In [24]:
```
#Importing all libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import preprocessing
```

In [25]:
```
df=pd.read_csv("https://raw.githubusercontent.com/Mukund94/Datasets/main/admission.
```

a) Visualize the 10 random rows of the data set

Q1. Perform Exploratory Data Analysis (EDA) tasks

a) Visualize the 10 random rows of the data set
b) Generate the description for numeric variables
c) Check the shape of the data set
d) Generate the correlation matrix
e) Generate a correlogram

In [26]:
```
df.sample(10)
```

Out[26]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
499	500	327	113	4	4.5	4.5	9.04	0	0.84
61	62	307	101	3	4.0	3.0	8.20	0	0.47
193	194	336	118	5	4.5	5.0	9.53	1	0.94
243	244	325	114	3	3.5	3.0	9.04	1	0.76
29	30	310	99	2	1.5	2.0	7.30	0	0.54
356	357	327	109	3	3.5	4.0	8.77	1	0.79
454	455	310	105	2	3.0	3.5	8.01	0	0.71
442	443	331	116	4	4.5	4.5	9.44	1	0.92
80	81	312	105	3	2.0	3.0	8.02	1	0.50
245	246	328	110	4	4.0	2.5	9.02	1	0.81

b) Generate the description for numeric variables

In [27]:
```
df.describe(include='number')
```

Out[27]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Resea
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000(
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560(
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496(
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000(
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000(
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000(
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000(
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000(

c) Check the shape of the data set

In [28]:
```
df.shape
```

Out[28]:
```
(500, 9)
```

d) Generate the correlation matrix

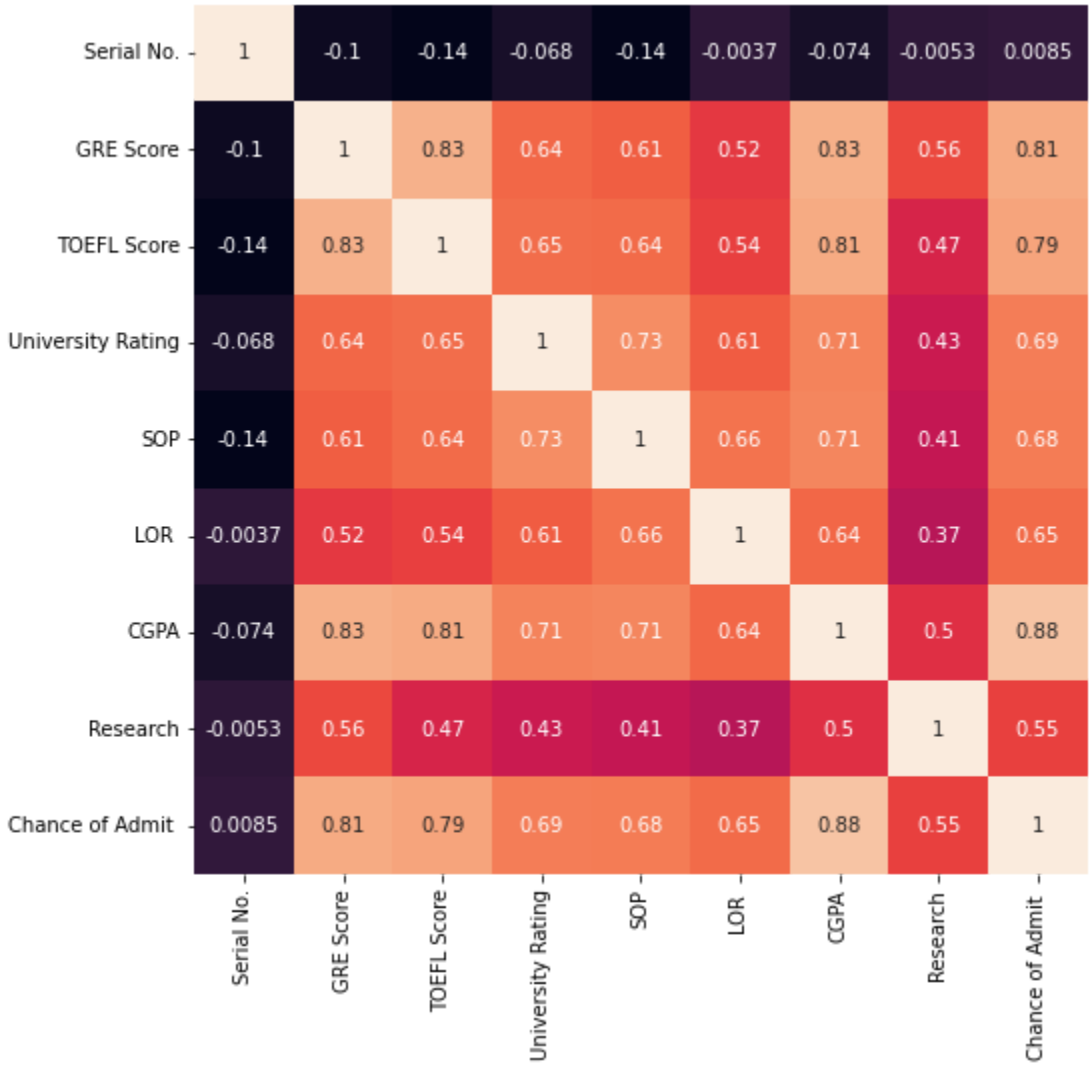
In [29]:
```
df.corr()
```

Out[29]:

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

e) Generate a correlogram

In [30]:
```
plt.figure(figsize=(8,8))
sns.heatmap(df.corr(),annot=True,cbar=False)
plt.show()
```



Q.2 Find out the minimum and maximum values for GRE score

In [31]:
```
#Minimum Value
df['GRE Score'].min()
```

Out[31]:
```
290
```

In [32]:
```
#Maximum Value
df['GRE Score'].max()
```

Out[32]:
```
340
```

Q.3 Find out the percentage of universities for each university rating

In [33]:
```
df['University Rating'].value_counts(normalize=True).mul(100).round(1).astype(str)
```

Out[33]:
```
3    32.4%
2    25.2%
4    21.0%
5    14.6%
1     6.8%
Name: University Rating, dtype: object
```

Q.4 Convert the target variable “Chance of Admit” to categorical having values 0 and 1,such that :

Students having the “Chance of Admit” value > 0.80, are assigned value 1, and
Students having the “Chance of Admit” value < 0.80, are assigned value 0
Where 0: Low chance of Admission and 1: High chance of admission

In [34]:
```
df['Chance of Admit '].value_counts()
```

Out[34]:
```
0.71    23
0.64    19
0.73    18
0.72    16
0.79    16
        ..
0.38     2
0.36     2
0.43     1
0.39     1
0.37     1
Name: Chance of Admit , Length: 61, dtype: int64
```

In [35]:
```
#Variable conversion
df['Chance of Admit '] = np.where(df['Chance of Admit '] >=0.80, 1, df['Chance of A
df['Chance of Admit '] = np.where(df['Chance of Admit '] <0.80, 0, df['Chance of Ad
```

In [36]:
```
df.head()
```

Out[36]:

	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	1.0
1	2	324	107	4	4.0	4.5	8.87	1	0.0
2	3	316	104	3	3.0	3.5	8.00	1	0.0
3	4	322	110	3	3.5	2.5	8.67	1	1.0
4	5	314	103	2	2.0	3.0	8.21	0	0.0

Now above 'Chance of Admit' variable is categorical

In [37]:
```
#Counts of 'Chance of Admit' variable
df['Chance of Admit '].value_counts()
```

Out[37]:
```
0.0    345
1.0    155
Name: Chance of Admit , dtype: int64
```

Q.5 Build a Decision Tree classifier, to predict whether a student has a low or high chance of admission to a chosen university. Perform Hyperparameter Tuning to improve the accuracy of the model.

In [38]:
```
X=df.drop('Chance of Admit ',axis=1 )
y=df['Chance of Admit ']
```

In [39]:
```
#Importing the 'train_test_split' package and Spliding the train and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_stat
```

In [40]:
```
#Importing the 'DecisionTreeClassifier' Model and

from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='gini',max_depth=4)
model.fit(X_train, y_train)
```

Out[40]:
```
DecisionTreeClassifier(max_depth=4)
```

In [41]:
```
model.score(X_train, y_train)
```

Out[41]:
```
0.94
```

In [42]:
```
model.score(X_test, y_test)
```

Out[42]:
```
0.94
```

In [43]:
```
from sklearn.metrics import accuracy_score

# sets of hyperparameters
params_1 = {'criterion': 'gini', 'splitter': 'best', 'max_depth': 50}
params_2 = {'criterion': 'entropy', 'splitter': 'random', 'max_depth': 70}
params_3 = {'criterion': 'gini', 'splitter': 'random', 'max_depth': 60}
params_4 = {'criterion': 'entropy', 'splitter': 'best', 'max_depth': 80}
params_5 = {'criterion': 'gini', 'splitter': 'best', 'max_depth': 40}

# Separate models
model_1 = DecisionTreeClassifier(**params_1)
model_2 = DecisionTreeClassifier(**params_2)
model_3 = DecisionTreeClassifier(**params_3)
model_4 = DecisionTreeClassifier(**params_4)
model_5 = DecisionTreeClassifier(**params_5)

model_1.fit(X_train, y_train)
model_2.fit(X_train, y_train)
model_3.fit(X_train, y_train)
model_4.fit(X_train, y_train)
model_5.fit(X_train, y_train)

# Prediction sets
preds_1 = model_1.predict(X_test)
preds_2 = model_3.predict(X_test)
preds_3 = model_3.predict(X_test)
preds_4 = model_4.predict(X_test)
preds_5 = model_5.predict(X_test)

print(f'Accuracy on Model 1: {round(accuracy_score(y_test, preds_1), 3)}')
print(f'Accuracy on Model 2: {round(accuracy_score(y_test, preds_2), 3)}')
print(f'Accuracy on Model 3: {round(accuracy_score(y_test, preds_3), 3)}')
print(f'Accuracy on Model 4: {round(accuracy_score(y_test, preds_4), 3)}')
print(f'Accuracy on Model 5: {round(accuracy_score(y_test, preds_5), 3)}')
```

```
Accuracy on Model 1: 0.94
Accuracy on Model 2: 0.9
Accuracy on Model 3: 0.9
Accuracy on Model 4: 0.94
Accuracy on Model 5: 0.93
```

In [ ]: