EX. NO.:3 WORKING WITH PANDAS DATAFRAMES

DATE:

Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns. In pandas, data structures can be created in two ways: series and dataframes.

AIM:

(i) To create a dataframe from a series
(ii) To create a dataframe from a dictionary
(iii) To create a dataframe from n-dimensional arrays
(iv) To load a dataset from an external source into a pandas dataframe

ALGORITHM:

Step 1: Start the program.

Step 2: Import the NumPy and pandas packages.

Step 3: Create a dataframe for the list of elements (numbers, dictionary, and n-dimensional arrays)

Step 4: Load a dataset from an external source into a pandas dataframe

Step 5: Display the output.

Step 6: Stop the program.

PROGRAM:

(i) CREATION OF A DATAFRAME FROM A SERIES

```
import numpy as np
import pandas as pd
print("Pandas Version:", pd.__version__)
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 500)
```

Ex.No 3 **Working of Pandas Data Frame**

1. **Write a Pandas program to create and display a DataFrame from a specified dictionary data has the index labels.**

Sample Data Frame:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1].
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', no', 'no', 'yes']}
labels = ['a', 'b','c', 'd','e', 'f' ,'g', 'h', 'i', 'j']

PROGRAM:

```
import pandas as pd
import numpy as np
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew','Laura', 'Kevin', 'Jonas'] , 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],'  attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1], 'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b','c', 'd','e', 'f' ,'g', 'h', 'i', 'j']
df=pd.DataFrame(exam_data, index=labels)
print(df)
print("Summary of the basic information about this Data Frame and its data:")
print(df.info())
```

Sample Output:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	NaN	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

Summary of the basic information about this DataFrame and its data:

<class 'pandas.core.frame.DataFrame'>

Index: 10 entries, a to j

Data columns (total 4 columns):

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, a to j
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   name      10 non-null     object
 1   score     8 non-null      float64
 2   attempts  10 non-null     int64
 3   qualify   10 non-null     object
dtypes: float64(1), int64(1), object(2)
memory usage: 400.0+ bytes
None
```

i. To get the first 3 rows of a given Data Frame.

print("First three rows of the data frame:")

print(df.iloc[:3])

Sample Output:

First three rows of the data frame:

```
          name  score  attempts qualify
a    Anastasia   12.5         1     yes
b         Dima    9.0         3      no
c    Katherine   16.5         2     yes
```

ii. To select the 'name' and 'score' columns from the following Data Frame.

print("Select specifie columns:")

print(df[['name', 'score']])

Sample Output:

```
          name  score
a    Anastasia   12.5
b         Dima    9.0
c    Katherine   16.5
d        James    NaN
e        Emily    9.0
f      Michael   20.0
g      Matthew   14.5
h        Laura    NaN
i        Kevin    8.0
j        Jonas   19.0
```

iii. To select the specified columns and rows from a given Data Frame. Select 'name' and 'score' columns in rows 1,3,5,6 from the following data frame.

```
print("Select specific columns and rows:")
print(df.iloc[[1, 3, 5, 6], [1, 31]])
```

Sample Output:

Select specific columns and rows:

```
   score qualify
b    9.0      no
d    NaN      no
f   20.0     yes
g   14.5     yes
```

iv. To select the rows where the number of attempts in the examination is greater than 2.

```
print("Number of attempts in the examination is greater than 2:")
print(df[df['attempts'] > 2])
```

Sample Output:

Number of attempts in the examination is greater than 2:

```
      name  score  attempts qualify
b     Dima    9.0         3      no
d    James    NaN         3      no
f  Michael   20.0         3     yes
```

v. To select the rows where the score is missing, i.e. is NaN.

```
print("Rows where score is missing:")
print(df[df['score'].isnull()])
```

Sample Output:

Rows where score is missing: attempts

```
    name  score  attempts qualify
d  James    NaN         3      no
h  Laura    NaN         1      no
```

vi. To change the score in row 'd' to 11.5.

```
print("nOriginal data frame:")
print(df)
print("\nChange the score in row 'd' to 11.5:")
df.loc['d','score'] = 11.5
```

```
print(df)
```

Sample Output:

Original data frame:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	NaN	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

Change the score in row 'd' to 11.5:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	11.5	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

vii. To calculate the sum of the examination score by the students.

```
print("\nSum of the examination attempts by the students:")
print(df['score'].sum())
```

Sample Output:

Sum of the examination attempts by the students: 108.5

viii. To append a new row 'k' to Data Frame with given values for each column. Now delete the new row and return the original data frame.

```
print("Original rows:",df)
print("\nAppend a new row:")
df.loc['k'] = ['Suresh', 15.5,1,'yes']
```

```
print("Print all records after insert a new record:",df)
print("\nDelete the new row and display the original rows:")
df = df.drop('k')
print(df)
```

Sample Output:

Original rows:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	NaN	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

Append a new row:

Print all records after insert a new record:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	11.5	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes
k	Suresh	15.5	1	yes

Delete the new row and display the original rows:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	11.5	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

ix. To delete the 'attempts' column from the Data Frame.

```
print("Original rows:")
print(df)
print("\nDelete the 'attempts' column from the data frame:")
df.pop('attempts')
print(df)
```

Sample Output:

Original rows:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	NaN	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

Delete the 'attempts' column from the data frame:

	name	score	qualify
a	Anastasia	12.5	yes
b	Dima	9.0	no
c	Katherine	16.5	yes
d	James	11.5	no
e	Emily	9.0	no
f	Michael	20.0	yes
g	Matthew	14.5	yes
h	Laura	NaN	no
i	Kevin	8.0	no
j	Jonas	19.0	yes

x. To insert a new column in existing Data Frame.

```
print("Original rows:")
print(df)
color=['Red','Blue','Orange','Red','White','White','Blue','Green','Green',"Red']
df['color'] = color
```

print("\n New Data Frame after inserting the 'color' column")

print(df)

Sample Output

Original rows:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	NaN	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

New Data Frame after inserting the 'color' column

	name	score	qualify	color
a	Anastasia	12.5	yes	Red
b	Dima	9.0	no	Blue
c	Katherine	16.5	yes	Orange
d	James	11.5	no	Red
e	Emily	9.0	no	White
f	Michael	20.0	yes	White
g	Matthew	14.5	yes	Blue
h	Laura	NaN	no	Green
i	Kevin	8.0	no	Green
j	Jonas	19.0	yes	Red

2. **Write a Pandas program to get the representation of an array by identifying distinct values of a column of a dataframe.**

Sample Output:

Original DataFrame:

	Name	Date_Of_Birth	Age
0	Alberto Franco	17/05/2002	18.5
1	Gino Mcneill	16/02/1999	21.2
2	Ryan Parkes	25/09/1998	22.5
3	Eesha Hinton	11/05/2002	22.0
4	Gino Mcneill	15/09/1997	23.0

Numeric representation of an array by identifying distinct values: [0 1 2 3 1]

Index: Index(['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton'], dtype='object')

Program:

```
import pandas as pd
data = {'Name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill'], 'Date_Of_Birth': ['17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002', '15/09/1997'], 'Age': [18.5, 21.2, 22.5, 22.0, 23.0]}
df = pd.DataFrame(data)
numeric_representation, unique_names = pd.factorize(df['Name'])
print("Original DataFrame:")
print(df)
print("\nNumeric representation of an array by identifying distinct values:")
print(numeric_representation)
print("Index:", unique_names)
```

Output:

Original DataFrame:

	Name	Date_Of_Birth	Age
0	Alberto Franco	17/05/2002	18.5
1	Gino Mcneill	16/02/1999	21.2
2	Ryan Parkes	25/09/1998	22.5
3	Eesha Hinton	11/05/2002	22.0
4	Gino Mcneill	15/09/1997	23.0

Numeric representation of an array by identifying distinct values: [0 1 2 3 1]

Index: Index(['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton'], dtype='object')

3. Write a Pandas program to check for inequality of two given dataframes.

Sample Output;

Original DataFrames:

	W	X	Y	Z
0	68.0	78.0	84	86
1	75.0	85.0	94	97
2	86.0	NaN	89	96
3	80.0	80.0	83	72
4	NaN	86.0	86	83

	W	X	Y	Z
0	78.0	78	84	86
1	75.0	85	84	97
2	86.0	96	89	96
3	80.0	80	83	72
4	NaN	76	86	83

Check for inequality of the said dataframes:

	W	X	Y	Z
0	True	False	False	False
1	False	False	True	False
2	False	True	False	False
3	False	False	False	False
4	True	True	False	False

Program:

```
import pandas as pd
import numpy as np
data1 = {'W': [68.0, 75.0, 86.0, 80.0, np.nan],
'X': [78.0, 85.0, np.nan, 80.0, 86.0],
'Y': [84, 94, 89, 83, 86],
'Z': [86, 97, 96, 72, 83]}
df1 = pd.DataFrame(data1)
data2 = {'W': [78.0, 75.0, 86.0, 80.0, np.nan],
'X': [78, 85, 96, 80, 76],
'Y': [84, 84, 89, 83, 86],
'Z': [86, 97, 96, 72, 83]}
df2 = pd.DataFrame(data2)
inequality_check = df1 != df2
print("Check for inequality of the two given DataFrames:")
print(inequality_check)
```

Output:

Check for inequality of the two given DataFrames:

	W	X	Y	Z
0	True	False	False	False
1	False	False	True	False
2	False	True	False	False
3	False	False	False	False
4	True	True	False	False

4. **Write a Pandas program to select all columns, except one given column in a dataframe.**

Sample Output:

Original DataFrame

	col1	col2	col3
0	1	4	7
1	2	5	8
2	3	6	12
3	4	9	1
4	7	5	11

All columns except 'col3':

	col1	col2
0	1	4
1	2	5
2	3	6
3	4	9
4	7	5

Program:

```
import pandas as pd
d = {'col1': [1, 2, 3, 4, 7], 'col2': [4, 5, 6, 9, 5], 'col3': [7, 8, 12, 1, 11]}
df = pd.DataFrame(data=d)
print("Original DataFrame")
print(df)
print("\nAll columns except 'col3':")
df = df.loc[:, df.columns != 'col3']
print(df)
```

Output:

Original DataFrame

	col1	col2	col3
0	1	4	7
1	2	5	8
2	3	6	12
3	4	9	1
4	7	5	11

All columns except 'col3':

	col1	col2
0	1	4
1	2	5
2	3	6
3	4	9
4	7	5

RESULT:

Thus, the working of Pandas Data frame using Dictionary was executed and verified successfully.