Practical no.1

Aim: Perform the analysis for the following:

1.A) Import the data warehouse data in Microsoft Excel and create the Pivot table and Pivot Chart.

Let us execute our T-SQL Script to create data warehouse with fact tables, dimensions and populate them with appropriate test values.

Download T-SQL script attached with this article for creation of Sales Data Warehouse or download from this article "**Create First Data Warehouse**" and run it in your SQL Server.

Follow the given steps to run the query in **SSMS** (SQL Server Management Studio).

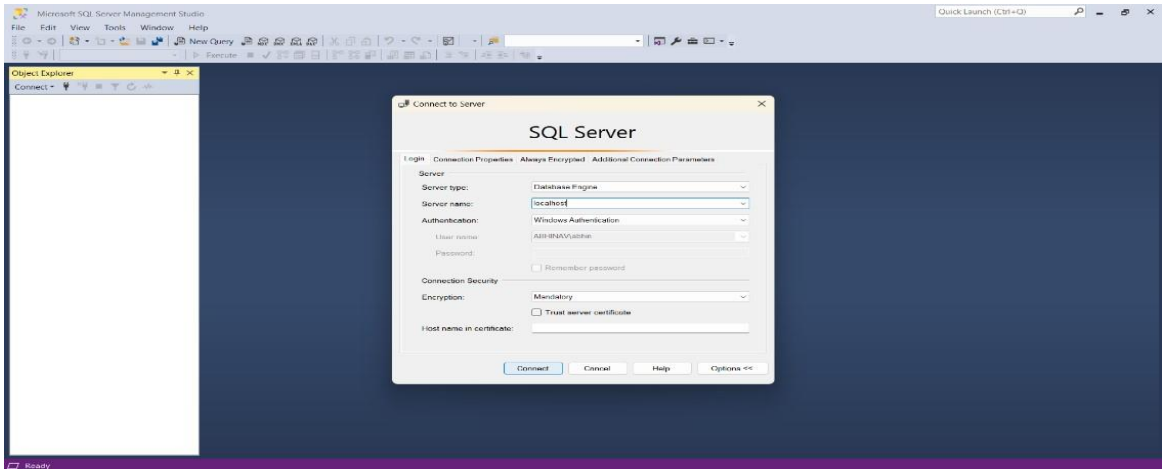
1. Open SQL Server Management Studio 2008
2. Connect Database Engine
3. Open **New Query** editor
4. Copy paste Scripts given below in various steps in new query editor window one by one
5. To run the given SQL Script, press **F5**
6. It will create and populate "**Sales_DW**" database on your SQL Server

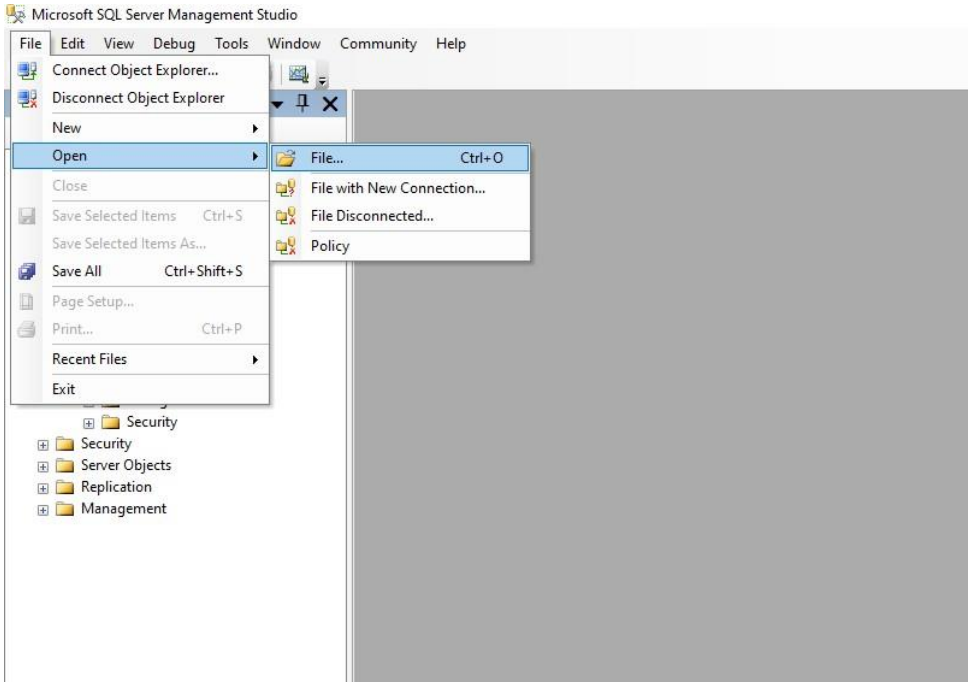
Steps:-

1.To Create a first datawarehouse -> search browser and download it

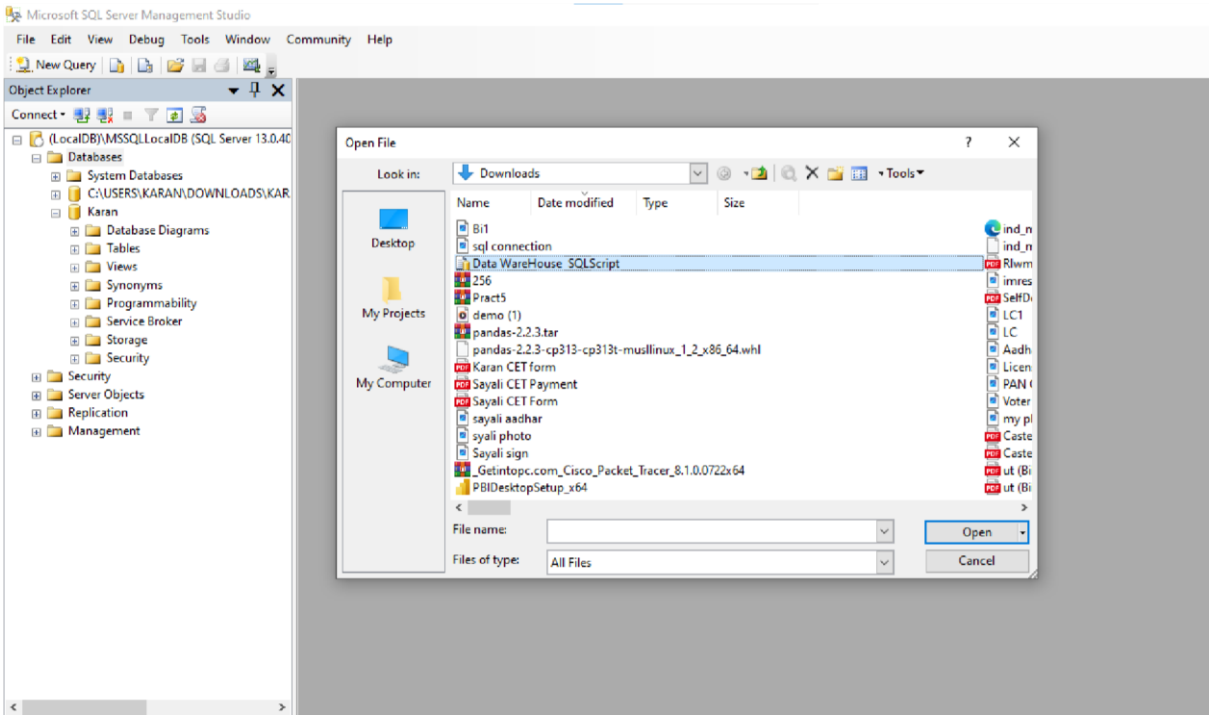
2.Open Microsoft SQL Server 2008 R2

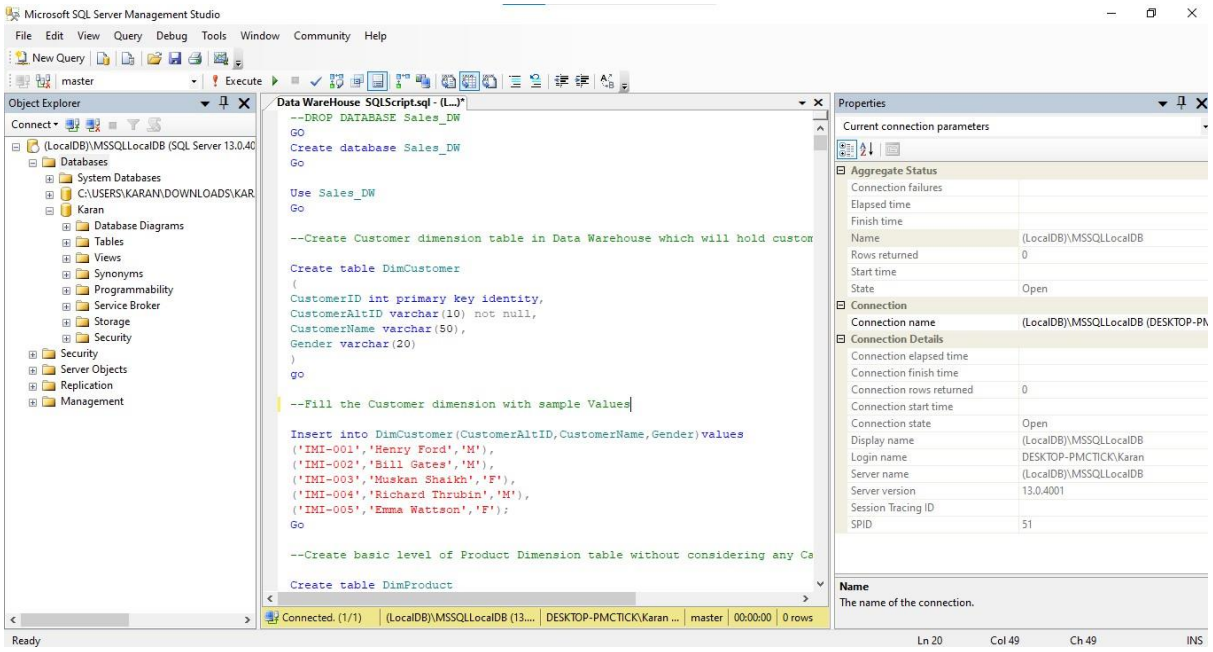
3.Server Name: localhost -> connect



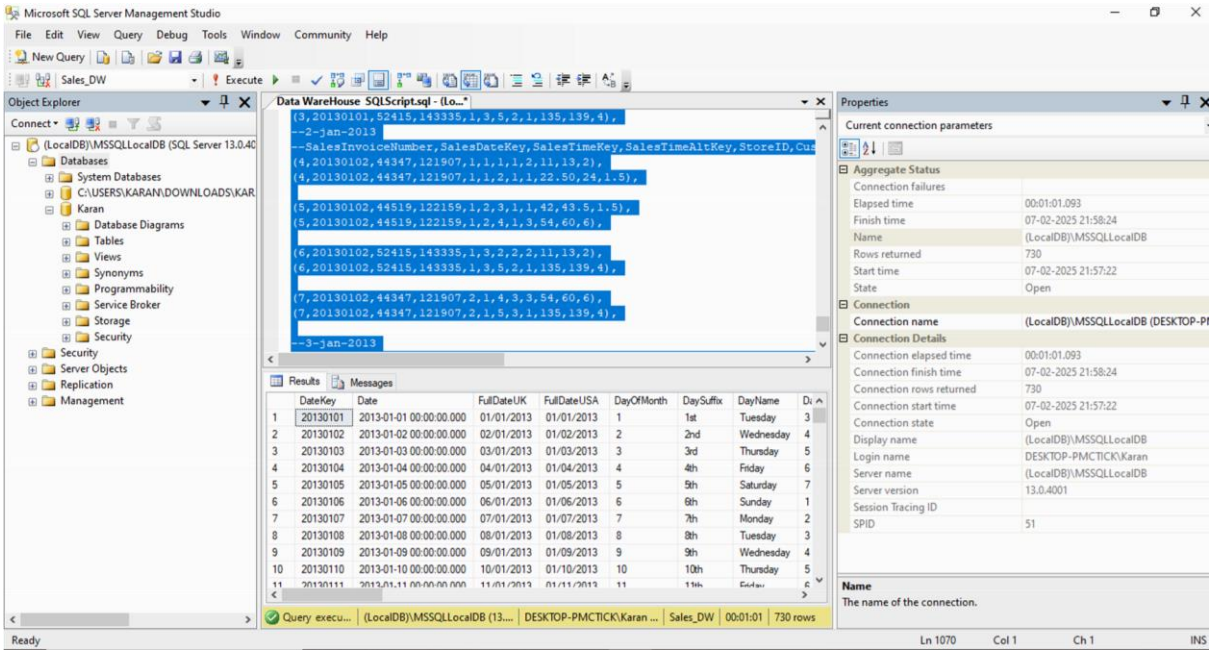


4. Now go to FILE option and Open -> select Datawarehouse file download



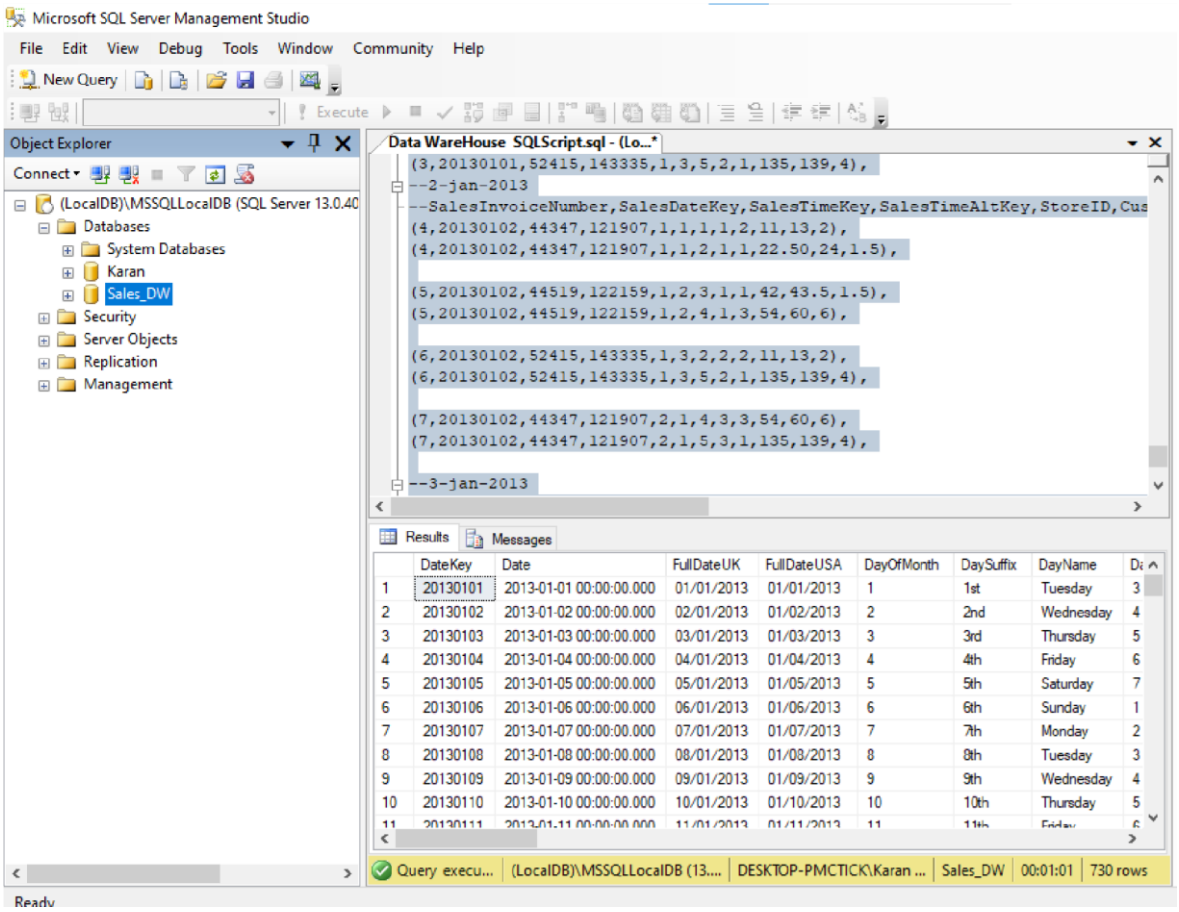


5.Select all data -> Cltr +A and Press F5

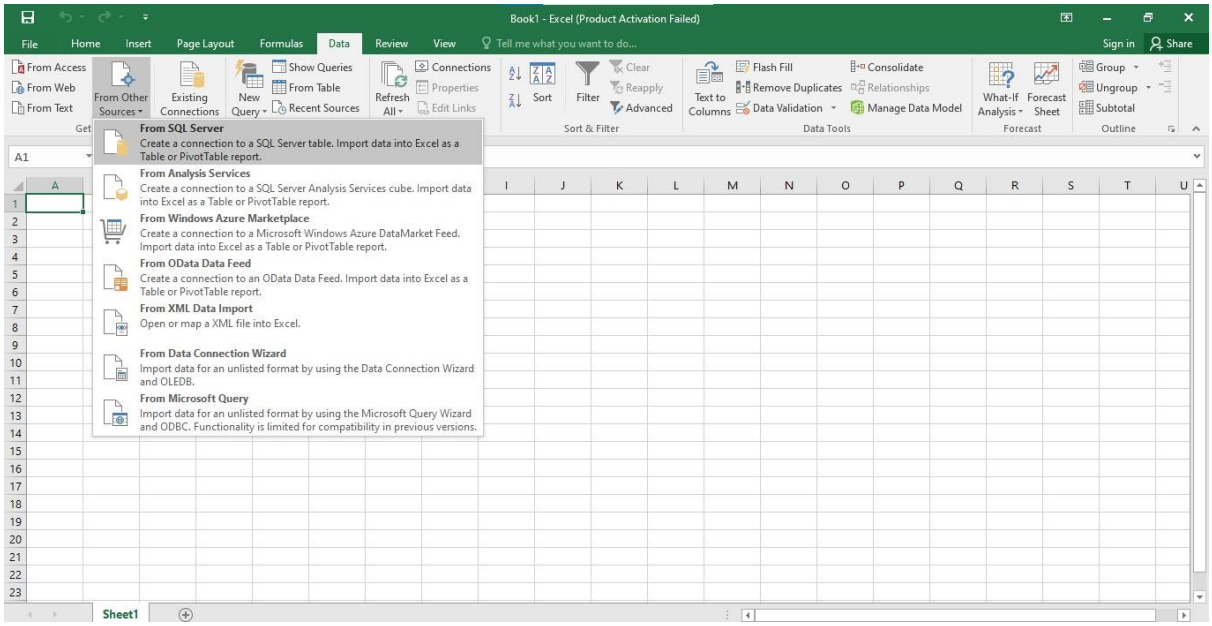


6. refresh and the display is folder created name is sale_DW

7.it database create successful

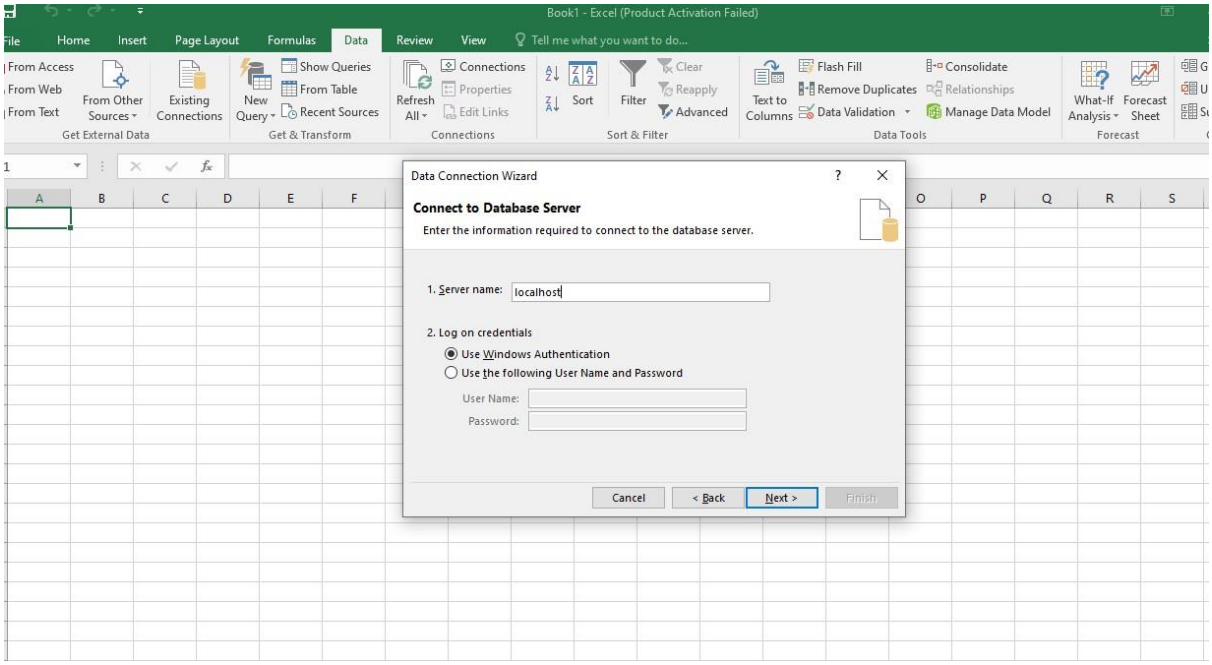


8. Open Microsoft Excel -> Data Option -> From Other Source -> Select From SQL Server



9. Server Name: localhost -> next -> dropdown select sale_dw -> tick Dim Product

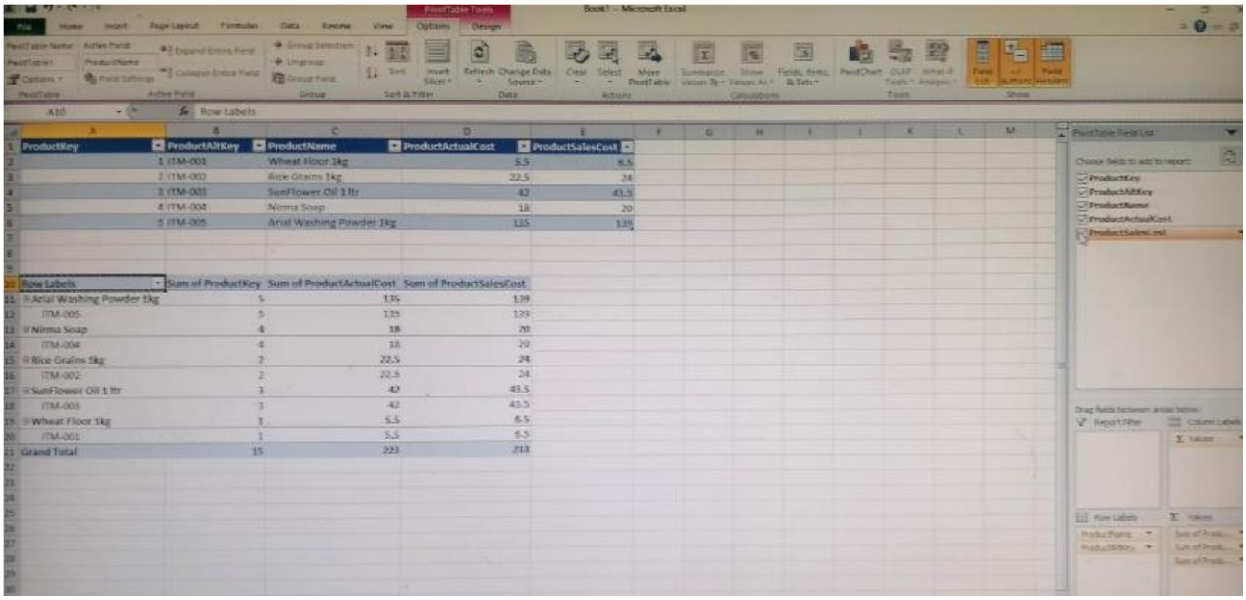
10. next-> finish-> ok



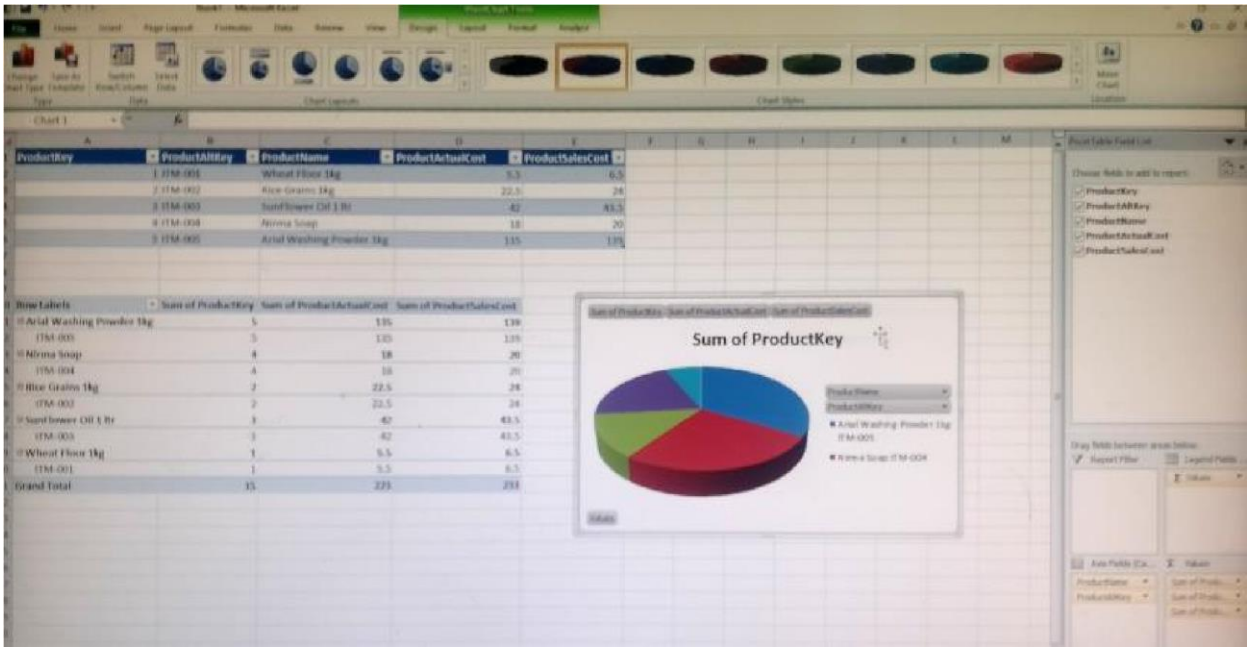
11.it create a table

- Select Table -> Insert -> Pivot Table -> Existing WorkingSheet ➲ Location Select any cell -> OK

- Select All Fields from Right Side Field List

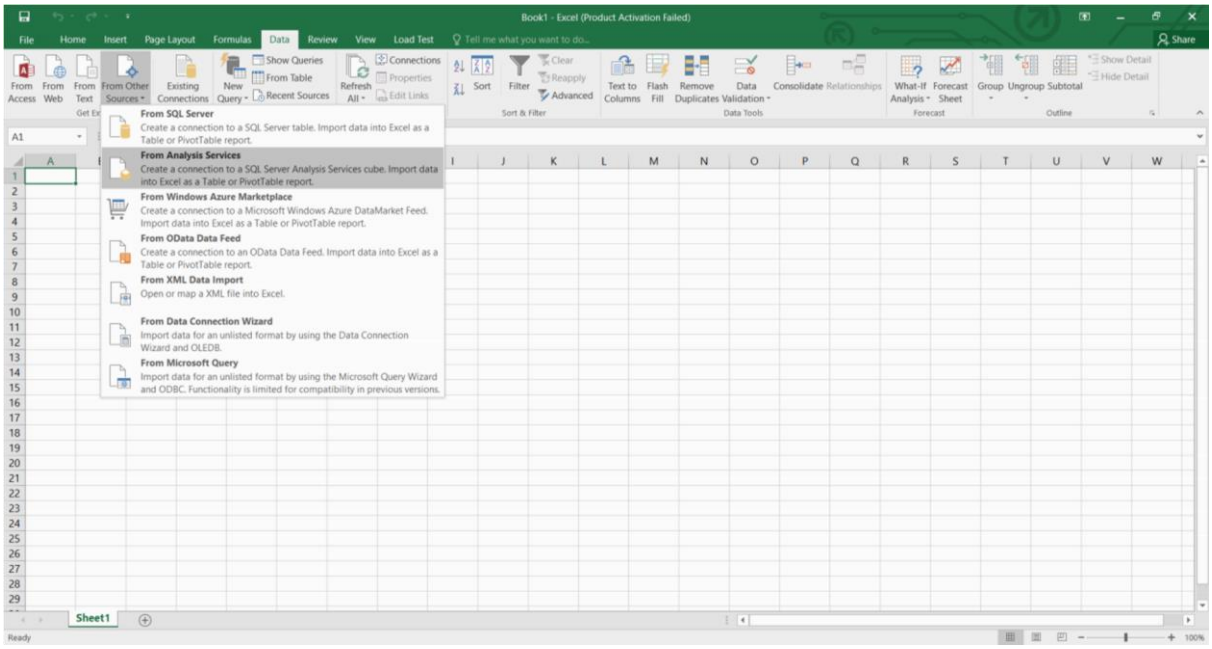


- Select -> Pivot Table -> Insert -> Pie Chart -> 3D Pie

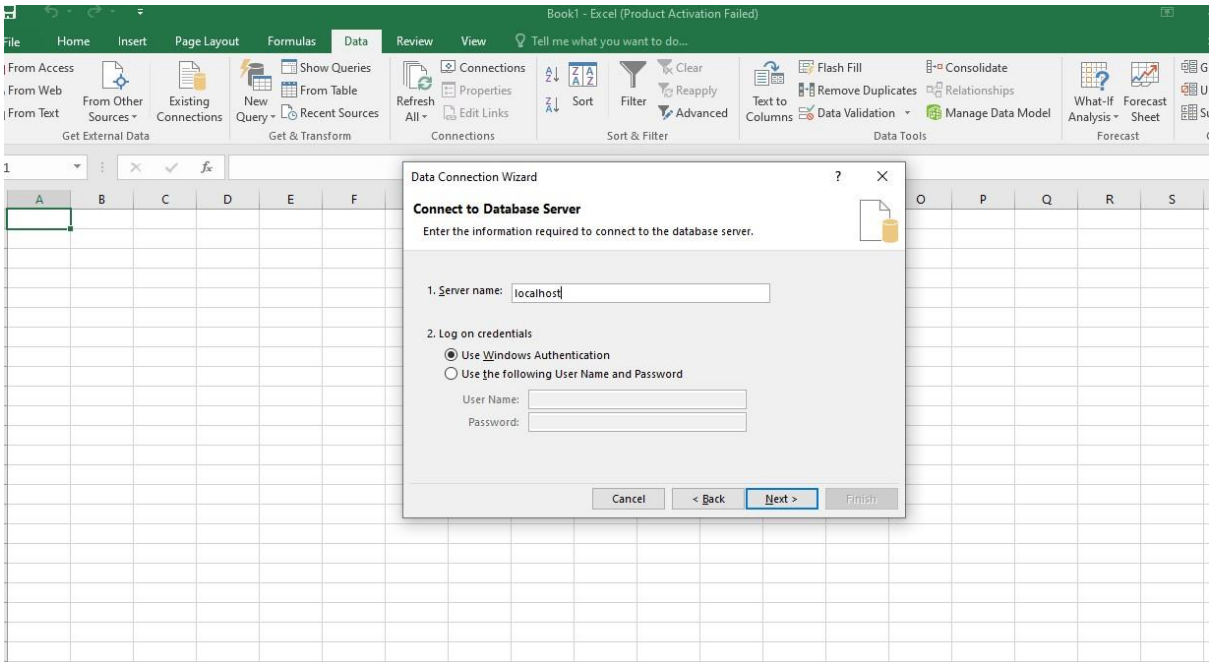

Sum of ProductKey

1.B) Import the cube in Microsoft Excel and create the Pivot table and Pivot Chart to perform data analysis.

1. First in excel click on data menu->select from other sources->from analysis services



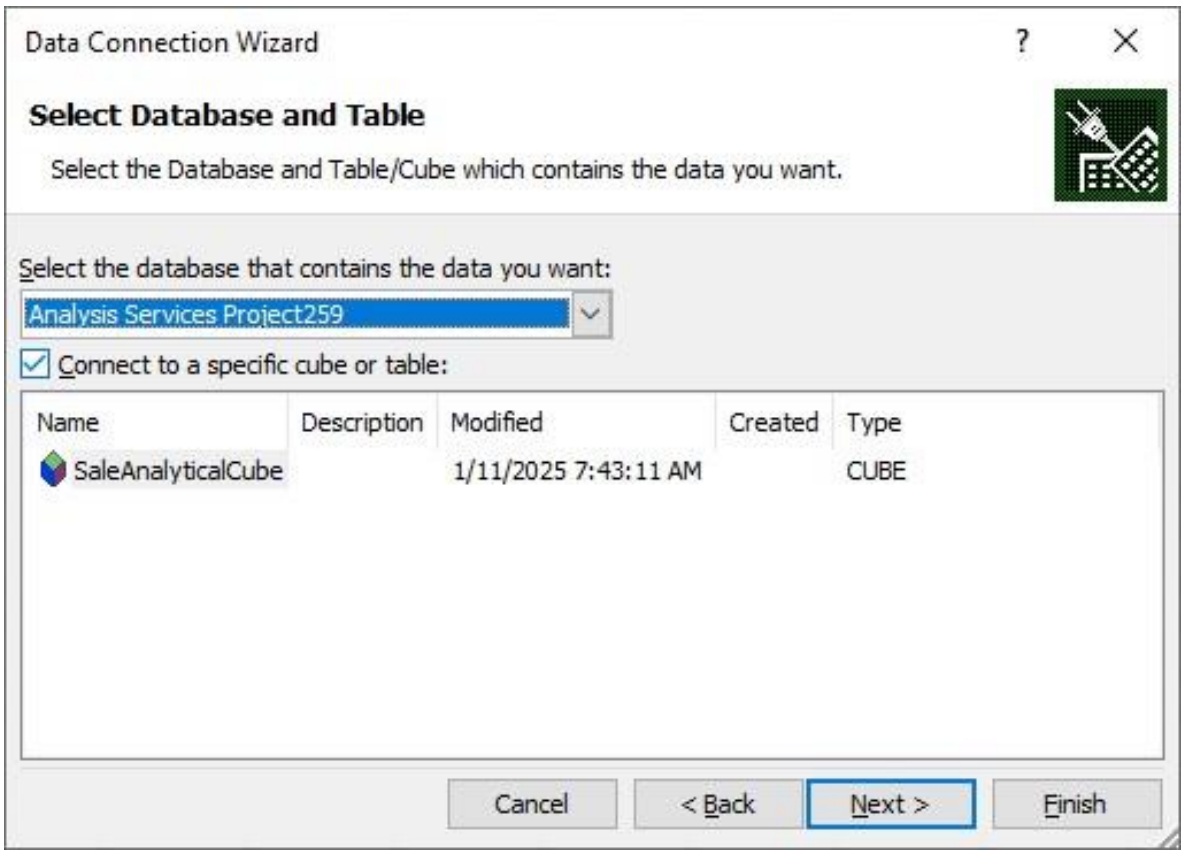
2. Enter your SQL Server name and log on to windows authentication and click next



3. In Data Connection Wizard, select database as cubepract4 and also select your cube name as SalesAnalyticlCube and click Next

4. Save file With default name and click finish

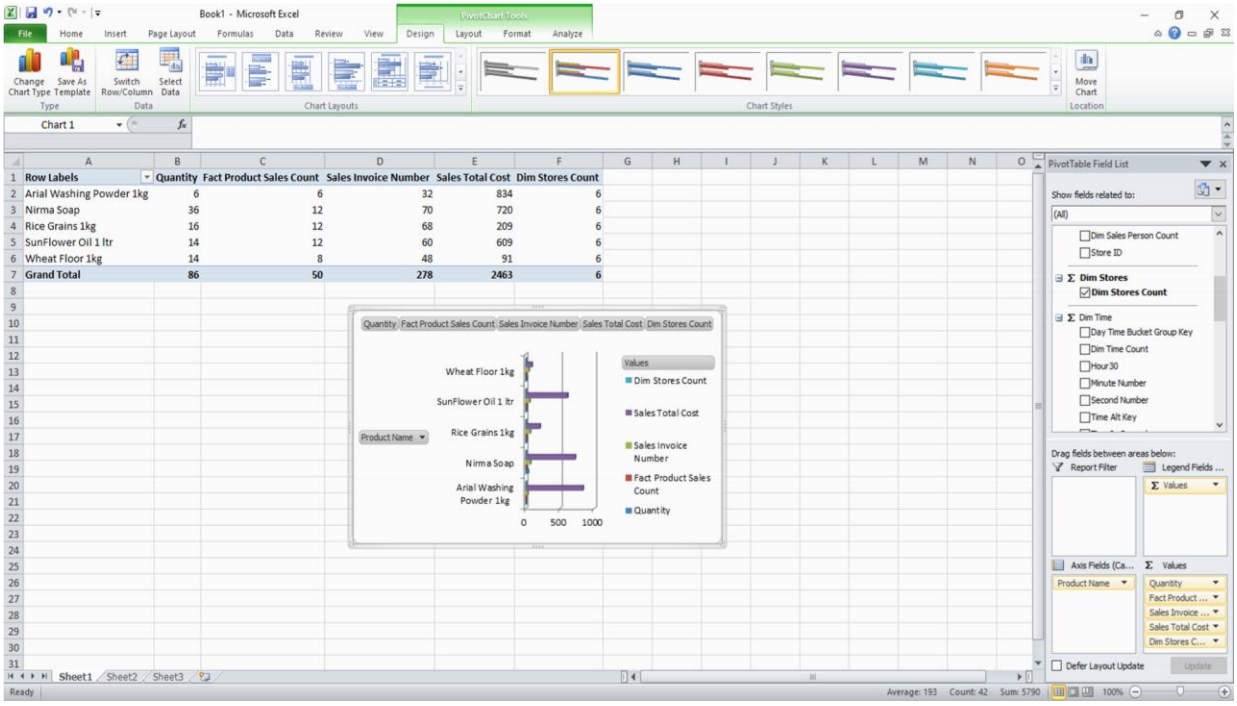
5. Now Click On Ok



6. In select pivot table fields we can fields we can choose fields to add to report -> here we are selecting all the fields

7. To create Pivot Chart again go to Insert Menu ->Pivot Chart and select any chart -> Click Ok

It will show you pivot chart as follows



Practical no.2

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

A book store and have 100 books in storage. You sell a certain % for the highest price of $50 and a certain % for the lower price of $20.

C8 =B4*(1-C4)

	A	B	C	D	E
1	**Book Store**				
2					
3		total number of books	% sold for the highest price		
4		100	60%		
5					
6			number of books	unit profit	
7		highest price	60	$50	
8		lower price	40	$20	
9					
10			total profit	$3,800	
11					

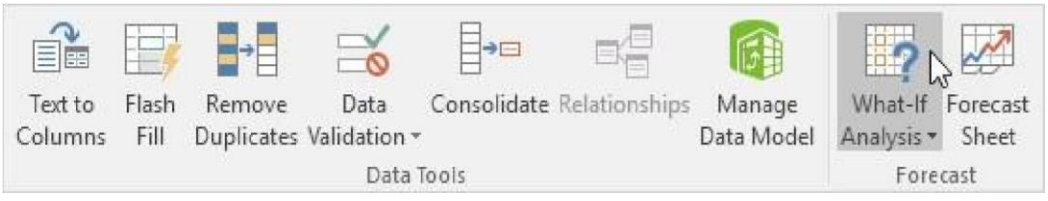
If you sell 60% for the highest price, cell D10 calculates a total profit of 60 * $50 + 40 * $20 = $3800.

Create Different Scenarios

But what if you sell 70% for the highest price? And what if you sell 80% for the highest price? Or 90%, or even 100%? Each different percentage is a different **scenario**. You can use the Scenario Manager to create these scenarios.

Note: You can simply type in a different percentage into cell C4 to see the corresponding result of a scenario in cell D10. However, what-if analysis enables you to easily compare the results of different scenarios. Read on.

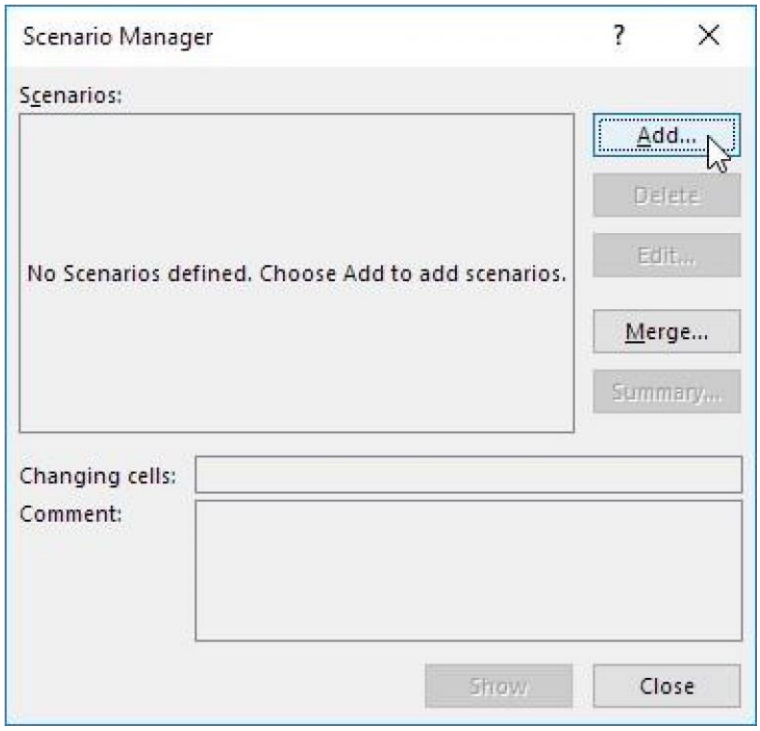
1. On the Data tab, in the Forecast group, click What-If Analysis.



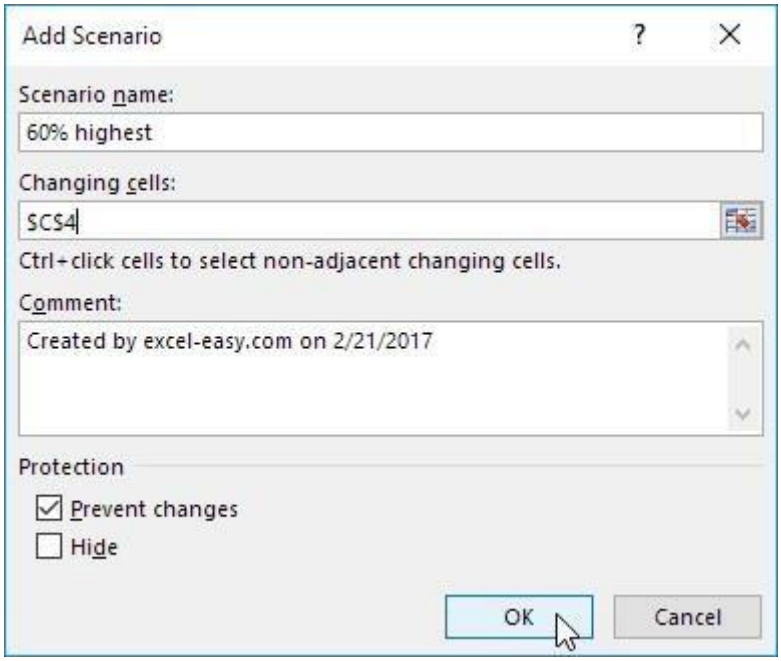
2. Click Scenario Manager.

The Scenario Manager dialog box appears.

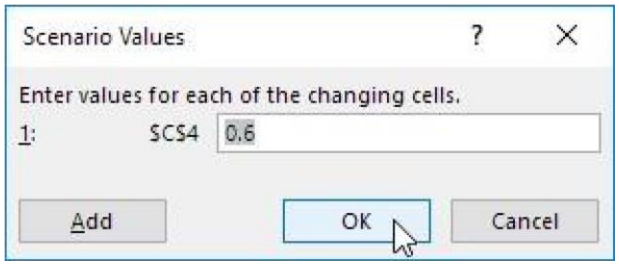
3. Add a scenario by clicking on Add.



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

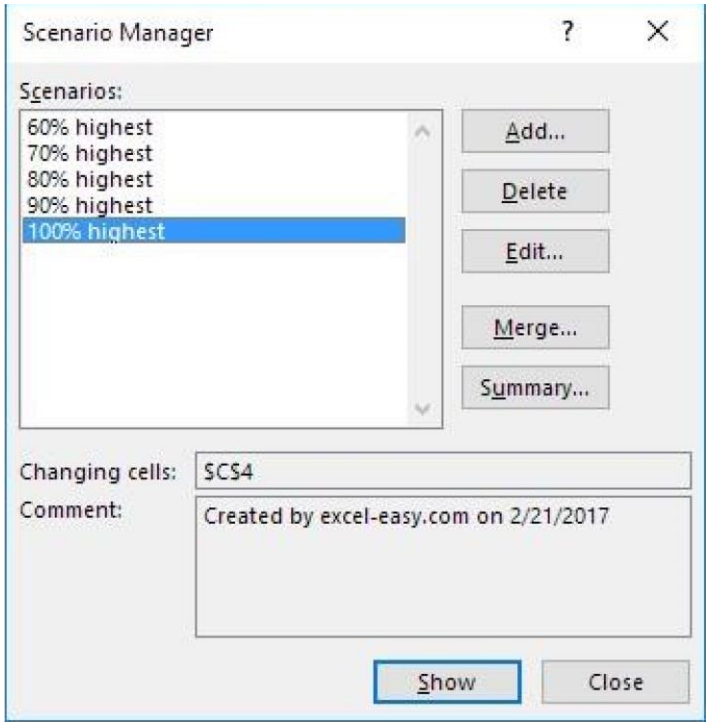


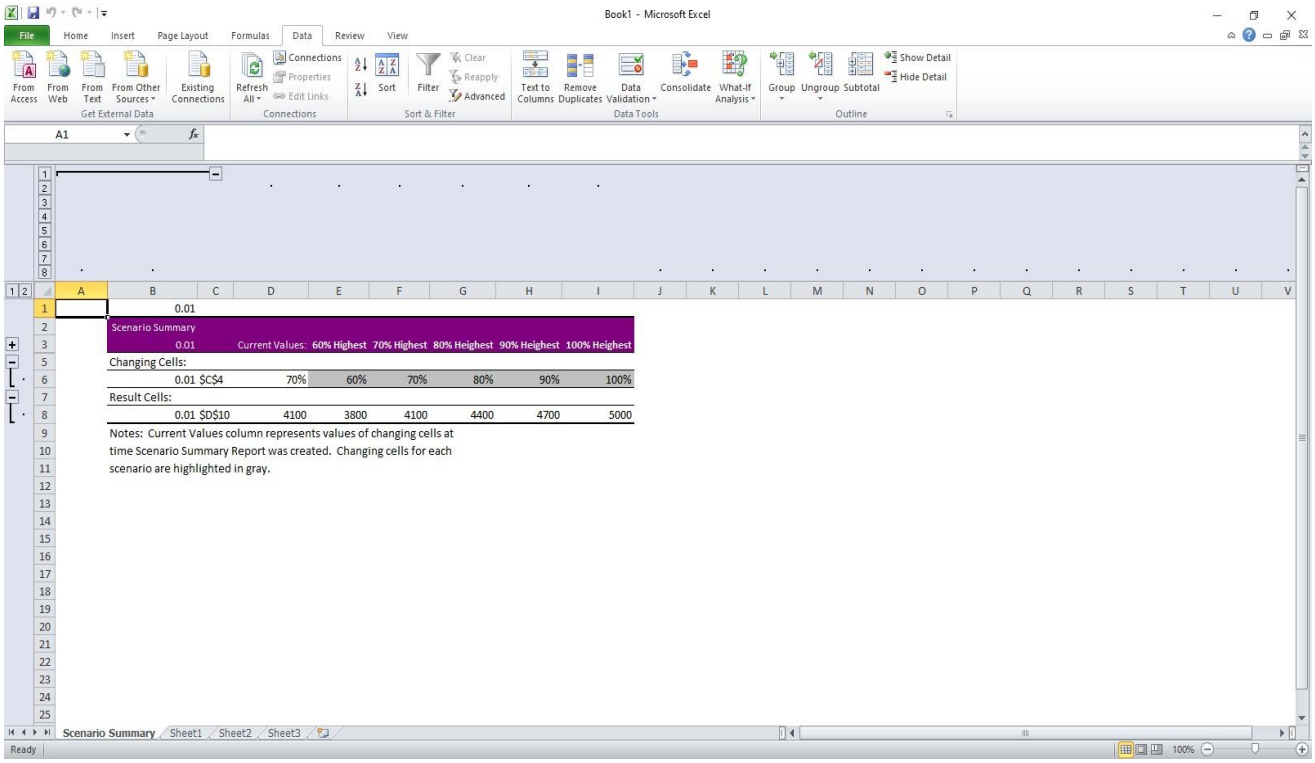
5.Enter the corresponding value 0.6 and click on OK again.



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

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





	0.01							
Scenario Summary								
	0.01	Current Values:	60% Highest	70% Highest	80% Heighest	90% Heighest	100% Heighest	
Changing Cells:								
	0.01 C4	70%	60%	70%	80%	90%	100%	
Result Cells:								
	0.01 D10	4100	3800	4100	4400	4700	5000	

Notes: Current Values column represents values of changing cells at time Scenario Summary Report was created. Changing cells for each scenario are highlighted in gray.

PRACTICAL NO: 3

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

Time series is a series of data points in which each data point is associated with a timestamp. A simple example is the price of a stock in the stock market at different points of time on a given day. Another example is the amount of rainfall in a region at different months of the year. R language

uses many functions to create, manipulate and plot the time series data. The data for the time series is stored in an R object called time-series object. It is also a R data object like a vector or data frame.

The time series object is created by using the ts() function.

Syntax

The basic syntax for ts() function in time series analysis is

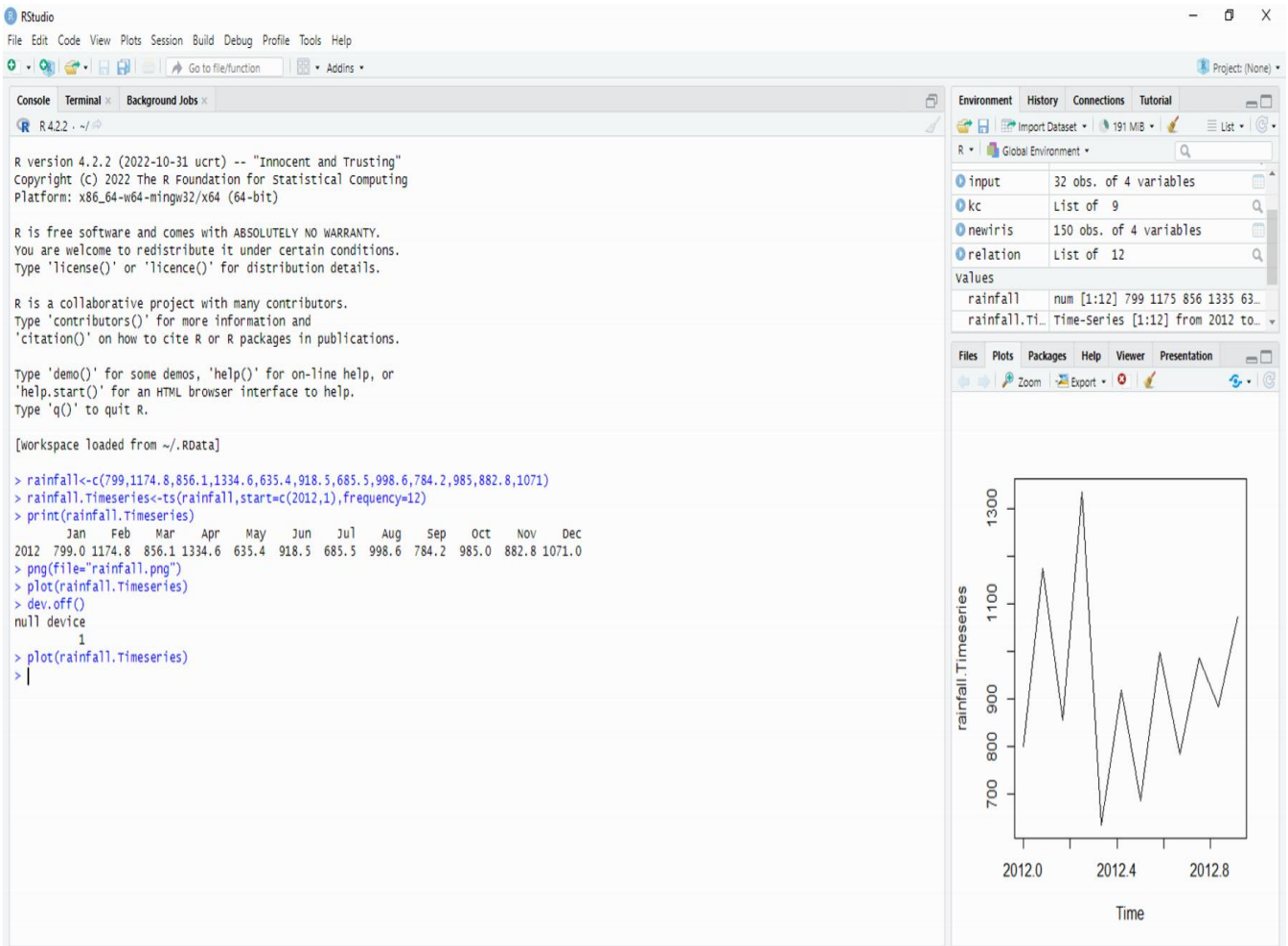
timeseries.object.name <- ts (data, start, end, frequency)

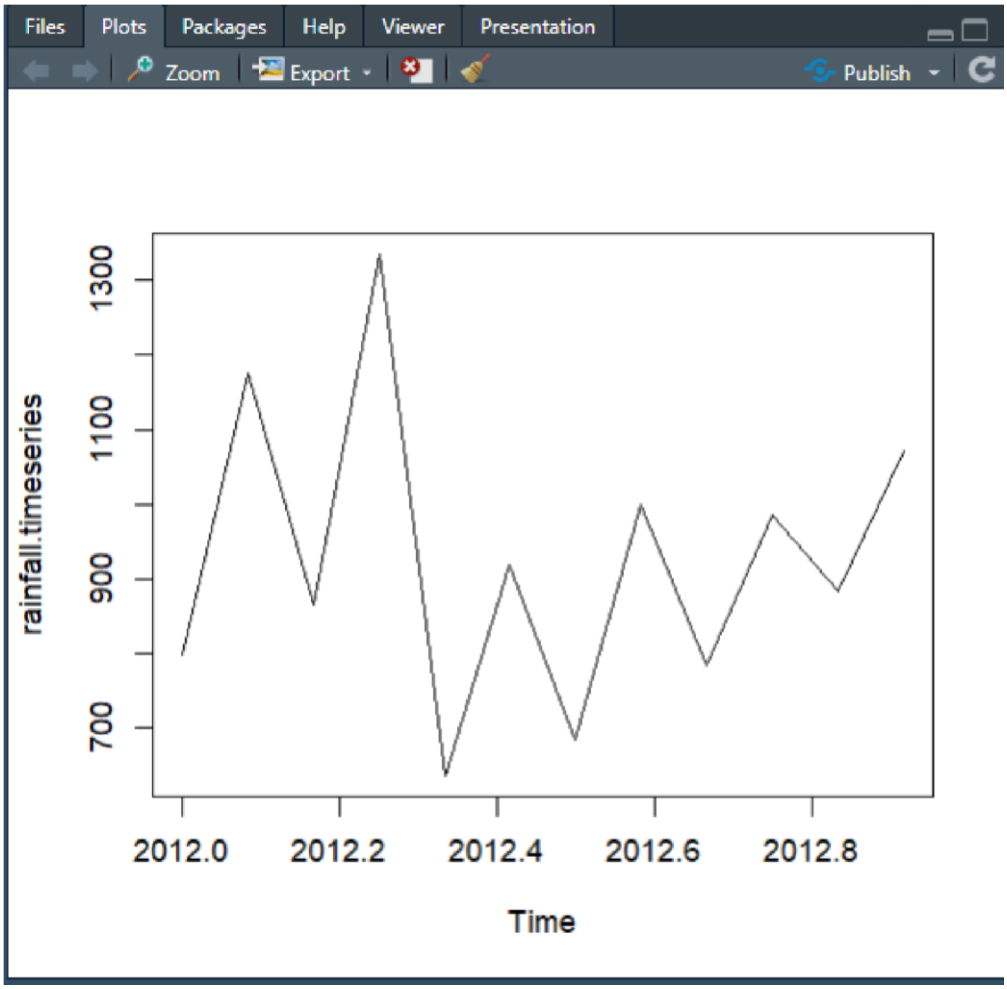
Following is the description of the parameters used -

- data is a vector or matrix containing the values used in the time series.
- start specifies the start time for the first observation in time series.
- end specifies the end time for the last observation in time series.
- frequency specifies the number of observations per unit time.

Example

Consider the annual rainfall details at a place starting from January 2012. We create an R time series object for a period of 12 months and plot it.





PRACTICAL NO: 4

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

```
newiris <- iris

newiris$Species <- NULL

(kc - <kmeans(newiris.3))

print(kc)
```

#Compare the Species label with the clustering result.

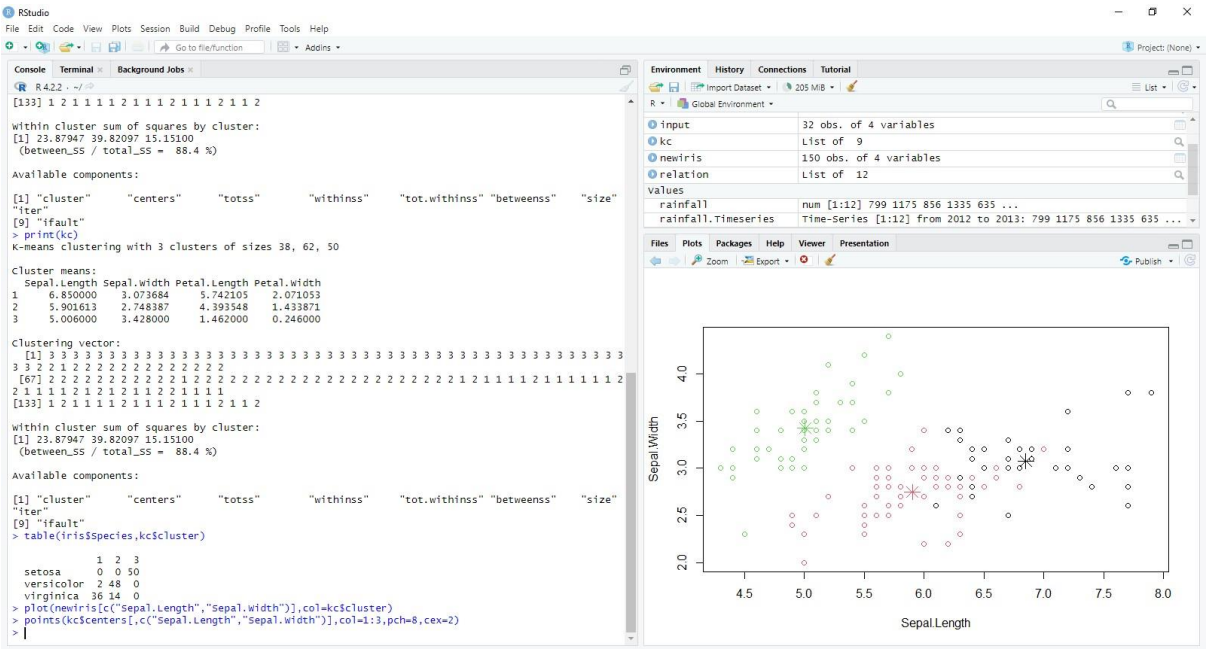
table (iris$Species,kc$cluster) plot(newiris[c("Sepal. Length",

"Sepal.Width")],col=kc$cluster)

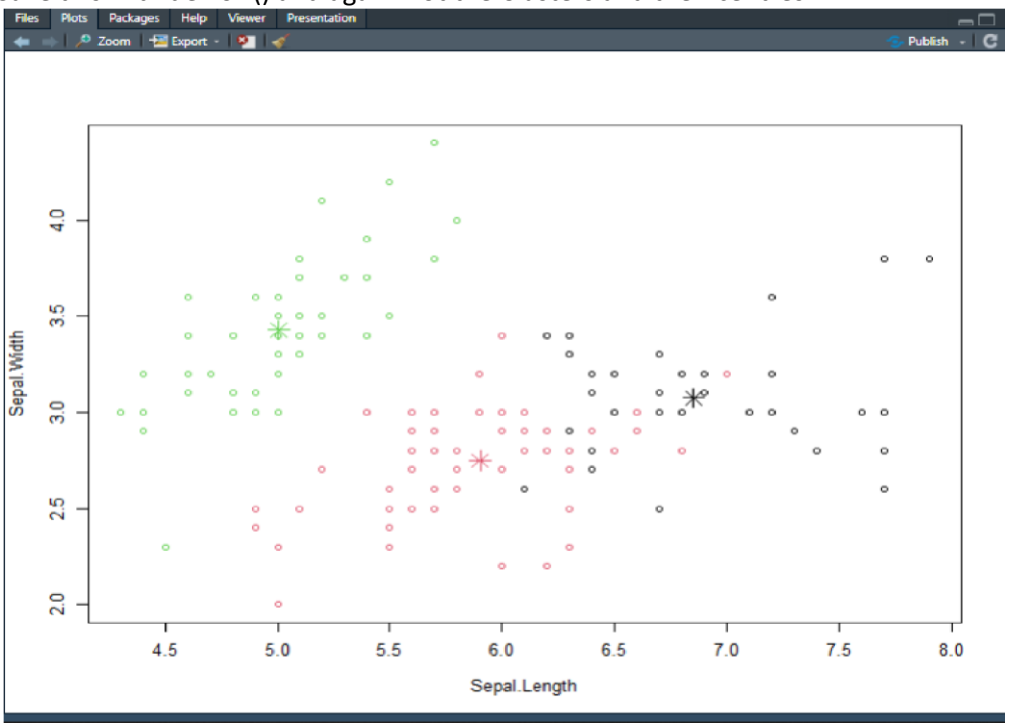
points(kc$centers[.e("Sepal.Length","Sepal.Width")].col=1:3;pch=8,cex=2)

```
> newiris<-iris
> newiris$Species<-NULL
> (kc<-kmeans(newiris,3))
K-means clustering with 3 clusters of sizes 38, 62, 50

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.901613    2.748387     4.393548    1.433871
3     5.006000    3.428000     1.462000    0.246000

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2
 [67] 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1
[133] 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1 1 2

Within cluster sum of squares by cluster:
[1] 23.87947 39.82097 15.15100
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> |
```

```
> print(kc)
K-means clustering with 3 clusters of sizes 38, 62, 50

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.901613    2.748387     4.393548    1.433871
3     5.006000    3.428000     1.462000    0.246000

Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2
 [67] 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1
[133] 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1 1 2

Within cluster sum of squares by cluster:
[1] 23.87947 39.82097 15.15100
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> |
```



Save this with dev.off() and again Plot the Clusters and their centres.



PRACTICAL NO: 5

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

In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

y = ax + b is an equation for linear regression.

Where, y is the response variable, x is the predictor variable and a and b are constants which are called the coefficients.

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is :

- Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
- Create a relationship model using the Im() functions in R.
- Find the coefficients from the model created and create the mathematical equation using these
- Get a summary of the relationship model to know the average error in prediction. Also called residuals.
- To predict the weight of new persons, use the predict() function in R.

Input Data

Below is the sample data representing the observations -

#Values of height.

151, 174, 138, 186, 128, 136, 179, 163, 152, 131

#Values of weight.

63, 81, 56, 91, 47, 57, 76, 72, 62, 48

Im() Function

This function creates the relationship model between the predictor and the response variable.

Syntax

The basic syntax for Im() function in linear regression is - lm(formula,data)

Following is the description of the parameters used :

• formula is a symbol presenting the relation between x and y. •
data is the vector on which the formula will be applied

Create Relationship Model & get the Coefficients

```
>x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

>y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

#          Apply the 1m ()

function. >relation <- 1m

(y-x)

>print (relation)
```

When we execute the above code, it produces the following result -

```
Call:

lm (formula = y ~  x)

Coefficients:
(Intercept)                    X

-38.4551                    0.6746
```

Get the Summary of the Relationship

```
>x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

>y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

#     Apply the lm()

function. >relation <-

lm (y~x)

>print (summary (relation))
```

When we execute the above code, it produces the following result -

```
call:

lm (formula = y ~ x)


Residuals:
```

Min 1Q Median 3Q Max -6.3002

-1.6629 0.0412 1.8944 3.9775

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -38.45509 8.04901 -4.778 0.00139 **

X 0.67461 0.05191 12.997 1.16e-06*** ---

Signif. codes: 0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 3.253 on 8 degrees of freedom Multiple R-squared: 0.9548, Adjusted Rsquared: 0.9491 F-statistic: 168.9 on 1 and 8 DF, p-value: 1.164e-06 **predict() Function**

Syntax

The basic syntax for predict() in linear regression is - predict(object,

newdata)

Following is the description of the parameters used

- object is the formula which is already created using the Im() function.
- newdata is the vector containing the new value for predictor variable.

Predict the weight of new persons

The predictor vector.

>x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

The resposne vector.

>y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

Apply the lm() function.

>relation <-

1m(y-x)

Find weight of a person with height

170

>a <- data.frame(x = 170)

>result <-predict (relation, a)

>print (result)

Result:

1

76.22869

Visualize the Regression Graphically

Create the predictor and response variable.

>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 <- 1m (y-x)

Give the chart file a name. >png(file

"linearregression.png")

Plot the chart.

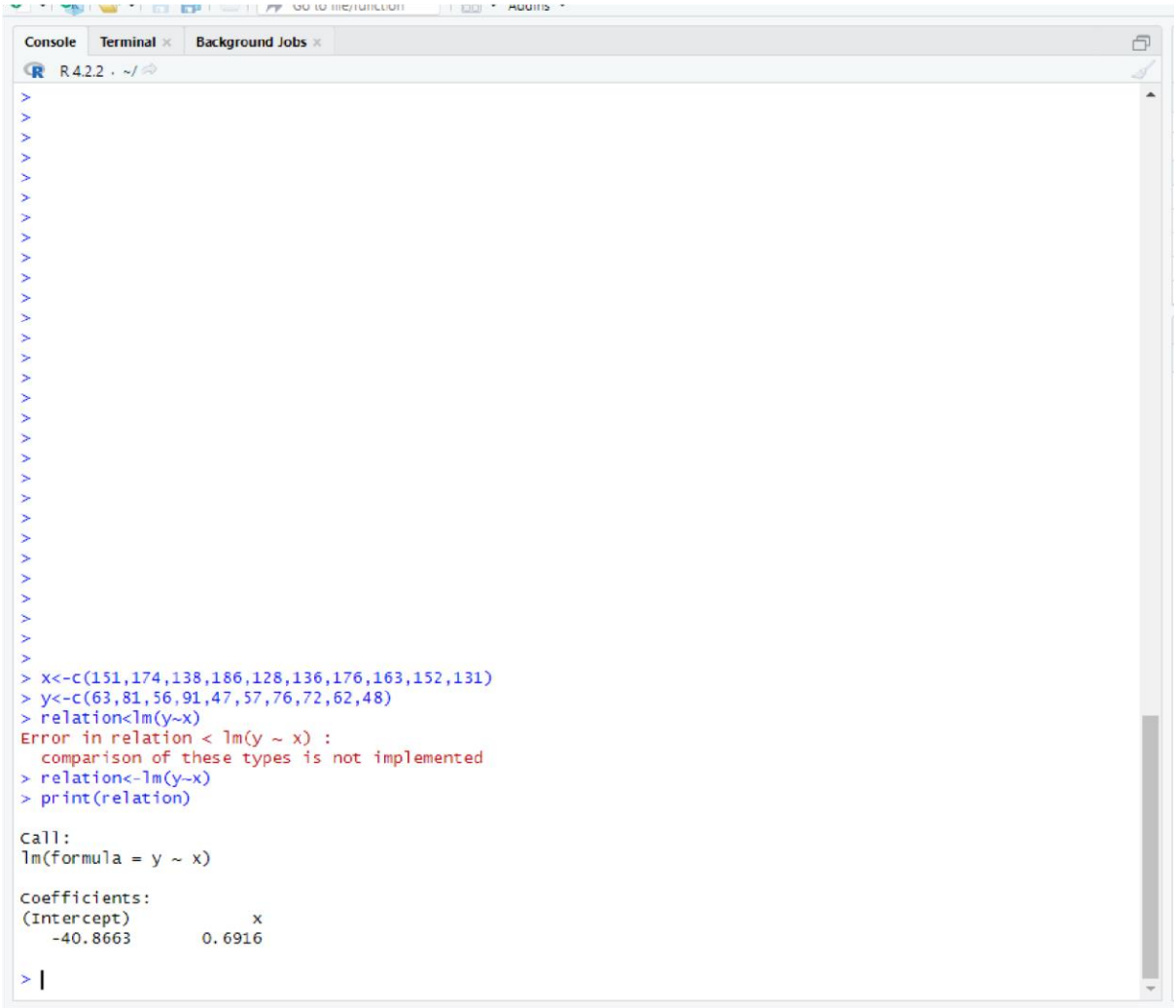
>plot (y,x,col= "blue", main = "Height & Weight Regression", abline (lm (x ~ y)),

cex 1.3, pch = 16, xlab = "Weight in Kg", ylab= "Height in cm")

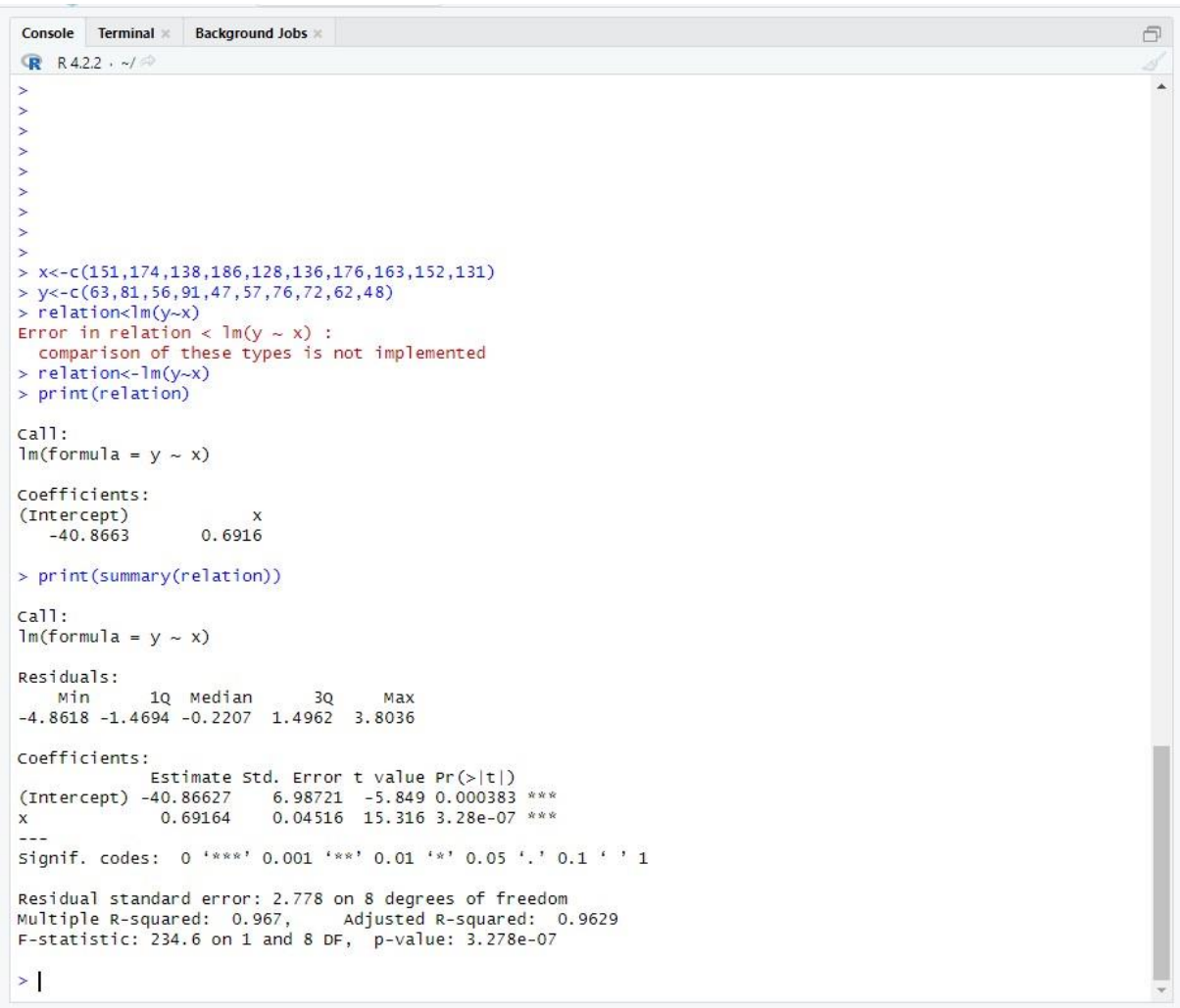
Save thefile.

>dev.off()

1)Create relationship model and get the coefficient



2)Get the summary of relationship

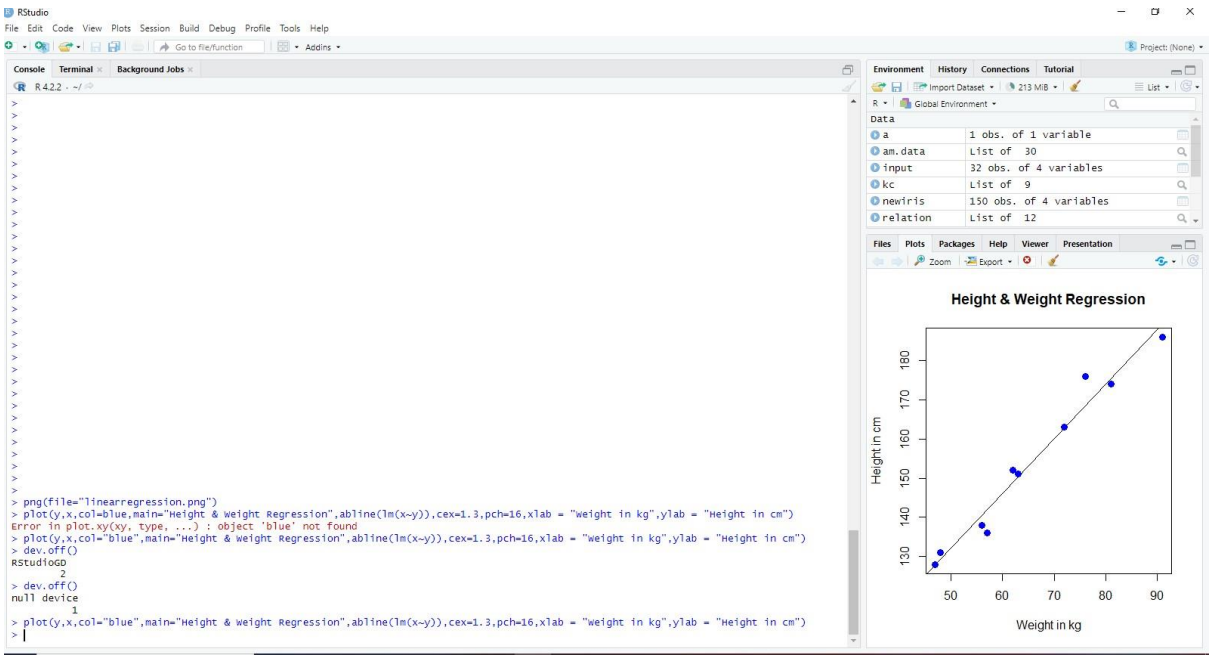


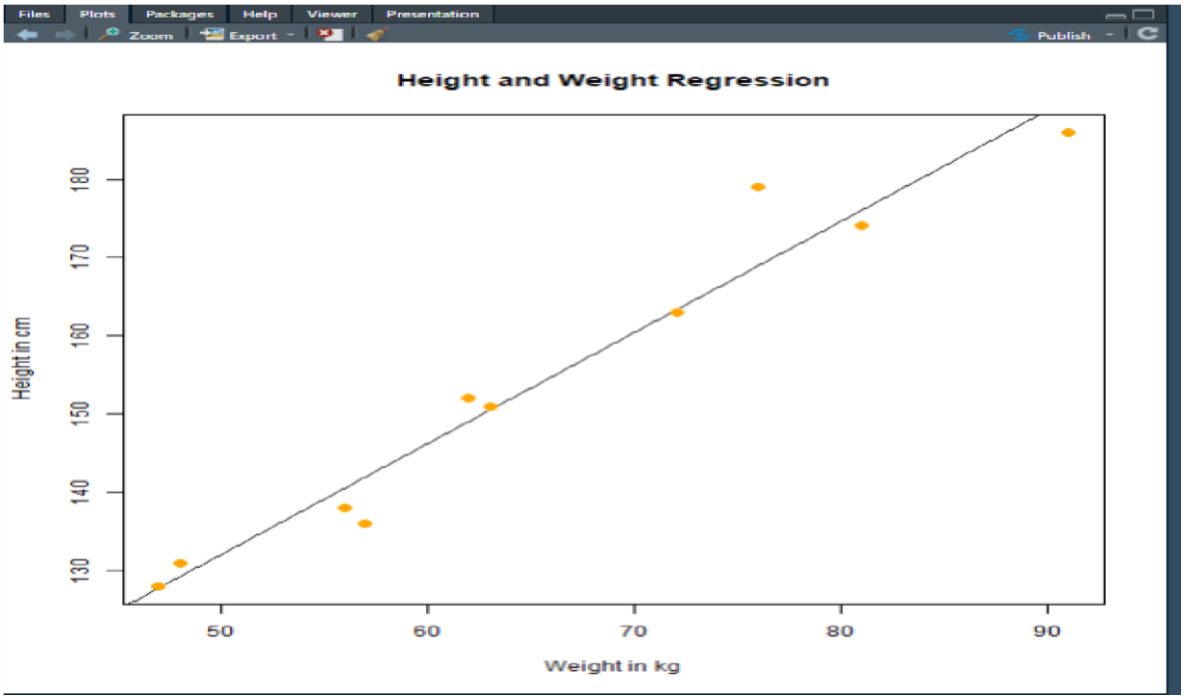
3)Predict the weight of new person

3a)Find weight of person with height 170

```
> y<-c(63,81,56,91,47,57,76,72,62,48)
> relation<lm(y~x)
Error in relation < lm(y ~ x) :
  comparison of these types is not implemented
> relation<-lm(y~x)
> print(relation)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   -40.8663       0.6916

> print(summary(relation))

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-4.8618 -1.4694 -0.2207  1.4962  3.8036

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -40.86627    6.98721  -5.849 0.000383 ***
x             0.69164    0.04516  15.316 3.28e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.778 on 8 degrees of freedom
Multiple R-squared:  0.967,     Adjusted R-squared:  0.9629
F-statistic: 234.6 on 1 and 8 DF,  p-value: 3.278e-07

> x<-c(151,174,138,186,128,136,176,163,152,131)
> y<-c(63,81,56,91,47,57,76,72,62,48)
> relation<-lm(y~x)
> a<-data.frame(x=170)
> predict(relation,a)
       1
76.71201
> print(result)
       1
76.22869
>
```

3b)Visualize the regression graphically




Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Height and Weight Regression
Height in cm
130
140
150
160
170
180
50
60
70
80
90
Weight in kg

PRACTICAL NO: 6

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

The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1. It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables.

The general mathematical equation for logistic regression is - y

= 1/(1+e^-(a+b1x1+b2x2+b3x3+...))

Following is the description of the parameters used -

- y is the response variable.
- x is the predictor variable.
- a and b are the coefficients which are numeric constants.

The function used to create the regression model is the glm() function.

Syntax

The basic syntax for glm() function in logistic regression is glm(formula,

data, family)

Following is the description of the parameters used

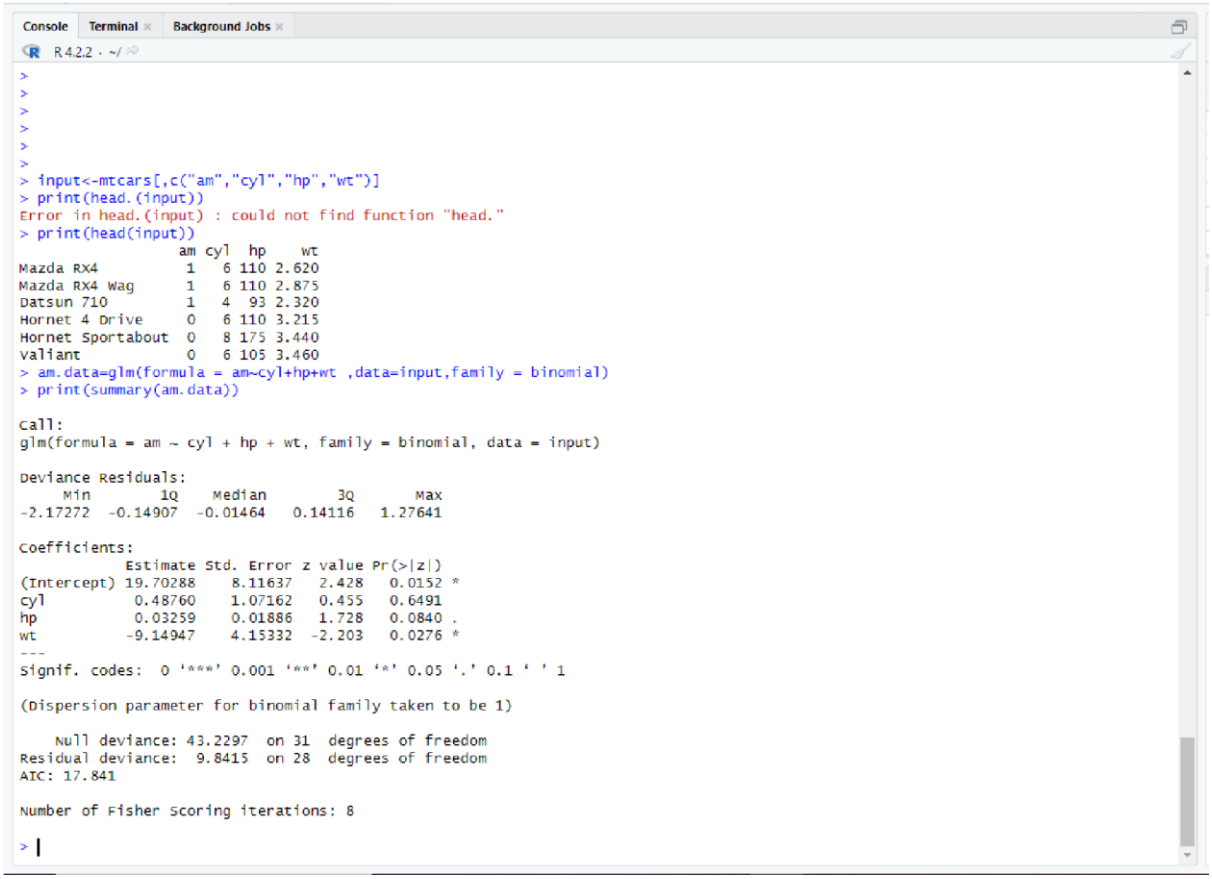
- formula is the symbol presenting the relationship between the variables.
- data is the data set giving the values of these variables.
- family is R object to specify the details of the model. It's value is binomial for logistic regression.

Example

The in-built data set "mtcars" describes different models of a car with their various engine specifications. In "mtcars" data set, the transmission mode (automatic or manual) is described by the column am which is a binary value (0 or 1). We can create a logistic regression model between the columns "am" and 3 other columns - hp, wt and cyl.

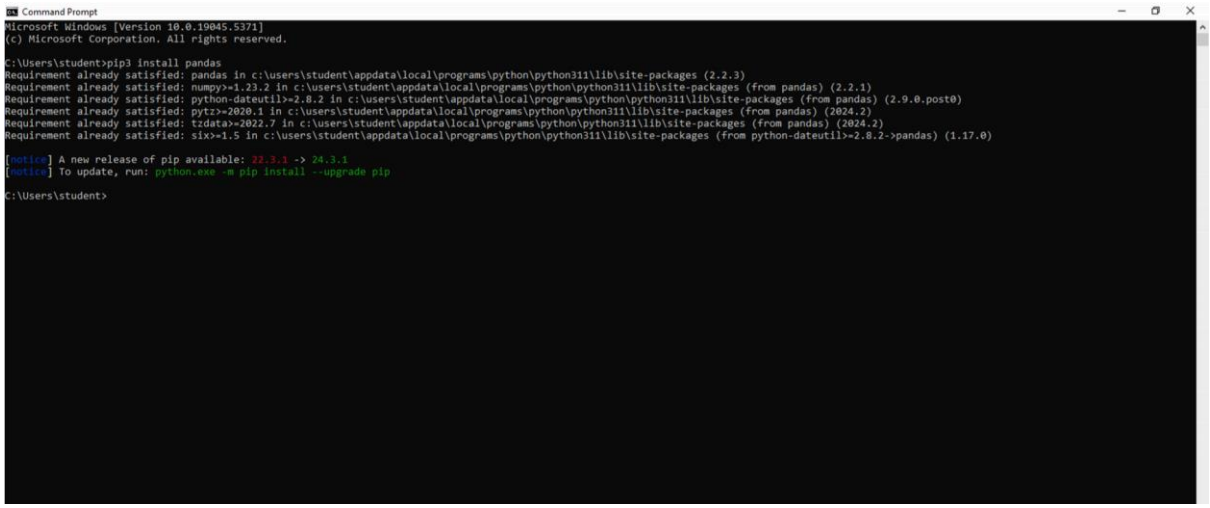
Conclusion:

In the summary as the p-value in the last column is more than 0.05 for the variables "cyl" and "hp", we consider them to be insignificant in contributing to the value of the variable "am". Only weight (wt) impacts the "am" value in this regression model.



PRACTICAL NO: 7

Aim : 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).



In this section we are going to do simple data analysis on csv file and figure out the answers to the following questions:-

1. How many male and female passengers were onboard the Titanic?

2. How many male and female members survived the Titanic shipwreck?

3. What is the median age of each sex?

7.1)HOW MANY MALES AND FEMALES import

pandas as pd df=pd.read_csv('train.csv')

print(df.columns)

print(df['Sex'].value_counts())

print(df['Sex'].value_counts(normalize=True))

Insights:

The above analysis shows that 65% of people on Titanic were Male and 35% were Female.

7.2)HOW MANY SURVIVED IN TITANIC import pandas as pd

df=pd.read_csv('train.csv') print(df.columns)

print(df[df['Survived']==1]['Sex'].value_counts())

print(df[df['Survived']==1]['Sex'].value_counts(normalize=True))

In the above code, first you filter the dataframe for surviving passengers and then use the value_counts() method to find out the unique male and female passengers.

Insights:

The above analysis shows that 68% of surviving people on the Titanic were Female.

7C)MEDIAN AGE OF EACH SEX import pandas as pd

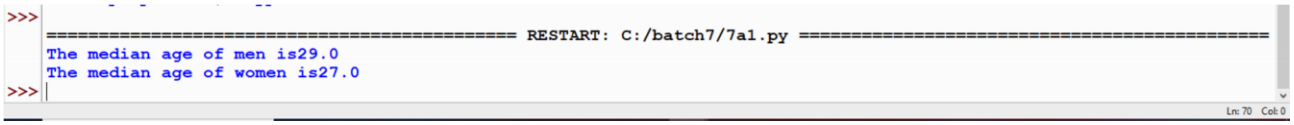
df=pd.read_csv('train.csv') median_age_men=

df[df['Sex']=='male'] ['Age'].median() median_age_women=

df[df['Sex']=='female'] ['Age'].median() print(f"The median

age of men is{median_age_men}")

print(f"The median age of women is{median_age_women}")



Insights:

The above analysis shows that median age of male was 29 whereas median age of female was 27.

PRACTICAL NO: 8

Aim : Perform data visualization

8.A) Perform data visualization using Python on any sales data.

Python provides various libraries that come with different features for visualizing data. All these libraries come with different features and can support various types of graphs.

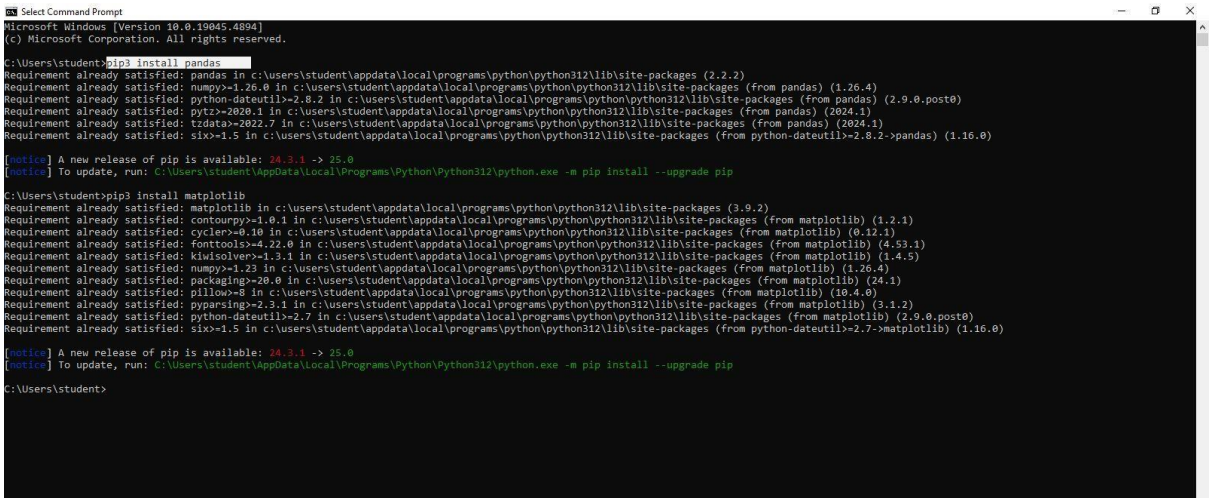
We will discuss the matplotlib library here and will plot some most commonly used graphs.

Tips Database

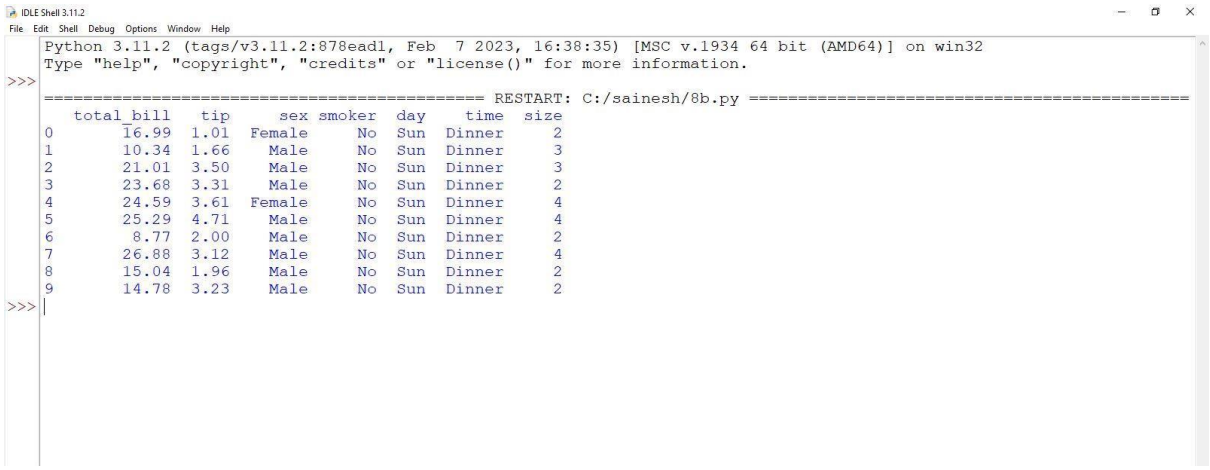
Tips database is the record of the tip given by the customers in a restaurant for two and a half months in the early 1990s. It contains 6 columns such as total_bill, tip, sex, smoker, day, time, size.

To install pandas library below command in the terminal.

pip3 install pandas




```
import pandas as pd data=pd.read_csv("tips.csv")

print(data.head(10))
```

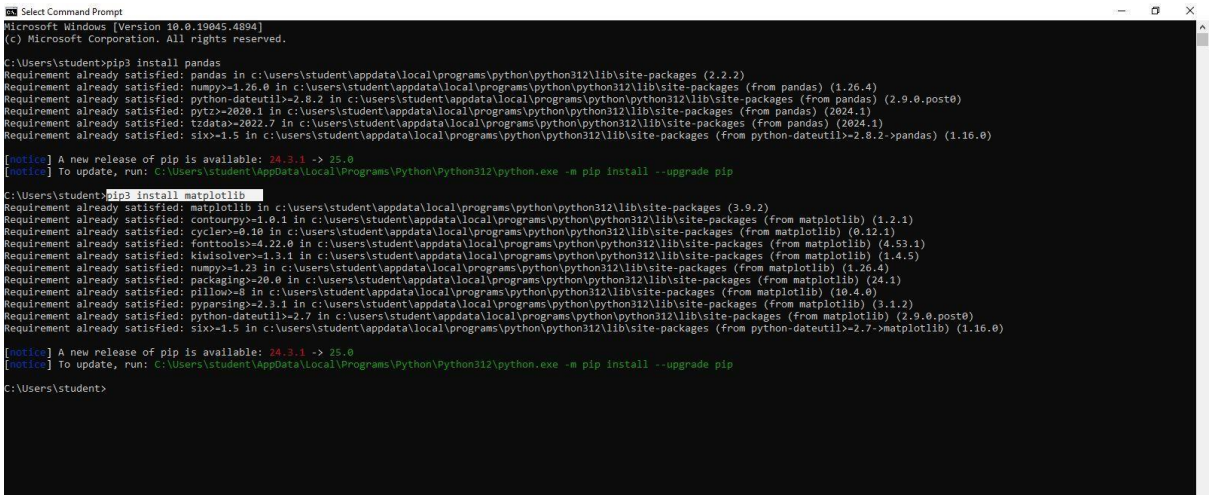


Matplotlib is an easy-to-use, low-level data visualization library that is built on NumPy arrays. It consists of various plots like scatter plot, line plot, histogram, etc. Matplotlib provides a lot of flexibility.

To install this type the below command in the terminal. **pip3**

install matplotlib

After installing Matplotlib, let's see the most commonly used plots using this library.

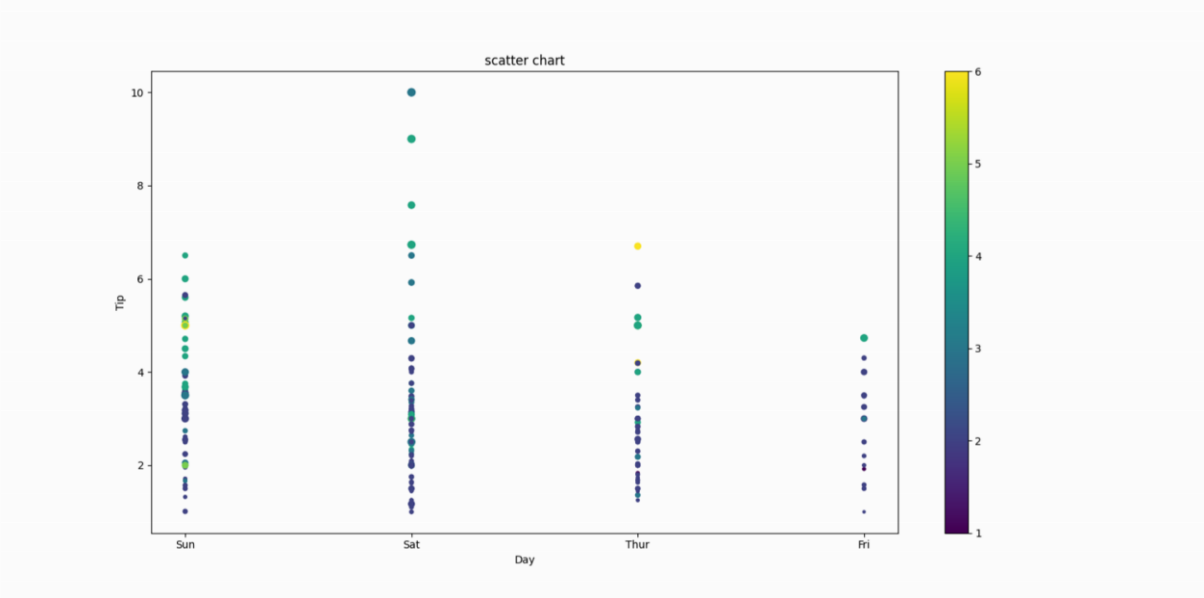


1. Scatter Plot

Scatter plots are used to observe relationships between variables and uses dots to represent the relationship between them. The scatter() method in the matplotlib library is used to draw a scatter plot. We can add colors and also change the size of the points. We can do this by using the c and s parameters respectively of the scatter function. We can also show the color bar using the colorbar() method.

Example:

```
import pandas as pd import matplotlib.pyplot as plt data =
pd.read_csv("tips.csv")
plt.scatter(data['day'],data['tip'],c=data['size'],s=data['total_bill'])


plt.title("scatter chart")
plt.xlabel('Day')
plt.ylabel('Tip')
plt.colorbar()

plt.show()
```



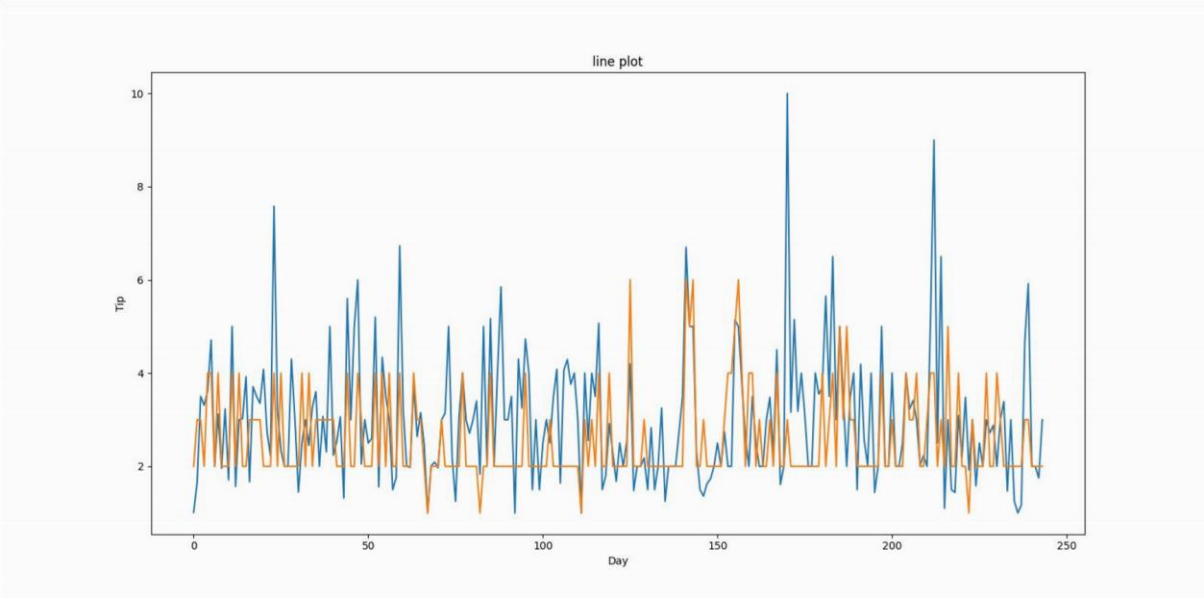
2. Line Chart

Line Chart is used to represent a relationship between two data X and Y on a different axis. It is plotted using the plot() function. Let's see the below example.

Example:

```
import pandas as pd import
matplotlib.pyplot as plt data =
pd.read_csv("tips.csv")
```

```
plt.plot(data['tip'])
plt.plot(data['size'])

plt.title("line plot")

plt.xlabel('Day')

plt.ylabel('Tip')

plt.show()
```

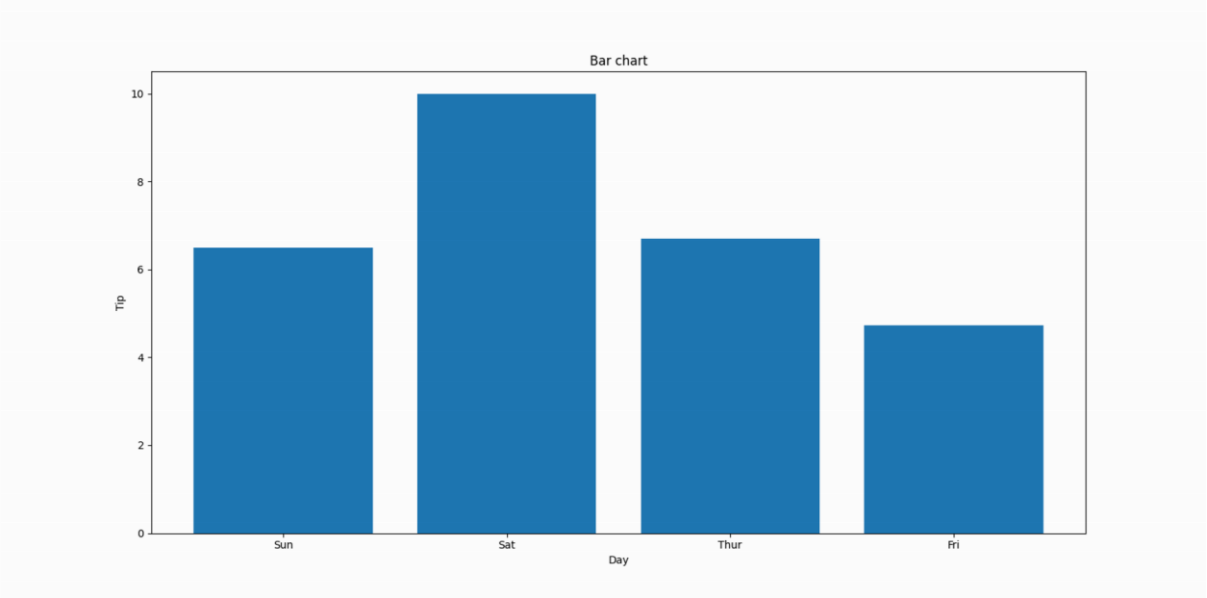


3.Bar Chart

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. It can be created using the bar() method.

Example:

```
import pandas as pd import

matplotlib.pyplot as plt data =

pd.read_csv("tips.csv")

plt.bar(data['day'],data['tip'])

plt.title("Bar chart")

plt.xlabel('Day') plt.ylabel('Tip')

plt.show()
```



4. Histogram

A histogram is basically used to represent data in the form of some groups. It is a type of bar plot where the X-axis represents the bin ranges while the Y-axis gives information about frequency. The hist() function is used to compute and create a histogram. In histogram, if we pass categorical data then it will automatically compute the frequency of that data i.e. how often each value occurred.

Example:

import pandas as pd import

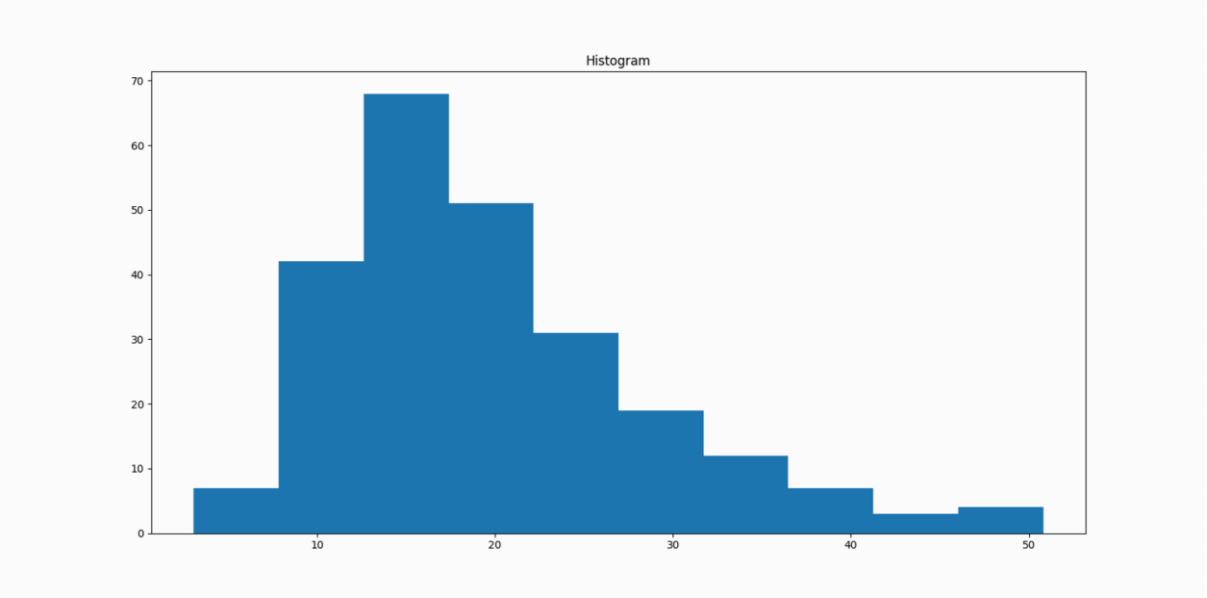
matplotlib.pyplot as plt data =

pd.read_csv("tips.csv")

plt.hist(data['total_bill'])

plt.title("Histogram")

plt.show()



5. Pie Chart

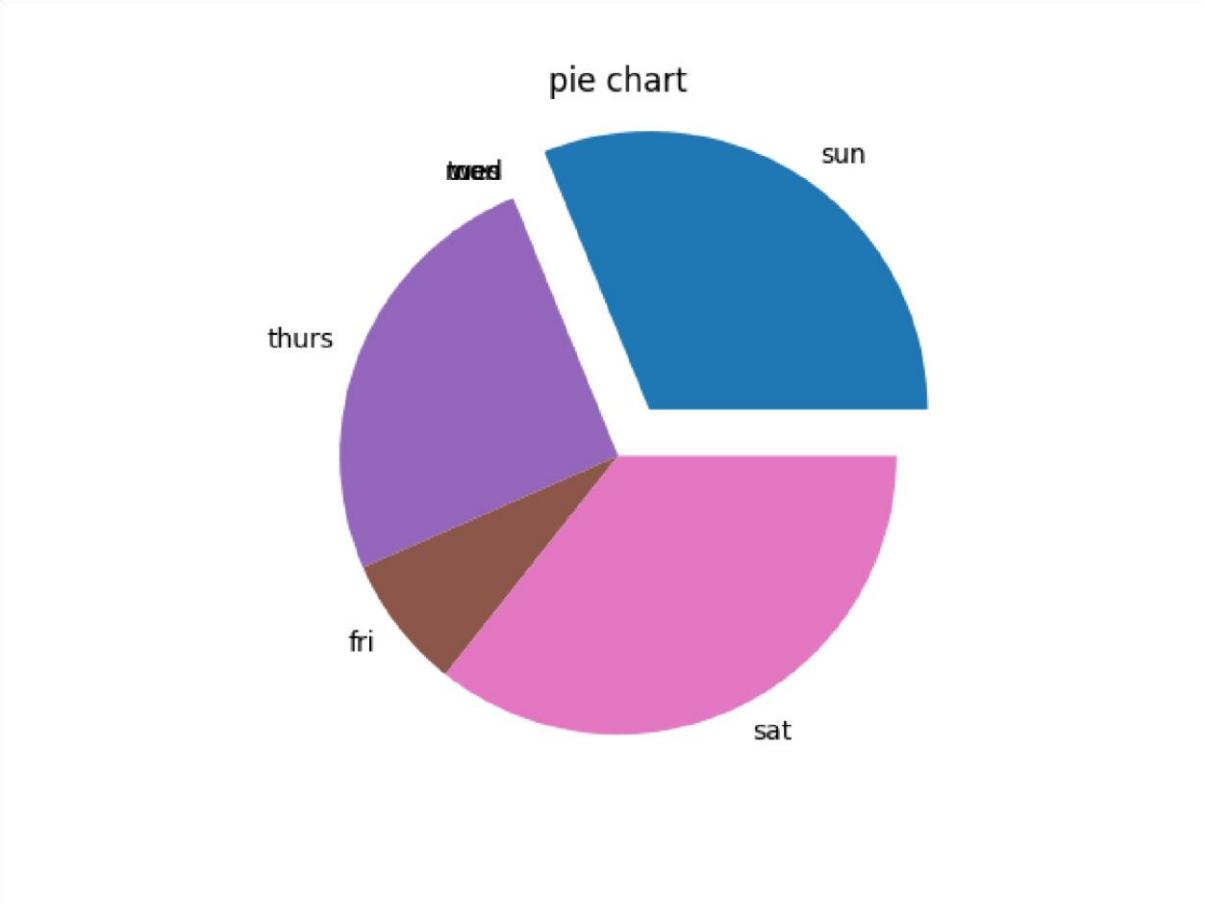
A Pie Chart is a circular statistical plot that can display only one series of data. The area of the chart is the total percentage of the given data. The area of slices of the pie represents the percentage of the parts of the data. The slices of pie are called wedges. The area of the wedge is determined by the length of the are of the wedge.

Example:

```
import pandas as pd import matplotlib.pyplot as

plt import array df=pd.read_csv("tips.csv")

count_sun=df['day'].value_counts().get('Sun',0)

print("occurance of sun",count_sun)

count_mon=df['day'].value_counts().get('Mon',0)

print("occurance of mon",count_mon)

count_tues=df['day'].value_counts().get('Tues',0)

print("occurance of tues",count_tues)

count_wed=df['day'].value_counts().get('Wed',0)

print("occurance of wed",count_wed)

count_thurs=df['day'].value_counts().get('Thur',0)

print("occurance of thurs",count_thurs)

count_fri=df['day'].value_counts().get('Fri',0)

print("occurance of fri",count_fri)
```

```
count_sat=df['day'].value_counts().get('Sat',0)
print("occurance of sat",count_sat)
mylabel=["sun","mon","tues","wed","thurs","fri",
"sat"] e=(0.2,0,0,0,0,0,0)
arr=array.array('i',[count_sun,count_mon,count_t
ues,count_wed,count_thurs,count_fri,count_sat])
plt.pie(arr,explode=e,labels=mylabel)
plt.title("pie chart") plt.show()
```

```
    n = int(2 ** np.ceil((eta2 - eta1) / halfpi))
ValueError: cannot convert float NaN to integer
>>>
========== RESTART: C:/sainesh/8b.2.py ==========
occurance of sun 76
occurance of mon 0
occurance of tues 0
occurance of wed 0
occurance of thurs 62
occurance of fri 19
occurance of sat 87
```



8.B) Perform data visualization using PowerBI on any sales data.

Step 1: Data Extraction:

The data extraction is first step of ETL. There are 2 Types of Data Extraction

1. Full Extraction: All the data from source systems or operational systems gets extracted to staging area. (Initial Load)

2. Partial Extraction: Sometimes we get notification from the source system to update specific date. It is called as Delta load.

Source System Performance: The Extraction strategies should not affect source system performance.

Step 2: Data Transformation:

The data transformation is second step. After extracting the data there is big need to do the transformation as per the target system. I would like to give you some bullet points of Data Transformation.

- Data Extracted from source system is in to Raw format. We need to transform it before loading in to target server.
- Data has to be cleaned, mapped and transformed
- There are following important steps of Data Transformation:

1.Selection: Select data to load in target

2. Matching: Match the data with target system

3.Data Transforming: We need to change data as per target table structures

Real life examples of Data Transformation:

- Standardizing data: Data is fetched from multiple sources so it needs to be standardized as per the target system
- Character set conversion: Need to transform the character sets as per the target systems. (Firstname and last name example)
- Calculated and derived values: In source system there is first val and second val and in target we need the calculation of first val and second val.
- Data Conversion in different formats: If in source system date in in DDMMYY format and in target the date is in DDMONYYYY format then this transformation needs to be done at transformation phase.

Step 3: Data Loading

- Data loading phase loads the prepared data from staging tables to main tables.

ETL Process in Power BI

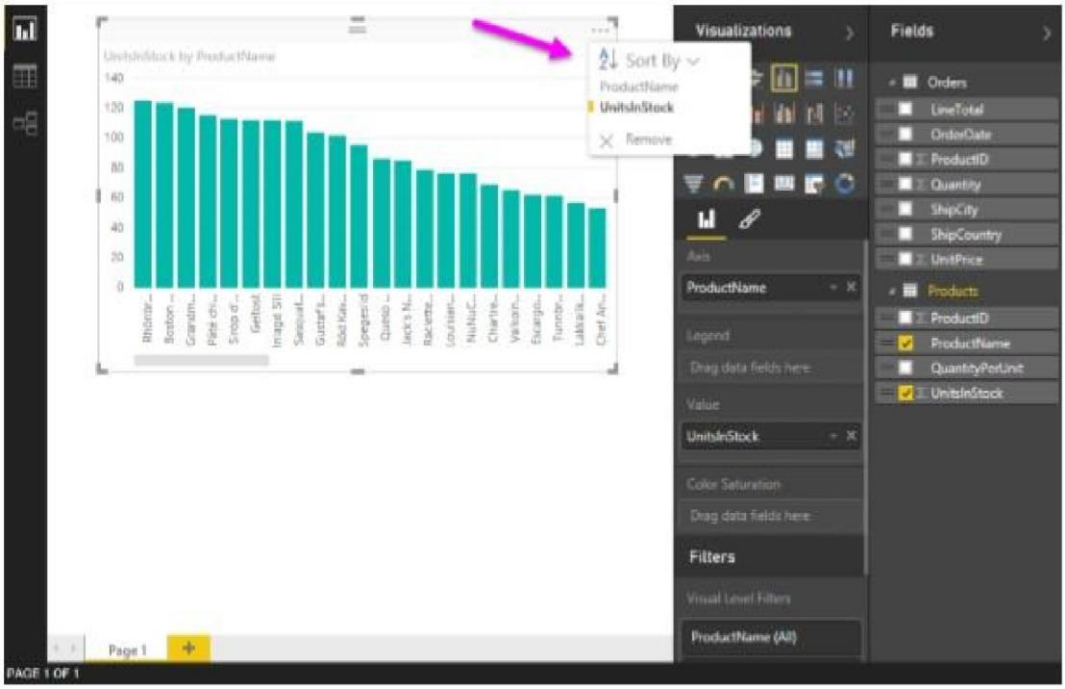
1) Remove other columns to only display columns of interest

In this step you remove all columns except **ProductID**, **ProductName**, **UnitsInStock**, and **Quantity PerUnit**

Power BI Desktop lets you create a variety of visualizations to gain insights from your data. You can build reports with multiple pages and each page can have multiple visuals. You can interact with your visualizations to help analyze and understand your data

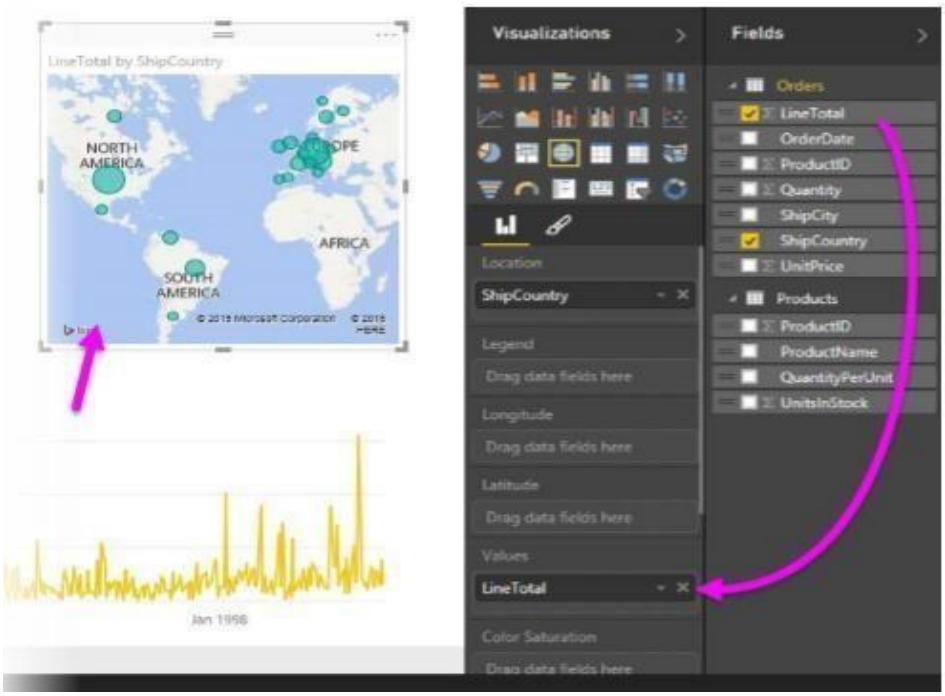
In this task, you create a report based on the data previously loaded. You use the Fields pane to select the columns from which you create the visualizations.

Step 1: Create charts showing Units in Stock by Product and Total Sales by Year 1. Drag UnitsInStock from the Field pane (the Fields pane is along the right of the screen) onto a blank space on the canvas. A Table visualization is created. Next, drag ProductName to the Axis box, found in the bottom half of the Visualizations pane. Then we then select Sort By > UnitsInStock using the skittles in the top right corer of the visualization.



2. Drag OrderDate to the canvas beneath the first chart, then drag LineTotal (again, from the Fields pane) onto the visual, then select Line Chart. The following visualization is created.

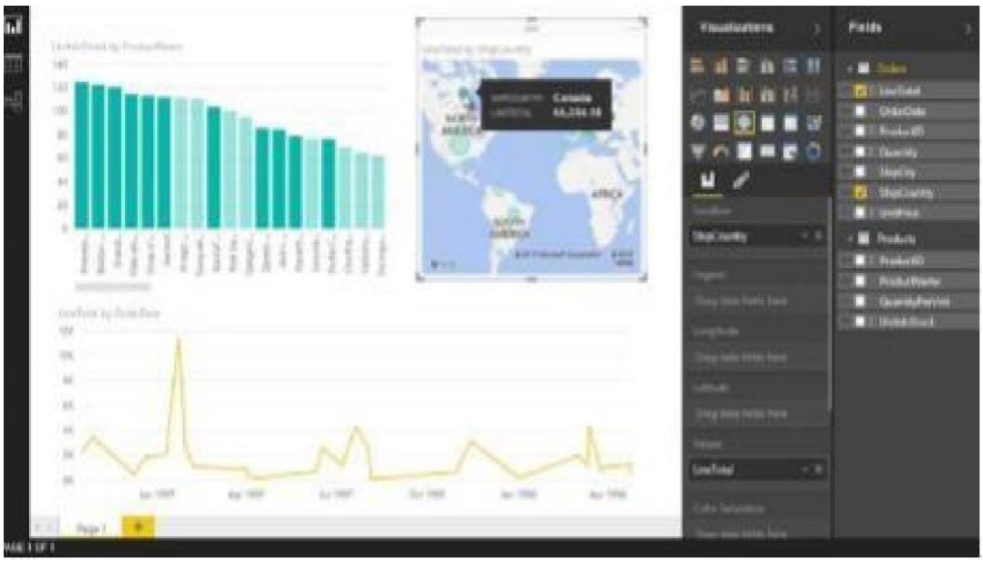
3. Next, drag ShipCountry to a space on the canvas in the top right. Because you selected a geographic field, a map was created automatically. Now drag LineTotal to the Values field; the circles on the map for each country are now relative in size to the LineTotal for orders shipped to that country.



Step 2: Interact with your report visuals to analyze further

Power BI Desktop lets you interact with visuals that cross-highlight and filter each other to uncover further trends.

1. Click on the light blue circle centered in Canada. Note how the other visuals are filtered to show Stock (ShipCountry) and Total Orders (LineTotal) just for Canada.



PRACTICAL NO: 9

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

STAGING AREA

It is a storage for the sources data used before the ETL process. The source data is preloaded into the staging area which is a separate database with a similar table structure to that of the original data source. Usually no manipulation is done to the data while the data is stored in the staging area. It acts as a snapshot of the original data source.

PREPARING THE STAGING AREA FOR THE DATA WAREHOUSE

Following are the steps involved in importing a CSV flat file into an SQL Server database as the preparation of the staging area.

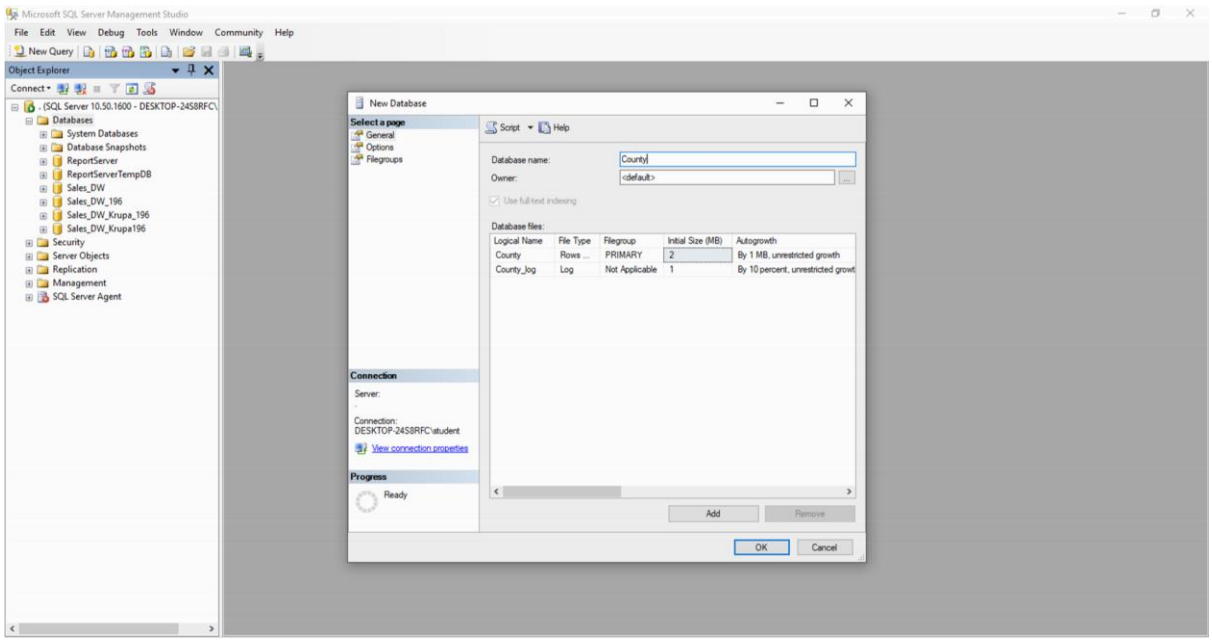
1. Download or obtain the source data set in CSV format

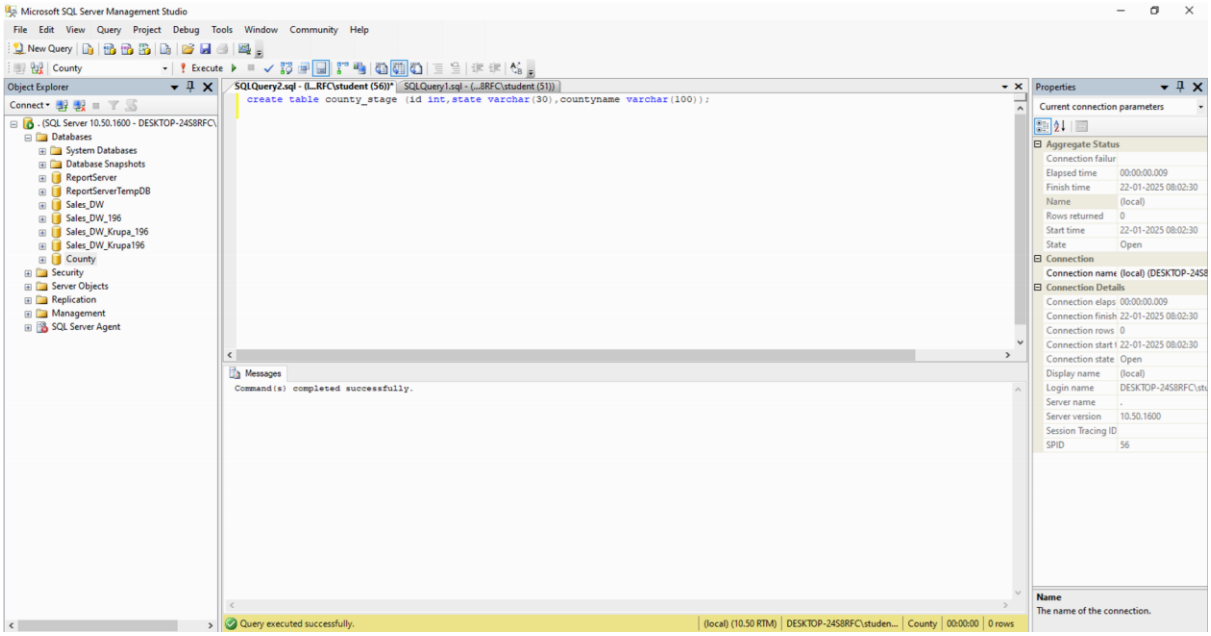
2. If the dataset is in the form of a denormalized table you can remove any unwanted columns or rows if applicable. It will speed up the load and also simplify the ETL process later.

3. Launch the SQL Server Management Studio and connect to the desired database engine. Make sure you have an account with privileges to create databases when connecting

4. Navigate to the Object Explorer and select Databases, right click and Select New Database... and create the database as required to store the staging data

5. Select the newly created database, right check and select New Query and required table schema to load the source data.

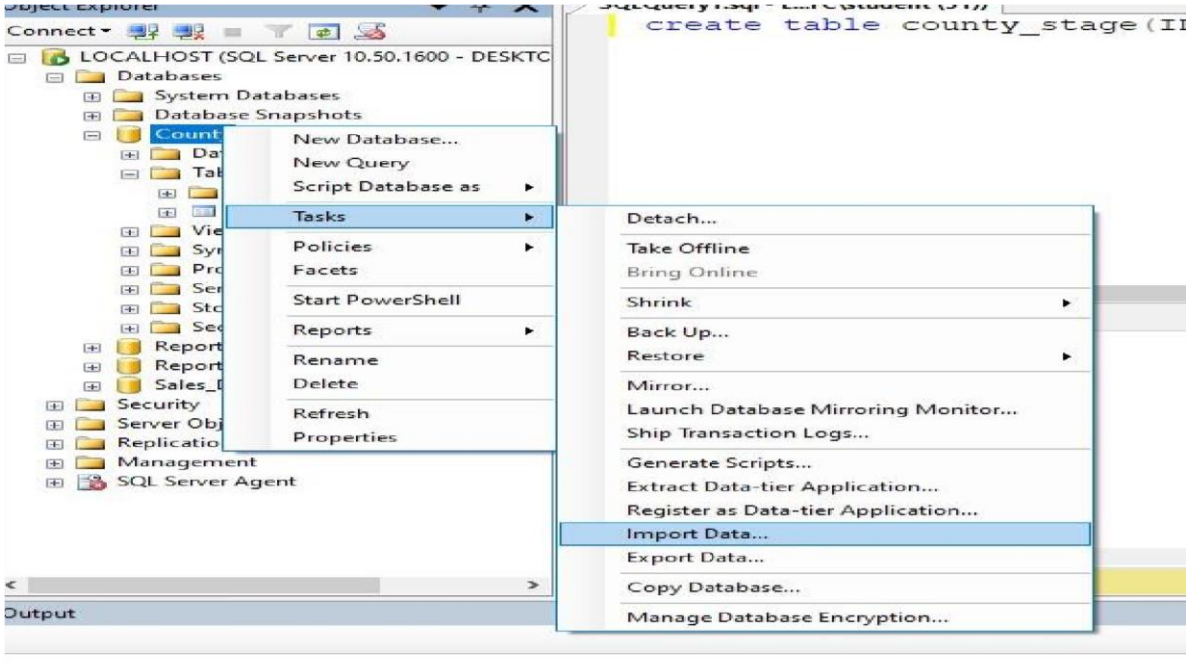




To create a single staging table called county stage using the below script and execute it by pressing F5 key.

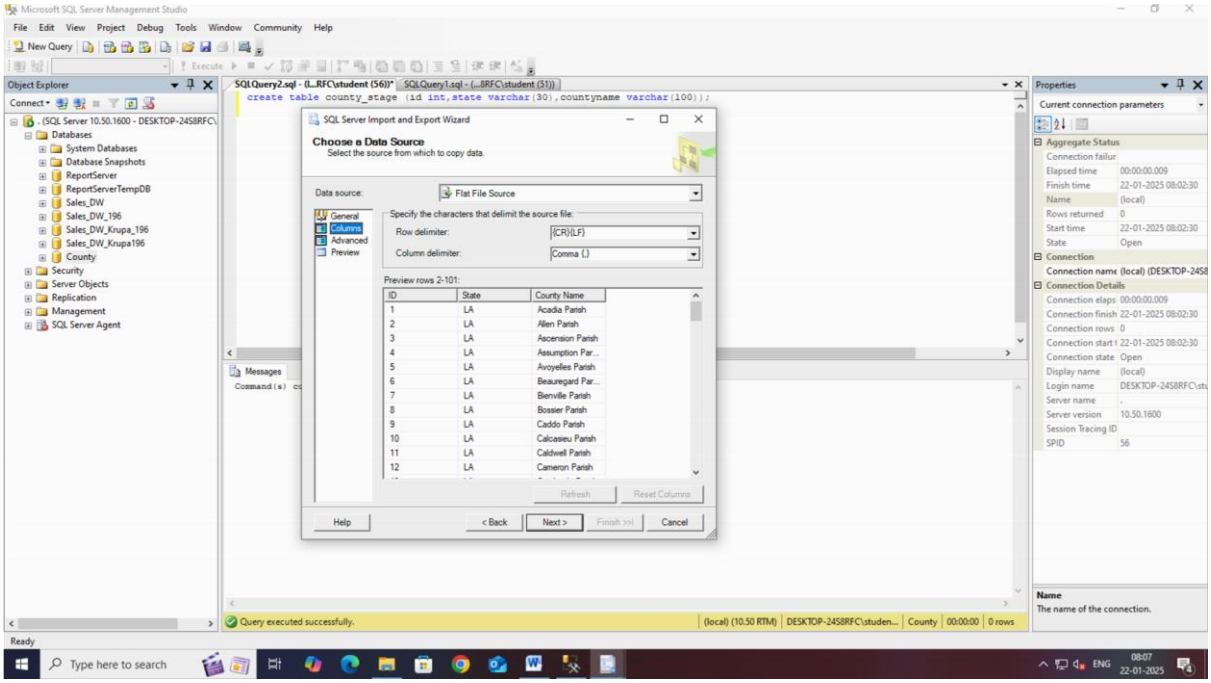
create table county_stage(id int state varchar(30) countyname varchar 100))

1. After the staging area database was created the loading of the data needs to be done For this the Import function of the SQL Server Management Studio was used, Following are the steps.
2. Right click on the created staging database, select Tasks and click on Import Data...



Click on Next

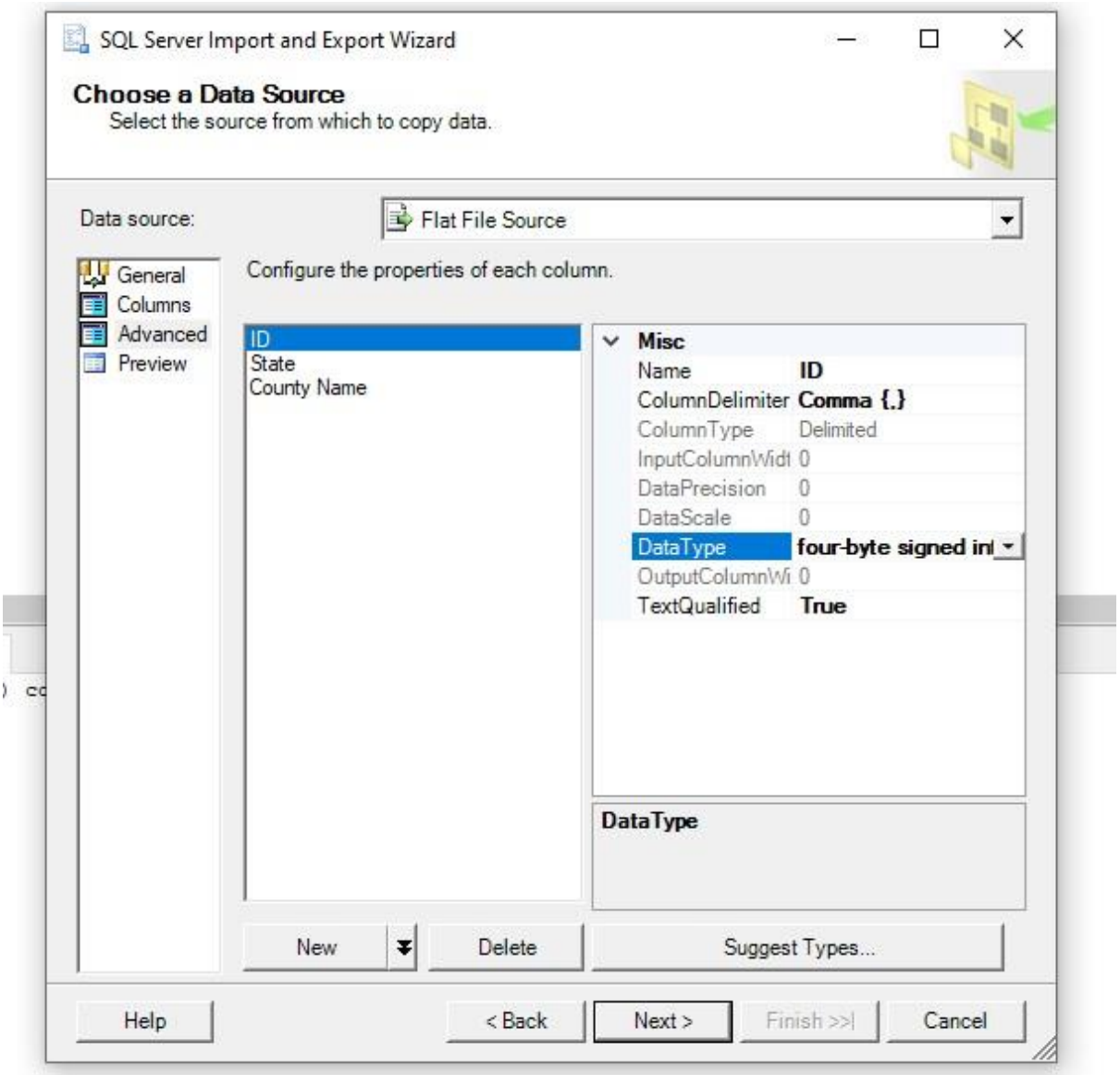
In Choose data source as **Flat File Source** and browse your CSV file and check(√) the columns in first data row



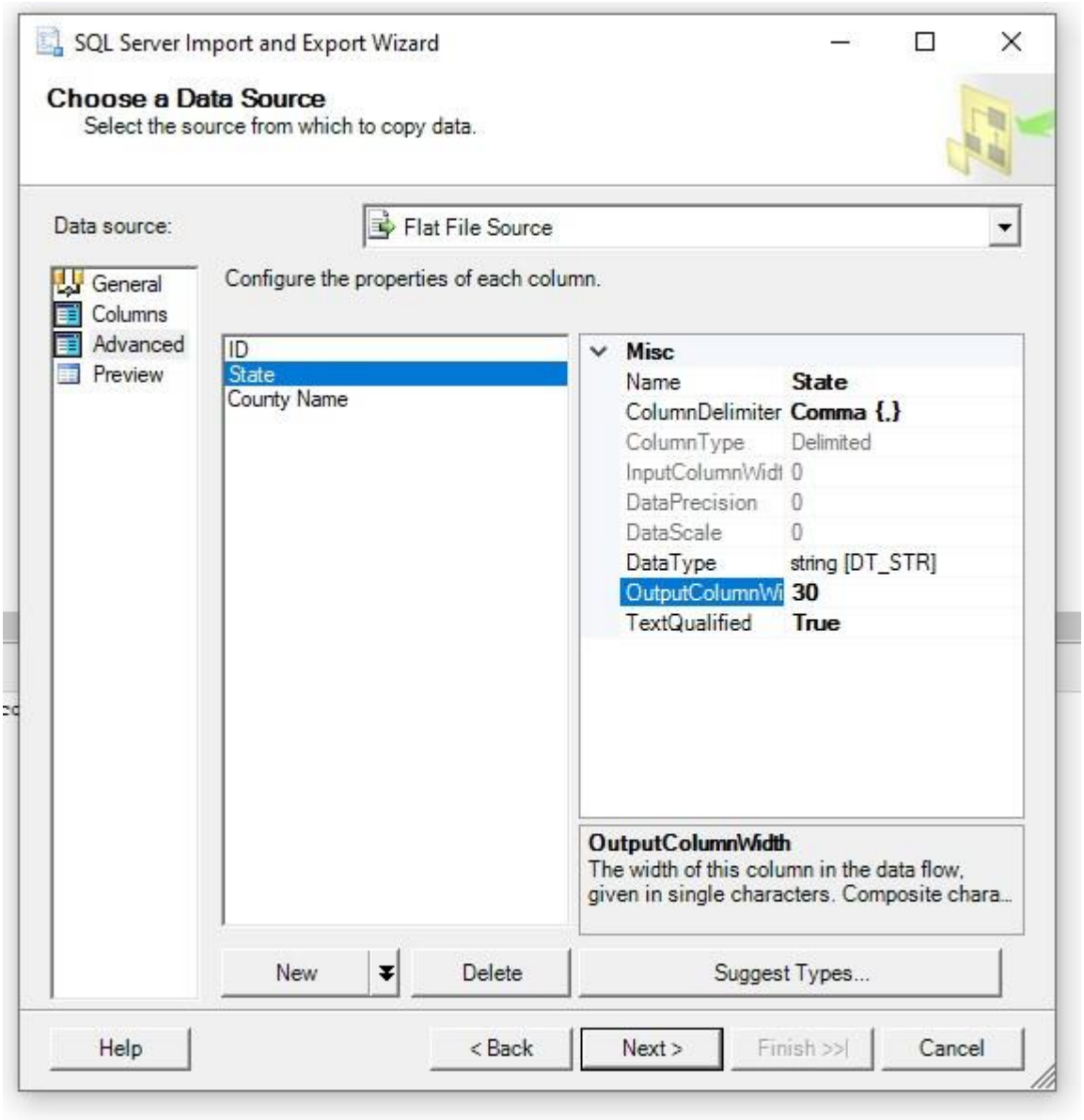
Then click on **Columns tab** and Set the proper delimiters and check the Columns tab to view if the configuration is working if not set the proper delimiters to get the desired column and row organization

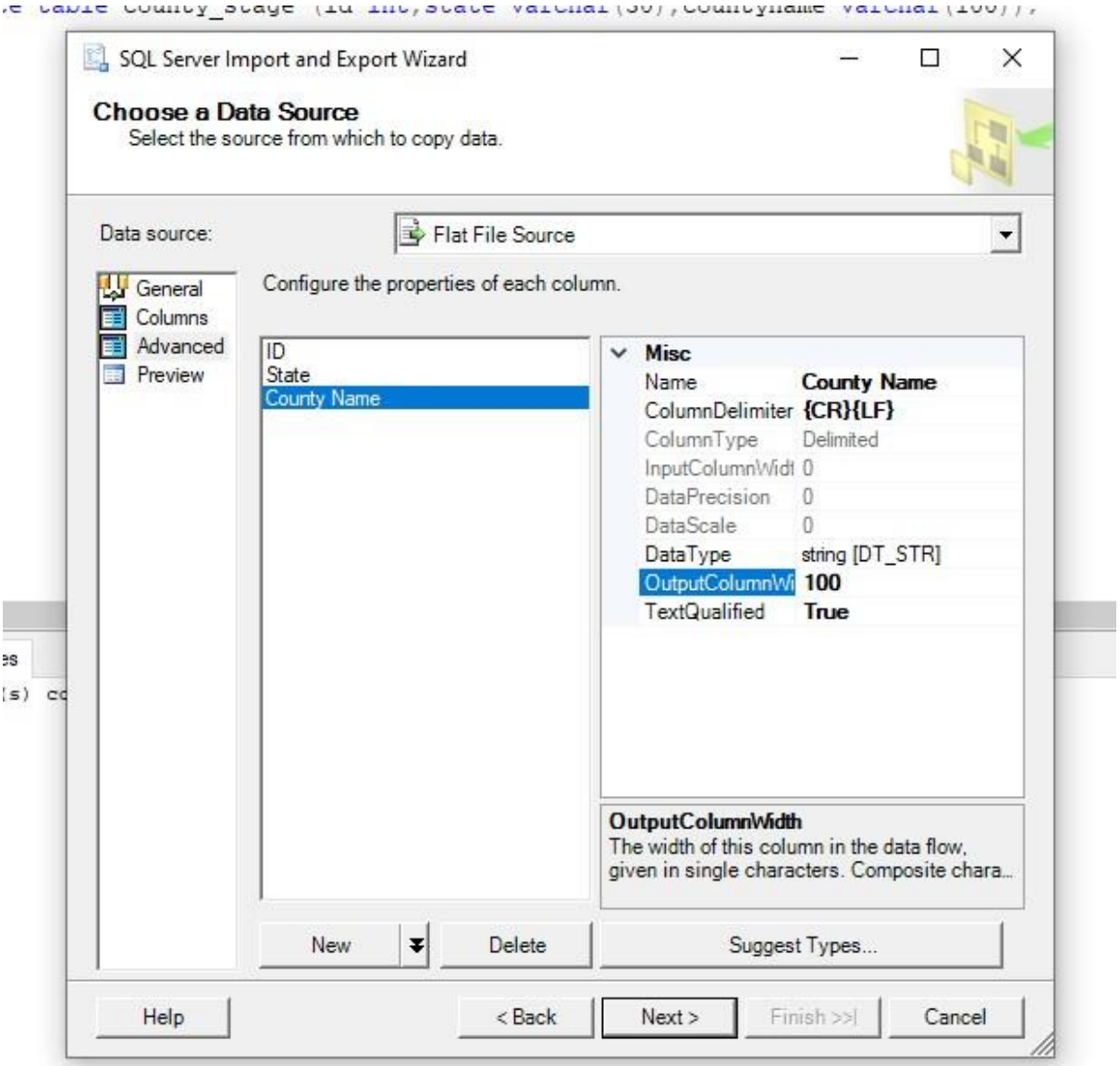
Next Move to **Advanced Tab** and configure properties of each column such as for ID set data type as four byte signed integer, State set data type as string and OutputColumn Width as 30 and for County Name type as string and OutputColumn Width as 100.

This is a very critical process, it is important to match the data types of the created staging table and the import data types. If a mismatch occurs the import process will fail.



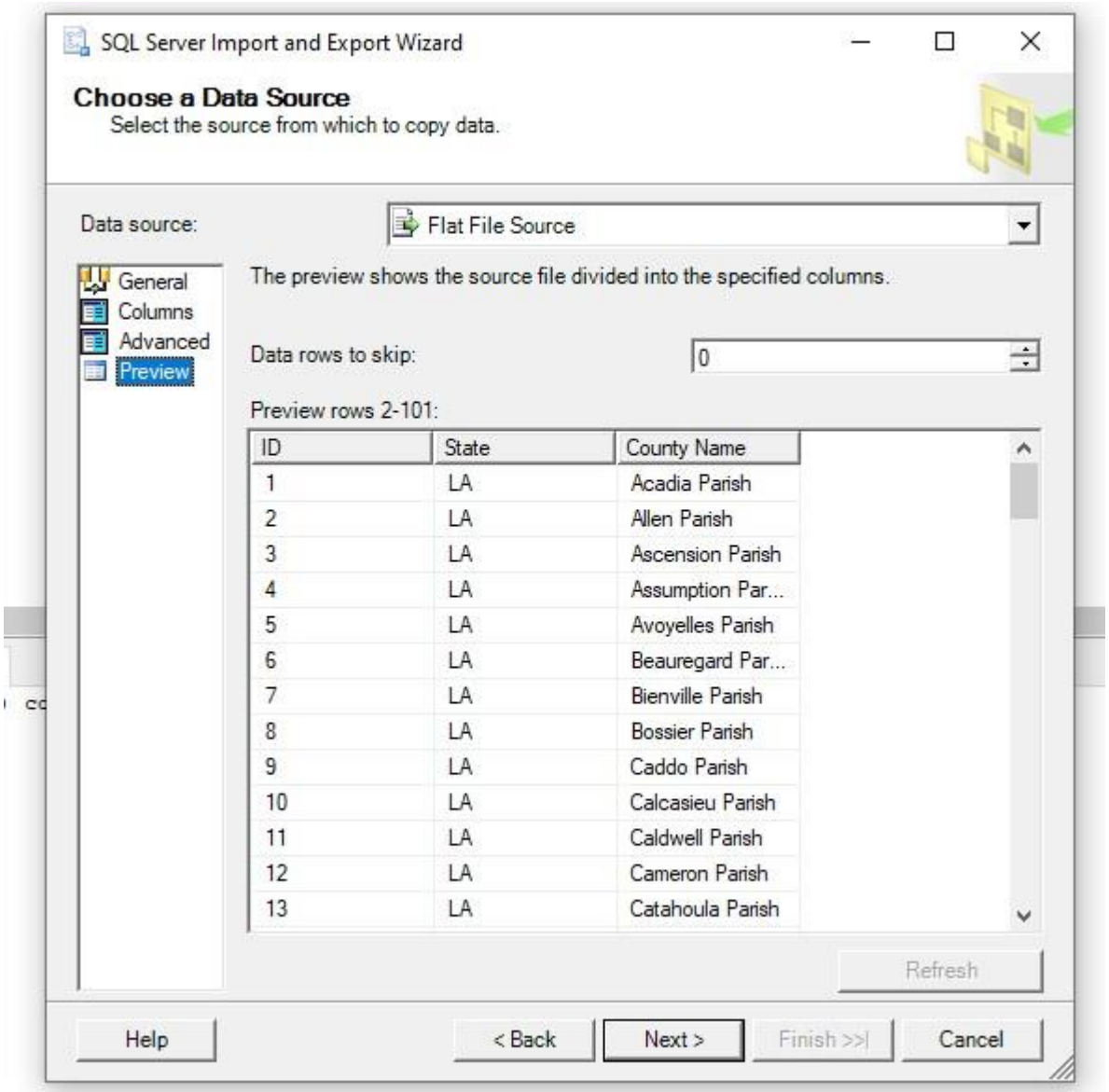
Move to the Preview tab to get a limited preview of the data to be important under the given columns. Click Next.


SQL Server Import and Export Wizard
Choose a Data Source
Select the source from which to copy data.
Data source:
Flat File Source
General
Columns
Advanced
Preview
Configure the properties of each column.
ID
State
County Name
Misc
Name State
ColumnDelimiter Comma {,}
ColumnType Delimited
InputColumnWidt 0
DataPrecision 0
DataScale 0
DataType string [DT_STR]
OutputColumnWi 30
TextQualified True
OutputColumnWidth
The width of this column in the data flow, given in single characters. Composite chara...
New
Delete
Suggest Types...
Help
< Back
Next >
Finish >>|
Cancel

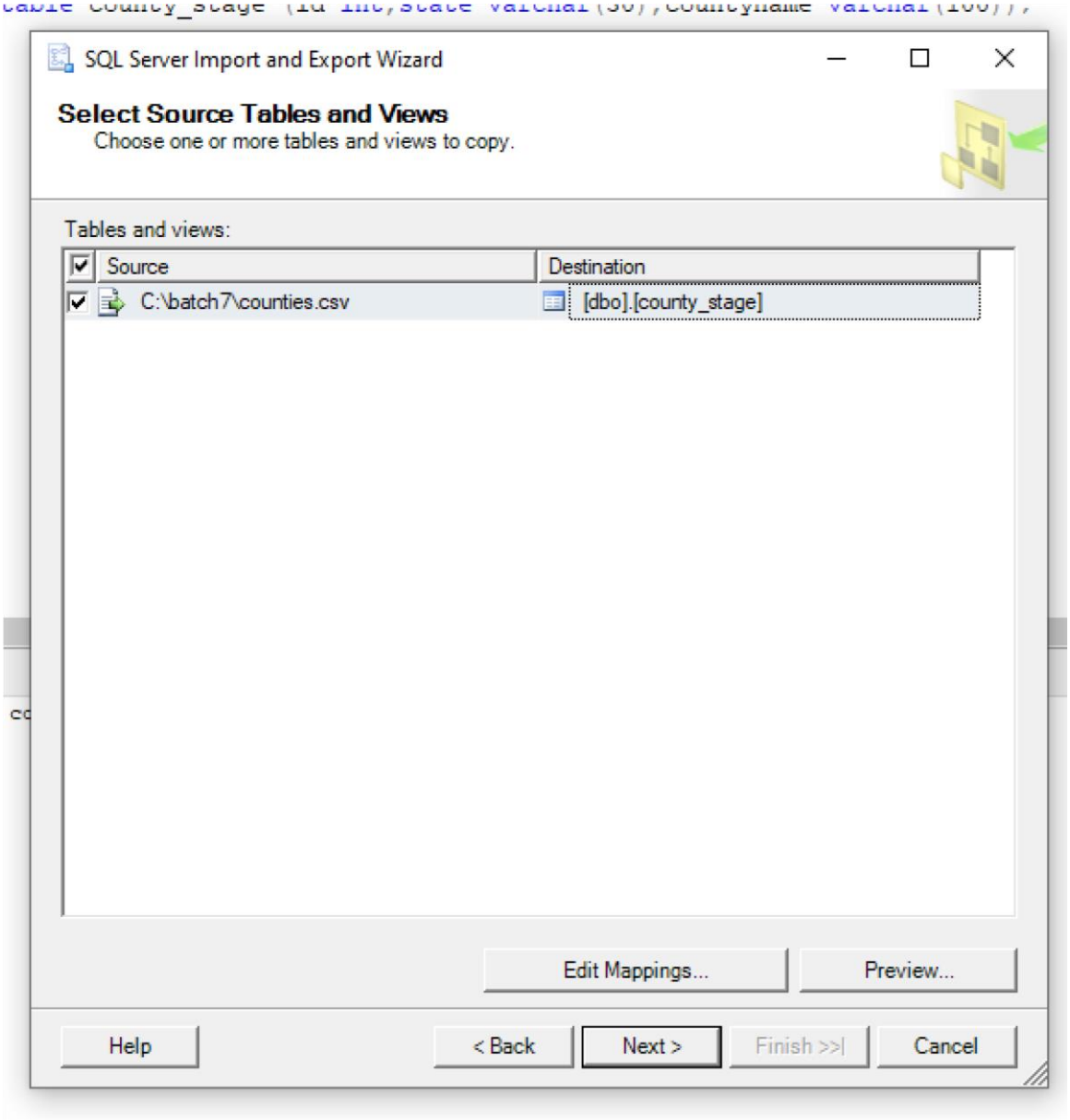


Go to preview and next

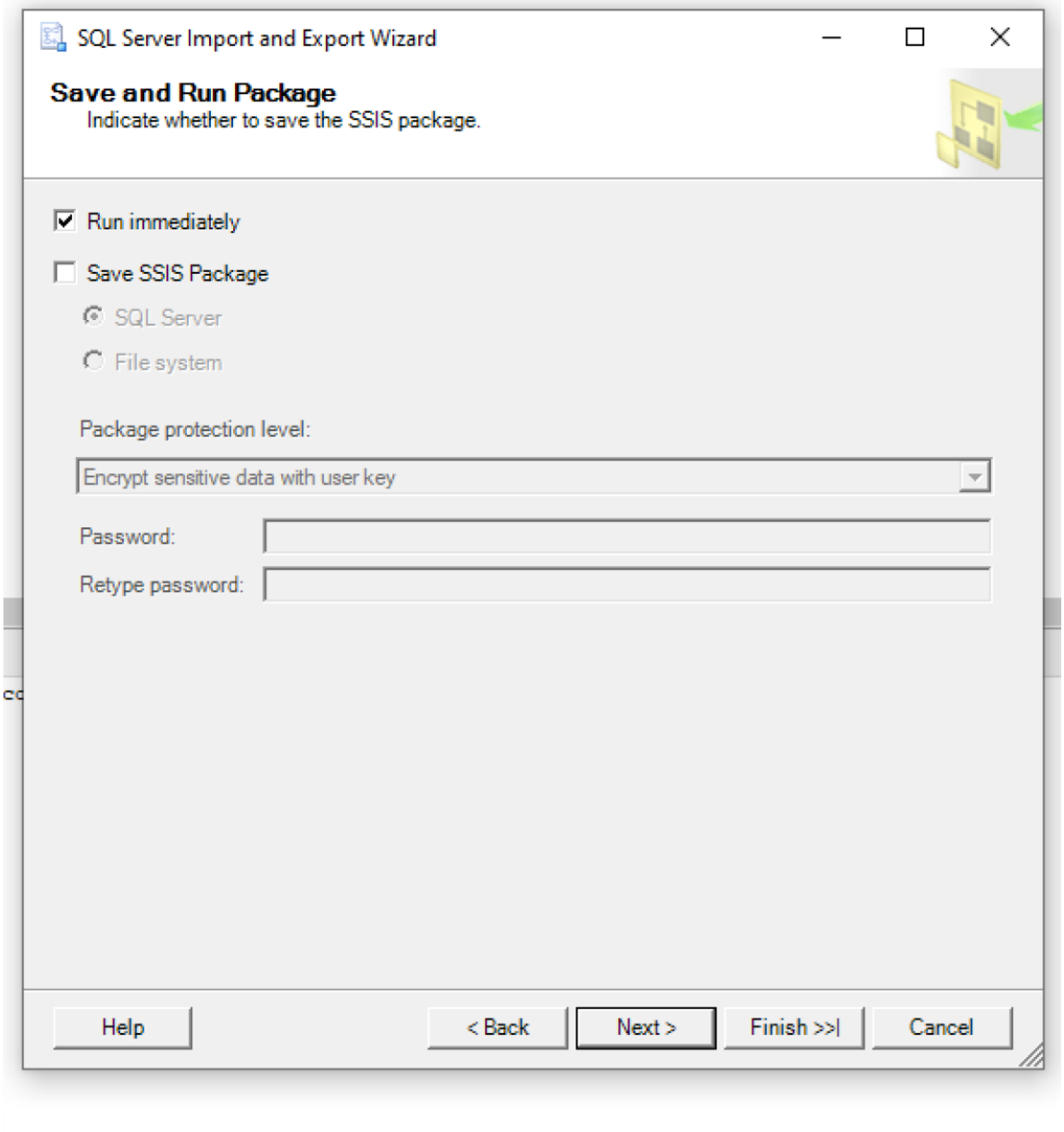
On the Choose **a Destination** page select appropriate Destination. Server Name**(localhost)** and provide a require authentication to connect (if required) and you will be able to select the staging database created above and the destination Database(County) Click Next



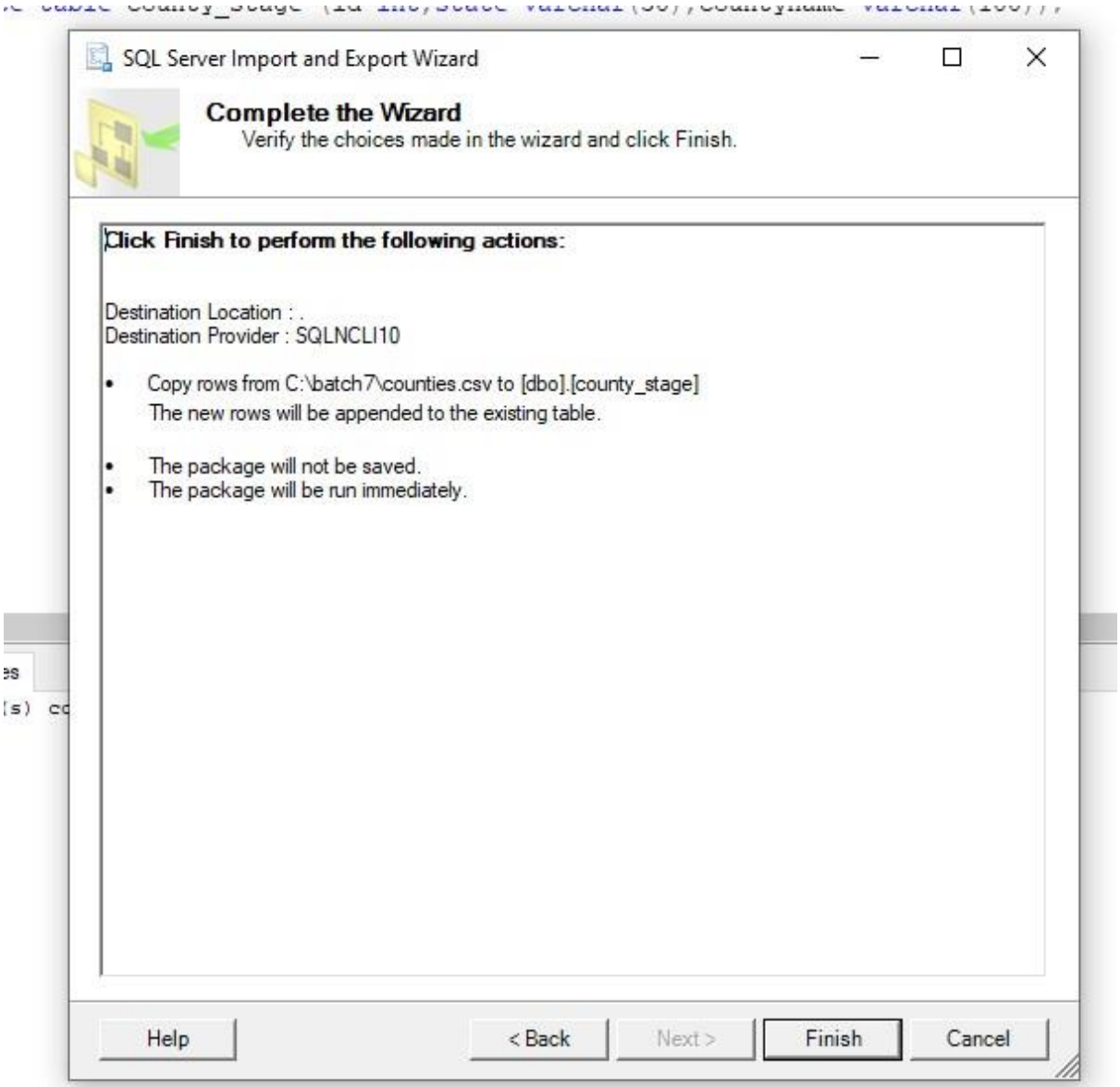
On the **Select Source Tables** and Views page, map the source to the destination tablets as required. Click Next

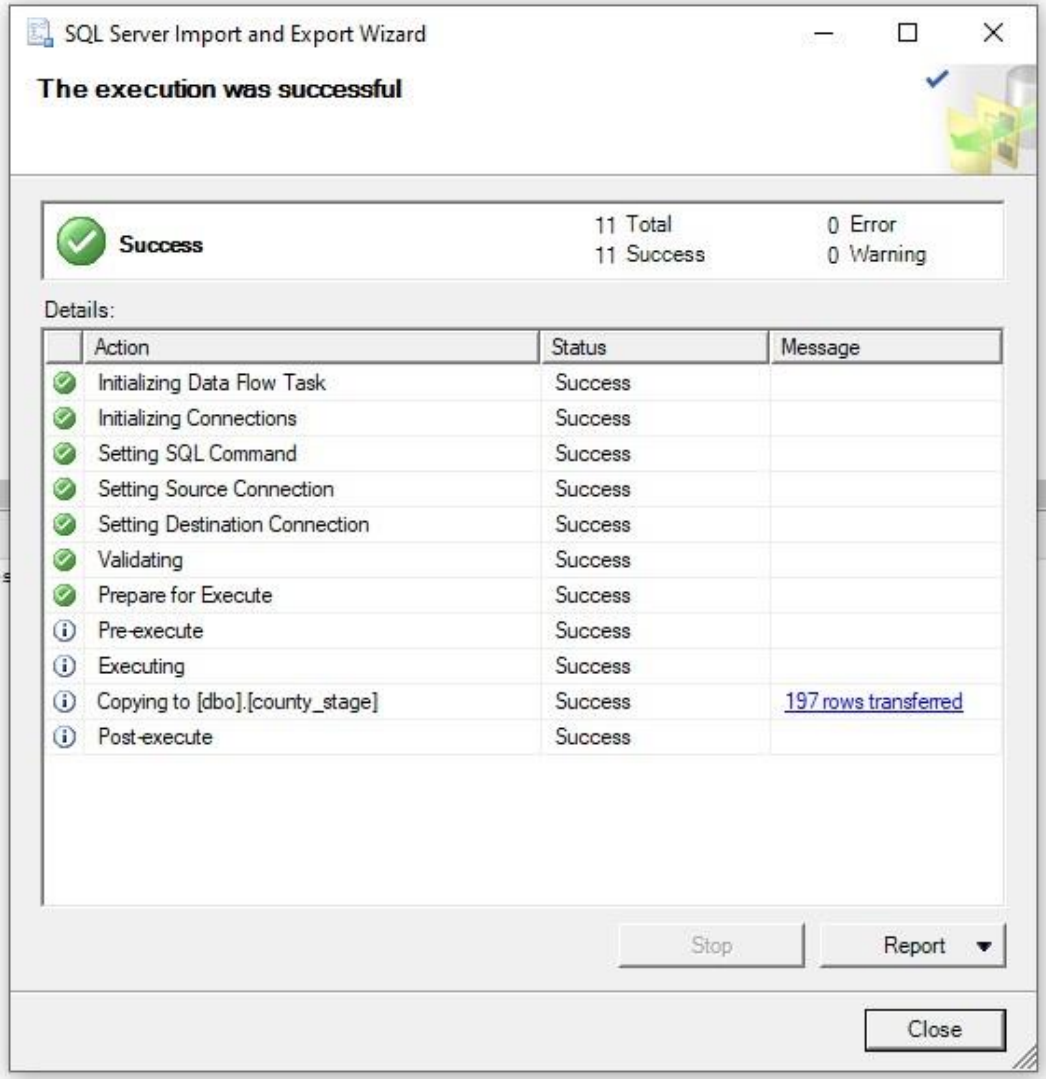


On the **Save and Run Package** page select **Run Immediately** to import the data after configuring the wizard. If you need to connect with a SSIS package you can configure it here. Click **Next**.



On the **Complete the Wizard** Page review the configurations.


SQL Server Import and Export Wizard
Complete the Wizard
Verify the choices made in the wizard and click Finish.
Click Finish to perform the following actions:
Destination Location : .
Destination Provider : SQLNCLI10
Copy rows from C:\batch7\counties.csv to [dbo].[county_stage]
The new rows will be appended to the existing table.
The package will not be saved.
The package will be run immediately.
Help
< Back
Next >
Finish
Cancel


SQL Server Import and Export Wizard
The execution was successful
Success
11 Total
11 Success
0 Error
0 Warning
Details:
Action
Status
Message
Initializing Data Flow Task
Success
Initializing Connections
Success
Setting SQL Command
Success
Setting Source Connection
Success
Setting Destination Connection
Success
Validating
Success
Prepare for Execute
Success
Pre-execute
Success
Executing
Success
Copying to [dbo].[county_stage]
Success
197 rows transferred
Post-execute
Success
Stop
Report
Close

PRACTICAL NO: 10

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

Creating Data Warehouse

Let us execute our T-SQL Script to create data warehouse with fact tables, dimensions and populate them with appropriate test values.

Download T-SQL script attached with this article for creation of Sales Data Warehouse or download from this article "Create First Data Warehouse" and run it in your SQL Server.

Follow the given steps to run the query in SSMS (SQL Server Management Studio).

1. Open SQL Server Management Studio 2008

2. Connect Database Engine

3. Open New Query editor

4. Open SQL Script in query editor.

5. To run the given SQL. Script, press F5

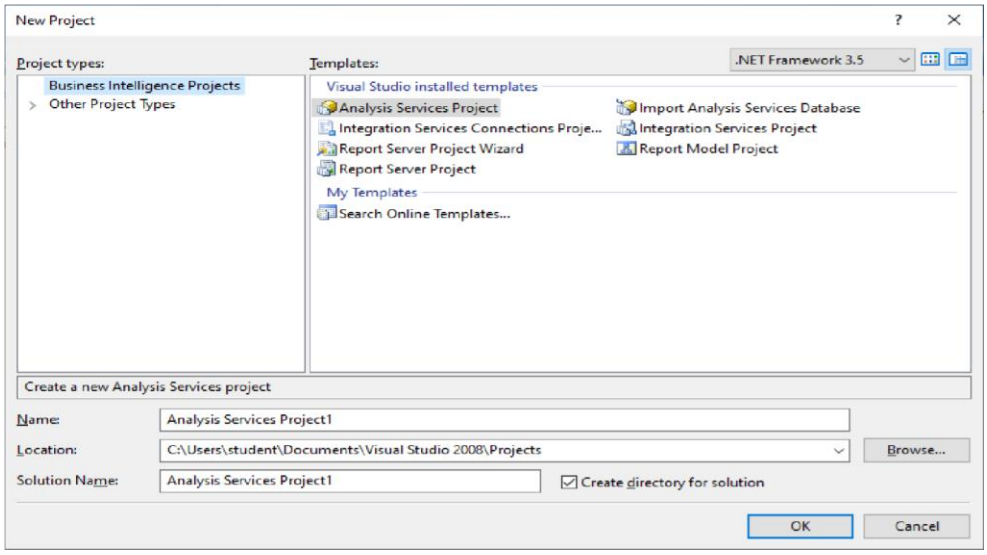
6. It will create and populate "Sales_DW" database on your SQL Server

Developing an OLAP Cube

For creation of OLAP Cube in Microsoft BIDS Environment, follow the 10 easy steps given below.

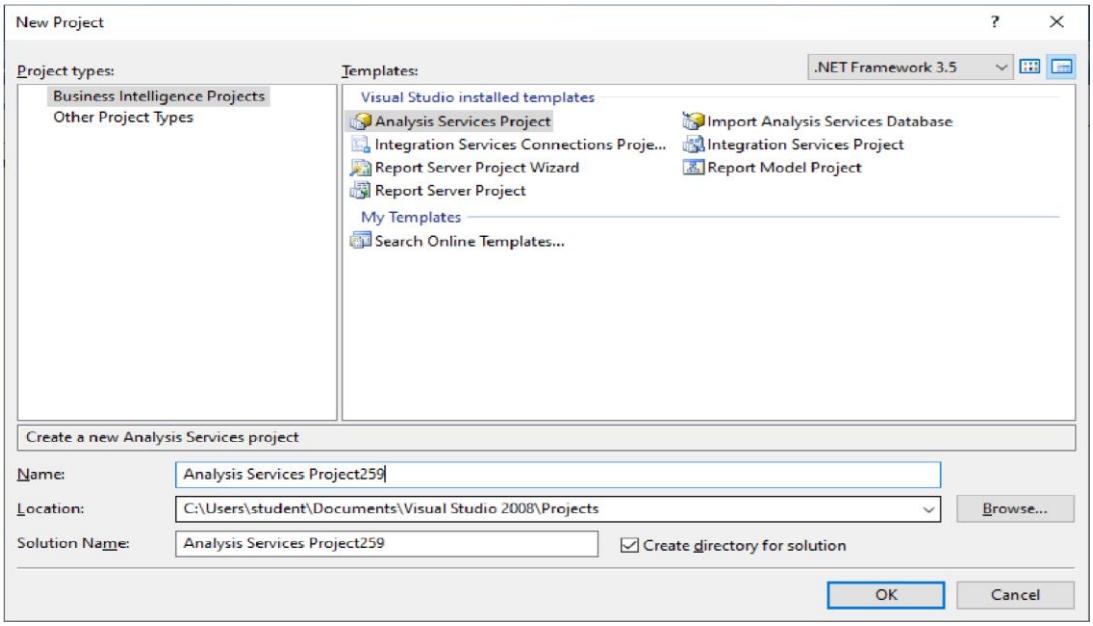
Step 1: Start BIDS Environment

Click on Start Menu -> Microsoft SQL Server 2008 R2 -> Click SQL Server Business Intelligence Development Studio.



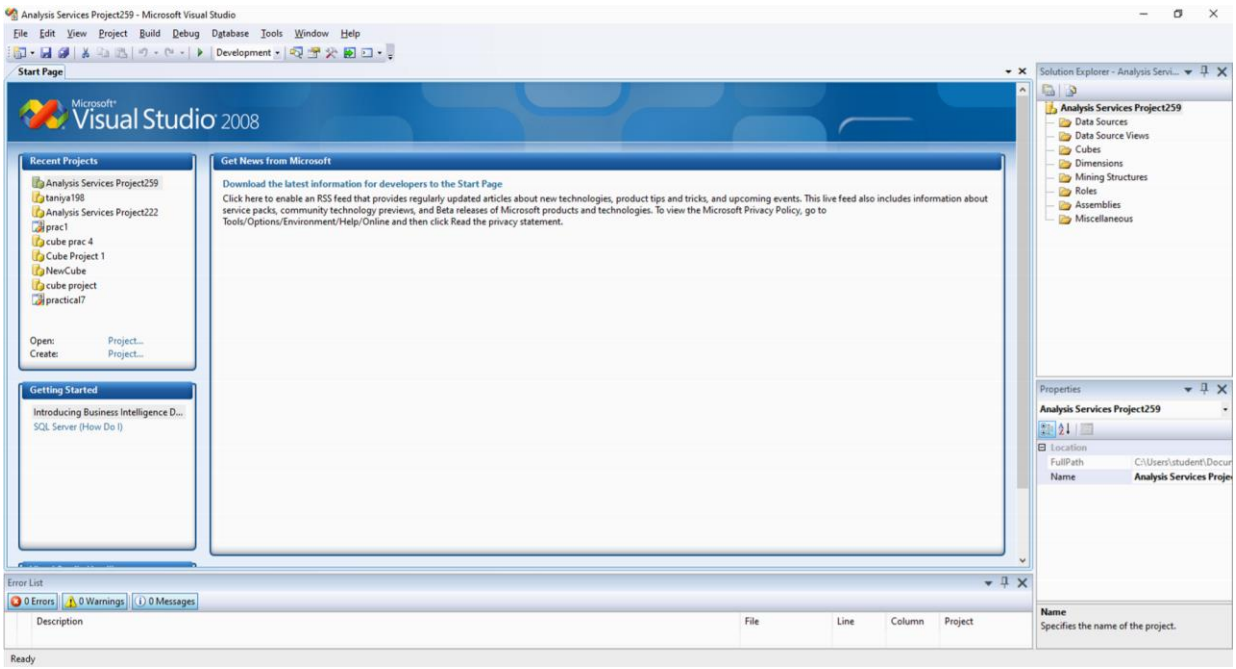
Step 2: Start Analysis Services Project

Click File -> New -> Project ->Business Intelligence Projects ->select Analysis Services Project-> Assign Project Name (cubepract4)-> Click OK

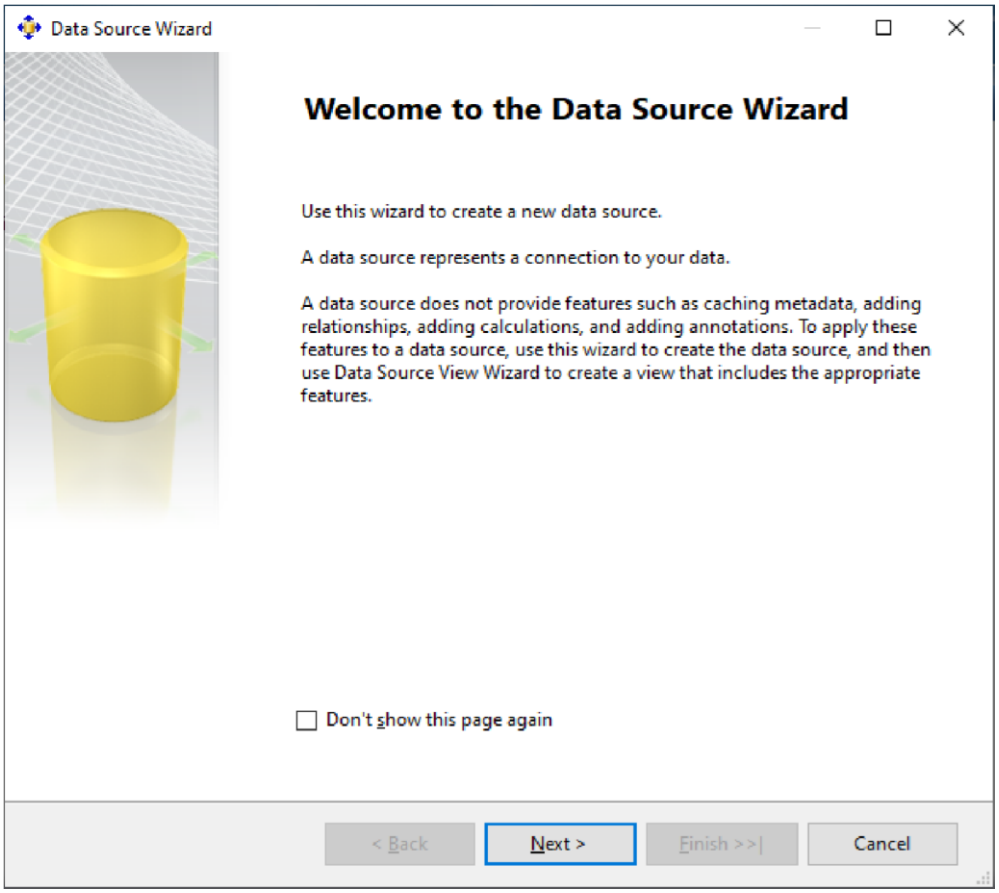


Step 3: Creating New Data Source

3.1 In Solution Explorer, Right Click on Data Source -> Click New Data Source



3.2 Click on Next



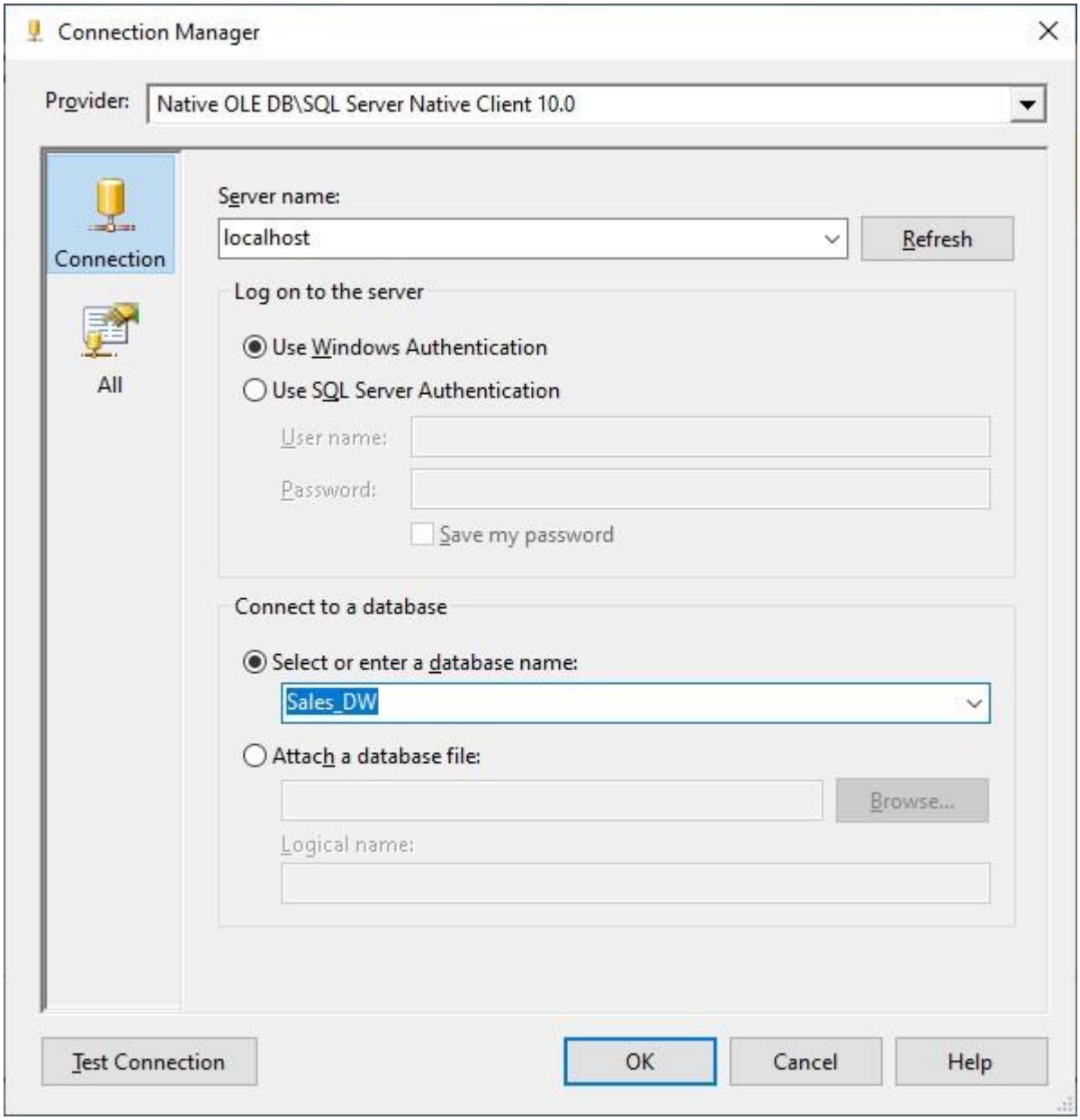
3.3 Click on New Button

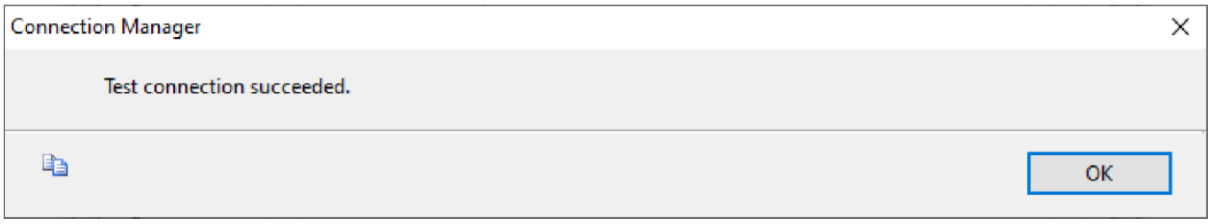
3.4 Creating New Connection

1. Specify Your SQL Server Name where your Data Warehouse was created
2. Select Radio Button according to your SQL Server Authentication mode 3.

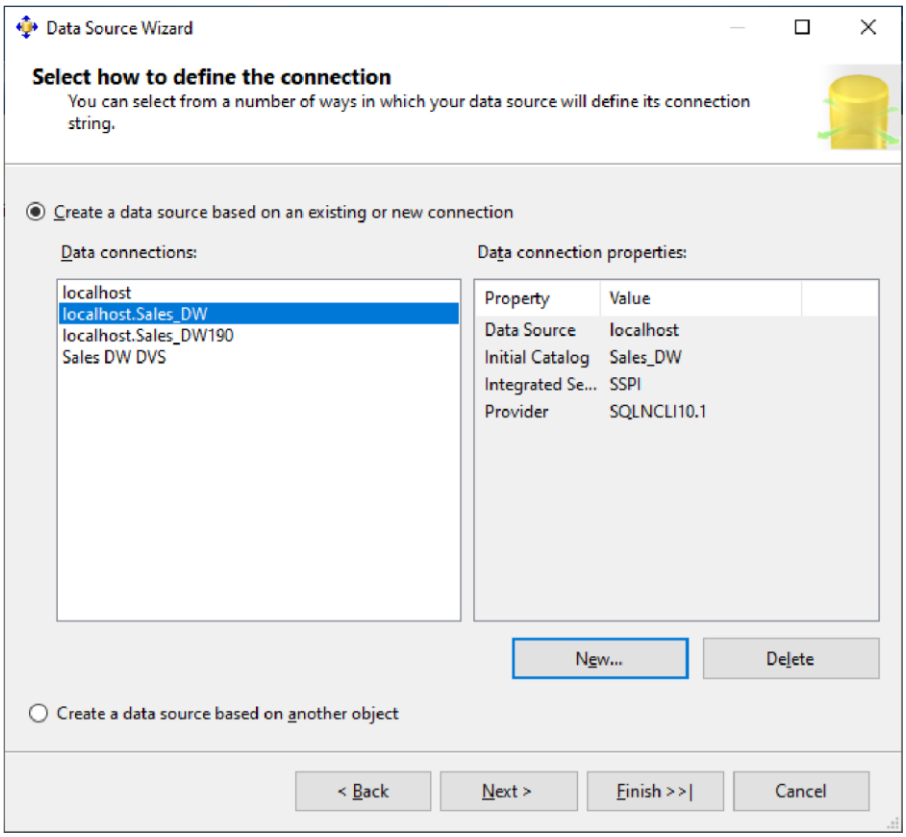
   Specify your Credentials using which you can connect to your SQL Server

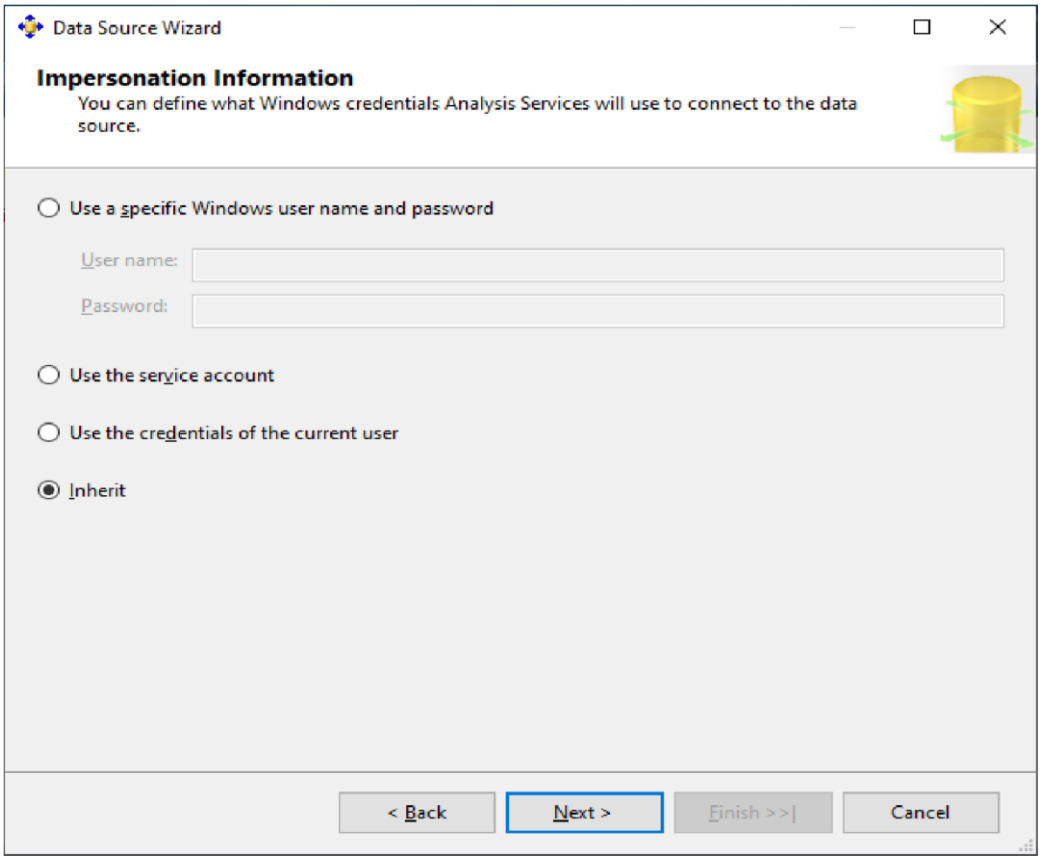
4. Select database Sales_DW.
5. Click on Test Connection and verify for its success
6. Click OK



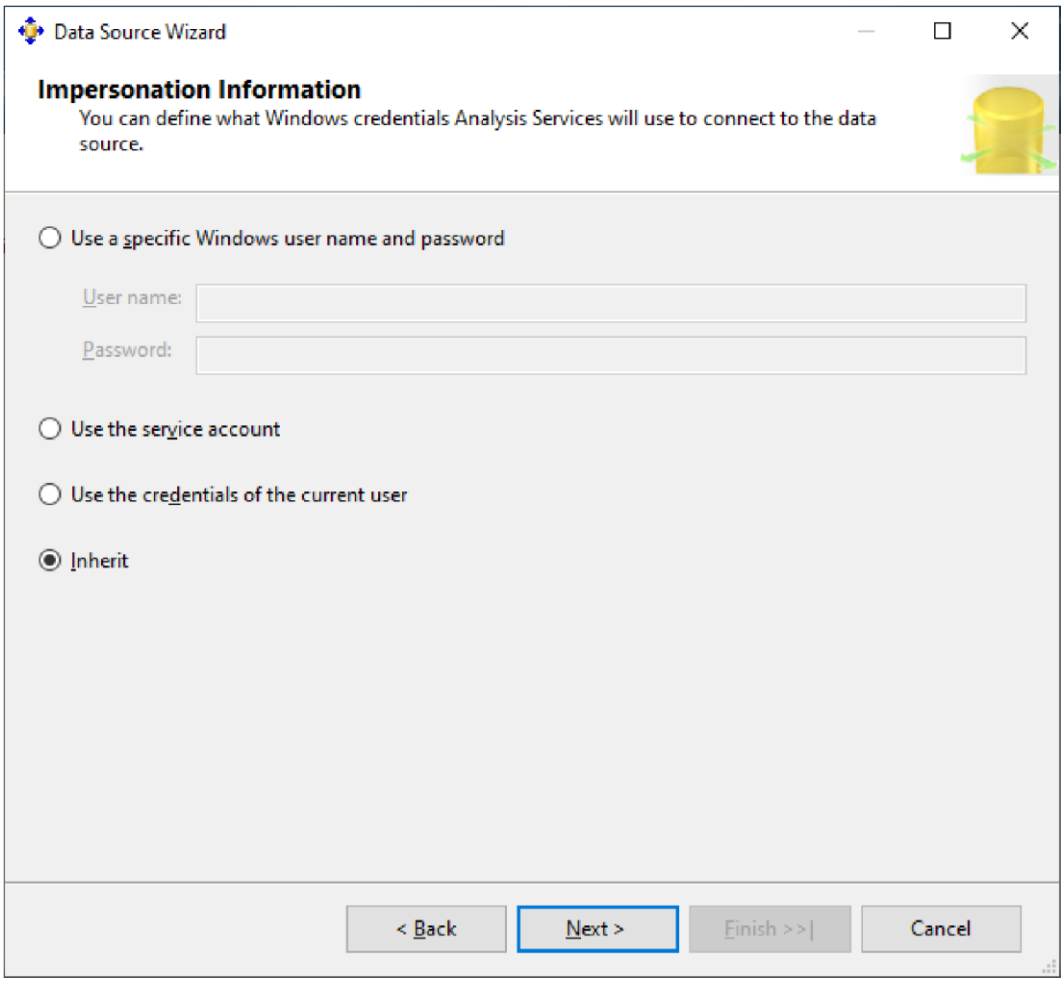


3.5 Select Connection Created in Data Connections -> Click Next


Data Source Wizard
Select how to define the connection
You can select from a number of ways in which your data source will define its connection string.
Create a data source based on an existing or new connection
Data connections:
localhost
localhost.Sales_DW
localhost.Sales_DW190
Sales DW DVS
Data connection properties:
Property
Value
Data Source
localhost
Initial Catalog
Sales_DW
Integrated Se...
SSPI
Provider
SQLNCLI10.1
New...
Delete
Create a data source based on another object
< Back
Next >
Finish >>|
Cancel

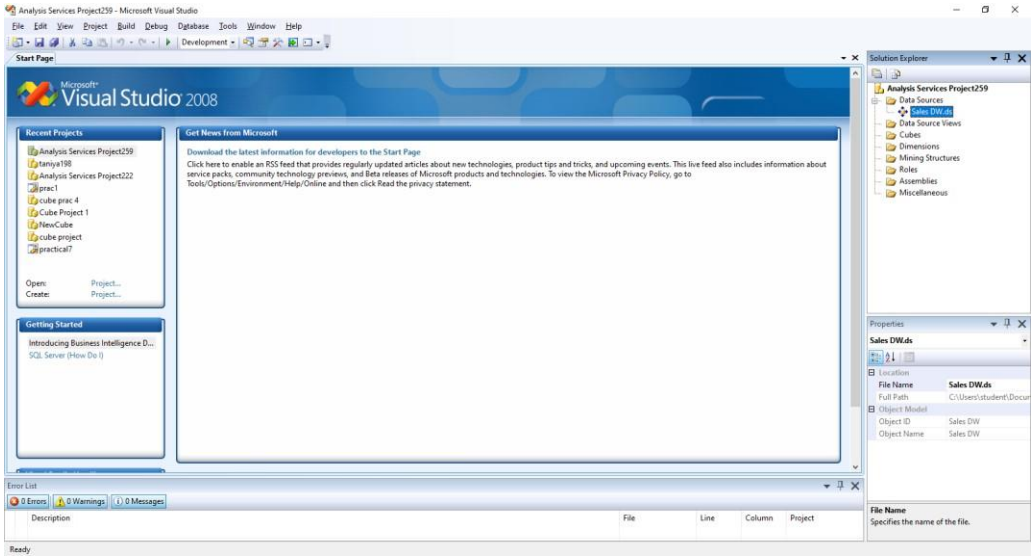

Data Source Wizard
Impersonation Information
You can define what Windows credentials Analysis Services will use to connect to the data source.
Use a specific Windows user name and password
User name:
Password:
Use the service account
Use the credentials of the current user
Inherit
< Back
Next >
Finish >>|
Cancel

3.7 Assign Data Source Name -> Click Finish



Step 4: Creating New Data Source View

4.1 In the Solution Explorer, Right Click on Data Source View -> Click on New Data Source View



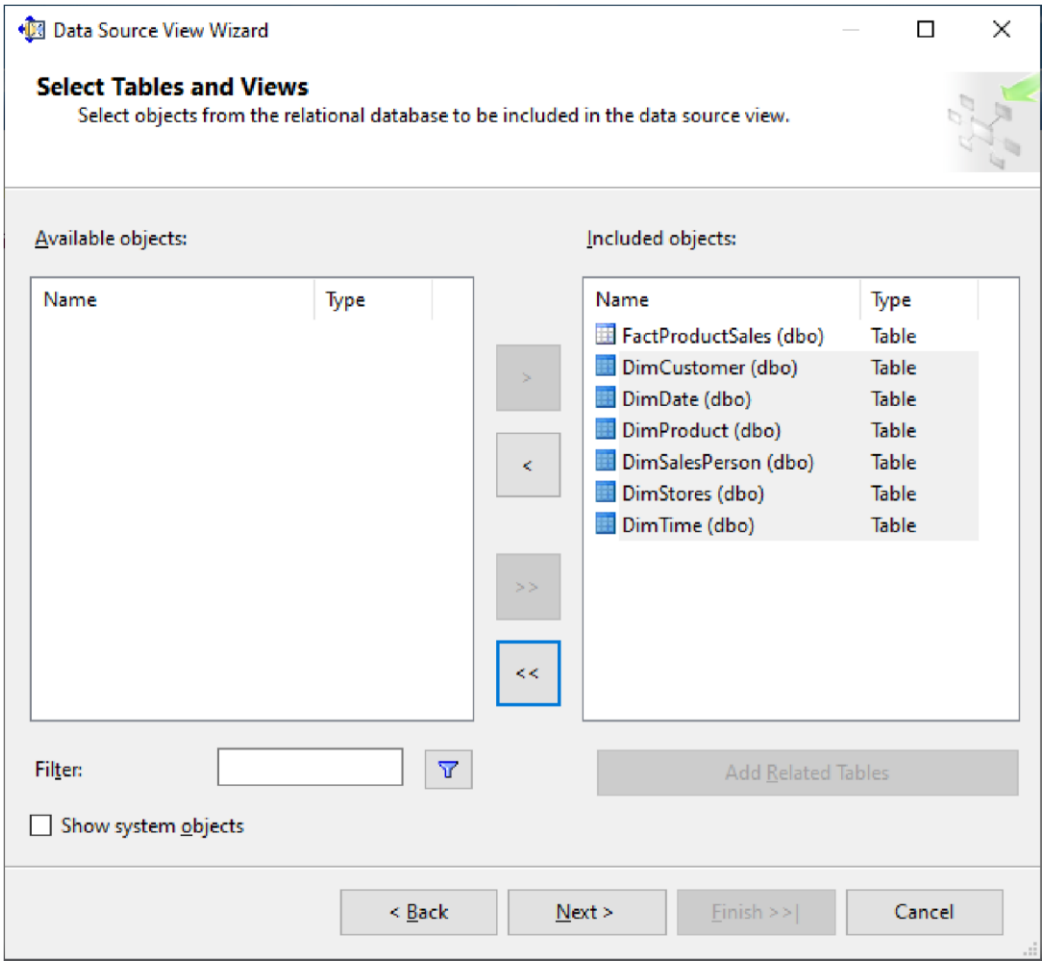
4.2 Click Next

4.3 Select Relational Data Source we have created previously (Sales_DW) -> Click Next

4.4 First move your Fact Table to the right side to include in object list

Select FactProductSales Table -> Click on Arrow Button to move the selected objects to Right Pane

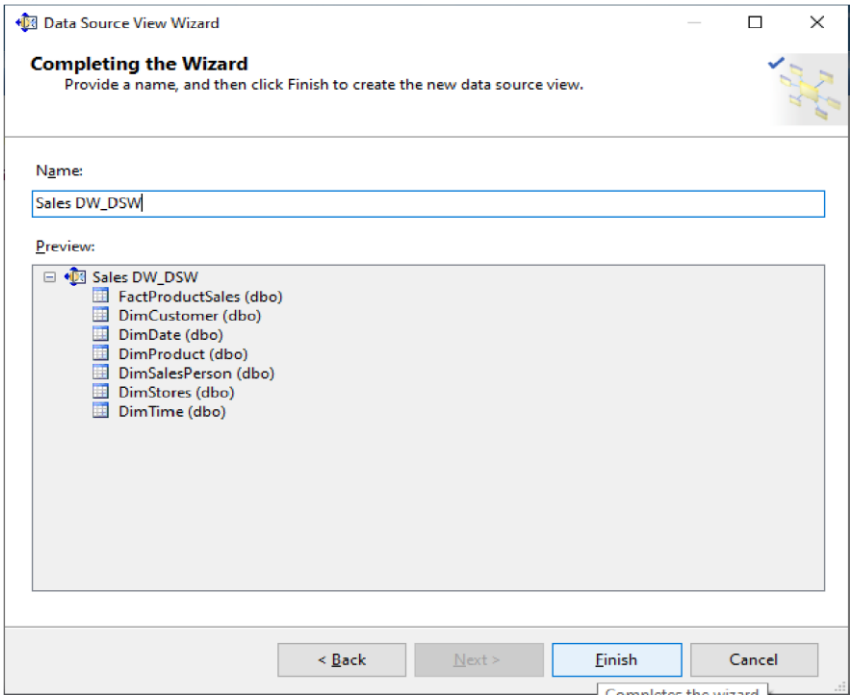
4.5 Now to add dimensions which are related to your Fact Table. Follow the given steps: Select Fact Table in Right Pane (Fact product Sales) -> Click On Add Related Tables



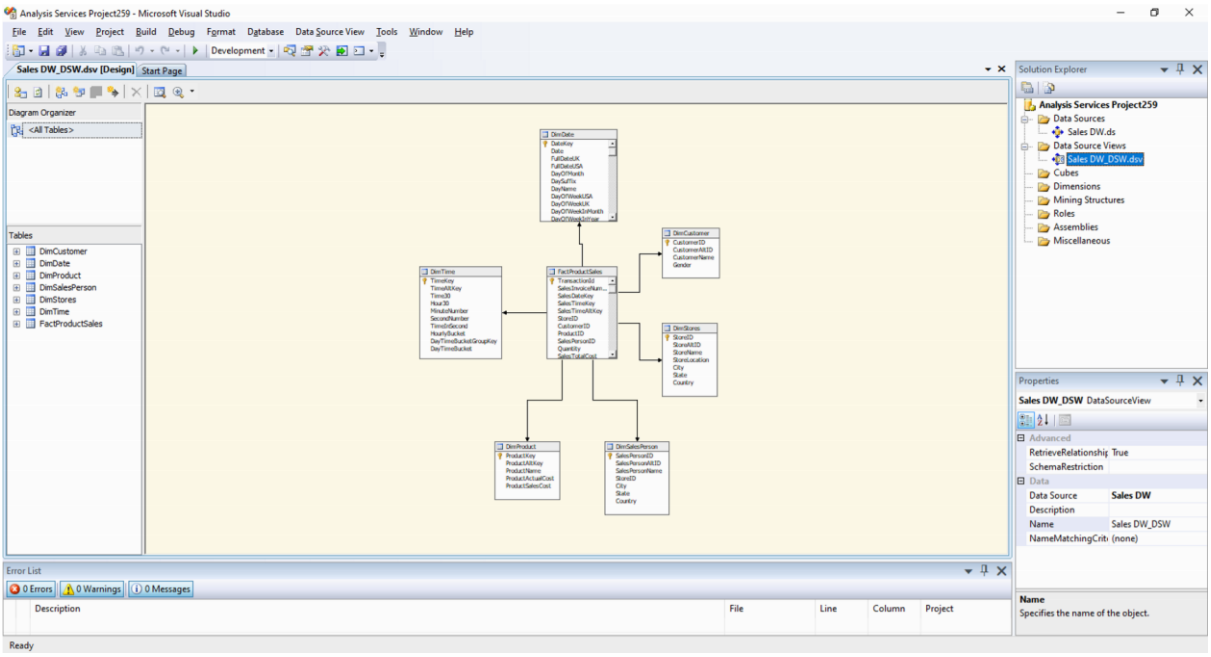
4.6 It will add all associated dimensions to your Fact table as per relationship specified in your SQL DW (Sales_DW).

Click Next.

4.7 Assign Name (SalesDW DSV) -> Click Finish



4.8 Now Data Source View is ready to use.

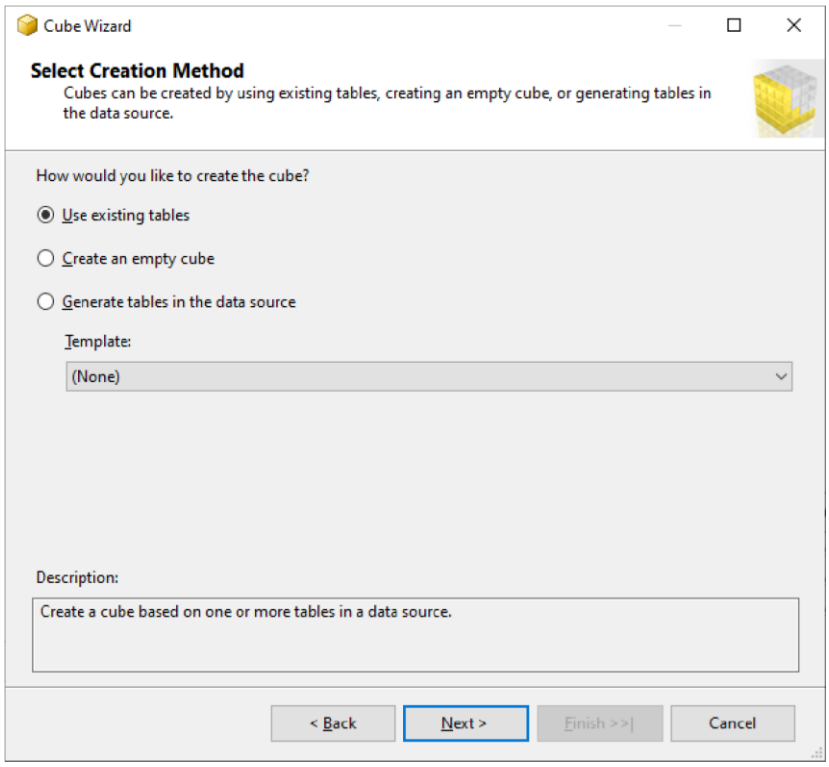


Steps 5: Creating New Cube

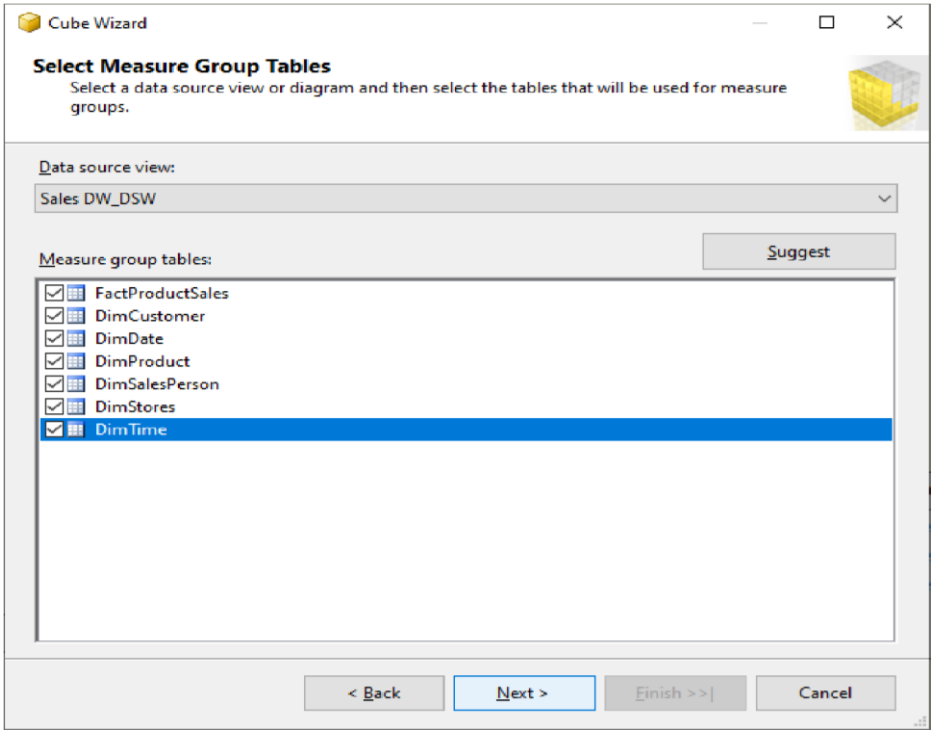
5.1 In Solution Explorer -> Right Click on Cube -> Click New Cube

5.2 Click Next

5.3 Select Option Use Existing Tables -> Click Next



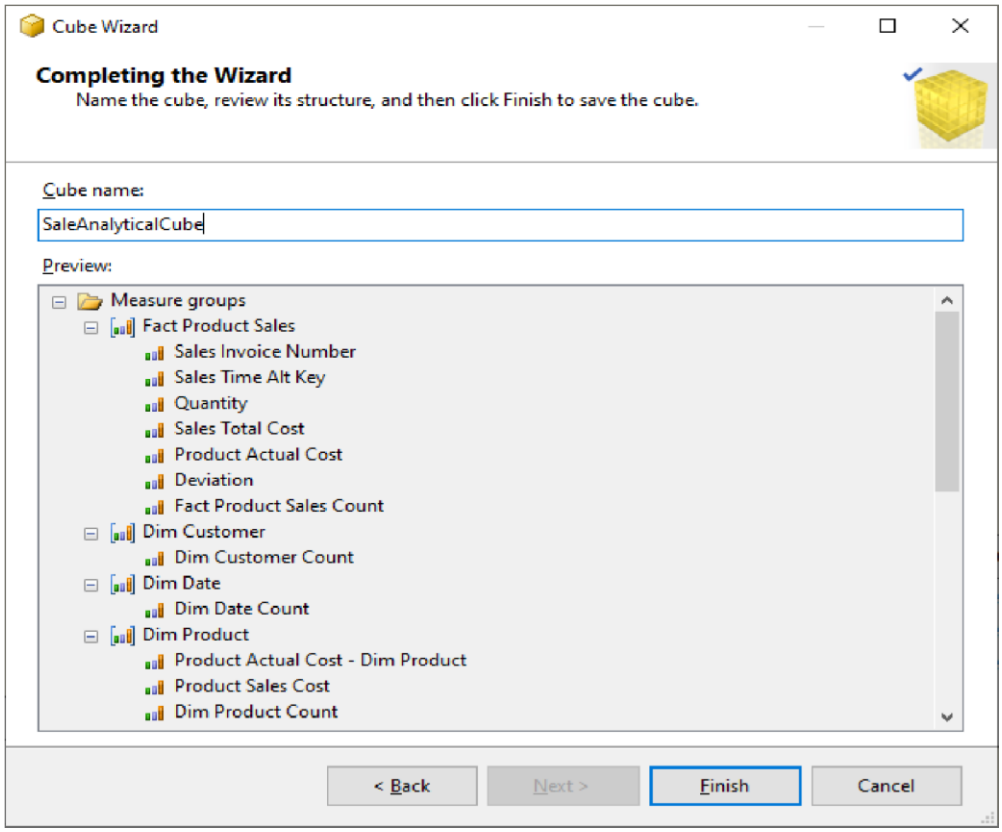
5.4 Select Fact Table Name from Measure Group Tables (FactProductSales) -> Click Next



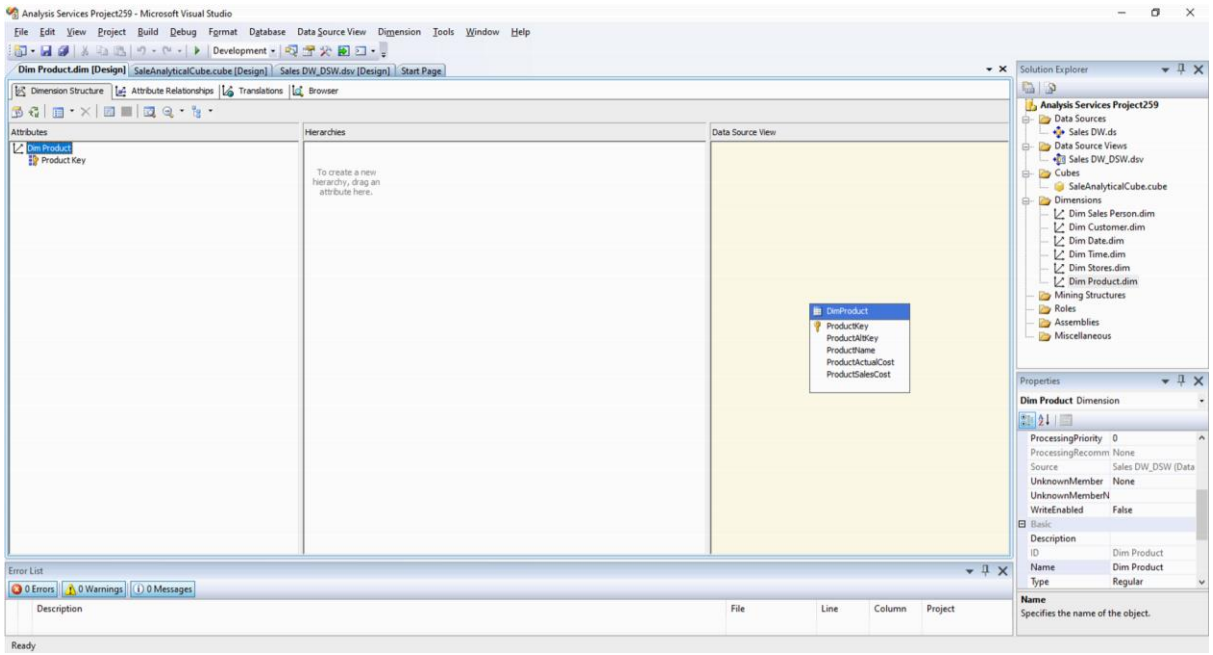
5.5 Choose Measures from the List which you want to place in your Cube -> Click Next

5.6 Select All Dimensions here which are associated with your Fact Table -> Click Next

5.7 Assign Cub Name (SalesAnalyticalCube) -> Click Finish



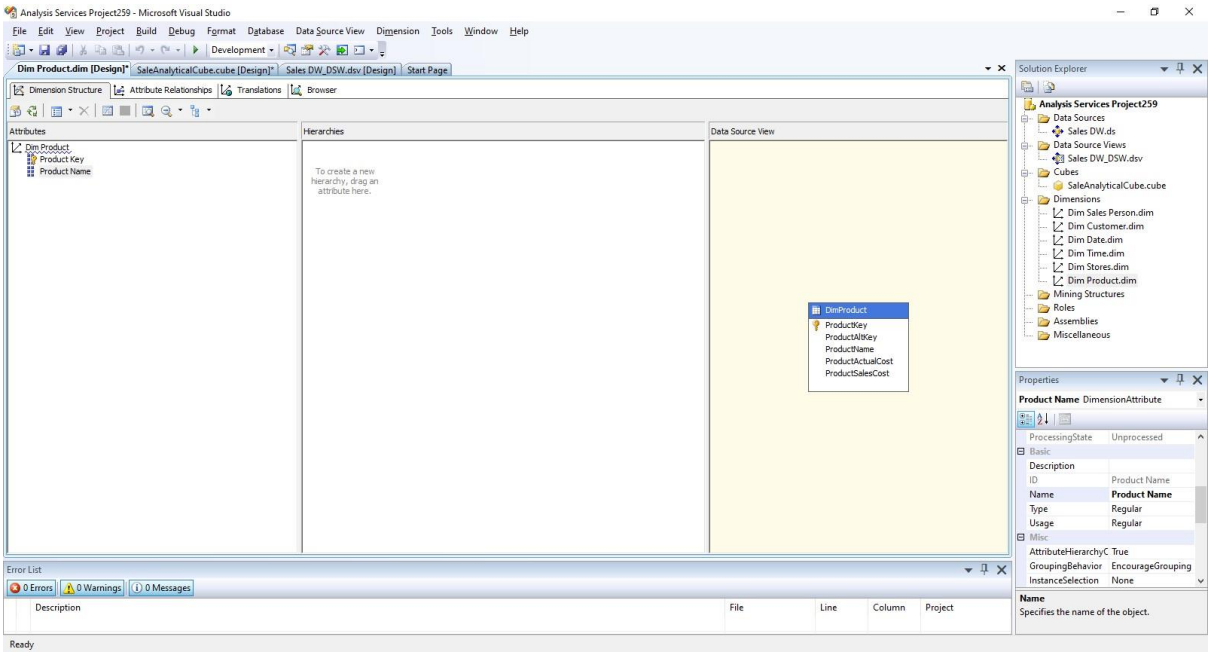
5.8 Now your Cube is ready, You can see the newly created cube and dimensions added in your solution explorer



Step 6: Dimension Modification

In Solution Explorer, Double Click on Dimension Dim Product-> Drag and Drop Product

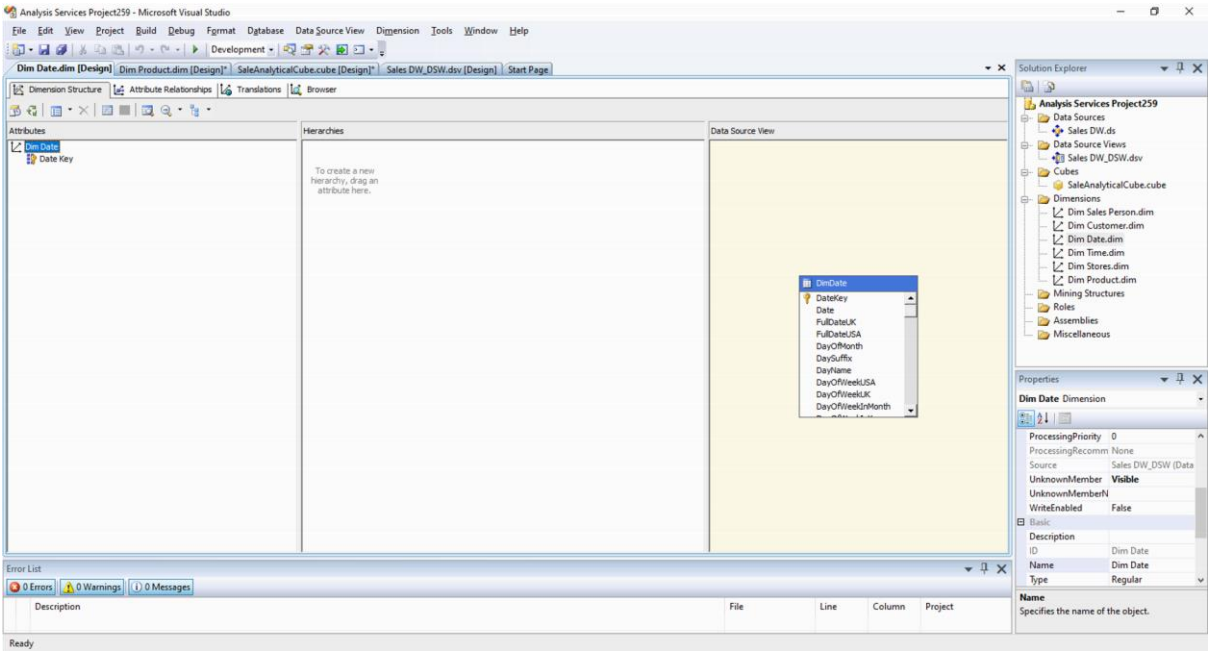
Name from Table in Data Source View and Add in Attribute Pane at left side.



Step 7: Creating Attribute Hierarchy In Date Dimension

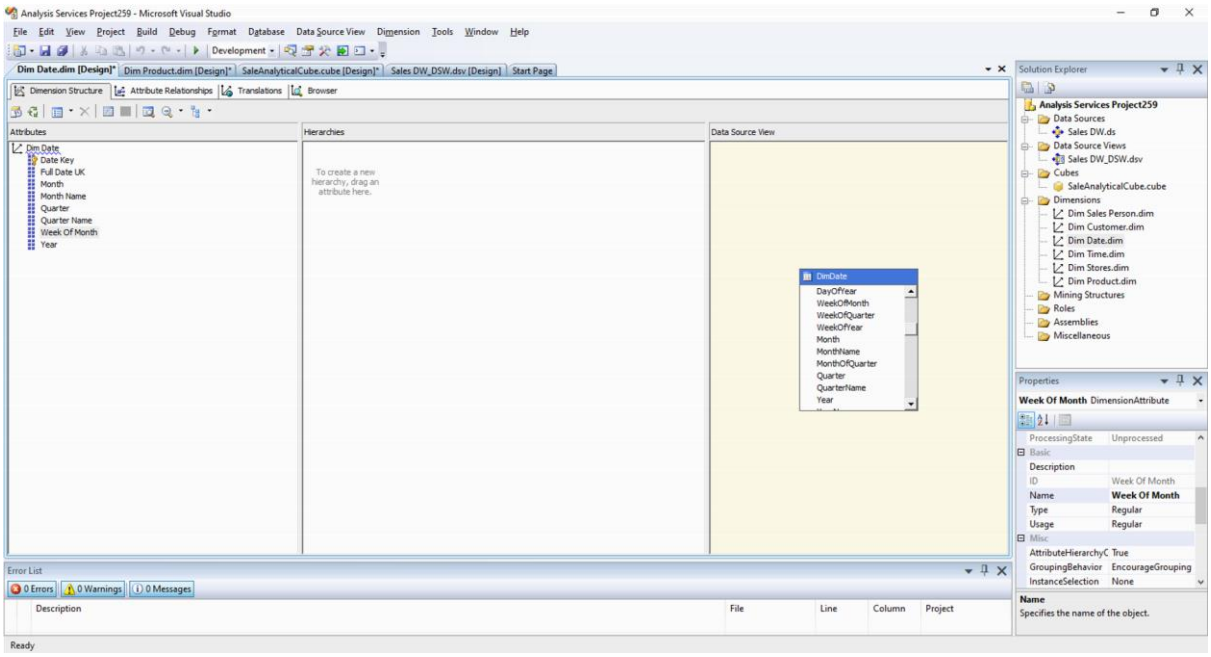
Double click On Dim Date dimension -> Drag and Drop Fields from Table shown in Data Source View to Attributes-> Drag and Drop attributes from leftmost pane of attributes to middle pane of Hierarchy.

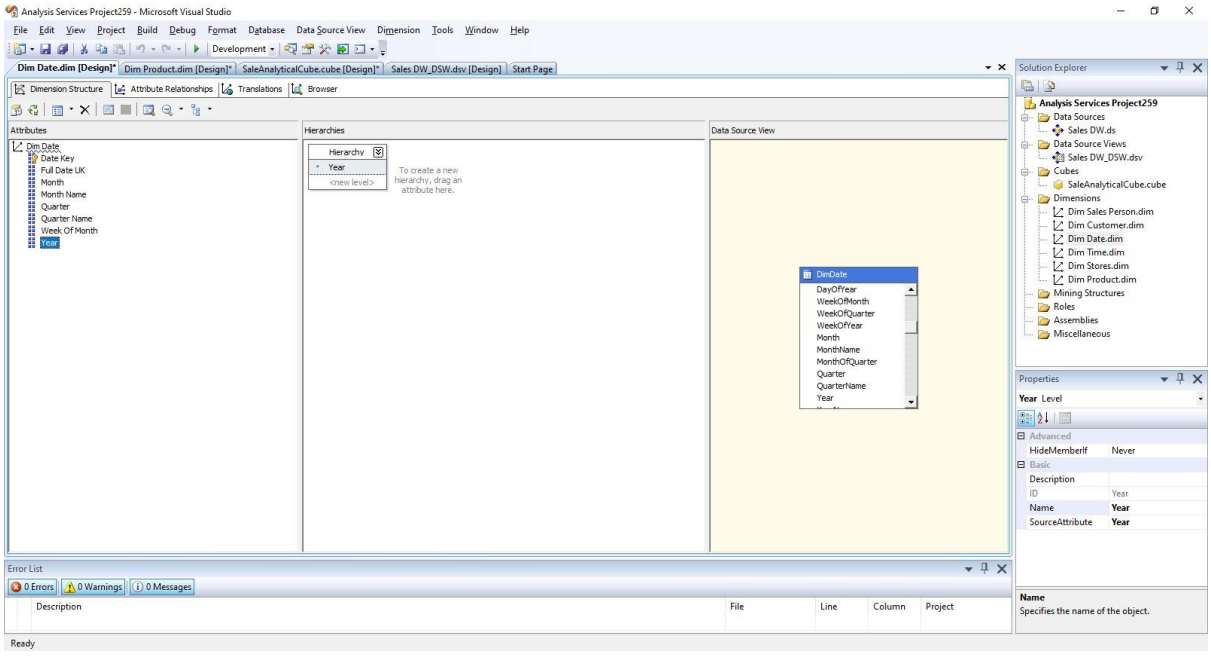
Drag fields in sequence from Attributes to Hierarchy window (Year, Quarter Name. Month Name, Week of the Month, Full Date UK),

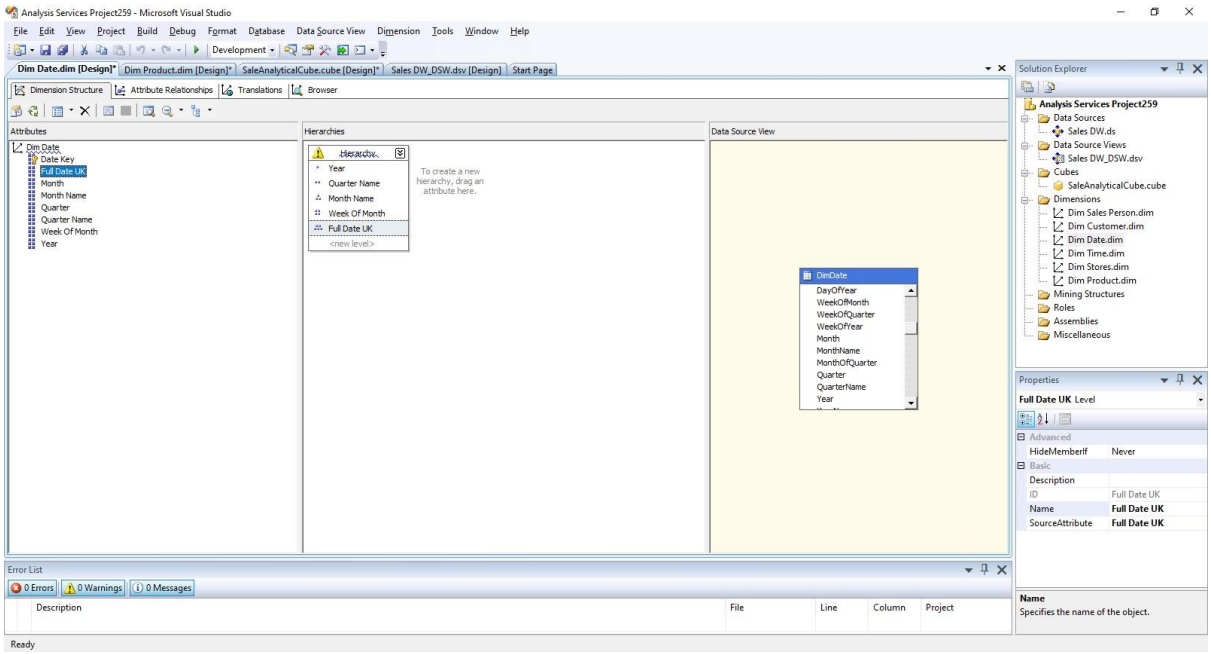


Step 8: Deploy the Cube

8.1 In Solution Explorer, right click on Project Name (cubepract4) --> Click Properties

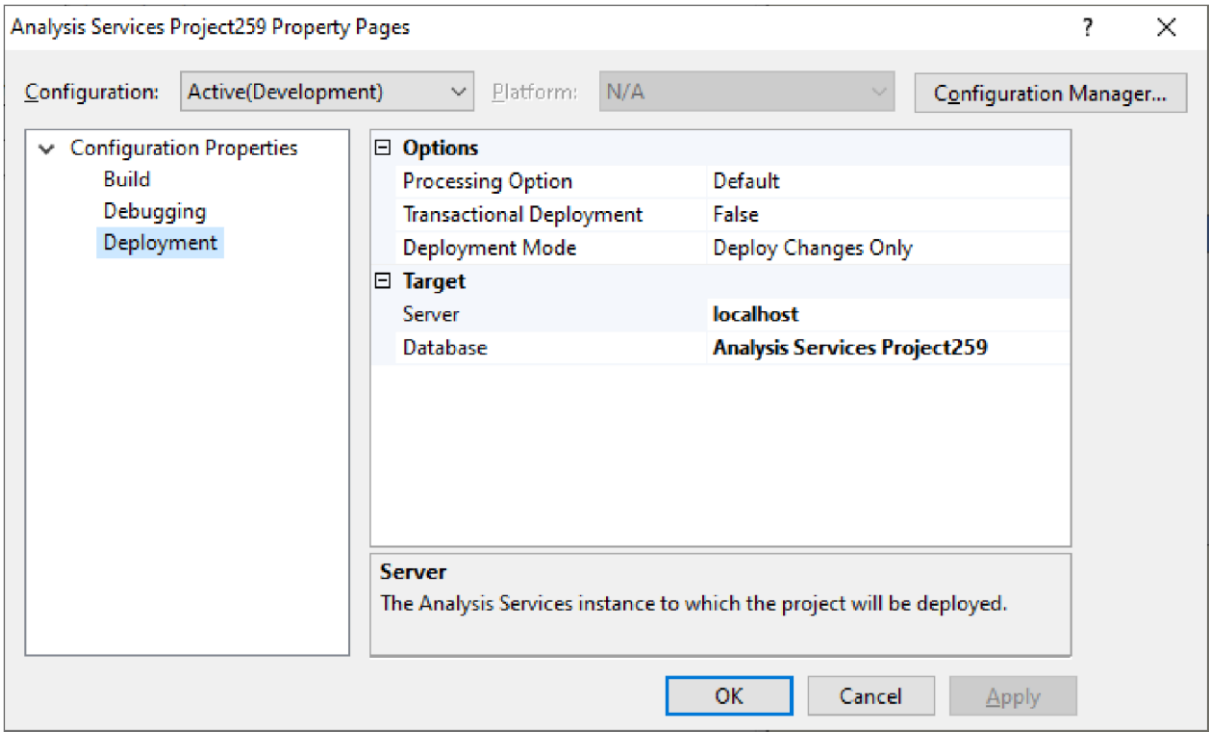


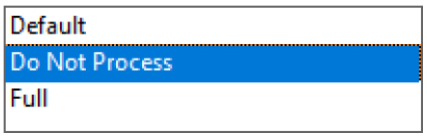


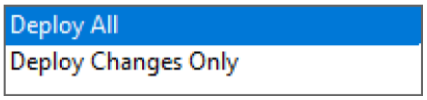


8.2 Set Deployment Properties First

In Configuration Properties, Select Deployment-> Assign Your SQL Server Instance Name Where Analysis Services Is Installed (localhost)(Machine Name\Instance Name)-> Choose Deployment Mode Deploy All as of now ->Select Processing Option Do Not Process -> Click OK

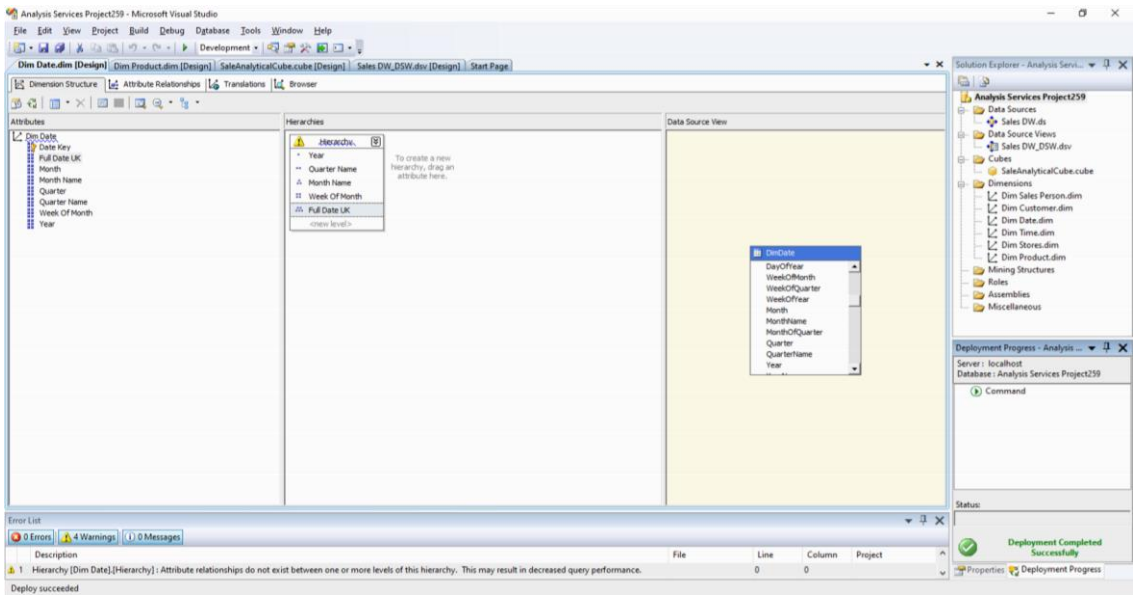






8.3 In Solution Explorer, right click on Project Name (SalesDataAnalysis) -> Click Deploy

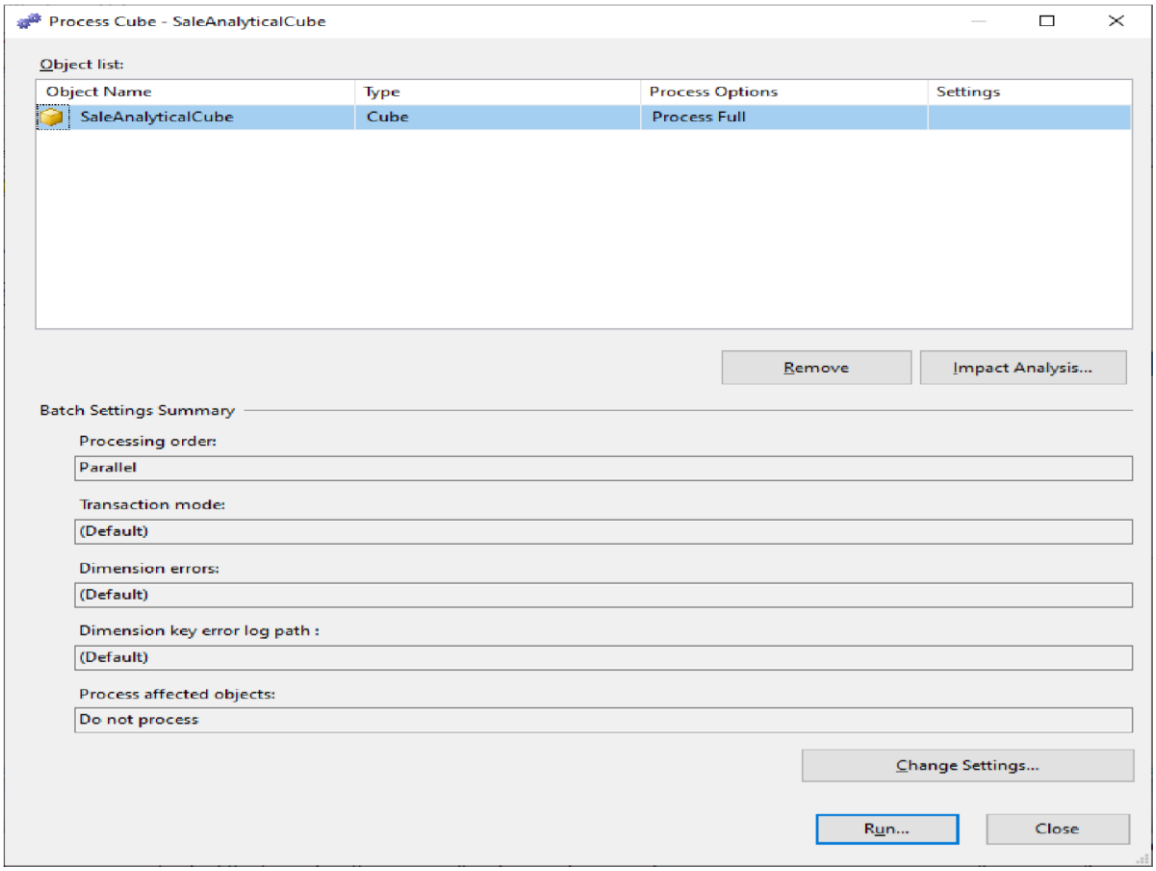
8.4 Once Deployment will finish, you can see the message Deployment Completed deployment Properties.



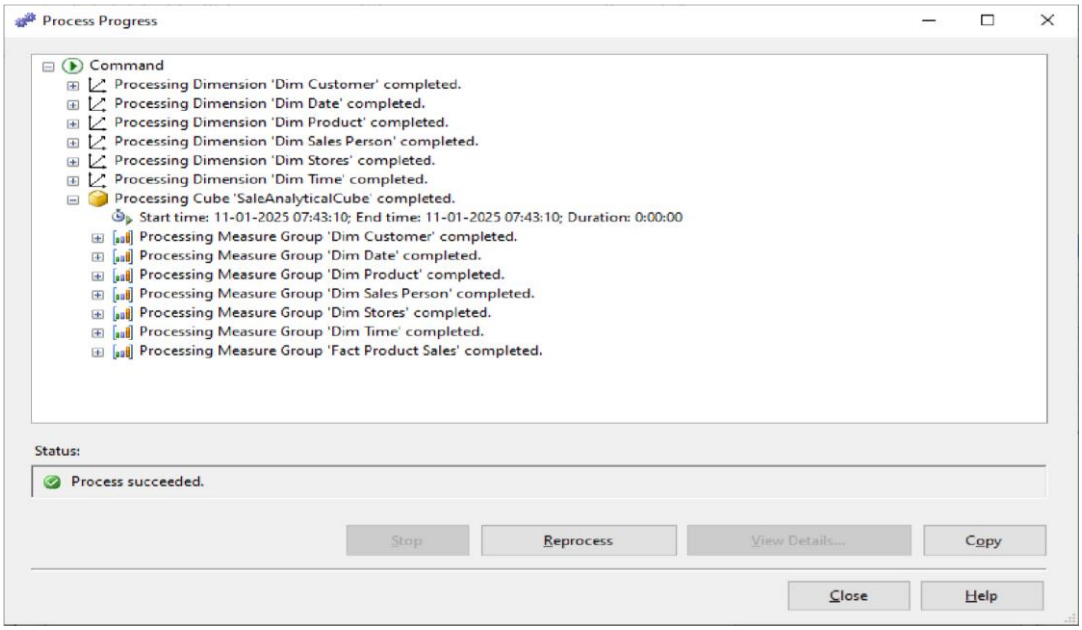
Step 9: Process the Cube

9.1 In Solution Explorer, right click on Project Name (SalesDataAnalysis) -> Click Process

9.2 Click on Run button to process the Cube

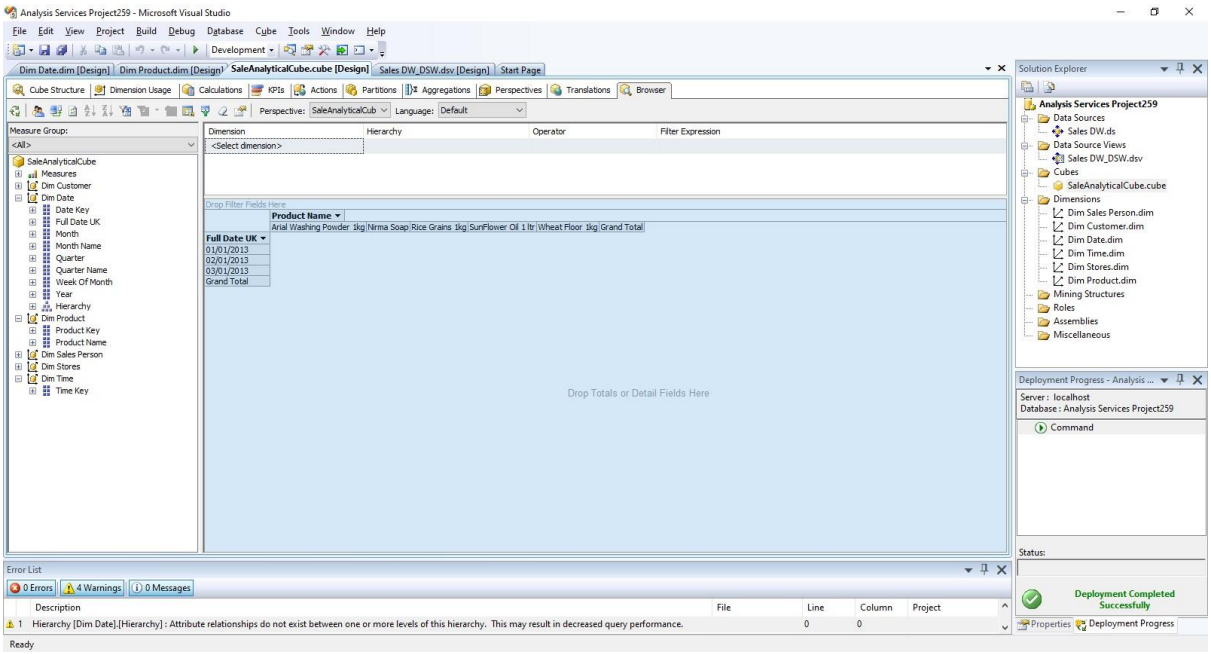


9.3 Once processing is complete, you can see Status as Process Succeeded -> Click Close both the open windows for processing one after the other.



Step 10: Browse the Cube for Analysis

10.1 In Solution Explorer, right click on Cube Name (SalesAnalyticalCube) -- > Click Browse



10.2 Drag and drop measures in to Detail fields, & Drag and Drop Dimension Attributes in Row Field or Column fields.

Now to Browse Our Cube

1.. Product Name Drag & Drop into Column

2. Full Date UK Drag & Drop into Row Field

3. FactProductSalesCount Drop this measure in Detail area