---

1) Download, Install, and Explore Packages

```
# Install required libraries (run in terminal or Jupyter Notebook)
# !pip install numpy scipy jupyter statsmodels pandas

# Explore Features
import numpy as np
import scipy
import pandas as pd
import statsmodels.api as sm

print("Numpy Version:", np.__version__)
print("Scipy Version:", scipy.__version__)
print("Pandas Version:", pd.__version__)
print("Statsmodels Version:", sm.__version__)
```

Explanation:

- **Numpy**: Used for array manipulations and numerical computations.
- **Scipy**: Extends Numpy for scientific computations.
- **Pandas**: Provides tools for data manipulation and analysis.
- **Statsmodels**: Used for statistical modeling and hypothesis testing.

---

2) Working with Numpy Arrays

2a) Basic Array Characteristics

```
array = np.array([[1, 2, 3], [4, 5, 6]])
print("Array:\n", array)
print("Shape:", array.shape)
print("Size:", array.size)
print("Data Type:", array.dtype)
```

2b) Create Numpy Array Using List and Tuple

```
# From list
list_array = np.array([1, 2, 3, 4])
print("Array from List:", list_array)

# From tuple
tuple_array = np.array((5, 6, 7, 8))
print("Array from Tuple:", tuple_array)
```

2c) Basic Operations and Transpose

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

# Basic operations
print("Addition:\n", A + B)
print("Subtraction:\n", A - B)
```

```
print("Multiplication:\n", A * B)
print("Division:\n", A / B)

# Transpose
print("Transpose of A:\n", A.T)
```

2d) Sorting Operation

```
unsorted_array = np.array([3, 1, 4, 2])
print("Unsorted Array:", unsorted_array)
print("Sorted Array:", np.sort(unsorted_array))
```

Explanation:

- **Array Characteristics**: Shape, size, and data type give metadata about the array.
- **Basic Operations**: Arithmetic operations are element-wise.
- **Transpose**: Rows become columns and vice versa.
- **Sorting**: Sorts the elements in ascending order.

---

3) Working with Pandas DataFrame

3A) Simple Pandas DataFrame Operations

```
data = {'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [24, 27, 22]}
df = pd.DataFrame(data)
print("DataFrame:\n", df)

# Adding a new column
df['Score'] = [85, 90, 88]
print("Updated DataFrame:\n", df)
```

3B) Visualization and Web Retrieval

```
import matplotlib.pyplot as plt

# Create DataFrame
df = pd.DataFrame({'Category': ['A', 'B', 'C'], 'Values': [3, 7, 5]})

# Plot Bar Chart
df.plot(kind='bar', x='Category', y='Values', color='skyblue', title='Bar
Chart')
plt.show()
```

3C) Reading Data from CSV, Text, Excel, and Web

```
# CSV Example
csv_data = pd.read_csv('sample.csv')  # Replace with an actual file path
print("CSV Data:\n", csv_data)

# Text File
text_data = pd.read_table('sample.txt', sep='\t')  # Replace with file path
print("Text Data:\n", text_data)

# Excel File
```

```
excel_data = pd.read_excel('sample.xlsx')  # Replace with file path
print("Excel Data:\n", excel_data)
```

Explanation:

- **Pandas**: Handles tabular data efficiently.
- **Visualization**: Visualizes data for better understanding.
- **File Reading**: Supports various file formats.

---

4) Descriptive Analytics on Iris Dataset

```
from sklearn.datasets import load_iris
iris = load_iris()

# Create DataFrame
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target

# Descriptive Analytics
print("Head:\n", iris_df.head())
print("Summary Statistics:\n", iris_df.describe())
print("Target Value Counts:\n", iris_df['target'].value_counts())
```

Explanation:

- **Iris Dataset**: A classic dataset for beginners.
- **Descriptive Analytics**: Summarizes data with mean, count, etc.

4A: Reading Data from Text Files, Excel, and the Web

Reading Data from Text Files

```
python
Copy code
# Assume iris_data.txt contains tab-separated data
iris_text = pd.read_csv('iris_data.txt', sep='\t')  # Replace with actual
file path
print("First 5 Rows from Text File:\n", iris_text.head())
```

Reading Data from Excel

```
python
Copy code
# Assuming iris_data.xlsx contains the iris dataset
iris_excel = pd.read_excel('iris_data.xlsx')  # Replace with actual file
path
print("First 5 Rows from Excel File:\n", iris_excel.head())
```

Exploring Commands for Descriptive Analytics

```
python
Copy code
# Descriptive Statistics
print("Summary Statistics:\n", iris_text.describe())

# Checking for Missing Values
print("Missing Values:\n", iris_text.isnull().sum())

# Column-wise Analysis
print("Column Names:\n", iris_text.columns)
print("Data Types:\n", iris_text.dtypes)

# Correlation
print("Correlation Matrix:\n", iris_text.corr())
```

4B: Reading Data from the Web and Descriptive Analytics on Iris Dataset

Reading Data from the Web

```
python
Copy code
# Using an example URL for the Iris dataset in CSV format
iris_url = 'https://raw.githubusercontent.com/mwaskom/seaborn-
data/master/iris.csv'
iris_web = pd.read_csv(iris_url)
print("First 5 Rows from Web Data:\n", iris_web.head())
```

Exploring Commands for Descriptive Analytics

```
python
Copy code
# Shape and Info
print("Shape of Data:\n", iris_web.shape)
print("Info:\n", iris_web.info())

# Descriptive Statistics
print("Summary Statistics:\n", iris_web.describe())

# Grouping by Species
print("Mean of Each Feature by Species:\n",
iris_web.groupby('species').mean())

# Checking Unique Values in a Column
print("Unique Species:\n", iris_web['species'].unique())

# Visualizing Distribution (Optional)
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(iris_web, hue='species')
plt.show()
```

Explanation

1. **Text File Reading**: `pd.read_csv` with `sep='\t'` handles tab-separated files.

2. **Excel File Reading**: `pd.read_excel` reads Excel sheets.
3. **Web Reading**: `pd.read_csv` fetches data from URLs pointing to CSV files.
4. **Descriptive Analytics**:
    - `describe()`: Provides summary statistics like mean, min, max, etc.
    - `isnull()`: Checks for missing data.
    - `groupby`: Groups data for analysis.
    - `pairplot`: Visualizes relationships between features.