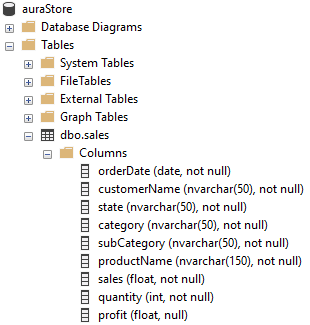
**SQL Analysis**

--using the new database

USE auraStore



--showing the imported table

SELECT \* FROM sales

--adding a new column

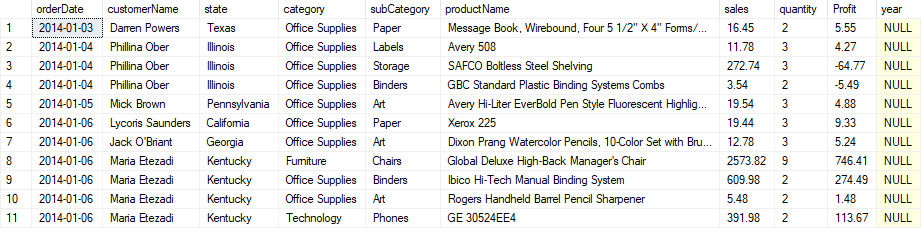
BEGIN TRAN

ALTER TABLE sales

ADD [year] INT

SELECT \* FROM sales

COMMIT TRAN



--getting the year

BEGIN TRAN

UPDATE

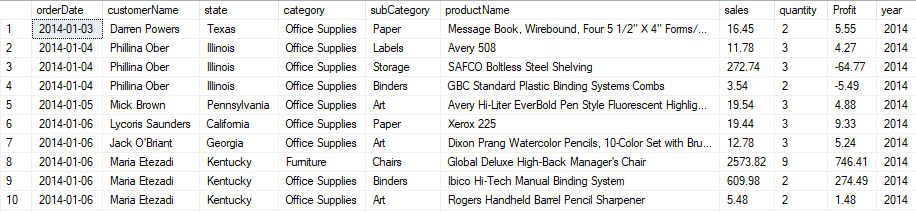
sales

SET

[year] = YEAR(orderDate)

SELECT \* FROM sales

COMMIT TRAN



--looking for duplicated rows

--Count of the rows

SELECT

COUNT(\*) AS TotalRows

FROM

Sales



--adding a number to duplicated rows

BEGIN TRAN;

SELECT

\*,

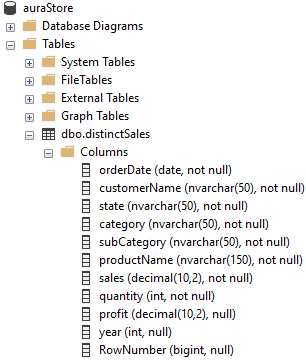
ROW\_NUMBER() OVER (PARTITION BY orderDate, customerName, state, category, subCategory, productName, sales, quantity, Profit, [year] ORDER BY orderDate) AS RowNumber

INTO distinctSales

FROM sales

SELECT \* FROM distinctSales

COMMIT TRAN





--deleting the duplicated rows, the rows with the row number 2

BEGIN TRAN;

DELETE

FROM

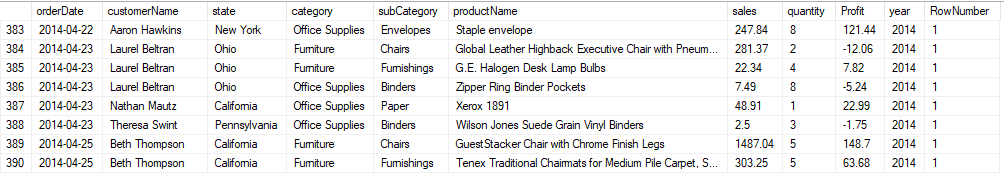
distinctSales

WHERE

RowNumber = 2

SELECT \* FROM distinctSales

COMMIT TRAN



--we dont need the row number column anymore

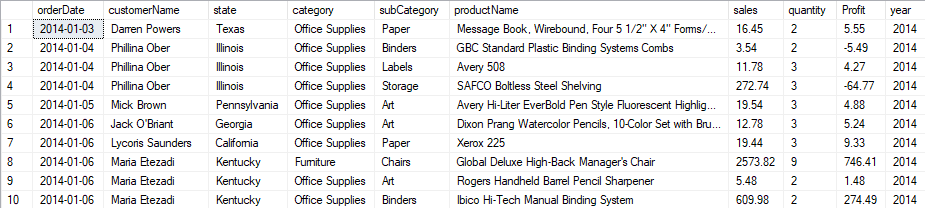
BEGIN TRAN;

ALTER TABLE distinctSales

DROP COLUMN RowNumber;

SELECT \* FROM distinctSales

COMMIT TRAN



--count without duplicates

SELECT

COUNT(\*) AS TotalRows

FROM

distinctSales



--now we work with the table distinctSales

--sales by subcategory

SELECT

subCategory,

SUM(sales) AS sumOfSales

FROM

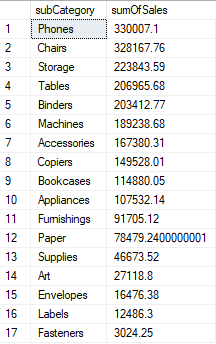
distinctSales

GROUP BY

subCategory

ORDER BY

sumOfSales DESC



--sales by subcategory with currency format

SELECT

subCategory,

SUM(sales) AS sumOfSales,

FORMAT(SUM(sales), 'C') AS sumOfSalesFormatted

FROM

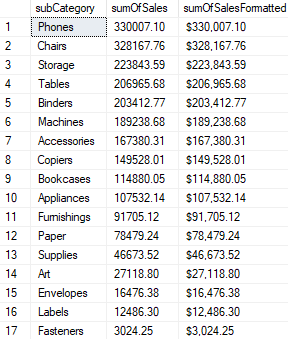
distinctSales

GROUP BY

subCategory

ORDER BY

sumOfSales DESC



--sales by subcategory

SELECT

year,

category,

ROUND(SUM(Profit),0) AS sumOfProfit

FROM

distinctSales

GROUP BY

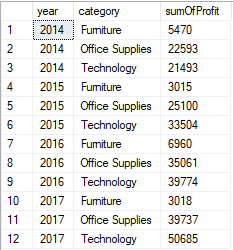
year,

category

ORDER BY

year ASC,

category ASC



--monthly sales with all the years

WITH monthAdded AS (

SELECT

\*,

FORMAT(

orderDate,

'MMMM'

) AS month

FROM

distinctSales

)

SELECT

month,

ROUND(SUM(sales),0) AS sumOfSales

FROM

monthAdded

GROUP BY

month

ORDER BY

sumOfSales DESC



--we create a store procedure to get the sales By Month And Year

CREATE PROCEDURE salesByMonthAndYear

@year INT

AS

BEGIN

WITH monthAdded AS (

SELECT

\*,

FORMAT(

orderDate,

'MMMM'

) AS month

FROM

distinctSales

)

SELECT

month,

ROUND(SUM(sales),0) AS sumOfSales

FROM

monthAdded

WHERE

year = @year

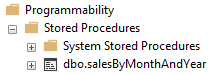
GROUP BY

month

ORDER BY

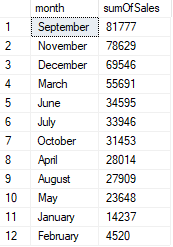
sumOfSales DESC

END



--exceuting the store procedure

EXEC salesByMonthAndYear @year = 2014



--top 5 customers making profits

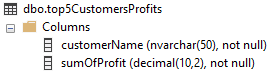
--we create a new table to insert the top 5 customers making profits

CREATE TABLE [dbo].[top5CustomersProfits] (

customerName NVARCHAR(50) NOT NULL,

sumOfProfit DECIMAL(10,2) NOT NULL

)



--showing the table

SELECT \* FROM top5CustomersProfits



--we get the sum of profit by the top 5 customers to later transform it to percentages

BEGIN TRAN

INSERT INTO top5CustomersProfits (customerName, sumOfProfit)

SELECT customerName, sumOfProfit

FROM (

SELECT

TOP 5 customerName,

SUM(Profit) AS SumOfProfit

FROM

distinctSales

GROUP BY

customerName

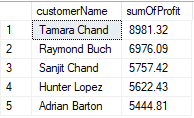
ORDER BY

SumOfProfit DESC

) AS subquery

SELECT \* FROM top5CustomersProfits

COMMIT TRAN



--we sum the profits of the top 5 customers

DECLARE @totalProfit DECIMAL(10,2);

SET @totalProfit = (SELECT SUM(sumOfProfit) FROM top5CustomersProfits)

--we get the percentage of each one

SELECT

customerName,

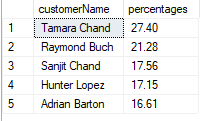
CAST((sumOfProfit / @totalProfit \* 100) AS DECIMAL(10,2)) AS percentages

FROM

top5CustomersProfits

ORDER BY

percentages DESC



--sales by state and category

--with all the categories

SELECT

state,

ROUND(SUM(sales),0) AS SumOfSales

FROM

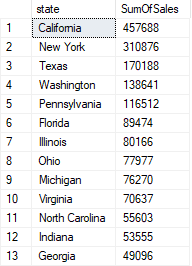
distinctSales

GROUP BY

state

ORDER BY

SumOfSales DESC



--we create a store procedure to get the sales by state and category

CREATE PROCEDURE salesByStateAndCategory

@category NVARCHAR(50)

AS

BEGIN

SELECT

state,

ROUND(SUM(sales),0) AS SumOfSales

FROM

distinctSales

WHERE

category = @category

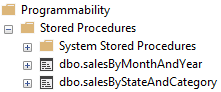
GROUP BY

state

ORDER BY

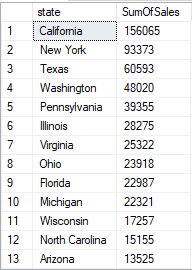
SumOfSales DESC

END



--exceuting the store procedure

EXEC salesByStateAndCategory @category = 'Furniture'



--count of distinct customer by year

--adding a number to duplicated rows

--Start a transaction, the temporary table stay and can be manipulated until the end of connection, it is located in the tempdb database

BEGIN TRANSACTION;

--Create a temporary table within the transaction

CREATE TABLE #customerNameDuplicates (year INT, customerName NVARCHAR(50), rowNumber INT);

SELECT \* FROM #customerNameDuplicates



--we insert a row number to identify the duplicated customers

INSERT INTO #customerNameDuplicates (year, customerName, rowNumber)

SELECT year, customerName, rowNumber

FROM (

SELECT

year,

customerName,

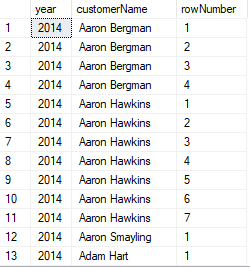
ROW\_NUMBER() OVER (PARTITION BY year, customerName ORDER BY year ASC, customerName ASC) AS rowNumber

FROM distinctSales

) AS subquery

--showing the table

SELECT \* FROM #customerNameDuplicates ORDER BY year ASC, customerName ASC



--eliminate the duplicates

DELETE

FROM

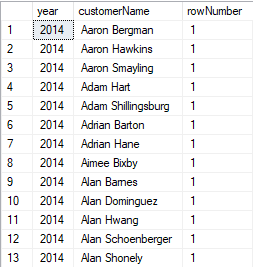
#customerNameDuplicates

WHERE

rowNumber > 1

--showing distinct customers only

SELECT \* FROM #customerNameDuplicates ORDER BY year ASC, customerName ASC



--we dont need the row column anymore

ALTER TABLE #customerNameDuplicates

DROP COLUMN rowNumber;

--showing the table

SELECT \* FROM #customerNameDuplicates ORDER BY year ASC, customerName ASC



--getting distinct customers by year with the temporary table

SELECT

year,

COUNT(\*) AS CountOfDistinctCustomers

FROM

#customerNameDuplicates

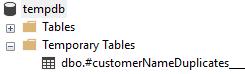
GROUP BY

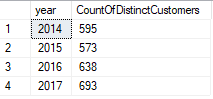
year

ORDER BY

year ASC

COMMIT TRAN





--percentages for pareto table by sales and subCategory

BEGIN TRANSACTION;

--we create a temporary table

CREATE TABLE #salesBySubCtg(subCategory NVARCHAR(50), sales FLOAT, salesPercentage DECIMAL(10,2));

SELECT \* FROM #salesBySubCtg



--create a variable with the sum of sales

DECLARE @sumTotalSales DECIMAL(10,2);

SET @sumTotalSales = (SELECT SUM(sales) FROM distinctSales);

--getting the percentages by subCategory

INSERT INTO #salesBySubCtg (subCategory, sales, salesPercentage)

SELECT subCategory, sumOfSales, salesPercentage

FROM (

SELECT TOP 17 --use top 17 to use order by in subqueries, inline funcions, etc

subCategory,

SUM(sales) AS sumOfSales,

CAST((SUM(sales) / @sumTotalSales \* 100) AS DECIMAL(10,2)) AS salesPercentage

FROM

distinctSales

GROUP BY

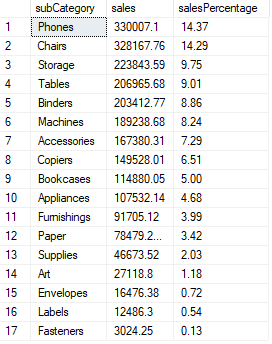
subCategory

ORDER BY

salesPercentage DESC

) AS subquery

SELECT \* FROM #salesBySubCtg



--to do a acumulative sum of percentages we asign numbers to the rows

WITH numbered\_data AS (

SELECT \*,

ROW\_NUMBER() OVER (ORDER BY salesPercentage DESC) AS row\_num

FROM #salesBySubCtg

)

--doing the acumulative sum

SELECT

subCategory,

sales,

salesPercentage,

SUM(salesPercentage) OVER (ORDER BY row\_num) AS cumulative\_number

FROM numbered\_data

ORDER BY cumulative\_number

COMMIT TRAN

