Getting the Data

In [31]:

```
import numpy as np
import pandas as pd

df = pd.read_csv("Admission_Predict.csv", header = 0,
                 names = ["serial_no", "GRE", "TOEFL", "uni_rating", "SOP", "LOR", "CGPA", "research", "chance_admit"])
print(df.head())
```

```
   serial_no  GRE  TOEFL  uni_rating  SOP  LOR  CGPA  research  chance_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
```

What is the average GRE score among applicants with an admission chance above 80%?

In [32]:

```
df_above80 = df[df.chance_admit > .80]
print(df_above80.head())
avg_GRE_above80 = np.mean(df_above80.GRE)
print(df_above80.shape)
print("Average GRE score among applicants with admissions chance above 80%: " + str(avg_GRE_above80))
```

```
    serial_no  GRE  TOEFL  uni_rating  SOP  LOR  CGPA  research  chance_admit
0           1  337    118           4  4.5  4.5  9.65         1          0.92
5           6  330    115           5  4.5  3.0  9.34         1          0.90
11         12  327    111           4  4.0  4.5  9.00         1          0.84
22         23  328    116           5  5.0  5.0  9.50         1          0.94
23         24  334    119           5  5.0  4.5  9.70         1          0.95
(117, 9)
Average GRE score among applicants with admissions chance above 80%: 328.7350427350427
```

On average, is the chance of admission different for those who have research experience versus those who don't have research experience? (Assume 0 means no research experience and 1 means has research experience.)

In [33]:

```
# H0: The mean chance_admit for those with research is not different from those who don't have research experience
# HA: The mean chance_admit for those with research is different from those who don't have research
# alpha: .05

# Getting experimental difference in mean
df_has_research = df[df.research == 1]
avg_chance_research = np.mean(df_has_research.chance_admit)
print("Average Admission Chance for Research Experience: " + str(avg_chance_research))

df_no_research = df[df.research == 0]
avg_chance_noresearch = np.mean(df_no_research.chance_admit)
print("Average Admission Chance for No Research Experience: " + str(avg_chance_noresearch))

exp_diff = avg_chance_research - avg_chance_noresearch
print("Experimental Difference in Mean: " + str(exp_diff))

# Creating new datasets with shifted means so that mean of research/no-research is the same
shifted_research = df_has_research.chance_admit - avg_chance_research + np.mean(df.chance_admit)
shifted_no_research = df_no_research.chance_admit - avg_chance_noresearch + np.mean(df.chance_admit)

# Drawing boostrap replicates from shifted arrays
np.random.seed(12)
bs_replicates = np.empty(1000)
for i in range(1000) :
    r_sample = np.random.choice(shifted_research, len(shifted_research))
    nr_sample = np.random.choice(shifted_no_research, len(shifted_no_research))
    bs_replicates[i] = np.mean(r_sample) - np.mean(nr_sample)

# calculating the p-value
p = np.sum(bs_replicates > exp_diff) / len(bs_replicates)
print("p-value: ", p)
```

```
# As this is a two-tailed t-test, p-value must be less than .025, which it is. Therefore, we reject the null hypothesis.
# There is sufficient evidence to suggest that the mean chance of admission for those with research experience is higher
# than the mean chance of admission for those without research experience
```

```
Average Admission Chance for Research Experience: 0.7959817351598172
Average Admission Chance for No Research Experience: 0.6376795580110497
Experimental Difference in Mean: 0.1583021771487675
p-value:  0.0
```

Put yourself in the shoes of someone applying to a Master's program. Based on your analysis of this dataset, what are some things you should focus on as an undergraduate to maximize your chance of getting into a Master's program? (This question is optional (and we mean it when we say that); it will not hurt your application if you choose not to answer it. If you want to take a stab at it, keep your answer to a few sentences.)

In [34]:

```
# Lasso Regression and correlation will be used to determine the most important variables used in predicting Admissions Chance

# Splitting data into X and y
X = df.iloc[:, 1:8]
y = df.chance_admit

# Finding the best Lasso Regression Model and alpha value for it
from sklearn import linear_model
from sklearn import model_selection
for a in [0, .01, .02, .03, .04, .05, .06, .07, .08, .09, .1] :
    Lreg = linear_model.Lasso(alpha = a)
    cv_scores = model_selection.cross_val_score(Lreg, X, y, cv = 5)
    Lreg.fit(X, y)
    print("Alpha: ", a)
    print("Mean CV-score: ", str(np.mean(cv_scores)))
    print(Lreg.coef_)

# Getting correlation between variables and admission chance
from scipy import stats
print("GRE Correlation: ", stats.pearsonr(df.GRE, df.chance_admit)[0])
print("TOEFL Correlation: ", stats.pearsonr(df.TOEFL, df.chance_admit)[0])
print("uni_rating Correlation: ", stats.pearsonr(df.uni_rating, df.chance_admit)[0])
print("SOP Correlation: ", stats.pearsonr(df.SOP, df.chance_admit)[0])
print("LOR Correlation: ", stats.pearsonr(df.LOR, df.chance_admit)[0])
print("CGPA Correlation: ", stats.pearsonr(df.CGPA, df.chance_admit)[0])
print("Research Correlation: ", stats.pearsonr(df.research, df.chance_admit)[0])
```

```
Alpha:  0
Mean CV-score:  0.7711794121066355
[ 0.00173741  0.00291958  0.00571666 -0.00330517  0.02235313  0.11893945
  0.02452511]
Alpha:  0.01
Mean CV-score:  0.7050196056963844
[0.00499735 0.0067262  0.00661483 0.         0.02121294 0.01294679
 0.        ]
Alpha:
```

```
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:515: UserWarning: With alpha=0, this algorithm does not converg
e well. You are advised to use the LinearRegression estimator
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:476: UserWarning: Coordinate descent with no regularizati
on may lead to unexpected results and is discouraged.
  positive)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:476: ConvergenceWarning: Objective did not converge. Y
ou might want to increase the number of iterations. Duality gap: 0.4571582540612917, tolerance: 0.000580283875
  positive)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:515: UserWarning: With alpha=0, this algorithm does not converg
e well. You are advised to use the LinearRegression estimator
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:476: UserWarning: Coordinate descent with no regularizati
on may lead to unexpected results and is discouraged.
  positive)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:476: ConvergenceWarning: Objective did not converge. Y
ou might want to increase the number of iterations. Duality gap: 0.5933321609253572, tolerance: 0.000607248875
  positive)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:515: UserWarning: With alpha=0, this algorithm does not converg
e well. You are advised to use the LinearRegression estimator
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:476: UserWarning: Coordinate descent with no regularizati
on may lead to unexpected results and is discouraged.
  positive)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:476: ConvergenceWarning: Objective did not converge. Y
ou might want to increase the number of iterations. Duality gap: 0.7034920919863558, tolerance: 0.00069395846875
  positive)
```

positive)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\model_selection_validation.py:515: UserWarning: With alpha=0, this algorithm does not converg
e well. You are advised to use the LinearRegression estimator
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:476: UserWarning: Coordinate descent with no regularizati
on may lead to unexpected results and is discouraged.
  positive)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:476: ConvergenceWarning: Objective did not converge. Y
ou might want to increase the number of iterations. Duality gap: 0.7353733866305826, tolerance: 0.00072643121875
  positive)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\model_selection_validation.py:515: UserWarning: With alpha=0, this algorithm does not converg
e well. You are advised to use the LinearRegression estimator
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:476: UserWarning: Coordinate descent with no regularizati
on may lead to unexpected results and is discouraged.
  positive)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:476: ConvergenceWarning: Objective did not converge. Y
ou might want to increase the number of iterations. Duality gap: 0.6372098569834954, tolerance: 0.000629989875
  positive)
C:\Users\Asus\anaconda3\lib\site-packages\ipykernel_launcher.py:13: UserWarning: With alpha=0, this algorithm does not converge well. You are ad
vised to use the LinearRegression estimator
  del sys.path[0]
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:476: UserWarning: Coordinate descent with no regularizati
on may lead to unexpected results and is discouraged.
  positive)
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:476: ConvergenceWarning: Objective did not converge. Y
ou might want to increase the number of iterations. Duality gap: 0.7973786486493799, tolerance: 0.0008114630999999999
  positive)

 0.02
Mean CV-score: 0.6631723038854276
[0.00591762 0.00796219 0.   0.   0.00882   0.
 0.   ]
Alpha: 0.03
Mean CV-score: 0.6475207960486429
[0.00625483 0.00789834 0.   0.   0.   0.
 0.   ]
Alpha: 0.04
Mean CV-score: 0.646315670767989
[0.00640152 0.0073944 0.   0.   0.   0.
 0.   ]
Alpha: 0.05
Mean CV-score: 0.6446159012380315
[0.00654821 0.00689045 0.   0.   0.   0.
 0.   ]
Alpha: 0.06
Mean CV-score: 0.6424214874587719
[0.0066949 0.00638651 0.   0.   0.   0.
 0.   ]
Alpha: 0.07
Mean CV-score: 0.6397295790907416
[0.00684188 0.0058821 0.   0.   0.   0.
 0.   ]
Alpha: 0.08
Mean CV-score: 0.6365459472569738
[0.00698854 0.00537819 0.   0.   0.   0.
 0.   ]
Alpha: 0.09
Mean CV-score: 0.6328662557841943
[0.00713521 0.00487429 0.   0.   0.   0.
 0.   ]
Alpha: 0.1
Mean CV-score: 0.6286913821191427
[0.00728218 0.00436988 0.   0.   0.   0.
 0.   ]
GRE Correlation: 0.8026104595903504
TOEFL Correlation: 0.7915939869351045
uni_rating Correlation: 0.7112502503917222
SOP Correlation: 0.6757318583886719
LOR Correlation: 0.6698887920106938
CGPA Correlation: 0.8732890993553002
Research Correlation: 0.5532021370190403