In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:

```
df = pd.read_csv("heart.csv")
df
```

Out[5]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	outp
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

303 rows × 14 columns

There are 303 row and 14 columns as shown. We can see the first and last 5 datas of our dataset.

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trtbps    303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalachh  303 non-null    int64  
 8   exng      303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64  
 11  caa       303 non-null    int64  
 12  thall     303 non-null    int64  
 13  output    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

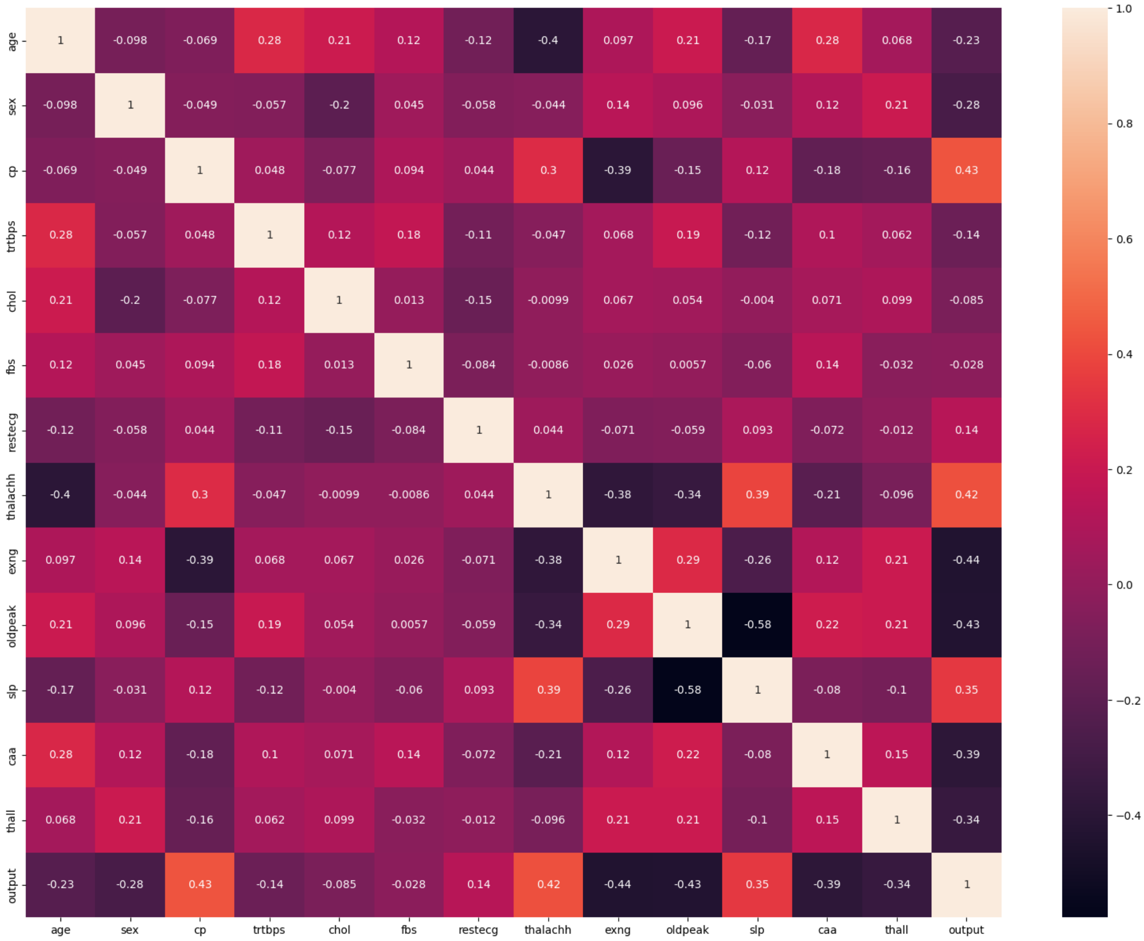
As we see, there is no null data in any columns. All colummns has 303 non-null datas. Our target column will be the "output" which mean "y" in our model.

In [7]:

```
plt.figure(figsize=(20,15))
sns.heatmap(df.corr(),annot=True)
```

Out[7]:

```
<AxesSubplot: >
```



"trtbps", "chol", "fbs" columns' correlation with output is lower than 0.15, so lets drop these columns.

In [8]:

```
df.drop(columns=["trtbps", "chol", "fbs"], inplace=True)
df
```

Out[8]:

	age	sex	cp	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	0	150	0	2.3	0	0	1	1
1	37	1	2	1	187	0	3.5	0	0	2	1
2	41	0	1	0	172	0	1.4	2	0	2	1
3	56	1	1	1	178	0	0.8	2	0	2	1
4	57	0	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	1	123	1	0.2	1	0	3	0
299	45	1	3	1	132	0	1.2	1	0	3	0
300	68	1	0	1	141	0	3.4	1	2	3	0
301	57	1	0	1	115	1	1.2	1	1	3	0
302	57	0	1	0	174	0	0.0	1	1	2	0

303 rows × 11 columns

Now, let's scale our numeric datas.

In [9]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df.oldpeak = scaler.fit_transform(df.oldpeak.values.reshape(-1,1))
df.thalachh = scaler.fit_transform(df.thalachh.values.reshape(-1,1))
```

In [10]:

```
y = df.pop("output")
X = df
```

In [11]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42
```

In [ ]:

```
from sklearn.linear_model import LogisticRegression
logistic = LogisticRegression()
logistic.fit(X_train,y_train)
test_pred = logistic.predict(X_test)
train_pred = logistic.predict(X_train)
```

In [13]:

```
from sklearn.metrics import accuracy_score
print("Train score: ", accuracy_score(y_train,train_pred)*100)
print("Test score: ", accuracy_score(y_test,test_pred)*100)
```

```
Train score:  85.9504132231405
Test score:  88.52459016393442
```

The scores seem so nice, it's about 85-88% and test train scores close each other. We can say there is no over or underfitting problem.

In [14]:

```
y.value_counts()
```

Out[14]:

```
1    165
0    138
Name: output, dtype: int64
```

Our target column has two metric which are 1(more chance of heart attack) and 0(less chance of heart attack). Our data types so close each other, so the data seem balanced.

In [18]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, test_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.86      0.88        29
           1       0.88      0.91      0.89        32

    accuracy                           0.89        61
   macro avg       0.89      0.88      0.88        61
weighted avg       0.89      0.89      0.89        61
```

In [19]:

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,test_pred))
```

```
[[25  4]
 [ 3 29]]
```