Jobs in STEM

Christopher Chen, Daniel Boules, Noah De Mers, William Mayuga

Initializing Libraries and Reading in the Data

```
%matplotlib inline
import matplotlib
import pandas as pd
import numpy as np
import seaborn as sns
import pylab as py
import math as math
from scipy.stats import chi2_contingency
from sklearn.linear_model import LinearRegression

all_companies_df = pd.read_csv("Levels_Fyi_Salary_Data.csv")

df = pd.read_csv("Levels_Fyi_Salary_Data.csv")
df.head()
```

Initializing Libraries and Reading in the Data

	timestamp	company	level	title	totalyearlycompensation	location	yearsofexperience	yearsatcompany	tag	basesalary	...	Doctorate_Degree
0	6/7/2017 11:33:27	Oracle	L3	Product Manager	127000	Redwood City, CA	1.5	1.5	NaN	107000.0	...	0
1	6/10/2017 17:11:29	eBay	SE 2	Software Engineer	100000	San Francisco, CA	5.0	3.0	NaN	0.0	...	0
2	6/11/2017 14:53:57	Amazon	L7	Product Manager	310000	Seattle, WA	8.0	0.0	NaN	155000.0	...	0
3	6/17/2017 0:23:14	Apple	M1	Software Engineering Manager	372000	Sunnyvale, CA	7.0	5.0	NaN	157000.0	...	0
4	6/20/2017 10:58:51	Microsoft	60	Software Engineer	157000	Mountain View, CA	5.0	3.0	NaN	0.0	...	0

tag	basesalary	...	Doctorate_Degree	Highschool	Some_College	Race_Asian	Race_White	Race_Two_Or_More	Race_Black	Race_Hispanic	Race	Education
NaN	107000.0	...	0	0	0	0	0	0	0	0	NaN	NaN
NaN	0.0	...	0	0	0	0	0	0	0	0	NaN	NaN
NaN	155000.0	...	0	0	0	0	0	0	0	0	NaN	NaN
NaN	157000.0	...	0	0	0	0	0	0	0	0	NaN	NaN
NaN	0.0	...	0	0	0	0	0	0	0	0	NaN	NaN

Cleaning the Data

```
number_of_entries_by_company = df.groupby("company")["timestamp"].count()
print(number_of_entries_by_company)
```

```
company
10x Genomics     6
23andMe          7
2U               7
3M              21
3m               3
                ..
zoom             1
zoominfo         1
zoox             3
zynga            1
 Google          1
Name: timestamp, Length: 1631, dtype: int64
```

Because we later want to use K-nearest neighbors for a regression to predict income, we want companies with more than just a few entries. We chose companies that have at least 500 entries.

```
statistics_by_company_df = np.round(pd.pivot_table(df, index = ["company"], values =
                              ["totalyearlycompensation", "yearsofexperience", "yearsatcompany"]
                              , aggfunc='mean'))

statistics_by_company_df['counts'] = number_of_entries_by_company
statistics_by_company_df = statistics_by_company_df[~(statistics_by_company_df['counts'] <= 500)]
    # Limits the number of companies by removing companies with 500 or less rows in the dataset.

print(statistics_by_company_df)
```

```
                totalyearlycompensation  yearsatcompany  yearsofexperience  \
company
Amazon                         227352.0             2.0                7.0
Apple                          277930.0             3.0                8.0
Bloomberg                      211050.0             2.0                5.0
Capital One                    148808.0             2.0                6.0
Cisco                          196243.0             5.0                9.0
Facebook                       344527.0             1.0                7.0
Google                         283290.0             3.0                7.0
IBM                            137724.0             4.0                7.0
Intel                          180525.0             7.0                9.0
JPMorgan Chase                 136250.0             3.0                7.0
LinkedIn                       307680.0             2.0                7.0
Microsoft                      208501.0             4.0                8.0
Oracle                         212571.0             3.0                9.0
Qualcomm                       191547.0             5.0                8.0
Salesforce                     260550.0             2.0                9.0
Uber                           304648.0             2.0                7.0
VMware                         214721.0             3.0                9.0

                counts
company
Amazon            8126
Apple             2028
Bloomberg          537
Capital One        778
Cisco              907
Facebook          2990
Google            4330
IBM                907
Intel              949
JPMorgan Chase     541
LinkedIn           701
Microsoft         5216
Oracle            1128
Qualcomm           565
Salesforce        1056
Uber               880
VMware             657
```

Getting a feel for the Data

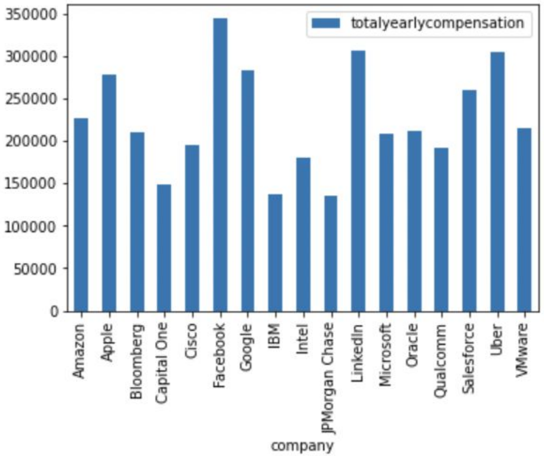
```
overall_statistics = [
                        [df['totalyearlycompensation'].min(),
                        df['totalyearlycompensation'].max(),
                        df['totalyearlycompensation'].std(),
                        df['totalyearlycompensation'].mean()
                        ],
                        [df['yearsofexperience'].min(),
                         df['yearsofexperience'].max(),
                         df['yearsofexperience'].std(),
                         df['yearsofexperience'].mean()
                        ],
                        [df['yearsatcompany'].min(),
                         df['yearsatcompany'].max(),
                         df['yearsatcompany'].std(),
                         df['yearsatcompany'].mean()
                        ]
                      ]

type_names = ['Yearly Compensation', 'Years of Experience', 'Years at Company']
statistic_names = ['Min', 'Max', 'Standard Deviation', 'Mean']

index_outer = 0;
for i in type_names:
    print(i + ':')
    index_inner = 0;
    for j in statistic_names:
        print('    - ' + j + ': ' + str(overall_statistics[index_outer][index_inner]))
        index_inner += 1;
    index_outer += 1;
    print('')
```

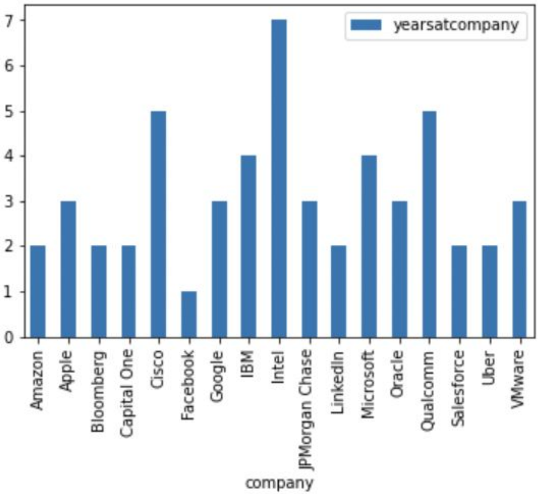
```
Yearly Compensation:
  - Min: 10000
  - Max: 4980000
  - Standard Deviation: 138033.7463773671
  - Mean: 216300.37364707384

Years of Experience:
  - Min: 0.0
  - Max: 69.0
  - Standard Deviation: 5.84037534823308
  - Mean: 7.2041350850866825

Years at Company:
  - Min: 0.0
  - Max: 69.0
  - Standard Deviation: 3.263655591673307
  - Mean: 2.7020929408384147
```

Averages:

Total Yearly Compensation by Company

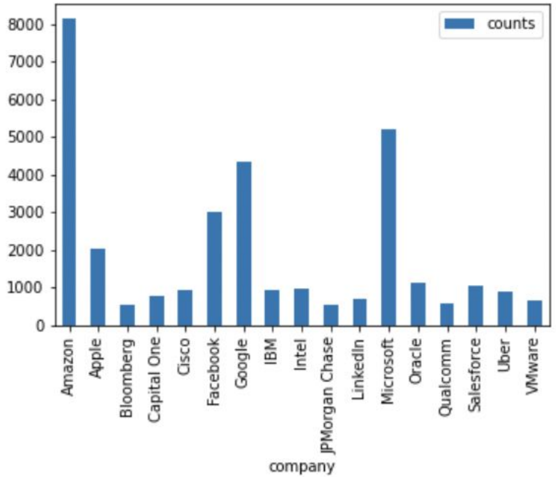
Years at Company



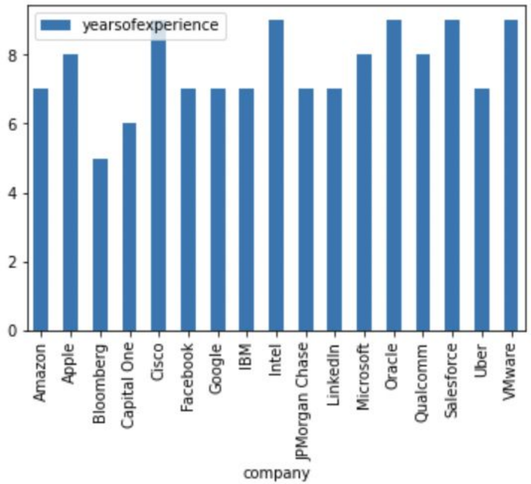


Averages:

Number of Data Entries We Have Per Company



Years of Experience at Company



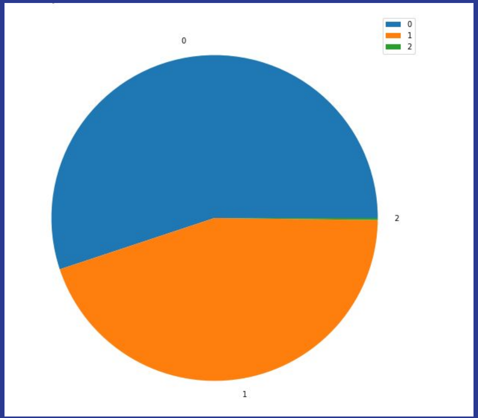
Distribution of Education

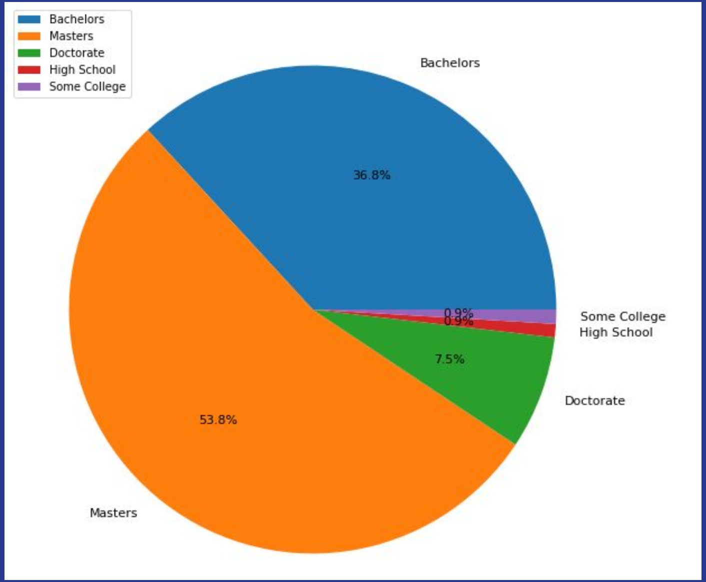
```
df["num_degrees"] = df["Bachelors_Degree"] + df["Masters_Degree"] + df["Doctorate_Degree"]

education_pivot_some_college = pd.crosstab(df['num_degrees'], df['Some_College'])
education_pivot_highschool = pd.crosstab(df['num_degrees'], df['Highschool'])

highschool_sum = df['Highschool'].sum()
some_college_sum = df['Some_College'].sum()
doctorate_sum = df['Doctorate_Degree'].sum()
masters_sum = df['Masters_Degree'].sum()
bachelors_sum = df['Bachelors_Degree'].sum()

print('Doctorate: ' + str(doctorate_sum) + ' | Masters: ' + str(masters_sum) + ' | Bachelors: '
      + str(bachelors_sum), '\n')

education_pivot_some_college['Number of Degrees'] = education_pivot_some_college[0]
                                                    + education_pivot_some_college[1]

print(education_pivot_some_college, '\n\n', education_pivot_highschool, '\n')

# Seeing if multiple have degrees have been Marked for a significant portion of sample
#(Not really, refer to second pie chart for more usable data)

education_pivot_some_college.plot.pie(y='Number of Degrees', ylabel='', figsize=(20, 10))

# Comparing the Highest Education People Received
data = [['Bachelors', bachelors_sum], ['Masters', masters_sum], ['Doctorate', doctorate_sum],
        ['High School', highschool_sum], ['Some College', some_college_sum]]
education_df = pd.DataFrame(data, columns = ['Type of Degree', 'Counts'])
education_df.set_index('Type of Degree', inplace=True)

education_df.plot.pie(y='Counts', figsize=(20, 10), ylabel='', autopct='%1.1f%%', fontsize=11)
```

Distribution of Education

```
Doctorate: 1803 | Masters: 15391 | Bachelors: 12605

Some_College       0    1  Number of Degrees
num_degrees
0              32592  355              32947
1              29591    0              29591
2                104    0                104

 Highschool        0    1
num_degrees
0              32627  320
1              29591    0
2                104    0
```

Distribution of Education





A surprising amount of the workforce at these larger companies has a master's degree over a bachelor's degree. We initially attempted to compare the number of degrees people have but it seems most people who filled out the survey only the selected the highest amount of education they received. We decided to shift some of our analysis to be based off of their highest level of education.

Linear Regression to Analyze Significance of Factors on Pay

```
print(len(df.index))

train = df.loc[:(len(df.index) * 0.8)].copy()
test = df.loc[(len(df.index) * 0.8 + 1):].copy()

model = LinearRegression()
model.fit(
    X=train[['yearsofexperience', 'yearsatcompany', 'num_degrees']],
    y=train['totalyearlycompensation']
)

model.predict(
    X=test[['yearsofexperience', 'yearsatcompany', 'num_degrees']]
)

print(model.coef_, model.intercept_, '\n\n', 3 * model.coef_[0]
      + 2 * model.coef_[1] + 1 * model.coef_[2] + model.intercept_)
```

```
32296
[11176.746204   -1612.76711213   4861.33217403] 167396.79079127527

 202562.82735303743
```

Years of experience have the greatest positive impact on pay while the number of years spent at a company has the greatest negative impact. Having a degree does not seem nearly as important as having years of experience in the long run.

We can see an interesting correlation in yearly income with the number of years of experience, the number of years at a company, and whether or not people have a degree. The most significant factor in yearly income of STEM jobs is the number of years of experience you have. Surprisingly, the longer you are at a company, the less money you tend to make. And lastly, having a degree will increase your pay but not by a significant amount.

Predict Yearly Income Based on Multiple Factors

Based on the factors of your choice, what might your pay look like?

```
standard_inputs_list = ["yearsofexperience", "yearsatcompany"]
one_hot_inputs_list = ["location", "company", "title"]
all_inputs_list = ["yearsofexperience", "yearsatcompany", "location", "company", "title'
questions_for_user = ["How many years of experience?", "How many years at the company?"

user_entries = []

index = 0
for i in questions_for_user:
    elements = df[all_inputs_list[index]].unique()
    if(len(elements) > 100):
        print(elements[:20])
    else:
        print(elements)
    user_entries.append(input(i))
    index += 1

standard_features = []
one_hot_features = []
all_features = []

if (user_entries[0] != ''):
    standard_features.append(standard_inputs_list[0])
    all_features.append(standard_inputs_list[0])
if (user_entries[1] != ''):
    standard_features.append(standard_inputs_list[1])
    all_features.append(standard_inputs_list[1])
if (user_entries[2] != ''):
    one_hot_features.append(one_hot_inputs_list[0])
    all_features.append(one_hot_inputs_list[0])
if (user_entries[3] != ''):
    one_hot_features.append(one_hot_inputs_list[1])
    all_features.append(one_hot_inputs_list[1])
if (user_entries[4] != ''):
    one_hot_features.append(one_hot_inputs_list[2])
    all_features.append(one_hot_inputs_list[2])
```

```
[ 1.5   8.    7.    5.    8.5  15.    4.    3.   12.   16.   10.    1.
  9.    2.   17.   14.    6.   20.   28.   13.   19.    0.5  11.    0.
 11.5   3.5   6.5  18.    2.5  21.    3.8   0.8   6.75 37.   24.   25.
 30.    5.5  32.   26.   23.   34.   27.   35.   22.   29.   33.   40.
 39.   31.   42.   41.    0.25 36.   38.   45.    4.5   0.6   1.4  10.5 ]

How many years of experience? #USER INPUT
```

Predict Yearly Income Based on Multiple Factors

Based on the factors of your choice, what might your pay look like?

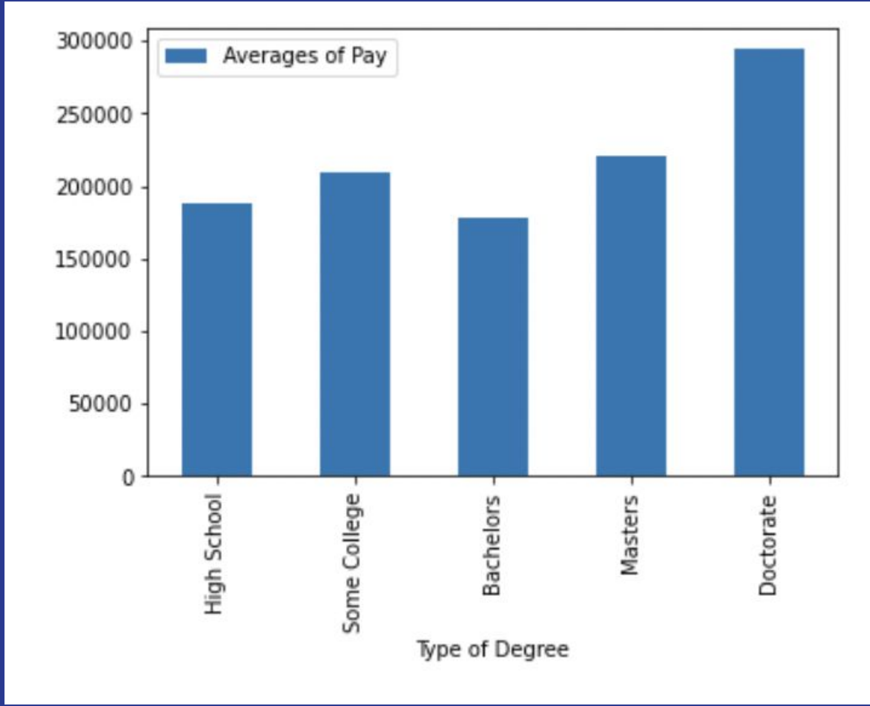
```
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder

if (len(standard_features) == 0 and len(one_hot_features) == 0):
    print('Cannot Perform Regression with No Features')
else:
    if (len(standard_features) != 0 and len(one_hot_features) != 0):
        ct = make_column_transformer(
            (StandardScaler(), standard_features),
            (OneHotEncoder(), one_hot_features),
            remainder="drop"  # all other columns in X will be dropped.
        )
    elif (len(standard_features)):
        ct = make_column_transformer(
            (StandardScaler(), standard_features),
            remainder="drop"  # all other columns in X will be dropped.
        )
    elif (len(one_hot_features) != 0):
            ct = make_column_transformer(
            (OneHotEncoder(), one_hot_features),
            remainder="drop"  # all other columns in X will be dropped.
        )
    else:
        print('Cannot Perform Operation')


    from sklearn.pipeline import make_pipeline
    from sklearn.neighbors import KNeighborsRegressor

    pipeline = make_pipeline(
        ct,
        KNeighborsRegressor(n_neighbors=10)
    )

    pipeline.fit(X=df[all_features],
                 y=df["totalyearlycompensation"])

    x_test = pd.Series(dtype=float)
    if (user_entries[0] != ''):
        x_test["yearsofexperience"] = user_entries[0]
    if (user_entries[1] != ''):
        x_test["yearsatcompany"] = user_entries[1]
    if (user_entries[2] != ''):
        x_test["location"] = user_entries[2]
    if (user_entries[3] != ''):
        x_test["company"] = user_entries[3]
    if (user_entries[4] != ''):
        x_test["title"] = user_entries[4]

    print(pipeline.predict(X=pd.DataFrame([x_test])))
```

This regression was built with the idea that you can choose the factors that describe what you want from your job and predict the total yearly compensation of it. You have the option to inout how many years of experience you have, how long you have spent at a company, your desired location, the company you want to work for, and your title. At the very least, the number of years of experience should be input to get an idea of pay for your years of experience. Other factors can contribute to a higher or lower pay but every regression is different based on the factors you choose.

How does pay vary with the amount of education received?

```
doctorate_df = df[~(df['Doctorate_Degree'] == 0)]
masters_df = df[~(df['Masters_Degree'] == 0)]
bachelors_df = df[~(df['Bachelors_Degree'] == 0)]
highschool_df = df[~(df['Highschool'] == 0)]
some_college_df = df[~(df['Some_College'] == 0)]

avg_of_doctorate = doctorate_df['totalyearlycompensation'].mean()
avg_of_masters = masters_df['totalyearlycompensation'].mean()
avg_of_bachelors = bachelors_df['totalyearlycompensation'].mean()
avg_of_some_college = some_college_df['totalyearlycompensation'].mean()
avg_of_highschool = highschool_df['totalyearlycompensation'].mean()

data = [['High School', avg_of_highschool], ['Some College', avg_of_some_college],
        ['Bachelors', avg_of_bachelors],['Masters', avg_of_masters], ['Doctorate', avg_of_doctorate]]

degree_df = pd.DataFrame(data, columns = ['Type of Degree', 'Averages of Pay'])
degree_df.set_index('Type of Degree', inplace=True)
degree_df.plot.bar()
```

How does pay vary with the amount of education received?



On average, having completed a high school education or having some college can result in a pay similar to that of a Master's degree. These are probably exceptional cases in which people have taken it on themselves to become really good at their field. Otherwise, pay clearly increases the further into you college education you go with Bachelor's degrees getting paid least and Doctorates the most.

Correlation Analysis

```
random = df.sample(10, replace=False, axis=0)
random
```

	timestamp	company	level	title	totalyearlycompensation	location	yearsofexperience	yearsatcompany	tag	basesalary	...
24577	7/20/2020 23:21:02	Apple	ICT4	Technical Program Manager	300000	Cupertino, CA	9.0	1.0	iOS	170000.0	...
23682	7/6/2020 17:26:30	Oracle	IC-2	Software Engineer	195000	Seattle, WA	0.0	0.0	Distributed Systems (Back-End)	120000.0	...
44519	3/10/2021 8:45:00	Facebook	E4	Data Scientist	142000	Austin, TX	3.0	0.0	Data analyst	128000.0	...
9886	8/20/2019 0:24:30	Microsoft	64	Software Engineering Manager	227000	Redmond, WA	16.0	14.0	Distributed Systems (Back-End)	163000.0	...
20717	5/9/2020 13:32:07	Amazon	L4	Software Engineer	150000	Toronto, ON, Canada	2.0	0.0	Distributed Systems (Back-End)	118000.0	...
38201	12/27/2020 21:04:12	Amazon	L5	Technical Program Manager	221000	Sunnyvale, CA	3.0	0.0	API Development (Back-End)	165000.0	...
53624	6/7/2021 1:52:10	Microsoft	62	Software Engineer	248000	Sunnyvale, CA	8.0	8.0	Full Stack	167000.0	...
34162	10/29/2020 19:18:48	Uber	L5A	Software Engineer	260000	San Francisco, CA	8.0	5.0	DevOps	180000.0	...
1773	10/4/2018 18:13:40	Facebook	E5	Software Engineer	400000	Menlo Park, CA	10.0	6.0	Web Development (Front-End)	0.0	...
16191	2/6/2020 13:45:23	Microsoft	64	Solution Architect	248000	Los Angeles, CA	10.0	1.0	Security	165000.0	...

10 rows × 30 columns

```
company = pd.crosstab(random.totalyearlycompensation, random.yearsatcompany)

c, p, dof, expected = chi2_contingency(company)
c, p, dof
```

(42.50000000000001, 0.36387004160284564, 40)

Correlation Between Salary and Years of Experience

```
experience = pd.crosstab(random.totalyearlycompensation, random.yearsofexperience)


c, p, dof, expected = chi2_contingency(experience)
c, p, dof
```

(55.00000000000001, 0.22671477824213157, 48)

Correlation Between Salary and Level of Education

```
education = pd.crosstab(random.totalyearlycompensation, random.num_degrees)


c, p, dof, expected = chi2_contingency(education)
c, p, dof
```

(10.000000000000004, 0.2650259152973615, 8)

Correlation Between Salary and Race

```
random["num_races"] = random["Race_Asian"] + random["Race_White"] + random["Race_Black"]
                    + random["Race_Hispanic"] + random["Race_Two_Or_More"]

races = pd.crosstab(random.totalyearlycompensation, random.num_races)

c, p, dof, expected = chi2_contingency(races)
c, p, dof
```

(10.000000000000004, 0.2650259152973615, 8)

Correlation Analysis - Description

Salary and Years at Company

With a degree of 40, our value on the significance table at 0.05 would be 55.76. Compared to our chi2 value of 42.50, our value is less than the significance table value. This means we accept the null hypothesis declaring that there is no correlation between the two variables.

Salary and Years of Experience

With a degree of 50, the chi2 value at 0.05 is 67.50. Compared to our chi2 value of 55.00, our value is less than the value in the significance table. This means we accept the null hypothesis declaring our two variables to have no correlation.

Salary and Level of Education

With a degree of 8, our value for chi2 is 10.00. For the significance table value at 0.05, it is 15.51. Since our value is less than the significance table value, we can accept the null hypothesis and declare our two variables to have no correlation

Salary and Race

The significance table value for 0.05 at a degree of freedom 8 is 15.51. Because our chi2 value is 10.00, we can accept the null hypothesis. This means we can declare our two variables have no correlation

Questions

Question 1

What percentage of the workforce in STEM fields have at least a Master's Degree?

Review Slide 10

Question 2

Based on our linear regression, what factor has the most significant impact on yearly income?

Review Slide 11

Question 3

Is it more likely for your total yearly income to be greater if you have a Doctorate vs a Bachelor's? If so, approximately by how much?

Review Slide 15