

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import MinMaxScaler
```

```
# Read data from file
df = pd.read_csv("jamboree_admission.csv")
```

```
# Check the basic structure of dataframe
df.head()
```



	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

```
df.tail()
```



	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96
497	498	330	120	5	4.5	5.0	9.56	1	0.93
498	499	312	103	4	4.0	5.0	8.43	0	0.73
499	500	327	113	4	4.5	4.5	9.04	0	0.84

```
# Checking how many rows and columns are there in dataframe
df.shape
```



```
(500, 9)
```

```
# Checking the data type of each column
df.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Serial No.            500 non-null   int64
1   GRE Score             500 non-null   int64
2   TOEFL Score           500 non-null   int64
3   University Rating     500 non-null   int64
4   SOP                   500 non-null   float64
5   LOR                   500 non-null   float64
6   CGPA                  500 non-null   float64
7   Research              500 non-null   int64
8   Chance of Admit       500 non-null   float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

```
# Statistical Summary
df.describe()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

```
# Check for null values
df.isnull().sum()
```

	0
Serial No.	0
GRE Score	0
TOEFL Score	0
University Rating	0
SOP	0
LOR	0
CGPA	0
Research	0
Chance of Admit	0

```
# Check number of values in Serial No. column
df["Serial No."].nunique()
```

```
500
```

```
# Checking the number of unique GRE scores
df["GRE Score"].value_counts()
```



count

GRE Score

312	24
324	23
316	18
321	17
322	17
327	17
311	16
320	16
314	16
317	15
325	15
315	13
308	13
323	13
326	12
319	12
313	12
304	12
300	12
318	12
305	11
301	11
310	11
307	10
329	10
299	10
298	10
331	9
340	9
328	9

```
# Checking the number of unique TOEFL Score values
df["TOEFL Score"].value_counts()
```



	count
TOEFL Score	
110	44
105	37
104	29
107	28
106	28
112	28
103	25
100	24
102	24
99	23
101	20
111	20
108	19
113	19
109	19
114	18
116	16
115	11
118	10
98	10
119	10
120	9
117	8
97	7
96	6
95	3
93	2
94	2
92	1

dtype: int64

```
# Check the number of unique University Rating values
df["University Rating"].value_counts()
```



	count
University Rating	
3	162
2	126
4	105
5	73
1	34

dtype: int64

```
# Check the unique SOP values
df["SOP"].value_counts()
```



	count
SOP	
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

dtype: int64

```
# Check the unique LOR values
df["LOR"].value_counts()
```



	count
LOR	
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

dtype: int64

```
# Check the number of unique CGPA values
df["CGPA"].value_counts()
```



	count
CGPA	
8.76	9
8.00	9
8.12	7
8.45	7
8.54	7
...	...
9.92	1
9.35	1
8.71	1
9.32	1
7.69	1

184 rows × 1 columns

dtype: int64

```
# Check the number of student who have Reasearch experience and those who don't
df["Research"].value_counts()
```

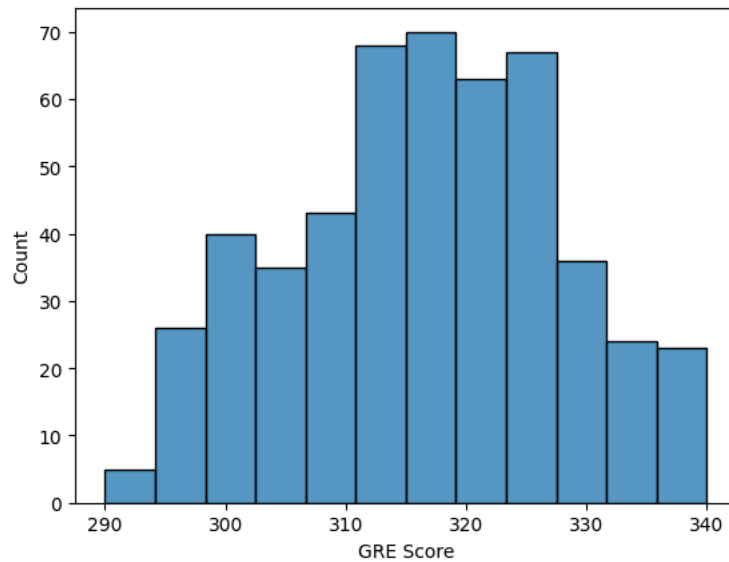


Research	count
1	280
0	220

```
# checking the distribution of GRE Score  
sns.histplot(df["GRE Score"])
```



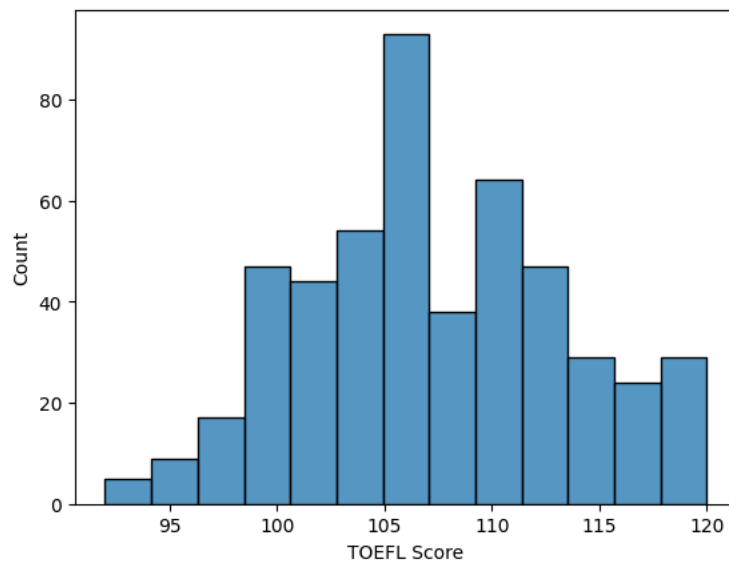
<Axes: xlabel='GRE Score', ylabel='Count'>




```
# Checking the distribution of TOEFL Score  
sns.histplot(df["TOEFL Score"])
```

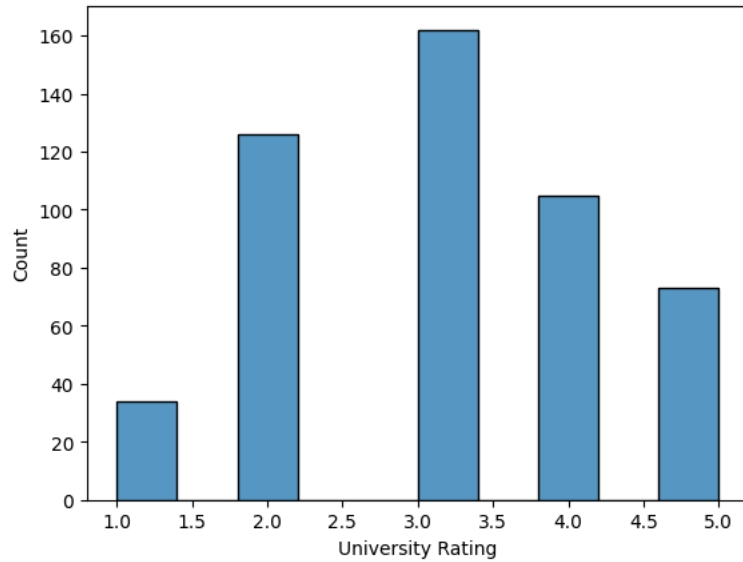


<Axes: xlabel='TOEFL Score', ylabel='Count'>




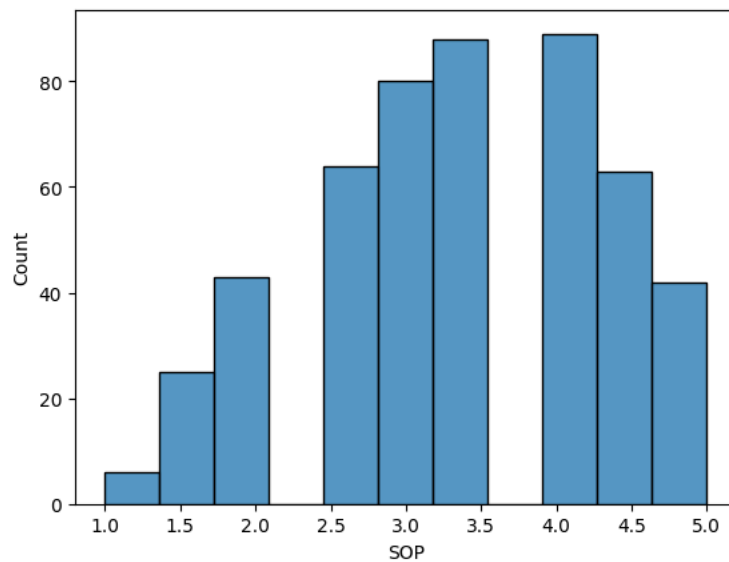
```
# Checking the distribution of University Rating  
sns.histplot(df["University Rating"])
```

 <Axes: xlabel='University Rating', ylabel='Count'>




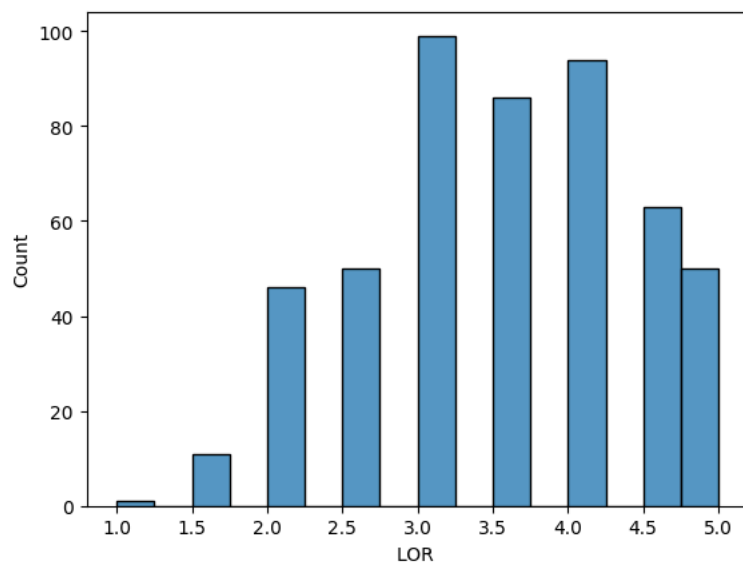
```
# Checking the distribution of SOP  
sns.histplot(df["SOP"])
```

 <Axes: xlabel='SOP', ylabel='Count'>



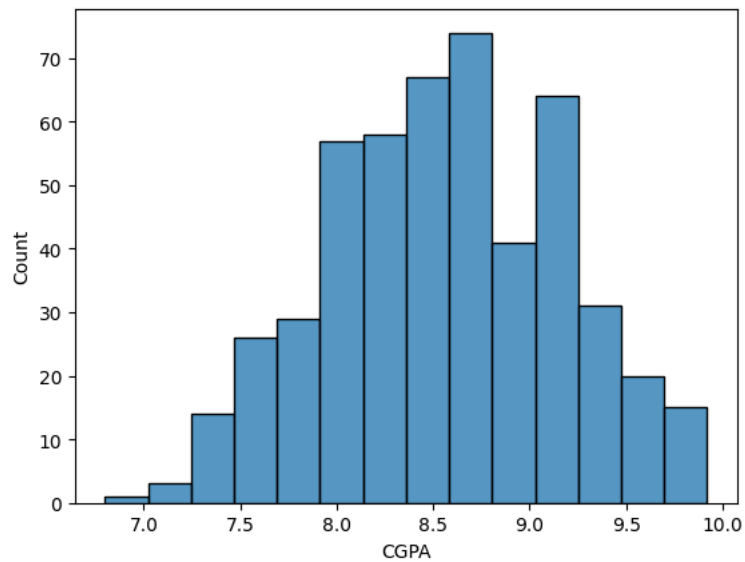
```
# Checking the distribution of LOR  
sns.histplot(df["LOR "])
```

 <Axes: xlabel='LOR ', ylabel='Count'>




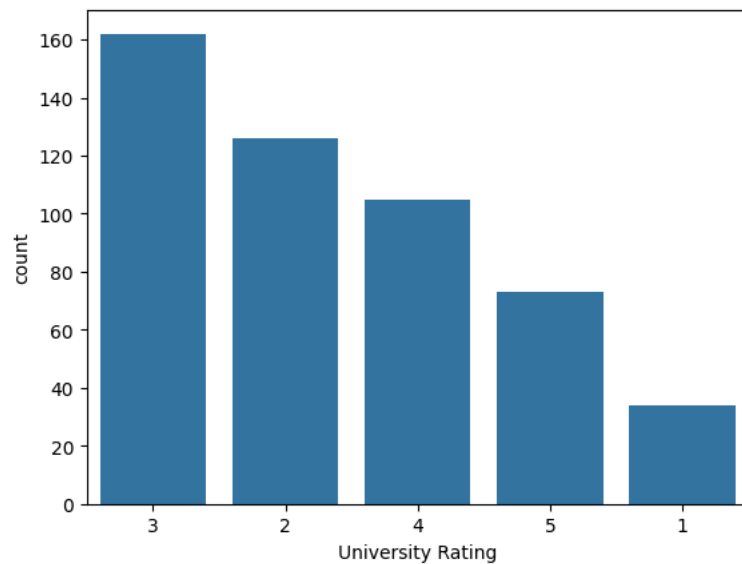
```
# Checking the distribution of CGPA  
sns.histplot(df["CGPA"])
```

 <Axes: xlabel='CGPA', ylabel='Count'>




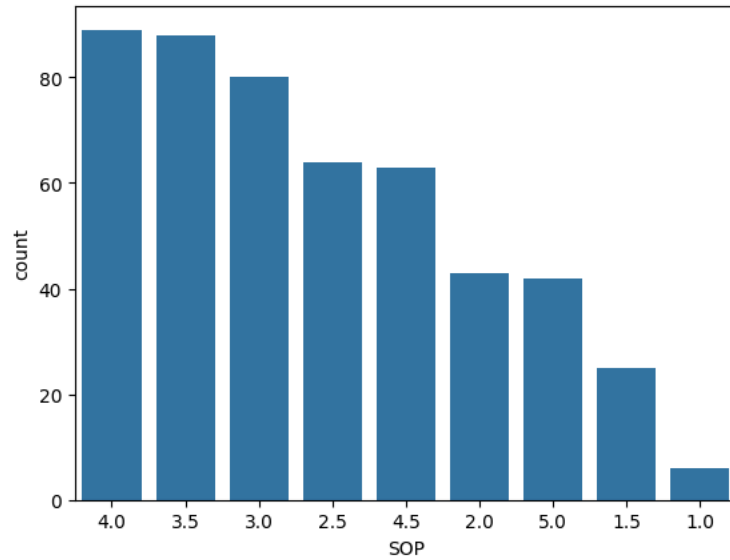
```
# Checking the count of University Rating  
sns.countplot(data=df, x="University Rating", order=df["University Rating"].value_counts().index)
```

 <Axes: xlabel='University Rating', ylabel='count'>




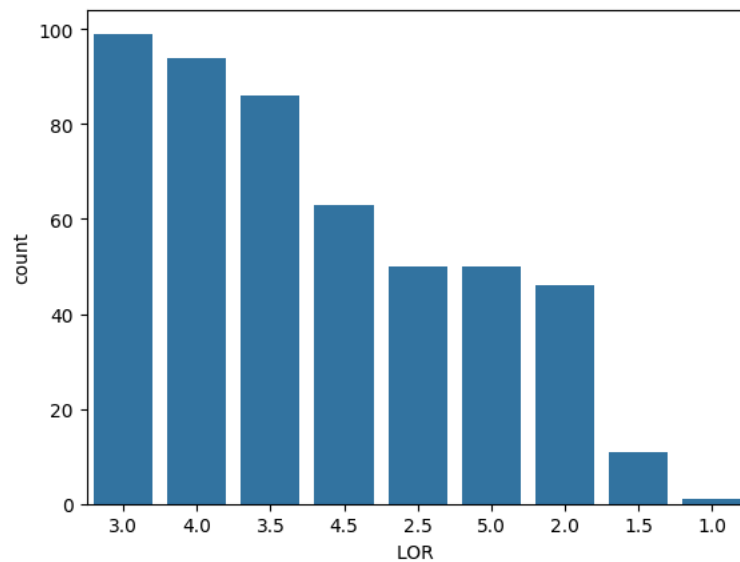
```
# Checking the count of SOP  
sns.countplot(data=df, x="SOP", order=df["SOP"].value_counts().index)
```


 <Axes: xlabel='SOP', ylabel='count'>




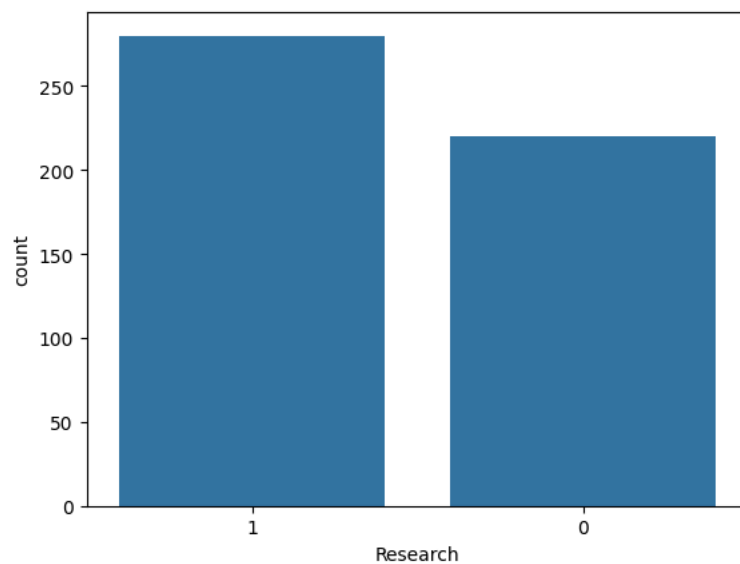
```
# Checking the count of LOR
sns.countplot(data=df, x="LOR ", order=df["LOR "].value_counts().index)
```

 <Axes: xlabel='LOR ', ylabel='count'>



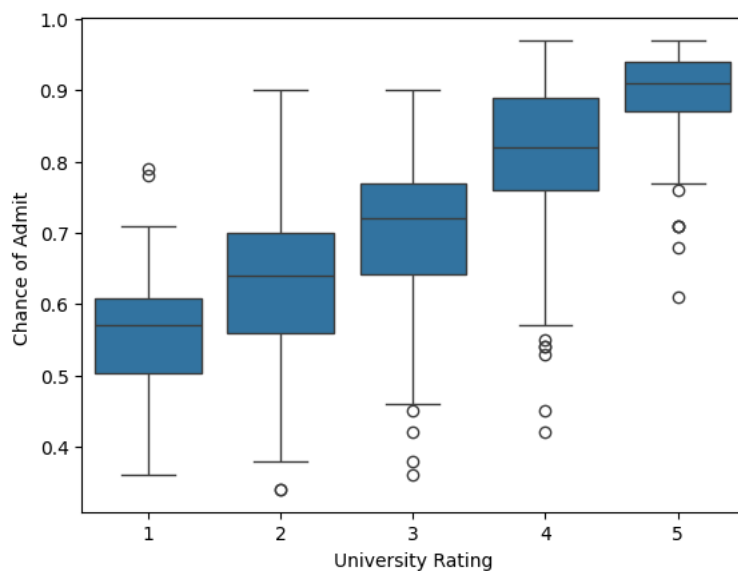
```
# Checking the distribution of Research experience
sns.countplot(data=df, x="Research", order=df["Research"].value_counts().index)
```

 <Axes: xlabel='Research', ylabel='count'>



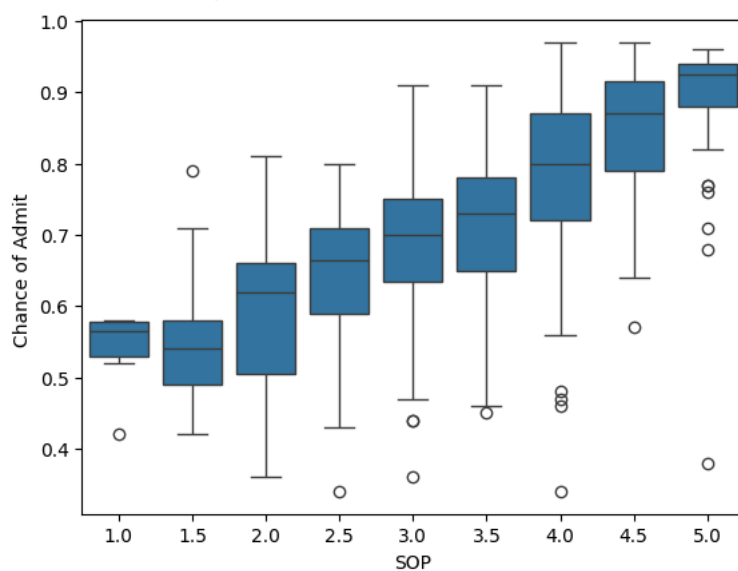
```
# Which University Rating group has more chance of admit.  
sns.boxplot(data=df,x="University Rating", y="Chance of Admit ")
```

<Axes: xlabel='University Rating', ylabel='Chance of Admit '>



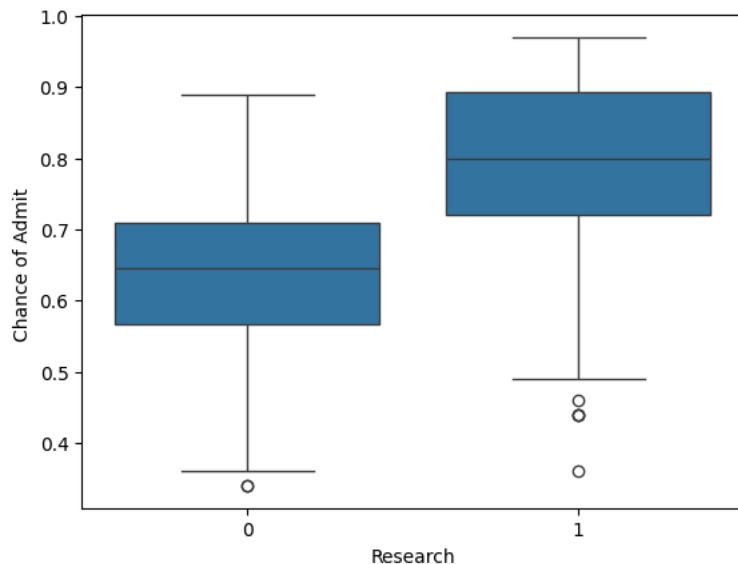
```
# Which SOP group has more chance of admit.  
sns.boxplot(data = df,x = "SOP", y = "Chance of Admit ")
```

<Axes: xlabel='SOP', ylabel='Chance of Admit '>



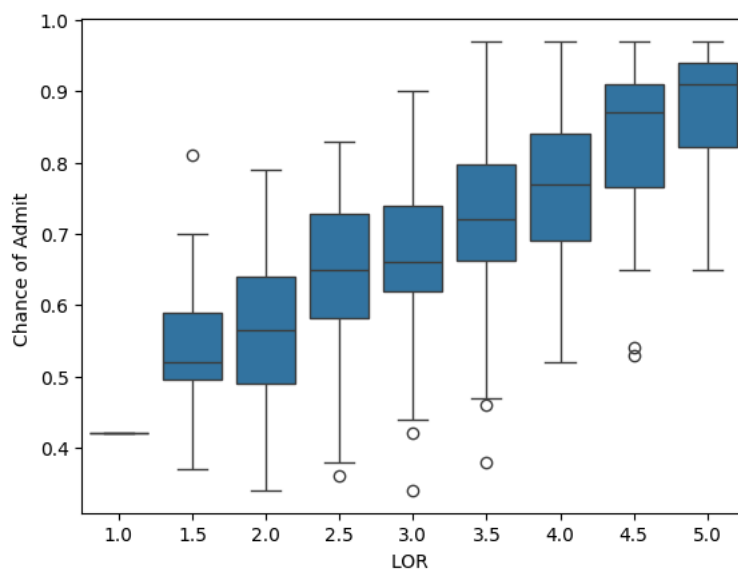
```
# Do students with Research experience have more chance of admit.  
sns.boxplot(data=df, x="Research", y="Chance of Admit ")
```

```
<Axes: xlabel='Research', ylabel='Chance of Admit '>
```



```
# Which LOR group has more chance of admit.
sns.boxplot(data = df,x = "LOR ", y = "Chance of Admit ")
```

```
<Axes: xlabel='LOR ', ylabel='Chance of Admit '>
```



```
# We can see how University Rating effects Chance of Admit
pd.crosstab(df["Chance of Admit "],df["University Rating"])
```

```
<Table: University Rating 1 2 3 4 5>
```

Chance of Admit	1	2	3	4	5
0.34	0	2	0	0	0
0.36	1	0	1	0	0
0.37	1	0	0	0	0
0.38	0	1	1	0	0
0.39	0	1	0	0	0
...
0.93	0	0	0	5	7
0.94	0	0	0	2	11
0.95	0	0	0	3	2
0.96	0	0	0	3	5
0.97	0	0	0	1	3

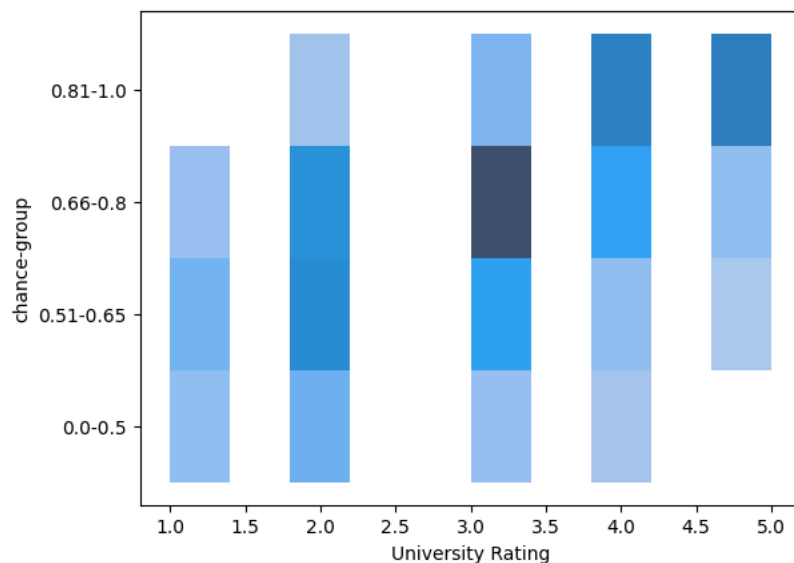
61 rows x 5 columns

```
#Creating bins for Chance of Admit
```

```
df.loc[df["Chance of Admit "].between(0,0.5,"both"),"chance-group"] = "0.0-0.5"
df.loc[df["Chance of Admit "].between(0.51,0.65,"both"),"chance-group"] = "0.51-0.65"
df.loc[df["Chance of Admit "].between(0.66,0.8,"both"),"chance-group"] = "0.66-0.8"
df.loc[df["Chance of Admit "].between(0.81,1.0,"both"),"chance-group"] = "0.81-1.0"
```

```
# Checking the co-relation between University Rating and Chace of Admit group
sns.histplot(x = df["University Rating"], y = df["chance-group"])
```

```
<Axes: xlabel='University Rating', ylabel='chance-group'>
```



```
# Contingency table for chance group pertaining to LOR
```

```
pd.crosstab(df["chance-group"],df["LOR "])
```

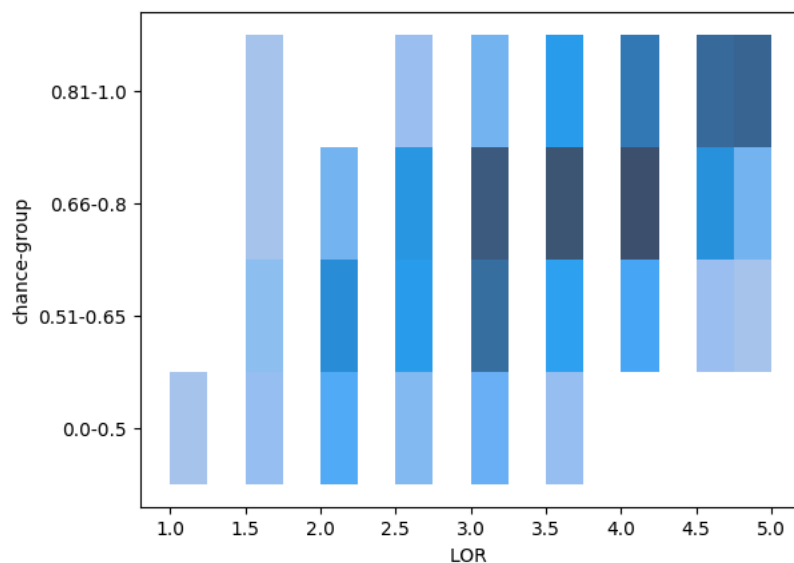
```
<Table>
```

	LOR	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
chance-group										
0.0-0.5		1	4	13	7	10	4	0	0	0
0.51-0.65		0	5	24	19	36	17	14	3	1
0.66-0.8		0	1	9	21	44	46	48	22	9
0.81-1.0		0	1	0	3	9	19	32	38	40

```
# Co-Relation between LOR and chance group
```

```
sns.histplot(x = df["LOR "], y = df["chance-group"])
```

```
<Axes: xlabel='LOR ', ylabel='chance-group'>
```



```
# Contingency table between purchase group specific to a particular marital status
```

```
pd.crosstab(df["chance-group"],df["CGPA"])
```

	CGPA	6.80	7.20	7.21	7.23	7.25	7.28	7.30	7.34	7.36	7.40	...	9.70	9.74	9.76	9.78	9.80	9.82	9.86	9.87	9.91	9.92
chance-group																						
0.0-0.5		1	1	1	0	0	1	0	2	1	0	...	0	0	0	0	0	0	0	0	0	0
0.51-0.65		0	0	0	1	1	0	1	0	0	1	...	0	0	0	0	0	0	0	0	0	0
0.66-0.8		0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0.84-1.0		0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

```
# Checking the co-relation between each and every variable using pairplot
```

```
sns.pairplot(data = df)
```



```
# Convert the 'CGPA' column to numeric, handling the range values
```

```
def convert_to_numeric(value):
```

```
try:
```

```
    return float(value)
```

```
except ValueError:
```

```
    if '-' in value:
```

```
        lower, upper = map(float, value.split('-'))
```

```
        return (lower + upper) / 2
```

```
else:
```

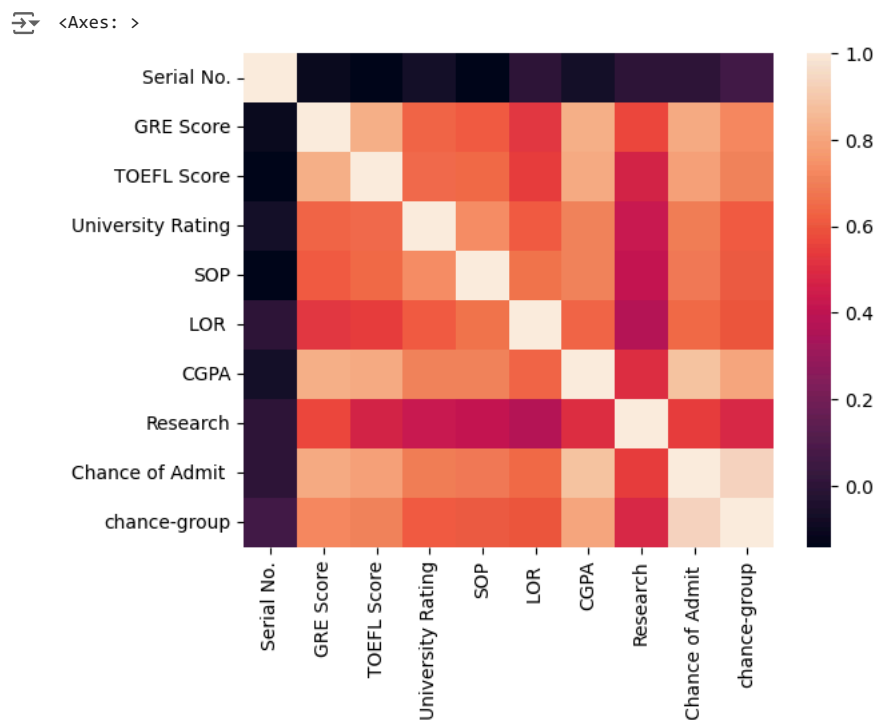
```
    return float('nan') # Handle invalid values as NaN
```

```
# Apply the function to the 'CGPA' column
df['chance-group'] = df['chance-group'].apply(convert_to_numeric)
```

```
# Now calculate the correlation
df1 = df.corr()
df1
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit	chance-group
Serial No.	1.000000	-0.103839	-0.141696	-0.067641	-0.137352	-0.003694	-0.074289	-0.005332	0.008505	0.060601
GRE Score	-0.103839	1.000000	0.827200	0.635376	0.613498	0.524679	0.825878	0.563398	0.810351	0.720849
TOEFL Score	-0.141696	0.827200	1.000000	0.649799	0.644410	0.541563	0.810574	0.467012	0.792228	0.708051
University Rating	-0.067641	0.635376	0.649799	1.000000	0.728024	0.608651	0.705254	0.427047	0.690132	0.611752
SOP	-0.137352	0.613498	0.644410	0.728024	1.000000	0.663707	0.712154	0.408116	0.684137	0.605260
LOR	-0.003694	0.524679	0.541563	0.608651	0.663707	1.000000	0.637469	0.372526	0.645365	0.597306
CGPA	-0.074289	0.825878	0.810574	0.705254	0.712154	0.637469	1.000000	0.501311	0.882413	0.800949
Research	-0.005332	0.563398	0.467012	0.427047	0.408116	0.372526	0.501311	1.000000	0.545871	0.484834
Chance of										

```
# Heatmap for depicting co-relation amongst variables: darker block (black) means that attributes are least co-related.
sns.heatmap(data = df1)
```



```
# Checking whether there are any missing values present or not
df.isna().sum(axis = 0)
```

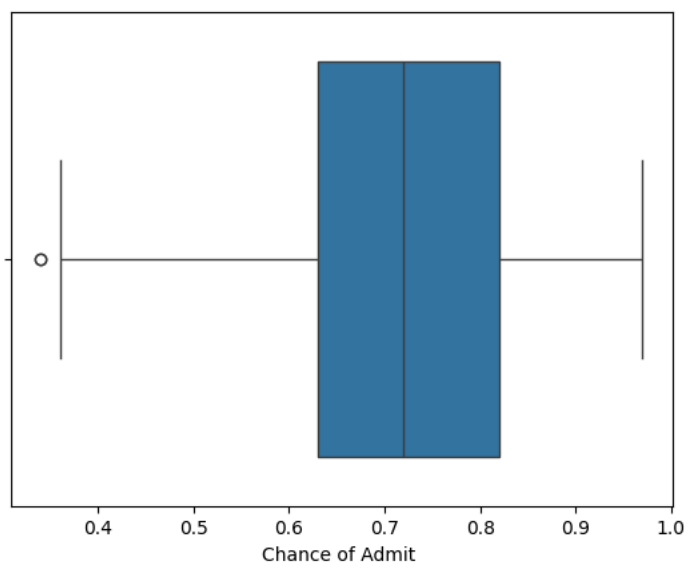
	0
Serial No.	0
GRE Score	0
TOEFL Score	0
University Rating	0
SOP	0
LOR	0
CGPA	0
Research	0
Chance of Admit	0
chance-group	0

```
# Checking Outliers
df.describe()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit	chance-group
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174	0.706560
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114	0.177588
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000	0.250000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000	0.580000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000	0.730000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000	0.905000

```
# Plotting the boxplot for checking outliers: Outliers lie below the value 0.4
sns.boxplot(data = df,
            x = "Chance of Admit ")
```

```
<Axes: xlabel='Chance of Admit '>
```



```
# Checking how many outliers are present in the column chance of admit
df.loc[df["Chance of Admit "]<0.4]["Chance of Admit "].count()
```

```
8
```


```
df.groupby("Research")["Chance of Admit "].agg(["min","max","count","sum"])
```



	min	max	count	sum
Research				
0	0.34	0.89	220	139.68
1	0.36	0.97	280	221.19

Lets separate our dataframe into research experience students and no research experience students and check the mean of averages for :
of different sub-samples.


```
df_no_experience = df[df["Research"] == 0]
df_no_experience.reset_index()
df_no_experience.head()
```



	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit	chance-group
4	5	314	103	2	2.0	3.0	8.21	0	0.65	0.58
7	8	308	101	2	3.0	4.0	7.90	0	0.68	0.73
8	9	302	102	1	2.0	1.5	8.00	0	0.50	0.25
9	10	323	108	3	3.5	3.0	8.60	0	0.45	0.25
15	16	314	105	3	3.5	2.5	8.30	0	0.54	0.58

checking the total number of rows and columns in df_no_experience


```
df_no_experience.shape
```



```
(220, 10)
```

Checking which CGPA in no research experience student tends to admitted to university more.

```
pd.crosstab(df["CGPA"],df["chance-group"])
```



chance-group	0.250	0.580	0.730	0.905
CGPA				
6.80	1	0	0	0
7.20	1	0	0	0
7.21	1	0	0	0
7.23	0	1	0	0
7.25	0	1	0	0
...
9.82	0	0	0	1
9.86	0	0	0	1
9.87	0	0	0	2
9.91	0	0	0	1
9.92	0	0	0	1

184 rows x 4 columns

```
df1 = df.drop(columns=['Serial No.'])
df1
```


	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit	chance-group
0	337	118	4	4.5	4.5	9.65	1	0.92	0.905
1	324	107	4	4.0	4.5	8.87	1	0.76	0.730
2	316	104	3	3.0	3.5	8.00	1	0.72	0.730
3	322	110	3	3.5	2.5	8.67	1	0.80	0.730
4	314	103	2	2.0	3.0	8.21	0	0.65	0.580
...
495	332	108	5	4.5	4.0	9.02	1	0.87	0.905
496	337	117	5	5.0	5.0	9.87	1	0.96	0.905
497	330	120	5	4.5	5.0	9.56	1	0.93	0.905
498	312	103	4	4.0	5.0	8.43	0	0.73	0.730
499	327	113	4	4.5	4.5	9.04	0	0.84	0.905

500 rows × 9 columns

```
for i in df1.columns:
    print(i, ":", '\n', df1[i].unique(), '\n')
```

```
GRE Score :
[337 324 316 322 314 330 321 308 302 323 325 327 328 307 311 317 319 318
 303 312 334 336 340 298 295 310 300 338 331 320 299 304 313 332 326 329
 339 309 315 301 296 294 306 305 290 335 333 297 293]
```

```
TOEFL Score :
[118 107 104 110 103 115 109 101 102 108 106 111 112 105 114 116 119 120
 98 93 99 97 117 113 100 95 96 94 92]
```

```
University Rating :
[4 3 2 5 1]
```

```
SOP :
[4.5 4. 3. 3.5 2. 5. 1.5 1. 2.5]
```

```
LOR :
[4.5 3.5 2.5 3. 4. 1.5 2. 5. 1. ]
```

```
CGPA :
[9.65 8.87 8. 8.67 8.21 9.34 8.2 7.9 8.6 8.4 9. 9.1 8.3 8.7
 8.8 8.5 9.5 9.7 9.8 9.6 7.5 7.2 7.3 8.1 9.4 9.2 7.8 7.7
 9.3 8.85 7.4 7.6 6.8 8.92 9.02 8.64 9.22 9.16 9.64 9.76 9.45 9.04
 8.9 8.56 8.72 8.22 7.54 7.36 8.02 9.36 8.66 8.42 8.28 8.14 8.76 7.92
 7.66 8.03 7.88 7.84 8.96 9.24 8.88 8.46 8.12 8.25 8.47 9.05 8.78 9.18
 9.46 9.38 8.48 8.68 8.34 8.45 8.62 7.46 7.28 8.84 9.56 9.48 8.36 9.32
 8.71 9.35 8.65 9.28 8.77 8.16 9.08 9.12 9.15 9.44 9.92 9.11 8.26 9.43
 9.06 8.75 8.89 8.69 7.86 9.01 8.97 8.33 8.27 7.98 8.04 9.07 9.13 9.23
 8.32 8.98 8.94 9.53 8.52 8.43 8.54 9.91 9.87 7.65 7.89 9.14 9.66 9.78
 9.42 9.26 8.79 8.23 8.53 8.07 9.31 9.17 9.19 8.37 7.68 8.15 8.73 8.83
 8.57 9.68 8.09 8.17 7.64 8.01 7.95 8.49 7.87 7.97 8.18 8.55 8.74 8.13
 8.44 9.47 8.24 7.34 7.43 7.25 8.06 7.67 9.54 9.62 7.56 9.74 9.82 7.96
 7.45 7.94 8.35 7.42 8.95 9.86 7.23 7.79 9.25 9.67 8.86 7.57 7.21 9.27
 7.81 7.69]
```

```
Research :
[1 0]
```

```
Chance of Admit :
[0.92 0.76 0.72 0.8 0.65 0.9 0.75 0.68 0.5 0.45 0.52 0.84 0.78 0.62
 0.61 0.54 0.66 0.63 0.64 0.7 0.94 0.95 0.97 0.44 0.46 0.74 0.91 0.88
 0.58 0.48 0.49 0.53 0.87 0.86 0.89 0.82 0.56 0.36 0.42 0.47 0.55 0.57
 0.96 0.93 0.38 0.34 0.79 0.71 0.69 0.59 0.85 0.77 0.81 0.83 0.67 0.73
 0.6 0.43 0.51 0.39 0.37]
```

```
chance-group :
[0.905 0.73 0.58 0.25 ]
```

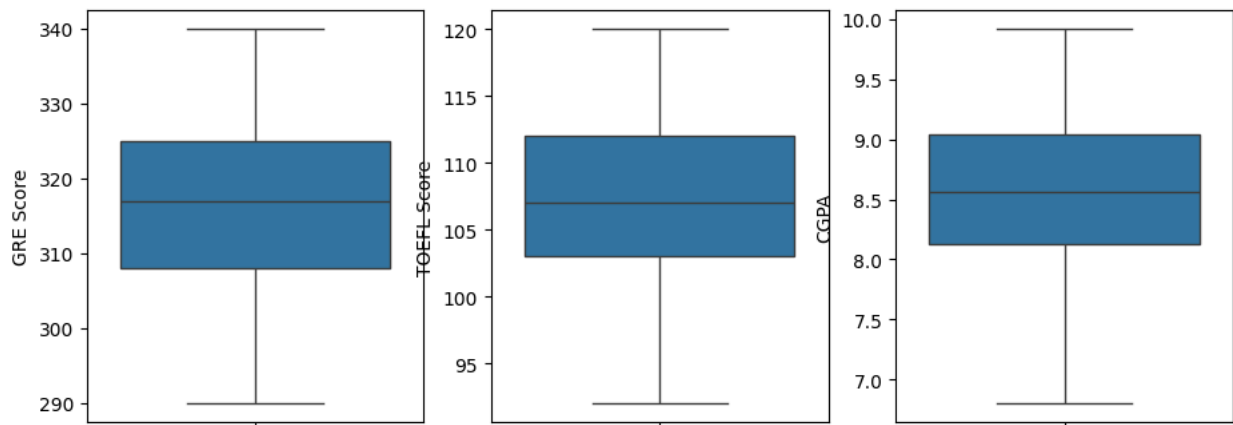
```
df1 = df1[['University Rating', 'SOP', 'LOR ', 'Research', 'GRE Score', 'TOEFL Score', 'CGPA', 'Chance of Admit ']]
df1 = df1.rename(columns={'LOR ': 'LOR', 'Chance of Admit ': 'Chance of Admit'})
df1
```



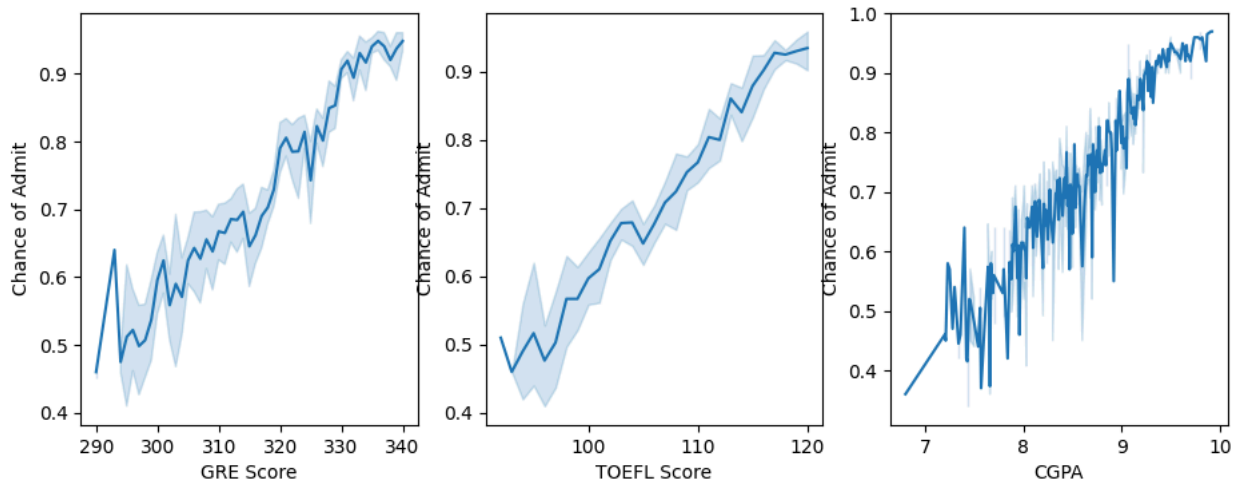
	University Rating	SOP	LOR	Research	GRE Score	TOEFL Score	CGPA	Chance of Admit
0	4	4.5	4.5	1	337	118	9.65	0.92
1	4	4.0	4.5	1	324	107	8.87	0.76
2	3	3.0	3.5	1	316	104	8.00	0.72
3	3	3.5	2.5	1	322	110	8.67	0.80
4	2	2.0	3.0	0	314	103	8.21	0.65
...
495	5	4.5	4.0	1	332	108	9.02	0.87
496	5	5.0	5.0	1	337	117	9.87	0.96
497	5	4.5	5.0	1	330	120	9.56	0.93
498	4	4.0	5.0	0	312	103	8.43	0.73
499	4	4.5	4.5	0	327	113	9.04	0.84

500 rows × 8 columns

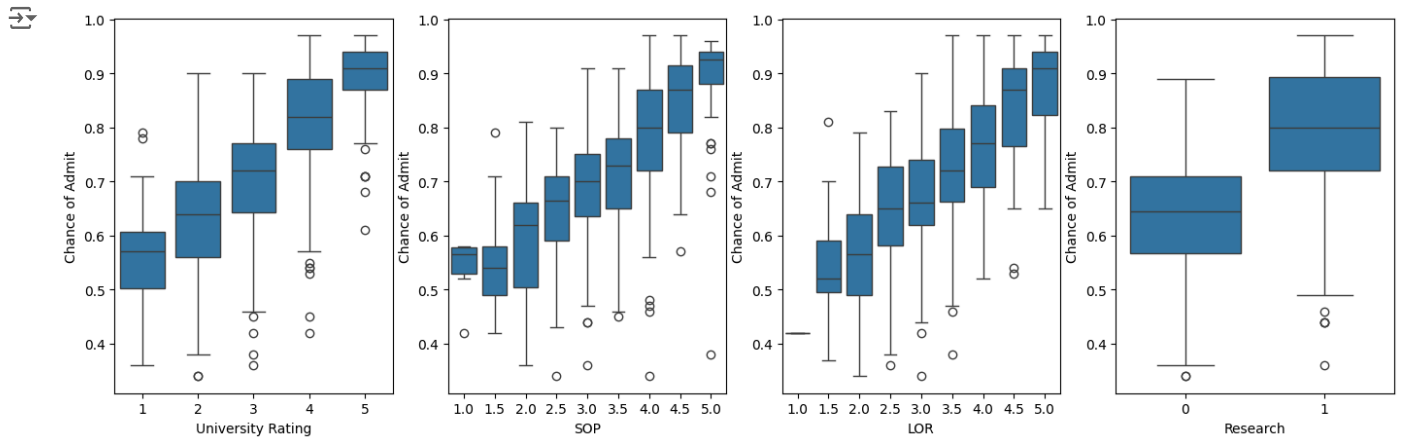
```
# univariate analysis
count = 1
plt.figure(figsize=(15,4))
for i in ['GRE Score', 'TOEFL Score', 'CGPA']:
    plt.subplot(1,4,count)
    sns.boxplot(y= df1[i])
    count += 1
plt.show()
```



```
# Bivariate analysis
count = 1
plt.figure(figsize=(15,4))
for i in ['GRE Score', 'TOEFL Score', 'CGPA']:
    plt.subplot(1,4,count)
    sns.lineplot(y =df1['Chance of Admit'], x= df1[i])
    count += 1
plt.show()
```

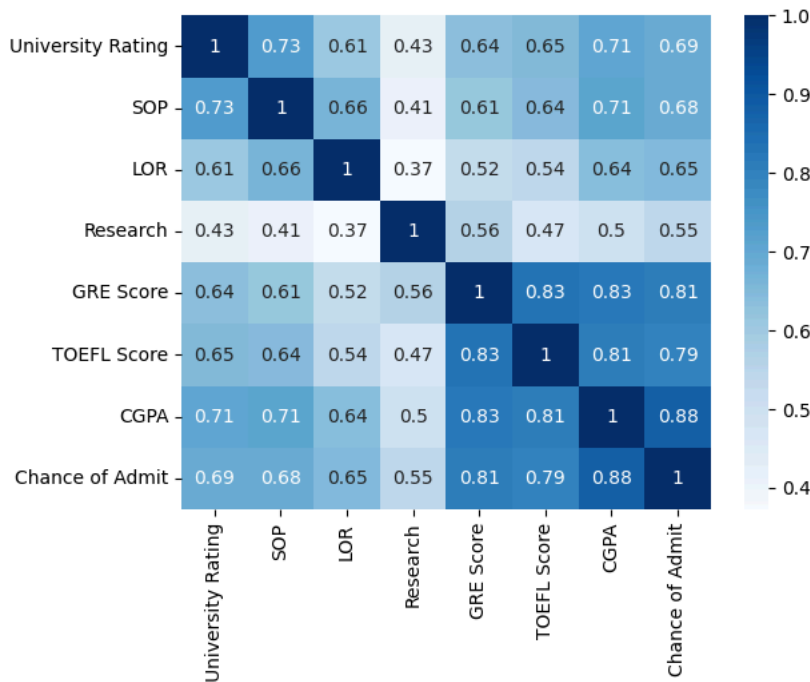


```
# Multivariate analysis
count = 1
plt.figure(figsize=(17,5))
for i in ['University Rating', 'SOP', 'LOR', 'Research']:
    plt.subplot(1,4,count)
    sns.boxplot(y =df1['Chance of Admit'], x= df1[i])
    count += 1
plt.show()
```



```
# Continous variable analysis
sns.heatmap(df1.corr(), annot=True, cmap='Blues')
```

<Axes: >



```
# Outliers Treatment:
count = 1
plt.figure(figsize=(17,5))
for i in ['University Rating', 'SOP', 'LOR', 'Research']:
    plt.subplot(1,4,count)
    sns.boxplot( y= df1[i])
    count += 1
plt.show()
```

