Choosing a college to attend is one of the most important decision one can make. What measures the success of a college degree? It can be either the height of the societal ladder that you can climb, or how much content you are with your life. For the sake of this project, we are measuring a quantitative variable: Salary.

The college in exploration are divided based on 3 categories:

- Colleges by Type
- Colleges by Region
- Salary by major

The dataset is from The Wall Street Journal, available on Kaggle at: https://www.kaggle.com/wsj/college-salaries (https://www.kaggle.com/wsj/college-salaries)

First, let's import some library we will use later

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
```

Let's load our first dataset: Colleges by Type

```
In [2]: college_type = pd.read_csv("salaries-by-college-type.csv")
        college_region = pd.read_csv("salaries-by-region.csv")
```

Take a peek at our data

```
In [3]: college_type.head(5)
```

Out[3]:

	School Name	School Type	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 25th Percentile Salary	Mid-Care 75 Percent Sala
0	Massachusetts Institute of Technology (MIT)	Engineering	$72,200.00	$126,000.00	$76,800.00	$99,200.00	$168,000.
1	California Institute of Technology (CIT)	Engineering	$75,500.00	$123,000.00	NaN	$104,000.00	$161,000.
2	Harvey Mudd College	Engineering	$71,800.00	$122,000.00	NaN	$96,000.00	$180,000.
3	Polytechnic University of New York, Brooklyn	Engineering	$62,400.00	$114,000.00	$66,800.00	$94,300.00	$143,000.
4	Cooper Union	Engineering	$62,200.00	$114,000.00	NaN	$80,200.00	$142,000.

```
In [4]: college_region.head(5)
```

Out[4]:

	School Name	Region	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 25th Percentile Salary	Mid-Career 75th Percentile Salary	M
0	Stanford University	California	$70,400.00	$129,000.00	$68,400.00	$93,100.00	$184,000.00	$2
1	California Institute of Technology (CIT)	California	$75,500.00	$123,000.00	NaN	$104,000.00	$161,000.00	Na
2	Harvey Mudd College	California	$71,800.00	$122,000.00	NaN	$96,000.00	$180,000.00	Na
3	University of California, Berkeley	California	$59,900.00	$112,000.00	$59,500.00	$81,000.00	$149,000.00	$2
4	Occidental College	California	$51,900.00	$105,000.00	NaN	$54,800.00	$157,000.00	Na

Notice that the salary collumn is identical, let's combine these 2 dataset so we can have both location and school type for each college

```
In [5]: df = pd.merge(college_type, college_region)
```

In [6]:
```
df.head()
```

Out[6]:

	School Name	School Type	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 25th Percentile Salary	Mid-Care 75 Percent Sala
0	Massachusetts Institute of Technology (MIT)	Engineering	$72,200.00	$126,000.00	$76,800.00	$99,200.00	$168,000.
1	California Institute of Technology (CIT)	Engineering	$75,500.00	$123,000.00	NaN	$104,000.00	$161,000.
2	Harvey Mudd College	Engineering	$71,800.00	$122,000.00	NaN	$96,000.00	$180,000.
3	Polytechnic University of New York, Brooklyn	Engineering	$62,400.00	$114,000.00	$66,800.00	$94,300.00	$143,000.
4	Cooper Union	Engineering	$62,200.00	$114,000.00	NaN	$80,200.00	$142,000.

Ah, the region is there but it's at the last collumn, which is not visible much. Let's move it to the second collumn

In [7]:
```
region = df.pop("Region")
df.insert(1, 'Region', region)
```

```
In [8]: df.head()
```

Out[8]:

	School Name	Region	School Type	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Car 2 Percen Sal
0	Massachusetts Institute of Technology (MIT)	Northeastern	Engineering	$72,200.00	$126,000.00	$76,800.00	$99,200.0
1	California Institute of Technology (CIT)	California	Engineering	$75,500.00	$123,000.00	NaN	$104,000
2	Harvey Mudd College	California	Engineering	$71,800.00	$122,000.00	NaN	$96,000.0
3	Polytechnic University of New York, Brooklyn	Northeastern	Engineering	$62,400.00	$114,000.00	$66,800.00	$94,300.0
4	Cooper Union	Northeastern	Engineering	$62,200.00	$114,000.00	NaN	$80,200.0

Nice! Now let's take a quick look overview of our data

```
In [9]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 260 entries, 0 to 259
Data columns (total 9 columns):
School Name                          260 non-null object
Region                               260 non-null object
School Type                          260 non-null object
Starting Median Salary               260 non-null object
Mid-Career Median Salary             260 non-null object
Mid-Career 10th Percentile Salary    223 non-null object
Mid-Career 25th Percentile Salary    260 non-null object
Mid-Career 75th Percentile Salary    260 non-null object
Mid-Career 90th Percentile Salary    223 non-null object
dtypes: object(9)
memory usage: 20.3+ KB
```

In [10]:
```
df.describe()
```

Out[10]:

	School Name	Region	School Type	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid Caree 25tl Percentil Salar
count	260	260	260	260	260	223	260
unique	240	5	5	142	162	135	172
top	Pennsylvania State University (PSU)	Northeastern	State	$42,600.00	$72,100.00	$40,100.00	$54,100.0
freq	2	69	169	7	5	6	6

Notice that all the variables are object type. Let's change School Name, Region, School Type to string, and the rest to numeric values

Before we change it to numeric values, we need to drop the "$" sign before the value

In [11]:
```
# Remove the dollar $ sign
colstocheck = df.columns
df[colstocheck] = df[colstocheck].replace({'\$': '', ',': ''}, regex = True)
```

In [17]:
```
to_change = ["Starting Median Salary", "Mid-Career Median Salary", "Mid-Career
10th Percentile Salary", "Mid-Career 25th Percentile Salary", "Mid-Career 75th
Percentile Salary", "Mid-Career 90th Percentile Salary"]
for i in to_change:
    df[i] = df[i].astype(float)

#df["Mid-Career 90th Percentile Salary"] = df["Mid-Career 90th Percentile Sala
ry"].astype(float)
```

```
In [18]: df.head()
```

```
Out[18]:
```

	School Name	Region	School Type	Starting Median Salary	Mid-Career Median Salary	Mid-Career 10th Percentile Salary	Mid-Career 25th Percentile Salary	Per
0	Massachusetts Institute of Technology (MIT)	Northeastern	Engineering	72200.0	126000.0	76800.0	99200.0	168
1	California Institute of Technology (CIT)	California	Engineering	75500.0	123000.0	NaN	104000.0	161
2	Harvey Mudd College	California	Engineering	71800.0	122000.0	NaN	96000.0	180
3	Polytechnic University of New York Brooklyn	Northeastern	Engineering	62400.0	114000.0	66800.0	94300.0	143
4	Cooper Union	Northeastern	Engineering	62200.0	114000.0	NaN	80200.0	142

```
In [19]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 260 entries, 0 to 259
Data columns (total 9 columns):
School Name                          260 non-null object
Region                               260 non-null object
School Type                          260 non-null object
Starting Median Salary               260 non-null float64
Mid-Career Median Salary             260 non-null float64
Mid-Career 10th Percentile Salary    223 non-null float64
Mid-Career 25th Percentile Salary    260 non-null float64
Mid-Career 75th Percentile Salary    260 non-null float64
Mid-Career 90th Percentile Salary    223 non-null float64
dtypes: float64(6), object(3)
memory usage: 20.3+ KB
```

Great! Looks like we have pretty much done with the pre-processing data. Now the last step would be to process the missing values.

There are several ways to handle missing data:

1. Delete the entire rows which have the missing values. This is the simplest, but not ideal because the more data we have for this project, the better! There are 15% of the data go missing in this dataset.
2. Fill in the missing values with the mean, median, or mode of the same type of variable. Let's go with this option

Actually after doing some research, there are 6 ways to handle this! Let's try some methods recommended here: https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779 (https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779)

```
In [20]: #import sys
         #from impyute.imputation.cs import fast_knn
         #sys.setrecursionlimit(100000) #Increase the recursion limit of the OS

         # start the KNN training
         #imputed_training=fast_knn(train.values, k=30)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-20-79d38c8586f7> in <module>()
      1 import sys
----> 2 from impyute.imputation.cs import fast_knn
      3 sys.setrecursionlimit(100000) #Increase the recursion limit of the OS
      4 
      5 # start the KNN training

ModuleNotFoundError: No module named 'impyute'
```

In [27]:

```
# Method 1: Replacing NaN values with mean value of that variable
#Impute the values using scikit-learn SimpleImpute Class
####import sklearn
####from sklearn.preprocessing import Imputer
####imputer = Imputer(missing_values="NaN", strategy="mean", axis=1)
####imputed_mean_df = imputer.fit_transform(df)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-27-4ecc358f4152> in <module>()
      4 from sklearn.preprocessing import Imputer
      5 imputer = Imputer(missing_values="NaN", strategy="mean", axis=1)
----> 6 imputed_mean_df = imputer.fit_transform(df)
      7 #from sklearn.impute import SimpleImputer
      8 #imp_mean = SimpleImputer(strategy='mean') #for median imputation rep
lace 'mean' with 'median'

~\Anaconda3\lib\site-packages\sklearn\base.py in fit_transform(self, X, y, **
fit_params)
    515         if y is None:
    516             # fit method of arity 1 (unsupervised transformation)
--> 517             return self.fit(X, **fit_params).transform(X)
    518         else:
    519             # fit method of arity 2 (supervised transformation)

~\Anaconda3\lib\site-packages\sklearn\preprocessing\imputation.py in transfor
m(self, X)
    321         else:
    322             X = check_array(X, accept_sparse='csr', dtype=FLOAT_DTYPE
S,
--> 323                             force_all_finite=False, copy=self.copy)
    324 
    325             if sparse.issparse(X):

~\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(arra
y, accept_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd,
 ensure_min_samples, ensure_min_features, warn_on_dtype, estimator)
    431                                       force_all_finite)
    432     else:
--> 433         array = np.array(array, dtype=dtype, order=order, copy=copy)
    434 
    435         if ensure_2d:

ValueError: could not convert string to float: 'State'
```

```
In [31]: df.isnull().sum()
```

```
Out[31]: School Name                           0
         Region                                0
         School Type                           0
         Starting Median Salary                0
         Mid-Career Median Salary              0
         Mid-Career 10th Percentile Salary    37
         Mid-Career 25th Percentile Salary     0
         Mid-Career 75th Percentile Salary     0
         Mid-Career 90th Percentile Salary    37
         dtype: int64
```

```
In [32]: df.isnull().sum().sum()
```

```
Out[32]: 74
```

```
 In [ ]: #from sklearn.impute import SimpleImputer
         #imp_mean = SimpleImputer(strategy='mean') #for median imputation replace 'mea
         n' with 'median'
         #imp_mean.fit(df)
         #imputed_mean_df = imp_mean.transform(df)
         #Impute the values using scikit-learn SimpleImpute Class
```