DataBase Management and System

Assignment No	:-	03
Course code	:-	CSA0563
Register ID	:-	192372037
student NAME	:-	V. Sreekanth
No. of pages	:-	07.
Date of submission	:-	27-07-2024

Question 1:-

ER Diagram Question:- Traffic flow mangement System:-

Scenario :-

you are tasked with desiging an entity-relationship diagram for a traffic flow management system (TFMS)

Task 1:- Entity Identifacation and attributes.

Roads	Intersections	Traffic signals	Traffic Data.
Road ID (pk) Road name Length (m) speed Limit (km)	Intersection id (pk) Insertion name Latitude Longitude	signal ID (pk) Intersection ID (FK) signal status timer.	Traffic Data ID (pk) Road ID (pk) Time stamp Speed Congestion level.

Task 2:- Relationship modeling.

Relationships.

1. Roads to Intersections.

-> One road up can connect to multiple intersections.

-> An intersection can be connected by multiple roads.

2. Intersection to traffic Signals:

-> One intersection can host multiple traffic data entities

Cardinality and optiomatily

1. Roads of intersection

-> one road can connect to zero or more intersection

-> one intersection can conn'd to one or more roads

2. Intersections to traffic signals.

-> one intersection can here zero or more traffic. signals

-> one Traffic signal must be associated with one intersection.

3. Roads to Traffic Data :

-> one roads can have zero or more traffic data entities.

-> one traffic data entry must be associated with one road.

Task 4:- Justification & normalization.

1. Scalability

-> The design allows for easy addition of new roads, intersections, traffic signals, and traffic data entities without modifying the core.

2. Real-time Data Processing.

-> A Real-time traffic data integration is facilated by the traffic data.

3. Efficient Traffic mangement :-

-> A the clear Seperation of entities.

Deliverable :-

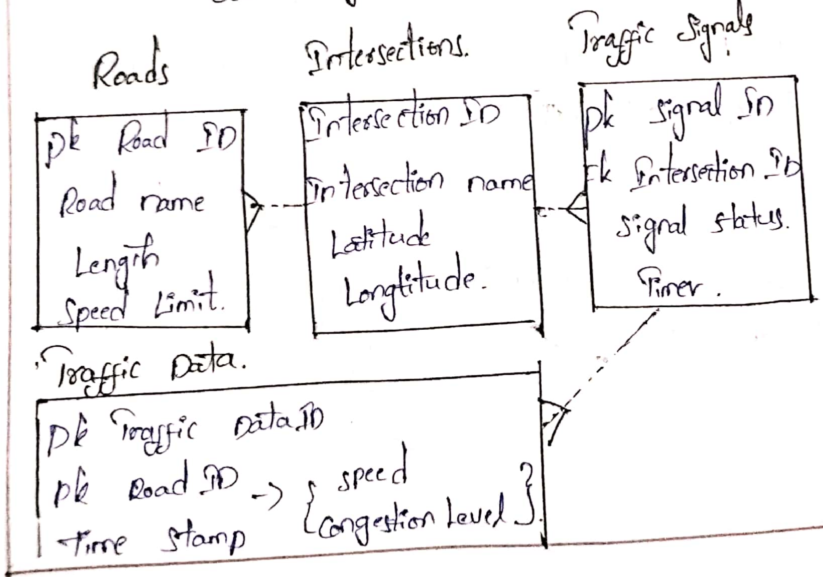
Er diagram :- provided above in plain Text format.

entity defnition: Listed in Task 1.

Relationship Descriptions.

Justification Document.

Task 3 :- ER diagram Design :-



Question 2 :-

Question 1 :- Top 3 Departments with highest average salary.

```
Sql Query
  with Avg Salaries as {
  SELECT
      d. department ID,
      d. Department Name,
     AVG (e. salary) as avg salary.
  FROM
     Departments.d
   LEFT Join Employees          d. department ID = Department ID.
     Group By.
       d. department ID,
       d. Department Name
    )
    SELECT
       Department ID,
       Department name
    )
    SELECT
        Department ID.
        Department name,
 Avg Salary.
From
 avg salaries
ORDER BY
  avg Salary Desc Nulls LAST
 LIMIT 3;
```

Question 2:- Retrieving Hierachial category Paths

SQL Query:

```
 WITH Recursive category Path AS(
 SELECT
   c. category ID,
   c. category name,
   c. parent category ID
Cast (C.categoryname AS VARCHAR (255)) AS path
 FROM
   categories.c
WHERE
 C. Current category ID ISNL
UNION ALL
SELECT
  c. category ID
 C. category Name,
  C. Parent category ID,
CAST (CP. Path ||', '||c. category Name AS VARCHAR (255) AS path
FROM
Catagories C
  INNER JOIN category Path cp ON c Parent Category ID = cp. category ID
 )
 SELECT
   category ID,
   category Name,
    path.
 FROM
   category Paths;
```

Final Query:

→ Select 'category ID', 'category Name' and the hierachial 'path' from the 'category paths' CTE.

→ This Query effictively Trouverses the heirachard category Structure and builds the Jued for each category.

Question 3:- Total Distinct Customers By month

SQL Query:-

```
WITH MONTHS AS (
SELECT DATE_FORMAT (DATE_ADD (CURDATE(), Interval-*
n MONTH), '%Y-%m') AS month-year
FROM
(SELECT @ row: = @ row+1 AS n FROM (SELECT 1 unioul ALL
  SELECT 2 unioul 2 SELECT 4) AS MONTHS
)
SELECT
    M. month year AS MONTH NAME,
COUNT (DISTINCT O. customer_ID) AS customer count
FROM
    MONTHS m.
LEF JOIN
    orders O ON DATE_FORMAT (Corder date; %Y.%m')=m.year
Group By
    m. month year
ORDER By
    m. month year;
```

Question 4:- Finding closest Locations.

SQL Query.

```
SELECT
    Location ID,
    Location NAME,
    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: optimizing query for orders table.

SQL Query.

```
SELECT
  *
 FROM
   orders.
```

```
WHERE
 Order Date >= (CURDATE()- INTERNAL 7 Day.
Order By
     order Date DesC;
```

③ Question 7 :- Handling Division operation:-

SQL Query.

```
DECLARE
v_ numerator Number := 100;
v_ divisor NUMBER;
v_ result NUMBER;
BEGIN
  v_divisoR: = & user_divisor;
  v_ result: = v_numertor /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 allow'));
WHEN OTHERS THEN
   DBMS_output. put_LINE ('An unexpected error || SQLERM);
END;
```

Question 8:- updating Rows with FORALL.

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 Employess
  SET salary = salary + 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_REFUCRSOR.
) As.
BEGIN
  OPEN p_emp_list FOR
  SELECT Employee ID, First NAME, LastNAME.
  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. LastNAME % 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_firstname
    || ' ' || V_Lastname)
    END Loop;
     CLOSE V_emp_cursor;
  Exception
      WHEN OTHERS THEN
   DBMS_output. put_LINE ('An error occurred:' || SQLERM);
  END;
```

Questions:- Designing pipelined function for sales Data.

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
CREATE OR REPLACE FUNCTION get_Sales_date (P_month IN Number,
                                          P_year IN NUMBER).
 RETURN Sales_table PIPELINED
 AS
   BEGIN
    WHERE EXTRACT (MONTH' FROM order date) = P_month
    AND EXTRACT (YEAR FROM orderdate) = P_year
    )
   Loop
    PIPE Row (Sales_Record (Order_ID, V_customer ID));
 END Loop;
 END;
```