Ex 1 Setting up the Python environment and libraries - Juypter Notebook

AIM:

To understand the working of Jupyter Notebook, write and execute Python code, create new code and Markdown cells, and demonstrate the use of Jupyter Widgets and Jupyter AI.

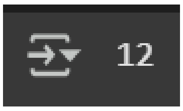
1. Create a new notebook for Python

Open Anaconda Navigator or Jupyter Lab / Notebook.
Click **New** → **Python 3** to open a fresh notebook.

2. Write and execute Python code

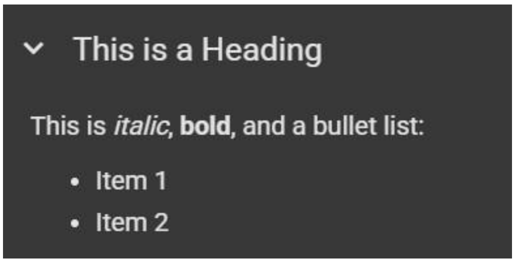
```
a = 5

b = 7

print(a +

b)
```

OUTPUT:



```
12
```

3. Create new cells for code and Markdown



This is a Heading

This is *italic*, **bold**, and a bullet list:

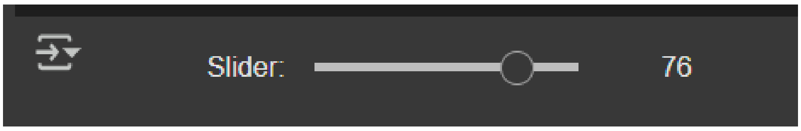
- Item 1
- Item 2

4. Demonstrate the application of Jupyter Widgets, Jupyter AI

```
!jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

```
import ipywidgets as widgets
widgets.IntSlider(
    value=10,
    min=0,
    max=100,
    step=1,
    description='Slider:',
    continuous_update=True
)
```

OUTPUT:



Ex 2 EDA-Data Import and Export

AIM:

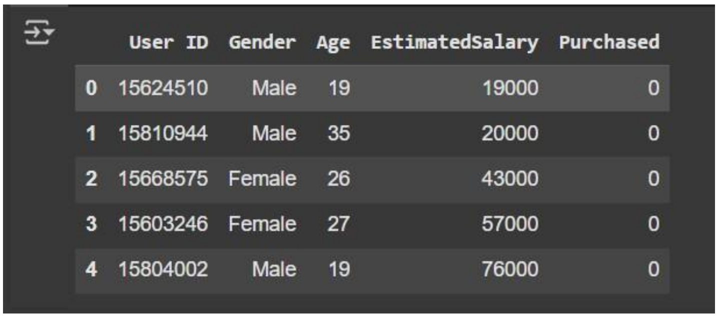
To perform exploratory data analysis by importing data from various sources such as CSV, Excel, SQL, and web scraping, and export DataFrames into Excel and CSV formats using Python.

1. Importing data from CSV, Excel, SQL databases, and web scraping

- **CSV**

```
import pandas as pd

df = pd.read_csv('/content/suv_data.csv')

df.head()
```

OUTPUT:

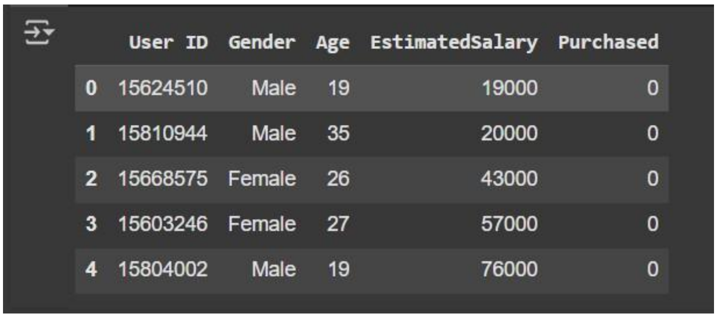


	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

- **EXCEL**

```
!pip install openpyxl

df.to_excel('suv_data.xlsx', index=False)

df2 = pd.read_excel('suv_data.xlsx')

df2.head()
```

OUTPUT:

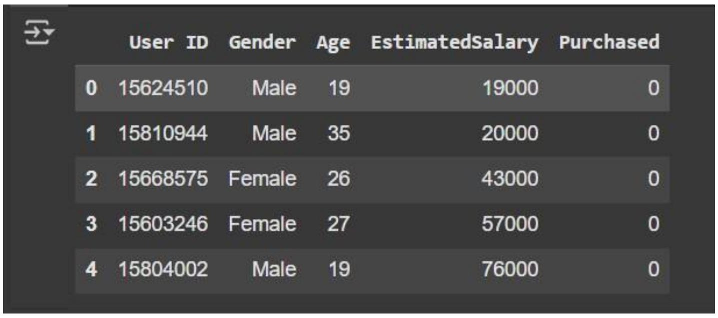


	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

- **SQL DB**

```
import sqlite3
import pandas as pd
conn = sqlite3.connect('mydata.db')
df = pd.read_csv('suv_data.csv')  # Your existing data
df.to_sql('suv_table', conn, if_exists='replace', index=False)  # Store to SQL
df_sql = pd.read_sql_query("SELECT * FROM suv_table", conn)
df_sql.head()
conn.close()
```

OUTPUT:



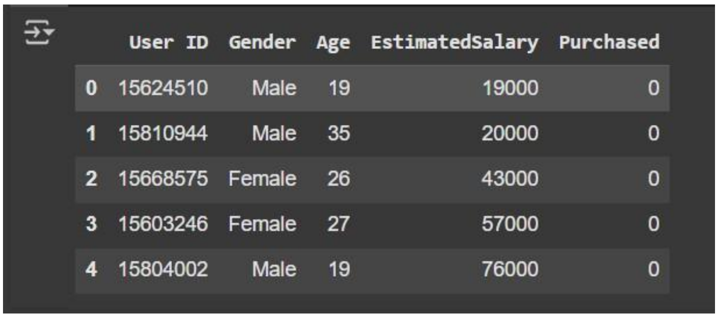
	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

- **WEB SCRAPING**

```
import pandas as pd
```

```
url = 'https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)'
tables = pd.read_html(url)  # This will return a list of tables
print(len(tables))  # See how many tables were found
df_web = tables[1]  # You can try 0, 1, 2, etc.
df_web.head()
```

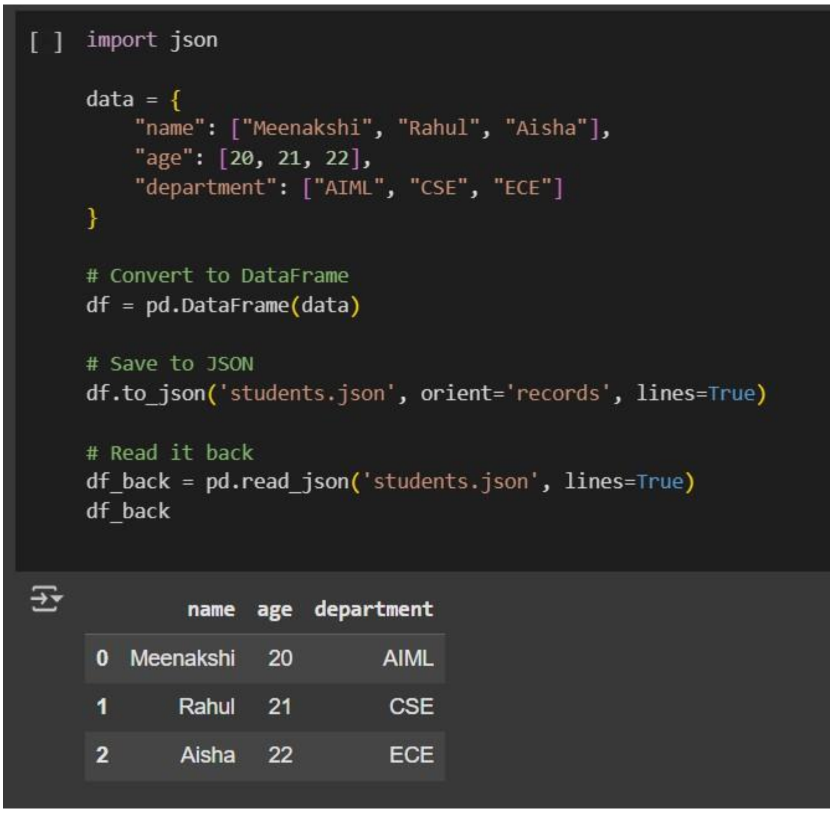
OUTPUT:



	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

2. Handling different data formats

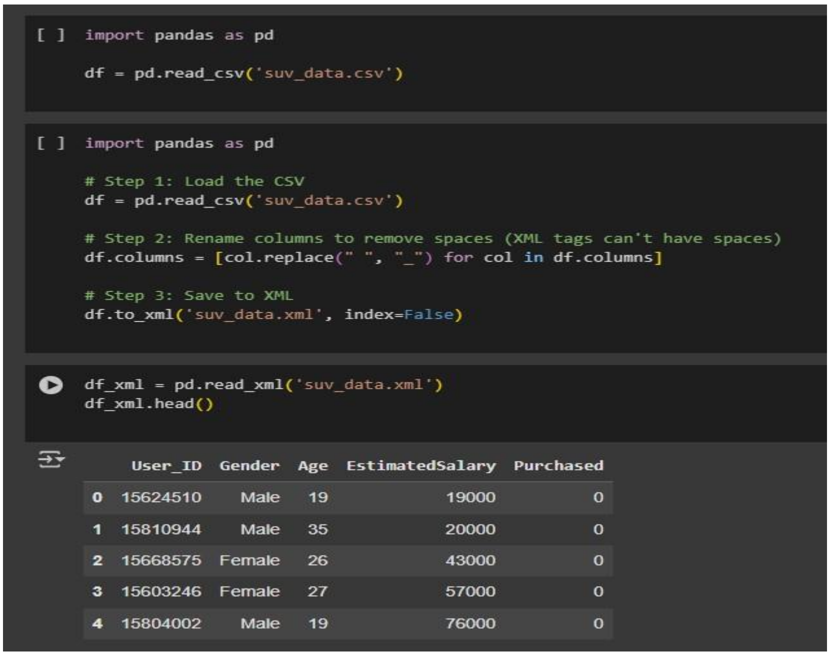
- **JSON**



```
import json

data = {
    "name": ["Meenakshi", "Rahul", "Aisha"],
    "age": [20, 21, 22],
    "department": ["AIML", "CSE", "ECE"]
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Save to JSON
df.to_json('students.json', orient='records', lines=True)

# Read it back
df_back = pd.read_json('students.json', lines=True)
df_back
```

	name	age	department
0	Meenakshi	20	AIML
1	Rahul	21	CSE
2	Aisha	22	ECE

- **XML**



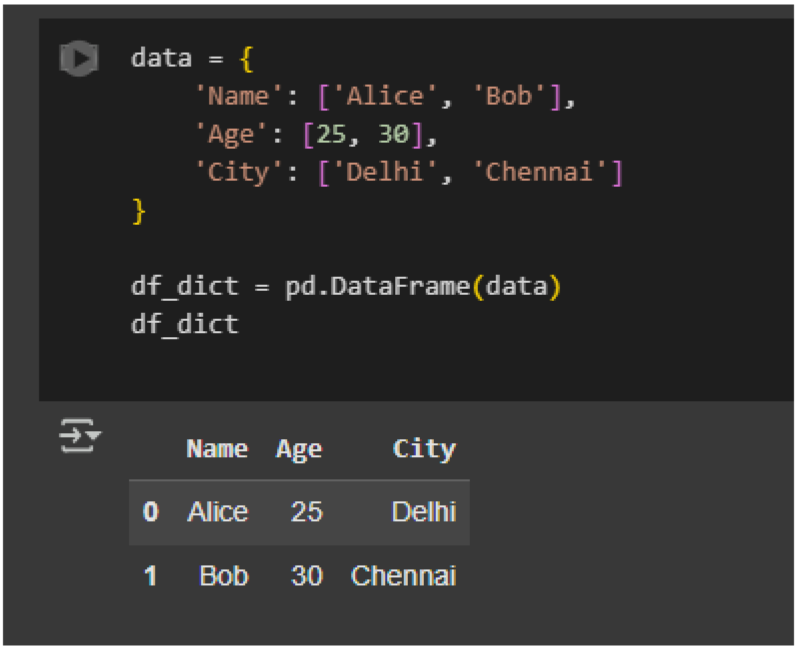
```
import pandas as pd

df = pd.read_csv('suv_data.csv')
```

```
import pandas as pd

# Step 1: Load the CSV
df = pd.read_csv('suv_data.csv')

# Step 2: Rename columns to remove spaces (XML tags can't have spaces)
df.columns = [col.replace(" ", "_") for col in df.columns]

# Step 3: Save to XML
df.to_xml('suv_data.xml', index=False)
```

```
df_xml = pd.read_xml('suv_data.xml')
df_xml.head()
```

	User_ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

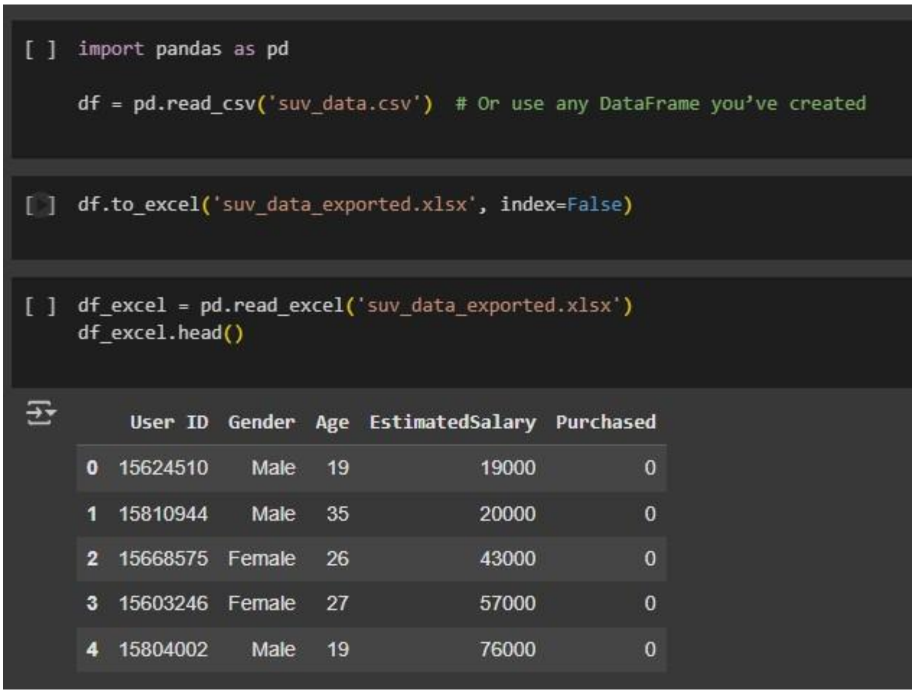
- **PYTHON DICTIONARY**



```python
data = {
    'Name': ['Alice', 'Bob'],
    'Age': [25, 30],
    'City': ['Delhi', 'Chennai']
}

df_dict = pd.DataFrame(data)
df_dict
```

```
    Name  Age     City
0  Alice   25    Delhi
1    Bob   30  Chennai
```

3. Export a DataFrame to an Excel file.



```python
import pandas as pd

df = pd.read_csv('suv_data.csv')  # Or use any DataFrame you’ve created
```

```python
df.to_excel('suv_data_exported.xlsx', index=False)
```

```python
df_excel = pd.read_excel('suv_data_exported.xlsx')
df_excel.head()
```

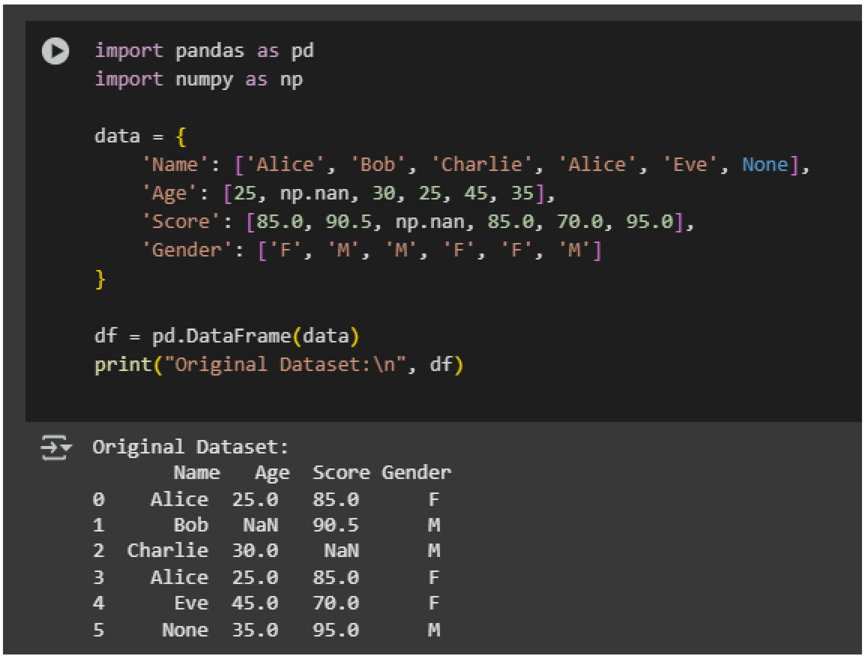
```
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0
```

Ex 3 EDA-Data Cleaning

AIM:

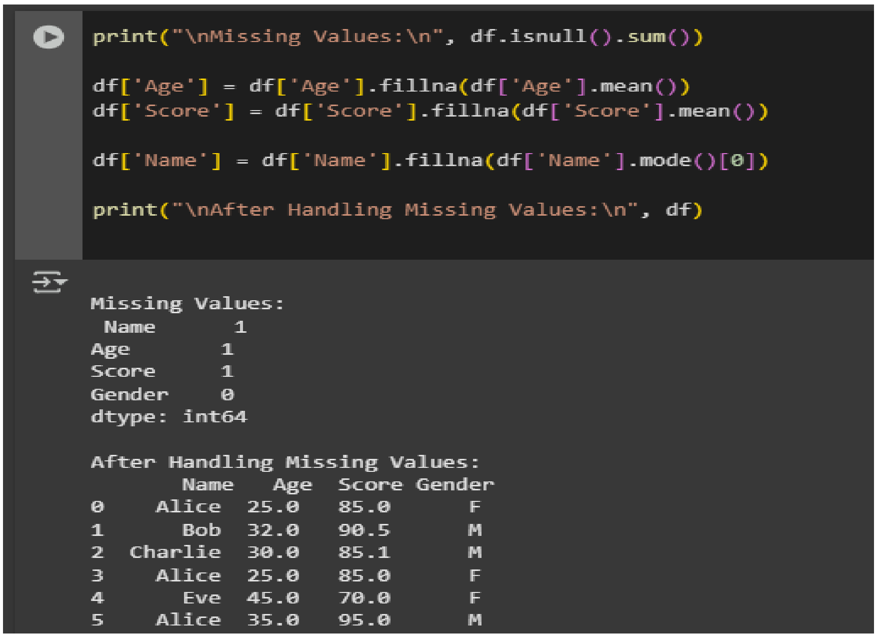
To clean the dataset by handling missing values, removing duplicates, performing data type conversion, and normalizing data using standardization and min-max scaling.

1. Handlingmissing values: detection, filling, and dropping



```
import pandas as pd
import numpy as np

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'Alice', 'Eve', None],
    'Age': [25, np.nan, 30, 25, 45, 35],
    'Score': [85.0, 90.5, np.nan, 85.0, 70.0, 95.0],
    'Gender': ['F', 'M', 'M', 'F', 'F', 'M']
}

df = pd.DataFrame(data)
print("Original Dataset:\n", df)
```

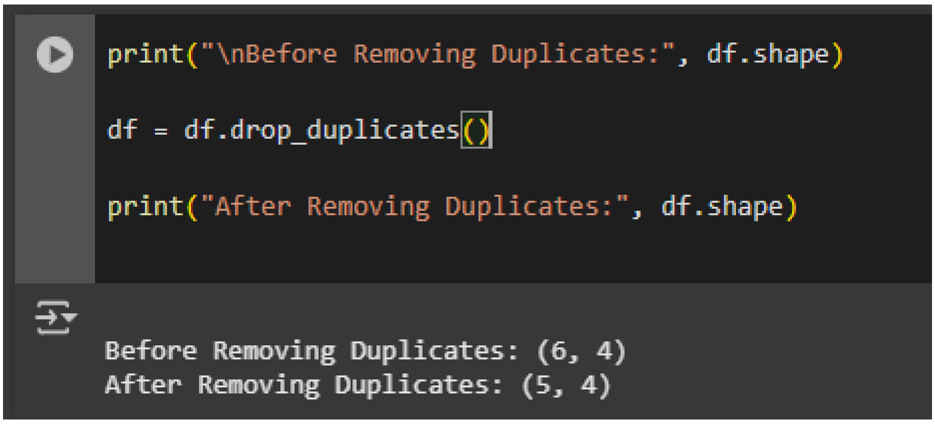
```
Original Dataset:
       Name   Age  Score Gender
0    Alice  25.0   85.0      F
1      Bob   NaN   90.5      M
2  Charlie  30.0    NaN      M
3    Alice  25.0   85.0      F
4      Eve  45.0   70.0      F
5     None  35.0   95.0      M
```



```
print("\nMissing Values:\n", df.isnull().sum())

df['Age'] = df['Age'].fillna(df['Age'].mean())
df['Score'] = df['Score'].fillna(df['Score'].mean())

df['Name'] = df['Name'].fillna(df['Name'].mode()[0])

print("\nAfter Handling Missing Values:\n", df)
```

```
Missing Values:
 Name      1
Age       1
Score     1
Gender    0
dtype: int64

After Handling Missing Values:
       Name   Age  Score Gender
0    Alice  25.0   85.0      F
1      Bob  32.0   90.5      M
2  Charlie  30.0   85.1      M
3    Alice  25.0   85.0      F
4      Eve  45.0   70.0      F
5    Alice  35.0   95.0      M
```

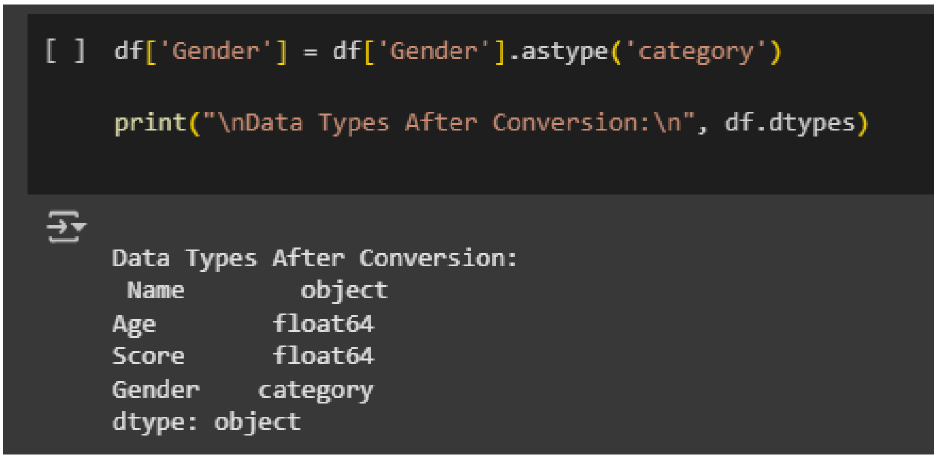
2. Removing duplicates and unnecessary data



```
print("\nBefore Removing Duplicates:", df.shape)

df = df.drop_duplicates()

print("After Removing Duplicates:", df.shape)
```

```
Before Removing Duplicates: (6, 4)
After Removing Duplicates: (5, 4)
```

2. Data type conversion and ensuring consistency



```
df['Gender'] = df['Gender'].astype('category')

print("\nData Types After Conversion:\n", df.dtypes)
```

```
Data Types After Conversion:
 Name        object
Age        float64
Score      float64
Gender    category
dtype: object
```

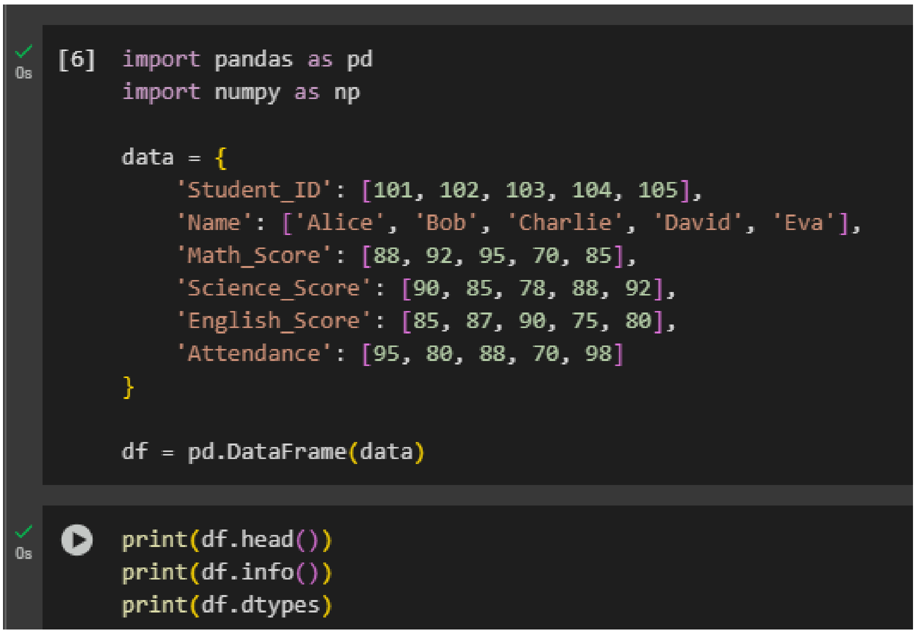
3. Normalize data (e.g., standardization, min-max scaling).

Ex 4 EDA-Data Inspection and Analysis

AIM:

To inspect and analyze datasets by viewing DataFrames, filtering and subsetting data using conditions, and calculating descriptive statistics including measures of central tendency and dispersion.

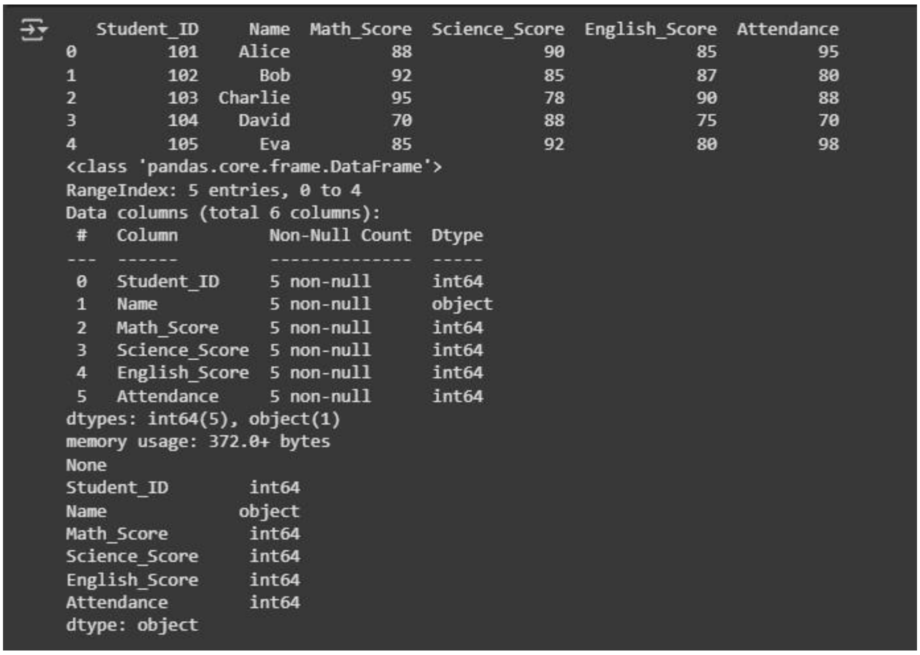
1. Viewing and inspecting DataFrames



```
import pandas as pd
import numpy as np

data = {
    'Student_ID': [101, 102, 103, 104, 105],
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Math_Score': [88, 92, 95, 70, 85],
    'Science_Score': [90, 85, 78, 88, 92],
    'English_Score': [85, 87, 90, 75, 80],
    'Attendance': [95, 80, 88, 70, 98]
}

df = pd.DataFrame(data)
```

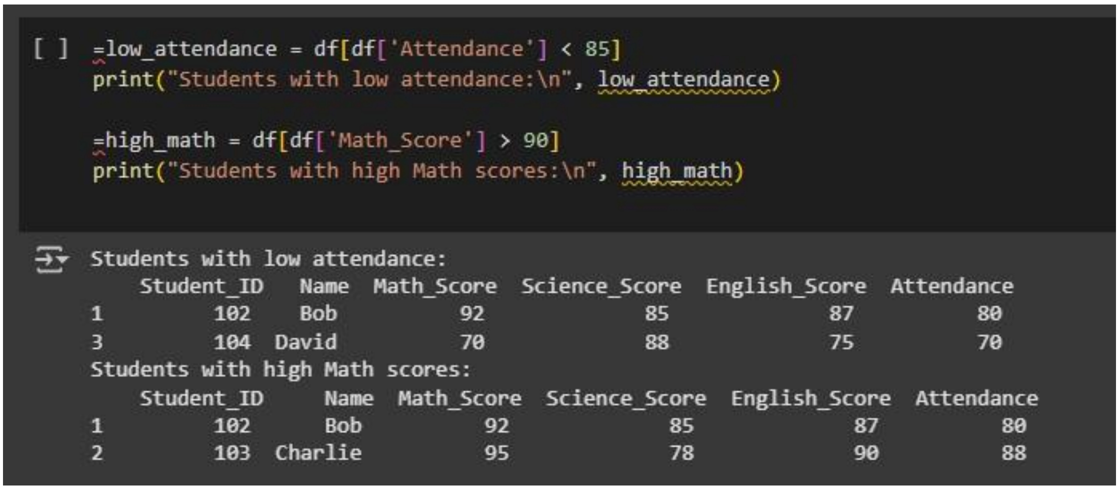
```
print(df.head())
print(df.info())
print(df.dtypes)
```

OUTPUT:



```
   Student_ID     Name  Math_Score  Science_Score  English_Score  Attendance
0         101    Alice          88             90             85          95
1         102      Bob          92             85             87          80
2         103  Charlie          95             78             90          88
3         104    David          70             88             75          70
4         105      Eva          85             92             80          98
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Student_ID     5 non-null      int64 
 1   Name           5 non-null      object
 2   Math_Score     5 non-null      int64 
 3   Science_Score  5 non-null      int64 
 4   English_Score  5 non-null      int64 
 5   Attendance     5 non-null      int64 
dtypes: int64(5), object(1)
memory usage: 372.0+ bytes
None
Student_ID        int64
Name             object
Math_Score        int64
Science_Score     int64
English_Score     int64
Attendance        int64
dtype: object
```

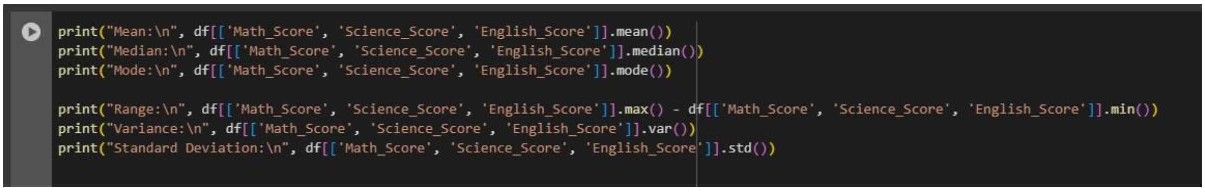
2. Filtering and subsetting data using conditions



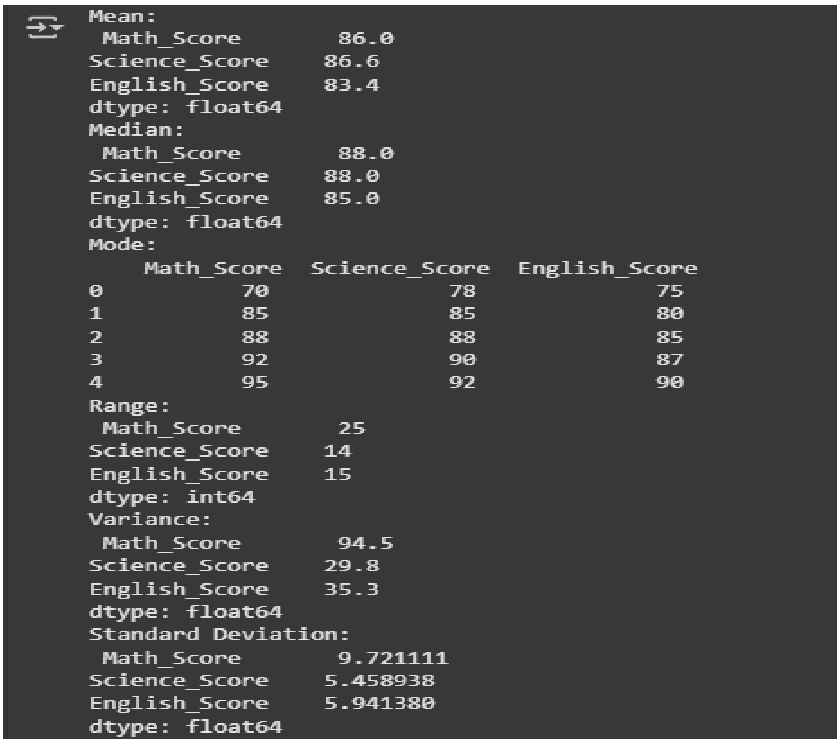
```
[ ] =low_attendance = df[df['Attendance'] < 85]
    print("Students with low attendance:\n", low_attendance)

    =high_math = df[df['Math_Score'] > 90]
    print("Students with high Math scores:\n", high_math)
```

```
Students with low attendance:
    Student_ID   Name  Math_Score  Science_Score  English_Score  Attendance
1         102    Bob          92             85             87          80
3         104  David          70             88             75          70
Students with high Math scores:
    Student_ID     Name  Math_Score  Science_Score  English_Score  Attendance
1         102      Bob          92             85             87          80
2         103  Charlie          95             78             90          88
```

3. Descriptive statistics: measures of central tendency (mean, median, mode) and measures of dispersion (range, variance, standard deviation)



```
print("Mean:\n", df[['Math_Score', 'Science_Score', 'English_Score']].mean())
print("Median:\n", df[['Math_Score', 'Science_Score', 'English_Score']].median())
print("Mode:\n", df[['Math_Score', 'Science_Score', 'English_Score']].mode())

print("Range:\n", df[['Math_Score', 'Science_Score', 'English_Score']].max() - df[['Math_Score', 'Science_Score', 'English_Score']].min())
print("Variance:\n", df[['Math_Score', 'Science_Score', 'English_Score']].var())
print("Standard Deviation:\n", df[['Math_Score', 'Science_Score', 'English_Score']].std())
```

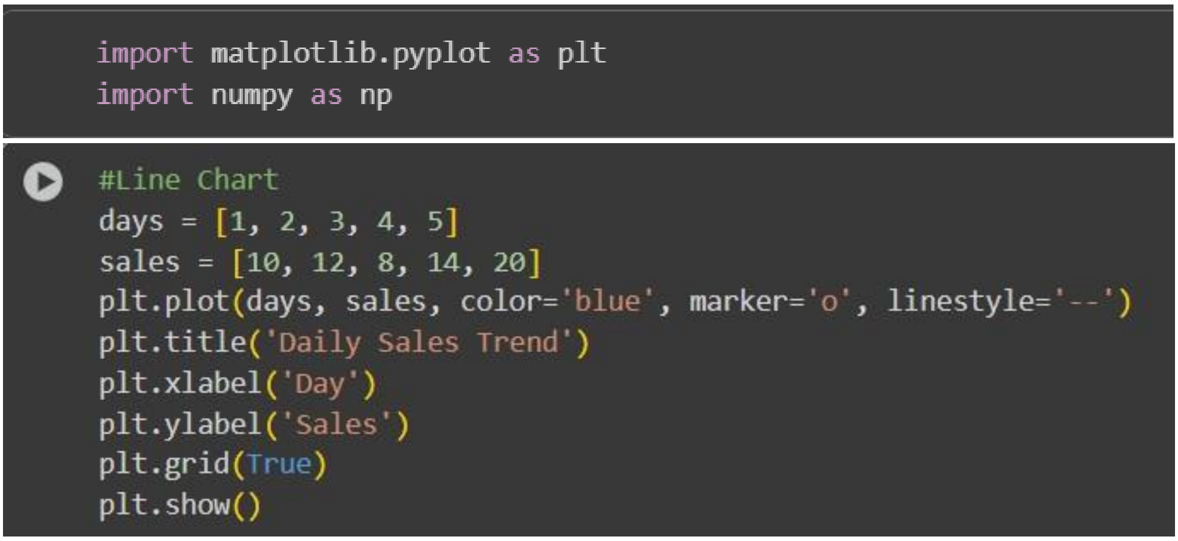


```
Mean:
 Math_Score       86.0
Science_Score    86.6
English_Score    83.4
dtype: float64
Median:
 Math_Score       88.0
Science_Score    88.0
English_Score    85.0
dtype: float64
Mode:
    Math_Score  Science_Score  English_Score
0          70             78             75
1          85             85             80
2          88             88             85
3          92             90             87
4          95             92             90
Range:
 Math_Score       25
Science_Score    14
English_Score    15
dtype: int64
Variance:
 Math_Score       94.5
Science_Score    29.8
English_Score    35.3
dtype: float64
Standard Deviation:
 Math_Score       9.721111
Science_Score    5.458938
English_Score    5.941380
dtype: float64
```

Ex 5 EDA-Data Visualization with Matplotlib

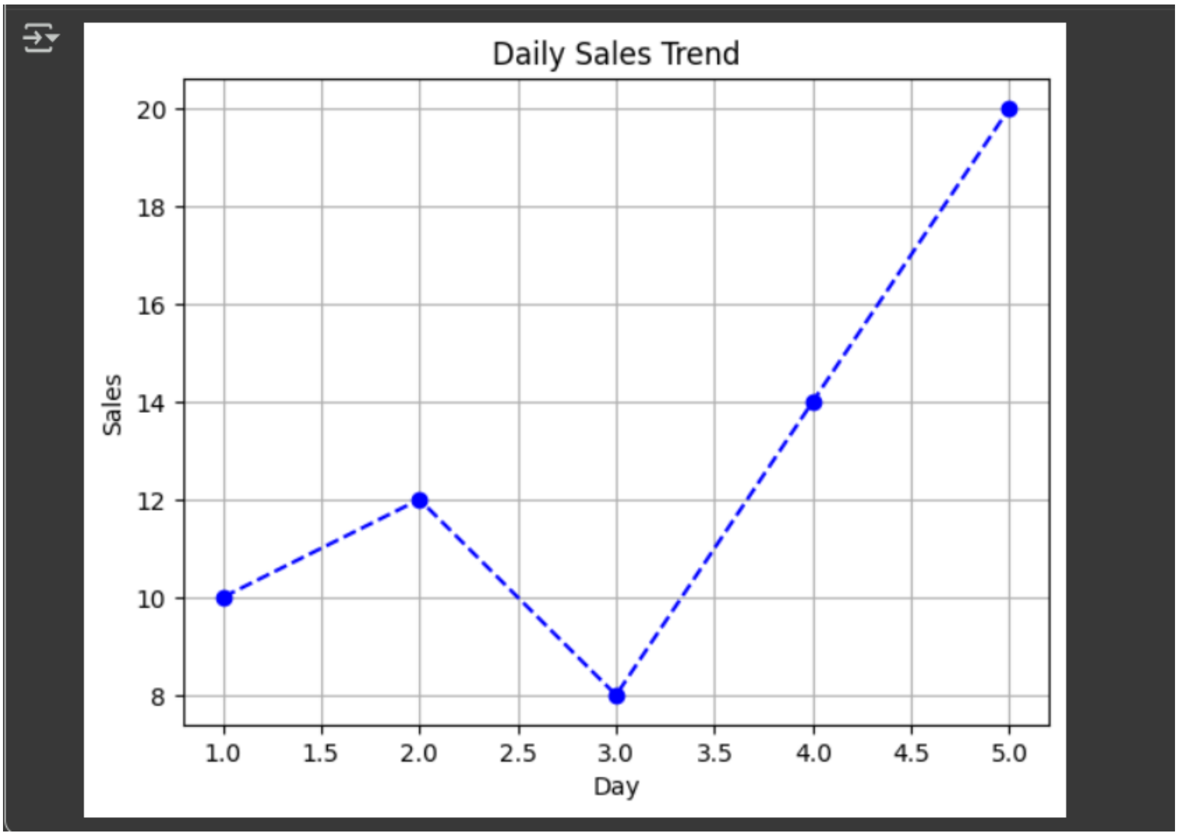
AIM:

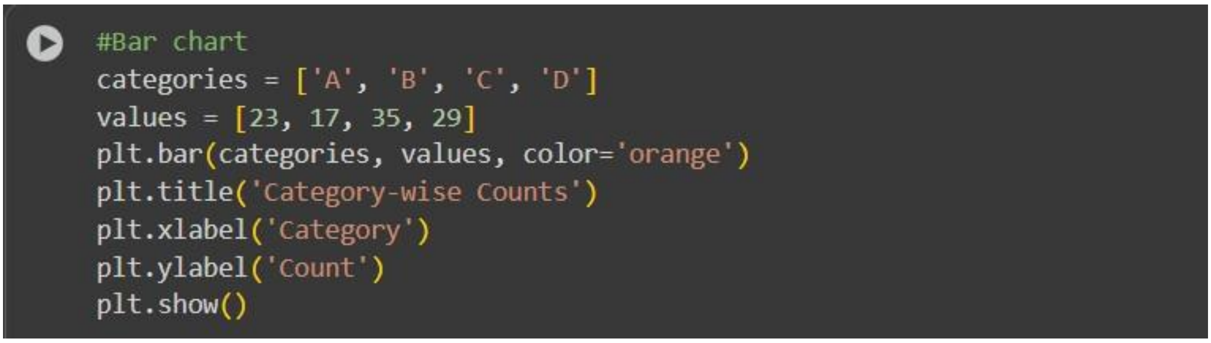
To visualize and understand datasets using basic plots - **line charts, bar charts** and **histograms** with the Matplotlib library in Python.



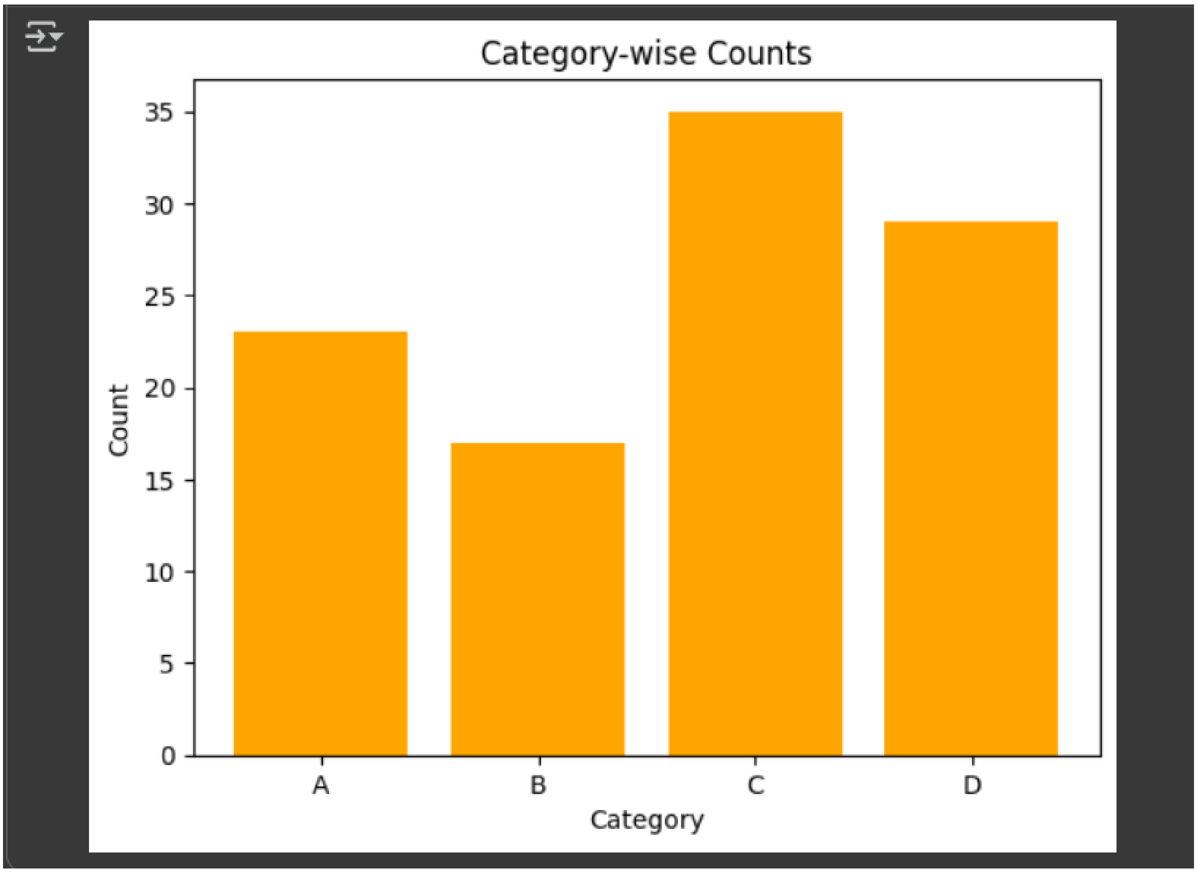
```
import matplotlib.pyplot as plt
import numpy as np
```

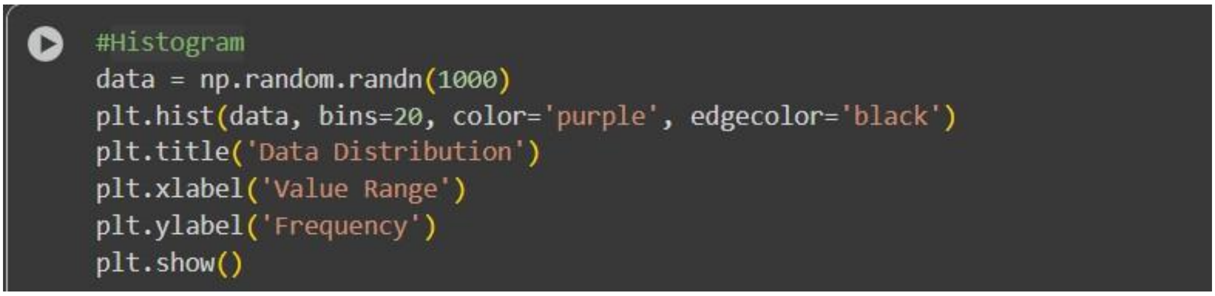
```
#Line Chart
days = [1, 2, 3, 4, 5]
sales = [10, 12, 8, 14, 20]
plt.plot(days, sales, color='blue', marker='o', linestyle='--')
plt.title('Daily Sales Trend')
plt.xlabel('Day')
plt.ylabel('Sales')
plt.grid(True)
plt.show()
```



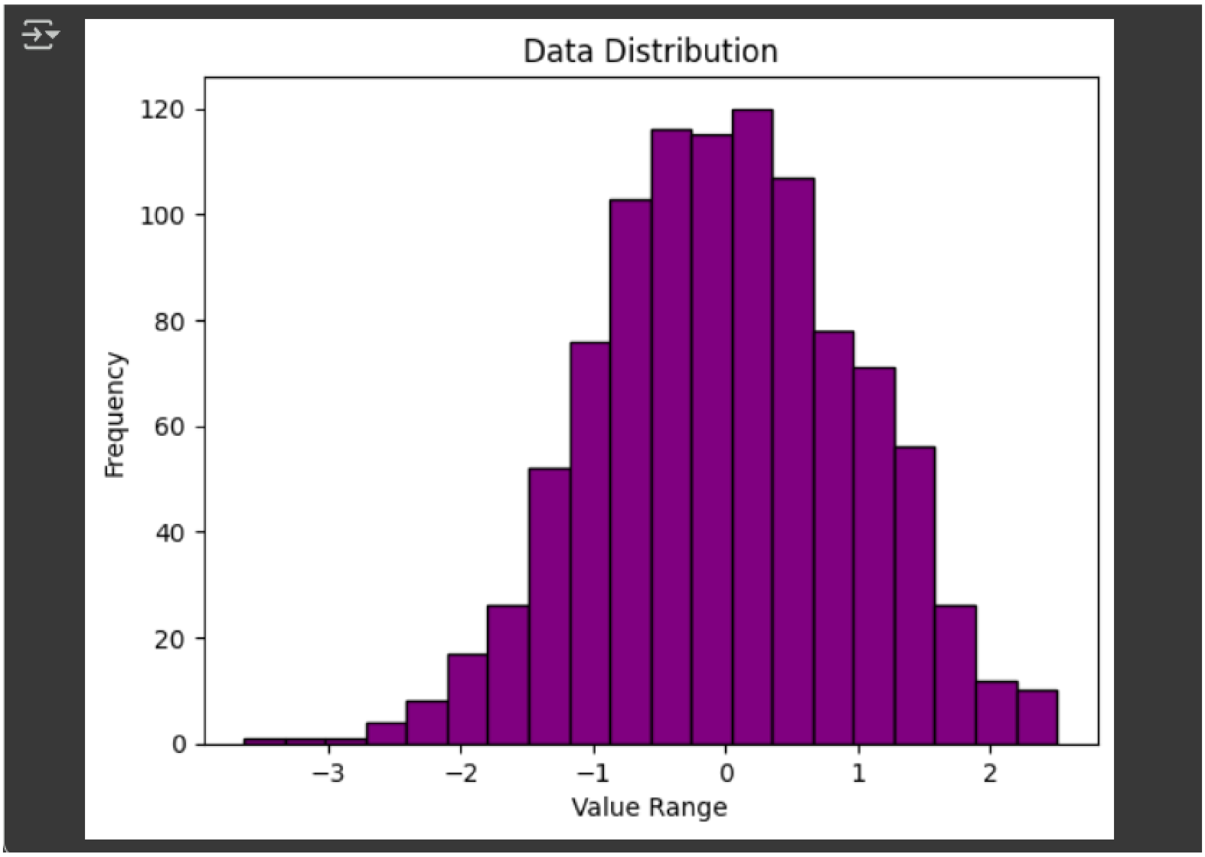


```
#Bar chart
categories = ['A', 'B', 'C', 'D']
values = [23, 17, 35, 29]
plt.bar(categories, values, color='orange')
plt.title('Category-wise Counts')
plt.xlabel('Category')
plt.ylabel('Count')
plt.show()
```





```
#Histogram
data = np.random.randn(1000)
plt.hist(data, bins=20, color='purple', edgecolor='black')
plt.title('Data Distribution')
plt.xlabel('Value Range')
plt.ylabel('Frequency')
plt.show()
```



Ex 6 Data Visualization Using PowerBi

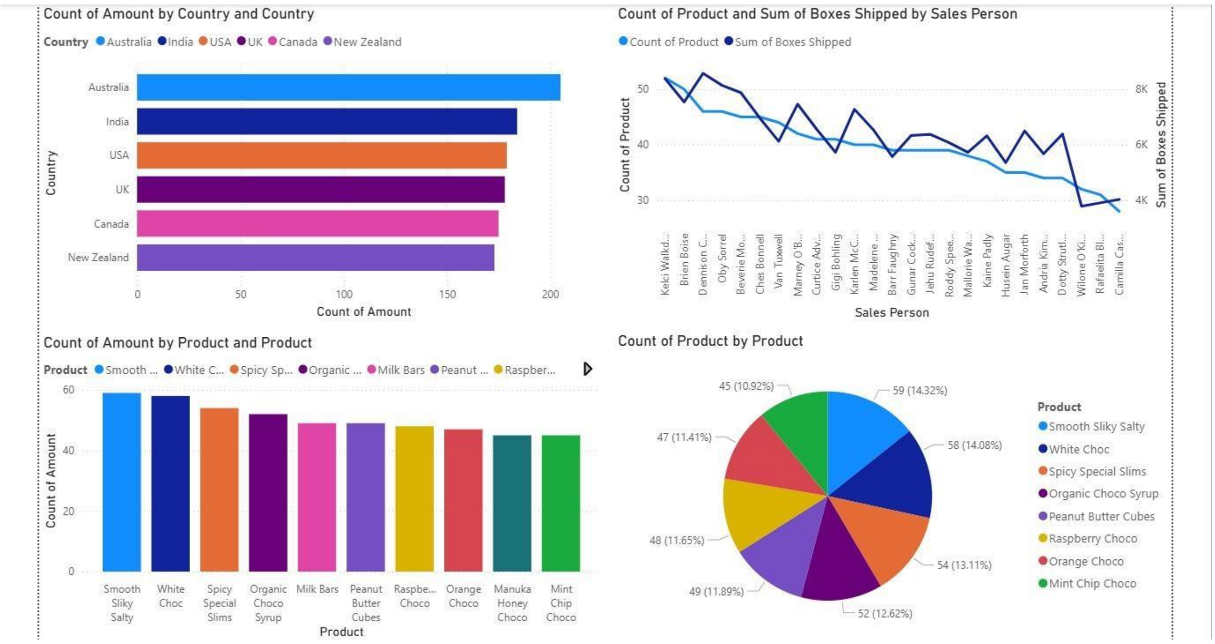
AIM:

To visualize and analyze data using Microsoft Power BI by connecting to various data sources, creating basic charts, and building an interactive dashboard.

PROCEDURE:

1. Open Power BI Desktop and explore the interface (Report, Data, and Model views).
2. Connect to Data Sources such as Excel, CSV, or SQL databases using the Get Data option.
3. Load the Data into Power BI and verify the tables in the Fields pane.
4. Create Visualizations like bar, line, and pie charts by dragging fields onto the canvas.
5. Add Calculated Columns and Measures using DAX.
6. Design a Dashboard by arranging visuals and adding slicers or filters for interactivity.
7. Save and Publish the report if needed.

DASHBOARD:



RESULT:

A Power BI dashboard was successfully created by connecting to data sources and visualizing key insights through charts and calculated metrics.

Ex 7 Data Visualization Using Tableau

AIM:

To understand data visualization using Tableau by connecting to different data sources, creating visualizations, adding calculated fields, and building interactive dashboards and stories.

PROCEDURE:

- Open Tableau Desktop and explore its interface (Data Pane, Toolbar, Worksheet, Dashboard, and Story tabs).
- Connect to Data Sources such as Excel, CSV, or SQL databases using the Connect option on the start page.
- Load the Dataset and view data in the Data Source tab.
- Create Visualizations on a new worksheet:

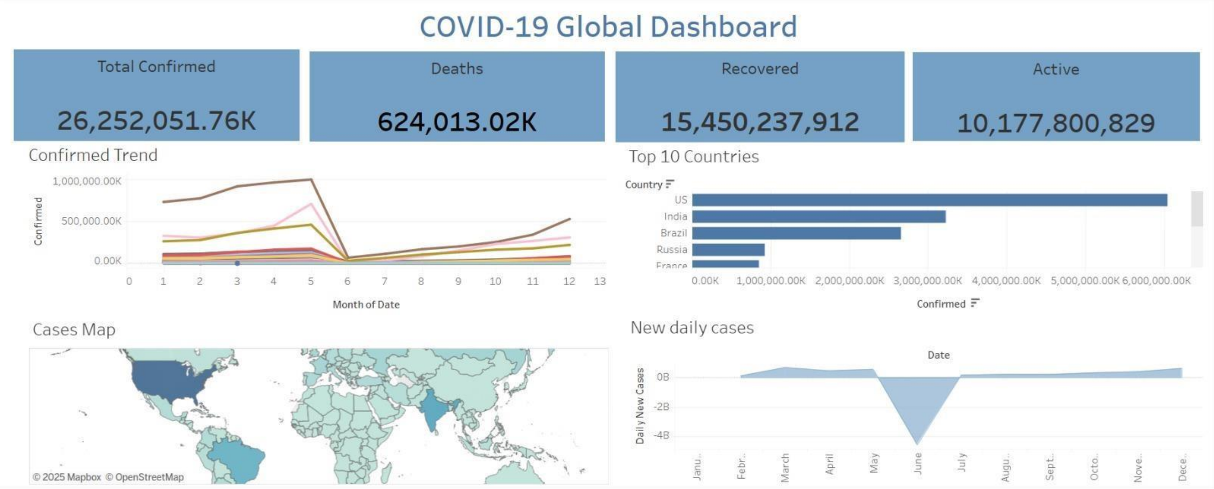
Bar Chart: Compare categories (e.g., sales by region).

Line Chart: Show trends over time.

Pie Chart: Display percentage distribution.

- Add Calculated Fields using formulas (e.g., Profit = [Sales] - [Cost]).
- Build a Dashboard by combining multiple charts for interactive analysis.
- Create a Story by arranging dashboards and worksheets to present insights sequentially.
- Save and Export the project.

DASHBOARD:



RESULT:

A Tableau dashboard and story were successfully created using multiple data sources, calculated fields, and visualizations, providing clear and interactive insights into the dataset.

MINI PROJECT

AIM:

To perform data visualization and analysis on the World Tourism Economy Dataset using Python, Power BI, and Tableau, in order to explore global tourism patterns, understand economic contributions, and uncover insights on tourism's impact on GDP and employment across regions and years.

ALGORITHM:

A. Python Visualization (Matplotlib & Seaborn)

1. Import Libraries
   Import required libraries — pandas, matplotlib.pyplot, and seaborn.
2. Load Dataset
   Load tourism_economy_dataset.csv using pandas.read_csv().
3. Data Preprocessing
   - Check for null or missing values.
   - Convert data types if needed (e.g., Year to integer).
   - Clean or filter data for relevant analysis.
4. Exploratory Data Analysis (EDA)
   - Display info and statistics using .info() and .describe().
   - Identify relationships between tourism receipts, GDP contribution, and employment.
5. Visualization using Matplotlib & Seaborn
   - Line Chart → Tourism Receipts over Years.
   - Bar Chart → Top 10 Countries by Receipts.
   - Scatter Plot → Receipts vs GDP Contribution.
   - Heatmap → Correlation between numerical variables.

Analyze charts to understand tourism growth trends, economic impact, and regional differences.

B. Power BI Visualization (Power BI Online)

Algorithm:

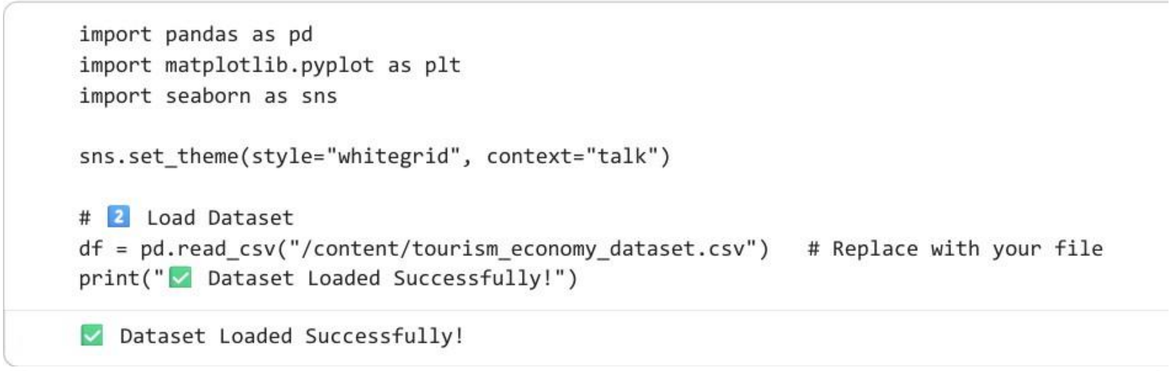
1. **Import Dataset**
   Upload tourism_economy_dataset.csv into Power BI Service (Online Workspace).
2. **Data Preparation**
   - Review column data types (Year, Region, GDP, etc.).
   - Rename fields and format numeric columns for readability.
3. **Report Editor Setup**
   Open **Report Editor View** to create visuals.
4. **Create Visualizations**
   - **Line Chart:** Tourism Receipts vs Year (Region as Legend).
   - **Bar Chart:** Top 10 Countries by Receipts.
   - **Scatter Chart:** GDP Contribution vs Tourism Receipts.
   - **Pie Chart:** Regional share of GDP Contribution.
5. **Publish / Save**
   Save and publish the report in Power BI Online Workspace for cloud access.

C. Tableau Visualization (Tableau Public / Online)

Algorithm:

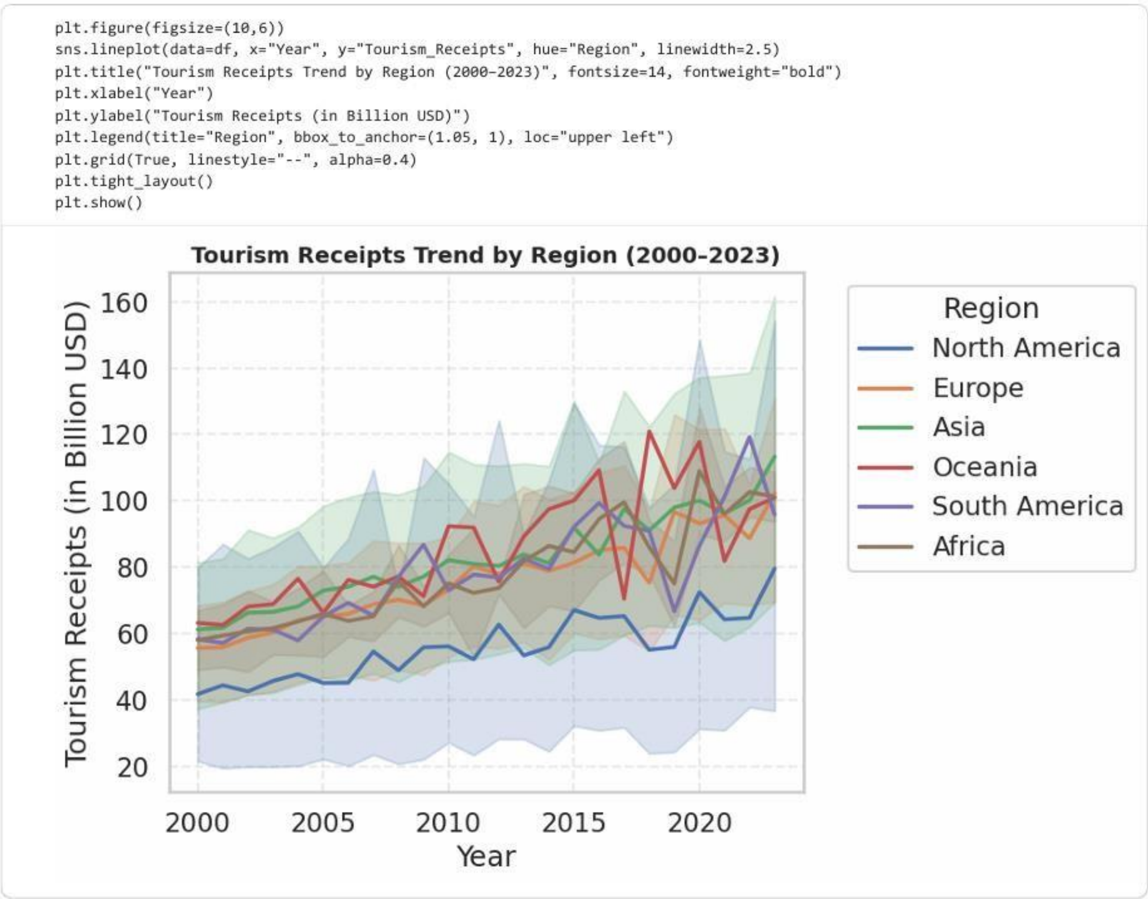
1. **Load Dataset**
   Open Tableau Public → Upload tourism_economy_dataset.csv.
2. **Data Connection**
   Verify field types (Country, Region = Dimension; GDP, Receipts = Measure).
3. **Create Individual Sheets for Visuals**
   - **Line Chart:** Receipts over Years (by Region).
   - **Bar Chart:** Top 10 Countries by Receipts.
   - **Scatter Plot:** Receipts vs GDP Contribution (Bubble size = Employment).
   - **Donut Chart:** GDP Contribution by Region.
4. **Design Dashboard Layout**
   - Place charts and filters in a balanced layout.
   - Apply consistent color palette by region.
5. **Publish**

   Save and publish to Tableau Public.

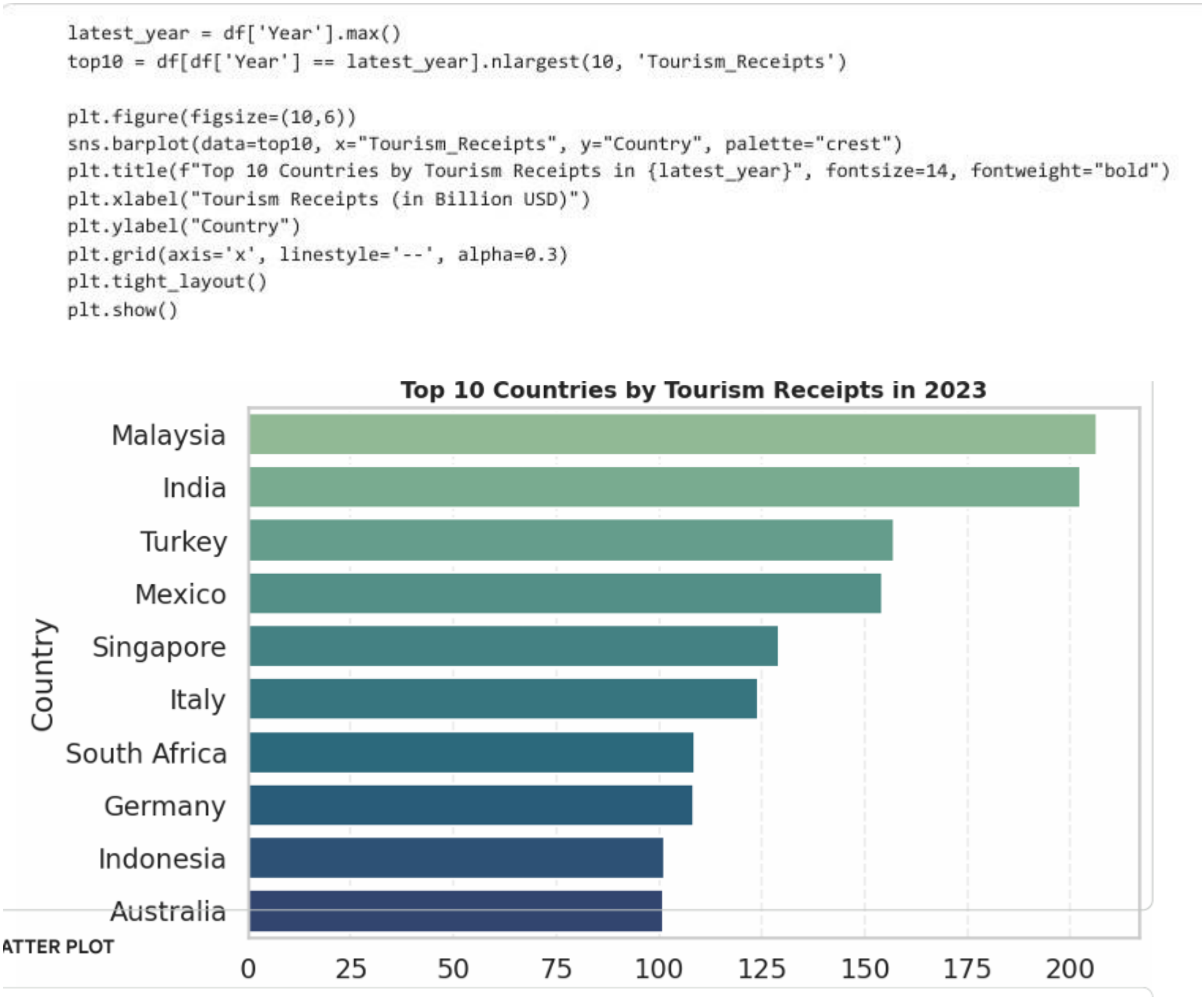


```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_theme(style="whitegrid", context="talk")

# 2️⃣ Load Dataset
df = pd.read_csv("/content/tourism_economy_dataset.csv")   # Replace with your file
print("✅ Dataset Loaded Successfully!")
```

```
✅ Dataset Loaded Successfully!
```

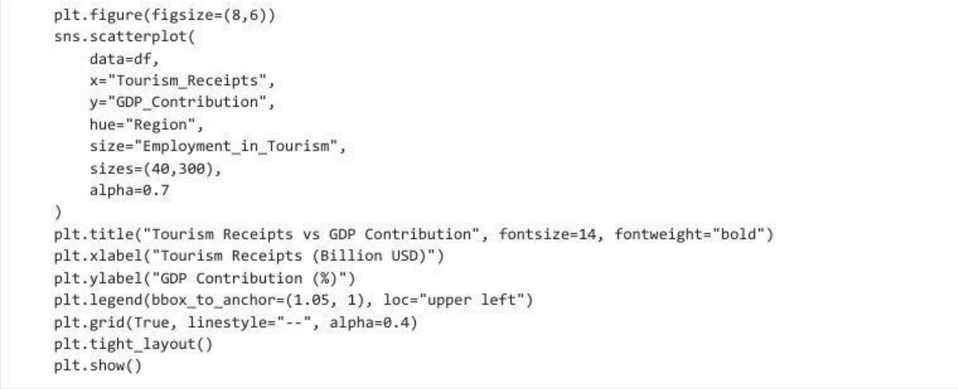


```
plt.figure(figsize=(10,6))
sns.lineplot(data=df, x="Year", y="Tourism_Receipts", hue="Region", linewidth=2.5)
plt.title("Tourism Receipts Trend by Region (2000-2023)", fontsize=14, fontweight="bold")
plt.xlabel("Year")
plt.ylabel("Tourism Receipts (in Billion USD)")
plt.legend(title="Region", bbox_to_anchor=(1.05, 1), loc="upper left")
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.show()
```

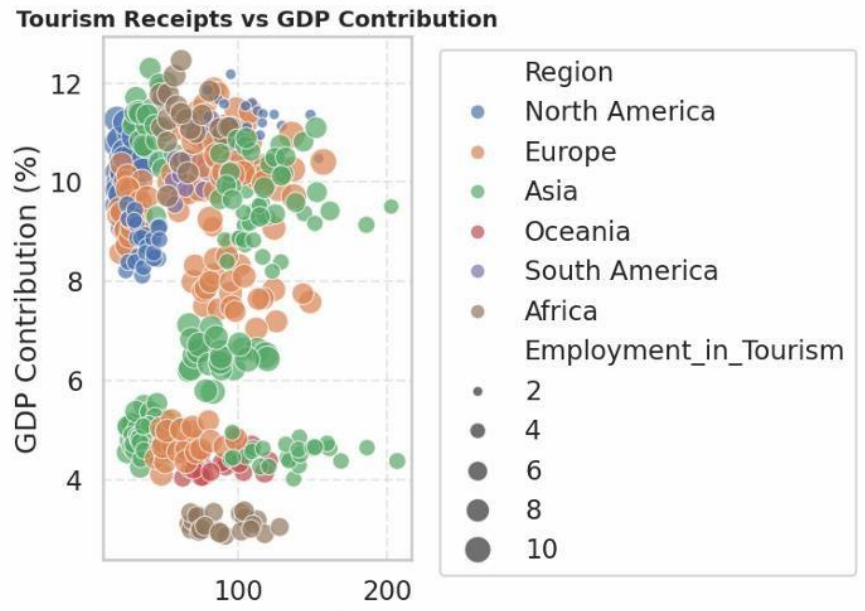


```
latest_year = df['Year'].max()
top10 = df[df['Year'] == latest_year].nlargest(10, 'Tourism_Receipts')

plt.figure(figsize=(10,6))
sns.barplot(data=top10, x="Tourism_Receipts", y="Country", palette="crest")
plt.title(f"Top 10 Countries by Tourism Receipts in {latest_year}", fontsize=14, fontweight="bold")
plt.xlabel("Tourism Receipts (in Billion USD)")
plt.ylabel("Country")
plt.grid(axis='x', linestyle='--', alpha=0.3)
plt.tight_layout()
plt.show()
```

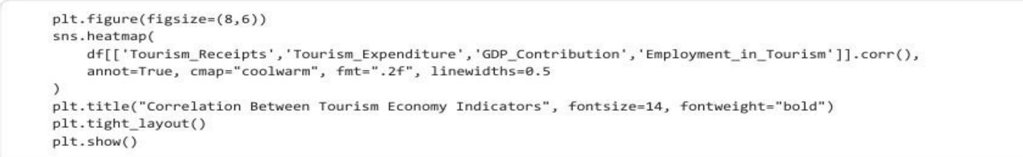
ATTER PLOT



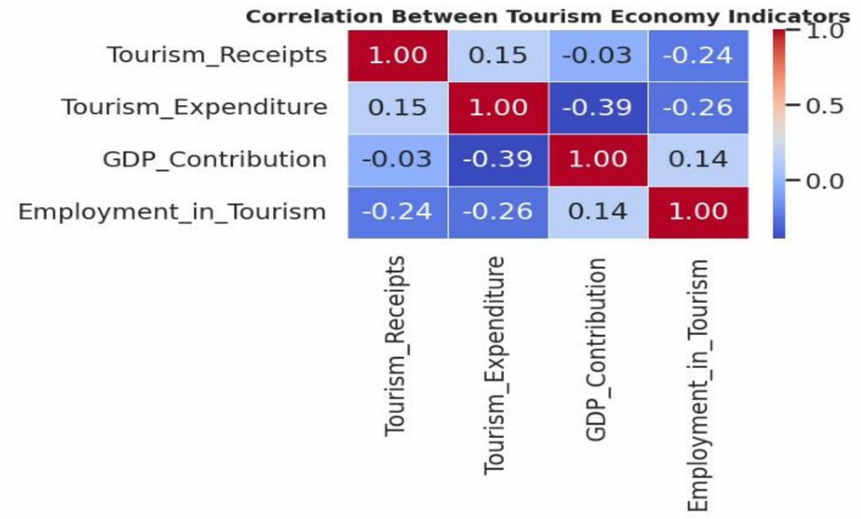
```
plt.figure(figsize=(8,6))
sns.scatterplot(
    data=df,
    x="Tourism_Receipts",
    y="GDP_Contribution",
    hue="Region",
    size="Employment_in_Tourism",
    sizes=(40,300),
    alpha=0.7
)
plt.title("Tourism Receipts vs GDP Contribution", fontsize=14, fontweight="bold")
plt.xlabel("Tourism Receipts (Billion USD)")
plt.ylabel("GDP Contribution (%)")
plt.legend(bbox_to_anchor=(1.05, 1), loc="upper left")
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.show()
```



HEATMAP



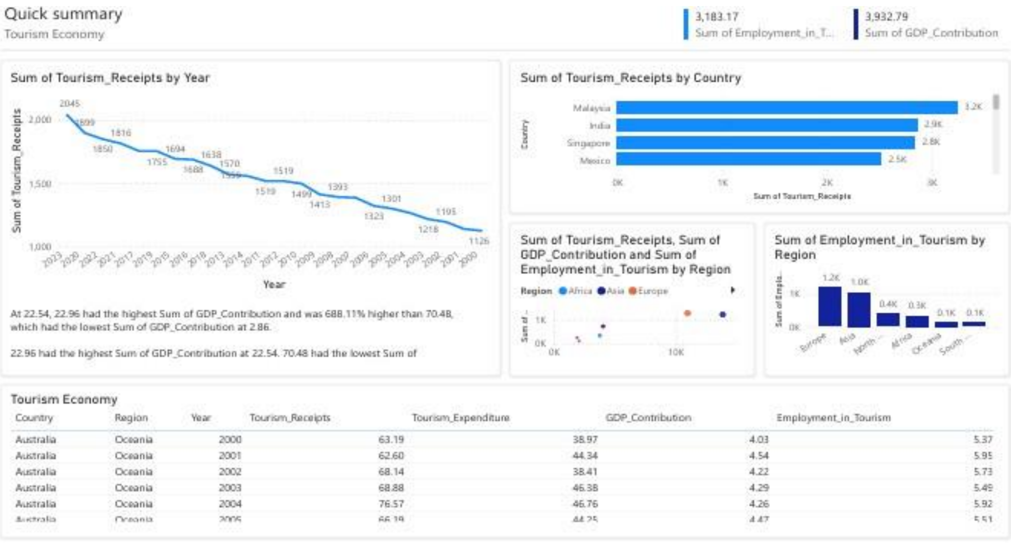
```
plt.figure(figsize=(8,6))
sns.heatmap(
    df[['Tourism_Receipts','Tourism_Expenditure','GDP_Contribution','Employment_in_Tourism']].corr(),
    annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5
)
plt.title("Correlation Between Tourism Economy Indicators", fontsize=14, fontweight="bold")
plt.tight_layout()
plt.show()
```



	Tourism_Receipts	Tourism_Expenditure	GDP_Contribution	Employment_in_Tourism
Tourism_Receipts	1.00	0.15	-0.03	-0.24
Tourism_Expenditure	0.15	1.00	-0.39	-0.26
GDP_Contribution	-0.03	-0.39	1.00	0.14
Employment_in_Tourism	-0.24	-0.26	0.14	1.00

DASHBOARD

POWER BI



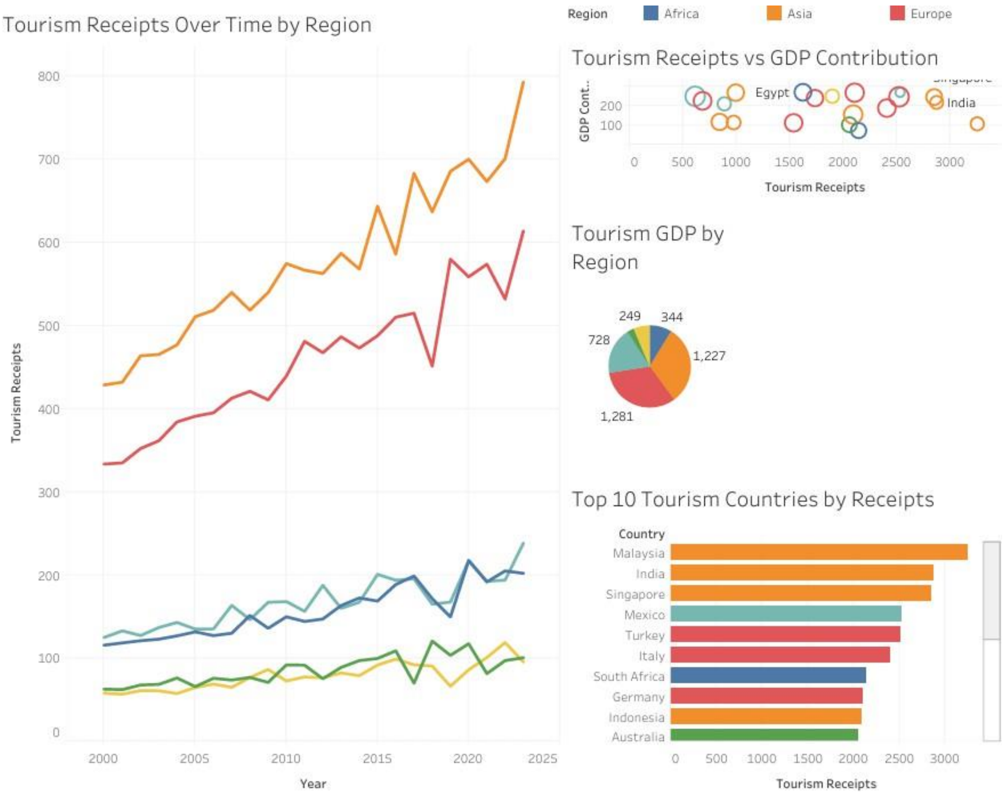
At 22.54, 22.96 had the highest Sum of GDP_Contribution and was 688.11% higher than 70.48, which had the lowest Sum of GDP_Contribution at 2.86.

22.96 had the highest Sum of GDP_Contribution at 22.54. 70.48 had the lowest Sum of

Tourism Economy

Country	Region	Year	Tourism_Receipts	Tourism_Expenditure	GDP_Contribution	Employment_in_Tourism
Australia	Oceania	2000	63.19	38.97	4.03	5.37
Australia	Oceania	2001	62.60	44.34	4.54	5.95
Australia	Oceania	2002	68.14	38.41	4.22	5.73
Australia	Oceania	2003	68.88	46.38	4.29	5.49
Australia	Oceania	2004	76.57	46.76	4.26	5.92
Australia	Oceania	2005	66.19	44.25	4.47	5.51

TABLEAU



RESULT:

Python, PowerBi ,Tableau visualizations has been executed successfully.