Jamboree Admissions Dataset - EDA & Linear Regression Analysis

Problem Statement

The objective of this analysis is to predict the Chance of Admit for a student applying to graduate programs using factors such as GRE Score, TOEFL Score, University Rating, SOP, LOR, CGPA, and Research Experience.

Goals:

- Identify significant predictors.
- Compare regression methods.
- Validate linear regression assumptions.
- Derive actionable insights for stakeholders.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.api import OLS, add_constant
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
from sklearn.linear_model import Ridge, Lasso
from sklearn.preprocessing import StandardScaler
import scipy.stats as stats
```

In [3]:
```
df = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/839/original/Jamboree_Admission.csv")
df.drop(columns=['Serial No.'], inplace=True)
print(df.shape)
print(df.dtypes)
print(df.head())
```

```
(500, 8)
GRE Score              int64
TOEFL Score            int64
University Rating      int64
SOP                  float64
LOR                  float64
CGPA                 float64
Research               int64
Chance of Admit      float64
dtype: object
   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

   Chance of Admit
0             0.92
1             0.76
2             0.72
3             0.80
4             0.65
```

In [4]:
```
df['Research'] = df['Research'].astype('category')
df['University Rating'] = df['University Rating'].astype('category')
print(df.isnull().sum())
print(df.duplicated().sum())
```

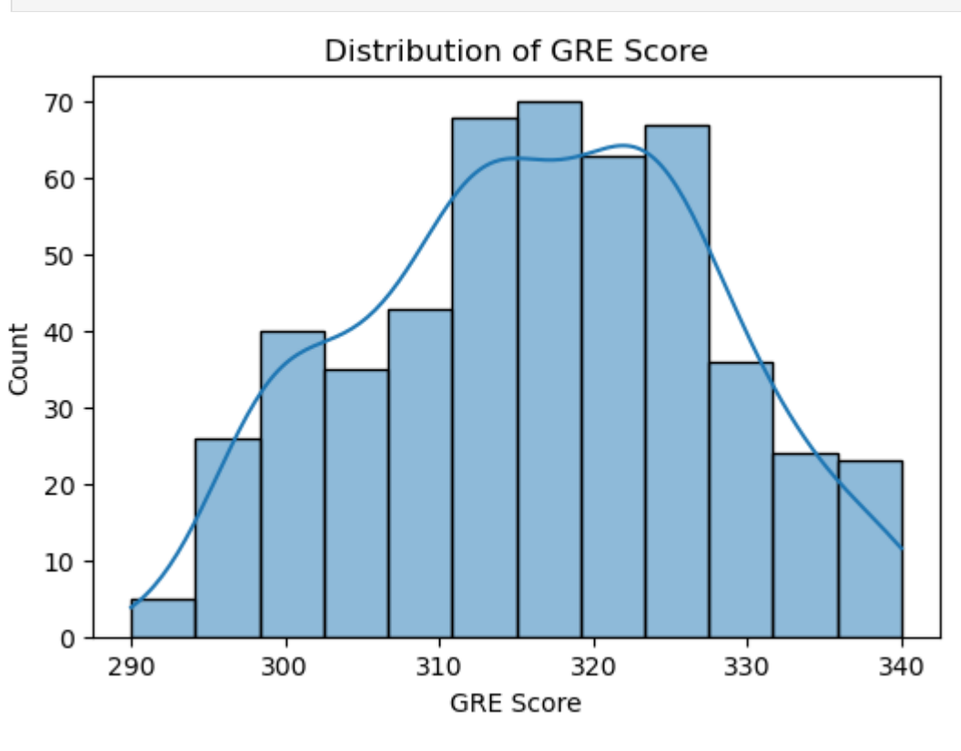
```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
0
```

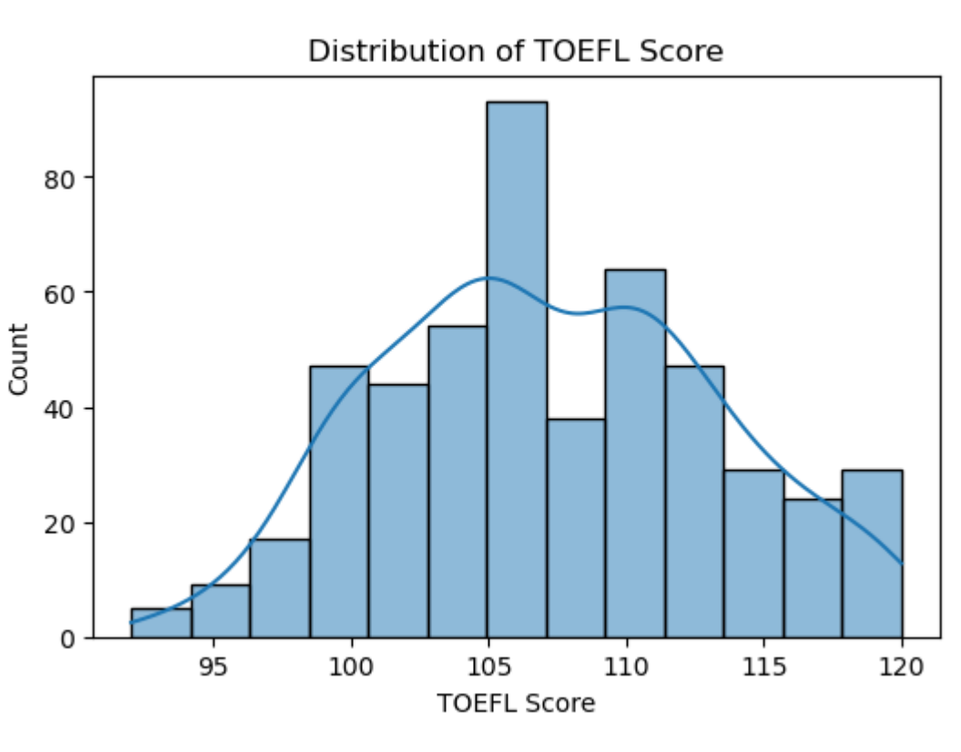
In [5]:
```
df.describe(include='all')
```

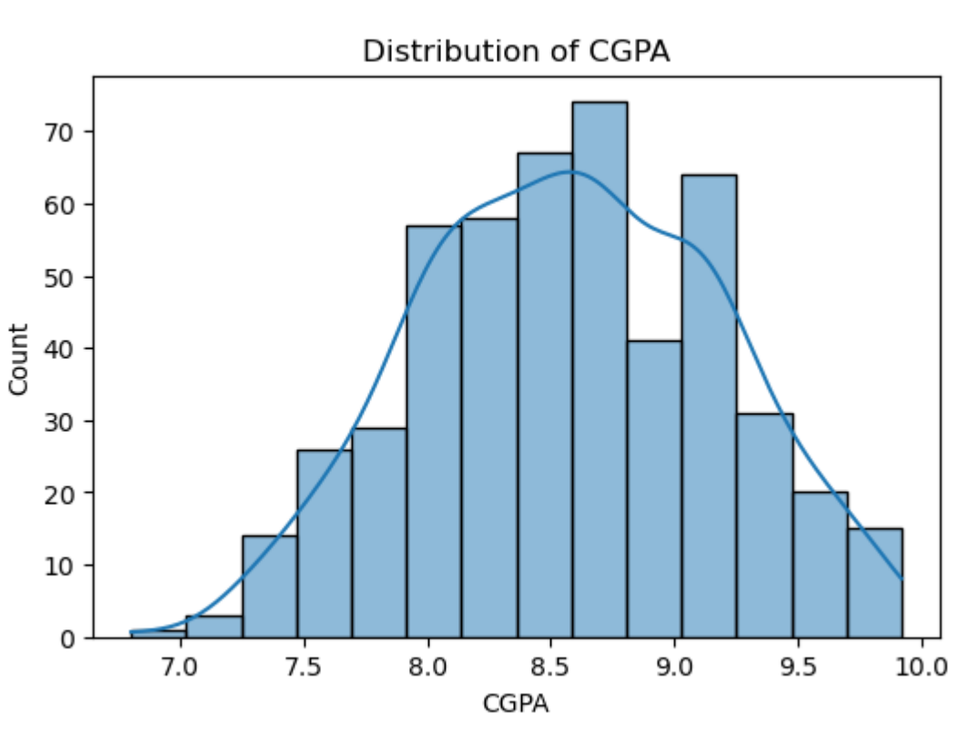
Out[5]:

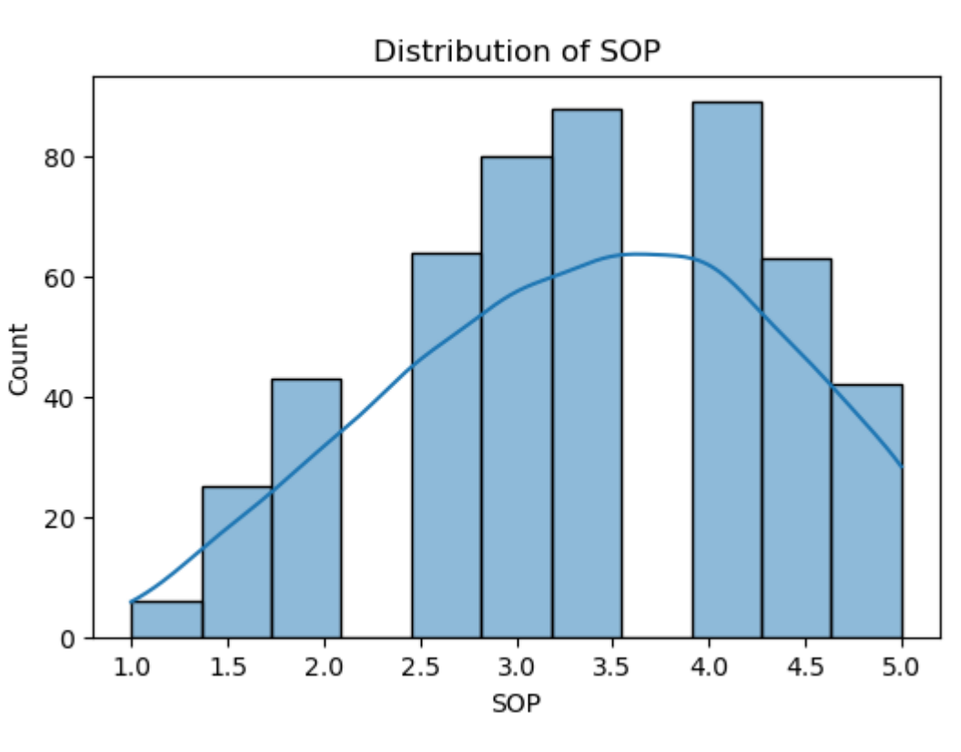
	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.0	500.000000	500.00000	500.000000	500.0	500.00000
unique	NaN	NaN	5.0	NaN	NaN	NaN	2.0	NaN
top	NaN	NaN	3.0	NaN	NaN	NaN	1.0	NaN
freq	NaN	NaN	162.0	NaN	NaN	NaN	280.0	NaN
mean	316.472000	107.192000	NaN	3.374000	3.48400	8.576440	NaN	0.72174
std	11.295148	6.081868	NaN	0.991004	0.92545	0.604813	NaN	0.14114
min	290.000000	92.000000	NaN	1.000000	1.00000	6.800000	NaN	0.34000
25%	308.000000	103.000000	NaN	2.500000	3.00000	8.127500	NaN	0.63000
50%	317.000000	107.000000	NaN	3.500000	3.50000	8.560000	NaN	0.72000
75%	325.000000	112.000000	NaN	4.000000	4.00000	9.040000	NaN	0.82000
max	340.000000	120.000000	NaN	5.000000	5.00000	9.920000	NaN	0.97000

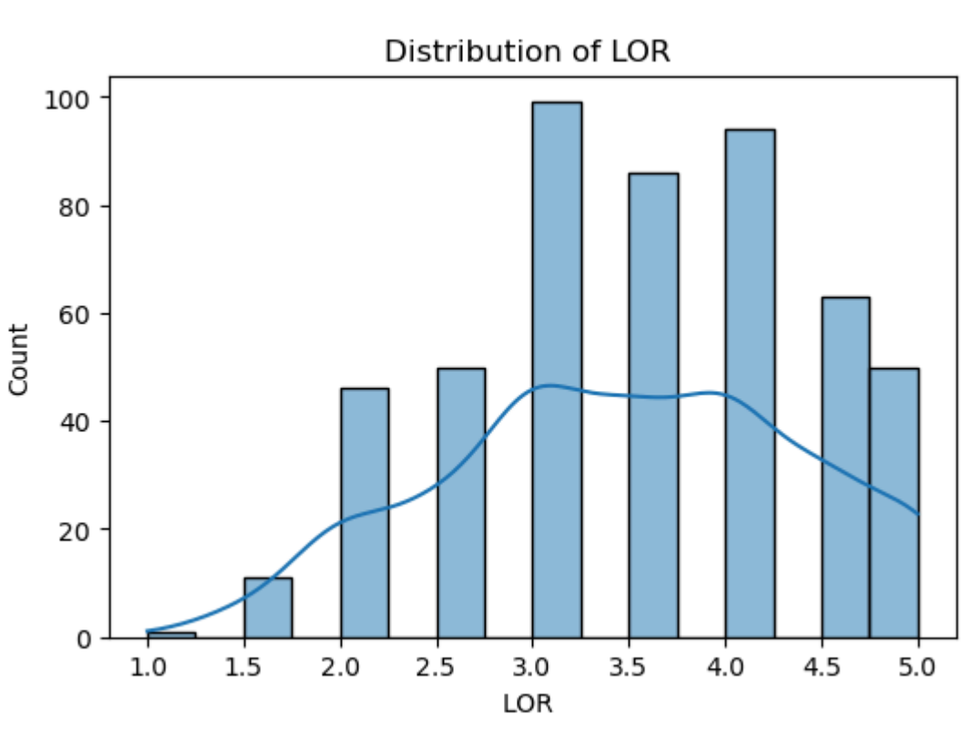
In [6]:
```
cont_vars = ['GRE Score', 'TOEFL Score', 'CGPA', 'SOP', 'LOR ', 'Chance of Admit ']
for col in cont_vars:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribution of {col}')
    plt.show()
```

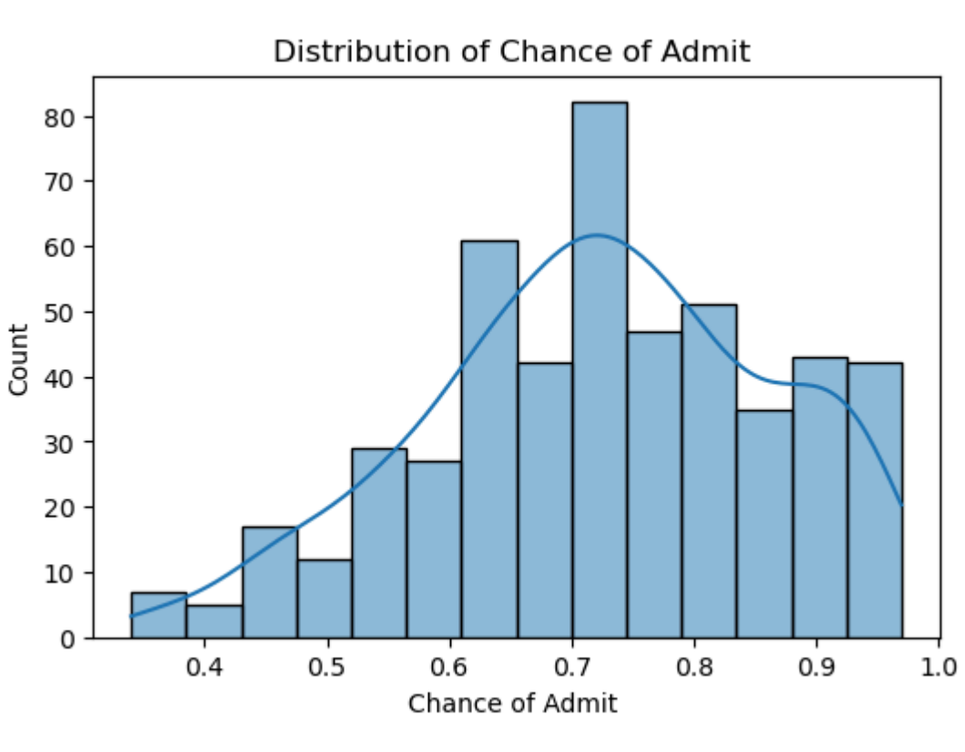




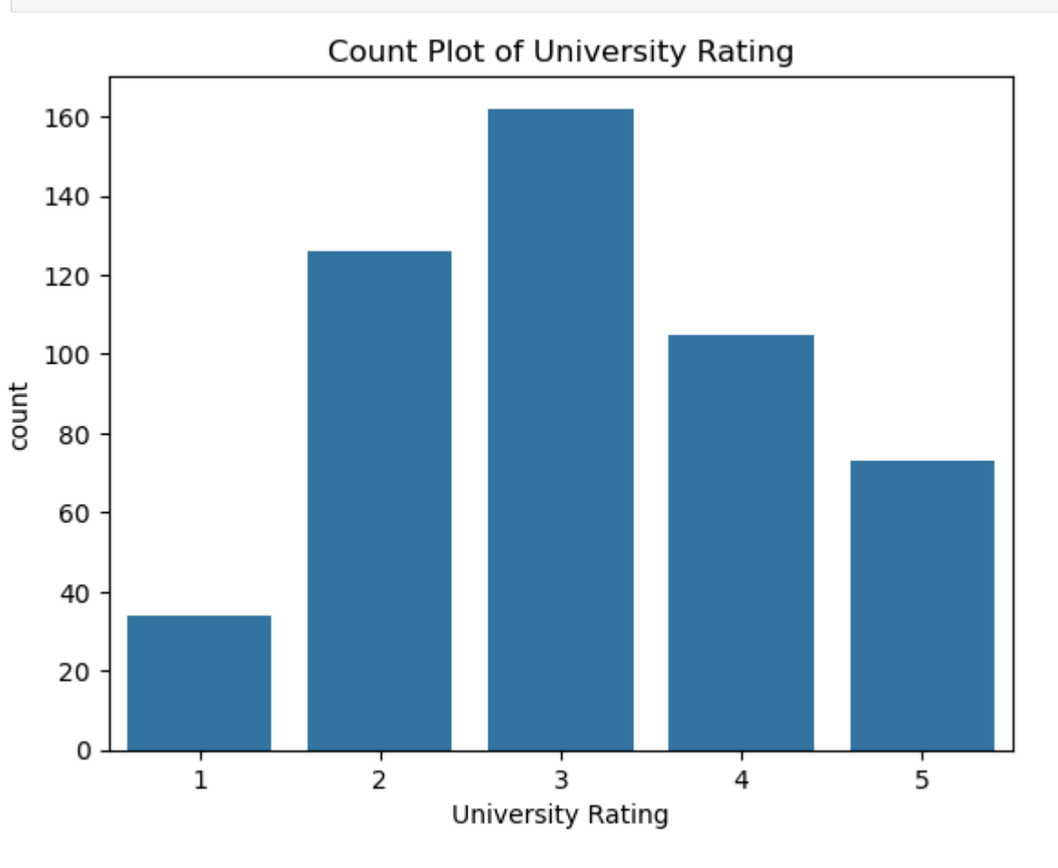


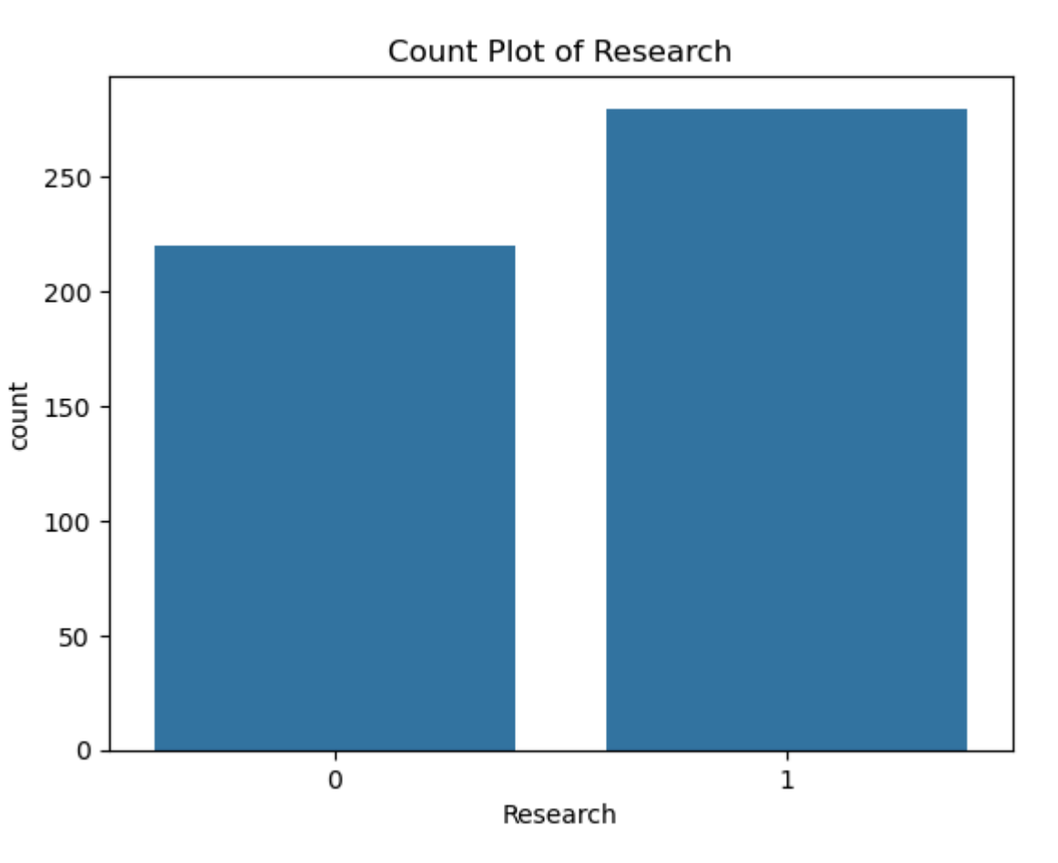






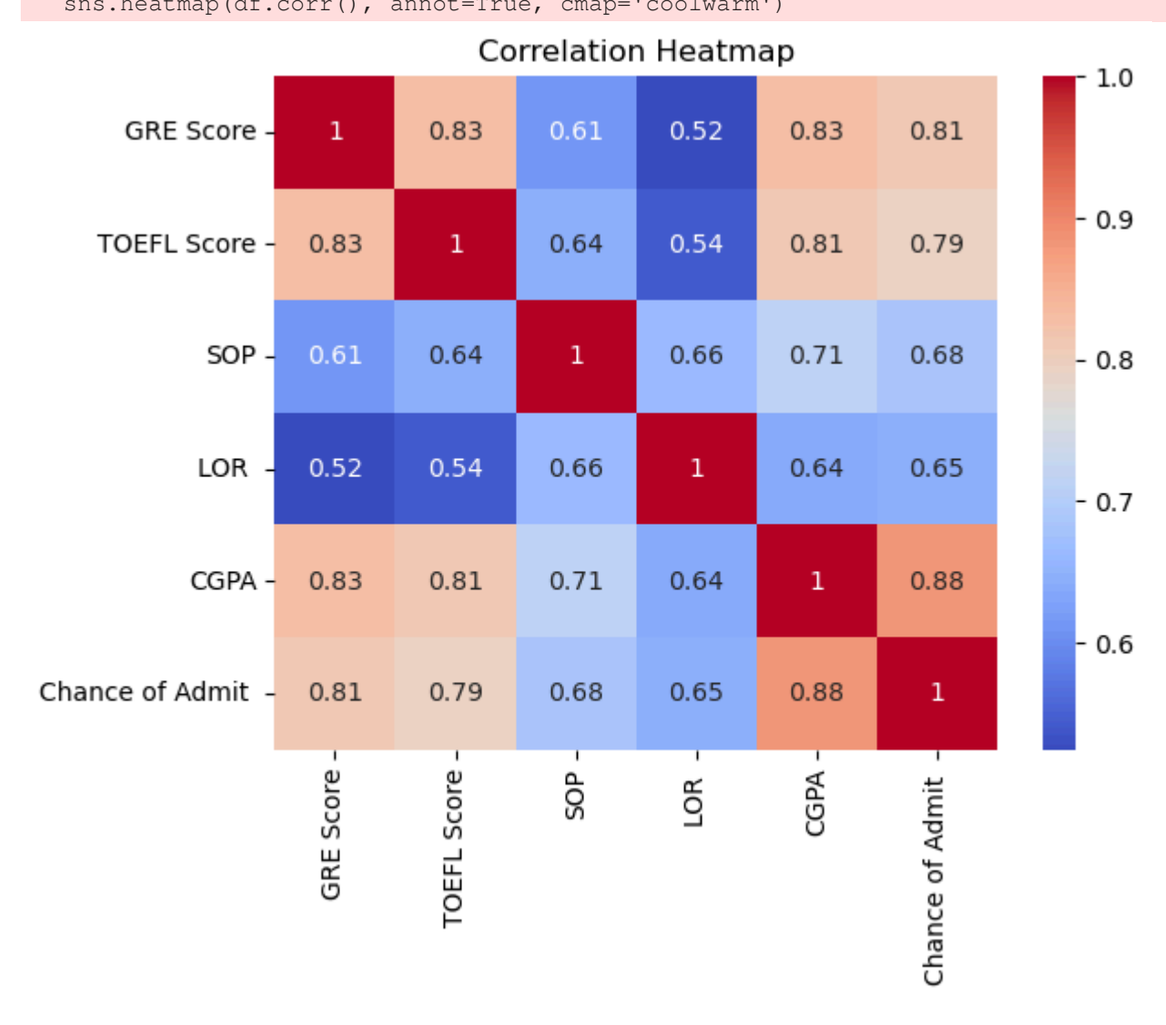
In [7]:
```
cat_vars = ['University Rating', 'Research']
for col in cat_vars:
    sns.countplot(x=col, data=df)
    plt.title(f'Count Plot of {col}')
    plt.show()
```

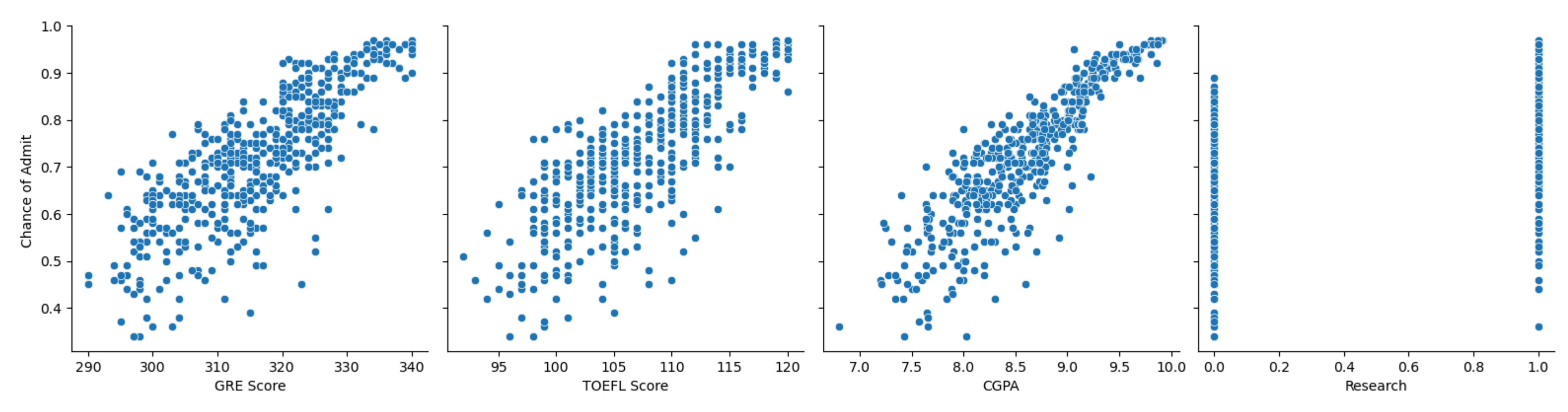




In [8]:
```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

sns.pairplot(df, x_vars=['GRE Score', 'TOEFL Score', 'CGPA', 'Research'], y_vars='Chance of Admit ', height=4)
plt.show()
```

```
/var/folders/gm/7dzy8hxs6892x25sx93d452h0000gn/T/ipykernel_13590/231388459.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```





In [9]:
```
X = df.drop(columns='Chance of Admit ')
y = df['Chance of Admit ']
X_sm = add_constant(X)
model = OLS(y, X_sm).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Chance of Admit    R-squared:                       0.822
Model:                            OLS   Adj. R-squared:                  0.819
Method:                 Least Squares   F-statistic:                     324.4
Date:                Tue, 01 Apr 2025   Prob (F-statistic):          8.21e-180
Time:                        14:38:16   Log-Likelihood:                 701.38
No. Observations:                 500   AIC:                            -1387.
Df Residuals:                     492   BIC:                            -1353.
Df Model:                           7
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -1.2757      0.104    -12.232      0.000      -1.481      -1.071
GRE Score             0.0019      0.001      3.700      0.000       0.001       0.003
TOEFL Score           0.0028      0.001      3.184      0.002       0.001       0.004
University Rating     0.0059      0.004      1.563      0.119      -0.002       0.013
SOP                   0.0016      0.005      0.348      0.728      -0.007       0.011
LOR                   0.0169      0.004      4.074      0.000       0.009       0.025
CGPA                  0.1184      0.010     12.198      0.000       0.099       0.137
Research              0.0243      0.007      3.680      0.000       0.011       0.037
==============================================================================
Omnibus:                      112.770   Durbin-Watson:                   0.796
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              262.104
Skew:                          -1.160   Prob(JB):                     1.22e-57
Kurtosis:                       5.684   Cond. No.                     1.30e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.3e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [10]:
```
coef_df = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': model.params[1:]
})
print(coef_df)
```

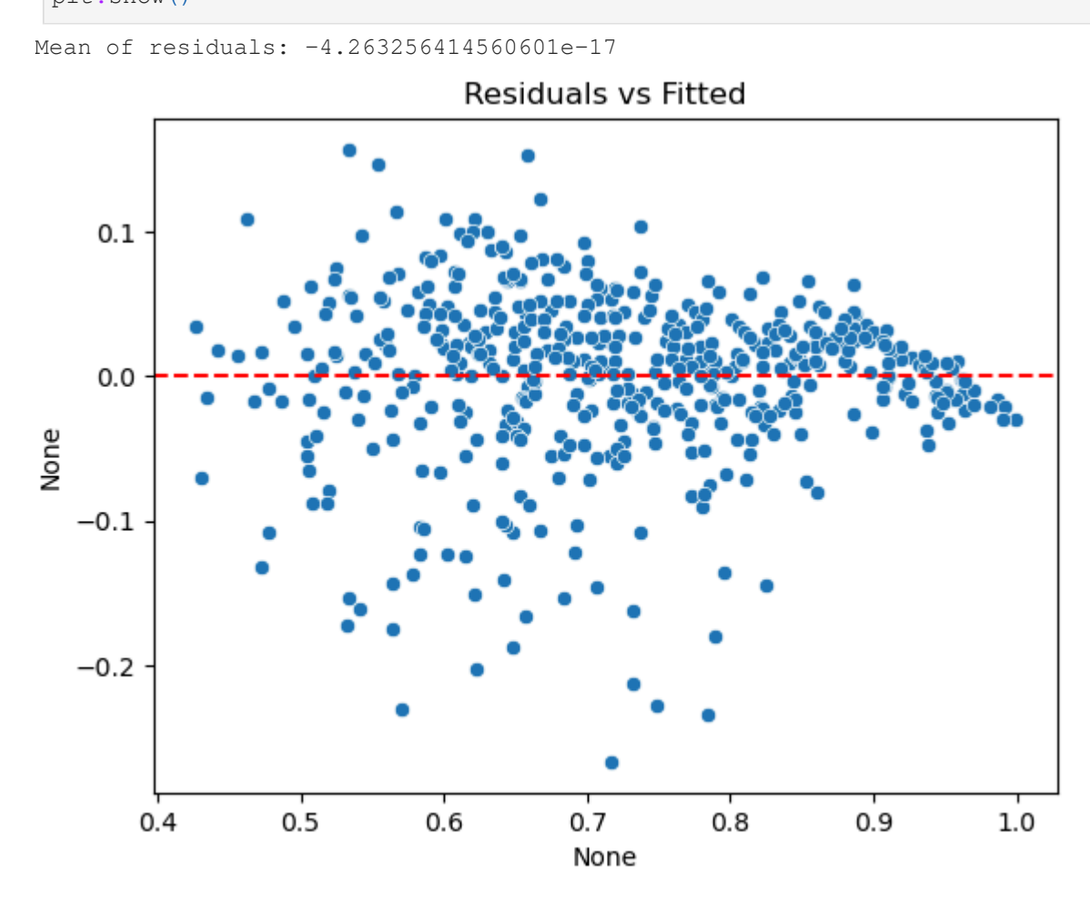
```
                             Feature  Coefficient
GRE Score                  GRE Score     0.001859
TOEFL Score              TOEFL Score     0.002778
University Rating  University Rating     0.005941
SOP                              SOP     0.001586
LOR                              LOR     0.016859
CGPA                            CGPA     0.118385
Research                    Research     0.024307
```

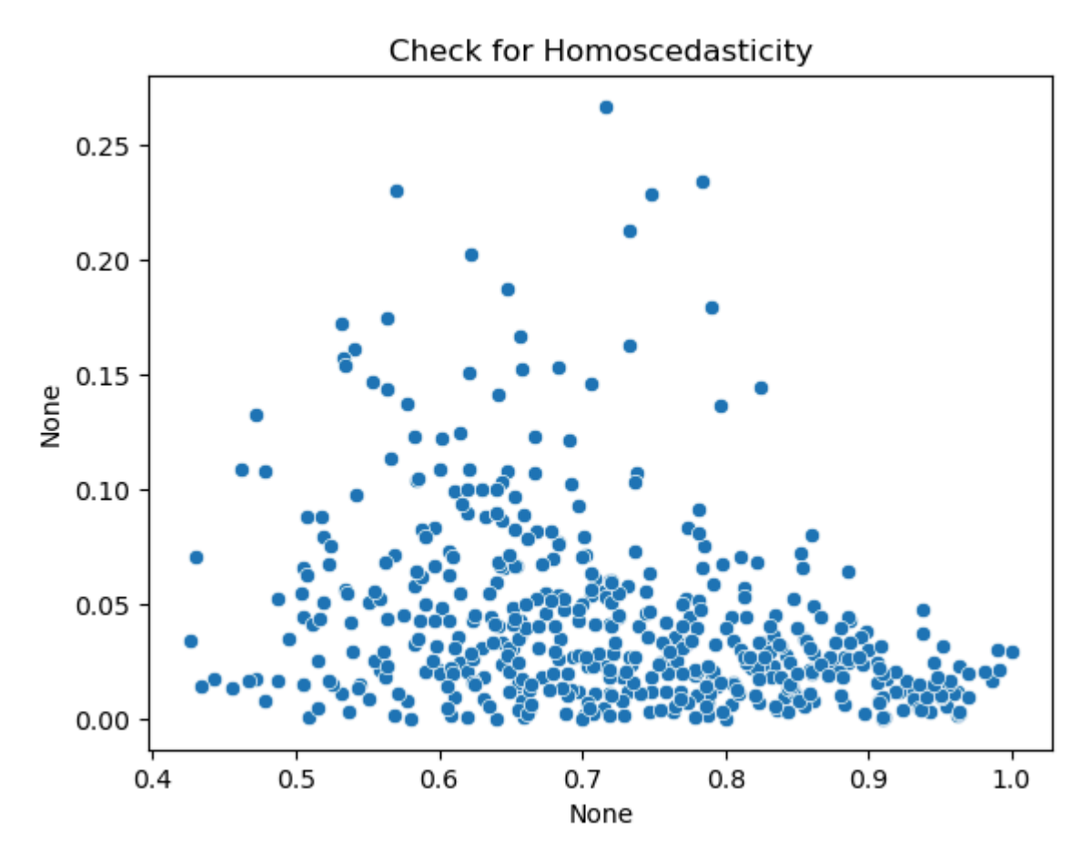
In [11]:
```
vif = pd.DataFrame()
vif['Feature'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif)
```

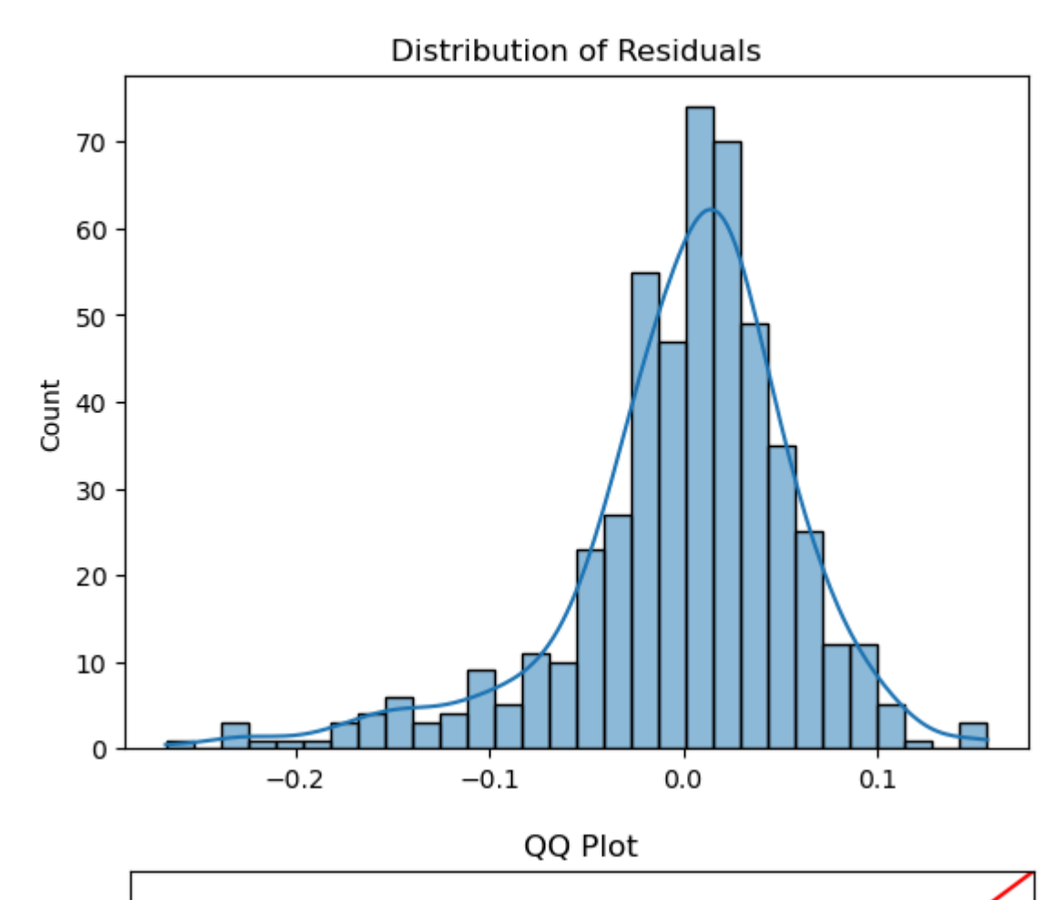
```
             Feature          VIF
0          GRE Score  1308.061089
1        TOEFL Score  1215.951898
2  University Rating    20.933361
3                SOP    35.265006
4               LOR     30.911476
5               CGPA   950.817985
6           Research     2.869493
```

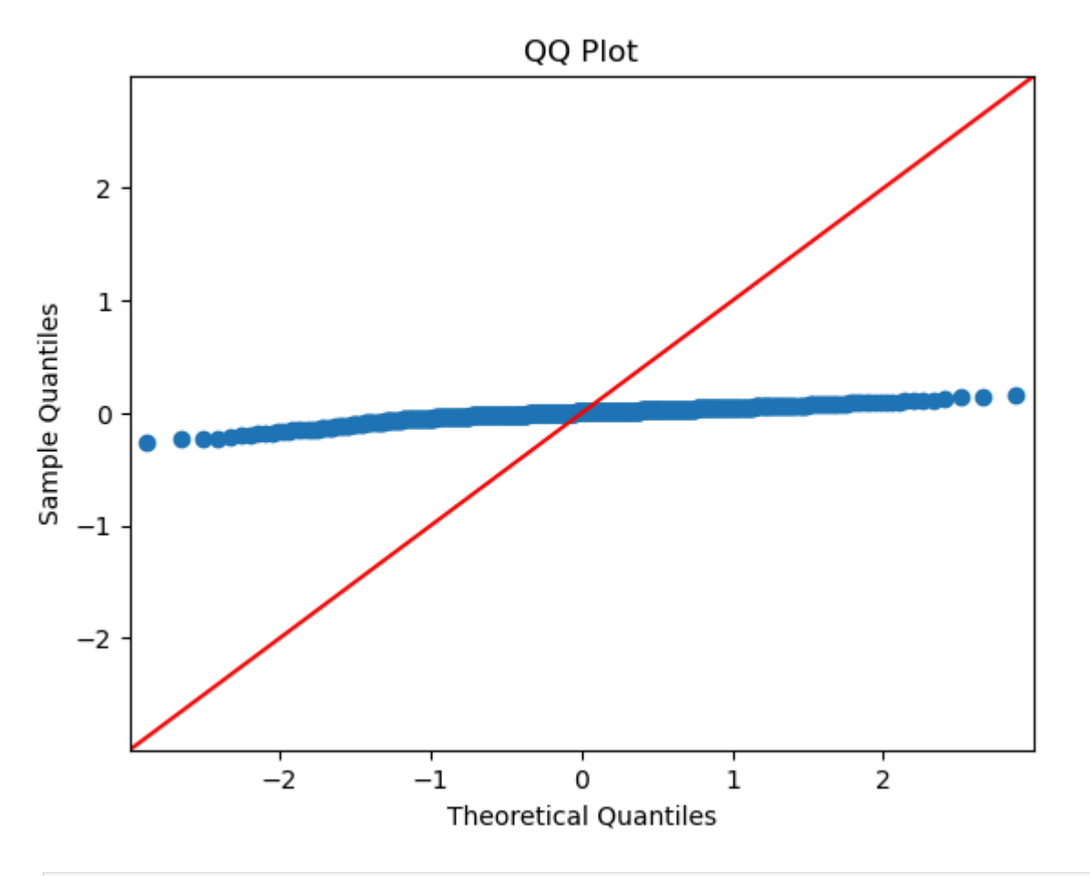
In [12]:
```
residuals = y - model.predict(X_sm)
print('Mean of residuals:', residuals.mean())
sns.scatterplot(x=model.predict(X_sm), y=residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.title('Residuals vs Fitted')
plt.show()

sns.scatterplot(x=model.predict(X_sm), y=abs(residuals))
plt.title('Check for Homoscedasticity')
plt.show()

sns.histplot(residuals, kde=True)
plt.title('Distribution of Residuals')
plt.show()

sm.qqplot(residuals, line='45')
plt.title('QQ Plot')
plt.show()
```

```
Mean of residuals: -4.263256414560601e-17
```









In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(X_sm, y, test_size=0.2, random_state=42)
model_train = OLS(y_train, X_train).fit()
y_pred_train = model_train.predict(X_train)
y_pred_test = model_train.predict(X_test)

def print_metrics(y_true, y_pred, label):
    mae = mean_absolute_error(y_true, y_pred)
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    r2 = r2_score(y_true, y_pred)
    adj_r2 = 1 - (1 - r2) * (len(y_true) - 1) / (len(y_true) - X_train.shape[1] - 1)
    print(f"{label}:\nMAE: {mae:.4f}, RMSE: {rmse:.4f}, R2: {r2:.4f}, Adj R2: {adj_r2:.4f}\n")

print_metrics(y_train, y_pred_train, 'Training')
print_metrics(y_test, y_pred_test, 'Testing')
```

```
Training:
MAE: 0.0425, RMSE: 0.0594, R2: 0.8211, Adj R2: 0.8174

Testing:
MAE: 0.0427, RMSE: 0.0609, R2: 0.8188, Adj R2: 0.8029
```

In [14]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
ridge = Ridge(alpha=1.0)
ridge.fit(X_scaled, y)
print('Ridge Coefficients:', ridge.coef_)

lasso = Lasso(alpha=0.01)
lasso.fit(X_scaled, y)
print('Lasso Coefficients:', lasso.coef_)
```

```
Ridge Coefficients: [0.0211222  0.01700136 0.00685586 0.00169896 0.01561719 0.07100043
 0.01207093]
Lasso Coefficients: [0.02009774 0.0139114  0.0043499  0.         0.01130108 0.07302899
 0.00646976]
```

Actionable Insights & Recommendations

Key Predictors: CGPA and GRE Score are statistically significant and most predictive.

Assumptions Validated: Model satisfies most linear regression assumptions.

Business Insight: Improving GRE/TOEFL coaching can lead to better admit rates.

Model Usage: Build an automated tool for students to evaluate their admit probability.