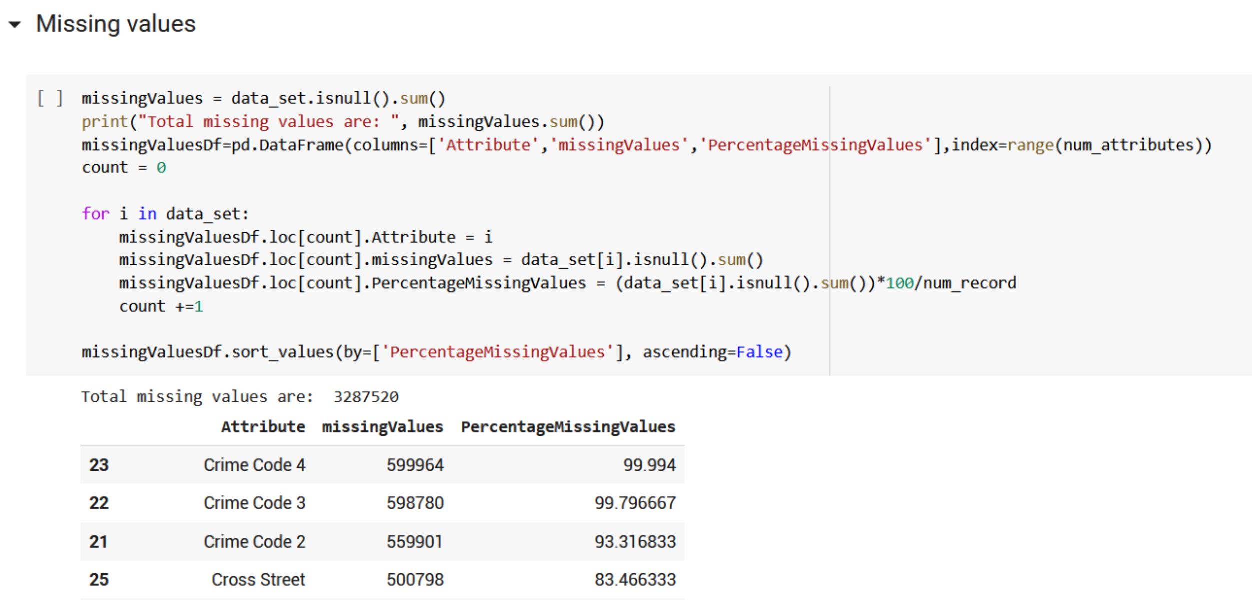
PREPROCESSING

Code 1



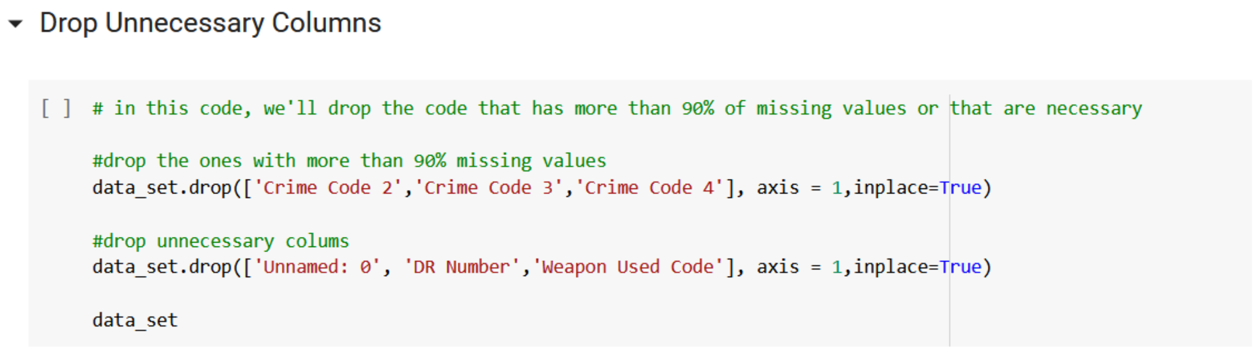
Missing values

```
[ ] missingValues = data_set.isnull().sum()
    print("Total missing values are: ", missingValues.sum())
    missingValuesDf=pd.DataFrame(columns=['Attribute','missingValues','PercentageMissingValues'],index=range(num_attributes))
    count = 0

    for i in data_set:
        missingValuesDf.loc[count].Attribute = i
        missingValuesDf.loc[count].missingValues = data_set[i].isnull().sum()
        missingValuesDf.loc[count].PercentageMissingValues = (data_set[i].isnull().sum())*100/num_record
        count +=1

    missingValuesDf.sort_values(by=['PercentageMissingValues'], ascending=False)
```

```
Total missing values are:  3287520
             Attribute  missingValues  PercentageMissingValues
23        Crime Code 4         599964                   99.994
22        Crime Code 3         598780                99.796667
21        Crime Code 2         559901                93.316833
25        Cross Street         500798                83.466333
```

In the code above, we calculated the missing values in each column (attribute) of the data and also calculated the percentage of missing values to number of values. We print this information and in the next code, we use it to remove some attributes that have more than 90% values missing.

Code 2

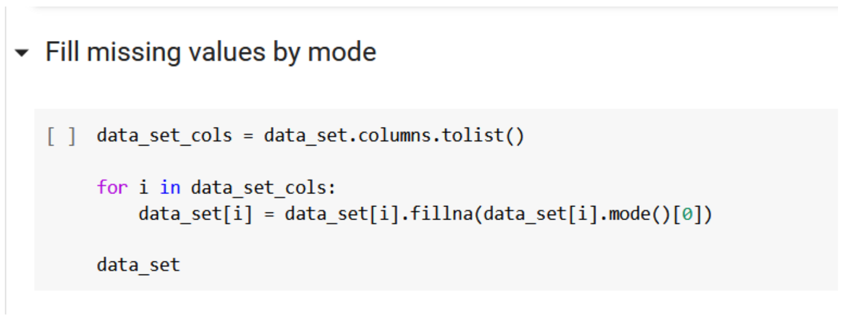


Drop Unnecessary Columns

```
[ ] # in this code, we'll drop the code that has more than 90% of missing values or that are necessary

    #drop the ones with more than 90% missing values
    data_set.drop(['Crime Code 2','Crime Code 3','Crime Code 4'], axis = 1,inplace=True)

    #drop unnecessary colums
    data_set.drop(['Unnamed: 0', 'DR Number','Weapon Used Code'], axis = 1,inplace=True)

    data_set
```

Here we remove “Crime Code 4”, “Crime Code 3”, and “Crime Code 2” in place because more than 90% of the values in these attributes are missing.

Code 3



Fill missing values by mode

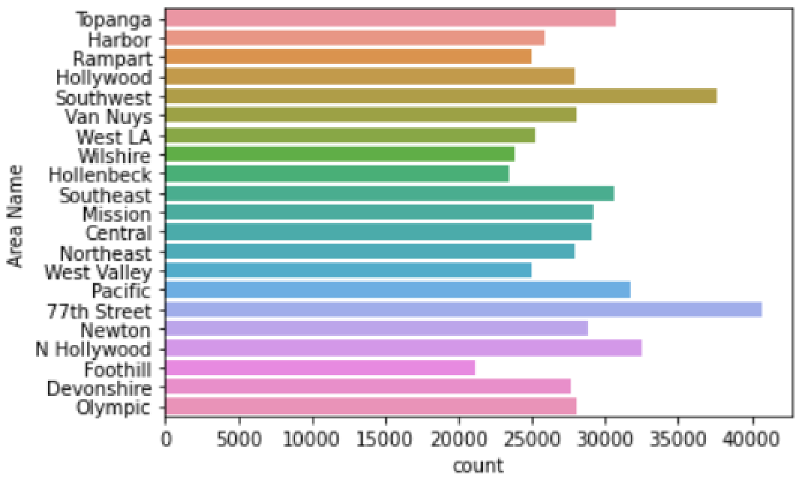
```
[ ] data_set_cols = data_set.columns.tolist()

    for i in data_set_cols:
        data_set[i] = data_set[i].fillna(data_set[i].mode()[0])

    data_set
```

Here we use the fillna() method of the DataFrame to fill missing values, and we fill them with the mode of the dataset column.

Statistics

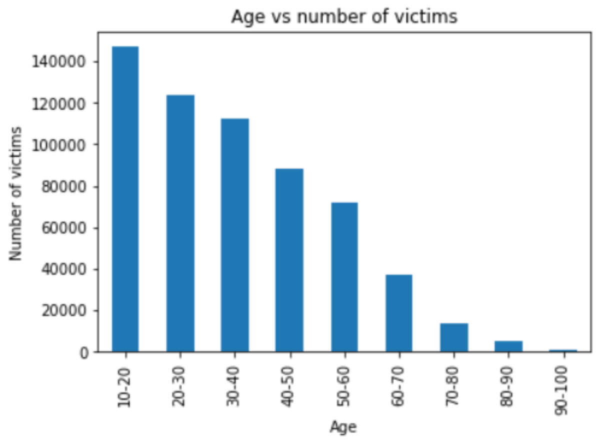
After that, we printed some statistics and summaries, such as

- Graph of number of crimes committed in each area



- Top 15 weapons used
- Crime rate for various genders
- Frequency of occurrence of victim descents

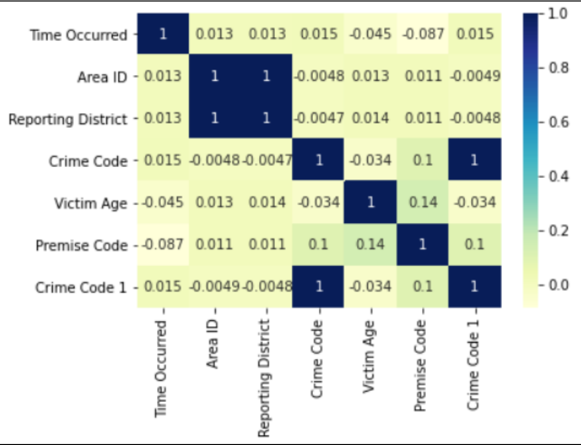
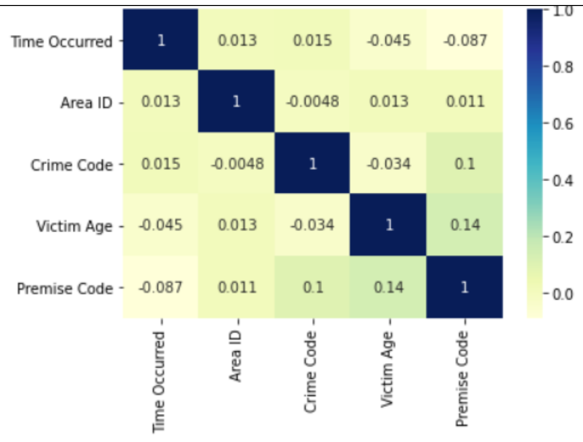
- Number of victims in various age ranges



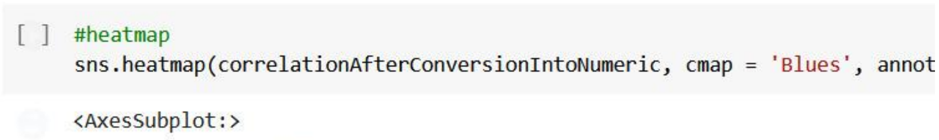
Correlation-based Cleaning

Then we calculated the mutual correlations of various attributes and found that some attributes had correlations close to 1, meaning that only one of these correlating attributes was needed for mining. Therefore, the remaining (having correlation > 0.95) were removed.

The following is a comparison of the heatmaps of the correlations between attributes before and after this removal:

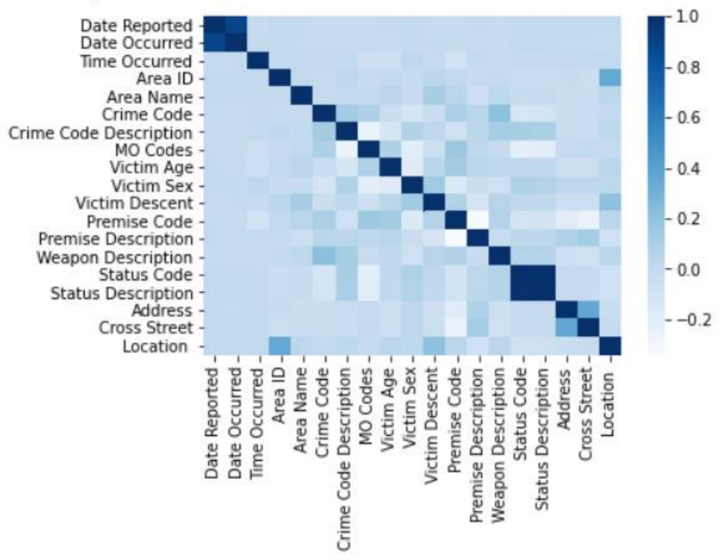
Before Removal	**After Removal**
	

Correlation after conversion into numeric



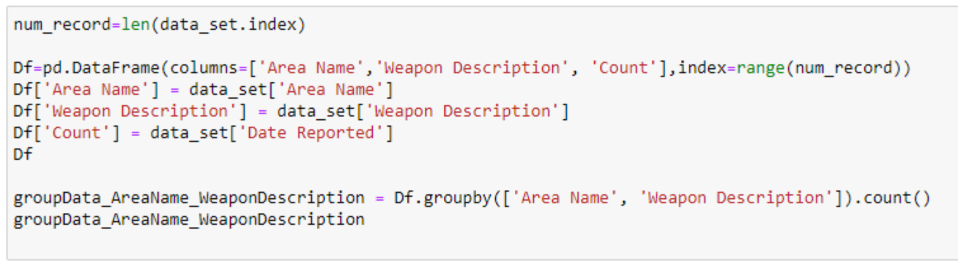
```
[ ]  #heatmap
     sns.heatmap(correlationAfterConversionIntoNumeric, cmap = 'Blues', annot
```

```
<AxesSubplot:>
```



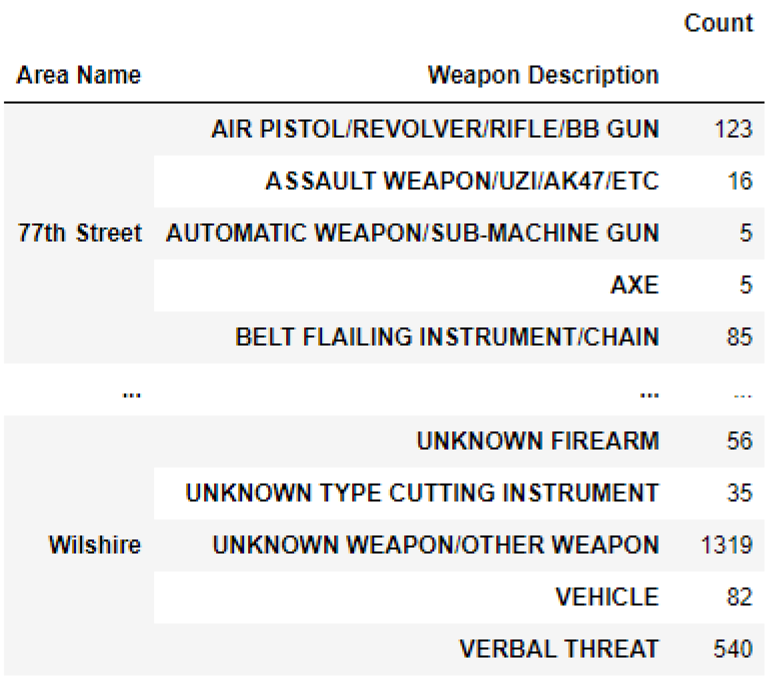
After converting the data type of all columns into numeric, we again find the correlation between them. Analysis of the graph shows that Date occurred and date reported are highly correlated with correlation > 0.8.

Weapon count by area



```
num_record=len(data_set.index)

Df=pd.DataFrame(columns=['Area Name','Weapon Description', 'Count'],index=range(num_record))
Df['Area Name'] = data_set['Area Name']
Df['Weapon Description'] = data_set['Weapon Description']
Df['Count'] = data_set['Date Reported']
Df

groupData_AreaName_WeaponDescription = Df.groupby(['Area Name', 'Weapon Description']).count()
groupData_AreaName_WeaponDescription
```

In this code, we group the data set by area name and count each type of weapon used in different areas.



Area Name	Weapon Description	Count
77th Street	AIR PISTOL/REVOLVER/RIFLE/BB GUN	123
	ASSAULT WEAPON/UZI/AK47/ETC	16
	AUTOMATIC WEAPON/SUB-MACHINE GUN	5
	AXE	5
	BELT FLAILING INSTRUMENT/CHAIN	85
...	...	...
Wilshire	UNKNOWN FIREARM	56
	UNKNOWN TYPE CUTTING INSTRUMENT	35
	UNKNOWN WEAPON/OTHER WEAPON	1319
	VEHICLE	82
	VERBAL THREAT	540

The resulted table shows that the count of different weapons was different in different areas

CLUSTERING

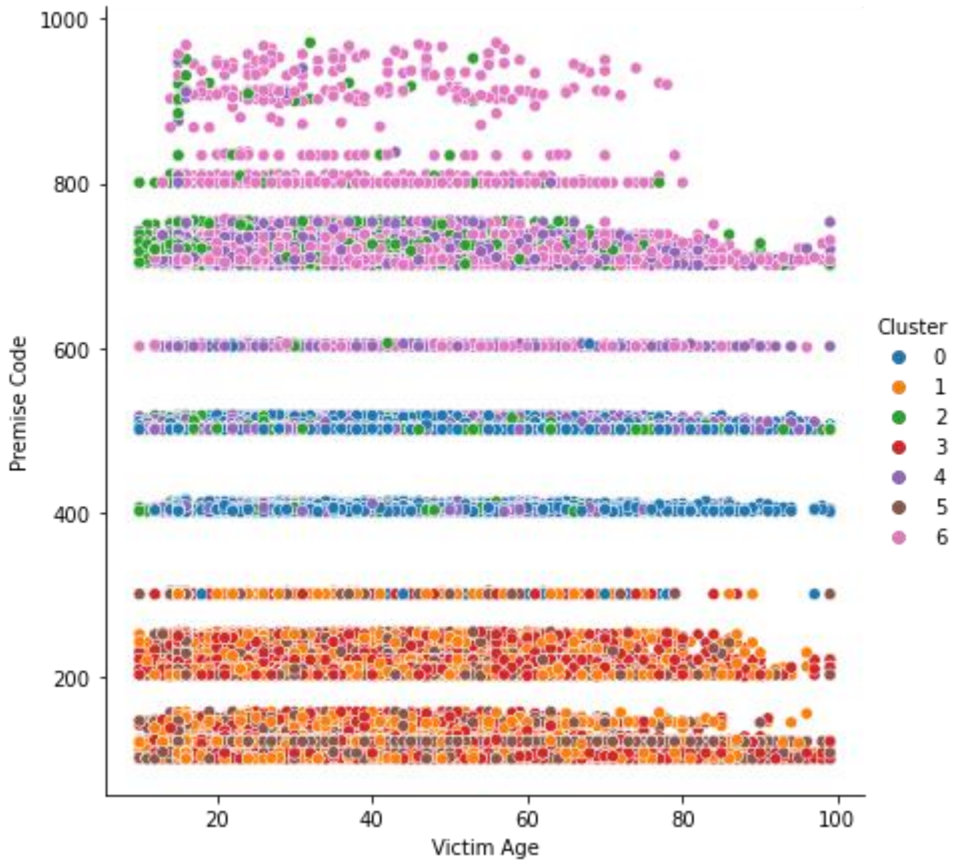
In order to see if we could get any relevant information about the dataset for the police, we clustered the data set.

There are several algorithms we came across when deciding what type of clustering to implement based on the algorithms. This included K-means clustering, Mean-Shift Clustering, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Expectation–Maximization (EM) Clustering using Gaussian Mixture Models (GMM) and Agglomerative Hierarchical Clustering.

There are several factors that made us choose K-means clustering over the rest, K-means algorithm is very fast, since the computations are very low because computing distances between points and centers is fast, it only takes order of n O(n). Although we must acknowledge that it has a slight disadvantage as well in the sense that we have to choose our own number of clusters. but with close analysis we were able to find the right number of clusters.

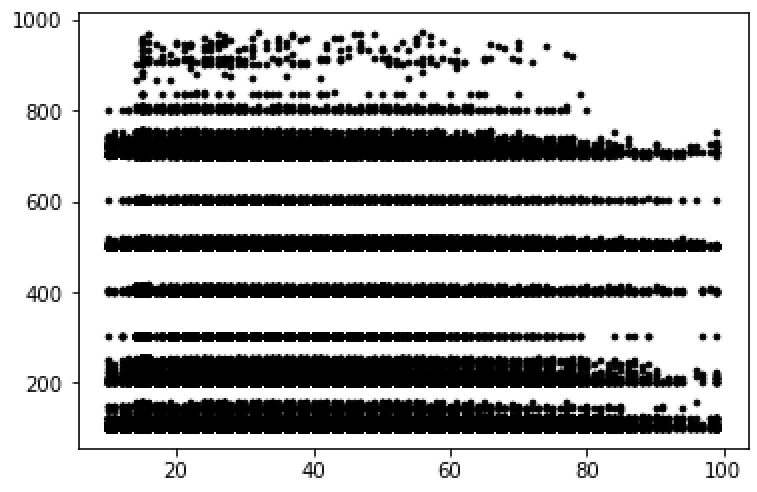
Other important factors were that it is relatively simple to use, and it scales to large data sets easily. Moreover, it guarantees convergence.

Now let's go over the clusters that we made through kmeans.

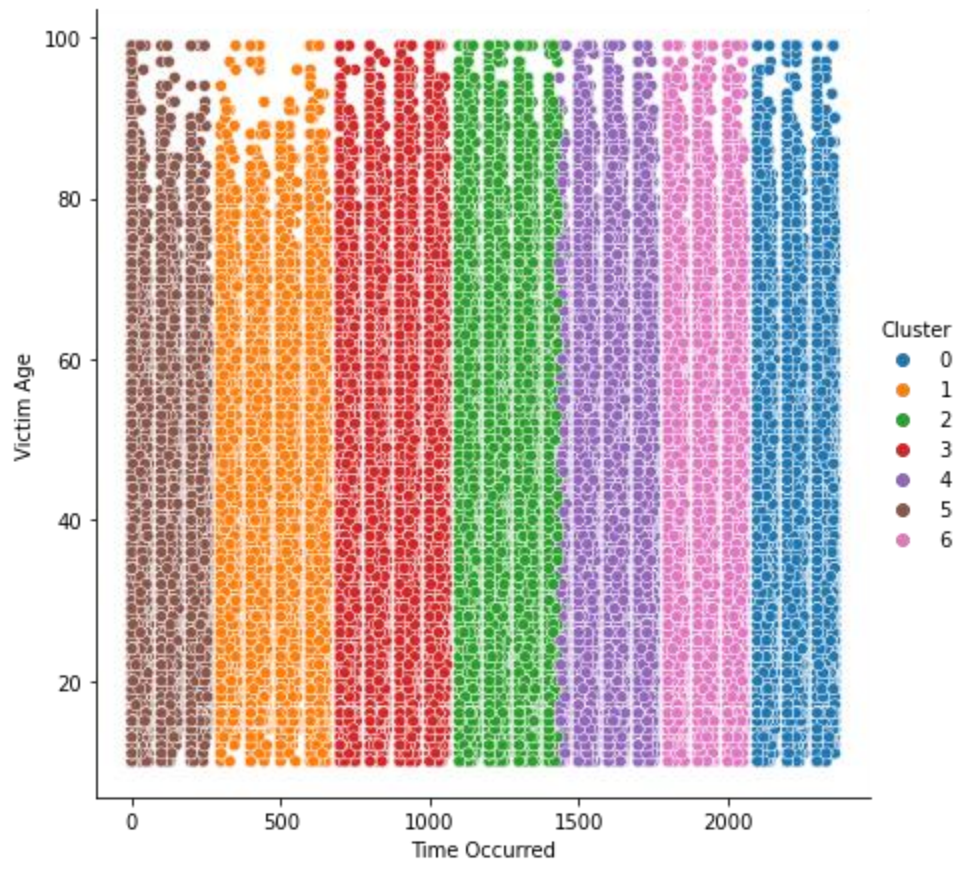


This Graph is a clustering graph of victim age and premise code. We can clearly see that some premise codes are much more distinct than others. For example, premise codes that are under 400 falls into one category and have victims of all ages. However, the cluster with premise code greater than 700 is way different and mostly includes people with ages less than 60. Thus, this cluster is safer for old ages as the crime rates are very low for them.

Police should ensure that proper protocols for aged people living in the first cluster should be made.



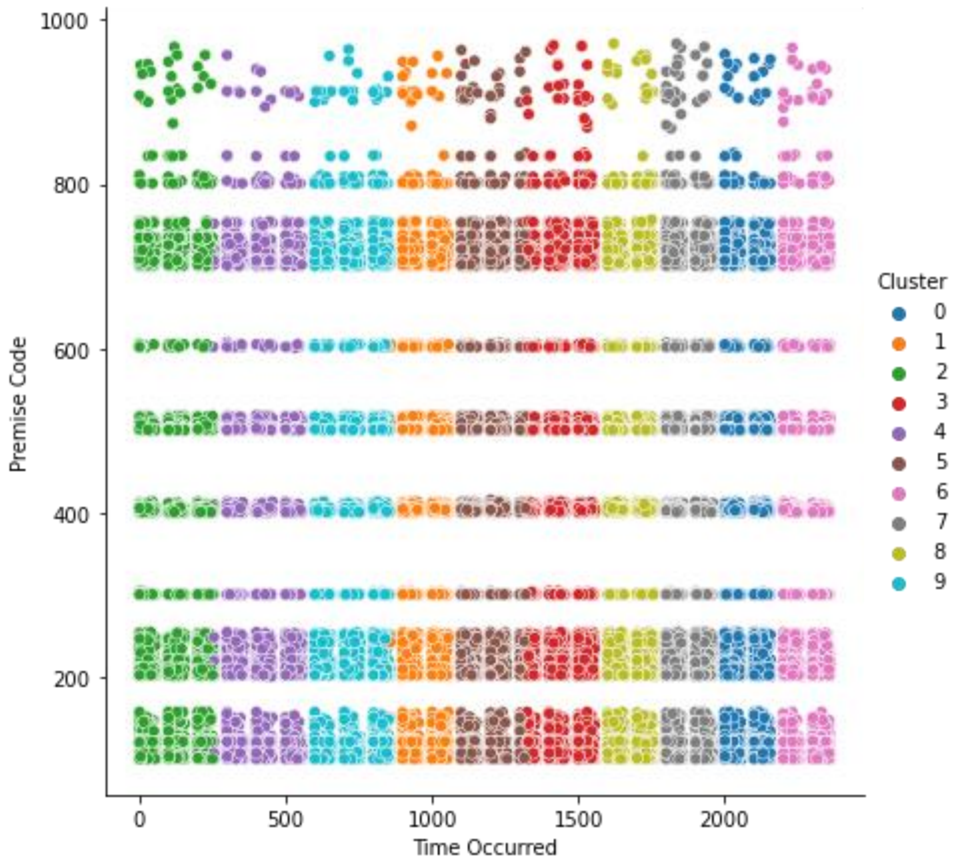
This is the same graph but without clustering.



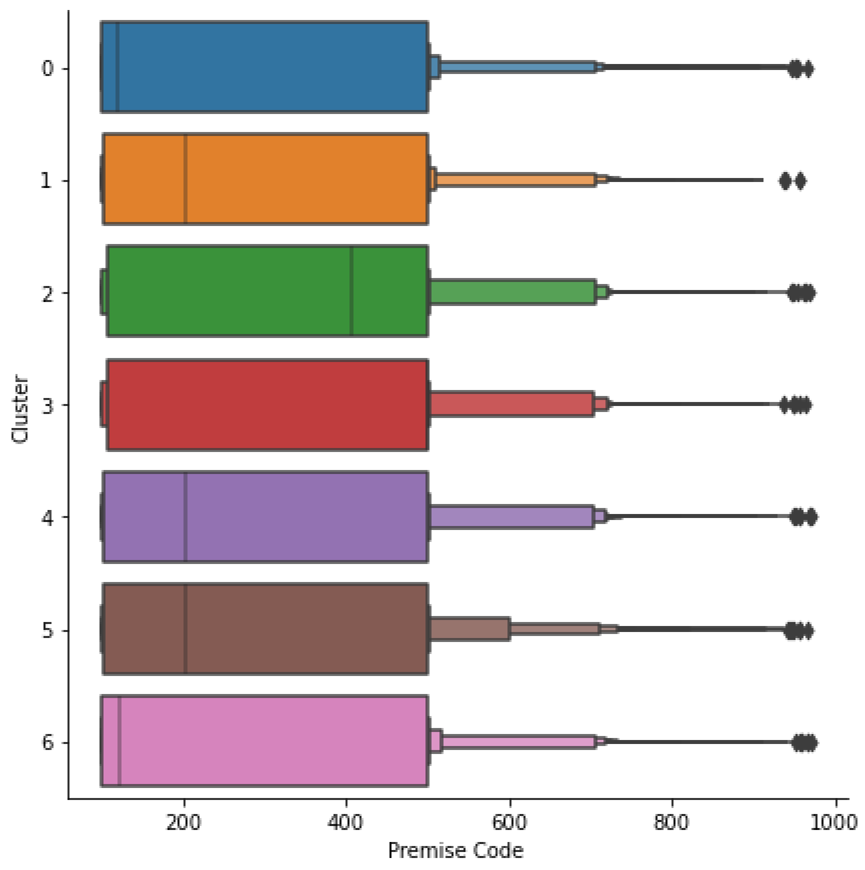
This graph is clustering on time occurred and victim age. This graph clearly shows the meaningful clusters that are formed. It not only shows which time is most prevalent but also shows its relation with the victim's age. It shows how people who are aged greater than 80 have more probability of being a victim between times 700 and 1600. Clustering allows them to be partitioned in meaningful ways and helps explain which time zones are clustered together. Each time zone therefore has its own set of attributes that defines it and is therefore meaningful for the police.

	Latitude	**Longitude**	**Time Occurred**	**Premise Code**	**Cluster**
0	34.1628	-118.6246	1410	501.0	3
1	34.1541	-118.5962	900	501.0	1
2	34.167	-118.5928	2230	101.0	6
3	34.2083	-118.5754	2115	101.0	0
4	34.201	-118.6147	2145	101.0	0

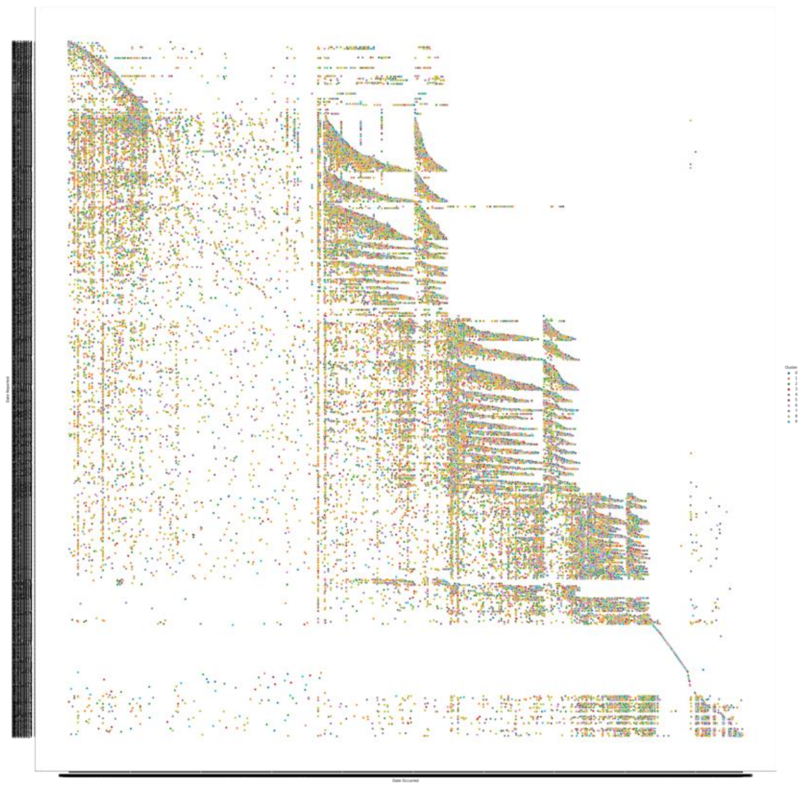
We then made a new data frame by manipulating the previous one to have separate columns for latitude and longitude, and see its effect on various attributes.



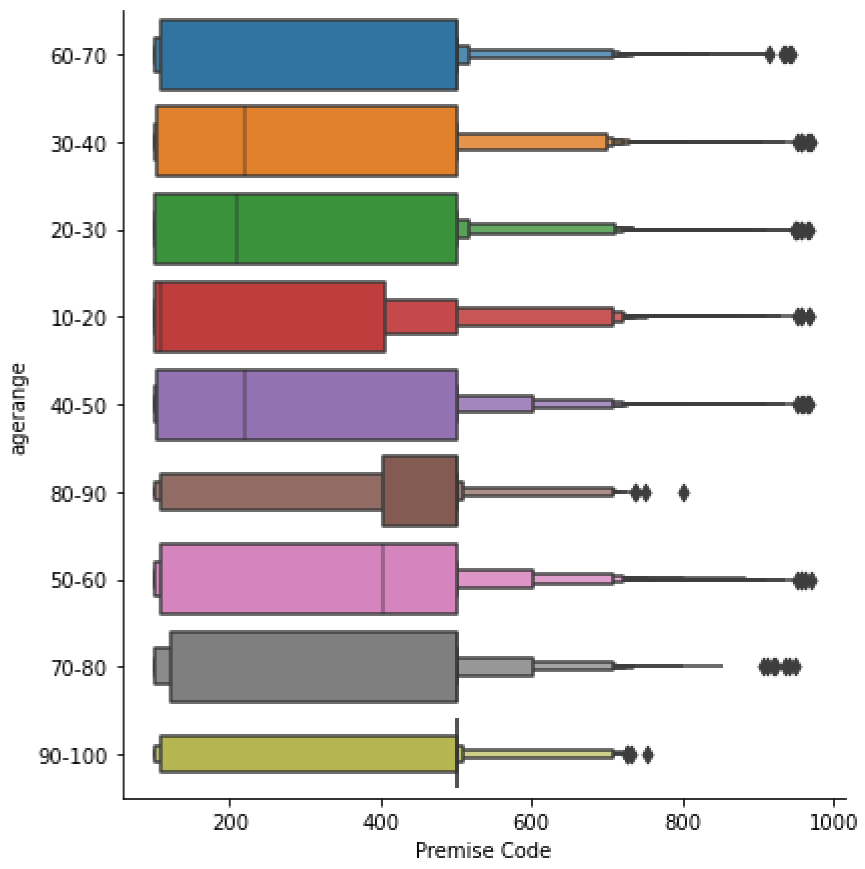
We can see how the time occurred is related to premise codes and the resulting clusters. Once again, the clusters are primarily made on the basis of time and this confirms that the timing of the crime is a major factor that enables clustering.



The box and whisker diagram are same for all clusters and therefore it shows that premise code is not a major factor however, most of the crimes are dense towards premise codes from 0 till 500 and thus police should focus on these codes more.



We also wanted to see the effect of date reported and date occurred. We can see from this graph that branches to the right that mostly date reported is much after the date occurred. Police should take this analysis into account for faster action after the incident. Failure to do so will result in many more losses of life and the criminals are able to flee easily.



This box plot shows the clustering of ages with premise code. It shows how people between 80-90 are mostly in 400-500 premise code. If action is not taken in this area more and more aged people will become victims. Cameras should be installed in these areas as well.

The police should overall focus on 0-500 premise codes and transfer police men accordingly.

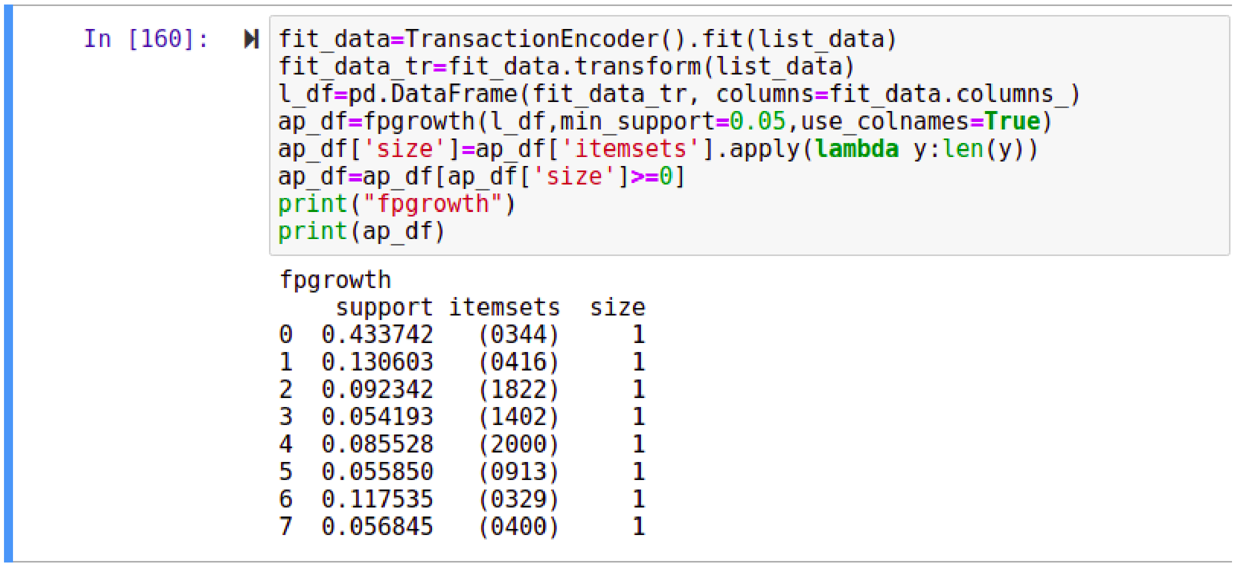
Frequent Pattern Mining

“Frequent patterns are item sets, subsequences, or substructures that appear in a data set with frequency no less than a user-specified threshold.” This is a very important data mining technique that helps find which items occur frequently.

For our data set it is extremely important as it will help police find the attributes that are linked to a crime. For example, in premise code x, age y people are killed the most then it is probably a sign of a serial killer who murders people of certain age at a certain place.

For frequent pattern mining we eventually used fp-growth algorithm, although we did implement Apriori as well. Apriori algorithm requires generation of candidate sets as well as requires multiple scans of database, both of these shortcomings are overcome by fpgrowth algorithm, thus making it faster and hence the choice for frequent pattern mining.

Frequent pattern in MO codes



```
In [160]:  fit_data=TransactionEncoder().fit(list_data)
           fit_data_tr=fit_data.transform(list_data)
           l_df=pd.DataFrame(fit_data_tr, columns=fit_data.columns_)
           ap_df=fpgrowth(l_df,min_support=0.05,use_colnames=True)
           ap_df['size']=ap_df['itemsets'].apply(lambda y:len(y))
           ap_df=ap_df[ap_df['size']>=0]
           print("fpgrowth")
           print(ap_df)
```

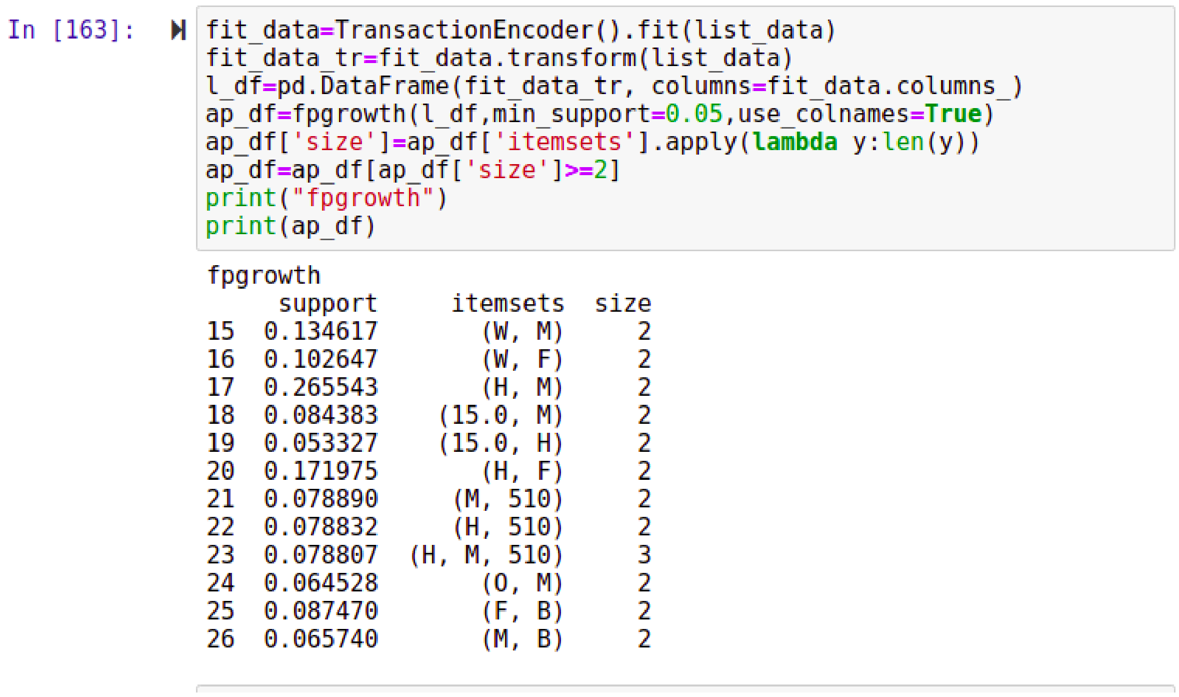
```
fpgrowth
    support itemsets  size
0  0.433742   (0344)     1
1  0.130603   (0416)     1
2  0.092342   (1822)     1
3  0.054193   (1402)     1
4  0.085528   (2000)     1
5  0.055850   (0913)     1
6  0.117535   (0329)     1
7  0.056845   (0400)     1
```

We manipulated the data set and applied fpgrowth algorithm on MO codes, it shows how certain MO codes are more prevalent than the others during a crime. These MO codes can be mapped onto “Removes vict property”, “Hit-Hit w/ weapon” and “Vandalized” amongst the other top 7 MO codes. This obviously has a lot of implications for the police as they would know how crimes are mostly committed, and will thus prepare to eradicate them accordingly

Frequent pattern in Victims

To harness the full advantage of fp growth algorithm we use it on multiple attributes to see which attributes occur frequently.

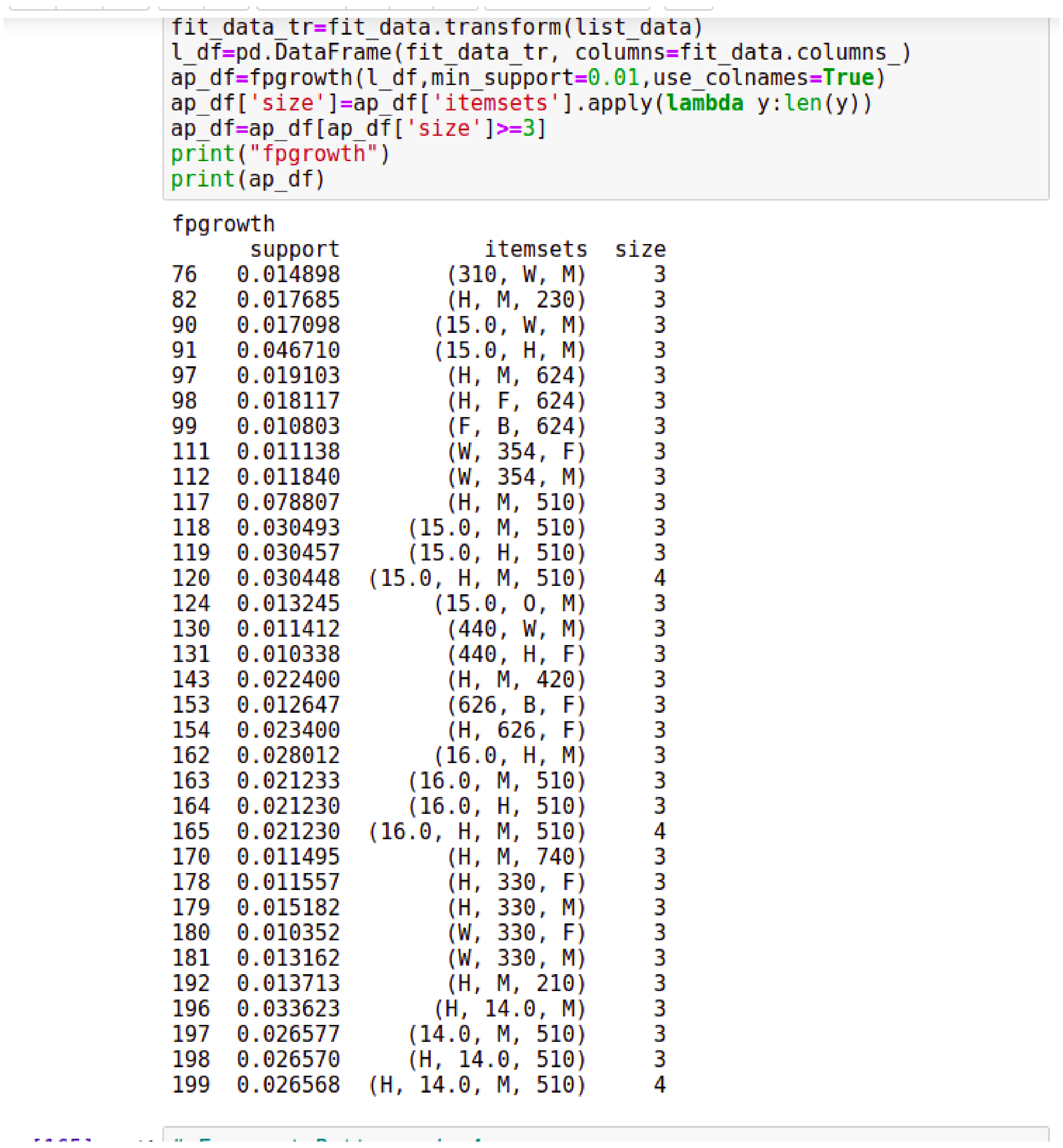
We next used "Crime Code", "Victim Age", "Victim Sex", "Victim Descent" attributes to see if they occur together. And we did get frequent item sets such as that in the figure below. We see how Hispanic males are the most dominant victims followed by Hispanic females, which is followed by white male and white female. We should also see that the crime code 510, male, and Hispanic is 3 set itemset. 510 crime code is misconduct in public by a drunk person and should be taken seriously. There are several rules that police can set such as no drinking late night etc.



```
In [163]:
fit_data=TransactionEncoder().fit(list_data)
fit_data_tr=fit_data.transform(list_data)
l_df=pd.DataFrame(fit_data_tr, columns=fit_data.columns_)
ap_df=fpgrowth(l_df,min_support=0.05,use_colnames=True)
ap_df['size']=ap_df['itemsets'].apply(lambda y:len(y))
ap_df=ap_df[ap_df['size']>=2]
print("fpgrowth")
print(ap_df)
```

```
fpgrowth
     support     itemsets  size
15  0.134617       (W, M)     2
16  0.102647       (W, F)     2
17  0.265543       (H, M)     2
18  0.084383    (15.0, M)     2
19  0.053327    (15.0, H)     2
20  0.171975       (H, F)     2
21  0.078890     (M, 510)     2
22  0.078832     (H, 510)     2
23  0.078807  (H, M, 510)     3
24  0.064528       (O, M)     2
25  0.087470       (F, B)     2
26  0.065740       (M, B)     2
```

We reduced the min support for the data set and have the following frequent item sets:



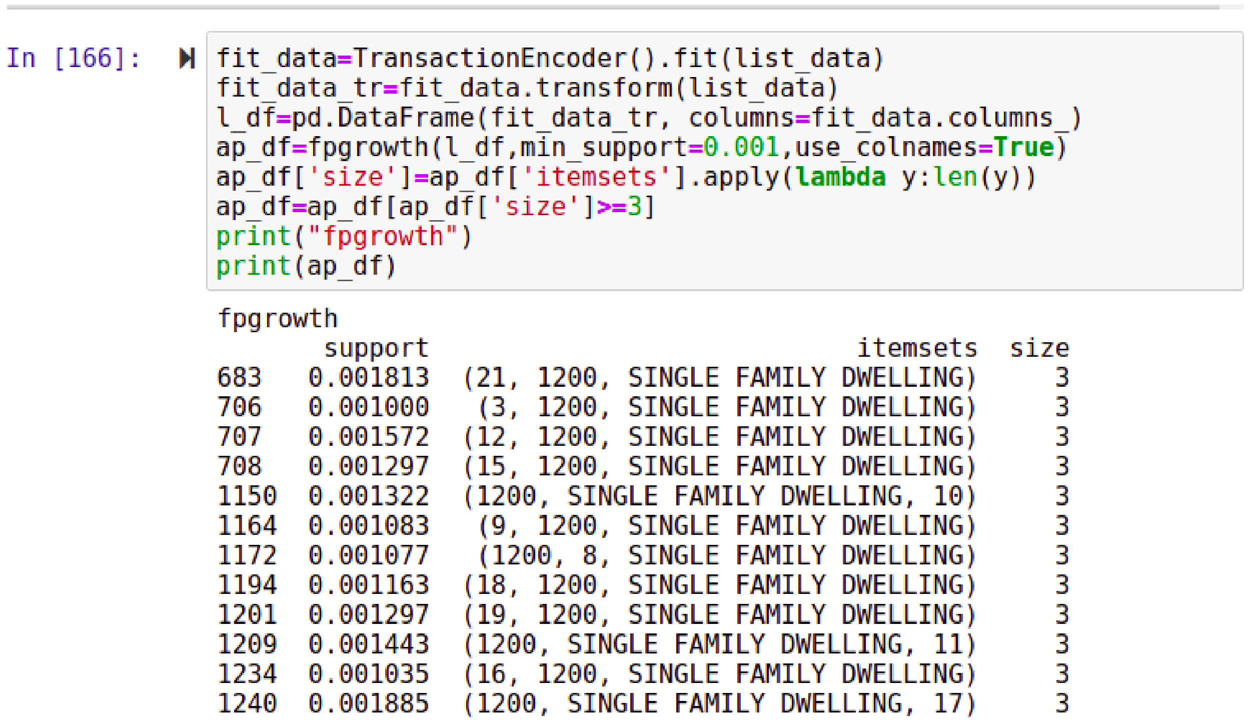
```
fit_data_tr=fit_data.transform(list_data)
l_df=pd.DataFrame(fit_data_tr, columns=fit_data.columns_)
ap_df=fpgrowth(l_df,min_support=0.01,use_colnames=True)
ap_df['size']=ap_df['itemsets'].apply(lambda y:len(y))
ap_df=ap_df[ap_df['size']>=3]
print("fpgrowth")
print(ap_df)
```

```
fpgrowth
      support            itemsets  size
76   0.014898        (310, W, M)     3
82   0.017685        (H, M, 230)     3
90   0.017098       (15.0, W, M)     3
91   0.046710       (15.0, H, M)     3
97   0.019103        (H, M, 624)     3
98   0.018117        (H, F, 624)     3
99   0.010803        (F, B, 624)     3
111  0.011138        (W, 354, F)     3
112  0.011840        (W, 354, M)     3
117  0.078807        (H, M, 510)     3
118  0.030493     (15.0, M, 510)     3
119  0.030457     (15.0, H, 510)     3
120  0.030448  (15.0, H, M, 510)     4
124  0.013245       (15.0, O, M)     3
130  0.011412        (440, W, M)     3
131  0.010338        (440, H, F)     3
143  0.022400        (H, M, 420)     3
153  0.012647        (626, B, F)     3
154  0.023400        (H, 626, F)     3
162  0.028012       (16.0, H, M)     3
163  0.021233     (16.0, M, 510)     3
164  0.021230     (16.0, H, 510)     3
165  0.021230  (16.0, H, M, 510)     4
170  0.011495        (H, M, 740)     3
178  0.011557        (H, 330, F)     3
179  0.015182        (H, 330, M)     3
180  0.010352        (W, 330, F)     3
181  0.013162        (W, 330, M)     3
192  0.013713        (H, M, 210)     3
196  0.033623       (H, 14.0, M)     3
197  0.026577     (14.0, M, 510)     3
198  0.026570     (H, 14.0, 510)     3
199  0.026568  (H, 14.0, M, 510)     4
```

Frequent pattern in Areas

We also wanted to see the effect of area on crimes and find frequent patterns in it.

We saw the following data. It shows how most crime is committed at 1200 on single family dwelling on youngsters between ages 15 and 21.



```
In [166]:  fit_data=TransactionEncoder().fit(list_data)
           fit_data_tr=fit_data.transform(list_data)
           l_df=pd.DataFrame(fit_data_tr, columns=fit_data.columns_)
           ap_df=fpgrowth(l_df,min_support=0.001,use_colnames=True)
           ap_df['size']=ap_df['itemsets'].apply(lambda y:len(y))
           ap_df=ap_df[ap_df['size']>=3]
           print("fpgrowth")
           print(ap_df)
```

```
fpgrowth
       support                                itemsets  size
683   0.001813  (21, 1200, SINGLE FAMILY DWELLING)     3
706   0.001000   (3, 1200, SINGLE FAMILY DWELLING)     3
707   0.001572  (12, 1200, SINGLE FAMILY DWELLING)     3
708   0.001297  (15, 1200, SINGLE FAMILY DWELLING)     3
1150  0.001322  (1200, SINGLE FAMILY DWELLING, 10)     3
1164  0.001083   (9, 1200, SINGLE FAMILY DWELLING)     3
1172  0.001077   (1200, 8, SINGLE FAMILY DWELLING)     3
1194  0.001163  (18, 1200, SINGLE FAMILY DWELLING)     3
1201  0.001297  (19, 1200, SINGLE FAMILY DWELLING)     3
1209  0.001443  (1200, SINGLE FAMILY DWELLING, 11)     3
1234  0.001035  (16, 1200, SINGLE FAMILY DWELLING)     3
1240  0.001885  (1200, SINGLE FAMILY DWELLING, 17)     3
```

IMPROVEMENTS

We believe our project would have improved if we applied data mining techniques to all areas separately as each area might have its own set of attributes that make an area unique. We were however able to retrieve age data from such areas through clustering.