*Scalable Data Analysis*

[*New York Parking Tickets Data*](https://data.cityofnewyork.us/City-Government/Parking-Violations-Issued-Fiscal-Year-2018/pvqr-7yc4)

*Final Project Report*

*Satvir Singh*

*01756353*

*dept.of Computer and Information Science*  
*University Of Massachusetts*

*Dartmouth*Dartmouth, MA, USA

slnu@umassd.edu

In this project, I undertook Data Analysis over Three [New York Parking Tickets Datasets](https://data.cityofnewyork.us/City-Government/Parking-Violations-Issued-Fiscal-Year-2017/2bnn-yakx), The NYC Department of Finance collects data on every parking ticket issued in NYC (~10M per year!). This data is made publicly available to aid in ticket resolution and to guide policymakers.

The Datasets have collectively 30 Million Rows and 43 column attributes :-

* Summons Number: Number
* Plate ID:Plain Text
* Registration State: Plain Text
* Plate Type: Plain Text
* Issue Date: Date & Time
* Violation Code: Number
* Vehicle Body Type: Plain Text
* Vehicle Make: Plain Text
* Issuing Agency: Plain Text
* Street Code1: Number
* Street Code2: Number
* Street Code3: Number
* Vehicle Expiration Date: Number
* Violation Location: Plain Text
* Violation Precinct: Number
* Issuer Precinct: Number
* Issuer Code: Number
* Issuer Command: Plain Text
* Issuer Squad: Plain Text
* Violation Time: Plain Text
* Time First Observed: Plain Text
* Violation County: Plain Text
* Violation In Front Of Or Opposite:Text
* House Number: Plain Text
* Street Name: Plain Text
* Intersecting Street: Plain Text
* Date First Observed: Number
* Law Section: Number
* Sub Division: Plain Text
* Violation Legal Code: Plain Text
* Days Parking In Effect: Plain Text
* From Hours In Effect: Plain Text
* To Hours In Effect: Plain Text
* Vehicle Color: Plain Text
* Unregistered Vehicle?: Plain Text
* Vehicle Year: Number
* Meter Number: Plain Text
* Feet From Curb: Number
* Violation Post Code: Plain Text
* Violation Description: Plain Text
* No Standing or Stopping Violation: Text
* Hydrant Violation: Plain Text
* Double Parking Violation: Plain Text

TOOLS USED :

* Python on Jupyter NoteBook
* Pandas
* PySpark
* AWS Services : EMR , S3

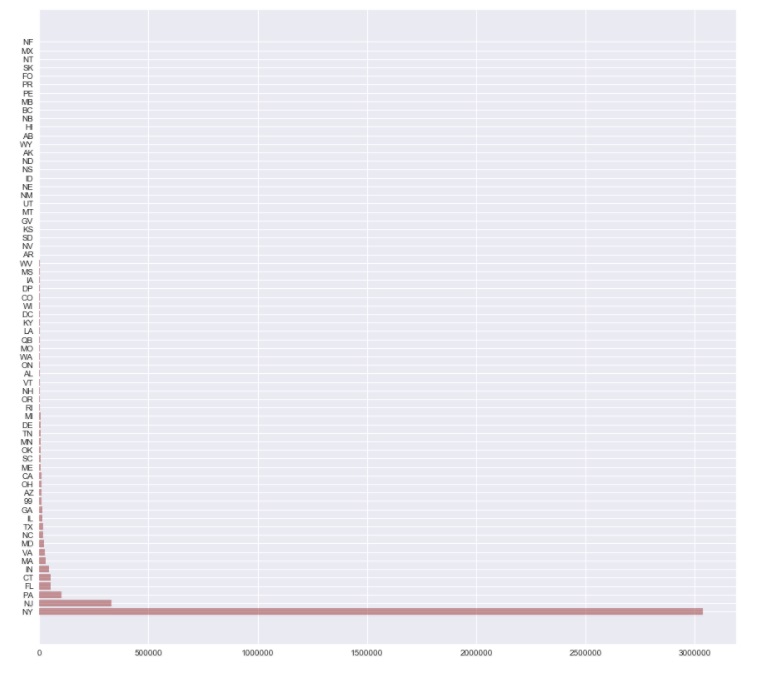
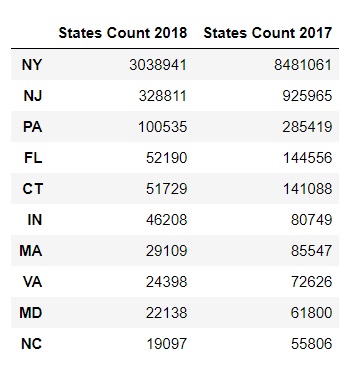
M4.large

3 nodes

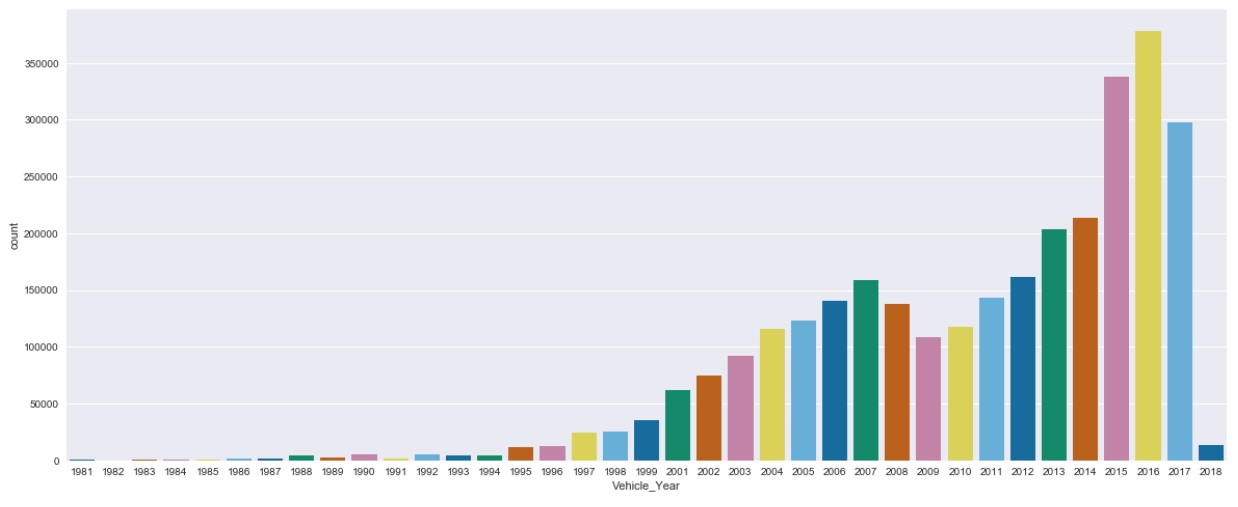
* Mobaxterm, S3 Browser

ANALYSIS TIMELINE :

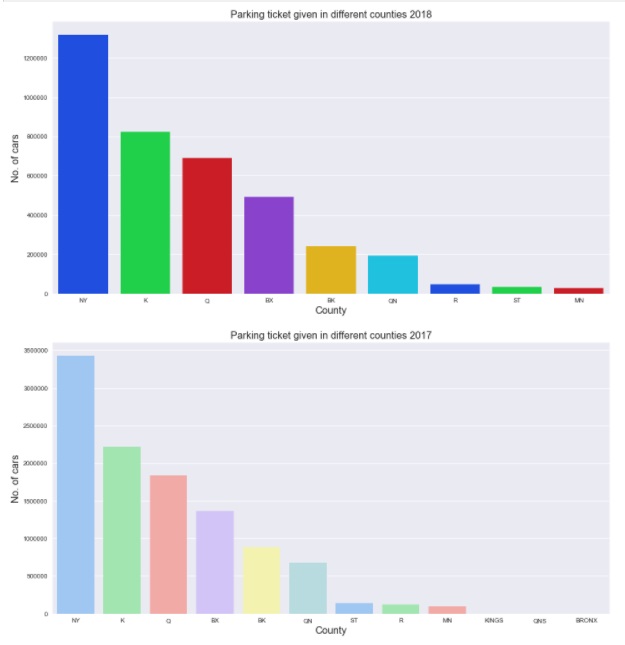
* 1. We start with installing Jupyter notebook on our local machine through Anaconda navigator and then start quering on the Dataset with Pandas and other libraries such as geopy, matplotlib, seaborn, numpy etc.
  2. We query on the data using filtering functions and group by functions as we refine our results and then we plot them using barplots.



The above table shows the total number of violations as per car registration State for the Relative year.



The Bar above shows the Car Make year of the Cars which got the parking Tickets.

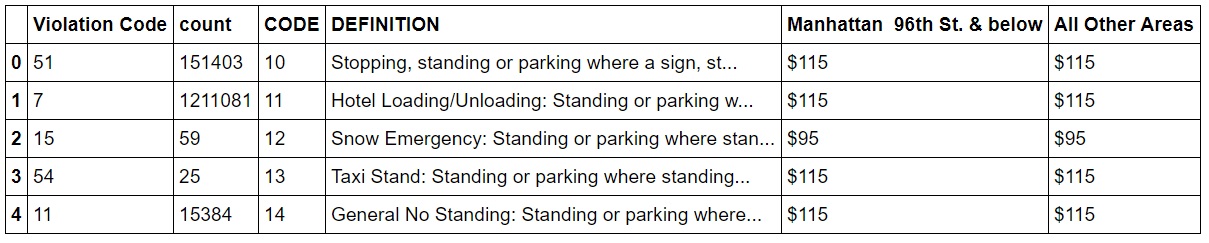


The Two Barplots show the parking Tickets given in two counties for year 2017 and year 2018

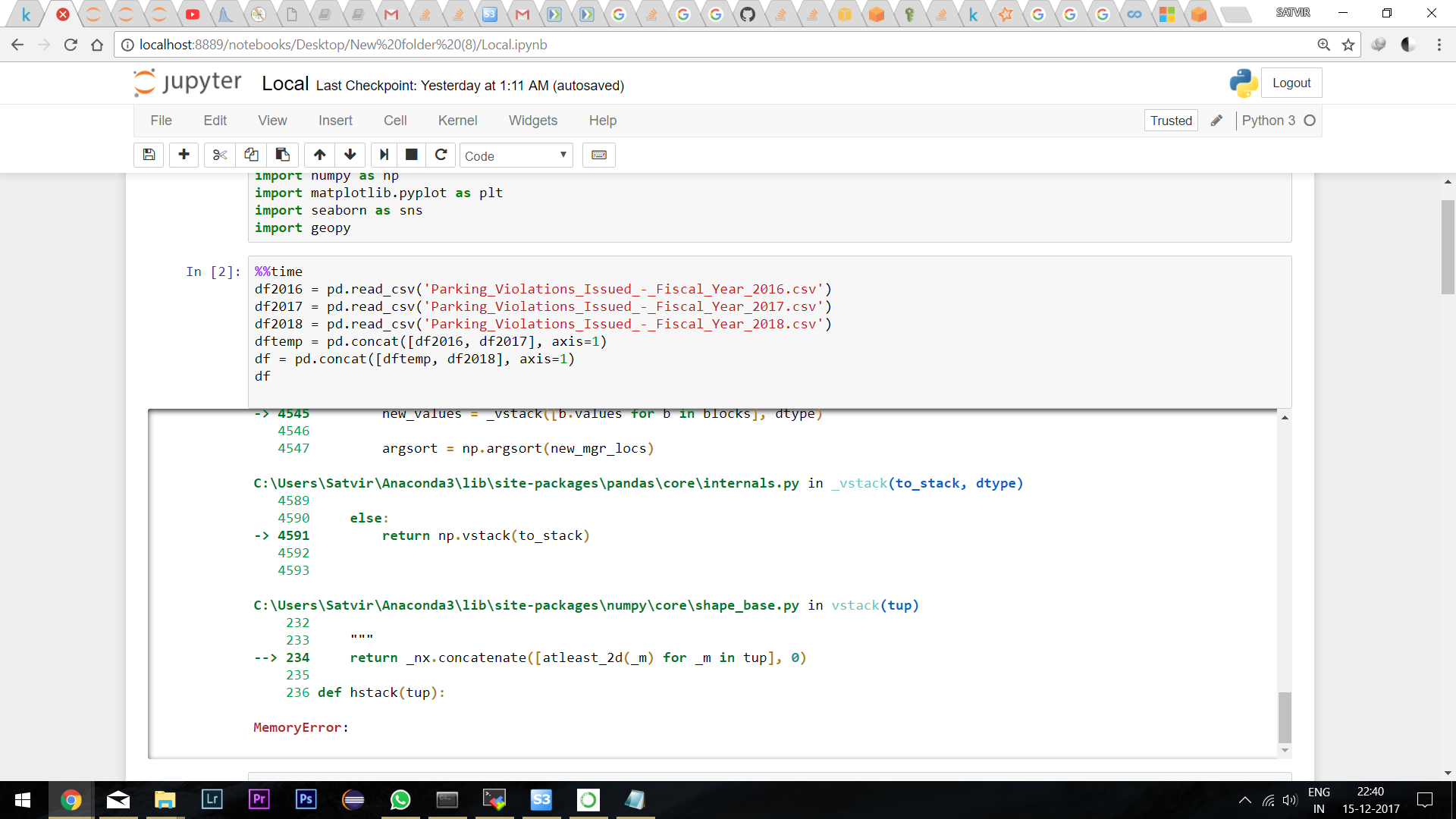
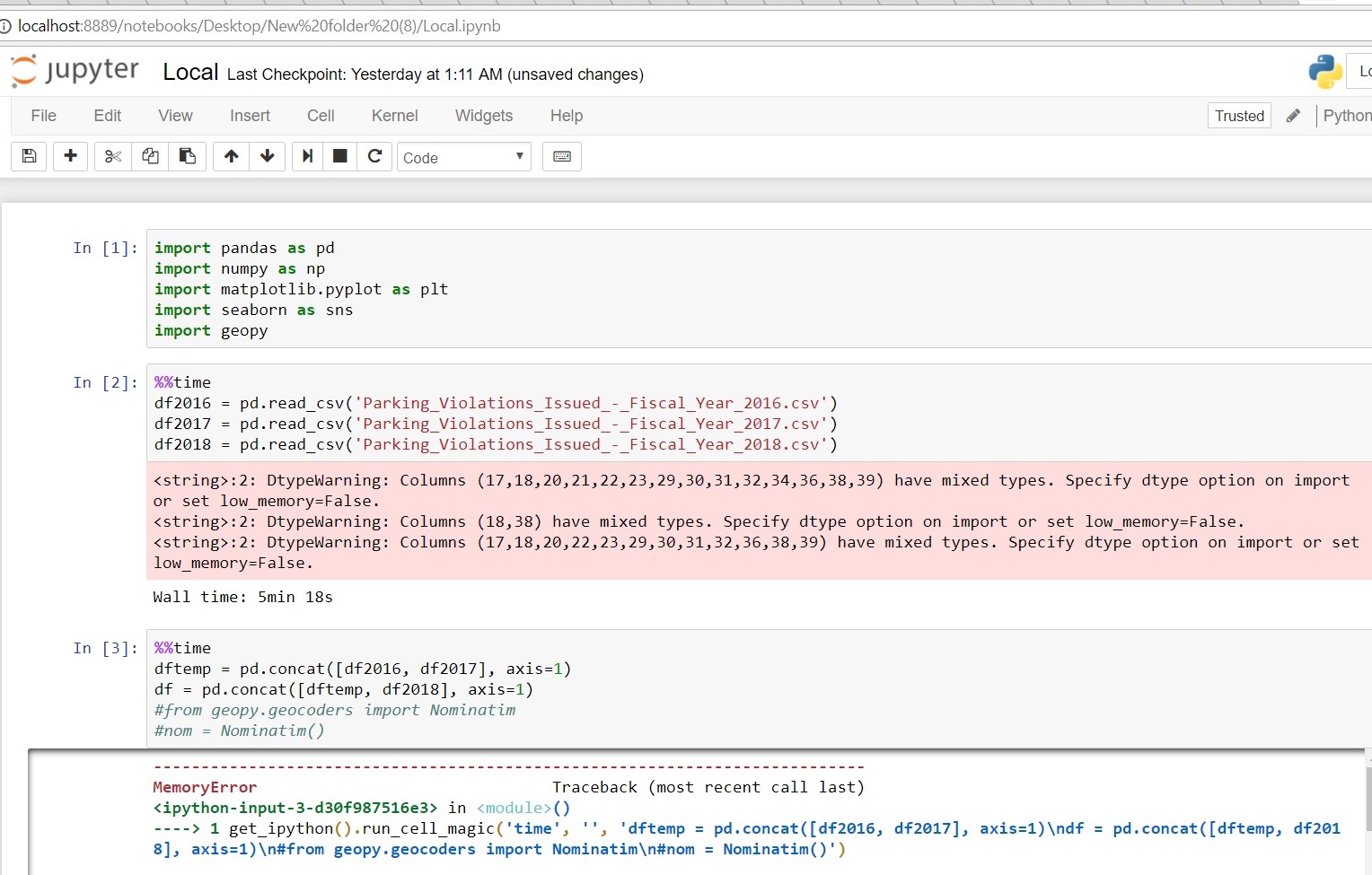


The Two Barplots show the data over two years of the most common violations and their Count, as we can see No. 20,36,38 and 34 are the most common parking tickets.

So to get the information about the Violation code and its definition we downloaded another [dataset](https://data.cityofnewyork.us/Transportation/DOF-Parking-Violation-Codes/ncbg-6agr) and merged with our current dataset to get views on the violation caus and the ticket charges for that violation code.

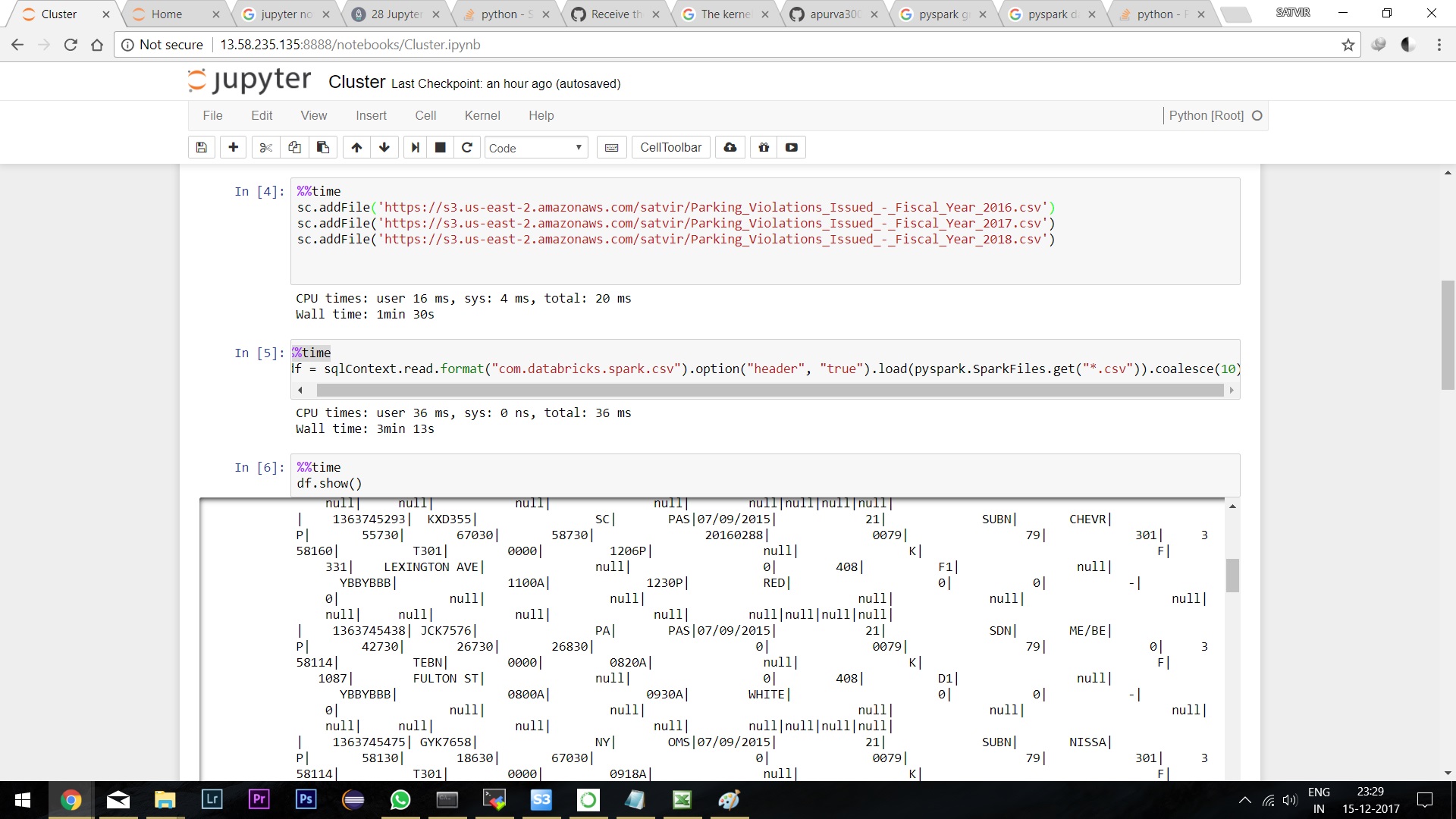


* 1. During out analysis, we first encountered scalability issues in the form of memory errors where our local machine could not do load 3 datasets sometimes and even if it loaded the dataset, we could not concatenate the datasets because of these errors, So we did individual queries on the datasets on our local machine.



We tried geopy, to extract longitude and latitude and eventually plot ticket locations over map but geopy timed out everytime because over dataset was large and geopy had limit to 1000 data only.

* 1. For doing larger queries which required more memory and for our working on one whole merged dataset instead of 3 individual dataset, we took our project to Amazon EMR which is a hadoop framework and we used Pyspark over pandas as even here we had out of memory issues while doing pandas concat functions, so pyspark dataframes are faster because they take data in memory and we had no memory issues. We kept our datasets on an S3 bucket, which works faster as SSD than our local machine SATA.



Datasets loaded collectively and concatenated well using pyspark dataframe over the cluster.

CONCLUSION : We noticed relatively faster loads and query processing on our cloud cluster, given the faster workflow of pySpark and because of our instance running on 3 nodes. Working with Spark and AWS provided a view of how big Data analysis can be undertaken efficiently and AWS`s helps scale you better as if our needs increase, we can simply add more nodes and storage to improve our Data Analysis needs