NUMPY

1) Create NumPy arrays from Python Data Structures, Intrinsic NumPy objects and Random Functions.

```
import numpy as np

# Create from list, NumPy function, and random
array1 = np.array([1, 2, 3]) array2
= np.ones((2, 2))
array3 = np.random.randint(0, 10, size=(2, 2))
print("List:",array1, "\nFunction:", array2, "\nRandom:",
array3)
```

2. Manipulation of NumPy arrays- Indexing, Slicing, Reshaping, Joining and Splitting.

```
arr = np.arange(1, 10).reshape(3, 3)
sliced = arr[0:2, 1:]      # Slicing reshaped
= arr.reshape(1, 9)  # Reshaping joined =
np.hstack((arr, arr))  # Joining split =
np.vsplit(arr, 3)  # Splitting
print("Sliced:\n", sliced, "\nReshaped:\n", reshaped,
"\nJoined:\n", joined, "\nSplit:\n", split)
```

3. Computation on NumPy arrays using Universal Functions and Mathematical methods.

```
arr = np.array([[1, 2], [3, 4]]) result
= {
   "sqrt": np.sqrt(arr),
   "sum": np.sum(arr),
   "mean": np.mean(arr),
   "comparison >2": arr > 2
}
print(result)
```

4. Import a CSV file and perform various Statistical and Comparison operations on rows/columns.

```
import numpy as np

# Load the CSV file, skipping the header and using only
numeric columns
data = np.genfromtxt('data.csv', delimiter=',',
skip_header=1, usecols=(1, 2, 3))

# Statistical operations
```

```
print("Subject-wise Mean:", np.mean(data, axis=0))
print("Student-wise Max:", np.max(data, axis=1))
print("Overall Std Deviation:", np.std(data))

# Comparison: find all scores greater than 90
high_scores = data > 90 print("Scores >
90:\n", high_scores)
```

data.csv

```
Name,Math,Science,English
Alice,85,90,78
Bob,75,88,82
Charlie,95,92,85
David,67,74,70
Eve,88,85,90
Frank,80,78,85
Grace,92,96,91
Hannah,70,68,75
Ivy,78,82,80
Jack,85,89,84
```

5. Load an image file and do crop and flip operation using NumPy indexing.

```
from PIL import Image
import numpy as np

img = Image.open('sample.jpg')
arr = np.array(img)
cropped = arr[100:300, 100:300]  # Crop to a region
flipped = np.flipud(cropped)    # Flip vertically
Image.fromarray(flipped).show()
```

PANDA

1. Create Pandas Series and DataFrame from various inputs.

```
import pandas as pd

# Series from list
```

```
series1 = pd.Series([10, 20, 30])
print("Series from list:\n", series1)

# Series from dictionary series2 =
pd.Series({'a': 1, 'b': 2, 'c': 3})
print("\nSeries from dictionary:\n", series2)

# DataFrame from dictionary
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}
df1 = pd.DataFrame(data)
print("\nDataFrame from dictionary:\n", df1)
```

sample.csv

```
Name,Age,City
Alice,25,New York
Bob,30,Los Angeles
Charlie,35,Chicago
David,28,Houston
Eva,32,San Francisco
Frank,40,Phoenix
Grace,22,Dallas
Hannah,27,Philadelphia
Ian,45,San Diego
```

```
Jane,29,San Antonio
```

2. Import any CSV file to Pandas DataFrame and perform the following:

```
import pandas as pd

# Read CSV file (make sure the file exists in the same
folder)
df = pd.read_csv('sample.csv')  # Change 'sample.csv' to
your actual filename
print("CSV File Loaded Successfully!\n") print(df)
```

3. Visualize the first and last 10 records

```
import pandas as pd

df = pd.read_csv('sample.csv')

print("First 10 rows:\n", df.head(10)) print("\nLast
10 rows:\n", df.tail(10))
```

4. Get the shape, index and column details

```
import pandas as pd

df = pd.read_csv('sample.csv')

print("Shape of DataFrame:",
df.shape)
print("Index of DataFrame:",
df.index)
print("Columns of DataFrame:",
df.columns)
```

5. Select/Delete the records(rows)/columns based on conditions.

```
import pandas as pd

df = pd.read_csv('sample.csv')

# Select rows where Age > 30 (ensure 'Age' column
exists) if 'Age' in df.columns:
    selected = df[df['Age'] > 30]
    print("Rows where Age > 30:\n", selected)

# Delete column 'City' if it exists df_without_city
= df.drop(columns=['City'], errors='ignore')
print("\nDataFrame after deleting 'City' column:\n",
df_without_city)

# Delete row with index 1
df_without_row = df.drop(index=1, errors='ignore')
print("\nDataFrame after deleting row with index 1:\n",
df_without_row)
```

6. Perform ranking and sorting operations.

```
import pandas as pd

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

```
# Rank based on 'Age' column if
'Age' in df.columns:
   df['Age_Rank'] = df['Age'].rank()
   print("DataFrame with Age Rank:\n", df)

# Sort by 'Age' in descending order if
'Age' in df.columns:
   sorted_df = df.sort_values(by='Age', ascending=False)
print("\nDataFrame sorted by Age (descending):\n",
sorted_df)
```

7. Do required statistical operations on the given columns.

```
 import pandas as
pd df =
pd.read_csv('sampl
e.csv')

# Statistical operations on 'Age' column if
'Age' in df.columns:
   print("Mean Age:", df['Age'].mean())
print("Median Age:", df['Age'].median())
print("Standard Deviation of Age:", df['Age'].std())

# Summary statistics for all numeric columns
```

```
print("\nSummary statistics:\n", df.describe())
```

MATPLOTLIB

Sample.csv

```
Name,Age,City
Alice,25,New York
Bob,30,Los Angeles
Charlie,35,Chicago
David,28,Houston
Eva,32,San Francisco
Frank,40,Phoenix
Grace,22,Dallas
Hannah,27,Philadelphia
Ian,45,San Diego
Jane,29,San Antonio
```

1. Import any CSV file to Pandas DataFrame and perform the following:

```
# Importing pandas library
import pandas as pd

# Load CSV file into DataFrame (Make sure to replace
'sample.csv' with your actual file path)
df = pd.read_csv('sample.csv')

# Display the first few rows of the dataframe
print("DataFrame loaded successfully:\n", df.head())
```

2. Handle missing data by detecting and dropping/ filling missing values.

```
# Importing pandas library
import pandas as pd

# Load the CSV file
df = pd.read_csv('sample.csv')

# Detect missing values missing_data
= df.isnull().sum()
print("Missing data per column:\n", missing_data)

# Drop rows with missing values df_dropped =
df.dropna() print("\nDataFrame after dropping rows
with missing values:\n", df_dropped)

# Fill missing values with a specific value (e.g., 0)
df_filled = df.fillna(0)
```

```
print("\nDataFrame after filling missing values with 0:\n", df_filled)

# Alternatively, fill missing values with the mean of the column
df_filled_mean = df.fillna(df['Age'].mean())
print("\nDataFrame after filling missing values with column mean:\n", df_filled_mean)
```

3. Transform data using apply() and map() method.

```
# Importing pandas library
import pandas as pd

# Load CSV file
df = pd.read_csv('sample.csv')

# Using apply() to increase 'Age' by 5 years
df['Age_plus_5'] = df['Age'].apply(lambda x: x + 5)
print("\nDataFrame with transformed 'Age' column:\n", df)
# Using map() to transform 'City' column to uppercase
df['City_upper'] = df['City'].map(lambda x: x.upper())
print("\nDataFrame with transformed 'City' column (uppercased):\n", df)
```

4. Detect and filter outliers.

```
# Importing pandas library
import pandas as pd

# Load CSV file
df = pd.read_csv('sample.csv')

# Detecting outliers in 'Age' column using IQR
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1

# Calculate the bounds for outliers lower_bound
= Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter out the outliers
df_no_outliers = df[(df['Age'] >= lower_bound) &
(df['Age'] <= upper_bound)]
print("\nDataFrame without outliers in 'Age':\n",
df_no_outliers)
```

5. Perform Vectorized String operations on Pandas Series.

```
# Importing pandas library
import pandas as pd
```

```
# Load CSV file df =
pd.read_csv('sample.csv') #
Convert all 'City' names to
uppercase using vectorized
string operation
df['City_upper'] = df['City'].str.upper()
print("\nDataFrame with vectorized string operation
(uppercased cities):\n", df)

# Extract the first letter of each city df['City_first_letter']
= df['City'].str[0]
print("\nDataFrame with first letter of each city:\n", df)
```

6. Visualize data using Line Plots, Bar Plots, Histograms, Density Plots and Scatter Plots

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

# Load CSV file
df = pd.read_csv('sample.csv')
```

```
# Line Plot (Visualize Age)
plt.plot(df['Age']) plt.title('Line
Plot of Age') plt.xlabel('Index')
plt.ylabel('Age')
plt.show()

# Bar Plot (Count of Cities)
df['City'].value_counts().plot(kind='bar', color='skyblue')
plt.title('Bar Plot of Cities') plt.xlabel('City')
plt.ylabel('Count')
plt.show()

# Histogram (Age Distribution) plt.hist(df['Age'],
bins=5, color='green',
edgecolor='black')
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency') plt.show()

# Density Plot (Using Histogram as a substitute for
density plot) plt.hist(df['Age'], bins=5, density=True,
color='orange', edgecolor='black', alpha=0.6)
plt.title('Density Plot of Age') plt.xlabel('Age')
plt.ylabel('Density') plt.show()

# Scatter Plot (Age vs Index)
plt.scatter(df.index, df['Age'], color='purple')
plt.title('Scatter Plot of Age vs Index') plt.xlabel('Index')
plt.ylabel('Age')
plt.show()
```

POWER BI

*HERE IS ONLY STEPS PROVIDE TO PERFORM IN POWER BI TOOL:

1. Import the legacy data from different sources such as (CSV, Text, Excel, SqlServer, Oracle etc.) and load in the target system.

Step 1: Open Power BI Desktop.

Step 2: In the 'Home' tab, click on 'Get Data'. This will open a window with various data source options.

Step 3: Choose the data source:

CSV/Text Files: Select ‘Text/CSV’, browse for your file, and click ‘Open’.

You will see a preview of your data.
Click 'Load' to load the data directly or 'Transform Data' if you want to clean and transform before loading.

Excel Files: Select ‘Excel Workbook’, choose your file, and select the sheet or table you want to import. Click 'Load' to import or 'Transform Data' to adjust the data before importing.

SQL Server: Select ‘SQL Server’, enter the server and database information.
Choose the authentication method and click ‘OK’ to connect. Select the tables or views to load.

Oracle: For Oracle, ensure the Oracle client is installed. Select ‘Oracle Database’, enter the necessary server details, and click ‘OK’. Select the data to load.

Step 4: Review the preview of the data, and if everything looks good, click ‘Load’ to import the data into Power BI or ‘Transform Data’ to clean and adjust it before loading.

2. Perform the different Transformation process to construct the database.

[11:06 PM, 5/14/2025] Shivam: Step 1: Click on 'Transform Data' to open the Power Query Editor.

Step 2: In Power Query Editor, you can perform the following transformations:

Remove Columns: Right-click on a column header and select 'Remove' to remove unnecessary columns.

Rename Columns: Double-click on a column header to rename it.

Change Data Types: Click the icon next to a column name to change the data type.

Filter Rows: Click the drop-down filter icon in column headers to filter specific values.

Remove Duplicates: Select a column, go to 'Remove Rows', and choose 'Remove Duplicates' to eliminate any duplicate rows.

Split Columns: Use 'Transform' > 'Split Column' to split data based on delimiter or number of characters.

Merge Queries: To combine data from different tables, go to 'Home' > 'Merge Queries', and select the tables and columns to merge.

Append Queries: Combine rows from multiple queries by choosing 'Append Queries'.

Step 3: After performing the necessary transformations, click 'Close & Apply' to apply the changes and load the data into Power BI.
[11:06 PM, 5/14/2025] Shivam:

3. Create calculated tables, calculated columns, and simple measures using Data Analysis Expressions

Step 1: Go to either ‘Data View’ or ‘Model View’.

Step 2: To create a calculated column:

Go to ‘Modeling’ > ‘New Column’.

Enter a DAX formula like TotalSales = Sales[Quantity] * Sales[UnitPrice].

Step 3: To create a calculated table:

Go to ‘Modeling’ > ‘New Table’.

Enter a DAX expression like HighValueCustomers = FILTER(Sales, Sales[Amount] > 10000).

Step 4: To create a measure:

Go to ‘Modeling’ > ‘New Measure’.

Example: TotalSales = SUM(Sales[Amount]).

4. Design Data Modeling on dateset.

Step 1: Go to ‘Model View’ to visualize your dataset.

Step 2: Create relationships by dragging fields from one table to related fields in another table (e.g., link 'CustomerID' from ‘Sales’ to ‘CustomerID’ in ‘Customers’).

Step 3: Check the cardinality of each relationship (e.g., One-to-Many or Many-to-One) and select the correct one. Set the cross-filter direction to 'Single' or 'Both' as needed.

Step 4: Ensure a proper Date Table exists. Mark it as a Date Table by selecting 'Modeling' > 'Mark as Date Table'.

5. Create measures with DAX expressions involving filter context manipulation

Step 1: Use DAX functions like CALCULATE() and FILTER() to modify the filter context in your measures.

Step 2: Example DAX formulas:

```
TotalWestSales = CALCULATE(SUM(Sales[Amount]),
Sales[Region] = "West").
```

```
SalesWithoutFilter = CALCULATE(SUM(Sales[Amount]),
ALL(Sales[Region])).
```

These formulas manipulate the filter context applied by slicers, visuals, or other measures.

6. Demonstrate DAX functions

power BI supports many useful DAX functions:

SUM(): Returns the sum of a column.

AVERAGE(): Returns the average of a column.

IF(): Creates conditional statements.

CALCULATE(): Changes the context of a calculation.

FILTER(): Filters a table based on a condition.

ALL(): Removes any filters on a table or column.

Example:

YTDSales = TOTALYTD(SUM(Sales[Amount]), 'Date'[Date]).

HighSales = IF(Sales[Amount] > 10000, "High", "Low").

7. Design and generate necessary reports based on the data

step 1: Create Visuals for Your Report

After importing and transforming your data, go to the Report View in Power BI. This is where you can start adding visualizations.

To add a visualization, click on the type of chart you need from the Visualizations pane (e.g., bar chart, pie chart, line chart, table).

For each visualization, drag relevant fields from the Fields pane to the Values, Axis, or Legend areas in the Visualizations pane, depending on the type of visualization.

Step 2: Format the Visualizations

To make your report more visually appealing, you can format the visuals. You can adjust properties like color, font size, labels, and axis scales under the Format tab.

Use the Title, Legend, and Data Labels formatting options to adjust how the visual looks.

Step 3: Add Multiple Visuals

You can add multiple visuals on the same report page. Resize and position them as needed to make your report easy to read.

Consider using a Grid Layout for organization. The Snap to Grid feature can help align your visuals perfectly.

Step 4: Adding Filters and Slicers

You can add slicers (filter controls) to let users filter data dynamically. For instance, if you want to filter the report by Region or Product Category, add a slicer and drag the relevant field to it.

Step 5: Preview the Report

You can view your report in Reading View to check how it will appear to users.

Test any slicers, tooltips, or interactions to ensure everything works as expected.

8. Demonstrate slicer in Power BI

step 1: Add a Slicer

Click on the Slicer icon in the Visualizations pane to add a slicer to your report.

Drag a field (e.g., Country, Region, Year) to the slicer's Values section.

Step 2: Customize the Slicer

Once the slicer is added, you can change its appearance and behavior:

In the Visualizations pane, click the Format button (paint roller icon).

You can change the slicer's orientation (vertical or horizontal), background color, and font size.

Step 3: Using Slicers for Data Filtering

You can use slicers to interactively filter data on the report. For example, if you add a Year slicer, users can select a particular year to update all visuals on the report.

Step 4: Multiple Slicers

You can add more than one slicer. For example, you could add a slicer for Region and another for Product Category. This allows users to drill down into the data by selecting multiple filters.

9. Creating a Power BI Dashboard

Step 1: Combine Visuals into a Dashboard

A dashboard in Power BI is a collection of multiple reports, graphs, and visualizations. The process involves grouping various charts, tables, and visuals together on the same page.

Step 2: Create Interactive Elements

Add interactivity to your dashboard by enabling drillthrough features or using bookmarks. For instance, clicking on a bar chart can reveal additional details about that category in another report page.

Step 3: Use Tiles and Custom Visuals

Power BI offers custom visuals in the AppSource (external marketplace). You can download and use these custom visuals for advanced dashboards, such as Funnel Charts, KPI Indicators, or Gauge Charts.

Step 4: Arrange Visuals

Make sure the visuals are arranged logically. Important metrics should be placed at the top or center. Group similar data together for better user experience.

Step 5: Dashboard Interactivity

Enable visual-level filters to let users dynamically filter data in the visuals. Use slicers, as discussed earlier, for even more interactive control

10. Publishing and Sharing Dashboard and reports Power BI

Step 1: Publish to Power BI Service

After creating your report in Power BI Desktop, click on the Publish button located in the Home tab of the ribbon.

Sign in to your Power BI account, and select a workspace where you want to publish the report.

Step 2: Check the Report in Power BI Service

After publishing, go to the Power BI Service (https://app.powerbi.com/), and navigate to the workspace where your report was published.

You can check the report's visualizations and also modify the layout if needed.

Step 3: Sharing the Report

In the Power BI Service, open the report and click on the Share button to send the report to specific users or generate a shareable link.

You can also embed reports into other web pages or applications using Power BI's embedding capabilities.

Step 4: Set Up Row-Level Security (RLS)

If your report needs security (e.g., different users should see different data), you can set up Row-Level Security

(RLS). This involves defining roles and filtering data based on the user’s credentials.

11. Demonstrate different charts Power BI supports various map visualizations:

Step 1: Bar Chart

Bar charts are useful for comparing different categories.

Add a Bar Chart by selecting it from the Visualizations pane.

Drag a categorical field (like Product Category) to the Axis and a numerical field (like Sales Amount) to the Values.

Step 2: Line Chart

Line charts are ideal for visualizing data trends over time.

Select a Line Chart visualization.

Drag a Date field to the Axis and a Measure (e.g., Sales Amount) to the Values.

Step 3: Pie Chart

Pie charts represent the proportion of each category.

Select the Pie Chart visualization.

Add a categorical field (e.g., Product Category) to the Legend and a measure to the Values.

Step 4: Area Chart

Area charts show the volume of change over time.

Drag Date to the Axis and a measure to the Values.

Step 5: TreeMap

TreeMaps are useful for hierarchical data, showing the proportions of categories within each level.

Add a TreeMap from the Visualizations pane.

Drag a hierarchical field (e.g., Region, Product Category) to the Group and a measure (e.g., Sales) to the Values.

12. Demonstrate at list 5 Maps

Step 1: Basic Map

Insert a Map from the Visualizations pane.

Drag a geographical field (e.g., City, Country, Latitude, Longitude) to the Location section, and a measure to the Size section.

Step 2: Filled Map

A Filled Map is used to color regions based on a measure.

Drag the geographical field (e.g., Country or State) to the Location field.

Drag a measure to the Values field to color regions based on that measure.

Step 3: Shape Map

Shape maps show data for specific geographic regions (e.g., regions, states).

In the Visualizations pane, select Shape Map.

You can use custom map shapes by uploading GeoJSON files.

Drag a region field (e.g., State) and a measure to this map.

13. Demonstrate Table and Matrix

Step 1: Table Visualization

Add a Table visual by selecting it from the Visualizations pane.

Drag fields into the Values area to populate the table.

Step 2: Matrix Visualization

The Matrix visual allows for hierarchical data representation.

Drag fields into the Rows and Columns to create hierarchies (e.g., Region → Product Category).

Use the Values section to display the measure.

14. Demonstrate total and Subtotal in matrix

step 1: Enable Subtotals

In the Matrix visual, enable subtotals for rows or columns by clicking on the Format pane, expanding the Subtotals section, and turning them on.

You can choose to show subtotals for all rows/columns or for selected levels.

Step 2: Grand Totals

You can display Grand Totals for rows and columns by enabling them in the Grand Total section under Format.

You can toggle the visibility of the grand total for both rows and columns.

15. Deploying the Power BI Dashboard and Report on Server

Step 1: Power BI Report Server

For deploying reports in an enterprise setting, use Power BI Report Server (which is part of Power BI Premium).

Go to Power BI Desktop and save your report as a .pbix file.

Upload this file to the Power BI Report Server via its web portal.

Step 2: Schedule Refreshes

In the Power BI Service or Power BI Report Server, you can schedule periodic data refreshes to ensure your reports stay up to date with the source data.

Step 3: User Permissions

Set up permissions in Power BI Service or Power BI Report Server to control who can view or edit the reports. You can grant access to individual users or user groups.
