In [1]:
```
#importing all the necessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:
```
# importing the dataset from github
```

In [3]:
```
ga = pd.read_csv('https://raw.githubusercontent.com/divyansha1115/Graduate-Admiss
```

In [4]:
```
#finding first 5 rows of dataset
ga.head()
```

Out[4]:

	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

In [5]:
```
#statistical analysis of our data
ga.describe()
```

Out[5]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Rese
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.00
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.54
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.49
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.00
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.00
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.00
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.00
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.00

In [6]:
```
#finding the datatypes of the data
ga.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         400 non-null    int64  
 1   GRE Score          400 non-null    int64  
 2   TOEFL Score        400 non-null    int64  
 3   University Rating  400 non-null    int64  
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64  
 8   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 28.2 KB
```

In [7]:
```
#finding the features
ga.columns
```

Out[7]:
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

In [8]:
```
ga.columns.unique()
```

Out[8]:
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

In [9]:
```
#dropping the serial no from our dataset
ga.drop(['Serial No.'], axis =1, inplace = True)
```

In [10]:
```
ga.head()
```

Out[10]:

	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 [11]:
```
#finding the shape of dataset
ga.shape
```

Out[11]:
```
(400, 8)
```

In [12]:
```
#checking the null values
ga.isnull().sum()
```

Out[12]:
```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [13]:
```
#checking the duplicate values
ga.duplicated().sum()
```

Out[13]:
```
0
```

In [14]:
```
#finding the correlation
correlation = ga.corr()
```

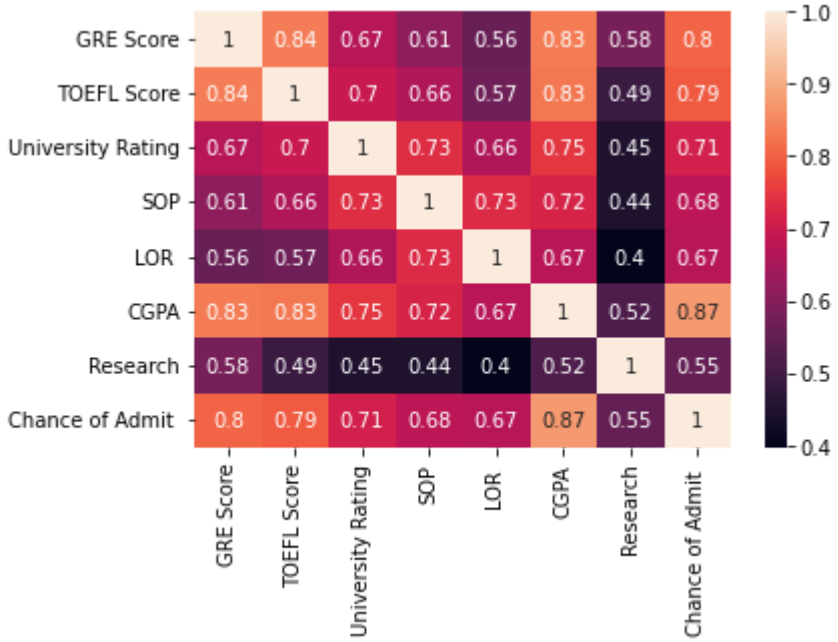
In [15]:
```
correlation
```

Out[15]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	1.000000	0.835977	0.668976	0.612831	0.557555	0.833060	0.580391	0.802610
TOEFL Score	0.835977	1.000000	0.695590	0.657981	0.567721	0.828417	0.489858	0.791594
University Rating	0.668976	0.695590	1.000000	0.734523	0.660123	0.746479	0.447783	0.711250
SOP	0.612831	0.657981	0.734523	1.000000	0.729593	0.718144	0.444029	0.675732
LOR	0.557555	0.567721	0.660123	0.729593	1.000000	0.670211	0.396859	0.669889
CGPA	0.833060	0.828417	0.746479	0.718144	0.670211	1.000000	0.521654	0.873289
Research	0.580391	0.489858	0.447783	0.444029	0.396859	0.521654	1.000000	0.553202
Chance of Admit	0.802610	0.791594	0.711250	0.675732	0.669889	0.873289	0.553202	1.000000

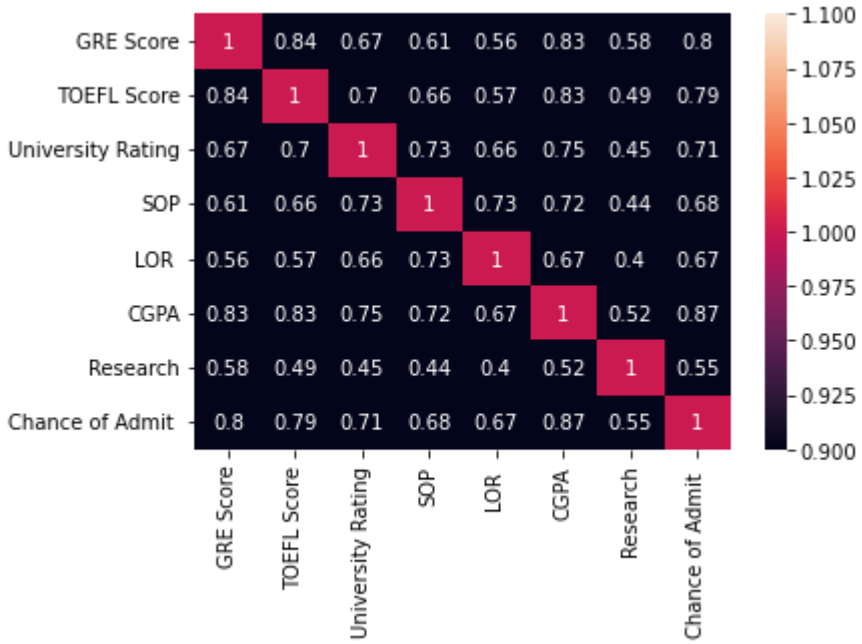
In [16]:
```
#plotting the heatmap
sns.heatmap(data = correlation, annot=True)
```

Out[16]: <AxesSubplot:>



In [17]:
```
sns.heatmap(data = correlation, annot=True, vmin=1, vmax=1)
```

Out[17]: <AxesSubplot:>



In [18]:
```
#find the covaraiance
ga.cov()
```

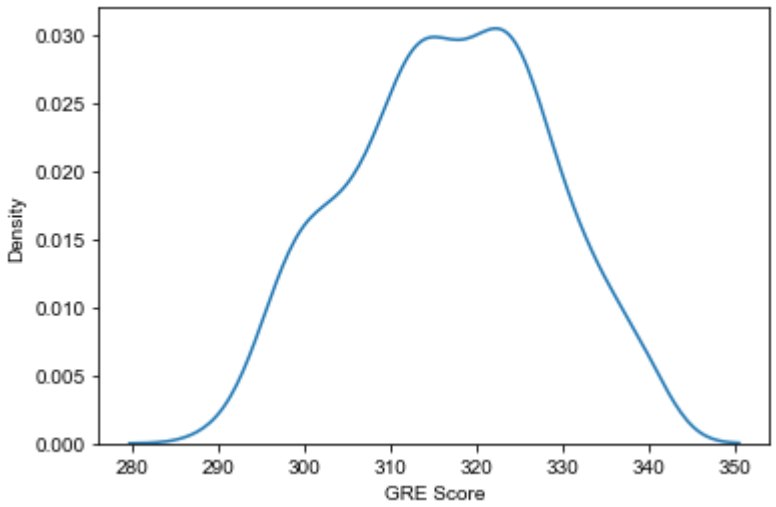
Out[18]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	131.644555	58.216967	8.778791	7.079699	5.747726	5.699742	3.318690	1.313271
TOEFL Score	58.216967	36.838997	4.828697	4.021053	3.095965	2.998337	1.481729	0.685179
University Rating	8.778791	4.828697	1.308114	0.845865	0.678352	0.509117	0.255232	0.116009
SOP	7.079699	4.021053	0.845865	1.013784	0.660025	0.431183	0.222807	0.097028
LOR	5.747726	3.095965	0.678352	0.660025	0.807262	0.359084	0.177701	0.085834
CGPA	5.699742	2.998337	0.509117	0.431183	0.359084	0.355594	0.155026	0.074265
Research	3.318690	1.481729	0.255232	0.222807	0.177701	0.155026	0.248365	0.039317
Chance of Admit	1.313271	0.685179	0.116009	0.097028	0.085834	0.074265	0.039317	0.020337

In [19]:
```
#finding the datatypes
ga.dtypes
```

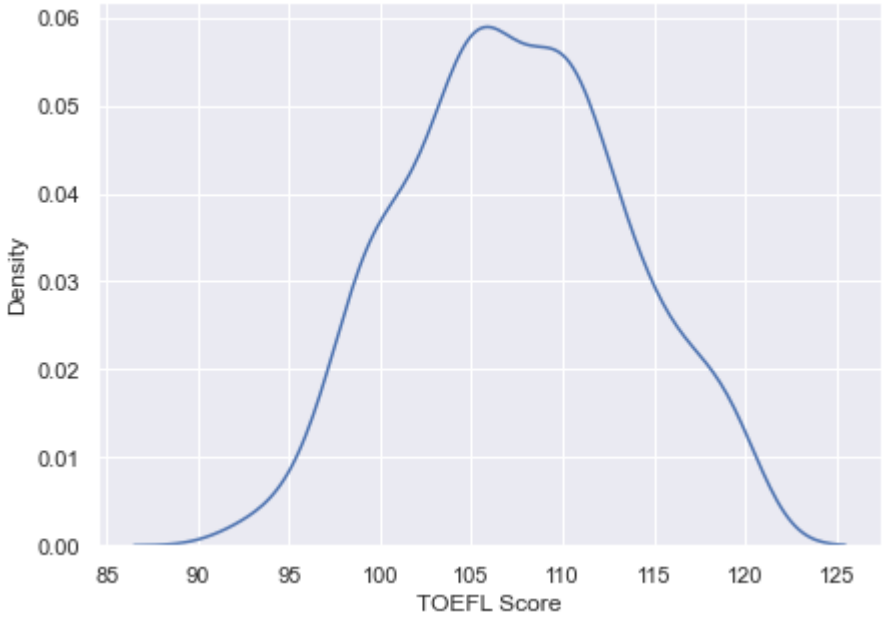
Out[19]:
```
GRE Score            int64
TOEFL Score          int64
University Rating    int64
SOP                float64
LOR                float64
CGPA               float64
Research             int64
Chance of Admit    float64
dtype: object
```

In [20]:
```
#plotting the kde plot for indivisual feature
sns.kdeplot(data= ga, x= 'GRE Score')
sns.set(rc={'figure.figsize':(7,5)})
```

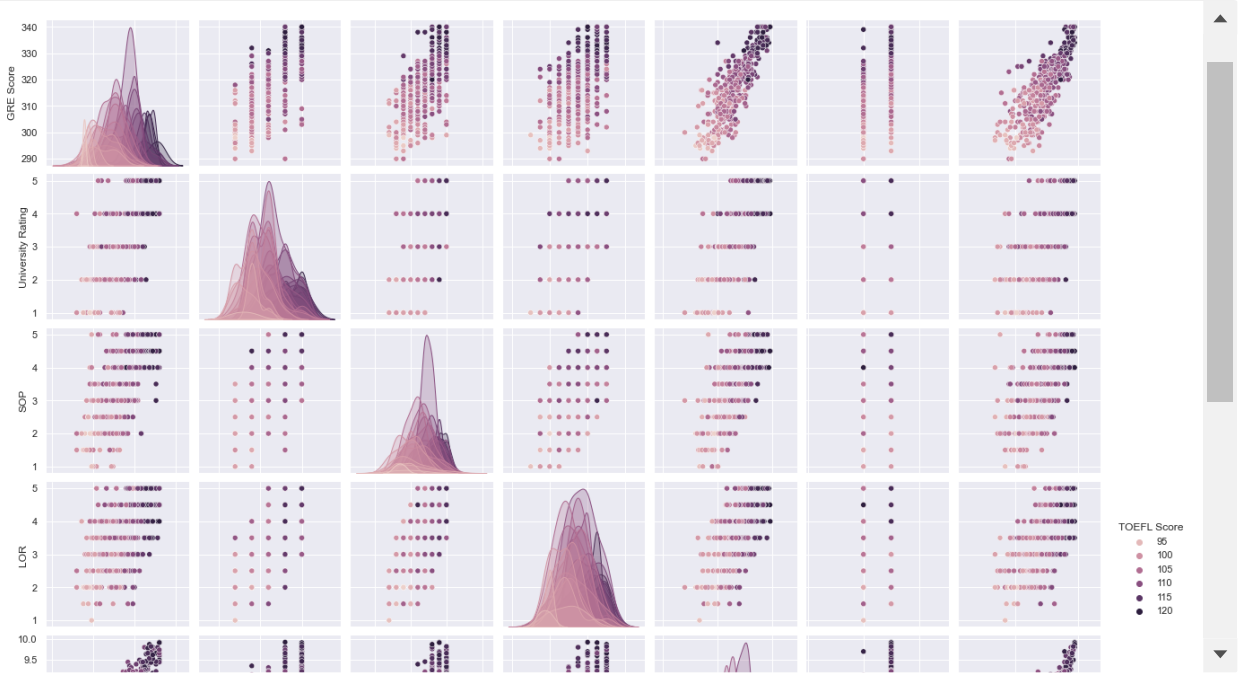


In [21]:
```
sns.kdeplot(data= ga, x= 'TOEFL Score')
```

Out[21]:
```
<AxesSubplot:xlabel='TOEFL Score', ylabel='Density'>
```

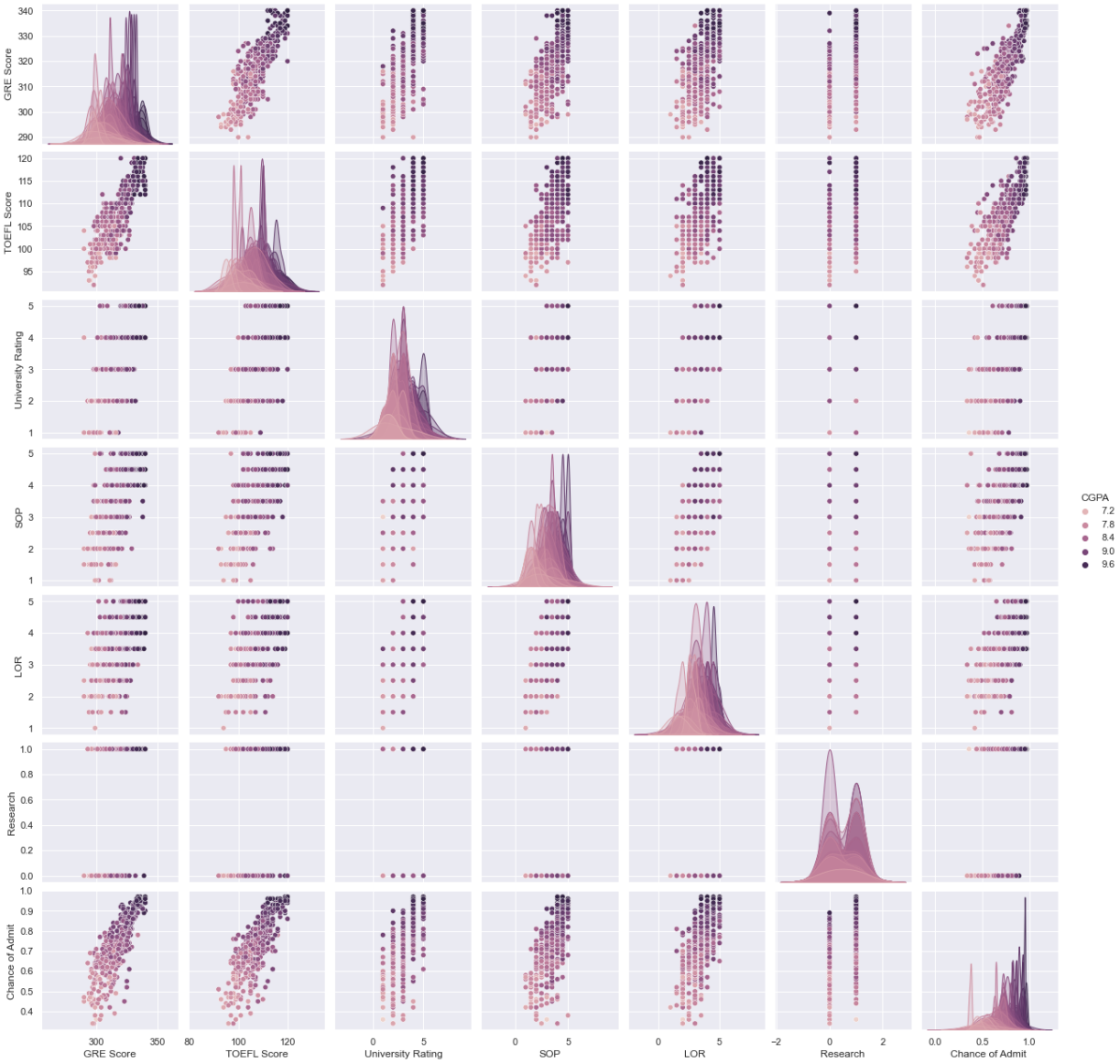


In [22]:
```
sns.pairplot(ga,hue = 'TOEFL Score')
```



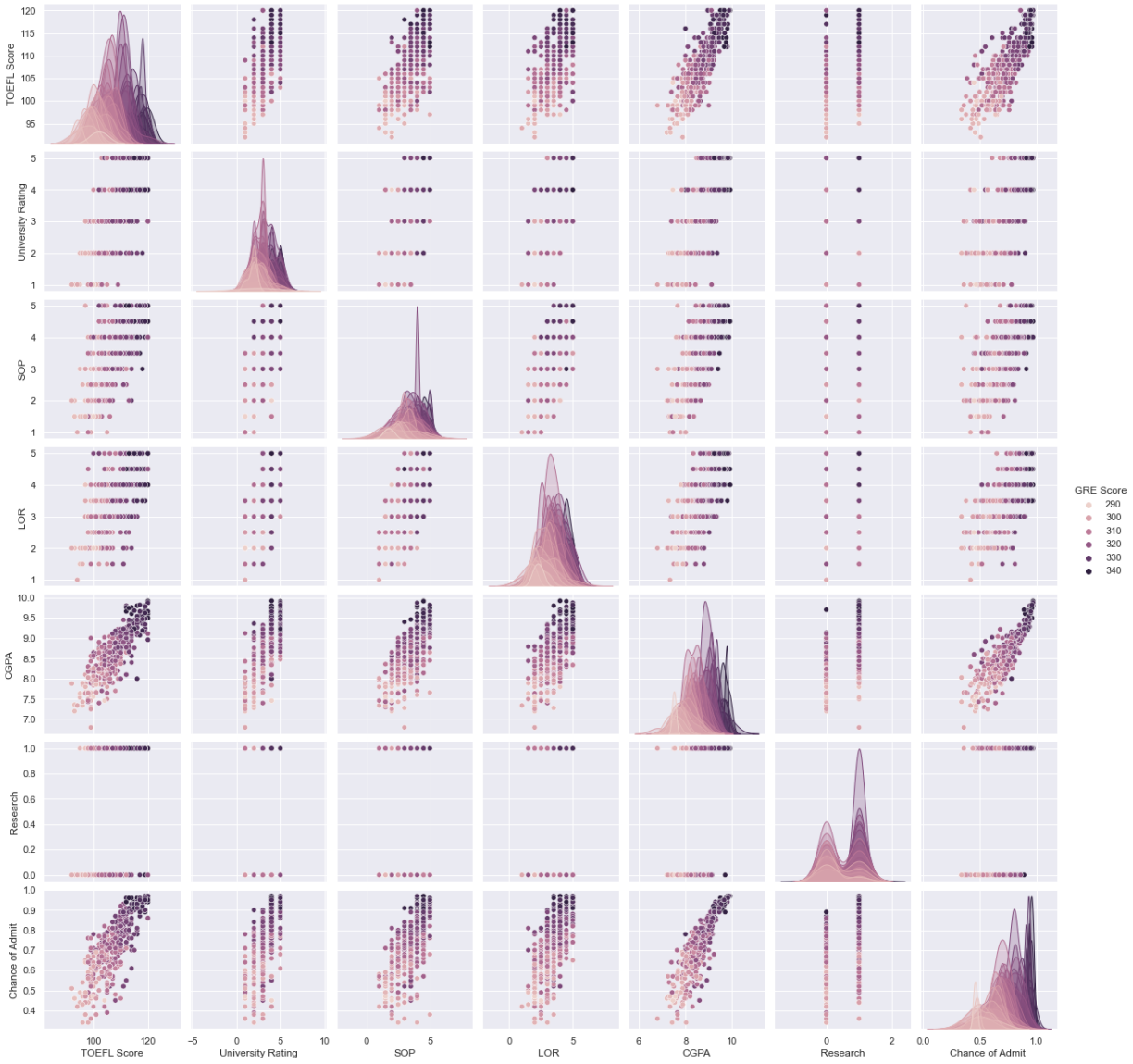
In [24]:
```
sns.pairplot(ga,hue = 'CGPA')
```

Out[24]: <seaborn.axisgrid.PairGrid at 0x22733e8a940>



In [25]:
```
sns.pairplot(ga,hue = 'GRE Score')
```

Out[25]: <seaborn.axisgrid.PairGrid at 0x2273494fac0>



In [26]:
```
#dependent and indepedent variable
x = ga.iloc[:,:-1]
y = ga.iloc[:,-1]
```

In [27]:
```
x
```

Out[27]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	337	118	4	4.5	4.5	9.65	1
1	324	107	4	4.0	4.5	8.87	1
2	316	104	3	3.0	3.5	8.00	1
3	322	110	3	3.5	2.5	8.67	1
4	314	103	2	2.0	3.0	8.21	0
...	...	...	...	...	...	...	...
395	324	110	3	3.5	3.5	9.04	1
396	325	107	3	3.0	3.5	9.11	1
397	330	116	4	5.0	4.5	9.45	1
398	312	103	3	3.5	4.0	8.78	0
399	333	117	4	5.0	4.0	9.66	1

400 rows × 7 columns

In [28]:
```
y
```

Out[28]:
```
0      0.92
1      0.76
2      0.72
3      0.80
4      0.65
       ...
395    0.82
396    0.84
397    0.91
398    0.67
399    0.95
Name: Chance of Admit , Length: 400, dtype: float64
```

In [29]:
```
#train test split from sklearn
from sklearn.model_selection import train_test_split
```

In [30]:
```
x_train, x_test, y_train, y_test = train_test_split(
...     x, y, test_size=0.33, random_state=42)
```

In [31]:
```
#finding x_train, y_train shape
x_train.shape, y_train.shape
```

Out[31]:
```
((268, 7), (268,))
```

In [32]:
```
#perform standard scaling
#importing standard scaler from sklearn

from sklearn.preprocessing import StandardScaler
```

In [33]:
```
sc = StandardScaler()
```

In [34]:
```
sc
```

Out[34]:
```
StandardScaler()
```

In [35]:
```
#fit and transform our training data in standard scaler
x_train_tf = sc.fit_transform(x_train, y_train)
```

In [36]:
```
#only transforming our test data in standard scaler
x_test_tf = sc.transform(x_test)
```

In [37]:
```
#importing svr from sklearn svm, since its a regression problem
from sklearn.svm import SVR
```

In [38]:
```
svr = SVR()
```

In [39]:
```
svr
```

Out[39]:
```
SVR()
```

In [40]:
```
#fitting our data in to our model svr
svr.fit(x_train_tf, y_train)
```

Out[40]:
```
SVR()
```

In [41]:
```
#predicting the values
y_pred = svr.predict(x_test_tf)
```

In [42]:
```
#performing permance metix
from sklearn.metrics import r2_score
```

In [43]:
```
#finding r2 score
svr_r2score = r2_score(y_test, y_pred)
```

In [44]:
```
svr_r2score
```

Out[44]:
```
0.7312041068084392
```

```
Observation: here we got 73%
```

In [ ]:
```

```