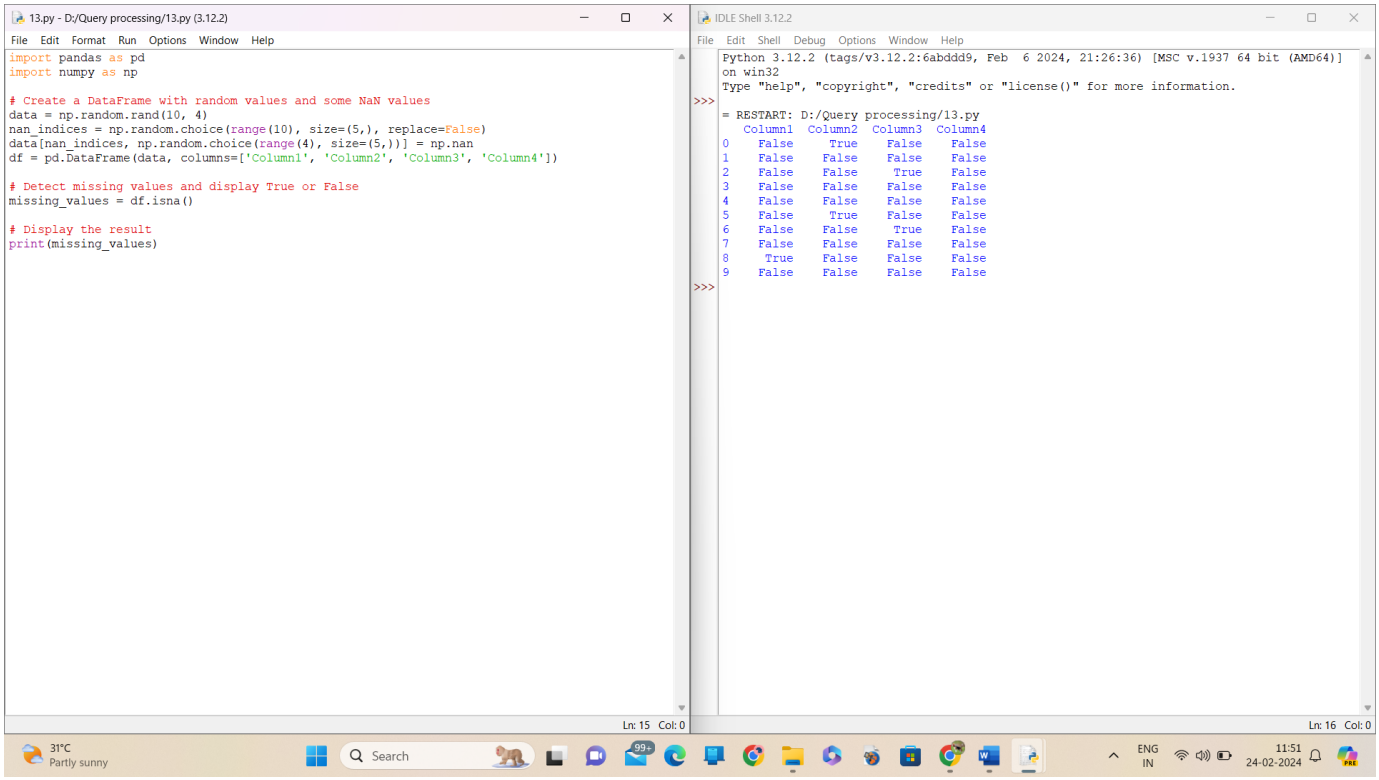
QUERY PROCESSING LAB

13.



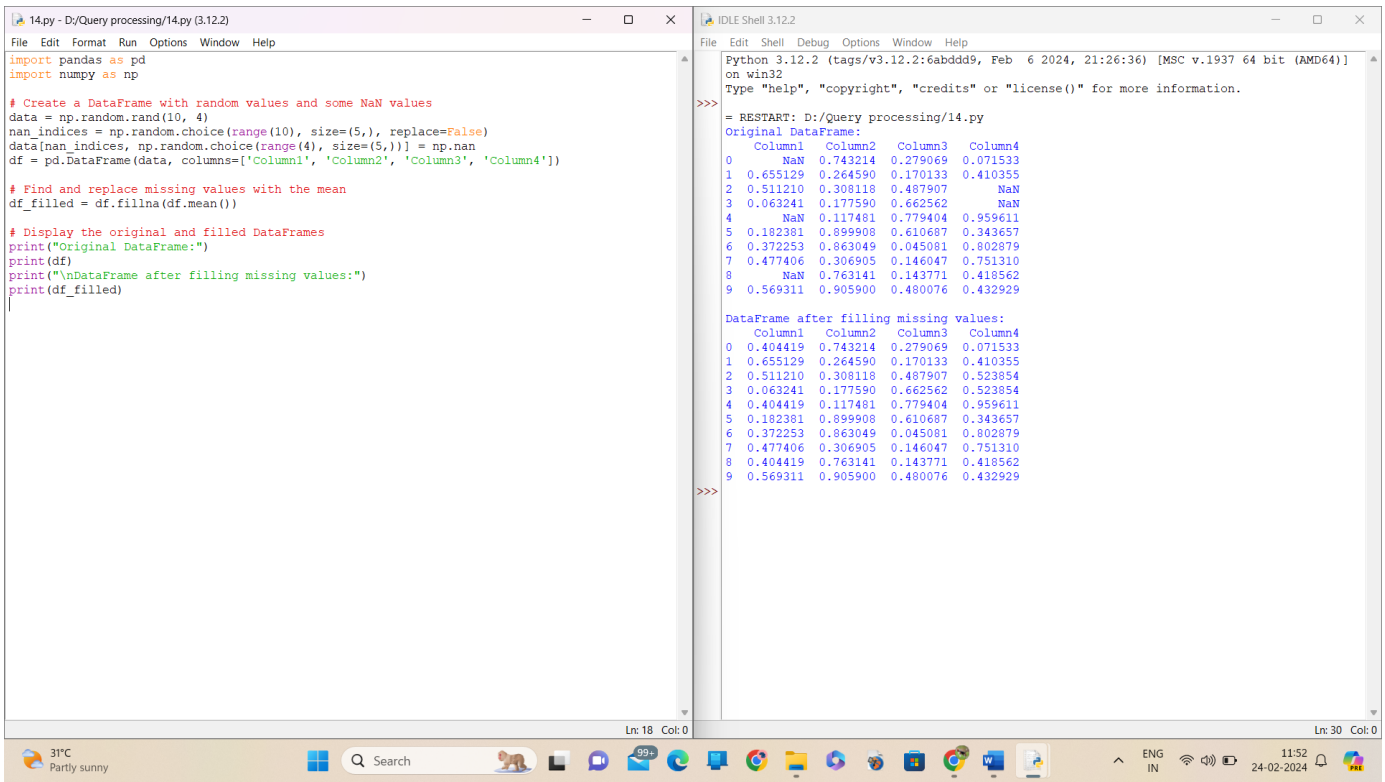
13.py - D:/Query processing/13.py (3.12.2)

```
import pandas as pd
import numpy as np

# Create a DataFrame with random values and some NaN values
data = np.random.rand(10, 4)
nan_indices = np.random.choice(range(10), size=(5,), replace=False)
data[nan_indices, np.random.choice(range(4), size=(5,))] = np.nan
df = pd.DataFrame(data, columns=['Column1', 'Column2', 'Column3', 'Column4'])

# Detect missing values and display True or False
missing_values = df.isna()

# Display the result
print(missing_values)
```

IDLE Shell 3.12.2

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Query processing/13.py
   Column1  Column2  Column3  Column4
0    False     True    False    False
1    False    False    False    False
2    False    False     True    False
3    False    False    False    False
4    False    False    False    False
5    False     True    False    False
6    False    False     True    False
7    False    False    False    False
8     True    False    False    False
9    False    False    False    False
>>>
```

14.



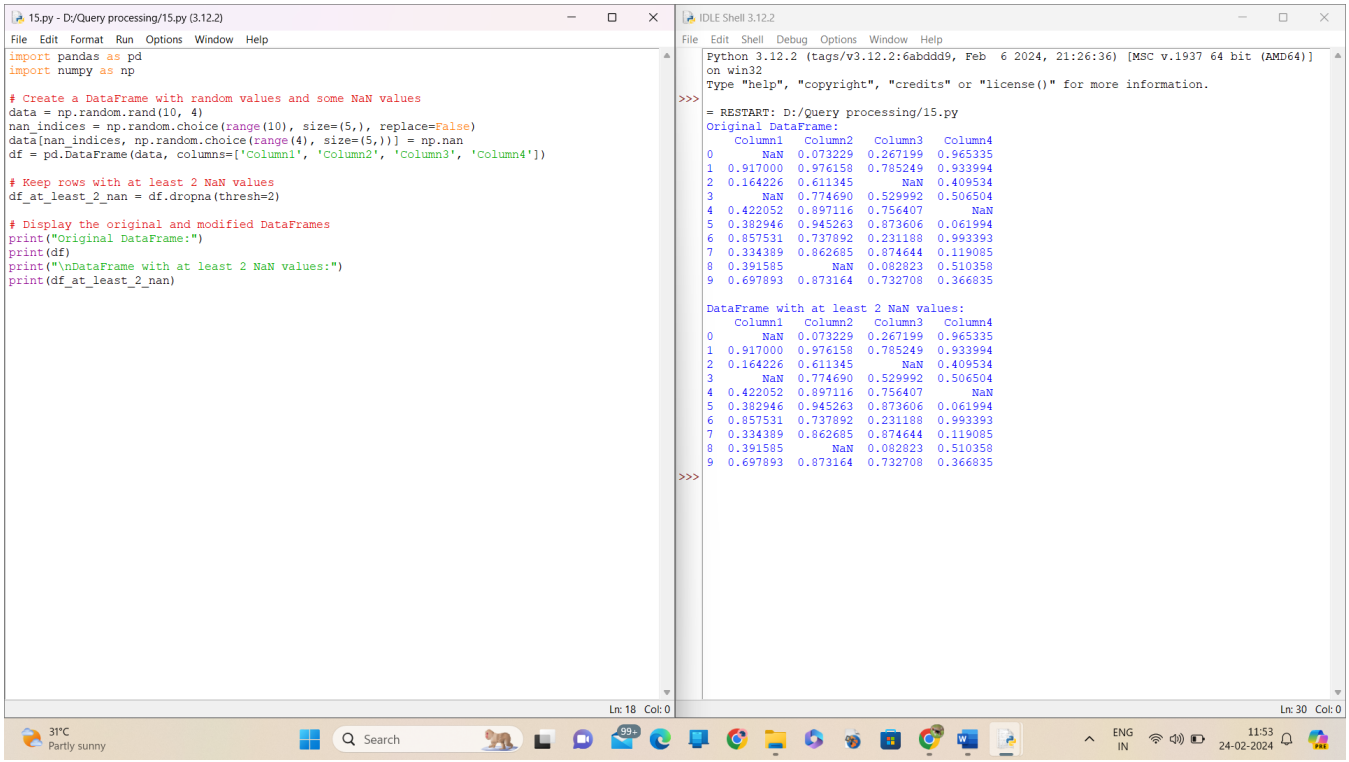
14.py - D:/Query processing/14.py (3.12.2)

```
import pandas as pd
import numpy as np

# Create a DataFrame with random values and some NaN values
data = np.random.rand(10, 4)
nan_indices = np.random.choice(range(10), size=(5,), replace=False)
data[nan_indices, np.random.choice(range(4), size=(5,))] = np.nan
df = pd.DataFrame(data, columns=['Column1', 'Column2', 'Column3', 'Column4'])

# Find and replace missing values with the mean
df_filled = df.fillna(df.mean())

# Display the original and filled DataFrames
print("Original DataFrame:")
print(df)
print("\nDataFrame after filling missing values:")
print(df_filled)
```

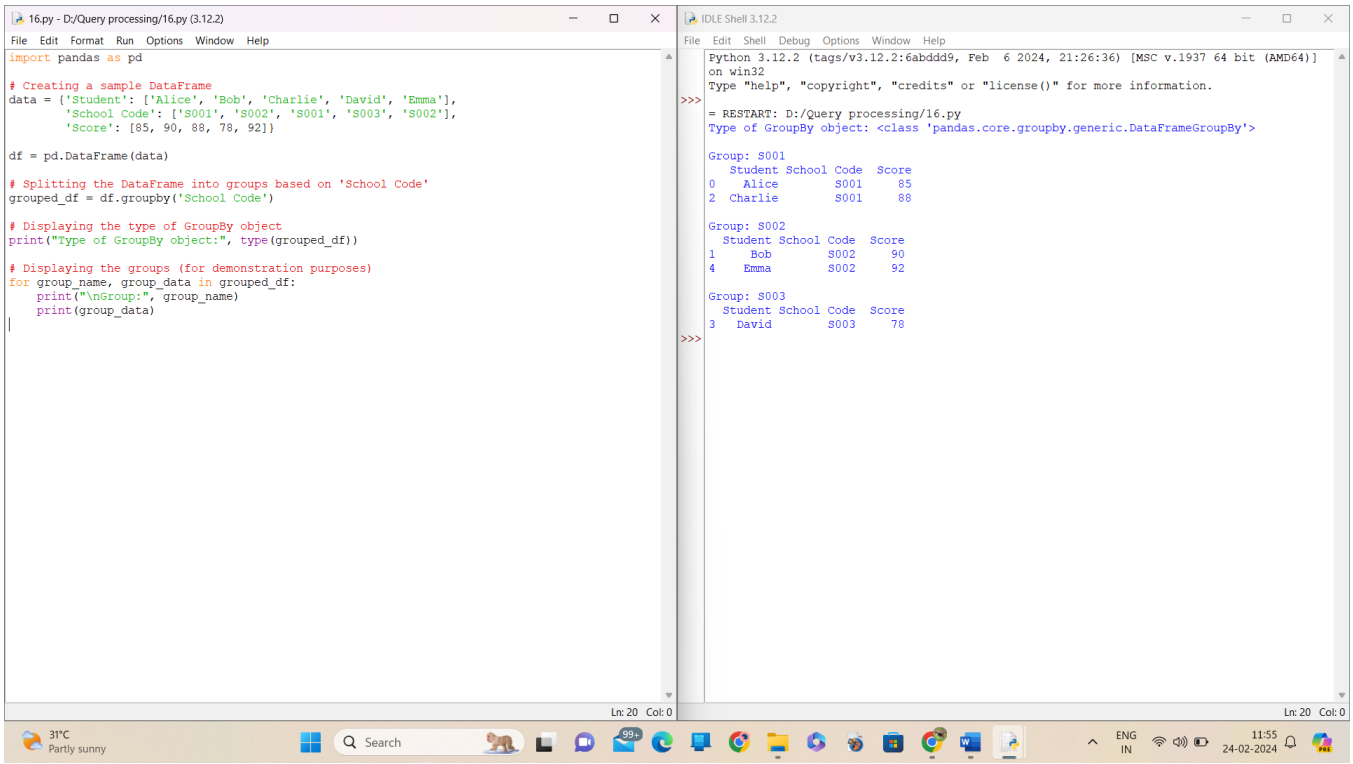
IDLE Shell 3.12.2

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Query processing/14.py
Original DataFrame:
    Column1   Column2   Column3   Column4
0       NaN  0.743214  0.279069  0.071533
1  0.655129  0.264590  0.170133  0.410355
2  0.511210  0.308118  0.487907       NaN
3  0.063241  0.177590  0.662562       NaN
4       NaN  0.117481  0.779404  0.959611
5  0.182381  0.899908  0.610687  0.343657
6  0.372253  0.863049  0.045081  0.802879
7  0.477406  0.306905  0.146047  0.751310
8       NaN  0.763141  0.143771  0.418562
9  0.569311  0.905900  0.480076  0.432929

DataFrame after filling missing values:
    Column1   Column2   Column3   Column4
0  0.404419  0.743214  0.279069  0.071533
1  0.655129  0.264590  0.170133  0.410355
2  0.511210  0.308118  0.487907  0.523854
3  0.063241  0.177590  0.662562  0.523854
4  0.404419  0.117481  0.779404  0.959611
5  0.182381  0.899908  0.610687  0.343657
6  0.372253  0.863049  0.045081  0.802879
7  0.477406  0.306905  0.146047  0.751310
8  0.404419  0.763141  0.143771  0.418562
9  0.569311  0.905900  0.480076  0.432929
>>>
```

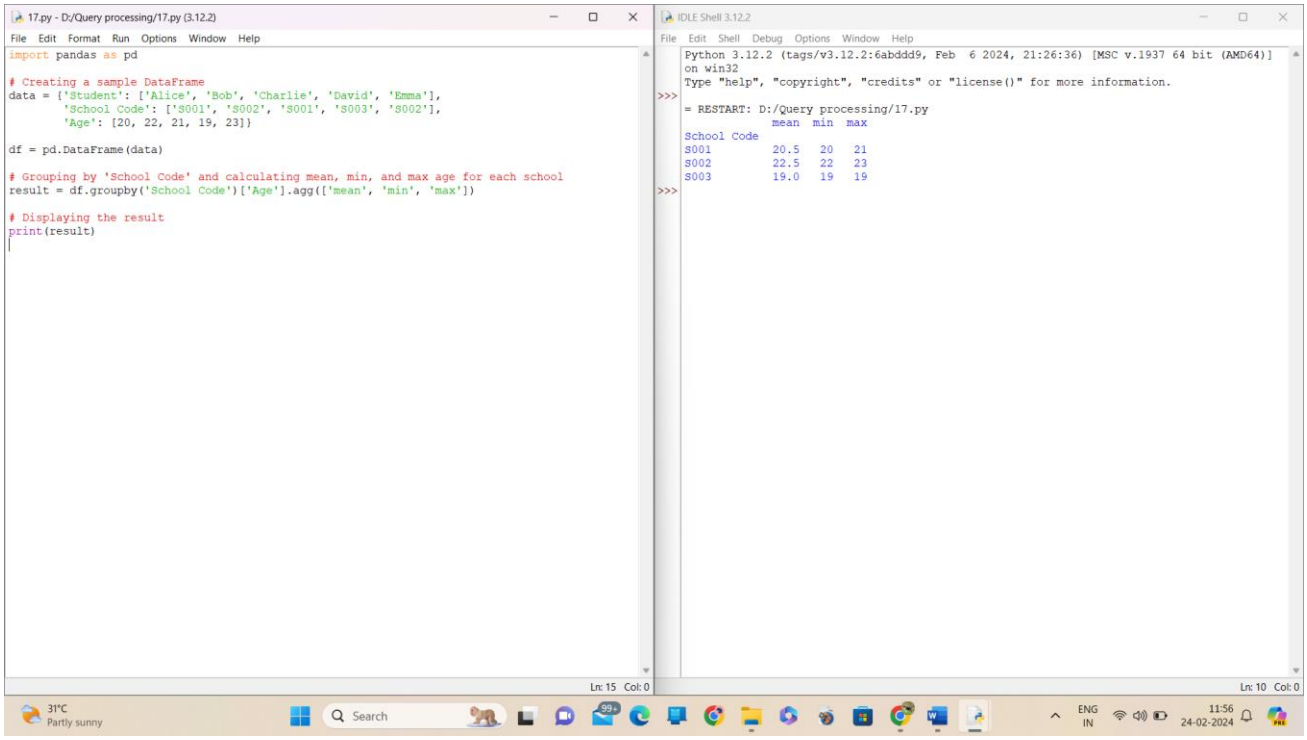
15.


15.py - D:/Query processing/15.py (3.12.2)
File Edit Format Run Options Window Help
import pandas as pd
import numpy as np
# Create a DataFrame with random values and some NaN values
data = np.random.rand(10, 4)
nan_indices = np.random.choice(range(10), size=(5,), replace=False)
data[nan_indices, np.random.choice(range(4), size=(5,))] = np.nan
df = pd.DataFrame(data, columns=['Column1', 'Column2', 'Column3', 'Column4'])
# Keep rows with at least 2 NaN values
df_at_least_2_nan = df.dropna(thresh=2)
# Display the original and modified DataFrames
print("Original DataFrame:")
print(df)
print("\nDataFrame with at least 2 NaN values:")
print(df_at_least_2_nan)
Ln: 18 Col: 0
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Query processing/15.py
Original DataFrame:
Column1 Column2 Column3 Column4
0 NaN 0.073229 0.267199 0.965335
1 0.917000 0.976158 0.785249 0.933994
2 0.164226 0.611345 NaN 0.409534
3 NaN 0.774690 0.529992 0.506504
4 0.422052 0.897116 0.756407 NaN
5 0.382946 0.945263 0.873606 0.061994
6 0.857531 0.737892 0.231188 0.993393
7 0.334389 0.862685 0.874644 0.119085
8 0.391585 NaN 0.082823 0.510358
9 0.697893 0.873164 0.732708 0.366835
DataFrame with at least 2 NaN values:
Column1 Column2 Column3 Column4
0 NaN 0.073229 0.267199 0.965335
1 0.917000 0.976158 0.785249 0.933994
2 0.164226 0.611345 NaN 0.409534
3 NaN 0.774690 0.529992 0.506504
4 0.422052 0.897116 0.756407 NaN
5 0.382946 0.945263 0.873606 0.061994
6 0.857531 0.737892 0.231188 0.993393
7 0.334389 0.862685 0.874644 0.119085
8 0.391585 NaN 0.082823 0.510358
9 0.697893 0.873164 0.732708 0.366835
>>>
Ln: 30 Col: 0
31°C Partly sunny
Search
ENG IN
11:53 24-02-2024

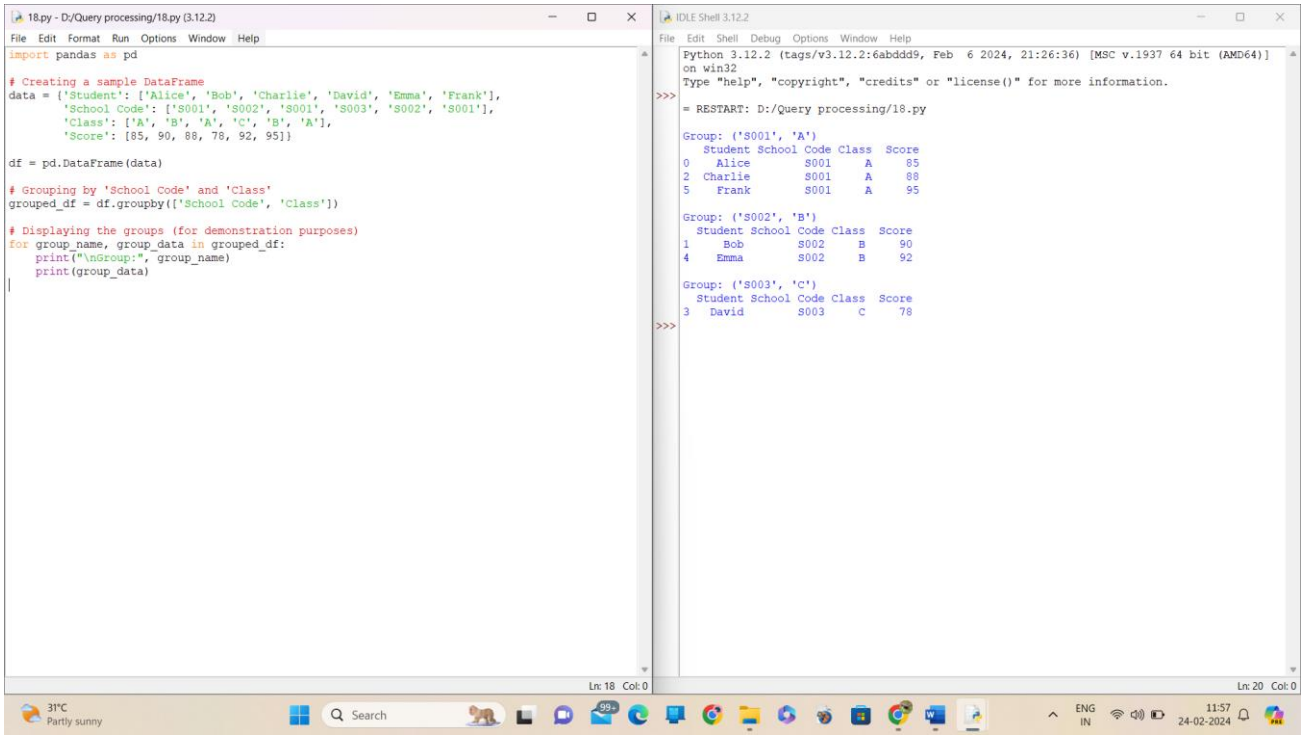
16.


16.py - D:/Query processing/16.py (3.12.2)
File Edit Format Run Options Window Help
import pandas as pd
# Creating a sample DataFrame
data = {'Student': ['Alice', 'Bob', 'Charlie', 'David', 'Emma'],
'School Code': ['S001', 'S002', 'S001', 'S003', 'S002'],
'Score': [85, 90, 88, 78, 92]}
df = pd.DataFrame(data)
# Splitting the DataFrame into groups based on 'School Code'
grouped_df = df.groupby('School Code')
# Displaying the type of GroupBy object
print("Type of GroupBy object:", type(grouped_df))
# Displaying the groups (for demonstration purposes)
for group_name, group_data in grouped_df:
print("\nGroup:", group_name)
print(group_data)
Ln: 20 Col: 0
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Query processing/16.py
Type of GroupBy object: <class 'pandas.core.groupby.generic.DataFrameGroupBy'>
Group: S001
Student School Code Score
0 Alice S001 85
2 Charlie S001 88
Group: S002
Student School Code Score
1 Bob S002 90
4 Emma S002 92
Group: S003
Student School Code Score
3 David S003 78
>>>
Ln: 20 Col: 0
31°C Partly sunny
Search
ENG IN
11:55 24-02-2024

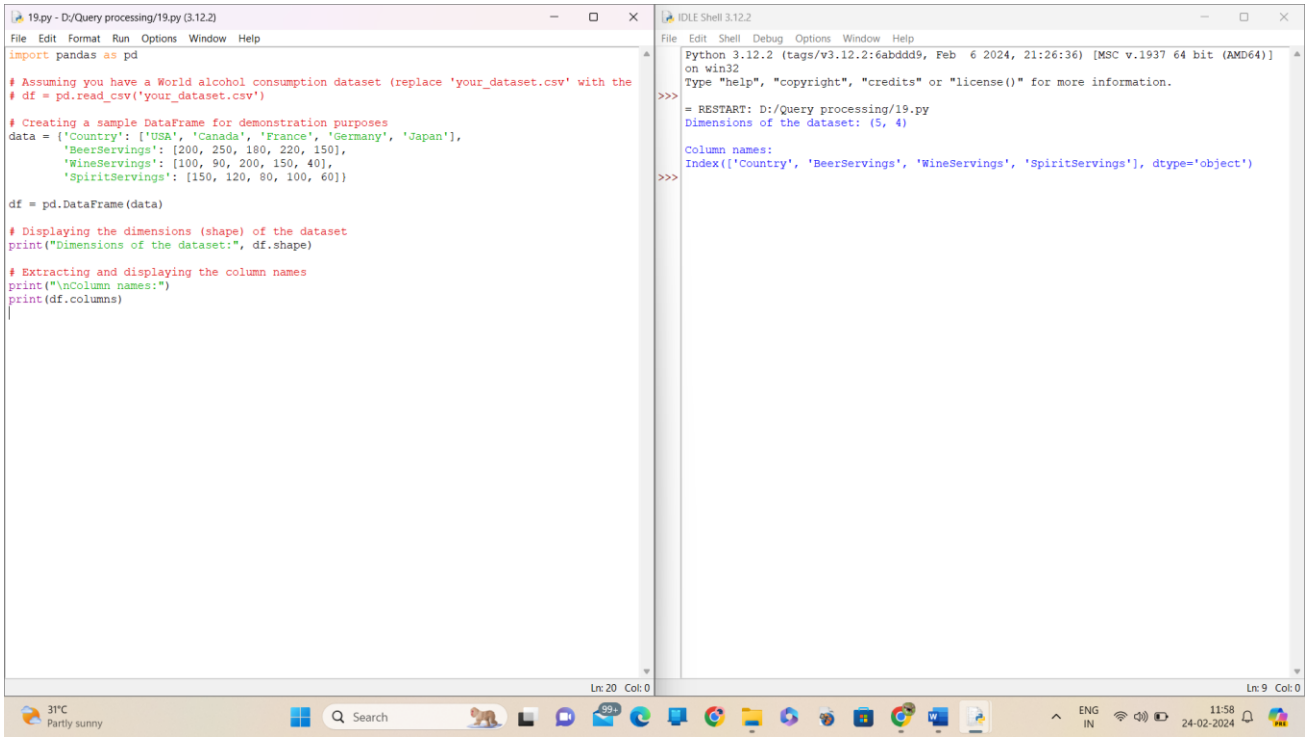
17.


17.py - D:/Query processing/17.py (3.12.2)
File Edit Format Run Options Window Help
import pandas as pd
# Creating a sample DataFrame
data = {'Student': ['Alice', 'Bob', 'Charlie', 'David', 'Emma'],
'School Code': ['S001', 'S002', 'S001', 'S003', 'S002'],
'Age': [20, 22, 21, 19, 23]}
df = pd.DataFrame(data)
# Grouping by 'School Code' and calculating mean, min, and max age for each school
result = df.groupby('School Code')['Age'].agg(['mean', 'min', 'max'])
# Displaying the result
print(result)
Ln: 15 Col: 0
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Query processing/17.py
mean min max
School Code
S001 20.5 20 21
S002 22.5 22 23
S003 19.0 19 19
>>>
Ln: 10 Col: 0
31°C
Partly sunny
Search
ENG
IN
11:56
24-02-2024

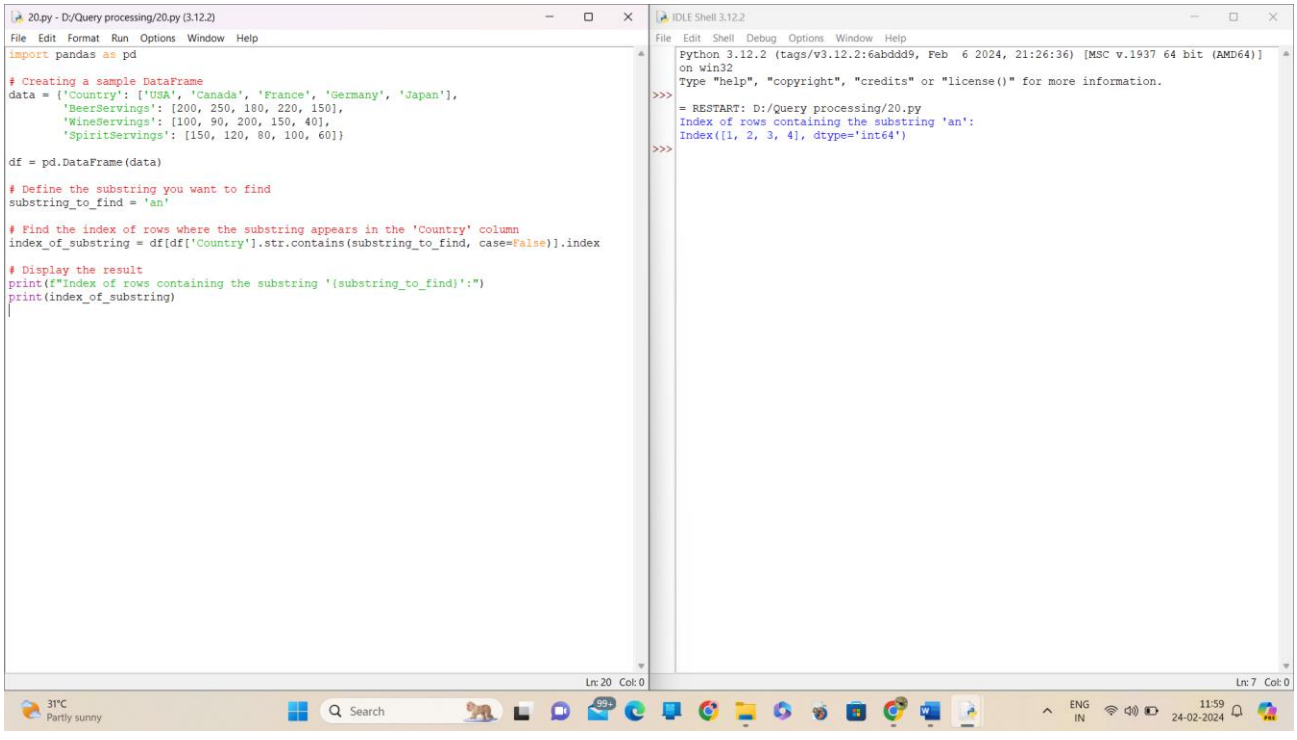
18.


18.py - D:/Query processing/18.py (3.12.2)
File Edit Format Run Options Window Help
import pandas as pd
# Creating a sample DataFrame
data = {'Student': ['Alice', 'Bob', 'Charlie', 'David', 'Emma', 'Frank'],
'School Code': ['S001', 'S002', 'S001', 'S003', 'S002', 'S001'],
'Class': ['A', 'B', 'A', 'C', 'B', 'A'],
'Score': [85, 90, 88, 78, 92, 95]}
df = pd.DataFrame(data)
# Grouping by 'School Code' and 'Class'
grouped_df = df.groupby(['School Code', 'Class'])
# Displaying the groups (for demonstration purposes)
for group_name, group_data in grouped_df:
print("\nGroup:", group_name)
print(group_data)
Ln: 18 Col: 0
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Query processing/18.py
Group: ('S001', 'A')
Student School Code Class Score
0 Alice S001 A 85
2 Charlie S001 A 88
5 Frank S001 A 95
Group: ('S002', 'B')
Student School Code Class Score
1 Bob S002 B 90
4 Emma S002 B 92
Group: ('S003', 'C')
Student School Code Class Score
3 David S003 C 78
>>>
Ln: 20 Col: 0
31°C
Partly sunny
Search
ENG
IN
11:57
24-02-2024

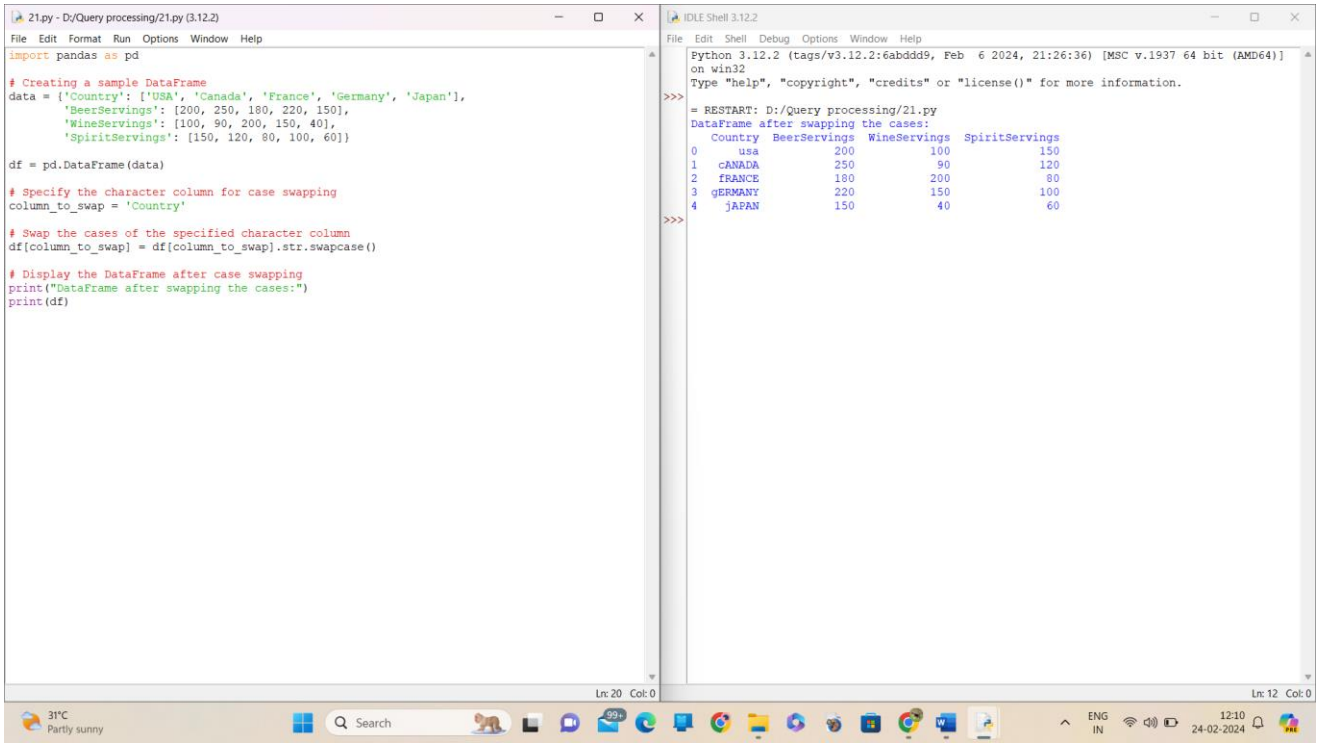
19.


19.py - D:/Query processing/19.py (3.12.2)
File Edit Format Run Options Window Help
import pandas as pd
# Assuming you have a World alcohol consumption dataset (replace 'your_dataset.csv' with the
# df = pd.read_csv('your_dataset.csv')
# Creating a sample DataFrame for demonstration purposes
data = {'Country': ['USA', 'Canada', 'France', 'Germany', 'Japan'],
'BeerServings': [200, 250, 180, 220, 150],
'WineServings': [100, 90, 200, 150, 40],
'SpiritServings': [150, 120, 80, 100, 60]}
df = pd.DataFrame(data)
# Displaying the dimensions (shape) of the dataset
print("Dimensions of the dataset:", df.shape)
# Extracting and displaying the column names
print("\nColumn names:")
print(df.columns)
Ln: 20 Col: 0
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Query processing/19.py
Dimensions of the dataset: (5, 4)
Column names:
Index(['Country', 'BeerServings', 'WineServings', 'SpiritServings'], dtype='object')
>>>
Ln: 9 Col: 0
31°C
Partly sunny
Search
ENG
IN
11:58
24-02-2024

20.


20.py - D:/Query processing/20.py (3.12.2)
File Edit Format Run Options Window Help
import pandas as pd
# Creating a sample DataFrame
data = {'Country': ['USA', 'Canada', 'France', 'Germany', 'Japan'],
'BeerServings': [200, 250, 180, 220, 150],
'WineServings': [100, 90, 200, 150, 40],
'SpiritServings': [150, 120, 80, 100, 60]}
df = pd.DataFrame(data)
# Define the substring you want to find
substring_to_find = 'an'
# Find the index of rows where the substring appears in the 'Country' column
index_of_substring = df[df['Country'].str.contains(substring_to_find, case=False)].index
# Display the result
print(f"Index of rows containing the substring '{substring_to_find}':")
print(index_of_substring)
Ln: 20 Col: 0
IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Query processing/20.py
Index of rows containing the substring 'an':
Index([1, 2, 3, 4], dtype='int64')
>>>
Ln: 7 Col: 0
31°C
Partly sunny
Search
ENG
IN
11:59
24-02-2024

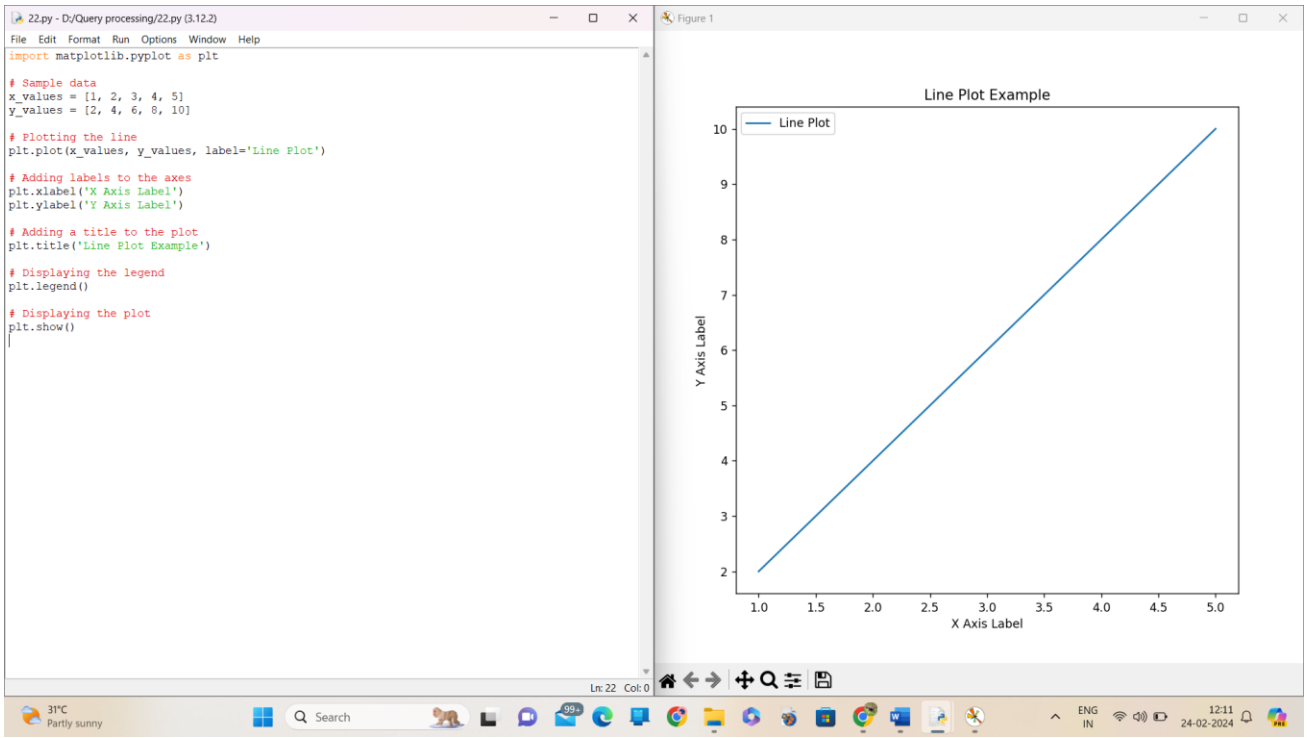
21.



```
import pandas as pd

# Creating a sample DataFrame
data = {'Country': ['USA', 'Canada', 'France', 'Germany', 'Japan'],
        'BeerServings': [200, 250, 180, 220, 150],
        'WineServings': [100, 90, 200, 150, 40],
        'SpiritServings': [150, 120, 80, 100, 60]}

df = pd.DataFrame(data)

# Specify the character column for case swapping
column_to_swap = 'Country'

# Swap the cases of the specified character column
df[column_to_swap] = df[column_to_swap].str.swapcase()

# Display the DataFrame after case swapping
print("DataFrame after swapping the cases:")
print(df)
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:/Query processing/21.py
DataFrame after swapping the cases:
   Country  BeerServings  WineServings  SpiritServings
0      usa           200           100             150
1   cANADA           250            90             120
2   fRANCE           180           200              80
3  gERMANY           220           150             100
4    jAPAN           150            40              60
>>>
```

22.



```
import matplotlib.pyplot as plt

# Sample data
x_values = [1, 2, 3, 4, 5]
y_values = [2, 4, 6, 8, 10]

# Plotting the line
plt.plot(x_values, y_values, label='Line Plot')

# Adding labels to the axes
plt.xlabel('X Axis Label')
plt.ylabel('Y Axis Label')

# Adding a title to the plot
plt.title('Line Plot Example')

# Displaying the legend
plt.legend()

# Displaying the plot
plt.show()
```