Practical 1:

1. Import all the required Python Libraries.
2. Locate an open source data from the web (e.g., https://www.kaggle.com). Provide a clear description of the data and its source (i.e., URL of the web site).
3. Load the Dataset into pandas dataframe.
4. **Data Preprocessing:**
    - Check for missing values in the data using pandas isnull()
    - describe() function to get some initial statistics
5. **Provide variable descriptions**. Types of variables etc. Check the dimensions of the data frame.
6. **Data Formatting and Data Normalization:**
    - Summarize the types of variables by checking the data types of the variables (i.e., character, numeric, integer, factor, and logical).
    - If variables are not in the correct data type, apply proper type conversions.
    - Turn categorical variables into quantitative variables in Python

```
# Importing the required Python libraries
import pandas as pd
import numpy as np

# Loading the dataset into a pandas DataFrame
df = pd.read_csv('Pr1.csv')  # Replace 'Pr1.csv' with the path to your dataset

# Displaying the DataFrame to get an overview of the data
df

# Generating descriptive statistics for the DataFrame
df.describe()

# Checking for missing values in the DataFrame
df.isnull()

# Summing the missing values for each column to get a clear count
df.isnull().sum()

# Providing information about the DataFrame, including data types and non-null
counts
df.info()

# Checking the dimensions of the DataFrame (number of rows and columns)
df.shape

# Summarizing the data types of the variables in the DataFrame
df.dtypes

# Data Preprocessing: Replacing NaN values with 0.0 (or you could choose a different
strategy)
df = df.replace(to_replace=np.nan, value=0.0)

# Displaying the DataFrame after replacing NaN values
df
```

```
# Converting categorical variable 'Gender' into quantitative variables (0 for 'F', 1 for 'M')
df['Gender'].replace(to_replace=['F', 'M'], value=[0, 1], inplace=True)

# Displaying the DataFrame after converting 'Gender'
df

# Grouping the DataFrame by the 'Division' column
df2 = df.groupby('Division')

# Getting the group of data where 'Division' is 'A'
df2.get_group('A')
```

Explanation of Comments

1. Import Libraries: Import necessary libraries for data manipulation and numerical operations.
2. Load Dataset: Load the dataset from a CSV file into a pandas DataFrame.
3. Display DataFrame: Show the DataFrame to understand the structure and contents of the data.
4. Descriptive Statistics: Generate and display descriptive statistics to get insights into the numerical data.
5. Check for Missing Values: Identify where missing values are present in the dataset.
6. Count Missing Values: Sum missing values to see how many are present in each column.
7. DataFrame Info: Provide a summary of the DataFrame, including data types and counts of non-null values.
8. Check Dimensions: Display the shape of the DataFrame to know the number of rows and columns.
9. Data Types Summary: Summarize the data types of the variables to understand what types of data are present.
10. Replace NaN Values: Handle missing data by replacing NaN values with 0.0 (or another method).
11. Convert Categorical to Quantitative: Convert the 'Gender' categorical variable into numerical values for analysis.
12. Group Data: Group the DataFrame by a specific column (e.g., 'Division') for further analysis.
13. Get Specific Group: Retrieve data for a specific category within the grouped DataFrame.

Practical 2:
Create an “Academic performance” dataset of students and perform the following operations using Python.
1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them.
2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.
3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons:
to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into

a normal distribution.
Reason and document your approach properly.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
from sklearn.preprocessing import MinMaxScaler

# Load dataset
df = pd.read_csv("Pr2.csv")

# Data overview and check for missing values
print(df)
print(df.describe())
print(df.info())
print(df.isnull().sum())

# Handle missing values in 'Discussion' column
avg_val = df["Discussion"].astype("float").mean()
df['Discussion'].replace(to_replace=np.nan, value=avg_val, inplace=True)
print(df.isnull().sum())

# Visualizations
sns.boxplot(x=df['AnnouncementsView'])
plt.show()
sns.regplot(x='Sno', y='AnnouncementsView', data=df)
plt.show()

# Outlier detection and replacement
z = np.abs(stats.zscore(df['AnnouncementsView']))
threshold = 3
print(np.where(z > threshold))
avg_val = int(df[df.index != 419]["AnnouncementsView"].mean())
df.at[419, 'AnnouncementsView'] = avg_val
sns.boxplot(x=df['AnnouncementsView'])
plt.show()

# MinMax Scaling
scaler = MinMaxScaler()
discussion_values = df['Discussion'].values.reshape(-1, 1)
df['Discussion_scaled'] = scaler.fit_transform(discussion_values)

# Plot original and scaled distributions
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(df['Discussion'], kde=True, color='blue')
plt.title('Original Distribution')
plt.subplot(1, 2, 2)
sns.histplot(df['Discussion_scaled'], kde=True, color='green')
plt.title('MinMaxScaled Distribution')
plt.tight_layout()
plt.show()
```

Practical 3:

Descriptive Statistics - Measures of Central Tendency and variability
Perform the following operations on any open source dataset (e.g., data.csv) 1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables
grouped by one of the qualitative (categorical) variable. For example, if your categorical variable is age groups and quantitative variable
is income, then provide summary statistics of income grouped by the age groups.
Create a list that contains a numeric value for each response
to the categorical variable.
2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of
'Iris-setosa', 'Iris-versicolor' and 'Iris-versicolor' of iris.csv dataset.
Provide the codes with outputs and explain everything that you do in this step.

```
import pandas as pd
import numpy as np
import statistics as st

df = pd.read_csv("Mall_Customers.csv")
print(df)
print(df.mean(numeric_only=True))
print(df['Age'].mean())
print(df.mean(numeric_only=True, axis=1))
print(df.median(numeric_only=True))
print(df.mode())
print(df.min())
print(df.max())
print(df.std(numeric_only=True))
print(df.groupby(['Genre'])['Age'].mean())
df.rename(columns={'Annual Income (k$)': 'Income'}, inplace=True)
print(df)
print(df.groupby(['Genre']).Income.mean())

df_iris = pd.read_csv("Iris.csv")
print(df_iris)
print('Iris-setosa')
setosa = df_iris['Species'] == 'Iris-setosa'
print(df_iris[setosa].describe())
print('\nIris-versicolor')
versicolor = df_iris['Species'] == 'Iris-versicolor'
print(df_iris[versicolor].describe())
print('\nIris-virginica')
virginica = df_iris['Species'] == 'Iris-virginica'
print(df_iris[virginica].describe())
```

Practical 7:

Text Analytics
1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization. 2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

```
sentence1 = "I will walk 500 miles and I would walk 500 more. Just to be the man who
walks a thousand miles to fall down at your door!"
sentence2 = "I played the play playfully as the players were playing in the play with
playfulness"

from nltk import word_tokenize
print('Tokenized words:', word_tokenize(sentence1))

from nltk import pos_tag
token = word_tokenize(sentence1) + word_tokenize(sentence2)
tagged = pos_tag(token)
print("Tagging Parts of Speech:", tagged)

from nltk.corpus import stopwords
stop_words = stopwords.words('english')
token = word_tokenize(sentence1)
cleaned_token = []
for word in token:
    if word not in stop_words:
        cleaned_token.append(word)
print('Unclean version:', token)
print('\nCleaned version:', cleaned_token)

from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(token) for token in cleaned_token]
print(stemmed_tokens)

from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(token) for token in cleaned_token]
print(lemmatized_tokens)

from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform([sentence1])
print(tfidf_matrix.toarray())
print("\n# FEATURED NAMES \n")
print(tfidf_vectorizer.get_feature_names_out())
```

Practical 8:
Data Visualization I
1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data.
2. Write a code to check how the price of the ticket (column name: 'fare') for each

passenger is distributed by plotting a histogram.

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

data = pd.read_csv('Titanic.csv')
data.describe()
data.isnull().sum()

data['Age'] = data['Age'].fillna(data['Age'].mean(numeric_only=True))
data['Cabin'] = data['Cabin'].fillna(data['Cabin'].mode()[0])
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])
data.isnull().sum()

sns.countplot(x='Survived', data=data)
sns.countplot(x='Pclass', data=data)
sns.countplot(x='Embarked', data=data)
sns.countplot(x='Sex', data=data)

sns.boxplot(data['Age'])
sns.boxplot(data['Fare'])
sns.boxplot(data['Pclass'])
sns.boxplot(data['SibSp'])

sns.catplot(x='Pclass', y='Age', data=data, kind='box')
sns.catplot(x='Pclass', y='Fare', data=data, kind='strip')
sns.catplot(x='Sex', y='Fare', data=data, kind='strip')
sns.catplot(x='Sex', y='Age', data=data, kind='strip')

sns.pairplot(data)
sns.scatterplot(x='Fare', y='Pclass', hue='Survived', data=data)
sns.scatterplot(x='Survived', y='Fare', data=data)

sns.histplot(data['Age'], kde=True)
sns.histplot(data['Fare'], kde=True)

sns.jointplot(x='Survived', y='Fare', kind='scatter', data=data)

sns.heatmap(data.corr(numeric_only=True), cmap="YlGnBu")
plt.title('Correlation')

sns.catplot(x='Pclass', y='Fare', data=data, kind='bar')
plt.hist(data['Fare'])
```

Practical 9:

Data Visualization II

1. Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for distribution of age with respect to each gender along with the information about whether they survived or not. (Column names : 'sex' and 'age')

2. Write observations on the inference from the above statistics.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('Titanic.csv')
data['Age'] = data['Age'].fillna(data['Age'].mean(numeric_only = True))
data['Cabin'] = data['Cabin'].fillna(data['Cabin'].mode()[0])
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])
sns.boxplot(x='Sex', y='Age', data=data, palette="Set2", hue="Survived")
```

Practical 10:

Data Visualization III

Download the Iris flower dataset or any other dataset into a DataFrame. Scan the dataset and give the inference as:

1. List down the features and their types (e.g., numeric, nominal) available in the dataset. 2. Create a histogram for each feature in the dataset to illustrate the feature distributions. 3. Create a boxplot for each feature in the dataset.

4. Compare distributions and identify outliers.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
df = pd.read_csv('Iris.csv')
df.isnull().sum()
df.info()
np.unique(df["Species"])
df.describe()

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 8))
sns.histplot(df['SepalLengthCm'], kde=True, color='blue', ax = axes[0,0])
axes[0,0].set_title('Sepal Length')
sns.histplot(df['SepalWidthCm'], kde=True, color='green', ax = axes[0,1])
axes[0,1].set_title('Sepal Width')
sns.histplot(df['PetalLengthCm'], kde=True, color='yellow', ax = axes[1,0])
axes[1,0].set_title('Petal Length')
sns.histplot(df['PetalWidthCm'], kde=True, color='red', ax = axes[1,1])
axes[1,1].set_title('Petal Width')
plt.tight_layout()
plt.show()

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 8))
sns.boxplot(df['SepalLengthCm'], color='blue', ax = axes[0,0])
axes[0,0].set_title('Sepal Length')
sns.boxplot(df['SepalWidthCm'], color='green', ax = axes[0,1])
axes[0,1].set_title('Sepal Width')
sns.boxplot(df['PetalLengthCm'], color='yellow', ax = axes[1,0])
axes[1,0].set_title('Petal Length')
```

```
sns.boxplot(df['PetalWidthCm'], color='red', ax = axes[1,1])
axes[1,1].set_title('Petal Width')
plt.tight_layout()
plt.show()

data_to_plot = [df[x] for x in df.columns[0:4]]
fig, axes = plt.subplots(1, figsize=(12,8))
bp = axes.boxplot(data_to_plot)
```