In [ ]:

In [80]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier

from imblearn.over_sampling import SMOTE



# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the input directory

# Load data and print a few rows
FILE_NAME = '/Users/wsyed2/Documents/JNTU/Python/Assignments/heart_disease_uci.csv'

     # Loading the Data
df = pd.read_csv(FILE_NAME)
# Any results you write to the current directory are saved as output.
df = pd.read_csv(FILE_NAME)
#Chcking first few rows of Data from Dataset
df.head(10)
```

Out[80]:

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
0	1	63	Male	Cleveland	typical angina	145.0	233.0	True	lv hypertrophy	150.0	False	2.3	downsloping	0.0	fixed defect	0
1	2	67	Male	Cleveland	asymptomatic	160.0	286.0	False	lv hypertrophy	108.0	True	1.5	flat	3.0	normal	2
2	3	67	Male	Cleveland	asymptomatic	120.0	229.0	False	lv hypertrophy	129.0	True	2.6	flat	2.0	reversable defect	1
3	4	37	Male	Cleveland	non-anginal	130.0	250.0	False	normal	187.0	False	3.5	downsloping	0.0	normal	0
4	5	41	Female	Cleveland	atypical angina	130.0	204.0	False	lv hypertrophy	172.0	False	1.4	upsloping	0.0	normal	0
5	6	56	Male	Cleveland	atypical angina	120.0	236.0	False	normal	178.0	False	0.8	upsloping	0.0	normal	0
6	7	62	Female	Cleveland	asymptomatic	140.0	268.0	False	lv hypertrophy	160.0	False	3.6	downsloping	2.0	normal	3
7	8	57	Female	Cleveland	asymptomatic	120.0	354.0	False	normal	163.0	True	0.6	upsloping	0.0	normal	0
8	9	63	Male	Cleveland	asymptomatic	130.0	254.0	False	lv hypertrophy	147.0	False	1.4	flat	1.0	reversable defect	2
9	10	53	Male	Cleveland	asymptomatic	140.0	203.0	True	lv hypertrophy	155.0	True	3.1	downsloping	0.0	reversable defect	1

In [81]:
```
#Remove Id and dataset
df.drop(['id','dataset'], axis=1, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       920 non-null    int64
 1   sex       920 non-null    object
 2   cp        920 non-null    object
 3   trestbps  861 non-null    float64
 4   chol      890 non-null    float64
 5   fbs       830 non-null    object
 6   restecg   918 non-null    object
 7   thalch    865 non-null    float64
 8   exang     865 non-null    object
 9   oldpeak   858 non-null    float64
 10  slope     611 non-null    object
 11  ca        309 non-null    float64
 12  thal      434 non-null    object
 13  num       920 non-null    int64
dtypes: float64(5), int64(2), object(7)
memory usage: 100.8+ KB
```

In [36]:
```
#Checking the datatypes of the columns
df.describe()
```

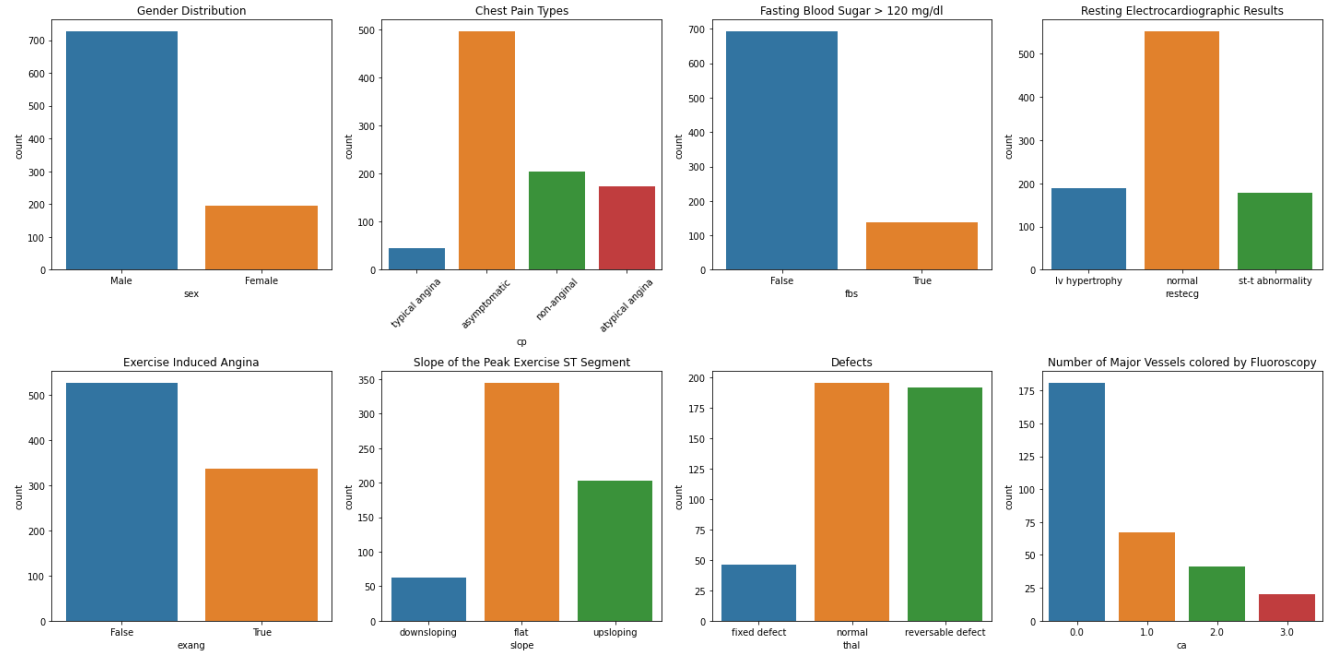
Out[36]:

	age	trestbps	chol	thalch	oldpeak	ca	num
count	920.000000	861.000000	890.000000	865.000000	858.000000	309.000000	920.000000
mean	53.510870	132.132404	199.130337	137.545665	0.878788	0.676375	0.995652
std	9.424685	19.066070	110.780810	25.926276	1.091226	0.935653	1.142693
min	28.000000	0.000000	0.000000	60.000000	-2.600000	0.000000	0.000000
25%	47.000000	120.000000	175.000000	120.000000	0.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	140.000000	0.500000	0.000000	1.000000
75%	60.000000	140.000000	268.000000	157.000000	1.500000	1.000000	2.000000
max	77.000000	200.000000	603.000000	202.000000	6.200000	3.000000	4.000000

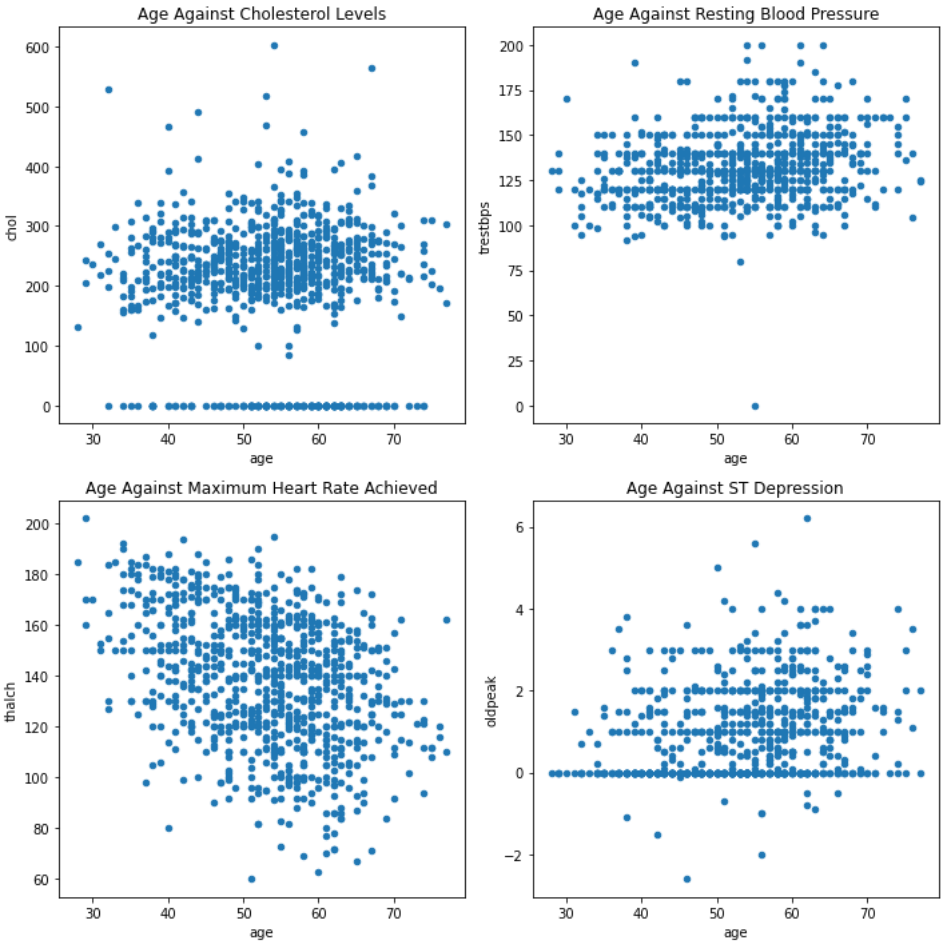
In [38]:
```
# Now Separate numeric and categoriecal variables for visualization purpose
CATEGORICAL_COLS = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'thal', 'ca']
NUMERICAL_COLS = ['age', 'trestbps', 'chol', 'thalch', 'oldpeak']

df_cat = df[CATEGORICAL_COLS]
df_num = df[NUMERICAL_COLS]

df_cat.nunique()#Summary Stastics of columns
```

Out[38]:
```
sex         2
cp          4
fbs         2
restecg     3
exang       2
slope       3
thal        3
ca          4
dtype: int64
```

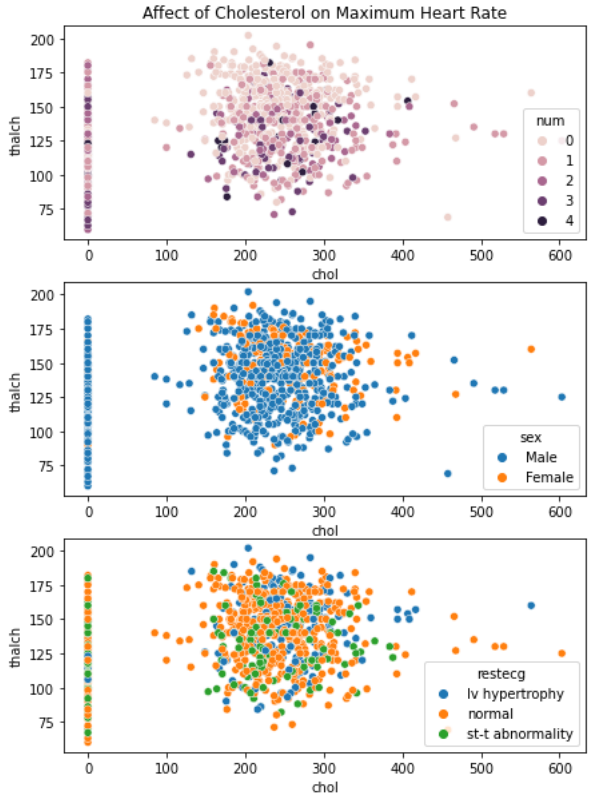
In [40]:
```
#Analyze key relationships in categorical data
fig, axes = plt.subplots(2, 4, figsize=(20,10))

sns.countplot(x='sex', data=heart_cat, ax=axes[0,0])
axes[0,0].set_title('Gender Distribution')

sns.countplot(x='cp', data=heart_cat, ax=axes[0,1])
axes[0,1].tick_params(axis='x', rotation=45)
axes[0,1].set_title('Chest Pain Types')

sns.countplot(x='fbs', data=heart_cat, ax=axes[0,2])
axes[0,2].set_title('Fasting Blood Sugar > 120 mg/dl')

sns.countplot(x='restecg', data=heart_cat, ax=axes[0,3])
axes[0,3].set_title('Resting Electrocardiographic Results')

sns.countplot(x='exang', data=heart_cat, ax=axes[1,0])
axes[1,0].set_title('Exercise Induced Angina')

sns.countplot(x='slope', data=heart_cat, ax=axes[1,1])
axes[1,1].set_title('Slope of the Peak Exercise ST Segment')

sns.countplot(x='thal', data=heart_cat, ax=axes[1,2])
axes[1,2].set_title('Defects')

sns.countplot(x='ca', data=heart_cat, ax=axes[1,3])
axes[1,3].set_title('Number of Major Vessels colored by Fluoroscopy')
plt.tight_layout()
plt.show()
```



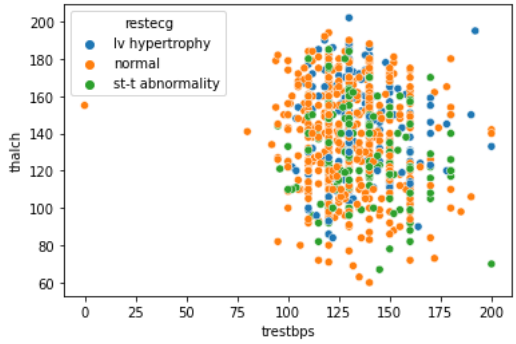
In [41]:
```
# use scatterplots to visualize key relationships in numerical data

fig, axes = plt.subplots(2, 2, figsize=(10,10))

heart_num.plot('age', 'chol', kind='scatter', ax=axes[0,0])
axes[0,0].set_title('Age Against Cholesterol Levels')

heart_num.plot('age', 'trestbps', kind='scatter', ax=axes[0,1])
axes[0,1].set_title('Age Against Resting Blood Pressure')

heart_num.plot('age', 'thalch', kind='scatter', ax=axes[1,0])
axes[1,0].set_title('Age Against Maximum Heart Rate Achieved')

heart_num.plot('age', 'oldpeak', kind='scatter', ax=axes[1,1])
axes[1,1].set_title('Age Against ST Depression')

plt.tight_layout()
plt.show()
```



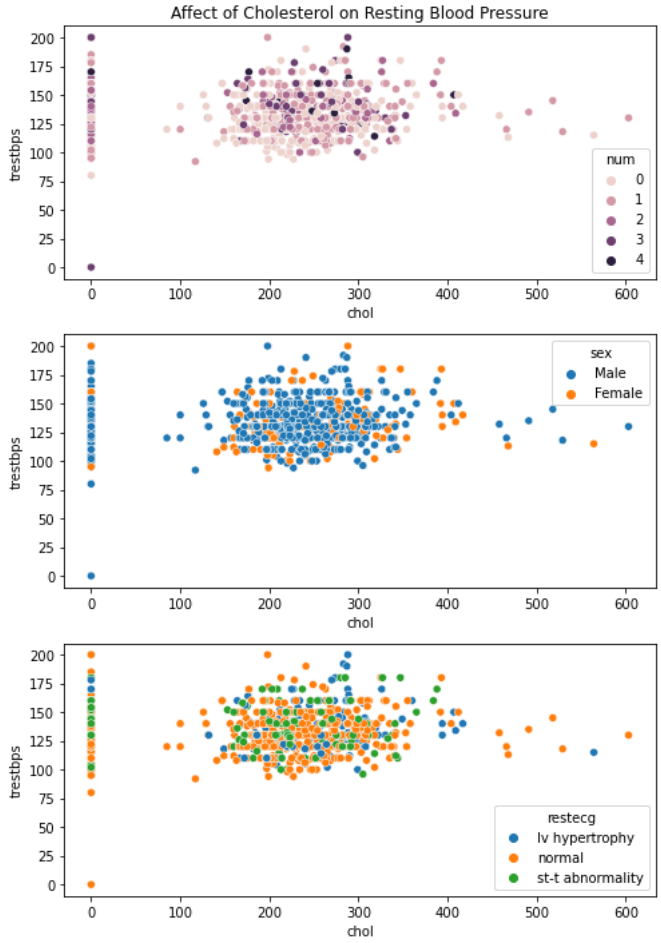
In [43]:
```
fig, axes = plt.subplots(3, figsize=(7,10))

sns.scatterplot(x='chol', y='thalch', hue='num', data=df, ax=axes[0])
axes[0].set_title('Affect of Cholesterol on Maximum Heart Rate')

sns.scatterplot(x='chol', y='thalch', hue='sex', data=df, ax=axes[1])

sns.scatterplot(x='chol', y='thalch', hue='restecg', data=df, ax=axes[2])
plt.show()
```



In [44]:
```
sns.scatterplot(x='trestbps', y='thalch', hue='restecg', data=df)
plt.show()
```



In [45]:
```
fig, axes = plt.subplots(3, figsize=(7,10))

axes[0].set_title('Affect of Cholesterol on Resting Blood Pressure')
sns.scatterplot(x='chol', y='trestbps', hue='num', data=df, ax=axes[0])
sns.scatterplot(x='chol', y='trestbps', hue='sex', data=df, ax=axes[1])
sns.scatterplot(x='chol', y='trestbps', hue='restecg', data=df, ax=axes[2])

plt.tight_layout()
plt.show()
```



In [47]:
```
df.groupby('num').mean()
```

Out[47]:

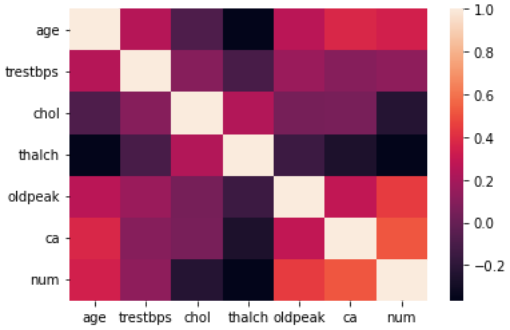
num	age	trestbps	chol	thalch	oldpeak	ca
0	50.547445	129.913043	227.905612	148.800512	0.418205	0.278788
1	53.528302	132.861111	195.255814	131.035714	1.001200	0.741379
2	57.577982	133.613861	143.859813	128.666667	1.353465	1.222222
3	59.214953	136.152174	159.716981	120.500000	1.581319	1.459459
4	59.214286	138.720000	192.148148	127.846154	2.307692	1.692308

In [50]:
```
print('Average Cholesterol Level Based on Target Variable and Chest Pain Type')
print(pd.crosstab(index=df.num, columns=df.cp, values=df.chol, aggfunc=np.mean))
print('\n')

print('Average Cholesterol Level Based on Target Variable and Patient Gender')
print(pd.crosstab(index=df.num, columns=df.sex, values=df.chol, aggfunc=np.mean))
print('\n')

print('Average Cholesterol Level Based on Target Variable and Cardiographic Results')
print(pd.crosstab(index=df.num, columns=df.restecg, values=df.chol, aggfunc=np.mean))
```

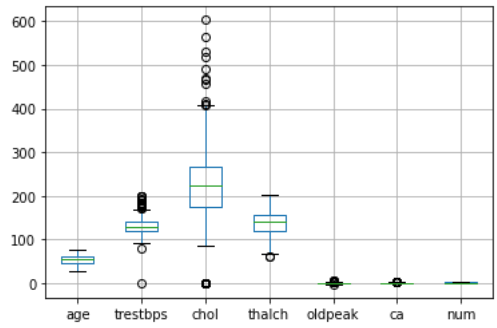
```
Average Cholesterol Level Based on Target Variable and Chest Pain Type
cp   asymptomatic  atypical angina  non-anginal  typical angina
num
0    227.843137    233.957143       222.209677   222.730769
1    193.273684    250.157895       170.756757   215.250000
2    152.321839    123.000000       118.642857   58.500000
3    157.219512    200.000000       152.888889   228.666667
4    196.478261    NaN              146.000000   231.000000


Average Cholesterol Level Based on Target Variable and Patient Gender
sex      Female        Male
num
0    248.102190  217.054902
1    221.366667  191.820175
2    216.400000  136.381443
3    216.250000  155.102041
4    316.000000  182.240000


Average Cholesterol Level Based on Target Variable and Cardiographic Results
restecg  lv hypertrophy      normal  st-t abnormality
num
0        251.768293      227.797619  194.637931
1        216.318182      194.243902  181.395833
2        231.666667      116.629630  132.187500
3        241.230769      130.408163  137.677419
4        238.166667      137.875000  175.285714
```

In [51]:
```
# Display correlation matrix and heatmap
corr = df.corr()
print(corr)

sns.heatmap(corr)
plt.show()
```

```
              age  trestbps      chol    thalch   oldpeak        ca       num
age       1.000000  0.244253 -0.086234 -0.365778  0.258243  0.370416  0.339596
trestbps  0.244253  1.000000  0.092853 -0.104899  0.161908  0.093705  0.122291
chol     -0.086234  0.092853  1.000000  0.236121  0.047734  0.051606 -0.231547
thalch   -0.365778 -0.104899  0.236121  1.000000 -0.151174 -0.264094 -0.366265
oldpeak   0.258243  0.161908  0.047734 -0.151174  1.000000  0.281817  0.443084
ca        0.370416  0.093705  0.051606 -0.264094  0.281817  1.000000  0.516216
num       0.339596  0.122291 -0.231547 -0.366265  0.443084  0.516216  1.000000
```



In [53]:
```
# Display boxplot to visualize outliers in the data

df.boxplot()
plt.show()
```



In [56]:
```
df.loc[df['chol']==0,:]
```

Out[56]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	thal	num
597	32	Male	typical angina	95.0	0.0	NaN	normal	127.0	False	0.7	upsloping	NaN	NaN	1
598	34	Male	asymptomatic	115.0	0.0	NaN	NaN	154.0	False	0.2	upsloping	NaN	NaN	1
599	35	Male	asymptomatic	NaN	0.0	NaN	normal	130.0	True	NaN	NaN	NaN	reversable defect	3
600	36	Male	asymptomatic	110.0	0.0	NaN	normal	125.0	True	1.0	flat	NaN	fixed defect	1
601	38	Female	asymptomatic	105.0	0.0	NaN	normal	166.0	False	2.8	upsloping	NaN	NaN	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
818	43	Male	asymptomatic	122.0	0.0	False	normal	120.0	False	0.5	upsloping	NaN	NaN	1
819	63	Male	non-anginal	130.0	0.0	True	st-t abnormality	160.0	False	3.0	flat	NaN	NaN	0
822	48	Male	non-anginal	102.0	0.0	NaN	st-t abnormality	110.0	True	1.0	downsloping	NaN	NaN	1
839	56	Male	asymptomatic	NaN	0.0	False	lv hypertrophy	NaN	NaN	NaN	NaN	NaN	NaN	1
840	62	Male	non-anginal	NaN	0.0	True	st-t abnormality	NaN	NaN	NaN	NaN	NaN	NaN	2

172 rows × 14 columns

In [ ]:
```
#Data Cleaning
```

In [57]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       920 non-null    int64
 1   sex       920 non-null    object
 2   cp        920 non-null    object
 3   trestbps  861 non-null    float64
 4   chol      890 non-null    float64
 5   fbs       830 non-null    object
 6   restecg   918 non-null    object
 7   thalch    865 non-null    float64
 8   exang     865 non-null    object
 9   oldpeak   858 non-null    float64
 10  slope     611 non-null    object
 11  ca        309 non-null    float64
 12  thal      434 non-null    object
 13  num       920 non-null    int64
dtypes: float64(5), int64(2), object(7)
memory usage: 100.8+ KB
```

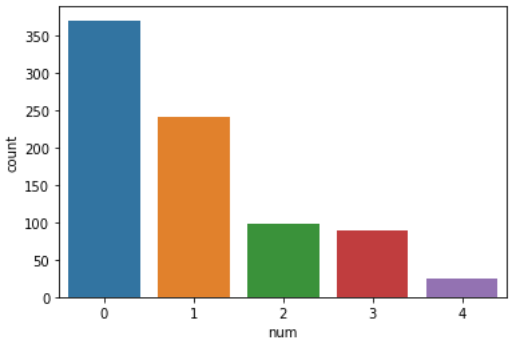
In [62]:
```
# Cholesterol Levels

median_chol = df.loc[df['chol']!=0, 'chol'].median()
df_chol = df.fillna(value={'chol': median_chol})
df_chol.loc[df_chol['chol']==0, 'chol'] = median_chol
```

In [63]:
```
# Resting Blood Pressure

mean_bp = df.loc[df['trestbps']!=0,'trestbps'].mean()
df_bp = df.fillna(value={'trestbps': mean_bp})
df_bp.loc[df_bp['trestbps']==0, 'trestbps'] = mean_bp
```

In [65]:
```
# Maximum Heart Rate

mean_hr = df.loc[df['thalch']!=0,'thalch'].mean()
df_hr = df.fillna(value={'thalch': mean_hr})
df_hr.loc[df_hr['thalch']==0, 'thalch'] = mean_hr
```

In [66]:
```
# Old Peak

mean_peak = df.oldpeak.mean()
df_pk = df.fillna(value={'oldpeak': mean_peak})
df_pk.loc[df_pk['oldpeak']==0, 'oldpeak'] = mean_peak
```

In [67]:
```
# Drop columns with a great number of missing values and reassign datatypes

df.drop(labels=['ca','thal','slope'], axis=1, inplace=True)
df = df.astype({'sex':'category', 'cp':'category', 'fbs':'bool', 'restecg':'category', 'exang':'bool'})

# Drop remaining rows with missing values and display distribution for target variables

df.dropna(inplace=True)
sns.countplot('num', data=df)
plt.show()
```



In [ ]:
```
#Preparing the Data for MOdel Training
```

In [68]:
```
# One hot encode the categorical variables and split the target and independent variables
df_onehot = pd.get_dummies(df, columns=['sex','cp', 'fbs', 'restecg', 'exang'])

X = df_onehot.drop('num', axis=1)
y = df_onehot.num

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

y_train.value_counts()
```

Out[68]:
```
0    298
1    196
2    78
3    70
4    18
Name: num, dtype: int64
```

In [69]:
```
df_onehot.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 825 entries, 0 to 919
Data columns (total 19 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       825 non-null    int64
 1   trestbps                  825 non-null    float64
 2   chol                      825 non-null    float64
 3   thalch                    825 non-null    float64
 4   oldpeak                   825 non-null    float64
 5   num                       825 non-null    int64
 6   sex_Female                825 non-null    uint8
 7   sex_Male                  825 non-null    uint8
 8   cp_asymptomatic           825 non-null    uint8
 9   cp_atypical angina        825 non-null    uint8
 10  cp_non-anginal            825 non-null    uint8
 11  cp_typical angina         825 non-null    uint8
 12  fbs_False                 825 non-null    uint8
 13  fbs_True                  825 non-null    uint8
 14  restecg_lv hypertrophy    825 non-null    uint8
 15  restecg_normal            825 non-null    uint8
 16  restecg_st-t abnormality  825 non-null    uint8
 17  exang_False               825 non-null    uint8
 18  exang_True                825 non-null    uint8
dtypes: float64(4), int64(2), uint8(13)
memory usage: 55.6 KB
```

In [73]:
```
weights = {0:1, 1:0.5, 2:0.5, 3:0.5, 4:0.5}

clf = DecisionTreeClassifier(criterion='entropy', max_depth=5)
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.74      0.70      0.72        73
           1       0.39      0.61      0.47        46
           2       0.12      0.10      0.11        21
           3       0.12      0.05      0.07        19
           4       0.00      0.00      0.00         6

    accuracy                           0.50       165
   macro avg       0.28      0.29      0.28       165
weighted avg       0.47      0.50      0.47       165
```

In [74]:
```
# Perform Decision Tree model with class weighting
weights = {0:1, 1:0.5, 2:0.5, 3:0.5, 4:0.5}

clf = DecisionTreeClassifier(criterion='entropy', max_depth=5, class_weight='balanced')
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.59      0.72        73
           1       0.26      0.13      0.17        46
           2       0.17      0.38      0.24        21
           3       0.24      0.53      0.33        19
           4       0.11      0.17      0.13         6

    accuracy                           0.41       165
   macro avg       0.34      0.36      0.32       165
weighted avg       0.54      0.41      0.44       165
```

In [75]:
```
gradient_booster = GradientBoostingClassifier(learning_rate=0.02, max_depth=3, n_estimators=150)
gradient_booster.fit(X_train, y_train)
y_pred = gradient_booster.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.71      0.86      0.78        73
           1       0.41      0.48      0.44        46
           2       0.11      0.05      0.07        21
           3       0.11      0.05      0.07        19
           4       0.00      0.00      0.00         6

    accuracy                           0.53       165
   macro avg       0.27      0.29      0.27       165
weighted avg       0.45      0.53      0.48       165
```

In [76]:
```
clf = RandomForestClassifier(n_estimators=150)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.79      0.77        73
           1       0.38      0.52      0.44        46
           2       0.27      0.19      0.22        21
           3       0.40      0.21      0.28        19
           4       0.00      0.00      0.00         6

    accuracy                           0.55       165
   macro avg       0.36      0.34      0.34       165
weighted avg       0.52      0.55      0.52       165
```

In [77]:
```
clf = RandomForestClassifier(n_estimators=150, class_weight='balanced_subsample')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.75      0.76        73
           1       0.38      0.54      0.45        46
           2       0.33      0.24      0.28        21
           3       0.42      0.26      0.32        19
           4       0.00      0.00      0.00         6

    accuracy                           0.55       165
   macro avg       0.38      0.36      0.36       165
weighted avg       0.54      0.55      0.53       165
```

In [82]:
```
smt = SMOTE(sampling_strategy='not majority')

print('Before', y_train.value_counts())

X_train_SM, y_train_SM = smt.fit_resample(X_train, y_train)

val, counter = np.unique(y_train_SM, return_counts=True)
print('After', (val, counter))
```

```
Before 0    298
1    196
2    78
3    70
4    18
Name: num, dtype: int64
After (array([0, 1, 2, 3, 4]), array([298, 298, 298, 298, 298]))
```

In [83]:
```
clf = DecisionTreeClassifier(criterion='entropy', max_depth=6)
clf.fit(X_train_SM, y_train_SM)
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.68      0.75        73
           1       0.39      0.26      0.31        46
           2       0.15      0.19      0.17        21
           3       0.21      0.37      0.27        19
           4       0.07      0.17      0.10         6

    accuracy                           0.45       165
   macro avg       0.33      0.33      0.32       165
weighted avg       0.52      0.45      0.48       165
```

In [ ]:
```

```