In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
heart_df=pd.read_csv('Heart.csv')
```

In [6]:
```
heart_df.shape
```

Out[6]:
```
(303, 15)
```

In [7]:
```
heart_df.head()
```

Out[7]:

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

In [26]:
```
pd.DataFrame({
'Column': ['age',
'sex',
'cp',
'trestbps',
'chol',
'fbs',
'restecg',
'thalach',
'exang',
'oldpeak',
'slope',
'ca',
'thal',
'target'
],
'Description':['age in years',
'(1 = male; 0 = female)',
'chest pain type',
"resting blood pressure (in mm Hg on admission to the hospital)",
'serum cholestoral in mg/dl',
'(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)',
'resting electrocardiographic results',
'maximum heart rate achieved',
'exercise induced angina (1 = yes; 0 = no)',
'ST depression induced by exercise relative to rest',
'the slope of the peak exercise ST segment',
'number of major vessels (0-3) colored by flourosopy',
'0 = normal; 1 = fixed defect; 2 = reversable defect',
'1 or 0'
]
})
```

Out[26]:

	Column	Description
0	age	age in years
1	sex	(1 = male; 0 = female)
2	cp	chest pain type
3	trestbps	resting blood pressure (in mm Hg on admission ...
4	chol	serum cholestoral in mg/dl
5	fbs	(fasting blood sugar > 120 mg/dl) (1 = true; 0...
6	restecg	resting electrocardiographic results
7	thalach	maximum heart rate achieved
8	exang	exercise induced angina (1 = yes; 0 = no)
9	oldpeak	ST depression induced by exercise relative to ...
10	slope	the slope of the peak exercise ST segment
11	ca	number of major vessels (0-3) colored by flour...
12	thal	0 = normal; 1 = fixed defect; 2 = reversable d...
13	target	1 or 0

In [29]:
```
heart_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  303 non-null    int64  
 1   age         303 non-null    int64  
 2   sex         303 non-null    int64  
 3   cp          303 non-null    int64  
 4   trestbps    303 non-null    int64  
 5   chol        303 non-null    int64  
 6   fbs         303 non-null    int64  
 7   restecg     303 non-null    int64  
 8   thalach     303 non-null    int64  
 9   exang       303 non-null    int64  
 10  oldpeak     303 non-null    float64
 11  slope       303 non-null    int64  
 12  ca          303 non-null    int64  
 13  thal        303 non-null    int64  
 14  target      303 non-null    int64  
dtypes: float64(1), int64(14)
memory usage: 35.6 KB
```

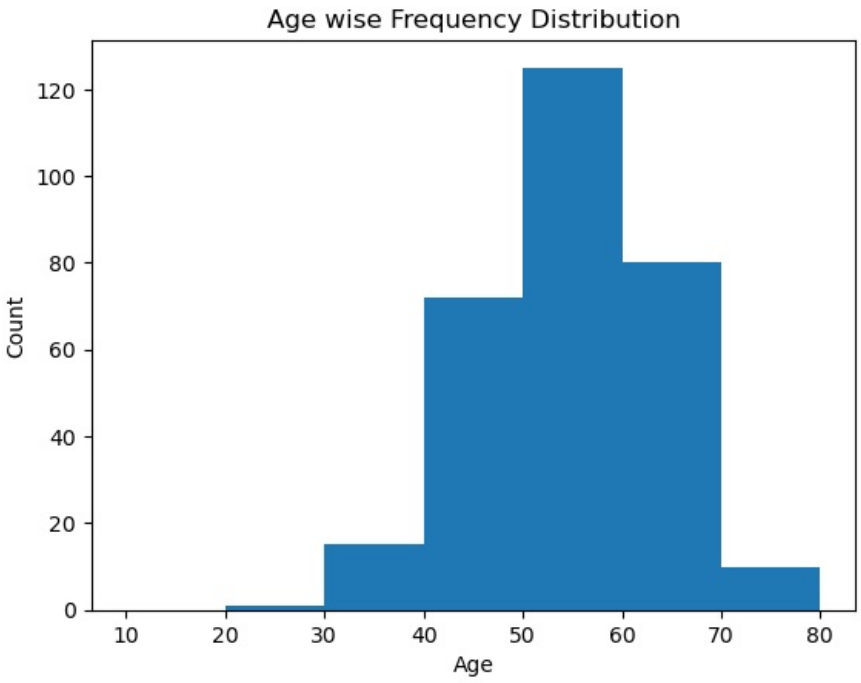
In [18]:
```
heart_df.describe()
```

Out[18]:

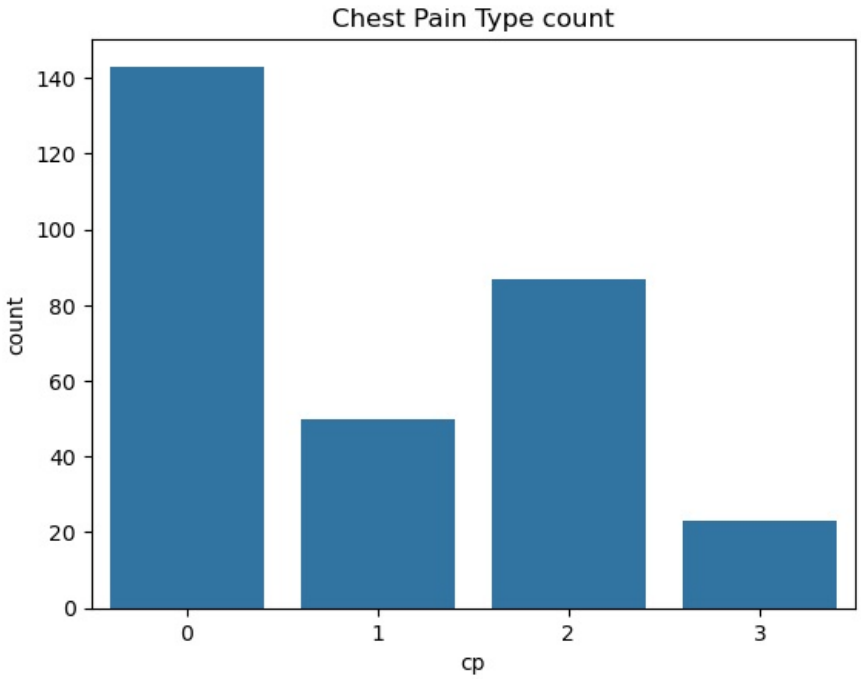
	Unnamed: 0	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	3
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	151.000000	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	
std	87.612784	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	
min	0.000000	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	
25%	75.500000	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	
50%	151.000000	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	
75%	226.500000	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	
max	302.000000	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	

In [19]:
```
plt.hist(heart_df['age'], bins=[10, 20, 30, 40, 50, 60, 70, 80])
plt.title("Age wise Frequency Distribution")
plt.xlabel('Age')
plt.ylabel('Count')
```

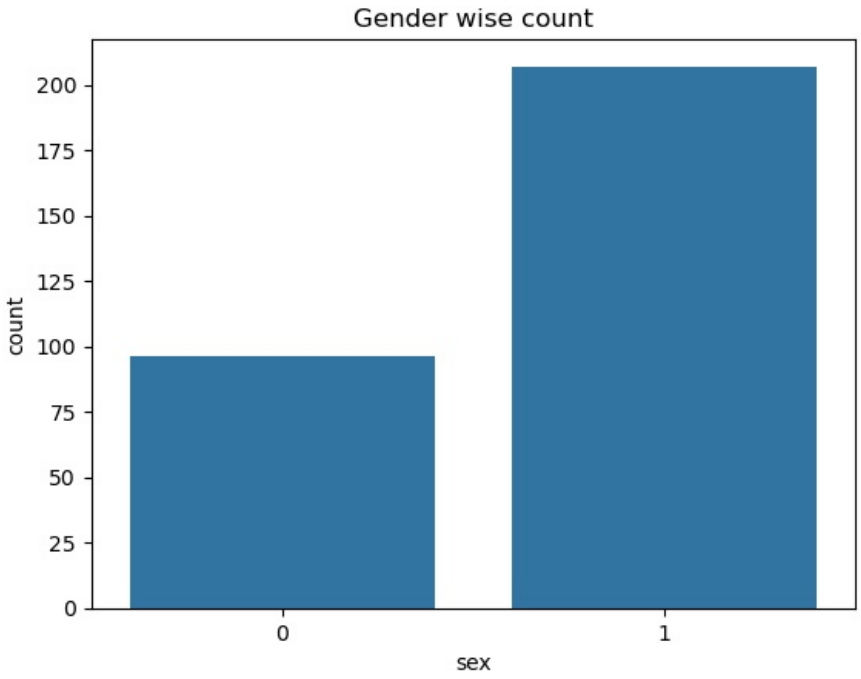
Out[19]:
```
Text(0, 0.5, 'Count')
```



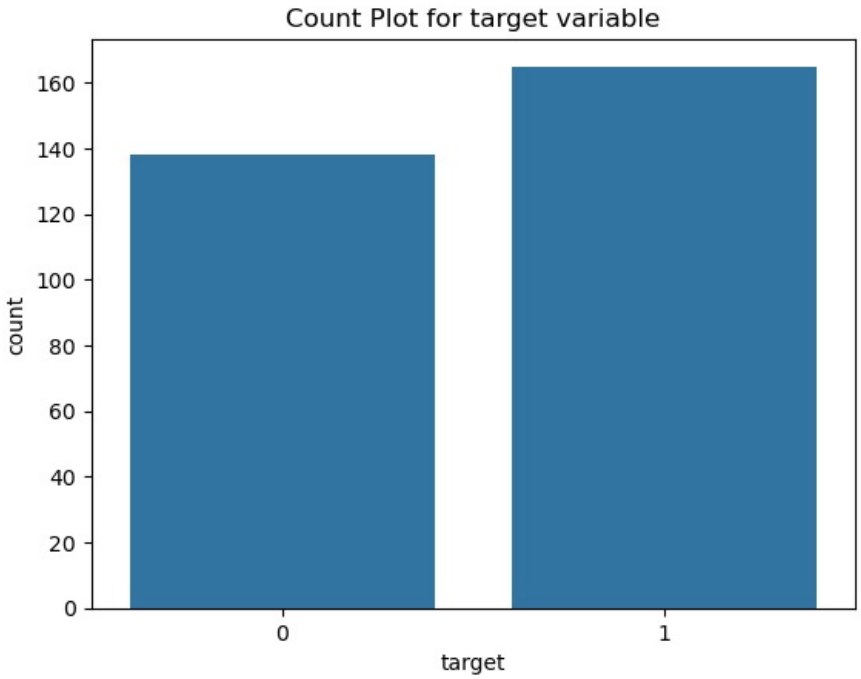
In [21]:
```
sns.countplot(data=heart_df, x='cp')
plt.title("Chest Pain Type count")
plt.show()
```



In [23]:
```
sns.countplot(data=heart_df, x='sex')
plt.title("Gender wise count")
plt.show()
```



In [35]:
```
sns.countplot(data=heart_df, x='target')
plt.title("Count Plot for target variable")
plt.show()
```



In [36]:
```
sns.countplot(data=heart_df, x='target', hue='sex')
plt.title("Count Plot for target variable w.r.t gender")
plt.show()
```

In [37]:
```
heart_df.duplicated().sum()
```

Out[37]:
```
0
```

In [38]:
```
heart_df.drop_duplicates(inplace=True)
```

In [39]:
```
heart_df.shape
```

Out[39]:
```
(303, 15)
```

In [40]:
```
X = heart_df.drop('target', axis=1)
y = heart_df['target']
```

In [41]:
```
from sklearn.preprocessing import StandardScaler
```

In [42]:
```
scaler = StandardScaler()
```

In [43]:
```
X = scaler.fit_transform(X)
```

In [44]:
```
from sklearn.model_selection import train_test_split
```

In [45]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
 random_state=42)
```

In [47]:
```
from sklearn.linear_model import LogisticRegression
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
```

Out[47]:
```
LogisticRegression()
```

In [48]:
```
logistic_model.score(X_train, y_train)
```

Out[48]:
```
0.987603305785124
```

In [49]:
```
logistic_model.score(X_test, y_test)
```

Out[49]:
```
0.9672131147540983
```

In [50]:
```
y_predict = logistic_model.predict(X_test)
```

In [53]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, recall_score,precision_score, f1_score
```

In [54]:
```
confusion_matrix(y_test, y_predict)
```

Out[54]:
```
array([[29,  0],
       [ 2, 30]], dtype=int64)
```

In [55]:
```
accuracy_score(y_test, y_predict)
```

Out[55]:
```
0.9672131147540983
```

In [56]:
```
precision_score(y_test, y_predict)
```

Out[56]:
```
1.0
```

In [57]:
```
recall_score(y_test, y_predict)
```

Out[57]:
```
0.9375
```

In [59]:
```
f1_score(y_test, y_predict)
```

Out[59]:
```
0.967741935483871
```

In [ ]:
```

```