CSA09: DATABASE MANAGEMENT SYSTEMS-ASSIGNMENT QUESTIONS

BY: PRIYAL T 192324132

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).

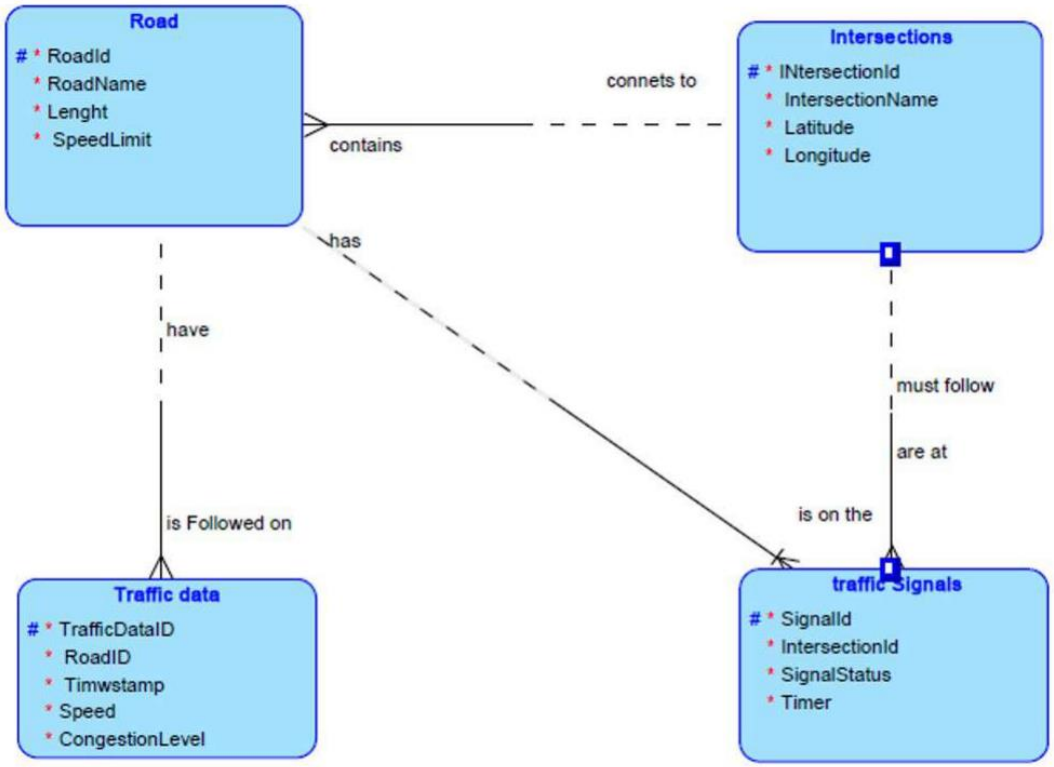
3.Roads to Traffic Data:

- 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.

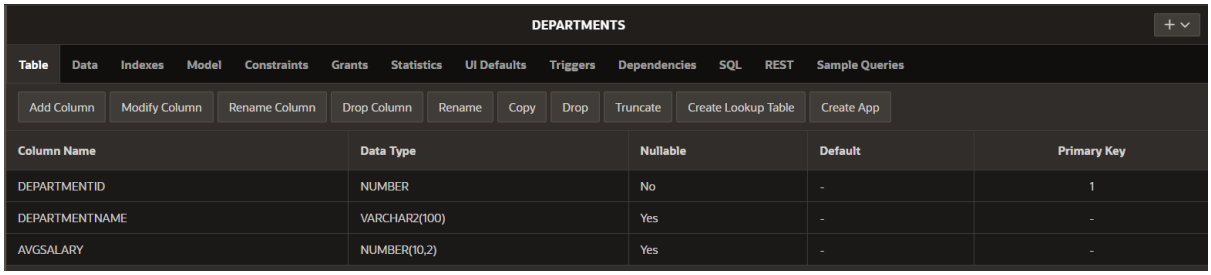
Question 2:

Question 1: Top 3 Departments with Highest Average Salary

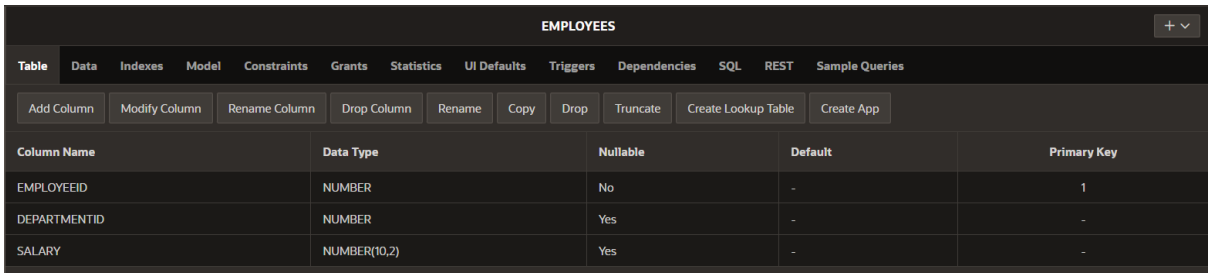
Task:

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 VARCHAR2(100), AvgSalary NUMBER(10, 2) -- Changed from DECIMAL to NUMBER);



CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, DepartmentID INT, Salary NUMBER(10, 2), -- Changed from DECIMAL to NUMBER CONSTRAINT fk_department FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID));



INSERT INTO Departments VALUES (1, 'HR', NULL),

INSERT INTO Departments VALUES (2, 'Finance', NULL),

INSERT INTO Departments VALUES (3, 'Engineering', NULL),

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

DEPARTMENTS

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Query | Count Rows | Insert Row | Load Data

EDIT	DEPARTMENTID	DEPARTMENTNAME	AVGSALARY
	1	HR	-
	3	Engineering	-
	2	Finance	-
	4	Marketing	-

INSERT INTO Employees VALUES (1, 1, 50000.00);

INSERT INTO Employees VALUES (2, 1, 60000.00);

INSERT INTO Employees VALUES (3, 2, 70000.00);

INSERT INTO Employees VALUES (4, 3, 80000.00);

INSERT INTO Employees VALUES (5, 3, 90000.00);

EMPLOYEES

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Query | Count Rows | Insert Row | Load Data

EDIT	EMPLOYEEID	DEPARTMENTID	SALARY
	4	3	80000
	1	1	50000
	5	3	90000
	2	1	60000
	3	2	70000

UPDATE Departments d SET d.AvgSalary = (SELECT AVG(e.Salary) FROM Employees e WHERE e.DepartmentID = d.DepartmentID);

DEPARTMENTS

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Query | Count Rows | Insert Row | Load Data

EDIT	DEPARTMENTID	DEPARTMENTNAME	AVGSALARY
	1	HR	55000
	3	Engineering	85000
	2	Finance	70000
	4	Marketing	-

SELECT DepartmentID, DepartmentName, AvgSalary FROM Departments ORDER BY AvgSalary DESC FETCH FIRST 3 ROWS ONLY;

DEPARTMENTID	DEPARTMENTNAME	AVGSALARY
4	Marketing	-
3	Engineering	85000
2	Finance	70000

3 rows returned in 0.01 seconds Download

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 VARCHAR2(100),
    ParentCategoryID INT,
    FOREIGN KEY (ParentCategoryID) REFERENCES Categories(CategoryID)
);
```

CATEGORIES

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Add Column | Modify Column | Rename Column | Drop Column | Rename | Copy | Drop | Truncate | Create Lookup Table | Create App

Column Name	Data Type	Nullable	Default	Primary Key
CATEGORYID	NUMBER	No	-	1
CATEGORYNAME	VARCHAR2(100)	Yes	-	-
PARENTCATEGORYID	NUMBER	Yes	-	-

```
-- Insert each category separately
INSERT INTO Categories (CategoryID, CategoryName, ParentCategoryID) VALUES (1, 'Electronics', NULL);  -- Root category
```

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

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

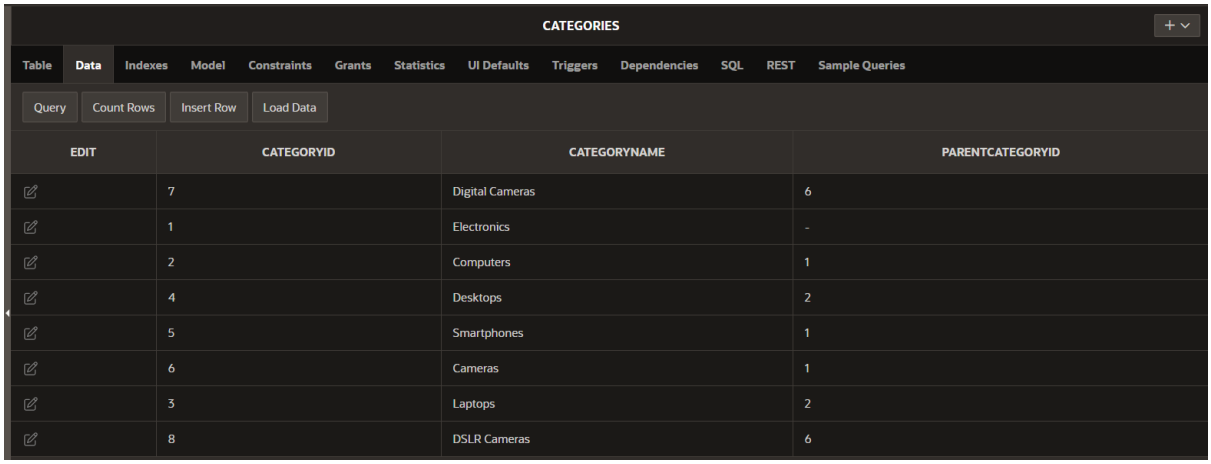
INSERT INTO Categories (CategoryID, CategoryName, ParentCategoryID) VALUES (4, 'Desktops', 2);

INSERT INTO Categories (CategoryID, CategoryName, ParentCategoryID) VALUES (5, 'Smartphones', 1);

INSERT INTO Categories (CategoryID, CategoryName, ParentCategoryID) VALUES (6, 'Cameras', 1);

INSERT INTO Categories (CategoryID, CategoryName, ParentCategoryID) VALUES (7, 'Digital Cameras', 6);

INSERT INTO Categories (CategoryID, CategoryName, ParentCategoryID) VALUES (8, 'DSLR Cameras', 6);



CATEGORIES

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Query | Count Rows | Insert Row | Load Data

EDIT	CATEGORYID	CATEGORYNAME	PARENTCATEGORYID
	7	Digital Cameras	6
	1	Electronics	-
	2	Computers	1
	4	Desktops	2
	5	Smartphones	1
	6	Cameras	1
	3	Laptops	2
	8	DSLR Cameras	6

WITH CategoryHierarchy (CategoryID, CategoryName, ParentCategoryID, HierarchicalPath) AS (

SELECT

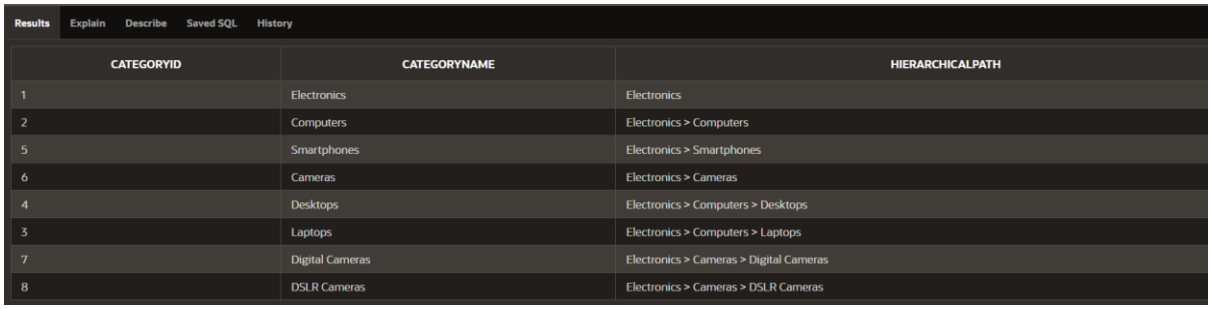
CategoryID,

CategoryName,

ParentCategoryID,

```
    CategoryName AS HierarchicalPath
  FROM
    Categories
  WHERE
    ParentCategoryID IS NULL

  UNION ALL

  SELECT
    c.CategoryID,
    c.CategoryName,
    c.ParentCategoryID,
    ch.HierarchicalPath || ' > ' || c.CategoryName AS HierarchicalPath
  FROM
    Categories c
  INNER JOIN
    CategoryHierarchy ch ON c.ParentCategoryID = ch.CategoryID
)
-- Select final result
SELECT
  CategoryID,
  CategoryName,
  HierarchicalPath
FROM
```

CategoryHierarchy;



Results | Explain | Describe | Saved SQL | History

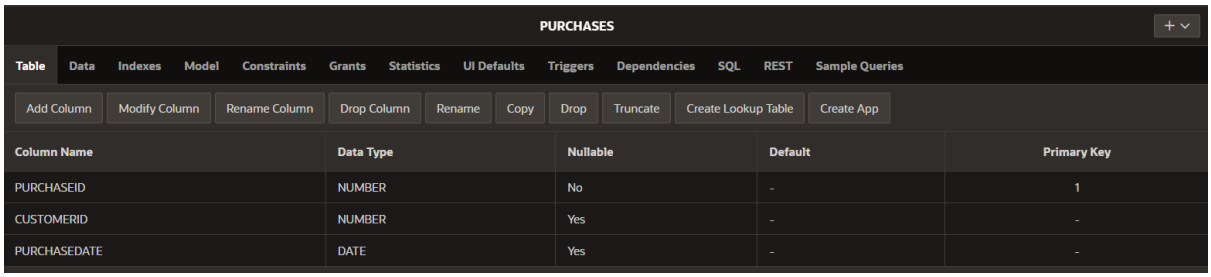
CATEGORYID	CATEGORYNAME	HIERARCHICALPATH
1	Electronics	Electronics
2	Computers	Electronics > Computers
5	Smartphones	Electronics > Smartphones
6	Cameras	Electronics > Cameras
4	Desktops	Electronics > Computers > Desktops
3	Laptops	Electronics > Computers > Laptops
7	Digital Cameras	Electronics > Cameras > Digital Cameras
8	DSLR Cameras	Electronics > Cameras > DSLR Cameras

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 Purchases (
    PurchaseID INT PRIMARY KEY,
    CustomerID INT,
    PurchaseDate DATE
);
```



PURCHASES

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Add Column | Modify Column | Rename Column | Drop Column | Rename | Copy | Drop | Truncate | Create Lookup Table | Create App

Column Name	Data Type	Nullable	Default	Primary Key
PURCHASEID	NUMBER	No	-	1
CUSTOMERID	NUMBER	Yes	-	-
PURCHASEDATE	DATE	Yes	-	-

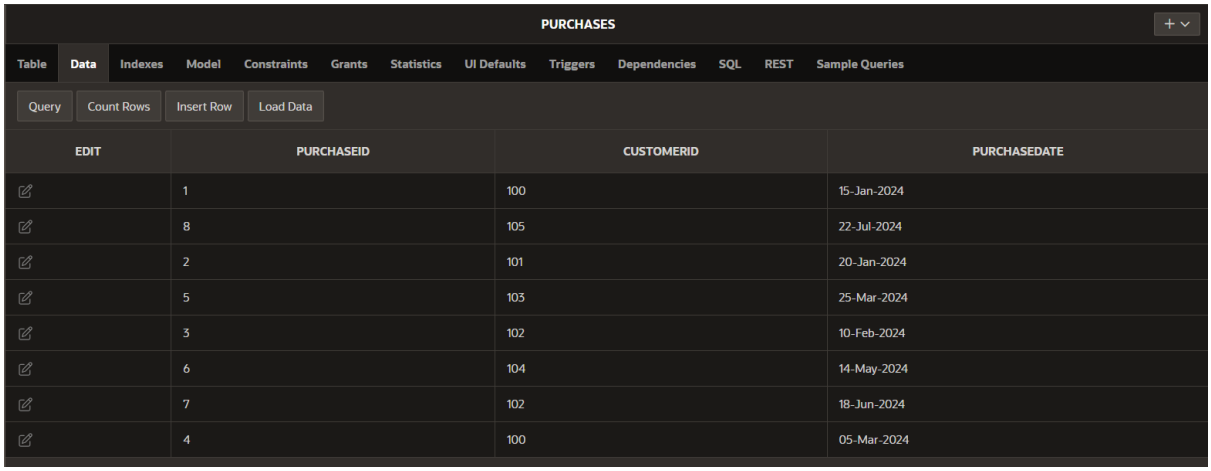
```
INSERT INTO Purchases VALUES (1, 100, DATE '2024-01-15');
INSERT INTO Purchases VALUES (2, 101, DATE '2024-01-20');
INSERT INTO Purchases VALUES (3, 102, DATE '2024-02-10');
INSERT INTO Purchases VALUES (4, 100, DATE '2024-03-05');
```

INSERT INTO Purchases VALUES (5, 103, DATE '2024-03-25');

INSERT INTO Purchases VALUES (6, 104, DATE '2024-05-14');

INSERT INTO Purchases VALUES (7, 102, DATE '2024-06-18');

INSERT INTO Purchases VALUES (8, 105, DATE '2024-07-22');



PURCHASES

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Query | Count Rows | Insert Row | Load Data

EDIT	PURCHASEID	CUSTOMERID	PURCHASEDATE
	1	100	15-Jan-2024
	8	105	22-Jul-2024
	2	101	20-Jan-2024
	5	103	25-Mar-2024
	3	102	10-Feb-2024
	6	104	14-May-2024
	7	102	18-Jun-2024
	4	100	05-Mar-2024

```
WITH Months AS (
   SELECT TO_CHAR(ADD_MONTHS(TRUNC(SYSDATE, 'YYYY'), LEVEL - 1), 'YYYY-MM') AS
MonthName
   FROM dual
   CONNECT BY LEVEL <= 12
),
CustomerActivity AS (
   SELECT
      TO_CHAR(PurchaseDate, 'YYYY-MM') AS MonthName,
      COUNT(DISTINCT CustomerID) AS CustomerCount
   FROM Purchases
   WHERE EXTRACT(YEAR FROM PurchaseDate) = EXTRACT(YEAR FROM SYSDATE)
   GROUP BY TO_CHAR(PurchaseDate, 'YYYY-MM')
```

```
)

SELECT

    m.MonthName,

    NVL(ca.CustomerCount, 0) AS CustomerCount

FROM

    Months m

LEFT JOIN

    CustomerActivity ca ON m.MonthName = ca.MonthName

ORDER BY

    m.MonthName;
```

MONTHNAME	CUSTOMERCOUNT
2024-01	2
2024-02	1
2024-03	2
2024-04	0
2024-05	1
2024-06	1
2024-07	1
2024-08	0
2024-09	0
2024-10	0

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.02 seconds Download

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(100),
```

```
    Latitude NUMBER(9, 6),
    Longitude NUMBER(9, 6)
);
```

LOCATIONS

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Add Column | Modify Column | Rename Column | Drop Column | Rename | Copy | Drop | Truncate | Create Lookup Table | Create App

Column Name	Data Type	Nullable	Default	Primary Key
LOCATIONID	NUMBER	No	-	1
LOCATIONNAME	VARCHAR2(100)	Yes	-	-
LATITUDE	NUMBER(9,6)	Yes	-	-
LONGITUDE	NUMBER(9,6)	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', 37.774929, -122.419418);
INSERT INTO Locations VALUES (4, 'Location D', 40.712776, -74.005974);
INSERT INTO Locations VALUES (5, 'Location E', 41.878113, -87.629799);
INSERT INTO Locations VALUES (6, 'Location F', 39.099727, -94.578568);
INSERT INTO Locations VALUES (7, 'Location G', 32.776665, -96.796989);
```

LOCATIONS

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Query | Count Rows | Insert Row | Load Data

EDIT	LOCATIONID	LOCATIONNAME	LATITUDE	LONGITUDE
	1	Location A	34.052235	-118.243683
	5	Location E	41.878113	-87.629799
	4	Location D	40.712776	-74.005974
	2	Location B	36.169941	-115.139832
	6	Location F	39.099727	-94.578568
	3	Location C	37.774929	-122.419418
	7	Location G	32.776665	-96.796989

```
WITH LocationDistances AS (
    SELECT
```

```
        LocationID,
        LocationName,
        Latitude,
        Longitude,
        3959 * ACOS(
            COS(RADIANS(:lat_point)) * COS(RADIANS(Latitude)) *
            COS(RADIANS(Longitude) - RADIANS(:lon_point)) +
            SIN(RADIANS(:lat_point)) * SIN(RADIANS(Latitude))
        ) AS Distance
    FROM
        Locations
)
SELECT
    LocationID,
    LocationName,
    Latitude,
    Longitude,
    Distance
FROM
    LocationDistances
ORDER BY
    Distance
FETCH FIRST 5 ROWS ONLY;
```

Bind Variable	Value
:LAT_POINT	34.052235
:LON_POINT	-118.243683

Question 5: Optimizing Query for Orders Table

Task:

Write a SQL query to retrieve orders placed in the last 7 days from a large Orders1 table, sorted by order date in descending order.

```
CREATE TABLE Orders1 (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    OrderAmount DECIMAL(10, 2)
);
```

ORDERS1

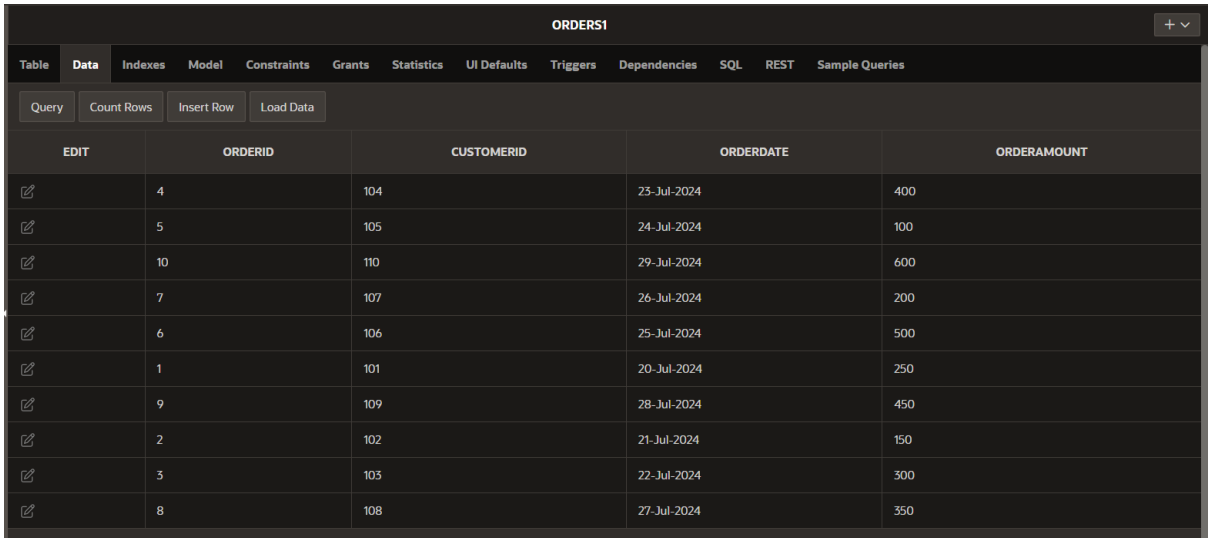
Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Add Column | Modify Column | Rename Column | Drop Column | Rename | Copy | Drop | Truncate | Create Lookup Table | Create App

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

```
INSERT INTO Orders1 VALUES (1, 101, DATE '2024-07-20', 250.00);
INSERT INTO Orders1 VALUES (2, 102, DATE '2024-07-21', 150.00);
INSERT INTO Orders1 VALUES (3, 103, DATE '2024-07-22', 300.00);
INSERT INTO Orders1 VALUES (4, 104, DATE '2024-07-23', 400.00);
```

```
INSERT INTO Orders1 VALUES (5, 105, DATE '2024-07-24', 100.00);

INSERT INTO Orders1 VALUES (6, 106, DATE '2024-07-25', 500.00);

INSERT INTO Orders1 VALUES (7, 107, DATE '2024-07-26', 200.00);

INSERT INTO Orders1 VALUES (8, 108, DATE '2024-07-27', 350.00);

INSERT INTO Orders1 VALUES (9, 109, DATE '2024-07-28', 450.00);

INSERT INTO Orders1 VALUES (10, 110, DATE '2024-07-29', 600.00);
```



ORDERS1

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL | REST | Sample Queries

Query | Count Rows | Insert Row | Load Data

EDIT	ORDERID	CUSTOMERID	ORDERDATE	ORDERAMOUNT
	4	104	23-Jul-2024	400
	5	105	24-Jul-2024	100
	10	110	29-Jul-2024	600
	7	107	26-Jul-2024	200
	6	106	25-Jul-2024	500
	1	101	20-Jul-2024	250
	9	109	28-Jul-2024	450
	2	102	21-Jul-2024	150
	3	103	22-Jul-2024	300
	8	108	27-Jul-2024	350

```
SELECT
    OrderID,
    CustomerID,
    OrderDate,
    OrderAmount
FROM
    Orders1
WHERE
    OrderDate >= SYSDATE - 7
ORDER BY
```

OrderDate DESC;

ORDERID	CUSTOMERID	ORDERDATE	ORDERAMOUNT
10	110	29-Jul-2024	600
9	109	28-Jul-2024	450
8	108	27-Jul-2024	350
7	107	26-Jul-2024	200
6	106	25-Jul-2024	500
5	105	24-Jul-2024	100