In [26]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
df = pd.read_csv("US_graduate_schools_admission_parameters_dataset.csv")
```

In [3]:

```
df
```

Out[3]:

	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
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 9 columns

In [4]:

```
df.info
```

Out[4]:

```
<bound method DataFrame.info of      Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  \
0             1        337          118                  4  4.5   4.5  9.65
1             2        324          107                  4  4.0   4.5  8.87
2             3        316          104                  3  3.0   3.5  8.00
3             4        322          110                  3  3.5   2.5  8.67
4             5        314          103                  2  2.0   3.0  8.21
..          ...        ...          ...                ...  ...   ...   ...
395         396        324          110                  3  3.5   3.5  9.04
396         397        325          107                  3  3.0   3.5  9.11
397         398        330          116                  4  5.0   4.5  9.45
398         399        312          103                  3  3.5   4.0  8.78
399         400        333          117                  4  5.0   4.0  9.66

     Research  Chance of Admit
0           1              0.92
1           1              0.76
2           1              0.72
3           1              0.80
4           0              0.65
..        ...               ...
395         1              0.82
396         1              0.84
397         1              0.91
398         0              0.67
399         1              0.95

[400 rows x 9 columns]>
```

In [5]:

```
df.describe()
```

Out[5]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

In [6]:

```
df.columns
```

Out[6]:

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

In [7]:

```
df.isnull()
```

Out[7]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
395	False	False	False	False	False	False	False	False	False
396	False	False	False	False	False	False	False	False	False
397	False	False	False	False	False	False	False	False	False
398	False	False	False	False	False	False	False	False	False
399	False	False	False	False	False	False	False	False	False

400 rows × 9 columns

In [8]:

```
df = df.drop(columns="Serial No.",axis=1)
```

In [9]:

```
df
```

Out[9]:

	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
...	...	...	...	...	...	...	...	...
395	324	110	3	3.5	3.5	9.04	1	0.82
396	325	107	3	3.0	3.5	9.11	1	0.84
397	330	116	4	5.0	4.5	9.45	1	0.91
398	312	103	3	3.5	4.0	8.78	0	0.67
399	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 8 columns

In [10]:

```
df.head
```

Out[10]:

```
<bound method NDFrame.head of      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

     Chance of Admit
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

[400 rows x 8 columns]>
```

In [11]:

```
dataset1 = df[['GRE Score','TOEFL Score','University Rating','CGPA','Research', 'Chance of Admit ']]
```

In [12]:

```
dataset1
```

Out[12]:

	GRE Score	TOEFL Score	University Rating	CGPA	Research	Chance of Admit
0	337	118	4	9.65	1	0.92
1	324	107	4	8.87	1	0.76
2	316	104	3	8.00	1	0.72
3	322	110	3	8.67	1	0.80
4	314	103	2	8.21	0	0.65
...	...	...	...	...	...	...
395	324	110	3	9.04	1	0.82
396	325	107	3	9.11	1	0.84
397	330	116	4	9.45	1	0.91
398	312	103	3	8.78	0	0.67
399	333	117	4	9.66	1	0.95

400 rows × 6 columns

In [13]:

```
from sklearn.preprocessing import StandardScaler
threshold = 0.75
dataset1['Chance of admit(y/n)'] = dataset1['Chance of Admit '].apply(lambda x: 'Yes' if x > threshold else 'No')
dataset1.drop(columns='Chance of Admit ', inplace=True)
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_22108\598640578.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy)
  dataset1['Chance of admit(y/n)'] = dataset1['Chance of Admit '].apply(lambda x: 'Yes' if x > threshold else 'No')
C:\Users\DELL\AppData\Local\Temp\ipykernel_22108\598640578.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy)
  dataset1.drop(columns='Chance of Admit ', inplace=True)
```

In [14]:

```
dataset1
```

Out[14]:

	GRE Score	TOEFL Score	University Rating	CGPA	Research	Chance of admit(y/n)
0	337	118	4	9.65	1	Yes
1	324	107	4	8.87	1	Yes
2	316	104	3	8.00	1	No
3	322	110	3	8.67	1	Yes
4	314	103	2	8.21	0	No
...	...	...	...	...	...	...
395	324	110	3	9.04	1	Yes
396	325	107	3	9.11	1	Yes
397	330	116	4	9.45	1	Yes
398	312	103	3	8.78	0	No
399	333	117	4	9.66	1	Yes

400 rows × 6 columns

In [15]:

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
dataset1['Chance of admit(y/n)'] = label_encoder.fit_transform(dataset1['Chance of admit(y/n)'])
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_22108\3398778307.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy)
  dataset1['Chance of admit(y/n)'] = label_encoder.fit_transform(dataset1['Chance of admit(y/n)'])
```

In [16]:

```
dataset1
```

Out[16]:

	GRE Score	TOEFL Score	University Rating	CGPA	Research	Chance of admit(y/n)
0	337	118	4	9.65	1	1
1	324	107	4	8.87	1	1
2	316	104	3	8.00	1	0
3	322	110	3	8.67	1	1
4	314	103	2	8.21	0	0
...	...	...	...	...	...	...
395	324	110	3	9.04	1	1
396	325	107	3	9.11	1	1
397	330	116	4	9.45	1	1
398	312	103	3	8.78	0	0
399	333	117	4	9.66	1	1

400 rows × 6 columns

In [17]:

```
from sklearn.model_selection import train_test_split

X = dataset1.drop(columns=['GRE Score','TOEFL Score','University Rating','CGPA','Research'])
y = dataset1['Chance of admit(y/n)']
```

In [18]:

```
dataset1['Chance of admit(y/n)']
```

Out[18]:

```
0      1
1      1
2      0
3      1
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Chance of admit(y/n), Length: 400, dtype: int32
```

In [19]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [20]:

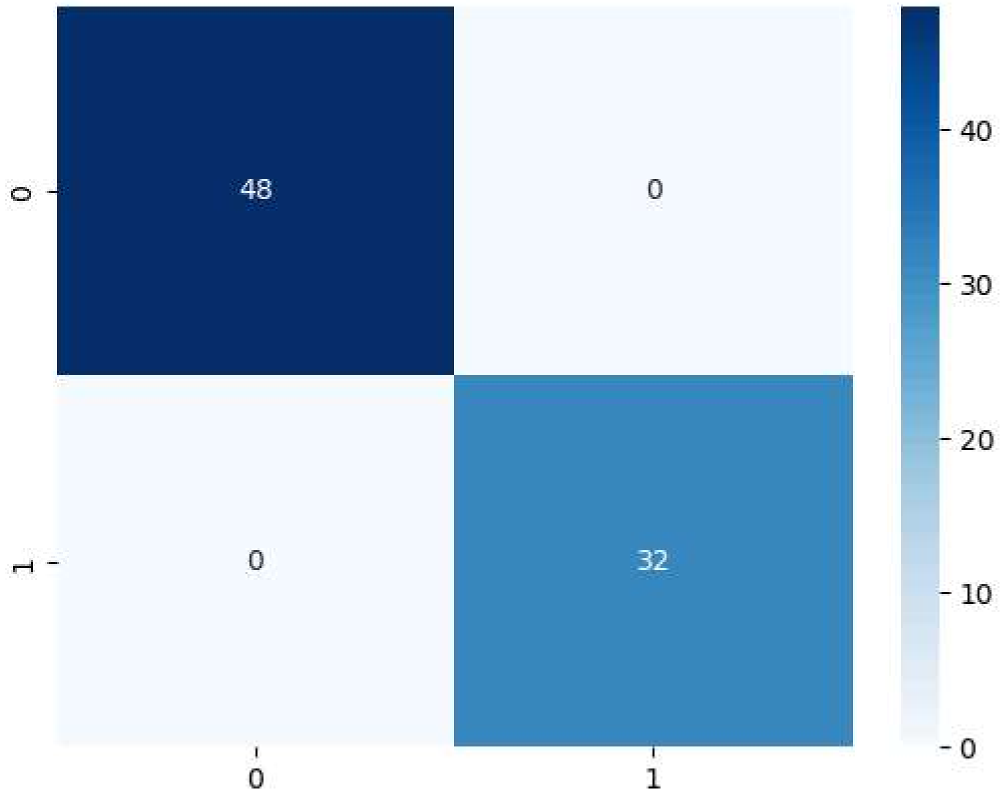
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

report = classification_report(y_test, y_pred)

print(report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        48
           1       1.00      1.00      1.00        32

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80
```

In [23]:

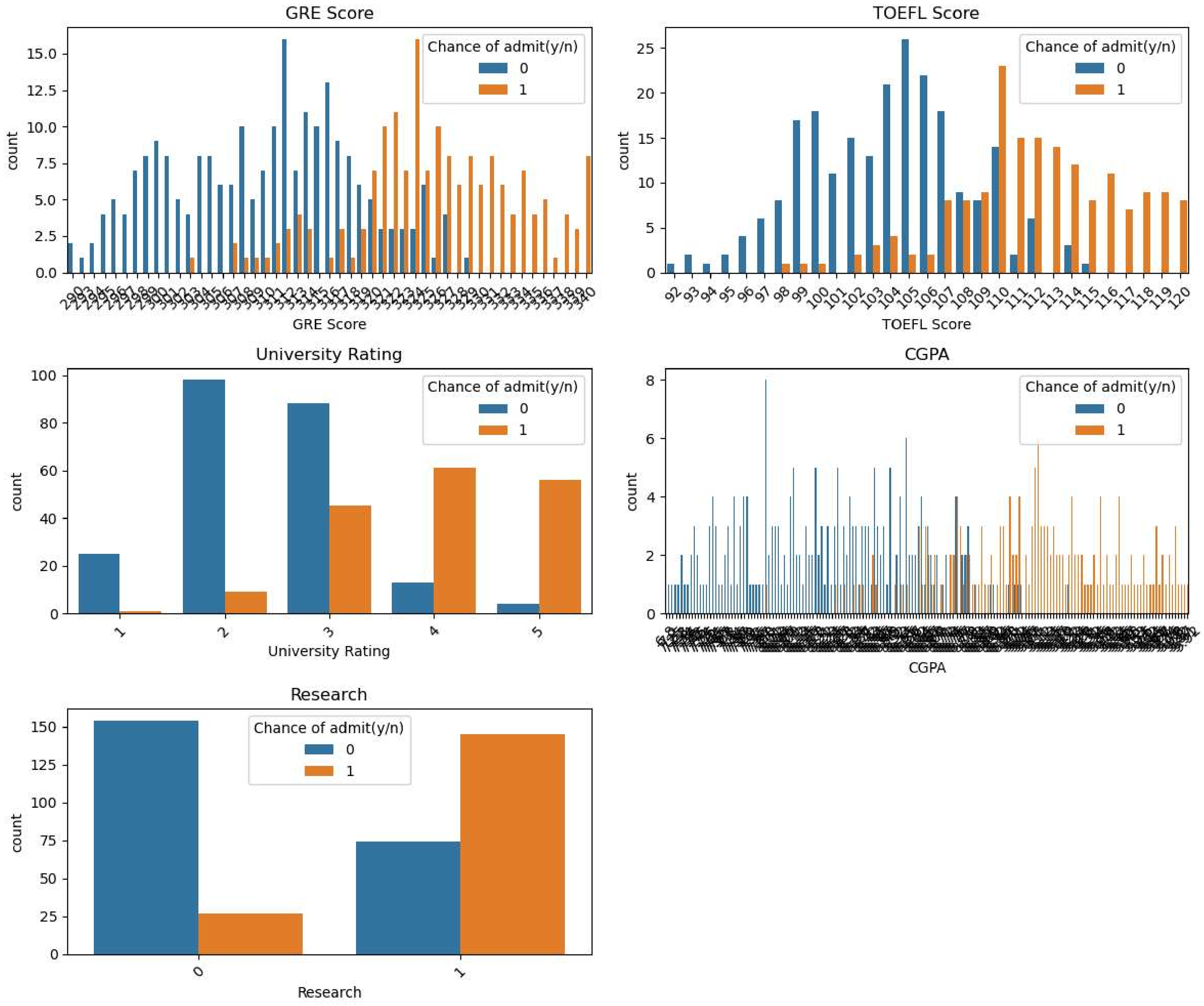
```
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[23], line 4
      2 conf_matrix = confusion_matrix(y_test, y_pred)
      3 sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')
----> 4 plt.xlabel('Predicted')
      5 plt.ylabel('Actual')
      6 plt.title('Confusion Matrix')

File ~\anaconda3\Lib\site-packages\matplotlib\_api\__init__.py:226, in caching_module_getattr.<locals>.__getattr__(na
me)
    224 if name in props:
    225     return props[name].__get__(instance)
--> 226 raise AttributeError(
    227     f"module {cls.__module__!r} has no attribute {name!r}")

AttributeError: module 'matplotlib' has no attribute 'xlabel'
```



In [34]:

```
categorical_columns = ['GRE Score', 'TOEFL Score', 'University Rating', 'CGPA', 'Research']
plt.figure(figsize=(12, 10))
for i, col in enumerate(categorical_columns, 1):
    plt.subplot(3, 2, i)
    sns.countplot(x=dataset1[col], hue=dataset1['Chance of admit(y/n)'])
    plt.title(col)
    plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [32]:

```
print(dataset1.columns)
```

```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'CGPA', 'Research',
       'Chance of admit(y/n)'],
      dtype='object')
```

In [ ]: