assignment=20

July 28, 2023

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

read_csv(): This function is used to read a CSV file and create a DataFrame object.

head(): This function is used to display the first few rows of a DataFrame.

groupby(): This function is used to group rows of a DataFrame based on one or more columns.

fillna(): This function is used to replace missing values (NaN) in a DataFrame with a specified value.

plot(): This function is used to create plots from data in a DataFrame.

2 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.

[1]:
```
import pandas as pd

def reindex_df(df):
    new_index = pd.RangeIndex(start=1, step=2, stop=len(df)*2)
    return df.reindex(new_index)
```

3 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.

[2]:
```
import pandas as pd

def sum_first_three(df):
    values_sum = 0
    for i in range(3):
```

```
        values_sum += df['Values'][i]
    print("Sum of the first three values in 'Values' column: ", values_sum)

df = pd.DataFrame()
df['Values']= [10, 20, 30, 40, 50]
sum_first_three(df)
```

```
Sum of the first three values in 'Values' column:  60
```

4 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.

[3]:
```
import pandas as pd
def createNewCol(df):
    df['Word_Count'] = []
    return df
df = pd.DataFrame()
df['Text'] = []
createNewCol(df)
```

[3]:
```
Empty DataFrame
Columns: [Text, Word_Count]
Index: []
```

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

DataFrame.size and DataFrame.shape are two different attributes of a pandas DataFrame object that provide different information about the DataFrame:

DataFrame.size: This attribute returns the total number of elements in the DataFrame, which is equal to the product of the number of rows and columns. For example, if a DataFrame has 5 rows and 3 columns, df.size will return 15.

DataFrame.shape: This attribute returns a tuple containing the number of rows and columns in the DataFrame, respectively. For example, if a DataFrame has 5 rows and 3 columns, df.shape will return (5, 3).

In summary, DataFrame.size returns the total number of elements in the DataFrame, while DataFrame.shape returns the number of rows and columns as a tuple. Both attributes can be useful in different contexts depending on the information you need about the DataFrame.

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

read_excel()

7 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.

[4]:
```
import pandas as pd

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

df = pd.DataFrame()
df['Email'] = ['username@domain.com', 'sonaljinjala@gmail.com']
extract_username(df)
```

[4]:

	Email	Username
0	username@domain.com	username
1	sonaljinjala@gmail.com	sonaljinjala

8 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
```

[5]:
```
import pandas as pd

def select_rows(df):
    selected_rows = df[(df['A'] > 5) & (df['B'] < 10)]
    return selected_rows

def main():
```

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

    # select rows where A > 5 and B < 10
    selected_df = select_rows(df)

    # print the selected rows
    print(selected_df)

if __name__ == '__main__':
    main()
```

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

9 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.

[6]:
```
import pandas as pd

def calculate_stats(df):
    mean = df['Values'].mean()
    median = df['Values'].median()
    std = df['Values'].std()
    return mean, median, std
df = pd.DataFrame()
df['Values']= [10, 20, 30, 40, 50]
result=calculate_stats(df)
mean = result[0]
median = result[1]
std = result[2]
print("Mean:",str(mean)+ ", Median:",str(median)+ ", Standard deviation:
 ↪",str(std))
```

```
Mean: 30.0, Median: 30.0, Standard deviation: 15.811388300841896
```

10 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.

[8]:
```python
import pandas as pd

def add_moving_average(df):
    # Create a rolling window of size 7 for the 'Sales' column
    rolling_sales = df['Sales'].rolling(window=7)

    # Calculate the moving average for each window
    moving_average = rolling_sales.mean()

    # Add the 'MovingAverage' column to the DataFrame
    df['MovingAverage'] = moving_average

    return df
```

11 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.
```

[9]:
```
import pandas as pd

def add_weekday(df):
    df['Date'] = pd.to_datetime(df['Date'])
    df['Weekday'] = df['Date'].dt.day_name()
    return df

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

if __name__ == '__main__':
    main()
```

```
        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
```

12 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'.

[10]:
```
import pandas as pd

def select_january_dates(df):
    df['Date'] = pd.to_datetime(df['Date'])
    start_date = '2023-01-01'
    end_date = '2023-01-31'
    mask = (df['Date'] >= start_date) & (df['Date'] <= end_date)
    return df.loc[mask]

def main():
    df = pd.DataFrame({'Date': ['2022-12-31', '2023-01-01', '2023-01-15',␣
 ↪'2023-01-31', '2023-02-15']})
    print("Original DataFrame:")
    print(df)
    df = select_january_dates(df)
    print("\nDataFrame with January dates only:")
    print(df)

if __name__ == '__main__':
```

```
    main()
```

```
Original DataFrame:
         Date
0  2022-12-31
1  2023-01-01
2  2023-01-15
3  2023-01-31
4  2023-02-15

DataFrame with January dates only:
        Date
1 2023-01-01
2 2023-01-15
3 2023-01-31
```

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

import pandas as pd

As pandas is dependent on the Numpy libary, We need to import thisndependency.

[1]:
```
import pandas as pd
import numpy as np
```

[7]:
```
#not mabdatory but in case,we  want to check.

print('np:{}'.format(np.__version__))
print('pd:{}'.format(pd.__version__))
```

```
np:1.23.4
pd:1.5.1
```

[ ]:
```

```