In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
data = pd.read_csv('admission_data.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

In [4]:

```
data.tail()
```

Out[4]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
495	332	108	5	4.5	4.0	9.02	1	0.87
496	337	117	5	5.0	5.0	9.87	1	0.96
497	330	120	5	4.5	5.0	9.56	1	0.93
498	312	103	4	4.0	5.0	8.43	0	0.73
499	327	113	4	4.5	4.5	9.04	0	0.84

In [5]:

```
1 data.shape
```

Out[5]:

```
(500, 8)
```

In [6]:

```
1 data.columns
```

Out[6]:

```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CG
PA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

In [7]:

```
1 data.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

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

Out[8]:

```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [9]:

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   GRE Score          500 non-null    int64  
 1   TOEFL Score        500 non-null    int64  
 2   University Rating  500 non-null    int64  
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64  
 7   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 31.4 KB
```

In [10]:

```
1 data.describe()
```

Out[10]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Cl
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	50
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	

In [11]:

```
1 data.nunique()
```

Out[11]:

```
GRE Score            49
TOEFL Score          29
University Rating     5
SOP                   9
LOR                   9
CGPA                184
Research              2
Chance of Admit      61
dtype: int64
```

In [12]:

```
1 data['University Rating'].unique()
```

Out[12]:

```
array([4, 3, 2, 5, 1], dtype=int64)
```

In [13]:

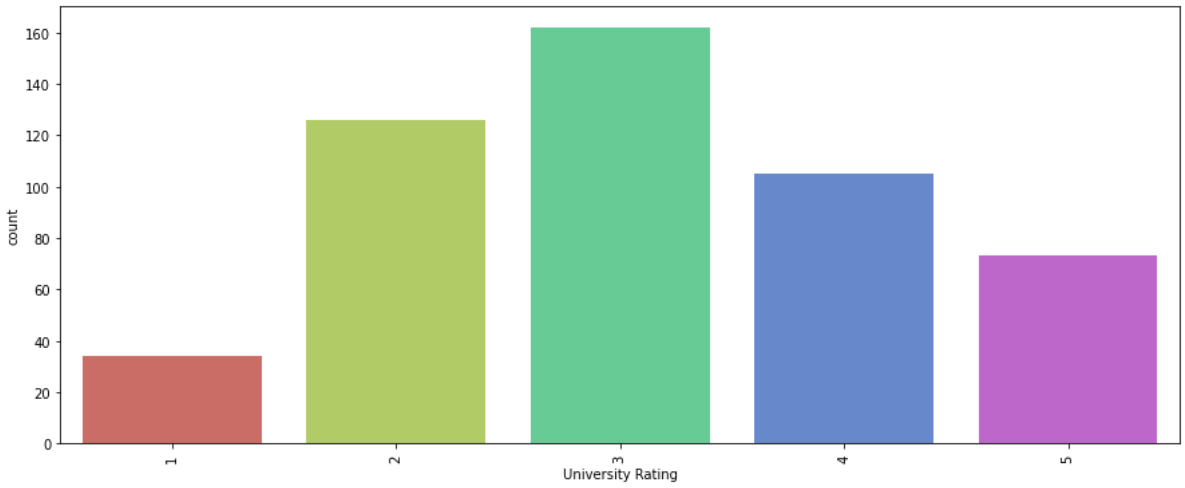
```
1 data['University Rating'].value_counts()
```

Out[13]:

```
3    162
2    126
4    105
5     73
1     34
Name: University Rating, dtype: int64
```

In [14]:

```
plt.figure(figsize=(15,6))
sns.countplot('University Rating', data = data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [15]:

```
data['SOP'].unique()
```

Out[15]:

```
array([4.5, 4. , 3. , 3.5, 2. , 5. , 1.5, 1. , 2.5])
```

In [16]:

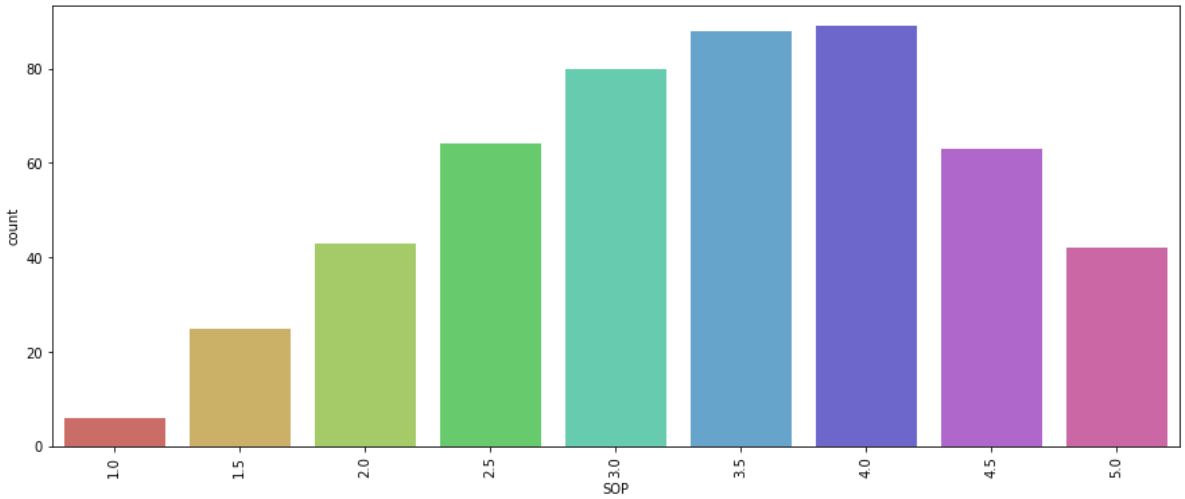
```
data['SOP'].value_counts()
```

Out[16]:

```
4.0    89
3.5    88
3.0    80
2.5    64
4.5    63
2.0    43
5.0    42
1.5    25
1.0     6
Name: SOP, dtype: int64
```

In [17]:

```
plt.figure(figsize=(15,6))
sns.countplot('SOP', data = data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



In [18]:

```
data['LOR '].unique()
```

Out[18]:

```
array([4.5, 3.5, 2.5, 3. , 4. , 1.5, 2. , 5. , 1. ])
```

In [19]:

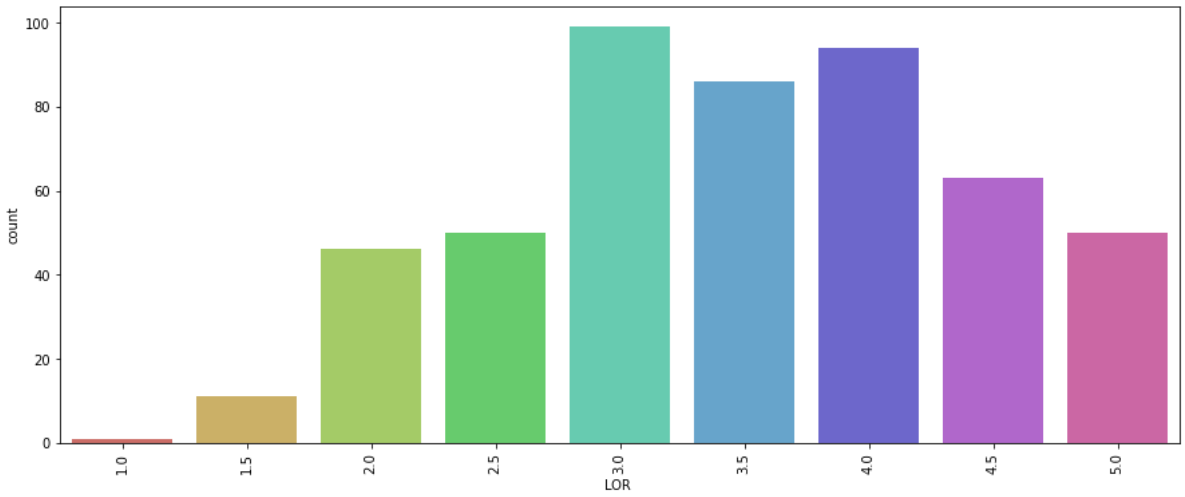
```
1 data['LOR '].value_counts()
```

Out[19]:

```
3.0    99
4.0    94
3.5    86
4.5    63
2.5    50
5.0    50
2.0    46
1.5    11
1.0     1
Name: LOR , dtype: int64
```

In [20]:

```
1 plt.figure(figsize=(15,6))
2 sns.countplot('LOR ', data = data, palette = 'hls')
3 plt.xticks(rotation = 90)
4 plt.show()
```



In [21]:

```
1 data['Research'].unique()
```

Out[21]:

```
array([1, 0], dtype=int64)
```

In [22]:

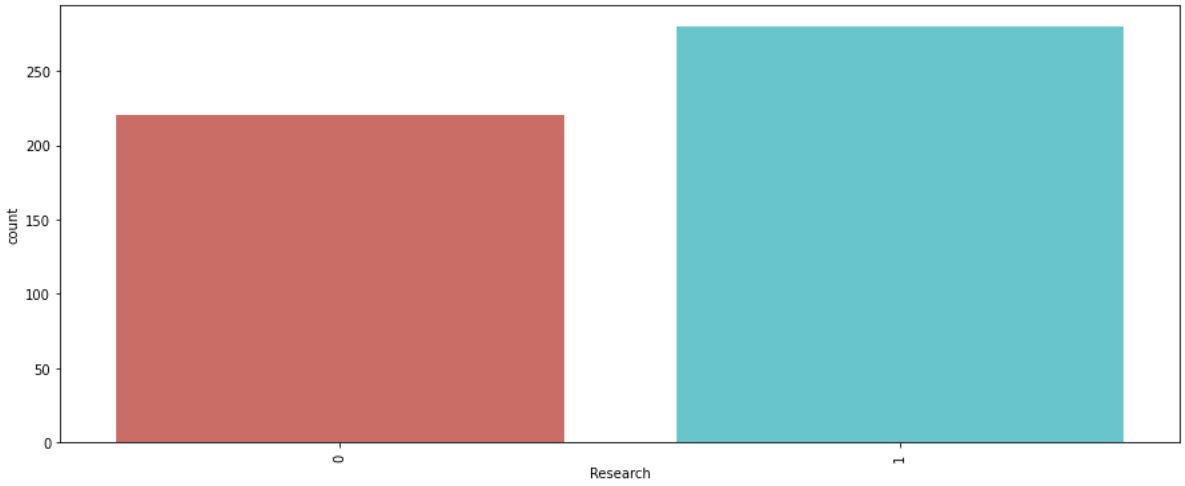
```
1 data['Research'].value_counts()
```

Out[22]:

```
1    280
0    220
Name: Research, dtype: int64
```

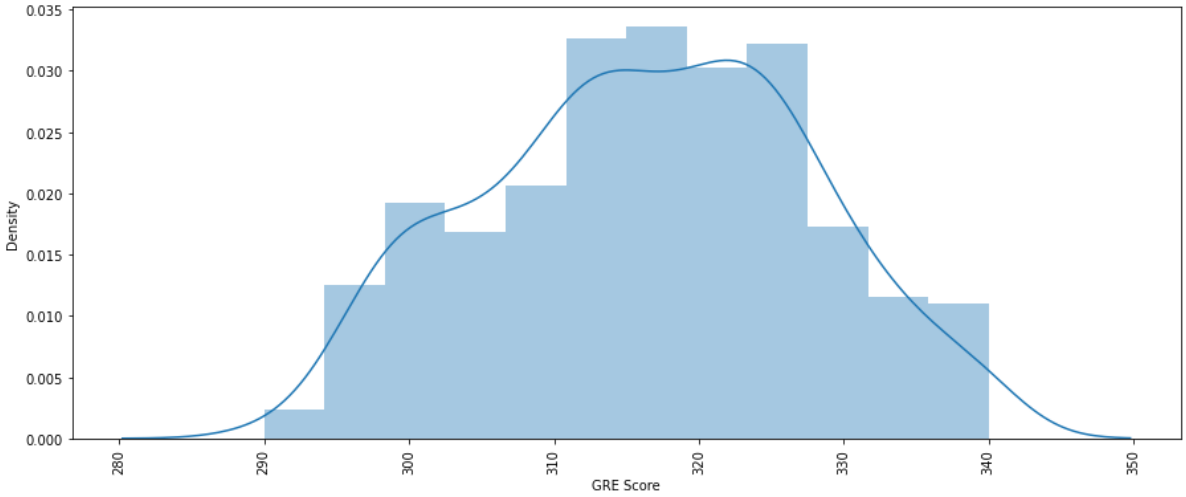
In [23]:

```
plt.figure(figsize=(15,6))
sns.countplot('Research', data = data, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```



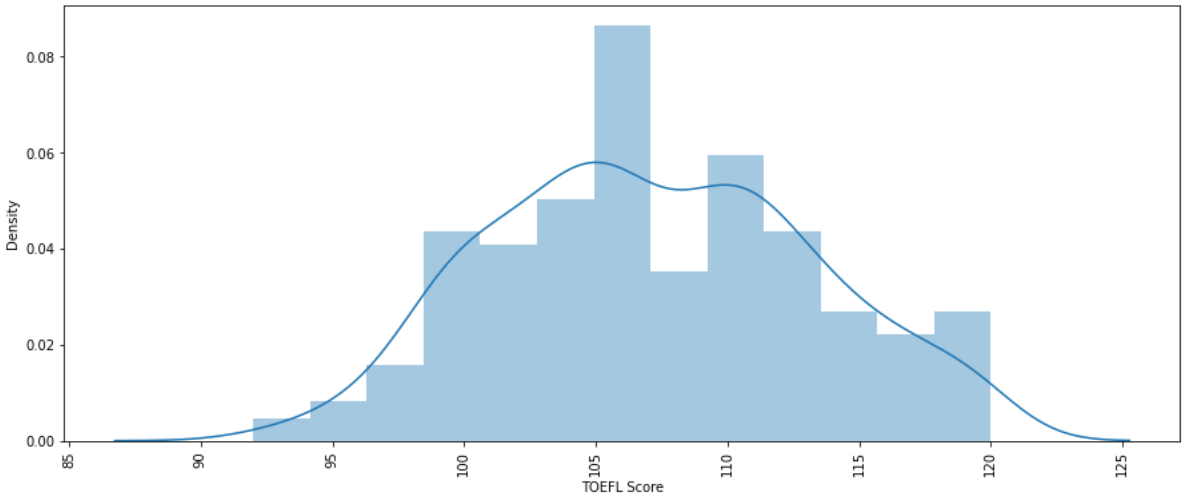
In [24]:

```
plt.figure(figsize=(15,6))
sns.distplot(data['GRE Score'])
plt.xticks(rotation = 90)
plt.show()
```



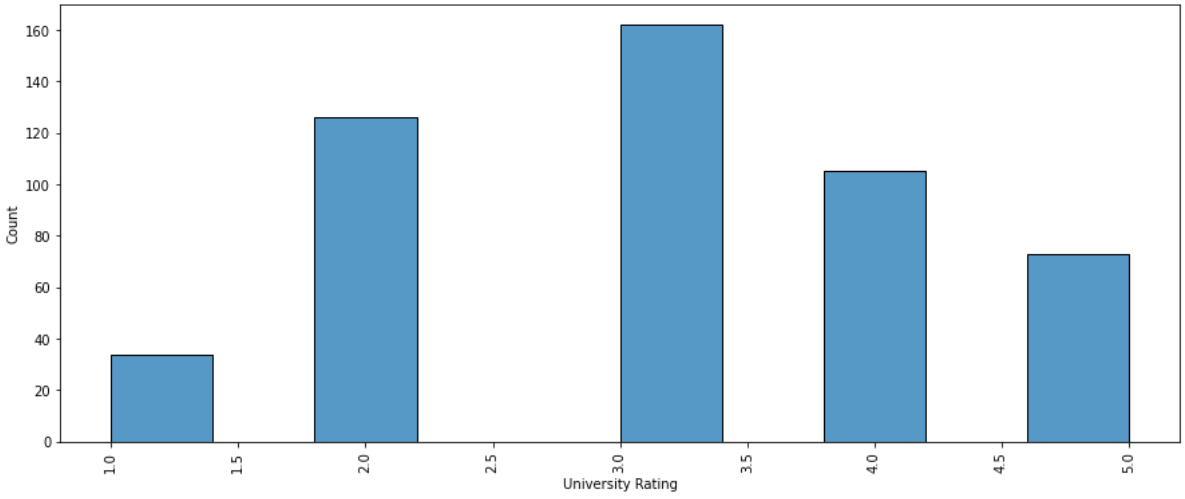
In [25]:

```
plt.figure(figsize=(15,6))
sns.distplot(data['TOEFL Score'])
plt.xticks(rotation = 90)
plt.show()
```



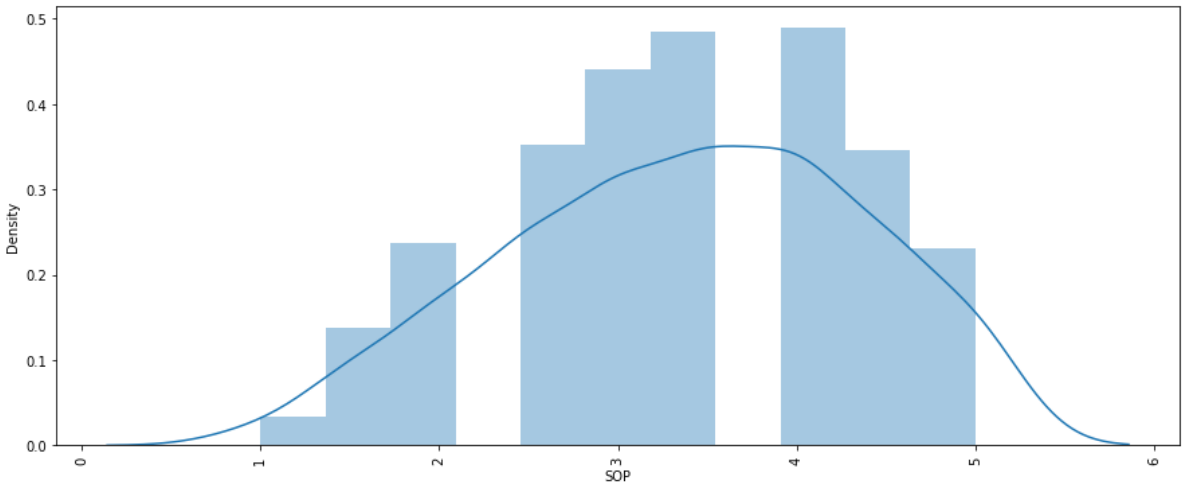
In [26]:

```
plt.figure(figsize=(15,6))
sns.histplot(data['University Rating'])
plt.xticks(rotation = 90)
plt.show()
```



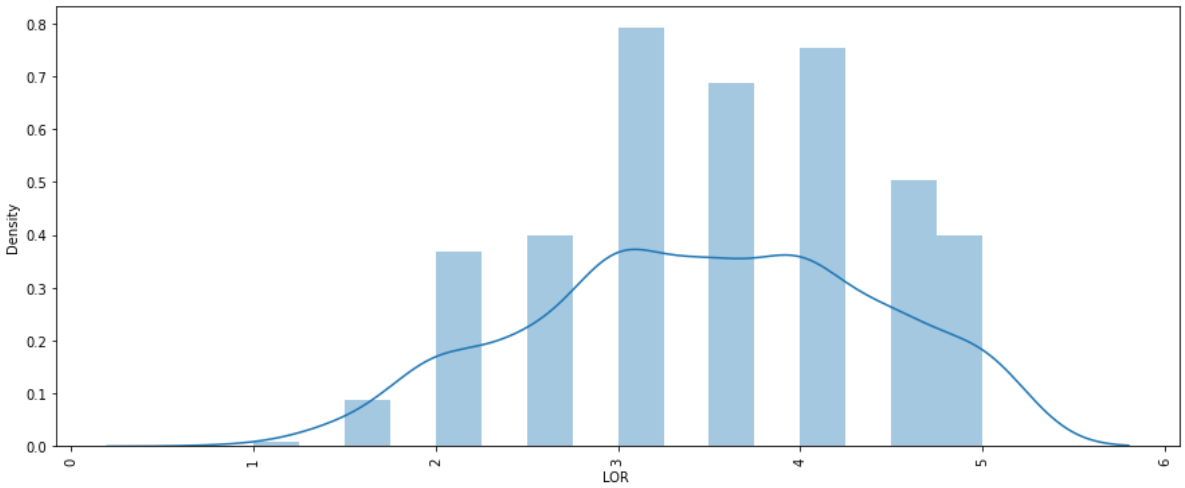
In [27]:

```
plt.figure(figsize=(15,6))
sns.distplot(data['SOP'])
plt.xticks(rotation = 90)
plt.show()
```



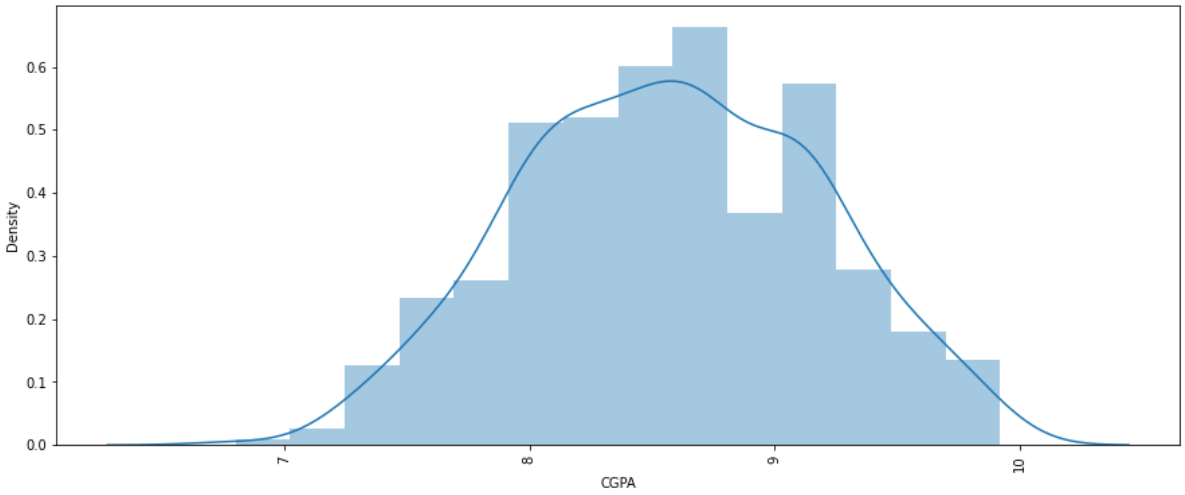
In [28]:

```
plt.figure(figsize=(15,6))
sns.distplot(data['LOR '])
plt.xticks(rotation = 90)
plt.show()
```



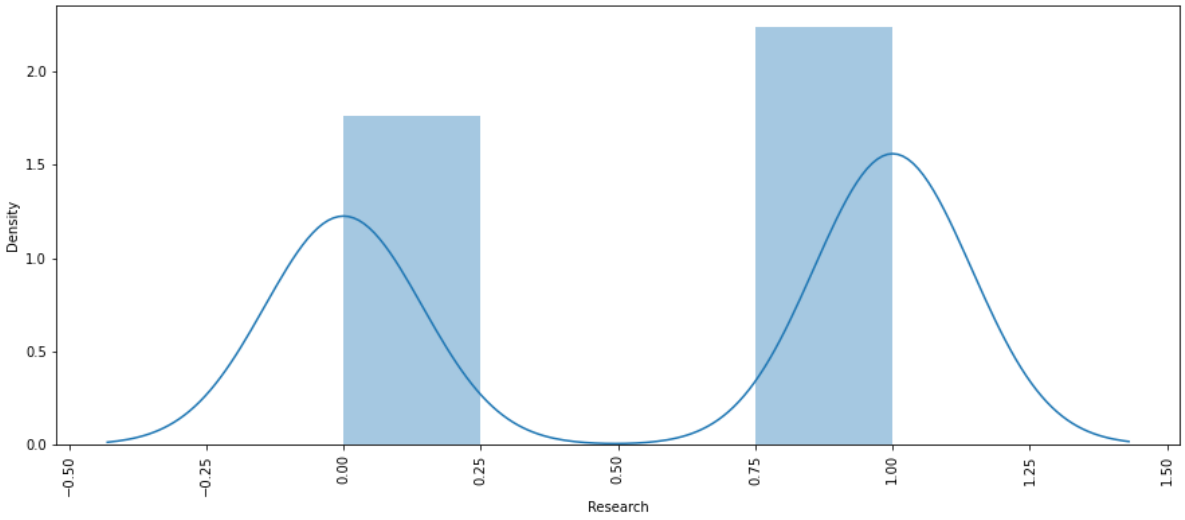
In [29]:

```
plt.figure(figsize=(15,6))
sns.distplot(data['CGPA'])
plt.xticks(rotation = 90)
plt.show()
```



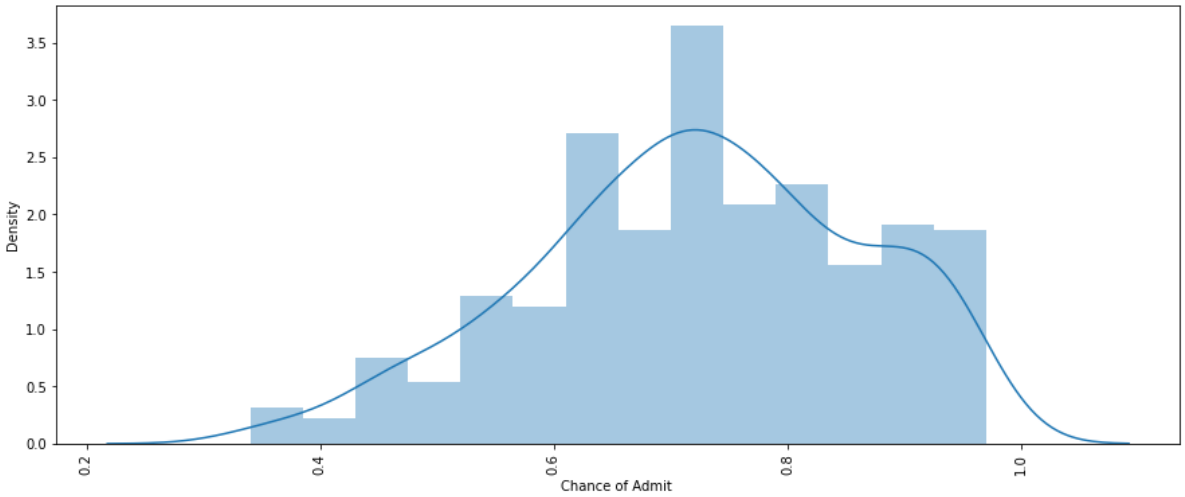
In [30]:

```
plt.figure(figsize=(15,6))
sns.distplot(data['Research'])
plt.xticks(rotation = 90)
plt.show()
```



In [31]:

```
plt.figure(figsize=(15,6))
sns.distplot(data['Chance of Admit '])
plt.xticks(rotation = 90)
plt.show()
```



In [32]:

```
corrmat = data.corr()
corrmat
```

Out[32]:

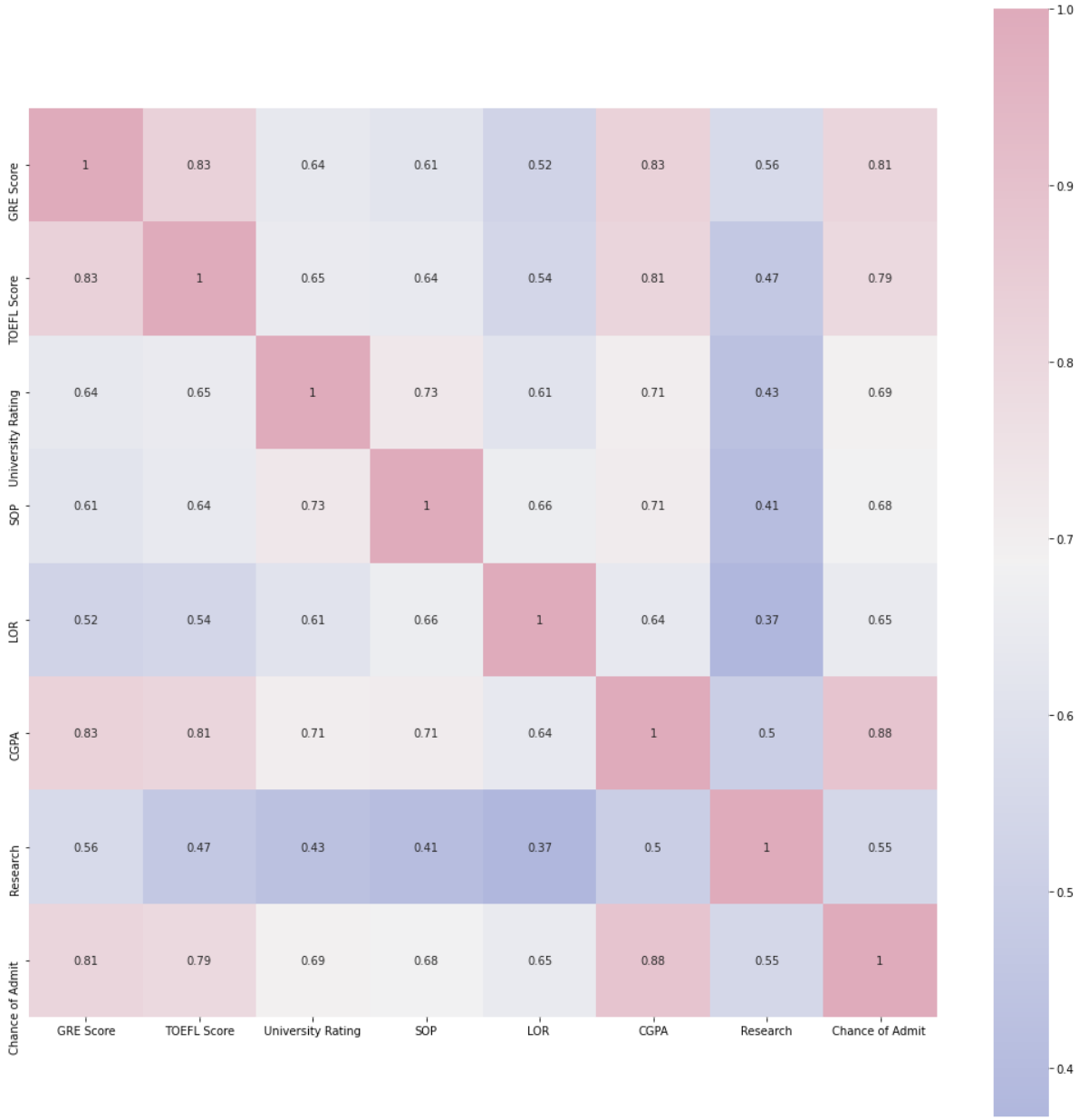
	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	1.000000	0.827200	0.635376	0.613498	0.524679	0.825878	0.563398	0.810351
TOEFL Score	0.827200	1.000000	0.649799	0.644410	0.541563	0.810574	0.467012	0.792228
University Rating	0.635376	0.649799	1.000000	0.728024	0.608651	0.705254	0.427047	0.690132
SOP	0.613498	0.644410	0.728024	1.000000	0.663707	0.712154	0.408116	0.684137
LOR	0.524679	0.541563	0.608651	0.663707	1.000000	0.637469	0.372526	0.645365
CGPA	0.825878	0.810574	0.705254	0.712154	0.637469	1.000000	0.501311	0.882413
Research	0.563398	0.467012	0.427047	0.408116	0.372526	0.501311	1.000000	0.545871
Chance of Admit	0.810351	0.792228	0.690132	0.684137	0.645365	0.882413	0.545871	1.000000

In [33]:

```
cmap = sns.diverging_palette(260,-10,s=50, l=75, n=6, as_cmap=True)
plt.subplots(figsize=(18,18))
sns.heatmap(corrmat,cmap= cmap,annot=True, square=True)
```

Out[33]:

<AxesSubplot:>

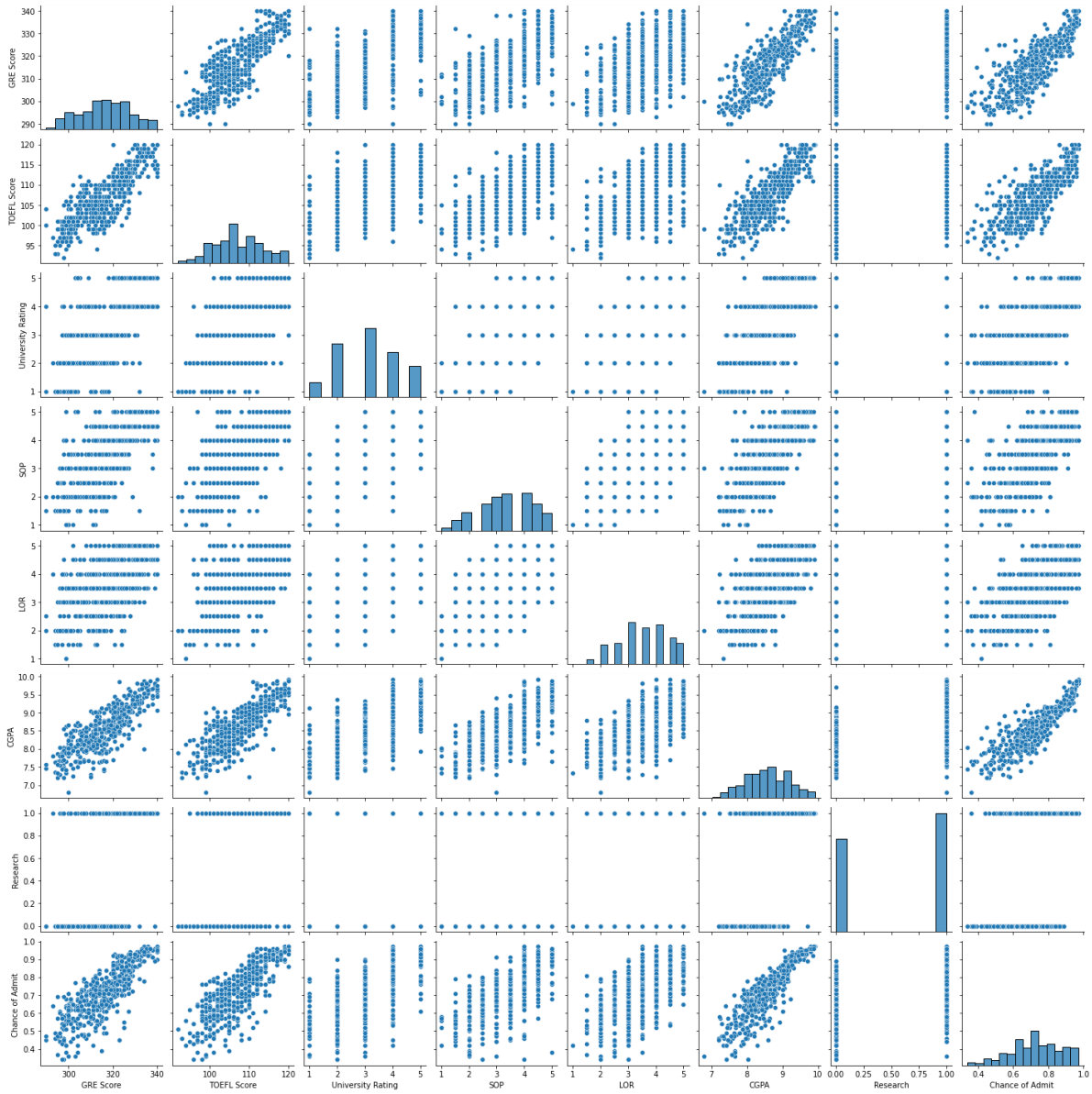


In [34]:

```
1 sns.pairplot(data)
```

Out[34]:

```
<seaborn.axisgrid.PairGrid at 0x6763d6f820>
```



In [35]:

```
1 X=data.drop('Chance of Admit ',axis=1)
2 Y=data['Chance of Admit ']
```

In [36]:

```
1 from sklearn.model_selection import train_test_split
```

In [37]:

```
1 X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,
2                                                random_state=42)
```

In [38]:

```
from sklearn.linear_model import LinearRegression
```

In [39]:

```
lr=LinearRegression()
```

In [40]:

```
lr.fit(X_train,y_train)
```

Out[40]:

```
LinearRegression()
```

In [41]:

```
lr_predict=lr.predict(X_test)
```

In [42]:

```
train_accuracy_lr=lr.score(X_train,y_train)
test_accuracy_lr=lr.score(X_test,y_test)
print("Training Accuracy is", train_accuracy_lr)
print("Test Accuracy is", test_accuracy_lr)
```

```
Training Accuracy is 0.8210671369321554
Test Accuracy is 0.8188432567829629
```