Milestone 2

July 4, 2023

Cleaning/Formatting Flat File Source

[57]:
```
# Import libraries
import pandas as pd
```

[58]:
```
data = pd.read_csv('Sale_Prices_City.csv')
```

[59]:
```
data.shape
```

[59]:
```
(3728, 150)
```

[60]:
```
data.head()
```

[60]:
```
   Unnamed: 0  RegionID   RegionName   StateName  SizeRank   2008-03  \
0           0      6181     New York    New York         1       NaN
1           1     12447  Los Angeles  California         2  507600.0
2           2     39051      Houston       Texas         3  138400.0
3           3     17426      Chicago    Illinois         4  325100.0
4           4      6915  San Antonio       Texas         5  130900.0

    2008-04   2008-05   2008-06   2008-07  …   2019-06   2019-07   2019-08  \
0       NaN       NaN       NaN       NaN  …  563200.0  570500.0  572800.0
1  489600.0  463000.0  453100.0  438100.0  …  706800.0  711800.0  717300.0
2  135500.0  132200.0  131000.0  133400.0  …  209700.0  207400.0  207600.0
3  314800.0  286900.0  274600.0  268500.0  …  271500.0  266500.0  264900.0
4  131300.0  131200.0  131500.0  131600.0  …  197100.0  198700.0  200200.0

    2019-09   2019-10   2019-11   2019-12   2020-01   2020-02   2020-03
0  569900.0  560800.0  571500.0  575100.0  571700.0  568300.0  573600.0
1  714100.0  711900.0  718400.0  727100.0  738200.0  760200.0       NaN
2  207000.0  211400.0  211500.0  217700.0  219200.0  223800.0       NaN
3  265000.0  264100.0  264300.0  270000.0  281400.0  302900.0  309200.0
4  200800.0  203400.0  203800.0  205400.0  205400.0  208300.0       NaN

[5 rows x 150 columns]
```

Step 1: Remove Unnamed Column

```
[61]: """
      In the data, there is a column that is unnamed. This column will not be needed␣
        ↪for analysis.
      The code below uses the drop function and indexes the first column to be␣
        ↪dropped.
      """
      data = data.drop(columns=data.columns[0])
```

```
[62]: data.head()
```

```
[62]:    RegionID   RegionName   StateName  SizeRank   2008-03   2008-04   2008-05  \
      0      6181     New York    New York         1       NaN       NaN       NaN
      1     12447  Los Angeles  California         2  507600.0  489600.0  463000.0
      2     39051      Houston       Texas         3  138400.0  135500.0  132200.0
      3     17426      Chicago    Illinois         4  325100.0  314800.0  286900.0
      4      6915  San Antonio       Texas         5  130900.0  131300.0  131200.0

          2008-06   2008-07   2008-08  …   2019-06   2019-07   2019-08   2019-09  \
      0       NaN       NaN       NaN  …  563200.0  570500.0  572800.0  569900.0
      1  453100.0  438100.0  423200.0  …  706800.0  711800.0  717300.0  714100.0
      2  131000.0  133400.0  135400.0  …  209700.0  207400.0  207600.0  207000.0
      3  274600.0  268500.0  264400.0  …  271500.0  266500.0  264900.0  265000.0
      4  131500.0  131600.0  132300.0  …  197100.0  198700.0  200200.0  200800.0

          2019-10   2019-11   2019-12   2020-01   2020-02   2020-03
      0  560800.0  571500.0  575100.0  571700.0  568300.0  573600.0
      1  711900.0  718400.0  727100.0  738200.0  760200.0       NaN
      2  211400.0  211500.0  217700.0  219200.0  223800.0       NaN
      3  264100.0  264300.0  270000.0  281400.0  302900.0  309200.0
      4  203400.0  203800.0  205400.0  205400.0  208300.0       NaN

      [5 rows x 149 columns]
```

Step 2: Remove RegionID

```
[63]: """
      In the data, there is a column named RedionID that will provide an additional␣
        ↪identifier for the data.
      However, this column is only apart of this data set so it will not serve a␣
        ↪purpose for the other sources.
      The code below uses the drop function and indexes the first column to be␣
        ↪dropped.
      """
      data = data.drop(columns=data.columns[0])
```

```
[64]: data.head()
```

```
[64]:      RegionName   StateName  SizeRank   2008-03   2008-04   2008-05   2008-06  \
      0     New York    New York         1       NaN       NaN       NaN       NaN
      1  Los Angeles  California         2  507600.0  489600.0  463000.0  453100.0
      2      Houston       Texas         3  138400.0  135500.0  132200.0  131000.0
      3      Chicago    Illinois         4  325100.0  314800.0  286900.0  274600.0
      4  San Antonio       Texas         5  130900.0  131300.0  131200.0  131500.0

          2008-07   2008-08   2008-09  …   2019-06   2019-07   2019-08   2019-09  \
      0       NaN       NaN       NaN  …  563200.0  570500.0  572800.0  569900.0
      1  438100.0  423200.0  407800.0  …  706800.0  711800.0  717300.0  714100.0
      2  133400.0  135400.0  138000.0  …  209700.0  207400.0  207600.0  207000.0
      3  268500.0  264400.0  267100.0  …  271500.0  266500.0  264900.0  265000.0
      4  131600.0  132300.0  131600.0  …  197100.0  198700.0  200200.0  200800.0

          2019-10   2019-11   2019-12   2020-01   2020-02   2020-03
      0  560800.0  571500.0  575100.0  571700.0  568300.0  573600.0
      1  711900.0  718400.0  727100.0  738200.0  760200.0       NaN
      2  211400.0  211500.0  217700.0  219200.0  223800.0       NaN
      3  264100.0  264300.0  270000.0  281400.0  302900.0  309200.0
      4  203400.0  203800.0  205400.0  205400.0  208300.0       NaN

      [5 rows x 148 columns]
```

Step 3: Rename RegionName to CityName

```
[65]: """
      When looking at the table the column "RegionName" indicates cities.
      If there is a join needed by city, this column will need to be renamed.
      The code below used the rename function to select "RegionName" and renames it␣
       ↪to "CityName"
      """
      data = data.rename(columns={'RegionName':'CityName'})
```

```
[66]: data.head()
```

```
[66]:        CityName   StateName  SizeRank   2008-03   2008-04   2008-05   2008-06  \
      0     New York    New York         1       NaN       NaN       NaN       NaN
      1  Los Angeles  California         2  507600.0  489600.0  463000.0  453100.0
      2      Houston       Texas         3  138400.0  135500.0  132200.0  131000.0
      3      Chicago    Illinois         4  325100.0  314800.0  286900.0  274600.0
      4  San Antonio       Texas         5  130900.0  131300.0  131200.0  131500.0

          2008-07   2008-08   2008-09  …   2019-06   2019-07   2019-08   2019-09  \
      0       NaN       NaN       NaN  …  563200.0  570500.0  572800.0  569900.0
      1  438100.0  423200.0  407800.0  …  706800.0  711800.0  717300.0  714100.0
      2  133400.0  135400.0  138000.0  …  209700.0  207400.0  207600.0  207000.0
      3  268500.0  264400.0  267100.0  …  271500.0  266500.0  264900.0  265000.0
```

```
4  131600.0  132300.0  131600.0  …  197100.0  198700.0  200200.0  200800.0

    2019-10   2019-11   2019-12   2020-01   2020-02   2020-03
0  560800.0  571500.0  575100.0  571700.0  568300.0  573600.0
1  711900.0  718400.0  727100.0  738200.0  760200.0       NaN
2  211400.0  211500.0  217700.0  219200.0  223800.0       NaN
3  264100.0  264300.0  270000.0  281400.0  302900.0  309200.0
4  203400.0  203800.0  205400.0  205400.0  208300.0       NaN

[5 rows x 148 columns]
```

Step 4: Fill NaN Values with 0

[67]:
```
"""
The data containes some NaN values and if calculations are done, this could␣
↪create some issues.
The code below uses fillna function to replace all NaN values with 0
"""
data = data.fillna(0)
```

[68]:
```
data.head()
```

[68]:
```
      CityName   StateName  SizeRank   2008-03   2008-04   2008-05   2008-06  \
0     New York    New York         1       0.0       0.0       0.0       0.0
1  Los Angeles  California         2  507600.0  489600.0  463000.0  453100.0
2      Houston       Texas         3  138400.0  135500.0  132200.0  131000.0
3      Chicago    Illinois         4  325100.0  314800.0  286900.0  274600.0
4  San Antonio       Texas         5  130900.0  131300.0  131200.0  131500.0

    2008-07   2008-08   2008-09  …   2019-06   2019-07   2019-08   2019-09  \
0       0.0       0.0       0.0  …  563200.0  570500.0  572800.0  569900.0
1  438100.0  423200.0  407800.0  …  706800.0  711800.0  717300.0  714100.0
2  133400.0  135400.0  138000.0  …  209700.0  207400.0  207600.0  207000.0
3  268500.0  264400.0  267100.0  …  271500.0  266500.0  264900.0  265000.0
4  131600.0  132300.0  131600.0  …  197100.0  198700.0  200200.0  200800.0

    2019-10   2019-11   2019-12   2020-01   2020-02   2020-03
0  560800.0  571500.0  575100.0  571700.0  568300.0  573600.0
1  711900.0  718400.0  727100.0  738200.0  760200.0       0.0
2  211400.0  211500.0  217700.0  219200.0  223800.0       0.0
3  264100.0  264300.0  270000.0  281400.0  302900.0  309200.0
4  203400.0  203800.0  205400.0  205400.0  208300.0       0.0

[5 rows x 148 columns]
```

Step 5: Remove the column SizeRank

```
[69]: """
      The column SizeRank provides information that will not be used later on.
      To keep the data clean, this column can be removed.
      The code below, selects the column size rank and removes it.
      """
      data.drop('SizeRank', axis=1, inplace=True)
```

```
[70]: data.head()
```

```
[70]:       CityName   StateName   2008-03   2008-04   2008-05   2008-06   2008-07  \
      0     New York    New York       0.0       0.0       0.0       0.0       0.0
      1  Los Angeles  California  507600.0  489600.0  463000.0  453100.0  438100.0
      2      Houston       Texas  138400.0  135500.0  132200.0  131000.0  133400.0
      3      Chicago    Illinois  325100.0  314800.0  286900.0  274600.0  268500.0
      4  San Antonio       Texas  130900.0  131300.0  131200.0  131500.0  131600.0

          2008-08   2008-09   2008-10  …   2019-06   2019-07   2019-08   2019-09  \
      0       0.0       0.0       0.0  …  563200.0  570500.0  572800.0  569900.0
      1  423200.0  407800.0  396300.0  …  706800.0  711800.0  717300.0  714100.0
      2  135400.0  138000.0  136400.0  …  209700.0  207400.0  207600.0  207000.0
      3  264400.0  267100.0  268400.0  …  271500.0  266500.0  264900.0  265000.0
      4  132300.0  131600.0  131800.0  …  197100.0  198700.0  200200.0  200800.0

          2019-10   2019-11   2019-12   2020-01   2020-02   2020-03
      0  560800.0  571500.0  575100.0  571700.0  568300.0  573600.0
      1  711900.0  718400.0  727100.0  738200.0  760200.0       0.0
      2  211400.0  211500.0  217700.0  219200.0  223800.0       0.0
      3  264100.0  264300.0  270000.0  281400.0  302900.0  309200.0
      4  203400.0  203800.0  205400.0  205400.0  208300.0       0.0

      [5 rows x 147 columns]
```

Step 6: Format all numbers to be comma separated

```
[71]: """
      The data is easily read however, when there are hundreds of thousands, there␣
        ↪can be a difficulty differentiating
      the amount.
      The code below changes all the columns after the second column to be formated␣
        ↪with a comma at each appropriate place.
      """
      data.loc[:,2:] = data.iloc[:,2:].applymap(lambda x: '{:,}'.format(x))
```

```
/var/folders/sr/xvmzsbj91c91yq0f0qnq71xh0000gn/T/ipykernel_53036/3161285015.py:4
: FutureWarning: Slicing a positional slice with .loc is not supported, and will
raise TypeError in a future version.  Use .loc with labels or .iloc with
positions instead.
  data.loc[:,2:] = data.iloc[:,2:].applymap(lambda x: '{:,}'.format(x))
```

```
[72]: data.head()
```

```
[72]:       CityName   StateName    2008-03    2008-04    2008-05    2008-06  \
      0     New York    New York        0.0        0.0        0.0        0.0
      1  Los Angeles  California  507,600.0  489,600.0  463,000.0  453,100.0
      2      Houston       Texas  138,400.0  135,500.0  132,200.0  131,000.0
      3      Chicago    Illinois  325,100.0  314,800.0  286,900.0  274,600.0
      4  San Antonio       Texas  130,900.0  131,300.0  131,200.0  131,500.0

           2008-07    2008-08    2008-09    2008-10  …    2019-06    2019-07  \
      0        0.0        0.0        0.0        0.0  …  563,200.0  570,500.0
      1  438,100.0  423,200.0  407,800.0  396,300.0  …  706,800.0  711,800.0
      2  133,400.0  135,400.0  138,000.0  136,400.0  …  209,700.0  207,400.0
      3  268,500.0  264,400.0  267,100.0  268,400.0  …  271,500.0  266,500.0
      4  131,600.0  132,300.0  131,600.0  131,800.0  …  197,100.0  198,700.0

           2019-08    2019-09    2019-10    2019-11    2019-12    2020-01  \
      0  572,800.0  569,900.0  560,800.0  571,500.0  575,100.0  571,700.0
      1  717,300.0  714,100.0  711,900.0  718,400.0  727,100.0  738,200.0
      2  207,600.0  207,000.0  211,400.0  211,500.0  217,700.0  219,200.0
      3  264,900.0  265,000.0  264,100.0  264,300.0  270,000.0  281,400.0
      4  200,200.0  200,800.0  203,400.0  203,800.0  205,400.0  205,400.0

           2020-02    2020-03
      0  568,300.0  573,600.0
      1  760,200.0        0.0
      2  223,800.0        0.0
      3  302,900.0  309,200.0
      4  208,300.0        0.0

      [5 rows x 147 columns]
```