Northeastern University

US Traffic Accidents Data Analysis

Apache Hadoop

INFO 7245 Engineering Big Data

Professor: Yusuf Ozbek

Tanvi Rajendra Gurav
NUID: 001306848

Table of Contents

DATASET (https://www.kaggle.com/sobhanmoosavi/us-accidents)

Description:

This is a countrywide car accident dataset, which covers 49 states of the USA. The accident data are collected from February 2016 to Dec 2020, using two APIs that provide streaming traffic incident (or event) data. These APIs broadcast traffic data captured by a variety of entities, such as the US and state departments of transportation, law enforcement agencies, traffic cameras, and traffic sensors within the road-networks. Currently, there are about 4.2 million accident records in this dataset.

Acknowledgements:

Please cite the following papers if you use this dataset:

Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath. "A Countrywide Traffic Accident Dataset.", 2019.

Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath. "Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights." In proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2019.

Content:

This dataset has been collected in real-time, using multiple Traffic APIs. Currently, it contains accident data that are collected from February 2016 to Dec 2020 for the Contiguous United States. Check here to learn more about this dataset.

Usage Policy and Legal Disclaimer:

This dataset is being distributed only for Research purposes, under Creative Commons Attribution-Noncommercial-ShareAlike license (CC BY-NC-SA 4.0). By clicking on download button(s) below, you are agreeing to use this data only for non-commercial, research, or academic applications. You may need to cite the above papers if you use this dataset.

#	Attribute	Description	Null
1	ID	This is a unique identifier of the accident record.	No
2	Source	Indicates source of the accident report (i.e. the API which reported the accident.).	No
3	TMC	A traffic accident may have a Traffic Message Channel (TMC) code which provides more detailed description of the event.	Yes
4	Severity	Shows the severity of the accident, a number between 1 and 4, where 1 indicates the least impact on traffic (i.e., short delay as a result of the accident) and 4 indicates a significant impact on traffic (i.e., long delay).	No
5	Start_Time	Shows start time of the accident in local time zone.	No
6	End_Time	Shows end time of the accident in local time zone. End time here refers to when the impact of accident on traffic flow was dismissed.	No
7	Start_Lat	Shows latitude in GPS coordinate of the start point.	No
8	Start_Lng	Shows longitude in GPS coordinate of the start point.	No
9	End_Lat	Shows latitude in GPS coordinate of the end point.	Yes
10	End_Lng	Shows longitude in GPS coordinate of the end point.	Yes
11	Distance(mi)	The length of the road extent affected by the accident.	No
12	Description	Shows natural language description of the accident.	No
13	Number	Shows the street number in address field.	Yes
14	Street	Shows the street name in address field.	Yes
15	Side	Shows the relative side of the street (Right/Left) in address field.	Yes
16	City	Shows the city in address field.	Yes
17	County	Shows the county in address field.	Yes
18	State	Shows the state in address field.	Yes
19	Zipcode	Shows the zipcode in address field.	Yes
20	Country	Shows the country in address field.	Yes
21	Timezone	Shows timezone based on the location of the accident (eastern, central, etc.).	Yes
22	Airport_Code	Denotes an airport-based weather station which is the closest one to location of the accident.	Yes
23	Weather_Timestamp	Shows the time-stamp of weather observation record (in local time).	Yes
24	Temperature(F)	Shows the temperature (in Fahrenheit).	Yes

#	Attribute	Description	Null
25	Wind_Chill(F)	Shows the wind chill (in Fahrenheit).	Yes
26	Humidity(%)	Shows the humidity (in percentage).	Yes
27	Pressure(in)	Shows the air pressure (in inches).	Yes
28	Visibility(mi)	Shows visibility (in miles).	Yes
29	Wind_Direction	Shows wind direction.	Yes
30	Wind_Speed(mph)	Shows wind speed (in miles per hour).	Yes
31	Precipitation(in)	Shows precipitation amount in inches, if there is any.	Yes
32	Weather_Condition	Shows the weather condition (rain, snow, thunderstorm, fog, etc.)	Yes
33	Amenity	A POI annotation which indicates presence of amenity in a nearby location.	No
34	Bump	A POI annotation which indicates presence of speed bump or hump in a nearby location.	No
35	Crossing	A POI annotation which indicates presence of crossing in a nearby location.	No
36	Give_Way	A POI annotation which indicates presence of give_way in a nearby location.	No
37	Junction	A POI annotation which indicates presence of junction in a nearby location.	No
38	No_Exit	A POI annotation which indicates presence of no_exit in a nearby location.	No
39	Railway	A POI annotation which indicates presence of railway in a nearby location.	No
40	Roundabout	A POI annotation which indicates presence of roundabout in a nearby location.	No
41	Station	A POI annotation which indicates presence of station in a nearby location.	No
42	Stop	A POI annotation which indicates presence of stop in a nearby location.	No
43	Traffic_Calming	A POI annotation which indicates presence of traffic_calming in a nearby location.	No
44	Traffic_Signal	A POI annotation which indicates presence of traffic_signal in a nearby location.	No
45	Turning_Loop	A POI annotation which indicates presence of turning_loop in a nearby location.	No
46	Sunrise_Sunset	Shows the period of day (i.e. day or night) based on sunrise/sunset.	Yes
47	Civil_Twilight	Shows the period of day (i.e. day or night) based on civil twilight.	Yes
48	Nautical_Twilight	Shows the period of day (i.e. day or night) based on nautical twilight.	Yes
49	Astronomical_Twilight	Shows the period of day (i.e. day or night) based on astronomical twilight.	Yes

1. Number of Traffic Accidents occurred by State

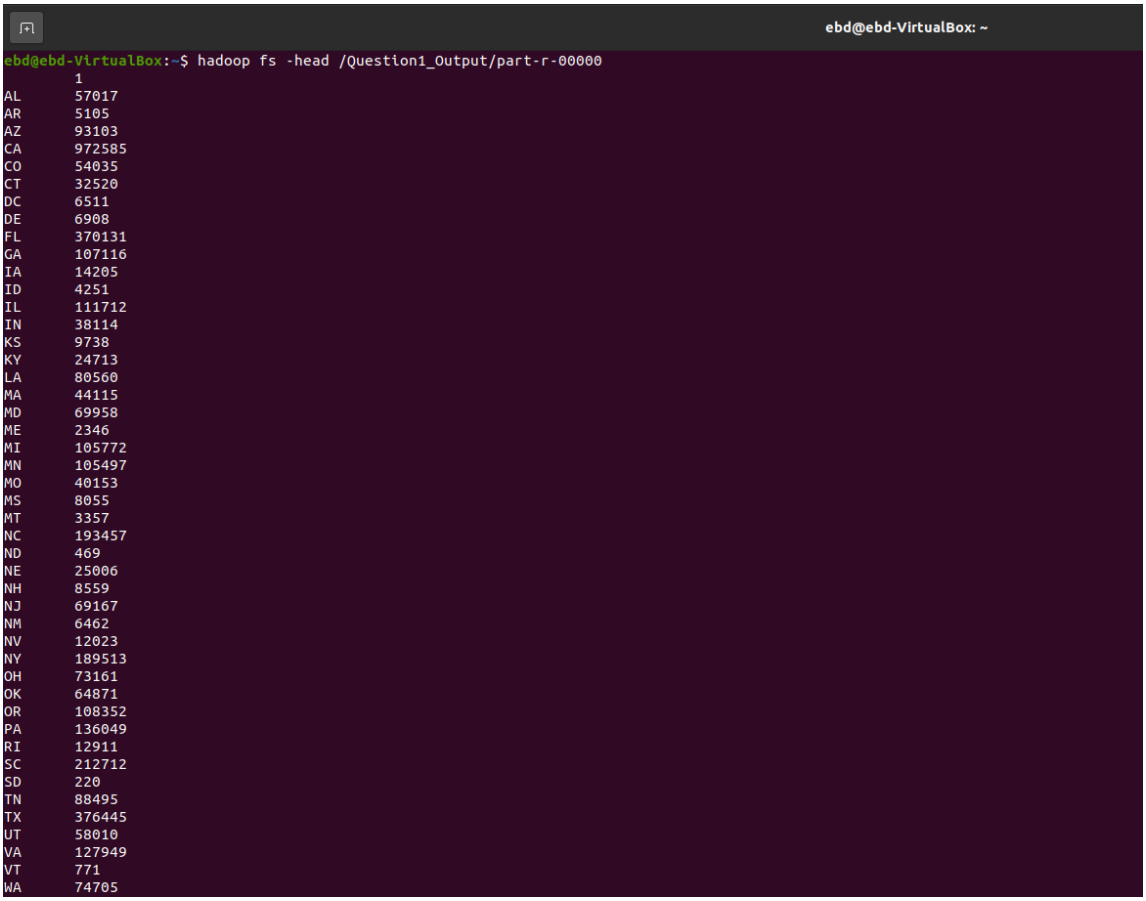
Description: This is a simple MapReduce Job that gives us an aggregate count of accidents occurred in each state in United States.

Analysis: We could analyze that the state of California (CA) has highest number of accidents i.e., 972,585.

Command to execute the jar:

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question1.DriverClass /US_Accidents_Dec20.csv
/Question1_Output
```

Output of the MapReduce Job:



2. Number of Accidents near Junction VS not a Junction

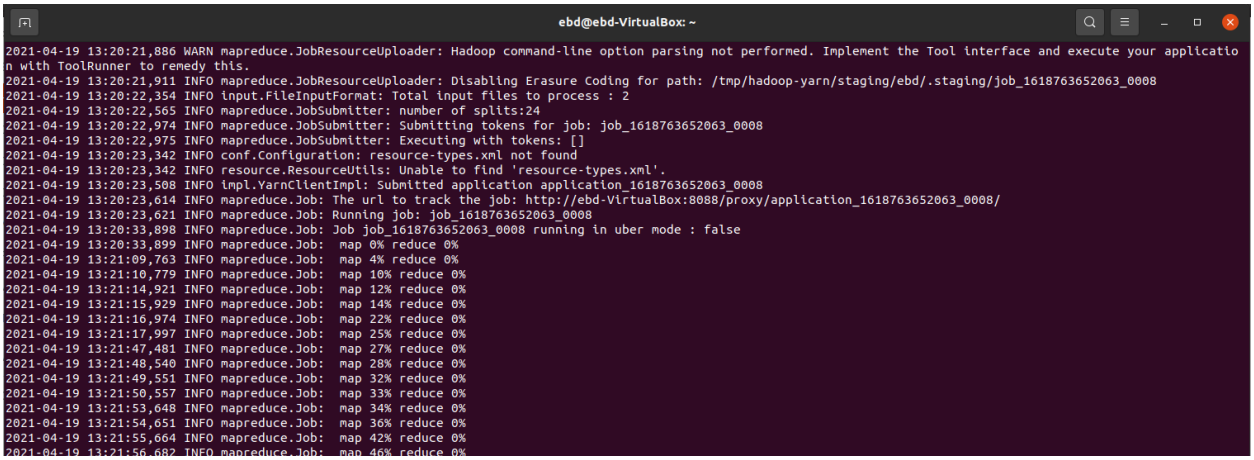
Description: This is a simple MapReduce Job that gives us an aggregate count of accidents occurred near a Junction versus number of accidents not near a Junction. **Junction is a Boolean column** having 'True' or 'False' values. Hence, we map a meaningful string to Boolean value using HashMap with key as Junction Boolean column and substitute string as value.

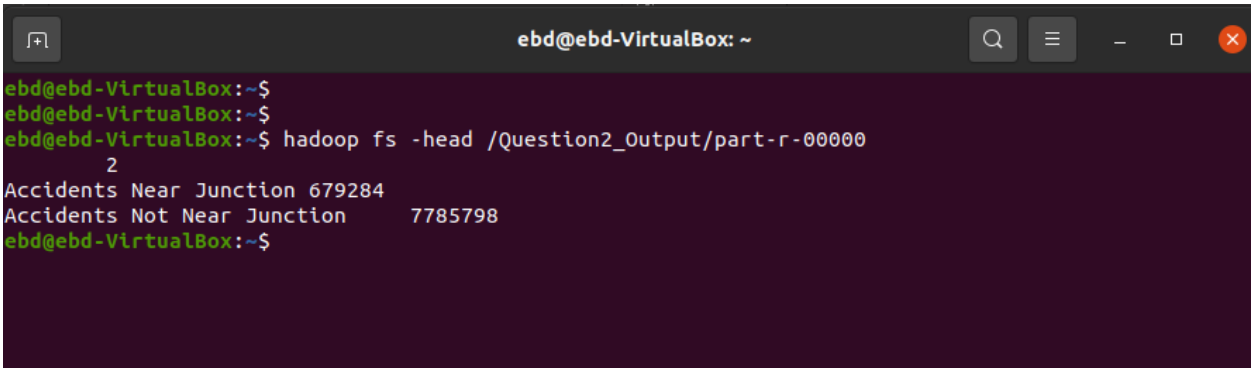
Analysis: We could analyze that a greater number of accidents took place where junction was not present i.e., 7,785,798

Command to execute the jar:

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question2.DriverClass /US_Accidents_Dec20.csv
/Question2_Output
```

Output of the MapReduce Job:





3. Average, Min, Max Visibility (miles), Max StartDate Per Accident Severity

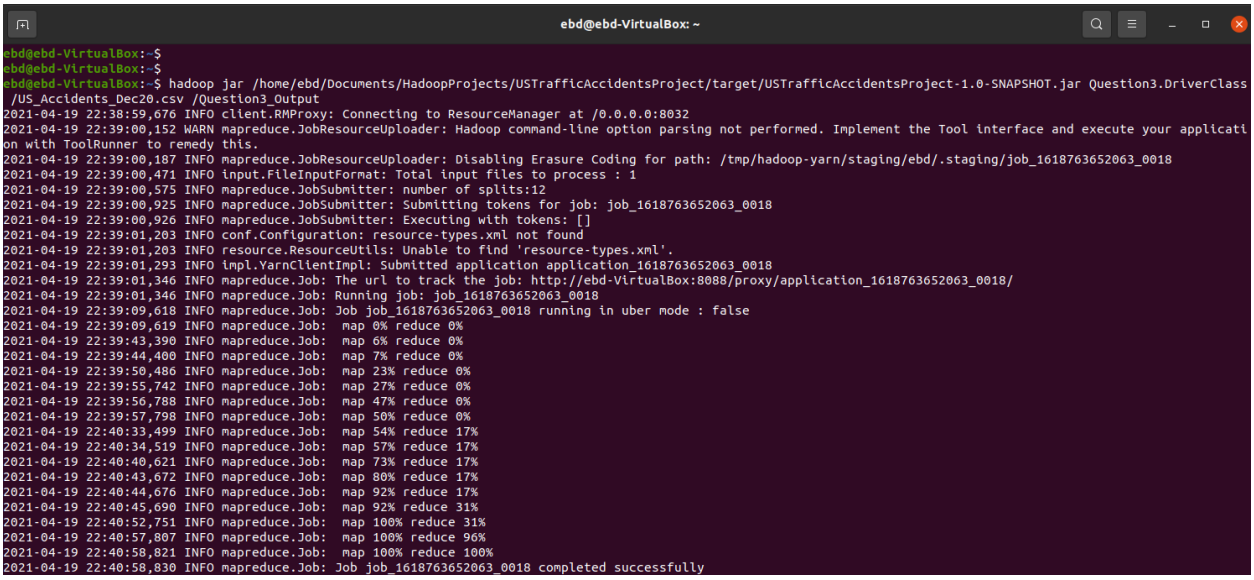
Description: This MapReduce Job performs Numerical Summarization that gives **Maximum, Minimum & Average Visibility along with Maximum Accident Start Date according to accident Severity** which ranges from 1 to 4. This is achieved with a custom class which implements Writable class.

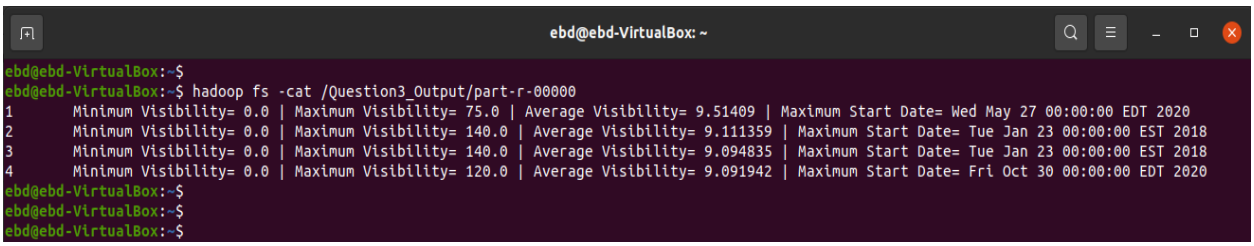
Analysis: We could analyze that lowest average visibility of 9.091942 has caused most severe accident (Severity 4) with most recent accident case on 31st October 2020.

Command to execute the jar:

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question3.DriverClass /US_Accidents_Dec20.csv
/Question3_Output
```

Output of the MapReduce Job:





4. Most Recent Accident Date per City

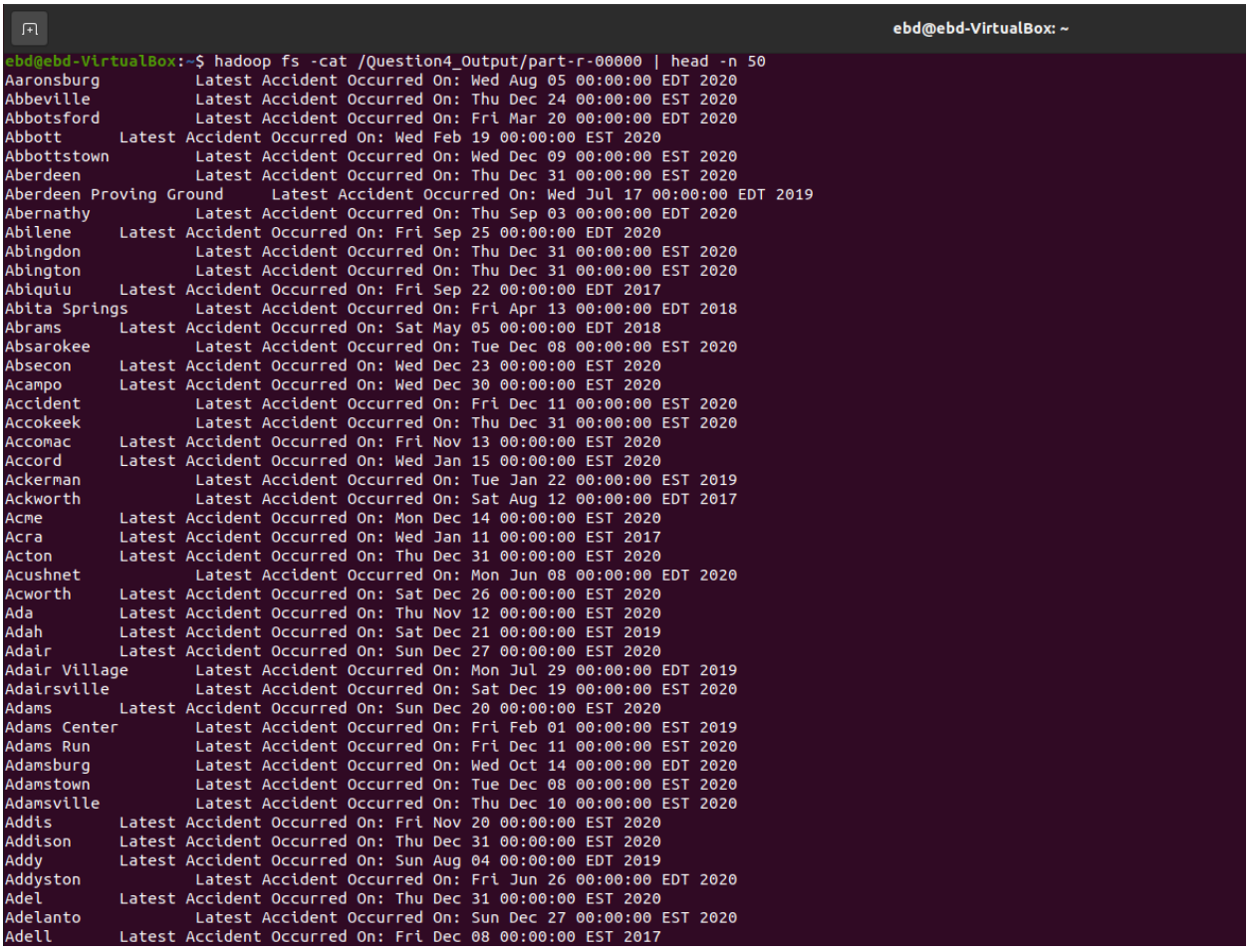
Description: This MapReduce Job performs Numerical Summarization that gives **Maximum Accident Date for each City** in United States.

Analysis: We could analyze that in most of the cities, latest accident case was in the year 2020 with month ranging from Jan to Dec.

Command to execute the jar:

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question4.DriverClass /US_Accidents_Dec20.csv
/Question4_Output
```

Output of the MapReduce Job:



5. Percentage of accidents in each month

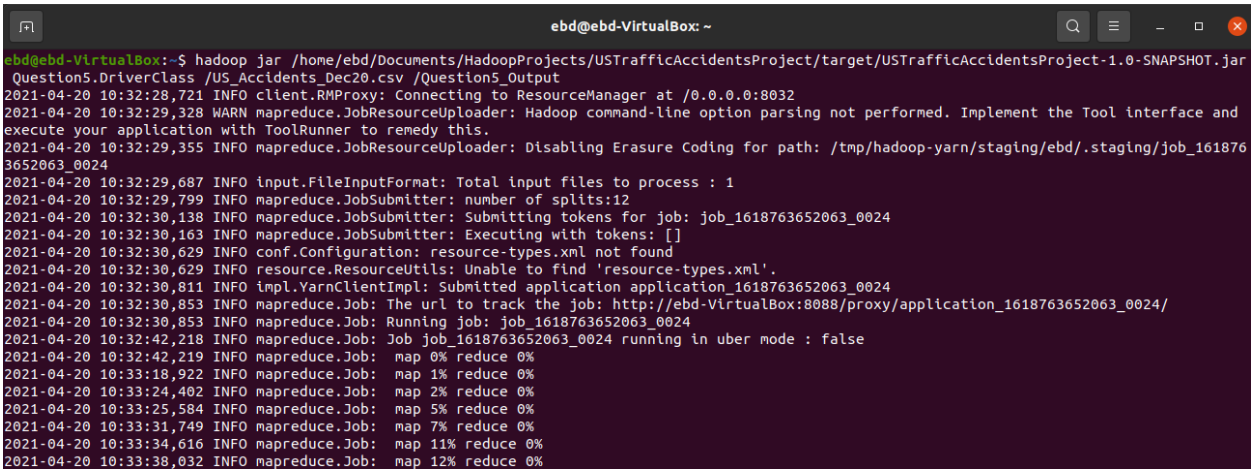
Description: This MapReduce Job gives **the Percentage of Accidents took place in each month** from 2016 to 2020.

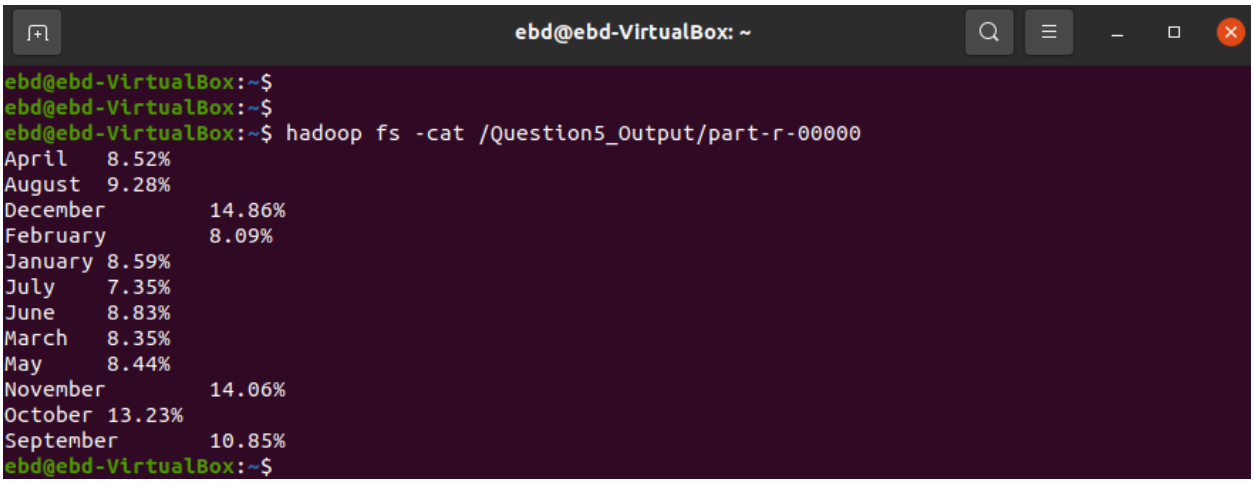
Analysis: We could analyze that the percentage of accidents were more in later 4 months (Fall Season) of each year from October to December ranging from 10.85% to 14.86%.

Command to execute the jar:

```
hadoop jar /home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question5.DriverClass /US_Accidents_Dec20.csv /Question5_Output
```

Output of the MapReduce Job:





6. Top 10 States with Highest Number of Accidents

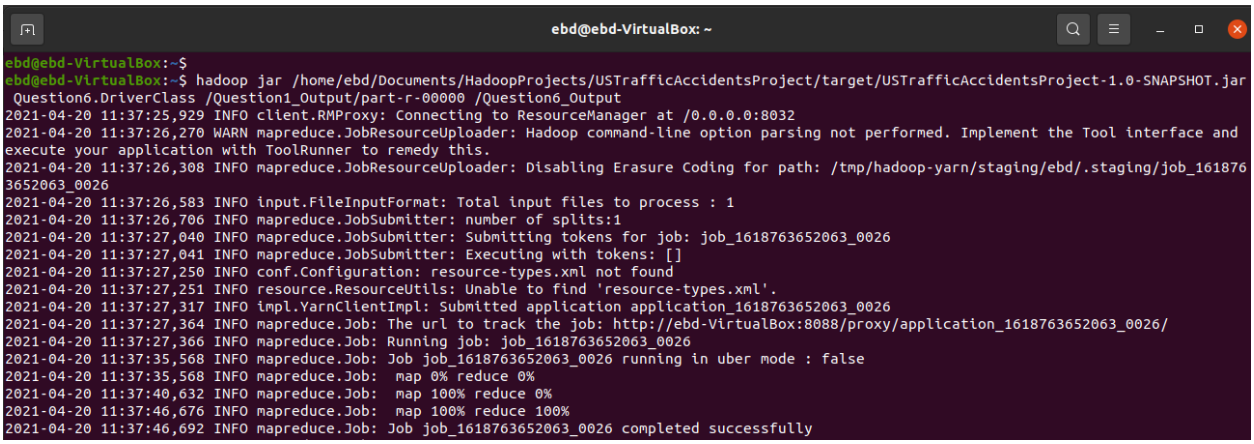
Description: This MapReduce Job performs Top 'N' Filtering Pattern that gives Top 10 States with highest number of Accidents from year 2016 to 2020.

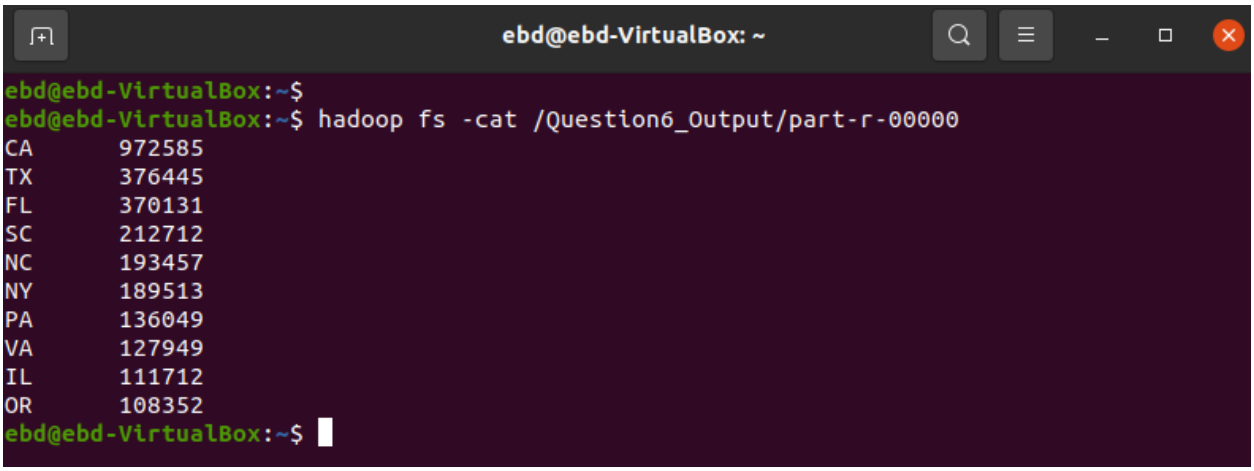
Analysis: We could analyze that the state of California tops the number of Traffic Accidents in United States following Texas and Florida.

Command to execute the jar:

```
hadoop jar /home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question6.DriverClass /Question1_Output/part-r-00000 /Question6_Output
```

Output of the MapReduce Job:





7. Top 5 Weather Conditions most conducive to Traffic Accidents

Description: This MapReduce Job performs Top 'N' Filtering Pattern that gives **Top 5 Weather Conditions most conducive to Traffic Accidents.**

Analysis: We could analyze that most of the traffic accidents occur when the weather is Fair/Clear or Mostly/Partly Cloudy or Outcast

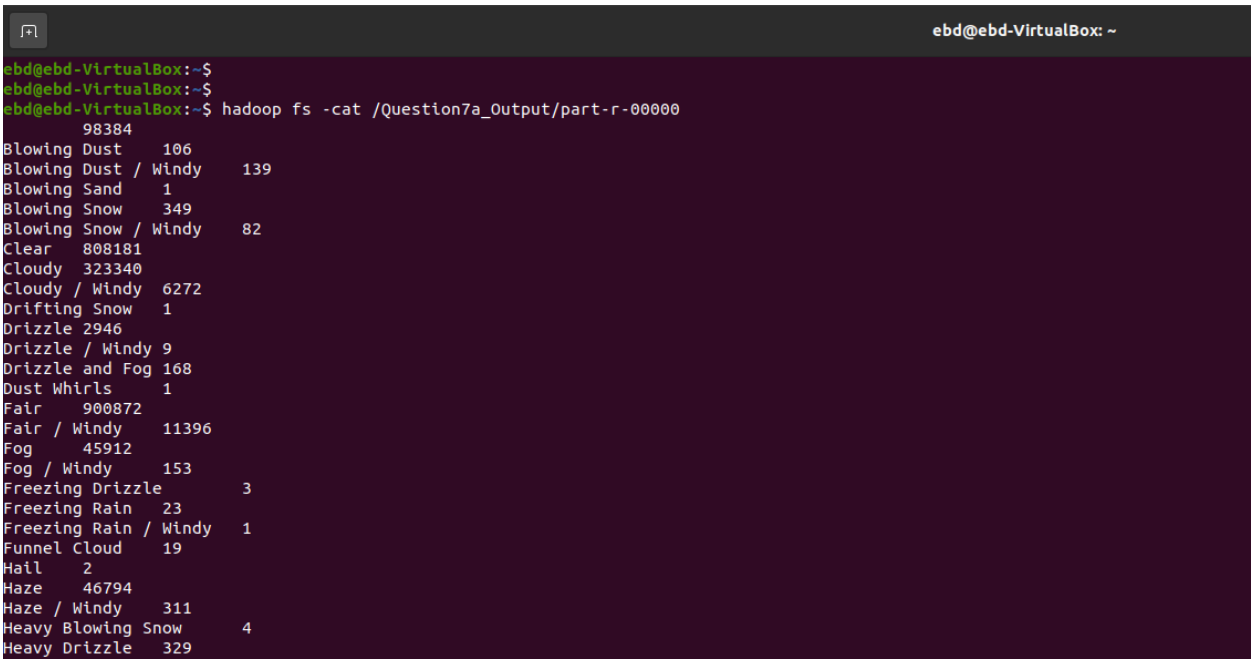
a. Command to get the count of accidents per weather condition:

```
hadoop jar /home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question7.a.DriverClass /US_Accidents_Dec20.csv /Question7a_Output
```

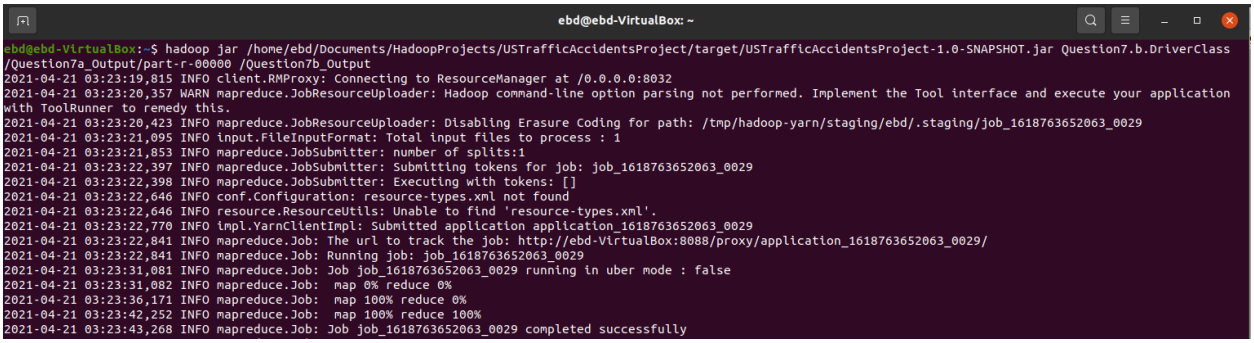
b. Command to get the Top 5 weather conditions from above output:

```
hadoop jar /home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question7.b.DriverClass /Question7a_Output/part-r-00000 /Question7b_Output
```

Output of the first MapReduce Job:



Using the first output as input to the second job for filtering



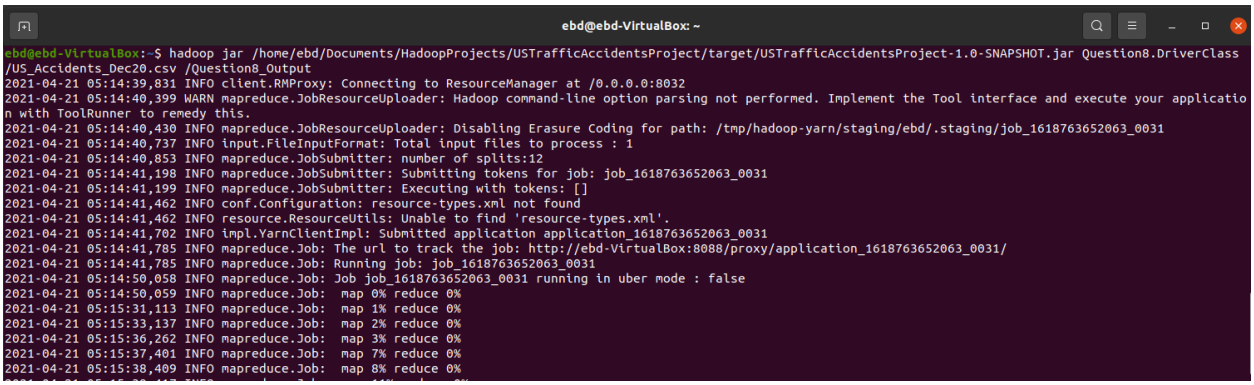
8. Number of Accidents Per County Per Year

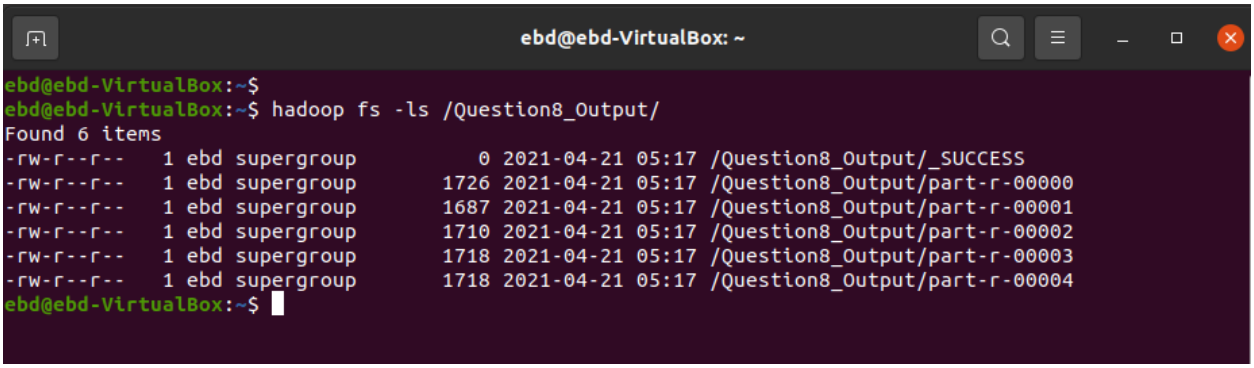
Description: This MapReduce Job performs 'Secondary Sorting' where we are sorting, and partitioning based on year from 2016 to 2020. We get 5 partitions as shown in the output.

Command to execute the jar:

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question8.DriverClass /US_Accidents_Dec20.csv
/Question8_Output
```

Output of the MapReduce Job:





```
ebd@ebd-VirtualBox:~$
ebd@ebd-VirtualBox:~$ hadoop fs -cat /Question8_Output/part-r-00001 | head -n 10
County: Abbeville| Year: 2016 | Accidents:      58
County: Acadia| Year: 2016 | Accidents:         8
County: Accomack| Year: 2016 | Accidents:       12
County: Ada| Year: 2016 | Accidents:    74
County: Adair| Year: 2016 | Accidents:  14
County: Adams| Year: 2016 | Accidents:  361
County: Addison| Year: 2016 | Accidents:        1
County: Aiken| Year: 2016 | Accidents:  662
County: Aitkin| Year: 2016 | Accidents:         6
County: Alachua| Year: 2016 | Accidents:        129
cat: Unable to write to output stream.
ebd@ebd-VirtualBox:~$
```

```
ebd@ebd-VirtualBox:~$
ebd@ebd-VirtualBox:~$ hadoop fs -cat /Question8_Output/part-r-00002 | head -n 10
County: Abbeville| Year: 2017 | Accidents:      97
County: Acadia| Year: 2017 | Accidents:         23
County: Accomack| Year: 2017 | Accidents:       21
County: Ada| Year: 2017 | Accidents:    53
County: Adair| Year: 2017 | Accidents:  37
County: Adams| Year: 2017 | Accidents:  1307
County: Addison| Year: 2017 | Accidents:        13
County: Aiken| Year: 2017 | Accidents:  1377
County: Aitkin| Year: 2017 | Accidents:         22
County: Alachua| Year: 2017 | Accidents:        454
cat: Unable to write to output stream.
ebd@ebd-VirtualBox:~$
```

```
ebd@ebd-VirtualBox:~$
ebd@ebd-VirtualBox:~$ hadoop fs -cat /Question8_Output/part-r-00003 | head -n 10
County: Abbeville| Year: 2018 | Accidents:      138
County: Acadia| Year: 2018 | Accidents:         43
County: Accomack| Year: 2018 | Accidents:       20
County: Ada| Year: 2018 | Accidents:    90
County: Adair| Year: 2018 | Accidents:  10
County: Adams| Year: 2018 | Accidents:  2285
County: Addison| Year: 2018 | Accidents:        3
County: Aiken| Year: 2018 | Accidents:  1296
County: Aitkin| Year: 2018 | Accidents:         15
County: Alachua| Year: 2018 | Accidents:        608
cat: Unable to write to output stream.
ebd@ebd-VirtualBox:~$
```

```
ebd@ebd-VirtualBox:~$
ebd@ebd-VirtualBox:~$ hadoop fs -cat /Question8_Output/part-r-00004 | head -n 10
County: Abbeville| Year: 2019 | Accidents:      112
County: Acadia| Year: 2019 | Accidents:         19
County: Accomack| Year: 2019 | Accidents:       24
County: Ada| Year: 2019 | Accidents:    70
County: Adair| Year: 2019 | Accidents:  44
County: Adams| Year: 2019 | Accidents:  2663
County: Addison| Year: 2019 | Accidents:        5
County: Aiken| Year: 2019 | Accidents:  1019
County: Aitkin| Year: 2019 | Accidents:         78
County: Alachua| Year: 2019 | Accidents:        294
cat: Unable to write to output stream.
ebd@ebd-VirtualBox:~$
```

```
ebd@ebd-VirtualBox:~$
ebd@ebd-VirtualBox:~$ hadoop fs -cat /Question8_Output/part-r-00000 | head -n 10
County: Abbeville| Year: 2020 | Accidents:      213
County: Acadia| Year: 2020 | Accidents:         34
County: Accomack| Year: 2020 | Accidents:       34
County: Ada| Year: 2020 | Accidents:    1107
County: Adair| Year: 2020 | Accidents:  103
County: Adams| Year: 2020 | Accidents:  2603
County: Addison| Year: 2020 | Accidents:        11
County: Aiken| Year: 2020 | Accidents:  1061
County: Aitkin| Year: 2020 | Accidents:         153
County: Alachua| Year: 2020 | Accidents:        795
cat: Unable to write to output stream.
ebd@ebd-VirtualBox:~$
```

9. Partitioning based on Wind Direction

Description: This MapReduce Job performs 'Partitioning' where we partition our input with 4.2million records based on 23 Wind Directions. Each file contains the records corresponding to specific Wind Direction and maps the Wind Direction Abbreviation to its description using 'Inner Join'.

Command to execute the jar:

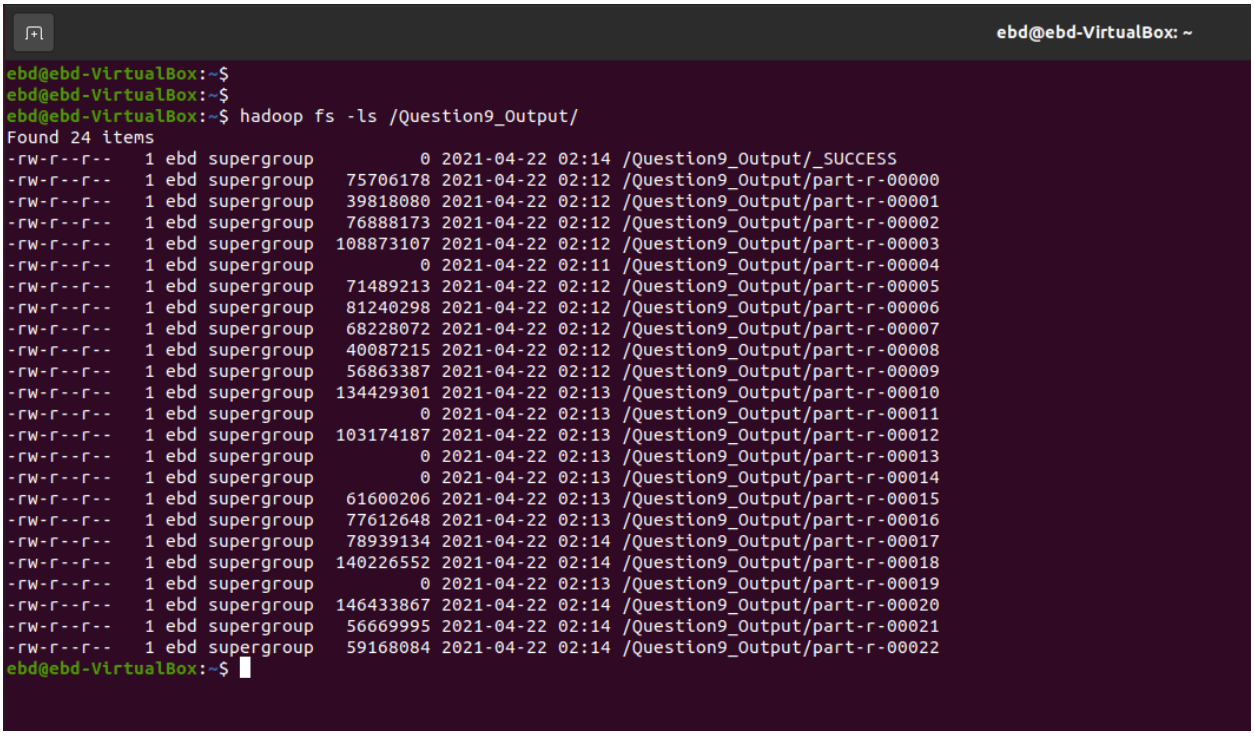
```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question9.DriverClass /US_Accidents_Dec20.csv
/WindDirection.csv /Question9_Output
```

Wind Direction Input:

	A	B
1	Wind_Direction	Description
2	Calm	Calm
3	E	East
4	East	East
5	ENE	East-Northeast
6	ESE	East-Southeast
7	N	North
8	NE	Northeast
9	NNE	North-Northeast
10	NNW	North-Northwest
11	North	North
12	NW	Northwest
13	S	South
14	SE	Southeast
15	South	South
16	SSE	South-Southeast
17	SSW	South-Southwest
18	SW	Southwest
19	VAR	Variable
20	Variable	Variable
21	W	West
22	West	West
23	WNW	West-Northwest
24	WSW	West-Southwest

Output of the MapReduce Job:

23 Partitions, each file containing data related to one Wind Direction.



Output of individual file:

10. Cities present in each County

Description: This MapReduce Job performs **'Invert Index Algorithm'** where we find out all the cities corresponding to each county from our data.

Analysis: We could analyze the number of cities where traffic accidents have taken place, that is captured in our data which is grouped under corresponding County.

Command to execute the jar:

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question10.DriverClass /US_Accidents_Dec20.csv
/Question10_Output
```

Output Format:

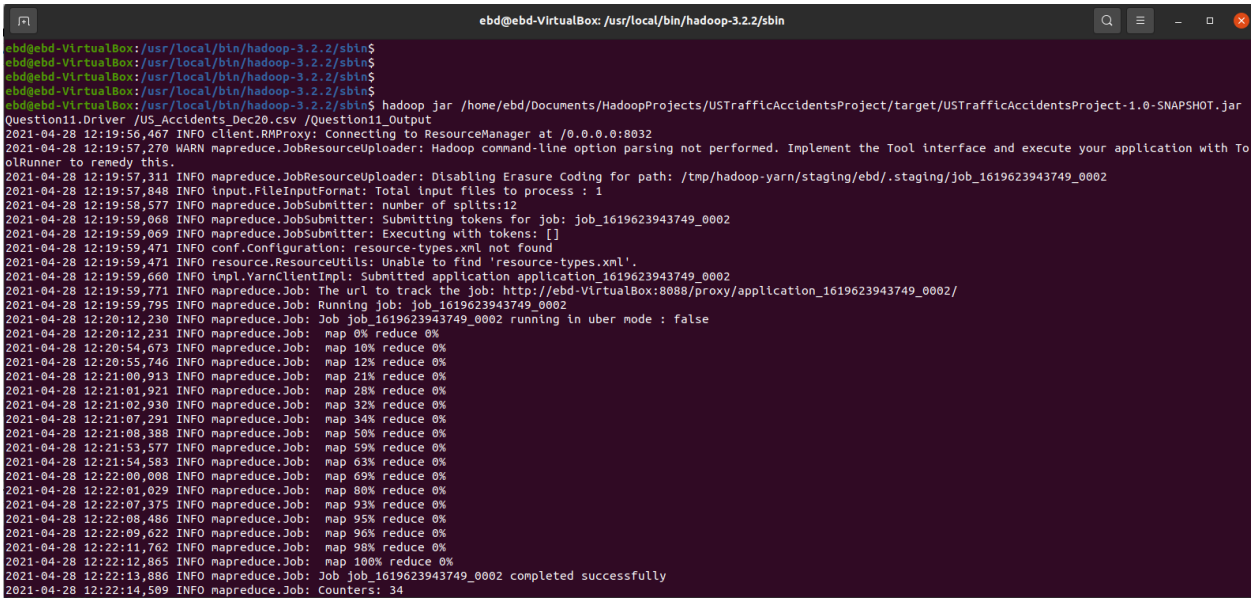
County City1, City2, City3, City4…. CityN

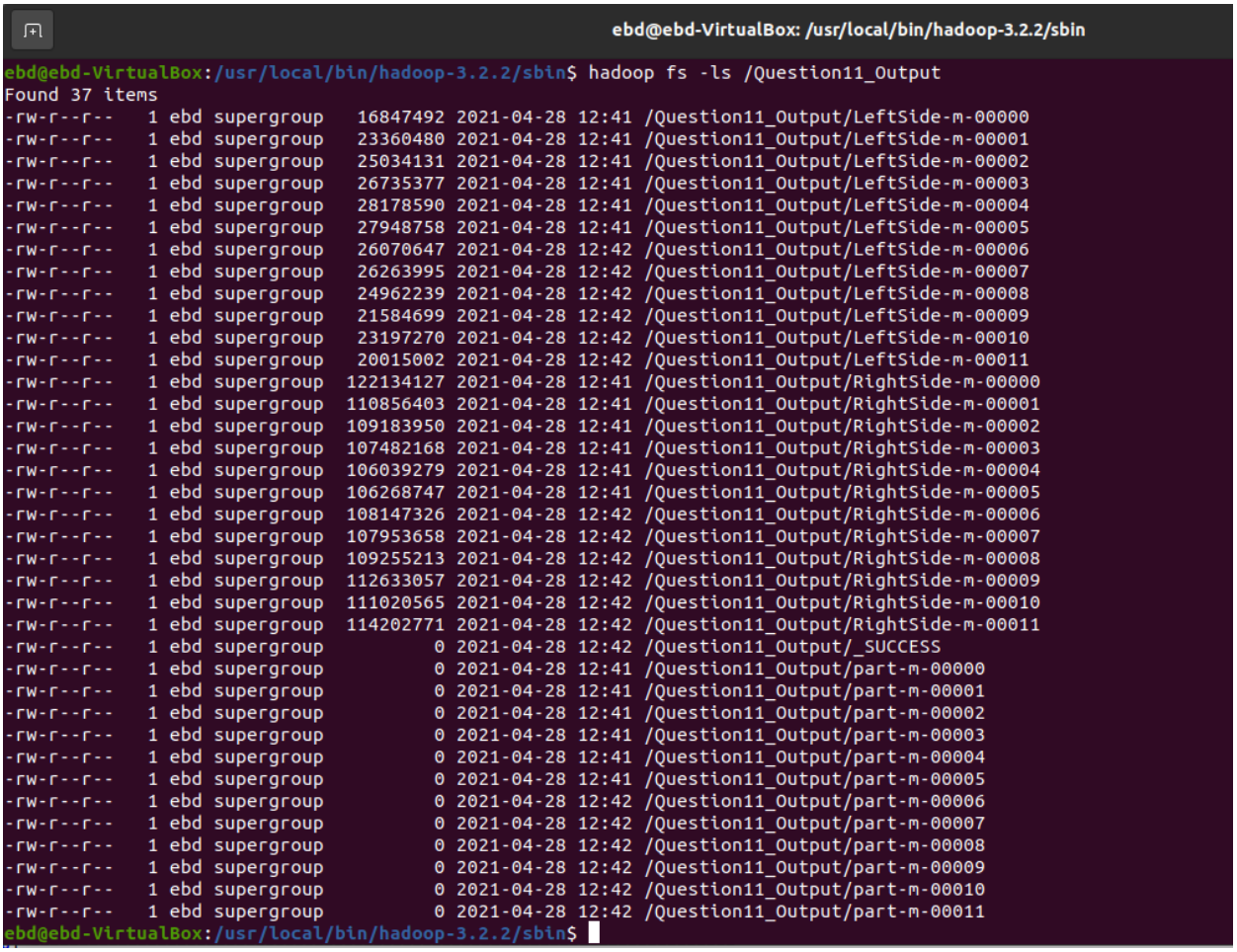
11. Binning based on Accidents on each side of the road (Left/Right)

Description: This MapReduce Job performs **'Binning'** that partitions the data based on accidents occurred on each side of the road

Command to execute the jar:

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question11.Driver /US_Accidents_Dec20.csv
/Question11_Output
```




ebd@ebd-VirtualBox: /usr/local/bin/hadoop-3.2.2/sbin
ebd@ebd-VirtualBox:/usr/local/bin/hadoop-3.2.2/sbin$ hadoop fs -ls /Question11_Output
Found 37 items
-rw-r--r--   1 ebd supergroup   16847492 2021-04-28 12:41 /Question11_Output/LeftSide-m-00000
-rw-r--r--   1 ebd supergroup   23360480 2021-04-28 12:41 /Question11_Output/LeftSide-m-00001
-rw-r--r--   1 ebd supergroup   25034131 2021-04-28 12:41 /Question11_Output/LeftSide-m-00002
-rw-r--r--   1 ebd supergroup   26735377 2021-04-28 12:41 /Question11_Output/LeftSide-m-00003
-rw-r--r--   1 ebd supergroup   28178590 2021-04-28 12:41 /Question11_Output/LeftSide-m-00004
-rw-r--r--   1 ebd supergroup   27948758 2021-04-28 12:41 /Question11_Output/LeftSide-m-00005
-rw-r--r--   1 ebd supergroup   26070647 2021-04-28 12:42 /Question11_Output/LeftSide-m-00006
-rw-r--r--   1 ebd supergroup   26263995 2021-04-28 12:42 /Question11_Output/LeftSide-m-00007
-rw-r--r--   1 ebd supergroup   24962239 2021-04-28 12:42 /Question11_Output/LeftSide-m-00008
-rw-r--r--   1 ebd supergroup   21584699 2021-04-28 12:42 /Question11_Output/LeftSide-m-00009
-rw-r--r--   1 ebd supergroup   23197270 2021-04-28 12:42 /Question11_Output/LeftSide-m-00010
-rw-r--r--   1 ebd supergroup   20015002 2021-04-28 12:42 /Question11_Output/LeftSide-m-00011
-rw-r--r--   1 ebd supergroup 122134127 2021-04-28 12:41 /Question11_Output/RightSide-m-00000
-rw-r--r--   1 ebd supergroup 110856403 2021-04-28 12:41 /Question11_Output/RightSide-m-00001
-rw-r--r--   1 ebd supergroup 109183950 2021-04-28 12:41 /Question11_Output/RightSide-m-00002
-rw-r--r--   1 ebd supergroup 107482168 2021-04-28 12:41 /Question11_Output/RightSide-m-00003
-rw-r--r--   1 ebd supergroup 106039279 2021-04-28 12:41 /Question11_Output/RightSide-m-00004
-rw-r--r--   1 ebd supergroup 106268747 2021-04-28 12:41 /Question11_Output/RightSide-m-00005
-rw-r--r--   1 ebd supergroup 108147326 2021-04-28 12:42 /Question11_Output/RightSide-m-00006
-rw-r--r--   1 ebd supergroup 107953658 2021-04-28 12:42 /Question11_Output/RightSide-m-00007
-rw-r--r--   1 ebd supergroup 109255213 2021-04-28 12:42 /Question11_Output/RightSide-m-00008
-rw-r--r--   1 ebd supergroup 112633057 2021-04-28 12:42 /Question11_Output/RightSide-m-00009
-rw-r--r--   1 ebd supergroup 111020565 2021-04-28 12:42 /Question11_Output/RightSide-m-00010
-rw-r--r--   1 ebd supergroup 114202771 2021-04-28 12:42 /Question11_Output/RightSide-m-00011
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:42 /Question11_Output/_SUCCESS
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:41 /Question11_Output/part-m-00000
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:41 /Question11_Output/part-m-00001
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:41 /Question11_Output/part-m-00002
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:41 /Question11_Output/part-m-00003
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:41 /Question11_Output/part-m-00004
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:41 /Question11_Output/part-m-00005
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:42 /Question11_Output/part-m-00006
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:42 /Question11_Output/part-m-00007
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:42 /Question11_Output/part-m-00008
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:42 /Question11_Output/part-m-00009
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:42 /Question11_Output/part-m-00010
-rw-r--r--   1 ebd supergroup   0 2021-04-28 12:42 /Question11_Output/part-m-00011
ebd@ebd-VirtualBox:/usr/local/bin/hadoop-3.2.2/sbin$


ebd@ebd-VirtualBox: /usr/local/bin/hadoop-3.2.2/sbin
ebd@ebd-VirtualBox:/usr/local/bin/hadoop-3.2.2/sbin$
ebd@ebd-VirtualBox:/usr/local/bin/hadoop-3.2.2/sbin$
ebd@ebd-VirtualBox:/usr/local/bin/hadoop-3.2.2/sbin$ hadoop fs -head /Question11_Output/LeftSide-m-00000
A-3866274,Bing,,2,2019-08-31 11:43:00,2019-08-31 12:36:31,38.057425,-121.261317,38.057425,-121.261317,0.0,At CA-99 - Accident.,10901.0,
N Highway 99,L,Lodi,San Joaquin,CA,95240,US,US/Pacific,KSCK,2019-08-31 11:55:00,89.0,89.0,32.0,29.9,10.0,WNW,7.0,0.0,Fair,False,False,Fa
lse,False,False,False,False,False,False,False,False,False,Day,Day,Day,Day
A-3866285,Bing,,2,2019-08-31 12:31:00,2019-08-31 13:04:33,38.532699,-121.464689,38.532699,-121.464689,0.0,At 14th Ave/12th Ave Byp - Acc
ident.,4701.0, Martin Luther King Jr Blvd,L,Sacramento,Sacramento,CA,95820,US,US/Pacific,KSAC,2019-08-31 12:53:00,89.0,89.0,35.0,29.9,10
.0,SSW,3.0,0.0,Fair,False,False,False,False,False,False,False,False,False,False,False,False,Day,Day,Day,Day
A-3866286,Bing,,2,2019-08-31 12:30:00,2019-08-31 13:22:31,39.420068,-123.807962,39.420068,-123.807962,0.0,At Fort Bragg-Willits Rd - Acc
ident.,1299.0, S Main St,L,Fort Bragg,Mendocino,CA,95437-5312,US,US/Pacific,KUKI,2019-08-31 12:56:00,91.0,91.0,21.0,29.27,10.0,VAR,3.0,0
.0,Fair,False,ebd@ebd-VirtualBox:/usr/local/bin/hadoop-3.2.2/sbin$
ebd@ebd-VirtualBox:/usr/local/bin/hadoop-3.2.2/sbin$
ebd@ebd-VirtualBox:/usr/local/bin/hadoop-3.2.2/sbin$ hadoop fs -head /Question11_Output/RightSide-m-00000
A-3866272,Bing,,2,2019-08-31 11:38:00,2019-08-31 12:12:33,37.748,-122.15031,37.748,-122.15031,0.0,At 98th Ave/Golf Links Rd - Accident.,
,I-580 W,R,Oakland,Alameda,CA,94605,US,US/Pacific,KOAK,2019-08-31 11:53:00,75.0,75.0,51.0,29.96,10.0,WSW,5.0,0.0,Fair,False,False,False,
False,False,False,False,False,False,False,False,False,Day,Day,Day,Day
A-3866273,Bing,,2,2019-08-31 11:38:00,2019-08-31 13:51:26,38.732177,-121.364635,38.732177,-121.364635,0.0,At Baseline Rd - Accident.,936
5.0, Walerga Rd,R,Roseville,Placer,CA,95747-9724,US,US/Pacific,KMCC,2019-08-31 11:45:00,84.0,84.0,37.0,29.83,10.0,NW,3.0,,Fair,False,Fal
se,False,False,False,False,False,False,False,False,False,False,False,Day,Day,Day,Day
A-3866275,Bing,,2,2019-08-31 11:41:00,2019-08-31 12:54:30,37.359543,-120.64392,37.359543,-120.64392,0.0,At CR-J18/Central Ave/Westside B
lvd - Accident.,,CA-99 N,R,Winton,Merced,CA,95388,US,US/Pacific,KMER,2019-08-31 06:15:00,,,,29.71,10.0,CALM,0.0,0.0,Fair,False,False,Fal
ebd@ebd-VirtualBox:/usr/local/bin/hadoop-3.2.2/sbin$

12. Implementing Data Analysis using Apache Pig

a) Accidents in cities belonging to US/Eastern Time zone

query1.pig

Plugins supporting *.pig files found. Install plugins Ignore extension

```
us_accidents = LOAD '/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/pig/US_Accidents_Dec20.csv' USING PigStorage(',') AS (ID:CHARARRAY, Source:CHARARRA
columns = FOREACH us_accidents GENERATE $20 as Timezone, $15 as City ;
timezone = FILTER columns BY NOT City == '' AND Timezone == 'US/Eastern';
groupby = GROUP timezone BY CONCAT (Timezone, City);
accident_count = FOREACH groupby GENERATE group, COUNT(timezone) AS count;
sorted = ORDER accident_count BY count DESC;
-- DUMP sorted;
STORE sorted INTO '/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/pig/query1_output.txt';
```

```
ebd@ebd-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
2021-04-27 01:05:12,929 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2021-04-27 01:05:12,941 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - 100% complete
2021-04-27 01:05:12,963 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.SimplePigStats - Script Statistics:

HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
3.2.2   0.16.0  ebd     2021-04-27 01:04:10     2021-04-27 01:05:12     GROUP_BY,ORDER_BY,FILTER

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime
job_local1617638199_0002        1       1       n/a     n/a     n/a     n/a     n/a     n/a     n/a     n/a     sorted  SAMPLER
job_local2081118173_0003        1       1       n/a     n/a     n/a     n/a     n/a     n/a     n/a     n/a     sorted  ORDER_BY        /home/ebd/Documents/H
ery1_output.txt,
job_local392432042_0001 49      2       n/a     n/a     n/a     n/a     n/a     n/a     n/a     n/a     accident_count,columns,groupby,timezone,us_accidents

Input(s):
Successfully read 4232542 records from: "/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/pig/US_Accidents_Dec20.csv"

Output(s):
Successfully stored 6756 records in: "/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/pig/query1_output.txt"

Counters:
Total records written : 6756
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_local392432042_0001 ->      job_local1617638199_0002,
job_local1617638199_0002        ->      job_local2081118173_0003,
job_local2081118173_0003
```

part-r-00000

```
US/EasternCharlotte 88874
US/EasternMiami 63086
US/EasternRaleigh   52871
US/EasternAtlanta   46319
US/EasternOrlando   39560
US/EasternRichmond  24337
US/EasternJacksonville  23702
US/EasternColumbia  22610
US/EasternIndianapolis  22479
US/EasternGreenville    22154
US/EasternTampa 17345
US/EasternRochester 17325
US/EasternDayton    16527
```

b) Counties in CA where accidents happened with poor visibility (Inner Join)

query1.pig | query2.pig

Plugins supporting *.pig files found. Install plugins Ignore extension

```
-- Counties in CA where accidents happened where visibility < 5

us_accidents = LOAD '/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/pig/US_Accidents_Dec20.csv' USING PigStorage(',') AS (ID:CHARARRAY, Source:CHARARRAY, TMC:DOUB

columns = FOREACH us_accidents GENERATE ID, State, Visibility;

county = FOREACH us_accidents GENERATE ID, County;

state = FILTER columns BY State == 'CA' AND NOT Visibility is null;

visibility = FILTER state BY Visibility < 5;

inner_join = JOIN county BY ID, visibility BY ID;

records = FOREACH inner_join GENERATE State, County, Visibility;

county_visibility = DISTINCT records;

STORE county_visibility INTO '/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/pig/query2_output.txt' using PigStorage(';');
```

ebd@ebd-VirtualBox: ~

```
2021-04-27 11:18:35,347 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.SimplePigStats - Script Statistics:

HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
3.2.2   0.16.0  ebd     2021-04-27 11:14:16     2021-04-27 11:18:35     HASH_JOIN,DISTINCT,FILTER

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime      MinMapTime      AvgMapTime      MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime
job_local368707957_0001 98      4       n/a     n/a     n/a     n/a     n/a     n/a     n/a     n/a     columns,county,inner_join,records,state,us_accidents
job_local561651534_0002 1       1       n/a     n/a     n/a     n/a     n/a     n/a     n/a     n/a             DISTINCT        /home/ebd/Documents/HadoopProj
ig/query2_output.txt,

Input(s):
Successfully read 4232542 records from: "/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/pig/US_Accidents_Dec20.csv"
Successfully read 4232542 records from: "/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/pig/US_Accidents_Dec20.csv"

Output(s):
Successfully stored 0 records in: "/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/pig/query2_output.txt"

Counters:
Total records written : 0
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_local368707957_0001 ->      job_local561651534_0002,
job_local561651534_0002
```

part-r-00000

```
CA;Inyo;0.75
CA;Inyo;1.0
CA;Inyo;1.8
CA;Inyo;2.0
CA;Inyo;3.0
CA;Kern;0.0
CA;Kern;0.06
CA;Kern;0.2
CA;Kern;0.25
CA;Kern;0.5
CA;Kern;0.75
CA;Kern;0.8
CA;Kern;1.0
CA;Kern;1.2
CA;Kern;1.5
```

Data Analysis using Hive

Creating and loading data

```
CREATE TABLE us_accidents(ID STRING, Source STRING, TMC DOUBLE, Severity INT, Start_Time
STRING, End_Time STRING, Start_Lat DOUBLE, Start_Lng DOUBLE, End_Lat DOUBLE, End_Lng DOUBLE,
Distance DOUBLE, Description STRING, Number DOUBLE, Street STRING, Side STRING, City STRING,
County STRING, State STRING, Zipcode STRING, Country STRING, Timezone STRING, Airport_Code
STRING, Weather_Timestamp STRING, Temperature DOUBLE, Wind_Chill DOUBLE,Humidity DOUBLE,
Pressure DOUBLE, Visibility DOUBLE, Wind_Direction STRING, Wind_Speed DOUBLE, Precipitation
DOUBLE, Weather_Condition STRING, Amenity BOOLEAN, Bump BOOLEAN, Crossing BOOLEAN, Give_Way
BOOLEAN, Junction BOOLEAN, No_Exit BOOLEAN, Railway BOOLEAN, Roundabout BOOLEAN, Station
BOOLEAN, Stop BOOLEAN, Traffic_Calming BOOLEAN, Traffic_Signal BOOLEAN, Turning_Loop BOOLEAN,
Sunrise_Sunset STRING, Civil_Twilight STRING, Nautical_Twilight STRING, Astronomical_Twilight
STRING) row format delimited fields terminated by ',';

CREATE TABLE Wind_Direction(Wind_Direction STRING, Description STRING) row format delimited
fields terminated by ',';


LOAD DATA INPATH '/US_Accidents_Dec20.csv' OVERWRITE INTO TABLE us_accidents;
LOAD DATA INPATH '/WindDirection.csv' OVERWRITE INTO TABLE Wind_Direction;
```

Analysis Queries

```
-- Cities in the State of Ohio which caused delays on a rainy day

SELECT City, COUNT(*) AS NumberOfAccidents FROM us_accidents WHERE
CHARINDEX('delays',Description) > 0 AND City <> '' AND Weather_Condition='Heavy Rain' AND
State='OH' GROUP BY City;



-- Inner join based on Wind Direction

SELECT a.Wind_Direction, b.Description, a.Wind_Speed FROM us_accidents a JOIN Wind_Direction b
ON (a.Wind_Direction=b.Wind_Direction) WHERE a.Wind_Direction <> ''



-- Maximum Temperature for each Weather Condition during accidents


SELECT Weather_Condition, MAX(Temperature) FROM us_accidents WHERE Weather_Condition <> ''
GROUP BY Weather_Condition ORDER BY MAX(Temperature) DESC;
```

Consolidated shell script to run all the map reduce jobs:

```
#!/bin/bash

# Copying the data to HDFS

hadoop fs -put ~/Downloads/US_Accidents_Dec20.csv /
hadoop fs -put ~/Downloads/WindDirection.csv /


# Simple MR Job: Number of Traffic Accidents occurred by State

hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsPr
oject-1.0-SNAPSHOT.jar Question1.DriverClass /US_Accidents_Dec20.csv /Question1_Output


# Simple MR Job: Number of Accidents near Junction VS not a Junction

hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsPr
oject-1.0-SNAPSHOT.jar Question2.DriverClass /US_Accidents_Dec20.csv /Question2_Output


# Numerical Summarization: Average, Min, Max Visibility (miles), Max StartDate Per Accident
Severity

hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsPr
oject-1.0-SNAPSHOT.jar Question3.DriverClass /US_Accidents_Dec20.csv /Question3_Output


# Numerical Summarization: Most Recent (Max) Accident Date Per City

hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsPr
oject-1.0-SNAPSHOT.jar Question4.DriverClass /US_Accidents_Dec20.csv /Question4_Output


# Numerical Summarization: Percentage of Accidents in each Month

hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsPr
oject-1.0-SNAPSHOT.jar Question5.DriverClass /US_Accidents_Dec20.csv /Question5_Output
```

Top ‘N’ Filtering Pattern: Top 10 States with Highest Number of Accidents

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question6.DriverClass /Question1_Output/part-r-00000
/Question6_Output
```

Top ‘N’ Filtering Pattern: Top 5 Accident Conducive Weather Conditions

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question7.a.DriverClass /US_Accidents_Dec20.csv
/Question7a_Output

hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question7.b.DriverClass /Question7a_Output/part-r-00000
/Question7b_Output
```

Secondary Sorting: Number of Accidents Per County Per Year

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question8.DriverClass /US_Accidents_Dec20.csv /Question8_Output
```

Partitioning: Partitioning based on Wind Directions

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question9.DriverClass /US_Accidents_Dec20.csv /WindDirection.csv
/Question9_Output
```

Inverted Index Pattern: Cities present in each County

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question10.DriverClass /US_Accidents_Dec20.csv
/Question10_Output
```

Binning: Binning based on Accidents on each side of the road (Left/Right)

```
hadoop jar
/home/ebd/Documents/HadoopProjects/USTrafficAccidentsProject/target/USTrafficAccidentsProject-1.0-SNAPSHOT.jar Question11.Driver /US_Accidents_Dec20.csv /Question11_Output
```

Appendix

Simple MR Job: Number of Traffic Accidents occurred by State

DriverClass.java

```java
public class DriverClass {
    public static void main(String args[]) throws IOException, ClassNotFoundException, InterruptedException {
        // Create a new Job
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, jobName: "AccidentsPerState");
        job.setJarByClass(DriverClass.class);

        //Telling the program which is mapper/reducer class
        job.setMapperClass(MapperClass.class);
        job.setReducerClass(ReducerClass.class);

        //Setting the InputFormat (What the data will be like)
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);

        FileSystem fs = FileSystem.get(job.getConfiguration());

        //Passing the file on which word count will be performed
        //Setting the input and output paths
        Path outDir = new Path(args[1]);
        FileOutputFormat.setOutputPath(job, outDir);
        Path inDir = new Path(args[0]);
        FileInputFormat.addInputPath(job, inDir);

        //Delete the output directory if it exists be execution
        if(fs.exists(outDir)){
            fs.delete(outDir, b: true);
        }

        System.exit(job.waitForCompletion( verbose: true) ? 1 : 0);
    }
```

MapperClass.java

```java
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class MapperClass extends Mapper<LongWritable, Text, Text, LongWritable> {

    Text state = new Text();
    LongWritable count = new LongWritable( value: 1);

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        //Convert the each line which is comma separated into String array
        String line = value.toString();
        String[] tokens = line.split( regex: ",");

        //Check if the first column is ID
        if(!tokens[0].equals("ID")) {

            //set column 17 to Text variable
            state.set(tokens[17]);
        }

        //The mapper will emit state key and 1 as count
        context.write(state, count);
    }
}
```

ReducerClass.java

```java
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class ReducerClass extends Reducer<Text, LongWritable, Text, LongWritable> {

    LongWritable count = new LongWritable();

    @Override
    protected void reduce(Text key, Iterable<LongWritable> values, Context context) throws IOException, InterruptedException {

        long sum = 0;

        // Iterating through each values and adding
        for(LongWritable val: values){
            sum+=val.get();
        }

        // Converting primitive Long to Hadoop LongWritable
        count.set(sum);

        // Emitting out state and aggregate count
        context.write(key, count);

    }
}
```

Simple MR Job: Number of Accidents near Junction VS not a Junction

DriverClass.java

```java
public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

    // Create a new Job
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, jobName: "AccidentsNearJunction");
    job.setJarByClass(DriverClass.class);

    //Telling the program which is mapper/reducer class
    job.setMapperClass(MapperClass.class);
    job.setReducerClass(ReducerClass.class);

    //Setting the InputFormat (What the data will be like)
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(LongWritable.class);

    job.setNumReduceTasks(1);

    FileSystem fs = FileSystem.get(job.getConfiguration());

    //Setting the input and output paths
    Path outDir = new Path(args[1]);
    FileOutputFormat.setOutputPath(job, outDir);
    Path inDir = new Path(args[0]);
    FileInputFormat.addInputPath(job, inDir);

    //Delete the output directory if it exists be execution
    if(fs.exists(outDir)){
        fs.delete(outDir, b: true);
    }

    System.exit(job.waitForCompletion( verbose: true) ? 1 : 0);
```

MapperClass.java

```java
public class MapperClass extends Mapper<LongWritable, Text, Text, LongWritable> {

    Text junction = new Text();
    LongWritable count = new LongWritable( value: 1);
    JunctionClass junctionMap = new JunctionClass();

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        //Convert the each line which is comma separated into String array
        String line = value.toString();
        String[] tokens = line.split( regex: ",");
        String junct = "";

        //Check if the first column is ID
        if(!tokens[0].equals("ID")) {

            //The object of class Junction Class fetches the method that maps true --> Near Junction and false --> Not Near Junction
            junct = junctionMap.getJunctionMethod(tokens[36]);

        }
        //Converting the above mapped object to Text datatype
        junction.set(junct);

        //The mapper will emit state key and 1 as count
        context.write(junction, count);
    }
}
```

ReducerClass.java

```java
package Question2;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class ReducerClass extends Reducer<Text, LongWritable, Text, LongWritable> {

    LongWritable count = new LongWritable();

    @Override
    protected void reduce(Text key, Iterable<LongWritable> values, Context context) throws IOException, InterruptedException {

        long sum = 0;

        // Iterating through each values and adding
        for(LongWritable val: values){
            sum+=val.get();
        }

        // Converting primitive Long to Hadoop LongWritable
        count.set(sum);

        // Emitting out state and count
        context.write(key, count);
    }
}
```

JunctionClass.java

```java
package Question2;

import java.util.Map;
import java.util.HashMap;

public class JunctionClass {

    //Creating a static final Map (key-value) so that one copy of variable exists and can't be reinitialize
    public static final Map<String, String> JUNCTION_MAP;

    //Creating an immutable map using a static initializer
    static {
        JUNCTION_MAP = new HashMap<String, String>();
        JUNCTION_MAP.put( k: "True",  v: "Accidents Near Junction");
        JUNCTION_MAP.put( k: "False",  v: "Accidents Not Near Junction");
        JUNCTION_MAP.put( k: "",  v: "Unknown");

    }

    //Returns the mapped value for each key (true/false)
    public String getJunctionMethod(String junction) { return JUNCTION_MAP.get(junction); }

}
```

Numerical Summarization: Avg, Min, Max Visibility, Max StartDate Per Accident Severity

DriverClass.java

```java
public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

    // Create a new Job
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, jobName: "AvgMinMaxVisibility");
    job.setJarByClass(DriverClass.class);

    //Telling the program which is mapper/reducer class
    job.setMapperClass(MapperClass.class);
    job.setReducerClass(ReducerClass.class);

    //Setting the InputFormat (What the data will be like)
    job.setInputFormatClass(TextInputFormat.class);

    //Mapper emits key which is Severity (IntWritable) and value as Custom Writable class
    job.setMapOutputKeyClass(IntWritable.class);
    job.setMapOutputValueClass(MinMaxTuple.class);

    job.setOutputKeyClass(IntWritable.class);
    job.setOutputValueClass(MinMaxTuple.class);

    job.setNumReduceTasks(1);

    //Setting the input and output paths
    Path outDir = new Path(args[1]);
    FileOutputFormat.setOutputPath(job, outDir);
    Path inDir = new Path(args[0]);
    FileInputFormat.addInputPath(job, inDir);

    //Delete the output directory if it exists be execution
    FileSystem fs = FileSystem.get(job.getConfiguration());
    if(fs.exists(outDir)){
        fs.delete(outDir, b: true);
    }
```

MapperClass.java

```java
public class MapperClass extends Mapper<LongWritable, Text, IntWritable, MinMaxTuple>{

    private final static SimpleDateFormat sdformat = new SimpleDateFormat( pattern: "yy-MM-dd");
    IntWritable s = new IntWritable();
    MinMaxTuple tuple = new MinMaxTuple();

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        //Convert the each line which is comma separated into String array
        String line = value.toString();
        String[] tokens  = line.split( regex: ",");


        if(!tokens[0].equals("ID")){

            //Checks if the Severity and Visibility are null or empty
            if(tokens[3].isEmpty() || tokens[3].equals(" ") || tokens[27].equals(" ")|| tokens[27].isEmpty()  ) {
                return;
            }

            int severity = Integer.parseInt(tokens[3]);

            //Setting the corresponding visibility in the tuple for each key
            tuple.setMaxVisibility(Float.parseFloat(tokens[27]));
            tuple.setMinVisibility(Float.parseFloat(tokens[27]));
            tuple.setAvgVisibility(Float.parseFloat(tokens[27]));
            tuple.setCount(1);

            try{
                tuple.setMaxStartDate(sdformat.parse(tokens[4]));
            }catch (ParseException e){
                e.printStackTrace();
            }
            s.set(severity);
            context.write(s ,tuple);
        }
```

ReducerClass.java

```java
public class ReducerClass extends Reducer<IntWritable, MinMaxTuple, IntWritable, MinMaxTuple> {

    MinMaxTuple tuple = new MinMaxTuple();
    @Override
    protected void reduce(IntWritable key, Iterable<MinMaxTuple> values, Context context) throws IOException, InterruptedException {

        float maxVisibility = Float.MIN_VALUE;
        float minVisibility = Float.MAX_VALUE;
        long count = 0;
        float countVisibility = 0;
        Date maxDate = null;
        //Iterating through all the values (MinMaxTuple) for each key
        for(MinMaxTuple t: values){

            if(maxDate == null) {
                maxDate = t.getMaxStartDate();
            }
            countVisibility += t.getAvgVisibility() * t.getCount();
            count += t.getCount();
            if(t.getMinVisibility() < minVisibility) {
                minVisibility = t.getMinVisibility();
            }
            if(t.getMaxVisibility() > maxVisibility) {
                maxVisibility = t.getMaxVisibility();
                maxDate = t.getMaxStartDate();
            }
        }
        //Setting the aggregated values in the tuple
        tuple.setAvgVisibility(countVisibility / count);
        tuple.setMaxVisibility(maxVisibility);
        tuple.setMinVisibility(minVisibility);
        tuple.setMaxStartDate(maxDate);
        tuple.setCount(count);

        context.write(key, tuple);
```

MinMaxTuple.java

```java
public class MinMaxTuple implements Writable {

    private float minVisibility;
    private float maxVisibility;
    private float avgVisibility;
    private long count;
    private Date maxStartDate;

    public long getCount() { return count; }

    public void setCount(long count) { this.count = count; }
    public float getMinVisibility() { return minVisibility; }
    public void setMinVisibility(float minVisibility) { this.minVisibility = minVisibility; }
    public float getMaxVisibility() { return maxVisibility; }
    public void setMaxVisibility(float maxVisibility) { this.maxVisibility = maxVisibility; }
    public float getAvgVisibility() { return avgVisibility; }
    public void setAvgVisibility(float avgVisibility) { this.avgVisibility = avgVisibility; }
    public Date getMaxStartDate() { return maxStartDate; }
    public void setMaxStartDate(Date maxStartDate) { this.maxStartDate = maxStartDate; }

    //Serialize the fields of this object to out
    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeFloat(minVisibility);
        dataOutput.writeFloat(maxVisibility);
        dataOutput.writeFloat(avgVisibility);
        dataOutput.writeLong(count);
        dataOutput.writeLong(maxStartDate.getTime());
    }
    //Deserialize the fields of this object from in
    @Override
    public void readFields(DataInput dataInput) throws IOException {
        avgVisibility = dataInput.readFloat();
        minVisibility = dataInput.readFloat();
        maxVisibility = dataInput.readFloat();
        count = dataInput.readLong();
        maxStartDate =  new Date(dataInput.readLong());
    @Override
    public String toString() {
        return  "Minimum Visibility= " + minVisibility +
                " | Maximum Visibility= " + maxVisibility +
                " | Average Visibility= " + avgVisibility +
                " | Maximum Start Date= " + maxStartDate;
    }
}
```

Numerical Summarization: Most Recent (Max) Accident Date Per City

DriverClass.java

```
16  public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {
17
18      // Create a new Job
19      Configuration conf = new Configuration();
20      Job job = Job.getInstance(conf, jobName: "LastestAccidentDatePerCity");
21      job.setJarByClass(DriverClass.class);
22
23      //Telling the program which is mapper/reducer class
24      job.setMapperClass(MapperClass.class);
25      job.setReducerClass(ReducerClass.class);
26
27      //Setting the InputFormat (What the data will be like)
28      job.setInputFormatClass(TextInputFormat.class);
29
30      //Mapper emits key which is City (Text) and value as Custom Writable class
31      job.setMapOutputKeyClass(Text.class);
32      job.setMapOutputValueClass(MinMaxTuple.class);
33
34      job.setOutputKeyClass(Text.class);
35      job.setOutputValueClass(MinMaxTuple.class);
36
37      job.setNumReduceTasks(1);
38
39      //Setting the input and output paths
40      Path outDir = new Path(args[1]);
41      FileOutputFormat.setOutputPath(job, outDir);
42      Path inDir = new Path(args[0]);
43      FileInputFormat.addInputPath(job, inDir);
44
45      //Delete the output directory if it exists be execution
46      FileSystem fs = FileSystem.get(job.getConfiguration());
47      if (fs.exists(outDir)) {
48          fs.delete(outDir, b: true);
49      }
50
51      System.exit(job.waitForCompletion( verbose: true) ? 1 : 0);
```

MapperClass.java

```java
import ...

public class MapperClass extends Mapper<LongWritable, Text, Text, MinMaxTuple> {

    private final static SimpleDateFormat sdformat = new SimpleDateFormat( pattern: "yy-MM-dd");
    Text city = new Text();
    MinMaxTuple tuple = new MinMaxTuple();

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        //Convert the each line which is comma separated into String array
        String line = value.toString();
        String[] tokens  = line.split( regex: ",");

        //Check if the first column is ID
        if(!tokens[0].equals("ID")){

            //Checks if the City and Date are null or empty
            if(tokens[15].isEmpty() || tokens[15].equals(" ") || tokens[4].equals(" ")|| tokens[4].isEmpty() ) {
                return;
            }
            //Setting city
            String c = tokens[15];

            try{
                tuple.setMaxStartDate(sdformat.parse(tokens[4]));
            }catch (ParseException e){
                e.printStackTrace();
            }

            city.set(c);
            context.write(city ,tuple);
        }
```

ReducerClass.java

```java
package Question4;

import ...

public class ReducerClass extends Reducer<Text, MinMaxTuple, Text, MinMaxTuple> {

    MinMaxTuple tuple = new MinMaxTuple();

    @Override
    protected void reduce(Text key, Iterable<MinMaxTuple> values, Context context) throws IOException, InterruptedException {

        Date maxDate = null;

        for(MinMaxTuple t: values){

            if(maxDate == null) {
                maxDate = t.getMaxStartDate();
            }

            //Checking the Max date then the previously set
            if(maxDate.compareTo(t.getMaxStartDate())<=0) {
                maxDate = t.getMaxStartDate();
            }

        }

        tuple.setMaxStartDate(maxDate);
        context.write(key, tuple);

    }
}
```

MinMaxTuple.java

```java
package Question4;

import ...

public class MinMaxTuple implements Writable {

    private Date maxStartDate;

    public Date getMaxStartDate() { return maxStartDate; }

    public void setMaxStartDate(Date maxStartDate) { this.maxStartDate = maxStartDate; }

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeLong(maxStartDate.getTime());
    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {
        maxStartDate =  new Date(dataInput.readLong());
    }

    @Override
    public String toString() { return "    Latest Accident Occurred On: " + maxStartDate; }
}
```

Numerical Summarization: Percentage of Accidents in each Month

DriverClass.java

```java
public class DriverClass {

    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

        // Create a new Job
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, jobName: "PercentOfAccidentsPerMonth");
        job.setJarByClass(DriverClass.class);

        //Telling the program which is mapper/reducer class
        job.setMapperClass(MapperClass.class);
        job.setReducerClass(ReducerClass.class);

        //Setting the InputFormat (What the data will be like)
        job.setInputFormatClass(TextInputFormat.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        job.setNumReduceTasks(1);

        //Setting the input and output paths
        Path outDir = new Path(args[1]);
        FileOutputFormat.setOutputPath(job, outDir);
        Path inDir = new Path(args[0]);
        FileInputFormat.addInputPath(job, inDir);

        //Delete the output directory if it exists be execution
        FileSystem fs = FileSystem.get(job.getConfiguration());
        if (fs.exists(outDir)) {
            fs.delete(outDir, b: true);
        }

        System.exit(job.waitForCompletion( verbose: true) ? 1 : 0);
```

MapperClass.java

```java
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class MapperClass extends Mapper<LongWritable, Text, Text, IntWritable> {

        Date date = new Date();
        Text month = new Text();
        IntWritable one = new IntWritable( value: 1);

        SimpleDateFormat sdformat = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss");
        SimpleDateFormat monthformat = new SimpleDateFormat( pattern: "MMMMM");

        @Override
        protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

            //Convert the each line which is comma separated into String array
            String line = value.toString();
            String[] tokens  = line.split( regex: ",");

            //Check if the first column is ID
            if(tokens[0].equals("ID")) {
                String startDate = tokens[4];

                try {
                    date = sdformat.parse(startDate);
                } catch (ParseException e) {
                    e.printStackTrace();
                }
            }
            month.set(monthformat.format(date));
            context.write(month, one);
        }
}
```

ReducerClass.java

```java
package Question5;

import ...

public class ReducerClass extends Reducer<Text, IntWritable, Text, Text> {

    Text percent = new Text();

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

        // The total length of records in the dataset
        int total = 3513617;
        int sum = 0;

        for(IntWritable v : values) {
            sum += v.get();
        }
        //Calculating the percentage
        double percentage = ((double) sum / total) * 100;
        //Formatting into a string
        percent.set(String.format("%.2f", percentage) + "%");

        context.write(key, percent);
    }
}
```

Top ‘N’ Filtering Pattern: Top 10 States with Highest Number of Accidents

DriverClass.java

```java
public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

    // Create a new Job
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, jobName: "Top10AccidentStates");
    job.setJarByClass(DriverClass.class);

    //Telling the program which is mapper/reducer class
    job.setMapperClass(MapperClass.class);
    job.setReducerClass(ReducerClass.class);

    //Setting the InputFormat (What the data will be like)
    job.setInputFormatClass(TextInputFormat.class);

    //We use nullWritable class as key
    job.setMapOutputKeyClass(NullWritable.class);
    job.setMapOutputValueClass(Text.class);

    job.setOutputKeyClass(NullWritable.class);
    job.setOutputValueClass(Text.class);

    job.setNumReduceTasks(1);

    //Setting the input and output paths
    Path outDir = new Path(args[1]);
    FileOutputFormat.setOutputPath(job, outDir);
    Path inDir = new Path(args[0]);
    FileInputFormat.addInputPath(job, inDir);

    //Delete the output directory if it exists be execution
    FileSystem fs = FileSystem.get(job.getConfiguration());
    if (fs.exists(outDir)) {
        fs.delete(outDir, b: true);
    }

    System.exit(job.waitForCompletion( verbose: true) ? 1 : 0);
```

MapperClass.java

```java
package Question6;

import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class MapperClass extends Mapper<Object, Text, NullWritable, Text> {

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

        context.write(NullWritable.get(),value);
    }
}
```

ReducerClass.java

```java
public class ReducerClass extends Reducer<NullWritable, Text, NullWritable, Text> {

    static class SortingClass implements Comparator<Integer> {

        @Override
        public int compare(Integer o1, Integer o2) {
            return o2.compareTo(o1);
        }
    }

    //TreeMap save the data of top n
    private TreeMap<Integer, Text> countTreeMap = new TreeMap<>(new SortingClass());

    @Override
    protected void reduce(NullWritable key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

        for (Text value : values) {

            //We use the previous formed output from Question1 which is in form "State \t Count"
            String line = value.toString();
            String[] tokens  = line.split( regex: "\t");

            //Second column is the count
            int count = Integer.parseInt(tokens[1]);

            countTreeMap.put(count, new Text(value));

            if (countTreeMap.size() > 10)
                countTreeMap.remove(countTreeMap.lastKey());
        }

        for (Text t : countTreeMap.values())
            context.write(NullWritable.get(), t);
    }
}
```

Top ‘N’ Filtering Pattern: Top 5 Accident Conducive Weather Conditions

DriverClass.java

```java

public class DriverClass {

    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf,  jobName: "Top5WeatherConditions");
        job.setJarByClass(DriverClass.class);

        job.setMapperClass(MapperClass.class);
        job.setReducerClass(ReducerClass.class);

        job.setInputFormatClass(TextInputFormat.class);

        job.setMapOutputKeyClass(NullWritable.class);
        job.setMapOutputValueClass(Text.class);

        job.setOutputKeyClass(NullWritable.class);
        job.setOutputValueClass(Text.class);

        job.setNumReduceTasks(1);

        Path outDir = new Path(args[1]);
        FileOutputFormat.setOutputPath(job, outDir);
        Path inDir = new Path(args[0]);
        FileInputFormat.addInputPath(job, inDir);

        FileSystem fs = FileSystem.get(job.getConfiguration());
        if (fs.exists(outDir)) {
            fs.delete(outDir,  b: true);
        }

        System.exit(job.waitForCompletion( verbose: true) ? 1 : 0);

    }
}
```

MapperClass.java

```java
package Question7.b;

import ...

public class MapperClass extends Mapper<Object, Text, NullWritable, Text> {

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

        context.write(NullWritable.get(),value);
    }
}
```

ReducerClass.java

```java
package Question7.b;

import ...

public class ReducerClass extends Reducer<NullWritable, Text, NullWritable, Text> {

    static class SortingClass implements Comparator<Integer> {

        @Override
        public int compare(Integer o1, Integer o2) { return o2.compareTo(o1); }
    }

    private TreeMap<Integer, Text> countTreeMap = new TreeMap<>(new SortingClass());

    @Override
    protected void reduce(NullWritable key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

        for (Text value : values) {

            String[] tokens = value.toString().split( regex: "\t");
            int count = Integer.parseInt(tokens[1]);

            countTreeMap.put(count, new Text(value));

            if (countTreeMap.size() > 5)
                countTreeMap.remove(countTreeMap.lastKey());
        }

        for (Text t : countTreeMap.values())
            context.write(NullWritable.get(), t);
    }
}
```

Secondary Sorting: Number of Accidents Per County Per Year

DriverClass.java

```
19
20          // Create a new Job
21          Configuration config = new Configuration();
22          Job job = Job.getInstance(config,  jobName: "SecondarySorting");
23          job.setJarByClass(DriverClass.class);
24
25          job.setGroupingComparatorClass(NaturalKeyGroupComparator.class);  // Group comparator - this takes care of the grouping into the iterable t
26          job.setSortComparatorClass(CompositeKeySortComparator.class);    // SortComparator -- does secondary sorting
27          job.setPartitionerClass(PartitionerClass.class);        // Partitioner - this makes sure that the mapper output goes to relevant reducer
28
29          //Telling the program which is mapper/reducer class
30          job.setMapperClass(MapperClass.class);
31          job.setReducerClass(ReducerClass.class);
32
33          //Setting the InputFormat (What the data will be like)
34          job.setInputFormatClass(TextInputFormat.class);
35          job.setOutputFormatClass(TextOutputFormat.class);
36
37          job.setMapOutputKeyClass(WritableClass.class);       // for custom class how the reducer has to group
38          job.setMapOutputValueClass(IntWritable.class);
39
40          job.setOutputKeyClass(Text.class);
41          job.setOutputValueClass(IntWritable.class);
42
43          //Setting the input and output paths
44          Path outDir = new Path(args[1]);
45          FileOutputFormat.setOutputPath(job, outDir);
46          Path inDir = new Path(args[0]);
47          FileInputFormat.addInputPath(job, inDir);
48
49          //Number of reducers set to 5 -- based on 5 partitions
50          job.setNumReduceTasks(5);
51
52          //Delete the output directory if it exists be execution
53          FileSystem fs = FileSystem.get(job.getConfiguration());
54          if (fs.exists(outDir)) {
55              fs.delete(outDir,  b: true);
```

MapperClass.java

```java
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;

public class MapperClass extends Mapper<Object, Text, WritableClass, IntWritable> {

    SimpleDateFormat sdformat = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss");
    SimpleDateFormat yearformat = new SimpleDateFormat( pattern: "yyyy");

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

        String year = null;
        String line = value.toString();
        String[] tokens = line.split( regex: ",");

        if (!tokens[0].equals("ID")) {

            if (tokens[4].isEmpty() || tokens[16].isEmpty() || tokens[4].equals(" ") || tokens[16].equals(" ")) {
                return;
            }

            String county = tokens[16];

            try {
                year = yearformat.format(sdformat.parse(tokens[4]));
            } catch (ParseException e) {
                e.printStackTrace();
            }

            WritableClass writableObj = new WritableClass(county, year);
            context.write(writableObj, new IntWritable( value: 1));
        }
    }
}
```

ReducerClass.java

```java
package Question8;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class ReducerClass extends Reducer<WritableClass, IntWritable, Text, IntWritable> {

    Text text = new Text();
    IntWritable count = new IntWritable();

    @Override
    protected void reduce(WritableClass key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

        int sum = 0;

        for(IntWritable v : values) {
            sum += v.get();
        }
        text.set("County: "+key.getCounty()+ "| Year: " + key.getYear() + " | Accidents: ");

        count.set(sum);
        context.write(text, count);

    }
}
```

WritableClass.java

```java
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

public class WritableClass implements WritableComparable<WritableClass> {

    private String county;
    private String year;

    public WritableClass(String state, String year) {...}
    public WritableClass() { super(); }
    public String getCounty() { return county; }
    public void setCounty(String county) { this.county = county; }
    public String getYear() { return year; }
    public void setYear(String year) { this.year = year; }

    @Override
    public int compareTo(WritableClass o) {...}

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeUTF(county);
        dataOutput.writeUTF(year);
    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {
        county = dataInput.readUTF();
        year = dataInput.readUTF();
    }

    @Override
    public String toString() {
        return "county='" + county + '\'' +
                ", year='" + year + '\'';
    }
}
```

PartitionerClass.java

```java
package Question8;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Partitioner;

public class PartitionerClass extends Partitioner<WritableClass, IntWritable> {

    @Override
    public int getPartition(WritableClass compositeKeyWritable, IntWritable intWritable, int i) {
        return Integer.parseInt(compositeKeyWritable.getYear()) % i;
    }
}
```

NaturalKeyGroupComparator.java

```java
package Question8;

import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class NaturalKeyGroupComparator extends WritableComparator {

    public NaturalKeyGroupComparator() { super(WritableClass.class, createInstances: true); }

    //Which reducer to send
    @Override
    public int compare(WritableComparable a, WritableComparable b) {
        WritableClass w1 = (WritableClass) a;
        WritableClass w2 = (WritableClass) b;

        return w1.getCounty().compareTo(w2.getCounty());

    }
}
```

CompositeKeySortComparator.java

```java
package Question8;

import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class CompositeKeySortComparator extends WritableComparator {

    protected CompositeKeySortComparator() { super(WritableClass.class, createInstances: true); }

    @Override
    public int compare(WritableComparable a, WritableComparable b) {

        WritableClass w1 = (WritableClass) a;
        WritableClass w2 = (WritableClass) b;

        int result = w1.getYear().compareTo(w2.getYear());

        if(result == 0) {
            return w1.getCounty().compareTo(w2.getCounty());
        }

        return result;
    }

}
```

Partitioning: Partitioning based on Wind Directions

DriverClass.java

```java

public class DriverClass {

    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {

        Configuration configuration = new Configuration();
        Job job = Job.getInstance(configuration, jobName: "PartitionWindDirection");
        job.setJarByClass(DriverClass.class);

        job.getConfiguration().set("join.type", "inner");
        job.setReducerClass(ReducerClass.class);

        //Partitions the data that is sent to the reducer
        job.setPartitionerClass(PartitionerClass.class);

        job.setOutputFormatClass(TextOutputFormat.class);

        //Used MultipleOutputs as we are taking two data
        MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class, MapperClass.class);
        MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class, WindDirectionClass.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        Path outDir = new Path(args[2]);
        FileOutputFormat.setOutputPath(job, outDir);

        job.setNumReduceTasks(23);

        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(outDir)){
            fs.delete(outDir, b: true);
        }

        System.exit(job.waitForCompletion( verbose: true) ? 0 : 2);
    }
}
```

MapperClass.java

```java
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class MapperClass extends Mapper<LongWritable, Text, Text, Text> {

    Text outKey = new Text();
    Text outValue = new Text();
    boolean first = true;

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        String line = value.toString();
        String[] tokens  = line.split( regex: ",");

        if(tokens[28].equals(" ") || tokens[28].isEmpty() || tokens[0].equals("ID")) {
            return;
        }
        //Setting the wind direction from main dataset
        String windDir = tokens[28];

        //Converting to hadoop datatype
        outKey.set(windDir);

        //Setting the whole tuple as value and @ to identify its the tuple from first dataset
        outValue.set('@' +value.toString());

        context.write(outKey, outValue);
    }
}
```

ReducerClass.java

```
    private ArrayList<Text> list = new ArrayList<Text>();
    private ArrayList<Text> windDirList = new ArrayList<Text>();

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

        list.clear();
        windDirList.clear();

        //Segregates the tuples from each dataset into arraylist based on what it was tagged with.
        for (Text text : values) {
            if (text.charAt(0) == '@') {
                list.add(new Text(text.toString().substring(1)));
            } else if (text.charAt(0) == '#') {
                windDirList.add(new Text(text.toString().substring(1)));
            }
        }

        innerJoinFunction(context);
    }

    public void innerJoinFunction(Context context) throws IOException, InterruptedException {

        String join = context.getConfiguration().get("join.type");

        //Checks the join type from the one set in Driver Class
        if (join.equalsIgnoreCase( anotherString: "inner")) {
            if (!list.isEmpty() && !windDirList.isEmpty()) {

                for (Text dir : windDirList) {

                    for (Text mainTuple : list) {
                        context.write(dir, mainTuple);
                    }
                }
            }
        }
```

PartitionerClass.java

```
package Question9;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Partitioner;

public class PartitionerClass extends Partitioner<Text, Text> {

    @Override
    public int getPartition(Text text, Text text2, int i) {
        return text.hashCode() % i;
    }
}
```

WindDirectionClass.java

```java
package Question7;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class WindDirectionClass extends Mapper<LongWritable, Text, Text, Text> {

    Text outKey = new Text();
    Text outValue = new Text();

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        String line = value.toString();
        String[] tokens  = line.split( regex: ",");

        //Sets the columns from wind dataset
        String description = tokens[1];
        String windDir = tokens[0];

        outKey.set(windDir);

        //Setting the description as value and # to identify its from second dataset
        outValue.set("#" + description);

        context.write(outKey, outValue);
    }
}
```

Inverted Index Pattern: Cities present in each County

DriverClass.java

```java
public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

    //Create a job
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, jobName: "");
    job.setJarByClass(DriverClass.class);

    //Telling the program which is mapper/reducer class
    job.setMapperClass(MapperClass.class);
    job.setReducerClass(ReducerClass.class);

    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);

    //Setting the InputFormat (What the data will be like)
    job.setInputFormatClass(TextInputFormat.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    //Set number of reducers to 1
    job.setNumReduceTasks(1);

    //Setting the input and output paths
    Path outDir = new Path(args[1]);
    FileOutputFormat.setOutputPath(job, outDir);
    Path inDir = new Path(args[0]);
    FileInputFormat.addInputPath(job, inDir);

    //Delete the output directory if it exists be execution
    FileSystem fs = FileSystem.get(job.getConfiguration());
    if(fs.exists(outDir)){
        fs.delete(outDir, b: true);
    }
    //wait job to finish and then exists
    System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
}
```

MapperClass.java

```java
package Question10;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class MapperClass extends Mapper<LongWritable, Text, Text, Text> {

    Text city = new Text();
    Text county = new Text();

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

        String co = "";
        String ci = "";
        String line = value.toString();
        String[] tokens  = line.split( regex: ",");

        if(!tokens[0].equals("ID")) {
            co = tokens[16];
            ci = tokens[15];
        }
        city.set(ci);
        county.set(co);
        context.write(county, city);

    }
}
```

ReducerClass.java

```java
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
import java.util.HashSet;

public class ReducerClass extends Reducer<Text, Text, Text, Text> {

    //Random storage of input
    HashSet<String> hs = new HashSet<>();

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

        hs.clear();
        StringBuffer sb = new StringBuffer("");

        //Put all cities in hashset
        for(Text v: values){
            hs.add(v.toString());
        }
        //Iterate hashset
        for(String v: hs) {
            sb.append(v);
            sb.append(" , ");
        }

        context.write(key, new Text(sb.toString()));
    }
}
```

Binning: Binning based on Accidents on each side of the road (Left/Right)

```java
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {

        //Create a job
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf,  jobName: "BiningBasedOnSides");

        //Binning splits data up in the map phase instead of in the partitioner
        //Hence not needing a reducer
        job.setJarByClass(Driver.class);
        job.setMapperClass(MapperClass.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(NullWritable.class);

        //MultipleOutputs class sets up the job's output to write multiple distinct files
        MultipleOutputs.addNamedOutput(job,  namedOutput: "RoadSideBins", TextOutputFormat.class, Text.class, IntWritable.class);
        MultipleOutputs.setCountersEnabled(job,  enabled: true);

        job.setNumReduceTasks(0);

        //Setting the input and output paths
        Path outDir = new Path(args[1]);
        FileOutputFormat.setOutputPath(job, outDir);
        Path inDir = new Path(args[0]);
        FileInputFormat.addInputPath(job, inDir);

        //Delete the output directory if it exists be execution
        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(outDir)){
            fs.delete(outDir,  b: true);
        }

        System.exit(job.waitForCompletion( verbose: true) ? 0 : 1);
    }
}
```

MapperClass.java

```java

public class MapperClass extends Mapper<Object, Text, Text, NullWritable> {

    private MultipleOutputs<Text, NullWritable> multipleOutputs =null;

    @Override
    protected void setup(Context context) throws IOException, InterruptedException {
        multipleOutputs = new MultipleOutputs(context);
    }

    @Override
    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

        String line = value.toString();
        String[] tokens  = line.split( regex: ",");

        if(!tokens[0].equals("ID")) {
            return;
        }
        if(!tokens[14].equalsIgnoreCase( anotherString: "")) {
            String roadSide = tokens[14];
            //bin name, key, value, out put bin name
            if (roadSide.equalsIgnoreCase( anotherString: "L"))
                multipleOutputs.write( namedOutput: "RoadSideBins", value, NullWritable.get(),  baseOutputPath: "LeftSide");
            if (roadSide.equalsIgnoreCase( anotherString: "R"))
                multipleOutputs.write( namedOutput: "RoadSideBins", value, NullWritable.get(),  baseOutputPath: "RightSide");
        }
    }

    //After mapper it runs and closes the prog
    @Override
    protected void cleanup(Context context) throws IOException, InterruptedException {
        multipleOutputs.close();
    }

}
```

References

https://timepasstechies.com/category/programming/data-analytics/mapreduce/
https://arxiv.org/abs/1906.05409
https://arxiv.org/abs/1909.09638
https://smoosavi.org/datasets/us_accidents
https://github.com/aamend/hadoop-mapreduce/blob/master/design-pattern/src/main/java/com/aamend/hadoop/mapreduce/designpattern/join/ReduceSideJoin.java