Experiment No. 11
Program to perform Exploratory Data Analysis using Numpy and Pandas
Date of Performance:
Date of Submission:

Experiment No. 11

Title: Program to perform Exploratory Data Analysis using Numpy and Pandas

Aim: To study and implement Exploratory Data Analysis using Numpy and Pandas

Objective: To introduce Panda package

Theory:

What is Pandas?

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Why Use Pandas?

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.

:}

Data Science: is a branch of computer science where we study how to store, use and analyze data for deriving information from it.

What Can Pandas Do?

Pandas gives you answers about the data. Like:

- Is there a correlation between two or more columns?
- What is average value?
- Max value?
- Min value?

Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called *cleaning* the data.

Installation of Pandas

If you have Python and PIP already installed on a system, then installation of Pandas is very easy.

Install it using this command:

```
C:\Users\Your Name>pip install pandas
```

If this command fails, then use a python distribution that already has Pandas installed like, Anaconda, Spyder etc.

Import Pandas

Once Pandas is installed, import it in your applications by adding the import keyword:

```
import pandas
```

Now Pandas is imported and ready to use.

Example

```
import pandas

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}

myvar = pandas.DataFrame(mydataset)

print(myvar)
```

```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

Pandas as pd

Pandas is usually imported under the pd alias.

alias: In Python alias are an alternate name for referring to the same thing.

Create an alias with the as keyword while importing:

```
import pandas as pd
```

Now the Pandas package can be referred to as pd instead of pandas.

Example

```
import pandas as pd

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}

myvar = pd.DataFrame(mydataset)

print(myvar)
```

Checking Pandas Version

The version string is stored under __version__ attribute.

Example

```
import pandas as pd

print(pd.__version__)
```

What is a Series?

A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.

Example

Create a simple Pandas Series from a list:

```
import pandas as pd

a = [1, 7, 2]
```

```
myvar = pd.Series(a)

print(myvar)
```

Labels

If nothing else is specified, the values are labeled with their index number. First value has index 0, second value has index 1 etc.

This label can be used to access a specified value.

Example

Return the first value of the Series:

```
print(myvar[0])
```

```
1
```

Create Labels

With the index argument, you can name your own labels.

Example

Create your own labels:

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)
```

```
x    1
y    7
z    2
dtype: int64
```

When you have created labels, you can access an item by referring to the label.

Example

Return the value of "y":

```
print(myvar["y"])
```

Key/Value Objects as Series

You can also use a key/value object, like a dictionary, when creating a Series.

Example

Create a simple Pandas Series from a dictionary:

```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)

print(myvar)
```

To select only some of the items in the dictionary, use the index argument and specify only the items you want to include in the Series.

Example

Create a Series using only data from "day1" and "day2":

```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories, index = ["day1", "day2"])

print(myvar)
```

DataFrames

Data sets in Pandas are usually multi-dimensional tables, called DataFrames.

Series is like a column, a DataFrame is the whole table.

Example

Create a DataFrame from two Series:

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

myvar = pd.DataFrame(data)

print(myvar)
```

Read CSV Files

A simple way to store big data sets is to use CSV files (comma separated files).

CSV files contains plain text and is a well know format that can be read by everyone including Pandas.

In our examples we will be using a CSV file called 'data.csv'.

Example

Load the CSV into a DataFrame:

```
import pandas as pd

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

print(df.to_string())
```

Example:

Print the DataFrame without the to_string() method:

```
import pandas as pd

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

print(df)
```

max_rows

The number of rows returned is defined in Pandas option settings.

You can check your system's maximum rows with the pd.options.display.max_rows statement.

Example

Check the number of maximum returned rows:

```
import pandas as pd

print(pd.options.display.max_rows)
```

Example

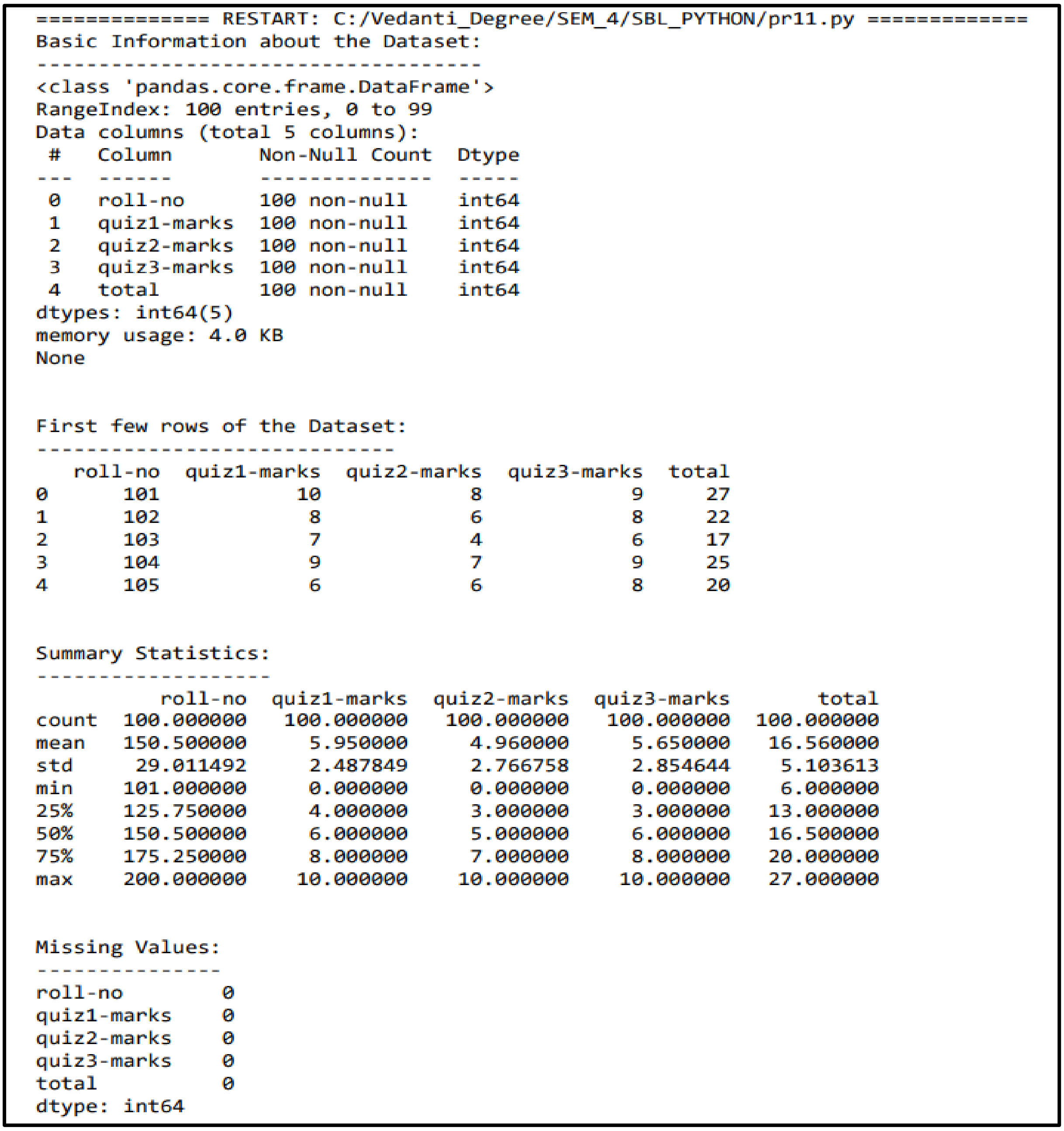
Increase the maximum number of rows to display the entire DataFrame:

```
import pandas as pd

pd.options.display.max_rows = 9999

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

print(df)
```

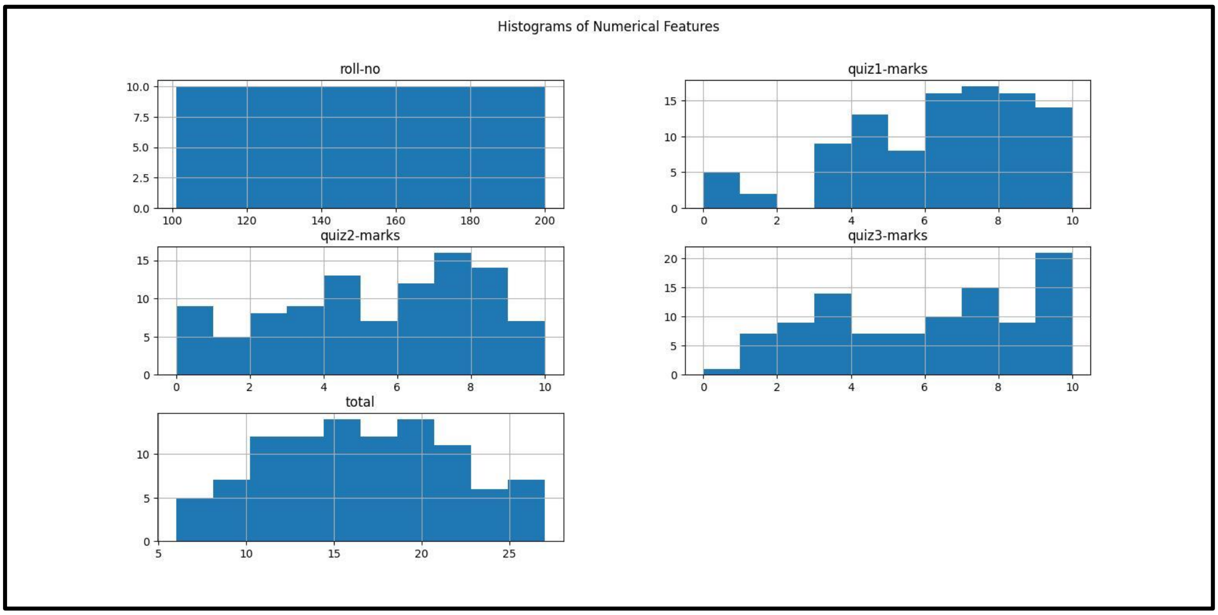
CODE:

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

data = pd.read_csv('data.csv')

print("Basic Information about the Dataset:")
print("------------------------------------")
print(data.info())
print("\n")

print("First few rows of the Dataset:")
print("-----------------------------")
print(data.head())
print("\n")

print("Summary Statistics:")
print("-------------------")
print(data.describe())
print("\n")
print("Missing Values:")
print("---------------")
print(data.isnull().sum())
print("\n")

data.hist(figsize=(10, 8))
plt.suptitle('Histograms of Numerical Features')
plt.show()
```

OUTPUT:



```
============== RESTART: C:/Vedanti_Degree/SEM_4/SBL_PYTHON/pr11.py =============
Basic Information about the Dataset:
------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   roll-no       100 non-null    int64
 1   quiz1-marks   100 non-null    int64
 2   quiz2-marks   100 non-null    int64
 3   quiz3-marks   100 non-null    int64
 4   total         100 non-null    int64
dtypes: int64(5)
memory usage: 4.0 KB
None


First few rows of the Dataset:
-----------------------------
   roll-no  quiz1-marks  quiz2-marks  quiz3-marks  total
0      101           10            8            9     27
1      102            8            6            8     22
2      103            7            4            6     17
3      104            9            7            9     25
4      105            6            6            8     20


Summary Statistics:
-------------------
          roll-no  quiz1-marks  quiz2-marks  quiz3-marks       total
count  100.000000   100.000000   100.000000   100.000000  100.000000
mean   150.500000     5.950000     4.960000     5.650000   16.560000
std     29.011492     2.487849     2.766758     2.854644    5.103613
min    101.000000     0.000000     0.000000     0.000000    6.000000
25%    125.750000     4.000000     3.000000     3.000000   13.000000
50%    150.500000     6.000000     5.000000     6.000000   16.500000
75%    175.250000     8.000000     7.000000     8.000000   20.000000
max    200.000000    10.000000    10.000000    10.000000   27.000000


Missing Values:
---------------
roll-no        0
quiz1-marks    0
quiz2-marks    0
quiz3-marks    0
total          0
dtype: int64
```



CONCLUSION:

In conclusion, Exploratory Data Analysis (EDA) is a fundamental step in the data analysis process that involves summarizing, visualizing, and understanding the main characteristics of a dataset. EDA not only helps in understanding the data but also guides subsequent analysis steps and hypothesis generation. This program provides a foundational framework for conducting EDA using Python libraries such as NumPy, Pandas, and Matplotlib.