import the Diamond dataset

In [7]:

```
import pandas as pd
df=pd.read_csv("diamonds.csv")
df=df.drop(columns='Unnamed: 0')
```

1. Write a Pandas program to find the number of rows and columns and data type of each column of diamonds Dataframe.

In [8]:

```
### code here

print("Number of rows:",df.shape[0])
print("Number of columns:",df.shape[1])
print("Data types:",df.dtypes)
```

```
Number of rows: 53940
Number of columns: 10
Data types: carat      float64
cut         object
color       object
clarity     object
depth      float64
table      float64
price        int64
x          float64
y          float64
z          float64
dtype: object
```

2. Write a Pandas program to summarize only 'object' columns of the diamonds Dataframe.

In [9]:

```
### code here
df.describe(include='object')
```

Out[9]:

	cut	color	clarity
count	53940	53940	53940
unique	5	7	8
top	Ideal	G	SI1
freq	21551	11292	13065

3. Write a Pandas program to remove the second column of the diamonds Dataframe. (don't use original dataset)

In [10]:

```
### code here
diamonds_data_without_second_column = df.drop(columns=df.columns[1])

# Print the modified DataFrame
print(diamonds_data_without_second_column)
```

```
       carat color clarity  depth  table  price     x     y     z
0       0.23     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1       0.21     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2       0.23     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3       0.29     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4       0.31     J     SI2   63.3   58.0    335  4.34  4.35  2.75
...      ...   ...     ...    ...    ...    ...   ...   ...   ...
53935   0.72     D     SI1   60.8   57.0   2757  5.75  5.76  3.50
53936   0.72     D     SI1   63.1   55.0   2757  5.69  5.75  3.61
53937   0.70     D     SI1   62.8   60.0   2757  5.66  5.68  3.56
53938   0.86     H     SI2   61.0   58.0   2757  6.15  6.12  3.74
53939   0.75     D     SI2   62.2   55.0   2757  5.83  5.87  3.64

[53940 rows x 9 columns]
```

4. Write a Pandas program to remove multiple rows at once (axis=0 refers to rows) from diamonds dataframe. (dont use original dataset)

In [11]:

```
### code here
df.copy = df.copy()

# Remove rows at index 1, 2 and 3
df.copy = df.copy.drop([1, 2, 3], axis=0)

# Print the updated DataFrame
print(df.copy)
```

```
       carat        cut color clarity  depth  table  price     x     y     z
0       0.23      Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
4       0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
5       0.24  Very Good     J    VVS2   62.8   57.0    336  3.94  3.96  2.48
6       0.24  Very Good     I    VVS1   62.3   57.0    336  3.95  3.98  2.47
7       0.26  Very Good     H     SI1   61.9   55.0    337  4.07  4.11  2.53
...      ...        ...   ...     ...    ...    ...    ...   ...   ...   ...
53935   0.72      Ideal     D     SI1   60.8   57.0   2757  5.75  5.76  3.50
53936   0.72       Good     D     SI1   63.1   55.0   2757  5.69  5.75  3.61
53937   0.70  Very Good     D     SI1   62.8   60.0   2757  5.66  5.68  3.56
53938   0.86    Premium     H     SI2   61.0   58.0   2757  6.15  6.12  3.74
53939   0.75      Ideal     D     SI2   62.2   55.0   2757  5.83  5.87  3.64

[53937 rows x 10 columns]
```

5. Write a Pandas program to sort the 'cut' Series in ascending order (returns a Series) of diamonds Dataframe.

In [12]:

```
### code here
df.cut.sort_values()
```

Out[12]:

```
3850          Fair
51464         Fair
51466         Fair
10237         Fair
10760         Fair
           ...
7402     Very Good
43101    Very Good
16893    Very Good
16898    Very Good
21164    Very Good
Name: cut, Length: 53940, dtype: object
```

6. Write a Pandas program to sort the entire diamonds DataFrame by the 'carat' Series in ascending and descending order.

In [13]:

```
### code here
print("ascending:",df.sort_values(by="carat"))
print("descending:",df.sort_values(by="carat",ascending=False))
```

```
ascending:        carat      cut color clarity  depth  table  price      x
y     z
31593   0.20  Premium     E     VS2   61.1   59.0    367   3.81   3.78  2.32
31597   0.20    Ideal     D     VS2   61.5   57.0    367   3.81   3.77  2.33
31596   0.20  Premium     F     VS2   62.6   59.0    367   3.73   3.71  2.33
31595   0.20    Ideal     E     VS2   59.7   55.0    367   3.86   3.84  2.30
31594   0.20  Premium     E     VS2   59.7   62.0    367   3.84   3.80  2.28
...      ...      ...   ...     ...    ...    ...    ...    ...    ...   ...
25999   4.01  Premium     J      I1   62.5   62.0  15223  10.02   9.94  6.24
25998   4.01  Premium     I      I1   61.0   61.0  15223  10.14  10.10  6.17
27130   4.13     Fair     H      I1   64.8   61.0  17329  10.00   9.85  6.43
27630   4.50     Fair     J      I1   65.8   58.0  18531  10.23  10.16  6.72
27415   5.01     Fair     J      I1   65.5   59.0  18018  10.74  10.54  6.98

[53940 rows x 10 columns]
descending:        carat      cut color clarity  depth  table  price      x
y     z
27415   5.01     Fair     J      I1   65.5   59.0  18018  10.74  10.54  6.98
27630   4.50     Fair     J      I1   65.8   58.0  18531  10.23  10.16  6.72
27130   4.13     Fair     H      I1   64.8   61.0  17329  10.00   9.85  6.43
25999   4.01  Premium     J      I1   62.5   62.0  15223  10.02   9.94  6.24
25998   4.01  Premium     I      I1   61.0   61.0  15223  10.14  10.10  6.17
...      ...      ...   ...     ...    ...    ...    ...    ...    ...   ...
31592   0.20  Premium     E     VS2   59.0   60.0    367   3.81   3.78  2.24
31591   0.20  Premium     E     VS2   59.8   62.0    367   3.79   3.77  2.26
31601   0.20  Premium     D     VS2   61.7   60.0    367   3.77   3.72  2.31
14      0.20  Premium     E     SI2   60.2   62.0    345   3.79   3.75  2.27
31596   0.20  Premium     F     VS2   62.6   59.0    367   3.73   3.71  2.33

[53940 rows x 10 columns]
```

7. Write a Pandas program to filter the DataFrame rows to only show carat weight at least 0.3.

In [14]:

```
#### code here
carat_at_least_03 = df[df['carat'] >= 0.3]

# Print filtered DataFrame
print(carat_at_least_03)
```

```
       carat        cut color clarity  depth  table  price     x     y     z
4       0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
10      0.30       Good     J     SI1   64.0   55.0    339  4.25  4.28  2.73
13      0.31      Ideal     J     SI2   62.2   54.0    344  4.35  4.37  2.71
15      0.32    Premium     E      I1   60.9   58.0    345  4.38  4.42  2.68
16      0.30      Ideal     I     SI2   62.0   54.0    348  4.31  4.34  2.68
...      ...        ...   ...     ...    ...    ...    ...   ...   ...   ...
53935   0.72      Ideal     D     SI1   60.8   57.0   2757  5.75  5.76  3.50
53936   0.72       Good     D     SI1   63.1   55.0   2757  5.69  5.75  3.61
53937   0.70  Very Good     D     SI1   62.8   60.0   2757  5.66  5.68  3.56
53938   0.86    Premium     H     SI2   61.0   58.0   2757  6.15  6.12  3.74
53939   0.75      Ideal     D     SI2   62.2   55.0   2757  5.83  5.87  3.64

[52341 rows x 10 columns]
```

8. Write a Pandas program to find the details of the diamonds where length>5, width>5 and depth>5.

In [15]:

```
### code here
di = df[(df['x'] > 5) & (df['y'] > 5) & (df['z'] > 5)]
di
```

Out[15]:

	carat	cut	color	clarity	depth	table	price	x	y	z
11778	1.83	Fair	J	I1	70.0	58.0	5083	7.34	7.28	5.12
13002	2.14	Fair	J	I1	69.4	57.0	5405	7.74	7.70	5.36
13118	2.15	Fair	J	I1	65.5	57.0	5430	8.01	7.95	5.23
13562	1.96	Fair	F	I1	66.6	60.0	5554	7.59	7.56	5.04
13757	2.22	Fair	J	I1	66.7	56.0	5607	8.04	8.02	5.36
...	...	...	...	...	...	...	...	...	...	...
27748	2.00	Very Good	G	SI1	63.5	56.0	18818	7.90	7.97	5.04
27749	2.29	Premium	I	VS2	60.8	60.0	18823	8.50	8.47	5.16
48410	0.51	Very Good	E	VS1	61.8	54.7	1970	5.12	5.15	31.80
49189	0.51	Ideal	E	VS1	61.8	55.0	2075	5.15	31.80	5.12
49905	0.50	Very Good	G	VVS1	63.7	58.0	2180	5.01	5.04	5.06

1457 rows × 10 columns

9. Write a Pandas program to calculate the mean of each row of diamonds DataFrame.

In [16]:

```
### code here
df.mean(axis=1)
```

```
C:\Users\pamar\AppData\Local\Temp\ipykernel_25520\2232954823.py:2: FutureWar
ning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_on
ly=None') is deprecated; in a future version this will raise TypeError.  Sel
ect only valid columns before calling the reduction.
  df.mean(axis=1)
```

Out[16]:

```
0         64.727143
1         65.292857
2         65.651429
3         66.535714
4         66.864286
            ...
53935    412.932857
53936    412.981429
53937    413.628571
53938    413.267143
53939    412.898571
Length: 53940, dtype: float64
```

10. Write a Pandas program to calculate the mean of price for each cut and find maximum top 3 of diamonds DataFrame.

In [17]:

```
#### code here
mean_prices = df.groupby('cut')['price'].mean()
print(mean_prices)
print(mean_prices.nlargest(3))
```

```
cut
Fair         4358.757764
Good         3928.864452
Ideal        3457.541970
Premium      4584.257704
Very Good    3981.759891
Name: price, dtype: float64
cut
Premium      4584.257704
Fair         4358.757764
Very Good    3981.759891
Name: price, dtype: float64
```

In [ ]:

```

```

11. Write a Pandas program to calculate count, minimum, maximum price for each cut of diamonds DataFrame.

In [18]:

```
### code here
df.groupby('cut')['price'].agg(['count', 'min', 'max'])
```

Out[18]:

	count	min	max
cut			
Fair	1610	337	18574
Good	4906	327	18788
Ideal	21551	326	18806
Premium	13791	326	18823
Very Good	12082	336	18818

12. Write a Pandas program to display and count the unique values in cut series of diamonds DataFrame.

In [19]:

```
### code here
df["cut"].unique()
```

Out[19]:

```
array(['Ideal', 'Premium', 'Good', 'Very Good', 'Fair'], dtype=object)
```

######13. Write a Pandas program to count the number of missing values in each Series of diamonds DataFrame.

In [20]:

```
### code here
df.isnull().sum()
```

Out[20]:

```
carat      0
cut        0
color      0
clarity    0
depth      0
table      0
price      0
x          0
y          0
z          0
dtype: int64
```

14. Write a Pandas program to calculate the multiply of x, y and z for each cut of diamonds DataFrame.

In [21]:

```
### code here
df['volume'] = df['x'] * df['y'] * df['z']
print(df['volume'])
df.groupby('cut')['volume'].sum()
```

```
0         38.202030
1         34.505856
2         38.076885
3         46.724580
4         51.917250
            ...
53935    115.920000
53936    118.110175
53937    114.449728
53938    140.766120
53939    124.568444
Name: volume, Length: 53940, dtype: float64
```

Out[21]:

```
cut
Fair         2.655704e+05
Good         6.684782e+05
Ideal        2.486876e+06
Premium      2.000414e+06
Very Good    1.582739e+06
Name: volume, dtype: float64
```

15. Write a Pandas program to read rows 0 through 2 (inclusive), columns ‘color’ and ‘price’ of diamonds DataFrame.

In [22]:

```
## code here
df.loc[0:2, ['color', 'price']]
```

Out[22]:

	color	price
0	E	326
1	E	326
2	E	327

16. Write a Pandas program to read rows in positions 0 and 1, columns in positions 0 and 3 of diamonds DataFrame.

In [23]:

```
### code here
df.iloc[[0, 1], [0, 3]]
```

Out[23]:

	carat	clarity
0	0.23	SI2
1	0.21	SI1

17. Write a Pandas program to get randomly sample rows from diamonds DataFrame.

In [24]:

```
### code here
df.sample(n=5)
```

Out[24]:

	carat	cut	color	clarity	depth	table	price	x	y	z	volume
24834	2.02	Premium	H	SI1	61.4	61.0	13229	8.09	8.03	4.95	321.565365
7804	0.90	Very Good	D	SI1	61.8	59.0	4291	6.13	6.16	3.80	143.491040
49740	0.59	Ideal	G	VVS2	62.3	56.0	2155	5.34	5.39	3.34	96.133884
15573	1.30	Premium	I	VS2	62.7	58.0	6246	6.97	6.90	4.35	209.204550
40600	0.32	Very Good	F	SI1	60.7	62.0	497	4.40	4.43	2.68	52.238560

18. Write a Pandas program to get sample 75% of the diamonds DataFrame's rows without replacement and store the remaining 25% of the rows in another DataFrame.

In [25]:

```
### code here
sample=df.sample(frac=0.75, replace=False)
sample
```

Out[25]:

	carat	cut	color	clarity	depth	table	price	x	y	z	volume
41607	0.32	Premium	J	VS2	61.9	58.0	393	4.35	4.38	2.70	51.443100
31941	0.30	Ideal	F	VS2	61.3	55.0	776	4.32	4.30	2.64	49.040640
20543	1.21	Ideal	G	VS1	61.8	55.0	8864	6.81	6.87	4.23	197.899281
39196	0.43	Premium	D	SI1	60.1	58.0	1064	4.93	4.89	2.95	71.117715
38853	0.40	Ideal	G	VVS2	62.4	56.0	1050	4.68	4.64	2.91	63.191232
...	...	...	...	...	...	...	...	...	...	...	...
42773	0.44	Ideal	G	IF	62.2	53.0	1348	4.90	4.94	3.06	74.070360
13637	1.02	Premium	G	VS2	62.0	57.0	5581	6.48	6.43	4.00	166.665600
12941	1.20	Ideal	G	SI2	62.2	56.0	5385	6.74	6.84	4.22	194.548752
31753	0.40	Premium	D	SI1	59.9	60.0	772	4.75	4.77	2.85	64.573875
51167	0.80	Premium	H	SI2	62.2	58.0	2346	5.99	5.93	3.71	131.781797

40455 rows × 11 columns

In [26]:

```
df[~df.index.isin(sample.index)]
```

Out[26]:

	carat	cut	color	clarity	depth	table	price	x	y	z	volume
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75	51.917250
10	0.30	Good	J	SI1	64.0	55.0	339	4.25	4.28	2.73	49.658700
11	0.23	Ideal	J	VS1	62.8	56.0	340	3.93	3.90	2.46	37.704420
12	0.22	Premium	F	SI1	60.4	61.0	342	3.88	3.84	2.33	34.715136
13	0.31	Ideal	J	SI2	62.2	54.0	344	4.35	4.37	2.71	51.515745
...	...	...	...	...	...	...	...	...	...	...	...
53913	0.80	Good	G	VS2	64.2	58.0	2753	5.84	5.81	3.74	126.899696
53917	0.90	Very Good	J	SI1	63.2	60.0	2753	6.12	6.09	3.86	143.865288
53927	0.79	Good	F	SI1	58.1	59.0	2756	6.06	6.13	3.54	131.503212
53936	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61	118.110175
53939	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64	124.568444

13485 rows × 11 columns

19. Write a Pandas program to read the diamonds DataFrame and detect duplicate color.

In [27]:

```
#### code here
df[df.duplicated(subset='color', keep=False)]
```

Out[27]:

	carat	cut	color	clarity	depth	table	price	x	y	z	volume
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43	38.202030
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31	34.505856
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31	38.076885
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63	46.724580
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75	51.917250
...	...	...	...	...	...	...	...	...	...	...	...
53935	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50	115.920000
53936	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61	118.110175
53937	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56	114.449728
53938	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74	140.766120
53939	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64	124.568444

53940 rows × 11 columns

20. Write a Pandas program to count the duplicate rows of diamonds DataFrame.

In [28]:

```
#### code here
df.duplicated().sum()
```

Out[28]:

```
146
```

In [ ]:

In [ ]: