Practical-1

```
import numpy as np
import pandas as pd
data = np.array([['','Col1','Col2'], ['Row1',1,2],
['Row2',3,4]])
print(pd.DataFrame(data=data[1:,1:],
index = data [1:,0], columns=data[0,1:]))
# Take a 2D array as input to your DataFrame
my_2darray = np.array([[1, 2, 3], [4, 5, 6]])
print(pd.DataFrame(my_2darray))
# Take a dictionary as input to your DataFrame
my_dict = {1: ['1', '3'], 2: ['1', '2'], 3: ['2', '4']}
print(pd.DataFrame(my_dict))
# Take a DataFrame as input to your DataFrame
my_df = pd.DataFrame(data=[4,5,6,7], index=range(0,4), columns=['A'])
print(pd.DataFrame(my_df))
# Take a Series as input to your DataFrame
my_series = pd.Series({"United Kingdom":"London", "India":"New Delhi", "United
States":"Washington", "Belgium":"Brussels"})
print(pd.DataFrame(my_series))
df = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6]]))
# Use the `shape` property print(df.shape)
# Or use the `len()` function with the `index` property
print(len(df.index))
```

Practical-2: Implement Python Program for NumPy Arrays

```
import numpy as np
arr = np.array([[1, 2, 3],
[4, 2, 5]])
print("Array is of type: ", type(arr))
print("No. of dimensions: ", arr.ndim)
print("Shape of array: ", arr.shape)
print("Size of array: ", arr.size)
print("Array stores elements of type: ", arr.dtype)
```

Practical-2.1: More about NumPy Arrays and data frames

```
import numpy as np
import pandas as pd
data = np.array([['','Col1','Col2'], ['Row1',1,2],
['Row2',3,4]])
print(pd.DataFrame(data=data[1:,1:],
index = data[1:,0], columns=data[0,1:]))
my_2darray = np.array([[1, 2, 3], [4, 5, 6]])
print(pd.DataFrame(my_2darray))
my_dict = {1: ['1', '3'], 2: ['1', '2'], 3: ['2', '4']}
print(pd.DataFrame(my_dict))
my_df = pd.DataFrame(data=[4,5,6,7], index=range(0,4), columns=['A'])
print(pd.DataFrame(my_df))
my_series = pd.Series({"United Kingdom":"London", "India":"New Delhi",
"United States":"Washington", "Belgium":"Brussels"})
print(pd.DataFrame(my_series))
df = pd.DataFrame(np.array([[1, 2, 3], [4, 5, 6]]))
print(len(df.index))
```

Lab Manual 3

```
# Creating DataFrames (Multiple Methods)
import pandas as pd
data = {
'ID': [1, 2, 3, 4],
'Name': ['Alice', 'Bob', 'Charlie', 'David'],
'Score': [78, 85, 90, 88]
}
df = pd.DataFrame(data)
print(df)
data2 = [[101, 'Math', 85], [102, 'Science', 90]]
df2 = pd.DataFrame(data2, columns=['Roll', 'Subject', 'Marks'])
print(df2)
# Inspecting and Understanding Data
print(df.head())
print(df.tail())
print(df.shape)
print(df.info())
print(df.describe())
# Row and Column Operations
df['Bonus'] = 5
df['Final_Score'] = df['Score'] + df['Bonus']
print(df)
# Drop row and column
df.drop(3, axis=0, inplace=True)
df.drop('Bonus', axis=1, inplace=True)
print(df)
# Filtering and Conditional Selection
high_scores = df[df['Final_Score'] > 85]
print(high_scores)
# Multiple conditions
filtered = df[(df['Final_Score'] > 80) & (df['ID'] > 1)]
```

```
print(filtered)
# Sorting and Ranking
print(df.sort_values(by='Final_Score', ascending=False))
df['Rank'] = df['Final_Score'].rank(ascending=False)
print(df)
# Working with Missing Data
import numpy as np
df.loc[1, 'Final_Score'] = np.nan
print(df.isnull())
print(df.fillna(df['Final_Score'].mean()))
print(df.dropna())
# Renaming Columns and Index
df.rename(columns={'Final_Score': 'Total'}, inplace=True)
df.index = ['A', 'B', 'C']
print(df)
# String Operations
df['Name_Upper'] = df['Name'].str.upper()
df['Name_Length'] = df['Name'].str.len()
print(df)
#GroupBy Operations
data3 = {
'Dept': ['IT', 'IT', 'HR', 'HR'],
'Employee': ['A', 'B', 'C', 'D'],
'Salary': [50000, 60000, 45000, 48000] }
emp_df = pd.DataFrame(data3)
print(emp_df.groupby('Dept')['Salary'].mean())
# Applying Functions
df['Grade'] = df['Total'].apply(lambda x: 'A' if x >= 90 else 'B')
print(df)
# Exporting Data
df.to_csv('output.csv', index=False)
print("Data exported successfully")
```

Practical-4

write a program Reading data from text files, Excel and the web and exploring various commands for doing descriptive analytics on the Iris data set

```
import pandas as pd

import numpy as np

#Upload CSV File (Colab only)

from google.colab import files

files.upload()

# Read CSV File

 #Upload Excel File

files.upload()

# Read Excel File

df_excel = pd.read_excel("iris.xlsx")

df_excel.head()

# Load Data Directly from URL

url = "https://raw.githubusercontent.com/mwaskom/seaborn

data/master/iris.csv"

df_web = pd.read_csv(url)

df_web.head()

df_web.shape

df_web.describe()

df_web.mean(numeric_only=True)

df_web.median(numeric_only=True)

df_web.mode()

df_web.std(numeric_only=True)

df_web.var(numeric_only=True)

df_web.skew(numeric_only=True)

df_web.kurtosis(numeric_only=True)

df_web['species'].value_counts()

df_web.groupby('species').mean()

df_web.corr(numeric_only=True)

df_web.isnull().sum()

df_web[['sepal_length', 'petal_length']].head()
```

#P5

Pima dataset import

A. Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis.

```
import  pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score, confusion_matrix


url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFuncton', 'Age', 'Outcome']
df_pima = pd.read_csv(url)
df_pima = pd.read_csv(url, names=columns)
df_pima.head()
from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
df_uci = pd.DataFrame(diabetes.data,columns=diabetes.feature_names)
df_uci['target'] = diabetes.target
print(df_uci.head())


df_pima.describe()
df_pima.mean(numeric_only=True)
df_pima.median(numeric_only=True)


df_pima.mode()
df_pima.std(numeric_only=True)


df_pima.var(numeric_only=True)
for col in df_pima.columns[:-1]:
```

```
print(df_pima[col].value_counts())

df_pima.skew(numeric_only=True)

df_pima.kurtosis(numeric_only=True)

# Example for Pima
for col in df_pima.columns[:-1]: # exclude Outcome
  sns.histplot(df_pima[col], kde=True)
  plt.title(f'Pima: {col}')
  plt.show()
```

 # Bivariate Analysis – Pima - Correlation Heatmap

```
# Correlation heatmap - numeric Vs numeric
plt.figure(figsize=(10,6))
sns.heatmap(df_pima.corr(), annot=True,
cmap='coolwarm')
plt.title("Correlation between Variables")
plt.show()
# Loop through all columns except 'Outcome'
for col in df_pima.columns[:-1]:
  plt.figure(figsize=(6, 4))
  sns.boxplot(x='Outcome', y=col, data=df)
  plt.title(f'{col} vs Outcome (Pima)')
  plt.show()
```

```
# Bivariate Linear Regression
X = df_pima[['Glucose']]
y = df_pima['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
lin_model = LinearRegression()
lin_model.fit(X_train, y_train)

y_pred = lin_model.predict(X_test)

print("Coefficient:", lin_model.coef_[0])
print("Intercept:", lin_model.intercept_)
print("R² score:", r2_score(y_test, y_pred))

plt.scatter(X, y, alpha=0.5)
plt.plot(X, lin_model.predict(X), color='red')
plt.xlabel("Glucose")
plt.ylabel("Outcome")
plt.title("Linear Regression: Glucose vs Outcome")
plt.show()

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

y_pred = log_model.predict(X_test)

print("Coefficient:", log_model.coef_[0][0])
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
X_range = np.linspace(X.min(), X.max(), 100).reshape(-1, 1)
y_prob = log_model.predict_proba(X_range)[:, 1]

plt.scatter(X, y, alpha=0.4)
plt.plot(X_range, y_prob, color='red')
plt.xlabel("Glucose")
plt.ylabel("Probability of Diabetes")
plt.title("Logistic Regression: Glucose vs Outcome")
plt.show()

# MULTIPLE REGRESSION ANALYSIS
features = ['Glucose', 'BMI', 'Age', 'Pregnancies']
X = df_pima[features]
y = df_pima['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
multi_lin_model = LinearRegression()
multi_lin_model.fit(X_train, y_train)

y_pred = multi_lin_model.predict(X_test)

coefficients = pd.DataFrame(multi_lin_model.coef_, index=features,
columns=['Coefficient'])
print(coefficients)
print("R² score:", r2_score(y_test, y_pred))

coefficients.plot(kind='bar')
plt.title("Multiple Linear Regression Coefficients")
plt.ylabel("Coefficient Value Or features importance")
plt.show()
```

```
multi_log_model = LogisticRegression(max_iter=1000)

multi_log_model.fit(X_train, y_train)


y_pred = multi_log_model.predict(X_test)


coefficients = pd.DataFrame(multi_log_model.coef_[0], index=features,
columns=['Coefficient'])

print(coefficients)

print("Accuracy:", accuracy_score(y_test, y_pred))


coefficients.plot(kind='bar', color='orange')

plt.title("Multiple Logistic Regression Coefficients")

plt.ylabel("Coefficient Value")

plt.show()
```

UCI dataset

 # A. Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis.

#same as PIMA REPLACE df_pima with df_uci

#Distribution visualisation

```
for col in df_uci.columns[:-1]: # exclude Outcome
  sns.histplot(df_uci[col], kde=True)
  plt.title(f'UCI: {col}')
  plt.show()
```

Bivariate Analysis – UCI - Correlation Heatmap

 # Correlation heatmap - numeric Vs numeric

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

sns.heatmap(df_uci.corr(), annot=True,cmap='coolwarm')

plt.title("Correlation between Variables")

plt.show()
```

#Box Plots for UCI Diabetes Dataset

```
#Create target bins (e.g., Low, Medium, High)
df_uci['target_bin'] = pd.qcut(df_uci['target'], q=3, labels=['Low', 'Medium', 'High'])
for col in df_uci.columns[:-2]:  # exclude 'target' and 'target_bin'
  plt.figure(figsize=(6, 4))
  sns.boxplot(x='target_bin', y=col, data=df_uci)
  plt.title(f'{col} vs Target Bin (UCI)')
  plt.show()
```

Bivariate Linear Regression

```
X_bmi = df_uci[['bmi']]
y = df_uci['target']

X_train, X_test, y_train, y_test = train_test_split(
    X_bmi, y, test_size=0.2, random_state=42
)
model_lr = LinearRegression()
model_lr.fit(X_train, y_train)

y_pred = model_lr.predict(X_test)

plt.figure(figsize=(6,4))
plt.scatter(X_test, y_test, color="blue", label="Actual values")
plt.plot(X_test, y_pred, color="red", label="Regression line")
plt.xlabel("BMI")
plt.ylabel("Target")
plt.title("Linear Regression: BMI vs Target")
plt.legend()
plt.show()
```

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("MSE:", mse)
print("R² Score:", r2)

# MULTIPLE REGRESSION ANALYSIS
model_multi = LinearRegression()
model_multi.fit(X_train, y_train)

y_pred_multi = model_multi.predict(X_test)

mse_multi = mean_squared_error(y_test, y_pred_multi)
r2_multi = r2_score(y_test, y_pred_multi)

print("Multiple Regression MSE:", mse_multi)
print("Multiple Regression R²:", r2_multi)

plt.figure(figsize=(6,4))
plt.scatter(y_test, y_pred_multi)
plt.xlabel("Actual Target")
plt.ylabel("Predicted Target")
plt.title("Actual vs Predicted Values (Multiple Regression)")
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color="red")
plt.show()
```