**BDAD**

**Project Data Ingestion Report**

## **How Safe NYC really is?**

### **Project Synopsis:**

In this project we intend to focus on the crimes committed in New York City. By analyzing data from complaints registered with the NYPD, arrests made by NYPD, distressed 911 calls described with various subcategories, we aim to find valuable insights of crime statistics across 5 boroughs of New York City. Our findings we believe shall help diagnosing and prescribing solutions such as delays in 911 dispatch help, prevention of certain types of crime in key areas using preemptive measures as well as demographics vis-a-vis crime-type to identify better programmes to alleviate the issues. By discovering the locations of most dangerous crimes , the timing stats can help in establishing better patrolling and prevention policies.

**Keywords**:Big Data, Apache Spark, Scala, Tableau, Spark SQL, MS Excel

**Dataset 1:**

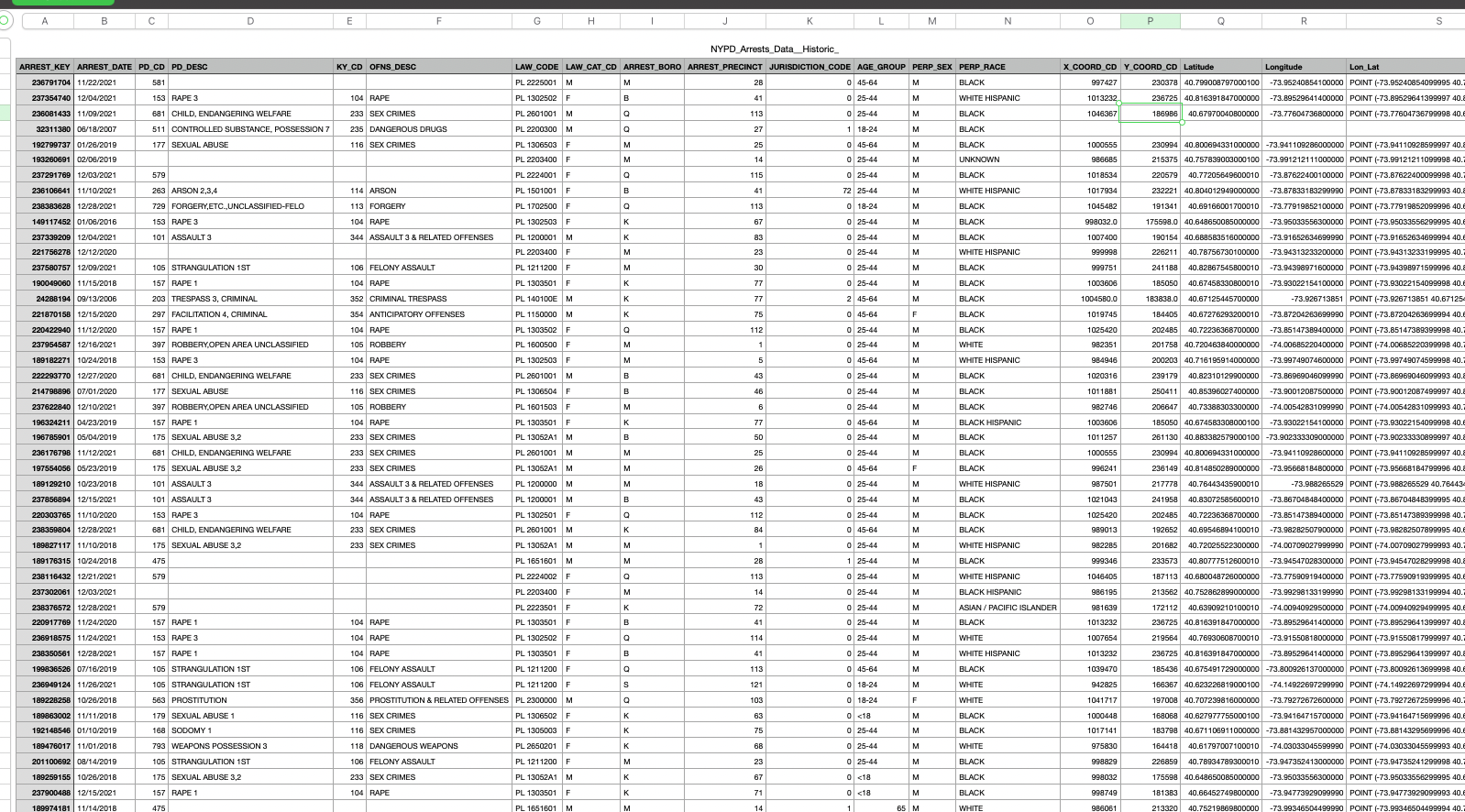
**NYPD Arrests**:

A list of each arrest made in New York City from the beginning of 2006 till the conclusion of the preceding year. It includes a breakdown of each arrest that the NYPD made in NYC from 2006 through the end of the previous calendar year. Every three months, the Office of Management Analysis and Planning manually extracts this data, reviews it, and then posts it on the NYPD website.

<https://catalog.data.gov/dataset/nypd-arrests-data-historic>

File size : 1.13 GB

**Preview of the data:**



Here we have 19 columns in total:

**ARREST\_KEY**: Unique ID for each of the arrests made

**ARREST\_DATE**: Date on which the arrest has happened

**PD\_CD:** Crime internal ID used by NYPD

**PD\_DESC**: Description of the Crime

**KY\_CD**:

**OFNS\_DESC**: Description of the offense

**LAW\_CODE**: ID of the officer who is responsible for the arrest

**LAW\_CAT\_CD**: Gender of the officer that made the arrest

**ARREST\_BORO**: New York City borough in which the arrest happened

**ARREST\_PRECINCT**: Precinct in which the arrest is made

**JURISDICTION\_CODE**: Code of the jurisdiction to which the precinct belongs to

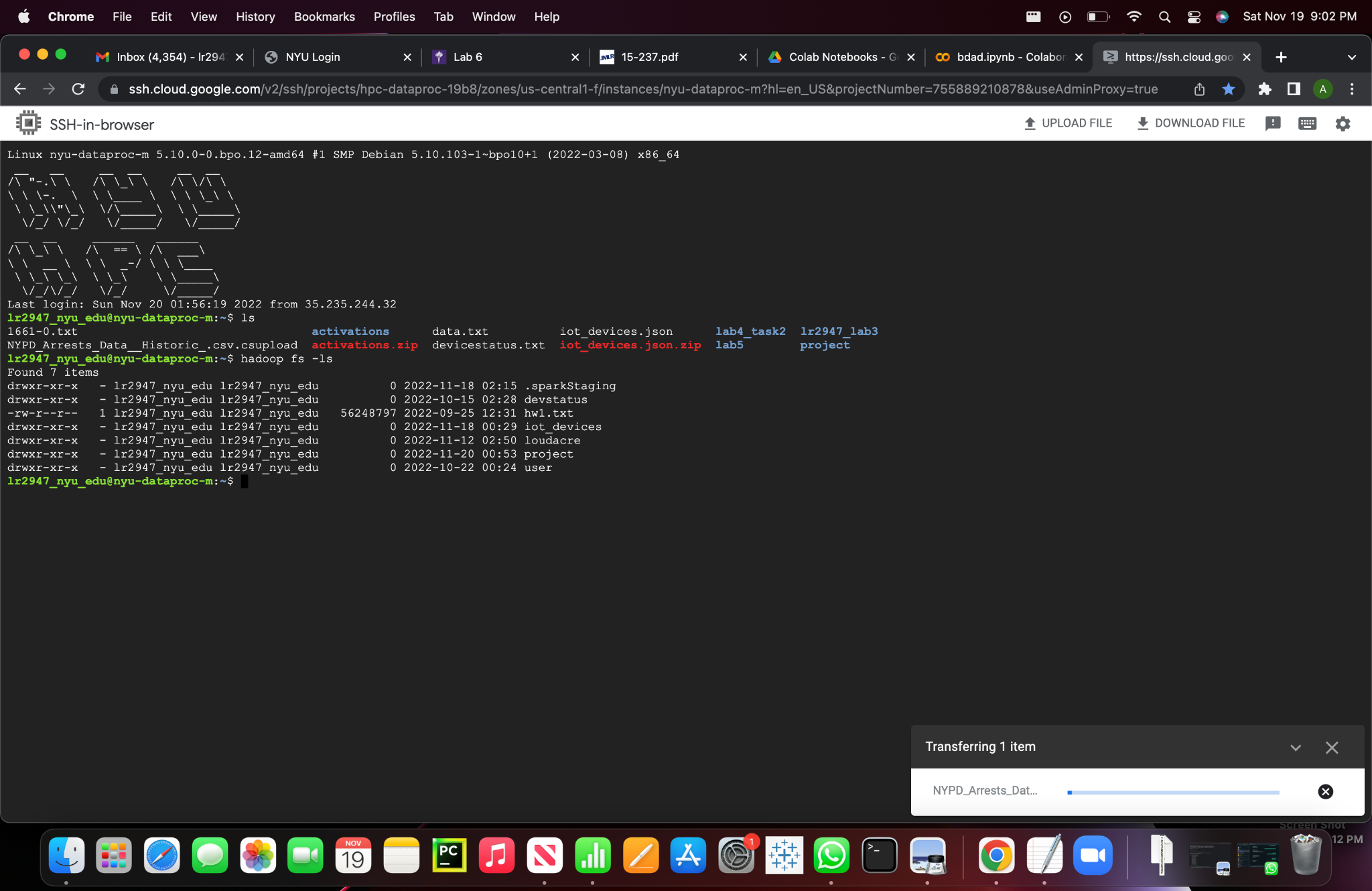
**AGE\_GROUP**: Age group to which the perpetrator belongs to

**PERP\_SEX**: Sex of the perpetrator

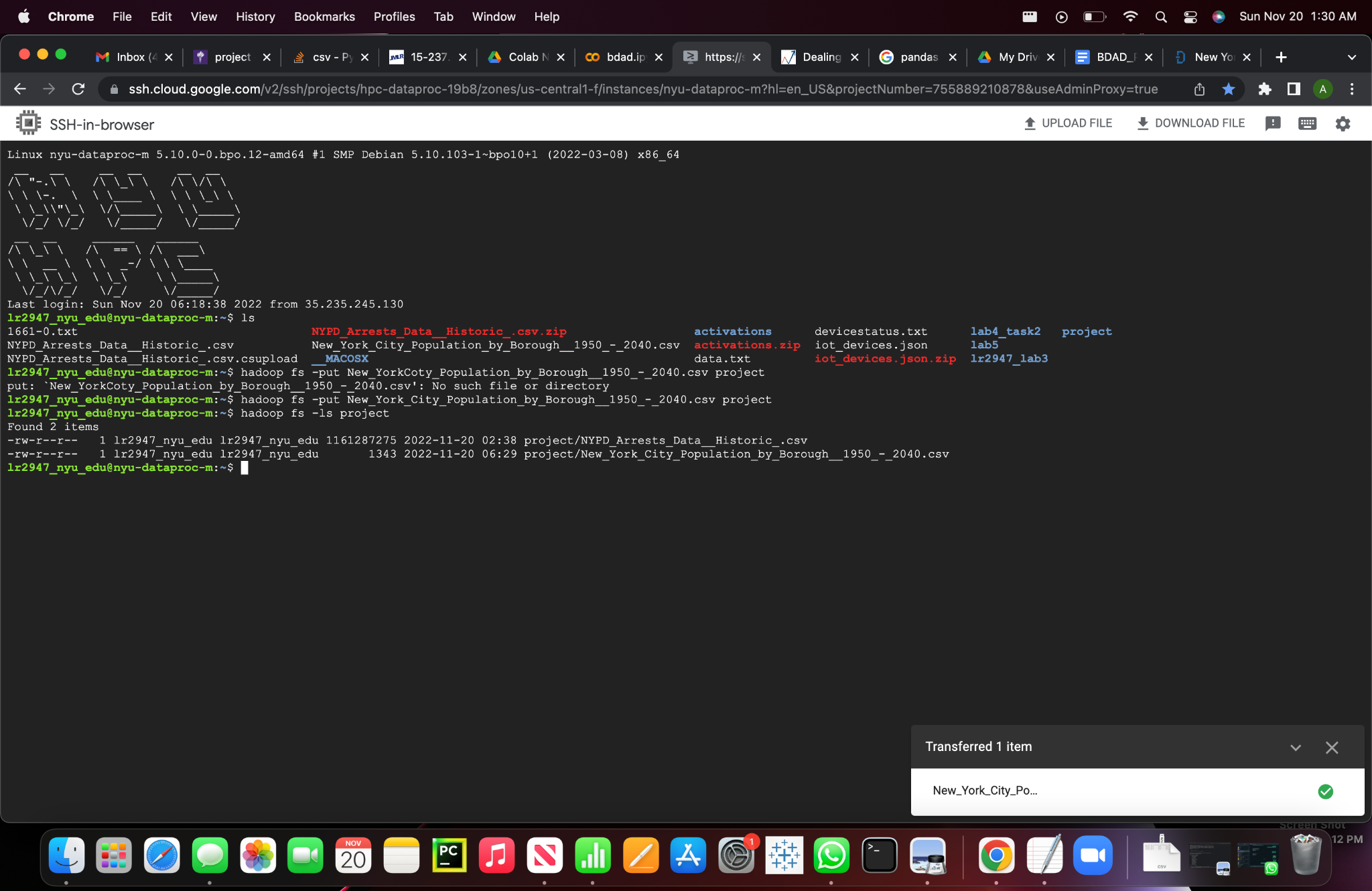
**PERP\_RACE**: Race of the perpetrator

(**X-Coord**, **Y-Coord)** and **Lat** and **Long**: These columns provide the geographical location of the arrest

We have placed our dataset into the hadoop cluster.



### 



Here, I would be dropping the columns, ARREST\_KEY, PD\_DESC, PD\_CD, KY\_CD, LAW\_CODE and LAW\_CAT\_CD, as there is not much information that comes from these columns. Also from the co-ordinates we do not get information for our initial analysis but this would be useful during the visualization of the arrests made by NYPD.

**Scala commands**:

-> val arrestDataPath = "/user/lr2947\_nyu\_edu/project/NYPD\_Arrests\_Data\_\_Historic\_.csv"

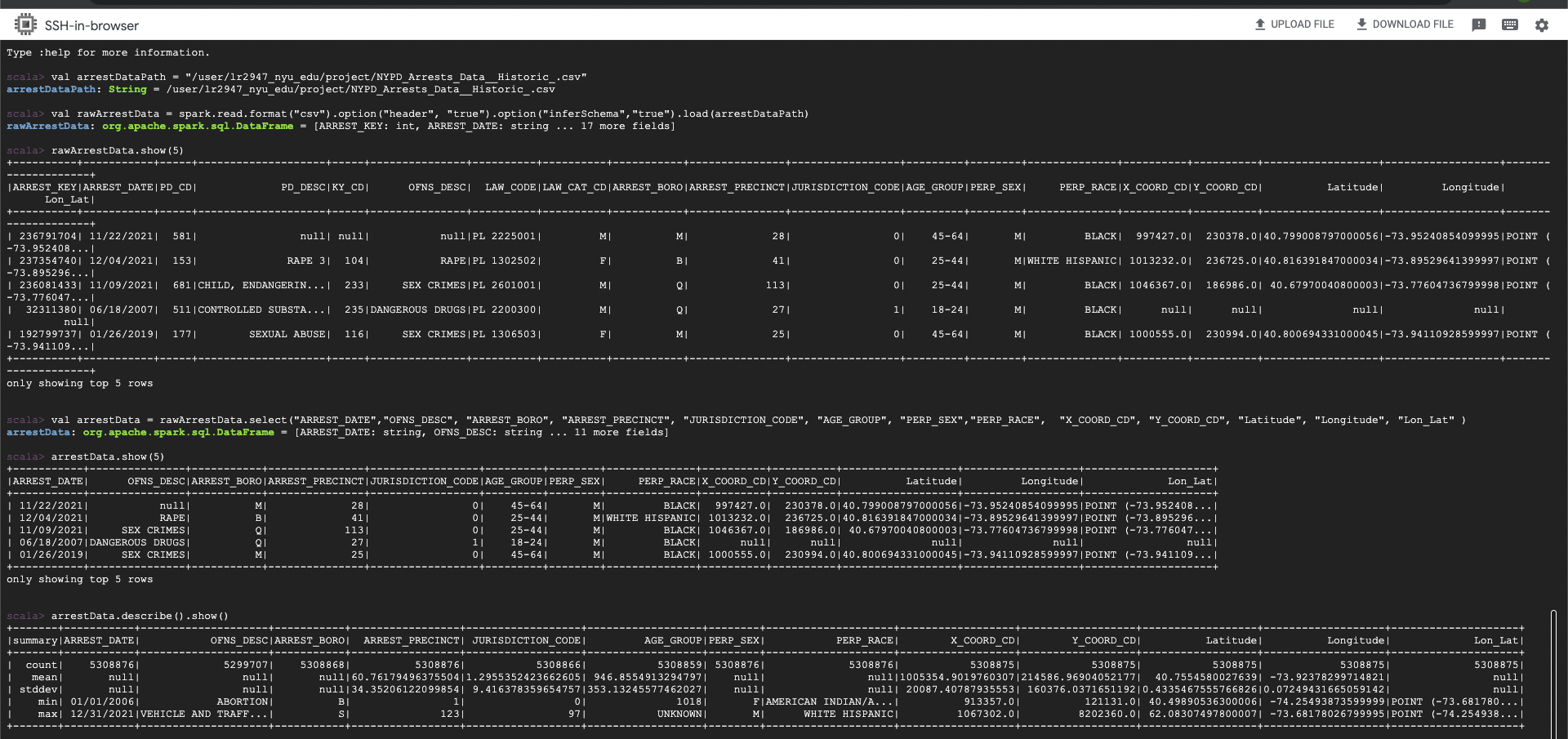
-> val rawArrestData = spark.read.format("csv").option("header", "true").option("inferSchema","true").load(arrestDataPath)

-> rawArrestData.show(5)

->val arrestData = raw ArrestData.select("ARREST\_DATE","OFNS\_DESC", "ARREST\_BORO", "ARREST\_PRECINCT", "JURISDICTION\_CODE", "AGE\_GROUP", "PERP\_SEX","PERP\_RACE", "X\_COORD\_CD", "Y\_COORD\_CD", "Latitude", "Longitude", "Lon\_Lat" )

-> arrestData.show(5)

-> arrestData.describe().show()



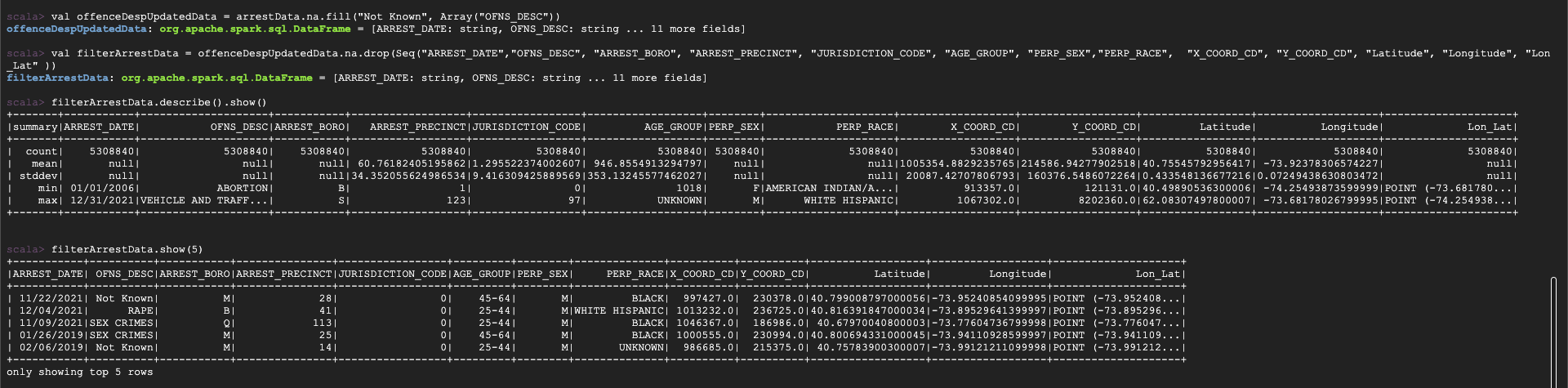
Have replaced the rows where offense description is not mentioned as Not known, and dropped null values

**Scala commands:**

-> val offenceDespUpdatedData = arrestData.na.fill("Not Known", Array("OFNS\_DESC"))

-> val filterArrestData = offenceDespUpdatedData.na.drop(Seq("ARREST\_DATE","OFNS\_DESC", "ARREST\_BORO", "ARREST\_PRECINCT", "JURISDICTION\_CODE", "AGE\_GROUP", "PERP\_SEX","PERP\_RACE", "X\_COORD\_CD", "Y\_COORD\_CD", "Latitude", "Longitude", "Lon\_Lat" ))

-> filterArrestData.describe().show()



Now lets see how many arrests have happened in each borough:

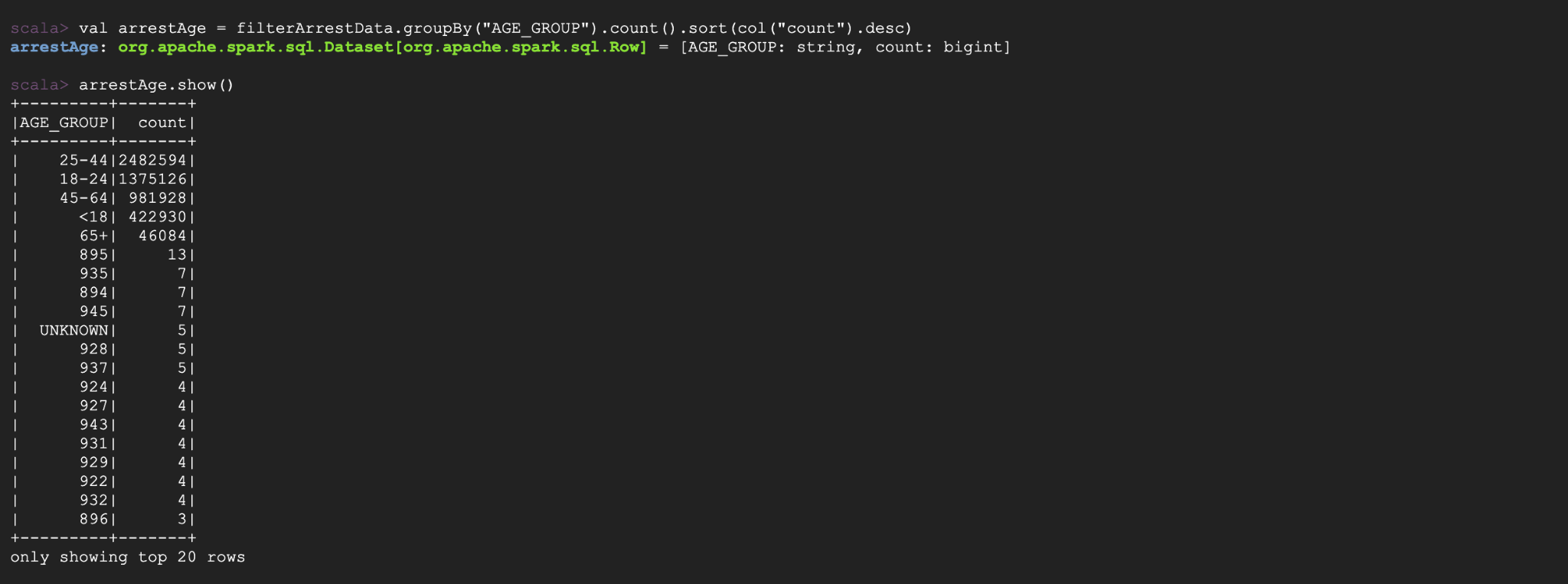
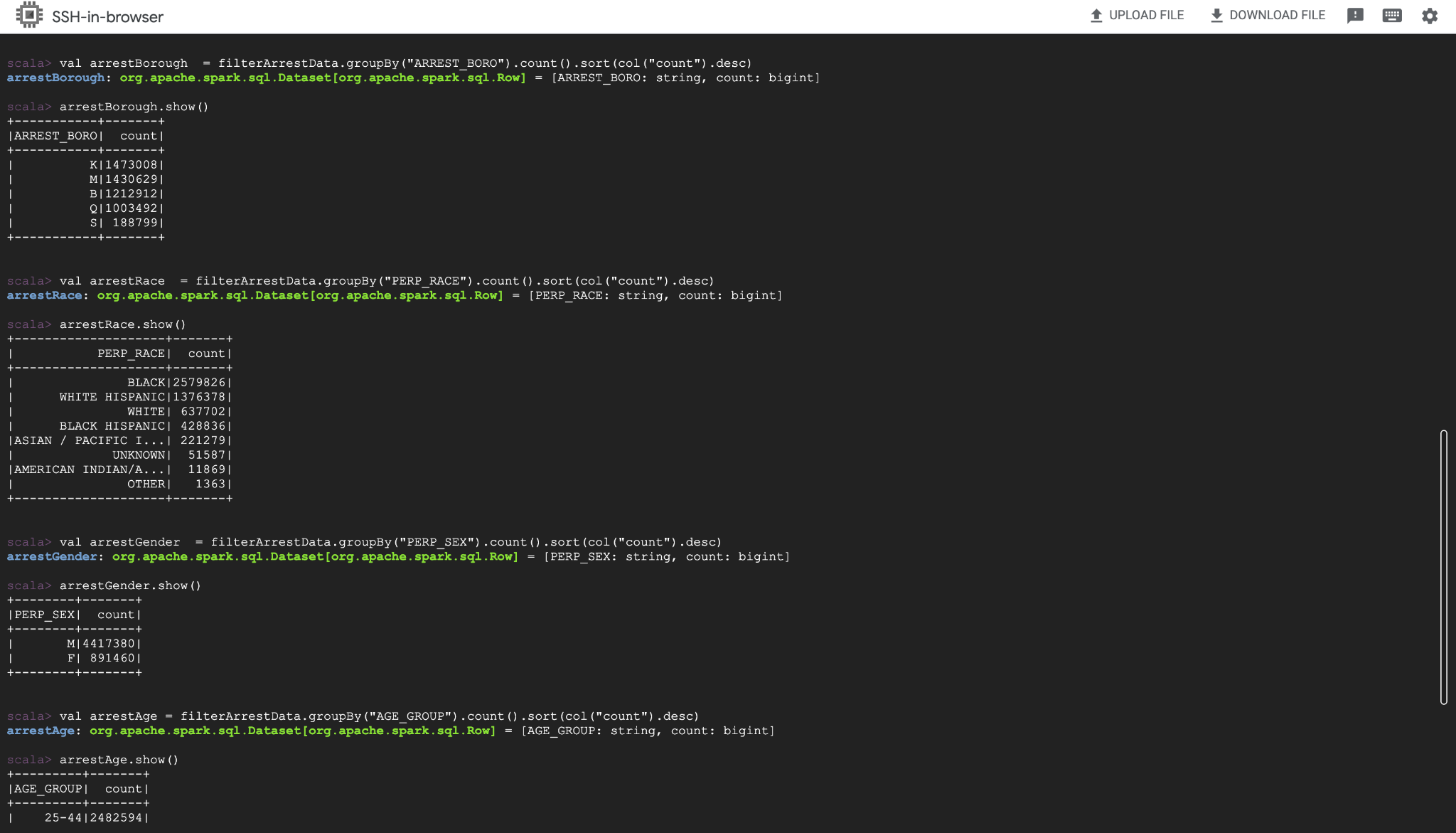
Scala command:

val arrestBorough = filterArrestData.groupBy("ARREST\_BORO").count().sort(col("count").desc)

val arrestRace = filterArrestData.groupBy("PERP\_RACE").count().sort(col("count").desc)

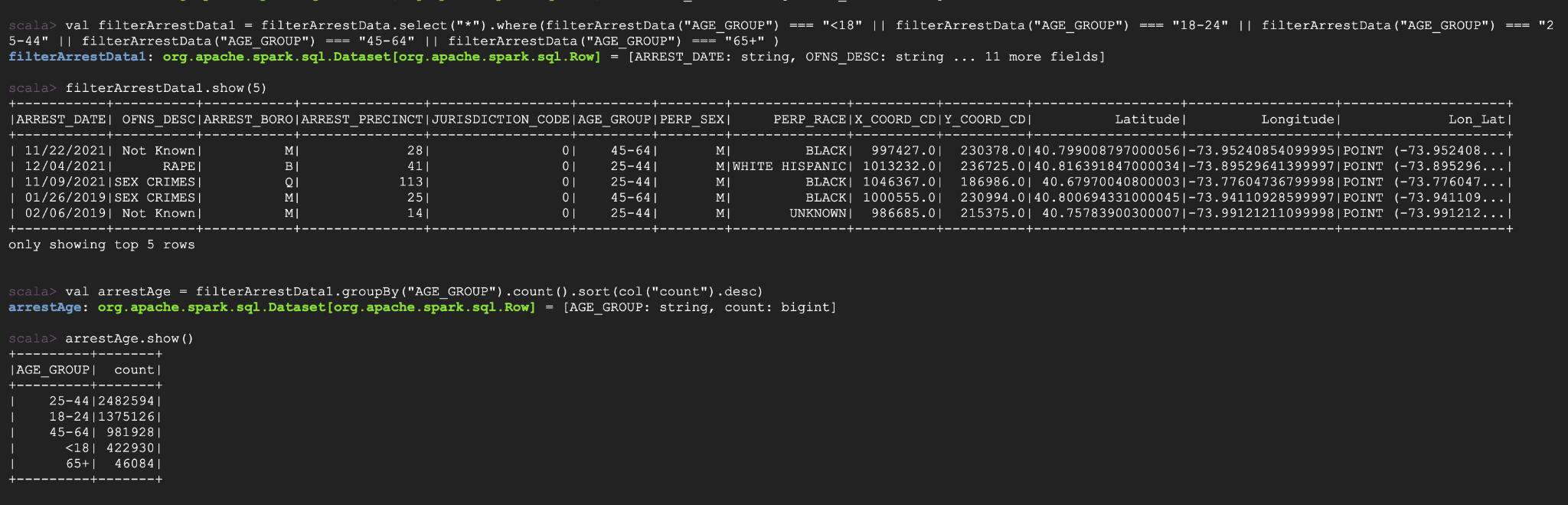
val arrestGender = filterArrestData.groupBy("PERP\_SEX").count().sort(col("count").desc)

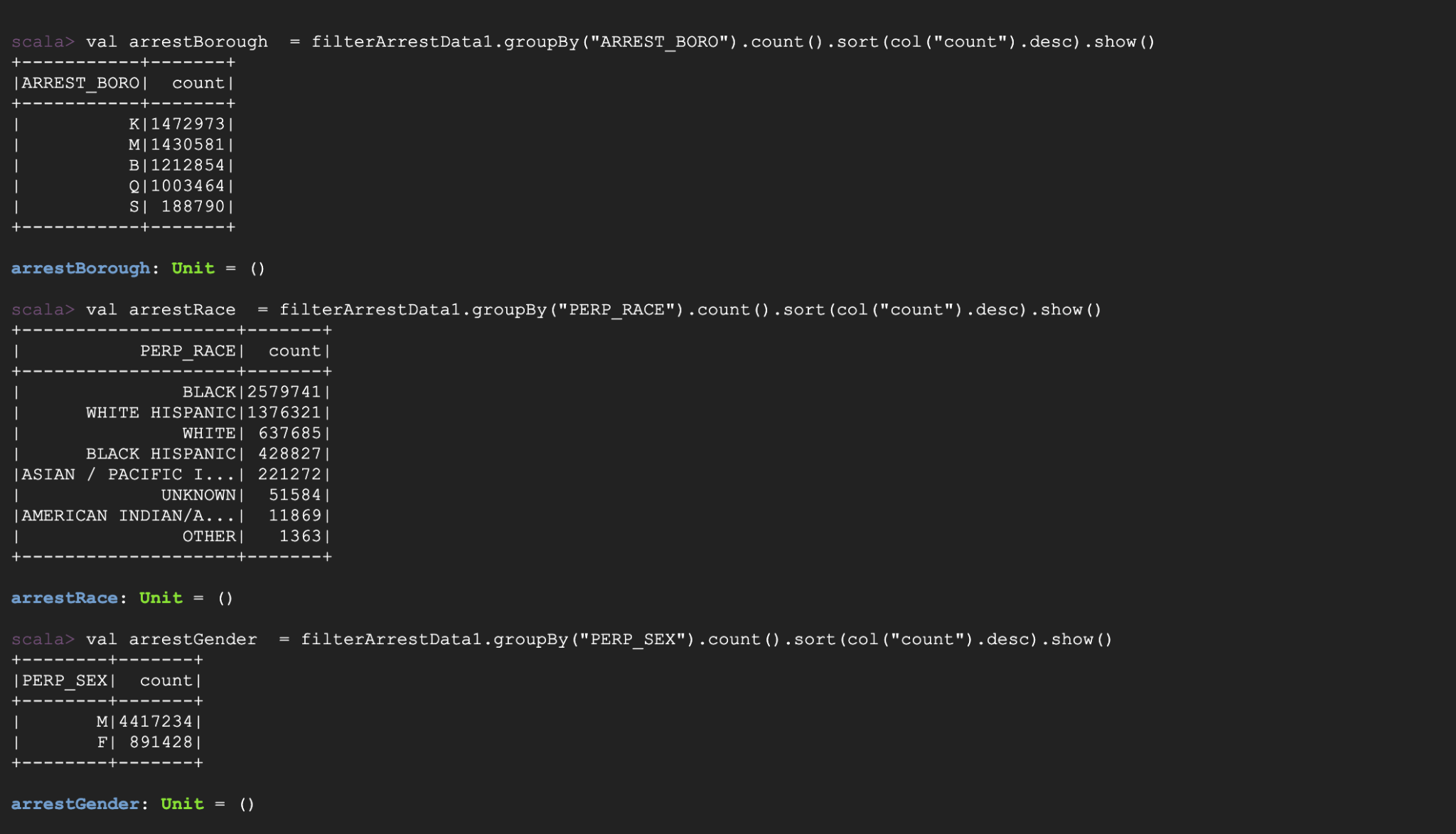
val arrestAge = filterArrestData.groupBy("AGE\_GROUP").count().sort(col("count").desc)



As we can see we need to clean the data further since age groups after 65+ make no sense.

We need to drop those rows as well and the data entry for these rows is not reliable





Scala Commands:

-> val filterArrestData1 = filterArrestData.select("\*").where(filterArrestData("AGE\_GROUP") === "<18" || filterArrestData("AGE\_GROUP") === "18-24" || filterArrestData("AGE\_GROUP") === "25-44" || filterArrestData("AGE\_GROUP") === "45-64" || filterArrestData("AGE\_GROUP") === "65+" )

-> val arrestAge = filterArrestData1.groupBy("AGE\_GROUP").count().sort(col("count").desc)

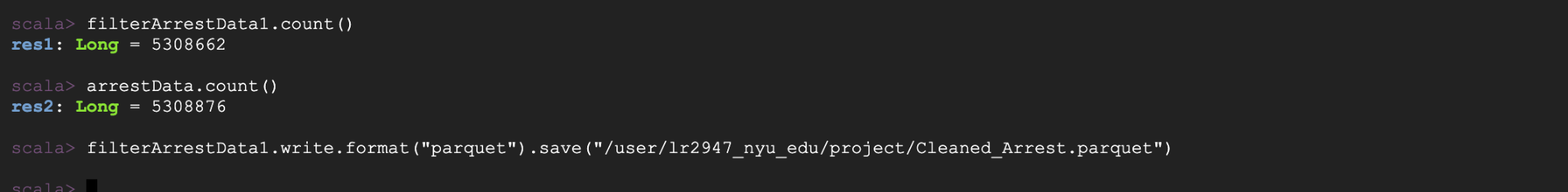
-> val arrestBorough = filterArrestData1.groupBy("ARREST\_BORO").count().sort(col("count").desc).show()

-> val arrestRace = filterArrestData1.groupBy("PERP\_RACE").count().sort(col("count").desc).show()

-> val arrestGender = filterArrestData1.groupBy("PERP\_SEX").count().sort(col("count").desc).show()

Initially there are about 5308876 records, and post cleaning we have 5308662 records

Saving the cleaned data: filterArrestData1.write.format("parquet").save("/user/lr2947\_nyu\_edu/project/Cleaned\_Arrest.parquet")



Dataset 2:

**NYC population by Borough**

Summary table of New York City population numbers and percentage shares by Borough, including school-age (5 to 17), 65 and Over, and total population for over the years 1950-2040. The data is updated and maintained by the NYC government and is made publically available.

<https://catalog.data.gov/dataset/new-york-city-population-by-borough-1950-2040>

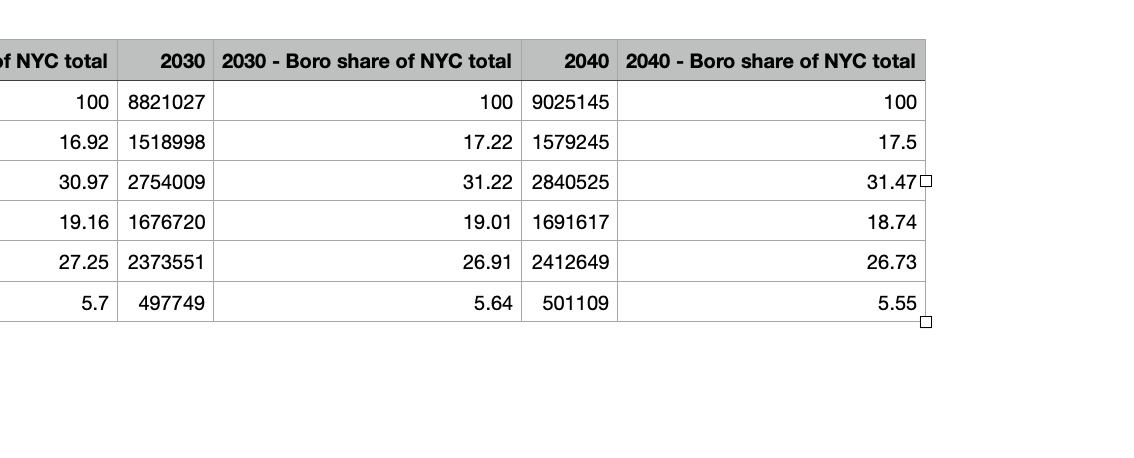
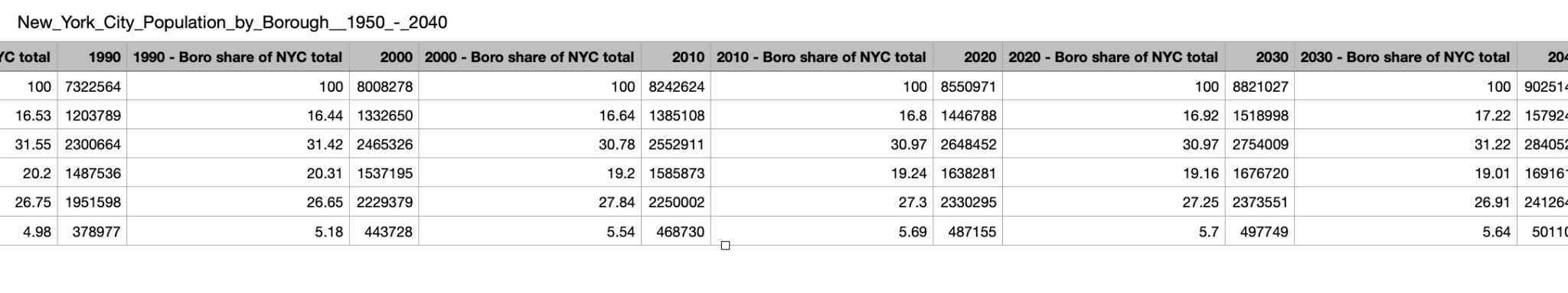
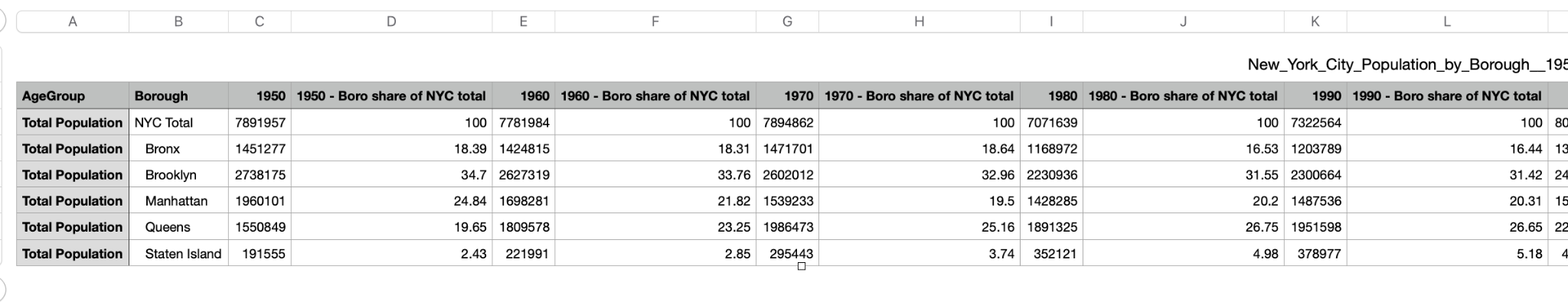
Columns:

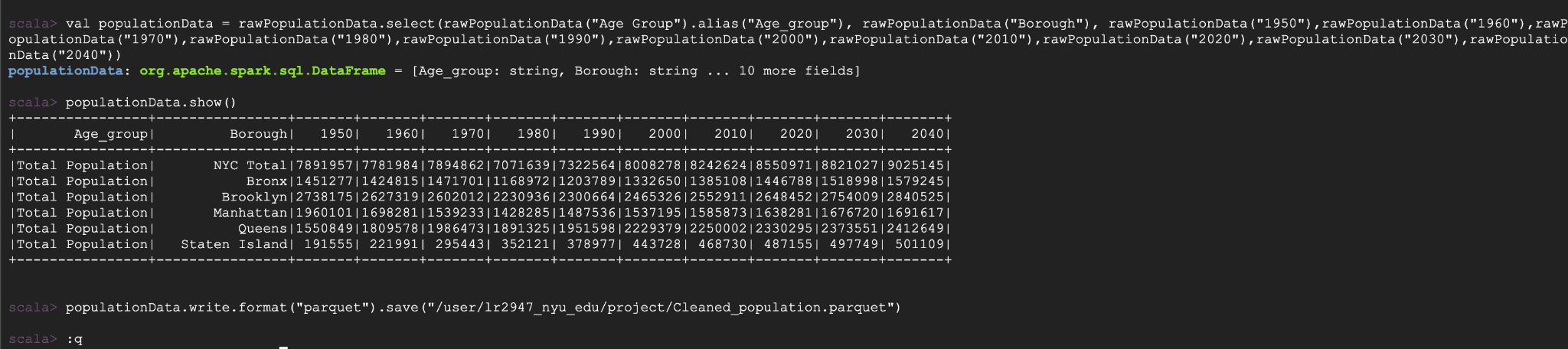
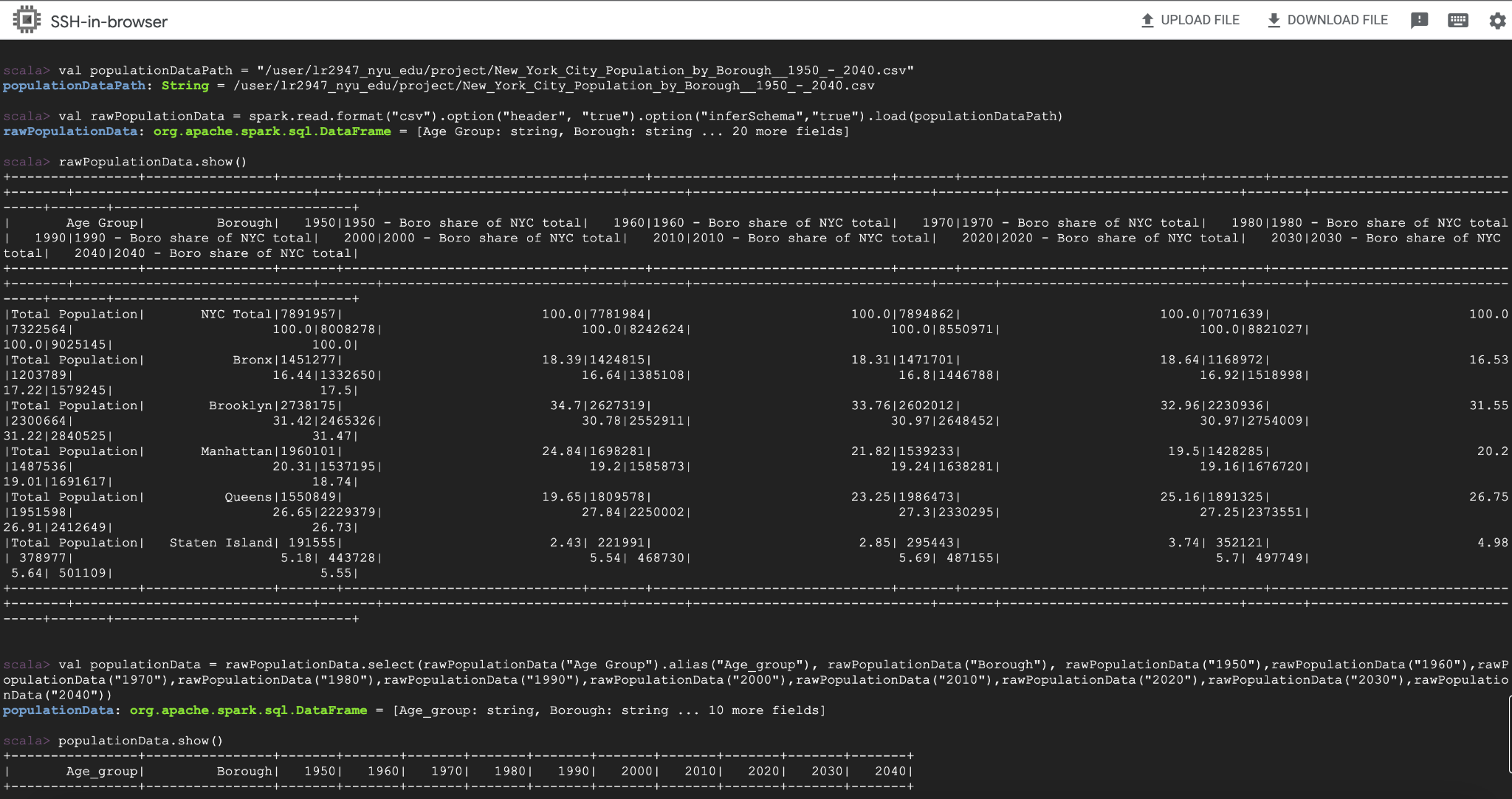
**Age Group**: Indicates total population

**Borough:** NYC Boroughs

**1950, 1960, … 2040**: Population in that year in NYC and borough wise

**1950- Boro share of NYC total, 1960- Boro share of NYC total, …, 2040- Boro share of NYC total**: Percentage population of NYC in each Borough





Have kept only population year wise, since we can calculate by percentage if required for the further analysis.

Scala commands:

-> val populationDataPath = "/user/lr2947\_nyu\_edu/project/New\_York\_City\_Population\_by\_Borough\_\_1950\_-\_2040.csv"

-> val rawPopulationData = spark.read.format("csv").option("header", "true").option("inferSchema","true").load(populationDataPath)

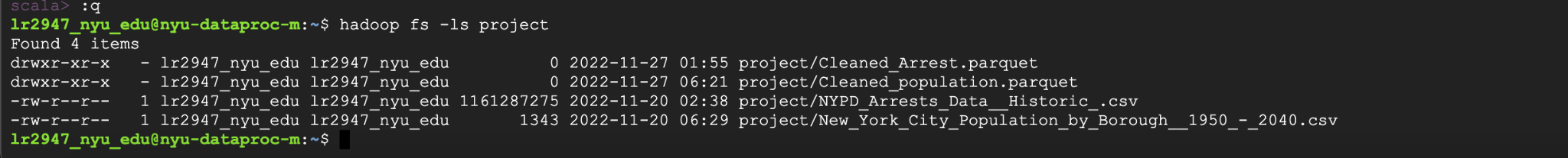
-> rawPopulationData.show()

-> val populationData = rawPopulationData.select(rawPopulationData("Age Group").alias("Age\_group"), rawPopulationData("Borough"), rawPopulationData("1950"),rawPopulationData("1960"),rawPopulationData("1970"),rawPopulationData("1980"),rawPopulationData("1990"),rawPopulationData("2000"),rawPopulationData("2010"),rawPopulationData("2020"),rawPopulationData("2030"),rawPopulationData("2040"))

-> populationData.write.format("parquet").save("/user/lr2947\_nyu\_edu/project/Cleaned\_population.parquet")

As this dataset is relatively small, much cleaning was not needed.

Both the files after cleaning are stored in the cluster.



**NOTE:** Further processing and cleaning might be required in the future as the project develops and we integrate it overall.