Experiment - 4.1.

AIM: i) Write a Pandas program to create and display a one-dimensional array-like object containing an array of data using Pandas module.

Description:

n this program, we are using the Pandas library in Python to create and display a one-dimensional array-like data structure known as a Series.

A Pandas Series is similar to a single column in an Excel sheet or a database table.
It can hold a collection of data (like numbers or strings), and each element in the Series is associated with a label called an index.

By default, Pandas assigns numeric indexes starting from 0, but we can also assign our own custom indexes if needed.

Algorithm

1. Start
2. Import the Pandas library
   → Use the statement: import pandas as pd
   This allows us to use all the functions of the Pandas module.
3. Create a list of data elements
   → Example: data_list = [10, 20, 30, 40, 50]
4. Create a Pandas Series from the list
   → Use the function: data_series = pd.Series(data_list)
   This converts the list into a one-dimensional labeled array (Series).
5. Display the Series on the screen
   → Use the print() function to display:
   print("Pandas Series:")
   print(data_series)
6. End

Program:

```
import pandas as pd

data=pd.Series([1,20,3,40,5])

print("one-dimensional array-like object(pandas series)")

print(data)
```

Output:

```
one-dimensional array-like object(pandas series)
0     1
1    20
2     3
3    40
4     5
dtype: int64
```

Result:

Hence to write a Pandas program to create and display a one-dimensional array-like object containing an array of data using Pandas module has been executed and its output is verified.

Experiment - 4.1.

AIM: ii) Write a Pandas program to convert a Panda module Series to Python list and it's type.

Description:

In this program, we are learning how to convert a Pandas Series into a normal Python list and check its type.
This is useful when we want to use the data outside Pandas, in regular Python operations.

Let's go through the program step by step

Step 1: Import the Pandas library

import pandas as pd

- This imports the Pandas library and assigns it a short alias pd for easy usage.
- Pandas provides various data structures and functions to handle data efficiently.

Step 2: Create data for the Series

data_list = [10, 20, 30, 40, 50]

- A normal Python list is created containing five numeric elements.
- This list will be used to create a Pandas Series.

Step 3: Create a Pandas Series

data_series = pd.Series(data_list)

- This converts the Python list into a Pandas Series.
- A Series is a one-dimensional labeled array, where each element has an index.

Step 4: Convert the Series into a Python list

python_list = data_series.tolist()

- The .tolist() method is a built-in Pandas function.
- It converts the Series data into a Python list (removing the index).

Step 5: Display the converted list and its type

print(python_list)

print(type(python_list))

- The first print statement displays the list elements.
- The second print statement shows the data type of the variable — <class 'list'>.

Algorithm

1. Start
2. Import the Pandas module using import pandas as pd.
3. Create a list of elements to store data.
4. Create a Pandas Series using pd.Series(data_list).
5. Convert the Series to a Python list using the .tolist() method.
6. Display both the converted list and its type using the print() function.
7. Stop

Program:

```
import pandas as pd
# Create a Pandas Series
data = pd.Series([1, 2, 3, 4, 5])
# Convert Series to Python list
data_list = data.tolist()
# Display the list and its type
print("Python list:", data_list)
print("Type:", type(data_list))
```

Output:

```
Python list: [1, 2, 3, 4, 5]
Type: <class 'list'>
```

Result:

Hence to write a Pandas program to convert a Panda module Series to Python list and it's type has been executed and its output is verified.

Experiment - 4.2

AIM: Consider Sample Python dictionary data and list labels:

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']

i) Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.

Description:

This program demonstrates how to create a Pandas DataFrame from a dictionary and assign custom row labels (index labels). A DataFrame is a two-dimensional labeled structure with rows and columns, similar to a spreadsheet, widely used for data analysis and manipulation.

Step-by-Step Explanation:

1. Import Libraries

import pandas as pd

import numpy as np.

2. Create Dictionary of Data

exam_data = {'name': [...], 'score': [...], 'attempts': [...], 'qualify': [...]}

- Keys → column names
- Values → list of data for each column, including missing values

3. Create List of Index Labels

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

4. Create DataFrame

df = pd.DataFrame(exam_data, index=labels)

- Converts dictionary into a DataFrame with specified row labels.

5. Display DataFrame

print(df)

- Prints the DataFrame showing columns, custom index, and data values including NaN

Algorithm

1. Start
2. Import pandas and numpy.
3. Create a dictionary with column data.
4. Create a list of custom row labels.
5. Create a DataFrame using pd.DataFrame(data, index=labels).
6. Print the DataFrame.
7. Stop

Program:

```
import pandas as pd
import numpy as np
# Sample dictionary data
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'] }
# List of index labels
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
# Create DataFrame
df = pd.DataFrame(exam_data, index=labels)
# Display DataFrame
print("DataFrame from the given dictionary data:")
print(df)
```

Output:

```
DataFrame from the given dictionary data:
        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
```

Result : Hence to Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels has been executed and its output is verified.

Experiment - 4.2

AIM: ii) Write a Pandas program to change the name 'James' to 'Suresh' in name column of the DataFrame.

Description:

A dictionary named exam_data is converted into a Pandas DataFrame using the pd.DataFrame() function with custom row labels. The DataFrame contains columns such as name, score, attempts, and qualify. The purpose of the program is to change the value 'James' to 'Suresh' in the name column. This is done using the replace() function, which searches the entire name column for 'James' and replaces it with 'Suresh'. Alternatively, the same can be achieved using the loc[] method by specifying the condition df.loc[df['name'] == 'James', 'name'] = 'Suresh'. After this operation, the updated DataFrame reflects the name change while keeping the rest of the data unchanged.

Algorithm

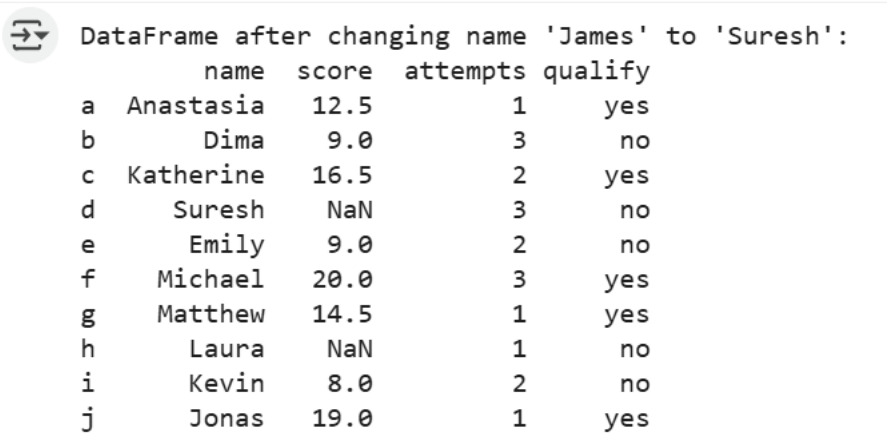
1. Start
2. Import the pandas library as pd and numpy library as np.
3. Create a dictionary named exam_data
4. Create a list named labels to represent index labels for the DataFrame.
5. Create a DataFrame df using the dictionary exam_data and the index labels.
6. Locate the row(s) where the value in the 'name' column is 'James'.
7. Replace the value 'James' with 'Suresh' in the 'name' column using:
8. df.loc[df['name'] == 'James', 'name'] = 'Suresh'
9. Display the message: "DataFrame after changing name 'James' to 'Suresh':".
10. Print the updated DataFrame df.
11. Stop

Program:

```
import pandas as pd
import numpy as np
# Sample dictionary data
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']
}
# List of index labels
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
# Create DataFrame
df = pd.DataFrame(exam_data, index=labels)
# Change name 'James' to 'Suresh'
df.loc[df['name'] == 'James', 'name'] = 'Suresh'
# Display updated DataFrame
print("DataFrame after changing name 'James' to 'Suresh':")
print(df)
```

Output:



```
DataFrame after changing name 'James' to 'Suresh':
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d     Suresh    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
```

Result: Hence to write a Pandas program to change the name 'James' to 'Suresh' in name column of the DataFrame has been executed and its output is verified.

Experiment - 4.2

AIM: iii) Write a Pandas program to insert a new column in existing DataFrame**.**

Description:

This Python program uses the Pandas library to create a DataFrame from a dictionary and then inserts a new column named 'age' into the existing DataFrame.

- The dictionary exam_data contains details such as names, scores, attempts, and qualification status of students.
- A list labels is used to assign custom index labels to each row.
- After creating the DataFrame using pd.DataFrame(), a new column 'age' is added by assigning a list of age values to df['age'].
- Finally, the updated DataFrame with the new column is displayed.

Algorithm

1. Start
2. Import pandas as pd and numpy as np
3. Create a dictionary exam_data with columns name, score, attempts, and qualify
4. Create a list labels for index values
5. Create DataFrame df using exam_data and labels
6. Insert new column age with a list of integer values
7. Print message “DataFrame after inserting new column 'age':”
8. Display updated DataFrame df
9. Stop

Program:

```
import pandas as pd
import numpy as np
# Sample dictionary data
exam_data = {
    'name': ['Anastasia', 'Dima', 'Katherine', 'Suresh', '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']
}
# List of index labels
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
# Create DataFrame
df = pd.DataFrame(exam_data, index=labels)
# Insert new column 'age' with values
df['age'] = [18, 20, 19, 21, 20, 22, 19, 18, 20, 21]
# Display updated DataFrame
print("DataFrame after inserting new column 'age':")
print(df)
```

Output:

```
DataFrame after inserting new column 'age':
        name  score  attempts qualify  age
a  Anastasia   12.5         1     yes   18
b       Dima    9.0         3      no   20
c  Katherine   16.5         2     yes   19
d     Suresh    NaN         3      no   21
e      Emily    9.0         2      no   20
f    Michael   20.0         3     yes   22
g    Matthew   14.5         1     yes   19
h      Laura    NaN         1      no   18
i      Kevin    8.0         2      no   20
j      Jonas   19.0         1     yes   21
```

Result: Hence to write a Pandas program to insert a new column in existing DataFrame has been executed and its output is verified.

Experiment - 4.2

AIM: iv) Write a Pandas program to get list from DataFrame column headers.

Description:

This Python program uses the Pandas library to create a DataFrame from a dictionary and then retrieves the list of column headers from it.

- The dictionary exam_data contains student information such as name, score, attempts, and qualification status.
- The list labels provides index labels for the DataFrame.
- After creating the DataFrame using pd.DataFrame(), the column headers are accessed using df.columns.
- The .tolist() function converts the column names into a Python list, which is then printed on the screen.

Algorithm

1. Start
2. Import pandas as pd and numpy as np
3. Create a dictionary exam_data with keys name, score, attempts, and qualify
4. Create a list labels for index values
5. Create a DataFrame df using exam_data and labels
6. Get list of column headers using df.columns.tolist() and store in column_list
7. Print message “List of column headers:”
8. Display the list column_list
9. Stop

Program:

```
import pandas as pd

import numpy as np

# Sample dictionary data

exam_data = {

   'name': ['Anastasia', 'Dima', 'Katherine', 'Suresh', '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']
}
# List of index labels
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
# Create DataFrame
df = pd.DataFrame(exam_data, index=labels)
# Get list of column headers
column_list = df.columns.tolist()
# Display the list
print("List of column headers:")
print(column_list)
```

Output:

```
List of column headers:
['name', 'score', 'attempts', 'qualify']
```

Result: Hence to write a Pandas program to get list from DataFrame column headers has been executed and its output is verified.

Experiment - 4.2

AIM: v) Write a Pandas program to get list from DataFrame row headers.

Description:

In a DataFrame, every row has an associated index label, which can either be automatically generated by Pandas (like 0, 1, 2, …) or manually assigned by the user (like ‘a’, ‘b’, ‘c’, …).

1 A dictionary named exam_data is created that stores information about students — their names, scores, number of attempts, and qualification status.

2 A list named labels is created to represent custom index labels ('a' to 'j').

3 Using this data and index labels, a DataFrame named df is created with pd.DataFrame().

4 The row headers (index labels) of this DataFrame are accessed using the df.index attribute.

5 Since df.index returns an Index object, it is converted into a standard Python list using .tolist().

6 Finally, the list of row headers is printed.

Algorithm

1 Start the program.

2 Import the Pandas and NumPy libraries.

3 Create a dictionary exam_data with keys: name, score, attempts, and qualify.

4 Create a list labels containing index labels from 'a' to 'j'.

5 Create a DataFrame df using exam_data and labels as the index.

6 Retrieve the row headers using df.index.tolist().

7 Store the result in a variable row_labels.

8 Print the list of row headers.

9 End the program.

Program:

```
import pandas as pd
import numpy as np
```

```
# Sample dictionary data
exam_data = {
    'name': ['Anastasia', 'Dima', 'Katherine', 'Suresh', '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']
}
# List of index labels
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
# Create DataFrame
df = pd.DataFrame(exam_data, index=labels)
# Get list of row headers (index labels)
row_labels = df.index.tolist()
# Display the list
print("List of row headers (index labels):")
print(row_labels)
```

Output:

```
List of row headers (index labels):
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

Result: Hence to write a Pandas program to get list from DataFrame row headers has been executed and its output is verified.