In [28]:

```
# Heart Disease Data Usage with Classification and Regression
```

In [29]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [30]:

```
df=pd.read_csv("heart.csv")
df.head()
```

Out[30]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

In [31]:

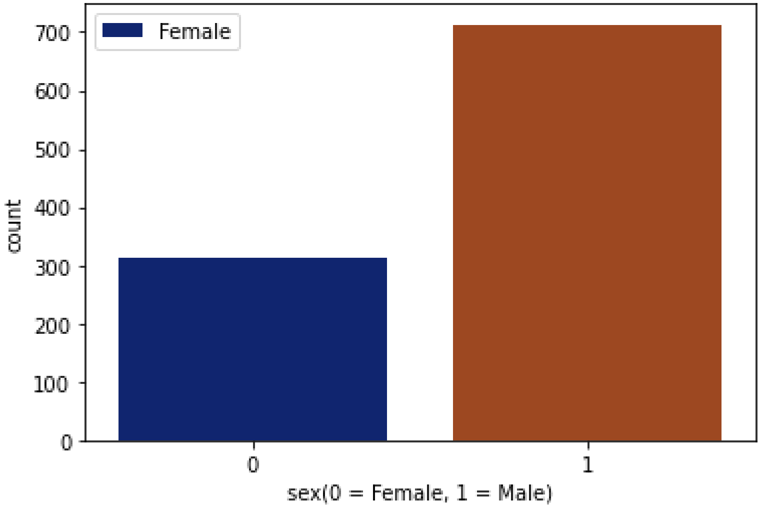
```
print("Percentage of patients with Heart disease:{:.2f}%".format(len(df[df.target==1])*100/len(df.target)))

print("Percentage of patients with no Heart disease:{:.2f}%".format(len(df[df.target==0])*100/len(df.target)))
```

```
Percentage of patients with Heart disease:51.32%
Percentage of patients with no Heart disease:48.68%
```

In [32]:

```
sns.countplot(x='sex',data=df,palette="dark")
plt.legend(["Female","Male"])
plt.xlabel('sex(0 = Female, 1 = Male)')
plt.show()
```

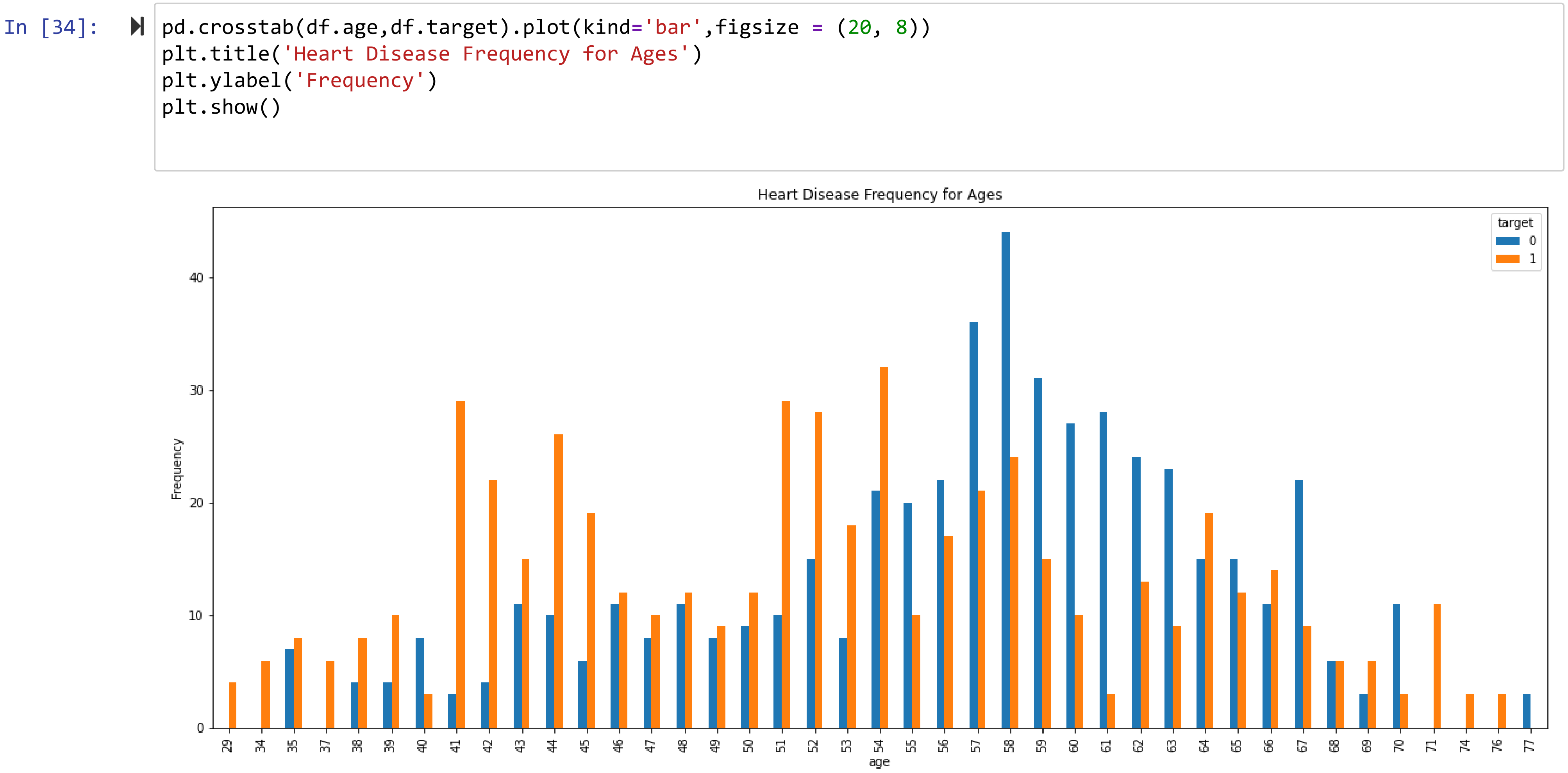


Observation: Male Members were high Disease Rate

In [33]:

```
print("Percentage of Female Patients:{:.2f}%".format(len(df[df.sex==0])*100/len(df.sex)))
print("Percentage of Male Patients:{:.2f}%".format(len(df[df.sex==1])*100/len(df.sex)))
```

```
Percentage of Female Patients:30.44%
Percentage of Male Patients:69.56%
```



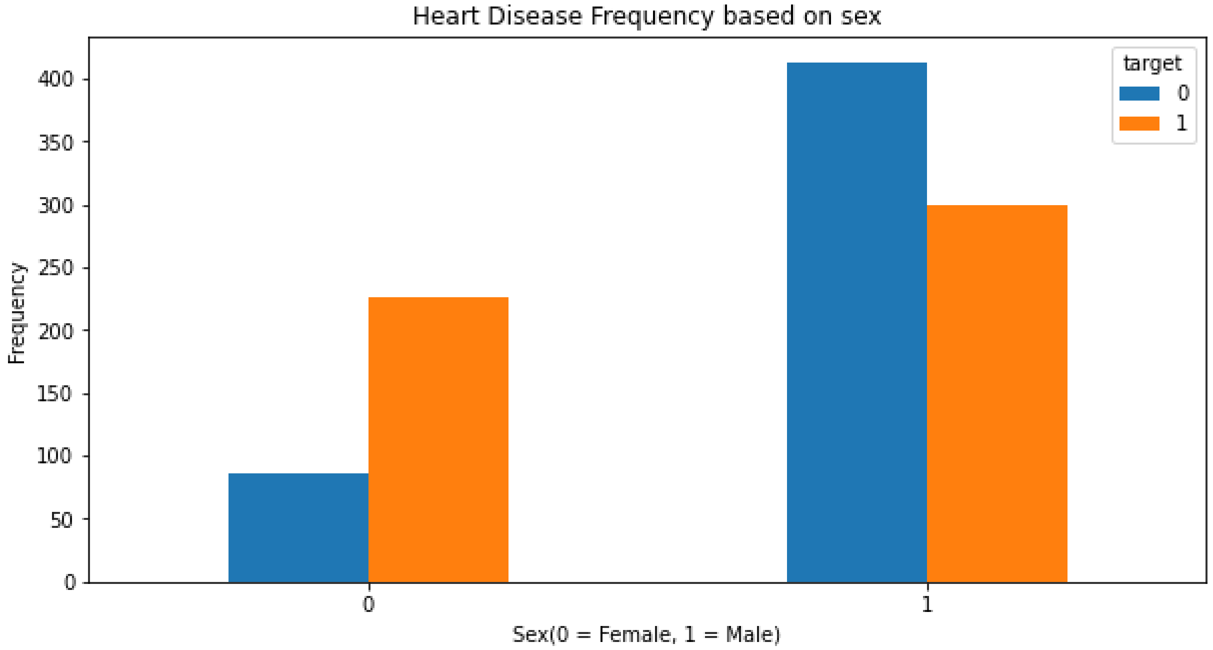
In [34]:

```
pd.crosstab(df.age,df.target).plot(kind='bar',figsize = (20, 8))
plt.title('Heart Disease Frequency for Ages')
plt.ylabel('Frequency')
plt.show()
```

Observation: Male At 58 Age More Frequency of Getting Heart Disease, Female at 54 Age max Risk

In [35]:

```
pd.crosstab(df.sex,df.target).plot(kind='bar',figsize = (10, 5))
plt.title('Heart Disease Frequency based on sex')
plt.xticks(rotation=0)
plt.xlabel('Sex(0 = Female, 1 = Male)')
plt.ylabel('Frequency')
plt.show()
```



In [36]:

```
x = df.drop(['target'], axis = 1)
```

In [37]:

```
y = df.target.values
```

In [38]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.3,random_state=0)
```

Logistic Regression Usage

In [39]:

```
accuracies = {}

lr = LogisticRegression()
lr.fit(x_train,y_train)
acc = lr.score(x_test,y_test)*100

accuracies['Logistic Regression'] = acc
print("Test Accuracy Logistic {:.2f}%".format(acc))
```

```
Test Accuracy Logistic 87.01%

C:\Users\I5262\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessi
ng.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/
modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

K Nearest Neighbor

In [40]:

```
# KNN Model
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 2)  # n_neighbors means k
knn.fit(x_train, y_train)
prediction = knn.predict(x_test)
acc=knn.score(x_test, y_test)*100
accuracies['KNN'] = acc
#print("{} KNN Score: {:.2f}%".format(acc))
print("Test Accuracy KNN {:.2f}%".format(acc))
```

```
Test Accuracy KNN 92.53%
```

Support Vector Classifier

In [41]:

```
from sklearn.svm import SVC
svm = SVC(random_state = 1)
svm.fit(x_train, y_train)

acc = svm.score(x_test,y_test)*100
accuracies['SVM'] = acc
print("Test Accuracy of SVM Algorithm: {:.2f}%".format(acc))
```

```
Test Accuracy of SVM Algorithm: 75.00%
```

Naive Bayes Algorithm

In [42]:

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train, y_train)

acc = nb.score(x_test,y_test)*100
accuracies['Naive Bayes'] = acc
print("Test Accuracy of Naive Bayes: {:.2f}%".format(acc))
```

```
Test Accuracy of Naive Bayes: 84.42%
```

In [43]:

```
# Decison Tree
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(x_train, y_train)

acc = dtc.score(x_test, y_test)*100
accuracies['Decision Tree'] = acc
print("Test Accuracy Decision Tree {:.2f}%".format(acc))
```

```
Test Accuracy Decision Tree 100.00%
```

In [44]:

```
# Random Forest Classification
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 1000, random_state = 2)
rf.fit(x_train, y_train)

acc = rf.score(x_test,y_test)*100
accuracies['Random Forest'] = acc
print("Test Accuracy Random Forest : {:.2f}%".format(acc))
```

```
Test Accuracy Random Forest : 100.00%
```

Observations:

1.Test Accuracy Logistic : 87.01% 2.Test Accuracy KNN : 92.53% 3.Test Accuracy of SVM Algorithm: 75.00% 4.Test Accuracy of Naive Bayes: 84.42% 5.Test Accuracy Decision Tree 100.00% 6.Test Accuracy Random Forest : 100.00%

So Decision Tree and Random Forest Recorded High Accuracy in this Heart Disease Data Set is 100% Next Best Accuracy Recorded By KNN is 92.53% Next Best Accuracy is 87.01%