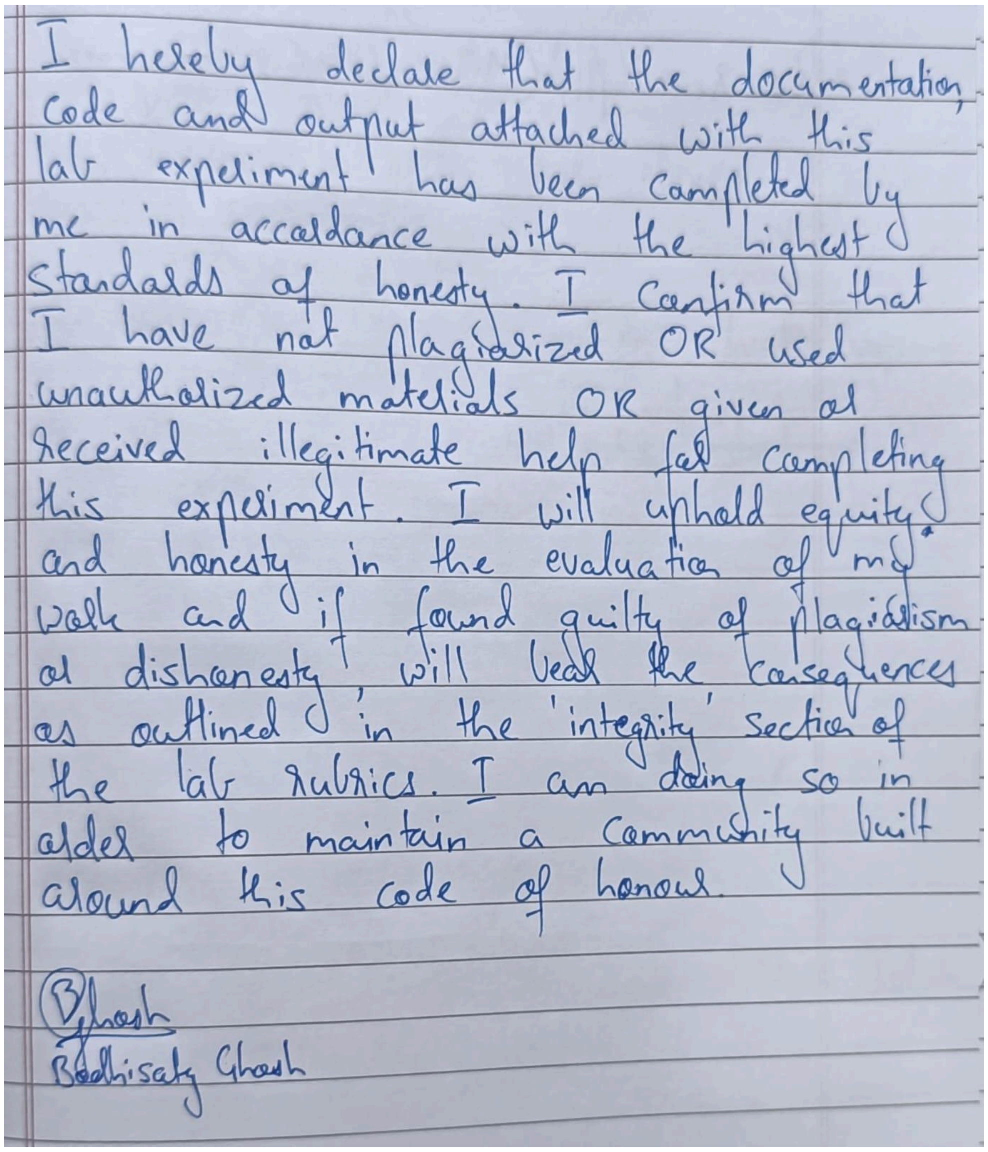
Name	**Bodhisatya Ghosh**
UID no.	**2021700026**

Experiment 3	
HONOUR PLEDGE	 I hereby declare that the documentation, code and output attached with this lab experiment has been completed by me in accordance with the highest standards of honesty. I confirm that I have not plagiarized OR used unauthorized materials OR given or received illegitimate help for completing this experiment. I will uphold equity and honesty in the evaluation of my work and if found guilty of plagiarism or dishonesty, will bear the consequences as outlined in the 'integrity' section of the lab rubrics. I am doing so in order to maintain a community built around this code of honour. B Ghosh Bodhisaty Ghosh
PROBLEM STATEMENT :	**Data Integration and Reshaping:** **1. Merge two or more data frames based on a common key.** ● Create a new pandas dataframe with containing 20 records and 5 attributes.

- One attribute should compulsorily be a categorical variable, which is common with a categorical attribute from the CSV dataset that has been used earlier by you.
- The other 4 attributes can be generated on reasonable assumptions.
- Merge these 2 datasets on the common key

2. Concatenate multiple Data Frames vertically or horizontally

- Create 5 new rows of the same schema as the original csv dataframe.
- Use a new categorical value for the common key attribute.
- Concatenate this horizontally with the existing dataframe
- Similarly, execute a vertical concatenation with a mock dataframe.

3. Pivot a Data Frame from long to wide format or vice versa

- Add reasoning for using pivot. Explain with a relevant example how pivot operation is useful in data analysis

4. Stack and unstack columns or levels in the Data Frame

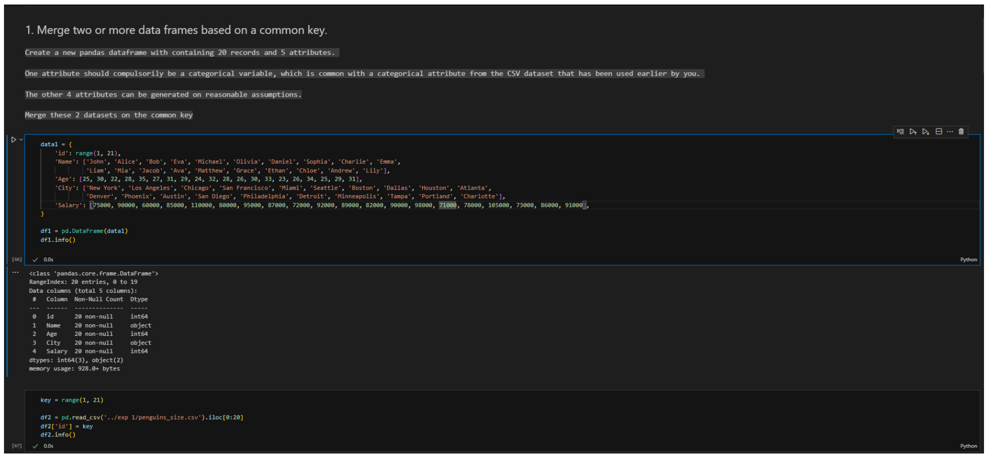
- Reason about the use and application of stacking and unstacking with the help of the current dataframe or another example.

5. Data Wrangling

- Experiment with other techniques in data wrangling to convert and reshape your dataframe into its final state which can be used for analysis

THEORY:

PROGRAM:



1. Merge two or more data frames based on a common key.

```
Create a new pandas dataframe with containing 20 records and 5 attributes.

One attribute should compulsorily be a categorical variable, which is common with a categorical attribute from the CSV dataset that has been used earlier by you.

The other 4 attributes can be generated on reasonable assumptions.

Merge these 2 datasets on the common key
```

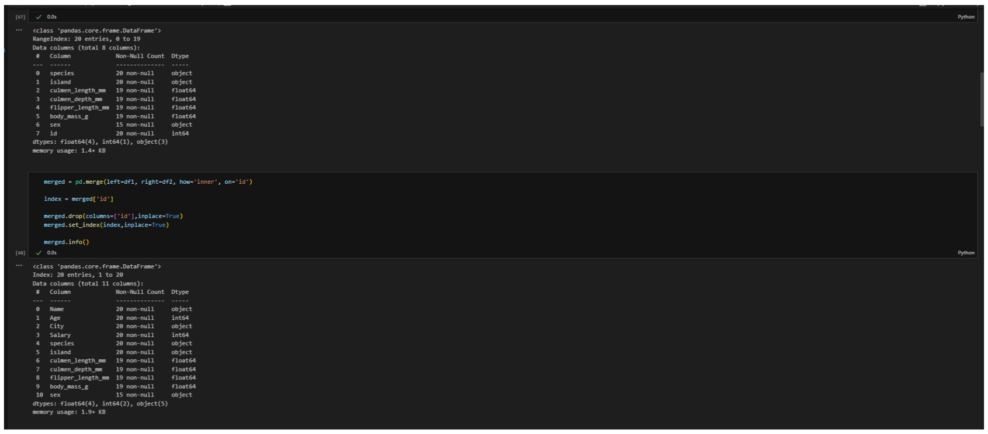
```python
data1 = {
    'id': range(1, 21),
    'Name': ['John', 'Alice', 'Bob', 'Eva', 'Michael', 'Olivia', 'Daniel', 'Sophia', 'Charlie', 'Emma',
             'Liam', 'Mia', 'Jacob', 'Ava', 'Matthew', 'Grace', 'Ethan', 'Chloe', 'Andrew', 'Lily'],
    'Age': [25, 30, 22, 28, 35, 27, 31, 29, 24, 32, 28, 26, 30, 33, 23, 26, 34, 25, 29, 31],
    'City': ['New York', 'Los Angeles', 'Chicago', 'San Francisco', 'Miami', 'Seattle', 'Boston', 'Dallas', 'Houston', 'Atlanta',
             'Denver', 'Phoenix', 'Austin', 'San Diego', 'Philadelphia', 'Detroit', 'Minneapolis', 'Tampa', 'Portland', 'Charlotte'],
    'Salary': [75000, 90000, 60000, 85000, 110000, 80000, 95000, 87000, 72000, 92000, 89000, 82000, 90000, 98000, 71000, 78000, 105000, 73000, 86000, 91000],
}

df1 = pd.DataFrame(data1)
df1.info()
```

[66] ✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      20 non-null     int64 
 1   Name    20 non-null     object
 2   Age     20 non-null     int64 
 3   City    20 non-null     object
 4   Salary  20 non-null     int64 
dtypes: int64(3), object(2)
memory usage: 928.0+ bytes
```

```python
key = range(1, 21)

df2 = pd.read_csv('../exp 1/penguins_size.csv').iloc[0:20]
df2['id'] = key
df2.info()
```

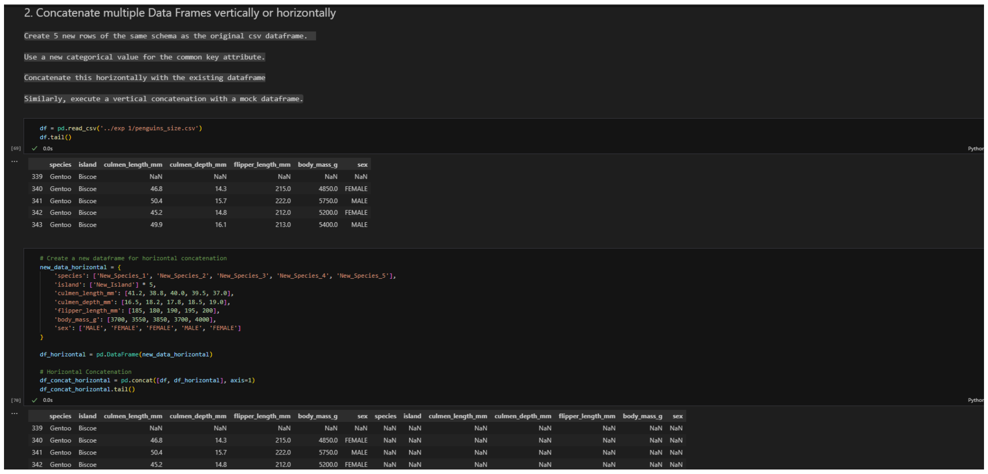
[67] ✓ 0.0s



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            20 non-null     object 
 1   island             20 non-null     object 
 2   culmen_length_mm   19 non-null     float64
 3   culmen_depth_mm    19 non-null     float64
 4   flipper_length_mm  19 non-null     float64
 5   body_mass_g        19 non-null     float64
 6   sex                15 non-null     object 
 7   id                 20 non-null     int64  
dtypes: float64(4), int64(1), object(3)
memory usage: 1.4+ KB
```

```python
merged = pd.merge(left=df1, right=df2, how='inner', on='id')

index = merged['id']

merged.drop(columns=['id'],inplace=True)
merged.set_index(index,inplace=True)

merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 20 entries, 1 to 20
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               20 non-null     object 
 1   Age                20 non-null     int64  
 2   City               20 non-null     object 
 3   Salary             20 non-null     int64  
 4   species            20 non-null     object 
 5   island             20 non-null     object 
 6   culmen_length_mm   19 non-null     float64
 7   culmen_depth_mm    19 non-null     float64
 8   flipper_length_mm  19 non-null     float64
 9   body_mass_g        19 non-null     float64
 10  sex                15 non-null     object 
dtypes: float64(4), int64(2), object(5)
memory usage: 1.9+ KB
```



2. Concatenate multiple Data Frames vertically or horizontally

`Create 5 new rows of the same schema as the original csv dataframe.`

`Use a new categorical value for the common key attribute.`

`Concatenate this horizontally with the existing dataframe`

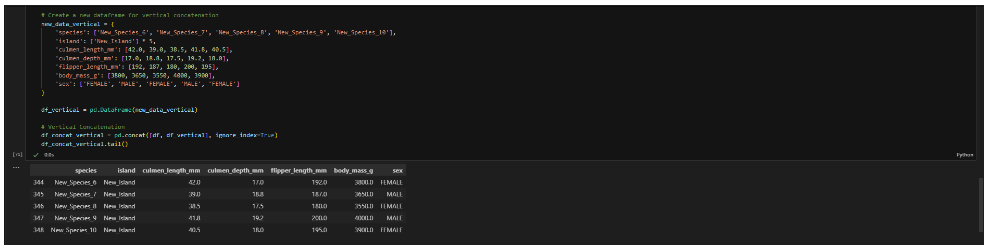
`Similarly, execute a vertical concatenation with a mock dataframe.`

```python
df = pd.read_csv('../exp 1/penguins_size.csv')
df.tail()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	FEMALE
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	MALE
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	FEMALE
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	MALE

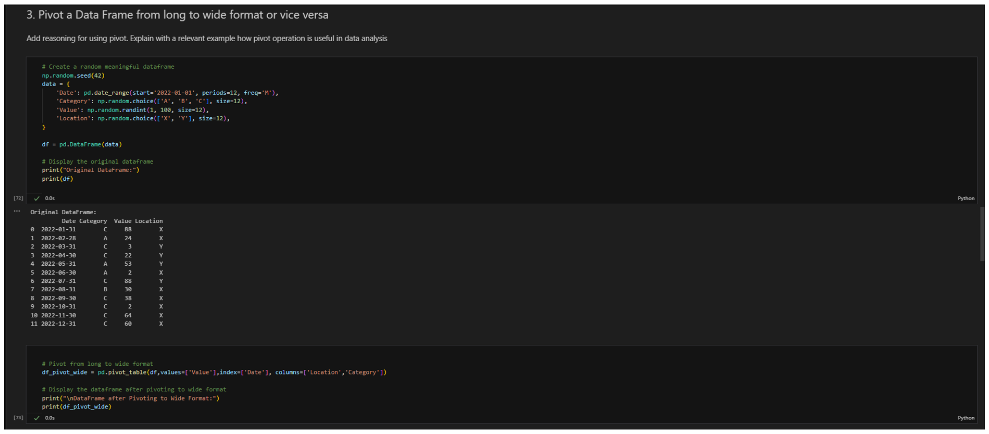
```python
# Create a new dataframe for horizontal concatenation
new_data_horizontal = {
    'species': ['New_Species_1', 'New_Species_2', 'New_Species_3', 'New_Species_4', 'New_Species_5'],
    'island': ['New_Island'] * 5,
    'culmen_length_mm': [41.2, 38.8, 40.0, 39.5, 37.0],
    'culmen_depth_mm': [16.5, 18.2, 17.8, 18.5, 19.0],
    'flipper_length_mm': [185, 180, 190, 195, 200],
    'body_mass_g': [3700, 3550, 3850, 3700, 4000],
    'sex': ['MALE', 'FEMALE', 'FEMALE', 'MALE', 'FEMALE']
}

df_horizontal = pd.DataFrame(new_data_horizontal)

# Horizontal Concatenation
df_concat_horizontal = pd.concat([df, df_horizontal], axis=1)
df_concat_horizontal.tail()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	FEMALE	NaN	NaN	NaN	NaN	NaN	NaN	NaN
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	MALE	NaN	NaN	NaN	NaN	NaN	NaN	NaN
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	FEMALE	NaN	NaN	NaN	NaN	NaN	NaN	NaN



```python
# Create a new dataframe for vertical concatenation
new_data_vertical = {
    'species': ['New_Species_6', 'New_Species_7', 'New_Species_8', 'New_Species_9', 'New_Species_10'],
    'island': ['New_Island'] * 5,
    'culmen_length_mm': [42.0, 39.0, 38.5, 41.8, 40.5],
    'culmen_depth_mm': [17.0, 18.8, 17.5, 19.2, 18.0],
    'flipper_length_mm': [192, 187, 180, 200, 195],
    'body_mass_g': [3800, 3650, 3550, 4000, 3900],
    'sex': ['FEMALE', 'MALE', 'FEMALE', 'MALE', 'FEMALE']
}

df_vertical = pd.DataFrame(new_data_vertical)

# Vertical Concatenation
df_concat_vertical = pd.concat([df, df_vertical], ignore_index=True)
df_concat_vertical.tail()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
344	New_Species_6	New_Island	42.0	17.0	192.0	3800.0	FEMALE
345	New_Species_7	New_Island	39.0	18.8	187.0	3650.0	MALE
346	New_Species_8	New_Island	38.5	17.5	180.0	3550.0	FEMALE
347	New_Species_9	New_Island	41.8	19.2	200.0	4000.0	MALE
348	New_Species_10	New_Island	40.5	18.0	195.0	3900.0	FEMALE



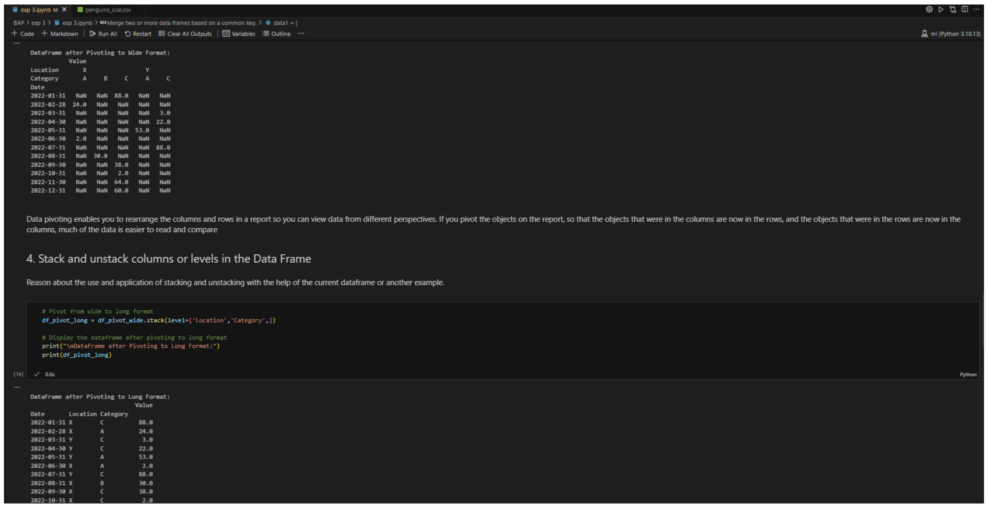
3. Pivot a Data Frame from long to wide format or vice versa

Add reasoning for using pivot. Explain with a relevant example how pivot operation is useful in data analysis

```python
# Create a random meaningful dataframe
np.random.seed(42)
data = {
    'Date': pd.date_range(start='2022-01-01', periods=12, freq='M'),
    'Category': np.random.choice(['A', 'B', 'C'], size=12),
    'Value': np.random.randint(1, 100, size=12),
    'Location': np.random.choice(['X', 'Y'], size=12),
}

df = pd.DataFrame(data)

# Display the original dataframe
print("Original DataFrame:")
print(df)
```

```
Original DataFrame:
         Date Category  Value Location
0  2022-01-31        C     88        X
1  2022-02-28        A     24        X
2  2022-03-31        C      3        Y
3  2022-04-30        C     22        Y
4  2022-05-31        A     53        Y
5  2022-06-30        A      2        X
6  2022-07-31        C     88        Y
7  2022-08-31        B     30        X
8  2022-09-30        C     38        X
9  2022-10-31        C      2        X
10 2022-11-30        C     64        X
11 2022-12-31        C     60        X
```

```python
# Pivot from long to wide format
df_pivot_wide = pd.pivot_table(df,values=['Value'],index=['Date'], columns=['Location','Category'])

# Display the dataframe after pivoting to wide format
print("\nDataFrame after Pivoting to Wide Format:")
print(df_pivot_wide)
```



```
DataFrame after Pivoting to Wide Format:
           Value
Location       X                 Y
Category       A     B     C     A     C
Date
2022-01-31   NaN   NaN  88.0   NaN   NaN
2022-02-28  24.0   NaN   NaN   NaN   NaN
2022-03-31   NaN   NaN   NaN   NaN   3.0
2022-04-30   NaN   NaN   NaN   NaN  22.0
2022-05-31   NaN   NaN   NaN  53.0   NaN
2022-06-30   2.0   NaN   NaN   NaN   NaN
2022-07-31   NaN   NaN   NaN   NaN  88.0
2022-08-31   NaN  30.0   NaN   NaN   NaN
2022-09-30   NaN   NaN  38.0   NaN   NaN
2022-10-31   NaN   NaN   2.0   NaN   NaN
2022-11-30   NaN   NaN  64.0   NaN   NaN
2022-12-31   NaN   NaN  60.0   NaN   NaN
```

Data pivoting enables you to rearrange the columns and rows in a report so you can view data from different perspectives. If you pivot the objects on the report, so that the objects that were in the columns are now in the rows, and the objects that were in the rows are now in the columns, much of the data is easier to read and compare

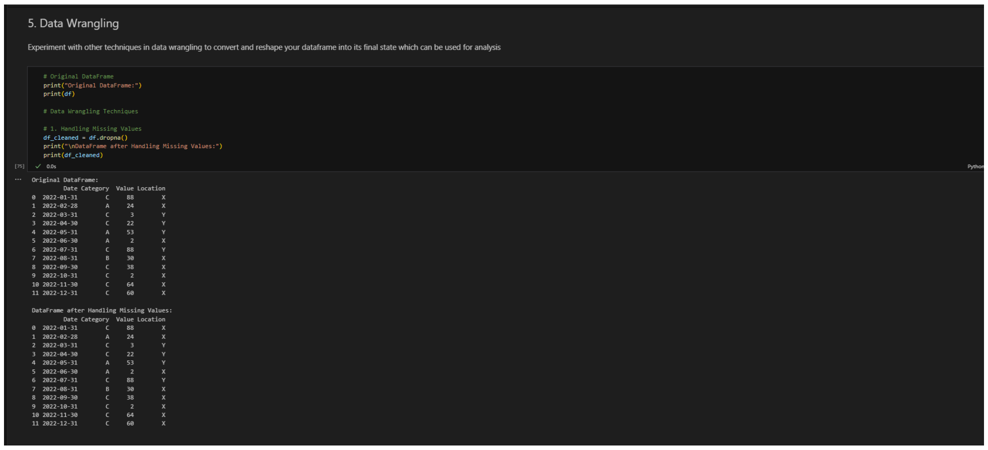
4. Stack and unstack columns or levels in the Data Frame

Reason about the use and application of stacking and unstacking with the help of the current dataframe or another example.

```python
# Pivot from wide to long format
df_pivot_long = df_pivot_wide.stack(level=['Location','Category',])

# Display the dataframe after pivoting to long format
print("\nDataFrame after Pivoting to Long Format:")
print(df_pivot_long)
```

[74] ✓ 0.0s Python

```
DataFrame after Pivoting to Long Format:
                             Value
Date       Location Category
2022-01-31 X        C         88.0
2022-02-28 X        A         24.0
2022-03-31 Y        C          3.0
2022-04-30 Y        C         22.0
2022-05-31 Y        A         53.0
2022-06-30 X        A          2.0
2022-07-31 Y        C         88.0
2022-08-31 X        B         30.0
2022-09-30 X        C         38.0
2022-10-31 X        C          2.0
```



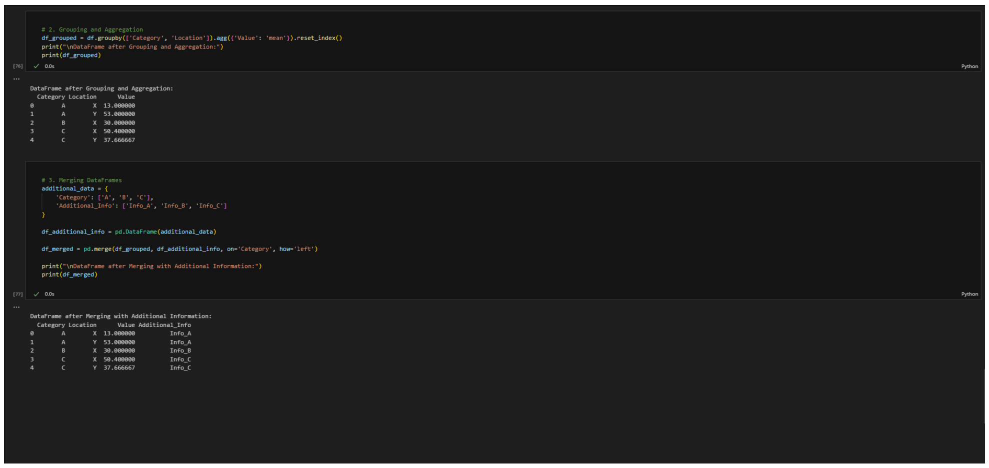
5. Data Wrangling

Experiment with other techniques in data wrangling to convert and reshape your dataframe into its final state which can be used for analysis

```python
# Original DataFrame
print("Original DataFrame:")
print(df)

# Data Wrangling Techniques

# 1. Handling Missing Values
df_cleaned = df.dropna()
print("\nDataFrame after Handling Missing Values:")
print(df_cleaned)
```

[75] ✓ 0.0s Python

```
Original DataFrame:
         Date Category  Value Location
0  2022-01-31        C     88        X
1  2022-02-28        A     24        X
2  2022-03-31        C      3        Y
3  2022-04-30        C     22        Y
4  2022-05-31        A     53        Y
5  2022-06-30        A      2        X
6  2022-07-31        C     88        Y
7  2022-08-31        B     30        X
8  2022-09-30        C     38        X
9  2022-10-31        C      2        X
10 2022-11-30        C     64        X
11 2022-12-31        C     60        X

DataFrame after Handling Missing Values:
         Date Category  Value Location
0  2022-01-31        C     88        X
1  2022-02-28        A     24        X
2  2022-03-31        C      3        Y
3  2022-04-30        C     22        Y
4  2022-05-31        A     53        Y
5  2022-06-30        A      2        X
6  2022-07-31        C     88        Y
7  2022-08-31        B     30        X
8  2022-09-30        C     38        X
9  2022-10-31        C      2        X
10 2022-11-30        C     64        X
11 2022-12-31        C     60        X
```



```python
# 2. Grouping and Aggregation
df_grouped = df.groupby(['Category', 'Location']).agg({'Value': 'mean'}).reset_index()
print("\nDataFrame after Grouping and Aggregation:")
print(df_grouped)
```

[76] ✓ 0.0s Python

```
DataFrame after Grouping and Aggregation:
  Category Location      Value
0        A        X  13.000000
1        A        Y  53.000000
2        B        X  30.000000
3        C        X  50.400000
4        C        Y  37.666667
```

```python
# 3. Merging DataFrames
additional_data = {
    'Category': ['A', 'B', 'C'],
    'Additional_Info': ['Info_A', 'Info_B', 'Info_C']
}

df_additional_info = pd.DataFrame(additional_data)

df_merged = pd.merge(df_grouped, df_additional_info, on='Category', how='left')

print("\nDataFrame after Merging with Additional Information:")
print(df_merged)
```

[77] ✓ 0.0s Python

```
DataFrame after Merging with Additional Information:
  Category Location      Value Additional_Info
0        A        X  13.000000          Info_A
1        A        Y  53.000000          Info_A
2        B        X  30.000000          Info_B
3        C        X  50.400000          Info_C
4        C        Y  37.666667          Info_C
```

RESULT:	
References:	pandas.merge — pandas 2.2.0 documentation (pydata.org) Pivoting data (microstrategy.com) Reshaping and pivot tables — pandas 2.2.0 documentation (pydata.org)
CONCLUSION: In conclusion, the above used data manipulation techniques, including merging, concatenating, pivoting, stacking, and data wrangling have been performed successfully	