In [204]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans
```

In [205]:

```
df= pd.read_csv(r"C:\Users\dtdee\OneDrive\Desktop\Letsupgrade_Python\Machine_Learning\KMeans\1681627696073_blood_pressure.csv"
```

In [206]:

```
df
```

Out[206]:

	age	sex	BP	cholestrol
0	70	1	130	322
1	67	0	115	564
2	57	1	124	261
3	64	1	128	263
4	74	0	120	269
...	...	...	...	...
265	52	1	172	199
266	44	1	120	263
267	56	0	140	294
268	57	1	140	192
269	67	1	160	286

270 rows × 4 columns

In [207]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         270 non-null    int64
 1   sex         270 non-null    int64
 2   BP          270 non-null    int64
 3   cholestrol  270 non-null    int64
dtypes: int64(4)
memory usage: 8.6 KB
```

In [208]:

```
df.duplicated().sum()
```

Out[208]:

```
0
```

In [209]:

```
df.isnull().sum()
```

Out[209]:

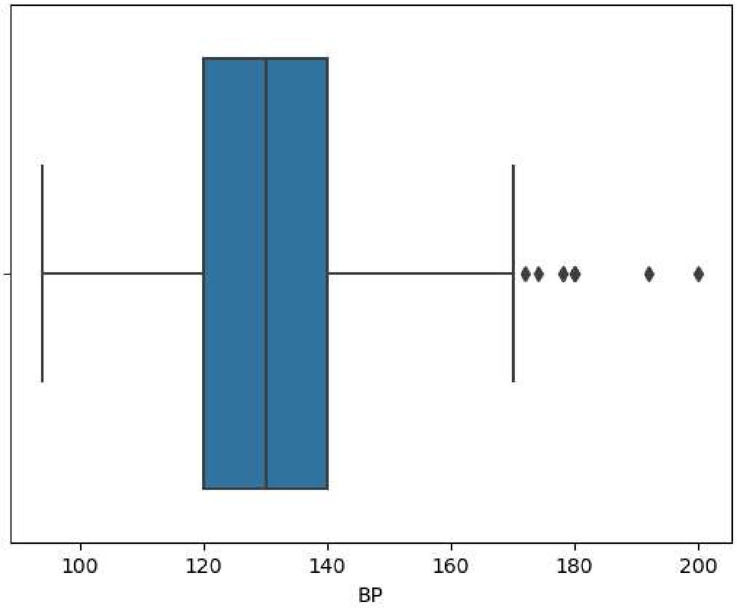
```
age           0
sex           0
BP            0
cholestrol    0
dtype: int64
```

In [210]:

```
sns.boxplot(x='BP',data=df)
```

Out[210]:

```
<AxesSubplot:xlabel='BP'>
```

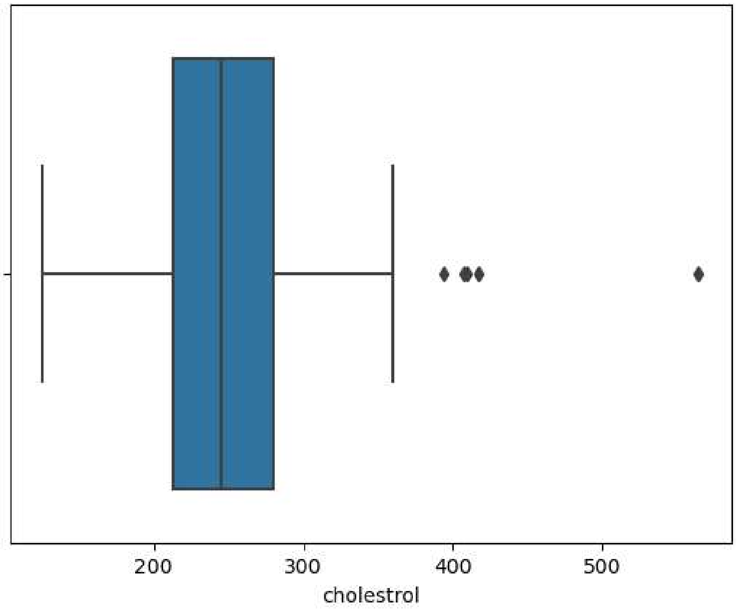


In [211]:

```
sns.boxplot(x='cholestrol',data=df)
```

Out[211]:

```
<AxesSubplot:xlabel='cholestrol'>
```



In [212]:

```
# This is method WCSS to find best value of K

loss=[]

for i in range(1,10):
    kmeans=KMeans(n_clusters=i,max_iter=200,random_state=20,init='k-means++')
    kmeans.fit(df)
    loss.append(kmeans.inertia_)
    print('loss of the model is=',loss)
```

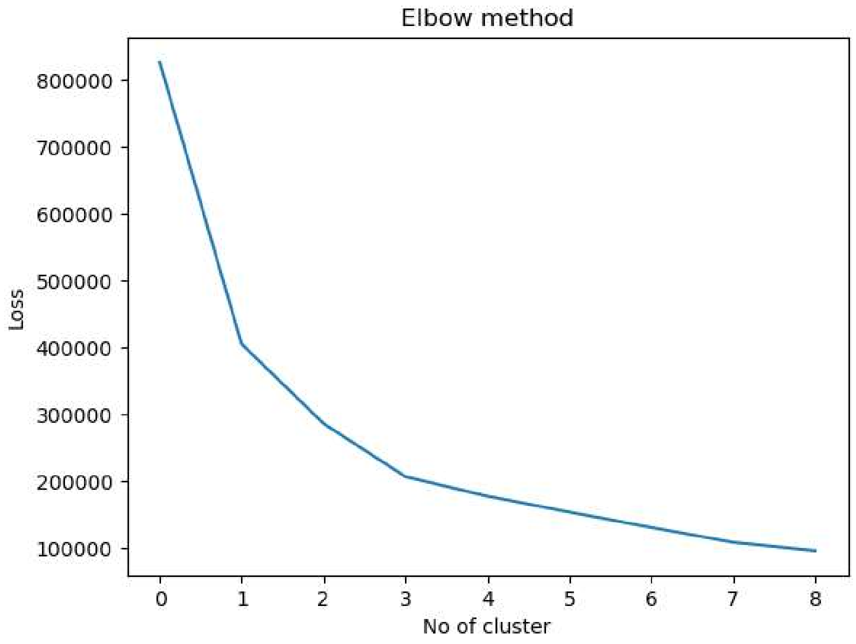
```
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have
a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=2.
  warnings.warn(

loss of the model is= [826824.8851851847]
loss of the model is= [826824.8851851847, 405017.95670995687]
loss of the model is= [826824.8851851847, 405017.95670995687, 286021.24050290015]
loss of the model is= [826824.8851851847, 405017.95670995687, 286021.24050290015, 206477.32211121658]
loss of the model is= [826824.8851851847, 405017.95670995687, 286021.24050290015, 206477.32211121658, 177218.317
45140717]
loss of the model is= [826824.8851851847, 405017.95670995687, 286021.24050290015, 206477.32211121658, 177218.317
45140717, 153469.81030715944]
loss of the model is= [826824.8851851847, 405017.95670995687, 286021.24050290015, 206477.32211121658, 177218.317
45140717, 153469.81030715944, 130377.87635239164]
loss of the model is= [826824.8851851847, 405017.95670995687, 286021.24050290015, 206477.32211121658, 177218.317
45140717, 153469.81030715944, 130377.87635239164, 108236.40138673228]
loss of the model is= [826824.8851851847, 405017.95670995687, 286021.24050290015, 206477.32211121658, 177218.317
45140717, 153469.81030715944, 130377.87635239164, 108236.40138673228, 95712.65418482812]
```

In [213]:

```
plt.plot(loss)
plt.xlabel('No of cluster')
plt.ylabel('Loss')
plt.title('Elbow method')
```

Out[213]:

```
Text(0.5, 1.0, 'Elbow method')
```



In [214]:

```
# Another method to find the value of K by hyperparameter tuning


kmeans=KMeans(n_clusters=1,random_state=20,init='k-means++')
kmeans.fit(df)
kmeans.inertia_
```

```
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have
a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
```

Out[214]:

```
826824.8851851847
```

In [215]:

```
kmeans=KMeans(n_clusters=2,random_state=20,init='k-means++')
kmeans.fit(df)
kmeans.inertia_
```

Out[215]:

```
405017.95670995687
```

In [216]:

```
kmeans=KMeans(n_clusters=3,random_state=20,init='k-means++')
kmeans.fit(df)
kmeans.inertia_
```

Out[216]:

```
286021.24050290015
```

In [217]:

```
kmeans=KMeans(n_clusters=4,random_state=20,init='k-means++')
kmeans.fit(df)
kmeans.inertia_
```

Out[217]:

```
206477.32211121658
```

In [218]:

```
kmeans=KMeans(n_clusters=5,random_state=20,init='k-means++')
kmeans.fit(df)
kmeans.inertia_
```

Out[218]:

```
177218.31745140717
```

In [219]:

```
# So we can take the value of K=2
```

In [220]:

```
model=KMeans(n_clusters=2,init='k-means++',random_state=20)
model.fit(df)
model.inertia_
```

Out[220]:

```
405017.95670995687
```

In [ ]:

```

```

In [221]:

```
model.cluster_centers_
```

Out[221]:

```
array([[ 56.67619048,   0.57142857, 135.82857143, 298.95238095],
       [ 53.00606061,   0.74545455, 128.49090909, 218.29090909]])
```

In [222]:

```
df
```

Out[222]:

	age	sex	BP	cholestrol
0	70	1	130	322
1	67	0	115	564
2	57	1	124	261
3	64	1	128	263
4	74	0	120	269
...	...	...	...	...
265	52	1	172	199
266	44	1	120	263
267	56	0	140	294
268	57	1	140	192
269	67	1	160	286

270 rows × 4 columns

In [ ]:

In [223]:

```
model.labels_
```

Out[223]:

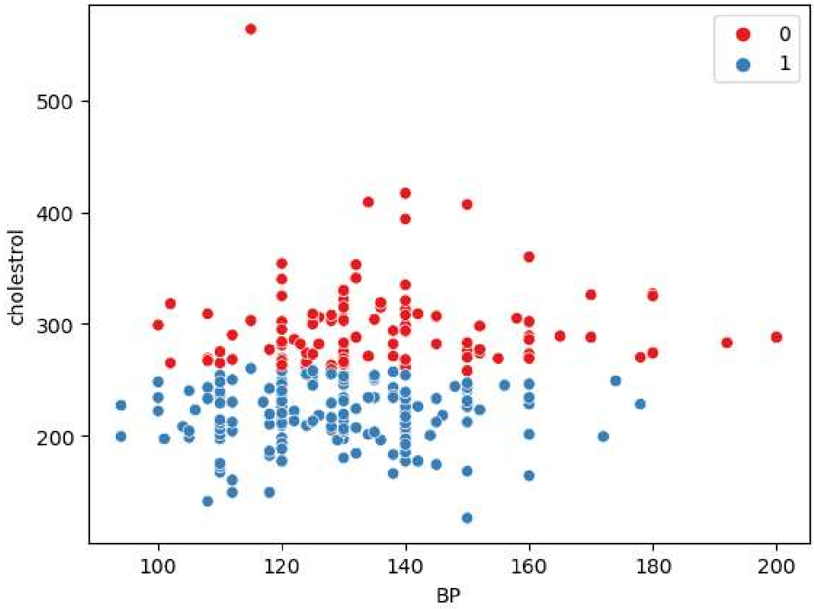
```
array([0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
       0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 1, 0, 0, 1, 0])
```

In [224]:

```
sns.scatterplot(x='BP' ,y='cholestrol' ,data=df ,hue=model.labels_ ,palette= 'Set1')
```

Out[224]:

```
<AxesSubplot:xlabel='BP', ylabel='cholestrol'>
```



In [225]:

```
model3=KMeans(n_clusters=3,init='k-means++',random_state=20)
model3.fit(df)
model3.inertia_
```

Out[225]:

```
286021.24050290015
```

In [226]:

```
print(model3.labels_)

print()
print()

print(np.unique(model3.labels_))
```

```
[1 1 0 0 0 2 0 0 1 1 0 0 0 0 1 2 1 2 2 2 0 0 1 2 1 0 0 1 0 0 0 0 2 1 2 0 2
 0 2 2 2 2 0 2 2 2 2 2 0 0 1 2 1 2 0 2 2 2 0 0 2 1 2 0 0 0 0 1 0 0 0 1 0 2
 2 1 2 2 2 0 2 2 0 2 2 0 2 0 1 0 2 1 0 2 2 0 1 0 1 0 2 0 0 1 0 0 2 0 2 2 1
 0 1 1 1 0 0 1 0 0 2 0 0 1 0 2 0 2 2 0 0 2 0 0 0 2 0 0 2 2 2 0 0 2 1 0 0 2
 2 0 0 2 0 1 0 2 2 1 2 0 0 0 0 0 2 1 0 2 0 0 0 0 1 2 2 1 1 0 0 2 0 1 2 0 0
 1 2 0 1 0 0 2 1 0 0 0 0 2 0 1 0 1 0 1 1 1 1 0 1 0 1 0 2 2 2 2 0 2 0 2 0 0
 2 1 2 2 0 1 0 2 1 2 0 0 0 2 0 2 0 1 0 1 0 0 1 0 0 0 2 0 2 2 1 2 2 1 0 1 1
 0 1 2 0 0 0 2 0 1 2 1]


[0 1 2]
```

In [227]:

```
C=model3.cluster_centers_
C
```

Out[227]:

```
array([[ 55.56910569,   0.72357724, 131.62601626, 252.54471545],
       [ 56.45762712,   0.50847458, 138.44067797, 320.74576271],
       [ 51.48863636,   0.72727273, 126.19318182, 197.96590909]])
```

In [228]:

```
C[:,3]
```

Out[228]:

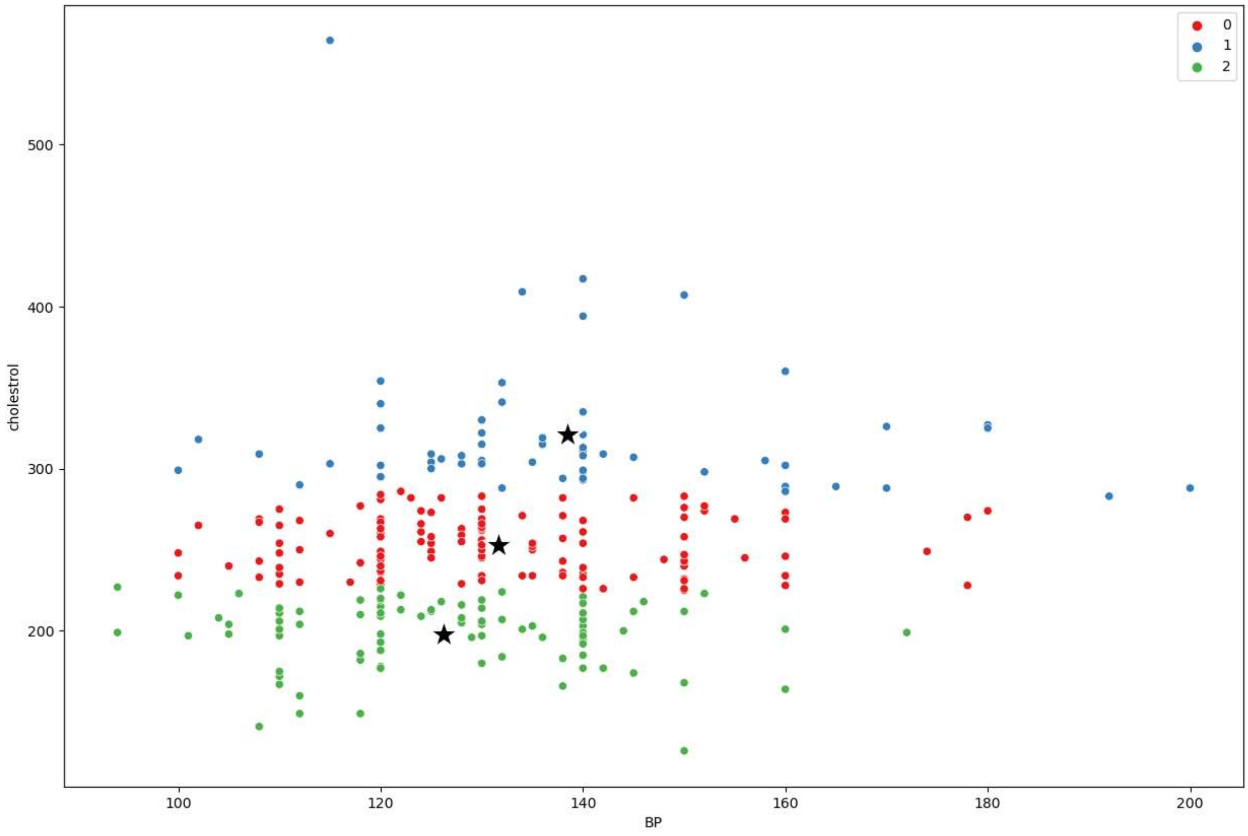
```
array([252.54471545, 320.74576271, 197.96590909])
```

In [229]:

```
plt.subplots(figsize=(15,10))
sns.scatterplot(x= 'BP',y='cholestrol' ,data=df ,hue=model3.labels_,palette='Set1')
sns.scatterplot(x= C[:,2],y= C[:,3],marker='*',s=500,color='Black')
```

Out[229]:

```
<AxesSubplot:xlabel='BP', ylabel='cholestrol'>
```



In [230]:

```
# If we take the value of K=4 and try to get out put from the model

model4=KMeans(n_clusters=4,init='k-means++',random_state=20)
model4.fit(df)
model4.inertia_
model4.labels_
centeriod=model4.cluster_centers_
```

In [231]:

```
centeriod[:,2]

centeriod[:,3]
```

Out[231]:

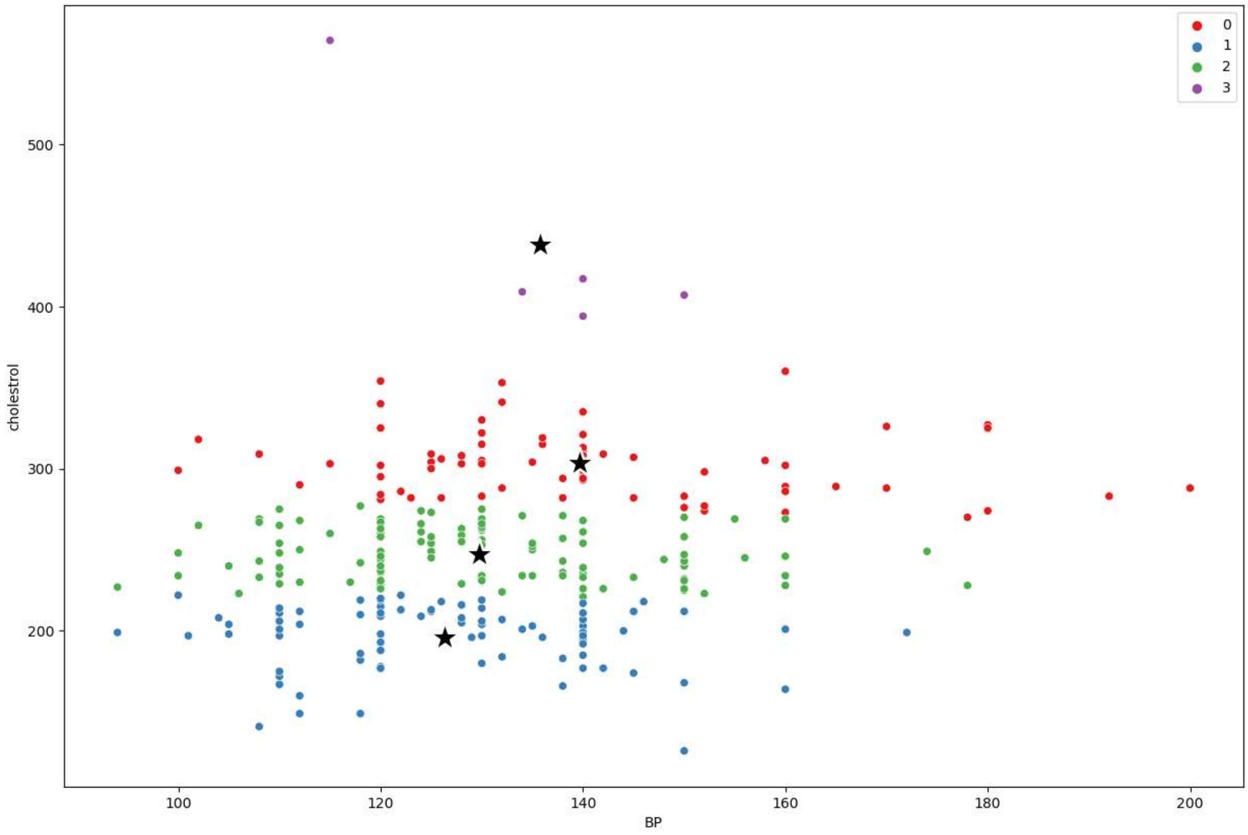
```
array([303.2173913 , 196.06097561, 247.52631579, 438.2       ])
```

In [232]:

```
plt.subplots(figsize=(15,10))
sns.scatterplot(x= 'BP',y='cholestrol' ,data=df ,hue=model4.labels_,palette='Set1')
sns.scatterplot(x=centeriod[:,2],y= centeriod[:,3],marker='*',s=500,color='Black')
```

Out[232]:

```
<AxesSubplot:xlabel='BP', ylabel='cholestrol'>
```



CONCLUSION

We can take the best value of k=2,3,4 out of which model is predicting the best cluster in 2 and 3 clusters

We can Apply DBSCAN Method to check the clusters Knowing the fact that DBSCAN Algo runs best into those datasets where there is Nosie and outliers present in data

DBSCAN METHOD

In [233]:

```
df1=df[['BP','cholestrol']]
df1
```

Out[233]:

	BP	cholestrol
0	130	322
1	115	564
2	124	261
3	128	263
4	120	269
...	...	...
265	172	199
266	120	263
267	140	294
268	140	192
269	160	286

270 rows × 2 columns

In [243]:

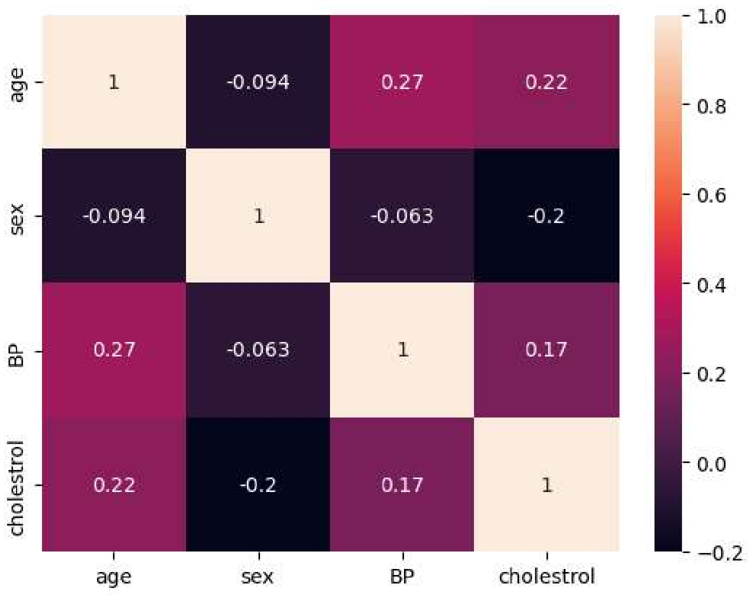
```
X=df1.iloc[:,0].values
Y=df1.iloc[:,1].values
```

In [244]:

```
sns.heatmap(df.corr(),annot=True)
```

Out[244]:

```
<AxesSubplot:>
```

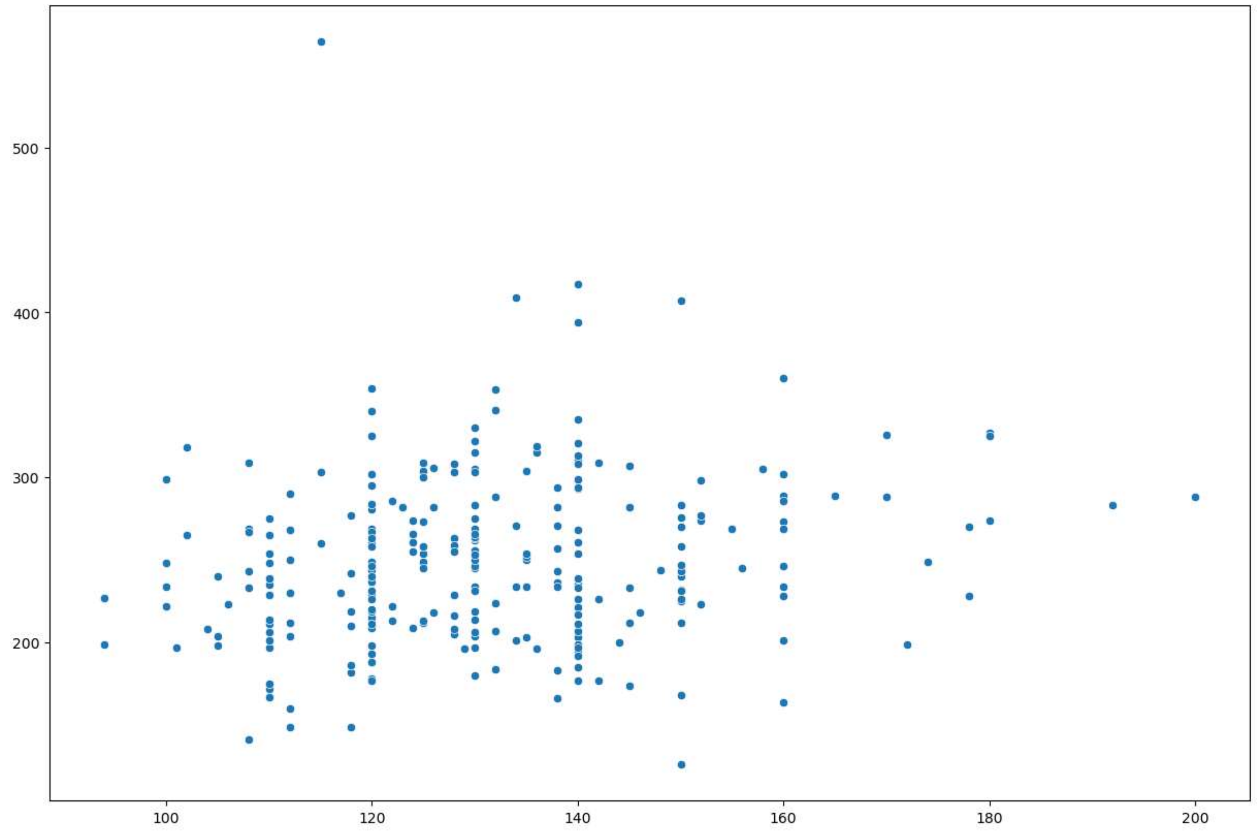


In [245]:

```
plt.subplots(figsize=(15,10))
sns.scatterplot(x=X ,y= Y ,data=df1.values,palette='Set1')
```

Out[245]:

```
<AxesSubplot:>
```



In [237]:

```
scan1=DBSCAN(eps=20,min_samples=50)
scan1.fit(df1.values)
print()
print('Lables are=',scan1.labels_)
print()
print('No of unique classes generated=',np.unique(scan1.labels_))
```

```
Lables are= [-1 -1  0  0  0 -1  0  0 -1 -1  0  0  0  0 -1 -1 -1  0  0  0  0  0 -1  0
 -1  0  0 -1  0  0  0 -1  0 -1  0  0  0 -1 -1  0 -1  0  0 -1  0  0 -1  0
 -1  0 -1  0 -1  0  0 -1 -1 -1 -1  0 -1 -1  0  0  0 -1  0 -1  0  0  0 -1
  0  0  0 -1  0 -1  0  0  0  0 -1  0  0  0  0 -1 -1  0  0 -1  0  0  0  0
 -1  0 -1  0 -1  0  0 -1  0  0 -1  0 -1 -1 -1  0 -1 -1 -1  0  0 -1  0  0
  0 -1 -1 -1  0 -1  0  0  0  0  0  0  0  0  0 -1 -1  0  0  0  0  0  0  0
 -1  0  0 -1  0  0  0  0  0 -1  0  0  0 -1  0 -1  0  0  0  0  0 -1  0  0
  0 -1 -1  0 -1  0 -1 -1 -1  0  0  0  0 -1 -1  0  0 -1 -1  0 -1 -1  0 -1
 -1  0  0  0  0  0  0 -1  0 -1  0 -1 -1 -1 -1  0 -1  0 -1  0  0  0  0  0
  0 -1  0  0  0  0 -1 -1 -1  0  0 -1 -1 -1 -1  0  0  0  0 -1  0 -1  0 -1
 -1 -1  0  0 -1  0  0 -1 -1 -1  0  0 -1 -1 -1 -1  0 -1 -1  0 -1  0  0  0
  0 -1  0 -1  0 -1]

No of unique classes generated= [-1  0]
```

In [238]:

```
scan2=DBSCAN(eps=15,min_samples=20)
scan2.fit(df1.values)
print()
print('Lables are=',scan2.labels_)
print()
print('No of unique classes generated=',np.unique(scan2.labels_))
```

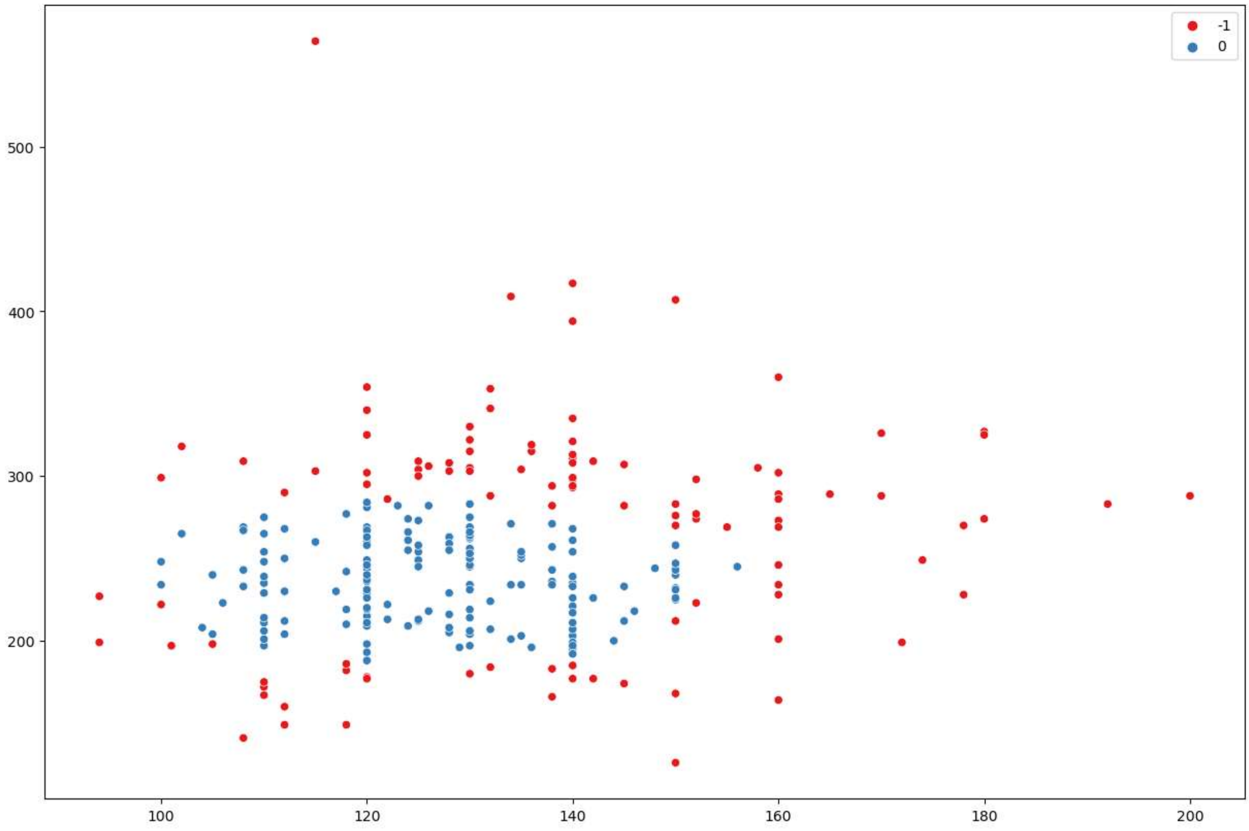
```
Lables are= [ 1 -1  0  0  0 -1  0  0  1 -1  0  0  0  0  1 -1  1  0  0  0  0  0  1  0
  0  0  0  1  0  0  0  0  0 -1  0  0  0 -1 -1  0  0  0  0  0  0  0 -1  0
  0  0  1  0 -1  0  0  0 -1 -1 -1  0 -1  1  0  0  0  0  0  1  0  0  0 -1
  0  0  0  1  0 -1  0  0  0  0  0  0  0  0  0 -1  1  0  0  1  0  0  0  0
  1  0  1  0 -1  0  0 -1  0  0  0  0 -1 -1 -1  0 -1  1  1  0  0 -1  0  0
  0  0  0 -1  0 -1  0  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0  0  0
 -1  0  0 -1  0  0  0  0  0  1  0  0  0  1  0 -1  0  0  0  0  0 -1  0  0
  0  0  0  0 -1  0 -1  1  1  0  0  0  0 -1 -1  0  0  1  0  0 -1 -1  0 -1
 -1  0  0  0  0  0  0  1  0  1  0  1 -1 -1 -1  0  1  0 -1  0  0  0  0  0
  0 -1  0  0  0  0 -1  1  0  0  0 -1 -1  0  1  0  0  0  0 -1  0 -1  0  1
 -1  1  0  0  1  0  0 -1  0  0  0  0  0 -1  0 -1  0 -1 -1  0 -1  0  0  0
  0 -1  0  1  0 -1]

No of unique classes generated= [-1  0  1]
```

In [246]:

```
plt.subplots(figsize=(15,10))
sns.scatterplot(x=X ,y= Y ,data=df1.values,hue= scan1.labels_  ,palette='Set1')
```

Out[246]:

```
<AxesSubplot:>
```

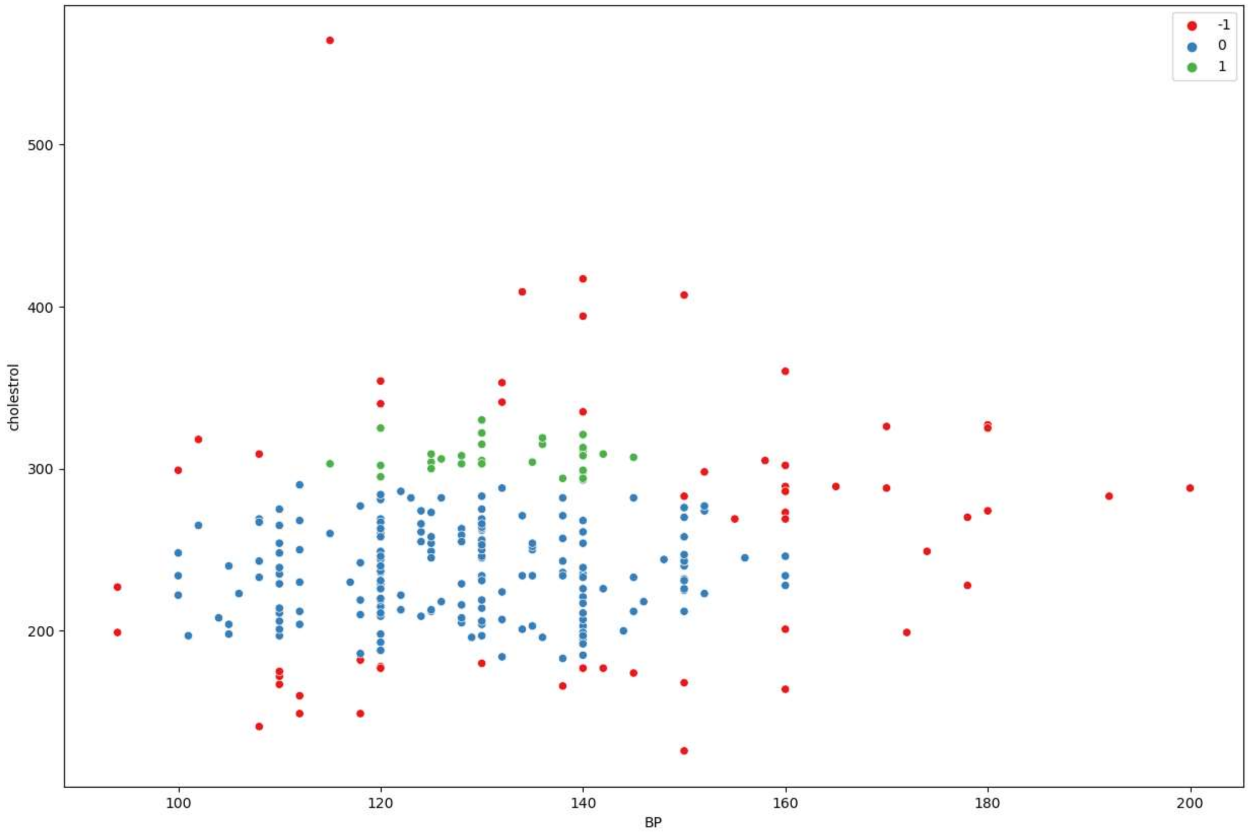


In [240]:

```
plt.subplots(figsize=(15,10))
sns.scatterplot(x='BP'  ,y= 'cholestrol' ,data=df1,hue= scan2.labels_  ,palette='Set1')
```

Out[240]:

```
<AxesSubplot:xlabel='BP', ylabel='cholestrol'>
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: