UNIVERSITY OF MUMBAI

DEPARTMENT OF COMPUTER SCIENCE

मुंबई विद्यापीठ
University of Mumbai
Re-accredited with A++ Grade
(CGPA 3.65) by NAAC (3rd Cycle 2021)

M.Sc. Computer Science – Semester III

Track D: Data Science

Elective I: Data Visualization

JOURNAL

2023-2024

Seat No. ___________________

मुंबई विद्यापीठ
University of Mumbai
Re-accredited with A++ Grade
(CGPA 3.65) by NAAC (3rd Cycle 2021)

UNIVERSITY OF MUMBAI

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the work entered in this journal was done in the University Department of Computer Science laboratory by Mr.___**Shikne Ritesh Mahesh**___ Seat No. _______________ for the course of M.Sc. Computer Science - Semester III (CBCS) (Revised) during the academic year 2023- 2024 in a satisfactory manner.

____

Subject In-charge

____

Head of Department

____

External Examiner

INDEX

PRACTICAL: 01

AIM: Create one-dimensional data using series and perform various operations on it.

THEORY

A one-dimensional array (or array) is a data structure that stores a sequence of (references to) objects. We refer to the objects within an array as its elements. The method that we use to refer to elements in an array is numbering and then indexing them.

Creating a series from Scalar value: In order to create a series from scalar value, an index must be provided. The scalar value will be repeated to match the length of the index.

SOURCE CODE & OUTPUT

# Series

```
import numpy as np

import pandas as pd

data = pd.Series([15,32,45,62,25,3,12,21])

data
```

```
0    15
1    32
2    45
3    62
4    25
5     3
6    12
7    21
dtype: int64
```

# Accessing elements

print("First element:-", data[0])

print("second Last element:-", data[len(data) - 2])

```
First element:- 15
second Last element:- 12
```

# Sliced

print("Sliced data (index 3 to 7):-")

print(data[3:7])

```
Sliced data (index 3 to 7):-
3    62
4    25
5     3
6    12
dtype: int64
```

# Arithmetic operations:

print("Add 2 to each element:-")

print(data+2)

print("Multiply each element by 3:-")

print(data *3)

```
Add 2 to each element:-
0    17
1    34
2    47
3    64
4    27
5     5
6    14
7    23
dtype: int64
Multiply each element by 3:-
0     45
1     96
2    135
3    186
4     75
5      9
6     36
7     63
dtype: int64
```

# Statistical operations

```
print("Mean:-", data.mean())
print("Sum:-", data.sum())
print("Maximum :-", data.max())
print("Minimum :-", data.min())
```

```
Mean:- 26.875
Sum:- 215
Maximum :- 62
Minimum :- 3
```

# Conditions

```
print("Elements greater than 21:-")
print(data[data>21])
```

```
Elements greater than 21:-
1    32
2    45
3    62
4    25
dtype: int64
```

PRACTICAL-2

AIM:- Create Two-dimensional data with the help of data frames and perform ifferent operations on it.

THEORY

A 2D array is an array of arrays that can be represented in matrix form, like rows and columns. In this array, the position of data elements is defined with two indices instead of a single index. In Python, we can access two-dimensional array elements using two indices.

Data Frame is a 2-dimensional labelled data structure with columns of potentially different types. You can think of it like a spread sheet or SQL table, or a dict of Series objects. It is generally the most commonly used pandas object.

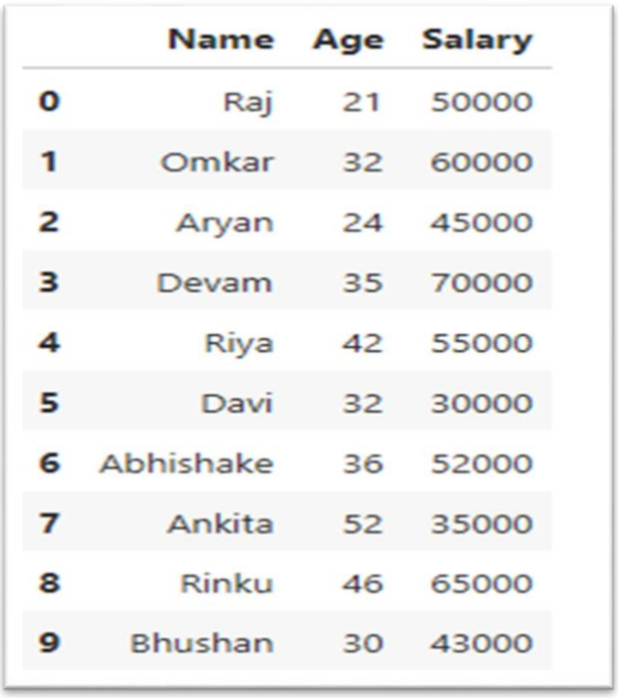
SOURCE CODE AND OUTPUTS

```
import numpy as np

import pandas as pd

Employee = {

   'Name':['Raj','Omkar','Aryan','Devam','Riya','Davi','Abhishake','Ankita','Rinku','Bhushan'],

   'Age':[21,32,24,35,42,32,36,52,46,30],

   'Salary':[50000, 60000, 45000, 70000, 55000,30000, 52000, 35000, 65000, 43000]

   }
```

# Data frame

df =pd.DataFrame(Employee)

df



	Name	Age	Salary
0	Raj	21	50000
1	Omkar	32	60000
2	Aryan	24	45000
3	Devam	35	70000
4	Riya	42	55000
5	Davi	32	30000
6	Abhishake	36	52000
7	Ankita	52	35000
8	Rinku	46	65000
9	Bhushan	30	43000

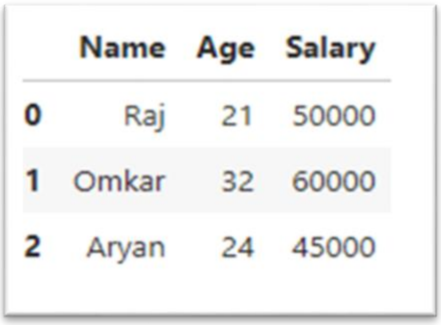
# Accessing columns

print(df['Name'])

```
0          Raj
1        Omkar
2        Aryan
3        Devam
4         Riya
5         Davi
6    Abhishake
7       Ankita
8        Rinku
9      Bhushan
Name: Name, dtype: object
```

# Slicing rows head first (3 value)

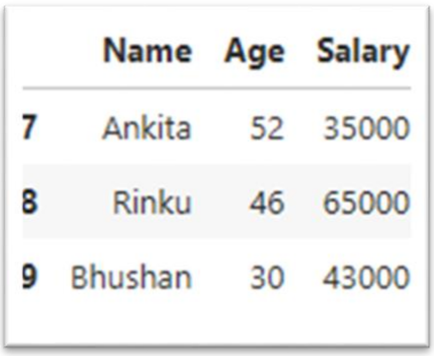
df.head(3)



	Name	Age	Salary
0	Raj	21	50000
1	Omkar	32	60000
2	Aryan	24	45000

# Slicing rows tail last (3 value)

df.tail(3)



	Name	Age	Salary
7	Ankita	52	35000
8	Rinku	46	65000
9	Bhushan	30	43000

# Filtering (conditions)

print(df[df['Salary']> 50000])

print(' ')

print(df[df['Age'] < 30])

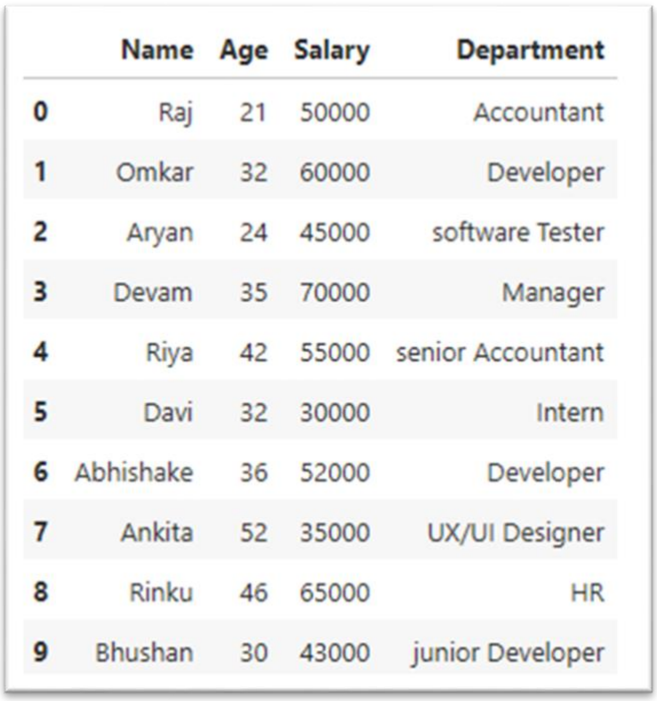
```
  Name   Age  Salary
1      Omkar   32   60000
3      Devam   35   70000
4       Riya   42   55000
6  Abhishake   36   52000
8      Rinku   46   65000

    Name  Age  Salary
0    Raj   21   50000
2  Aryan   24   45000
```

# Adding a new column

df['Department']=['Accountant','Developer','softwareTester','Manager','senior ccountant','Intern','Developer','UX/UI Designer','HR','junior Developer']

# Show DataFrame

df



	Name	Age	Salary	Department
0	Raj	21	50000	Accountant
1	Omkar	32	60000	Developer
2	Aryan	24	45000	software Tester
3	Devam	35	70000	Manager
4	Riya	42	55000	senior Accountant
5	Davi	32	30000	Intern
6	Abhishake	36	52000	Developer
7	Ankita	52	35000	UX/UI Designer
8	Rinku	46	65000	HR
9	Bhushan	30	43000	junior Developer

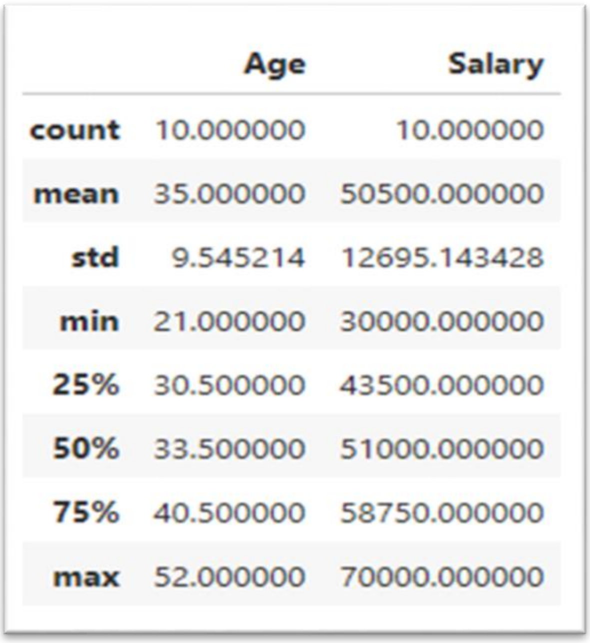
# Information

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Name        10 non-null     object
 1   Age         10 non-null     int64 
 2   Salary      10 non-null     int64 
 3   Department  10 non-null     object
dtypes: int64(2), object(2)
memory usage: 448.0+ bytes
```

# Describe

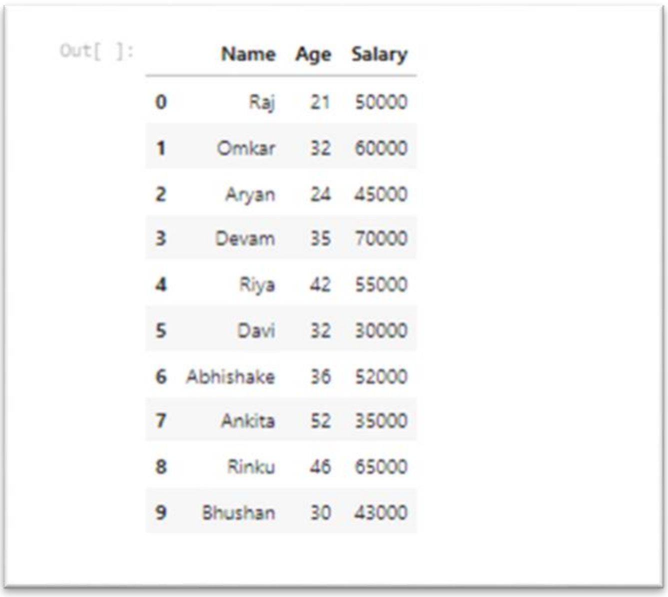
df.describe()



	Age	Salary
count	10.000000	10.000000
mean	35.000000	50500.000000
std	9.545214	12695.143428
min	21.000000	30000.000000
25%	30.500000	43500.000000
50%	33.500000	51000.000000
75%	40.500000	58750.000000
max	52.000000	70000.000000

# Removing

df = df.drop(columns=['Department'])



Out[ ]:

	Name	Age	Salary
0	Raj	21	50000
1	Omkar	32	60000
2	Aryan	24	45000
3	Devam	35	70000
4	Riya	42	55000
5	Davi	32	30000
6	Abhishake	36	52000
7	Ankita	52	35000
8	Rinku	46	65000
9	Bhushan	30	43000

# Creating Data Visualization

import pandas as pd

import matplotlib.pyplot as plt

# Load the data into a Series object

series = pd.Series([1, 2, 3, 4, 5])

# Explore the data

print(series.head())

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

print(series.dtype)

```
int64
```

print(len(series))

```
5
```

print(series.unique())

```
[1 2 3 4 5]
```

print(series.isna())

```
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

# Analyze the data

print(series.describe())

```
count    5.000000
mean     3.000000
std      1.581139
min      1.000000
25%      2.000000
50%      3.000000
75%      4.000000
max      5.000000
dtype: float64
```

# Plot the data

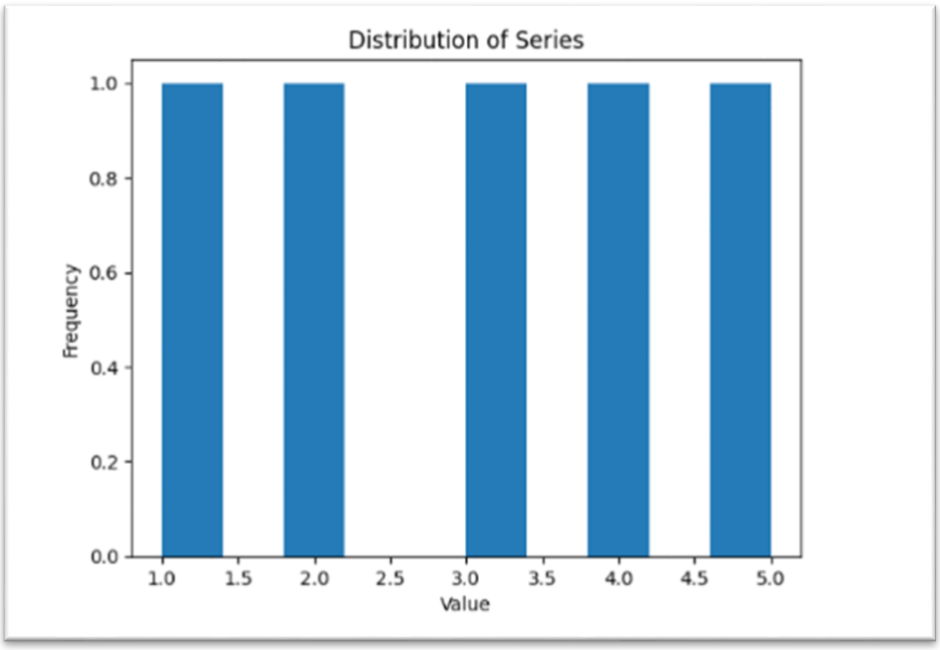
plt.hist(series)

plt.xlabel("Value")

plt.ylabel("Frequency")

plt.title("Distribution of Series")

plt.show()



PRACTICAL – 3

AIM: Write a code to read data from the different file formats like JSON, HTML, and CSV files and check for missing data and outlier values and handle them.

THEORY

• Different file format

CSV (or Comma Separated Value) file is the most common type of file that a data scientist will ever work with. These files use a "," as a delimiter to separate the values and each row in a CSV file is a data record.

HTML (Hyper Text Markup Language) the standard markup language for creating Web pages. It describes the structure of a Web page and consists of a series of elements. It's elements tell the browser how to display the content.

JSON (JavaScript Object Notation) files are lightweight and human-readable to store and exchange data. It is easy for machines to parse and generate these files and are based on the JavaScript programming language. JSON files store data within {} similar to how a dictionary stores it in Python.

• Missing and Outlier value treatment

Data Cleaning is the process of finding and correcting the inaccurate/incorrect data that are present in the dataset. Missing values are usually represented in the form of Nan or null or None in the dataset. an Outlier is an observation in a given dataset that lies far from the rest of the observations. That means an outlier is vastly larger or smaller than the remaining values in the set. If our dataset is small, we can detect the outlier by just looking at the dataset.

SOURCE CODE AND OUTPUT

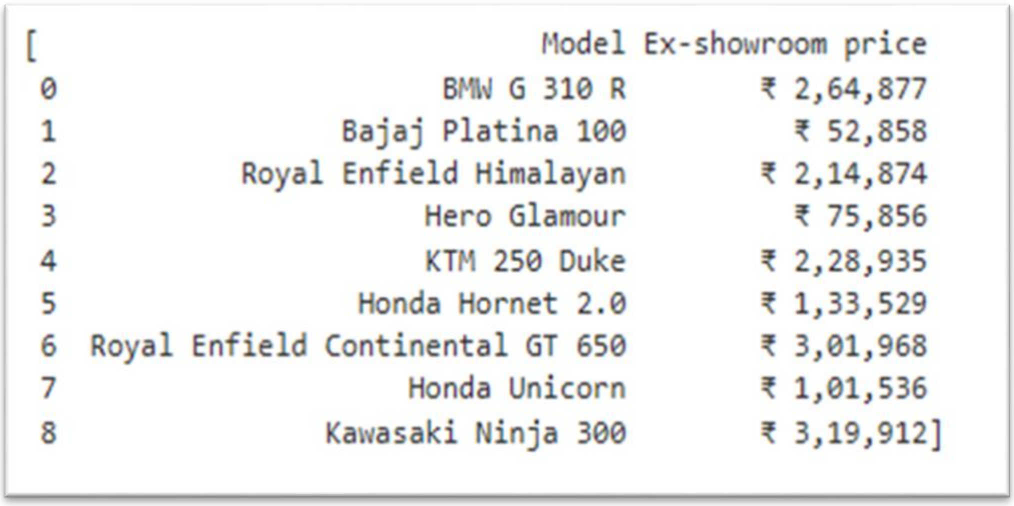
1. `Important labour for data cleaning

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns
```

2. Read data from the different file formats like JSON, HTML, and CSV files

a) HTML file

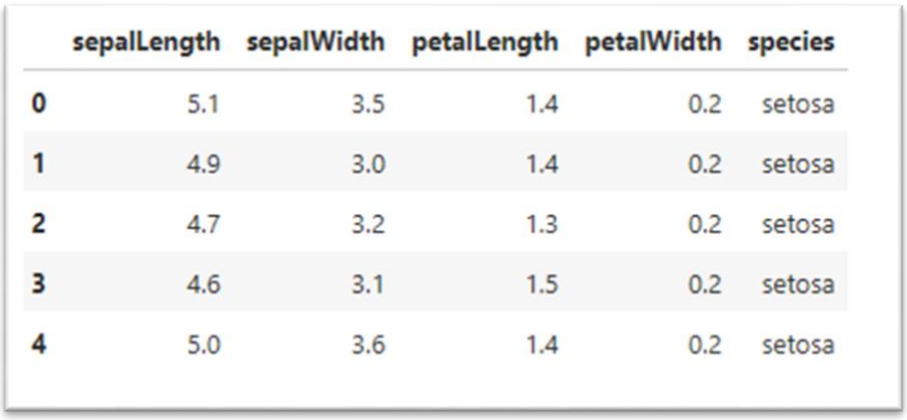
```
    Bikes_df  =  pd.read_html("/content/sample_data/Best  Bikes  in
India - April 2022 _ Top 10 Bikes -  BikeWale.html")

  Bikes_df
```



```
[                               Model Ex-showroom price
 0                        BMW G 310 R       ₹ 2,64,877
 1                  Bajaj Platina 100         ₹ 52,858
 2            Royal Enfield Himalayan       ₹ 2,14,874
 3                       Hero Glamour         ₹ 75,856
 4                      KTM 250 Duke       ₹ 2,28,935
 5                   Honda Hornet 2.0       ₹ 1,33,529
 6  Royal Enfield Continental GT 650       ₹ 3,01,968
 7                       Honda Unicorn       ₹ 1,01,536
 8                  Kawasaki Ninja 300       ₹ 3,19,912]
```

b) JSON file

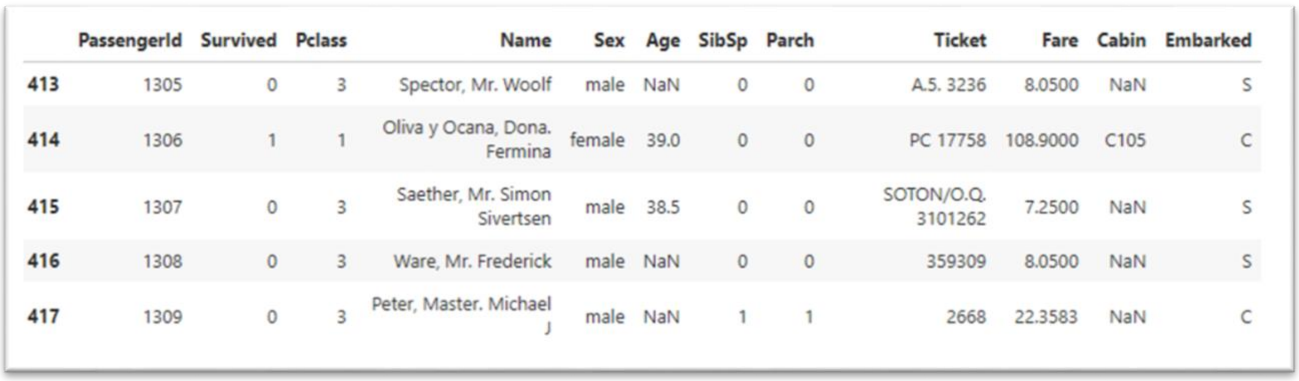
```
iris_df = pd.read_json("/content/sample_data/iris.json")

iris_df.shape

(150, 5)

iris_df.head()
```



	sepalLength	sepalWidth	petalLength	petalWidth	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

c) CSV file

```
tit_df = pd.read_csv("/content/sample_data/Titanic dataset.csv")

tit_df.shape

(418, 12)
tit_df.head()

tit_df.tail()
```



	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	0	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
417	1309	0	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

tit_df["Survived"].unique()

```
array([0, 1])
```

tit_df["Survived"].value_counts()

```
 0    266
1    152
Name: Survived, dtype: int64
```

tit_df["Pclass"].value_counts()

```
3    218
1    107
2     93
Name: Pclass, dtype: int64
```

tit_df['Pclass'].unique()

```
 array([3, 2, 1])
```

tit_df["Sex"].value_counts()

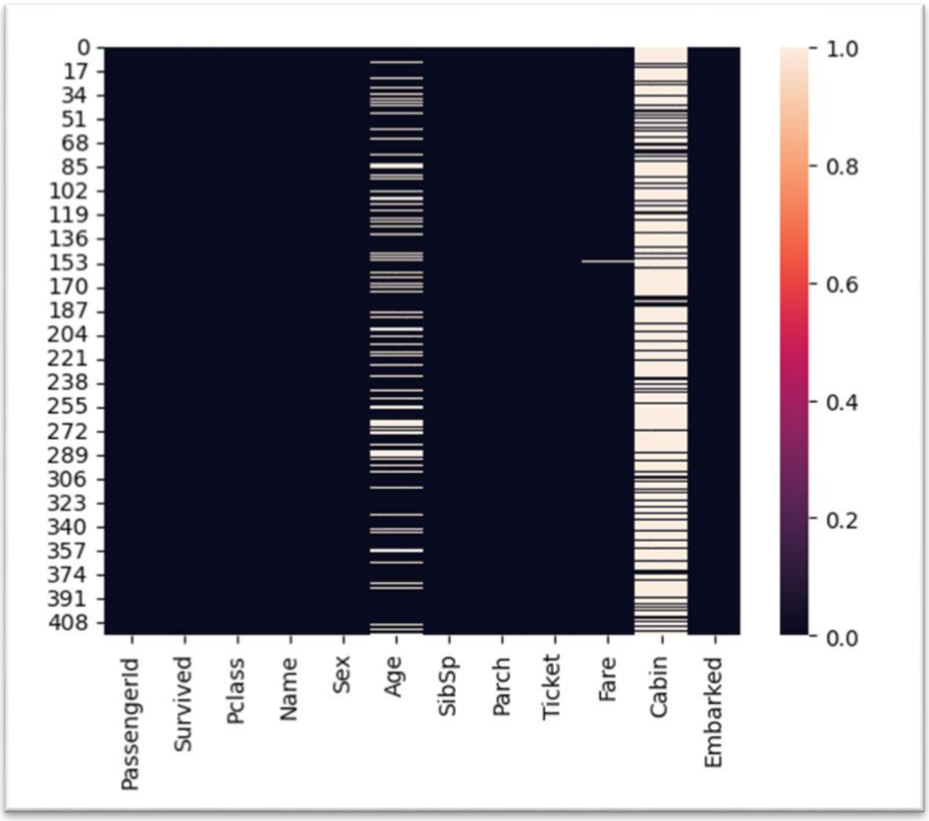
```
 male      266
female    152
Name: Sex, dtype: int64
```

tit_df.isnull().sum()

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

# To check the null value in tit_dif the white line in coloumn (Cabin,Fare,Age)

```
sns.heatmap(tit_df.isnull())
plt.show()
```



# No. of rows

tit_df.shape[0]

```
418
```

#No. of columns

tit_df.shape[1]

```
12
```

# Number of rows in the DataFrame

tit_df.isnull().sum()*100/tit_df.shape[0]

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            20.574163
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.239234
Cabin          78.229665
Embarked        0.000000
dtype: float64
```

# Display the data types

tit_df.dtypes

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

# Count of unique values

tit_df["Age"].value_counts()

```
21.0    17
24.0    17
22.0    16
30.0    15
18.0    13
        ..
76.0     1
28.5     1
22.5     1
62.0     1
38.5     1
Name: Age, Length: 79, dtype: int64
```

# Find the mode

tit_df["Age"].mode()

```
0    21.0
1    24.0
Name: Age, dtype: float64
```

#Filiing nul values of age in average/mean of age permenant

m = np.round(tit_df["Age"].mean())

m

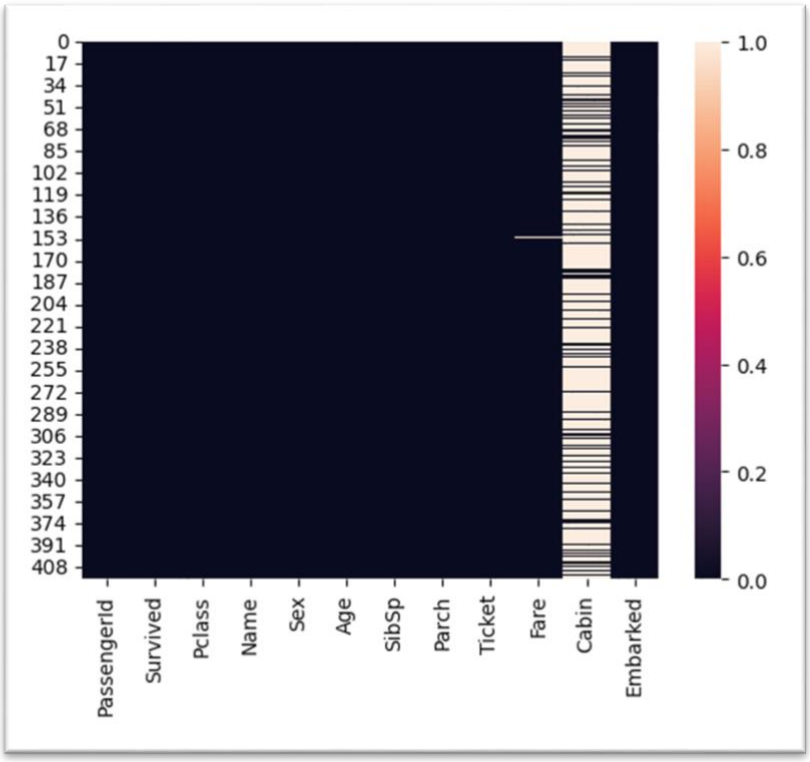
```
30.0
```

tit_df["Age"].fillna(m, inplace=True)

# To cheack the null value in tit_dif the white line in coloumn (Cabin,Fare,Age)

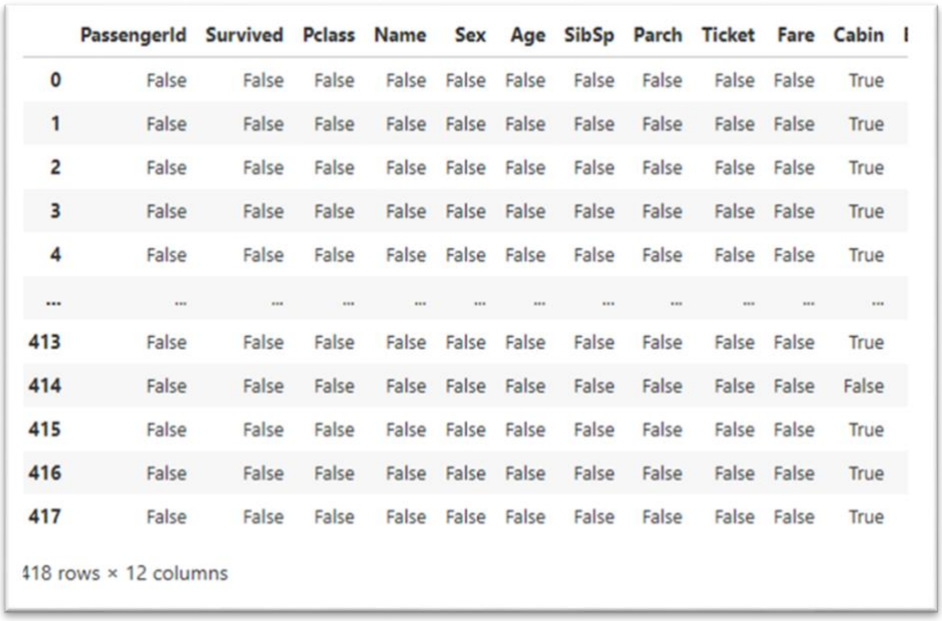
sns.heatmap(tit_df.isnull())

plt.show()



# Create a Boolean mask

tit_df.isnull()



	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	I
0	False	False	False	False	False	False	False	False	False	False	True	
1	False	False	False	False	False	False	False	False	False	False	True	
2	False	False	False	False	False	False	False	False	False	False	True	
3	False	False	False	False	False	False	False	False	False	False	True	
4	False	False	False	False	False	False	False	False	False	False	True	
...	...	...	...	...	...	...	...	...	...	...	...	
413	False	False	False	False	False	False	False	False	False	False	True	
414	False	False	False	False	False	False	False	False	False	False	False	
415	False	False	False	False	False	False	False	False	False	False	True	
416	False	False	False	False	False	False	False	False	False	False	True	
417	False	False	False	False	False	False	False	False	False	False	True	

418 rows × 12 columns

# Obtain a count of unique

tit_df["Fare"].value_counts()

```
7.7500     21
26.0000    19
13.0000    17
8.0500     17
7.8958     11
           ..
7.8208      1
8.5167      1
78.8500     1
52.0000     1
22.3583     1
Name: Fare, Length: 169, dtype: int64
```

tit_df["Fare"].mode()

```
0    7.75
Name: Fare, dtype: float64
```

n = tit_df['Fare'].mode()

n

```
0    7.75
Name: Fare, dtype: float64
```

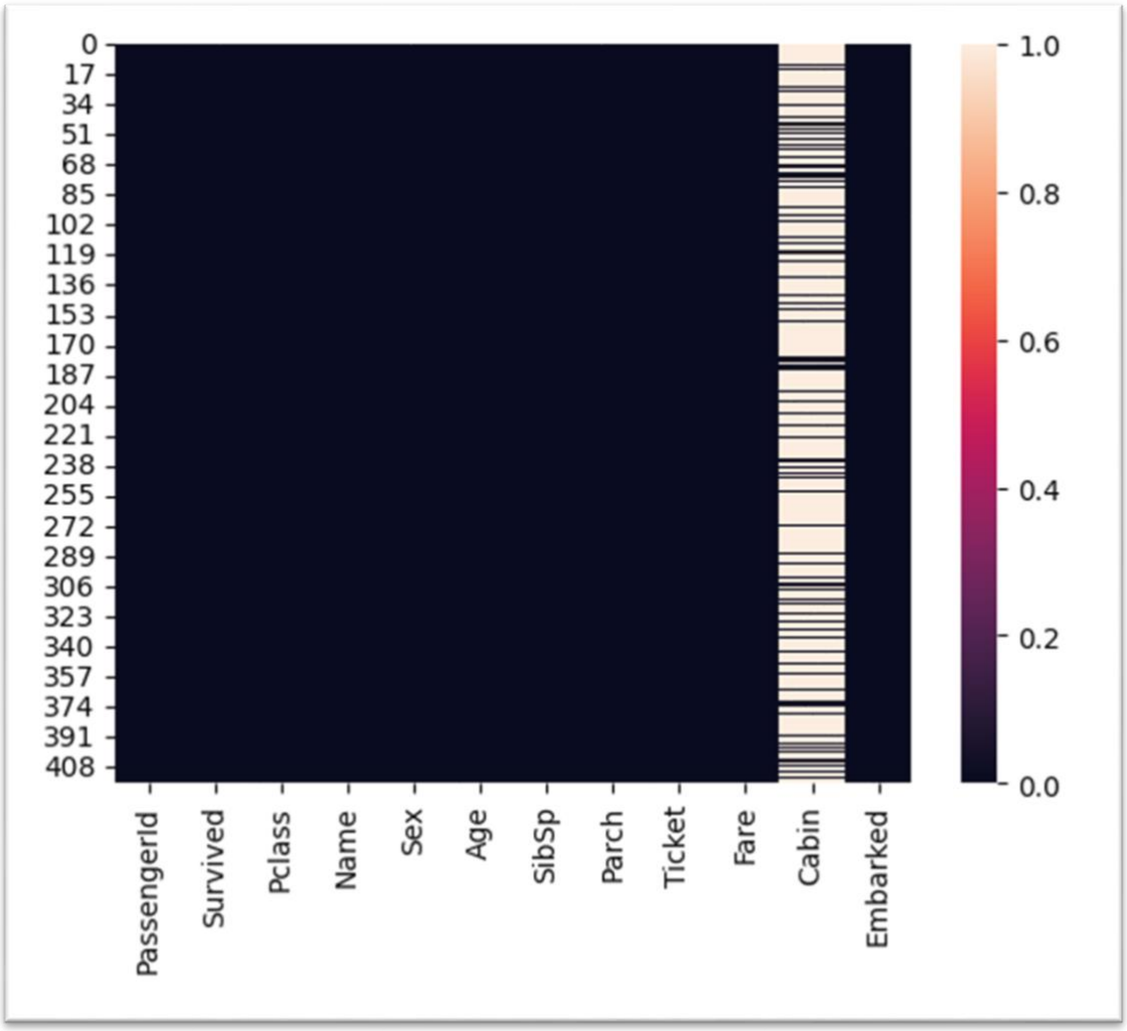
tit_df["Fare"].fillna(" 7.75", inplace=True)

tit_df.isnull().sum()

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          327
Embarked         0
dtype: int64
```

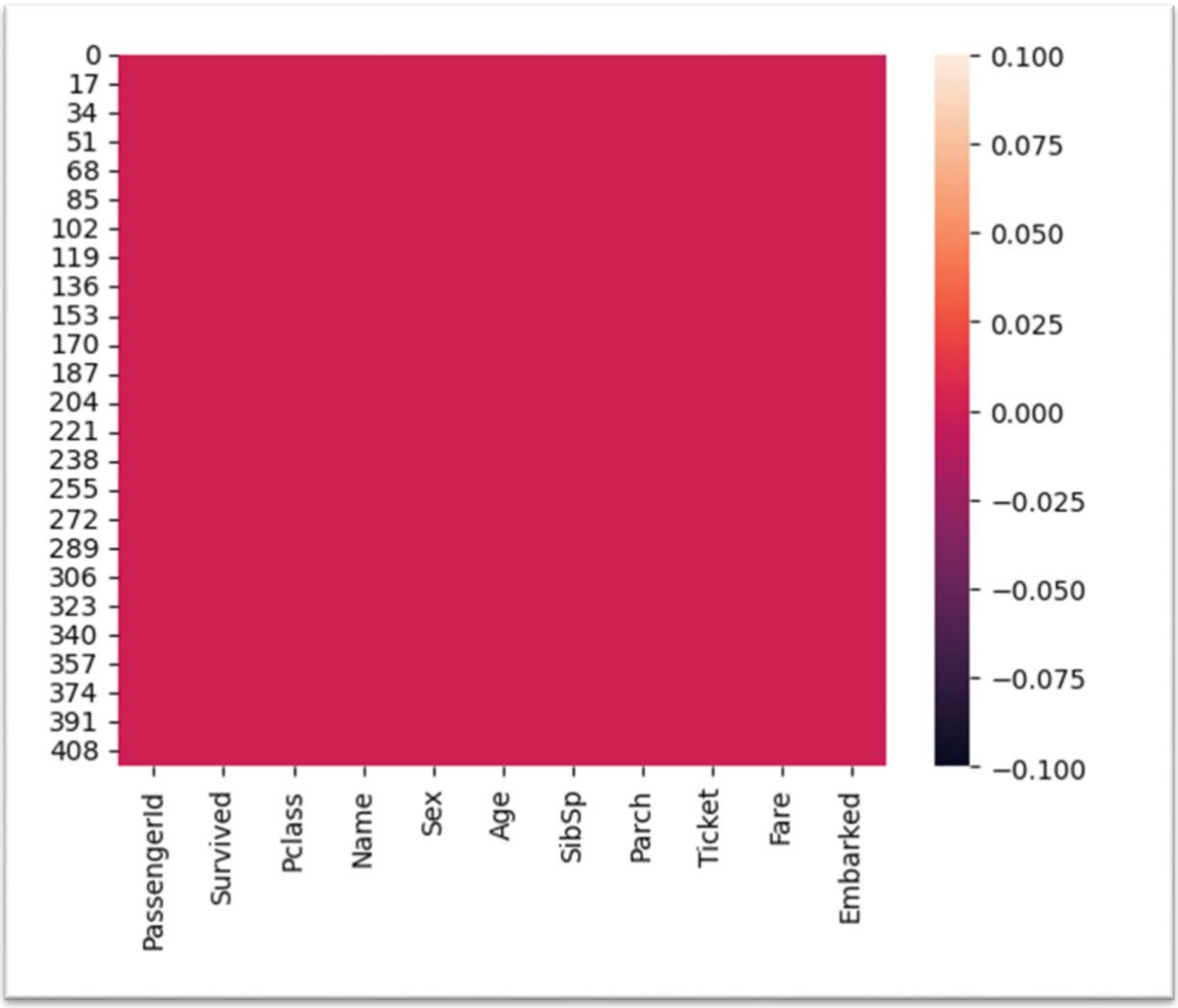
# to check the null value in tit_dif the white line in column (Cabin,Fare,Age)

```
sns.heatmap(tit_df.isnull())
plt.show()
```



# To delete cabin column permanently from given dataset

```
tit_df.drop('Cabin',axis=1, inplace=True)
sns.heatmap(tit_df.isnull())
plt.show()
```



#To check if there is any duplicate rows

```
tit_df.duplicated().sum()
```

```
0
```

PRACTICAL – 4

AIM: Perform Reshaping of the hierarchical data and pivoting data frame data.

THEORY

In pandas, we can arrange data within the data frame from the existing data frame. For example, we have the same name with different features, instead of writing the name all time, we can write only once.

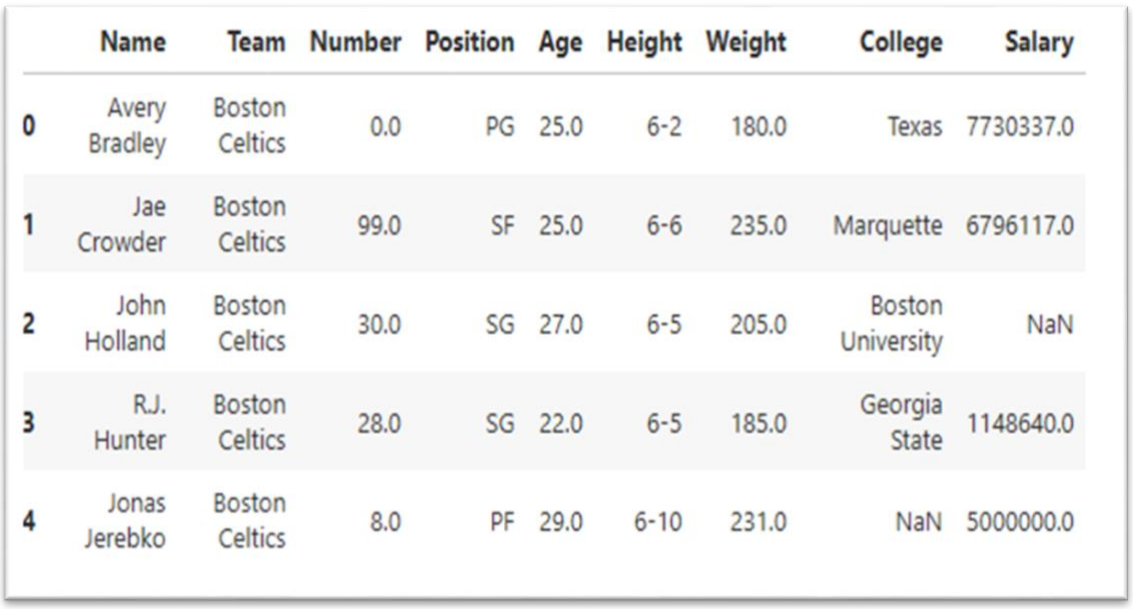
We can create hierarchical data from the existing data frame using pandas. Stacking a Data Frame means moving (also rotating or pivoting) the innermost column index to become the innermost row index. The inverse operation is called unstacking. It means moving the innermost row index to become the innermost column index.

The pivot function is used to create a new derived table out of a given one. Pivot takes 3 arguments with the following names: index, columns, and values. As a value for each of these parameters you need to specify a column name in the original table. Then the pivot function will create a new table, whose row and column indices are the unique values of the respective parameters

SOURCE CODE AND OUTPUT

# Reshaping of the hierarchical data

```
import numpy as np

import pandas as pd

df=pd.read_csv("/content/sample_data/nba.csv")

df.head()
```



	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0

#Used to stack (or melt) the DataFrame

df_stacked=df.stack()

df_stacked.head(26)

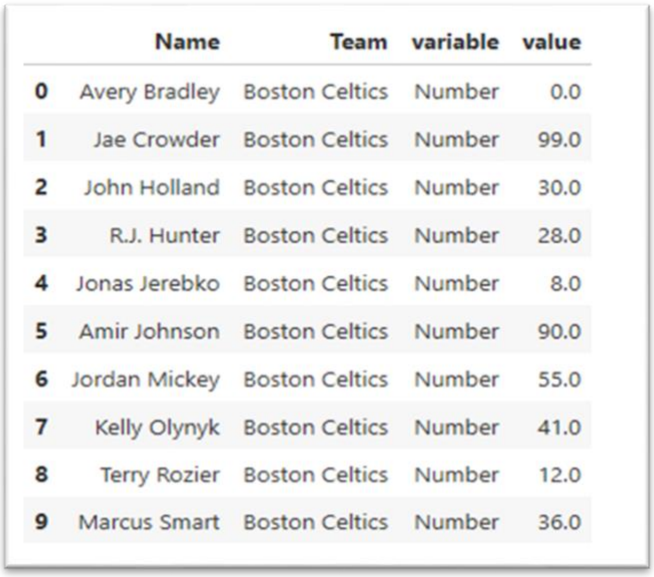
```
0  Name            Avery Bradley
   Team           Boston Celtics
   Number                    0.0
   Position                   PG
   Age                      25.0
   Height                    6-2
   Weight                  180.0
   College                 Texas
   Salary              7730337.0
1  Name              Jae Crowder
   Team           Boston Celtics
   Number                   99.0
   Position                   SF
   Age                      25.0
   Height                    6-6
   Weight                  235.0
   College            Marquette
   Salary              6796117.0
2  Name             John Holland
   Team           Boston Celtics
   Number                   30.0
   Position                   SG
   Age                      27.0
   Height                    6-5
   Weight                  205.0
   College     Boston University
dtype: object
```

df_unstacked=df_stacked=df.unstack()

df_unstacked.head(10)

```
Name  0    Avery Bradley
      1      Jae Crowder
      2     John Holland
      3      R.J. Hunter
      4    Jonas Jerebko
      5     Amir Johnson
      6    Jordan Mickey
      7     Kelly Olynyk
      8     Terry Rozier
      9     Marcus Smart
dtype: object
```

```
df_melt = df.melt(id_vars =['Name', 'Team'])

df_melt.head(10)
```



```
          Name            Team  variable  value
0  Avery Bradley  Boston Celtics  Number    0.0
1    Jae Crowder  Boston Celtics  Number   99.0
2   John Holland  Boston Celtics  Number   30.0
3    R.J. Hunter  Boston Celtics  Number   28.0
4  Jonas Jerebko  Boston Celtics  Number    8.0
5   Amir Johnson  Boston Celtics  Number   90.0
6  Jordan Mickey  Boston Celtics  Number   55.0
7   Kelly Olynyk  Boston Celtics  Number   41.0
8   Terry Rozier  Boston Celtics  Number   12.0
9   Marcus Smart  Boston Celtics  Number   36.0
```

#Pivoting data frame data

```
import pandas as pd

import numpy as np

import warnings

warnings.filterwarnings('ignore')
```

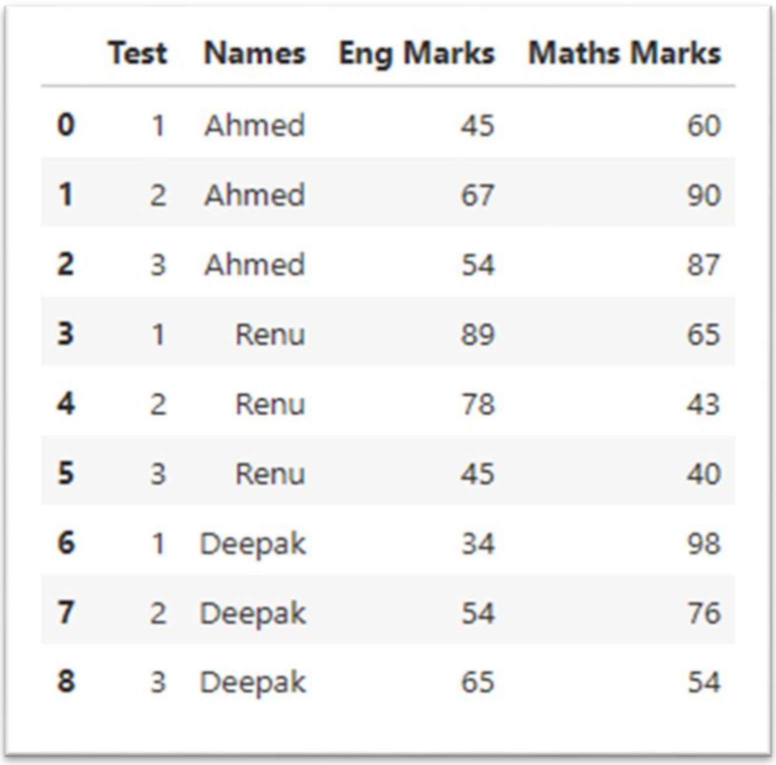
#Create a dummy datasframe from dictionary

```
dict={'Test':[1,2,3,1,2,3,1,2,3],

      'Names':['Ahmed','Ahmed','Ahmed','Renu','Renu','Renu',

               'Deepak','Deepak','Deepak'],

     'Eng Marks':[45,67,54,89,78,45,34,54,65],

     'Maths Marks':[60,90,87,65,43,40,98,76,54]}
```

#Converts dictionary into dataframe

df=pd.DataFrame(dict)

df

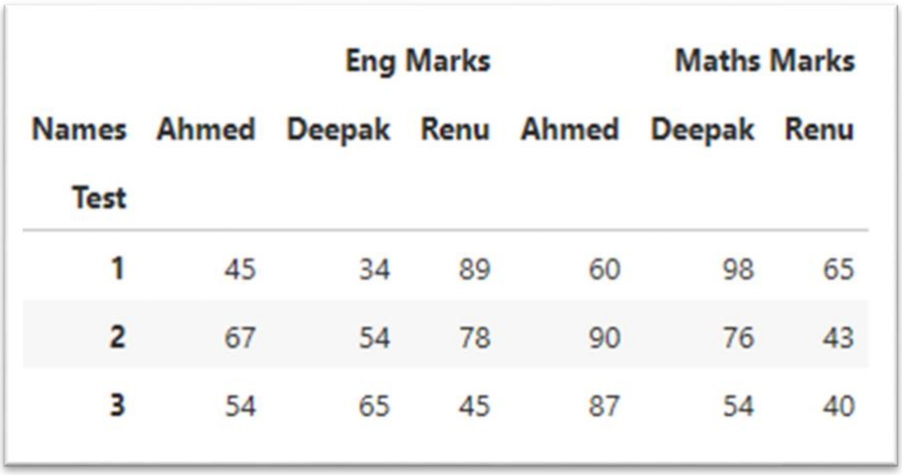


	Test	Names	Eng Marks	Maths Marks
0	1	Ahmed	45	60
1	2	Ahmed	67	90
2	3	Ahmed	54	87
3	1	Renu	89	65
4	2	Renu	78	43
5	3	Renu	45	40
6	1	Deepak	34	98
7	2	Deepak	54	76
8	3	Deepak	65	54

#Here long dataset means no. of rows=9 and no. of columns=4
#9>4

#pivot() inbuilt method of Pandas , use for DataFrame

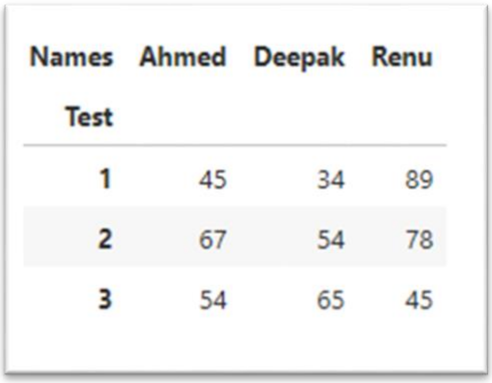
df.pivot(index='Test',columns='Names')



	Eng Marks			Maths Marks		
Names	**Ahmed**	**Deepak**	**Renu**	**Ahmed**	**Deepak**	**Renu**
Test						
1	45	34	89	60	98	65
2	67	54	78	90	76	43
3	54	65	45	87	54	40

#Only show marks of english of each students

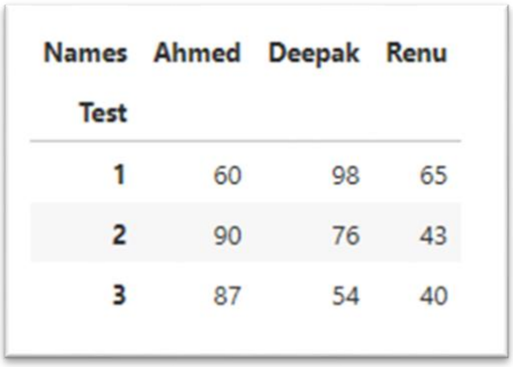
```
df.pivot(index='Test',columns='Names',values='Eng Marks')
```



Names	Ahmed	Deepak	Renu
Test			
1	45	34	89
2	67	54	78
3	54	65	45

#Only show marks of maths of each students

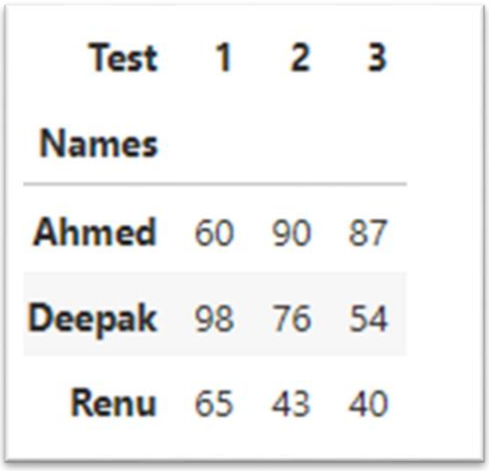
```
df.pivot(index='Test',columns='Names',values='Maths Marks')
```



Names	Ahmed	Deepak	Renu
Test			
1	60	98	65
2	90	76	43
3	87	54	40

#Use pivot_table()

```
df1=df.pivot_table(index='Names',columns='Test',values='Maths Marks',
                aggfunc='mean')
#by default aggfunc='mean'
#aggragate function : sum() ,mean() ,max() ,min() ,count()


df1
```



Test	1	2	3
Names			
Ahmed	60	90	87
Deepak	98	76	54
Renu	65	43	40

PRACTICAL-05

AIM: - Connecting and extracting with various data resources in Tableau.

THEORY

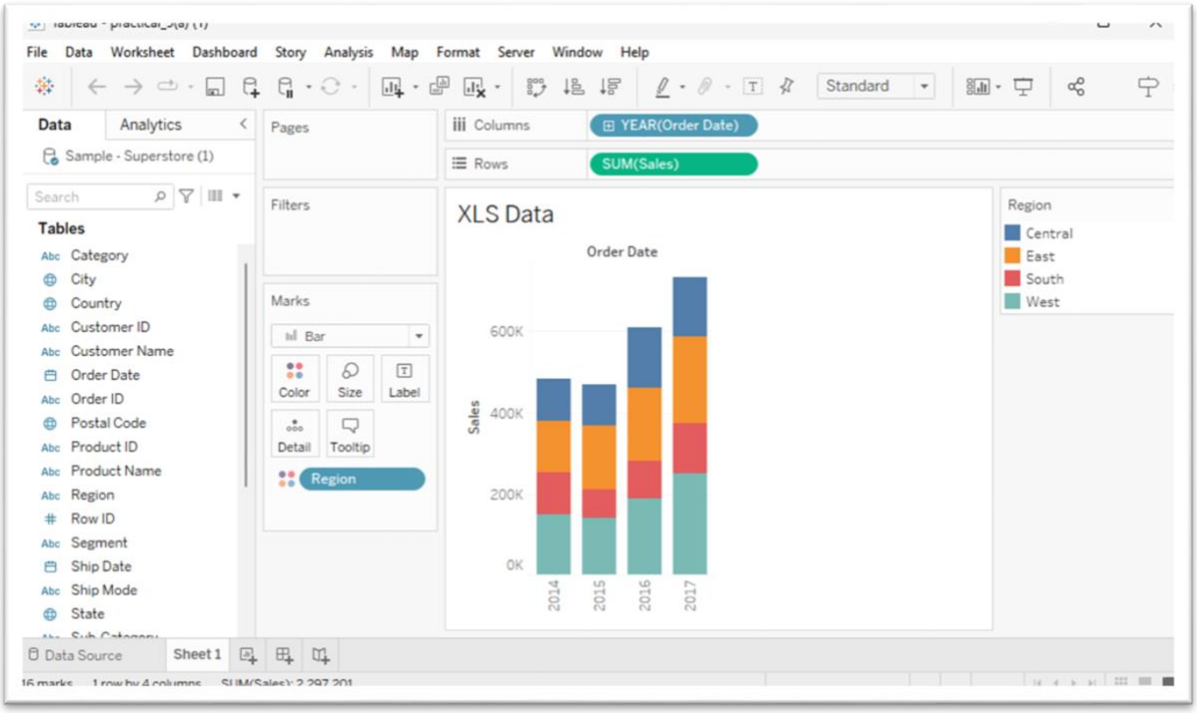
Tableau offers a myriad of data sources such as local text files, MS Excel, PDFs, JSON or databases and servers like Tableau Server, MySQL Server, Microsoft SQL Server, etc. Categorically, there are two types of data sources that you can connect to in Tableau; To a file and To a server.

STEPS

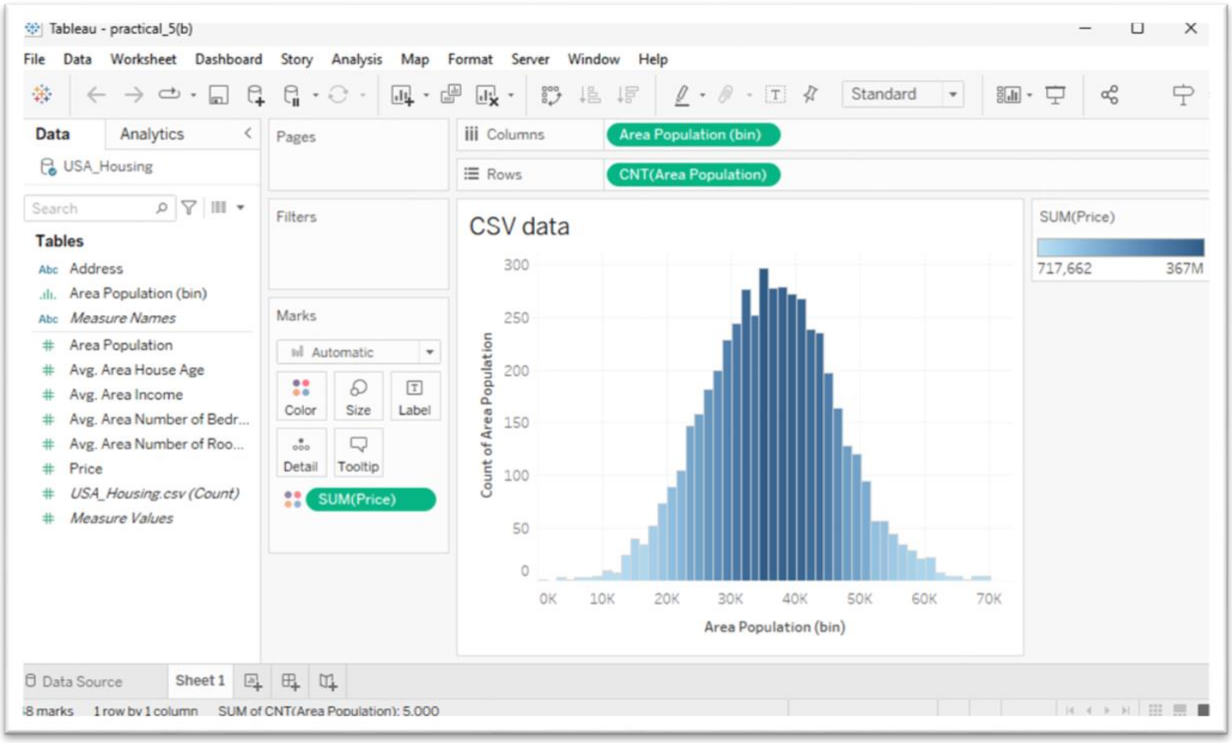
1) For the Excel graph (sample_-superstore.xls), the sample data set for Super Store is being used. For representation of year (ship date) to sum of profit representing the region of people according to the different colours. It has parameters such as order ID, order date, product code, country, region, etc.
2) For the CSV graph (USA_housing.csv), the sample dataset used is USA housing, which represents the area according to the population and the sum of prices in that particular area. It talks about the dataset that can be used when considering buying a house and considers parameters such as the number of bedrooms, price, etc.
3) For JSON GRAPH (iris.json), this is the iris data set that revolves around the iris of a flower and has parameters such as petal length, width, sepal length, width, etc. This is to show the petal length in the count of petal lengths.

RESULTS

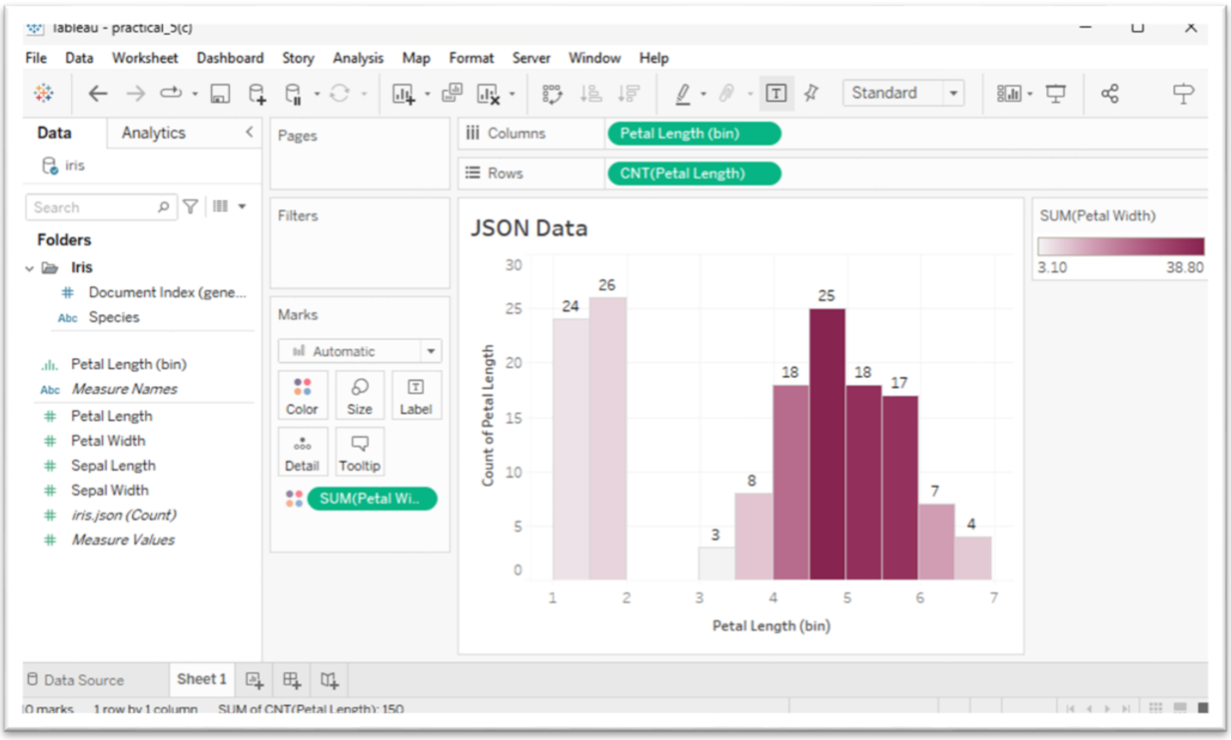
a) XLS Dataset Graph



b) CSV data Graph



c) JSON Data Graph



PRACTICAL -06

AIM: - Performing calculations and creating parameters in Tableau.

THEORY

A calculation is often referred to as a Calculated Field in Tableau. Calculations consist of code that's made up of functions, operations, and references to other fields, parameters, constants groups, or sets. This code returns a value.

Types of calculations:

•Row- level Calculations: These calculations are performed for every row of underlying data.

•Aggregate Calculations: These calculations are performed at an aggregate level, which is usually defined by the dimensions used in the view.

•Level of detail Calculations: These special calculations are aggregations that are performed at a specified level of detail, with the results available at row level.

•Table Calculations: These calculations are performed on the table of aggregate data has been returned by the data source to Tableau. A parameter in Tableau is a placeholder for a single, global value such as a number, date, or string.

STEPS

1. Parameter with filter

a) Import the superstore dataset.

b) Use sum of sale as columns and sub category with rows shelf, sort the data in descending order.

c) Drag the sub category from the tables into the filters pane. A dialogue box appear> click on top > click on by field, the default is top 10 sales which canbe changed later on if required. Click on apply > and ok.

d) But if I need to make changes into the dashboard with a particular graph we might need a parameter.

I. Right click on the sub category> edit filter > top> click on the boxwith 10 written on it> we get an option of create new parameter.

II. Name the parameter as Top 10 Sales> let the current value be 10>change the maximum value to 10(for the user to change the values only up to 10)>apply>AND>OK

III. The parameter is created in the parameter pane > right click on it> show parameter. The parameter is visible on the right side of the chart.

2. Parameter with sets

a) Import the superstore dataset.
b) Use sum of sale as columns and sub category with rows shelf, sort the data in descending order.
c) To highlight the subcategory we have to create a set > click on sub category dropdown > create > set >.

i. Change the name as MySet > click on Top.
ii. Click on by field> change the value 10 to the name of parameter withfilter (Top 10 Sales)> ok.
iii. The MySet is created on the left data pane.
iv. Show the parameter control on the right > right click on Top 10 Sales> show parameter.
v. Drag the MySet into the colour within the marks pane.
vi. Based on the range in the parameter the sets are created.

3. Calculated fields.

a) Import superstore dataset.
b) Double click on subcategory > sales > profit > sort in descending order.
c) If we need to calculate the cost that is sales- profit we need a measure butthat is not available so we follow the below steps:

i. Right click on any measure > create > calculated field.
ii. We can name the calculated field and calculations, on the left sidethere are calculations we can perform.
iii. Name the field as cost and write SUM ([Sales])- SUM([Profit]) >apply > ok.
iv. We get a cost measure name created. Drag and drop the cost into thesheet with other fields.

RESULTS

I. Parameter with filter

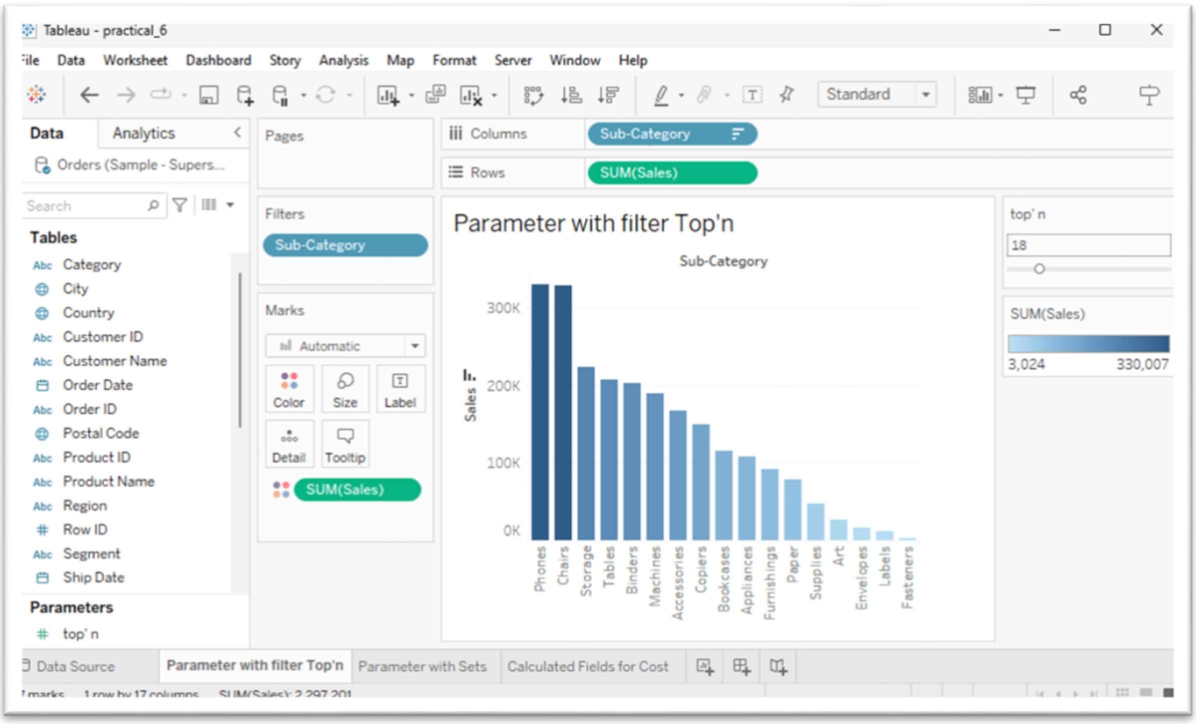
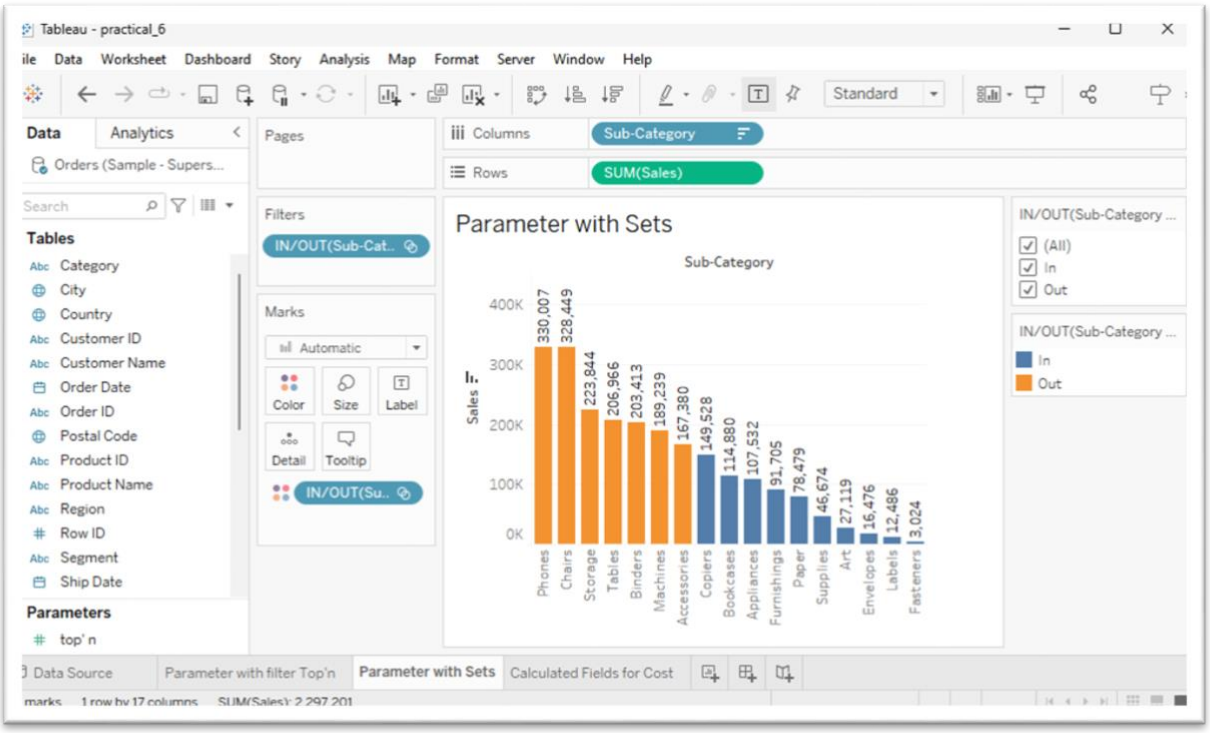


Tableau - practical_6
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Standard
Data Analytics
Orders (Sample - Supers...
Tables
Category
City
Country
Customer ID
Customer Name
Order Date
Order ID
Postal Code
Product ID
Product Name
Region
Row ID
Segment
Ship Date
Parameters
top' n
Pages
Columns Sub-Category
Rows SUM(Sales)
Filters Sub-Category
Marks Automatic
Color Size Label Detail Tooltip
SUM(Sales)
Parameter with filter Top'n
Sub-Category
Sales
300K
200K
100K
0K
Phones Chairs Storage Tables Binders Machines Accessories Copiers Bookcases Appliances Furnishings Paper Supplies Art Envelopes Labels Fasteners
top' n
18
SUM(Sales)
3,024 330,007
Data Source Parameter with filter Top'n Parameter with Sets Calculated Fields for Cost
1 row by 17 columns SUM(Sales): 2,297,201

II. Parameter with Sets



III. Calculated Fields

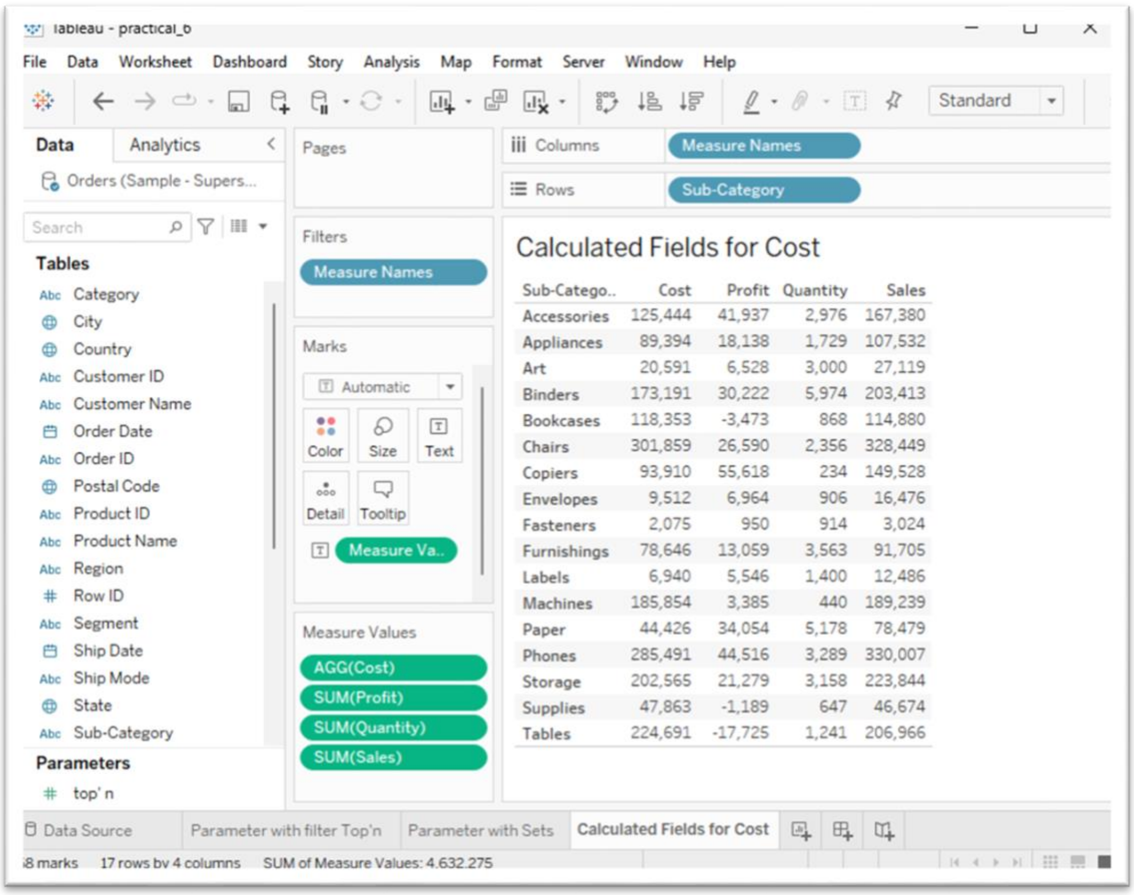


Tableau - practical_6
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Standard
Data Analytics
Orders (Sample - Supers...
Pages
Columns: Measure Names
Rows: Sub-Category
Filters: Measure Names
Marks: Automatic
Color Size Text Detail Tooltip
Measure Va..
Measure Values: AGG(Cost) SUM(Profit) SUM(Quantity) SUM(Sales)
Tables: Category, City, Country, Customer ID, Customer Name, Order Date, Order ID, Postal Code, Product ID, Product Name, Region, Row ID, Segment, Ship Date, Ship Mode, State, Sub-Category
Parameters: top' n
Data Source | Parameter with filter Top'n | Parameter with Sets | Calculated Fields for Cost

Calculated Fields for Cost

Sub-Catego..	Cost	Profit	Quantity	Sales
Accessories	125,444	41,937	2,976	167,380
Appliances	89,394	18,138	1,729	107,532
Art	20,591	6,528	3,000	27,119
Binders	173,191	30,222	5,974	203,413
Bookcases	118,353	-3,473	868	114,880
Chairs	301,859	26,590	2,356	328,449
Copiers	93,910	55,618	234	149,528
Envelopes	9,512	6,964	906	16,476
Fasteners	2,075	950	914	3,024
Furnishings	78,646	13,059	3,563	91,705
Labels	6,940	5,546	1,400	12,486
Machines	185,854	3,385	440	189,239
Paper	44,426	34,054	5,178	78,479
Phones	285,491	44,516	3,289	330,007
Storage	202,565	21,279	3,158	223,844
Supplies	47,863	-1,189	647	46,674
Tables	224,691	-17,725	1,241	206,966

68 marks 17 rows by 4 columns SUM of Measure Values: 4,632,275

PRACTICAL 07

AIM: - Designing Tableau Dashboards for different displays and devices.

THEORY

A dashboard is a collection of several views, letting you compare a variety of data simultaneously. For example, if you have a set of views that you review every day, you can create a dashboard that displays all the views at once, rather than navigate to separate worksheets. Device layouts appear on the Dashboard tab, under Default. Initially, each device layout contains every item in the Default dashboard and derives its size and layout from Default aswell. Think of the Default dashboard as the parent, and the device layouts (desktop, tablet, and phone) as its children. Any view, filter, action, legend or parameter that you want to add to adevice layout must first exist in the Default dashboard.

STEPS

1. Open a dashboard.
2. On the dashboard tab on the left, click Device Preview.
3. Take a moment to click through the device types and models and explore the different screen sizes. Then set these options:

a. To see how the dashboard will look in landscape vs. portrait mode, click on the option available. Usually, landscape is optimal for tablets and portrait is the best forphones.

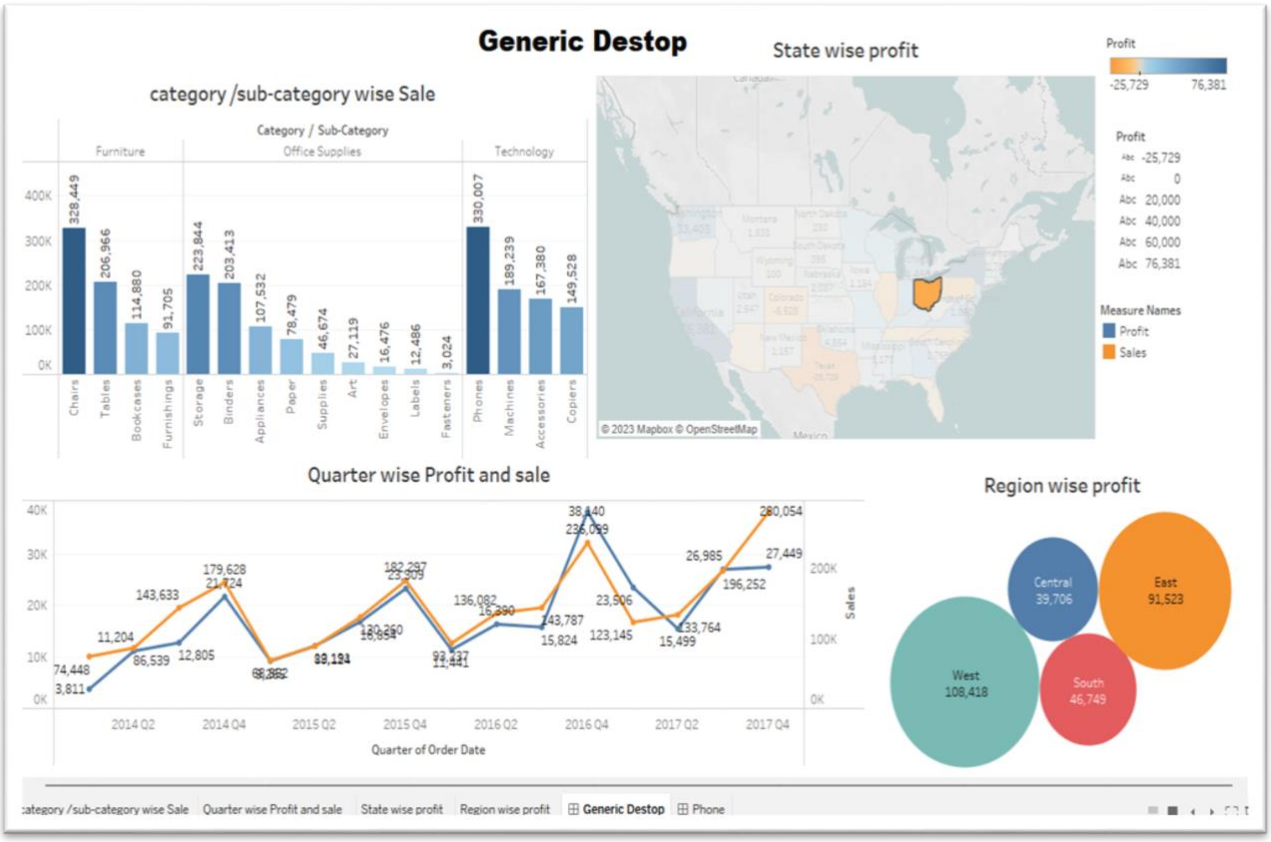
b. Select tableau mobile app to see how the dashboard will look with the app insteadof the browser. The option is available for IOS or Android devices and shrinks thedashboard slightly, leaving space for the app controls.

4. Choose a device type, such as tablet.
5. Click through the device model options to see how the layout will appear on differentmodels.
6. At left, explore the options under size.

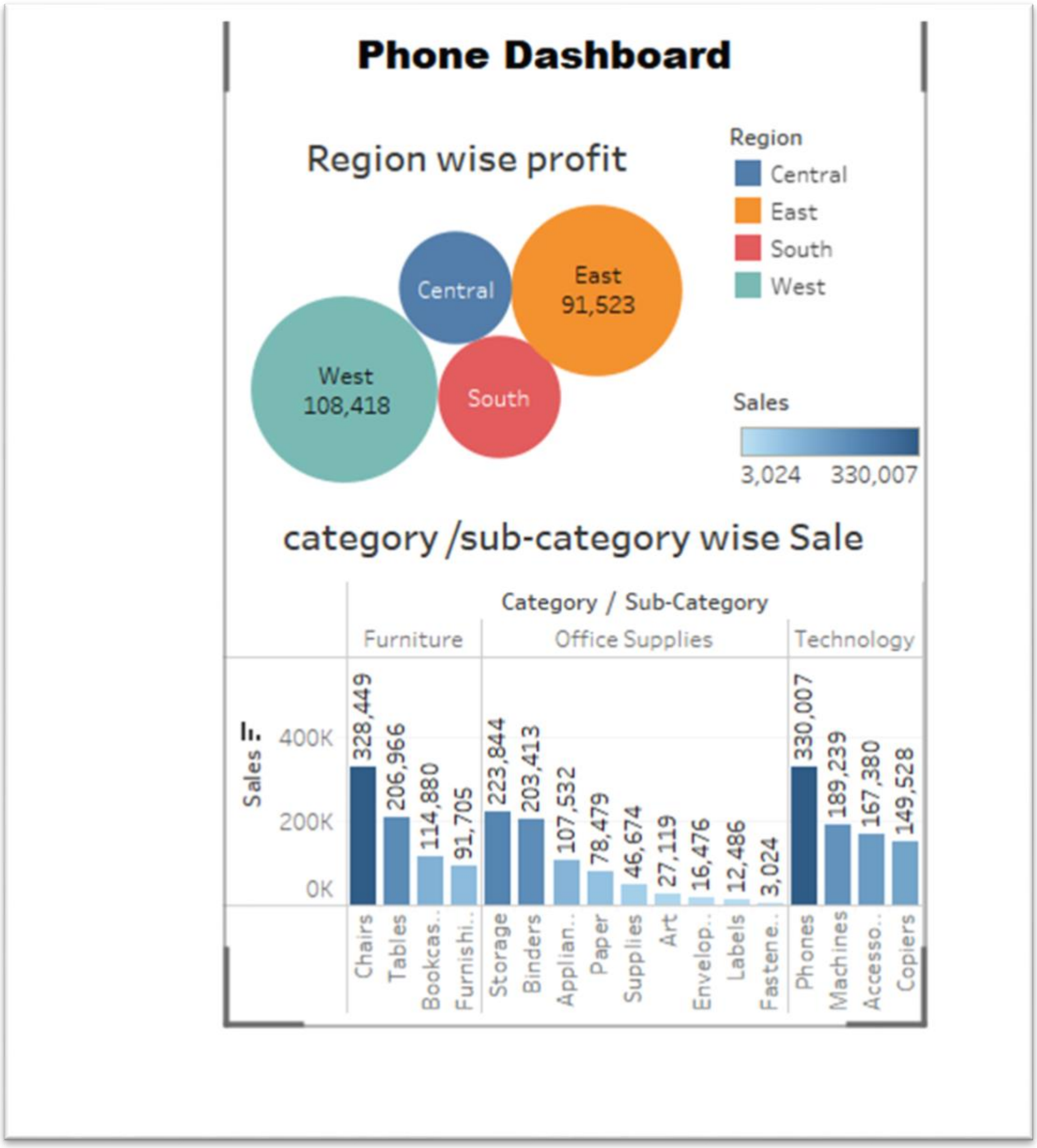
i. Default.
ii. Fit all
iii. Fit width

RESULTS

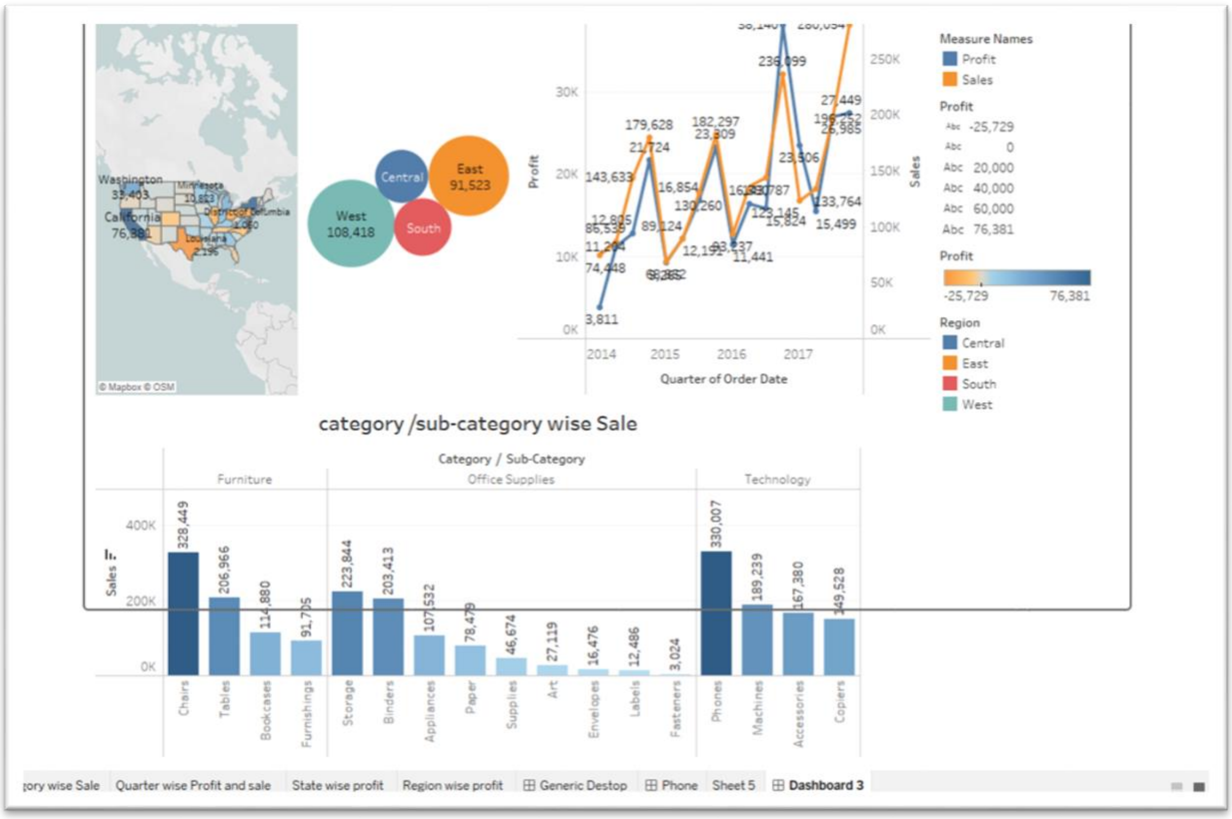
i) Desktop preview



ii) Phone preview



iii) Tablet preview



PRACTICAL 08

AIM: Create a Trend Model using data, analyze it and use it for forecasting.

THEORY

-

Forecasting in Tableau uses a technique known as exponential smoothing. Forecast algorithms try to find a regular pattern in measures that can be continued into the future. All forecast algorithms are simple models of a real-world data generating process (DGP). For a high quality forecast, a simple pattern in the DGP must match the pattern described by the model reasonably well. Quality metrics measure how well the model matches the DGP. If the quality is low, theprecision measured by the confidence bands is not important because it measures the precision of an inaccurate estimate.

STEPS

1. Connect to data
   a. In tableau desktop, connect to **superstore sample data** provided by tableau.
2. Create the visualization

a. Create a line chart with ship date (year) in the columns shelf and Sales in theRows shelf.

b. Go to the analysis tab and click on forecast under model category.

c. On completing the above step, we find the options to set various options for forecast.

d. Choose the forecast length as 2 years and leave the forecast model to automatic and then click ok.

e. We also get minute details of the forecast model by choosing the option describe forecast. To get this option, right-click on forecast diagram.

f. For trend model create a line chart with order date (year) in the columns and sales in the row shelf.

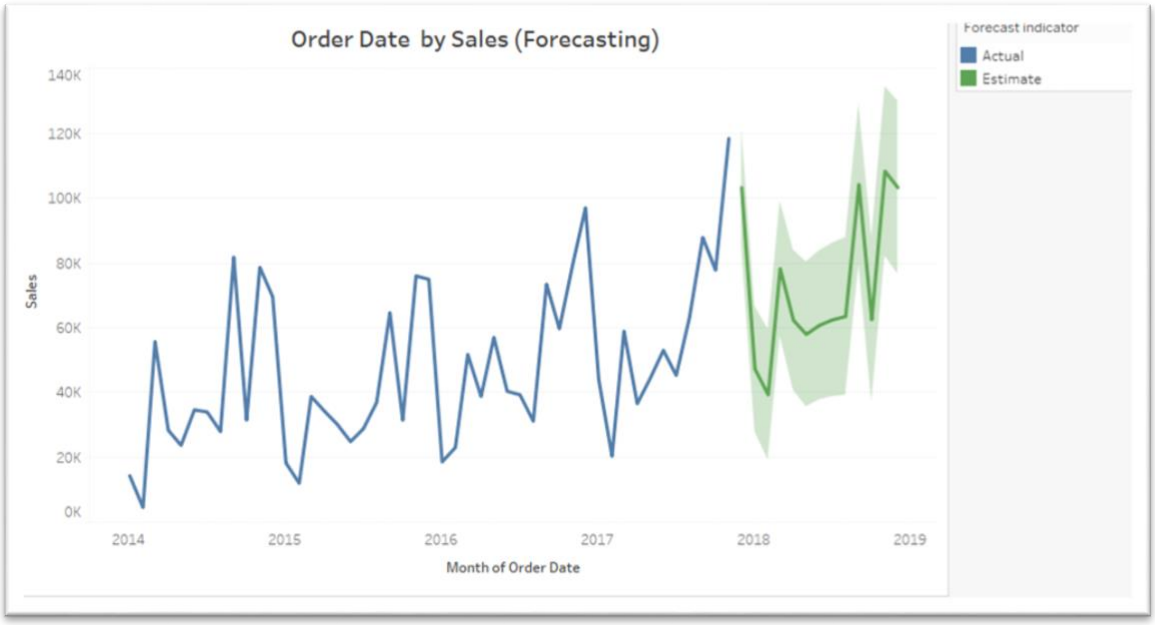
g. After the graph is completed we have to go to analysis tab and click on trend line under model category.

h. After the completion of above step we get a fine dashed line across the data line.

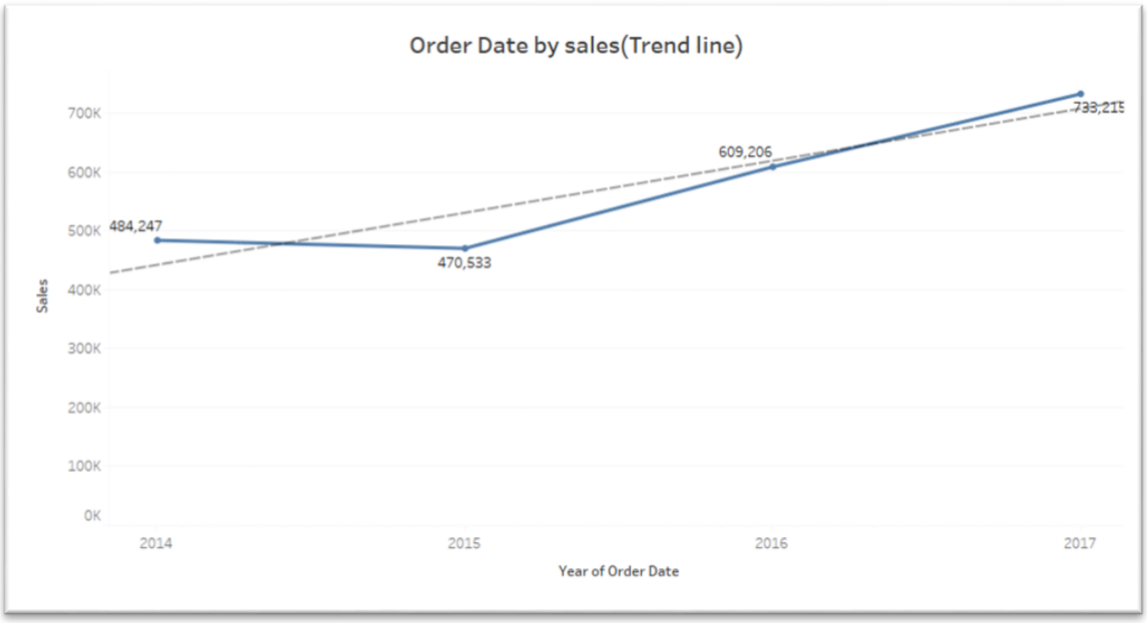
i. We can also get minute details of the trend line model by choosing the describe trend line. To get this option, right-click on the trend line.

RESULTS

Forecasting



Trend Model Line



PRACTICAL-09

AIM:-: Creating geospatial feature maps in tableau using geospatial data

THEORY

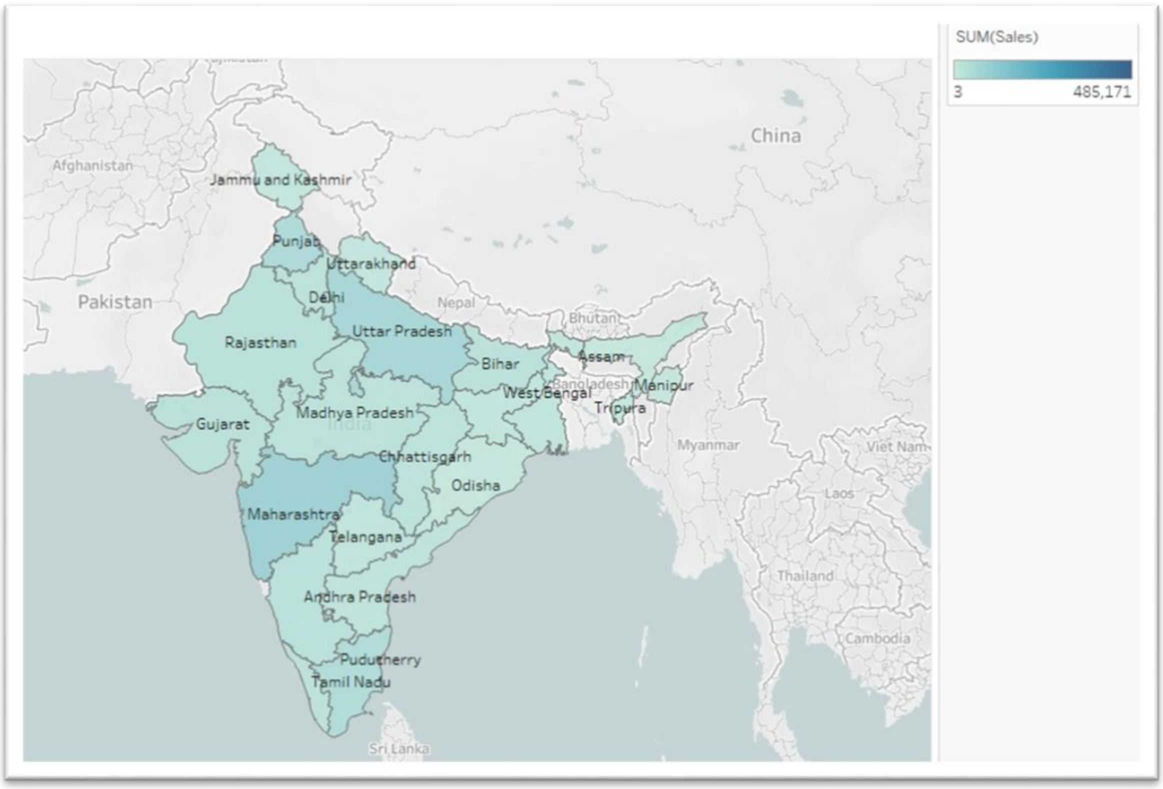
In Tableau Desktop, you can connect to the following spatial file types: Shape files, MapInfo tables, KML (Keyhole Mark-up Language) files, GeoJSON files, TopoJSON files, and Esri File Geodatabases. You can then create point, line, or polygon maps using the data in those files. With a Creator license in Tableau Cloud or Tableau Server, you can upload spatial file formats that only require one file (KML, GeoJSON, TopoJSON, Esri shape files packaged in a.zip, and Esri File Geodatabases with the extension .gdb.zip) in the Files tab when you create a new workbook and connect to data. In current versions of tableau, you can only connect to point geometrics, linear geometrics, or polygons. You cannot connect to mixed geometry types.

DATASET: The dataset speaks of different county/region with respect to family, happiness, generosity etc.

STEPS

1. In tableau desktop: click the new data source icon and select spatial file.
   OR in tableau cloud or tableau server: select create > workbook. Select the files tab.
2. Navigate to the folder that contains the spatial data, select the spatial file you want toconnect to, and then click open.
3. Drag and drop the country or state wise data we want to represent in the geospatialchart, into the sheet.
4. In the data pane, under the measures, the longitude and latitude is generated in thecolumns and rows respectively.
5. Drag and drop sum of happiness into colour under the marks date pane.
6. On the right the sum (happiness rank) exists and the colour can be changed.

RESULT


SUM(Sales)
3
485,171
China
Afghanistan
Pakistan
Nepal
Bhutan
Bangladesh
Myanmar
Viet Nam
Laos
Thailand
Cambodia
Sri Lanka
India
Jammu and Kashmir
Punjab
Uttarakhand
Delhi
Uttar Pradesh
Rajasthan
Bihar
Assam
Manipur
West Bengal
Tripura
Gujarat
Madhya Pradesh
Chhattisgarh
Odisha
Maharashtra
Telangana
Andhra Pradesh
Puducherry
Tamil Nadu

PRACTICAL 10

AIM: Create dashboard and storytelling using tableau.

THEORY

In Tableau, a **story** is a sequence of visualizations that work together to convey information. You can create stories to tell a data narrative, provide context, demonstrate how decisions relate to outcomes, or to simply make a compelling case. A story is a sheet, so the methods you use to create, name, and manage worksheets and dashboards also apply to stories. At the same time, a story is also a collection of sheets, arranged in a sequence. Each individual sheet in a story is called a **story point**. When you share a story —for example, by publishing a workbook to Tableau Public, Tableau Server, or Tableau Cloud—users can interact with the story to reveal new findingsor ask new questions of the data.

Dataset: The sample superstore dataset provided by tableau.

STEPS

1. Click the new story tab. Tableau opens a new story as your starting point.
2. By default, your story gets its title from the sheet name. To edit it, right-click the sheettab, and choose rename sheet.
3. To start building your story, double-click a sheet on the left to add it to a story point. In Tableau Desktop, you can also drag sheets into your story point. When you add a sheet to a story point, that sheet remains connected to the original sheet. If you modify the original sheet, your changes will automatically be reflected on the story points thatuse it.
4. Click Add a caption to summarize the story point.
5. To further highlight the main idea of this story point, you can change a filter or sort ona field in the view. Then save your changes by clicking **Update** on the story toolbar above the navigator box.
6. Add another story point by doing one of the following:

a. Click **Blank** to use a fresh sheet for the next story point.

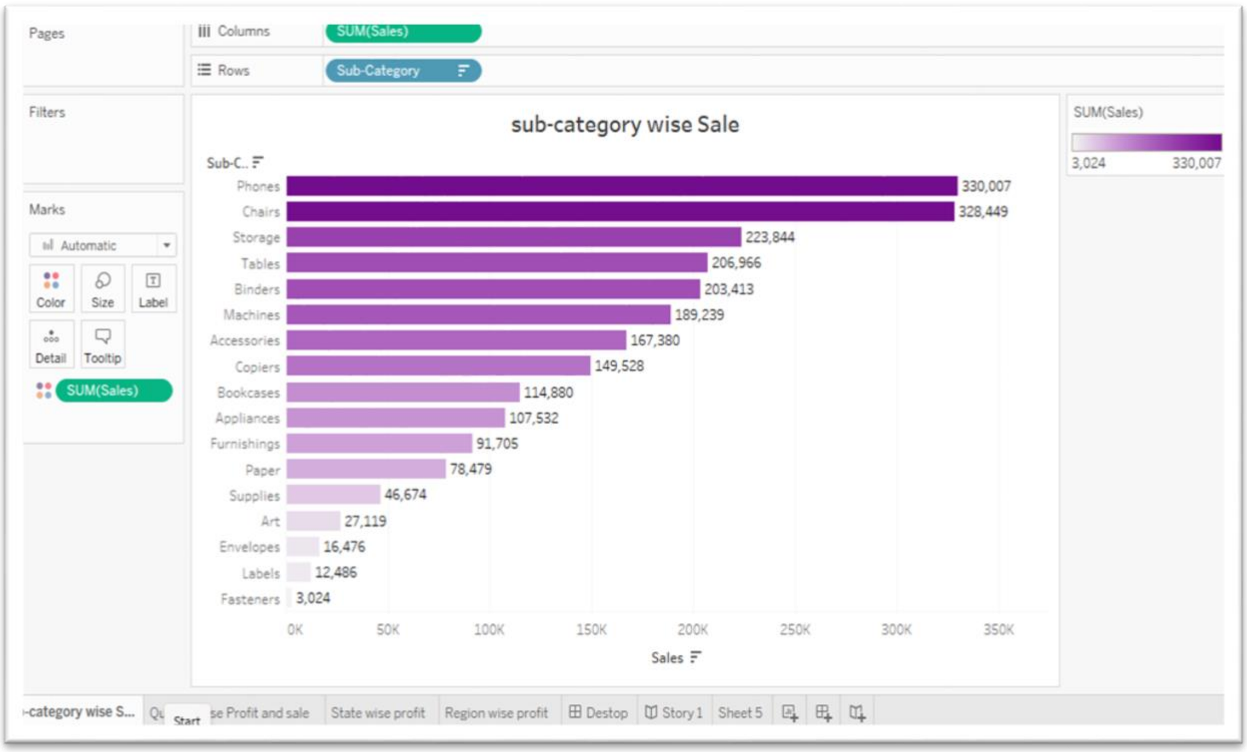
b. Start customizing a story point and click **Save as New** on the toolbar above the navigator box.

c. Click **Duplicate** to use the current story point as the basis for a new one.

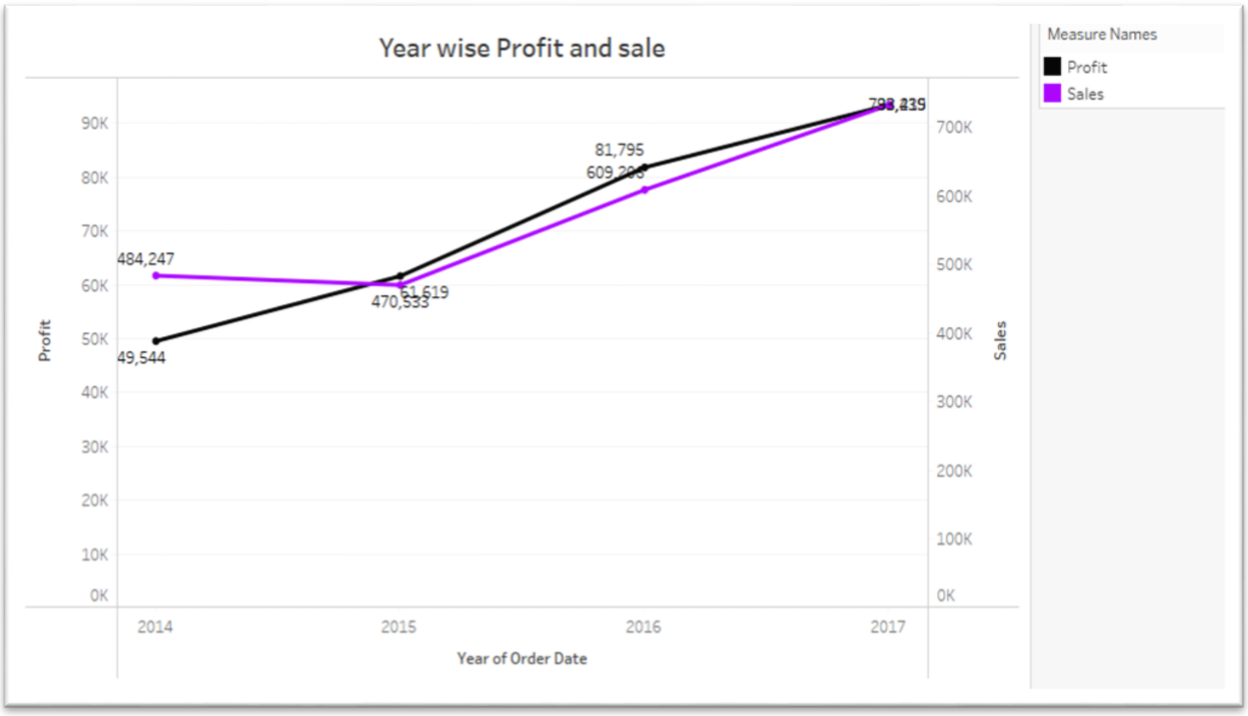
7. You can refine the look of your story using the options on the **Layout** tab.
a) Click the **Layout** tab.
b) Choose a navigator style that best suits your story, and show or hide the next and previous arrows.
8. Click the **Size** drop-down menu and select the story you want the dashboard to fitinside.

RESULTS

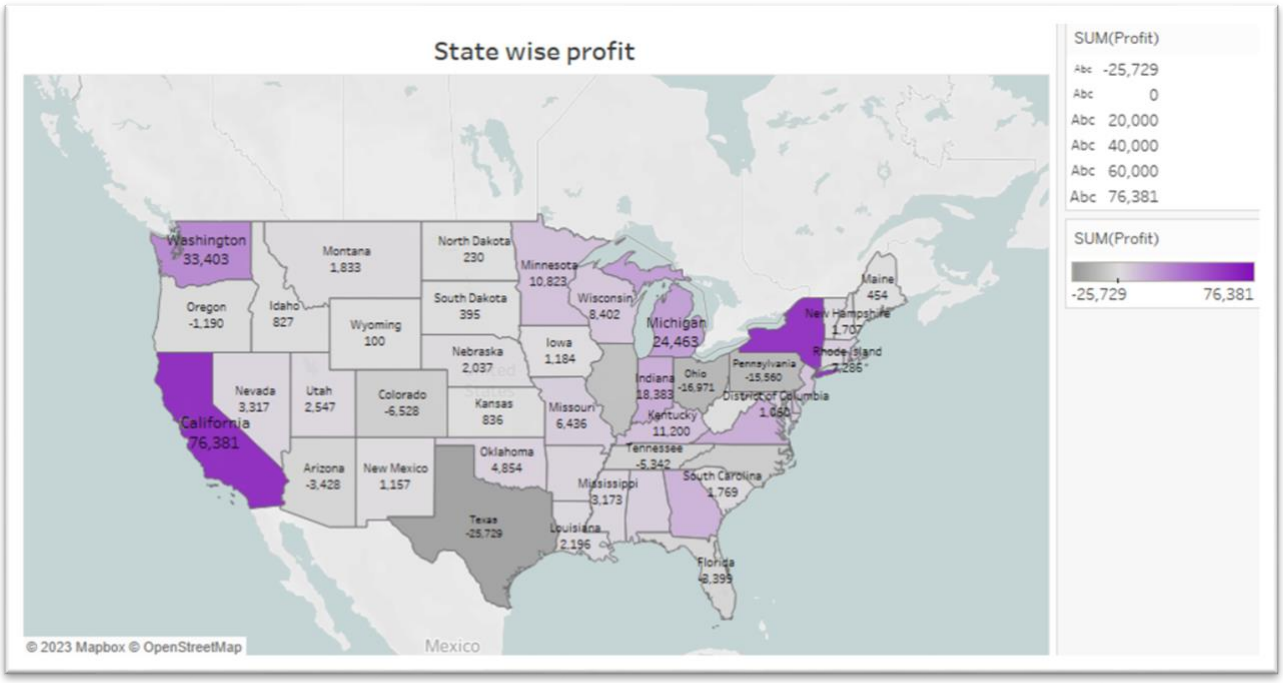
a) Sheet 01



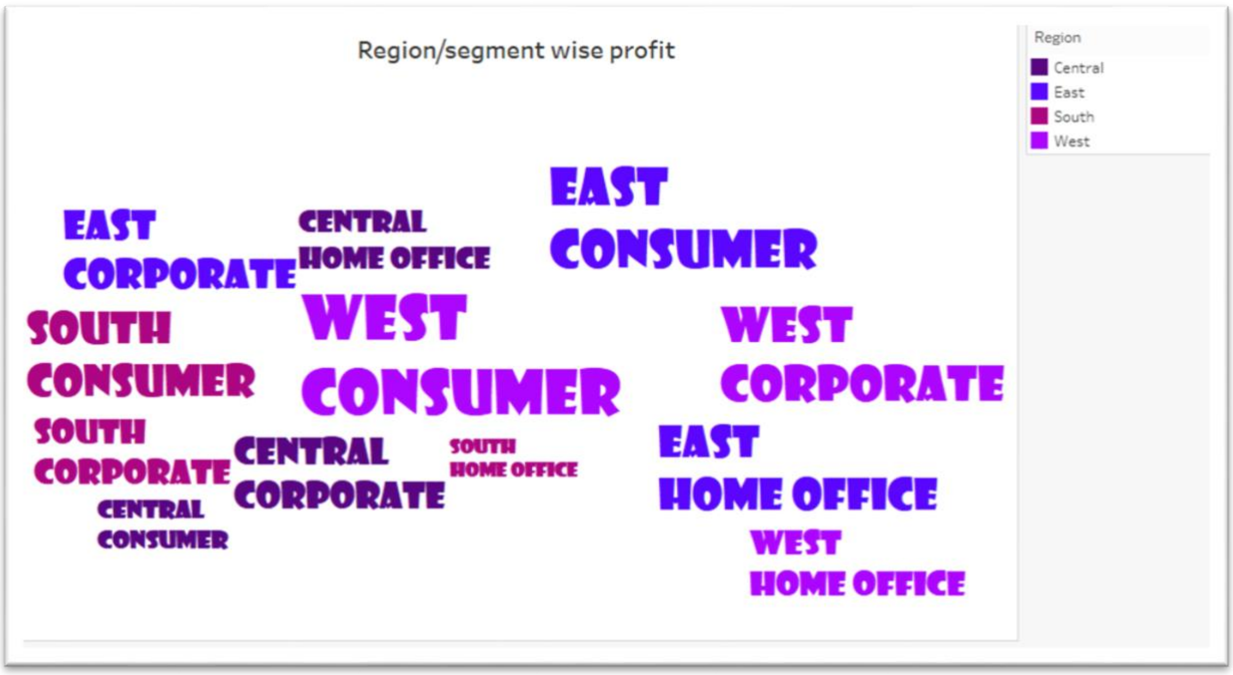
b) Sheet 02



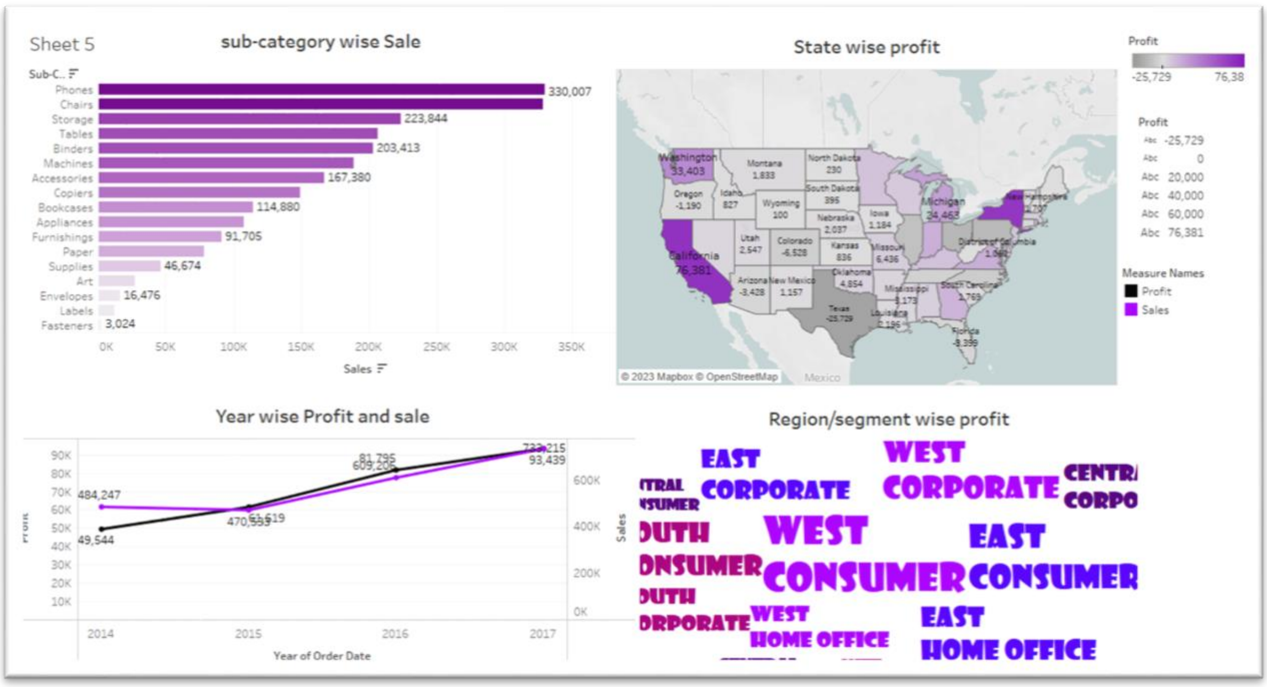
c) Sheet 03



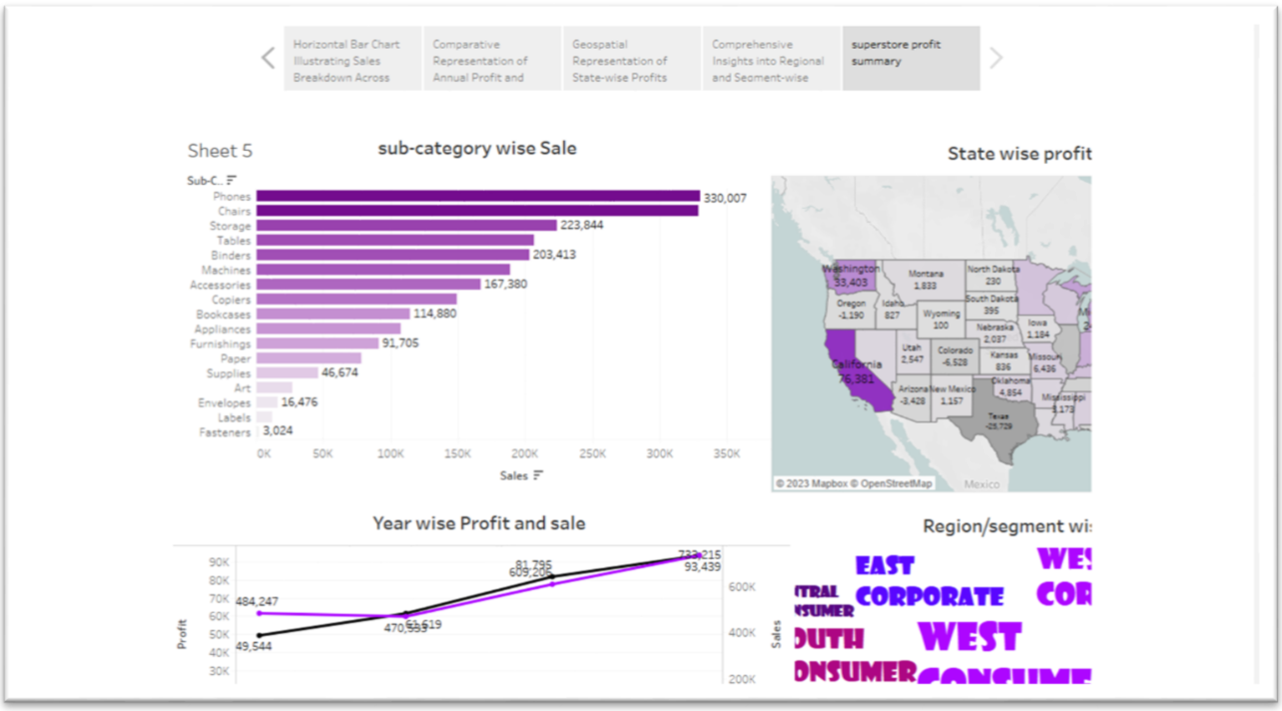
d) Sheet 04



e) Sheet 05



f) Sheet 06


Horizontal Bar Chart Illustrating Sales Breakdown Across
Comparative Representation of Annual Profit and
Geospatial Representation of State-wise Profits
Comprehensive Insights into Regional and Segment-wise
superstore profit summary
Sheet 5
sub-category wise Sale
State wise profit
Sub-C..
Phones
Chairs
Storage
Tables
Binders
Machines
Accessories
Copiers
Bookcases
Appliances
Furnishings
Paper
Supplies
Art
Envelopes
Labels
Fasteners
330,007
223,844
203,413
167,380
114,880
91,705
46,674
16,476
3,024
0K
50K
100K
150K
200K
250K
300K
350K
Sales
Washington 33,403
Montana 1,833
North Dakota 230
Oregon -1,190
Idaho 827
Wyoming 100
South Dakota 395
Nebraska 2,037
Iowa 1,184
California 76,381
Utah 2,547
Colorado -6,528
Kansas 836
Missouri 6,436
Arizona -3,428
New Mexico 1,157
Oklahoma 4,854
Mississippi 3,172
Texas -25,729
© 2023 Mapbox © OpenStreetMap
Mexico
Year wise Profit and sale
Profit
Sales
90K
80K
70K
60K
50K
40K
30K
600K
400K
200K
484,247
49,544
470,533
61,619
81,795
609,206
733,215
93,439
Region/segment wi
EAST
WES
CORPORATE
COR
TRAL
NSUMER
OUTH
WEST
ONSUMER
CONSUME