Sara Memon

AI Tasks

1. **Data Handling with NumPy & Pandas.**
   Task Description: Go through these concepts conceptually and theoretically.

NumPy is a powerful library for numerical computing in Python. It provides help for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them.

Pandas (Panel Data) is a high-level data manipulation tool built on NumPy. It provides two primary data structures:

i. Series (1D labeled array)
ii. Data Frame (2D labeled data structure, similar to Excel sheets or SQL tables)

Key Differences Between NumPy and Pandas		
Features	**NumPy**	**Pandas**
Structure	n-dimensional arrays	Series, Data Frame
Data Types	Homogeneous	Heterogeneous
Speed	Generally faster	Slightly slower
Use Case	Numerical computation	Data analysis & manipulation

- **Data Handling with Pandas and NumPy:**

NumPy and Pandas together create the foundation of data handling in Python, enabling fast, efficient, and readable data analysis workflows.

- We use NumPy for raw numerical operations and simulations such as:
  - Array creation
  - Broadcasting
  - Indexing and slicing
  - Random number generation
  - Linear algebra functions
- We use Pandas for high-level data analysis and manipulation such as:
  - Data filtering, grouping, sorting
  - Handling missing data

- Merging, joining, and reshaping data
- Reading/writing from CSV, Excel, SQL, etc.

2. Data Cleaning & Manipulation.
Task Description: also add examples regarding these topics on GitHub.

Data cleaning and manipulation involve preparing raw data for analysis by correcting errors, resolving formatting inconsistencies, addressing missing values, and transforming the data into a usable structure.

A. Data Cleaning Examples:

- **Detecting missing values:**

```
df.isnull()         # Shows True/False for missing values
df.isnull().sum()   # Count of missing values per column
```

- **Removing missing data:**

```
df.dropna()     # Drops rows with any missing value
df.dropna(axis=1)    # Drops columns with missing values
```

- **Filling missing data:**

```
df.fillna(0)    # Replaces missing values with 0
df['Age'].fillna(df['Age'].mean())   # Fill with mean of the column
```

- **Removing duplicates:**

```
df.duplicated()      # Identify duplicate rows
df.drop_duplicates()      # Remove them
```

- **Fixing datatypes:**

```
df['Date'] = pd.to_datetime(df['Date'])      # Convert string to datetime
df['Price'] = df['Price'].astype(float)      # Convert data type
```

- **Renaming columns:**

```
df.rename(columns={'oldName': 'newName'}, inplace=True)
```

- **String cleaning:**

```
df['Name'] = df['Name'].str.strip()          # Remove leading/trailing spaces
df['City'] = df['City'].str.lower()          # Convert to lowercase
df['Code'] = df['Code'].str.replace('-', '')  # Remove specific characters
```

B. Data Manipulation Examples:

- **Filtering Rows**

```
df[df['Score'] > 70]        # Rows where Score > 70
```

- **Sorting**

```
df.sort_values(by='Score', ascending=False)   # Sort by Score descending
```

- **Creating New Columns**

```
df['Total'] = df['Math'] + df['Science'] + df['English']
df['Passed'] = df['Total'] > 150
```

- **Grouping and Aggregating**

  df.groupby('Gender')['Score'].mean() # Mean score by gender

  df.groupby('Class').agg({'Math': 'mean', 'Science': 'sum'})
- **Merging & Joining DataFrames**

  pd.merge(df1, df2, on='ID', how='inner') # SQL-style join
- **Pivot Tables**

  df.pivot_table(index='Class', columns='Subject', values='Score', aggfunc='mean')

```
#Data Cleaning & Manipulation Example
import pandas as pd
data = {
    'Name': [' Sara ', 'Yousaf', 'Bilal', 'Mustafa', None],
    'Score': [85, 90, None, 90, 78],
    'Gender': ['F', 'M', 'M', 'M', 'F']
}
df = pd.DataFrame(data)

# Clean
df['Name'] = df['Name'].str.strip()
df['Score'].fillna(df['Score'].mean(), inplace=True)
df.dropna(inplace=True)

# Manipulate
df['Passed'] = df['Score'] > 80
df_grouped = df.groupby('Gender')['Score'].mean()
```

Task	Functions
Handle missing data	isnull(), dropna(), fillna()
Clean text	str.strip(), str.lower()
Change data types	astype(), to_datetime()
Filter/sort/aggregate	filter(), sort_values(), groupby()
Combine datasets	merge(), concat()

Sara Memon

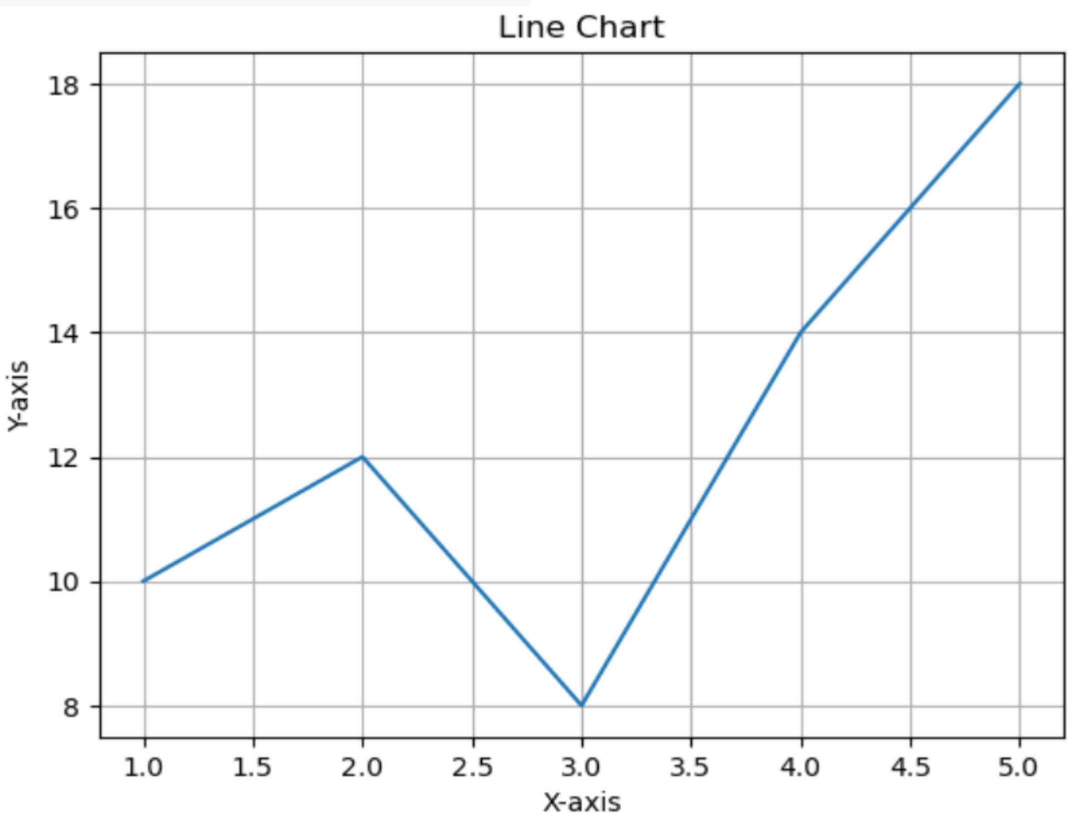
3. Visualization with Matplotlib, Seaborn, Plotly.

Task Description: go through these topics in details, also see code regarding these topics.

- **Data visualization helps us:**
  - Understand trends and patterns.
  - Communicate insights effectively.
  - Make data-driven decisions.

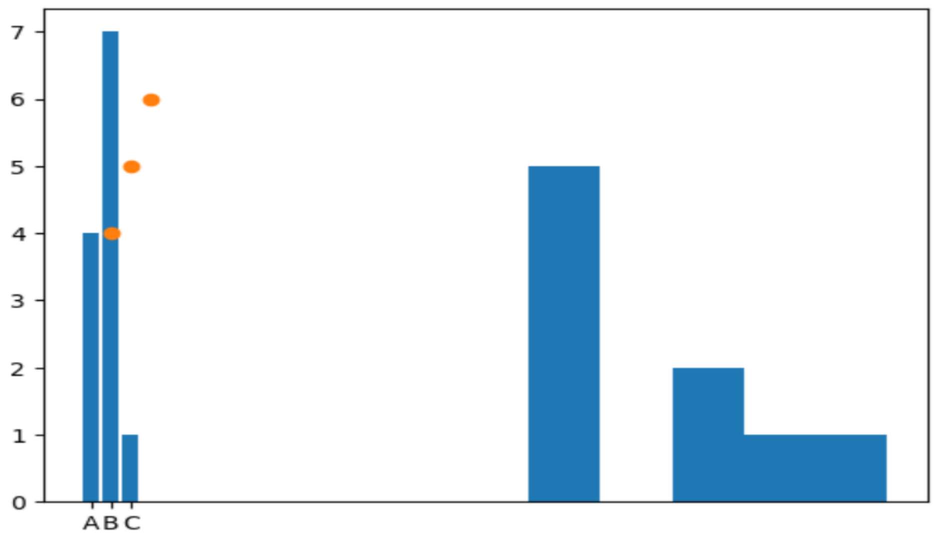
A. Matplotlib – is Low-level, powerful plotting library. Foundation for many other Python plotting libraries. Gives full control over every element of the plot.

```
import matplotlib.pyplot as plt

# Simple line plot
x = [1, 2, 3, 4, 5]
y = [10, 12, 8, 14, 18]

plt.plot(x, y)
plt.title("Line Chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid(True)
plt.show()
```



```
# Bar chart
categories = ['A', 'B', 'C']
values = [4, 7, 1]
plt.bar(categories, values)

# Histogram
plt.hist([22, 22, 23, 24, 25, 30, 30, 35, 40], bins=5)

# Scatter plot
plt.scatter([1, 2, 3], [4, 5, 6])
```

```
<matplotlib.collections.PathCollection at 0x137867bb250>
```

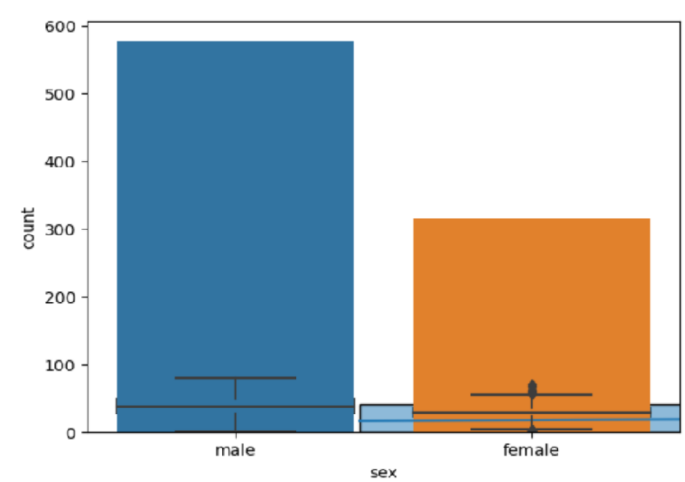


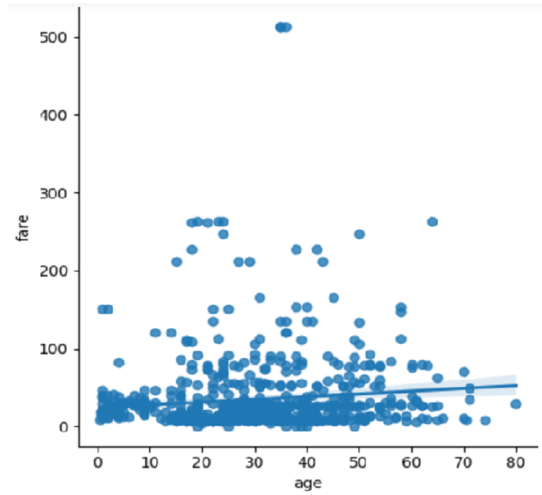
B. Seaborn – is built on Matplotlib, easier and more elegant. It is High-level interface for statistical visualizations. Seaborn automatically adds themes and statistical context. Seaborn works with Pandas data frames and supports complex visualizations such as heatmaps, pair plots, violin plots, etc.

```
import seaborn as sns
import pandas as pd

# Load built-in Titanic dataset
df = sns.load_dataset("titanic")

# Histogram with KDE
sns.histplot(df['age'], kde=True)

# Boxplot
sns.boxplot(x='class', y='age', data=df)

# Countplot (like bar plot for categories)
sns.countplot(x='sex', data=df)

# Scatter plot with linear regression
sns.lmplot(x='age', y='fare', data=df)
```

```
C:\Users\PMLS\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

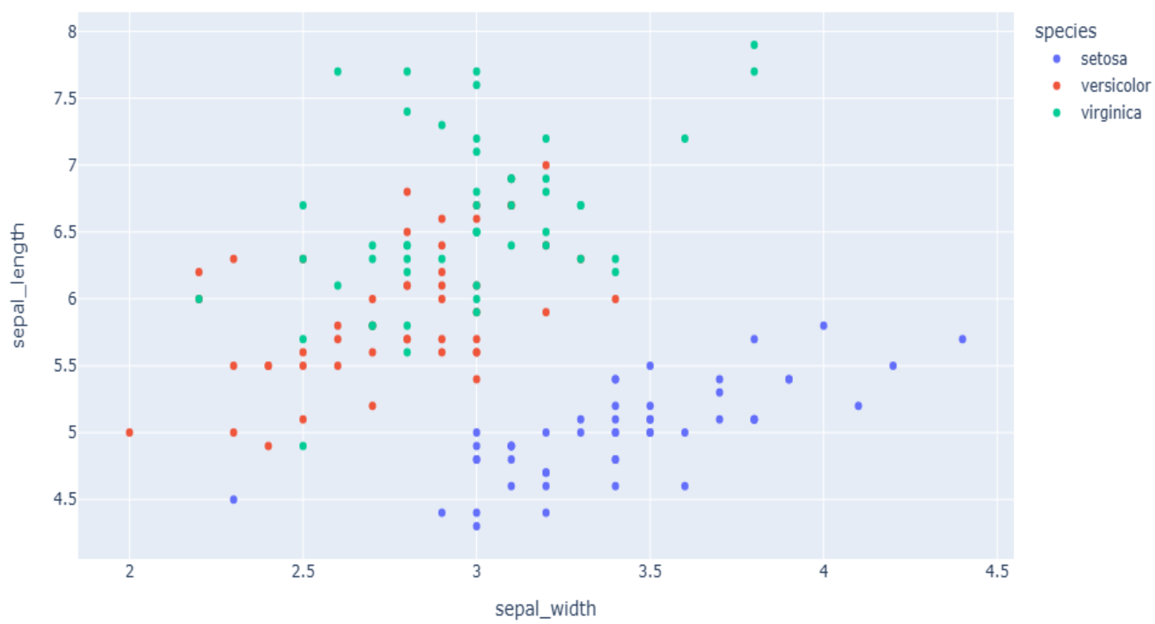
<seaborn.axisgrid.FacetGrid at 0x13792d8da10>





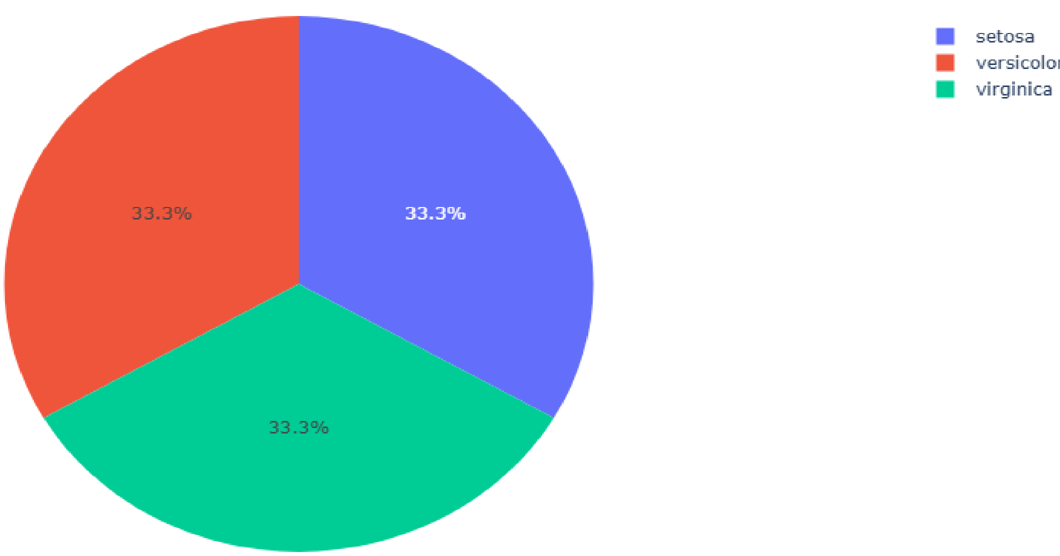
C. Plotly – is interactive, browser-based charts. It is ideal for dashboards, apps, and sharing online. And highly interactive (hover, zoom, tooltips).

```
#Basic Example
import plotly.express as px

# Load built-in data
df = px.data.iris()

# Interactive scatter plot
fig = px.scatter(df, x='sepal_width', y='sepal_length', color='species')
fig.show()
```

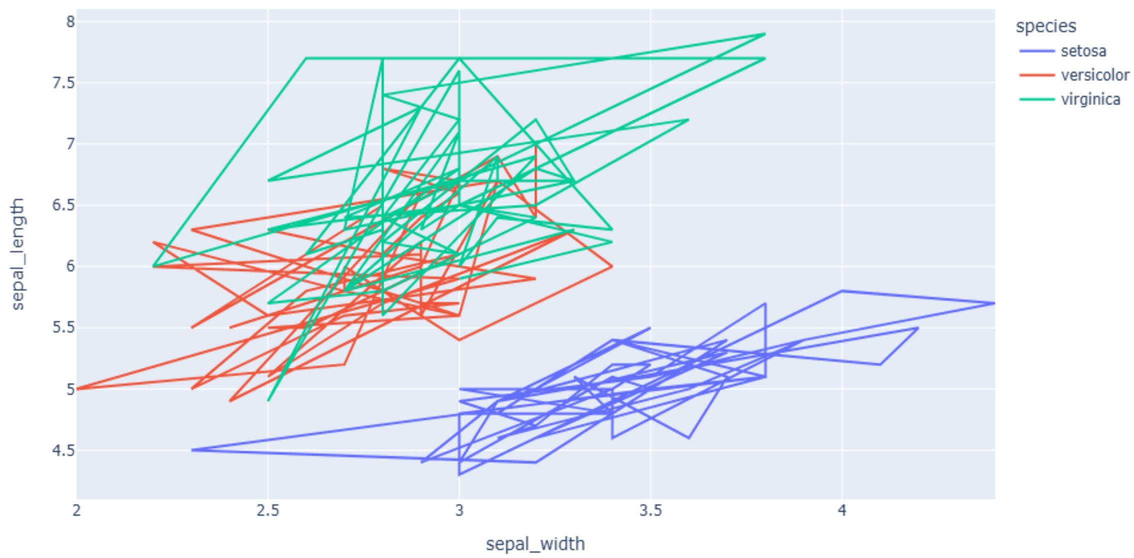


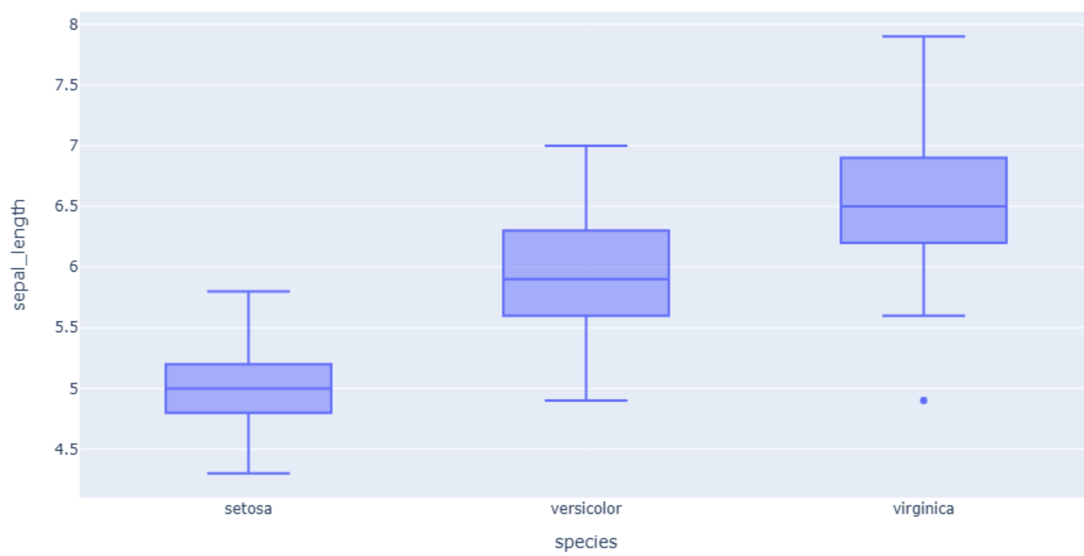
Sara Memon

```
# Common Plotly Charts:
# Bar plot
fig = px.bar(df, x='species', y='sepal_length')
fig.show()

# Pie chart
fig = px.pie(df, names='species')
fig.show()

# Line chart
fig = px.line(df, x='sepal_width', y='sepal_length', color='species')
fig.show()

# Box plot
fig = px.box(df, x='species', y='sepal_length')
fig.show()
```



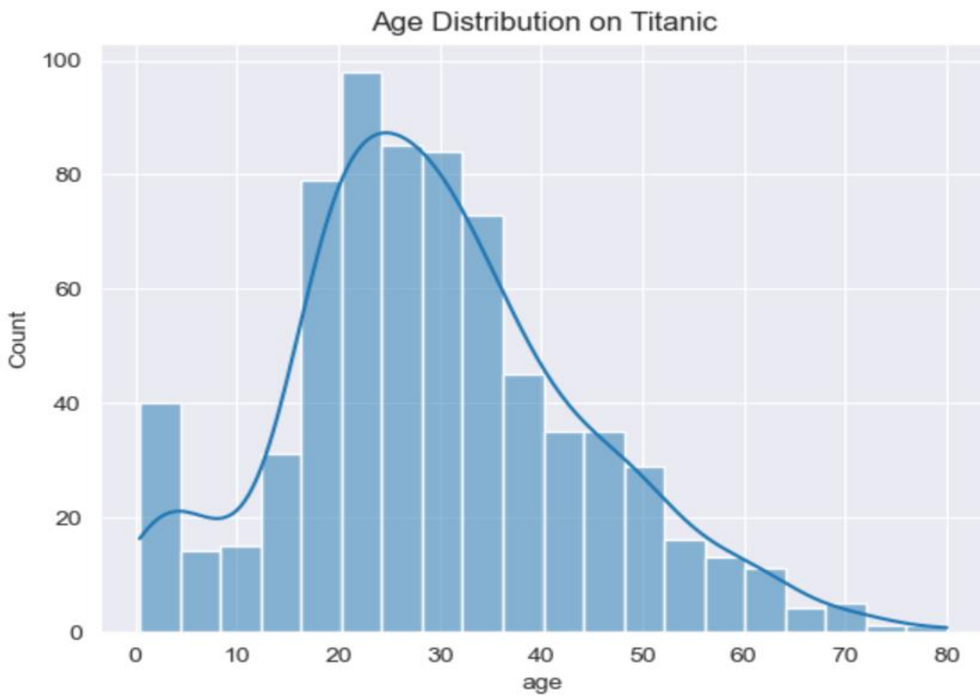
Sara Memon






```
# Matplot, Seaborn & Plotly Combine Example
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style("darkgrid")
sns.histplot(df['age'], kde=True)
plt.title("Age Distribution on Titanic")
plt.show()
```

Sara Memon



Feature	Matplotlib	Seaborn	Plotly
Level	Low-level	High-level wrapper	High-level & Interactive
Output	Static images	Static images	Interactive plots (HTML)
Style	Manual styling	Beautiful by default	Modern + responsive
Best For	Custom control	Statistical graphics	Dashboards, Web Apps