

```
In [2]: import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import warnings
```

```
In [3]: df=pd.read_csv('TESTE Data.csv')
df.head()
```

```
Out[3]:
```

	id	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression
0	630000	58	1	3	120	288	0	2	145	1	0.8
1	630001	55	0	2	120	209	0	0	172	0	0.0
2	630002	54	1	4	120	268	0	0	150	1	0.0
3	630003	44	0	3	112	177	0	0	168	0	0.9
4	630004	43	1	1	138	267	0	0	163	0	1.8

```
In [4]: df.dtypes
```

```
Out[4]: id                int64
Age                int64
Sex                int64
Chest pain type    int64
BP                int64
Cholesterol        int64
FBS over 120       int64
EKG results        int64
Max HR            int64
Exercise angina    int64
ST depression      float64
Slope of ST        int64
Number of vessels  int64
Thallium           int64
dtype: object
```

```
In [5]: df.describe()
```

Out[5]:

	id	Age	Sex	Chest pain type	BP
count	270000.000000	270000.000000	270000.000000	270000.000000	270000.000000
mean	764999.500000	54.159870	0.716315	3.315107	130.555089
std	77942.430678	8.255471	0.450787	0.851762	15.007908
min	630000.000000	29.000000	0.000000	1.000000	94.000000
25%	697499.750000	48.000000	0.000000	3.000000	120.000000
50%	764999.500000	54.000000	1.000000	4.000000	130.000000
75%	832499.250000	60.000000	1.000000	4.000000	140.000000
max	899999.000000	77.000000	1.000000	4.000000	200.000000

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

Out[6]:

id	0
Age	0
Sex	0
Chest pain type	0
BP	0
Cholesterol	0
FBS over 120	0
EKG results	0
Max HR	0
Exercise angina	0
ST depression	0
Slope of ST	0
Number of vessels fluro	0
Thallium	0

dtype: int64

In [7]:

```
missing_percentage = df.isnull().mean() * 100
threshold=30
columns_to_drop = missing_percentage[missing_percentage > threshold].index
df=df.drop(columns=columns_to_drop)
print(f"column dropped:{list(columns_to_drop)}")
print(f"Remaining columns:{df.columns}")
```

column dropped:[]
Remaining columns:Index(['id', 'Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol',
 'FBS over 120', 'EKG results', 'Max HR', 'Exercise angina',
 'ST depression', 'Slope of ST', 'Number of vessels fluro', 'Thallium'],
 dtype='object')

In [8]:

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=["object"]).columns:
    df[column] = label_encoder.fit_transform(df[column])
```

In [13]: `df.head()`

Out[13]:

	id	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression
0	630000	58	1	3	120	288	0	2	145	1	0.8
1	630001	55	0	2	120	209	0	0	172	0	0.0
2	630002	54	1	4	120	268	0	0	150	1	0.0
3	630003	44	0	3	112	177	0	0	168	0	0.9
4	630004	43	1	1	138	267	0	0	163	0	1.8

```
In [14]: for column in df.select_dtypes(include=['float64']).columns:
          mean_value=df[column].mean()
          df[column].fillna(mean_value)
          for column in df.select_dtypes(include=['object']).columns:
            mode_value= df[column].mode[0]
            df[column].fillna(mode_value)
```

```
In [15]: data=df.copy()
          data.head()
```

Out[15]:

	id	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression
0	630000	58	1	3	120	288	0	2	145	1	0.8
1	630001	55	0	2	120	209	0	0	172	0	0.0
2	630002	54	1	4	120	268	0	0	150	1	0.0
3	630003	44	0	3	112	177	0	0	168	0	0.9
4	630004	43	1	1	138	267	0	0	163	0	1.8

```
In [10]: data=df.copy()
def detect_outlier_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5*IQR
    return(column < lower_bound) | (column > upper_bound)
for col in data.columns:
    data[f'{col}_outlier'] = detect_outlier_iqr(data[col])
    outlier_only = data[data.filter(like='outlier').any(axis=1)]
```

```
In [16]: data.head(10)
```

Out[16]:

	id	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression
0	630000	58	1	3	120	288	0	2	145	1	0.8
1	630001	55	0	2	120	209	0	0	172	0	0.0
2	630002	54	1	4	120	268	0	0	150	1	0.0
3	630003	44	0	3	112	177	0	0	168	0	0.9
4	630004	43	1	1	138	267	0	0	163	0	1.8
5	630005	61	1	4	130	254	0	2	139	1	1.4
6	630006	48	0	2	120	330	0	2	166	0	0.0
7	630007	57	1	4	142	282	0	0	153	0	0.0
8	630008	60	1	4	130	249	0	2	131	1	0.4
9	630009	43	0	3	120	250	0	2	160	0	0.9

In [12]: `data.dtypes`

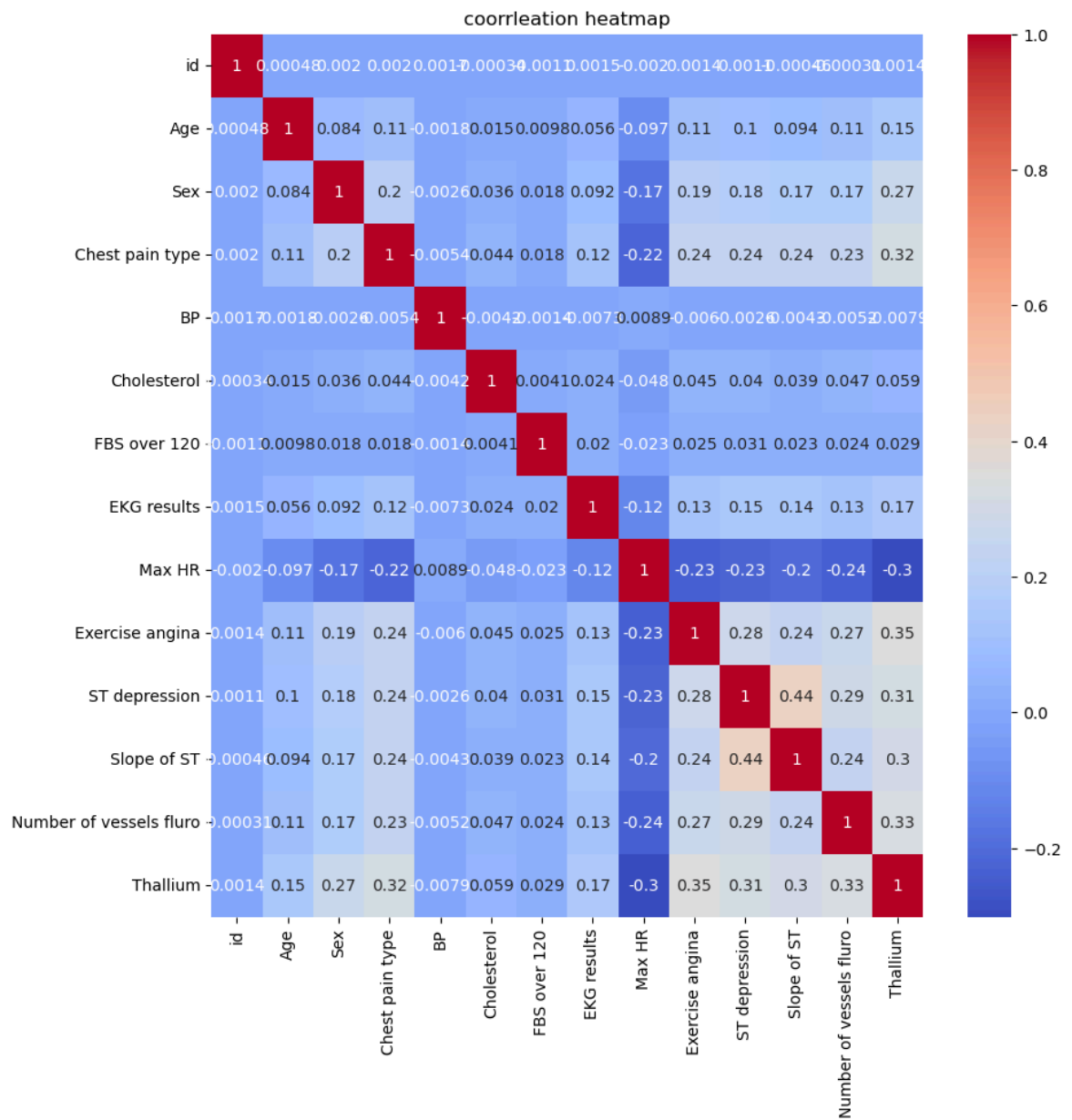
```
Out[12]: id                int64
Age                int64
Sex                int64
Chest pain type    int64
BP                 int64
Cholesterol         int64
FBS over 120        int64
EKG results         int64
Max HR             int64
Exercise angina     int64
ST depression       float64
Slope of ST         int64
Number of vessels fluro int64
Thallium            int64
dtype: object
```

In [13]: `data.isnull().sum()`

```
Out[13]: id                0
Age                0
Sex                0
Chest pain type    0
BP                 0
Cholesterol         0
FBS over 120        0
EKG results         0
Max HR             0
Exercise angina     0
ST depression       0
Slope of ST         0
Number of vessels fluro 0
Thallium            0
dtype: int64
```

In [15]: `plt.figure(figsize=(10, 10))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')`

```
plt.title('coorlleation heatmap')
plt.show()
```



In []: