Course Code PSDSP504	Course Title Data Analysis and Visualization Practical	DATE	REMARK
1	Implement Data Loading, Storage and File Formats. Read data and store them in text format.	20/07/24	
2	Implement the code to interact with Web APIs and to perform web scrapping.	27/07/24	
3	Demonstrate Data Cleaning and Preparation.	03/08/24	
4	Implement Data wrangling on a data set.	10/08/24	
5	Demonstrate the handling of missing data and string manipulation.	17/08/24	
6	Create common charts with title, labels and descriptions using Tableau.	24/08/24	
7	Perform sorting and filtering using tableau, create visualizations and publish it on Tableau Cloud.	31/08/24	
8	Perform data visualization using Power BI.	14/09/24	
9	Create reports using Power BI.	28/09/24	
10	Create a data story in Tableau or power BI.	05/10/24	

PRACTICAL NO:01

AIM: Implement Data Loading, Storage and File Formats. Read data and store them in text format.

CODE:

```
In [1]: import pandas as pd
        df = pd.read_csv('prac1.csv')
        df
```

Out[1]:

	name	department	salary
0	jay	sales	255500
1	max	backend	49999
2	rayan	sales	45555

```
In [2]: pd.read_table('prac1.csv',sep=',')
```

Out[2]:

	name	department	salary
0	jay	sales	255500
1	max	backend	49999
2	rayan	sales	45555

```
In [3]: pd.read_csv('prac1.csv',header=None)
```

Out[3]:

	0	1	2
0	name	department	salary
1	jay	sales	255500
2	max	backend	49999
3	rayan	sales	45555

```
In [4]: pd.read_csv('prac1.csv',names=['a','b','c'])
```

Out[4]:

	a	b	c
0	name	department	salary
1	jay	sales	255500
2	max	backend	49999
3	rayan	sales	45555

```
In [6]: pd.read_csv('prac1.csv',names=names,index_col='a')
```

Out[6]:

	b	c
a		
name	department	salary
jay	sales	255500
max	backend	49999
rayan	sales	45555

```
In [7]: list(open('prac.txt'))
Out[7]: ['product\tquality\tsales\n',
         'apple\tA\t25\n',
         'mango\tB\t66\n',
         'kiwi\tA\t43\n']
```

```
In [9]: result = pd.read_table('prac.txt', sep=',')
        result
```

Out[9]:

	product\tquality\tsales
0	apple\tA\t25
1	mango\tB\t66
2	kiwi\tA\t43

```
In [10]: pd.read_csv('prac1.csv', skiprows=[0, 2])
```

Out[10]:

	jay	sales	255500
0	rayan	sales	45555

```
In [11]: result = pd.read_csv('prac1b.csv')
         result
```

Out[11]:

	name	area	gender
0	ray	newyork	male
1	alexa	NaN	NaN
2	maximillion	france	male
3	sara	NaN	female

```
In [12]: pd.isnull(result)
```

Out[12]:

	name	area	gender
0	False	False	False
1	False	True	True
2	False	False	False
3	False	True	False

```
In [13]: result = pd.read_csv('prac1b.csv', na_values=['NULL'])
```

```
In [14]: result
```

Out[14]:

	name	area	gender
0	ray	newyork	male
1	alexa	NaN	NaN
2	maximillion	france	male
3	sara	NaN	female

```
In [15]: sentinels = {'name': ['alexa', 'NA'], 'area': ['france']}
         pd.read_csv('prac1b.csv', na_values=sentinels)
```

Out[15]:

	name	area	gender
0	ray	newyork	male
1	NaN	NaN	NaN
2	maximillion	NaN	male
3	sara	NaN	female

```
In [16]: chunker = pd.read_csv('prac1b.csv', chunksize=1000)
         chunker
Out[16]: <pandas.io.parsers.readers.TextFileReader at 0x210951b0750>
```

```
In [17]: data = pd.read_csv('prac1b.csv')
         data
```

Out[17]:

	name	area	gender
0	ray	newyork	male
1	alexa	NaN	NaN
2	maximillion	france	male
3	sara	NaN	female

```
In [18]: data.to_csv('prac1b')
```

```
In [19]: import sys
         data.to_csv(sys.stdout, sep='|')
```

```
|name|area|gender
0|ray|newyork|male
1|alexa||
2|maximillion|france|male
3|sara||female
```

```
In [20]: data.to_csv(sys.stdout, na_rep='NULL')
```

```
,name,area,gender
0,ray,newyork,male
1,alexa,NULL,NULL
2,maximillion,france,male
3,sara,NULL,female
```

```
In [21]: data.to_csv(sys.stdout, index=False, header=False)
```

```
ray,newyork,male
alexa,,
maximillion,france,male
sara,,female
```

```
In [23]: data.to_csv(sys.stdout, index=False, columns=['name', 'area', 'gender'])
```

```
name,area,gender
ray,newyork,male
alexa,,
maximillion,france,male
sara,,female
```

PRACTICAL NO:02

AIM: Implement the code to interact with Web APIs and to perform web scrapping.

CODE:

A.Interacting with a Web API

1. Send an HTTP GET request to the API endpoint
2. Parse the JSON response

```
In [1]: import requests

In [2]: from bs4 import BeautifulSoup

In [3]: api_url = "https://api.github.com"

In [4]: response = requests.get(api_url)

In [5]: if response.status_code == 200:
            data = response.json()
            print("Data from API:")
            print(data)
        else:
            print("Failed to fetch data from the API")
```

OUTPUT:

```
Data from API:
{'current_user_url': 'https://api.github.com/user', 'current_user_authorizations_html_url': 'https://github.com/settings/connec
tions/applications{/client_id}', 'authorizations_url': 'https://api.github.com/authorizations', 'code_search_url': 'https://ap
i.github.com/search/code?q={query}{&page,per_page,sort,order}', 'commit_search_url': 'https://api.github.com/search/commits?q=
{query}{&page,per_page,sort,order}', 'emails_url': 'https://api.github.com/user/emails', 'emojis_url': 'https://api.github.com/
emojis', 'events_url': 'https://api.github.com/events', 'feeds_url': 'https://api.github.com/feeds', 'followers_url': 'https://
api.github.com/user/followers', 'following_url': 'https://api.github.com/user/following{/target}', 'gists_url': 'https://api.gi
thub.com/gists{/gist_id}', 'hub_url': 'https://api.github.com/hub', 'issue_search_url': 'https://api.github.com/search/issues?q
={query}{&page,per_page,sort,order}', 'issues_url': 'https://api.github.com/issues', 'keys_url': 'https://api.github.com/user/k
eys', 'label_search_url': 'https://api.github.com/search/labels?q={query}&repository_id={repository_id}{&page,per_page}', 'noti
fications_url': 'https://api.github.com/notifications', 'organization_url': 'https://api.github.com/orgs/{org}', 'organization_
repositories_url': 'https://api.github.com/orgs/{org}/repos{?type,page,per_page,sort}', 'organization_teams_url': 'https://api.
github.com/orgs/{org}/teams', 'public_gists_url': 'https://api.github.com/gists/public', 'rate_limit_url': 'https://api.github.
com/rate_limit', 'repository_url': 'https://api.github.com/repos/{owner}/{repo}', 'repository_search_url': 'https://api.github.
com/search/repositories?q={query}{&page,per_page,sort,order}', 'current_user_repositories_url': 'https://api.github.com/user/re
pos{?type,page,per_page,sort}', 'starred_url': 'https://api.github.com/user/starred{/owner}{/repo}', 'starred_gists_url': 'http
s://api.github.com/gists/starred', 'topic_search_url': 'https://api.github.com/search/topics?q={query}{&page,per_page}', 'user_
url': 'https://api.github.com/users/{user}', 'user_organizations_url': 'https://api.github.com/user/orgs', 'user_repositories_u
rl': 'https://api.github.com/users/{user}/repos{?type,page,per_page,sort}', 'user_search_url': 'https://api.github.com/search/u
sers?q={query}{&page,per_page,sort,order}'}
```

B. Perform web scrapping

1. Send an HTTP GET request to the website
2. Check if the request was successful
3. Parse the HTML content of the page
4. Find and print the title of the page

CODE:

```
In [1]: import requests
        from bs4 import BeautifulSoup

In [2]: web_url="https://www.w3School.com"

In [3]: response = requests.get(web_url)

In [6]: if response.status_code == 200:
            html_content = response.text
            soup = BeautifulSoup(html_content, 'html.parser')
            title_tag = soup.find('title')
            if title_tag:
                title = title_tag.get_text()
                print("\nTitle of the OpenAI blog page:")
                print(title)
            else:
                print("Title not found on the page")
        else:
            print("Failed to fetch data from the website")

        Title of the OpenAI blog page:
        Mega site of Bible Information
```

PRACTICAL NO:03

AIM: Demonstrate Data Cleaning and Preparation.

CODE:

```
import pandas as pd

data={'name':['jack','jay','tom','alice','sally'],'age':[None,34,32,None,33],'salary':[40000,None,
600000,

46777,80000]}

df=pd.DataFrame(data)

print("Missing Values:")
```

```
Missing Values:
```

```
print(df.isnull())
```

```
    name    age  salary
0  False   True   False
1  False  False    True
2  False  False   False
3  False   True   False
4  False  False   False
```

```
df_cleaned=df.dropna()

print("\nDataframe after droppping rows with the missing values:")

print(df_cleaned)
```

```
 Dataframe after droppping rows with the missing values:
    name   age    salary
2    tom  32.0  600000.0
4  sally  33.0   80000.0
```

```
df_filled=df.fillna({'age':df['age'].mean(),'salary':df['salary'].median()})

print("\nDatFrame after fillling missing values:")

print(df_filled)
```

```
DatFrame after fillling missing values:
    name   age    salary
0   jack  33.0   40000.0
1    jay  34.0   63388.5
2    tom  32.0  600000.0
3  alice  33.0   46777.0
4  sally  33.0   80000.0
```

print("\nDupilicated rows:")

print(df.duplicated())

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

df_no_duplicates=df.drop_duplicates()

print("\nDataframe after removing duplicates:")

print(df_no_duplicates)

```
  Dataframe after removing duplicates:
      name   age    salary
  0   jack   NaN   40000.0
  1    jay  34.0       NaN
  2    tom  32.0  600000.0
  3  alice   NaN   46777.0
  4  sally  33.0   80000.0
```

df_renamed=df.rename(columns={'name':'full name'})

print(df_renamed)

```
  full name   age    salary
0      jack   NaN   40000.0
1       jay  34.0       NaN
2       tom  32.0  600000.0
3     alice   NaN   46777.0
4     sally  33.0   80000.0
```

df_reordered=df[['salary','age','name']]

print(df_reordered)

```
     salary   age   name
0   40000.0   NaN   jack
1       NaN  34.0    jay
2  600000.0  32.0    tom
3   46777.0   NaN  alice
4   80000.0  33.0  sally
```

PRACTICAL NO:04

AIM: Implement Data wrangling on a data set.

CODE:

```
import pandas as pd

a=pd.read_csv('SALARY.csv')

print(a)
```

```
    NAME   SALARY GENDER  AGE
0    JAY  50000.0      F   43
1  RADAN      NaN      M   33
2    LEX  56000.0    NaN   23
```

```
a.isnull()
```

	NAME	SALARY	GENDER	AGE
0	False	False	False	False
1	False	True	False	False
2	False	False	True	False

```
a.isnull().any()
```

```
NAME       False
SALARY      True
GENDER      True
AGE        False
dtype: bool
```

```
a.dropna()
```

	NAME	SALARY	GENDER	AGE
0	JAY	50000.0	F	43

```
a.fillna("")
```

	NAME	SALARY	GENDER	AGE
0	JAY	50000.0	F	43
1	RADAN		M	33
2	LEX	56000.0		23

```
a['AREA']=['urban','rural','urban']

a
```

	NAME	SALARY	GENDER	AGE	area	AREA
0	JAY	50000.0	F	43	urban	urban
1	RADAN	NaN	M	33	rural	rural
2	LEX	56000.0	NaN	23	urban	urban

a['GENDER']=a['GENDER'].map({'M':0,'F':1,}).astype(float)

a

	NAME	SALARY	GENDER	AGE	area	AREA
0	JAY	50000.0	NaN	43	urban	urban
1	RADAN	NaN	NaN	33	rural	rural
2	LEX	56000.0	NaN	23	urban	urban

a.groupby('NAME').GENDER.value_counts()

```
Series([], Name: count, dtype: int64)
```

a.GENDER.unique()

```
array([nan])
```

a[a['SALARY']>50000]

	NAME	SALARY	GENDER	AGE	area	AREA
2	LEX	56000.0	NaN	23	urban	urban

a[a['AREA']=='urban']

	NAME	SALARY	GENDER	AGE	area	AREA
0	JAY	50000.0	NaN	43	urban	urban
2	LEX	56000.0	NaN	23	urban	urban

b=pd.DataFrame({'DEPARTMENT':['RESEARCH','ANALYSE','RESEARCH'],'NAME':['JAY','RADAN','LEX']})

b

	DEPARTMENT	NAME
0	RESEARCH	JAY
1	ANALYSE	RADAN
2	RESEARCH	LEX

c=print(pd.merge(a,b,on='NAME'))

c

	NAME	SALARY	GENDER	AGE	area	AREA	DEPARTMENT
0	JAY	50000.0	NaN	43	urban	urban	RESEARCH
1	RADAN	NaN	NaN	33	rural	rural	ANALYSE
2	LEX	56000.0	NaN	23	urban	urban	RESEARCH

a.pivot(index='AGE',columns='GENDER',values='SALARY')

GENDER	NaN
AGE	
23	56000.0
33	NaN
43	50000.0

A

	NAME	SALARY	GENDER	AGE	area	AREA
0	JAY	50000.0	NaN	43	urban	urban
1	RADAN	NaN	NaN	33	rural	rural
2	LEX	56000.0	NaN	23	urban	urban
3	RADAN	none	M	33	rural	rural

cow=pd.DataFrame([{'NAME':'RADAN','SALARY':
'none','GENDER':'M','AGE':33,'area':'rural','AREA':'rural'}])

a=pd.concat([a,cow],ignore_index=True)

a

	NAME	SALARY	GENDER	AGE	area	AREA
0	JAY	50000.0	NaN	43	urban	urban
1	RADAN	NaN	NaN	33	rural	rural
2	LEX	56000.0	NaN	23	urban	urban
3	RADAN	none	M	33	rural	rural
4	RADAN	none	M	33	rural	rural

a.drop_duplicates()

	NAME	SALARY	GENDER	AGE	area	AREA
0	JAY	50000.0	NaN	43	urban	urban
1	RADAN	NaN	NaN	33	rural	rural
2	LEX	56000.0	NaN	23	urban	urban
3	RADAN	none	M	33	rural	rural

PRACTICAL NO-05

AIM: Demonstrate the handling of missing data and string manipulation.

CODE:

In [1] : val = ‘a,b, guido’

In [2] : val.split(‘,’)

```
Out[2]: ['a', 'b', ' guido']
```

In [3] : pieces = [x.strip() for x in val.split(‘,’)]

In [4] : pieces

```
Out[4]: ['a', 'b', 'guido']
```

In [5] : first, second, third = pieces

In [6] : first :: + second + :: + third

```
Out[6]: 'a::b::guido'
```

In [7] : ‘::’.join(pieces)

```
Out[7]: 'a::b::guido'
```

In [8] : ‘guido’in val

```
Out[8]: True
```

In [9] : val.index(‘,’)

```
Out[9]: 1
```

In [10] : val.find(‘:’)

```
Out[10]: -1
```

In [11] : val.index(‘:’)

```
---------------------------------------------------------------
-----------------
ValueError                                Traceback (most
recent call last)
Cell In[11], line 1
----> 1 val.index(':')

ValueError: substring not found
```

In [12] : val.count(‘,’)

```
Out[12]: 2
```

In [13] : val.replace(‘,’, ‘::’)

```
Out[13]: 'a::b:: guido'
```

In [14] : val.replace(‘,’, “”)

```
Out[14]: 'ab guido'
```

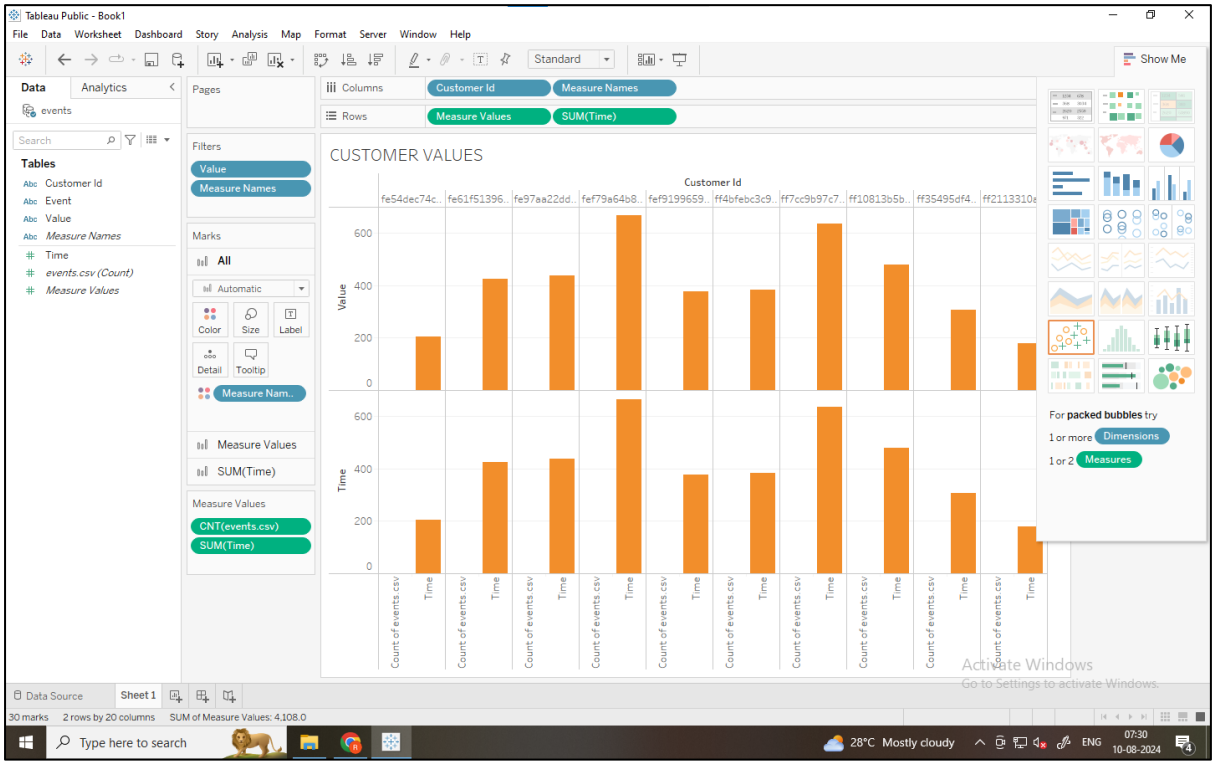
PRACTICAL NO :06

AIM: Create common charts with title, labels and descriptions using Tableau.

Steps:

1. Right click on the worksheet title.
2. Select "Edit title"
3. In the dialog box that appears create the title that you require, inserting any of the fields to customise the title from the "Insert" drop-down menu. In the above case, my title reads: ...
4. Hit "OK", and you are done.

OUTPUT:



DISCRIPTION:

Sheet Description

Description of "Sheet 1"

Time, count of events.csv and Time for each Customer Id. Color shows details about Time and count of events.csv. The data is filtered on Value, which keeps 10 of 5,121 members.

Measure Values Properties

Marks

The mark type is Bar (Automatic).
Stacked marks is on.

Shelves

Rows: Measure Values, Time
Columns: Customer Id, Measure Names
Filters: Value, Measure Names
Color: Measure Names

Sum of Time Properties

Marks

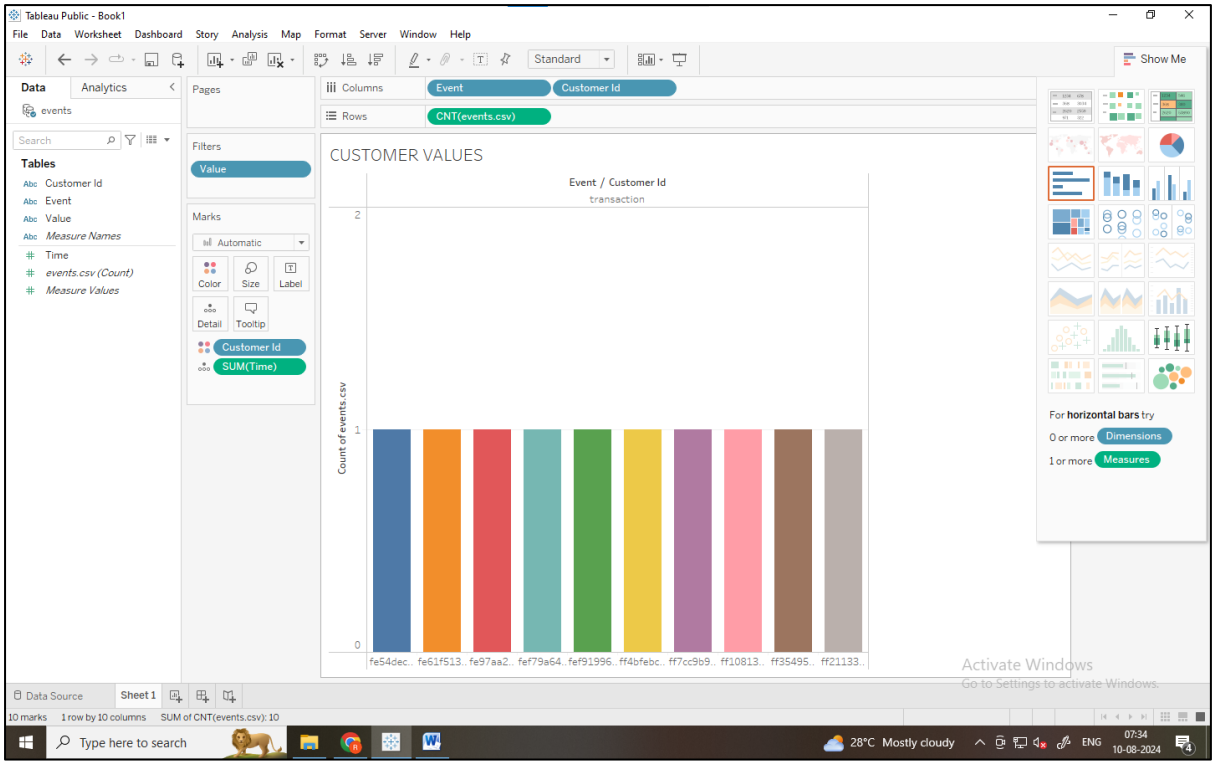
The mark type is Bar (Automatic).
Stacked marks is on.

Shelves

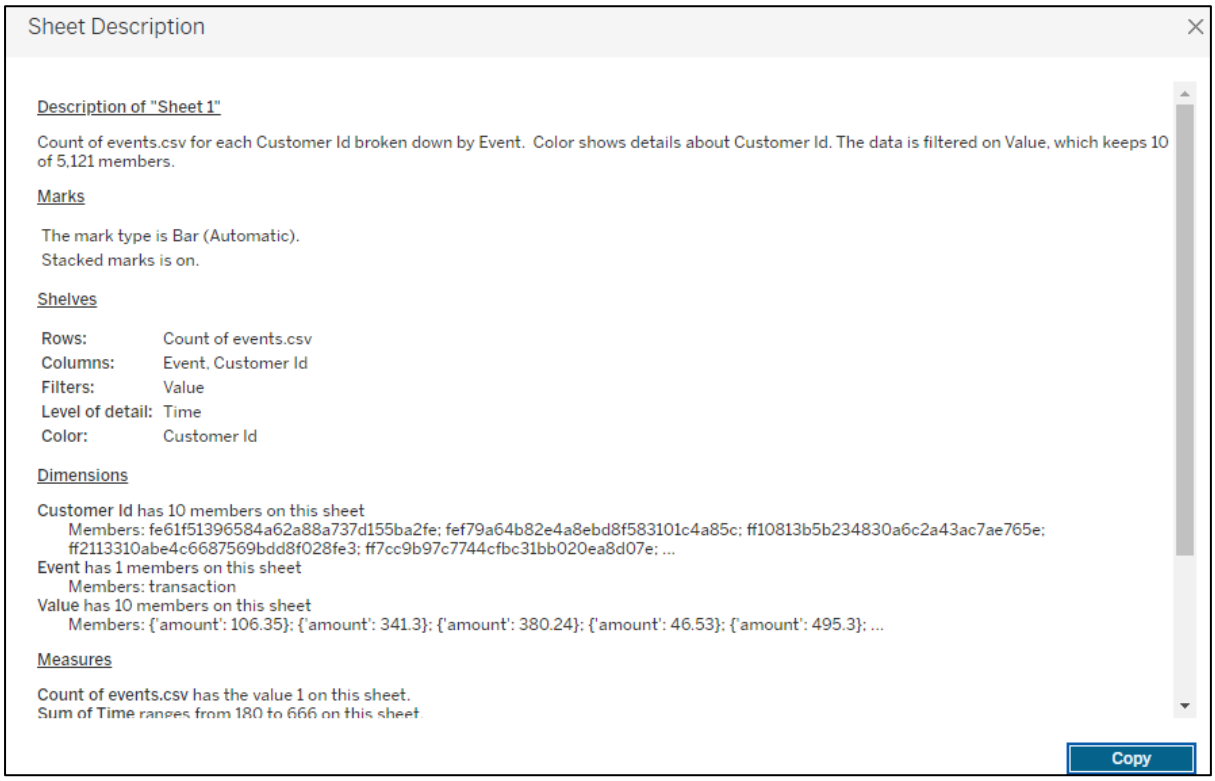
Rows: Measure Values, Time
Columns: Customer Id, Measure Names
Filters: Value, Measure Names

Copy

OUTPUT:



DISCRIPTION:



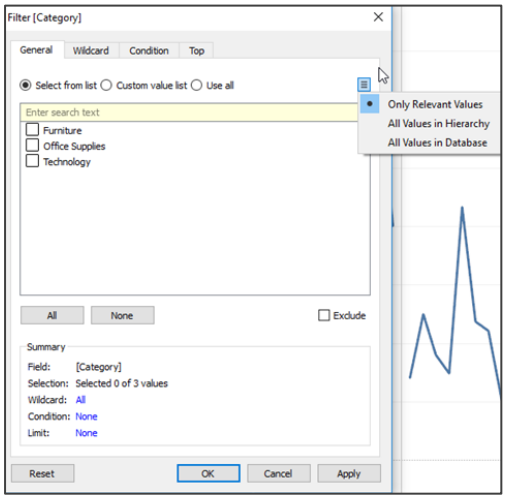
PRACTICAL NO:07

AIM: Perform sorting and filtering using tableau, create visualizations and publish it on Tableau Cloud.

Steps:

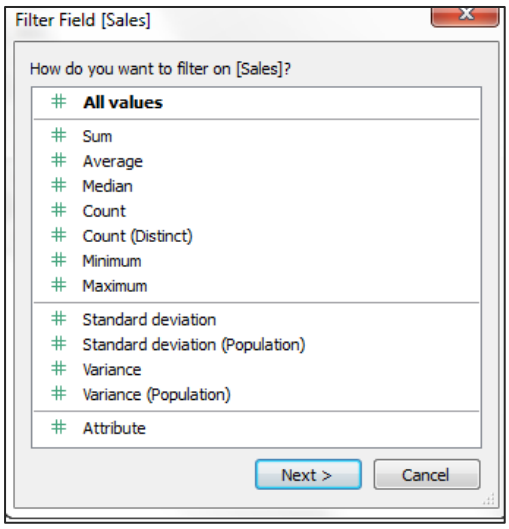
A. Filter categorical data (dimensions):

1. Dimensions contain discrete categorical data, so filtering this type of field generally involves selecting the values to include or exclude.
2. When you drag a dimension from the Data pane to the Filters shelf in Tableau Desktop, the following Filter dialog box appears:



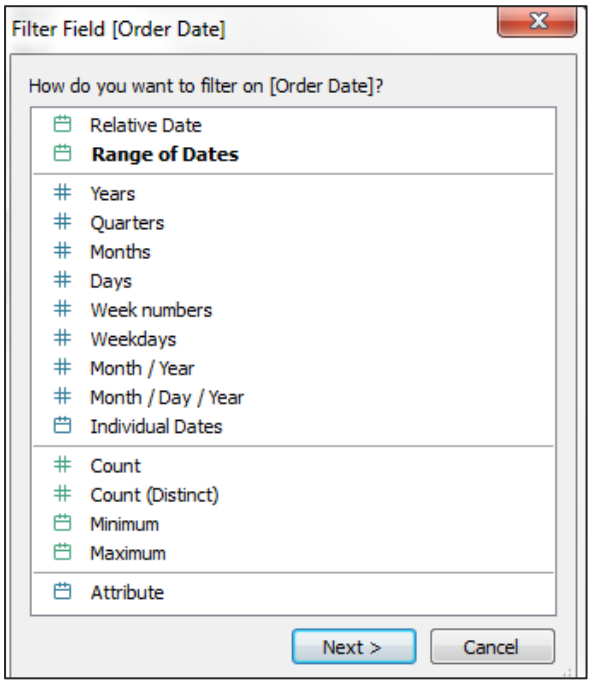
B. Filter Measures are the filters applied on the measure fieldS:

1. Measures contain quantitative data, so filtering this type of field generally involves selecting a range of values that you want to include.
2. When you drag a measure from the Data pane to the Filters shelf in Tableau Desktop, the following dialog box appears:



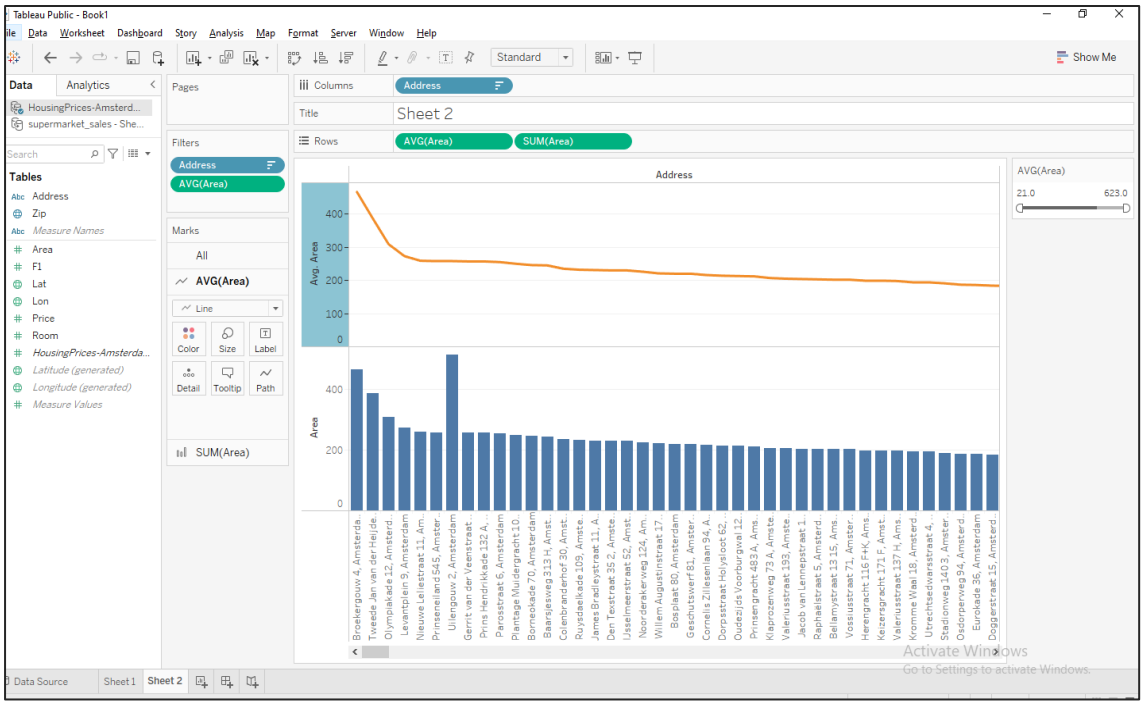
C. Filter Dates are the filters applied on the date fields.

1. When you drag a date field from the Data pane to the Filters shelf in Tableau Desktop, the following Filter Field dialog box appears:

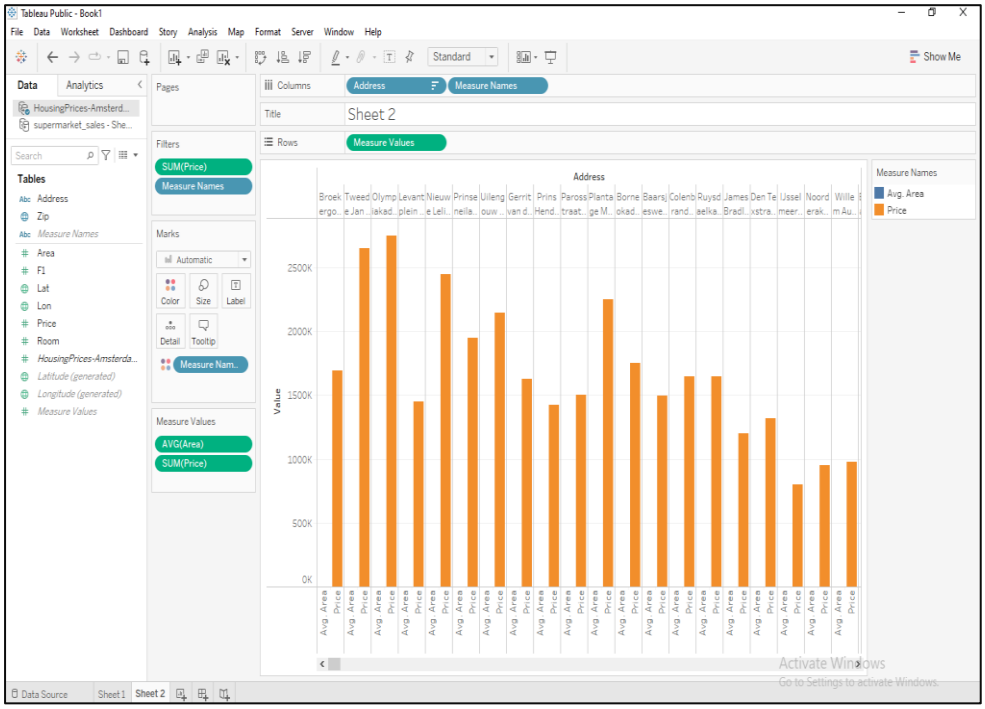


EXECUTION:

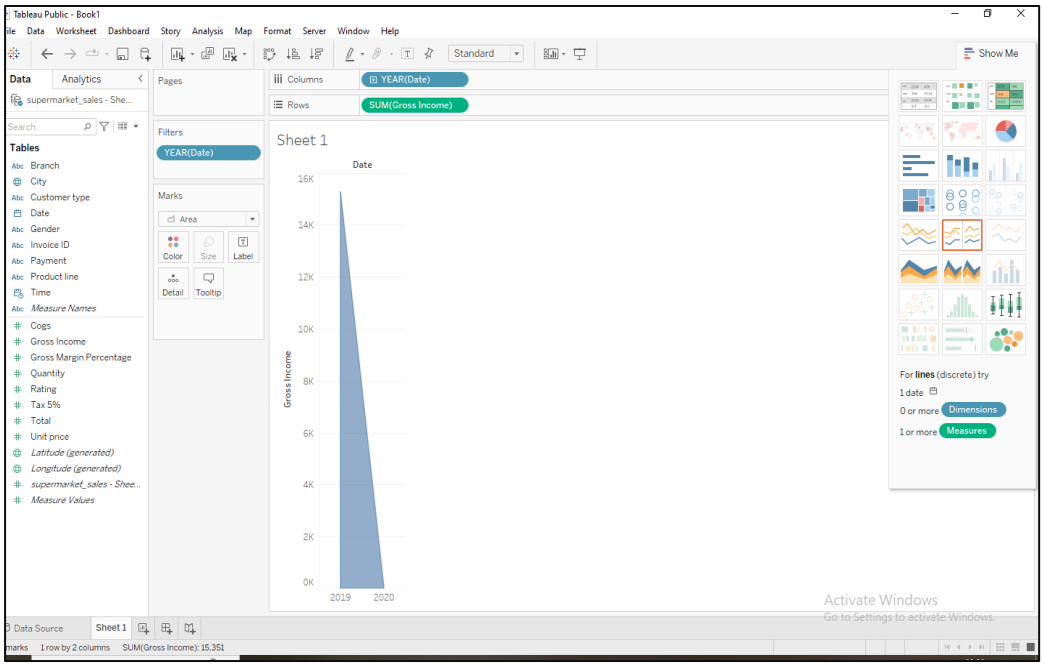
- Filter Dimensions are the filters applied on the dimension fields.



- Filter Measures are the filters applied on the measure fields.



- Filter Dates are the filters applied on the date fields.



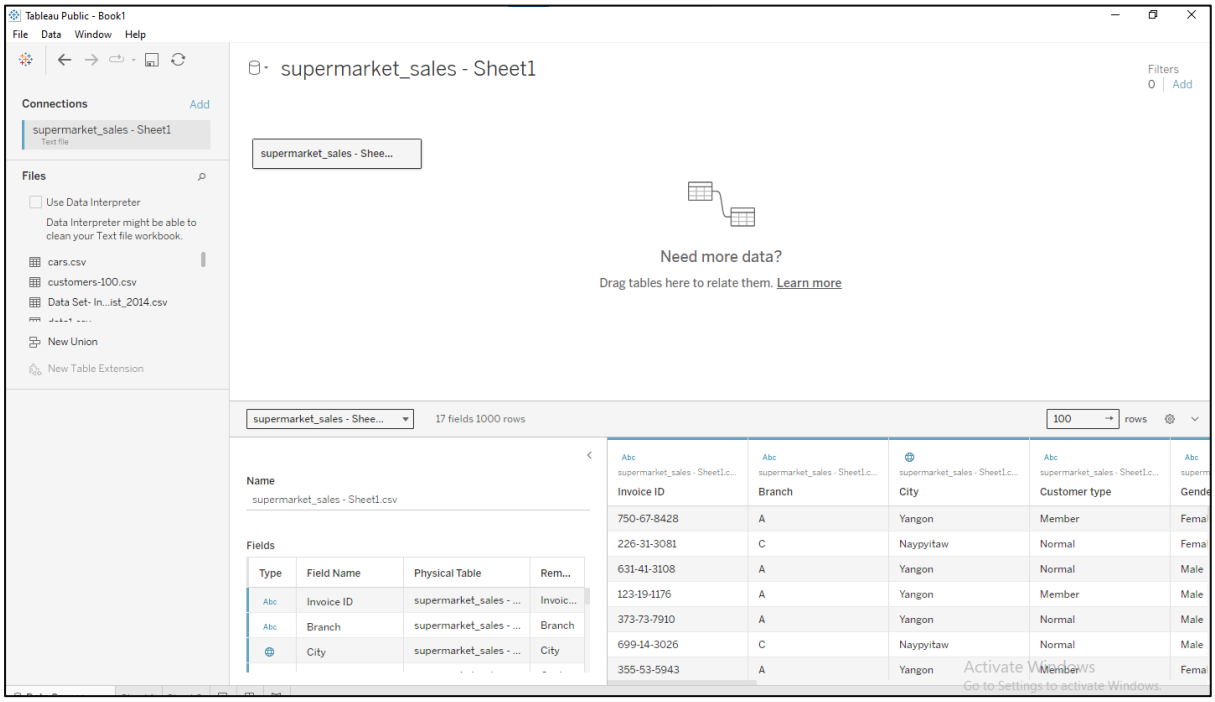
Sorting:

Tableau Sort Data:

Data present in the worksheet can be sorted based on the requirement. It can sort the data based on the data source such as ascending, descending order, or depend on any measured value.

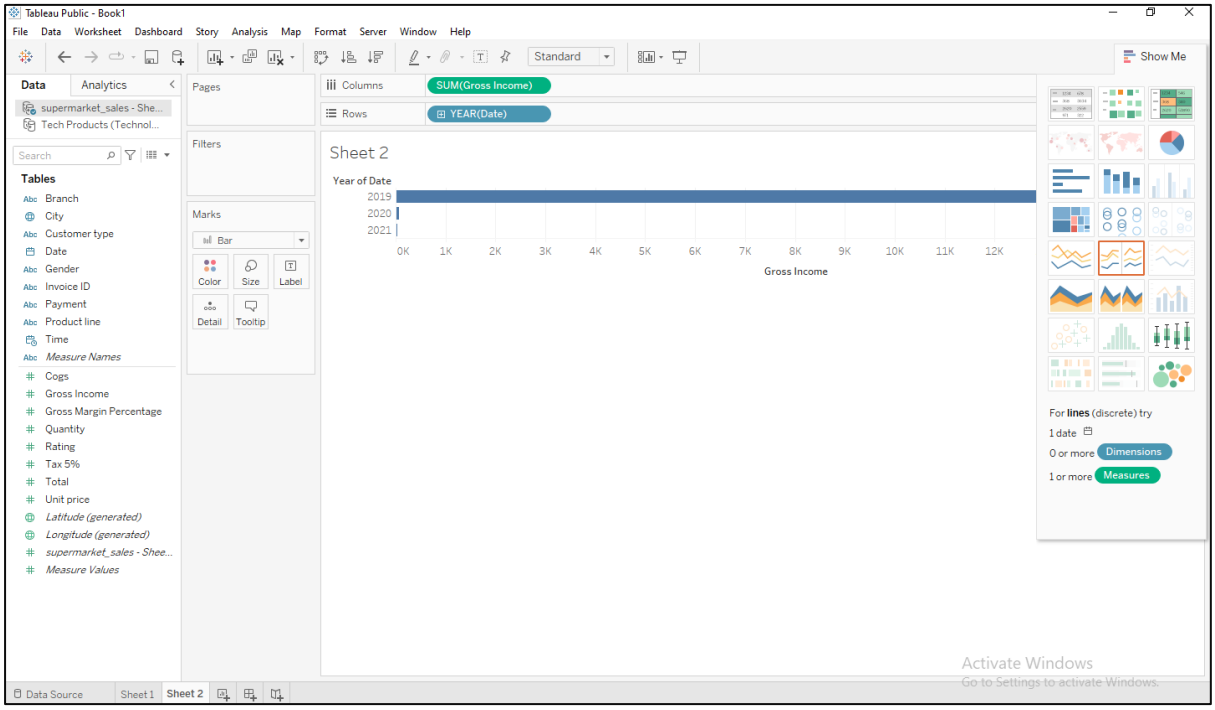
- Step1: Add the sample-superstore data source with Tableau and drag the Order table to the

pane shown in the below screenshot.



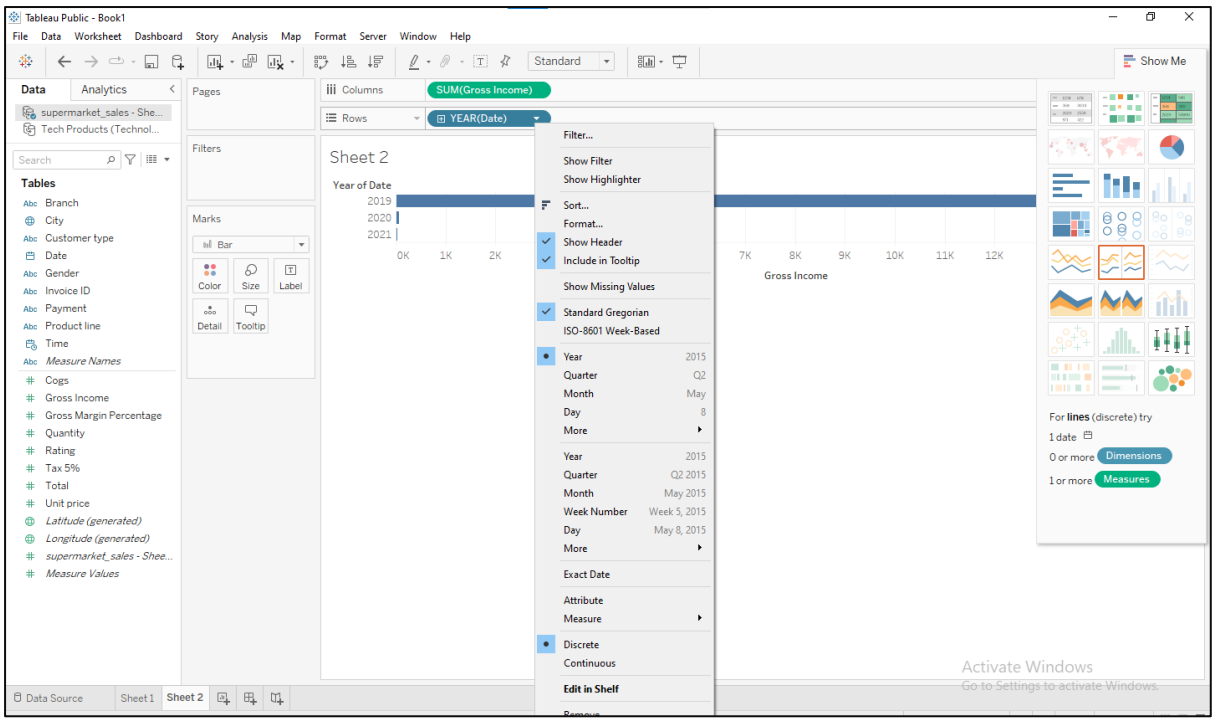
- Step2: Go to the worksheet and drag the date to the row shelf and the

Gross income to the column shelf.

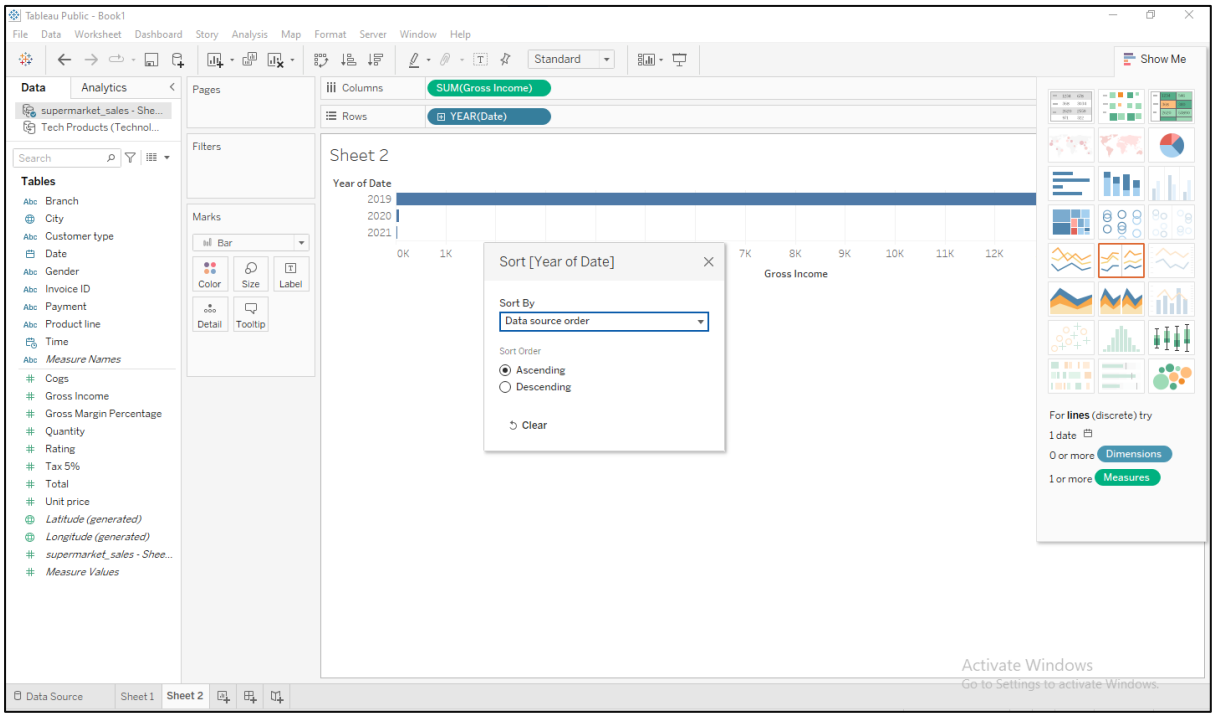


It creates a horizontal bar chart. Category field present in the visual order, and it is sorted based on data source by default. We can change the order of sorting by following the below procedure.

- Step3: Right-click on the date and select Sort option.



After that, it opens the Sort window. All options present inside the sort window is shown below as follows:



Sort Order:

- Ascending: It sorts the order of selected dimensions and measures in ascending order.
- Descending: It sorts the order of selected dimensions and measures in descending
- order.

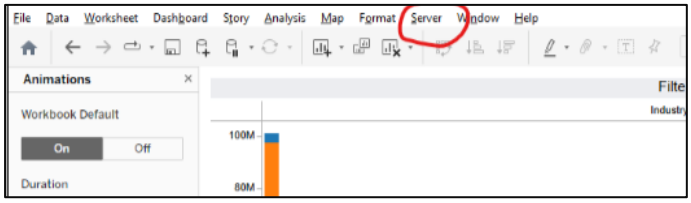
Sort By:

The field can be sorted in different types of methods that are explained below as follows.

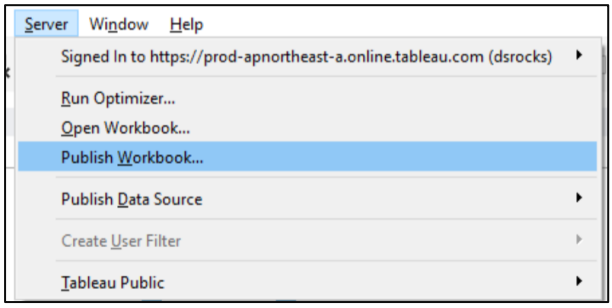
- Data source order: It sorts the field based on data source order.
- Alphabetic: It sorts the dimensions and measures in alphabetical order.
- Field: It sorts the field based on the other measure or dimension values.
- Manual: It can manually sort the data.

Publish Data on Tableau cloud

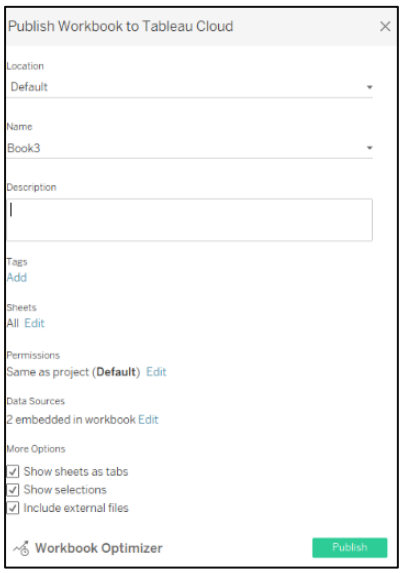
- Step 1: Create worksheets / dashboards in tableau.
- Step2: Click on “server” placed at top row selections



Step3: Click on publish workbook.



- Step4: Enter appropriate details in pop-up box.



Click on Publish button placed at bottom-right of pop-up box.

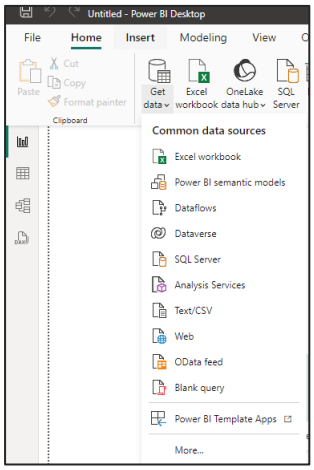
After clicking on “Publish” your workbook will be published on Tableau cloud.

PRACTICAL NO-08

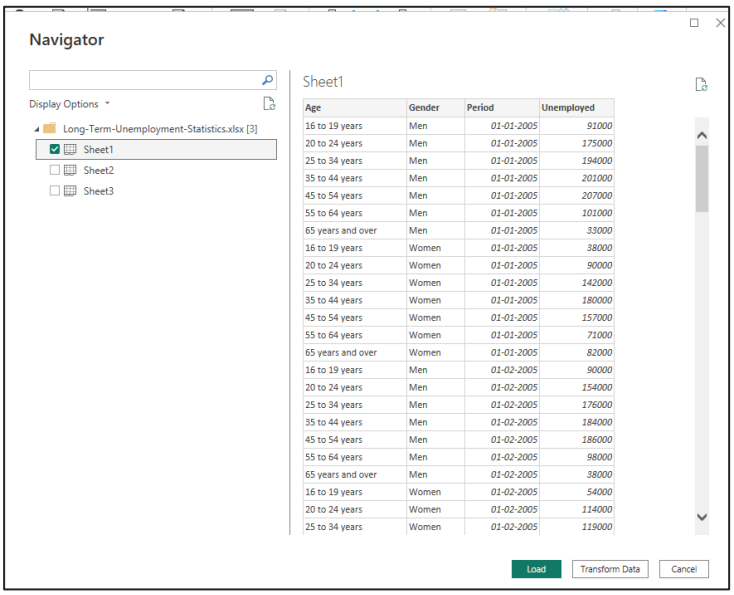
AIM: Perform data visualization using Power BI.

STEPS:

1. Click on the **"Home"** tab in the ribbon at the top.
2. Select **"Get Data"** from the options available,from the home tab to connect to your data source (e.g., Excel, SQL Server, CSV).



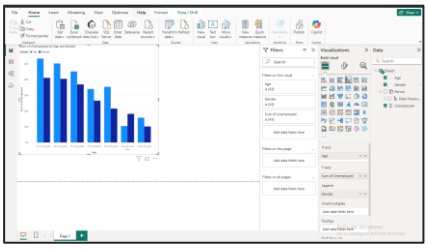
3. **Load the selected data source**



Age	Gender	Period	Unemployed
16 to 19 years	Men	01-01-2005	91000
20 to 24 years	Men	01-01-2005	175000
25 to 34 years	Men	01-01-2005	194000
35 to 44 years	Men	01-01-2005	201000
45 to 54 years	Men	01-01-2005	207000
55 to 64 years	Men	01-01-2005	101000
65 years and over	Men	01-01-2005	33000
16 to 19 years	Women	01-01-2005	38000
20 to 24 years	Women	01-01-2005	90000
25 to 34 years	Women	01-01-2005	142000
35 to 44 years	Women	01-01-2005	180000
45 to 54 years	Women	01-01-2005	157000
55 to 64 years	Women	01-01-2005	71000
65 years and over	Women	01-01-2005	82000
16 to 19 years	Men	01-02-2005	90000
20 to 24 years	Men	01-02-2005	154000
25 to 34 years	Men	01-02-2005	176000
35 to 44 years	Men	01-02-2005	184000
45 to 54 years	Men	01-02-2005	186000
55 to 64 years	Men	01-02-2005	98000
65 years and over	Men	01-02-2005	38000
16 to 19 years	Women	01-02-2005	54000
20 to 24 years	Women	01-02-2005	114000
25 to 34 years	Women	01-02-2005	119000

4. Create an visual

- In the **"Visual"** view, which is usually the default view when you open Power BI Desktop, you can start creating your report.
- Drag and drop fields from the **"Fields"** pane on the right to the **"Report Canvas"** in the center.
- Choose the type of visualization you want to create by selecting from the options in the **"Visualizations"** pane.

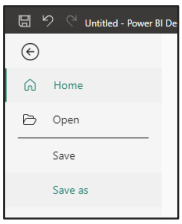


5. Customize your visuals

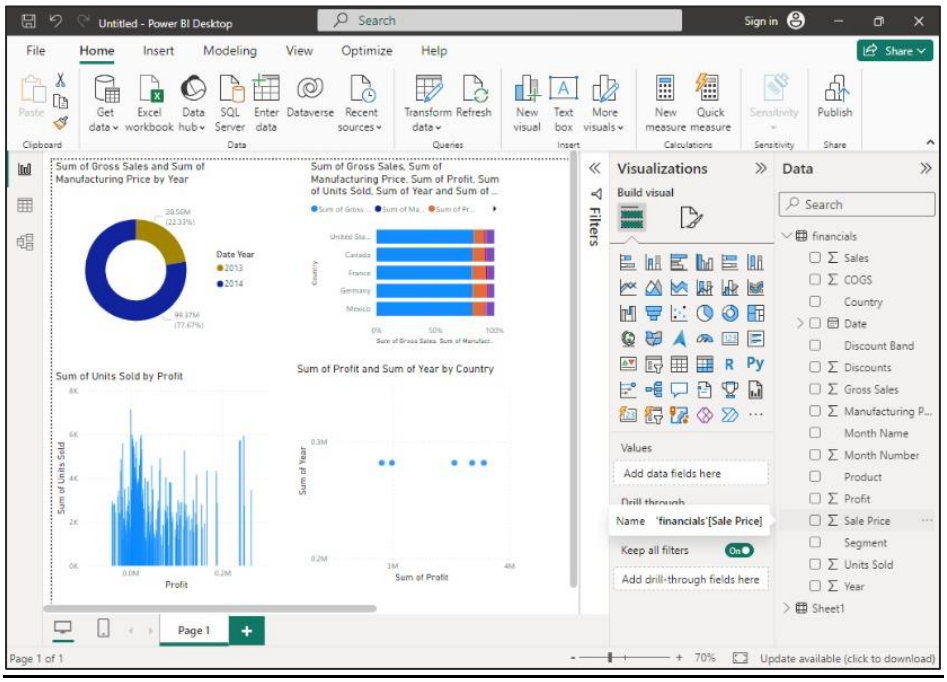
- Use the **"Format"** pane to adjust the appearance of your visualizations, such as changing colors, fonts, and labels.
- Add additional elements like text boxes, images, or shapes by using the options in the **"Insert"** tab.

6. Save the created visual

- Click **"File"** in the top left corner.
- Select **"Save"** to save your report



OUTPUT:

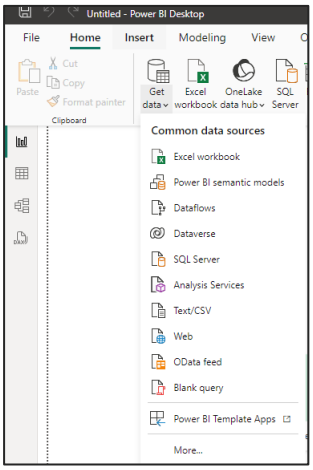


PRACTICAL NO-09

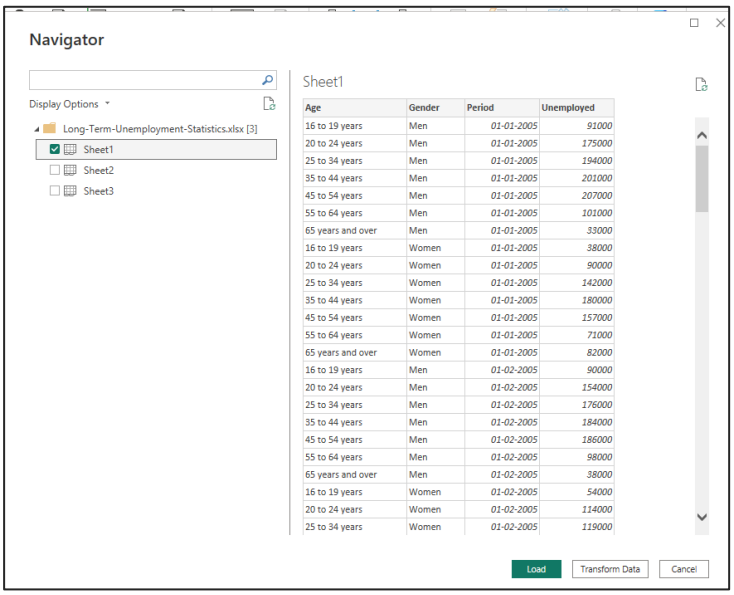
AIM: Create reports using Power BI.

STEPS:

1. Click on the **"Home"** tab in the ribbon at the top.
2. Select **"Get Data"** from the options available,from the home tab to connect to your data source (e.g., Excel, SQL Server, CSV).



3. **Load the selected data source**



4. Create an report

- In the **"Report"** view, which is usually the default view when you open Power BI Desktop, you can start creating your report.
- Drag and drop fields from the **"Fields"** pane on the right to the **"Report Canvas"** in the center.

5. Choose the type of visualization you want to create by selecting from the options in the **"Visualizations"** pane

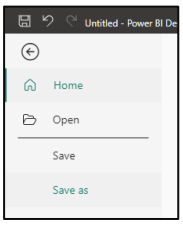
6. Customize your report

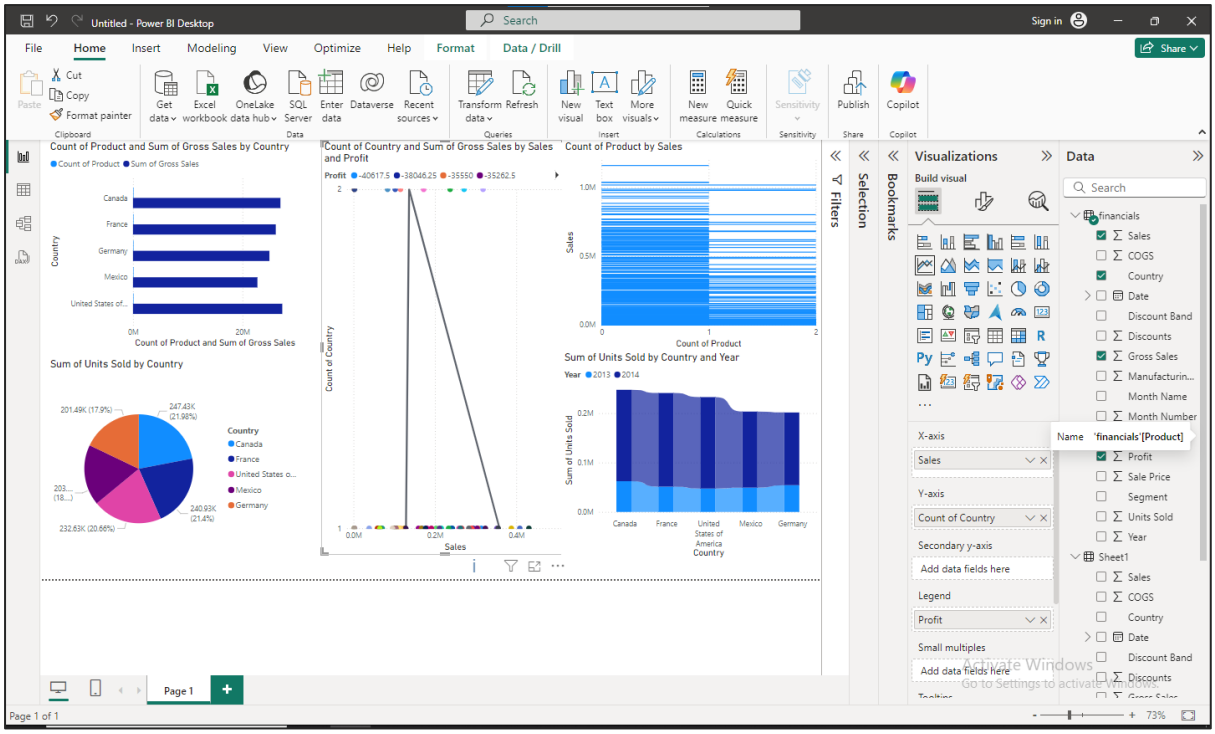
- Use the "**Format**" pane to adjust the appearance of your visualizations, such as changing colors, fonts, and labels.
- Add additional elements like text boxes, images, or shapes by using the options in the "**Insert**" tab.

7. Save the created report

- Click "**File**" in the top left corner.

Select "**Save**" to save your report



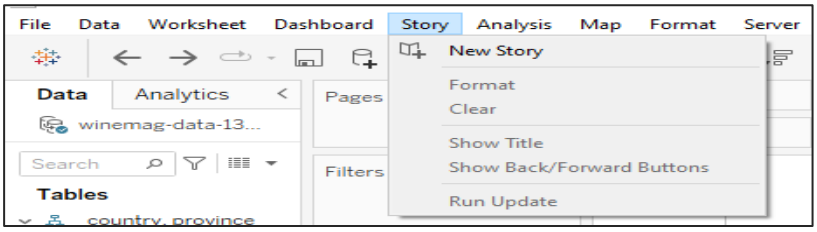


PRACTICAL NO-10

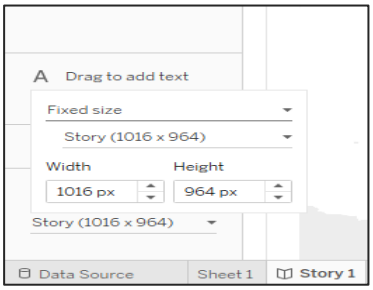
AIM: Create a data story in Tableau or power BI.

STEPS:

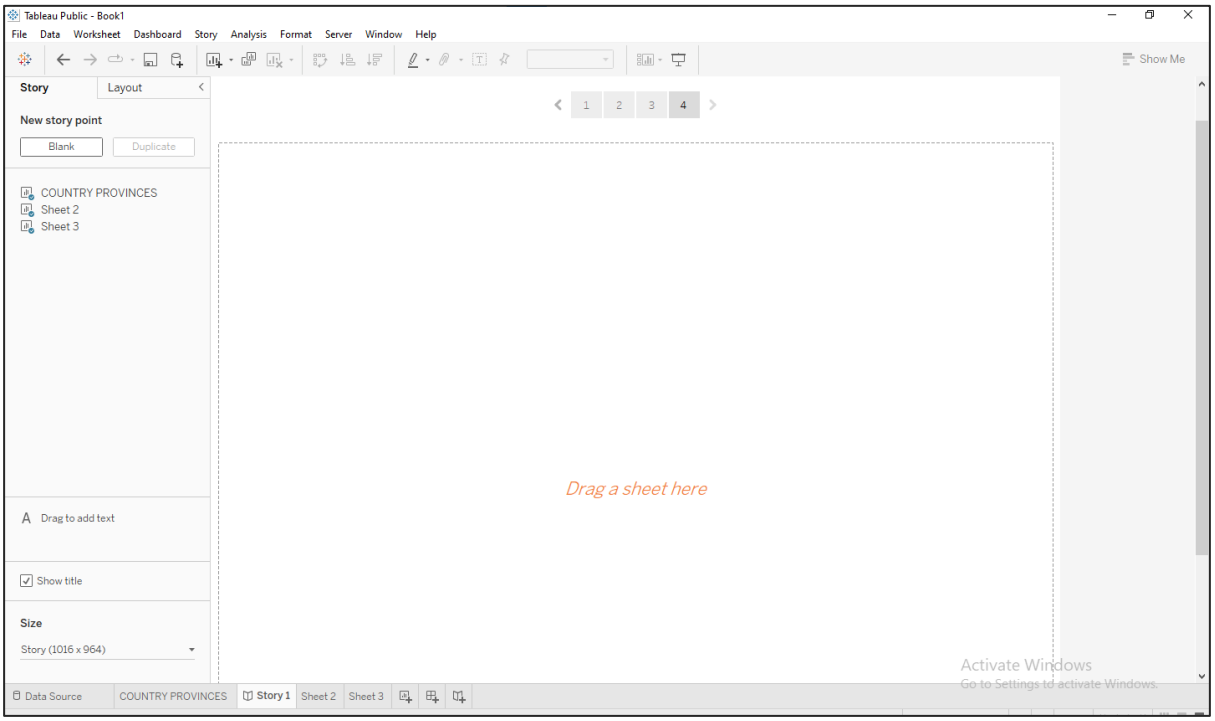
1. Click the New Story tab



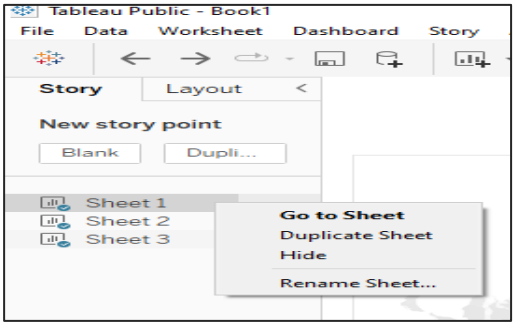
2. In the lower-left corner of the screen, choose a size for your story. Choose from one of the predefined sizes, or set a custom size, in pixels:



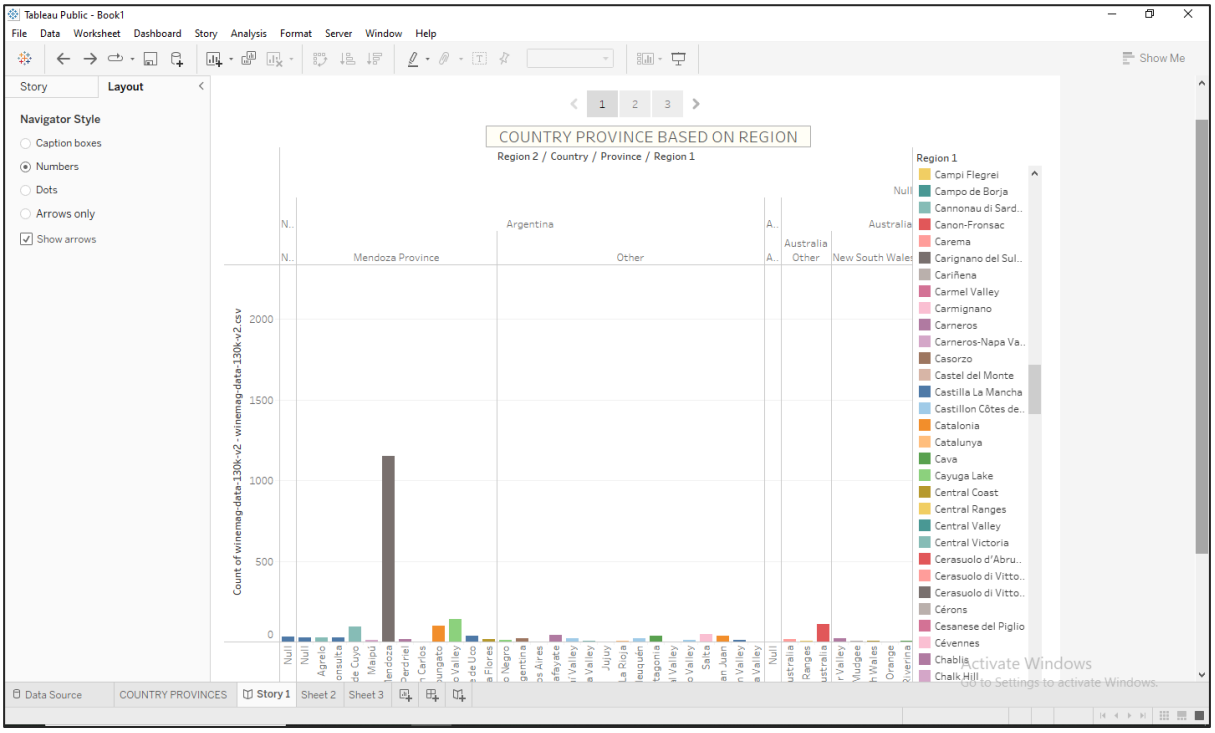
3. To start building your story, double-click a sheet on the left to add it to a story point.In Tableau Desktop, you can also drag sheets into your story point.



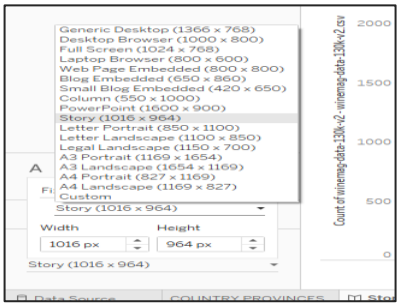
4. By default, your story gets its title from the sheet name. To edit it, right-click the sheet tab,and choose Rename Sheet. If you're using Tableau Desktop, you can also rename a story by double-clicking the title.



5. Click the Layout tab. Choose a navigator style that best suits your story, and show or hide the next and previous arrows.



6. Click the Size drop-down menu and select the story you want the dashboard to fit inside.



7. Present your story

OUTPUT:

Creating story in tableau

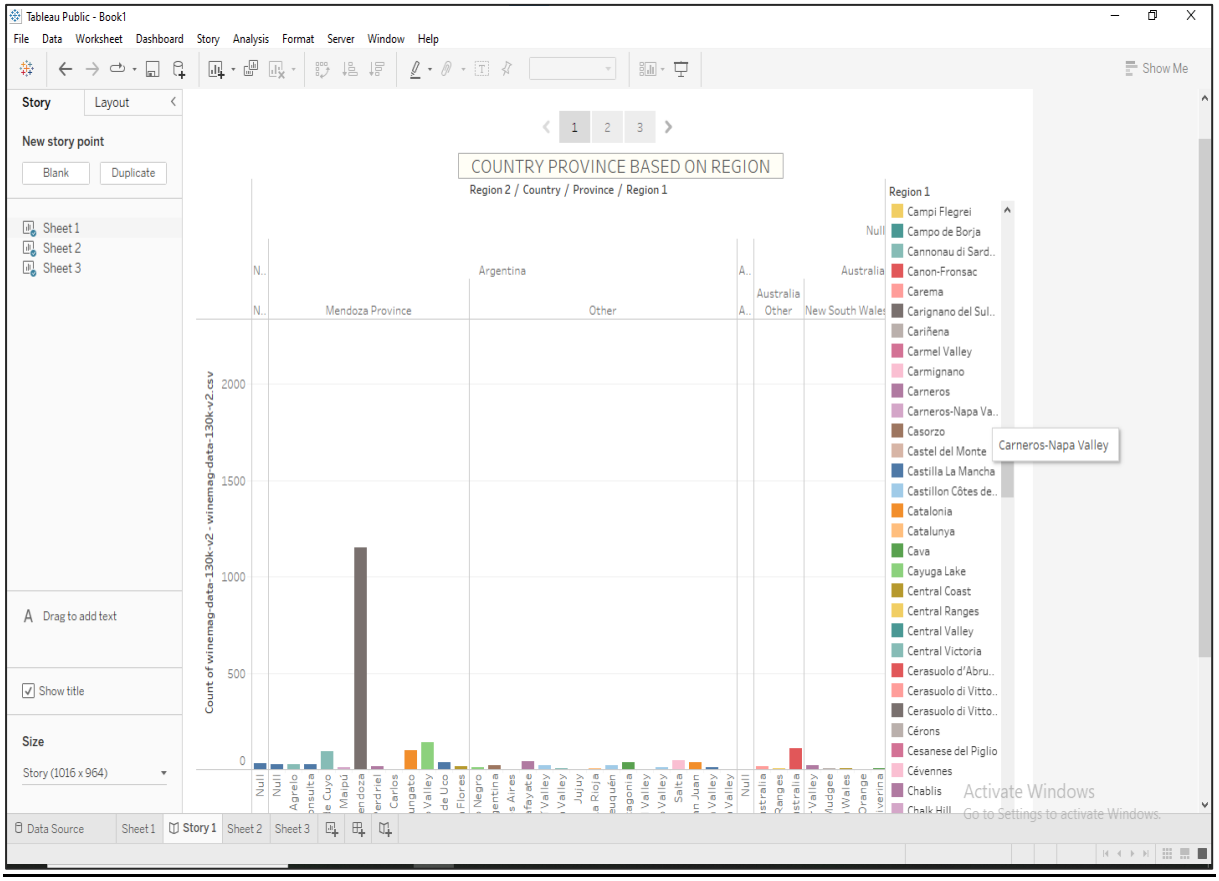


Tableau Public - Book1
File Data Worksheet Dashboard Story Analysis Format Server Window Help
Show Me
Story Layout
New story point
Blank Duplicate
Sheet 1
Sheet 2
Sheet 3
Drag to add text
Show title
Size
Story (1016 x 964)
COUNTRY PROVINCE BASED ON REGION
Region 2 / Country / Province / Region 1
Argentina
Australia
Mendoza Province
Other
Australia Other
New South Wales
Count of winemag-data-130k-v2-winemag-data-130k-v2.csv
2000
1500
1000
500
0
Region 1
Campi Flegrei
Campo de Borja
Cannonau di Sard..
Canon-Fronsac
Carema
Carignano del Sul..
Cariñena
Carmel Valley
Carmignano
Carneros
Carneros-Napa Va..
Casorzo
Castel del Monte
Castilla La Mancha
Castillon Côtes de..
Catalonia
Catalunya
Cava
Cayuga Lake
Central Coast
Central Ranges
Central Valley
Central Victoria
Cerasuolo d'Abru..
Cerasuolo di Vitto..
Cerasuolo di Vitto..
Cérons
Cesanese del Piglio
Cévennes
Chablis
Carneros-Napa Valley
Activate Windows
Go to Settings to activate Windows.
Data Source
Sheet 1
Story 1
Sheet 2
Sheet 3

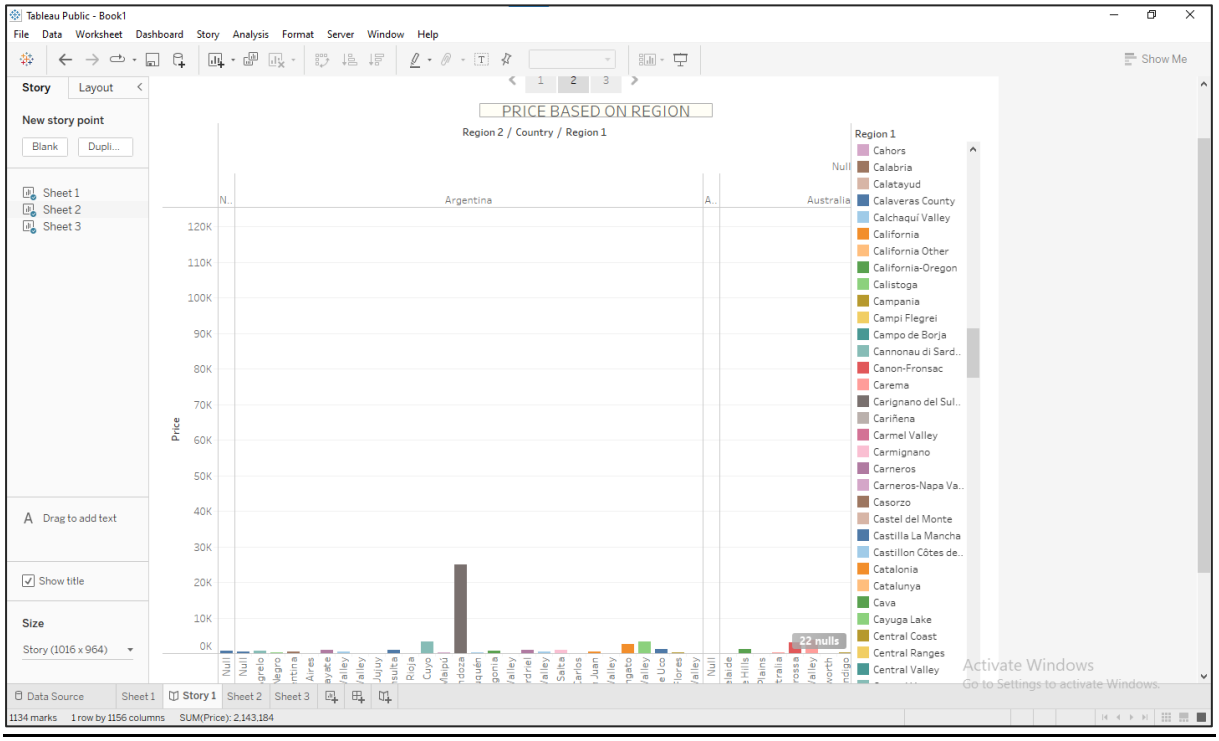


Tableau Public - Book1
File Data Worksheet Dashboard Story Analysis Format Server Window Help
Show Me
Story Layout
New story point
Sheet 1
Sheet 2
Sheet 3
Drag to add text
Show title
Size
Story (1016 x 964)
PRICE BASED ON REGION
Region 2 / Country / Region 1
Argentina
Australia
Price
120K
110K
100K
90K
80K
70K
60K
50K
40K
30K
20K
10K
0K
22 nulls
Region 1
Cahors
Calabria
Calatayud
Calaveras County
Calchaquí Valley
California
California Other
California-Oregon
Calistoga
Campania
Campi Flegrei
Campo de Borja
Cannonau di Sard..
Canon-Fronsac
Carema
Carignano del Sul..
Cariñena
Carmel Valley
Carmignano
Carneros
Carneros-Napa Va..
Casorzo
Castel del Monte
Castilla La Mancha
Castillon Côtes de..
Catalonia
Catalunya
Cava
Cayuga Lake
Central Coast
Central Ranges
Central Valley
Activate Windows
Go to Settings to activate Windows.
Data Source
Sheet 1
Story 1
Sheet 2
Sheet 3
1134 marks 1 row by 1156 columns SUM(Price): 2,143,184

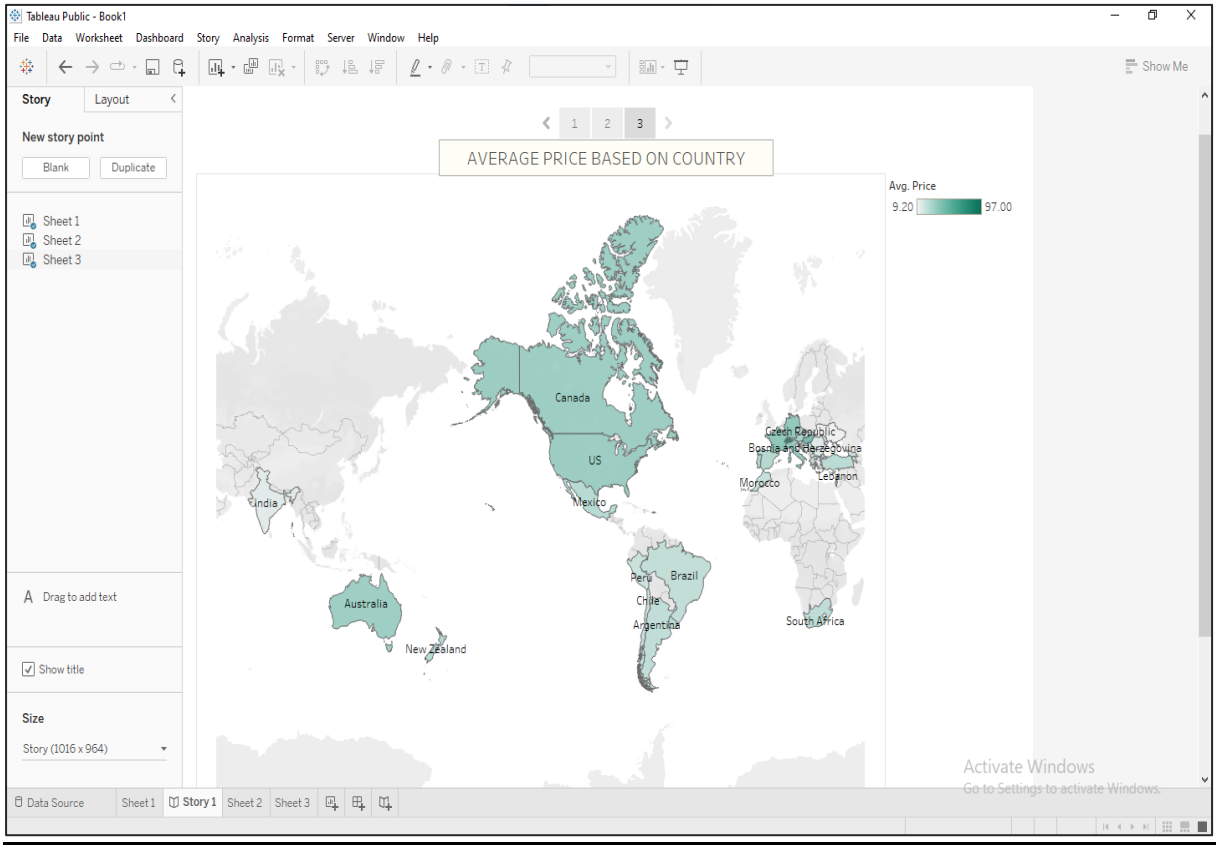


Tableau Public - Book1
File Data Worksheet Dashboard Story Analysis Format Server Window Help
Show Me
Story
Layout
New story point
Blank
Duplicate
Sheet 1
Sheet 2
Sheet 3
Drag to add text
Show title
Size
Story (1016 x 964)
1
2
3
AVERAGE PRICE BASED ON COUNTRY
Avg. Price
9.20
97.00
Canada
US
Mexico
India
Australia
New Zealand
Peru
Brazil
Chile
Argentina
Czech Republic
Bosnia and Herzegovina
Morocco
Lebanon
South Africa
Activate Windows
Go to Settings to activate Windows.
Data Source
Sheet 1
Story 1
Sheet 2
Sheet 3