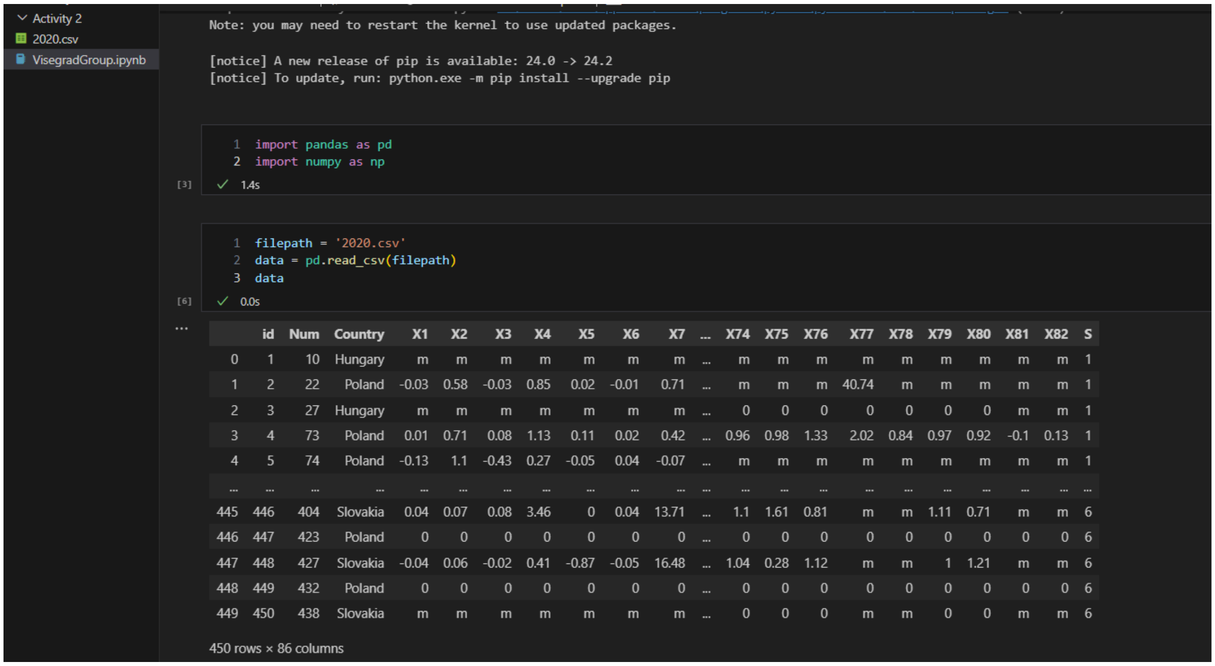
PSMDSRC103 – INTRODUCTION TO NUMPY AND PANDAS
Submitted by: BASAL, RAFFY
Performed on: October 8, 2024
Submitted on: October 8, 2024

Submitted to: Engr. Roman M. Richard



```
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

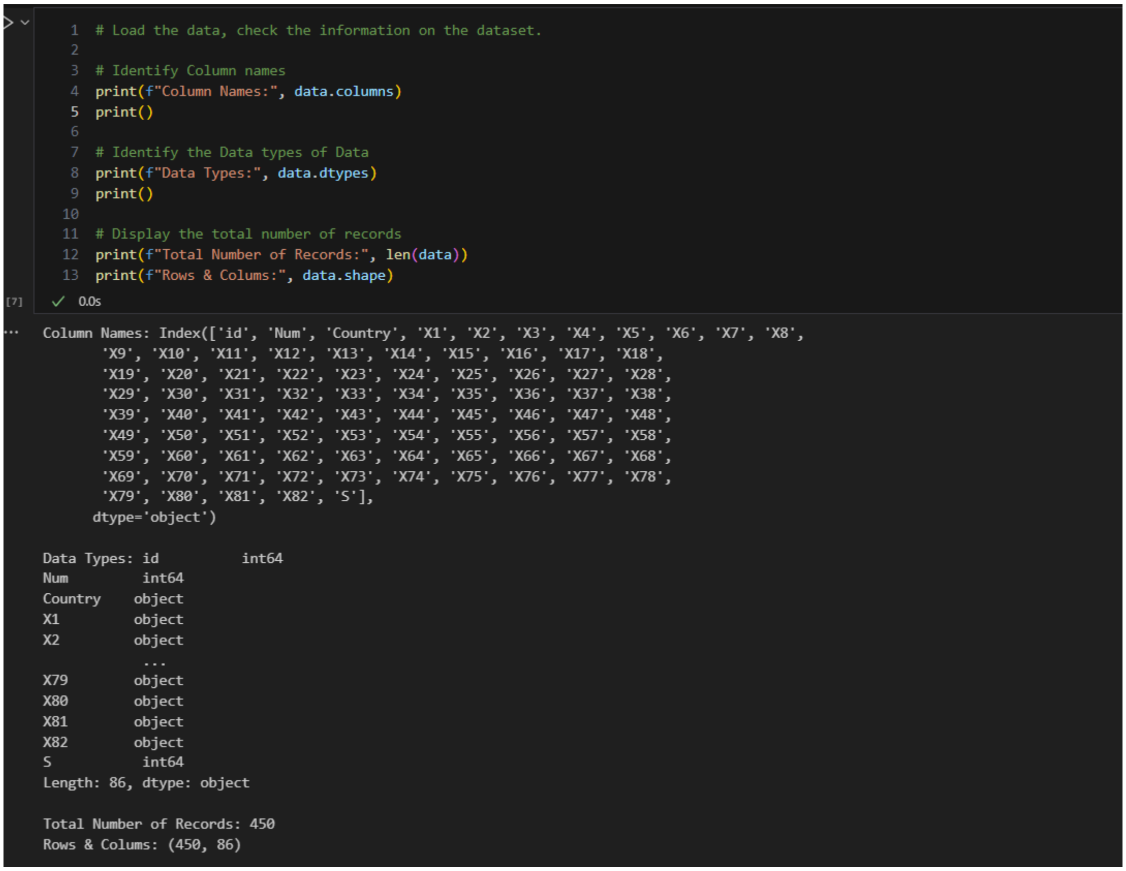
```python
import pandas as pd
import numpy as np
```

```python
filepath = '2020.csv'
data = pd.read_csv(filepath)
data
```

	id	Num	Country	X1	X2	X3	X4	X5	X6	X7	...	X74	X75	X76	X77	X78	X79	X80	X81	X82	S
0	1	10	Hungary	m	m	m	m	m	m	m	...	m	m	m	m	m	m	m	m	m	1
1	2	22	Poland	-0.03	0.58	-0.03	0.85	0.02	-0.01	0.71	...	m	m	m	40.74	m	m	m	m	m	1
2	3	27	Hungary	m	m	m	m	m	m	m	...	0	0	0	0	0	0	0	m	m	1
3	4	73	Poland	0.01	0.71	0.08	1.13	0.11	0.02	0.42	...	0.96	0.98	1.33	2.02	0.84	0.97	0.92	-0.1	0.13	1
4	5	74	Poland	-0.13	1.1	-0.43	0.27	-0.05	0.04	-0.07	...	m	m	m	m	m	m	m	m	m	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
445	446	404	Slovakia	0.04	0.07	0.08	3.46	0	0.04	13.71	...	1.1	1.61	0.81	m	m	1.11	0.71	m	m	6
446	447	423	Poland	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	6
447	448	427	Slovakia	-0.04	0.06	-0.02	0.41	-0.87	-0.05	16.48	...	1.04	0.28	1.12	m	m	1	1.21	m	m	6
448	449	432	Poland	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	6
449	450	438	Slovakia	m	m	m	m	m	m	m	...	0	0	0	m	m	0	0	m	m	6

450 rows × 86 columns

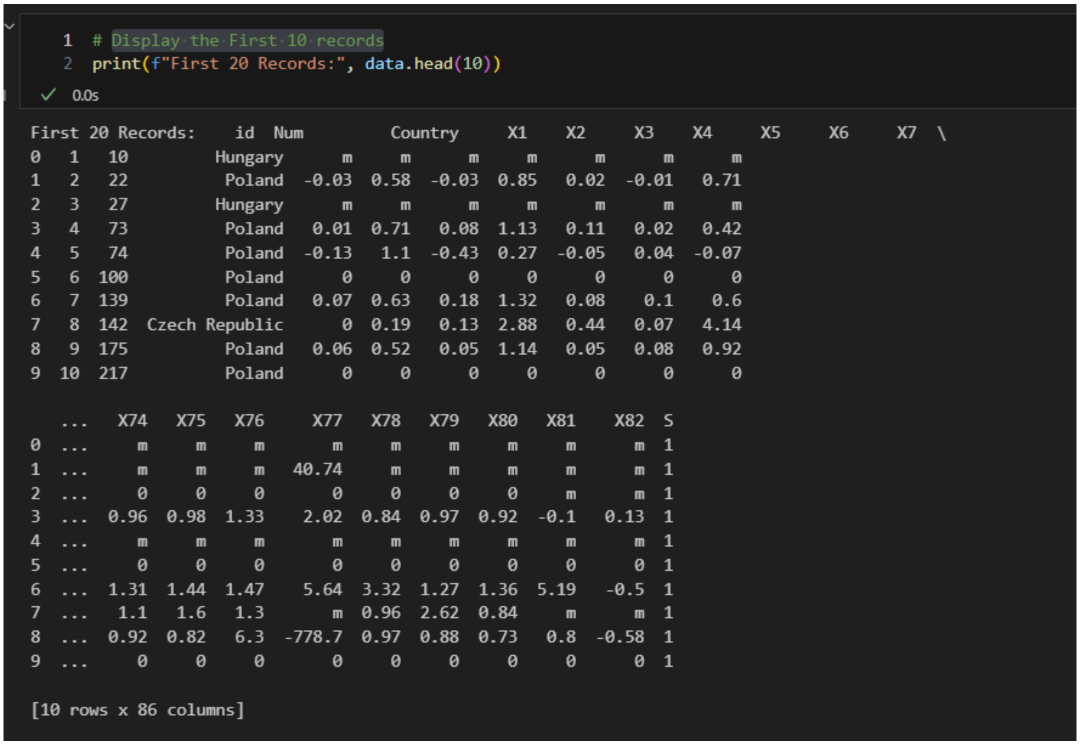
1. Load the data, check the information on the dataset.



```python
# Load the data, check the information on the dataset.

# Identify Column names
print(f"Column Names:", data.columns)
print()

# Identify the Data types of Data
print(f"Data Types:", data.dtypes)
print()

# Display the total number of records
print(f"Total Number of Records:", len(data))
print(f"Rows & Colums:", data.shape)
```

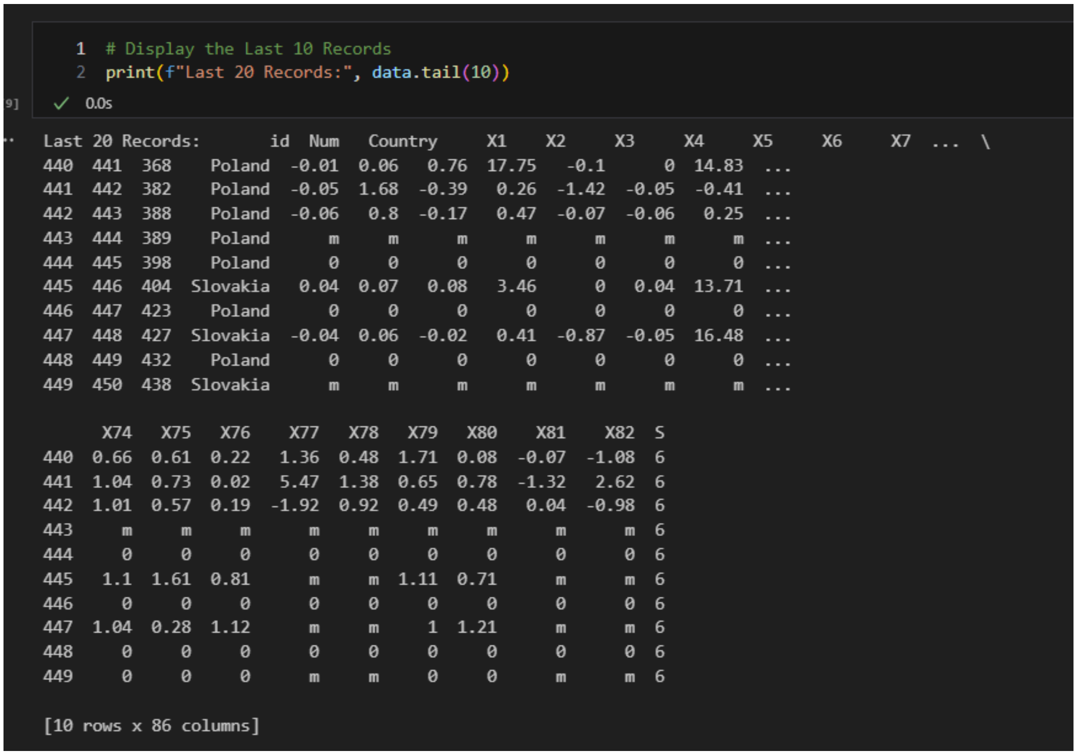
```
Column Names: Index(['id', 'Num', 'Country', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8',
       'X9', 'X10', 'X11', 'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18',
       'X19', 'X20', 'X21', 'X22', 'X23', 'X24', 'X25', 'X26', 'X27', 'X28',
       'X29', 'X30', 'X31', 'X32', 'X33', 'X34', 'X35', 'X36', 'X37', 'X38',
       'X39', 'X40', 'X41', 'X42', 'X43', 'X44', 'X45', 'X46', 'X47', 'X48',
       'X49', 'X50', 'X51', 'X52', 'X53', 'X54', 'X55', 'X56', 'X57', 'X58',
       'X59', 'X60', 'X61', 'X62', 'X63', 'X64', 'X65', 'X66', 'X67', 'X68',
       'X69', 'X70', 'X71', 'X72', 'X73', 'X74', 'X75', 'X76', 'X77', 'X78',
       'X79', 'X80', 'X81', 'X82', 'S'],
      dtype='object')

Data Types: id          int64
Num          int64
Country     object
X1          object
X2          object
             ...
X79         object
X80         object
X81         object
X82         object
S            int64
Length: 86, dtype: object

Total Number of Records: 450
Rows & Colums: (450, 86)
```

2. Display the First 10 and the Last 10 records.



```python
# Display the First 10 records
print(f"First 20 Records:", data.head(10))
```

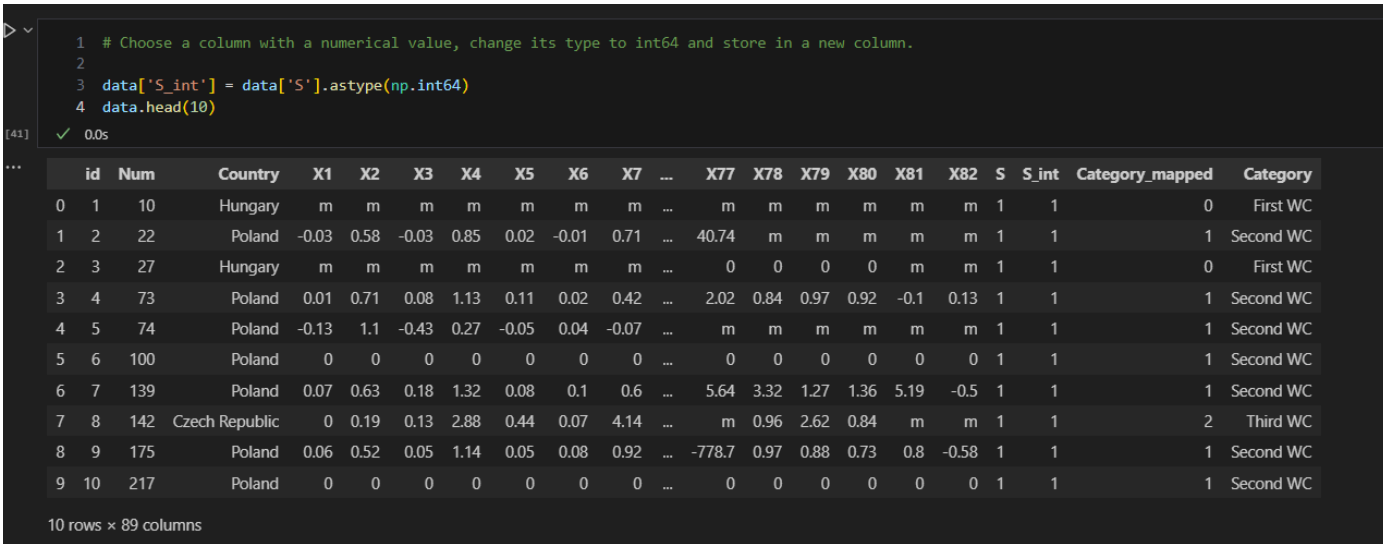
```
First 20 Records:     id  Num         Country     X1    X2     X3    X4     X5     X6     X7  \
0   1   10         Hungary      m     m      m     m      m      m      m
1   2   22          Poland  -0.03  0.58  -0.03  0.85   0.02  -0.01   0.71
2   3   27         Hungary      m     m      m     m      m      m      m
3   4   73          Poland   0.01  0.71   0.08  1.13   0.11   0.02   0.42
4   5   74          Poland  -0.13   1.1  -0.43  0.27  -0.05   0.04  -0.07
5   6  100          Poland      0     0      0     0      0      0      0
6   7  139          Poland   0.07  0.63   0.18  1.32   0.08    0.1    0.6
7   8  142  Czech Republic      0  0.19   0.13  2.88   0.44   0.07   4.14
8   9  175          Poland   0.06  0.52   0.05  1.14   0.05   0.08   0.92
9  10  217          Poland      0     0      0     0      0      0      0

   ...   X74   X75   X76     X77   X78   X79   X80   X81    X82  S
0  ...     m     m     m       m     m     m     m     m      m  1
1  ...     m     m     m   40.74     m     m     m     m      m  1
2  ...     0     0     0       0     0     0     0     m      m  1
3  ...  0.96  0.98  1.33    2.02  0.84  0.97  0.92  -0.1   0.13  1
4  ...     m     m     m       m     m     m     m     m      m  1
5  ...     0     0     0       0     0     0     0     0      0  1
6  ...  1.31  1.44  1.47    5.64  3.32  1.27  1.36  5.19   -0.5  1
7  ...   1.1   1.6   1.3       m  0.96  2.62  0.84     m      m  1
8  ...  0.92  0.82   6.3  -778.7  0.97  0.88  0.73   0.8  -0.58  1
9  ...     0     0     0       0     0     0     0     0      0  1

[10 rows x 86 columns]
```



```python
# Display the Last 10 Records
print(f"Last 20 Records:", data.tail(10))
```

```
Last 20 Records:       id  Num   Country     X1    X2     X3     X4     X5     X6     X7  ...  \
440  441  368    Poland  -0.01  0.06   0.76  17.75   -0.1      0  14.83  ...
441  442  382    Poland  -0.05  1.68  -0.39   0.26  -1.42  -0.05  -0.41  ...
442  443  388    Poland  -0.06   0.8  -0.17   0.47  -0.07  -0.06   0.25  ...
443  444  389    Poland      m     m      m      m      m      m      m  ...
444  445  398    Poland      0     0      0      0      0      0      0  ...
445  446  404  Slovakia   0.04  0.07   0.08   3.46      0   0.04  13.71  ...
446  447  423    Poland      0     0      0      0      0      0      0  ...
447  448  427  Slovakia  -0.04  0.06  -0.02   0.41  -0.87  -0.05  16.48  ...
448  449  432    Poland      0     0      0      0      0      0      0  ...
449  450  438  Slovakia      m     m      m      m      m      m      m  ...

      X74   X75   X76    X77   X78   X79   X80    X81    X82  S
440  0.66  0.61  0.22   1.36  0.48  1.71  0.08  -0.07  -1.08  6
441  1.04  0.73  0.02   5.47  1.38  0.65  0.78  -1.32   2.62  6
442  1.01  0.57  0.19  -1.92  0.92  0.49  0.48   0.04  -0.98  6
443     m     m     m      m     m     m     m      m      m  6
444     0     0     0      0     0     0     0      0      0  6
445   1.1  1.61  0.81      m     m  1.11  0.71      m      m  6
446     0     0     0      0     0     0     0      0      0  6
447  1.04  0.28  1.12      m     m     1  1.21      m      m  6
448     0     0     0      0     0     0     0      0      0  6
449     0     0     0      m     m     0     0      m      m  6

[10 rows x 86 columns]
```

3. Choose a column with a numerical value, change its type to int64 and store in a new column.



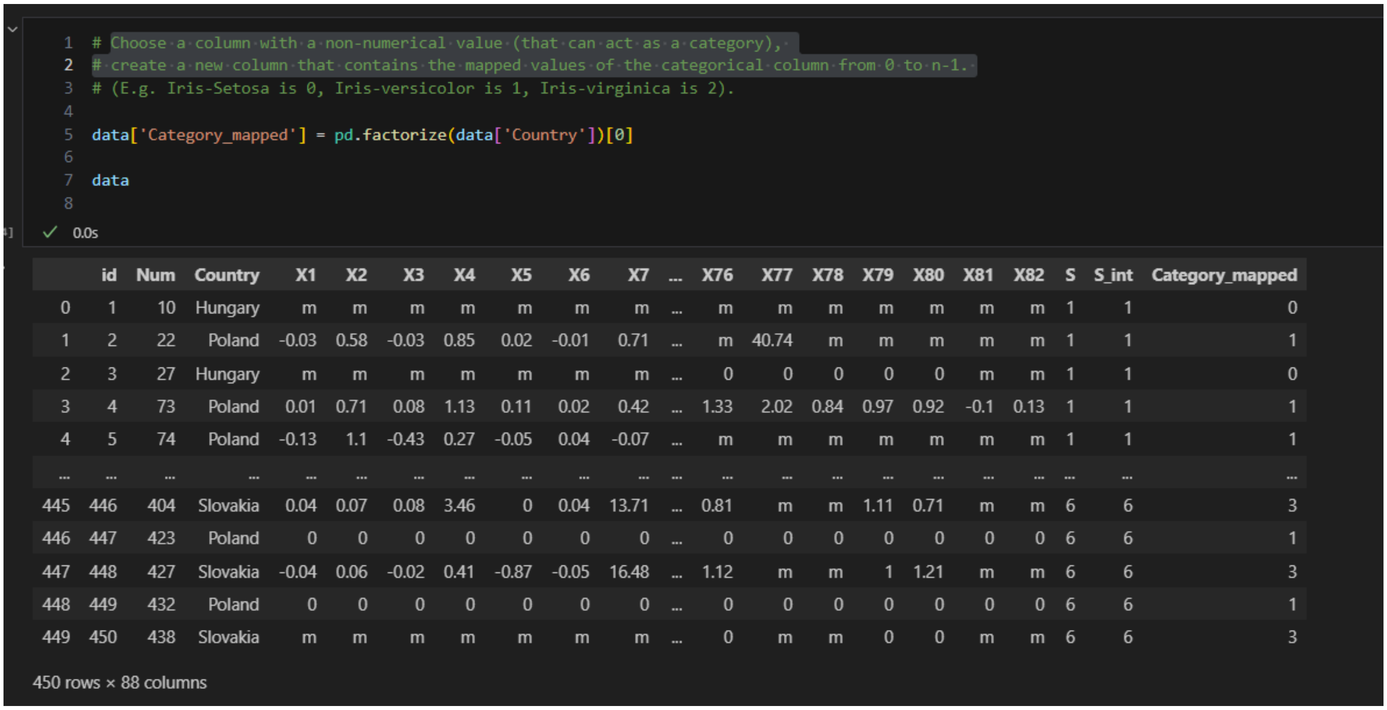
```
# Choose a column with a numerical value, change its type to int64 and store in a new column.

data['S_int'] = data['S'].astype(np.int64)
data.head(10)
```

[41] ✓ 0.0s

	id	Num	Country	X1	X2	X3	X4	X5	X6	X7	...	X77	X78	X79	X80	X81	X82	S	S_int	Category_mapped	Category
0	1	10	Hungary	m	m	m	m	m	m	m	...	m	m	m	m	m	m	1	1	0	First WC
1	2	22	Poland	-0.03	0.58	-0.03	0.85	0.02	-0.01	0.71	...	40.74	m	m	m	m	m	1	1	1	Second WC
2	3	27	Hungary	m	m	m	m	m	m	m	...	0	0	0	0	m	m	1	1	0	First WC
3	4	73	Poland	0.01	0.71	0.08	1.13	0.11	0.02	0.42	...	2.02	0.84	0.97	0.92	-0.1	0.13	1	1	1	Second WC
4	5	74	Poland	-0.13	1.1	-0.43	0.27	-0.05	0.04	-0.07	...	m	m	m	m	m	m	1	1	1	Second WC
5	6	100	Poland	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	1	1	Second WC
6	7	139	Poland	0.07	0.63	0.18	1.32	0.08	0.1	0.6	...	5.64	3.32	1.27	1.36	5.19	-0.5	1	1	1	Second WC
7	8	142	Czech Republic	0	0.19	0.13	2.88	0.44	0.07	4.14	...	m	0.96	2.62	0.84	m	m	1	1	2	Third WC
8	9	175	Poland	0.06	0.52	0.05	1.14	0.05	0.08	0.92	...	-778.7	0.97	0.88	0.73	0.8	-0.58	1	1	1	Second WC
9	10	217	Poland	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	1	1	Second WC

10 rows × 89 columns

4. Choose a column with a non-numerical value (that can act as a category), create a new column that contains the mapped values of the categorical column from 0 to n-1.



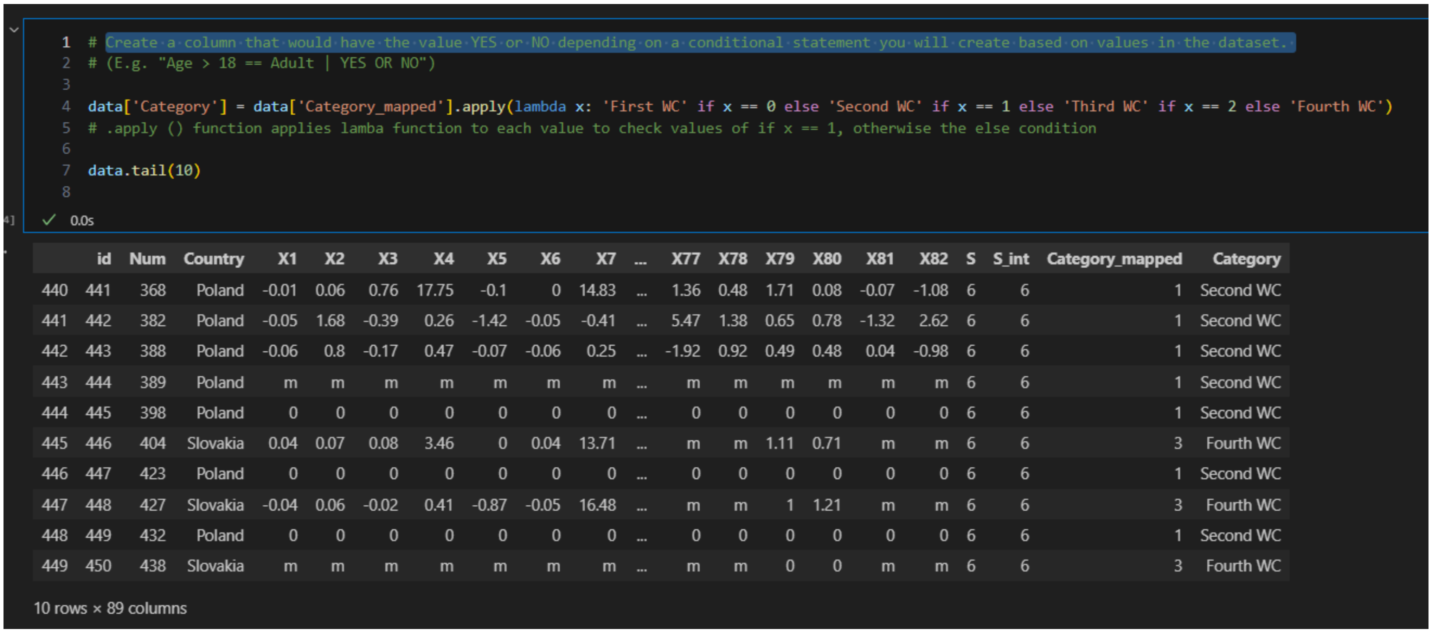
```
# Choose a column with a non-numerical value (that can act as a category),
# create a new column that contains the mapped values of the categorical column from 0 to n-1.
# (E.g. Iris-Setosa is 0, Iris-versicolor is 1, Iris-virginica is 2).

data['Category_mapped'] = pd.factorize(data['Country'])[0]

data

```

✓ 0.0s

	id	Num	Country	X1	X2	X3	X4	X5	X6	X7	...	X76	X77	X78	X79	X80	X81	X82	S	S_int	Category_mapped
0	1	10	Hungary	m	m	m	m	m	m	m	...	m	m	m	m	m	m	m	1	1	0
1	2	22	Poland	-0.03	0.58	-0.03	0.85	0.02	-0.01	0.71	...	m	40.74	m	m	m	m	m	1	1	1
2	3	27	Hungary	m	m	m	m	m	m	m	...	0	0	0	0	0	m	m	1	1	0
3	4	73	Poland	0.01	0.71	0.08	1.13	0.11	0.02	0.42	...	1.33	2.02	0.84	0.97	0.92	-0.1	0.13	1	1	1
4	5	74	Poland	-0.13	1.1	-0.43	0.27	-0.05	0.04	-0.07	...	m	m	m	m	m	m	m	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
445	446	404	Slovakia	0.04	0.07	0.08	3.46	0	0.04	13.71	...	0.81	m	m	1.11	0.71	m	m	6	6	3
446	447	423	Poland	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	6	6	1
447	448	427	Slovakia	-0.04	0.06	-0.02	0.41	-0.87	-0.05	16.48	...	1.12	m	m	1	1.21	m	m	6	6	3
448	449	432	Poland	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	6	6	1
449	450	438	Slovakia	m	m	m	m	m	m	m	...	0	m	m	0	0	m	m	6	6	3

450 rows × 88 columns

5. Create a column that would have the value YES or NO depending on a conditional statement you will create based on values in the dataset.



```python
# Create a column that would have the value YES or NO depending on a conditional statement you will create based on values in the dataset.
# (E.g. "Age > 18 == Adult | YES OR NO")

data['Category'] = data['Category_mapped'].apply(lambda x: 'First WC' if x == 0 else 'Second WC' if x == 1 else 'Third WC' if x == 2 else 'Fourth WC')
# .apply () function applies lamba function to each value to check values of if x == 1, otherwise the else condition

data.tail(10)

```

	id	Num	Country	X1	X2	X3	X4	X5	X6	X7	...	X77	X78	X79	X80	X81	X82	S	S_int	Category_mapped	Category
440	441	368	Poland	-0.01	0.06	0.76	17.75	-0.1	0	14.83	...	1.36	0.48	1.71	0.08	-0.07	-1.08	6	6	1	Second WC
441	442	382	Poland	-0.05	1.68	-0.39	0.26	-1.42	-0.05	-0.41	...	5.47	1.38	0.65	0.78	-1.32	2.62	6	6	1	Second WC
442	443	388	Poland	-0.06	0.8	-0.17	0.47	-0.07	-0.06	0.25	...	-1.92	0.92	0.49	0.48	0.04	-0.98	6	6	1	Second WC
443	444	389	Poland	m	m	m	m	m	m	m	...	m	m	m	m	m	m	6	6	1	Second WC
444	445	398	Poland	0	0	0	0	0	0	0	...	0	0	0	0	0	0	6	6	1	Second WC
445	446	404	Slovakia	0.04	0.07	0.08	3.46	0	0.04	13.71	...	m	m	1.11	0.71	m	m	6	6	3	Fourth WC
446	447	423	Poland	0	0	0	0	0	0	0	...	0	0	0	0	0	0	6	6	1	Second WC
447	448	427	Slovakia	-0.04	0.06	-0.02	0.41	-0.87	-0.05	16.48	...	m	m	1	1.21	m	m	6	6	3	Fourth WC
448	449	432	Poland	0	0	0	0	0	0	0	...	0	0	0	0	0	0	6	6	1	Second WC
449	450	438	Slovakia	m	m	m	m	m	m	m	...	m	m	0	0	m	m	6	6	3	Fourth WC

10 rows × 89 columns

6.3 Conclusion

I’m currently learning how to use lambda functions in Pandas to manipulate data in a DataFrame based on multiple conditions. By applying the apply() function with a lambda expression, I can map the values of a specific column to new categories. This approach helps me efficiently transform data, like mapping numeric values to classifications such as 'First WC’, ‘Second WC’, ‘Third WC’, or ‘Fourth WC’ depending on the conditions I set.

In summary, when handling more than two conditions, I can use if-elif-else clauses inside the lambda function. This lets me create flexible, multi-condition transformations, which is especially useful for data preprocessing and feature engineering. It’s a powerful method that simplifies how I work with conditional logic in my datasets.