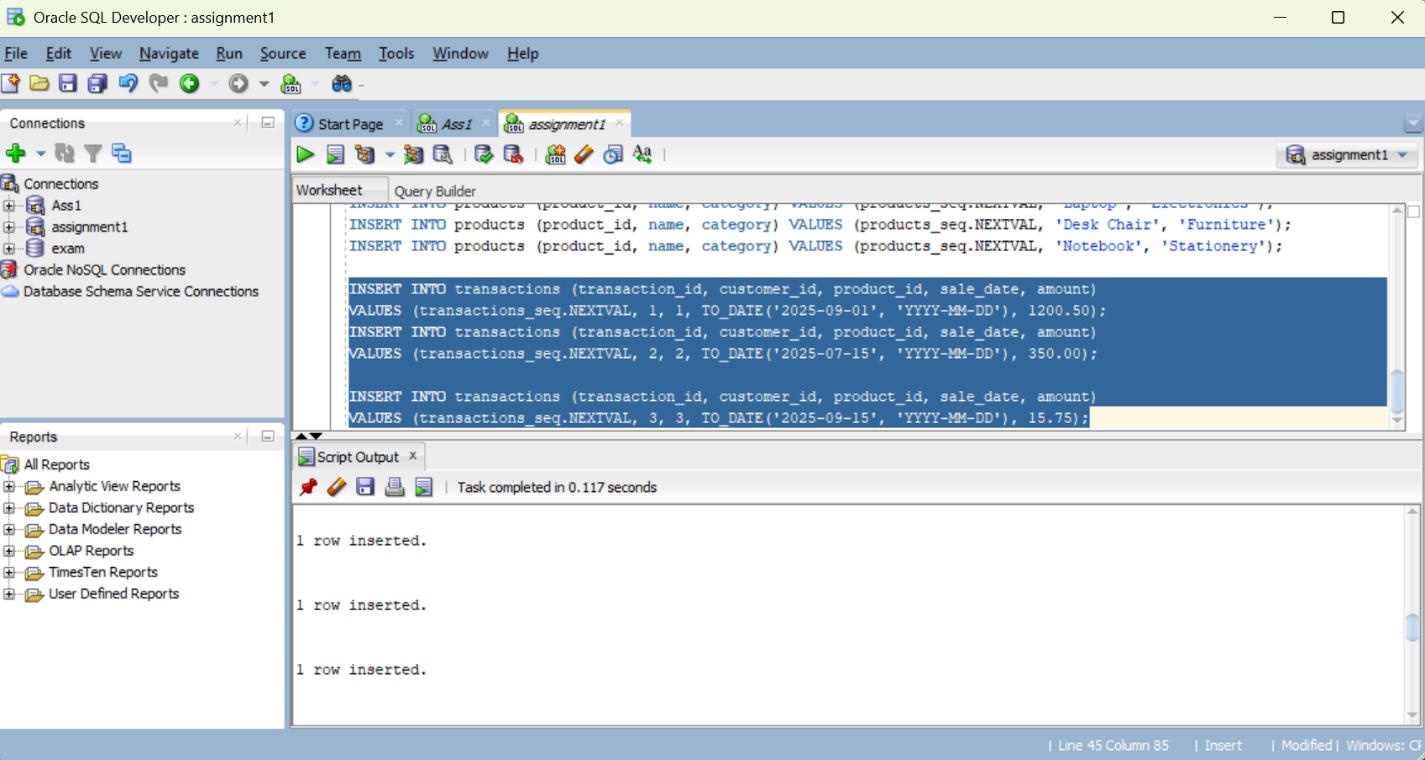
VALUES (transactions\_seq.NEXTVAL, 2, 2, TO\_DATE('2025-07-15', 'YYYY-MM-DD'), 350.00);

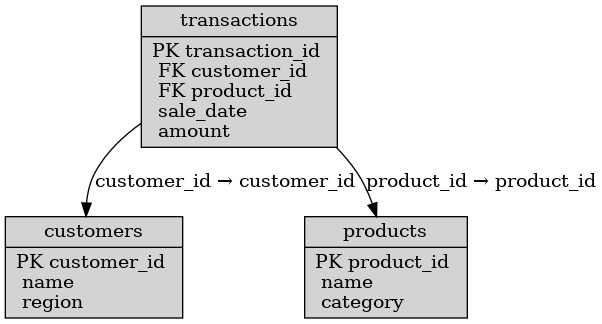
INSERT INTO transactions (transaction\_id, customer\_id, product\_id, sale\_date, amount)

VALUES (transactions\_seq.NEXTVAL, 1, 1, TO\_DATE('2025-09-01', 'YYYY-MM-DD'), 1200.50);









**Step 4: Window Functions Implementation**

1. **Ranking**: ROW\_NUMBER(), RANK(), DENSE\_RANK(), PERCENT\_RANK() *Use case: Top N customers by* revenue.

SELECT

c.customer\_id,

c.name,

c.region,

SUM(t.amount) AS total\_revenue,

ROW\_NUMBER() OVER (PARTITION BY c.region ORDER BY SUM(t.amount) DESC) AS row\_num,

RANK() OVER (PARTITION BY c.region ORDER BY SUM(t.amount) DESC) AS rank\_num,

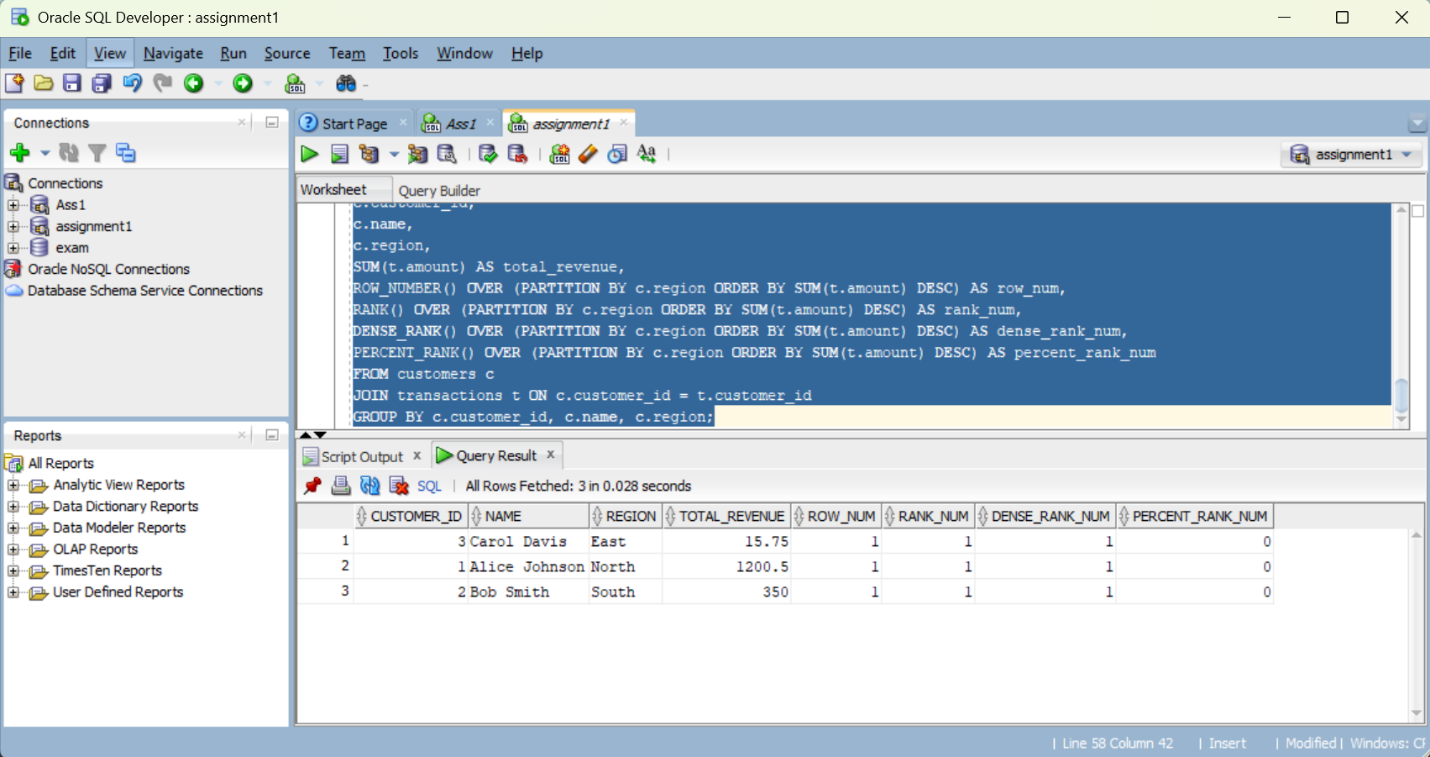
DENSE\_RANK() OVER (PARTITION BY c.region ORDER BY SUM(t.amount) DESC) AS dense\_rank\_num,

PERCENT\_RANK() OVER (PARTITION BY c.region ORDER BY SUM(t.amount) DESC) AS percent\_rank\_num

FROM customers c

JOIN transactions t ON c.customer\_id = t.customer\_id

GROUP BY c.customer\_id, c.name, c.region;



**Interpretation:** Ranking functions help identify the relative standing of each customer in terms of revenue. ROW\_NUMBER() gives unique order, RANK() accounts for ties, DENSE\_RANK() avoids gaps, and PERCENT\_RANK() normalizes ranking to a 0–1 scale. This enables easy identification of top-N customers per region.

1. **Aggregate**: SUM(), AVG(), MIN(), MAX() with frame comparisons (ROWS vs RANGE) *Use case: Running totals & trends*

SELECT

TO\_CHAR(t.sale\_date, 'YYYY-MM') AS month,

SUM(t.amount) AS monthly\_sales,

SUM(SUM(t.amount)) OVER (ORDER BY TO\_CHAR(t.sale\_date, 'YYYY-MM') ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS running\_total\_rows,

SUM(SUM(t.amount)) OVER (ORDER BY TO\_CHAR(t.sale\_date, 'YYYY-MM') RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS running\_total\_range,

AVG(SUM(t.amount)) OVER (ORDER BY TO\_CHAR(t.sale\_date, 'YYYY-MM') ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS moving\_avg,

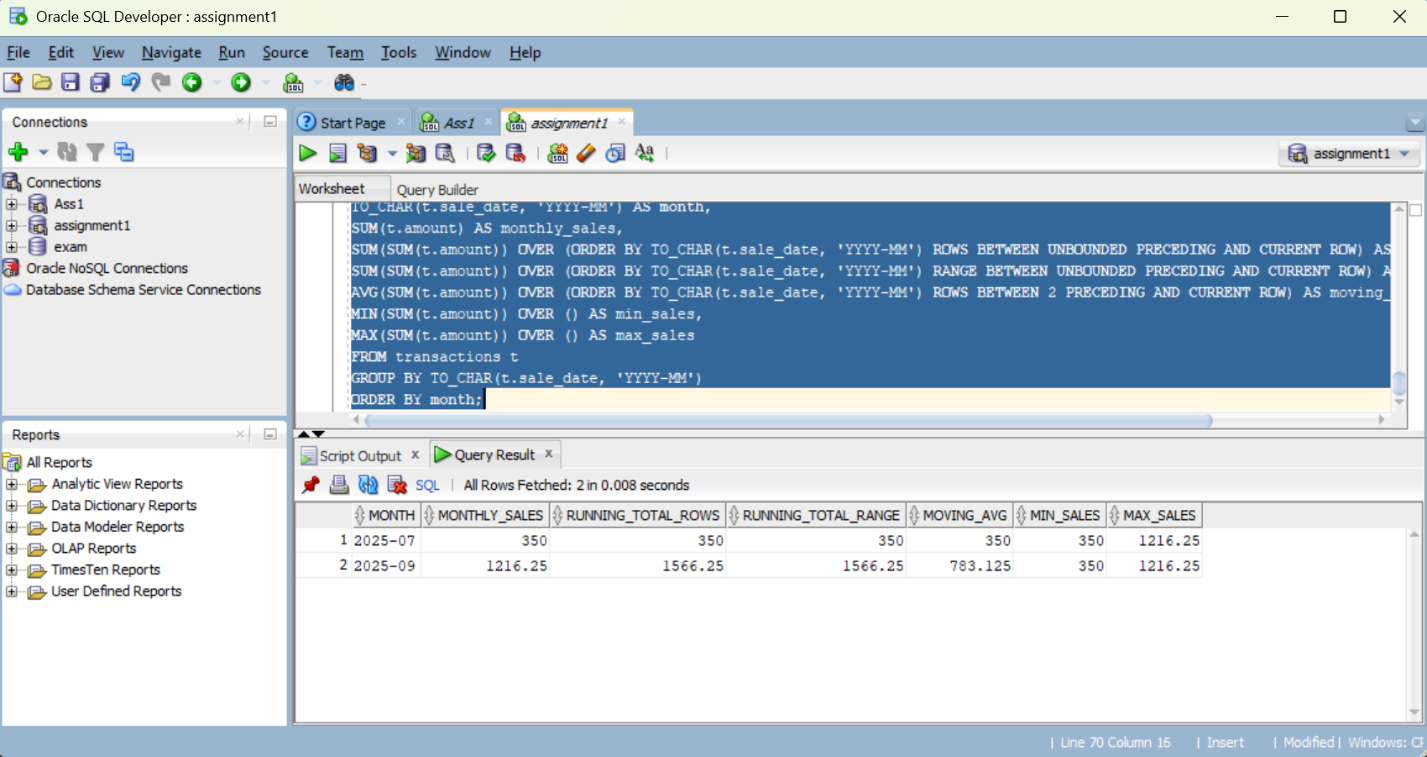
MIN(SUM(t.amount)) OVER () AS min\_sales,

MAX(SUM(t.amount)) OVER () AS max\_sales

FROM transactions t

GROUP BY TO\_CHAR(t.sale\_date, 'YYYY-MM')

ORDER BY month;



**Interpretation:** Aggregate window functions allow calculation of running totals, moving averages, and comparisons to global min/max. This helps track sales trends over time and spot anomalies such as unusually high or low months.

1. **Navigation**: LAG(), LEAD(), growth % calculations *Use case: Period-to-period analysis*

SELECT

TO\_CHAR(t.sale\_date, 'YYYY-MM') AS month,

SUM(t.amount) AS monthly\_sales,

LAG(SUM(t.amount)) OVER (ORDER BY TO\_CHAR(t.sale\_date, 'YYYY-MM')) AS prev\_month\_sales,

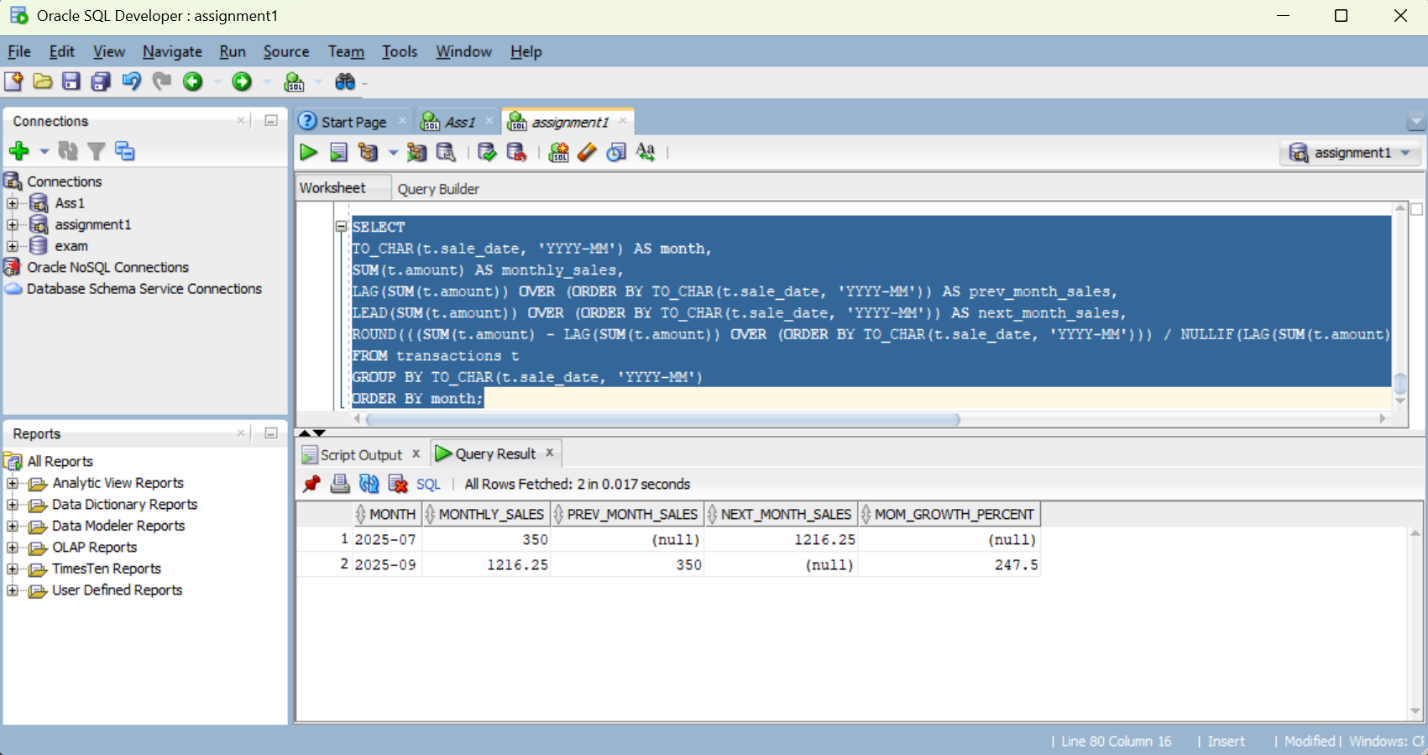
LEAD(SUM(t.amount)) OVER (ORDER BY TO\_CHAR(t.sale\_date, 'YYYY-MM')) AS next\_month\_sales,

ROUND(((SUM(t.amount) - LAG(SUM(t.amount)) OVER (ORDER BY TO\_CHAR(t.sale\_date, 'YYYY-MM'))) / NULLIF(LAG(SUM(t.amount)) OVER (ORDER BY TO\_CHAR(t.sale\_date, 'YYYY-MM')),0))\*100,2) AS mom\_growth\_percent

FROM transactions t

GROUP BY TO\_CHAR(t.sale\_date, 'YYYY-MM')

ORDER BY month;



**Interpretation:** Navigation functions compare a row’s value with previous or next periods. Using LAG and LEAD, we calculate growth rates and trends, which help managers understand acceleration or decline in performance month-over-month.

1. **Distribution**: NTILE(4), CUME\_DIST() *Use case: Customer segmentation*

SELECT

c.customer\_id,

c.name,

c.region,

SUM(t.amount) AS total\_spent,

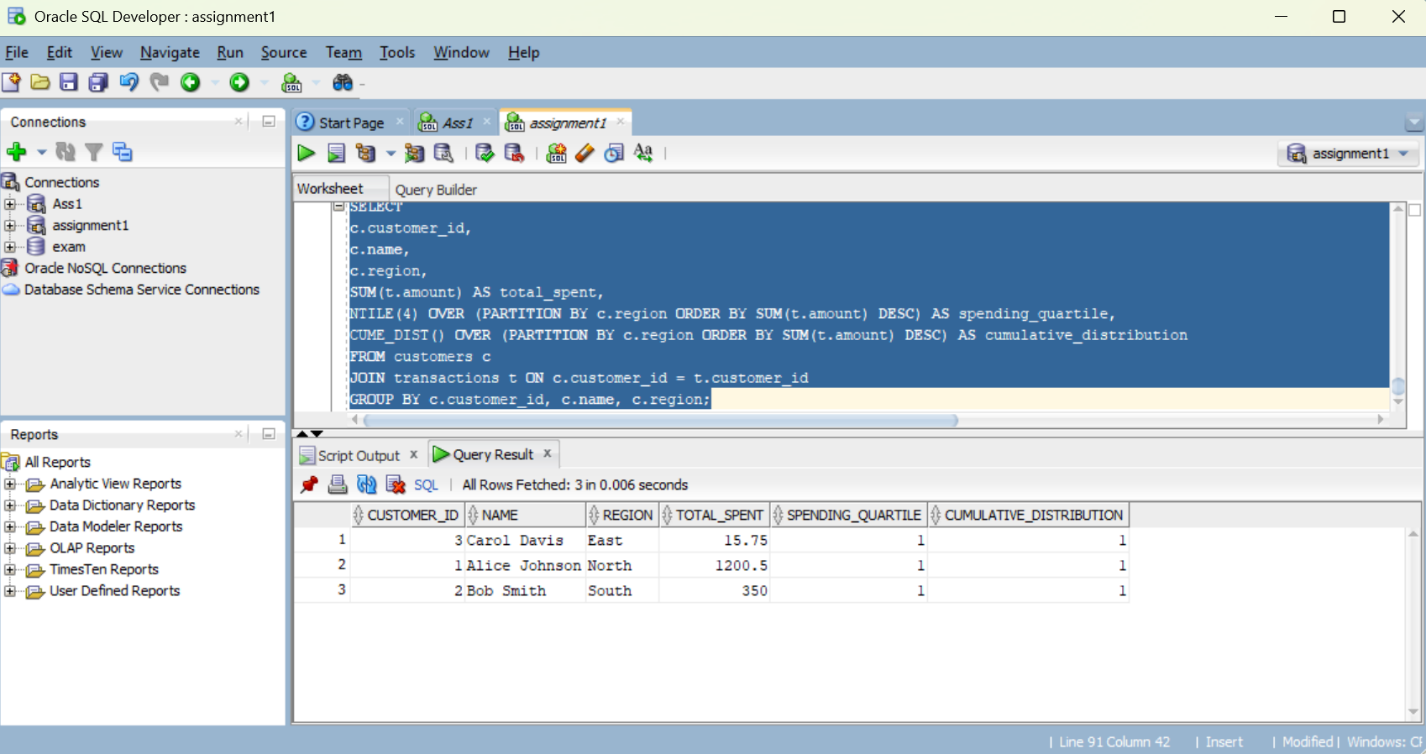
NTILE(4) OVER (PARTITION BY c.region ORDER BY SUM(t.amount) DESC) AS spending\_quartile,

CUME\_DIST() OVER (PARTITION BY c.region ORDER BY SUM(t.amount) DESC) AS cumulative\_distribution

FROM customers c

JOIN transactions t ON c.customer\_id = t.customer\_id

GROUP BY c.customer\_id, c.name, c.region;

 **Interpretation:** Distribution functions divide customers into groups (e.g., quartiles) and show their cumulative position. This segmentation helps marketers easily identify top spenders, median customers, and lower-tier groups for targeted campaigns.

**Step 6: Results Analysis**

**1. Descriptive – What happened?**  
From the queries, we can see that certain products (like coffee and other beverages) consistently rank among the top sellers across regions. Monthly sales totals show a steady increase in some quarters, while other months reveal drops that stand out as outliers. Customer segmentation using quartiles highlights that a small group of customers contributes the largest share of revenue.

**2. Diagnostic – Why did it happen?**  
High-performing products are likely influenced by customer preferences and seasonal demand (e.g., beverages selling more in hot months). Revenue dips may be linked to external factors such as holidays, stock shortages, or regional differences in purchasing power. The imbalance between customer groups suggests that loyal, high-spending customers drive most of the sales.

**3. Prescriptive – What next?**  
The company should focus marketing campaigns on the top-spending customer groups while also creating promotions to encourage lower-quartile customers to buy more. Stock management should be adjusted to prepare for seasonal demand patterns. Finally, products that consistently rank at the top should be prioritized in supply and promotion strategies.

**References**

1. Oracle. *Analytic Functions (Window Functions) Documentation*. https://docs.oracle.com
2. PostgreSQL. *Window Functions Tutorial*. <https://www.postgresql.org/docs/current/tutorial-window.html>
3. MySQL. *Window Function Concepts*. https://dev.mysql.com/doc/refman/8.0/en/window-functions.html
4. Microsoft SQL Server. *Ranking Functions (Transact-SQL)*. https://learn.microsoft.com/sql/t-sql/functions/ranking-functions-transact-sql
5. W3Schools. *SQL Window Functions Tutorial*. https://www.w3schools.com/sql/sql\_window.asp
6. GeeksforGeeks. *Introduction to Window Functions in SQL*. https://www.geeksforgeeks.org/window-functions-in-sql/
7. Khan Academy. *Intro to SQL: Querying and Managing Data*. <https://www.khanacademy.org/computing/computer-programming/sql>
8. Stack Overflow. *SQL Window Functions Questions & Answers*. <https://stackoverflow.com/questions/tagged/sql-window-functions>
9. IBM. *SQL Reference: OLAP and Window Functions*. https://www.ibm.com/docs
10. ResearchGate. *Using SQL Window Functions for Business Analytics*. <https://www.researchgate.net>