Data Ingestion and Profiling:

This is the first checkpoint of the project, which describes how I ingested, cleaned, preprocessed and profiled the dataset which I will be working on.
DOT Traffic Speed NBE is a collection of traffic speed data in NYC with data ranging from 2017 to the present day. This has over 70 million rows and is more than 30 GB in size.

These are the steps to ingest the data to a format which can be easily analyzed:

1. Use two **Map-Reduce jobs** to profile some columns of the data:
    a. WordCount - “BOROUGH” and “STATUS” word count for these columns
    b. Statistics - “SPEED” and “TRAVEL_TIME” columns min, max, mean, median
2. A **Map-Reduce job** to sample a small portion of the data. This was done to do some local analysis of the data before writing the ingestion pipeline, and test it on a small scale
3. A **Map-Reduce job** for cleaning and preprocessing the data which does the following:
    a. Format some column values to support Hive queries(such as Date queries, joins)
    b. Filter some rows based on some column values
    c. Produce part files as the job output, which can be loaded into Hive
4. Loading the part files (generated by the Map-Reduce job) into a **Hive table** which can support easy analysis.
   Before analyzing, I have created a profile of the data which I have summarized below.

Data Cleaning and Preprocessing:

Map-Reduce jobs for Initial Profiling -

1. **Word Count:**
   The code for this Map-Reduce task can be found in the WordCounter folder.
   i) BOROUGH - Get how the data is distributed between the 5 boroughs of New York City. The result produced by the job is stored in BOROUGHS.txt
   ii) STATUS - Shows the status of the recorded row (ie if it is valid (0) or invalid (-101)). The result produced by the job is stored in STATUS.txt

   Shell command (The last digit is for the column index):
   `hadoop jar wordCounter.jar ProjectData/traffic_speed.csv WordCounter/output 11`

2. **Statistical Analysis:**
   The code for this job can be found in StatisticalAnalysis. It can analyze numerical columns and will give the (min, max, average, median) for the values in the column.
   i) SPEED - Find the min, max, average, median across all recorded speeds (SPEED.txt)
   ii) TRAVEL_TIME - Find the min, max, average, median travel time (TRAVEL_TIME.txt)

   Shell command (The last digit is for the column index):
   `hadoop jar statAnalysis.jar ProjectData/traffic_speed.csv StatAnalysis/output 1`

Random Sampling -

Since the original data is ~ 35 GB in size, I wanted to get some idea about the dataset using a randomly chosen sample. The purpose behind this was manifold -

i) Run experiments on a smaller portion of the data which could be loaded to my local disk
ii) Get an idea about the street addresses so that I could design rules to format them
iii) Understand the values of some columns which allows filtering data in the ingestion stage
iv) Test the Map-Reduce framework on a smaller scale

This was done using a Map-Reduce job, the code for which is located in the "RandomSampler" folder. I chose to sample only 2% of the data for this, resulting in a file of size ~ 700 MB.

Shell command to run the job:
hadoop jar randomSampler.jar ProjectData/traffic_speed.csv RandomSampler/output

Data Preprocessing -

The data cleaning and preprocessing is necessary to have the data in a form which can be easily loaded into Hive tables.
This was done through another map-reduce job - DataPreprocessor - which produces multiple part files that are filtered, cleaned, and formatted in the required format.

This job does the following:

1. Filter invalid rows - Based on the value of the column STATUS (-101 is invalid).
2. Drop unnecessary columns in the original data - Columns such as LINK_ID, LINK_POINTS, ENCODED_POLY_LINE are not required for analysis and are dropped.
3. Format the value of columns - The following columns require formatting before analysis:
    a. **DATE_AS_OF**: The date needs to be reformatted to the following format: "yyyy-MM-dd HH:mm:ss" as this is the format recognized by Hive for the "TIMESTAMP" datatype.
    b. **LINK_NAME**: This column is split into two separate columns (FROM_STREET, TO_STREET) and the street addresses are individually formatted based on the rules specified in the Appendix.
    This is required for uniformity in the address values, since we might potentially perform joins between multiple tables based on these addresses for analysis.

The final part files created by this job can be loaded into a Hive table for further analysis.

Shell command to run the job:
hadoop jar preProcessor.jar ProjectData/traffic_speed.csv PreProcessor/output

Data Profiling:

The data profiling was done through map-reduce jobs described above.

These are some rows from the dataset before cleaning and pre-processing:

ID,SPEED,TRAVEL_TIME,STATUS,DATA_AS_OF,LINK_ID,LINK_POINTS,ENCODED_POLY_LINE,ENCODED_POLY_LINE_LVLS,OWNER,TRANSCOM_ID,BOROUGH,LINK_NAME
385,62.13,23,0,12/13/2021 12:18:10 AM,4616208,"40.6077805,-74.14091 40.60826,-74.132101",sezvFtsocM_Bav@,BB,NYC_DOT_LIC,4616208,Staten Island,SIE W BRADLEY AVENUE - WOOLEY AVENUE
258,15.53,268,-101,12/13/2021 12:18:10 AM,4616220,"40.6162405,-74.02612 40.61923,-74.0236140.62362",oz{vFffybMuQuNmZsVsFyC}D}AeLmDeO_E{JkCaEg@,BBBB BBBBB,NYC_DOT_LIC,4616220,Brooklyn,GOW N 92ND STREET - 7TH AVENUE
129,63.37,73,0,12/13/2021 12:18:10 AM,4616246,"40.8240706,-73.874311 40.8247",mmdxFjq{aM}Bm\aBwUa@sIeB_\qCq`@sB{Y,BBBBBBB,NYC_DOT_LIC,4616246, Bronx,BE N STRATFORD AVENUE - CASTLE HILL AVE

These are some rows from the dataset after cleaning and pre-processing:

SPEED,TRAVEL_TIME,STATUS,DATA_AS_OF,BOROUGH,FROM_STREET,TO_STREET
57.78,95,0,2020-04-11 20:48:11,Staten Island,CLOVE RD,FINGERBOARD RD
29.82,149,0,2020-04-11 20:48:07,Bronx,HARLEM RIVER PARK,AMSTERDAM AVE
57.16,134,0,2020-04-11 20:48:07,Staten Island,TYRELLAN AVE,FRANCIS ST

The ingested data (after filtering and cleaning) has **62.3 million** rows.

Data Profile for Columns of Interest:

1) SPEED

The following profile was generated by the StatisticalAnalysis map-reduce job, with the job output available in SPEED.txt

Minimum	Maximum	Mean	Median
0.0	186.41	39.66	39.15

The above values are in miles/hr based on the imperial system.

2) TRAVEL_TIME

The following profile was also generated by the StatisticalAnalysis map-reduce job, with the job output available in SPEED.txt. The mapper accepts the column_index as a parameter.

Minimum	Maximum	Mean	Median
0.0	86801.0	222.71	116.0

The above values are in minutes (which is the time taken to traverse the link).

3) STATUS

The following profile was created using the WordCounter map-reduce job, with the output available in STATUS.txt

0 (Valid)	-101 (Invalid)
64,087,357 (82.5%)	13,587,573 (17.5%)

The Invalid rows were thus filtered in the pre-processing map-reduce job.

4) DATA_AS_OF

The following profile was created using this Hive query after loading the data into the table:

```
SELECT
   YEAR(DATE_TIME) AS year,
   ROUND(((count(*) * 100.0) / CAST((SELECT count(*) FROM traffic_speed_test2) AS
     DECIMAL(10, 2))), 2) AS row_count
FROM
  traffic_speed_data
GROUP BY
  YEAR(date_time)
ORDER BY
  YEAR(date_time) ASC;
```

2017	2018	2019	2020	2021	2022	2023
4,472,786 (7.18%)	11,601,284 (18.61%)	10,961,885 (17.59%)	9,866,031 (15.83%)	10,013,091 (16.07%)	7,754,308 (12.44%)	7,656,852 (12.29%)

The data is thus fairly distributed across all years from 2017, and is updated daily.

5) BOROUGH

The following profile was created using the WordCounter map-reduce job, with the output available in BOROUGHS.txt

Bronx	Brooklyn	Manhattan	Queens	Staten Island
11,099,393 (17.8%)	7,122,983 (11.4%)	11,744,035 (18.8%)	19,907,231 (31.9%)	12,550,968 (20.1%)

All the boroughs thus have a fair share of the recordings. Queens and Staten Island probably have a greater share, owing to the ease of driving and lack of public transit.

6) FROM_STREET

These are the cleaned and formatted street names. The original column - LINK_NAME - which is of the format “From_Street - To_Street” has been thus separated into 2 columns and properly formatted.
No value in this column is blank/default since the data was clean and consistent, thankfully.

7) TO_STREET

These are also cleaned and formatted street names.
However, 6 rows have the default value (“N/A”) since the value wasn’t recorded in the data.

This is the command I used to load the preprocessed data into Hive:

```
CREATE EXTERNAL TABLE
        traffic_speed
                (SPEED decimal(6, 2),
                TRAVEL_TIME int,
                STATUS int,
                DATE_TIME timestamp,
                BOROUGH string,
                FROM_STREET string,
                TO_STREET string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION     'PreProcessor/output/';
```

Appendix:

Rules to split the Link Name (From_Street - To_Street) into 2 separate columns, and to format them for generalization. The street names have to be formatted because we could potentially perform joins among different tables (Traffic Speed/Volume/Crashes) based on the Street Name.

Link values are of the form (Street1 - Street2). First we split this by '-' to get the to/from streets. Now to pre-process the to/from streets individually:

The first step is to choose the right portion of the Link value:

1. Split by `[" N ", " S ", " E ", " W ", " @ ", " WB ", " NB "]` - this would give you 2 parts (left / right).
2. If one is NOT NULL while the other is NULL, choose the NOT NULL value.
3. Choose whichever includes one of `["ST", "STREET", "AVE", "AVENUE", "ROAD", "BLVD"]` with higher precedence to the right value.
4. Choose whichever includes one of `["TPKE", "HWY", "PKWY", "EXPRESSWAY", "PLAZA"]` with higher precedence to the right value.
5. Choose whichever has more number of words (no precedence)
6. Choose whichever has more characters (precedence to right)

Then, we further pre-process this Link value for uniformity:

1. Remove "(" and ")" from the link name
2. Remove `["ST", "ND", "RD", "TH"]` after numerals
3. Map ("`STREET`" : "`ST`", "`AVENUE`" : "`AVE`", "`AV`" : "`AVE`", "`BOULEVARD`": "`BLVD`", "`ROAD`": "`RD`", "`PARKWAY`": "`PKY`", "`HIGHWAY`": "`HWY`", "`COURT`": "`CT`", "`PLACE`": "`PL`", "`SQUARE`": "`SQ`", "`TURNPIKE`": "`TPKE`", "`LANE`": "`LN`", "`POINT`": "`PT`", "`PLAZA`": "`PZ`")
4. Remove these strings, if present, and the word preceding them: `["LEVEL"]`
5. Remove these strings, if present, and the word after them: `["EXIT"]`
6. Skip the word if it has one of these substrings: [`"."`]

The following rules have been implemented in PreProcessorMapper.java (DataPreprocessor).