Question 1 – ER Diagram Question:

Traffic Flow Management System (TFMS) Scenario 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.

The city administration has decided to implement a TFMS to address growing traffic congestion issues. The system will integrate real-time data from traffic sensors, cameras, and historical traffic patterns to provide intelligent traffic management solutions. Key functionalities include:

1. **Road Network Management:**

o **Roads**: The city has a network of roads, each identified by a unique RoadID. Roads have attributes such as RoadName, Length (in meters), and SpeedLimit (in km/h).

2. **Intersection Control:**
   o **Intersections**: These are key points where roads meet and are crucial for traffic management. Each intersection is uniquely identified by IntersectionID and has attributes like IntersectionName and geographic Coordinates (Latitude, Longitude).
3. **Traffic Signal Management:**
   o **Traffic Signals**: Installed at intersections to regulate traffic flow. Each signal is identified by SignalID and has attributes such as SignalStatus (Green, Yellow, Red) indicating current state and Timer (countdown to next change).
4. **Real-Time Data Integration:**
   o **Traffic Data**: Real-time data collected from sensors includes TrafficDataID, Timestamp, Speed (average speed on the road), and CongestionLevel (degree of traffic congestion).
5. **Functionality Requirements:**
   o Route Optimization: Algorithms will be implemented to suggest optimal routes based on current traffic conditions.
   o **Traffic Signal Control**: Adaptive control algorithms will adjust signal timings dynamically based on real-time traffic flow and congestion data.
   o **Historical Analysis**: The system will store historical traffic data for analysis and planning future improvements.

ER Diagram Design Requirements

1. **Entities and Attributes**:
   o Clearly define entities (Roads, Intersections, Traffic Signals, Traffic Data) and their attributes based on the scenario provided.
   o Include primary keys (PK) and foreign keys (FK) where necessary to establish relationships between entities.
2. **Relationships**:
   o Illustrate relationships between entities (e.g., Roads connecting to Intersections, Intersections hosting Traffic Signals).
   o Specify cardinality (one-to-one, one-to-many, many-to-many) and optionality constraints (mandatory vs. optional relationships).
3. **Normalization Considerations**:
   o Discuss how you would ensure the ER diagram adheres to normalization principles (1NF, 2NF, 3NF) to minimize redundancy and improve data integrity. **Tasks**

Task 1: Entity Identification and Attributes Identify and list the entities relevant to the TFMS based on the scenario provided (e.g., Roads, Intersections, Traffic Signals, Traffic Data).

Define attributes for each entity, ensuring clarity and completeness.

Task 2: Relationship Modeling

Illustrate the relationships between entities in the ER diagram (e.g., Roads connecting to Intersections, Intersections hosting Traffic Signals).

Specify cardinality (one-to-one, one-to-many, many-to-many) and optionality constraints (mandatory vs. optional relationships).

Task 3: ER Diagram Design

Draw the ER diagram for the TFMS, incorporating all identified entities, attributes, and relationships.

Label primary keys (PK) and foreign keys (FK) where applicable to establish relationships between entities.

Task 4: Justification and Normalization

Justify your design choices, including considerations for scalability, real-time data processing, and efficient traffic management.

Discuss how you would ensure the ER diagram adheres to normalization principles (1NF, 2NF, 3NF) to minimize redundancy and improve data integrity.

Deliverables

1. **ER Diagram**: A well-drawn ER diagram that accurately reflects the structure and relationships of the TFMS database.

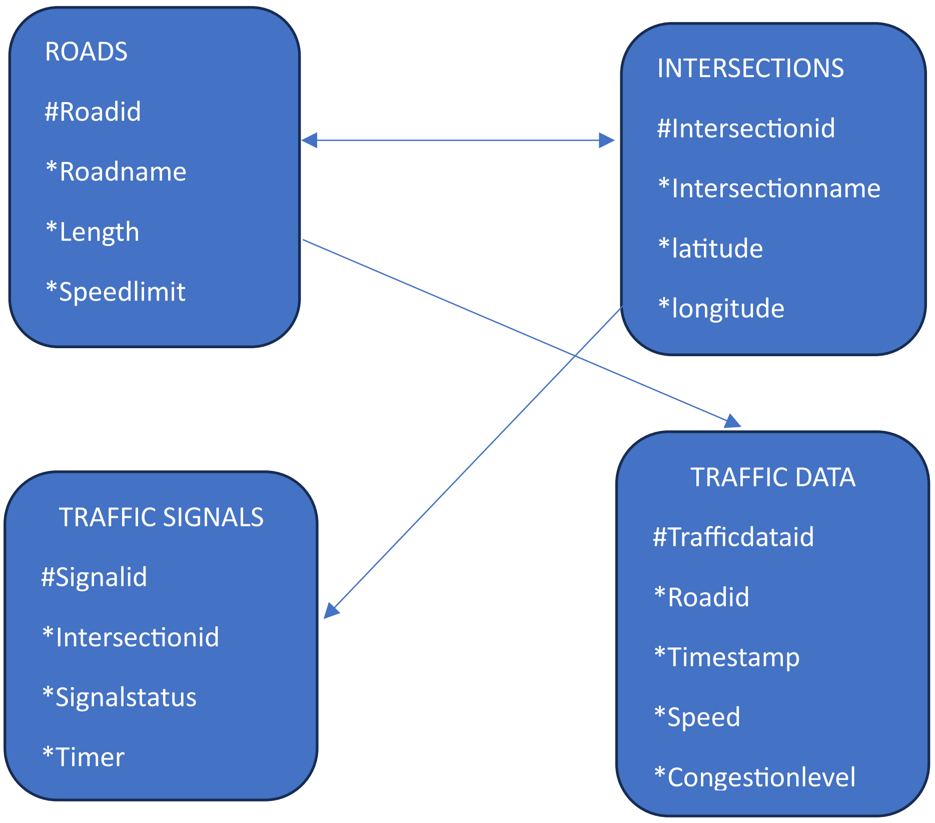
2. Entity Definitions: Clear definitions of entities and their attributes, supporting the ER diagram.

3. Relationship Descriptions: Detailed descriptions of relationships with cardinality and optionality constraints.

4. Justification Document: A document explaining design choices, normalization considerations, and how the ER diagram supports TFMS functionalities.

Roads (RoadID, RoadName, Length, SpeedLimit)

ANSWER: ER DIAGRAM DESIGN



QUESTION_2:SQL

Question 1: Top 3 Departments with Highest Average Salary

Task:

1. 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.

Deliverables:

1. SQL query that retrieves DepartmentID, DepartmentName, and AvgSalary for the top 3 departments.

2. Explanation of how the query handles departments with no employees and calculates average salary.

Question 2: Retrieving Hierarchical Category Paths

Task:

1. 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).

Deliverables:

1. SQL query that uses recursive CTE to fetch CategoryID, CategoryName, and hierarchical path.

2. Explanation of how the recursive CTE works to traverse the hierarchical data.

Question 3: Total Distinct Customers by Month

Task:

1.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.

Deliverables:

1. SQL query that retrieves MonthName and CustomerCount for each month.

2. Explanation of how the query ensures all months are included and handles zero customer counts.

Question 4: Finding Closest Locations

Task:

1. 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.

Deliverables:

1. SQL query that calculates the distance and retrieves LocationID, LocationName, Latitude, and Longitude for the closest 5 locations.

2. Explanation of the spatial or mathematical approach used to determine proximity.

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.

Deliverables:

1. SQL query optimized for performance, considering indexing, query rewriting, or other techniques.

2. Discussion of strategies used to optimize the query and improve performance.

<u>**ANSWER**</u>:

Question 1 Ans: Top 3 Departments with Highest Average Salary

```sql
SELECT DepartmentID, DepartmentName, AVG(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

LIMIT 3;
```

Question 2 Ans: Retrieving Hierarchical Category Paths

```sql
WITH RECURSIVE CategoryPath AS (

  SELECT

    CategoryID,

    CategoryName,

    CAST(CategoryName AS VARCHAR(255)) AS Path

  FROM Categories

  WHERE ParentCategoryID IS NULL


  UNION ALL


  SELECT

    c.CategoryID,

    c.CategoryName,

    CONCAT(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 Ans: Total Distinct Customers by Month

```
WITH Months AS (
    SELECT 1 AS MonthNumber UNION ALL
    SELECT 2 UNION ALL
    SELECT 3 UNION ALL
    SELECT 4 UNION ALL
    SELECT 5 UNION ALL
    SELECT 6 UNION ALL
    SELECT 7 UNION ALL
    SELECT 8 UNION ALL
    SELECT 9 UNION ALL
    SELECT 10 UNION ALL
    SELECT 11 UNION ALL
    SELECT 12
)
SELECT
    MONTHNAME(STR_TO_DATE(m.MonthNumber, '%m')) AS MonthName,
    COUNT(DISTINCT c.CustomerID) AS CustomerCount
FROM
    Months m
LEFT JOIN
    Orders o ON MONTH(o.OrderDate) = m.MonthNumber AND YEAR(o.OrderDate) =
YEAR(CURDATE())
LEFT JOIN
    Customers c ON o.CustomerID = c.CustomerID
GROUP BY
    m.MonthNumber
```

```
ORDER BY
    m.MonthNumber;
```

Question 4 Ans: Finding Closest Locations

```
SELECT
    LocationID,
    LocationName,
    Latitude,
    Longitude,
    (
        6371 * ACOS(
            COS(RADIANS(@latitude)) * COS(RADIANS(Latitude)) *
            COS(RADIANS(Longitude) - RADIANS(@longitude)) +
            SIN(RADIANS(@latitude)) * SIN(RADIANS(Latitude))
        )
    ) AS Distance
FROM
    Locations
ORDER BY
    Distance
LIMIT 5;
```

Question 5 Ans: Optimizing Query for Orders Table

```
SELECT OrderID, OrderDate, CustomerID, TotalAmount
FROM Orders
WHERE OrderDate >= CURDATE() - INTERVAL 7 DAY
ORDER BY OrderDate DESC;
```

Question 3: PL/SQL Questions

Question 1: Handling Division Operation

Task:

1. Write a PL/SQL block to perform a division operation where the divisor is obtained from user input. Handle the ZERO_DIVIDE exception gracefully with an appropriate error message.

Deliverables:

1. PL/SQL block that performs the division operation and handles exceptions.
2. Explanation of error handling strategies implemented.

Question 2: Updating Rows with FORALL

Task:

1. Use the FORALL statement to update multiple rows in the Employees table based on arrays of employee IDs and salary increments.

Deliverables:

1. PL/SQL block that uses FORALL to update salaries efficiently.
2. Description of how FORALL improves performance for bulk updates.

Question 3: Implementing Nested Table Procedure

Task:

1. Implement a PL/SQL procedure that accepts a department ID as input, retrieves employees belonging to the department, stores them in a nested table type, and returns this collection as an output parameter.

Deliverables:

1. PL/SQL procedure with nested table implementation.
2. Explanation of how nested tables are utilized and returned as output.

Question 4: Using Cursor Variables and Dynamic SQL

Task:

1. Write a PL/SQL block demonstrating the use of cursor variables (REF CURSOR) and dynamic SQL. Declare a cursor variable for querying EmployeeID, FirstName, and LastName based on a specified salary threshold.

Deliverables:

1. PL/SQL block that declares and uses cursor variables with dynamic SQL.
2. Explanation of how dynamic SQL is constructed and executed.

Question 5: Designing Pipelined Function for Sales Data

Task:

1. Design a pipelined PL/SQL function get_sales_data that retrieves sales data for a given month and year. The function should return a table of records containing OrderID, CustomerID, and OrderAmount for orders placed in the specified month and year.

Deliverables:

1. PL/SQL code for the pipelined function get_sales_data.
2. Explanation of how pipelined table functions improve data retrieval efficiency.

ANSWER

Question 1 Ans: Handling Division Operation DECLARE

```
DECLARE

   numerator NUMBER := 100;

   divisor NUMBER;

   result NUMBER;

BEGIN

   -- Obtain divisor from user input

   divisor := &divisor;



   -- Perform division operation

   result := numerator / divisor;



   -- Display result

   DBMS_OUTPUT.PUT_LINE('Result: ' || result);



EXCEPTION

   WHEN ZERO_DIVIDE THEN

      DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.');

END;

/
```

Question 2 Ans: Updating Rows with FORALL

```
DECLARE
    TYPE EmployeeIDArray IS TABLE OF Employees.EmployeeID%TYPE;
    TYPE SalaryIncrementArray IS TABLE OF NUMBER;

    employee_ids EmployeeIDArray := EmployeeIDArray(1, 2, 3);
    salary_increments SalaryIncrementArray := SalaryIncrementArray(500, 1000, 1500);
BEGIN
    FORALL i IN employee_ids.FIRST .. employee_ids.LAST
        UPDATE Employees
        SET Salary = Salary + salary_increments(i)
        WHERE EmployeeID = employee_ids(i);
END;
/
```

Question 3 Ans: Implementing Nested Table Procedure

```
CREATE OR REPLACE TYPE EmployeeTableType AS TABLE OF Employees%ROWTYPE;

CREATE OR REPLACE PROCEDURE GetEmployeesByDepartment(
    p_DepartmentID IN Employees.DepartmentID%TYPE,
    p_EmployeeTable OUT EmployeeTableType
) IS
BEGIN
    SELECT * BULK COLLECT INTO p_EmployeeTable
    FROM Employees
    WHERE DepartmentID = p_DepartmentID;
END;
```

```
/
```

Question 4 Ans: Using Cursor Variables and Dynamic SQL

```
DECLARE
    TYPE CursorType IS REF CURSOR;
    c CursorType;
    v_EmployeeID Employees.EmployeeID%TYPE;
    v_FirstName Employees.FirstName%TYPE;
    v_LastName Employees.LastName%TYPE;
    v_SalaryThreshold NUMBER := &salary_threshold;
BEGIN
    OPEN c FOR 'SELECT EmployeeID, FirstName, LastName FROM Employees WHERE Salary >
:1' USING v_SalaryThreshold;
    LOOP
        FETCH c INTO v_EmployeeID, v_FirstName, v_LastName;
        EXIT WHEN c%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('EmployeeID: ' || v_EmployeeID || ', FirstName: ' || v_FirstName ||
', LastName: ' || v_LastName);
    END LOOP;
    CLOSE c;
END;
/
```

Question 5 Ans: Designing Pipelined Function for Sales Data

```
CREATE OR REPLACE TYPE SalesRecordType AS OBJECT (
    OrderID NUMBER,
    CustomerID NUMBER,
    OrderAmount NUMBER
);


CREATE OR REPLACE TYPE SalesRecordTableType AS TABLE OF SalesRecordType;


CREATE OR REPLACE FUNCTION GetSalesData(p_Month IN NUMBER, p_Year IN NUMBER)
  RETURN SalesRecordTableType PIPELINED
IS
  v_OrderID Orders.OrderID%TYPE;
  v_CustomerID Orders.CustomerID%TYPE;
  v_OrderAmount Orders.OrderAmount%TYPE;
BEGIN
  FOR r IN (SELECT OrderID, CustomerID, OrderAmount
        FROM Orders
        WHERE EXTRACT(MONTH FROM OrderDate) = p_Month
         AND EXTRACT(YEAR FROM OrderDate) = p_Year)
  LOOP
    PIPE ROW(SalesRecordType(r.OrderID, r.CustomerID, r.OrderAmount));
  END LOOP;
  RETURN;
END;
/
```