Course Project Part 1

Nikhil Pereira (nmp54) and Julia Hoffman (jh2334)

February 20, 2022

1. Choosing a data set - College Score Card

In [1]:
```
#from google.colab import drive
import pandas as pd
import warnings
df = pd.read_csv('CS_subset.csv', encoding='latin-1')
warnings.filterwarnings('ignore')
# drive.mount('/content/drive')

# # julia file path
# #colab_path = '/content/drive/MyDrive/LID/CS_subset.csv'

# # nikhil file path
# colab_path = '/content/drive/MyDrive/Code Shared/LID Project/CS_subset.csv'
# #names_path = '/content/drive/MyDrive/Code Shared/LID Project/CS_subset_dict.csv'
# #pd.read_csv('u.item', sep='|', names=m_cols, encoding='latin-1')
# df = pd.read_csv(colab_path, encoding='latin-1')
# drive.mount('/content/drive')
```

```
C:\Users\quick\Anaconda3\envs\cs5781env\lib\site-packages\numpy\_distributor_init.py:32: UserWarning: loaded more than 1 DLL from .libs:
C:\Users\quick\Anaconda3\envs\cs5781env\lib\site-packages\numpy\.libs\libopenblas.NOIJJG62EMASZI6NYURL6JBKM4EVBGM7.gfortran-win_amd64.dll
C:\Users\quick\Anaconda3\envs\cs5781env\lib\site-packages\numpy\.libs\libopenblas.XWYDX2IKJW2NMTWSFYNGFUWKQU3LYTCZ.gfortran-win_amd64.dll
  stacklevel=1)
```

In [2]:

```
df.describe()
```

Out[2]:

	PREDDEG	CONTROL	LOCALE	SATVRMID	SATMTMID	SATWRMID	ACTCMMID	ACTENMID	ACTMTMID	ACTWRMID	...	PPTUG_EF	NPT4_PUB	NPT4_PRIV
count	7804.000000	7804.000000	7380.000000	1301.000000	1315.000000	793.000000	1342.000000	1165.000000	1166.000000	300.000000	...	7072.000000	1923.000000	4753.000000
mean	1.788954	2.216427	19.589024	521.812452	530.771863	521.239596	23.120715	22.724464	22.584906	7.736667	...	0.224056	9583.515861	18071.811908
std	1.034792	0.837223	9.380431	67.925198	71.641408	77.548001	3.421697	3.764611	3.394150	1.054052	...	0.245819	4598.626814	7250.903684
min	0.000000	1.000000	11.000000	290.000000	310.000000	350.000000	2.000000	2.000000	2.000000	5.000000	...	0.000000	-1643.000000	-1220.000000
25%	1.000000	1.000000	12.000000	475.000000	483.000000	470.000000	21.000000	20.000000	21.000000	7.000000	...	0.000000	6320.000000	13132.000000
50%	2.000000	2.000000	21.000000	515.000000	520.000000	510.000000	23.000000	22.000000	22.000000	7.000000	...	0.150350	8792.000000	18259.000000
75%	3.000000	3.000000	22.000000	555.000000	565.000000	559.000000	25.000000	25.000000	24.000000	9.000000	...	0.372750	12480.500000	22485.000000
max	4.000000	3.000000	43.000000	760.000000	785.000000	755.000000	34.000000	34.000000	35.000000	12.000000	...	1.000000	27199.000000	87570.000000

8 rows × 32 columns

In [3]:

```
df.columns
```

Out[3]:

```
Index(['INSTNM', 'CITY', 'STABBR', 'PREDDEG', 'CONTROL', 'LOCALE', 'SATVRMID',
       'SATMTMID', 'SATWRMID', 'ACTCMMID', 'ACTENMID', 'ACTMTMID', 'ACTWRMID',
       'SAT_AVG', 'DISTANCEONLY', 'UGDS', 'UGDS_WHITE', 'UGDS_BLACK',
       'UGDS_HISP', 'UGDS_ASIAN', 'UGDS_AIAN', 'UGDS_NHPI', 'UGDS_2MOR',
       'UGDS_NRA', 'UGDS_UNKN', 'PPTUG_EF', 'NPT4_PUB', 'NPT4_PRIV', 'PCTPELL',
       'RET_FT4', 'RET_FTL4', 'RET_PT4', 'RET_PTL4', 'PCTFLOAN', 'UG25abv',
       'GRAD_DEBT_MDN_SUPP', 'GRAD_DEBT_MDN10YR_SUPP', 'RPY_3YR_RT_SUPP',
       'C150_4_POOLED_SUPP', 'C200_L4_POOLED_SUPP', 'md_earn_wne_p10',
       'gt_25k_p6'],
      dtype='object')
```

2. Investigating and exploring the dataset

(a) Describe the dataset you have selected. Explain how the data was collected, and explain the meaning of the columns. Do you have any concerns about the data collection process, or about the completeness and accuracy of the data itself? Note: This is also a good time to go through some basic data cleaning: if there are columns that are obviously extraneous to data analysis (e.g., IDs or metadata that have no bearing on your analysis), you can remove those now to make your life easier.

The dataset my team chose was from the U.S Department of Education that has metrics for every college in the United States. The colleges range from part-time, community, online and full-time universities amounting to 7,804 institutions. The covariates in the dataset include university characteristics, predominant/highest degree awarded, average test scores (ACT/SAT), diversity percentages, student completion rates/retention rates, debt and repayment, and median earnings. Since the data is collected by a government agency and updated every year, it is most likely to be very trustworthy and accurate. What may have bias is columns such as median earnings which rely on students to report back to the university. This column cannot be verified easily by the university and the department of education and may need further investigation. One concern is that some columns lack completeness and may have to be thrown out as a feature.

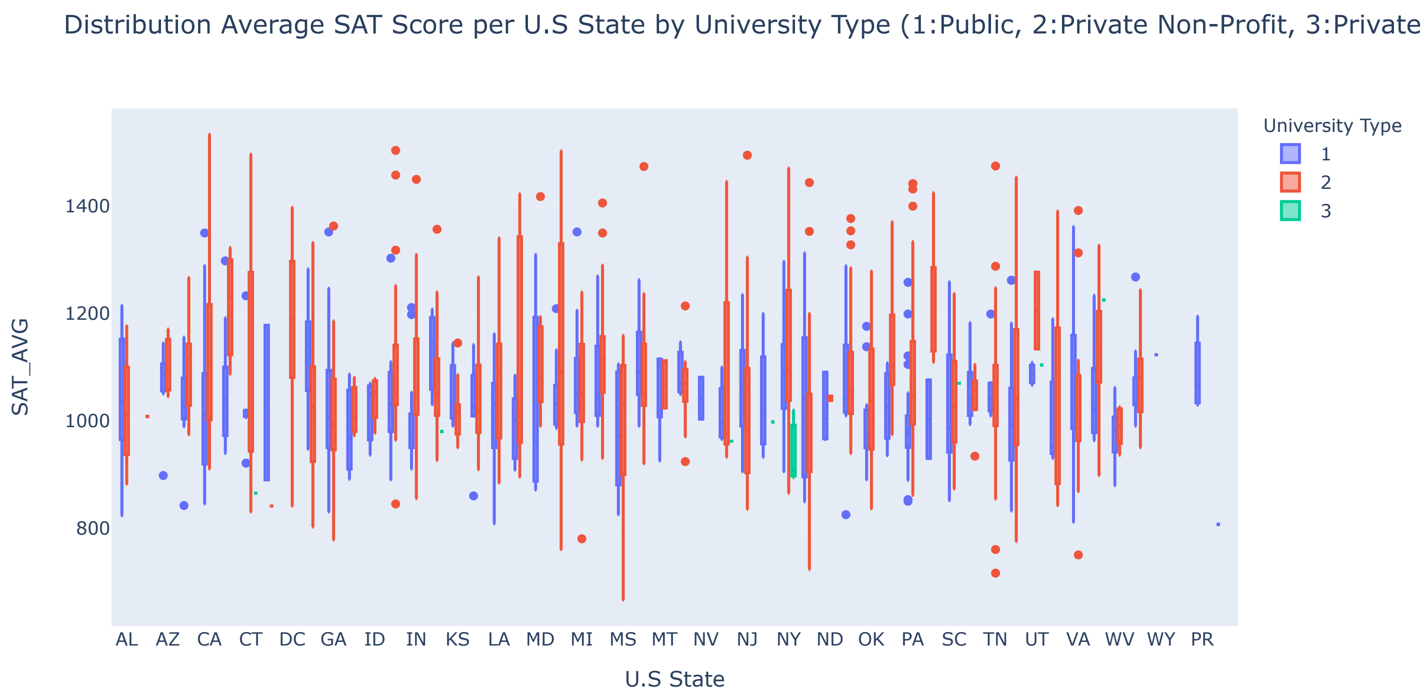
(b) Are any values in your dataset NULL or NA? Think of what you will do with rows with such entries: do you plan to delete them, or still work with the remaining columns for such rows? (You don't need to report anything to us for this part.)

In [4]:
```
# Only keeping relevant columns
data = df.drop(columns = ['LOCALE', 'DISTANCEONLY', 'UGDS_UNKN', 'PPTUG_EF', 'RET_FTL4', 'RET_PT4', 'RET_PTL4', 'UG25abv', 'GRAD_DEBT_MDN_SUPP', 'RPY_

# Only looking into colleges that have undergrad predominantly
undergrads = data[data.PREDDEG == 3]

# Seeing which columns have null or nan values
undergrads.isna().sum()
```

Out[4]:
```
INSTNM                      0
CITY                        0
STABBR                      0
PREDDEG                     0
CONTROL                     0
SAT_AVG                   782
UGDS                        2
UGDS_WHITE                  2
UGDS_BLACK                  2
UGDS_HISP                   2
UGDS_ASIAN                  2
UGDS_AIAN                   2
UGDS_NHPI                   2
UGDS_2MOR                   2
UGDS_NRA                    2
NPT4_PUB                 1574
NPT4_PRIV                 704
PCTPELL                     3
RET_FT4                   141
PCTFLOAN                    3
GRAD_DEBT_MDN10YR_SUPP     74
C150_4_POOLED_SUPP        183
md_earn_wne_p10           133
dtype: int64
```

In [5]:

```
# Plotting the state and control (public vs private) average sat score
import plotly.express as px
fig = px.box(undergrads, x="STABBR", y="SAT_AVG", color = 'CONTROL', title='Distribution Average SAT Score per U.S State by University Type (1:Public,
fig.show()
```



Preprocessing the SAT_AVG Column

In [6]:

```python
# Filling in the SAT_AVG nulls with the state and control median
def sat_avg(state_name, control_type):
  temp = undergrads[(undergrads.STABBR == str(state_name)) & (undergrads.CONTROL == control_type)]
  median = temp.SAT_AVG.median()
  return median

# making state and control lists
states = list(undergrads.STABBR.unique())
controls = list(undergrads.CONTROL.unique())

# Making a mapper to fill in the state/control median
mapper = {}
for state in states:
  for control in controls:
    mapper[state + str(control)] = sat_avg(state, control)

# Making an intermediate column
undergrads['STABBR_CONTROL'] = undergrads.STABBR + undergrads.CONTROL.astype(str)

# Mapping the state/control median
undergrads['SAT_AVG_COMP'] = undergrads['STABBR_CONTROL'].map(mapper)

# Filling in the state/control median
undergrads['SAT_AVG'] = undergrads['SAT_AVG'].fillna(undergrads['SAT_AVG_COMP'])

# Filling the rest of the nulls with the overall median
undergrads['SAT_AVG'] = undergrads['SAT_AVG'].fillna(undergrads['SAT_AVG'].median())
```

In [7]:
```
undergrads.isna().sum()
```

Out[7]:
```
INSTNM                      0
CITY                        0
STABBR                      0
PREDDEG                     0
CONTROL                     0
SAT_AVG                     0
UGDS                        2
UGDS_WHITE                  2
UGDS_BLACK                  2
UGDS_HISP                   2
UGDS_ASIAN                  2
UGDS_AIAN                   2
UGDS_NHPI                   2
UGDS_2MOR                   2
UGDS_NRA                    2
NPT4_PUB                 1574
NPT4_PRIV                 704
PCTPELL                     3
RET_FT4                   141
PCTFLOAN                    3
GRAD_DEBT_MDN10YR_SUPP     74
C150_4_POOLED_SUPP        183
md_earn_wne_p10           133
STABBR_CONTROL              0
SAT_AVG_COMP              300
dtype: int64
```

In [8]:

```python
# It's okay to have nans in the NPT4_PUB column if the institution is private
undergrads[(undergrads.NPT4_PUB.isna()) & (undergrads.CONTROL == 2)]

# It's okay if the NPT4_PRIV column is null if the instition is public
undergrads[(undergrads.NPT4_PRIV.isna()) & (undergrads.CONTROL == 1)]


# If the insitution is public then this should not be null
undergrads[(undergrads.NPT4_PUB.isna()) & (undergrads.CONTROL == 1)].NPT4_PUB = undergrads.NPT4_PUB.median()

# If the insitution is public then this should not be null
undergrads[(undergrads.NPT4_PRIV.isna()) & (undergrads.CONTROL == 2)].NPT4_PRIV = undergrads.NPT4_PRIV.median()

undergrads
```

Out[8]:

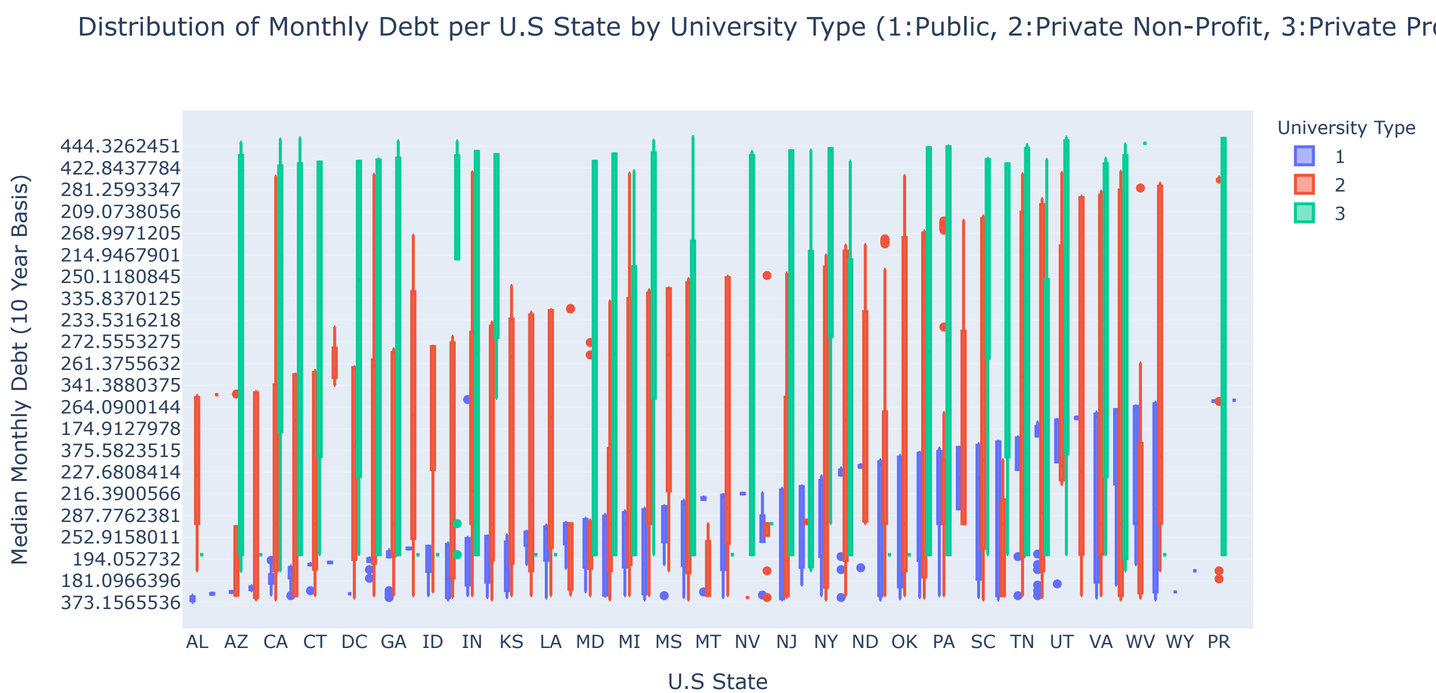
	INSTNM	CITY	STABBR	PREDDEG	CONTROL	SAT_AVG	UGDS	UGDS_WHITE	UGDS_BLACK	UGDS_HISP	...	NPT4_PUB	NPT4_PRIV	PCTPELL	RET_FT4	PCTFLO
0	Alabama A & M University	Normal	AL	3	1	823.0	4051.0	0.0279	0.9501	0.0089	...	13415.0	NaN	0.7115	0.6314	0.8
1	University of Alabama at Birmingham	Birmingham	AL	3	1	1146.0	11200.0	0.5987	0.2590	0.0258	...	14805.0	NaN	0.3505	0.8016	0.5
2	Amridge University	Montgomery	AL	3	2	1011.0	322.0	0.2919	0.4224	0.0093	...	NaN	7455.0	0.6839	0.3750	0.7
3	University of Alabama in Huntsville	Huntsville	AL	3	1	1180.0	5525.0	0.7012	0.1310	0.0338	...	17520.0	NaN	0.3281	0.8098	0.4
4	Alabama State University	Montgomery	AL	3	1	830.0	5354.0	0.0161	0.9285	0.0114	...	11936.0	NaN	0.8265	0.6219	0.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7781	DeVry University-Virginia	Arlington	VA	3	3	1050.0	783.0	0.2363	0.4151	0.1111	...	NaN	19151.0	0.4802	0.6667	0.5
7782	DeVry University-Washington	Federal Way	WA	3	3	1225.0	466.0	0.5494	0.0966	0.0579	...	NaN	17667.0	0.5855	0.6667	0.7
7783	DeVry University-Wisconsin	Milwaukee	WI	3	3	1050.0	148.0	0.5135	0.3041	0.0811	...	NaN	20867.0	0.6125	0.5000	0.8
7784	University of North Georgia	Dahlonega	GA	3	1	1009.0	14502.0	0.7901	0.0449	0.0859	...	14534.0	NaN	0.3793	0.7964	0.3
7802	Arizona State University-Skysong	Scottsdale	AZ	3	1	898.0	8227.0	0.6216	0.0889	0.1811	...	12823.0	NaN	0.4287	0.7158	0.6

2133 rows × 25 columns

In [9]:
```
# Dropping rows without enough data
undergrads = undergrads.dropna(subset=['UGDS_WHITE', 'PCTPELL', 'PCTFLOAN', 'C150_4_POOLED_SUPP', 'md_earn_wne_p10'])
```

In [10]:
```
# Filling in the RET_FT4 with the median
undergrads.RET_FT4 = undergrads.RET_FT4.fillna(undergrads.RET_FT4.median())
```

In [11]:
```
fig = px.box(undergrads, x="STABBR", y=undergrads["GRAD_DEBT_MDN10YR_SUPP"], color = 'CONTROL', title='Distribution of Monthly Debt per U.S State by Ui
fig.show()
```



In [12]:

```python
# Filling in the GRAD_DEBT_MDN10YR_SUPP with the median for that control
pub_med = pd.to_numeric(undergrads[undergrads.CONTROL == 1]['GRAD_DEBT_MDN10YR_SUPP'],errors='coerce').median()
priv_non_med = pd.to_numeric(undergrads[undergrads.CONTROL == 2]['GRAD_DEBT_MDN10YR_SUPP'],errors='coerce').median()
priv_for_med = pd.to_numeric(undergrads[undergrads.CONTROL == 3]['GRAD_DEBT_MDN10YR_SUPP'],errors='coerce').median()

# Making mapper
debt_mapper = {1: pub_med, 2: priv_non_med, 3: priv_for_med}

# Filling in string values with nulls
undergrads['GRAD_DEBT_MDN10YR_SUPP'].replace({'PrivacySuppressed': None},inplace =True)
undergrads['C150_4_POOLED_SUPP'].replace({'PrivacySuppressed': None},inplace =True)
undergrads['md_earn_wne_p10'].replace({'PrivacySuppressed': None},inplace =True)
undergrads['C150_4_POOLED_SUPP'].replace({'blank': None},inplace =True)
undergrads['md_earn_wne_p10'].replace({'blank': None},inplace =True)


undergrads['GRAD_DEBT_MDN10YR_SUPP'] = undergrads['GRAD_DEBT_MDN10YR_SUPP'].fillna(undergrads['CONTROL'].map(debt_mapper)).astype(float)
undergrads=undergrads.dropna(subset = ['C150_4_POOLED_SUPP','md_earn_wne_p10'])
undergrads[['C150_4_POOLED_SUPP', 'md_earn_wne_p10']] = undergrads[['C150_4_POOLED_SUPP', 'md_earn_wne_p10']].astype(float)
undergrads['Tuition'] = undergrads.NPT4_PUB.fillna(0) + undergrads.NPT4_PRIV.fillna(0)
```

In [13]:

```python
# Now the data is fully cleaned
cleaned = undergrads.reset_index(drop = True).rename(columns = {'md_earn_wne_p10':'Median Salary 1+', 'C150_4_POOLED_SUPP':'Graduation Rate Over 50%'}
cleaned['Graduation Rate Over 50%'] = cleaned['Graduation Rate Over 50%'].apply(lambda x: 1 if x > 0.5 else 0)
```

(c) Randomly choose a test set (representing 20% of your rows), and keep it for later. You will not touch this test set again until the end of the course! Fix this set from the beginning, and use the remaining 80% for exploration, model selection, and validation. (You don't need to report anything to us for this part.)

In [14]:

```python
# Splitting into train test split
import numpy as np
from sklearn.model_selection import train_test_split

# Establishing design matrix and potential responses
X = cleaned.drop(columns = ['Graduation Rate Over 50%', 'Median Salary 1+', 'STABBR_CONTROL', 'SAT_AVG_COMP','UGDS_NRA', 'NPT4_PUB', 'NPT4_PRIV', 'SAT_

# Making dummy variables for the states
encoded_states = pd.get_dummies(X.STABBR, prefix='STATE')
X = pd.concat([X, encoded_states], axis = 1).drop(columns = ['STABBR'])

y = cleaned[['Graduation Rate Over 50%', 'Median Salary 1+']]

# Splitting the data into a test and train set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

(d) Compute the mean and variance for each of the columns (you don't need to report this to us). Are there any columns that appear to be random noise?

In [15]:

```
means = X_train.describe().loc['mean',:].to_frame()
variances = X_train.var().to_frame()
stat=pd.concat([means,variances],axis=1).rename(columns = {0:'variance'})
stat['variance'] = stat['variance'].map(int)
stat.sort_values(by = 'variance', ascending=False)
```

Out[15]:

	mean	variance
Tuition	19026.476466	49829596
UGDS	4764.880521	46270056
SAT_AVG	1060.140840	13378
GRAD_DEBT_MDN10YR_SUPP	274.858822	3862
PREDDEG	3.000000	0
...	...	...
STATE_IL	0.040550	0
STATE_IN	0.028240	0
STATE_KS	0.015206	0
STATE_KY	0.015206	0
STATE_WY	0.000724	0

70 rows × 2 columns

Tuition, the number of undergrads enrolled, and sat average have the highest variances in the design matrix. This could indicate that the dataset has many outliers and might need additional preprocessing to remove extraneous values.

(e) Suggest at least one possibility for a continuous outcome variable (a.k.a. response variable) that may be of interest to measure the effect of a potential intervention. Explain your choice and the variable's potential meaning. Compute the mean and variance of this variable.

The continuous variable we chose to measure is the median salary earning of students at the university one year after graduation who are not enrolled with the university after ten years (Doctorate students and other edge cases). The salary earned after university indicates university prestigiousness and success of students. The mean of this response variable is 41,477 with a variance of 112,236,100.

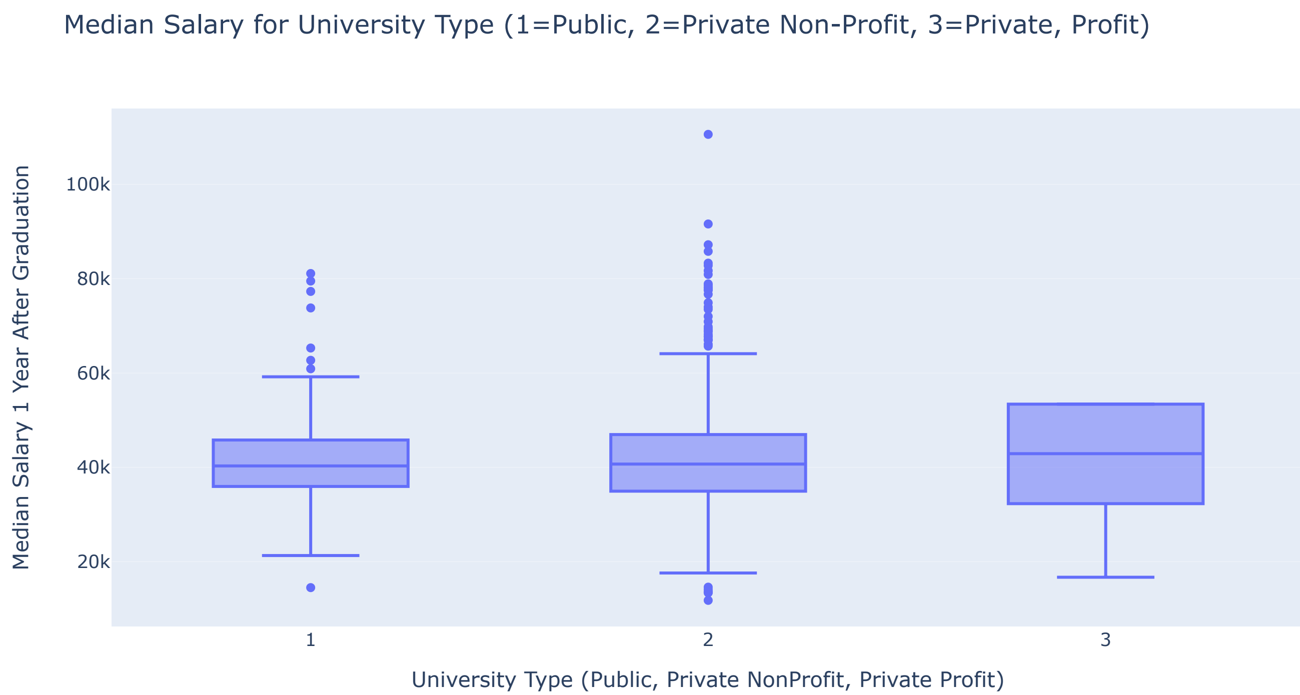
In [16]:

```
means = y_train.describe().loc['mean',:].to_frame()
variances = y_train.var().to_frame()
stat=pd.concat([means,variances],axis=1).rename(columns = {0:'variance'})
stat['variance'] = stat['variance'].map(int)
stat
```

Out[16]:

	mean	variance
Graduation Rate Over 50%	0.498914	0
Median Salary 1+	41477.914555	112236055

In [17]:
```
fig = px.box(x=X_train["CONTROL"], y=y_train["Median Salary 1+"],title='Median Salary for University Type (1=Public, 2=Private Non-Profit, 3=Private, I
fig.show()
```



(f) Suggest at least one possibility for a binary outcome variable. Compute the mean of this variable (i.e., the fraction of rows for which this variable is 1.) Explain your choice.

The binary outcome variable we chose to measure is the universities with graduation rates higher than 50% for students within a 6 year time frame. We chose this variable because it summarizes retention and quality of education at a particular university. It indicates the likelihood of a student graduating with 6 years of attendance. The fraction of universities with 50% or higher graduation rate was 49.89% with a variance of 25%

In [18]:
```
print('Average Graduation Rate within 6 years > 50', 'mean: ', y_train['Graduation Rate Over 50%'].mean(), 'variance ', y_train['Graduation Rate Over !
y_train['Graduation Rate Over 50%'].value_counts()
```

```
Average Graduation Rate within 6 years > 50 mean:  0.498913830557567 variance  0.2501799788013307
```

Out[18]:
```
0    692
1    689
Name: Graduation Rate Over 50%, dtype: int64
```

(g) Find the five covariates that are the most strongly positively correlated, as well as most strongly negatively correlated, with your choice of continuous outcome variable. (You should do the same for your choice of binary outcome variable, but you don't need to report the results.) Are there variables you think should affect your outcome variable but actually have weak correlation with your outcome variable?

SAT Average, percentage of asian undergraduates, four year retention rate, tution, and percentage of undergraduates with 2 or more ethnicity were the most positively correlated with median earnings. The top five had an r-value between 0.18-0.5. One variable that I thought would be more corrleated with median earnings is whether the institution is Private or Public (CONTROL) because I thought that private universities may have a richer alumni network to help students get jobs. This variable was only had an r-value of 0.02.

Percentage of students receiving the Pell Grant scholarship, percentage of Black undergrads, percent of students who receive federal student loans, percent Hispanic, and percent Native American are all negatively correlated with median earnings. This is significant because it shows that universities with higher student debt actually have lower median earnings indicating a financial bubble. It is also disturbing that universities with higher Black, Hispanic, or Native American populations are more likely to earn less indicating systemic discrimination in the university system.

In [19]:

```
numeric = X_train.select_dtypes(include=['float64', 'int64'])
corr = numeric.apply(lambda x: x.corr(y_train['Median Salary 1+']))
print('Top 5 Positive Correlated Covariates with Median Earnings\n', corr.sort_values(ascending=False)[:5])
print('\nTop 5 Negative Correlated Covariates with Median Earnings\n', corr.sort_values(ascending=True)[:5])
first5 = pd.DataFrame(corr.sort_values(ascending=False)[:5])
last5 = pd.DataFrame(corr.sort_values(ascending=True)[:5])
ten_covariates = list(first5.index) + list(last5.index)
```

```
Top 5 Positive Correlated Covariates with Median Earnings
 SAT_AVG       0.505248
UGDS_ASIAN    0.453705
RET_FT4       0.384706
Tuition       0.316921
UGDS_2MOR     0.183229
dtype: float64

Top 5 Negative Correlated Covariates with Median Earnings
 PCTPELL      -0.554872
UGDS_BLACK   -0.217487
PCTFLOAN     -0.195294
UGDS_HISP    -0.117436
UGDS_AIAN    -0.105465
dtype: float64
```

In [20]:

```
corr
```

Out[20]:

```
PREDDEG                        NaN
CONTROL                   0.026451
SAT_AVG                   0.505248
UGDS                      0.177987
UGDS_WHITE                0.076628
UGDS_BLACK               -0.217487
UGDS_HISP                -0.117436
UGDS_ASIAN                0.453705
UGDS_AIAN                -0.105465
UGDS_NHPI                -0.023840
UGDS_2MOR                 0.183229
PCTPELL                  -0.554872
RET_FT4                   0.384706
PCTFLOAN                 -0.195294
GRAD_DEBT_MDN10YR_SUPP   -0.023217
Tuition                   0.316921
dtype: float64
```

In [22]:

```
corr2 = numeric.apply(lambda x: x.corr(y_train['Graduation Rate Over 50%']))
print('Top 5 Positive Correlated Covariates with Graduation Rate\n', corr2.sort_values(ascending=False)[:5])
print('\nTop 5 Negative Correlated Covariates with Graduation Rate\n', corr2.sort_values(ascending=True)[:5])
```

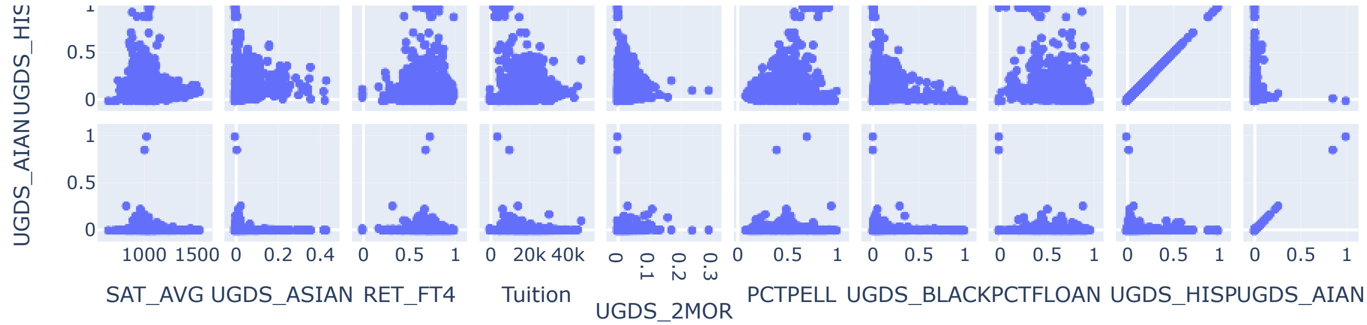
```
Top 5 Positive Correlated Covariates with Graduation Rate
 RET_FT4       0.624909
SAT_AVG       0.507087
Tuition       0.377875
UGDS_WHITE    0.368355
UGDS_ASIAN    0.233250
dtype: float64

Top 5 Negative Correlated Covariates with Graduation Rate
 PCTPELL                  -0.583411
UGDS_BLACK               -0.339184
UGDS_HISP                -0.161823
PCTFLOAN                 -0.134525
GRAD_DEBT_MDN10YR_SUPP   -0.127971
dtype: float64
```

(h) Now find mutual correlations among the ten variables you identified in the last part. Create a scatterplot for every pair of covariates you believe correlates well to the outcome variable. Are correlations associative in your data? That is, if A is correlated strongly with B, and B with C, is A also correlated strongly with C in your data?

Figure 1 shows the correlation matrix between the 10 covariates identified. It seems that University SAT average score and retention rate are highly correlated with each other as well as positively correlated with Median Earnings (Figure 2). This indicates that it is unlikely that people with high SAT scores will not drop out of college because they're invested in their education. SAT scores are also highly correlated with universities who have a percentage of Asian students (Figure 3). On the other hand, unfortunately, student loans are correlated with lower median earnings (Figure 4) which is also correlated with the universities who are made up of a higher percentage of underprivileged communities (Blacks, Hispanics, Native Americans) (Figure 5). This again shows the potential systemic bias in the U.S education system.

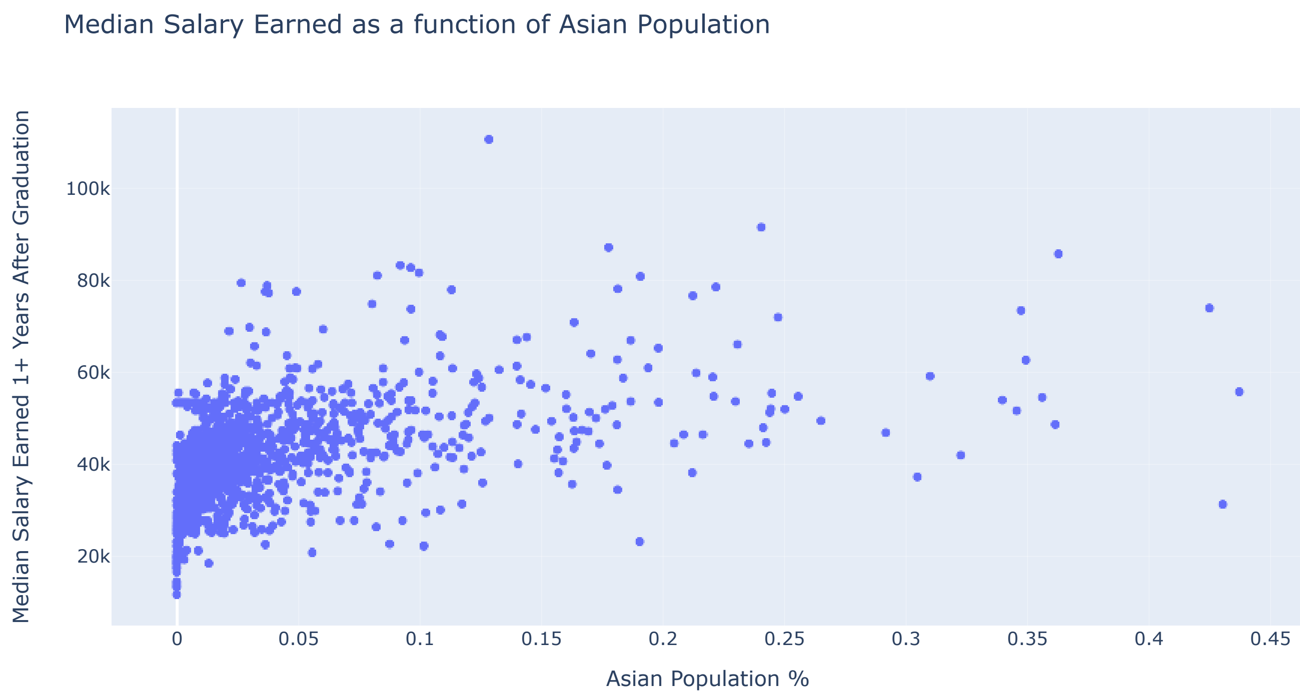
In [23]:

```
scatter_mat = X_train.loc[:,ten_covariates]
fig = px.scatter_matrix(scatter_mat)
fig.update_layout(
    title='College Score Card Data Set of Top Ten Covariates',
    dragmode='select',
    width=1000,
    height=1000,
    hovermode='closest',
)

fig.show()
```



In [24]:

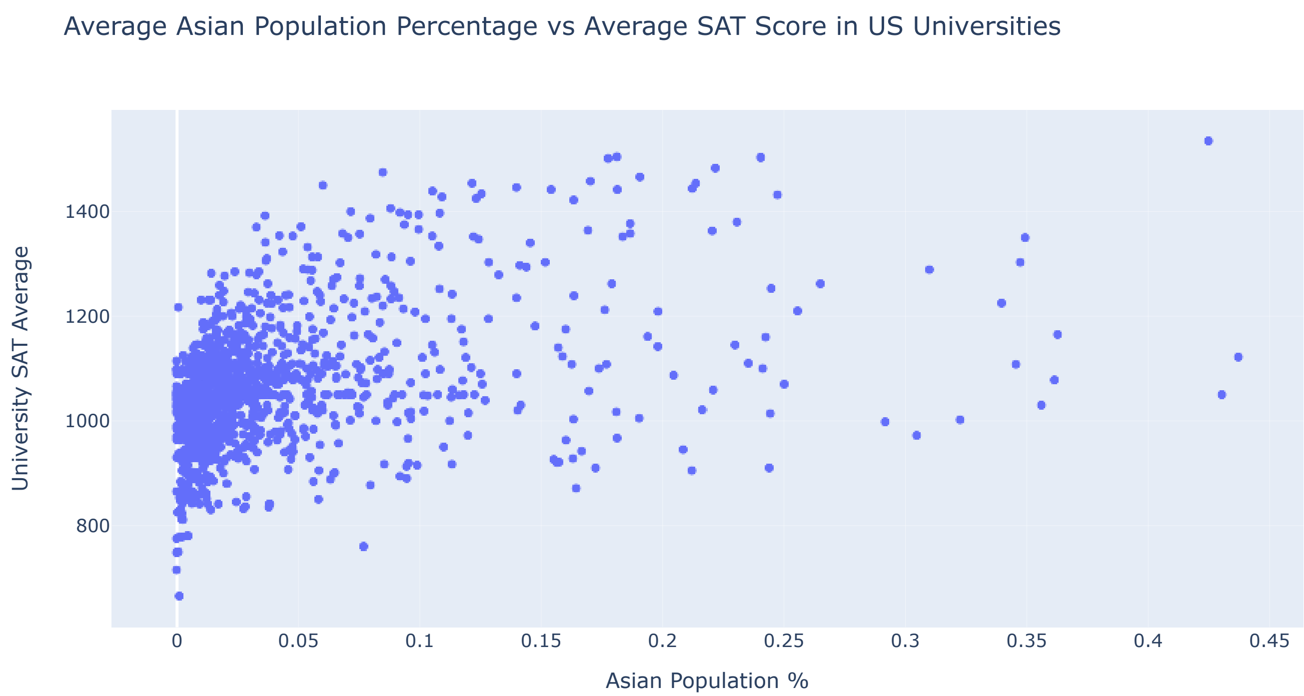
```
px.scatter(x=X_train['UGDS_ASIAN'],y=y_train['Median Salary 1+'],title='Median Salary Earned as a function of Asian Population', labels={
    'x': 'Asian Population %',
    'y': 'Median Salary Earned 1+ Years After Graduation'})
```



Asian Population % and SAT Average Score are associative and correlated with Median Earnings

In [25]:

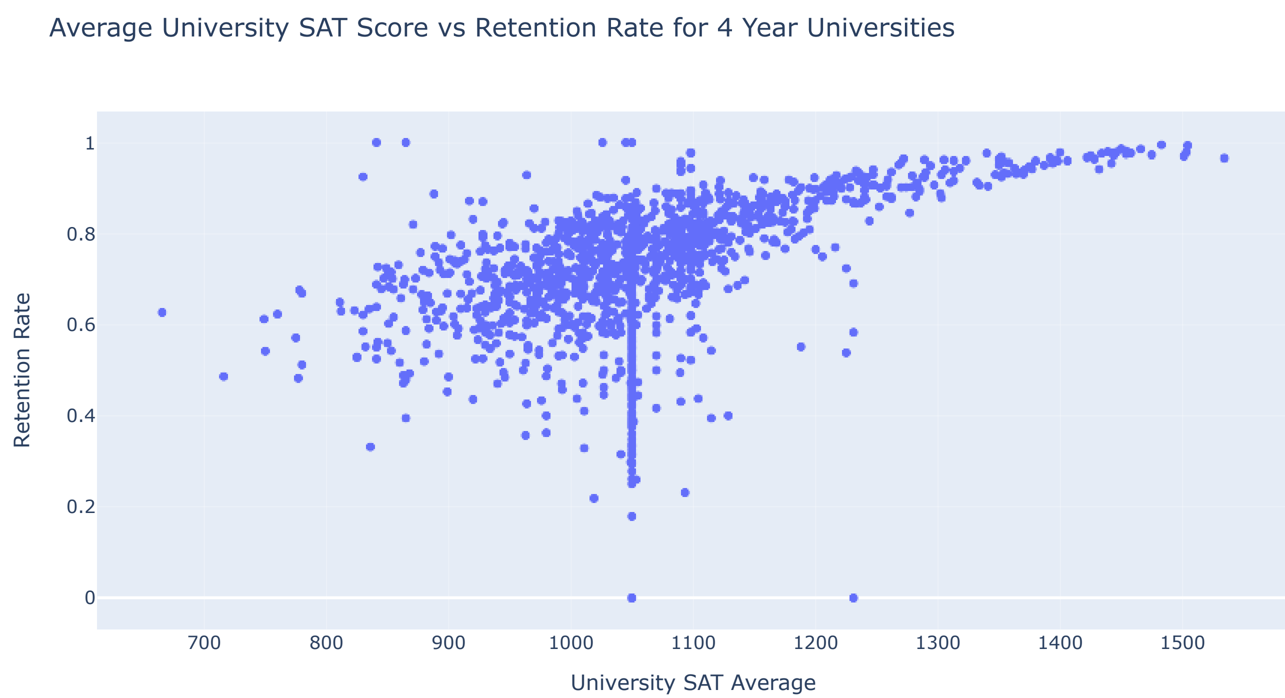
```
px.scatter(X_train, x='UGDS_ASIAN',y='SAT_AVG',title='Average Asian Population Percentage vs Average SAT Score in US Universities', labels={
    'SAT_AVG': 'University SAT Average',
    'UGDS_ASIAN': 'Asian Population %'})
```



Average SAT Score and Retention Rate for 4 Year Universities are associative and correlated with Median Earnings

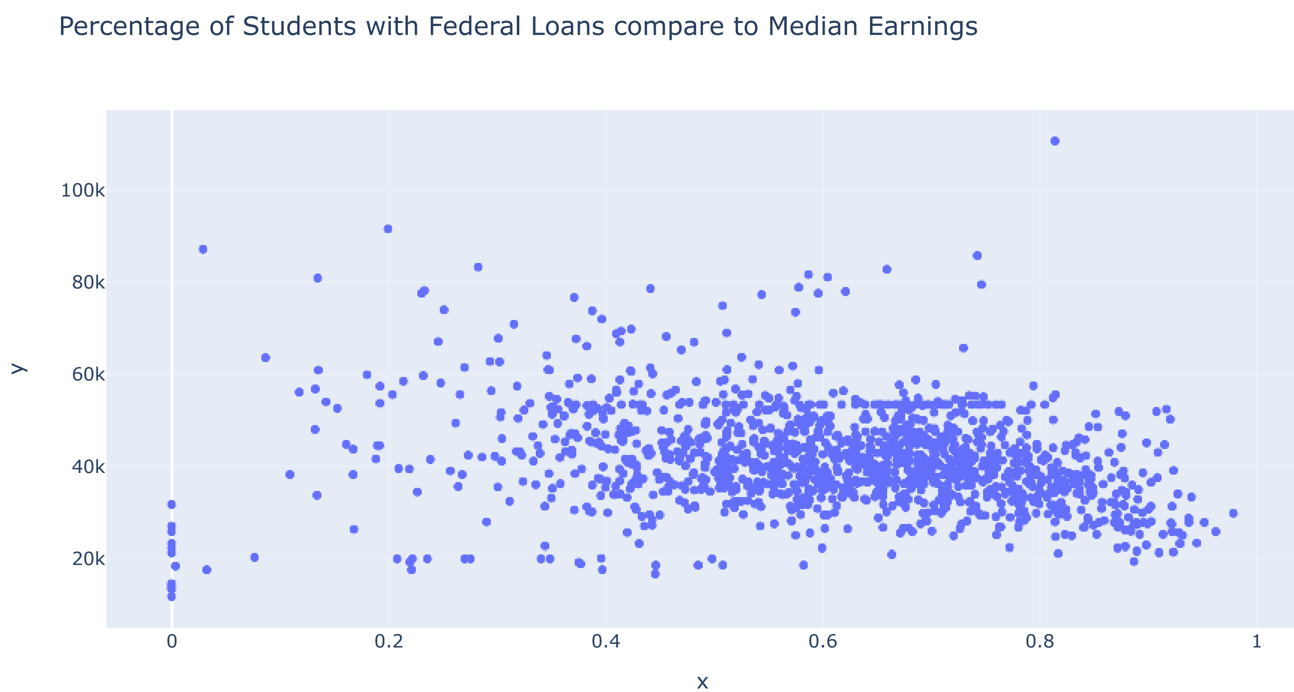
In [26]:

```
fig = px.scatter(X_train, x='SAT_AVG',y='RET_FT4',title='Average University SAT Score vs Retention Rate for 4 Year Universities', labels={
    'SAT_AVG': 'University SAT Average',
    'RET_FT4': 'Retention Rate'})
fig.show()
```



The percentage of students on federal student loans and median salary is negatively correlated

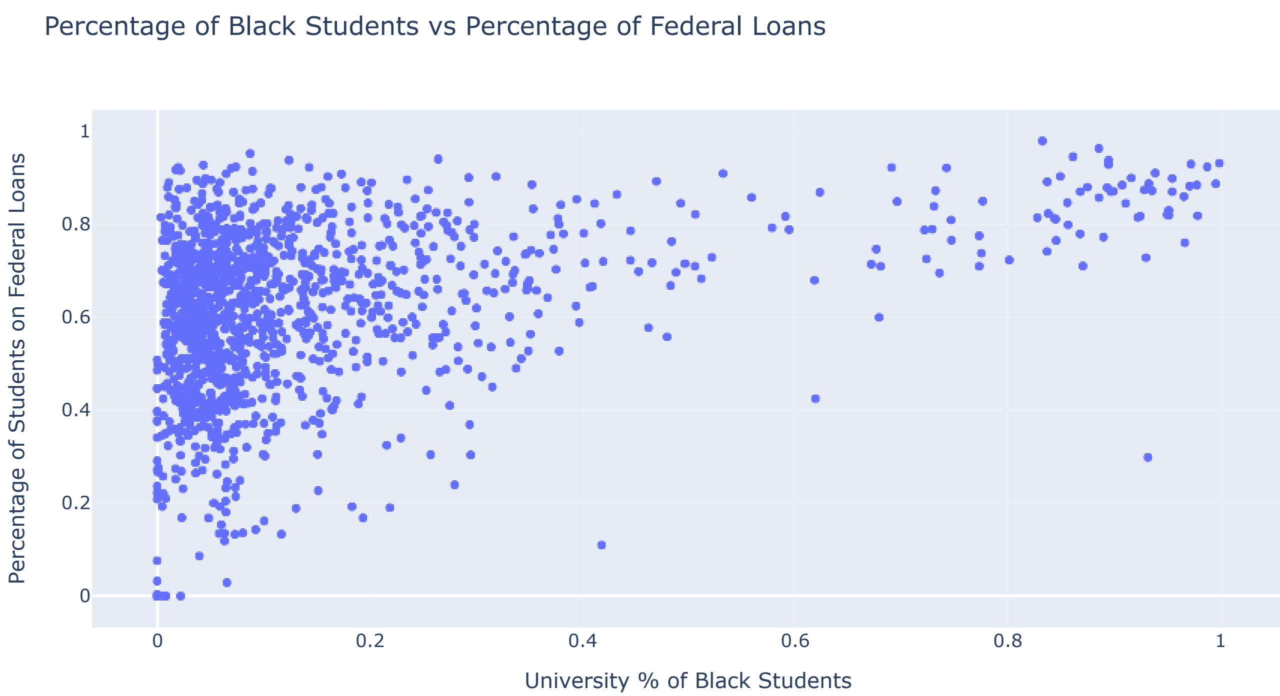
In [27]:
```
px.scatter(x=X_train['PCTFLOAN'],y=y_train['Median Salary 1+'],title='Percentage of Students with Federal Loans compare to Median Earnings', labels={
    'Median Salary 1+': 'Median Salary 1 Year After Graduation',
    'PCTFLOAN': 'Percentage of Students on Federal Loans'})
```



The percentage of Black students and percentages of students with federal loans are associative, and negatively correlated with median earnings

In [28]:

```
px.scatter(X_train, x='UGDS_BLACK',y='PCTFLOAN',title='Percentage of Black Students vs Percentage of Federal Loans', labels={
    'UGDS_BLACK': 'University % of Black Students',
    'PCTFLOAN': 'Percentage of Students on Federal Loans'})
```



(i) Are there variables you would like to add to your dataset as you embark on your analysis? For example, are there interactions or higher order terms that might be relevant? (You don’t need to report your answer to this part.)

I think that if there were additional variables to university endowment, average salary of teachers, funding, and available scholarships we could better assess how university financials affect student outcomes. This would help with highlighting discrepancies between median earnings, affordability, and chance of student success measured in post-graduation earnings.