ETL Project Report

Our overarching goal is to generate a dataset that allow us to perform statistical analysis and visualization on the demographics in the 77 community areas of Chicago, and whether any demographic characteristics is associated with the rate of flu shot given in that area.

Extract & Transform

Three datasets were used for this project:

1. Chicago community data snapshot (.csv). Source: CMAP Data Hub.
   This dataset includes key demographic data from each of Chicago's community area from 2013-2017. The dataset has 203 columns that covers information like employment status, nature of employment, education level, age, housing status, housing size, vehicle ownership, etc.

	GEOG	2000_POP	2010_POP	TOT_POP	UND19	A20_34	A35_49	A50_64	A65_74	A75_84	...	ARABIC	KOREAN	OTHE
0	Albany Park	57655	51542	51575	13354	13738	11975	8232	2329	1343	...	621	650	
1	Archer Heights	12644	13393	13233	4016	2933	2879	1897	907	369	...	24	0	
2	Armour Square	12032	13391	13699	2997	2824	2539	2541	1371	1043	...	1	6	
3	Ashburn	39584	41081	43283	12983	8368	8998	8477	2855	977	...	444	9	
4	Auburn Gresham	55928	48743	45770	11432	9224	7941	9587	4041	2700	...	0	11	

5 rows × 203 columns

   For our purpose, we decide to extract only community number, age demographics (8 columns: under 19, age 20-34, 35-49… and median age), ethnicity (5 columns: white, Hispanic, black, asian, other), employment status (4 columns: employed, unemployed, in labor force, not in labor force), education status (6 columns: high school, college etc), and income (7 columns: less than 25K, 25-50K… median income). We also renamed the columns for ease of interpretation. Our final cleaned table has 31 columns:

	Area	<19y	20-34y	35-59y	50-64y	65-74y	75-84y	85y<	Median Age	WHITE	...	ASSOC	BACH	GRAD_PROF	INC_LT_25K	IN
0	Albany Park	13354	13738	11975	8232	2329	1343	604	33.6	14932	...	2047	7339	4041	3627	
1	Archer Heights	4016	2933	2879	1897	907	369	231	33.0	2477	...	278	663	195	946	
2	Armour Square	2997	2824	2539	2541	1371	1043	385	42.0	1599	...	505	1313	884	2483	
3	Ashburn	12983	8368	8998	8477	2855	977	625	35.5	5044	...	2239	3595	1906	2021	
4	Auburn Gresham	11432	9224	7941	9587	4041	2700	845	39.6	257	...	2113	2771	1805	7497	

From the same data source, we obtained a separate .csv file that contains the key that identify each community with its community area number:

	Community Area Number	Area
0	1.0	Rogers Park
1	2.0	West Ridge
2	3.0	Uptown
3	4.0	Lincoln Square
4	5.0	North Center

A merge function is called to combine this key to the above cleaned community dataset.

	community_area_number	Area	<19y	20-34y	35-59y	50-64y	65-74y	75-84y	85y<	Median Age	...	ASSOC	BACH	GRAD_PROF
0	1.0	Rogers Park	12092	16967	12526	9130	2718	1268	799	33.8	...	1825	9693	6627
1	2.0	West Ridge	19951	16264	14581	13704	5138	2765	1320	35.6	...	2830	12247	7161
2	3.0	Uptown	7030	18725	13253	10706	3333	1977	1272	37.5	...	2329	15315	9176
3	4.0	Lincoln Square	7306	13696	10180	6588	2228	976	509	34.7	...	1527	11589	7831
4	5.0	North Center	8266	10302	9454	4024	1602	808	486	33.8	...	925	10509	7017

5 rows × 32 columns

The resulting dataset is exported as a new .csv file named “cleaned_census.df.csv.” This dataset has these columns:

In [12]:
```
#Export to CSV file
combined_census.to_csv("cleaned_census_df.csv")
```

In [13]:
```
combined_census.columns
```

Out[13]:
```
Index(['community_area_number', 'Area', '<19y', '20-34y', '35-59y', '50-64y',
       '65-74y', '75-84y', '85y<', 'Median Age', 'WHITE', 'HISP', 'BLACK',
       'ASIAN', 'OTHER', 'IN_LBFRC', 'EMP', 'UNEMP', 'NOT_IN_LBFRC', 'LT_HS',
       'HS', 'SOME_COLL', 'ASSOC', 'BACH', 'GRAD_PROF', 'INC_LT_25K',
       'INC_25_50K', 'INC_50_75K', 'INC_75_100K', 'INC_100_150K', 'INC_GT_150',
       'median_income'],
      dtype='object')
```

2. Flu shot clinic locations in Chicago (.csv). Source: data.cityofchicago.org.
   This dataset includes a list of free flu clinics offered throughout Chicago. Seasons are identified by the years they span (eg. 2015-16) instead of the year in which they begin (eg. 2015). The raw dataset has 32 columns, including detailed information on the clinic location (latitude, longitude, street, city (all Chicago), postal code, etc)

	Season	Facility ID	Latitude	Longitude	Street1	Street2	City	State	Postal Code	Country
0	2017-2018	267	41.968500	-87.728760	4010 W LAWRENCE AVE	NaN	CHICAGO	IL	60630	United States of America
1	2015-2016	202	41.981429	-87.668555	5440 N CLARK ST	NaN	CHICAGO	IL	60640	United States of America
2	2015-2016	208	41.884543	-87.627803	151 N STATE ST	NaN	CHICAGO	IL	60601	United States of America
3	2015-2016	213	41.844305	-87.707719	3303 W 26TH ST	NaN	CHICAGO	IL	60623	United States of America
4	2016-2017	60	41.968300	-87.738086	4404 W. Lawrence Ave.	NaN	Chicago	IL	60630	United States of America

5 rows × 32 columns

We selected for only the columns that pertain to our interest, namely season, latitude, longitude, postal code, and community area number. Final dataframe has 5 columns:

	Season	Latitude	Longitude	Postal Code	community_area_number
0	2017-2018	41.968500	-87.728760	60630	14.0
1	2015-2016	41.981429	-87.668555	60640	76.0
2	2015-2016	41.884543	-87.627803	60601	38.0
3	2015-2016	41.844305	-87.707719	60623	32.0
4	2016-2017	41.968300	-87.738086	60630	14.0

The data is exported to a new .csv file, named “cleaned_flu_df.csv.”

```
1  #Export to CSV file
2  flushot_df.to_csv("cleaned_flu_df.csv")
```

3. Chicago crime data (.csv). Source: Chicago Data Portal.
https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2

The URL of our data source is included here because the raw datafile is too big to upload to github. The raw dataset has 22 columns that details each reported crime case in the Chicago PD database.

	ID	Case Number	Date	Block	IUCR	Primary Type	Description	Location Description	Arres
0	11668131	JC240018	04/26/2019 11:58:00 PM	017XX N CENTRAL AVE	1150	DECEPTIVE PRACTICE	CREDIT CARD FRAUD	GAS STATION	Fals
1	11668274	JC240043	04/26/2019 11:58:00 PM	008XX N MAY ST	0620	BURGLARY	UNLAWFUL ENTRY	APARTMENT	Fals
2	11668155	JC240031	04/26/2019 11:56:00 PM	046XX N MELVINA AVE	2093	NARCOTICS	FOUND SUSPECT NARCOTICS	PARK PROPERTY	Tru
3	11668197	JC240026	04/26/2019 11:51:00 PM	004XX W 83RD ST	143A	WEAPONS VIOLATION	UNLAWFUL POSS OF HANDGUN	STREET	Tru
4	11668158	JC239985	04/26/2019 11:49:00 PM	049XX W JACKSON BLVD	0486	BATTERY	DOMESTIC BATTERY SIMPLE	APARTMENT	Fals

5 rows × 22 columns

```
crimedata.columns
```

```
Index(['ID', 'Case Number', 'Date', 'Block', 'IUCR', 'Primary Type',
       'Description', 'Location Description', 'Arrest', 'Domestic', 'Bea
t',
       'District', 'Ward', 'Community Area', 'FBI Code', 'X Coordinate',
       'Y Coordinate', 'Year', 'Updated On', 'Latitude', 'Longitude',
       'Location', 'Time', 'AmPm'],
      dtype='object')
```

We selected only the columns of interest, the final dataset has 11 columns. We also split the Date column from '04/26/2019 11:58:00 PM" to a "Date" column consist only of date "04/26/2019" and a "Time" column that has time in 12-hour format, and a third column "AmPm."

Year	Latitude	Longitude	Time	AmPm
2019	41.912867	-87.765636	11:58:00	PM
2019	41.896700	-87.655246	11:58:00	PM
2019	41.965938	-87.781969	11:56:00	PM
2019	41.743674	-87.634697	11:51:00	PM
2019	41.876749	-87.747879	11:49:00	PM

The resulting dataset is exported to .csv file as "crime2013_2019.csv."

Further, we extracted the year from all the three tables to make mysql import easier.

Splitting Date strings into month, day, and year for easier mysql import

In [25]:
```
crime_df["Month"]=crime_df["Date"].str.split('/', expand=False).str[0].astype(str).astype(int)
crime_df["Day"]=crime_df["Date"].str.split('/', expand=False).str[1].astype(str).astype(int)
crime_df["Year"]=crime_df["Date"].str.split('/', expand=False).str[2].astype(str).astype(int)
```

In [26]:
```
crime_df.head()
```

Out[26]:

	ID	Date	Primary Type	Location Description	Arrest	Community Area	Year	Latitude	Longitude	Time	AmPm	Month	Day
0	11668131	04/26/2019	DECEPTIVE PRACTICE	GAS STATION	False	25.0	2019	41.912867	-87.765636	11:58:00	PM	4	26
1	11668274	04/26/2019	BURGLARY	APARTMENT	False	24.0	2019	41.896700	-87.655246	11:58:00	PM	4	26
2	11668155	04/26/2019	NARCOTICS	PARK PROPERTY	True	15.0	2019	41.965938	-87.781969	11:56:00	PM	4	26
3	11668197	04/26/2019	WEAPONS VIOLATION	STREET	True	71.0	2019	41.743674	-87.634697	11:51:00	PM	4	26
4	11668158	04/26/2019	BATTERY	APARTMENT	False	25.0	2019	41.876749	-87.747879	11:49:00	PM	4	26

n [41]:
```
flushot_df["Year"]=flushot_df["Season"].str.split('-', expand=False).str[0]
flushot_df.head()
```

ut[41]:

	Season	Latitude	Longitude	Postal Code	community_area_number	Year
0	2017-2018	41.968500	-87.728760	60630	14.0	2017
1	2015-2016	41.981429	-87.668555	60640	76.0	2015
2	2015-2016	41.884543	-87.627803	60601	38.0	2015
3	2015-2016	41.844305	-87.707719	60623	32.0	2015
4	2016-2017	41.968300	-87.738086	60630	14.0	2016

Load

Since all three of our dataset contain the “community area number” column that identify the geographical location in Chicago, a relational database (MySQL) is best suited to our needs. We established MySQL connection using sqlalchemy. To verify connection, we check for the list of empty tables that are created to load our dataframe into. We then load all three of our cleaned dataframe into sql and confirmed that the data has been added. An example is shown here:

Connect to the database

In [30]:
```
# Creating a connection string and engine for mysql local database
# replace {password} with your password for mysql database connection
connection_string = 'mysql://root:[password]@localhost:3306/shot_chicago'
engine = create_engine(connection_string)
```

In [38]:
```
# Checking connection using dummy tablename
engine.table_names()
```

Out[38]:
```
['census', 'chicrime', 'flushot']
```

Creating tables and inserting data for the dataframe in sql

In [32]:
```
crime_df.to_sql(name='chicrime', con=engine, if_exists='replace', index=False)
```

In [42]:
```
flushot_df.to_sql(name='flushot', con=engine, if_exists='replace', index=False)
```

In [59]:
```
census_df.to_sql(name='census', con=engine, if_exists='replace', index=False)
```

Checking if the data import was done successfully

[35]:
```
pd.read_sql_query('select * from chicrime', con=engine).head()
```

:[35]:

	ID	Date	Primary Type	Location Description	Arrest	Comn
0	11668131	04/26/2019	DECEPTIVE PRACTICE	GAS STATION	0	
1	11668274	04/26/2019	BURGLARY	APARTMENT	0	
2	11668155	04/26/2019	NARCOTICS	PARK PROPERTY	1	
3	11668197	04/26/2019	WEAPONS VIOLATION	STREET	1	
4	11668158	04/26/2019	BATTERY	APARTMENT	0	

Lastly, we combine the dataset using the “community area number,” an example is shown here:

```
#Combine flushot data and census data

query = """
SELECT
*
FROM censusdata A
INNER JOIN flushot B on A.community_area_number = B.community_area_numb
"""

pd.read_sql_query(query, con=engine).head()
```

	Unnamed: 0	community_area_number	Area	<19y	20-34y	35-59y	50-64y	65-74y	75-84y	85y<	...
0	13.0	14.0	Albany Park	13354	13738	11975	8232	2329	1343	604	...
1	13.0	14.0	Albany Park	13354	13738	11975	8232	2329	1343	604	...
2	NaN	14.0	Albany Park	13354	13738	11975	8232	2329	1343	604	...
3	13.0	14.0	Albany Park	13354	13738	11975	8232	2329	1343	604	...
4	NaN	14.0	Albany Park	13354	13738	11975	8232	2329	1343	604	...

5 rows × 39 columns

We did the rest of the queries in mysql because that was much faster than jupyter queries of mysql. The respective files for queries and jupyter notebook are also available in the repository.

Load:

The final database called 'shot_chicago' is basically the representation of chicago city crime, flushot and demographic data from Chicago census and has these three tables. They all have the "community area number" which it is linked with.

Purpose: Our purpose was to have an overall sense of how these demographic factors are linked with the crime in the city.

We started the project to see the effect of flushots on demographics and the crime in the city with this idea in mind that education or median income might have a positive impact on flushot takers. We haven't yet found a credible source of flushots taken in the limited time. The current data has flushot centers which should be directly proportional to the flushot capacity, which we understand, is a far-fetched assumption.

The query (available in query file "***ETLtables***") helps creation of various views in which we study the following:

a) The number of flushot centres in each area compared to the population living in that area
b) Exported view in centre_population.csv
c) The number of crimes conducted in relation with employment and median income of the area

d) Crime Rate (normalised by total population of the area) in relation with the employment and median income of the area
e) Employment studied as percent employed -- defined by proportion of people working who are in labor force
f) Employment studied as percent working -- defined by proportion of people working out of the total population living in that area. Views available in `crime_employment.csv`
g) Finding the link between the Weapons Violation incidents -- Shooting in chicago and the flushots and education in the area. There is a big diffusion of crimes w.r.t person doing it vs the area they choose to do it and hence this data might not be very relevant. A link to policing might give better insight. Views available in `centre_population.csv`