BDAD PROJECT REPORT

Project on : Crime in NYC(How Safe NYC really is?)

Submitted By : Yashasvi Choukikar

NetID: yc4953

## Data Ingestion

### DATASET 1:

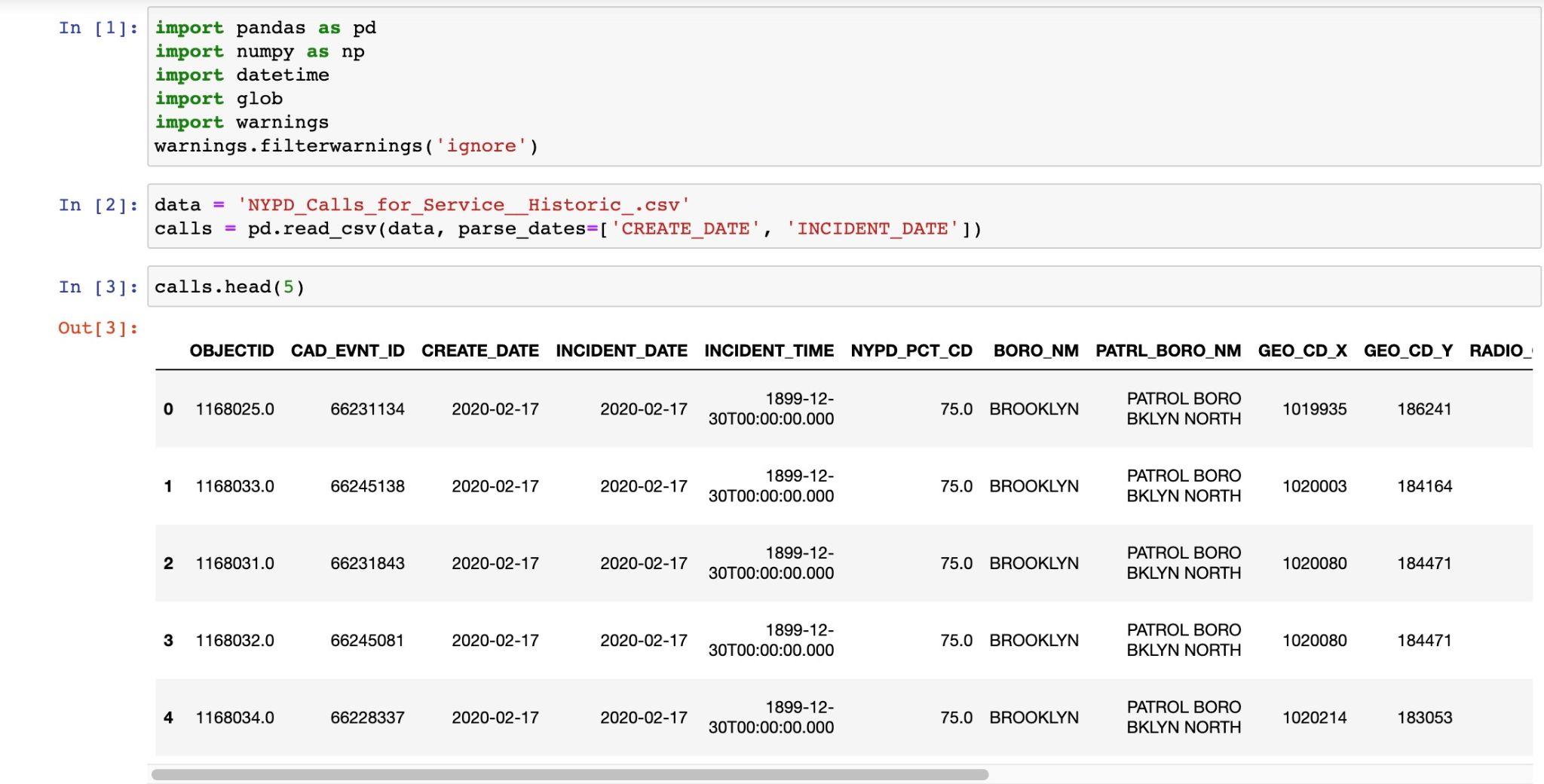
Source: <https://data.cityofnewyork.us/Public-Safety/NYPD-Calls-for-Service-Historic-/d6zx-ckhd>

File size : 6.8 GB

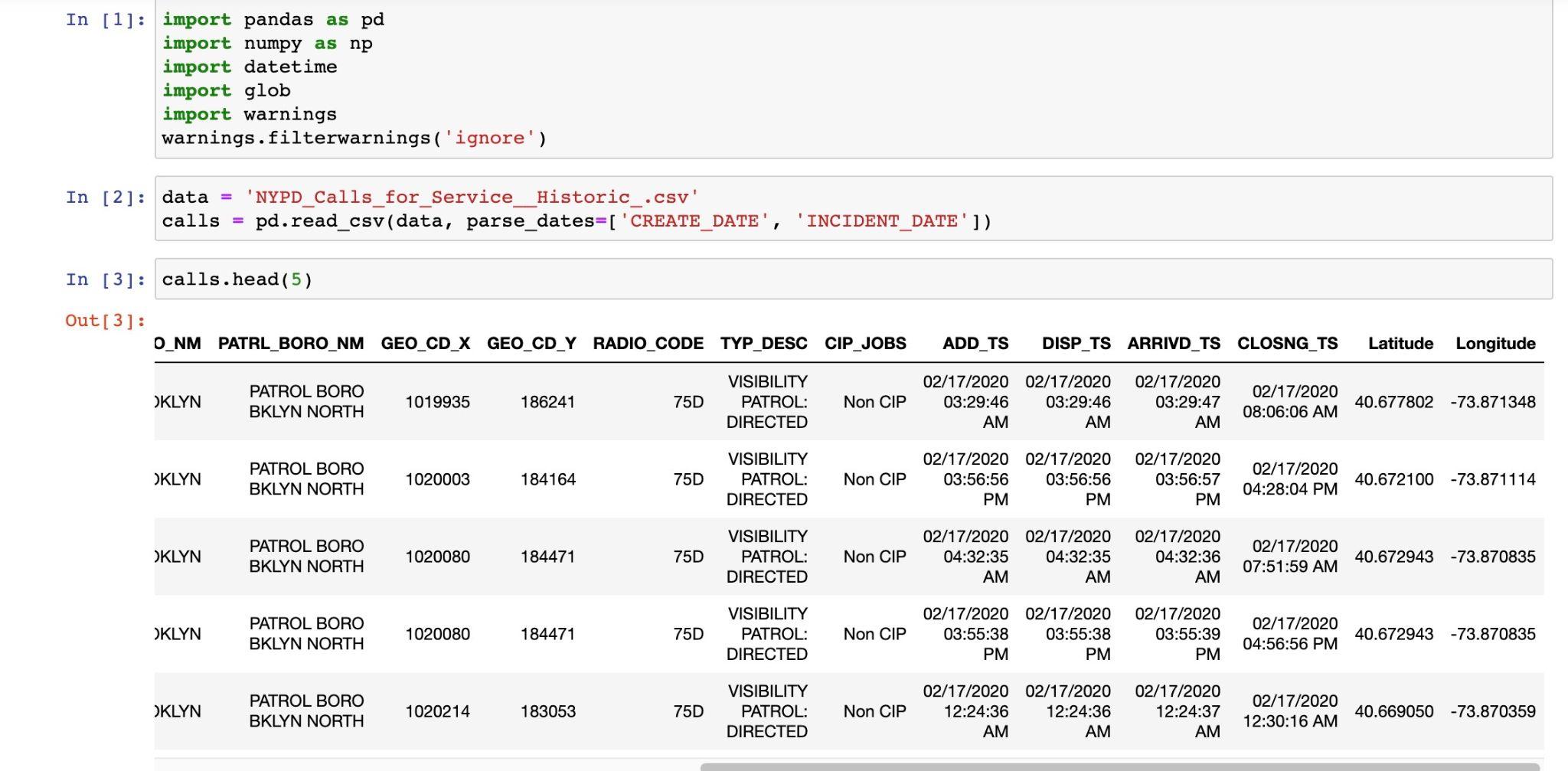
Description : The NYPD is the owner of the dataset below. To communicate with callers and the NYPD, phone takers and dispatchers use this information. A system entry is represented by each record. Both entries created by the general public and ones started by NYPD employees are included in the data. The information can be used for problems that the NYPD is addressing. It was released to the public on 5/3/2021 and contains historical data.

Snippet of Dataset:

Since the file size is huge, have used pandas to show the dataset(only for display purposes)



More columns:



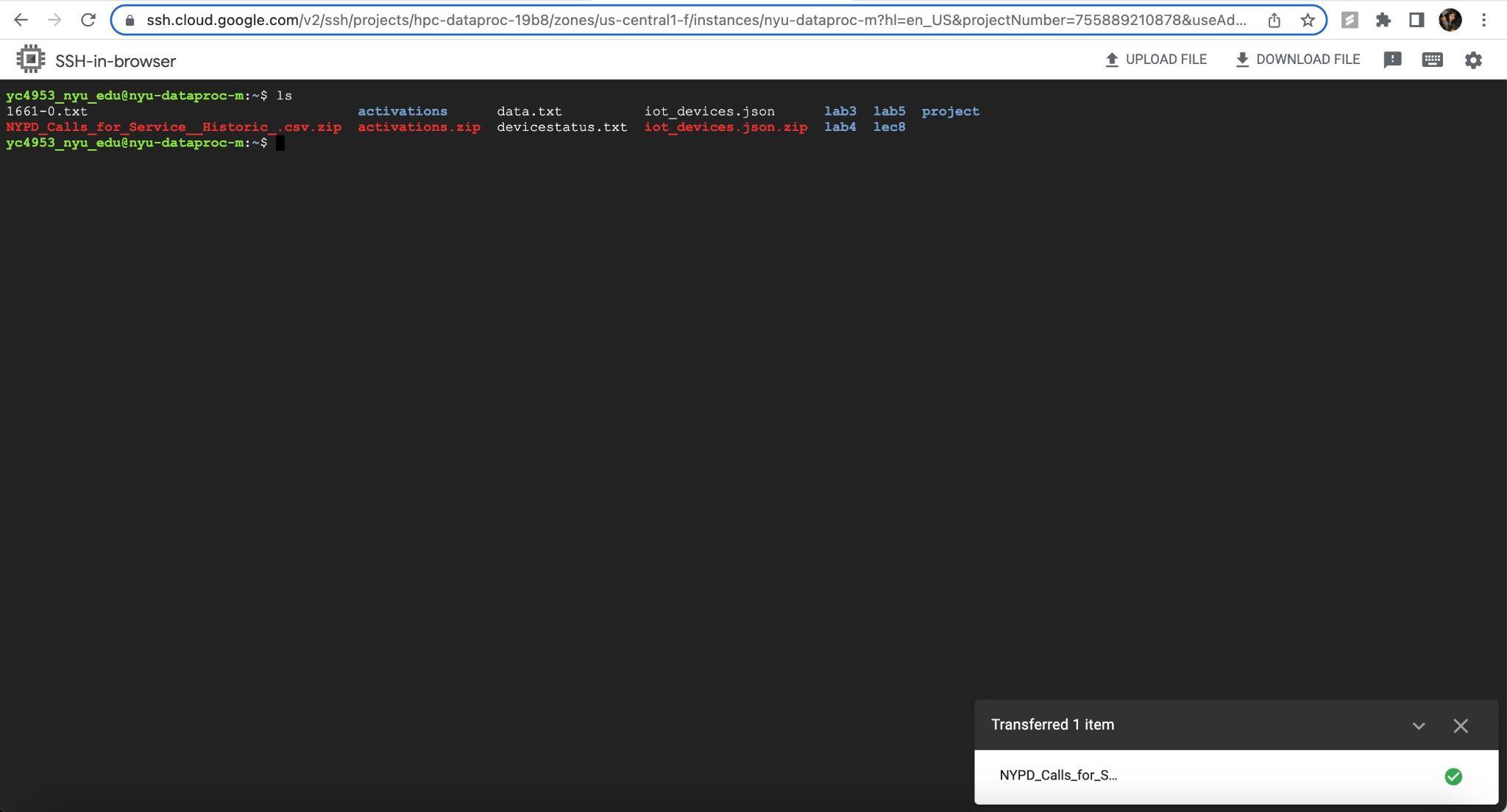
Column Information:

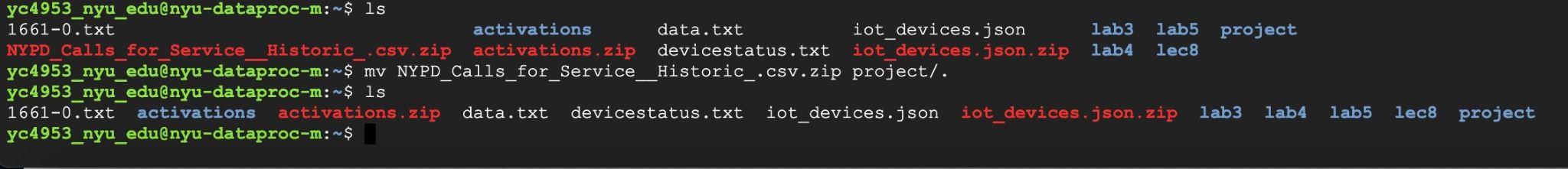
As described in : <https://data.cityofnewyork.us/Public-Safety/NYPD-Calls-for-Service-Historic-/d6zx-ckhd>

| **Column Name** | **Column Description** | **Remarks** |
| --- | --- | --- |
| OBJECTID | Row identifier for each call |  |
| CAD\_EVNT\_ID | Unique identifier generated by the the ICAD 911 system |  |
| CREATE\_DATE | Date of call |  |
| INCIDENT\_DATE | Date of incident | Calls can come in for issues that occurred in the past |
| INCIDENT\_TIME | Time of incident |  |
| NYPD\_PCT\_CD | NYPD precinct call is in |  |
| BORO\_NM | Borough call is in |  |
| PATRL\_BORO\_NM | NYPD patrol Borough call is in |  |
| GEO\_CD\_X | The X-Coordinate of the midblock of the street segment where the violation was issued | New York State Plane Coordinate system, Long Island Zone, units feet |
| GEO\_CD\_Y | The Y-Coordinate of the midblock of the street segment where the violation was issued | New York State Plane Coordinate system, Long Island Zone, units feet |
| RADIO\_CODE | NYPD code used to inform NYPD member of service the nature of the call |  |
| TYP\_DESC | Description based on RADIO\_CODE |  |
| CIP\_JOBS | Flag indicating if the call relates to a Crime In Progress (CIP) |  |
| ADD\_TS | Timestamp of when the call was added to the system |  |
| DISP\_TS | Timestamp of when the call was dispatched to a responding unit |  |
| ARRIVD\_TS | Timestamp of when the responding unit arrived on the scene | Not all calls will have an arrival time |
| CLOSNG\_TS | Timestamp of when the call was marked closed |  |
| Latitude | The Latitude of the midblock of the street segment where the violation was issued | WGS84, decimal degrees |
| Longitude | The Longitude of the midblock of the street segment where the violation was issued | WGS84, decimal degrees |

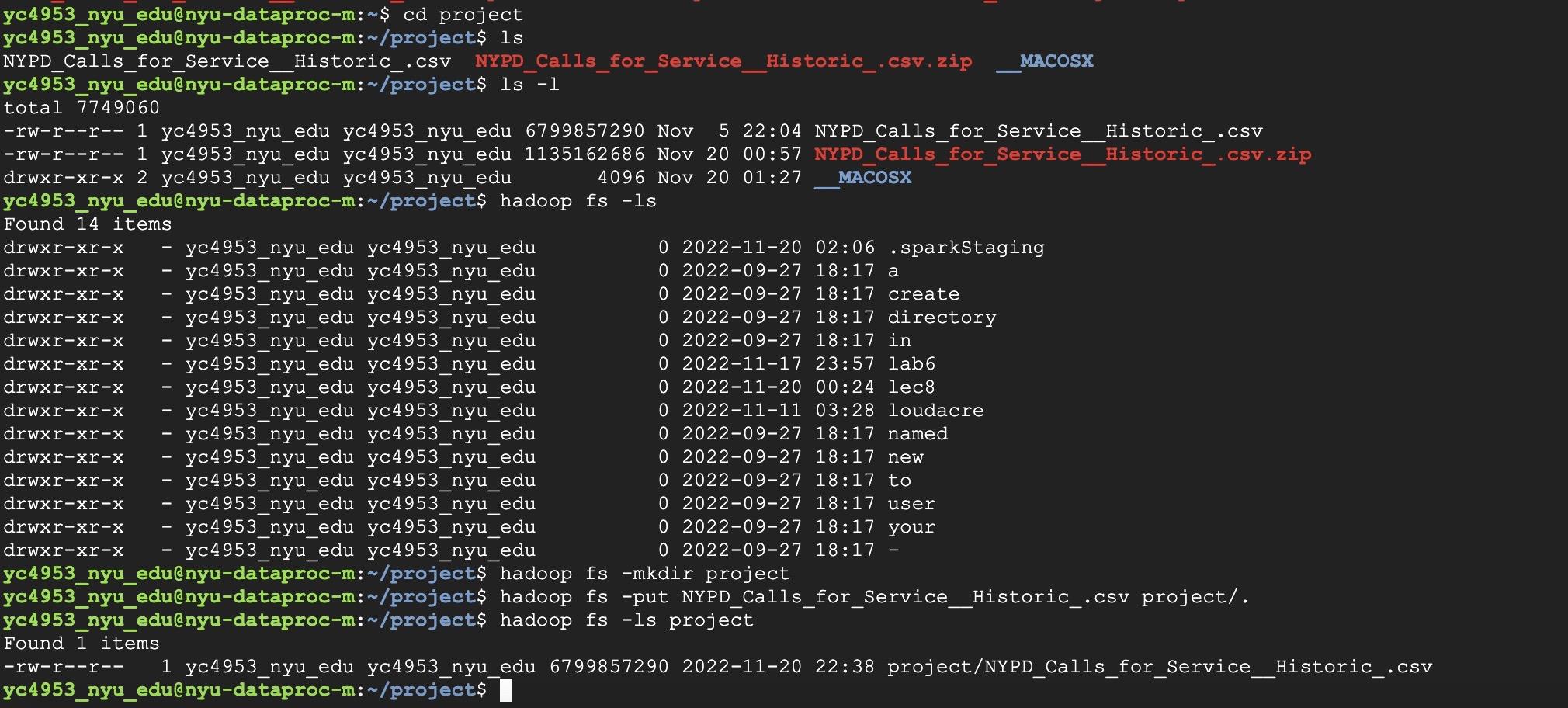
Loading the dataset in data proc cluster:

Since the file size is huge, I have zipped the file and uploaded the zipped file in dataproc cluster.









Step 1: Checking the columns and schema

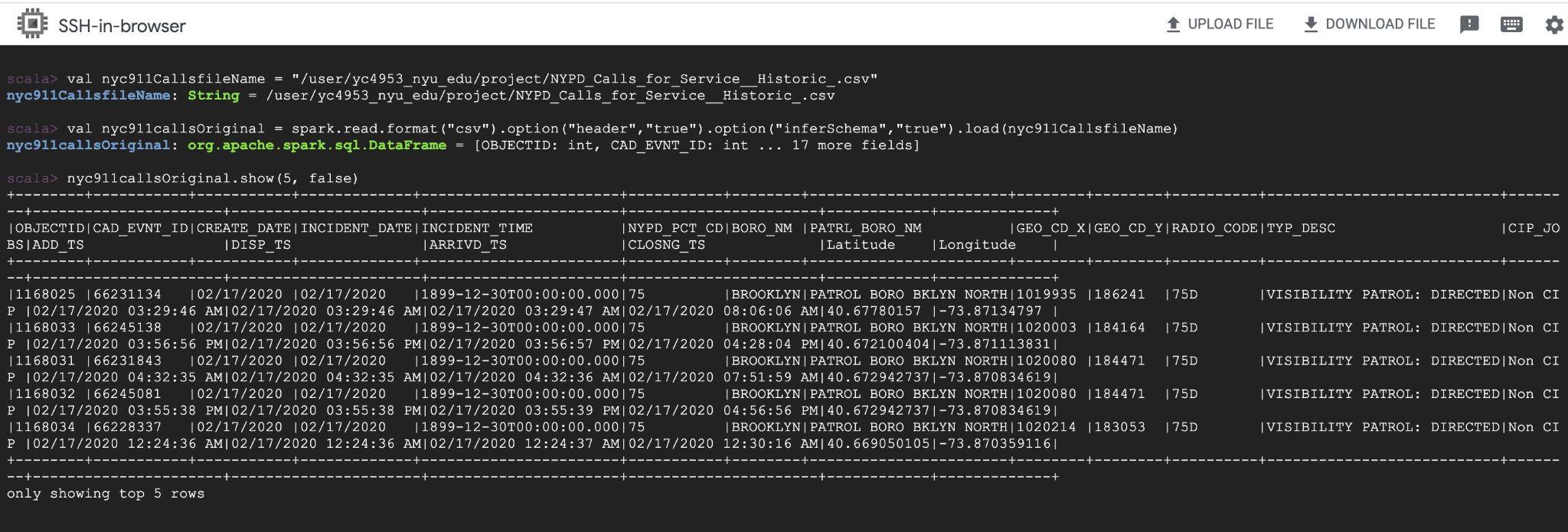
val nyc911CallsfileName = "/user/yc4953\_nyu\_edu/project/NYPD\_Calls\_for\_Service\_\_Historic\_.csv"

val nyc911callsOriginal = spark.read.format("csv").option("header","true").option("inferSchema","true").load(nyc911CallsfileName)

nyc911callsOriginal.show(5, false)

nyc911callsOriginal.printSchema

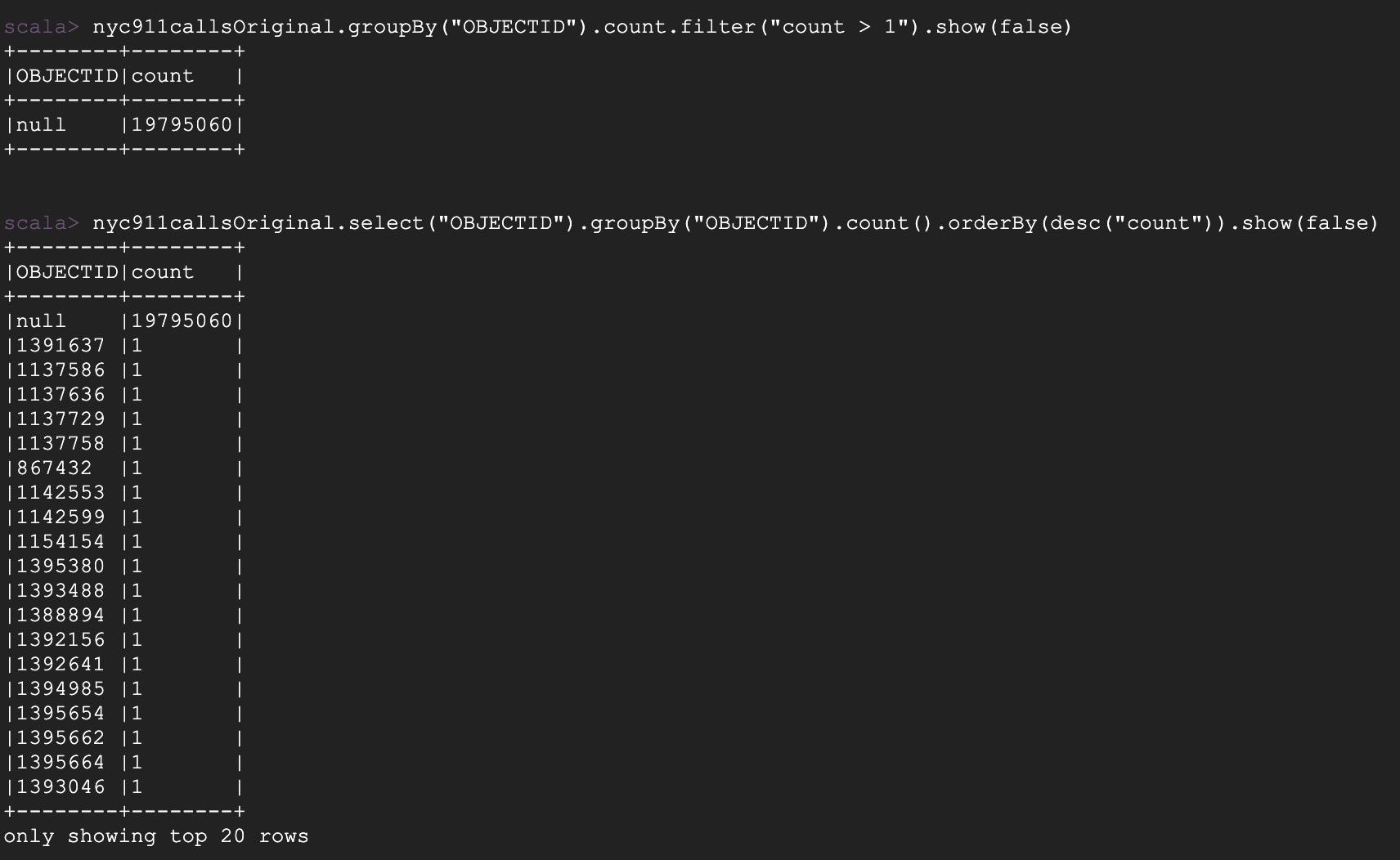
nyc911callsOriginal.columns.foreach(println)



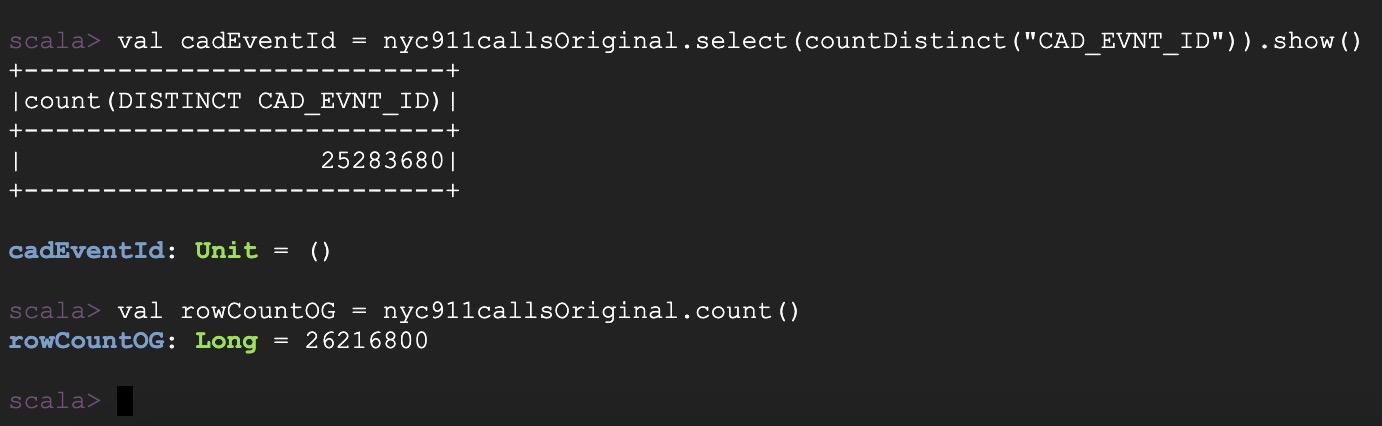
Step 2: Choosing columns

For first column, OBJECTID, I observe there are significant null values as shown by following queries:

* nyc911callsOriginal.groupBy("OBJECTID").count.filter("count > 1").show(false)
* nyc911callsOriginal.select("OBJECTID").groupBy("OBJECTID").count().orderBy(desc("count")).show(false)



Hence, we can drop column OBJECTID as I can still identify the row with other column names, more importantly with CAD\_EVNT\_ID. - **(A)**

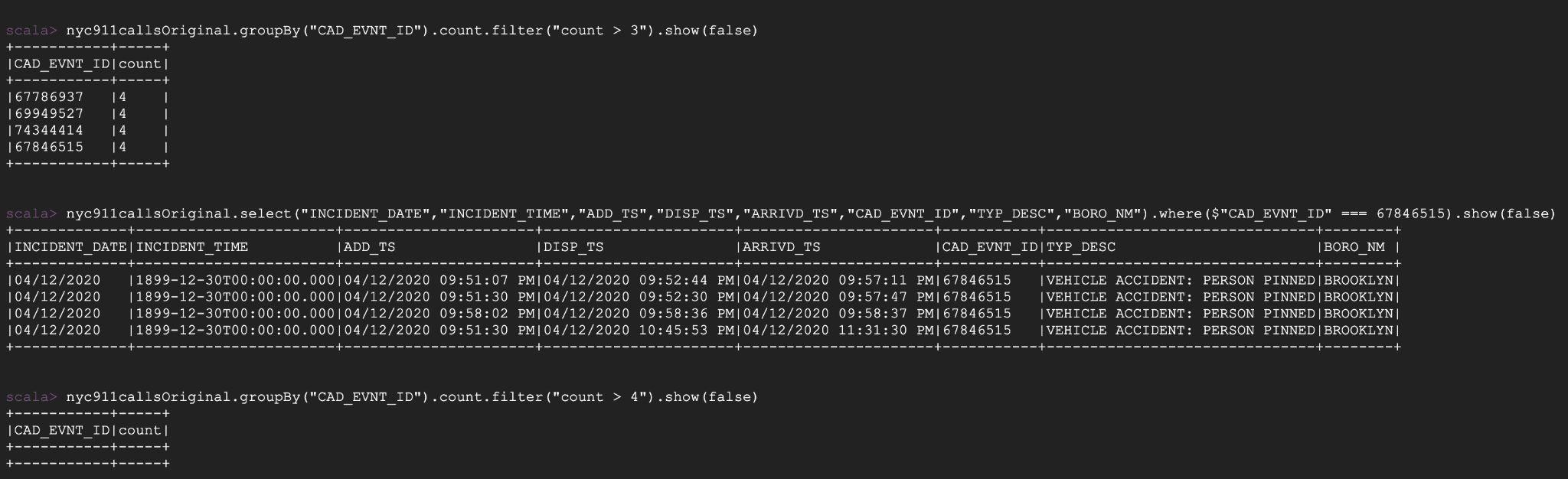
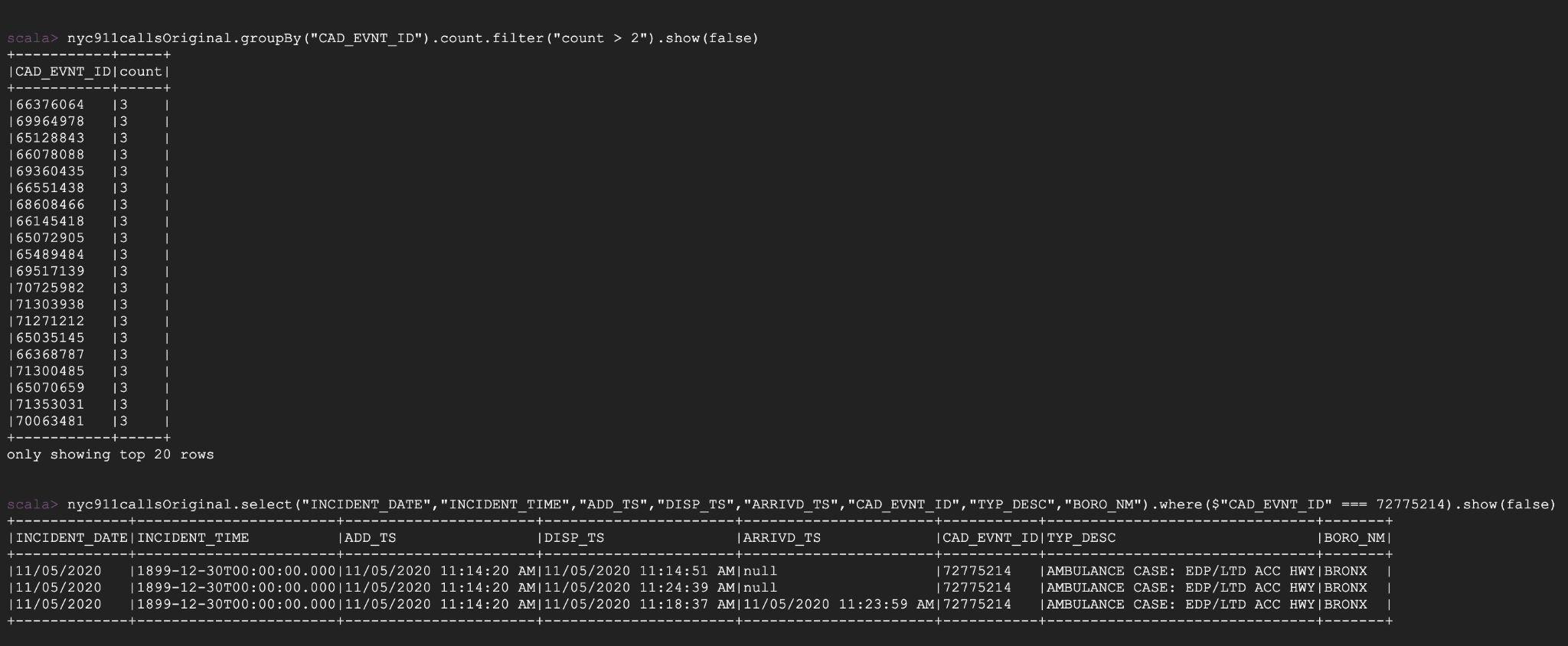
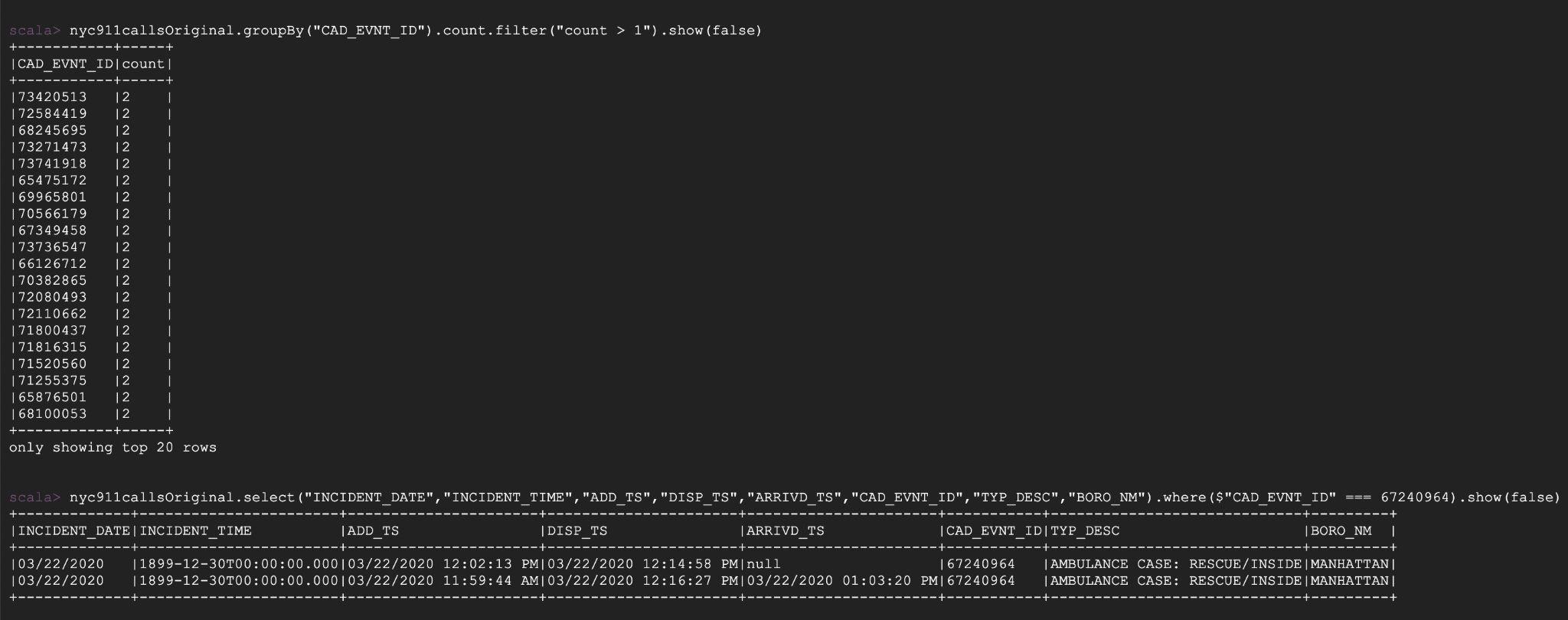
I proceed for column CAD\_EVNT\_ID which had interesting insights:  
As shown below, the count of distinct numbers of CAD\_EVNT\_ID is less than the total number of rows.

Further analysis shows the following:

* val cadEventId = nyc911callsOriginal.select(countDistinct("CAD\_EVNT\_ID")).show()
* nyc911callsOriginal.groupBy("CAD\_EVNT\_ID").count.filter("count > 1").show(false)
* nyc911callsOriginal.select("INCIDENT\_DATE","INCIDENT\_TIME","ADD\_TS","DISP\_TS","ARRIVD\_TS","CAD\_EVNT\_ID","TYP\_DESC","BORO\_NM").where($"CAD\_EVNT\_ID" === 67240964).show(false)
* nyc911callsOriginal.groupBy("CAD\_EVNT\_ID").count.filter("count > 2").show(false)
* nyc911callsOriginal.select("INCIDENT\_DATE","INCIDENT\_TIME","ADD\_TS","DISP\_TS","ARRIVD\_TS","CAD\_EVNT\_ID","TYP\_DESC","BORO\_NM").where($"CAD\_EVNT\_ID" === 72775214).show(false)
* nyc911callsOriginal.groupBy("CAD\_EVNT\_ID").count.filter("count > 3").show(false)
* nyc911callsOriginal.select("INCIDENT\_DATE","INCIDENT\_TIME","ADD\_TS","DISP\_TS","ARRIVD\_TS","CAD\_EVNT\_ID","TYP\_DESC","BORO\_NM").where($"CAD\_EVNT\_ID" === 67846515).show(false)
* nyc911callsOriginal.groupBy("CAD\_EVNT\_ID").count.filter("count > 4").show(false)

Note : I have taken the “CAD\_EVNT\_ID” value randomly from the display of count

The CAD\_EVNT\_ID passed in where clause has been taken randomly from the list displayed from the groupBy query. It has been observed that till count = 3, CAD\_EVNT\_ID duplicated.

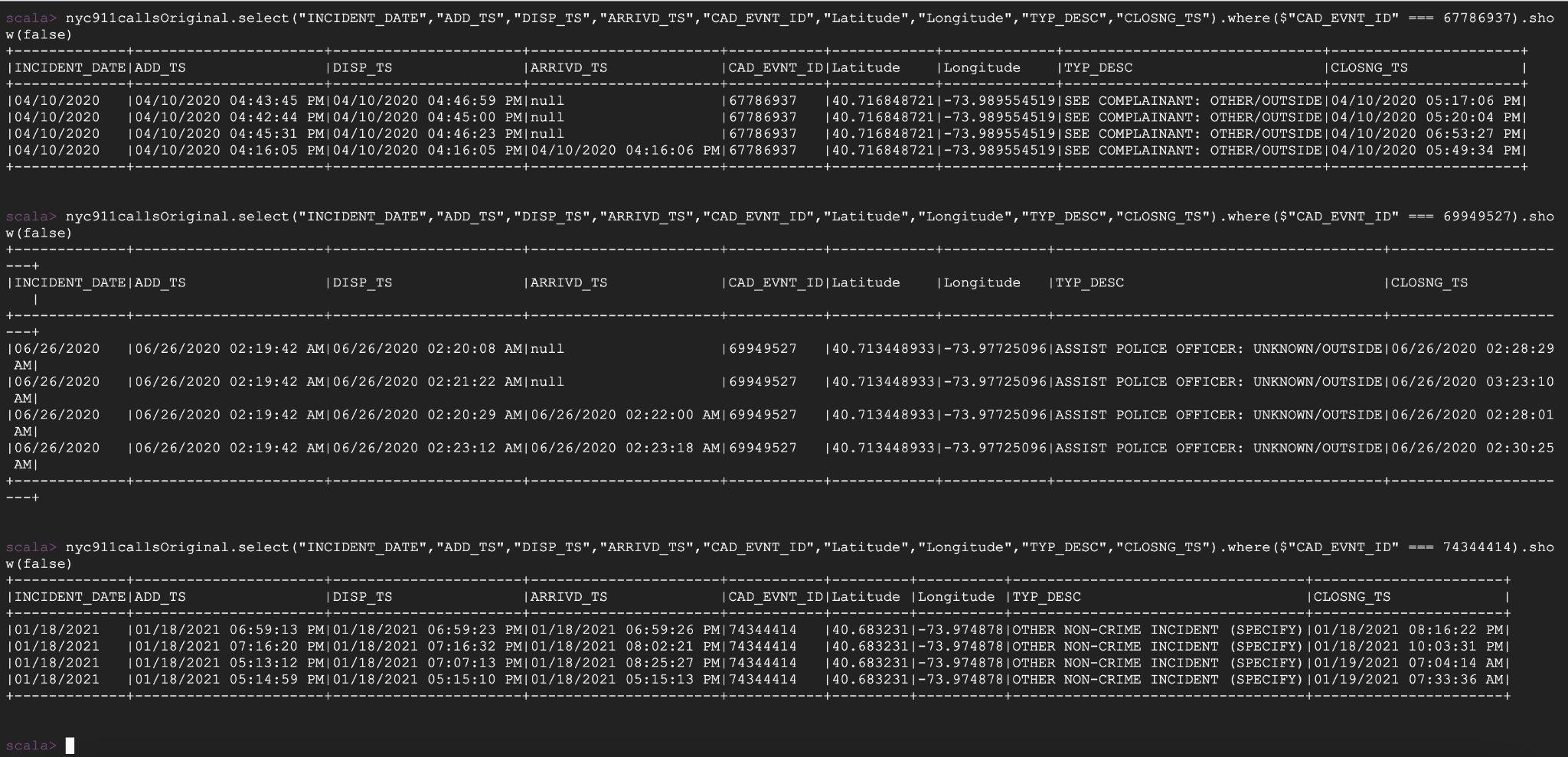


On having a closer look at other columns, I observe that where few instances exist where the arrival time was null(it is a valid scenario as described in the column metadata), the dispatch and closing time stamps vary as shown below:

(The values CAD\_EVNT\_ID are taken for case where count > 3)

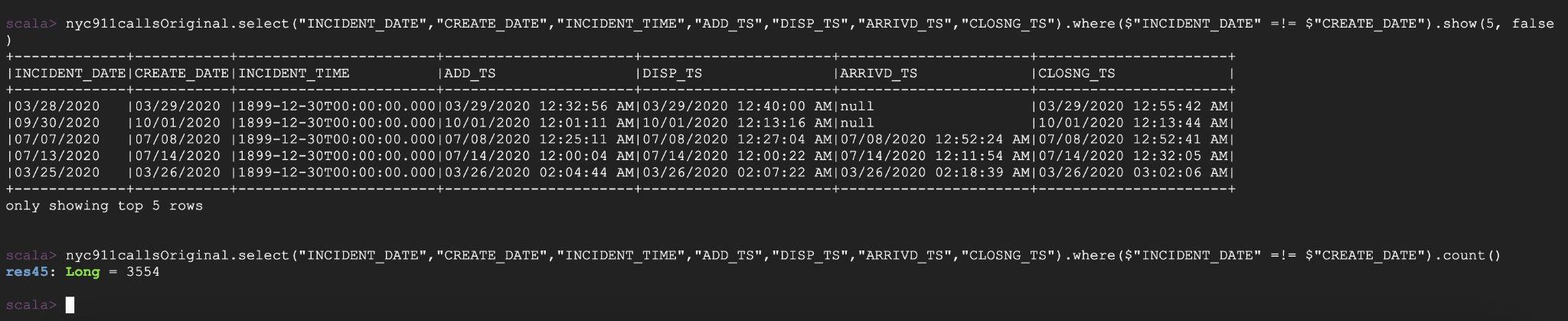
* nyc911callsOriginal.select("INCIDENT\_DATE","ADD\_TS","DISP\_TS","ARRIVD\_TS","CAD\_EVNT\_ID","Latitude","Longitude","TYP\_DESC","CLOSNG\_TS").where($"CAD\_EVNT\_ID" === 67786937).show(false)
* nyc911callsOriginal.select("INCIDENT\_DATE","ADD\_TS","DISP\_TS","ARRIVD\_TS","CAD\_EVNT\_ID","Latitude","Longitude","TYP\_DESC","CLOSNG\_TS").where($"CAD\_EVNT\_ID" === 69949527).show(false)
* nyc911callsOriginal.select("INCIDENT\_DATE","ADD\_TS","DISP\_TS","ARRIVD\_TS","CAD\_EVNT\_ID","Latitude","Longitude","TYP\_DESC","CLOSNG\_TS").where($"CAD\_EVNT\_ID" === 74344414).show(false)

The latitudes and longitude remain same which means there could be multiple calls made from the same location by multiple folks implying it's a public area. Hence have kept it as is. This insight shall be useful to obtain any pattern in public areas if possible. Hence shall keep this column. - **(B)**



CREATE\_DATE and INCIDENT\_DATE columns are both important. To check for redundancy, queries the following:

* nyc911callsOriginal.select("INCIDENT\_DATE","CREATE\_DATE","INCIDENT\_TIME","ADD\_TS","DISP\_TS","ARRIVD\_TS","CLOSNG\_TS").where($"INCIDENT\_DATE" =!= $"CREATE\_DATE").show(5, false)
* nyc911callsOriginal.select("INCIDENT\_DATE","CREATE\_DATE","INCIDENT\_TIME","ADD\_TS","DISP\_TS","ARRIVD\_TS","CLOSNG\_TS").where($"INCIDENT\_DATE" =!= $"CREATE\_DATE").count()



This is a valid scenario where the call may have been made after the incident so shall keep both columns. - **(C )**

The following columns provide police precinct information and hence shall be kept. - **(D)**

NYPD\_PCT\_CD

BORO\_NM

PATRL\_BORO\_NM

RADIO\_CODE

TYP\_DESC

CIP\_JOBS

The following columns are not needed as I can identify the location with latitude and longitude. GEO\_CD\_X , GEO\_CD\_Y - **(E)**

The following columns detail the time stamps of the calls , hence are important to keep - **(F)**

INCIDENT\_TIME

ADD\_TS

DISP\_TS

ARRIVD\_TS

CLOSNG\_TS

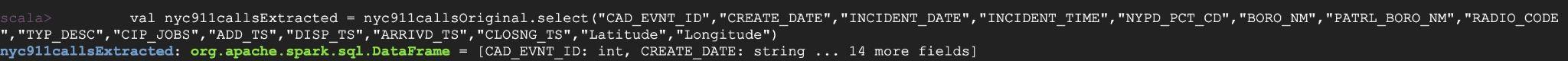
The following columns denote the geographical location hence are important to keep. - **(G)**

Latitude, Longitude

Thus, from A, B, C, D, E, F, G, the table summarizes which columns to keep:

| OBJECTID |  |
| --- | --- |
| CAD\_EVNT\_ID | ✓ |
| CREATE\_DATE | ✓ |
| INCIDENT\_DATE | ✓ |
| INCIDENT\_TIME | ✓ |
| NYPD\_PCT\_CD | ✓ |
| BORO\_NM | ✓ |
| PATRL\_BORO\_NM | ✓ |
| GEO\_CD\_X |  |
| GEO\_CD\_Y |  |
| RADIO\_CODE | ✓ |
| TYP\_DESC | ✓ |
| CIP\_JOBS | ✓ |
| ADD\_TS | ✓ |
| DISP\_TS | ✓ |
| ARRIVD\_TS | ✓ |
| CLOSNG\_TS | ✓ |
| Latitude | ✓ |
| Longitude | ✓ |

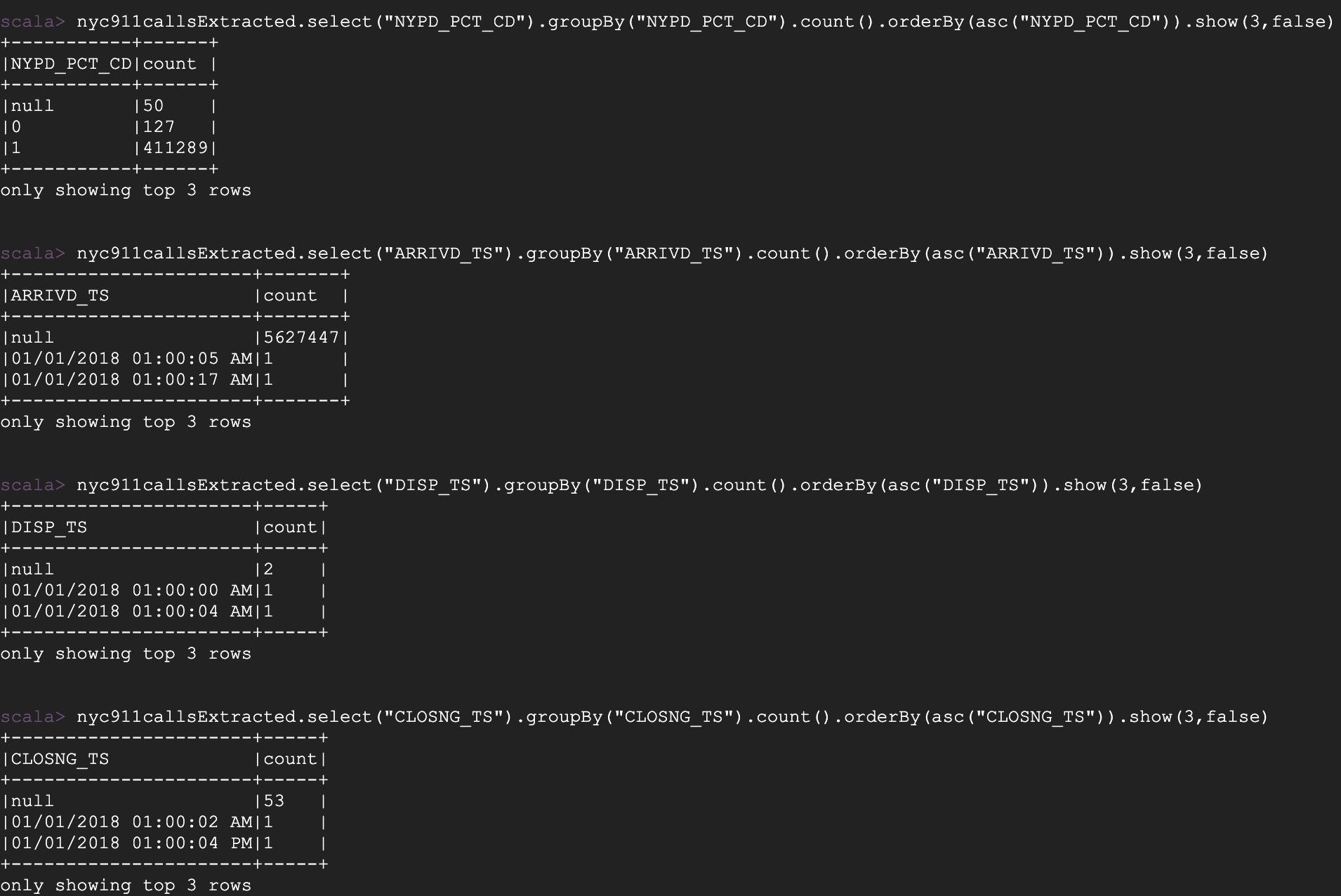
val nyc911callsExtracted = nyc911callsOriginal.select("CAD\_EVNT\_ID","CREATE\_DATE","INCIDENT\_DATE","INCIDENT\_TIME","NYPD\_PCT\_CD","BORO\_NM","PATRL\_BORO\_NM","RADIO\_CODE","TYP\_DESC","CIP\_JOBS","ADD\_TS","DISP\_TS","ARRIVD\_TS","CLOSNG\_TS","Latitude","Longitude")



Step 3: Drop null values from select columns

For the following columns, I can drop null values with na.drop function:

* nyc911callsExtracted.select("NYPD\_PCT\_CD").groupBy("NYPD\_PCT\_CD").count().orderBy(asc("NYPD\_PCT\_CD")).show(3,false)
* nyc911callsExtracted.select("ARRIVD\_TS").groupBy("ARRIVD\_TS").count().orderBy(asc("ARRIVD\_TS")).show(3,false)
* nyc911callsExtracted.select("DISP\_TS").groupBy("DISP\_TS").count().orderBy(asc("DISP\_TS")).show(3,false)
* nyc911callsExtracted.select("CLOSNG\_TS").groupBy("CLOSNG\_TS").count().orderBy(asc("CLOSNG\_TS")).show(3,false)



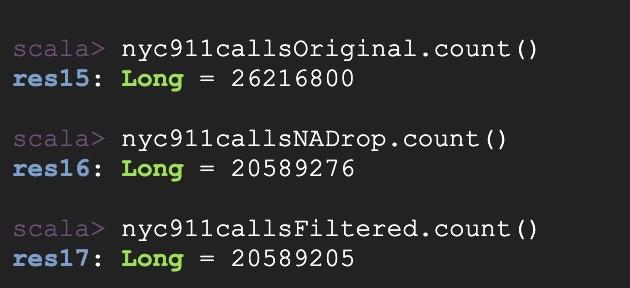
For the following columns , (null) is seen

* nyc911callsExtracted.select("TYP\_DESC").groupBy("TYP\_DESC").count().orderBy(asc("TYP\_DESC")).show(3,false)
* nyc911callsExtracted.select("BORO\_NM").groupBy("BORO\_NM").count().orderBy(asc("BORO\_NM")).show(3,false)
* nyc911callsExtracted.select("PATRL\_BORO\_NM").groupBy("PATRL\_BORO\_NM").count().orderBy(asc("PATRL\_BORO\_NM")).show(3,false)



val nyc911callsNADrop = nyc911callsExtracted.na.drop(Seq("ARRIVD\_TS","NYPD\_PCT\_CD","DISP\_TS","CLOSNG\_TS"))

val nyc911callsFiltered = nyc911callsNADrop.filter(!($"BORO\_NM".equalTo("(null)"))).filter(!($"PATRL\_BORO\_NM".equalTo("(null)"))).filter(!($"TYP\_DESC".equalTo("(null)")))



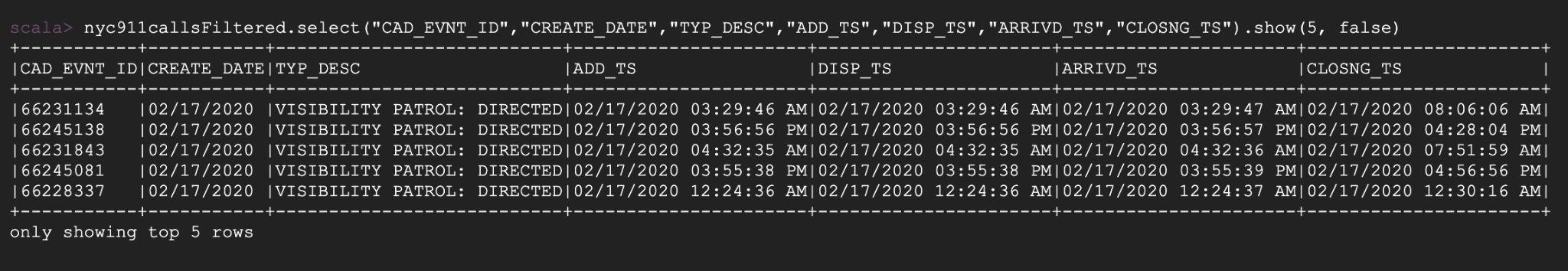
As seen above, originalData had 26216800 rows , NAdrop data has 20589276 rows, the filteredData has 20589205 rows, which implies 5627595 rows were dropped.

Step 4: Transforming data in few columns

The time stats for four columns shows both data and time in AM/PM.

Note: Showing the necessary columns

nyc911callsFiltered.select("CAD\_EVNT\_ID","CREATE\_DATE","TYP\_DESC","ADD\_TS","DISP\_TS","ARRIVD\_TS","CLOSNG\_TS").show(5, false)



To convert it into 24 hour format time used a User Defined Function (UDF) to apply to columns, "ADD\_TS","DISP\_TS","ARRIVD\_TS","CLOSNG\_TS"

import org.apache.spark.sql.functions.udf

import java.text.SimpleDateFormat

def getTs(dateStr: String): String = {

val inputStr = new SimpleDateFormat("hh:mm:ssaa")

val outputStr = new SimpleDateFormat("HH:mm:ss")

val timeStr = dateStr.split(" ")(1) + dateStr.split(" ")(2)

val timeTS = outputStr.format(inputStr.parse(timeStr))

timeTS

}

val transformTs = udf((x : String) => getTs(x))

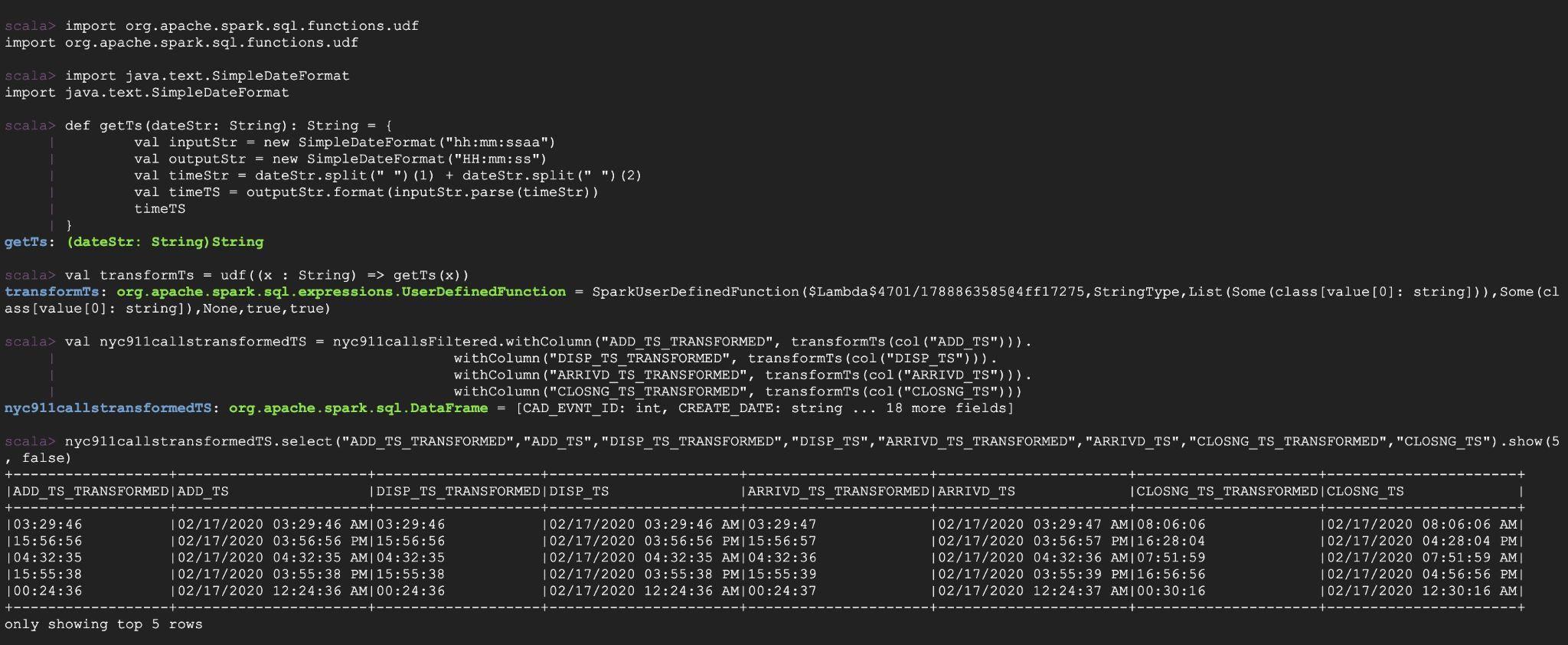
val nyc911callstransformedTS = nyc911callsFiltered.withColumn("ADD\_TS\_TRANSFORMED", transformTs(col("ADD\_TS"))).

withColumn("DISP\_TS\_TRANSFORMED", transformTs(col("DISP\_TS"))).

withColumn("ARRIVD\_TS\_TRANSFORMED", transformTs(col("ARRIVD\_TS"))).

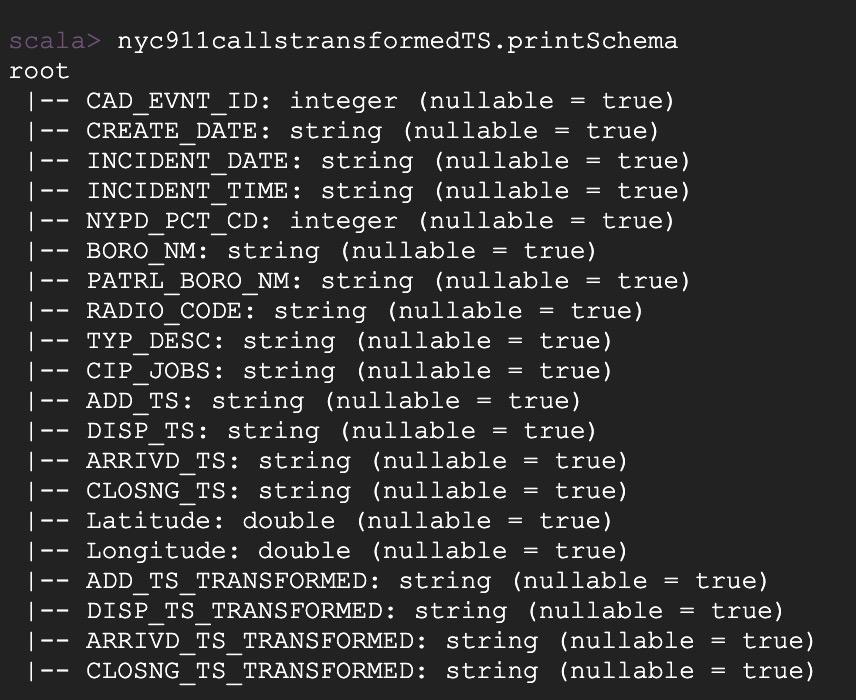
withColumn("CLOSNG\_TS\_TRANSFORMED", transformTs(col("CLOSNG\_TS")))

nyc911callstransformedTS.select("ADD\_TS\_TRANSFORMED","ADD\_TS","DISP\_TS\_TRANSFORMED","DISP\_TS","ARRIVD\_TS\_TRANSFORMED","ARRIVD\_TS","CLOSNG\_TS\_TRANSFORMED","CLOSNG\_TS").show(5, false)



The above has added additional four columns

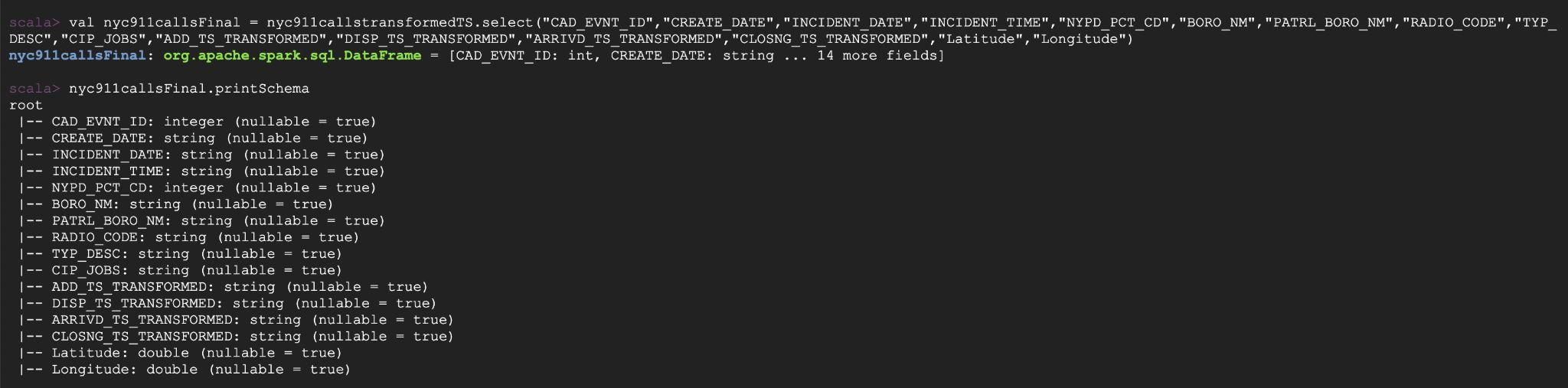
nyc911callstransformedTS.printSchema



Hence will drop the previous "ADD\_TS","DISP\_TS","ARRIVD\_TS","CLOSNG\_TS"

val nyc911callsFinal = nyc911callstransformedTS.select("CAD\_EVNT\_ID","CREATE\_DATE","INCIDENT\_DATE","INCIDENT\_TIME","NYPD\_PCT\_CD","BORO\_NM","PATRL\_BORO\_NM","RADIO\_CODE","TYP\_DESC","CIP\_JOBS","ADD\_TS\_TRANSFORMED","DISP\_TS\_TRANSFORMED","ARRIVD\_TS\_TRANSFORMED","CLOSNG\_TS\_TRANSFORMED","Latitude","Longitude")

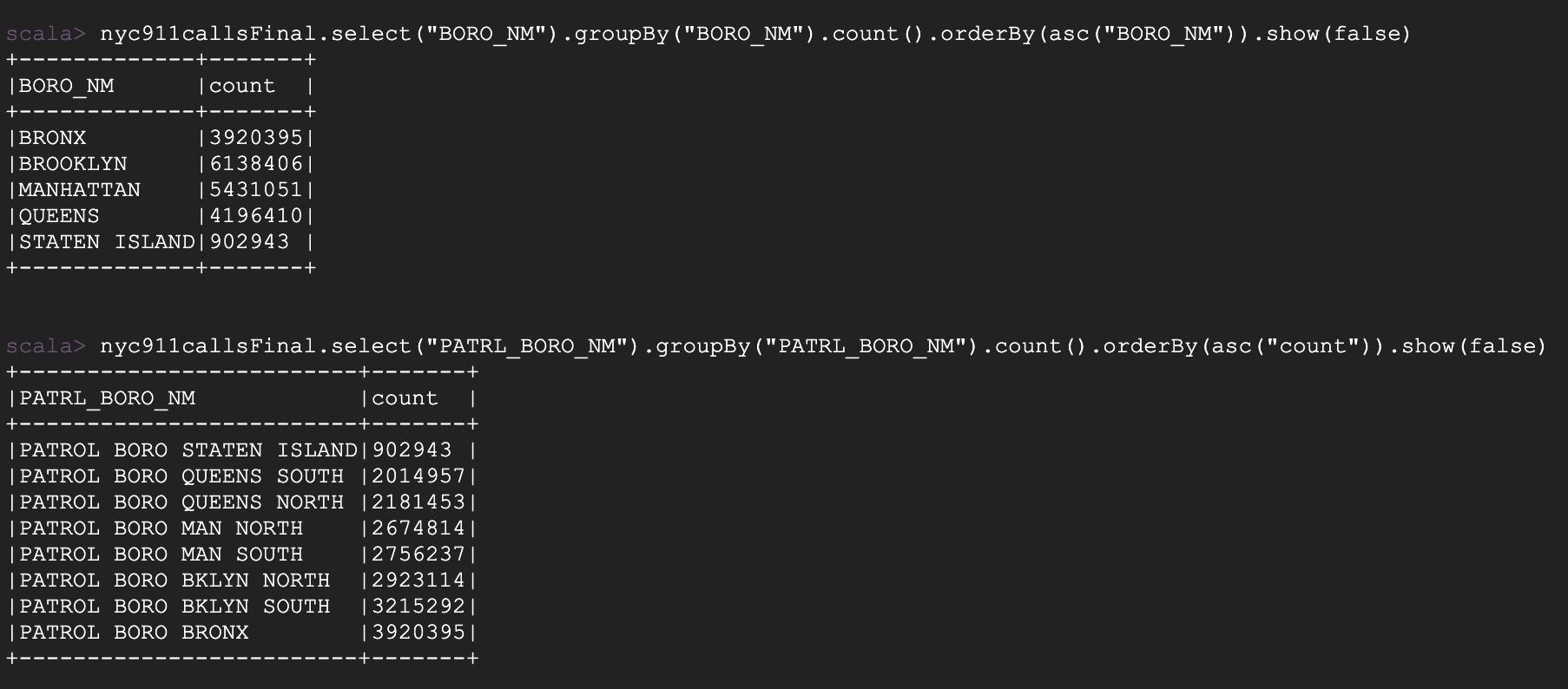
nyc911callsFinal.printSchema



Step 5: Column Statistics

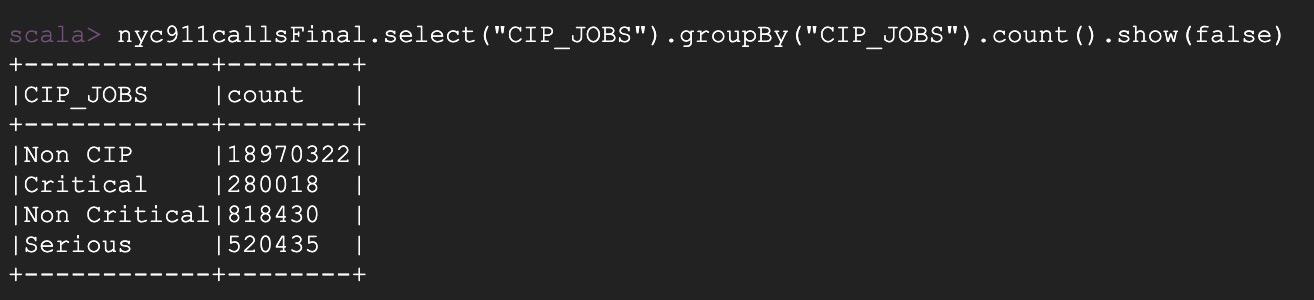
Identifying Burough Names and Patrol Names:

* nyc911callsFinal.select("BORO\_NM").groupBy("BORO\_NM").count().orderBy(asc("BORO\_NM")).show(false)
* nyc911callsFinal.select("PATRL\_BORO\_NM").groupBy("PATRL\_BORO\_NM").count().orderBy(asc("count")).show(false)



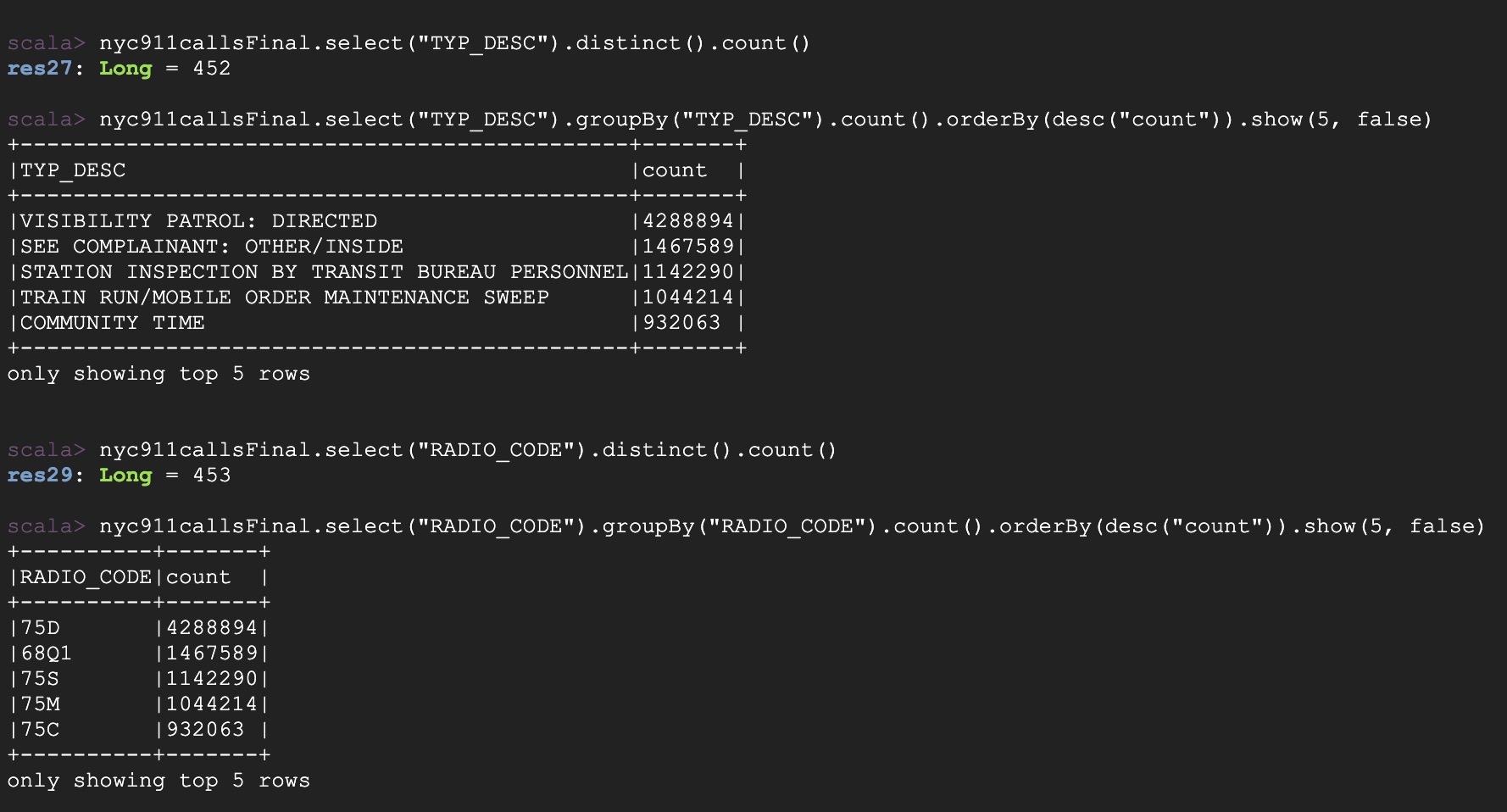
Identifying CIP Types:

nyc911callsFinal.select("CIP\_JOBS").groupBy("CIP\_JOBS").count().show(false)



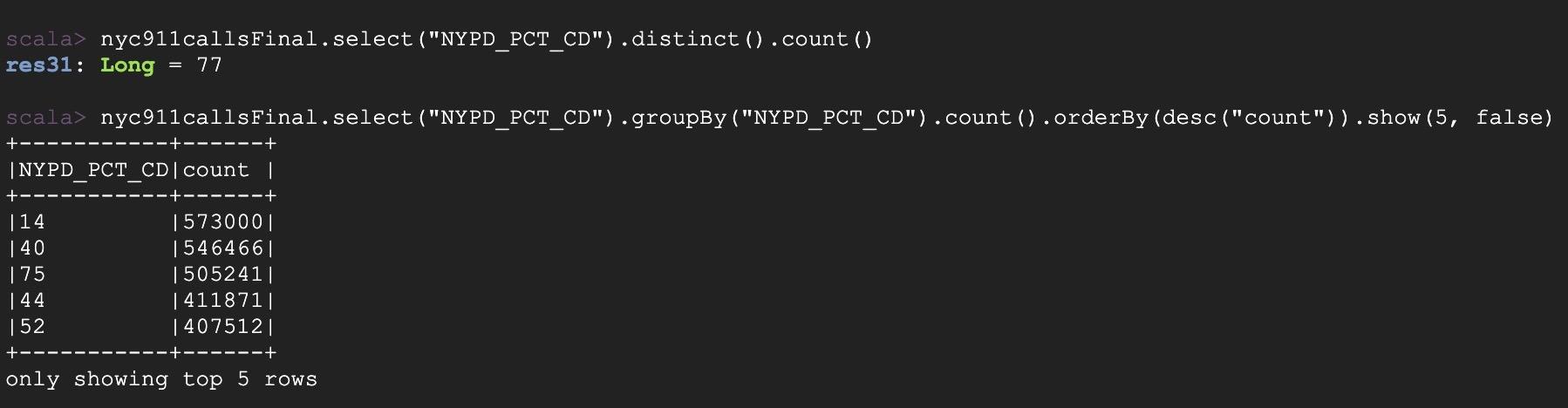
Radio Code and Type Description for Calls:

* nyc911callsFinal.select("TYP\_DESC").distinct().count()
* nyc911callsFinal.select("TYP\_DESC").groupBy("TYP\_DESC").count().orderBy(desc("count")).show(5, false)
* nyc911callsFinal.select("RADIO\_CODE").distinct().count()
* nyc911callsFinal.select("RADIO\_CODE").groupBy("RADIO\_CODE").count().orderBy(desc("count")).show(5, false)



NYPD Precinct Numbers

* nyc911callsFinal.select("NYPD\_PCT\_CD").distinct().count()
* nyc911callsFinal.select("NYPD\_PCT\_CD").groupBy("NYPD\_PCT\_CD").count().orderBy(desc("count")).show(5, false)



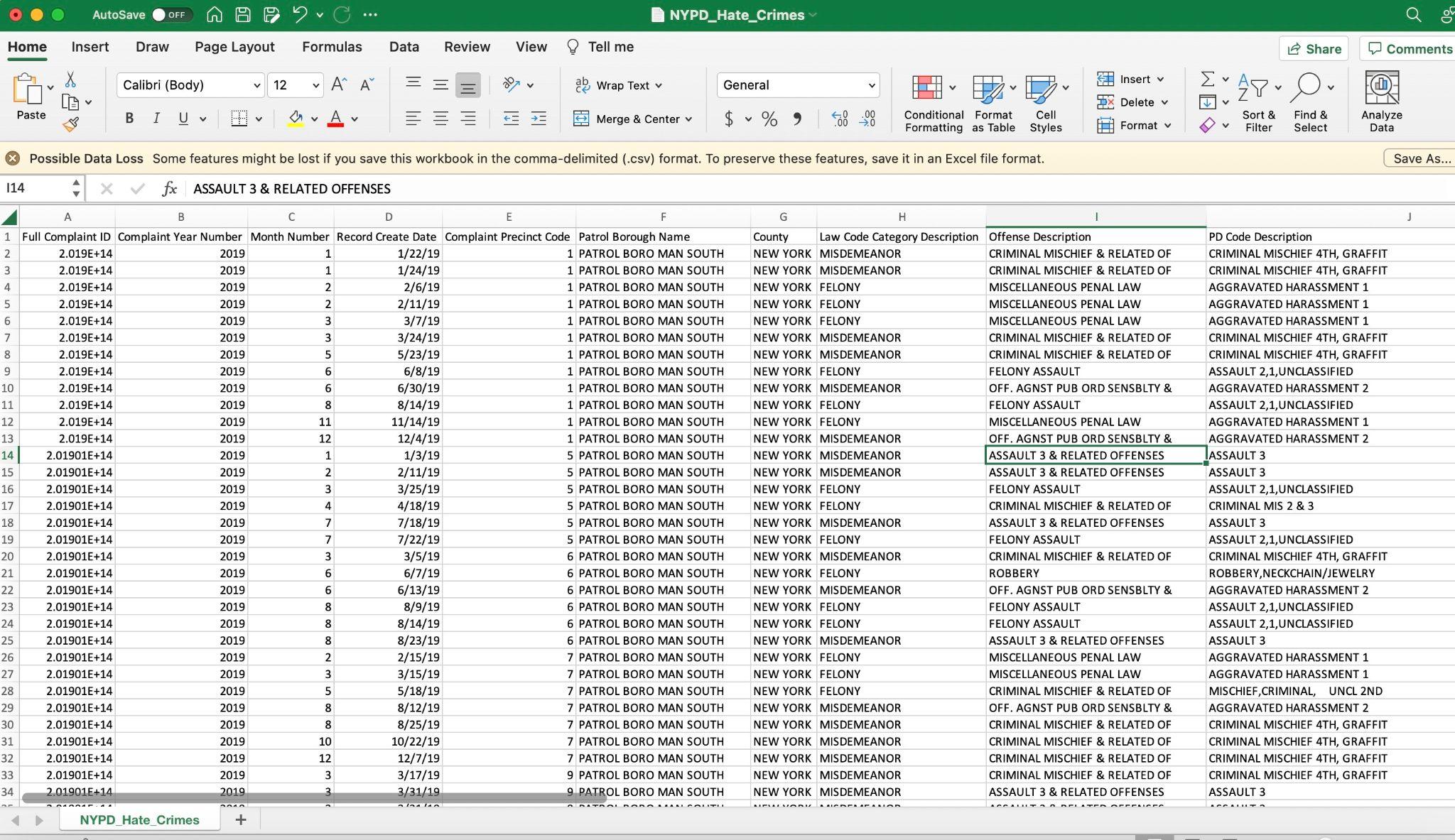
### DATASET 2:

Source: <https://data.cityofnewyork.us/Public-Safety/NYPD-Hate-Crimes/bqiq-cu78>

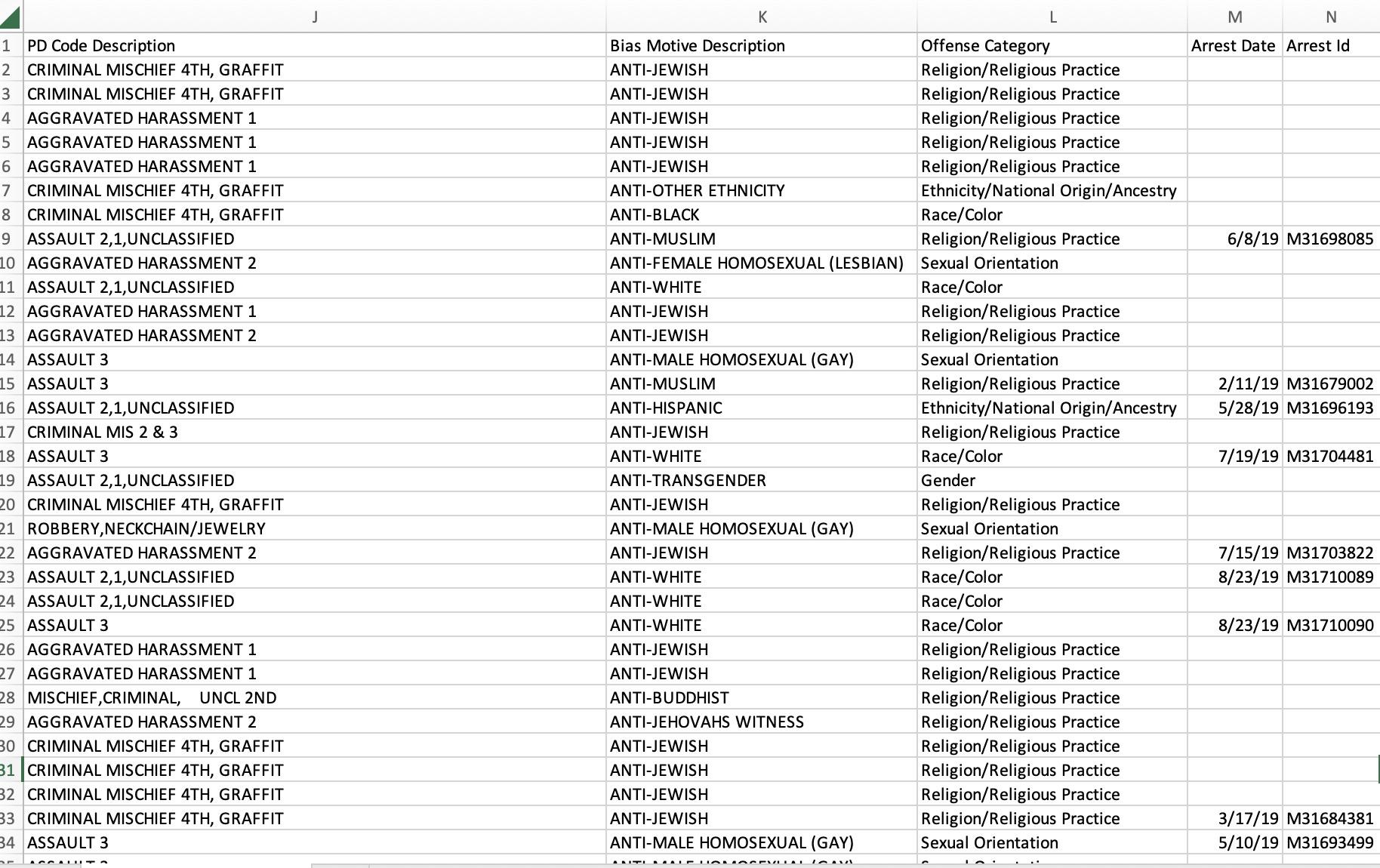
File size : 313 KB

Description : The NYPD owns the dataset that follows. Dataset of confirmed hate crimes in New York City from the years 2019-2022. This dataset comprises where the hate crime is committed and segmented into various categories of crimes like felony, aggravated assault and so on. The data is updated manually every quarter and the statistics and information is made publicly available.

Snippet of the Dataset:



More columns continued:

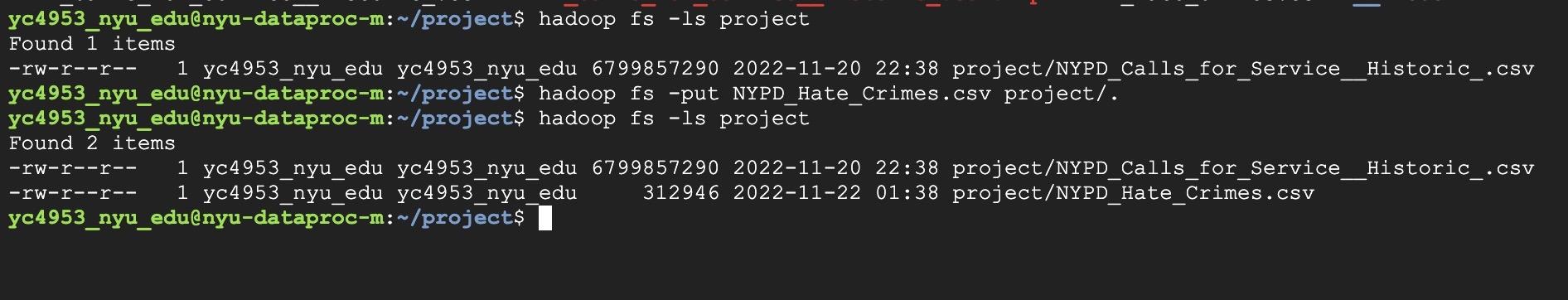
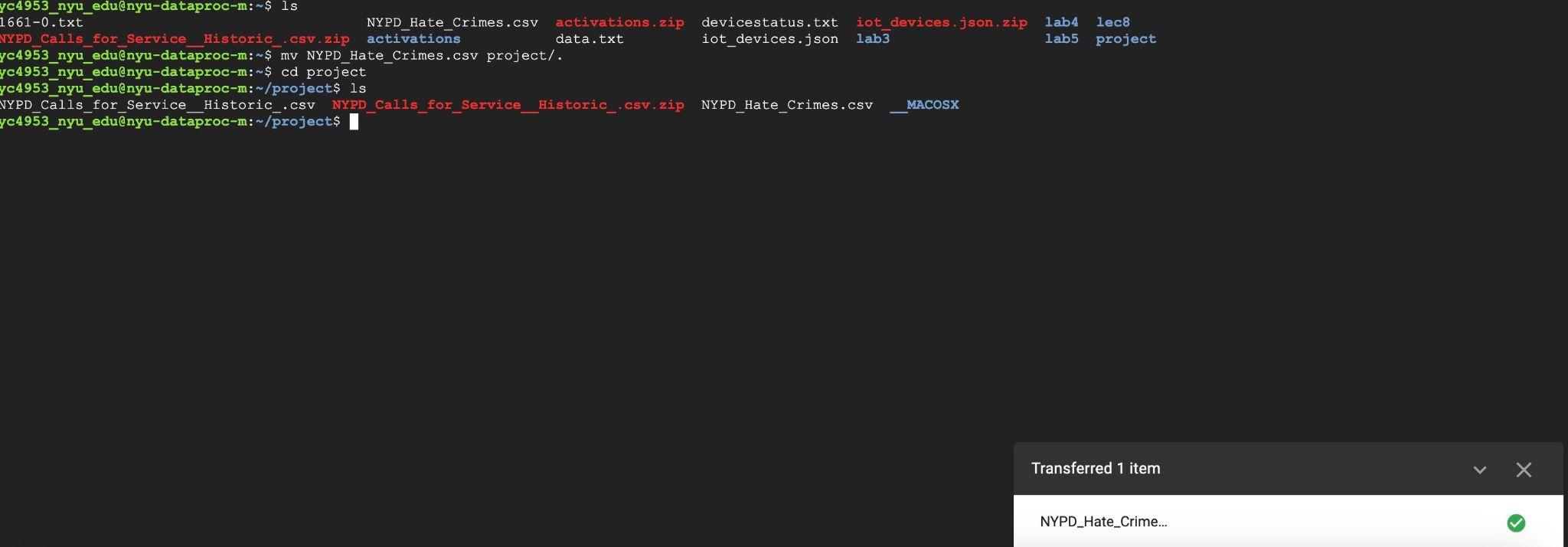


Column Information:

As described in : <https://data.cityofnewyork.us/Public-Safety/NYPD-Hate-Crimes/bqiq-cu78>

| **Column Name** | **Column Description** |
| --- | --- |
| Full Complaint ID | Identifier for the official complaint |
| Complaint Year Number | Year of incident |
| Month Number | Month of incident |
| Record Create Date | Date of incident |
| Complaint Precinct Code | NYPD precinct where incident complaint registered |
| Patrol Borough Name | NYPD Patrol Borough where incident complaint registered |
| County | County where incident took place |
| Law Code Category Description | Category of offense in law |
| Offense Description | Details of offense |
| PD Code Description | NYPD description of offense |
| Bias Motive Description | NYPD category of hate crime |
| Offense Category | Category of offense |
| Arrest Date | Date of arrest if arrest happened |
| Arrest Id | Identifier for arrest(if arrest happened) |

Loading the dataset to dataproc cluster:



Step 1: Checking the columns and schema

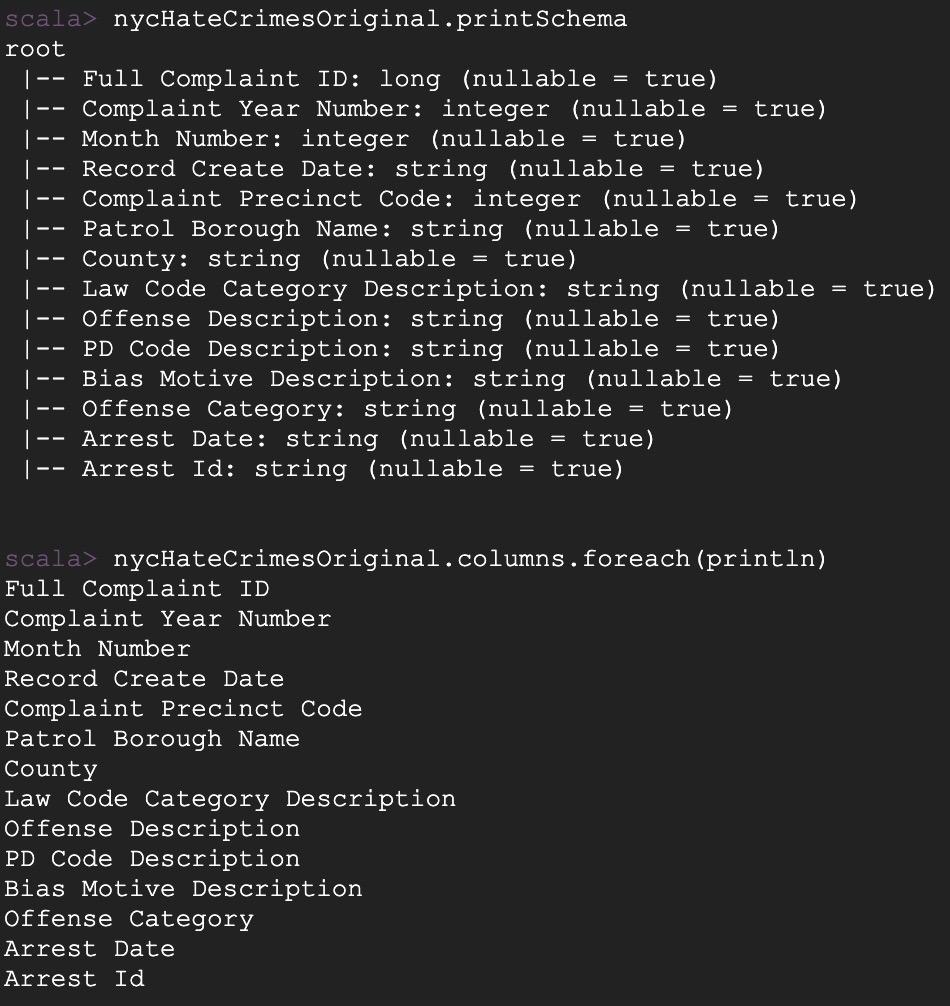
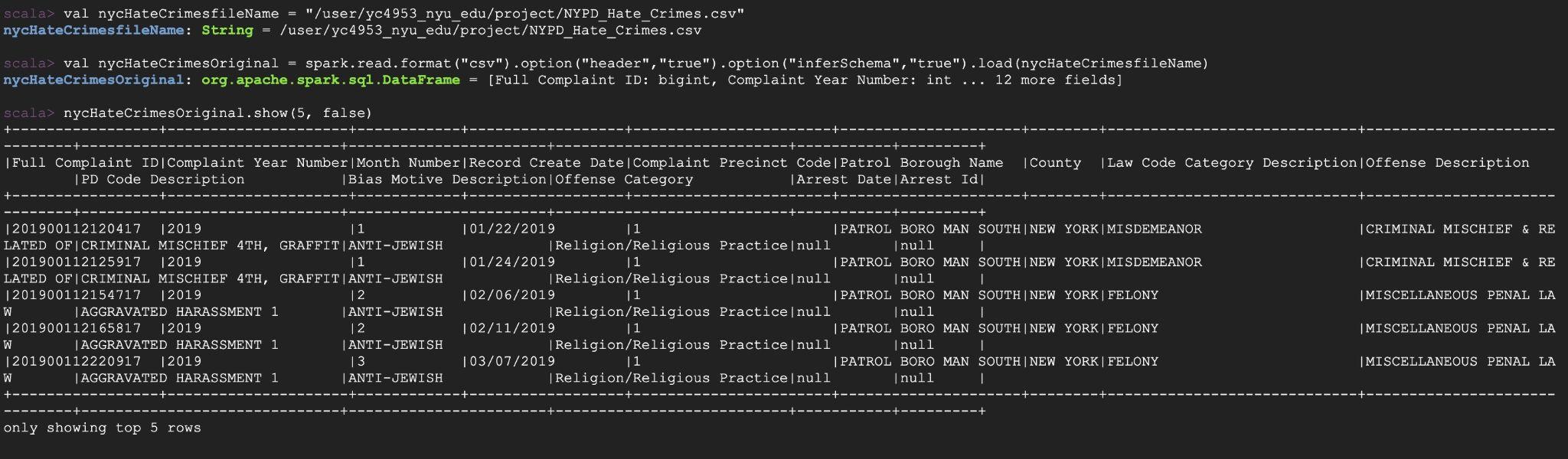
val nycHateCrimesfileName = "/user/yc4953\_nyu\_edu/project/NYPD\_Hate\_Crimes.csv"

val nycHateCrimesOriginal = spark.read.format("csv").option("header","true").option("inferSchema","true").load(nycHateCrimesfileName)

nycHateCrimesOriginal.show(5, false)

nycHateCrimesOriginal.printSchema

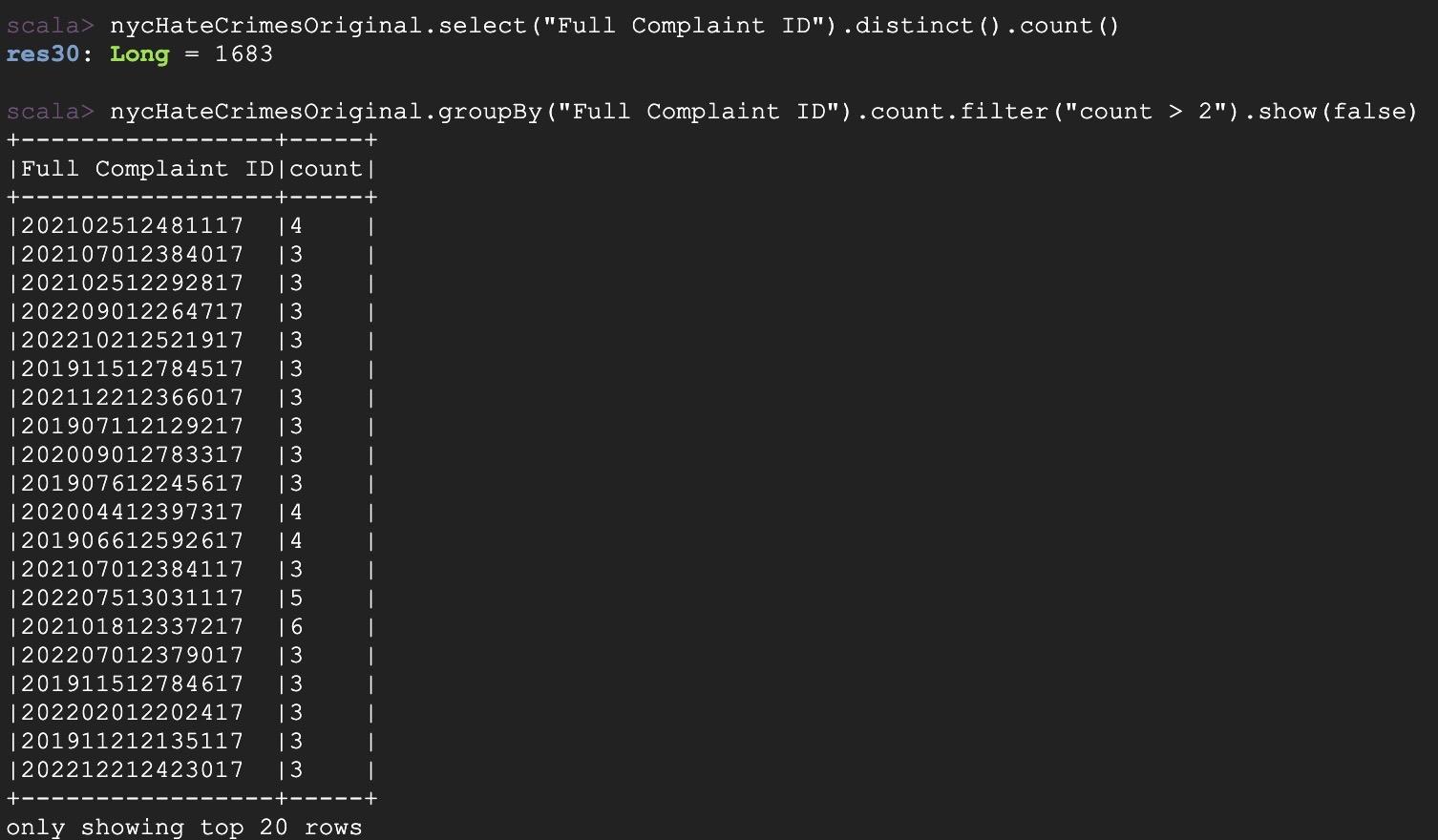
nycHateCrimesOriginal.columns.foreach(println)



Step 2: Choosing columns

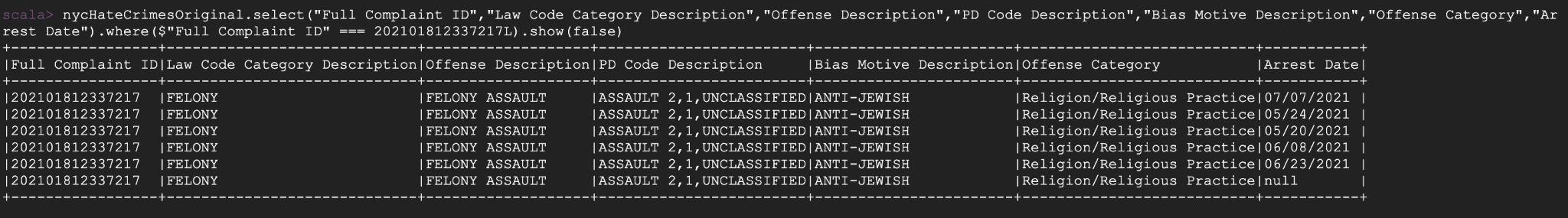
Since “Complaint Year Number” And “Month Number” can be derived from “Record Create Date”, I can drop these columns. It is sufficient to have “Arrest Date” if arrest has happened(implying these can be null values). Hence, I can also drop “Arrest Id”.

For the column “Full Complaint ID”, I observed the following:



On Checking the value for count=6,

nycHateCrimesOriginal.select("Full Complaint ID","Law Code Category Description","Offense Description","PD Code Description","Bias Motive Description","Offense Category","Arrest Date").where($"Full Complaint ID" === 202101812337217L).show(false)

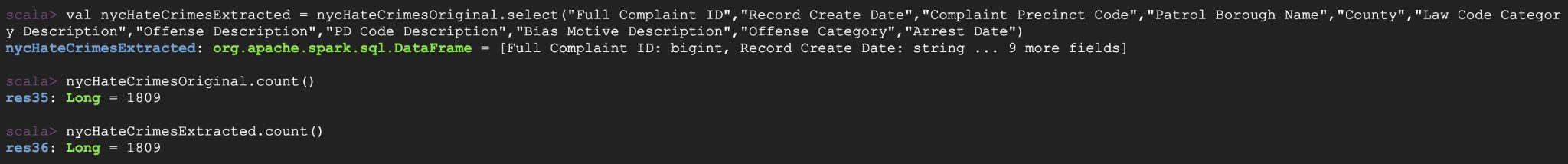


This implies that for the same “Full Complaint ID” value, there can be multiple arrests made (or none). Hence, it is better to keep the unique identifier.

Hence, from above, the following columns are kept:

| Full Complaint ID | ✓ |
| --- | --- |
| Complaint Year Number |  |
| Month Number |  |
| Record Create Date | ✓ |
| Complaint Precinct Code | ✓ |
| Patrol Borough Name | ✓ |
| County | ✓ |
| Law Code Category Description | ✓ |
| Offense Description | ✓ |
| PD Code Description | ✓ |
| Bias Motive Description | ✓ |
| Offense Category | ✓ |
| Arrest Date | ✓ |
| Arrest Id |  |

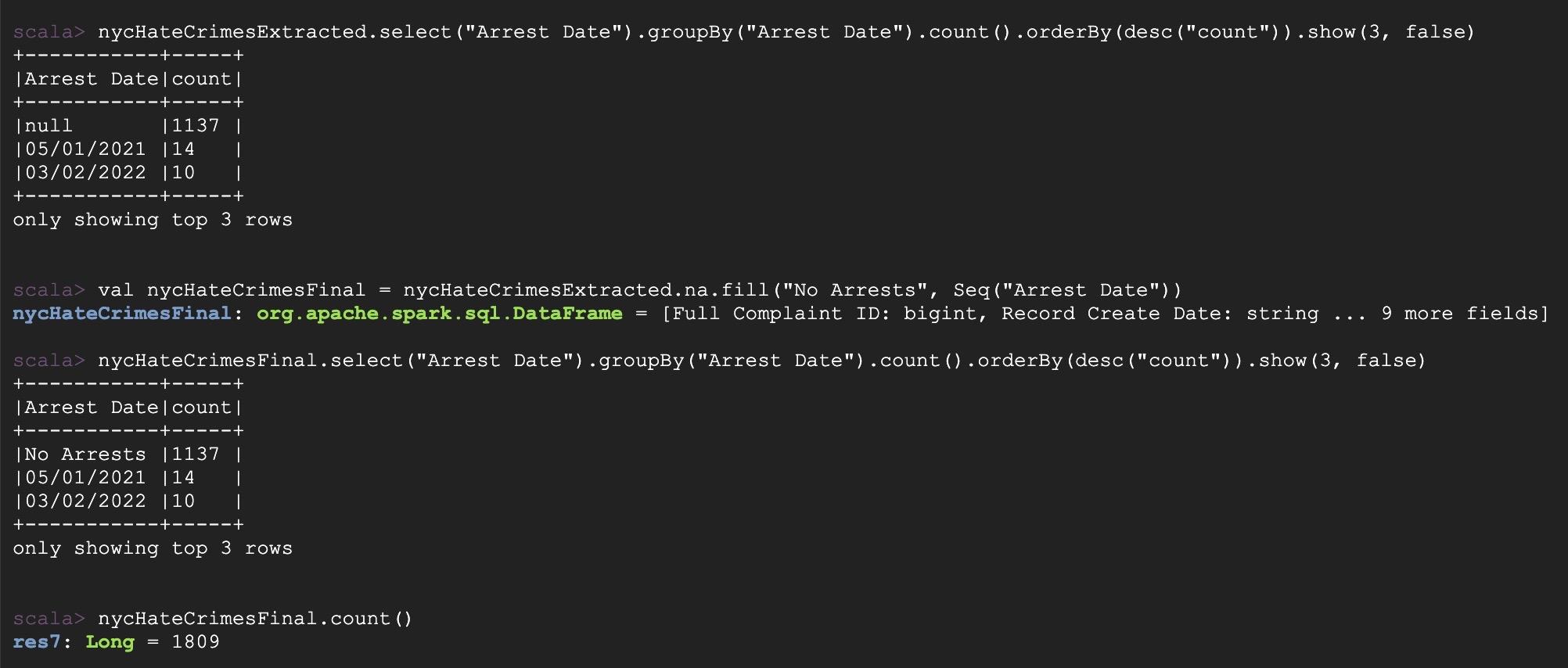
* val nycHateCrimesExtracted = nycHateCrimesOriginal.select("Full Complaint ID","Record Create Date","Complaint Precinct Code","Patrol Borough Name","County","Law Code Category Description","Offense Description","PD Code Description","Bias Motive Description","Offense Category","Arrest Date")
* nycHateCrimesOriginal.count()
* nycHateCrimesExtracted.count()



As seen above, originalData had 1809 rows , the same as the extractedData(1809 rows). As this dataset is relatively clean, only unnecessary columns were dropped.

“Arrest Date” column has null values as it is not necessary to have an arrest made. Hence , have filled all null value columns

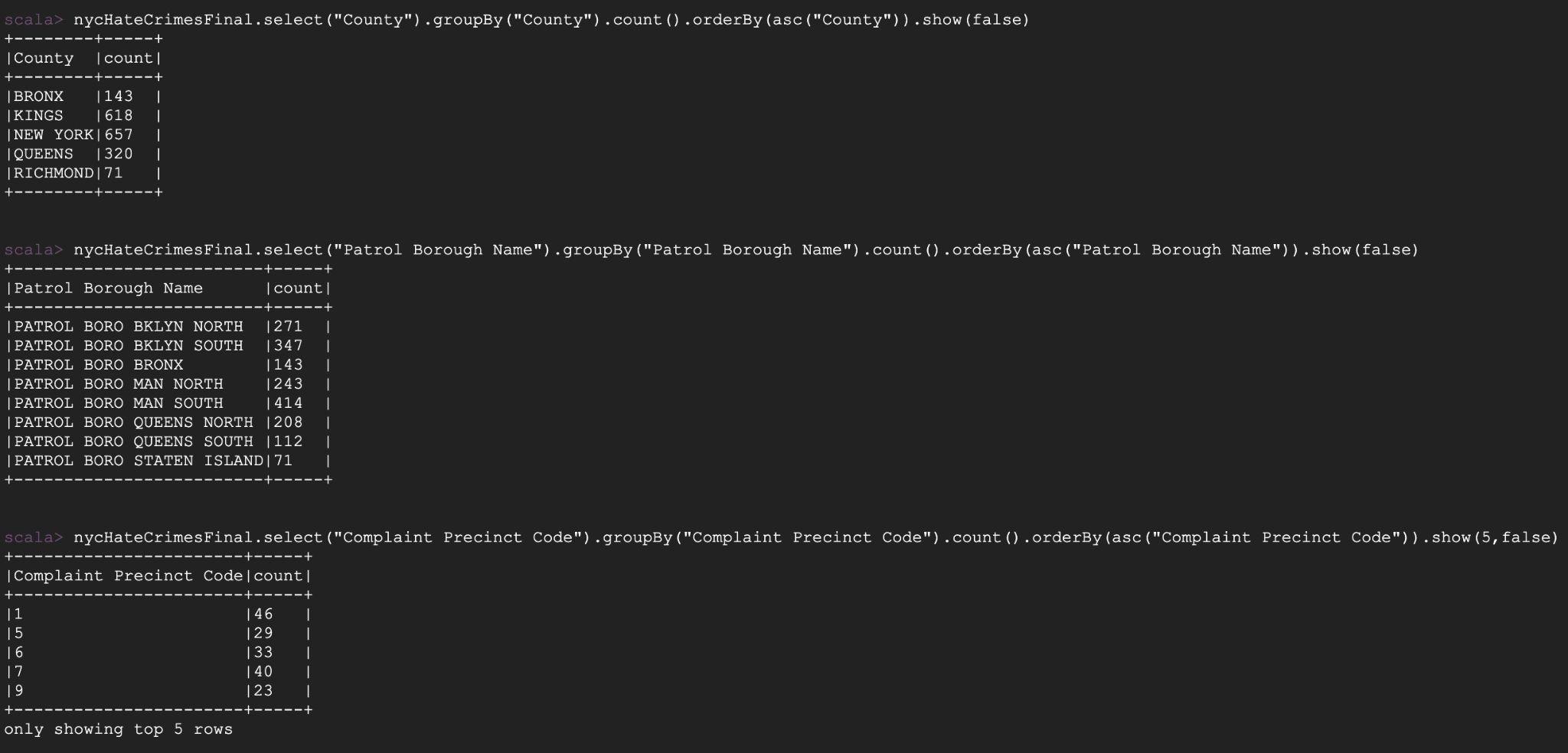
* nycHateCrimesExtracted.select("Arrest Date").groupBy("Arrest Date").count().orderBy(desc("count")).show(3, false)
* val nycHateCrimesFinal = nycHateCrimesExtracted.na.fill("No Arrests", Seq("Arrest Date"))
* nycHateCrimesFinal.select("Arrest Date").groupBy("Arrest Date").count().orderBy(desc("count")).show(3, false)



Step 3: Column Statistics

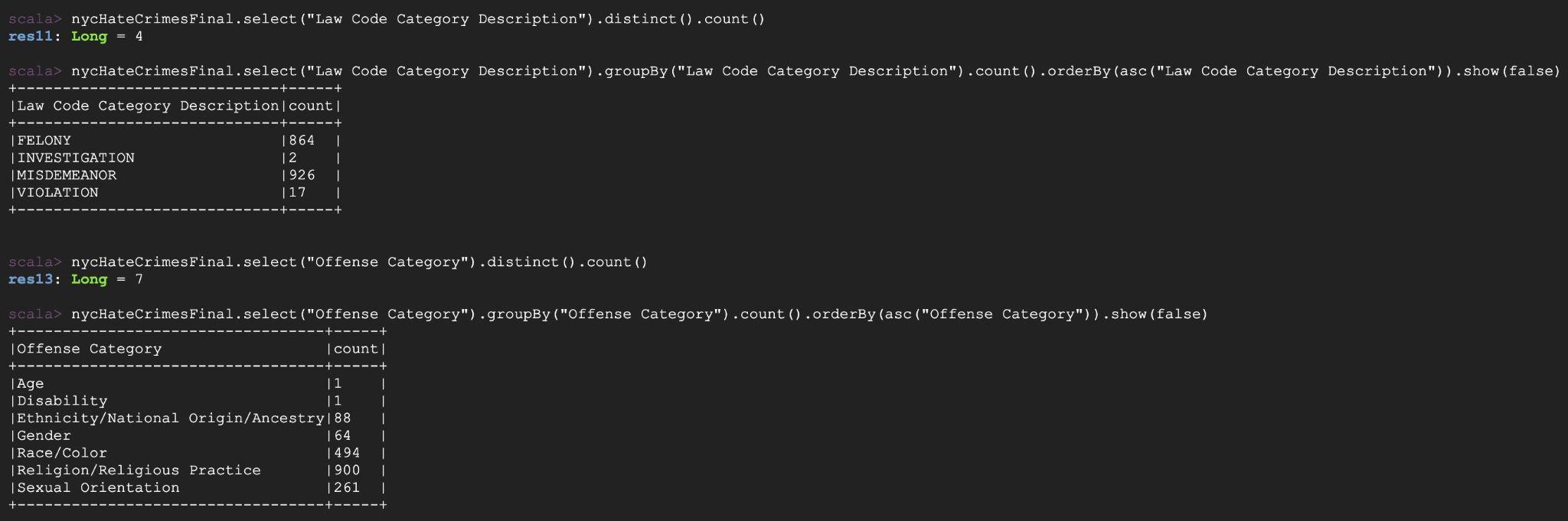
County, Patrol Borough, Precinct Code Statistics:

* nycHateCrimesFinal.select("County").groupBy("County").count().orderBy(asc("County")).show(false)
* nycHateCrimesFinal.select("Patrol Borough Name").groupBy("Patrol Borough Name").count().orderBy(asc("Patrol Borough Name")).show(false)
* nycHateCrimesFinal.select("Complaint Precinct Code").groupBy("Complaint Precinct Code").count().orderBy(asc("Complaint Precinct Code")).show(5,false)



For “Law Code Category Description”, “Offense Category”:

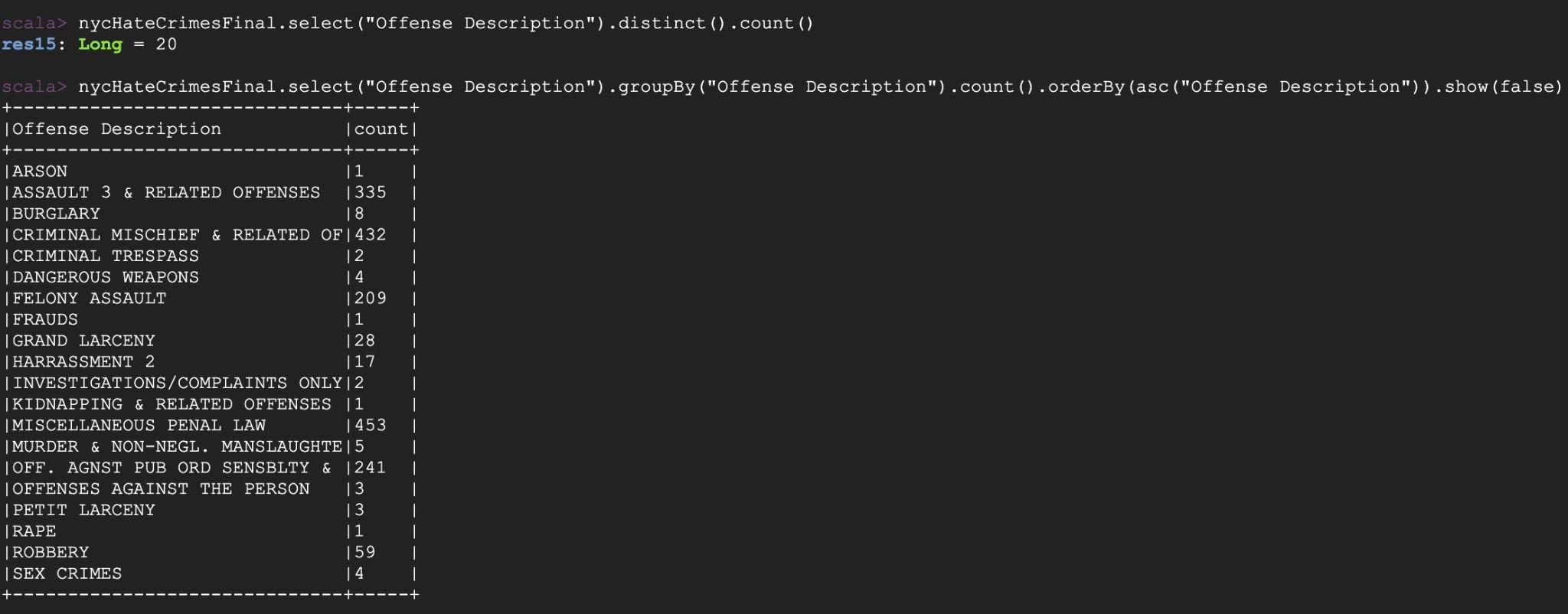
* nycHateCrimesFinal.select("Law Code Category Description").distinct().count()
* nycHateCrimesFinal.select("Law Code Category Description").groupBy("Law Code Category Description").count().orderBy(asc("Law Code Category Description")).show(false)
* nycHateCrimesFinal.select("Offense Category").distinct().count()
* nycHateCrimesFinal.select("Offense Category").groupBy("Offense Category").count().orderBy(asc("Offense Category")).show(false)



For “Offense Description”:

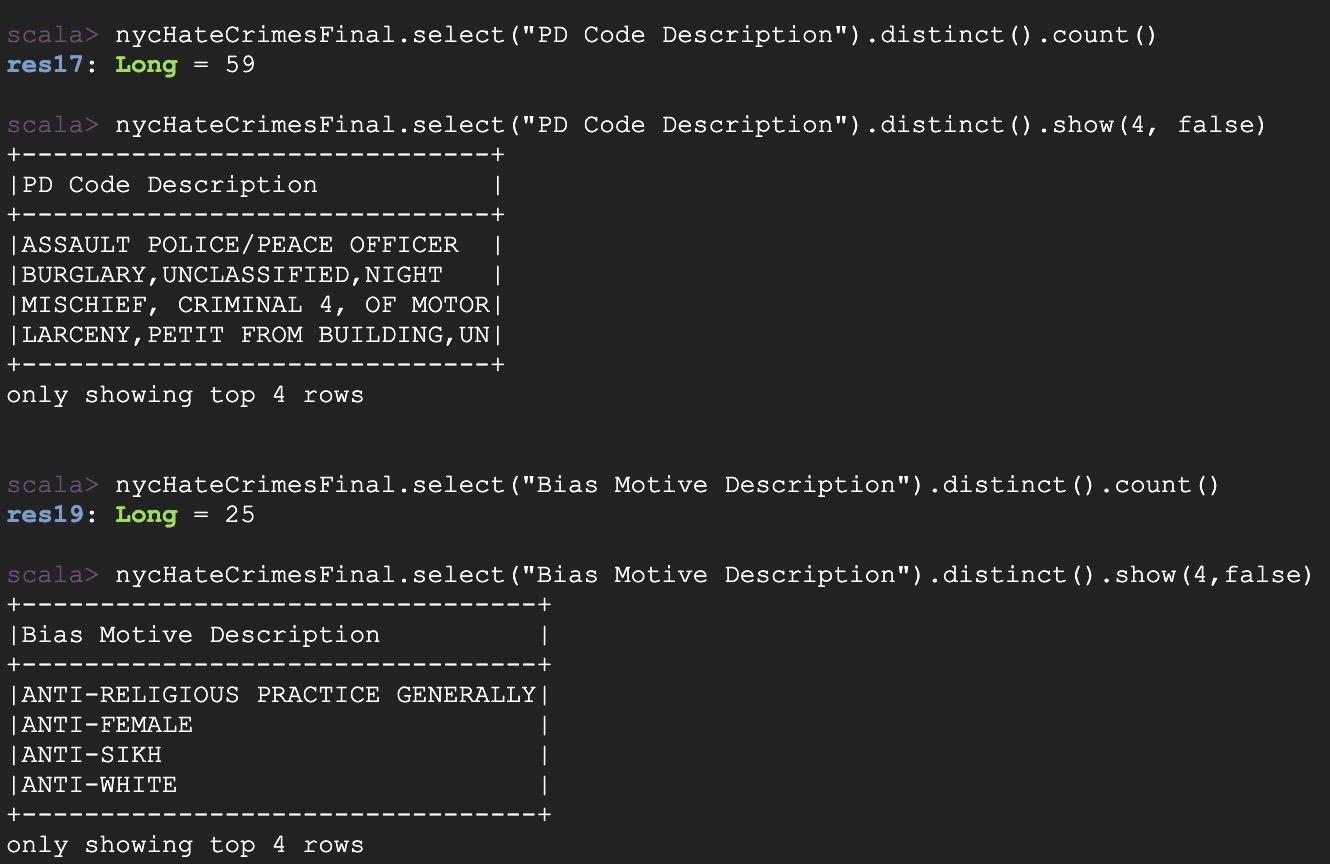
nycHateCrimesFinal.select("Offense Description").distinct().count()

nycHateCrimesFinal.select("Offense Description").groupBy("Offense Description").count().orderBy(asc("Offense Description")).show(false)



For “PD Code Description” and “Bias Motive Description” :

* nycHateCrimesFinal.select("PD Code Description").distinct().count()
* nycHateCrimesFinal.select("PD Code Description").distinct().show(4, false)
* nycHateCrimesFinal.select("Bias Motive Description").distinct().count()
* nycHateCrimesFinal.select("Bias Motive Description").distinct().show(4,false)



## Saving the files:

Creating DataIngestion.scala file with following content to submit spark job:

import org.apache.spark.SparkContext

import org.apache.spark.SparkContext.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.functions.{col}

import org.apache.spark.sql.functions.udf

import java.text.SimpleDateFormat

object DataIngestion {

def getTs(dateStr: String): String = {

val inputStr = new SimpleDateFormat("hh:mm:ssaa")

val outputStr = new SimpleDateFormat("HH:mm:ss")

val timeStr = dateStr.split(" ")(1) + dateStr.split(" ")(2)

val timeTS = outputStr.format(inputStr.parse(timeStr))

timeTS

}

def main(args: Array[String]) {

val spark = SparkSession.builder().getOrCreate()

val nyc911CallsfileName = "/user/yc4953\_nyu\_edu/project/NYPD\_Calls\_for\_Service\_\_Historic\_.csv"

val nyc911callsOriginal = spark.read.format("csv").option("header","true").option("inferSchema","true").load(nyc911CallsfileName)

val nyc911callsExtracted = nyc911callsOriginal.select("CAD\_EVNT\_ID","CREATE\_DATE","INCIDENT\_DATE","INCIDENT\_TIME","NYPD\_PCT\_CD","BORO\_NM","PATRL\_BORO\_NM","RADIO\_CODE","TYP\_DESC","CIP\_JOBS","ADD\_TS","DISP\_TS","ARRIVD\_TS","CLOSNG\_TS","Latitude","Longitude")

val nyc911callsNADrop = nyc911callsExtracted.na.drop(Seq("ARRIVD\_TS","NYPD\_PCT\_CD","DISP\_TS","CLOSNG\_TS"))

val nyc911callsFiltered = nyc911callsNADrop.filter(!(col("BORO\_NM").equalTo("(null)"))).filter(!(col("PATRL\_BORO\_NM").equalTo("(null)"))).filter(!(col("TYP\_DESC").equalTo("(null)")))

val transformTs = udf((x : String) => getTs(x))

val nyc911callstransformedTS = nyc911callsFiltered.withColumn("ADD\_TS\_TRANSFORMED", transformTs(col("ADD\_TS"))).

withColumn("DISP\_TS\_TRANSFORMED", transformTs(col("DISP\_TS"))).

withColumn("ARRIVD\_TS\_TRANSFORMED", transformTs(col("ARRIVD\_TS"))).

withColumn("CLOSNG\_TS\_TRANSFORMED", transformTs(col("CLOSNG\_TS")))

val nyc911callsFinal = nyc911callstransformedTS.select("CAD\_EVNT\_ID","CREATE\_DATE","INCIDENT\_DATE","INCIDENT\_TIME","NYPD\_PCT\_CD","BORO\_NM","PATRL\_BORO\_NM","RADIO\_CODE","TYP\_DESC","CIP\_JOBS","ADD\_TS\_TRANSFORMED","DISP\_TS\_TRANSFORMED","ARRIVD\_TS\_TRANSFORMED","CLOSNG\_TS\_TRANSFORMED","Latitude","Longitude")

nyc911callsFinal.write.option("header",true).format("csv").save("/user/yc4953\_nyu\_edu/project/NYC911CallsDataFinal")

val nycHateCrimesfileName = "/user/yc4953\_nyu\_edu/project/NYPD\_Hate\_Crimes.csv"

val nycHateCrimesOriginal = spark.read.format("csv").option("header","true").option("inferSchema","true").load(nycHateCrimesfileName)

val nycHateCrimesExtracted = nycHateCrimesOriginal.select("Full Complaint ID","Record Create Date","Complaint Precinct Code","Patrol Borough Name","County","Law Code Category Description","Offense Description","PD Code Description","Bias Motive Description","Offense Category","Arrest Date")

val nycHateCrimesFinal = nycHateCrimesExtracted.na.fill("No Arrests", Seq("Arrest Date"))

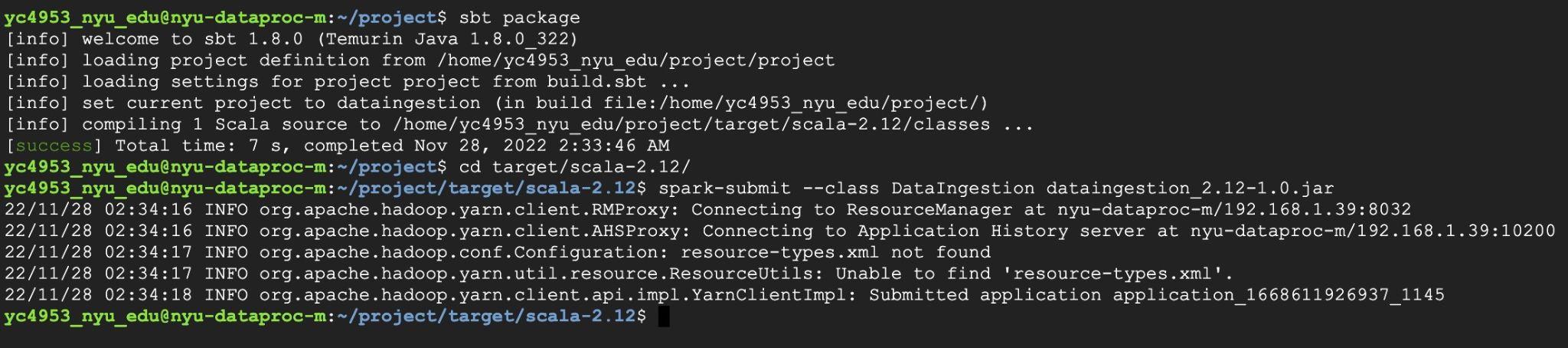
nycHateCrimesFinal.write.option("header",true).format("csv").save("/user/yc4953\_nyu\_edu/project/HateCrimeDataFinal")

}

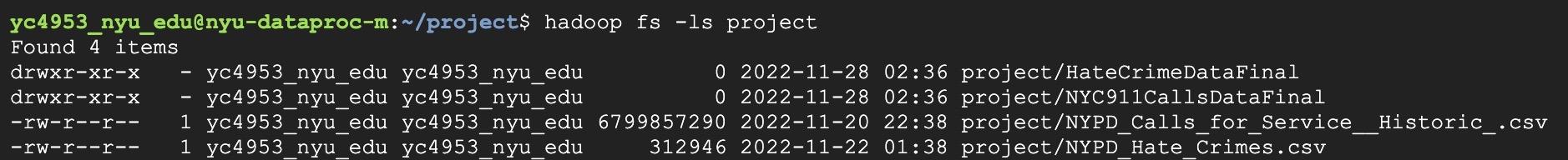
}

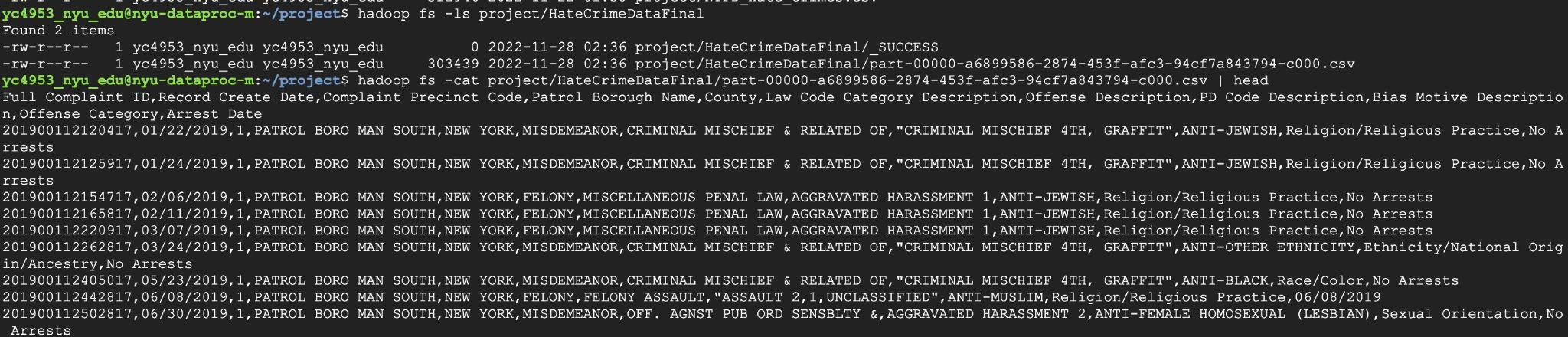
sbt package

spark-submit --class DataIngestion dataingestion\_2.12-1.0.jar

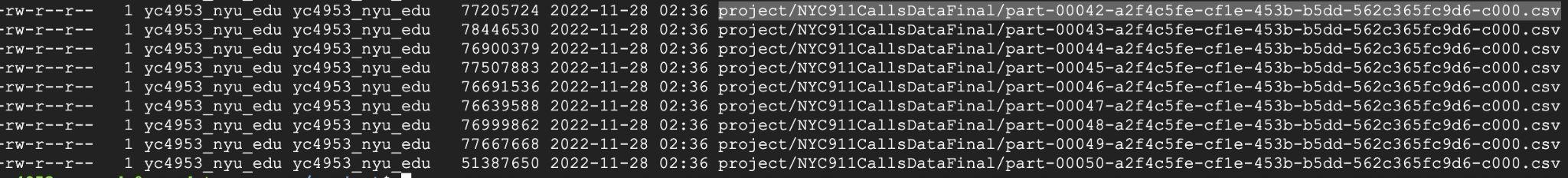


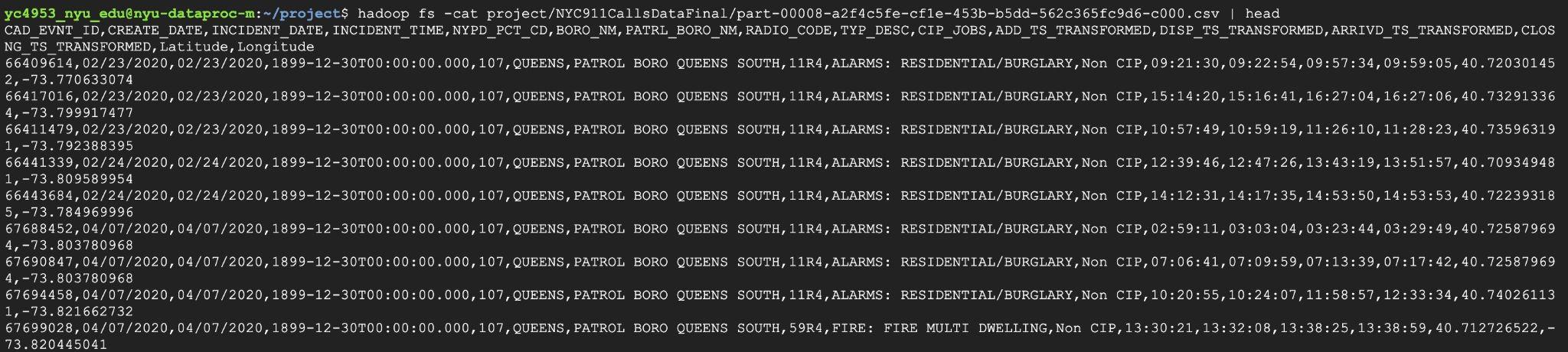
Confirming files have been generated as follows:











When the need arises, more processing will be done.