Link : https://uofi.box.com/s/zf2vu1pqygt9f4izvsbp21811j8lgz01

Gun Violence in USA

Smriiti Singhal, Sayed Shazeb

Introduction and Overview

While we were looking for datasets for the data cleaning project we wanted to work on dataset which is very “dirty” as well as useful for data analysis. After we clean the data we should able to use it for further analysis as it is the whole point of cleaning the data. So, we found Gun Violence incidents data on Kaggle and we thought that this will perfect for our project as checks both of our conditions.

We found this dataset on Kaggle, but it was compiled by gunviolencearchive.org which is a non-profit organization, they keep the database related to such type of incidents from the news articles. The dataset has around 240k Gun violence incidents from January 2013 and March 2018 with 29 features.

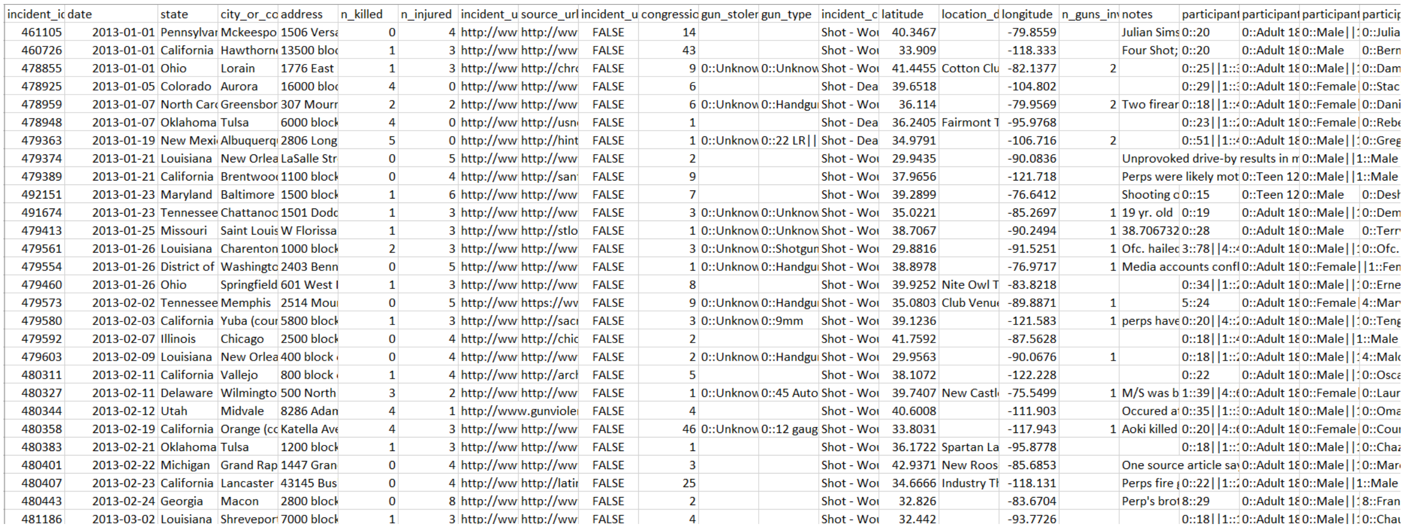
Gun Violence incidents nowadays are very prevalent in USA, more than so in any other country except for the war-stricken countries. What are the main causes for these types of these incidents? How can we prevent it? How many people are generally involved in these kinds of incidents? Do they target specific community? What is the most common type of guns used? Is it easier to get than other guns? Which states or cities in USA have these types of incidents more common? How many of the suspects are male or female? Are these types of incidents increasing in the USA? Are there any particular months or days when these incidents occur more often? All these kinds of question came into our mind when we found these data. So, we decided to work on this data to see how many questions we can answer and we might also find trends we were not looking for?

We wanted to see if there are any patterns emerging from these kinds of incidents and further identify the dependencies these incidents have with any feature given in the dataset. However, most important reason was this dataset wasn’t capable of data analysis if one wouldn’t clean the data first. So, to summarize our aim in sentence our main goal was to first clean the data and then do further analysis from these data and see if we find some interesting insights.

INITIAL ASSESMENT AND THE USE CASE

As I mentioned above our dataset consist of around 240k gun violence incidents and it has 29 columns which includes, incident, date, state, city or county, n_killed, n_injured, congressional district etc.

When we first looked the at the dataset, we immediately realized that this dataset is not useful for some in-depth data analysis unless we don't clean the data first. Below is the initial screenshot of the data

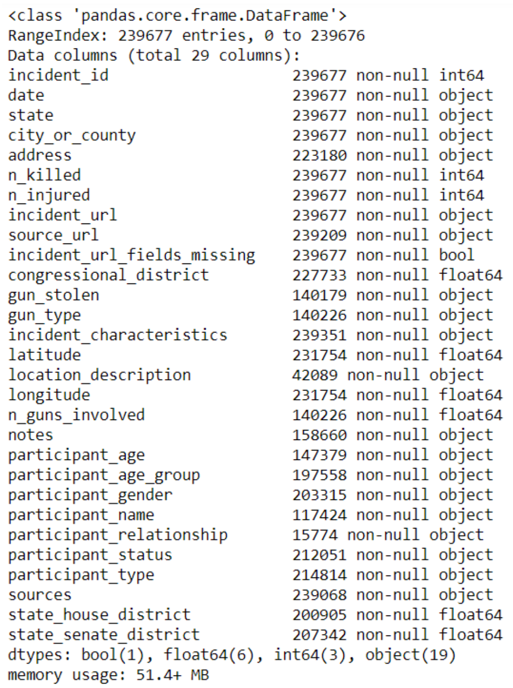


incident_ic	date	state	city_or_cc	address	n_killed	n_injured	incident_u	source_url	incident_u	congressic	gun_stoler	gun_type	incident_c	latitude	location_d	longitude	n_guns_in	notes	participant	participant	participant	particip
461105	2013-01-01	Pennsylvar	Mckeespo	1506 Versa	0	4	http://ww	http://ww	FALSE	14			Shot - Wou	40.3467		-79.8559		Julian Sims	0::20	0::Adult 18	0::Male||1	0::Julia
460726	2013-01-01	California	Hawthorne	13500 bloc	1	3	http://ww	http://ww	FALSE	43			Shot - Wou	33.909		-118.333		Four Shot;	0::20	0::Adult 18	0::Male	0::Bern
478855	2013-01-01	Ohio	Lorain	1776 East	1	3	http://ww	http://chrc	FALSE	9	0::Unknow	0::Unknow	Shot - Wou	41.4455	Cotton Clu	-82.1377	2		0::25||1::	0::Adult 18	0::Male||1	0::Dam
478925	2013-01-05	Colorado	Aurora	16000 bloc	4	0	http://ww	http://ww	FALSE	6			Shot - Dea	39.6518		-104.802			0::29||1::	0::Adult 18	0::Female|	0::Stac
478959	2013-01-07	North Carc	Greensbor	307 Mourr	2	2	http://ww	http://ww	FALSE	6	0::Unknow	0::Handgu	Shot - Wou	36.114		-79.9569	2	Two firear	0::18||1::	0::Adult 18	0::Female|	0::Dani
478948	2013-01-07	Oklahoma	Tulsa	6000 block	4	0	http://ww	http://usn	FALSE	1			Shot - Dea	36.2405	Fairmont T	-95.9768			0::23||1::	0::Adult 18	0::Female|	0::Rebe
479363	2013-01-19	New Mexi	Albuquerq	2806 Long	5	0	http://ww	http://hint	FALSE	1	0::Unknow	0::22 LR||	Shot - Dea	34.9791		-106.716	2		0::51||1::	0::Adult 18	0::Male||1	0::Greg
479374	2013-01-21	Louisiana	New Orlea	LaSalle Str	0	5	http://ww	http://ww	FALSE	2			Shot - Wou	29.9435		-90.0836		Unprovoked drive-by results in m			0::Male||1::Male	
479389	2013-01-21	California	Brentwoo	1100 block	0	4	http://ww	http://san	FALSE	9			Shot - Wou	37.9656		-121.718		Perps were likely mot		0::Teen 12	0::Male||1::Male	
492151	2013-01-23	Maryland	Baltimore	1500 block	1	6	http://ww	http://ww	FALSE	7			Shot - Wou	39.2899		-76.6412		Shooting c	0::15	0::Teen 12	0::Male	0::Desh
491674	2013-01-23	Tennessee	Chattanoc	1501 Dodc	1	3	http://ww	http://ww	FALSE	3	0::Unknow	0::Unknow	Shot - Wou	35.0221		-85.2697	1	19 yr. old	0::19	0::Adult 18	0::Male||1	0::Dem
479413	2013-01-25	Missouri	Saint Louis	W Florissa	1	3	http://ww	http://stlo	FALSE	1	0::Unknow	0::Unknow	Shot - Wou	38.7067		-90.2494	1	38.706732	0::28	0::Adult 18	0::Male	0::Terr
479561	2013-01-26	Louisiana	Charenton	1000 block	2	3	http://ww	http://ww	FALSE	3	0::Unknow	0::Shotgun	Shot - Wou	29.8816		-91.5251	1	Ofc. hailec	3::78||4::	0::Adult 18	0::Male||1	0::Ofc.
479554	2013-01-26	District of	Washingtc	2403 Benn	0	5	http://ww	http://ww	FALSE	1	0::Unknow	0::Handgu	Shot - Wou	38.8978		-76.9717	1	Media accounts confl		0::Adult 18	0::Female||1::Fen	
479460	2013-01-26	Ohio	Springfield	601 West I	1	3	http://ww	http://ww	FALSE	8			Shot - Wou	39.9252	Nite Owl T	-83.8218			0::34||1::	0::Adult 18	0::Male||1	0::Erne
479573	2013-02-02	Tennessee	Memphis	2514 Mou	0	5	http://ww	https://wv	FALSE	9	0::Unknow	0::Handgu	Shot - Wou	35.0803	Club Venue	-89.8871	1		5::24	0::Adult 18	0::Female|	4::Mar
479580	2013-02-03	California	Yuba (cour	5800 block	1	3	http://ww	http://sacı	FALSE	3	0::Unknow	0::9mm	Shot - Wou	39.1236		-121.583	1	perps have	0::20||4::	0::Adult 18	0::Male||1	0::Teng
479592	2013-02-07	Illinois	Chicago	2500 block	0	4	http://ww	http://chic	FALSE	2			Shot - Wou	41.7592		-87.5628			0::18||1::	0::Adult 18	0::Male||1::Male	
479603	2013-02-09	Louisiana	New Orlea	400 block	0	4	http://ww	http://ww	FALSE	2	0::Unknow	0::Handgu	Shot - Wou	29.9563		-90.0676	1		0::18||1::	0::Adult 18	0::Male||1	4::Malc
480311	2013-02-11	California	Vallejo	800 block	1	4	http://ww	http://arch	FALSE	5			Shot - Wou	38.1072		-122.228			0::22	0::Adult 18	0::Male||1	0::Osca
480327	2013-02-11	Delaware	Wilmingto	500 North	3	2	http://ww	http://ww	FALSE	1	0::Unknow	0::45 Auto	Shot - Wou	39.7407	New Castl	-75.5499	1	M/S was b	1::39||4::	0::Adult 18	0::Female|	0::Laur
480344	2013-02-12	Utah	Midvale	8286 Adan	4	1	http://www.gunviole		FALSE	4			Shot - Wou	40.6008		-111.903		Occured at	0::35||1::	0::Adult 18	0::Male||1	0::Oma
480358	2013-02-19	California	Orange (cc	Katella Ave	4	3	http://ww	http://ww	FALSE	46	0::Unknow	0::12 gaug	Shot - Wou	33.8031		-117.943	1	Aoki killed	0::20||4::	0::Adult 18	0::Female|	0::Cour
480383	2013-02-21	Oklahoma	Tulsa	1200 block	1	3	http://ww	http://ww	FALSE	1			Shot - Wou	36.1722	Spartan La	-95.8778			0::18||1::	0::Adult 18	0::Male||1	0::Chaz
480401	2013-02-22	Michigan	Grand Rap	1447 Gran	0	4	http://ww	http://ww	FALSE	3			Shot - Wou	42.9371	New Roos	-85.6853		One source article say		0::Adult 18	0::Male||1	0::Mar
480407	2013-02-23	California	Lancaster	43145 Bus	0	4	http://ww	http://latir	FALSE	25			Shot - Wou	34.6666	Industry Th	-118.131		Perps fire	0::22||1::	0::Adult 18	0::Male||1::Male	
480443	2013-02-24	Georgia	Macon	2800 block	0	8	http://ww	http://ww	FALSE	2			Shot - Wou	32.826		-83.6704		Perp's brot	8::29	0::Adult 18	0::Male||1	8::Fran
481186	2013-03-02	Louisiana	Shrevepor	7000 block	1	3	http://ww	http://ww	FALSE	4			Shot - Wou	32.442		-93.7726			0::18||1::	0::Adult 18	0::Male||1	0::Chau

Figure 1

Figure 1 shows a part of dataset as it was impossible for us to get the full screenshot of the data with all the columns in it because of the number of columns. Here by just looking at the screenshot you can see the format of the columns like gun_type, gun_stolen, participant_age, participants etc. They are in a format which makes them completely useless. This dataset also had lot of missing values for some of the variables.

As for the schema, our dataset consists 19 string columns, 6 float columns, 3 integer columns and 1 Boolean column. Although, the data types of some of these columns are messed up. As you can see in the summary of schema in figure 2



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 239677 entries, 0 to 239676
Data columns (total 29 columns):
incident_id                    239677 non-null int64
date                           239677 non-null object
state                          239677 non-null object
city_or_county                 239677 non-null object
address                        223180 non-null object
n_killed                       239677 non-null int64
n_injured                      239677 non-null int64
incident_url                   239677 non-null object
source_url                     239209 non-null object
incident_url_fields_missing    239677 non-null bool
congressional_district         227733 non-null float64
gun_stolen                     140179 non-null object
gun_type                       140226 non-null object
incident_characteristics       239351 non-null object
latitude                       231754 non-null float64
location_description           42089 non-null object
longitude                      231754 non-null float64
n_guns_involved                140226 non-null float64
notes                          158660 non-null object
participant_age                147379 non-null object
participant_age_group          197558 non-null object
participant_gender             203315 non-null object
participant_name               117424 non-null object
participant_relationship       15774 non-null object
participant_status             212051 non-null object
participant_type               214814 non-null object
sources                        239068 non-null object
state_house_district           200905 non-null float64
state_senate_district          207342 non-null float64
dtypes: bool(1), float64(6), int64(3), object(19)
memory usage: 51.4+ MB
```

Figure 2

Incident_id shouldn't be int64 because calculating statistic for it doesn't tell us about anything and it doesn't really make sense to calculate any type of statistic on this column in the first place. Same thing goes for state_house_district, state_senate_district. Next, we have date column which is of type string and all the other pipe and colon including columns are also specified in string format.

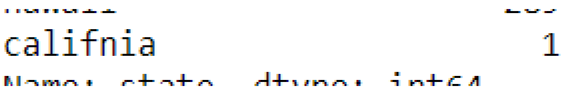
Use case:

Our initial use case was to extract the important information from these pipe and colon formatted columns which would make easy for us to answer the below questions –

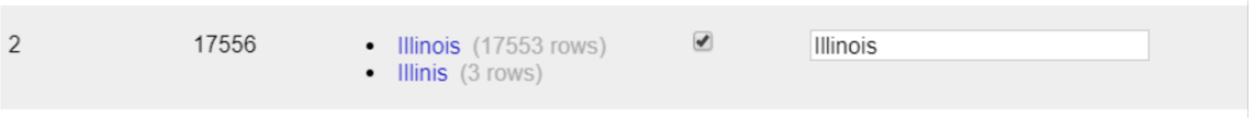
- How many males and females were involved in these incidents?
- How many victims were involved in an incident?
- How many suspects were there?
- What is the age group of the person involved in these incidents?
- What type of guns were involved?

Based on these use cases questions our goal for data cleaning was to extract the information from these pipe and colon formatted columns.

Although, on initial inspection the data looks that it'd be pretty easy to clean, but it's only when we started cleaning the data, we found out that it is not as simple as it looks. For example, state and city_or_county columns had lot of mistakes and inconsistencies with the spelling of the state and cities. For example, see in the below screen shot California and Illinois are misspelled.



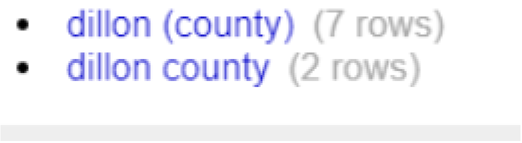
```
califnia                  1
```



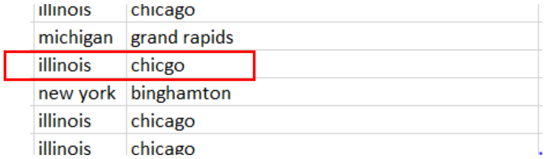
2	17556	• Illinois (17553 rows) • Illinis (3 rows)	☑	Illinois

Figure 4

Similarly, for city_or_county column there were many inconsistencies and in some cells the county was written with brackets and in some it was written without brackets and some spelling mistakes were also there as shown in the screenshot below



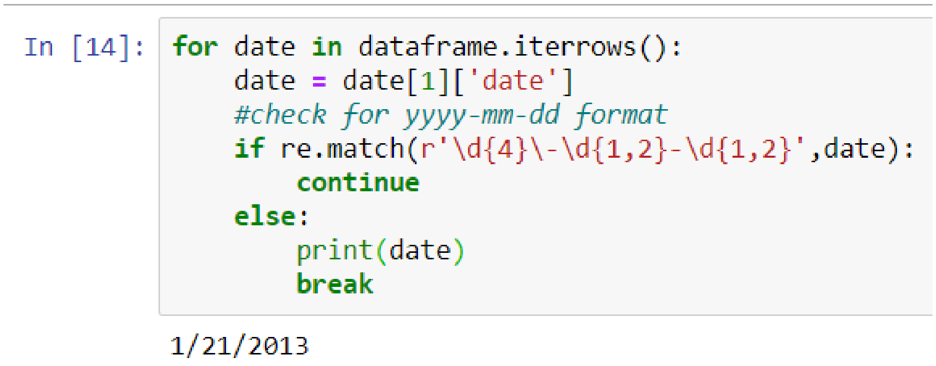
- dillon (county) (7 rows)
- dillon county (2 rows)



illinois	chicago
michigan	grand rapids
illinois	chicgo
new york	binghamton
illinois	chicago
illinois	chicago

Figure 5

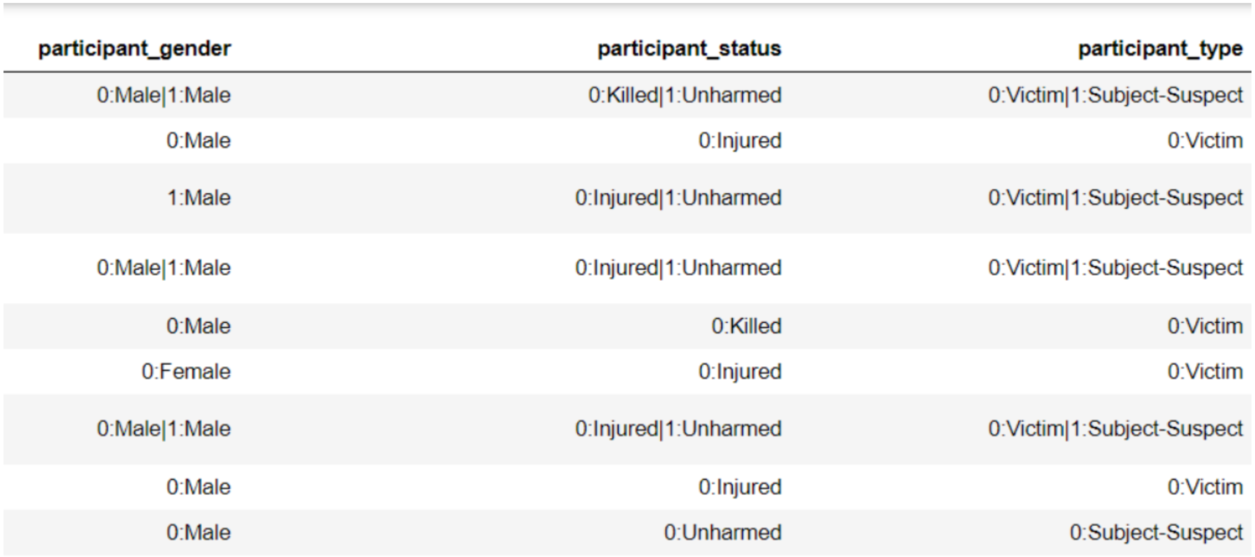
Next, we checked if we have consistent date format using some simple regular expression and we found there are some inconsistencies. Most of the dates are in form yyyy-mm-dd and it had few entries with format mm/dd/yyyy as shown in below screenshot



```
In [14]: for date in dataframe.iterrows():
             date = date[1]['date']
             #check for yyyy-mm-dd format
             if re.match(r'\d{4}\-\d{1,2}-\d{1,2}',date):
                 continue
             else:
                 print(date)
                 break

         1/21/2013
```

Figure 6

And just when we thought extracting the data from pipe and colon formatted data would be easy, we got to know that pipe and colon data is also inconsistent. As seen on figure 7, only one colon and pipe has been used as opposed to two pipes and two colons which is more common in these features. So, we had to take care of these things too.



participant_gender	participant_status	participant_type
0:Male|1:Male	0:Killed|1:Unharmed	0:Victim|1:Subject-Suspect
0:Male	0:Injured	0:Victim
1:Male	0:Injured|1:Unharmed	0:Victim|1:Subject-Suspect
0:Male|1:Male	0:Injured|1:Unharmed	0:Victim|1:Subject-Suspect
0:Male	0:Killed	0:Victim
0:Female	0:Injured	0:Victim
0:Male|1:Male	0:Injured|1:Unharmed	0:Victim|1:Subject-Suspect
0:Male	0:Injured	0:Victim
0:Male	0:Unharmed	0:Subject-Suspect

Figure 7

However, there were some use cases for which the data was already cleaned enough, these are listed below

- How many people killed in a state or a city due these incidents?

- How many people injured in a state or a city due these incidents?
- What is the exact location of the incidents? (can be derived from latitude and longitude column given in the dataset)
- How many guns were involved?

Now that we have listed the use cases which can be done without the data cleaning. There are some use cases which cannot be done even if we clean the data, they are

- How many of these incidents are gang related?
- How many of these incidents are result of some robbery?
- How many of them due to personal vendetta?
- How many of them were mass shootings?
- Whether these incidents were planned?
- If this incident was an act of self-defense?

One of the potential column to answer the above 3 or 4 questions is incident_characterstics but it had no fixed format and in some cases it was mentioned that if the incident is gang related or is mass shooting but in most cases it didn't mention about the type of shooting. Similarly, we can extract information related to the other questions from the notes column included in our data but it also have the same problem since it was written in an language format it has no consistent format and on the top of it there's lot of sparsity in this column so we wouldn't be able to extract the information for all the incidents.

Data cleaning methods and process

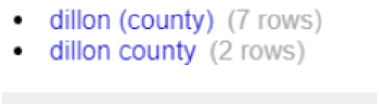
We performed data cleaning using two tools

1. Open Refine
2. Python

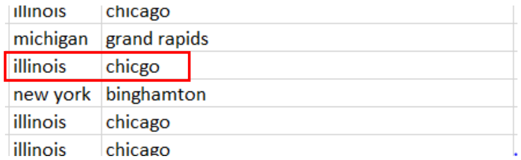
Using Open Refine

We used open refine because of its simple UI and powerful methods. It provides powerful methods to clean the data. Especially, the clustering capabilities it has, clustering the data which is supposed to be the same and as I mentioned above we had lot of spelling mistakes in our state and city_or_county column and also the inconsistencies related to the format in some columns. So, we decided it would be best if we use open refine for this.

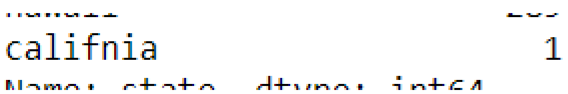
Below is the screenshot of the inconsistencies I am talking about



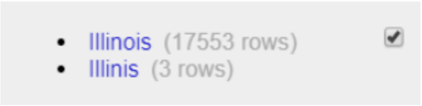
- dillon (county) (7 rows)
- dillon county (2 rows)



illinois	chicago
michigan	grand rapids
illinois	chicgo
new york	binghamton
illinois	chicago
illinois	chicago



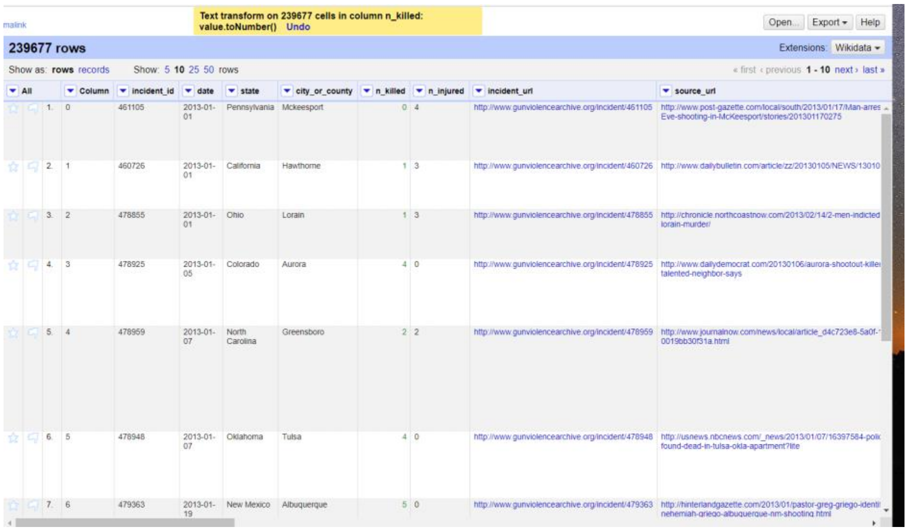
```
califnia                    1
Name: state, dtype: int64
```



- Illinois (17553 rows) ☑
- Illinis (3 rows)

And these are the below steps we took to clean the data in openrefine

1) Converted incident_id column to string format and change the data type of other columns to be appropriate data type



malink

Text transform on 239677 cells in column n_killed:
value.toNumber() Undo

Open... Export ▾ Help

239677 rows

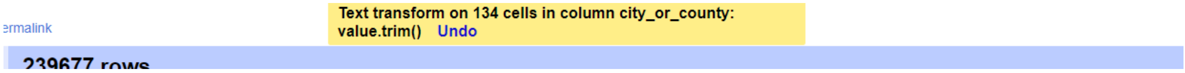
Extensions: Wikidata ▾

Show as: **rows** records   Show: 5 **10** 25 50 rows

« first ‹ previous **1 - 10** next › last »

All		Column	incident_id	date	state	city_or_county	n_killed	n_injured	incident_url	source_url
☆	1.	0	461105	2013-01-01	Pennsylvania	Mckeesport	0	4	http://www.gunviolencearchive.org/incident/461105	http://www.post-gazette.com/local/south/2013/01/17/Man-arres Eve-shooting-in-McKeesport/stories/201301170275
☆	2.	1	460726	2013-01-01	California	Hawthorne	1	3	http://www.gunviolencearchive.org/incident/460726	http://www.dailybulletin.com/article/zz/20130105/NEWS/13010
☆	3.	2	478855	2013-01-01	Ohio	Lorain	1	3	http://www.gunviolencearchive.org/incident/478855	http://chronicle.northcoastnow.com/2013/02/14/2-men-indicted lorain-murder/
☆	4.	3	478925	2013-01-05	Colorado	Aurora	4	0	http://www.gunviolencearchive.org/incident/478925	http://www.dailydemocrat.com/20130106/aurora-shootout-kille talented-neighbor-says
☆	5.	4	478959	2013-01-07	North Carolina	Greensboro	2	2	http://www.gunviolencearchive.org/incident/478959	http://www.journalnow.com/news/local/article_d4c723e8-5a0f- 0019bb30f31a.html
☆	6.	5	478948	2013-01-07	Oklahoma	Tulsa	4	0	http://www.gunviolencearchive.org/incident/478948	http://usnews.nbcnews.com/_news/2013/01/07/16397584-poli found-dead-in-tulsa-okla-apartment?lite
☆	7.	6	479363	2013-01-19	New Mexico	Albuquerque	5	0	http://www.gunviolencearchive.org/incident/479363	http://hinterlandgazette.com/2013/01/pastor-greg-griego-identi nehemiah-griego-albuquerque-nm-shooting.html

2) Removed the trailing and leading white spaces from the string-based columns



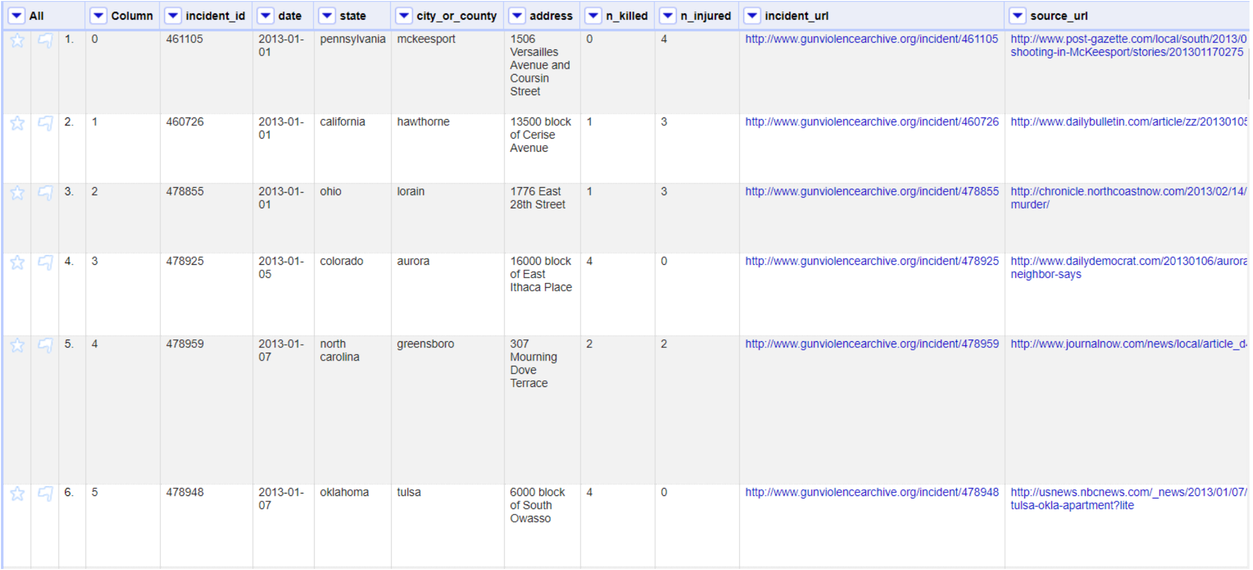
ermalink

Text transform on 134 cells in column city_or_county:
value.trim() Undo

239677 rows

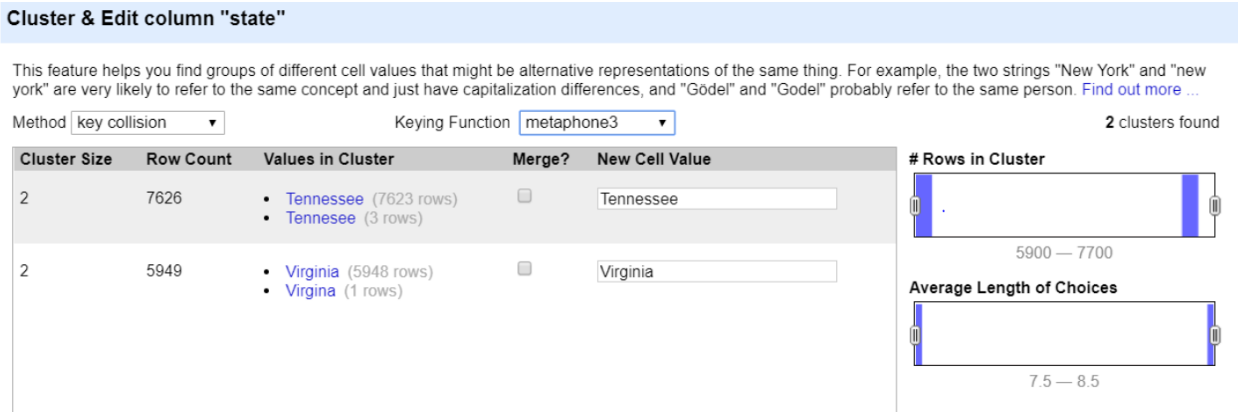
3) Removed the unwanted columns like address, incident_url, incident_url_fields_missing etc.

4) Changed the values of state and city_or_county column to lowercase



All		Column	incident_id	date	state	city_or_county	address	n_killed	n_injured	incident_url	source_url
☆	1.	0	461105	2013-01-01	pennsylvania	mckeesport	1506 Versailles Avenue and Coursin Street	0	4	http://www.gunviolencearchive.org/incident/461105	http://www.post-gazette.com/local/south/2013/0 shooting-in-McKeesport/stories/201301170275
☆	2.	1	460726	2013-01-01	california	hawthorne	13500 block of Cerise Avenue	1	3	http://www.gunviolencearchive.org/incident/460726	http://www.dailybulletin.com/article/zz/2013010
☆	3.	2	478855	2013-01-01	ohio	lorain	1776 East 28th Street	1	3	http://www.gunviolencearchive.org/incident/478855	http://chronicle.northcoastnow.com/2013/02/14/ murder/
☆	4.	3	478925	2013-01-05	colorado	aurora	16000 block of East Ithaca Place	4	0	http://www.gunviolencearchive.org/incident/478925	http://www.dailydemocrat.com/20130106/aurora neighbor-says
☆	5.	4	478959	2013-01-07	north carolina	greensboro	307 Mourning Dove Terrace	2	2	http://www.gunviolencearchive.org/incident/478959	http://www.journalnow.com/news/local/article_d
☆	6.	5	478948	2013-01-07	oklahoma	tulsa	6000 block of South Owasso	4	0	http://www.gunviolencearchive.org/incident/478948	http://usnews.nbcnews.com/_news/2013/01/07/ tulsa-okla-apartment?lite

5) Start clustering on state columns and merge the misspelled words using different method and keying function.

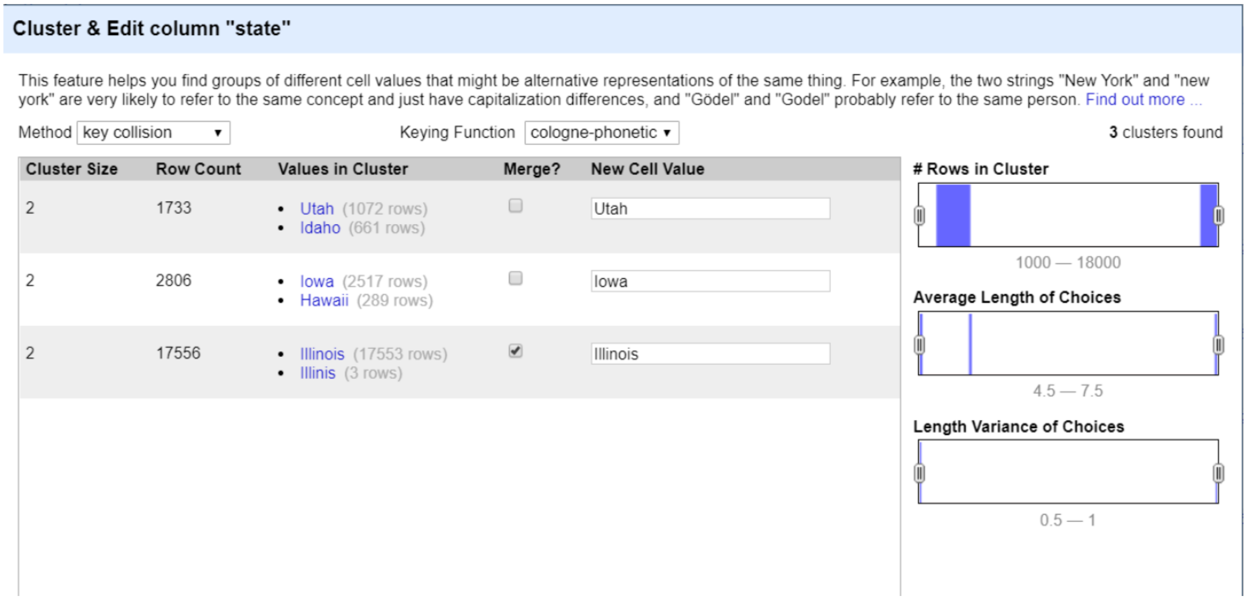


Cluster & Edit column "state"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more ...

Method: key collision ▾ Keying Function: metaphone3 ▾ **2** clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
2	7626	Tennessee (7623 rows) Tennesee (3 rows)	☐	Tennessee
2	5949	Virginia (5948 rows) Virgina (1 rows)	☐	Virginia



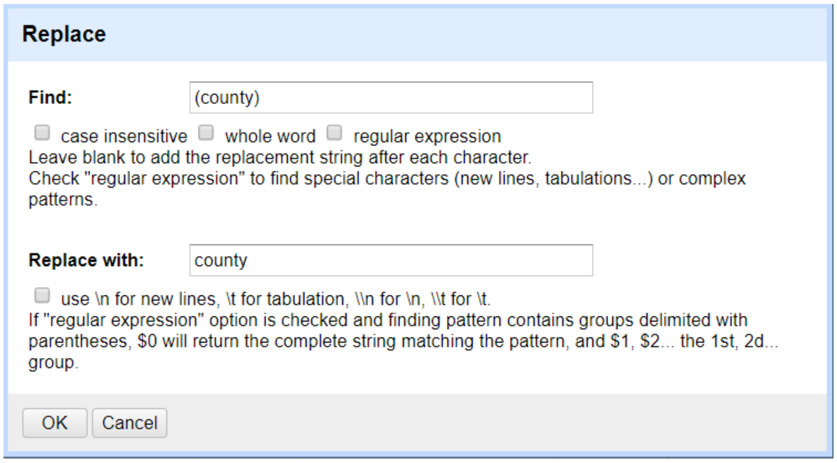
Cluster & Edit column "state"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more ...

Method: key collision ▾ Keying Function: cologne-phonetic ▾ **3** clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
2	1733	Utah (1072 rows) Idaho (661 rows)	☐	Utah
2	2806	Iowa (2517 rows) Hawaii (289 rows)	☐	Iowa
2	17556	Illinois (17553 rows) Illinis (3 rows)	☑	Illinois

6) Change the "(county)" format to "county" format in city_or_county column otherwise many clusters were formed.



Replace

Find: (county)

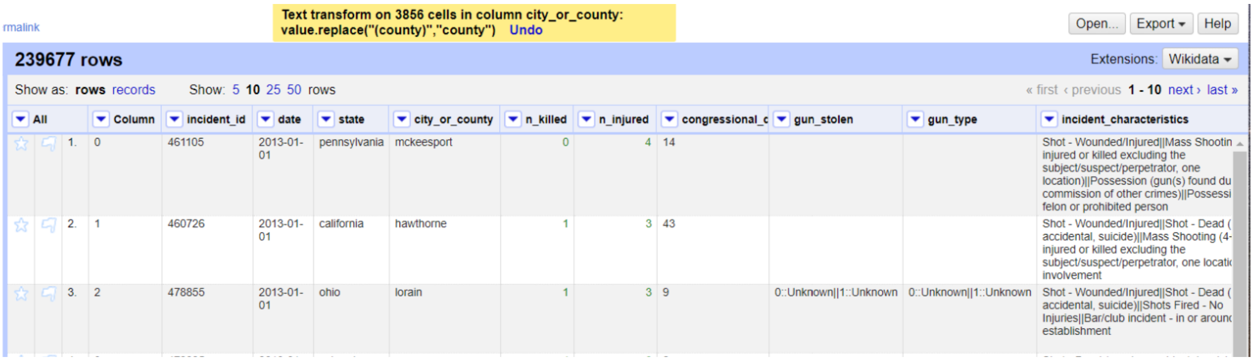
□ case insensitive □ whole word □ regular expression

Leave blank to add the replacement string after each character.
Check "regular expression" to find special characters (new lines, tabulations...) or complex patterns.

Replace with: county

□ use \n for new lines, \t for tabulation, \\n for \n, \\t for \t.

If "regular expression" option is checked and finding pattern contains groups delimited with parentheses, $0 will return the complete string matching the pattern, and $1, $2... the 1st, 2d... group.



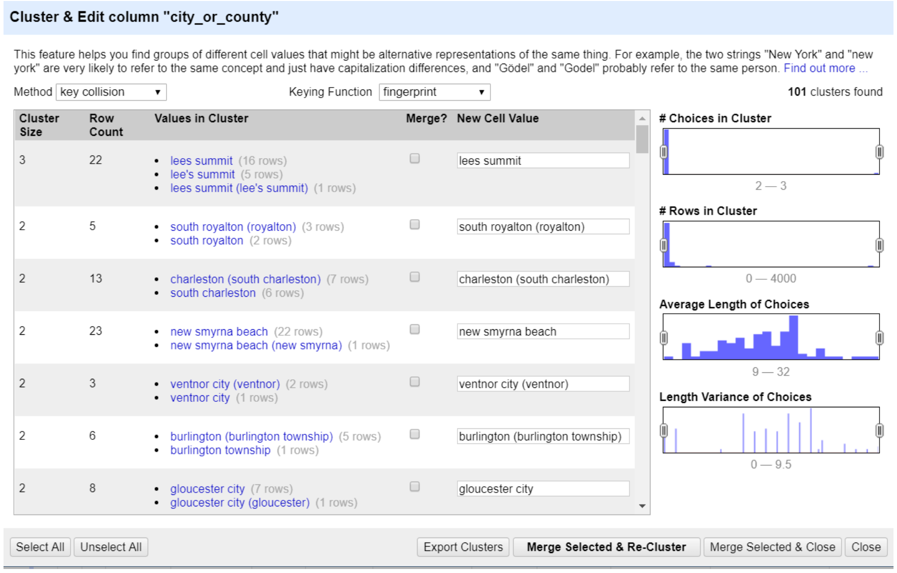
Text transform on 3856 cells in column city_or_county:
value.replace("(county)","county") Undo

239677 rows

Show as: **rows** records Show: 5 **10** 25 50 rows « first ‹ previous **1 - 10** next › last »

All			Column	incident_id	date	state	city_or_county	n_killed	n_injured	congressional_c	gun_stolen	gun_type	incident_characteristics
☆		1.	0	461105	2013-01-01	pennsylvania	mckeesport	0	4	14			Shot - Wounded/Injured||Mass Shootin injured or killed excluding the subject/suspect/perpetrator, one location)||Possession (gun(s) found du commission of other crimes)||Possessi felon or prohibited person
☆		2.	1	460726	2013-01-01	california	hawthorne	1	3	43			Shot - Wounded/Injured||Shot - Dead (accidental, suicide)||Mass Shooting (4- injured or killed excluding the subject/suspect/perpetrator, one locati involvement
☆		3.	2	478855	2013-01-01	ohio	lorain	1	3	9	0::Unknown||1::Unknown	0::Unknown||1::Unknown	Shot - Wounded/Injured||Shot - Dead (accidental, suicide)||Shots Fired - No Injuries||Bar/club incident - in or arounc establishment

7) Last step before we jump to python, cluster city_or_county column



Cluster & Edit column "city_or_county"

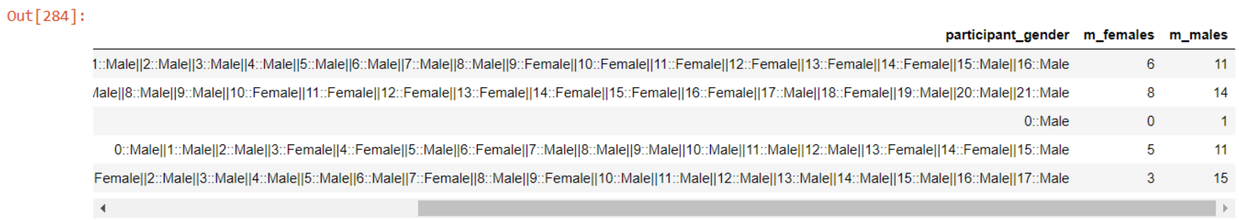
This feature helps you find groups of different cell values that might be alternative representations of the same thing. For example, the two strings "New York" and "new york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably refer to the same person. Find out more ...

Method: key collision ▾ Keying Function: fingerprint ▾ **101** clusters found

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
3	22	lees summit (16 rows) lee's summit (5 rows) lees summit (lee's summit) (1 rows)	□	lees summit
2	5	south royalton (royalton) (3 rows) south royalton (2 rows)	□	south royalton (royalton)
2	13	charleston (south charleston) (7 rows) south charleston (6 rows)	□	charleston (south charleston)
2	23	new smyrna beach (22 rows) new smyrna beach (new smyrna) (1 rows)	□	new smyrna beach
2	3	ventnor city (ventnor) (2 rows) ventnor city (1 rows)	□	ventnor city (ventnor)
2	6	burlington (burlington township) (5 rows) burlington township (1 rows)	□	burlington (burlington township)
2	8	gloucester city (7 rows) gloucester city (gloucester) (1 rows)	□	gloucester city

Using Python

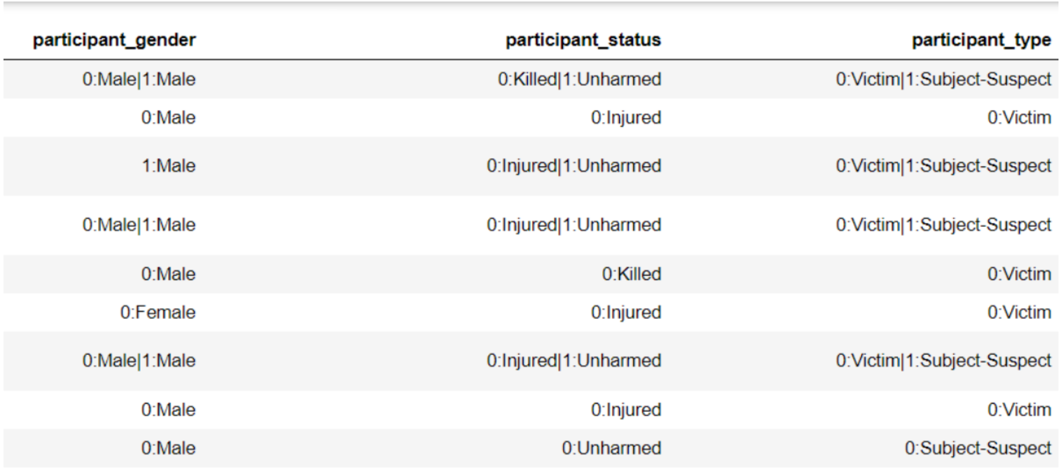
We started with 29 columns and now that we have dropped some columns and did the clustering, and after the cleaning in open refine we got 20 columns. Although, open refine is a good software, it lacks the complex methods that can be used to extract important information. For example, we want to extract number of males and females from participant gender like below.



Out[284]:

participant_gender	m_females	m_males
1::Male||2::Male||3::Male||4::Male||5::Male||6::Male||7::Male||8::Male||9::Female||10::Female||11::Female||12::Female||13::Female||14::Female||15::Male||16::Male	6	11
/ale||8::Male||9::Male||10::Female||11::Female||12::Female||13::Female||14::Female||15::Female||16::Female||17::Male||18::Female||19::Male||20::Male||21::Male	8	14
0::Male	0	1
0::Male||1::Male||2::Male||3::Female||4::Female||5::Male||6::Female||7::Male||8::Male||9::Male||10::Male||11::Male||12::Male||13::Female||14::Female||15::Male	5	11
Female||2::Male||3::Male||4::Male||5::Male||6::Male||7::Female||8::Male||9::Female||10::Male||11::Male||12::Male||13::Male||14::Male||15::Male||16::Male||17::Male	3	15

And as mentioned before the columns like participant_gender although mostly have double pipe and double colon format but some of the values in these columns had only one colon and one pipe for example see below



participant_gender	participant_status	participant_type
0:Male|1:Male	0:Killed|1:Unharmed	0:Victim|1:Subject-Suspect
0:Male	0:Injured	0:Victim
1:Male	0:Injured|1:Unharmed	0:Victim|1:Subject-Suspect
0:Male|1:Male	0:Injured|1:Unharmed	0:Victim|1:Subject-Suspect
0:Male	0:Killed	0:Victim
0:Female	0:Injured	0:Victim
0:Male|1:Male	0:Injured|1:Unharmed	0:Victim|1:Subject-Suspect
0:Male	0:Injured	0:Victim
0:Male	0:Unharmed	0:Subject-Suspect

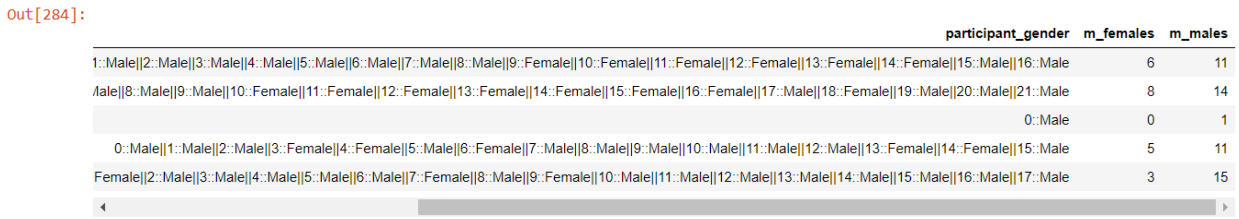
So, we really can't do use splitting method which we first thought of using OpenRefine. Next, there were some spelling mistakes which didn't get caught while doing clustering for example, see below

califnia	1
california	16305
colorado	3201

Here the word 'califnia' didn't get caught in the cluster. We corrected this using python and changed this specific value to the correct value.

Now Below are the steps we used to perform the data cleaning using python

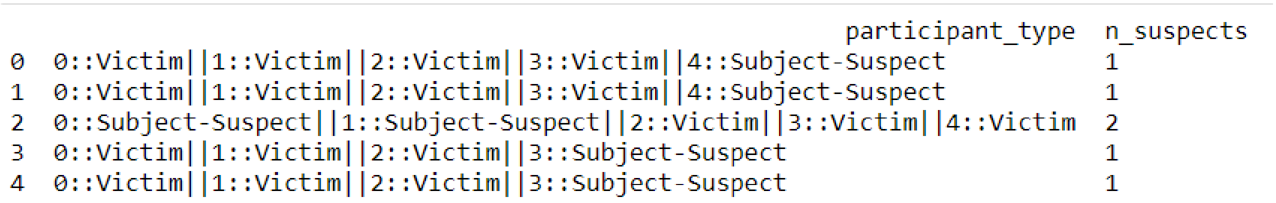
1) Fill null values with 0::Zero in participant_gender and participant type
2) Extract how many males and females are there using participant_gender column



Out[284]:

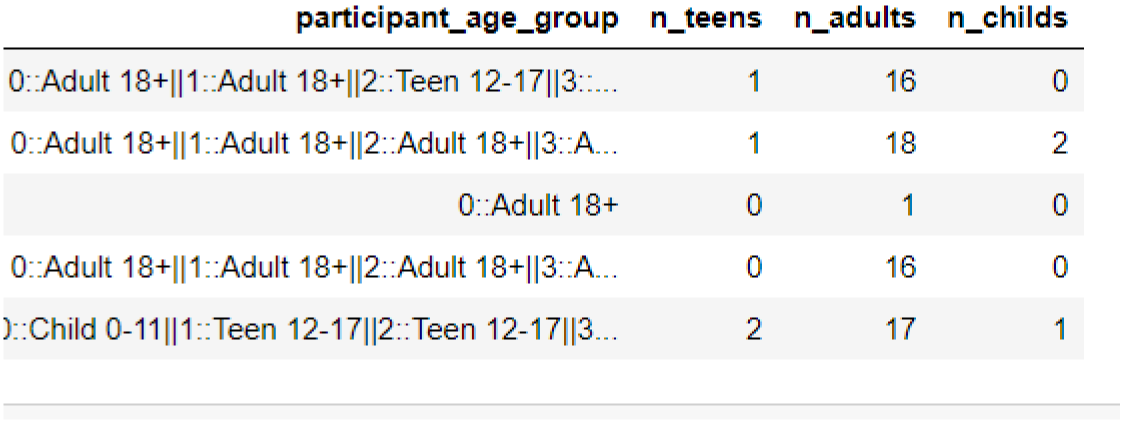
participant_gender	m_females	m_males
1::Male||2::Male||3::Male||4::Male||5::Male||6::Male||7::Male||8::Male||9::Female||10::Female||11::Female||12::Female||13::Female||14::Female||15::Male||16::Male	6	11
/ale||8::Male||9::Male||10::Female||11::Female||12::Female||13::Female||14::Female||15::Female||16::Female||17::Male||18::Female||19::Male||20::Male||21::Male	8	14
0::Male	0	1
0::Male||1::Male||2::Male||3::Female||4::Female||5::Male||6::Female||7::Male||8::Male||9::Male||10::Male||11::Male||12::Male||13::Female||14::Female||15::Male	5	11
Female||2::Male||3::Male||4::Male||5::Male||6::Male||7::Female||8::Male||9::Female||10::Male||11::Male||12::Male||13::Male||14::Male||15::Male||16::Male||17::Male	3	15

3) Count number of suspects in the incident using participant type



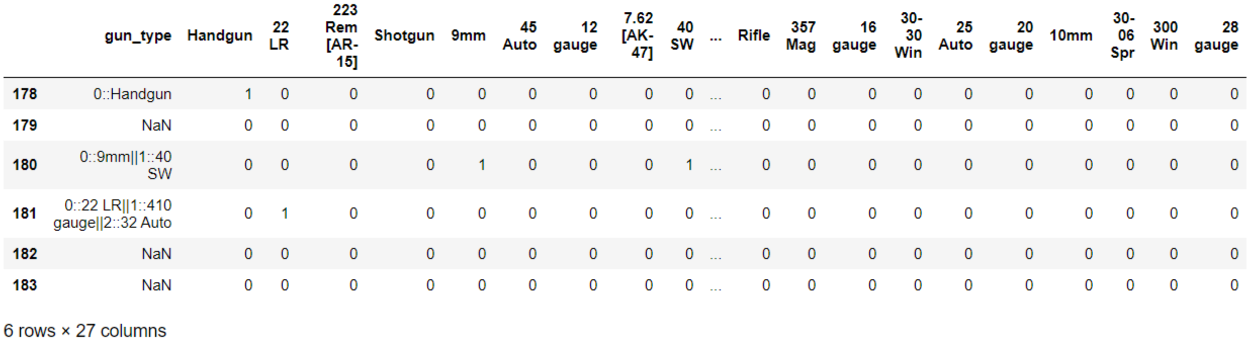
```
                                          participant_type  n_suspects
0  0::Victim||1::Victim||2::Victim||3::Victim||4::Subject-Suspect         1
1  0::Victim||1::Victim||2::Victim||3::Victim||4::Subject-Suspect         1
2  0::Subject-Suspect||1::Subject-Suspect||2::Victim||3::Victim||4::Victim  2
3  0::Victim||1::Victim||2::Victim||3::Subject-Suspect                    1
4  0::Victim||1::Victim||2::Victim||3::Subject-Suspect                    1
```

4) Count number of childs, adults and teens involved in the incident



participant_age_group	n_teens	n_adults	n_childs
0::Adult 18+||1::Adult 18+||2::Teen 12-17||3::...	1	16	0
0::Adult 18+||1::Adult 18+||2::Adult 18+||3::A...	1	18	2
0::Adult 18+	0	1	0
0::Adult 18+||1::Adult 18+||2::Adult 18+||3::A...	0	16	0
)::Child 0-11||1::Teen 12-17||2::Teen 12-17||3...	2	17	1

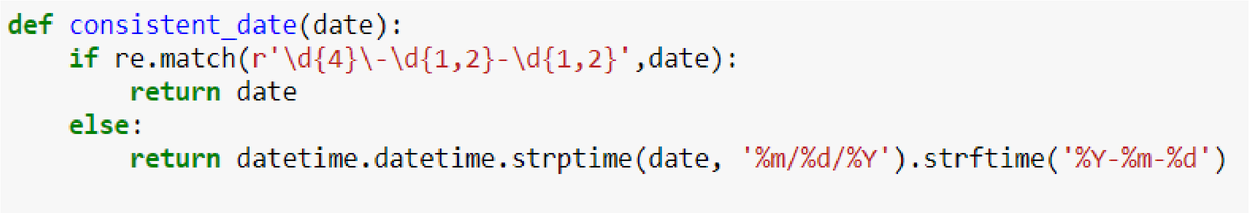
5) How many types of guns were used in each incident?



	gun_type	Handgun	22 LR	223 Rem [AR-15]	Shotgun	9mm	45 Auto	12 gauge	7.62 [AK-47]	40 SW	...	Rifle	357 Mag	16 gauge	30-30 Win	25 Auto	20 gauge	10mm	30-06 Spr	300 Win	28 gauge
178	0::Handgun	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
179	NaN	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
180	0::9mm||1::40 SW	0	0	0	0	1	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
181	0::22 LR||1::410 gauge||2::32 Auto	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
182	NaN	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
183	NaN	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

6 rows × 27 columns

6) Lastly, Make the dates consistent in yyyy-mm-dd format



```
def consistent_date(date):
    if re.match(r'\d{4}\-\d{1,2}-\d{1,2}',date):
        return date
    else:
        return datetime.datetime.strptime(date, '%m/%d/%Y').strftime('%Y-%m-%d')
```

For all the above operations we extensively used regular expressions. We also used pandas for adding new columns and applying inbuilt functions of this library. We will be attaching the link of my data cleaning notebook at the end of this document.

So, finally we cleaned the data we wanted to and dropped the columns we didn't want to get a summary of columns we dropped and the new columns we created see below.

Initially : 29 columns

After Open refine: 21 columns

After Python : 41 columns

Dropped:

1) 'address',
2) 'congressional_district',
3) 'date',
4) 'gun_stolen',
5) 'gun_type',
6) 'incident_characteristics',
7) 'incident_url',
8) 'incident_url_fields_missing',
9) 'location_description',
10) 'notes',
11) 'participant_age',
12) 'participant_age_group',
13) 'participant_gender',
14) 'participant_name',
15) 'participant_relationship',
16) 'participant_status',
17) 'participant_type',
18) 'source_url',
19) 'sources',
20) 'state_house_district',
21) 'state_senate_district'

Added:

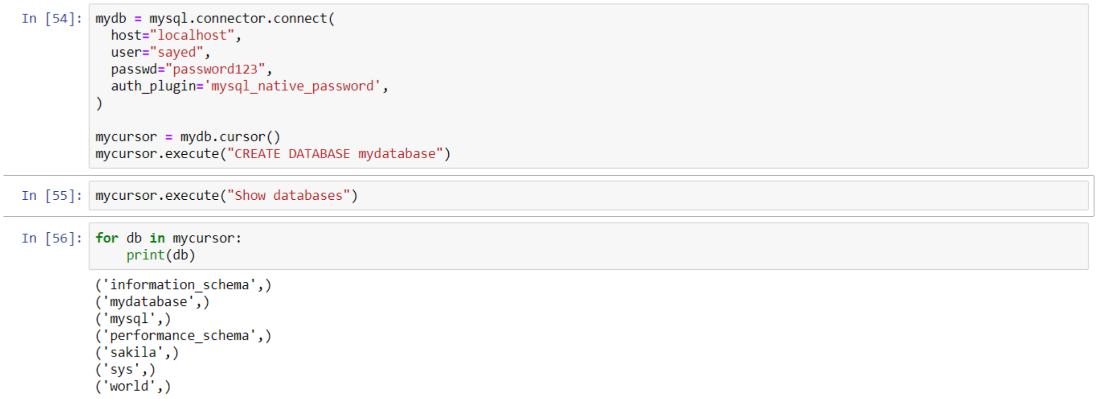
1) '10mm',
2) '12 gauge',

3) '16 gauge',
4) '20 gauge',
5) '22 LR',
6) '223 Rem [AR-15]',
7) '25 Auto',
8) '28 gauge',
9) '30-06 Spr',
10) '30-30 Win',
11) '300 Win',
12) '308 Win',
13) '32 Auto',
14) '357 Mag',
15) '38 Spl',
16) '380 Auto',
17) '40 SW',
18) '410 gauge',
19) '44 Mag',
20) '45 Auto',
21) '7.62 [AK-47]',
22) '9mm',
23) 'Handgun',
24) 'Other',
25) 'Rifle',
26) 'Shotgun',
27) 'Unnamed: 0',
28) 'consistent_date',
29) 'm_females',
30) 'm_males',
31) 'n_adults',
32) 'n_childs',
33) 'n_suspects',
34) 'n_teens'

Now that we have cleaned the data, we wanted to use Tableau visualization tool, to extract some insights. Tableau is a great tool for data visualization and business intelligence. It can be used to create some awesome dashboards. So, we imported the data into the tableau and then started analyzing the data visually.

Results

1. We have created the mysql database for this data and stored it in the table gun_violence. We used python library to interact with MySQL server and to create the data.



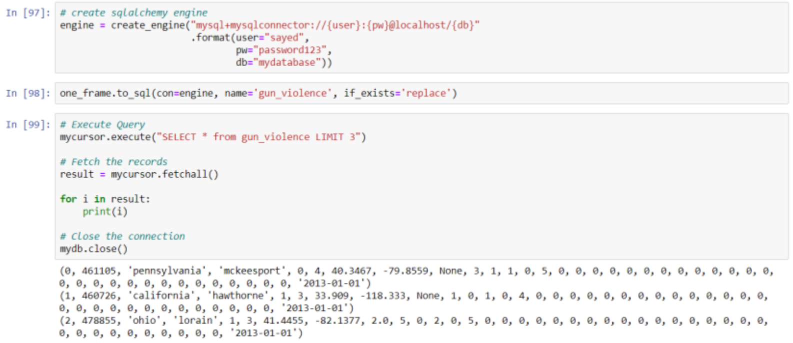
```
In [54]: mydb = mysql.connector.connect(
           host="localhost",
           user="sayed",
           passwd="password123",
           auth_plugin='mysql_native_password',
         )

         mycursor = mydb.cursor()
         mycursor.execute("CREATE DATABASE mydatabase")

In [55]: mycursor.execute("Show databases")

In [56]: for db in mycursor:
             print(db)

         ('information_schema',)
         ('mydatabase',)
         ('mysql',)
         ('performance_schema',)
         ('sakila',)
         ('sys',)
         ('world',)
```

Here you can see I setup the connection with MySQL server using mysql.connector.connect() method and then I created the database called ‘mydatabase’.

Next, to insert the results of our data cleaning, we used sql_alchemy module create_engine method and used the the code below to insert the dataframe into the database

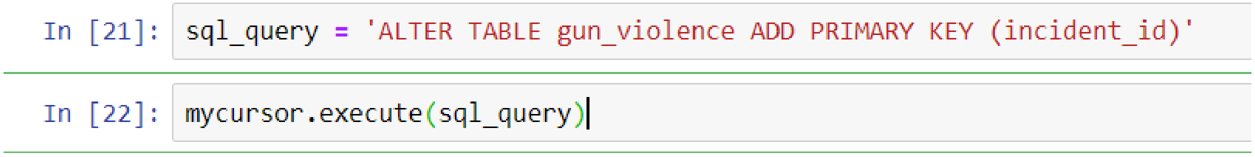
one_frame.to_sql(con=engine, name='gun_violence', if_exists='replace')

I saved the dataframe as a table in my database.



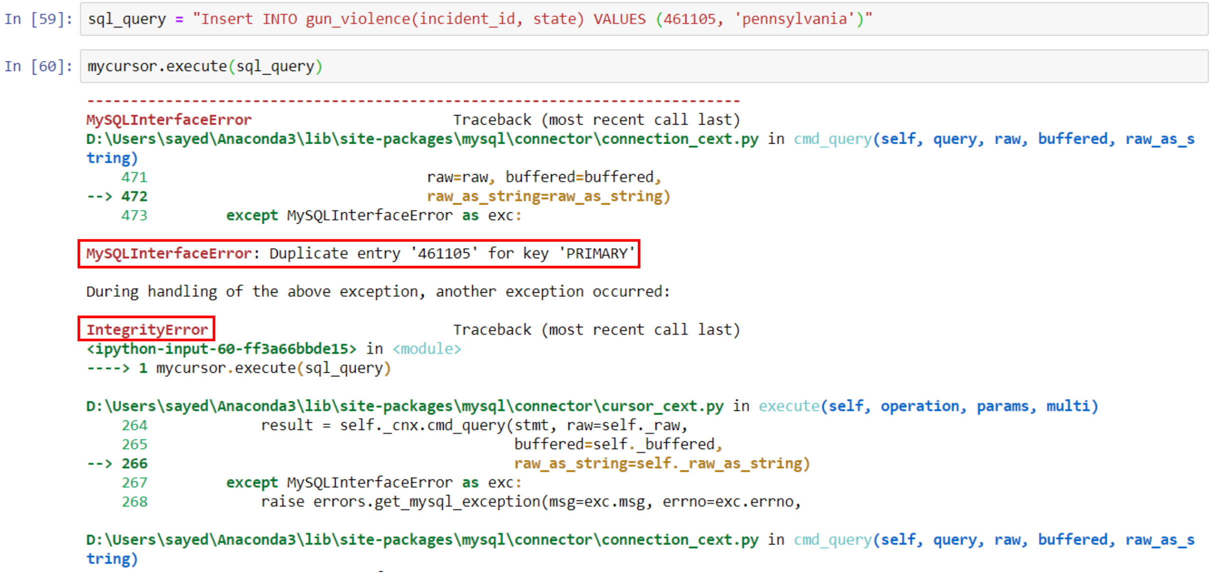
```
In [97]: # create sqlalchemy engine
         engine = create_engine("mysql+mysqlconnector://{user}:{pw}@localhost/{db}"
                                .format(user="sayed",
                                        pw="password123",
                                        db="mydatabase"))

In [98]: one_frame.to_sql(con=engine, name='gun_violence', if_exists='replace')

In [99]: # Execute Query
         mycursor.execute("SELECT * from gun_violence LIMIT 3")

         # Fetch the records
         result = mycursor.fetchall()

         for i in result:
             print(i)

         # Close the connection
         mydb.close()

         (0, 461105, 'pennsylvania', 'mckeesport', 0, 4, 40.3467, -79.8559, None, 3, 1, 1, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, '2013-01-01')
         (1, 460726, 'california', 'hawthorne', 1, 3, 33.909, -118.333, None, 1, 0, 1, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, '2013-01-01')
         (2, 478855, 'ohio', 'lorain', 1, 3, 41.4455, -82.1377, 2.0, 5, 0, 2, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, '2013-01-01')
```

After to enforce the primary key integrity constraint on incident_key I simply altered the table using below commands



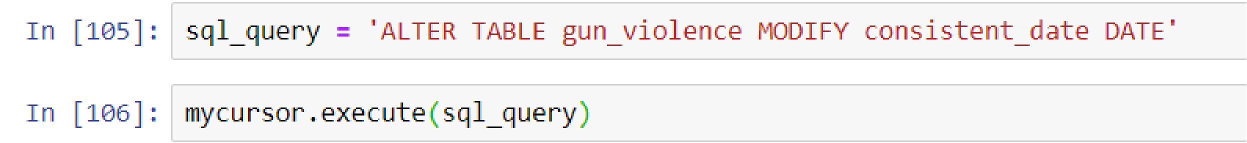
```
In [21]: sql_query = 'ALTER TABLE gun_violence ADD PRIMARY KEY (incident_id)'

In [22]: mycursor.execute(sql_query)
```

Next, to check whether the integrity constraint has been enforced I tried entering the adding the row with same primary key that is already there. And the result is shown below.



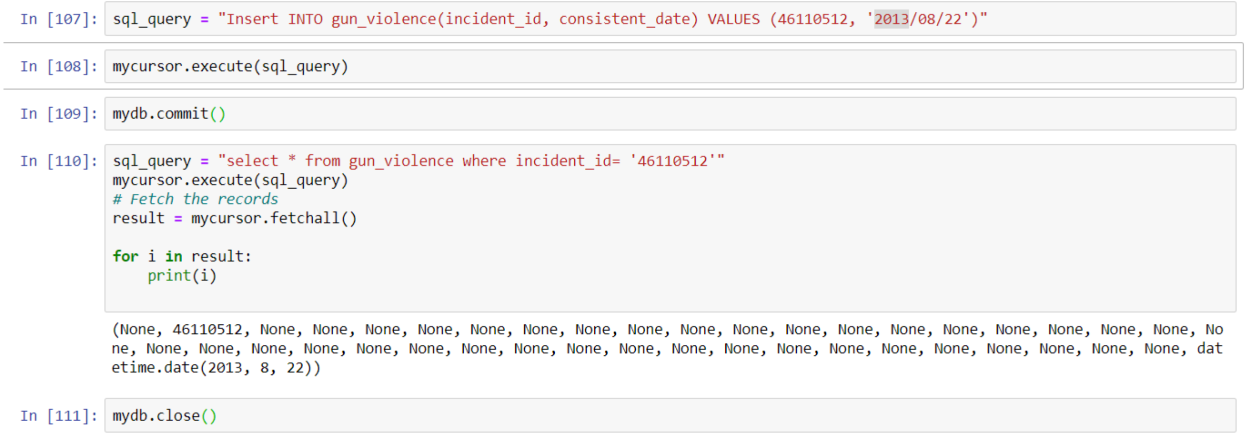
```
In [59]: sql_query = "Insert INTO gun_violence(incident_id, state) VALUES (461105, 'pennsylvania')"

In [60]: mycursor.execute(sql_query)

         ---------------------------------------------------------------------------
         MySQLInterfaceError                       Traceback (most recent call last)
         D:\Users\sayed\Anaconda3\lib\site-packages\mysql\connector\connection_cext.py in cmd_query(self, query, raw, buffered, raw_as_s
         tring)
             471                               raw=raw, buffered=buffered,
         --> 472                               raw_as_string=raw_as_string)
             473         except MySQLInterfaceError as exc:

         MySQLInterfaceError: Duplicate entry '461105' for key 'PRIMARY'

         During handling of the above exception, another exception occurred:

         IntegrityError                            Traceback (most recent call last)
         <ipython-input-60-ff3a66bbde15> in <module>
         ----> 1 mycursor.execute(sql_query)

         D:\Users\sayed\Anaconda3\lib\site-packages\mysql\connector\cursor_cext.py in execute(self, operation, params, multi)
             264             result = self._cnx.cmd_query(stmt, raw=self._raw,
             265                                          buffered=self._buffered,
         --> 266                                          raw_as_string=self._raw_as_string)
             267         except MySQLInterfaceError as exc:
             268             raise errors.get_mysql_exception(msg=exc.msg, errno=exc.errno,

         D:\Users\sayed\Anaconda3\lib\site-packages\mysql\connector\connection_cext.py in cmd_query(self, query, raw, buffered, raw_as_s
         tring)
```

As you can see when I tried to insert the incident_id which is our primary key same as the one that is already there. It showed me an Integrity error which was expected. Moreover, I altered the table to make the datatype of consistent_date column to be date using below query



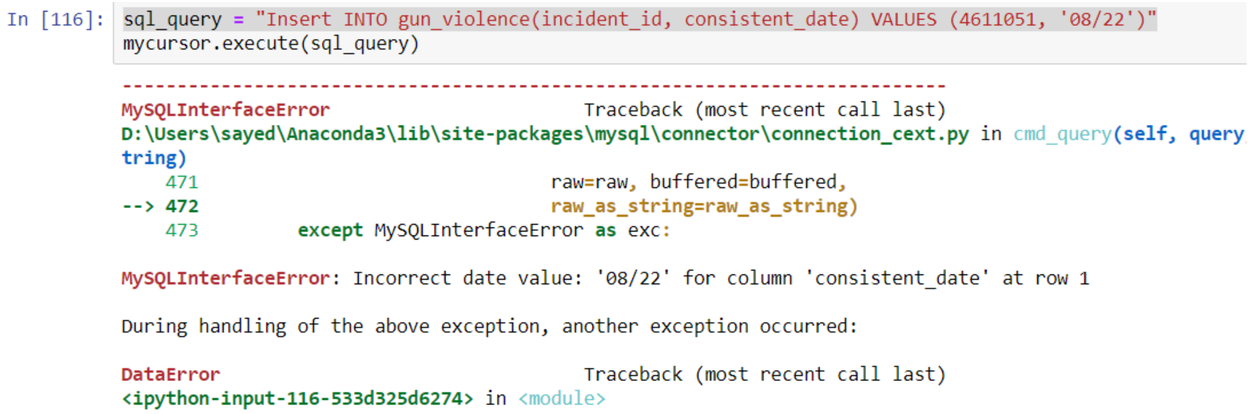
```
In [105]: sql_query = 'ALTER TABLE gun_violence MODIFY consistent_date DATE'

In [106]: mycursor.execute(sql_query)
```

The default date format for MySQL date datatype is yyyy-mm-dd. So, I intentionally tried adding the date with different format(yyyy/mm/dd) to check for domain constraints.



```
In [107]: sql_query = "Insert INTO gun_violence(incident_id, consistent_date) VALUES (46110512, '2013/08/22')"

In [108]: mycursor.execute(sql_query)

In [109]: mydb.commit()

In [110]: sql_query = "select * from gun_violence where incident_id= '46110512'"
          mycursor.execute(sql_query)
          # Fetch the records
          result = mycursor.fetchall()

          for i in result:
              print(i)

          (None, 46110512, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, No
          ne, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, dat
          etime.date(2013, 8, 22))

In [111]: mydb.close()
```

As you can see the date automatically changed to the correct format and inserted into the database. Now I tried entering a simple string without adding the year part in the sql query as shown below

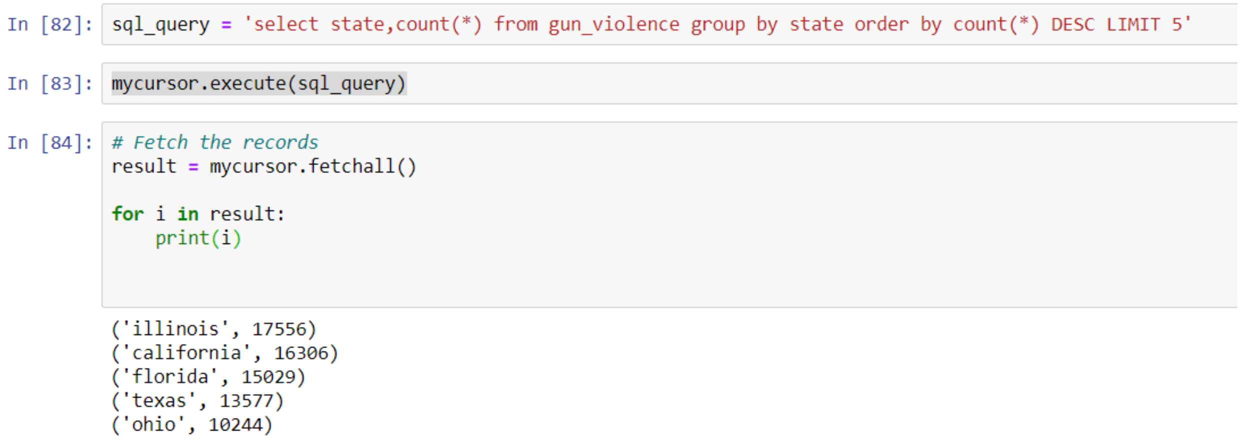


```
In [116]: sql_query = "Insert INTO gun_violence(incident_id, consistent_date) VALUES (4611051, '08/22')"
          mycursor.execute(sql_query)

          ---------------------------------------------------------------------------
          MySQLInterfaceError                       Traceback (most recent call last)
          D:\Users\sayed\Anaconda3\lib\site-packages\mysql\connector\connection_cext.py in cmd_query(self, query
          tring)
              471                                raw=raw, buffered=buffered,
          --> 472                                raw_as_string=raw_as_string)
              473             except MySQLInterfaceError as exc:

          MySQLInterfaceError: Incorrect date value: '08/22' for column 'consistent_date' at row 1

          During handling of the above exception, another exception occurred:

          DataError                                 Traceback (most recent call last)
          <ipython-input-116-533d325d6274> in <module>
```

As you can see, I got the huge error saying I’ve input the incorrect date value.

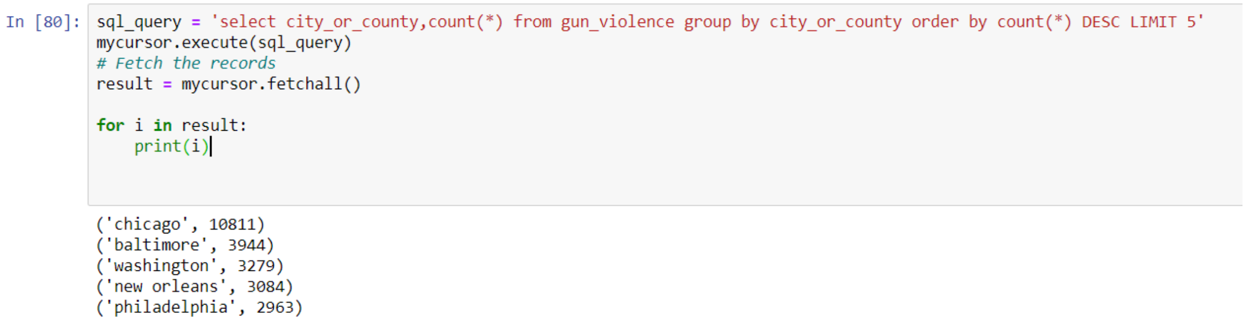
Now let us execute some SQL queries to profile the dataset

For example. To see what are top 5 state that has highest number of gun violence incident in USA?



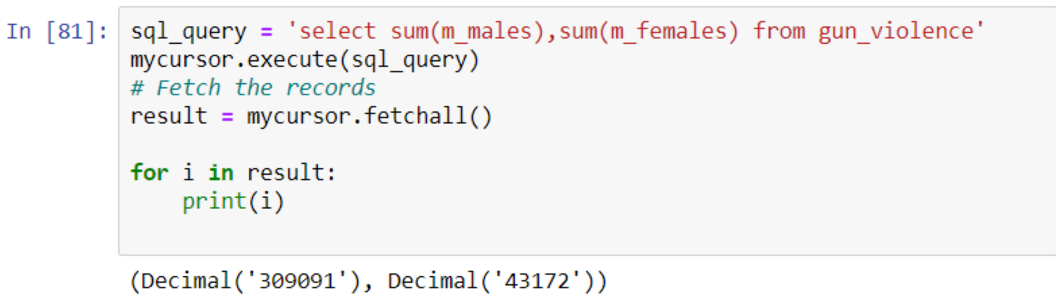
```
In [82]: sql_query = 'select state,count(*) from gun_violence group by state order by count(*) DESC LIMIT 5'

In [83]: mycursor.execute(sql_query)

In [84]: # Fetch the records
         result = mycursor.fetchall()

         for i in result:
             print(i)

         ('illinois', 17556)
         ('california', 16306)
         ('florida', 15029)
         ('texas', 13577)
         ('ohio', 10244)
```

As you can see Illinois has highest number of gun violence related incidents, followed by California, florida, Texas and ohio. Similarly, if we want to see what are the top 5 cities in who has highest number of gun violence incidents?



```
In [80]: sql_query = 'select city_or_county,count(*) from gun_violence group by city_or_county order by count(*) DESC LIMIT 5'
         mycursor.execute(sql_query)
         # Fetch the records
         result = mycursor.fetchall()

         for i in result:
             print(i)

         ('chicago', 10811)
         ('baltimore', 3944)
         ('washington', 3279)
         ('new orleans', 3084)
         ('philadelphia', 2963)
```

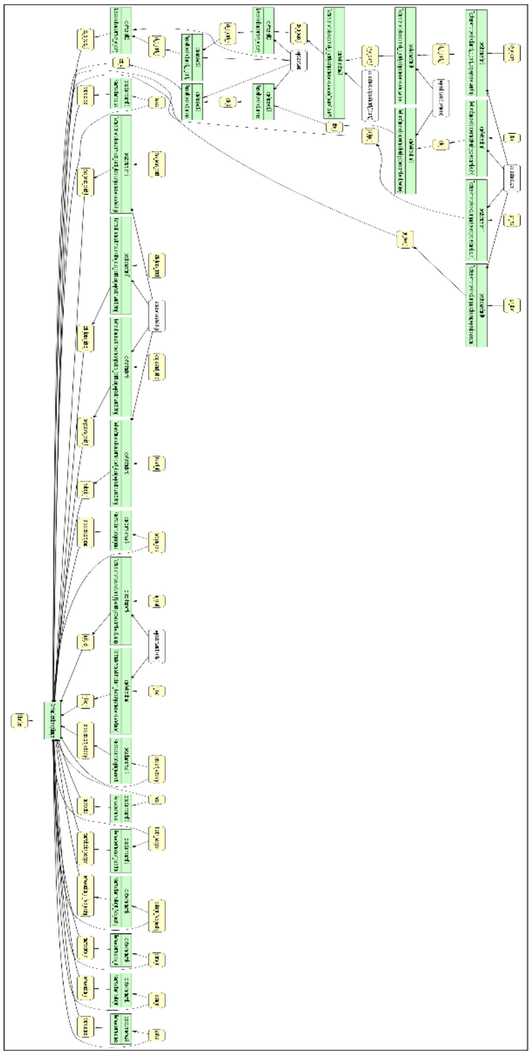
One more example, how many males and females were involved in such incidents?



```
In [81]: sql_query = 'select sum(m_males),sum(m_females) from gun_violence'
         mycursor.execute(sql_query)
         # Fetch the records
         result = mycursor.fetchall()

         for i in result:
             print(i)

         (Decimal('309091'), Decimal('43172'))
```

This is how many males and females involved in the incident

OPENREFINE WORKFLOW



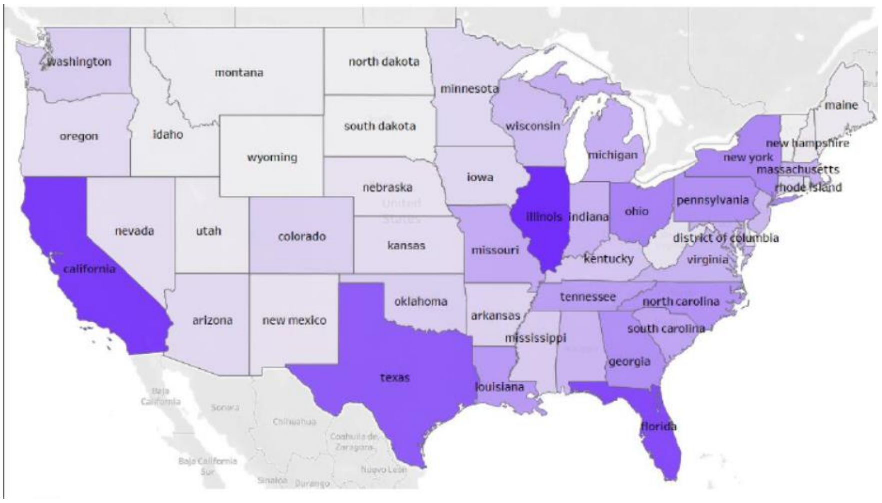
Input: comma separated value dataset

Dependencies: python, pandas, numpy, sql_connector

Changes Table:

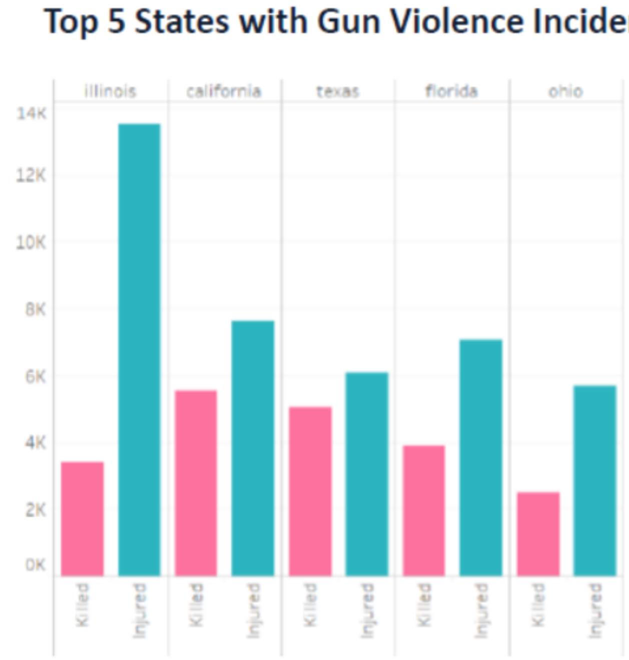
Operation	Column changed
value.toString()	incident_id
value.trim()	state
value.trim()	city_or_county
remove	address
value.toNumber()	n_killed
value_toNumber()	n_injured
remove	incident_url
remove	source_url
remove	incident_url_field_missing
remove	location_description
remove	notes
remove	participant_relationship
remove	participant_name
remove	sources
value.toLowercase()	city_or_county
value.toLowercase()	state
cluster and merge	state(31131 cells)
replace '(county)' with 'county'	city_or_county(3856 cells)
cluster and merge	city_or_county(9551 cells)
remove	incident_characteristics
changed the spelling of 'calfnia' to 'california'	state(1 cell)
fill na values	participant_type
fill na values	participant_gender
calculate number of males and females	participant_gender : m_males, m_females
calculate number of suspects	participant_type : n_suspects
calculate number of children, adults or teens involved	participant_age_group: n_teens, n_adults, n_childs
calculate number of gun categories	'gun_type','Handgun', '22 LR', '223 Rem [AR-15]', 'Shotgun', '9mm', '45 Auto',

Tableau Results

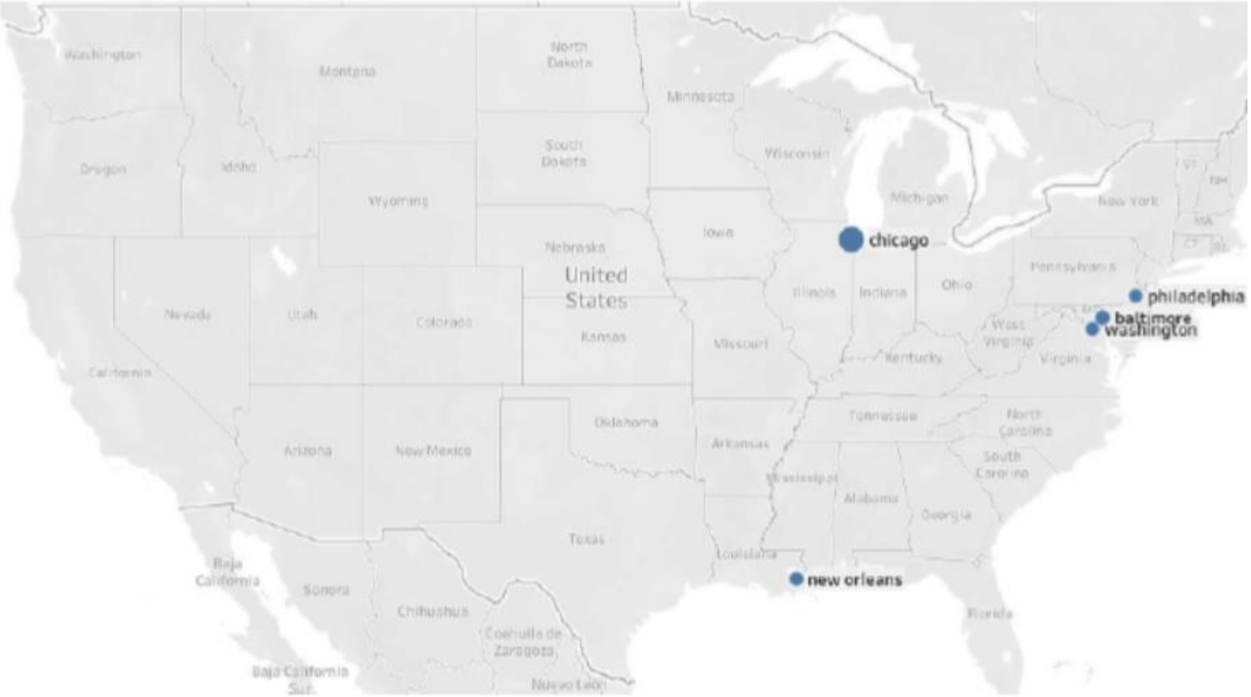


Top 5 states with most gun related incidents

1. Illinois
2. California
3. Texas
4. Florida

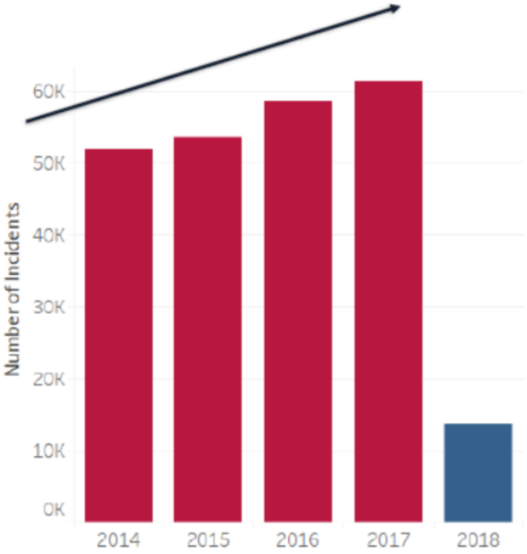

Top 5 States with Gun Violence Incidents

Although, number of incidents are more in Illinois but there are actually less number of killings as opposed to other states. Maybe the less harming gun was used? We shall see

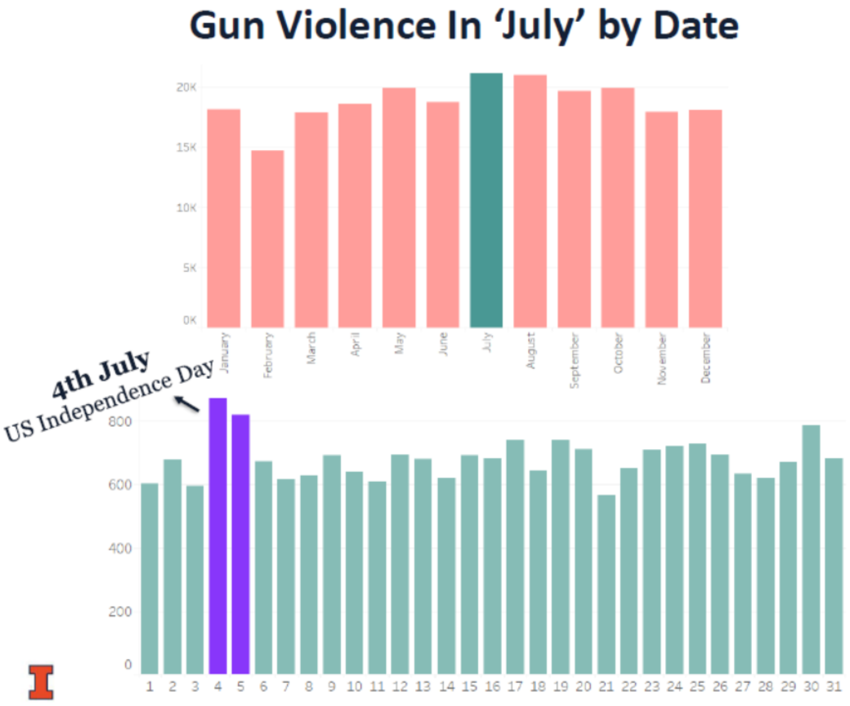


Top 5 cities with highest number of gun violence includes, Chicago, Baltimore, Philadelphia, Washington and new Orleans

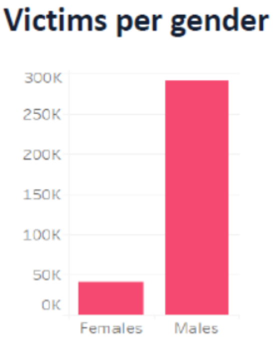
Around **20%** increase in a span of 4 years

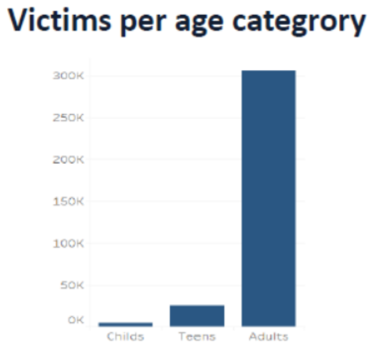


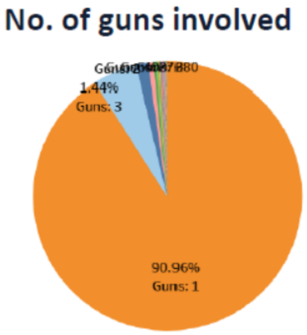
From the above figure we can see that there has been around 20% increase in such type of incidents, because we only had data till march for 2018 that bar for 2018 is small.

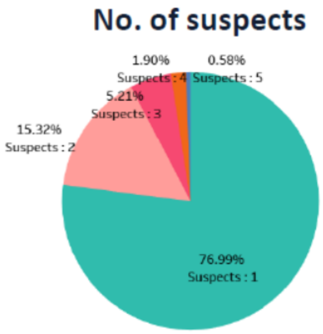


These incidents mostly happen on 4th of July. Is there a reason? Is it because US Independence Day? What is it got to do with such type of incidents?



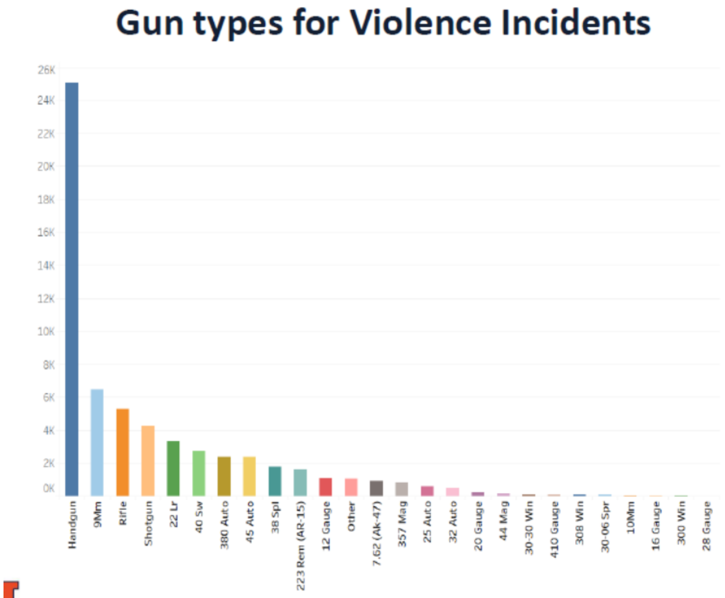






Victims of such type of incidents are mostly men and adults, around 24% of the time there were more that 2 suspects and around 10 percent of the time more than 1 guns have been used.

Lets us see what is the most common type of gun that has been used in such incidents.



Conclusion and Future Work

This project helped us apply the learnings of this course on a real data analysis project. When we read the phrase that “data cleaning takes 90% of the time in a real word project” first time on the internet we thought that it is an exaggeration but now that we have experienced it first-hand, we changed our mind.

The way we were asked to provide the information about our data cleaning task in this report will help clients understand the data cleaning process in our opinion. Including the workflow chart, table to show the changes on each column and explanations on why we changed something can be helpful for the clients to understand our perspective a little better.

We faced many problems while doing this project, for example, my partner and I had little idea about the python language especially regular expressions, and we needed to use regular expression extensively in this project. So, we learned the regular expression and did a lot of hit and trials to get exactly what we want from the columns.

The future work for this project is make a nice website and include some pretty visualization on it explaining the trends we found.