

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM - 602 105



**AD23632 – FRAMEWORK FOR DATA AND
VISUAL ANALYTICS**

LABORATORY LAB MANUAL

NAME: HARISH KUMAR V
.....

REGISTER NUMBER: 2116-231501057
.....

YEAR / BRANCH / SECTION: III YEAR / AIML / A
.....

SEMESTER: V SEMESTER
.....

ACADEMIC YEAR: 2025-2026
.....



BONAFIDE CERTIFICATE

CERTIFIED THAT THIS LABORATORY RECORD REPORT FOR “**FRAMEWORK FOR DATA AND VISUAL ANALYTICS**” IS THE BONAFIDE WORK OF “**HARISH KUMAR V [231501057]**” WHO CARRIED OUT THE PRACTICAL WORK UNDER MY SUPERVISION.

Submitted for the Practical Examination held on _____

SIGNATURE

**Dr. A. Rajasekar, AIML,
REC (Autonomous) Thandalam,
Chennai - 602 105**

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

REG NO: 231501057

NAME: HARISHKUMAR V

YEAR: III YEAR

BRANCH: AIML SEC: A

S.NO	DATE	EXPERIMENT TITLE	PAGE NO
1	16/07/2025	SETTING UP PYTHON ENVIRONMENT	2
2	23/07/2025	EDA-DATA IMPORT & EXPORT	6
3	06/08/2025	EDA- DATA CLEANING	11
4	13/08/2025	EDA- DATA INSPECTION & ANALYTICS	15
5	03/09/2025	EDA- DATA VISUALIZATION (MATPLOTLIB)	23
6	24/09/2025	DATA VISUALIZATION USING POWER BI	39
7	08/10/2025	DATA VISUALIZATION USING TABLEAU	48

EXPT.NO: 1	SETTING UP THE PYTHON ENVIRONMENT AND JUPYTER NOTEBOOK
DATE: 16/072025	

AIM:

To set up a Python environment using Jupyter Notebook and demonstrate code execution, Markdown formatting, and the use of Jupyter Widgets and Jupyter AI.

PROBLEM STATEMENT:

Create a Jupyter Notebook that showcases Python code execution, Markdown documentation, interactive widgets, and AI-assisted features.

ALGORITHM:

1. Install **Jupyter** Notebook using `pip install notebook`.
2. Launch Jupyter using `jupyter notebook`.
3. Create a new **Python 3** notebook.
4. Add and execute Python code cells.
5. Add Markdown cells for headings, lists, and descriptions.
6. Install and use **ipywidgets** for interactivity.
7. Explore **Jupyter AI**

IPYTHON WIDGETS

It is a Python library that lets you create interactive user interface controls in Jupyter Notebooks, JupyterLab, and JupyterLite.

These controls include:

- Sliders
- Dropdowns
- Buttons
- Text boxes

- Date pickers
- File uploads
- Tabs & Layout Containers

CODE:

```
jupyter --version
pip install ipywidgets
pip install jupyterlab-widgets # Step 1: Basic Python code print("Hello,
    Jupyter!")
# Step 2: Markdown cell (add this in a Markdown cell, not code) # ## Welcome
    to Jupyter Notebook
# This is a Markdown cell. You can write bold, italic, or code`. # Step
    3: Jupyter Widgets
import ipywidgets as widgets widgets.IntSlider(description='Slider:', min=0,
    max=100, step=5) Output:
```

```
# Jupyter Widgets
import ipywidgets as widgets
from IPython.display import display # Create an IntSlider widget for age age =
    widgets.IntSlider(
description="Age:", min=0,
max=100,

value=25
)
# Display the slider display(age) Output:
```

```
Code:
import ipywidgets as widgets
from IPython.display import display, clear_output # Personal Info Widgets
name = widgets.Text( description="Name:", placeholder="Enter your name"
)
age = widgets.IntSlider( description="Age:", min=0, max=100, value=25
)
gender = widgets.ToggleButtons( options=['Male', 'Female', 'Other'],
    description='Gender:'
)
birthdate = widgets.DatePicker( description='DOB:'
)
height = widgets.FloatSlider( description="Height (m):",
min=1.0, max=2.5, step=0.01, value=1.70
)
)
```

```

bio = widgets.Textarea( description="Bio:",
placeholder="Write something about yourself"
)
# Output display
profile_output = widgets.Output() # Submit button
submit_btn = widgets.Button( description="Create Profile",
    button_style='success', icon='check'
)
# Event handler def on_submit(b):
with profile_output: clear_output()
print(" Profile Summary \n") print(f"Name: {name.value}") print(f"Age:
    {age.value}") print(f"Height: {height.value} m") print(f"Gender:
    {gender.value}") print(f"Date of Birth: {birthdate.value}") print(f"Bio:
    {bio.value}")
submit_btn.on_click(on_submit) # Layout (No Tabs)
form = widgets.VBox([ name,
age, height,

gender, birthdate, bio, submit_btn,
profile_output
])

# Display the form display(form)

```

OUTPUT:

```

: # Python code cell
print("Hello, Jupyter!")

# Markdown cell
# ## This is a Markdown Heading

# Jupyter Widgets
import ipywidgets as widgets
widgets.IntSlider()

```

Hello, Jupyter!

:

Age:


Name:


Age: 5

Height (m): 1.70

Gender:

☒ Male ☐ Female ☐ Other

DOB: 

Bio: 

RESULT :

Thus, the program successfully created a **Jupyter Notebook** showcasing Python code execution, Markdown formatting, and the use of **interactive widgets**.

EXPT.NO: 2	EDA – DATA IMPORT AND EXPORT
DATE: 23/07/2025	

AIM:

To import data from CSV, Excel, and SQL databases and export DataFrames.

PROBLEM STATEMENT:

- Load datasets in multiple formats and export a DataFrame to Excel.

ALGORITHM:

STEP 1: IMPORT REQUIRED LIBRARIES

- Import pandas for data manipulation.
- Import sqlite3 for database handling.
- Import requests and BeautifulSoup for web scraping.

STEP 2: IMPORT DATA FROM CSV FILE

- Use `pd.read_csv(filename)` to load data from a CSV file into a DataFrame.
- Display the first few rows using `.head()`.

STEP 3: IMPORT DATA FROM EXCEL FILE

- Use `pd.read_excel(filename)` to load data from an Excel file.
- Display the first few rows using `.head()`.

STEP 4: IMPORT DATA FROM SQL DATABASE

- Connect to or create an SQLite database using `sqlite3.connect()`.
- Create a table (if not already exists).
- Insert sample records (if needed).

- Use `pd.read_sql_query(query, connection)` to load table data into a DataFrame.

STEP 5: IMPORT DATA FROM THE WEB (WEB SCRAPING)

- Use `requests.get(url)` to fetch HTML content.
- Parse HTML with BeautifulSoup.
- Locate the desired table using `soup.find()` or `soup.find_all()`.
- Convert the HTML table to a DataFrame using `pd.read_html()`.

STEP 6: HANDLE DIFFERENT DATA FORMATS

- Check for data type issues or format mismatches.
- Convert date columns using `pd.to_datetime()`.
- Convert categorical or boolean fields using `.astype()`.

STEP 7: EXPORT DATA TO EXCEL FILE

- Use `DataFrame.to_excel(filename, index=False)` to save a DataFrame to an Excel file.
- Confirm export success with a print statement.

SAMPLE CODE

```
# Import necessary libraries import pandas as pd
import sqlite3 import requests
from bs4 import BeautifulSoup

# 1. Importing data from CSV csv_df = pd.read_csv('Iris.csv') print("CSV Data:")
print(csv_df.head())

# 2. Importing data from Excel
excel_df = pd.read_excel('heart stalog dataset.xlsx') print("\nExcel Data:")
excel_df.head(5)

#import from SQL Database import sqlite3
# Connect to (or create) the database conn = sqlite3.connect('my_database.db')
cursor = conn.cursor()

# Create the 'employees' table cursor.execute('''
CREATE TABLE IF NOT EXISTS employees ( id INTEGER PRIMARY KEY,
name TEXT, department TEXT, salary REAL, hire_date TEXT
```

```

) '')

# Insert example records cursor.executemany('''
INSERT INTO employees (id, name, department, salary, hire_date) VALUES (?, ?, ?,
?, ?) ''', [
(1, 'Alice Smith', 'HR', 55000, '2018-05-01'),

(2, 'Bob Johnson', 'IT', 72000, '2019-07-15'),
(3, 'Carol White', 'Finance', 68000, '2017-09-30'),
(4, 'David Brown', 'Marketing', 60000, '2020-02-10'),
(5, 'Eva Green', 'IT', 75000, '2021-04-25'),
])

```

```

# Commit and close conn.commit()
print("Database and 'employees' table created with sample data.")

```

Database and 'employees' table created with sample data

```

sql_df = pd.read_sql_query("SELECT * FROM employees", conn) print(sql_df)

```

```

import pandas as pd import requests
from bs4 import BeautifulSoup

```

```

# URL of the Wikipedia page
url =
"https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population"

```

```

# Fetch the page
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")

```

```

# Find the first table with class 'wikitable' (Wikipedia uses this) html_table =
soup.find("table", {"class": "wikitable"})

```

```

# Use pandas to read the HTML table into a DataFrame web_df =
pd.read_html(str(html_table))[0]

```

```

# Show the first few rows print("\nWeb Scraped Data:") print(web_df.head())

```

```

# 5. Handling different data formats

```

```
# For example, converting a date column to datetime if 'date' in csv_df.columns:
csv_df['date'] = pd.to_datetime(csv_df['date']).dt.strftime('%Y-%m-%d')
# 6. Export a DataFrame to Excel
# Here we export the CSV data as an example csv_df.to_excel('exported_data.xlsx',
index=False) print("\nData exported to 'exported_data.xlsx' successfully.")
```

OUTPUT:

CSV Data:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Excel Data:

	age	sex	chest	resting_blood_pressure	serum_cholesterol	fasting_blood_sugar	resting_electrocardiogram
0	70	1	4	130	322	0	
1	67	0	3	115	564	0	
2	57	1	2	124	261	0	
3	64	1	4	128	263	0	
4	74	0	2	120	269	0	

	id	name	department	salary	hire_date
0	1	Alice Smith	HR	55000.0	2018-05-01
1	2	Bob Johnson	IT	72000.0	2019-07-15
2	3	Carol White	Finance	68000.0	2017-09-30
3	4	David Brown	Marketing	60000.0	2020-02-10
4	5	Eva Green	IT	75000.0	2021-04-25

Data exported to 'exported_data.xlsx' successfully.

Web Scraped Data:

	Location	Population	% of world	Date	\
0	World	8232000000	100%	13 Jun 2025	
1	India	1413324000	17.3%	1 Mar 2025	
2	China	1408280000	17.2%	31 Dec 2024	
3	United States	340110988	4.2%	1 Jul 2024	
4	Indonesia	282477584	3.5%	30 Jun 2024	

	Source (official or from the United Nations)	Notes
0	UN projection[1][3]	NaN
1	Official projection[4]	[b]
2	Official estimate[5]	[c]
3	Official estimate[6]	[d]
4	National annual projection[7]	NaN

RESULT :

Thus, the program successfully created a Jupyter Notebook showcasing Python code to import data from **CSV**, **Excel**, and **SQL** databases, as well as export **DataFrames**.

EXPT.NO: 3	EDA-DATA CLEANING
DATE: 06/08/2025	

AIM

To clean data by handling missing values, duplicates, data types, and normalization.

PROBLEM STATEMENT

Clean a dataset by removing nulls, duplicates, and normalizing numeric fields.

ALGORITHM

- Load dataset.
- Detect missing values (isnull).
- Fill or drop missing values.
- Remove duplicates.
- Convert data types.
- Normalize numeric columns.

SAMPLE CODE

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler import
matplotlib.pyplot as plt

# Step 1: Load dataset
df = pd.read_csv('StudentsPerformance.csv') df.head()

df.shape (1005, 8)
# Step 2: Handle Missing Values # Detect
missing_info = df.isnull().sum() print("Missing values:\n", missing_info)

# Fill or Drop (based on context) df.fillna({
```

```

'parental level of education': df['parental level of education'].mode()[0],
'lunch': df['lunch'].mode()[0]
}, inplace=True)
missing_info = df.isnull().sum() missing_info

duplicates = df[df.duplicated()] duplicates

duplicates.shape (5, 8)

# Drop duplicates df.drop_duplicates(inplace=True) df.shape

# Step 4: Convert Data Types (if needed)

# For consistency, make sure string columns are lowercase

categorical_cols = ['gender', 'race/ethnicity', 'parental level of
education', 'lunch', 'test preparation course']
for col in categorical_cols:
df[col] = df[col].astype(str).str.lower().str.strip() categorical_cols
['gender', 'race/ethnicity',
'parental level of education', 'lunch',
'test preparation course']

numeric_cols = ['math score', 'reading score', 'writing score'] numeric_cols

['math score', 'reading score', 'writing score']

plt.figure(figsize=(15, 4))
for i, col in enumerate(numeric_cols): plt.subplot(1, 3, i+1)
sns.histplot(df[col], kde=True, bins=20) plt.title(f'Before Normalization:
{col}')
plt.tight_layout() plt.show()

minmax_scaler = MinMaxScaler() df_minmax = df.copy()
df_minmax[numeric_cols] = minmax_scaler.fit_transform(df[numeric_cols])
plt.figure(figsize=(15, 4))
for i, col in enumerate(numeric_cols): plt.subplot(1, 3, i+1)
sns.histplot(df_minmax[col], kde=True, bins=20, color='green')
plt.title(f'Min-Max Normalized: {col}')
plt.tight_layout() plt.show()

# Standard Scaling (Z-score) zscore_scaler = StandardScaler() df_zscore =
df.copy()
df_zscore[numeric_cols] = zscore_scaler.fit_transform(df[numeric_cols])

```

```
plt.figure(figsize=(15, 4))
for i, col in enumerate(numeric_cols): plt.subplot(1, 3, i+1)
sns.histplot(df_zscore[col], kde=True, bins=20, color='orange')
plt.title(f'Z-score Normalized: {col}')
plt.tight_layout() plt.show()
```

OUTPUT:

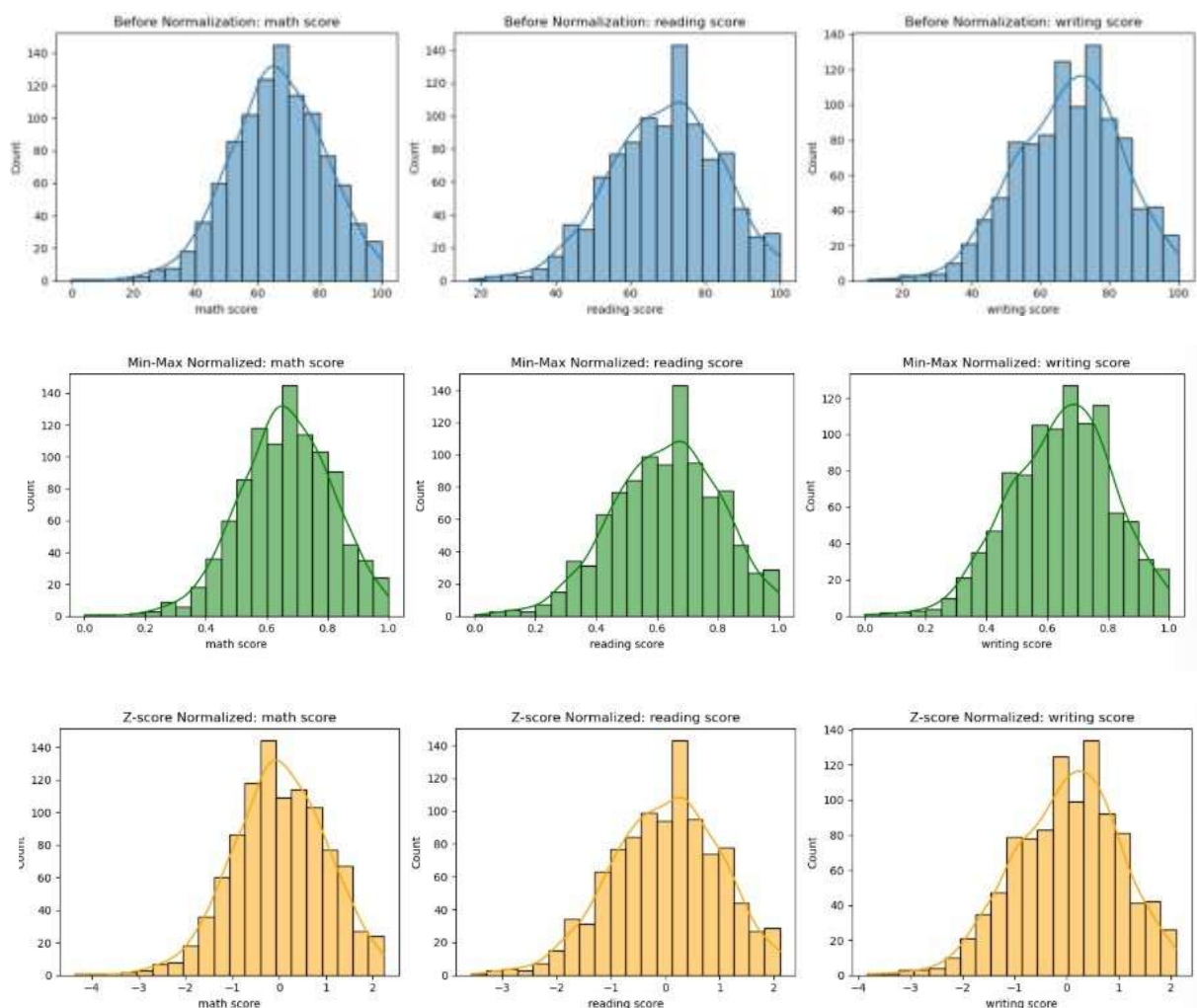
	gender	race/ethnicity	level of education	lunch	preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

Missing values:

```
gender          0
race/ethnicity  0
parental level of education  7
lunch           0
test preparation course  0
math score      0
reading score   0
writing score   0
dtype: int64
```

```
gender          0
race/ethnicity  0
parental level of education  0
lunch           0
test preparation course  0
math score      0
reading score   0
writing score   0
dtype: int64
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
1000	male	group D	some college	standard	none	76	64	66
1001	male	group C	associate's degree	standard	none	46	43	42
1002	female	group B	bachelor's degree	standard	none	67	86	83
1003	male	group E	some high school	standard	none	92	87	78
1004	male	group C	bachelor's degree	standard	completed	83	82	84



RESULT :

Thus, the program successfully created a **Jupyter Notebook** showcasing Python code handling missing values, removing duplicates and unnecessary data, Data type conversion & Normalizing data.

EXPT.NO: 4	EDA-DATA INSPECTION AND ANALYSIS
DATE: 13/08/2025	

AIM

To understand how to view, inspect, and summarize data stored in a DataFrame for initial exploration and analysis.

PROBLEM STATEMENT

Large datasets are hard to understand at first. To make them meaningful, we first view and inspect the data to know its structure, then filter and select only the required rows or columns, and finally calculate basic statistics like mean, median, and standard deviation to summarize the data.

ALGORITHM

- Step 1: Import pandas and load/create the **DataFrame**.
- Step 2: View data using **head(), tail(), shape, dtypes, and info()**.
- Step 3: Filter rows and select columns using conditions and logical operators.
- Step 4: Calculate **mean, median, mode, range, variance, and standard deviation**.
- Step 5: Interpret the results to find patterns and spread of data.

CODE:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler import
matplotlib.pyplot as plt

# Step 1: Load dataset
df = pd.read_csv('StudentsPerformance.csv') df.head()
```

```

df.head(3)

df.tail()

df.shape (1005, 8)
df.columns.tolist() ['gender', 'race/ethnicity',
'parental level of education', 'lunch',
'test preparation course', 'math score',
'reading score',

'writing score'] df.dtypes

df.info()

## Step 3: Filtering and Subsetting Data

print("\n---- Filtering and Subsetting  ")
# Students with math score > 70
print("\nStudents with math score > 70:\n", df[df["math score"] > 70])

# Female students only
print("\nFemale students:\n", df[df["gender"] == "female"])

# Select only 'gender' and 'math score' columns
print("\nSubset with gender and math score:\n", df[["gender", "math score"]])

print("\n---- Descriptive Statistics  ")
math_scores = df["math score"]

mean = math_scores.mean() median = math_scores.median()
mode = math_scores.mode()[0] # mode() returns a Series

_range = math_scores.max() - math_scores.min() variance = math_scores.var()
std_dev = math_scores.std()

print(f"\nMean (Math Score): {mean}") print(f"Median (Math Score): {median}")
print(f"Mode (Math Score): {mode}") print(f"Range (Math Score): {_range}")
print(f"Variance (Math Score): {variance}")
print(f"Standard Deviation (Math Score): {std_dev}")
---- Descriptive Statistics ----

```

```

Mean (Math Score): 66.12238805970149 Median (Math Score): 66.0
Mode (Math Score): 65 Range (Math Score): 100
Variance (Math Score): 230.2270381161917
Standard Deviation (Math Score): 15.173234266832885 print("\n---- Visualization ")
# 1. Bar chart: Average scores per subject avg_scores = {
"Math": df["math score"].mean(),
"Reading": df["reading score"].mean(), "Writing": df["writing score"].mean()

}

plt.figure(figsize=(6, 4)) plt.bar(avg_scores.keys(), avg_scores.values())
plt.title("Average Scores per Subject") plt.ylabel("Average Score")
plt.xlabel("Subjects")
plt.show()

# 2. Histogram: Distribution of math scores plt.figure(figsize=(6, 4))
plt.hist(df["math score"], bins=5, edgecolor="black") plt.title("Distribution of
Math Scores") plt.xlabel("Math Score")
plt.ylabel("Frequency") plt.show()

# 3. Boxplot: Spread of math scores plt.figure(figsize=(4, 4))
plt.boxplot(df["math score"]) plt.title("Boxplot of Math Scores") plt.ylabel("Math
Score")
plt.show()

import matplotlib.pyplot as plt
# Plot Histogram with Mean, Median, and Mode Lines plt.figure(figsize=(7, 4))
plt.hist(df["math score"], bins=5, edgecolor="black", alpha=0.6)
plt.axvline(mean, color='red', linestyle='--', linewidth=2, label=f"Mean:
{mean:.2f}") plt.axvline(median, color='green', linestyle='-.', linewidth=2,
label=f"Median: {median:.2f}") plt.axvline(mode, color='blue', linestyle=':',
linewidth=2, label=f"Mode: {mode}") plt.title("Math Score Distribution with Mean,
Median, and Mode")
plt.xlabel("Math Score") plt.ylabel("Frequency") plt.legend()
plt.show()

```

OUTPUT:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
1000	male	group D	some college	standard	none	76	64	66
1001	male	group C	associate's degree	standard	none	46	43	42
1002	female	group B	bachelor's degree	standard	none	67	86	83
1003	male	group E	some high school	standard	none	92	87	78
1004	male	group C	bachelor's degree	standard	completed	83	82	84

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93

```

gender                object
race/ethnicity        object
parental level of education  object
lunch                 object
test preparation course object
math score            int64
reading score         int64
writing score         int64
dtype: object

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1005 entries, 0 to 1004
Data columns (total 8 columns):
#   Column                                Non-Null Count  Dtype
---  ---                                -
0   gender                                1005 non-null   object
1   race/ethnicity                        1005 non-null   object
2   parental level of education          998 non-null    object
3   lunch                                1005 non-null   object
4   test preparation course              1005 non-null   object
5   math score                          1005 non-null   int64
6   reading score                       1005 non-null   int64
7   writing score                        1005 non-null   int64
dtypes: int64(3), object(5)
```

```
df.describe()
```

	math score	reading score	writing score
count	1005.000000	1005.000000	1005.000000
mean	66.122388	69.185075	68.066667
std	15.173234	14.614215	15.199095
min	0.000000	17.000000	10.000000
25%	57.000000	59.000000	58.000000
50%	66.000000	70.000000	69.000000
75%	77.000000	80.000000	79.000000
max	100.000000	100.000000	100.000000

---- Filtering and Subsetting ----

Students with math score > 70:

	gender	race/ethnicity	parental level of education	lunch
0	female	group B	bachelor's degree	standard
2	female	group B	master's degree	standard
4	male	group C	some college	standard
5	female	group B	associate's degree	standard
6	female	group B	some college	standard
...
995	female	group E	master's degree	standard
999	female	group D	some college	free/reduced
1000	male	group D	some college	standard
1003	male	group E	some high school	standard
1004	male	group C	bachelor's degree	standard

	test preparation course	math score	reading score	writing score
0	none	72	72	74
2	none	90	95	93
4	none	76	78	75
5	none	71	83	78
6	completed	88	95	92
...
995	completed	88	99	95
999	none	77	86	86
1000	none	76	64	66
1003	none	92	87	78
1004	completed	83	82	84

[394 rows x 8 columns]

```

Female students:
  gender race/ethnicity parental level of education lunch \
0   female      group B      bachelor's degree    standard
1   female      group C      some college         standard
2   female      group B      master's degree    standard
5   female      group B      associate's degree standard
6   female      group B      some college         standard
...   ...      ...      ...      ...
995 female      group E      master's degree    standard
997 female      group C      high school      free/reduced
998 female      group D      some college         standard
999 female      group D      some college      free/reduced
1002 female      group B      bachelor's degree    standard

  test preparation course math score reading score writing score
0      none              72             72             74
1    completed          69             90             88
2      none              90             95             93
5      none              71             83             78
6    completed          88             95             92
...   ...      ...      ...      ...
995    completed          88             99             95
997    completed          59             71             65
998    completed          68             78             77
999      none              77             86             86
1002      none              67             86             83

[519 rows x 8 columns]

```

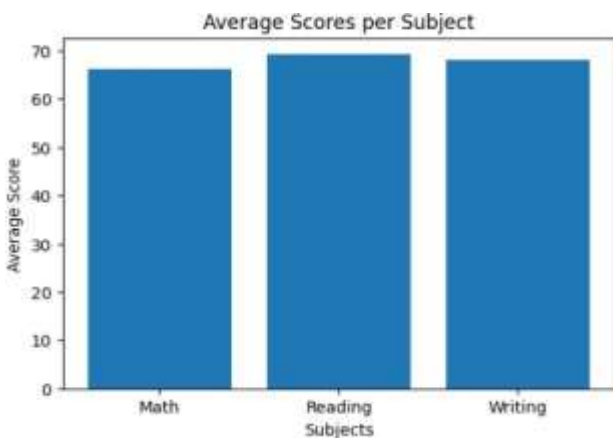
Subset with gender and math score:

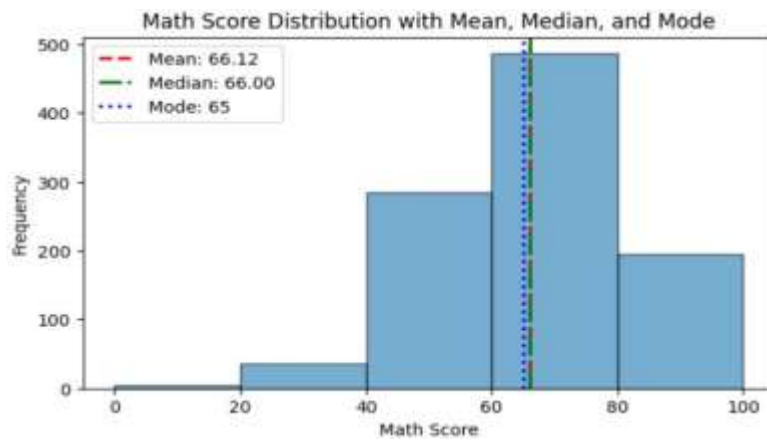
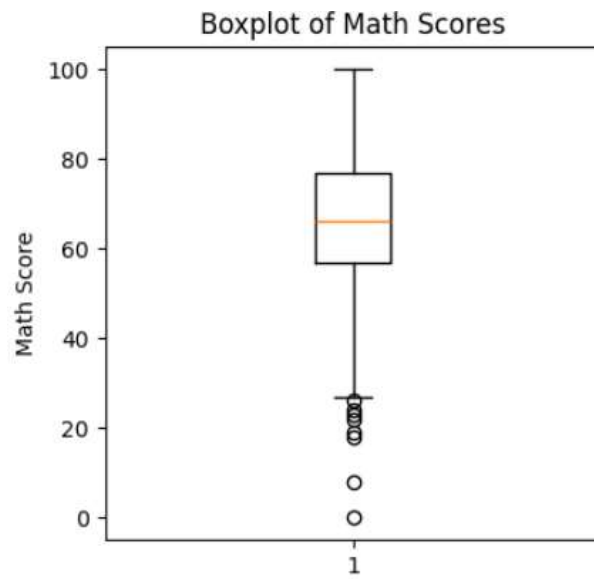
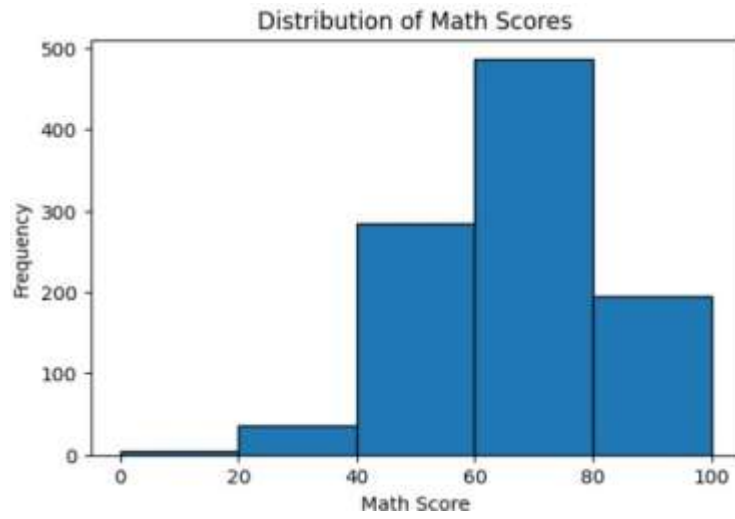
```

  gender math score
0   female      72
1   female      69
2   female      90
3    male      47
4    male      76
...   ...      ...
1000  male      76
1001  male      46
1002 female      67
1003  male      92
1004  male      83

```

[1005 rows x 2 columns]





RESULT:

Thus, the Exploratory Data Analysis (EDA) was successfully performed by viewing, filtering, and summarizing the dataset. Data visualization was done using bar charts, histograms, and boxplots in Matplotlib to better understand the distribution and trends in the students' performance.

EXPT.NO: 5	EDA – DATA VISUALIZATION WITH MATPLOTLIB
DATE: 03/09/2025	

AIM

The Python code aims to perform exploratory data analysis (EDA) by applying preprocessing steps and creating visualizations with Matplotlib. This helps to identify trends, compare group statistics, and observe data distributions using line charts, bar charts, and histograms.

PROBLEM STATEMENT

Raw datasets often contain large amounts of information that are not immediately meaningful. Without proper preprocessing and exploratory data analysis (EDA). Visualization techniques such as line charts, bar charts, and histograms help in summarizing the data and gaining insights.

ALGORITHM

STEP 1: Import pandas for data handling, matplotlib for visualization, and sklearn scalers for preprocessing.

STEP 2: Read the StudentsPerformance.csv dataset into a Pandas DataFrame.

STEP 3: Display the first few rows of the dataset using df.head() to understand its structure.

STEP 4: Group data by reading score and plot average math scores.

STEP 5: Plot separate lines for categories (e.g., gender) for comparison.

STEP 6: Group data by gender and calculate the average writing score.

STEP 7: Plot a bar chart to compare gender-based averages.

STEP 8: Plot the distribution of math scores to observe frequency patterns.

STEP 9: Apply StandardScaler or MinMaxScaler for feature normalization if needed for further analysis.

STEP 10: Analyze visualizations to identify trends, relationships, and score distributions.

SAMPLE CODE:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler

import matplotlib.pyplot as plt

# Step 1: Load dataset
df = pd.read_csv('StudentsPerformance.csv') df.head()

# Step 2: Line Chart - Average math score across reading score levels by
gender for gender in df["gender"].unique():
avg_scores = df[df["gender"] == gender].groupby("reading score")["math
score"].mean() plt.plot(avg_scores.index, avg_scores.values, marker='o',
label=gender)

plt.title("Average Math Score vs. Reading Score by Gender")
plt.xlabel("Reading Score")
plt.ylabel("Average Math Score") plt.legend()
plt.show()

# Step 3: Bar Chart - Average writing score by gender avg_writing =
df.groupby("gender")["writing score"].mean()

plt.bar(avg_writing.index, avg_writing.values, color=['skyblue',
'orange'], edgecolor='black')

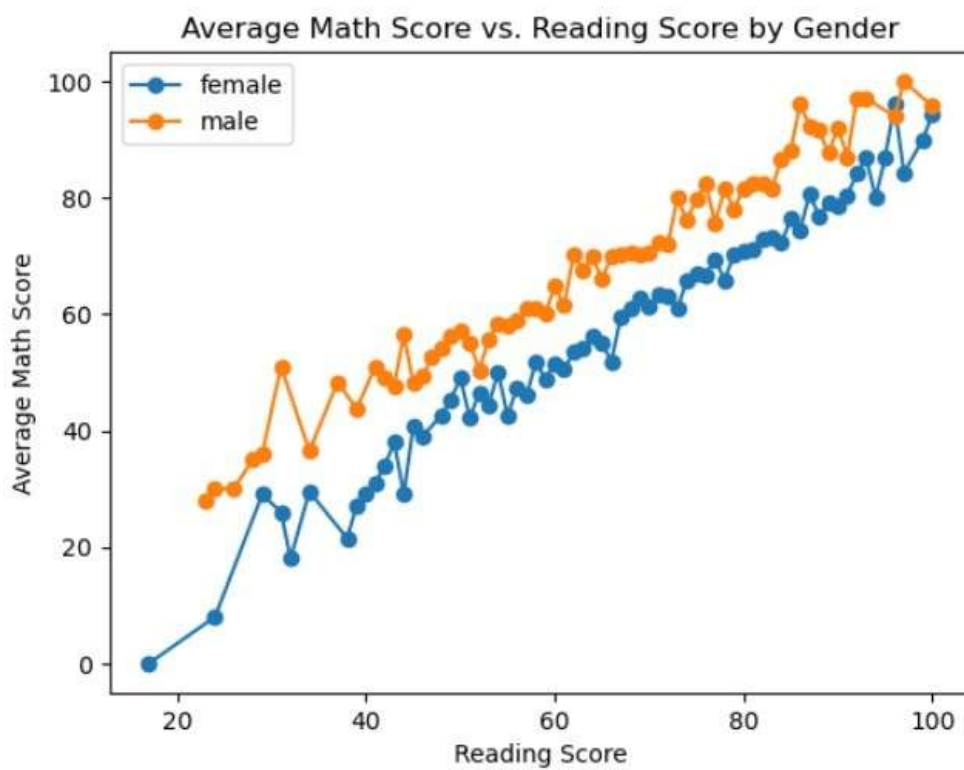
# Add values on top of bars
for i, val in enumerate(avg_writing.values):
plt.text(i, val + 0.5, round(val, 1), ha='center', fontsize=10)

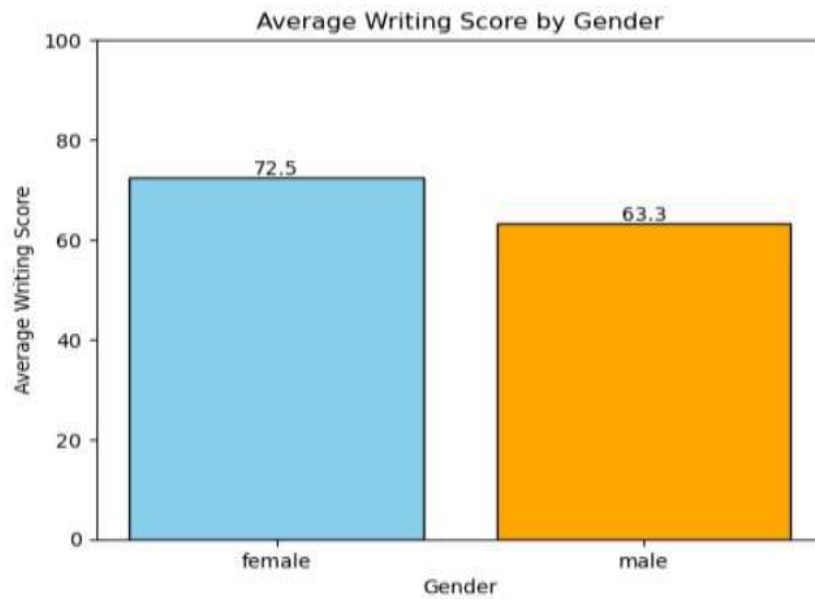
plt.title("Average Writing Score by Gender") plt.xlabel("Gender")
plt.ylabel("Average Writing Score")
plt.ylim(0, 100) # keep y-axis within score range plt.show()

# Step 4: Histogram - Distribution of math scores
plt.hist(df["math score"], bins=20, edgecolor='black', color='skyblue')
plt.title("Distribution of Math Scores")
plt.xlabel("Math Score") plt.ylabel("Number of Students")
plt.xlim(0, 100) # since scores are between 0-100 plt.show()
```

OUTPUT:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75





RESULT

Thus the EDA with data visualization with matplotlib was done using line, bar, and histogram charts.

Exp No: 6 Date:	Data Visualization Using Power BI
----------------------------------	------------------------------------------

Aim:

To learn the Power BI interface and develop skills in connecting to various data sources (Excel, CSV, SQL databases), creating basic visualizations (bar charts, line charts, pie charts), using calculated columns and measures, and building interactive dashboards.

Procedure:

Step 1: Launch Power BI Desktop

- Open Power BI Desktop. Familiarize yourself with the interface like Explore Ribbon (Home, Insert, Modeling, View), Fields Pane (contains tables and columns), Visualizations Pane, Report Canvas

Step 2: Connect to Data Sources

- Home → Get Data.
- Choose the data source type:
- Excel: Browse and select an Excel file, select sheets, and click Load.
- CSV: Browse and select the CSV file, preview, and click Load.
- SQL Database: Enter server name, database, credentials, select tables, and click Load.
- Ensure the data appears in the Fields Pane.

Step 3: Create Basic Visualizations

- Select a visualization type from the Visualizations Pane:
- Bar Chart: Drag a categorical field to the Axis and a numerical field to Values.
- Line Chart: Drag a time/date field to Axis and numerical field to Values.
- Pie Chart: Drag a categorical field to Legend and a numerical field to Values.

- Format charts using the Format options (colors, labels, titles).

Step 4: Create Calculated Columns and Measures

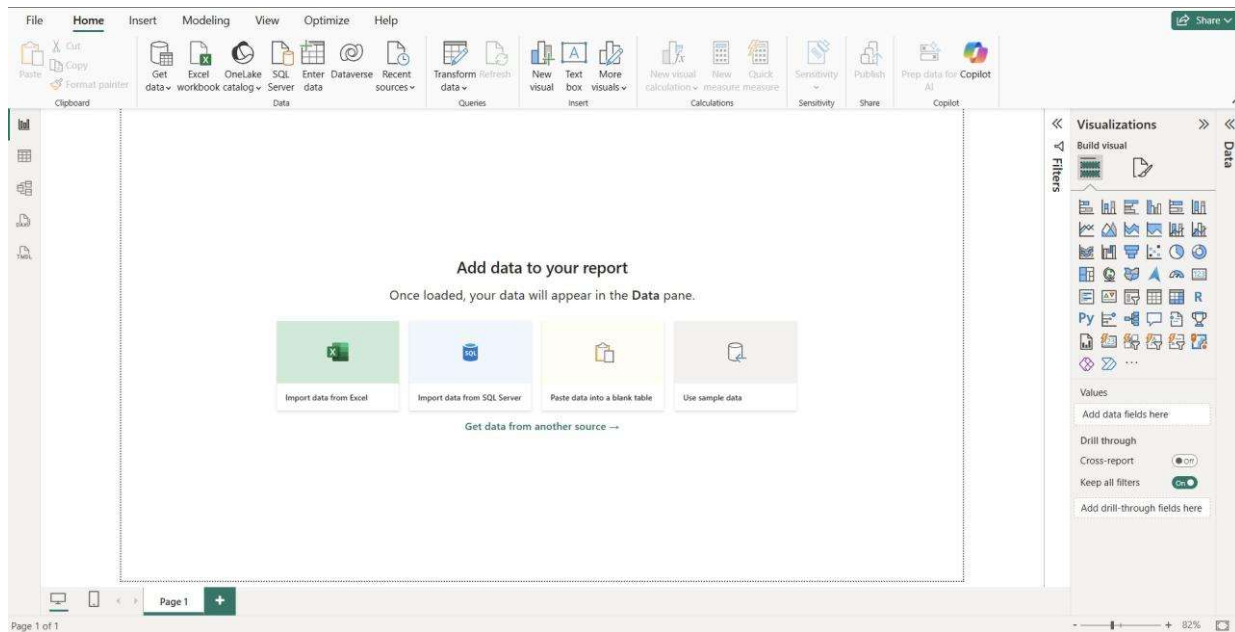
- Calculated Column:
 - Go to Modeling → New Column.
 - Enter DAX formula, e.g., $\text{TotalPrice} = \text{Quantity} * \text{UnitPrice}$.
- Measure:
 - Go to Modeling → New Measure.
 - Enter DAX formula, e.g., $\text{TotalSales} = \text{SUM}(\text{Sales}[\text{TotalPrice}])$.
 - Use these new fields in your visualizations.

Step 5: Build Dashboards

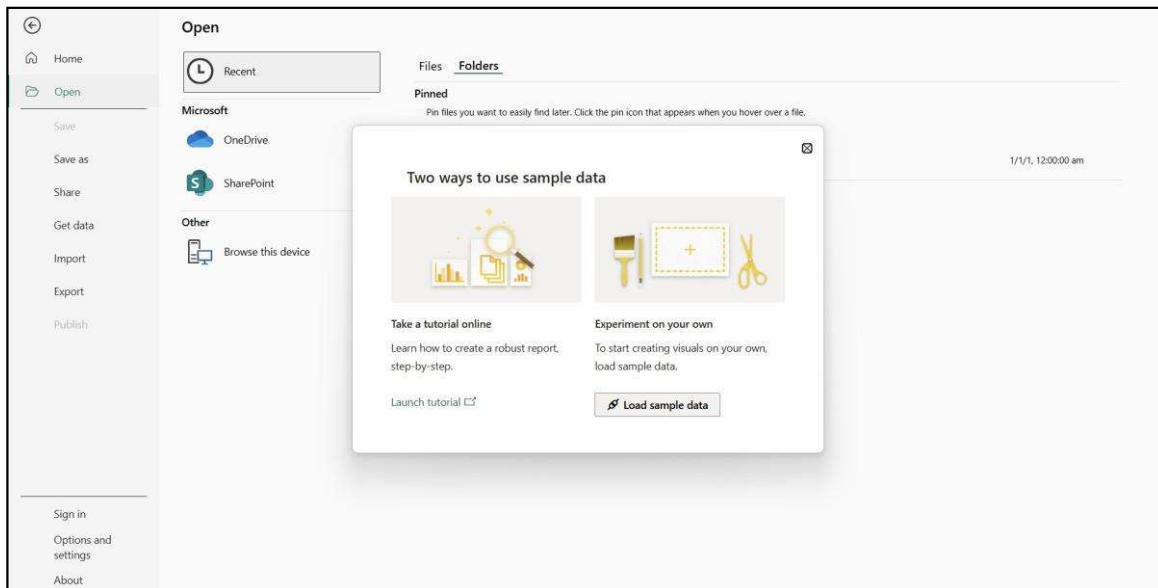
- Arrange multiple visualizations on a single Report Canvas.
- Add slicers to filter data dynamically (e.g., by region or date).
- Customize layout, colors, and titles for readability.
- Save the report: File → Save As.

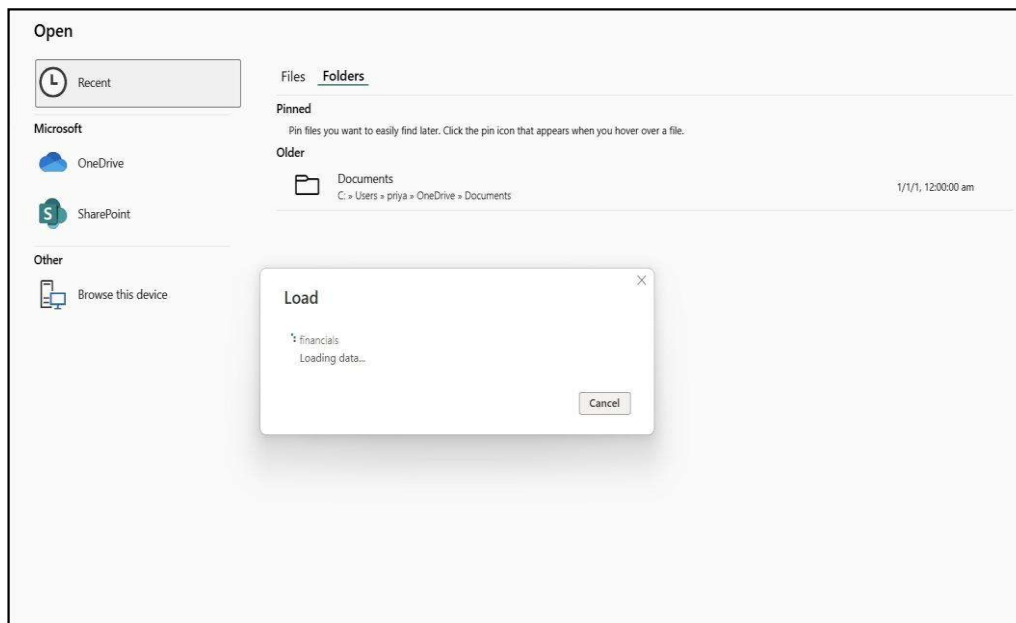
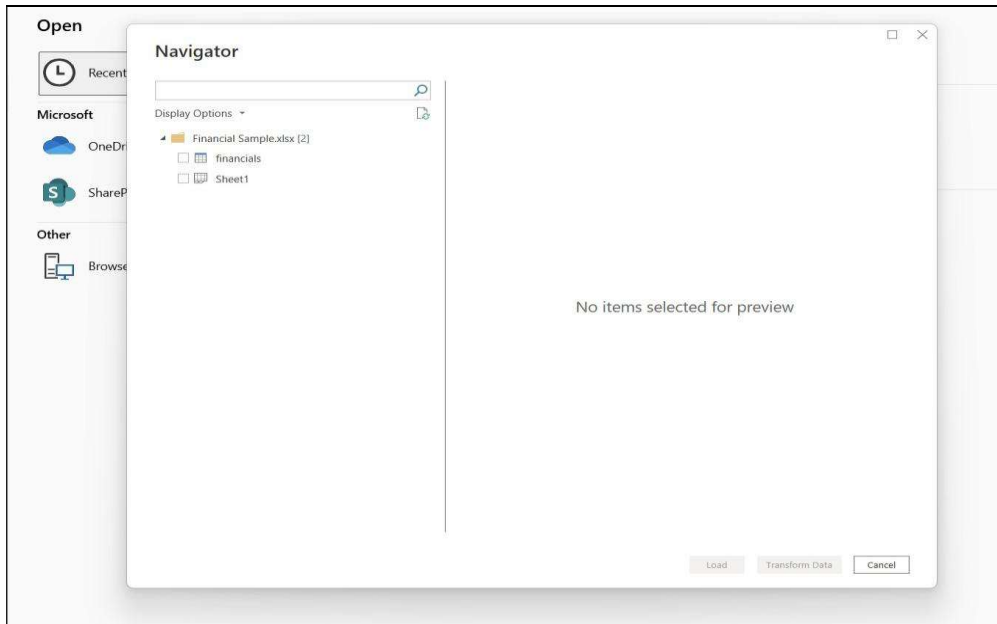
Execution Steps

6.1 Learning the Power BI Interface

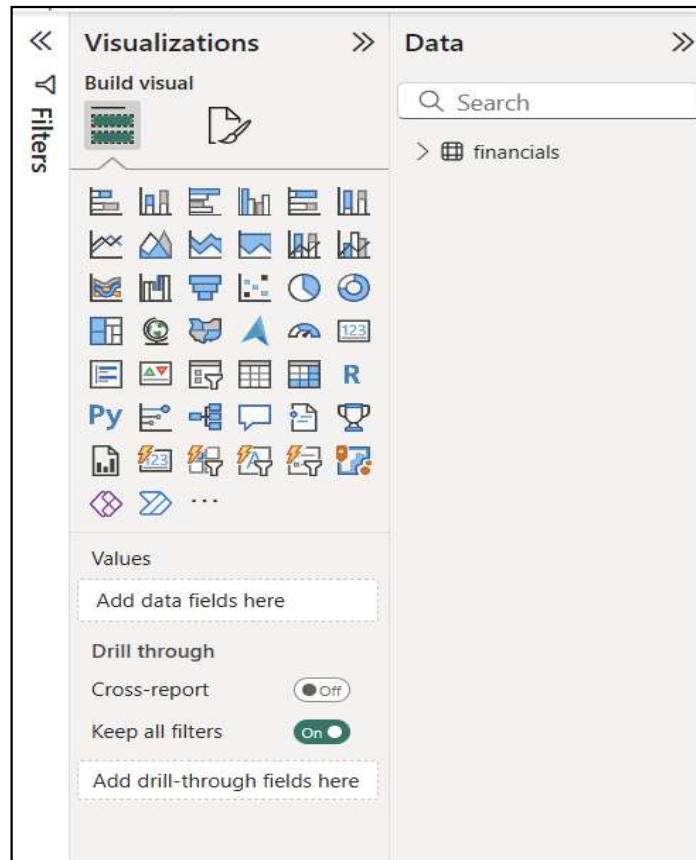


6.2 Connecting to various data sources (Excel, CSV, SQL databases)

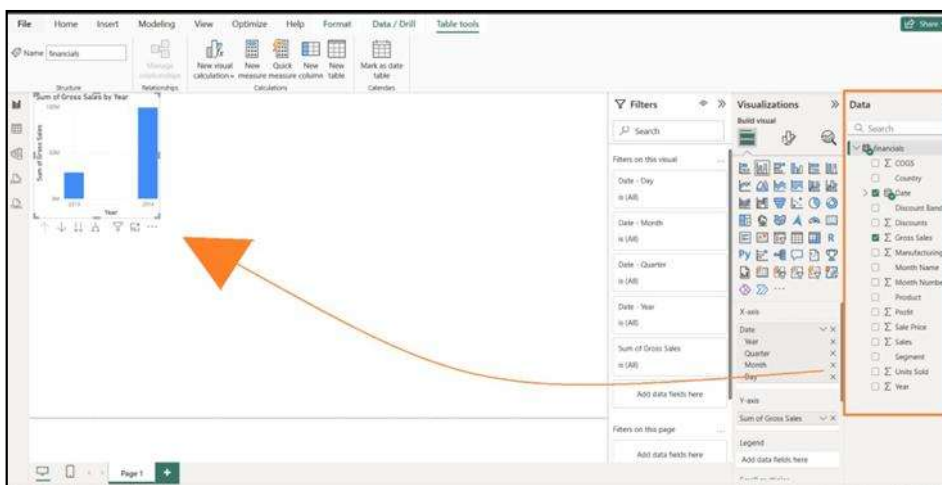




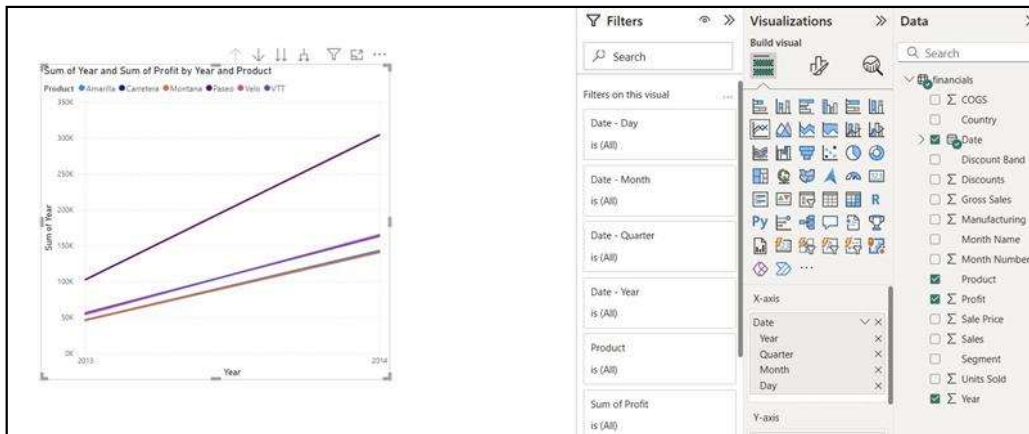
6.3 Creating basic visualizations: bar charts, line charts, pie charts



Bar charts

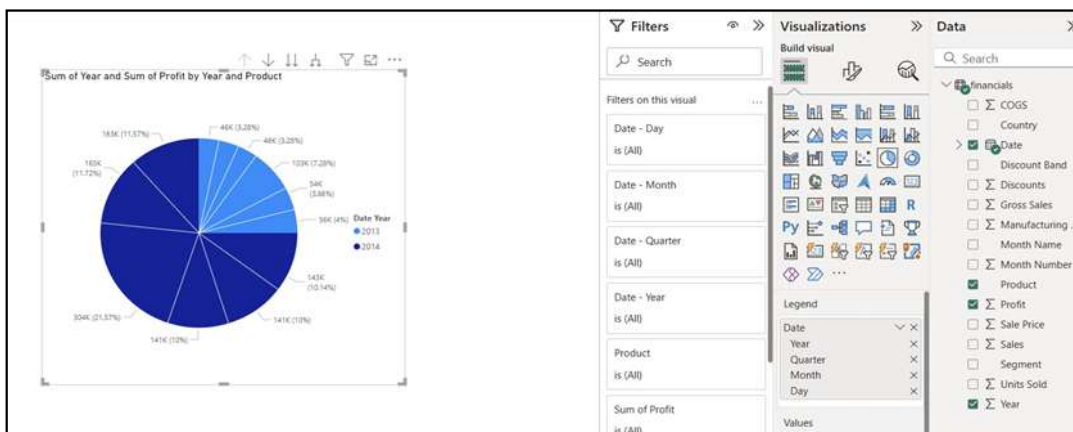


Line chart

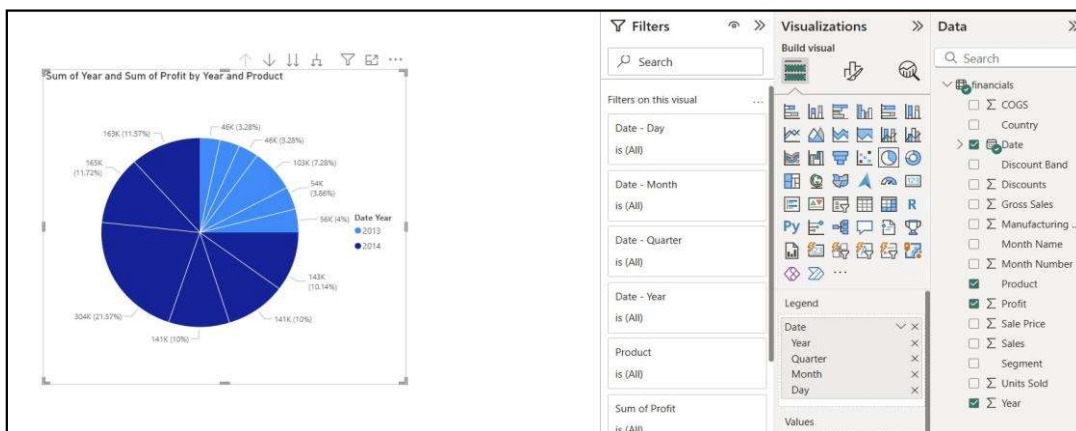
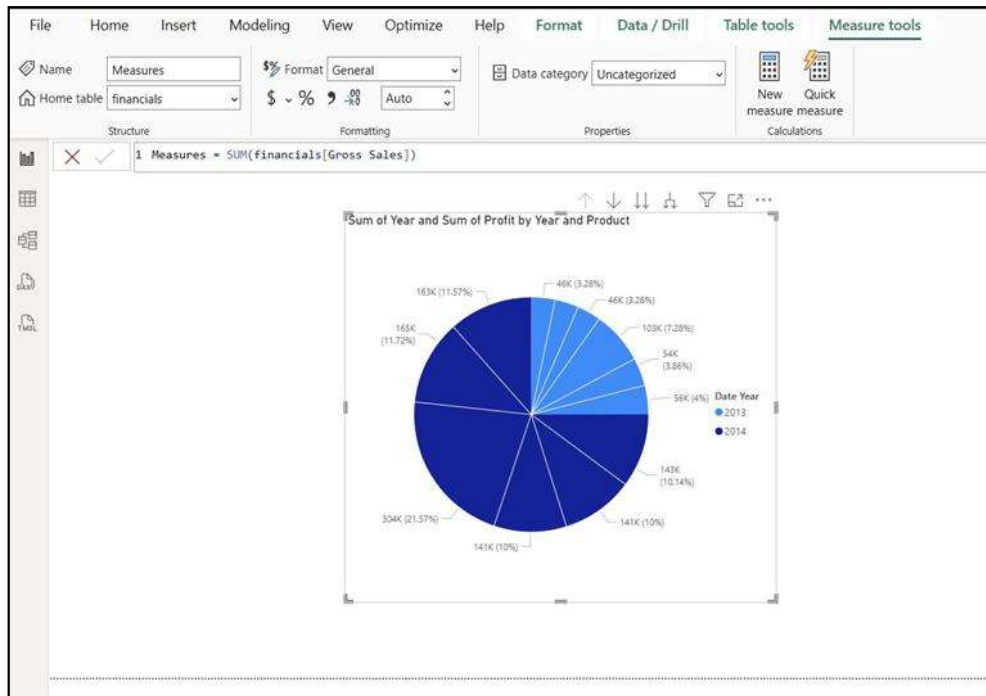


Pie chart

6.4 Creating Calculated Columns and Measures



Go to Modelling, select new measure, upload the formula.



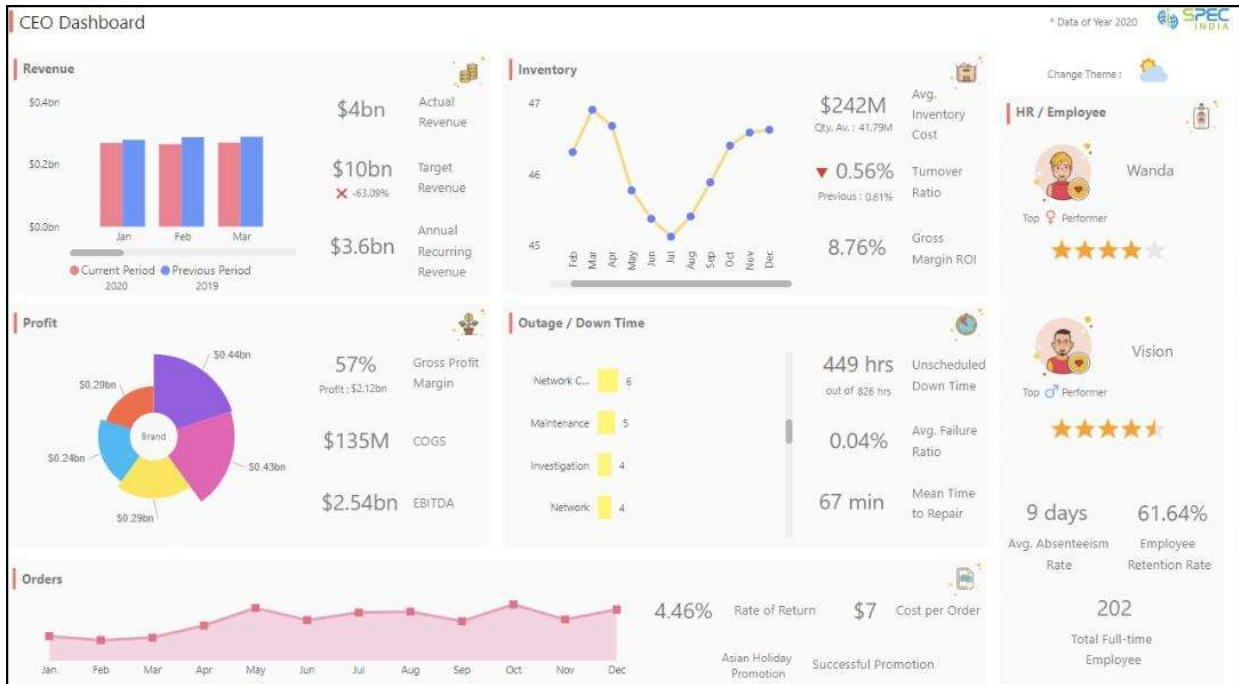
6.5 Building Dashboards

Create a report in Power BI Desktop by following the above mentioned steps and save it.

Publish this to Power BI Service: In Power BI Desktop → Home → Publish. Next, sign in → Select My Workspace (or a shared workspace). Create a dashboard in power BI Service.

Arrange, customize and save the final result.

Sample Power BI Dashboard:



Result:

This experiment provided hands-on experience with Power BI, including connecting to different data sources, creating basic visualizations, performing calculations with DAX, and building interactive dashboards. The process improves data analysis and visualization skills for business intelligence applications.

Exp No: 7 Date:	Data Visualization Using Power BI
----------------------------------	------------------------------------------

Aim:

To learn the Tableau interface and develop skills in connecting to various data sources (Excel, CSV, SQL databases), creating basic visualizations (bar charts, line charts, pie charts), creating calculated fields, and building interactive dashboards and stories.

Procedure:

Step 1: Launch Tableau and Explore Interface

- Open Tableau Desktop.
- Familiarize yourself with the interface:
- Start Page (Connect pane, Open options)
- Data Pane (lists tables and fields)
- Sheets (for building visualizations)
- Dashboard and Story tabs

Step 2: Connect to Data Sources

- In the Connect pane, choose your data source type:
- Excel: Browse and select an Excel file, choose the sheet, and click Sheet1.
- CSV: Browse and select the CSV file, click Sheet1.
- SQL Database: Enter server, database credentials, select tables, and connect.
- Ensure your data appears in the Data Pane.

Step 3: Create Basic Visualizations

- Drag fields from the Data Pane to the Rows and Columns shelves:
- Bar Chart: Place a categorical field on Columns and a numerical field on Rows.
- Line Chart: Place a time/date field on Columns and a numerical field on Rows.
- Pie Chart: Use Marks → Pie, drag a categorical field to Color and numerical field to Angle.
- Use the Show Me panel to explore recommended visualization types.
- Format visualizations with colors, labels, and titles.

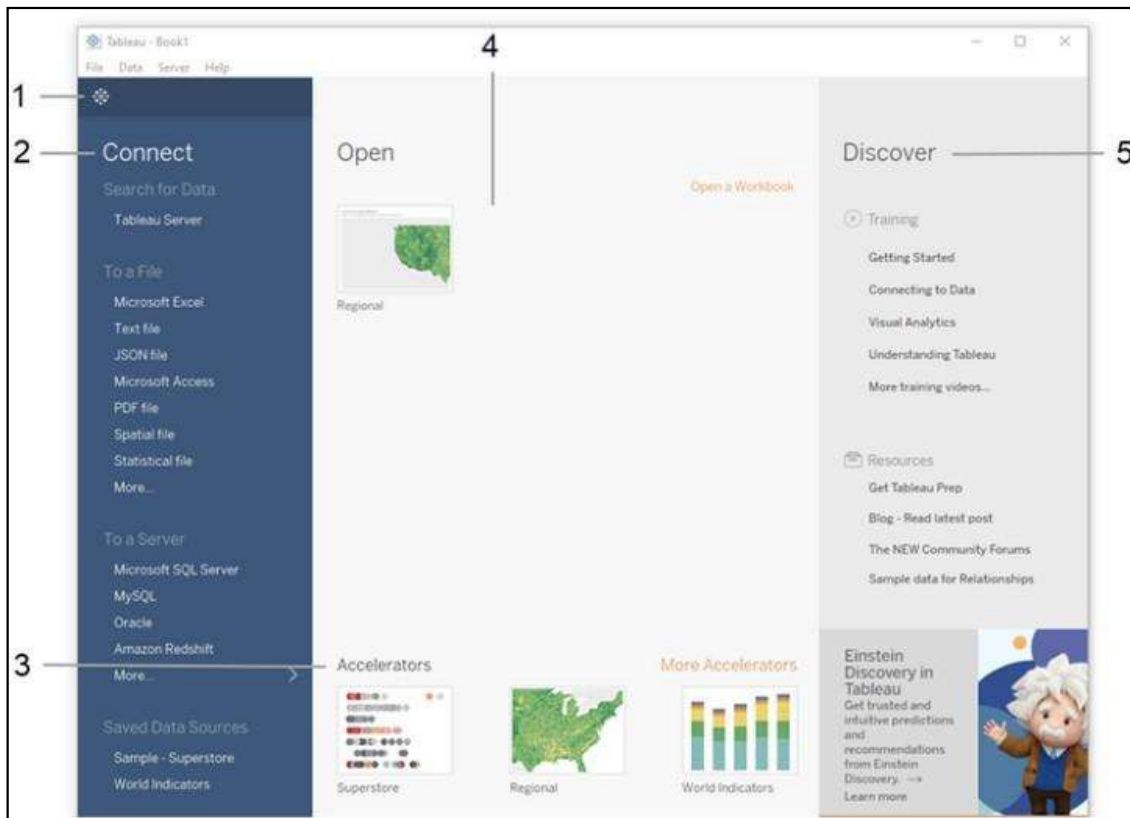
Step 4: Create Calculated Fields

- Click Analysis → Create Calculated Field.
- Enter a formula, e.g., $\text{TotalSales} = [\text{Quantity}] * [\text{UnitPrice}]$.
- Use the calculated field in your visualizations to enhance insights.

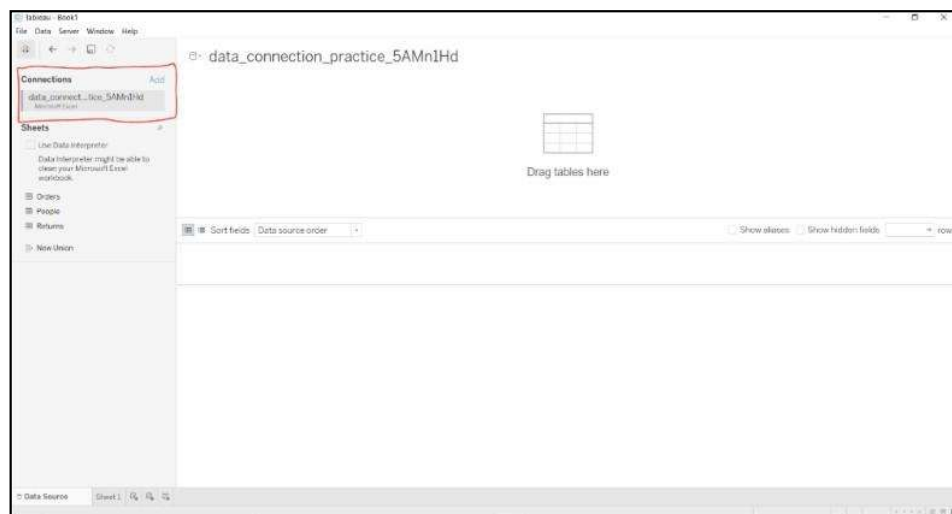
Step 5: Build Dashboards and Stories

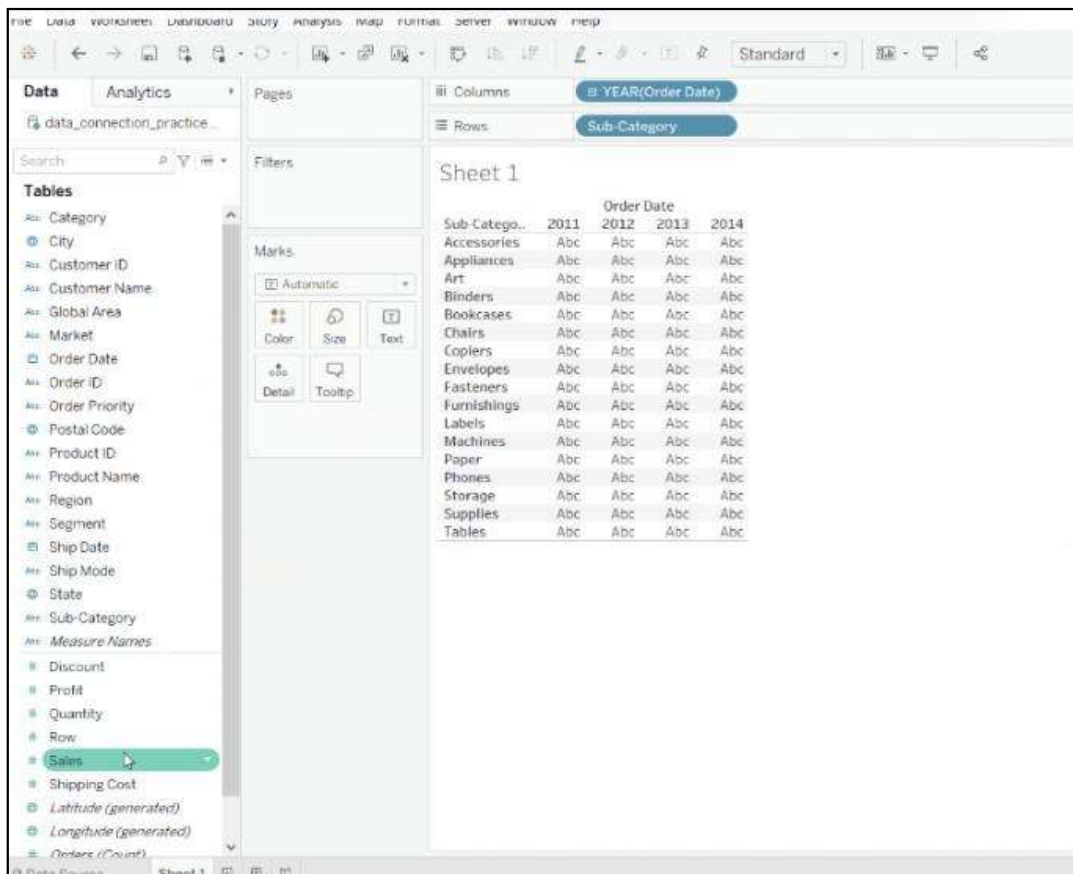
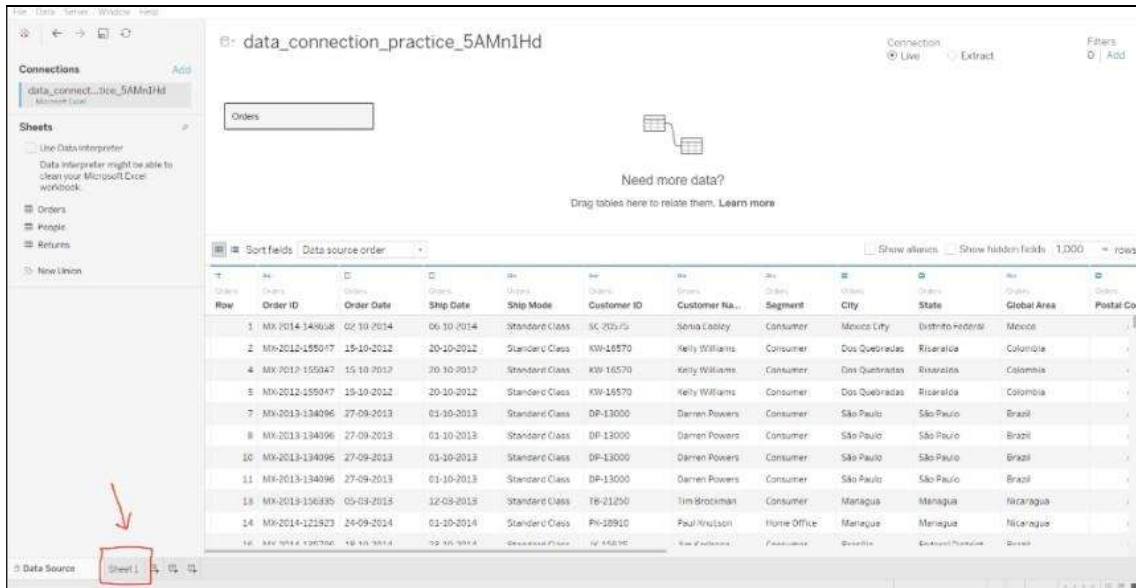
- Click Dashboard → New Dashboard:
- Drag multiple sheets onto the dashboard canvas.
- Add filters, legends, and interactivity.
- Click Story → New Story:
- Combine multiple dashboards and visualizations into a narrative format.
- Add captions and navigation points.
- Customize layout, formatting, and interactivity for clarity.
- Save the workbook: File → Save As.

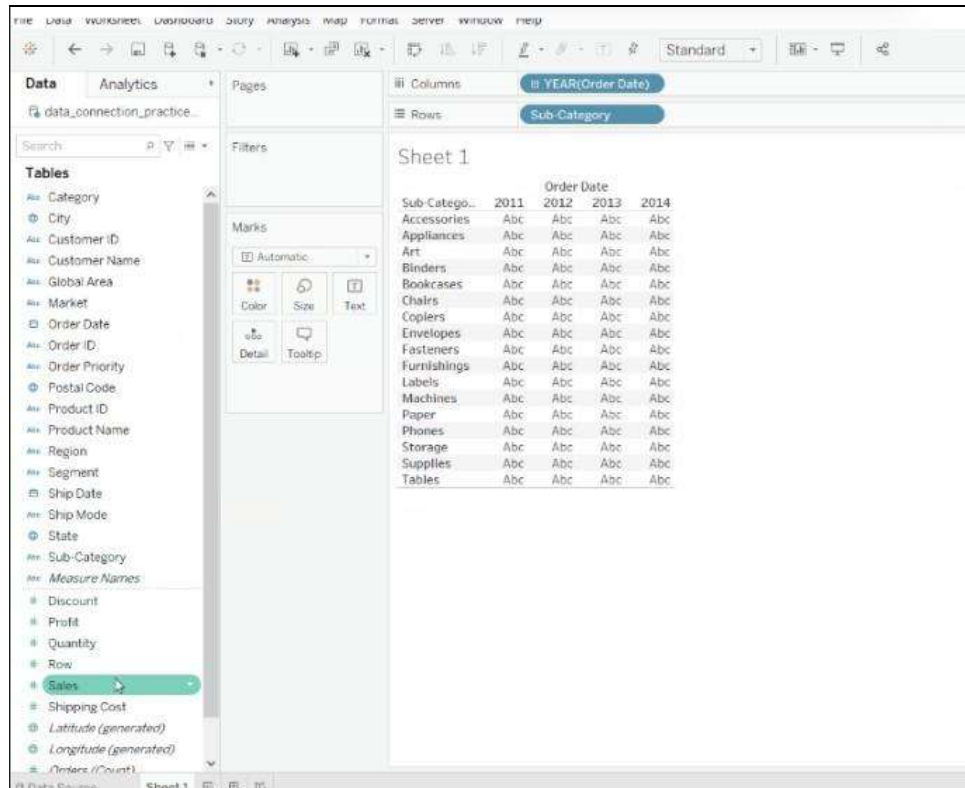
7.1 Introduction to Tableau and its interface



Connecting to various data sources (Excel, CSV, SQL databases)

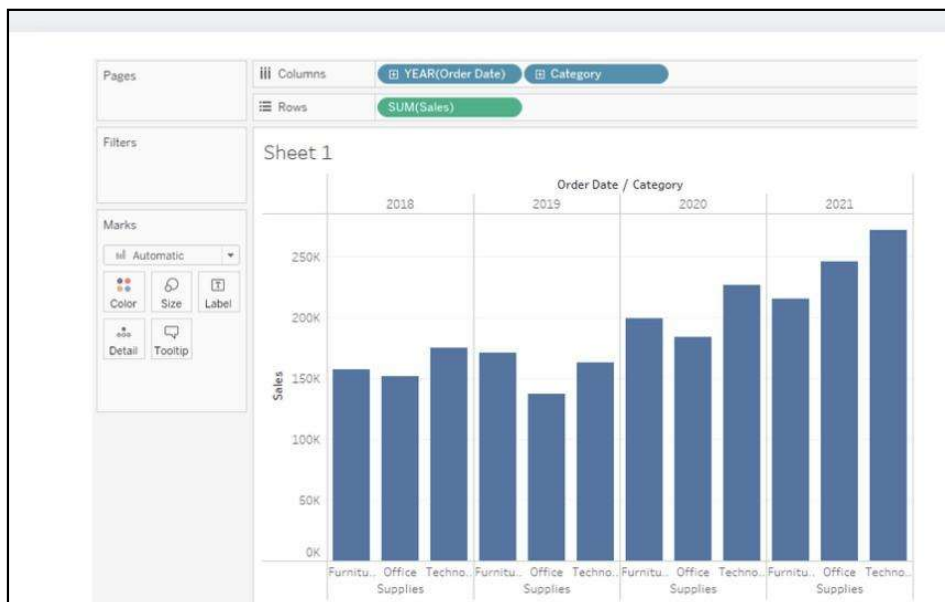




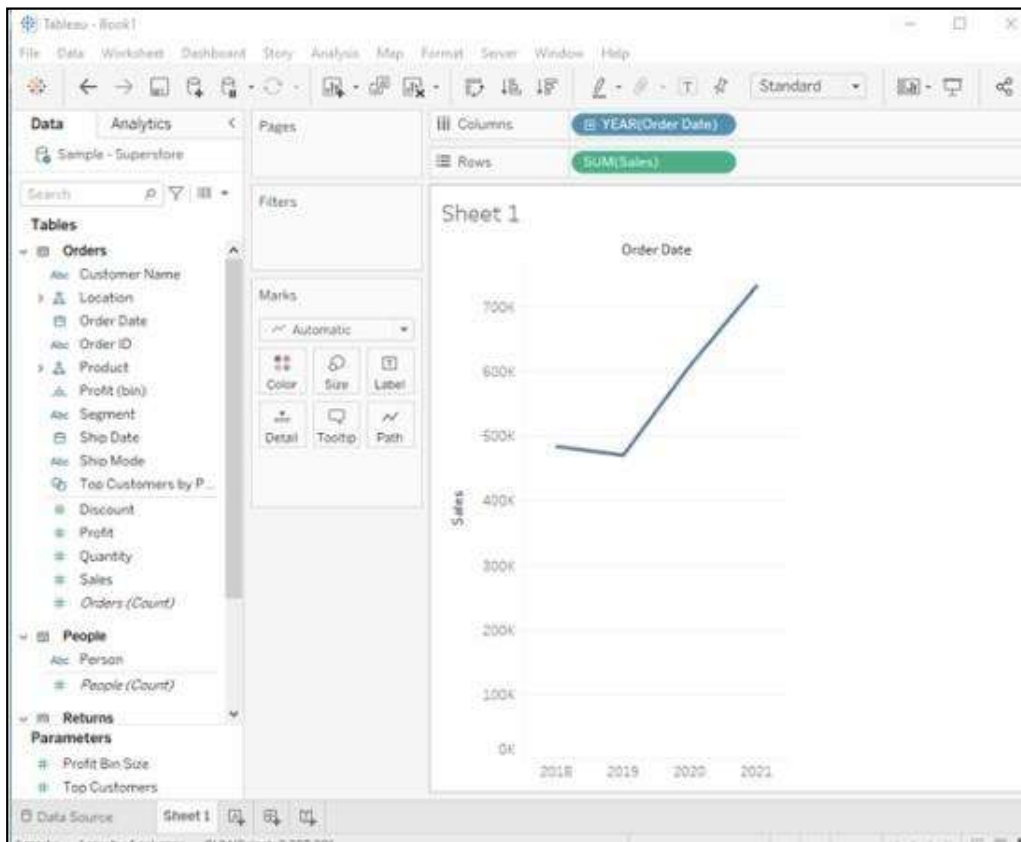


7.2 Creating basic visualizations: bar charts, line charts,

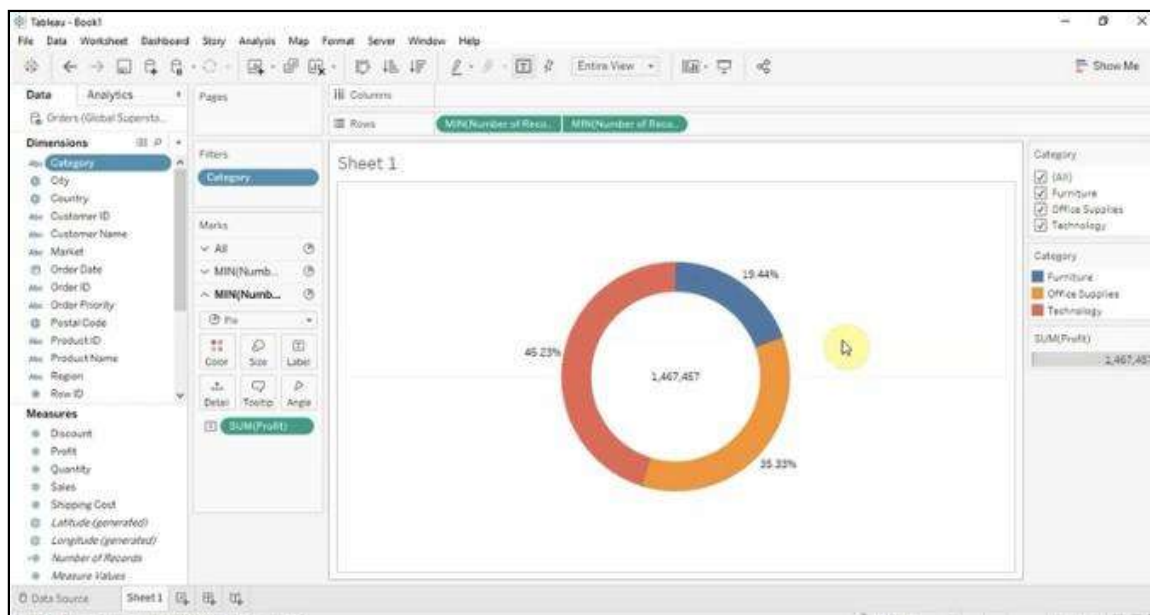
pie charts Bar chart



Line chart

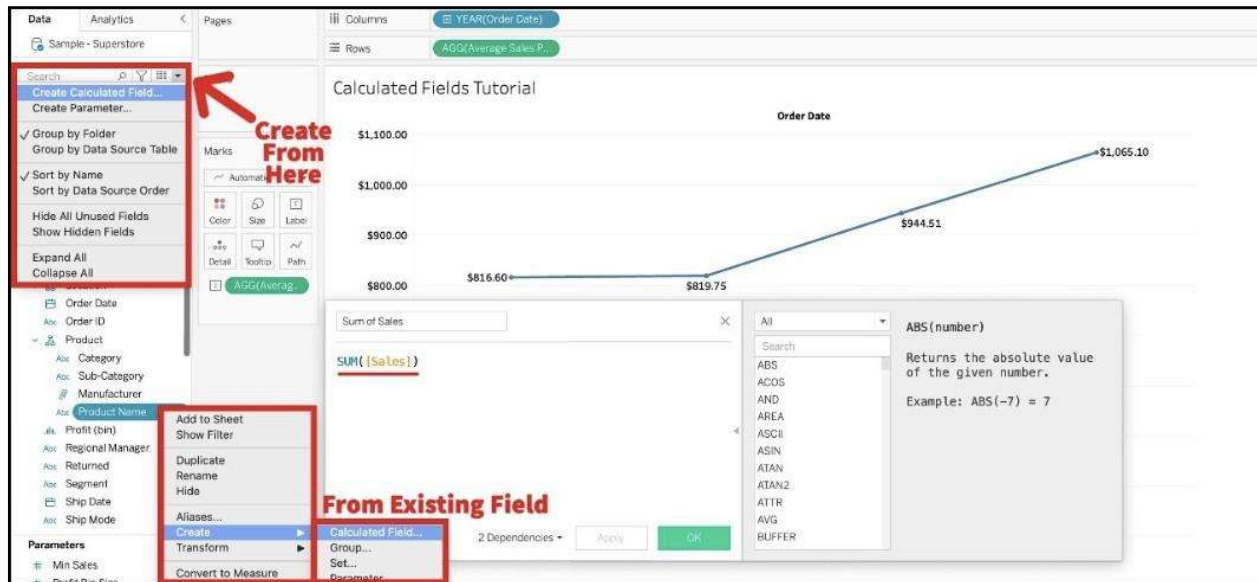


Pie chart



7.3 Creating calculated fields

A calculated field in Tableau is like making your own math rule using the data you already have



Result:

This experiment introduced Tableau's interface and workflow, including connecting to data sources, creating visualizations, using calculated fields, and building dashboards and stories. Tableau simplifies data exploration and helps communicate insights effectively. To learn the Power BI interface and develop skills in connecting to various data sources (Excel, CSV, SQL databases), creating basic visualizations (bar charts, line charts, pie charts), using calculated columns and measures, and building interactive dashboards.