Assignment - 3.

By.

M.M.Naveen

192311166

DSA0563-

DBMS.

1) ER Diagram Questions: Traffic Flow Managment System

ER Diagram Design Requirements:

(i) Entities and Attributes

(ii) Relationship

(iii) Normalization consideration

Task 1: Entity Identification and Attributes

1. Roads:

- Attributes.

(i) Road_id (Primary Keys)

(ii) RoadName

(iii) Length

(iv) speedlimit

2. Intersection:

- Attributes:

(i) Intersection_id (PK)

(ii) Intersectionname

(iii) Latitude

(iv) Longitude.

3. Traffic Signal

- Attributes

(i) signals (PK)

(ii) Signalstatus (Green, yellow, Red)

(iii) Times

- Foreign Key:

(i) Intersection (FK) - to indicate intersection the signal belongs to

4- Traffic Data

• Attributes :

(i) Traffic DataID (PK)

(ii) TimeStamp

(iii) speed

(iv) Congestion Level

• Foreign Key :

1. Road ID (FK) - to indicate which road the traffic data pertains to

Task 2 : Relationship Modeling

• Roads and intersation Relationship :

(i) One road can connect to multiple intersection (one-to-many)

(ii) One intersection cannot to multiple roads (one-to-many)

• Intersection and Traffic Signals Relationship :

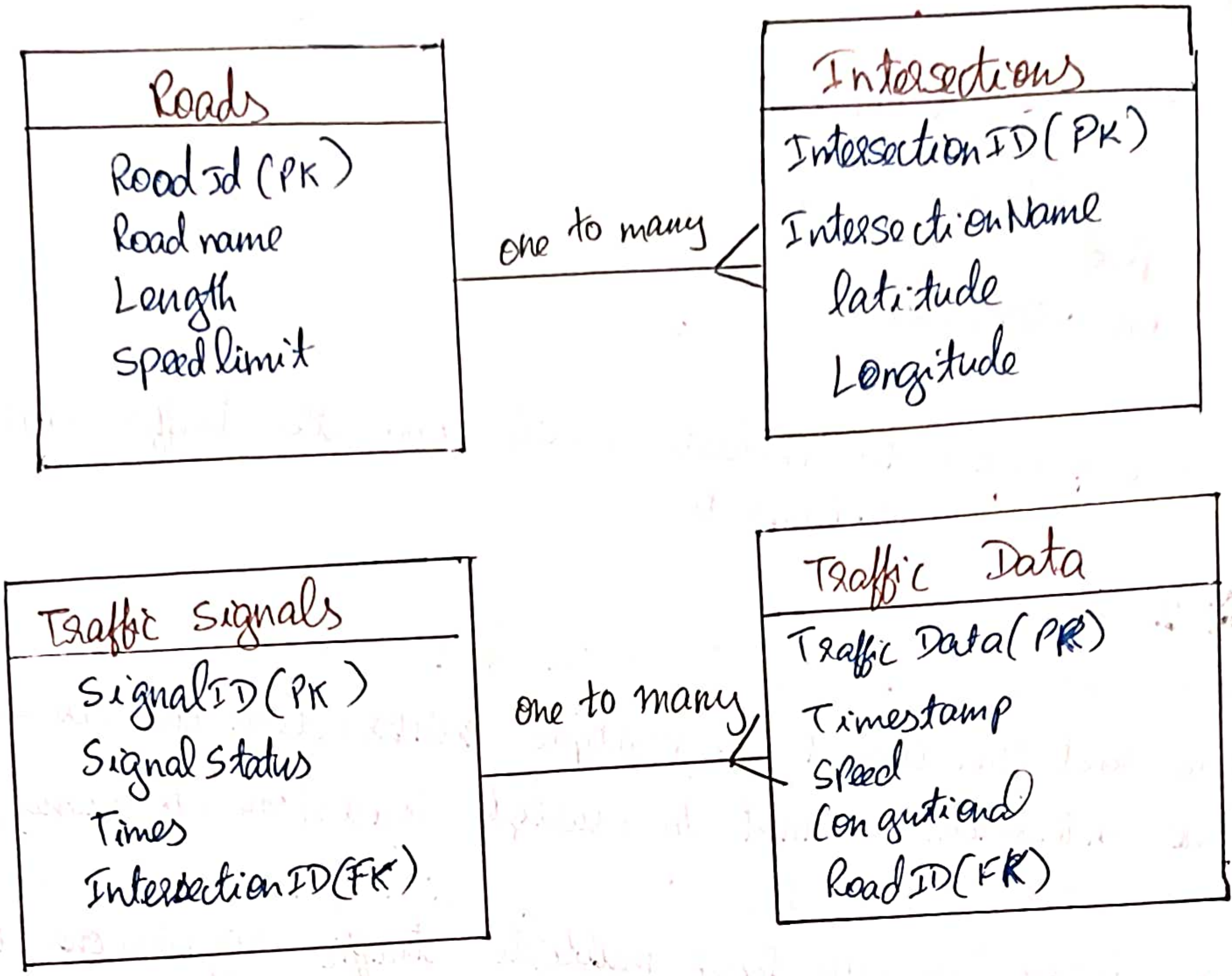
(i) one intersection can host multiple traffic signals (one-to-many)

(ii) Each traffic signal belongs to exactly one intersection (mandatory relationship)

• Traffic Data Relationship:

(i) Traffic data is associated with a specific road (one-to-many).

Task 3 : ER Diagram Design



Task 4 : Justification and Normalization

- Scalability:

The design supports scalability by clearly defining entities and relationship. For example, adding new roads (or) traffic signals can be easily accommodated without redesigning the core structure.

- Real-time Data Processing:

The inclusion of 'Traffic Data' entity allows capturing ~~and~~ real time information such as speed and congestion level crucial for dynamic traffic management.

- efficient Traffic Management:
  Relationship like 'Traffic Data' to Road enable effective monitoring and analysis of traffic condition and signal control.

Normaliyation:

- 1NF:
  Attributes are atomic and each table has a Primary Key.
- 2NF:
  No Partial dependencies, all non-Key attributes depend on the whole Primary Key.
- 3NF
  No transitive dependencies, all non-Keys attributes depends only on the Primary Key.

2) Question 1 ;

Top 3 Departments with Highest Average Salary.

Task : write sql query to find the top 3 department with the highest average salary of employees. Ensure department with no employe show an average salary of NULL

```
 WITH DeptAvgsalary ABC
 ~~WITH Deptav~~
 SELECT
       d.DepartmentID,
       d.DepartmentName,
       AVC( salary) AS Assgsalary
  FROM
     Department d
 LEFT JOIN
     Employee  ON d.Department = d.DepartmentID
  GROUP BY
     d.DepartmentID, d.DepartmentName.

  SELECT
    DepartmentID,
    DepartmentName , AvgSalary
 FROM DeptAvgsalary
 ORDER BY Avgsalary DESC
```

Question 2 : Retrieving Hierachical Category Paths

Task : ~~Retrieving~~ Write a SQL query using recursive Common Table Expression (CTE) to retrieve all categories along with their full hierarchical Path (eg. Category > subcategory > sub-subcategory)

```
WITH RECURSIVE CategoryPaths AS C
  SELECT CategoryID, CategoryName, CAST(Category Name AS
                                   VARCHARC(255)) AS Path
  FROM categories
  WHERE ParentCatagoryID IS NULL
  UNION ALL
  SELECT C. CategoryID, C. CategoryName, CONCAT(CP.Path, '>
                                   C. category Name)
  FROM categories C
  JOIN categoryPath CP ON C. Parent Category YED = CP. categoryID
>
SELECT CategoryID, categoryName, Path AS HierarchicalPath
FROM category Path
```

Question 3 : Total Distinct Customer by Month

```
SELECT DATE-FORMAT(orderDate , '%M') AS MonthName,
          COUNT (DISTINCT (CustomID) AS customecount
FROM Order
WHERE YEAR (Orderats) = Year (current-Date)
GROUP BY MONTH (orderDate)
ORDER BY MONTH (orderDate);
```

Question 4 : Finding Closest location

Task : write a SQL query to find the element 5 location to a given Period specified by lattiude and longitude use spatial functions (or) advanced mathematical Calculation for Proximity.

```
SELECT LocationID, LocationName, Latitude, Longitude,
       SQRT(POW(Latitude - given-date2) + POW(Longitude-given
                                      long.2)) AS Distance

FROM Locations
ORDER BY Distance LIMITS;
```

Question 5 : optimizing query for order table

Task : Write a SQL query to retrive order Placed in the last 7 8 days from a large orders table, sorted by order data in descending order.

```
SELECT ORDER_IR, Order_Date, customer ID, Total Amount.
FROM Order
WHERE OrderDate >= DATE_SUB(current_Date, Interval 7 DAY)
ORDER BY OrderDate DESC:
```

3) Question 1:

Handling Division operation

Task : write an SQL block to Perform a division operation where the listen is obtained from user input Handle the ZERO_DIVIDE exception gracefully with an appropriate error message.

```
DECLARE
      numerator NUMBER = 100;
      denominator NUMBER;
      result NUMBER;
BEGIN
    denominator := &user_input;
    BEGIN
      result := numerator / denominator;
      DBMS_OUTPUT_PUT_LINE(Result: '|| result);
Exception
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE(Error: Division by zero
                                        is not allowed');
    END;
END;
```

Question 2 : updating Rows for ALL :

Task : Use the FORALL satement to update multiple rows in the Employee table based on arrays of employee ID's and Salary increment.

```
DECLARE
    TYPE emp-id-array IS TABLE of Employee.Employee%TYPE ;
    TYPE salary-inc-array IS TABLE of Employee.Salary%TYPE ;

    emp_id emp-id-array := emp-id_array(101, 102,103);
    Salary-incs salary-inc-array = salary-inc-array(1000,1500,
                                                        3000);

 BEGIN
    FORALL IN_emp-ids_COUNT
       UPDATE Employee
       SET salary = salary + salary-incs(i)
       WHERE EmployeeID = emp-ids(i);

  COMMIT :
    DBMS_OUTPUT_PUT_LINE(salary update applied Successfully);

END;
```

Question 3: Implementing Nested Table Procedure

Task: Implement a SQL Procedure that except a department as input retives employes belonging to the department stores them in a nested table type, and returned this selection as an output Parameter.

```
CREATE OR REPLACE PROCEDURE GetEmployesByDept (
      P_department_id . IN Deparment. DepartmentID %. TYPE,
       P_deployee - list OUT SYS. REFCURSOR
  )
AS
   TYPE emp-list-type is TABLE OF Employee %. ROWTYPE;
      emp-list  emp-list type = emp-list-types();
BEGIN
     SELECT *
     BULK COLLECT INTO d. emp_list
     FROM employee
     WHERE DepartmentID = P. department_id;

OPEN P. employee_list FOR
      SELECT *
      FROM TABLE (l - emp. list);

END;
```

Question 4: using cursor variable and Dynamic SQL

Task: write a SQL block demonstrating the use of cursor variable (REF CURSOR) and dynamic SQL. Declare a cursor variable for querying EmployeeID, FirstName, and LastName based on a specified salary threshold.

```
DECLARE
    TYPE emp-ref-cursor IS REF CURSOR;
    emp-cursor emp-ref-cursor;
    v-sql VARCHAR2(200);
    v-min-salary NUMBER = 50000;
    v-emp-ID Employees. Employee ID %. TYPE;
     v- first_name Employees. firstname %. TYPE;
     v- Last_name Employee . Lastname %. TYPE;
BEGIN :
v-sql = SEELECT EmployeeID, firstName, Last Name FROM
                                                    employee
                              WHERE Salary >= : min-salary;

OPEN em-cursor For v-sql USING v-min- salary;

Loop
   FETCH emp-cursor INTO v-emp-id . v. first-name,
                                             v - last name ;

   EXIT WHEN emp-cursor %. NOT FOUND;
   DBMS_OUTPUT_PUT -LINE (`Employee ID ; ||v-empid ||;
                           Name :' || v. first-name || `` || v.last-nan
```

```
END LOOP ;.
CLOSE emprossion ;
END;
```

Question 5 : Dessigning Pipelined function for sales Dad .

Task : Design a Pipelined SQL function get. sales-data that relative sales data for a given mohth and years. The function should return a table of records condainting ODRID, CustomerID and order Amount for order Placed in the specified month and year.

```
CREATE OR REPLACE FONETION get_sales_data(
      P_month NUMBER;
      P_year NUMBER
 )Return SYS-RESCORSOR PIPELINED
 AS
   TYPE sales_data IS RECORED(
       ORDER_Id order orderID%. TYPE,
       Customer_Id order customerID%. TYPE,
       OrderAmount order TotalAmount %. TYPE
   );
v.sales data . sales_data,
BEGIN
  FOR rec IN(
    SELECT orderID, customerID , TotalAmount
    FROM orders
WHERE EXTRACT (YEAR FROM orderdate )= P.;
AND EXTRACT('YEAR From order-data) = P. years

)LOOP
  V-sales-data-orderID = rec.orderID;
  V-sales-data-CustomerID = DE C .CustomerID;
  V-sales-data -orderAmount = rec.TotalAmount;
  TYPE Row(v-sales-data);

END LOOP :
  RETURN;
END;
```