https://dataedo.com/download/AdventureWorks.pdf.

TASK 1

1. Specify the set of rules and domain constraints.
2. Provide a revised and complete version of the data model (complete UML class diagram).
3. Create a logical data model as (omitting, if possible, native SQL implementation constructs). Further, test the created data model.
4. Create a physical data model as a DDL SQL script (including domain rules and constraints) while trying to comply with the SQL standard (omitting, if possible, native SQL implementation constructs). Create a database in MS SQL Server. Implemented database should be a physical data model of the modeled part of the reality.
5. Test the created database. Enter several records into each table, checking the correctness of the implementation (remember to check both correct data and inconsistent data with the applicable rules – please comment and explain the obtained messages from the DBMS system)

TASK 1 - SOLUTIONS:

Use this section to provide your solutions:

1 DOMAIN RULES

Since the provided conceptual data model lacks a store table based on the specified set of rules and constraints, I modified the model as adding following table:

Store Table:

- *StoreID (PK)*
- *StoreName NOT NULL*
- *StoreLocation*

Constraints:

- *Every store must have a unique storeID.*
- *StoreName cannot be NULL.*

Relationships: We should establish a relationship between the order table and the Store table indicating which store is which order to belong to.

Customer table -- Store table \\ Store table – product table

R01 - A customer can repeatedly shop in the same store.

- *This indicates that customer can make multiple purchases from the same store. There should be one-to-many relationship between the Customer table and the Store table. Because each customer may make multiple purchases from the same store.*

Customer table 

Store table

0.. represents that each customer can shop at multiple stores.*
1 represents customer can repeatedly shop in the same store.

R02 - <u>Any customer can shop in any store.</u>

- *This indicates that customer have opportunity to choose any store to shop. There should be many-to-many relationship between the Customer table and Store table. Because each customer may shop at multiple stores also each of the store may have multiple customers.*

Customer table - 

- Store table

R03 <u>- Each purchase is made by the customer in the store on a specific date and time</u>.

- *This indicates that each of the purchase is associated wit the specific customer, store, and order date time. Orders needs timestamp to capture the purchase time. Order table has an FK relationship to the Customer table(CustomerID) and store table with the date and time of the order column by OrderDate().*

R04 <u>- A store must offer at least one or more product</u>.

- *This indicates that all the stores must have at least one product available for a purchase. There should be one-to-many relationship between the Store table and Product table. Because each of the store may offer multiple products and purchase made from one store.*

R05 - <u>The same product (type) can be offered by multiple stores</u>.

- *This indicates that multiple stores may offer the same type of product. There should be one-to-many relationship (one mandatory or more) between the Store table and Product table. Because each product can be offered by multiple stores and each store services offered the multiple of product.*

R06 - <u>Each store can individually propose the price and quantity of the offered product.</u>

- *This indicates that each store has the autonomy to set its own depending price and quantity for the products It offers. The given rules rule is implying attributes within the relationship between the Store table and the Product table to store the price and quantity information specific to each store offerings.*

COMMENTS:

After incorporating the Store table and update the rules and relationships accordingly, we are able to ensure that the planned conceptual data model is accurately reflects the domain requirements and the supports the given rules.

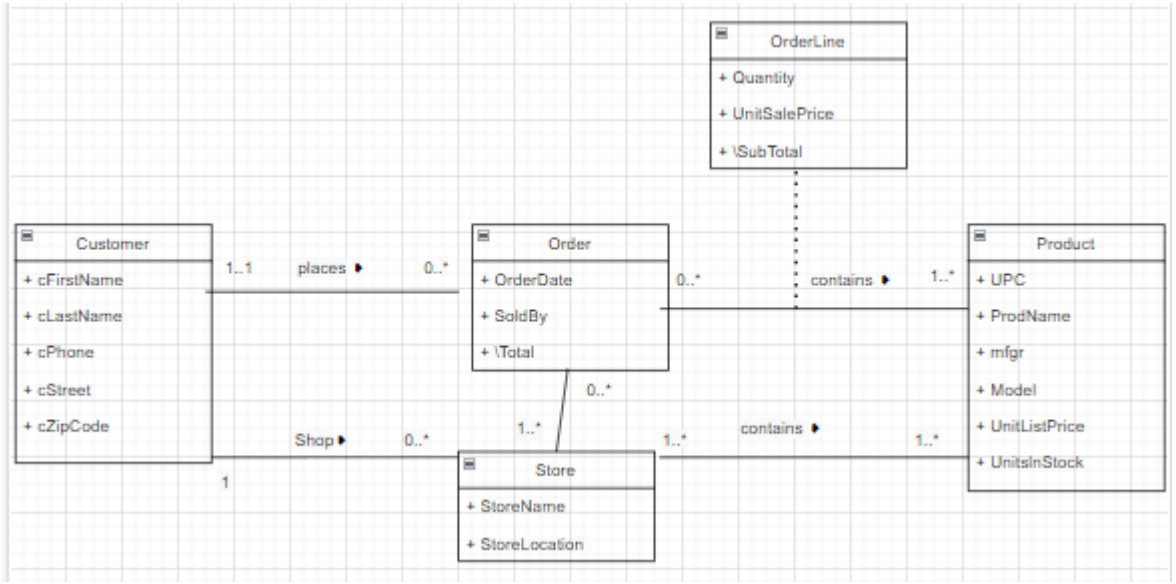


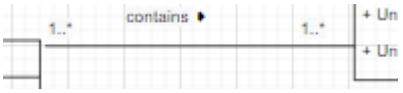
Figure 1 Conceptual Data Model, UML

COMMENTS:

There is above presentation of re-created Conceptual Data Model image from app.diagrams.net.

We want customer to be able to shop in same store so customer table 

store table,means 1 customer may visit same store. Or multiple store. So we can define as provided.

Store must offer one or more product and same product can be offered by multiple stores are presented as following: store table 

Product table

I added Store table to connect them with customer, order and product table to assscіate with them.

Customer can place multiple orders (Customer to order) -> CustomerID FK in the Order Table.

Each order contains one or more products (Order to OrderLine table) ->ORDERID FK in the OrderLine table.

One customer can shop in multiple store as presented association.

Products can be sold in multiple stores, and stores can sell many products (Store to Product table).

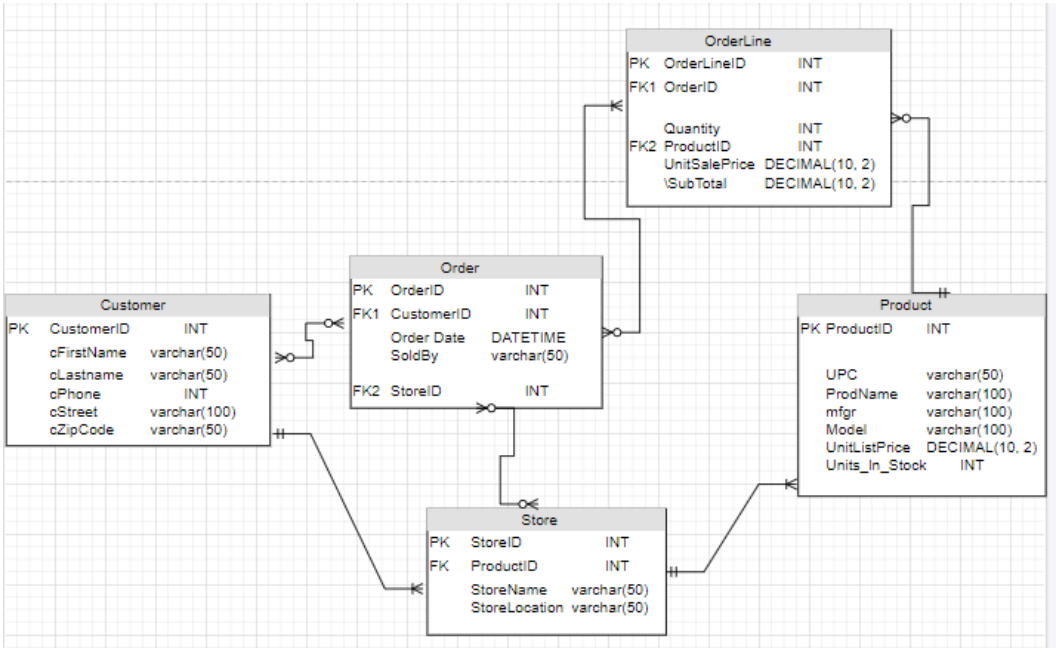


Figure 2 Logical Data Model

Above image is representation as Logical Data model of our given conceptual Model created as database diagram in SSMS.

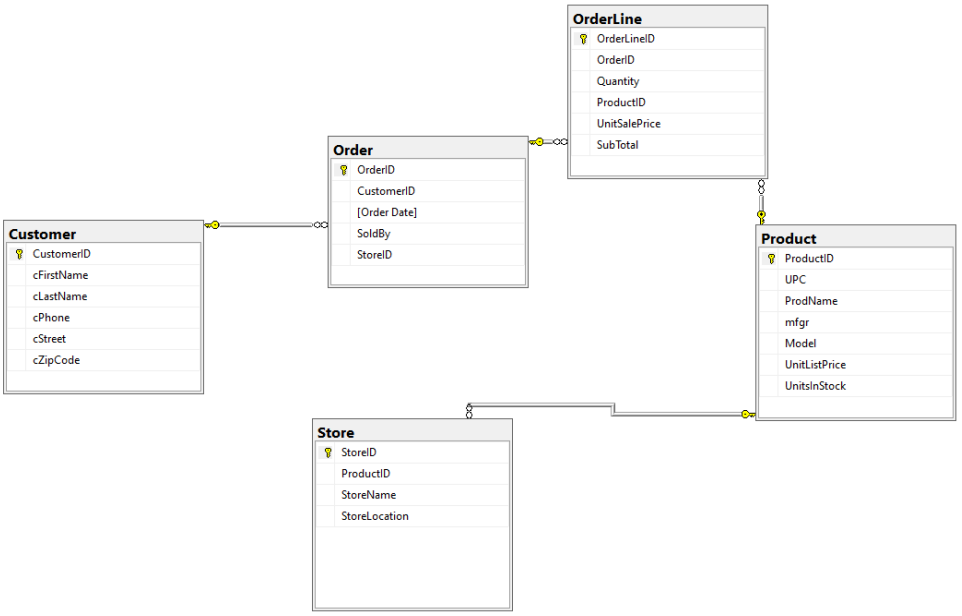


Figure 3 Physical Data Model

COMMENTS:

Order: Contains information about orders. Each order is identified by an OrderID, and each order has details such as CustomerID, StoreID, OrderDate, and Total (total amount).

Customer: Contains information about customers. CustomerID identifies each customer, and it includes personal information such as FirstName, LastName, Phone, Street, and ZipCode.

Order_Line: Contains details of the order. It is identified by the combination of OrderID and UPC (Universal Product Code), and each line includes Quantity, UnitSalePrice, and Subtotal.

Product: Contains information about products. Each product is uniquely identified by UPC, and it includes details such as ProdName (product name), Mfg (manufacturer), Model, UnitListPrice (unit list price), and UnitInStock (units in stock).

Store_Product: Contains stock information of a store. It is identified by StoreID and UPC, and each record includes Price and StockQuantity.

Store: Contains store information. Each store is identified by a StoreID, and it includes Name and Location.

The relationships in the schema can be understood as follows:

Each Order can have multiple Order_Line entries (multiple products can be sold in one order).

Each Customer can have multiple Order entries (a customer can place multiple orders).

Each Order_Line is associated with a single Product (each order line represents one product).

Each Product can be associated with multiple Store_Product entries (a product can be found in different stores).

Each Store_Product is associated with a single Store (each stock entry represents a store).

4 SQL IMPLEMENTATION

```sql
CREATE DATABASE lAB02;

use LAB02;

CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    cFirstName VARCHAR(50),
    cLastName VARCHAR(50),
    cPhone varchar(15),
    cStreet VARCHAR(100),
    cZipCode VARCHAR(50),

);

CREATE TABLE [Order] (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    [Order Date] DATETIME,
    SoldBy VARCHAR(50),
    StoreID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)

);
CREATE TABLE Product (
    ProductID INT PRIMARY KEY,
    UPC VARCHAR(50),
    ProdName VARCHAR(100),
    mfgr VARCHAR(100),
```

```
    Model VARCHAR(100),
    UnitListPrice DECIMAL(10, 2),
    UnitsInStock INT
);
CREATE TABLE Store (
    StoreID INT PRIMARY KEY,
    ProductID INT,
    StoreName VARCHAR(50),
    StoreLocation VARCHAR(50),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);

CREATE TABLE OrderLine (
    OrderLineID INT PRIMARY KEY,
    OrderID INT,
    Quantity INT,
    ProductID INT,
    UnitSalePrice DECIMAL(10, 2),
    SubTotal DECIMAL(10, 2),
    FOREIGN KEY (OrderID) REFERENCES [Order](OrderID),
    FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
```

COMMENTS:

Foreign Key (FK) Tables:

Order Table:

CustomerID (FK) references Customer(CustomerID)

StoreID (FK) references Store(StoreID)

Store Table

ProductID (FK) references Product(ProductID)

OrderLine Table

OrderID (FK) references Order

ProductID (FK) references Product(ProductID)

These groupings help clarify which tables primarily define their data based on their own keys (PK) and which tables establish relationships with other tables through foreign keys (FK).

A customer can place multiple orders (Customer table to the Order), throught the FK in Order table.

Each order contains one or more products (Order to OrderLine) thus FK in OrderLine table.

Products can be sold in multiple stores, and stores can sell many products (Store to Product), that's why I have the StoreProduct associative table.

…

```
SET IDENTITY_INSERT Customer On;
SET IDENTITY_INSERT [Order] Off;
SET IDENTITY_INSERT OrderLine On;

INSERT INTO Customer (CustomerID, cFirstName, cLastName, cPhone, cStreet, cZipCode)
VALUES
(1, 'John', 'Doe', 123456789, '123 Main St', '12345'),
(2, 'Jane', 'Smith', 987654321, '456 Oak St', '54321');

INSERT INTO [Order] (OrderID, CustomerID, [Order Date], SoldBy, StoreID)
VALUES
(1, 1, '2024-03-20', 'Salesperson A', 1),
(2, 2, '2024-03-21', 'Salesperson B', 2);

INSERT INTO Product (ProductID, UPC, ProdName, mfgr, Model, UnitListPrice,
UnitsInStock)
VALUES
(1, '123456789012', 'Product A', 'Manufacturer X', 'Model X1', 10.99, 100),
(2, '987654321098', 'Product B', 'Manufacturer Y', 'Model Y1', 19.99, 50);

INSERT INTO Store (StoreID, ProductID, StoreName, StoreLocation)
VALUES
(1, 1, 'Store A', '123 Elm St'),
(2, 2, 'Store B', '456 Pine St');

INSERT INTO OrderLine (OrderLineID, OrderID, Quantity, ProductID, UnitSalePrice,
SubTotal)
VALUES
(1, 1, 2, 1, 10.99, 21.98),
(2, 1, 1, 2, 19.99, 19.99),
(3, 2, 3, 1, 10.99, 32.97);

INSERT INTO Customer (CustomerID, cFirstName, cLastName, cPhone, cStreet, cZipCode)
VALUES
(3, 'Alice', 'Johnson', 5551234567, '789 Maple St', '67890'),
(4, 'Bob', 'Williams', 5559876543, '910 Oak St', '45678');

INSERT INTO [Order] (OrderID, CustomerID, [Order Date], SoldBy, StoreID)
VALUES
(3, 3, '2024-03-22', 'Salesperson C', 1),
(4, 4, '2024-03-23', 'Salesperson D', 2);

INSERT INTO OrderLine (OrderLineID, OrderID, Quantity, ProductID, UnitSalePrice,
SubTotal)
VALUES
(4, 3, 2, 1, 10.99, 21.98),
(5, 3, 1, 2, 19.99, 19.99),
(6, 4, 3, 1, 10.99, 32.97); --

I used Alter table  to insert ID.
```

Retrieve all customers along with their orders:

```
SELECT c.CustomerID, c.cFirstName, c.cLastName, o.OrderID, o.[Order Date], o.SoldBy,
o.StoreID
FROM Customer c
LEFT JOIN [Order] o ON c.CustomerID = o.CustomerID;
```

Results | Messages

	CustomerID	cFirstName	cLastName	OrderID	Order Date	SoldBy	StoreID
1	1	John	Doe	1	2024-03-20 00:00:00.000	Salesperson A	1
2	2	Jane	Smith	2	2024-03-21 00:00:00.000	Salesperson B	2
3	3	Alice	Johnson	3	2024-03-22 00:00:00.000	Salesperson C	1
4	4	Bob	Williams	4	2024-03-23 00:00:00.000	Salesperson D	2

```
-- Retrieve all orders made by a specific customer:
SELECT *
FROM [Order]
WHERE CustomerID = 1;
```

Results | Messages

	OrderID	CustomerID	Order Date	SoldBy	StoreID
1	1	1	2024-03-20 00:00:00.000	Salesperson A	1
2	3	1	2024-03-22 00:00:00.000	Salesperson C	1
3	4	1	2024-03-24 00:00:00.000	Salesperson D	2

```
-- Check if the same product (type) can be offered by multiple stores

SELECT p.ProductID, COUNT(s.StoreID) AS StoreCount
FROM Product p
INNER JOIN Store s ON p.ProductID = s.ProductID
GROUP BY p.ProductID;
```

Results | Messages

	ProductID	StoreCount
1	1	1
2	2	1

```
-- Check if each store can individually propose the price and quantity of the offered
product
SELECT s.StoreID, s.ProductID, p.UnitListPrice, p.UnitsInStock
FROM Store s
INNER JOIN Product p ON s.ProductID = p.ProductID;
```

Results | Messages

	StoreID	ProductID	UnitListPrice	UnitsInStock
1	1	1	10.99	100
2	2	2	19.99	50

Each order is identified by an OrderID, and each order has details such as CustomerID, StoreID, OrderDate, and Total (total

COMMENTS:

I added same of person with same CustomerID and added multiple order examples to it too see it in the test part. After inserted the values I checked with the following query examples based on provided set of rules and domain cons.More over to alter table set set the On

```
SET IDENTITY_INSERT Customer On;
SET IDENTITY_INSERT [Order] Off;
SET IDENTITY_INSERT OrderLine On;
```

To insert the duplicated value.

TASK 2 - DETAILS:

Focus on using the following set of tables (not limited to): Production.Product, Production.ProductSubCategory, Production.ProductCategory, Sales.SalesOrderHeader, Sales.SalesOrderDetail, Sales.Customer, Person.Person, Sales.Store, Sales.SalesTerritory, Sales.SalesPerson. Look at the data dictionary and data stored within the AdventureWorks database. ***Store each result as a separate sheet (please use meaningful names)****.*

1. Prepare a pivot table that displays the number of orders per different order statuses and order types (online/instore).
2. Prepare a pivot table that displays the number of products per color and subcategory – limit to subcategories containing word "Bike" in their name.
3. Prepare a pivot table that displays the percentage of the number of orders (number of transactions) made by customers per different territories (use territory name).
4. Prepare a pivot table that displays the sum of orders (value-wise) and total volume (quantity-wise) per product category (category name is required) with a simple PivotChart (use bar chart).
    a. (*) Add a slicer with order date and filter the data to year 2013.
5. (*) Create a pivot table that displays the number of customers per different customer types and sales location.

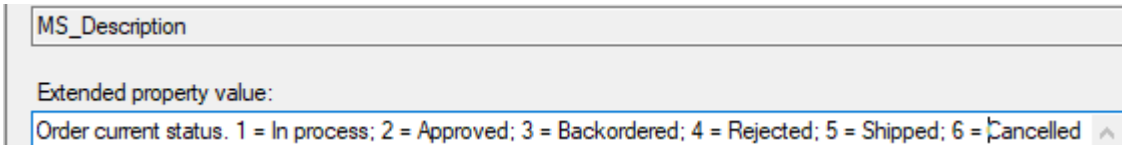
All results should be stored in a single MS Excel file (with all pivots in each sheet) and a text file (PDF – with all short description and answers). After completing the tasks, double check your approach and please present your results to the teacher.

TASK 2 - SOLUTIONS:

Use this section to provide your solutions, please attach a screenshot of the resultant table/chart (if it is large, just an excerpt) and a screenshot of the pivot design (specifying which attributes are usedfor columns/rows/values), try to briefly comment your approach:

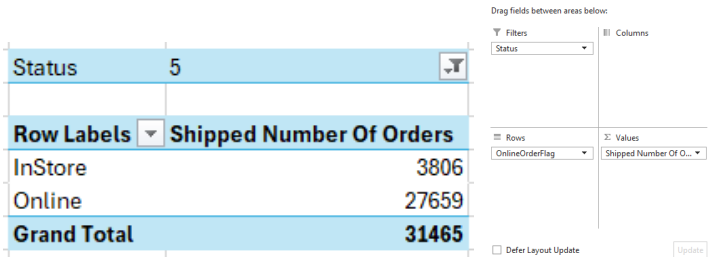
1

Status table, current status values description in Column properties:



Status filter option is defines the based on the selected current status.

However, all the provided Status of Order are seems as 5=shipped in the data of the Status column. Thus, provided values are presents the shipped Number of the Orders.



Provided Pivot table at the above image, created through the Sales.SalesorderHeader table.

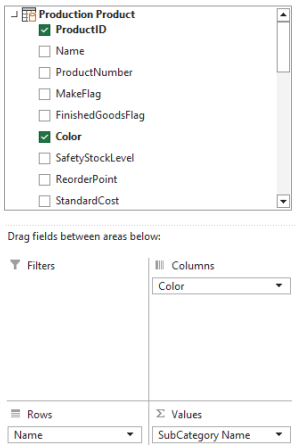
(count)SalesOrderID: To find the number of orders.

OnlineOrderFlag: used to see if shipped product is True\False which means of Online or InStore. I change the value in the pivot table as if the value was true than online and if False than InStore.

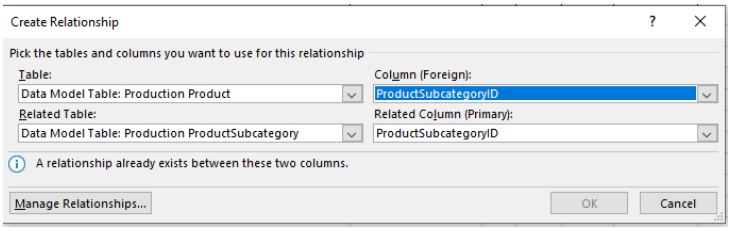
2

Final solution (created pivot table) based on the given requirements and displaying number of products per color and subcategory which are limited to name contains 'Bike', found as below image:

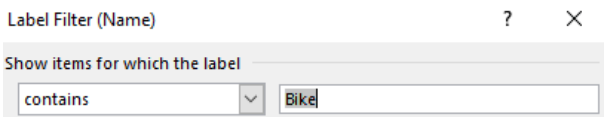
SubCategory Name	Column Labels						
Row Labels	(blank)	Black	Blue	Red	Silver	Yellow	Grand Total
Bike Racks	1						1
Bike Stands	1						1
Mountain Bikes		16			16		32
Road Bikes		14		20		9	43
Touring Bikes			13			9	22
Grand Total	2	30	13	20	16	18	99



There is relationship has done between the Production.Product and Production.SubCategory table by ID as presented in the below image.



I order to filter the names that contains Bike:

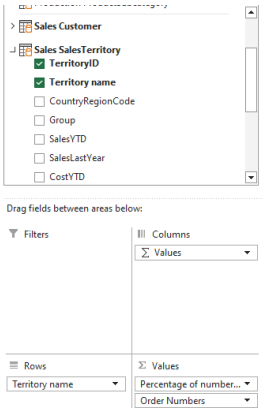


3

…

Final solution (created pivot table) based on the given requirements and displaying the percentage of the number of orders (number of transactions) made by customers per different territories (use territory name) are presented below as image taken from Excel file.

Row Labels	Percentage of number of orders	Order Numbers
Australia	18.49%	9
Canada	9.04%	6
Central	0.67%	3
France	9.51%	7
Germany	9.34%	8
Northeast	0.57%	2
Northwest	17.76%	1
Southeast	0.89%	5
Southwest	23.69%	4
United Kingdom	10.05%	10
Grand Total	100.00%	55



In order to connect two related table to reach out the requested value used : Sales.Customer and Sales.SalesTerritory table with the common TerritoryID.

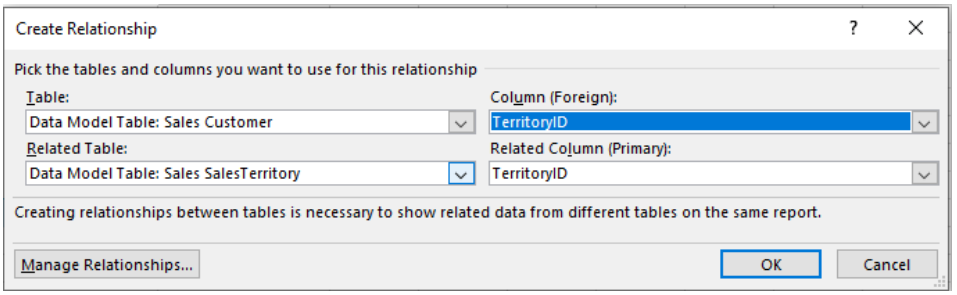
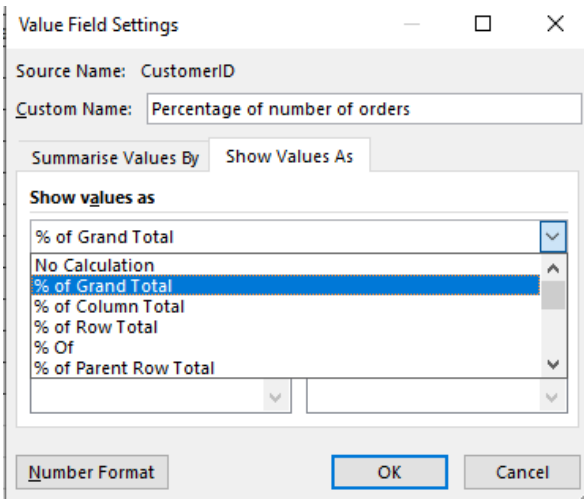


Image below is the representation of the setting value field as Grand total to see the result as percentage value.



4

Results are provided based on: SalesOrderHeader -> TotalDue (Sum of orders)

SalesOrderDetail -> TotalQty (Total Volume)

Production.Category -> Name (Category Name)

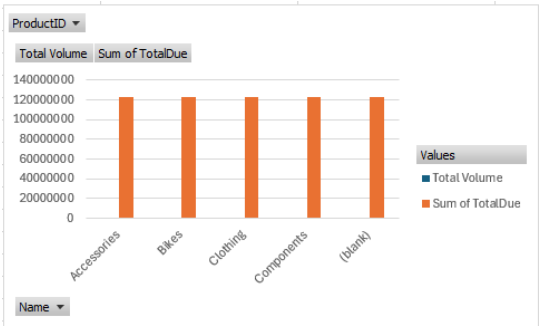
- *2 connections are created .*

Final connections:

Active	Production Product (ProductSubcategoryID)	Production ProductSubcategory (ProductSubc...
Active	Production ProductSubcategory (ProductCate...	Production ProductCategory (ProductCategor...
Active	Sales Customer (TerritoryID)	Sales SalesTerritory (TerritoryID)
Active	Sales SalesOrderDetail (ProductID)	Production Product (ProductID)
Active	Sales SalesOrderDetail (SalesOrderID)	Sales SalesOrderHeader (SalesOrderID)

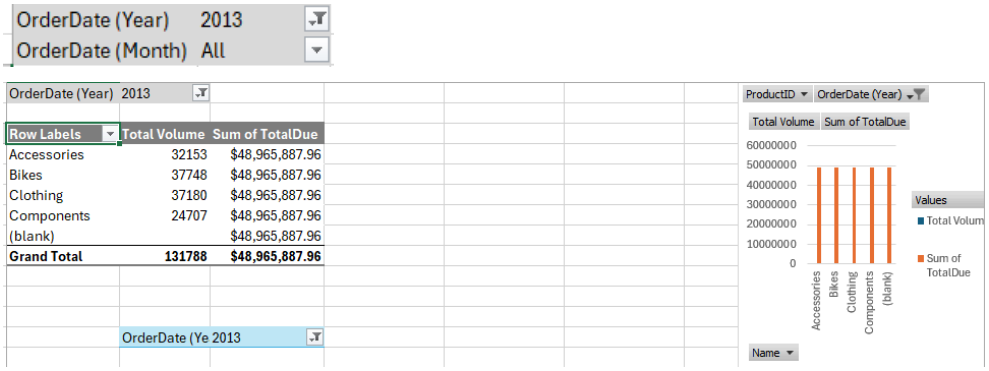
Final result: :/

ProductID	All	
Row Labels	**Total Volume**	**Sum of TotalDue**
Accessories	61932	$123,216,786.12
Bikes	90268	$123,216,786.12
Clothing	73670	$123,216,786.12
Components	49044	$123,216,786.12
(blank)		$123,216,786.12
Grand Total	**274914**	**$123,216,786.12**



4a)

The filter section provided above, filters the data based on the requried year or month. Provided in a basic filter option as requested.I' ve extra add a filter option (Order Date column) from the SalesOrderHeader table.



5(*)

…

Number of customers -> Sales.Customer(CustomerID)

Per different Customer type -> Person.person (person type). On these optionswe need

IN:individual customer and SC: Store Customer…

	EM	SC	GC
	SP	VC	IN

And Sales Location -> Sales.SalesTerritory (Name)

Name
Northwest
Northeast
Central
Southwest
Southeast
Canada
France
Germany
Australia
United Kingdom
NULL

CustomerID
1
2
3
4
5
6
7
8
9
10
11
12
13

Connection:

Active	Sales Customer (TerritoryID)	Sales SalesTerritory (TerritoryID)

Active	Person Person (BusinessEntityID)	Sales SalesPerson (BusinessEntityID)

Final result;

Distinct Count of CustomerID	**Column Labels**		
Row Labels	**IN**	**SC**	**Grand Total**
Australia	3665	3665	3665
Canada	1791	1791	1791
Central	132	132	132
France	1884	1884	1884
Germany	1852	1852	1852
Northeast	113	113	113
Northwest	3520	3520	3520
Southeast	176	176	176
Southwest	4696	4696	4696
United Kingdom	1991	1991	1991
Grand Total	**19820**	**19820**	**19820**

At the table above belong to representation of displays the number of customers per different customer types and sales location. As we can see in the result territory names are sorted based on the customer tpes which are IN and SC.

TASK 2 - CONCLUSIONS:

Since working with task number 4 and 5 was challenging to display the pivot table based on the requirements, I had difficulties to be able to find the correct result to display in the pivot table. However, I gained the data analysis skills in general and learned a lot for creating of pivot tables. Such as rows, filtering, values part as like visualization tool. Because I practiced how those parts works and how we can define the values as our wish such as average, min, max filter option and more.

Moreover, I believe the most challenging part was setting te connection between the tables and decide to which table to work with. Because for example in task 4 we had more than 3 tables which are the ID's are not the same so that I needed to find the ID with other tables that are matching so that I can set the connections between the tables.

TASK 3

TASK 3 - DETAILS:

1. Prepare a single query which contains basic information about product's sales:
    a. Note that you need to include product id, product name, product category name and product subcategory name, product color, product weight;
        i. Consider including other attributes (use results from point 2).
    b. Note that you need to include sales amount and order quantity;
        i. Consider including other performance measure (use results from point 3)
    c. Please use the following basic sales info about time: use order's date;
        i. Please include separate attributes for day, month and year – use DATEPART (T-SQL function).
    d. Please include unique identifier of each order.
2. Connect to prepared data from MS Excel. In a new sheet (each pivot table or pivot chart as a new sheet) please prepare:
    a. Create a pivot chart that displays the total number of sold items by weight.
    b. Create a pivot chart that displays the average weight of products by subcategories.
    c. Create a pivot table to answer the question: is the profit from products with color "red" higher from other products?

---

1 SQL

…

1A)

Identifying tables: Production.Product, Production.ProductSubCategory, Production.ProductCategory table.

I did not used LEFT JOIN or RIGHT JOIN here because:
LEFT JOIN would include products without subcategories/categories, leading to incomplete data.
RIGHT JOIN wouldn't be relevant as we're primarily interested in product information.
In case our aim was a retrieve the all products regardless category or subcategory information, we could use LEFT JOIN on both PS and PC tables.

Created query to find requested columns is as below: Created model represents the solid query without addition of Volume or Sales amount from SOD table! :

```
SELECT ProductID,
       P.Name AS ProductName,
       PC.Name AS [Product Category Name],
       Ps.Name AS [Product SubCategory Name],
       color,
       Weight,
       Size
FROM
       Production.product as P
JOIN
       Production.ProductSubcategory AS PS ON P.ProductSubcategoryID =
PS.ProductSubcategoryID
JOIN
       Production.ProductCategory AS PC ON Ps.ProductCategoryID =
PC.ProductCategoryID;
```

Results | Messages

	ProductID	ProductName	Product Category Name	Product SubCategory Name	color	Weight	Size
1	680	HL Road Frame - Black, 58	Components	Road Frames	Black	2.24	58
2	706	HL Road Frame - Red, 58	Components	Road Frames	Red	2.24	58
3	707	Sport-100 Helmet, Red	Accessories	Helmets	Red	NULL	NULL
4	708	Sport-100 Helmet, Black	Accessories	Helmets	Black	NULL	NULL
5	709	Mountain Bike Socks, M	Clothing	Socks	White	NULL	M

1b)

Identifying tables: Production.Product, Production.ProductSubCategory, Production.ProductCategory and Sales.SalesOrderDetail (to get Volume and SalesAmount by summing the OrderQty and LineTotal) table.

In order to have a good analyzing and understanding of the data I used the data where color and weight is not null.
Query and a result below is final of my result including with OrderQty and LineTotal from the SOD table.

GROUP BY statement used because its essential for aggregates sales that used as a SUM function to present the OrderQTY and LineTotal. I group by all the product details to get total for each unique combination.

```
CREATE VIEW Analyze AS
 SELECT
 P.ProductID,
 P.Name AS ProductName,
 PC.Name AS [Product Category Name],
 PS.Name AS [Product SubCategory Name],
 P.Color,
 P.Weight,
 SOH.Status,
 SOH.SalesOrderID AS [SalesOrderID],
       YEAR(SOH.OrderDate) AS [Year],
       MONTH(SOH.OrderDate) AS [Month],
       DAY(SOH.OrderDate) AS "Day",
       SUM(SOD.OrderQty) AS [Volume],
       SUM(SOD.LineTotal) AS [Sales Amount],
       COUNT(DISTINCT SOD.SalesOrderID) AS [Number of Orders]
FROM
 Production.Product AS P
INNER JOIN
 Production.ProductSubcategory AS PS ON P.ProductSubcategoryID =
PS.ProductSubcategoryID
INNER JOIN
 Production.ProductCategory AS PC ON PS.ProductCategoryID = PC.ProductCategoryID
INNER JOIN
 Sales.SalesOrderDetail AS SOD ON P.ProductID = SOD.ProductID
INNER JOIN
       Sales.SalesOrderHeader AS SOH ON SOH.SalesOrderID = SOD.SalesOrderID
WHERE
 P.Color IS NOT NULL AND
 P.Weight IS NOT NULL
GROUP BY
 P.ProductID,
 P.Name,
 PC.Name,
 PS.Name,
 P.Color,
 P.Weight,
 SOH.Status,
```

```
SOH.SalesOrderID,
YEAR(SOH.OrderDate),
MONTH(SOH.OrderDate),
DAY(SOH.OrderDate);
--ORDER BY [Year], [Month], [Day] ASC;
```

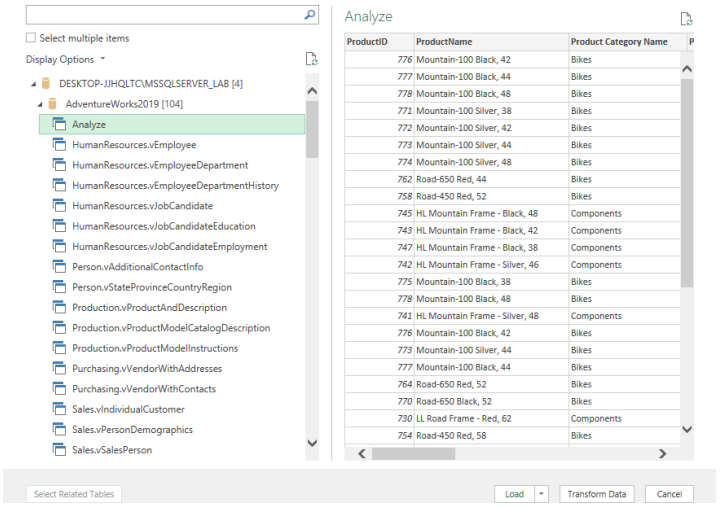
Results | Messages

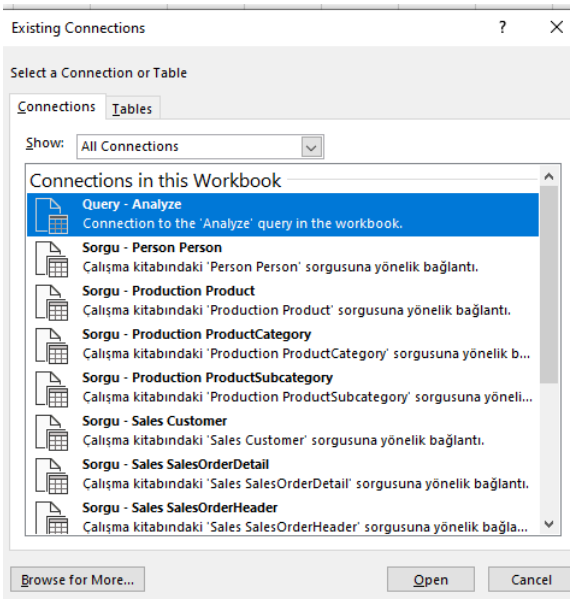
	ProductID	ProductName	Product Category Name	Product SubCategory Name	Color	Weight	Status	SalesOrderID	Year	Month	Day	Volume	Sales Amount	Number of Orders
1	725	LL Road Frame - Red, 44	Components	Road Frames	Red	2.32	5	43668	2011	5	31	2	367.876400	1
2	725	LL Road Frame - Red, 44	Components	Road Frames	Red	2.32	5	43685	2011	5	31	1	183.938200	1
3	722	LL Road Frame - Black, 58	Components	Road Frames	Black	2.46	5	43689	2011	5	31	1	178.580800	1

And than I created view AS Analyze on SSMS query to call it from excel file by deleting the Order by statement due to cause of error!

```
CREATE VIEW Analyze AS
  SELECT
  P.ProductID,
  P.Name AS ProductName,
```

<u>To connect the created view statement on SSMS to the Excel file :</u>
I got the data from SQL server database and then Selected transform data…
There is below the steps doing it:

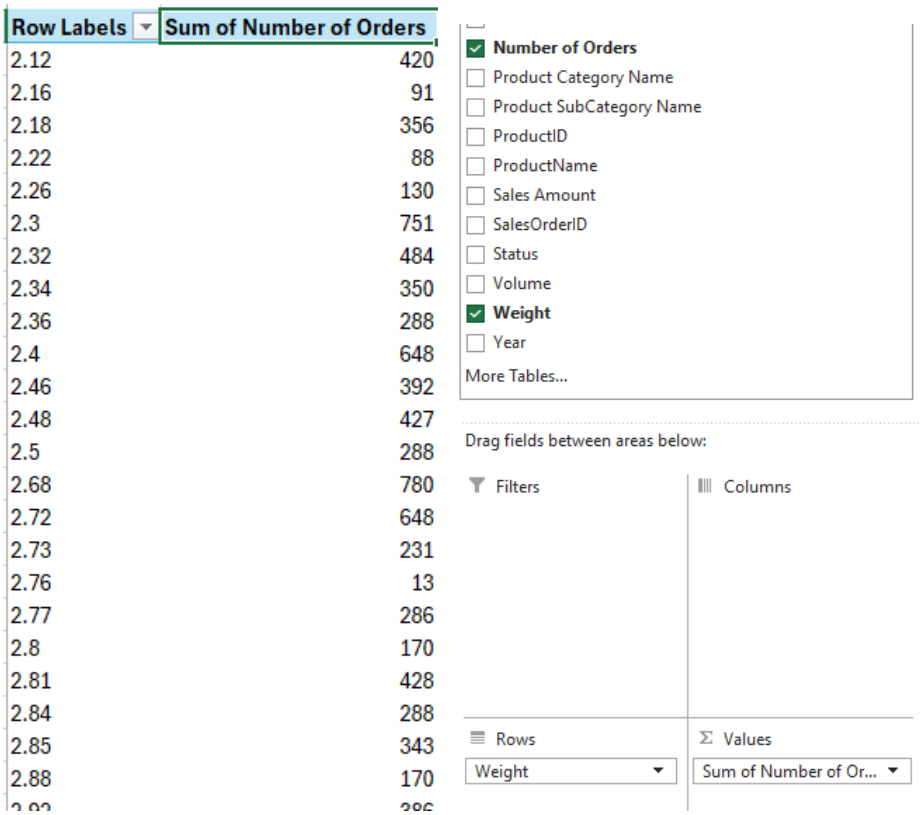




2 Pivot 5a

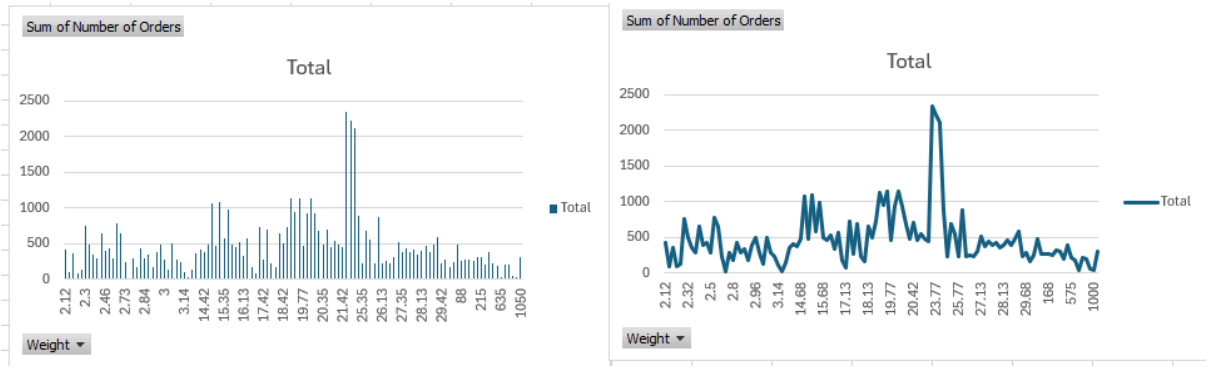
After load the data into Excel file, I've Created an pivot table as required in task : Create a pivot chart that displays the total number of sold items by weight.

At this task we are aimed to display the total number of sold items by weight. So to do that I decided to firstly get the Number of Orders from the loaded data and calculate the sum of these orders. And then selected the weight attribute as rows to get the result. ->



Row Labels	Sum of Number of Orders
2.12	420
2.16	91
2.18	356
2.22	88
2.26	130
2.3	751
2.32	484
2.34	350
2.36	288
2.4	648
2.46	392
2.48	427
2.5	288
2.68	780
2.72	648
2.73	231
2.76	13
2.77	286
2.8	170
2.81	428
2.84	288
2.85	343
2.88	170

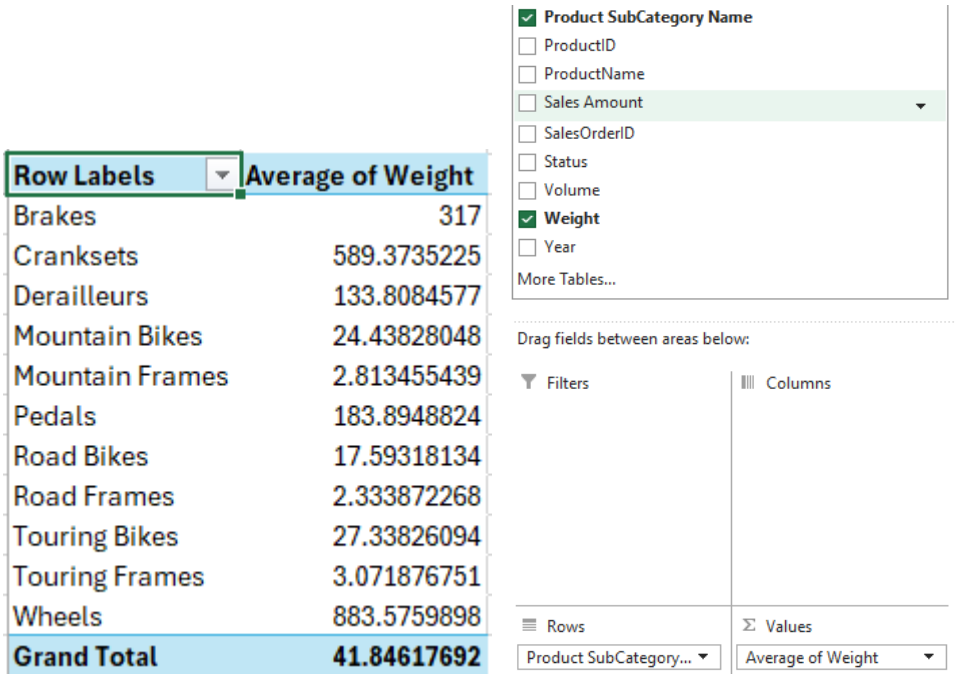
Created Pivot Charts are below as presented:



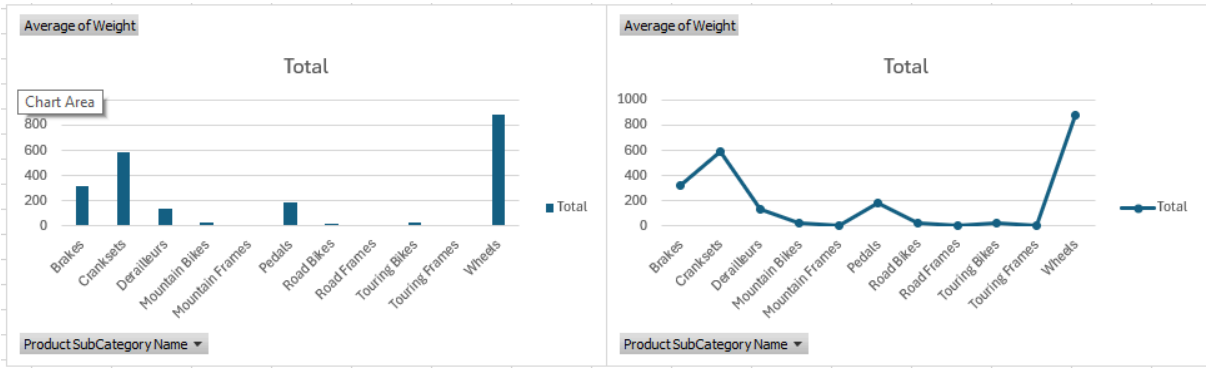
As we can see and analyze from the created pivot chart, around 21-42 to 25 (weight) is the generally the highest total. Moreover, we can check the chart and get the necessary information based on our business requirements for further analysis.

3 Pivot 5B

At this part we are aimed to display the average weight of products by subcategories and to do that, I used Weight column by getting its average weight in the values section as provided and Product.SubCategory name to display as required.



Row Labels	Average of Weight
Brakes	317
Cranksets	589.3735225
Derailleurs	133.8084577
Mountain Bikes	24.43828048
Mountain Frames	2.813455439
Pedals	183.8948824
Road Bikes	17.59318134
Road Frames	2.333872268
Touring Bikes	27.33826094
Touring Frames	3.071876751
Wheels	883.5759898
Grand Total	**41.84617692**



As we can see in the provided pivot chart examples, there is provided average of weight based on the product Subcategory names. Wheels is the highest and we are able analyse more based on our business requirements in the further process.

4 Pivot 5C

At this part we are aimed to decide the profit from products with color: 'red' which are higher from other products. In order to demonstrate that, First I decide to find out the Sum of Sales amount (Sales amount attribute) and compare the result by the color ->

ROWS: Color

Values: Sum of Sales Amount

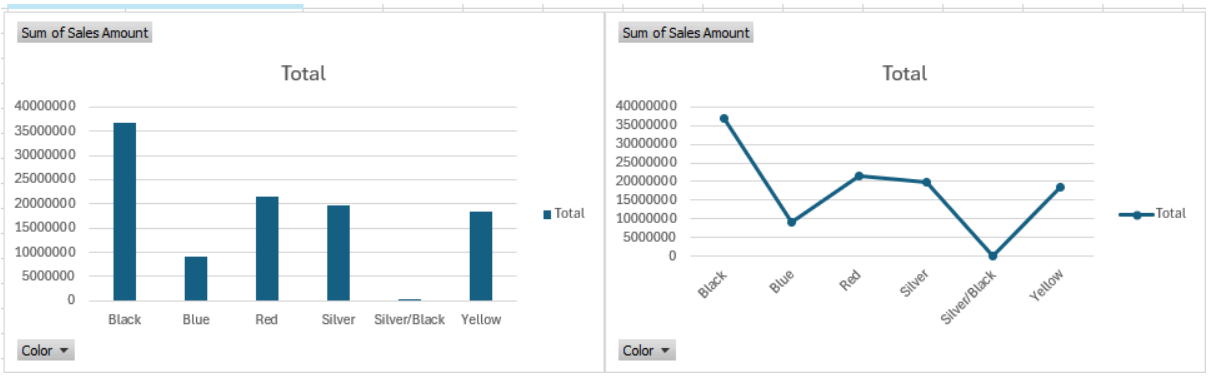
Filters: Color, rows: ProductName,

Values: Sum of Sales Amount

Row Labels	Sum of Sales Amount
Black	36826164.95
Blue	9177955.973
Red	21465497.15
Silver	19662135.82
Silver/Black	140340.5918
Yellow	18348306.93
Grand Total	**105620401.4**

Color	Red
Row Labels	**Sum of Sales Amount**
HL Road Frame - Red, 44	395182.6993
HL Road Frame - Red, 48	89872.1736
HL Road Frame - Red, 62	394255.5724
LL Road Frame - Red, 44	194692.5991
LL Road Frame - Red, 48	132125.2522
LL Road Frame - Red, 52	20104.4434
LL Road Frame - Red, 60	195933.4094
LL Road Frame - Red, 62	137213.4851
ML Road Frame - Red, 48	89224.5
ML Road Frame - Red, 52	32120.82
Road-150 Red, 44	1340419.942
Road-150 Red, 48	1540803.062
Road-150 Red, 52	1415563.612
Road-150 Red, 56	1847818.628
Road-150 Red, 62	1769096.688
Road-250 Red, 44	1448122.479

Second picture is another pivot table to practice and see the clear result of each name based on filtered color.



Above Pivot charts are created based on the first image in goal of to compare the results as in the graph.

By the results displayed as in the created pivot charts, red is not the highest compared the other products. The highest score is belong to Black…

TASK 3 – Conclusions:

This data analysis exercise demonstrates us to power of combining SQL and Excel for transforming raw sales data into actionable insights. I learned to extract, aggregate, and visualize key metrics, product development decisions based on the discovered trends. Through this project I strengthened the SQL skills including with filtering data, joining tables, and calculating sales metrics so on. Also, I got more experienced about the creation of pivot tables and pivot chart in Excel, visually representing product sales performance across weights, categories, and even colors.

After the completion of the Task 3 I realized that the most important part was the creation of query I did on SSMS to analyse the data in future. Because creating view statement based on created query associated by the multiple tables, allows us to detailed analyse option in Excel extensions. By structuring the data appropriately, I have now better understanding of importance of data preparation and analysis techniques in deriving insights.