In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

a.Perform preliminary data inspection and report the findings on the structure of the data, missing values, duplicates, etc.

In [2]:

```
df=pd.read_excel("1645792390_cep1_dataset.xlsx")
```

In [3]:

```
df.head()
```

Out[3]:

	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 [4]:

```
df.shape
```

Out[4]:

```
(303, 14)
```

In [5]:

```
df.isna().sum()
```

Out[5]:

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

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   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 [7]:

```
#there are no missing values
```

a.Get a preliminary statistical summary of the data and explore the measures of central tendencies and spread of the data

In [8]:

```
df.describe()
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	3C
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	

In [9]:

```
# The average age among the list considered is 54 years. Min and max ages being 29 years and 77 years.
```

b.Identify the data variables which are categorical and describe and explore these variables using the appropriate tools, such as count plot

In [10]:

```
# age, sex and cholesterol levels can be classified as categorical variables
```

In [11]:

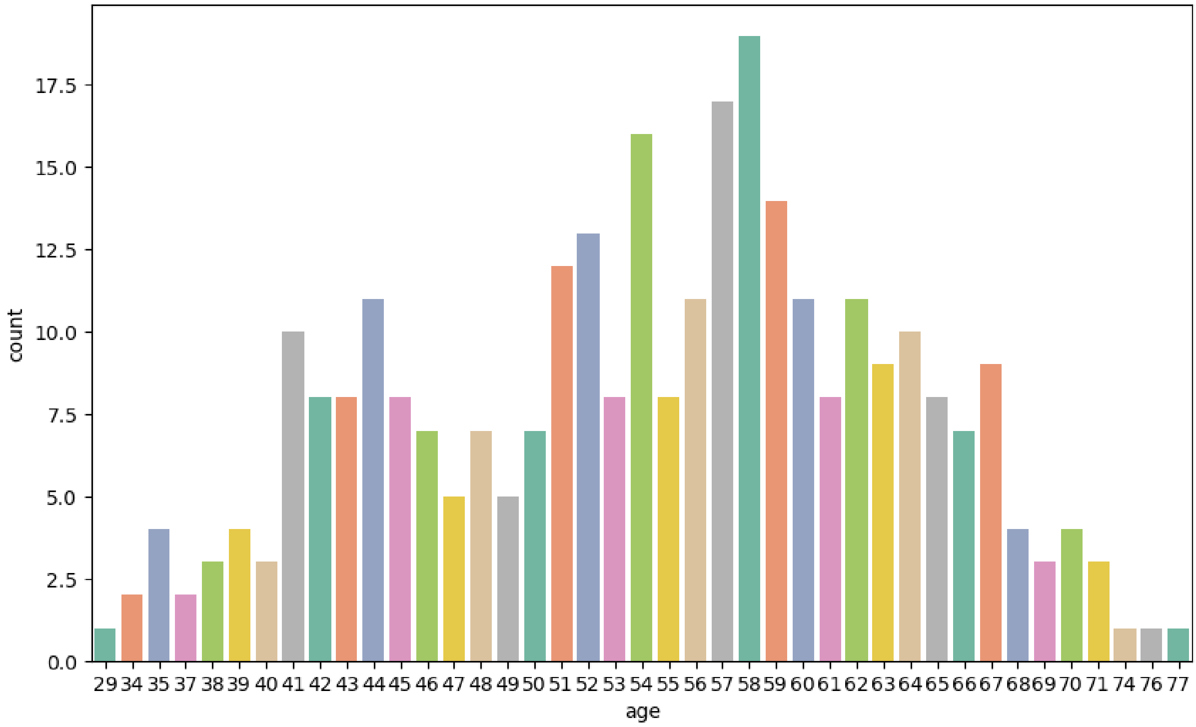
```
import seaborn as sns
```

In [12]:

```
plt.figure(figsize=(10,6))
sns.countplot(x=df['age'],palette='Set2')
```

Out[12]:

```
<AxesSubplot:xlabel='age', ylabel='count'>
```

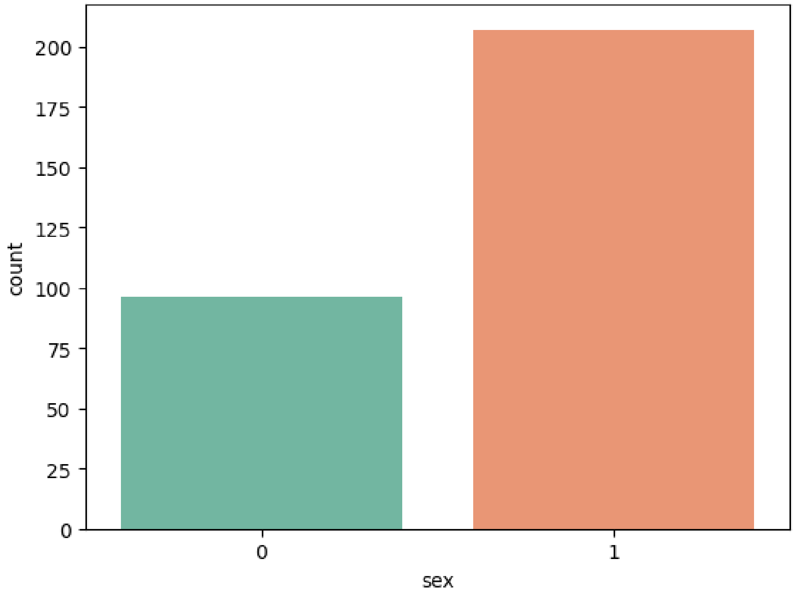


In [13]:

```
sns.countplot(x=df['sex'],palette='Set2')
```

Out[13]:

```
<AxesSubplot:xlabel='sex', ylabel='count'>
```

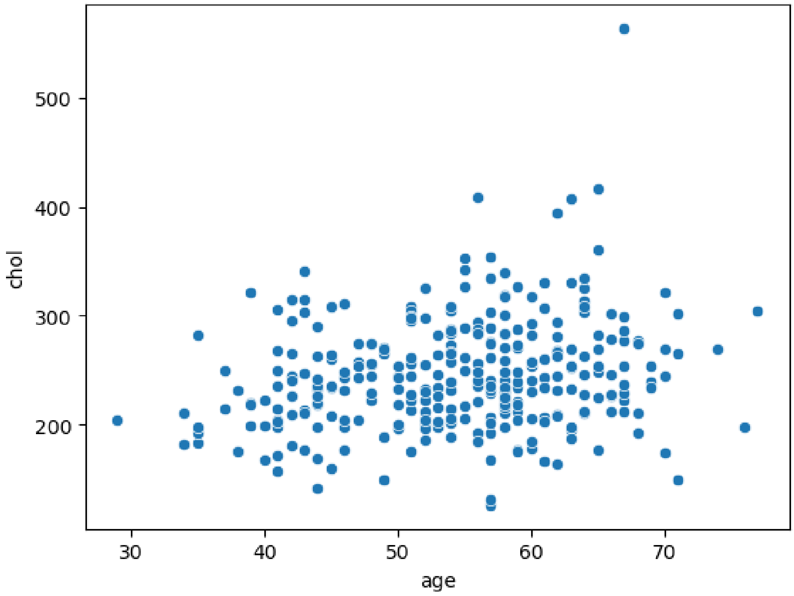


In [14]:

```
sns.scatterplot(x='age',y='chol',data=df)
```

Out[14]:

```
<AxesSubplot:xlabel='age', ylabel='chol'>
```



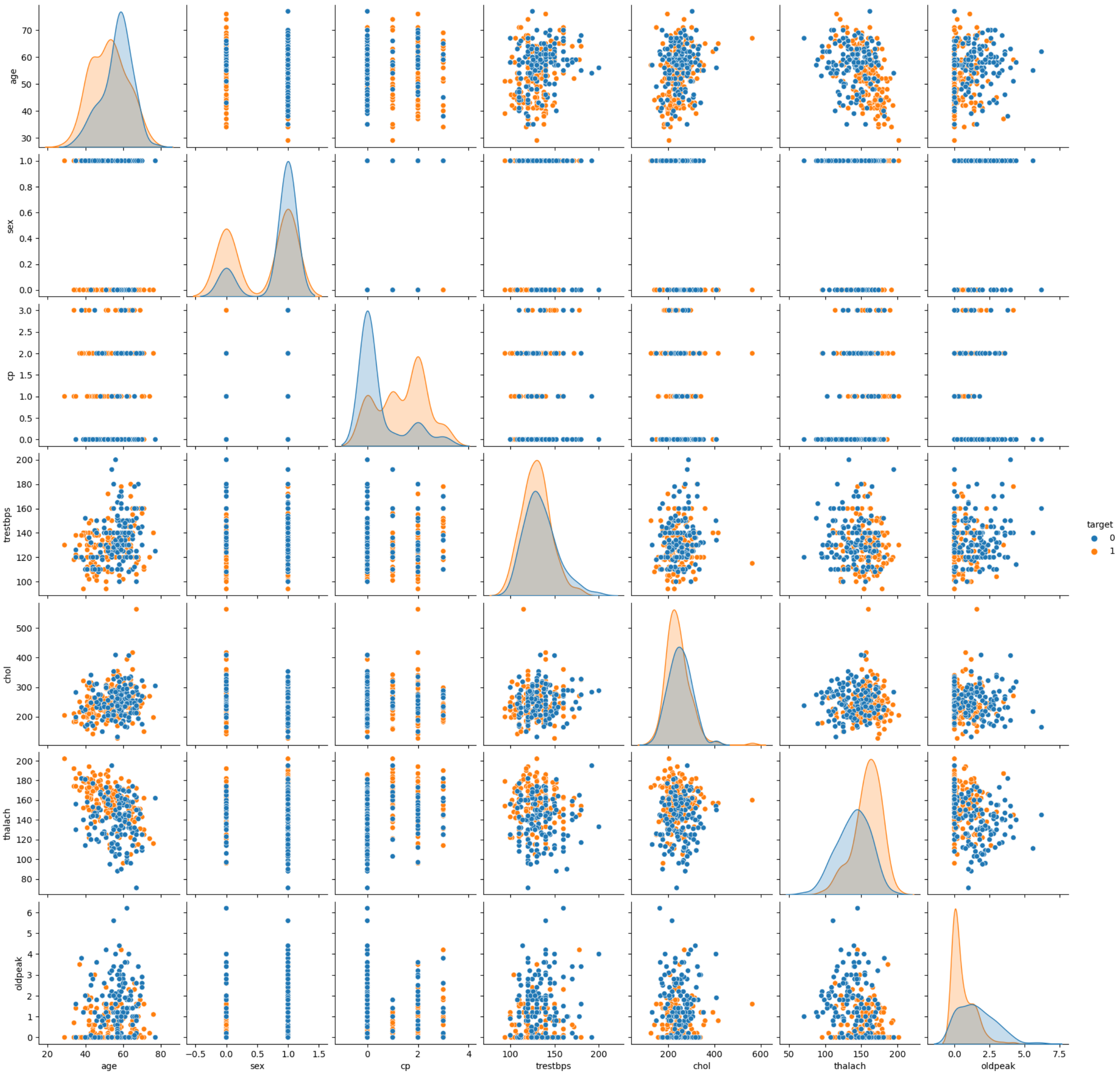
In [ ]:

In [15]:

```
columns=['age','sex','cp','trestbps','chol','thalach','oldpeak','target']
sns.pairplot(df[columns],hue='target')
```

Out[15]:

```
<seaborn.axisgrid.PairGrid at 0x1f00b5d1e80>
```

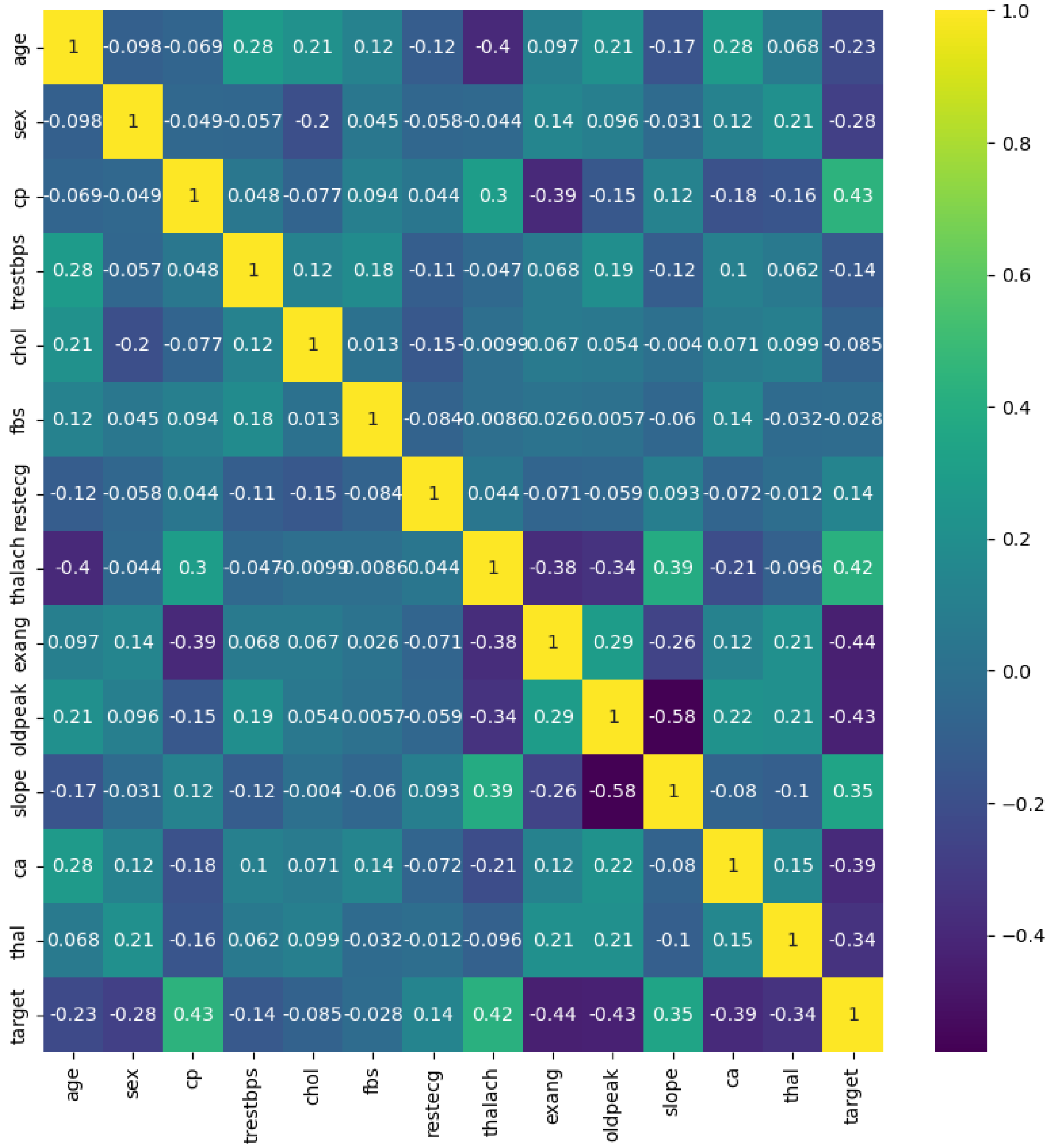


In [16]:

```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),cmap='viridis',annot=True)
```

Out[16]:

```
<AxesSubplot:>
```



In [17]:

```
np.abs(df.corr()['target']).sort_values().tail(6)
```

Out[17]:

```
ca          0.391724
thalach     0.421741
oldpeak     0.430696
cp          0.433798
exang       0.436757
target      1.000000
Name: target, dtype: float64
```

In [18]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [19]:

```
print(df.target)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

In [20]:

```
df.data=df.iloc[:,:-1]
```

In [21]:

```
df.data
```

Out[21]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
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 × 13 columns

In [22]:

```
df.target=df.iloc[:,-1]
```

In [23]:

```
df.target
```

Out[23]:

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

In [24]:

```
x_train,x_test,y_train,y_test=train_test_split(df.data,df.target,stratify=df.target,random_state=42,train_size=0.7)
```

In [25]:

```
log_reg=LogisticRegression()
log_reg.fit(x_train,y_train)
y_pred=log_reg.predict(x_test)
```

In [26]:

```
y_pred
```

Out[26]:

```
array([1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 0, 0], dtype=int64)
```

In [27]:

```
df_predicted=pd.DataFrame()
df_predicted['Actual']=y_test
df_predicted['Predicted']=y_pred
df_predicted.head()
```

Out[27]:

	Actual	Predicted
123	1	1
283	0	1
206	0	0
95	1	0
271	0	1

In [28]:

```
mislabel=np.sum(y_test!=y_pred)
```

In [29]:

```
mislabel
```

Out[29]:

```
23
```

In [30]:

```
len(y_test)
```

Out[30]:

```
91
```

In [31]:

```
68/91
```

Out[31]:

```
0.7472527472527473
```

In [32]:

```
from sklearn.metrics import accuracy_score,f1_score
accuracy_score(y_test,y_pred)*100
```

Out[32]:

```
74.72527472527473
```

In [33]:

```
from sklearn.metrics import confusion_matrix
```

In [34]:

```
cm=(confusion_matrix(y_test,y_pred))
```

In [35]:

```
cm
```

Out[35]:

```
array([[28, 13],
       [10, 40]], dtype=int64)
```

In [36]:

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

```
              precision    recall  f1-score   support

           0       0.74      0.68      0.71        41
           1       0.75      0.80      0.78        50

    accuracy                           0.75        91
   macro avg       0.75      0.74      0.74        91
weighted avg       0.75      0.75      0.75        91
```

In [37]:

```
f1_score(y_test,y_pred)*100
```

Out[37]:

```
77.66990291262137
```

In [ ]: