ASSIGNMENT-3

G. Raddy prakash
192365026.

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, Trafic data) Define attributes for each entity ensuring clarity and completeness

eg:
1. Roads.
   - Attributes: Road(PK), Road Name, Length, speed Limit
2. Intersections:
   - Attributes: Intersection ID(PK), Intersection Name, Latitude, Longitude.
3. Traffic Signals:
   - Attributes: Signal ID(PK), Signal status (Green, Yellow, Red), Timer (countdown to next change) Intersection ID(PK)
4. Traffic data.
   Attributes: Traffic Data ID(PK), Time Stamp, speed, congestion Level, Road ID(PK)

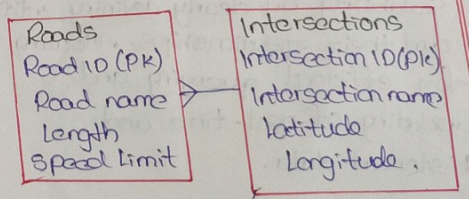
Roads	Intersections	Traffic Signals	Traffic data.
Road ID, Road name, Length	Intersection ID (PK), Intersection, Latitude	Signal ID (PK), Intersection ID, Signal status	Traffic data ID (PK), Road ID(FK), Time Stamp

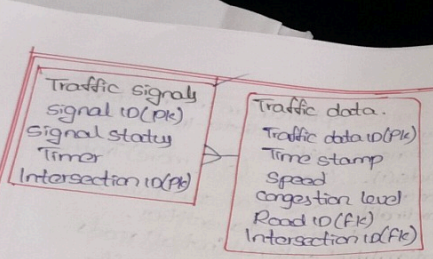
Task 2: Relationship modeling
Illustrate the relationships between entities in the ER diagram (e.g. Roads connecting to intersections hosting Traffic Signals).
Specify cardinality (one-to-one, one-to-many, many-to-many) and optionality constraints. (mandatory Vs optinal relation ships).

- 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)....(hods)--(1 or more) Traffic signals
Cardinality: One intersection hots 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
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, in corporate all identified entities, attributes and relationships.
Label primary keys (PK) and foreign keys (FK) where applicable to establish the relationship 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 adhere to normalization principles (1NF, 2NF, 3NF) to minimize redundancy and improve data integrity.

Ans: 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 hosted 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 exist; 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.

Question
Q1: Top 3 department with highest Avgpge salary
Task 1:
i) Write a SQL query to find the top 3 departments with the hgest average salary of employees, ensure department with no employee shows an average salary of NULL.

```
SELECT
d.Department ID,
d.Department Name,
Avg(e.salary) As Avg Salary.
FROM
    Departmentid
LEFT JOIN
    EMPLOYees e ON d.department ID=e.
                                Department ID
GROUP By
    d.Department ID, d.Department Name.
ORDER By
    AVG Salary DESC
    LIMIT 3;
```

Question 4: Using Cursor variables and Dynamic SQL
Task:
1) Write a PL/SQL block demonstrating the user of coursor variables and dynamic SQL Declare a cursor variables for quering employee ID, firstname, and last name.

```
DECLARE
 TYPE emp-cursor-type
 IS REF CURSOR;
v-EMP-cursor.emp-cusor-type;
v-Salary-thrashold NOBER := 5000;
v-employee-ID employee.employee-idx Type;
v-lastname employees.first_name-Type;
v-last name employees.Last name Type
BEGIN
 OPEN v.emp_cursor FOR
SELECT employees-id, first_name, last_
name
FROM employees.
 WHERE salary >:1
 USING v-salary-thersold;
 LOOP
DBMS-output:1Nt(1fmpolce1D:||v-
   employee-id))
      ;namel|v-first_namell,
       v-Lastname.
END LOOP
```

Q.5 Designing pipelined function for
Sales data.

```
CREATE OR REPLACE TYPES sales-record AS
                                OBJECT
 order-id NUMBER:
 customer-id NUMBER.
 ):
CREATE OR REPLACE Type sales-table
 IS TABLE of sales-record
 CREATE OR REPLACE FUNCTION get-sales
 data (P.month IN NUMBER. IN NUMBER
 RETURN sales-table. PIPELINED IS
  BEGIN
  FOR re
   SELECT order
  FROM sale
WHERE Extract (month from order)
AND extract (year from order.date)
```

Q3

Handling Division operation.

```
DECLARE
  v-divided Number =100
  v-divisor Number;
  v-result Number
v-divisor; divisor-input;
 v-result: v-divided /v.divisor
 WHERE other then
DBMS output-Line ("An Extend error
    occured ||SQL|FROM);
  END,
```

Q. Updating Rows with ... statement
Use FORALL and update the statement

```
DECLARE
TYPE emp.id-array IS TABLE Of employees.
Type sal-increment-array IS TABLE of number;
 BEGIN
FORALL i IN INDICES Of v-emp ids
 UPDATE Employees
  SET salary = salary+v.v-salaries(i)
  WHERE employee-id = v.emp-ids(i);
   COMMIT;
   END;
```

3) Implementing Nested Table procedure
Task
Implement a PL/SQL procedure that accepts a depart ID as input.

```
 CREATE OR REPLACE PROCEDURE get-
employees-by-dept(
P.dept-id IN employees. Department-id
                       type;
 P-Employees OUT Says-REFCURSOR
 )AS
 OPEN P.Employee FOR
 SELECT employee-id, first-name,
                  last-name

   from employees
  WHERE department-id = P-dept-id;
   END;.
```

paths.

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 category path 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

```
SELECT
FROM (Purchase Date, 'MM') As month name,
COUNT(DISINCT 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.

```
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 order table.

```
  SELET
    Order ID,
    Order Name,
    Order Date,
    Customer Amount
    Customer ID
     Total Amount
     Order location,
     FROM
       Orders
WHERE
Order Date >= DATE_SUB(CURRENT_DATE_
                        INTERVAL 7 DAY)

ORDER By
     Order Date DESE.
```