1. Data Importing

In [295]:
```
import pandas as pd
import matplotlib.pylab as plt
import seaborn as sns
```

In [296]:
```
df=pd.read_csv('placement.csv')
```

2. Data Inspecting

In [297]:
```
df.head()
```

Out[297]:

	StudentID	CGPA	Internships	Projects	Workshops/Certifications	AptitudeTestScore	SoftSkillsRating	ExtracurricularActivities	PlacementTraining	SSC_Marks	HSC_Marks	PlacementStatus
0	1	7.5	1	1	1	65	4.4	No	No	61	79	NotPlaced
1	2	8.9	0	3	2	90	4.0	Yes	Yes	78	82	Placed
2	3	7.3	1	2	2	82	4.8	Yes	No	79	80	NotPlaced
3	4	7.5	1	1	2	85	4.4	Yes	Yes	81	80	Placed
4	5	8.3	1	2	2	86	4.5	Yes	Yes	74	88	Placed

In [298]:
```
df.tail()
```

Out[298]:

	StudentID	CGPA	Internships	Projects	Workshops/Certifications	AptitudeTestScore	SoftSkillsRating	ExtracurricularActivities	PlacementTraining	SSC_Marks	HSC_Marks	PlacementStatus
9995	9996	7.5	1	1	2	72	3.9	Yes	No	85	66	NotPlaced
9996	9997	7.4	0	1	0	90	4.8	No	No	84	67	Placed
9997	9998	8.4	1	3	0	70	4.8	Yes	Yes	79	81	Placed
9998	9999	8.9	0	3	2	87	4.8	Yes	Yes	71	85	Placed
9999	10000	8.4	0	1	1	66	3.8	No	No	62	66	NotPlaced

In [299]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   StudentID                  10000 non-null  int64
 1   CGPA                       10000 non-null  float64
 2   Internships                10000 non-null  int64
 3   Projects                   10000 non-null  int64
 4   Workshops/Certifications   10000 non-null  int64
 5   AptitudeTestScore          10000 non-null  int64
 6   SoftSkillsRating           10000 non-null  float64
 7   ExtracurricularActivities  10000 non-null  object
 8   PlacementTraining          10000 non-null  object
 9   SSC_Marks                  10000 non-null  int64
 10  HSC_Marks                  10000 non-null  int64
 11  PlacementStatus            10000 non-null  object
dtypes: float64(2), int64(7), object(3)
memory usage: 937.6+ KB
```

3. Data Preprocessing

In [300]:
```
df.columns
```

Out[300]:
```
Index(['StudentID', 'CGPA', 'Internships', 'Projects',
       'Workshops/Certifications', 'AptitudeTestScore', 'SoftSkillsRating',
       'ExtracurricularActivities', 'PlacementTraining', 'SSC_Marks',
       'HSC_Marks', 'PlacementStatus'],
      dtype='object')
```

3.1 Removing StudentID column:

In [301]:
```
df=df.drop(columns='StudentID')
df
```

Out[301]:

	CGPA	Internships	Projects	Workshops/Certifications	AptitudeTestScore	SoftSkillsRating	ExtracurricularActivities	PlacementTraining	SSC_Marks	HSC_Marks	PlacementStatus
0	7.5	1	1	1	65	4.4	No	No	61	79	NotPlaced
1	8.9	0	3	2	90	4.0	Yes	Yes	78	82	Placed
2	7.3	1	2	2	82	4.8	Yes	No	79	80	NotPlaced
3	7.5	1	1	2	85	4.4	Yes	Yes	81	80	Placed
4	8.3	1	2	2	86	4.5	Yes	Yes	74	88	Placed
...	...	...	...	...	...	...	...	...	...	...	...
9995	7.5	1	1	2	72	3.9	Yes	No	85	66	NotPlaced
9996	7.4	0	1	0	90	4.8	No	No	84	67	Placed
9997	8.4	1	3	0	70	4.8	Yes	Yes	79	81	Placed
9998	8.9	0	3	2	87	4.8	Yes	Yes	71	85	Placed
9999	8.4	0	1	1	66	3.8	No	No	62	66	NotPlaced

10000 rows × 11 columns

3.2 Check for null values

In [302]:
```
df.isnull().sum()
```

Out[302]:
```
CGPA                         0
Internships                  0
Projects                     0
Workshops/Certifications     0
AptitudeTestScore            0
SoftSkillsRating             0
ExtracurricularActivities    0
PlacementTraining            0
SSC_Marks                    0
HSC_Marks                    0
PlacementStatus              0
dtype: int64
```

3.3 Check for duplicate values

In [303]:
```
df.duplicated().sum()
```

Out[303]: 72

3.4 Check for outliers

In [304]:
```
df.describe()
```

Out[304]:

	CGPA	Internships	Projects	Workshops/Certifications	AptitudeTestScore	SoftSkillsRating	SSC_Marks	HSC_Marks
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	7.698010	1.049200	2.026600	1.013200	79.449900	4.323960	69.159400	74.501500
std	0.640131	0.665901	0.867968	0.904272	8.159997	0.411622	10.430459	8.919527
min	6.500000	0.000000	0.000000	0.000000	60.000000	3.000000	55.000000	57.000000
25%	7.400000	1.000000	1.000000	0.000000	73.000000	4.000000	59.000000	67.000000
50%	7.700000	1.000000	2.000000	1.000000	80.000000	4.400000	70.000000	73.000000
75%	8.200000	1.000000	3.000000	2.000000	87.000000	4.700000	78.000000	83.000000
max	9.100000	2.000000	3.000000	3.000000	90.000000	4.800000	90.000000	88.000000

4. Exploratory Data Analysis

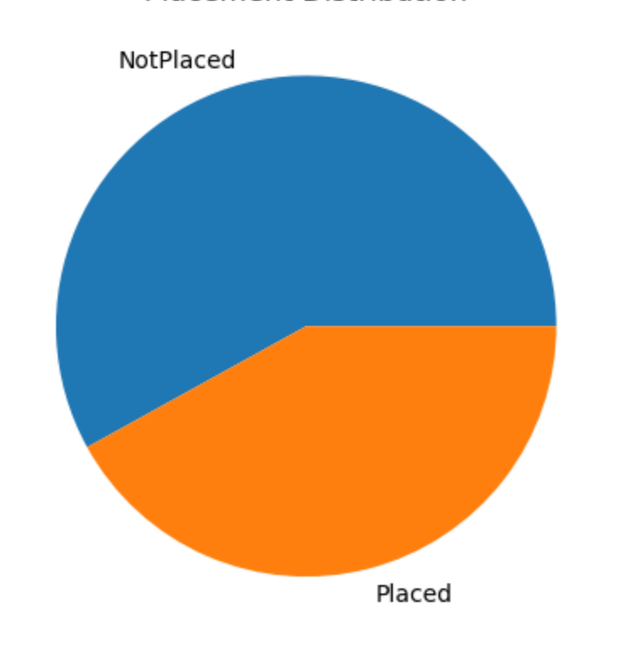
4.1 Placed vs Unplaced

In [305]:
```
df['PlacementStatus'].value_counts()
```

Out[305]:
```
PlacementStatus
NotPlaced    5803
Placed       4197
Name: count, dtype: int64
```

In [306]:
```
df.groupby('PlacementStatus').size().plot(kind='pie')
plt.title('Placement Distribution')
```

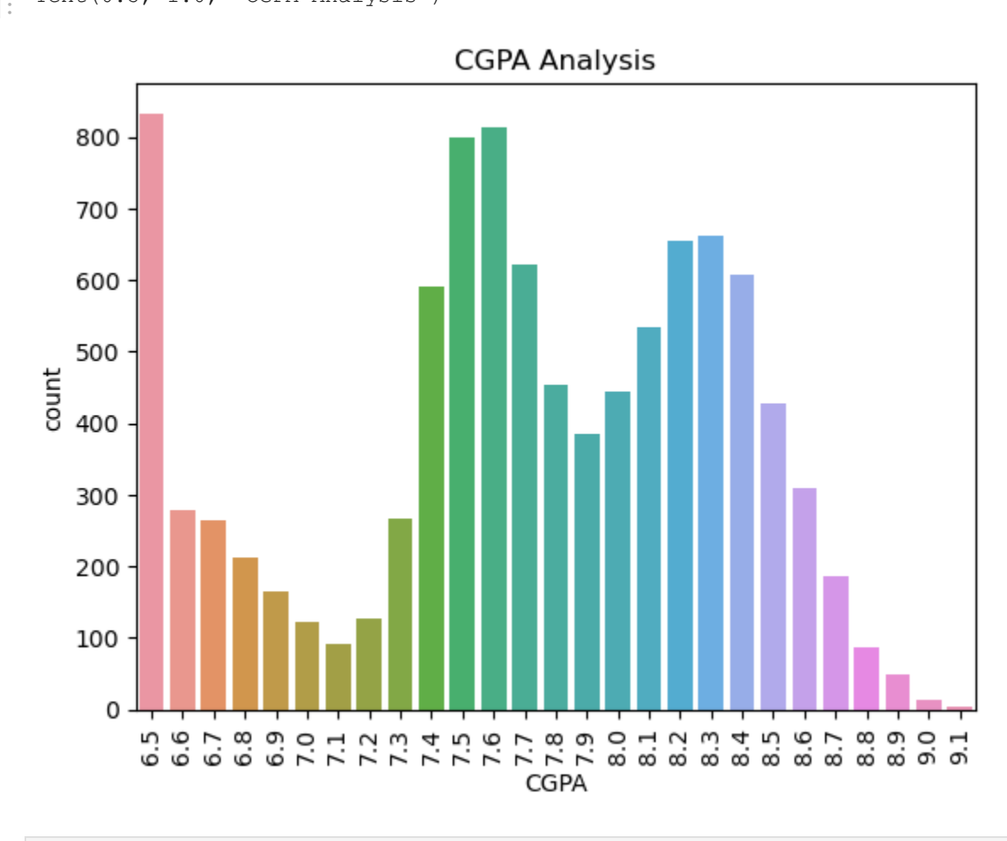
Out[306]: Text(0.5, 1.0, 'Placement Distribution')



4.2 Impact of cgpa over placement

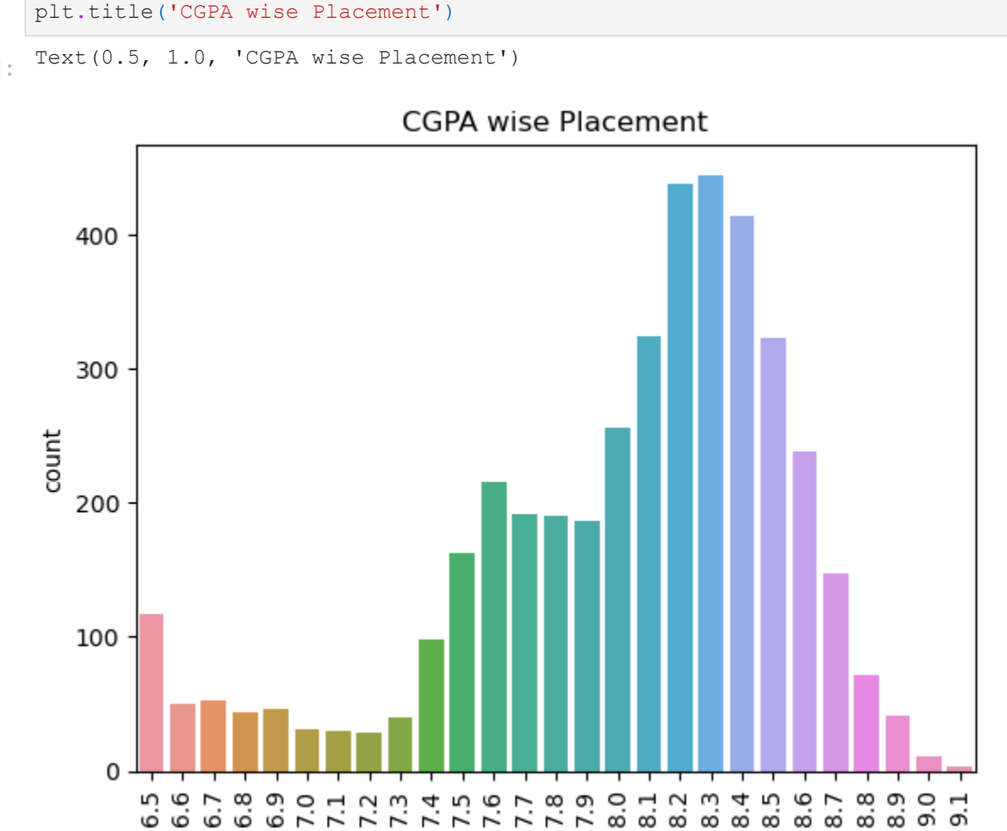
In [307]:
```
sns.countplot(data=df,x='CGPA')
plt.xticks(rotation=90)
plt.title('CGPA Analysis')
```

Out[307]: Text(0.5, 1.0, 'CGPA Analysis')



In [310]:
```
sns.countplot(data=df.loc[(df.PlacementStatus=='Placed')],x='CGPA')
plt.xticks(rotation=90)
plt.title('CGPA wise Placement')
```

Out[310]: Text(0.5, 1.0, 'CGPA wise Placement')



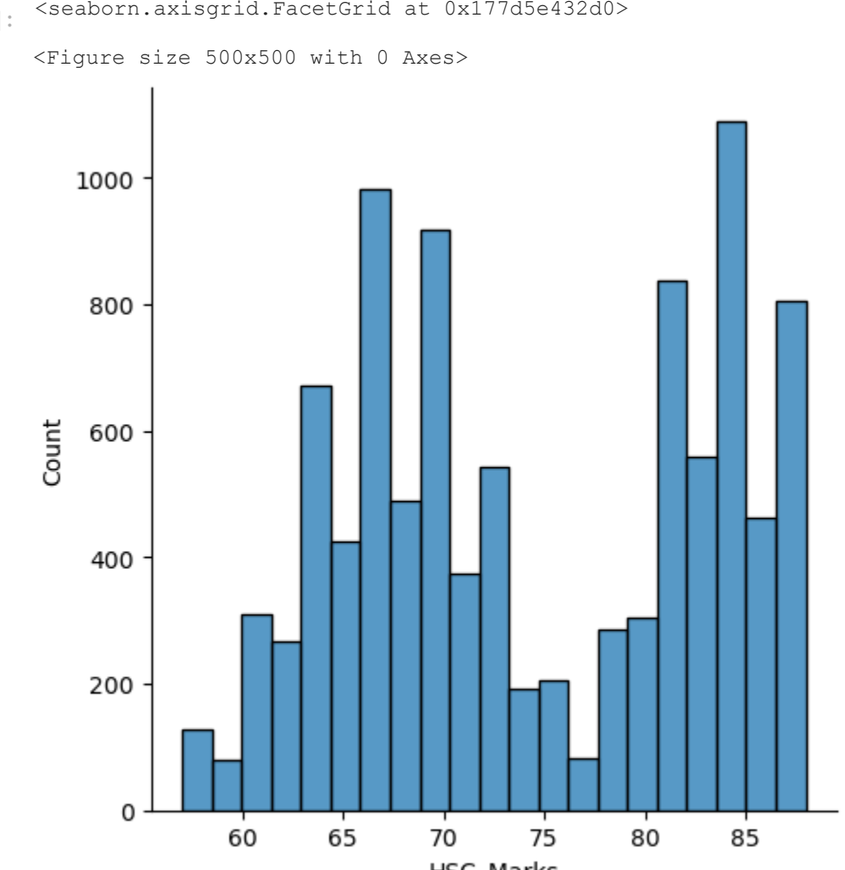
4.3 Impact of HSC_Marks over placement

In [311]:
```
plt.figure(figsize=(5,5))
sns.displot(df['HSC_Marks'])
```

```
C:\Users\DELL\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

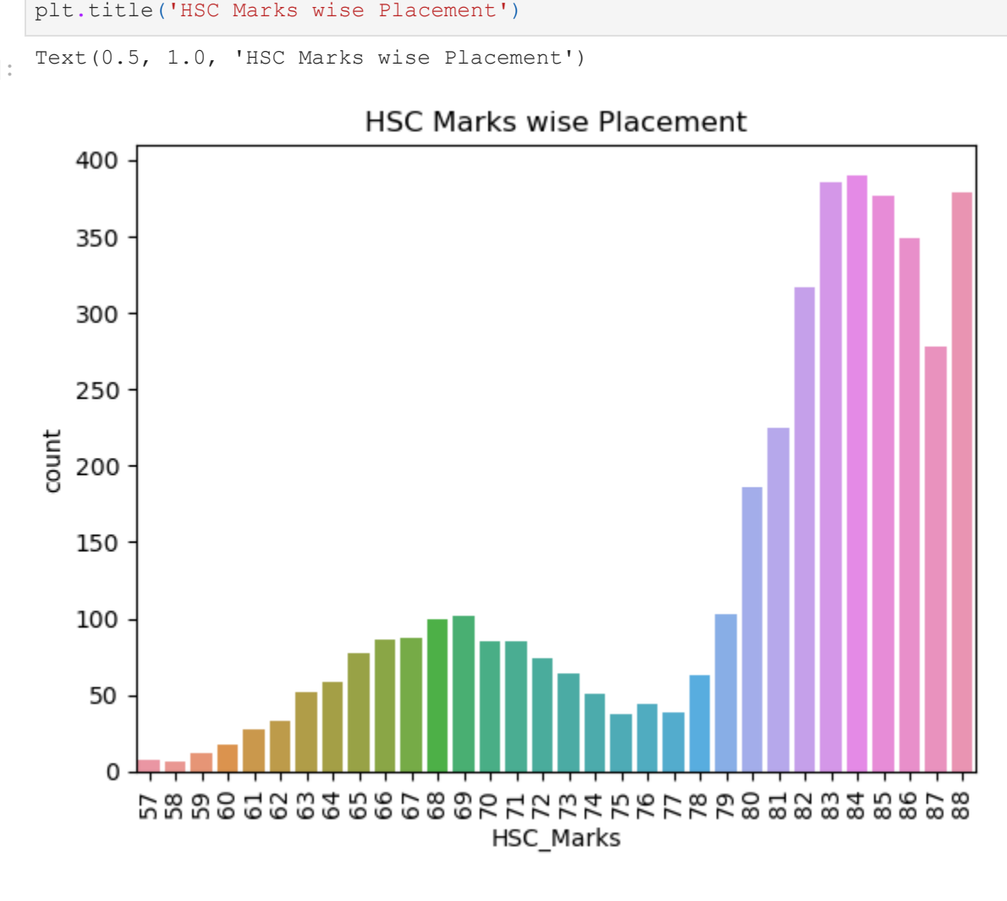
Out[311]: <seaborn.axisgrid.FacetGrid at 0x177d5e432d0>

<Figure size 500x500 with 0 Axes>



In [313]:
```
sns.countplot(data=df.loc[(df.PlacementStatus=='Placed')],x='HSC_Marks')
plt.xticks(rotation=90)
plt.title('HSC Marks wise Placement')
```

Out[313]: Text(0.5, 1.0, 'HSC Marks wise Placement')

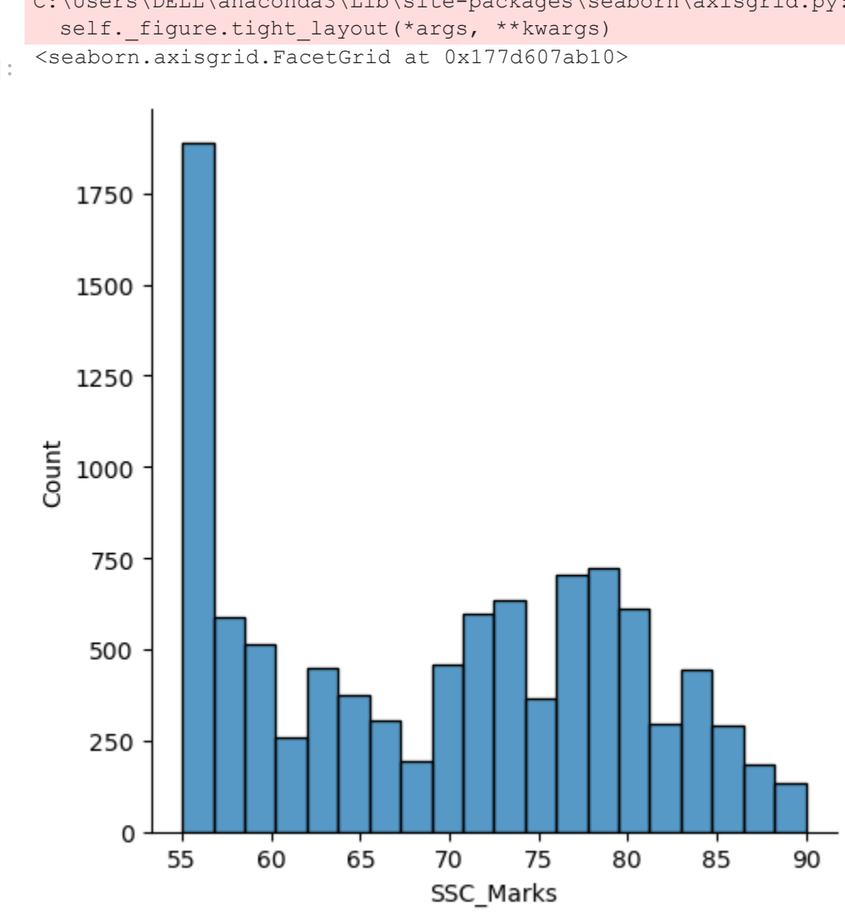


4.4 Impact of SSC_Marks over placement

In [314]:
```
sns.displot(df['SSC_Marks'])
```

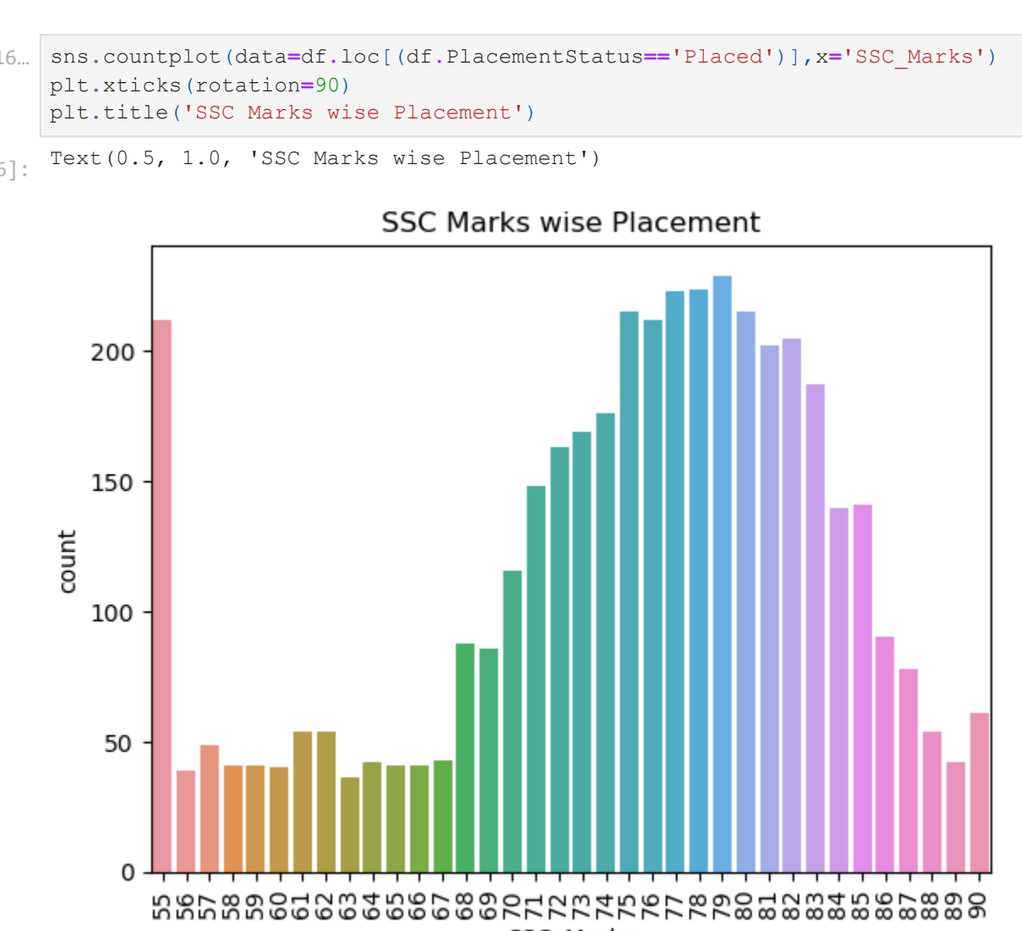
```
C:\Users\DELL\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[314]: <seaborn.axisgrid.FacetGrid at 0x177d607ab10>



In [316]:
```
sns.countplot(data=df.loc[(df.PlacementStatus=='Placed')],x='SSC_Marks')
plt.xticks(rotation=90)
plt.title('SSC Marks wise Placement')
```

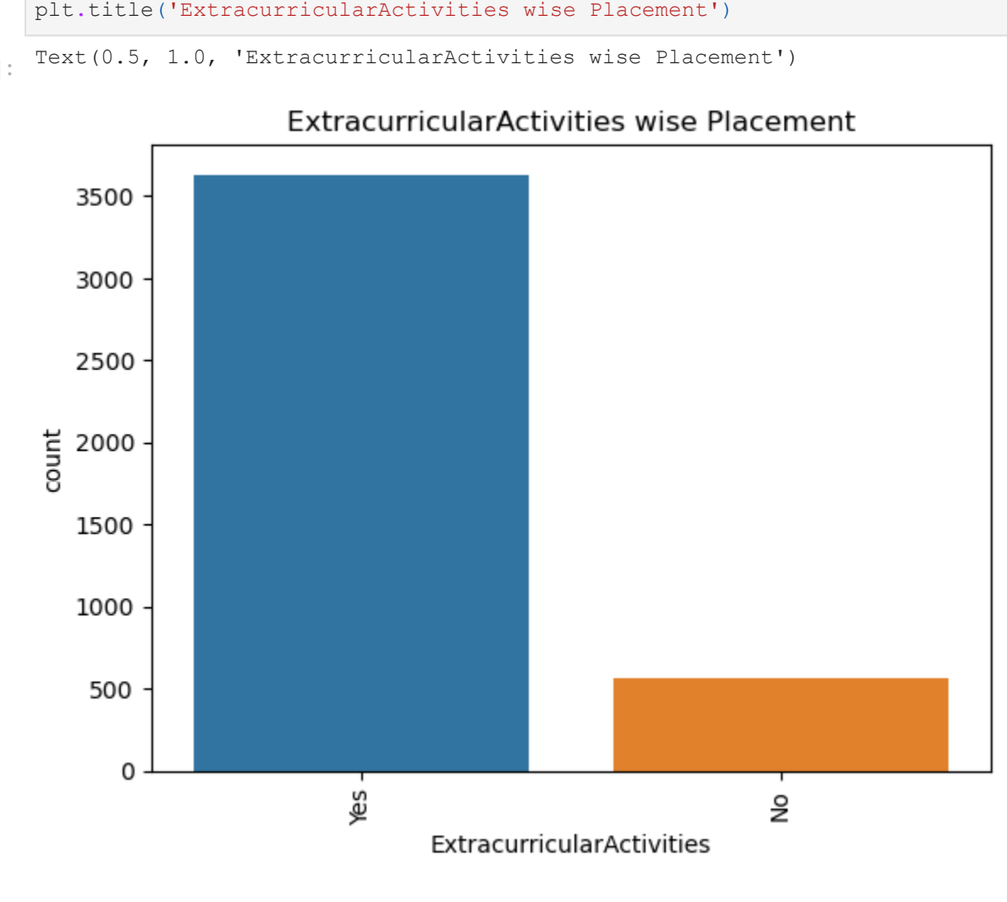
Out[316]: Text(0.5, 1.0, 'SSC Marks wise Placement')



4.5 Impact of ExtracurricularActivities over placement

In [320]:
```
sns.countplot(data=df.loc[(df.PlacementStatus=='Placed')],x='ExtracurricularActivities')
plt.xticks(rotation=90)
plt.title('ExtracurricularActivities wise Placement')
```

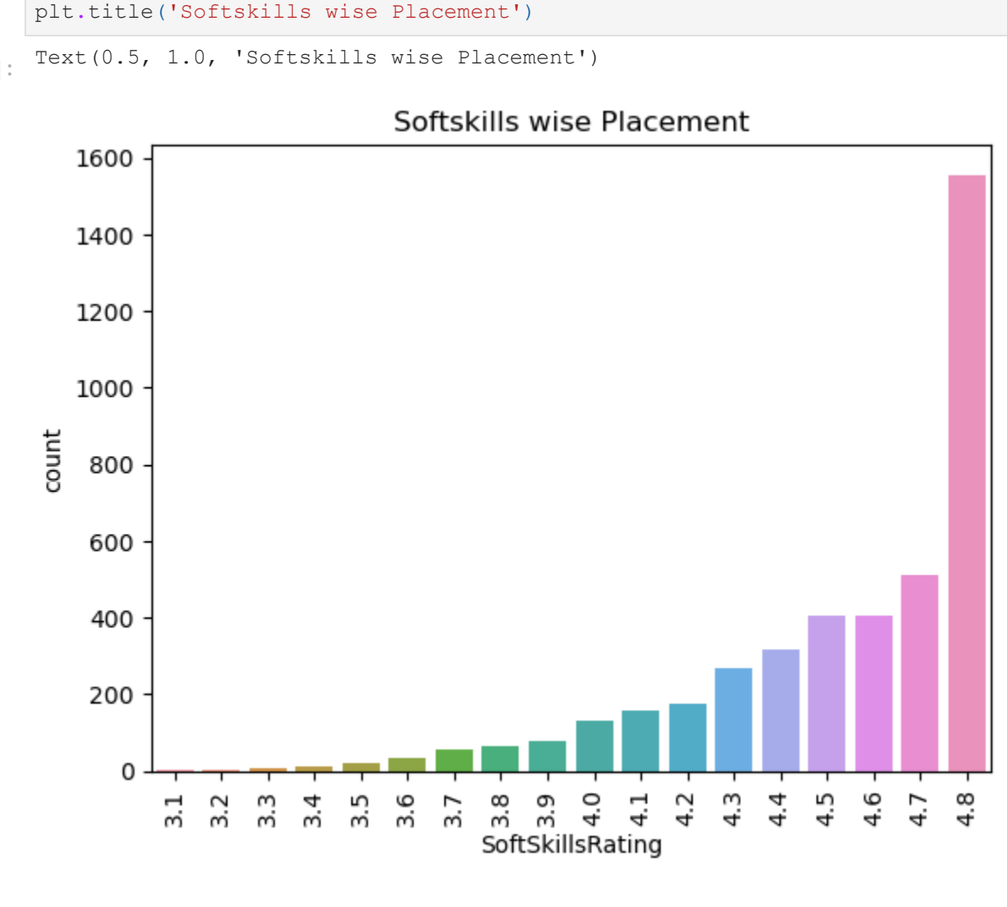
Out[320]: Text(0.5, 1.0, 'ExtracurricularActivities wise Placement')



4.6 Impact of Softskills over placement

In [322]:
```
sns.countplot(data=df.loc[(df.PlacementStatus=='Placed')],x='SoftSkillsRating')
plt.xticks(rotation=90)
plt.title('Softskills wise Placement')
```

Out[322]: Text(0.5, 1.0, 'Softskills wise Placement')



4.7 Impact of Projects over placement

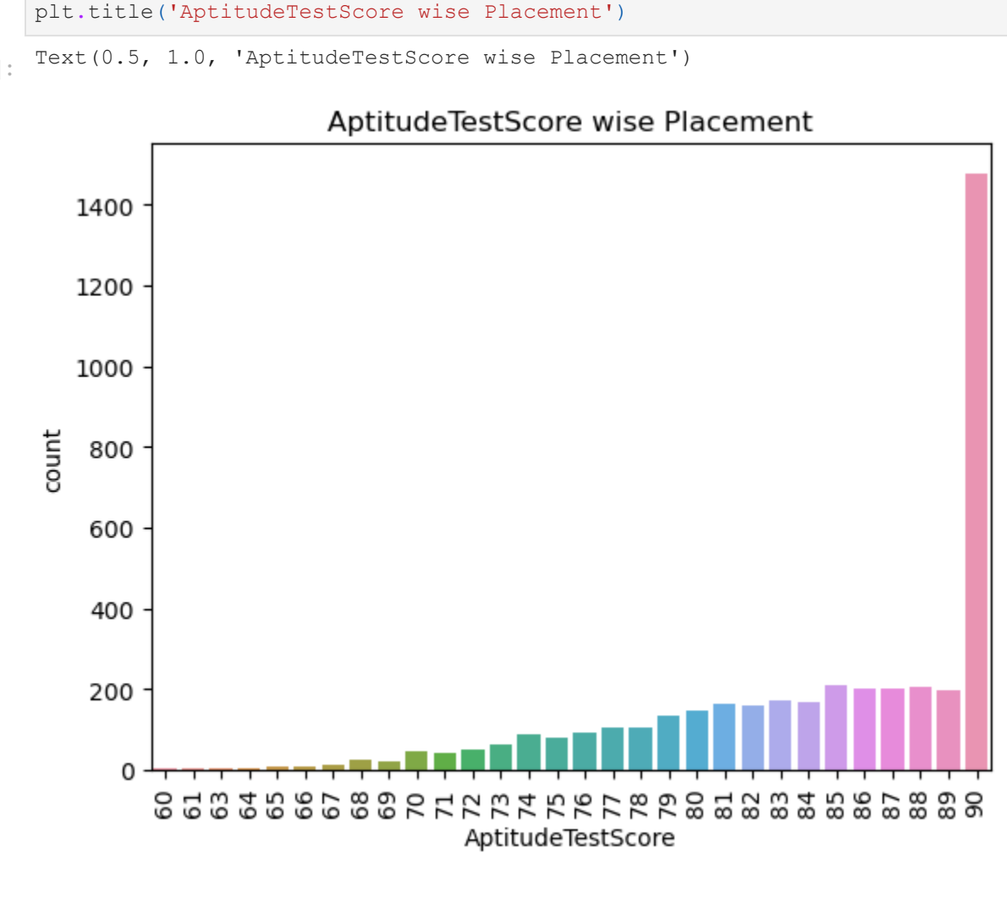
In [324]:
```
sns.countplot(data=df.loc[(df.PlacementStatus=='Placed')],x='Projects')
plt.xticks(rotation=90)
plt.title('Projects wise Placement')
```

Out[324]: Text(0.5, 1.0, 'Projects wise Placement')

4.8 Impact of AptitudeTestScore over placements

In [326]:
```
sns.countplot(data=df.loc[(df.PlacementStatus=='Placed')],x='AptitudeTestScore')
plt.xticks(rotation=90)
plt.title('AptitudeTestScore wise Placement')
```

Out[326]: Text(0.5, 1.0, 'AptitudeTestScore wise Placement')



5. Data Encoding

5.1 Label_Encoding

In [327]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
object_cols = df.select_dtypes(include=['object']).columns
for column in object_cols:
    df[column]=labelencoder.fit_transform(df[column])
```

In [328]:
```
df.head()
```

Out[328]:

	CGPA	Internships	Projects	Workshops/Certifications	AptitudeTestScore	SoftSkillsRating	ExtracurricularActivities	PlacementTraining	SSC_Marks	HSC_Marks	PlacementStatus
0	7.5	1	1	1	65	4.4	0	0	61	79	0
1	8.9	0	3	2	90	4.0	1	1	78	82	1
2	7.3	1	2	2	82	4.8	1	0	79	80	0
3	7.5	1	1	2	85	4.4	1	1	81	80	1
4	8.3	1	2	2	86	4.5	1	1	74	88	1

5.2 Correlation Matrix

In [329]:
```
correlation_matrix=df.corr()
plt.figure(figsize=(12,6))
sns.heatmap(correlation_matrix,annot=True,cmap='coolwarm')
```

Out[329]: <Axes: >

6. Data Splitting

In [330]:
```
X=df.drop(columns='PlacementStatus')
y=df.PlacementStatus
```

In [331]:
```
from sklearn.model_selection import train_test_split
```

In [332]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)
print('X_train shape is ' , X_train.shape)
print('X_test shape is ' , X_test.shape)
print('y_train shape is ' , y_train.shape)
print('y_test shape is ' , y_test.shape)
```

```
X_train shape is  (7500, 10)
X_test shape is  (2500, 10)
y_train shape is  (7500,)
y_test shape is  (2500,)
```

7.Model Training and Evaluation

7.1 Logistic Regression

```
--> Accuracy Score
-->Classification Report
-->Confusion Matrix
```

In [333]:
```
from sklearn.linear_model import LogisticRegression
```

In [334]:
```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,ConfusionMatrixDisplay
```

In [335]:
```
lr=LogisticRegression(max_iter=50000,penalty=None)
lr.fit(X_train,y_train)
prediction=lr.predict(X_test)
score1=accuracy_score(prediction,y_test)
```

In [336]:
```
print('Accuracy Score is:',score1)
```

Accuracy Score is: 0.7932

In [337]:
```
target_names=['NotPlaced','Placed']
```

In [338]:
```
print('Classification Report is : \n',classification_report(y_test,prediction,target_names=target_names))
```

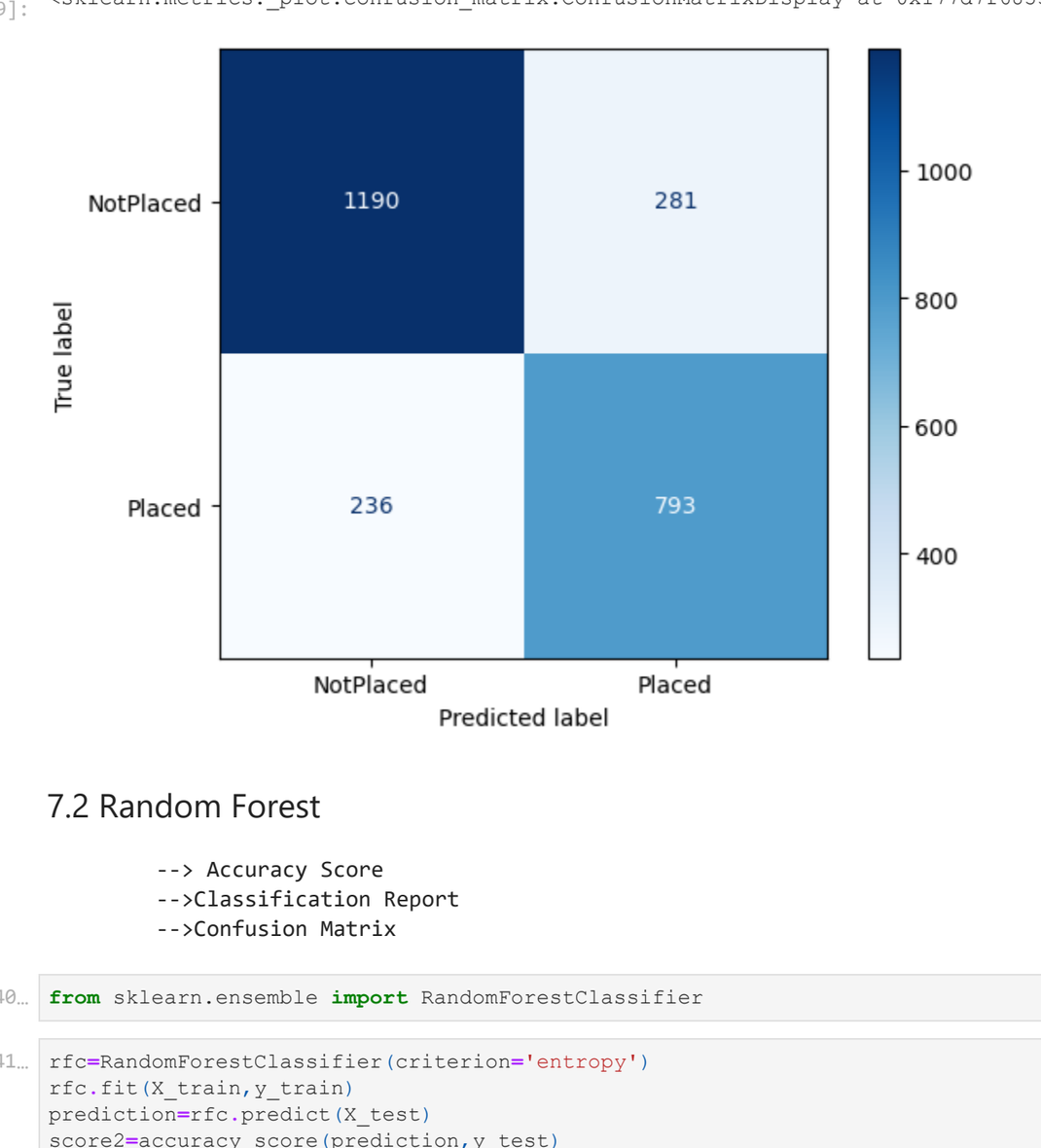
```
Classification Report is :
               precision    recall  f1-score   support

   NotPlaced       0.83      0.81      0.82      1471
      Placed       0.74      0.77      0.75      1029

    accuracy                           0.79      2500
   macro avg       0.79      0.79      0.79      2500
weighted avg       0.79      0.79      0.79      2500
```

In [339]:
```
cm=confusion_matrix(y_test,prediction)
ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=target_names).plot(cmap='Blues')
```

Out[339]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x177d7f68550>



7.2 Random Forest

```
--> Accuracy Score
-->Classification Report
-->Confusion Matrix
```

In [340]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [341]:
```
rfc=RandomForestClassifier(criterion='entropy')
rfc.fit(X_train,y_train)
prediction=rfc.predict(X_test)
score2=accuracy_score(prediction,y_test)
```

In [342]:
```
print('Accuracy Score is:',score2)
```

Accuracy Score is: 0.7796

In [343]:
```
target_names=['NotPlaced','Placed']
print('Classification Report is : \n',classification_report(y_test,prediction,target_names=target_names))
```

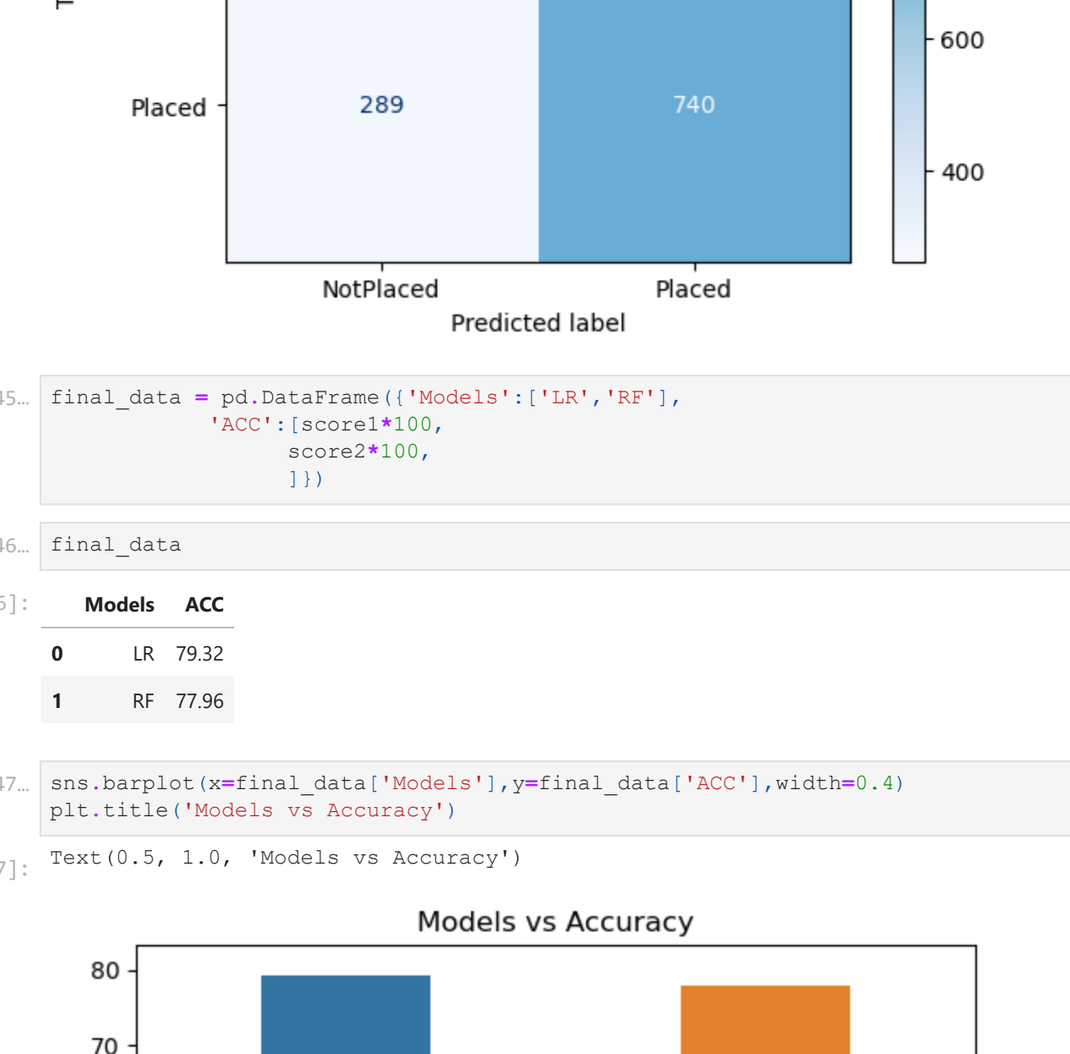
```
Classification Report is :
               precision    recall  f1-score   support

   NotPlaced       0.81      0.82      0.81      1471
      Placed       0.74      0.72      0.73      1029

    accuracy                           0.78      2500
   macro avg       0.77      0.77      0.77      2500
weighted avg       0.78      0.78      0.78      2500
```

In [344]:
```
cm=confusion_matrix(y_test,prediction)
ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=target_names).plot(cmap='Blues')
```

Out[344]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x177d84ea750>



In [345]:
```
final_data = pd.DataFrame({'Models':['LR','RF'],
             'ACC':[score1*100,
                 score2*100,
                 ]})
```

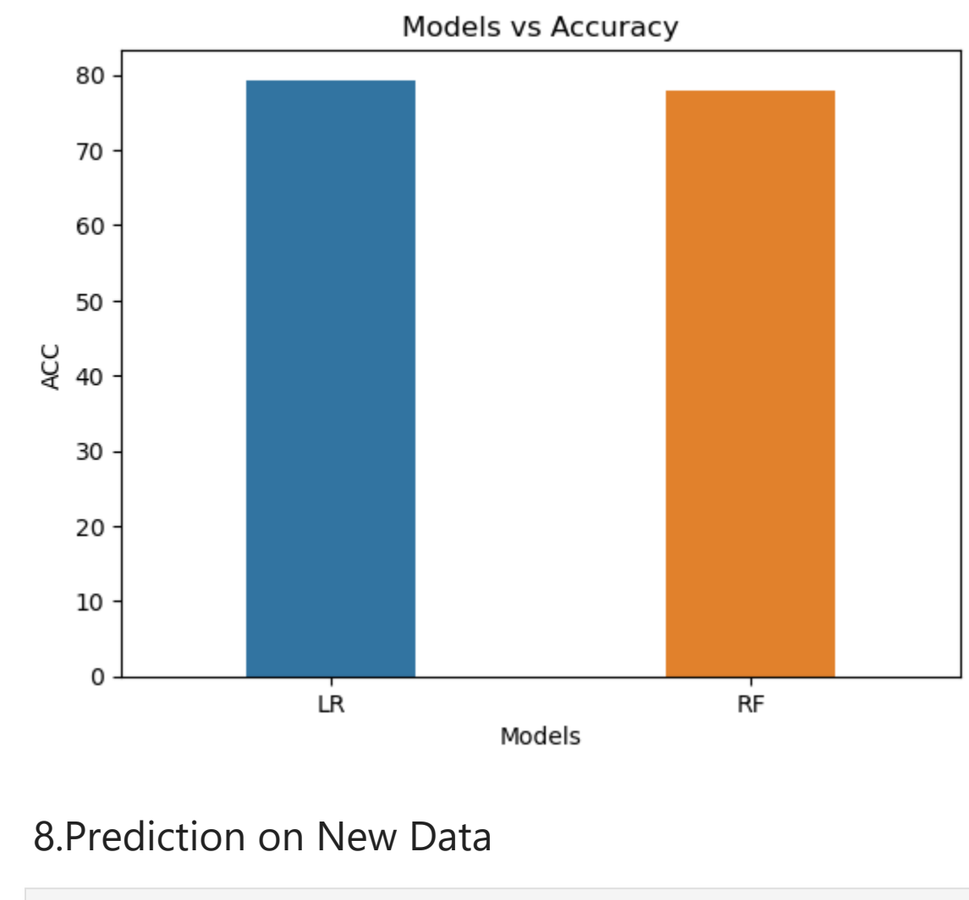
In [346]:
```
final_data
```

Out[346]:

	Models	ACC
0	LR	79.32
1	RF	77.96

In [347]:
```
sns.barplot(x=final_data['Models'],y=final_data['ACC'],width=0.4)
plt.title('Models vs Accuracy')
```

Out[347]: Text(0.5, 1.0, 'Models vs Accuracy')



8.Prediction on New Data

In [348]:
```
new_data = pd.DataFrame({
    'CGPA':8.8,
    'Internships':3,
    'Projects':2,
    'Workshops/Certifications':7,
    'AptitudeTestScore':96,
    'SoftSkillsRating':4.5,
    'ExtracurricularActivities':1,
    'PlacementTraining':1,
    'SSC_Marks':89,
    'HSC_Marks':79,
},index=[0])
```

In [353]:
```
lr= LogisticRegression()
lr.fit(X,y)
```

Out[353]:
```
▾ LogisticRegression
LogisticRegression()
```

In [359]:
```
p=lr.predict(new_data)
prob=lr.predict_proba(new_data)
if p==1:
    print('Placed')
    print(f"You will be placed with probability of {prob[0][1]:.2f}")
else:
    print("You wont get placed,keep working hard")
```

```
Placed
You will be placed with probability of 0.96
```