1-Read all sheets file in excel file?

To read all sheets from an Excel file simultaneously using pandas, you can iterate through each sheet and load it into a list or dictionary.

```
import pandas as pd
# Replace 'file_path.xlsx' with the path to your Excel file
file_path = 'file_path.xlsx'

# Read all sheets from the Excel file into a dictionary
dfs = pd.read_excel(file_path, sheet_name=None)

# Concatenate all dataframes in the dictionary into a single dataframe
combined_df = pd.concat(dfs.values(), ignore_index=True)

# combined_df now contains all data from all sheets in the Excel file
```

pd.read_excel() is used to read all sheets from the Excel file into a dictionary where the keys are the sheet names and the values are the corresponding DataFrames.

pd.concat() is then used to concatenate all DataFrames in the dictionary into a single DataFrame, combined_df. ignore_index=True ensures that the index is reset after concatenation.

Now, combined_df contains all the data from all sheets in the Excel file. If you want to maintain information about which rows originally belonged to each sheet, you can do so by creating a new column in combined_df to store the sheet name before concatenation.

2- How to read Random rows from our data?

To read random rows from your data using pandas, you can use the `sample()` function.

import pandas as pd

```
# Load your data into a pandas DataFrame (replace 'file.csv' with your file)
data = pd.read_csv('file.csv')
# Set the number of random rows you want to read
num_random_rows = 10  # Change this to the number of random rows you want to read
# Read random rows from the data
random_rows = data.sample(n=num_random_rows)
# Now random_rows contains the specified number of random rows from your data
```

pd.read_csv() is used to load your data into a pandas DataFrame. Replace 'file.csv' with the path to your file and adjust any additional parameters as needed.

num_random_rows specifies the number of random rows you want to read from your data.

data.sample(n=num_random_rows) selects a random sample of rows from your DataFrame. Adjust the n parameter to specify the number of random rows you want to read.

3- How to save 2 column and last 3 rows only?

To save only the last 3 rows of a DataFrame and specific columns using pandas, you can use the iloc indexer to select the desired rows and columns, and then save the resulting DataFrame to a file.

```
import pandas as pd
# Sample DataFrame (replace 'data' with your DataFrame)
data = {'A': [1, 2, 3, 4, 5],
        'B': [6, 7, 8, 9, 10],
        'C': [11, 12, 13, 14, 15]}
df = pd.DataFrame(data)
# Select the last 3 rows and specific columns ('A' and 'B')
selected_data = df[['A', 'B']].iloc[-3:]

or
df[['A', 'B']].iloc[-3,0:2]
```

4- How to append to an existing csv file or excel file same sheet?

To append data to an existing CSV file or Excel file on the same sheet using pandas, you can first read the existing file, then concatenate the new data with the existing data, and finally write the combined data back to the file.

```
import pandas as pd
# Sample DataFrame (replace 'data' with your DataFrame)
data = {'A': [1, 2, 3, 4, 5],
        'B': [6, 7, 8, 9, 10],
        'C': [11, 12, 13, 14, 15]}
df = pd.DataFrame(data)
# Select the last 3 rows and specific columns ('A' and 'B')
selected_data = df[['A', 'B']].iloc[-3:]
```

```
or
df[['A', 'B']].iloc[-3,0:2]

import pandas as pd
# Sample DataFrame containing new data to append
new_data = {'A': [6, 7, 8],
            'B': [9, 10, 11]}
df_new = pd.DataFrame(new_data)
# Read existing Excel file
existing_df = pd.read_excel('existing_data.xlsx')
# Append new data to existing data
combined_df = pd.concat([existing_df, df_new], ignore_index=True)
# Write combined data back to the Excel file (same sheet)
combined_df.to_excel('existing_data.xlsx', index=False)
```

pd.concat() is used to concatenate the existing DataFrame (existing_df) with the new DataFrame (df_new).

ignore_index=True ensures that the index is reset after concatenation.

to_csv() or to_excel() is used to write the combined data back to the same file.

5- How to Replace data in specific column from specific data value using panda?

To replace data in a specific column based on a specific value using pandas, you can use boolean indexing combined with the .loc[] accessor.

```
import pandas as pd
# Sample DataFrame
data = {'A': [1, 2, 3, 4, 5],
        'B': ['apple', 'banana', 'orange', 'apple', 'banana']}
df = pd.DataFrame(data)
# Replace all occurrences of 'apple' in column 'B' with 'kiwi'
df.loc[df['B'] == 'apple', 'B'] = 'kiwi'
# Now, 'apple' in column 'B' is replaced with 'kiwi'
print(df)
```

We use boolean indexing df['B'] == 'apple' to create a boolean mask indicating where the value in column 'B' is equal to 'apple'.

We use this mask inside .loc[] to select rows where the condition is True.

We specify the column 'B' to replace values in, and use the assignment operator = to assign the new value 'kiwi'.

After this operation, all occurrences of 'apple' in column 'B' will be replaced with 'kiwi'.

6- How to detect and remove outliers using pandas?

Detecting and removing outliers is an important step in data preprocessing. One common method to detect outliers is using the Z-score

import pandas as pd

```
import pandas as pd
# Sample DataFrame
data = {'A': [1, 2, 3, 4, 5],
        'B': ['apple', 'banana', 'orange', 'apple', 'banana']}
df = pd.DataFrame(data)
# Replace all occurrences of 'apple' in column 'B' with 'kiwi'
df.loc[df['B'] == 'apple', 'B'] = 'kiwi'
# Now, 'apple' in column 'B' is replaced with 'kiwi'
print(df)

# Sample DataFrame (replace 'data' with your DataFrame)
data = {'value': [23, 25, 27, 21, 30, 35, 18, 1000]}
df = pd.DataFrame(data)
# Calculate Z-score
df['Z-score'] = (df['value'] - df['value'].mean()) / df['value'].std()
# Define threshold for outliers (e.g., Z-score > 3 or < -3)
threshold = 3
# Detect outliers
outliers = df[(df['Z-score'] > threshold) | (df['Z-score'] < -threshold)]
# Remove outliers
clean_df = df[(df['Z-score'] <= threshold) & (df['Z-score'] >= -
threshold)].drop(columns='Z-score')
print("Outliers:")
print(outliers)
print("\nDataFrame after removing outliers:")
print(clean_df)
```

OR

```
data = {'Name' : ['youssef','ahmed','mohamed','aly'],'Age' : [1,2,3,1000]}
df = pd.DataFrame(data)
df
```

```
q1 = df['Age'].quantile(0.25)
q3 = df['Age'].quantile(0.75)
iqr = q3 - q1
lb = q1 - 1.5 * iqr
ub = q3 + 1.5 * iqr
outliers = (df['Age'] < lb) | (df['Age'] > ub)
new = df[~outliers]
print(new)
```

Calculate the Z-score for each value in the DataFrame.

Define a threshold for outliers (commonly Z-score > 3 or < -3).

Detect outliers by filtering rows where the absolute Z-score exceeds the threshold.

Remove outliers by filtering rows where the Z-score is within the threshold range.

Drop the Z-score column from the clean DataFrame.

7- how to sort data frame by index, how to sort data frame by specific column using pandas?

To sort a DataFrame by its index or by a specific column, you can use the `sort_index()` or `sort_values()` method

Sorting DataFrame by Index:

```
import pandas as pd
# Create a sample DataFrame
data = {'A': [1, 2, 3],
        'B': [4, 5, 6]}
index = ['row3', 'row1', 'row2']  # Sample index
df = pd.DataFrame(data, index=index)
# Sort DataFrame by index
sorted_df_by_index = df.sort_index()
print("Sorted DataFrame by index:")
print(sorted_df_by_index)
```

Sorting DataFrame by a Specific Column:

```
import pandas as pd
# Create a sample DataFrame
data = {'A': [1, 3, 2],
        'B': [4, 6, 5]}
df = pd.DataFrame(data)
# Sort DataFrame by a specific column (e.g., column 'A')
sorted_df_by_column = df.sort_values(by='A')
print("Sorted DataFrame by column 'A':")
print(sorted_df_by_column)
```

8- How to merge more than 2 data frame at the same time using pandas?

you can merge multiple dataframes at once using the `pd.merge()` function and passing a list of dataframes to it

```
import pandas as pd
# Suppose you have three dataframes df1, df2, and df3
# df1, df2, df3 = pd.DataFrame(...), pd.DataFrame(...), pd.DataFrame(...)
# List of dataframes
dfs = [df1, df2, df3]
# Merge dataframes
merged_df = pd.merge(dfs[0], dfs[1], on='common_column').merge(dfs[2],
on='common_column')
# common_column is the column on which you want to merge the dataframes
# Adjust the 'on' parameter based on the column you want to merge on
# merged_df now contains the result of merging all three dataframes
```

This code first merges the first two dataframes using `pd.merge()` and then merges the resulting dataframe with the third dataframe using `merge()` again. You can continue this pattern for merging more than three dataframes. Adjust the `'on'` parameter based on the column you want to merge on.

Replace `common_column` with the actual column name on which you want to merge your dataframes. This approach iteratively merges each dataframe in the list `dfs` based on the specified column.