Assignment

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

Ans:`read_csv()` - Read a CSV file into a DataFrame:

- This function is used to read data from a CSV file and create a DataFrame.

python

Copy code

```
import        as


                'example.csv'
print
```

`head()` - Display the first n rows of a DataFrame:

- This function is used to display the first few rows of a DataFrame.

python

Copy code

```


print         3
```

`groupby()` - Group a DataFrame by one or more columns:

- This function is used to group the DataFrame based on one or more columns and perform aggregate operations.

python

Copy code

```


                    'Category'   'Value'
print
```

`fillna()` - Fill missing values in a DataFrame:

- This function is used to fill missing (NaN) values in a DataFrame with a specified value or a method.

python

Copy code

```
                    0
print
```

`merge()` - Merge two DataFrames based on a common column or index:

- This function is used to merge two DataFrames based on a specified column or index.

python

Copy code

```
                                'common_column'
print
```

These examples demonstrate how these Pandas functions are used for common data manipulation tasks, including reading data, displaying information, grouping, filling missing values, and merging DataFrames.

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.
Ans:an use the `set_index` function to re-index the DataFrame with a new index starting from 1 and incrementing by 2 for each row. Here's a Python function that does this:
python

Copy code

```
import       as

def reindex_with_custom_index

                            1       len       2       2


 return
```

```
                    'A'   10  20  30   'B'   40  50  60   'C'   70  80  90
```

```
print "Original DataFrame:"
print

print "\nDataFrame after re-indexing:"
print
```

This function creates a new index starting from 1 and incrementing by 2 using `pd.RangeIndex`. Then, it sets this new index to the DataFrame using `set_index`. The resulting DataFrame (`result_df`) will have the desired custom index.

Please replace the example DataFrame (`df`) with your actual DataFrame when using the function in your code.

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.
Ans:can use the `iloc` indexer to select the first three values in the 'Values' column and then calculate their sum. Here's a Python function that does this:
python

Copy code

```
import        as

def calculate_sum_of_first_three

 if 'Values' in

                        'Values'        3
```

```
                                        sum
print f"Sum of the first three values: {sum_of_first_three}"
else
print "Error: 'Values' column not found in the DataFrame."


                'Values'   10  20  30  40  50
```

Replace the example DataFrame (`df`) with your actual DataFrame when using the function in your code. This function checks if the 'Values' column exists in the DataFrame before proceeding to avoid potential errors.

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.
Ans:can use the `apply` function along with a custom function to count the number of words in each row of the 'Text' column and create a new column 'Word_Count'. Here's a Python function that does this:

python

Copy code

```
import        as

def add_word_count_column

 if 'Text' in

 def count_words

 return len


   'Word_Count'       'Text'
else
print "Error: 'Text' column not found in the DataFrame."
```

```
                    'Text'    'This is a sample text.'   'Another example.'   'Pandas
is great.'



print
```

Replace the example DataFrame (`df`) with your actual DataFrame when using the function in your code. This function checks if the 'Text' column exists in the DataFrame before proceeding to avoid potential errors.

Q5. How are DataFrame.size() and DataFrame.shape() different?
Ans:There is a difference in the usage and purpose between `DataFrame.size` and `DataFrame.shape` in Pandas.

DataFrame.size:

- Type: `DataFrame.size` is an attribute (not a method) of a DataFrame.
- Purpose: It returns the total number of elements in the DataFrame, which is equal to the product of the number of rows and the number of columns.
- Usage:
- python
- Copy code

```
import        as

                    'A'    1  2  3    'B'    4  5  6

print
```

-
- Output: In this example, the output will be `6` because there are 2 columns and 3 rows, making a total of 6 elements.

DataFrame.shape:

- Type: `DataFrame.shape` is a method (not an attribute) of a DataFrame.
- Purpose: It returns a tuple representing the dimensions of the DataFrame, where the first element is the number of rows and the second element is the number of columns.
- Usage:
- python

- Copy code

```
import        as

                'A'   1  2  3   'B'   4  5  6

print
```

-
- Output: In this example, the output will be `(3, 2)` because there are 3 rows and 2 columns.

In summary:

- Use `DataFrame.size` when you want the total number of elements (rows * columns) in the DataFrame.
- Use `DataFrame.shape` when you want to know the dimensions (number of rows and columns) of the DataFrame.

Note that `DataFrame.size` is an attribute, not a method, so you don't use parentheses when accessing it. On the other hand, `DataFrame.shape` is a method, and you need to use parentheses to call it.

Q6. Which function of pandas do we use to read an excel file?
Ans:In Pandas, the function used to read an Excel file is `pd.read_excel()`. This function reads the data from an Excel file and returns it as a DataFrame.

Here's a simple example of how to use `pd.read_excel()`:

python

Copy code

```
import        as

          'your_file.xlsx'



print
```

In this example, the `pd.read_excel()` function is used to read the contents of the Excel file specified by the `file_path` variable, and the result is stored in the DataFrame `df`. You need to provide the correct path or filename of your Excel file as an argument to `pd.read_excel()`.

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.
The username is the part of the email address that appears before the '@' symbol. For example, if the
email address is 'john.doe@example.com', the 'Username' column should contain 'john.doe'. Your
function should extract the username from each email address and store it in the new 'Username'
column.

Ans:can use the `apply` function along with a custom function to extract the username from each email address and create a new 'Username' column. Here's a Python function that does this:

python

Copy code

```
import        as

def extract_username

 if 'Email' in

 def get_username
 return             '@'  0



    'Username'       'Email'
 else
 print "Error: 'Email' column not found in the DataFrame."



                  'Email'   'john.doe@example.com'  'alice.smith@example.com'
'bob.jones@example.com'
```

```
print
```

Replace the example DataFrame (`df`) with your actual DataFrame when using the function in your code. This function checks if the 'Email' column exists in the DataFrame before proceeding to avoid potential errors. The `get_username` function uses the `split('@')` method to split the email address at the '@' symbol and extracts the part before it.

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.
Ans:can use boolean indexing to filter rows in Pandas based on multiple conditions. Here's a Python function that selects rows where the value in column 'A' is greater than 5 and the value in column 'B' is less than 10:
python

Copy code

```
import        as

def select_rows

 if 'A' in            and 'B' in

                      'A'    5        'B'    10


 return
 else
 print "Error: Columns 'A' and 'B' not found in the DataFrame."
```

```
                'A'   3  8  6  2  9
'B'   5  2  9  3  1
'C'   1  7  4  5  2



print
```

Replace the example DataFrame (`df`) with your actual DataFrame when using the function in your code. The `selected_df` DataFrame will contain only the rows that meet the specified conditions.

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.
Ans:can use the `mean()`, `median()`, and `std()` functions in Pandas to calculate the mean, median, and standard deviation of the values in the 'Values' column of a DataFrame. Here's a Python function that does this:
python

Copy code

```
import        as

def calculate_statistics

 if 'Values' in

                'Values'
                  'Values'
                   'Values'


 print f"Mean: {mean_value}"
 print f"Median: {median_value}"
 print f"Standard Deviation: {std_deviation}"
 else
 print "Error: 'Values' column not found in the DataFrame."
```

```
                    'Values'    10  20  30  40  50
```

Replace the example DataFrame (`df`) with your actual DataFrame when using the function in your code. This function checks if the 'Values' column exists in the DataFrame before proceeding to avoid potential errors. The calculated mean, median, and standard deviation are then printed to the console.

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.
Ans:can use the `rolling` function in Pandas to calculate the moving average for a specified window size. Here's a Python function that creates a new column 'MovingAverage' containing the moving average of the 'Sales' column for the past 7 days, including the current day:
python

Copy code

```
import        as

def calculate_moving_average

 if 'Sales' in            and 'Date' in

    'Date'                      'Date'


                       'Date'


                  'Date'


    'MovingAverage'       'Sales'                 7             1
```

```
 return
 else
 print "Error: 'Sales' or 'Date' column not found in the DataFrame."
```

```
 'Date'   '2023-01-01'  '2023-01-02'  '2023-01-03'  '2023-01-04'  '2023-01-05'
'2023-01-06'  '2023-01-07'
 'Sales'   10  15  20  25  30  35  40
```

```
print
```

Replace the example DataFrame (`df`) with your actual DataFrame when using the function in your code. This function checks if the 'Sales' and 'Date' columns exist in the DataFrame before proceeding to avoid potential errors. The resulting DataFrame (`result_df`) will contain the 'MovingAverage' column.

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.
Ans:can use the `dt` accessor in Pandas along with the `strftime` method to extract the weekday name from the 'Date' column. Here's a Python function that creates a new 'Weekday' column in the DataFrame:
python

Copy code

```
import        as

def add_weekday_column

 if 'Date' in

    'Date'                       'Date'


    'Weekday'       'Date'              '%A'

 return
 else
 print "Error: 'Date' column not found in the DataFrame."



                  'Date'   '2023-01-01'  '2023-01-02'  '2023-01-03'
'2023-01-04'  '2023-01-05'




print
```

Replace the example DataFrame (`df`) with your actual DataFrame when using the function in your code. This function checks if the 'Date' column exists in the DataFrame before proceeding to avoid potential errors. The resulting DataFrame (`result_df`) will contain the 'Weekday' column with weekday names.

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'.
Ans:can use boolean indexing to filter rows in Pandas based on a date range. Here's a Python function that selects rows where the 'Date' column is between '2023-01-01' and '2023-01-31':
python

Copy code

```
import        as

def select_rows_in_date_range

 if 'Date' in

    'Date'                       'Date'


                        'Date'     '2023-01-01'        'Date'     '2023-01-31'

 return
 else
 print "Error: 'Date' column not found in the DataFrame."


                   'Date'    '2023-01-01'  '2023-01-15'  '2023-01-25'
'2023-02-05'




print
```

Replace the example DataFrame (`df`) with your actual DataFrame when using the function in your code. This function checks if the 'Date' column exists in the DataFrame before proceeding to avoid potential errors. The resulting DataFrame (`selected_df`) will contain only the rows within the specified date range.

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

Ans:To use the basic functions of Pandas, you need to import the Pandas library itself. The most common convention is to import Pandas using the alias `pd`. Here's the typical import statement:

python

Copy code

```
import        as
```

After importing Pandas this way, you can use the various functions and features provided by Pandas using the `pd` alias. For example, you can create DataFrames, manipulate data, perform statistical operations, and more using the Pandas library.