Project - Data Visualisation based on 311 Service Requests, Census Demographics and Crimes in Baltimore

Problem Statement :

To find a relationship between the services requested by the population of different age groups in a neighborhood. Also, to find a relationship of crime rates with respect to the services requested. For this, 3 datasets were explored - Census Demographics, Crime by Neighborhood and 311 Customer Service Requests.

Description:

We have found the relationship between Services Requested in particular neighborhood and the census demographics. For example, as the population increases, number of service requests also increases. We also evaluated majority of age groups in particular neighborhood and type of requests that were majorly created in that particular area. Also, we found that the method of creating requests varied based on the age group in that particular neighborhood. For example, if a particular neighborhood contains large number of young crowd, it will evidently show large number of internet reported service requests. Also, we related Crime by neighborhood dataset with 311 Service requests by comparing crime rate and the requests rate. This explains the relationship between the number of requests raised and the number of crimes committed in a neighborhood. Another thing that we concluded based on plots is that the areas with higher population shows maximum number of crimes.

About Datasets

Here, We are fetching 3 datasets viz- 1) District 311 Service Request dataset present in City service section of Open Baltimore 2) Crime by neighborhood present in Crime section of Open Baltimore 3) Census Neighborhoods dataset found on Data.gov website.(All the columns are similar to Census dataset from Open Baltimore. But the neighborhood values from 311 Service requests and Crime did not match. Therefore, chosen to use it from Data.gov)

Service Type tells you about the type of issue that is raised, for example, housing and community development (HCD), transportation roadway maintenance (e.g. fixing potholes), etc. Method Received tells us about the mode of communicating the request. It can be Phone, API or Internal. Some important dates in context of Service requests are also provided like Due date, created date and Status date. Also the timestamp of the last activity that is performed for the particular request is also mentioned.

Census by Neighborhood dataset consists of Neighborhood, total population in that neighborhood, total female and male population, percentage of population in different age groups like 0-4,5-11,65 above, etc.

Crime dataset consists of crime date, crime code which states the type of crime that is committed, neighborhood where the crime was committed, location states actual location of crime, description describes the crime, weapon that was used and district.

How is it useful?

- The data consists of details about every phone call, query, complaint and request addressed by the 311 system adopted by Baltimore government. This dataset is useful to address how the service requests were handled by government workers and the type of problems that bothered Baltimore citizens the most.
- Crime dataset and census demographics dataset gives the overall information about the crimes and population in baltimore. Census demographic dataset will be very useful to check what is the majority age group in particular neighborhood of baltimore. Crimes gives idea about the type of crimes and the details about the type of crime in every neighborhood of baltimore.

1. Initially import all the required libraries

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import datetime
```

2. Now fetch the dataset downloaded from Open Baltimore website

In [3]:
```
df = pd.read_csv("D:\MS CS Sem 1\DS\Assignment 1\Dbs\District_4_311_Service_Requests.csv")
```

In [4]:
```
df2 = pd.read_csv("D:\MS CS Sem 1\DS\Project\Dataset\Census_Neighborhoods.csv", low_memory = False)
```

In [5]:
```
df3 = pd.read_csv("D:\MS CS Sem 1\DS\Project\Dataset\Crime_By_Neighborhood.csv")
```

In [6]:
```
df.shape
```

Out[6]: (20685, 13)

In [7]:
```
df2.shape
```

Out[7]: (278, 37)

In [8]:
```
df3.shape
```

Out[8]: (49559, 7)

It shows that the service requests dataset consists of **20685 rows** and **13 columns**. Whereas Census Demographic dataset consists of **56 rows** and **28 columns**. And Crime by neighborhood contains **49559 rows** and **7 columns**.

Drawing connections between three datasets:

The essence of our connections between three datasets lies in 'Neighborhood' column. All the three datasets i.e. Crime by neighborhood, Census Demographics and 311 service requests revolves around neighborhood. So based on this field, we were able to merge all the datasets. This was also a reason why we could find out the relationships between them.
Why was this a good idea?

Since the crimes can be compared with service requests, service requests can be compared with both Crimes and Population and Population can be compared with Crimes using neighborhood, we thought this was a good choice.

3. Checking all the columns available and selecting few amongst them

In [9]:
```
df.columns
```

Out[9]:
```
Index(['ServiceRequestNum', 'SRType', 'Neighborhood', 'Address', 'ZipCode',
       'MethodReceived', 'CreatedDate', 'DueDate', 'StatusDate', 'SRStatus',
       'LastActivity', 'Outcome', 'LastActivityDate'],
      dtype='object')
```

In [10]:
```
df_service = df[['ServiceRequestNum','SRType','MethodReceived','CreatedDate',
'StatusDate','DueDate','LastActivityDate','Neighborhood']]
```

In [11]:
```
df_census = df2.reindex(columns=['Name','Population','Male','Female','AGE0_4',
'AGE5_11','AGE12_14','AGE15_17','AGE18_24','AGE25_34','AGE35_44','AGE45_64','A
GE65ovr'])
```

Change the column names called 'CSA2010' to 'Neighborhood' and 'tpop10' to Population

In [12]:
```
df_census = df_census.rename(columns={"Name":"Neighborhood"})
```

Dataframe df_service now contains only the required 7 columns and df_census1 contains 13.

4. Now let's fetch first few records of both the datasets.

In [13]:
```
df_service.head()
```

Out[13]:

	ServiceRequestNum	SRType	MethodReceived	CreatedDate	StatusDate	DueDate
0	15-00000620	BGE-StLight(s) Out	API	1/1/2015 17:45	1/2/2015	1/5/2015
1	15-00000187	TRS-Abandoned Vehicle	Phone	1/1/2015 10:25	1/8/2015	1/6/2015
2	15-00000217	TRM-Pothole Repair	API	1/1/2015 10:54	1/2/2015	1/3/2015
3	15-00000236	BGE-StLight(s) Out	Phone	1/1/2015 11:09	1/2/2015	1/5/2015
4	15-00000241	BGE-StLight(s) Out	Phone	1/1/2015 11:14	1/2/2015	1/5/2015

In [14]:
```
df_census.head()
```

Out[14]:

	Neighborhood	Population	Male	Female	AGE0_4	AGE5_11	AGE12_14	AGE15_17
0	Abell	889.0	424.0	465.0	55.0	43.0	5.0	10.0
1	Allendale	3554.0	1536.0	2018.0	179.0	315.0	136.0	181.0
2	Arcadia	1235.0	592.0	643.0	101.0	85.0	40.0	55.0
3	Boyd-Booth	822.0	408.0	414.0	60.0	94.0	26.0	40.0
4	Arlington	2598.0	1219.0	1379.0	148.0	237.0	92.0	114.0

In [15]:
```
df3.head()
```

Out[15]:

	CrimeDate	CrimeCode	Location	Description	Weapon	District	Neighbo
0	1/1/2012	4A	500 CHATEAU AV	AGG. ASSAULT	FIREARM	NORTHERN	Winston-Govans
1	1/1/2012	4E	2200 W NORTH AV	COMMON ASSAULT	HANDS	WESTERN	Coppin Heights/A Co-Eas
2	1/1/2012	4E	0 S LINWOOD AV	COMMON ASSAULT	HANDS	SOUTHEASTERN	Patterson Neighborl
3	1/1/2012	6D	5600 LOCH RAVEN BD	LARCENY FROM AUTO	NaN	NORTHEASTERN	Loch Rav
4	1/1/2012	6D	6300 PIONEER DR	LARCENY FROM AUTO	NaN	NORTHEASTERN	Hamilton

- There are many missing values present in LastActivityDate column.
- The data types of 3 date columns varies. Columns with the missing values can not be used for service requests analysis.
- There are 4 types of Methods for Receiving requests. 4 different types of values are present in column 'MethodReceived'

5. Working with missing values in datasets:

Drop the null values present in Service and Crime dataset. Also, we deleted null values in Services. The reason for deleting these values was that, it might impact the results that we concluded.

In [16]:
```
len(df_service)
```

Out[16]: 20685

In [17]:
```
df_service = df_service.dropna()
```

In [18]:
```
len(df_service)
```

Out[18]: 14943

In [19]:
```
df_crime = df3.dropna()
len(df_crime)
```

Out[19]: 17141

In [20]:
```
df_crime
```

Out[20]:

	CrimeDate	CrimeCode	Location	Description	Weapon	Dist
0	1/1/2012	4A	500 CHATEAU AV	AGG. ASSAULT	FIREARM	NORTHERN
1	1/1/2012	4E	2200 W NORTH AV	COMMON ASSAULT	HANDS	WESTERN
2	1/1/2012	4E	0 S LINWOOD AV	COMMON ASSAULT	HANDS	SOUTHEASTE
6	1/1/2012	4E	500 N GLOVER ST	COMMON ASSAULT	HANDS	SOUTHEASTE
7	1/1/2012	4E	4900 GOODNOW RD	COMMON ASSAULT	HANDS	NORTHEASTE
11	1/1/2012	4C	500 LONEYS LA	AGG. ASSAULT	OTHER	SOUTHEASTE
12	1/1/2012	4E	1500 GORSUCH AV	COMMON ASSAULT	HANDS	NORTHEASTE
14	1/1/2012	4C	400 E BALTIMORE ST	AGG. ASSAULT	OTHER	CENTRAL
18	1/1/2012	4E	300 N HIGHLAND AV	COMMON ASSAULT	HANDS	SOUTHEASTE
20	1/1/2012	4C	2000 GRINNALDS AV	AGG. ASSAULT	OTHER	SOUTHWESTE
21	1/1/2012	4E	500 N ELLWOOD AV	COMMON ASSAULT	HANDS	SOUTHEASTE
27	1/1/2012	4B	1900 E NORTH AV	AGG. ASSAULT	KNIFE	EASTERN
28	1/1/2012	3CF	1800 RUSSELL ST	ROBBERY - COMMERCIAL	FIREARM	SOUTHERN
34	1/1/2012	4C	0 E RANDALL ST	AGG. ASSAULT	OTHER	SOUTHERN
36	1/1/2012	4E	1000 N WOLFE ST	COMMON ASSAULT	HANDS	EASTERN
37	1/1/2012	4E	1000 N WOLFE ST	COMMON ASSAULT	HANDS	EASTERN
38	1/1/2012	4E	2800 CARVER RD	COMMON ASSAULT	HANDS	SOUTHERN

	CrimeDate	CrimeCode	Location	Description	Weapon	Dist
41	1/1/2012	9S	500 OAKLAND AV	SHOOTING	FIREARM	NORTHERN
42	1/1/2012	4A	500 OAKLAND AVE	AGG. ASSAULT	FIREARM	NORTHERN
45	1/1/2012	4E	400 N ROSE ST	COMMON ASSAULT	HANDS	SOUTHEASTE
46	1/1/2012	4C	5700 PLAINFIELD AV	AGG. ASSAULT	OTHER	NORTHEASTE
48	1/1/2012	4C	2100 BAKER ST	AGG. ASSAULT	OTHER	WESTERN
50	1/1/2012	4E	2600 GREENMOUNT AV	COMMON ASSAULT	HANDS	NORTHERN
52	1/1/2012	4E	1800 W LAFAYETTE AV	COMMON ASSAULT	HANDS	WESTERN
61	1/1/2012	3JF	700 N FULTON AV	ROBBERY - RESIDENCE	FIREARM	WESTERN
64	1/1/2012	4C	400 BOULDIN ST	AGG. ASSAULT	OTHER	SOUTHEASTE
65	1/1/2012	4E	0 N GAY ST	COMMON ASSAULT	HANDS	CENTRAL
66	1/1/2012	4E	4000 EIERMAN AV	COMMON ASSAULT	HANDS	NORTHEASTE
67	1/1/2012	4E	1100 KEVIN RD	COMMON ASSAULT	HANDS	SOUTHWESTE
69	1/1/2012	4E	800 BETHUNE RD	COMMON ASSAULT	HANDS	SOUTHERN
...	...	...	...	...	...	...
49462	12/31/2012	3AF	3700 ELLERSLIE AV	ROBBERY - STREET	FIREARM	NORTHERN
49463	12/31/2012	3AF	3700 ELLERSLIE AV	ROBBERY - STREET	FIREARM	NORTHERN
49467	12/31/2012	4E	2400 FREDERICK AV	COMMON ASSAULT	HANDS	SOUTHWESTE
49468	12/31/2012	3AK	0 E HENRIETTA ST	ROBBERY - STREET	KNIFE	SOUTHERN
49474	12/31/2012	4E	800 PARK AV	COMMON ASSAULT	HANDS	CENTRAL

	CrimeDate	CrimeCode	Location	Description	Weapon	Dist
49475	12/31/2012	3CF	4100 NORFOLK AV	ROBBERY - COMMERCIAL	FIREARM	NORTHWESTE
49481	12/31/2012	1F	2100 BRYANT AV	HOMICIDE	FIREARM	WESTERN
49483	12/31/2012	4E	1300 HARLEM AV	COMMON ASSAULT	HANDS	WESTERN
49488	12/31/2012	4E	200 N HOWARD ST	COMMON ASSAULT	HANDS	CENTRAL
49490	12/31/2012	4E	1200 ROSSITER AV	COMMON ASSAULT	HANDS	NORTHEASTE
49491	12/31/2012	4C	0 S ABINGTON AV	AGG. ASSAULT	OTHER	SOUTHWESTE
49492	12/31/2012	4C	2600 MARBOURNE AV	AGG. ASSAULT	OTHER	SOUTHERN
49496	12/31/2012	4A	5600 WINNER AV	AGG. ASSAULT	FIREARM	NORTHWESTE
49501	12/31/2012	4E	300 S PULASKI ST	COMMON ASSAULT	HANDS	SOUTHWESTE
49503	12/31/2012	4E	1200 LINWOOD AVE	COMMON ASSAULT	HANDS	EASTERN
49504	12/31/2012	4E	300 STOCKHOLM ST	COMMON ASSAULT	HANDS	SOUTHERN
49505	12/31/2012	4E	300 STOCKHOLM ST	COMMON ASSAULT	HANDS	SOUTHERN
49509	12/31/2012	4E	5100 GOODNOW RD	COMMON ASSAULT	HANDS	NORTHEASTE
49519	12/31/2012	4C	200 MONTFORD AVE	AGG. ASSAULT	OTHER	SOUTHEASTE
49520	12/31/2012	4E	4900 MIDWOOD AV	COMMON ASSAULT	HANDS	NORTHERN
49521	12/31/2012	4E	900 DANTREY CT	COMMON ASSAULT	HANDS	SOUTHERN
49522	12/31/2012	4E	900 DANTREY CT	COMMON ASSAULT	HANDS	SOUTHERN

	CrimeDate	CrimeCode	Location	Description	Weapon	Dist
49528	12/31/2012	4E	2000 WILKINS AVE	COMMON ASSAULT	HANDS	SOUTHERN
49535	12/31/2012	4E	2000 WILKENS AV	COMMON ASSAULT	HANDS	SOUTHERN
49539	12/31/2012	4E	2800 HINSDALE DR	COMMON ASSAULT	HANDS	SOUTHERN
49540	12/31/2012	4E	3500 CHESTERFIELD AV	COMMON ASSAULT	HANDS	NORTHEASTE
49544	12/31/2012	3AF	6000 NORTHWOOD DR	ROBBERY - STREET	FIREARM	NORTHERN
49545	12/31/2012	3AF	3800 W FOREST PARK AV	ROBBERY - STREET	FIREARM	NORTHWESTE
49546	12/31/2012	4B	1000 CATHEDRAL ST	AGG. ASSAULT	KNIFE	CENTRAL
49551	12/31/2012	4E	200 S DALLAS CT	COMMON ASSAULT	HANDS	SOUTHEASTE

17141 rows × 7 columns

Creating a copy of census dataframe. Deleting the rows where Population is equal to zero.Population in any of the neighborhood can not be equal to zero. Such values from the dataset must thus be removed.

In [21]:
```
df_census1 = df_census
columns=['Population']
len(df_census1)
```

Out[21]: 278

In [22]:
```
df_census1 = df_census1.replace(0, pd.np.nan).dropna(axis=0, how='any', subset
=columns).fillna(0)
```

In [23]:
```
len(df_census1)
```

Out[23]: 256

6. Computing new column using existing columns to derive useful information

Since the request created date and Date of the last activity performed is present in the above dataset, We'll calculate the number of days required to resolve each request. We'll assign 'Number of Days' to dataframe as a new column

Also, Young Citizens column will be created using addition of columns upto 35-44 age groups.

Similarly, senior Citizen column is created using addition of columns having age group higher that 44.

In [24]:
```
import datetime
df_service.LastActivityDate = pd.to_datetime(df_service.LastActivityDate)
df_service.DueDate = pd.to_datetime(df_service.DueDate)
df_service.CreatedDate = pd.to_datetime(df_service.CreatedDate)

df_service = df_service.assign(DaysRequired = df_service.LastActivityDate - df
_service.CreatedDate + datetime.timedelta(days=1))
```

In [25]:
```
df_service.columns
```

Out[25]:
```
Index(['ServiceRequestNum', 'SRType', 'MethodReceived', 'CreatedDate',
       'StatusDate', 'DueDate', 'LastActivityDate', 'Neighborhood',
       'DaysRequired'],
      dtype='object')
```

In [26]:
```
df_census1['Young_citizens'] = df_census1['AGE0_4'] + df_census1['AGE5_11'] +
df_census1['AGE12_14'] + df_census1['AGE15_17'] + df_census1['AGE18_24'] + df_
census1['AGE25_34'] + df_census1['AGE35_44']
```

In [27]:
```
df_census1['Senior_citizens'] = df_census1['AGE45_64'] + df_census1['AGE65ovr'
]
```

7. Processing of the columns values

I'll change the data type of 'Number of columns' to only 'Days', since I'am interested to know only the number of days for plotting the graph

In [28]:
```
df_service.DaysRequired = pd.to_timedelta(df_service.DaysRequired,unit='d').as
type('timedelta64[D]')
df_service.DaysRequired.head()
df_service.head()
```

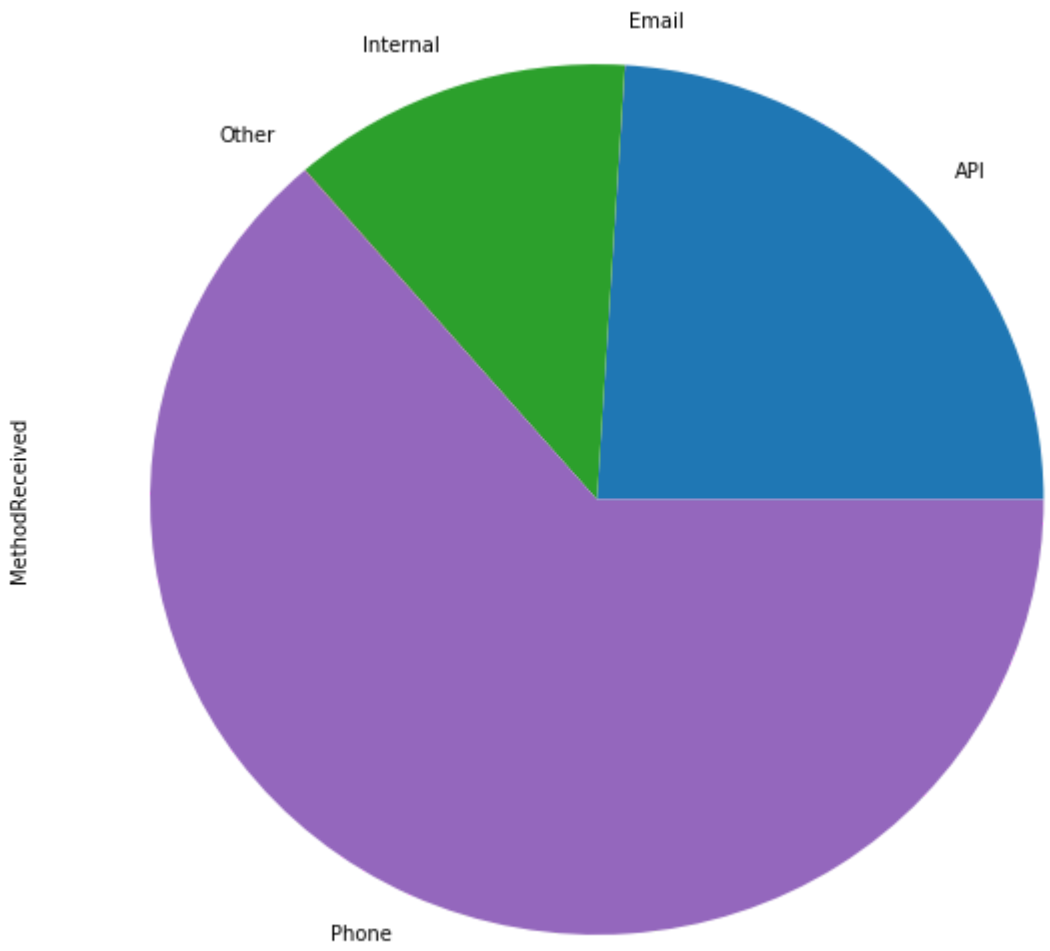
Out[28]:

	ServiceRequestNum	SRType	MethodReceived	CreatedDate	StatusDate	DueDate	L
2	15-00000217	TRM-Pothole Repair	API	2015-01-01 10:54:00	1/2/2015	2015-01-03	2
5	15-00000412	SW-Dirty Alley	Phone	2015-01-01 13:09:00	1/5/2015	2015-01-08	2
13	15-00001181	SW-Dirty Street	Phone	2015-01-02 09:05:00	1/5/2015	2015-01-09	2
16	15-00002685	TRM-Pickup Pothole	Internal	2015-01-02 14:31:00	1/2/2015	2015-01-02	2
17	15-00002689	TRM-Pickup Pothole	Internal	2015-01-02 14:33:00	1/2/2015	2015-01-02	2

8. Based on Method via which the request was received (Method Received column)

The pie chart below shows that the maximum requests in baltimore neighborhoods were sent via phone calls whereas only single request was raised via Email.

In [29]:
```
fig, ax = plt.subplots(figsize=(10,10))
AverageNumberOfDays = df_service.groupby('MethodReceived')['MethodReceived'].c
ount().plot.pie(subplots=True)
```



9. Find relationship between population and number of service request using 311 service request and Census Demographic datasets

Merging the Neighborhood and services from Services dataset and Population from Census demographic dataset for displaying the relationship between them.

In [30]:
```
df_service1 = df_service.groupby(["Neighborhood"]).size()
```

In [31]:
```
df_services = pd.Series.to_frame(df_service1)
df_census2 = df_census1[['Neighborhood','Population']]
```

In [32]:
```
df_merged = pd.merge(df_census2,df_services, on='Neighborhood')
```

In [33]:
```
df_merged = df_merged.rename(columns={0:"Services"})
```

In [34]:
```
df_merged.head()
```

Out[34]:

	Neighborhood	Population	Services
0	Bellona-Gittings	599.0	162
1	Cameron Village	1435.0	648
2	Cedarcroft	726.0	200
3	Chinquapin Park	1309.0	573
4	Loch Raven	6163.0	1604

Since the String values can not be plotted on 3D graph, we converted the values to numeric form. These numeric values will thus be used to plot graphs.

In [35]:
```
neighborhood_num = {"Neighborhood": {"Bellona-Gittings":1,
"Cameron Village":2,
"Cedarcroft":3,
"Chinquapin Park":4,
"Loch Raven":5,
"Glen Oaks":6,
"Guilford":7,
"Homeland":8,
"Idlewood":9,
"Kenilworth Park":10,
"Kernewood":11,
"Lake Evesham":12,
"Lake Walker":13,
"Loyola/Notre Dame":14,
"Mid-Govans":15,
"New Northwood":16,
"Pen Lucy":17,
"Radnor-Winston":18,
"Ramblewood":19,
"Richnor Springs":20,
"Rosebank":21,
"Villages Of Homeland":22,
"Woodbourne Heights":23,
"Wilson Park":24,
"Evesham Park":25,
"Belvedere":26,
"York-Homeland":27}}
```

In [36]:
```
df_merged.replace(neighborhood_num, inplace = True)
```

In [37]:
```
df_merged.Neighborhood
```

Out[37]:
```
0      1
1      2
2      3
3      4
4      5
5      6
6      7
7      8
8      9
9     10
10    11
11    12
12    13
13    14
14    15
15    16
16    17
17    18
18    19
19    20
20    21
21    22
22    23
23    24
24    25
25    26
26    27
Name: Neighborhood, dtype: int64
```

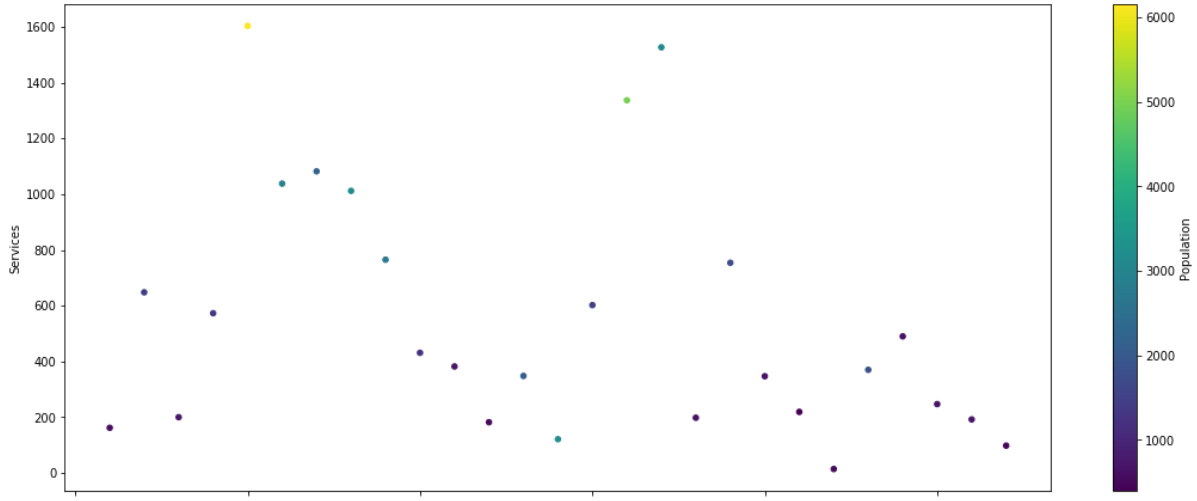
3D plot is plotted against Neighborhood values which are converted into integers on x axis, Count of services for each neighborhood on y axis and Total Population on Z axis. Viridis colormap style is used. Lowest color shade shows highest population.

In [38]:
```
df_merged.Population
```

Out[38]:
```
0      599.0000
1     1435.0000
2      726.0000
3     1309.0000
4     6163.0000
5     2887.0000
6     2214.0000
7     3201.0000
8     2676.0000
9     1206.0000
10     726.0000
11     543.0000
12    2095.0000
13    3208.0000
14    1465.0000
15    4972.0000
16    3078.0000
17     684.0000
18    1705.0000
19     676.0000
20     394.0000
21     562.3255
22    1795.0000
23     632.3202
24     709.0000
25     720.0000
26     482.6745
Name: Population, dtype: float64
```

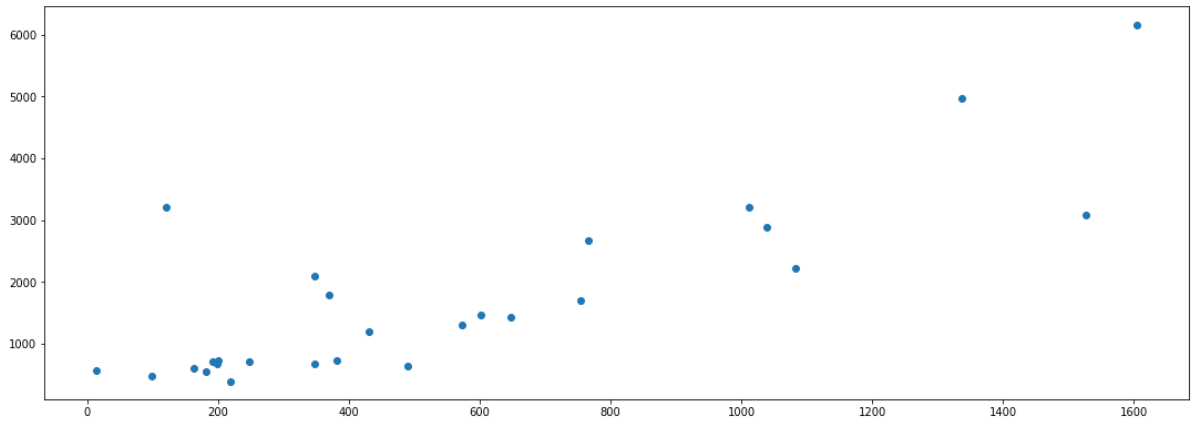
3D plot is plotted against Neighborhood values which are converted into integers on x axis, Count of services for each neighborhood on y axis and Total Population on Z axis. Viridis colormap style is used. Lowest color shade shows highest population.

In [39]:
```
ax2  = df_merged.plot.scatter(x='Neighborhood',
                              y='Services',
                              c='Population',
                              colormap='viridis',
                              figsize=(18.5,7.5))
```



Plotting the 2D graph of Services vs Population shows that the Number of Service request for any given neighborhood in Baltimore are directly proportional to the total population in that particular area. There are negligible outliers which can be ignored.

In [40]:
```
plt.figure(figsize=(18.5,6.5))
plt.scatter(df_merged['Services'], df_merged['Population'])
plt.show()
```



10. Plot relationship between majority of age groups and method of raising the service request.

In [41]:
```
df_census1.columns
```

Out[41]:
```
Index(['Neighborhood', 'Population', 'Male', 'Female', 'AGE0_4', 'AGE5_11',
       'AGE12_14', 'AGE15_17', 'AGE18_24', 'AGE25_34', 'AGE35_44', 'AGE45_6
4',
       'AGE65ovr', 'Young_citizens', 'Senior_citizens'],
      dtype='object')
```

In [42]:
```
df_cen_sorted = df_census1.sort_values(by=['Young_citizens'])
df_cen_sorted.head()
```

Out[42]:

	Neighborhood	Population	Male	Female	AGE0_4	AGE5_11	AGE12_14	AGE15_17
122	Johns Hopkins Homewood	18.0	8.0	10.0	4.0	0.0	0.0	0.0
19	Blythewood	72.0	38.0	34.0	2.0	12.0	2.0	5.0
116	Hopkins Bayview	103.0	46.0	57.0	2.0	3.0	1.0	2.0
50	Dickeyville	156.0	73.0	83.0	12.0	5.0	0.0	1.0
171	Saint Paul	107.0	55.0	52.0	9.0	14.0	5.0	3.0

In [43]:
```
df_cen_sorted.tail()
```

Out[43]:

	Neighborhood	Population	Male	Female	AGE0_4	AGE5_11	AGE12_14	AGE15_1
25	Brooklyn	9996.0	4870.0	5126.0	1055.0	1163.0	378.0	390.0
276	Charles Village	8906.0	4676.0	4230.0	146.0	58.0	26.0	47.0
29	Canton	12192.0	6017.0	6175.0	512.0	232.0	80.0	88.0
274	Belair-Edison	16690.0	7500.0	9190.0	1134.0	1734.0	794.0	873.0
45	Frankford	17694.0	8056.0	9638.0	1383.0	1708.0	654.0	748.0

In [44]:
```
df_service.loc[df_service['Neighborhood'] == 'Chinquapin Park'].groupby('Metho
dReceived').size()
```

Out[44]:
```
MethodReceived
API          92
Internal     82
Phone       399
dtype: int64
```

In [45]:
```
neighborhood = df_merged['Neighborhood']
neighborhood = pd.Series.to_frame(neighborhood)
print(neighborhood)
```

```
    Neighborhood
0              1
1              2
2              3
3              4
4              5
5              6
6              7
7              8
8              9
9             10
10            11
11            12
12            13
13            14
14            15
15            16
16            17
17            18
18            19
19            20
20            21
21            22
22            23
23            24
24            25
25            26
26            27
```

In [46]:
```
df12 = df_service[['Neighborhood','MethodReceived']]
df12 = df12.sort_values(by=['Neighborhood'])
df12
```

Out[46]:

	Neighborhood	MethodReceived
18002	Bellona-Gittings	Internal
18914	Bellona-Gittings	Phone
3143	Bellona-Gittings	API
6367	Bellona-Gittings	API
3175	Bellona-Gittings	API
17561	Bellona-Gittings	API
12066	Bellona-Gittings	API
12086	Bellona-Gittings	API
14089	Bellona-Gittings	API
14070	Bellona-Gittings	Phone
3251	Bellona-Gittings	API
5415	Bellona-Gittings	API
8999	Bellona-Gittings	Phone
8932	Bellona-Gittings	Phone
8905	Bellona-Gittings	Phone
19085	Bellona-Gittings	Phone
14017	Bellona-Gittings	API
8830	Bellona-Gittings	Phone
5637	Bellona-Gittings	API
8765	Bellona-Gittings	Phone
17380	Bellona-Gittings	API
5684	Bellona-Gittings	API
5690	Bellona-Gittings	Phone
19074	Bellona-Gittings	API
5695	Bellona-Gittings	Internal
5413	Bellona-Gittings	API
5411	Bellona-Gittings	API
11423	Bellona-Gittings	API
11424	Bellona-Gittings	API
18684	Bellona-Gittings	API
...	...	...
16070	York-Homeland	Phone

	Neighborhood	MethodReceived
5553	York-Homeland	Internal
1319	York-Homeland	Phone
4250	York-Homeland	Internal
7666	York-Homeland	Phone
10217	York-Homeland	Phone
481	York-Homeland	Phone
14879	York-Homeland	API
20241	York-Homeland	Phone
10975	York-Homeland	Internal
4354	York-Homeland	Phone
10976	York-Homeland	Internal
2836	York-Homeland	Phone
13839	York-Homeland	API
6106	York-Homeland	Phone
11658	York-Homeland	API
17935	York-Homeland	API
19076	York-Homeland	API
4416	York-Homeland	Internal
7259	York-Homeland	Internal
72	York-Homeland	Internal
19881	York-Homeland	Phone
15071	York-Homeland	Phone
3770	York-Homeland	API
4173	York-Homeland	Internal
11866	York-Homeland	Phone
13804	York-Homeland	API
5316	York-Homeland	Phone
17934	York-Homeland	Phone
14335	York-Homeland	Phone

14943 rows × 2 columns

In [47]:
```
api = df_service[ (df_service.MethodReceived == 'API') | (df_service.MethodRec
eived == 'API')]
api = api.groupby('Neighborhood').size()
api = pd.Series.to_frame(api)
api_sorted = api.sort_values(by=['Neighborhood'])
api_sorted = api_sorted.rename(columns={0:"Api_count"})
type(api_sorted)
```

Out[47]: pandas.core.frame.DataFrame

In [48]:
```
df_cen_api = pd.merge(df_cen_sorted,api_sorted, on='Neighborhood')
df_cen_api
```

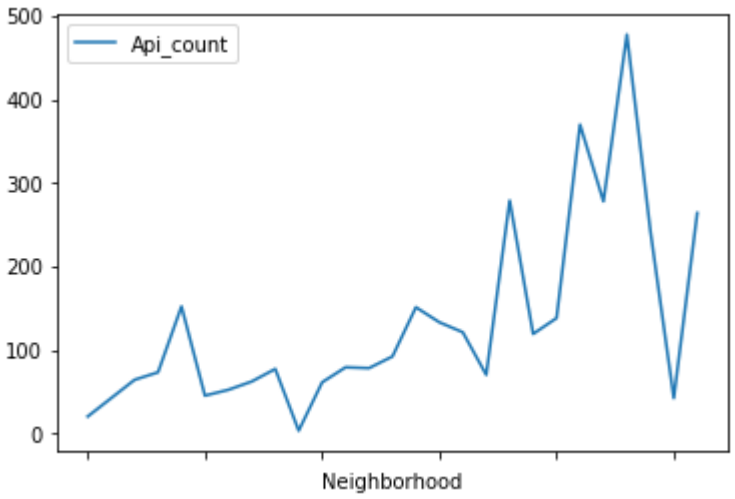
Out[48]:

	Neighborhood	Population	Male	Female	AGE0_4	AGE5_11	AGE12_14	AGE
0	York-Homeland	482.6745	204.898	277.7765	17.2415	24.2415	5.4765	4.95
1	Rosebank	394.0000	181.000	213.0000	19.0000	28.0000	9.0000	7.00
2	Bellona-Gittings	599.0000	286.000	313.0000	38.0000	58.0000	20.0000	21.0
3	Lake Evesham	543.0000	275.000	268.0000	39.0000	47.0000	12.0000	13.0
4	Wilson Park	632.3202	283.463	348.8572	42.6508	54.0926	34.9881	28.5
5	Richnor Springs	676.0000	288.000	388.0000	38.0000	70.0000	26.0000	27.0
6	Radnor-Winston	684.0000	312.000	372.0000	39.0000	72.0000	21.0000	14.0
7	Belvedere	720.0000	339.000	381.0000	39.0000	57.0000	24.0000	25.0
8	Cedarcroft	726.0000	352.000	374.0000	54.0000	101.0000	26.0000	21.0
9	Villages Of Homeland	562.3255	210.102	352.2235	18.7585	22.7585	3.5235	9.04
10	Kernewood	726.0000	330.000	396.0000	46.0000	63.0000	35.0000	34.0
11	Evesham Park	709.0000	334.000	375.0000	60.0000	58.0000	17.0000	10.0
12	Kenilworth Park	1206.0000	546.000	660.0000	74.0000	98.0000	37.0000	51.0
13	Chinquapin Park	1309.0000	584.000	725.0000	75.0000	93.0000	50.0000	47.0
14	Cameron Village	1435.0000	642.000	793.0000	101.0000	127.0000	41.0000	67.0
15	Mid-Govans	1465.0000	661.000	804.0000	103.0000	150.0000	66.0000	77.0
16	Ramblewood	1705.0000	776.000	929.0000	83.0000	140.0000	82.0000	68.0
17	Woodbourne Heights	1795.0000	781.000	1014.0000	119.0000	129.0000	62.0000	57.0
18	Guilford	2214.0000	1099.000	1115.0000	132.0000	218.0000	80.0000	79.0
19	Lake Walker	2095.0000	955.000	1140.0000	174.0000	116.0000	40.0000	32.0
20	Idlewood	2676.0000	1200.000	1476.0000	173.0000	256.0000	114.0000	129.
21	Homeland	3201.0000	1558.000	1643.0000	256.0000	311.0000	101.0000	108.
22	Glen Oaks	2887.0000	1282.000	1605.0000	202.0000	284.0000	117.0000	125.
23	Pen Lucy	3078.0000	1383.000	1695.0000	245.0000	265.0000	117.0000	140.
24	New Northwood	4972.0000	2272.000	2700.0000	286.0000	313.0000	154.0000	231.

	Neighborhood	Population	Male	Female	AGE0_4	AGE5_11	AGE12_14	AGE
25	Loyola/Notre Dame	3208.0000	1195.000	2013.0000	2.0000	0.0000	2.0000	6.00
26	Loch Raven	6163.0000	2557.000	3606.0000	347.0000	474.0000	191.0000	216.

In [49]:
```
df_cen_api.plot.line(x='Neighborhood',y='Api_count')
```

Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0xb143710>



In [50]:
```
df_cen_api_sorted = df_cen_api.sort_values(by=['Young_citizens'])
df_cen_api_sorted
```

Out[50]:

	Neighborhood	Population	Male	Female	AGE0_4	AGE5_11	AGE12_14	AGE
0	York-Homeland	482.6745	204.898	277.7765	17.2415	24.2415	5.4765	4.95
1	Rosebank	394.0000	181.000	213.0000	19.0000	28.0000	9.0000	7.00
2	Bellona-Gittings	599.0000	286.000	313.0000	38.0000	58.0000	20.0000	21.0
3	Lake Evesham	543.0000	275.000	268.0000	39.0000	47.0000	12.0000	13.0
4	Wilson Park	632.3202	283.463	348.8572	42.6508	54.0926	34.9881	28.5
5	Richnor Springs	676.0000	288.000	388.0000	38.0000	70.0000	26.0000	27.0
6	Radnor-Winston	684.0000	312.000	372.0000	39.0000	72.0000	21.0000	14.0
7	Belvedere	720.0000	339.000	381.0000	39.0000	57.0000	24.0000	25.0
8	Cedarcroft	726.0000	352.000	374.0000	54.0000	101.0000	26.0000	21.0
9	Villages Of Homeland	562.3255	210.102	352.2235	18.7585	22.7585	3.5235	9.04
10	Kernewood	726.0000	330.000	396.0000	46.0000	63.0000	35.0000	34.0
11	Evesham Park	709.0000	334.000	375.0000	60.0000	58.0000	17.0000	10.0
12	Kenilworth Park	1206.0000	546.000	660.0000	74.0000	98.0000	37.0000	51.0
13	Chinquapin Park	1309.0000	584.000	725.0000	75.0000	93.0000	50.0000	47.0
14	Cameron Village	1435.0000	642.000	793.0000	101.0000	127.0000	41.0000	67.0
15	Mid-Govans	1465.0000	661.000	804.0000	103.0000	150.0000	66.0000	77.0
16	Ramblewood	1705.0000	776.000	929.0000	83.0000	140.0000	82.0000	68.0
17	Woodbourne Heights	1795.0000	781.000	1014.0000	119.0000	129.0000	62.0000	57.0
18	Guilford	2214.0000	1099.000	1115.0000	132.0000	218.0000	80.0000	79.0
19	Lake Walker	2095.0000	955.000	1140.0000	174.0000	116.0000	40.0000	32.0
20	Idlewood	2676.0000	1200.000	1476.0000	173.0000	256.0000	114.0000	129.
21	Homeland	3201.0000	1558.000	1643.0000	256.0000	311.0000	101.0000	108.
22	Glen Oaks	2887.0000	1282.000	1605.0000	202.0000	284.0000	117.0000	125.
23	Pen Lucy	3078.0000	1383.000	1695.0000	245.0000	265.0000	117.0000	140.
24	New Northwood	4972.0000	2272.000	2700.0000	286.0000	313.0000	154.0000	231.

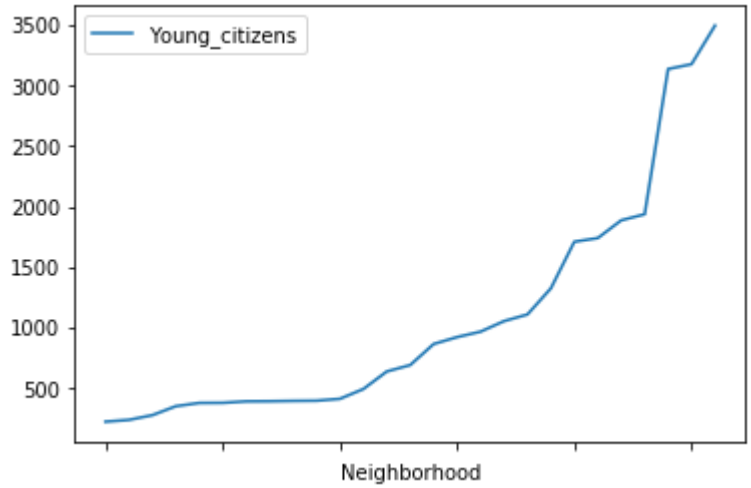
	Neighborhood	Population	Male	Female	AGE0_4	AGE5_11	AGE12_14	AGE
25	Loyola/Notre Dame	3208.0000	1195.000	2013.0000	2.0000	0.0000	2.0000	6.00
26	Loch Raven	6163.0000	2557.000	3606.0000	347.0000	474.0000	191.0000	216.

In [51]:
```
df_cen_api_sorted.Neighborhood
```

Out[51]:
```
0            York-Homeland
1                Rosebank
2         Bellona-Gittings
3             Lake Evesham
4              Wilson Park
5           Richnor Springs
6            Radnor-Winston
7                Belvedere
8               Cedarcroft
9     Villages Of Homeland
10               Kernewood
11            Evesham Park
12          Kenilworth Park
13          Chinquapin Park
14          Cameron Village
15               Mid-Govans
16               Ramblewood
17      Woodbourne Heights
18                 Guilford
19              Lake Walker
20                 Idlewood
21                 Homeland
22                Glen Oaks
23                 Pen Lucy
24           New Northwood
25       Loyola/Notre Dame
26               Loch Raven
Name: Neighborhood, dtype: object
```

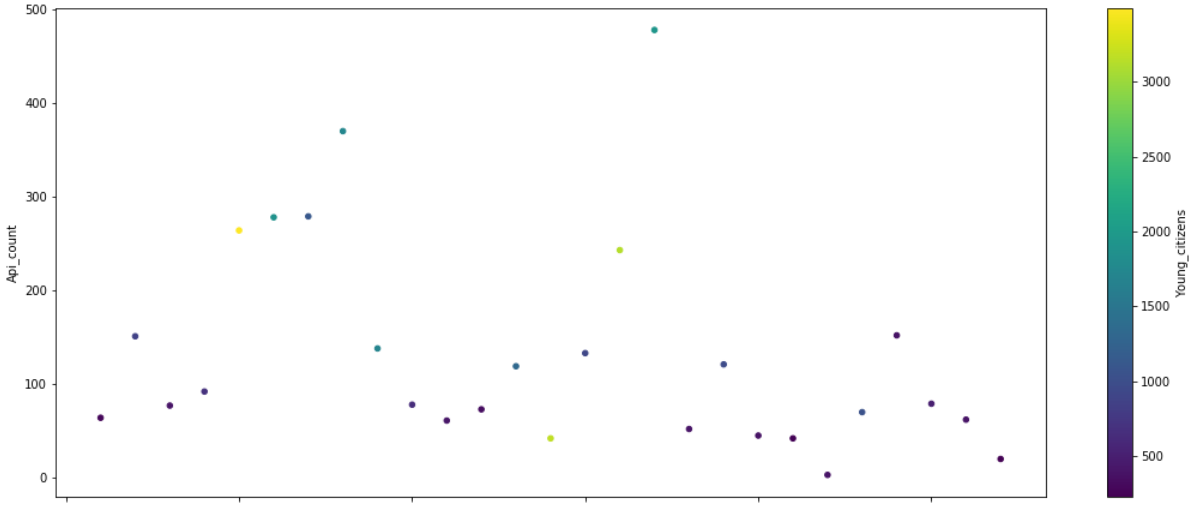
In [52]:
```
df_cen_api_sorted.plot.line(x='Neighborhood',y='Young_citizens')
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0xb1b5898>



This graph shows that their is a relationship between Young age group baltimore citizens in particular neighborhood and number of service requests that are sent using Internet.

In [53]:
```
df_cen_api_sorted.replace(neighborhood_num, inplace = True)
ax2  = df_cen_api_sorted.plot.scatter(x='Neighborhood',
                                      y='Api_count',
                                      c='Young_citizens',
                                      colormap='viridis',
                                      figsize=(18.5,7.5))
```



11. Exploring which neighborhoods in Balrimore are safe, moderate or dangerous based on crimes.

In [54]:
```
df_crime = df_crime.groupby(["Neighborhood"]).size()
type(df_crime)
df_crime
```

Out[54]:

Neighborhood	
Abell	31
Allendale	67
Arcadia	21
Arlington	101
Armistead Gardens	60
Ashburton	33
Baltimore Highlands	143
Barclay	93
Barre Circle	13
Bayview	50
Beechfield	54
Belair-Edison	380
Belair-Parkside	11
Belvedere	10
Berea	124
Better Waverly	126
Beverly Hills	5
Biddle Street	49
Bolton Hill	62
Boyd-Booth	25
Brewers Hill	19
Bridgeview/Greenlawn	58
Broadway East	234
Broening Manor	52
Brooklyn	315
Burleith-Leighton	18
Butcher's Hill	43
CARE	55
Callaway-Garrison	34
Cameron Village	38
...	...
Upper Fells Point	94
Upton	322
Villages Of Homeland	2
Violetville	62
Wakefield	35
Walbrook	79
Waltherson	77
Washington Hill	78
Washington Village/Pigtow	166
Waverly	84
West Arlington	36
West Forest Park	40
West Hills	17
Westfield	43
Westgate	43
Westport	80
Wilhelm Park	7
Wilson Park	18
Winchester	40
Windsor Hills	28
Winston-Govans	37
Woodberry	16
Woodbourne Heights	31
Woodbourne-McCabe	25
Woodmere	109

```
Wrenlane                    17
Wyman Park                   7
Wyndhurst                    3
Yale Heights                47
York-Homeland               12
Length: 270, dtype: int64
```

In [55]:
```
df_crime = pd.Series.to_frame(df_crime)
type(df_crime)
```

Out[55]:
```
pandas.core.frame.DataFrame
```

In [56]:
```
df_cen_api_crime = pd.merge(df_cen_api,df_crime, on='Neighborhood')
df_cen_api_crime = df_cen_api_crime.rename(columns={0:"Crimes"})
```

In [57]:
```
df_cen_api_crime
```

Out[57]:

	Neighborhood	Population	Male	Female	AGE0_4	AGE5_11	AGE12_14	AGE
0	York-Homeland	482.6745	204.898	277.7765	17.2415	24.2415	5.4765	4.95
1	Rosebank	394.0000	181.000	213.0000	19.0000	28.0000	9.0000	7.00
2	Lake Evesham	543.0000	275.000	268.0000	39.0000	47.0000	12.0000	13.0
3	Wilson Park	632.3202	283.463	348.8572	42.6508	54.0926	34.9881	28.5
4	Richnor Springs	676.0000	288.000	388.0000	38.0000	70.0000	26.0000	27.0
5	Radnor-Winston	684.0000	312.000	372.0000	39.0000	72.0000	21.0000	14.0
6	Belvedere	720.0000	339.000	381.0000	39.0000	57.0000	24.0000	25.0
7	Cedarcroft	726.0000	352.000	374.0000	54.0000	101.0000	26.0000	21.0
8	Villages Of Homeland	562.3255	210.102	352.2235	18.7585	22.7585	3.5235	9.04
9	Kernewood	726.0000	330.000	396.0000	46.0000	63.0000	35.0000	34.0
10	Evesham Park	709.0000	334.000	375.0000	60.0000	58.0000	17.0000	10.0
11	Kenilworth Park	1206.0000	546.000	660.0000	74.0000	98.0000	37.0000	51.0
12	Chinquapin Park	1309.0000	584.000	725.0000	75.0000	93.0000	50.0000	47.0
13	Cameron Village	1435.0000	642.000	793.0000	101.0000	127.0000	41.0000	67.0
14	Mid-Govans	1465.0000	661.000	804.0000	103.0000	150.0000	66.0000	77.0
15	Ramblewood	1705.0000	776.000	929.0000	83.0000	140.0000	82.0000	68.0
16	Woodbourne Heights	1795.0000	781.000	1014.0000	119.0000	129.0000	62.0000	57.0
17	Guilford	2214.0000	1099.000	1115.0000	132.0000	218.0000	80.0000	79.0
18	Lake Walker	2095.0000	955.000	1140.0000	174.0000	116.0000	40.0000	32.0
19	Idlewood	2676.0000	1200.000	1476.0000	173.0000	256.0000	114.0000	129.
20	Homeland	3201.0000	1558.000	1643.0000	256.0000	311.0000	101.0000	108.
21	Glen Oaks	2887.0000	1282.000	1605.0000	202.0000	284.0000	117.0000	125.
22	Pen Lucy	3078.0000	1383.000	1695.0000	245.0000	265.0000	117.0000	140.
23	New Northwood	4972.0000	2272.000	2700.0000	286.0000	313.0000	154.0000	231.
24	Loyola/Notre Dame	3208.0000	1195.000	2013.0000	2.0000	0.0000	2.0000	6.00

	Neighborhood	Population	Male	Female	AGE0_4	AGE5_11	AGE12_14	AGE
25	Loch Raven	6163.0000	2557.000	3606.0000	347.0000	474.0000	191.0000	216.

In [58]:
```
df_services = df_services.rename(columns={0:"Services"})
df_services
```

Out[58]:

	Services
Neighborhood	
Bellona-Gittings	162
Belvedere	192
Cameron Village	648
Cedarcroft	200
Chinquapin Park	573
Evesham Park	247
Glen Oaks	1038
Guilford	1082
Homeland	1012
Idlewood	765
Kenilworth Park	431
Kernewood	382
Lake Evesham	182
Lake Walker	348
Loch Raven	1604
Loyola/Notre Dame	121
Mid-Govans	602
New Northwood	1337
Pen Lucy	1527
Radnor-Winston	198
Ramblewood	754
Richnor Springs	347
Rosebank	219
Villages Of Homeland	14
Wilson Park	490
Woodbourne Heights	370
York-Homeland	98

In [59]:
```
df_cen_api_crime = pd.merge(df_cen_api_crime,df_services, on='Neighborhood')
```

In [60]:
```
df_cen_api_crime
```

Out[60]:

	Neighborhood	Population	Male	Female	AGE0_4	AGE5_11	AGE12_14	AGE
0	York-Homeland	482.6745	204.898	277.7765	17.2415	24.2415	5.4765	4.95
1	Rosebank	394.0000	181.000	213.0000	19.0000	28.0000	9.0000	7.00
2	Lake Evesham	543.0000	275.000	268.0000	39.0000	47.0000	12.0000	13.0
3	Wilson Park	632.3202	283.463	348.8572	42.6508	54.0926	34.9881	28.5
4	Richnor Springs	676.0000	288.000	388.0000	38.0000	70.0000	26.0000	27.0
5	Radnor-Winston	684.0000	312.000	372.0000	39.0000	72.0000	21.0000	14.0
6	Belvedere	720.0000	339.000	381.0000	39.0000	57.0000	24.0000	25.0
7	Cedarcroft	726.0000	352.000	374.0000	54.0000	101.0000	26.0000	21.0
8	Villages Of Homeland	562.3255	210.102	352.2235	18.7585	22.7585	3.5235	9.04
9	Kernewood	726.0000	330.000	396.0000	46.0000	63.0000	35.0000	34.0
10	Evesham Park	709.0000	334.000	375.0000	60.0000	58.0000	17.0000	10.0
11	Kenilworth Park	1206.0000	546.000	660.0000	74.0000	98.0000	37.0000	51.0
12	Chinquapin Park	1309.0000	584.000	725.0000	75.0000	93.0000	50.0000	47.0
13	Cameron Village	1435.0000	642.000	793.0000	101.0000	127.0000	41.0000	67.0
14	Mid-Govans	1465.0000	661.000	804.0000	103.0000	150.0000	66.0000	77.0
15	Ramblewood	1705.0000	776.000	929.0000	83.0000	140.0000	82.0000	68.0
16	Woodbourne Heights	1795.0000	781.000	1014.0000	119.0000	129.0000	62.0000	57.0
17	Guilford	2214.0000	1099.000	1115.0000	132.0000	218.0000	80.0000	79.0
18	Lake Walker	2095.0000	955.000	1140.0000	174.0000	116.0000	40.0000	32.0
19	Idlewood	2676.0000	1200.000	1476.0000	173.0000	256.0000	114.0000	129.
20	Homeland	3201.0000	1558.000	1643.0000	256.0000	311.0000	101.0000	108.
21	Glen Oaks	2887.0000	1282.000	1605.0000	202.0000	284.0000	117.0000	125.
22	Pen Lucy	3078.0000	1383.000	1695.0000	245.0000	265.0000	117.0000	140.
23	New Northwood	4972.0000	2272.000	2700.0000	286.0000	313.0000	154.0000	231.
24	Loyola/Notre Dame	3208.0000	1195.000	2013.0000	2.0000	0.0000	2.0000	6.00

	Neighborhood	Population	Male	Female	AGE0_4	AGE5_11	AGE12_14	AGE
25	Loch Raven	6163.0000	2557.000	3606.0000	347.0000	474.0000	191.0000	216.

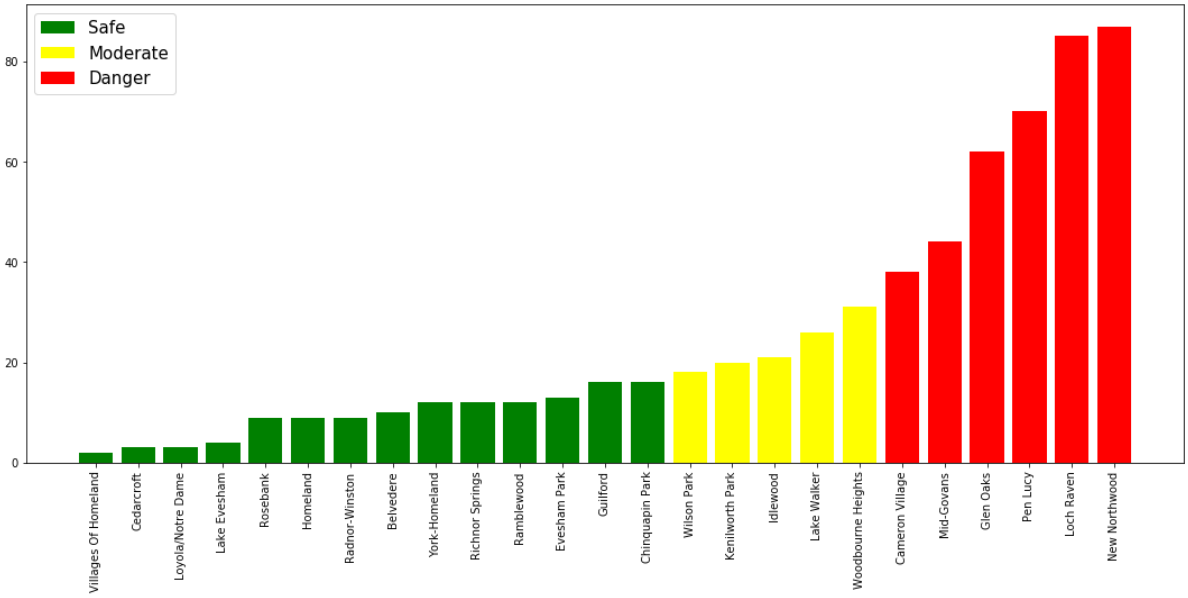
Ploting the bar graph that shows the neighborhoods which are safe, moderately safe and dangerous based on the crime rate at every neighborhood.

In [61]:
```
import matplotlib.patches as mpatches

df_cen_api_crime = df_cen_api_crime.sort_values(by=['Crimes'])
x= df_cen_api_crime['Neighborhood']
y = df_cen_api_crime['Crimes']
filter1 = y<17
filter2 = y>17
filter3 = y > 35

Safety = ['Safe', 'Moderate', 'Danger']

plt.figure(figsize=(18.5,7.5))
plt.xticks(rotation=90)
plt.bar(x[filter1],y[filter1], color='green')
plt.bar(x[filter2],y[filter2], color='yellow')
plt.bar(x[filter3],y[filter3], color='red')
plt.legend(Safety, prop={'size':15})
plt.show()
#df_cen_api_crime.plot.bar(x='Neighborhood', y='Crimes', figsize=(18.5,7.5))
```



12. Exploring different types of methods in which the requests are received based on neighborhood

In [62]:
```
api = df_service[ (df_service.MethodReceived == 'API')]
api = api.groupby('Neighborhood').size()
api = pd.Series.to_frame(api)
api_sorted = api.sort_values(by=['Neighborhood'])
api_sorted = api_sorted.rename(columns={0:"Api_count"})
type(api_sorted)
```

Out[62]:
```
pandas.core.frame.DataFrame
```

In [63]:
```
api_sorted
```

Out[63]:

Neighborhood	Api_count
Bellona-Gittings	64
Belvedere	62
Cameron Village	151
Cedarcroft	77
Chinquapin Park	92
Evesham Park	79
Glen Oaks	278
Guilford	279
Homeland	370
Idlewood	138
Kenilworth Park	78
Kernewood	61
Lake Evesham	73
Lake Walker	119
Loch Raven	264
Loyola/Notre Dame	42
Mid-Govans	133
New Northwood	243
Pen Lucy	478
Radnor-Winston	52
Ramblewood	121
Richnor Springs	45
Rosebank	42
Villages Of Homeland	3
Wilson Park	152
Woodbourne Heights	70
York-Homeland	20

Only citizen belonging to Pen Lucy neighborhood has used Email to raise service request

In [64]:
```
email = df_service[(df_service.MethodReceived == 'Email')]
email = email.groupby('Neighborhood').size()
email = pd.Series.to_frame(email)
email_sorted = email.sort_values(by=['Neighborhood'])
email_sorted = email_sorted.rename(columns={0:"Email_count"})
email_sorted
```

Out[64]:

	Email_count
Neighborhood	
Pen Lucy	1

In [65]:
```
Internal = df_service[ (df_service.MethodReceived == 'Internal')]
Internal = Internal.groupby('Neighborhood').size()
Internal = pd.Series.to_frame(Internal)
Internal_sorted = Internal.sort_values(by=['Neighborhood'])
Internal_sorted = Internal_sorted.rename(columns={0:"Internal_count"})
Internal_sorted
```

Out[65]:

	Internal_count
Neighborhood	
Bellona-Gittings	6
Belvedere	16
Cameron Village	83
Cedarcroft	23
Chinquapin Park	82
Evesham Park	27
Glen Oaks	132
Guilford	188
Homeland	177
Idlewood	72
Kenilworth Park	69
Kernewood	60
Lake Evesham	18
Lake Walker	17
Loch Raven	111
Loyola/Notre Dame	8
Mid-Govans	56
New Northwood	167
Pen Lucy	115
Radnor-Winston	89
Ramblewood	93
Richnor Springs	38
Rosebank	64
Wilson Park	72
Woodbourne Heights	30
York-Homeland	29

In [66]:
```
Phone = df_service[ (df_service.MethodReceived == 'Phone')]
Phone = Phone.groupby('Neighborhood').size()
Phone = pd.Series.to_frame(Phone)
Phone_sorted = Phone.sort_values(by=['Neighborhood'])
Phone_sorted = Phone_sorted.rename(columns={0:"Phone_count"})
Phone_sorted
```

Out[66]:

	Phone_count
Neighborhood	
Bellona-Gittings	92
Belvedere	114
Cameron Village	414
Cedarcroft	100
Chinquapin Park	399
Evesham Park	141
Glen Oaks	628
Guilford	615
Homeland	465
Idlewood	554
Kenilworth Park	284
Kernewood	261
Lake Evesham	91
Lake Walker	212
Loch Raven	1229
Loyola/Notre Dame	71
Mid-Govans	413
New Northwood	927
Pen Lucy	933
Radnor-Winston	57
Ramblewood	540
Richnor Springs	264
Rosebank	113
Villages Of Homeland	11
Wilson Park	265
Woodbourne Heights	270
York-Homeland	49

In [67]:
```
Other = df_service[ (df_service.MethodReceived == 'Other')]
Other = Other.groupby('Neighborhood').size()
Other = pd.Series.to_frame(Other)
Other_sorted = Other.sort_values(by=['Neighborhood'])
Other_sorted = Other_sorted.rename(columns={0:"Other_count"})
Other_sorted
```

Out[67]:

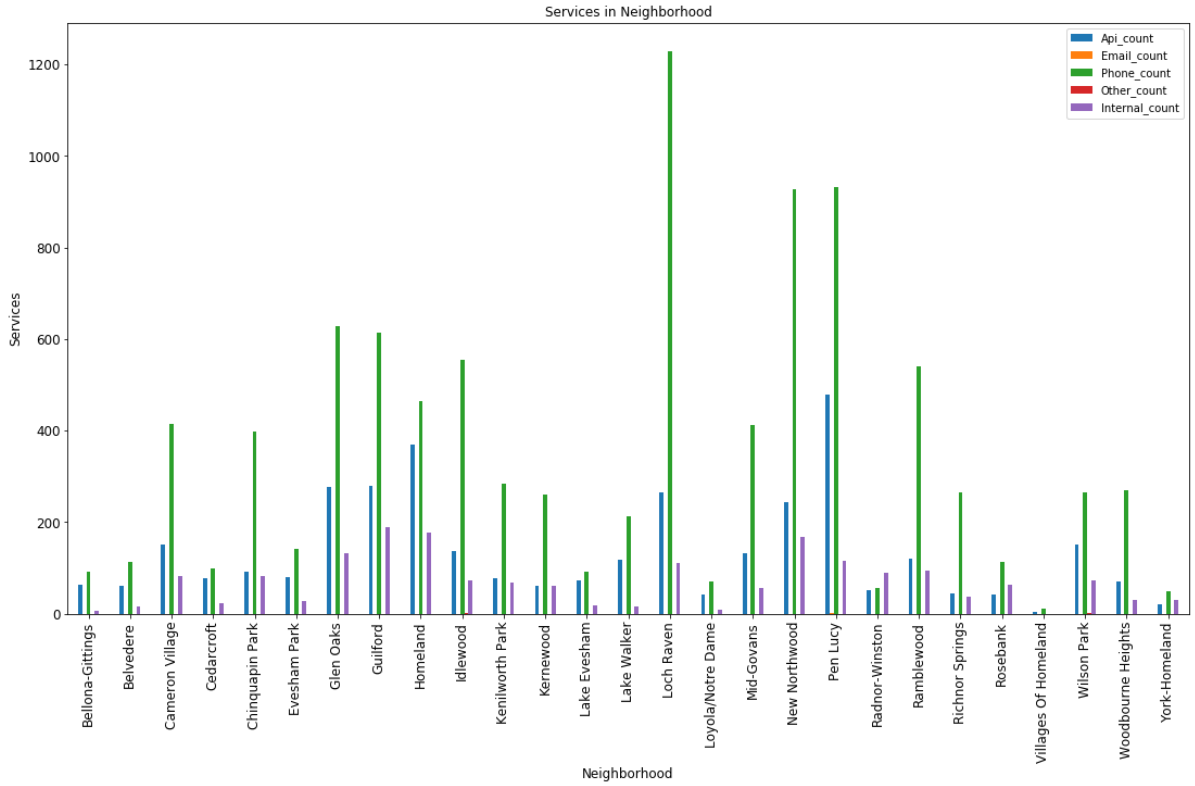
	Other_count
Neighborhood	
Idlewood	1
Wilson Park	1

In [68]:
```
df_service_merge = pd.merge(api_sorted,email_sorted, on='Neighborhood', how='o
uter')
df_service_merge = pd.merge(df_service_merge,Phone_sorted, on='Neighborhood',
how='outer')
df_service_merge = pd.merge(df_service_merge,Other_sorted, on='Neighborhood',
how='outer')
df_service_merge = pd.merge(df_service_merge,Internal_sorted, on='Neighborhoo
d', how='outer')
df_service_merge = df_service_merge.fillna(0)
df_service_merge
```

Out[68]:

	Api_count	Email_count	Phone_count	Other_count	Internal_count
Neighborhood					
Bellona-Gittings	64	0.0	92	0.0	6.0
Belvedere	62	0.0	114	0.0	16.0
Cameron Village	151	0.0	414	0.0	83.0
Cedarcroft	77	0.0	100	0.0	23.0
Chinquapin Park	92	0.0	399	0.0	82.0
Evesham Park	79	0.0	141	0.0	27.0
Glen Oaks	278	0.0	628	0.0	132.0
Guilford	279	0.0	615	0.0	188.0
Homeland	370	0.0	465	0.0	177.0
Idlewood	138	0.0	554	1.0	72.0
Kenilworth Park	78	0.0	284	0.0	69.0
Kernewood	61	0.0	261	0.0	60.0
Lake Evesham	73	0.0	91	0.0	18.0
Lake Walker	119	0.0	212	0.0	17.0
Loch Raven	264	0.0	1229	0.0	111.0
Loyola/Notre Dame	42	0.0	71	0.0	8.0
Mid-Govans	133	0.0	413	0.0	56.0
New Northwood	243	0.0	927	0.0	167.0
Pen Lucy	478	1.0	933	0.0	115.0
Radnor-Winston	52	0.0	57	0.0	89.0
Ramblewood	121	0.0	540	0.0	93.0
Richnor Springs	45	0.0	264	0.0	38.0
Rosebank	42	0.0	113	0.0	64.0
Villages Of Homeland	3	0.0	11	0.0	0.0
Wilson Park	152	0.0	265	1.0	72.0
Woodbourne Heights	70	0.0	270	0.0	30.0
York-Homeland	20	0.0	49	0.0	29.0

In [69]:
```
ax = df_service_merge[['Api_count','Email_count','Phone_count','Other_count',
'Internal_count']].plot(kind='bar', title ="Services in Neighborhood", figsize
=(18.5, 10), legend=True, fontsize=12)
ax.set_xlabel("Neighborhood", fontsize=12)
ax.set_ylabel("Services", fontsize=12)
plt.show()
```



We can conclude from above graph that majority of service requests comes from Loch Raven and New Northwood where as Village of Homelands shows the least number of Service requests.

13. District-wise distribution of topmost Crime Codes and their corresponding neighborhoods

In [70]:
```
df_crimecode = df3.groupby('CrimeCode').size()
df_crimecode = pd.Series.to_frame(df_crimecode)
df_crimecode = df_crimecode.rename(columns={0: 'CountOfCrimeCode'})
df_crimecode = df_crimecode.sort_values(by=['CountOfCrimeCode'], ascending = False)
df_crimecode.head(5)
```

Out[70]:

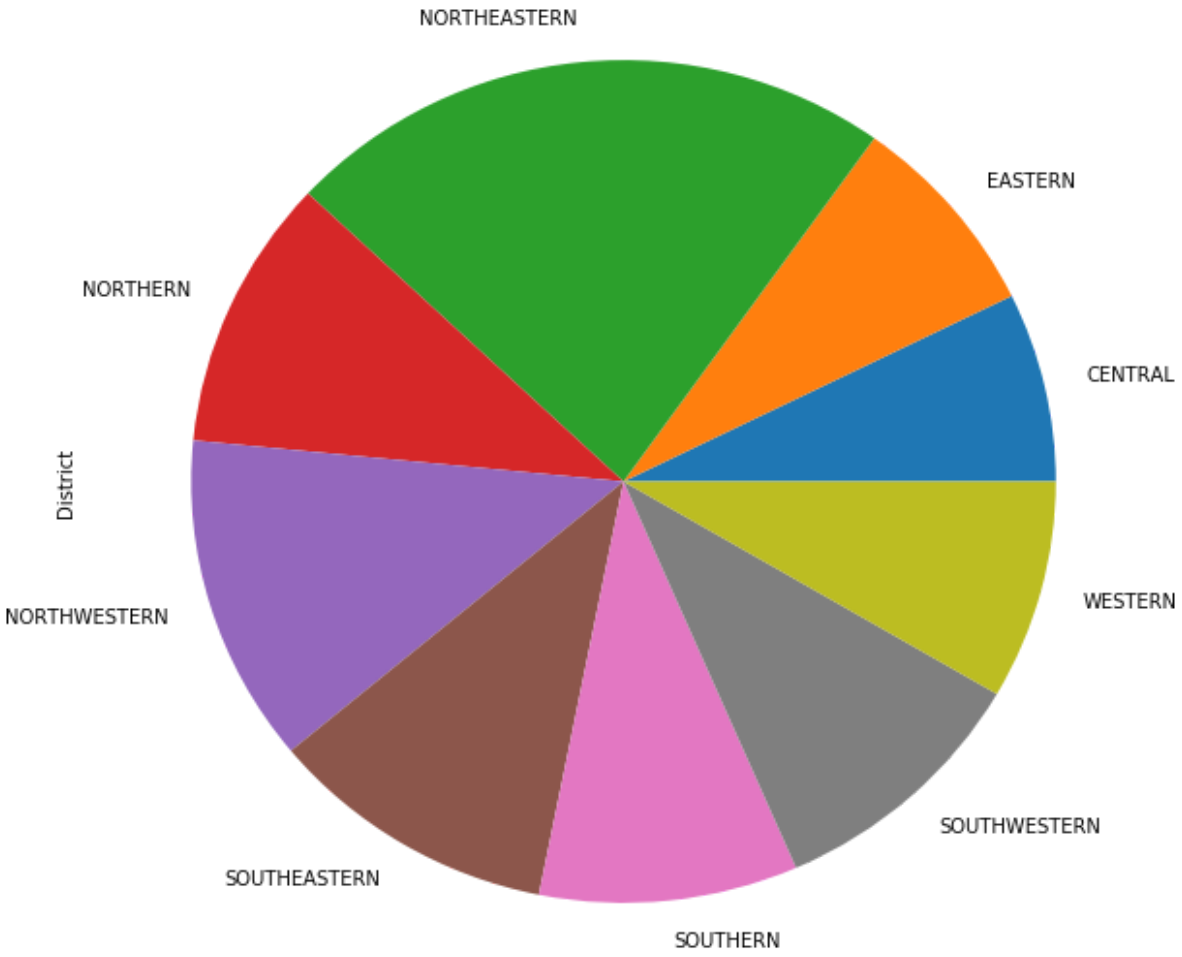
	CountOfCrimeCode
CrimeCode	
4E	9257
6D	7032
5A	4719
7A	3472
6G	3292

In [71]:
```
df_crime_district = df3[['CrimeCode','District']]
df_crime_district = df_crime_district[(df_crime_district['CrimeCode'] == '7A')]
df_crime_district.head()
```

Out[71]:

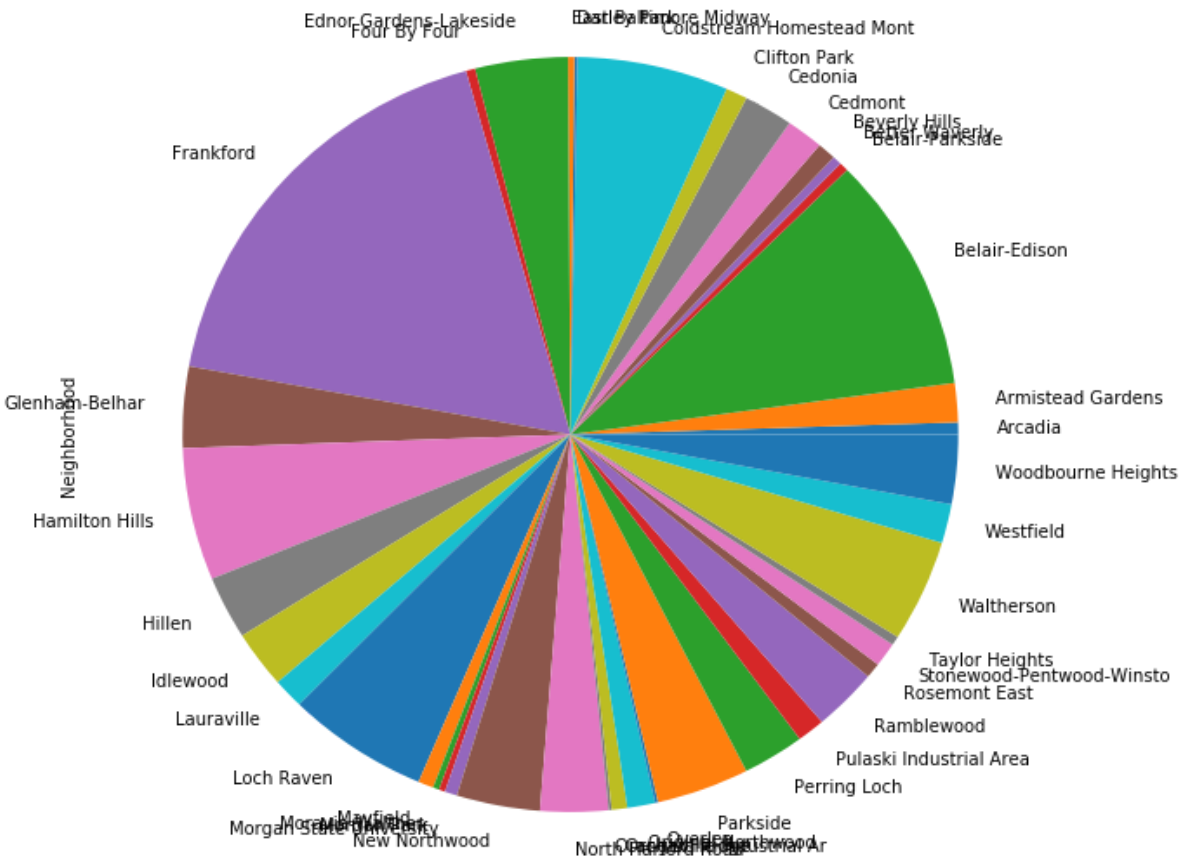
	CrimeCode	District
73	7A	SOUTHWESTERN
81	7A	WESTERN
97	7A	CENTRAL
100	7A	NORTHWESTERN
131	7A	NORTHEASTERN

In [92]:
```
fig, ax = plt.subplots(figsize=(10,10))
CrimeInDistrict = df_crime_district.groupby('District')['District'].count().pl
ot.pie(subplots=True)
```



In [93]:
```
df_crime_district_neighborhood = df3[['CrimeCode','District','Neighborhood']]
df_crime_district_neighborhood = df_crime_district_neighborhood[(df_crime_dist
rict_neighborhood['CrimeCode'] == '7A')]
df_crime_district_neighborhood = df_crime_district_neighborhood[(df_crime_dist
rict_neighborhood['District'] == 'NORTHEASTERN')]

fig, ax = plt.subplots(figsize=(10,10))
df_crime_district_neighborhood = df_crime_district_neighborhood.groupby('Neigh
borhood')['Neighborhood'].count().plot.pie(subplots=True)
```

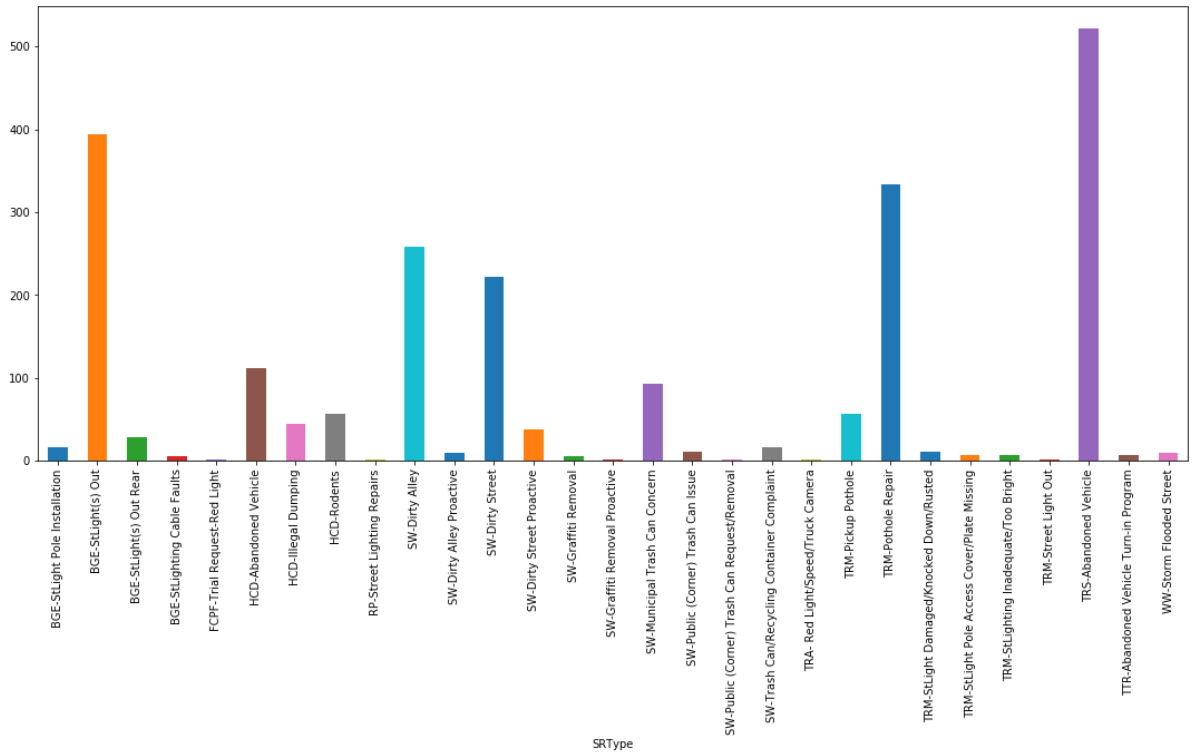


Since Loch Raven shows highest number of Crime rate as well as highest number of service requsts, let's dig more into the type of service requests they encounter.

In [94]:
```
df_service_method = df[['SRType','Neighborhood']]
df_service_method = df_service_method[(df_service_method['Neighborhood'] == 'L
och Raven')]
df_service_method = df_service_method.groupby('SRType').size()


df_service_method.plot.bar(figsize=(18.5,7.5))
```

Out[94]: <matplotlib.axes._subplots.AxesSubplot at 0x21eca9926a0>



Above graph shows that the Loch raven raises Abandoned Vehicles and Street lights out service requests the most.

14. Displaying direct relationship between the number of service requests and the crimes

In [76]:
```
response = df_service.groupby(['Neighborhood'])['DaysRequired'].mean()
response = pd.Series.to_frame(response)
```

In [77]:
```
df_cen_api_crime_days = df_cen_api_crime[['Neighborhood','Population','Young_c
itizens','Senior_citizens','Api_count','Crimes','Services']]
```

In [78]:
```
df_cen_api_crime_days = pd.merge(df_cen_api_crime_days,response, on='Neighborh
ood')
```

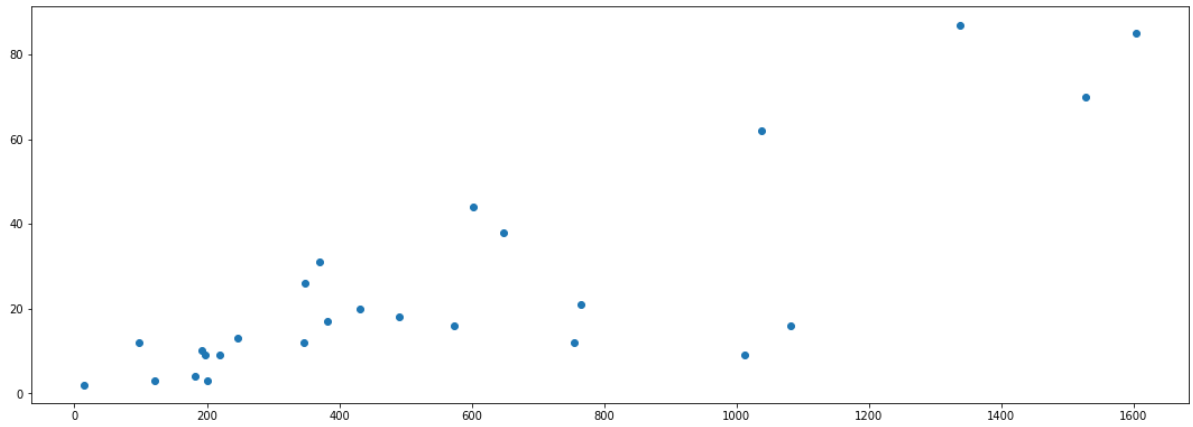
In [79]:
```
df_cen_api_crime_days.sort_values(by=['Crimes'])
```

Out[79]:

	Neighborhood	Population	Young_citizens	Senior_citizens	Api_count	Crimes	Serv
0	Villages Of Homeland	562.3255	396.6805	165.6450	3	2	14
1	Cedarcroft	726.0000	395.0000	331.0000	77	3	200
2	Loyola/Notre Dame	3208.0000	3177.0000	31.0000	42	3	121
3	Lake Evesham	543.0000	351.0000	192.0000	73	4	182
4	Rosebank	394.0000	238.0000	156.0000	42	9	219
5	Homeland	3201.0000	1740.0000	1461.0000	370	9	1012
6	Radnor-Winston	684.0000	391.0000	293.0000	52	9	198
7	Belvedere	720.0000	392.0000	328.0000	62	10	192
8	York-Homeland	482.6745	223.3195	259.3550	20	12	98
9	Richnor Springs	676.0000	379.0000	297.0000	45	12	347
10	Ramblewood	1705.0000	967.0000	738.0000	121	12	754
11	Evesham Park	709.0000	493.0000	216.0000	79	13	247
13	Chinquapin Park	1309.0000	691.0000	618.0000	92	16	573
12	Guilford	2214.0000	1109.0000	1105.0000	279	16	1082
14	Kernewood	726.0000	412.0000	314.0000	61	17	382
15	Wilson Park	632.3202	377.9855	254.3347	152	18	490
16	Kenilworth Park	1206.0000	638.0000	568.0000	78	20	431
17	Idlewood	2676.0000	1710.0000	966.0000	138	21	765
18	Lake Walker	2095.0000	1325.0000	770.0000	119	26	348
19	Woodbourne Heights	1795.0000	1054.0000	741.0000	70	31	370
20	Cameron Village	1435.0000	865.0000	570.0000	151	38	648
21	Mid-Govans	1465.0000	922.0000	543.0000	133	44	602
22	Glen Oaks	2887.0000	1887.0000	1000.0000	278	62	1038
23	Pen Lucy	3078.0000	1937.0000	1141.0000	478	70	1527
24	Loch Raven	6163.0000	3496.0000	2667.0000	264	85	1604

	Neighborhood	Population	Young_citizens	Senior_citizens	Api_count	Crimes	Serv
25	New Northwood	4972.0000	3137.0000	1835.0000	243	87	1337

In [95]:
```
plt.figure(figsize=(18.5,6.5))
plt.scatter(df_cen_api_crime_days['Services'], df_cen_api_crime_days['Crimes'
])
plt.show()
```



From the above graph we can see that the Services requested are directly proportional to the crimes in a neighborhood. It tells us that as the crime in an area increases, the request raised for services increases.

15. Graphs we tried but could not find any relationship

Since we found above that services are directly proportional to crime, we tried to find relation between crimes and the days required to resolve the service request. We were expecting a graph which shows that the particular neighborhood has more crimes as the services are not resolved on time. But the actual output did not show this expected result.

In [80]:
```
df3.dropna()
```

Out[80]:

	CrimeDate	CrimeCode	Location	Description	Weapon	Dist
0	1/1/2012	4A	500 CHATEAU AV	AGG. ASSAULT	FIREARM	NORTHERN
1	1/1/2012	4E	2200 W NORTH AV	COMMON ASSAULT	HANDS	WESTERN
2	1/1/2012	4E	0 S LINWOOD AV	COMMON ASSAULT	HANDS	SOUTHEASTE
6	1/1/2012	4E	500 N GLOVER ST	COMMON ASSAULT	HANDS	SOUTHEASTE
7	1/1/2012	4E	4900 GOODNOW RD	COMMON ASSAULT	HANDS	NORTHEASTE
11	1/1/2012	4C	500 LONEYS LA	AGG. ASSAULT	OTHER	SOUTHEASTE
12	1/1/2012	4E	1500 GORSUCH AV	COMMON ASSAULT	HANDS	NORTHEASTE
14	1/1/2012	4C	400 E BALTIMORE ST	AGG. ASSAULT	OTHER	CENTRAL
18	1/1/2012	4E	300 N HIGHLAND AV	COMMON ASSAULT	HANDS	SOUTHEASTE
20	1/1/2012	4C	2000 GRINNALDS AV	AGG. ASSAULT	OTHER	SOUTHWESTE
21	1/1/2012	4E	500 N ELLWOOD AV	COMMON ASSAULT	HANDS	SOUTHEASTE
27	1/1/2012	4B	1900 E NORTH AV	AGG. ASSAULT	KNIFE	EASTERN
28	1/1/2012	3CF	1800 RUSSELL ST	ROBBERY - COMMERCIAL	FIREARM	SOUTHERN
34	1/1/2012	4C	0 E RANDALL ST	AGG. ASSAULT	OTHER	SOUTHERN
36	1/1/2012	4E	1000 N WOLFE ST	COMMON ASSAULT	HANDS	EASTERN
37	1/1/2012	4E	1000 N WOLFE ST	COMMON ASSAULT	HANDS	EASTERN
38	1/1/2012	4E	2800 CARVER RD	COMMON ASSAULT	HANDS	SOUTHERN

	CrimeDate	CrimeCode	Location	Description	Weapon	Dist
41	1/1/2012	9S	500 OAKLAND AV	SHOOTING	FIREARM	NORTHERN
42	1/1/2012	4A	500 OAKLAND AVE	AGG. ASSAULT	FIREARM	NORTHERN
45	1/1/2012	4E	400 N ROSE ST	COMMON ASSAULT	HANDS	SOUTHEASTE
46	1/1/2012	4C	5700 PLAINFIELD AV	AGG. ASSAULT	OTHER	NORTHEASTE
48	1/1/2012	4C	2100 BAKER ST	AGG. ASSAULT	OTHER	WESTERN
50	1/1/2012	4E	2600 GREENMOUNT AV	COMMON ASSAULT	HANDS	NORTHERN
52	1/1/2012	4E	1800 W LAFAYETTE AV	COMMON ASSAULT	HANDS	WESTERN
61	1/1/2012	3JF	700 N FULTON AV	ROBBERY - RESIDENCE	FIREARM	WESTERN
64	1/1/2012	4C	400 BOULDIN ST	AGG. ASSAULT	OTHER	SOUTHEASTE
65	1/1/2012	4E	0 N GAY ST	COMMON ASSAULT	HANDS	CENTRAL
66	1/1/2012	4E	4000 EIERMAN AV	COMMON ASSAULT	HANDS	NORTHEASTE
67	1/1/2012	4E	1100 KEVIN RD	COMMON ASSAULT	HANDS	SOUTHWESTE
69	1/1/2012	4E	800 BETHUNE RD	COMMON ASSAULT	HANDS	SOUTHERN
...	...	...	...	...	...	...
49462	12/31/2012	3AF	3700 ELLERSLIE AV	ROBBERY - STREET	FIREARM	NORTHERN
49463	12/31/2012	3AF	3700 ELLERSLIE AV	ROBBERY - STREET	FIREARM	NORTHERN
49467	12/31/2012	4E	2400 FREDERICK AV	COMMON ASSAULT	HANDS	SOUTHWESTE
49468	12/31/2012	3AK	0 E HENRIETTA ST	ROBBERY - STREET	KNIFE	SOUTHERN
49474	12/31/2012	4E	800 PARK AV	COMMON ASSAULT	HANDS	CENTRAL

	CrimeDate	CrimeCode	Location	Description	Weapon	Dist
49475	12/31/2012	3CF	4100 NORFOLK AV	ROBBERY - COMMERCIAL	FIREARM	NORTHWESTE
49481	12/31/2012	1F	2100 BRYANT AV	HOMICIDE	FIREARM	WESTERN
49483	12/31/2012	4E	1300 HARLEM AV	COMMON ASSAULT	HANDS	WESTERN
49488	12/31/2012	4E	200 N HOWARD ST	COMMON ASSAULT	HANDS	CENTRAL
49490	12/31/2012	4E	1200 ROSSITER AV	COMMON ASSAULT	HANDS	NORTHEASTE
49491	12/31/2012	4C	0 S ABINGTON AV	AGG. ASSAULT	OTHER	SOUTHWESTE
49492	12/31/2012	4C	2600 MARBOURNE AV	AGG. ASSAULT	OTHER	SOUTHERN
49496	12/31/2012	4A	5600 WINNER AV	AGG. ASSAULT	FIREARM	NORTHWESTE
49501	12/31/2012	4E	300 S PULASKI ST	COMMON ASSAULT	HANDS	SOUTHWESTE
49503	12/31/2012	4E	1200 LINWOOD AVE	COMMON ASSAULT	HANDS	EASTERN
49504	12/31/2012	4E	300 STOCKHOLM ST	COMMON ASSAULT	HANDS	SOUTHERN
49505	12/31/2012	4E	300 STOCKHOLM ST	COMMON ASSAULT	HANDS	SOUTHERN
49509	12/31/2012	4E	5100 GOODNOW RD	COMMON ASSAULT	HANDS	NORTHEASTE
49519	12/31/2012	4C	200 MONTFORD AVE	AGG. ASSAULT	OTHER	SOUTHEASTE
49520	12/31/2012	4E	4900 MIDWOOD AV	COMMON ASSAULT	HANDS	NORTHERN
49521	12/31/2012	4E	900 DANTREY CT	COMMON ASSAULT	HANDS	SOUTHERN
49522	12/31/2012	4E	900 DANTREY CT	COMMON ASSAULT	HANDS	SOUTHERN

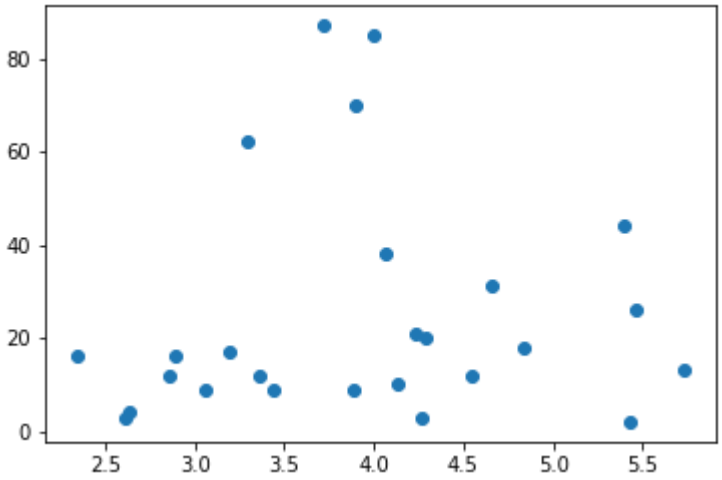
	CrimeDate	CrimeCode	Location	Description	Weapon	Dist
49528	12/31/2012	4E	2000 WILKINS AVE	COMMON ASSAULT	HANDS	SOUTHERN
49535	12/31/2012	4E	2000 WILKENS AV	COMMON ASSAULT	HANDS	SOUTHERN
49539	12/31/2012	4E	2800 HINSDALE DR	COMMON ASSAULT	HANDS	SOUTHERN
49540	12/31/2012	4E	3500 CHESTERFIELD AV	COMMON ASSAULT	HANDS	NORTHEASTE
49544	12/31/2012	3AF	6000 NORTHWOOD DR	ROBBERY - STREET	FIREARM	NORTHERN
49545	12/31/2012	3AF	3800 W FOREST PARK AV	ROBBERY - STREET	FIREARM	NORTHWESTE
49546	12/31/2012	4B	1000 CATHEDRAL ST	AGG. ASSAULT	KNIFE	CENTRAL
49551	12/31/2012	4E	200 S DALLAS CT	COMMON ASSAULT	HANDS	SOUTHEASTE

17141 rows × 7 columns

Below graph is plotted against Days required to solve a service request and Crimes

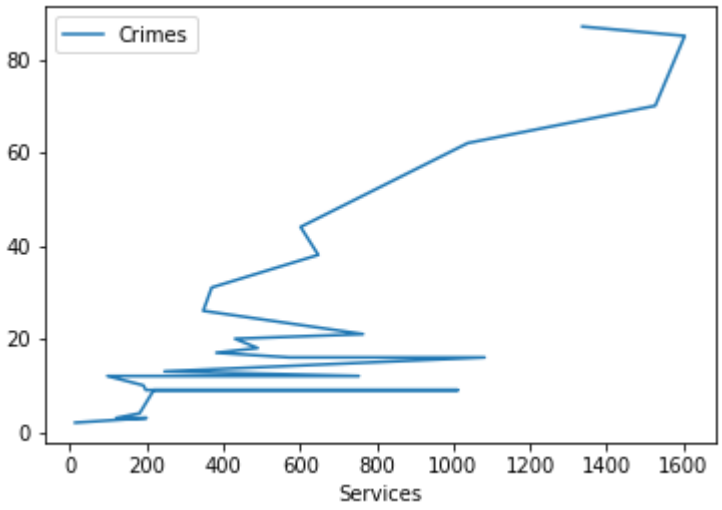
In [81]:
```
#df_cen_api_crime_days.scatter(x='DaysRequired',y='Crimes')

plt.scatter(df_cen_api_crime_days['DaysRequired'], df_cen_api_crime_days['Crimes'])
plt.show()
```



In [82]:
```
df_cen_api_crime_days.plot(x='Services',y='Crimes')
```

Out[82]: <matplotlib.axes._subplots.AxesSubplot at 0x21ecb4b9668>



What were the insights gained from the data?

- **Based on the scattered plot, we came to the conclusion that population and service requests are directly proportional. As the population increases, the number of service requests also increases.**
- **There is a direct relationship between the age groups in particular neighborhood and way in which the maximum requests are raised there. For example, requests raised using APIs were maximum in the neighborhoods where there was a majoriy of young crowd and the area where senior citizens were in majority shows maximum amount of requests raised using phone calls.**
- **We tried to display the safety of some baltimore neighborhood based on the number of crimes occuring in each. The area whcih is most dangerous was New Northwood and lochraven whereas Villages of Homeland and Cedarcroft are the safest neighborhoods. We have ploted the bar graphs showing degree of safety for each neighborhood using different colors.**
- **The number of service requests were higher where large number of crimes occured. We can infer from this that since the crimes were more, there were lot of issues in the public services prvided or may be other way round.**
- **Auto theft, Common Assault, Burglary are some of the highly occuring crimes in Baltimore.**
- **Taking into consideration Auto theft crime (Type of crime that occured maximum), Northeastern districts has maximum occurances of this crime particularly, Frankfort and Belair-Edison Neighborhoods of this district.**
- **As we found above that Loch ravens is most dangerous neighborhood, we also found that the TRS-abandoned vehicles and BGE-street lights out are the highest services that are requests here. We can conclude from this that due to frequently occuring street lights out, crimes can occur more here as is seen via plots.**

Why is what we were trying to do valuable, and what did we learn?

The dataset that we are trying to plot is valuable in a sense that it gives the insight of the how safe is the area where we stay. Also, the population ditsribution based on age, gender etc can be understod. The relation between the age groups and method of raising requests can be interesting to know about what mode of communication is prefered by which age groups. Crime and service requests relationship will give us insights of how the number of service requests are impacted based on crimes occuring in any given neighborhood.

Why would someone else be interested in knowing what we learned? How might they use that knowledge?

Someone staying in baltimore will be interested to know which is the safe neighbourhood and which is the dangerous so that the person can make the decision for accomodation. People will also be interested in our insights which shows how fast does the service is handled in which heighborhood of baltimore. The further study can be made to check the root causes of why the particular neighborhood takes longer to resolve service requests. This might help to know the scope of improvement in the performance.