1. Data Wrangling I

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#dataset (can also use a local file path instead)
df=pd.read_csv(“iris.csv”)
df

# Define the column names as the dataset does not include headers
columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
# Step 4: Data Preprocessing
# Check for missing values in each column
print("Missing Values:\n", df.isnull().sum())

# Get summary statistics of the dataset
print("Summary Statistics:\n", df.describe())

# Check dimensions (rows, columns) of the dataset
print(f"Dataset Dimensions: {df.shape}")

# Check the data types of each column
print("Data Types:\n", df.dtypes)

# Step 5: Data Formatting and Data Normalization
# Convert categorical variable 'species' to numeric codes
df['species'] = df['species'].astype('category').cat.codes

# Check the updated DataFrame
print("Updated DataFrame:\n", df.head())
```

---

2. Data Wrangling II

Step 1: Scan for Missing Values and Inconsistencies

```
import pandas as pd
import numpy as np
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt
# Load dataset
df = pd.read_csv('student.csv')
# Check missing values
print("Missing Values:\n", df.isnull().sum())
# Fill missing numeric values with mean
df.fillna(df.select_dtypes(include=[np.number]).mean(), inplace=True)
```

Step 2: Scan for Outliers

```
# Visualize with boxplots
sns.boxplot(data=df.select_dtypes(include=[np.number]))
plt.title("Boxplot for Numeric Columns")
plt.xticks(rotation=90)
plt.show()
# Remove outliers using Z-score
z_scores = np.abs(stats.zscore(df.select_dtypes(include=[np.number])))
df = df[(z_scores < 3).all(axis=1)]
```

Step 3: Apply Data Transformation

```
# Apply log transformation on a score column (e.g., math_score)
if 'math_score' in df.columns:
    df['log_math_score'] = np.log(df['math_score'] + 1)
```

---

3. Descriptive Statistics - Measures of Central Tendency and Variability

Step 1: Summary Statistics Grouped by Categorical Variable

```
import pandas as pd

# Load the dataset
df = pd.read_csv('iris.csv')

# Grouped summary statistics
print(df.groupby('species').describe())
```

Step 2: Python Program for Statistical Details of Iris Dataset

```
# Filter data for each species
setosa = df[df['species'] == 'Iris-setosa']
versicolor = df[df['species'] == 'Iris-versicolor']
virginica = df[df['species'] == 'Iris-virginica']
```

```
# Print basic stats
print("Setosa Summary:\n", setosa.describe())
print("\nVersicolor Summary:\n", versicolor.describe())
print("\nVirginica Summary:\n", virginica.describe())
```

---

4. Data Analytics I

Linear Regression Model for Home Prices

You can use the **Boston Housing Dataset** from Kaggle for this task. Here's how to load it and perform linear regression using **Scikit-learn**:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load the dataset
df =
pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv
')

# Split data into features and target variable
X = df.drop('medv', axis=1)
y = df['medv']

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

---

5. Data Analytics II

Logistic Regression for Social Network Ads Classification

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score

# Load the dataset
df = pd.read_csv("Social_Network_Ads.csv")

# Preprocess data and split
X = df.drop('Purchased', axis=1)
y = df['Purchased']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Logistic Regression
model = LogisticRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluate the model
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print(f'Confusion Matrix:\n {conf_matrix}')
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
```

---

6. Data Analytics III

Simple Naïve Bayes Classification

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score

# Load the dataset
df = pd.read_csv("iris.csv")

# Preprocess and split the data
X = df.drop('species', axis=1)
y = df['species']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Naïve Bayes Model
model = GaussianNB()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluate the model
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')

print(f'Confusion Matrix:\n {conf_matrix}')
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
```

---

7. Text Analytics

Preprocessing and TF-IDF Calculation

```
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

# Sample text
document = "The quick brown fox jumped over the lazy dog."


# Tokenization
tokens = word_tokenize(document)

# Remove stopwords
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]

# Stemming
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]

# TF-IDF
vectorizer = TfidfVectorizer()
```

```
tfidf_matrix = vectorizer.fit_transform([document])

print(f"Tokens: {tokens}")
print(f"Filtered Tokens: {filtered_tokens}")
print(f"Stemmed Tokens: {stemmed_tokens}")
print(f"TF-IDF Matrix:\n {tfidf_matrix.toarray()}")
```

---

8. Data Visualization I

Titanic Data Distribution

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

# Load Titanic dataset
titanic = sns.load_dataset('titanic')
print(titanic.head())

# 1. Survival count by gender
sns.countplot(x='survived', hue='sex', data=titanic)
plt.title('Survival Count by Gender')
plt.xlabel('Survived (0 = No, 1 = Yes)')
plt.ylabel('Number of Passengers')
plt.legend(title='Gender')
plt.show()

# 2. Fare distribution
sns.histplot(data=titanic, x='fare', bins=30, kde=True, color='green')
plt.title('Fare Distribution')
plt.xlabel('Fare')
plt.ylabel('Number of Passengers')
plt.show()
```

---

9. Data Visualization II

Boxplot for Age Distribution by Gender and Survival

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

# Load Titanic dataset
```

```
titanic = pd.read_csv("titanic.csv")  # Make sure the correct filename is used

# Boxplot: Age distribution by gender and survival
sns.boxplot(x='sex', y='age', hue='survived', data=titanic)
plt.title('Age Distribution by Gender and Survival Status')
plt.xlabel('Gender')
plt.ylabel('Age')
plt.legend(title='Survived', labels=['No', 'Yes'])
plt.show()
```

---

10. Data Visualization III

Dataset Analysis and Visualizations for Iris Dataset

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt


# Load dataset

df = pd.read_csv("iris.csv")

df.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']


# Print feature types

print(df.dtypes)


# Histograms

plt.figure(figsize=(10, 6))

for i, col in enumerate(df.columns[:-1]):

    plt.subplot(2, 2, i+1)

    sns.histplot(df[col], kde=True)

    plt.title(f'{col} Distribution')
```

```
plt.tight_layout()
plt.show()

# Boxplots by species
fig, axes = plt.subplots(2, 2, figsize=(10, 6))
for ax, col in zip(axes.flat, df.columns[:-1]):
    sns.boxplot(x='species', y=col, data=df, ax=ax)
    ax.set_title(f'{col} by Species')
plt.tight_layout()
plt.show()
```