In [1]:
```
# i. Importing Libraries
```

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

In [ ]:
```

```

In [3]:
```
# ii. Acquiring Data
```

In [4]:
```
df = pd.read_csv('CampusRecruitment.csv')
```

In [5]:
```
df
```

Out[5]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	et
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	
...	...	...	...	...	...	...	...	...	...	...	
210	211	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	
211	212	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	
212	213	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	
213	214	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	
214	215	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	

215 rows × 15 columns

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

Out[6]:
```
(215, 15)
```

In [7]:

```
df.describe()
```

Out[7]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	108.000000	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	62.209324	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	1.000000	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	54.500000	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	108.000000	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	161.500000	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	215.000000	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

In [8]:

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

Out[8]:

```
sl_no              0
gender             0
ssc_p              0
ssc_b              0
hsc_p              0
hsc_b              0
hsc_s              0
degree_p           0
degree_t           0
workex             0
etest_p            0
specialisation     0
mba_p              0
status             0
salary            67
dtype: int64
```

In [9]:

```
df_copy=df.copy()
```

In [10]:

```
df_copy
```

Out[10]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	et
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	
...	...	...	...	...	...	...	...	...	...	...	
210	211	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	
211	212	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	
212	213	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	
213	214	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	
214	215	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	

215 rows × 15 columns

In [11]:

```
# Dropping unwanted columns

df_copy = df_copy.drop(['sl_no','salary','gender','ssc_b','hsc_b'], axis = 1)
df_copy
```

Out[11]:

	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	st
0	67.00	91.00	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Pl
1	79.33	78.33	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Pl
2	65.00	68.00	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Pl
3	56.00	52.00	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Pl
4	85.80	73.60	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Pl
...	...	...	...	...	...	...	...	...	...	
210	80.60	82.00	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Pl
211	58.00	60.00	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Pl
212	67.00	67.00	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Pl
213	74.00	66.00	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Pl
214	62.00	58.00	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Pl

215 rows × 10 columns

In [ ]:

```

```

In [12]:

```
# iii. Preprocessing the Data
```

In [13]:

```
# Converting Categorical Columns into Numerical Columns
```

In [14]:

```
from sklearn.preprocessing import LabelEncoder

cat_num = ['hsc_s', 'degree_t','workex','specialisation','mba_p','status']

le = LabelEncoder()

for i in cat_num:
    df_copy[i] = le.fit_transform(df_copy[i])

df_copy
```

Out[14]:

	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status
0	67.00	91.00	1	58.00	2	0	55.0	1	64	1
1	79.33	78.33	2	77.48	2	1	86.5	0	153	1
2	65.00	68.00	0	64.00	0	0	75.0	0	50	1
3	56.00	52.00	2	52.00	2	0	66.0	1	72	0
4	85.80	73.60	1	73.30	0	0	96.8	0	28	1
...	...	...	...	...	...	...	...	...	...	...
210	80.60	82.00	1	77.60	0	0	91.0	0	199	1
211	58.00	60.00	2	72.00	2	0	74.0	0	14	1
212	67.00	67.00	1	73.00	0	1	59.0	0	179	1
213	74.00	66.00	1	58.00	0	0	70.0	1	81	1
214	62.00	58.00	2	53.00	0	0	89.0	1	80	0

215 rows × 10 columns

In [15]:
```
df_copy.describe()
```

Out[15]:

	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specia
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	215
mean	67.303395	66.333163	1.372093	66.370186	0.600000	0.344186	72.100558	0
std	10.827205	10.897509	0.580978	7.358743	0.890238	0.476211	13.275956	0
min	40.890000	37.000000	0.000000	50.000000	0.000000	0.000000	50.000000	0
25%	60.600000	60.900000	1.000000	61.000000	0.000000	0.000000	60.000000	0
50%	67.000000	65.000000	1.000000	66.000000	0.000000	0.000000	71.000000	0
75%	75.700000	73.000000	2.000000	72.000000	2.000000	1.000000	83.500000	1
max	89.400000	97.700000	2.000000	91.000000	2.000000	1.000000	98.000000	1

In [16]:
```
X = df_copy.drop(['status'], axis = 1)
y = df_copy['status']
```

In [17]:
```
X
```

Out[17]:

	ssc_p	hsc_p	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p
0	67.00	91.00	1	58.00	2	0	55.0	1	64
1	79.33	78.33	2	77.48	2	1	86.5	0	153
2	65.00	68.00	0	64.00	0	0	75.0	0	50
3	56.00	52.00	2	52.00	2	0	66.0	1	72
4	85.80	73.60	1	73.30	0	0	96.8	0	28
...	...	...	...	...	...	...	...	...	...
210	80.60	82.00	1	77.60	0	0	91.0	0	199
211	58.00	60.00	2	72.00	2	0	74.0	0	14
212	67.00	67.00	1	73.00	0	1	59.0	0	179
213	74.00	66.00	1	58.00	0	0	70.0	1	81
214	62.00	58.00	2	53.00	0	0	89.0	1	80

215 rows × 9 columns

```
In [18]: y
```

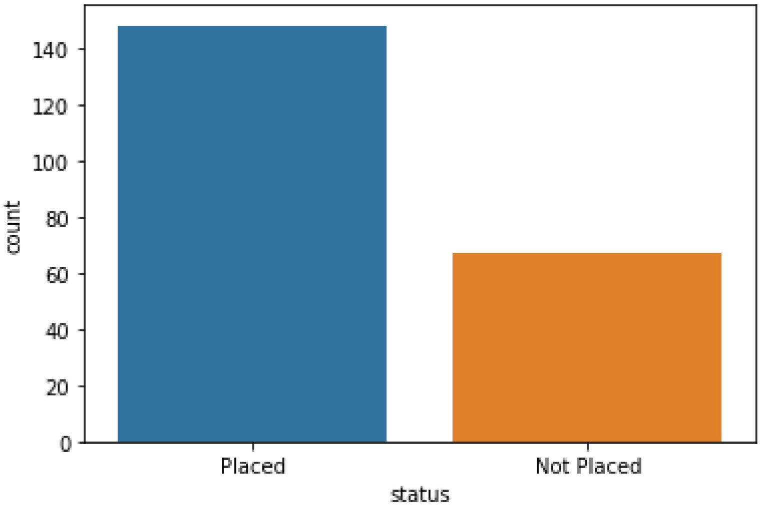
```
Out[18]: 0      1
         1      1
         2      1
         3      0
         4      1
                ..
         210    1
         211    1
         212    1
         213    1
         214    0
         Name: status, Length: 215, dtype: int32
```

```
In [ ]:
```

```
In [19]: # iv. Visualising the Data
```

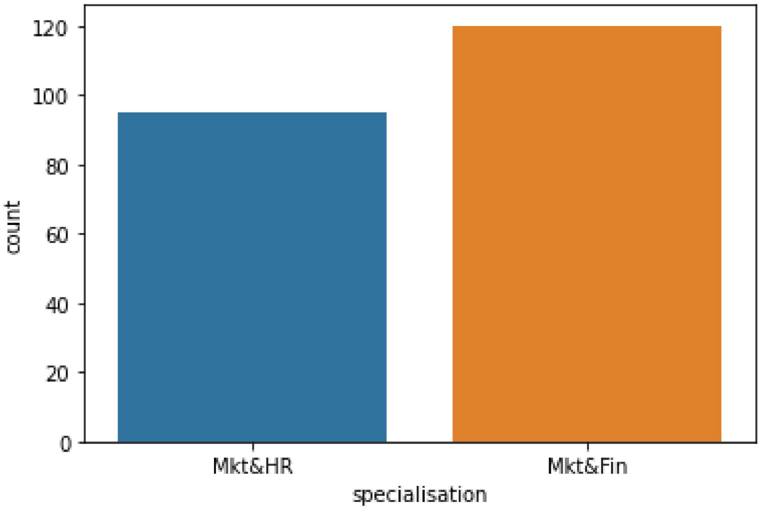
```
In [20]: sns.countplot(x = 'status', data = df)
```

```
Out[20]: <AxesSubplot:xlabel='status', ylabel='count'>
```



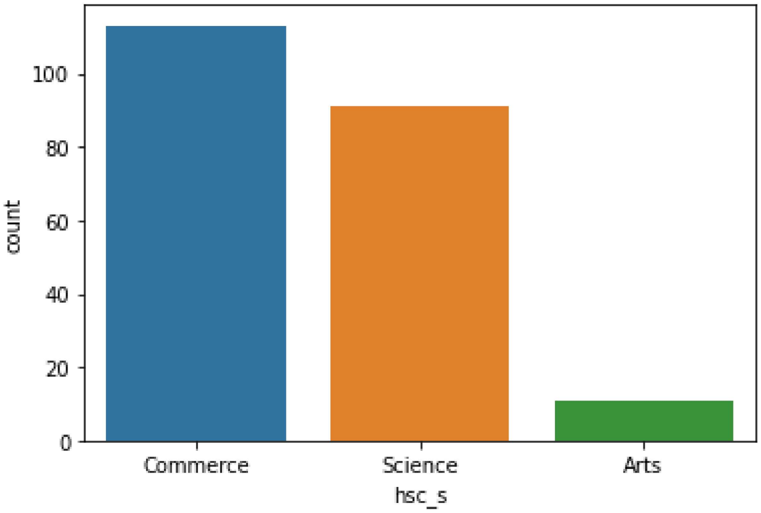
```
In [21]: sns.countplot(x = 'specialisation', data = df)
```

```
Out[21]: <AxesSubplot:xlabel='specialisation', ylabel='count'>
```



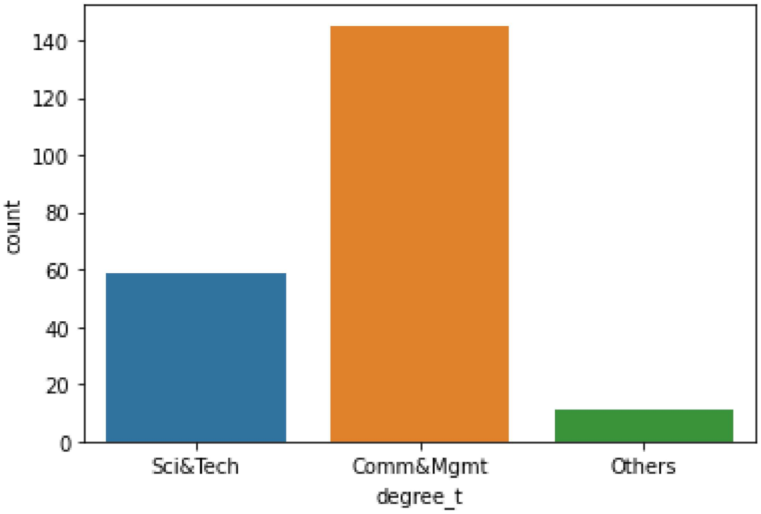
```
In [22]: sns.countplot(x = 'hsc_s', data = df)
```

```
Out[22]: <AxesSubplot:xlabel='hsc_s', ylabel='count'>
```



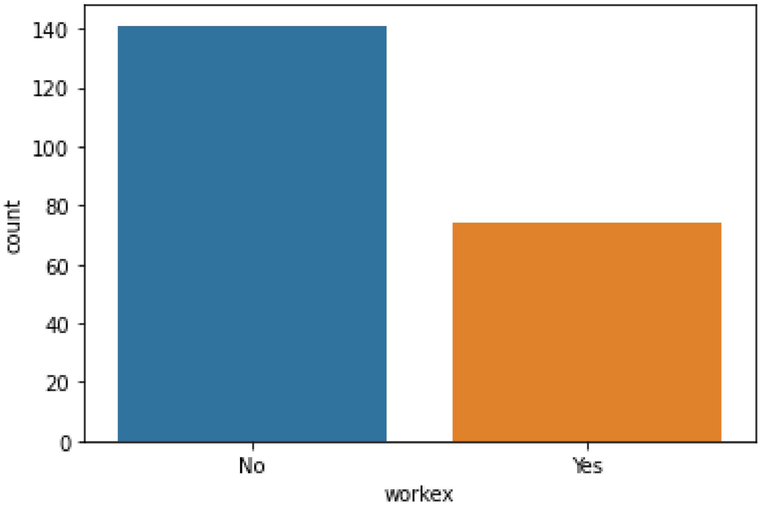
In [23]:
```
sns.countplot(x = 'degree_t', data = df)
```

Out[23]:
```
<AxesSubplot:xlabel='degree_t', ylabel='count'>
```



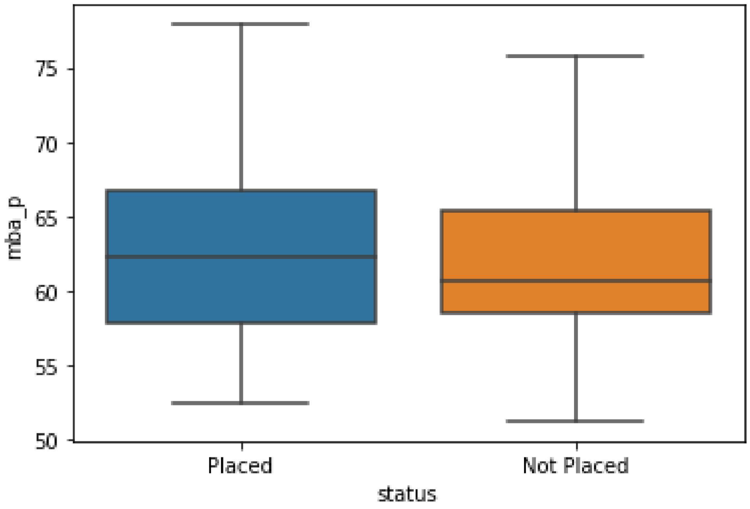
In [24]:
```
sns.countplot(x = 'workex', data = df)
```

Out[24]:
```
<AxesSubplot:xlabel='workex', ylabel='count'>
```



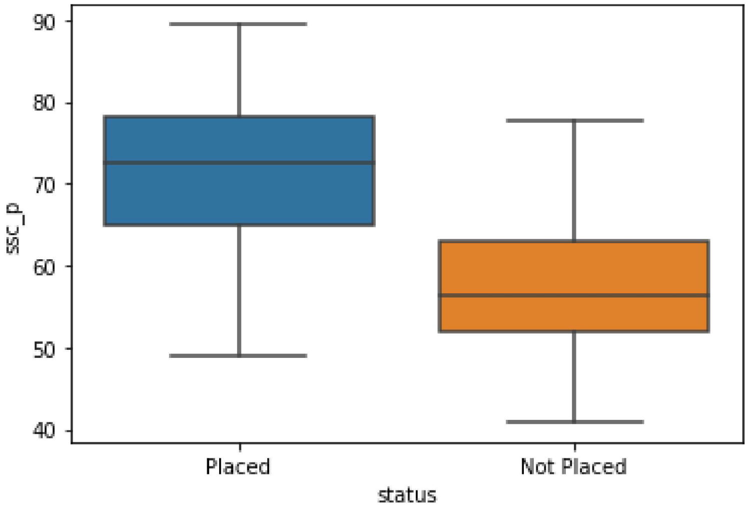
```
In [25]: # Box Plot

         sns.boxplot(x = 'status', y = 'mba_p',data = df)
```

```
Out[25]: <AxesSubplot:xlabel='status', ylabel='mba_p'>
```



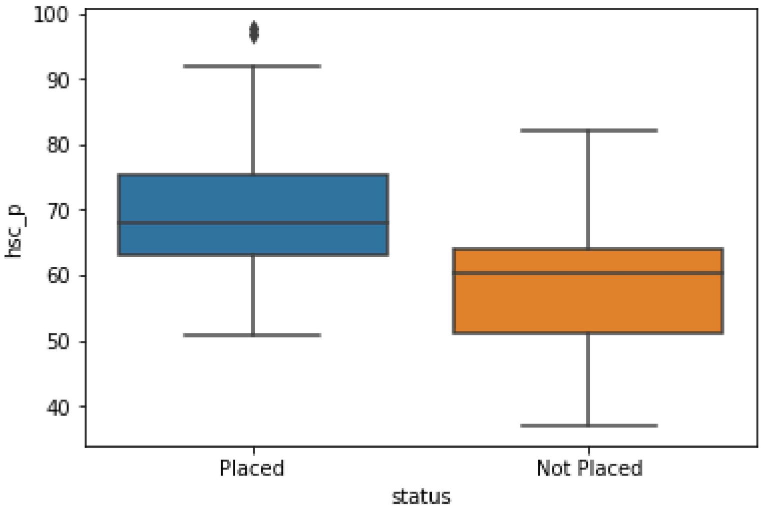
```
In [26]: sns.boxplot(x = 'status', y = 'ssc_p',data = df)
```

```
Out[26]: <AxesSubplot:xlabel='status', ylabel='ssc_p'>
```



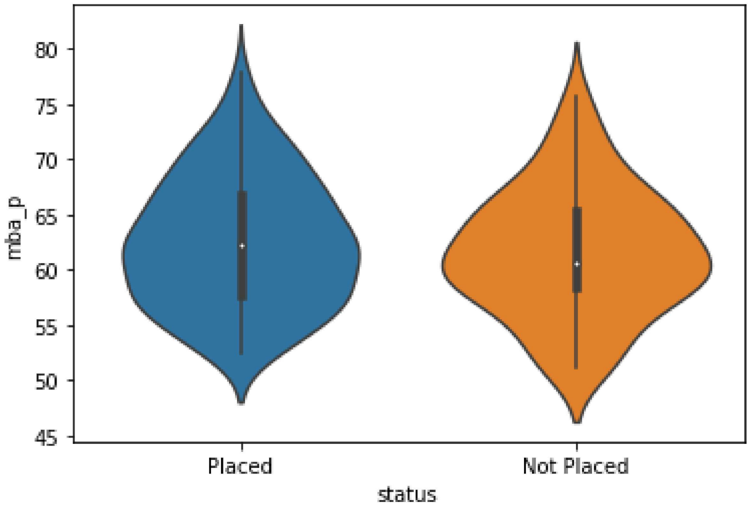
In [27]:
```
sns.boxplot(x = 'status', y = 'hsc_p',data = df)
```

Out[27]:
```
<AxesSubplot:xlabel='status', ylabel='hsc_p'>
```



In [28]:
```
sns.violinplot(x = 'status', y = 'mba_p', data = df, size = 8)
```

Out[28]:
```
<AxesSubplot:xlabel='status', ylabel='mba_p'>
```



In [ ]:

In [29]:
```
# v. Splitting Data into Training and Testing Set.
```

In [30]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X , y , train_size =0.8 , rar
```

In [31]:

```
# As our data is not normally distributed, apply standard scaler

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [ ]:

In [32]:

```
# vi. Training and Testing Data
```

In [33]:

```
# A) Logistic Regresssion
```

In [34]:

```
from sklearn.linear_model import LogisticRegression
```

In [35]:

```
my_model = LogisticRegression()
result = my_model.fit(X_train, y_train)
```

In [36]:

```
# 4) Test the Model

predictions = result.predict(X_test)
predictions
```

Out[36]:

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

In [37]:

```
from sklearn.metrics import accuracy_score

print("Accuracy Using Logistic Regression ", accuracy_score(y_test, predictions))
```

```
Accuracy Using Logistic Regression  0.8837209302325582
```

In [38]:

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

In [39]:

```
confusion_mat = confusion_matrix(y_test, predictions)
confusion_mat
```

Out[39]:

```
array([[12,  3],
       [ 2, 26]], dtype=int64)
```

In [40]:

```
confusion_df = pd.DataFrame(confusion_mat,
                            index = ['Actual neg','Actual pos'],
                           columns = ['Predicted neg', 'Predicted pos'])
```

In [41]:

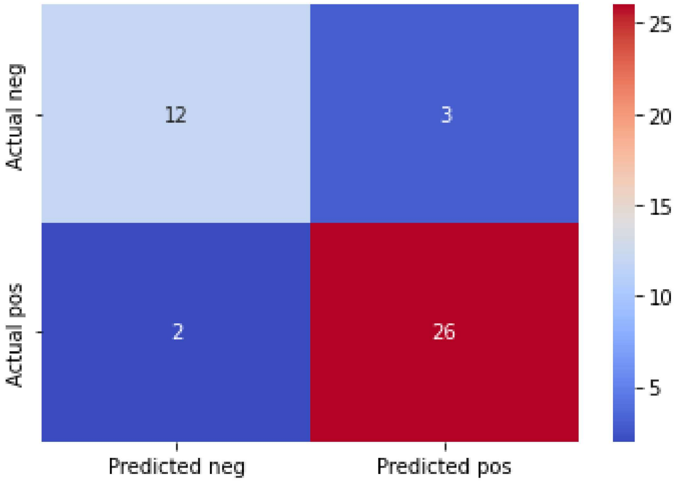
```
confusion_df
```

Out[41]:

	Predicted neg	Predicted pos
Actual neg	12	3
Actual pos	2	26

In [42]:

```
Color_conf_matrix = sns.heatmap(confusion_df, cmap = 'coolwarm',annot = True)
# annot is annotations (87,20,18,54)
```



In [43]:

```
from sklearn import metrics
print('\n**Classification Report:\n',
      metrics.classification_report(y_test,predictions))
```

```
**Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.80      0.83        15
           1       0.90      0.93      0.91        28

    accuracy                           0.88        43
   macro avg       0.88      0.86      0.87        43
weighted avg       0.88      0.88      0.88        43
```

In [44]:

```
# Deploy the Model

pred_new = my_model.predict([[67.00,91.00,1,58.00,2,0,55.0,1,64]])
pred_new
```

Out[44]:

```
array([1])
```

In [45]:

```
# Unknown Values

pred_new = my_model.predict([[89.40,69.00,2,80.00,1,0,64.0,1,64]])
pred_new
```

Out[45]:

```
array([1])
```

In [ ]:

In [46]:

```
# B) Decision Tree Classifier
```

In [47]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [48]:

```
my_model = DecisionTreeClassifier(random_state = 0)
result = my_model.fit(X_train, y_train)
```

In [49]:

```
predictions = result.predict(X_test)
predictions
```

Out[49]:

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

In [50]:

```
from sklearn.metrics import mean_absolute_error

mean_absolute_error(y_test, predictions) # mean absolute error = 1 - accuracy
```

Out[50]:

```
0.23255813953488372
```

In [51]:

```
# Accuracy Score

print("Accuracy Using Decision Tree Classifier ", accuracy_score(y_test, predicti
```

```
Accuracy Using Decision Tree Classifier  0.7674418604651163
```

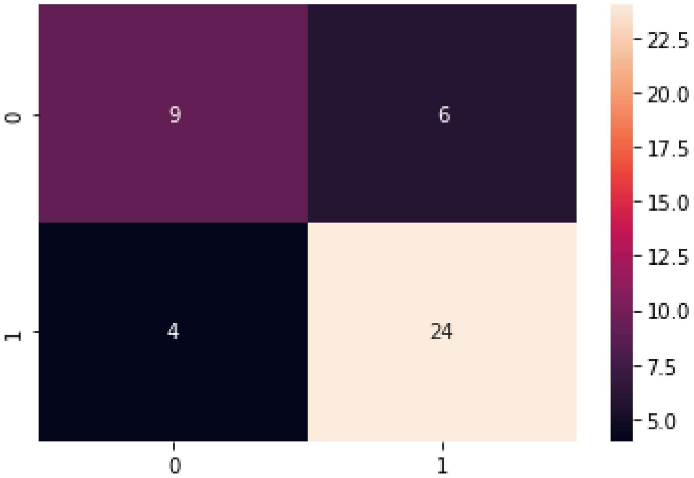
In [52]:

```
from sklearn import metrics
print('\n**Classification Report:\n',
      metrics.classification_report(y_test,predictions))
```

```

**Classification Report:
               precision    recall  f1-score   support

           0       0.69      0.60      0.64        15
           1       0.80      0.86      0.83        28

    accuracy                           0.77        43
   macro avg       0.75      0.73      0.74        43
weighted avg       0.76      0.77      0.76        43
```

In [53]:
```
from sklearn.metrics import confusion_matrix

confusion_mat = confusion_matrix(y_test, predictions)
confusion_mat

sns.heatmap(confusion_mat,annot = True)
```

Out[53]:
```
<AxesSubplot:>
```



In [54]:
```
# Deploy the Model

pred_new = my_model.predict([[67.00,91.00,1,58.00,2,0,55.0,1,64]])
pred_new
```

Out[54]:
```
array([1])
```

In [55]:
```
# Unknown Values

pred_new = my_model.predict([[89.40,69.00,2,80.00,1,0,64.0,1,64]])
pred_new
```

Out[55]:
```
array([1])
```

In [ ]:

In [56]:
```
# C) Random Forest Classifier
```

In [57]:
```
from sklearn.ensemble import RandomForestClassifier

my_model = RandomForestClassifier(n_estimators = 50, criterion = 'entropy',
                                  random_state = 42)
result = my_model.fit(X_train, y_train)
```

In [58]:
```
predictions = result.predict(X_test)
predictions
```

Out[58]:
```
array([1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1])
```

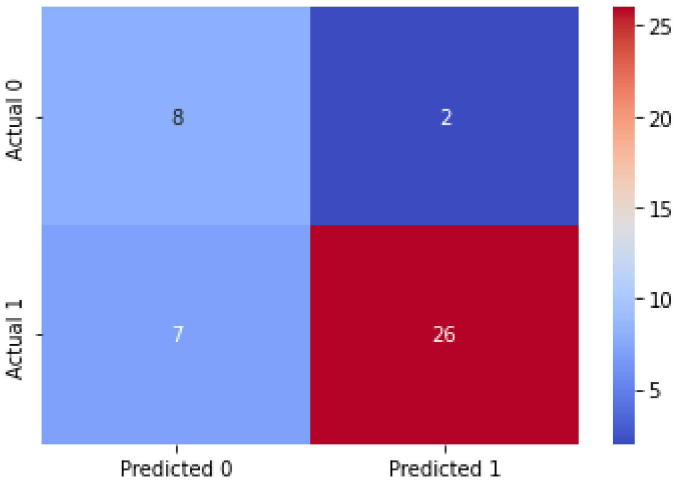
In [59]:
```
from sklearn import metrics
```

In [60]:
```
print("Accuracy using Random Forest Classifer ",metrics.accuracy_score(y_test, pr
```

```
Accuracy using Random Forest Classifer  0.7906976744186046
```

In [61]:
```
from sklearn.metrics import confusion_matrix

conf_matrix =confusion_matrix(predictions,y_test)
confusion_df = pd.DataFrame(conf_matrix, index=['Actual 0','Actual 1'],
                            columns=['Predicted 0','Predicted 1'])
sns.heatmap(confusion_df, cmap='coolwarm', annot=True)
```

Out[61]:
```
<AxesSubplot:>
```



In [62]:
```
from sklearn import metrics

print('\n**Classification Report:\n',
      metrics.classification_report(y_test,predictions))
```

```

**Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.53      0.64        15
           1       0.79      0.93      0.85        28

    accuracy                           0.79        43
   macro avg       0.79      0.73      0.75        43
weighted avg       0.79      0.79      0.78        43
```

In [63]:
```
# Deploy the Model

pred_new = my_model.predict([[67.00,91.00,1,58.00,2,0,55.0,1,64]])
pred_new
```

Out[63]:
```
array([1])
```

In [64]:
```
# Unknown Values

pred_new = my_model.predict([[89.40,69.00,2,80.00,1,0,64.0,1,64]])
pred_new
```

Out[64]:
```
array([1])
```

In [ ]:
```

```

In [65]:
```
# D) SVM
```

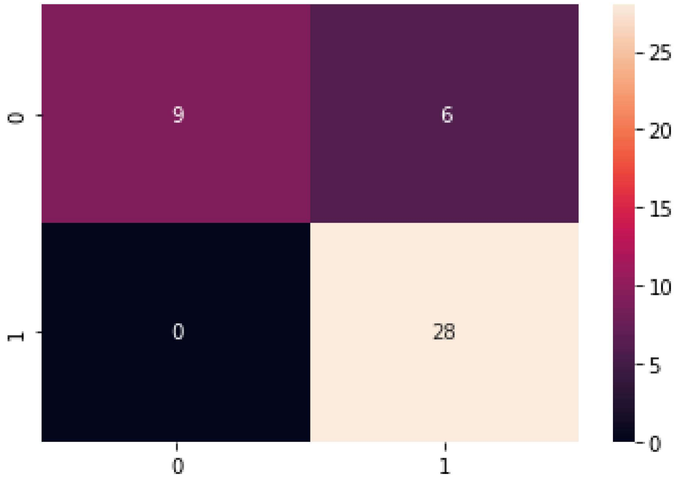
In [66]:
```
from sklearn.svm import SVC

my_model = SVC(kernel = 'rbf', random_state = 0)
result = my_model.fit(X_train, y_train)
```

In [67]:
```
predictions = result.predict(X_test)
predictions
```

Out[67]:
```
array([1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1])
```

In [68]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, predictions)

sns.heatmap(cm, annot = True, fmt = '2.0f')
```

Out[68]:
```
<AxesSubplot:>
```



In [69]:

```
from sklearn.metrics import accuracy_score

print('Accuracy using SVM',accuracy_score(y_test,predictions))
```

```
Accuracy using SVM 0.8604651162790697
```

In [70]:

```
from sklearn import metrics

print('\n**Classification Report:\n',
      metrics.classification_report(y_test,predictions))
```

```

**Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.60      0.75        15
           1       0.82      1.00      0.90        28

    accuracy                           0.86        43
   macro avg       0.91      0.80      0.83        43
weighted avg       0.89      0.86      0.85        43

```

In [71]:

```
# Deploy the Model

pred_new = my_model.predict([[67.00,91.00,1,58.00,2,0,55.0,1,64]])
pred_new
```

Out[71]:

```
array([0])
```

In [72]:

```
# Unknown Values

pred_new = my_model.predict([[89.40,69.00,2,80.00,1,0,64.0,1,64]])
pred_new
```

Out[72]:

```
array([0])
```

In [ ]:

In [73]:

```
# E) K Neighbor Classifer
```

In [74]:

```
from sklearn.neighbors import KNeighborsClassifier

my_model = KNeighborsClassifier(n_neighbors = 10)
result = my_model.fit(X_train, y_train)
```

In [75]:

```
predictions = result.predict(X_test)
predictions
```

Out[75]:

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

In [76]:

```
# now Measure our model's performance by using performace matrix

print('With KNN (K=10) accuracy is: ', result.score(X_test,y_test))
```

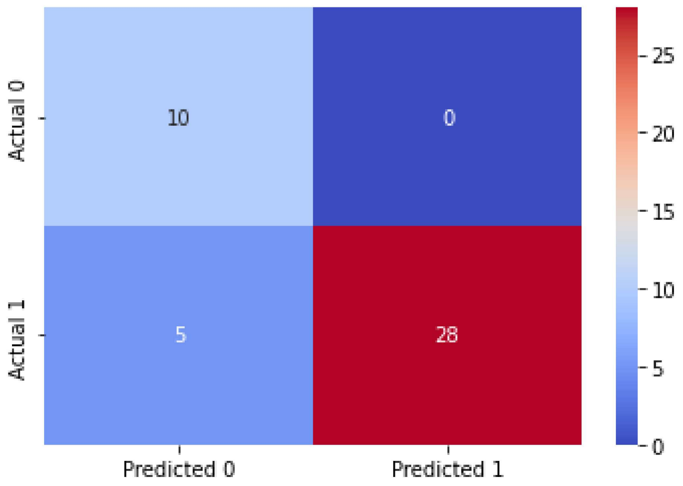
```
With KNN (K=10) accuracy is:  0.8837209302325582
```

In [77]:

```
from sklearn.metrics import confusion_matrix

conf_matrix =confusion_matrix(predictions,y_test)
confusion_df = pd.DataFrame(conf_matrix, index=['Actual 0','Actual 1'],
                            columns=['Predicted 0','Predicted 1'])
sns.heatmap(confusion_df, cmap='coolwarm', annot=True)
```

Out[77]:

```
<AxesSubplot:>
```



In [78]:

```
from sklearn import metrics

print('\n**Classification Report:\n',
      metrics.classification_report(y_test,predictions))
```

```

**Classification Report:
               precision    recall  f1-score   support

           0       1.00      0.67      0.80        15
           1       0.85      1.00      0.92        28

    accuracy                           0.88        43
   macro avg       0.92      0.83      0.86        43
weighted avg       0.90      0.88      0.88        43

```

In [79]:

```
# Deploy the Model

pred_new = my_model.predict([[67.00,91.00,1,58.00,2,0,55.0,1,64]])
pred_new
```

Out[79]:

```
array([1])
```

In [80]:

```
# Unknown Values

pred_new = my_model.predict([[89.40,69.00,2,80.00,1,0,64.0,1,64]])
pred_new
```

Out[80]:

```
array([1])
```

In [ ]:

```

```

In [81]:

```
# Result -

# Accuracy Using Logistic Regression  0.8837209302325582
# Accuracy Using Decision Tree Classifier  0.7674418604651163
# Accuracy using Random Forest Classifer  0.7906976744186046
# Accuracy using SVM 0.8604651162790697
# With KNN (K=10) accuracy is:  0.8837209302325582
```