USE AdventureWorks2022;

-- TASK 1

/\*

1. Provide information about the global sales amount (money), number of orders and volume

(items sold) of the AdventureWorks business.

\*/

-- OrderQty smallint Quantity ordered per product

-- UnitPrice money Selling price of a single product.

SELECT SUM(SOH.TotalDue) 'SALES AMOUNT',

SUM(SOD.OrderQty) 'VOLUME',

COUNT(SOH.SalesOrderID) AS 'NUMBER OF ORDERS' FROM

Sales.SalesOrderDetail AS SOD INNER JOIN Sales.SalesOrderHeader AS SOH

ON SOD.SalesOrderID = SOH.SalesOrderID

-- TASK 2

/\*

Provide information about the sales amount, volume, and number of orders in individual years

of operation of the business.

\*/

SELECT YEAR(SOH.OrderDate) 'YEAR', SUM(SOH.TotalDue) 'SALES AMOUNT',

SUM(SOD.OrderQty) 'VOLUME',

COUNT(SOH.SalesOrderID) AS 'NUMBER OF ORDERS' FROM

Sales.SalesOrderDetail AS SOD INNER JOIN Sales.SalesOrderHeader AS SOH

ON SOD.SalesOrderID = SOH.SalesOrderID

GROUP BY YEAR(SOH.OrderDate)

ORDER BY YEAR(SOH.OrderDate)

-- TASK 3

/\*

Prepare a SQL query that provides top 5 customers with the highest number of orders, try

using the customer name (it might be tricky).

\*/

/\*

    USE MAX TO AVOID THE ERROR:

    Msg 8120, Level 16, State 1, Line 48

    Column 'Sales.vIndividualCustomer.FirstName' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

    Msg 8120, Level 16, State 1, Line 48

    Column 'Sales.vIndividualCustomer.LastName' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

\*/

SELECT TOP 5 SOH.CustomerID 'CustomerID', CONCAT(MAX(Customer.FirstName), ' ', MAX(Customer.LastName)) 'Full name',COUNT(SOH.SalesOrderID) 'Number Of Orders'

FROM Sales.SalesOrderHeader AS SOH

INNER JOIN Sales.vIndividualCustomer AS Customer

ON SOH.CustomerID = Customer.BusinessEntityID

GROUP BY SOH.CustomerID

ORDER BY COUNT(SOH.SalesOrderID) DESC

-- TASK 4

/\*

Prepare a SQL query that provides the names of all individual customers with the total sum of

purchases (use SalesOrderHeader.SubTotal) greater than USD 1500 sorted (descending)

by the total sales amount.

\*/

SELECT SOH.CustomerID 'CustomerID', CONCAT(MAX(Customer.FirstName), ' ', MAX(Customer.LastName)) 'Full name', SUM(SOH.TotalDue) 'Sales Amount'

FROM Sales.SalesOrderHeader AS SOH

INNER JOIN Sales.vIndividualCustomer AS Customer

ON SOH.CustomerID = Customer.BusinessEntityID

GROUP BY SOH.CustomerID

HAVING SUM(SOH.TotalDue) > 1500

ORDER BY SUM(SOH.TotalDue) DESC

-- TASK 5

/\*

Prepare a query that provides information about average price, total sales amount, and total

volume in individual product categories of the AdventureWorks business.

\*/

SELECT

    PPC.ProductCategoryID 'CategoryID',

    MAX(PPC.Name) 'Category Name',

    AVG(SOD.UnitPrice) 'Average Price',

    SUM(SOD.UnitPrice \* SOD.OrderQty) 'TOTAL SALES AMOUNT',

    SUM(SOD.OrderQty) 'VOLUME'

    FROM

Sales.SalesOrderHeader AS SOH

INNER JOIN Sales.SalesOrderDetail AS SOD

ON SOH.SalesOrderID = SOD.SalesOrderID

INNER JOIN  Production.Product AS PP

ON SOD.ProductID = PP.ProductID

INNER JOIN Production.ProductSubcategory AS PPS

ON PP.ProductSubcategoryID = PPS.ProductSubcategoryID

INNER JOIN Production.ProductCategory AS PPC

ON PPS.ProductCategoryID = PPC.ProductCategoryID

GROUP BY PPC.ProductCategoryID

ORDER BY PPC.ProductCategoryID

-- there is some number of records where there is no ProductSubcategoryID

SELECT \* FROM Production.Product as p WHERE p .ProductSubcategoryID is null;

-- TASK 6

/\*

    Display all subcategories whose average price is higher than the average price of all

    categories.

\*/

-- (JUST USING AVERAGE UNIT PRICE, SINCE AVERAGE PRICE OVER ALL CATEGORIES)

SELECT

    PPS.ProductSubcategoryID AS 'SubcategoryID',

    MAX(PPS.Name) AS 'Subcategory name',

    AVG(SOD.UnitPrice) AS 'Average Price',

    (SELECT AVG(Sales.SalesOrderDetail.UnitPrice) FROM Sales.SalesOrderDetail) 'AvgPrices'

FROM

    Sales.SalesOrderHeader AS SOH

    INNER JOIN Sales.SalesOrderDetail AS SOD ON SOH.SalesOrderID = SOD.SalesOrderID

    INNER JOIN Production.Product AS PP ON SOD.ProductID = PP.ProductID

    INNER JOIN Production.ProductSubcategory AS PPS ON PP.ProductSubcategoryID = PPS.ProductSubcategoryID

GROUP BY

    PPS.ProductSubcategoryID

HAVING

    AVG(SOD.UnitPrice) > (SELECT AVG(Sales.SalesOrderDetail.UnitPrice) FROM Sales.SalesOrderDetail)

ORDER BY PPS.ProductSubcategoryID

-- FIND THE AVERAGE OF AVERAGES FOR ALL CATEGORIES( IT IS USED INSIDE OF PREVIOUS QUERY) -- HERE THE AVERAGE VALUE IS DIFFERENT BECAUSE NULL VALUES ARE NOT INCLUDED FOR CATEGORIES

SELECT AVG(AveragePrice) AS 'Average of Averages'

FROM (

    SELECT AVG(SOD.UnitPrice) AS 'AveragePrice'

    FROM Sales.SalesOrderHeader AS SOH

    INNER JOIN Sales.SalesOrderDetail AS SOD ON SOH.SalesOrderID = SOD.SalesOrderID

    INNER JOIN Production.Product AS PP ON SOD.ProductID = PP.ProductID

    INNER JOIN Production.ProductSubcategory AS PPS ON PP.ProductSubcategoryID = PPS.ProductSubcategoryID

    INNER JOIN Production.ProductCategory AS PPC ON PPS.ProductCategoryID = PPC.ProductCategoryID

    GROUP BY PPC.ProductCategoryID

) AS AvgPrices;

-- TASK 7

/\*

    Select sales territory (name) with sales in May 2013 higher than the average monthly sales

    per sales territory.

\*/

-- monthly sales in may 2013

WITH MonthlySales AS (

    SELECT

        SOH.TerritoryID,

        MAX(SST.Name) AS TerritoryName,

        SUM(SOH.TotalDue) AS MonthlySales

    FROM

        Sales.SalesOrderHeader AS SOH

        INNER JOIN Sales.SalesTerritory AS SST ON SOH.TerritoryID = SST.TerritoryID

    WHERE

        YEAR(SOH.OrderDate) = 2013

        AND MONTH(SOH.OrderDate) = 5

    GROUP BY

        SOH.TerritoryID

),

-- average sales in each sale territory for entire period of time

AverageSales AS(

    SELECT

        SOH.TerritoryID,

        AVG(SOH.TotalDue) AS AvgMonthlySales

    FROM

        Sales.SalesOrderHeader AS SOH

    GROUP BY

        SOH.TerritoryID

)

SELECT

    MS.TerritoryID,

    MAX(MS.TerritoryName) AS 'Territory Name',

    MS.MonthlySales AS 'Sales May 13',

    AvS.AvgMonthlySales AS 'Average Monthly Sales'

FROM

    MonthlySales AS MS

INNER JOIN

    AverageSales AS AvS ON MS.TerritoryID = AvS.TerritoryID

GROUP BY

    MS.TerritoryID, MS.MonthlySales, AvS.AvgMonthlySales

HAVING

    MS.MonthlySales > AvS.AvgMonthlySales;

/\*

So, your task involves querying the Adventure Works database to calculate the average number of orders made by customers within each sales territory, considering only customers

who have made more than 10 orders in total. Then, you need to provide both the real value of this average and the largest integer less than this average for each sales territory.

\*/

WITH CustomersWithMoreThanTenOrders AS (

    SELECT

        SOH.CustomerID,

        COUNT(\*) AS numberOfOrders,

        MAX(SOH.TerritoryID) AS territoryID

    FROM

        Sales.SalesOrderHeader AS SOH

    GROUP BY

        SOH.CustomerID

    HAVING

        COUNT(\*) > 10

)

SELECT

    C.territoryID,

    AVG(CAST(C.numberOfOrders AS DECIMAL(10,2))) AverageNumberOfOrders,

    FLOOR(AVG(C.numberOfOrders)) AverageNumberOfOrdersINT

FROM

    CustomersWithMoreThanTenOrders AS C

GROUP BY

    C.territoryID;

-- TASK 9

/\*

Show monthly sales amount by each sales territory in 2013 and calculate the difference with

the previous month (use 0 for 12/2012) to identify trends.

\*/

WITH MonthlySales AS (

    SELECT

        SOH.TerritoryID,

        MAX(SST.Name) AS TerritoryName,

        MONTH(SOH.OrderDate) AS Month,

        SUM(SOH.TotalDue) AS MonthSalesAmount

    FROM

        Sales.SalesOrderHeader AS SOH

    INNER JOIN

        Sales.SalesTerritory AS SST ON SOH.TerritoryID = SST.TerritoryID

    WHERE

        YEAR(SOH.OrderDate) = 2013

    GROUP BY

        SOH.TerritoryID,

        MONTH(SOH.OrderDate)

),

MonthlySalesWithPrev AS (

    SELECT

        TerritoryID,

        TerritoryName,

        Month,

        MonthSalesAmount,

        LAG(MonthSalesAmount, 1, 0) OVER (PARTITION BY TerritoryID ORDER BY Month) AS PrevMonthSalesAmount

    FROM

        MonthlySales

)

SELECT

    TerritoryID,

    TerritoryName,

    Month,

    MonthSalesAmount,

    MonthSalesAmount - PrevMonthSalesAmount AS SalesDifferenceWithPrevMonth

FROM

    MonthlySalesWithPrev;