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

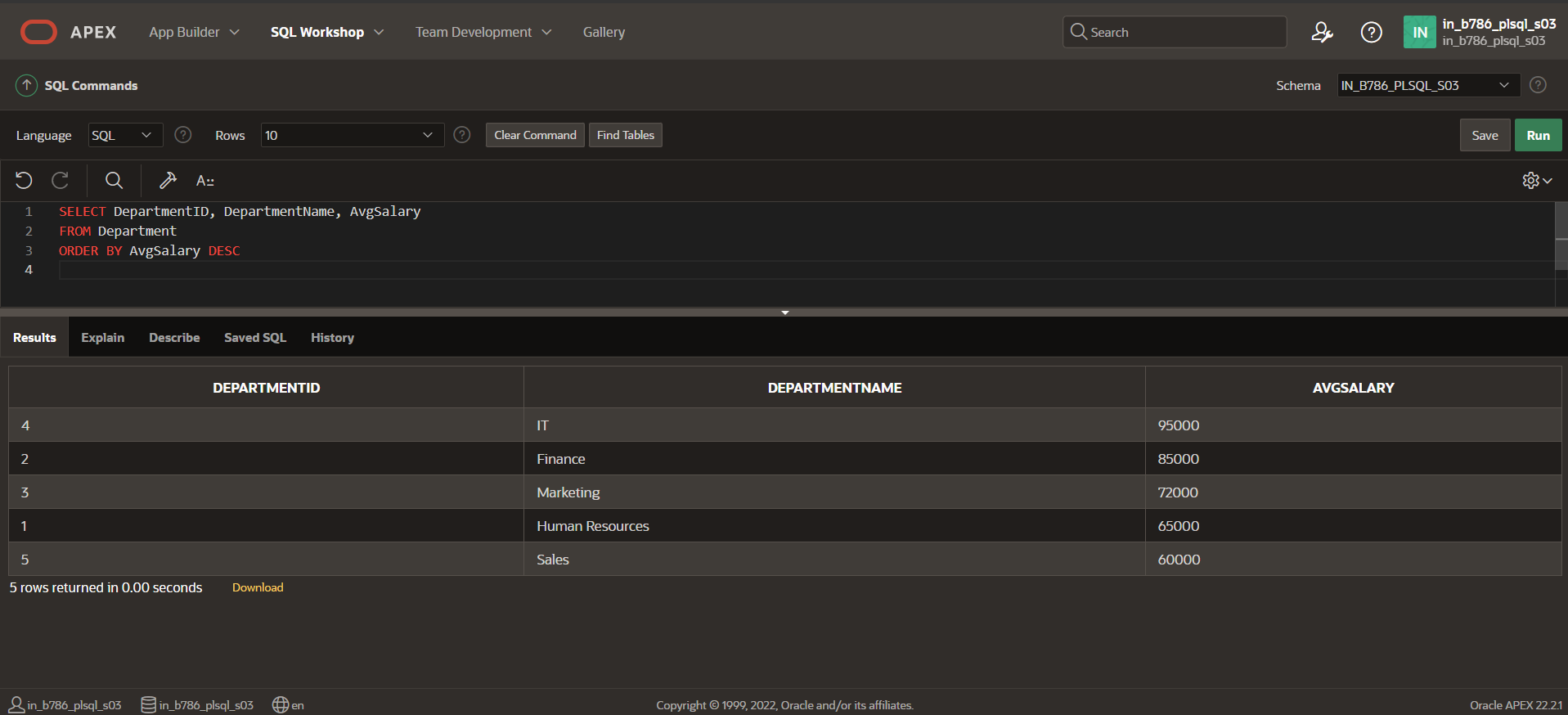
**Name : P.Pranesh**

**Reg number :192311003**

**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

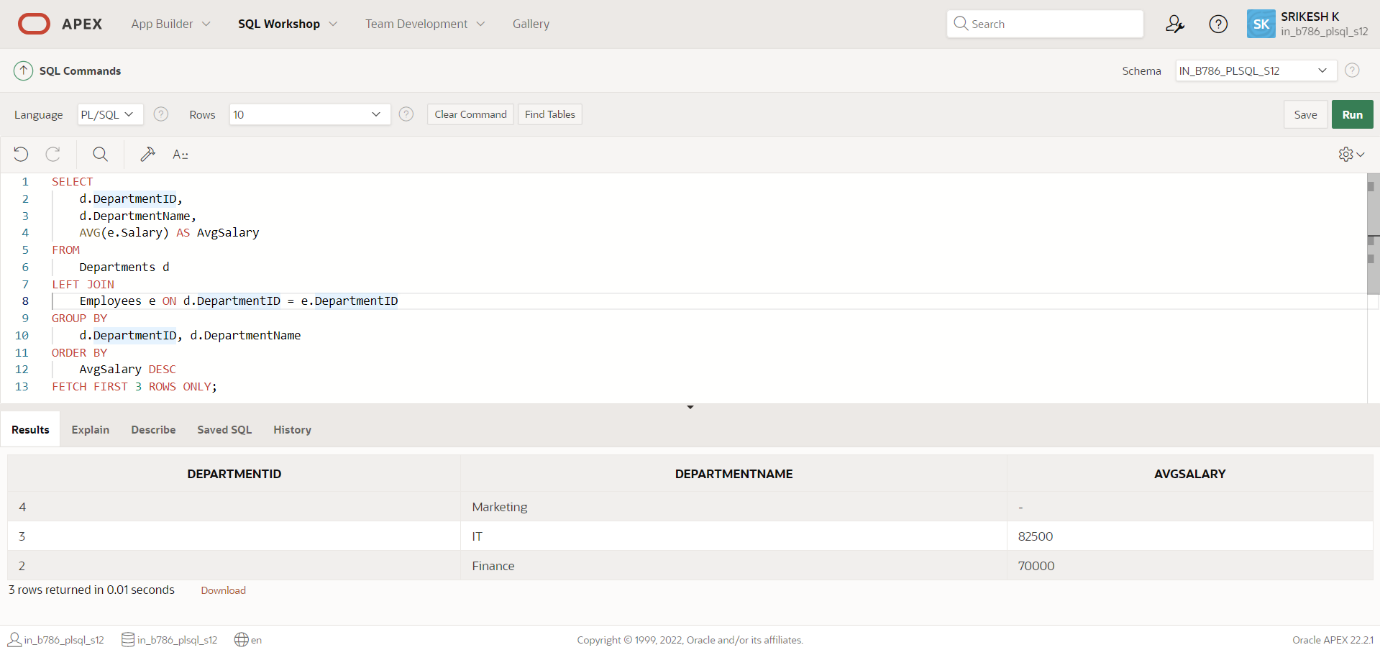
GROUP BY

d.DepartmentID, d.DepartmentName

ORDER BY

AvgSalary DESC

FETCH FIRST 3 ROWS ONLY;

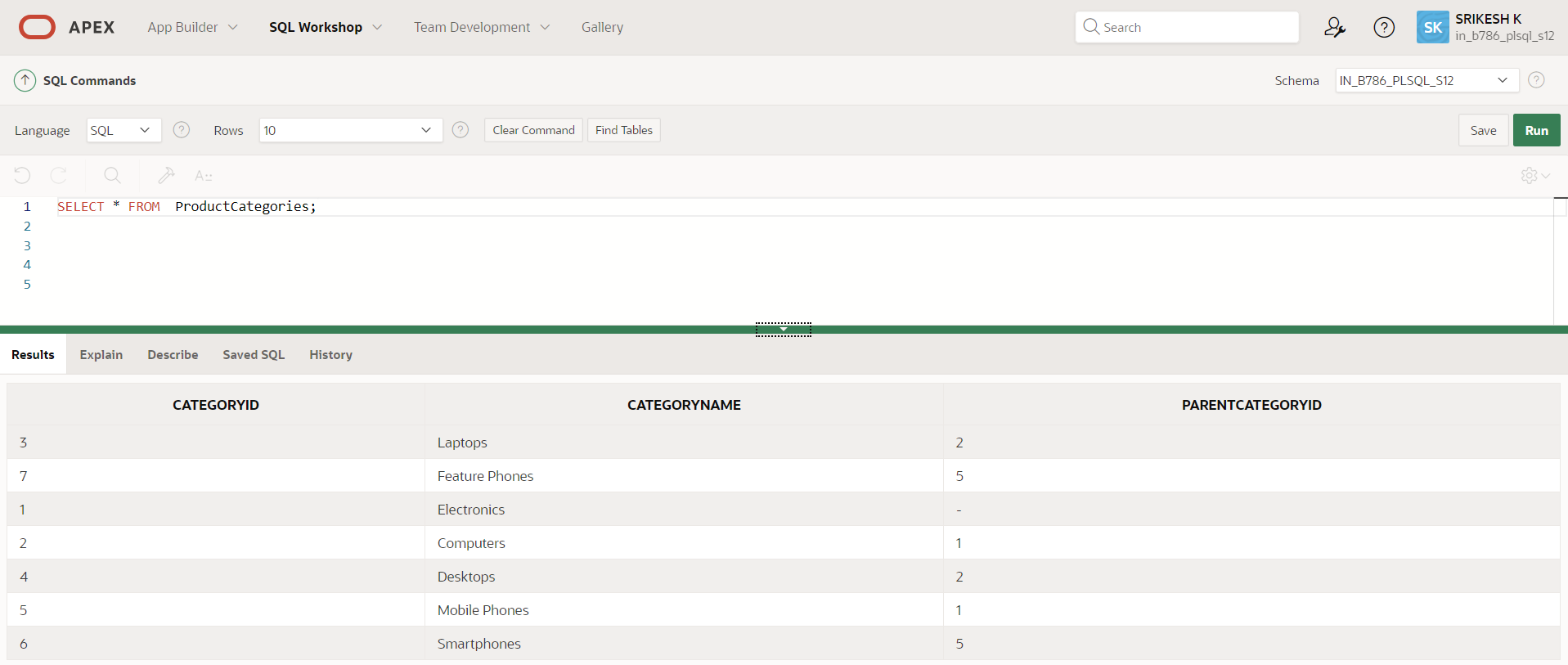


**Explanation**:- The output of this query will be a list of up to three departments, each with their ID, name, and the average salary of its employees. The list will be sorted so that the department with the highest average salary is first. If some departments have no employees, their average salary will be calculated as NULL, and they may still appear in the result set if they fall within the top three departments by average salary.

**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

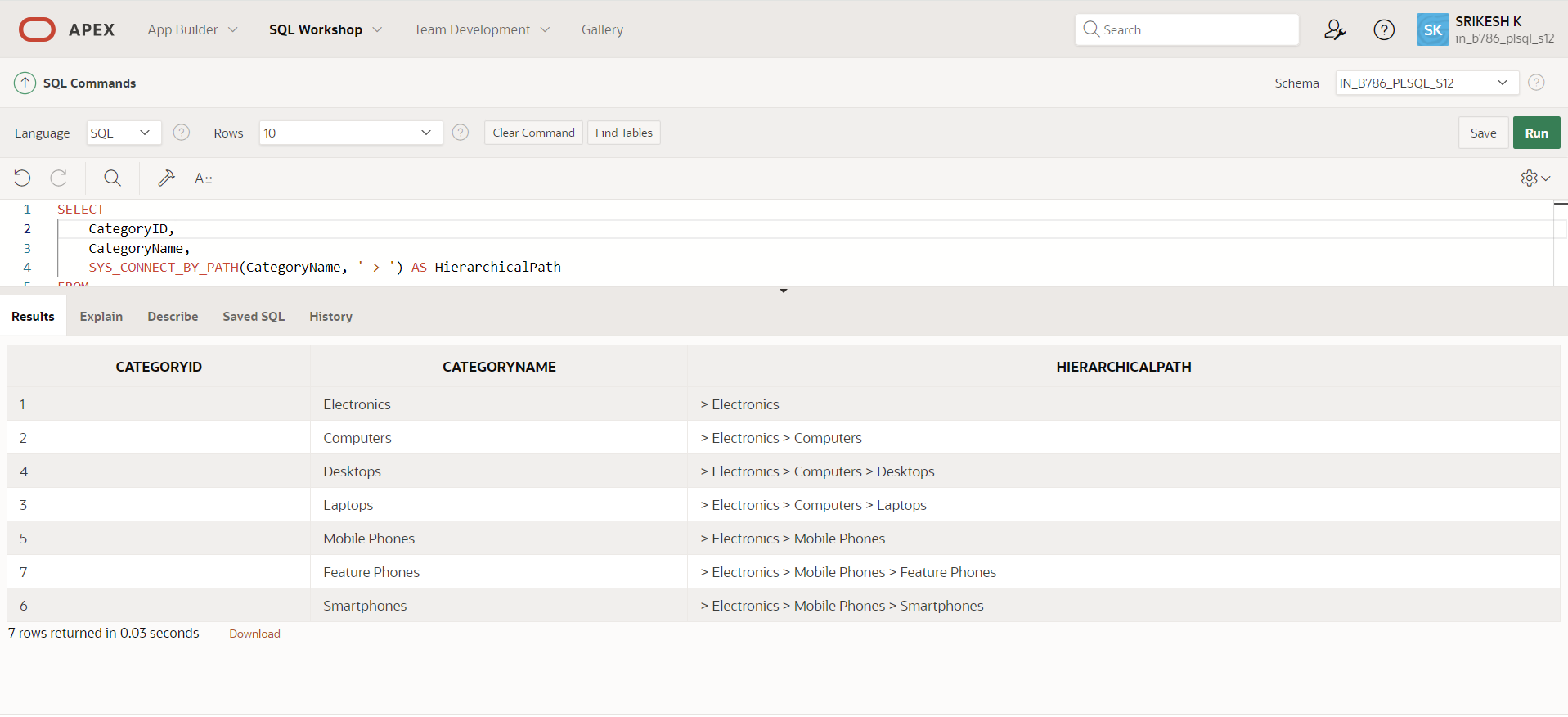
ParentCategoryID IS NULL

CONNECT BY

PRIOR CategoryID = ParentCategoryID

ORDER BY

HierarchicalPath;



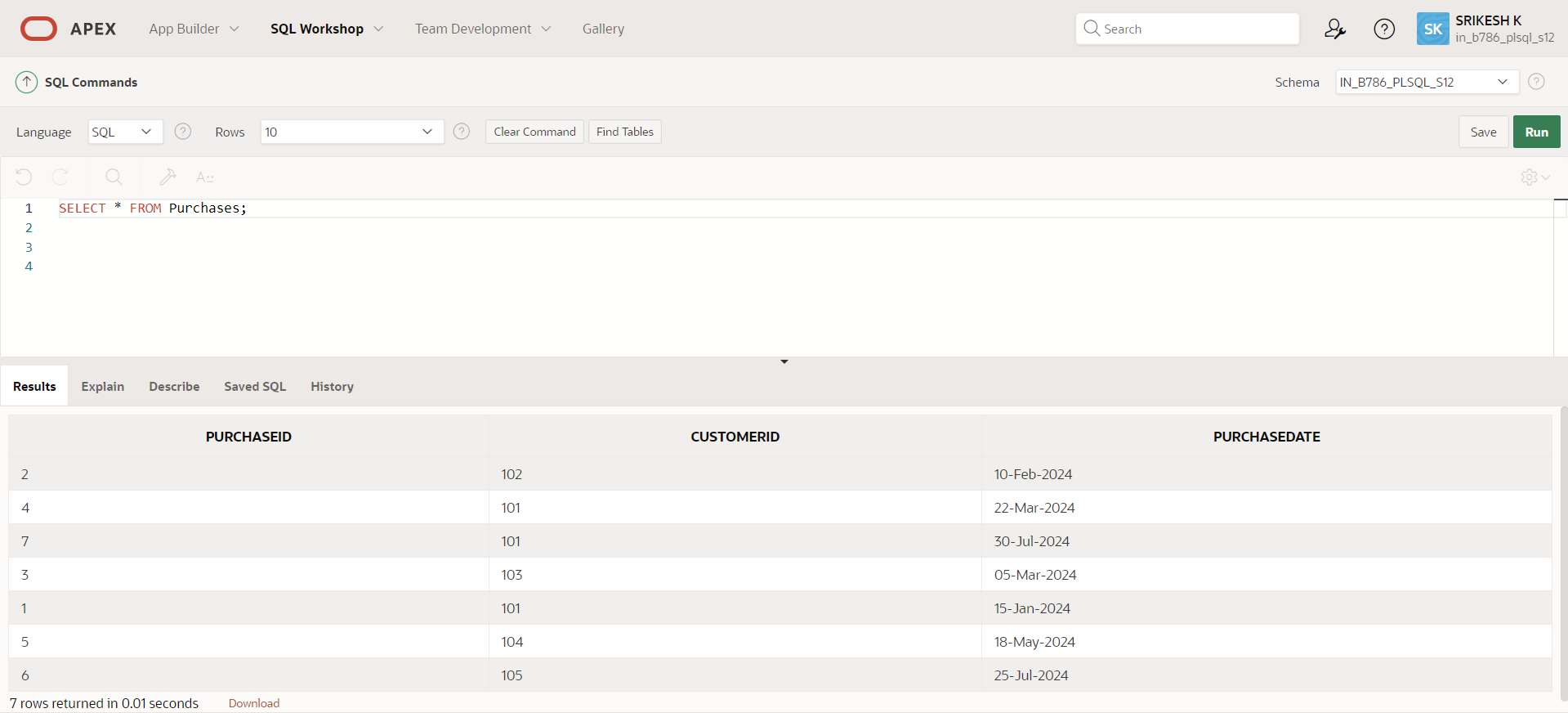
**Explanation:-**

The output of this query will be a list of categories, each with its ID, name, and the hierarchical path from the root category to the current category. The hierarchical path is built using the SYS\_CONNECT\_BY\_PATH function.

**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')

)

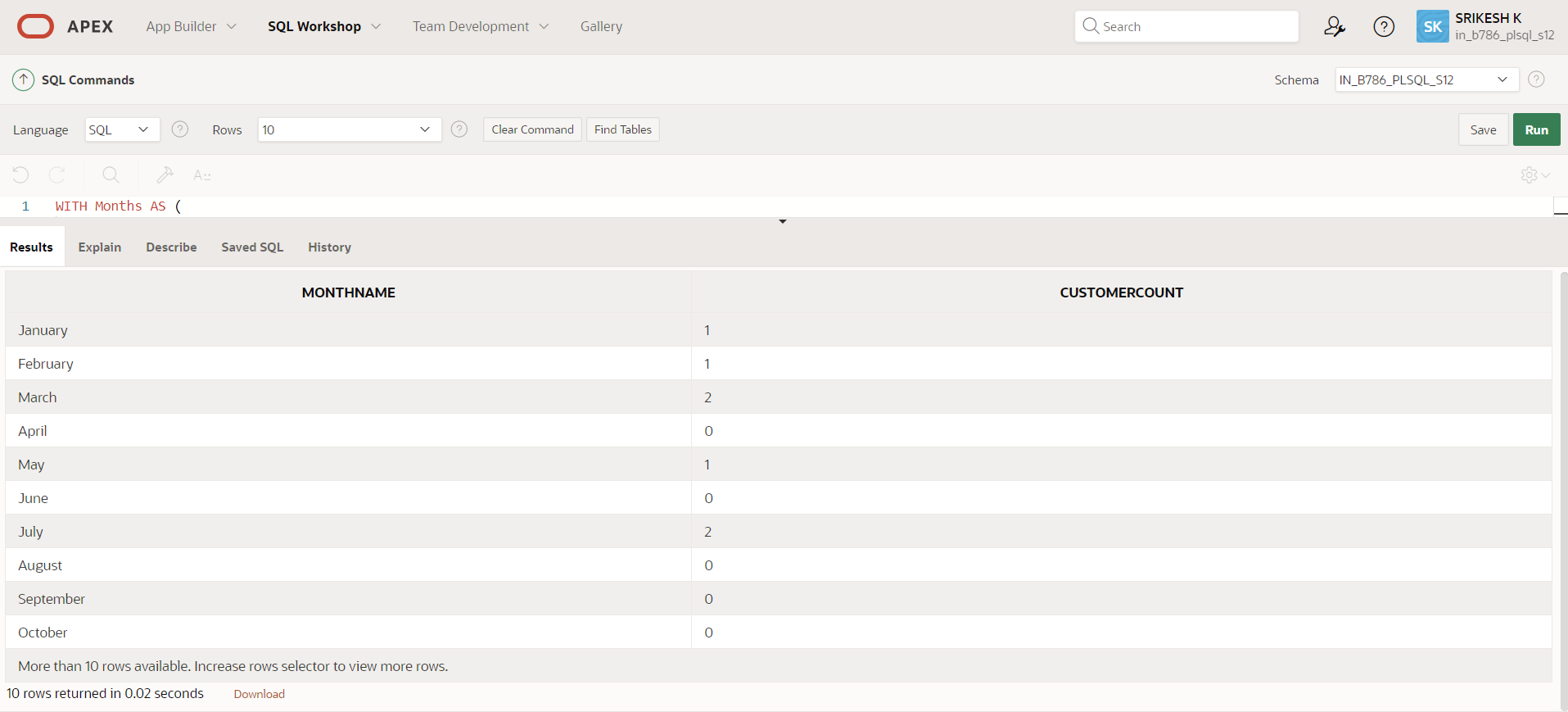
SELECT m.MonthName,

NVL(c.CustomerCount, 0) AS CustomerCount

FROM Months m

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

ORDER BY m.MonthNumber;

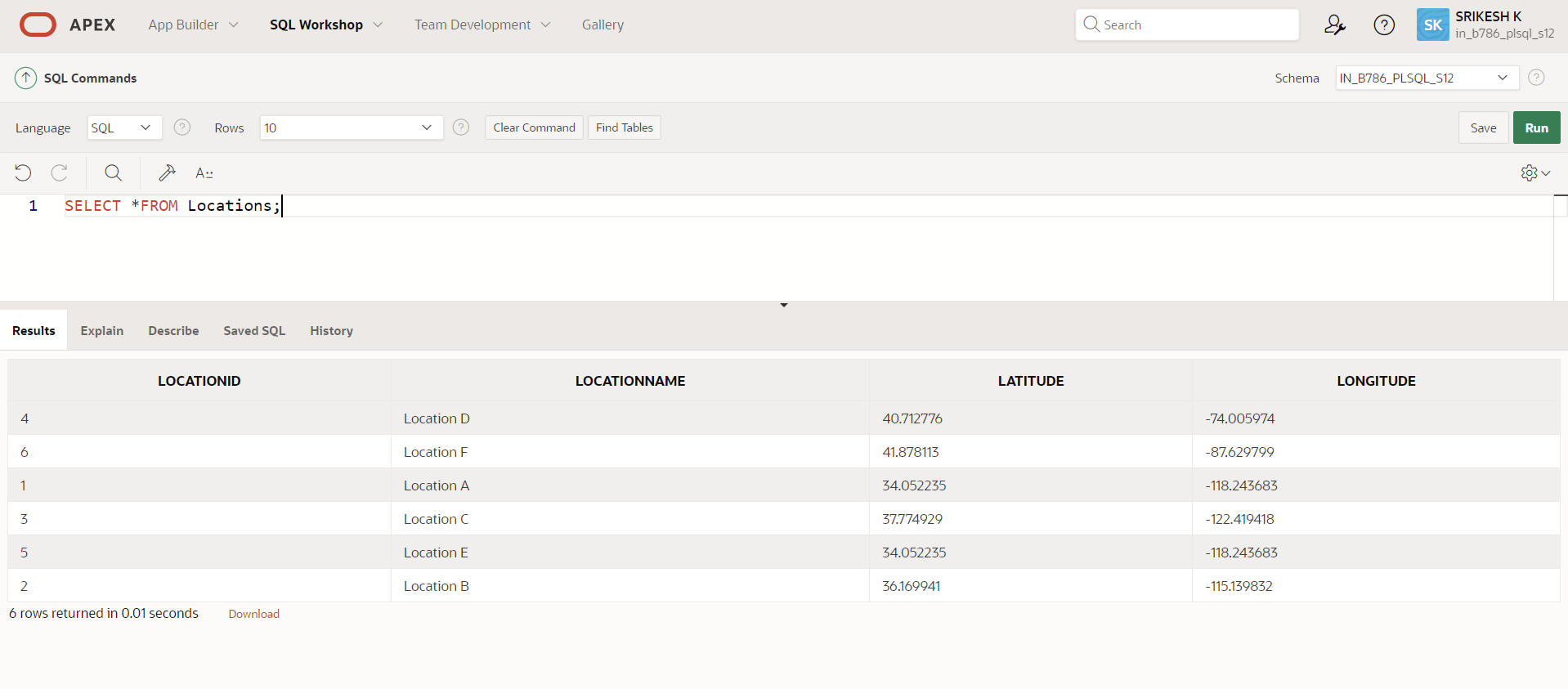


**Explanation:-** The output of this query will be a list of months, each with the month name and the corresponding customer count. Months with no customer count data will display a count of 0.

**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

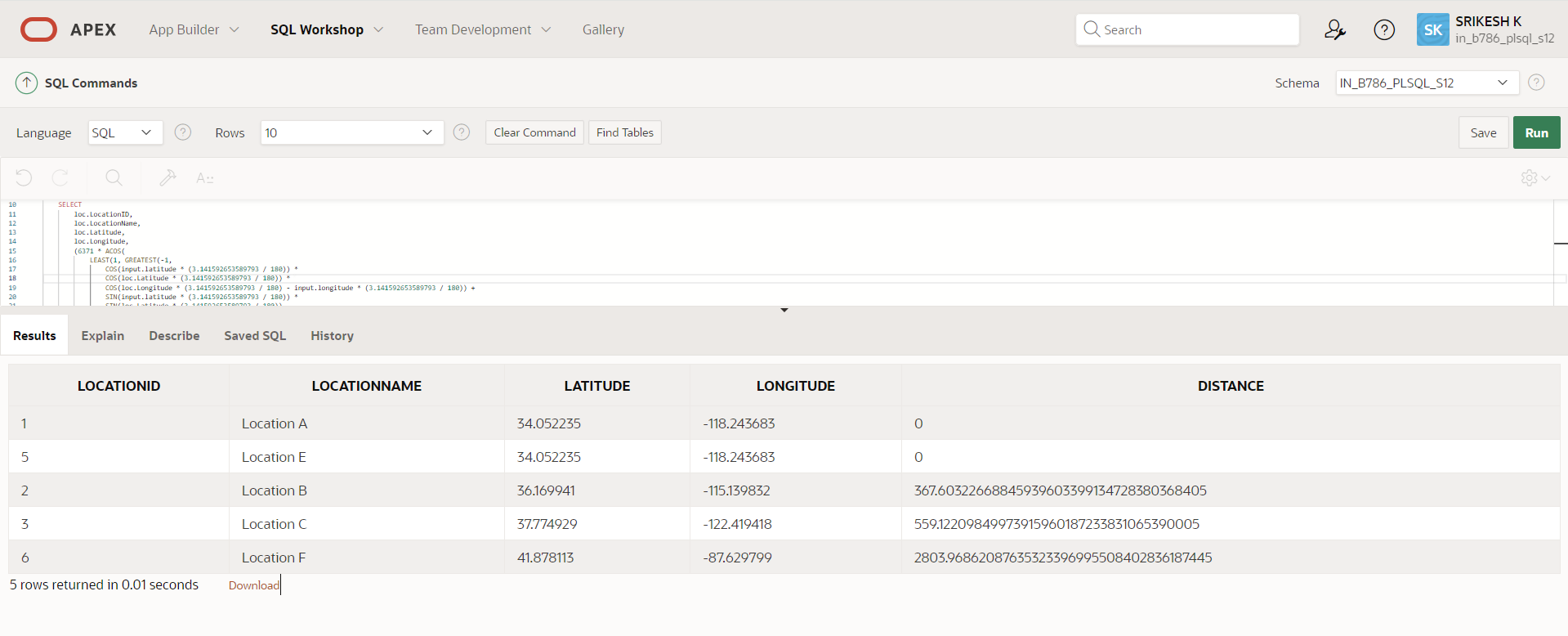
Locations loc, input

ORDER BY

Distance

)

WHERE ROWNUM <= 5;

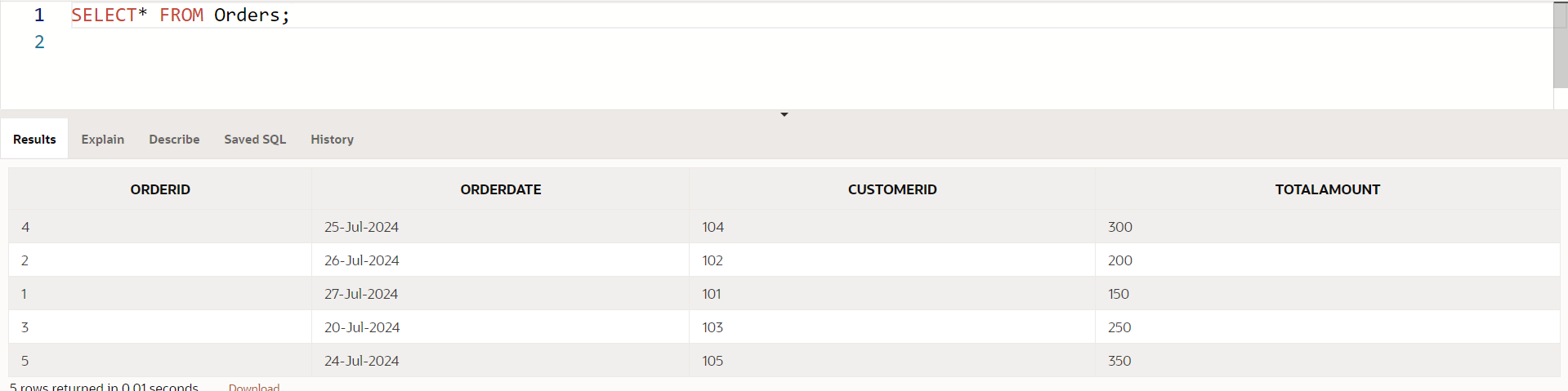


**Explanation:-** The output of this query will be the five closest locations to the given coordinates (Los Angeles). Each row will include the location ID, name, latitude, longitude, and the calculated distance from the input coordinates.

**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 :

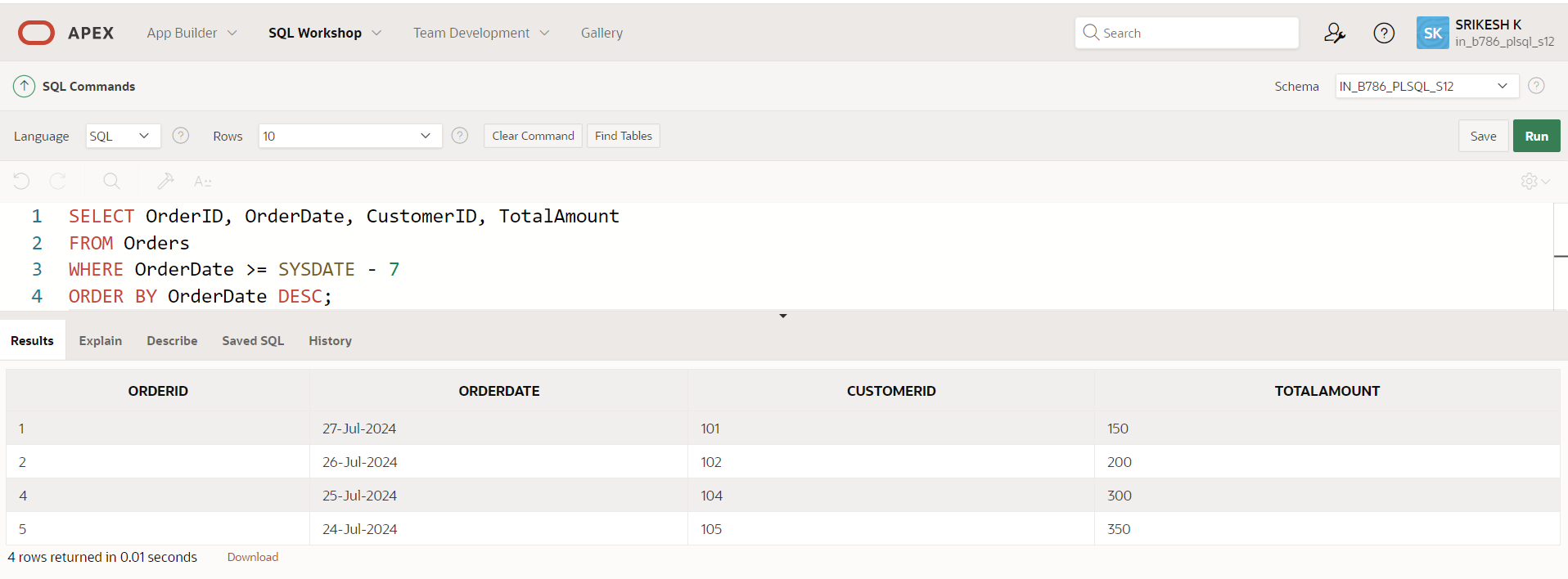
CREATE INDEX idx\_orderdate ON Orders(OrderDate);

SELECT OrderID, OrderDate, CustomerID, TotalAmount

FROM Orders

WHERE OrderDate >= SYSDATE - 7

ORDER BY OrderDate DESC;



Explanation:-

The output of this query will be a list of orders placed in the last 7 days, with each row containing the order ID, order date, customer ID, and total amount, sorted by order date in descending order.