4.2 Lesson Summary – Exploring Pandas

It is not uncommon for data sets in data analytics to be quite unwieldy. Fortunately, Pandas offers a number of tools to efficiently target the data that you want. Even in a very large data set you can quickly get to a specific row or set of rows. You can also sort, filter, and group your data as needed.

Concept: Pandas allows you to quickly target the data you are after. The DataFrame object provides two methods to help you access the data you want, *loc()* and *iloc()*. *loc()* will return data based on the text labels you have provided. *iloc()* will target data based on integers you provide for the row and column numbers desired. For example:

*data\_located = df.loc["Row Locator", "Column Locator"]*

*other\_data\_located = df.iloc[1, 4]*

* Activity: 01-Ins\_LocAndIloc, 02-Stu\_GoodMovies

Concept: Raw data often requires processing before it is ready to be used in analysis. Often times rows will have missing data points. The simplest way to handle these aberrations is to use the *dropna()* function to remove the entire row. For example: *df = df.dropna(how='any')*. Data is not always properly formatted in a DataFrame. To store the data more efficiently and insure that it will be properly processed Pandas provides conversion functions for your data. If you have in DataFrame, *df*, with a column named "Amount" you could convert this to numeric data using the following line of code: *df['Amount'] = pd.to\_numeric(df['Amount'])*. To get a feel for all of the data in your DataFrame you can use the *value\_counts* method. To replace values in an "Employer" column and then view the new values you could use the following code:

*df['Employer'] = df['Employer'].replace({'Self Employed': 'Self-Employed', 'Self': 'Self-Employed'})*

*df['Employer'].value\_counts()*

* Activity: 03-Ins\_CleaningData, 04-Par\_PortlandCrime, 05-Evr\_PandasRecap

Concept: Combining data can be very helpful in noticing patterns. If you had a DataFrame of user purchases and you wanted to see the total purchases of each user you could use the following code:

*grouped\_user\_purchases\_df = user\_purchases\_df.groupby(['user'])*

* Activity: 06-Ins\_GroupBy, 07-Par\_Pokemon

Concept: Sorting data organizes it in a logical order. To sort user purchase data by purchase amount you could use the following code:

*sorted\_user\_purchases\_df = user\_purchases\_df.sort\_values('PurchaseAmount')*

* Activity: 08-Ins\_Sorting, 09-Stu\_SearchForTheWorst