**CSA05: DATABASE MANAGEMENT SYSTEMS-ASSIGNMENT-2**

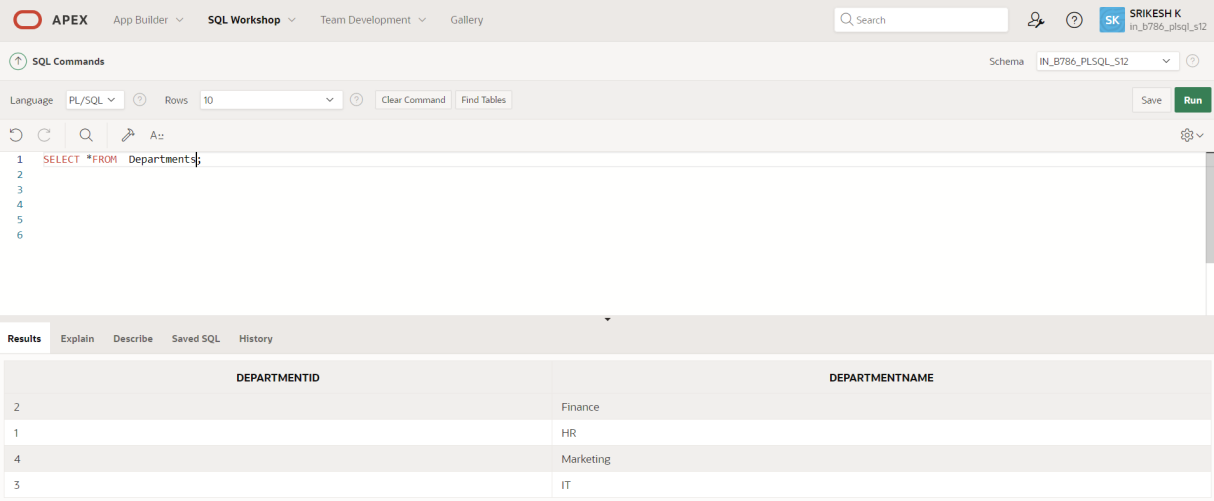
**Name : R charan tej**

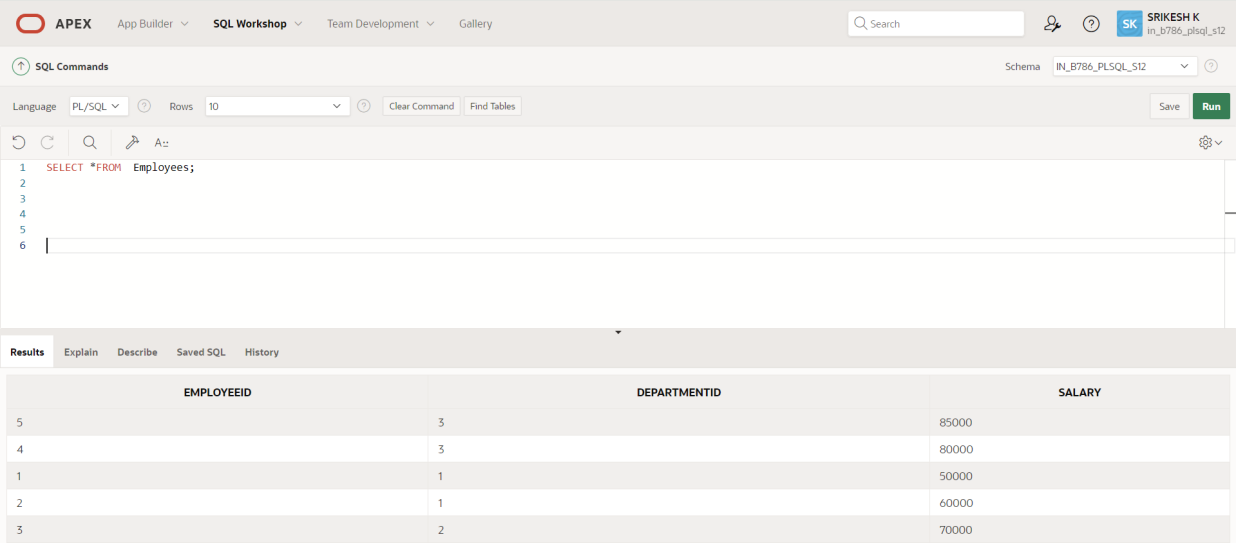
**Reg num :192311001**

**Question 1: Top 3 Departments with Highest Average Salary**

**Write a SQL query to find the top 3 departments with the highest average salary of employees. Ensure departments with no employees show an average salary of NULL.**

CREATING TABLES :

****

****

QUERY :

SELECT

d.DepartmentID,

d.DepartmentName,

AVG(e.Salary) AS AvgSalary

FROM

Departments d

LEFT JOIN

Employees e ON d.DepartmentID = e.DepartmentID

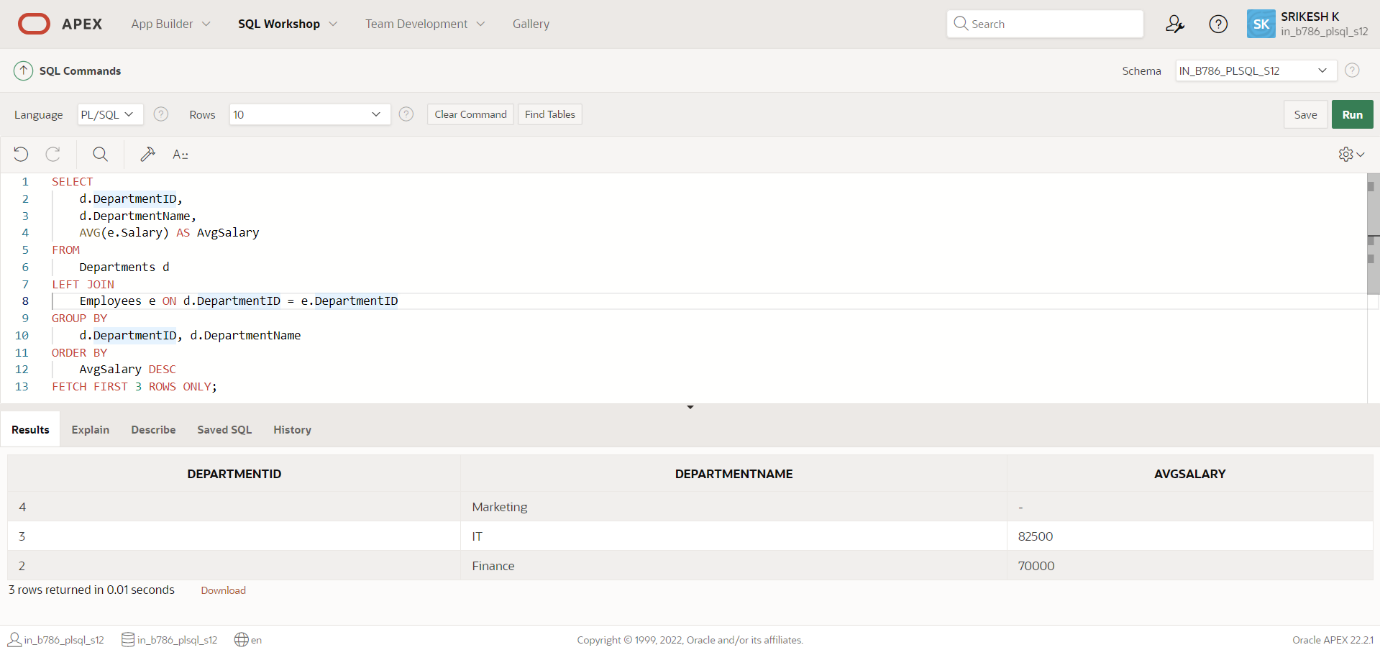
GROUP BY

d.DepartmentID, d.DepartmentName

ORDER BY

AvgSalary DESC

FETCH FIRST 3 ROWS ONLY;



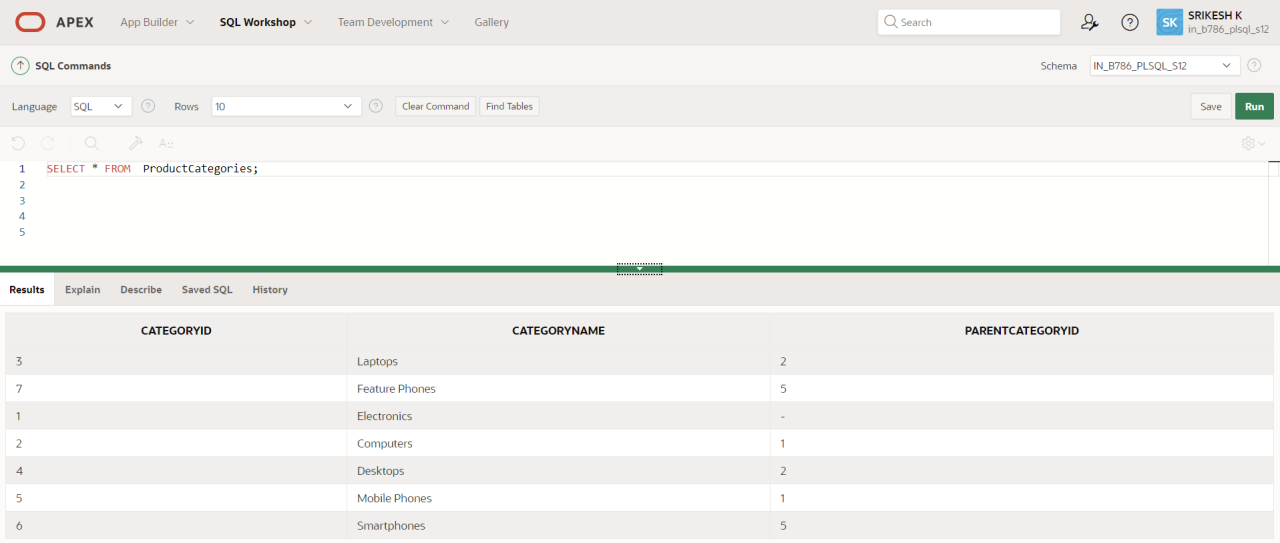
**Explanation:**

1. **Goal**: Find the top 3 departments with the highest average salary of employees, including departments with no employees.
2. **Handling Departments with No Employees**: Use a LEFT JOIN to include all departments, even those with no employees, and the AVG function to calculate the average salary. If a department has no employees, the average salary will be NULL.

**Question 2: Retrieving Hierarchical Category Paths**

**Write a SQL query using recursive Common Table Expressions (CTE) to retrieve all categories along with their full hierarchical path (e.g., Category > Subcategory > Sub-subcategory).**

CREATING TABLE :



QUERY :

SELECT

CategoryID,

CategoryName,

SYS\_CONNECT\_BY\_PATH(CategoryName, ' > ') AS HierarchicalPath

FROM

ProductCategories

START WITH

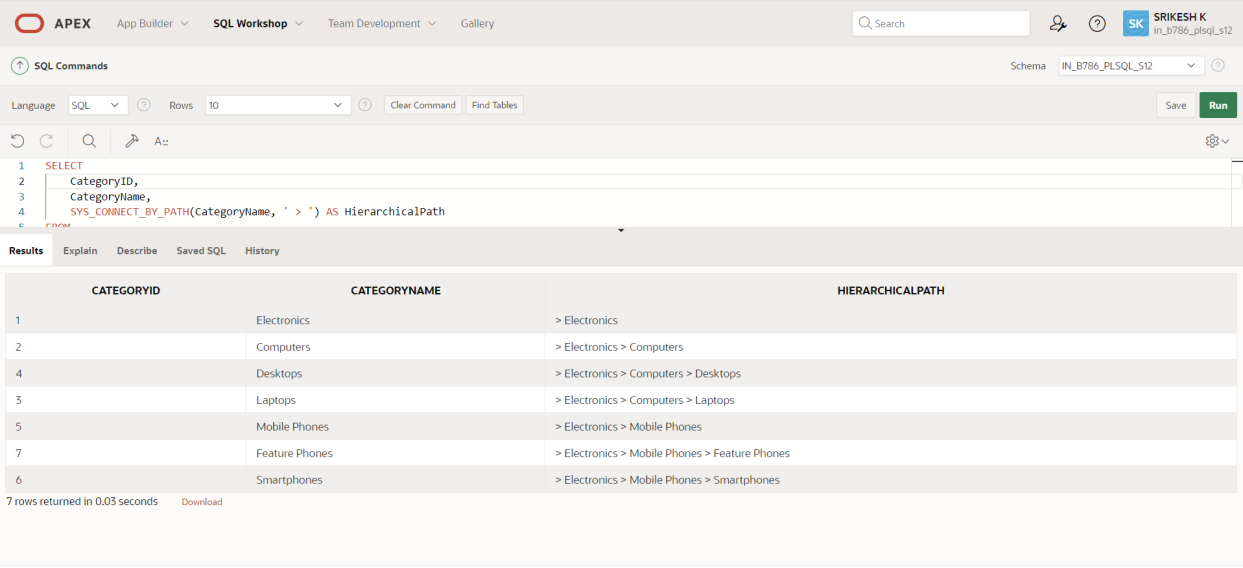
ParentCategoryID IS NULL

CONNECT BY

PRIOR CategoryID = ParentCategoryID

ORDER BY

HierarchicalPath;



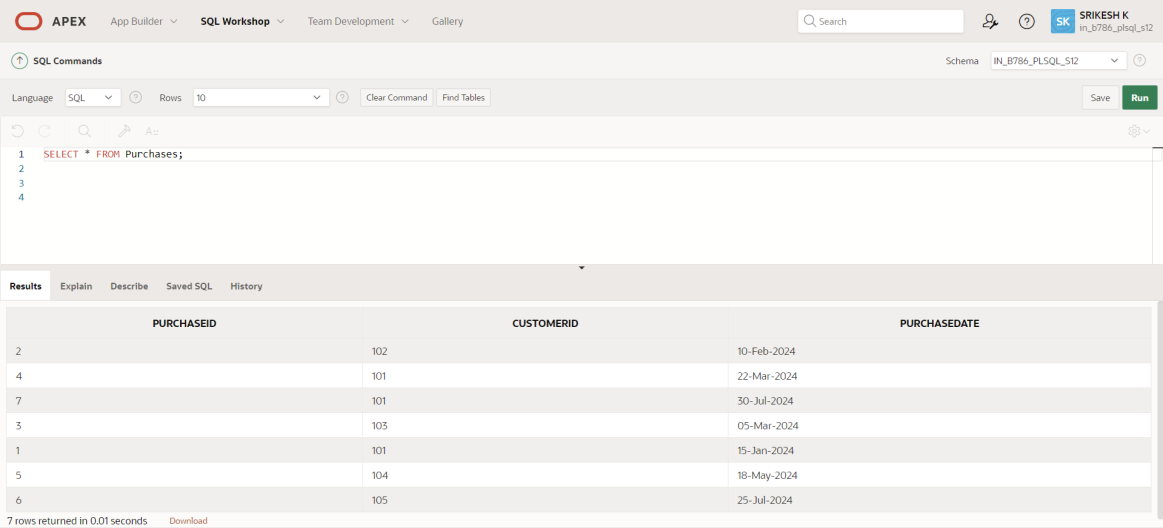
**Explanation:**

1. **Goal**: Retrieve all categories along with their full hierarchical path using a recursive Common Table Expression (CTE).
2. **Recursive CTE**: The CTE starts with the base level categories and recursively joins to retrieve child categories, building the full path for each category.

**Question 3: Total Distinct Customers by Month**

**Design a SQL query to find the total number of distinct customers who made a purchase in each month of the current year. Ensure months with no customer activity show a count of 0.**

CREATING TABLE :



QUERY :

WITH Months AS (

SELECT TO\_CHAR(ADD\_MONTHS(TRUNC(SYSDATE, 'YEAR'), LEVEL - 1), 'Month') AS MonthName,

LEVEL AS MonthNumber

FROM DUAL

CONNECT BY LEVEL <= 12

),

CustomerCounts AS (

SELECT TO\_CHAR(PurchaseDate, 'Month') AS MonthName,

COUNT(DISTINCT CustomerID) AS CustomerCount

FROM Purchases

WHERE EXTRACT(YEAR FROM PurchaseDate) = EXTRACT(YEAR FROM SYSDATE)

GROUP BY TO\_CHAR(PurchaseDate, 'Month'), TO\_CHAR(PurchaseDate, 'MM')

)

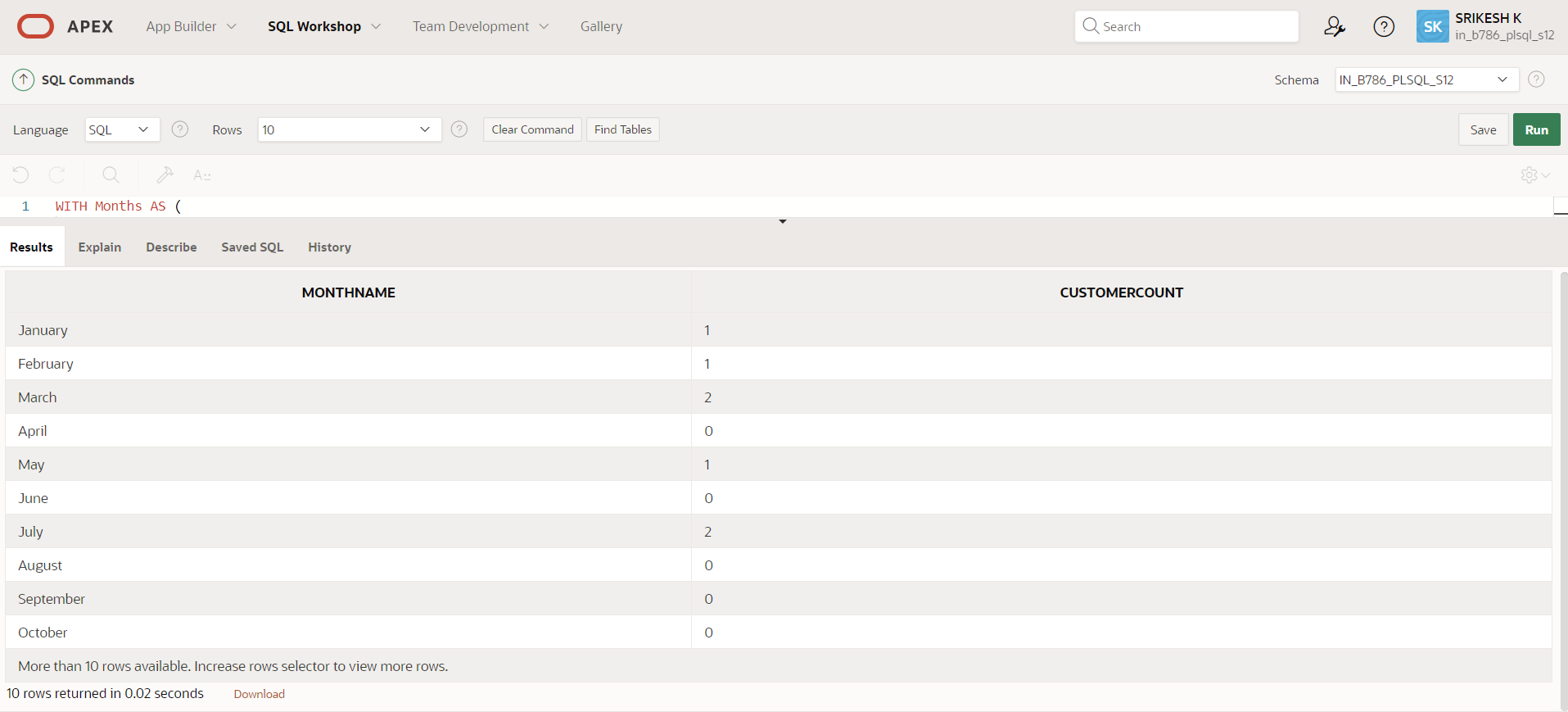
SELECT m.MonthName,

NVL(c.CustomerCount, 0) AS CustomerCount

FROM Months m

LEFT JOIN CustomerCounts c ON m.MonthName = c.MonthName

ORDER BY m.MonthNumber;



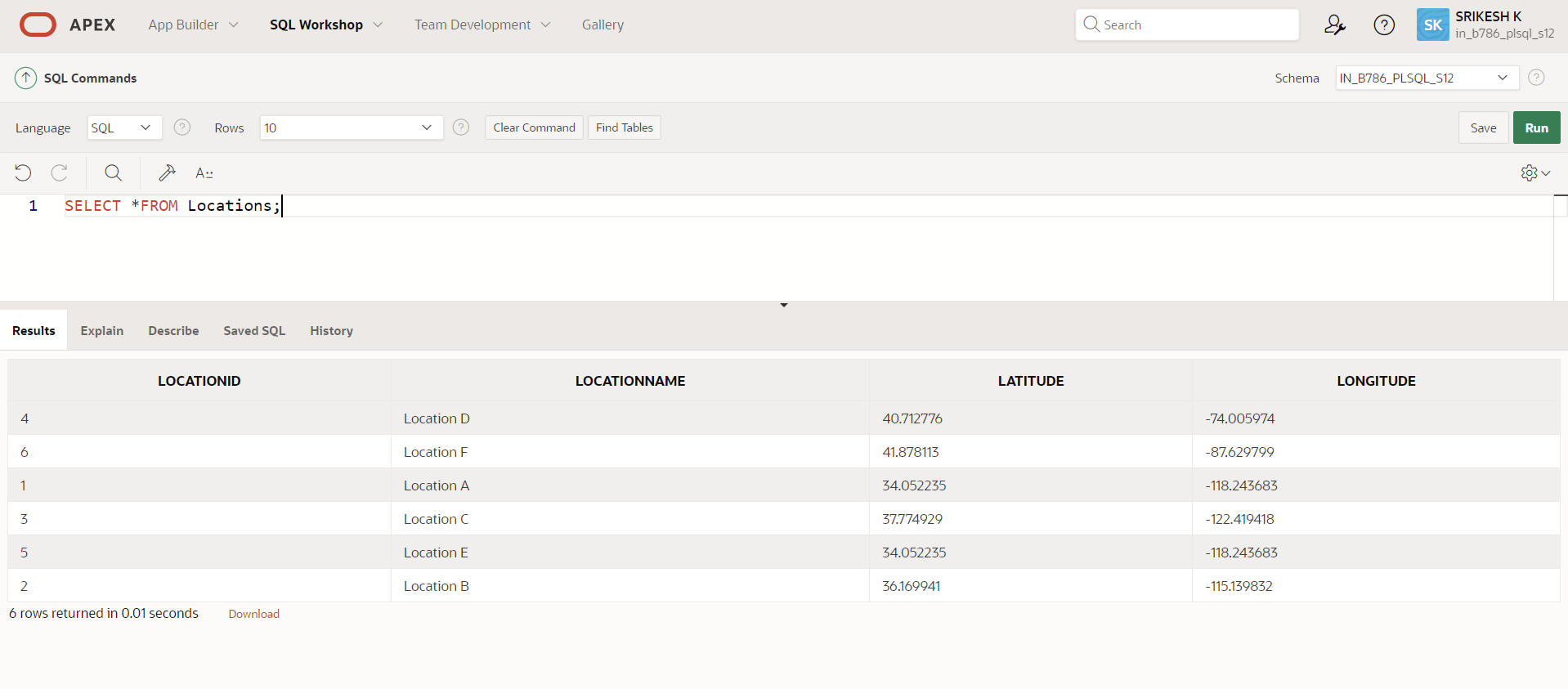
**Explanation:**

1. **Goal**: Find the total number of distinct customers who made a purchase in each month of the current year, including months with no customer activity.
2. **Handling Zero Customer Counts**: Use a LEFT JOIN with a calendar table or a generated series of months to ensure all months are included, and COALESCE to handle zero counts.

**Question 4: Finding Closest Locations**

**Write a SQL query to find the closest 5 locations to a given point specified by latitude and longitude. Use spatial functions or advanced mathematical calculations for proximity.**

CREATING TABLE :



QUERY :

WITH input AS (

SELECT

34.052235 AS latitude,

-118.243683 AS longitude

FROM dual

)

SELECT \*

FROM (

SELECT

loc.LocationID,

loc.LocationName,

loc.Latitude,

loc.Longitude,

(6371 \* ACOS(

LEAST(1, GREATEST(-1,

COS(input.latitude \* (3.141592653589793 / 180)) \*

COS(loc.Latitude \* (3.141592653589793 / 180)) \*

COS(loc.Longitude \* (3.141592653589793 / 180) - input.longitude \* (3.141592653589793 / 180)) +

SIN(input.latitude \* (3.141592653589793 / 180)) \*

SIN(loc.Latitude \* (3.141592653589793 / 180))

))

)) AS Distance

FROM

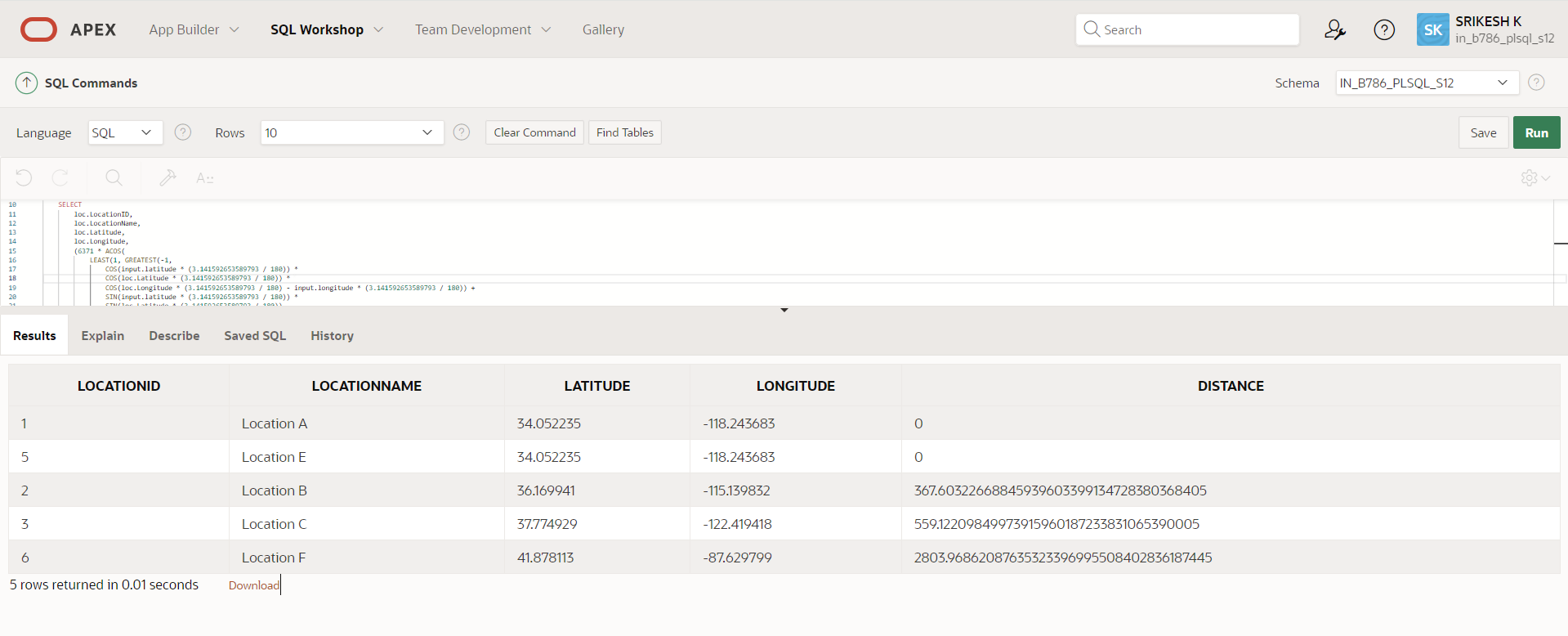
Locations loc, input

ORDER BY

Distance

)

WHERE ROWNUM <= 5;



**Explanation :**

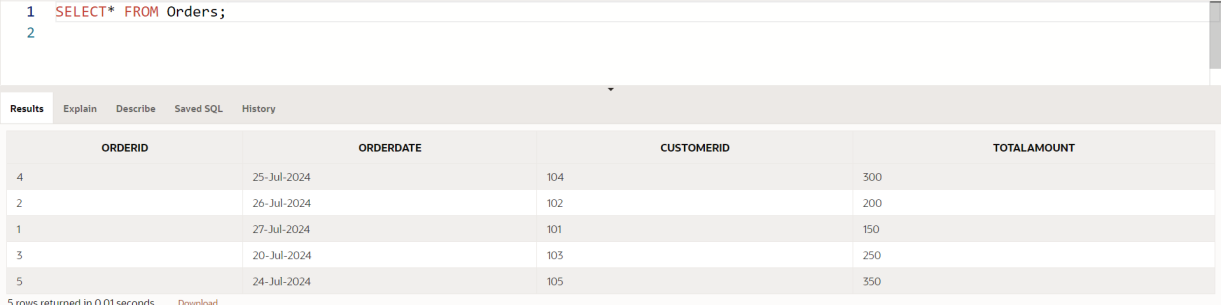
 **Goal**: Find the closest 5 locations to a given point specified by latitude and longitude using spatial functions.

 **Calculating Distance**: Use the Haversine formula to calculate the distance between two points on the Earth’s surface. The formula uses trigonometric functions to determine the shortest distance over the earth’s surface.

**Question 5: Optimizing Query for Orders Table**

**Write a SQL query to retrieve orders placed in the last 7 days from a large Orders table, sorted by order date in descending order.**

CREATING TABLE :



QUERY :

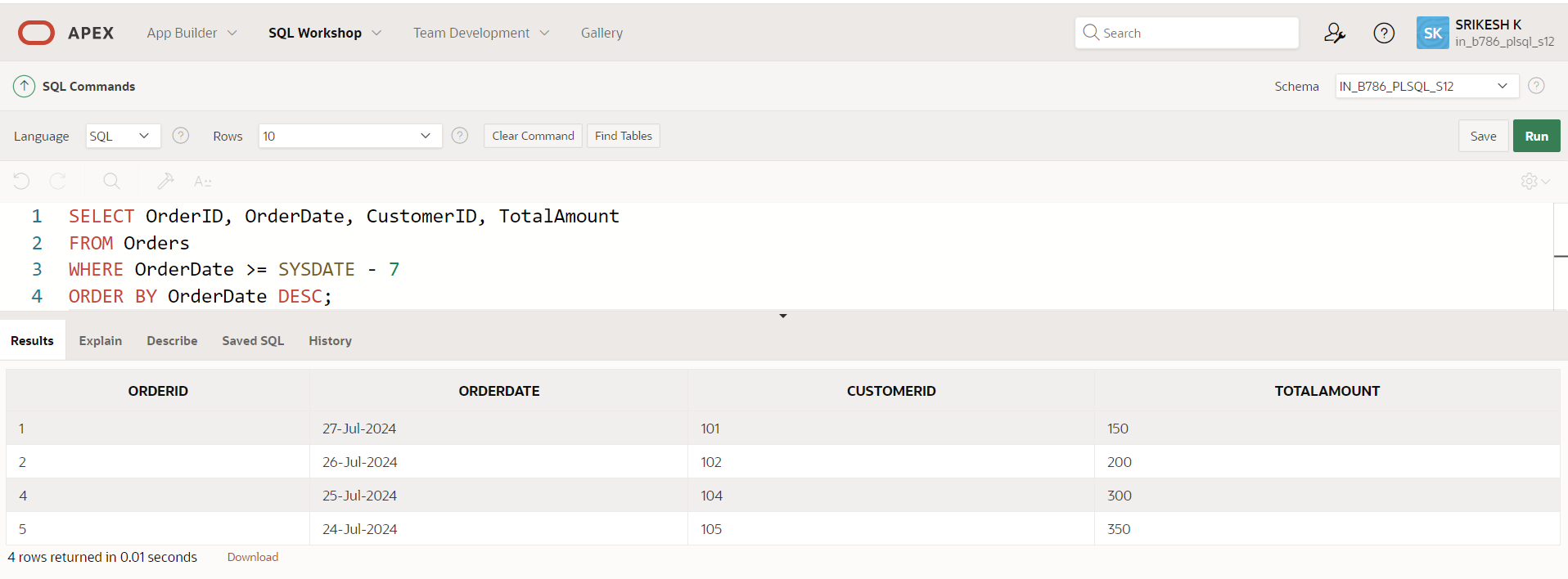
CREATE INDEX idx\_orderdate ON Orders(OrderDate);

SELECT OrderID, OrderDate, CustomerID, TotalAmount

FROM Orders

WHERE OrderDate >= SYSDATE - 7

ORDER BY OrderDate DESC;



**Explanation:**

1. **Goal**: Retrieve orders placed in the last 7 days from a large Orders table, sorted by order date in descending order.
2. **Optimization Strategies**:
   * **Indexing**: Create an index on the OrderDate column to speed up retrieval.
   * **Efficient Query Structure**: Use SYSDATE - 7 for date filtering and ensure the use of the index.