CS 779 Advance Database Management
Term Project

Covid19 social impact mining by Spark

Student: Chen Wu

Professor: Jack Polnar

Description

Covid19 has been heavily impacting social life this year, we could see changes on every social aspect. This project measures public safety, energy usage changes associated to covid19.

- Goal: Explore social changes caused by Covid-19 this year.

- How to: collecting data from New York Government, specific in Covid-19 data, crime, utility usage, vehicle collisions and shooting cases.

- Measure: To see how those data change by month in 2020.

- Tool: Spark on Databricks

- Language: SQL, Python

Data collection

I collected data from New York City public data: https://opendata.cityofnewyork.us/

Source:

- NYC covid-19:
  https://data.cityofnewyork.us/Health/COVID-19-Daily-Counts-of-Cases-Hospitalizations-an/rc75-m7u3

- Energy and water:
  https://data.cityofnewyork.us/Environment/Energy-and-Water-Data-Disclosure-for-Local-Law-84-/qb3v-bbre

- Vehicle Collisions:
  https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95

- NYC death cases(by year, used to compare 2020):
  https://data.cityofnewyork.us/Health/New-York-City-Leading-Causes-of-Death/jb7j-dtam

- NYC prisoner:
  https://data.cityofnewyork.us/Public-Safety/Daily-Inmates-In-Custody/7479-ugqb

- Violence:
  https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Current-Year-To-Date-/5uac-w243
- Shooting:
  https://data.cityofnewyork.us/Public-Safety/NYPD-Shooting-Incident-Data-Year-To-Date-/5ucz-vwe8

Data Transformation and Cleaning

It varies by the dataset, general steps are below:

1. Read csv files into spark data frames

2. Attribute selection

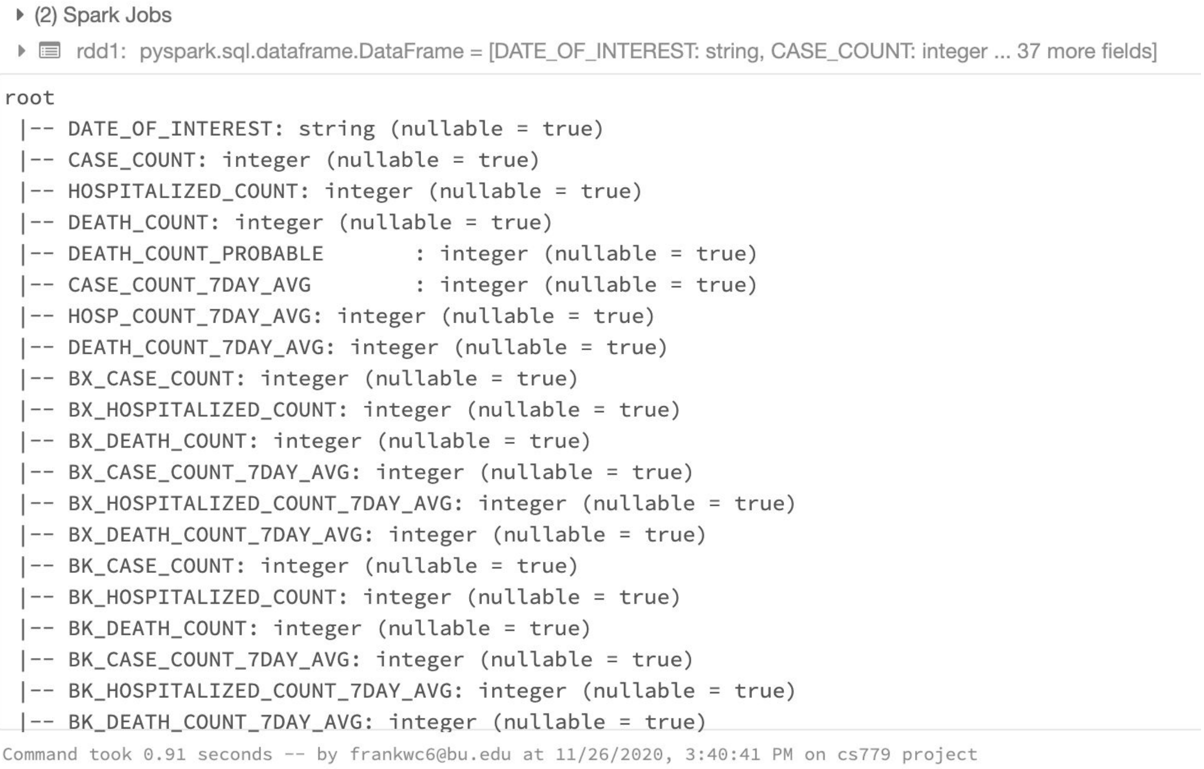
3. Convert date to date type with extraction of year, month, day.

4. transform data to what will be used on tables if needed.

5. sort by date.

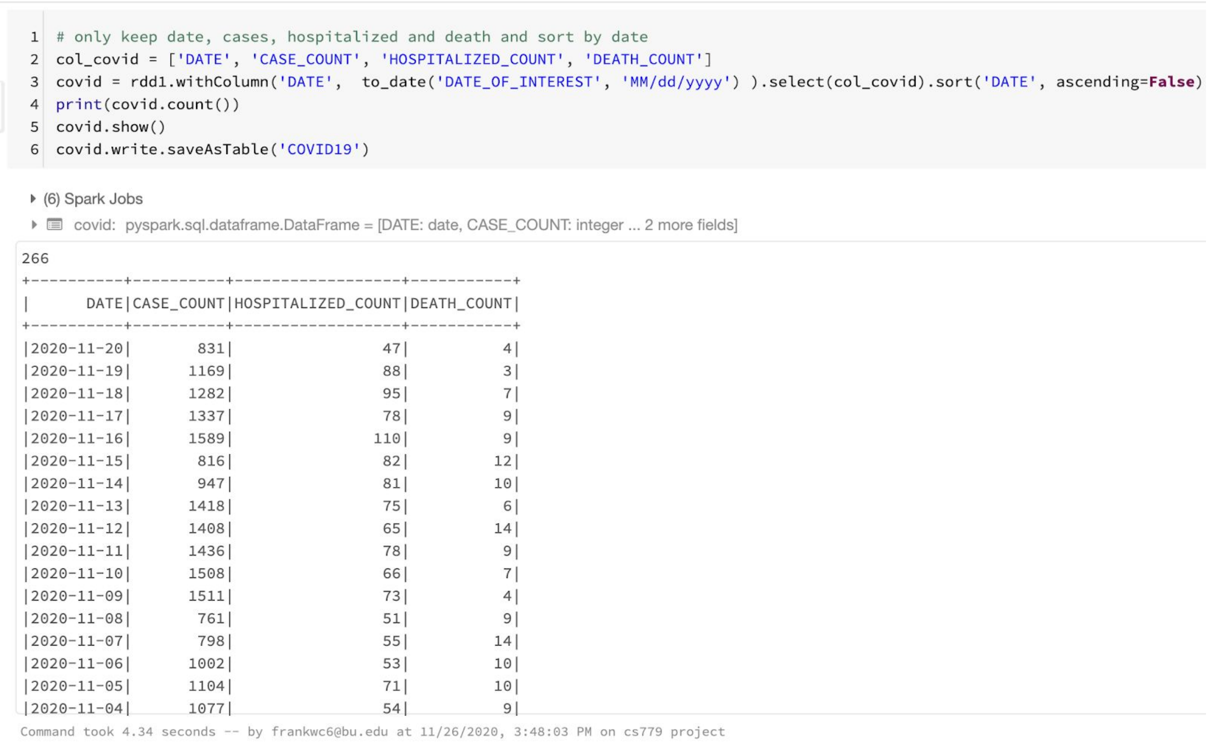
Process Covid19 data

Before transformation:



```
▸ (2) Spark Jobs
▸ rdd1:  pyspark.sql.dataframe.DataFrame = [DATE_OF_INTEREST: string, CASE_COUNT: integer ... 37 more fields]
root
 |-- DATE_OF_INTEREST: string (nullable = true)
 |-- CASE_COUNT: integer (nullable = true)
 |-- HOSPITALIZED_COUNT: integer (nullable = true)
 |-- DEATH_COUNT: integer (nullable = true)
 |-- DEATH_COUNT_PROBABLE       : integer (nullable = true)
 |-- CASE_COUNT_7DAY_AVG        : integer (nullable = true)
 |-- HOSP_COUNT_7DAY_AVG: integer (nullable = true)
 |-- DEATH_COUNT_7DAY_AVG: integer (nullable = true)
 |-- BX_CASE_COUNT: integer (nullable = true)
 |-- BX_HOSPITALIZED_COUNT: integer (nullable = true)
 |-- BX_DEATH_COUNT: integer (nullable = true)
 |-- BX_CASE_COUNT_7DAY_AVG: integer (nullable = true)
 |-- BX_HOSPITALIZED_COUNT_7DAY_AVG: integer (nullable = true)
 |-- BX_DEATH_COUNT_7DAY_AVG: integer (nullable = true)
 |-- BK_CASE_COUNT: integer (nullable = true)
 |-- BK_HOSPITALIZED_COUNT: integer (nullable = true)
 |-- BK_DEATH_COUNT: integer (nullable = true)
 |-- BK_CASE_COUNT_7DAY_AVG: integer (nullable = true)
 |-- BK_HOSPITALIZED_COUNT_7DAY_AVG: integer (nullable = true)
 |-- BK_DEATH_COUNT_7DAY_AVG: integer (nullable = true)
Command took 0.91 seconds -- by frankwc6@bu.edu at 11/26/2020, 3:40:41 PM on cs779 project
```

After transformation:



```
# only keep date, cases, hospitalized and death and sort by date
col_covid = ['DATE', 'CASE_COUNT', 'HOSPITALIZED_COUNT', 'DEATH_COUNT']
covid = rdd1.withColumn('DATE',  to_date('DATE_OF_INTEREST', 'MM/dd/yyyy') ).select(col_covid).sort('DATE', ascending=False)
print(covid.count())
covid.show()
covid.write.saveAsTable('COVID19')
```

covid: pyspark.sql.dataframe.DataFrame = [DATE: date, CASE_COUNT: integer ... 2 more fields]

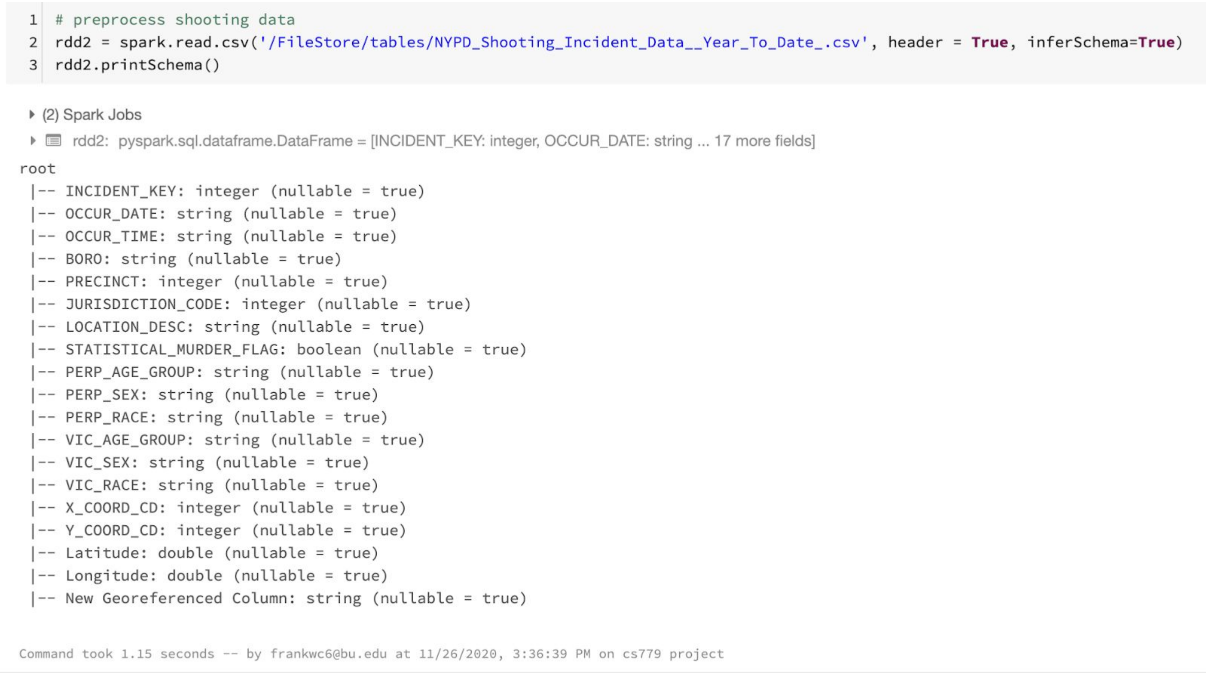
```
266
+----------+----------+------------------+-----------+
|      DATE|CASE_COUNT|HOSPITALIZED_COUNT|DEATH_COUNT|
+----------+----------+------------------+-----------+
|2020-11-20|       831|                47|          4|
|2020-11-19|      1169|                88|          3|
|2020-11-18|      1282|                95|          7|
|2020-11-17|      1337|                78|          9|
|2020-11-16|      1589|               110|          9|
|2020-11-15|       816|                82|         12|
|2020-11-14|       947|                81|         10|
|2020-11-13|      1418|                75|          6|
|2020-11-12|      1408|                65|         14|
|2020-11-11|      1436|                78|          9|
|2020-11-10|      1508|                66|          7|
|2020-11-09|      1511|                73|          4|
|2020-11-08|       761|                51|          9|
|2020-11-07|       798|                55|         14|
|2020-11-06|      1002|                53|         10|
|2020-11-05|      1104|                71|         10|
|2020-11-04|      1077|                54|          9|
```

Command took 4.34 seconds -- by frankwc6@bu.edu at 11/26/2020, 3:48:03 PM on cs779 project

1. Extracted needed attributes.
2. Sort data by date.
3. Save as a table “Covid19”.

Process Shooting Incidents data

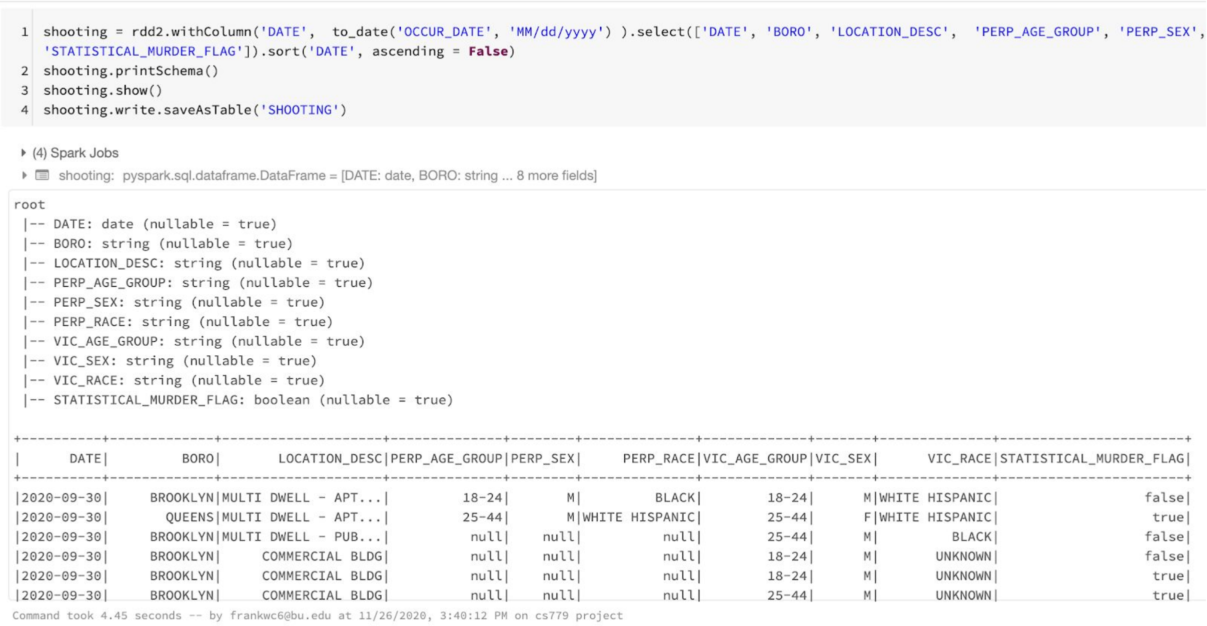
Before:



```
# preprocess shooting data
rdd2 = spark.read.csv('/FileStore/tables/NYPD_Shooting_Incident_Data__Year_To_Date_.csv', header = True, inferSchema=True)
rdd2.printSchema()
```

```
▸ (2) Spark Jobs
▸ rdd2:  pyspark.sql.dataframe.DataFrame = [INCIDENT_KEY: integer, OCCUR_DATE: string ... 17 more fields]
root
 |-- INCIDENT_KEY: integer (nullable = true)
 |-- OCCUR_DATE: string (nullable = true)
 |-- OCCUR_TIME: string (nullable = true)
 |-- BORO: string (nullable = true)
 |-- PRECINCT: integer (nullable = true)
 |-- JURISDICTION_CODE: integer (nullable = true)
 |-- LOCATION_DESC: string (nullable = true)
 |-- STATISTICAL_MURDER_FLAG: boolean (nullable = true)
 |-- PERP_AGE_GROUP: string (nullable = true)
 |-- PERP_SEX: string (nullable = true)
 |-- PERP_RACE: string (nullable = true)
 |-- VIC_AGE_GROUP: string (nullable = true)
 |-- VIC_SEX: string (nullable = true)
 |-- VIC_RACE: string (nullable = true)
 |-- X_COORD_CD: integer (nullable = true)
 |-- Y_COORD_CD: integer (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
 |-- New Georeferenced Column: string (nullable = true)

Command took 1.15 seconds -- by frankwc6@bu.edu at 11/26/2020, 3:36:39 PM on cs779 project
```

After:



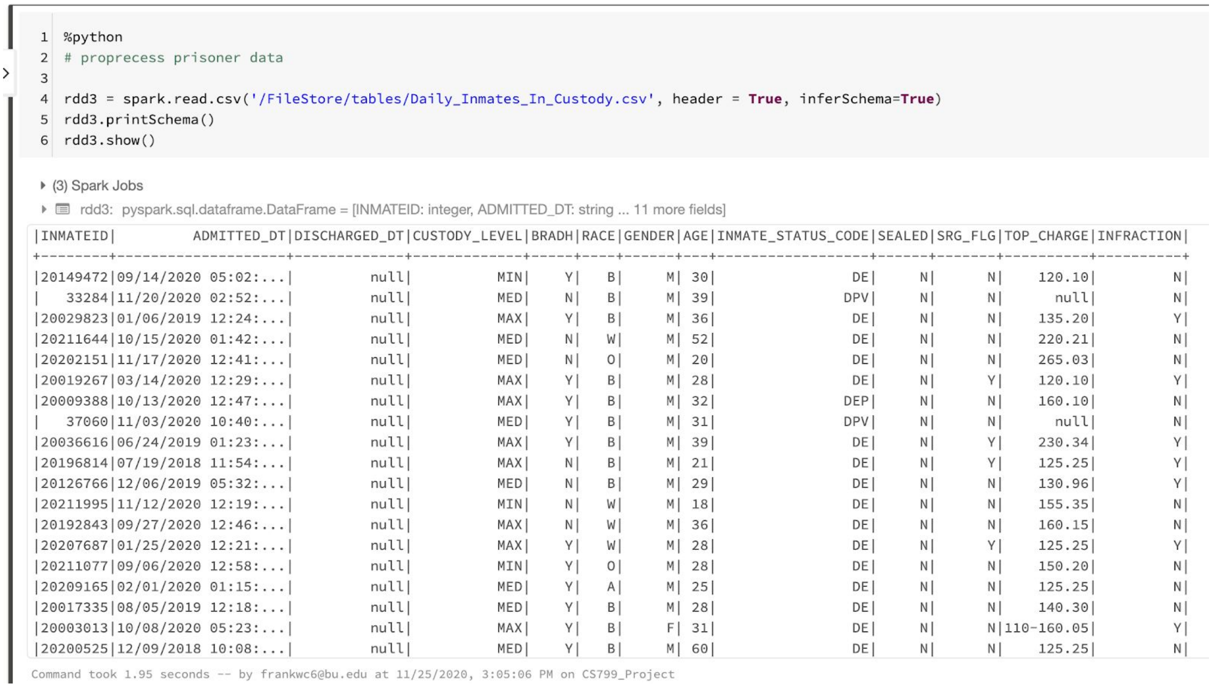
```
shooting = rdd2.withColumn('DATE',  to_date('OCCUR_DATE', 'MM/dd/yyyy') ).select(['DATE', 'BORO', 'LOCATION_DESC',  'PERP_AGE_GROUP', 'PERP_SEX',
'STATISTICAL_MURDER_FLAG']).sort('DATE', ascending = False)
shooting.printSchema()
shooting.show()
shooting.write.saveAsTable('SHOOTING')
```

```
▸ (4) Spark Jobs
▸ shooting:  pyspark.sql.dataframe.DataFrame = [DATE: date, BORO: string ... 8 more fields]
root
 |-- DATE: date (nullable = true)
 |-- BORO: string (nullable = true)
 |-- LOCATION_DESC: string (nullable = true)
 |-- PERP_AGE_GROUP: string (nullable = true)
 |-- PERP_SEX: string (nullable = true)
 |-- PERP_RACE: string (nullable = true)
 |-- VIC_AGE_GROUP: string (nullable = true)
 |-- VIC_SEX: string (nullable = true)
 |-- VIC_RACE: string (nullable = true)
 |-- STATISTICAL_MURDER_FLAG: boolean (nullable = true)

+----------+------------+--------------------+--------------+--------+--------------+-------------+-------+--------------+-----------------------+
|      DATE|        BORO|       LOCATION_DESC|PERP_AGE_GROUP|PERP_SEX|     PERP_RACE|VIC_AGE_GROUP|VIC_SEX|      VIC_RACE|STATISTICAL_MURDER_FLAG|
+----------+------------+--------------------+--------------+--------+--------------+-------------+-------+--------------+-----------------------+
|2020-09-30|    BROOKLYN|MULTI DWELL - APT...|         18-24|       M|         BLACK|        18-24|      M|WHITE HISPANIC|                  false|
|2020-09-30|      QUEENS|MULTI DWELL - APT...|         25-44|       M|WHITE HISPANIC|        25-44|      F|WHITE HISPANIC|                   true|
|2020-09-30|    BROOKLYN|MULTI DWELL - PUB...|          null|    null|          null|        25-44|      M|         BLACK|                  false|
|2020-09-30|    BROOKLYN|     COMMERCIAL BLDG|          null|    null|          null|        18-24|      M|       UNKNOWN|                  false|
|2020-09-30|    BROOKLYN|     COMMERCIAL BLDG|          null|    null|          null|        18-24|      M|       UNKNOWN|                   true|
|2020-09-30|    BROOKLYN|     COMMERCIAL BLDG|          null|    null|          null|        25-44|      M|       UNKNOWN|                   true|
Command took 4.45 seconds -- by frankwc6@bu.edu at 11/26/2020, 3:40:12 PM on cs779 project
```

1. Attribute selection.
2. Convert date type and sort by date.
3. Save as a table “Shooting”.

Process Prisoner data

Before:



```
%python
# proprecess prisoner data

rdd3 = spark.read.csv('/FileStore/tables/Daily_Inmates_In_Custody.csv', header = True, inferSchema=True)
rdd3.printSchema()
rdd3.show()
```

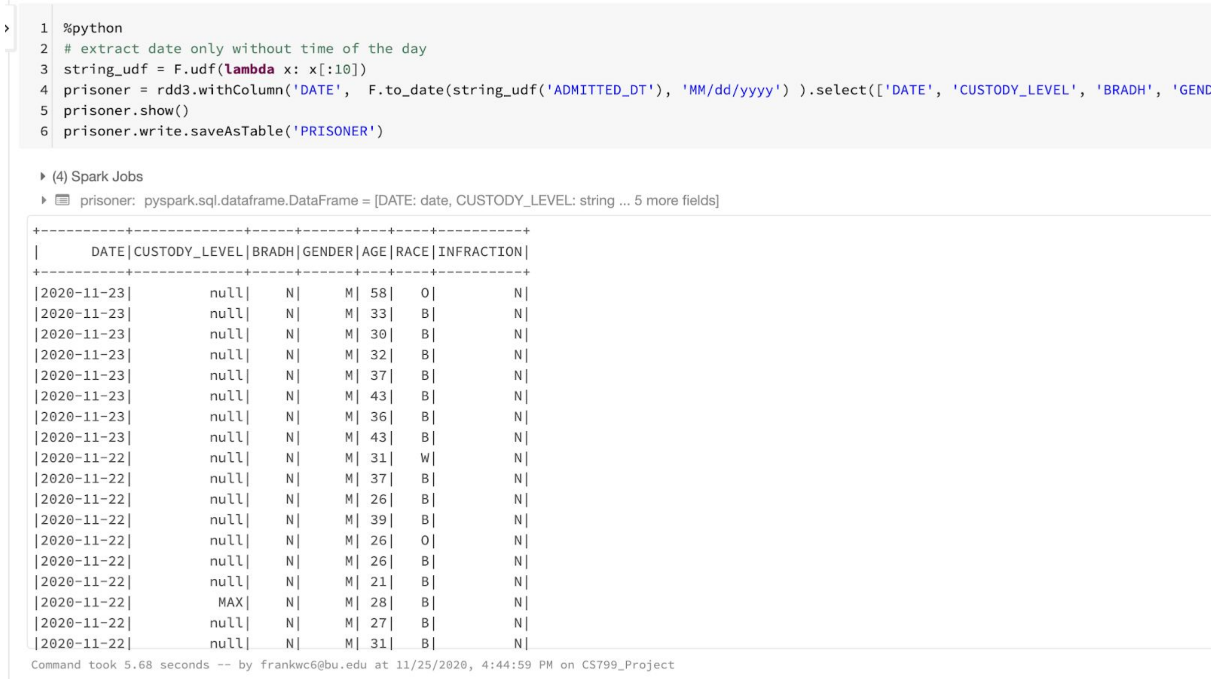
(3) Spark Jobs

rdd3: pyspark.sql.dataframe.DataFrame = [INMATEID: integer, ADMITTED_DT: string ... 11 more fields]

```
|INMATEID|         ADMITTED_DT|DISCHARGED_DT|CUSTODY_LEVEL|BRADH|RACE|GENDER|AGE|INMATE_STATUS_CODE|SEALED|SRG_FLG|TOP_CHARGE|INFRACTION|
+--------+--------------------+-------------+-------------+-----+----+------+---+------------------+------+-------+----------+----------+
|20149472|09/14/2020 05:02:...|         null|          MIN|    Y|   B|     M| 30|                DE|     N|      N|    120.10|         N|
|   33284|11/20/2020 02:52:...|         null|          MED|    N|   B|     M| 39|               DPV|     N|      N|      null|         N|
|20029823|01/06/2019 12:24:...|         null|          MAX|    Y|   B|     M| 36|                DE|     N|      N|    135.20|         Y|
|20211644|10/15/2020 01:42:...|         null|          MED|    N|   W|     M| 52|                DE|     N|      N|    220.21|         N|
|20202151|11/17/2020 12:41:...|         null|          MED|    N|   O|     M| 20|                DE|     N|      N|    265.03|         N|
|20019267|03/14/2020 12:29:...|         null|          MAX|    Y|   B|     M| 28|                DE|     N|      Y|    120.10|         Y|
|20009388|10/13/2020 12:47:...|         null|          MAX|    Y|   B|     M| 32|               DEP|     N|      N|    160.10|         N|
|   37060|11/03/2020 10:40:...|         null|          MED|    Y|   B|     M| 31|               DPV|     N|      N|      null|         N|
|20036616|06/24/2019 01:23:...|         null|          MAX|    Y|   B|     M| 39|                DE|     N|      Y|    230.34|         Y|
|20196814|07/19/2018 11:54:...|         null|          MAX|    N|   B|     M| 21|                DE|     N|      Y|    125.25|         Y|
|20126766|12/06/2019 05:32:...|         null|          MED|    N|   B|     M| 29|                DE|     N|      N|    130.96|         Y|
|20211995|11/12/2020 12:19:...|         null|          MIN|    N|   W|     M| 18|                DE|     N|      N|    155.35|         N|
|20192843|09/27/2020 12:46:...|         null|          MAX|    N|   W|     M| 36|                DE|     N|      N|    160.15|         N|
|20207687|01/25/2020 12:21:...|         null|          MAX|    Y|   W|     M| 28|                DE|     N|      Y|    125.25|         Y|
|20211077|09/06/2020 12:58:...|         null|          MIN|    Y|   O|     M| 28|                DE|     N|      N|    150.20|         N|
|20209165|02/01/2020 01:15:...|         null|          MED|    Y|   A|     M| 25|                DE|     N|      N|    125.25|         N|
|20017335|08/05/2019 12:18:...|         null|          MED|    Y|   B|     M| 28|                DE|     N|      N|    140.30|         N|
|20003013|10/08/2020 05:23:...|         null|          MAX|    Y|   B|     F| 31|                DE|     N|      N|110-160.05|         Y|
|20200525|12/09/2018 10:08:...|         null|          MED|    Y|   B|     M| 60|                DE|     N|      N|    125.25|         N|
```

Command took 1.95 seconds -- by frankwc6@bu.edu at 11/25/2020, 3:05:06 PM on CS799_Project

After:



```
%python
# extract date only without time of the day
string_udf = F.udf(lambda x: x[:10])
prisoner = rdd3.withColumn('DATE',  F.to_date(string_udf('ADMITTED_DT'), 'MM/dd/yyyy') ).select(['DATE', 'CUSTODY_LEVEL', 'BRADH', 'GEND
prisoner.show()
prisoner.write.saveAsTable('PRISONER')
```

```
prisoner:  pyspark.sql.dataframe.DataFrame = [DATE: date, CUSTODY_LEVEL: string ... 5 more fields]
```

```
+----------+-------------+-----+------+---+----+----------+
|      DATE|CUSTODY_LEVEL|BRADH|GENDER|AGE|RACE|INFRACTION|
+----------+-------------+-----+------+---+----+----------+
|2020-11-23|         null|    N|     M| 58|   O|         N|
|2020-11-23|         null|    N|     M| 33|   B|         N|
|2020-11-23|         null|    N|     M| 30|   B|         N|
|2020-11-23|         null|    N|     M| 32|   B|         N|
|2020-11-23|         null|    N|     M| 37|   B|         N|
|2020-11-23|         null|    N|     M| 43|   B|         N|
|2020-11-23|         null|    N|     M| 36|   B|         N|
|2020-11-23|         null|    N|     M| 43|   B|         N|
|2020-11-22|         null|    N|     M| 31|   W|         N|
|2020-11-22|         null|    N|     M| 37|   B|         N|
|2020-11-22|         null|    N|     M| 26|   B|         N|
|2020-11-22|         null|    N|     M| 39|   B|         N|
|2020-11-22|         null|    N|     M| 26|   O|         N|
|2020-11-22|         null|    N|     M| 26|   B|         N|
|2020-11-22|         null|    N|     M| 21|   B|         N|
|2020-11-22|          MAX|    N|     M| 28|   B|         N|
|2020-11-22|         null|    N|     M| 27|   B|         N|
|2020-11-22|         null|    N|     M| 31|   B|         N|
```

```
Command took 5.68 seconds -- by frankwc6@bu.edu at 11/25/2020, 4:44:59 PM on CS799_Project
```

1. Extract date as “yyyy-mm-dd” format without day time.

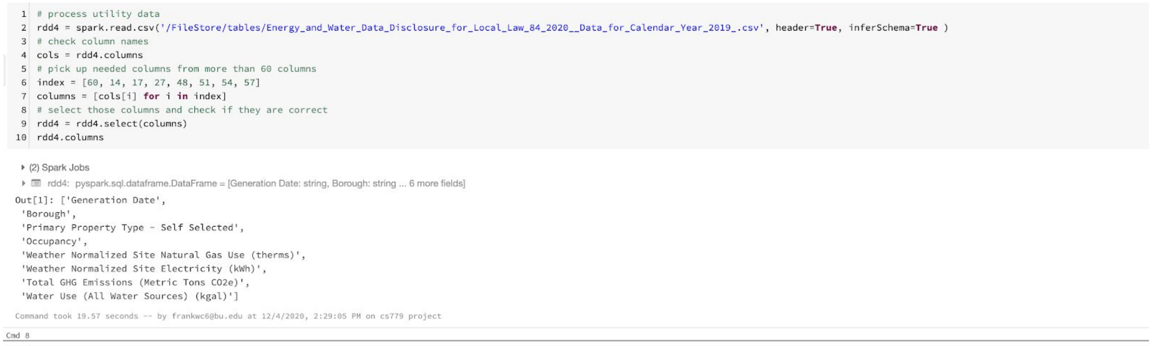
2. Attribute selection.

3. Sort by date and save as table “Prisoner”

Process utility data

Data and metrics on water and energy consumption in buildings over 25,000 ft2. The original dataset has more than 60 attributes, I use only averaged gas, water, electricity consumption, emission and building usage. Numeric attributes were divided by the number of occupants to get the means of those columns.

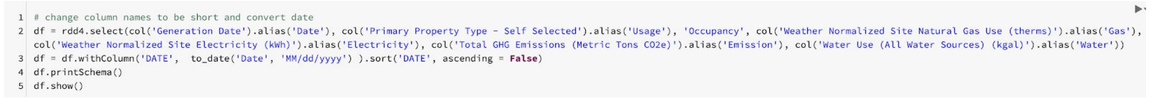
Code and the final table:



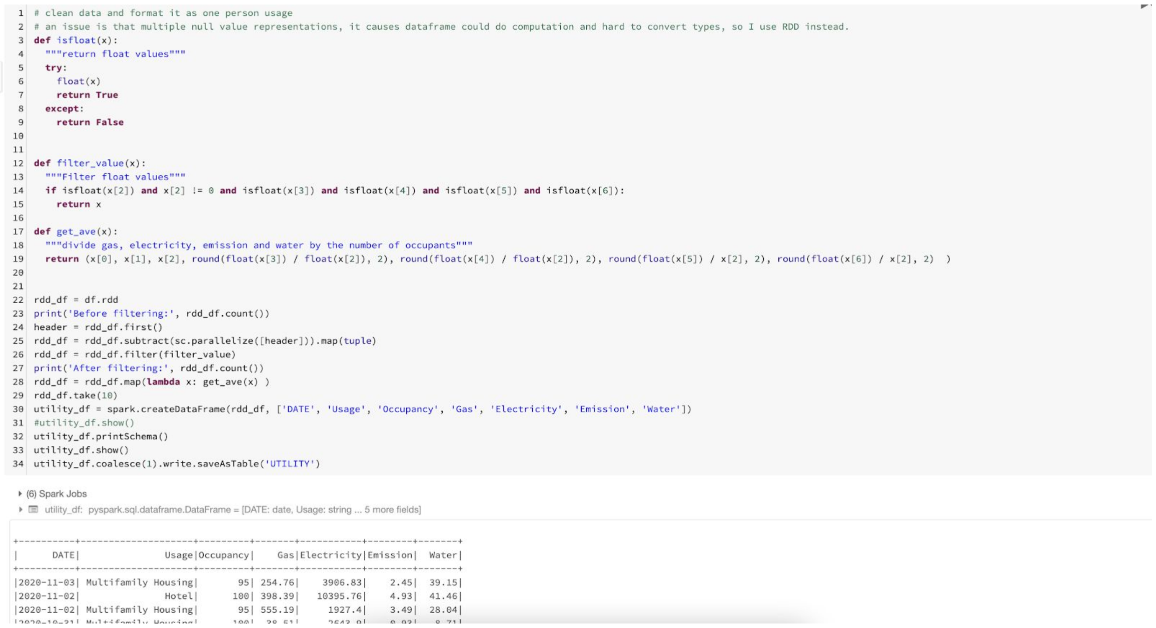
```
# process utility data
rdd4 = spark.read.csv('/FileStore/tables/Energy_and_Water_Data_Disclosure_for_Local_Law_84_2020__Data_for_Calendar_Year_2019_.csv', header=True, inferSchema=True )
# check column names
cols = rdd4.columns
# pick up needed columns from more than 60 columns
index = [60, 14, 17, 27, 48, 51, 54, 57]
columns = [cols[i] for i in index]
# select those columns and check if they are correct
rdd4 = rdd4.select(columns)
rdd4.columns
```

```
▸ (2) Spark Jobs
▸ rdd4:  pyspark.sql.dataframe.DataFrame = [Generation Date: string, Borough: string ... 6 more fields]
Out[1]: ['Generation Date',
 'Borough',
 'Primary Property Type - Self Selected',
 'Occupancy',
 'Weather Normalized Site Natural Gas Use (therms)',
 'Weather Normalized Site Electricity (kWh)',
 'Total GHG Emissions (Metric Tons CO2e)',
 'Water Use (All Water Sources) (kgal)']

Command took 19.57 seconds -- by frankwc6@bu.edu at 12/4/2020, 2:29:05 PM on cs779 project
```

Cmd 8



```
# change column names to be short and convert date
df = rdd4.select(col('Generation Date').alias('Date'), col('Primary Property Type - Self Selected').alias('Usage'), 'Occupancy', col('Weather Normalized Site Natural Gas Use (therms)').alias('Gas'),
col('Weather Normalized Site Electricity (kWh)').alias('Electricity'), col('Total GHG Emissions (Metric Tons CO2e)').alias('Emission'), col('Water Use (All Water Sources) (kgal)').alias('Water'))
df = df.withColumn('DATE',  to_date('Date', 'MM/dd/yyyy') ).sort('DATE', ascending = False)
df.printSchema()
df.show()
```



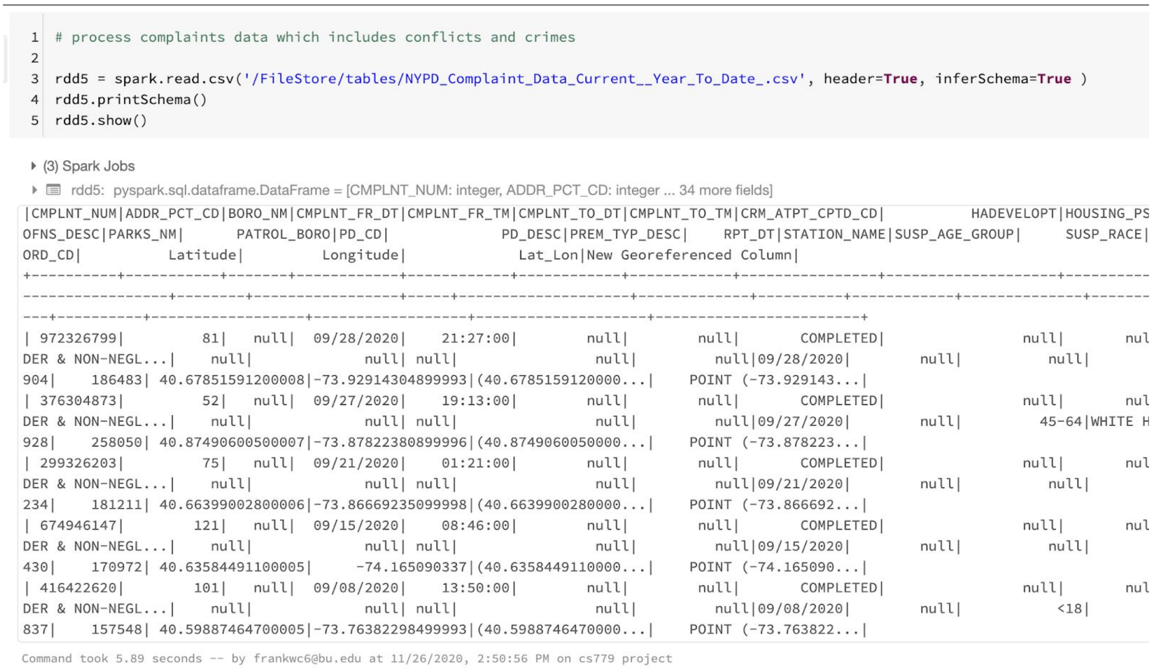
```
# clean data and format it as one person usage
# an issue is that multiple null value representations, it causes dataframe could do computation and hard to convert types, so I use RDD instead.
def isfloat(x):
  """return float values"""
  try:
    float(x)
    return True
  except:
    return False


def filter_value(x):
  """Filter float values"""
  if isfloat(x[2]) and x[2] != 0 and isfloat(x[3]) and isfloat(x[4]) and isfloat(x[5]) and isfloat(x[6]):
    return x

def get_ave(x):
  """divide gas, electricity, emission and water by the number of occupants"""
  return (x[0], x[1], x[2], round(float(x[3]) / float(x[2]), 2), round(float(x[4]) / float(x[2]), 2), round(float(x[5]) / x[2], 2), round(float(x[6]) / x[2], 2)  )


rdd_df = df.rdd
print('Before filtering:', rdd_df.count())
header = rdd_df.first()
rdd_df = rdd_df.subtract(sc.parallelize([header])).map(tuple)
rdd_df = rdd_df.filter(filter_value)
print('After filtering:', rdd_df.count())
rdd_df = rdd_df.map(lambda x: get_ave(x) )
rdd_df.take(10)
utility_df = spark.createDataFrame(rdd_df, ['DATE', 'Usage', 'Occupancy', 'Gas', 'Electricity', 'Emission', 'Water'])
#utility_df.show()
utility_df.printSchema()
utility_df.show()
utility_df.coalesce(1).write.saveAsTable('UTILITY')
```

```
▸ (6) Spark Jobs
▸ utility_df:  pyspark.sql.dataframe.DataFrame = [DATE: date, Usage: string ... 5 more fields]

+----------+--------------------+---------+-------+-----------+--------+-------+
|      DATE|               Usage|Occupancy|    Gas|Electricity|Emission|  Water|
+----------+--------------------+---------+-------+-----------+--------+-------+
|2020-11-03| Multifamily Housing|       95| 254.76|    3906.83|    2.45|  39.15|
|2020-11-02|               Hotel|      100| 398.39|   10395.76|    4.93|  41.46|
|2020-11-02| Multifamily Housing|       95| 555.19|     1927.4|    3.49|  28.04|
|2020-10-31| Multifamily Housing|      100|  38.51|     2643.9|    0.93|   8.71|
```

Process complaint data

This dataset includes all valid felony, misdemeanor, and violation crimes reported to the New York City Police Department (NYPD).

Before:



```
# process complaints data which includes conflicts and crimes

rdd5 = spark.read.csv('/FileStore/tables/NYPD_Complaint_Data_Current__Year_To_Date_.csv', header=True, inferSchema=True )
rdd5.printSchema()
rdd5.show()
```

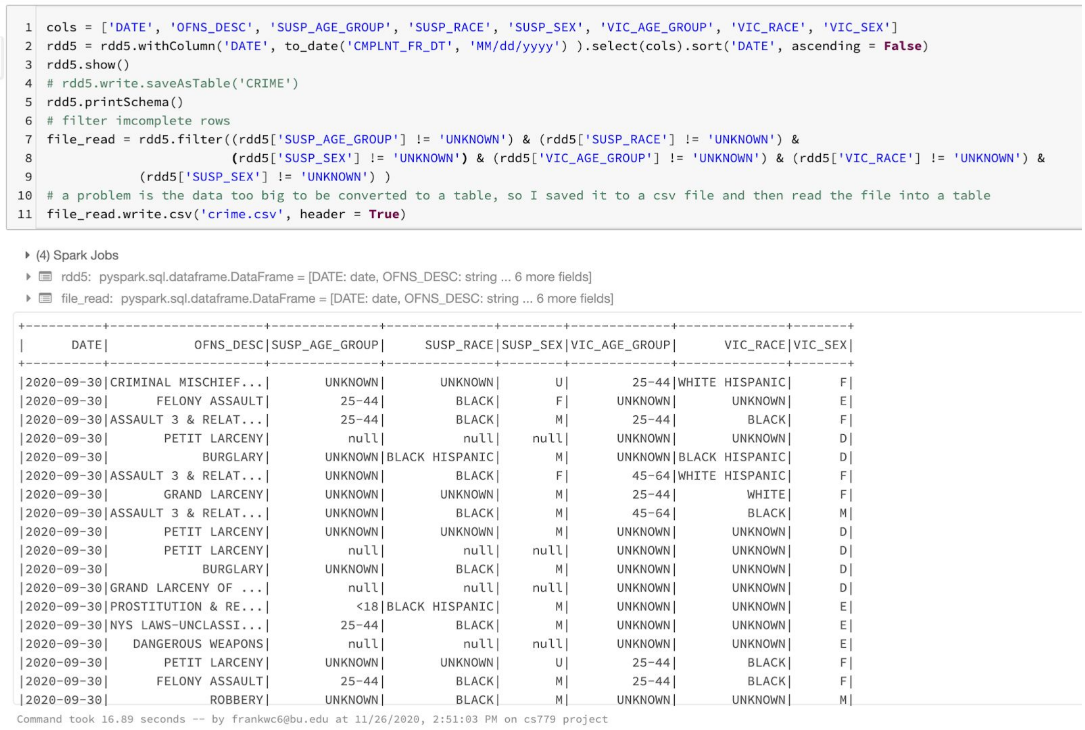
▸ (3) Spark Jobs

▸ rdd5: pyspark.sql.dataframe.DataFrame = [CMPLNT_NUM: integer, ADDR_PCT_CD: integer ... 34 more fields]

```
|CMPLNT_NUM|ADDR_PCT_CD|BORO_NM|CMPLNT_FR_DT|CMPLNT_FR_TM|CMPLNT_TO_DT|CMPLNT_TO_TM|CRM_ATPT_CPTD_CD|          HADEVELOPT|HOUSING_PS
OFNS_DESC|PARKS_NM|       PATROL_BORO|PD_CD|             PD_DESC|PREM_TYP_DESC|    RPT_DT|STATION_NAME|SUSP_AGE_GROUP|      SUSP_RACE|
ORD_CD|          Latitude|         Longitude|              Lat_Lon|New Georeferenced Column|
+----------+-----------+-------+------------+------------+------------+------------+----------------+--------------------+----------
-----------------+--------+-----------------+-----+--------------------+-------------+----------+------------+--------------+-------
---+----------+------------------+------------------+--------------------+------------------------+
| 972326799|         81|   null|  09/28/2020|    21:27:00|        null|        null|       COMPLETED|                null|       nul
DER & NON-NEGL...|    null|             null| null|                null|         null|09/28/2020|        null|          null|
904|    186483| 40.67851591200008|-73.92914304899993|(40.6785159120000...|    POINT (-73.929143...|
| 376304873|         52|   null|  09/27/2020|    19:13:00|        null|        null|       COMPLETED|                null|       nul
DER & NON-NEGL...|    null|             null| null|                null|         null|09/27/2020|        null|         45-64|WHITE H
928|    258050| 40.87490600500007|-73.87822380899996|(40.8749060050000...|    POINT (-73.878223...|
| 299326203|         75|   null|  09/21/2020|    01:21:00|        null|        null|       COMPLETED|                null|       nul
DER & NON-NEGL...|    null|             null| null|                null|         null|09/21/2020|        null|          null|
234|    181211| 40.66399002800006|-73.86669235099998|(40.6639900280000...|    POINT (-73.866692...|
| 674946147|        121|   null|  09/15/2020|    08:46:00|        null|        null|       COMPLETED|                null|       nul
DER & NON-NEGL...|    null|             null| null|                null|         null|09/15/2020|        null|          null|
430|    170972| 40.63584491100005|     -74.165090337|(40.6358449110000...|    POINT (-74.165090...|
| 416422620|        101|   null|  09/08/2020|    13:50:00|        null|        null|       COMPLETED|                null|       nul
DER & NON-NEGL...|    null|             null| null|                null|         null|09/08/2020|        null|           <18|
837|    157548| 40.59887464700005|-73.76382298499993|(40.5988746470000...|    POINT (-73.763822...|
```

Command took 5.89 seconds -- by frankwc6@bu.edu at 11/26/2020, 2:50:56 PM on cs779 project

After:



```
cols = ['DATE', 'OFNS_DESC', 'SUSP_AGE_GROUP', 'SUSP_RACE', 'SUSP_SEX', 'VIC_AGE_GROUP', 'VIC_RACE', 'VIC_SEX']
rdd5 = rdd5.withColumn('DATE', to_date('CMPLNT_FR_DT', 'MM/dd/yyyy') ).select(cols).sort('DATE', ascending = False)
rdd5.show()
# rdd5.write.saveAsTable('CRIME')
rdd5.printSchema()
# filter imcomplete rows
file_read = rdd5.filter((rdd5['SUSP_AGE_GROUP'] != 'UNKNOWN') & (rdd5['SUSP_RACE'] != 'UNKNOWN') &
                        (rdd5['SUSP_SEX'] != 'UNKNOWN') & (rdd5['VIC_AGE_GROUP'] != 'UNKNOWN') & (rdd5['VIC_RACE'] != 'UNKNOWN') &
             (rdd5['SUSP_SEX'] != 'UNKNOWN') )
# a problem is the data too big to be converted to a table, so I saved it to a csv file and then read the file into a table
file_read.write.csv('crime.csv', header = True)
```

▸ (4) Spark Jobs

▸ rdd5: pyspark.sql.dataframe.DataFrame = [DATE: date, OFNS_DESC: string ... 6 more fields]

▸ file_read: pyspark.sql.dataframe.DataFrame = [DATE: date, OFNS_DESC: string ... 6 more fields]

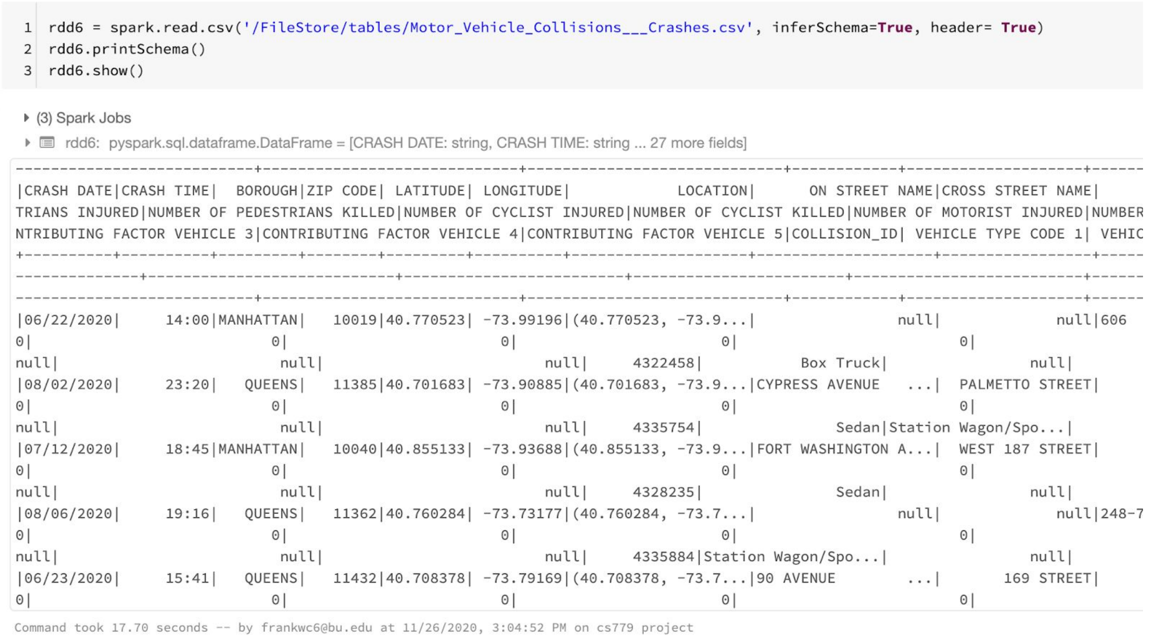
```
+----------+--------------------+--------------+--------------+--------+-------------+--------------+-------+
|      DATE|           OFNS_DESC|SUSP_AGE_GROUP|     SUSP_RACE|SUSP_SEX|VIC_AGE_GROUP|      VIC_RACE|VIC_SEX|
+----------+--------------------+--------------+--------------+--------+-------------+--------------+-------+
|2020-09-30|CRIMINAL MISCHIEF...|       UNKNOWN|       UNKNOWN|       U|        25-44|WHITE HISPANIC|      F|
|2020-09-30|      FELONY ASSAULT|         25-44|         BLACK|       F|      UNKNOWN|       UNKNOWN|      E|
|2020-09-30|ASSAULT 3 & RELAT...|         25-44|         BLACK|       M|        25-44|         BLACK|      F|
|2020-09-30|       PETIT LARCENY|          null|          null|    null|      UNKNOWN|       UNKNOWN|      D|
|2020-09-30|            BURGLARY|       UNKNOWN|BLACK HISPANIC|       M|      UNKNOWN|BLACK HISPANIC|      D|
|2020-09-30|ASSAULT 3 & RELAT...|       UNKNOWN|         BLACK|       F|        45-64|WHITE HISPANIC|      F|
|2020-09-30|       GRAND LARCENY|       UNKNOWN|       UNKNOWN|       M|        25-44|         WHITE|      F|
|2020-09-30|ASSAULT 3 & RELAT...|       UNKNOWN|         BLACK|       M|        45-64|         BLACK|      M|
|2020-09-30|       PETIT LARCENY|       UNKNOWN|       UNKNOWN|       M|      UNKNOWN|       UNKNOWN|      D|
|2020-09-30|       PETIT LARCENY|          null|          null|    null|      UNKNOWN|       UNKNOWN|      D|
|2020-09-30|            BURGLARY|       UNKNOWN|         BLACK|       M|      UNKNOWN|       UNKNOWN|      D|
|2020-09-30|GRAND LARCENY OF ...|          null|          null|    null|      UNKNOWN|       UNKNOWN|      D|
|2020-09-30|PROSTITUTION & RE...|           <18|BLACK HISPANIC|       M|      UNKNOWN|       UNKNOWN|      E|
|2020-09-30|NYS LAWS-UNCLASSI...|         25-44|         BLACK|       M|      UNKNOWN|       UNKNOWN|      E|
|2020-09-30|   DANGEROUS WEAPONS|          null|          null|    null|      UNKNOWN|       UNKNOWN|      E|
|2020-09-30|       PETIT LARCENY|       UNKNOWN|       UNKNOWN|       U|        25-44|         BLACK|      F|
|2020-09-30|      FELONY ASSAULT|         25-44|         BLACK|       M|        25-44|         BLACK|      F|
|2020-09-30|             ROBBERY|       UNKNOWN|         BLACK|       M|      UNKNOWN|       UNKNOWN|      M|
```

Command took 16.89 seconds -- by frankwc6@bu.edu at 11/26/2020, 2:51:03 PM on cs779 project

Process Vehicle Collision data

The Motor Vehicle Collisions crash table contains details on the crash event. Each row represents a crash event. The Motor Vehicle Collisions data tables contain information from all police reported motor vehicle collisions in NYC.

Before:



```
rdd6 = spark.read.csv('/FileStore/tables/Motor_Vehicle_Collisions___Crashes.csv', inferSchema=True, header= True)
rdd6.printSchema()
rdd6.show()
```

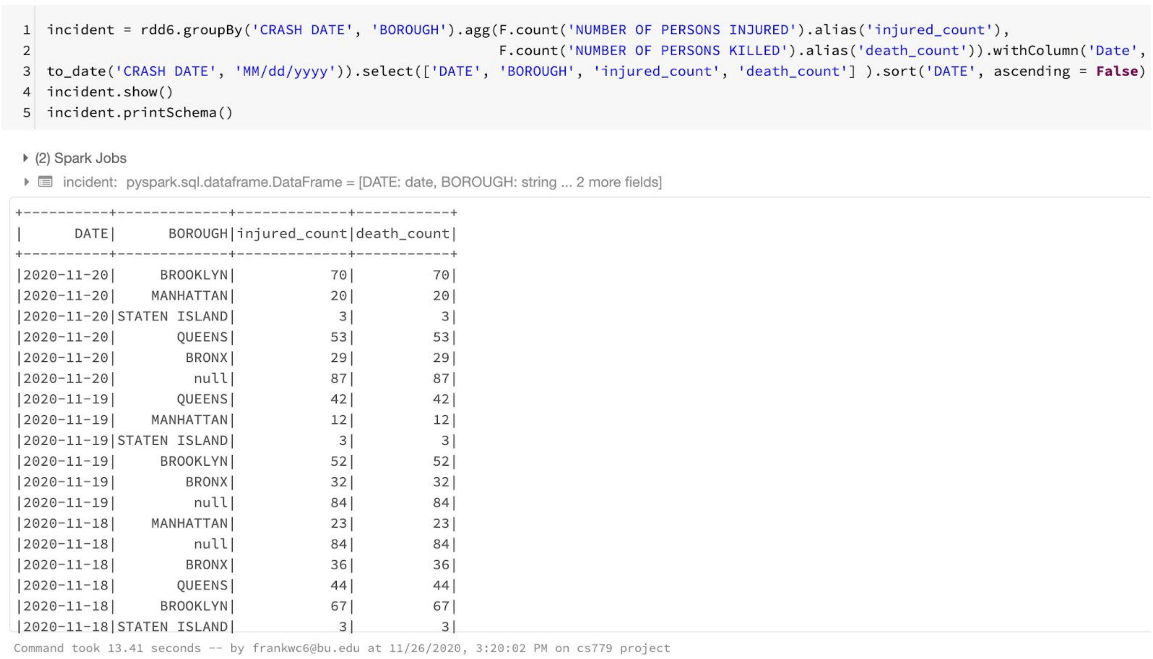
(3) Spark Jobs

rdd6: pyspark.sql.dataframe.DataFrame = [CRASH DATE: string, CRASH TIME: string ... 27 more fields]

```
---------------------------+-----------------------------+-----------------------------+------------+---------------------+------
|CRASH DATE|CRASH TIME|  BOROUGH|ZIP CODE| LATITUDE| LONGITUDE|           LOCATION|      ON STREET NAME|CROSS STREET NAME|
TRIANS INJURED|NUMBER OF PEDESTRIANS KILLED|NUMBER OF CYCLIST INJURED|NUMBER OF CYCLIST KILLED|NUMBER OF MOTORIST INJURED|NUMBER
NTRIBUTING FACTOR VEHICLE 3|CONTRIBUTING FACTOR VEHICLE 4|CONTRIBUTING FACTOR VEHICLE 5|COLLISION_ID| VEHICLE TYPE CODE 1| VEHIC
+----------+----------+---------+--------+---------+----------+--------------------+--------------------+-----------------+-----
--------------+-----------------------------+--------------------------+-------------------------+--------------------------+-----
---------------------------+-----------------------------+-----------------------------+------------+---------------------+------
|06/22/2020|     14:00|MANHATTAN|   10019|40.770523| -73.99196|(40.770523, -73.9...|                null|             null|606
0|                            0|                        0|                        0|                         0|
null|                          null|                         null|     4322458|            Box Truck|                null|
|08/02/2020|     23:20|   QUEENS|   11385|40.701683| -73.90885|(40.701683, -73.9...|CYPRESS AVENUE   ...|  PALMETTO STREET|
0|                            0|                        0|                        0|                         0|
null|                          null|                         null|     4335754|                Sedan|Station Wagon/Spo...|
|07/12/2020|     18:45|MANHATTAN|   10040|40.855133| -73.93688|(40.855133, -73.9...|FORT WASHINGTON A...|  WEST 187 STREET|
0|                            0|                        0|                        0|                         0|
null|                          null|                         null|     4328235|                Sedan|                null|
|08/06/2020|     19:16|   QUEENS|   11362|40.760284| -73.73177|(40.760284, -73.7...|                null|             null|248-7
0|                            0|                        0|                        0|                         0|
null|                          null|                         null|     4335884|Station Wagon/Spo...|                null|
|06/23/2020|     15:41|   QUEENS|   11432|40.708378| -73.79169|(40.708378, -73.7...|90 AVENUE        ...|       169 STREET|
0|                            0|                        0|                        0|                         0|
```

Command took 17.70 seconds -- by frankwc6@bu.edu at 11/26/2020, 3:04:52 PM on cs779 project

After:



```
incident = rdd6.groupBy('CRASH DATE', 'BOROUGH').agg(F.count('NUMBER OF PERSONS INJURED').alias('injured_count'),
                                                     F.count('NUMBER OF PERSONS KILLED').alias('death_count')).withColumn('Date',
to_date('CRASH DATE', 'MM/dd/yyyy')).select(['DATE', 'BOROUGH', 'injured_count', 'death_count'] ).sort('DATE', ascending = False)
incident.show()
incident.printSchema()
```

(2) Spark Jobs

incident: pyspark.sql.dataframe.DataFrame = [DATE: date, BOROUGH: string ... 2 more fields]

```
+----------+-------------+-------------+-----------+
|      DATE|      BOROUGH|injured_count|death_count|
+----------+-------------+-------------+-----------+
|2020-11-20|     BROOKLYN|           70|         70|
|2020-11-20|    MANHATTAN|           20|         20|
|2020-11-20|STATEN ISLAND|            3|          3|
|2020-11-20|       QUEENS|           53|         53|
|2020-11-20|        BRONX|           29|         29|
|2020-11-20|         null|           87|         87|
|2020-11-19|       QUEENS|           42|         42|
|2020-11-19|    MANHATTAN|           12|         12|
|2020-11-19|STATEN ISLAND|            3|          3|
|2020-11-19|     BROOKLYN|           52|         52|
|2020-11-19|        BRONX|           32|         32|
|2020-11-19|         null|           84|         84|
|2020-11-18|    MANHATTAN|           23|         23|
|2020-11-18|         null|           84|         84|
|2020-11-18|        BRONX|           36|         36|
|2020-11-18|       QUEENS|           44|         44|
|2020-11-18|     BROOKLYN|           67|         67|
|2020-11-18|STATEN ISLAND|            3|          3|
```

Command took 13.41 seconds -- by frankwc6@bu.edu at 11/26/2020, 3:20:02 PM on cs779 project

I summarized this dataset by counting the number of injuries and deaths. A finding is injury and death are almost the same, it shows the fatality rate of vehicle collision is close to 100%

Construct Tables

1. Save data frames into tables.

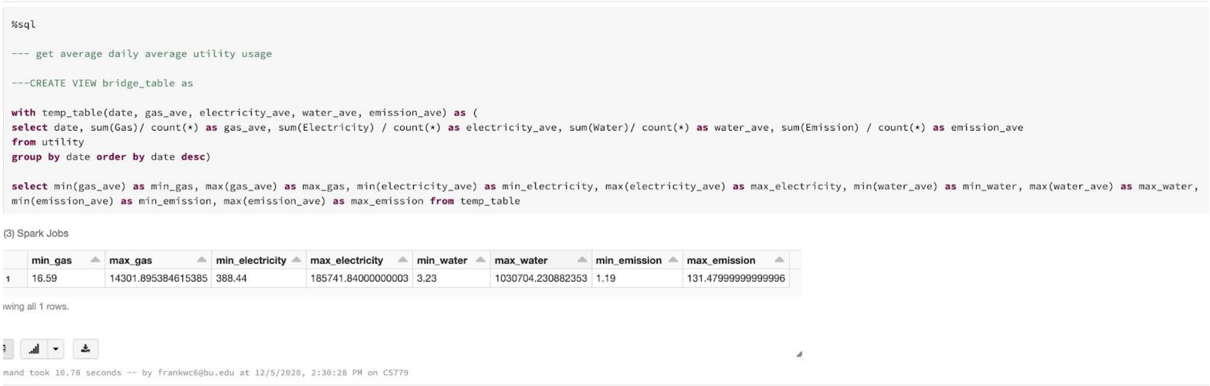
2. Save large data frames into csv files and read files by tables.

3. Summarize tables into views as dimension tables.

4. Load summarized data into a fact table. I’m not sure this can be called to be a fact table because those dimensions only share year, month and day.

5. Do queries. The query pattern is: fact table--- views --- original table. This could improve performance significantly if we want to find out a specific question.

The most complicated view was the utility view. A big problem was gas, water, electricity and emission are not on the same scale. So, I normalized those values by using the function of (value - min) / (max - min). This function transforms all values to around 1. After getting all 1, I scaled them up by multiplying 10000.



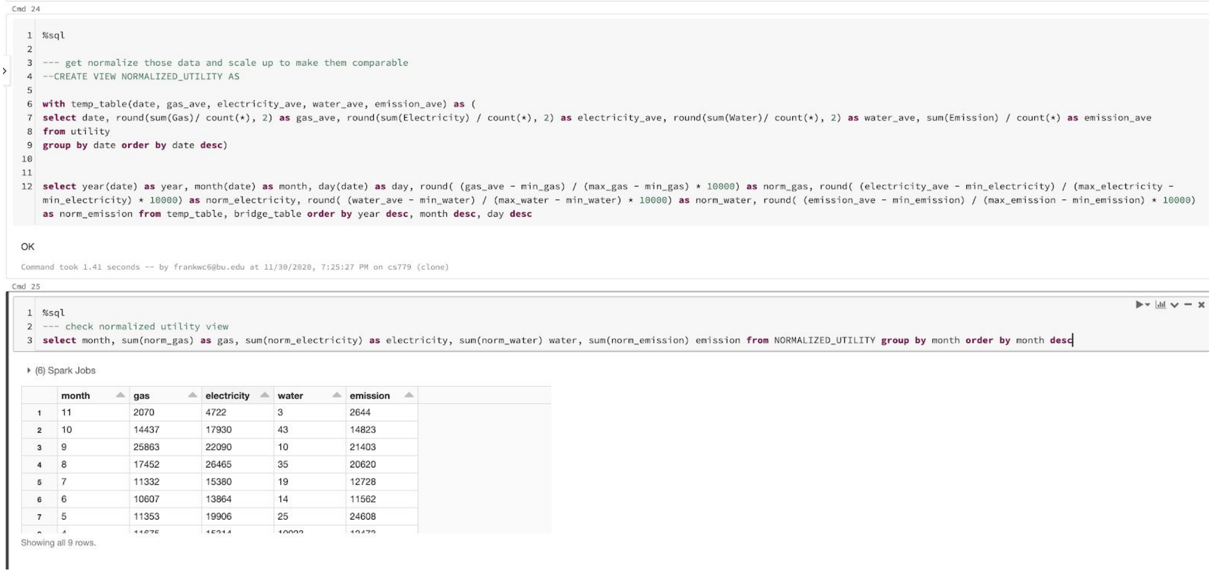
```
%sql

--- get average daily average utility usage

---CREATE VIEW bridge_table as

with temp_table(date, gas_ave, electricity_ave, water_ave, emission_ave) as (
select date, sum(Gas)/ count(*) as gas_ave, sum(Electricity) / count(*) as electricity_ave, sum(Water)/ count(*) as water_ave, sum(Emission) / count(*) as emission_ave
from utility
group by date order by date desc)

select min(gas_ave) as min_gas, max(gas_ave) as max_gas, min(electricity_ave) as min_electricity, max(electricity_ave) as max_electricity, min(water_ave) as min_water, max(water_ave) as max_water,
min(emission_ave) as min_emission, max(emission_ave) as max_emission from temp_table
```

(3) Spark Jobs

	min_gas	max_gas	min_electricity	max_electricity	min_water	max_water	min_emission	max_emission
1	16.59	14301.895384615385	388.44	185741.84000000003	3.23	1030704.230882353	1.19	131.47999999999996

›wing all 1 rows.

```
mand took 10.78 seconds -- by frankwc6@bu.edu at 12/5/2020, 2:30:28 PM on CS779
```



Cmd 24

```
%sql

--- get normalize those data and scale up to make them comparable
--CREATE VIEW NORMALIZED_UTILITY AS

with temp_table(date, gas_ave, electricity_ave, water_ave, emission_ave) as (
select date, round(sum(Gas)/ count(*), 2) as gas_ave, round(sum(Electricity) / count(*), 2) as electricity_ave, round(sum(Water)/ count(*), 2) as water_ave, sum(Emission) / count(*) as emission_ave
from utility
group by date order by date desc)


select year(date) as year, month(date) as month, day(date) as day, round( (gas_ave - min_gas) / (max_gas - min_gas) * 10000) as norm_gas, round( (electricity_ave - min_electricity) / (max_electricity - min_electricity) * 10000) as norm_electricity, round( (water_ave - min_water) / (max_water - min_water) * 10000) as norm_water, round( (emission_ave - min_emission) / (max_emission - min_emission) * 10000) as norm_emission from temp_table, bridge_table order by year desc, month desc, day desc
```

OK

Command took 1.41 seconds -- by frankwc6@bu.edu at 11/30/2020, 7:25:27 PM on cs779 (clone)

Cmd 25

```
%sql
--- check normalized utility view
select month, sum(norm_gas) as gas, sum(norm_electricity) as electricity, sum(norm_water) water, sum(norm_emission) emission from NORMALIZED_UTILITY group by month order by month desc
```

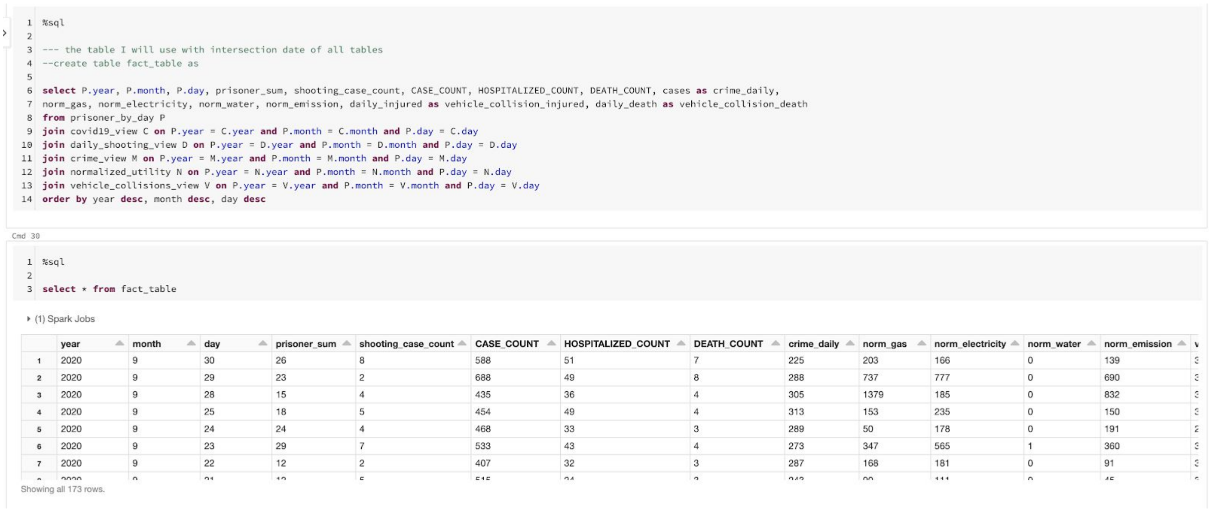
(6) Spark Jobs

	month	gas	electricity	water	emission
1	11	2070	4722	3	2644
2	10	14437	17930	43	14823
3	9	25863	22090	10	21403
4	8	17452	26465	35	20620
5	7	11332	15380	19	12728
6	6	10607	13864	14	11562
7	5	11353	19906	25	24608

Showing all 9 rows.

For other views were only transformation to daily counts of certain attributes.

The last step is to join all tables to a new summary table. The final table only has data from March 2020 to September 2020. As the primary of this project, it focuses on data from the beginning of covid to the end of intersection among all views. After getting the summary table, we can do queries now.



```
%sql

--- the table I will use with intersection date of all tables
--create table fact_table as

select P.year, P.month, P.day, prisoner_sum, shooting_case_count, CASE_COUNT, HOSPITALIZED_COUNT, DEATH_COUNT, cases as crime_daily,
norm_gas, norm_electricity, norm_water, norm_emission, daily_injured as vehicle_collision_injured, daily_death as vehicle_collision_death
from prisoner_by_day P
join covid19_view C on P.year = C.year and P.month = C.month and P.day = C.day
join daily_shooting_view D on P.year = D.year and P.month = D.month and P.day = D.day
join crime_view M on P.year = M.year and P.month = M.month and P.day = M.day
join normalized_utility N on P.year = N.year and P.month = N.month and P.day = N.day
join vehicle_collisions_view V on P.year = V.year and P.month = V.month and P.day = V.day
order by year desc, month desc, day desc
```

Cmd 30

```
%sql

select * from fact_table
```

(1) Spark Jobs

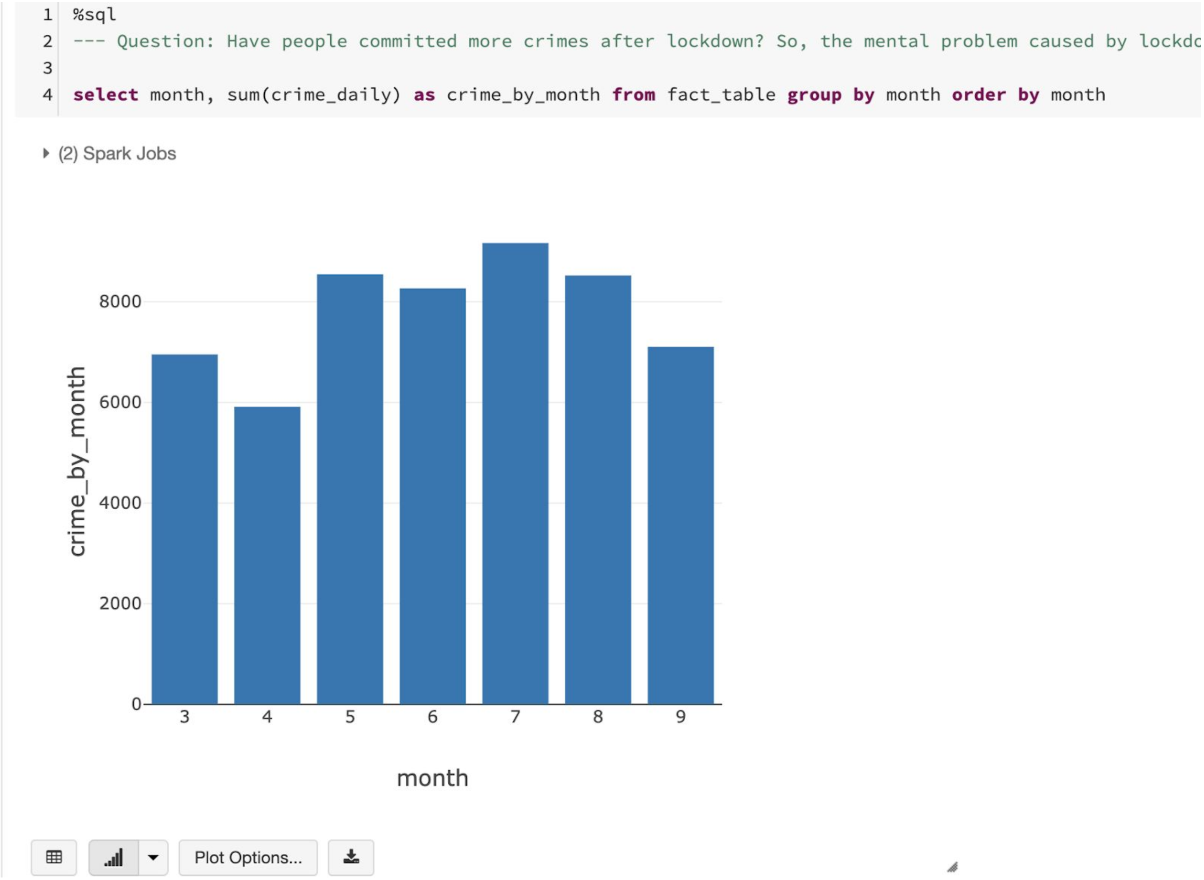
	year	month	day	prisoner_sum	shooting_case_count	CASE_COUNT	HOSPITALIZED_COUNT	DEATH_COUNT	crime_daily	norm_gas	norm_electricity	norm_water	norm_emission
1	2020	9	30	26	8	588	51	7	225	203	166	0	139
2	2020	9	29	23	2	688	49	8	288	737	777	0	690
3	2020	9	28	15	4	435	36	4	305	1379	185	0	832
4	2020	9	25	18	5	454	49	4	313	153	235	0	150
5	2020	9	24	24	4	468	33	3	289	50	178	0	191
6	2020	9	23	29	7	533	43	4	273	347	565	1	360
7	2020	9	22	12	2	407	32	3	287	168	181	0	91

Showing all 173 rows.

Questions to explore

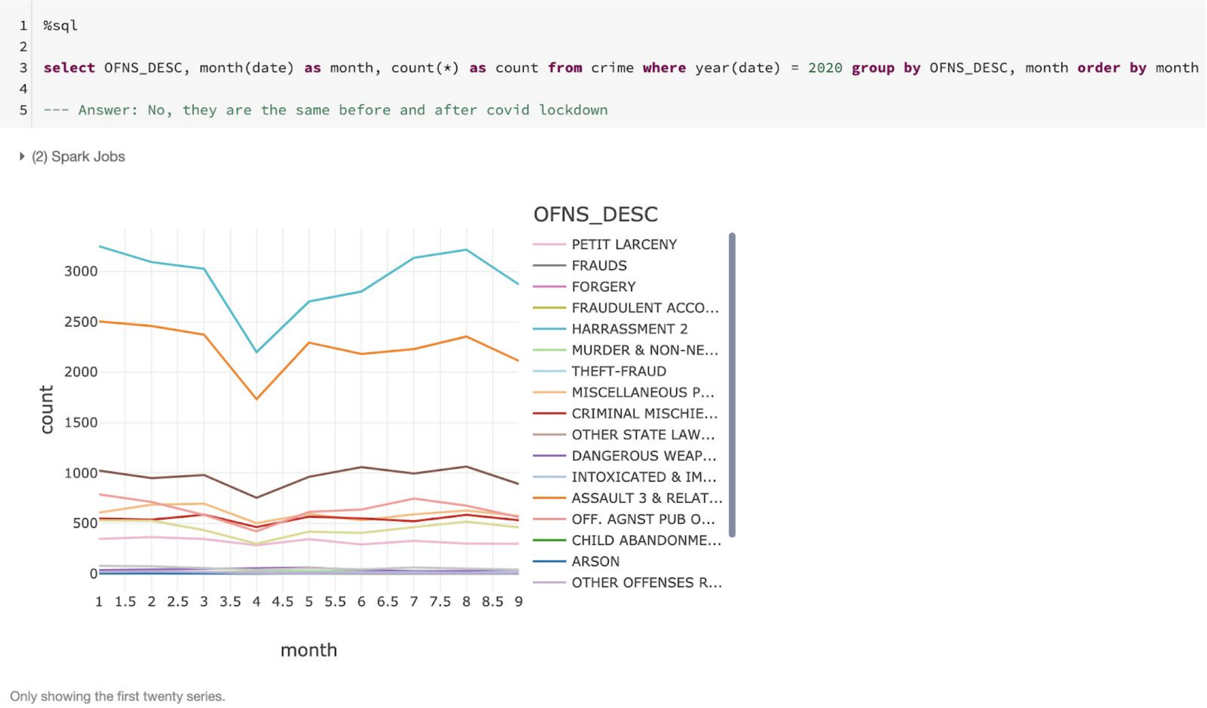
1. Have people committed more crimes after lockdown? So, the mental problem caused by lockdown made public safety worse?

We first see the crime rate by month in 2020 from the summary table. It shows a decline from March to April and then bounced back in June. This change shows the decline of crime during lockdown but did not continue after reopening the country. Now, we need to know if covid19 changed the crime by a social impact but not the lockdown time.



```
%sql
--- Question: Have people committed more crimes after lockdown? So, the mental problem caused by lockd

select month, sum(crime_daily) as crime_by_month from fact_table group by month order by month
```

Let’s check if any category of crime has been changed by covid19.



```
%sql

select OFNS_DESC, month(date) as month, count(*) as count from crime where year(date) = 2020 group by OFNS_DESC, month order by month

--- Answer: No, they are the same before and after covid lockdown
```

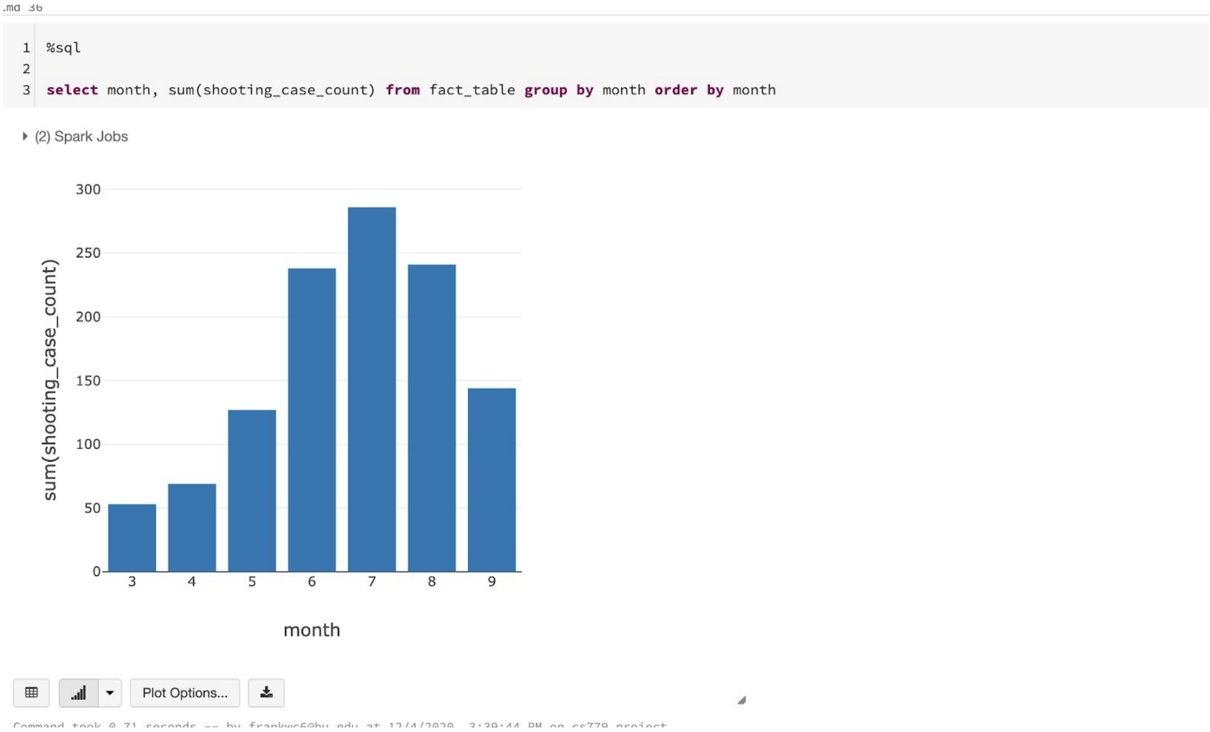
▸ (2) Spark Jobs

Only showing the first twenty series.

The plot shows almost the same count before and after lockdown. So, we can conclude that Covid19 did not change the crime rate in New York city.

2. If the crime rate didn't change, what about shooting cases?

A thing should be noticed is the crime dataset exclude shooting cases, they are general cases. So, I will keep digging into the question of have covid19 made more shooting cases. The bar plot shows a significant increase in shooting cases after April 2020.

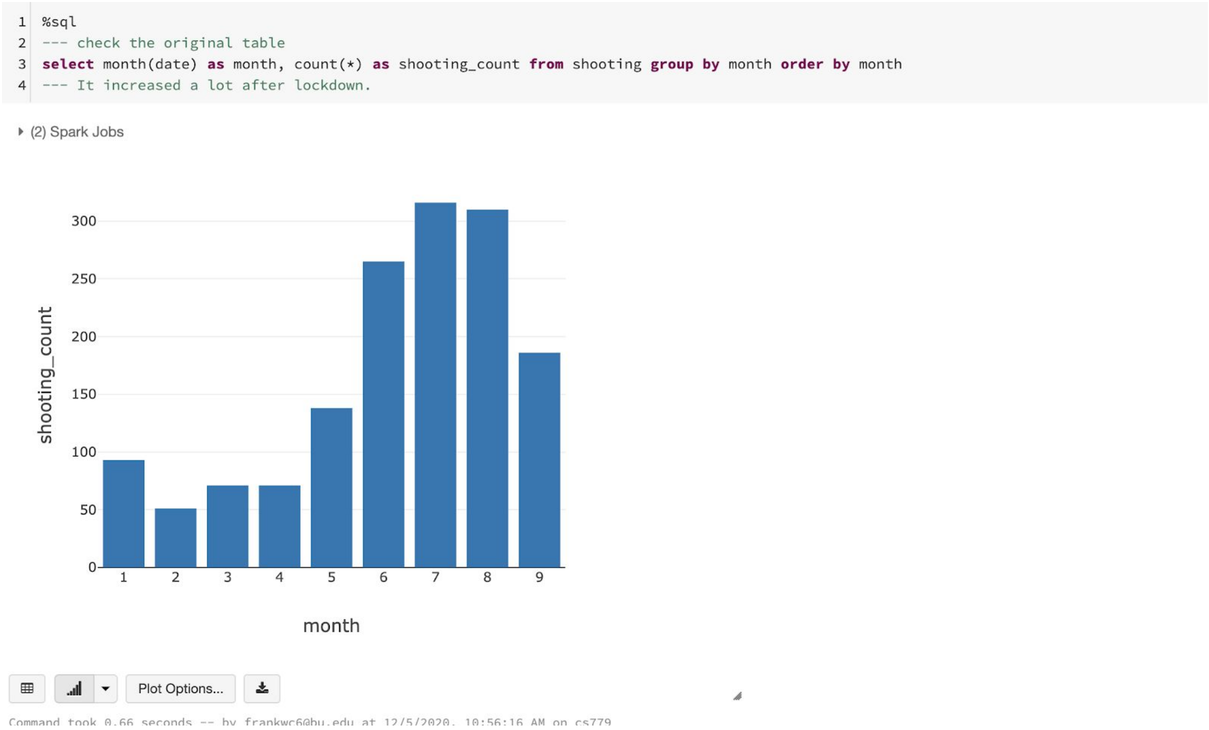

.md 36

```
%sql

select month, sum(shooting_case_count) from fact_table group by month order by month
```

(2) Spark Jobs

Command took 0.71 seconds -- by frankwc6@bu.edu at 12/4/2020, 3:39:44 PM on cs779 project

To confirm the result of increasing shooting cases after lockdown, we check the original shooting cases table, which has data from January to September. It showed the same trend of shooting cases increasing after lockdown.



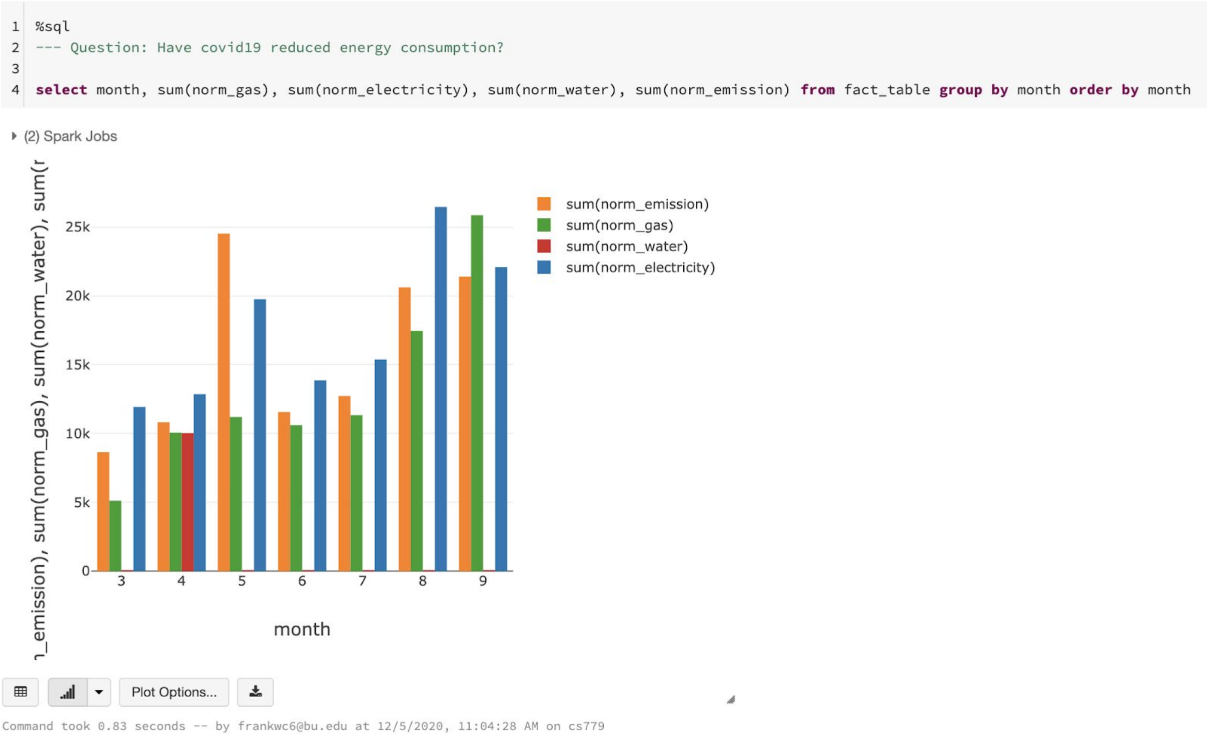
```
%sql
--- check the original table
select month(date) as month, count(*) as shooting_count from shooting group by month order by month
--- It increased a lot after lockdown.
```

(2) Spark Jobs

Command took 0.66 seconds -- by frankwc6@bu.edu at 12/5/2020, 10:56:16 AM on cs779

But January was affected by covid19. Since we don’t have historic data of shooting cases, we can’t conclude that covid19 made shooting cases more by causing mental problems. There might be a seasonal reason or data integrity issue.

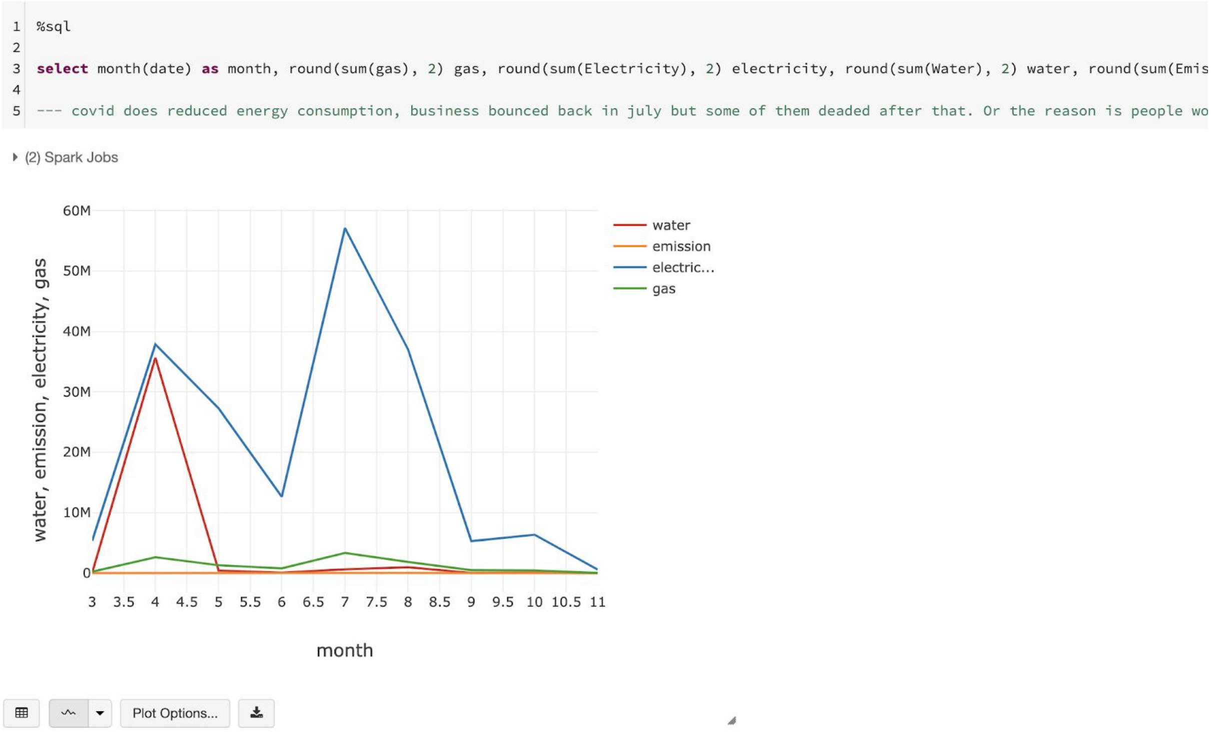
3. Has covid19 reduced energy consumption?



```
%sql
--- Question: Have covid19 reduced energy consumption?

select month, sum(norm_gas), sum(norm_electricity), sum(norm_water), sum(norm_emission) from fact_table group by month order by month
```

Command took 0.83 seconds -- by frankwc6@bu.edu at 12/5/2020, 11:04:28 AM on cs779

We can see a drop in March and April and then a recovery from May, even more consumption after lockdown. For this result, air conditionings were widely used in the summer, which are energy-consuming. Especially we use data from buildings with big size, they are mostly office buildings or shopping malls. Let’s check the whole year.



```
%sql

select month(date) as month, round(sum(gas), 2) gas, round(sum(Electricity), 2) electricity, round(sum(Water), 2) water, round(sum(Emis

--- covid does reduced energy consumption, business bounced back in july but some of them deaded after that. Or the reason is people wo
```

The assumption is true, energy usage dropped after summer. We can’t conclude the drop of energy usage was caused by covid19 because people have been working at home since March. I need to get more data for the past years.

Challenges

1. Figuring out advantages and disadvantages of spark dataframe and SQL both, the data transformation process was based on the structure of tables. The tables serve for query use.

2. Transforming data is not as easy as cleaning data. I had to dissect query plans to the data needed. To make data comparable, I normalized the utility table.

3. Query plans could be hieratical. For example, I want to know whether the total crime cases increased and which one increased during covid-19. I should go to the fact table to get

overall statistics by month and go to the summary crime view to find categories. For some questions, we need to go back to the original tables to find relative attributes, such as stricts, location descriptions and more information about criminals.

Conclusion

1. Spark is designed like a distribution database, data was stored on multiple nodes with replicas. It has more flexibility for data manipulation by using dataframes but less efficient for doing queries.It does not as fast as querying a table.

2. A relational database could keep data consistent. This made queries much easier and prevented data from getting corrupted and easy to connect data.

3. To organize data and extract intelligence is beyond simple processing. This is one of the reasons why designing a data warehouse is difficult. Because it should be designed to serve analysis and it will be hard to make changes once massive data is loaded into a data warehouse.