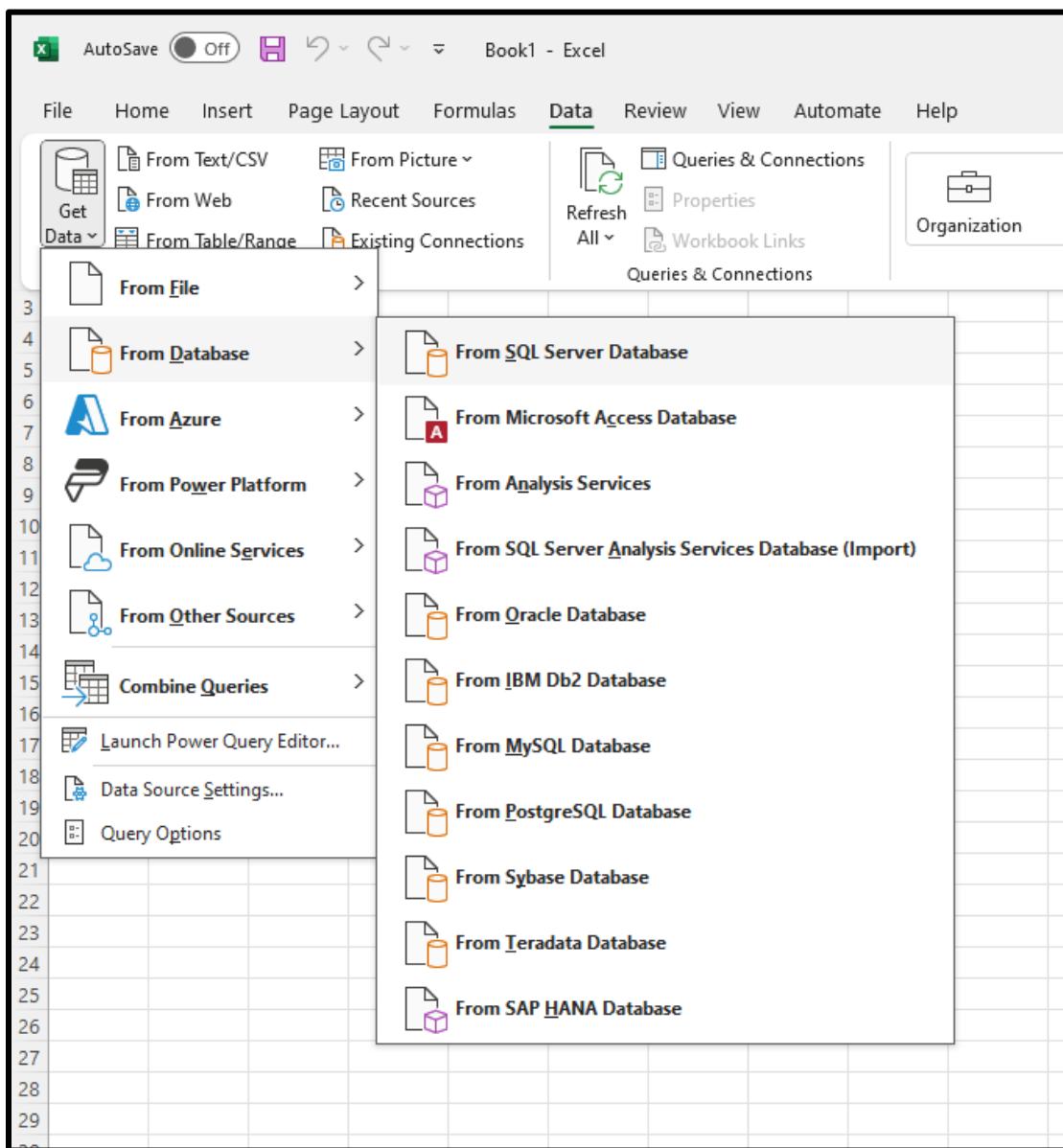


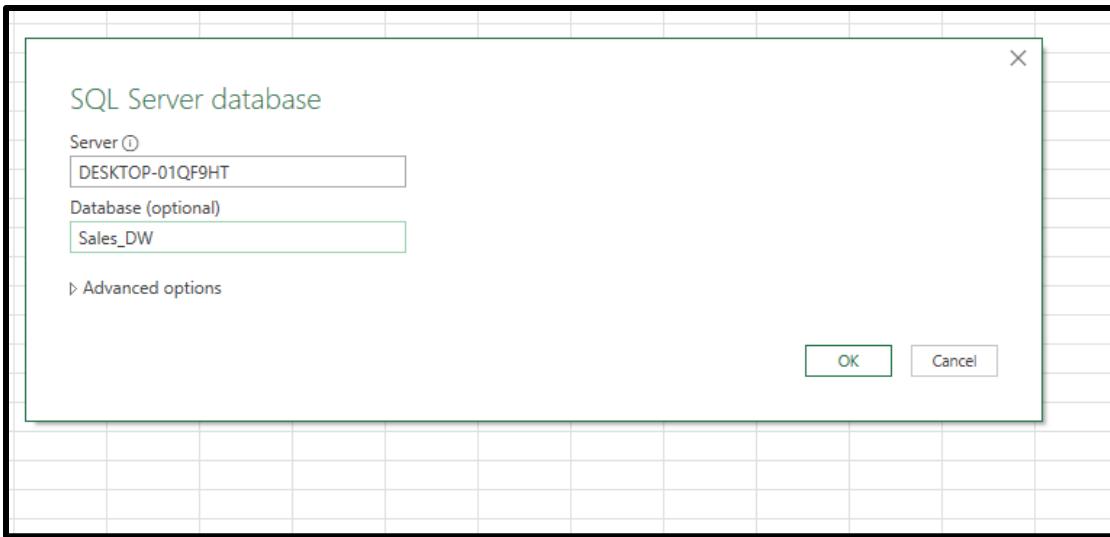
Practical 1

Perform the analysis for the following:

- a. Import the data warehouse data in Microsoft Excel and create the Pivot table and Pivot Chart.
- b. Import the cube in Microsoft Excel and create the Pivot table and Pivot Chart to perform data analysis.

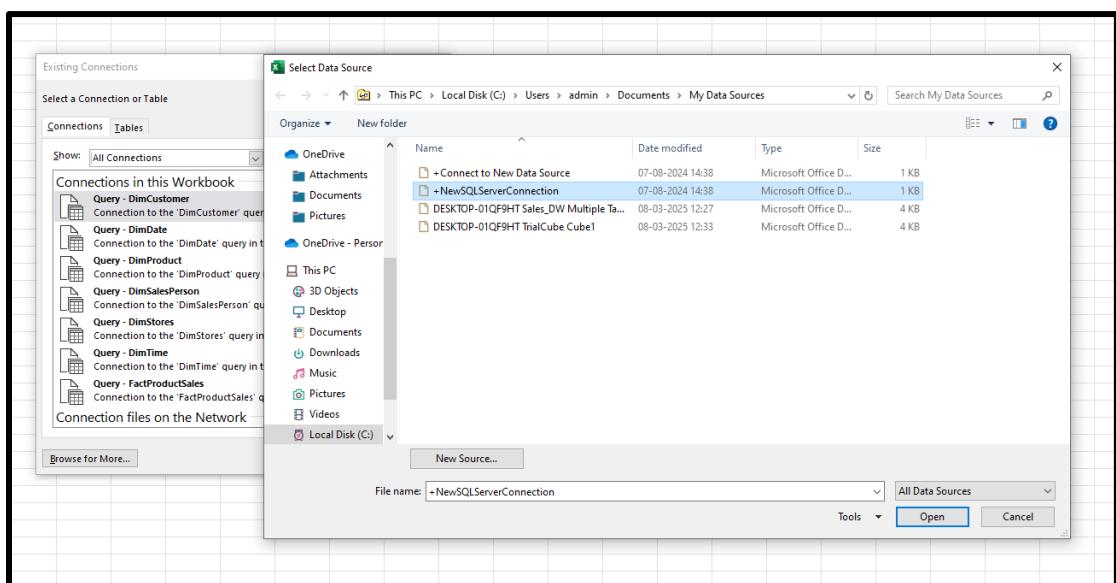
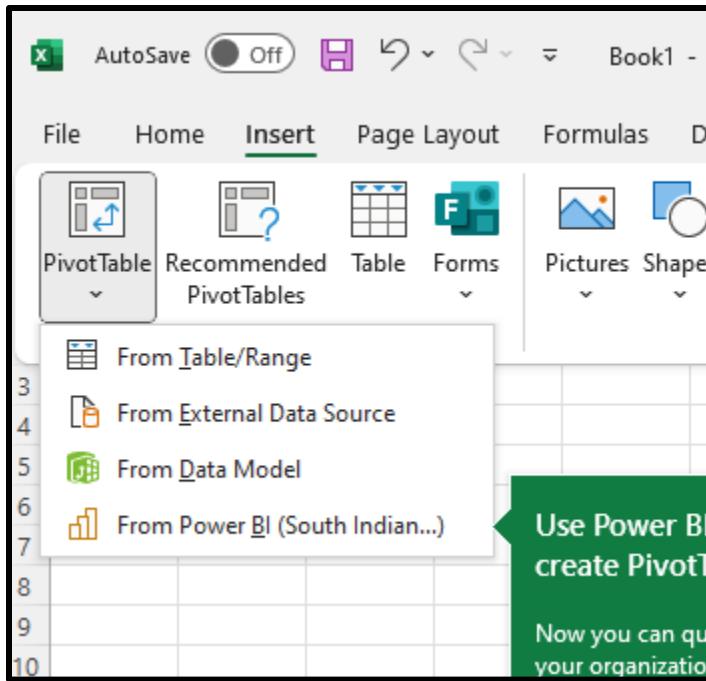
a.

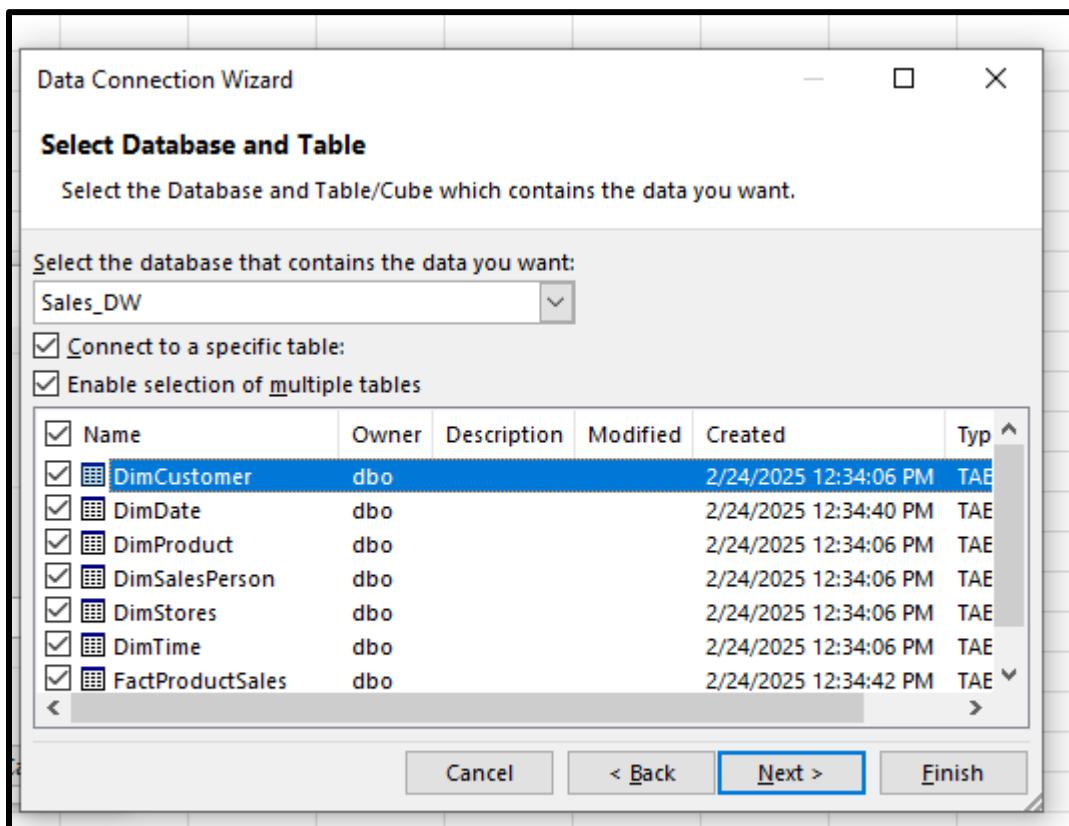
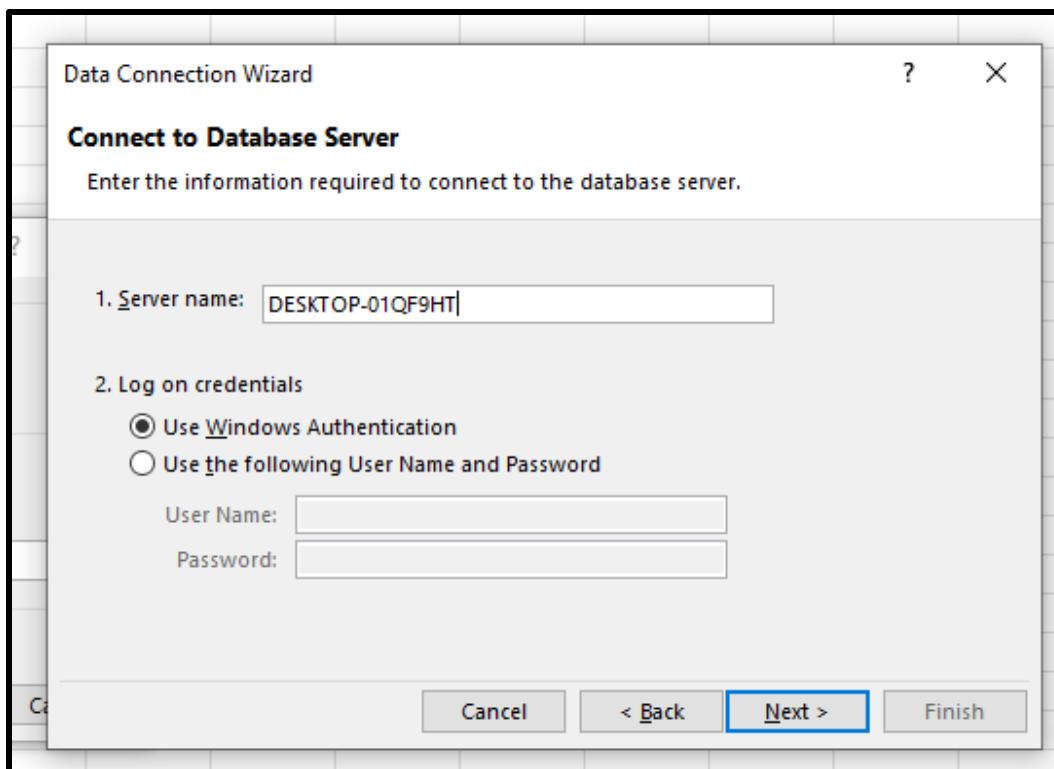


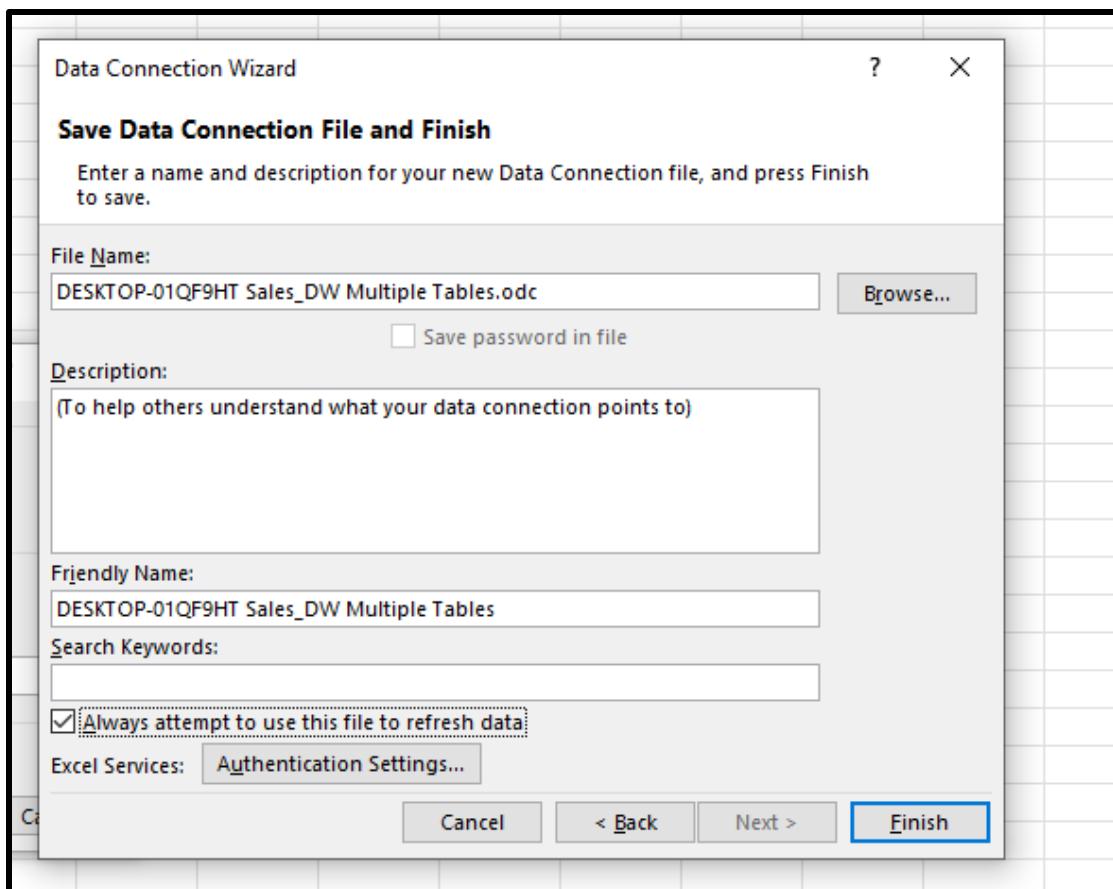


The Navigator window shows the connection "DESKTOP-01QF9HT: Sales_DW [7]" expanded, displaying tables: DimCustomer, DimDate, DimProduct, DimSalesPerson, DimStores, DimTime, and FactProductSales. The FactProductSales table is selected and highlighted with a green border. The table preview shows columns: TransactionId, SalesInvoiceNumber, SalesDateKey, SalesTimeKey, and SalesTime. The data consists of 23 rows of sales transactions from January 2013.

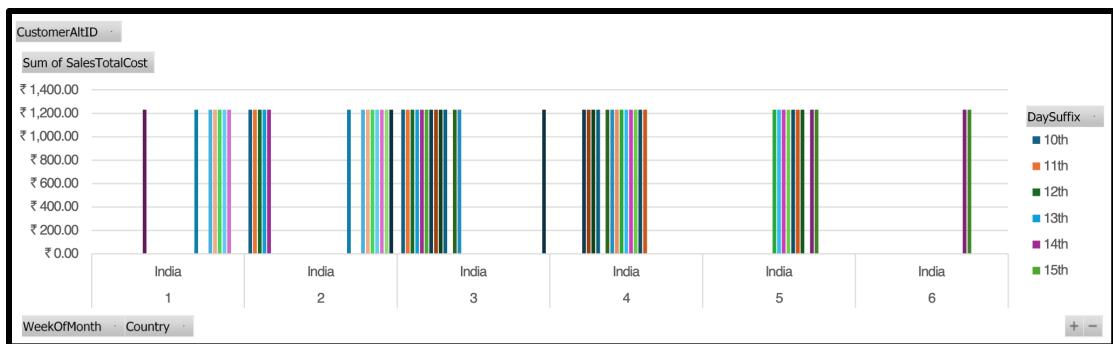
TransactionId	SalesInvoiceNumber	SalesDateKey	SalesTimeKey	SalesTime
1	1	20130101	44347	
2	1	20130101	44347	
3	1	20130101	44347	
4	2	20130101	44519	
5	2	20130101	44519	
6	3	20130101	52415	
7	3	20130101	52415	
8	3	20130101	52415	
9	3	20130101	52415	
10	4	20130102	44347	
11	4	20130102	44347	
12	5	20130102	44519	
13	5	20130102	44519	
14	6	20130102	52415	
15	6	20130102	52415	
16	7	20130102	44347	
17	7	20130102	44347	
18	8	20130103	59326	
19	8	20130103	59326	
20	9	20130103	59349	
21	9	20130103	59349	
22	10	20130103	67390	
23	10	20130103	67390	



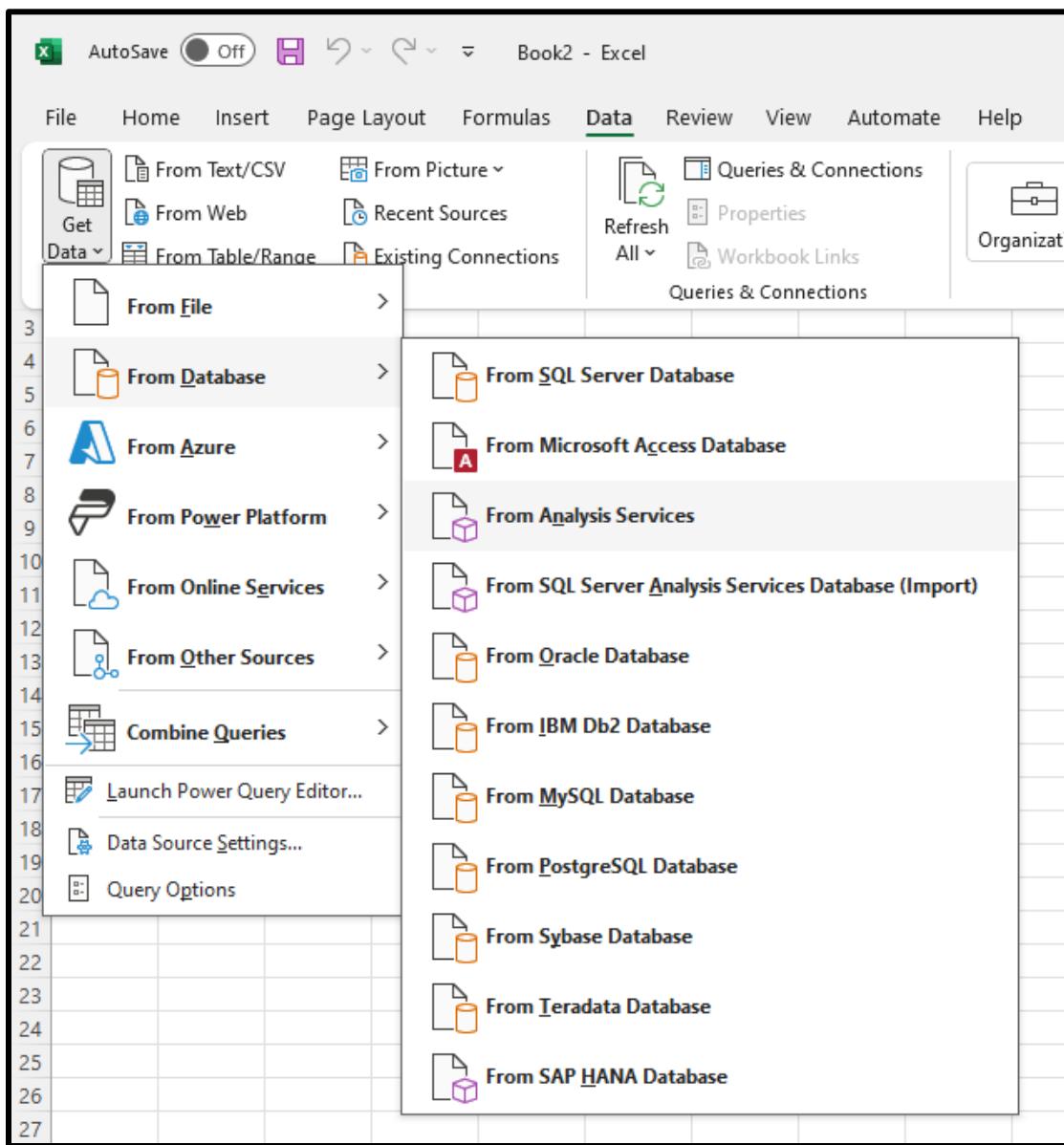


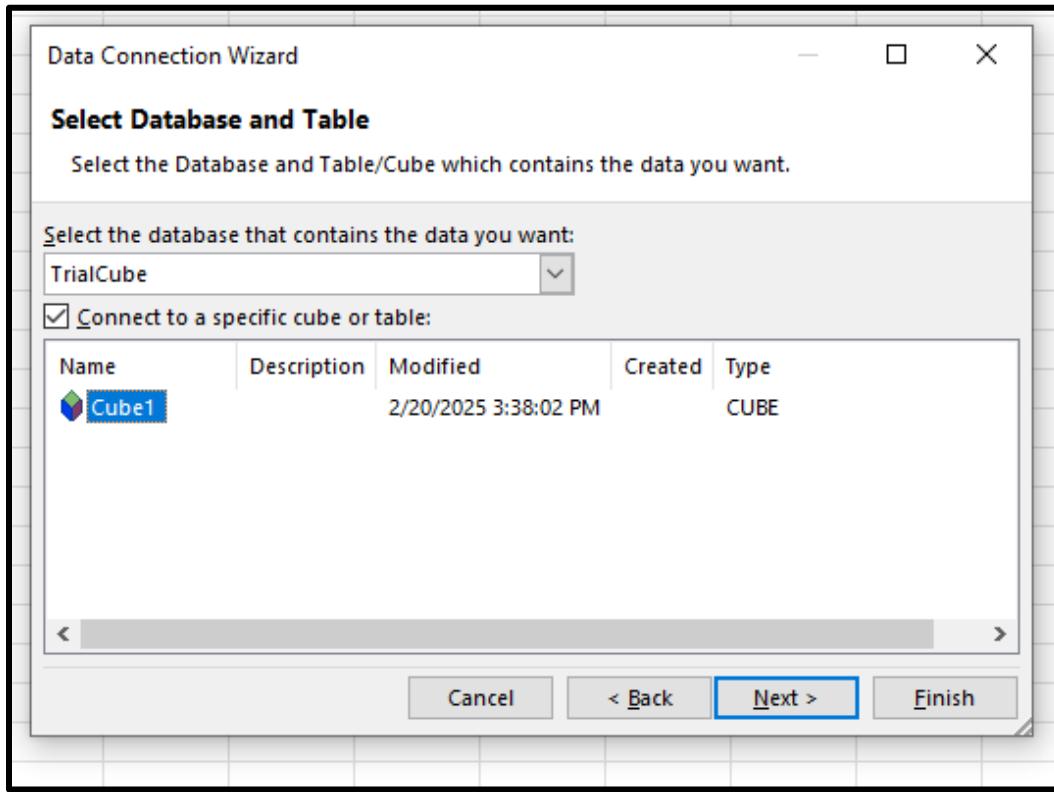
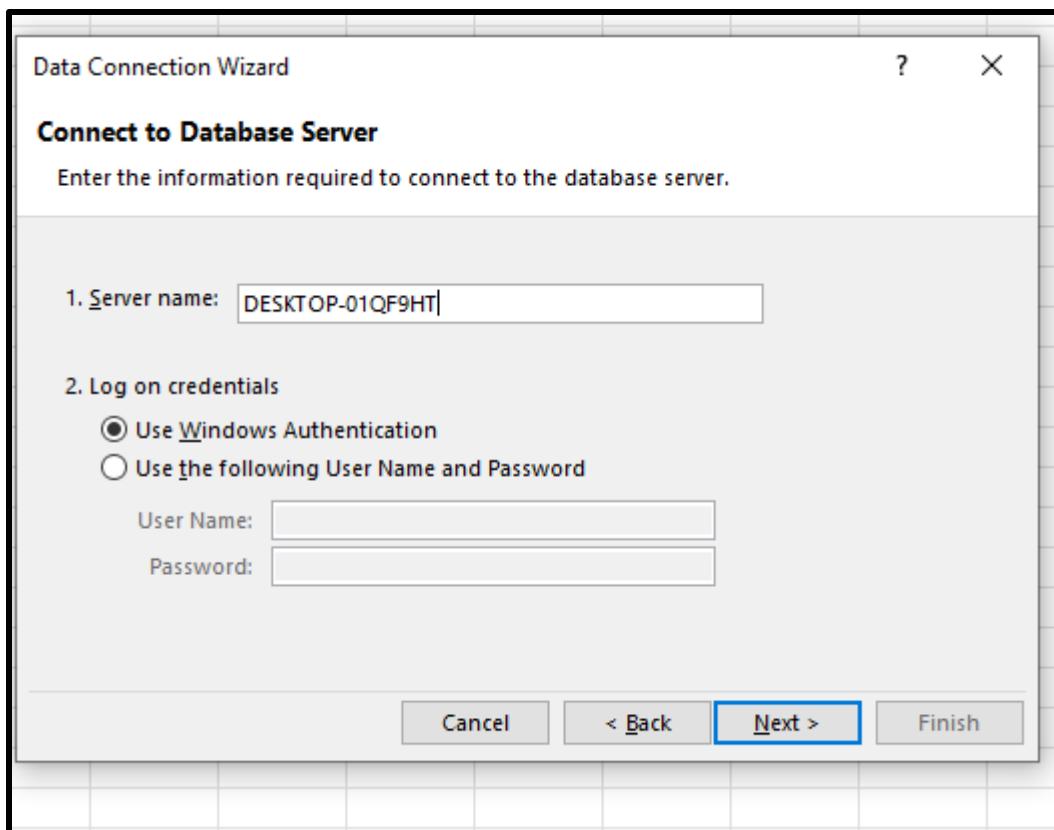


Output:

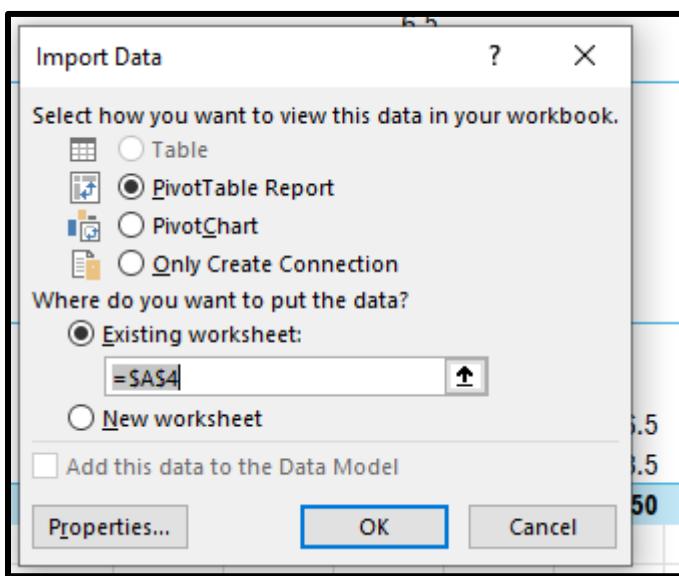
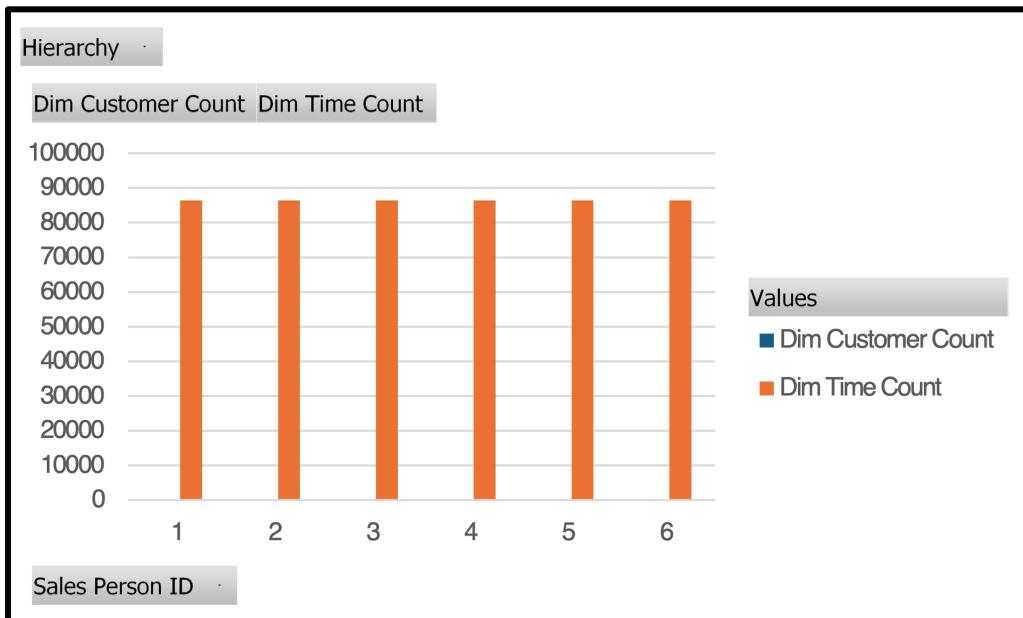


b.





Output:



Output:

A	B	C	D	E	F	G	H	I
1 Store ID	All							
2 Hierarchy	All							
3								
4 Sales Total Cost	Column Labels							
5	44347	44519	52415	59326	59349	67390	74877	Grand Total
6 Row Labels	1	2	3	1	2	3	1	
7 1								
8 Nirma Soap		120		60			180	
9 Rice Grains 1kg	48			24			72	
10 SunFlower Oil 1 ltr	43.5	87		87			217.5	
11 Wheat Floor 1kg	26			6.5			32.5	
12 2								
13 Arial Washing Powder 1kg		278					278	
14 Nirma Soap		60		60			120	
15 Rice Grains 1kg		26					26	
16 SunFlower Oil 1 ltr		43.5					43.5	
17 Wheat Floor 1kg				13			13	
18 3								
19 Arial Washing Powder 1kg	139						139	
20 Nirma Soap	60						60	
21 Rice Grains 1kg				6.5			6.5	
22 SunFlower Oil 1 ltr				43.5			43.5	
23 Grand Total	316.5	207	407.5	147	30.5	73	50	1231.5
24								
25								

Practical 2

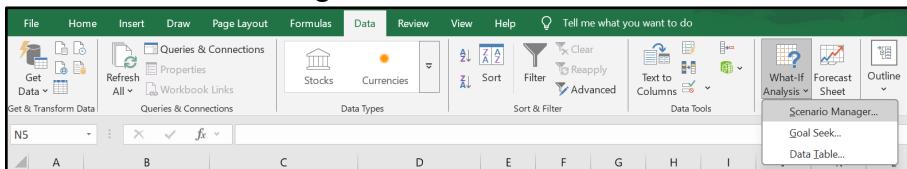
Apply the what – if Analysis for data visualization. Design and generate necessary reports based on the data warehouse data. Use Excel.

A	B	C	D	E
Book Store				
3	Total number of books	% sold for the highest price		
4	100	60%		
5				
6	Number of books	Unit profit		
7	Highest price	\$50		
8	Lower price	\$20		
9				
10	Total profit	\$3,800		
11				

1. On the Data tab, in the Forecast group, click What-If Analysis.

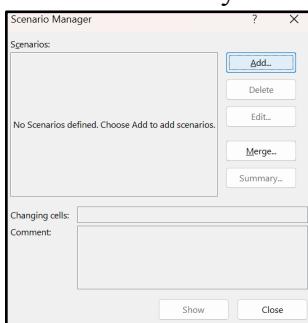


2. Click on Scenario Manager

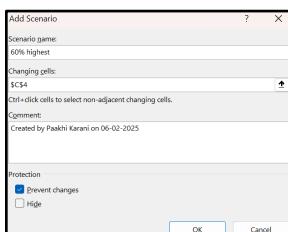


The Scenario Manager dialog box appears.

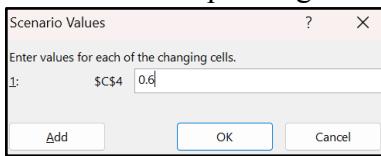
3. Add a scenario by clicking on Add.



4. Type a name (60% highest), select cell C4 (% sold for the highest price) for the Changing cells and click on OK.

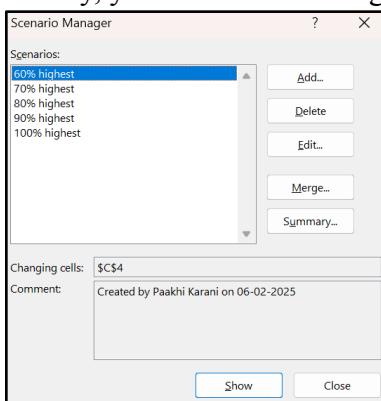


5. Enter the corresponding value 0.6 and click on OK again.

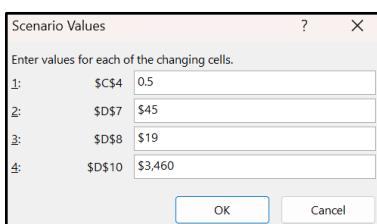
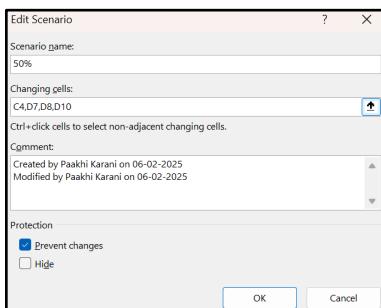


6. Next, add 4 other scenarios (70%, 80%, 90% and 100%).

Finally, your Scenario Manager should be consistent with the picture below:



OR



C4 : 50% fx

Book Store

	Total number of books	% sold for the highest price	
1	100	50%	
	Number of books	Unit profit	
2	Highest price	60	\$45
3	Lower price	40	\$19
4	Total profit	\$3,460	

Scenario Manager

Scenarios:

- 60% highest
- 70% highest
- 80% highest
- 90% highest
- 100% highest
- 50% (selected)

Changing cells: \$C\$4:\$D\$7:\$D\$8:\$D\$10

Comment: Created by Paakhi Karani on 06-02-2025
Modified by Paakhi Karani on 06-02-2025

Show Close

Practical 3

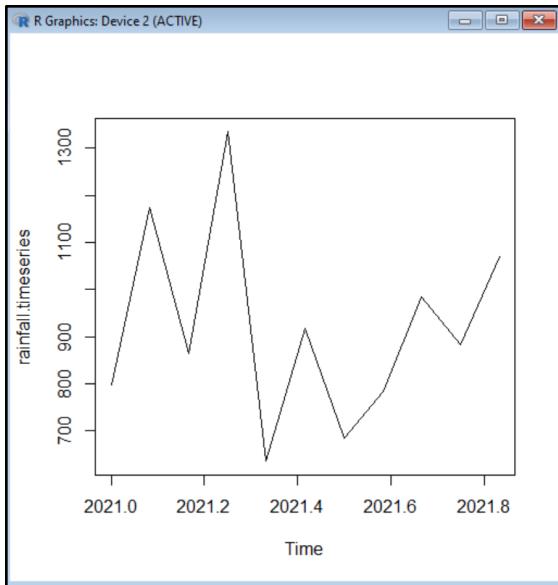
Perform the data classification using classification algorithm using R/Python.

Code:

```
> rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,784.2,985,882.8,1071)
> rainfall.timeseries <- ts(rainfall,start = c(2021,1),frequency = 12)
> print(rainfall.timeseries)
    Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov
2021 799.0 1174.8 865.1 1334.6 635.4 918.5 685.5 784.2 985.0 882.8 1071.0
> png(file = "rainfall.png")
> plot(rainfall.timeseries)
> dev.off()
null device
1
> plot(rainfall.timeseries)
```

Output:

```
> rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,784.2,985,882.8,1071)
> rainfall.timeseries <- ts(rainfall,start = c(2021,1),frequency = 12)
> print(rainfall.timeseries)
    Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov
2021 799.0 1174.8 865.1 1334.6 635.4 918.5 685.5 784.2 985.0 882.8 1071.0
>
> png(file = "rainfall.png")
> plot(rainfall.timeseries)
> dev.off()
null device
1
> plot(rainfall.timeseries)
> |
```



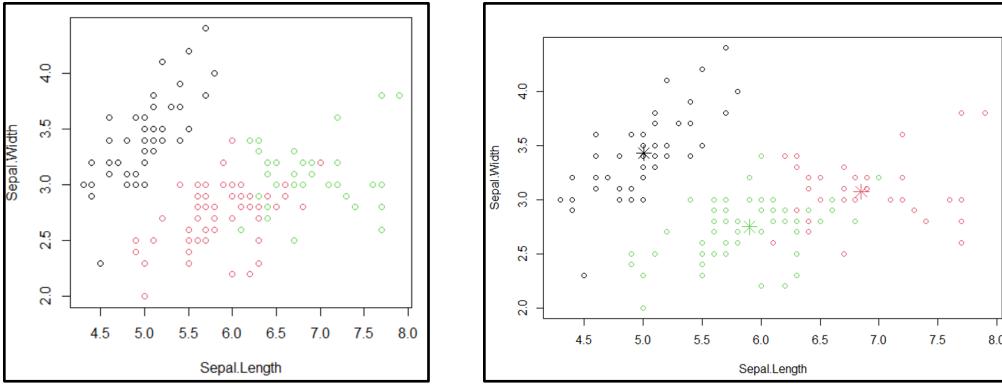
Practical 4

Perform the data clustering using clustering algorithm using R/Python.

Code:

```
> newiris <- iris  
> newiris$Species <- NULL  
> (kc <- kmeans(newiris,3))  
> table(iris$Species,kc$cluster)  
> plot(newiris[c("Sepal.Length","Sepal.Width")],col=kc$cluster)  
> points(kc$centers[,c("Sepal.Length","Sepal.Width")],col=1:3,pch=8,cex=2)
```

Output:



Practical 5

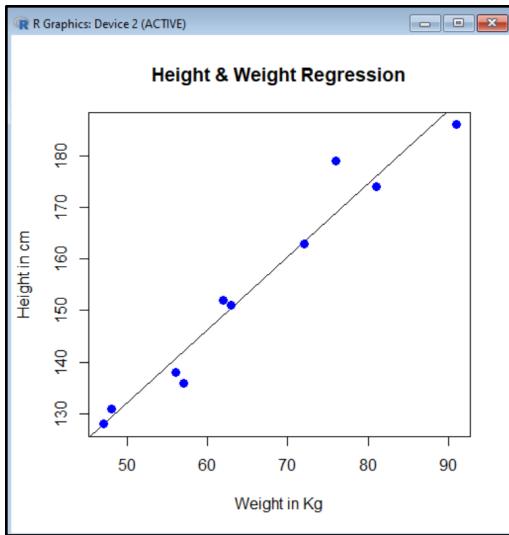
Perform the Linear regression on the given data warehouse data using R/Python.

Code:

```
> x <- c(151,174,138,186,128,136,179,163,152,131)
> y <- c(63,81,56,91,47,57,76,72,62,48)
> relation <- lm(y~x)
> png(file = "linearregression.png")
> plot(y,x,col = "blue",main = "Height & Weight Regression",abline(lm(x~y)),cex=1.3, pch=16,
xlab="Weight in Kg", ylab="Height in cm")
> dev.off()
```

Output:

```
> x <- c(151,174,138,186,128,136,179,163,152,131)
> y <- c(63,81,56,91,47,57,76,72,62,48)
> relation <- lm(y~x)
> png(file = "linearregression.png")
> plot(y,x,col = "blue",main = "Height & Weight Regression",abline(lm(x~y)),cex =1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")
> dev.off()
null device
      1
> plot(y,x,col = "blue",main = "Height & Weight Regression",abline(lm(x~y)),cex =1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")
|
```



Practical 6

Perform the logistic regression on the given data warehouse data using R/Python.

Code:

```
> quality<-read.csv("C:/quality.csv")
> str(quality)
> table(quality$PoorCare)
> install.packages("caTools")
> library(caTools)
> qualityTrain = subset(quality,split==TRUE)
> qualityTest =subset(quality,split==FALSE)
> nrow(qualityTrain)
> nrow(qualityTest)
> QualityLog=glm(PoorCare~OfficeVisits + Narcotics,data=qualityTrain,family = binomial)
> summary(QualityLog)
> predictTrain=predict(QualityLog,type = "response")
> summary(predictTrain)
```

```

> tapply(predictTrain, qualityTrain$PoorCare, mean)
> table(qualityTrain$PoorCare,predictTrain>0.5)
10/25
70/74
> table(qualityTrain$PoorCare,predictTrain>0.7)
8/25
73/74
> table(qualityTrain$PoorCare,predictTrain<0.2)
16/25
54/74

> install.packages("ROCR")
> library(ROCR)
> ROCRpred=prediction(predictTrain,qualityTrain$PoorCare)
> ROCRperf=performance(ROCRpred,"tpr","fpr")
> plot(ROCRperf)
> plot(ROCRperf,colorize=TRUE)
> plot(ROCRperf,colorize=TRUE,print.cutoffs.at=seq(0,1,by=0.1),text.adj=c(-0.2,10.7))

```

Practical 7

Write a Python program to read data from a CSV file, perform simple data analysis, and generate basic insights. (Use Pandas is a Python library).

Code:

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Load the CSV file into a DataFrame
file_path = 'C:\WineQT.csv'
df = pd.read_csv(file_path)
#df = pd.read.csv(file_path)

```

```
#Summary statistics
summary_stats = df.describe()
print("Summary Statistics")
print(summary_stats)

#Correlation matrix
correlation_matrix = df.corr()
print("\nCorrelation Matrix: ")
print(correlation_matrix)

#Quality distribution
quality_distribution = df['quality'].value_counts().sort_index()
print("\nQuality Distribution: ")
print(quality_distribution)

#Correlation with quality
quality_correlation = correlation_matrix['quality'].sort_values(ascending=False)
print("\nQuality Correlation: ")
print(quality_correlation)

# Correlation with quality
quality_correlation = correlation_matrix ['quality'].sort_values (ascending=False)
print("\nCorrelation with Quality:")
print (quality_correlation)

#Plotting
plt.figure (figsize=(10, 6))

#Quality distribution plot
plt.subplot (2, 2, 1)
sns.countplot (x='quality', data=df, palette='viridis')
plt.title('Quality Distribution')

#Heatmap of correlation matrix
plt.subplot(2, 2, 2)
sns.heatmap (correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')

#Alcohol vs Quality
plt.subplot (2, 2, 3)
```

```

sns.boxplot(x='quality', y='alcohol', data=df, palette='viridis')
plt.title('Alcohol vs Quality')

```

#Density vs Quality

```

plt.subplot(2, 2, 4)

```

```

sns.boxplot(x='quality', y='density', data=df, palette='viridis')

```

```

plt.title('Density vs Quality')

```

```

plt.tight_layout()

```

```

plt.show()

```

Output:

```

Python 3.11.3 (tags/v3.11.3:f3f2e88, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> %matplotlib notebook
RESTART: C:/Users/karan/AppData/Local/Programs/Python/Python311/prac7_scikit6.py =====
Summary Statistics
fixed acidity  volatile acidity ... quality Id
count          1143.000000 1143.000000 1143.000000
mean           8.311111  0.531339 ... 804.969379
std            1.411515  0.120000 ... 479.000000
min            4.600000  0.120000 ... 3.000000
25%           7.100000  0.392500 ... 5.000000 411.000000
50%           7.900000  0.520000 ... 6.000000 794.000000
75%           9.100000  0.640000 ... 6.000000 1209.500000
max          15.900000  1.580000 ... 8.000000 1597.000000
[8 rows x 13 columns]

Correlation Matrix:
fixed acidity  volatile acidity ... quality Id
fixed acidity  1.000000
volatile acidity -0.250728  1.000000 -0.407394 -0.007892
citric acid    0.673157 -0.544187 ... 0.240821 -0.139011
residual sugar 0.171831 -0.005751 ... 0.022002 -0.046344
chlorides      0.107889  0.056336 ... -0.124085 -0.086099

```

```

fixed acidity  volatile acidity ... quality Id
fixed acidity -0.114831 -0.001962 ... -0.063260 0.095268
total sulfur dioxide -0.110628  0.077748 ... -0.183339 -0.107389
density       0.681501  0.016512 ... -0.175208 -0.363926
pH            0.6895163 0.221492 ... -0.052453 0.132904
sulphates     0.174590 -0.276309 ... 0.247712 0.103954
alcohol        0.010559 -0.023909 ... 0.484866 0.079097
quality       0.121970 -0.407394 ... 1.000000 0.069708
Id            -0.275826 -0.007892 ... 0.069708 1.000000
[13 rows x 13 columns]

Quality Distribution:
quality
3   6
4   33
5   483
6   462
7   143
8   16
Name: count, dtype: int64

Quality Correlation:
quality  1.000000
alcohol  0.494866
sulphates 0.257710

```

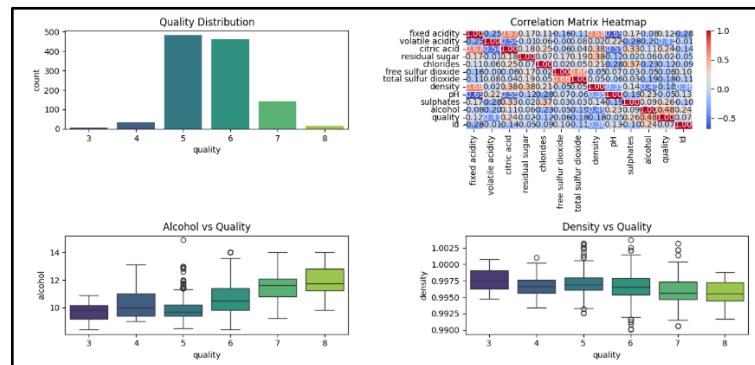
```

sulfur dioxide  0.257710
citric acid    0.240821
fixed acidity  0.121970
Id             0.069708
residual sugar 0.022002
pH            -0.052453
free sulfur dioxide -0.063260
chlorides      -0.124085
density        -0.175208
total sulfur dioxide -0.183339
volatile acidity -0.407394
Name: quality, dtype: float64

Correlation with Quality:
quality  1.000000
alcohol  0.494866
sulphates 0.257710
citric acid 0.121970
fixed acidity 0.069708
residual sugar 0.022002
pH        -0.052453
free sulfur dioxide -0.063260
chlorides      -0.124085
density        -0.175208
total sulfur dioxide -0.183339
volatile acidity -0.407394
Name: quality, dtype: float64

Correlation with Quality:
quality  1.000000
alcohol  0.494866
sulphates 0.257710
citric acid 0.121970
fixed acidity 0.069708
residual sugar 0.022002
pH        -0.052453
free sulfur dioxide -0.063260
chlorides      -0.124085
density        -0.175208
total sulfur dioxide -0.183339
volatile acidity -0.407394
Name: quality, dtype: float64

```



Practical 8

Perform data visualization

- a. Perform data visualization using Python on any sales data.

Code:

```
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns
```

```

# Step 1: Load the CSV data into a pandas DataFrame
def load_data(csv_file):
    try:
        data = pd.read_csv('C:\sales_data.csv', encoding="ISO-8859-1")
        print("Data loaded successfully!")
        print(data)
        return data
    except Exception as e:
        print(f"Error loading data: {e}")
        return None

# Step 2: Perform basic data inspection
def data_summary(data):
    print("\nFirst 5 rows of the data:")
    print(data.head()) # Displays the first 5 rows

    print("\nData Structure:")
    print(data.info()) # Get information on the DataFrame such as column types and non-null
counts

    print("\nStatistical summary of numeric columns:")
    print(data.describe()) # Gives summary statistics for numeric columns

# Step 3: Visualize the data

# Visualization 1: Total Sales Over Time
def plot_sales_over_time(data):
    # Convert 'Date' to datetime if it's not already in the correct format
    data['Date'] = pd.to_datetime(data['Date'])

    # Aggregate data by Date and calculate the total sales for each day
    sales_by_date = data.groupby('Date')['Sales_Amount'].sum().reset_index()

    plt.figure(figsize=(10, 6))
    sns.lineplot(x='Date', y='Sales_Amount', data=sales_by_date, marker='o')
    plt.title('Total Sales Over Time')
    plt.xlabel('Date')
    plt.ylabel('Total Sales ($)')
    plt.xticks(rotation=45)
    plt.tight_layout()

```

```

plt.show()

# Visualization 2: Sales Distribution by Region
def plot_sales_by_region(data):
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='Region', y='Sales_Amount', data=data)
    plt.title('Sales Distribution by Region')
    plt.xlabel('Region')
    plt.ylabel('Sales Amount ($)')
    plt.tight_layout()
    plt.show()

# Visualization 3: Sales vs Quantity Sold (Scatter Plot)
def plot_sales_vs_quantity(data):
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x='Quantity_Sold', y='Sales_Amount', data=data, hue='Product',
                    palette='viridis')
    plt.title('Sales Amount vs Quantity Sold')
    plt.xlabel('Quantity Sold')
    plt.ylabel('Sales Amount ($)')
    plt.tight_layout()
    plt.show()

# Visualization 4: Top Products by Sales
def plot_top_products_by_sales(data):
    top_products = data.groupby('Product')['Sales_Amount'].sum().reset_index()
    top_products = top_products.sort_values('Sales_Amount', ascending=False).head(10)

    plt.figure(figsize=(10, 6))
    sns.barplot(x='Sales_Amount', y='Product', data=top_products, palette='Blues_d')
    plt.title('Top 10 Products by Sales Amount')
    plt.xlabel('Sales Amount ($)')
    plt.ylabel('Product')
    plt.tight_layout()
    plt.show()

# Main function to load data and generate visualizations
def main():
    csv_file = 'sales_data.csv' # Replace with your actual CSV file path
    data = load_data(csv_file)

```

if data is not None:

```
# Step 2: Data summary and inspection
data_summary(data)
```

```
# Step 3: Data Visualizations
```

```
plot_sales_over_time(data)
plot_sales_by_region(data)
plot_sales_vs_quantity(data)
plot_top_products_by_sales(data)
```

```
# Entry point of the program
```

```
if __name__ == "__main__":
    main()
```

Output:

```
RESTART: C:/Users/Karan/AppData/Local/Programs/Python/Python311/prac8a_se6.py ======
Data loaded successfully!
[5 rows x 26 columns]

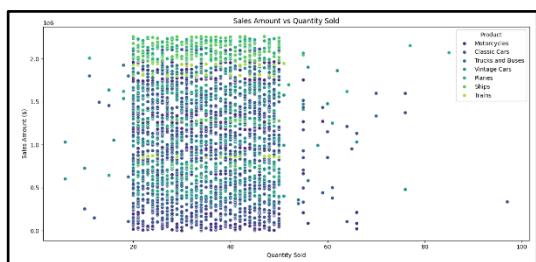
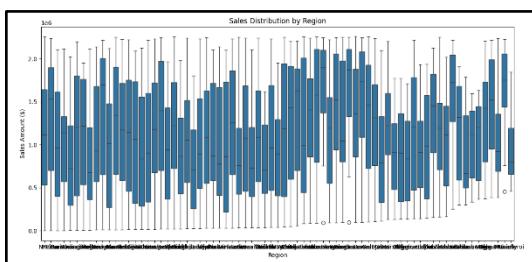
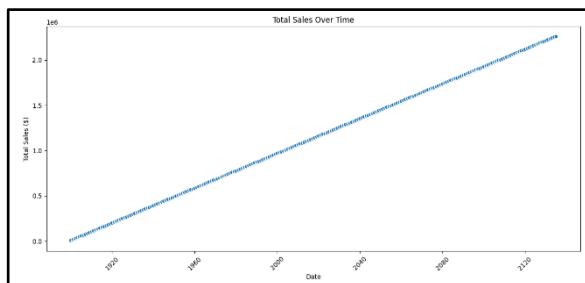
Data Structure:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype  
--- 
 0   ORDERNUMBER      2823 non-null   int64  
 1   Quantity_Sold   2823 non-null   object 
 2   DEALSIZE        2823 non-null   object 
 3   Sales_Amount    2823 non-null   float64
 4   ...              ...           ...
 5   ...              ...           ...
 6   ...              ...           ...
 7   ...              ...           ...
 8   ...              ...           ...
 9   ...              ...           ...
 10  ...              ...           ...
 11  ...              ...           ...
 12  ...              ...           ...
 13  ...              ...           ...
 14  ...              ...           ...
 15  ...              ...           ...
 16  ...              ...           ...
 17  Region          2823 non-null   object 
 18  STATE           1337 non-null   object 
 19  PRODUCTCODE    2743 non-null   object 
 20  COUNTRY         2823 non-null   object 
 21  TERRITORY       1749 non-null   object 
 22  CONTACTLASTNAME 2823 non-null   object 
 23  CONTACTFIRSTNAME 2823 non-null   object 
 24  DEALSIZE        2823 non-null   object 
 25  Sales_Amount    2823 non-null   int64 
dtypes: float64(1), int64(8), object(16)
memory usage: 573.6+ KB
None
```

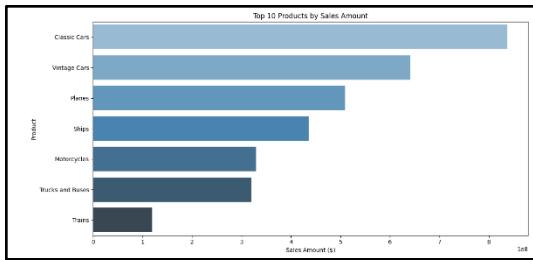
```
RESTART: C:/Users/Karan/AppData/Local/Programs/Python/Python311/prac8a_se6.py ======
[5 rows x 26 columns]

Data Structure:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype  
--- 
 0   ORDERNUMBER      2823 non-null   int64  
 1   Quantity_Sold   2823 non-null   int64  
 2   DEALSIZE        2823 non-null   float64
 3   PRICEEACH       2823 non-null   float64
 4   ORDERLINENUMBER 2823 non-null   int64  
 5   Date            2823 non-null   object 
 6   STATUS          2823 non-null   object 
 7   YEAR_ID         2823 non-null   int64  
 8   MONTH_ID        2823 non-null   int64  
 9   YEAR_ID         2823 non-null   int64  
 10  Product         2823 non-null   object 
 11  Month           2823 non-null   int64  
 12  PRODUCTCODE    2823 non-null   object 
 13  CUSTOMERNAME   2823 non-null   object 
 14  ADDRESSLINE1    2823 non-null   object 
 15  ADDRESSLINE1    2823 non-null   object 
 16  ADDRESSLINE2    302 non-null    object
```

```
RESTART: C:/Users/Karan/AppData/Local/Programs/Python/Python311/prac8a_se6.py ======
[8 rows x 10 columns]

Statistical summary of numeric columns:
   ORDERNUMBER  Quantity_Sold  ...  Sales_Amount
count    2823.000000  2823.000000  ...  2823.000000
mean    10109.000000  35.000000  ...  2.823000e+03
std     92.05478  9.741443  ...  40.187912  6.520594e+05
min    10100.000000  6.000000  ...  33.000000  1.000000e+03
25%    10102.000000  25.000000  ...  49.000000  1.100000e+03
50%    10262.000000  35.000000  ...  99.000000  1.130100e+06
75%    10333.500000  43.000000  ...  124.000000  1.694500e+06
max    10425.000000  97.000000  ...  214.000000  2.258900e+06
```





b. Perform data visualization using PowerBI on any sales data.

Step 1: Install and Open Power BI

- 1. Download & Install Power BI Desktop**

- If you haven't installed it yet, download it from [Power BI Download](#).
- Install and open **Power BI Desktop**.

Step 2: Load the Sales Data

1. Open **Power BI Desktop**.
2. Click on "Home" > "Get Data" > "Text/CSV".
3. Browse and select the **sales_data_sample.csv** file you uploaded.
4. Click **Open**, then click **Load** (you can check the data preview before loading).

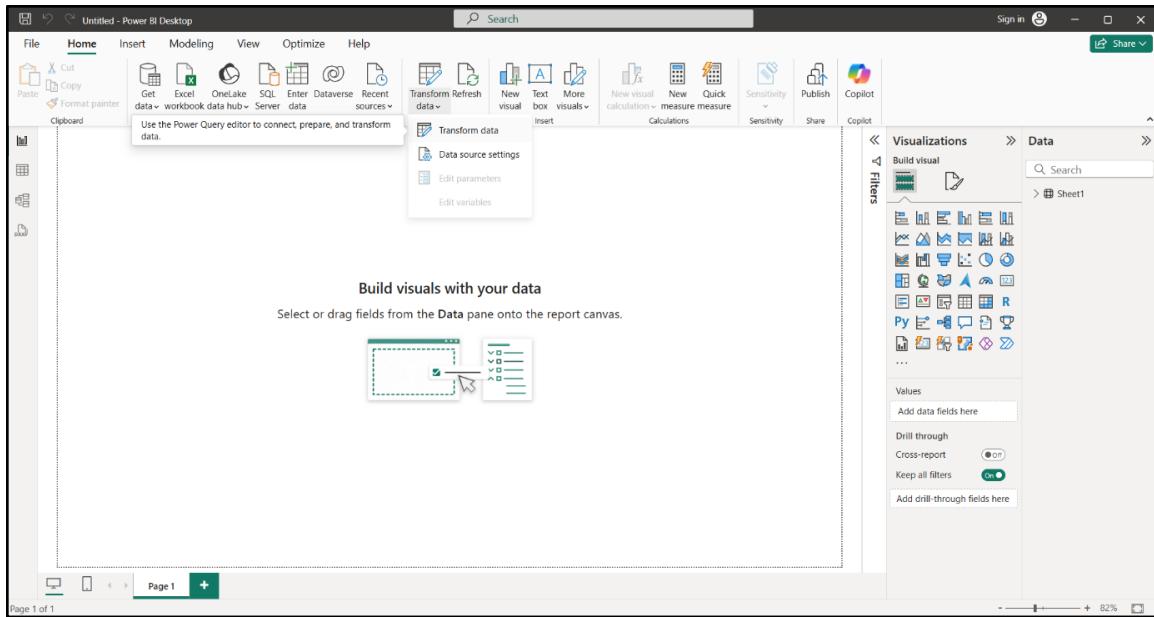
The screenshot shows the Power BI Navigator window. On the left, there's a tree view under 'Display Options' showing a folder named 'sales_data_sample.xlsx [1]' with a single item 'Sheet1' checked. The main area is titled 'Sheet1' and displays a table of sales data with columns: ORDERNUMBER, QUANTITYORDERED, PRICEEACH, ORDERLINENUMBER, and SALES. The data consists of 30 rows of sales records. Below the table, a message says 'The data in the preview has been truncated due to size limits.' At the bottom are buttons for 'Load', 'Transform Data', and 'Cancel'.

ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES
10107	30	95.7	2	2871
10121	34	81.35	5	2765.9
10134	41	94.74	2	3884.34
10145	45	83.26	6	3746.7
10159	49	100	14	5205.27
10168	36	96.66	1	3479.76
10180	29	86.13	9	2497.77
10188	48	100	1	5512.32
10201	22	98.57	2	2168.54
10211	41	100	14	4708.44
10223	37	100	1	3965.66
10237	23	100	7	2333.12
10251	28	100	2	3188.64
10263	34	100	2	3676.76
10275	45	92.83	1	4177.35
10285	36	100	6	4099.68
10299	23	100	9	2597.39
10309	41	100	5	4394.38
10318	46	94.74	1	4358.04
10329	42	100	1	4396.14

Step 3: Transform Data in Power Query Editor

Before building reports, let's clean and format the data:

1. Click on "Transform Data" to open **Power Query Editor**.



2. Review columns and check for missing values:

- **ADDRESSLINE2, STATE, TERRITORY** have missing values, so you can remove them or fill in with "Unknown".
- To remove a column: Right-click on the column header > Click Remove.

Queries [1] = Table.RemoveColumns(#"Changed Type", {"ADDRESSLINE2"})

Sheet1

	ABC PHONE	ADDRESSLINE1	CITY	STATE	POSTALCODE
1	212557818	897 Long Airport Avenue	NYC	NY	
2	26.47.1555	59 rue de l'Abbaye	Reims		
3	+33 1 46 62 7555	27 rue du Colonel Pierre Avia	Paris		
4	626557265	78934 Hillside Dr.	Pasadena	CA	
5	650553186	7734 Strong St.	San Francisco	CA	
6	650556809	9408 Furth Circle	Burlingame	CA	
7	20.16.1555	184, chausse de Tournai	Lille		
8	+47 2267 3215	Drammen 121, PR 744 Sentrum	Bergen		
9	650555787	5557 North Pendale Street	San Francisco	CA	
10	(1) 47.55.6555	25, rue Lauriston	Paris		
11	395204555	636 St Kilda Road	Melbourne	Victoria	
12	2125515100	2678 Kingston Rd.	NYC	NY	
13	2015559350	7476 Moss Rd.	Newark	NJ	
14	2035552970	25993 South Bay Ln.	Bridgewater	CT	
15	40.67.8555	67, rue des Cinqante Otages	Nantes		
16	617555855	39323 Spinnaker Dr.	Cambridge	MA	
17	90-224 8555	Keskuskatu 45	Helsinki		
18	07-98 9555	Erling Skakkes gate 78	Stavern		
19	2155515155	7586 Pompton St.	Allentown	PA	
20	3116667010	9871 Long Airport Avenue	MVR		

24 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 13:10

3. Convert ORDERDATE column to Date format:

- Select ORDERDATE column.
- In the top menu, click Transform > Data Type > Date.

Queries [1] = Table.RemoveColumns(#"Changed Type", {"ADDRESSLINE2", "STATE", "TERRITORY"})

Sheet1

	NUMBER	1_2 SALES	ORDERSDATE	STATUS	QTR_ID
1	Time	2	2871	24-02-2003 00:00:00	Shipped
2	Date/Time/Timezone	5	2765.9	07-05-2003 00:00:00	Shipped
3	Duration	2	3884.34	01-07-2003 00:00:00	Shipped
4	Text	6	3746.7	25-08-2003 00:00:00	Shipped
5	True/False	14	5205.27	10-10-2003 00:00:00	Shipped
6	Binary	1	3479.76	28-10-2003 00:00:00	Shipped
7	86.13	9	2497.77	11-11-2003 00:00:00	Shipped
8	100	1	5512.32	18-11-2003 00:00:00	Shipped
9	98.57	2	2168.54	01-12-2003 00:00:00	Shipped
10	100	14	4708.44	15-01-2004 00:00:00	Shipped
11	100	1	3965.66	20-02-2004 00:00:00	Shipped
12	100	7	2333.12	05-04-2004 00:00:00	Shipped
13	100	2	3188.64	18-05-2004 00:00:00	Shipped
14	100	2	3676.76	28-06-2004 00:00:00	Shipped
15	92.83	1	4177.35	23-07-2004 00:00:00	Shipped
16	100	6	4099.68	27-08-2004 00:00:00	Shipped
17	100	9	2597.39	30-09-2004 00:00:00	Shipped
18	100	5	4394.38	15-10-2004 00:00:00	Shipped
19	94.74	1	4358.04	02-11-2004 00:00:00	Shipped
20	100	1	4202.14	15-11-2004 00:00:00	Shipped

22 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 13:10

4. Ensure SALES, PRICEEACH are in Decimal Number format.

Power Query Editor - Untitled - Power Query Editor

Queries [1]

Sheet1

Table

22 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 13:10

APPLIED STEPS

- Removed Columns

5. Click "Close & Apply".

Power Query Editor - Untitled - Power Query Editor

Queries [1]

Sheet1

Table

22 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 13:10

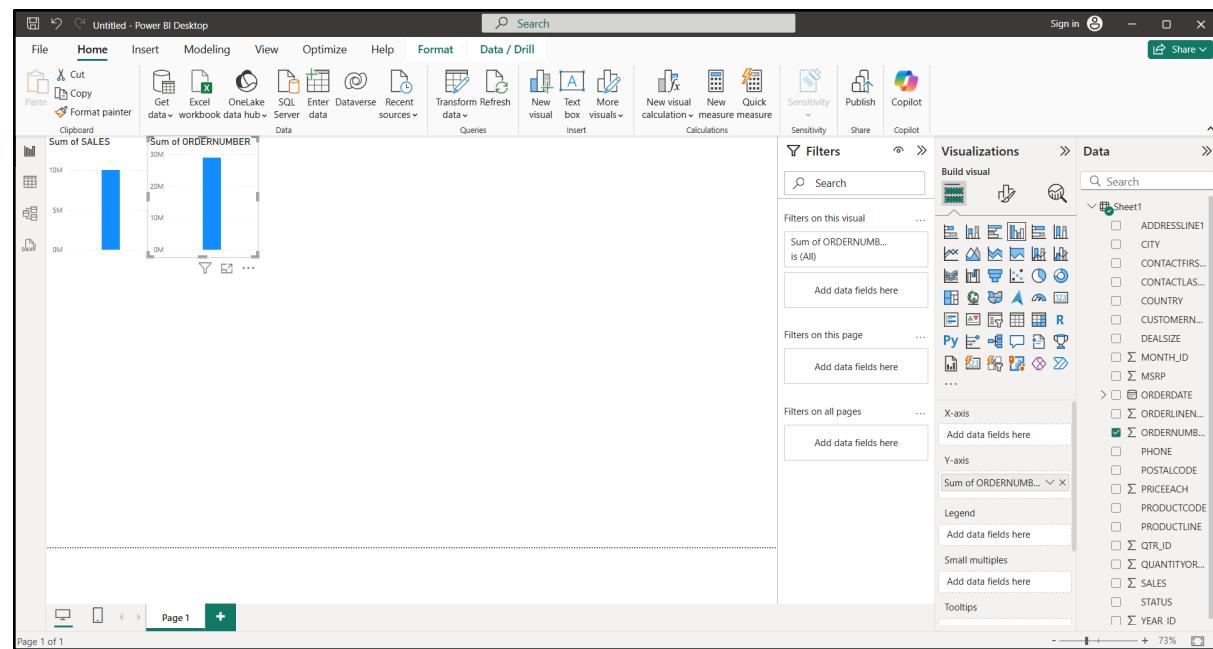
APPLIED STEPS

- Removed Columns

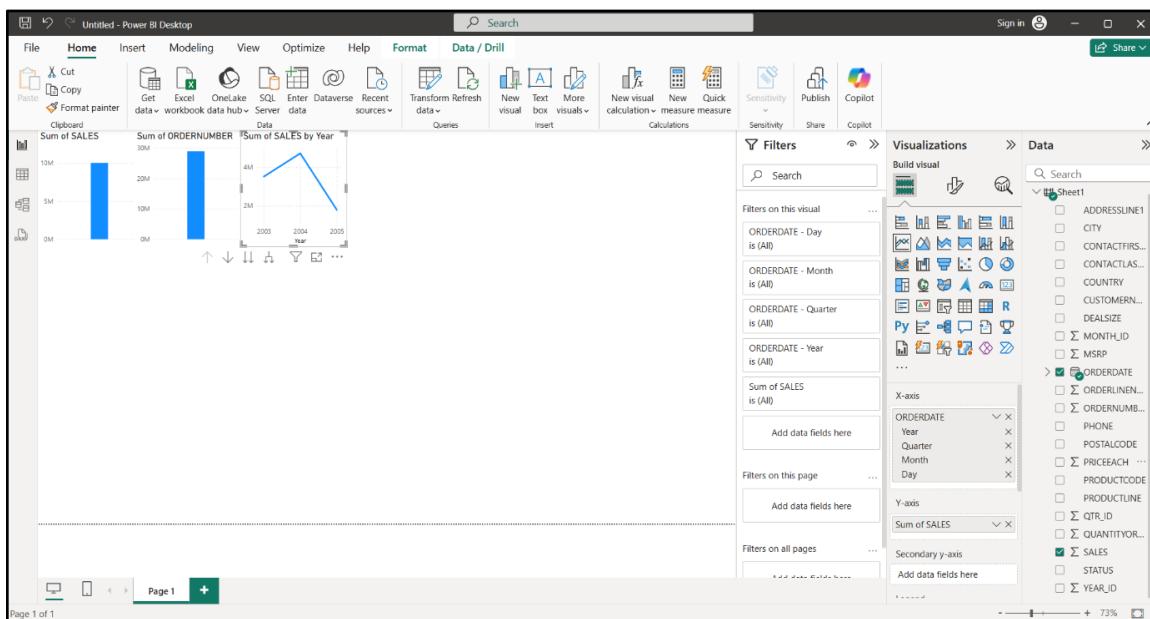
Step 4: Create Basic Visualizations

1. Sales Performance Overview

- Total Sales & Orders
 1. Go to Report View (bottom left panel).
 2. Drag SALES to a Card Visual (this shows total revenue).
 3. Drag ORDERNUMBER to another Card Visual (shows total orders).

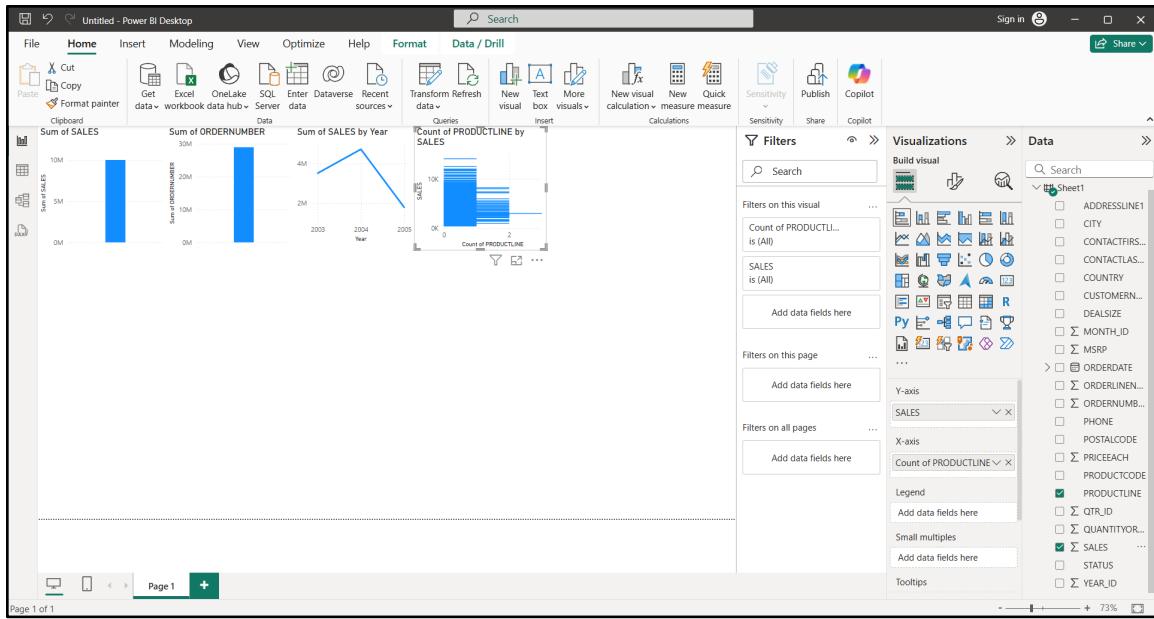


- **Sales Trend Over Time**
 4. Drag ORDERDATE to X-Axis in a Line Chart.
 5. Drag SALES to Y-Axis (shows sales over time).



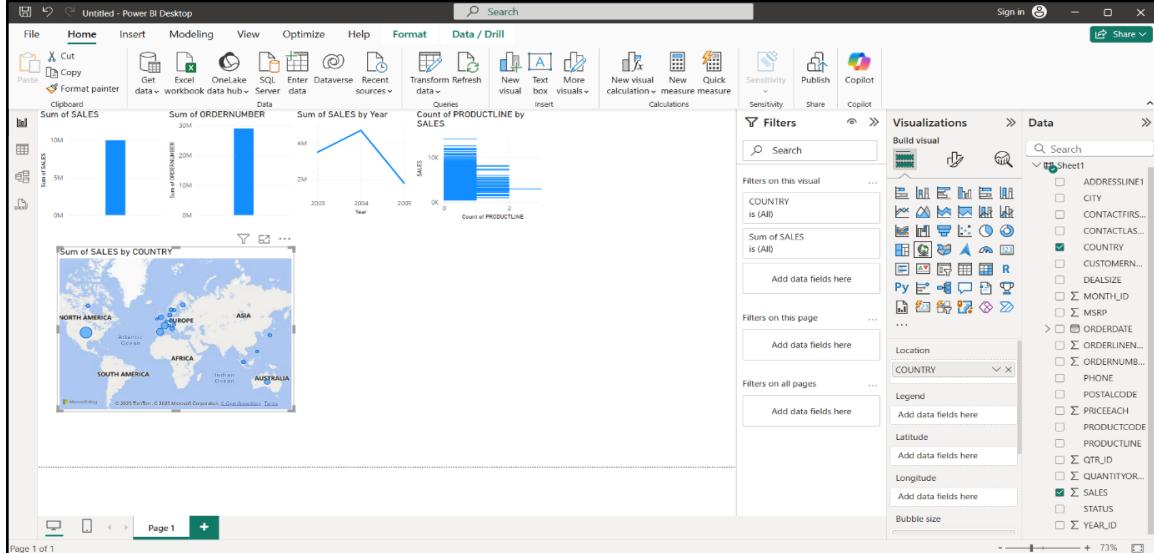
2. Sales by Product Line

1. Insert a Bar Chart.
2. Drag PRODUCTLINE to X-Axis.
3. Drag SALES to Y-Axis (shows revenue per product category).



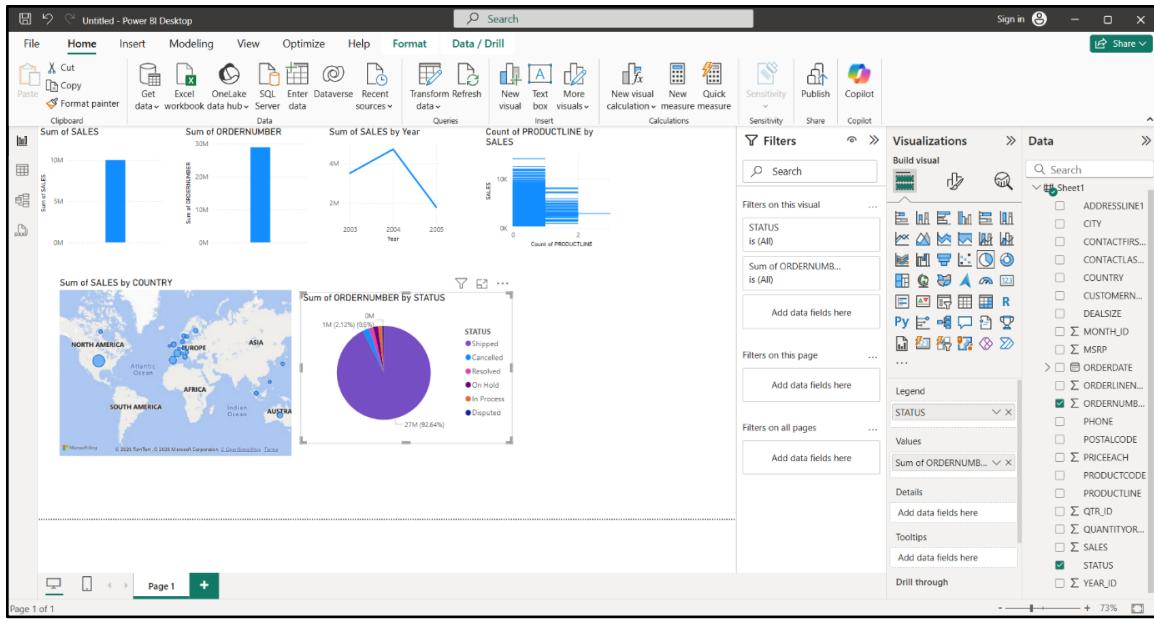
3. Sales by Country

1. Insert a Map Visual (Globe icon).
2. Drag COUNTRY to Location field.
3. Drag SALES to Values field (shows sales by country).



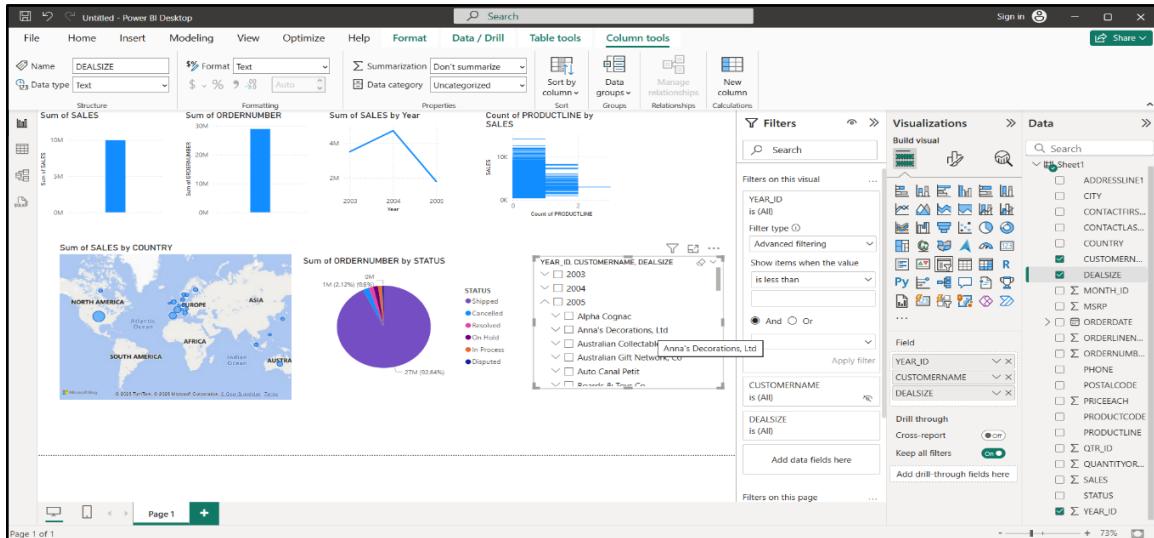
4. Order Status Breakdown

1. Insert a Pie Chart.
2. Drag STATUS to Legend.
3. Drag ORDERNUMBER to Values (shows percentage of orders in each status).



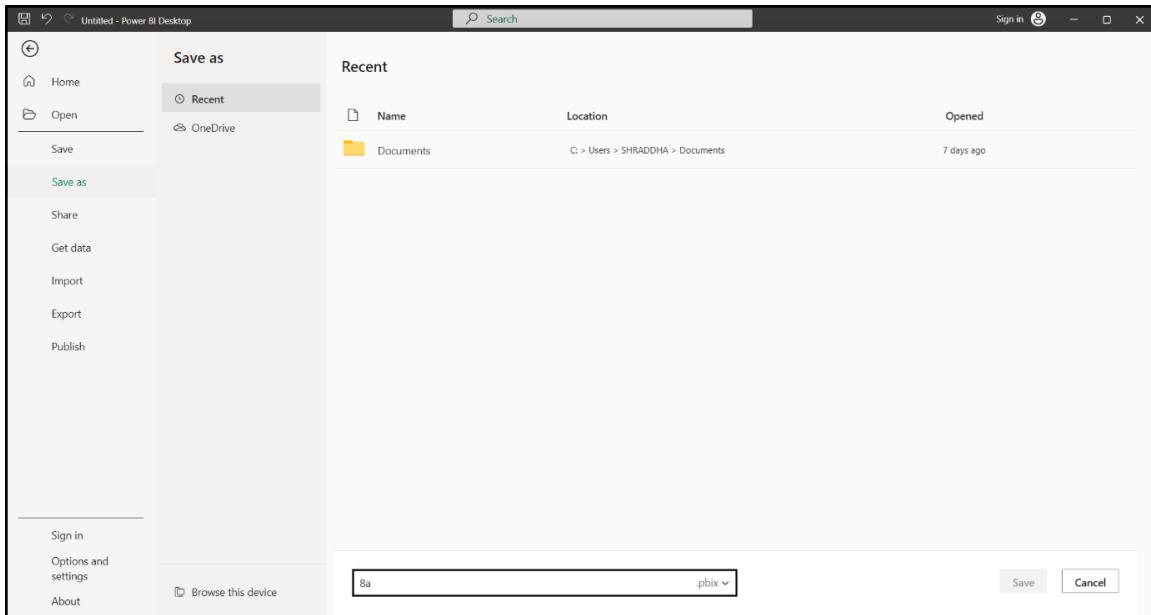
Step 5: Add Filters and Slicers

- Click on "Slicer" and add YEAR_ID to filter by year.
- Add CUSTOMERNAME to filter by customer.
- Add DEALSIZE to filter by deal size.



Step 6: Save & Publish Report

- Click File > Save As and save the Power BI report.
- Click Publish to share on Power BI Service (if needed).



Steps to Perform Data Visualization in Power BI

1. Import the Data:

- Open Power BI Desktop.
- Click on "Get Data" > "Excel" and select the uploaded file.
- Load the relevant sheets into Power BI.

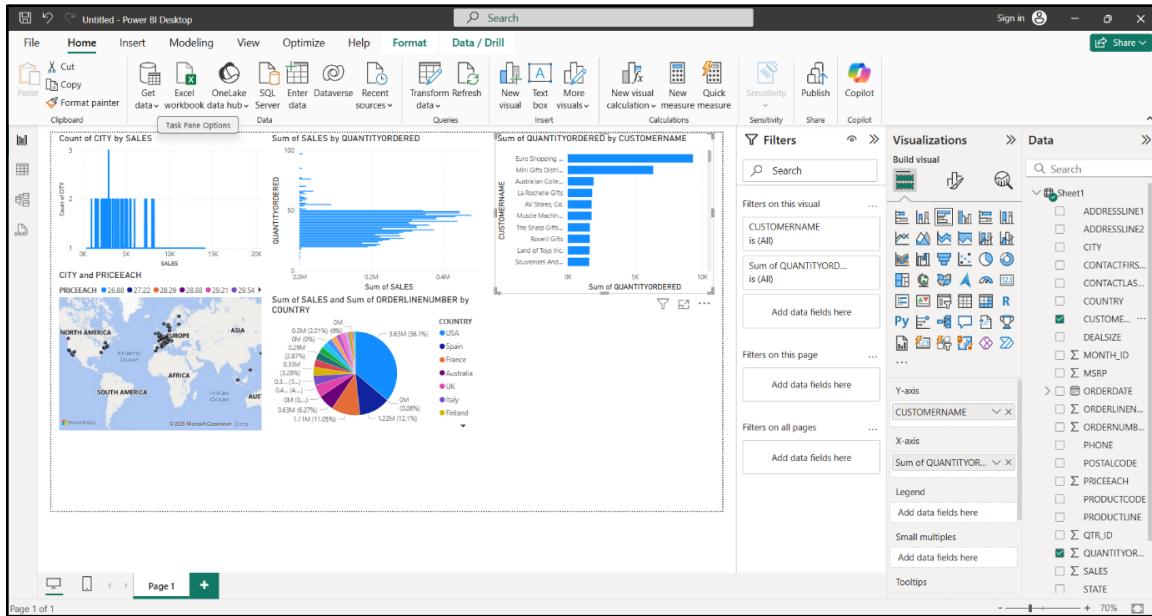
2. Data Preparation:

- Check for missing values or inconsistent data.
- Apply transformations (if needed) using Power Query Editor.

3. Create Key Visuals:

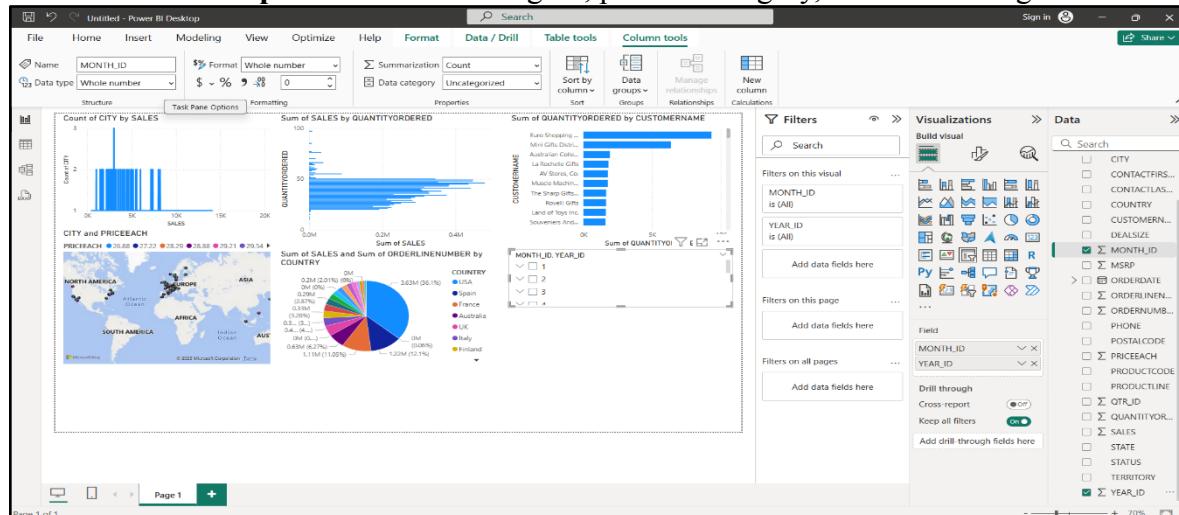
- **Sales Trends:** Use a Line Chart to show sales performance over time.
- **Regional Sales Analysis:** Use a Map Visual to display sales by region.
- **Top Products Sold:** Use a Bar Chart to highlight best-selling products.

- **Revenue Breakdown:** Use a **Pie Chart** or **Treemap** to show sales by category.
- **Customer Segmentation:** Use **Clustered Bar Charts** to group customers based on sales.



4. Add Filters and Slicers:

- Include **Date Slicers** to filter data by month, quarter, or year.
- Use **Dropdown Filters** for region, product category, or customer segment.



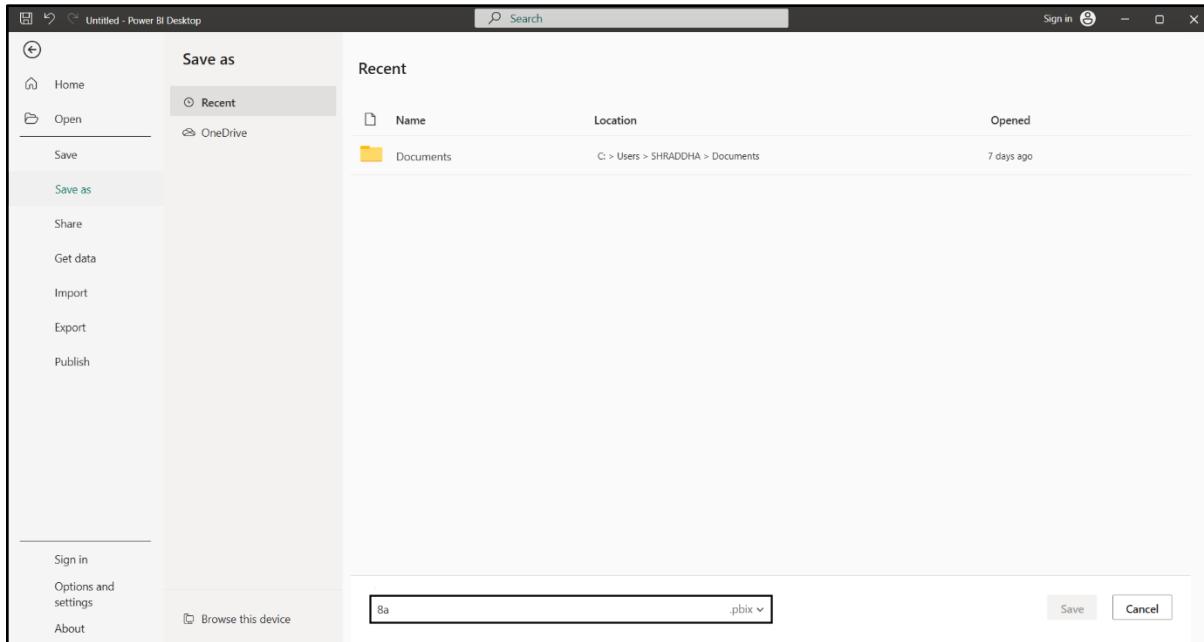
5. Enhance with DAX Measures:

- Create calculated measures such as:
- $\text{Total Sales} = \text{SUM}(\text{Sales}[\text{Revenue}])$
- $\text{Sales Growth} = (\text{[Total Sales]} - \text{PREVIOUSMONTH}([\text{Total Sales}])) / \text{PREVIOUSMONTH}([\text{Total Sales}])$
- Use **KPIs** for profit margin and sales targets.

```
1 Total Sales = SUM(Sales[Revenue])
2 Sales Growth = ([Total Sales] - PREVIOUSMONTH([Total Sales])) / PREVIOUSMONTH([Total Sales])
```

6. Publish and Share:

- Save the report and publish it to **Power BI Service**.
- Share with stakeholders via dashboards.



Practical 9

Create the Data staging area for the selected database using SQL.

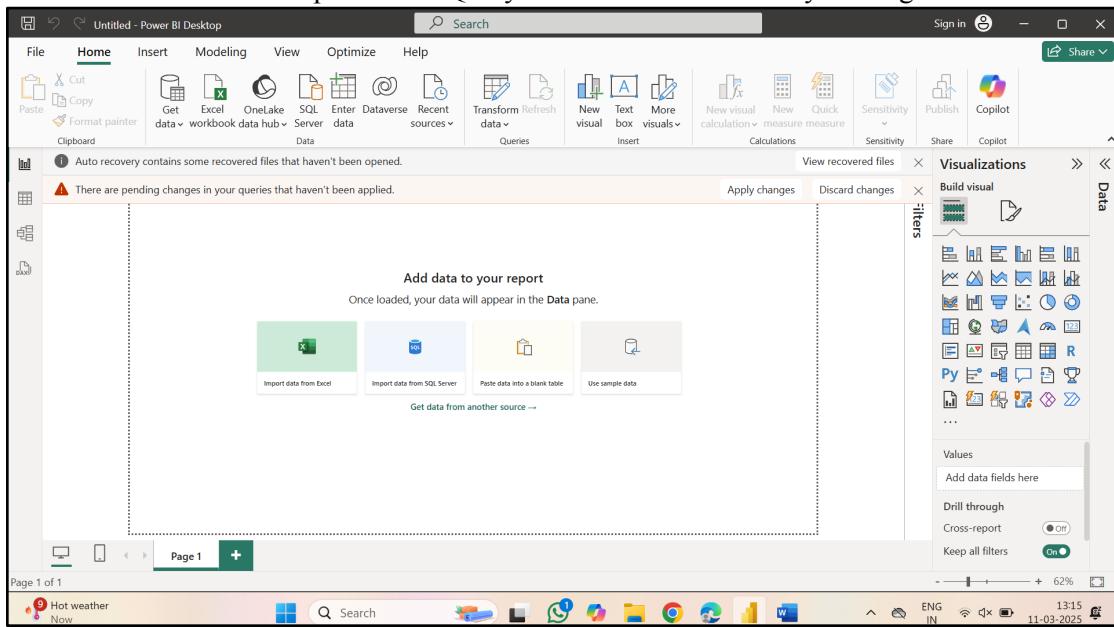
1. Load Data into Power BI

Open Power BI Desktop.

Click Home > Get Data > Excel Workbook.

Browse and select sales_data_sample.xlsx.

Click Transform Data to open Power Query Editor instead of directly loading the data.



This screenshot shows the Power BI Desktop interface with the "Transform Data" dialog box open over the main workspace. The dialog is titled "sales_data_sample.csv" and contains a preview of the data with 200 rows. It includes sections for "File Origin" (set to "1252: Western European (Windows)"), "Delimiter" (set to "Comma"), and "Data Type Detection" (based on the first 200 rows). Below the preview, there are buttons for "Extract Table Using Examples", "Load", "Transform Data", and "Cancel". The main workspace shows the same Power BI interface as the previous screenshot, with the ribbon, visualizations pane, and status bar visible.

2. Perform Data Transformations in Power Query

A. Inspect and Rename Columns

Check all column names and rename them to follow a clear naming convention (e.g., Order_Date instead of order date).

Untitled - Power Query Editor

Queries [1]

= Table.RenameColumns(#"Changed Type", {{"ORDERDATE", "Order_Date"}})

L2 SALES

	Order_Date	STATUS	QTR_ID	MONTH_ID
1	2871 2/24/2003 0:00	Shipped	2	
2	2765.9 5/7/2003 0:00	Shipped	2	
3	3884.34 7/1/2003 0:00	Shipped	3	
4	3746.7 8/25/2003 0:00	Shipped	3	
5	5205.27 10/10/2003 0:00	Shipped	4	
6	3479.76 10/28/2003 0:00	Shipped	4	
7	2497.77 11/11/2003 0:00	Shipped	4	
8	5512.32 11/18/2003 0:00	Shipped	4	
9	2168.54 12/1/2003 0:00	Shipped	4	
10	4708.44 1/15/2004 0:00	Shipped	1	
11	3965.66 2/20/2004 0:00	Shipped	1	
12	2333.12 4/5/2004 0:00	Shipped	2	
13	3188.64 5/18/2004 0:00	Shipped	2	
14	3676.76 6/28/2004 0:00	Shipped	2	
15	4177.35 7/23/2004 0:00	Shipped	3	
16	4099.68 8/27/2004 0:00	Shipped	3	
17	2597.39 9/30/2004 0:00	Shipped	3	
18	4394.38 10/15/2004 0:00	Shipped	4	
19	4358.04 11/1/2004 0:00	Shipped	4	
20	4396.14 11/15/2004 0:00	Shipped	4	
21				

25 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 13:21

PROPERTIES

Name: sales_data_sample

APPLIED STEPS

- Source
- Promoted Headers
- Changed Type
- Renamed Columns

B. Remove Unnecessary Columns

If there are extra columns not needed for analysis, remove them to improve efficiency.

Untitled - Power Query Editor

Queries [1]

= Table.RemoveColumns(#"Renamed Columns", {"ADDRESSLINE1", "ADDRESSLINE2", "STATE", "POSTALCODE"})

L2 SALES

	PHONE	CITY	COUNTRY	TERRITORY	CONTACTS
1	2125557818	NYC	USA	NA	Yu
2	26.47.1555	Reims	France	EMEA	Henriot
3	+33 1 46 62 7555	Paris	France	EMEA	Da Cunha
4	6265557265	Pasadena	USA	NA	Young
5	o.	San Francisco	USA	NA	Brown
6	6505556809	Burlingame	USA	NA	Hirano
7	20.16.1555	Lille	France	EMEA	Rance
8	+47 2267 3215	Bergen	Norway	EMEA	Oeztan
9	6505559787	San Francisco	USA	NA	Murphy
10	(1) 47.55.6555	Paris	France	EMEA	Perrier
11	Co.	Melbourne	Australia	APAC	Ferguson
12	03 9520 4555	NYC	USA	NA	Frick
13	2125551500	Newark	USA	NA	Brown
14	2015559350	Bridgewater	USA	NA	King
15	40.67.8555	Nantes	France	EMEA	Labrune
16	6175558555	Cambridge	USA	NA	Hernandez
17	90-224 8555	Helsinki	Finland	EMEA	Karttunen
18	07-98 9555	Stavern	Norway	EMEA	Bergulfsen
19	21255551555	Allentown	USA	NA	Yu
20	2125557818	NYC	USA	NA	Yu
21					

21 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 13:22

PROPERTIES

Name: sales_data_sample

APPLIED STEPS

- Source
- Promoted Headers
- Changed Type
- Renamed Columns
- Removed Columns

C. Handle Missing Data

Check for null values in key fields such as Order_ID, Customer_Name, Product_Code, etc.

Apply one of the following:

Use Remove Rows > Remove Blank Rows for completely empty records.

Queries [1] ✓

= Table.SelectRows(#"Removed Columns", each not List.IsEmpty(List.RemoveMatchingItems

	PHONE	CITY	COUNTRY	TERRITORY	CONTACTS
1	2125557818	NYC	USA	NA	Yu
2	26.47.1555	Reims	France	EMEA	Henriot
3	+33 1 46 62 7555	Paris	France	EMEA	Da Cunha
4	6265557265	Pasadena	USA	NA	Young
5	6505551386	San Francisco	USA	NA	Brown
6	6505556809	Burlingame	USA	NA	Hirano
7	20.16.1555	Lille	France	EMEA	Rance
8	+47 2267 3215	Bergen	Norway	EMEA	Oeztan
9	6505555787	San Francisco	USA	NA	Murphy
10	(1) 47.55.6555	Paris	France	EMEA	Perrier
11	03 9520 4555	Melbourne	Australia	APAC	Ferguson
12	2125551500	NYC	USA	NA	Frick
13	2015559350	Newark	USA	NA	Brown
14	2035552570	Bridgewater	USA	NA	King
15	40.67.8555	Nantes	France	EMEA	Labrune
16	6175558555	Cambridge	USA	NA	Hernandez
17	90-224 8555	Helsinki	Finland	EMEA	Karttunen
18	07-98 9555	Stavern	Norway	EMEA	Bergulfsen
19	2155551555	Allentown	USA	NA	Yu
20	2125557818	NYC	USA	NA	Yu
21

21 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 13:24

Use Replace Values to fill nulls with "Unknown" (for text) or 0 (for numeric fields).

Use Fill Down (for fields where previous values should be carried forward).

D. Change Data Types

Ensure that each column has the correct data type:

Dates: Convert to Date type (Order_Date, Ship_Date).

Numbers: Convert Quantity, Price, Total_Amount to Decimal Number or Whole Number.

Text: Convert Customer_Name, Product_Name, Region to Text.

E. Remove Duplicates

Identify and remove duplicates based on Order_ID or Invoice_ID (whichever is the unique identifier).

Queries [1] ✓

= Table.Distinct(#"Changed Type2", {"ORDERNUMBER"})

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES
1	10107	30.00	95.7	2	2
2	10121	34.00	81.35	5	276
3	10134	41.00	94.74	2	3884
4	10145	45.00	83.26	6	374
5	10159	49.00	100	14	5205
6	10168	36.00	96.66	1	3475
7	10180	29.00	86.13	9	2497
8	10188	48.00	100	1	5512
9	10201	22.00	98.57	2	2166
10	10211	41.00	100	14	4706
11	10223	37.00	100	1	3965
12	10237	23.00	100	7	2335
13	10251	28.00	100	2	3186
14	10263	34.00	100	2	3676
15	10275	45.00	92.83	1	4177
16	10285	36.00	100	6	4095
17	10299	23.00	100	9	2597
18	10309	41.00	100	5	4394
19	10318	46.00	94.74	1	4356
20	10329	42.00	100	1	4396
21

20 COLUMNS, 307 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 13:28

F. Standardize Text Formatting

Convert text fields to Proper Case (e.g., Product Name, Customer Name):

Text.Proper([Column_Name])

Trim spaces from text fields to avoid mismatches:

Text.Trim([Column_Name])

The screenshot shows the Power Query Editor interface with the following details:

- File**, **Home**, **Transform**, **Add Column**, **View**, **Tools**, **Help** menu items.
- Transform ribbon** with various tools like Transpose, Data Type, Replace Values, Unpivot Columns, Move, Pivot Column, Convert to List, Split Column, Format, Parse, Statistics, Standard, Scientific, Trigonometry, Date, Time, Duration, Run R script, and Run Python script.
- Queries [1]** pane showing the query name `sales_data_sample`.
- Preview pane** showing the transformed data with columns: ORDERLINENUMBER, SALES, STATUS, QTR_ID, and MONTH_ID.
- Applied Steps pane** listing steps: Source, Promoted Headers, Changed Type, Renamed Columns, Removed Columns, Removed Blank Rows, Replaced Value, Changed Type1, Removed Columns1, Changed Type2, Removed Duplicates, Capitalized Each Word, and Trimmed Text (highlighted).
- Query Settings pane** with Name set to `sales_data_sample`.
- Bottom status bar: 20 COLUMNS, 307 ROWS, Column profiling based on top 1000 rows, PREVIEW DOWNLOADED AT 13:30.

3. Create Staging Tables

Fact Table (FactSales):

Keep transactional data such as Order_ID, Date, Product_Code, Quantity, Total_Sales.

The screenshot shows the Power Query Editor interface with the following details:

- File**, **Home**, **Transform**, **Add Column**, **View**, **Tools**, **Help** menu items.
- Transform ribbon** with various tools like Close & Apply, New Source, Recent Sources, Enter Data, Data source settings, Manage Parameters, Refresh, Advanced Editor, Choose Columns, Remove Columns, Keep Rows, Remove Rows, Reduce Rows, Sort, Data Type, Use First Row as Headers, Merge Queries, Append Queries, Text Analytics, Vision, Azure Machine Learning, and Combine.
- Queries [4]** pane showing the query name `FactSales` and its dependencies: DimCustomers, DimProducts, and DimDates.
- Preview pane** showing the transformed data with columns: ORDERNUMBER, QUANTITYORDERED, PRICEEACH, ORDERLINENUMBER, and SALES.
- Applied Steps pane** listing steps: Source, Promoted Headers, Changed Type, Renamed Columns, Removed Columns, Removed Blank Rows, Replaced Value, Changed Type1, Removed Columns1, Changed Type2, Removed Duplicates, Capitalized Each Word, and Trimmed Text (highlighted).
- Query Settings pane** with Name set to `FactSales`.
- Bottom status bar: 20 COLUMNS, 307 ROWS, Column profiling based on top 1000 rows, PREVIEW DOWNLOADED AT 13:39.

Dimension Tables:

DimCustomers: Extract unique Customer_ID, Customer_Name, Region.

Untitled - Power Query Editor

File Home Transform Add Column View Tools Help

Queries [3]

FactSales DimCustomers DimProducts

CustomerName CONTACTLASTNAME CONTACTFIRSTNAME

	A ₁ C CUSTOMERNAME	A ₁ C CONTACTLASTNAME	A ₁ C CONTACTFIRSTNAME
1	Land Of Toys Inc.	Yu	Kwai
2	Reims Collectables	Henriot	Paul
3	Lyon Souveniers	Da Cunha	Daniel
4	Toys4Grownups.Com	Young	Julie
5	Corporate Gift Ideas Co.	Brown	Julie
6	Technics Stores Inc.	Hirano	Juri
7	Daedalus Designs Imports	Rance	Martine
8	Herku Gifts	Oeztan	Veysel
9	Mini Wheels Co.	Murphy	Julie
10	Auto Canal Petit	Perrier	Dominique
11	Australian Collectors, Co.	Ferguson	Peter
12	Vitachrome Inc.	Frick	Michael
13	Tekni Collectables Inc.	Brown	William
14	Gift Depot Inc.	King	Julie
15	La Rochelle Gifts	Labrune	Janine
16	Marta'S Replicas Co.	Hernandez	Marta
17	Toys Of Finland, Co.	Karttunen	Matti
18	Baane Mini Imports	Bergulsen	Jonas
19	Diecast Classics Inc.	Yu	Kyung
20	Land Of Toys Inc.	Yu	Kwai
21	Salzburg Collectables	Pipps	Georg

3 COLUMNS, 307 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 13:44

DimProducts: Extract unique Product_Code, Product_Name, Category.

Untitled - Power Query Editor

File Home Transform Add Column View Tools Help

Queries [3]

FactSales DimCustomers DimProducts

ORDERNUMBER QUANTITYORDERED PRICEEACH A₁C STATUS QTR_ID

	i ₂ 3 ORDERNUMBER	\$ QUANTITYORDERED	1.2 PRICEEACH	A ₁ C STATUS	i ₂ 3 QTR_ID
1	10107	30.00	95.7	Shipped	
2	10121	34.00	81.35	Shipped	
3	10134	41.00	94.74	Shipped	
4	10145	45.00	83.26	Shipped	
5	10159	49.00	100	Shipped	
6	10168	36.00	96.66	Shipped	
7	10180	29.00	86.13	Shipped	
8	10188	48.00	100	Shipped	
9	10201	22.00	98.57	Shipped	
10	10211	41.00	100	Shipped	
11	10223	37.00	100	Shipped	
12	10237	23.00	100	Shipped	
13	10251	28.00	100	Shipped	
14	10263	34.00	100	Shipped	
15	10275	45.00	92.83	Shipped	
16	10285	36.00	100	Shipped	
17	10299	23.00	100	Shipped	
18	10309	41.00	100	Shipped	
19	10318	46.00	94.74	Shipped	
20	10329	42.00	100	Shipped	
21					

18 COLUMNS, 307 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 13:45

DimDates: Create a Date Table if not present (Order_Date, Year, Month, Quarter).

Untitled - Power Query Editor

Queries [4]

DimDates

DimCustomers

DimProducts

DimDates

= Table.TransformColumns(#"Capitalized Each Word", {"STATUS", Text.Trim, type text})

	MONTH_ID	YEAR_ID	PRODUCTLINE	MSRP	PRODUCTCODE
1	1	2	2003 Motorcycles	95 \$10_1678	
2	2	5	2003 Motorcycles	95 \$10_1678	
3	3	7	2003 Motorcycles	95 \$10_1678	
4	3	8	2003 Motorcycles	95 \$10_1678	
5	4	10	2003 Motorcycles	95 \$10_1678	
6	4	10	2003 Motorcycles	95 \$10_1678	
7	4	11	2003 Motorcycles	95 \$10_1678	
8	4	11	2003 Motorcycles	95 \$10_1678	
9	4	12	2003 Motorcycles	95 \$10_1678	
10	1	1	2004 Motorcycles	95 \$10_1678	
11	1	2	2004 Motorcycles	95 \$10_1678	
12	2	4	2004 Motorcycles	95 \$10_1678	
13	2	5	2004 Motorcycles	95 \$10_1678	
14	2	6	2004 Motorcycles	95 \$10_1678	
15	3	7	2004 Motorcycles	95 \$10_1678	
16	3	8	2004 Motorcycles	95 \$10_1678	
17	3	9	2004 Motorcycles	95 \$10_1678	
18	4	10	2004 Motorcycles	95 \$10_1678	
19	4	11	2004 Motorcycles	95 \$10_1678	
20	4	11	2004 Motorcycles	95 \$10_1678	
21					

20 COLUMNS, 307 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 13:50

To create a Date Table in Power Query:

Go to New Source > Blank Query.

Open Advanced Editor and paste:

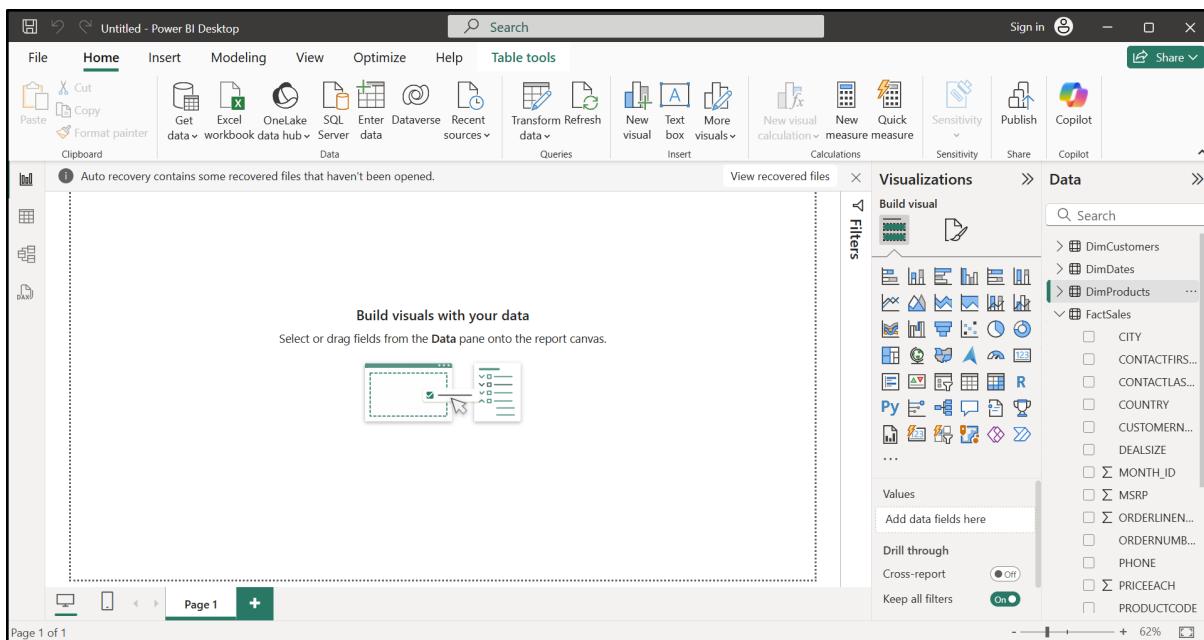
let

```
StartDate = #date(2020, 1, 1),
EndDate = #date(2030, 12, 31),
DateList = List.Dates(StartDate, Number.From(EndDate - StartDate) + 1, #duration(1, 0, 0, 0)),
DateTable = Table.FromList(DateList, Splitter.SplitByNothing(), {"Date"}),
ChangedType = Table.TransformColumnTypes(DateTable, {"Date", type date})
```

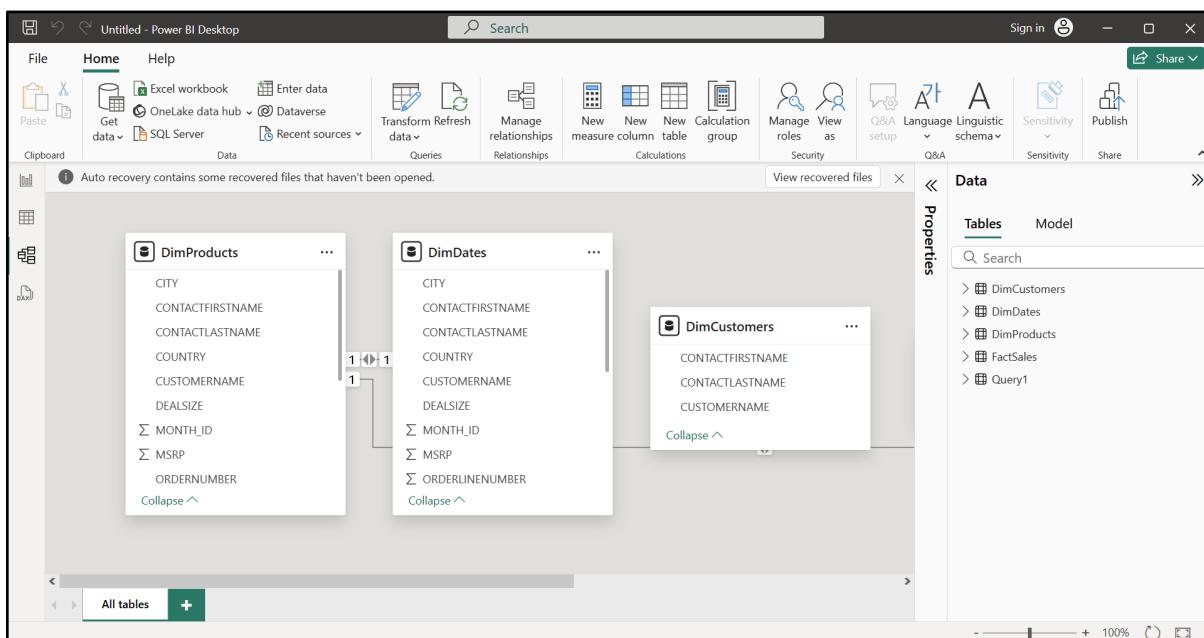
in

ChangedType

Click Close & Apply.



4. Define Relationships in Power BI Model



Go to Model View and establish relationships:

FactSales[Customer_ID] → DimCustomers[Customer_ID]
 FactSales[Product_Code] → DimProducts[Product_Code]
 FactSales[Order_Date] → DimDates[Date]

5. Create Data Validation Reports

Use DAX Measures to check data quality:

Total Sales:

$\text{Total_Sales} = \text{SUM}(\text{FactSales}[\text{Total_Amount}])$

Count of Missing Customers:

$\text{Missing_Customers} = \text{COUNTROWS}(\text{FILTER}(\text{FactSales}, \text{ISBLANK}(\text{FactSales}[\text{Customer_ID}])))$

Auto recovery contains some recovered files that haven't been opened.

View recovered files 

```
1 Missing_Customers = COUNTROWS(FILTER(FactSales, ISBLANK(FactSales[Customer_ID])))
```

Duplicate Orders Check:

Duplicate_Orders = COUNTROWS(FactSales) - DISTINCTCOUNT(FactSales[Order_ID])

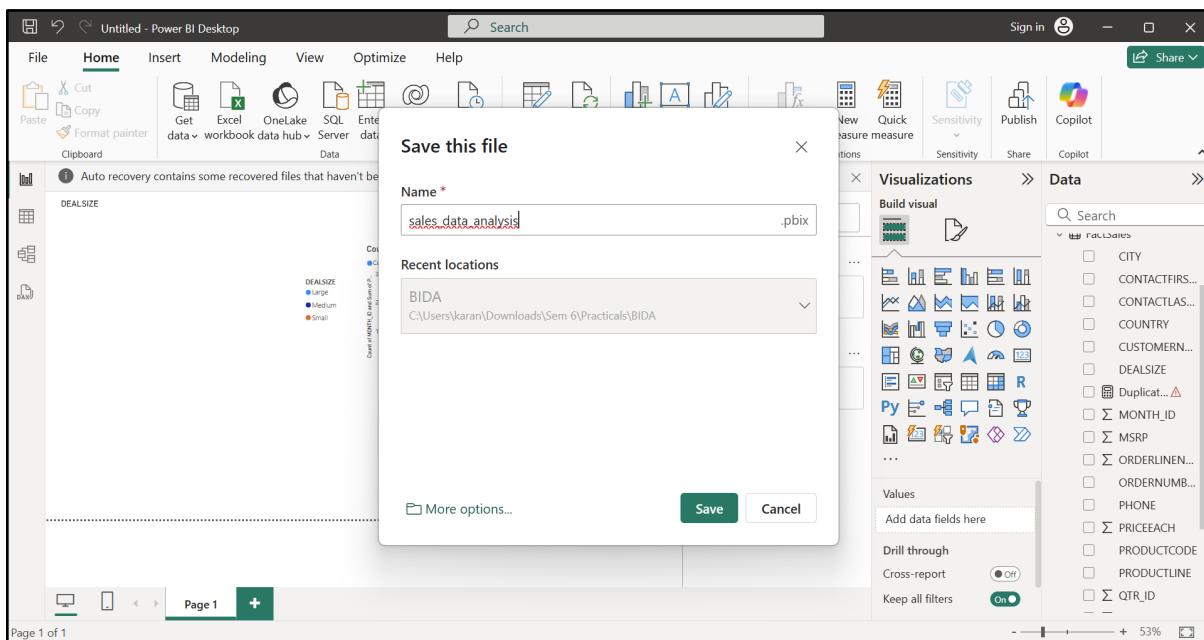
Auto recovery contains some recovered files that haven't been opened.

View recovered files 

```
1 Duplicate_Orders = COUNTROWS(FactSales) - DISTINCTCOUNT(FactSales[Order_ID])
```

Create a Table Visual for missing data analysis.

6. Save & Publish

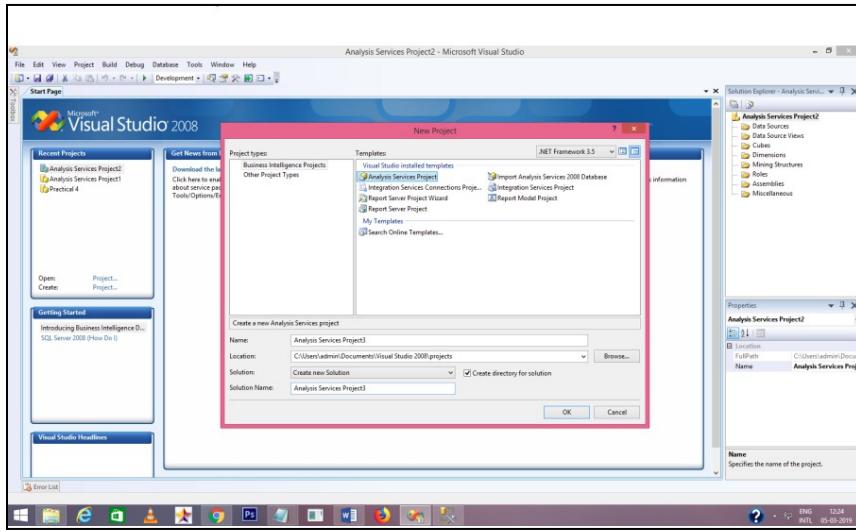


Click Close & Apply to load the transformed data into Power BI.

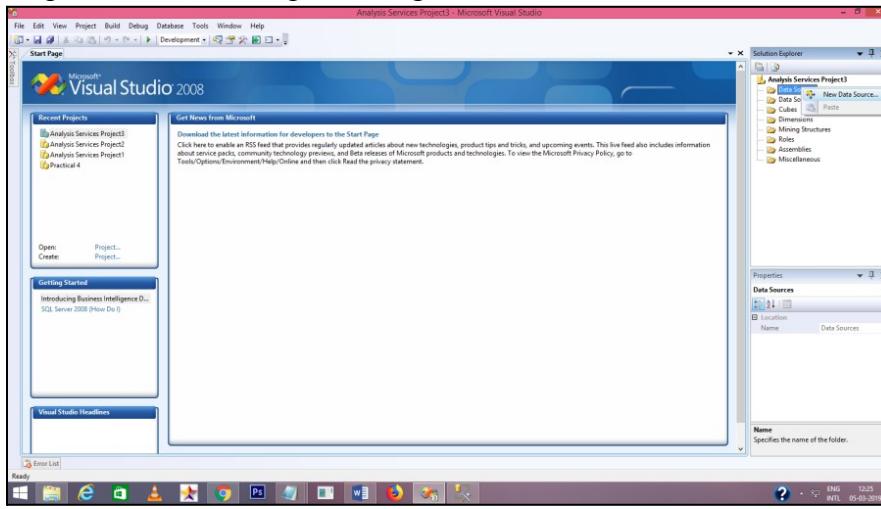
Practical 10

Create the cube with suitable dimension and fact tables based on ROLAP, MOLAP and HOLAP model.

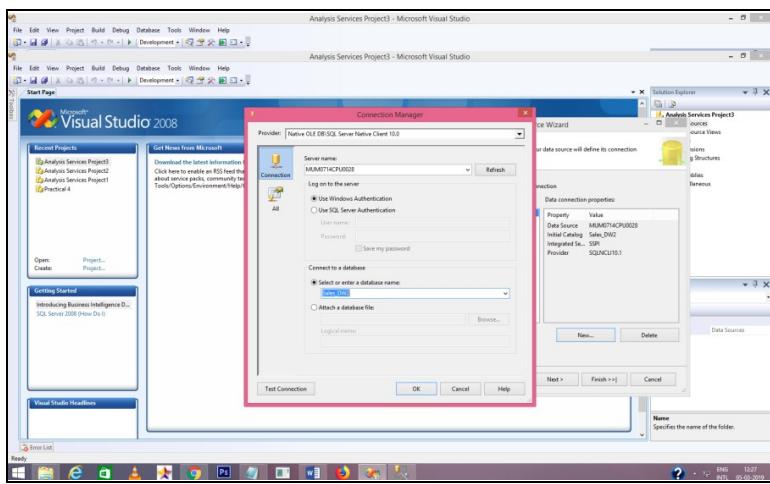
Step 1. Click File -> New -> Project -> Business Intelligence Projects -> select Analysis Services Project -> Assign Project Name -> Click OK.



Step 2. In Solution Explorer, Right click on Data Source -> Click New Data Source

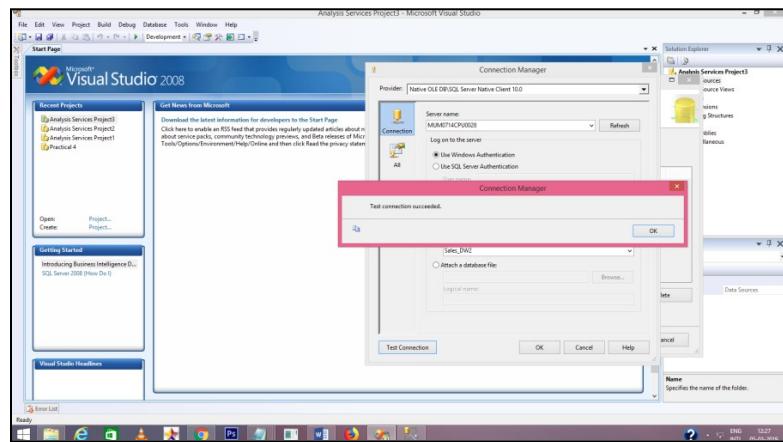


Step 3. Click on New

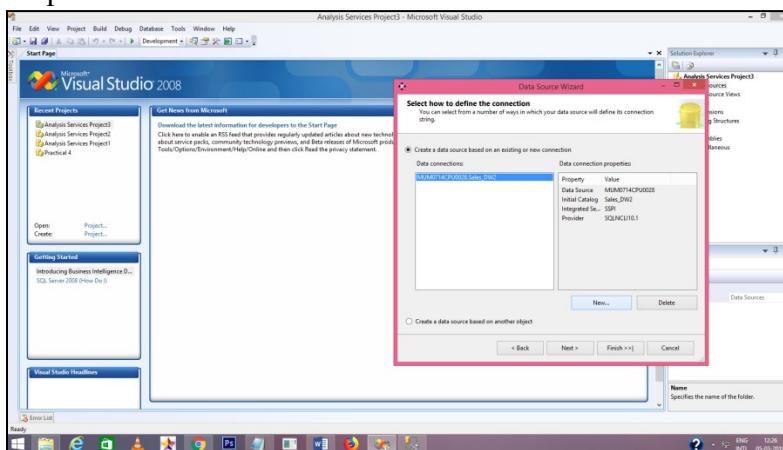


Step 4. Creating New Connection.

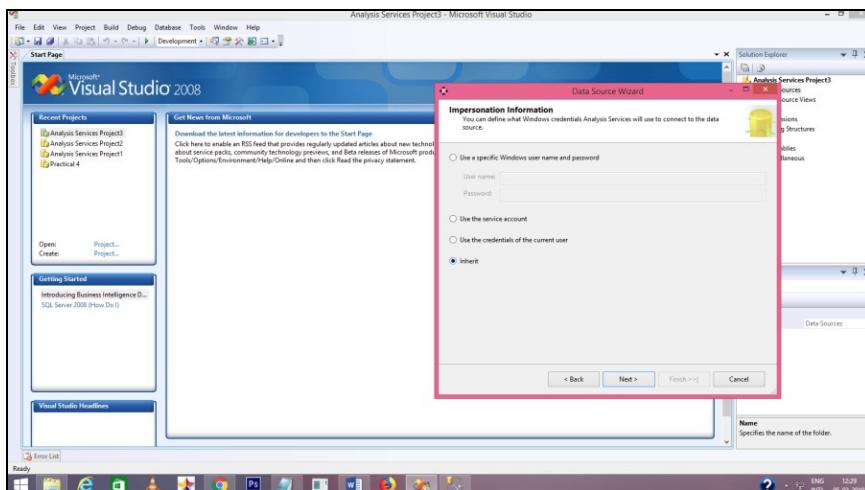
Step 5. Click on Test Connection and verify for its success.



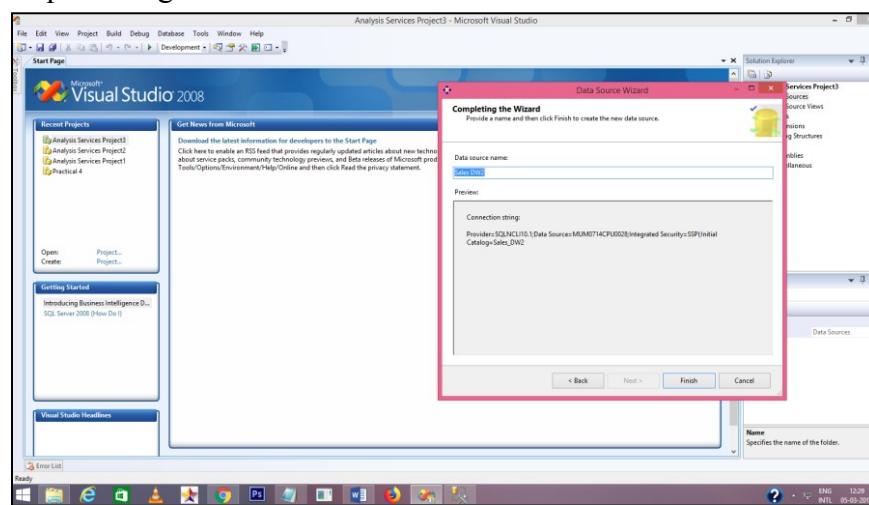
Step 6. Select Connection created in Data Connections-> Click Next.



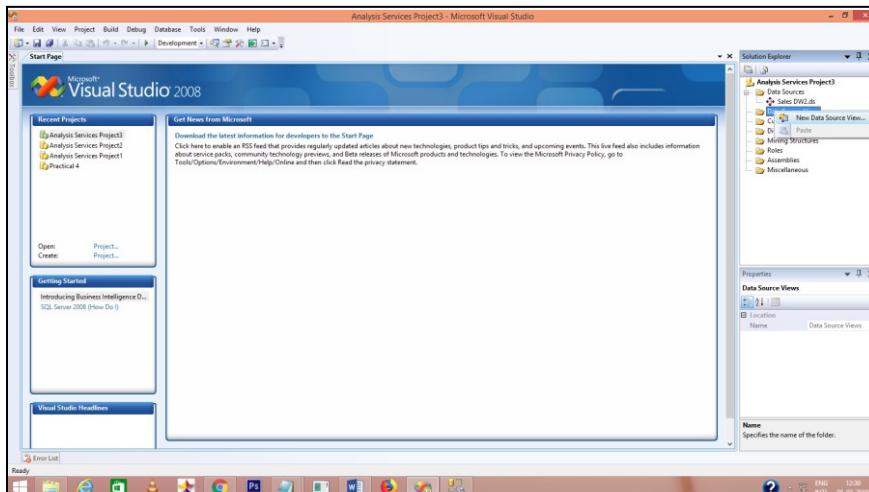
Step 7. Select Option Inherit.



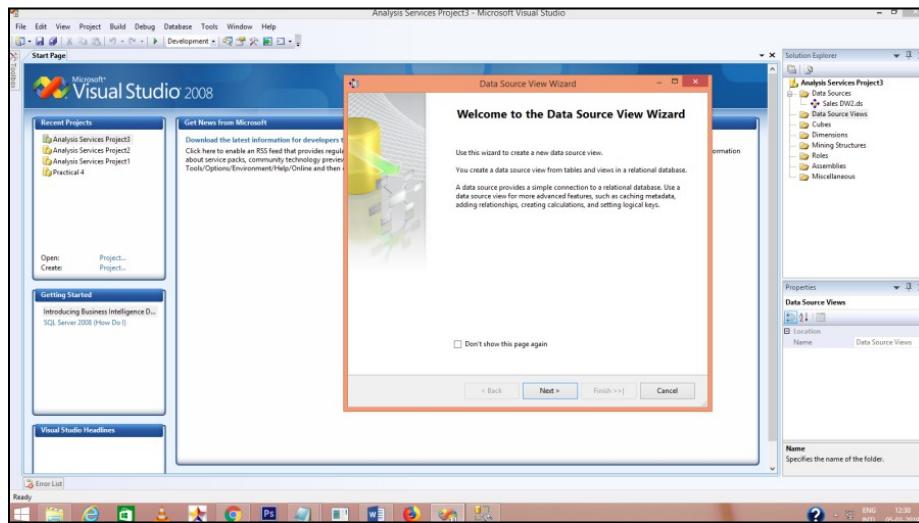
Step 8. Assign Data Source Name -> Click Finish



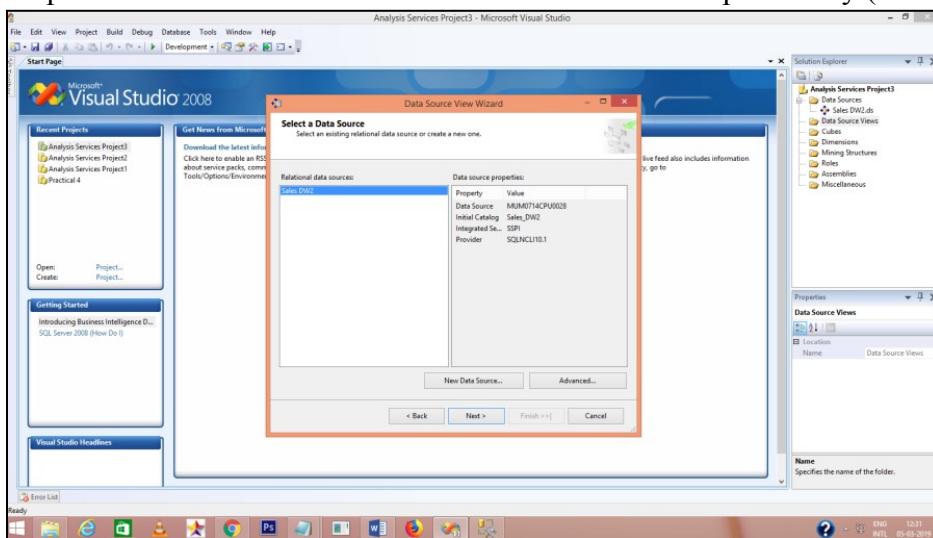
Step 9. In the Solution Explorer, Right Click on Data Source View -> Click on New Data Source View.



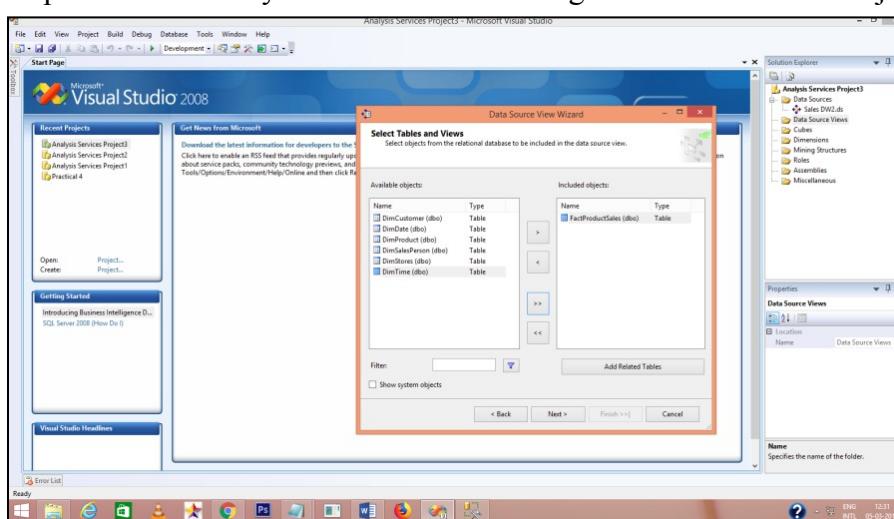
Step 10. Click Next.



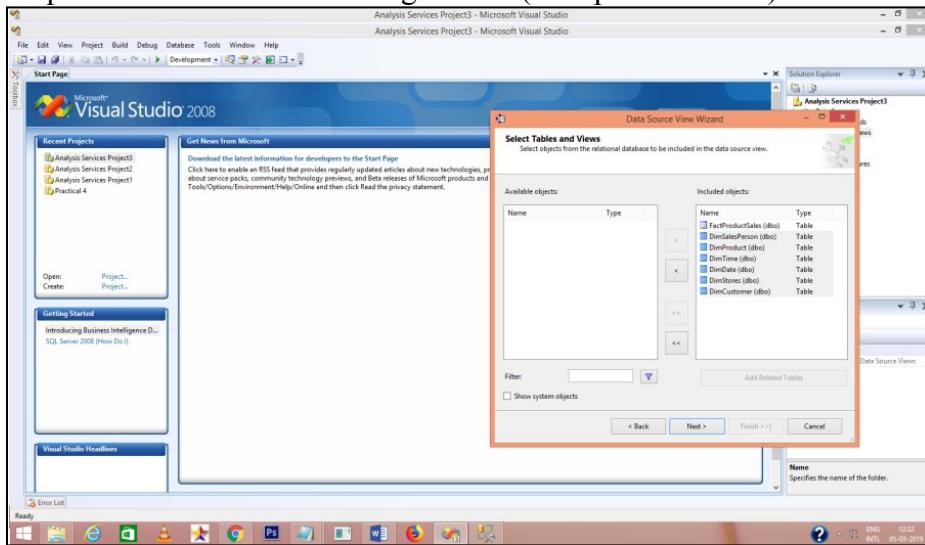
Step 11. Select Relational Data Source we have created previously (Sales DW)-> Click Next.



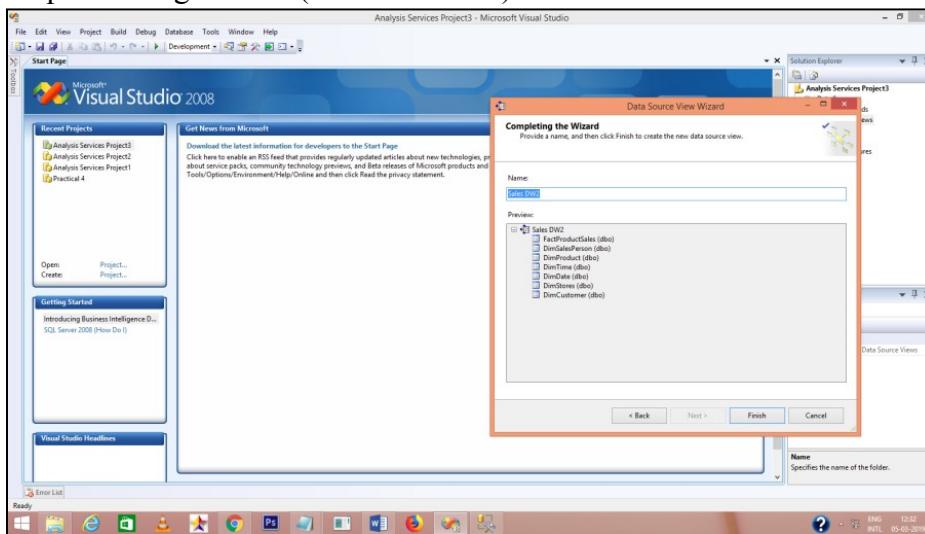
Step 12. First move your Fact Table to the right side to include in object list.



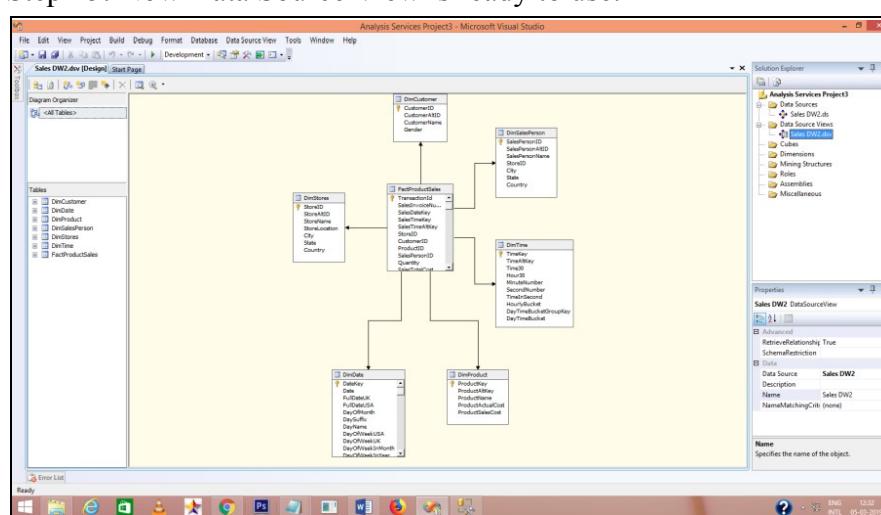
Step 13. Select Fact Table in Right Pane (Fact product Sales) -> Click On Add Related Tables.



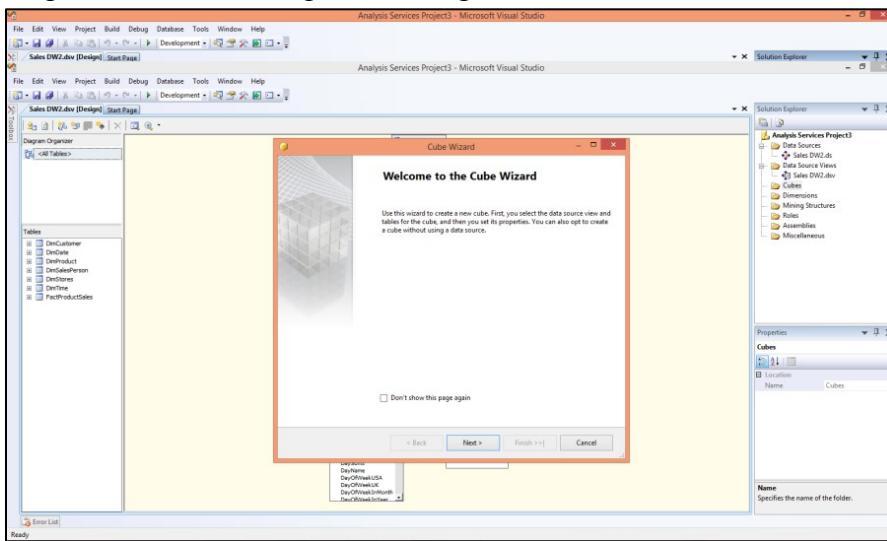
Step 14. Assign Name (SalesDW DSV)-> Click Finish



Step 15. Now Data Source View is ready to use.

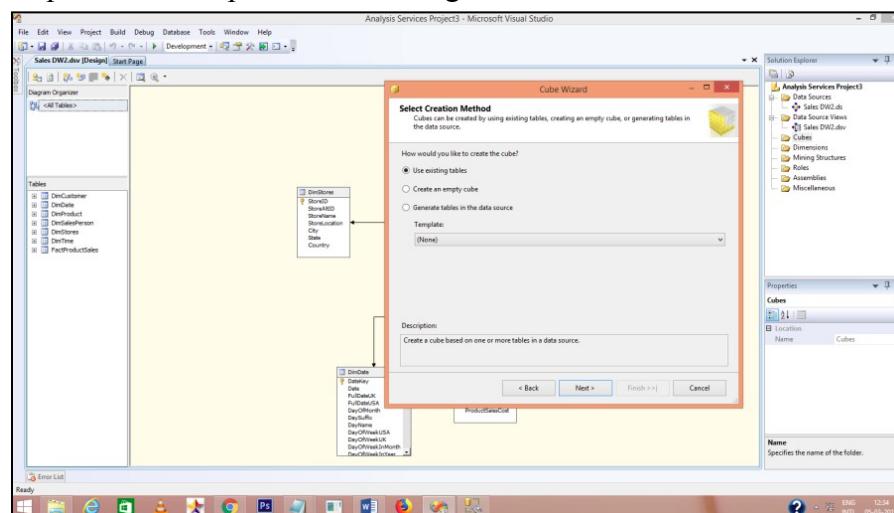


Step 16. In Solution Explorer -> Right Click on Cube-> Click New Cube.

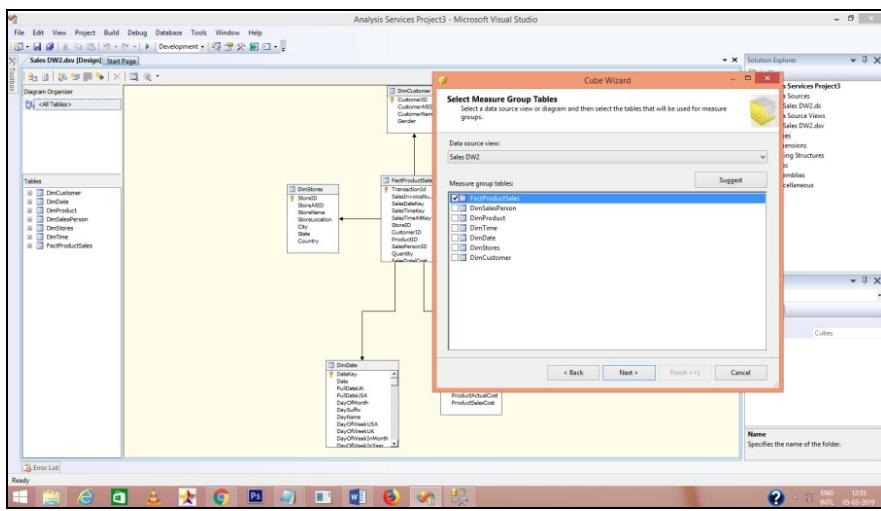


Step 17. Click Next.

Step 18. Select Option Use existing Tables -> Click Next.

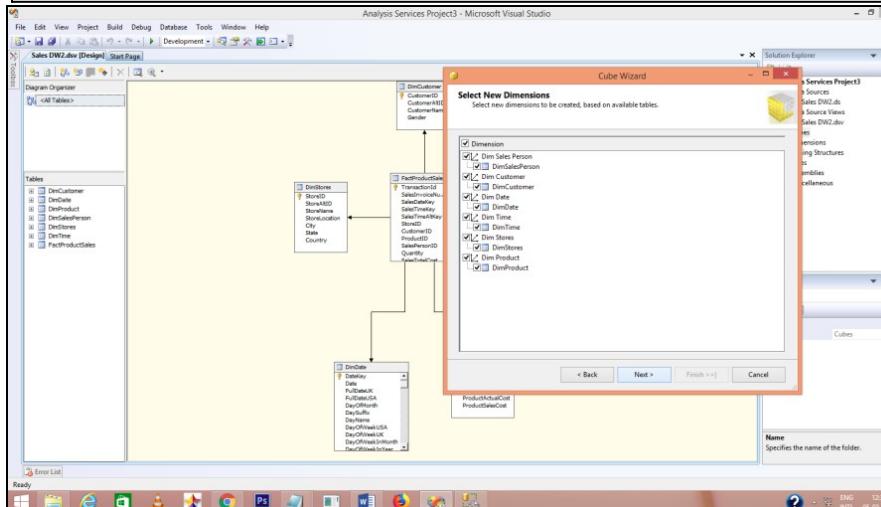
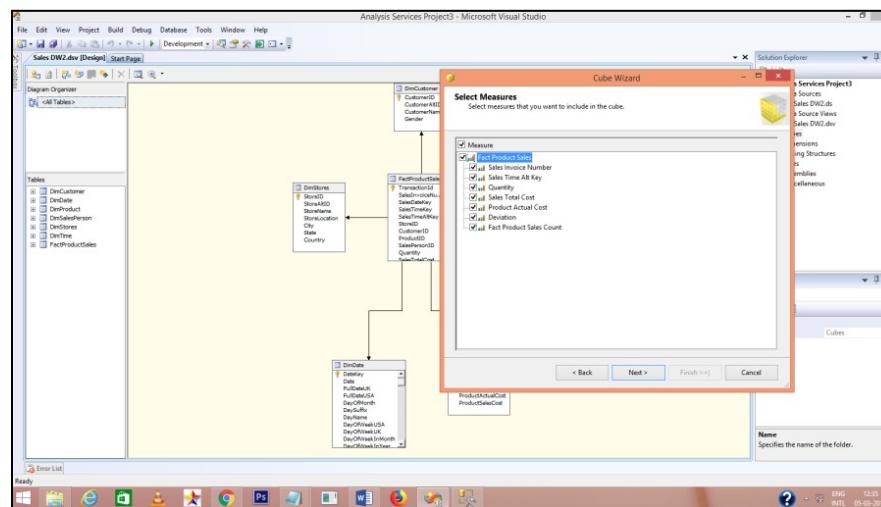


Step 19. Select Fact Table Name from Measure Group Tables (FactProductSales) -> Click Next.

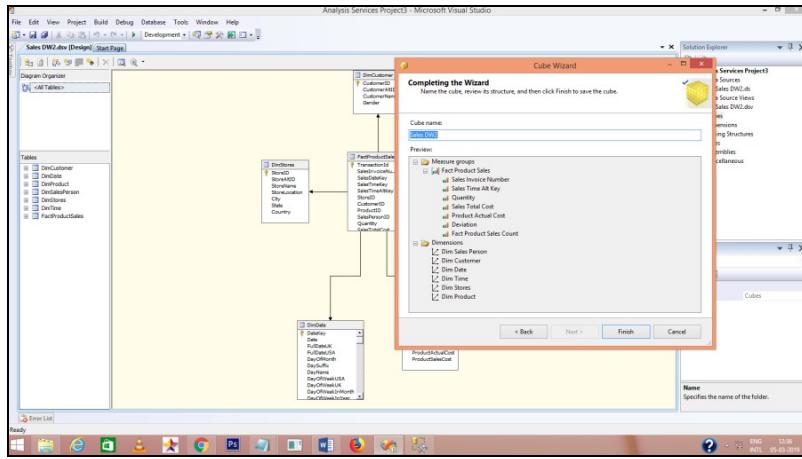


Step 20. Choose Measures from the List which you want to place in your Cube --> Click Next.

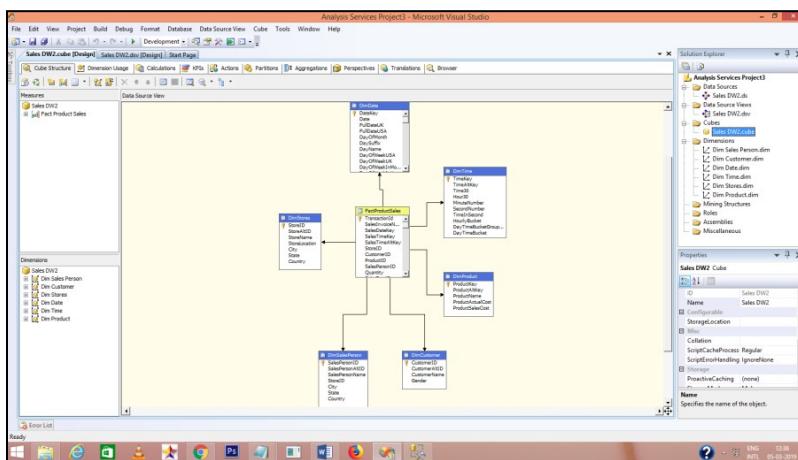
Step 21. Select All Dimensions here which are associated with your Fact Table-> Click Next



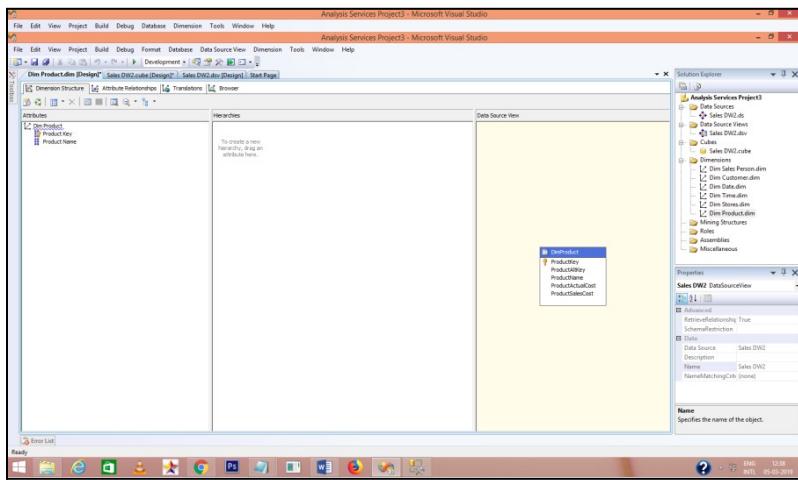
Step 22. Assign Cube Name (SalesDW2) -> Click Finish.



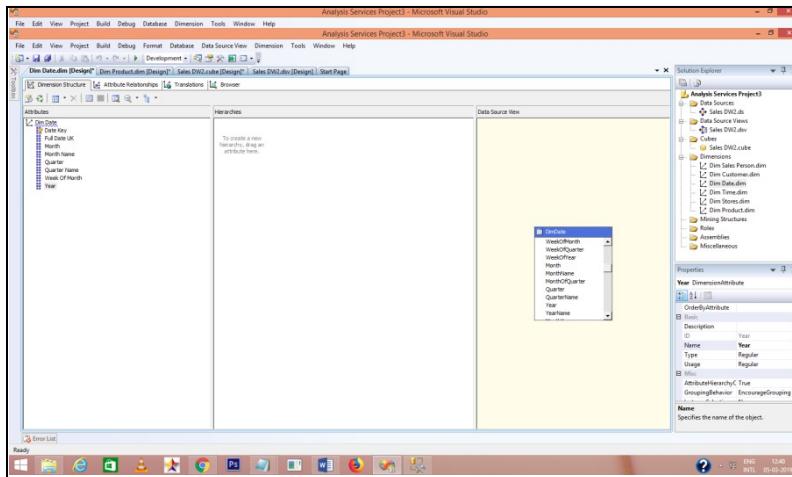
Step 23. Now your Cube is ready, you can see the newly created cube and dimensions added in your solution explorer.



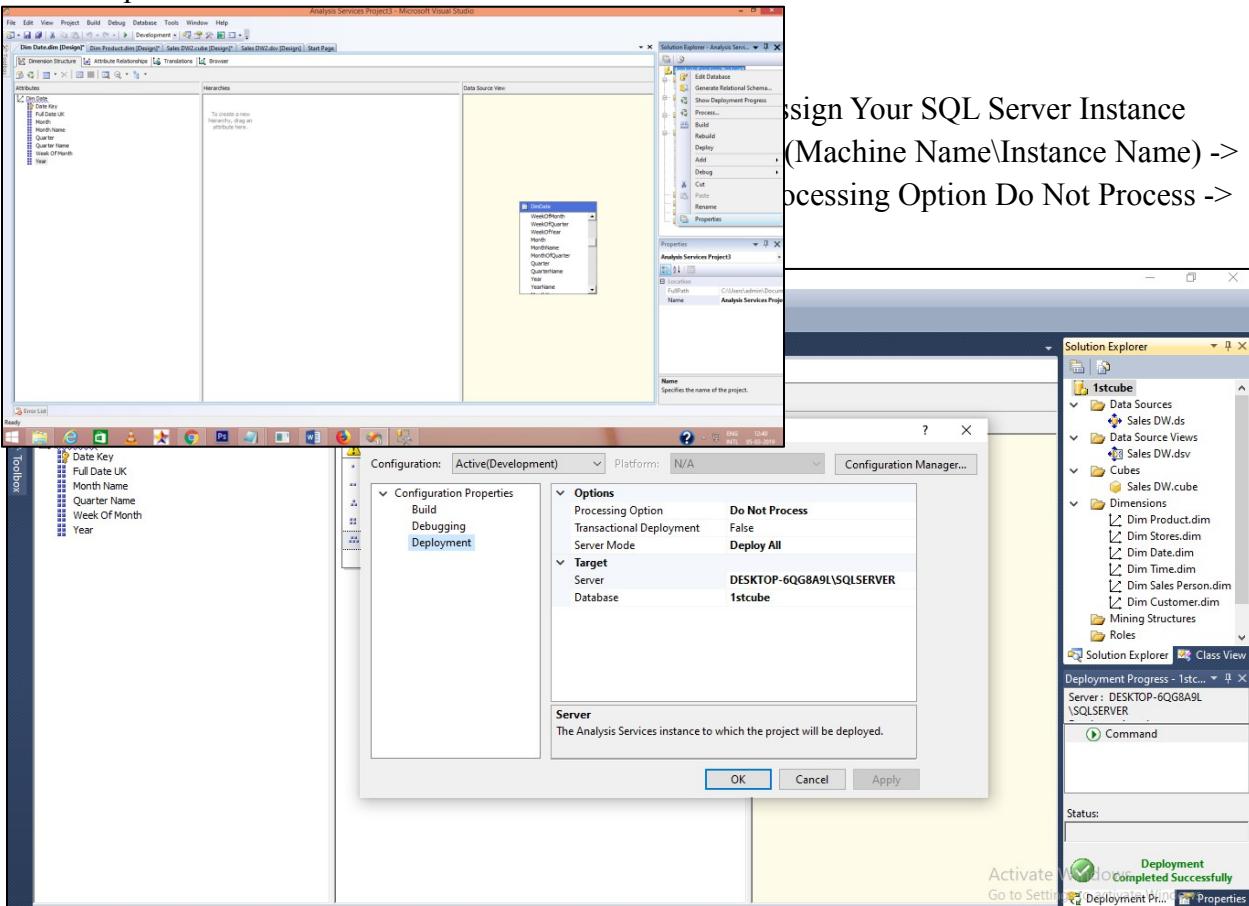
Step 24. In Solution Explorer, double click on dimension Dim Product -> Drag and Drop Product Name from Table in Data Source View and Add in Attribute Pane at left side.



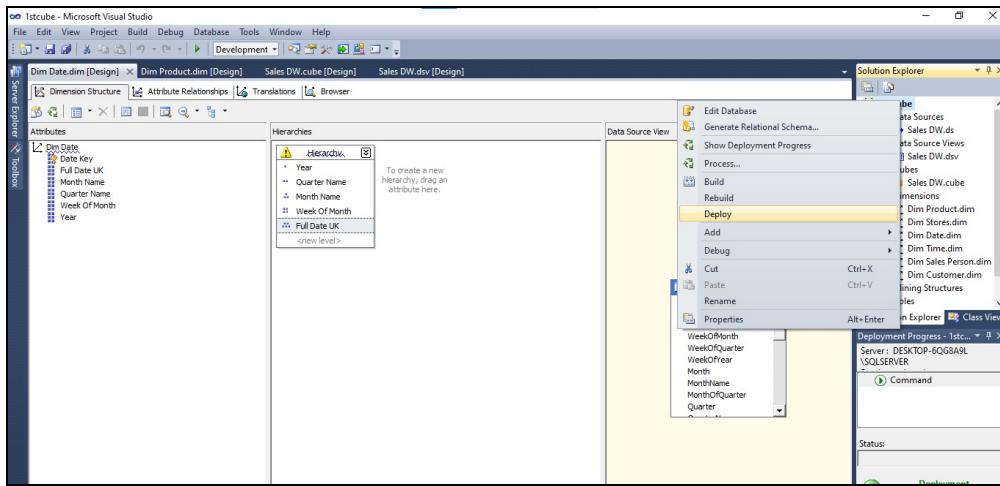
Step 25. Double click On Dim Date dimension -> Drag and Drop Fields from Table shown in Data Source View to Attributes-> Drag and Drop attributes from leftmost pane of attributes to middle pane of Hierarchy.



Step 26. In Solution Explorer, right click on Project Name (Analysis Services Project3) Click Properties.

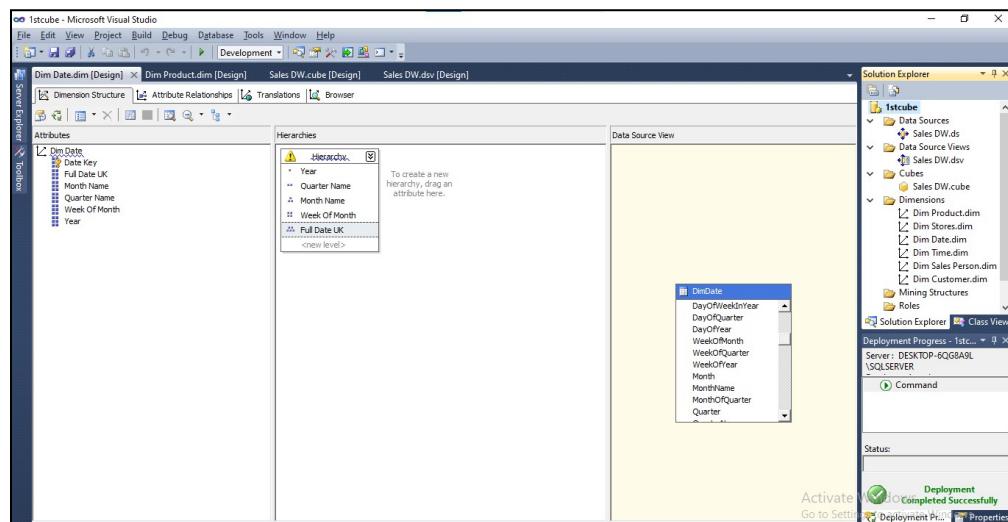


Step 28. In Solution Explorer, right click on Project Name (AnalysisServicesProject)

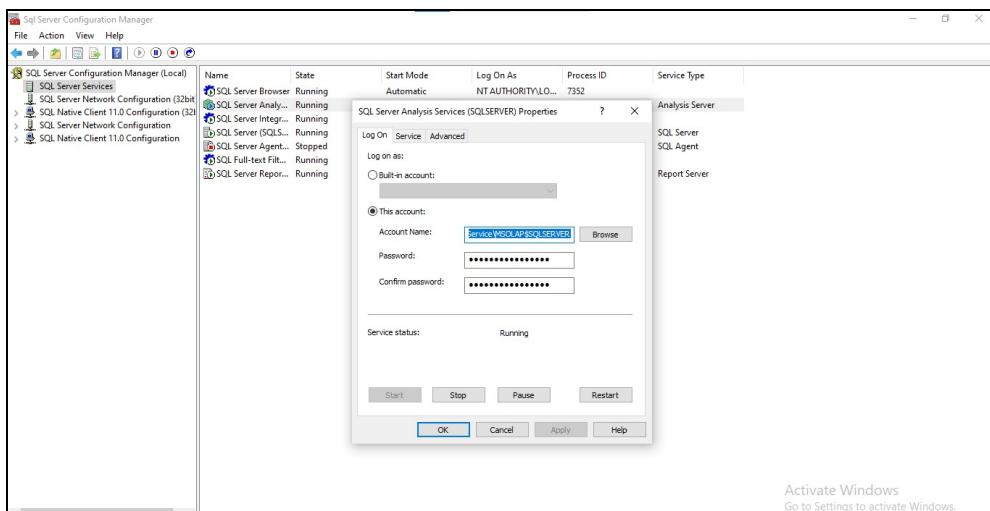


Click Deploy.

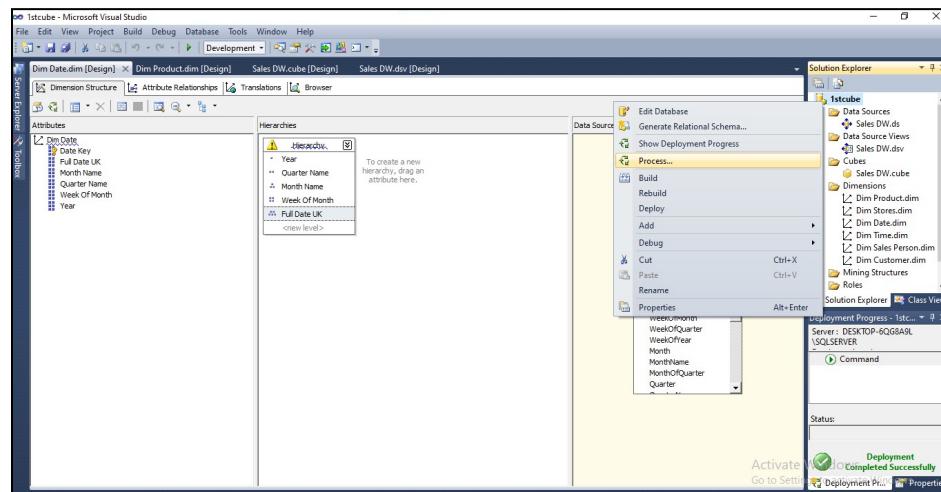
Step 29. Once Deployment will finish, you can see the message Deployment Completed in deployment Properties.



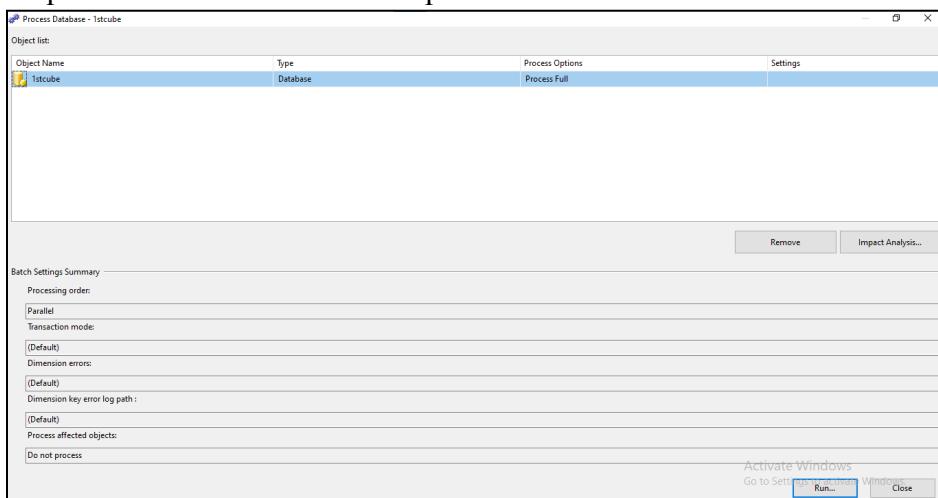
Step 30. Open SQL Server Configuration Server→Click on SQL Analysis Server→ Copy the account name and close the window.



Step 31. In Solution Explorer, right click on Project Name (AnalysisServicesProject3)
→Click Process.

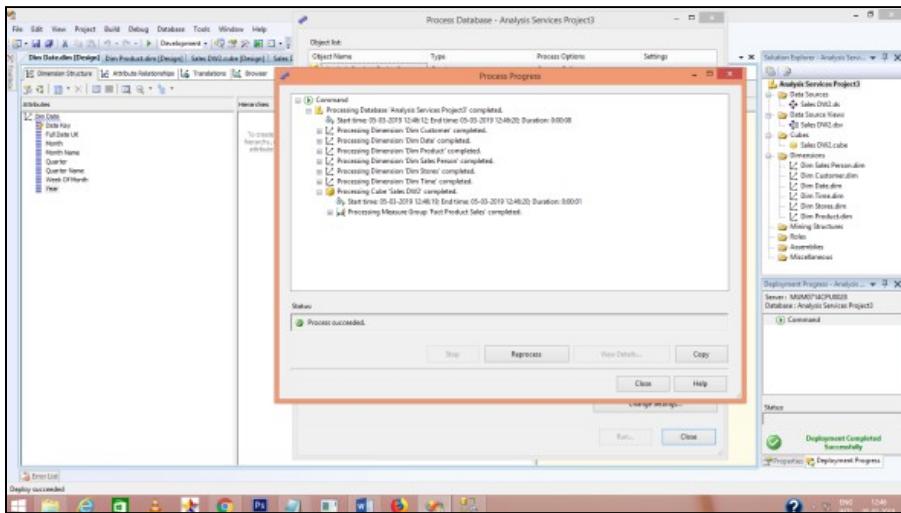


Step 32. Click on Run Button to process the 1stcube.

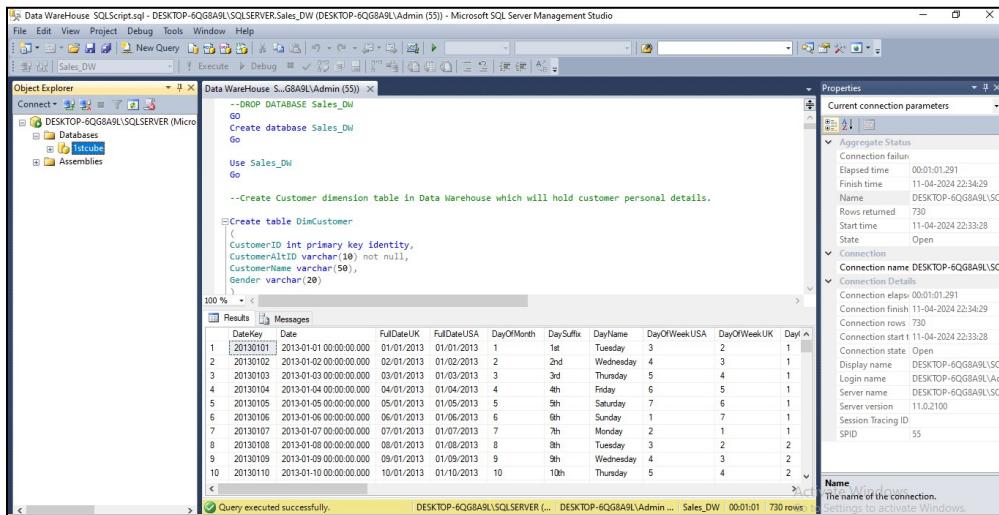


Step 33. Once processing is complete, you can see Status as Process Succeeded

-->Click Close to close both the open windows for processing one after the other.



Step 34. Now go to SQL Server Management Studio, disconnect the existing database and connect to Analysis Services→ in the database folder you will see the 1stcube has been successfully created.



The screenshot shows the Microsoft SQL Server Management Studio interface. A script named "Data Warehouse SQLScript.sql" is being run against the "Sales_DW" database. The script creates a database named "Sales_DW" and then uses it to create a "DimCustomer" dimension table. The table has columns for CustomerID (primary key), CustomerAltID, CustomerName, and Gender. The results pane displays 730 rows of data from the "DimCustomer" table, showing various dates and day names. The properties pane on the right shows connection details like connection name, elapsed time, and start time.

```

--DROP DATABASE Sales_DW
GO
Create database Sales_DW
Go

Use Sales_DW
Go

--Create Customer dimension table in Data Warehouse which will hold customer personal details.

Create table DimCustomer
(
    CustomerID int primary key identity,
    CustomerAltID varchar(10) not null,
    CustomerName varchar(50),
    Gender varchar(20)
)

```

DateKey	Date	FullDateUK	FullDateUSA	DayOfMonth	DaySuffix	DayName	DayOfWeekUSA	DayOfWeekUK	Day
1	20130101	2013-01-01 00:00:00.000	01/01/2013	1	1st	Tuesday	3	2	1
2	20130102	2013-01-02 00:00:00.000	02/01/2013	2	2nd	Wednesday	4	3	1
3	20130103	2013-01-03 00:00:00.000	03/01/2013	3	3rd	Thursday	5	4	1
4	20130104	2013-01-04 00:00:00.000	04/01/2013	4	4th	Friday	6	5	1
5	20130105	2013-01-05 00:00:00.000	05/01/2013	5	5th	Saturday	7	6	1
6	20130106	2013-01-06 00:00:00.000	06/01/2013	6	6th	Sunday	1	7	1
7	20130107	2013-01-07 00:00:00.000	07/01/2013	7	7th	Monday	2	1	1
8	20130108	2013-01-08 00:00:00.000	08/01/2013	8	8th	Tuesday	3	2	2
9	20130109	2013-01-09 00:00:00.000	09/01/2013	9	9th	Wednesday	4	3	2
10	20130110	2013-01-10 00:00:00.000	10/01/2013	10	10th	Thursday	5	4	2