Assignment-III

Name:- C. Eliyaraj
Register No:- 192311162
Course Code:- CSA0563
Course Name:- Data Base Management System

1 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.

1. Roads
   - Attributes: RoadID (PK), Road Name, Length, Speed Limit
2. Intersections
   - Attributes: IntersectionID (PK), Intersection Name, Latitude, Longitude.
3. Traffic Signals
   - Attributes: Signal ID (PK), Signal Status (Green, Yellow, Red), Timer (countdown to next change), Intersection ID (FK)
4. Traffic Data
   - Attributes: Traffic Data ID (PK), Time stamp, Speed, congestion Level, Road ID (FK)

Roads	Intersections	Traffic Signals	Traffic Data
RoadID (PK)	IntersectionID (PK)	SignalID (PK)	Traffic DataID (PK)
Road Name	Intersection name	IntersectionID (FK)	RoadID (FK)
length	latitude	signal status	Time stamp
Speed Limit	longitude	Timer	speed
			congestion level

Task 2: Relationship Modeling

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

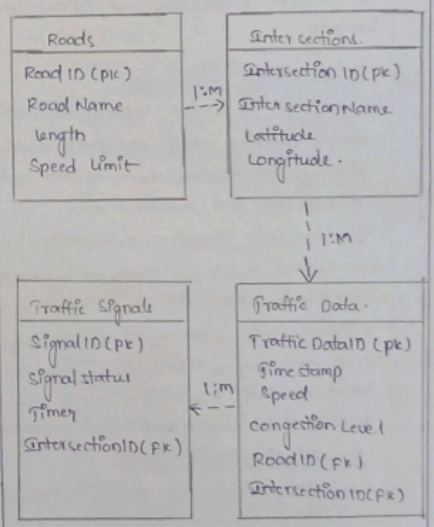
Specify cardinality (one-to-one, one-to-many, many-to-many) and optionality constraints (mandatory vs. optional relationships).

- Roads (1) -- (connects to) -- (1 or more) Intersections
  - cardinality: one road connects to one or more intersections.
  - optionality: Mandatory (every road must connect to at least one intersection).
- Intersections (1) -- (hosts) -- (1 or more) Traffic Signals
  - cardinality: one intersection hosts one or more traffic signals.
  - optionality: optional (an intersection may not have any traffic signals).
- Intersections (1) -- (generates) -- (1 or more) Traffic Data.
  - cardinality: one intersection generates one or more traffic data entities.
  - optionality: optional (an intersection may not have immediate traffic data if sensors fail)
- Roads (1) -- (has) -- (0 or more) Traffic Data
  - cardinality: one road can have zero or more traffic data entities.
  - optionality: optional (not all roads might have real-time traffic data collected).

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



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.

```
SELECT
    FORMAT (PurchaseDate, 'MMM') AS MonthName,
    COUNT (DISTINCT CustomerID) AS customerCount
FROM
    Purchases
WHERE
    YEAR (PurchaseDate) = YEAR (CURRENT_DATE)
GROUP BY
    FORMAT (PurchaseDate, 'MMM')
ORDER BY
    MIN (PurchaseDate);
```

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.

```
SELECT
    LocationID,
    LocationName,
    Latitude,
    Longitude,
    SQRT (POW( (Latitude - @given_latitude ,2) +
        POW(Longitude - @given_longitude ,2) ) AS Distance
FROM
    Locations
ORDER BY
    Distance
LIMIT 5;
```

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,
    OrderDate,
    CustomerID,
    Total-Amount
FROM
    Orders
WHERE
    Order Date >= DATE_SUB( CURRENT_DATE, INTERVAL
                                7 DAY)
ORDER BY
    OrderDate DESC;
```

③

Question 2:

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.

```
DECLARE
  v_divided NUMBER := 100;
  v_divisor NUMBER;
  v_result NUMBER;
BEGIN
  v_divisor := &divisor_input;
  v_result := v_dividend / v_divisor;
  DBMS_OUTPUT.PUT_LINE ('Result of
      division :' || v_result);
EXCEPTION
  WHEN ZERO_DIVIDE THEN
    DBMS_OUTPUT.PUT_LINE ('Error:
    Division by zero is not allowed!');
WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE ('An unexcepted
    error occured ' || SQLERRM);
END;
/
```

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 increment.

```
DECLARE
  TYPE emp_id_array IS TABLE OF
      employees.employee_id%TYPE;
  TYPE sal_increment_array IS TABLE
      OF NUMBER;
v_emp_ids emp_id_array := emp_id_
array (1001, 1002, 1003);
```

```
v_salaries sal_increment_array := sal_increment
        _array(500,600,700);
BEGIN
  FORALL i IN INDICES OF v_emp_ids
    UPDATE employees
    SET salary = salary + v_salaries(i)
    WHERE employee_id = v_emp_ids(i);
  COMMIT;
END;
```

Question3: 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 return the collections as an output parameter.

```
CREATE OR REPLACE PROCEDURE
      get_employees_by_dept (
  p_dept_id IN employees.department_id%TYPE,
  p_employees OUT SYS_REFCURSOR
  ) AS
  BEGIN
    OPEN p_employees FOR
    SELECT employee_id, first_name, last_name
    FROM employees
    WHERE department_id = p_dept_id;
  END;
  /
```

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, First_name and last name based on a specified salary threshold.

```
DECLARE
  TYPE emp_cursor_type IS REF CURSOR;
  v_emp_cursor emp_cursor_type;
  v_salary_threshold NUMBER := 50000;
  v_employee_id employees.employee_id%TYPE;
  v_first_name employees.first_name%TYPE;
  v_last_name employees.last_name%TYPE;
BEGIN
  OPEN v_emp_cursor FOR
  'SELECT employee_id, first_name, last_name
   FROM employees
   WHERE v_salary_threshold';
LOOP
  FETCH v_emp_cursor INTO v_employee_id, v_first_name,
        v_last_name;
  EXIT WHEN v_emp_cursor%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('Employee ID:' || v_employee
                                      _id ||
                       ', Name:' || v_firstname || ' '
                       || v_last_name);
  END LOOP;
  CLOSE v_emp_cursor;
END;
/
```

Question 5: Designing Pipelined Functions for Sales Data.

Task:

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

```
CREATE OR REPLACE TYPE sales_record AS
  OBJECT (
    order_id NUMBER
    customer_id NUMBER
    order_amount NUMBER
  );
CREATE OR REPLACE TYPE sales_table IS TABLE
  OF sales_record;
CREATE OR REPLACE FUNCTION get_sales_data
 (p_month IN NUMBER, p_year IN NUMBER)
RETURN sales_table PIPELINED IS
BEGIN
  FOR rec IN (
    SELECT order_id, customer_id, order_amount
    FROM sales
    WHERE EXTRACT(MONTH FROM order_date)
                    = p_month
    AND EXTRACT(YEAR FROM order_date)
                    = p_year
  ) LOOP
  PIPE ROW (sales_record(rec.order_id, rec.
        customer_id, rec.order_amount));
  END LOOP;
END;
/
```

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.

Sol

- Design choices Justification:
- Scalability: The design supports scalability by clearly defining entities and their relationships, allowing for efficient querying and updating of real-time and historical data.
- Real-time Data Processing:- Entities like Traffic Data and Traffic Signals are structured to handle real-time updates and dynamic changes in traffic conditions.
- Efficient Traffic Management:- Relationships such as Roads connecting to Intersections and Traffic Signals being located at Intersections enable effective traffic flow control and signal management.
- Normalization considerations:
- 1NF: All attributes are atomic (indivisible) and each table has a distinct primary key.
- 2NF: No partial dependencies exist; all non-key attributes are fully functionally dependent on the primary key.
- 3NF: Elimination of transitive dependencies ensures that each attribute directly relates to the primary key, promoting data integrity and minimizing redundancy.

Question 2:

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.

```
SELECT SELECT
    d.DepartmentID,
    d.DepartmentName,
    AVG(e.Salary) AS Avg_Salary
FROM
    Department d
LEFT JOIN
    Employees e ON d.DepartmentID = e.DepartmentID
GROUP BY
    d.DepartmentID, d.DepartmentName
ORDER BY
    Avg_Salary DESC
LIMIT 3;
```

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 sub category).

```
WITH RECURSIVE CategoryPaths AS (
    SELECT
        CategoryID,
        CategoryName,
        CAST(CategoryName AS VARCHAR(255)) AS CategoryPath
FROM
    Categories
WHERE
    ParentCategoryID IS NULL
UNION ALL
SELECT
    c.CategoryID,
    c.CategoryName,
    CONCAT(cp.CategoryPath, ' > ', c.CategoryName)
FROM
    Categories c
JOIN
    CategoryPaths cp ON c.ParentCategoryID
        = cp.CategoryID
)
SELECT
    CategoryID,
    CategoryName,
    CategoryPath
FROM
    CategoryPaths.
```