In [1]:

```
#Importing the Dependencies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#Data Collection
data=pd.read_csv("E:\Data Science\Titanicdataset.csv")
data
```

Out[1]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	
...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	

891 rows × 12 columns

In [2]:
```
#Data processing
data.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cab
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na

In [3]:
```
data.describe()
```

Out[3]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [4]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [5]:

```
#print the Nullvalues in each column
data.isnull().sum()
```

Out[5]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [6]:

```
#Handle the Null values
data["Age"].fillna(data["Age"].mean(skipna=True),inplace=True)
data.isnull().sum()
```

Out[6]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [7]:

```
#Drop the Cabin column
data=data.drop(columns='Cabin',axis=1)
data.isnull().sum()
```

Out[7]:

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       2
dtype: int64
```

In [29]:

```
data.tail()
```

Out[29]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
886	887	0	2	0	27.000000	0	0	211536	13.00	0
887	888	1	1	1	19.000000	0	0	112053	30.00	0
888	889	0	3	1	29.699118	1	2	W./C. 6607	23.45	0
889	890	1	1	0	26.000000	0	0	111369	30.00	1
890	891	0	3	0	32.000000	0	0	370376	7.75	2

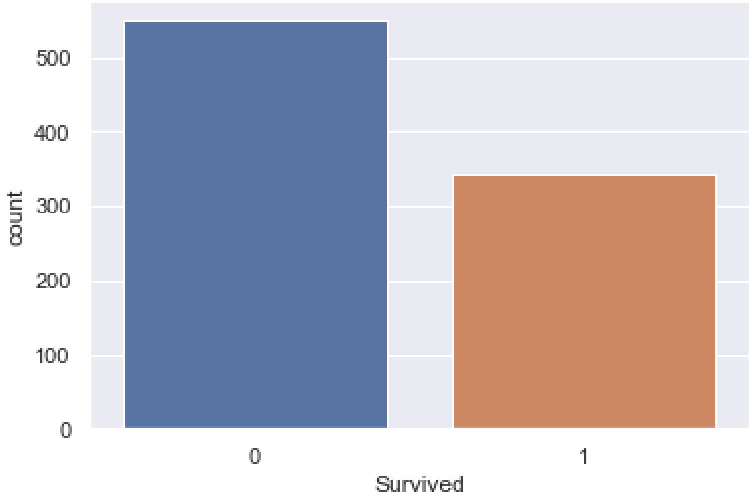
In [30]:

```
import seaborn as sns
sns.countplot(data['Survived'])
#0-Not Survived;1-Survived
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[30]:

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



In [10]:

```
#Converting categorical into numerical value
print(data['Embarked'].mode())
```

```
0    S
Name: Embarked, dtype: object
```

In [11]:

```
data['Embarked'].fillna(data['Embarked'].mode()[0],inplace=True)
```

In [12]:

```
data.isnull().sum()
```

Out[12]:

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [13]:

```
data['Survived'].value_counts()
```

Out[13]:

```
0    549
1    342
Name: Survived, dtype: int64
```

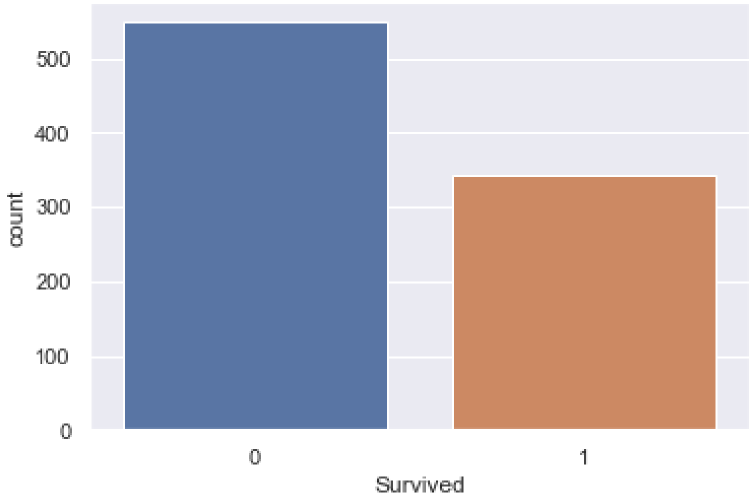
In [14]:

```
#Data Visualization
sns.set()
sns.countplot('Survived',data=data)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[14]:

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



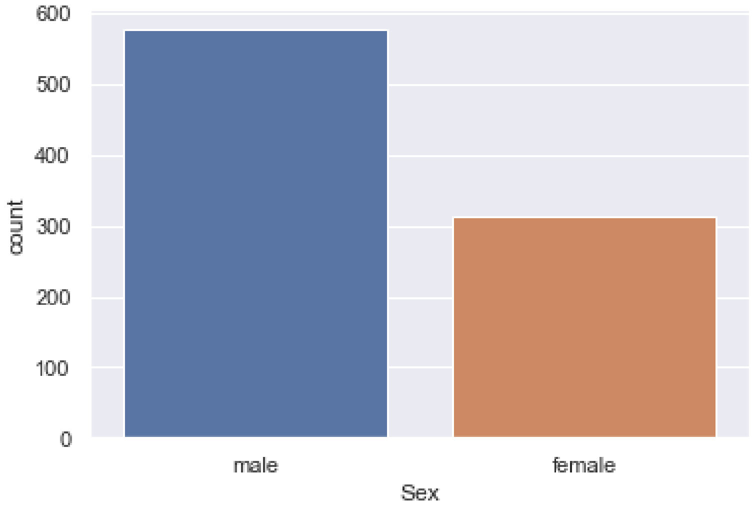
In [15]:

```
sns.countplot('Sex',data=data)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[15]:

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



In [16]:

```
data['Sex'].value_counts()
```

Out[16]:

```
male      577
female    314
Name: Sex, dtype: int64
```

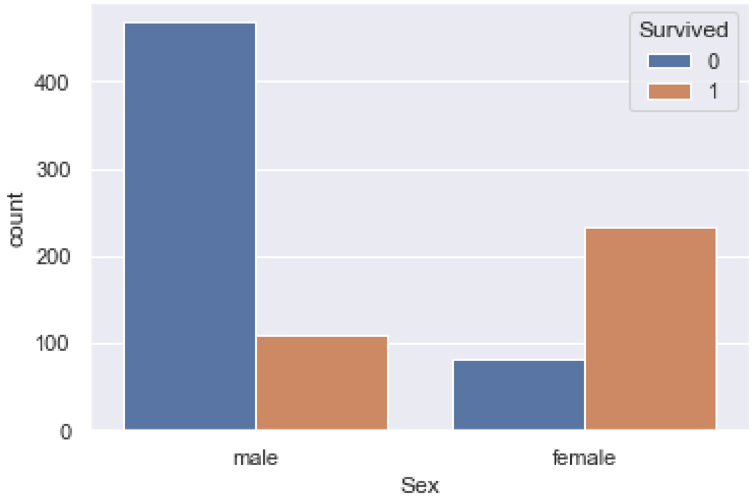
In [17]:

```
sns.countplot('Sex',hue='Survived',data=data)
#0-Not Survived;1-Survived
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[17]:

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



In [18]:
```python
data=data.drop('Name',axis=1)
data
```

Out[18]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embark
0	1	0	3	male	22.000000	1	0	A/5 21171	7.2500	
1	2	1	1	female	38.000000	1	0	PC 17599	71.2833	
2	3	1	3	female	26.000000	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	female	35.000000	1	0	113803	53.1000	
4	5	0	3	male	35.000000	0	0	373450	8.0500	
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	male	27.000000	0	0	211536	13.0000	
887	888	1	1	female	19.000000	0	0	112053	30.0000	
888	889	0	3	female	29.699118	1	2	W./C. 6607	23.4500	
889	890	1	1	male	26.000000	0	0	111369	30.0000	
890	891	0	3	male	32.000000	0	0	370376	7.7500	

891 rows × 10 columns

In [19]:
```python
data['Embarked'].value_counts()
```

Out[19]:
```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

In [20]:
```python
data.replace({'Sex':{'male':0,'female':1},'Embarked':{'S':0,'C':1,'Q':2
}},inplace=True)
```

In [21]:
```python
data.head()
```

Out[21]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	0	22.0	1	0	A/5 21171	7.2500	0
1	2	1	1	1	38.0	1	0	PC 17599	71.2833	1
2	3	1	3	1	26.0	0	0	STON/O2. 3101282	7.9250	0
3	4	1	1	1	35.0	1	0	113803	53.1000	0
4	5	0	3	0	35.0	0	0	373450	8.0500	0

In [22]:

```
#Training the data by using different models
from sklearn.model_selection import train_test_split
X=data.drop(columns=['PassengerId','Ticket','Survived'],axis=1)
Y=data['Survived']
print(X)
print(Y)
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=4
```

```
     Pclass  Sex        Age  SibSp  Parch     Fare  Embarked
0         3    0  22.000000      1      0   7.2500         0
1         1    1  38.000000      1      0  71.2833         1
2         3    1  26.000000      0      0   7.9250         0
3         1    1  35.000000      1      0  53.1000         0
4         3    0  35.000000      0      0   8.0500         0
..      ...  ...        ...    ...    ...      ...       ...
886       2    0  27.000000      0      0  13.0000         0
887       1    1  19.000000      0      0  30.0000         0
888       3    1  29.699118      1      2  23.4500         0
889       1    0  26.000000      0      0  30.0000         1
890       3    0  32.000000      0      0   7.7500         2

[891 rows x 7 columns]
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

In [23]:

```
# LogisticRegression()
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train,Y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:8
14: 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/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion)
  n_iter_i = _check_optimize_result(
```

Out[23]:

```
LogisticRegression()
```

In [24]:

```
lr=model.score(X_test,Y_test)*100
print("Accuracy of Titanic Survival Prediction using Logistic Regression is",lr
```

```
Accuracy of Titanic Survival Prediction using Logistic Regression is 79.888268
15642457
```

In [25]:

```
#LinearRegression()
from sklearn.linear_model import LinearRegression
model1=LinearRegression()
model1.fit(X_train,Y_train)
lr1=model1.score(X_test,Y_test)*100
print("Accuracy of Titanic Survival Prediction using Linear Regression is",lr1)
```

```
Accuracy of Titanic Survival Prediction using Linear Regression is 44.49047661
8016515
```

In [26]:

```
#DecisionTreeClassifier()
from sklearn.tree import DecisionTreeClassifier
f1=DecisionTreeClassifier()
f1.fit(X_train,Y_train)
l1=f1.score(X_test,Y_test)*100
print("Accuracy of Titanic Survival Prediction using DecisionTreeClassifier is"
```

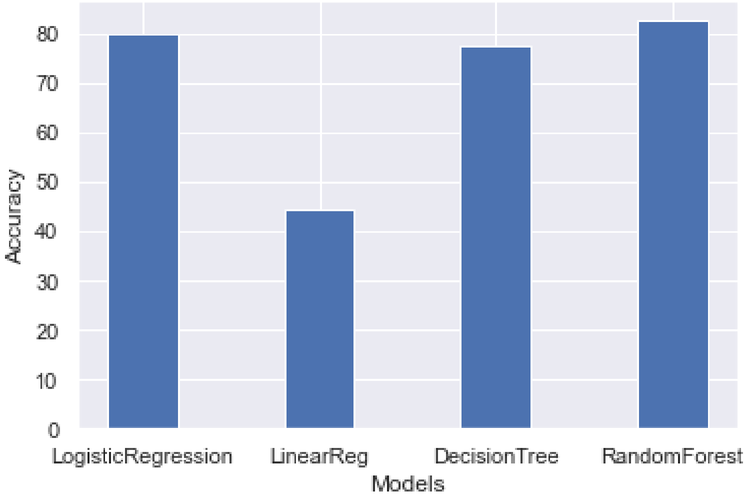
```
Accuracy of Titanic Survival Prediction using DecisionTreeClassifier is 77.653
6312849162
```

In [33]:

```
#RandomForestClassifier()
from sklearn.ensemble import RandomForestClassifier
f11=RandomForestClassifier()
f11.fit(X_train,Y_train)
l11=f11.score(X_test,Y_test)*100
print("Accuracy of Titanic Survival Prediction using RandomForestClassifier is"

#Variations among different model
S=["LogisticRegression","LinearReg","DecisionTree","RandomForest"]
T=[lr,lr1,l1,l11]
plt.bar(S,T,width=0.4)
plt.xlabel("Models")
plt.ylabel("Accuracy")
plt.show()
```

```
Accuracy of Titanic Survival Prediction using RandomForestClassifier is 82.681
56424581005
```



In [28]:

```
#Testing the data
a1=eval(input())
a2=eval(input())
a3=eval(input())
a4=eval(input())
a5=eval(input())
a6=eval(input())
a7=eval(input())
a={'Pclass':[a1],
   'Sex':[a2],
   'Age':[a3],
   'SibSp':[a4],
   'Parch':[a5],
   'Fare':[a6],
   'Embarked':[a7]
   
   
   
  }
res=f11.predict(pd.DataFrame(a))
if res:
    print("The person can survive in titanic")
else:
    print("The person cannot survive in titanic")
```

```
3
0
32
1
0
40
1
The person cannot survive in titanic
```

In [ ]:

In [ ]: