In [1]:
```
# imports
import pandas as pd
```

Exercise 2)

1)

We first read the heart dataset.

In [2]:
```
df = pd.read_csv('heart.csv')
df
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1.0	3.0	145.0	233.0	1.0	0.0	150.0	0	2.3	0.0	0.0	1.0	1
1	63	1.0	3.0	145.0	233.0	1.0	0.0	150.0	0	2.3	0.0	0.0	1.0	1
2	37	1.0	2.0	130.0	250.0	0.0	1.0	187.0	0	3.5	0.0	0.0	2.0	1
3	41	0.0	1.0	130.0	204.0	0.0	0.0	172.0	0	1.4	2.0	0.0	2.0	1
4	56	1.0	1.0	120.0	236.0	0.0	1.0	178.0	0	0.8	2.0	0.0	2.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
307	57	0.0	0.0	140.0	241.0	0.0	1.0	123.0	1	0.2	1.0	0.0	3.0	0
308	45	1.0	3.0	110.0	264.0	0.0	1.0	132.0	0	1.2	1.0	0.0	3.0	0
309	68	1.0	0.0	144.0	193.0	1.0	1.0	141.0	0	3.4	1.0	2.0	3.0	0
310	57	1.0	0.0	130.0	131.0	0.0	1.0	115.0	1	1.2	1.0	1.0	3.0	0
311	57	0.0	1.0	130.0	236.0	0.0	0.0	174.0	0	0.0	1.0	1.0	2.0	0

312 rows × 14 columns

As it stated above, it has 312 rows and 14 columns.

2)

We used nunique function to count unique valuse of every attribute.

In [3]:
```
print("number of unique values:")
print("age:\t\t",       df['age'].nunique())
print("sex:\t\t",       df['sex'].nunique())
print("cp:\t\t",        df['cp'].nunique())
print("trestbps:\t",    df['trestbps'].nunique())
print("chol:\t\t",      df['chol'].nunique())
print("fbs:\t\t",       df['fbs'].nunique())
print("restecg:\t",     df['restecg'].nunique())
print("thalach:\t",     df['thalach'].nunique())
print("exang:\t\t",     df['exang'].nunique())
print("oldpeak:\t",     df['oldpeak'].nunique())
print("slope:\t\t",     df['slope'].nunique())
print("ca:\t\t",        df['ca'].nunique())
print("thal:\t\t",      df['thal'].nunique())
print("target:\t\t",    df['target'].nunique())
```

```
number of unique values:
age:            41
sex:            2
cp:             4
trestbps:       49
chol:           152
fbs:            2
restecg:        3
thalach:        91
exang:          2
oldpeak:        40
slope:          3
ca:             5
thal:           4
target:         2
```

3)

We store the duplicated rows in duplicateRows. the dataset has 9 duplicated rows.

In [4]:
```
duplicateRows = df[df.duplicated()]
duplicateRows.shape[0]
```

Out[4]: 9

We drop all duplicate rows and keep the first occurences.

In [5]:
```
wodup = df.drop_duplicates(keep = 'first')
wodup
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1.0	3.0	145.0	233.0	1.0	0.0	150.0	0	2.3	0.0	0.0	1.0	1
2	37	1.0	2.0	130.0	250.0	0.0	1.0	187.0	0	3.5	0.0	0.0	2.0	1
3	41	0.0	1.0	130.0	204.0	0.0	0.0	172.0	0	1.4	2.0	0.0	2.0	1
4	56	1.0	1.0	120.0	236.0	0.0	1.0	178.0	0	0.8	2.0	0.0	2.0	1
5	57	0.0	0.0	120.0	354.0	0.0	1.0	163.0	1	0.6	2.0	0.0	2.0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
307	57	0.0	0.0	140.0	241.0	0.0	1.0	123.0	1	0.2	1.0	0.0	3.0	0
308	45	1.0	3.0	110.0	264.0	0.0	1.0	132.0	0	1.2	1.0	0.0	3.0	0
309	68	1.0	0.0	144.0	193.0	1.0	1.0	141.0	0	3.4	1.0	2.0	3.0	0
310	57	1.0	0.0	130.0	131.0	0.0	1.0	115.0	1	1.2	1.0	1.0	3.0	0
311	57	0.0	1.0	130.0	236.0	0.0	0.0	174.0	0	0.0	1.0	1.0	2.0	0

303 rows × 14 columns

As we saw before, we had 312 rows and 9 of them were duplicated rows. Now that they are deleted, we have 303 unique rows.

4)

We can count the number of missing values in the dataset by isna function

In [6]:
```
df.isna().sum().sum()
```

Out[6]: 17

5)

1. We can delete rows with NaN value.
2. For columns with numeric continuous values, we can replace them with 0 and for columns with categorical values we can replace them with empty string.
3. For columns with numeric continuous values, we can replace them with the mean, median, or mode of remaining values in the column. By this method, we can prevent the loss of data. Also, we can prevent change of mean of columns in case of replacing with mean. For columns with categorical values we can replace them with the most frequent category.
4. We can predict the missing values. We can use the regression or classification model for the prediction of missing values.

We decided to implement the second method, and because all the columns in this dataset have numerical values we can replace all missing values with 0.

In [7]:
```
df = df.fillna(0)
df.isna().sum().sum()
```

Out[7]: 0

As we can see above, all the 17 missig values have replaced by 0 and there are no more NaN value.