Pandas Advance Assignment 20

October 27, 2023

Q1. List any five functions of the pandas library with execution.

ANS:

0.0.1 read_csv()

The ‘read_csv() function is used to read data from a CSV file and create a Pandas DataFrame.

[ ]:
```
import pandas as pd

# Read data from a CSV file into a DataFrame
df = pd.read_csv('data.csv')
```

0.0.2 head()

The head() function is used to display the first few rows of a DataFrame to get a quick overview of the data.

[ ]:
```
# Display the first 5 rows of the DataFrame
df.head()
```

0.0.3 groupby()

The groupby() function is used for grouping and aggregating data in a DataFrame.

[ ]:
```
# Group data by a specific column and calculate the mean of another column
grouped = df.groupby('Category')['Value'].mean()
```

0.0.4 fillna()

The fillna() function is used to fill missing values in a DataFrame with specified values or methods.

[ ]:
```
# Fill missing values in a DataFrame with a specific value
df.fillna(0, inplace=True)
```

0.0.5 pivot_table()

The pivot_table() function is used to create a pivot table from a DataFrame, which is useful for summarizing and reshaping data.

```
[ ]: # Create a pivot table from a DataFrame
     pivot = pd.pivot_table(df, values='Sales', index='Category', columns='Region',␣
      ↪aggfunc='sum')
```

```
[ ]:
```

Q2. Given a Pandas DataFrame df with columns 'A', 'B', and 'C', write a Python function to re-index the DataFrame with a new index that starts from 1 and increments by 2 for each row.

```
[4]: import pandas as pd

     data={'A':(10,20,30),'B':(40,50,60),'C':(70,80,90)}
     df=pd.DataFrame(data)
```

```
[5]: df
```

```
[5]:     A   B   C
     0  10  40  70
     1  20  50  80
     2  30  60  90
```

```
[6]: def reindex_with_custom_index(df):
         new_index=range(1,len(df)*2,2)
         df.index=new_index
         return df
```

```
[7]: df_reindexed = reindex_with_custom_index(df)
     print(df_reindexed)
```

```
    A   B   C
1  10  40  70
3  20  50  80
5  30  60  90
```

```
[ ]:
```

Q3. You have a Pandas DataFrame df with a column named 'Values'. Write a Python function that iterates over the DataFrame and calculates the sum of the first three values in the 'Values' column. The function should print the sum to the console. For example, if the 'Values' column of df contains the values [10, 20, 30, 40, 50], your function should calculate and print the sum of the first three values, which is 60.

```
[8]: import pandas as pd

     data = {'Values': [10, 20, 30, 40, 50]}
     df = pd.DataFrame(data)
```

```python
def calculate_sum_of_first_three(df):
    first_three_values = df['Values'][:3]

    sum_of_first_three = first_three_values.sum()

    print("Sum of the first three values:", sum_of_first_three)

calculate_sum_of_first_three(df)
```

```
Sum of the first three values: 60
```

[ ]:

Q4. Given a Pandas DataFrame df with a column 'Text', write a Python function to create a new column 'Word_Count' that contains the number of words in each row of the 'Text' column.

[9]:
```python
import pandas as pd

data = {'Text': ['This is a sample sentence.',
                 'Another example with more words.',
                 'Only one word.']}
df = pd.DataFrame(data)

def add_word_count_column(df):
    df['Word_Count'] = df['Text'].apply(lambda x: len(x.split()))

add_word_count_column(df)

print(df)
```

```
                               Text  Word_Count
0        This is a sample sentence.           5
1  Another example with more words.           5
2                    Only one word.           3
```

[ ]:

Q5. How are DataFrame.size() and DataFrame.shape() different?

ANS:

DataFrame.size: This attribute returns the total number of elements in the DataFrame. It is calculated as the product of the number of rows and the number of columns in the DataFrame. The value represents the total count of individual data points in the DataFrame. For example, if you have a DataFrame with 3 rows and 4 columns, the `size` attribute will return 12.

DataFrame.shape: This attribute returns a tuple containing two values. The first value is the number of rows in the DataFrame, and the second value is the number of columns. It represents the dimensions or shape of the DataFrame. For example, if you have a DataFrame with 3 rows and

4 columns, the `shape` attribute will return (3, 4), indicating that the DataFrame has 3 rows and 4 columns.

```
[ ]:
```

Q6. Which function of pandas do we use to read an excel file?

ANS:

We use the read_excel() function in Pandas to read data from an Excel file into a DataFrame. This function allows you to import data from Excel spreadsheets and work with it in a Pandas DataFrame.

```
[ ]:
```

Q7. You have a Pandas DataFrame df that contains a column named 'Email' that contains email addresses in the format 'username@domain.com'. Write a Python function that creates a new column 'Username' in df that contains only the username part of each email address.

```
[10]: import pandas as pd

data = {'Email': ['john@example.com', 'jane@test.com', 'bob@mydomain.net']}
df = pd.DataFrame(data)

def extract_username(df):
    df['Username'] = df['Email'].str.split('@').str[0]

extract_username(df)
print(df)
```

```
              Email Username
0  john@example.com     john
1     jane@test.com     jane
2  bob@mydomain.net      bob
```

```
[ ]:
```

Q8. You have a Pandas DataFrame df with columns 'A', 'B', and 'C'. Write a Python function that selects all rows where the value in column 'A' is greater than 5 and the value in column 'B' is less than 10. The function should return a new DataFrame that contains only the selected rows.

For example, if df contains the following values:

A B C

0 3 5 1

1 8 2 7

2 6 9 4

3 2 3 5

4 9 1 2

Your function should select the following rows: A B C

1 8 2 7

4 9 1 2

The function should return a new DataFrame that contains only the selected rows.

[11]:
```python
import pandas as pd

data = {'A': [3, 8, 6, 2, 9],
        'B': [5, 2, 9, 3, 1],
        'C': [1, 7, 4, 5, 2]}
df = pd.DataFrame(data)

def select_rows(df):

    selected_rows = df[(df['A'] > 5) & (df['B'] < 10)]

    return selected_rows

selected_df = select_rows(df)
print(selected_df)
```

```
   A  B  C
1  8  2  7
2  6  9  4
4  9  1  2
```

[ ]:

Q9. Given a Pandas DataFrame df with a column 'Values', write a Python function to calculate the mean, median, and standard deviation of the values in the 'Values' column.

[12]:
```python
import pandas as pd

data = {'Values': [10, 20, 30, 40, 50]}
df = pd.DataFrame(data)

def calculate_statistics(df):
    mean = df['Values'].mean()
    median = df['Values'].median()
    std_dev = df['Values'].std()

    return mean, median, std_dev

mean, median, std_dev = calculate_statistics(df)
```

```
print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Standard Deviation: {std_dev}")
```

```
Mean: 30.0
Median: 30.0
Standard Deviation: 15.811388300841896
```

[ ]:

```

```

Q10. Given a Pandas DataFrame df with a column ‘Sales’ and a column ‘Date’, write a Python function to create a new column ‘MovingAverage’ that contains the moving average of the sales for the past 7 days for each row in the DataFrame. The moving average should be calculated using a window of size 7 and should include the current day.

[13]:

```
import pandas as pd

data = {'Date': ['2023-10-01', '2023-10-02', '2023-10-03', '2023-10-04',␣
 ↪'2023-10-05', '2023-10-06', '2023-10-07'],
        'Sales': [10, 20, 30, 40, 50, 60, 70]}
df = pd.DataFrame(data)

df['Date'] = pd.to_datetime(df['Date'])

def calculate_moving_average(df):
    df = df.sort_values(by='Date')

    df['MovingAverage'] = df['Sales'].rolling(window=7, min_periods=1).mean()

    return df

df_with_moving_average = calculate_moving_average(df)

print(df_with_moving_average)
```

```
        Date  Sales  MovingAverage
0 2023-10-01     10           10.0
1 2023-10-02     20           15.0
2 2023-10-03     30           20.0
3 2023-10-04     40           25.0
4 2023-10-05     50           30.0
5 2023-10-06     60           35.0
6 2023-10-07     70           40.0
```

[ ]:

```

```

Q11. You have a Pandas DataFrame df with a column ‘Date’. Write a Python function that creates a new column ‘Weekday’ in the DataFrame. The ‘Weekday’ column should contain the weekday name (e.g. Monday, Tuesday) corresponding to each date in the 'Date' column.

For example, if df contains the following values:

Date

0 2023-01-01

1 2023-01-02

2 2023-01-03

3 2023-01-04

4 2023-01-05

Your function should create the following DataFrame:

Date Weekday

0 2023-01-01 Sunday

1 2023-01-02 Monday

2 2023-01-03 Tuesday

3 2023-01-04 Wednesday

4 2023-01-05 Thursday

The function should return the modified DataFrame.

[14]:
```python
import pandas as pd

data = {'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04',␣
  ↪'2023-01-05']}
df = pd.DataFrame(data)

df['Date'] = pd.to_datetime(df['Date'])

def add_weekday_column(df):
    df['Weekday'] = df['Date'].dt.strftime('%A')

    return df

df_with_weekday = add_weekday_column(df)

print(df_with_weekday)
```

```
        Date    Weekday
0 2023-01-01     Sunday
1 2023-01-02     Monday
2 2023-01-03    Tuesday
3 2023-01-04  Wednesday
4 2023-01-05   Thursday
```

```
[ ]:
```

Q12. Given a Pandas DataFrame df with a column 'Date' that contains timestamps, write a Python function to select all rows where the date is between '2023-01-01' and '2023-01-31'.

```
[15]: import pandas as pd

data = {'Date': ['2023-01-15', '2023-02-10', '2023-01-05', '2023-01-25',␣
 ↪'2023-03-08']}
df = pd.DataFrame(data)

df['Date'] = pd.to_datetime(df['Date'])

def select_rows_between_dates(df):
    selected_rows = df[(df['Date'] >= '2023-01-01') & (df['Date'] <=␣
 ↪'2023-01-31')]

    return selected_rows

selected_df = select_rows_between_dates(df)
print(selected_df)
```

```
        Date
0 2023-01-15
2 2023-01-05
3 2023-01-25
```

```
[ ]:
```

Q13. To use the basic functions of pandas, what is the first and foremost necessary library that needs to be imported?

ANS:

0.1 import pandas as pd

```
[ ]:
```