1.

a) What code would you use to combine two DataFrames(df1, df2) along the columns?

```
combined_df = pd.concat([df1, df2], axis=1)
```

b) What function would you use to display the first five rows of a DataFrame, and why is it useful?

```
df.iloc[:5, :]
```

Or

```
df.head(5)
```

c) Write a code using Pandas to replace all NaN values in a DataFrame with zeros.

```
df.fillna(0, inplace=True)
```

d) How can you check for missing values in a DataFrame?

In Pandas, missing data is represented as NaN for numeric types and None for non-numeric types.

```
df.isnull().sum()
```

e) Write code to merge two DataFrames, df1 and df2 ?

```
merged_df = pd.merge(df1, df2, on='common_column_name')
```

Or

```
merged_df = pd.merge(df1, df2, on='common_column', how='inner')
```

Explanation: The pd.merge() function is used to combine two DataFrames based on one or more common columns. The on parameter specifies the column(s) to merge on, and the how parameter specifies the type of join (e.g., 'inner', 'outer', 'left', 'right'). In this case, we're performing an inner join, which will only include rows where the 'common_column' values match in both DataFrames.

f) How do Pandas represent missing data in numeric and non-numeric data types?

In Pandas, missing data is represented as NaN for numeric types and None for non-numeric types.

Explanation: NaN (Not a Number) is used for numeric data, while None is used for other data types. This helps Pandas identify and handle missing data appropriately.

g) Write code to filter a DataFrame df to display only rows where the 'Age' column is greater than 25.

```
filtered_df = df[df['Age'] > 25]
```

h) How can you check for missing values in a DataFrame?

We can check for missing values in a DataFrame using the isnull() or isna() functions.

Explanation: The isnull() and isna() functions are the primary way to check for missing values in a DataFrame. They return a DataFrame of the same shape but with boolean values indicating whether each element is NaN or not. This information can be used to identify the extent of missing data in your dataset.

i) Write the command to plot the points (1, 2), (3, 6), (2, 4) as a simple line plot.

```
import matplotlib.pyplot as plt
plt.plot([1, 3, 2], [2, 6, 4])
plt.show()
```

Or

```
import matplotlib.pyplot as plt
x = [1, 3, 2]
y = [2, 6, 4]
plt.plot(x, y)
plt.show()
```

j) What does the fillna() function do, and why is it useful? Provide a code example.

The fillna() function replaces missing values in a DataFrame. It is useful for handling incomplete data.

Example:

```
df.fillna(0, inplace=True)
```

k) Describe 'ffill' or 'bfill' method.

The ***ffill*** method ***fills*** missing values by propagating the last valid observation forward, while bfill propagates the next valid observation backwards.

Explanation: These methods are useful for filling gaps in time series data or other datasets where forward or backward values can logically fill the gaps.

```
df = df.fillna(method='ffill')  # Fill missing values with the previous
non-missing value

df = df.fillna(method='bfill')  # Fill missing values with the next
non-missing value
```

l) What is the purpose of the pd.concat() function in Pandas?

The purpose of pd.concat() is to concatenate two or more DataFrames along a particular axis (rows or columns).

m) Describe how to concatenate DataFrames df1 and df2 along columns.

```
combined_df = pd.concat([df1, df2], axis=1)
```

Explanation: The pd.concat() function can be used to concatenate DataFrames along either the rows (axis=0) or the columns (axis=1). In this case, we're using axis=1 to combine df1 and df2 by adding their columns side-by-side, resulting in a new DataFrame combined_df that contains all the columns from both input DataFrames.

n) How can you display a line plot after plotting it with plt.plot()?

```
plt.show()
```

2. Create two DataFrames: one for df1 and one for df2, each with id, name, and marks. Then use the pd.concat() function to combine these DataFrames.

```
import pandas as pd

# Create DataFrames
df1 = pd.DataFrame({'id': [1, 2, 3], 'name': ['Alice', 'Bob', 'Charlie'],
```

```
'marks': [90, 85, 92]})
df2 = pd.DataFrame({'id': [4, 5, 6], 'name': ['David', 'Eve', 'Frank'],
'marks': [88, 82, 95]})

# Concatenate DataFrames
combined_df = pd.concat([df1, df2])
```

3. Using the DataFrames from the previous task, combine them on the id column using pd.merge().

```
merged_df = pd.merge(df1, df2, on='id', how='inner')
```

4. Add a new column bonus to the combined DataFrame, where the bonus is 10% of the marks (from 2 no question).

```
merged_df['bonus'] = 0.1 * merged_df['marks']
```

5. Import seaborn library and load titanic dataset. Show the first 10 rows of the dataset. Check if there is any null value or not. If you find any null value, replace it with the mean value.

```
import seaborn as sns

# Load the Titanic dataset
titanic_df = sns.load_dataset('titanic')

# Show the first 10 rows
print(titanic_df.head(10))

# Check for null values
print(titanic_df.isnull().sum())

# Replace null values with the mean
titanic_df = titanic_df.fillna(titanic_df.mean())
```

6. How to read csv files? If you have been provided with a proper dataset then how will you read them?

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

7. You have the following data on monthly rainfall (in millimeters) in your city for the first six months of the year: Months January February March April May June Rainfall 80 mm 65 mm 90 mm 78 mm 105 mm 120 mm Using the data provided:

a) Create a Pandas Series to represent this data, labeling each month accordingly.

```
import pandas as pd

rainfall_data = {
    'January': 80,
    'February': 65,
    'March': 90,
    'April': 78,
    'May': 105,
    'June': 120
}

rainfall_series = pd.Series(rainfall_data)
```

Or

```
rainfall = pd.Series([80, 65, 90, 78, 105, 120], index=['January',
'February', 'March', 'April', 'May', 'June'])
```

b) Select the rainfall data for May using the Series as if it were a dictionary.

```
may_rainfall = rainfall_series['May']
```

8. You have two DataFrames with columns name, age, and score. Use pd.concat() to combine these datasets.

```
df1 = pd.DataFrame({'name': ['Alice', 'Bob', 'Charlie'], 'age': [25, 30,
35], 'score': [90, 85, 92]})
df2 = pd.DataFrame({'name': ['David', 'Eve', 'Frank'], 'age': [28, 32, 40],
'score': [88, 82, 95]})

combined_df = pd.concat([df1, df2])
```

9. Create two 1-dimensional NumPy arrays: array1 with values [1, 2, 3] and array2 with values [4, 5, 6]. Use np.concatenate() to merge them into a single array.

```
import numpy as np

array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])

combined_array = np.concatenate([array1, array2])
```