CSA09: DATABASE MANAGEMENT SYSTEMS-ASSIGNMENT QUESTIONS

BY: DHARSHINI J 192324059

Question 1:

ER Diagram Question: Traffic Flow Management System (TFMS)

You are tasked with designing an Entity-Relationship (ER) diagram for a Traffic Flow Management System (TFMS) used in a city to optimize traffic routes, manage intersections, and control traffic signals. The TFMS aims to enhance transportation efficiency by utilizing real-time data from sensors and historical traffic patterns.

Entities and Attributes: (TASK 1)

1.Roads

- **Road ID**
- **Road Name**
- **Length**
- **Speed Limit**

2.Intersection

- **Intersection ID**
- **Intersection Name**
- **Latitude**
- **Longitude**

3.Traffic Signals

- **Signal ID**
- **Intersection ID**
- **Signal status**
- **Timer**

4.Traffic Data

- **Traffic Data ID**
- **Road ID**
- **Timestamp**
- **Speed**
- **Congestion level**

Relationship Modelling: (Task 2)

1.Roads to Intersection:

- **A road can connect to multiple intersections (one to many).**
- **An intersection can be connected to multiple roads (many to many).**

2. Intersection to Traffic Signals:

- **An Intersection can host multiple traffic signals (one to many).**
- **A Traffic Signal is installed at a single intersection (many to one).**

3.Roads to Traffic Data:

- **A road can have multiple traffic data (one to many).**
- **A traffic data record is associated with a single road (many to one).**

4.Roads to Traffic Lights:

- **A road can have many traffic lights(one to many).**
- **A traffic light is installed at a single road(one to many).**

ER Diagram (Task 3):

Design Justification:

- **Scalability**
- **Real Time Data Processing**
- **Efficient Traffic Management**

Justification and Normalization: (Task 4)

1.First Normal Form (1NF)

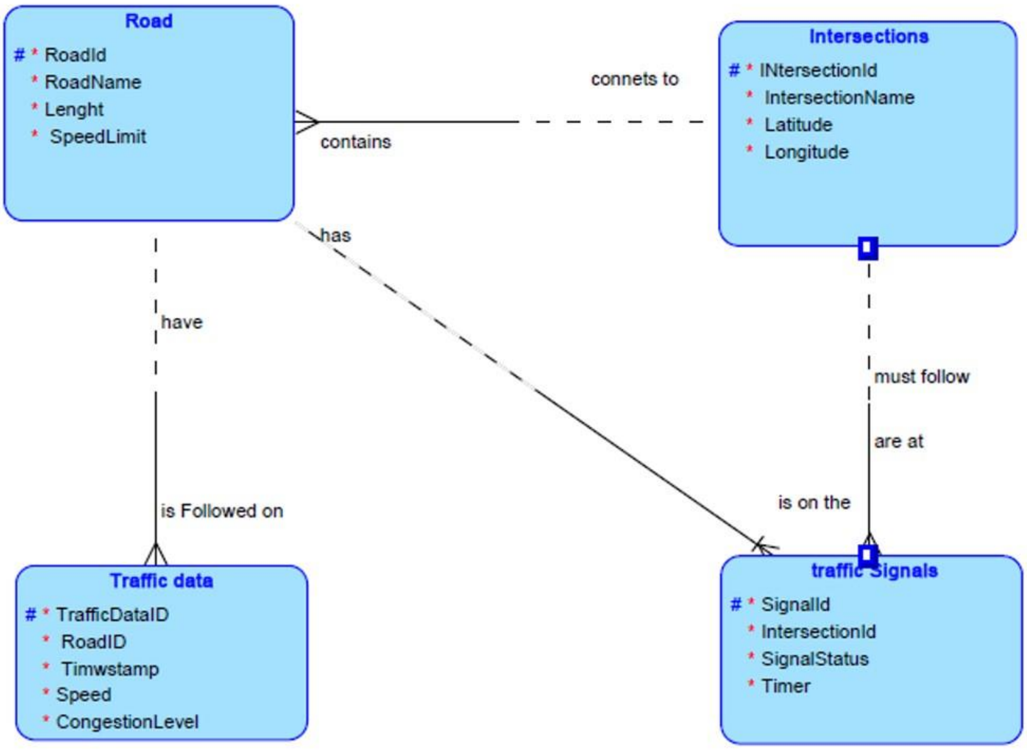
- **Ensure each table has atomic columns and there is no repeating groups**

2. Second Normal Form (2NF):

- **Ensure all non-key attributes are fully functional dependant on the primary key.**

3. Third Normal Form(3NF):

- **Ensure no transitive dependencies exists between non-key attributes**


Road
# * RoadId
* RoadName
* Lenght
* SpeedLimit
contains
connets to
Intersections
# * INtersectionId
* IntersectionName
* Latitude
* Longitude
has
have
must follow
are at
is on the
is Followed on
Traffic data
# * TrafficDataID
* RoadID
* Timwstamp
* Speed
* CongestionLevel
traffic Signals
# * SignalId
* IntersectionId
* SignalStatus
* Timer

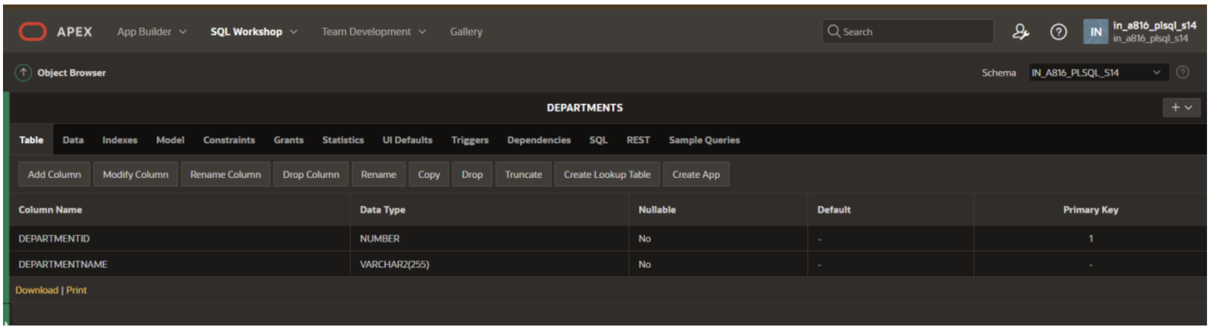
Question 2:

Question 1: Top 3 Departments with Highest Average Salary

Task:

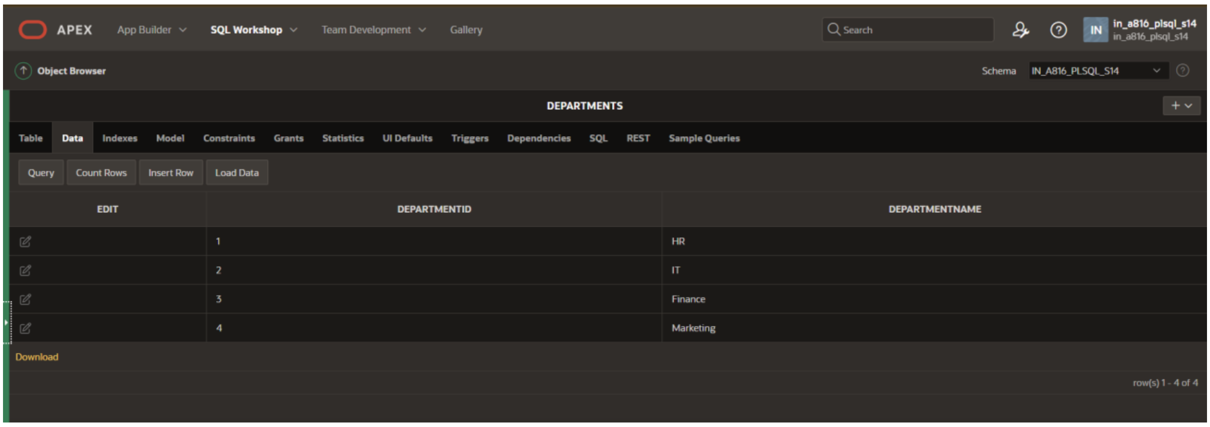
Write a SQL query to find the top 3 departments with the highest average salary of employees. Ensure departments with no employees show an average salary of NULL.

```
CREATE TABLE Departments (

    DepartmentID INT PRIMARY KEY,

    DepartmentName VARCHAR(255) NOT NULL

);
```



Column Name	Data Type	Nullable	Default	Primary Key
DEPARTMENTID	NUMBER	No	-	1
DEPARTMENTNAME	VARCHAR2(255)	No	-	-

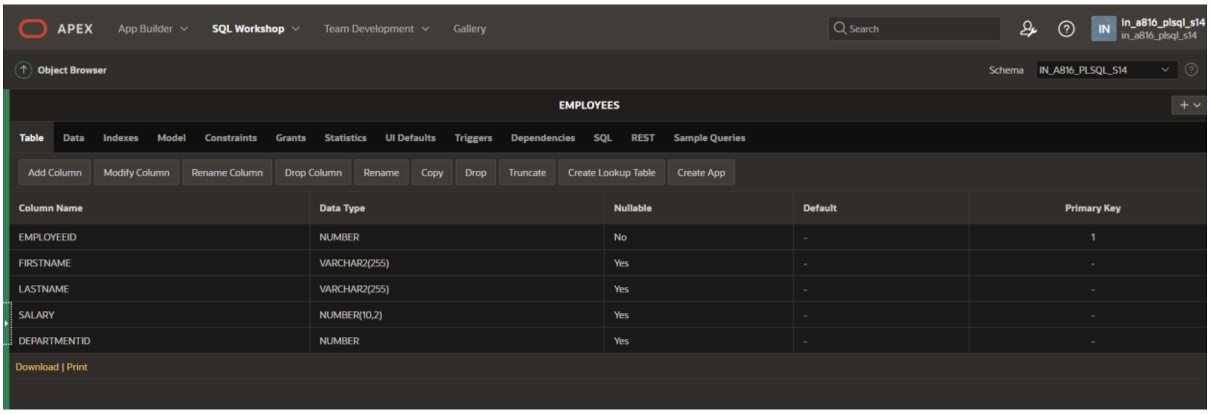
```
INSERT INTO Departments   VALUES (1, 'HR');

INSERT INTO Departments  VALUES (2, 'IT');

INSERT INTO Departments   VALUES (3, 'Finance');

 INSERT INTO Departments   VALUES (4, 'Marketing');
```



EDIT	DEPARTMENTID	DEPARTMENTNAME
✎	1	HR
✎	2	IT
✎	3	Finance
✎	4	Marketing

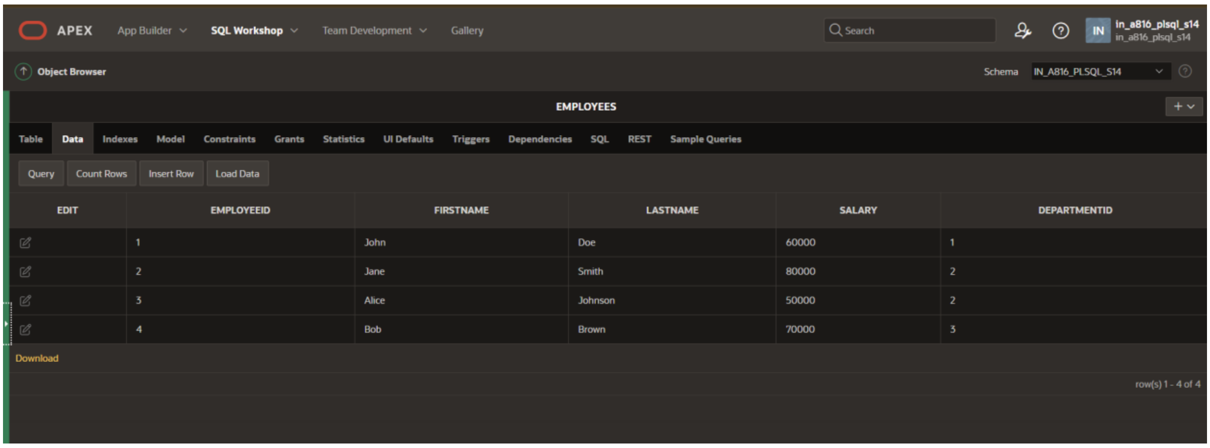
row(s) 1 - 4 of 4

```
CREATE TABLE Employees (

    EmployeeID INT PRIMARY KEY,

    FirstName VARCHAR(255),

    LastName VARCHAR(255),

    Salary DECIMAL(10, 2),

    DepartmentID INT,

    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)

);
```



Column Name	Data Type	Nullable	Default	Primary Key
EMPLOYEEID	NUMBER	No	-	1
FIRSTNAME	VARCHAR2(255)	Yes	-	-
LASTNAME	VARCHAR2(255)	Yes	-	-
SALARY	NUMBER(10,2)	Yes	-	-
DEPARTMENTID	NUMBER	Yes	-	-

```
INSERT INTO Employees  VALUES (1, 'John', 'Doe', 60000, 1);

INSERT INTO Employees  VALUES (2, 'Jane', 'Smith', 80000, 2);

INSERT INTO Employees  VALUES (3, 'Alice', 'Johnson', 50000, 2);

INSERT INTO Employees  VALUES (4, 'Bob', 'Brown', 70000, 3);
```

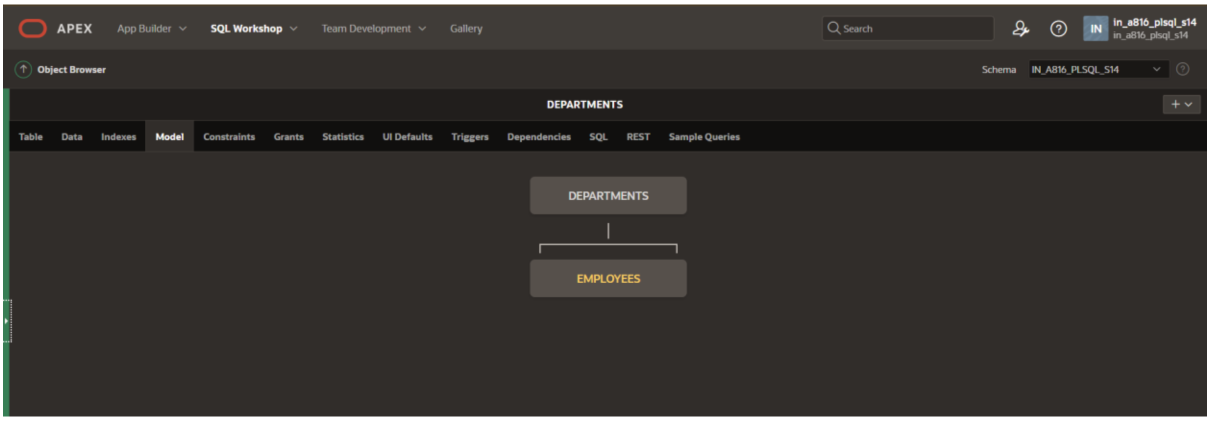


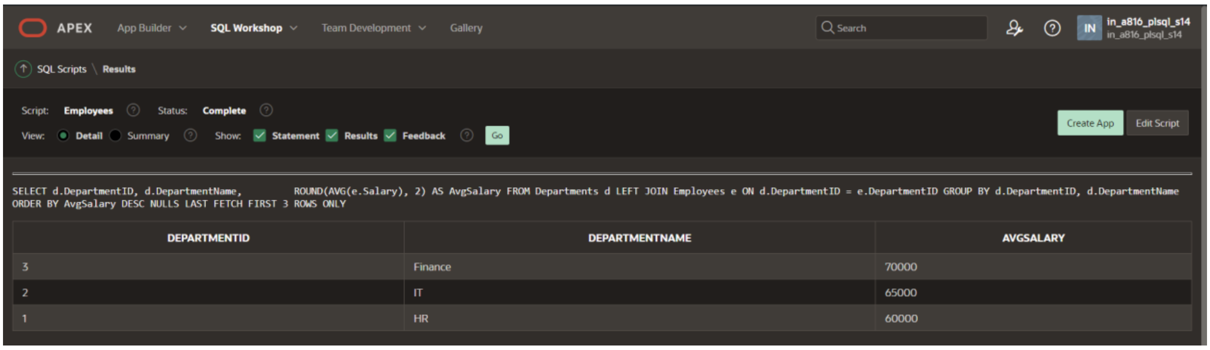
EDIT	EMPLOYEEID	FIRSTNAME	LASTNAME	SALARY	DEPARTMENTID
	1	John	Doe	60000	1
	2	Jane	Smith	80000	2
	3	Alice	Johnson	50000	2
	4	Bob	Brown	70000	3

row(s) 1 - 4 of 4

```
SELECT d.DepartmentID, d.DepartmentName,

 ROUND(AVG(e.Salary), 2) 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;
```





Script: Employees Status: Complete

View: ● Detail ● Summary Show: ☑ Statement ☑ Results ☑ Feedback

```
SELECT d.DepartmentID, d.DepartmentName,        ROUND(AVG(e.Salary), 2) AS AvgSalary FROM Departments d LEFT JOIN Employees e ON d.DepartmentID = e.DepartmentID GROUP BY d.DepartmentID, d.DepartmentName
ORDER BY AvgSalary DESC NULLS LAST FETCH FIRST 3 ROWS ONLY
```

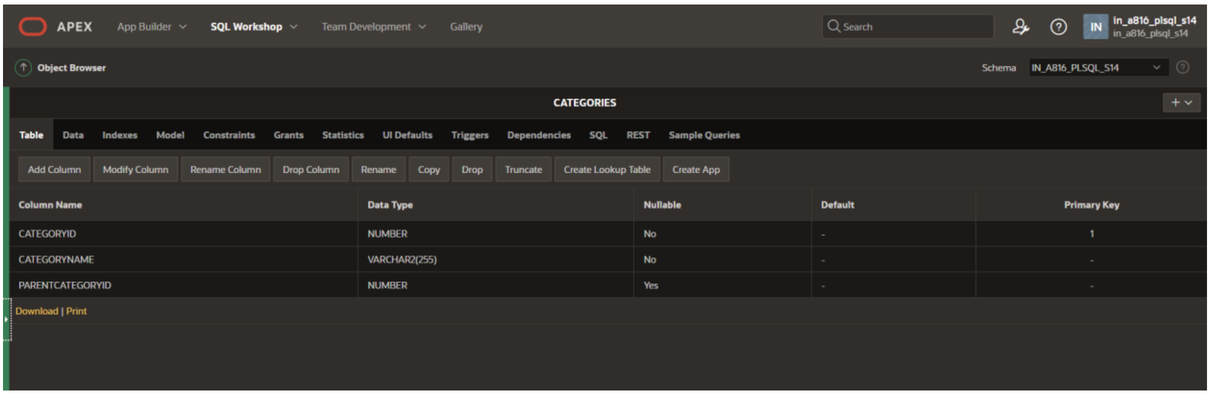
DEPARTMENTID	DEPARTMENTNAME	AVGSALARY
3	Finance	70000
2	IT	65000
1	HR	60000

Question 2: Retrieving Hierarchical Category Paths

Task:

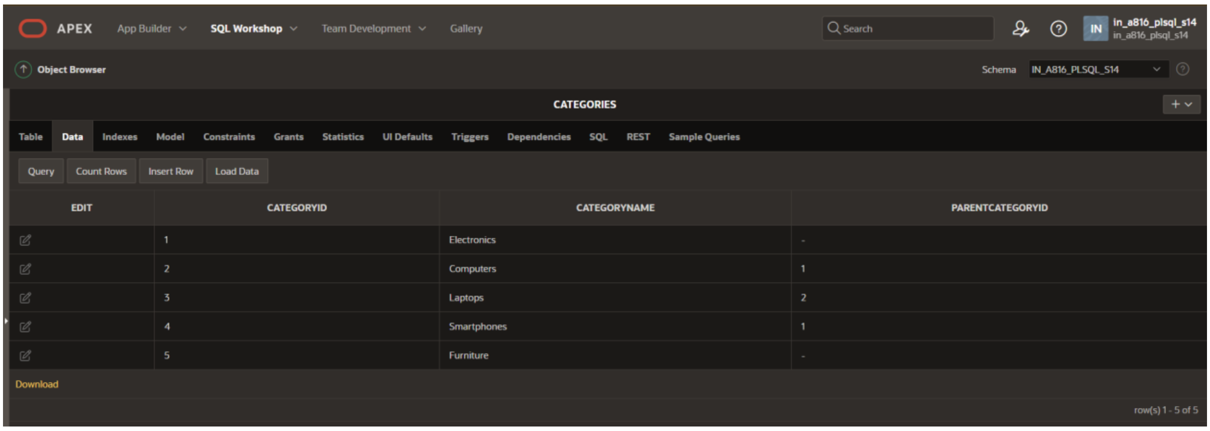
Write a SQL query using recursive Common Table Expressions (CTE) to retrieve all categories along with their full hierarchical path (e.g., Category > Subcategory > Sub-subcategory).

```
CREATE TABLE Categories (
   CategoryID INT PRIMARY KEY,
   CategoryName VARCHAR(255) NOT NULL,
   ParentCategoryID INT,
   FOREIGN KEY (ParentCategoryID) REFERENCES Categories(CategoryID)
);
```



Column Name	Data Type	Nullable	Default	Primary Key
CATEGORYID	NUMBER	No	-	1
CATEGORYNAME	VARCHAR2(255)	No	-	-
PARENTCATEGORYID	NUMBER	Yes	-	-

```
INSERT INTO Categories  VALUES (1, 'Electronics', NULL);

INSERT INTO Categories  VALUES (2, 'Computers', 1);

INSERT INTO Categories  VALUES (3, 'Laptops', 2);

INSERT INTO Categories  VALUES (4, 'Smartphones', 1);

INSERT INTO Categories  VALUES (5, 'Furniture', NULL);
```

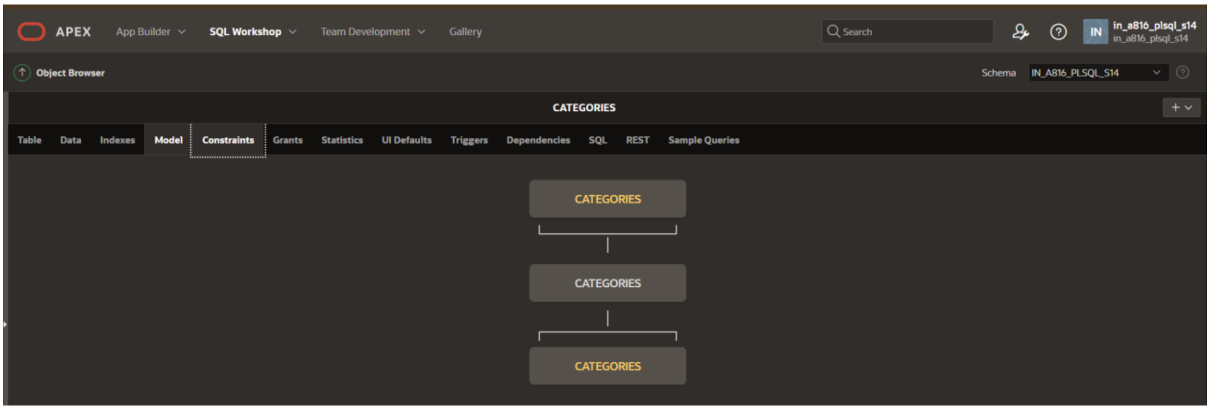


EDIT	CATEGORYID	CATEGORYNAME	PARENTCATEGORYID
✎	1	Electronics	-
✎	2	Computers	1
✎	3	Laptops	2
✎	4	Smartphones	1
✎	5	Furniture	-

row(s) 1 - 5 of 5

```
WITH CategoryPath (CategoryID, CategoryName, ParentCategoryID, Path) AS (
   SELECT CategoryID, CategoryName, ParentCategoryID,
         CategoryName AS Path
   FROM Categories
   WHERE ParentCategoryID IS NULL
   UNION ALL
   SELECT c.CategoryID, c.CategoryName, c.ParentCategoryID,
         cp.Path || ' > ' || c.CategoryName AS Path
   FROM Categories c
   JOIN CategoryPath cp ON c.ParentCategoryID = cp.CategoryID
)
SELECT CategoryID, CategoryName, Path
```

FROM CategoryPath;



Question 3: Total Distinct Customers by Month

Task:

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**.**

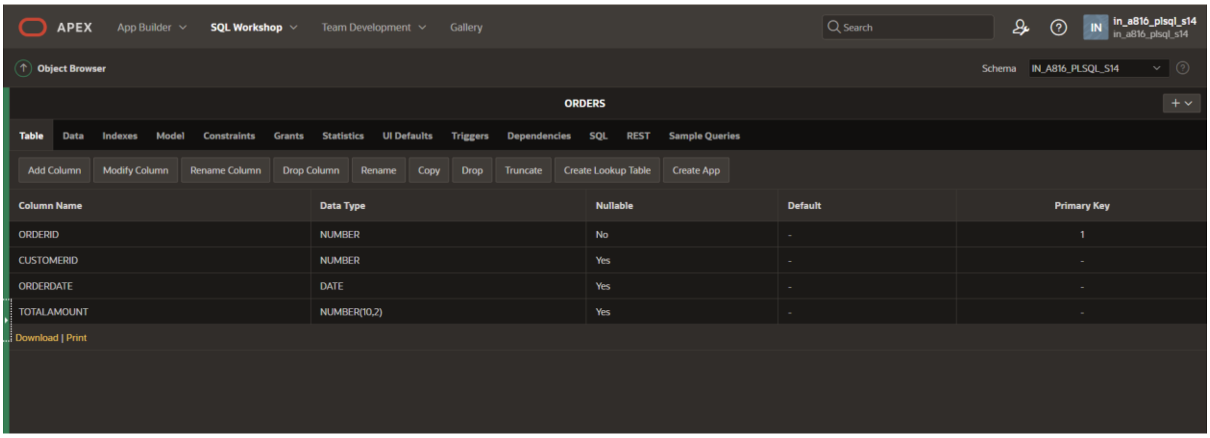
CREATE TABLE Orders

(OrderID INT PRIMARY KEY

, CustomerID INT,

OrderDate DATE,

TotalAmount NUMBER(10, 2));



ORDERS

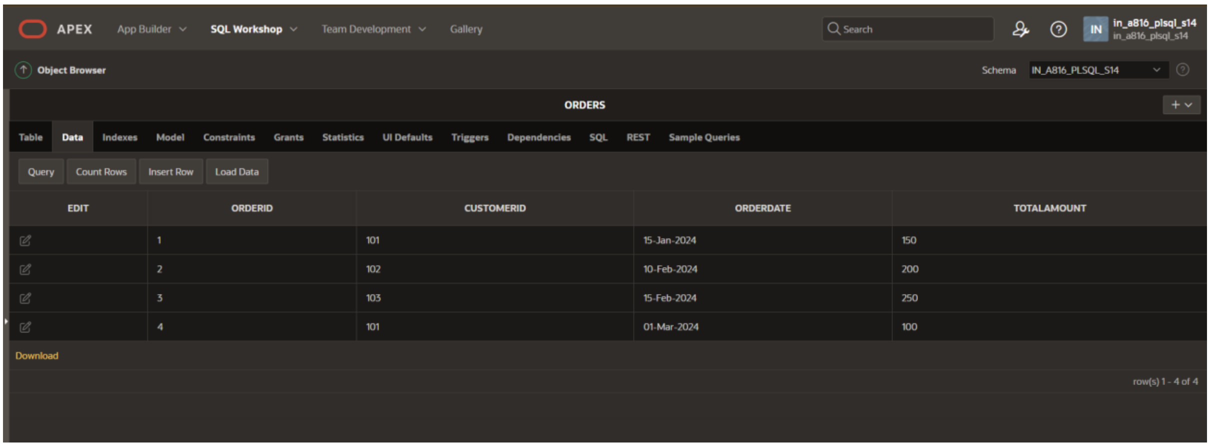
Column Name	Data Type	Nullable	Default	Primary Key
ORDERID	NUMBER	No	-	1
CUSTOMERID	NUMBER	Yes	-	-
ORDERDATE	DATE	Yes	-	-
TOTALAMOUNT	NUMBER(10,2)	Yes	-	-

INSERT INTO Orders VALUES (1, 101, TO_DATE('2024-01-15', 'YYYY-MM-DD'), 150.00);

INSERT INTO Orders VALUES (2, 102, TO_DATE('2024-02-10', 'YYYY-MM-DD'), 200.00);

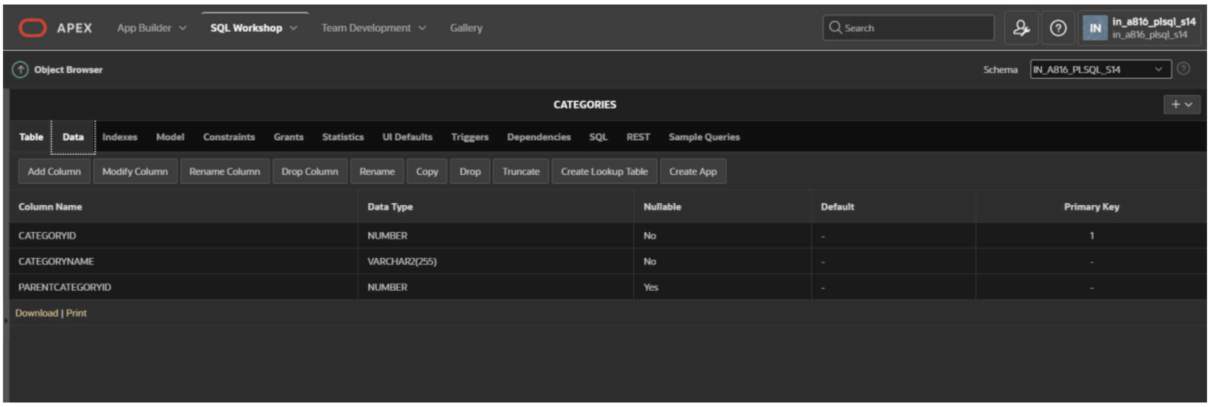
INSERT INTO Orders VALUES (3, 103, TO_DATE('2024-02-15', 'YYYY-MM-DD'), 250.00);

INSERT INTO Orders VALUES (4, 101, TO_DATE('2024-03-01', 'YYYY-MM-DD'), 100.00);



EDIT	ORDERID	CUSTOMERID	ORDERDATE	TOTALAMOUNT
	1	101	15-Jan-2024	150
	2	102	10-Feb-2024	200
	3	103	15-Feb-2024	250
	4	101	01-Mar-2024	100

```
WITH Months AS (

SELECT LEVEL AS MonthNumber

FROM DUAL

CONNECT BY LEVEL <= 12 ),

 CustomerCounts AS (

SELECT EXTRACT(MONTH FROM OrderDate) AS MonthNumber,

           COUNT(DISTINCT CustomerID) AS CustomerCount

FROM Orders

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

GROUP BY EXTRACT(MONTH FROM OrderDate) )

SELECT TO_CHAR(TO_DATE(m.MonthNumber, 'MM'), 'Month') AS MonthName,

            COALESCE(c.CustomerCount, 0) AS CustomerCount

 FROM Months m

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

ORDER BY m.MonthNumber;
```



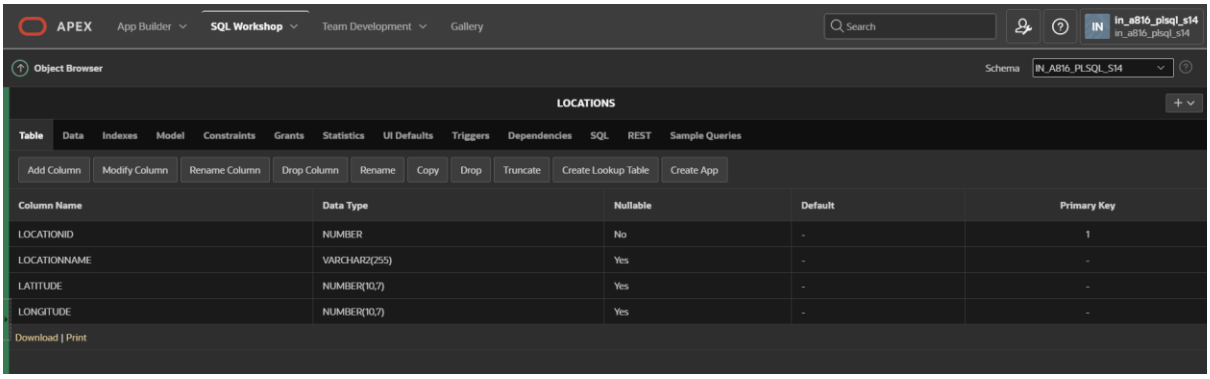
Column Name	Data Type	Nullable	Default	Primary Key
CATEGORYID	NUMBER	No	-	1
CATEGORYNAME	VARCHAR2(255)	No	-	-
PARENTCATEGORYID	NUMBER	Yes	-	-

Question 4: Finding Closest Locations

Task:

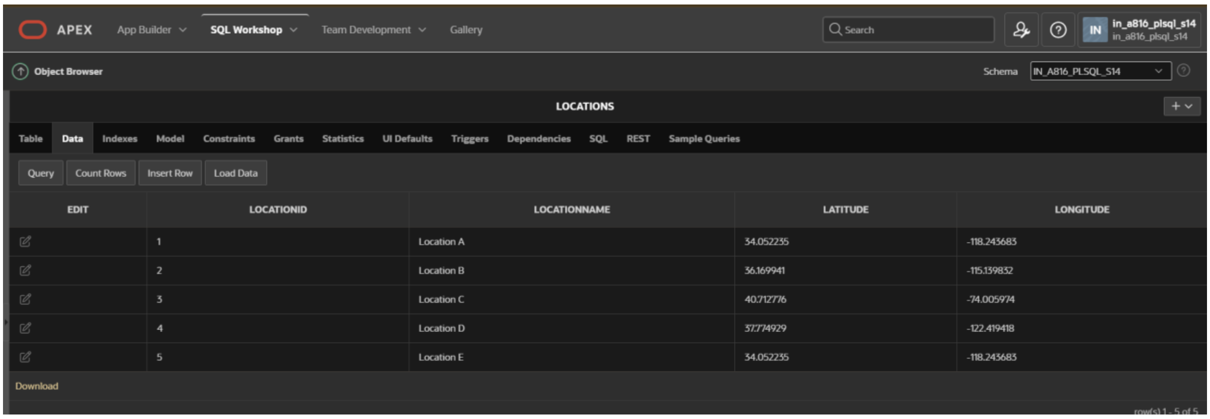
Write a SQL query to find the closest 5 locations to a given point specified by latitude and longitude. Use spatial functions or advanced mathematical calculations for proximity.

```
CREATE TABLE Locations (

LocationID INT PRIMARY KEY,

LocationName VARCHAR2(255),

Latitude NUMBER(10, 7),

Longitude NUMBER(10, 7)

);
```



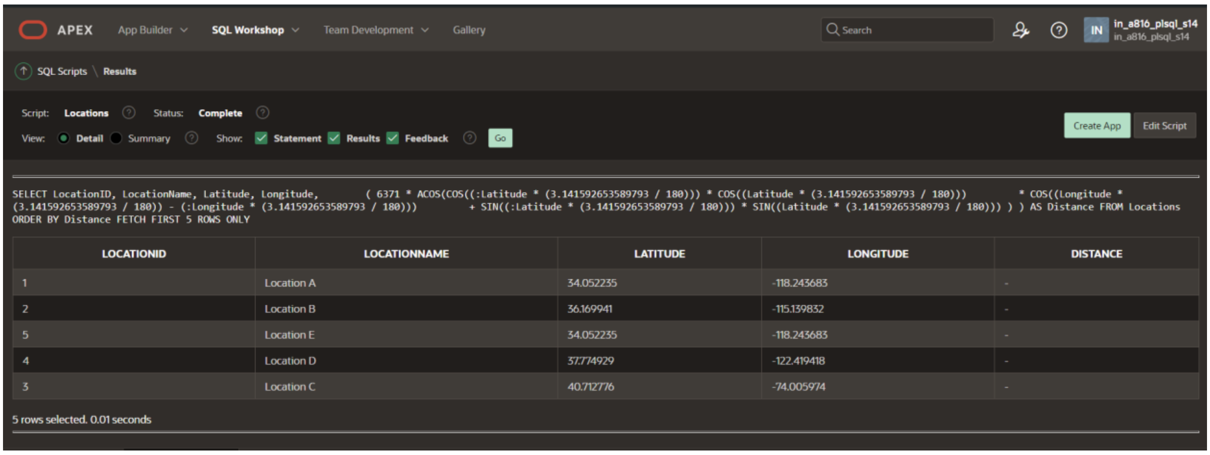
Column Name	Data Type	Nullable	Default	Primary Key
LOCATIONID	NUMBER	No	-	1
LOCATIONNAME	VARCHAR2(255)	Yes	-	-
LATITUDE	NUMBER(10,7)	Yes	-	-
LONGITUDE	NUMBER(10,7)	Yes	-	-

```
INSERT INTO Locations VALUES (1, 'Location A', 34.052235, -118.243683);

INSERT INTO Locations VALUES (2, 'Location B', 36.169941, -115.139832);

INSERT INTO Locations VALUES (3, 'Location C', 40.712776, -74.005974);

INSERT INTO Locations VALUES (4, 'Location D', 37.774929, -122.419418);

INSERT INTO Locations VALUES (5, 'Location E', 34.052235, -118.243683);
```



EDIT	LOCATIONID	LOCATIONNAME	LATITUDE	LONGITUDE
✎	1	Location A	34.052235	-118.243683
✎	2	Location B	36.169941	-115.139832
✎	3	Location C	40.712776	-74.005974
✎	4	Location D	37.774929	-122.419418
✎	5	Location E	34.052235	-118.243683

```
SELECT LocationID, LocationName, Latitude, Longitude,

    ( 6371 * ACOS(COS((:Latitude * (3.141592653589793 / 180))) * COS((Latitude *
(3.141592653589793 / 180)))

    * COS((Longitude * (3.141592653589793 / 180)) - (:Longitude * (3.141592653589793 / 180)))

    + SIN((:Latitude * (3.141592653589793 / 180))) * SIN((Latitude * (3.141592653589793 / 180))) ) )
AS Distance

FROM Locations

ORDER BY Distance

FETCH FIRST 5 ROWS ONLY;
```



Script: Locations Status: Complete

View: ● Detail ● Summary Show: ☑ Statement ☑ Results ☑ Feedback

```
SELECT LocationID, LocationName, Latitude, Longitude,       ( 6371 * ACOS(COS((:Latitude * (3.141592653589793 / 180))) * COS((Latitude * (3.141592653589793 / 180)))        * COS((Longitude *
(3.141592653589793 / 180)) - (:Longitude * (3.141592653589793 / 180)))        + SIN((:Latitude * (3.141592653589793 / 180))) * SIN((Latitude * (3.141592653589793 / 180))) ) ) AS Distance FROM Locations
ORDER BY Distance FETCH FIRST 5 ROWS ONLY
```

LOCATIONID	LOCATIONNAME	LATITUDE	LONGITUDE	DISTANCE
1	Location A	34.052235	-118.243683	-
2	Location B	36.169941	-115.139832	-
5	Location E	34.052235	-118.243683	-
4	Location D	37.774929	-122.419418	-
3	Location C	40.712776	-74.005974	-

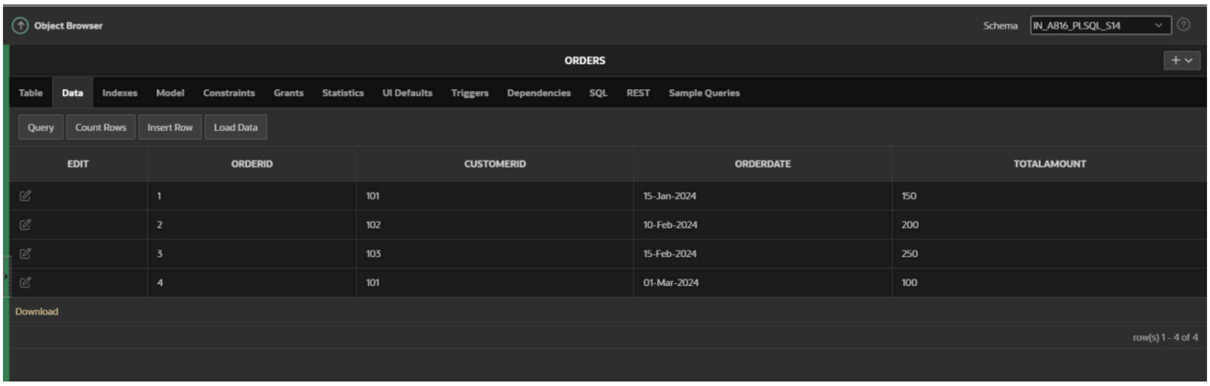
5 rows selected. 0.01 seconds

Question 5: Optimizing Query for Orders Table

Task:

1. 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.

```
SELECT

OrderID,

CustomerID,

OrderDate,

TotalAmount FROM Orders

WHERE OrderDate >= SYSDATE - INTERVAL '7' DAY

ORDER BY OrderDate DESC;
```



ORDERS

EDIT	ORDERID	CUSTOMERID	ORDERDATE	TOTALAMOUNT
	1	101	15-Jan-2024	150
	2	102	10-Feb-2024	200
	3	103	15-Feb-2024	250
	4	101	01-Mar-2024	100

row(s) 1 - 4 of 4