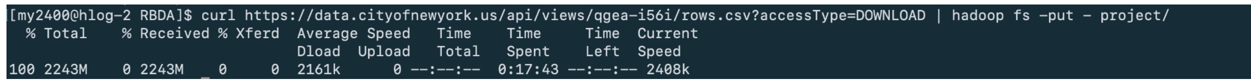
Data Source:

1. The dataset I used in this project is New York City Police Department (NYPD) Historic Complaint Data. This dataset includes all valid felony, misdemeanor, and violation crimes reported to the NYPD from 2006 to the end of 2019. [1]
2. The dataset is 2.243GB, containing 7.38M rows and 35 columns.
3. The following table describes the fields that are useful in this project. The description of all fields can be found at NYPD Complaint Incident Level Data Footnotes [2].

Field Name	**Description**
CMPLNT_NUM (int)	Randomly generated persistent ID for each complaint
KEY_CD (int)	"Key Code": Offense Classification Code (3 digits)
OFNS_DESC	Description of offense corresponding with key code
CMPLNT_FR_DT	Date of occurrence for the reported event
Latitude	Global Latitude of Location where Incident Occurred
Longitude	Global Longitude of Location where Incident Occurred

4. I used curl to download the dataset and redirected the output to HDFS. Downloading the dataset takes around 18 minutes. I renamed the output as project/crime.csv in HDFS. The shell command and output is shown in the following screenshot.



```
[my2400@hlog-2 RBDA]$ curl https://data.cityofnewyork.us/api/views/qgea-i56i/rows.csv?accessType=DOWNLOAD | hadoop fs -put - project/
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 2243M    0 2243M    0     0  2161k      0 --:--:--  0:17:43 --:--:-- 2408k
```

5. In the code submission, small-crime.csv is a snippet of the dataset. It contains the first 10 rows of the dataset, which is a great resource for me to test the MapReduce code in a short debug cycle.

[WIP]What I did for the data - Clean:

1. In the cleaner mapper (GetZipcodeAndCleanMapper.java), I tokenized the input lines by comma (except for the enclosed string) and extracted the complaints_id, report_date, offense_key_code, offense_description, latitude and longitude. All the other columns are ignored.
2. If any of the fields mentioned in step 1 is empty, I treat the row as malformatted and ignore it. If latitude and longitude are not numeric, the row will be ignored too.
3. I also converted latitude and longitude into zip code for each line in the cleaner mapper.
    a. I maintained an array called geo_zipcode in the mapper class. Each string in the array contains a NYC zip code and its associated latitude and longitude. I got the information from an open database collected from the US census [3].
    b. To convert the geolocation of each incident to its zip code, the mapper searches the geolocation that is closest to the incident's geolocation in the geo_zipcode array and returns the association zip code.
4. In all, the cleaner mapper (GetZipcodeAndCleanMapper.java) removes unnecessary columns, malformatted rows, and converts latitude and longitude to zip code. The output of the mapper uses complaint_id as key and the combination of zip code, report_date, offense_code and offense_description as value. Each field in the value is separated by "|".
5. The cleaner reducer (GetZipcodeAndCleanReducer.java) is almost like an identity reducer. The only difference is that it ignores the complaint_id. The output of the reducer uses NullWritable as key and the value of mapper as the value. Here, I only used one reducer and the output is stored in project/output/clean.

[WIP]What I did for the data - Get Total Complaints per Zip Code:

1. The input of this job is crime-data-clean.csv, the clean data with zip code that was obtained from the last job. The output of this job is the total number of complaints for each zip code.
2. Mapper (ZipcodeMapper.java) for this job is relatively simple. It extracts the zip code from the input line and writes it as a key. Other parts of the input line will be written as value. This mapper is very generous and can be used in other jobs as well.
3. Reducer (GetTotalComplaintsReducer.java) for this job maintains a variable called total_complaints. For each information it receives of the zip code, it increments 1 to total_complaints. The output of the reducer has zip code as key, and total_complaints as value.

[WIP]What I did for the data - Group Complaints by Type and Get the Counts for Zip Code:

1. The input of this job is crime-data-clean.csv, the clean data with zip code that was obtained from the last job. The output of this job is the total number of complaints for each offense type and each zip code.
2. The Mapper of this job is still ZipcodeMapper.java, as described in the last page.
3. The Reducer of this job is GroupComplaintsTypeReducer.java. It maintains a TreeMap. For each information of a zip code, the reducer stores the occurrence of the offense type in the TreeMap. The output uses the zip code and offense type as key, separated by ":", and the associated value in the TreeMap.

Reference:

1. https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgea-i56i
2. https://data.cityofnewyork.us/api/views/qgea-i56i/files/b21ec89f-4d7b-494e-b2e9-f69ae7f4c228?download=true&filename=NYPD_Complaint_Incident_Level_Data_Footnotes.pdf
3. https://gist.github.com/erichurst/7882666