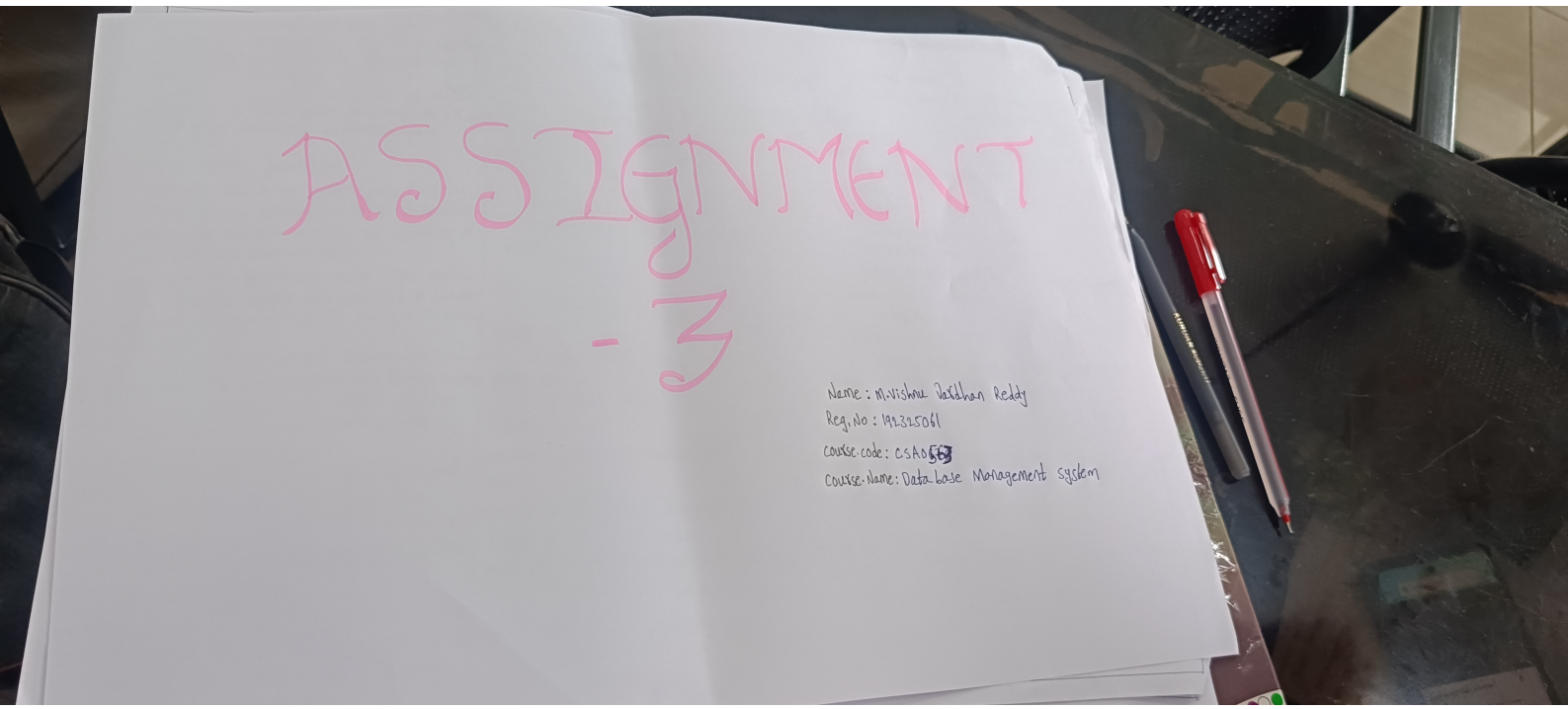

ASSIGNMENT
-3
Name : M.Vishnu Vardhan Reddy
Reg.No : 192325061
course code : CSA0
Course Name : Data base Management system

Project - 2 :

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.

Sol 1. Roads

- Attributes : Road (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)	Intersection ID (PK)	Signal ID (PK)	Traffic Data ID (PK)
Road Name	Intersection name	Intersection ID (PK)	Road ID (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 hosting Traffic signals).

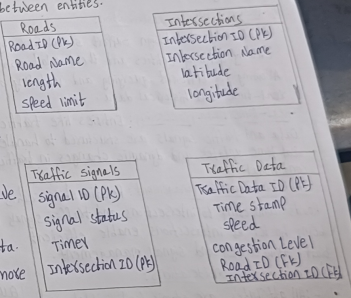
specify cardinality (one-to-one, one-to-many, many-to-many) and optionality constraints, (mandatory vs optional relationships)

- 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 intersections)
- Intersections (1) -- (hosts) -- (1 or more) Traffic signals.
  - Cardinality : one intersection hosts one or more traffic signals.
  - optionality : optional can 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 :

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.



Project Exercise 1

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 adhere to normalization principles (1NF, 2NF, 3NF) to minimize redundancy and improve data integrity.

Sol,

- Design choices Justification:
- Scalability: The design support 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 conditional.
- Efficient Traffic Management: Relationships such as Roads connecting to intersections and traffic signals being hostel at intersections enable efficient 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 Exit; all non-key attributes are fully functionally dependent on the Primary key.
- 3NF: Elimination of transitive dependencies ensure that each attribute directly relates to the Primary key, Promoting data integrity and minimizing redundancy.

2 Question

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
    d.Department ID,
    d.Department Name,
    Avg(e.salary) AS Avg salary
FROM
    Department d
LEFT JOIN
    Employees e ON d.DepartmentID=e.Department ID
GROUP BY
    d.DepartmentID, d.Department Name.
ORDER BY
    Avg salary DESC
LIMIT 3;
```

Question 2: Retrieving Hierarchical category Paths.

Task!
1. Write a SQL query using recursive common Job Expressions (CTE) to retreive all categories along with their full heirarchical Path (e.g. category > subcategory > sub sub category).

```
WITH RECURSIVE Category Paths AS (
    SELECT (
        category ID,
        category Name,
        CAST(category Name AS VARCHAR(255)) AS
            category Path
    FROM
        categories
    WHERE
        Parent category ID IS NULL
    UNION ALL
    SELECT
        c.category ID,
        c.category Name,
        CONCAT(cp.category Path, '>', C.category Name)
    FROM
        categories c
    JOIN
        category Paths cp ON c.Parent category ID
            = cp.category ID.
)
SELECT
    category ID,
    category Name,
    category Path,
FROM
    category Paths.
```

Question 3: Total District customers by month.

Task:

1. Design a sql query to find the total number of district customers who made a Purchase in each month of the current year. Ensure month with no customer activity show a count of 0.

```
SELECT
  FORMAT(Purchase Date, 'MMM') AS month name,
  COUNT (DISTINCT customer ID) AS customer count
FROM
  Purchases
WHERE
  YEAR(Purchase Date) = YEAR(CURRENT_DATE)
GROUP BY
  FORMAT(Purchase Date, 'MMM')
ORDER BY
  MIN(Purchase Date);
```

Question 4: Finding closest locations

Task:

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

```
SELECT
    Location ID,
    Location Name,
    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 Tables.

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
    order ID,
    order Date,
    customer ID,
    Total Amont,
    order location,
FROM
    orders
WHERE
  order Date >= DATE_SUB(CURRENT_DATE, INTERUAL
                                    7 DAY)
ORDER BY
    order Date DESC;
```

3 Question

Question 1: Handling Division operation.

SQL QUERY

```
DECLARE
V_numerator Number:=100;
V_divisor NUMBER;
V_result NUMBER;
BEGIN
  V_divisor := &user_divisor;
  V_result := V_number / V_divisor;
DBMS_OUTPUT.PUT_LINE('Result of division' || V_result);
Exception
WHEN zero_divide Then
 DBMS_output.Put_LINE('Error: Divison by zero not allowed');
 WHEN OTHERS THEN
 DBMS_output.Put_LINE('An unexpected error || SQLERRM);
 END;
```

Question 2: updating Rows with for all

SQL query

```
DECLARE
  Type emp_id_array IS TABLE off Number;
  Type salary_array IS TABLE off Number;
V_emp_ids emp_id_array := emp_id_array(101, 102, 103);
V_salaries salary_array := salary_array(500, 600, 700);
BEGIN
 FORALL i IN 1..V_emp_ids.count
   update employees
    SET salary = salaray + V_salaries(i)
```

```
where Employee ID = v_emp_ids(i);
COMIT;
DBMS_output.PUT_LINE ('salaries is updated successfully
Exception
WHEN OTHERS THEN
   DBMS_output.PUT_LINE ('AN error occured: ||
                          SQLERM);
 ROLL BACK;
END;
```

Question 3:- Implementing Nested Table Procedure.

SQL QUERY

```
CREATE OR REPLACE PROCEDURE GET_Employees
                    -BY_Dept(
   P. dept_id IN Number,
   P_emp_list OUT SYS_REFCURSOR
)AS
BEGIN
  OPEN P_emp_list 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

SQL QUERY

```
DECLARE
 Type emp_cursor IS REF CURSOR;
V_emp_cursor emp_cursor;
 V_salaray_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 salary > V_salary_threshold;
LOOP
FETCH V_emp_cursor INTO V_employee_id,
                V_first Name, V_last_name;
EXIT WHEN V_emp_cursor %. NOT FOUND;
DBMS_OUTPUT_LINE('ID': || V_employee_id || 'Name
 || V_first name || ' ' || V_last name)
END LOOP;
CLOSE V_EMP_cursor;
Exception
   WHEN OTHERS THEN
DBMS_OUTPut.PUT_LINE('AN error occured:'||SQLERM);
 END;
```

Question 5: Designing Pipelined function salesData

SQL QUERY

```
CREATE OR REPLACE TYPE Sales_Record object(
      order ID Number;
      Customer ID number;
      order Amount Number
   );
CREATE OR Replace type Sales_Table IS Table of
   Sales_Record
CREATOR Replace function get_sales_data(P_month
 IN Number, P_YEAR IN NUMBER)
RETURN Sales_table PIPELINED
 AS
BEGIN
WHERE EXTRACT(MONTH FROM Order data) = P.month
AND EXTRACT (YEAR FROM order dates = P.Yea
)
LOOP
 PIPE ROW(Sales_Record (order ID, V_customer ID));
END LOOP;
END;
```