In [1]:
```
import pandas
```

In [2]:
```
import seaborn as sn # we are importing seaborn to be able to access the sample data sets in the library
```

In [3]:
```
# Seaborn contains sample data that we can use for our work.
sn.get_dataset_names()
```

Out[3]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

In [4]:
```
# Load the iris dataset
sn.load_dataset('iris')
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [6]:
```
# Loading the diamonds data

sn.load_dataset('diamonds')
```

Out[6]:

	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
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...
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

53940 rows × 10 columns

In [11]:
```
# How to save a dataframe to csv on your computer

df = sn.load_dataset('diamonds')
```

In [12]:
```
print(df)
```

```
       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
1       0.21    Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2       0.23       Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3       0.29    Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4       0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
...      ...        ...   ...     ...    ...    ...    ...   ...   ...   ...
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

[53940 rows x 10 columns]
```

In [20]:
```
# to save to file as a csv file, use the pandas to_csv()

df.to_csv('c://workspace/diamonds.csv') # this method also saved the index column to file
```

In [21]:
```
# to save to file as a csv file without the index column, use the pandas to_csv()

df.to_csv('c://workspace/diamonds1.csv', index=False) # the index=False parameter stops pandas from saving the index co
```

In [17]:
```
x = df.index
```

In [18]:
```
x
```

Out[18]:
```
RangeIndex(start=0, stop=53940, step=1)
```

In [ ]:
```
# How to load datasets from file onto a Jupyter Notebook

# use the pandas function

# read_csv for csv file
# read_excel for excel files (xls)
# read_json for .json files
```

In [22]:
```
import pandas as pd

filename  = 'diamonds1.csv'
filepath  = 'c://workspace/'

fullFilePath = filepath + filename

print(fullFilePath)
```

```
c://workspace/diamonds1.csv
```

In [27]:
```
# load the diamonds file

dataset = pd.read_csv(fullFilePath)
```

In [28]:
```
print(dataset)
```

	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
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...
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

[53940 rows x 10 columns]

In [ ]:
```
# Dataframes are matrix in structure consisting of data columns(called FEATURES) and data rows (called SAMPLES)
```

In [29]:
```
# To get the shape of a dataset (number of features and samples )

dataset.shape
```

Out[29]:
```
(53940, 10)
```

In [30]:
```
# Get the names of the features in the data

dataset.columns
```

Out[30]:
```
Index(['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'price', 'x', 'y',
       'z'],
      dtype='object')
```

In [31]:
```
# converting the dataset columns to a list

cols = list(dataset.columns)

cols
```

Out[31]:
```
['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'price', 'x', 'y', 'z']
```

In [32]:
```
for col in cols:
    print(col)
```

```
carat
cut
color
clarity
depth
table
price
x
y
z
```

In [33]:
```
df2 = pd.read_csv( "C:/workspace/diamonds1.csv")
```

In [34]:
```
df2
```

Out[34]:

	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
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...
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

53940 rows × 10 columns

In [37]:
```
path = 'C:/workspace/'
filename = 'diamonds1.csv'
```

In [38]:
```
df3 = pd.read_csv(path + filename)
```

In [39]:
```
df3
```

Out[39]:

	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
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...
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

53940 rows × 10 columns

In [40]:
```
# get dataset shape  number of features and samples

dataset.shape
```

Out[40]:
```
(53940, 10)
```

In [42]:
```
# get column names

dataset.columns
```

Out[42]:
```
Index(['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'price', 'x', 'y',
       'z'],
      dtype='object')
```

In [43]:
```
cols = list(dataset.columns)
cols
```

Out[43]:
```
['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'price', 'x', 'y', 'z']
```

In [45]:
```
# inspect the properties and datatype of the features in the data using .info()

dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   carat    53940 non-null  float64
 1   cut      53940 non-null  object 
 2   color    53940 non-null  object 
 3   clarity  53940 non-null  object 
 4   depth    53940 non-null  float64
 5   table    53940 non-null  float64
 6   price    53940 non-null  int64  
 7   x        53940 non-null  float64
 8   y        53940 non-null  float64
 9   z        53940 non-null  float64
dtypes: float64(6), int64(1), object(3)
memory usage: 4.1+ MB
```

In [ ]:
```
# CATEGORICAL VARIABLES:
# These are data types that are not numbers. for example, state, color, gender, etc
```

In [46]:
```
print(dataset)
```

```
       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
1       0.21    Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2       0.23       Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3       0.29    Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4       0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
...      ...        ...   ...     ...    ...    ...    ...   ...   ...   ...
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

[53940 rows x 10 columns]
```

In [ ]:
```
# Peeking at the data using .head() and .tail()
```

In [47]:
```
dataset.head() # returns 5 rows if we dont specify number of rows to return
```

Out[47]:

	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
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [48]:
```
dataset.head(20) # returns 20 rows
```

Out[48]:

	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
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
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
8	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
9	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39
10	0.30	Good	J	SI1	64.0	55.0	339	4.25	4.28	2.73
11	0.23	Ideal	J	VS1	62.8	56.0	340	3.93	3.90	2.46
12	0.22	Premium	F	SI1	60.4	61.0	342	3.88	3.84	2.33
13	0.31	Ideal	J	SI2	62.2	54.0	344	4.35	4.37	2.71
14	0.20	Premium	E	SI2	60.2	62.0	345	3.79	3.75	2.27
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
17	0.30	Good	J	SI1	63.4	54.0	351	4.23	4.29	2.70
18	0.30	Good	J	SI1	63.8	56.0	351	4.23	4.26	2.71
19	0.30	Very Good	J	SI1	62.7	59.0	351	4.21	4.27	2.66

In [49]:
```
dataset.tail() # returns the bottom rows
```

Out[49]:

	carat	cut	color	clarity	depth	table	price	x	y	z
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

In [51]:
```
dataset.sample(10) # grab a random sample of rows from the dataset
```

Out[51]:

	carat	cut	color	clarity	depth	table	price	x	y	z
11043	0.25	Ideal	F	VS1	62.1	57.0	595	4.01	4.04	2.50
25694	0.36	Ideal	F	SI1	59.7	57.0	644	4.69	4.72	2.81
3520	0.72	Very Good	G	IF	63.3	54.0	3405	5.64	5.73	3.60
45979	0.30	Premium	E	SI1	61.3	60.0	526	4.27	4.31	2.63
40228	0.40	Ideal	G	VVS1	61.4	55.0	1123	4.72	4.76	2.91
4634	0.74	Ideal	D	VS2	61.4	58.0	3668	5.80	5.83	3.57
20944	1.26	Very Good	G	VVS2	62.1	58.0	9158	6.92	6.83	4.27
53536	0.72	Ideal	E	VS2	62.7	55.0	2691	5.76	5.73	3.60
5916	0.71	Premium	E	VVS1	61.8	56.0	3936	5.71	5.68	3.52
50079	0.51	Ideal	F	VVS2	61.2	56.0	2203	5.19	5.17	3.17

In [52]:
```
# Teaser

dataset.isna().sum() # checking for missing vales in our data
```

Out[52]:
```
carat      0
cut        0
color      0
clarity    0
depth      0
table      0
price      0
x          0
y          0
z          0
dtype: int64
```

In [ ]:
```
# dataframe slicing
```

In [53]:
```
dfx = dataset
```

In [ ]:
```
# Getting one column from the data
```

In [54]:
```
cols
```

Out[54]:
```
['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'price', 'x', 'y', 'z']
```

In [57]:
```
# get onkly the cut column

cutCol = dfx['cut']
```

In [58]:
```
cutCol
```

Out[58]:
```
0            Ideal
1          Premium
2             Good
3          Premium
4             Good
           ...
53935        Ideal
53936         Good
53937    Very Good
53938      Premium
53939        Ideal
Name: cut, Length: 53940, dtype: object
```

In [59]:
```
# Get uniqu number of labels in the cut column


```

```
dfx.cut.unique()
```

Out[59]:
```
array(['Ideal', 'Premium', 'Good', 'Very Good', 'Fair'], dtype=object)
```

In [60]:
```
dfx['cut'].unique()
```

Out[60]:
```
array(['Ideal', 'Premium', 'Good', 'Very Good', 'Fair'], dtype=object)
```

In [ ]:
```
# Challenge: Get all the samples in the datag where cut is of label Premiun
```

In [63]:
```
# getting all diamond sample that are premium

premiumSamples = dfx.loc[dfx['cut'] == 'Premium']
```

In [64]:
```
premiumSamples
```

Out[64]:

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
12	0.22	Premium	F	SI1	60.4	61.0	342	3.88	3.84	2.33
14	0.20	Premium	E	SI2	60.2	62.0	345	3.79	3.75	2.27
15	0.32	Premium	E	I1	60.9	58.0	345	4.38	4.42	2.68
...	...	...	...	...	...	...	...	...	...	...
53928	0.79	Premium	E	SI2	61.4	58.0	2756	6.03	5.96	3.68
53930	0.71	Premium	E	SI1	60.5	55.0	2756	5.79	5.74	3.49
53931	0.71	Premium	F	SI1	59.8	62.0	2756	5.74	5.73	3.43
53934	0.72	Premium	D	SI1	62.7	59.0	2757	5.69	5.73	3.58
53938	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74

13791 rows × 10 columns

In [65]:
```
vGoodSamples = dfx.loc[dfx['cut'] == 'Very Good']
```

```
vGoodSamples
```

In [66]:

Out[66]:

	carat	cut	color	clarity	depth	table	price	x	y	z
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
9	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39
19	0.30	Very Good	J	SI1	62.7	59.0	351	4.21	4.27	2.66
...	...	...	...	...	...	...	...	...	...	...
53921	0.70	Very Good	E	VS2	62.8	60.0	2755	5.59	5.65	3.53
53922	0.70	Very Good	D	VS1	63.1	59.0	2755	5.67	5.58	3.55
53932	0.70	Very Good	E	VS2	60.5	59.0	2757	5.71	5.76	3.47
53933	0.70	Very Good	E	VS2	61.2	59.0	2757	5.69	5.72	3.49
53937	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56

12082 rows × 10 columns

In [72]:
```
priceBelow300 = dfx.loc[dfx['price'] < 400]
```

In [73]:
```
priceBelow300
```

Out[73]:

	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
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...
50612	0.26	Very Good	H	VS1	62.0	56.0	399	4.08	4.11	2.54
50613	0.25	Very Good	D	VS2	61.1	60.0	399	4.07	4.11	2.50
50614	0.27	Very Good	G	VS1	58.8	57.0	399	4.25	4.36	2.53
50615	0.24	Good	F	VS1	62.5	60.0	399	3.96	3.98	2.48
50616	0.25	Good	D	VS1	57.6	60.0	399	4.15	4.18	2.40

247 rows × 10 columns

In [75]:
```
# Get max value of a datacolumn

df.price.min()
```

Out[75]:
```
326
```

In [76]:
```
df.price.max()
```

Out[76]:
```
18823
```

In [82]:
```
dfi = pd.read_csv('C:/workspace/diamonds1.csv')
cols = dfi.columns.tolist()
missingrec = dfi.isna().sum()
dataLen = len(dfi)

for col, rec in zip(cols, missingrec):
    if rec > 0:
        #diff = dataLen - len(dfi[col])
        print(f'{col} is missing {rec} number of samples')
```

```
    else:
        print('no missing data')
```

```
no missing data
no missing data
no missing data
clarity is missing 6 number of samples
depth is missing 11 number of samples
table is missing 25 number of samples
price is missing 17 number of samples
x is missing 13 number of samples
y is missing 7 number of samples
no missing data
```

In [81]:
```
dfi.isna().sum()
```

Out[81]:
```
carat       0
cut         0
color       0
clarity     6
depth      11
table      25
price      17
x          13
y           7
z           0
dtype: int64
```

In [ ]:
```
```

In [85]:
```
def missingValsFinder(dataset):

    missingReport = []
    dfi = dataset
    cols = dfi.columns.tolist()
    missingrec = dfi.isna().sum()
    dataLen = len(dfi)

    for col, rec in zip(cols, missingrec):
        if rec > 0:
            #diff = dataLen - len(dfi[col])
            #print(f'{col} is missing {rec} number of samples')
            missingReport.append(f'{col} is missing {rec} number of samples')
```

```
        else:
            #print('no missing data')
            missingReport.append('no missing data')

    return missingReport
```

In [86]:
```
missingValsFinder(dfi)
```

Out[86]:
```
['no missing data',
 'no missing data',
 'no missing data',
 'clarity is missing 6 number of samples',
 'depth is missing 11 number of samples',
 'table is missing 25 number of samples',
 'price is missing 17 number of samples',
 'x is missing 13 number of samples',
 'y is missing 7 number of samples',
 'no missing data']
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: