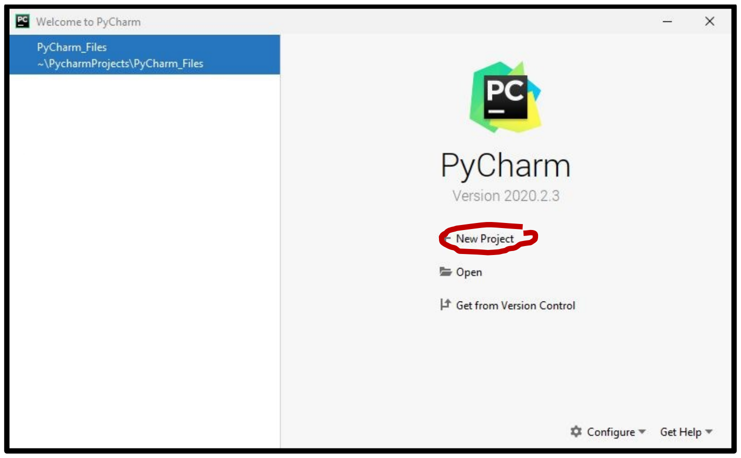
Data Wrangling

What constitutes Data Wrangling?

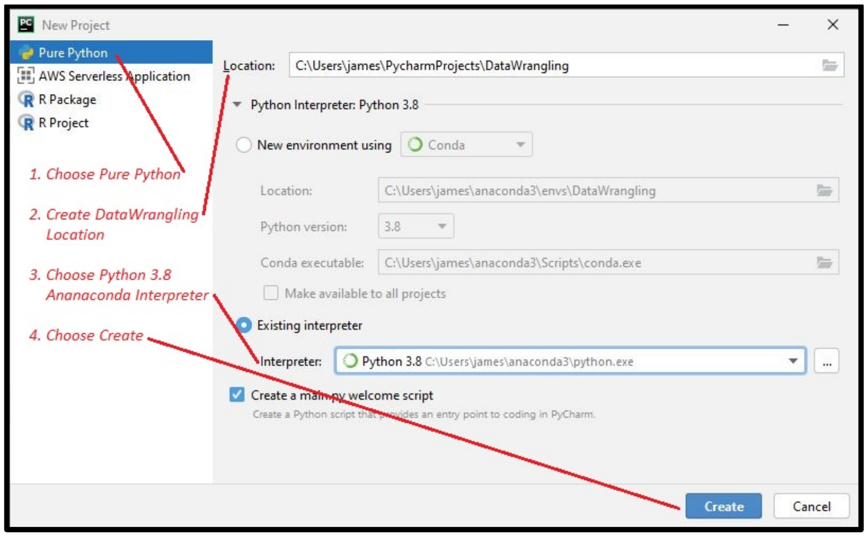
Data wrangling involves processing data in various formats like - merging, grouping, concatenating etc. for the purpose of analyzing or getting data ready to be used with another data set. Data wrangling is different from data profiling. Data profiling is used for data familiarity and inspection. Data extraction, transformation, and loading (ETL) is formalized data wrangling to collect data from operational data stores and other external sources, wrangle the data into standardized formats, and write the data into a permanent data warehouse archive. Python has built-in features to help in data wrangling methods for various data sets. For more on data wrangling not in this tutorial, refer to the URL:
https://www.tutorialspoint.com/python_data_science/python_data_wrangling.htm

Create a Data Wrangling Project in PyCharm

0) Open PyCharm and choose a New Project.



1) Choose a Pure Python project,
2) Change your project location to create a DataWrangling folder
3) Be sure to choose the Python interpreter as Python 3.8 associated with Anaconda.
4) Choose Create to open PyCharm with these settings.


New Project
Pure Python
AWS Serverless Application
R Package
R Project
1. Choose Pure Python
2. Create DataWrangling Location
3. Choose Python 3.8 Ananaconda Interpreter
4. Choose Create
Create
Cancel

Location: C:\Users\james\PycharmProjects\DataWrangling

Python Interpreter: Python 3.8

New environment using Conda

Location: C:\Users\james\anaconda3\envs\DataWrangling

Python version: 3.8

Conda executable: C:\Users\james\anaconda3\Scripts\conda.exe

Make available to all projects

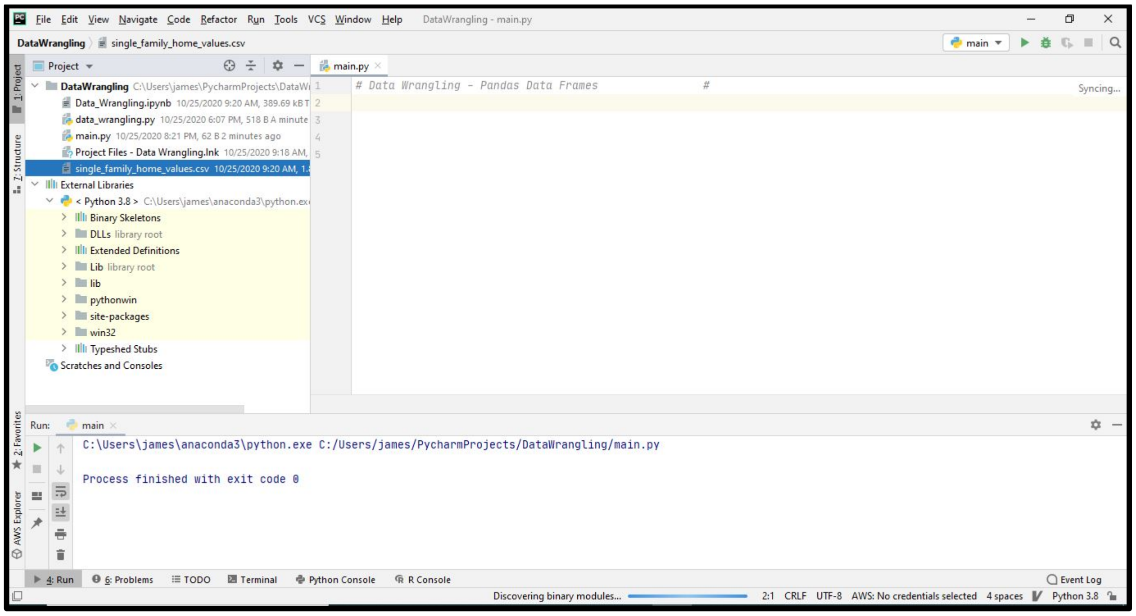
Existing interpreter

Interpreter: Python 3.8 C:\Users\james\anaconda3\python.exe

Create a main.py welcome script

Create a Python script that provides an entry point to coding in PyCharm.

5) *Download* the single_family_home_values.csv file from the Week 09 Canvas Module to the DataWrangling directory created in the last step.


File Edit View Navigate Code Refactor Run Tools VCS Window Help DataWrangling - main.py
DataWrangling 〉 single_family_home_values.csv
main
Project
DataWrangling C:\Users\james\PycharmProjects\DataWr
Data_Wrangling.ipynb 10/25/2020 9:20 AM, 389.69 kB
data_wrangling.py 10/25/2020 6:07 PM, 518 B A minute
main.py 10/25/2020 8:21 PM, 62 B 2 minutes ago
Project Files - Data Wrangling.lnk 10/25/2020 9:18 AM,
single_family_home_values.csv 10/25/2020 9:20 AM, 1.
External Libraries
< Python 3.8 > C:\Users\james\anaconda3\python.ex
Binary Skeletons
DLLs library root
Extended Definitions
Lib library root
lib
pythonwin
site-packages
win32
Typeshed Stubs
Scratches and Consoles
main.py
Syncing...
Run: main
4: Run 6: Problems TODO Terminal Python Console R Console Event Log
Discovering binary modules... 2:1 CRLF UTF-8 AWS: No credentials selected 4 spaces Python 3.8

```
# Data Wrangling - Pandas Data Frames                    #
```

```
C:\Users\james\anaconda3\python.exe C:/Users/james/PycharmProjects/DataWrangling/main.py

Process finished with exit code 0
```

Reading a CSV file into a Pandas DataFrame:

Pandas is an open-source Python Library used for high-performance data manipulation and data analysis using its powerful data structures. Pandas use Series and DataFrame on top of Numpy arrary. In this tutorial, we will look at DataFrame or df. Python with pandas is in use in a variety of academic and commercial domains, including Finance, Economics, Statistics, Advertising, Web Analytics, and more. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, organize, manipulate, model, and analyze the data. For more specific details about Pandas not covered in this tutorial, please refer to the following URL: https://www.tutorialspoint.com/python_data_science/python_pandas.htm

1) Anaconda will load the libraries in PyCharm for you. To access the three libraries we are using for Tutorial Seven (e.g., pandas, seaborn, and matplotlib.pyplot), enter the following code into Main.py in the DataWrangling project directory of PyCharm:

```
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
```

2) To format the run window to handle the size of the csv data file we will be wrangling in Tutorial Seven, enter the following Python code:

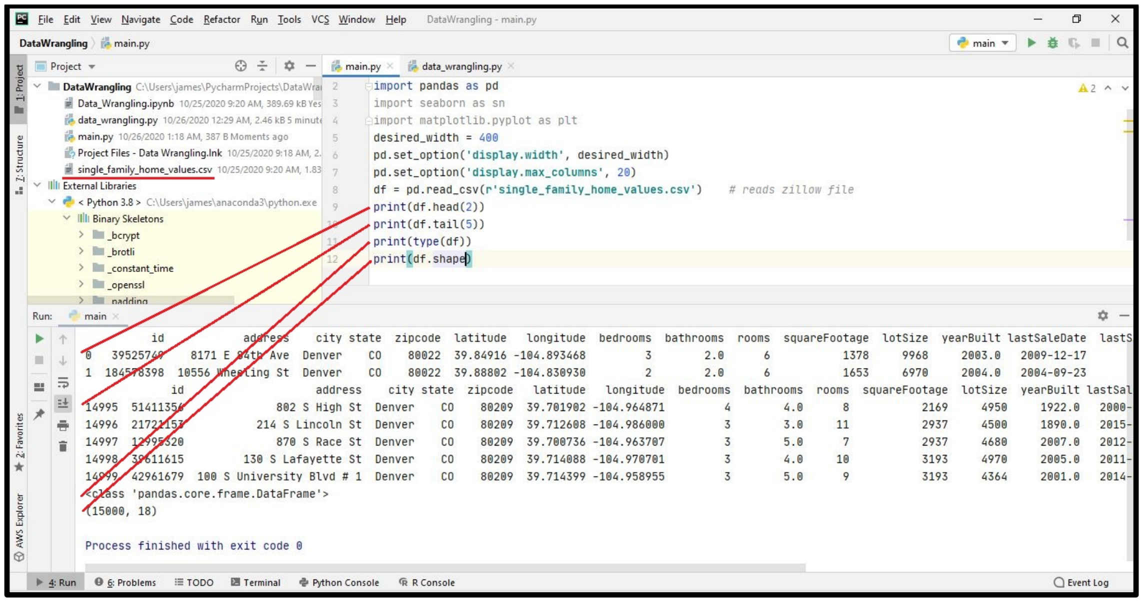
```
desired_width = 40
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
```

3) To read the single_family_home_values.csv file into a pandas dataframe, type the following Python code into main.py via PyCharm in the line below the previous code above.

```
df = pd.read_csv(r'single_family_home_values.csv')     # reads Zillow file
```

4) Once the csv file is loaded into a dataframe (df), we can begin data wrangling. Any of the following Python functions may be used with a df, once it has been assigned.

```
print(df.head(2))    # returns top two rows
print(df.tail(5))    # returns bottom five rows
print(type(df))      # returns type of df
print(df.shape)      # returns row & column count
```



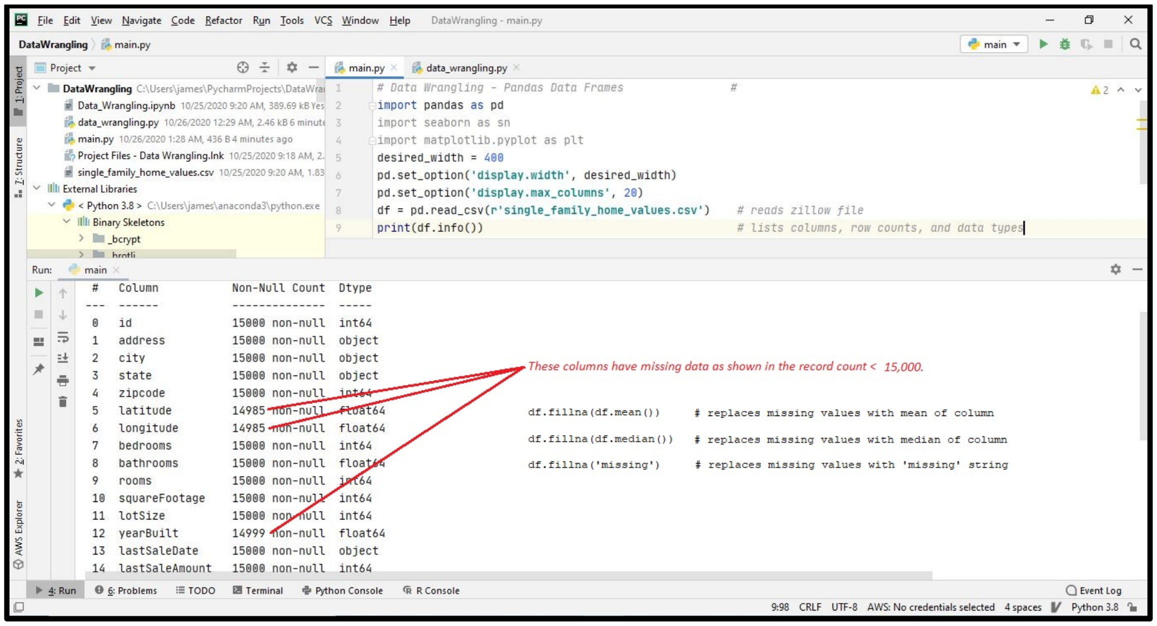
```
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')    # reads zillow file
print(df.head(2))
print(df.tail(5))
print(type(df))
print(df.shape)
```

```
          id            address    city state  zipcode  latitude   longitude  bedrooms  bathrooms  rooms  squareFootage  lotSize  yearBuilt lastSaleDate  lastS
0   39525749    8171 E 84th Ave  Denver    CO    80022  39.84916 -104.893468         3        2.0      6           1378     9968     2003.0   2009-12-17
1  184578398  10556 Wheeling St  Denver    CO    80022  39.88802 -104.830930         2        2.0      6           1653     6970     2004.0   2004-09-23
             id                  address    city state  zipcode   latitude   longitude  bedrooms  bathrooms  rooms  squareFootage  lotSize  yearBuilt lastSal
14995  51411356           802 S High St  Denver    CO    80209  39.701902 -104.964871         4        4.0      8           2169     4950     1922.0  2000-
14996  21721153        214 S Lincoln St  Denver    CO    80209  39.712608 -104.986000         3        3.0     11           2937     4500     1890.0  2015-
14997  12995320          870 S Race St  Denver    CO    80209  39.700736 -104.963707         3        5.0      7           2937     4680     2007.0  2012-
14998  39611615      130 S Lafayette St  Denver    CO    80209  39.714088 -104.970701         3        4.0     10           3193     4970     2005.0  2011-
14999  42961679  100 S University Blvd # 1  Denver    CO    80209  39.714399 -104.958955         3        5.0      9           3193     4364     2001.0  2014-
<class 'pandas.core.frame.DataFrame'>
(15000, 18)

Process finished with exit code 0
```

These images can be viewed at 400% magnification for details of the screen shots.

```
print(df.info())     # returns column attributes
```



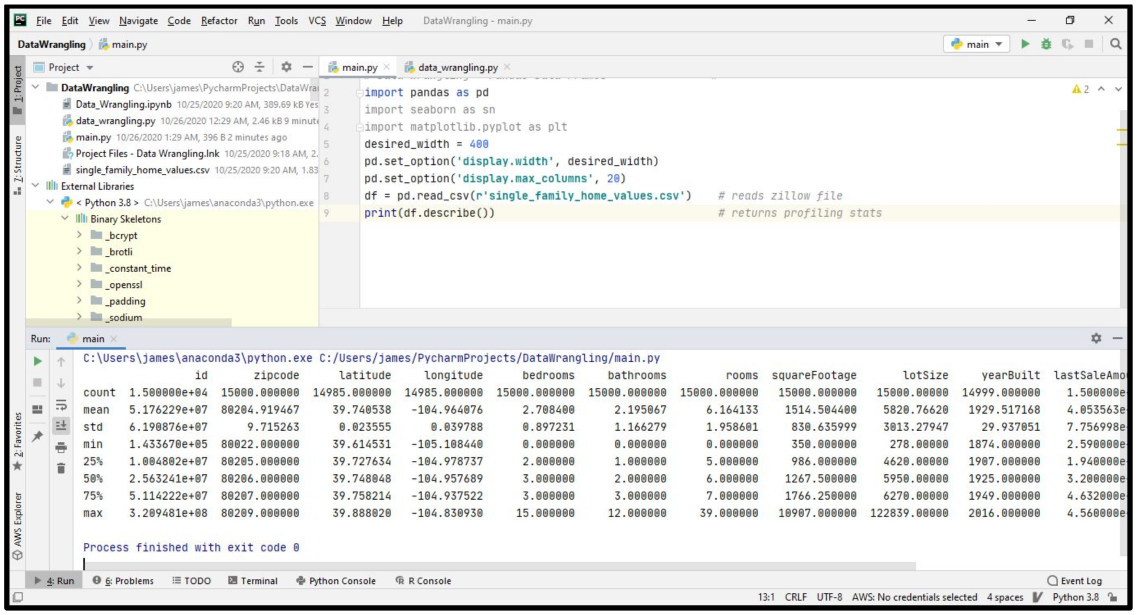
```
# Data Wrangling - Pandas Data Frames                         #
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')    # reads zillow file
print(df.info())                                      # lists columns, row counts, and data types
```

```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              15000 non-null  int64
 1   address         15000 non-null  object
 2   city            15000 non-null  object
 3   state           15000 non-null  object
 4   zipcode         15000 non-null  int64
 5   latitude        14985 non-null  float64
 6   longitude       14985 non-null  float64
 7   bedrooms        15000 non-null  int64
 8   bathrooms       15000 non-null  float64
 9   rooms           15000 non-null  int64
 10  squareFootage   15000 non-null  int64
 11  lotSize         15000 non-null  int64
 12  yearBuilt       14999 non-null  float64
 13  lastSaleDate    15000 non-null  object
 14  lastSaleAmount  15000 non-null  int64
```

These columns have missing data as shown in the record count < 15,000.

```
df.fillna(df.mean())      # replaces missing values with mean of column
df.fillna(df.median())    # replaces missing values with median of column
df.fillna('missing')      # replaces missing values with 'missing' string
```

```
print(df.describe()) # returns profiling stats
```

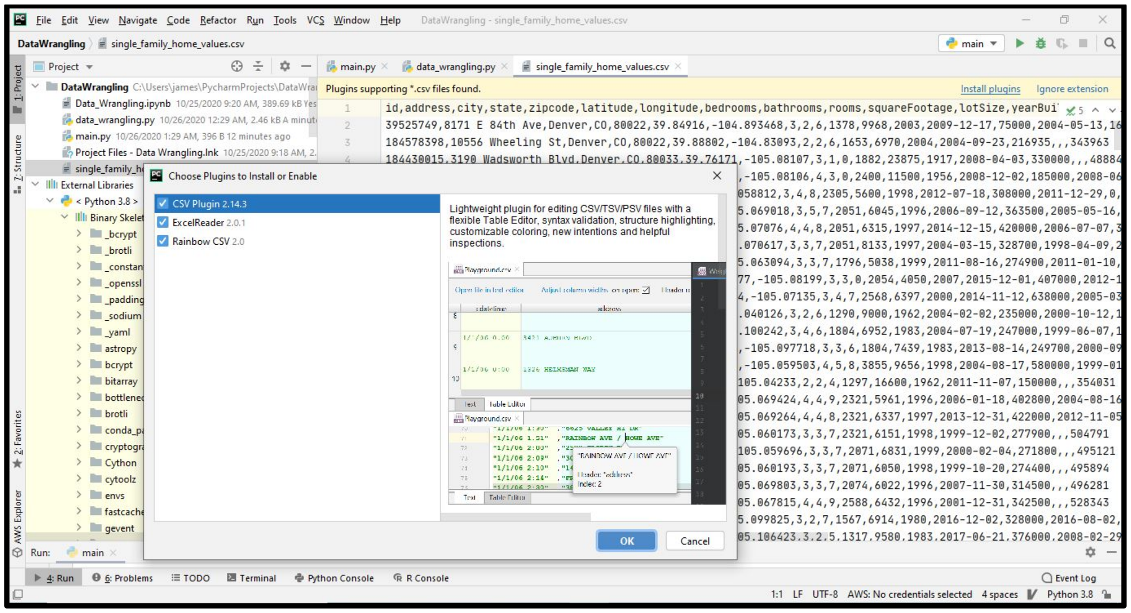


```python
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')    # reads zillow file
print(df.describe())                                  # returns profiling stats
```

```
C:\Users\james\anaconda3\python.exe C:/Users/james/PycharmProjects/DataWrangling/main.py
                 id       zipcode      latitude     longitude      bedrooms     bathrooms         rooms  squareFootage       lotSize     yearBuilt  lastSaleAmo
count  1.500000e+04  15000.000000  14985.000000  14985.000000  15000.000000  15000.000000  15000.000000   15000.000000   15000.00000  14999.000000  1.500000e
mean   5.176229e+07  80204.919467     39.740538   -104.964076      2.708400      2.195067      6.164133    1514.504400    5820.76620   1929.517168  4.053563e
std    6.190876e+07      9.715263      0.023555      0.039788      0.897231      1.166279      1.958601     830.635999    3013.27947     29.937051  7.756998e
min    1.433670e+05  80022.000000     39.614531   -105.108440      0.000000      0.000000      0.000000     350.000000     278.00000   1874.000000  2.590000e
25%    1.004802e+07  80205.000000     39.727634   -104.978737      2.000000      1.000000      5.000000     986.000000    4620.00000   1907.000000  1.940000e
50%    2.563241e+07  80206.000000     39.748048   -104.957689      3.000000      2.000000      6.000000    1267.500000    5950.00000   1925.000000  3.200000e
75%    5.114222e+07  80207.000000     39.758214   -104.937522      3.000000      3.000000      7.000000    1766.250000    6270.00000   1949.000000  4.632000e
max    3.209481e+08  80209.000000     39.888020   -104.830930     15.000000     12.000000     39.000000   10907.000000  122839.00000   2016.000000  4.560000e

Process finished with exit code 0
```

5) In PyCharm, print(df) will display the DataFrame. You can also open the csv file in PyCharm by finding the file name ***single_family_home_values.csv*** in the project window. Double click on the csv file name and it will open for you to few all 15,000 records. You can also configure PyCharm to use a 3rd-party CSV viewer as shown in the example below.



Plugins supporting *.csv files found.

```
id,address,city,state,zipcode,latitude,longitude,bedrooms,bathrooms,rooms,squareFootage,lotSize,yearBui
39525749,8171 E 84th Ave,Denver,CO,80022,39.84916,-104.893468,3,2,6,1378,9968,2003,2009-12-17,75000,2004-05-13,16
184578398,10556 Wheeling St,Denver,CO,80022,39.88802,-104.83093,2,2,6,1653,6970,2004,2004-09-23,216935,,,343963
184430015,3190 Wadsworth Blvd,Denver,CO,80033,39.76171,-105.08107,3,1,0,1882,23875,1917,2008-04-03,330000,,,48884
,-105.08106,4,3,0,2400,11500,1956,2008-12-02,185000,2008-06
058812,3,4,8,2305,5600,1998,2012-07-18,308000,2011-12-29,0,
5.069018,3,5,7,2051,6045,1996,2006-09-12,363500,2005-05-16,
5.07076,4,4,8,2051,6315,1997,2014-12-15,420000,2006-07-07,3
.070617,3,3,7,2051,8133,1997,2004-03-15,328700,1998-04-09,2
5.063094,3,3,7,1796,5038,1999,2011-08-16,274900,2011-01-10,
77,-105.08199,3,3,0,2054,4050,2007,2015-12-01,407000,2012-1
4,-105.07135,3,4,7,2568,6397,2000,2014-11-12,638000,2005-03
.040126,3,2,6,1290,9000,1962,2004-02-02,235000,2000-10-12,1
.100242,3,4,6,1804,6952,1983,2004-07-19,247000,1999-06-07,1
,-105.097718,3,3,6,1804,7439,1983,2013-08-14,249700,2000-09
,-105.059503,4,5,8,3855,9656,1998,2004-08-17,580000,1999-01
105.04233,2,2,4,1297,16600,1962,2011-11-07,150000,,,354031
05.069424,4,4,9,2321,5961,1996,2006-01-18,402800,2004-08-16
05.069264,4,4,8,2321,6337,1997,2013-12-31,422000,2012-11-05
05.060173,3,3,7,2321,6151,1998,1999-12-02,277900,,,504791
105.059696,3,3,7,2071,6831,1999,2000-02-04,271800,,,495121
05.060193,3,3,7,2071,6050,1998,1999-10-20,274400,,,495894
05.069803,3,3,7,2074,6022,1996,2007-11-30,314500,,,496281
05.067815,4,4,9,2588,6432,1996,2001-12-31,342500,,,528343
5.099825,3,2,7,1567,6914,1980,2016-12-02,328000,2016-08-02,
05.106423.3.2.5.1317.9580.1983.2017-06-21.376000.2008-02-29
```

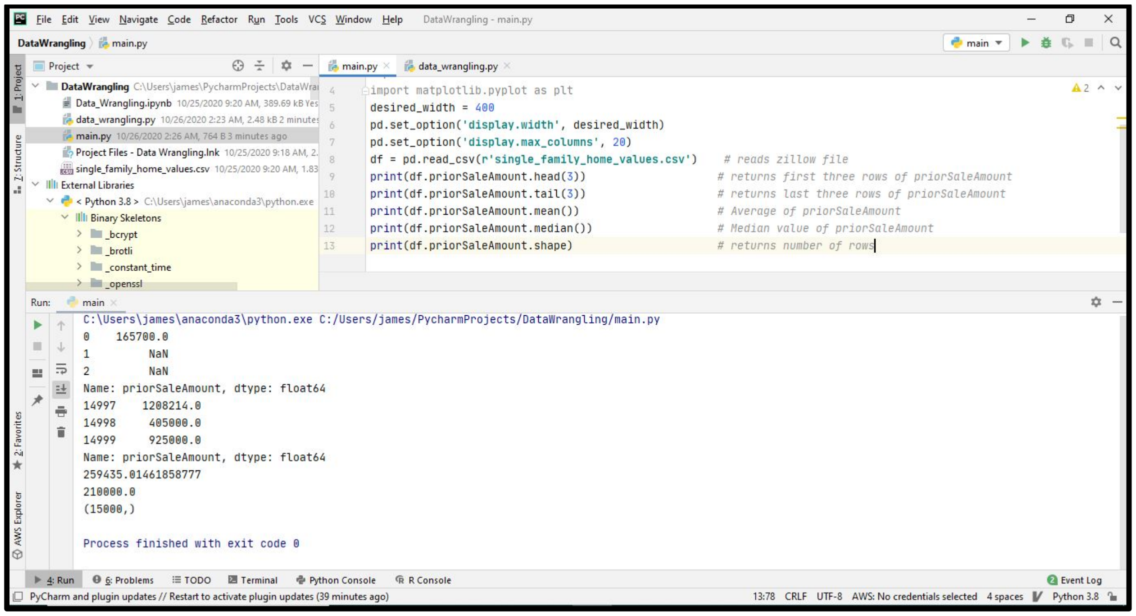
Choose Plugins to Install or Enable

- ✓ CSV Plugin 2.14.3
- ✓ ExcelReader 2.0.1
- ✓ Rainbow CSV 2.0

Lightweight plugin for editing CSV/TSV/PSV files with a flexible Table Editor, syntax validation, structure highlighting, customizable coloring, new intentions and helpful inspections.

6) The pandas df functions df.shape, df.head(), df.tail(), df.describe(), and df.shape may all be used as-is to return pandas dataframe details as illustrated in the previous screenshots. Additionally, these pandas df functions may also be used on individual column attributes. The following examples illustrate theses pandas df functions for the attribute priorSaleAmount.

df.priorSaleAmount.head(), df.priorSaleAmount.tail(), df.priorSaleAmount.shape

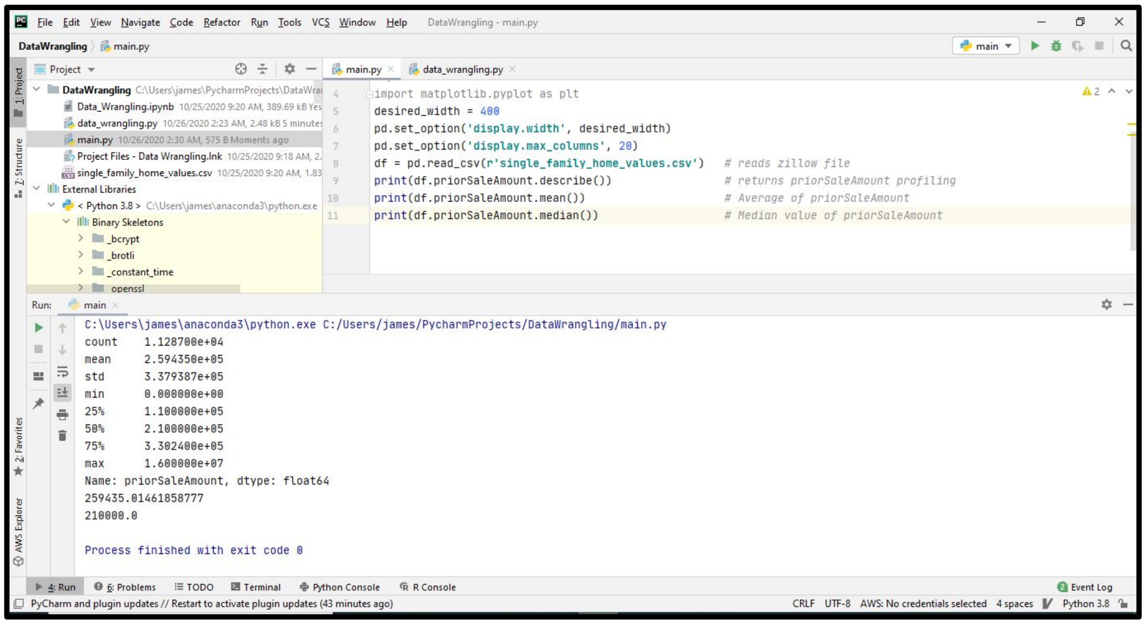


```
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')    # reads zillow file
print(df.priorSaleAmount.head(3))                     # returns first three rows of priorSaleAmount
print(df.priorSaleAmount.tail(3))                     # returns last three rows of priorSaleAmount
print(df.priorSaleAmount.mean())                      # Average of priorSaleAmount
print(df.priorSaleAmount.median())                    # Median value of priorSaleAmount
print(df.priorSaleAmount.shape)                       # returns number of rows
```

```
C:\Users\james\anaconda3\python.exe C:/Users/james/PycharmProjects/DataWrangling/main.py
0    165700.0
1         NaN
2         NaN
Name: priorSaleAmount, dtype: float64
14997    1208214.0
14998     405000.0
14999     925000.0
Name: priorSaleAmount, dtype: float64
259435.01461858777
210000.0
(15000,)

Process finished with exit code 0
```

df.priorSaleAmount.describe(), df.priorSaleAmount.mean(), df.priorSaleAmount.median()



```
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')    # reads zillow file
print(df.priorSaleAmount.describe())                  # returns priorSaleAmount profiling
print(df.priorSaleAmount.mean())                      # Average of priorSaleAmount
print(df.priorSaleAmount.median())                    # Median value of priorSaleAmount
```

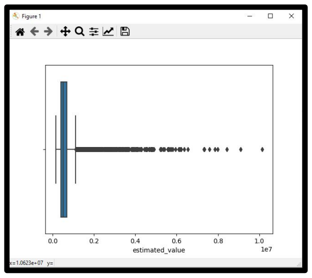
```
C:\Users\james\anaconda3\python.exe C:/Users/james/PycharmProjects/DataWrangling/main.py
count    1.128700e+04
mean     2.594350e+05
std      3.379387e+05
min      0.000000e+00
25%      1.100000e+05
50%      2.100000e+05
75%      3.302400e+05
max      1.600000e+07
Name: priorSaleAmount, dtype: float64
259435.01461858777
210000.0

Process finished with exit code 0
```

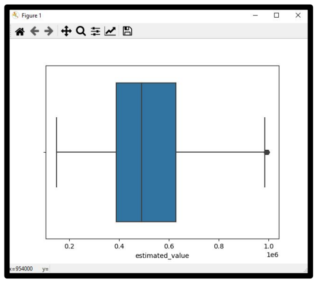
7) **Exercise deliverable:** Construct three examples of pandas df functions in Python code. Use either the single_family_home_values.csv or your data project csv. (1)The first PyCharm screenshot should include Python code on importing pandas, reading the csv file, and displaying the attribute columns. (2)The second PyCharm screenshot should be using the df.describe() function on an attribute or the entire dataframe. (3)The third PyCharm screenshot example should show the first two rows, last three rows, as well as the median of an attribute column. Capture the three PyCharm screenshots for use later in this tutorial.

Dealing with missing data and outliers:

The pandas df function df.info() lists all the column attributes and the number of rows associated with each attribute. As noted earlier, df.fillna() provides opportunity to compensate for missing data. Outliers can also be identified in a bloxplot and then screened from being used in data analysis. The boxplots of estimated_value data points created using the Python code below them show the impact of outliers. Notice how the tail on the boxplot with outliers is troublesome:



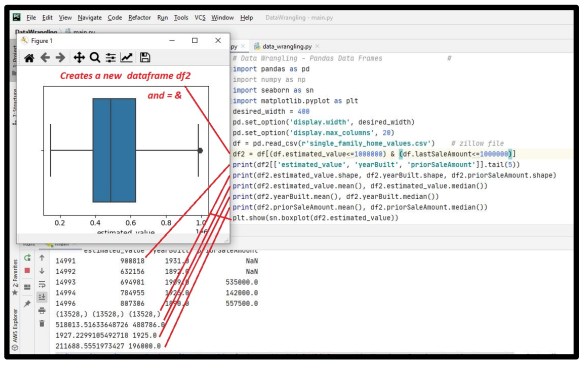
With Outliers



Without Outliers

```
# Data Wrangling - Pandas Data Frames                    #
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')    # zillow file
print(df[['estimated_value', 'yearBuilt', 'priorSaleAmount']].tail(5))
print(df.estimated_value.shape, df.yearBuilt.shape, df.priorSaleAmount.shape)
sn.boxplot(df.estimated_value)
plt.show()
```

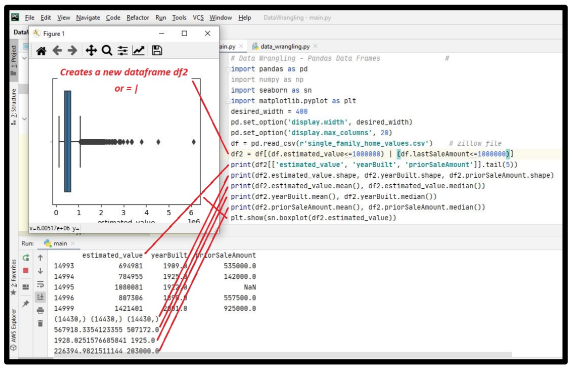
The following PyCharm screenshots creates a df2 dataframe to screen homes whose last sale was over $1,000,000 and/or estimated value is over $1,000,000. The two example screenshots use an intersection of homes estimated below and (&) previously sold below $1,000,000 as well as a union between homes estimated below $1,000,000 or (|)previously sold for less than $1,000,000.


File Edit View Navigate Code Refactor Run Tools VCS Window Help DataWrangling - main.py
Figure 1
data_wrangling.py
Creates a new dataframe df2
and = &
0.2
0.4
0.6
0.8
1.0
1e6
estimated_value
Run: main
2: Favorites
AWS Explorer

```
# Data Wrangling - Pandas Data Frames                    #
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')    # zillow file
df2 = df[(df.estimated_value<=1000000) & (df.lastSaleAmount<=1000000)]
print(df2[['estimated_value', 'yearBuilt', 'priorSaleAmount']].tail(5))
print(df2.estimated_value.shape, df2.yearBuilt.shape, df2.priorSaleAmount.shape)
print(df2.estimated_value.mean(), df2.estimated_value.median())
print(df2.yearBuilt.mean(), df2.yearBuilt.median())
print(df2.priorSaleAmount.mean(), df2.priorSaleAmount.median())
plt.show(sn.boxplot(df2.estimated_value))
```

```
       estimated_value  yearBuilt  priorSaleAmount
14991           900818     1931.0              NaN
14992           632156     1892.0              NaN
14993           694981     1909.0         535000.0
14994           784955     1925.0         142000.0
14996           807306     1890.0         557500.0
(13528,) (13528,) (13528,)
518013.51633648726 488786.0
1927.2299105492718 1925.0
211688.5551973427 196000.0
```

estimated_value and priorSalesAmount


File Edit View Navigate Code Refactor Run Tools VCS Window Help DataWrangling - main.py
Figure 1
main.py
data_wrangling.py
Creates a new dataframe df2
or = |
0
1
2
3
4
5
6
1e6
estimated_value
x=6.00517e+06 y=
Run: main
2: Favorites
AWS Explorer

```
# Data Wrangling - Pandas Data Frames                    #
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')    # zillow file
df2 = df[(df.estimated_value<=1000000) | (df.lastSaleAmount<=1000000)]
print(df2[['estimated_value', 'yearBuilt', 'priorSaleAmount']].tail(5))
print(df2.estimated_value.shape, df2.yearBuilt.shape, df2.priorSaleAmount.shape)
print(df2.estimated_value.mean(), df2.estimated_value.median())
print(df2.yearBuilt.mean(), df2.yearBuilt.median())
print(df2.priorSaleAmount.mean(), df2.priorSaleAmount.median())
plt.show(sn.boxplot(df2.estimated_value))
```

```
       estimated_value  yearBuilt  priorSaleAmount
14993           694981     1909.0         535000.0
14994           784955     1925.0         142000.0
14995          1080081     1922.0              NaN
14996           807306     1890.0         557500.0
14999          1421401     2001.0         925000.0
(14430,) (14430,) (14430,)
567918.3354123355 507172.0
1928.025157668584l 1925.0
226394.9821511144 203000.0
```

estimated_value or priorSalesAmount

1) **Exercise deliverable:** Construct an example screenshot of using pandas df functions in Python code to remove outliers in either the single_family_home_values.csv or your data project csv. If you are using the csv above, then choose another column attribute than priorSalesAmount. (1)Be sure to run a preliminary boxplot. (2) Remove the outliers in a new dataframe. (3) Run a new boxplot after the outliers removal. Save the PyCharm screenshot for use later in this tutorial.

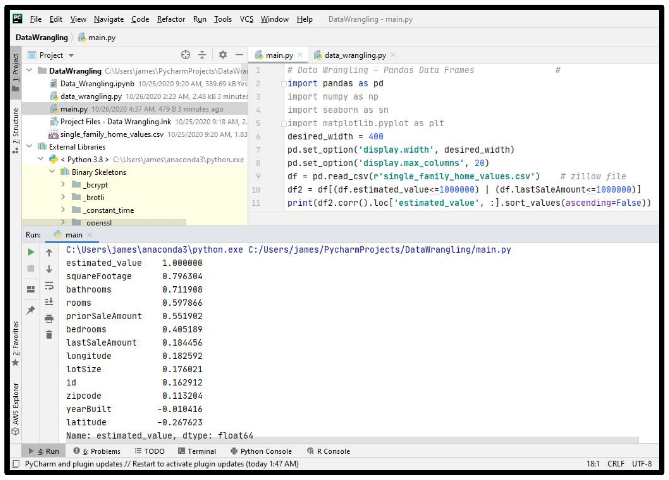
Simple Statistics in Python:

1) In addition to the pandas df function df.describe() reporting mean, standard deviation, minimum, maximum, percentiles (e.g., 25%, 50%, and 75%) metrics, the df.corr() function will provide simple correlation between all non-string column attributes. Review the following Python code:

```
# Data Wrangling --- df2.corr()for one column vs. : (i.e., all columns    #
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')     # zillow file
df2 = df[(df.estimated_value<=1000000) | (df.lastSaleAmount<=1000000)]
print(df2.corr().loc['estimated_value', :].sort_values(ascending=False))

# Data Wrangling --- df2.corr()between all columns                          #
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')     # zillow file
df2 = df[(df.estimated_value<=1000000) | (df.lastSaleAmount<=1000000)]
print(df2.corr())
```

Note that in the first df2.corr() example above, the correlation is between estimated_value and all the other columns. The correlation results are sorted descending. The following screenshot is of the first df2.corr() correlation code example.


File Edit View Navigate Code Refactor Run Tools VCS Window Help DataWrangling - main.py
DataWrangling main.py
Project
main.py data_wrangling.py
DataWrangling C:\Users\james\PycharmProjects\DataWra
Data_Wrangling.ipynb 10/25/2020 9:20 AM, 389.69 kB Yes
data_wrangling.py 10/26/2020 2:23 AM, 2.48 kB 3 minutes
main.py 10/26/2020 4:37 AM, 479 B 3 minutes ago
Project Files - Data Wrangling.lnk 10/25/2020 9:18 AM, 2.
single_family_home_values.csv 10/25/2020 9:20 AM, 1.83
External Libraries
< Python 3.8 > C:\Users\james\anaconda3\python.exe
Binary Skeletons
_bcrypt
_brotli
_constant_time
openssl
Run: main
4: Run 6: Problems TODO Terminal Python Console R Console
PyCharm and plugin updates // Restart to activate plugin updates (today 1:47 AM)
18:1 CRLF UTF-8

```
# Data Wrangling - Pandas Data Frames          #
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')    # zillow file
df2 = df[(df.estimated_value<=1000000) | (df.lastSaleAmount<=1000000)]
print(df2.corr().loc['estimated_value', :].sort_values(ascending=False))
```

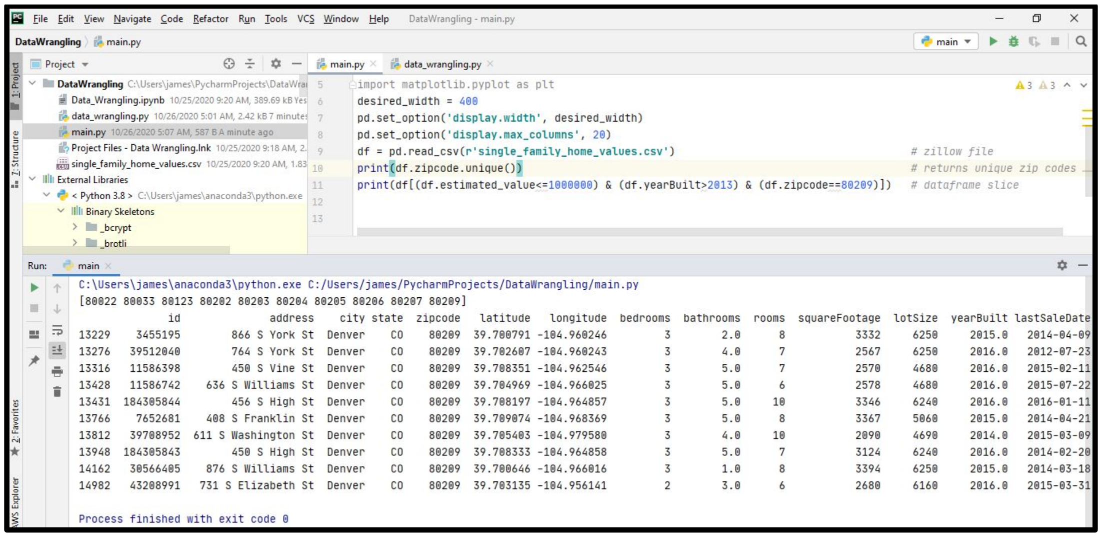
```
C:\Users\james\anaconda3\python.exe C:/Users/james/PycharmProjects/DataWrangling/main.py
estimated_value    1.000000
squareFootage      0.796304
bathrooms          0.711908
rooms              0.597866
priorSaleAmount    0.551902
bedrooms           0.405189
lastSaleAmount     0.184456
longitude          0.182592
lotSize            0.176021
id                 0.162912
zipcode            0.113204
yearBuilt         -0.010416
latitude          -0.267623
Name: estimated_value, dtype: float64
```

Exercise deliverable: Construct an example screenshot of using pandas df.corr() function in Python code to display the correlation between a column attribute and all other columns. Sort the correlation coefficients in decreasing order. Use either the single_family_home_values.csv or your data project csv. . Save the PyCharm screenshot for use later in this tutorial.

Slicing a DataFrame:

Pandas df functions allow OLAP slicing. Review the following Python code:

```
# Data Wrangling - Pandas Data Frames                    #
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')
print(df.zipcode.unique())
print(df[(df.estimated_value<=1000000) & (df.yearBuilt>2013) & (df.zipcode==80209)])
```



```
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')                              # zillow file
print(df.zipcode.unique())                                                      # returns unique zip codes
print(df[(df.estimated_value<=1000000) & (df.yearBuilt>2013) & (df.zipcode==80209)])   # dataframe slice
```

```
C:\Users\james\anaconda3\python.exe C:/Users/james/PycharmProjects/DataWrangling/main.py
[80022 80033 80123 80202 80203 80204 80205 80206 80207 80209]
              id              address    city state  zipcode   latitude   longitude  bedrooms  bathrooms  rooms  squareFootage  lotSize  yearBuilt lastSaleDate
13229    3455195         866 S York St  Denver    CO    80209  39.700791 -104.960246         3        2.0      8           3332     6250     2015.0   2014-04-09
13276   39512040         764 S York St  Denver    CO    80209  39.702607 -104.960243         3        4.0      7           2567     6250     2016.0   2012-07-23
13316   11586398         450 S Vine St  Denver    CO    80209  39.708351 -104.962546         3        5.0      7           2570     4680     2016.0   2015-02-11
13428   11586742     636 S Williams St  Denver    CO    80209  39.704969 -104.966025         3        5.0      6           2578     4680     2016.0   2015-07-22
13431  184305844         456 S High St  Denver    CO    80209  39.708197 -104.964857         3        5.0     10           3346     6240     2016.0   2016-01-11
13766    7652681     408 S Franklin St  Denver    CO    80209  39.709074 -104.968369         3        5.0      8           3367     5060     2015.0   2014-04-21
13812   39708952   611 S Washington St  Denver    CO    80209  39.705403 -104.979580         3        4.0     10           2090     4690     2014.0   2015-03-09
13948  184305843         450 S High St  Denver    CO    80209  39.708333 -104.964858         3        5.0      7           3124     6240     2016.0   2014-02-20
14162   30566405     876 S Williams St  Denver    CO    80209  39.700646 -104.966016         3        1.0      8           3394     6250     2015.0   2014-03-18
14982   43208991    731 S Elizabeth St  Denver    CO    80209  39.703135 -104.956141         2        3.0      6           2680     6160     2016.0   2015-03-31

Process finished with exit code 0
```

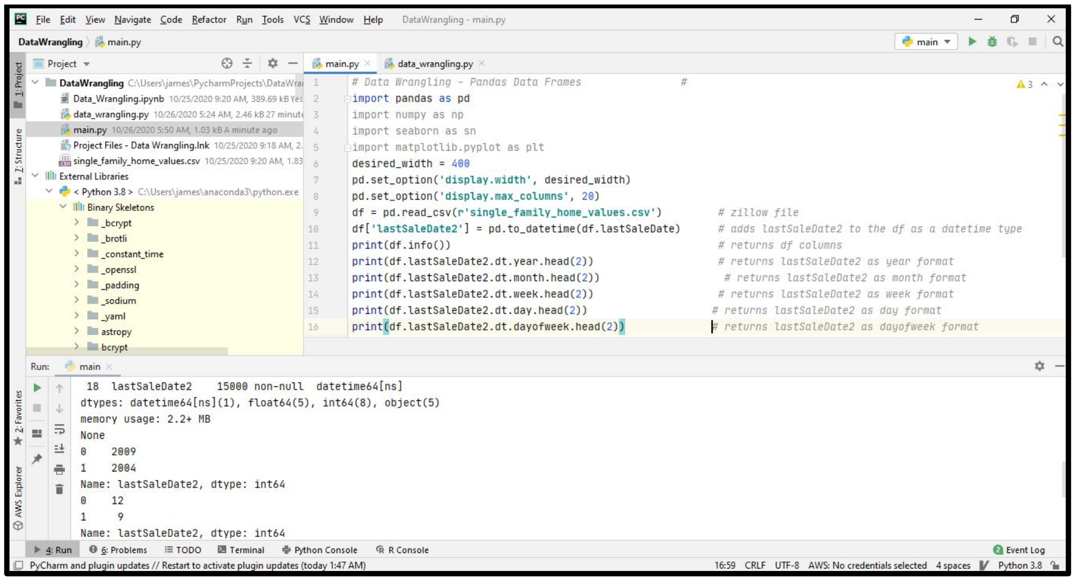
Exercise deliverable: Construct an example screenshot of using pandas df functions in Python code to slice a dataframe by three or more column attributes. Use either the single_family_home_values.csv or your data project csv. If you choose the csv file above, do not use the exact same attributes. The zip code may be reused. Save the PyCharm screenshot for use later in this tutorial.

Converting a Column, GroupBy(), and Merge():

Pandas df functions allow adding or dropping a column in the dataframe, grouping columns by column attributes, and merging dataframes. The following series of Python code represents examples each:

1) ***Converting a Column:*** In PyCharm, the csv file column attribute lastSaleDate is a string object. Converting the column attribute to a date format type has advantages in data mining and OLAP slices. The following Python code and PyCharm screenshot illustrates how to add a converted column (i.e., lastSaleDate2) to a new df2.

```
# Data Wrangling - Pandas Data Frames                    #
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')      # zillow file
df['lastSaleDate2'] = pd.to_datetime(df.lastSaleDate)  # adds to df as datetime type
print(df.info())                                        # returns df columns
print(df.lastSaleDate2.dt.year.head(2))         # returns lastSaleDate2 as year format
print(df.lastSaleDate2.dt.month.head(2))        # returns lastSaleDate2 as month format
print(df.lastSaleDate2.dt.week.head(2))         # returns lastSaleDate2 as week format
print(df.lastSaleDate2.dt.day.head(2))          # returns lastSaleDate2 as day format
print(df.lastSaleDate2.dt.dayofweek.head(2))    # returns as dayofweek format
```

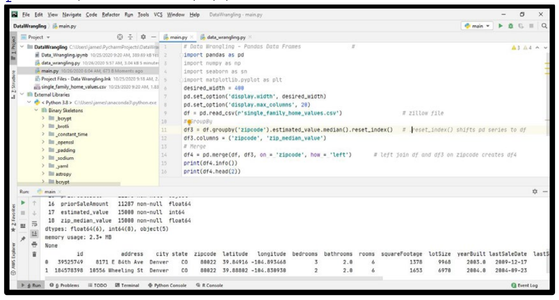


```
# Data Wrangling - Pandas Data Frames                    #
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')          # zillow file
df['lastSaleDate2'] = pd.to_datetime(df.lastSaleDate)       # adds lastSaleDate2 to the df as a datetime type
print(df.info())                                            # returns df columns
print(df.lastSaleDate2.dt.year.head(2))                     # returns lastSaleDate2 as year format
print(df.lastSaleDate2.dt.month.head(2))                     # returns lastSaleDate2 as month format
print(df.lastSaleDate2.dt.week.head(2))                      # returns lastSaleDate2 as week format
print(df.lastSaleDate2.dt.day.head(2))                      # returns lastSaleDate2 as day format
print(df.lastSaleDate2.dt.dayofweek.head(2))                # returns lastSaleDate2 as dayofweek format
```

```
 18  lastSaleDate2     15000 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(5), int64(8), object(5)
memory usage: 2.2+ MB
None
0    2009
1    2004
Name: lastSaleDate2, dtype: int64
0    12
1     9
Name: lastSaleDate2, dtype: int64
```

2) ***GroupBy & Merge:*** The pandas df function groupby() allows grouping and aggregate scalar values. In the cvs file, we can average the expected_value by zipcode into a new df3. The new df3 can then be merged with the old df to create a new df4. The Python code to perform this task and a screenshot of the code executed in PyCharm follows:

```
Data Wrangling - Pandas Data Frames                #
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')                # zillow file
# GroupBy
df3 = df.groupby('zipcode').estimated_value.median().reset_index()  # .reset_index()
#shifts pd series to df
df3.columns = ('zipcode', 'zip_median_value')
# Merge
df4 = pd.merge(df, df3, on = 'zipcode', how = 'left')        # left join df and df3 on
#zipcode creates df4
print(df4.info())
print(df4.head(2))
```



```
# Data Wrangling - Pandas Data Frames                #
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
desired_width = 400
pd.set_option('display.width', desired_width)
pd.set_option('display.max_columns', 20)
df = pd.read_csv(r'single_family_home_values.csv')                  # zillow file
#GroupBy
df3 = df.groupby('zipcode').estimated_value.median().reset_index()   # .reset_index() shifts pd series to df
df3.columns = ('zipcode', 'zip_median_value')
# Merge
df4 = pd.merge(df, df3, on = 'zipcode', how = 'left')       # left join df and df3 on zipcode creates df4
print(df4.info())
print(df4.head(2))
```

```
 16  priorSaleAmount   11287 non-null  float64
 17  estimated_value   15000 non-null  int64
 18  zip_median_value  15000 non-null  float64
dtypes: float64(6), int64(8), object(5)
memory usage: 2.3+ MB
None
          id           address    city state  zipcode  latitude   longitude  bedrooms  bathrooms  rooms  squareFootage  lotSize  yearBuilt lastSaleDate  lastS
0   39525749    8171 E 84th Ave  Denver    CO    80022  39.84916 -104.893468         3        2.0      6           1378     9968     2003.0   2009-12-17
1  184578398  10556 Wheeling St  Denver    CO    80022  39.88802 -104.830930         2        2.0      6           1653     6970     2004.0   2004-09-23
```

3) ***Exercise deliverable:*** Construct an example screenshot of using pandas df functions in Python code for a df column conversion, groupby, and/or merge. Use either the single_family_home_values.csv or your data project csv. If you choose the csv file above, do not use the exact same attributes. The zip code may be reused. Save the PyCharm screenshot for use later in this tutorial.

Deliverables from Exercises Above:

Take the noted deliverables (i.e., PyCharm screenshots) from each exercise above (i.e., total of 4 screenshots), insert the screenshots into a MS Word document as you label each screenshot deliverable from the corresponding exercise, save the document as one PDF, and submit the PDF to the Tutorial Seven Canvas Assignment uplink having the following qualities:

a. Your screenshot for each of the two exercises above is included.
b. Each screenshot is labeled appropriately for each exercise.
   i. Reading a CSV file into a Pandas DataFrame …………. 3 screenshots
   ii. Dealing with missing data and outliers ………………… 2 screenshots
   iii. Simple Statistics in Python …………………………………… 1 screenshot
   iv. Slicing a DataFrame ……………………………………………… 1 screenshot
   v. Converting a Column, GroupBy(), and Merge() ……… 1 screenshot
   Total PyCharm screenshots to submit …………………… 8 screenshots
c. All screenshots are legible output for each PyCharm exercise deliverable.