NYC Parking Ticket

Do certain factors have an association with a higher likelihood of issuing parking tickets?

Team: Mollika Tahsin and Subhransu Sekhar Nanda

1. INTRODUCTION

Parking tickets are an important source of revenue of many cities' general funds and they are also crucial for ensuring that cars are not left in a place that can block the road or be dangerous for others. However, although the application of parking ticket laws should be universal throughout the city under a given city's jurisdiction, sometimes there can be an inherent bias in 'which, when and where' vehicles are ticketed more due to factors other than road parking rules. In order to ensure that parking rules are administered in a fair and universal manner, it is important to investigate these confounding variables. In this paper, we analyze parking tickets issued in NYC to find out if there are certain factors that are positively correlated with a higher likelihood of getting a parking ticket.

2.1 DATA SUMMARY

The dataset was sourced from NYC's Department of Finance, which gathers NYC's parking ticket issue data, amounting to almost 10 million tickets per year. It gets updated every third week of the month. This data is made available publicly to help with ticket resolutions and policy making guidance. The dataset consists of NYC's ticket data from August 2013 to June 2017, making upto 8.36GB of data with a total of 42.3 million instances. However, the fiscal year considered is from July 1 to June 30.

This was a dataset that was large enough for the requirements of this project. Secondly, the parking tickets issued and the variables considered in the dataset all had a lot of real world implications which can be leveraged by NYC and other big cities to understand ticketing behavior and address bias accordingly. Hence, this dataset was found to be interesting.

2.2 DATA COLLECTION

For data collection, the options were to choose between Amazon EMR, DataBricks, and Jupyter Notebook. DataBricks was eventually selected because it is feature rich and helps with vibrant, insightful and simple visualizations. Kaggle API and Shell commands were used to automate data collection. The data was downloaded from the original Kaggle link and unzipped into csv files. In order to integrate design patterns, the unzipped dataset files were converted to Databricks file system (DBFS) in order to load the dataset files into clusters.

2.3 DATA PREPARATION :

For data preparation, first the “printSchema” function was used to get an idea of all the data types for all the columns. Then, the “describe” function was used to give a summary statistic of the loaded dataset file. A total of 9,100,278 rows and 51 columns were found. For further analysis, the null and missing values in each column were found and all the columns with a high number of null or missing values were removed. The “dropDuplicate” function was used to remove all duplicate rows, if any. There seemed to be no duplicate values in all the fiscal years except 2014-2015 fiscal year. Further data preparation activities were conducted for each respective question as mentioned below under the “DATA ANALYSIS” section.

2.4 DATA CHALLENGES

When dataset files were unzipped, those files could not be loaded initially because they were not yet converted into DataBricks file system. After using different algorithms like “describe” function and others, it was found that the dataset for each fiscal year had a hefty amount of missing or null values. Because the dataset size was huge and the DataBricks community edition (which only offers two CPU cores) was being used for this project, there was a huge computational time lag between running the code and getting the results. There were also inconsistencies in data types for columns across the dataset. Hence, the data types had to be changed to the appropriate type before any computation. After further investigation, some columns were found to have high cardinality (that is, many number of categories in a single column). For example, for the “Vehicle_Color” column, for the color “GREEN”, there were a variety of labels like “GN, GR, GRN” to refere to the same green color of the vehicle. Thus it can be seen that a standardized naming convention was not followed by the source data collector for each categorical column. Hence, the labels were replaced with the right color label.

2.5 DATA ANALYSIS

The dataset was analyzed one fiscal year at a time. The results were used to answer the following 6 real-world questions:

i) When are tickets most likely to be issued? Is there any seasonality?

The DataFrame API method was used to answer the first question. Two columns, "Issue Date" and "Summons Number" were analyzed. First, the "Issue Date" datatype was converted from string to date datatype. Then both these columns were selected and the dates were filtered according to the respective fiscal year (July 1 to June 30) as specified in the Kaggle overview of the dataset. Then aggregate functions like “groupBy” and “sorting” were used to create a DataFrame by counting the "Summons Number" column values as Ticket Frequency and ordering them in descending order. The “Month” values were extracted from the "Issue Date" column and each “Month” was then assigned to a season category based on public meteorological data. Then a new dataframe was created and the tickets were consolidated by seasons. From this analysis, it was found that the highest number of tickets were issued in Spring for all the years except for the fiscal year 2015 - 2016.

Spark SQL queries were used to analyze the dataset further for the rest of the questions. In order to do that, the white space in the column names had to be replaced with underscore to adhere to the SQL queries' naming convention. Secondly, the DataFrame was converted to a temporary table so that SQL queries can be used to analyze the data further. The other questions are as follows:

ii) Out of all the vehicles that were issued tickets, which states were they most registered at?

To answer this question, an SQL query was written to select "Registration_State" and "Summons_Number" columns and their column values were added to a DataFrame where all the values of "Registration_State" belonged to the list of all the US states and the DataFrame was in descending order with respect to the count of "Summons_Number" values as "TicketFrequency". It was found that the greatest number of vehicles ticketed across all years were registered in New York state.

iii) Which color vehicles were the most issued tickets for the given fiscal year?

To answer this question, a DataFrame was created using an SQL query for selecting two columns "Vehicle_Color" and the count of "Summons_Number"(as "TicketFrequency" in its descending order). It was found that the "Vehicle_Color" has high cardinality since different color labels were used to denote the same color. In order to tackle this cardinality, the first 100 rows and all labels were replaced with their appropriate consistent values. However, it should be noted that the value "OTHER" was included to capture all other values which didn't fit the color labeling conditions. The outcomes showed that due to high cardinality a big chunk of labels couldn't satisfy the defined labeling conditions and hence were labeled as "OTHER". If the "OTHER" label is ignored, then black can be seen as the color of the most ticketed vehicles.

iv) In which county were most tickets issued in the given fiscal year?

The two columns, “Violation_County” and “Summons_Number” were queried by counting the latter column’s values into the column TicketFrequency. NYC has 5 counties, namely New York County (NYC), Kings County (KINGS), Queens County (QUEENS), Richmond County (RICHMOND) and Bronx County (BRONX). From the output, it was found that some values had been mislabelled. Each column’s datatype was checked and the mislabelled values were consolidated into 5 properly labeled county names. From the output, NYC seemed to be the county where most tickets were issued.

v) What was the year-of-manufacture for most vehicles ticketed in the given fiscal year?

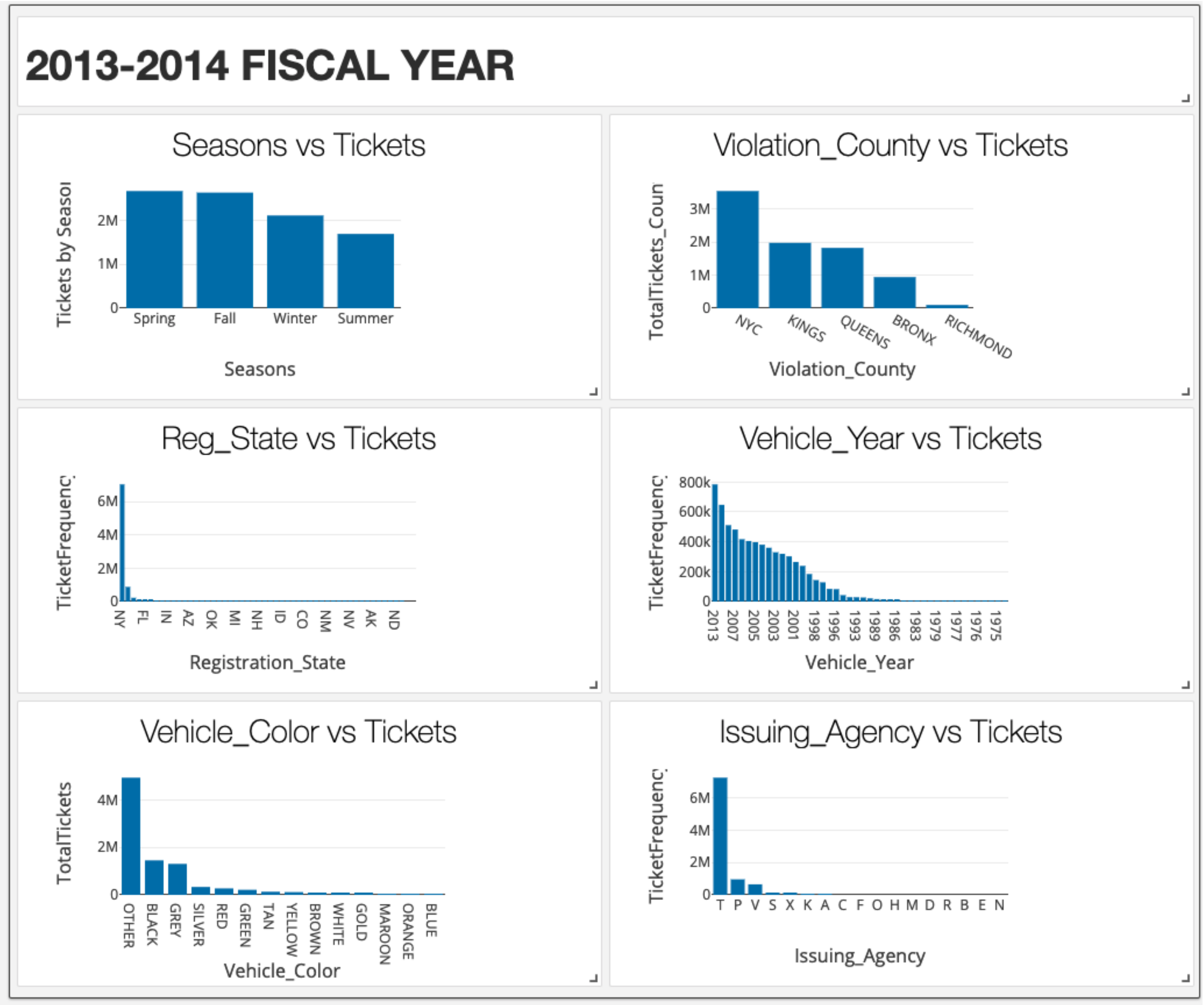
As done earlier, SQL query was used to create DataFrame by selecting “Vehicle_Year” and the count of “Summons_Number” as “TicketFrequency”. It had to be ensured that only the cars manufactured in the respective fiscal year or older were considered for the given fiscal year. It was found that newer vehicles were ticketed more than older vehicles.

vi) Which issuing agency issued the most number of tickets this fiscal year?

Here the same 2 column query concept as above were used with “Issuing_Agency” being the first variable and the count of “Summons_Number” as TicketFrequency as the second variable. By running the query and ordering by “TicketFrequency” in the descending order, it was found that the issuing agency with the label "T" had issued most tickets across all fiscal years.

2.6 DATA VISUALIZATION

For answering the respective questions above, the dataset was visualized in the Databricks notebook for each fiscal year. The dashboards below show the relationship between ticketing and the variables (Season, State Registration, Vehicle Color, County, Vehicle Year, and Issuing Agency) for each of the 4 fiscal years. The respective variables show similar patterns of ticketing across all years as seen in the dashboards below. Hence, they can be used to support our conclusions below. However, it should be noted that since multiple numerical columns had null or missing values, the columns could not be used for other types of numerical variable visualizations like scatterplots.


2013-2014 FISCAL YEAR

Seasons vs Tickets

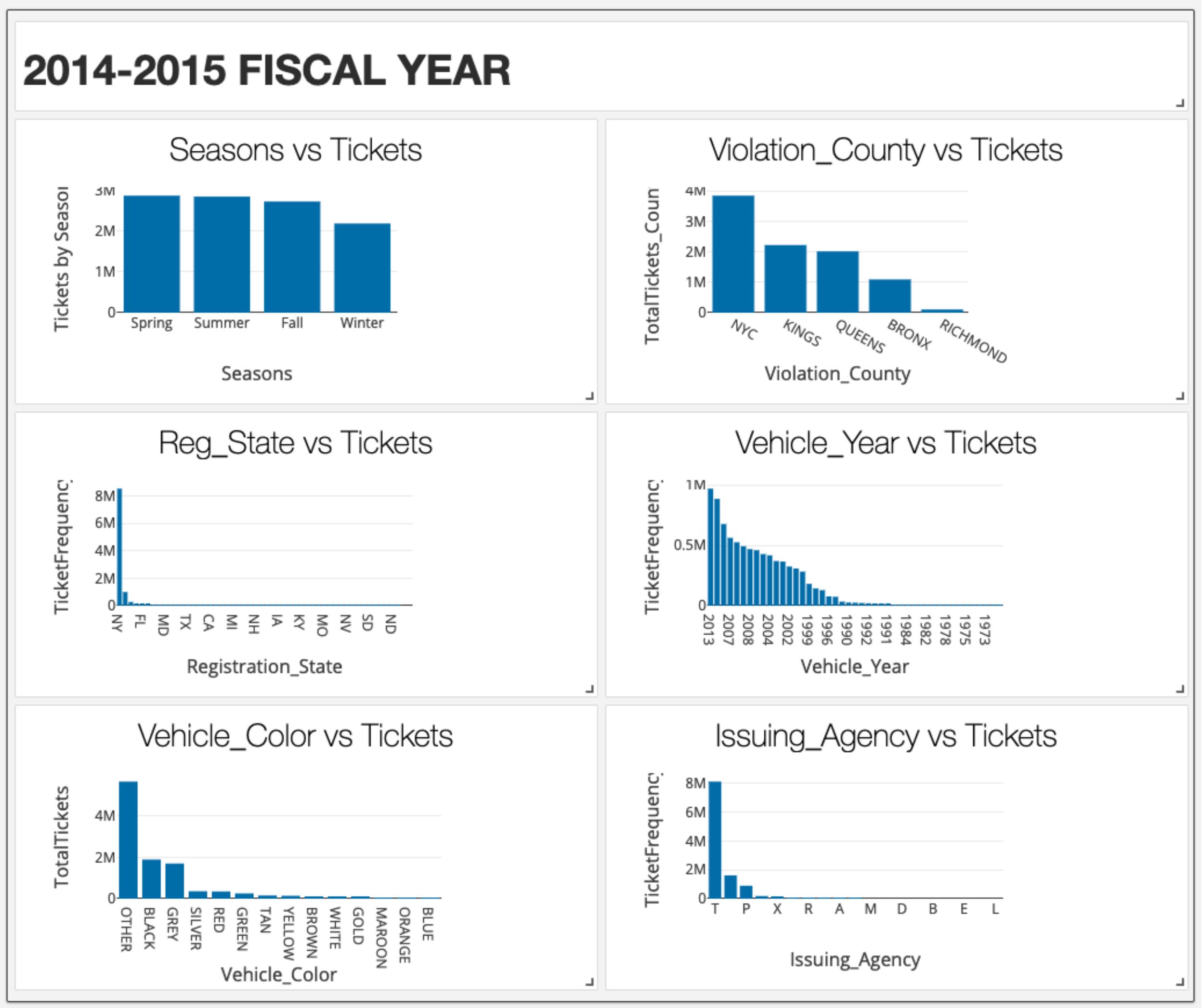
Violation_County vs Tickets

Reg_State vs Tickets

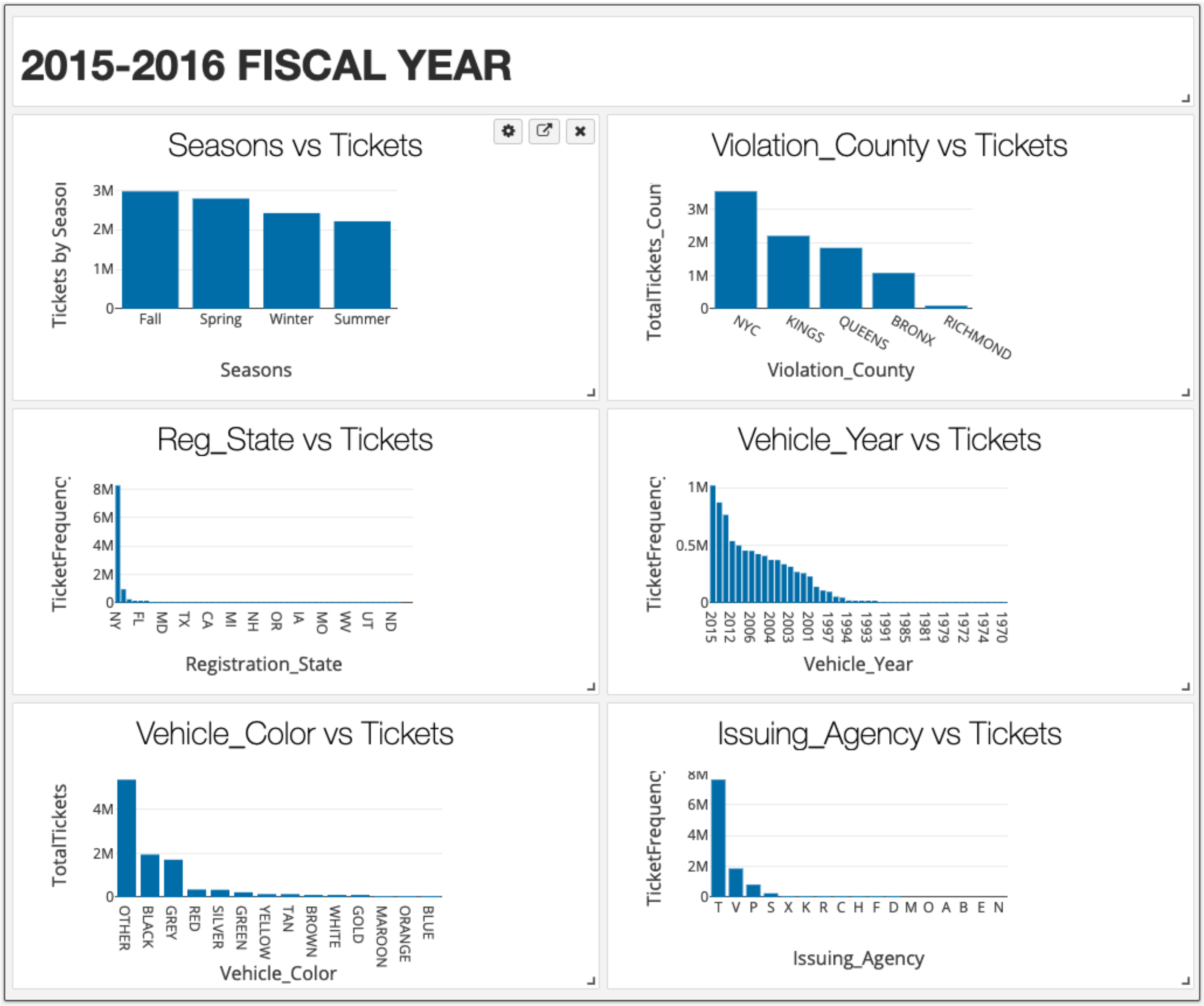
Vehicle_Year vs Tickets

Vehicle_Color vs Tickets

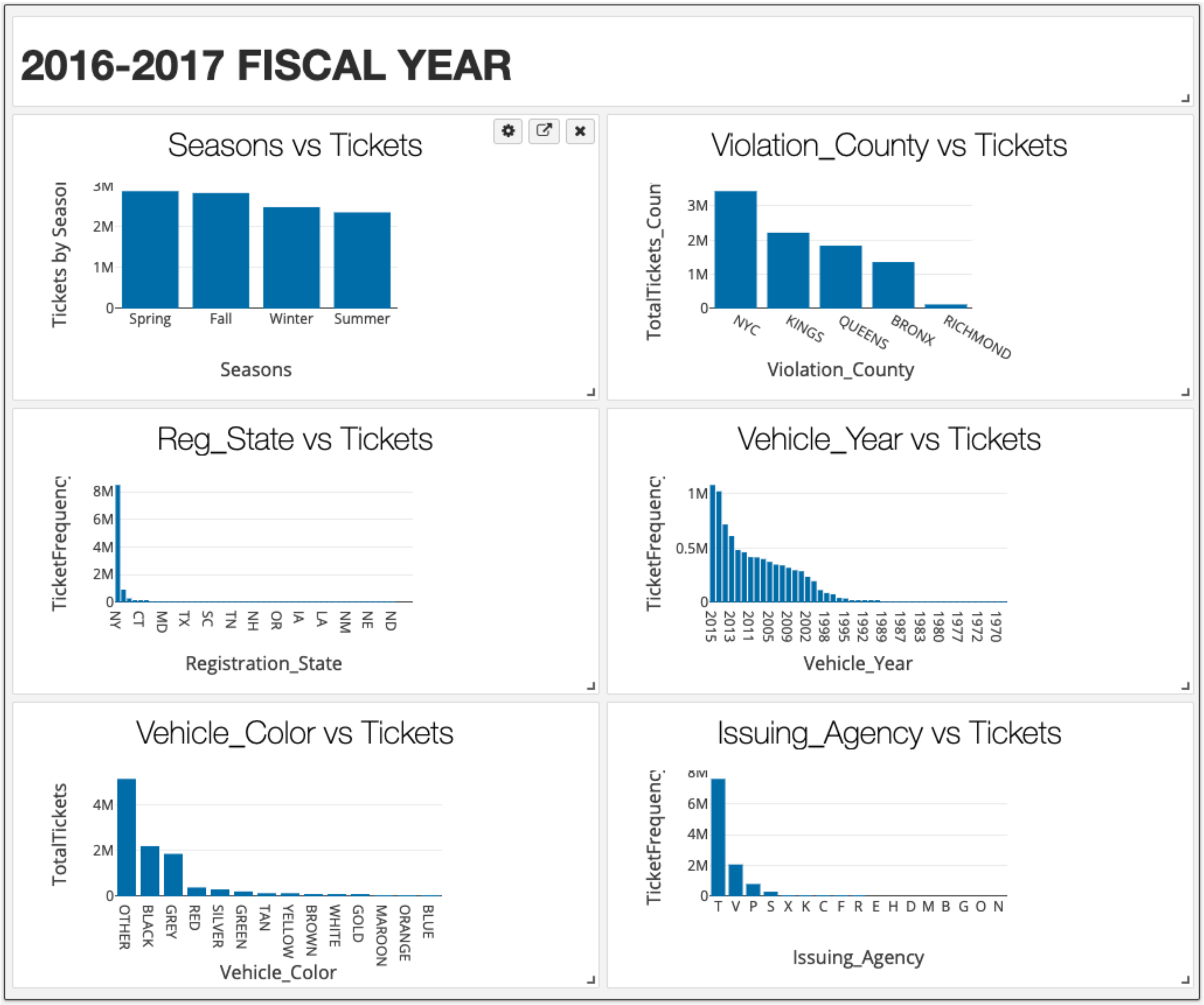
Issuing_Agency vs Tickets



2014-2015 FISCAL YEAR



2015-2016 FISCAL YEAR



3. CONCLUSION

Some general conclusions can be drawn by considering the insights gained from each fiscal year. More often than not, “Spring” is the season when most tickets are issued with “Fall” being a close second, with “Fall” even superseding “Spring” in 2015-2016. This can be explained by either the agencies being more active during warmer times after winter (easier to patrol on foot) or there could be more cars as people prefer to be outside more when spring comes after an indoor winter lifestyle. Secondly, New York state registered vehicles got the highest number of tickets. This is understandable since it is likely that there will be more New York state registered vehicles in New York City. Thirdly, although there were a lot of mislabeled values, Black vehicles were ticketed the most. It can be because there are a higher number of black vehicles on the streets compared to other colored vehicles. It was also seen that New York County issued the most tickets which is understandable since more of the office buildings are located in Manhattan. There is a consistent trend of newer vehicles getting ticketed across all years. It could be because there are a higher numbers of newer cars on streets as NYC has a generally high income per capita. The issuing agency labeled "T" has issued more tickets than any other issuing agency over the years. These analysis insights can be leveraged by policymakers to address inherent bias in ticketing and design more universal impartial ticketing policies. However, further investigation is necessary to evaluate and validate the results' explanation assumptions mentioned in this conclusion.

4. ACKNOWLEDGMENTS

We would like to thank Professor Cort Lunke for everything we have learned so far about Big Data Engineering. This project consumed a huge amount of work, research and dedication. Still, implementation would not have been possible without clear instruction and proper guidance throughout the course.

5. REFERENCES

1. “%STARTSWITH - InterSystems SQL Reference - InterSystems IRIS Data Platform 2020.4.” InterSystems. Accessed May 17, 2021. https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_STARTSWITH.

2. 21 Mar 2019Dipanjan (DJ) Sarkar (Red Hat)&nbsp;Feed305up1 comment, and Dipanjan (DJ) Sarkar (Red Hat). “How to Use Spark SQL: A Hands-on Tutorial.” Opensource.com. Accessed May 17, 2021. https://opensource.com/article/19/3/apache-spark-and-dataframes-tutorial.

3. “Advancing Spark - The Hidden Databricks Dashboard Tool!” YouTube. YouTube, August 20, 2020. https://www.youtube.com/watch?v=3va2K-vfTQE&amp;t=791s.

4. Ankit GuptaAnkit is currently working as a data scientist at UBS who has solved complex data mining problems in many domains. He is eager to learn more about data science and machine learning algorithms. “Pyspark Data Frames: Dataframe Operations In Pyspark.” Analytics Vidhya, April 1, 2020. https://www.analyticsvidhya.com/blog/2016/10/spark-dataframe-and-operations/#:~:text=In%20Apache%20Spark%2C%20a%20DataFrame,but%20can't%20change%20it.

5. “Automatic Exporting of Dashboard.” Automatic exporting of dashboard - Databricks Community Forum. Accessed May 17, 2021. https://forums.databricks.com/questions/11942/automatic-exporting-of-dashboard.html.

6. Carnes, Beau. “Basic SQL Commands - The List of Database Queries and Statements You Should Know.” freeCodeCamp.org. freeCodeCamp.org, April 28, 2021. https://www.freecodecamp.org/news/basic-sql-commands/.

7. Christophe, 13scoobie, and Anton. “Running SQL Queries on DataFrames in Spark SQL [Updated].” Discourse, April 13, 2016. https://discourse.snowplowanalytics.com/t/running-sql-queries-on-dataframes-in-spark-sql-updated/119.

8. curtispcurtisp 1, zero323zero323 282k7979 gold badges854854 silver badges878878 bronze badges, hamedhamed 1, and Janaka EkanayakeJanaka Ekanayake 2. “Spark DataFrame TimestampType - How to Get Year, Month, Day Values from Field?” Stack Overflow, February 1, 1964. https://stackoverflow.com/questions/30949202/spark-dataframe-timestamptype-how-to-get-year-month-day-values-from-field.

9. “Dashboards.” Dashboards | Databricks on AWS. Accessed May 17, 2021. https://docs.databricks.com/notebooks/dashboards.html#dashboards-notebook.

10. “Data Types.” Data Types - Spark 3.1.1 Documentation. Accessed May 17, 2021. https://spark.apache.org/docs/latest/sql-ref-datatypes.html.

11. EdamameEdamame 17.3k4444 gold badges143143 silver badges254254 bronze badges, GregologyGregology 1, AkavallAkavall 67.7k3939 gold badges179179 silver badges227227 bronze badges, rjurneyrjurney 4, Kamaldeep SinghKamaldeep Singh 39322 silver badges88 bronze badges, and Aaron RobesonAaron Robeson 12311 gold badge33 silver badges1010

bronze badges. “Pyspark: ValueError: Some of Types Cannot Be Determined after Inferring.” Stack Overflow, July 1, 1965. https://stackoverflow.com/questions/40517553/pyspark-valueerror-some-of-types-cannot-be-determined-after-inferring.

12. Gaatjeniksaan, gaatjeniksaangaatjeniksaan 99711 gold badge88 silver badges1717 bronze badges, mrsrinivasmrsrinivas 27.7k1111 gold badges107107 silver badges118118 bronze badges, joaofbsmjoaofbsm 52533 silver badges1111 bronze badges, and caffreydcaffreyd 83311 gold badge1515 silver badges2323 bronze badges. “Filter Df When Values Matches Part of a String in Pyspark.” Stack Overflow, October 1, 1965. https://stackoverflow.com/questions/41889974/filter-df-when-values-matches-part-of-a-string-in-pyspark.

13. “Get Day, Week, Month, Year and Quarter from Date in Pyspark.” DataScience Made Simple, December 23, 2020. https://www.datasciencemadesimple.com/get-month-year-and-quarter-from-date-in-pyspark/.

14. “HAVING - InterSystems SQL Reference - InterSystems IRIS Data Platform 2020.4.” InterSystems. Accessed May 17, 2021. https://docs.intersystems.com/irislatest/csp/docbook/Doc.View.cls?KEY=RSQL_having.

15. JenksJenks 1, santonsanton 3, Hugo ReyesHugo Reyes 1, FrankFrank 59577 silver badges55 bronze badges, ManriqueManrique 1, Santosh kumar MandaSantosh kumar Manda 29866 silver badges88 bronze badges, and Vishwajeet PolVishwajeet Pol 3155 bronze badges. “Convert Pyspark String to Date Format.” Stack Overflow, March 1, 1965. https://stackoverflow.com/questions/38080748/convert-pyspark-string-to-date-format.

16. Nat, NatNat 111 silver badge44 bronze badges, and gawgaw 1. “Positional Argument Error When Using 'When' in Pyspark.” Stack Overflow, May 1, 1967. https://stackoverflow.com/questions/52420831/positional-argument-error-when-using-when-in-pyspark/52421459.

17. Nguyen, Dan. “Using LIKE, IN, BETWEEN, and Wildcards to Match Multiple Values in SQL.” Public Affairs Data Journalism at Stanford University. Accessed May 17, 2021. http://2015.padjo.org/tutorials/sql-basics/fuzzy-matching-like-in-where/.

18. Nnk. “PySpark Replace Column Values in DataFrame - SparkByExamples.” Spark by {Examples}, April 19, 2021. https://sparkbyexamples.com/pyspark/pyspark-replace-column-values/.

19. Nnk. “PySpark to_date() - Convert String to Date Format - SparkByExamples.” Spark by {Examples}, February 24, 2021. https://sparkbyexamples.com/pyspark/pyspark-to_date-convert-string-to-date-format/.

20. Nnk. “PySpark Where Filter Function: Multiple Conditions - SparkByExamples.” Spark by {Examples}, April 1, 2021. https://sparkbyexamples.com/pyspark/pyspark-where-filter/.

21. Nnk. “Spark Replace NULL Values on DataFrame - SparkByExamples.” Spark by {Examples}, November 30, 2020. https://sparkbyexamples.com/spark/spark-how-to-replace-null-values/#:~:text=In%20Spark%2C%20fill()%20function,or%20any%20constant%20literal%20values.

22. “Notebook on Nbviewer.” Jupyter Notebook Viewer. Accessed May 17, 2021. https://nbviewer.jupyter.org/github/julioasotodv/spark-df-profiling/blob/master/examples/Demo.ipynb.

23. “Notebooks.” Notebooks | Databricks on AWS. Accessed May 17, 2021. https://docs.databricks.com/notebooks/index.html#export-a-notebook.

24. Razamh. “NYC Parking Tickets: An Exploratory Analysis.” Kaggle. Kaggle, February 19, 2021. https://www.kaggle.com/razamh/nyc-parking-tickets-an-exploratory-analysis.

25. “Removing White Spaces From Data in Spark.” Analyticshut, September 13, 2020. https://analyticshut.com/removing-white-spaces-in-spark/.

26. says:, Lana, Kris Wenzel says: Gray says: Sana says: and David Says: “How to Filter Query Results Using WHERE.” Essential SQL, May 1, 2020. https://www.essentialsql.com/get-ready-to-learn-sqlserver-how-to-filter-your-query-results/.

27. sowmiyasowmiya 271010 bronze badges, and ndriccandricca 33033 silver badges1111 bronze badges. “ERROR: Typeerror When() Missing 1 Required Positional Argument 'Value' in PySpark.” Stack Overflow, October 1, 1968. https://stackoverflow.com/questions/60163587/error-typeerror-when-missing-1-required-positional-argument-value-in-pyspar.

28. “Spark SQL - SQL Queries On Dataframes.” CloudxLab. Accessed May 17, 2021. https://cloudxlab.com/assessment/displayslide/607/spark-sql-sql-queries-on-dataframes.

29. “Spark SQL Query.” Modern Data Integration for DataOps. Accessed May 17, 2021. https://streamsets.com/documentation/transformer/latest/help/transformer/Processors/SparkSQLQuery.html.

30. Spark-Examples. “Spark-Examples/Pyspark-Examples.” GitHub. Accessed May 17, 2021.

https://github.com/spark-examples/pyspark-examples/blob/master/pyspark-drop-column.py.

31. SQL IN Operator. Accessed May 17, 2021. https://www.w3schools.com/sql/sql_in.asp.

32. SQL SELECT Statement. Accessed May 17, 2021. https://www.w3schools.com/sql/sql_select.asp.

33. "Visualizations." Visualizations | Databricks on AWS. Accessed May 17, 2021. https://docs.databricks.com/notebooks/visualizations/index.html#display-function-1.

34. "Where Can I Find a Tutorial for Creating Dashboards with Databricks?" Where can I find a tutorial for creating dashboards with Databricks? - Databricks Community Forum. Accessed May 17, 2021. https://forums.databricks.com/questions/931/where-can-i-find-a-tutorial-for-creating-dashboard.html.

35. XallZallXallZall 21111 gold badge22 silver badges55 bronze badges, and Mike WilliamsonMike Williamson 4. "Missing 1 Required Positional Argument." Stack Overflow, March 1, 1963. https://stackoverflow.com/questions/24896613/missing-1-required-positional-argument.

36. Yi DuYi Du 19511 silver badge1111 bronze badges, ShuShu 22.6k22 gold badges1616 silver badges3737 bronze badges, blackbishopblackbishop 15.5k66 gold badges4343 silver badges5959 bronze badges, and ashkanashkan 1955 bronze badges. “How to Create Date from Year, Month and Day in PySpark?” Stack Overflow, December 1, 1968. https://stackoverflow.com/questions/60954146/how-to-create-date-from-year-month-and-day-in-pyspark.

37. York, City of New. “NYC Parking Tickets.” Kaggle, May 10, 2020. https://www.kaggle.com/new-york-city/nyc-parking-tickets.

38. ———. “PySpark When Otherwise: SQL Case When Usage - SparkByExamples.” Spark by {Examples}, March 31, 2021. https://sparkbyexamples.com/pyspark/pyspark-when-otherwise/.

39. ———. “Spark regexp_replace() - Replace String Value - SparkByExamples.” Spark by {Examples}, April 29, 2021. https://sparkbyexamples.com/spark/spark-regexp_replace-replace-string-value/.