ETL Project Report

Title: Toronto Neighbourhood Population Census vs Crime Data

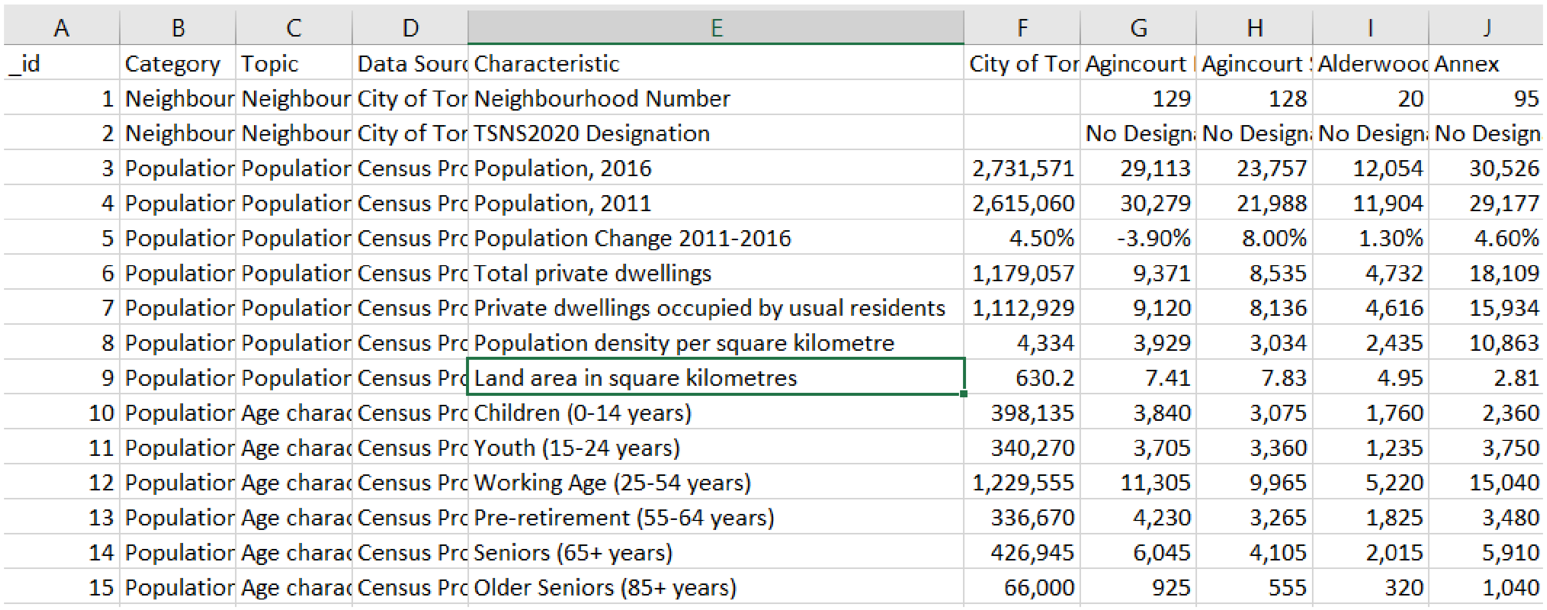
Finding Data (Extract)

Data Sources:

- **Toronto Neighbourhood 2016 Census of Population**
  https://ckan0.cf.opendata.inter.prod-toronto.ca/en_AU/dataset/neighbourhood-profiles

  Data Format – **CSV** file

  In this csv file all the data was formatted in an unstructured manner where all the neighbourhood was aligned in columns and all the various population data was aligned in rows as below.

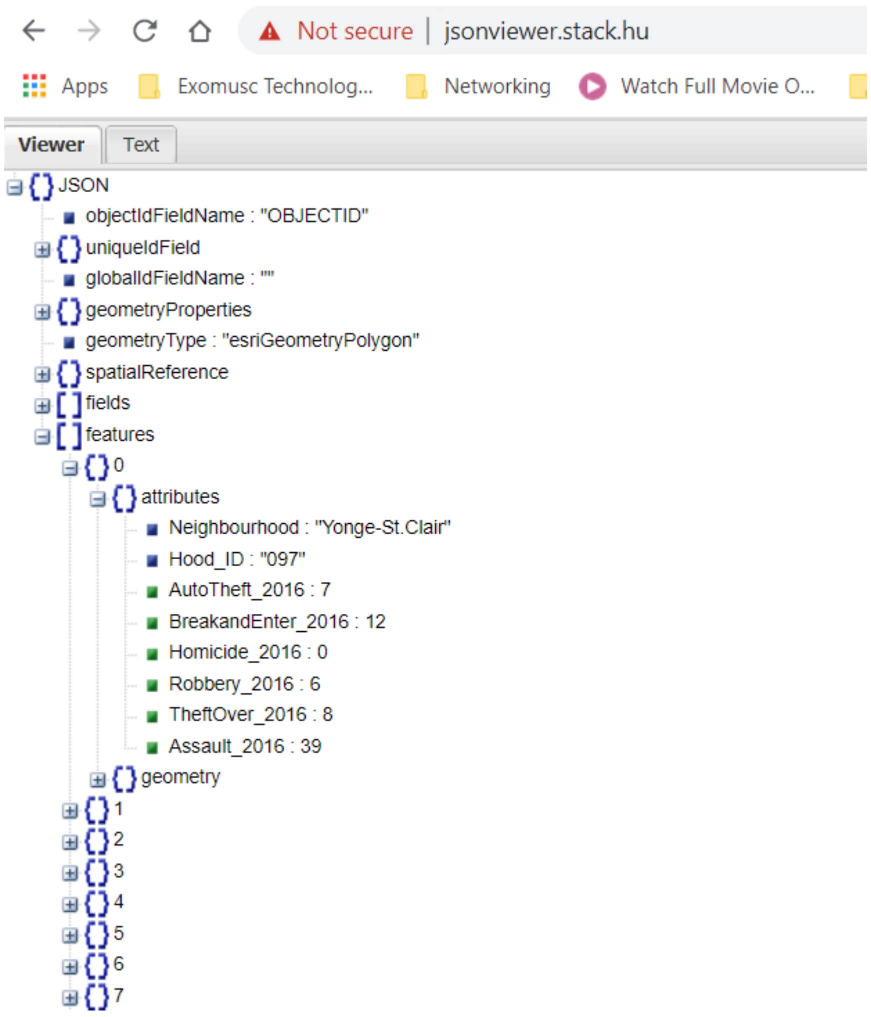


A	B	C	D	E	F	G	H	I	J
_id	Category	Topic	Data Sourc	Characteristic	City of Tor	Agincourt	Agincourt	Alderwood	Annex
1	Neighbour	Neighbour	City of Tor	Neighbourhood Number		129	128	20	95
2	Neighbour	Neighbour	City of Tor	TSNS2020 Designation		No Design	No Design	No Design	No Design
3	Populatior	Populatior	Census Prc	Population, 2016	2,731,571	29,113	23,757	12,054	30,526
4	Populatior	Populatior	Census Prc	Population, 2011	2,615,060	30,279	21,988	11,904	29,177
5	Populatior	Populatior	Census Prc	Population Change 2011-2016	4.50%	-3.90%	8.00%	1.30%	4.60%
6	Populatior	Populatior	Census Prc	Total private dwellings	1,179,057	9,371	8,535	4,732	18,109
7	Populatior	Populatior	Census Prc	Private dwellings occupied by usual residents	1,112,929	9,120	8,136	4,616	15,934
8	Populatior	Populatior	Census Prc	Population density per square kilometre	4,334	3,929	3,034	2,435	10,863
9	Populatior	Populatior	Census Prc	Land area in square kilometres	630.2	7.41	7.83	4.95	2.81
10	Populatior	Age charac	Census Prc	Children (0-14 years)	398,135	3,840	3,075	1,760	2,360
11	Populatior	Age charac	Census Prc	Youth (15-24 years)	340,270	3,705	3,360	1,235	3,750
12	Populatior	Age charac	Census Prc	Working Age (25-54 years)	1,229,555	11,305	9,965	5,220	15,040
13	Populatior	Age charac	Census Prc	Pre-retirement (55-64 years)	336,670	4,230	3,265	1,825	3,480
14	Populatior	Age charac	Census Prc	Seniors (65+ years)	426,945	6,045	4,105	2,015	5,910
15	Populatior	Age charac	Census Prc	Older Seniors (85+ years)	66,000	925	555	320	1,040

- **Toronto Neighbourhood 2016 Crime Data**
  http://data.torontopolice.on.ca/datasets/neighbourhood-crime-rates-boundary-file-/geoservice?geometry=-80.421%2C43.542%2C-78.335%2C43.890&orderBy=Hood_ID

  Data Format – **GeoJSON** file

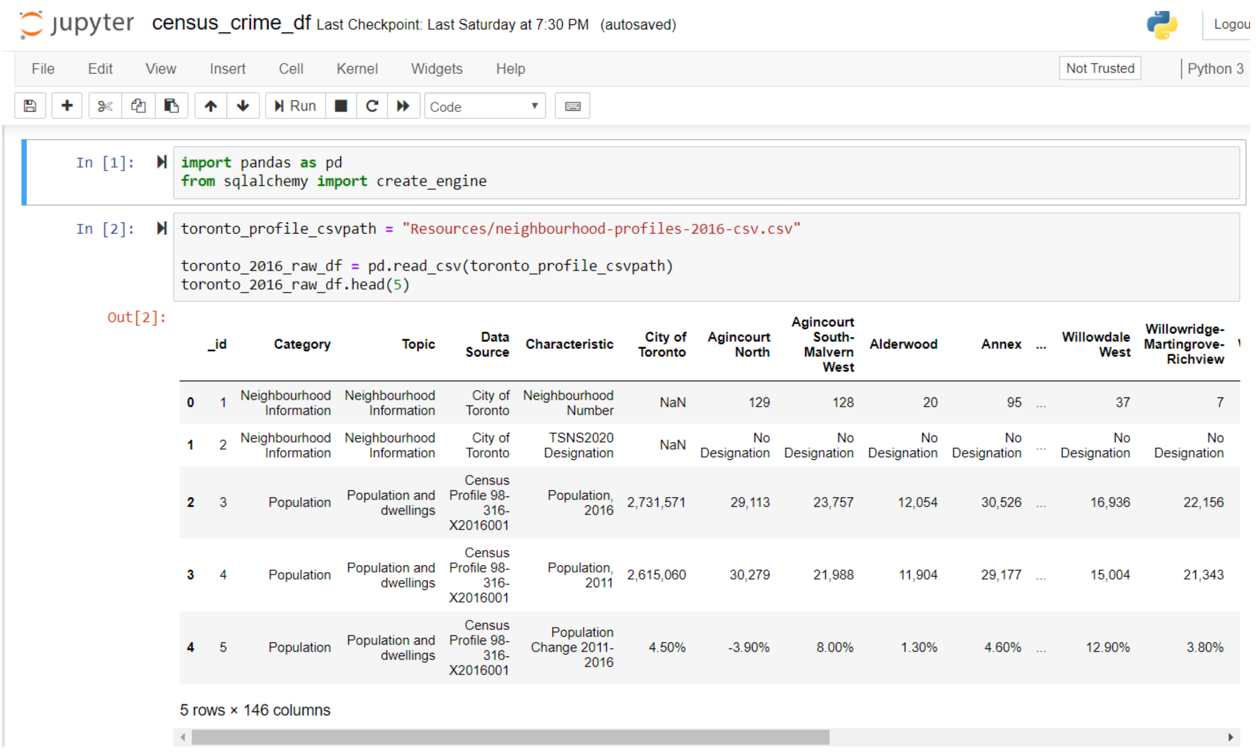
  From this link, we opted for 2016 crime data for each Toronto neighbourhood in GeoJSON format. In JSON viewer, the data structure looked like below.



```
{ } JSON
    objectIdFieldName : "OBJECTID"
  { } uniqueIdField
    globalIdFieldName : ""
  { } geometryProperties
    geometryType : "esriGeometryPolygon"
  { } spatialReference
  [ ] fields
  [ ] features
    { } 0
      { } attributes
          Neighbourhood : "Yonge-St.Clair"
          Hood_ID : "097"
          AutoTheft_2016 : 7
          BreakandEnter_2016 : 12
          Homicide_2016 : 0
          Robbery_2016 : 6
          TheftOver_2016 : 8
          Assault_2016 : 39
      { } geometry
    { } 1
    { } 2
    { } 3
    { } 4
    { } 5
    { } 6
    { } 7
```

Data Cleanup and & Analysis (Transform)

From the 2016 Toronto population data, all the neighbourhood names and its hood id and age population spectrum data were extracted. In original data, the neighbourhood name was the column index, hood id was in row format and the population spectrum was various row format structure shown as below.



jupyter census_crime_df Last Checkpoint: Last Saturday at 7:30 PM (autosaved)

```
In [1]: import pandas as pd
        from sqlalchemy import create_engine
```

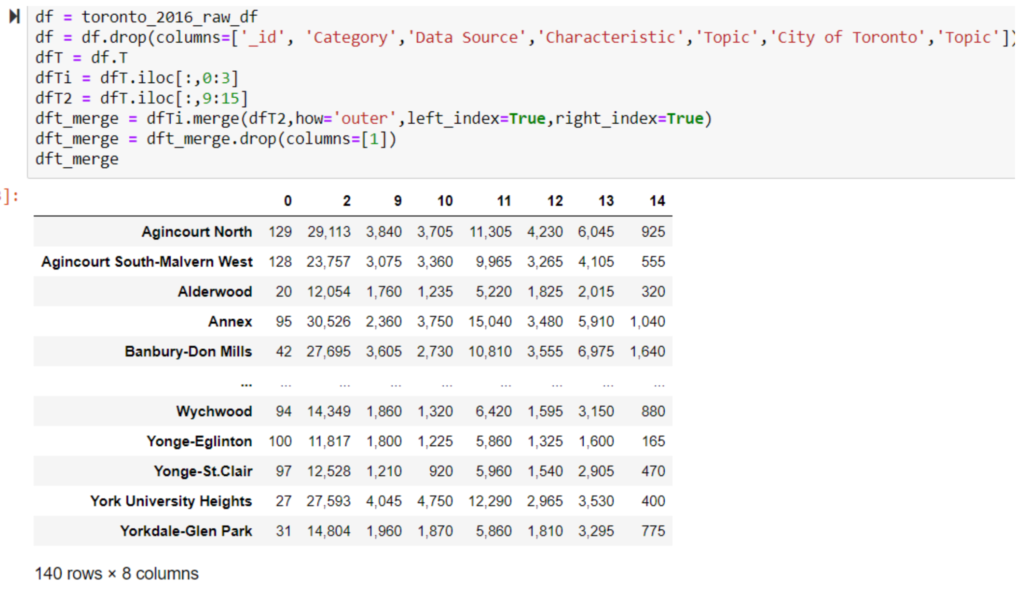
```
In [2]: toronto_profile_csvpath = "Resources/neighbourhood-profiles-2016-csv.csv"

        toronto_2016_raw_df = pd.read_csv(toronto_profile_csvpath)
        toronto_2016_raw_df.head(5)
```

Out[2]:

	_id	Category	Topic	Data Source	Characteristic	City of Toronto	Agincourt North	Agincourt South-Malvern West	Alderwood	Annex	...	Willowdale West	Willowridge-Martingrove-Richview
0	1	Neighbourhood Information	Neighbourhood Information	City of Toronto	Neighbourhood Number	NaN	129	128	20	95	...	37	7
1	2	Neighbourhood Information	Neighbourhood Information	City of Toronto	TSNS2020 Designation	NaN	No Designation	No Designation	No Designation	No Designation	...	No Designation	No Designation
2	3	Population	Population and dwellings	Census Profile 98-316-X2016001	Population, 2016	2,731,571	29,113	23,757	12,054	30,526	...	16,936	22,156
3	4	Population	Population and dwellings	Census Profile 98-316-X2016001	Population, 2011	2,615,060	30,279	21,988	11,904	29,177	...	15,004	21,343
4	5	Population	Population and dwellings	Census Profile 98-316-X2016001	Population Change 2011-2016	4.50%	-3.90%	8.00%	1.30%	4.60%	...	12.90%	3.80%

5 rows × 146 columns

The unwanted columns were dropped, and the specific columns and rows were extracted and put in the individual dataframes by using iloc function and the dataframes were merged to create a single neighbourhood population dataframe as below



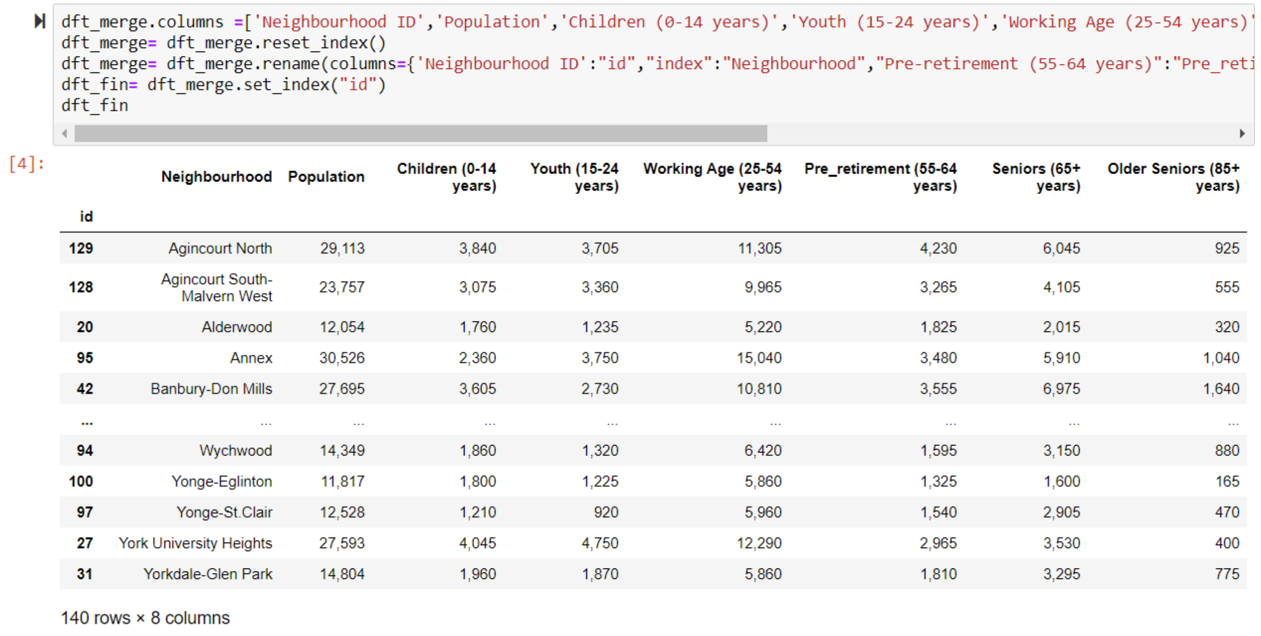
```
df = toronto_2016_raw_df
df = df.drop(columns=['_id', 'Category','Data Source','Characteristic','Topic','City of Toronto','Topic'])
dfT = df.T
dfTi = dfT.iloc[:,0:3]
dfT2 = dfT.iloc[:,9:15]
dft_merge = dfTi.merge(dfT2,how='outer',left_index=True,right_index=True)
dft_merge = dft_merge.drop(columns=[1])
dft_merge
```

[3]:

	0	2	9	10	11	12	13	14
Agincourt North	129	29,113	3,840	3,705	11,305	4,230	6,045	925
Agincourt South-Malvern West	128	23,757	3,075	3,360	9,965	3,265	4,105	555
Alderwood	20	12,054	1,760	1,235	5,220	1,825	2,015	320
Annex	95	30,526	2,360	3,750	15,040	3,480	5,910	1,040
Banbury-Don Mills	42	27,695	3,605	2,730	10,810	3,555	6,975	1,640
...	...	...	...	...	...	...	...	...
Wychwood	94	14,349	1,860	1,320	6,420	1,595	3,150	880
Yonge-Eglinton	100	11,817	1,800	1,225	5,860	1,325	1,600	165
Yonge-St.Clair	97	12,528	1,210	920	5,960	1,540	2,905	470
York University Heights	27	27,593	4,045	4,750	12,290	2,965	3,530	400
Yorkdale-Glen Park	31	14,804	1,960	1,870	5,860	1,810	3,295	775

140 rows × 8 columns

As columns transformed into rows and rows into columns, we had to name all the individual columns indexes its appropriate names.



```
dft_merge.columns =['Neighbourhood ID','Population','Children (0-14 years)','Youth (15-24 years)','Working Age (25-54 years)'
dft_merge= dft_merge.reset_index()
dft_merge= dft_merge.rename(columns={'Neighbourhood ID':"id","index":"Neighbourhood","Pre-retirement (55-64 years)":"Pre_reti
dft_fin= dft_merge.set_index("id")
dft_fin
```

[4]:

	Neighbourhood	Population	Children (0-14 years)	Youth (15-24 years)	Working Age (25-54 years)	Pre_retirement (55-64 years)	Seniors (65+ years)	Older Seniors (85+ years)
id								
129	Agincourt North	29,113	3,840	3,705	11,305	4,230	6,045	925
128	Agincourt South-Malvern West	23,757	3,075	3,360	9,965	3,265	4,105	555
20	Alderwood	12,054	1,760	1,235	5,220	1,825	2,015	320
95	Annex	30,526	2,360	3,750	15,040	3,480	5,910	1,040
42	Banbury-Don Mills	27,695	3,605	2,730	10,810	3,555	6,975	1,640
...	...	...	...	...	...	...	...	...
94	Wychwood	14,349	1,860	1,320	6,420	1,595	3,150	880
100	Yonge-Eglinton	11,817	1,800	1,225	5,860	1,325	1,600	165
97	Yonge-St.Clair	12,528	1,210	920	5,960	1,540	2,905	470
27	York University Heights	27,593	4,045	4,750	12,290	2,965	3,530	400
31	Yorkdale-Glen Park	14,804	1,960	1,870	5,860	1,810	3,295	775

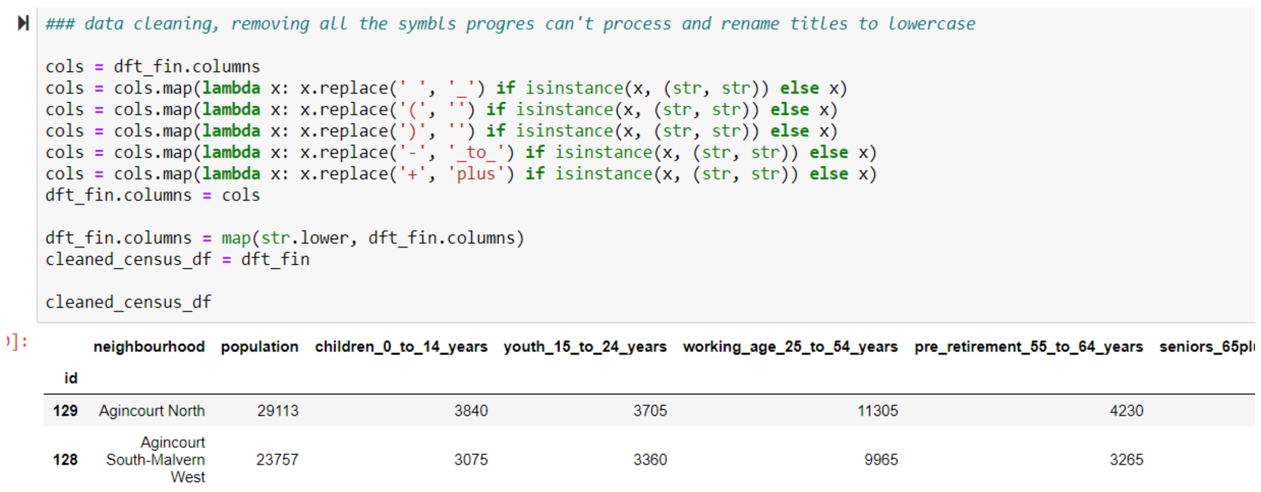
140 rows × 8 columns

Database transformation from "jupyter notebook" to "pgAdmin" in sql format requires the keys and values of the source databases to be presented in the format as pgAdmin would if the database tables are created with the program's query tools, failure to format properly would result in pgAdmin unable to load the source data.

A few steps were taken to clean the column keys (column titles);

1. no symbols, such as hyphen, plus sign and parentheses
2. no space, which we had to replace with underscore
3. no capital letters, all capital letters in column title has to changed into lower case (code used: df.columns=map(str.lower,df.columns))

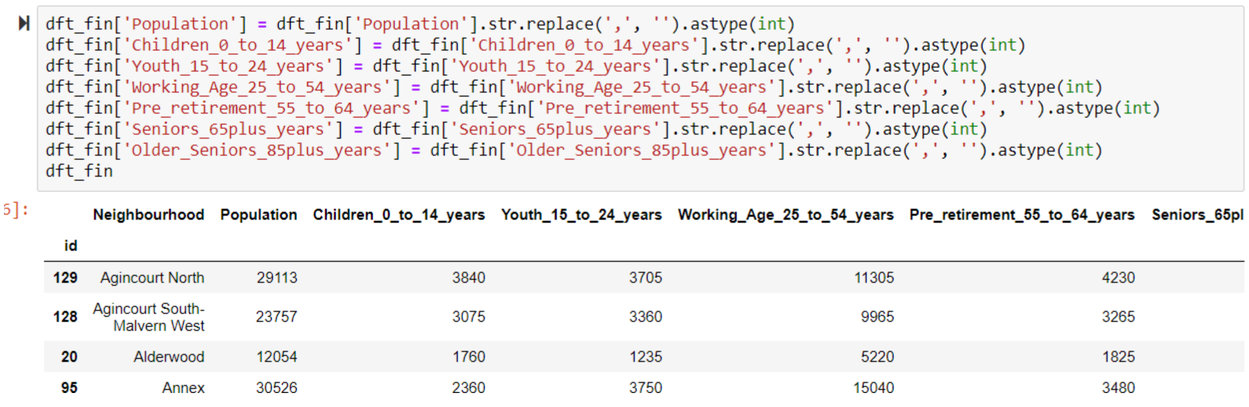
In addition, In order for pgAdmin to process dataframe numbers as integers, during the cleaning process, we had to make sure, I.E 3,711 is expressed as 3711, removing any delimiter.



```
### data cleaning, removing all the symbls progres can't process and rename titles to lowercase

cols = dft_fin.columns
cols = cols.map(lambda x: x.replace(' ', '_') if isinstance(x, (str, str)) else x)
cols = cols.map(lambda x: x.replace('(', '') if isinstance(x, (str, str)) else x)
cols = cols.map(lambda x: x.replace(')', '') if isinstance(x, (str, str)) else x)
cols = cols.map(lambda x: x.replace('-', '_to_') if isinstance(x, (str, str)) else x)
cols = cols.map(lambda x: x.replace('+', 'plus') if isinstance(x, (str, str)) else x)
dft_fin.columns = cols

dft_fin.columns = map(str.lower, dft_fin.columns)
cleaned_census_df = dft_fin

cleaned_census_df
```

]:

	neighbourhood	population	children_0_to_14_years	youth_15_to_24_years	working_age_25_to_54_years	pre_retirement_55_to_64_years	seniors_65pl
id							
129	Agincourt North	29113	3840	3705	11305	4230	
128	Agincourt South-Malvern West	23757	3075	3360	9965	3265	

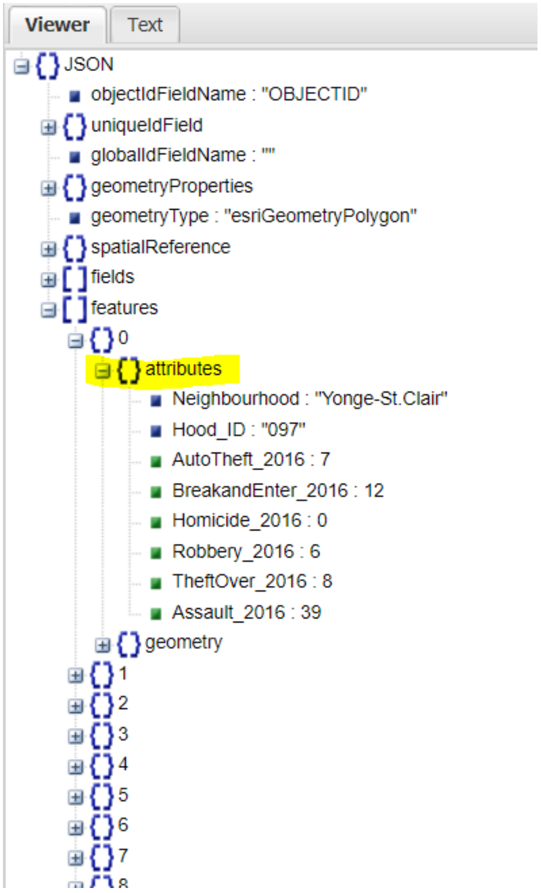


```
dft_fin['Population'] = dft_fin['Population'].str.replace(',', '').astype(int)
dft_fin['Children_0_to_14_years'] = dft_fin['Children_0_to_14_years'].str.replace(',', '').astype(int)
dft_fin['Youth_15_to_24_years'] = dft_fin['Youth_15_to_24_years'].str.replace(',', '').astype(int)
dft_fin['Working_Age_25_to_54_years'] = dft_fin['Working_Age_25_to_54_years'].str.replace(',', '').astype(int)
dft_fin['Pre_retirement_55_to_64_years'] = dft_fin['Pre_retirement_55_to_64_years'].str.replace(',', '').astype(int)
dft_fin['Seniors_65plus_years'] = dft_fin['Seniors_65plus_years'].str.replace(',', '').astype(int)
dft_fin['Older_Seniors_85plus_years'] = dft_fin['Older_Seniors_85plus_years'].str.replace(',', '').astype(int)
dft_fin
```

5]:

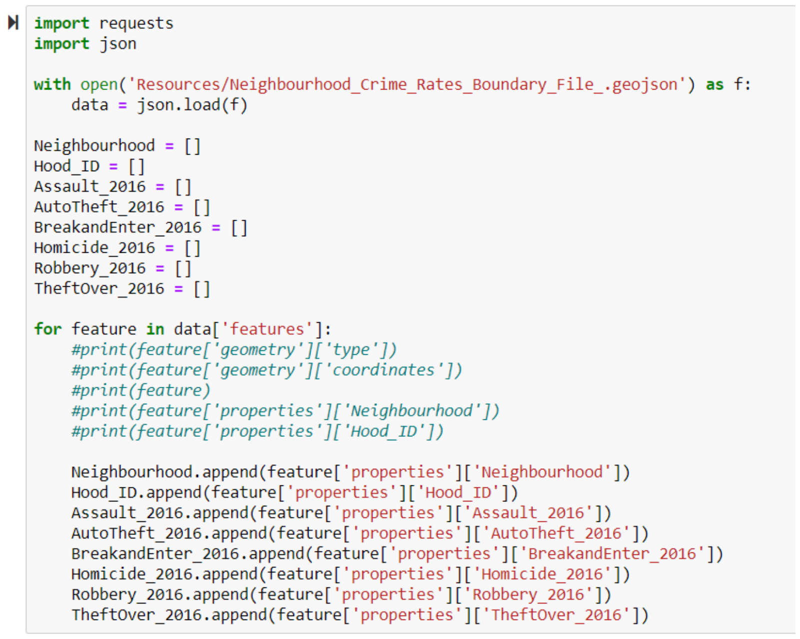
	Neighbourhood	Population	Children_0_to_14_years	Youth_15_to_24_years	Working_Age_25_to_54_years	Pre_retirement_55_to_64_years	Seniors_65pl
id							
129	Agincourt North	29113	3840	3705	11305	4230	
128	Agincourt South-Malvern West	23757	3075	3360	9965	3265	
20	Alderwood	12054	1760	1235	5220	1825	
95	Annex	30526	2360	3750	15040	3480	

For the downloaded data for Toronto Neighbourhood 2016 crime was in GeoJSON format. It seemed very unstructured compared to usual JSON format, shown below:



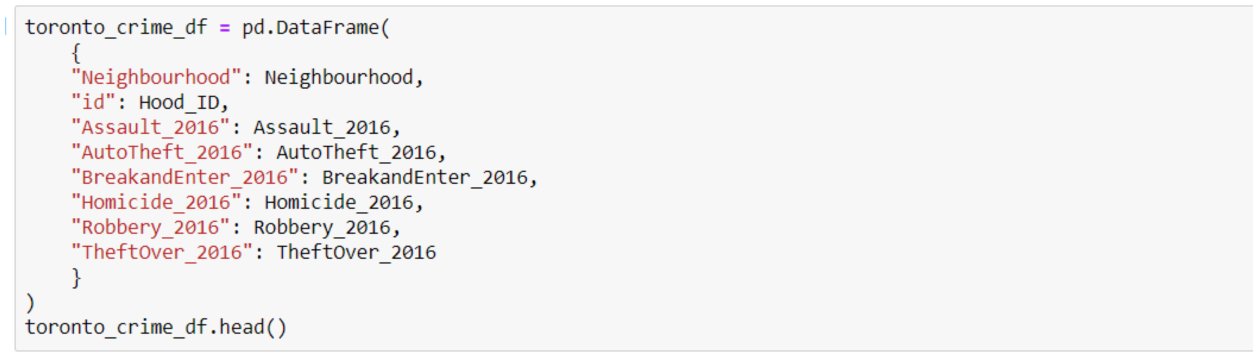
```
{ } JSON
    objectIdFieldName : "OBJECTID"
  { } uniqueIdField
    globalIdFieldName : ""
  { } geometryProperties
    geometryType : "esriGeometryPolygon"
  { } spatialReference
  [ ] fields
  [ ] features
    { } 0
      { } attributes
          Neighbourhood : "Yonge-St.Clair"
          Hood_ID : "097"
          AutoTheft_2016 : 7
          BreakandEnter_2016 : 12
          Homicide_2016 : 0
          Robbery_2016 : 6
          TheftOver_2016 : 8
          Assault_2016 : 39
      { } geometry
    { } 1
    { } 2
    { } 3
    { } 4
    { } 5
    { } 6
    { } 7
    { } 8
```

From this data, the attributes dictionary was the only data needed. Everything else was noise. Here 'features' list contains 'attributes' data and 'Neighbourhood', 'Hood_ID', etc. are properties of 'attributes' dictionary.
In order to extract the targeted data, json was imported, json.load function was used and all the required data was appended within the individual lists.

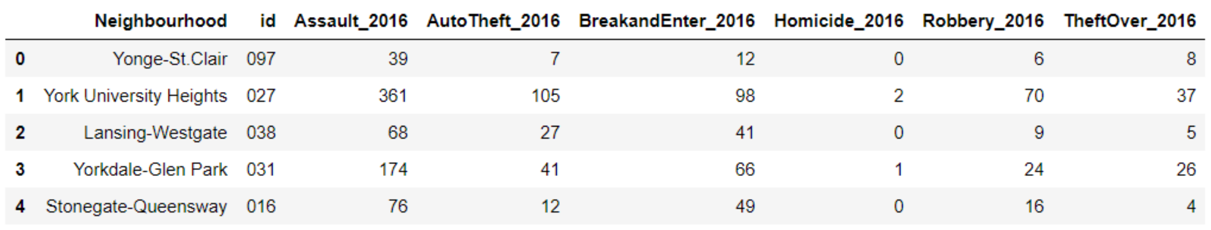


```python
import requests
import json

with open('Resources/Neighbourhood_Crime_Rates_Boundary_File_.geojson') as f:
    data = json.load(f)

Neighbourhood = []
Hood_ID = []
Assault_2016 = []
AutoTheft_2016 = []
BreakandEnter_2016 = []
Homicide_2016 = []
Robbery_2016 = []
TheftOver_2016 = []

for feature in data['features']:
    #print(feature['geometry']['type'])
    #print(feature['geometry']['coordinates'])
    #print(feature)
    #print(feature['properties']['Neighbourhood'])
    #print(feature['properties']['Hood_ID'])

    Neighbourhood.append(feature['properties']['Neighbourhood'])
    Hood_ID.append(feature['properties']['Hood_ID'])
    Assault_2016.append(feature['properties']['Assault_2016'])
    AutoTheft_2016.append(feature['properties']['AutoTheft_2016'])
    BreakandEnter_2016.append(feature['properties']['BreakandEnter_2016'])
    Homicide_2016.append(feature['properties']['Homicide_2016'])
    Robbery_2016.append(feature['properties']['Robbery_2016'])
    TheftOver_2016.append(feature['properties']['TheftOver_2016'])
```

Toronto_crime dataframe was created with all the data extracted within the individual lists.

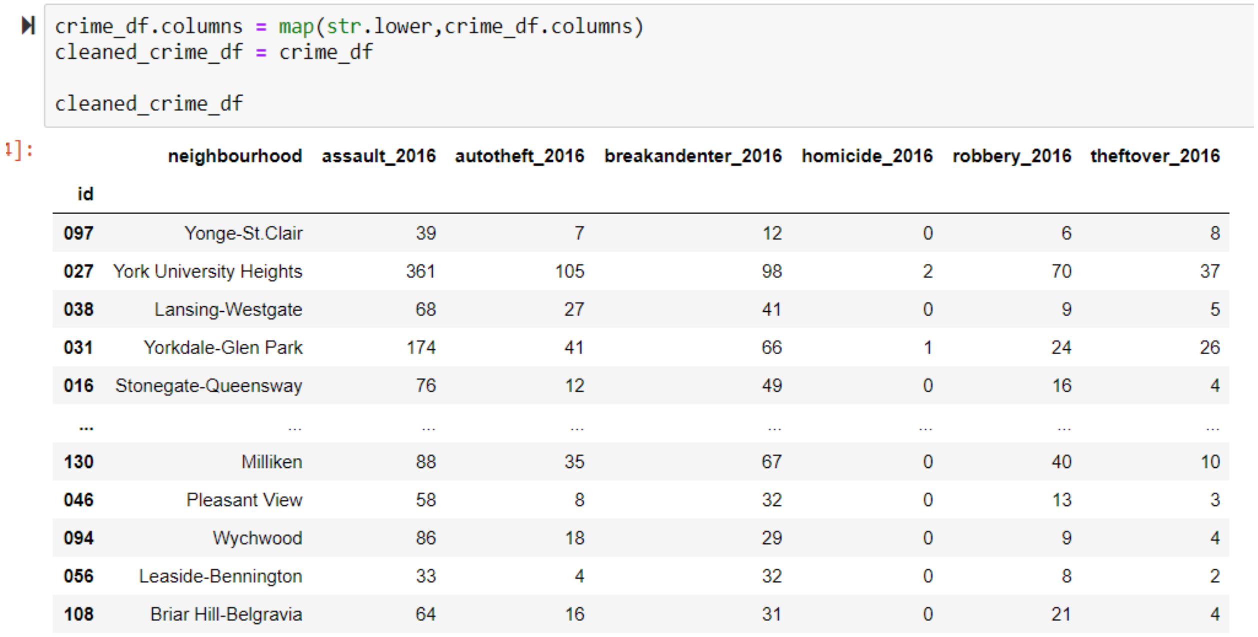


```python
toronto_crime_df = pd.DataFrame(
    {
    "Neighbourhood": Neighbourhood,
    "id": Hood_ID,
    "Assault_2016": Assault_2016,
    "AutoTheft_2016": AutoTheft_2016,
    "BreakandEnter_2016": BreakandEnter_2016,
    "Homicide_2016": Homicide_2016,
    "Robbery_2016": Robbery_2016,
    "TheftOver_2016": TheftOver_2016
    }
)
toronto_crime_df.head()
```



	Neighbourhood	id	Assault_2016	AutoTheft_2016	BreakandEnter_2016	Homicide_2016	Robbery_2016	TheftOver_2016
0	Yonge-St.Clair	097	39	7	12	0	6	8
1	York University Heights	027	361	105	98	2	70	37
2	Lansing-Westgate	038	68	27	41	0	9	5
3	Yorkdale-Glen Park	031	174	41	66	1	24	26
4	Stonegate-Queensway	016	76	12	49	0	16	4

As mentioned before, column index names were converted to lowercase as PGAdmin was being difficult accepting this table as is.

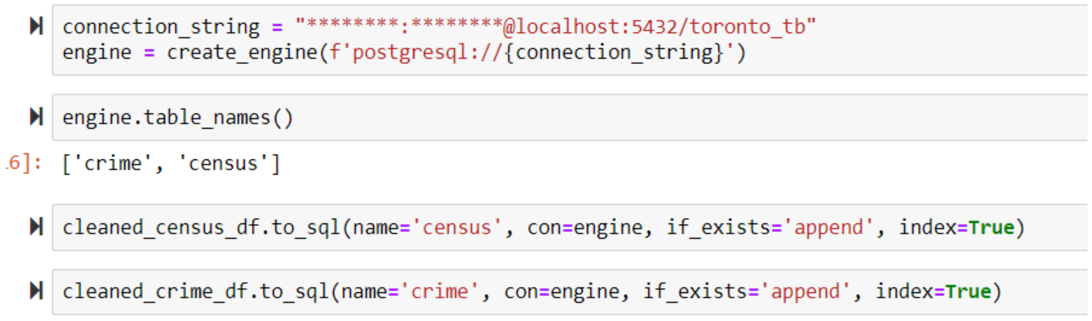


```
crime_df.columns = map(str.lower,crime_df.columns)
cleaned_crime_df = crime_df

cleaned_crime_df
```

1]:

	neighbourhood	assault_2016	autotheft_2016	breakandenter_2016	homicide_2016	robbery_2016	theftover_2016
id							
097	Yonge-St.Clair	39	7	12	0	6	8
027	York University Heights	361	105	98	2	70	37
038	Lansing-Westgate	68	27	41	0	9	5
031	Yorkdale-Glen Park	174	41	66	1	24	26
016	Stonegate-Queensway	76	12	49	0	16	4
...	...	...	...	...	...	...	...
130	Milliken	88	35	67	0	40	10
046	Pleasant View	58	8	32	0	13	3
094	Wychwood	86	18	29	0	9	4
056	Leaside-Bennington	33	4	32	0	8	2
108	Briar Hill-Belgravia	64	16	31	0	21	4

Connection to the SQL postgres as below.



```
connection_string = "********:********@localhost:5432/toronto_tb"
engine = create_engine(f'postgresql://{connection_string}')
```

```
engine.table_names()
```

```
.6]: ['crime', 'census']
```

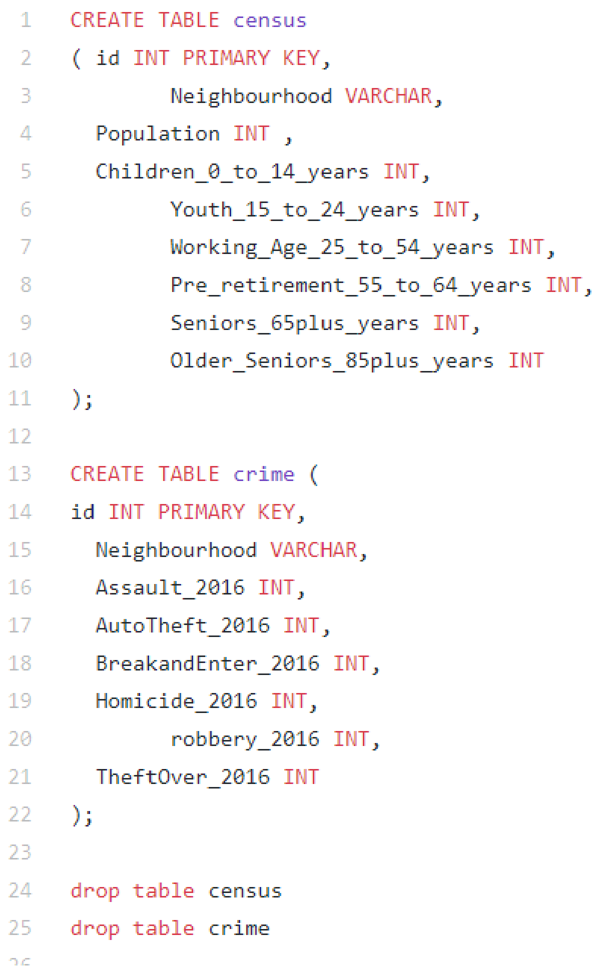
```
cleaned_census_df.to_sql(name='census', con=engine, if_exists='append', index=True)
```

```
cleaned_crime_df.to_sql(name='crime', con=engine, if_exists='append', index=True)
```

Loading to Database (Load)

Among the two different data sources, there is a common key which the data sources each used to identify census area, which is the Toronto neighbourhood number (neighbourhood ID, referred simply as ‘ID’ in the final table).

Data base was created in pgAdmin and tables were created as below:

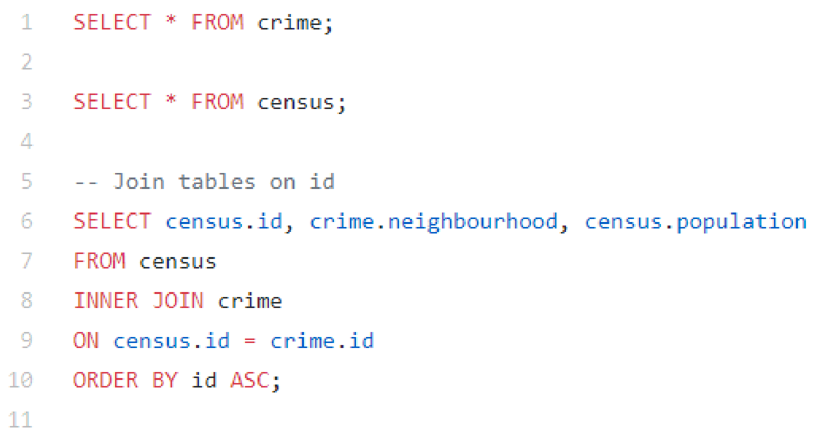


```
CREATE TABLE census
( id INT PRIMARY KEY,
        Neighbourhood VARCHAR,
  Population INT ,
  Children_0_to_14_years INT,
        Youth_15_to_24_years INT,
        Working_Age_25_to_54_years INT,
        Pre_retirement_55_to_64_years INT,
        Seniors_65plus_years INT,
        Older_Seniors_85plus_years INT
);

CREATE TABLE crime (
id INT PRIMARY KEY,
  Neighbourhood VARCHAR,
  Assault_2016 INT,
  AutoTheft_2016 INT,
  BreakandEnter_2016 INT,
  Homicide_2016 INT,
        robbery_2016 INT,
  TheftOver_2016 INT
);

drop table census
drop table crime
```

The databases are loaded through jupyter’s notebook and then connected and transferred to pgAdmin for easier merging and browsing.

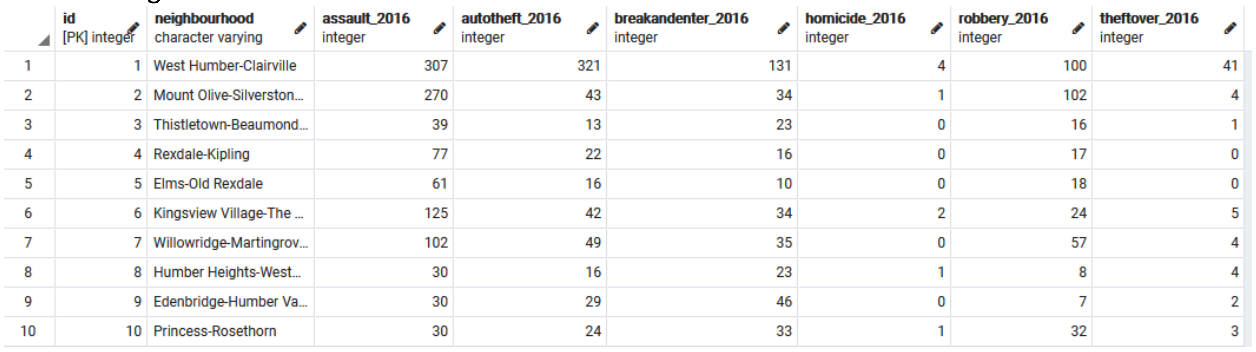
The Neighbourhood Profile database contains data of population in each neighbourhood and their respective age demographic, whilst the Toronto Crime Rates database contains data of the count of different crimes committed in different neighbourhood.

SQL queries:



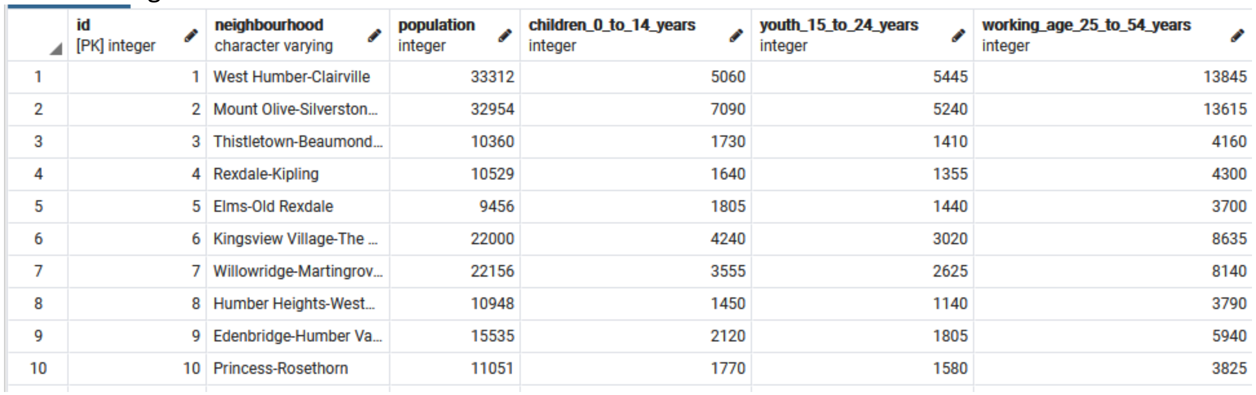
```sql
SELECT * FROM crime;

SELECT * FROM census;

-- Join tables on id
SELECT census.id, crime.neighbourhood, census.population
FROM census
INNER JOIN crime
ON census.id = crime.id
ORDER BY id ASC;

```

Toronto Neighbourhood Profiles:



	id [PK] integer	neighbourhood character varying	assault_2016 integer	autotheft_2016 integer	breakandenter_2016 integer	homicide_2016 integer	robbery_2016 integer	theftover_2016 integer
1	1	West Humber-Clairville	307	321	131	4	100	41
2	2	Mount Olive-Silverston...	270	43	34	1	102	4
3	3	Thistletown-Beaumond...	39	13	23	0	16	1
4	4	Rexdale-Kipling	77	22	16	0	17	0
5	5	Elms-Old Rexdale	61	16	10	0	18	0
6	6	Kingsview Village-The ...	125	42	34	2	24	5
7	7	Willowridge-Martingrov...	102	49	35	0	57	4
8	8	Humber Heights-West...	30	16	23	1	8	4
9	9	Edenbridge-Humber Va...	30	29	46	0	7	2
10	10	Princess-Rosethorn	30	24	33	1	32	3

Toronto Neighbourhood Crime Rates:



	id [PK] integer	neighbourhood character varying	population integer	children_0_to_14_years integer	youth_15_to_24_years integer	working_age_25_to_54_years integer
1	1	West Humber-Clairville	33312	5060	5445	13845
2	2	Mount Olive-Silverston...	32954	7090	5240	13615
3	3	Thistletown-Beaumond...	10360	1730	1410	4160
4	4	Rexdale-Kipling	10529	1640	1355	4300
5	5	Elms-Old Rexdale	9456	1805	1440	3700
6	6	Kingsview Village-The ...	22000	4240	3020	8635
7	7	Willowridge-Martingrov...	22156	3555	2625	8140
8	8	Humber Heights-West...	10948	1450	1140	3790
9	9	Edenbridge-Humber Va...	15535	2120	1805	5940
10	10	Princess-Rosethorn	11051	1770	1580	3825

By merging the two tables together, we are hoping to discover; 1), which neighbourhood is more prominent in crime and 2) if there is a relation between crime rate and the age demographic.