Assignment-1

Name: T. Sanjith kumar Reddy

Reg No: 192116710

Sub Code: CSA0563

Sub name: Database management system

No of pages used: 5

1

ER Diagram Design For Traffic Flow Management System (TFMS)

Entities and Attributes (Task 1)

a) Road
- Road ID
- Road Name
- Length
- Speed Limit

b) Intersection
- Intersection ID
- Intersection Name
- Latitude
- Longitude

c) Traffic Signal
- Signal ID
- Signal Status
- Timer

d) Traffic Data
- Traffic Data ID
- Time stamp
- Speed
- Congestion Level

Relationships (Task 2)

1) Road - Intersection (many-to-many)
- Road ID (FK)
- Intersection ID (FK)

2) Intersection - Traffic (many-to-many)
- Intersection (FK)
- Traffic (FK)

3) Road - Traffic Data (one-to-many)
- Road (FK)
- Traffic (FK)

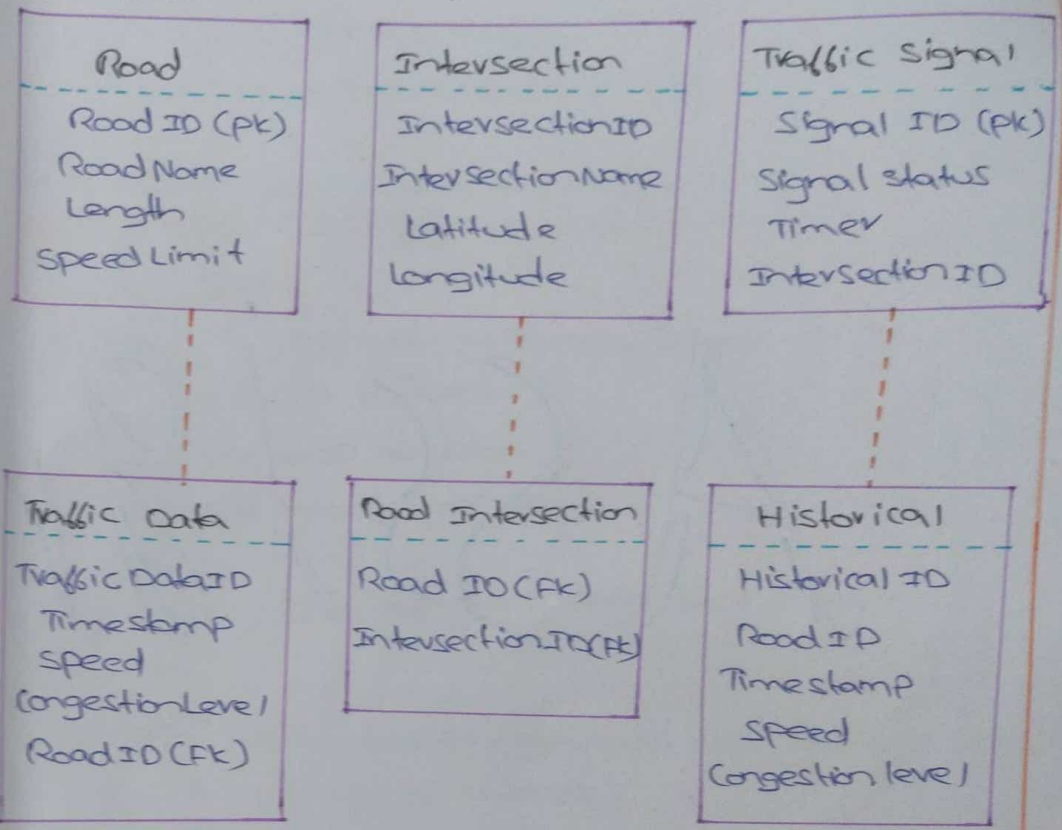
Normalisation (Task 4.1) (conserdelations)

1) First Normal Form (1NF)
Ensure that all attributes contain atomic values

2) Second Normal Form (2NF)
Ensure that non-key attributes are fully functionally

3) Third Normal Form (3NF)
Ensure that all attributes are on depent only primary key

ER Diagram (Task 3)



Justification Task (4.2)

1) Scalability : The design supports the addition of new roads, intersections, traffic signals and data records without structral changes.

2) Real-Time data processing

The traffic data entity is designed to accomidate high frequency updates with timestamped records The one-to-many relationships between Road and Traffic

Question-1 Top 3 Departments with Highest Average salary

SQL QUERY

```
SELECT
    DepartmentID,
    DepartmentName,
    AVG(salary) AS AvgSalary

FROM
    Employees
LEFT JOIN
  Departments ON Employees.DepartmentID=
                 Departments.Department ID


Group By
    DepartmentID, DepartmentName
ORDER BY
    Avg salary DESC
LIMIT 3;
```

Question 2: Retriveing Hierarchical category paths

SQL QUERY

```
WITH RECURSIVE CategoryHiravchy AS (
SELECT
      CategoryID,
      CategoryName,
      CAST(CategoryName AS VARCHAR(MAX)) AS path
FROM
      Categories
WHERE
    parentCategoryID IS NULL
UNION ALL
 SLECT
     C.CategoryID,
     C.CategoryName,
   CAST(Ch.path +'>'+ C.CategoryName AS VARCHAR
                    (MAX)) AS path
 FROM
     Categories C
INNER JOIN
 CategoryHiravchy ch ON C.parentCategoryID
                      = ch.CategoryID

)
SELECT
     CategoryID,
     CategoryName,
     path
FROM
     CategoryHiveachy .
ORDER BY
      path;
```

Question 3: Total Distinct Customers By Month

SQL Query:

```
WITH Months AS (
SLECT DATE_Format(Date_ADD(CURDATE(),Interval-1*
```

```
n MONTH), '%Y-%m') AS month_year
FROM
(SELECT @row := @row+1 AS n FROM (SELECT 1 UNION ALL
    SELECT 2 UNION ALL SELECT 3)  AS months
)
SELECT
    m.month_year AS MonthName,
COUNT(DISTINCT o.customer_ID) AS customerCount
FROM
    months m
LEFT JOIN
    orders o ON DATE_format(o.orderdate, '%Y-%m') = m.year
GROUP BY
    m.month_year
ORDER BY
    m.month_year;
```

Question 4: Finding closest Locations

SQL QUERY

```
SELECT
    LocationID,
    LocationName,
    Latitude
    Longitude,
    (3371 * 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: optimizing Query for orders Table

SQL QUERY

```
SELECT
    *
FROM
    orders
WHERE
    orderDate >= CURDATE() - INTERVAL 7 DAY
ORDER BY
    orderDate DESC;
```

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_numerator / V_divisor;
  DBMS_OUTPUT.PUT_LINE ('Result of division' || V_result);
EXCEPTION
 WHEN ZERO_DIVIDE THEN
  DBMS_OUTPUT.PUT_LINE ('Error : Division by zero not allowed');
 WHEN OTHERS THEN
  DBMS_output.put_LINE ('An unexpected error' || SQLERRM);
END;
```

Question 2:- Updating Rows with FORALL

SQL QUERY

```
DECLARE
   TYPE emp_id_array IS TABLE OF NUMBER;
   TYPE Salary_array IS TABLE OF NUMBER;
 V_emp_ids emp_id_array := emp_id_array (101, 102, 103);
V_Salaries Salary_array := Salary_array (5000, 6000, 7000);
BEGIN
  FORALL i IN V_emp_ids.COUNT
    UPDATE Employees
     SET Salary = salary + V_salaries (i)
    WHERE Employee ID = V_emp_ids(i);
 COMIT;
 DBMS_OUTPUT.PUT_LINE('Salaries is updated SUCCESFULLY');
EXCEPTION
 WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE ('An error occured !' || SQLERRM);
    ROLLBACK;
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_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 salary > 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_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 :' || SQLERRM);
END;
```

Question 5:- Designing pipelined function for Sales Data

SQL QUERY

```
CREATE OR REPLACE TYPE sales_Record  object (
        OrderID NUMBER,
        CustomerID NUMBER,
        OrderAmount NUMBER
    );
CREATE OR REPLACE TYPE Sales_Table IS TABLE OF Sales_
    Record
CREATOR REPLACE function get_sales_date (p_month IN
  NUMBER , p_year IN NUMBER)
RETURN Sales_table PIPELINED
  AS
  BEGIN
   WHERE EXTRACT (month from orderdate) = p_month
   AND    EXTRACT (year from orderdate) = p_year
   )
   LOOP
      PIPE ROW (sales_Record (orderID, v.customerID));
  END LOOP;
END;
```