DATA EXPLORATION LAB RECORD

Pentakota Yashoda

April 13, 2023

1.Introduction to the Data Exploration Components (Series and Data Frames) using Pandas in python
a. Import Pandas
b. Loading the data various formats (.XLS, .TXT, .CSV, JSON) using Pandas
c. Describe Data, Modify Data, Grouping Data, Filtering Data
d. Converting a variable to a different data type back to a CSV, JSON, or SQL

a) **AIM** :
To write a python program to import pandas.

DESCRIPTION:
To import pandas portion of the code tells python to bring the pandas data analysis library into your current environment.The as pd portion of the code then tells python to give pandas alias of pd .This allows you to use pandas functions by simply typing pd function name rather than pandas.

PROGRAM:
```
import pandas
data= 'CHN':['COUNTRY':'china','pop':4.5,'Area':9.59,'GDP':12.2,'CONT':'Asia'],
'IND':['COUNTRY':'India','pop':13.5,'Area':3.54,'GDP':2.7,'CONT':'Asia'],
'USA':['COUNTRY':'US','pop':9.8,'Area':5.76,'GDP':3.8,'CONT':'America']
import pandas as pd
df=pd.Dataframe(data=data).T
print(df)
```

EXPECTED OUTPUT:

	COUNTRY	pop	Area	GDP	CONT
CHN	china	4.5	9.59	12.2	Asia
IND	India	13.5	13.5	2.7	Asia
USA	US	9.8	5.76	3.8	America

OBSERVED OUTPUT:

	COUNTRY	pop	Area	GDP	CONT
CHN	china	4.5	9.59	12.2	Asia
IND	India	13.5	13.5	2.7	Asia
USA	US	9.8	5.76	3.8	America

b) **AIM** :
TO write a python program Loading the data various formats (.XLS, .TXT, .CSV, JSON) using Pandas

DESCRIPTION:
Python CSV data is a basic with data science.A comma-seperated value file use commas to seperate values.It as a delimated text file that holds tabular data plaint text python csv file.We will use for our demo-id,title,genre,rating.
Json stands for javascript object notation and is an open standard file format.While it holds attribute-

value pairs and array data types,it uses human-readability text for this.
Once can read a text file(txt) by using the pandas read fwf() function fwf stands for fixed width lines.We can use this to read fixed length or variable length text files.

PROGRAM:

```
import pandas
data= 'CHN':['COUNTRY':'china','pop':4.5,'Area':9.59,'GDP':12.2,'CONT':'Asia'],
'IND':['COUNTRY':'India','pop':13.5,'Area':3.54,'GDP':2.7,'CONT':'Asia'],
'USA':['COUNTRY':'US','pop':9.8,'Area':5.76,'GDP':3.8,'CONT':'America']
import pandas as pd
df=pd.Dataframe(data=data).T
print(df)
```

* For Loading the data for csv file

```
import pandas as pd
df=pd.read csv("c:python 311.csv")
print(df)
```

EXPECTED OUTPUT:

	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
CHN	China	1398.72	9596.96	12234.78	Asia	NaN
IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

OBSERVED OUTPUT:

	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
CHN	China	1398.72	9596.96	12234.78	Asia	NaN
IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

* For Loading the data from excel file

```
import pandas as pd
df=pd.read excel("c:311.xlsx")
print(df)
```

EXPECTED OUTPUT:

	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
CHN	China	1398.72	9596.96	12234.78	Asia	NaN
IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

OBSERVED OUTPUT:

	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
CHN	China	1398.72	9596.96	12234.78	Asia	NaN
IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

* For Loading the data for Json file

```
import pandas as pd
df= pd.read json("c:python 311.json")
print(df)
```

EXPECTED OUTPUT:

	CHN	IND	USA
COUNTRY	China	India	US
POP	1398.72	1351.16	329.74
AREA	9596.96	3287.26	9833.52
GDP	12234.78	2575.67	19485.39
CONT	Asia	Asia	N.America
IND_DAY	NaN	1947-08-15	1776-07-04

OBSERVED OUTPUT:

	CHN	IND	USA
COUNTRY	China	India	US
POP	1398.72	1351.16	329.74
AREA	9596.96	3287.26	9833.52
GDP	12234.78	2575.67	19485.39
CONT	Asia	Asia	N.America
IND_DAY	NaN	1947-08-15	1776-07-04

* For Loading the data for Text file
import pandas as pd
df=pd.readcsv("c:python 311.txt")
print(df)

EXPECTED OUTPUT:

	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
CHN	China	1398.72	9596.96	12234.78	Asia	NaN
IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

OBSERVED OUTPUT:

	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
CHN	China	1398.72	9596.96	12234.78	Asia	NaN
IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

C) **AIM** :
To write a python program to Describe data,Modify data,Grouping data,Filtering data.

DESCRIPTION:
Pandas provide a wide range of methods for selecting data according to the position and label of the rows and columns.Grouply is a pretty simple concept .We can create a grouping of categories and apply a function to the categories.Filtering data from a data frame is one of the most common operations to filter the data grouping the data based on function.

PROGRAM:
import pandas as pd
data='Name':['Yashoda','Hari','Akhila','Swathi','Pardhu'],
'DOB':[2004,2003,2005,2001,2002],
'Branch':['CSE','INF','EEE','MECH','ECE'],
'Marks':[50,75,92,35,95],
'Gender':['Female','Male','Female','Female','Male']
df=pd.Dataframe(data)
print("Describing data...")
print(df.describe())
print("Modifying data...")
df['Marks']=df['Marks']+10
print(df)
print("Grouping data..")
groupeddata = df.groupby(['Gender'])
for name, group in groupeddata:
print("",name)
print(group)
print("Filtering data..")
filtereddata = df[df['DOB']¿2002]
print(filtereddata)

EXPECTED OUTPUT:

```
Describing data...
              DOB      Marks
count     5.000000   5.000000
mean   2003.000000  69.400000
std       1.581139  26.254523
min    2001.000000  35.000000
25%    2002.000000  50.000000
50%    2003.000000  75.000000
75%    2004.000000  92.000000
max    2005.000000  95.000000

Modifying data...
      Name   DOB Branch  Marks  Gender
0  Yashoda  2004    CSE     60  Female
1     Hari  2003    INF     85    Male
2   Akhila  2005    EEE    102  Female
3   Swathi  2001   MECH     45  Female
4   Pardhu  2002    ECE    105    Male

Grouping data...
 Female
      Name   DOB Branch  Marks  Gender
0  Yashoda  2004    CSE     60  Female
2   Akhila  2005    EEE    102  Female
3   Swathi  2001   MECH     45  Female

 Male
     Name   DOB Branch  Marks Gender
1    Hari  2003    INF     85   Male
4  Pardhu  2002    ECE    105   Male

Filtering data...
      Name   DOB Branch  Marks  Gender
0  Yashoda  2004    CSE     60  Female
1     Hari  2003    INF     85    Male
2   Akhila  2005    EEE    102  Female
```

OBSERVED OUTPUT:

```
Describing data...
              DOB      Marks
count     5.000000   5.000000
mean   2003.000000  69.400000
std       1.581139  26.254523
min    2001.000000  35.000000
25%    2002.000000  50.000000
50%    2003.000000  75.000000
75%    2004.000000  92.000000
max    2005.000000  95.000000

Modifying data...
      Name   DOB Branch  Marks  Gender
0  Yashoda  2004    CSE     60  Female
1     Hari  2003    INF     85    Male
2   Akhila  2005    EEE    102  Female
3   Swathi  2001   MECH     45  Female
4   Pardhu  2002    ECE    105    Male

Grouping data...
 Female
      Name   DOB Branch  Marks  Gender
0  Yashoda  2004    CSE     60  Female
2   Akhila  2005    EEE    102  Female
3   Swathi  2001   MECH     45  Female

 Male
     Name   DOB Branch  Marks Gender
1    Hari  2003    INF     85   Male
4  Pardhu  2002    ECE    105   Male

Filtering data...
      Name   DOB Branch  Marks  Gender
0  Yashoda  2004    CSE     60  Female
1     Hari  2003    INF     85    Male
2   Akhila  2005    EEE    102  Female
```

d) **AIM** :
Write a python program to converting a variable to a different data type back to a CSV,JSON, or SQL

using pandas

DESCRIPTION :

Converting a variable to a different data type and saving to CSV: You can use the pd.to csv() function in Pandas to convert a variable (such as a DataFrame) to a CSV format. You specify the file name and other options such as the delimiter, encoding, and whether to include or exclude index in the CSV file. Once converted, the data can be saved to a CSV file using the to csv() function. Converting a variable to a different data type and saving to JSON: You can use the pd.to json() function in Pandas to convert a variable (such as a DataFrame) to a JSON format. You specify the file name and other options such as the JSON structure (orient), compression, and indentation level. Once converted, the data can be saved to a JSON file using the to json() function. Converting a variable to a different data type and saving to SQL: You can use the pd.to sql() function in Pandas to convert a variable (such as a DataFrame) to a SQL format and save it to a SQL database. You need to specify the database connection details, table name, and other options such as if exists (what to do if the table already exists), and whether to include or exclude index in the SQL table. Once converted, the data can be saved to a SQL table using the to sql() function.

PROGRAM :

```
import pandas as pd
import io
import sqlite3
sample data =
'Name': ['John', 'Jane', 'Alice', 'Bob'],
'Age': [25, 30, 35, 40],
'Salary': [50000, 60000, 70000, 80000]
df = pd.DataFrame(sample data)
json data = df.to json(6)
df from json = pd.read json(json data)
csv data = df.to csv(index=False)
df from csv = pd.read csv(io.StringIO(csv data))
conn = sqlite3.connect('example.db')
df.to sql('employee', conn, if exists='replace', index=False)
df from sql = pd.read sql('SELECT * FROM employee', conn)
print('Original DataFrame:', df)
print('DataFrame from JSON:', df from json)
print('DataFrame from CSV:', df from csv)
print('DataFrame from SQL:', df from sql)
```

EXPECTED OUTPUT:

```
Original DataFrame:
     Name  Age  Salary
0    John   25   50000
1    Jane   30   60000
2   Alice   35   70000
3     Bob   40   80000

DataFrame from JSON:
     Name  Age  Salary
0    John   25   50000
1    Jane   30   60000
2   Alice   35   70000
3     Bob   40   80000

DataFrame from CSV:
     Name  Age  Salary
0    John   25   50000
1    Jane   30   60000
2   Alice   35   70000
3     Bob   40   80000

DataFrame from SQL:
     Name  Age  Salary
0    John   25   50000
1    Jane   30   60000
2   Alice   35   70000
3     Bob   40   80000
```

OBSERVED OUTPUT:

```
Original DataFrame:
     Name  Age  Salary
0    John   25   50000
1    Jane   30   60000
2   Alice   35   70000
3     Bob   40   80000

DataFrame from JSON:
     Name  Age  Salary
0    John   25   50000
1    Jane   30   60000
2   Alice   35   70000
3     Bob   40   80000

DataFrame from CSV:
     Name  Age  Salary
0    John   25   50000
1    Jane   30   60000
2   Alice   35   70000
3     Bob   40   80000

DataFrame from SQL:
     Name  Age  Salary
0    John   25   50000
1    Jane   30   60000
2   Alice   35   70000
3     Bob   40   80000
```

2.Reading and writing files

a. Reading a CSV File
b. Writing content of data frames to CSV File
c. Reading an Excel File
d. Writing content of data frames to Excel File

a) **AIM** :
To write a python program to Reading a CSV File

DESCRIPTION:
Reading from a CSV file is done using the reader object. The CSV file is opened as a text file with Python's built-in open() function, which returns a file object. This is then passed to the reader , which does the heavy lifting.

PROGRAM:
```
import pandas as pd
data = pd.readcsv('data.csv')
print(data)
```

EXPECTED OUTPUT:

	Unnamed: 0	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
0	CHN	China	1398.72	9596.96	12234.78	Asia	NaN
1	IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
2	USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

OBSERVED OUTPUT:

	Unnamed: 0	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
0	CHN	China	1398.72	9596.96	12234.78	Asia	NaN
1	IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
2	USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

b) **AIM** :
To write a python program to Writing content of data frames to CSV File

DESCRIPTION:
In python Pandas DataFrame tocsv() function exports the DataFrame to CSV format. If a file argument is provided, the output will be the CSV file. Otherwise, the return value is a CSV format like string.

PROGRAM:
```
import pandas as pd
data = 'name': ['Alice', 'Bob', 'Charlie'], 'age': [25, 30, 35]
df = pd.DataFrame(data)
df.tocsv('sample.csv')
print(df)
```

EXPECTED OUTPUT:

	name	age
0	Alice	25
1	Bob	30
2	Charlie	35

OBSERVED OUTPUT:

	name	age
0	Alice	25
1	Bob	30
2	Charlie	35

c) **AIM** :
To write to python program to Reading an Excel File

DESCRIPTION:
In Python, we can work with the data in the excel sheet with the help of the pandas module. There is a function called the pandas readexcel function for reading the excel file. There are lots of parameters for this function, like "io", "sheetname", "dtype", etc., for reading the data in different ways. We can also get a specific part of the data using pandas readexcel function parameters.

PROGRAM:

```
import pandas as pd
data = pd.readexcel('data.xlsx')
print(data)
```

EXPECTED OUTPUT:

	Unnamed: 0	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
0	CHN	China	1398.72	9596.96	12234.78	Asia	NaN
1	IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
2	USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

OBSERVED OUTPUT:

	Unnamed: 0	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
0	CHN	China	1398.72	9596.96	12234.78	Asia	NaN
1	IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
2	USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

d) **AIM** :
To write a python program to Writing content of data frames to Excel File

DESCRIPTION:
The Writing content of data frames to an Excel file is a common operation in data analysis and data science. It involves exporting the contents of a data frame, which is a data structure in Python that organizes data in rows and columns, to an Excel file format.To write the content of a data frame to an Excel file in Python, you can use the pandas library.

PROGRAM:

```
import pandas as pd
data = 'name': ['Alice', 'Bob', 'Charlie'], 'age': [25, 30, 35]
df = pd.DataFrame(data)
df.toexcel('sample.xlsx')
print(df)
```

EXPECTED OUTPUT:

```
      name  age
0    Alice   25
1      Bob   30
2  Charlie   35
```

OBSERVED OUTPUT:

```
      name  age
0    Alice   25
1      Bob   30
2  Charlie   35
```

3.Getting the Dataset

a. Viewing your data
b. Data Set Description
c. Describe as category
d. Handling duplicates
e. Number of observations Per Category
f. Column cleanup

a) **AIM** :
To write a python program to Viewing your data

DESCRIPTION:
The head() function is used for Viewing data in pandas from starting rows and tail() is used to view the data from the bottom, by default it views the 5 rows. Using describe() we can get the description of the data, like mean, standard deviation, percentage of the same data, etc.

PROGRAM:

```
import pandas as pd
data = pd.readcsv('data.csv')
print(data)
```

EXPECTED OUTPUT:

	Unnamed: 0	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
0	CHN	China	1398.72	9596.96	12234.78	Asia	NaN
1	IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
2	USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

OBSERVED OUTPUT:

	Unnamed: 0	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
0	CHN	China	1398.72	9596.96	12234.78	Asia	NaN
1	IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
2	USA	US	329.74	9833.52	19485.39	N.America	1776-07-04

b) **AIM** :
To write a python program to Data Set Description

DESCRIPTION:
Pandas DataFrame describe() Method.The describe() method returns description of the data in the DataFrame. If the DataFrame contains numerical data, the description contains these information for each column: count - The number of not-empty values. mean - The average (mean) value.

PROGRAM:

```
import pandas as pd
data = pd.read_csv('data.csv')
description = data.describe()
print(description)
```

EXPECTED OUTPUT:

	Age	Salary
count	9.000000	9.000000
mean	30.444444	62222.222222
std	8.589399	16791.201400
min	18.000000	40000.000000
25%	25.000000	50000.000000
50%	29.000000	60000.000000
75%	35.000000	75000.000000
max	45.000000	90000.000000

OBSERVED OUTPUT:

	Age	Salary
count	9.000000	9.000000
mean	30.444444	62222.222222
std	8.589399	16791.201400
min	18.000000	40000.000000
25%	25.000000	50000.000000
50%	29.000000	60000.000000
75%	35.000000	75000.000000
max	45.000000	90000.000000

c) **AIM** :
To write a python program to Describe as category

DESCRIPTION:
In python to describe dataset as category Categorical variables can take on only a limited, and usually fixed number of possible values. Besides the fixed length, categorical data might have an order but cannot perform numerical operation. Categorical are a Pandas data type.

PROGRAM:
import pandas as pd
data = pd.read_csv('data.csv')
cat_var = 'City'
print("Value counts for", cat_var, ":",
data[cat_var].value_counts())
EXPECTED OUTPUT:

```
Value counts for City :
 New York         1
Los Angeles       1
San Francisco     1
Chicago           1
Miami             1
Houston           1
Boston            1
Denver            1
Seattle           1
Name: City, dtype: int64
```

OBSERVED OUTPUT:

```
Value counts for City :
 New York         1
Los Angeles       1
San Francisco     1
Chicago           1
Miami             1
Houston           1
Boston            1
Denver            1
Seattle           1
Name: City, dtype: int64
```

d) **AIM** :
To write a python program to Handling duplicates

DESCRIPTION:
You can identify such duplicate rows in a Pandas dataframe by calling the duplicated function. The duplicated function returns a Boolean series with value True indicating a duplicate row. By default the first row in a duplicated set is marked as False and all others marked as True.

PROGRAM:
import pandas as pd
data = pd.read_csv('data.csv')
print("Number of duplicate rows:", data.duplicated().sum())

```
import pandas as pd
data = pd.read_csv('data.csv')
data.drop_duplicates(inplace=True)
print("Number of rows after removing duplicates:", len(data))
```

EXPECTED OUTPUT:

```
Number of duplicate rows: 0
Number of rows after removing duplicates: 9
```

OBSERVED OUTPUT:

```
Number of duplicate rows: 0
Number of rows after removing duplicates: 9
```

e) **AIM** :
To write a python program to Number of observations Per Category

DESCRIPTION:
In Python Pandas, "number of observations per category" refers to the count of data points (observations) that belong to each category or group in a categorical variable. This can be calculated using the "groupby" method in Pandas, which groups the data by the categories in the categorical variable and then applies a function (such as "count") to calculate the number of observations in each group.

PROGRAM:

```
import pandas as pd
data = pd.read_csv('data.csv')
cat_counts = data[cat_var].value_counts()
print("Counts per category for", cat_var, ":", cat_counts)
```

EXPECTED OUTPUT:

```
Counts per category for City :
 New York         1
Los Angeles       1
San Francisco     1
Chicago           1
Miami             1
Houston           1
Boston            1
Denver            1
Seattle           1
Name: City, dtype: int64
```

OBSERVED OUTPUT:

```
Counts per category for City :
 New York         1
Los Angeles       1
San Francisco     1
Chicago           1
Miami             1
Houston           1
Boston            1
Denver            1
Seattle           1
Name: City, dtype: int64
```

f) **AIM** :
To write a python program to Column cleanup

DESCRIPTION:
In Python Pandas, "column cleanup" refers to the process of preparing a dataset by making sure that each column contains accurate and consistent data, and is in the appropriate format for analysis
PROGRAM:

```
import pandas as pd
data = pd.read_csv('data.csv')
columns_to_remove = ['Gender', 'Age']
data.drop(columns=columns_to_remove, inplace=True)
print(data)
```

EXPECTED OUTPUT:

	Name	City	Salary
0	John	New York	50000
1	Mary	Los Angeles	75000
2	Peter	San Francisco	40000
3	Ann	Chicago	90000
4	Tom	Miami	60000
5	Jake	Houston	55000
6	Emma	Boston	65000
7	Steve	Denver	80000
8	Lisa	Seattle	45000

OBSERVED OUTPUT:

	Name	City	Salary
0	John	New York	50000
1	Mary	Los Angeles	75000
2	Peter	San Francisco	40000
3	Ann	Chicago	90000
4	Tom	Miami	60000
5	Jake	Houston	55000
6	Emma	Boston	65000
7	Steve	Denver	80000
8	Lisa	Seattle	45000

4. Getting the Dataset continuation

a. Removing null values
b. Understanding your variables
c. Relationships between continuous variables
d. DataFrame slicing, selecting, extracting
e. Conditional selections

a) **AIM** :
To write a python program to Removing null values

DESCRIPTION:
Removing null values refers to the process of identifying and eliminating missing or null values from a dataset. Null values, also known as missing values, are data points that are not available or have not been recorded for a particular variable or observation.

PROGRAM:

```
import pandas as pd
df = pd.DataFrame('Column1': [1, 2, None, 4], 'Column2': [5, None, 7, 8])
print('Original dataframe:', df)
df = df.dropna()
print('Dataframe after removing null values:', df)
```

EXPECTED OUTPUT:

```
Original dataframe:
    Column1  Column2
0       1.0      5.0
1       2.0      NaN
2       NaN      7.0
3       4.0      8.0
Dataframe after removing null values:
    Column1  Column2
0       1.0      5.0
3       4.0      8.0
```

OBSERVED OUTPUT:

```
Original dataframe:
    Column1  Column2
0       1.0      5.0
1       2.0      NaN
2       NaN      7.0
3       4.0      8.0
Dataframe after removing null values:
    Column1  Column2
0       1.0      5.0
3       4.0      8.0
```

b) **AIM** :
To write a python program to Understanding your variables

DESCRIPTION:
1.Understanding your variables is a crucial step in data analysis and modeling, which involves exploring and describing the characteristics and properties of the variables in a dataset. 2.Understanding your variables can help you to identify patterns, relationships, and outliers in the data, and to select appropriate analysis techniques or models.

PROGRAM:

```
import pandas as pd
```

```
df = pd.DataFrame('Name': ['Alice', 'Bob', 'Charlie', 'David'],
'Age': [25, 30, 35, 40],
'Gender': ['F', 'M', 'M', 'M'],
'Salary': [50000, 70000, 60000, 80000])
print('Number of unique values in each column:', df.nunique())
```

EXPECTED OUTPUT:

```
Number of unique values in each column:
 Name      4
Age       4
Gender    2
Salary    4
dtype: int64
```

OBSERVED OUTPUT:

```
Number of unique values in each column:
 Name      4
Age       4
Gender    2
Salary    4
dtype: int64
```

c) **AIM** :
To write a python program to Relationships between continuous variables

DESCRIPTION:
The relationship between continuous variables refers to the association or pattern of behavior that exists between two or more variables that are measured on a continuous scale. Continuous variables are those that can take on any value within a certain range, such as age, height, weight, and temperature.
PROGRAM:

```
import pandas as pd
df = pd.DataFrame('variable_1': [1, 2, 3, 4, 5],
'variable_2': [10, 15, 20, 25, 30])
correlation_coefficient = df['variable_1'].corr(df['variable_2'])
print("The correlation coefficient between variable_1 and variable_2 is:", correlation_coefficient)
```

EXPECTED OUTPUT:

```
The correlation coefficient between variable_1 and variable_2 is: 1.0
```

OBSERVED OUTPUT:

```
The correlation coefficient between variable_1 and variable_2 is: 1.0
```

d) **AIM** :
To write a python program to DataFrame slicing, selecting, extracting

DESCRIPTION:
1.Slicing involves selecting a subset of the rows and/or columns from a DataFrame based on a specific range of indices. This can be achieved using the '.loc'. 2.Selecting involves filtering the rows and/or columns of a DataFrame based on certain conditions. 3.Extracting involves retrieving a specific column or row of data from a DataFrame. This can be done using the indexing operator' [] ', which allows you to select a column by label, or the' .loc' and '.iloc 'accessor methods.

PROGRAM:
```
import pandas as pd
data = 'Name': ['John', 'Mary', 'Bob', 'Alice', 'Kate'],
'Age': [25, 30, 35, 40, 45],
'Gender': ['Male', 'Female', 'Male', 'Female', 'Female'],
'Salary': [50000, 60000, 70000, 80000, 90000]
df = pd.DataFrame(data)
print(df.iloc[2:4])
print(df.loc[:, 'Name'])
print(df[df['Age'] ¿ 30])
```

EXPECTED OUTPUT:
```
    Name  Age  Gender  Salary
2    Bob   35    Male   70000
3  Alice   40  Female   80000
0     John
1     Mary
2      Bob
3    Alice
4     Kate
Name: Name, dtype: object
    Name  Age  Gender  Salary
2    Bob   35    Male   70000
3  Alice   40  Female   80000
4   Kate   45  Female   90000
```

OBSERVED OUTPUT:
```
    Name  Age  Gender  Salary
2    Bob   35    Male   70000
3  Alice   40  Female   80000
0     John
1     Mary
2      Bob
3    Alice
4     Kate
Name: Name, dtype: object
    Name  Age  Gender  Salary
2    Bob   35    Male   70000
3  Alice   40  Female   80000
4   Kate   45  Female   90000
```

e) **AIM** :
To write a python program to Conditional selections

DESCRIPTION:
Conditional selection is a technique used in data analysis to extract specific subsets of data from a larger dataset based on certain conditions or criteria. In pandas, a popular Python library for data analysis, conditional selection can be achieved using Boolean indexing.

PROGRAM:
```
import pandas as pd
data = 'Name': ['John', 'Mary', 'Bob', 'Alice', 'Kate'],
'Age': [25, 30, 35, 40, 45],
'Gender': ['Male', 'Female', 'Male', 'Female', 'Female'],
'Salary': [50000, 60000, 70000, 80000, 90000]
```

df = pd.DataFrame(data)
age_above_30 = df[df['Age'] ¿ 30]
print("Rows where Age is greater than 30:")
print(age_above_30)
female_above_70k = df[(df['Gender'] == 'Female') (df['Salary'] ¿ 70000)]
print("where Gender is Female and Salary is greater than 70000:")
print(female_above_70k)

EXPECTED OUTPUT:

```
Rows where Age is greater than 30:
    Name  Age  Gender  Salary
2    Bob   35    Male   70000
3  Alice   40  Female   80000
4   Kate   45  Female   90000

Rows where Gender is Female and Salary is greater than 70000:
    Name  Age  Gender  Salary
3  Alice   40  Female   80000
4   Kate   45  Female   90000
```

OBSERVED OUTPUT:

```
Rows where Age is greater than 30:
    Name  Age  Gender  Salary
2    Bob   35    Male   70000
3  Alice   40  Female   80000
4   Kate   45  Female   90000

Rows where Gender is Female and Salary is greater than 70000:
    Name  Age  Gender  Salary
3  Alice   40  Female   80000
4   Kate   45  Female   90000
```

5. Getting Preview of DataFrame

a. Creating DataFrames from scratch
b. Looking at top n records
c. Looking at bottom n records
d. View columns names

a) **AIM** :
To write a python program to Creating DataFrames from scratch

DESCRIPTION:
Creating DataFrames from scratch:
We can create DataFrames from scratch using various methods, such as creating a dictionary and converting it to a DataFrame, using a list of lists, or using NumPy arrays. Pandas provide the DataFrame() function to create a DataFrame from a data structure.

PROGRAM:

```
import pandas as pd
df = pd.DataFrame()
df['Name'] = ['John', 'Sarah', 'Michael', 'David']
df['Age'] = [25, 30, 35, 40]
df['Gender'] = ['Male', 'Female', 'Male', 'Male']
print(df)
```

EXPECTED OUTPUT:

```
      Name  Age  Gender
0     John   25    Male
1    Sarah   30  Female
2  Michael   35    Male
3    David   40    Male
```

OBSERVED OUTPUT:

```
      Name  Age  Gender
0     John   25    Male
1    Sarah   30  Female
2  Michael   35    Male
3    David   40    Male
```

b) **AIM** :
To write a python program to Looking at top n records

DESCRIPTION:
Looking at top n records:
To preview the top records of a DataFrame, we can use the head() function, which returns the first n rows of the DataFrame. By default, it returns the first five rows of the DataFrame, but we can specify the number of rows we want to see.

PROGRAM:

```
import pandas as pd
data = 'Name': ['John', 'Sarah', 'Michael', 'David', 'Emily'],
'Age': [25, 30, 35, 40, 45],
'Gender': ['Male', 'Female', 'Male', 'Male', 'Female']
df = pd.DataFrame(data)
top_n = 3
top_records = df.head(top_n)
```

print(top_records)

EXPECTED OUTPUT:

```
      Name  Age  Gender
0     John   25    Male
1    Sarah   30  Female
2  Michael   35    Male
```

OBSERVED OUTPUT:

```
      Name  Age  Gender
0     John   25    Male
1    Sarah   30  Female
2  Michael   35    Male
```

c) **AIM** :
To write a python program to Looking at bottom n records

DESCRIPTION:
Looking at bottom n records:
To preview the bottom records of a DataFrame, we can use the tail() function, which returns the last n rows of the DataFrame. By default, it returns the last five rows of the DataFrame, but we can specify the number of rows we want to see.

PROGRAM:

```
import pandas as pd
data = 'Name': ['John', 'Sarah', 'Michael', 'David', 'Emily'],
'Age': [25, 30, 35, 40, 45],
'Gender': ['Male', 'Female', 'Male', 'Male', 'Female']
df = pd.DataFrame(data)
bottom_n = 2
bottom_records = df.tail(bottom_n)
print(bottom_records)
```

EXPECTED OUTPUT:

```
    Name  Age  Gender
3  David   40    Male
4  Emily   45  Female
```

OBSERVED OUTPUT:

```
    Name  Age  Gender
3  David   40    Male
4  Emily   45  Female
```

d) **AIM** :
To write a python program to View columns names

DESCRIPTION:
View column names:
We can view the column names of a DataFrame by using the columns attribute, which returns an index

object containing the column names. Alternatively, we can use the head() function with a parameter of 0 to view the column names. This will return only the column names and not any data from the DataFrame.

PROGRAM:

```
import pandas as pd
data = 'Name': ['John', 'Sarah', 'Michael', 'David'],
'Age': [25, 30, 35, 40],
'Gender': ['Male', 'Female', 'Male', 'Male']
df = pd.DataFrame(data)
columns = df.columns
print(columns)
```

EXPECTED OUTPUT:

```
Index(['Name', 'Age', 'Gender'], dtype='object')
```

OBSERVED OUTPUT:

```
Index(['Name', 'Age', 'Gender'], dtype='object')
```

6. Creating New Columns, Rename Columns of Data Frames

a. Rename method helps to rename column of data frame

b. To rename the column of existing data frame set inplace = True

a) **AIM** :

To write a python program to Rename method helps to rename column of data frame

DESCRIPTION:

The rename() method is a powerful tool in pandas, a popular Python library for data analysis, that allows you to rename columns of a DataFrame. This method provides a convenient way to rename columns without having to modify the original DataFrame object

PROGRAM:

```
import pandas as pd
data = 'Name': ['John', 'Sarah', 'Michael', 'David'],
'Age': [25, 30, 35, 40],
'Gender': ['Male', 'Female', 'Male', 'Male']
df = pd.DataFrame(data)
df = df.rename(columns='Age': 'Years')
print(df.columns)
```

EXPECTED OUTPUT:

```
Index(['Name', 'Years', 'Gender'], dtype='object')
```

OBSERVED OUTPUT:

```
Index(['Name', 'Years', 'Gender'], dtype='object')
```

b) **AIM** :

To write a python program to To rename the column of existing data frame set inplace=True

DESCRIPTION:

When you want to rename columns of an existing pandas DataFrame in place, you can use the rename() method with the inplace=True parameter. This parameter allows you to modify the original DataFrame object without creating a new object.

To rename columns in place, you can use the rename() method and set inplace=True to modify the original DataFrame object. For example, the following code renames the column "old_name" to "new_name" in a DataFrame called df.

PROGRAM:

```
import pandas as pd
data = 'Name': ['John', 'Sarah', 'Michael', 'David'],
'Age': [25, 30, 35, 40],
'Gender': ['Male', 'Female', 'Male', 'Male']
df = pd.DataFrame(data)
df.rename(columns='Age': 'Years', inplace=True)
print(df.columns)
```

EXPECTED OUTPUT:

```
Index(['Name', 'Years', 'Gender'], dtype='object')
```

OBSERVED OUTPUT:

```
Index(['Name', 'Years', 'Gender'], dtype='object')
```