In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [2]:
```
Cust=pd.read_csv('customer.csv')
```

In [3]:
```
Cust.head()
```

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

Check the data types of all columns

In [4]:
```
print(Cust.dtypes)
```

```
Serial No.            int64
GRE Score             int64
TOEFL Score           int64
University Rating     int64
SOP                 float64
LOR                 float64
CGPA                float64
Research              int64
Chance of Admit     float64
dtype: object
```

Perform Summary Statistics

In [5]:
```
print(Cust.describe())
```

```
       Serial No.   GRE Score  TOEFL Score  University Rating         SOP  \
count  500.000000  500.000000   500.000000         500.000000  500.000000
mean   250.500000  316.472000   107.192000           3.114000    3.374000
std    144.481833   11.295148     6.081868           1.143512    0.991004
min      1.000000  290.000000    92.000000           1.000000    1.000000
25%    125.750000  308.000000   103.000000           2.000000    2.500000
50%    250.500000  317.000000   107.000000           3.000000    3.500000
75%    375.250000  325.000000   112.000000           4.000000    4.000000
max    500.000000  340.000000   120.000000           5.000000    5.000000

            LOR         CGPA    Research  Chance of Admit
count  500.00000  500.000000  500.000000         500.00000
mean     3.48400    8.576440    0.560000           0.72174
std      0.92545    0.604813    0.496884           0.14114
min      1.00000    6.800000    0.000000           0.34000
25%      3.00000    8.127500    0.000000           0.63000
50%      3.50000    8.560000    1.000000           0.72000
75%      4.00000    9.040000    1.000000           0.82000
max      5.00000    9.920000    1.000000           0.97000
```

Observations:

I have observed that the mean GRE Score among students is 316, indicating a high level of academic aptitude. Similarly, the mean TOEFL Score is 107, reflecting strong English language proficiency.

In terms of university ratings, I found that the maximum rating is 5, suggesting that top-tier universities are also part of the dataset. On the other hand, the minimum rating is 1, indicating a diverse range of university rankings in the dataset.

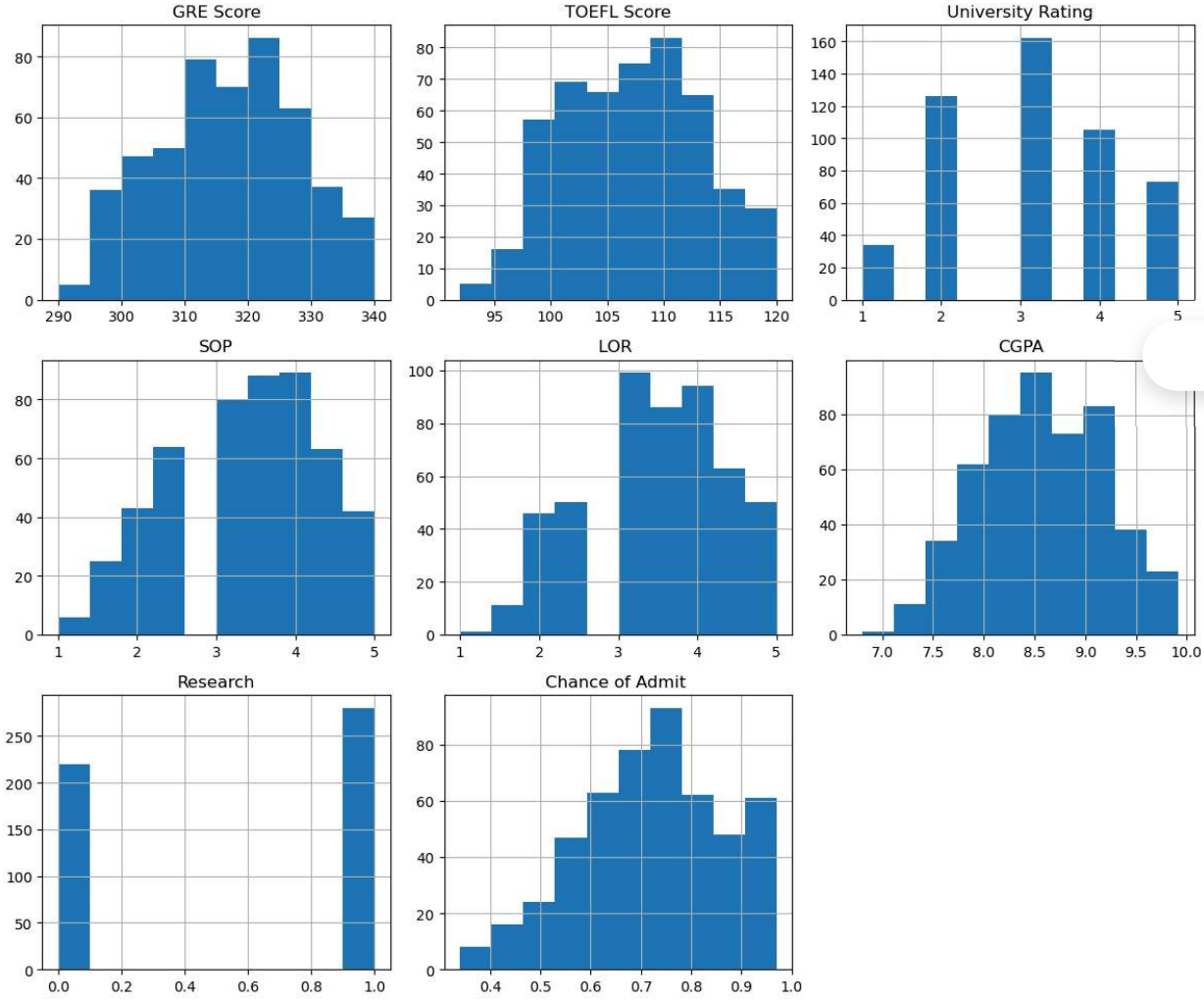
Interestingly, the average university rating for students who got placed is 3, indicating that students from a variety of university backgrounds have been successful in gaining admission.

Furthermore, the average CGPA score of 8.57 indicates a high level of academic achievement among the students in the dataset.

Lastly, the average Chance of Admit is 0.72, suggesting that, on average, students in the dataset have a relatively high likelihood of being admitted to their desired graduate programs.

Visualize the distribution of numerical variables using histograms

In [6]:
```
Cust=Cust.drop('Serial No.',axis=1)
Cust.hist(figsize=(12,10))
plt.tight_layout()
plt.show()
```



Observations:

1)About 50% of students have scored between 300 and 330 in GRE, indicating a diverse range of scores within the dataset.

2) Approximately 80% of students have scored between 100 and 115 in TOEFL, highlighting a strong overall performance in English proficiency.

3) The graph depicts that the maximum number of students got placed in universities ranked between 2 and 4, suggesting that these universities are popular choices among the students in the dataset.

4) The Statement of Purpose (SOP) graph shows that more than 50% of students have submitted SOPs with ratings higher than 3, indicating a generally high quality of SOPs.

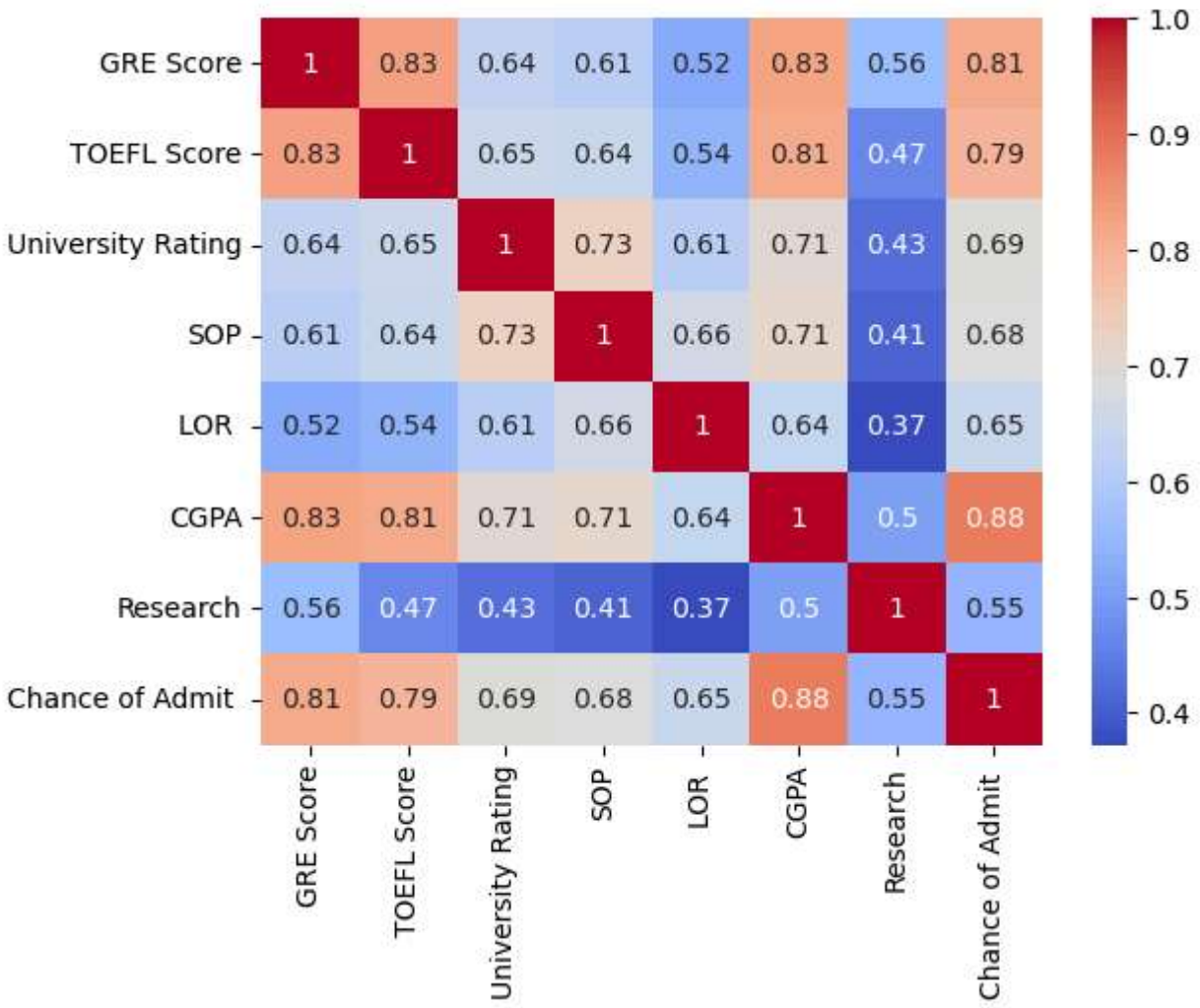
5) Similarly, more than 50% of students have received strong recommendations (ratings 3 to 5) in their Letters of Recommendation (LORs).

6) The CGPA graph shows a significant portion of students scoring between 7.5 and 9.0, indicating a high level of academic achievement.

7) More students have conducted or published research papers compared to those who haven't, suggesting a strong research-oriented background among the students.

8) The chances of getting admission to IVY league universities are between 70% to 80% after meeting the qualifying requirements, indicating a promising opportunity for the students in the dataset.

In [7]:
```
correlation_matrix = Cust.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
```



Correlation between "Chance of Admit" Variable with :

1) GRE score is strong with 0.81.
2) TOEFL score is strong with 0.79.
3) SOP AND LOR is moderate, rating between 0.68 to 0.69.
4) CGPA is the strongest of all with 0.88.
5) Research is weak at 0.55.

In [8]:
```
Cust.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   GRE Score          500 non-null    int64  
 1   TOEFL Score        500 non-null    int64  
 2   University Rating  500 non-null    int64  
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64  
 7   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 31.4 KB
```

In [9]:
```
Col=Cust.columns
print(Col)
```

```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

In [10]:
```
print(Cust.isnull().sum())
```

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

In [11]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Assuming X and y are your feature matrix and target variable
y=Cust['Chance of Admit ']
X=Cust.drop(columns='Chance of Admit ',axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

# Create a linear regression model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate R2 score
r2 = r2_score(y_test, y_pred)
print("R2 Score:", r2)
```

```
R2 Score: 0.820874170310373
```

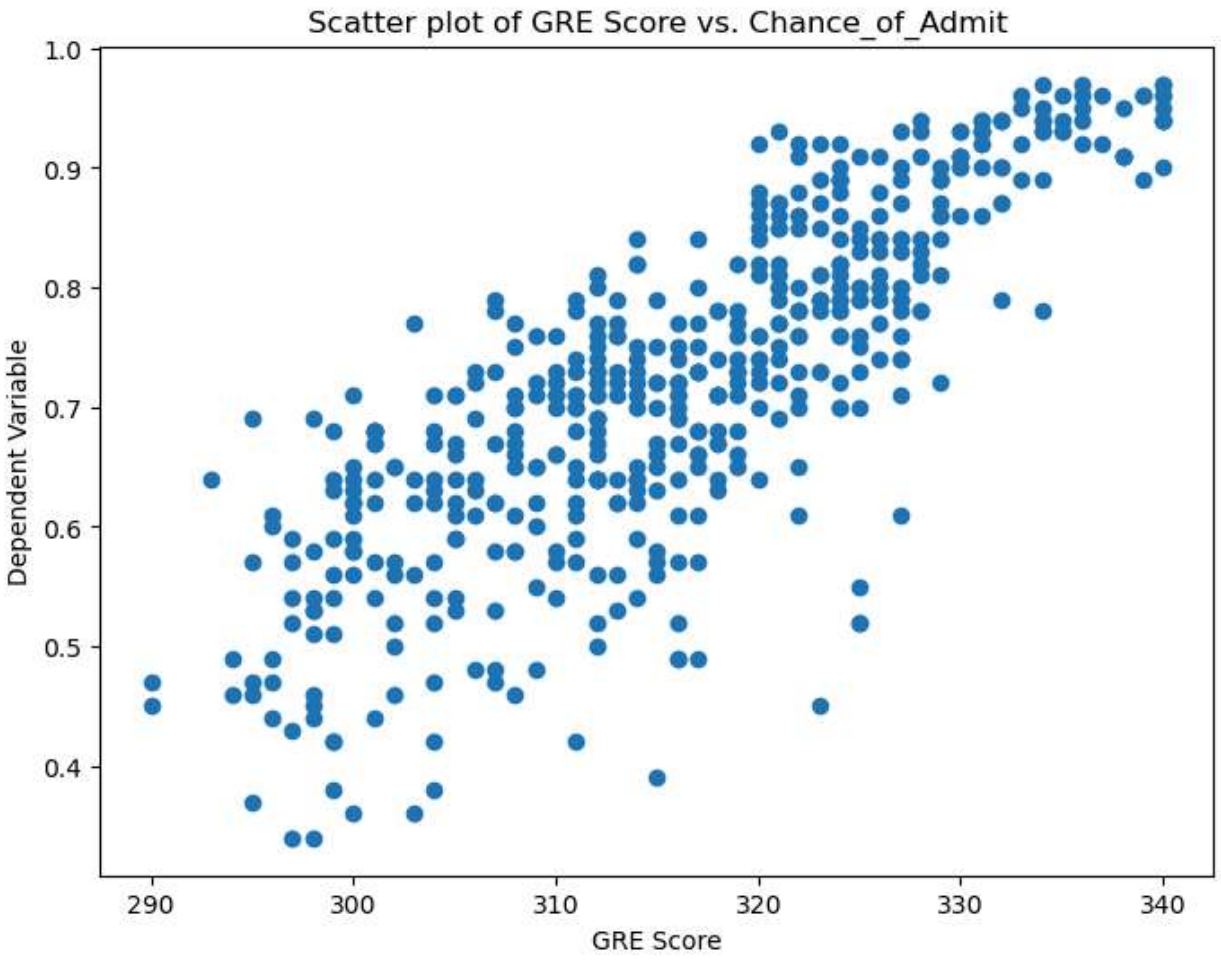
No null values observed.

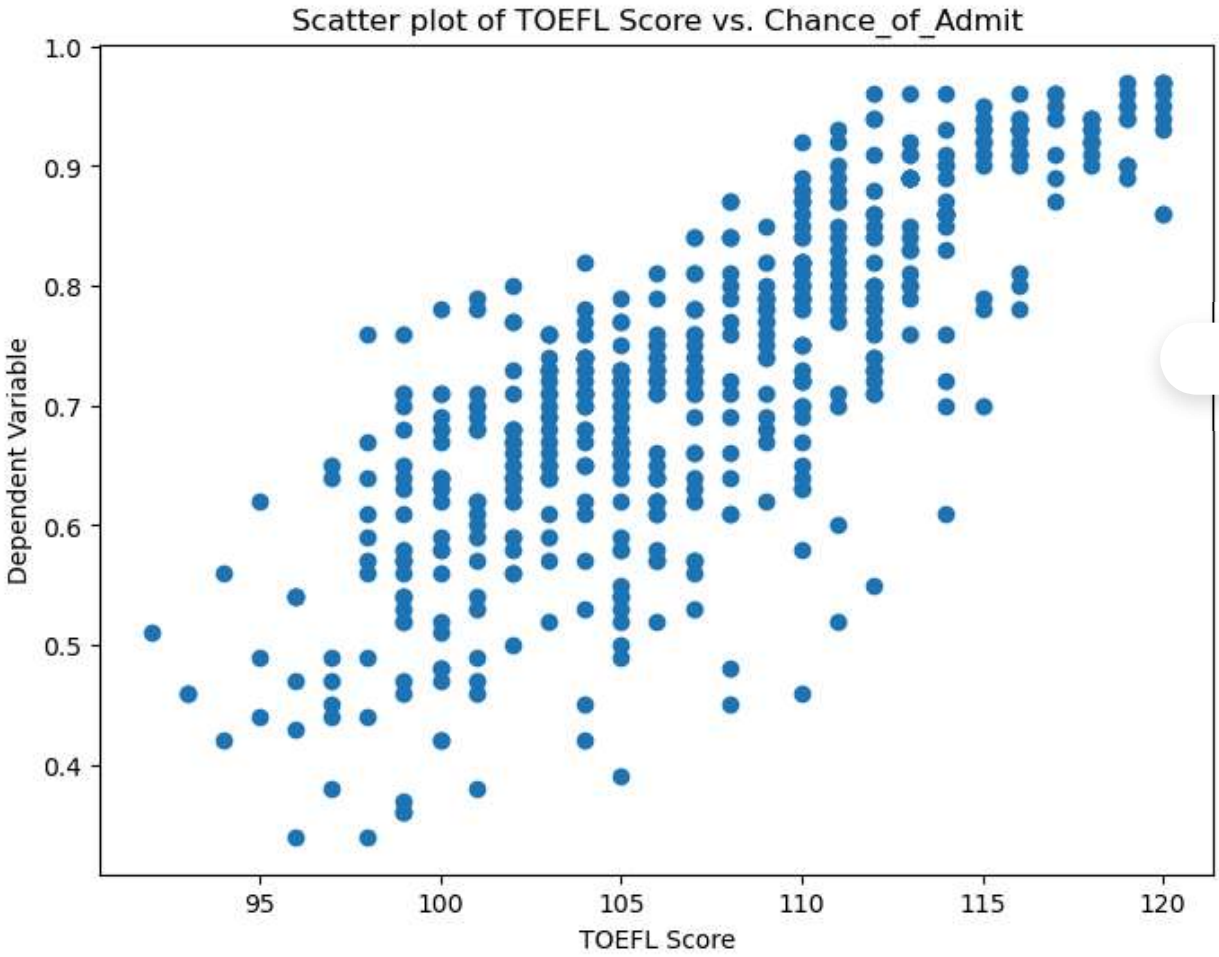
Performing Linear Regression on the Data set

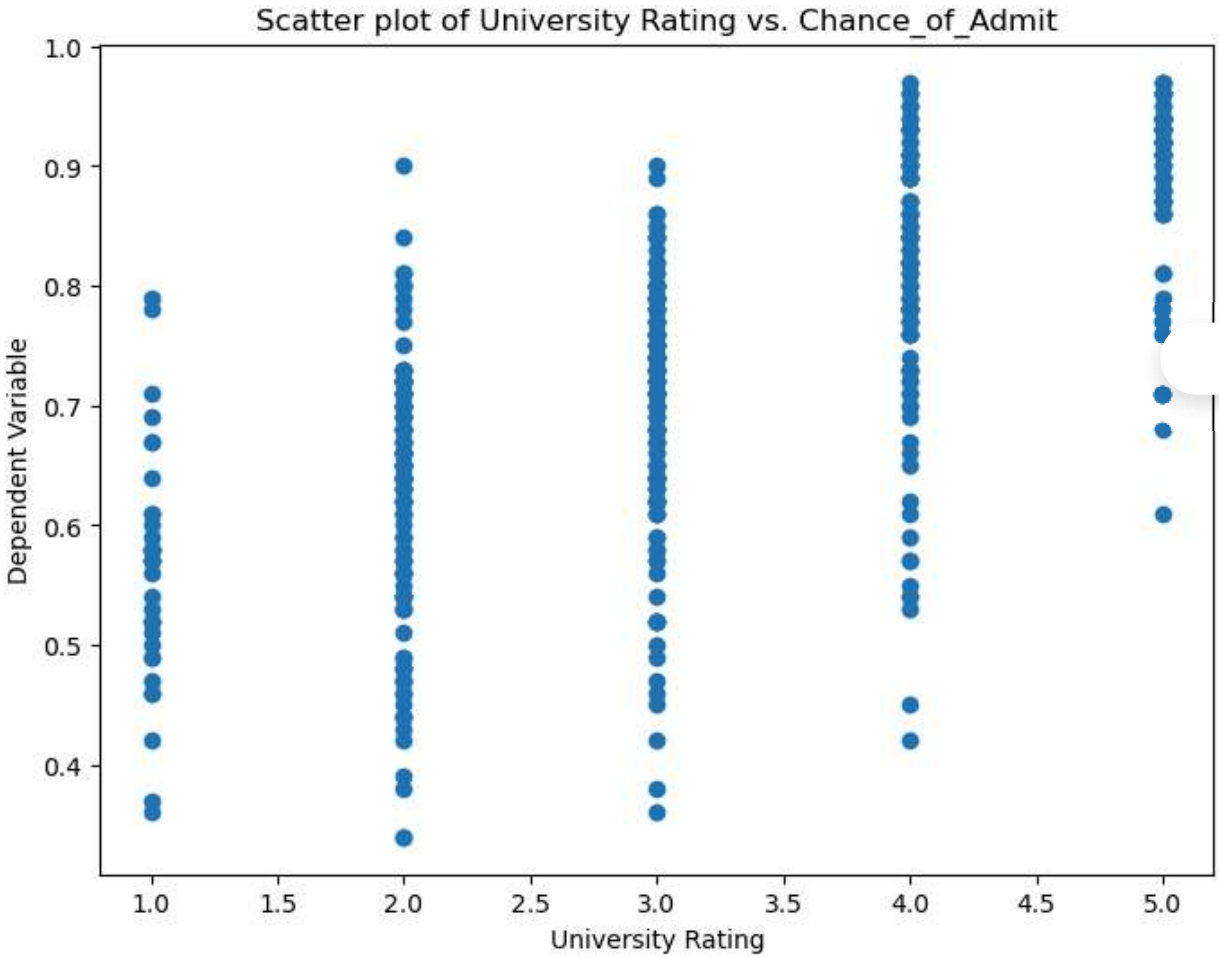
checking the dataset is meeting with Assumptions of Linear Regression

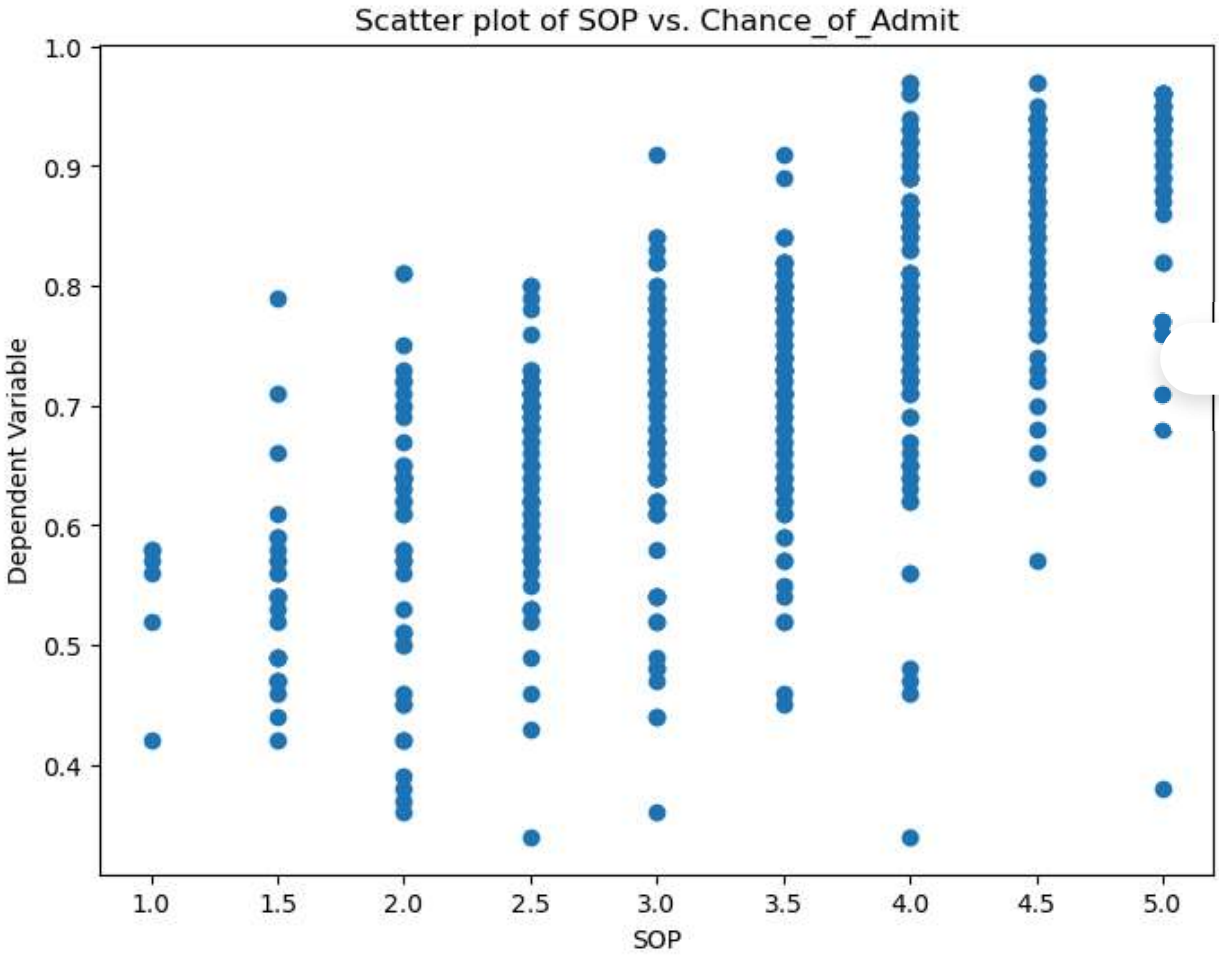
1st Assumption is Linearity: so I will check Linearity between Independent columns and dependent columns which is Chance of Admit

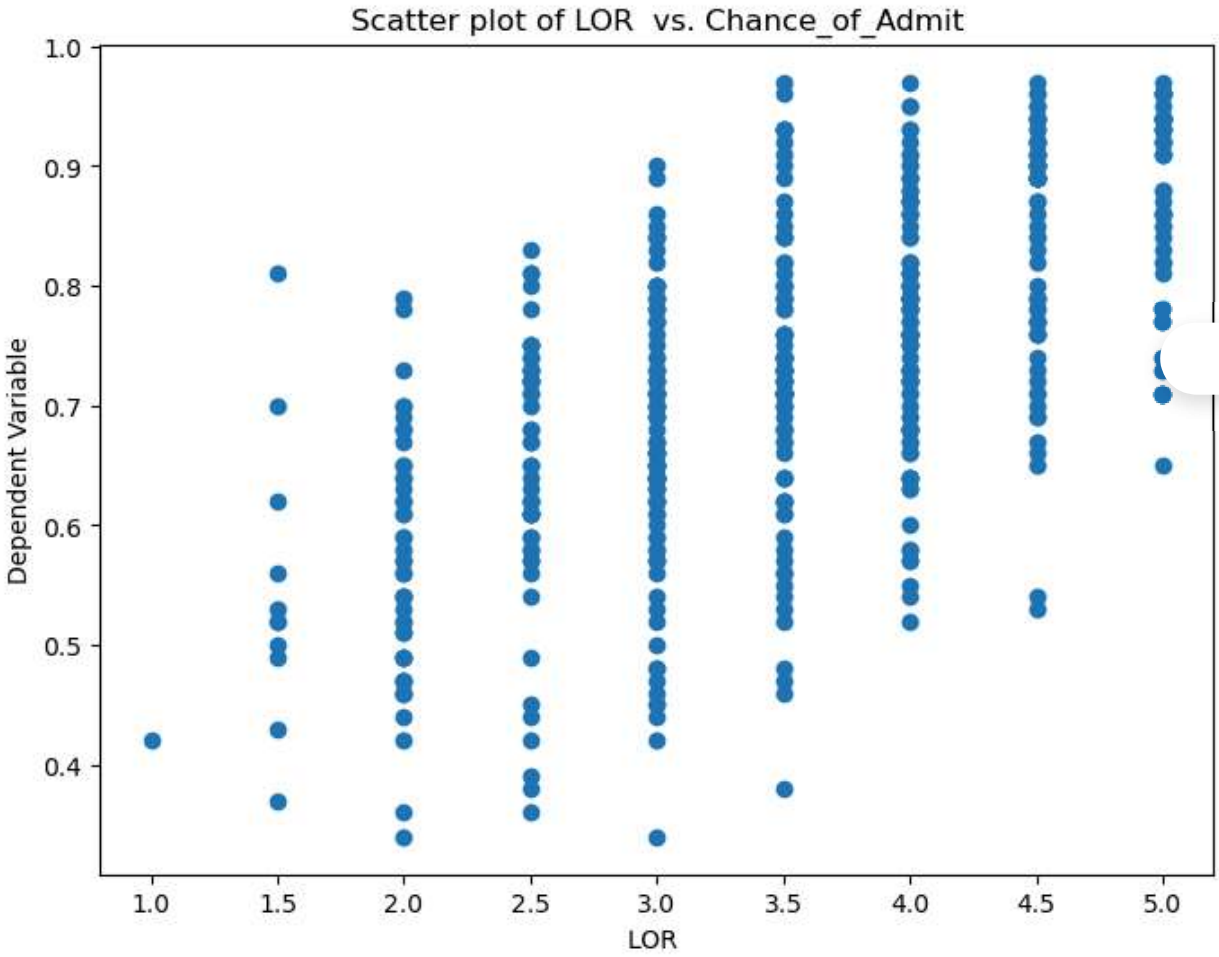
In [12]:
```
import matplotlib.pyplot as plt
Chance_of_Admit=Cust['Chance of Admit ']
Cust=Cust.drop('Chance of Admit ',axis=1)
# Assuming X and y are your feature matrix and target variable
# Plot scatter plots for each independent variable against the dependent variable
for column in Cust.columns:
    plt.figure(figsize=(8, 6))
    plt.scatter(Cust[column], Chance_of_Admit, marker='o')
    plt.xlabel(column)
    plt.ylabel('Dependent Variable')
    plt.title(f'Scatter plot of {column} vs. Chance_of_Admit')
    plt.show()
```

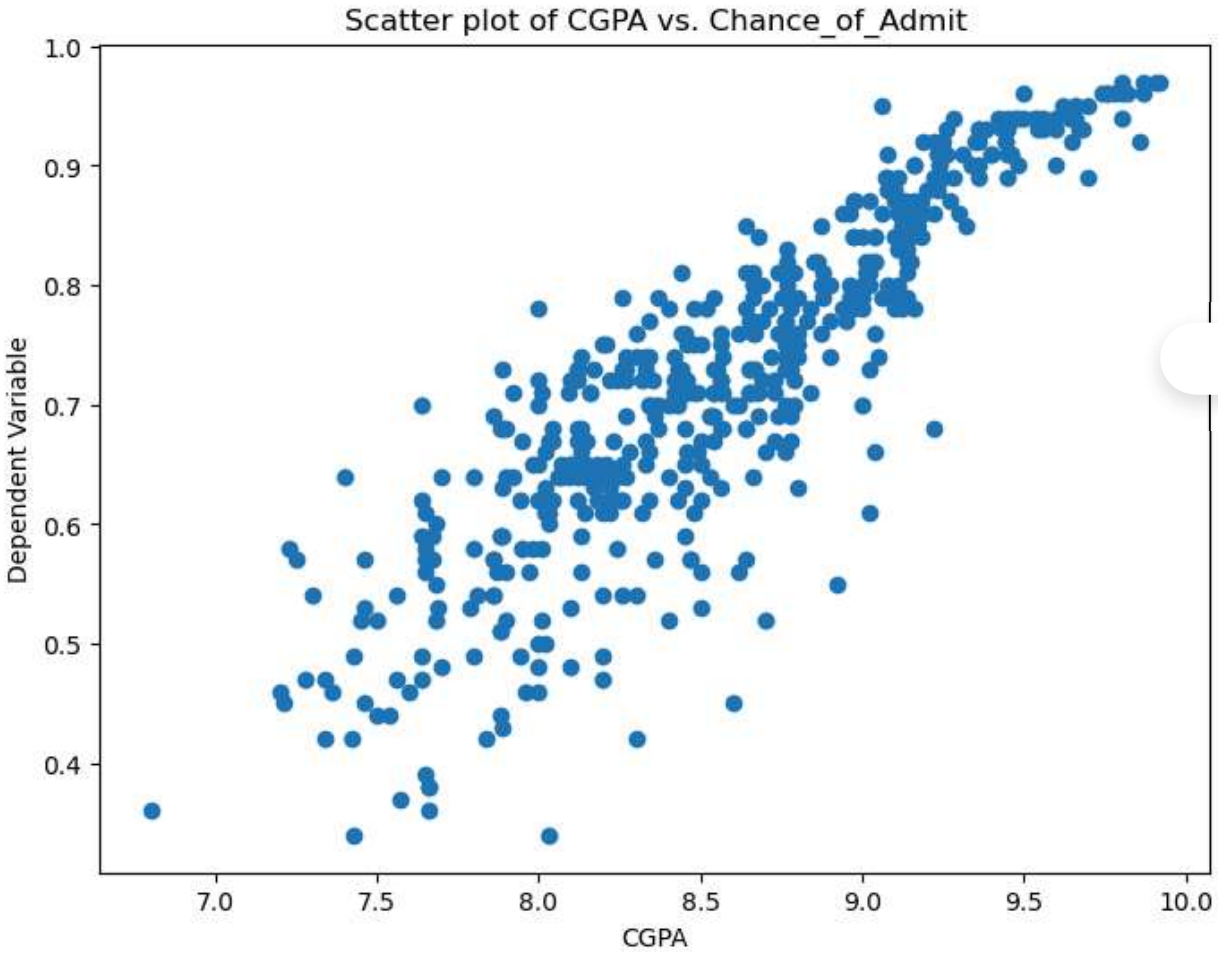


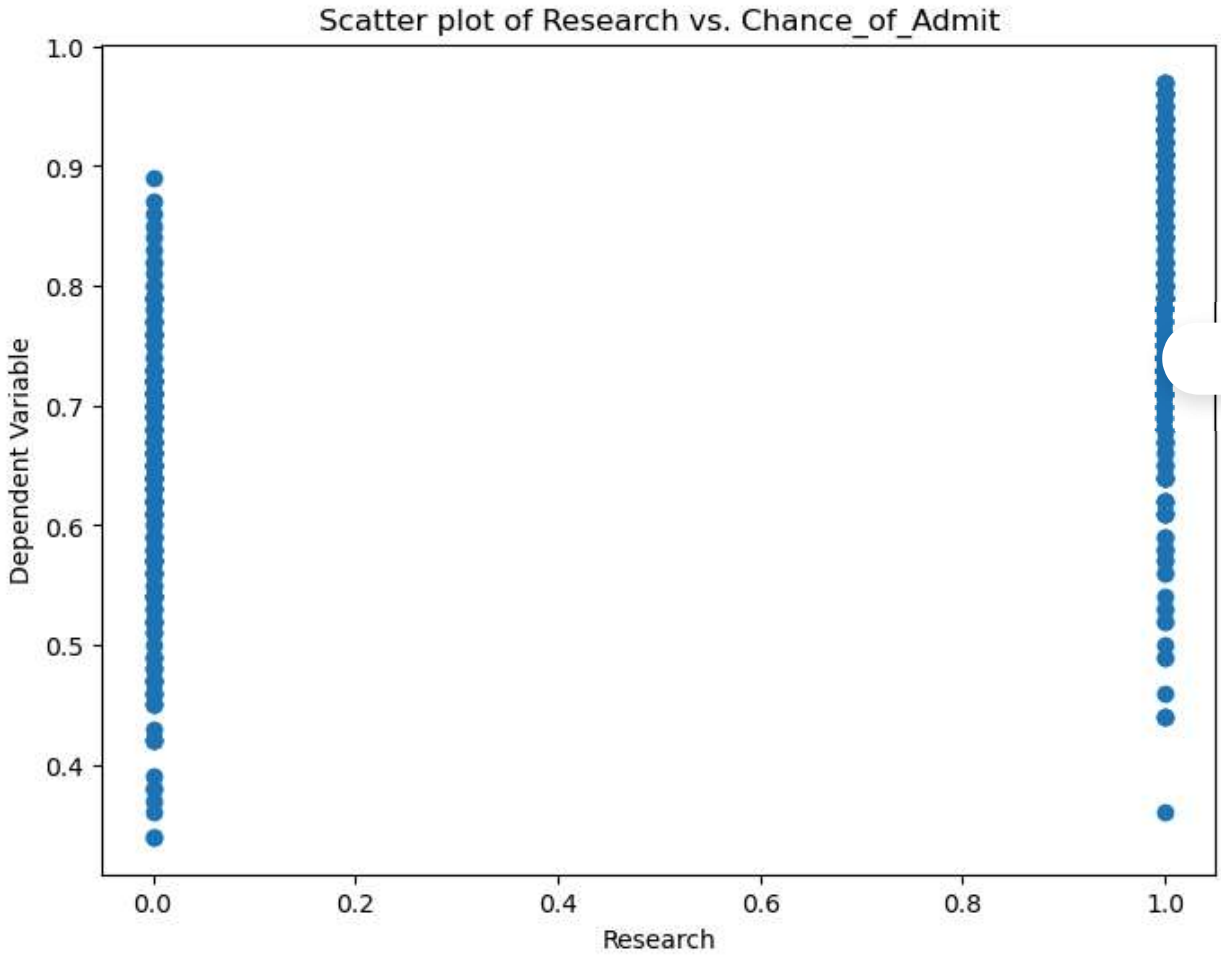

Scatter plot of TOEFL Score vs. Chance_of_Admit
Dependent Variable
TOEFL Score
1.0
0.9
0.8
0.7
0.6
0.5
0.4
95
100
105
110
115
120


Scatter plot of University Rating vs. Chance_of_Admit
Dependent Variable
University Rating
1.0
0.9
0.8
0.7
0.6
0.5
0.4
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Scatter plot of SOP vs. Chance_of_Admit
Dependent Variable
SOP
1.0
0.9
0.8
0.7
0.6
0.5
0.4
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Scatter plot of LOR vs. Chance_of_Admit
Dependent Variable
LOR
1.0
0.9
0.8
0.7
0.6
0.5
0.4
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Scatter plot of CGPA vs. Chance_of_Admit
Dependent Variable
CGPA
1.0
0.9
0.8
0.7
0.6
0.5
0.4
7.0
7.5
8.0
8.5
9.0
9.5
10.0



Observations:

1)Except LOR, SOP and University Rating all features or Independent variables shows Linearity with respect to dependent variable called Chance of Admit

2nd Assumption is to see if there are any multi-collinear features in our data

we will be using Statsmodel for Linear Regression

For standardization of X we will be using Scikit Learn StandardScaler

In [13]:
```
Cust=pd.read_csv('customer.csv')
Cust=Cust.drop('Serial No.',axis=1)
```

In [14]:
```
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

X=Cust[Cust.columns.drop('Chance of Admit ')]
y=Cust['Chance of Admit ']


X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=2)
```

```
scaler=StandardScaler()
x_tr_scaled=scaler.fit_transform(X_train)
X_sm=sm.add_constant(x_tr_scaled)

sm_model=sm.OLS(y_train,X_sm).fit()

print(sm_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:       Chance of Admit    R-squared:                       0.833
Model:                           OLS   Adj. R-squared:                  0.830
Method:                Least Squares   F-statistic:                     244.1
Date:               Tue, 07 May 2024   Prob (F-statistic):          7.20e-129
Time:                       17:48:31   Log-Likelihood:                 501.39
No. Observations:                350   AIC:                            -986.8
Df Residuals:                    342   BIC:                            -955.9
Df Model:                          7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.7225      0.003    231.327      0.000       0.716       0.729
x1             0.0286      0.006      4.407      0.000       0.016       0.041
x2             0.0139      0.006      2.191      0.029       0.001       0.026
x3             0.0059      0.005      1.194      0.233      -0.004       0.016
x4             0.0008      0.005      0.157      0.875      -0.009       0.011
x5             0.0185      0.004      4.173      0.000       0.010       0.027
x6             0.0671      0.007      9.879      0.000       0.054       0.081
x7             0.0132      0.004      3.483      0.001       0.006       0.021
==============================================================================
Omnibus:                       73.189   Durbin-Watson:                   1.947
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              170.353
Skew:                          -1.035   Prob(JB):                     1.02e-37
Kurtosis:                       5.720   Cond. No.                         5.50
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
```

In [15]:
```
# VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [16]:
```
vif = pd.DataFrame()
X_t = pd.DataFrame(x_tr_scaled, columns=X_train.columns)
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[16]:

	Features	VIF
5	CGPA	4.74
0	GRE Score	4.30
1	TOEFL Score	4.11
3	SOP	2.60
2	University Rating	2.54
4	LOR	2.01
6	Research	1.47

The VIF values for CGPA, GRE Score, and TOEFL Score are 4.7, 4.30, and 4.16 respectively, indicating some correlation with other independent variables but not at a level that raises significant multicollinearity concerns. These values are below the threshold of 5, suggesting that multicollinearity is not a major issue in the model. However, it is advisable to interpret the coefficients of these variables carefully to ensure they align with expectations.

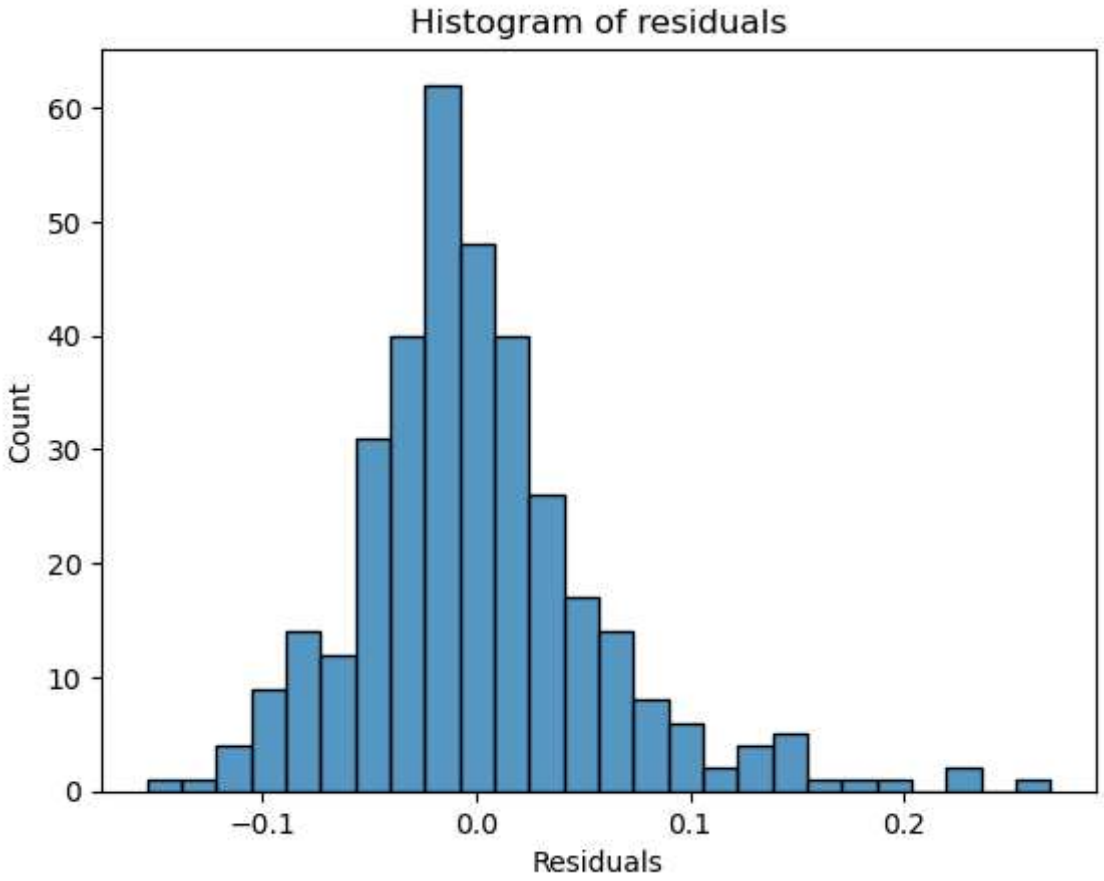
Lets See how is the error distribution for Customers

In [17]:
```
Y_hat = sm_model.predict(X_sm)
errors = Y_hat - y_train
```

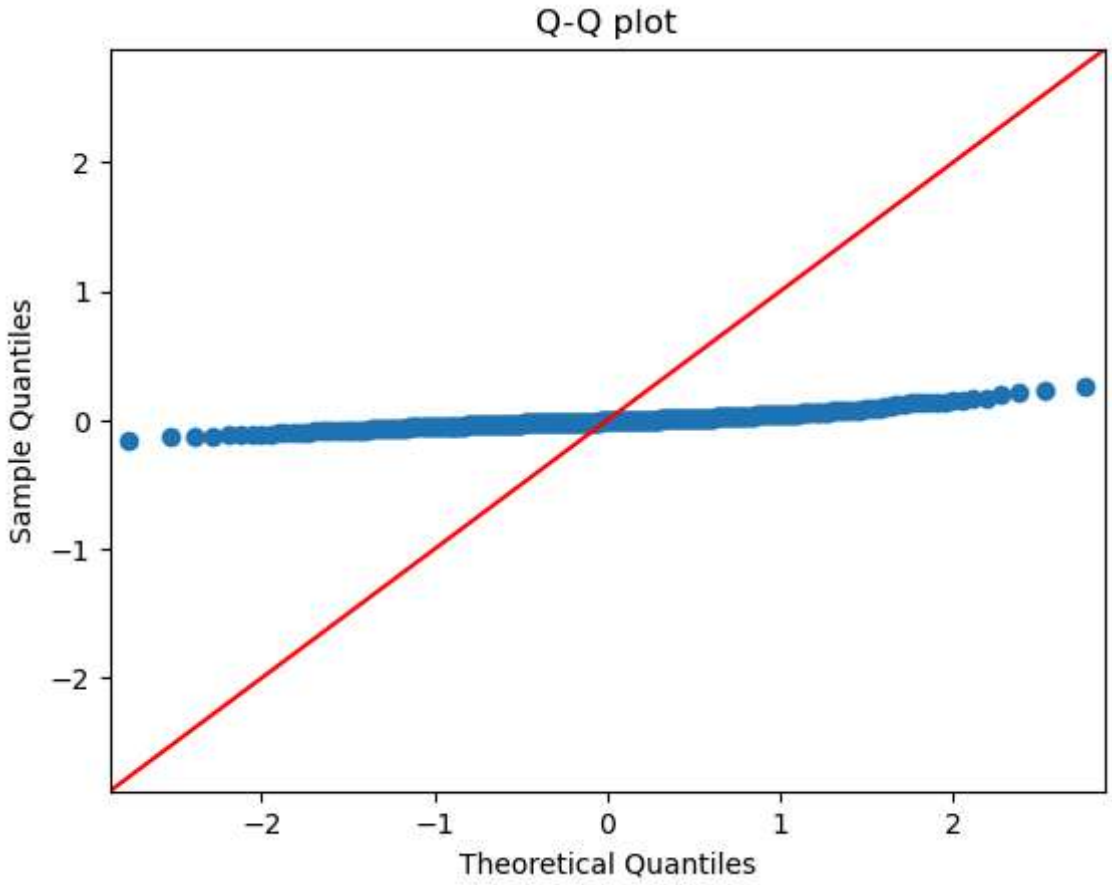
In [18]:
```
import seaborn as sns
sns.histplot(errors)
plt.xlabel("Residuals")
plt.title("Histogram of residuals")
```

```
C:\Users\adish\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert in
f values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[18]:
```
Text(0.5, 1.0, 'Histogram of residuals')
```



In [19]:
```
# Create Q-Q plot
sm.qqplot(errors, line='45')
plt.title('Q-Q plot')
plt.show()
```



In [20]:
```
errors.describe()
```

Out[20]:
```
count    3.500000e+02
mean    -1.733534e-16
std      5.784121e-02
min     -1.538169e-01
25%     -3.259827e-02
50%     -7.342898e-03
75%      2.496171e-02
max      2.683614e-01
Name: Chance of Admit , dtype: float64
```

Observations:

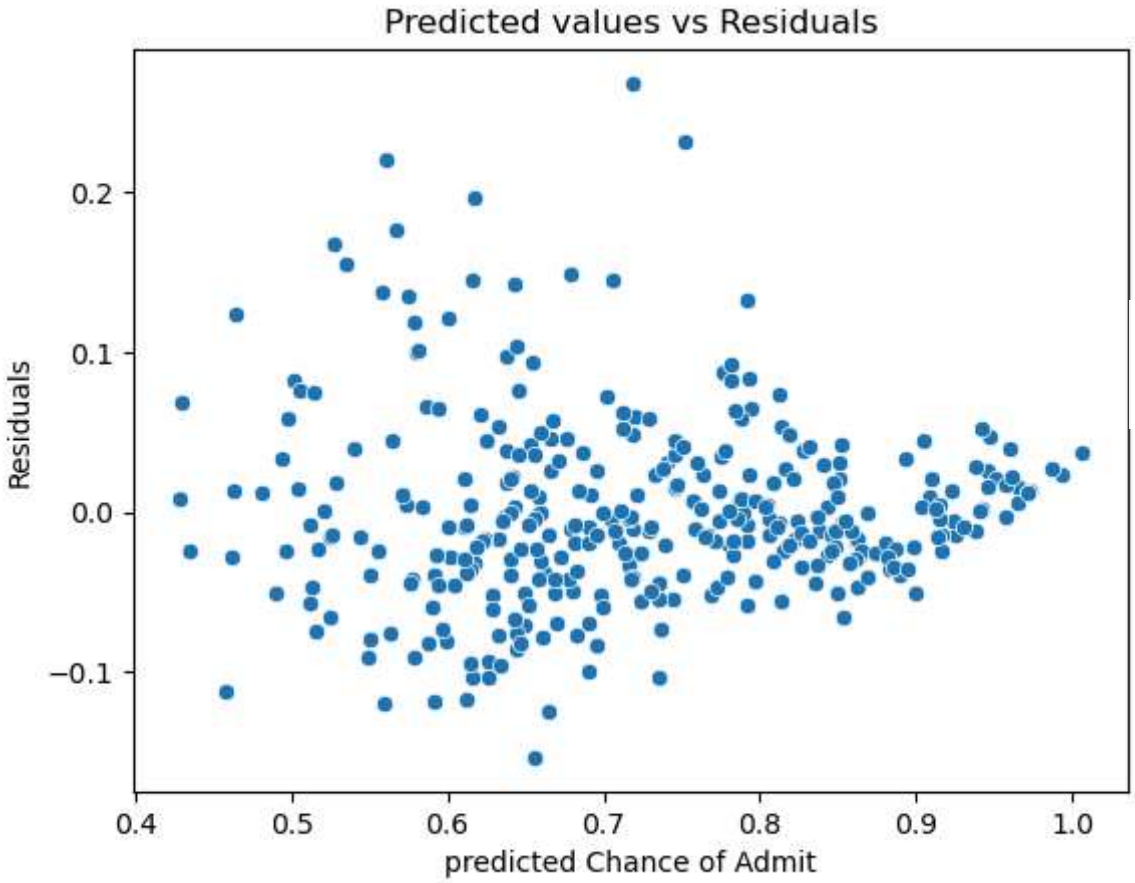
1) From above graph it can be said that the graph is right skewed.

4.Heteroskedasticity should not exist

In [21]:
```
Y_hat = sm_model.predict(X_sm)
errors = Y_hat - y_train
```

In [22]:
```
sns.scatterplot(x=Y_hat,y=errors)
plt.xlabel("predicted Chance of Admit")
plt.ylabel("Residuals")
plt.title("Predicted values vs Residuals")
```

Out[22]:
```
Text(0.5, 1.0, 'Predicted values vs Residuals')
```



We can assume that heteroskedasticity does not exist in our data

There are outliers present in the dataset

In [23]:
```
# Performing the Goldfeld-Quandt test to check for Homoscedasticity -
from statsmodels.compat import lzip
import statsmodels.stats.api as sms

name = ['F statistic', 'p-value']
test = sms.het_goldfeldquandt(y_train, X_sm)
lzip(name, test)
```

Out[23]:
```
[('F statistic', 0.9223637790731194), ('p-value', 0.6989321688927994)]
```

From the goldfeld-quandt test:

F Statistic comes out to be 0.922=> Implying minimal difference in variance between groups p-value of 0.698 indicates that this difference is statistically significant at conventional levels of significance (e.g., 0.05). Therefore, we accept the null hypothesis of homoscedasticity, and conclude that there is no strong evidence of heteroscedasticity in the data.

model evaluation- MAE, RMSE, R2 score, Adjusted R2.

In [24]:
```
Cust=pd.read_csv('customer.csv')
Cust=Cust.drop('Serial No.',axis=1)
y = Cust['Chance of Admit ']
X = Cust.drop('Chance of Admit ', axis=1)
y.shape, X.shape
```

Out[24]:
```
((500,), (500, 7))
```

In [25]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=
```

In [26]:
```
X_train.shape, y_train.shape
```

Out[26]:
```
((350, 7), (350,))
```

In [27]:
```
X_test.shape, y_test.shape
```

Out[27]:
```
((150, 7), (150,))
```

In [28]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[28]:
```
▾ LinearRegression
LinearRegression()
```

In [29]:
```
model.coef_
```

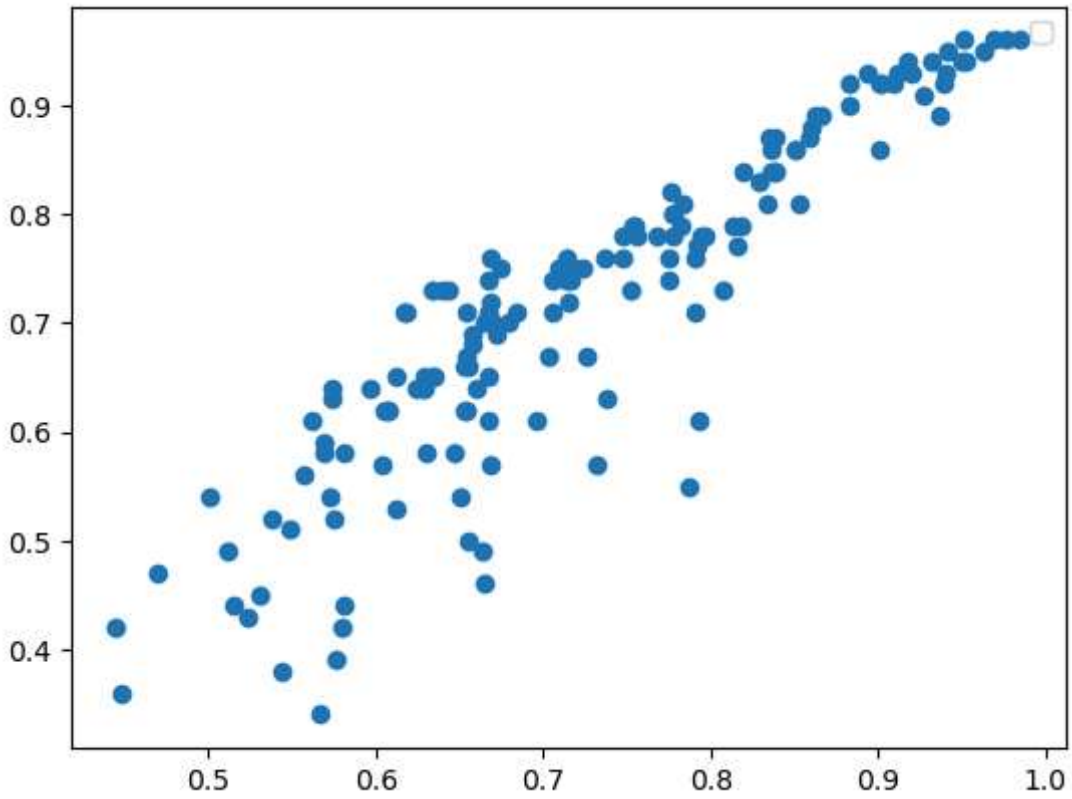
Out[29]:
```
array([ 0.00165342,  0.00381453,  0.01012349, -0.00100952,  0.01351732,
        0.10703419,  0.02813965])
```

In [30]:
```
model.intercept_
```

Out[30]:
```
-1.2161131174465947
```

In [31]:
```
import matplotlib.pyplot as plt
fig = plt.figure()
y_hat = model.predict(X_test)
plt.scatter(y_hat,y_test)
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start w
ith an underscore are ignored when legend() is called with no argument.
```



In [32]:
```
model.score(X_train, y_train)
```

Out[32]:
```
0.8209843725364347
```

In [33]:
```
model.score(X_test, y_test)
```

Out[33]:
```
0.8157672116057979
```

In [34]:
```
R2=model.score(X_test,y_test)
R2
```

Out[34]:
```
0.8157672116057979
```

In [35]:
```
y_hat = model.predict(X_test)
n=len(y_test)
Adj_R = 1 - (1-R2)*(len(y_test)-1)/(len(y_test)-(X_test.shape[1])-1)
print("Adjusted R-squared:", Adj_R )
```

```
Adjusted R-squared: 0.8066853135863654
```

In [36]:
```
MSE=mean_squared_error(y_test,y_hat)
print("Mean Square Error is:", MSE )
```

```
Mean Square Error is: 0.004125934236707799
```

In [37]:
```
mean_absolute_error(y_test,y_hat)
```

Out[37]:
```
0.043975442403392004
```

Observations

1) R2 score of 0.81 indicates that approximately 81% of the variance in the dependent variable (Chance of Admit) is explained by the independent variables in your model. This is a relatively high R2 score, suggesting that your model fits the data well.

2) The adjusted R2 score of 0.80 is very close to the R2 score, indicating that the additional independent variables in your model are not adding much explanatory power beyond what is already captured by the model.

3) The MSE of 0.004 represents the average squared difference between the actual and predicted values. A lower MSE indicates that your model is better at predicting the Chance of Admit.

4) The MAE of 0.0439 represents the average absolute difference between the actual and predicted values. It provides a more interpretable estimate of error compared to MSE.

Conclusion

1) In conclusion, the exploratory data analysis (EDA) revealed a dataset without null values or the need for data transformation, indicating the data's quality. The mean GRE score of 316 and TOEFL score of 107 indicate a high academic aptitude and strong English proficiency among students, respectively. The range of university ratings from 1 to 5 suggests a diverse representation of university rankings in the dataset.

2) Notably, the average university rating for admitted students is 3, demonstrating that students from various university backgrounds have been successful. The average CGPA score of 8.57 reflects a high level of academic achievement among students.

3) Additionally, the average Chance of Admit of 0.72 indicates a relatively high likelihood of admission for students in the dataset.

4) Regarding the assumptions of linearity, all features except LOR, SOP, and University Rating show linearity with the dependent variable, Chance of Admit. These findings provide valuable insights into the factors influencing graduate admissions and the interrelationships among them."

5) the VIF values for CGPA, GRE Score, and TOEFL Score are 4.7, 4.30, and 4.16 respectively, indicating some correlation with other independent variables but not at a level that raises significant multicollinearity concerns. These values are below the threshold of 5, suggesting that multicollinearity is not a major issue in the model. However, it is advisable to interpret the coefficients of these variables carefully to ensure they align with expectations.

6) The graph of the data is right-skewed, indicating that the distribution is not perfectly symmetrical.

7) Furthermore, the F Statistic of 0.922 suggests minimal difference in variance between groups, and the p-value of 0.698 indicates that this difference is not statistically significant at conventional levels of significance (e.g., 0.05). Therefore, we accept the null hypothesis of homoscedasticity, and conclude that there is no strong evidence of heteroscedasticity in the data.