

Crime Rate Analysis

Fall 2019

|  |  |
| --- | --- |
| **Name** | **Class ID** |
| Awais Ahmad Khan | 14 |
| Khalid Dhabbah | 6 |
| Shah Rukh Khan | 15 |

**Introduction**

The purpose of this report is to illustrate what we have implemented for this project. This project aims to utilize Hadoop stack and Apache Spark in order to analyze and visualize information about crimes occurred in Kansas City. This information we have is from open data for Kansas City. The project is divided into three sections. Firstly, we collect and clean the data we have. Secondly, we use big data tools, such as Hive to analyze it. Additionally, we will use SparkContext in Apache Spark in order to perform analysis and analytics. For the analytics part, we will use some algorithms in machine learning. Finally, we will show our implementation in a dashboard.

**Background**

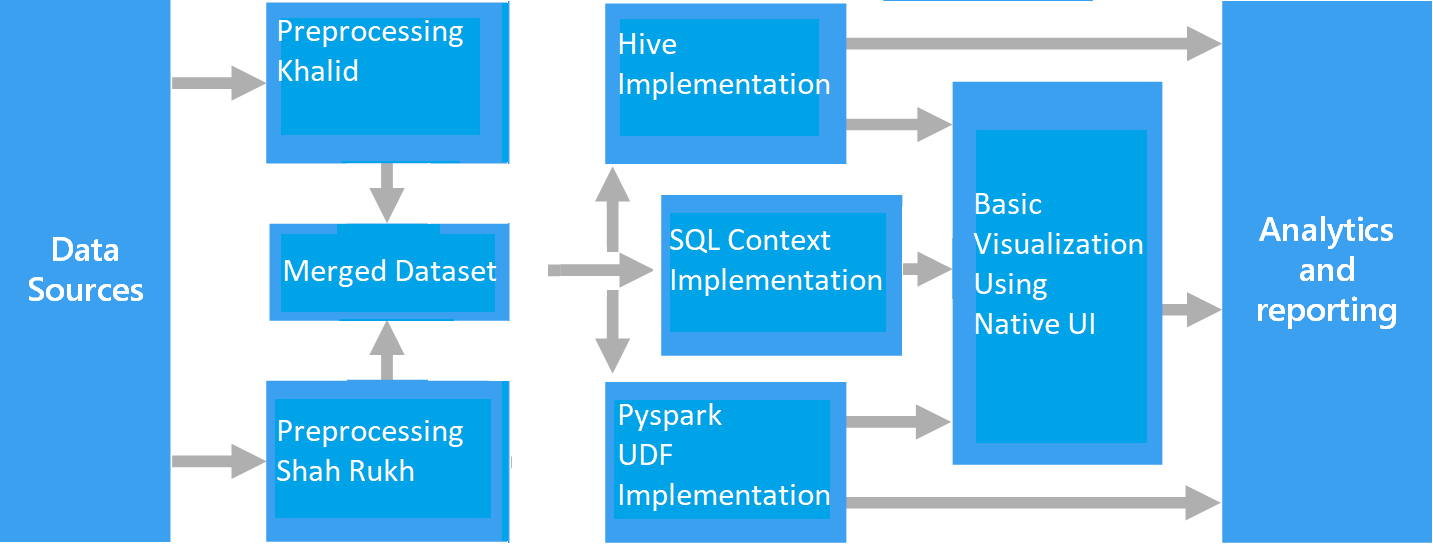
We are rewriting the background information here to avoid plagiarism from our own last submission. We have KC crime data and it is a real big data problem since we have to deal with over 10k records of crime reports. We are writing python scripts and adapting them to Hadoop, Pyspark implementation for a speeded-up implementation. Our scripts will be able to produce useful information and analysis on run time using big data programming.

The big data approach that we have used here is carefully selected since we first performed all these operations using simple python and dataframes for basic analysis but the impossibly huge size of the data made it nearly impossible for the normal scripts to give us real-time results. Thus we adapted it to big data infrastructure so that we can achieve our task of real-time analysis and implement services we learned in the class.

**Related Work**

We have not been able to get any help with this project because we could not find anything that could help us with this project. Since the work we are doing is one of a kind thus there is no related work available for this specific dataset and type of analysis. The work we are doing is unique in the sense that when we go online to seek any related work or help, we literally cannot find anything on this and no prewritten script can be found. All the scripts we submit with this work are 100% work of our own and can be sifted through plagiarism software.

**Architecture Diagram** and **Workflow Diagram**

****

**Machine learning**

Machine Learning is where we can use algorithms to let the system learn from the dataset we have. We have used python to perform these algorithms. Then we used Pyspark to do the same algorithm, which we aim to illustrate in this report. As a result, we found that performing machine learning algorithms in pySpark takes less time during training. We used different kinds of algorithms like Random Forest to train our model. Therefore, we had good accuracy. Training the model had us to follow some steps. First, we did feature engineering to let the algorithms work. Then we divided the crime dataset into two portions, one is training and the other is testing. Third, we trained our model by using the training part. Consequently, we used the testing part to test the trained model.

**Dataset**

The dataset includes yearly bases data for Crime reporting. We used this dataset for both the analysis and analytics.

* We have various features such as:
* Reported year
* Type of crime
* Involvement
* Race
* FireArms used
* Gender etc.

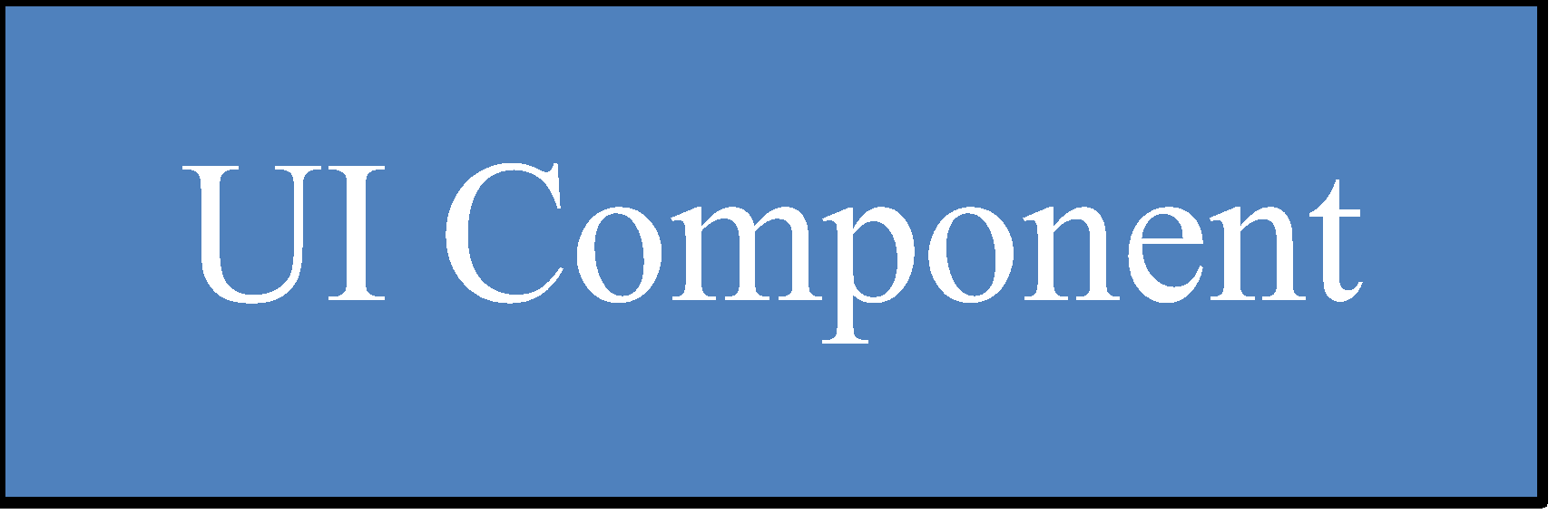
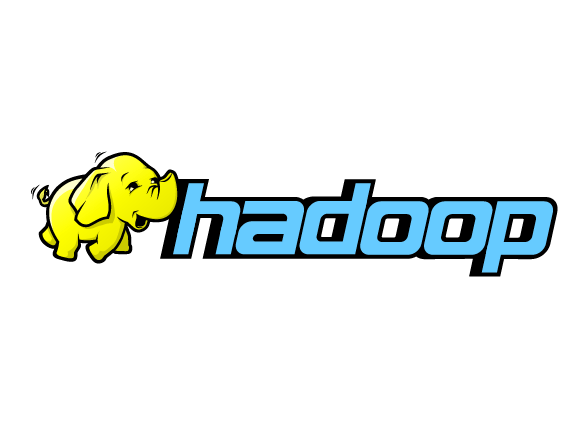
**Detailed Description**

We have used open data to obtain our datasets. We first had a dataset for the year 2018, and then we decided to have it bigger, so we collected datasets from 2010 until 2018. Some details concerning the crimes were illustrated. For instance, we could be able to know the type of crimes and the places where it took place. Not only that but also gender and race and et cetera. First, we had the dataset with 128938 offenses because we only used data from 2018, but then we had it more than one million, which is 1121574. We used Excel pivoting to clean the dataset at first. Then we used Python instead.

**Detail Design of Features**

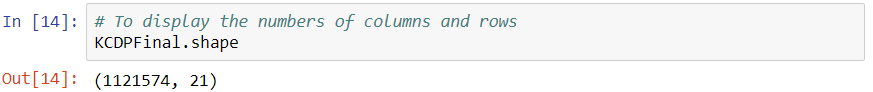
We had a goal that we implement our knowledge in analytics and analysis.

* **For the dataset**
  + We used Python to load the datasets and merge them then we do feature engineering due to the variety of libraries that Python has.
  + We implemented some of our knowledge about Hadoop tools and Spark.
* **For the visualization**
  + We used matplotlib to visualize the information.
  + We also developed a simple dashboard using Angular and flask to illustrate the visualization.



**Analysis of data**

This dataset has more than a million records as shown below.



We have used different ways to feature engineering the dataset. **Firstly**, removing the features. There were some minor features that were not important and had more than 50% of the data missing; we removed them by getting rid of the columns. **Secondly**, there were some features had obvious values, such as City. Because the crimes occurred in Kansas City, we completed the missing values of the feature. **Third**, using a unique category. For example, there are only three types of values in gender, which are male, female and unknown. Therefore, we filled the null values with the unknown category. **Fourthly**, We used the mean function to other missing values like the age. **Finally**, we removed the rest because they were few.

**Hive**

Using Hive, a tool used in Hadoop stack, give us more insight into our data. Some of the implementation we have done, it shows below:

1. Find the number of crime types that happened in Kansas City.
2. Find how many crimes were in 2016 and 2018 based on males’ and females’ involvement.
3. Find how many people were arrested in the Plaza Area.
4. Find how many victims were in 2016.
5. Find how many suspects based on their race.

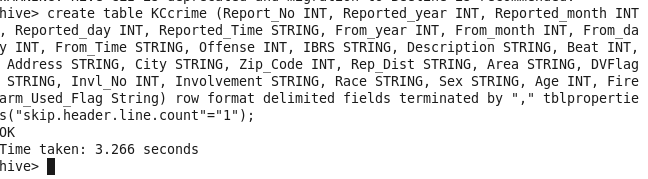
**Analysis Implementation**

* **Hive implementation**

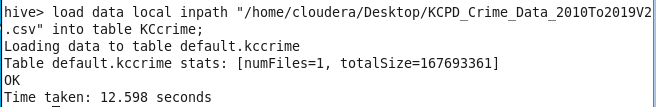
We used two portions here. First, we created a table and loaded the dataset to it. Second, we implemented the quires.

**Part 1:**

We created the table for the crime dataset using Hive.

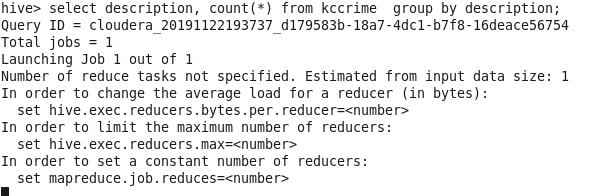


Then, we loaded the dataset.

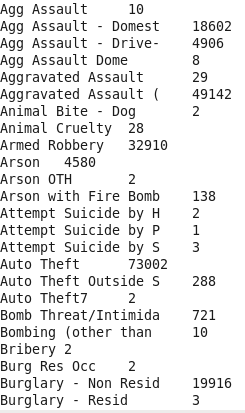


**Part 2:**

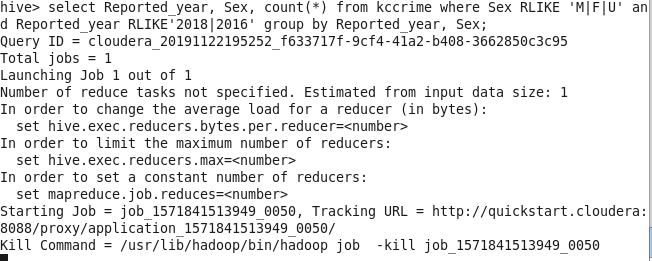
**First**, illustrating the number of crimes based on their types.



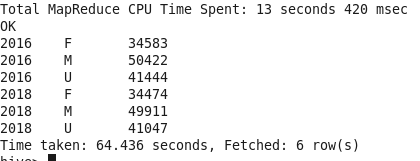
**Result**



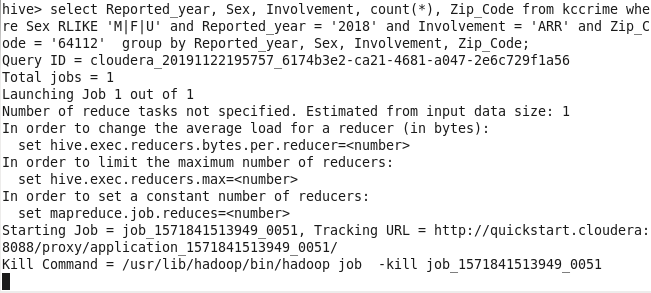
**Second**, illustrating how many people were involved based on their gender in 2016 and 2018.



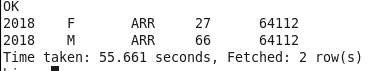
**Result**

****

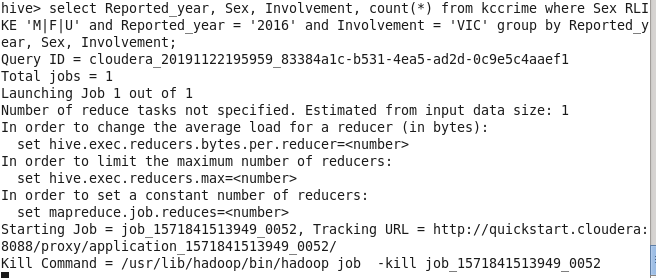
**Third**, illustrating the number of people who were arrested in the Plaza Area.



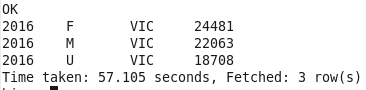
**Result**



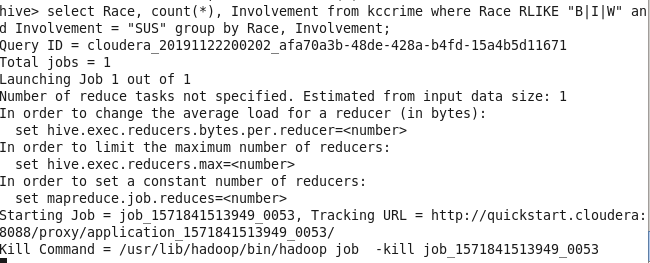
**Fourth**, illustrating the number of victims in 2016.



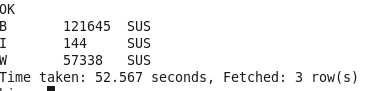
**Result**



**Fifth**, illustrating the number of crimes that people were suspected based on their race.



**Result**



* **Pyspark Implementation**

We have implemented following in Python functions and then called those functions using Pyspark User Defined Functions or UDF as you may.

We performed Analysis on the Data using the robust and speeded up operations via Pyspark by writing a function in plain python and then calling it via UDF.

• We can get crimes in a certain Radius

• We can get Crimes in a certain time frame

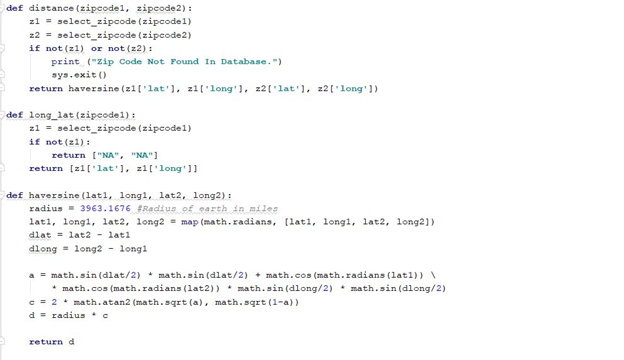
• We also combined the above two together.

• We get number of a specific type of crimes in a certain radius

• We get all different type of crimes that happened in general.

• We have been able to implement a safety index check

Code for Radius Calculation!



Functions are written in python but later called using Pyspark UDF.  
UDF maps any kind of function or class to Pyspark implementation without having to modify the code from scratch.

When we run this code, we are prompted to enter the filters. Such as Radius, ZipCode etc:



We decided to stick to zipcode and not long lat coordinates because they are consistent whereas any other form of location data is subject to redundancy and change.

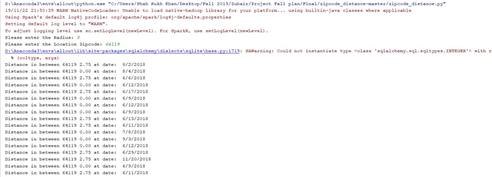
**Result:**



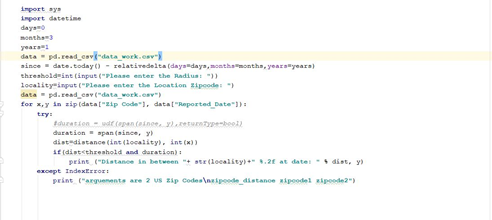
**Radius as 3 Miles and Zipcode 64119**

Now, this is getting me data for the past 3 months, I can change that and get much more data. As shown below!

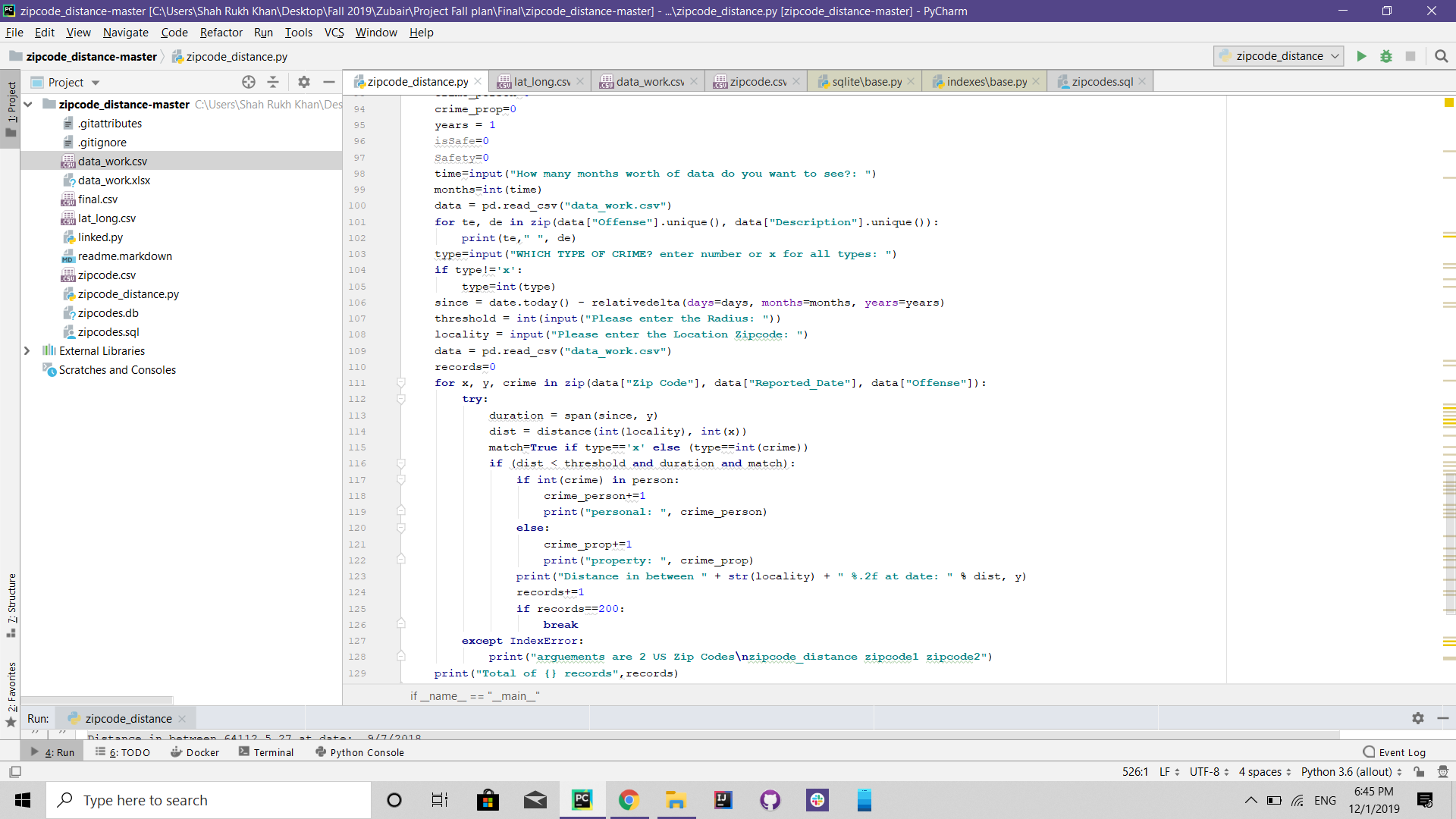
**Result:**

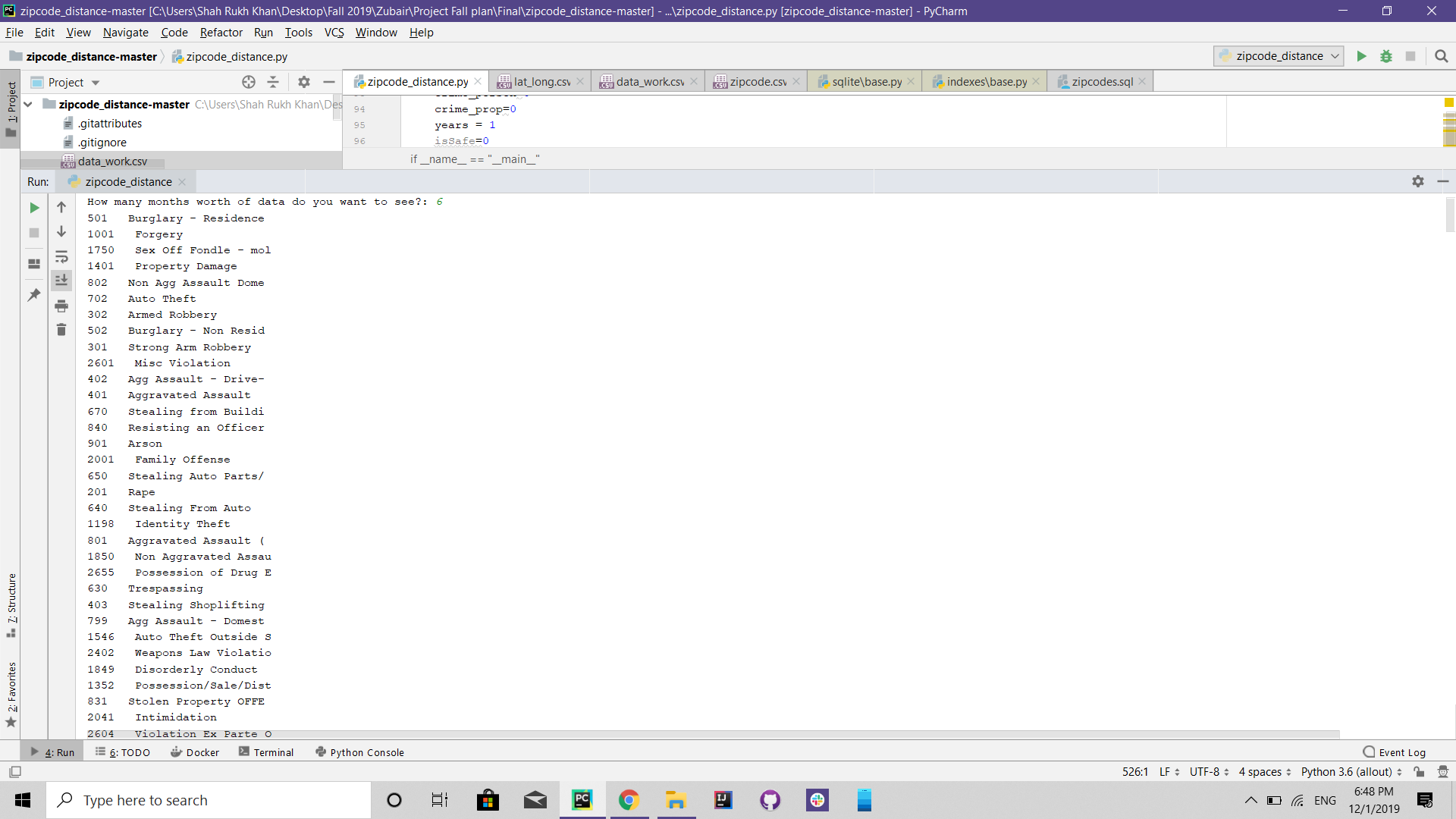


Code for the function that we called using UDF



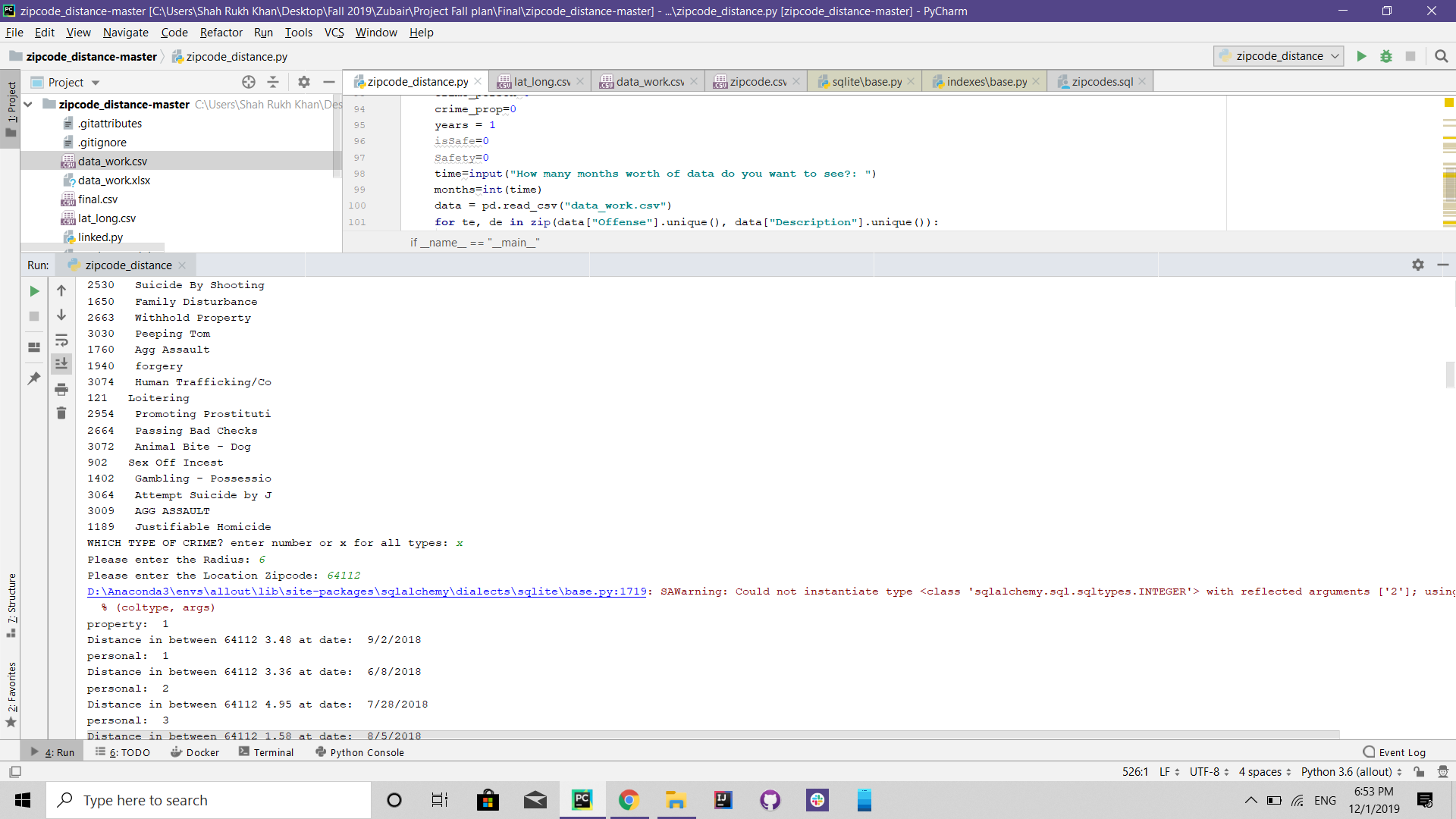
We now implemented the threshold value for safety spike in the code. We integrated it with the types of crimes and based on these we are finally able to get safety level information to customers for a certain location. We can check all different type of crimes and their frequency using below code:

  
Result for this code will be an option to the User where they are free to get statistical results for a certain data or are they want to get it for all the crimes.

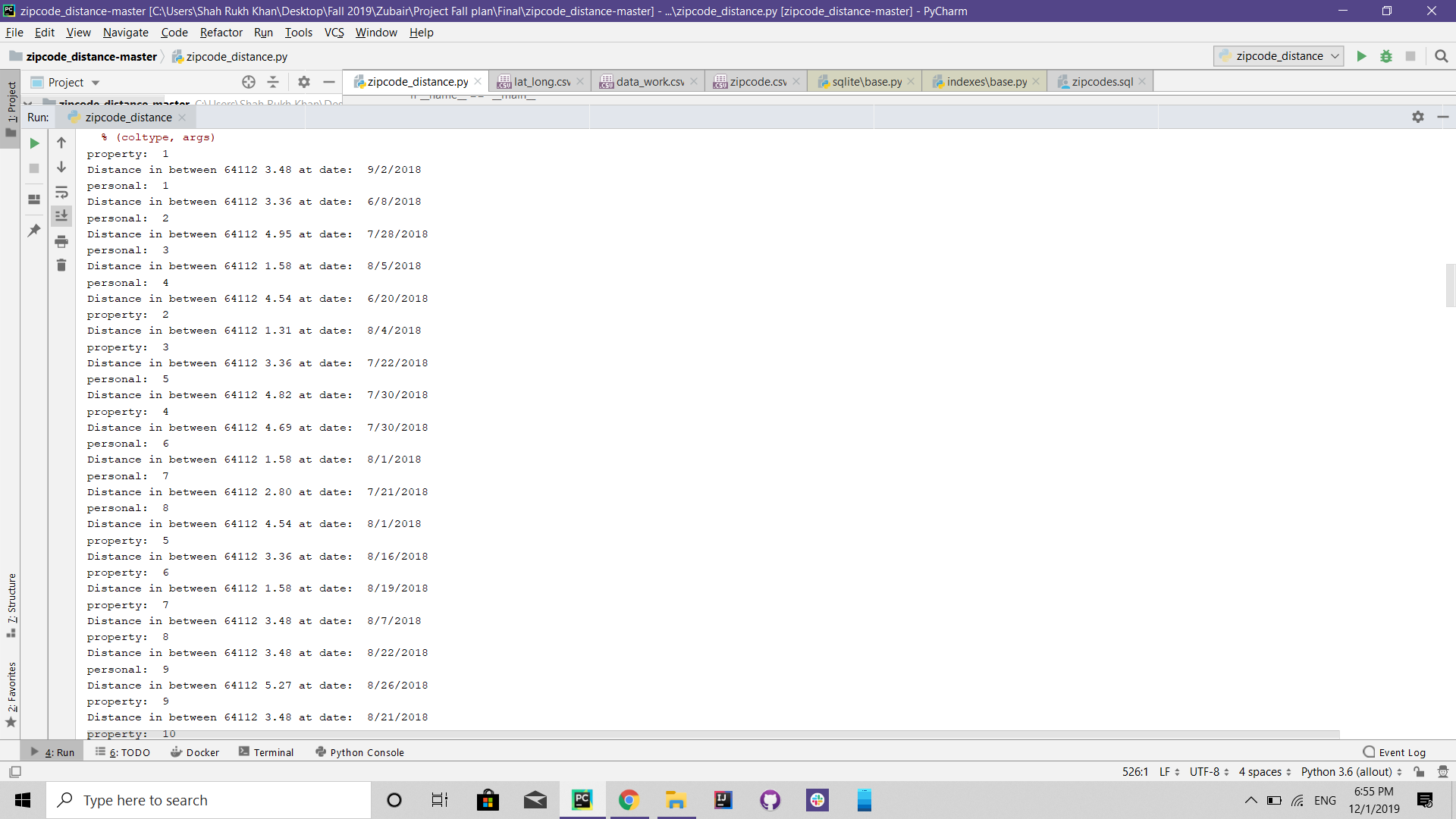


Now our user will select a crime here or enter x as default for all types. Once that is done, the code propagates into the next stage. Where it asks the user for the Radius of the area in which the statistical data needs to be generated.

And giving the ZipCode here like below:



We now have results start pouring in on the console as below:



We can see a few details in all these returning statements:

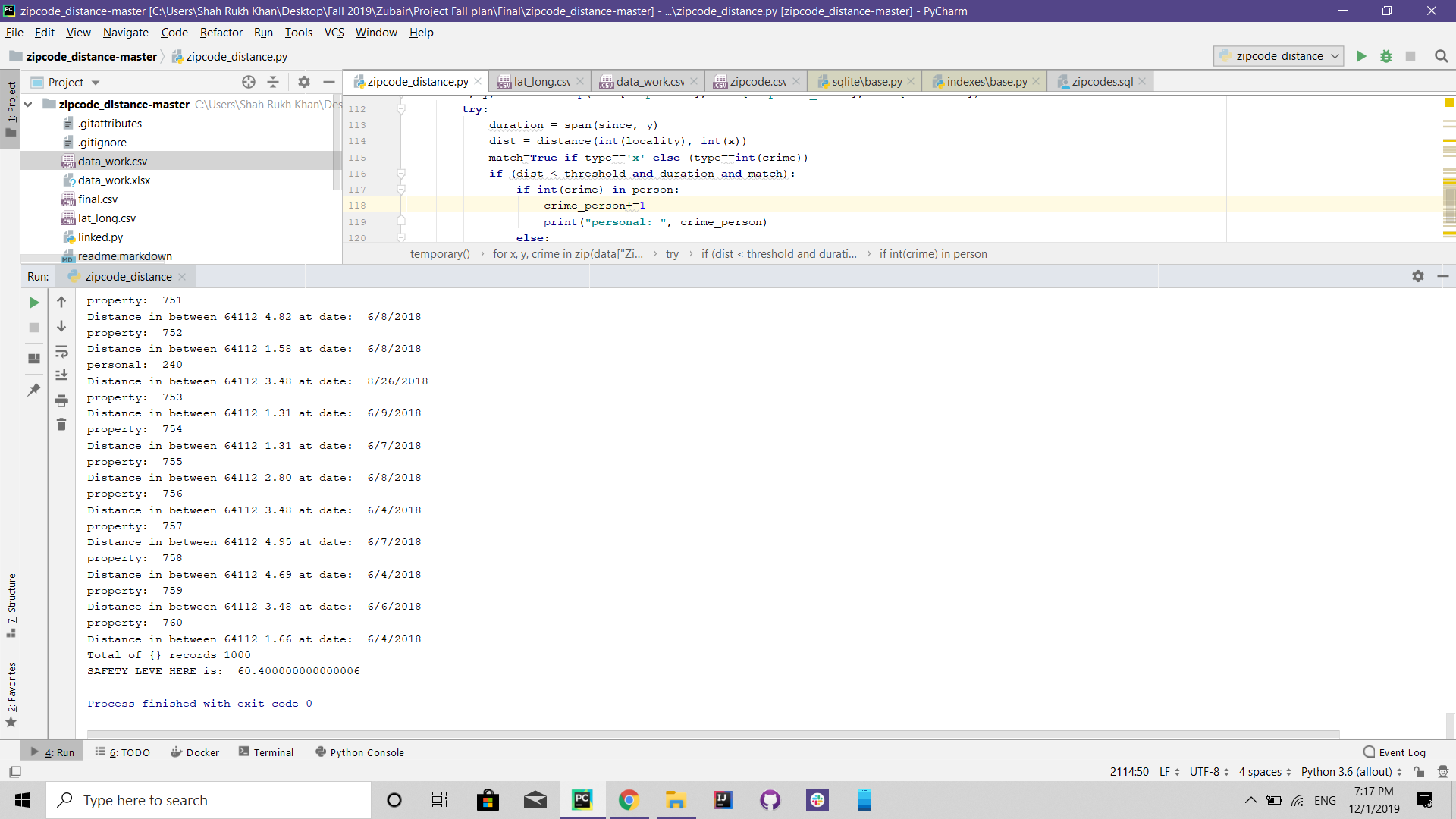
* We can see the distance from the epicenter of crime
* We can see the date on which the crime happened
* We can see the count of personal and property crimes that happened until that day.

We created a Peace Index for the Crime Data.  
The whole Safety level was gauged by careful analysis of human nature and the crime data that we have.  
We are able to confidently say that Humans feel danger or an attack on their safety in the presence of a crime that directly affects humans personally. Such as Assault, Rape and Hit and Run, etc.

Humans have a lasting traumatic and concussion experience when such an incident happens and the effect lasts up to 70% more than that of a non-personal attempt.  
Using this calculation we are able to get a percentage of the total crimes that happened where Humans were involved and where only property damage was involved.  
Using this we are very confidently able to report the safety or danger level in a certain radius of the point of interest. Our Safety index hence is an aggregation of 70% of crimes against humans and 30% of crimes against property or anything where humans were not directly affected by it.

This resulted in a very accurate depiction of safety levels.

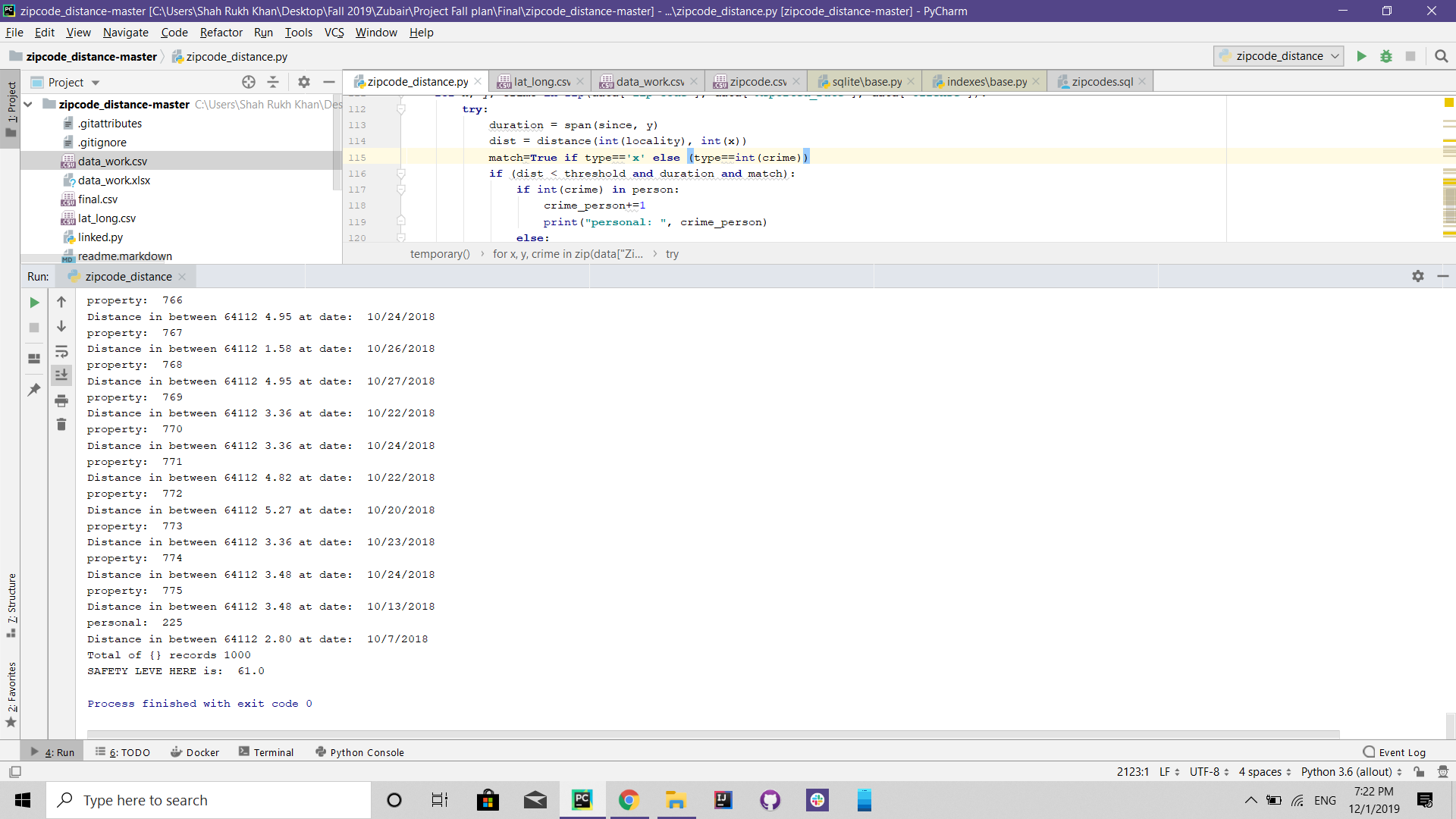
Below is the Safety index report of <=1000 crimes that happened at 64112 in the last 3 months in a radius of 6 miles:



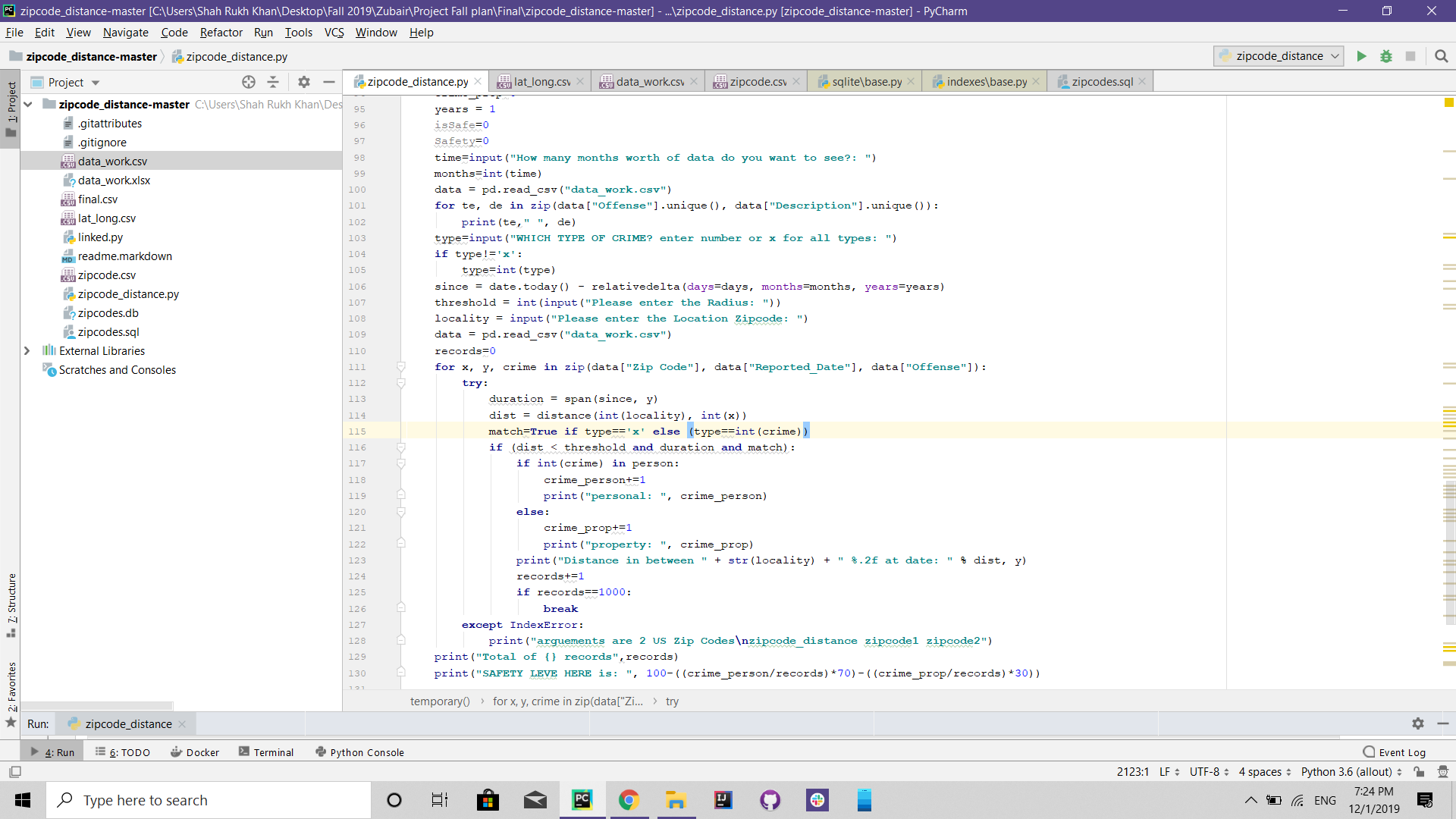
We get a Safety level of 60% for this location by looking at the past 6 months worth of data.

Let us check the same for the past 3 months of data!

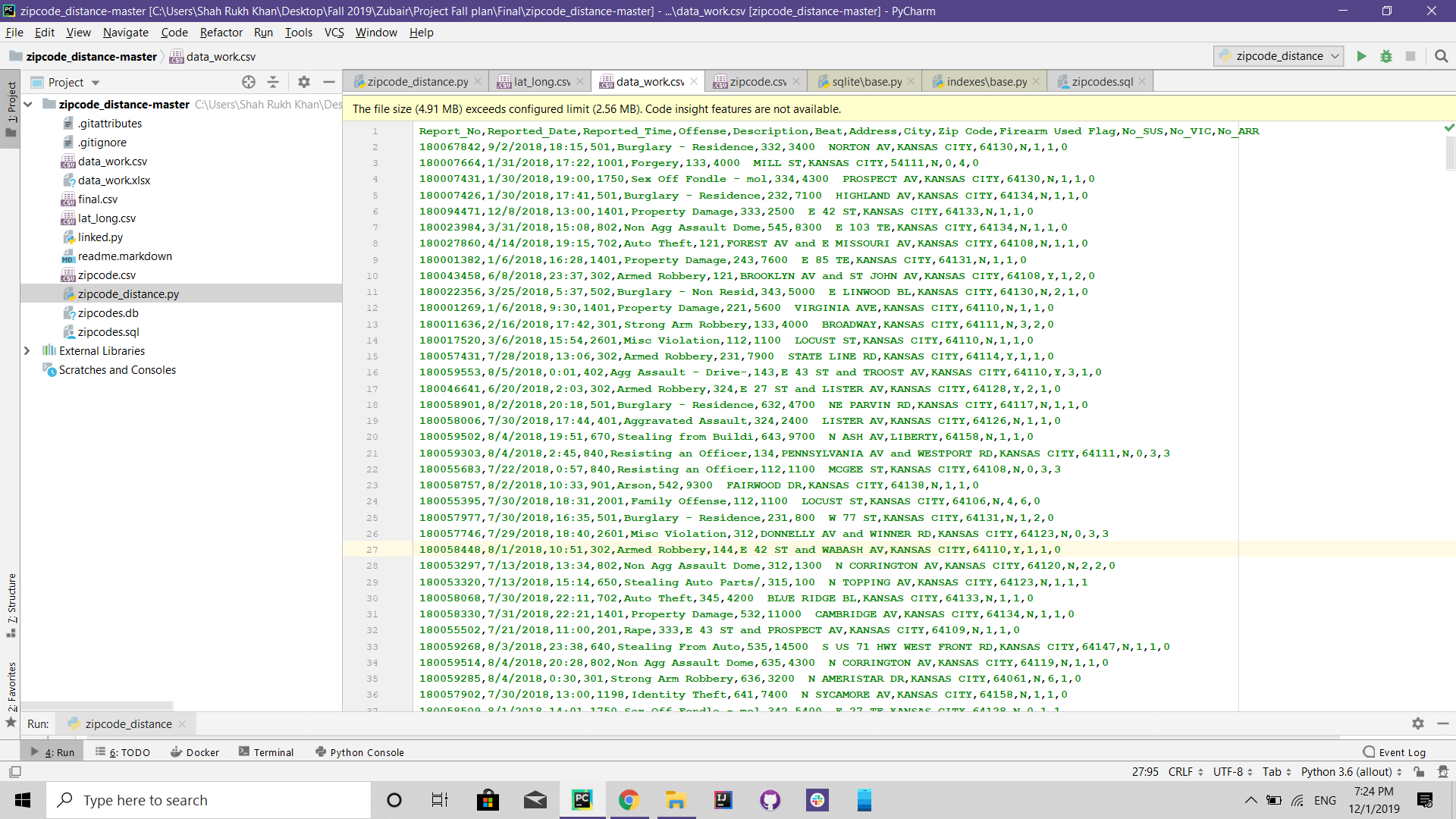
Here we get 61%



The Complete Code for this implementation is below:

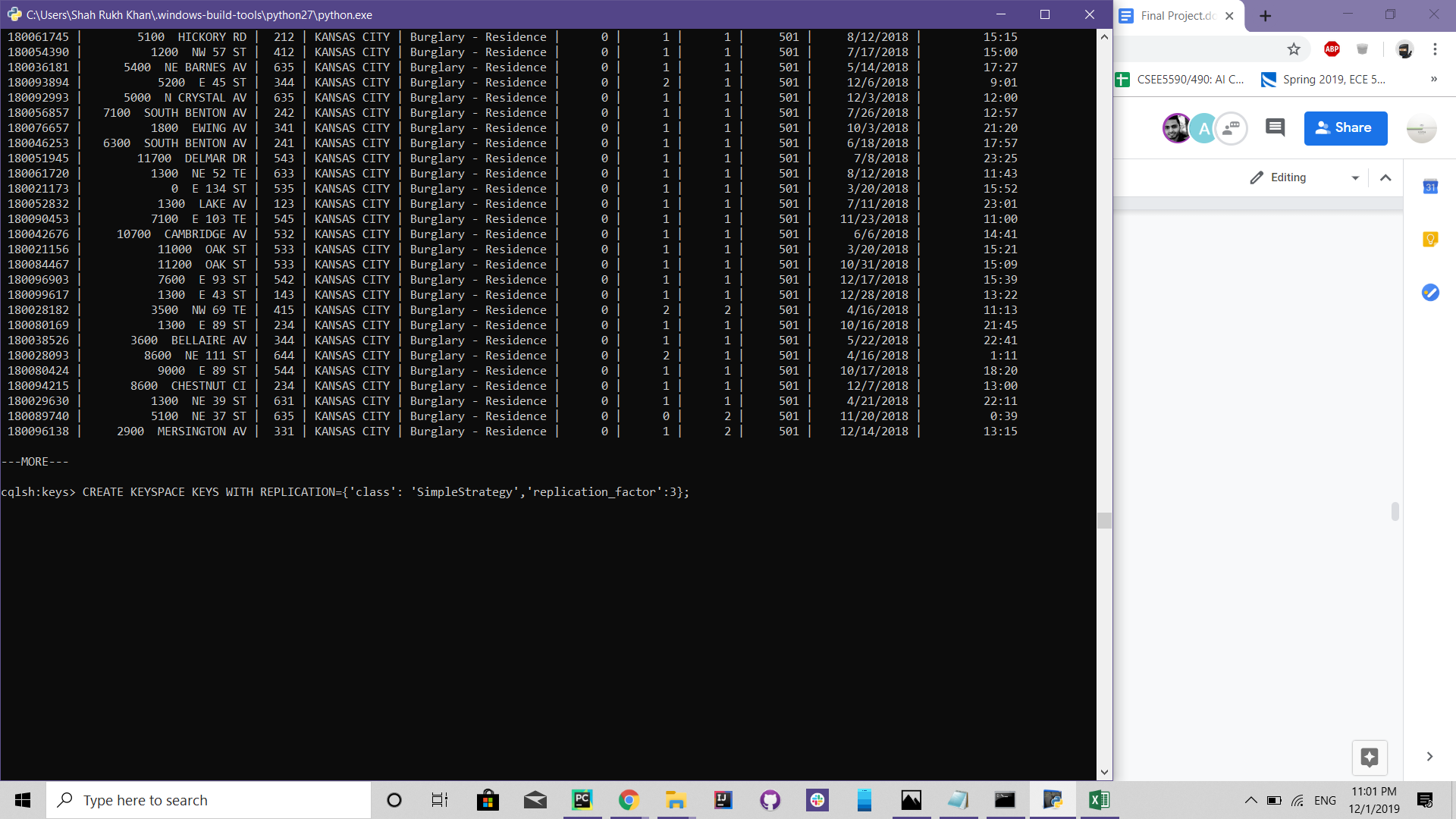


The Dataset on which this was implemented is Below and will be provided in the Github Repository!

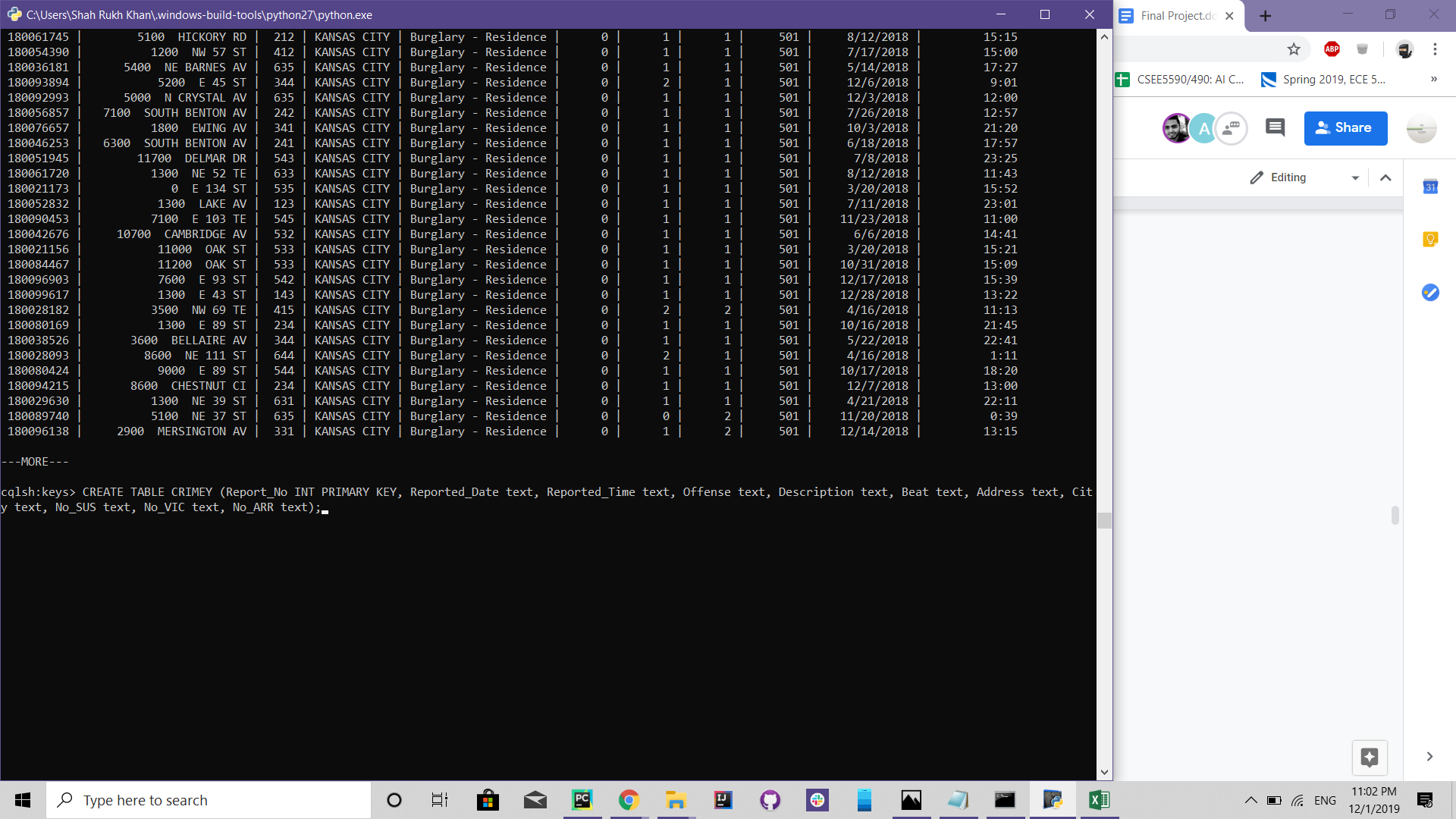


* **Cassandra CQL Analysis**

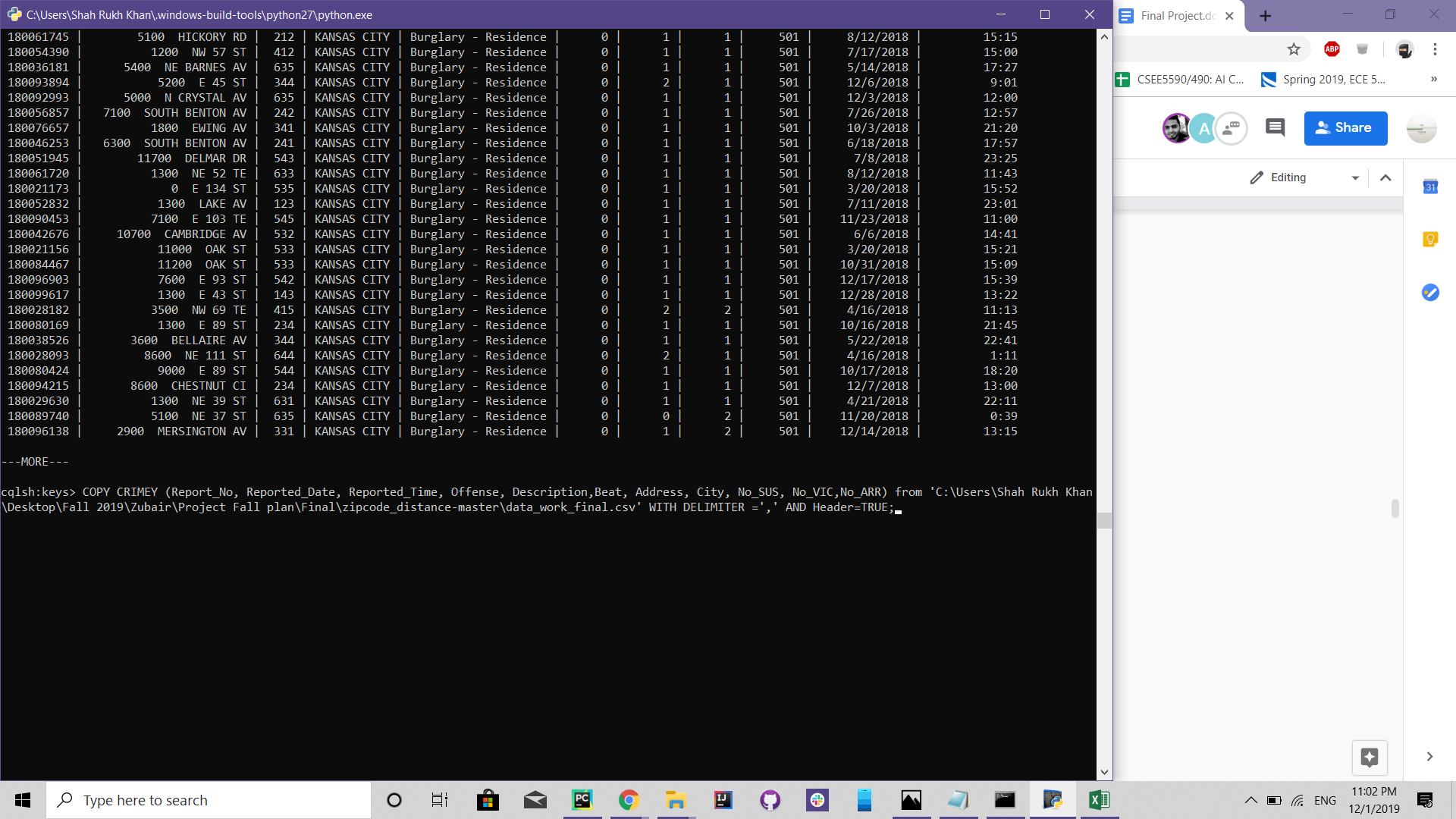
We also performed CQL Queries on our Dataset to get some valuable insight into the data.  
We first created a Workspace in Cassandra:

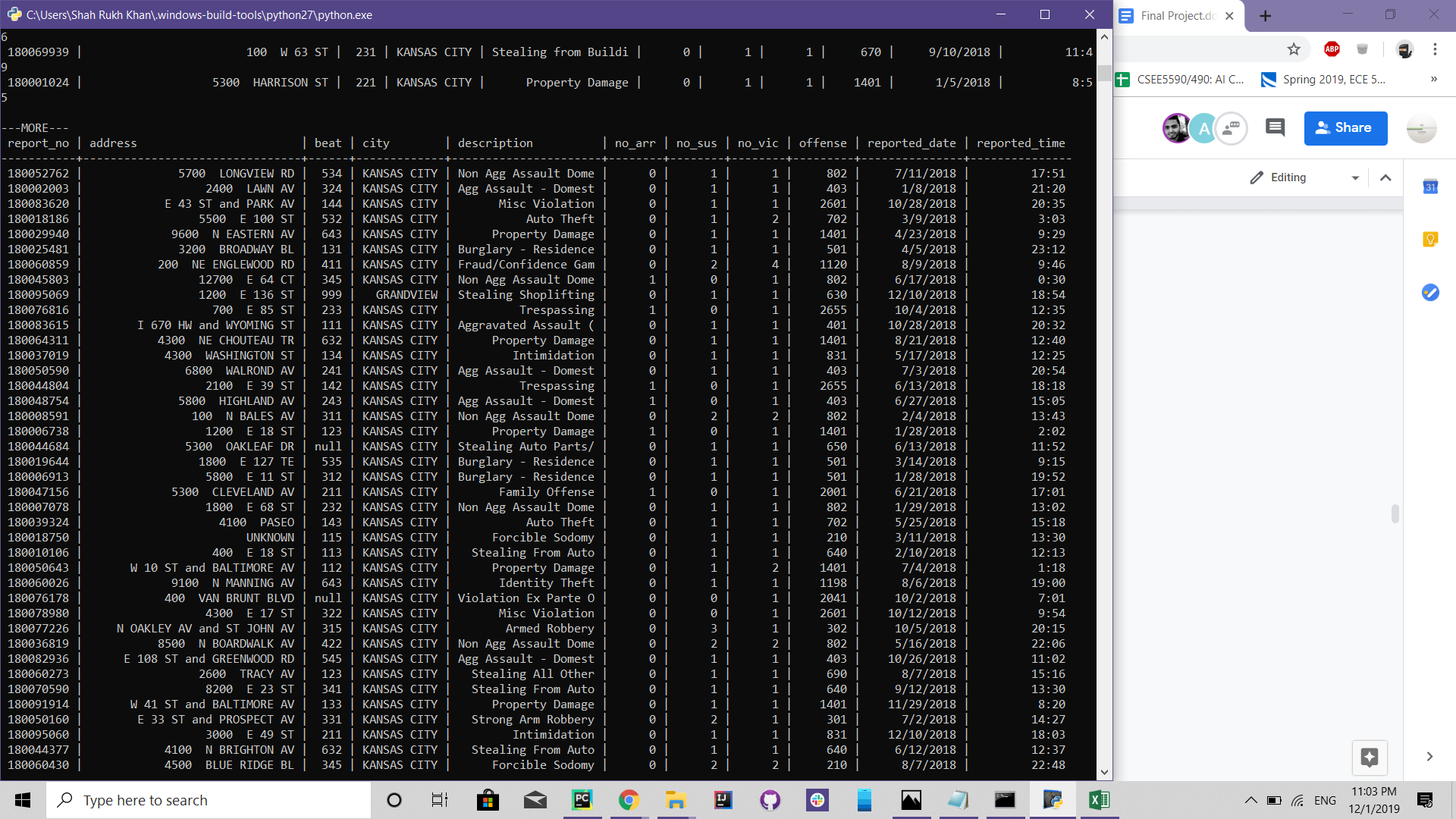


Then we created a Table for the Massive Data:



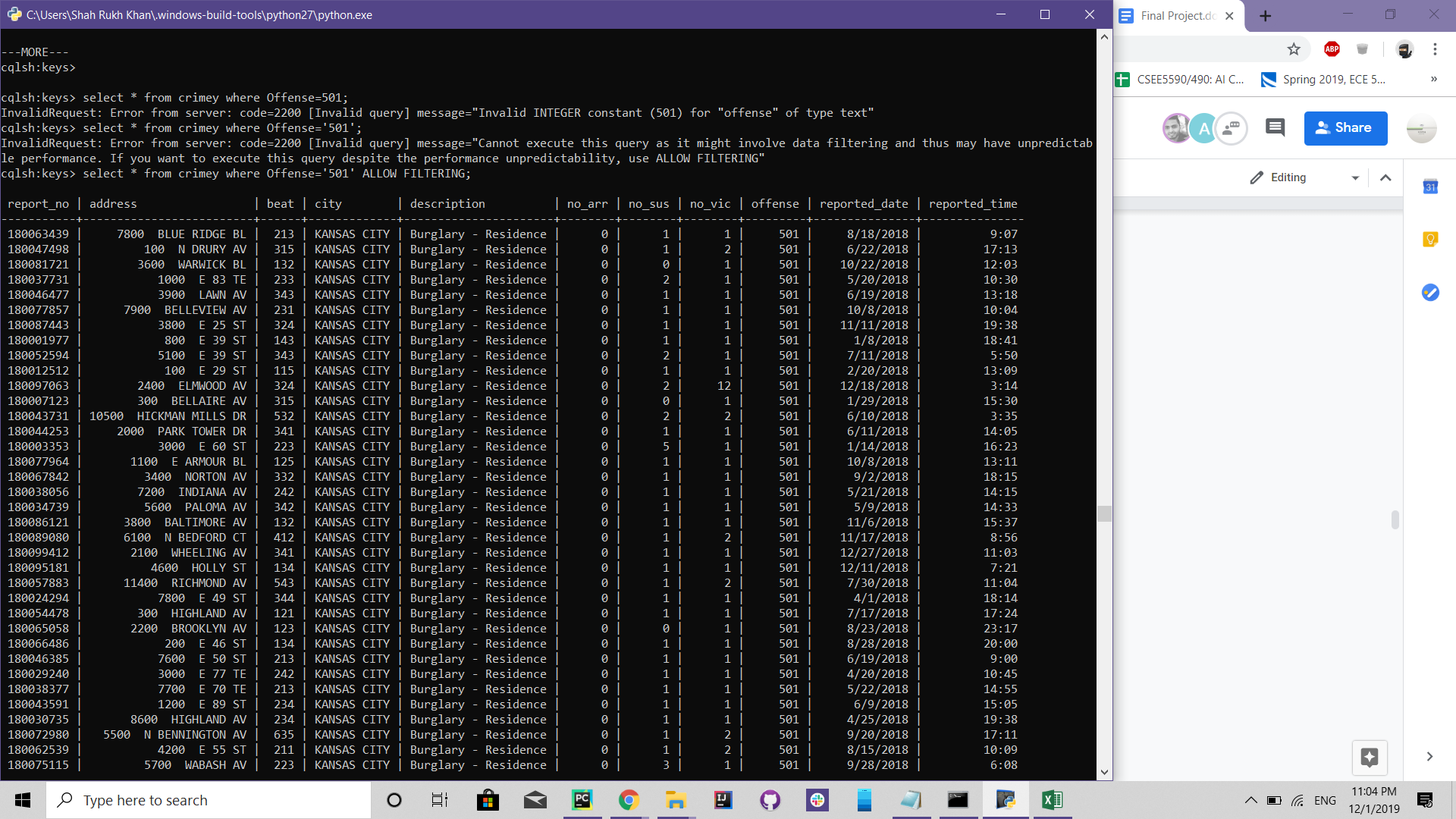
We then copy our data into this table:



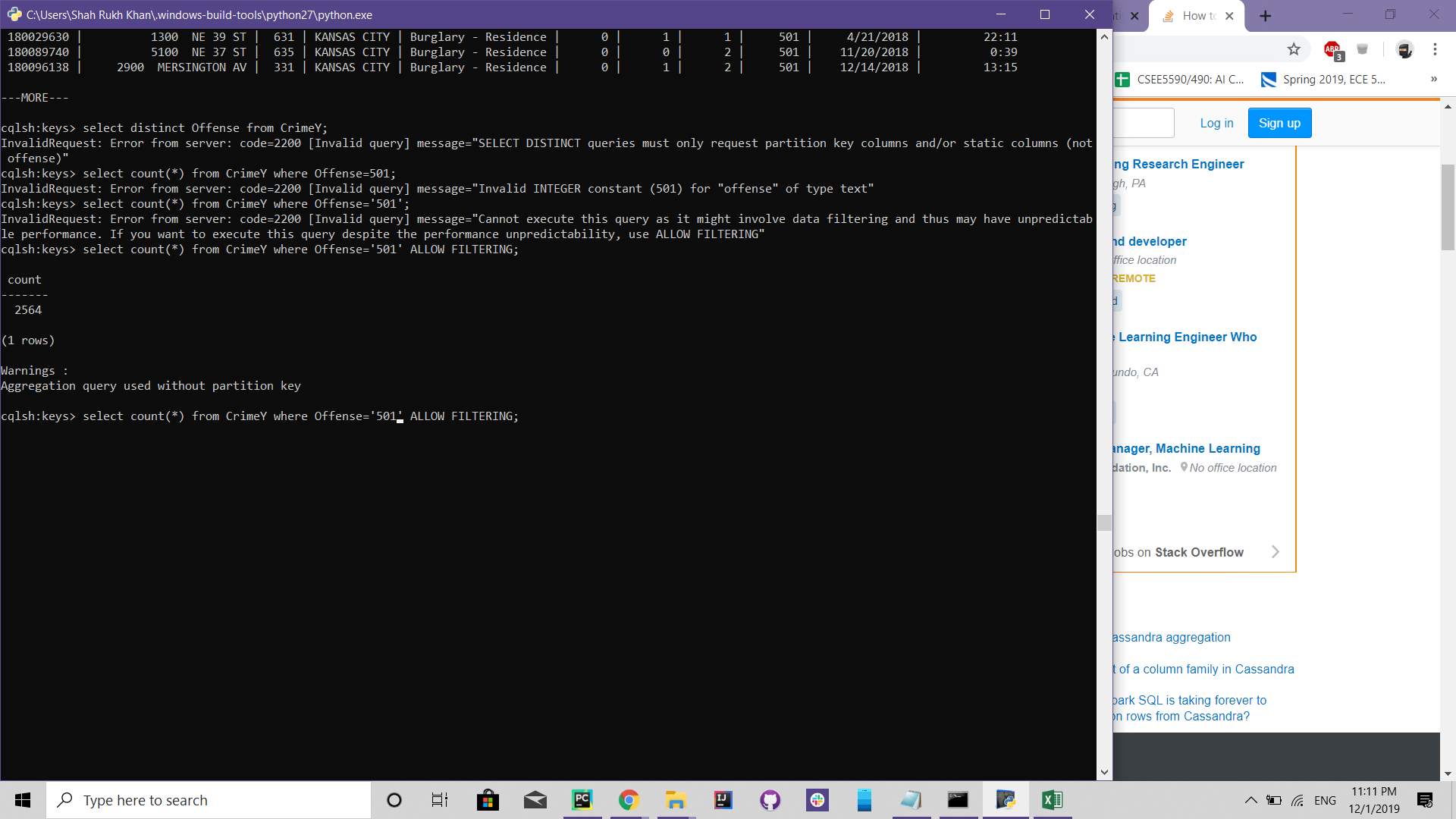
After we are done, we visualize our data to see if we have the table ready:  


Now comes the analysis:

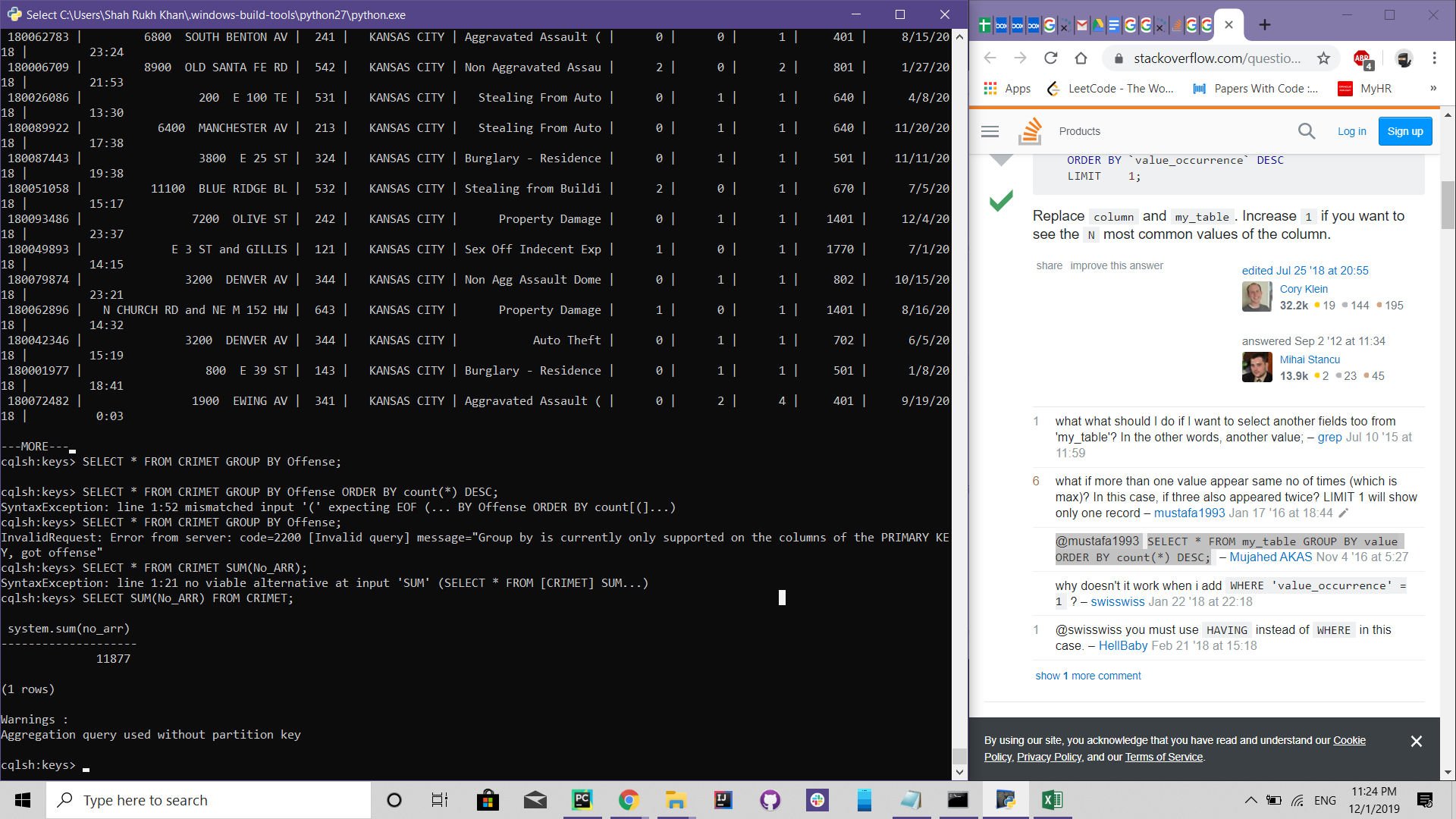
We can get all reportings for a specific type of crime: Say 501



We can count the type of crimes:



We can also see the total number of arrests!



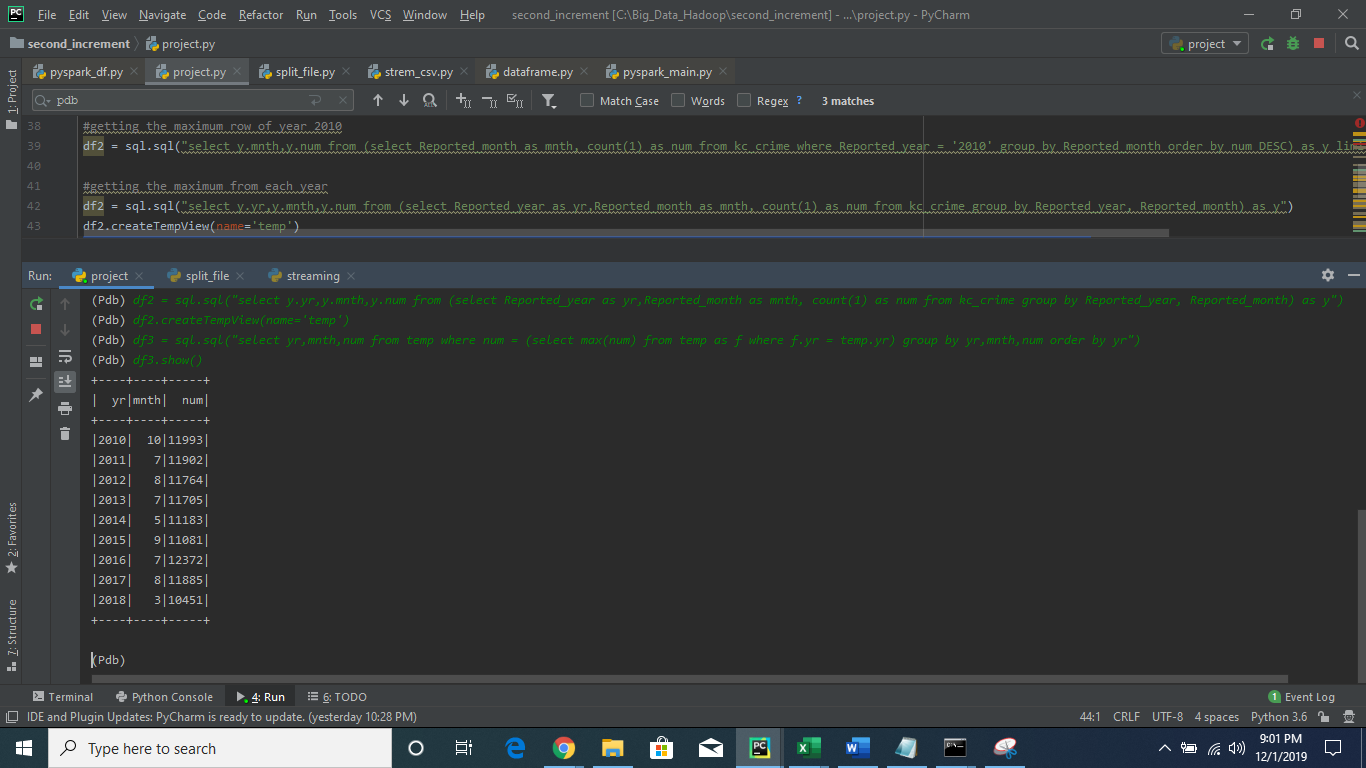
Cassandra is very fast and gives us good and quick enough analysis of the huge data and lives up to its good name.

* **Pyspark SQLContext**

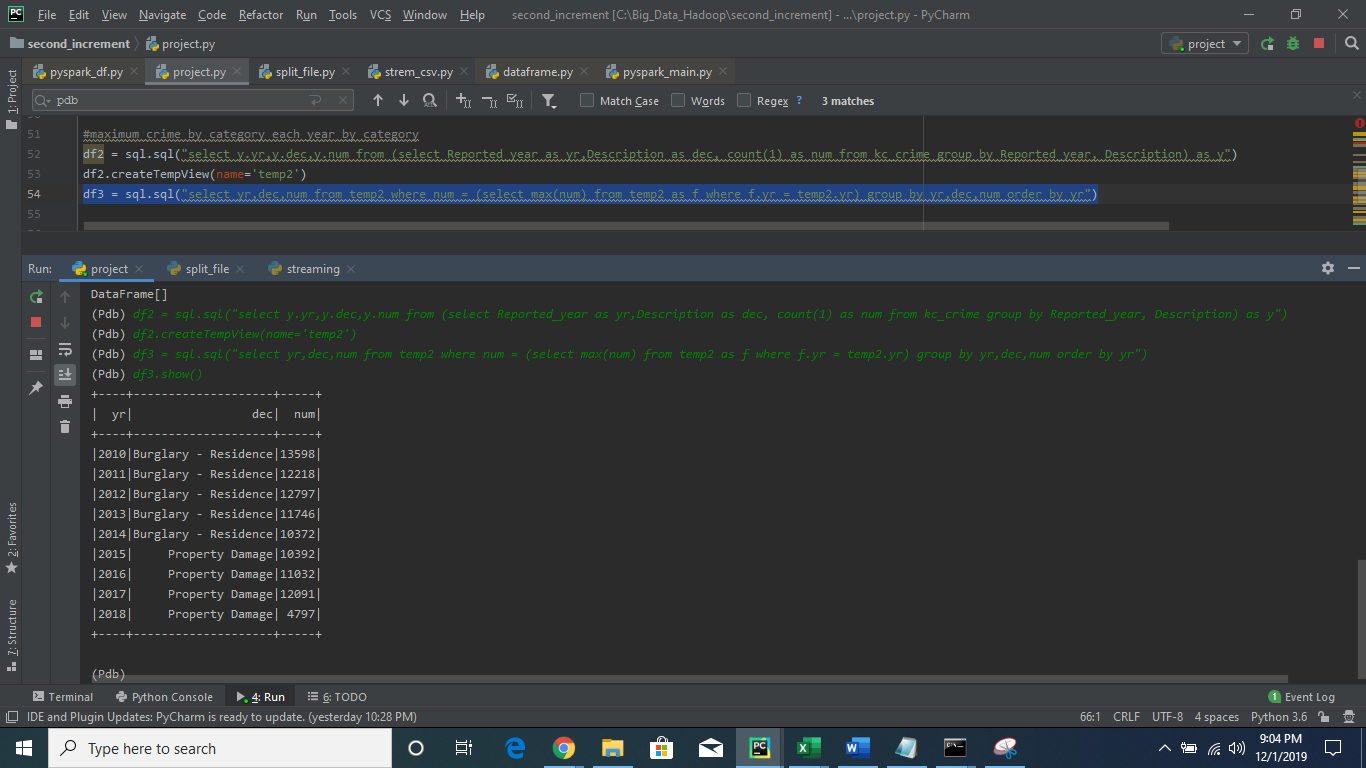
The time analysis (Year, Month, Week) is done through Pyspark SQL. To elaborate on the time analysis we can focus on the following points.

1. How many crimes have occurred every year.
2. Is the number of crimes increased or decreased in the last decade.
3. In which month the number of crimes is more per year.
4. What is the frequency of crimes at specific days of the week.
5. Are the crimes more during the weekend or during the weekdays.

To answer the above questions Pyspark SQL is very helpful. These questions sound very important but on the other hand the answer to these questions can easily be achieved by applying simple queries.

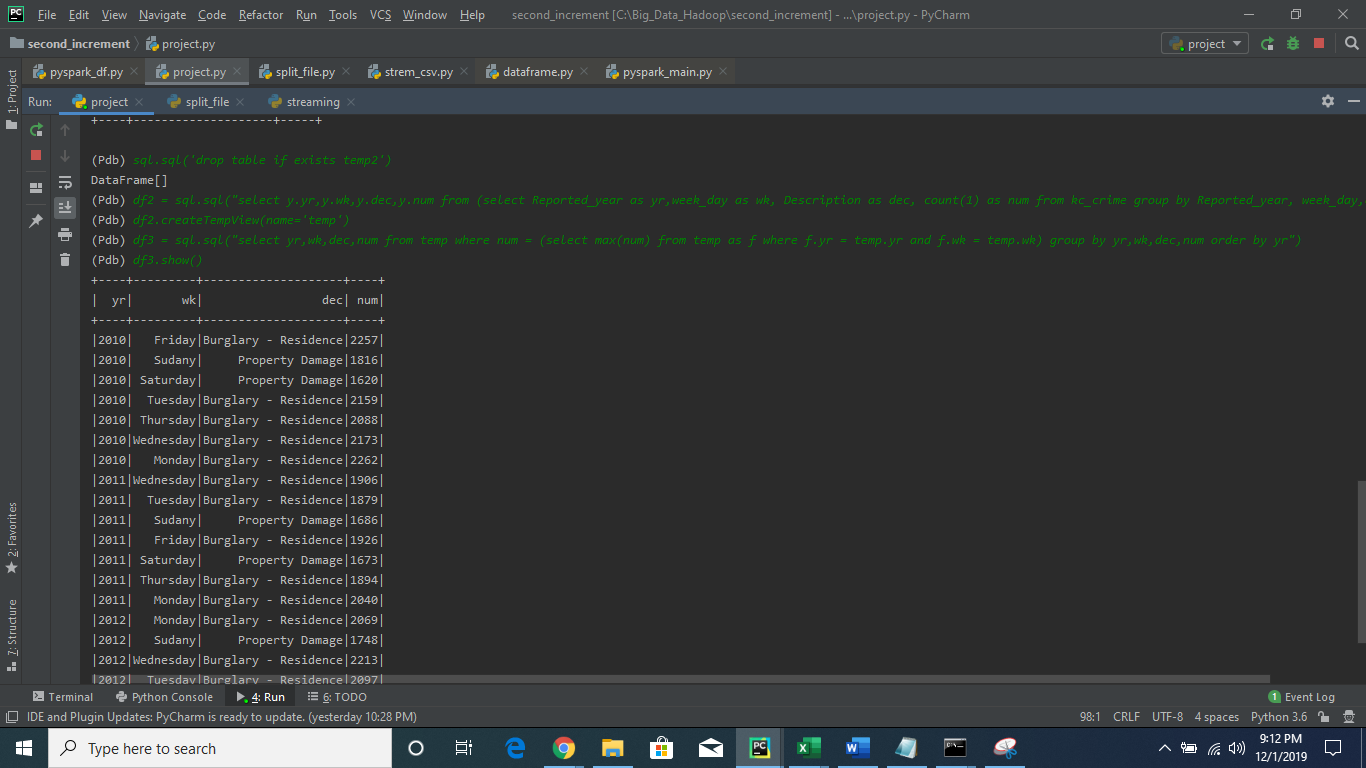


The above screenshot shows the months in which the maximum number of crimes have occurred in each year.



The above screenshot shows the occurrence of a maximum category of crime.

In order to answer the day in which the crime is maximum. We have applied the following query which answers our question.



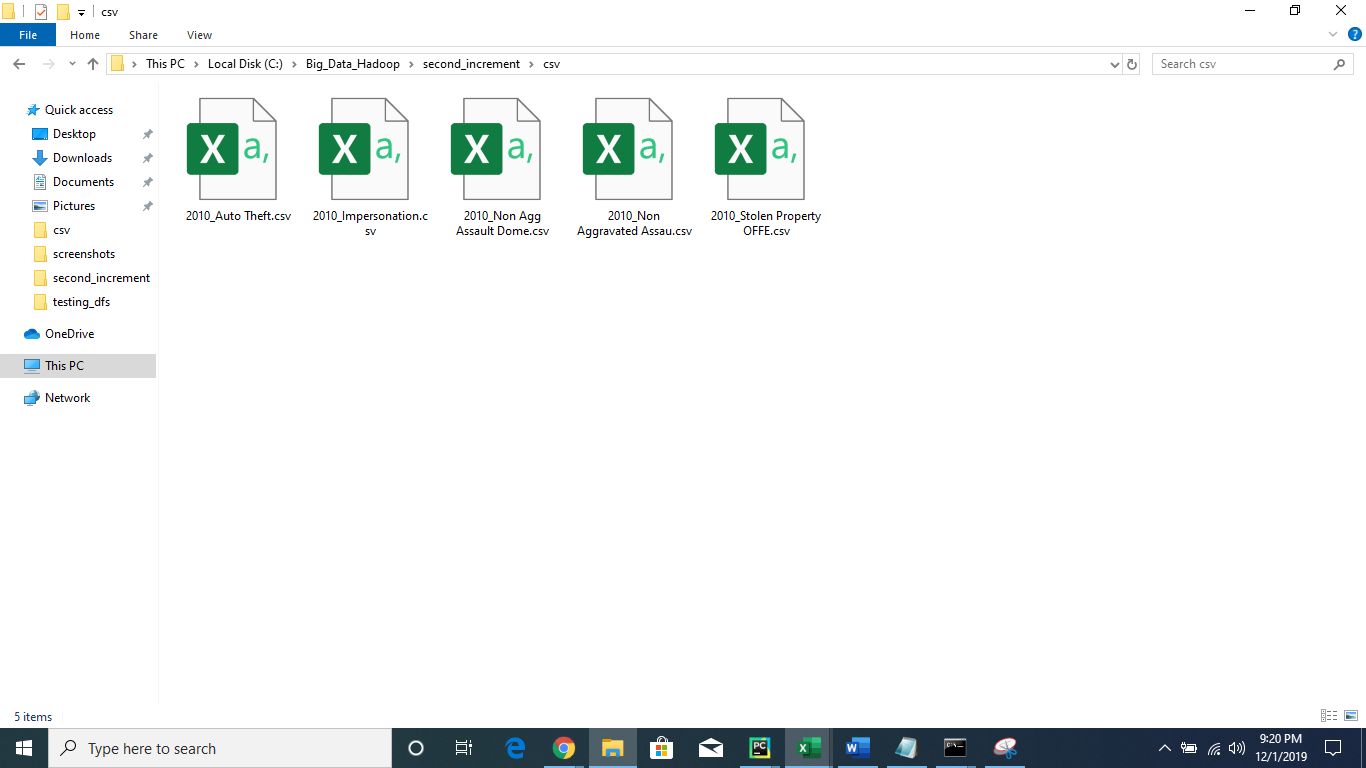
The above query shows us the number of crimes that have occurred at every day of the week per year.

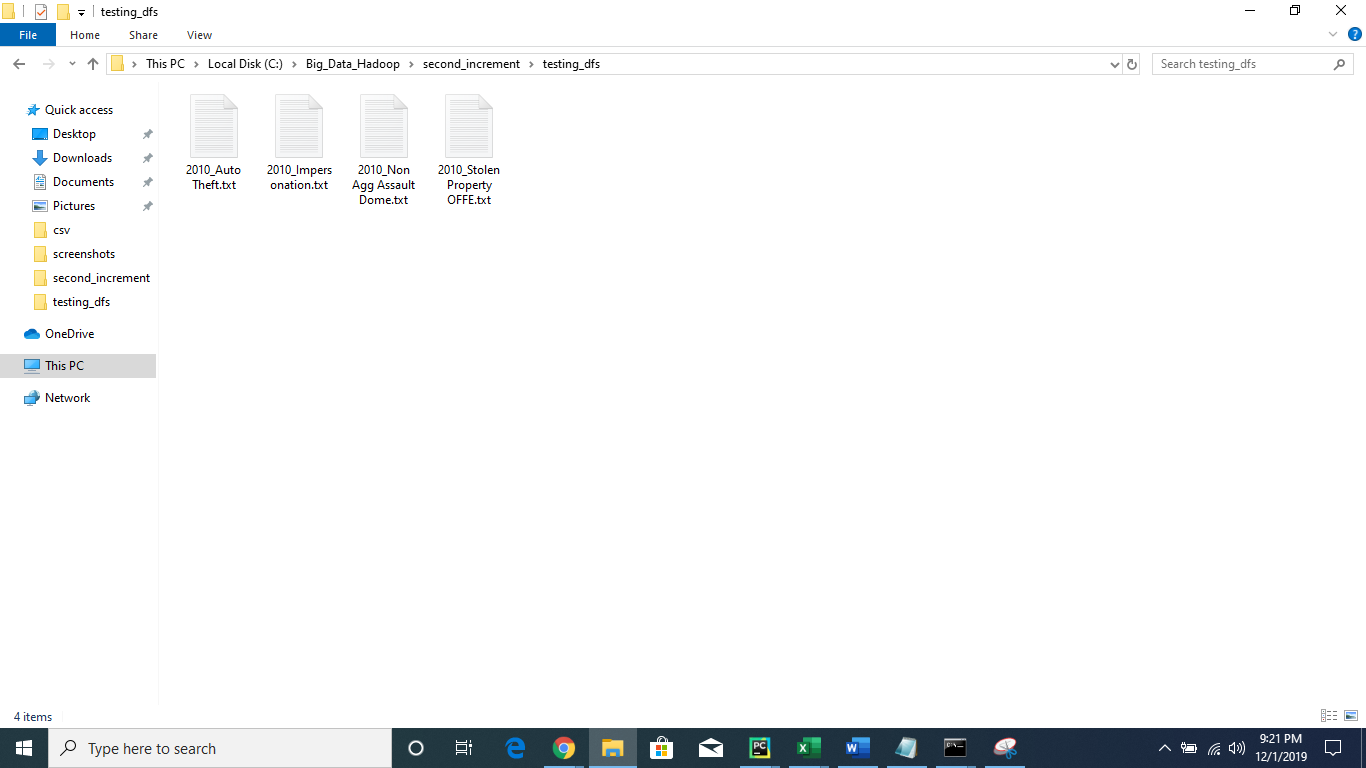
The above analysis can help us to answer any questions about the crime data at which year or month or even which day of the week the crime is more or which crime is more.

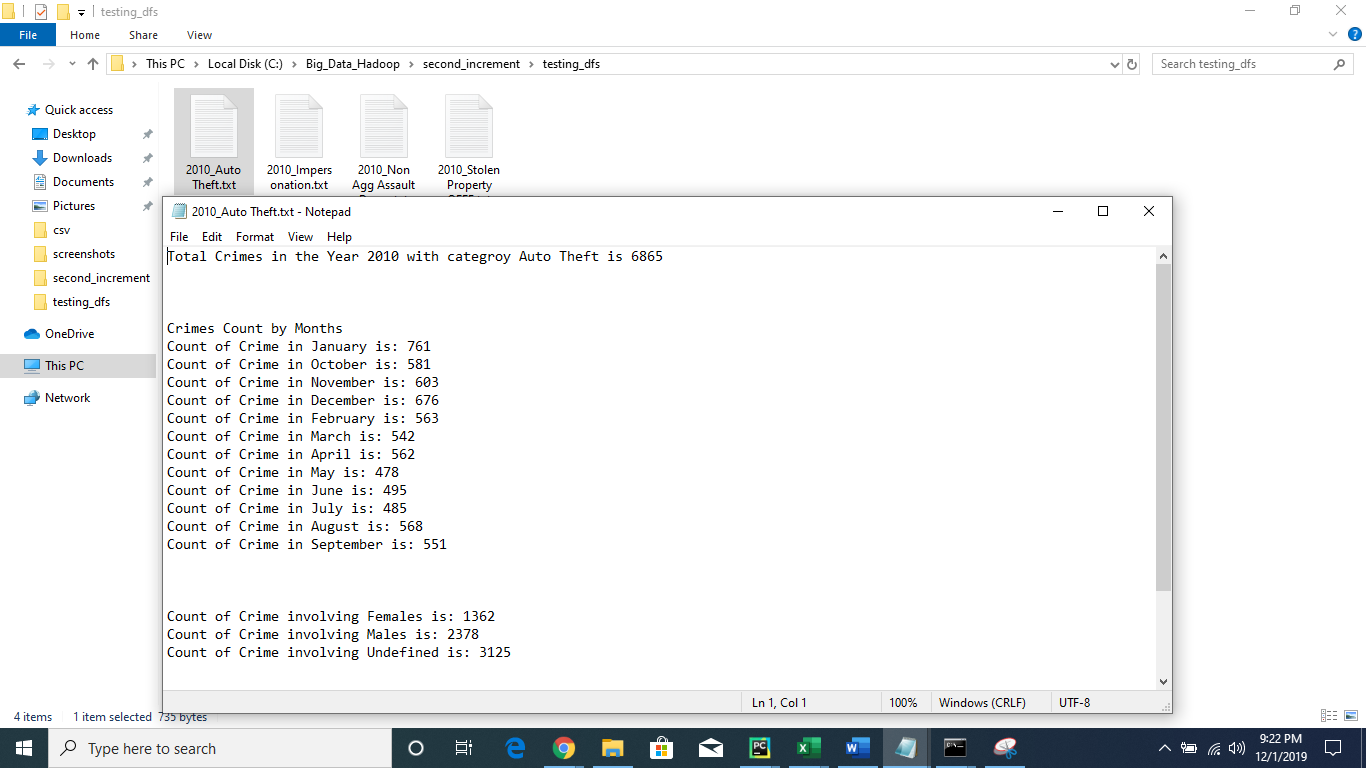
* PySpark Streaming

Pyspark Streaming solves the big problem of dividing our data into different batches according to year and category of the crime and do the analysis or processing on it individually. In the project, we have used pyspark text streaming to generate batches of our main file into years with one type of crime in each batch.

After generating these batches we read the file through our text stream and generate a dataframe from it. After generating the dataframe we analyze different properties of crime per year and save it in an individual text file to make it user-friendly for the other people.

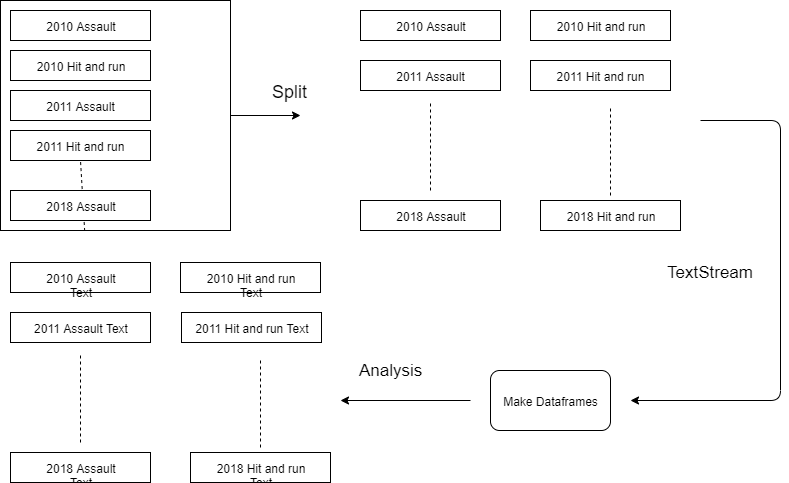






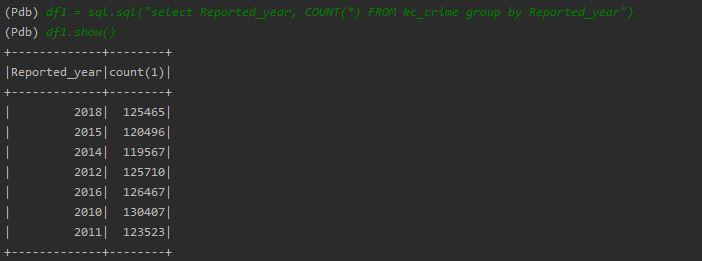
In the first screenshot, we see that batches of CSV files are created. After that, each CSV file is read through text stream and then a text file is made the contents of which can be shown in the last screenshot.

Below is the system level diagram of it.

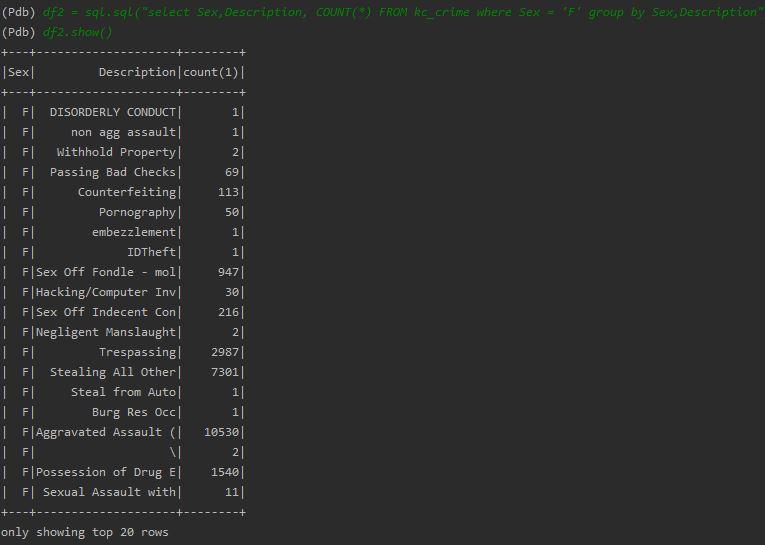


Below are some previous queries with a simple analysis.

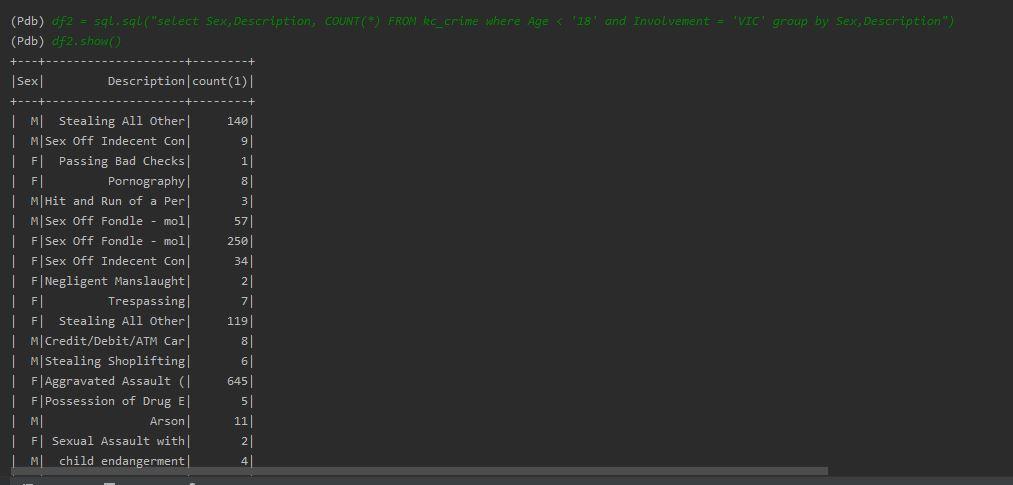
What is the total number of crimes per year



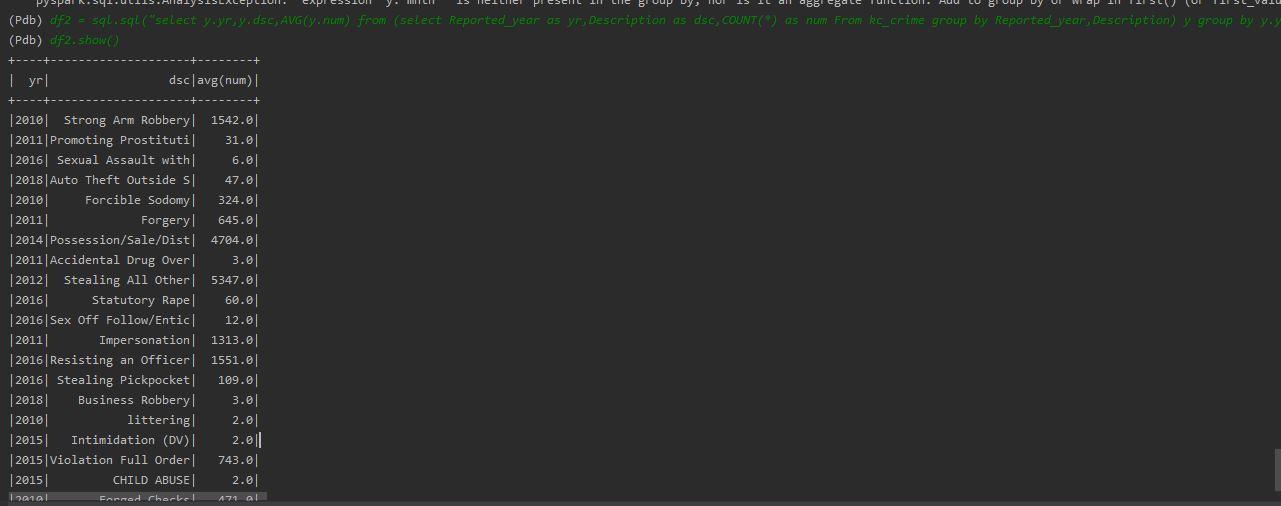
Counting the females who have been the victims of crimes.



Analysis of crime which involved minors.



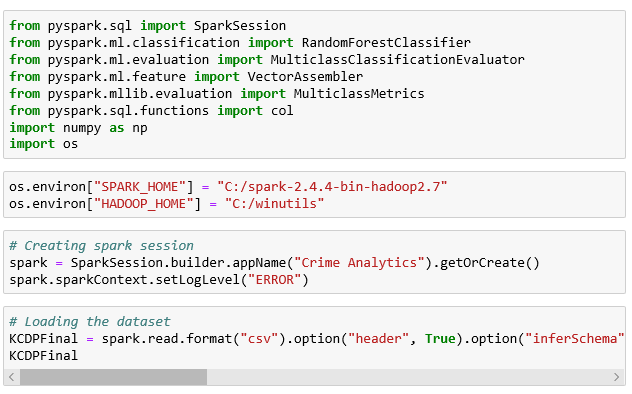
• Average analysis of crimes per year



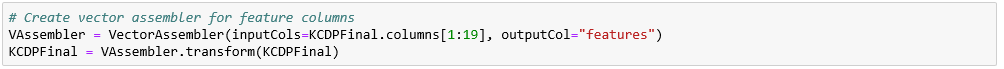
**Analytics Implementation**

One of our implementations is the analytics part. We have done some machine learning algorithms on our dataset. For instance, Random Forest, which is for classification, was used to train our model. We also performed Naive Bayes and implemented Decision tree, which gave us better results.

We first imported the libraries we needed and loaded the dataset in SparkContext.



Then we used vector assembler for feature columns.

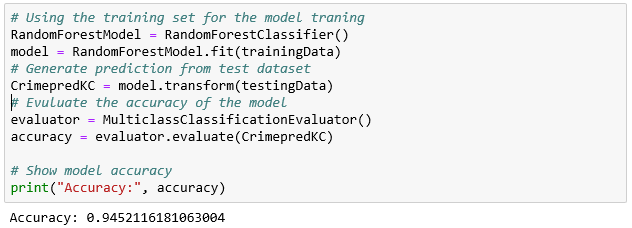


Next, we split the dataset into to parts, 70% for training and 30% for testing.

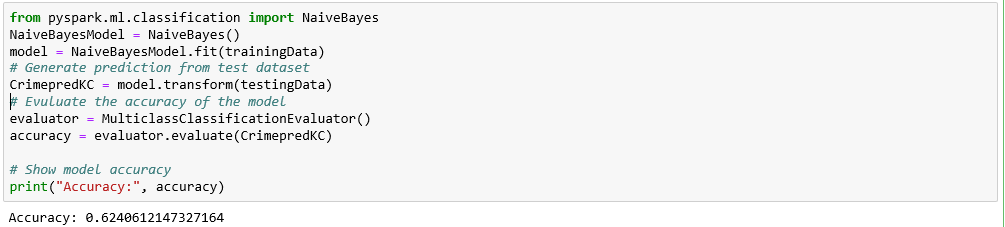


Finally, we have implemented the following algorithms:

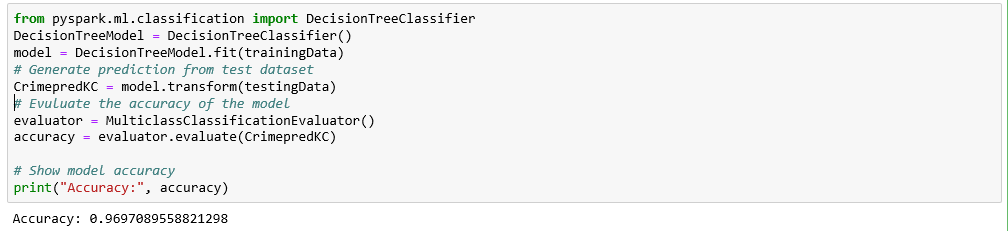
1- Random Forest.



2- Naive Bayes



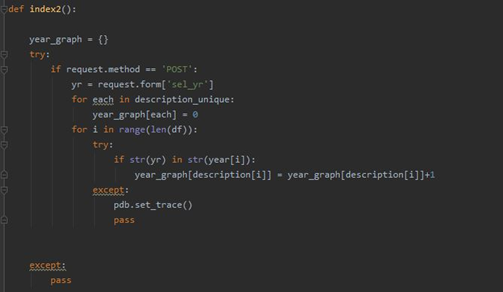
3- Decision Tree

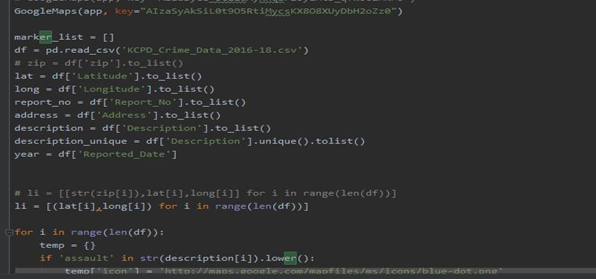


We found out that Decision Tree gave use more accuracy than the rest while we were expecting to have better accuracy using Random Forest.

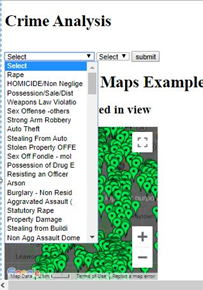
* **Simple Visualization on Flask**

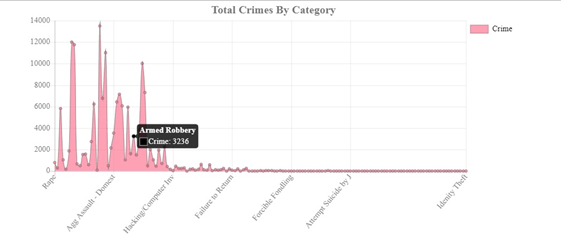
Some Simple visualization code has been implemented on flask which shows the locations of crimes on google maps.

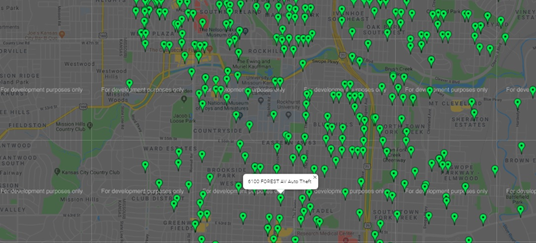




**Result**







**Conclusion**

To conclude, this project has many phases starting with cleaning the dataset and ending with doing analysis and analytics. For Hive, SQL context, and the time analysis, we can say that a lot of answers can be achieved by only doing the yearly, monthly and weekly analysis. Simple queries can help us achieve major milestones which can not only help us take precautions regarding those crimes but also help us to eradicate crimes using these data. In addition, by training a model with different algorithms, it gave us a better choice for the best algorithm. Then we were able to predict the crime that firearms were used

**Future Work**

For future perspective, we have data that focuses on the time at which the crime has occurred. The analysis of these times of the day can help us to predict and analyze which part of the day, the crime rate is more.

Spark Stream helps us to achieve fast processing. In our project spark streaming has helped us to develop different data sets which can not only be used by other data scientist but also help us to focus only on specific crimes rather than on whole datasets.

**Project Management**

1. **Implementation status report**

* **Work completed**

1. Gathering and cleaning more data in order to have big data was done by Khalid Dhabbah (6) 100 %.
2. Implementing queries in Hive with its visualization was done by Khalid Dhabbah (6) - 100%.
3. Pyspark implementation was done by Shah Rukh Khan (15) - 100%.
4. Cassandra CQL Implementation was done by Shah Rukh Khan (15) - 100%.
5. Pyspark SQL Context with time analysis done by Awais Ahmad Khan (14) - 100%
6. Pyspark Text Streaming and analysis of batched data done by Awais Ahmad Khan (14). 100%
7. Some visualization of data and its analysis queries were done by Awais (14) - 100%
8. Data Analytics was done by Khalid Dhabbah (6) - 100%

**VI. Issues and Concerns**

Some of the issues which we faced during the development of our projects are:

* The project in itself is really big having multiple applications in it. It is really hard to choose which application to pick and which to skip and most of them are integrated with each other.
* The datasets are picked from sites which are mostly consistent but small inconsistencies created big issues when loading in the code for generating tables and queries.
* Time constraint is another issue which has caused us to skip multiple analysis which could have been done to increase the efficiency of our work.
* Integration of multiple codes together involving machine learning and other pyspark modules.

**VII. References**

**[1].** KCPD Crime Data 2018: <https://data.kcmo.org/Crime/KCPD-Crime-Data-2018/dmjw-d28i/data>.

[2]. Pandas.DataFrame.dropna: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html>

[3]. Drop Rows with NAN/ NA: <http://www.datasciencemadesimple.com/drop-rows-with-nan-na-drop-missing-value-in-pandas-python-2/>

[4]. Pandas.DatetimeIndex: <https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.DatetimeIndex.html>

[5]. Pivot Tables: <https://www.excel-easy.com/data-analysis/pivot-tables.html>

[6]. Apache-hive: <https://mapr.com/products/apache-hive/>

[7]. Using Hive to perform some queries: <https://github.com/Dhabbah/CSEE5590-490-Big-Data-Programming/wiki/ICP4>

[8]. Numpy queries on Dataframes. <https://docs.scipy.org/doc/numpy/reference/arrays.indexing.html>

[9]. DSTREAMS examples <https://gokhanatil.com/2018/04/pyspark-examples-5-discretized-streams-dstreams.html>

[10]. Dataframe and RDDs <https://medium.com/@xuweimdm/how-to-make-a-dataframe-from-rdd-in-pyspark-c34f2888ea8>

[11]. Pyspark SQL modules. <https://spark.apache.org/docs/2.1.0/api/python/pyspark.sql.html>

[12]. User Defined Functions . <https://docs.databricks.com/spark/latest/spark-sql/udf-python.html>

[13]. SQL Aggregate functions. <https://www.w3resource.com/sql/aggregate-functions/Max-with-group-by.php>

[14]. New Column operations on dataframes. <https://stackoverflow.com/questions/33681487/how-do-i-add-a-new-column-to-a-spark-dataframe-using-pyspark>

[15]. Conversion of dataframe to python lists. <https://stackoverflow.com/questions/38610559/convert-spark-dataframe-column-to-python-list>