A Look at the Data

May 27, 2020

0.0.1 A Look at the Data

In order to get a better understanding of the data we will be looking at throughout this lesson, let's take a look at some of the characteristics of the dataset.

First, let's read in the data and necessary libraries.

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import ALookAtTheData as t
        from IPython import display
        %matplotlib inline

        df = pd.read_csv('./survey_results_public.csv')
        df.head()
```

```
Out[1]:    Respondent                                        Professional  \
        0           1                                             Student
        1           2                                             Student
        2           3                                Professional developer
        3           4  Professional non-developer who sometimes write...
        4           5                                Professional developer

                       ProgramHobby         Country      University  \
        0                 Yes, both   United States              No
        1                 Yes, both  United Kingdom  Yes, full-time
        2                 Yes, both  United Kingdom              No
        3                 Yes, both   United States              No
        4  Yes, I program as a hobby     Switzerland              No

                              EmploymentStatus  \
        0  Not employed, and not looking for work
        1                      Employed part-time
        2                      Employed full-time
        3                      Employed full-time
        4                      Employed full-time

                                          FormalEducation  \
```

```
0                                  Secondary school
1  Some college/university study without earning ...
2                                 Bachelor's degree
3                                   Doctoral degree
4                                   Master's degree

                                MajorUndergrad  \
0                                          NaN
1       Computer science or software engineering
2       Computer science or software engineering
3  A non-computer-focused engineering discipline
4       Computer science or software engineering

                                       HomeRemote  \
0                                             NaN
1             More than half, but not all, the time
2  Less than half the time, but at least one day ...
3  Less than half the time, but at least one day ...
4                                           Never

               CompanySize       ...       StackOverflowMakeMoney Gender  \
0                      NaN       ...            Strongly disagree   Male
1       20 to 99 employees       ...            Strongly disagree   Male
2  10,000 or more employees       ...                     Disagree   Male
3  10,000 or more employees       ...                     Disagree   Male
4       10 to 19 employees       ...                          NaN    NaN

  HighestEducationParents                          Race         SurveyLong  \
0             High school  White or of European descent  Strongly disagree
1       A master's degree  White or of European descent     Somewhat agree
2   A professional degree  White or of European descent     Somewhat agree
3       A doctoral degree  White or of European descent              Agree
4                     NaN                           NaN               NaN

  QuestionsInteresting QuestionsConfusing InterestedAnswers    Salary  \
0       Strongly agree           Disagree    Strongly agree       NaN
1       Somewhat agree           Disagree    Strongly agree       NaN
2                Agree           Disagree             Agree  113750.0
3                Agree     Somewhat agree    Strongly agree       NaN
4                  NaN                NaN               NaN       NaN

   ExpectedSalary
0             NaN
1         37500.0
2             NaN
3             NaN
4             NaN
```

```
[5 rows x 154 columns]
```

As you work through the notebook(s) in this and future parts of this program, you will see some consistency in how to test your solutions to assure they match what we achieved! In every environment, there is a solution file and a test file. There will be checks for each solution built into each notebook, but if you get stuck, you may also open the solution notebook to see how we find any of the solutions. Let's take a look at an example.

0.0.2 Question 1

1. Provide the number of rows and columns in this dataset.

```
In [2]: # We solved this one for you by providing the number of rows and columns:
        # You can see how we are prompted that we solved for the number of rows and cols correct

        num_rows = df.shape[0] #Provide the number of rows in the dataset
        num_cols = df.shape[1] #Provide the number of columns in the dataset

        t.check_rows_cols(num_rows, num_cols)
```

```
Nice job there are 19102 rows in the dataset!
Nice job there are 154 columns in the dataset!
```

```
In [3]: # If we made a mistake - a different prompt will appear

        flipped_num_rows = df.shape[1] #Provide the number of rows in the dataset
        flipped_num_cols = df.shape[0] #Provide the number of columns in the dataset

        t.check_rows_cols(flipped_num_rows, flipped_num_cols)
```

```
That doesn't look like what we were expecting for the number of rows.
That doesn't look like what we were expecting for the number of columns.
```

```
In [10]: # If you want to know more about what the test function is expecting,
         # you can read the documentation the same way as any other funtion

         t.check_rows_cols?
```

Now that you are familiar with how to test your code - let's have you answer your first question:

0.0.3 Question 2

2. Which columns had no missing values? Provide a set of column names that have no missing values.

```
In [4]: no_nulls = set(df.columns[df.isnull().mean()==0]) #Provide a set of columns with 0 missi

        display.HTML(t.no_null_cols(no_nulls))
```

```
Nice job that looks right!
```

```
Out[4]: <IPython.core.display.HTML object>
```

0.0.4 Question 3

3. Which columns have the most missing values? Provide a set of column names that have more than 75% if their values missing.

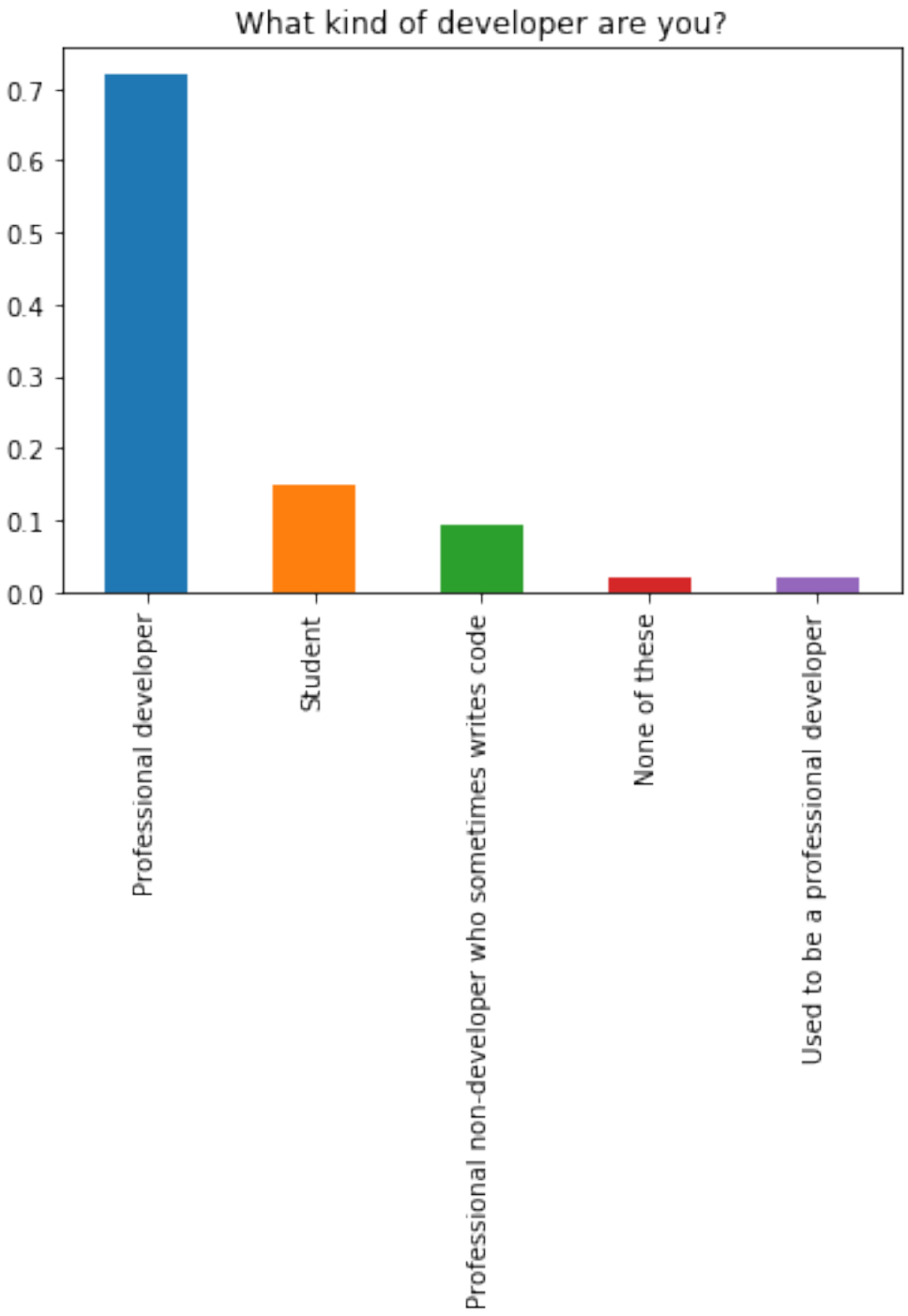
```
In [5]: most_missing_cols = set(df.columns[df.isnull().mean() > 0.75]) #Provide a set of columns

        t.most_missing_cols(most_missing_cols)
```

```
Nice job that looks right!
```

0.0.5 Question 4

4. Provide a pandas series of the different **Professional** status values in the dataset along with the count of the number of individuals with each status. Store this pandas series in **status_vals**. If you are correct, you should see a bar chart of the proportion of individuals in each status.

```
In [6]: status_vals = df.Professional.value_counts()#Provide a pandas series of the counts for e

        # The below should be a bar chart of the proportion of individuals in each professional
        # is set up correctly.

        (status_vals/df.shape[0]).plot(kind="bar");
        plt.title("What kind of developer are you?");
```

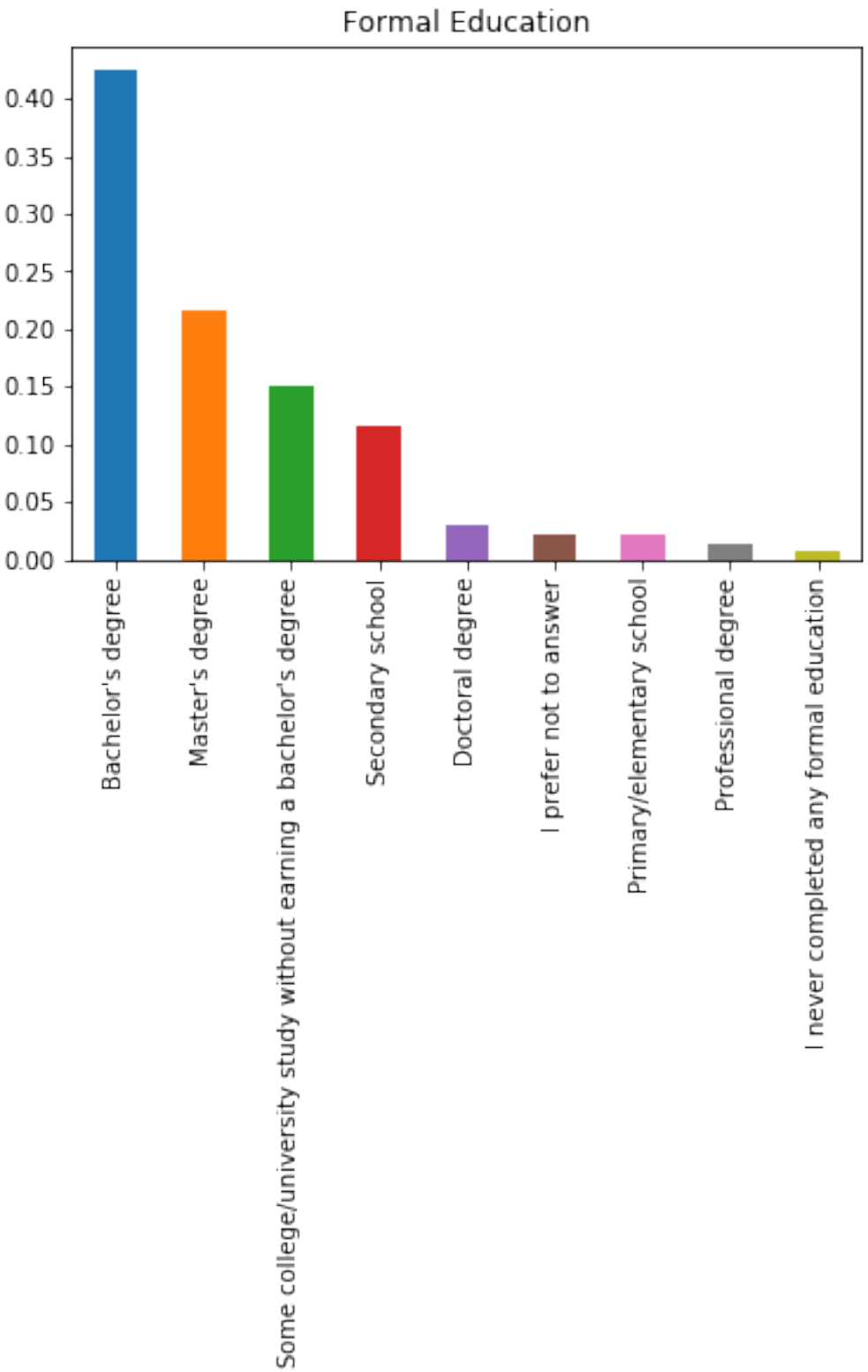


0.0.6 Question 5

5. Provide a pandas series of the different **FormalEducation** status values in the dataset along with the count of how many individuals received that formal education. Store this pandas series

in **ed_vals**. If you are correct, you should see a bar chart of the proportion of individuals in each status.

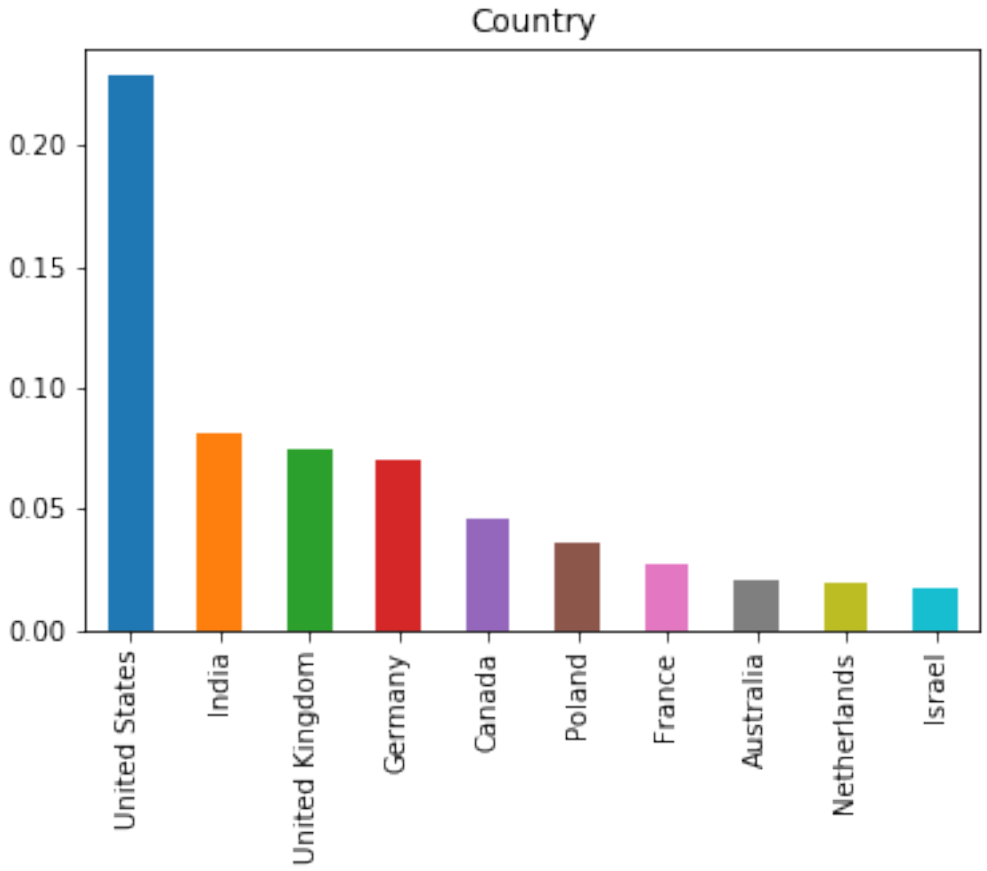
```
In [7]: ed_vals = df.FormalEducation.value_counts()#Provide a pandas series of the counts for ea

        # The below should be a bar chart of the proportion of individuals in your ed_vals
        # if it is set up correctly.

        (ed_vals/df.shape[0]).plot(kind="bar");
        plt.title("Formal Education");
```


Formal Education
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Bachelor's degree
Master's degree
Some college/university study without earning a bachelor's degree
Secondary school
Doctoral degree
I prefer not to answer
Primary/elementary school
Professional degree
I never completed any formal education

0.0.7 Question 6

6. Provide a pandas series of the different **Country** values in the dataset along with the count of how many individuals are from each country. Store this pandas series in **count_vals**. If you are correct, you should see a bar chart of the proportion of individuals in each country.

```
In [8]: count_vals = df.Country.value_counts()#Provide a pandas series of the counts for each Co

        # The below should be a bar chart of the proportion of the top 10 countries for the
        # individuals in your count_vals if it is set up correctly.

        (count_vals[:10]/df.shape[0]).plot(kind="bar");
        plt.title("Country");
```



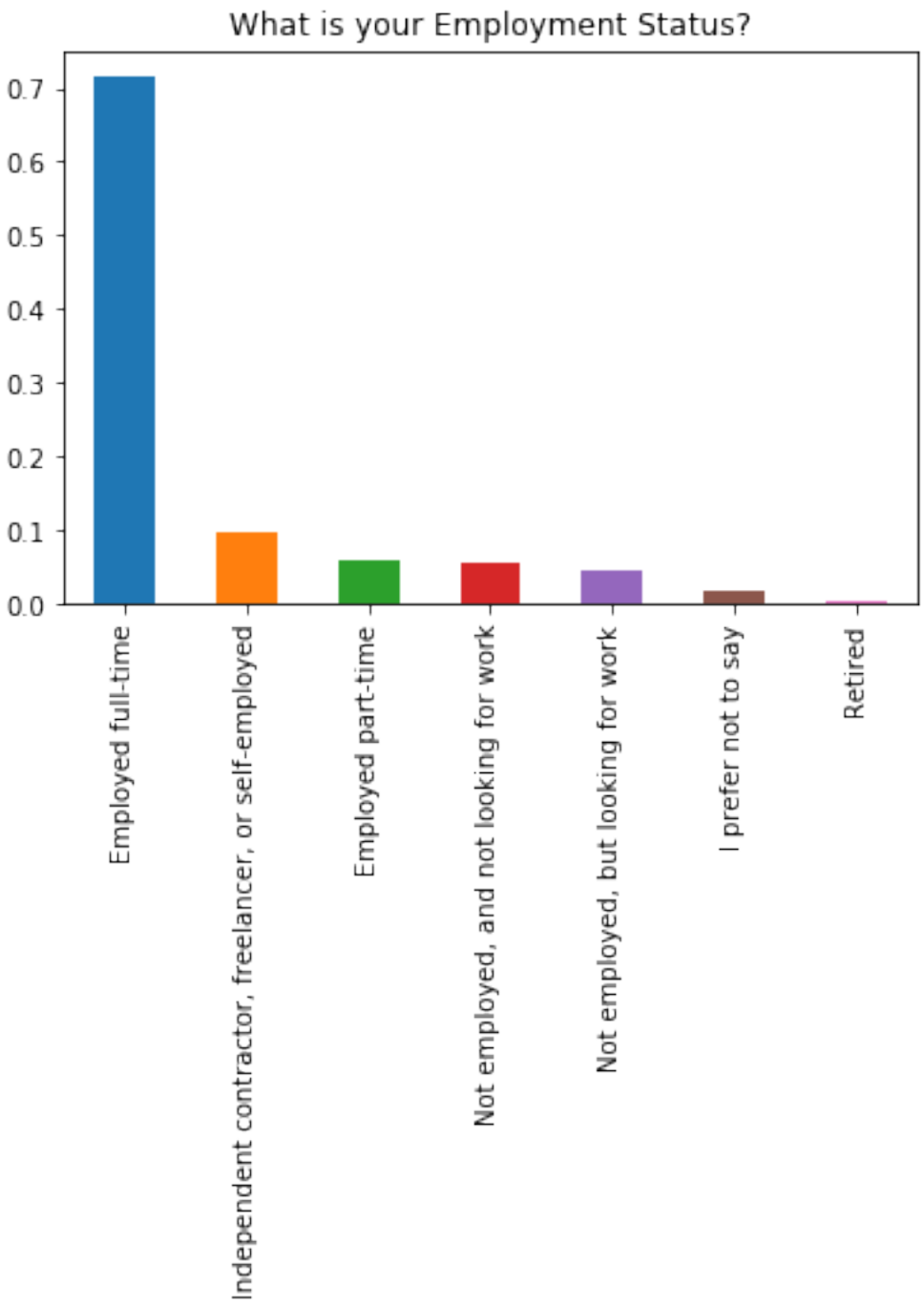
Feel free to explore the dataset further to gain additional familiarity with the columns and rows in the dataset. You will be working pretty closely with this dataset throughout this lesson.

```
In [9]: pd.DataFrame(df.query("Professional == 'Professional developer' and (Gender == 'Male' or

Out[9]:                                                                    Salary
        Gender FormalEducation
```

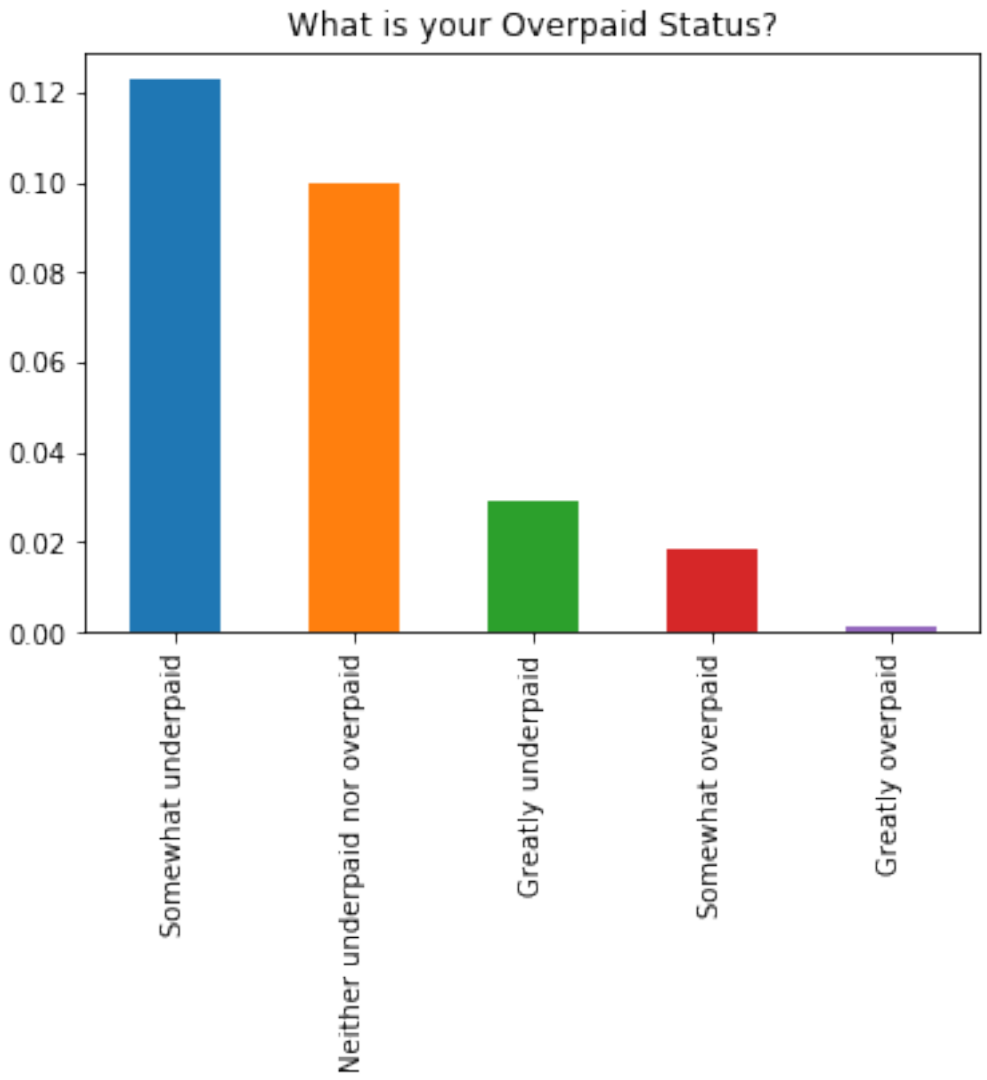
```
Female  Bachelor's degree                                    59901.894205
        Doctoral degree                                      83332.583351
        I prefer not to answer                               20691.397849
        Master's degree                                      51395.305901
        Professional degree                                  67521.095365
        Secondary school                                     38673.862023
        Some college/university study without earning a...   45149.521658
Male    Bachelor's degree                                    59538.372951
        Doctoral degree                                      77120.265378
        I never completed any formal education               43956.636637
        I prefer not to answer                               40850.596080
        Master's degree                                      61632.442196
        Primary/elementary school                            64022.483103
        Professional degree                                  45662.824726
        Secondary school                                     39544.122310
        Some college/university study without earning a...   60420.217368
```

In [12]:
```
# Data Understanding
def display_bar_chart(df, column, title):
    '''
    Displays a bar chart with a title

    Parameters:
    df: a dataframe
    column: the column which we want to show
    title: the title of the chart

    Returns:
    None

    '''
    status_vals = df[column].value_counts()
    (status_vals[:10]/df.shape[0]).plot(kind="bar");
    plt.title(title);
```

In [14]:
```
#Provide a panda series of counts for each employment status

display_bar_chart(df, "EmploymentStatus","What is your Employment Status?")
```




```
In [15]: #Provide a pandas series of the counts for each Overpaid status
         display_bar_chart(df, "Overpaid", "What is your Overpaid Status?")
```




```
In [18]: def handling_country(df):
             '''

             Return a dataframe with country seperate into west and east

             Parameters:
             df: a raw data dataframe

             Returns:
             df: a dataframe with a new column west_or_east

             '''
             # For Categorical variables "Country", we seperate them into
             # three sessions: western, eastern and other
```

```
        # The following lists define the countries to western and eastern
        western = ['United States', 'Liechtenstein', 'Switzerland',
               'Iceland', 'Norway', 'Israel', 'Denmark',
               'Ireland', 'Canada', 'United Kingdom', 'Germany',
               'Netherlands', 'Sweden', 'Luxembourg', 'Austria',
               'Finland', 'France', 'Belgium', 'Spain', 'Italy',
               'Poland']

        eastern = ['Thailand', 'Singapore', 'Hong Kong',
               'South Korea', 'Japan', 'China',
               'Taiwan', 'Malaysia', 'India',
               'Indonesia', 'Vietnam']

        #Add a new catagory seperating to western and eastern
        df['west_or_east'] = df['Country'].apply(lambda x: 'western' if x in western else (

        return df
```

In [19]:
```
def data_preparation(df):
    '''

    Return useful columns with query condition

    Parameters:
    df: a raw data dataframe

    Returns:
    useful_df: a filtered dataframe with only useful columns

    '''
    #Get some useful columns for analysis
    useful_columns = ['Country', 'YearsCodedJob', 'EmploymentStatus', 'CareerSatisfacti
    useful_df = pd.DataFrame(df.query("Professional == 'Professional developer' and (Ge
    return useful_df
```

In [20]:
```
#Get some useful columns for analysis
df = handling_country(df)
useful_df = data_preparation(df)
useful_df.head()
```

Out[20]:

	Country	YearsCodedJob	EmploymentStatus	CareerSatisfaction	\
2	United Kingdom	20 or more years	Employed full-time	8.0	
7	Poland	7 to 8 years	Employed full-time	7.0	
8	Colombia	7 to 8 years	Employed full-time	6.0	
13	Germany	15 to 16 years	Employed full-time	8.0	
14	United Kingdom	20 or more years	Employed full-time	8.0	

	JobSatisfaction	JobSeekingStatus	\

```
2               9.0                                                NaN
7               7.0  I'm not actively looking, but I am open to new...
8               6.0       I am not interested in new job opportunities
13              6.0                    I am actively looking for a job
14              8.0                                                NaN

    HoursPerWeek    Salary west_or_east                      Overpaid
2            NaN  113750.0      western  Neither underpaid nor overpaid
7            1.0       NaN      western                           NaN
8            2.0       NaN        other  Neither underpaid nor overpaid
13           3.0       NaN      western                           NaN
14           NaN  100000.0      western             Somewhat underpaid
```

In [22]:
```
def handling_overpaid(df):
    """

    Convert Overpaid from words to integer for calculating the mean

    Parameters:
    df: a dataframe that will be converted

    Returns:
    dataframe: a converted dataframe with Overpaid column becomes measurable

    """
    overpaid_map = {
        'Greatly underpaid' : 1,
        'Somewhat underpaid' : 2,
        'Neither underpaid nor overpaid' : 3,
        'Somewhat overpaid' : 4,
        'Greatly overpaid' : 5,
        np.nan: np.nan
    }
    df['Overpaid'] = df['Overpaid'].apply(lambda x: np.nan if x == np.nan else overpaid

    return df
```

In [23]:
```
#Compare selected indicators between western and eastern
useful_df = handling_overpaid(useful_df)
comparison = useful_df.groupby(['west_or_east','YearsCodedJob']).mean()
comparison
```

Out[23]:

		CareerSatisfaction	JobSatisfaction	\
west_or_east	YearsCodedJob			
eastern	1 to 2 years	6.948148	6.118519	
	10 to 11 years	7.666667	6.714286	
	11 to 12 years	7.062500	6.875000	
	12 to 13 years	7.466667	6.200000	

	13 to 14 years	7.250000	7.500000
	14 to 15 years	6.666667	6.833333
	15 to 16 years	6.500000	6.500000
	16 to 17 years	7.000000	6.000000
	17 to 18 years	8.000000	8.000000
	18 to 19 years	8.000000	8.500000
	19 to 20 years	7.000000	7.000000
	2 to 3 years	7.205128	6.418803
	20 or more years	7.428571	6.000000
	3 to 4 years	6.902439	6.182927
	4 to 5 years	7.025000	6.612500
	5 to 6 years	7.000000	6.218750
	6 to 7 years	6.694444	6.611111
	7 to 8 years	6.954545	6.818182
	8 to 9 years	6.850000	6.600000
	9 to 10 years	6.851852	6.296296
	Less than a year	6.424242	6.208955
other	1 to 2 years	7.562842	7.000000
	10 to 11 years	7.625000	6.965517
	11 to 12 years	7.652174	7.413043
	12 to 13 years	7.297297	6.621622
	13 to 14 years	7.642857	6.892857
	14 to 15 years	7.866667	7.862069
	15 to 16 years	7.595238	7.619048
	16 to 17 years	7.555556	7.416667
	17 to 18 years	7.272727	7.136364
...		...	...
	20 or more years	7.581081	7.337838
	3 to 4 years	7.445026	6.868421
	4 to 5 years	7.594286	7.297143
	5 to 6 years	7.274854	6.842105
	6 to 7 years	7.474747	6.868687
	7 to 8 years	7.711538	7.384615
	8 to 9 years	7.421687	6.710843
	9 to 10 years	7.463158	6.968421
	Less than a year	7.024691	6.876543
western	1 to 2 years	7.694915	7.248408
	10 to 11 years	7.542553	7.049822
	11 to 12 years	7.547297	7.299320
	12 to 13 years	7.558824	7.066176
	13 to 14 years	7.546296	6.935185
	14 to 15 years	7.623077	7.076923
	15 to 16 years	7.506250	7.118750
	16 to 17 years	7.744186	7.170543
	17 to 18 years	7.700000	7.218182
	18 to 19 years	7.513889	6.986111
	19 to 20 years	7.457143	6.885714
	2 to 3 years	7.571702	7.260038

	20 or more years	7.935135	7.388087
	3 to 4 years	7.587234	7.089362
	4 to 5 years	7.484091	7.015873
	5 to 6 years	7.590909	6.990431
	6 to 7 years	7.775439	7.291228
	7 to 8 years	7.557252	7.275862
	8 to 9 years	7.424893	7.077253
	9 to 10 years	7.530405	7.037162
	Less than a year	7.844749	7.675799

		HoursPerWeek	Salary	Overpaid
west_or_east	YearsCodedJob			
eastern	1 to 2 years	4.738636	10651.675740	2.233333
	10 to 11 years	5.200000	39969.486742	2.857143
	11 to 12 years	2.600000	62051.574699	2.500000
	12 to 13 years	3.000000	64672.604107	2.200000
	13 to 14 years	26.000000	62011.771544	2.666667
	14 to 15 years	4.666667	99370.629371	3.000000
	15 to 16 years	NaN	52855.674644	2.000000
	16 to 17 years	2.500000	50448.430493	2.000000
	17 to 18 years	NaN	NaN	NaN
	18 to 19 years	NaN	NaN	NaN
	19 to 20 years	0.666667	16150.345030	1.000000
	2 to 3 years	7.169492	9612.203018	2.137255
	20 or more years	0.500000	69920.958916	3.000000
	3 to 4 years	7.795455	16683.666527	2.233333
	4 to 5 years	7.976190	13889.764320	2.264706
	5 to 6 years	3.775000	19018.820888	2.250000
	6 to 7 years	4.375000	18402.376248	2.166667
	7 to 8 years	6.272727	39273.260237	2.333333
	8 to 9 years	2.181818	32829.053816	2.285714
	9 to 10 years	7.045455	27176.671097	2.000000
	Less than a year	4.783784	10551.554791	2.615385
other	1 to 2 years	2.889908	15700.318488	2.210526
	10 to 11 years	1.844444	36709.296802	2.500000
	11 to 12 years	0.863636	36143.501376	2.380952
	12 to 13 years	1.375000	48040.277445	2.470588
	13 to 14 years	2.285714	53714.902842	2.714286
	14 to 15 years	1.461538	57618.154748	2.642857
	15 to 16 years	1.000000	50865.747566	2.450000
	16 to 17 years	1.538462	43250.277699	2.333333
	17 to 18 years	0.923077	47927.242222	2.538462
...		...	...	...
	20 or more years	2.060606	62387.842106	2.538462
	3 to 4 years	3.967742	20998.815378	2.340659
	4 to 5 years	3.311111	24075.795555	2.455696
	5 to 6 years	3.089888	29026.651706	2.390244
	6 to 7 years	3.760000	31239.968422	2.472727

```
                    7 to 8 years          2.413043   39917.755721  2.620000
                    8 to 9 years          1.525000   30812.746141  2.525000
                    9 to 10 years         2.893617   31581.396582  2.490909
                    Less than a year      5.150000   13707.241332  2.250000
western             1 to 2 years          1.838235   48123.772554  2.380952
                    10 to 11 years        2.094488   77994.156887  2.494186
                    11 to 12 years        1.142857   84817.295256  2.493506
                    12 to 13 years        1.360000   99718.849348  2.696203
                    13 to 14 years        1.454545   87533.219797  2.565217
                    14 to 15 years        1.704918   89701.387143  2.701299
                    15 to 16 years        2.409836   88129.814607  2.465116
                    16 to 17 years        2.942308   84798.720367  2.507042
                    17 to 18 years        1.487179   99617.367013  2.750000
                    18 to 19 years        2.435897   89811.641376  2.710526
                    19 to 20 years        3.230769   90883.199023  2.472222
                    2 to 3 years          2.366972   52970.914900  2.322388
                    20 or more years      2.429224  105635.073958  2.485915
                    3 to 4 years          1.902778   58861.141690  2.295139
                    4 to 5 years          1.614583   63276.310318  2.253521
                    5 to 6 years          1.928962   64706.376967  2.408730
                    6 to 7 years          1.837398   69989.310207  2.452229
                    7 to 8 years          1.781818   72434.711370  2.496855
                    8 to 9 years          2.391753   74856.521782  2.419118
                    9 to 10 years         1.680328   79016.849514  2.470930
                    Less than a year      2.260870   45661.718139  2.464000

[63 rows x 5 columns]
```

In [24]:
```python
def handling_yearscodedjob(df):
    """

    Convert the working year to integer for calculating the mean

    Parameters:
    df: a dataframe that will be converted

    Returns:
    dataframe: a converted dataframe with YearsCodedJob column becomes measurable

    """
    year_map = {'1 to 2 years' : 1,
                '10 to 11 years' : 10,
                '11 to 12 years' : 11,
                '12 to 13 years' : 12,
                '13 to 14 years' : 13,
                '14 to 15 years' : 14,
                '15 to 16 years' : 15,
                '16 to 17 years' : 16,
```

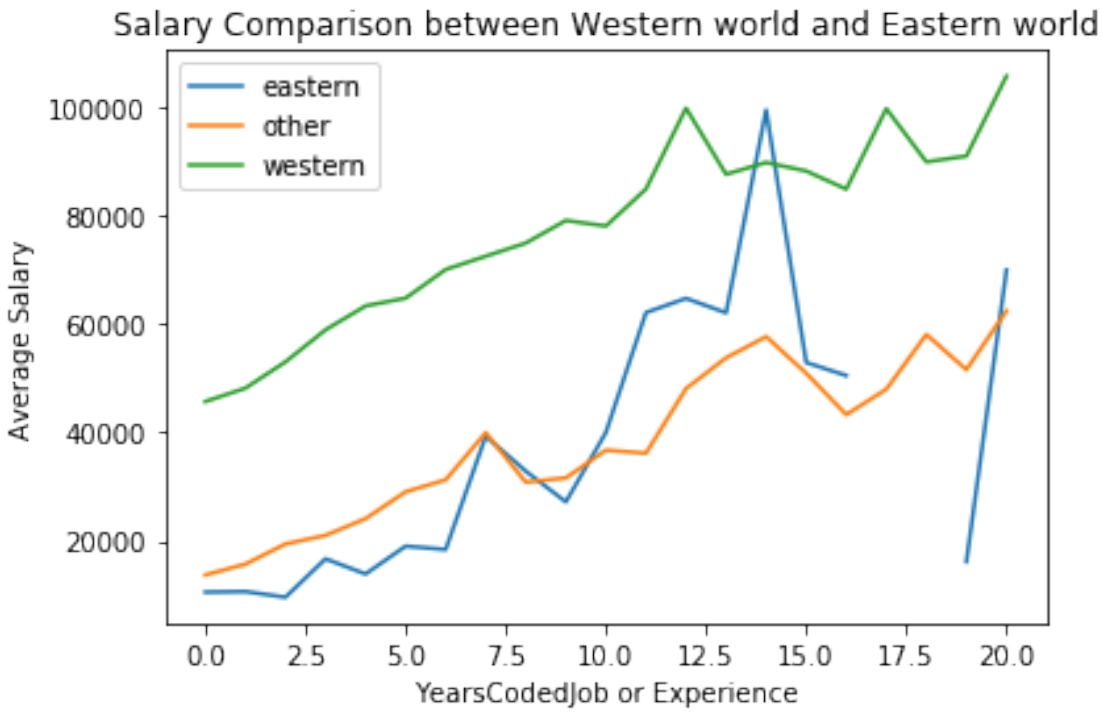
```
                        '17 to 18 years' : 17,
                        '18 to 19 years' : 18,
                        '19 to 20 years' : 19,
                        '2 to 3 years' : 2,
                        '20 or more years' : 20,
                        '3 to 4 years' : 3,
                        '4 to 5 years' : 4,
                        '5 to 6 years' : 5,
                        '6 to 7 years' : 6,
                        '7 to 8 years' : 7,
                        '8 to 9 years' : 8,
                        '9 to 10 years' : 9,
                        'Less than a year' : 0}

            df_graph = df.reset_index()
            df_graph['YearsCodedJob'] = df_graph['YearsCodedJob'].apply(lambda x: year_map[x])
            df_graph['YearsCodedJob'] = pd.to_numeric(df_graph['YearsCodedJob'])

            return df_graph
```

In [25]:
```
comparison_graph = handling_yearscodedjob(comparison)
comparison_graph = comparison_graph.sort_values(by='YearsCodedJob')
```

In [26]:
```
comparison_graph.set_index('YearsCodedJob', inplace=True)
```

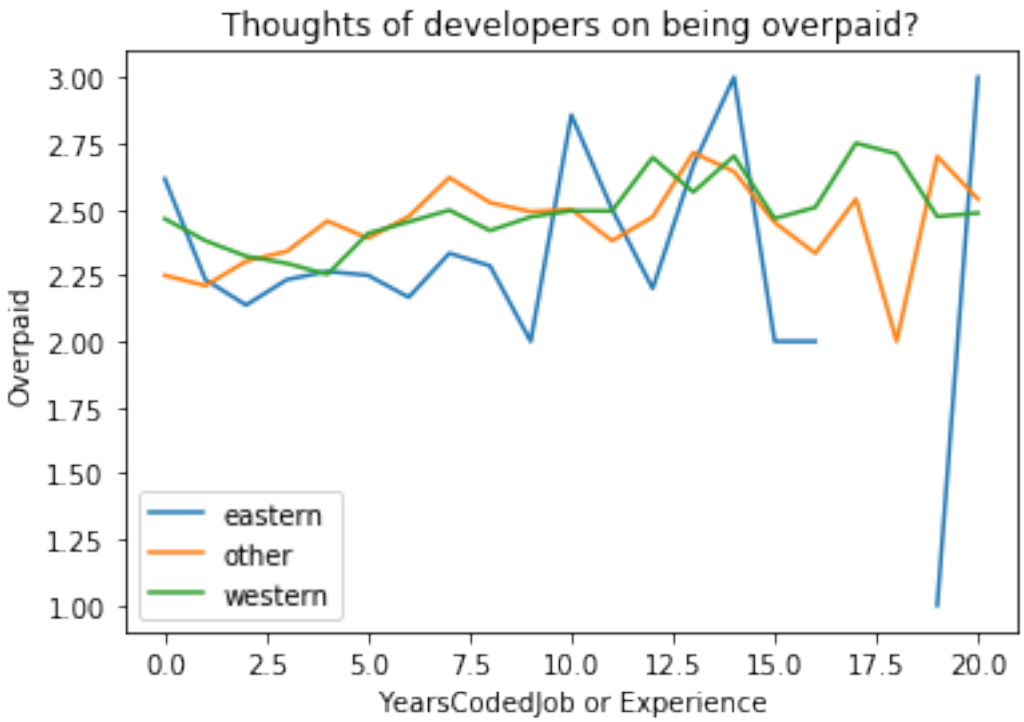
In [37]:
```
#Restuls Evaluation
#Plot the salary comparison between the Western coast and Eastern coast
comparison_graph.groupby('west_or_east')['Salary'].plot(legend=True)
plt.title("Salary Comparison between Western world and Eastern world");
plt.xlabel('YearsCodedJob or Experience')
plt.ylabel('Average Salary')
```

Out[37]:
```
Text(0,0.5,'Average Salary')
```



In [38]:
```
#Plot how the programmer thinks they are overpaid or not
comparison_graph.groupby('west_or_east')['Overpaid'].plot(legend=True)
plt.title("Thoughts of developers on being overpaid?");
plt.xlabel('YearsCodedJob or Experience')
plt.ylabel('Overpaid')
```

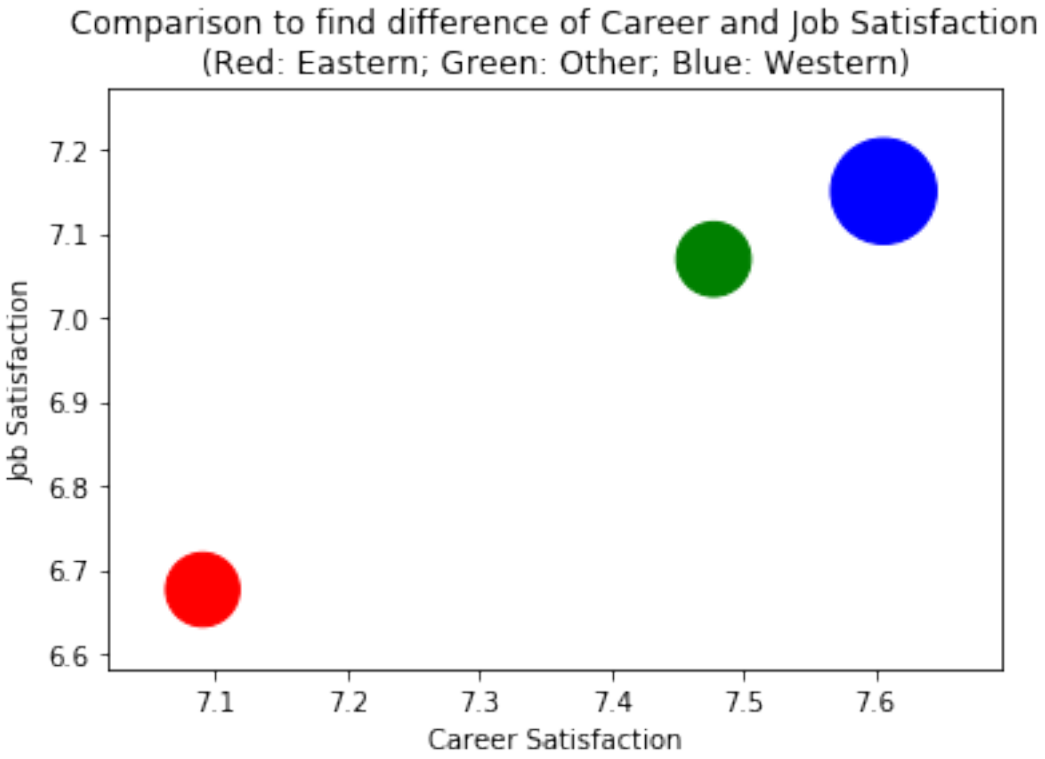
Out[38]:
```
Text(0,0.5,'Overpaid')
```




```
In [33]: comparison.groupby('west_or_east').mean().CareerSatisfaction

Out[33]: west_or_east
         eastern    7.090327
         other      7.477579
         western    7.606488
         Name: CareerSatisfaction, dtype: float64

In [34]: comparison.groupby('west_or_east').mean().JobSatisfaction

Out[34]: west_or_east
         eastern    6.676603
         other      7.069556
         western    7.150297
         Name: JobSatisfaction, dtype: float64

In [35]: comparison.groupby('west_or_east').mean().Salary/50

Out[35]: west_or_east
         eastern     753.200550
         other       764.457733
         western    1550.988907
         Name: Salary, dtype: float64
```

In [39]:
```
#Plot Comparison of Career and Job Satisfaction between Western World and Eastern World
plt.scatter(comparison.groupby('west_or_east').mean().CareerSatisfaction, comparison.gr

plt.title('Comparison to find difference of Career and Job Satisfaction\n(Red: Eastern;
plt.xlabel('Career Satisfaction')
plt.ylabel('Job Satisfaction')
```

Out[39]:
```
Text(0,0.5,'Job Satisfaction')
```



In [ ]: