SAVEETHA SCHOOL OF ENGINEERING

Affliated to AICTE | IET-UK Accreditation

Engineer to Excel

Assignment - 2

SAVEETHA SCHOOL OF ENGINEERING

Submitted by

VIMAL MAHENDRAN (192311068)

Submitted to

Dr. Christy Melwyn

Professor

Course Code: **CSA0556**

Course Name: **Database Management Systems for Relational Database**

Question 1: Top 3 Departments with Highest Average Salary

SQL Query

```
WITH DepartmentAverages AS (
  SELECT
    d.DepartmentID,
    d.DepartmentName,
    AVG(e.Salary) AS AvgSalary
  FROM
    Departments d
  LEFT JOIN Employees e ON d.DepartmentID = e.DepartmentID
  GROUP BY d.DepartmentID, d.DepartmentName
)
SELECT
  DepartmentID,
  DepartmentName,
  AvgSalary FROM
  DepartmentAverages ORDER BY
  AvgSalary DESC
LIMIT 3;
```

Explanation

- **LEFT JOIN:** Ensures that all departments are included, even if they have no employees.
- **AVG(e.Salary):** Calculates the average salary for each department.
- **ORDER BY AvgSalary DESC:** Sorts departments by average salary in descending order.
- **LIMIT 3:** Selects the top 3 departments.

Question 2: Retrieving Hierarchical Category Paths

SQL Query

```
WITH RecursiveCategories AS (
  SELECT
    CategoryID,
    CategoryName,
    CAST(CategoryName AS VARCHAR(MAX)) AS Path
  FROM
    Categories
  WHERE
    ParentCategoryID IS NULL
  UNION ALL
  SELECT
    c.CategoryID,
    c.CategoryName,
    CONCAT(rc.Path, ' > ', c.CategoryName) AS Path
  FROM
    RecursiveCategories rc
  INNER JOIN Categories c ON rc.CategoryID = c.ParentCategoryID
)
SELECT
  CategoryID,
  CategoryName,
  Path FROM
  RecursiveCategories;
```

Explanation

- **Recursive CTE:** The CTE recursively builds the hierarchical path for each category.
- **Anchor Member:** The first part of the CTE selects top-level categories (those with no parent).
- **Recursive Member:** The second part concatenates the parent's path with the current category's name to build the full path.

Question 3: Total Distinct Customers by Month

SQL Query

```
WITH Months AS (
  SELECT
    DATE_FORMAT(DATE_ADD('2024-01-01', INTERVAL m MONTH), '%Y-%m') AS Month
  FROM
   (SELECT 0 AS m UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION
ALL SELECT 4 UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL
SELECT 8 UNION ALL SELECT 9 UNION ALL SELECT 10 UNION ALL SELECT  11) AS months
)
SELECT
  m.Month,
  COUNT(DISTINCT o.CustomerID) AS CustomerCountFROM
  Months m LEFT JOIN Orders o ON DATE_FORMAT(o.OrderDate, '%Y-%m') = m.Month
GROUP BY
  m.Month;
```

Explanation

- **Months CTE:** Generates all months of the current year.
- **LEFT JOIN:** Joins the months with orders, including months with no orders.
- **COUNT(DISTINCT o.CustomerID):** Counts the number of distinct customers for each month.

Question 4: Finding Closest Locations

SQL Query (Assuming a spatial database like PostgreSQL)

```
SELECT
  LocationID,
  LocationName,
  Latitude,
  Longitude,
  ST_Distance(geography(ST_MakePoint(given_longitude, given_latitude)),
geography(ST_MakePoint(Longitude, Latitude))) AS Distance FROM
  Locations ORDER BY
  Distance
LIMIT 5;
```

Explanation

- **Spatial Functions:** Uses ST_Distance to calculate the distance between the given point and each location.
- **Geography Type:** Ensures correct distance calculation on a spherical surface.
- **ORDER BY Distance:** Sorts locations by distance.
- **LIMIT 5:** Selects the closest 5 locations.

Question 5: Optimizing Query for Orders Table

SQL Query

```
SELECT
  *FROM
  OrdersWHERE
  OrderDate >= DATE_SUB(CURDATE(), INTERVAL 7 DAY)ORDER BY
  OrderDate DESC;
```

Explanation

- **Index:** Ensure an index on the OrderDate column for efficient filtering.
- **Query Rewriting:** Consider using a time-based partitioning strategy for the Orders table to improve query performance on large datasets.
- **Limit Results:** If you only need a subset of columns, select only those columns to reduce data transfer.