National Technical University of Ukraine
“Igor Sikorsky Kyiv Polytechnic Institute”
Faculty of Informatics and Computer Science
Department of Information Systems and Technologies

Lab work No3

Pandas data structures

Performed by:
student of group IM-14
Full name KULUBECİOGLU Mehmet

Compiled by:
Yulia Timofeeva

KYİV.2023

Tasks

File diamonds.csv

1. Display information about dataset, main statistic characteristics, features type. Which features are quantitative and which are qualitative?

2. Create new DataFrame object by copying subset of dataset. Set your own indexes for rows and columns. Add new row.

3. Using original dataset:

a) Calculate median weight (in carats) of diamonds with different cut.

b) Calculate total cost of all diamond of Premium and Ideal class, which weight more than 0.3 carats.

c) Add new column with the price per carat.

d) Add new column with the average diamond length (y) for this color class.

My Full codes:

```python
import pandas as pd

# Load the diamonds dataset
diamonds_data = pd.read_csv('diamonds.csv')

# Task 1: Display information about the dataset
print("Task 1: Display information about the dataset")
print(diamonds_data.info())
print("\nMain Statistic Characteristics:")
print(diamonds_data.describe())
print("\nFeatures Type:")
print(diamonds_data.dtypes)

# Task 2: Create a new DataFrame with custom indexes and add a new row
print("\nTask 2: Create a new DataFrame with custom indexes and add a new row")
custom_index = ['row1', 'row2', 'row3', 'row4', 'row5']
custom_columns = diamonds_data.columns  # Use existing columns from the diamonds dataset
new_dataframe = pd.DataFrame(data=diamonds_data.sample(5).values,
                             index=custom_index,
                             columns=custom_columns)
print(new_dataframe)

# Task 3a: Calculate median weight (in carats) of diamonds with different cut
print("\nTask 3a: Calculate median weight (in carats) of diamonds with different cut")
median_weight_by_cut = diamonds_data.groupby('cut')['carat'].median()
print(median_weight_by_cut)
```

```python
# Task 3b: Calculate total cost of all diamonds of Premium and Ideal class, which weigh more than 0.3 carats
print("\nTask 3b: Calculate total cost of all diamonds of Premium and Ideal class, which weigh more than 0.3 carats")
total_cost_premium_ideal = diamonds_data[(diamonds_data['cut'].isin(['Premium', 'Ideal'])) & (diamonds_data['carat'] > 0.3)]['price'].sum()
print("Total Cost:", total_cost_premium_ideal)

# Task 3c: Add a new column with the price per carat
print("\nTask 3c: Add a new column with the price per carat")
diamonds_data['price_per_carat'] = diamonds_data['price'] / diamonds_data['carat']
print(diamonds_data[['price', 'carat', 'price_per_carat']])

# Task 3d: Add a new column with the average diamond length (y) for each color class
print("\nTask 3d: Add a new column with the average diamond length (y) for each color class")
average_length_by_color = diamonds_data.groupby('color')['y'].mean()
diamonds_data['average_length_by_color'] = diamonds_data['color'].map(average_length_by_color)
print(diamonds_data[['color', 'y', 'average_length_by_color']])
```

Now let me make my presentation step by step.

1. Display information about dataset, main statistic characteristics, features type. Which features are quantitative and which are qualitative?

```python
# Task 1: Display information about the dataset
print("Task 1: Display information about the dataset")
print(diamonds_data.info())
print("\nMain Statistic Characteristics:")
print(diamonds_data.describe())
print("\nFeatures Type:")
print(diamonds_data.dtypes)
```

```
Task 1: Display information about the dataset
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1514 entries, 0 to 1513
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  1514 non-null   int64
 1   carat       1514 non-null   float64
 2   cut         1514 non-null   object
 3   color       1514 non-null   object
 4   clarity     1514 non-null   object
 5   depth       1514 non-null   float64
 6   table       1514 non-null   float64
 7   price       1514 non-null   int64
 8   x           1514 non-null   float64
 9   y           1514 non-null   float64
 10  z           1514 non-null   float64
dtypes: float64(6), int64(2), object(3)
memory usage: 130.2+ KB
None

Main Statistic Characteristics:
        Unnamed: 0        carat        depth        table        price  \
count  1514.000000  1514.000000  1514.000000  1514.000000  1514.000000
mean    757.500000     0.697774    61.736856    57.611823  2541.036988
std     437.198468     0.191660     1.679440     2.396434   820.502099
min       1.000000     0.200000    53.000000    52.000000   326.000000
25%     379.250000     0.700000    60.900000    56.000000  2790.000000
50%     757.500000     0.720000    61.800000    57.000000  2846.000000
75%    1135.750000     0.790000    62.600000    59.000000  2913.000000
max    1514.000000     1.500000    69.500000    70.000000  2995.000000
```

```
                 x           y            z
count  1514.000000  1514.00000  1514.000000
mean      5.630997     5.62747     3.474947
std       0.606140     0.59511     0.380230
min       3.790000     3.75000     2.270000
25%       5.650000     5.66000     3.470000
50%       5.770000     5.77000     3.560000
75%       5.920000     5.93000     3.660000
max       7.260000     7.09000     4.700000

Features Type:
Unnamed: 0      int64
carat         float64
cut            object
color          object
clarity        object
depth         float64
table         float64
price           int64
x             float64
y             float64
z             float64
dtype: object
```

2. Create new DataFrame object by copying subset of dataset. Set your own indexes for rows and columns. Add new row.

```
# Task 2: Create a new DataFrame with custom indexes and add a new row
print("\nTask 2: Create a new DataFrame with custom indexes and add a new row")
custom_index = ['row1', 'row2', 'row3', 'row4', 'row5']
custom_columns = diamonds_data.columns  # Use existing columns from the diamonds dataset
new_dataframe = pd.DataFrame(data=diamonds_data.sample(5).values,
                             index=custom_index,
                             columns=custom_columns)
print(new_dataframe)
```

```
 Task 2: Create a new DataFrame with custom indexes and add a new row
      Unnamed: 0 carat        cut color clarity depth table price     x     y  \
 row1       1013   0.8  'Premium'   'F'   'SI1'  62.7  58.0  2901  5.91  5.93
 row2        492   0.7  'Premium'   'E'   'VS2'  60.2  60.0  2822  5.73   5.7
 row3        975  0.82     'Good'   'G'   'SI2'  59.9  62.0  2893  6.02  6.04
 row4        678  0.79     'Good'   'E'   'SI1'  64.1  54.0  2849  5.86  5.84
 row5       1022  0.75  'Premium'   'D'   'SI1'  62.8  60.0  2903  5.78  5.74

          z
 row1  3.71
 row2  3.44
 row3  3.61
 row4  3.75
 row5  3.62
```

3. Using original dataset:
a) Calculate median weight (in carats) of diamonds with different cut.

```
# Task 3a: Calculate median weight (in carats) of diamonds with different cut
print("\nTask 3a: Calculate median weight (in carats) of diamonds with different cut")
median_weight_by_cut = diamonds_data.groupby('cut')['carat'].median()
print(median_weight_by_cut)
```

```
 Task 3a: Calculate median weight (in carats) of diamonds with different cut
 cut
 'Fair'         0.91
 'Good'         0.71
 'Ideal'        0.71
 'Premium'      0.72
 'Very Good'    0.71
 Name: carat, dtype: float64
```

b) Calculate total cost of all diamond of Premium and Ideal class, which weight more than 0.3 carats.

```
# Task 3b: Calculate total cost of all diamonds of Premium and Ideal class, which weigh more than 0.3 carats
print("\nTask 3b: Calculate total cost of all diamonds of Premium and Ideal class, which weigh more than 0.3 carats")
total_cost_premium_ideal = diamonds_data[(diamonds_data['cut'].isin(['Premium', 'Ideal'])) & (diamonds_data['carat'] > 0.3)]['price'].sum()
print("Total Cost:", total_cost_premium_ideal)
```

```
Name: carat, dtype: float64

Task 3b: Calculate total cost of all diamonds of Premium and Ideal class, which weigh more than 0.3 carats
Total Cost: 0
```

c) Add new column with the price per carat.

```
# Task 3c: Add a new column with the price per carat
print("\nTask 3c: Add a new column with the price per carat")
diamonds_data['price_per_carat'] = diamonds_data['price'] / diamonds_data['carat']
print(diamonds_data[['price', 'carat', 'price_per_carat']])
```

```
Task 3c: Add a new column with the price per carat
      price  carat  price_per_carat
0       326   0.23      1417.391304
1       326   0.21      1552.380952
2       327   0.23      1421.739130
3       334   0.29      1151.724138
4       335   0.31      1080.645161
...     ...    ...              ...
1509   2994   0.81      3696.296296
1510   2994   1.24      2414.516129
1511   2994   0.81      3696.296296
1512   2994   0.81      3696.296296
1513   2995   0.73      4102.739726

[1514 rows x 3 columns]
```

d) Add new column with the average diamond length (y) for this color class.

```
# Task 3d: Add a new column with the average diamond length (y) for each color class
print("\nTask 3d: Add a new column with the average diamond length (y) for each color class")
average_length_by_color = diamonds_data.groupby('color')['y'].mean()
diamonds_data['average_length_by_color'] = diamonds_data['color'].map(average_length_by_color)
print(diamonds_data[['color', 'y', 'average_length_by_color']])
```

```
Task 3d: Add a new column with the average diamond length (y) for each color class
     color     y  average_length_by_color
0      'E'  3.98                 5.594877
1      'E'  3.84                 5.594877
2      'E'  4.07                 5.594877
3      'I'  4.23                 5.544870
4      'J'  4.35                 5.714828
...    ...   ...                      ...
1509   'G'  5.84                 5.617712
1510   'J'  6.85                 5.714828
1511   'G'  5.92                 5.617712
1512   'D'  5.93                 5.633057
1513   'D'  5.82                 5.633057

[1514 rows x 3 columns]
```