```python
#needed libraries
import pandas as pd
import seaborn as sns
from scipy import stats
import numpy as np
import matplotlib.pyplot as plt
```

Milestone 2

Perform at least 5 data transformation and/or cleansing steps to your flat file data. For example:

1. Replace Headers
2. Format data into a more readable format
3. Identify outliers and bad data
4. Find duplicates
5. Fix casing or inconsistent values
6. Conduct Fuzzy Matching

Reading in flat files

```python
#salaries_by_region.csv
#Reading flat files into DataFrame object
salaries_region = pd.read_csv("datasets/salaries-by-region.csv")
print(salaries_region.head())
```

```
                              School Name      Region  \
0                     Stanford University  California
1  California Institute of Technology (CIT)  California
2                       Harvey Mudd College  California
3         University of California, Berkeley  California
4                      Occidental College  California

  Starting Median Salary Mid-Career Median Salary  \
0            $70,400.00              $129,000.00
1            $75,500.00              $123,000.00
2            $71,800.00              $122,000.00
3            $59,900.00              $112,000.00
4            $51,900.00              $105,000.00

  Mid-Career 10th Percentile Salary Mid-Career 25th Percentile Salary  \
0                       $68,400.00                        $93,100.00
1                              NaN                       $104,000.00
2                              NaN                        $96,000.00
3                       $59,500.00                        $81,000.00
4                              NaN                        $54,800.00

  Mid-Career 75th Percentile Salary Mid-Career 90th Percentile Salary
0                      $184,000.00                       $257,000.00
1                      $161,000.00                               NaN
2                      $180,000.00                               NaN
3                      $149,000.00                       $201,000.00
4                      $157,000.00                               NaN
```

```python
#salaries-by-college-type.csv
salaries_type = pd.read_csv("datasets/salaries-by-college-type.csv")
print(salaries_type.head())
```

```
                                  School Name  School Type  \
0   Massachusetts Institute of Technology (MIT)  Engineering
1      California Institute of Technology (CIT)  Engineering
2                         Harvey Mudd College  Engineering
3  Polytechnic University of New York, Brooklyn  Engineering
4                               Cooper Union  Engineering

  Starting Median Salary Mid-Career Median Salary  \
0            $72,200.00              $126,000.00
1            $75,500.00              $123,000.00
2            $71,800.00              $122,000.00
3            $62,400.00              $114,000.00
4            $62,200.00              $114,000.00

  Mid-Career 10th Percentile Salary Mid-Career 25th Percentile Salary  \
0                       $76,800.00                        $99,200.00
1                              NaN                       $104,000.00
2                              NaN                        $96,000.00
3                       $66,800.00                        $94,300.00
4                              NaN                        $80,200.00

  Mid-Career 75th Percentile Salary Mid-Career 90th Percentile Salary
0                      $168,000.00                       $220,000.00
1                      $161,000.00                               NaN
2                      $180,000.00                               NaN
3                      $143,000.00                       $190,000.00
4                      $142,000.00                               NaN
```

Merging 2 Salaries Flat Files

```python
salaries_merged = pd.merge(salaries_region,salaries_type,on='School Name')
salaries_merged.head()
```

	School Name	Region	Starting Median Salary_x	Mid-Career Median Salary_x	Mid-Career 10th Percentile Salary_x	Mid-Career 25th Percentile Salary_x	Mid-Career 75th Percentile Salary_x	Mid-Career 90th Percentile Salary_x	School Type	Starting Median Salary_y	Mid-Career Median Salary_y	Mid Caree 10t Percentil Salary_
0	California Institute of Technology (CIT)	California	$75,500.00	$123,000.00	NaN	$104,000.00	$161,000.00	NaN	Engineering	$75,500.00	$123,000.00	Nal
1	Harvey Mudd College	California	$71,800.00	$122,000.00	NaN	$96,000.00	$180,000.00	NaN	Engineering	$71,800.00	$122,000.00	Nal
2	University of California, Berkeley	California	$59,900.00	$112,000.00	$59,500.00	$81,000.00	$149,000.00	$201,000.00	State	$59,900.00	$112,000.00	$59,500.0
3	Occidental College	California	$51,900.00	$105,000.00	NaN	$54,800.00	$157,000.00	NaN	Liberal Arts	$51,900.00	$105,000.00	Nal
4	Cal Poly San Luis Obispo	California	$57,200.00	$101,000.00	$55,000.00	$74,700.00	$133,000.00	$178,000.00	State	$57,200.00	$101,000.00	$55,000.0

Removing Duplicate and Unncessary Columns

For the percentile columns, we have values for the 10th, 25th, 50th, 75th, and 90th percentiles. Given that we can find the interquartile range for a dataset by using Q3 (75th) and Q1 (25th), I don't think we will need the 10th and 90th percentile fields. The IQR will give us a good idea of statistical dispersion, and is commonly used as a robust measure of scale.

I am also going to drop the 10th and 90th percentile columns from the first DataFrame.

```python
#removing second set of salary columns as they contain same data from first dataframe
salaries_merged = salaries_merged.drop(['Mid-Career 10th Percentile Salary_x','Mid-Career 90th Percentile Salar
```

```python
print(salaries_merged.head())
```

```
                              School Name      Region  \
0  California Institute of Technology (CIT)  California
1                       Harvey Mudd College  California
2         University of California, Berkeley  California
3                      Occidental College  California
4                Cal Poly San Luis Obispo  California

  Starting Median Salary_x Mid-Career Median Salary_x  \
0              $75,500.00                $123,000.00
1              $71,800.00                $122,000.00
2              $59,900.00                $112,000.00
3              $51,900.00                $105,000.00
4              $57,200.00                $101,000.00

  Mid-Career 25th Percentile Salary_x Mid-Career 75th Percentile Salary_x  \
0                        $104,000.00                         $161,000.00
1                         $96,000.00                         $180,000.00
2                         $81,000.00                         $149,000.00
3                         $54,800.00                         $157,000.00
4                         $74,700.00                         $133,000.00

    School Type
0   Engineering
1   Engineering
2         State
3  Liberal Arts
4         State
```

Rearranging Columns in Merged DataFrame

```python
#retrieve columns of the merged dataframe in list form
cols = salaries_merged.columns.tolist()
cols
```

```
['School Name',
 'Region',
 'Starting Median Salary_x',
 'Mid-Career Median Salary_x',
 'Mid-Career 25th Percentile Salary_x',
 'Mid-Career 75th Percentile Salary_x',
 'School Type']
```

```python
#reassign cols list in the order wanted
cols = ['School Name','Region','School Type','Starting Median Salary_x','Mid-Career Median Salary_x','Mid-Caree
salaries_merged = salaries_merged[cols]
```

```python
print(salaries_merged.head())
```

```
                              School Name      Region   School Type  \
0  California Institute of Technology (CIT)  California   Engineering
1                       Harvey Mudd College  California   Engineering
2         University of California, Berkeley  California         State
3                      Occidental College  California  Liberal Arts
4                Cal Poly San Luis Obispo  California         State

  Starting Median Salary_x Mid-Career Median Salary_x  \
0              $75,500.00                $123,000.00
1              $71,800.00                $122,000.00
2              $59,900.00                $112,000.00
3              $51,900.00                $105,000.00
4              $57,200.00                $101,000.00

  Mid-Career 25th Percentile Salary_x Mid-Career 75th Percentile Salary_x
0                        $104,000.00                         $161,000.00
1                         $96,000.00                         $180,000.00
2                         $81,000.00                         $149,000.00
3                         $54,800.00                         $157,000.00
4                         $74,700.00                         $133,000.00
```

Renaming Existing Columns

```python
salaries_merged = salaries_merged.rename(columns={'School Name':'school_name','School Type':'school_type','Star
```

```python
print(salaries_merged.head())
```

```
                              school_name      Region   school_type  \
0  California Institute of Technology (CIT)  California   Engineering
1                       Harvey Mudd College  California   Engineering
2         University of California, Berkeley  California         State
3                      Occidental College  California  Liberal Arts
4                Cal Poly San Luis Obispo  California         State

  starting_median_salary midCareer_median_salary midCareer_25th_salary  \
0            $75,500.00             $123,000.00           $104,000.00
1            $71,800.00             $122,000.00            $96,000.00
2            $59,900.00             $112,000.00            $81,000.00
3            $51,900.00             $105,000.00            $54,800.00
4            $57,200.00             $101,000.00            $74,700.00

  midCareer_75th_salary
0           $161,000.00
1           $180,000.00
2           $149,000.00
3           $157,000.00
4           $133,000.00
```

Checking for Duplicates

```python
#can check for duplicates using pandas duplicated function on each column

#for loop through each column in dataframe
print("Checking for duplicates in DataFrame\n")
for col in salaries_merged.columns:
    print(col + ": " + str(any(salaries_merged[col].duplicated())))
```

```
Checking for duplicates in DataFrame

school_name: True
Region: True
school_type: True
starting_median_salary: True
midCareer_median_salary: True
midCareer_25th_salary: True
midCareer_75th_salary: True
```

There are duplicates in the 'school_name' column which is not ideal as this is technically the key for our DataFrame and each record for each School Name should be unique.

I am going to investigate the duplicates in this column.

```python
#looking at the duplicate values in the school_name column
salaries_merged[salaries_merged.duplicated(['school_name'], keep=False)]

#20 schools show up twice in the dataframe
```

	school_name	Region	school_type	starting_median_salary	midCareer_median_salary	midCareer_25th_salary	midCareer_75th_salary
11	University of California, Santa Barbara (UCSB)	California	Party	$50,500.00	$95,000.00	$71,200.00	$129,000.00
12	University of California, Santa Barbara (UCSB)	California	State	$50,500.00	$95,000.00	$71,200.00	$129,000.00
34	Arizona State University (ASU)	Western	Party	$47,400.00	$84,100.00	$60,700.00	$114,000.00
35	Arizona State University (ASU)	Western	State	$47,400.00	$84,100.00	$60,700.00	$114,000.00
68	University of Illinois at Urbana-Champaign (UIUC)	Midwestern	Party	$52,900.00	$96,100.00	$68,900.00	$132,000.00
69	University of Illinois at Urbana-Champaign (UIUC)	Midwestern	State	$52,900.00	$96,100.00	$68,900.00	$132,000.00
80	Indiana University (IU), Bloomington	Midwestern	Party	$46,300.00	$84,000.00	$60,400.00	$119,000.00
81	Indiana University (IU), Bloomington	Midwestern	State	$46,300.00	$84,000.00	$60,400.00	$119,000.00
82	University of Iowa (UI)	Midwestern	Party	$44,700.00	$83,900.00	$61,100.00	$116,000.00
83	University of Iowa (UI)	Midwestern	State	$44,700.00	$83,900.00	$61,100.00	$116,000.00
106	Ohio University	Midwestern	Party	$42,200.00	$73,400.00	$52,800.00	$106,000.00
107	Ohio University	Midwestern	State	$42,200.00	$73,400.00	$52,800.00	$106,000.00
136	University of Maryland, College Park	Southern	Party	$52,000.00	$95,000.00	$68,300.00	$126,000.00
137	University of Maryland, College Park	Southern	State	$52,000.00	$95,000.00	$68,300.00	$126,000.00
139	University of Texas (UT) - Austin	Southern	Party	$49,700.00	$93,900.00	$67,400.00	$129,000.00
140	University of Texas (UT) - Austin	Southern	State	$49,700.00	$93,900.00	$67,400.00	$129,000.00
141	University of Florida (UF)	Southern	Party	$47,100.00	$87,900.00	$62,900.00	$120,000.00
142	University of Florida (UF)	Southern	State	$47,100.00	$87,900.00	$62,900.00	$120,000.00
143	Louisiana State University (LSU)	Southern	Party	$46,900.00	$87,800.00	$61,300.00	$120,000.00
144	Louisiana State University (LSU)	Southern	State	$46,900.00	$87,800.00	$61,300.00	$120,000.00
147	University of Georgia (UGA)	Southern	Party	$44,100.00	$86,000.00	$57,800.00	$118,000.00
148	University of Georgia (UGA)	Southern	State	$44,100.00	$86,000.00	$57,800.00	$118,000.00
151	Randolph-Macon College	Southern	Party	$42,600.00	$83,600.00	$54,100.00	$123,000.00
152	Randolph-Macon College	Southern	Liberal Arts	$42,600.00	$83,600.00	$54,100.00	$123,000.00
158	University of Alabama, Tuscaloosa	Southern	Party	$41,300.00	$81,400.00	$56,500.00	$117,000.00
159	University of Alabama, Tuscaloosa	Southern	State	$41,300.00	$81,400.00	$56,500.00	$117,000.00
164	University of Mississippi	Southern	Party	$41,400.00	$79,700.00	$53,500.00	$108,000.00
165	University of Mississippi	Southern	State	$41,400.00	$79,700.00	$53,500.00	$108,000.00
169	West Virginia University (WVU)	Southern	Party	$43,100.00	$78,100.00	$55,700.00	$106,000.00
170	West Virginia University (WVU)	Southern	State	$43,100.00	$78,100.00	$55,700.00	$106,000.00
174	University of Tennessee	Southern	Party	$43,800.00	$74,600.00	$53,200.00	$106,000.00
175	University of Tennessee	Southern	State	$43,800.00	$74,600.00	$53,200.00	$106,000.00
179	Florida State University (FSU)	Southern	Party	$42,100.00	$73,000.00	$52,800.00	$107,000.00
180	Florida State University (FSU)	Southern	State	$42,100.00	$73,000.00	$52,800.00	$107,000.00
231	State University of New York (SUNY) at Albany	Northeastern	Party	$44,500.00	$92,200.00	$63,100.00	$135,000.00
232	State University of New York (SUNY) at Albany	Northeastern	State	$44,500.00	$92,200.00	$63,100.00	$135,000.00
239	Pennsylvania State University (PSU)	Northeastern	Party	$49,900.00	$85,700.00	$62,000.00	$117,000.00
240	Pennsylvania State University (PSU)	Northeastern	State	$49,900.00	$85,700.00	$62,000.00	$117,000.00
253	University of New Hampshire (UNH)	Northeastern	Party	$41,800.00	$78,300.00	$56,400.00	$114,000.00
254	University of New Hampshire (UNH)	Northeastern	State	$41,800.00	$78,300.00	$56,400.00	$114,000.00

Removing Duplicates

```python
#sort dataframe by key, school_name
salaries_merged = salaries_merged.sort_values("school_name")
```

```python
#create new dataframe and drop duplicates
#keep = 'first' --> keep the first occurrence of the duplicate and remove the other one, leave one unique recor
salaries_unique = salaries_merged.drop_duplicates(subset="school_name",keep='first')
```

```python
orig_size = salaries_merged.shape
new_size = salaries_unique.shape
```

```python
print("Original size before removing duplicates: " + str(orig_size))
```

```
Original size before removing duplicates: (268, 7)
```

```python
print("New size after removing duplicates: " + str(new_size))
```

```
New size after removing duplicates: (248, 7)
```

Identifying Missing Values

```python
#loop through columns in dataframe
#check for any NaN values
for col in salaries_unique.columns:
    print(col + ": " + str(salaries_unique[col].isnull().values.any()))

#no missing values!
```

```
school_name: False
Region: False
school_type: False
starting_median_salary: False
midCareer_median_salary: False
midCareer_25th_salary: False
midCareer_75th_salary: False
```

Describing Data & Checking field types

```python
#describing our dataframe with duplicates removed
salaries_unique.describe()
```

	school_name	Region	school_type	starting_median_salary	midCareer_median_salary	midCareer_25th_salary	midCareer_75th_salary
count	248	248	248	248	248	248	248
unique	248	5	5	145	166	178	110
top	Western Michigan University (WMU)	Northeastern	State	$42,600.00	$72,100.00	$54,100.00	$122,000.00
freq	1	67	164	6	5	5	7

```python
salaries_unique.dtypes
```

```
school_name                object
Region                     object
school_type                object
starting_median_salary     object
midCareer_median_salary    object
midCareer_25th_salary      object
midCareer_75th_salary      object
dtype: object
```

All of the columns in our dataframe are of the Object type. We will need to cast the salary fields to be of type 'numeric', so we can properly work with them and apply functions/transformations.

Casting Variables

```python
#removing special characters from the salary fields so they can be cast to numeric
for col in ['starting_median_salary','midCareer_median_salary','midCareer_25th_salary','midCareer_75th_salary']
    salaries_unique[col] = salaries_unique[col].str.replace(',','')
    salaries_unique[col] = salaries_unique[col].str.replace('$','')
    salaries_unique[col] = salaries_unique[col].str.replace('\.00','')
```

```
<ipython-input-921-62552876a08c>: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#ret
urning-a-view-versus-a-copy
  salaries_unique[col] = salaries_unique[col].str.replace(',','')
<ipython-input-921-62552876a08c>:4: FutureWarning: The default value of regex will change from True to False in
a future version. In addition, single character regular expressions will*not* be treated as literal strings whe
n regex=True.
  salaries_unique[col] = salaries_unique[col].str.replace('$','')
<ipython-input-921-62552876a08c>:4: 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#ret
urning-a-view-versus-a-copy
  salaries_unique[col] = salaries_unique[col].str.replace('$','')
<ipython-input-921-62552876a08c>:5: FutureWarning: The default value of regex will change from True to False in
a future version.
  salaries_unique[col] = salaries_unique[col].str.replace('\.00','')
<ipython-input-921-62552876a08c>:5: 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#ret
urning-a-view-versus-a-copy
  salaries_unique[col] = salaries_unique[col].str.replace('\.00','')
```

```python
#casting salary columns to be of type numeric
salaries_unique[["starting_median_salary", "midCareer_median_salary",'midCareer_25th_salary',"midCareer_75th_sa
```

```
C:\Users\phill\Anaconda3\lib\site-packages\pandas\core\frame.py:3191: 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#ret
urning-a-view-versus-a-copy
  self[k1] = value[k2]
```

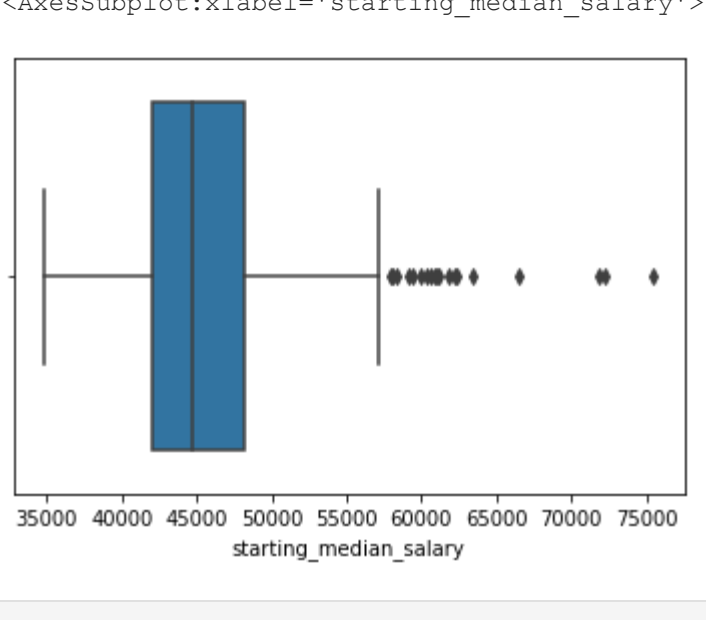
Identifying Outliers

```python
salaries_unique.head()
```

	school_name	Region	school_type	starting_median_salary	midCareer_median_salary	midCareer_25th_salary	midCareer_75th_salary
214	Amherst College	Northeastern	Liberal Arts	54500	107000	84900	162000
189	Appalachian State University	Southern	State	40400	69100	50400	90800
34	Arizona State University (ASU)	Western	Party	47400	84100	60700	114000
194	Arkansas State University (ASU)	Southern	State	38700	63300	45300	83900
149	Auburn University	Southern	State	45400	84700	62700	109000

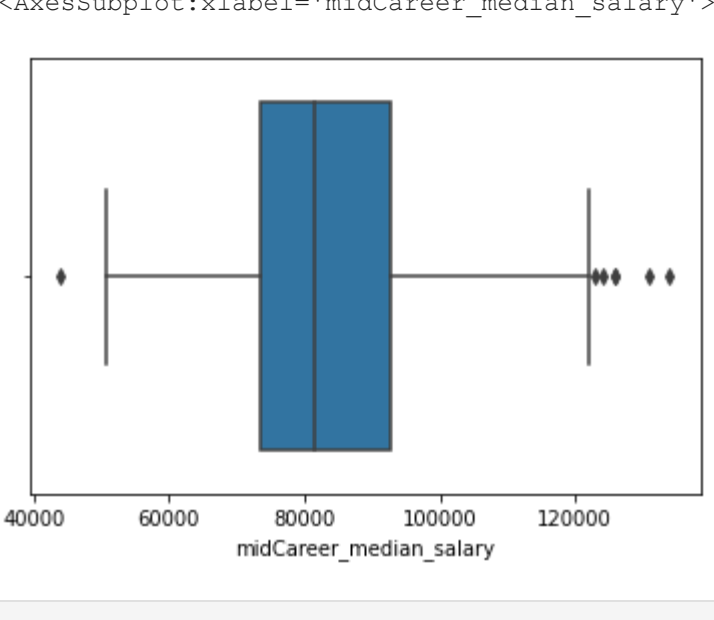
```python
#boxplot of salary columns

#Starting Median Salary
sns.boxplot(x=salaries_unique['starting_median_salary'])
```

```
<AxesSubplot:xlabel='starting_median_salary'>
```



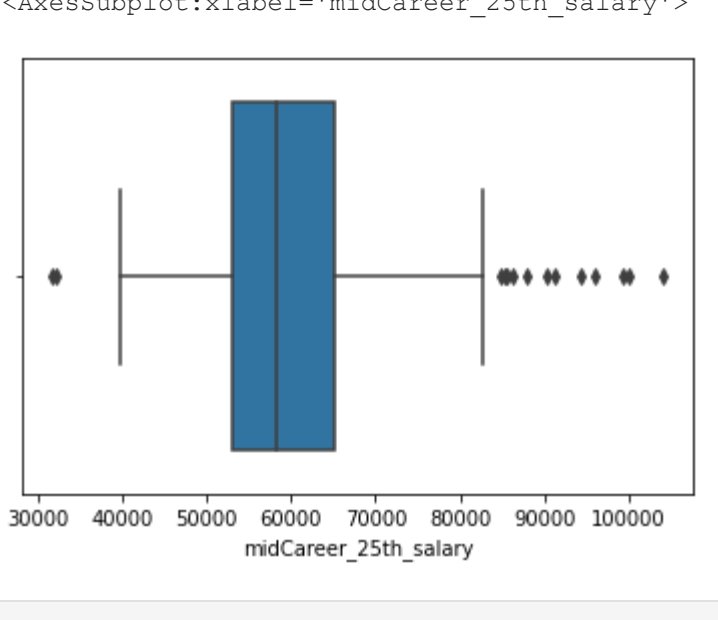

```python
#Mid-Career Median Salary
sns.boxplot(x=salaries_unique['midCareer_median_salary'])
```

```
<AxesSubplot:xlabel='midCareer_median_salary'>
```



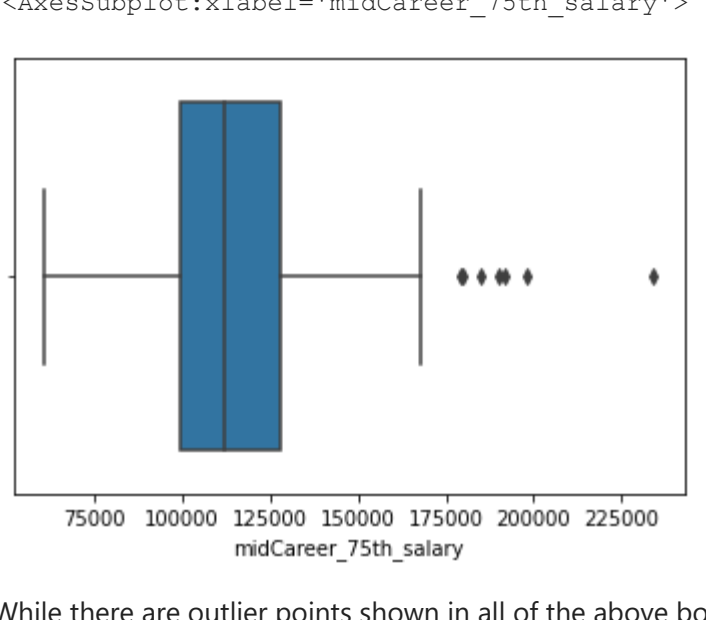

```python
#Mid-Career 25th Percentile Salary
sns.boxplot(x=salaries_unique['midCareer_25th_salary'])
```

```
<AxesSubplot:xlabel='midCareer_25th_salary'>
```




```python
#Mid-Career 75th Percentile Salary
sns.boxplot(x=salaries_unique['midCareer_75th_salary'])
```

```
<AxesSubplot:xlabel='midCareer_75th_salary'>
```



While there are outlier points shown in all of the above boxplots, I don't view any of them as being outliers.

Salary is a monetary value that varies immensely, especially depending on one's area of expertise, age, years working, etc. I think universities develop students who then go into many different career areas, which generate different degrees of salaries, and so, I don't think any of these salary values can be considered as outliers.

I think all of them will be useful in this analysis of understanding how universities can have an impact on the career and money that one makes post-grad.

Removing Parantheses from School Names

```python
salaries_unique['school_name'] = salaries_unique['school_name'].str.replace(r"\s*\([^()]*\)","").str.strip()
```

```
<ipython-input-928-106be6c012d3>:1: FutureWarning: The default value of regex will change from True to False in
a future version.
  salaries_unique['school_name'] = salaries_unique['school_name'].str.replace(r"\s*\([^()]*\)","").str.strip()
<ipython-input-928-106be6c012d3>:1: 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#ret
urning-a-view-versus-a-copy
  salaries_unique['school_name'] = salaries_unique['school_name'].str.replace(r"\s*\([^()]*\)","").str.strip()
```

```python
salaries_unique.head()
```

	school_name	Region	school_type	starting_median_salary	midCareer_median_salary	midCareer_25th_salary	midCareer_75th_salary
214	Amherst College	Northeastern	Liberal Arts	54500	107000	84900	162000
189	Appalachian State University	Southern	State	40400	69100	50400	90800
34	Arizona State University	Western	Party	47400	84100	60700	114000
194	Arkansas State University	Southern	State	38700	63300	45300	83900
149	Auburn University	Southern	State	45400	84700	62700	109000

Milestone 3

Cleaning/Formatting Website Data

```python
#needed libraries
import requests
import lxml.html as lh
import pandas as pd
```

Reading in Website Data

```python
#function to fetch the rows of the rankings table from the website I found
#the rows can be found between the <tr> tags in the underlying HTML
def get_tableData_Pages():
    #target we want to open
    URL = ['https://oedb.org/rankings/acceptance-rate/','https://oedb.org/rankings/acceptance-rate/page/2/#tabl

    tr_pages = []

    for url in range(0,6):
        #open with GET method
        resp=requests.get(URL[url])

        #check for response 200 -> OK
        if resp.status_code==200:
            print("Successfully opened the web page")
            print("Fetching elements")

            #store the contents of the website under a doc
            doc = lh.fromstring(resp.content)

            #parse data that are stored between <tr> tags of HTML --> each row in the table
            tr_elements = doc.xpath('//tr')

            tr_pages.append(tr_elements)
        else:
            print("Error")

    return tr_pages
```

```python
tr_pages = get_tableData_Pages()
```

```
Successfully opened the web page
Fetching elements
Successfully opened the web page
Fetching elements
Successfully opened the web page
Fetching elements
Successfully opened the web page
Fetching elements
Successfully opened the web page
Fetching elements
Successfully opened the web page
Fetching elements
```

Parsing the Table Header

```python
#function to fetch the rows of the rankings table from the website I found
#the rows can be found between the <tr> tags in the underlying HTML
def get_tableData():
    #target we want to open
    url = 'https://oedb.org/rankings/acceptance-rate/'
    #open with GET method
    resp=requests.get(url)

    #check for response 200 -> OK
    if resp.status_code==200:
        print("Successfully opened the web page")
        print("Fetching elements")

        #store the contents of the website under a doc
        doc = lh.fromstring(resp.content)

        #parse data that are stored between <tr> tags of HTML --> each row in the table
        tr_elements = doc.xpath('//tr')

        return tr_elements
    else:
        print("Error")
```

```python
tr_elements = get_tableData()
```

```
Successfully opened the web page
Fetching elements
```

```python
#Create empty list
col=[]
i=0
#For each row, store each first element (header) and an empty list
for t in tr_elements[0]:
    i+=1
    #getting the text for the various columns
    name=t.text_content()
    #output column number and column text
    print (i,name)
    col.append((name,[]))
```

```
1 Rank
2 School
3 Student to Faculty Ratio
4 Graduation Rate
5 Retention Rate
6 Acceptance Rate
7 Enrollment Rate
8 Institutional Aid Rate
9 Default Rate
```

Creating Pandas DataFrame

```python
#go through each table on each page of the website
for tbody in tr_pages:
    #first row is the header, data is stored from the second row onwards
    #loop through each row in each table
    for j in range(1,len(tbody)):
        #T is our j'th row
        T=tbody[j]

       # If row is not of size 15, the //tr data is not from our table
        if len(T)!=15:
            break

        #selecting first 9 elements since they align with headers --> other columns are fluff
        T = T[:9]

        #i is the index of our column
        i=0

        #Iterate through each element of the row
        for t in T:
            data=t.text_content()
            #Check if row is empty
            if i>0:
            #Convert any numerical value to integers
                try:
                    data=int(data)
                except:
                    pass
            #Append the data to the empty list of the i'th column
            col[i][1].append(data)
            #Increment i for the next column
            i+=1
```

```python
#creating the dataframe from a dict --> dict will hold the column name and the designated values that fall into
Dict={title:column for (title,column) in col}
df=pd.DataFrame(Dict)
```

In [938…:

```
print(df.shape)
print("571 rows and 9 columns")
```

```
(571, 9)
571 rows and 9 columns
```

In [939…:

```
print(df.head())
```

```
  Rank                       School Student to Faculty Ratio Graduation Rate  \
0    1           Harvard University                   7 to 1             98%
1    2              Yale University                   6 to 1             97%
2    3 University of Pennsylvania                     6 to 1             95%
3    4     Johns Hopkins University                  10 to 1             94%
4    5           Cornell University                   9 to 1             93%

  Retention Rate Acceptance Rate Enrollment Rate Institutional Aid Rate  \
0            98%              6%              4%                    44%
1            99%              7%              5%                    52%
2            98%             10%              7%                    54%
3            97%             14%              5%                    51%
4            97%             15%              8%                    55%

  Default Rate
0          N/A
1          N/A
2          N/A
3          N/A
4          N/A
```

In [940…:

```
print(df.tail())
```

```
    Rank                                   School Student to Faculty Ratio  \
566  567                 Touro University Worldwide                  13 to 1
567  568                             Unitek College                  16 to 1
568  569               University of Western States                  16 to 1
569  570                   Virginia Baptist College                   5 to 1
570  571  West Virginia Junior College-Morgantown                   25 to 1

    Graduation Rate Retention Rate Acceptance Rate Enrollment Rate  \
566             N/A           100%             N/A             N/A
567             N/A           100%             N/A             N/A
568             88%            N/A             N/A             N/A
569            100%            25%             N/A             N/A
570             53%            69%             N/A             N/A

    Institutional Aid Rate Default Rate
566                    76%           4%
567                    20%          N/A
568                    56%          N/A
569                    38%          N/A
570                    88%          N/A
```

Renaming Column Names

In [941…:

```
#renaming the columns with spaces so they are easier to call/access
#ratio, rate columns
df = df.rename(columns={'Student to Faculty Ratio':'Stud_Fac_Ratio','Graduation Rate':'Grad_Rate','Retention Ra
```

In [942…:

```
#looking at the first of data, new columns are shown
df.iloc[0]
```

Out[942…:

```
Rank                               1
School            Harvard University
Stud_Fac_Ratio                7 to 1
Grad_Rate                        98%
Reten_Rate                       98%
Accept_Rate                       6%
Enroll_Rate                       4%
Inst_Aid_Rate                    44%
Default_Rate                     N/A
Name: 0, dtype: object
```

Finding Duplicates

In [943…:

```
#using the DataFrame method duplicated to determine whether each row is a duplicate
df.duplicated()
```

Out[943…:

```
0      False
1      False
2      False
3      False
4      False
       ...
566    False
567    False
568    False
569    False
570    False
Length: 571, dtype: bool
```

In [944…:

```
#count the number of duplicates
df.duplicated().sum()
```

Out[944…:

```
0
```

In [945…:

```
print("No duplicates were found! All unique rows were loaded into the Data Frame")
```

```
No duplicates were found! All unique rows were loaded into the Data Frame
```

Finding Missing Data

In [946…:

```
#loop through columns in dataframe
#check for any NaN values
for col in df.columns:
    print(col + ": " + str(df[col].isnull().values.any()))
```

```
Rank: False
School: False
Stud_Fac_Ratio: False
Grad_Rate: False
Reten_Rate: False
Accept_Rate: False
Enroll_Rate: False
Inst_Aid_Rate: False
Default_Rate: False
```

In [947…:

```
print("No missing data was found, but I can see NA's in the dataset. Going to look into those!")
```

```
No missing data was found, but I can see NA's in the dataset. Going to look into those!
```

Replacing N/A values with np.NaN

In [948…:

```
#replacing N/A values with np.NaN so they are recognized as missing values
df = df.replace('N/A',np.NaN)
```

In [949…:

```
#checking for missing data again now that NaN values are in place
for col in df.columns:
    print(col + ": " + str(df[col].isnull().values.any()))
```

```
Rank: False
School: False
Stud_Fac_Ratio: False
Grad_Rate: True
Reten_Rate: True
Accept_Rate: True
Enroll_Rate: True
Inst_Aid_Rate: False
Default_Rate: True
```

In [950…:

```
print("Multiple columns have missing data: Graduation Rate, Retention Rate, Acceptance Rate, Enrollment Rate, a
```

```
Multiple columns have missing data: Graduation Rate, Retention Rate, Acceptance Rate, Enrollment Rate, and Defa
ult Rate
```

In [951…:

```
#getting counts of missing values in data frame
df.isnull().sum()
```

Out[951…:

```
Rank                0
School              0
Stud_Fac_Ratio      0
Grad_Rate           6
Reten_Rate          4
Accept_Rate        29
Enroll_Rate        29
Inst_Aid_Rate       0
Default_Rate      291
dtype: int64
```

Checking & Changing Data Types

In [952…:

```
#using dtypes function to find data types
df.dtypes
```

Out[952…:

```
Rank              object
School            object
Stud_Fac_Ratio    object
Grad_Rate         object
Reten_Rate        object
Accept_Rate       object
Enroll_Rate       object
Inst_Aid_Rate     object
Default_Rate      object
dtype: object
```

The rate columns are marked as type 'object'. I am going to cast them to be numeric, so we can utilize the numerical values as they represent rates in percentages.

In [953…:

```
def percent_to_float(col):
    df[col] = df[col].str.rstrip('%').astype('float')/100.0
```

In [954…:

```
for col in ['Grad_Rate','Reten_Rate','Accept_Rate','Enroll_Rate','Inst_Aid_Rate','Default_Rate']:
    percent_to_float(col)
```

In [955…:

```
df.dtypes
```

Out[955…:

```
Rank               object
School             object
Stud_Fac_Ratio     object
Grad_Rate         float64
Reten_Rate        float64
Accept_Rate       float64
Enroll_Rate       float64
Inst_Aid_Rate     float64
Default_Rate      float64
dtype: object
```

In [956…:

```
df.head()
```

Out[956…:

	Rank	School	Stud_Fac_Ratio	Grad_Rate	Reten_Rate	Accept_Rate	Enroll_Rate	Inst_Aid_Rate	Default_Rate
0	1	Harvard University	7 to 1	0.98	0.98	0.06	0.04	0.44	NaN
1	2	Yale University	6 to 1	0.97	0.99	0.07	0.05	0.52	NaN
2	3	University of Pennsylvania	6 to 1	0.95	0.98	0.10	0.07	0.54	NaN
3	4	Johns Hopkins University	10 to 1	0.94	0.97	0.14	0.05	0.51	NaN
4	5	Cornell University	9 to 1	0.93	0.97	0.15	0.08	0.55	NaN

Now that the rate values are classified as being of type 'float', we can replace any missing values with aggregated numerical values so as to best fit with the type of the columns.

Filling Missing Values

In [957…:

```
#fill the missing Default Rate data with median value
#round median values to two decimal places for percentage purposes
def fill_na_median(data, inplace=True):
    return data.fillna(round(data.median(),2), inplace=inplace)
```

Default Rate has the most missing values, 291 of them. This count is over half of the total size (number of rows) of the Data Frame. I am going to replace these with the median value for Default Rate, so we can handle any outliers as well.

In [958…:

```
#median value for Default Rate
print('Median Default Rate: ' + str(df['Default_Rate'].median()))
```

```
Median Default Rate: 0.06
```

In [959…:

```
fill_na_median(df['Default_Rate'])
```

In [960…:

```
df['Default_Rate']
```

Out[960…:

```
0      0.06
1      0.06
2      0.06
3      0.06
4      0.06
       ...
566    0.04
567    0.06
568    0.06
569    0.06
570    0.06
Name: Default_Rate, Length: 571, dtype: float64
```

For the other rate values, I will perform the same missing data activity by replacing the missing values with the median values for the corresponding columns.

In [961…:

```
#median values
print('Median Graduation Rate: ' + str(df['Grad_Rate'].median())) #0.66
print('Median Retention Rate: ' + str(df['Reten_Rate'].median())) #0.83
print('Median Acceptance Rate: ' + str(df['Accept_Rate'].median())) #0.65
print('Median Enrollment Rate: ' + str(df['Enroll_Rate'].median())) #0.18
```

```
Median Graduation Rate: 0.66
Median Retention Rate: 0.83
Median Acceptance Rate: 0.645
Median Enrollment Rate: 0.18
```

In [962…:

```
#fill in rest of missing data with median values
fill_na_median(df['Grad_Rate'])
fill_na_median(df['Reten_Rate'])
fill_na_median(df['Accept_Rate'])
fill_na_median(df['Enroll_Rate'])
```

In [963…:

```
#checking to make sure we handled all missing data
#getting counts of missing values in data frame
df.isnull().sum()
```

Out[963…:

```
Rank              0
School            0
Stud_Fac_Ratio    0
Grad_Rate         0
Reten_Rate        0
Accept_Rate       0
Enroll_Rate       0
Inst_Aid_Rate     0
Default_Rate      0
dtype: int64
```

Number of missing values in each column is now 0! We handled the missing data for the rate columns by replacing any missing data with the median value for the columns. The median will allow us to get a value which accounts for outliers, rather than taking the mean.

Detecting and Filtering Outliers

In [964…:

```
#describing the data
#gives us an idea of the distribution
df.describe()
```

Out[964…:

	Grad_Rate	Reten_Rate	Accept_Rate	Enroll_Rate	Inst_Aid_Rate	Default_Rate
count	571.000000	571.000000	571.000000	571.000000	571.000000	571.000000
mean	0.657653	0.827968	0.628546	0.201156	0.719772	0.060893
std	0.130232	0.085167	0.172070	0.120338	0.179739	0.021236
min	0.350000	0.250000	0.060000	0.040000	0.060000	0.010000
25%	0.560000	0.780000	0.530000	0.120000	0.580000	0.060000
50%	0.660000	0.830000	0.650000	0.180000	0.740000	0.060000
75%	0.740000	0.880000	0.750000	0.240000	0.870000	0.060000
max	1.000000	1.000000	1.000000	1.000000	1.000000	0.200000

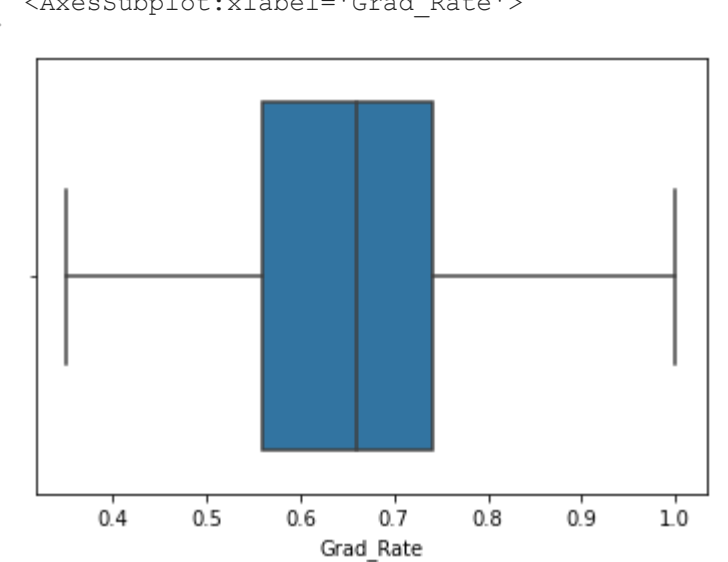
Boxplots of Numerical Fields

Graduation Rate

In [965…:

```
#boxplot of Graduation Rate

#Graduation Rate
sns.boxplot(x=df['Grad_Rate'])
```

Out[965…: <AxesSubplot:xlabel='Grad_Rate'>



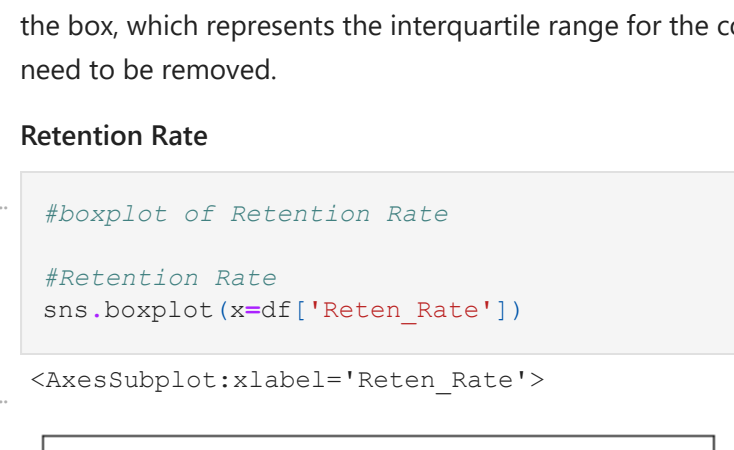
The Graduation Rate is relatively normally distributed. The median value (represented by the vertical line) is pretty much in the middle of the box, which represents the interquartile range for the column. I am pleased with the distribution, and there are no apparent outliers that need to be removed.

Retention Rate

In [966…:

```
#boxplot of Retention Rate

#Retention Rate
sns.boxplot(x=df['Reten_Rate'])
```

Out[966…: <AxesSubplot:xlabel='Reten_Rate'>



The outlier values for Retention Rate fall below ~0.65 or 65%. The minimum is 0.25 or a 25% retention rate, which I think is low for an university but it makes sense from the perspective of university.

This low retention rate can show how there are universities which experience difficulty in keeping students after freshman year, since I'm sure that is a blocker that many schools experience.

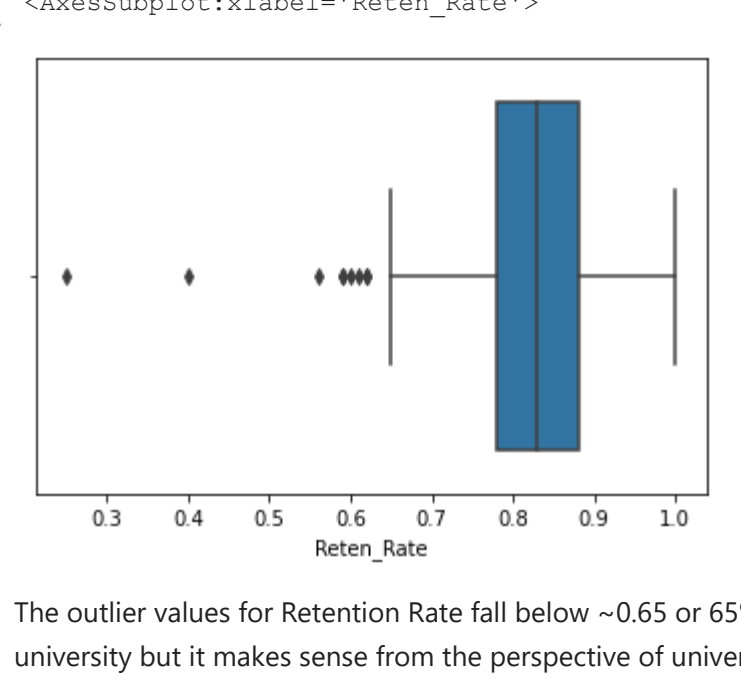
In terms of giving us information in regards to how one's university influences their post-grad salary, I think that knowing if the university they attended experiences trouble with retaining students, then it could give more influence to the student for graduating and still staying at the school.

Acceptance Rate

In [967…:

```
#boxplot of Acceptance Rate

#Acceptance Rate
sns.boxplot(x=df['Accept_Rate'])
```

Out[967…: <AxesSubplot:xlabel='Accept_Rate'>



There are acceptance rate outlier values, which fall below about 0.2 or 20% acceptance rate.

Values falling below this are not surprising as many Ivy League schools have a low acceptance rate, since they are prestigious and more difficult to attend/get into.

Therefore, I will not be removing these outliers that fall below ~0.2 as they give representation to the schools that are more difficult to get accepted into given their prestige.

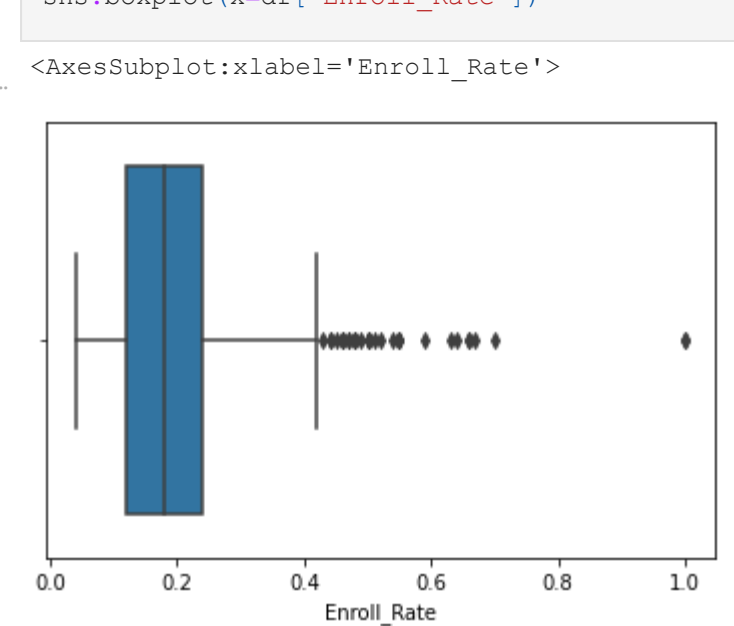
Enrollment Rate

Removing Outliers

In [968…:

```
#boxplot of Enrollment Rate

#Enrollment Rate
sns.boxplot(x=df['Enroll_Rate'])
```

Out[968…: <AxesSubplot:xlabel='Enroll_Rate'>



Enrollment rate represents the percentage of 18-to-24-year-olds enrolled as undergraduate or graduate students in 2- or 4- year institutions.

The outlier values for this field come above ~0.4 or 40% enrollment rate. There are universities with a higher amount of students enrolled in their courses/programs, which is good for that university!

However, for the outlier value at basically 1.0 or 100%, that is difficult for me to fathom that an university has 100% of its original students enrolled for school attendance. It would be very impressive of them, but I am viewing this university with this enrollment rate as an outlier. I am going to remove it.

In [969…:

```
#finding row with enrollment rate greater than 0.8
df[df.Enroll_Rate > 0.8]
```

Out[969…:

	Rank	School	Stud_Fac_Ratio	Grad_Rate	Reten_Rate	Accept_Rate	Enroll_Rate	Inst_Aid_Rate	Default_Rate
539	540	Luther Rice University & Seminary	23 to 1	1.00	1.0	1.0	1.0	0.65	0.06
541	542	Midwives College of Utah	5 to 1	0.66	1.0	1.0	1.0	0.30	0.06

In [970…:

```
#removing these two rows with 1.0 Enroll_Rate
df = df[df.Enroll_Rate <= 0.8]
```

In [971…:

```
print(df.shape)
print("New shape: 569 rows and 9 columns")
```

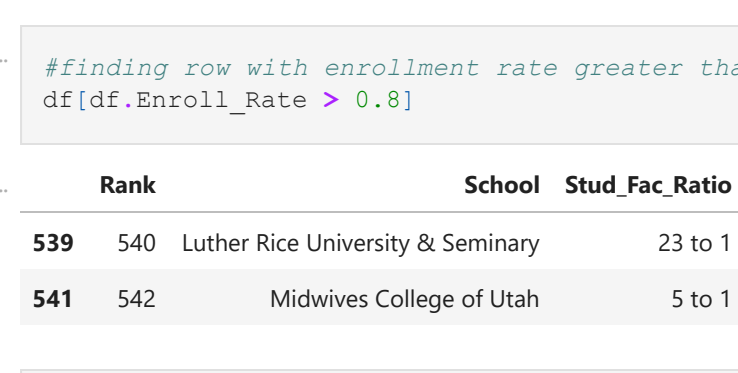
```
(569, 9)
New shape: 569 rows and 9 columns
```

Institutional Aid Rate

In [972…:

```
#boxplot of Institutional Aid Rate

#Institutional Aid Rate
sns.boxplot(x=df['Inst_Aid_Rate'])
```

Out[972…: <AxesSubplot:xlabel='Inst_Aid_Rate'>



Institutional Aid Rate: what percentage of college students get financial aid?

There is a designated outlier in this distribution that is less than about 0.15 or 15%. It looks like it is almost at 0.0. Let's look into the rows.

In [973…:

```
#finding row with institutional aid rate less than 0.1
df[df.Inst_Aid_Rate < 0.15]
```

Out[973…:

	Rank	School	Stud_Fac_Ratio	Grad_Rate	Reten_Rate	Accept_Rate	Enroll_Rate	Inst_Aid_Rate	Default_Rate
557	558	Mountain State College	17 to 1	0.92	0.83	0.65	0.18	0.06	0.18

The designated row actually has an Institutional Aid Rate of 6%, which means that 6% of their college students receive financial aid. This percentage is small, but it could designate that the students don't need financial aid or the university is not equipped/prepared to give it out.

In terms of our business problem of identifying salaries based on university attendance, this could designate that the students from this university are either not in need or they are using other ways to receive aid while in school.

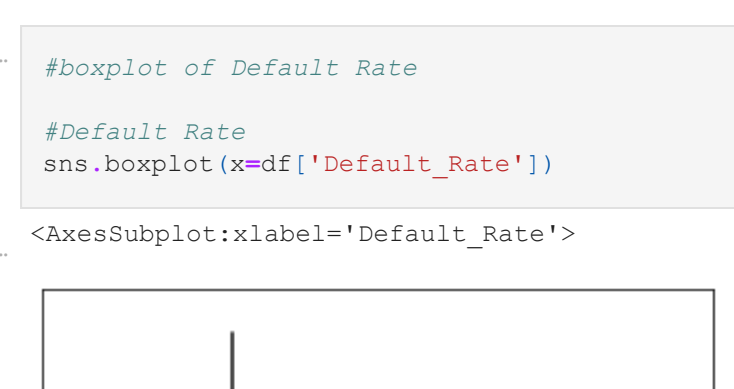
I am going to keep this row in; its other values are representative of data that we want to look into as well.

Default Rate

In [974…:

```
#boxplot of Default Rate

#Default Rate
sns.boxplot(x=df['Default_Rate'])
```

Out[974…: <AxesSubplot:xlabel='Default_Rate'>



Default rate is the percentage of all outstanding loans that a lender has written off as unpaid after a prolonged period of missed payments. In terms of how this relates to an univeristy, it would present the percentage of graduates that default on their student loans in the first 12 months of repayment.

I think this rate value is definitely indicative of how students are doing post-graduation in terms of their incomes/finances, because paying off loans is a huge financial responsibility for students and young adults.

Since the majority of the values in this Default Rate column were replaced with the median value of 0.06, the distribution is not very well distributed and shows any values outside of the median as outliers.

In order to keep variety and distribution for the column, I will not remove any of the outliers as we need to represent the values that do not fall into the median bucket that was used for replacement.

In [975…:

```
df.head()
```

Out[975…:

	Rank	School	Stud_Fac_Ratio	Grad_Rate	Reten_Rate	Accept_Rate	Enroll_Rate	Inst_Aid_Rate	Default_Rate
0	1	Harvard University	7 to 1	0.98	0.98	0.06	0.04	0.44	0.06
1	2	Yale University	6 to 1	0.97	0.99	0.07	0.05	0.52	0.06
2	3	University of Pennsylvania	6 to 1	0.95	0.98	0.10	0.07	0.54	0.06
3	4	Johns Hopkins University	10 to 1	0.94	0.97	0.14	0.05	0.51	0.06
4	5	Cornell University	9 to 1	0.93	0.97	0.15	0.08	0.55	0.06

Converting Categorical Data to Numerical Data

The Student to Faculty Ratio column is marked as being of object type since it represents the ratio values in string form, i.e. "7 to 1".

I am going to create a new column which represents the ratios in numerical form by dividing the first number in the ratio by the second number.

In [976…:

```
#function to retreieve the numbers in the ratio field from around "to"
#divides the two numbers that come from the split by eachother
#returns numerical ratio
def convertRatio(x):
    a,b = x.split('to')
    c = int(a)/int(b)
    return c
```

In [977…:

```
#create new column in DataFrame --> Student:Faculty Ratio as a number
#applies the above function on the object column from the original df
df['Stud_Fac_Ratio_Num'] = df['Stud_Fac_Ratio'].apply(convertRatio)
```

In [978…:

```
#new ratio number column
df['Stud_Fac_Ratio_Num']
```

Out[978…:

```
0       7.0
1       6.0
2       6.0
3      10.0
4       9.0
       ...
566    13.0
567    16.0
568    16.0
569     5.0
570    25.0
Name: Stud_Fac_Ratio_Num, Length: 569, dtype: float64
```

In [979…:

```
#view first 5 rows in data frame with new column
df.head()
```

Out[979…:

	Rank	School	Stud_Fac_Ratio	Grad_Rate	Reten_Rate	Accept_Rate	Enroll_Rate	Inst_Aid_Rate	Default_Rate	Stud_Fac_Ratio_Num
0	1	Harvard University	7 to 1	0.98	0.98	0.06	0.04	0.44	0.06	7.0
1	2	Yale University	6 to 1	0.97	0.99	0.07	0.05	0.52	0.06	6.0
2	3	University of Pennsylvania	6 to 1	0.95	0.98	0.10	0.07	0.54	0.06	6.0
3	4	Johns Hopkins University	10 to 1	0.94	0.97	0.14	0.05	0.51	0.06	10.0
4	5	Cornell University	9 to 1	0.93	0.97	0.15	0.08	0.55	0.06	9.0

Putting Numerical Ratio Column next to Categorical Ratio Column

In [980…:

```
df = df[['Rank','School','Stud_Fac_Ratio','Stud_Fac_Ratio_Num','Grad_Rate','Reten_Rate','Accept_Rate','Enroll_F
```

In [981…:

```
#rearrangement of columns --> view
df.head()
```

Out[981…:

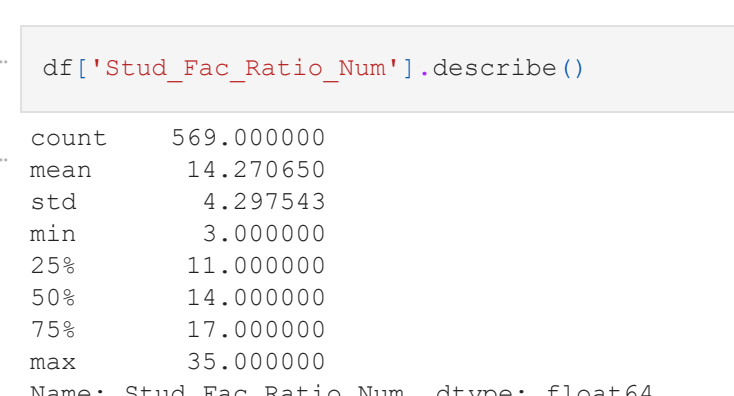
	Rank	School	Stud_Fac_Ratio	Stud_Fac_Ratio_Num	Grad_Rate	Reten_Rate	Accept_Rate	Enroll_Rate	Inst_Aid_Rate	Default_Rate
0	1	Harvard University	7 to 1	7.0	0.98	0.98	0.06	0.04	0.44	0.06
1	2	Yale University	6 to 1	6.0	0.97	0.99	0.07	0.05	0.52	0.06
2	3	University of Pennsylvania	6 to 1	6.0	0.95	0.98	0.10	0.07	0.54	0.06
3	4	Johns Hopkins University	10 to 1	10.0	0.94	0.97	0.14	0.05	0.51	0.06
4	5	Cornell University	9 to 1	9.0	0.93	0.97	0.15	0.08	0.55	0.06

Checking for Outliers in Numerical Ratio field

In [982…:

```
#boxplot of Student:Faculty Ratio as a number

#Stud_Fac_Ratio_Num
sns.boxplot(x=df['Stud_Fac_Ratio_Num'])
```

Out[982…: <AxesSubplot:xlabel='Stud_Fac_Ratio_Num'>



In [983…:

```
df['Stud_Fac_Ratio_Num'].describe()
```

Out[983…:

```
count    569.000000
mean      14.270650
std        4.297543
min        3.000000
25%       11.000000
50%       14.000000
75%       17.000000
max       35.000000
Name: Stud_Fac_Ratio_Num, dtype: float64
```

The distribution of values for student to faculty ratio is relatively normally distrubted, with the median value basically being in the middle of the interquartile range. The majority of values fall between 3.0 and 25.0 (3 students to 1 teacher, 25 students to 1 teacher).

There are outlier values shown that come above 25. However, it is not unlikely for some universities to have larger class sizes, especially depending on the overall class population, type of class, number of professors, etc.

I will not be removing these marked outliers from the boxplot.

Dropping Unneeded Columns

Since we don't need the categorical and the numerical columns for representing student to faculty ratio, I am going to drop the categorical one from the copied dataset so we can have all numerical columns, which is helpful for modeling!

In [984…:

```
#taking a copy of the dataframe with the certain columns removed
df_dropped = df.drop(['Stud_Fac_Ratio'],axis=1)
```

In [985…:

```
#print shape of the new dataframe --> should be same as original since we are essentially replacing a column
print(df_dropped.shape)
print("569 rows and 9 columns. Same as original shape!")
```

```
(569, 9)
569 rows and 9 columns. Same as original shape!
```

In [986…:

```
print(df_dropped.head())
```

```
  Rank                       School  Stud_Fac_Ratio_Num  Grad_Rate  Reten_Rate  \
0    1           Harvard University                 7.0       0.98        0.98
1    2              Yale University                 6.0       0.97        0.99
2    3 University of Pennsylvania                   6.0       0.95        0.98
3    4     Johns Hopkins University                10.0       0.94        0.97
4    5           Cornell University                 9.0       0.93        0.97

   Accept_Rate  Enroll_Rate  Inst_Aid_Rate  Default_Rate
0         0.06         0.04           0.44          0.06
1         0.07         0.05           0.52          0.06
2         0.10         0.07           0.54          0.06
3         0.14         0.05           0.51          0.06
4         0.15         0.08           0.55          0.06
```

Milestone 4

Step #1: Connecting to an API/Pulling in the Data and Cleaning/Formatting

In [987…:

```
#import required libaries
import urllib.parse
import urllib.error
import json
import os
import certifi
import ssl
from urllib.request import Request, urlopen
import requests
```

Load the secret API key from a JSON file stored in the same folder in a variable, by using json.loads

In [988…:

```
#converting text file to JSON
filename = 'project/APIkeys.txt'
#dictionary where the lines from text will be stored
dict1 = {}
#creating dictionary
with open(filename) as fh:
    for line in fh:
        #read each line and trim of extra spaces, only gets words
        command, description = line.strip().split(None,1)
        dict1[command]=description.strip()

#create JSON file
out_file = open("project/APIkeys_college.json","w")
json.dump(dict1,out_file,indent=4,sort_keys=False)
out_file.close()
```

Open the APIkeys.json file

In [989…:

```
#open the APIkeys.json file which holds api key
with open('project/APIkeys_college.json') as f:
    keys = json.load(f)
    #retrieve API key to feed into URL
    schoolkey = keys['Collegeapi']
```

In [990…:

```
apikey = schoolkey
```

Putting together URL with API Key

In [991…:

```
#The College Scorecard API is a GET API that lives at http://api.data.gov/ed/collegescorecard/
#The endpoint for querying all data is /v1/schools
base_url = "https://api.data.gov/ed/collegescorecard/v1/schools?"
base_url = f"{base_url}api_key={apikey}&fields="
base_url
```

Out[991…:

```
'https://api.data.gov/ed/collegescorecard/v1/schools?api_key=90UjJlueaK4Tuy92KYVQfJoJzzJ83jMDeNDJk7vr&fields='
```

Defining Search Conditions for the URL query

In [992…:

```
# List of all the search conditions
parameters = ["&school.degrees_awarded.predominant=3",
              "&school.operating=1"
              ]

# Appending all the conditions values to construct the conditionss_url
parameters_url = ""
for parameter in parameters:
    parameters_url = parameters_url + parameter
parameters_url
```

Out[992…:

```
'&school.degrees_awarded.predominant=3&school.operating=1'
```

Step #2: Define Header

Pulling Needed Fields from College Scorecard Data & Renaming Columns

Out[103…: (2006, 39)

Loading datasets into individual tables

In [103…:
```python
#import SQLite library
import sqlite3 as sql
```

salaries_unique DataFrame --> salaries-by-region.csv

In [103…:
```python
#open a connection to a new database for all data
conn = sql.connect('univ_sal.db')
```

In [103…:
```python
#create a new table in the univ_data database for CSV data
#salaries_unique.to_sql('grad_salaries',conn)
```

In [103…:
```python
#testing table connection
grad_salaries = pd.read_sql('SELECT * FROM grad_salaries',conn)
```

In [103…:
```python
grad_salaries.head()
```

Out[103…:

	index	school_name	Region	school_type	starting_median_salary	midCareer_median_salary	midCareer_25th_salary	midCareer_75th_salar
0	214	Amherst College	Northeastern	Liberal Arts	54500	107000	84900	16200
1	189	Appalachian State University	Southern	State	40400	69100	50400	9080
2	34	Arizona State University	Western	Party	47400	84100	60700	11400
3	194	Arkansas State University	Southern	State	38700	63300	45300	8390
4	149	Auburn University	Southern	State	45400	84700	62700	10900

df --> 'https://oedb.org/rankings/acceptance-rate/'

In [103…:
```python
#create new table in database for website data
#df_dropped.to_sql('univ_rates',conn)
```

school_df --> API data "https://api.data.gov/ed/collegescorecard/v1/schools?"

In [104…:
```python
#create new table in database for API data
#school_df.to_sql('schools',conn)
```

Merging 3 Tables into One Dataset

In [104…:
```python
#initiate cursor for executing SQL
cur = conn.cursor()
```

In [104…:
```python
#takes the schools dataset (API data) and joins all its rows to the rates table
#each university should have one corresponding rank and rates information
#join universities and their ranks to the information on salaries as only certain ones were included in the CSV
#cur.execute('''SELECT *
#               FROM schools
#               LEFT JOIN univ_rates ON schools.school_name = univ_rates.School
#               LEFT JOIN grad_salaries ON univ_rates.School = grad_salaries.school_name
#               ''')
```

In [104…:
```python
cur.execute('''SELECT *
               FROM grad_salaries
               LEFT JOIN schools ON schools.school_name = grad_salaries.school_name
               LEFT JOIN univ_rates ON univ_rates.School = schools.school_name
               ''')
```

Out[104…: <sqlite3.Cursor at 0x24901d550a0>

In [104…:
```python
data= pd.DataFrame(cur.fetchall()) #converts SQL query results into dataframe format
data.columns = [x[0] for x in cur.description] #labels the columns of the dataframe
data
```

Out[104…:

	index	school_name	Region	school_type	starting_median_salary	midCareer_median_salary	midCareer_25th_salary	midCareer_75th_sa
0	214	Amherst College	Northeastern	Liberal Arts	54500	107000	84900	162
1	189	Appalachian State University	Southern	State	40400	69100	50400	90
2	34	Arizona State University	Western	Party	47400	84100	60700	114
3	194	Arkansas State University	Southern	State	38700	63300	45300	83
4	149	Auburn University	Southern	State	45400	84700	62700	109
...	...	...	...	...	...	...	...	
245	44	Whitman College	Western	Liberal Arts	43500	80100	64800	111
246	220	Williams College	Northeastern	Liberal Arts	51700	102000	76400	143
247	96	Wittenberg University	Midwestern	Liberal Arts	39200	78200	54100	131
248	206	Worcester Polytechnic Institute	Northeastern	Engineering	61000	114000	91200	137
249	201	Yale University	Northeastern	Ivy League	59100	126000	80600	198

250 rows × 58 columns

In [104…:
```python
#converting 'Rank' to be of float type
data['Rank'] = data['Rank'].astype(float, errors = 'raise')
```

Visualizations

In [104…:
```python
import matplotlib.pyplot as plt
import seaborn as sns
```

Handling Column Names in Merged Dataset

In [105…:
```python
#renaming 'school_name' columns to be different
cols = []
count = 1
for column in data.columns:
    if column == 'school_name':
        cols.append(f'school_name_{count}')
        count+=1
        continue
    cols.append(column)
data.columns = cols
```

In [105…:
```python
data.school_name_1
```

Out[105…:
```
0                    Amherst College
1         Appalachian State University
2              Arizona State University
3             Arkansas State University
4                   Auburn University
                    ...
245                  Whitman College
246                 Williams College
247            Wittenberg University
248    Worcester Polytechnic Institute
249                  Yale University
Name: school_name_1, Length: 250, dtype: object
```

In [105…:
```python
data.head()
```

Out[105…:

	index	school_name_1	Region	school_type	starting_median_salary	midCareer_median_salary	midCareer_25th_salary	midCareer_75th_sal
0	214	Amherst College	Northeastern	Liberal Arts	54500	107000	84900	162
1	189	Appalachian State University	Southern	State	40400	69100	50400	90
2	34	Arizona State University	Western	Party	47400	84100	60700	114
3	194	Arkansas State University	Southern	State	38700	63300	45300	83
4	149	Auburn University	Southern	State	45400	84700	62700	109

5 rows × 58 columns

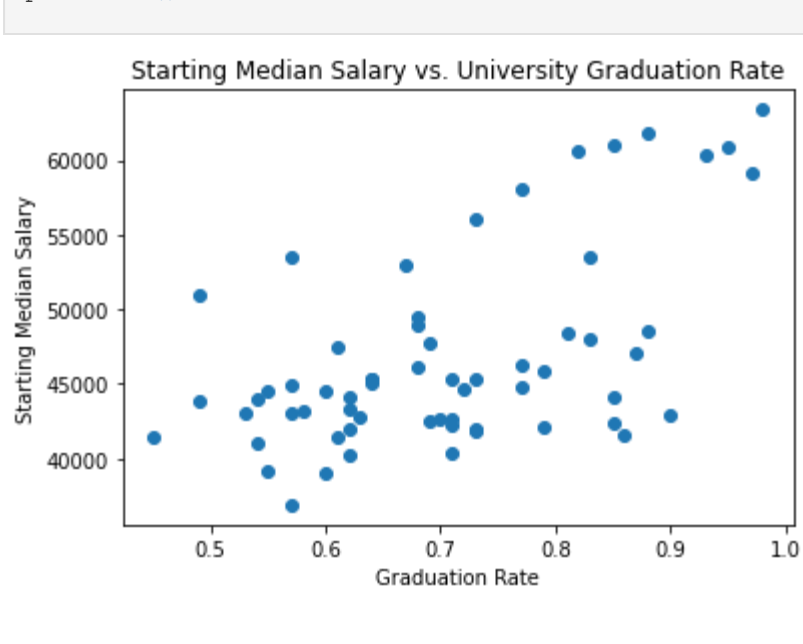
In [105…:
```python
data.columns
```

Out[105…:
```
Index(['index', 'school_name_1', 'Region', 'school_type',
       'starting_median_salary', 'midCareer_median_salary',
       'midCareer_25th_salary', 'midCareer_75th_salary', 'index',
       'school_name_2', 'school_id', 'school_state', 'school_ownership',
       'full_time_faculty_rate', 'faculty_avg_sal_monthly', 'stud_enroll_size',
       'percent_male_stud', 'percent_fem_stud', '4_yr_retention',
       'attendance_cost_per_year', '150%_completion_rate_4yr',
       'admission_rate', 'sat_avg_overall', 'sat_75th_percentile_math',
       'sat_75th_percentile_reading', 'mean_earnings_6yrs',
       'mean_male_earning_6yrs)', 'mean_fem_earning_6yrs', 'std_earning_6yrs',
       'percent_above_25k_6yrs', 'low_income_6yrs', 'medium_income_6yrs',
       'high_income_6yrs', 'low_mean_earn_6yrs', 'med_mean_earn_6yrs',
       'high_mean_earn_6yrs', 'mean_earnings_10yrs', 'mean_male_earn_10yrs',
       'mean_fem_earn_10yrs', 'std_earn_10yrs', 'percent_above_25k_10years',
       'low_income_10yrs', 'medium_income_10yrs', 'high_income_10yrs',
       'low_mean_earn_10yrs', 'med_mean_earn_10yrs', 'high_mean_earn_10yrs',
       'school_ownership_cat', 'index', 'Rank', 'School', 'Stud_Fac_Ratio_Num',
       'Grad_Rate', 'Reten_Rate', 'Accept_Rate', 'Enroll_Rate',
       'Inst_Aid_Rate', 'Default_Rate'],
      dtype='object')
```

Visualization #1: Scatter Plot Between Graduation Rate and Starting Median Salary

In [105…:
```python
x = data['Grad_Rate']
y = data['starting_median_salary']
```

In [105…:
```python
plt.scatter(x,y)
plt.xlabel('Graduation Rate')
plt.ylabel('Starting Median Salary')
plt.title('Starting Median Salary vs. University Graduation Rate')
plt.show()
```



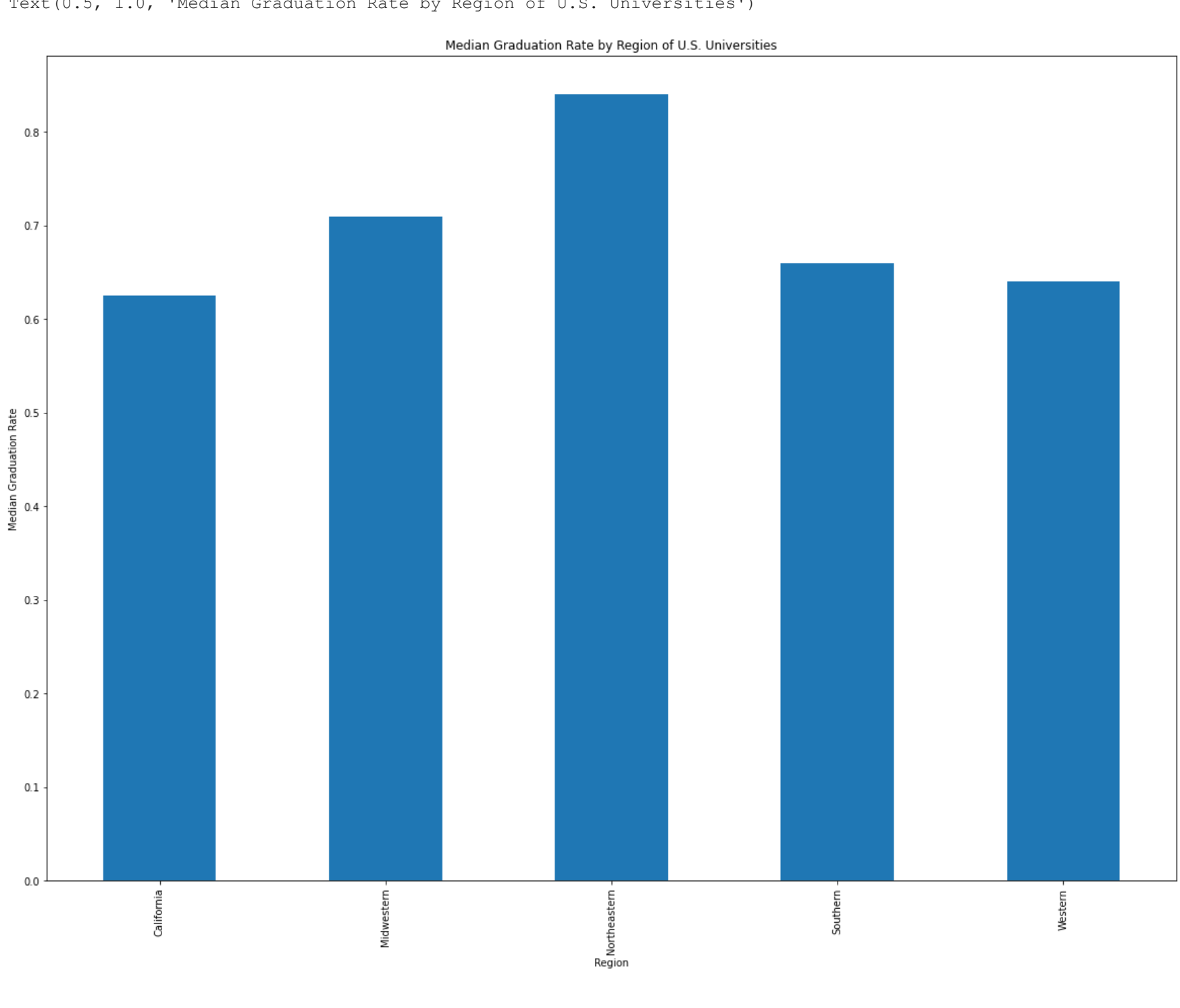
Visualization #2 - Median Graduation Rates Per Region Bar Plot

In [105…:
```python
#number of universities per region --> minimizing effect of sample size
data.groupby(['Region'])['Grad_Rate'].median()
```

Out[105…:
```
Region
California      0.625
Midwestern      0.710
Northeastern    0.840
Southern        0.660
Western         0.640
Name: Grad_Rate, dtype: float64
```

In [105…:
```python
#side-by-side bar chart grouped on year and Region
#Values shown are sums of Units Sold
data.groupby(['Region'])['Grad_Rate'].median().plot(kind='bar', stacked=False,figsize=(20,15))
plt.ylabel('Median Graduation Rate')
plt.xlabel('Region')
plt.title('Median Graduation Rate by Region of U.S. Universities')
```

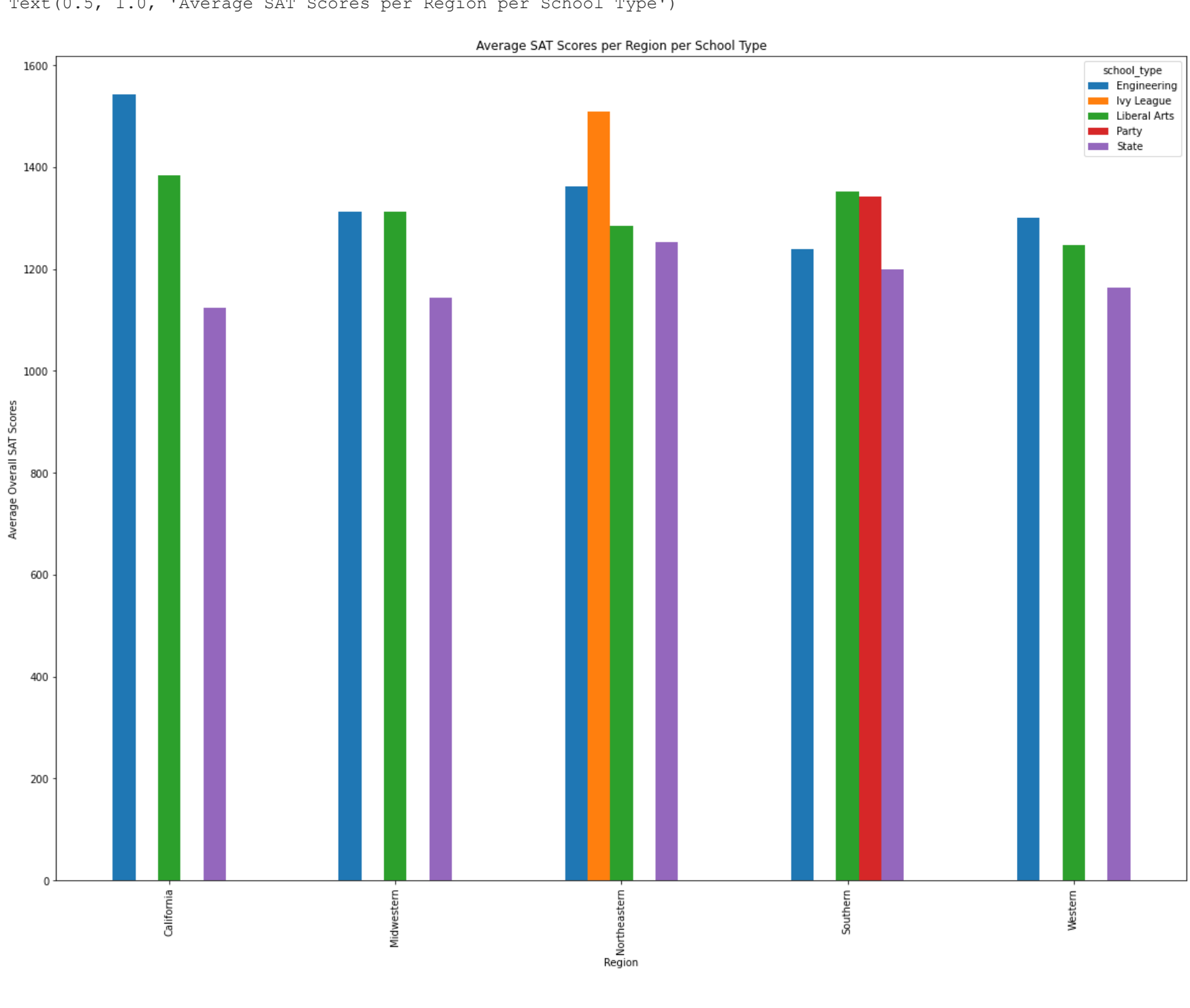
Out[105…: Text(0.5, 1.0, 'Median Graduation Rate by Region of U.S. Universities')



Visualization #3: Average Overall SAT Scores per Region per School Type

In [105…:
```python
#side-by-side bar chart grouped on region and school type
#Values shown are average SAT Overall
data.groupby(['Region','school_type'])['sat_avg_overall'].mean().unstack().plot(kind='bar', stacked=False,figsi
plt.ylabel('Average Overall SAT Scores')
plt.xlabel('Region')
plt.title('Average SAT Scores per Region per School Type')
```

Out[105…: Text(0.5, 1.0, 'Average SAT Scores per Region per School Type')



Visualization #4: Northeastern Universities' Boxplots of Starting & Mid-Career Salaries

In [105…:
```python
#northeastern subset
north_eastern = data[data['Region']=='Northeastern']
north_eastern.shape
```

Out[105…: (69, 58)

In [106…:
```python
#looking at universities in Northeastern region
north_eastern[['school_name_1','Rank']].sort_values('Rank')

#contains 4 of the universities ranked in the Top 5!
```

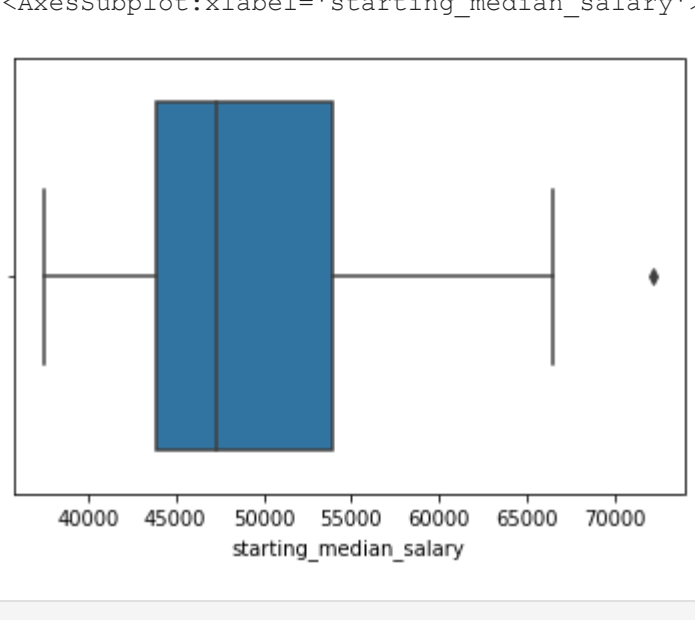
Out[106…:

	school_name_1	Rank
59	Harvard University	1.0
249	Yale University	2.0
200	University of Pennsylvania	3.0
37	Cornell University	5.0
26	Carnegie Mellon University	16.0
...	...	...
229	Ursinus College	NaN
232	Vassar College	NaN
238	Wellesley College	NaN
240	Wesleyan University	NaN
246	Williams College	NaN

69 rows × 2 columns

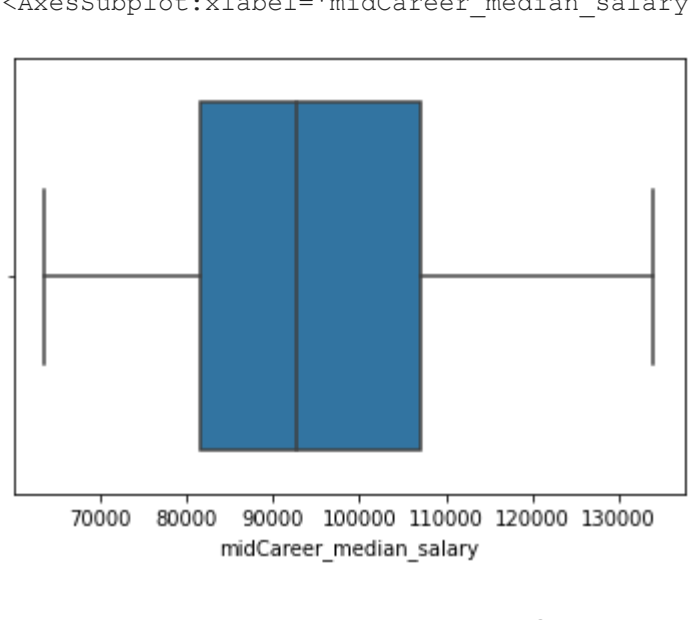
In [106…:
```python
#boxplot for the Northeastern Universities' students' starting median salaries
sns.boxplot(x=north_eastern['starting_median_salary'])
```

Out[106…: <AxesSubplot:xlabel='starting_median_salary'>



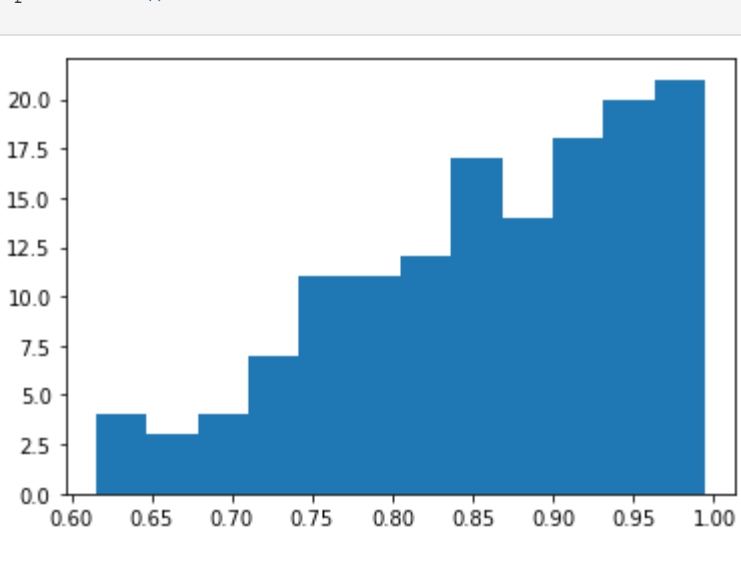
In [106…:
```python
#boxplot for the Northeastern Universities' students' mid-career median salaries
sns.boxplot(x=north_eastern['midCareer_median_salary'])
```

Out[106…: <AxesSubplot:xlabel='midCareer_median_salary'>



Visualization #5: Histogram of 4-Year Retention Rates

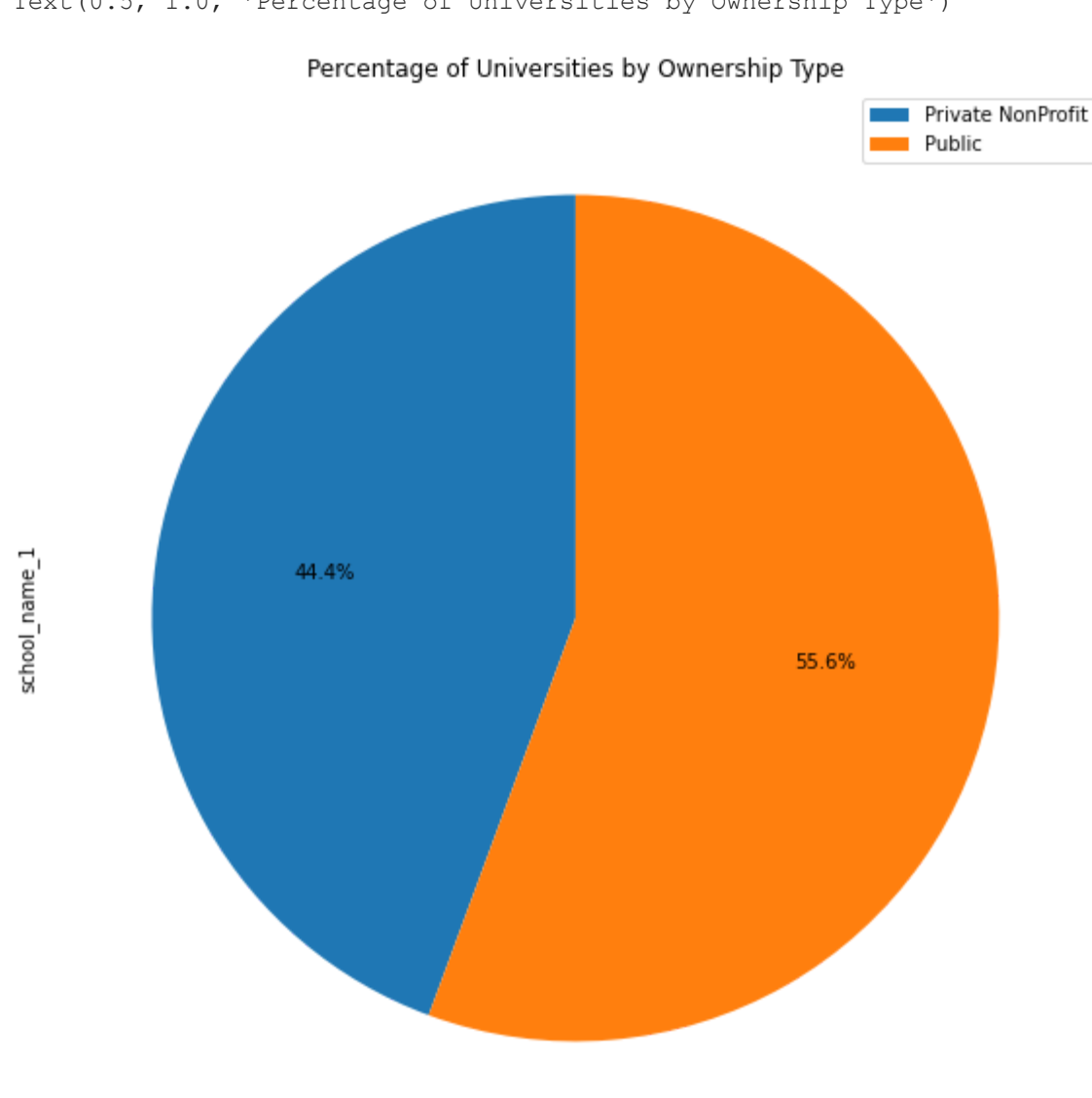
In [106…:
```python
x = data['4_yr_retention']
plt.hist(x,bins=12)
plt.show()
```



Visualization #6: Pie Chart of School Ownership Categories

In [106…:
```python
#pie chart with only ownership grouping
#start angle at 90 degrees -- 12 o'clock
#No labels -- just using legend

#using school name as y since it's the primary key and it holds a value for each survey respondent
data.groupby(['school_ownership_cat']).count().plot(kind='pie',y='school_name_1',figsize=(20,10), autopct='%1.1
#title for pie chart
plt.title('Percentage of Universities by Ownership Type')
```

Out[106…: Text(0.5, 1.0, 'Percentage of Universities by Ownership Type')



In [ ]: