Heart Attack Prediction

Information about Data

Columns is ::

1. -Age is the age of candidate.
2. -Sex has numeric values. 1 is male and 0 is female.
3. -(cp)Chest Pain pain has values between 0-3. The types of angina that are described in the research paper. The higher the number, the lesser are the odds of heart attack. — Value 1: typical angina — Value 2: atypical angina — Value 3: non-anginal pain
4. -(trestbps)Resting blood pressure is normal pressure with no exercise.
5. -(chol)Cholesterol means the blockage for blood supply in the blood vessels.
6. -(fbs)fasting blood sugar > 120 mg/dl (1 = true; 0 = false) blood sugar taken after a long gap between a meal and the test. Typically, it's taken before any meal in the morning.
7. -(restecg)Rest ECG results means ECG values taken while person is on rest which means no exercise and normal functioning of heart is happening.
8. -(thalach)The Maximum Heart Rate achieved.
9. -(exang): exercise induced angina (1 = yes; 0 = no) is chest pain while exercising or doing any physical activity.
10. -(oldpeak)ST Depression is the difference between value of ECG at rest and after exercise.
11. -(slope): the slope of the peak exercise ST segment — Value 1: upsloping — Value 2: flat — Value 3: downsloping
12. (ca): The number of major blood vessels(0-3) supplying blood to heart blocked.
13. -(thal):The Types of thalassemia (3 = normal; 6 = fixed defect; 7 = reversable defect)
14. -(target) (predicted attribute): diagnosis of heart disease (angiographic disease status) — Value 0: < 50% diameter narrowing — Value 1: > 50% diameter narrowing

**Heart attack prediction where 1 denotes Heart attack occured and 0 where it din't take occur.

Important libraries

In [1]:
```
#For uploading and accessing the data
import numpy as np
import pandas as pd
#For visualizations
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
```

Read Data

In [2]:
```
data = pd.read_csv('../input/heart-disease-uci/heart.csv' , sep=',' , encoding='u
```

In [3]:
```
#show information about data
data.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   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [4]:
```
data.shape
```

Out[4]:
```
(303, 14)
```

- -Data Containing of 303 row and 14 columns(features)
- -No found Null
- -all data is integer except oldpeak is float

In [5]:
```
#show first 5 row from data
data.head()
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [6]:
```
data.tail()
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

Renaming the column headers for better understanding of visualizations.

In [7]:
```
data.rename(columns = {'age':'Age','sex':'Gender','cp':'Chest_pain' ,'trestbps':'
                       'restecg':'ECG_results','thalach':'Maximum_heart_rate','exang
                      'thal':'Thalassemia_types','target':'Heart_attack','slope':'ST
```

In [8]:
```
#show heading of columns
data.head()
```

Out[8]:

	Age	Gender	Chest_pain	Resting_blood_pressure	Cholesterol	Fasting_blood_sugar	ECG_resul
0	63	1	3	145	233	1	
1	37	1	2	130	250	0	
2	41	0	1	130	204	0	
3	56	1	1	120	236	0	
4	57	0	0	120	354	0	

In [9]:
```
data.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   Gender                   303 non-null    int64  
 2   Chest_pain               303 non-null    int64  
 3   Resting_blood_pressure   303 non-null    int64  
 4   Cholesterol              303 non-null    int64  
 5   Fasting_blood_sugar      303 non-null    int64  
 6   ECG_results              303 non-null    int64  
 7   Maximum_heart_rate       303 non-null    int64  
 8   Exercise_induced_angina  303 non-null    int64  
 9   ST_depression            303 non-null    float64
 10  ST_slope                 303 non-null    int64  
 11  Major_vessels            303 non-null    int64  
 12  Thalassemia_types        303 non-null    int64  
 13  Heart_attack             303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [10]:
```
#show sum of null data
data.isnull().sum()
#also no found null
```

Out[10]:
```
Age                        0
Gender                     0
Chest_pain                 0
Resting_blood_pressure     0
Cholesterol                0
Fasting_blood_sugar        0
ECG_results                0
Maximum_heart_rate         0
Exercise_induced_angina    0
ST_depression              0
ST_slope                   0
Major_vessels              0
Thalassemia_types          0
Heart_attack               0
dtype: int64
```

In [11]:
```
#show all statistics
data.describe(include='all')
```

Out[11]:

	Age	Gender	Chest_pain	Resting_blood_pressure	Cholesterol	Fasting_blood_sug
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.0000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.1485
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.3561
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.0000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.0000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.0000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.0000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.0000

In [12]:
```
#replace number to object to make reading is easy
#data['Gender'].replace({1:'Male' , 0:'Female'},inplace = True)
```

In [13]:
```
#replace number to object to make reading is easy
#data['Heart_attack'].replace({1:'Heart_attack-Yes' , 0:'Heart_attack-No'} ,inpla
```

In [14]:
```
data.head()
```

Out[14]:

	Age	Gender	Chest_pain	Resting_blood_pressure	Cholesterol	Fasting_blood_sugar	ECG_resul
0	63	1	3	145	233	1	
1	37	1	2	130	250	0	
2	41	0	1	130	204	0	
3	56	1	1	120	236	0	
4	57	0	0	120	354	0	

In [15]:
```
data.tail()
```

Out[15]:

	Age	Gender	Chest_pain	Resting_blood_pressure	Cholesterol	Fasting_blood_sugar	ECG_res
298	57	0	0	140	241	0	
299	45	1	3	110	264	0	
300	68	1	0	144	193	1	
301	57	1	0	130	131	0	
302	57	0	1	130	236	0	

In [16]:
```
#GAH refer to gender , age and has Heart_attack
#count of member has Heart_attack according his gender and age
GAH = data.groupby(['Gender','Age'])['Heart_attack'].count().reset_index().sort_v
GAH.head(20).style.background_gradient(cmap='Purples')
```

Out[16]:

	Gender	Age	Heart_attack
59	1	58	13
58	1	57	13
60	1	59	13
53	1	52	12
55	1	54	11
45	1	44	9
52	1	51	8
57	1	56	8
61	1	60	7
65	1	64	7
24	0	62	7
43	1	42	6
49	1	48	6
62	1	61	6
44	1	43	6
68	1	67	6
20	0	58	6
42	1	41	6
25	0	63	5
54	1	53	5

- 1:Male
- 0:Female
- top 20 count of heart attack, males have witnessed more heart attacks for their ages.**

In [17]:
```
#GcH refer to gender , Chest_pain and has Heart_attack
#count of member has Heart_attack according his Chest_pain and Gender
GcH = data.groupby(['Gender' , 'Chest_pain'])['Heart_attack'].count().reset_index
GcH.head(20).style.background_gradient(cmap='Blues')
```

Out[17]:

	Gender	Chest_pain	Heart_attack
4	1	0	104
6	1	2	52
0	0	0	39
2	0	2	35
5	1	1	32
7	1	3	19
1	0	1	18
3	0	3	4

- The higher the number, the lesser are the odds of heart attack
- shows that large number of Male has Heart_Attack with Chest_pain less
- just 19 Man has Heart_Attack with Chest_pain higher
- for women 4 women has Heart_Attack with Chest_pain higher
- 39 women has Heart_Attack with Chest_pain less

In [18]:
```
data.columns
```

Out[18]:
```
Index(['Age', 'Gender', 'Chest_pain', 'Resting_blood_pressure', 'Cholesterol',
       'Fasting_blood_sugar', 'ECG_results', 'Maximum_heart_rate',
       'Exercise_induced_angina', 'ST_depression', 'ST_slope', 'Major_vessels',
       'Thalassemia_types', 'Heart_attack'],
      dtype='object')
```

In [19]:
```
data.Chest_pain
```

Out[19]:
```
0      3
1      2
2      1
3      1
4      0
      ..
298    0
299    3
300    0
301    0
302    1
Name: Chest_pain, Length: 303, dtype: int64
```

In [20]:
```
#GRH refer to gender , Resting_blood_pressure and has Heart_attack
#count of member has Heart_attack according his Resting_blood_pressure and Gender
GRH = data.groupby(['Gender' , 'Resting_blood_pressure'])['Heart_attack'].count()
GRH.head(8).style.background_gradient(cmap='coolwarm')
```

Out[20]:

	Gender	Resting_blood_pressure	Heart_attack
46	1	120	27
54	1	130	24
60	1	140	22
40	1	110	16
15	0	130	12
50	1	125	11
21	0	140	10
66	1	150	10

count of Male 16 has Resting_blood_pressure 110 but count of women 12 has Resting_blood_pressure 130

In [21]:
```
#GCH refer to gender , Cholesterol and has Heart_attack
#count of member has Heart_attack according his Cholesterol and Gender
GCH = data.groupby(['Gender' , 'Cholesterol'])['Heart_attack'].count().reset_inde
GCH.head(10).style.background_gradient(cmap='OrRd')
```

Out[21]:

	Gender	Cholesterol	Heart_attack
115	1	212	5
109	1	204	4
135	1	234	4
134	1	233	4
172	1	282	4
150	1	254	4
48	0	269	4
144	1	246	3
143	1	245	3
141	1	243	3

- count of Male with high Cholesterol less (212 -5)
- for female Cholesterol was high and count of persone has Heart_attack is smaller

In [22]:
```
#GFH refer to gender , Fasting_blood_sugar and has Heart_attack
#count of member has Heart_attack according his Fasting_blood_sugar and Gender
GFH = data.groupby(['Gender' , 'Fasting_blood_sugar'])['Heart_attack'].count().re
GFH.style.background_gradient(cmap='YlGn')
```

Out[22]:

	Gender	Fasting_blood_sugar	Heart_attack
2	1	0	174
0	0	0	84
3	1	1	33
1	0	1	12

When Fasting_blood_sugar was False --Heart_attack was highe for Male also with women

In [23]:
```
data.Fasting_blood_sugar.shape
```

Out[23]: (303,)

In [24]:
```
#GECH refer to gender , ECG_results and has Heart_attack
#count of member has Heart_attack according his ECG_results and Gender
GECH = data.groupby(['Gender' , 'ECG_results'])['Heart_attack'].count().reset_inc
GECH.style.background_gradient(cmap='bone')
```

Out[24]:

	Gender	ECG_results	Heart_attack
3	1	0	103
4	1	1	103
1	0	1	49
0	0	0	44
2	0	2	3
5	1	2	1

- When ECG_results is small was Heart_attack is highe
- Male higher from Female

In [25]:
```
#GEH refer to gender , Maximum_heart_rate and has Heart_attack
#count of member has Heart_attack according his Maximum_heart_rate and Gender
GEH = data.groupby(['Gender' , 'Maximum_heart_rate'])['Heart_attack'].count().res
GEH.head(10).style.background_gradient(cmap='summer')
```

Out[25]:

	Gender	Maximum_heart_rate	Heart_attack
86	1	144	7
77	1	132	7
103	1	162	7
113	1	173	7
82	1	140	6
42	0	172	6
91	1	150	6
70	1	125	6
97	1	156	6
34	0	163	5

The maximum heart rate were higher for males resulting in heart attack

In [26]:
```
#GEiH refer to gender , Exercise_induced_angina and has Heart_attack
#count of member has Heart_attack according his Exercise_induced_angina and Gende
GEiH = data.groupby(['Gender' , 'Exercise_induced_angina'])['Heart_attack'].count
GEiH.style.background_gradient(cmap='cool')
```

Out[26]:

	Gender	Exercise_induced_angina	Heart_attack
2	1	0	130
3	1	1	77
0	0	0	74
1	0	1	22

Exercise induced chest pain was higher in males and more resulted in Heart attacks.

In [27]:
```
#GSH refer to gender , ST_depression and has Heart_attack
#count of member has Heart_attack according his ST_depression and Gender
GSH = data.groupby(['Gender' , 'ST_depression'])['Heart_attack'].count().reset_ir
GSH.head(10).style.background_gradient(cmap='Oranges')
```

Out[27]:

	Gender	ST_depression	Heart_attack
27	1	0.000000	64
0	0	0.000000	35
37	1	1.200000	13
34	1	0.800000	10
36	1	1.000000	9
41	1	1.800000	8
38	1	1.400000	8
33	1	0.600000	8
43	1	2.000000	7
40	1	1.600000	7

The lower the depression, the higher the cases were for heart attack.

In [28]:
```
data.ST_depression.head()
```

Out[28]:
```
0    2.3
1    3.5
2    1.4
3    0.8
4    0.6
Name: ST_depression, dtype: float64
```

In [29]:
```
#GSlH refer to gender , ST_slope and has Heart_attack
#count of member has Heart_attack according his ST_slope and Gender
GSlH = data.groupby(['Gender' , 'ST_slope'])['Heart_attack'].count().reset_index(
GSlH.head(10).style.background_gradient(cmap='afmhot')
```

Out[29]:

	Gender	ST_slope	Heart_attack
5	1	2	96
4	1	1	95
2	0	2	46
1	0	1	45
3	1	0	16
0	0	0	5

- The lower ST_slope (2 less than 1 less than 0), the higher the cases were for heart attack.
- The higher the slope value, the higher were the cases for Heart attack

In [30]:
```
#GMH refer to gender , Major_vessels and has Heart_attack
#count of member has Heart_attack according his Major_vessels and Gender
GMH = data.groupby(['Gender' , 'Major_vessels'])['Heart_attack'].count().reset_ir
GMH.head(10).style.background_gradient(cmap='afmhot')
```

Out[30]:

	Gender	Major_vessels	Heart_attack
4	1	0	111
0	0	0	64
5	1	1	50
6	1	2	25
7	1	3	16
1	0	1	15
2	0	2	13
8	1	4	5
3	0	3	4

- The lower the number of vessels blocked, the higher were the heart attack cases.
- This means that 0 represents that all 4 major blood vessels were blocked
- and 4 represented all vessels were free for flow.
- Male > Female

In [31]:
```
#GTH refer to gender , Thalassemia_types and has Heart_attack
#count of member has Heart_attack according his Thalassemia_types and Gender
GTH = data.groupby(['Gender' , 'Thalassemia_types'])['Heart_attack'].count().rese
GTH.head(20).style.background_gradient(cmap='GnBu')
```

Out[31]:

	Gender	Thalassemia_types	Heart_attack
7	1	3	102
6	1	2	87
2	0	2	79
5	1	1	17
3	0	3	15
0	0	0	1
1	0	1	1
4	1	0	1

The higher the Thalassemia type, the higher were the cases of heart attack.

In [32]:
```
data.columns
```

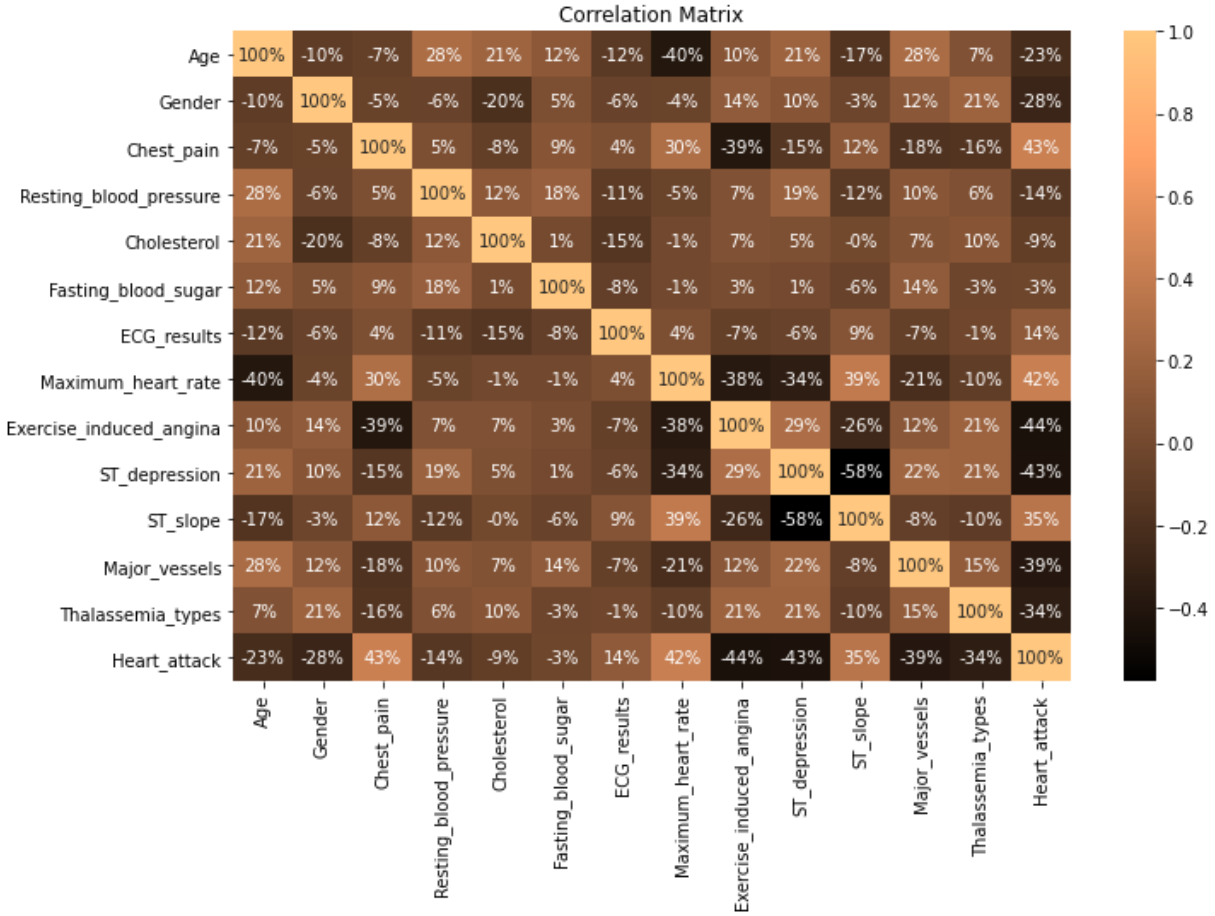
Out[32]:
```
Index(['Age', 'Gender', 'Chest_pain', 'Resting_blood_pressure', 'Cholesterol',
       'Fasting_blood_sugar', 'ECG_results', 'Maximum_heart_rate',
       'Exercise_induced_angina', 'ST_depression', 'ST_slope', 'Major_vessels',
       'Thalassemia_types', 'Heart_attack'],
      dtype='object')
```

In [33]:
```
data_corr=data.corr().style.background_gradient(cmap='plasma')
data_corr
```

Out[33]:

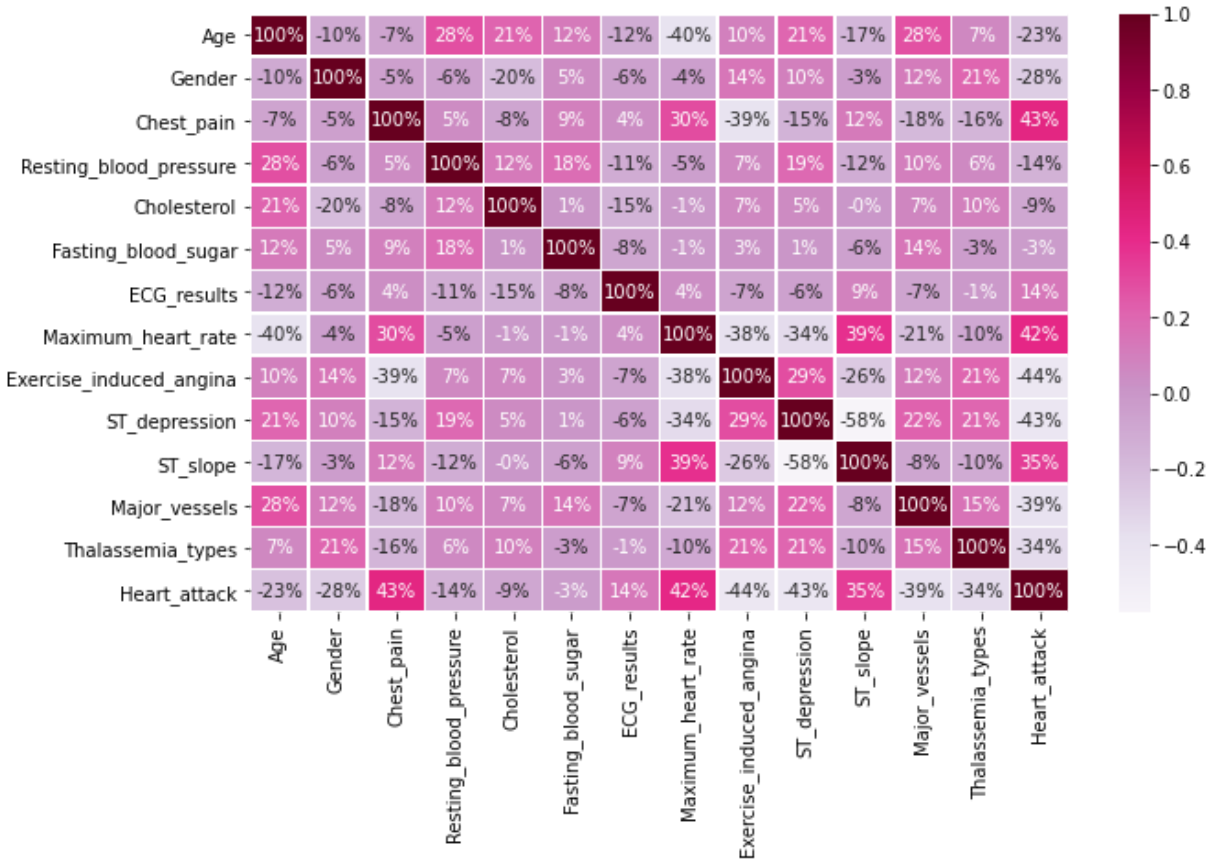
	Age	Gender	Chest_pain	Resting_blood_pressure	Cholesterol	Fa
Age	1.000000	-0.098447	-0.068653	0.279351	0.213678	
Gender	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	
Chest_pain	-0.068653	-0.049353	1.000000	0.047608	-0.076904	
Resting_blood_pressure	0.279351	-0.056769	0.047608	1.000000	0.123174	
Cholesterol	0.213678	-0.197912	-0.076904	0.123174	1.000000	
Fasting_blood_sugar	0.121308	0.045032	0.094444	0.177531	0.013294	
ECG_results	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	
Maximum_heart_rate	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	
Exercise_induced_angina	0.096801	0.141664	-0.394280	0.067616	0.067023	
ST_depression	0.210013	0.096093	-0.149230	0.193216	0.053952	
ST_slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	
Major_vessels	0.276326	0.118261	-0.181053	0.101389	0.070511	
Thalassemia_types	0.068001	0.210041	-0.161736	0.062210	0.098803	
Heart_attack	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	

In [34]:
```
#corelation matrix with all Data
plt.figure(figsize=(11,7))
sns.heatmap(cbar=True,annot=True,fmt=".0%",data=data.corr(),cmap='copper')
plt.title('Correlation Matrix')
plt.show()
```

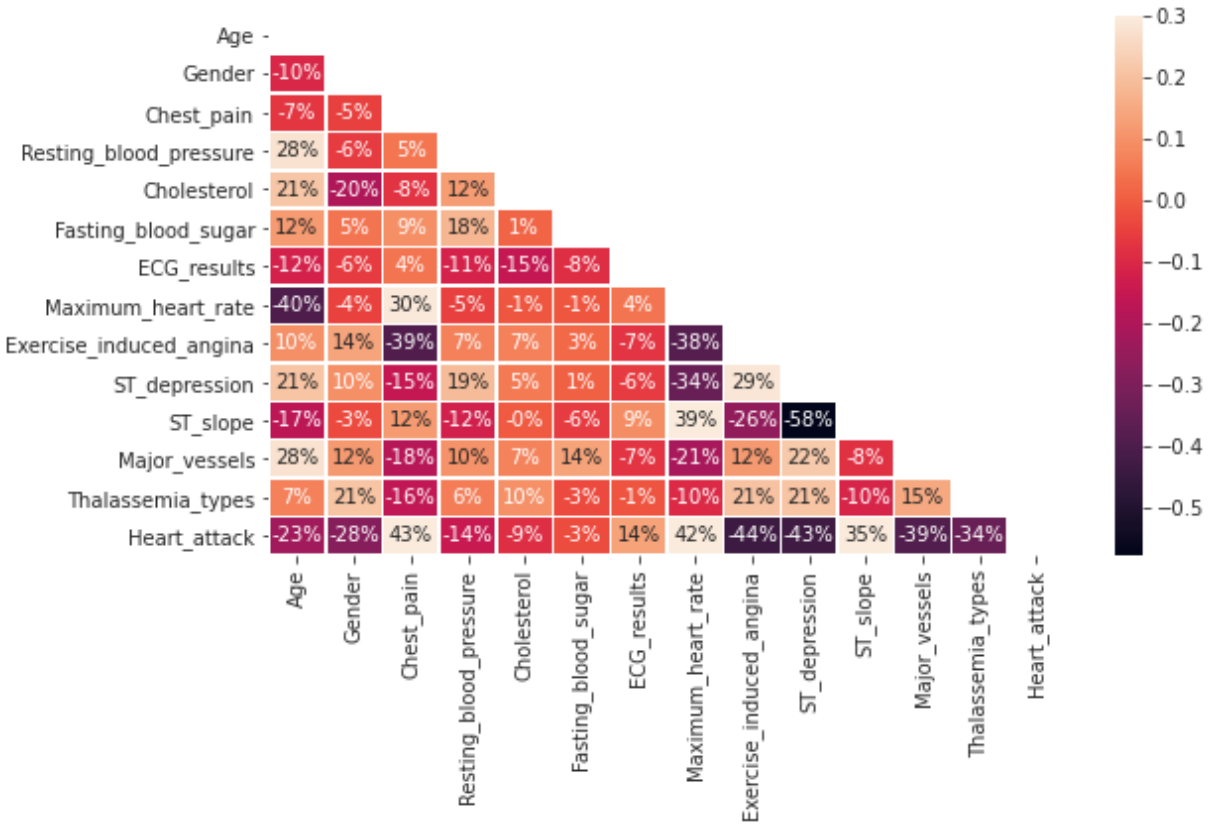


In [35]:
```
#corelation matrix with important Data
#data.drop(['Cholesterol' , 'Resting_blood_pressure', 'ECG_results' , 'Fasting_bl
  #       ,axis = 'columns' , inplace = True)
plt.figure(figsize=(10,6))
sns.heatmap(data.corr() , annot=True,fmt=".0%",linewidth=0.5, cmap='PuRd')#linewi
```

Out[35]: <AxesSubplot:>



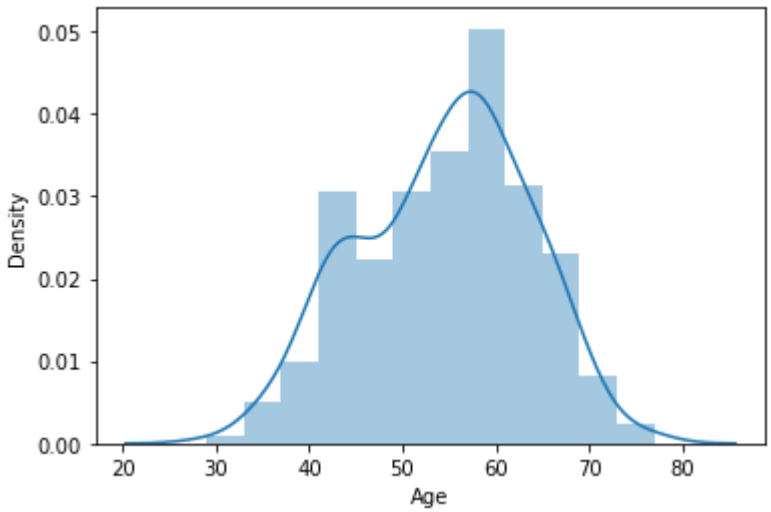
In [36]:
```
mask = np.zeros_like(data.corr())
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("ticks"):
    f, ax = plt.subplots(figsize=(9, 5))
    ax = sns.heatmap(data.corr(), mask=mask, vmax=.3,annot=True,fmt=".0%",linewic
    #annot is numbers in squares
    #square=True the square is small but square=False the square is large
```



In [37]:
```
#Show histogram for Age
sns.distplot(data['Age'])
```

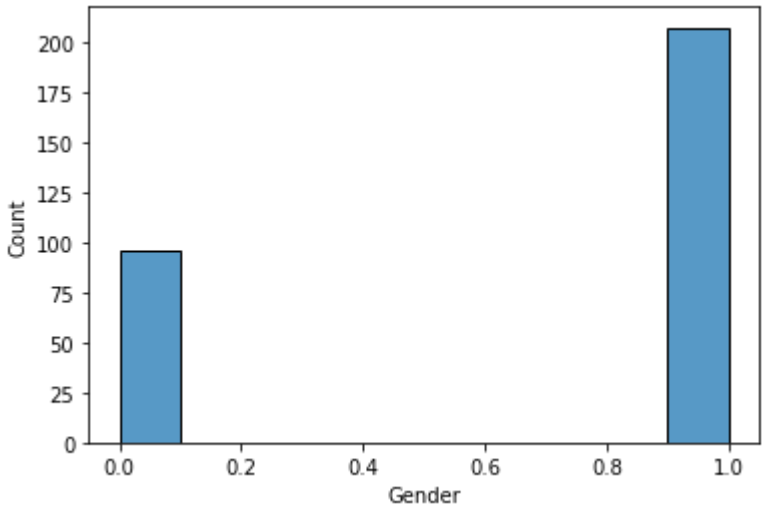
```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: FutureWar
ning: `distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level function wit
h similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[37]: <AxesSubplot:xlabel='Age', ylabel='Density'>



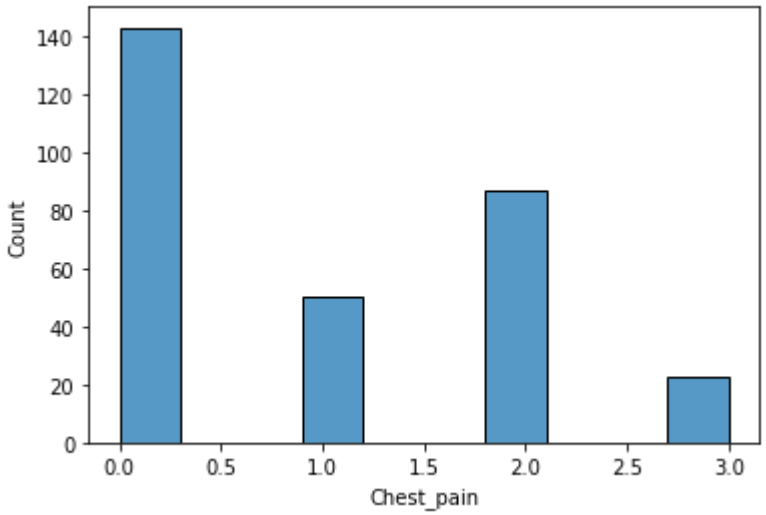
In [38]:
```
#Show histogram for Gender
sns.histplot(data['Gender'])
```

Out[38]: <AxesSubplot:xlabel='Gender', ylabel='Count'>



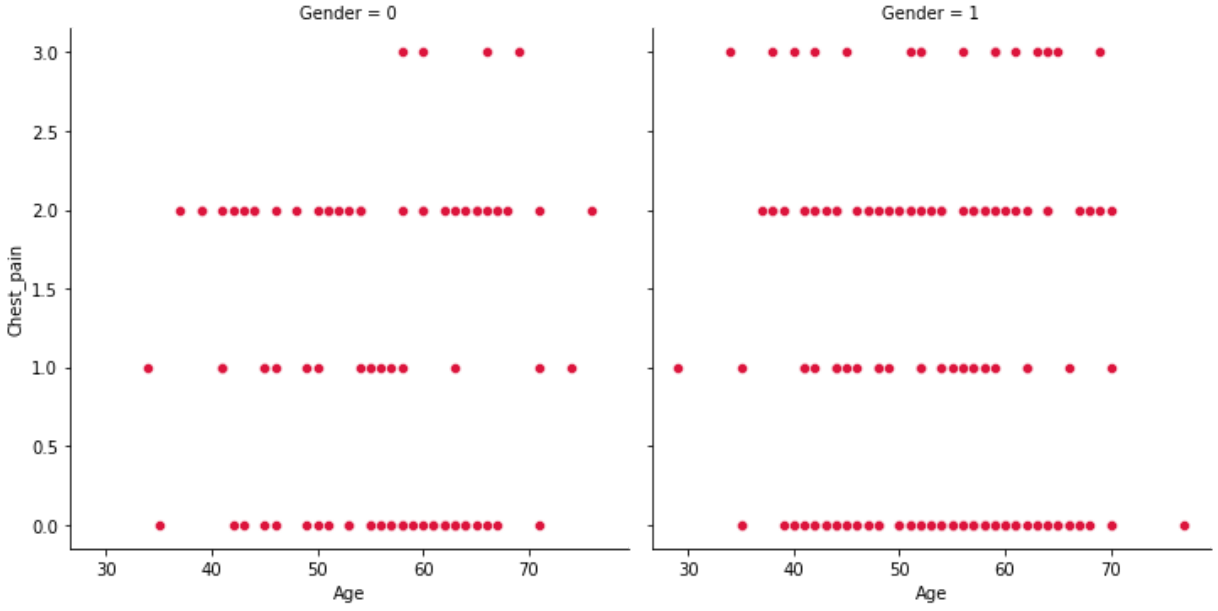
In [39]:
```
#Show histogram for Chest_pain
sns.histplot(data['Chest_pain'])
```

Out[39]:
```
<AxesSubplot:xlabel='Chest_pain', ylabel='Count'>
```



In [40]:
```
sns.relplot(x ='Age', y ='Chest_pain', col = 'Gender', data =data, color = 'crims
```

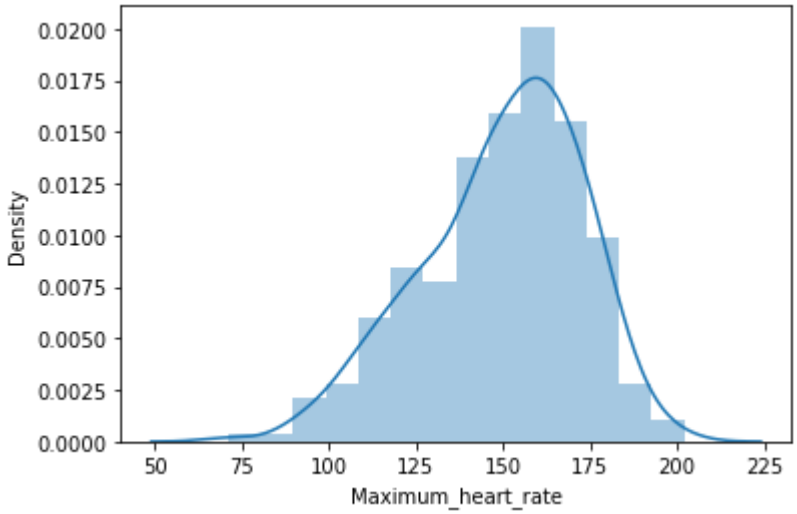
Out[40]:
```
<seaborn.axisgrid.FacetGrid at 0x7f82fb0baa50>
```



In [41]:
```
#Show histogram for Maximum_heart_rate
sns.distplot(data['Maximum_heart_rate'])
```

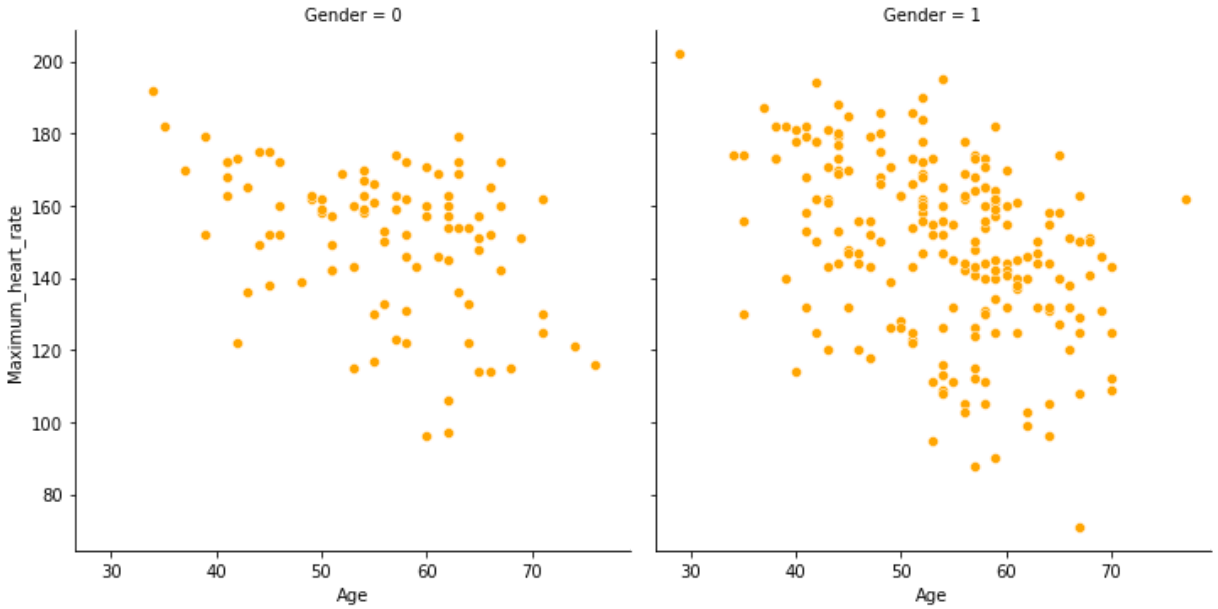
```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: FutureWar
ning: `distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level function wit
h similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[41]: <AxesSubplot:xlabel='Maximum_heart_rate', ylabel='Density'>



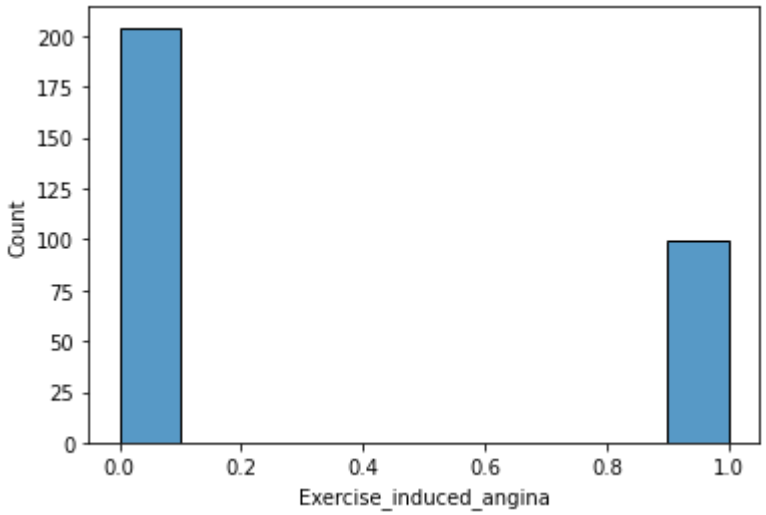
In [42]:
```
sns.relplot(x ='Age', y ='Maximum_heart_rate', col = 'Gender', data =data, color
```

Out[42]:
```
<seaborn.axisgrid.FacetGrid at 0x7f82f81c1310>
```



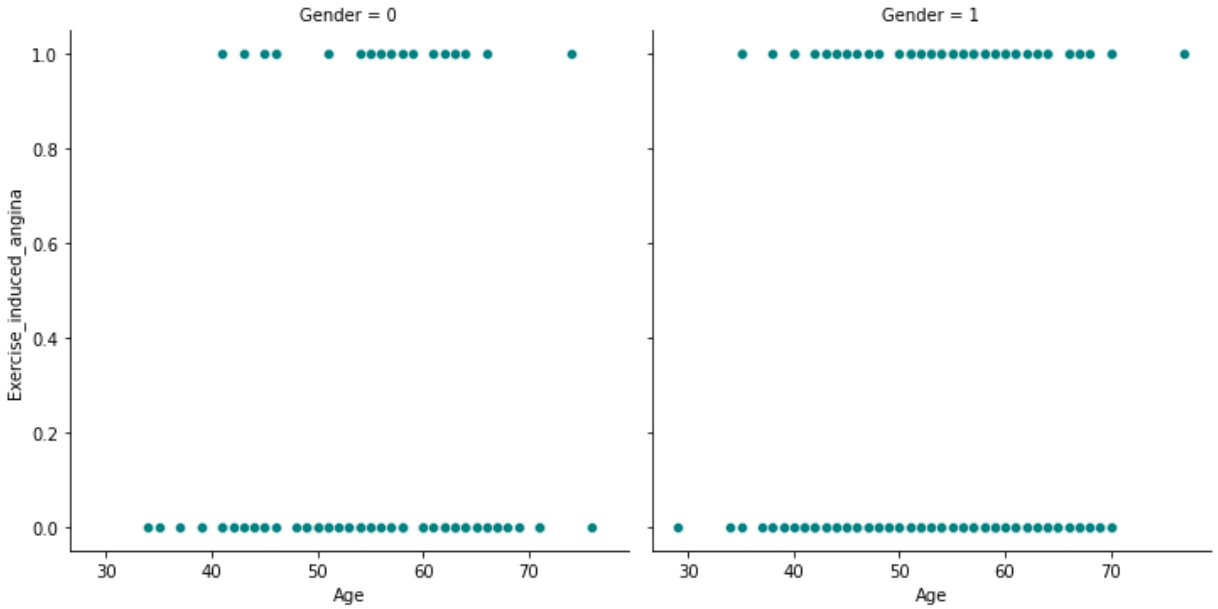
In [43]:
```
#Show histogram for Exercise_induced_angina
sns.histplot(data['Exercise_induced_angina'])
```

Out[43]:
```
<AxesSubplot:xlabel='Exercise_induced_angina', ylabel='Count'>
```



In [44]:
```
sns.relplot(x ='Age', y ='Exercise_induced_angina', col = 'Gender', data =data, c
```

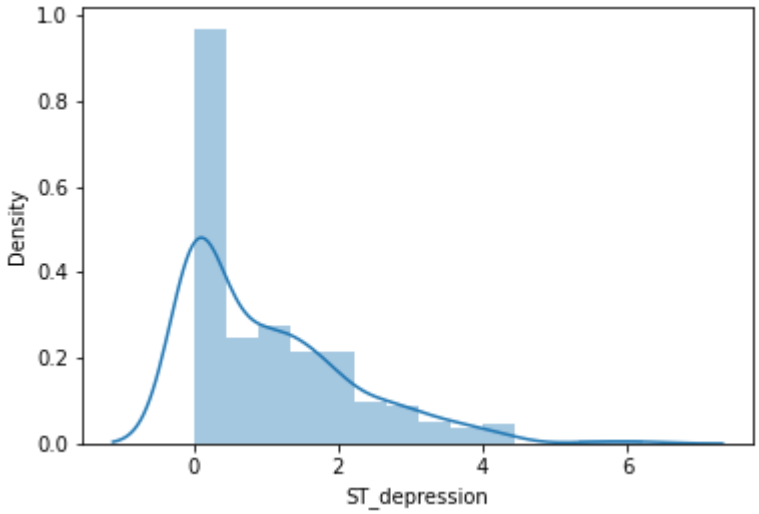
Out[44]:
```
<seaborn.axisgrid.FacetGrid at 0x7f82fa37cb10>
```



In [45]:
```
#Show histogram for ST_depression
sns.distplot(data['ST_depression'])
```

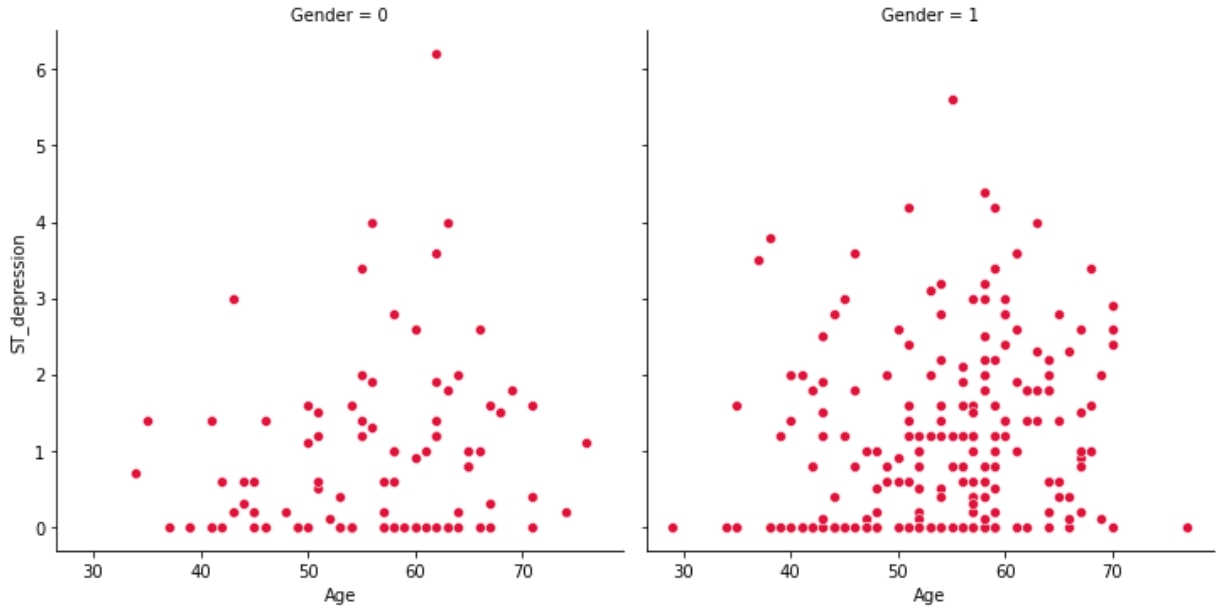
```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: FutureWar
ning: `distplot` is a deprecated function and will be removed in a future versi
on. Please adapt your code to use either `displot` (a figure-level function wit
h similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[45]: <AxesSubplot:xlabel='ST_depression', ylabel='Density'>



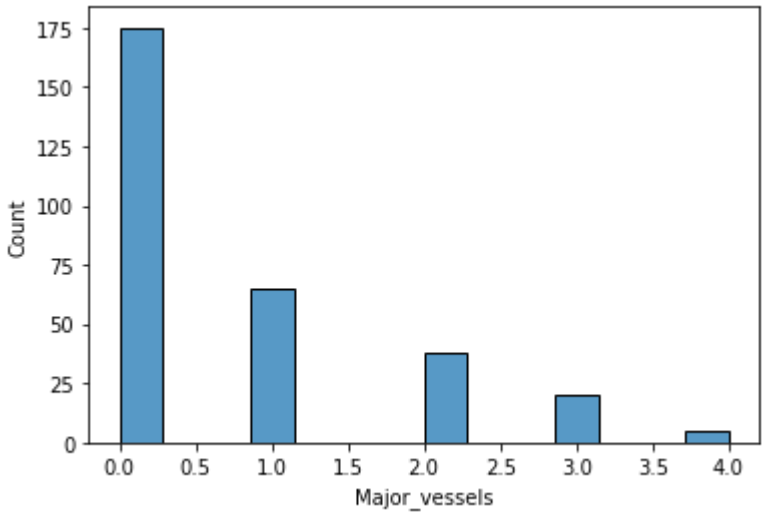
In [46]:
```
sns.relplot(x ='Age', y ='ST_depression', col = 'Gender', data =data, color = 'cr
```

Out[46]:
```
<seaborn.axisgrid.FacetGrid at 0x7f82fa0c7e10>
```



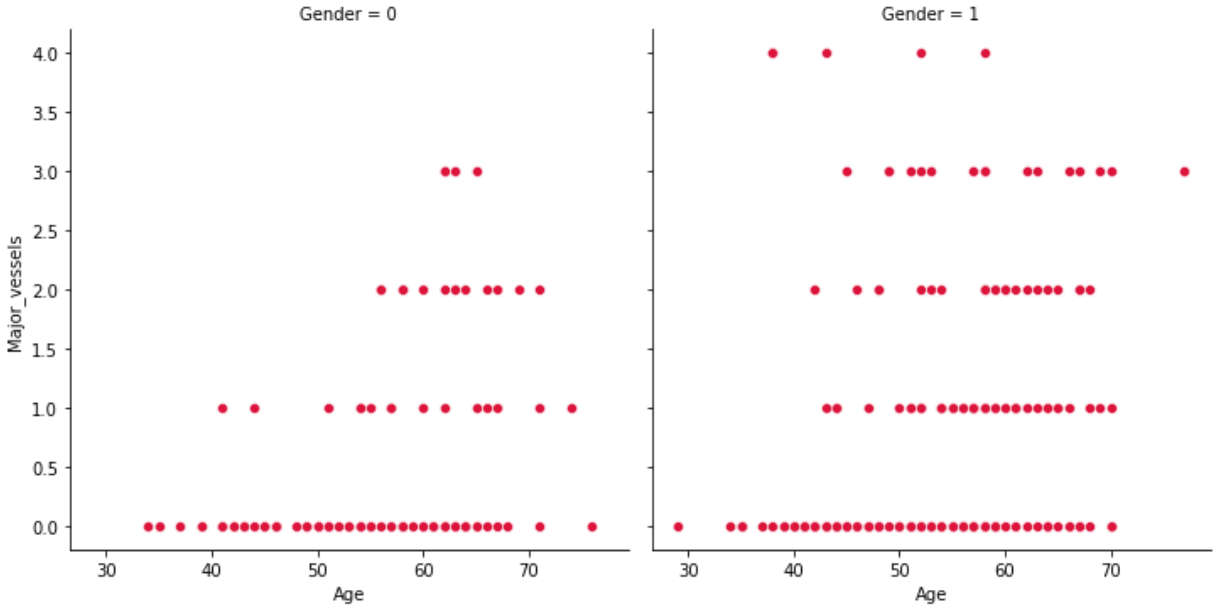
In [47]:
```
#Show histogram for Major_vessels
sns.histplot(data['Major_vessels'])
```

Out[47]:
```
<AxesSubplot:xlabel='Major_vessels', ylabel='Count'>
```



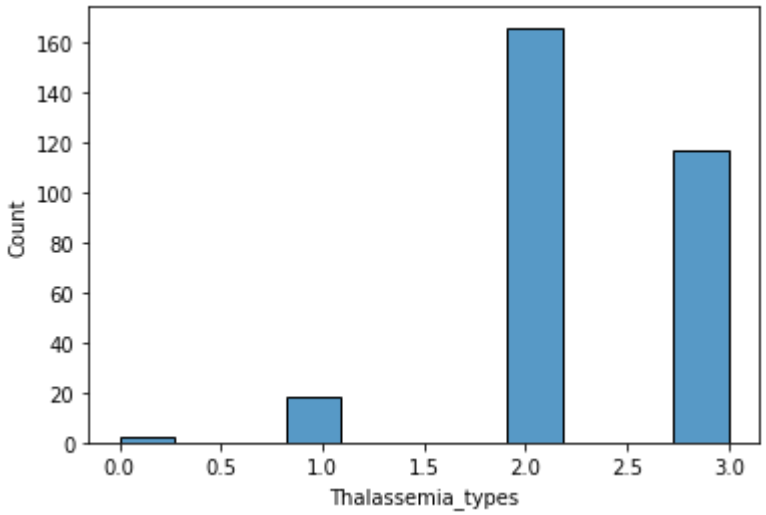
In [48]:
```
sns.relplot(x ='Age', y ='Major_vessels', col = 'Gender', data =data, color = 'cr
```

Out[48]:
```
<seaborn.axisgrid.FacetGrid at 0x7f82fa151cd0>
```



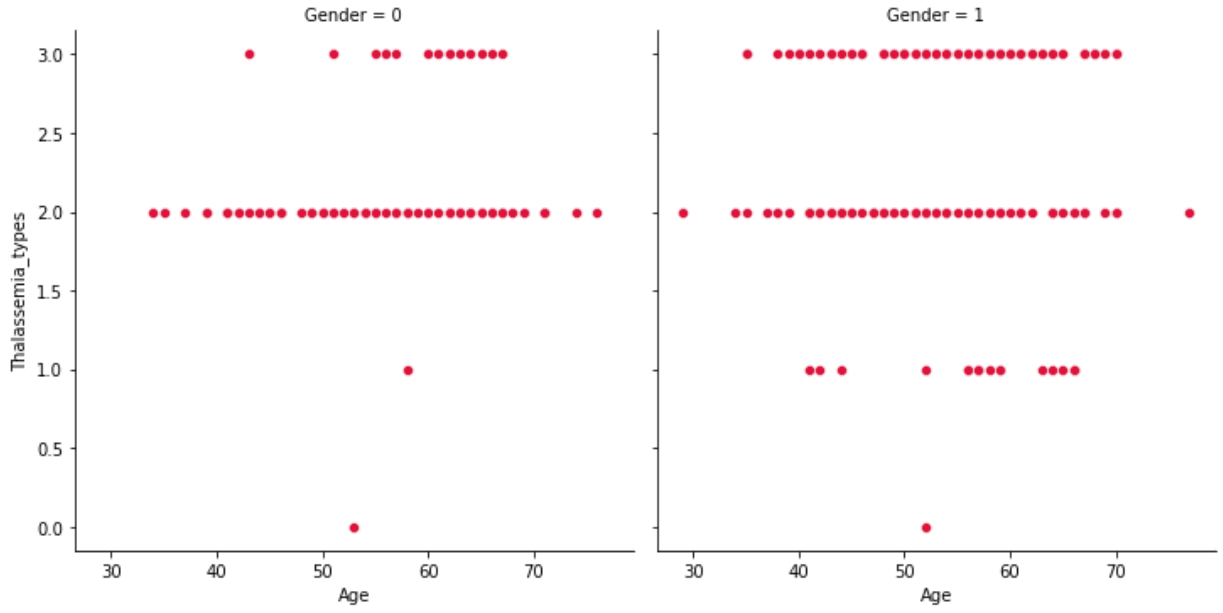
In [49]:
```
#Show histogram for Thalassemia_types
sns.histplot(data['Thalassemia_types'])
```

Out[49]:
```
<AxesSubplot:xlabel='Thalassemia_types', ylabel='Count'>
```



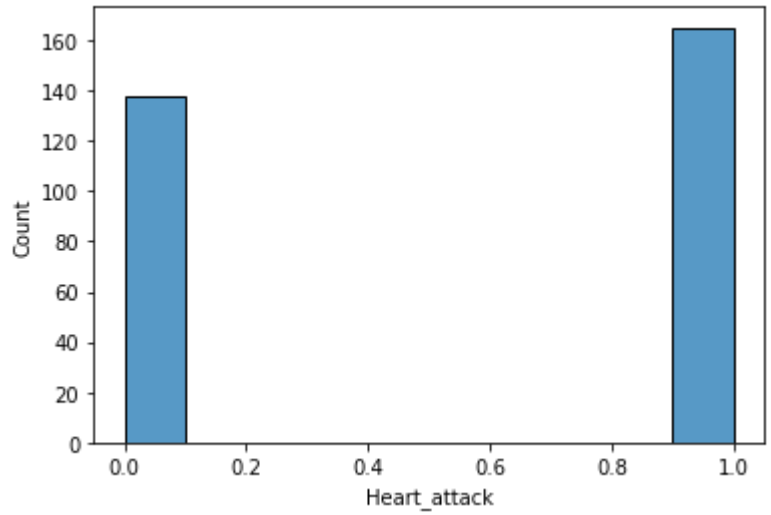
In [50]:
```
sns.relplot(x ='Age', y ='Thalassemia_types', col = 'Gender', data =data, color =
```

Out[50]:
```
<seaborn.axisgrid.FacetGrid at 0x7f82efd92ed0>
```



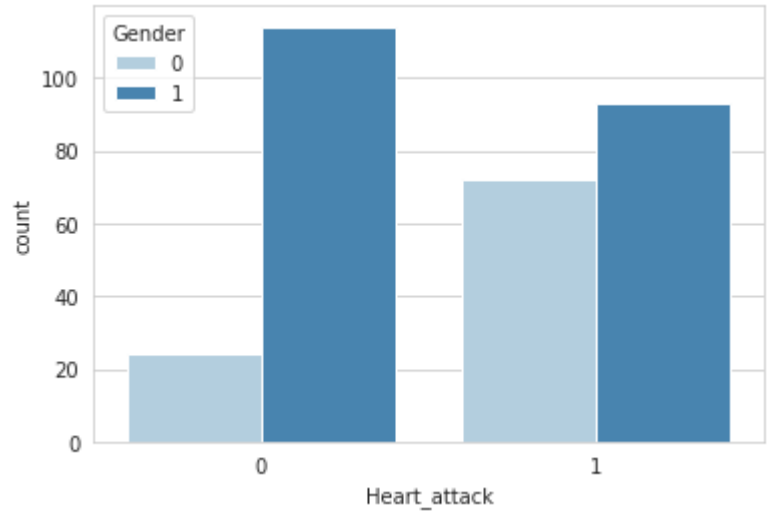
In [51]:
```
#Show histogram for Heart_attack
sns.histplot(data['Heart_attack'])
```

Out[51]:
```
<AxesSubplot:xlabel='Heart_attack', ylabel='Count'>
```



In [52]:
```
sns.set_style('whitegrid') #styling using sns
sns.countplot(x='Heart_attack',hue='Gender',data=data,palette='Blues') #palette t
#Insights from the graph are:
#females are highly suffering from heart disease as compared to males
```

Out[52]:
```
<AxesSubplot:xlabel='Heart_attack', ylabel='count'>
```



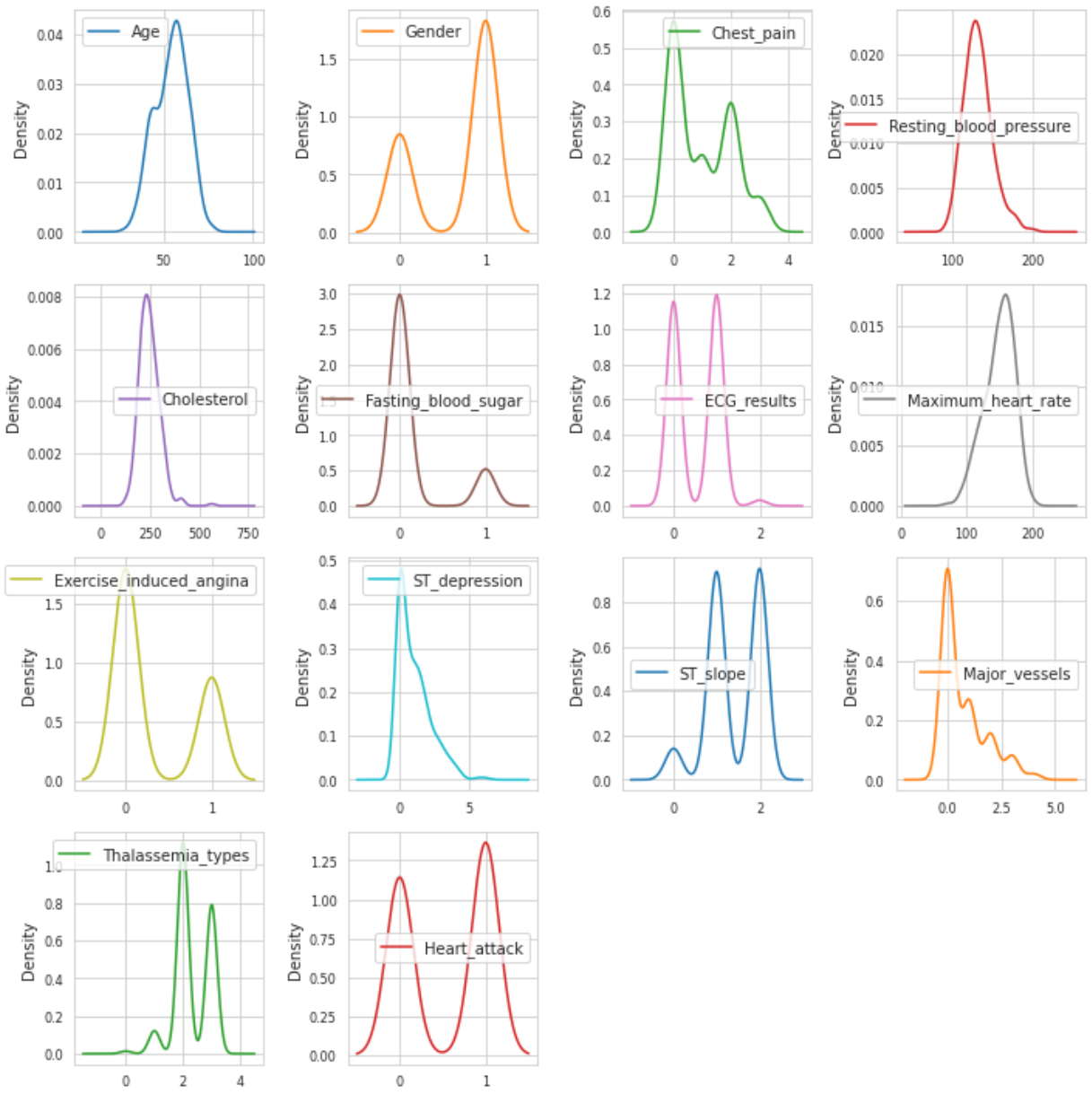
In [53]:
```
data.columns
```

Out[53]:
```
Index(['Age', 'Gender', 'Chest_pain', 'Resting_blood_pressure', 'Cholesterol',
       'Fasting_blood_sugar', 'ECG_results', 'Maximum_heart_rate',
       'Exercise_induced_angina', 'ST_depression', 'ST_slope', 'Major_vessels',
       'Thalassemia_types', 'Heart_attack'],
      dtype='object')
```

In [54]:
```
data.shape
```

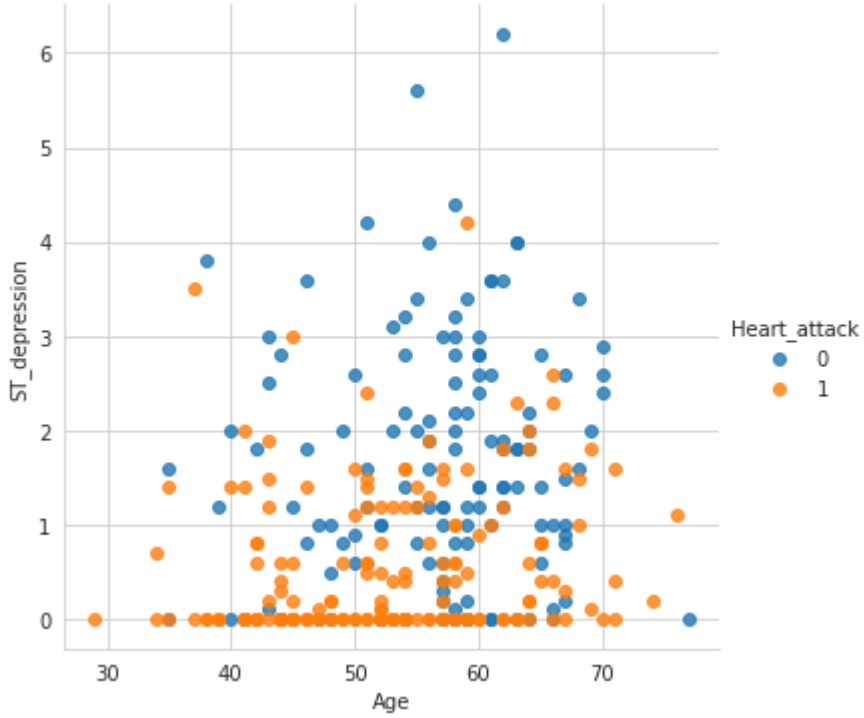
Out[54]:
```
(303, 14)
```

In [55]:
```
data.plot(kind='density' , subplots=True , layout=(4,4) , sharex=False ,
          fontsize=8 , figsize=(10,10))
plt.tight_layout()
```



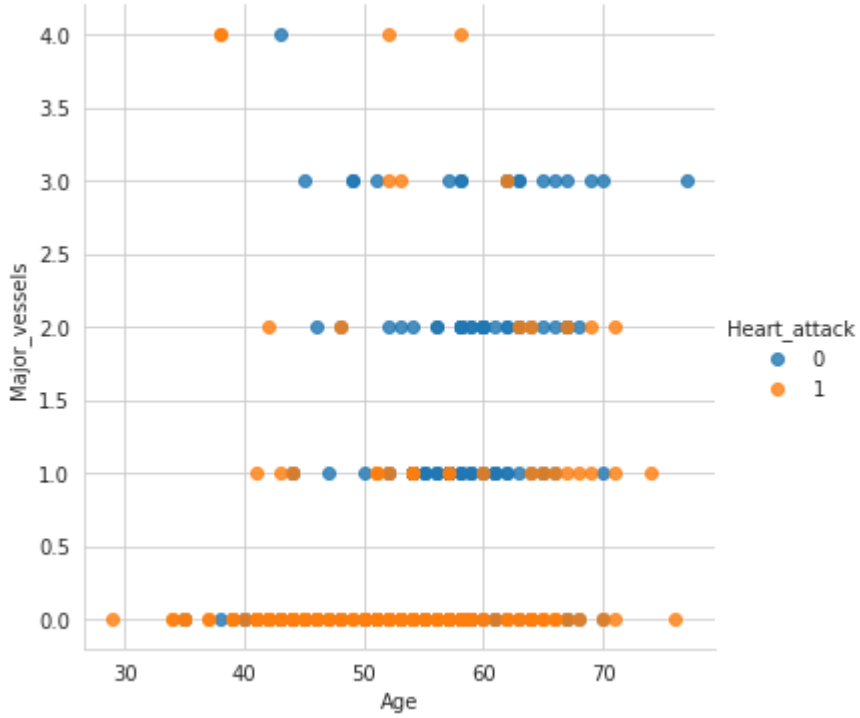
In [56]:
```
sns.lmplot('Age' , 'ST_depression' , data=data , hue='Heart_attack' , fit_reg=Fal
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



In [57]:
```
sns.lmplot('Age' , 'Major_vessels' , data=data , hue='Heart_attack' , fit_reg=Fal
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



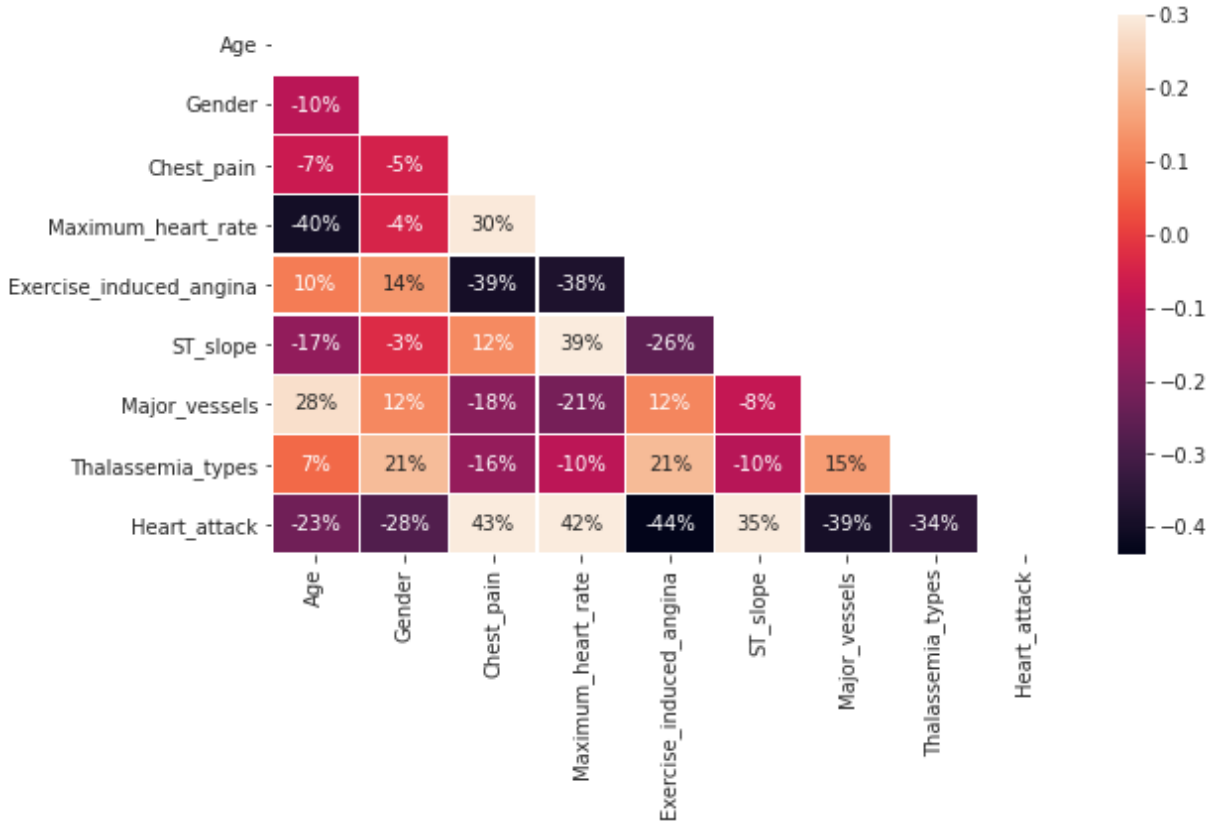
In [58]:
```
#sns.pairplot(data,size=5)
```

In [59]:
```
#data has correlation high with the target(Heart_attack)
data=data[['Age', 'Gender', 'Chest_pain', 'Maximum_heart_rate',
       'Exercise_induced_angina', 'ST_slope', 'Major_vessels',
       'Thalassemia_types', 'Heart_attack']]
```

In [60]:
```
data.columns
```

Out[60]:
```
Index(['Age', 'Gender', 'Chest_pain', 'Maximum_heart_rate',
       'Exercise_induced_angina', 'ST_slope', 'Major_vessels',
       'Thalassemia_types', 'Heart_attack'],
      dtype='object')
```

In [61]:
```
#show correlation
mask = np.zeros_like(data.corr())
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("ticks"):
    f, ax = plt.subplots(figsize=(9, 5))
    ax = sns.heatmap(data.corr(), mask=mask, vmax=.3,annot=True,fmt=".0%",linewic
    #annot is numbers in squares
    #square=True the square is small but square=False the square is large
```



In [62]:
```
#show all statistics
data.describe(include='all')
```

Out[62]:

	Age	Gender	Chest_pain	Maximum_heart_rate	Exercise_induced_angina	ST_slo
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000(
mean	54.366337	0.683168	0.966997	149.646865	0.326733	1.3993
std	9.082101	0.466011	1.032052	22.905161	0.469794	0.6162
min	29.000000	0.000000	0.000000	71.000000	0.000000	0.000(
25%	47.500000	0.000000	0.000000	133.500000	0.000000	1.000(
50%	55.000000	1.000000	1.000000	153.000000	0.000000	1.000(
75%	61.000000	1.000000	2.000000	166.000000	1.000000	2.000(
max	77.000000	1.000000	3.000000	202.000000	1.000000	2.000(

In [63]:
```
#important data
data
```

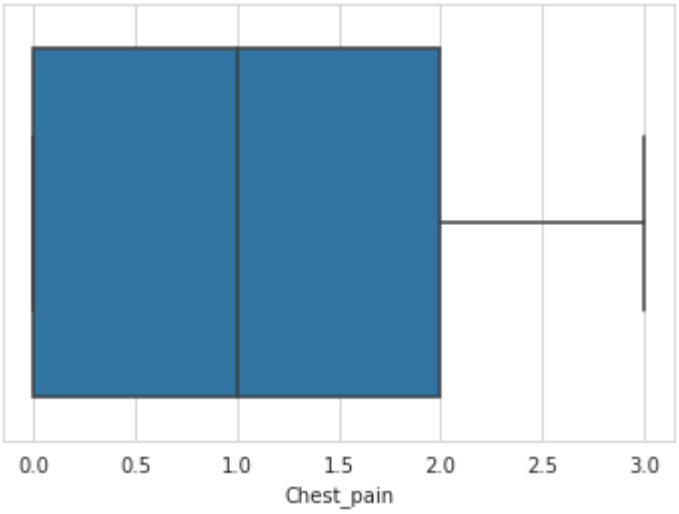
Out[63]:

	Age	Gender	Chest_pain	Maximum_heart_rate	Exercise_induced_angina	ST_slope	Major_ves
0	63	1	3	150	0	0	
1	37	1	2	187	0	0	
2	41	0	1	172	0	2	
3	56	1	1	178	0	2	
4	57	0	0	163	1	2	
...	...	...	...	...	...	...	
298	57	0	0	123	1	1	
299	45	1	3	132	0	1	
300	68	1	0	141	0	1	
301	57	1	0	115	1	1	
302	57	0	1	174	0	1	

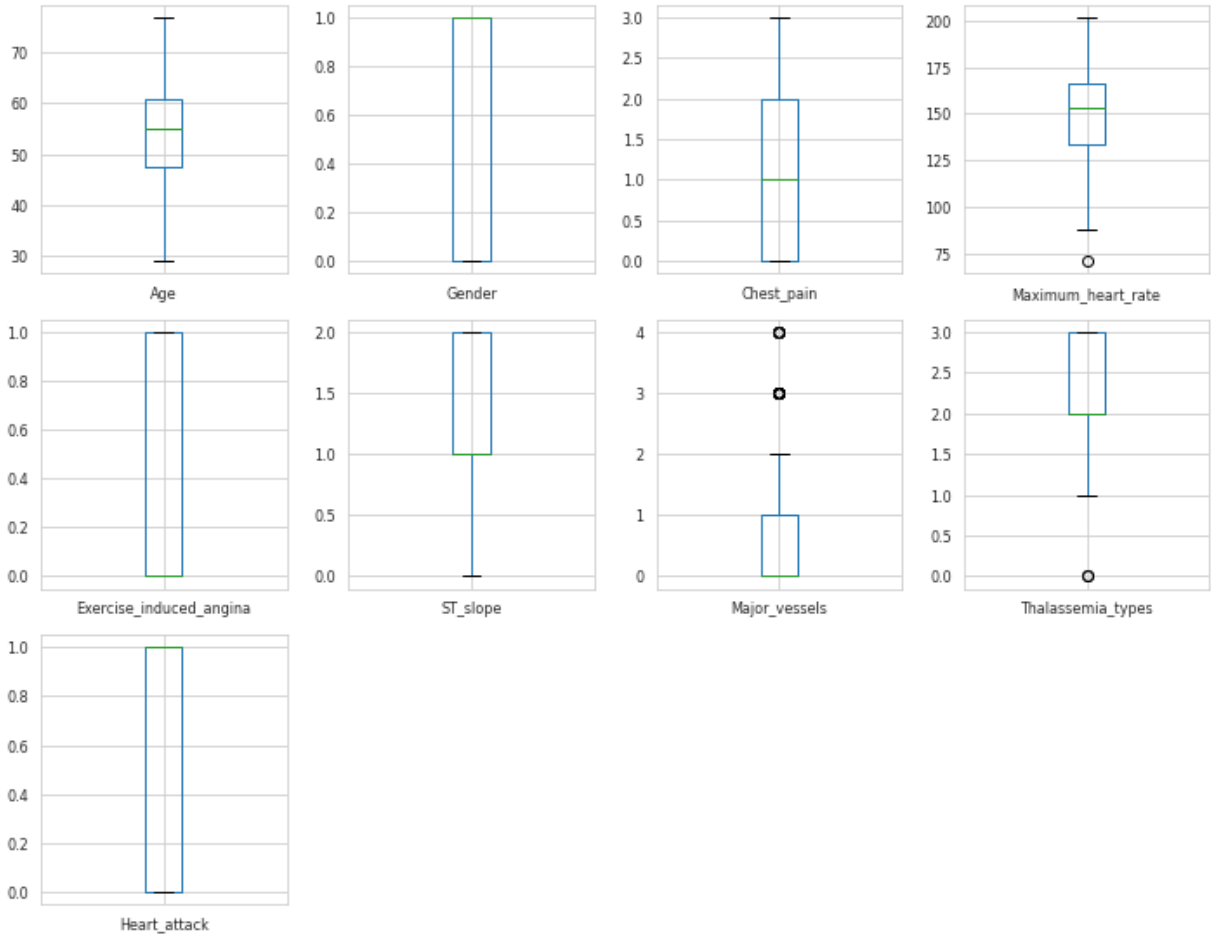
303 rows × 9 columns

In [64]:
```
#show outliers
sns.boxplot(x=data['Chest_pain'])
```

Out[64]: <AxesSubplot:xlabel='Chest_pain'>



In [65]:
```
data.plot(kind='box' , subplots=True , layout=(4,4) , sharex=False ,
          fontsize=8 , figsize=(10,10))
plt.tight_layout()
```

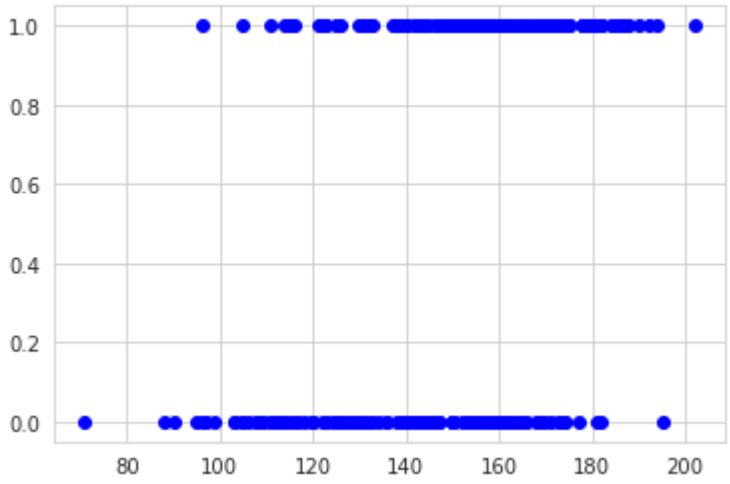


In [73]:
```
data.columns
        '
```

Out[73]:
```
Index(['Age', 'Gender', 'Chest_pain', 'Maximum_heart_rate',
       'Exercise_induced_angina', 'ST_slope', 'Major_vessels',
       'Thalassemia_types', 'Heart_attack'],
      dtype='object')
```

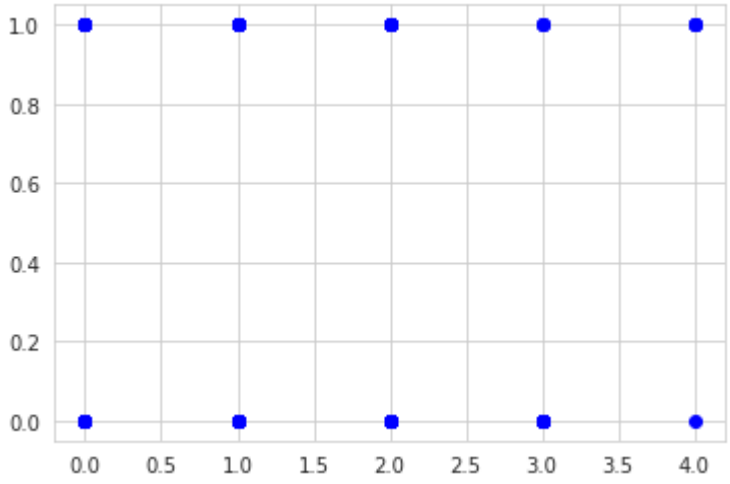
In [83]:
```
plt.scatter(data['Maximum_heart_rate'] , data['Heart_attack'] , color='blue')
```

Out[83]:
```
<matplotlib.collections.PathCollection at 0x7f82ef84b4d0>
```



In [91]:
```
plt.scatter(data['Major_vessels'] , data['Heart_attack'] , color='blue')
```

Out[91]:
```
<matplotlib.collections.PathCollection at 0x7f82ef85d350>
```



In [80]:
```
data[data['Maximum_heart_rate']<75]
```

Out[80]:

	Age	Gender	Chest_pain	Maximum_heart_rate	Exercise_induced_angina	ST_slope	Major_ves
272	67	1	0	71	0	1	

In [95]:
```
data[data['Major_vessels']>=3.5]
```

Out[95]:

	Age	Gender	Chest_pain	Maximum_heart_rate	Exercise_induced_angina	ST_slope	Major_ves
92	52	1	2	169	0	2	
158	58	1	1	144	0	1	
163	38	1	2	173	0	2	
164	38	1	2	173	0	2	
251	43	1	0	143	1	1	

In [85]:
```
#print outliers
print('outliers' , data[(data['Maximum_heart_rate']<75)]['Maximum_heart_rate'].co
```

```
outliers 1
```

In [104]:
```
#print outliers
print('outliers' , data[(data['Major_vessels']>=3)]['Major_vessels'].count())
```

```
outliers 25
```

In [132]:
```
#print outliers
print('outliers' , data[(data['Major_vessels']>=3.5)]['Major_vessels'].count())
```

```
outliers 5
```

In [133]:
```
#print outliers
print('outliers' , data[(data['Thalassemia_types']<=0)]['Thalassemia_types'].cour
```

```
outliers 2
```

In [134]:
```
data.shape
```

Out[134]:
```
(303, 9)
```

In [135]:
```
#remove row outliers
clean_data= data[(data['Major_vessels']<=3.5) ]
```

In [136]:
```
clean_data.shape
```

Out[136]:
```
(298, 9)
```

In [137]:
```
clean_data
```

Out[137]:

	Age	Gender	Chest_pain	Maximum_heart_rate	Exercise_induced_angina	ST_slope	Major_ves
0	63	1	3	150	0	0	
1	37	1	2	187	0	0	
2	41	0	1	172	0	2	
3	56	1	1	178	0	2	
4	57	0	0	163	1	2	
...	...	...	...	...	...	...	
298	57	0	0	123	1	1	
299	45	1	3	132	0	1	
300	68	1	0	141	0	1	
301	57	1	0	115	1	1	
302	57	0	1	174	0	1	

298 rows × 9 columns

In [138]:
```
data.columns
```

Out[138]:
```
Index(['Age', 'Gender', 'Chest_pain', 'Maximum_heart_rate',
       'Exercise_induced_angina', 'ST_slope', 'Major_vessels',
       'Thalassemia_types', 'Heart_attack'],
      dtype='object')
```

In [139]:
```
clean_data.columns
```

Out[139]:
```
Index(['Age', 'Gender', 'Chest_pain', 'Maximum_heart_rate',
       'Exercise_induced_angina', 'ST_slope', 'Major_vessels',
       'Thalassemia_types', 'Heart_attack'],
      dtype='object')
```

In [140]:
```
#Spliting Data
import numpy as np
from sklearn.model_selection import train_test_split
```

In [142]:
```
X = clean_data.drop(['Heart_attack'] , axis=1).values
```

In [143]:
```
Y = clean_data['Heart_attack'].values
```

In [144]:
```
#split data 70% for train and 30% for test
x_train , x_test , y_train , y_test = train_test_split(X,Y , test_size=0.03 ,ranc
```

In [145]:
```
x_train.shape
```

Out[145]:
```
(289, 8)
```

In [153]:
```
#Linear Regression
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(x_train , y_train)
y_pred = reg.predict(x_test)
```

In [156]:
```
#Training score
reg.score(x_train,y_train)
```

Out[156]:
```
0.5089475649292499
```

In [157]:
```
#Training score
reg.score(x_test,y_test)
```

Out[157]:
```
0.7818928765356215
```

This is OverFitting as Train score Higher than test score by 28%

In [158]:
```
#accuracy for Model
from sklearn.metrics import r2_score
r2 = r2_score(y_test , y_pred)
r2
```

Out[158]:
```
0.7818928765356215
```

In [67]:
```
#to download and save data
#data.to_csv("data_name.csv",header=True,index=True,encoding="utf-8")
```