

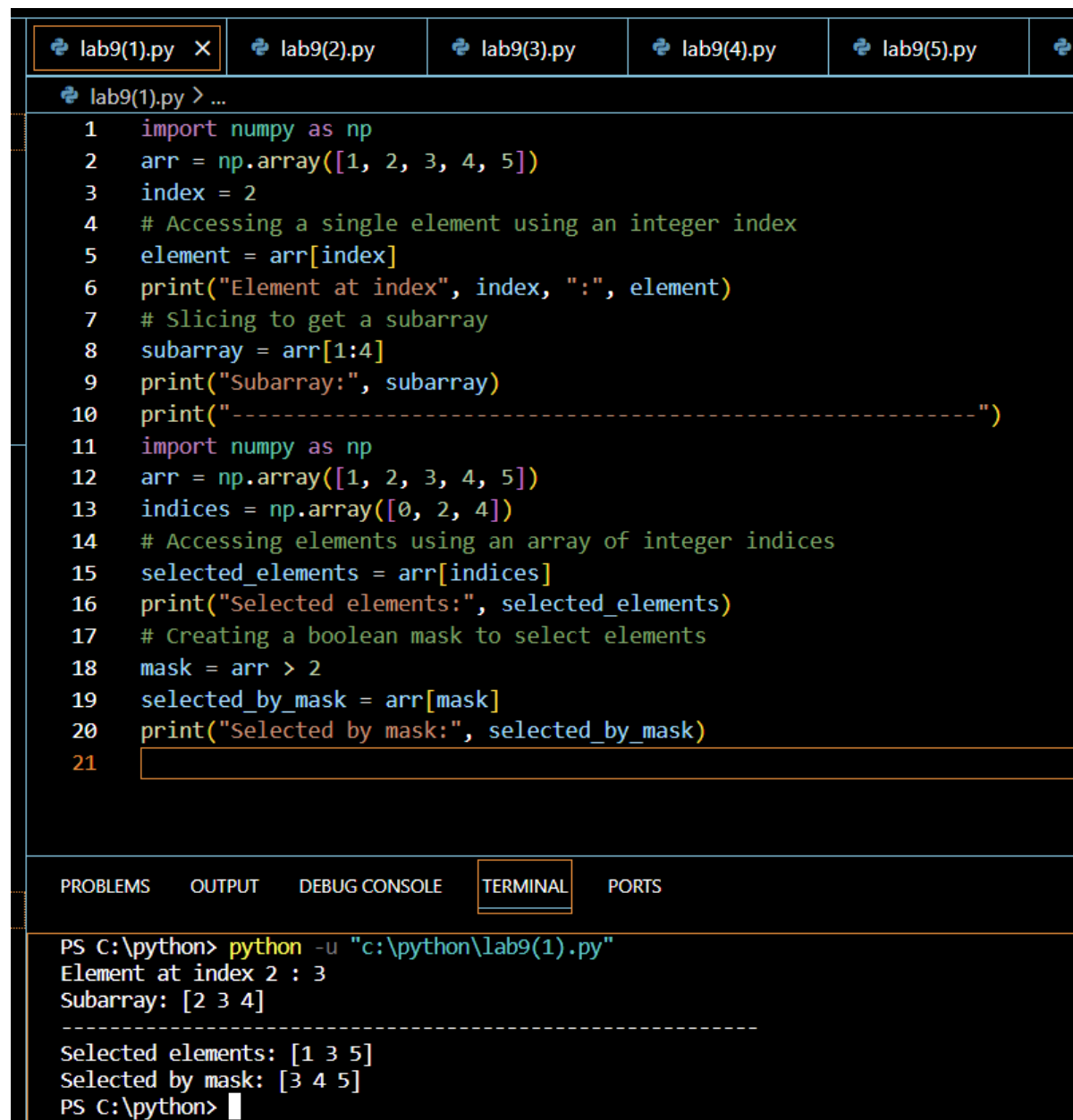
NAME: SHREYA GOEL

CLASS: 1MCA-B

REGNO: 2347255

## LAB EXERCISE-9

Q1



```
lab9(1).py X lab9(2).py lab9(3).py lab9(4).py lab9(5).py
```

```
lab9(1).py > ...
1  import numpy as np
2  arr = np.array([1, 2, 3, 4, 5])
3  index = 2
4  # Accessing a single element using an integer index
5  element = arr[index]
6  print("Element at index", index, ":", element)
7  # Slicing to get a subarray
8  subarray = arr[1:4]
9  print("Subarray:", subarray)
10 print("-----")
11 import numpy as np
12 arr = np.array([1, 2, 3, 4, 5])
13 indices = np.array([0, 2, 4])
14 # Accessing elements using an array of integer indices
15 selected_elements = arr[indices]
16 print("Selected elements:", selected_elements)
17 # Creating a boolean mask to select elements
18 mask = arr > 2
19 selected_by_mask = arr[mask]
20 print("Selected by mask:", selected_by_mask)
21
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

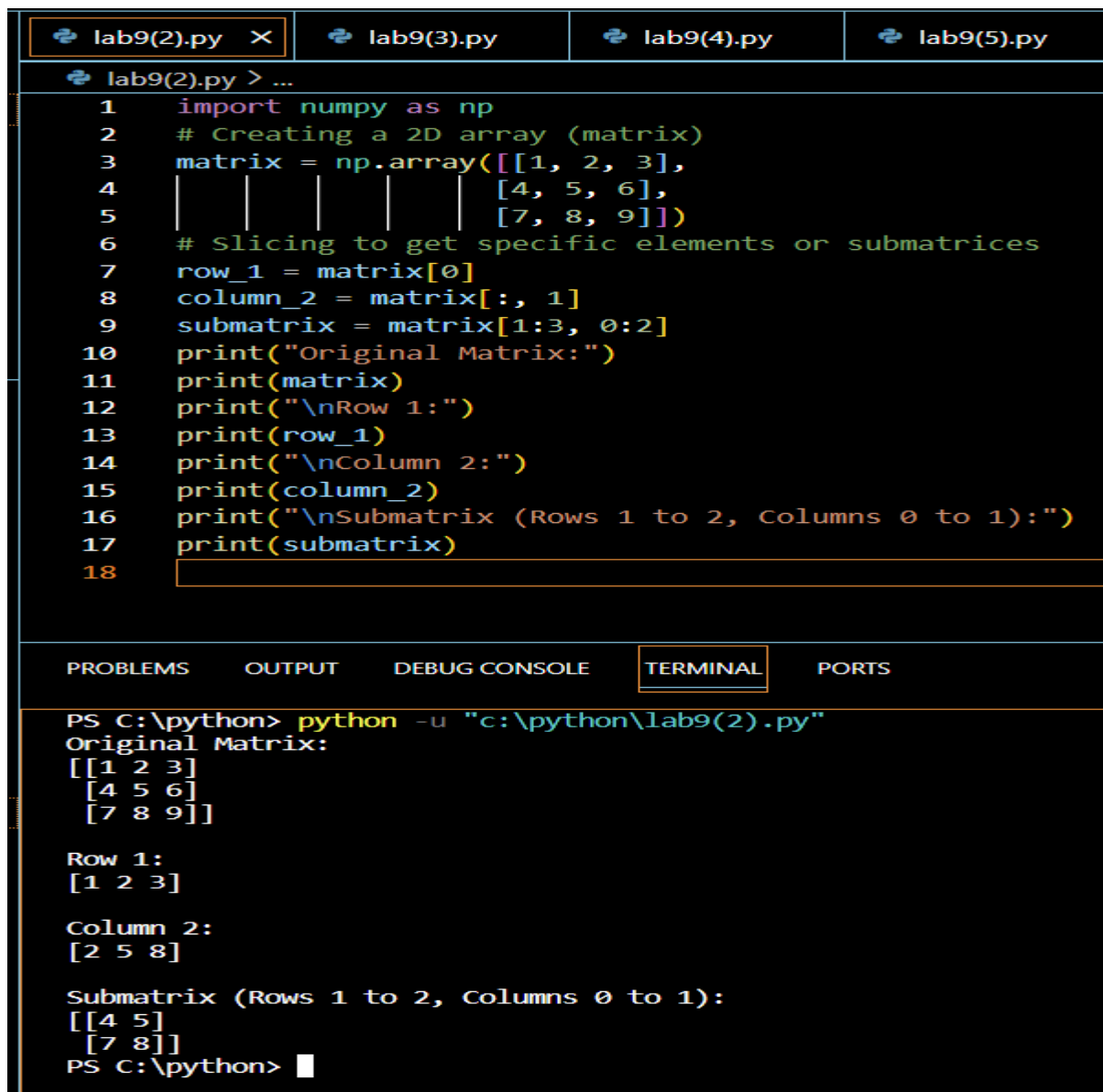
```
PS C:\python> python -u "c:\python\lab9(1).py"
Element at index 2 : 3
Subarray: [2 3 4]
-----
Selected elements: [1 3 5]
Selected by mask: [3 4 5]
PS C:\python>
```

NAME: SHREYA GOEL

CLASS: 1MCA-B

REGNO: 2347255

Q2



The image shows a code editor with four tabs: lab9(2).py, lab9(3).py, lab9(4).py, and lab9(5).py. The active tab is lab9(2).py, which contains the following Python code:

```
1 import numpy as np
2 # Creating a 2D array (matrix)
3 matrix = np.array([[1, 2, 3],
4                    [4, 5, 6],
5                    [7, 8, 9]])
6 # Slicing to get specific elements or submatrices
7 row_1 = matrix[0]
8 column_2 = matrix[:, 1]
9 submatrix = matrix[1:3, 0:2]
10 print("Original Matrix:")
11 print(matrix)
12 print("\nRow 1:")
13 print(row_1)
14 print("\nColumn 2:")
15 print(column_2)
16 print("\nSubmatrix (Rows 1 to 2, Columns 0 to 1):")
17 print(submatrix)
18
```

Below the code editor is a terminal window with the following output:

```
PS C:\python> python -u "c:\python\lab9(2).py"
Original Matrix:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Row 1:
[1 2 3]

Column 2:
[2 5 8]

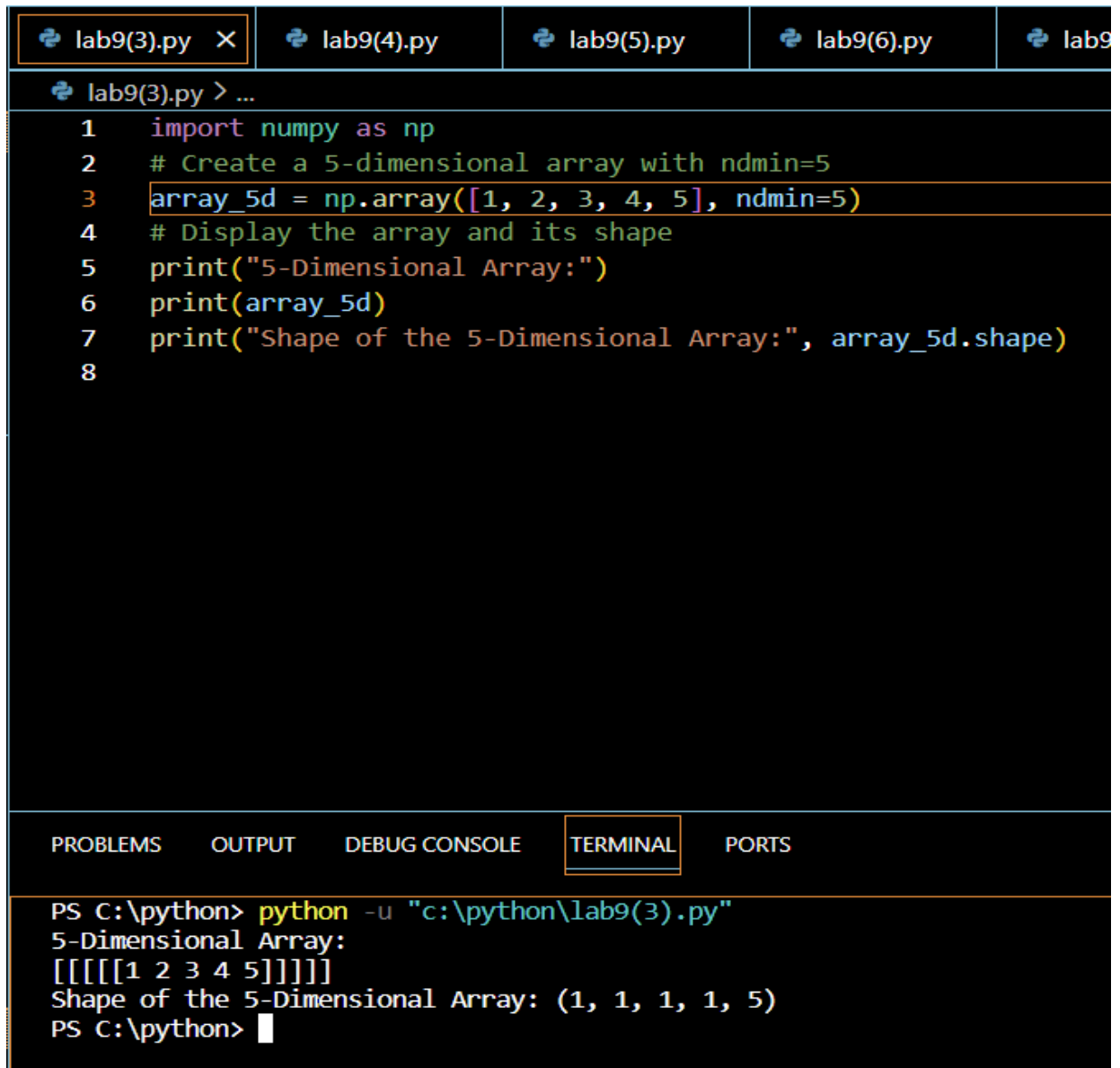
Submatrix (Rows 1 to 2, Columns 0 to 1):
[[4 5]
 [7 8]]
PS C:\python>
```

NAME: SHREYA GOEL

CLASS: 1MCA-B

REGNO: 2347255

Q3



The screenshot shows a Python IDE with a dark theme. At the top, there are tabs for lab9(3).py, lab9(4).py, lab9(5).py, lab9(6).py, and lab9(7).py. The lab9(3).py tab is active, showing a Python script. The script imports numpy as np, creates a 5-dimensional array named array\_5d with values [1, 2, 3, 4, 5] and ndmin=5, and prints the array and its shape. The output of the script is shown in the terminal window at the bottom.

```
lab9(3).py > ...  
1 import numpy as np  
2 # Create a 5-dimensional array with ndmin=5  
3 array_5d = np.array([1, 2, 3, 4, 5], ndmin=5)  
4 # Display the array and its shape  
5 print("5-Dimensional Array:")  
6 print(array_5d)  
7 print("Shape of the 5-Dimensional Array:", array_5d.shape)  
8  
  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
  
PS C:\python> python -u "c:\python\lab9(3).py"  
5-Dimensional Array:  
[[[[[1 2 3 4 5]]]]]  
Shape of the 5-Dimensional Array: (1, 1, 1, 1, 5)  
PS C:\python> 
```

NAME: SHREYA GOEL

CLASS: 1MCA-B

REGNO: 2347255

Q4

The image shows a Python IDE interface with a dark theme. At the top, there are four tabs: 'lab9(4).py' (active), 'lab9(5).py', 'lab9(6).py', and a partially visible 'lab9(7).py'. The main editor area displays the following Python code:

```
1 import numpy as np
2 # Create a 1-D array
3 array_1d = np.array([1, 2, 3, 4, 5, 6])
4 # Reshape it into a 2-D array
5 array_2d = np.reshape(array_1d, (2, 3))
6 # Display the 2-D array
7 print("2-D Array:")
8 print(array_2d)
9
```

Below the editor, there is a panel with five tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (active), and 'PORTS'. The 'TERMINAL' tab shows the execution of the script:

```
PS C:\python> python -u "c:\python\lab9(4).py"
2-D Array:
[[1 2 3]
 [4 5 6]]
PS C:\python>
```

NAME: SHREYA GOEL

CLASS: 1MCA-B

REGNO: 2347255

Q5

lab9(5).py X	lab9(6).py	lab9(7).py	lab9(8).py
--------------	------------	------------	------------

```
lab9(5).py > ...
1  import numpy as np
2  # Create two 1-D arrays
3  array1 = np.array([1, 2, 3])
4  array2 = np.array([4, 5, 6])
5  # Stack them vertically along a new axis (axis=0)
6  stacked_array = np.stack((array1, array2), axis=0)
7  print("Stacked Array:")
8  print(stacked_array)
9  print("-----")
10 # Create two 1-D arrays
11 array1 = np.array([1, 2, 3])
12 array2 = np.array([4, 5, 6])
13 # Stack them horizontally
14 hstacked_array = np.hstack((array1, array2))
15 print("Horizontally Stacked Array:")
16 print(hstacked_array)
17 print("-----")
18 # Create two 1-D arrays
19 array1 = np.array([1, 2, 3])
20 array2 = np.array([4, 5, 6])
21 # Stack them vertically
22 vstacked_array = np.vstack((array1, array2))
23 print("Vertically Stacked Array:")
24 print(vstacked_array)
25 print("-----")
26 # Create two 2-D arrays
27 array1 = np.array([[1, 2], [3, 4]])
28 array2 = np.array([[5, 6], [7, 8]])
29 # Stack them along the third axis
30 dstacked_array = np.dstack((array1, array2))
31 print("Depth-wise Stacked Array:")
32 print(dstacked_array)
33
```

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
----------	--------	---------------	----------	-------

```
PS C:\python> python -u "c:\python\lab9(5).py"
Stacked Array:
[[1 2 3]
 [4 5 6]]
-----
Horizontally Stacked Array:
[1 2 3 4 5 6]
-----
Vertically Stacked Array:
[[1 2 3]
 [4 5 6]]
-----
Depth-wise Stacked Array:
[[[1 5]
  [2 6]]

 [[3 7]
  [4 8]]]
PS C:\python>
```

NAME: SHREYA GOEL

CLASS: 1MCA-B

REGNO: 2347255

## Q6

lab9(6).py X	lab9(7).py	lab9(8).py	lab9(9).py	lab9(10).py
--------------	------------	------------	------------	-------------

```
lab9(6).py > ...
1 import numpy as np
2 # Create a sorted 1-D array
3 sorted_arr = np.array([1, 2, 3, 4, 5])
4 # Find the indices where elements should be inserted to maintain the sorted order
5 indices = np.searchsorted(sorted_arr, [2, 3, 6])
6 print("Indices:", indices)
7
```

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
----------	--------	---------------	----------	-------

```
PS C:\python> python -u "c:\python\lab9(6).py"
Indices: [1 2 5]
PS C:\python>
```

NAME: SHREYA GOEL

CLASS: 1MCA-B

REGNO: 2347255

Q7

lab9(7).py

lab9(8).py

lab9(9).py

lab9(10).py X

lab9(10).py > ...

```
1 import pandas as pd
2 import numpy as np
3 # Create a job portal DataFrame with hierarchical indexing
4 data = {'Job_Title': ['Software Engineer', 'Data Analyst', 'Product Manager', 'UI/UX Designer', 'Marketing Manager'],
5         'Salary': [90000.0, 75000.0, 110000.0, 80000.0, 95000.0],
6         'Location': ['San Francisco', 'New York', 'Seattle', 'Los Angeles', 'Chicago']}
7 # Create two levels of index
8 index1 = ['Entry Level', 'Entry Level', 'Mid Level', 'Entry Level', 'Mid Level']
9 index2 = ['Technology', 'Data', 'Management', 'Design', 'Management']
10 # Create a MultiIndex using the two levels
11 multi_index = pd.MultiIndex.from_arrays([index1, index2], names=('Level', 'Category'))
12 # Create the DataFrame with the MultiIndex
13 df = pd.DataFrame(data, index=multi_index)
14 # Display the DataFrame with hierarchical indexing
15 print("DataFrame with Hierarchical Indexing:")
16 print(df)
17
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

PS C:\python> python -u "c:\python\lab9(10).py"

DataFrame with Hierarchical Indexing:

		Job_Title	Salary	Location
Level	Category			
Entry Level	Technology	Software Engineer	90000.0	San Francisco
	Data	Data Analyst	75000.0	New York
Mid Level	Management	Product Manager	110000.0	Seattle
Entry Level	Design	UI/UX Designer	80000.0	Los Angeles
Mid Level	Management	Marketing Manager	95000.0	Chicago

PS C:\python>

NAME: SHREYA GOEL

CLASS: 1MCA-B

REGNO: 2347255

Q8

lab9(8).py X lab9(9).py lab9(10).py

lab9(8).py > ...  
1 import pandas as pd  
2 # Create data using lists  
3 data = {'Name': ['John', 'Anna', 'Peter', 'Linda', 'James'],  
4 'Age': [28, 24, 22, 32, 29],  
5 'City': ['New York', 'San Francisco', 'Chicago', 'Los Angeles', 'Boston']}  
6 # Create a DataFrame using the data  
7 df = pd.DataFrame(data)  
8 # Print the DataFrame  
9 print(df)  
10

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\python> python -u "c:\python\lab9(8).py"  
 Name Age City  
0 John 28 New York  
1 Anna 24 San Francisco  
2 Peter 22 Chicago  
3 Linda 32 Los Angeles  
4 James 29 Boston  
PS C:\python>



NAME: SHREYA GOEL

CLASS: 1MCA-B

REGNO: 2347255

Q9

```
lab9(9).py X lab9(10).py
lab9(9).py > ...
1 import pandas as pd
2 import numpy as np
3 # Create a job portal DataFrame with missing values
4 data = {'Job_Title': ['Software Engineer', 'Data Analyst', 'Product Manager', 'UI/UX Designer', 'Marketing Manager'],
5         'Salary': [90000.0, np.nan, 110000.0, 80000.0, np.nan],
6         'Location': ['San Francisco', 'New York', np.nan, 'Los Angeles', 'Chicago']}
7 df = pd.DataFrame(data)
8 # Display the DataFrame with missing values
9 print("DataFrame with Missing Values:")
10 print(df)
11 # Check for missing values
12 print("\nCheck for Missing Values (isnull()):")
13 print(df.isnull())
14 # Check for non-missing values
15 print("\nCheck for Non-Missing Values (notnull()):")
16 print(df.notnull())
17 # Drop rows with missing values
18 df_dropped = df.dropna()
19 print("\nDataFrame after Dropping Rows with Missing Values:")
20 print(df_dropped)
21 # Fill missing values with a specified value (e.g., 0)
22 df_filled = df.fillna(0)
23 print("\nDataFrame after Filling Missing Values with 0:")
24 print(df_filled)
25 # Replace missing values with a specific value (e.g., 'Not Available')
26 df_replaced = df.replace(np.nan, 'Not Available')
27 print("\nDataFrame after Replacing Missing Values with 'Not Available':")
28 print(df_replaced)
29 # Interpolate missing values using a linear method
30 df_interpolated = df.interpolate()
31 print("\nDataFrame after Interpolating Missing Values:")
32 print(df_interpolated)
33
```

Outputs: -

NAME: SHREYA GOEL

CLASS: 1MCA-B

REGNO: 2347255

```
lab9(9).py X lab9(10).py
lab9(9).py > ...
1 import pandas as pd

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

DataFrame after Replacing Missing Values with 'Not Available':
PS C:\python> python -u "c:\python\lab9(9).py"
DataFrame with Missing Values:
   Job_Title  Salary  Location
0  Software Engineer  90000.0  San Francisco
1    Data Analyst      NaN    New York
2  Product Manager  110000.0      NaN
3  UI/UX Designer   80000.0  Los Angeles
4  Marketing Manager      NaN    Chicago

Check for Missing Values (isnull()):
   Job_Title  Salary  Location
0     False     False     False
1     False     True     False
2     False     False     True
3     False     False     False
4     False     True     False

Check for Non-Missing Values (notnull()):
   Job_Title  Salary  Location
0     True     True     True
1     True    False     True
2     True     True     False
3     True     True     True
4     True    False     True

DataFrame after Dropping Rows with Missing Values:
   Job_Title  Salary  Location
0  Software Engineer  90000.0  San Francisco
3    UI/UX Designer   80000.0  Los Angeles

DataFrame after Filling Missing Values with 0:
   Job_Title  Salary  Location
0  Software Engineer  90000.0  San Francisco
1    Data Analyst     0.0    New York
2  Product Manager  110000.0     0
3  UI/UX Designer   80000.0  Los Angeles
4  Marketing Manager     0.0    Chicago

DataFrame after Replacing Missing Values with 'Not Available':
   Job_Title  Salary  Location
0  Software Engineer  90000.0  San Francisco
1    Data Analyst  Not Available  New York
```

NAME: SHREYA GOEL

CLASS: 1MCA-B

REGNO: 2347255

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

DataFrame after Replacing Missing Values with 'Not Available':
   Job Title      Salary Location
0  Software Engineer   90000.0  San Francisco
1    Data Analyst  Not Available   New York
2  Product Manager   110000.0  Not Available
3  UI/UX Designer    80000.0   Los Angeles
4  Marketing Manager  Not Available   Chicago
c:\python\lab9(9).py:30: FutureWarning: DataFrame.interpolate with object dtype is deprecated and will raise in a future version. Call obj.infer_objects(copy=False) before interpolating instead.
  df_interpolated = df.interpolate()

DataFrame after Interpolating Missing Values:
   Job Title      Salary Location
0  Software Engineer   90000.0  San Francisco
1    Data Analyst  100000.0   New York
2  Product Manager   110000.0      NaN
3  UI/UX Designer    80000.0   Los Angeles
4  Marketing Manager    80000.0   Chicago
PS C:\python>
```

## Q10

```
lab9(10).py X
lab9(10).py > ...
1  import pandas as pd
2  import numpy as np
3  # Create a job portal DataFrame with hierarchical indexing
4  data = {'Job_Title': ['Software Engineer', 'Data Analyst', 'Product Manager', 'UI/UX Designer', 'Marketing Manager'],
5         'Salary': [90000.0, 75000.0, 110000.0, 80000.0, 95000.0],
6         'Location': ['San Francisco', 'New York', 'Seattle', 'Los Angeles', 'Chicago']}
7  # Create two levels of index
8  index1 = ['Entry Level', 'Entry Level', 'Mid Level', 'Entry Level', 'Mid Level']
9  index2 = ['Technology', 'Data', 'Management', 'Design', 'Management']
10 # Create a MultiIndex using the two levels
11 multi_index = pd.MultiIndex.from_arrays([index1, index2], names=('Level', 'Category'))
12 # Create the DataFrame with the MultiIndex
13 df = pd.DataFrame(data, index=multi_index)
14 # Display the DataFrame with hierarchical indexing
15 print("DataFrame with Hierarchical Indexing:")
16 print(df)
17

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\python> python -u "c:\python\lab9(10).py"
DataFrame with Hierarchical Indexing:
   Level      Category      Job Title      Salary      Location
Entry Level Technology  Software Engineer   90000.0  San Francisco
Entry Level Data      Data Analyst    75000.0   New York
Mid Level Management  Product Manager  110000.0   Seattle
Entry Level Design    UI/UX Designer   80000.0   Los Angeles
Mid Level Management  Marketing Manager 95000.0   Chicago
PS C:\python>
```