Pandas Assignment

Q1. How do you load a CSV file into a Pandas DataFrame?

```
import pandas as pd
df = pd.read_csv('station.csv',header=None)
print(df)
```

Q2. How do you check the data type of a column in a Pandas DataFrame?

```
import pandas as pd
df = pd.read_csv('station.csv',header=None)
print(df.dtypes)
```

dtypes function gives the types of column in a Pandas Dataframe

Q3. How do you select rows from a Pandas DataFrame based on a condition?

You can use square brackets to access rows from Pandas DataFrame.

```
import pandas as pd
df = pd.read_csv('station.csv',header=None)
print(df[2:4])
```

Q4. How do you rename columns in a Pandas DataFrame?

```
import pandas as pd
df = pd.read_csv('station.csv',header=None)
df.columns = ['new_col1', 'new_col2', 'new_col3', 'new_col4']
```

Q5. How do you drop columns in a Pandas DataFrame?

By using drop() function, we can delete columns

```
import pandas as pd
df = pd.read_csv('station.csv',header=None)
df.drop(columns=”column_name”)
```

Q6. How do you find the unique values in a column of a Pandas DataFrame?

To select the unique values from a specific column in a Pandas dataframe you can use the unique() method. This is simply appended to the end of the column name.

```
import pandas as pd
df = pd.read_csv('station.csv',header=None)
df.drop(columns=”column_name”).unique()
```

Q7. How do you find the number of missing values in each column of a Pandas DataFrame?

```
import pandas as pd
df = pd.read_csv('station.csv',header=None)
df.isna().sum()
```

Q8. How do you fill missing values in a Pandas DataFrame with a specific value?

Pandas is a valuable Python data manipulation tool that helps to fix missing values in your dataset, among other things.The missing data by either dropping or filling them with other values.

1. Use the fillna() Method

The **fillna()** function iterates through your dataset and fills all empty rows with a specified value. This could be the mean, median, modal, or any other value.
This pandas operation accepts some optional arguments—take note of the following ones:
Value: This is the value you want to insert into the missing rows.
Method: Let you fill in missing values forward or in reverse. It accepts a bfill or ffill parameter.
Inplace: This accepts a conditional statement. If True, it modifies the DataFrame permanently. Otherwise, it doesn't.

This method involves replacing missing values with computed averages. Filling missing data with a mean or median value is applicable when the columns involved have integer or float data types.
You can also fill in missing data with the mode value, which is the most occurring value. This is also applicable to integers or floats. But it's handier when the columns in question contain strings.

```
import pandas as pd
df = pd.read_csv('station.csv',header=None)
#To insert the mean value of each column into its missing rows:
df.fillna(df.mean(numeric_only=True).round(1), inplace=True)
#For median:
df.fillna(df.median(numeric_only=True).round(1), inplace=True)
print(df)
```

Fill Null Rows With Values Using ffill (forward-filling)

This involves specifying the fill direction inside the fillna() function. This method fills each missing row with the value of the nearest one above it.

```
df.fillna(method='ffill', inplace=True)
```

Fill Missing Rows With Values Using bfill (backward-filling)

Here, you'll replace the ffill method mentioned above with bfill. It fills each missing row in the DataFrame with the nearest value below it.

```
df.fillna(method='bfill', inplace=True)
```

2. The replace() Method

This method is handy for replacing values other than empty cells, as it's not limited to Nan values. It alters any specified value within the DataFrame.
However, like the fillna() method, you can use replace() to replace the Nan values in a specific column with the mean, median, mode, or any other value. And it also accepts the inplace keyword argument.
See how this works by replacing the null rows in a named column with its mean, median, or mode:

```
#this requires that you've previously installed numpy
import pandas
import numpy

#Replace the null values with the mean:
df['column_1'].replace([numpy.nan], df['column_1'].mean(), inplace=True)

#Replace column A with the median:
df['column_2'].replace([numpy.nan], df['column_2'].median(), inplace=True)

#Use the modal value for column C:
df['column_3'].replace([numpy.nan], df['column_3'].mode()[0], inplace=True)
print(df)
```

3. Fill Missing Data With interpolate()

The interpolate() function uses existing values in the DataFrame to estimate the missing rows. Setting the inplace keyword to True alters the DataFrame permanently.
Run the following code to see how this works:

```
#Interpolate backwardly across the column:
df.interpolate(method ='linear', limit_direction ='backward', inplace=True)
#Interpolate in forward order across the column:
df.interpolate(method ='linear', limit_direction ='forward', inplace=True)
```

Q9. How do you concatenate two Pandas DataFrames?

1. **concat():** The concat() function in pandas is used to append either columns or rows from one DataFrame to another. The concat() function does all the heavy lifting of

performing concatenation operations along an axis while performing optional set logic (union or intersection) of the indexes (if any) on the other axes.

```
import pandas as pd
# First DataFrame
df1 = pd.DataFrame({'id': ['A01', 'A02', 'A03', 'A04'],'Name': ['ABC', 'PQR', 'DEF', 'GHI']})
# Second DataFrame
df2 = pd.DataFrame({'id': ['B05', 'B06', 'B07', 'B08'], 'Name': ['XYZ', 'TUV', 'MNO', 'JKL']})
frames = [df1, df2]
result = pd.concat(frames)
display(result)
```

2. **join:** When we concatenated our DataFrames we simply added them to each other i.e. stacked them either vertically or side by side. Another way to combine DataFrames is to use columns in each dataset that contain common values (a common unique id). Combining DataFrames using a common field is called “joining”. The columns containing the common values are called “join key(s)”. Joining DataFrames in this way is often useful when one DataF

- Take the union of them all, join=’outer’. This is the default option as it results in zero information loss.
- Take the intersection, join=’inner’.

```
import pandas as pd
df1 = pd.DataFrame({'id': ['A01', 'A02', 'A03', 'A04'], 'Name': ['ABC', 'PQR', 'DEF', 'GHI']})
df3 = pd.DataFrame({'City': ['MUMBAI', 'PUNE', 'MUMBAI', 'DELHI'], 'Age': ['12', '13', '14', '12']})
# the default behaviour is join='outer'
# inner join
result = pd.concat([df1, df3], axis=1, join='inner')
display(result)
```

3. **append():** A useful shortcut to concat() is append() instance method on Series and DataFrame. These methods actually predated concat.

```
import pandas as pd
# First DataFrame
df1 = pd.DataFrame({'id': ['A01', 'A02', 'A03', 'A04'], 'Name': ['ABC', 'PQR', 'DEF', 'GHI']})
# Second DataFrame
df2 = pd.DataFrame({'id': ['B05', 'B06', 'B07', 'B08'], 'Name': ['XYZ', 'TUV', 'MNO', 'JKL']})
# append method
result = df1.append(df2)
display(result)
```

append() method also takes multiple objects to concatenate.

```
import pandas as pd
```

```
# First DataFrame
df1 = pd.DataFrame({'id': ['A01', 'A02', 'A03', 'A04'], 'Name': ['ABC', 'PQR', 'DEF', 'GHI']})
# Second DataFrame
df2 = pd.DataFrame({'id': ['B05', 'B06', 'B07', 'B08'], 'Name': ['XYZ', 'TUV', 'MNO', 'JKL']})
df3 = pd.DataFrame({'City': ['MUMBAI', 'PUNE', 'MUMBAI', 'DELHI'],'Age': ['12', '13', '14', '12']})
  # appending multiple DataFrame
result = df1.append([df2, df3])
display(result)
```

Q10. How do you merge two Pandas DataFrames on a specific column?

We can merge two Pandas DataFrames on certain columns using the merge function by simply specifying the certain columns for merge.

Syntax: DataFrame.merge(right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False, sort=False, copy=True, indicator=False, validate=None)

Example1: Let's create a Dataframe and then merge them into a single dataframe.
Creating a Dataframe:

```
# importing modules
import pandas as pd
# creating a dataframe
df1 = pd.DataFrame({'Name':['Raju', 'Rani', 'Geeta', 'Sita', 'Sohit'],'Marks':[80, 90, 75, 88, 59]})
# creating another dataframe with different data
df2 = pd.DataFrame({'Name':['Raju', 'Divya', 'Geeta', 'Sita'], 'Grade':['A', 'A', 'B', 'A'],
                    'Rank':[3, 1, 4, 2 ],'Gender':['Male', 'Female', 'Female', 'Female']})
# display df1
display(df1)
# display df2
display(df2)
df1.merge(df2[['Name','Grade','Rank']])
```

Example 2: In the resultant dataframe Grade column of df2 is merged with df1 based on key column Name with merge type left i.e. all the values of left dataframe (df1) will be displayed.

```
# importing modules
import pandas as pd
# creating a dataframe
df1 = pd.DataFrame({'Name':['Raju', 'Rani', 'Geeta', 'Sita', 'Sohit'],'Marks':[80, 90, 75, 88, 59]})
# creating another dataframe with different data
df2 = pd.DataFrame({'Name':['Raju', 'Divya', 'Geeta', 'Sita'],'Grade':['A', 'A', 'B', 'A'],
                    'Rank':[3, 1, 4, 2 ], 'Gender':['Male', 'Female', 'Female', 'Female']})
# display df1
display(df1)
# display df2
display(df2)
# applying merge with more parameters
```

```
df1.merge(df2[['Grade', 'Name']], on = 'Name', how = 'left')
```

Example 3: In this example, we have merged df1 with df2. The Marks column of df1 is merged with df2 and only the common values based on key column Name in both the dataframes are displayed here.

```
# importing modules
import pandas as pd

# creating a dataframe
df1 = pd.DataFrame({'Name':['Raju', 'Rani', 'Geeta', 'Sita', 'Sohit'],'Marks':[80, 90, 75, 88, 59]})
# creating another dataframe with different data
df2 = pd.DataFrame({'Name':['Raju', 'Divya', 'Geeta', 'Sita'],  'Grade':['A', 'A', 'B', 'A'],
                    'Rank':[3, 1, 4, 2 ], 'Gender':['Male', 'Female', 'Female', 'Female']})
# display df1
display(df1)
# display df2
display(df2)
# applying merge with more parameters
df2.merge(df1[['Marks', 'Name']])
```

Q11. How do you group data in a Pandas DataFrame by a specific column and apply an aggregation function?

```
import pandas as pd
df = pd.read_csv('station.csv',header=None)
result = df.groupby('Courses')['Fee','Discount'].aggregate('sum')
print(result)
```

Q12. How do you pivot a Pandas DataFrame?

The pivot() function is used to reshape a given DataFrame organized by given index / column values. This function does not support data aggregation, multiple values will result in a MultiIndex in the columns.

Syntax: DataFrame.pivot(self, index=None, columns=None, values=None)

Q13. How do you change the data type of a column in a Pandas DataFrame?

The data type of a column in a Pandas dataframe can be changed in many ways. Datframe.astype() method is used to cast pandas objects to a specified dtype. This function also provides the capability to convert any suitable existing column to a categorical type.

We can pass pandas.to_numeric, pandas.to_datetime, and pandas.to_timedelta as arguments to apply the apply() function to change the data type of one or more columns to numeric, Date Time, and time delta respectively.

A new DataFrame with each column's data type changed to the best one is returned by the convert dtypes() method.

Q14. How do you sort a Pandas DataFrame by a specific column?

Repeated question as Q22.

Q15. How do you create a copy of a Pandas DataFrame?

The `copy()` method returns a copy of the DataFrame.

By default, the copy is a "deep copy" meaning that any changes made in the original DataFrame will NOT be reflected in the copy.

Q16. How do you filter rows of a Pandas DataFrame by multiple conditions?

By using df[], loc[], query(), eval() and numpy.where() we can filter Pandas DataFrame by multiple conditions. The process of applying multiple filter conditions in Pandas DataFrame is one of the most frequently performed tasks while manipulating data. Pandas provide several techniques to retrieve subsets of data from the DataFrame efficiently.

Q17. How do you calculate the mean of a column in a Pandas DataFrame?

```
import pandas as pd
data = [[10,12,13], [34,23,45],[32,34,21]]
df = pd.DataFrame(data, columns=["A","B","C"])
df.mean()
```

Output:

```
A    25.333333
B    23.000000
C    26.333333
dtype: float64
```

Q18. How do you calculate the standard deviation of a column in a Pandas DataFrame?

```
import pandas as pd
data = [[10,12,13], [34,23,45],[32,34,21]]
df = pd.DataFrame(data, columns=["A","B","C"])
df.std()
```

Output:

```
A    13.316656
B    11.000000
C    16.653328
dtype: float64
```

Q19. How do you calculate the correlation between two columns in a Pandas DataFrame?

```
import pandas as pd
data = [[10,12,13], [34,23,45],[32,34,21]]
df = pd.DataFrame(data, columns=["A","B","C"])
df.mean()
```

Output:

Q20. How do you select specific columns in a DataFrame using their labels?

DataFrame.loc[] is used to select a single column or multiple columns from pandas DataFrame by column names/label.

Q21. How do you select specific rows in a DataFrame using their indexes?

DataFrame.iloc[] is used to select a single column or multiple columns from pandas DataFrame by column index/position.

Q22. How do you sort a DataFrame by a specific column?

Pandas `sort_values()` method sorts a data frame in Ascending or Descending order of passed Column.

Q23. How do you create a new column in a DataFrame based on the values of another column?

DataFrame.apply() function can be used to create new column based on the values of another column.

Q24. How do you remove duplicates from a DataFrame?

Pandas drop_duplicates() method helps in removing duplicates from the Pandas Dataframe.

Q25. What is the difference between .loc and .iloc in Pandas?

.loc and .iloc are used in slicing data from the Pandas DataFrame. They help in the convenient selection of data from the DataFrame in Python. They are used in filtering the data according to some conditions.

The loc() function is label based data selecting method which means that we have to pass the name of the row or column which we want to select. This method includes the last element of the range passed in it.

The iloc() function is an indexed-based selecting method which means that we have to pass an integer index in the method to select a specific row/column. This method does not include the last element of the range passed in it.

==========END==========