Capstone AAA Projet Martin George mgeorgevienna@gmail.com

Reading Excel file in to

In [ ]:

```

```

In [4]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
data = pd.read_csv('member_sample.csv', index_col = 0)
```

In [5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21344 entries, 0 to 99998
Columns: 112 entries, Individual Key to Was Towed To AAR Referral
dtypes: float64(35), object(77)
memory usage: 18.4+ MB
```

In [6]:
```
data.head()
```

Out[6]:

	Individual Key	Household Key	Member Flag	City	State - Grouped	ZIP5	ZIP9	FSV CMSI Flag	FSV Credit Card Flag	FSV Deposit Program Flag	...	SC Vehicle Manufacturer Name	SC Vehicle Model Name	Fa I
0	10000003.0	10462590.0	Y	NEW HAVEN	CT	6511.0	65111349.0	N	N	N	...	NaN	NaN	
1	52211550.0	4500791.0	Y	WEST WARWICK	RI	2893.0	28933850.0	N	Y	N	...	TOYOTA	CAMRY	AS WREC SEF
2	52211550.0	4500791.0	Y	WEST WARWICK	RI	2893.0	28933850.0	N	Y	N	...	TOYOTA	CAMRY	Wr S
3	52211550.0	4500791.0	Y	WEST WARWICK	RI	2893.0	28933850.0	N	Y	N	...	TOYOTA	CAMRY	AS WREC SEF
4	52211550.0	4500791.0	Y	WEST WARWICK	RI	2893.0	28933850.0	N	Y	N	...	TOYOTA	CAMRY	AS WREC SEF

5 rows × 112 columns

In [7]:
```
nulls = data.isnull().sum()
```

In [8]:
```
type(nulls)
```

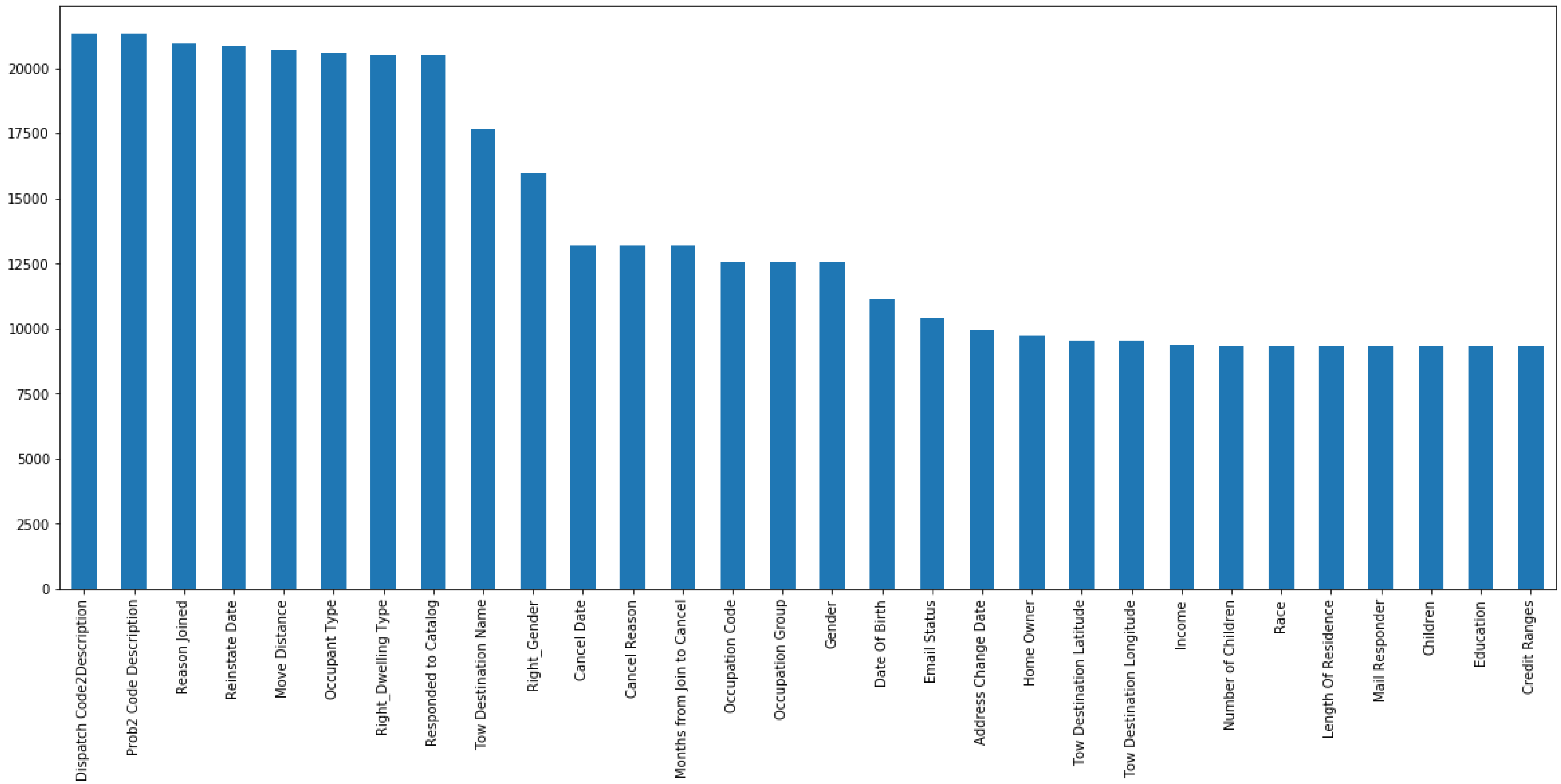
Out[8]:
```
pandas.core.series.Series
```

In [9]:
```
nulls
```

Out[9]:
```
Individual Key                  0
Household Key                   0
Member Flag                     0
City                            0
State - Grouped                 0
                             ...
Tow Destination Latitude     9531
Tow Destination Longitude    9531
Tow Destination Name        17652
Was Duplicated               7347
Was Towed To AAR Referral    7347
Length: 112, dtype: int64
```

In [10]:
```
nulls.nlargest(30).plot(kind='bar',figsize =(20,8))
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x19d75aacb48>



Loc and iloc

loc vs iloc: The loc indexer can also do boolean selection. For instance, if we are interested in finding all the rows where Age is less 30 and return just the Color and Height columns we can do the following. We can replicate this with iloc but we cannot pass it a boolean series. We must convert the boolean Series into a numpy array. loc gets rows (or columns) with particular labels from the index. iloc gets rows (or columns) at particular positions in the index (so it only takes integers).

Example

In [11]:
```
df = pd.DataFrame({'Age': [30, 20, 22, 40, 32, 28, 39],
                   'Color': ['Blue', 'Green', 'Red', 'White', 'Gray', 'Black',
                             'Red'],
                   'Food': ['Steak', 'Lamb', 'Mango', 'Apple', 'Cheese',
                            'Melon', 'Beans'],
                   'Height': [165, 70, 120, 80, 180, 172, 150],
                   'Score': [4.6, 8.3, 9.0, 3.3, 1.8, 9.5, 2.2],
                   'State': ['NY', 'TX', 'FL', 'AL', 'AK', 'TX', 'TX']
                   },
                  index=['Jane', 'Nick', 'Aaron', 'Penelope', 'Dean',
                         'Christina', 'Cornelia'])
print(df)
df.info()

print("\n -- loc -- \n")
print(df.loc[df['Age'] < 30, ['Color', 'Height']])

print("\n -- iloc -- \n")
print(df.iloc[(df['Age'] < 30).values, [1, 4]])
```

```
           Age  Color    Food  Height  Score State
Jane        30   Blue   Steak     165    4.6    NY
Nick        20  Green    Lamb      70    8.3    TX
Aaron       22    Red   Mango     120    9.0    FL
Penelope    40  White   Apple      80    3.3    AL
Dean        32   Gray  Cheese     180    1.8    AK
Christina   28  Black   Melon     172    9.5    TX
Cornelia    39    Red   Beans     150    2.2    TX
<class 'pandas.core.frame.DataFrame'>
Index: 7 entries, Jane to Cornelia
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Age     7 non-null      int64  
 1   Color   7 non-null      object 
 2   Food    7 non-null      object 
 3   Height  7 non-null      int64  
 4   Score   7 non-null      float64
 5   State   7 non-null      object 
dtypes: float64(1), int64(2), object(3)
memory usage: 392.0+ bytes

 -- loc --

           Color  Height
Nick       Green      70
Aaron        Red     120
Christina  Black     172

 -- iloc --

           Color  Score
Nick       Green    8.3
Aaron        Red    9.0
Christina  Black    9.5
```

In [12]:
```
data['Reason Joined'].head()
```

Out[12]:
```
0    NaN
1    NaN
2    NaN
3    NaN
4    NaN
Name: Reason Joined, dtype: object
```

In [13]:
```
data['Reason Joined'].value_counts()
```

Out[13]:
```
U                       168
Dependable Services     127
5                        45
Family Plan Avail        19
Nation Wide Rd Srv        7
Gift Membership           5
Free Membership           4
Club Reputation           3
3                         3
Other                     1
Prior Family Exp          1
Recommend/Referral        1
Convenient Offices        1
Variety of Services       1
7                         1
Direct Mail               1
Name: Reason Joined, dtype: int64
```

In [14]:
```
data.shape
```

Out[14]:
```
(21344, 112)
```

In [15]:
```
data['Reason Joined'].fillna('unknow',inplace=True)
```

In [16]:
```
data['Reason Joined'].value_counts()
```

Out[16]:
```
unknow                  20956
U                         168
Dependable Services       127
5                          45
Family Plan Avail          19
Nation Wide Rd Srv          7
Gift Membership             5
Free Membership             4
3                           3
Club Reputation             3
Other                       1
7                           1
Variety of Services         1
Convenient Offices          1
Recommend/Referral          1
Prior Family Exp            1
Direct Mail                 1
Name: Reason Joined, dtype: int64
```

'Reason Joined' has 20956 null values out of 21344 and not a good data for analysis

In [17]:
```
dependable_service_group = data.loc[data['Reason Joined'] == 'Dependable Services']
```

In [18]:
```
everyone_else = data.loc[data['Reason Joined'] != 'Dependable Services']
```

In [19]:
```
dependable_service_group.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 127 entries, 633 to 89877
Columns: 112 entries, Individual Key to Was Towed To AAR Referral
dtypes: float64(35), object(77)
memory usage: 112.1+ KB
```

In [20]:
```
everyone_else.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21217 entries, 0 to 99998
Columns: 112 entries, Individual Key to Was Towed To AAR Referral
dtypes: float64(35), object(77)
memory usage: 18.3+ MB
```

In [21]:
```
data['Dispatch Code1 Description'].value_counts()
```

```
Out[21]: Member Requests Battery Service              3804
         Flat Tire w/spare                            2410
         No Crank - Jump Start                        1192
         Key Locked In Passenger Compartment          1036
         Other Required Tow (describe)                 831
         Convenience/Member Concern Tow                744
         Tire Issue Requires Tow To Shop               426
         Engine Runs Poorly                            408
         Brake System Failure                          275
         Out Of Gasoline                               263
         Need Air In Tire                              233
         Engine Stalled While Driving                  225
         Engine Overheat                               223
         Transmission/Clutch Failure                   219
         Member Requests Tow                           189
         Known Starter Problem                         187
         Collision/Police Tow                          151
         No Crank - Bat Svc (non-AAA Bat)              147
         Known Alternator Problem                      133
         Extrication - Probable GO                     132
         Other Runs Won't Move Problem (describe)      112
         Axle/Driveshaft/Suspension Failure            105
         No Crank - Bat Svc (AAA Bat)                   79
         Ignition Key Won't Turn In Switch              68
         Key Locked In Trunk w/Trunk Release            59
         Leaking Fluids                                 47
         Light Service Redispatch As Tow                47
         Lock Issue Requires Tow To Dealer              37
         Undercar Component Dragging                    31
         Crank No Start                                 26
         Extrication - Probable TOW                     22
         Lost/Damaged Vehicle Key                       21
         Flat Tire w/o spare                            20
         Other Lockout Problem (describe)               19
         Frozen Door Lock                               15
         Car Alarm Issue                                10
         Key Broken In Ignition Switch                   8
         Multiple Flat Tires                             6
         Other Service (describe)                        5
         Taxi/Shuttle Service                            4
         Hood/Door Won't Close/Latch                     4
         Other Crank No Start Problem (describe)         4
         Parking Brake Won't Release                     4
```

```
Windshield Damage                        3
Lost/Damaged  Club  Key                  2
Other Locksmith Problem (describe)       2
Key Locked In Trunk - No Trunk Release   2
Vehicle Mis-Fueled                       1
Key Broken In Door - Ignition Key        1
Other Runs Poorly Problem (describe)     1
Vehicle Fire                             1
EV Out Of Charge Station Range           1
Home Lockout                             1
Name: Dispatch Code1 Description, dtype: int64
```

In [22]:
```
data.iloc[17:21]
```

Out[22]:

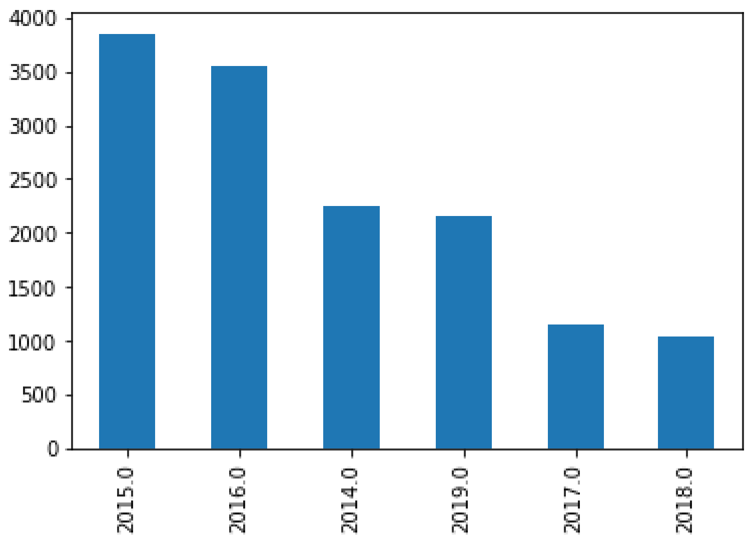
	Individual Key	Household Key	Member Flag	City	State - Grouped	ZIP5	ZIP9	FSV CMSI Flag	FSV Credit Card Flag	FSV Deposit Program Flag	...	SC Vehicle Manufacturer Name	SC Vehicle Model Name	F
17	2766868.0	11622991.0	Y	WARWICK	RI	2889.0	28892920.0	N	N	N	...	SUBARU	BRZ	AA RI SEI
18	2766868.0	11622991.0	Y	WARWICK	RI	2889.0	28892920.0	N	N	N	...	SUBARU	BRZ	AA RI SEI
19	2766868.0	11622991.0	Y	WARWICK	RI	2889.0	28892920.0	N	N	N	...	HYUNDAI	TUCSON	A C
20	13746947.0	579810.0	Y	CENTRAL FALLS	RI	2863.0	28631322.0	N	N	N	...	INFINITI	QX56	K SEI CE

4 rows × 112 columns

In [23]:
```
data['Date'] = pd.to_datetime(data['SC Date'])
```

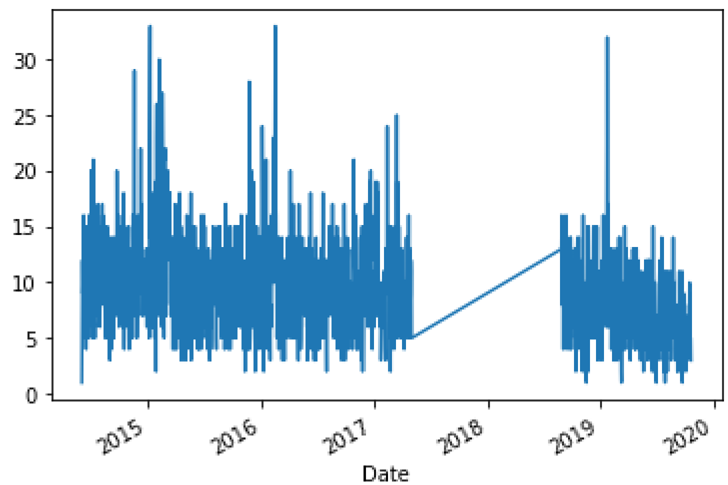
In [24]:
```
data['Date'].dt.year.value_counts().plot(kind = 'bar')
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x19d762586c8>



In [25]:
```
data.groupby('Date').size().plot()
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x19d7647ba88>



In [26]:
```
data['Reason Joined'].head()
```

Out[26]:
```
0    unknow
1    unknow
2    unknow
3    unknow
4    unknow
Name: Reason Joined, dtype: object
```

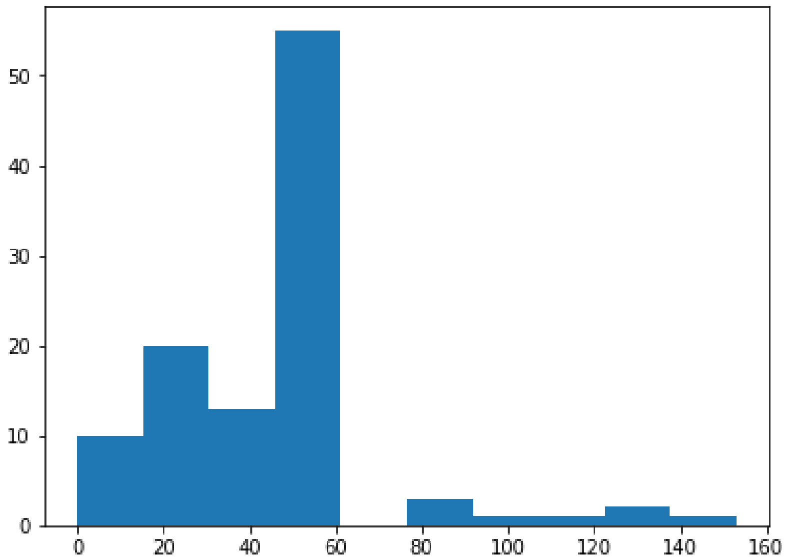
In [27]:
```
data['Reason Joined'].value_counts()
```

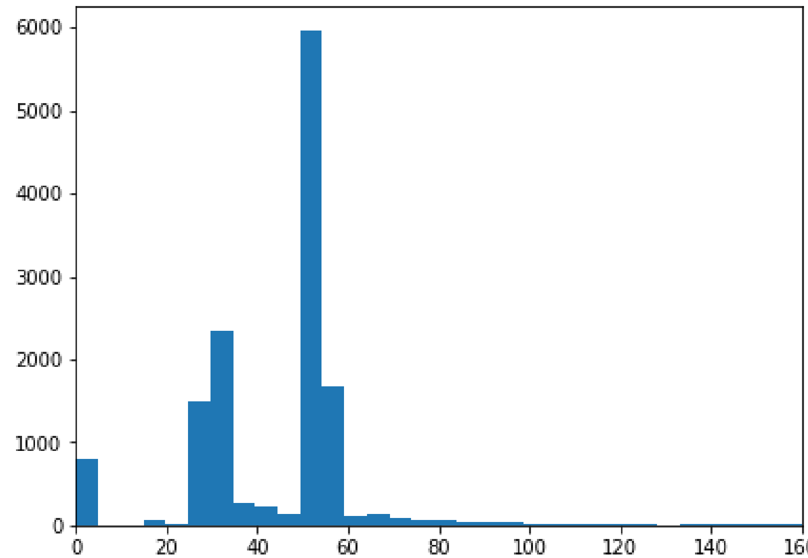
Out[27]:
```
unknow                   20956
U                          168
Dependable Services        127
5                           45
Family Plan Avail           19
Nation Wide Rd Srv           7
Gift Membership              5
Free Membership              4
3                            3
Club Reputation              3
Other                        1
7                            1
Variety of Services          1
Convenient Offices           1
Recommend/Referral           1
Prior Family Exp             1
Direct Mail                  1
Name: Reason Joined, dtype: int64
```

In [28]:
```
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (15, 5))
ax[0].hist(dependable_service_group['Total Cost']);
ax[1].hist(everyone_else['Total Cost'], bins = 2000);
ax[1].set_xlim(0,160)
```

```
C:\Users\unodc\anaconda3\lib\site-packages\numpy\lib\histograms.py:839: RuntimeWarning: invalid value encount
ered in greater_equal
  keep = (tmp_a >= first_edge)
C:\Users\unodc\anaconda3\lib\site-packages\numpy\lib\histograms.py:840: RuntimeWarning: invalid value encount
ered in less_equal
  keep &= (tmp_a <= last_edge)
```

Out[28]: (0, 160)





In [29]:
```
data['Income'].value_counts()
```

Out[29]:
```
100-149,999      2577
90-99,999        2400
70-79,999        1000
50-59,999         888
40-49,999         771
10-19,999         688
175 - 199,999     600
30-39,999         553
60-69,999         541
150 - 174,999     438
20-29,999         425
200 - 249,999     424
250K+             397
Under 10K         226
80-89,999          15
Name: Income, dtype: int64
```

In [30]:
```
data['Income'].fillna('unknown',inplace=True)
```

In [31]:
```
data['Income'].value_counts()
```

Out[31]:
```
unknown          9401
100-149,999      2577
90-99,999        2400
70-79,999        1000
50-59,999         888
40-49,999         771
10-19,999         688
175 - 199,999     600
30-39,999         553
60-69,999         541
150 - 174,999     438
20-29,999         425
200 - 249,999     424
250K+             397
Under 10K         226
80-89,999          15
Name: Income, dtype: int64
```

In [32]:
```
data['Home Owner'].value_counts()
```

Out[32]:
```
Home Owner              11121
Renter                    491
Probable Renter            10
Probable Home Owner         7
Name: Home Owner, dtype: int64
```

In [33]:
```
def yes_noer(x):
    if x== 'Y':
        return 1
    elif x == 'N':
        return 0
    else:
        return np.nan
```

'FSV CMSI Flag' is Engagement with AAA Financial Service

In [34]:
```
data['FSV CMSI Flag'] = data['FSV CMSI Flag'].apply(yes_noer)
```

In [35]:
```
data['FSV CMSI Flag'].value_counts()
```

Out[35]:
```
0    20393
1      951
Name: FSV CMSI Flag, dtype: int64
```

In [36]:
```
data.groupby(['Home Owner'])['FSV CMSI Flag'].mean()
```

Out[36]:
```
Home Owner
Home Owner              0.058988
Probable Home Owner     0.000000
Probable Renter         0.000000
Renter                  0.097760
Name: FSV CMSI Flag, dtype: float64
```

In [37]:
```
home_owners = data.loc[data['Home Owner'] == 'Home Owner']
```

In [38]:
```
home_owners.head()
```

Out[38]:

	Individual Key	Household Key	Member Flag	City	State - Grouped	ZIP5	ZIP9	FSV CMSI Flag	FSV Credit Card Flag	FSV Deposit Program Flag	...	SC Vehicle Model Name	SVC Facility Name	S Faci Ty
1	52211550.0	4500791.0	Y	WEST WARWICK	RI	2893.0	28933850.0	0	Y	N	...	CAMRY	ASTRO WRECKER SERVICE	independ rep
2	52211550.0	4500791.0	Y	WEST WARWICK	RI	2893.0	28933850.0	0	Y	N	...	CAMRY	Astro Wrecker Service	independ rep
3	52211550.0	4500791.0	Y	WEST WARWICK	RI	2893.0	28933850.0	0	Y	N	...	CAMRY	ASTRO WRECKER SERVICE	independ rep
4	52211550.0	4500791.0	Y	WEST WARWICK	RI	2893.0	28933850.0	0	Y	N	...	CAMRY	ASTRO WRECKER SERVICE	independ rep
5	52211550.0	4500791.0	Y	WEST WARWICK	RI	2893.0	28933850.0	0	Y	N	...	CAMRY	AAA SNE RI LIGHT SERVICE	mol batt serv

5 rows × 113 columns

In [39]:
```
home_owners['Home Owner'].count()
```

Out[39]:
```
11121
```

In [40]:
```
home_owners['Home Owner'].head()
```

Out[40]:
```
1    Home Owner
2    Home Owner
3    Home Owner
4    Home Owner
5    Home Owner
Name: Home Owner, dtype: object
```

In [41]:
```
data[['Tow Destination Latitude','Tow Destination Longitude']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21344 entries, 0 to 99998
Data columns (total 2 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Tow Destination Latitude   11813 non-null  float64
 1   Tow Destination Longitude  11813 non-null  float64
dtypes: float64(2)
memory usage: 500.2 KB
```

In [130]:
```
lat_long_0 = data[['Tow Destination Latitude','Tow Destination Longitude']].dropna().iloc[:5000]
lat_long = data[['Tow Destination Latitude','Tow Destination Longitude','Total Cost']].dropna().iloc[:5000]
```

In [106]:
```
lat_long.head()
```

Out[106]:

	Tow Destination Latitude	Tow Destination Longitude	Total Cost
1	41.0	-71.0	32.5
2	0.0	0.0	30.0
3	0.0	0.0	32.5
4	0.0	0.0	30.0
5	0.0	0.0	53.0

In [68]:
```
!pip install folium
```

```
Requirement already satisfied: folium in c:\users\unodc\anaconda3\lib\site-packages (0.11.0)
Requirement already satisfied: numpy in c:\users\unodc\anaconda3\lib\site-packages (from folium) (1.18.1)
Requirement already satisfied: branca>=0.3.0 in c:\users\unodc\anaconda3\lib\site-packages (from folium) (0.
4.1)
Requirement already satisfied: requests in c:\users\unodc\anaconda3\lib\site-packages (from folium) (2.22.0)
Requirement already satisfied: jinja2>=2.9 in c:\users\unodc\anaconda3\lib\site-packages (from folium) (2.11.
1)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\unodc\anaconda3\lib\site-packages (from requ
ests->folium) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\unodc\anaconda3\lib\site-packages (from request
s->folium) (2019.11.28)
Requirement already satisfied: idna<2.9,>=2.5 in c:\users\unodc\anaconda3\lib\site-packages (from requests->f
olium) (2.8)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in c:\users\unodc\anaconda3\lib\site-p
ackages (from requests->folium) (1.25.8)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\unodc\anaconda3\lib\site-packages (from jinja2>=
2.9->folium) (1.1.1)
```

In [69]:
```
import folium
```

In [120]:
```
loc_1 = lat_long_0.iloc[0]
```

In [121]:
```
loc_1
```

Out[121]:
```
Tow Destination Latitude     41.0
Tow Destination Longitude   -71.0
Name: 1, dtype: float64
```

In [82]:
```
#loc = data[['Tow Destination Latitude','Tow Destination Longitude','Total Cost']].dropna()
#loc.head()
#loc.shape
```

Out[82]:
```
(11798, 3)
```

In [ ]:
```

```

In [122]:
```
m = folium.Map(location = loc_1)
```

In [ ]:

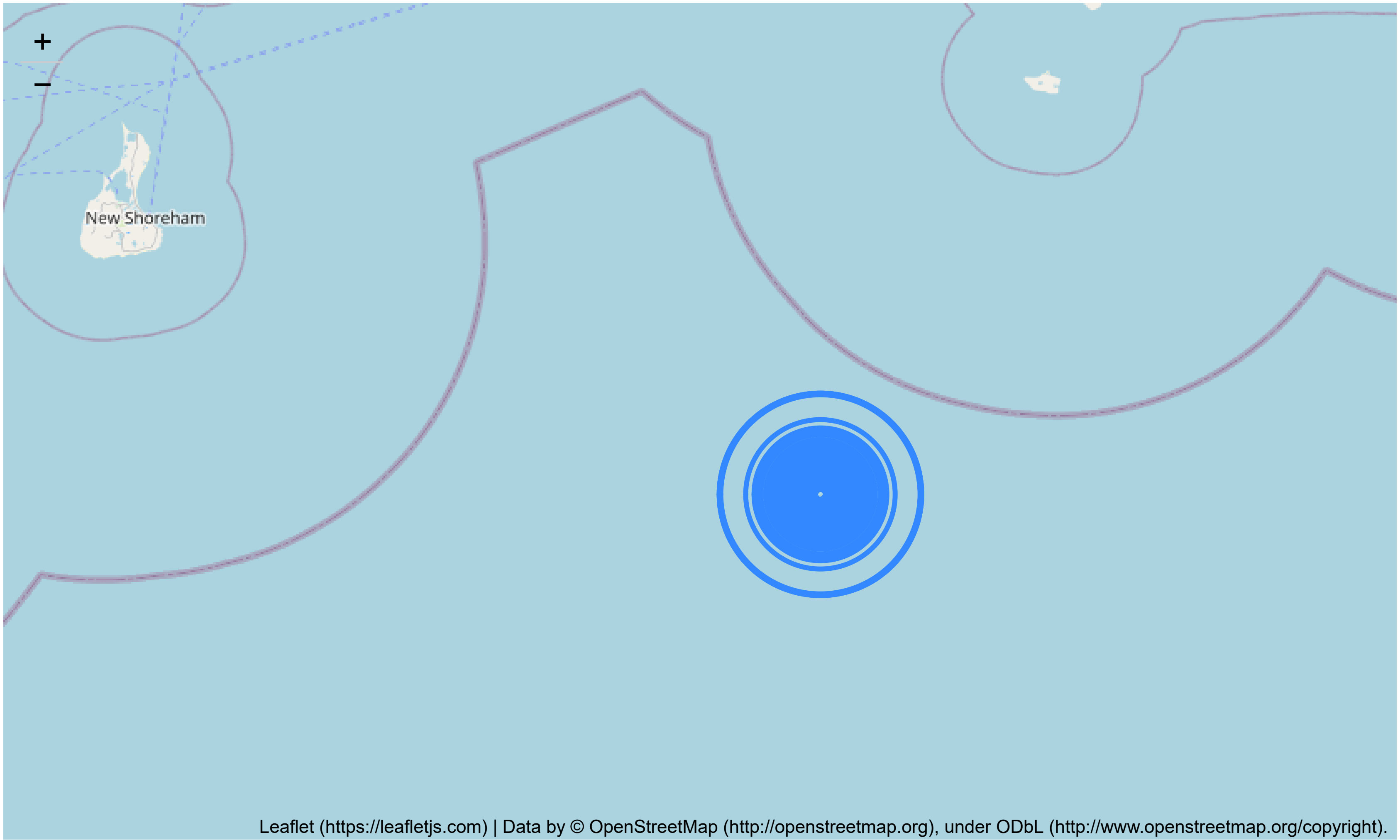
In [131]:
```
def add_marker(x):
    folium.CircleMarker(location = [x[0],x[1]], radius = x[2]/5).add_to(m)
```

In [132]:
```
lat_long.apply(add_marker, axis = 1)
```

Out[132]:
```
1        None
2        None
3        None
4        None
5        None
         ...
16344    None
16345    None
16348    None
16349    None
16354    None
Length: 5000, dtype: object
```

In [133]:
```
m
```

Out[133]:



K-Means Clustering

K-Means Clustering is an unsupervised machine learning algorithm. In contrast to traditional supervised machine learning algorithms, K-Means attempts to classify data without having first been trained with labeled data. Once the algorithm has been run and the groups are defined, any new data can be easily assigned to the most relevant group. The real world applications of K-Means include: customer profiling market segmentation computer vision search engines astronomy

Example Coding

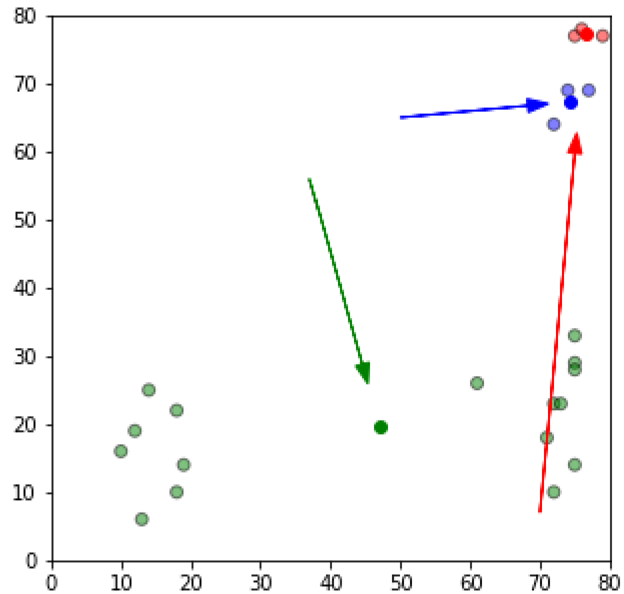
In [51]:
```
kmdf = pd.DataFrame ({
    'x' : [12,10,18,18,19,13,14,75,75,72,61,72,75,73,75,71,74,79,72,75,76,77],
    'y' : [19,16,10,22,14,6,25,29,33,10,26,23,28,23,14,18,69,77,64,77,78,69]
})

np.random.seed(40)

k = 3
centroids = {
i+1: [np.random.randint(0,80),np.random.randint(0,80)] for i in range(k)
}

print(centroids)

def assignment(kmdf,centroids):
    for i in centroids.keys():
        kmdf['distance_from_{}'.format(i)] = (
            np.sqrt(
                (kmdf['x'] - centroids[i][0]) ** 2 + (kmdf['y'] - centroids[i][0]) ** 2
            )
        )
    centroid_distance_cols = ['distance_from_{}'.format(i) for i in centroids.keys()]
    print(centroid_distance_cols)
    kmdf['closest'] =  kmdf.loc[:,centroid_distance_cols].idxmin(axis=1)
    #print(kmdf.head())
    #print(kmdf['closest'])
    kmdf['closest'] = kmdf['closest'].map(lambda x: int(x.lstrip('distance_from_')))
    kmdf['color'] = kmdf['closest'].map(lambda x: colmap[x])
    return kmdf
print(centroids)
colmap = { 1:'r', 2:'g', 3: 'b', 4:'y'}
kmdf =assignment(kmdf,centroids)
print(kmdf)

import copy
old_centroids = copy.deepcopy(centroids)

print(old_centroids)

def update(k):
    print("before update")
    print(k)
```

```
    for i in k.keys():
        k[i][0] = np.mean(kmdf[kmdf['closest'] == i]['x'])
        k[i][1] = np.mean(kmdf[kmdf['closest'] == i]['y'])
    print("after update")
    print(k)
    return k
#centroids = update(centroids)
#print(centroids)

while True:
    closest_centroids = kmdf['closest'].copy(deep=True)
    centroids = update(centroids)
    print(closest_centroids)
    kmdf = assignment(kmdf,centroids)
    if closest_centroids.equals(kmdf['closest']):
        break


fig = plt.figure(figsize=(5,5))
ax = plt.axes()
plt.scatter(kmdf['x'],kmdf['y'],color =kmdf['color'], alpha = 0.5, edgecolor='k')

for i in centroids.keys():
    plt.scatter(*centroids[i],color=colmap[i])
plt.xlim(0,80)
plt.ylim(0,80)

for i in old_centroids.keys():
    old_x = old_centroids[i][0]
    old_y = old_centroids[i][1]
    dx = ( centroids[i][0] - old_centroids[i][0]) * 0.75
    dy = ( centroids[i][1] - old_centroids[i][1]) * 0.75
    ax.arrow(old_x,old_y, dx,dy, head_width = 2 , head_length = 3, fc=colmap[i], ec=colmap[i])

plt.show()
```

```
{1: [70, 7], 2: [37, 56], 3: [50, 65]}
{1: [70, 7], 2: [37, 56], 3: [50, 65]}
['distance_from_1', 'distance_from_2', 'distance_from_3']
     x   y  distance_from_1  distance_from_2  distance_from_3  closest color
0   12  19        77.233412        30.805844        49.040799        2     g
1   10  16        80.721744        34.205263        52.497619        2     g
2   18  10        79.397733        33.015148        51.224994        2     g
3   18  22        70.767224        24.207437        42.520583        2     g
4   19  14        75.742986        29.206164        47.507894        2     g
5   13   6        85.702975        39.204592        57.489129        2     g
6   14  25        71.840100        25.942244        43.829214        2     g
7   75  29        41.303753        38.832976        32.649655        3     b
8   75  33        37.336309        38.209946        30.232433        3     b
9   72  10        60.033324        44.204072        45.650849        2     g
10  61  26        44.911023        26.400758        26.400758        2     g
11  72  23        47.042534        37.696154        34.828150        3     b
12  75  28        42.296572        39.051248        33.301652        3     b
13  73  23        47.095647        38.626416        35.468296        3     b
14  75  14        56.222771        44.418465        43.829214        3     b
15  71  18        52.009614        38.948684        38.275318        3     b
16  74  69         4.123106        48.918299        30.610456        1     r
17  79  77        11.401754        58.000000        39.623226        1     r
18  72  64         6.324555        44.204072        26.076810        1     r
19  75  77         8.602325        55.172457        36.796739        1     r
20  76  78        10.000000        56.586217        38.209946        1     r
21  77  69         7.071068        51.224994        33.015148        1     r
{1: [70, 7], 2: [37, 56], 3: [50, 65]}
before update
{1: [70, 7], 2: [37, 56], 3: [50, 65]}
after update
{1: [75.5, 72.33333333333333], 2: [26.333333333333332, 16.444444444444443], 3: [73.71428571428571, 24.0]}
0     2
1     2
2     2
3     2
4     2
5     2
6     2
7     3
8     3
9     2
10    2
11    3
```

```
12    3
13    3
14    3
15    3
16    1
17    1
18    1
19    1
20    1
21    1
Name: closest, dtype: int64
['distance_from_1', 'distance_from_2', 'distance_from_3']
before update
{1: [75.5, 72.33333333333333], 2: [26.333333333333332, 16.444444444444443], 3: [73.71428571428571, 24.0]}
after update
{1: [76.66666666666667, 77.33333333333333], 2: [40.61538461538461, 17.384615384615383], 3: [74.6666666666666
7, 48.666666666666664]}
0     2
1     2
2     2
3     2
4     2
5     2
6     2
7     3
8     3
9     2
10    2
11    2
12    3
13    2
14    2
15    2
16    3
17    1
18    3
19    1
20    1
21    3
Name: closest, dtype: int64
['distance_from_1', 'distance_from_2', 'distance_from_3']
before update
{1: [76.66666666666667, 77.33333333333333], 2: [40.61538461538461, 17.384615384615383], 3: [74.6666666666666
```

```
7, 48.666666666666664]}
after update
{1: [76.66666666666667, 77.33333333333333], 2: [47.0625, 19.75], 3: [74.33333333333333, 67.33333333333333]}
0     2
1     2
2     2
3     2
4     2
5     2
6     2
7     2
8     2
9     2
10    2
11    2
12    2
13    2
14    2
15    2
16    3
17    1
18    3
19    1
20    1
21    3
Name: closest, dtype: int64
['distance_from_1', 'distance_from_2', 'distance_from_3']
```



KMeans using sklearn :

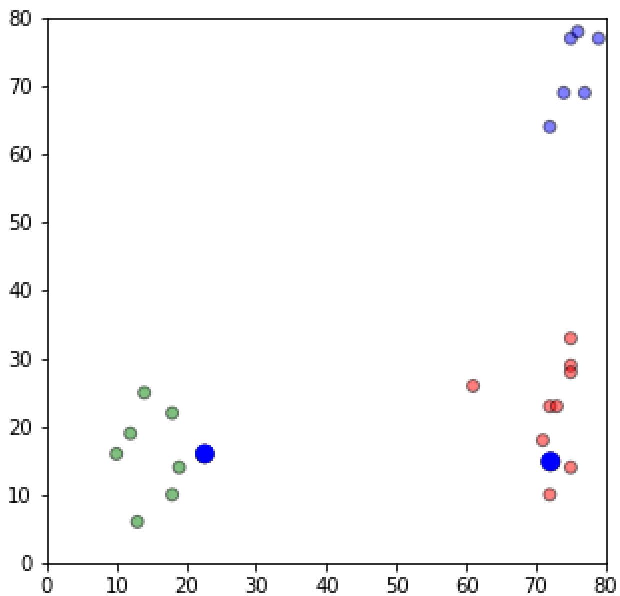
In [52]:
```
kmdf = pd.DataFrame ({
    'x' : [12,10,18,18,19,13,14,75,75,72,61,72,75,73,75,71,74,79,72,75,76,77],
    'y' : [19,16,10,22,14,6,25,29,33,10,26,23,28,23,14,18,69,77,64,77,78,69]
})

from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 3)
kmeans.fit(kmdf)
```

Out[52]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [53]:
```
labels =kmeans.predict(kmdf)
centroids = kmeans.cluster_centers_
```

In [54]:
```
print(labels)
colors = map(lambda x: colmap[x+1],labels)
colors1 = list(colors)
print(colors1)
fig = plt.figure(figsize=(5,5))
ax = plt.axes()
plt.scatter(kmdf['x'],kmdf['y'],color =colors1, alpha = 0.5, edgecolor='k')

for idx,centroid in enumerate(centroids):
    plt.scatter(*centroids ,color=colmap[idx+1])
plt.xlim(0,80)
plt.ylim(0,80)
plt.show()
```

```
[1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2]
['g', 'g', 'g', 'g', 'g', 'g', 'g', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'b', 'b', 'b', 'b', 'b',
'b']
```



In [55]:
```
#m1 = folium.Map(Location=[48.2,16.3], zoom_start =4)
#m1
```

In [56]:
```
from sklearn.cluster import KMeans
```

In [57]:
```
X = pd.get_dummies(data[['Total Cost', 'SC Vehicle Manufacturer Name']].dropna())
```

In [58]:
```
X.head()
```

Out[58]:

	Total Cost	SC Vehicle Manufacturer Name_ACURA	SC Vehicle Manufacturer Name_ALFA ROMEO	SC Vehicle Manufacturer Name_AMERICAN AUSTIN	SC Vehicle Manufacturer Name_APRILIA	SC Vehicle Manufacturer Name_AUDI	SC Vehicle Manufacturer Name_AUSTIN HEALEY	SC Vehicle Manufacturer Name_Audi	SC Vehicle Manufacturer Name_BICYCLE
1	32.5	0	0	0	0	0	0	0	0
2	30.0	0	0	0	0	0	0	0	0
3	32.5	0	0	0	0	0	0	0	0
4	30.0	0	0	0	0	0	0	0	0
5	53.0	0	0	0	0	0	0	0	0

5 rows × 87 columns

In [ ]:

In [140]:
```
kmeans = KMeans(n_clusters = 2)
```

In [141]:
```
kmeans.fit(X)
```

Out[141]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [142]:
```
X['Label'] = kmeans.labels_
```

In [147]:
```
X.head()
```

Out[147]:

	Total Cost	SC Vehicle Manufacturer Name_ACURA	SC Vehicle Manufacturer Name_ALFA ROMEO	SC Vehicle Manufacturer Name_AMERICAN AUSTIN	SC Vehicle Manufacturer Name_APRILIA	SC Vehicle Manufacturer Name_AUDI	SC Vehicle Manufacturer Name_AUSTIN HEALEY	SC Vehicle Manufacturer Name_Audi	SC Vehicle Manufacturer Name_BICYCLE
1	32.5	0	0	0	0	0	0	0	0
2	30.0	0	0	0	0	0	0	0	0
3	32.5	0	0	0	0	0	0	0	0
4	30.0	0	0	0	0	0	0	0	0
5	53.0	0	0	0	0	0	0	0	0

5 rows × 88 columns

In [144]:
```
X.groupby('Label').mean()
```

Out[144]:

Label	Total Cost	SC Vehicle Manufacturer Name_ACURA	SC Vehicle Manufacturer Name_ALFA ROMEO	SC Vehicle Manufacturer Name_AMERICAN AUSTIN	SC Vehicle Manufacturer Name_APRILIA	SC Vehicle Manufacturer Name_AUDI	SC Vehicle Manufacturer Name_AUSTIN HEALEY	SC Vehicle Manufacturer Name_Audi	SC Vel Manuf Name_
0	46.569276	0.011762	0.001004	0.000072	0.000072	0.004805	0.000072	0.000072	
1	9869.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	

2 rows × 87 columns

In [145]:
```
X.groupby('Label').mean().iloc[1].sort_values()
#X.groupby('Label').mean()
```

Out[145]:
```
SC Vehicle Manufacturer Name_JEEP                0.0
SC Vehicle Manufacturer Name_PATHFINDER          0.0
SC Vehicle Manufacturer Name_OLDSMOBILE          0.0
SC Vehicle Manufacturer Name_Nissan              0.0
SC Vehicle Manufacturer Name_NISSAN              0.0
                                               ...
SC Vehicle Manufacturer Name_FORD                0.0
SC Vehicle Manufacturer Name_FLINT               0.0
SC Vehicle Manufacturer Name_HHKHKJHKJHKJHKJ     0.0
SC Vehicle Manufacturer Name_HONDA               1.0
Total Cost                                    9869.0
Name: 1, Length: 87, dtype: float64
```

In [146]:
```
X.groupby('Label').mean().iloc[0].sort_values()
```

Out[146]:
```
SC Vehicle Manufacturer Name_nissan            0.000072
SC Vehicle Manufacturer Name_HHKHKJHKJHKJHKJ   0.000072
SC Vehicle Manufacturer Name_HINO              0.000072
SC Vehicle Manufacturer Name_HUNDAI            0.000072
SC Vehicle Manufacturer Name_UNK               0.000072
                                                 ...
SC Vehicle Manufacturer Name_CHEVROLET         0.066915
SC Vehicle Manufacturer Name_HONDA             0.107007
SC Vehicle Manufacturer Name_FORD              0.112888
SC Vehicle Manufacturer Name_TOYOTA            0.163953
Total Cost                                    46.569276
Name: 0, Length: 87, dtype: float64
```

In [149]:
```
#X.groupby('Label').mean().iloc[2].sort_values()
```

In [150]:
```
#X.groupby('Label').mean().iloc[3].sort_values()
```

In [65]:
```
data['FSV Mortgage Flag'].value_counts()
```

Out[65]:
```
N    21317
Y       27
Name: FSV Mortgage Flag, dtype: int64
```

In [185]:
```
res = data.groupby('SC Vehicle Manufacturer Name').agg({'Total Cost': ['sum','count','mean']})
```

res.reset_index(inplace = True)

In [186]:
```
res.columns
```

Out[186]:
```
MultiIndex([('Total Cost',   'sum'),
            ('Total Cost', 'count'),
            ('Total Cost',  'mean')],
           )
```

In [187]:
```
res.sort_values(by = [('Total Cost', 'sum')], ascending=False)
```

Out[187]:

	Total Cost		
	sum	count	mean
SC Vehicle Manufacturer Name			
TOYOTA	107688.66	2286	47.107900
FORD	74446.37	1574	47.297567
HONDA	73544.41	1493	49.259484
CHEVROLET	41628.97	933	44.618403
NISSAN	34854.75	776	44.915915
...	...	...	...
MECURY	29.00	1	29.000000
FLINT	29.00	1	29.000000
HINO	29.00	1	29.000000
AMERICAN AUSTIN	27.00	1	27.000000
HHKHKJHKJHKJHKJ	0.00	1	0.000000

86 rows × 3 columns

In [184]:
```
#res[:,:]
```

In [155]:
```
data['SC Vehicle Manufacturer Name'].value_counts()
```

Out[155]:
```
TOYOTA            2289
FORD              1580
HONDA             1494
CHEVROLET          935
NISSAN             780
                  ... 
PATHFINDER           1
Subaru               1
UNK                  1
APRILIA              1
AMERICAN AUSTIN      1
Name: SC Vehicle Manufacturer Name, Length: 86, dtype: int64
```

In [66]:
```
data.to_csv(r'member_sample_step_01.csv', index = False)
```