SQL Advance Assignment

**Creating the Database**

1. CREATE TABLE customers (

CustomerID SERIAL PRIMARY KEY

, CustomerName VARCHAR(250) NOT NULL

, ContactName VARCHAR(250) NOT NULL

, Address VARCHAR(250) NOT NULL

, City VARCHAR(250) NOT NULL

, PostalCode VARCHAR(250)

, Country VARCHAR(250) NOT NULL)

1. CREATE TABLE Categories (

CategoryID SERIAL PRIMARY KEY

, CategoryName VARCHAR(250) NOT NULL

, Description VARCHAR(300) NOT NULL)

1. CREATE TABLE Employees (

EmployeeID SERIAL PRIMARY KEY

, LastName VARCHAR(50) NOT NULL

, FirstName VARCHAR(50) NOT NULL

, BirthDate DATE NOT NULL

, Photo VARCHAR(50) NOT NULL

, Notes VARCHAR(500) NOT NULL)

1. CREATE TABLE Shippers (

ShipperID SERIAL PRIMARY KEY

, ShipperName VARCHAR(50) NOT NULL

, Phone VARCHAR(50) NOT NULL)

1. CREATE TABLE Suppliers (

SupplierID SERIAL PRIMARY KEY

, SupplierName VARCHAR(250) NOT NULL

, ContactName VARCHAR(250) NOT NULL

, Address VARCHAR(250) NOT NULL

, City VARCHAR(50) NOT NULL

, PostalCode VARCHAR(50) NOT NULL

, Country VARCHAR(50) NOT NULL

, Phone VARCHAR(50) NOT NULL)

1. CREATE TABLE Products (

ProductID SERIAL PRIMARY KEY

, ProductName VARCHAR(50) NOT NULL

, SupplierID INTEGER REFERENCES Suppliers(SupplierID)

, CategoryID INTEGER REFERENCES Categories(CategoryID)

, Unit VARCHAR(50) NOT NULL

, Price FLOAT NOT NULL)

1. CREATE TABLE Orders (

OrderID SERIAL PRIMARY KEY

, CustomerID INTEGER REFERENCES Customers(CustomerID)

, EmployeeID INTEGER REFERENCES Employees(EmployeeID)

, OrderDate DATE NOT NULL

, ShipperID INTEGER REFERENCES Shippers(ShipperID))

1. CREATE TABLE OrderDetails (

OrderDetailID SERIAL PRIMARY KEY

, OrderID INTEGER REFERENCES Orders(OrdersID)

, ProductID INTEGER REFERENCES Products(ProductID)

, Quantity INTEGER NOT NULL)

Inserting the data from CSV files 🡪 Import the data for each table from the CSV files

Assignment:

Practice - Find the names of the top 10 customers who spent the most money, and order them showing the highest first (round the total\_sales to 2 decimal points)

Answer:

The Executed Code:

SELECT Customers.CustomerID, CustomerName, TS.Total\_Spent

FROM Customers

INNER JOIN

(SELECT CustomerID, SUM(TOR.Total\_Order) AS Total\_Spent

FROM Orders

INNER JOIN

(SELECT TP.OrderID, SUM(TP.Total\_Price) AS Total\_Order

FROM

(SELECT OrderDetails.OrderID, ROUND(CAST((Price \* Quantity) AS NUMERIC),2) AS Total\_Price

FROM Orderdetails

INNER JOIN Products

ON OrderDetails.ProductID = Products.ProductID) TP

GROUP BY TP.OrderID) TOR

ON Orders.OrderID = TOR.OrderID

GROUP BY CustomerID

ORDER BY Total\_Spent DESC

LIMIT 10) TS

ON Customers.CustomerID = TS.CustomerID

ORDER BY TS.Total\_Spent DESC

The Result:



Q1- Find the names of the top 10 employees who sold the most number of orders, and order them in descending order showing the highest first

(The result should be Employee\_FirstName, Employee\_LastName, Number\_of\_Orders

Answer:

The Executed Code:

SELECT FirstName, LastName, COUNT(O.OrderID) AS Number\_Of\_Orders

FROM Employees AS E

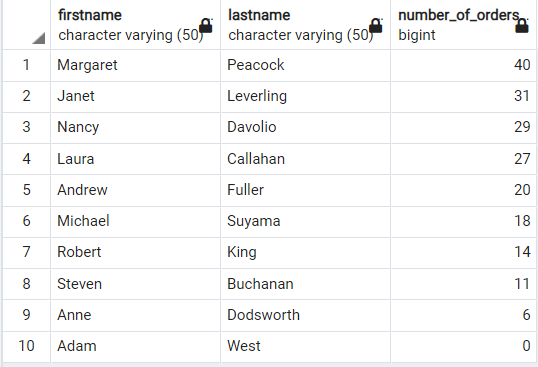
LEFT JOIN Orders AS O

ON E.EmployeeID = O.EmployeeID

GROUP BY O.EmployeeID, FirstName, LastName

ORDER BY COUNT(O.OrderID) DESC

The Result:



Q2- Find the name of the 10 employees who sold the highest quantity of sales, in “Beverages” Category, and order them is a descending order showing the highest first.

(The result should be Employee\_FirstName, Employee\_LastName, Sold\_Most\_Beverages)

Answer:

The Executed Code:

SELECT FirstName, LastName, COALESCE(M.Most\_Sold\_Beverages,0) AS Sold\_Most\_Beverages

FROM Employees AS E

LEFT JOIN

(SELECT O.EmployeeID AS EmployeeID, SUM(OD.Quantity) AS Most\_Sold\_Beverages

FROM OrderDetails AS OD

INNER JOIN Products AS P

ON OD.ProductID = P.ProductID

INNER JOIN Orders AS O

ON OD.OrderID = O.OrderID

INNER JOIN Categories AS C

ON P.CategoryID = C.CategoryID

WHERE C.CategoryName = 'Beverages'

GROUP BY O.EmployeeID) M

ON M.EmployeeID = E.EmployeeID

ORDER BY COALESCE(M.Most\_Sold\_Beverages,0) DESC

The Results:

