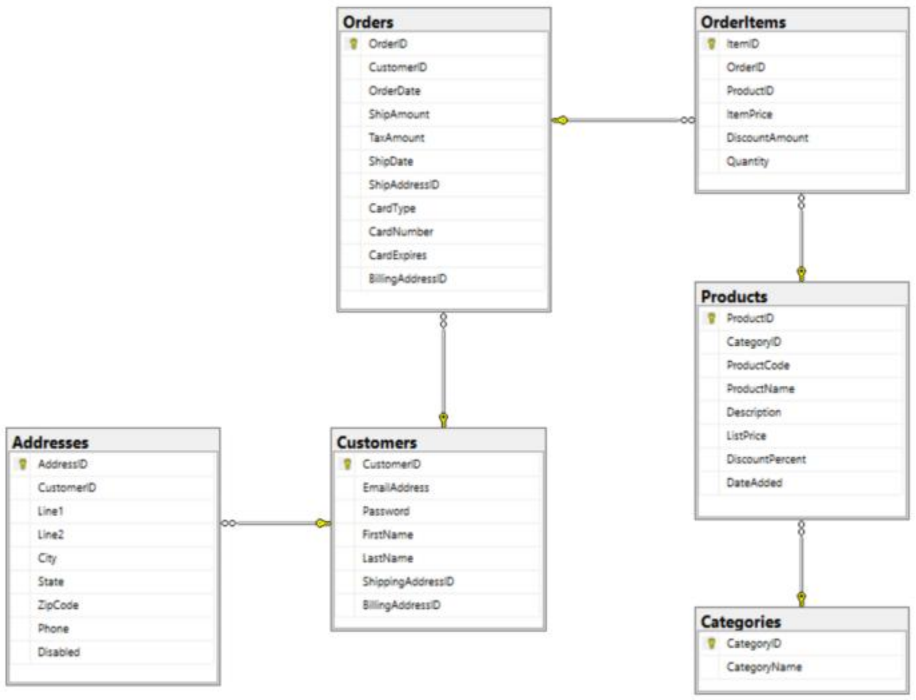


1- Write a stored procedure that when passed a category ID returns all the customers who have bought that category more than once. The SP should as well return a message that display the number of products in that category. Demonstrate using the SP for category ID =1

```
CREATE PROC Q1
@CategoryID int
AS

IF @CategoryID NOT IN (SELECT CateogryID FROM Categories)
        BEGIN
                PRINT 'Category does not exist'
                RETURN
        END

--Return the customers who have bound more than once
SELECT  C.CustomerID, FirstName, Lastname
FROM Customers C JOIN Orders O ON C.CustomerID = O.CustomerID
                        JOIN OrderItems OI ON O.OrderID = OI.OrderID
                        JOIN Products P ON OI.ProductID = P.ProductID
WHERE CategoryID =@CatogoryID
```

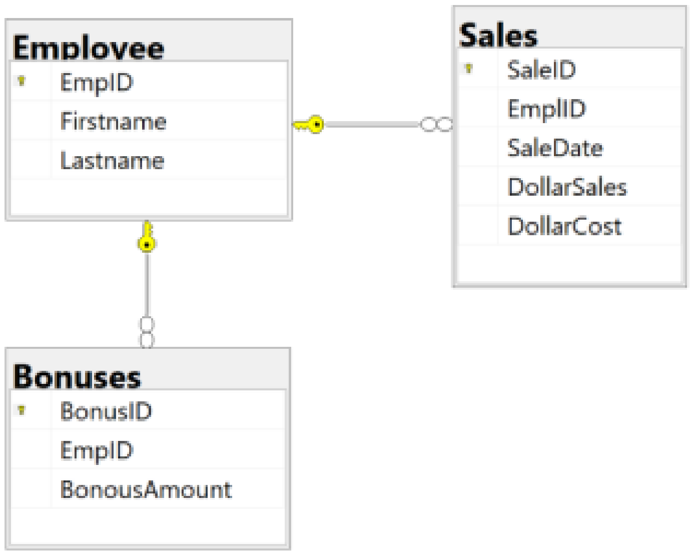
```
GROUP BY C.CustomerID, FirstName, Lastname
HAVING COUNT(*) >1




--Display a message similar to: There are 5 products in this category
DECLARE @countP int
SELECT @countP= COUNT(*)
FROM Products
WHERE CategoryID =@CatogoryID

PRINT CONCAT(‘There are ‘ , @countP, ‘ products in this category’)

--
EXEC Q1 1
```

2- Write a stored procedure that given the productid returns a message with the total quantity of sales for that product. Demonstrate using the SP for product ID =3

```
CREATE PROC Q2
@ProductID int
AS

--print a message similar to: a total of 20 quantity was sold for this product.
DECLARE @totalquant int
SELECT @totalquant =SUM(Quantity)
FROM OrderItems
WHERE ProductID = @ProductID
PRINT CONCAT(‘ A total of ‘, @totalquant, ‘ quantity was sold for this product’)
```

3- Write a store procedure that given a state returns all the customers that live there. If no customers in that state, raise an error.

```
CREATE PROC Q3
@state char(2)
AS

IF NOT EXISTS (SELECT * FROM Addresses WHERE State=@state)
        THROW 50000, ' no one living in that state', 1

SELECT  DISTINCT C.*
FROM Custemers C JOIN Addresses A ON C.CustomerID = A.CustomerID
WHERE State=@state
```



Write a SELECT query that returns the name of employees that have sold more than $3,000 in August and September of 2024.

```
SELECT E.EmpID, FirstName, Lastname, SUM(DollarSale)
FROM Employees E JOIN Sales S ON E.EmpID=S.EmplID
WHERE SaleDate BETWEEN '1/8/2024' AND '30/9/2024'
--WHERE YEAR(SaleDate)=2024 AND MONTH(SaleDate) IN (8,9)
--WE CAN NOT WRITE IT LIKE THIS:
--WHERE MONTH(SaleDate)= 8 OR 9 → fix it WHERE MONTH(SaleDate)= 8 OR
MONTH(SaleDATE)=9
GROUP BY E.EmpID, FirstName, Lastname
HAVING SUM(DollarSale) >3000
```

Write a SELECT query to return the employee with the most bonus

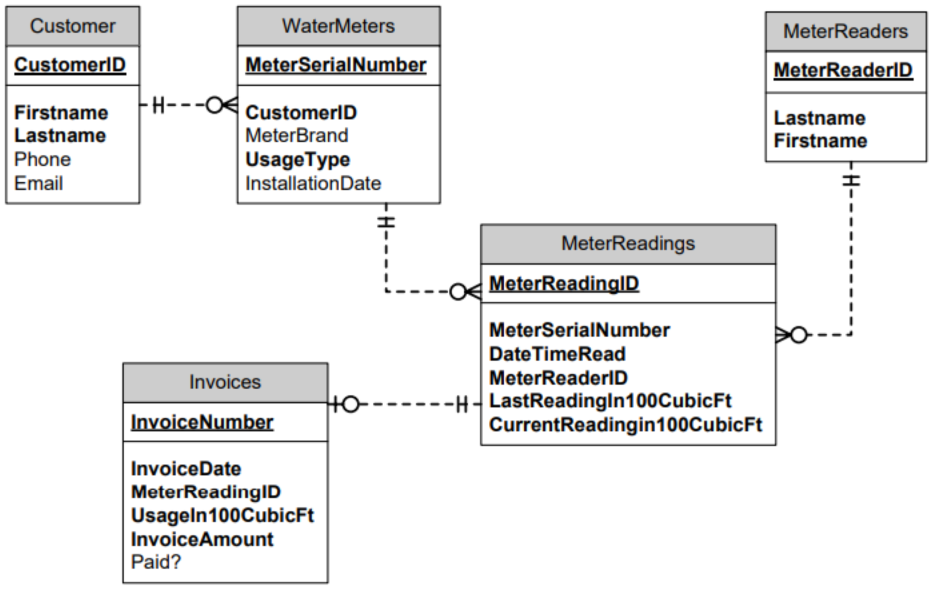
```
SELECT TOP 1 Firstname, Lastname, SUM(BonusAmount)
FROM Employees E JOIN Bonuses B ON E.EmpID = B.EmpID
GROUP BY Firstname, Lastname
ORDER BY SUM(BonusAmount) DESC
```

Create a stored procedure that given the EmpID and month returns all following:

• A table (EmpID, Name, Total sales amount, Number of sales) for the specified employee during that month of current year.

• A message similar to: “There were a total of 5 transactions adding up to $40,000 in sales.”

```
CREATE PROC Q6
@EmpID int,
@month int
AS

-- EmpID, Name, Total sales amount, Number of sales
SELECT E.EmplID, Firstname + ‘ ‘ + Lastname AS Name, SUM(DollarSales), COUNT(*)
FROM Employees E JOIN Sales S ON E.EmpID=S.EmplID
WHERE E.EmplID=@EmpID AND MONTH(SaleDate)=@month AND
YEAR(SaleDate)=YEAR(GETDATE())
GROUP BY E.EmplID, Firstname, lastname

--Print There were a total of 5 transactions adding up to $40,000 in sales.”
DECLARE @countt int, @totalt money

SELECT @totalt =SUM(DollarSales), @countt =COUNT(*)
FROM Employees E JOIN Sales S ON E.EmpID=S.EmplID
WHERE E.EmplID=@EmpID AND MONTH(SaleDate)=@month AND
YEAR(SaleDate)=YEAR(GETDATE())


PRINT CONCAT (‘There are a total of ‘, @countt, ‘ transactions adding up to ‘, @totalt, ‘
in sales’)
```



Which customers have more than one water meter installed?

SELECT Firstname, Lastname, COUNT(*)
FROM Customer C JOIN WaterMeters WM ON C.CustomerID = WM.CustomerID
GROUP BY Firstname, Lastname
HAVING COUNT(*) >1

Which brands have an average usage greater than 500 cubic feet? (Hint: Usage= CurrentReadingin100CubicFt - LastReadingIn100CubicFt)

SELECT MeterBrand , AVG(CurrentReadingin100CubicFt - LastReadingIn100CubicFt)
FROM WaterMeters WM JOIN MeterReadings MR ON WM.Serial#=MR.Serial#
GROUP BY MeterBrand
HAVING AVG(CurrentReadingin100CubicFt - LastReadingIn100CubicFt) >500

Which customers have water meters installed after January 1, 2020, and belong to the 'Residential' usage type?

SELECT C.*
FROM Customer C JOIN WaterMeters WM ON C.CustomerID = WM.CustomerID
WHERE MeterInstalled > ‘1/1/2020’ AND usagetype = ‘Residential’