INFO7250 ENGINEERING BIG DATA SYSTEMS

(FALL-2020)

Cycle Share Dataset Analysis Using Big Data Techniques

PROJECT REPORT

Professor: Yusuf Ozbek

Submitted by

Vidya Ravi Mohana Reddy

Submitted on

12-17-2020

Table of Contents

Analysis covered in this report

Analysis	Description	Big-Data Process
1	Number of trips by month-year	MapReduce
2	Min, Max and Average duration of trips from each station	MapReduce (Numerical summarization)
3	Total number of trips per station by year	MapReduce (Secondary Sorting)
4	Top 5 busy stations by month	MapReduce (Secondary sorting, Chaining, and Filtering. Uses Top n Filtering Pattern)
5	Most active age groups	MapReduce
6	Number of trips in a day from each station and the corresponding weather on that day	MapReduce (Reduce Side Joins Pattern)
7	Custom MapReduce algorithm to find the top 10 most busy routes	MapReduce (Chaining and Top n using a custom algorithm)
8	Count membership by gender	MapReduce (Counting with Counters)
9	Count of all the trips by station	Apache Hive
10	Top 5 busiest hours of the day	Apache Hive
11	Total number of trips that lasted more than 30mins (1800sec) in each station	Apache Pig
12	Number of trips in a day from each station and the corresponding weather on that day – joins patterns	Apache Pig (Join Pattern)

Overview of the Dataset

Dataset Source

The dataset is downloaded from Kaggle - https://www.kaggle.com/pronto/cycle-share-dataset?select=weather.csv

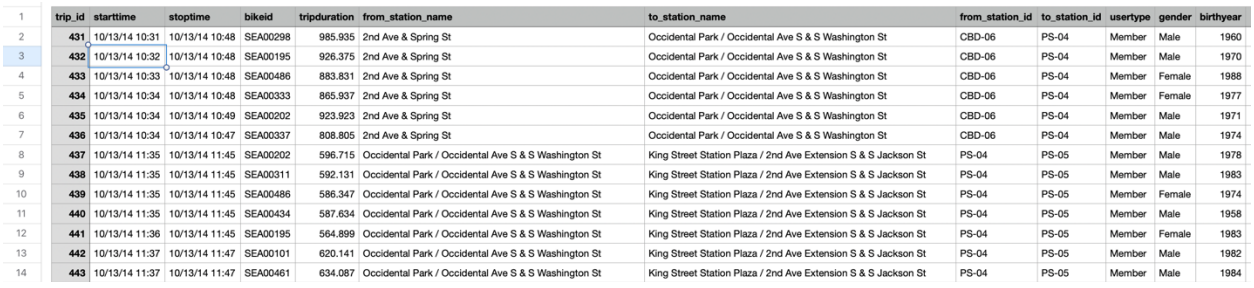
The Pronto Cycle Share system (**Pronto!**) was located in Seattle. It was operational between 2014 and 2017 and at its peak consisted of 500 bikes and 54 stations. The dataset provides open data on individual trips, stations, and daily weather collected by Pronto while it was operational. The dataset is divided into 3 files – *trips.csv*, *stations.csv*, and *weather.csv*.

trips.csv

File size: 254,814 records.

Data in this file:

Column	***Description***
trip_id	Unique ID for a trip
starttime	Day and time a trip started, in PST
stoptime	Day and time trip ended, in PST
bikeid	Unique ID of the bike
tripduration	Time of trip in seconds
from*station*name	Name of station where trip originated
to*station*name	Name of station where trip terminated
from*station*id	Unique ID of station where trip originated
to*station*id	Unique ID of station where trip terminated
usertype	"Short-Term Pass Holder" is a rider who purchased a 24-Hour or 3-Day Pass; "Member" is a rider who purchased a Monthly or an Annual Membership
gender	Gender of the member
birthyear	Birth year of the member



trip_id	starttime	stoptime	bikeid	tripduration	from_station_name	to_station_name	from_station_id	to_station_id	usertype	gender	birthyear
431	10/13/14 10:31	10/13/14 10:48	SEA00298	985.935	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washington St	CBD-06	PS-04	Member	Male	1960
432	10/13/14 10:32	10/13/14 10:48	SEA00195	926.375	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washington St	CBD-06	PS-04	Member	Male	1970
433	10/13/14 10:33	10/13/14 10:48	SEA00486	883.831	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washington St	CBD-06	PS-04	Member	Female	1988
434	10/13/14 10:34	10/13/14 10:48	SEA00333	865.937	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washington St	CBD-06	PS-04	Member	Female	1977
435	10/13/14 10:34	10/13/14 10:49	SEA00202	923.923	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washington St	CBD-06	PS-04	Member	Male	1971
436	10/13/14 10:34	10/13/14 10:47	SEA00337	808.805	2nd Ave & Spring St	Occidental Park / Occidental Ave S & S Washington St	CBD-06	PS-04	Member	Male	1974
437	10/13/14 11:35	10/13/14 11:45	SEA00202	596.715	Occidental Park / Occidental Ave S & S Washington St	King Street Station Plaza / 2nd Ave Extension S & S Jackson St	PS-04	PS-05	Member	Male	1978
438	10/13/14 11:35	10/13/14 11:45	SEA00311	592.131	Occidental Park / Occidental Ave S & S Washington St	King Street Station Plaza / 2nd Ave Extension S & S Jackson St	PS-04	PS-05	Member	Male	1983
439	10/13/14 11:35	10/13/14 11:45	SEA00486	586.347	Occidental Park / Occidental Ave S & S Washington St	King Street Station Plaza / 2nd Ave Extension S & S Jackson St	PS-04	PS-05	Member	Female	1974
440	10/13/14 11:35	10/13/14 11:45	SEA00434	587.634	Occidental Park / Occidental Ave S & S Washington St	King Street Station Plaza / 2nd Ave Extension S & S Jackson St	PS-04	PS-05	Member	Male	1958
441	10/13/14 11:36	10/13/14 11:45	SEA00195	564.899	Occidental Park / Occidental Ave S & S Washington St	King Street Station Plaza / 2nd Ave Extension S & S Jackson St	PS-04	PS-05	Member	Female	1983
442	10/13/14 11:37	10/13/14 11:47	SEA00101	620.141	Occidental Park / Occidental Ave S & S Washington St	King Street Station Plaza / 2nd Ave Extension S & S Jackson St	PS-04	PS-05	Member	Male	1982
443	10/13/14 11:37	10/13/14 11:47	SEA00461	634.087	Occidental Park / Occidental Ave S & S Washington St	King Street Station Plaza / 2nd Ave Extension S & S Jackson St	PS-04	PS-05	Member	Male	1984

stations.csv

File size: 58 records

Data in the file:

Column	***Description***
station_id	Unique ID for a station
name	Name of the station
lat	Latitude of station
long	Longitude of station
install_date	Date the station was placed in service
install_dockcount	Number of docks at each station on the installation date
modification_date	Date the station was modified, resulting in a change in location or dock count
current_dockcount	Number of docks at each station as on 8/31/2016
decommission_date	Date that station was placed out of service

station_id	name	lat	long	install_date	install_dockcount	modification_date	current_dockcount	decommission_date
BT-01	3rd Ave & Broad St	47.618418	-122.350964	10/13/2014	18		18	
BT-03	2nd Ave & Vine St	47.615829	-122.348564	10/13/2014	16		16	
BT-04	6th Ave & Blanchard St	47.616094	-122.341102	10/13/2014	16		16	
BT-05	2nd Ave & Blanchard St	47.61311	-122.344208	10/13/2014	14		14	
CBD-03	7th Ave & Union St	47.610731	-122.332447	10/13/2014	20		20	
CBD-04	Union St & 4th Ave	47.609221	-122.335596	7/27/2015	18		18	
CBD-05	1st Ave & Marion St	47.604058	-122.3358	10/13/2014	20		20	
CBD-06	2nd Ave & Spring St	47.60595	-122.335768	10/13/2014	20	11/9/2015	18	
CBD-07	City Hall / 4th Ave & James St	47.603509	-122.330409	10/13/2014	20		20	
CBD-13	2nd Ave & Pine St	47.610185	-122.339641	10/13/2014	18		18	
CD-01	12th Ave & E Yesler Way	47.602103	-122.316923	5/22/2015	16	8/9/2016	0	8/9/2016
CH-01	Summit Ave & E Denny Way	47.618633	-122.325249	10/13/2014	16		16	
CH-02	E Harrison St & Broadway Ave E	47.622063	-122.321251	10/13/2014	18	2/24/2015	20	
CH-03	Summit Ave E & E Republican St	47.623367	-122.325279	10/13/2014	16		16	
CH-05	15th Ave E & E Thomas St	47.620712	-122.312805	10/13/2014	16		16	

weather.csv

File size: 690 records

Data in the file:

Column	***Description***
Date	Date in MM/DD/YYYY format
Max_Temperature_F	Max temperature of the day
Mean_Temperature_F	Mean temperature of the day
Min_TemperatureF	Min temperature of the day
Max_Dew_Point_F	Max dew point of the day
MeanDew_Point_F	Mean dew point of the day
Min_Dewpoint_F	Min dew point of the day
Max_Humidity	Max humidity of the day
Mean_Humidity	Mean humidity of the day
Min_Humidity	Min humidity of the day
Max_Sea_Level_Pressure_In	Max pressure level of the day
Mean_Sea_Level_Pressure_In	Mean pressure level of the day
Min_Sea_Level_Pressure_In	Min pressure level of the day
Max_Visibility_Miles	Max visibility in miles
Mean_Visibility_Miles	Mean visibility in miles
Min_Visibility_Miles	Min visibility in miles
Max_Wind_Speed_MPH	Max wind speed
Mean_Wind_Speed_MPH	Mean wind speed
Max_Gust_Speed_MPH	Max wind speed
Precipitation_In	Precipitation
Events	Event of a day (rain, fog, snow)

Date	Max_Temperature_F	Mean_Temperature_F	Min_TemperatureF	Max_Dew_Point_F	MeanDew_Point_F	Min_Dewpoint_F	Max_Humidity	Mean_Humidity	Min_Humidity	Max_Sea_Level_Pressure_In	Mean_Sea_Level_Pressure_In	Min_Sea_Level_Pressure_In	Max_Visibility_Miles	Mean_Visibility_Miles	Min_Visibility_Miles
10/13/2014	71	62	54	55	51	46	87	68	46	30.03	29.79	29.65	10	10	4
10/14/2014	63	59	55	52	51	50	88	78	63	29.84	29.75	29.54	10	9	3
10/15/2014	62	58	54	53	50	46	87	77	67	29.98	29.71	29.51	10	9	3
10/16/2014	71	61	52	49	46	42	83	61	36	30.03	29.95	29.81	10	10	10
10/17/2014	64	60	57	55	51	41	87	72	46	29.83	29.78	29.73	10	10	6
10/18/2014	68	64	59	59	57	55	90	83	68	29.96	29.9	29.8	10	8	2
10/19/2014	73	64	55	57	55	53	94	74	52	29.8	29.73	29.67	10	10	6
10/20/2014	66	60	55	57	54	50	90	78	67	29.8	29.74	29.69	10	10	5
10/21/2014	64	58	55	52	49	46	87	70	58	29.93	29.85	29.74	10	10	6
10/22/2014	60	58	57	55	53	48	88	81	67	29.73	29.68	29.56	10	6	2
10/23/2014	62	55	50	49	47	44	86	76	62	30.01	29.83	29.71	10	10	10
10/24/2014	60	56	51	50	47	44	86	75	60	30.05	29.96	29.74	10	10	8
10/25/2014	64	58	52	53	49	44	87	78	58	29.74	29.48	29.26	10	10	6
10/26/2014	59	52	48	48	44	42	86	71	62	30.15	29.98	29.76	10	10	10
10/27/2014	62	54	45	46	43	41	87	72	48	30.17	30.09	29.98	10	10	10
10/28/2014	62	56	51	54	50	45	88	80	72	29.96	29.88	29.84	10	8	2
10/29/2014	64	59	54	54	51	50	88	76	60	30.14	30.06	29.94	10	10	9
10/30/2014	62	58	55	55	53	50	88	82	75	30.01	29.83	29.74	10	9	2
10/31/2014	59	54	52	52	49	46	89	83	67	29.88	29.79	29.69	10	8	2
11/1/2014	55	52	48	48	45	42	89	78	62	30.05	29.92	29.84	10	10	7
11/2/2014	57	51	45	52	47	42	90	82	62	30.15	30.1	30.06	10	9	4
11/3/2014	60	58	55	53	51	48	84	80	75	30.12	30.08	30.01	10	9	3
11/4/2014	61	58	55	54	51	48	90	79	67	30.29	30.14	29.99	10	9	3
11/5/2014	61	57	53	55	51	48	90	79	64	30.28	30.15	29.96	10	10	3
11/6/2014	64	60	54	55	50	45	90	75	58	30.3	29.99	29.8	10	10	4

Starting Hadoop

Navigate to Hadoop library

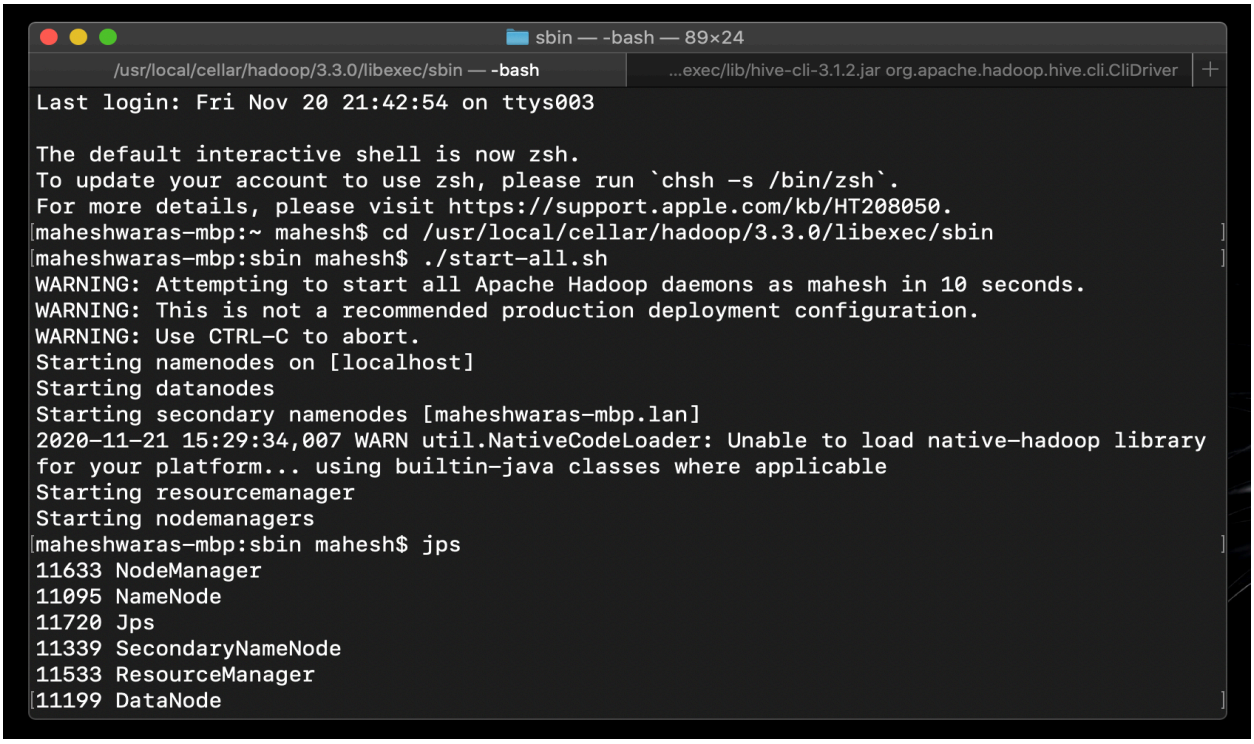
```
cd /usr/local/cellar/hadoop/3.3.0/libexec/sbin
```

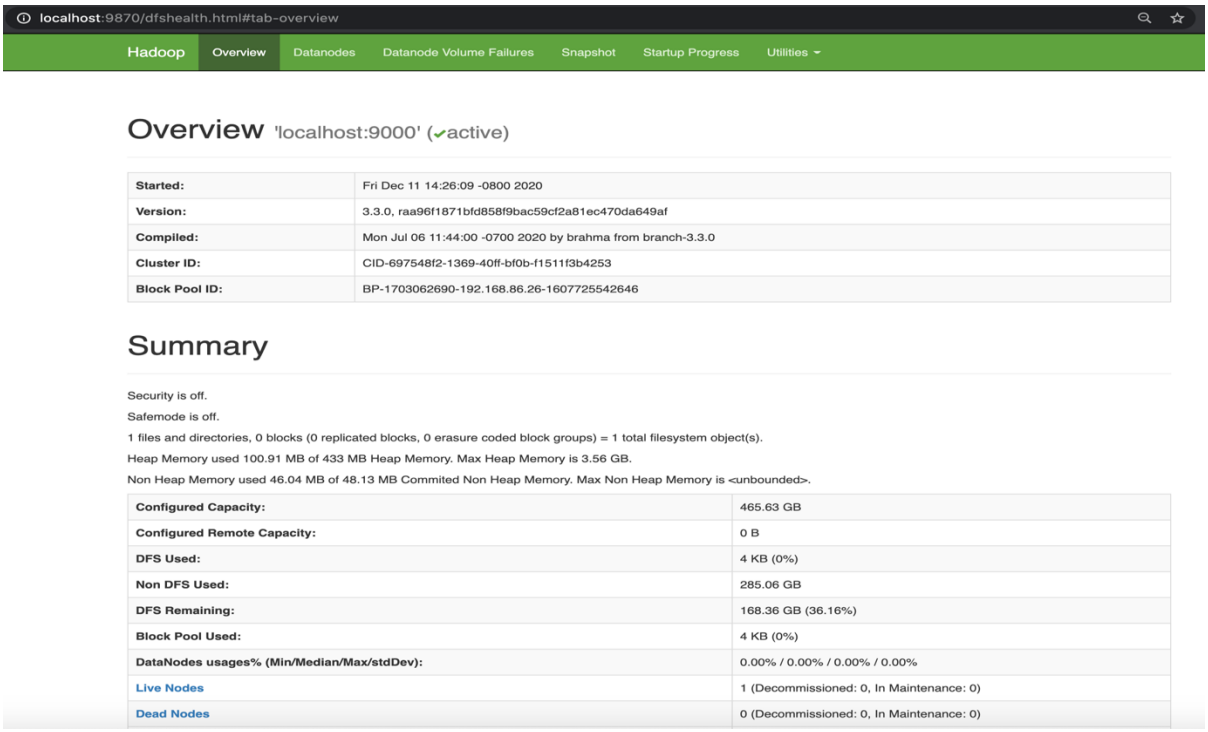
Start Hadoop Daemons

```
./start-all.sh
```

Check Hadoop Daemons

```
jps
```

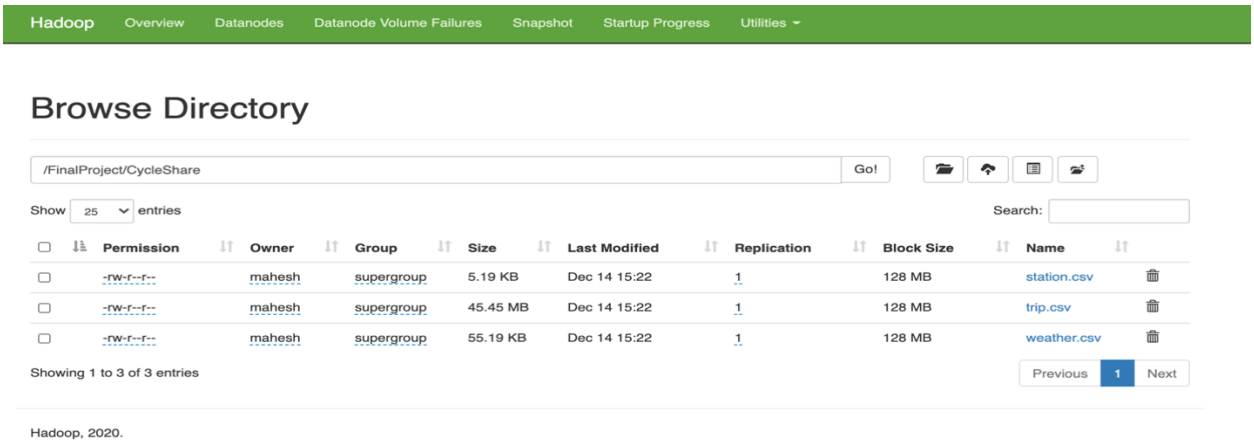




Upload the files from local file system to HDFS

```
hadoop fs –mkdir /FinalProject
hadoop fs –copyFromLocal /Users/mahesh/Desktop/CycleShare /FinalProject
hadoop fs –ls /FinalProject/CycleShare
```

```
maheshwaras-mbp:sbin mahesh$ hadoop fs -mkdir /FinalProject
2020-12-14 15:21:59,370 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
maheshwaras-mbp:sbin mahesh$ hadoop fs -copyFromLocal /Users/mahesh/Desktop/CycleShare /FinalProject
2020-12-14 15:22:20,623 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
maheshwaras-mbp:sbin mahesh$ hadoop fs -ls /FinalProject/CycleShare
2020-12-14 15:22:58,959 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 3 items
-rw-r--r--   1 mahesh supergroup       5316 2020-12-14 15:22 /FinalProject/CycleShare/station.csv
-rw-r--r--   1 mahesh supergroup   47662344 2020-12-14 15:22 /FinalProject/CycleShare/trip.csv
-rw-r--r--   1 mahesh supergroup      56516 2020-12-14 15:22 /FinalProject/CycleShare/weather.csv
```



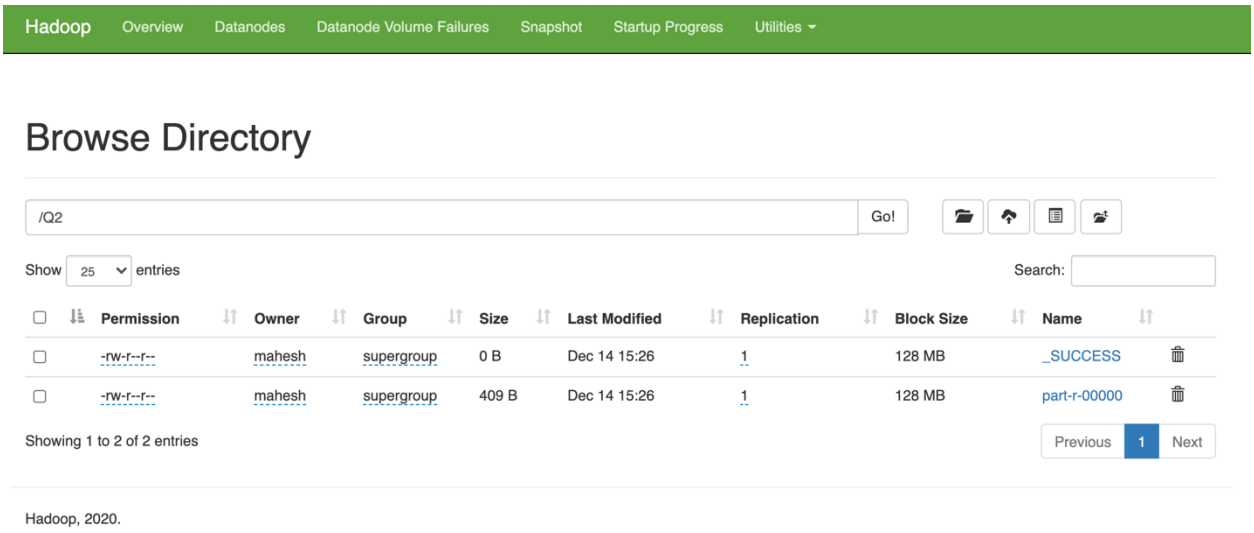
MapReduce Analysis

Analysis-1: Number of trips by month-year

Used MapReduce to find the total number of trips grouped on month-year.

```
hadoop jar /Users/mahesh/NEU_COURSES/Fall-2020/Bigdata/Project/JavaProgram/out/artifacts/JavaProgram_jar/JavaProgram.jar Q2NumberOfTripsByMonth.DriverClass /FinalProject/CycleShare/trip.csv /Q2
```

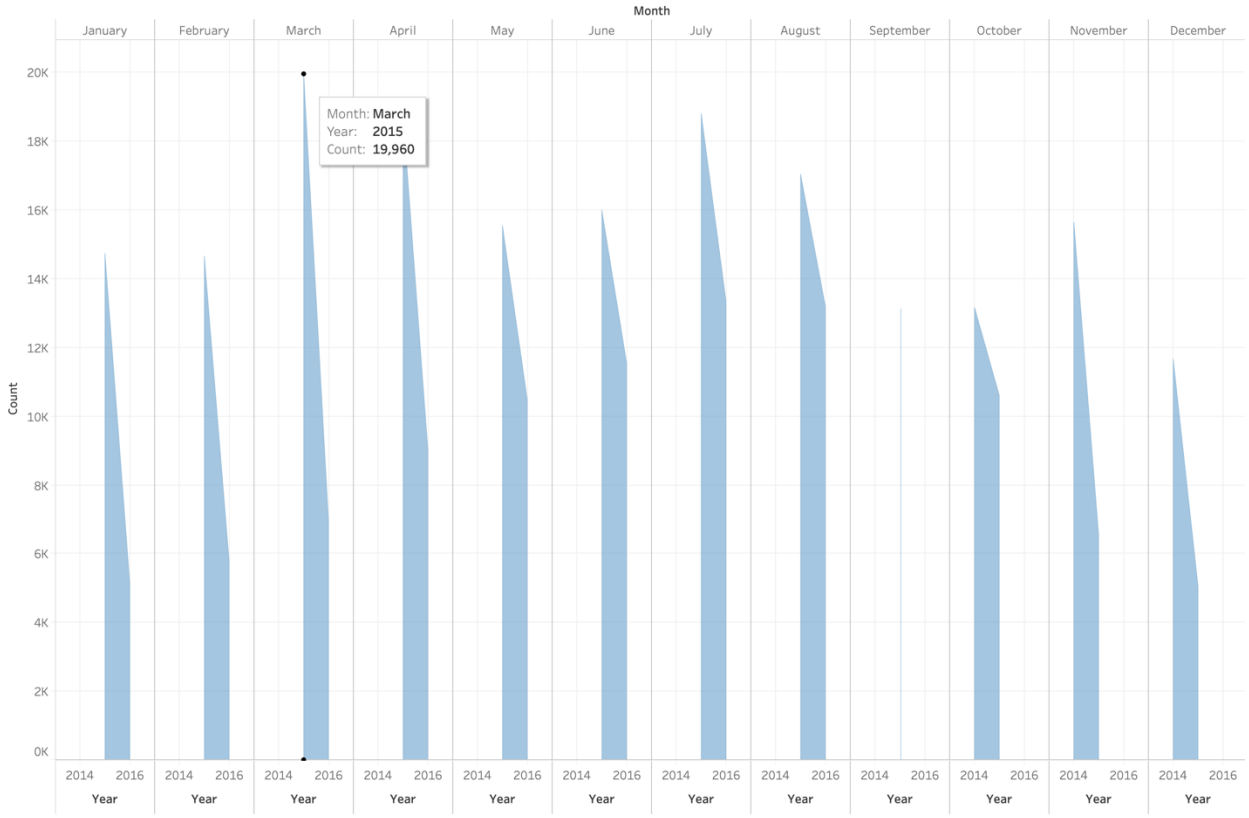
```
maheshwaras-mbp:sbin mahesh$ hadoop fs -cat /Q2/part-r-00000
2020-12-14 15:27:24,923 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
April-2015      18774
April-2016      9029
August-2015     17046
August-2016     13193
December-2014   11662
December-2015   5049
February-2015   14660
February-2016   5786
January-2015    14736
January-2016    5162
July-2015       18808
July-2016       13342
June-2015       15999
June-2016       11548
March-2015      19960
March-2016      6973
May-2015        15548
May-2016        10487
November-2014   15646
November-2015   6541
October-2014    13170
October-2015    10605
September-2015  13134
```



Analysis of the results

1. The bike usage shows a seasonal pattern. Summer months have higher ridership when compared with winter months.
2. Ridership peaked during the summer of 2015.

~~~~~~~~Representation of results in Tableau

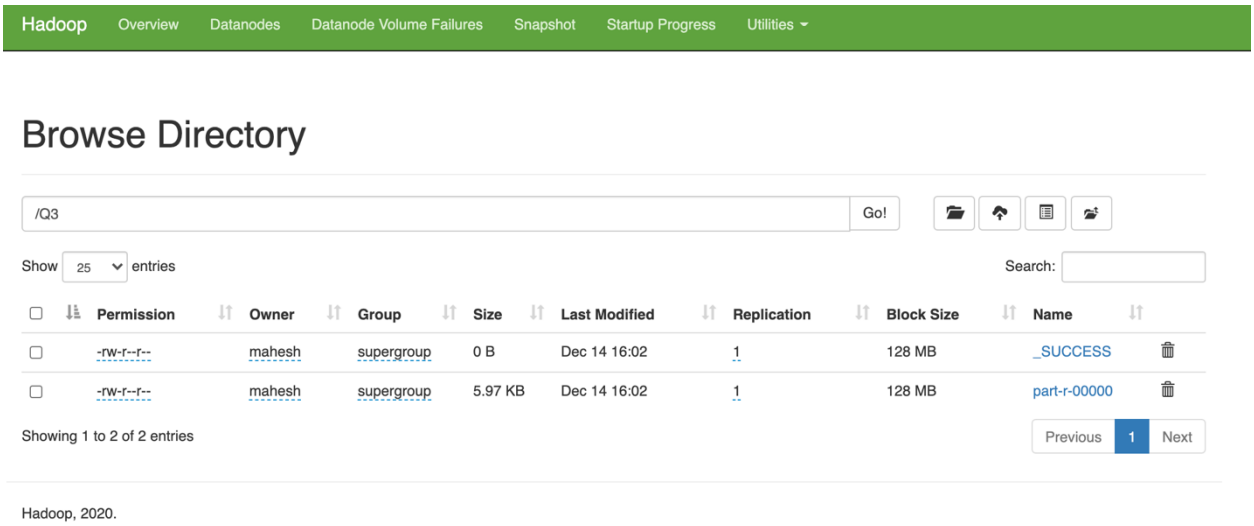

~~~~~~~~

Analysis-2: Min, Max and Average duration of trips from each station

Performed **Numerical Summarization** pattern to determine the minimum, maximum and average trip duration (in secs) from each station.
A custom **writable** object "TripWritable" is implemented to get the max, min, avgtrip and count values.

```
hadoop jar /Users/mahesh/NEU_COURSES/Fall-2020/Bigdata/Project/JavaProgram/out/artifacts/JavaProgram_jar/JavaProgram.jar Q3MinMaxAvgTripDuration.DriverClass /FinalProject/CycleShare/trip.csv /Q3
```

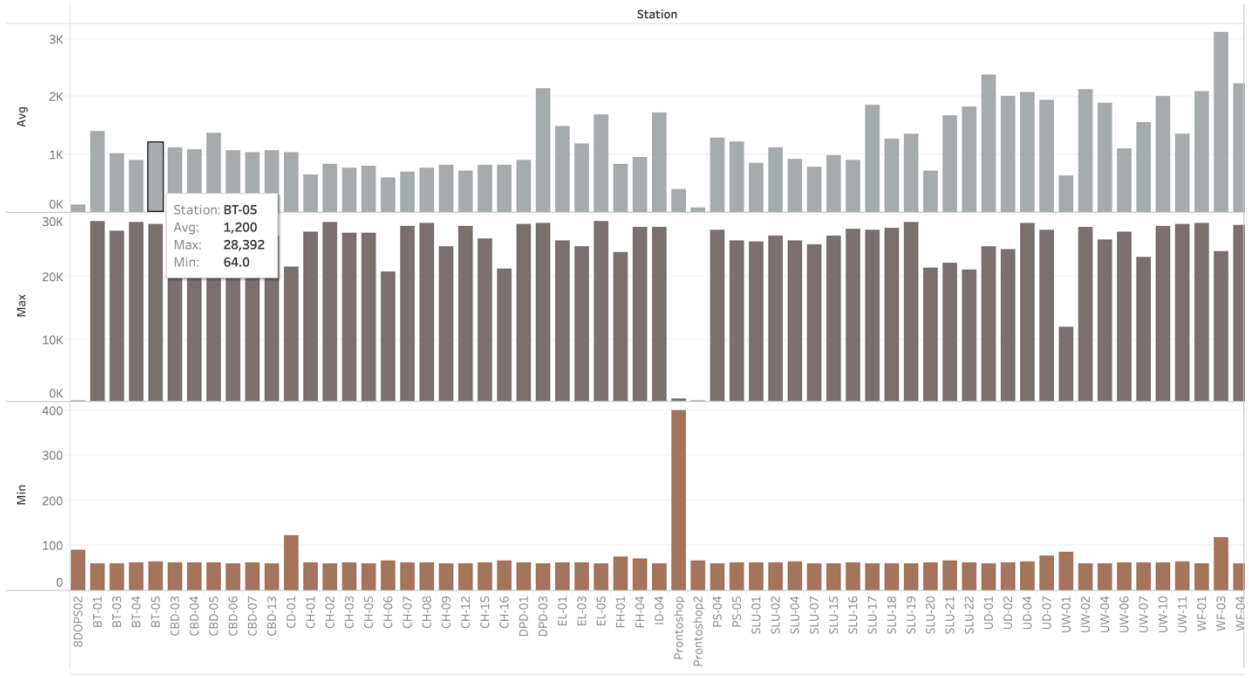
```
maheshwaras-mbp:sbin mahesh$ hadoop fs -head /Q3/part-r-00000
2020-12-14 16:03:56,954 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
8D OPS 02	Min-Trip-Duration: 90.845, Max-Trip-Duration: 176.39, Avg-Trip-Duration: 133.6175
BT-01	Min-Trip-Duration: 60.951, Max-Trip-Duration: 28779.864, Avg-Trip-Duration: 1396.3202623010818
BT-03	Min-Trip-Duration: 60.929, Max-Trip-Duration: 27293.135, Avg-Trip-Duration: 1015.4074971841358
BT-04	Min-Trip-Duration: 62.443, Max-Trip-Duration: 28658.706, Avg-Trip-Duration: 902.3080441929953
BT-05	Min-Trip-Duration: 63.995, Max-Trip-Duration: 28391.794, Avg-Trip-Duration: 1199.7352843700226
CBD-03	Min-Trip-Duration: 62.543, Max-Trip-Duration: 27959.745, Avg-Trip-Duration: 1121.8600190819066
CBD-04	Min-Trip-Duration: 61.406, Max-Trip-Duration: 28501.483, Avg-Trip-Duration: 1082.2675644262913
CBD-05	Min-Trip-Duration: 62.548, Max-Trip-Duration: 28312.515, Avg-Trip-Duration: 1375.6973330963401
CBD-06	Min-Trip-Duration: 60.386, Max-Trip-Duration: 27416.521, Avg-Trip-Duration: 1076.0194491215261
CBD-07	Min-Trip-Duration: 62.299, Max-Trip-Duration: 26828.558, Avg-Trip-Duration: 1033.391424834803
```



Analysis of the results

1. The station averages do not vary by orders of magnitude, which means the stations are spaced in comparable distances with each other.
2. Some stations have lower average than other, which means they have much closely spaced bike stations compared to others.

~~~~~~~~Representation of the results in Tableau


~~~~~~~~

Analysis-3: Total number of trips from a station in each year

Performed **Secondary sorting** to find the number of trips per station by year
Composite key = year + station
Extract year from the date and sort based on the station.

Following classes have been created to perform secondary sort on the data.
1. YearStationPair
2. SecSortComparator
3. SecSortGroupComparator
4. SecSortMapper
5. KeyPartitioner
6. SecSortReducer

```
hadoop jar /Users/mahesh/NEU_COURSES/Fall-2020/Bigdata/Project/JavaProgram/out/artifacts/JavaProgram_jar/JavaProgram.jar Q4CountTripsPerStationPerYear.DriverClass /FinalProject/CycleShare/trip.csv /Q4
```

```
maheshwaras-mbp:sbin mahesh$ hadoop fs -head /Q4/part-r-00000
2020-12-14 16:18:56,484 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2014,BT-01      1040
2014,BT-03      1306
2014,BT-04      710
2014,BT-05      990
2014,CBD-03     1084
2014,CBD-05     964
2014,CBD-06     972
2014,CBD-07     746
2014,CBD-13     1492
2014,CH-01      934
2014,CH-02      1666
2014,CH-03      1012
2014,CH-05      1240
2014,CH-06      614
2014,CH-07      1452
2014,CH-08      1280
2014,CH-09      938
2014,CH-12      1080
2014,CH-15      814
2014,DPD-01     766
2014,DPD-03     292
2014,EL-01      444
2014,EL-03      726
2014,EL-05      364
2014,FH-01      868
2014,FH-04      594
2014,ID-04      536
2014,PS-04      1032
2014,PS-05      484
2014,SLU-01     1570
2014,SLU-02     872
2014,SLU-04     932
2014,SLU-07     1156
2014,SLU-15     1314
2014,SLU-16     686
2014,SLU-17     586
2014,SLU-18     838
2014,SLU-19     746
2014,UD-01      420
2014,UD-02      322
2014,UD-04      448
2014,UD-07      388
2014,UW-01      210
2014,UW-02      262
2014,UW-04      420
2014,UW-06      554
2014,UW-07      238
2014,UW-10      210
2014,WF-01      1126
2014,WF-04      740
2015,BT-01      6940
2015,BT-03      4929
```

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/Q4 Go!

Show 25 entries Search:

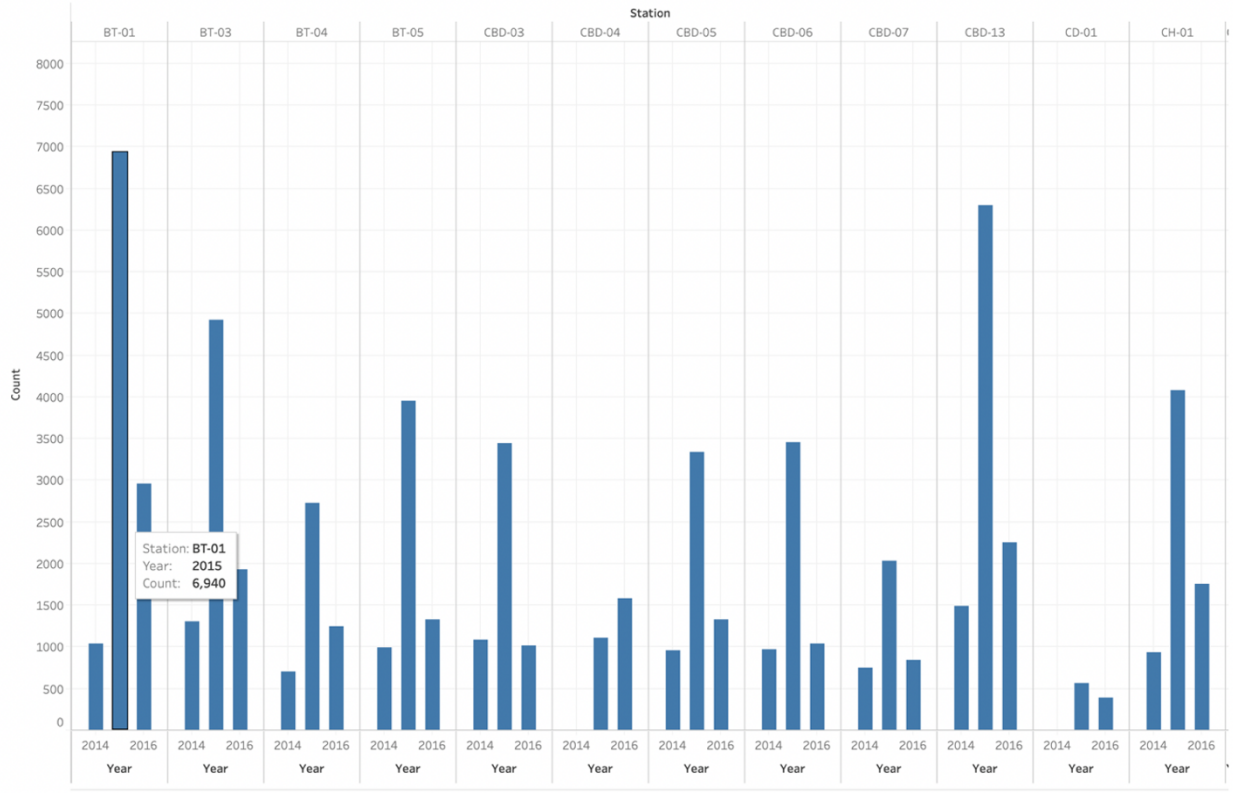
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	mahesh	supergroup	0 B	Dec 14 16:18	1	128 MB	_SUCCESS
-rw-r--r--	mahesh	supergroup	2.58 KB	Dec 14 16:18	1	128 MB	part-r-00000

Showing 1 to 2 of 2 entries Previous 1 Next

Hadoop, 2020.

~~~~~~~~Representation of the results in Tableau

(part of the results)


~~~~~~~~

Analysis-4: To find out the top 5 busy stations by month

Performed **Secondary sorting** to find the total number of trips from each station based on the month and **Chaining** MapReduce to determine the top 5 stations from output of the first MapReduce job.

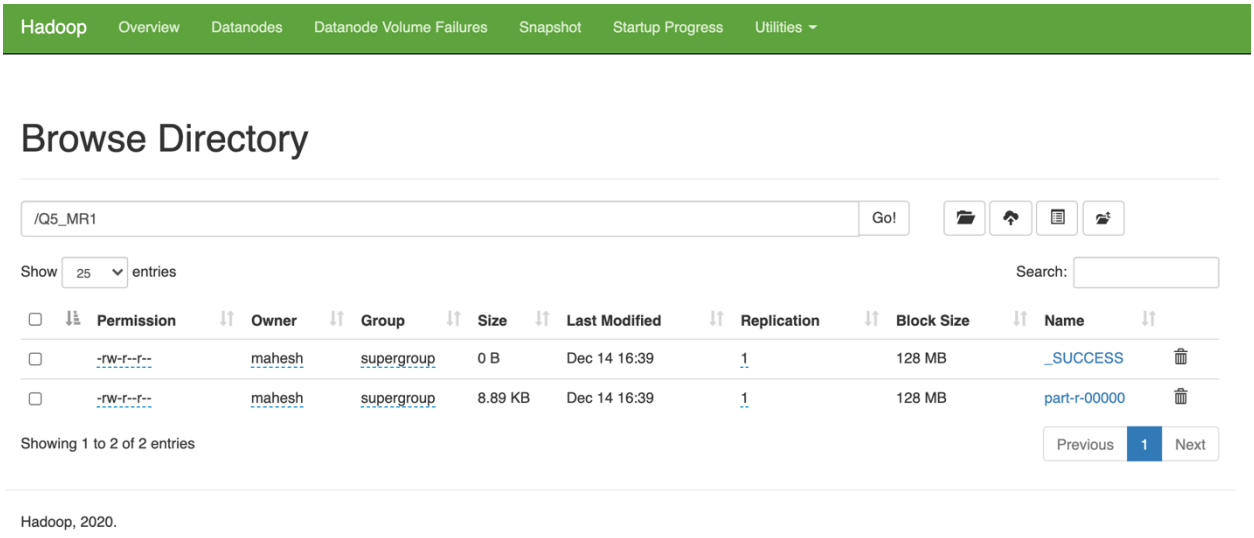
MapReduce job 01: Secondary sorting to perform partition of the file based on the month and station
Composite key = month + station
Extract month from the date and sort based on the station
Classes implemented to perform secondary sorting
1. MonthStationPair
2. SecSortComparator
3. SecSortGroupComparator
4. MapperClass
5. KeyPartitioner
6. ReducerClass

MapReduce job 02: Chaining MapReduce job to find the top 5 incoming stations by month
A custom **writable** object "StationCountWrittable" is implemented to get the station and count values.
Classes implemented
1. StationCountWrittable
2. Top5Mapper
3. Top5Reducer

```
hadoop jar /Users/mahesh/NEU_COURSES/Fall-
2020/Bigdata/Project/JavaProgram/out/artifacts/JavaProgram_jar/JavaProgram.ja
r Q5Top5BusiestStations.DriverClass /FinalProject/CycleShare/trip.csv /Q5_MR1
/Q5_MR2
```

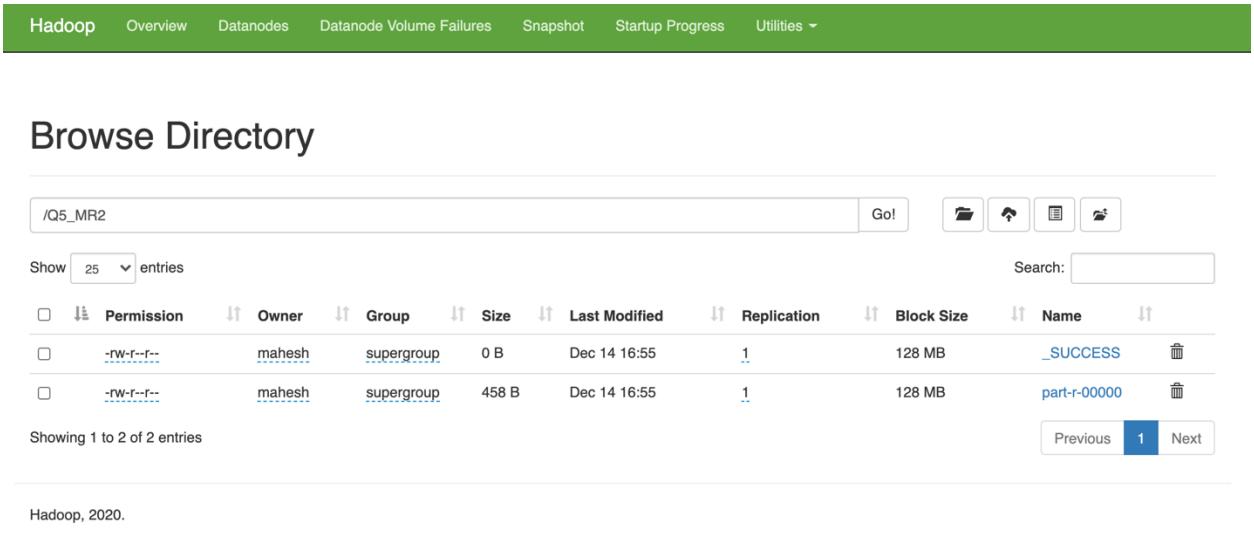
MapReduce job-1

```
maheshwaras-mbp:sbin mahesh$ hadoop fs -head /Q5_MR1/part-r-00000
2020-12-14 16:43:10,245 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
1, BT-01        601
1, BT-03        665
1, BT-04        359
1, BT-05        518
1, CBD-03       545
1, CBD-04       114
1, CBD-05       396
1, CBD-06       402
1, CBD-07       235
1, CBD-13       653
1, CD-01        35
1, CH-01        561
1, CH-02        752
1, CH-03        489
1, CH-05        562
1, CH-06        241
1, CH-07        994
1, CH-08        749
1, CH-09        376
1, CH-12        490
1, CH-15        417
1, DPD-01       426
1, DPD-03       154
1, EL-01        208
1, EL-03        473
1, EL-05        181
1, FH-01        292
1, FH-04        295
1, ID-04        160
1, PS-04        317
1, PS-05        239
1, SLU-01       829
1, SLU-02       428
1, SLU-04       479
1, SLU-07       656
1, SLU-15       753
1, SLU-16       475
1, SLU-17       265
1, SLU-18       332
1, SLU-19       443
1, SLU-20       64
1, SLU-21       24
1, UD-01        202
1, UD-02        95
1, UD-04        185
1, UD-07        153
1, UW-01        66
1, UW-02        114
1, UW-04        174
1, UW-06        213
1, UW-07        96
1, UW-10        105
```



MapReduce job-2

```
maheshwaras-mbp:sbin mahesh$ hadoop fs -head /Q5_MR2/part-r-00000
2020-12-14 16:56:35,645 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
1       CH-07, SLU-01, SLU-15, CH-02, CH-08
2       CH-07, WF-01, SLU-15, SLU-01, CH-02
3       WF-01, CH-07, SLU-15, CH-08, BT-01
4       WF-01, BT-01, CH-07, CBD-13, SLU-15
5       WF-01, BT-01, SLU-15, CBD-13, CH-07
6       WF-01, BT-01, CBD-13, CH-07, SLU-15
7       WF-01, BT-01, CH-07, CBD-13, SLU-15
8       WF-01, BT-01, CH-07, SLU-15, CBD-13
9       WF-01, BT-01, CH-07, CH-08, CBD-13
10      CH-02, CH-07, CH-08, SLU-15, CBD-13
11      CH-07, CH-02, CBD-13, SLU-01, SLU-15
12      CH-07, SLU-01, CH-02, CBD-13, BT-01
```

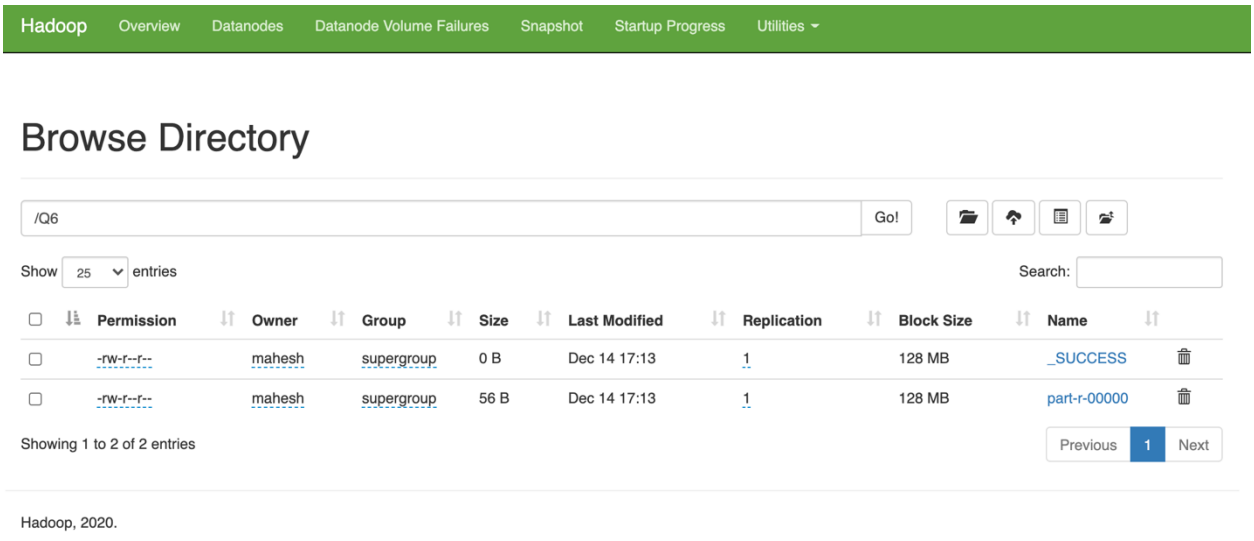


Analysis-5: Most active age groups

Performed a MapReduce job to determine which age group is most active in using the bike service.
A **composite** key “MemberandAge” is implemented to get the age band.

```
hadoop jar /Users/mahesh/NEU_COURSES/Fall-2020/Bigdata/Project/JavaProgram/out/artifacts/JavaProgram_jar/JavaProgram.jar Q6MostActiveAgeGroup.DriverClass /FinalProject/CycleShare/trip.csv /Q6
```

```
maheshwaras-mbp:sbin mahesh$ hadoop fs -head /Q6/part-r-00000
2020-12-14 17:14:45,320 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[30-40) 137030
[40-50) 55526
[50+)   43922
[LT-30) 126628
```



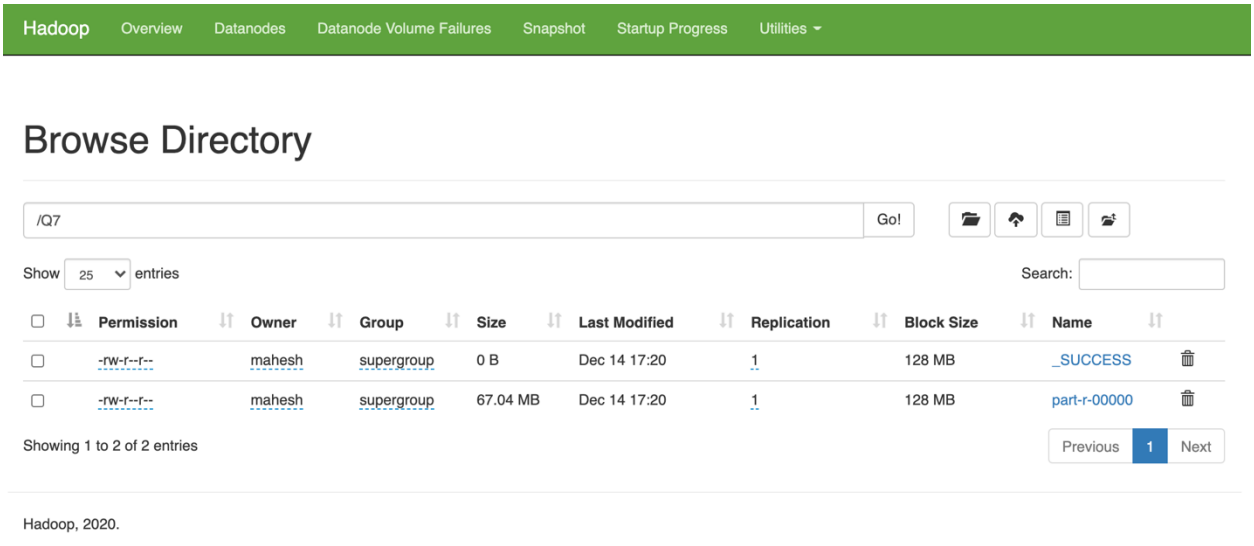
Analysis of the results
Age groups between 30 and 40 years are the most active among all the members.

Analysis-6: Number of trips in a day from each station and the corresponding weather on that day

Performed the MapReduce - **Reduce side join** - inner join operation on trips.csv table and weather.csv table to determine the weather corresponding to each day.

Classes implemented
1. Trip Mapper
2. Weather Mapper
3. Reducer

```
hadoop jar /Users/mahesh/NEU_COURSES/Fall-2020/Bigdata/Project/JavaProgram/out/artifacts/JavaProgram_jar/JavaProgram.jar Q7JoinPattern.DriverClass /FinalProject/CycleShare/trip.csv /FinalProject/CycleShare/weather.csv inner /Q7
```



Analysis-7: Custom MapReduce algorithm to find the top 10 most busy routes

Performed **secondary sorting** to find the total number of trips between two stations. Used **chaining** MapReduce to determine the top 10 station-pairs based on traffic from output of the first MapReduce job.

MapReduce job 01: Secondary sorting to perform partition of the file based on from and to station
Composite key = station-1 + station-2
(where station-1 and station-2 are from and to stations such that station-1 is sorted before station-2. This allows the Mapper to group a trip from Station-B to Station-A together with a trip from Station-A to Station-B)
Classes implemented to perform secondary sorting
1. FromToStationPair
2. SecSortComparator
3. SecSortGroupComparator
4. SecSortMapper
5. KeyPartitioner
6. SecSortReducer

MapReduce job 02: To find the top 10 busy routes
A custom algorithm is implemented using the priority queue to find the top 10 busy routes
Classes implemented
1. Top10Mapper
2. Top10Reducer

```
hadoop jar /Users/mahesh/NEU_COURSES/Fall-
2020/Bigdata/Project/JavaProgram/out/artifacts/JavaProgram_jar/JavaProgram.ja
r Q8Top5BusyRoutes.DriverClass /FinalProject/CycleShare/trip.csv /Q8_MR1
/Q8_MR2
```

```
maheshwaras-mbp:sbin mahesh$ hadoop fs -head /Q8_MR1/part-r-00000
2020-12-14 20:57:01,801 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
8D OPS 02, 8D OPS 02    2
BT-01, BT-01    738
BT-01, BT-03    562
BT-01, BT-04    337
BT-01, BT-05    917
BT-01, CBD-0    1602
BT-01, CBD-1    1247
BT-01, CD-01    21
BT-01, CH-01    401
BT-01, CH-02    125
BT-01, CH-03    144
BT-01, CH-05    67
BT-01, CH-06    30
BT-01, CH-07    227
BT-01, CH-08    154
BT-01, CH-09    175
BT-01, CH-12    326
BT-01, CH-15    61
BT-01, CH-16    26
BT-01, DPD-0    215
BT-01, EL-01    561
BT-01, EL-03    237
BT-01, EL-05    58
BT-01, FH-01    67
BT-01, FH-04    48
BT-01, ID-04    200
BT-01, PS-04    518
BT-01, PS-05    541
BT-01, SLU-0    1322
BT-01, SLU-1    1335
BT-01, SLU-2    137
BT-01, UD-01    16
BT-01, UD-02    19
BT-01, UD-04    41
BT-01, UD-07    15
BT-01, UW-01    4
BT-01, UW-02    14
BT-01, UW-04    46
BT-01, UW-06    13
BT-01, UW-07    4
BT-01, UW-10    4
BT-01, UW-11    3
BT-01, WF-01    1291
BT-01, WF-03    15
BT-01, WF-04    1443
BT-01", CBD-03  266
BT-01", CBD-04  152
BT-01", CBD-05  240
BT-01", CBD-06  587
BT-01", CBD-07  249
BT-01", CBD-13  1111
BT-01", DPD-01  164
```

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/Q8_MR1 Go!

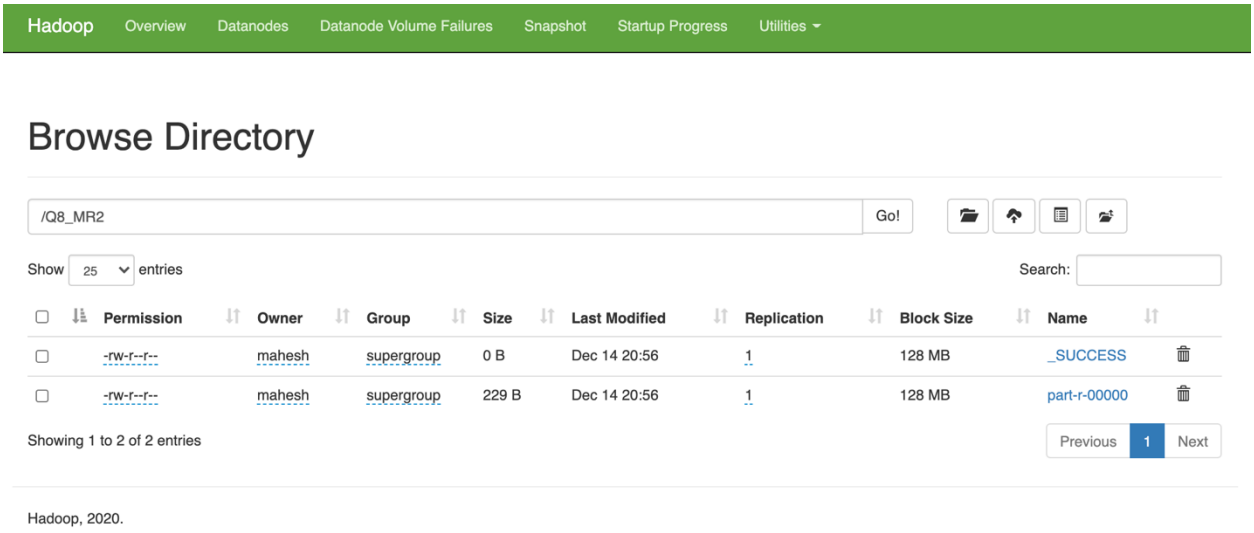
Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	mahesh	supergroup	0 B	Dec 14 20:55	1	128 MB	_SUCCESS
-rw-r--r--	mahesh	supergroup	29.52 KB	Dec 14 20:55	1	128 MB	part-r-00000

Showing 1 to 2 of 2 entries Previous 1 Next

Hadoop, 2020.

```
BT-0maheshwaras-mbp:sbin mahesh$ hadoop fs -head /Q8_MR2/part-r-00000
2020-12-14 20:57:09,261 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Top 10 Routes   WF-01 <-> WF-01:4881
WF-01 <-> WF-04:4613
PS-05 <-> WF-01:1866
SLU-01 <-> SLU-16:1696
BT-01 <-> CBD-0:1602
CH-02 <-> CH-08:1568
SLU-02 <-> SLU-07:1522
CH-02 <-> CH-09:1521
CH-07 <-> SLU-1:1521
EL-03 <-> SLU-1:1515
```



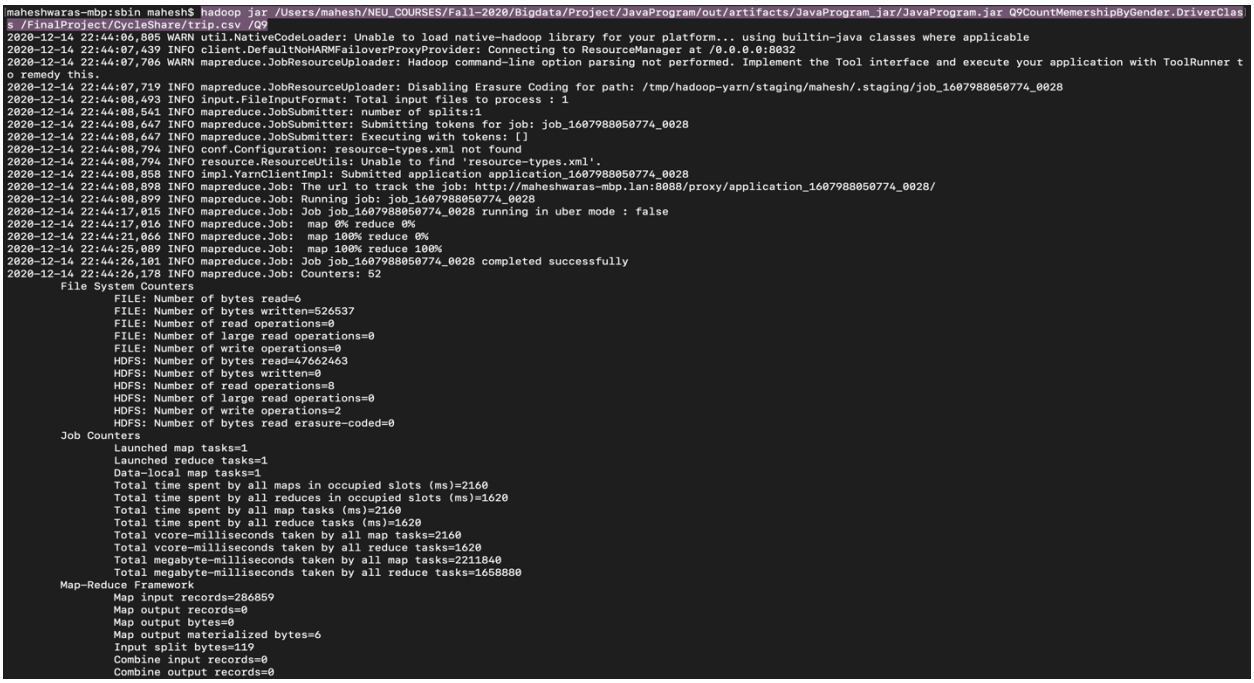
Analysis of the results:

Busy routes help to decide where the excess bikes are going to be.

Analysis-8: Count membership by gender

Performed **Counting with counters** algorithm to find the number of male and female who hold the membership.
Output is not stored in the directory, it is displayed in the console.

```
hadoop jar /Users/mahesh/NEU_COURSES/Fall-2020/Bigdata/Project/JavaProgram/out/artifacts/JavaProgram_jar/JavaProgram.jar Q9CountMemershipByGender.DriverClass /FinalProject/CycleShare/trip.csv /Q9
```



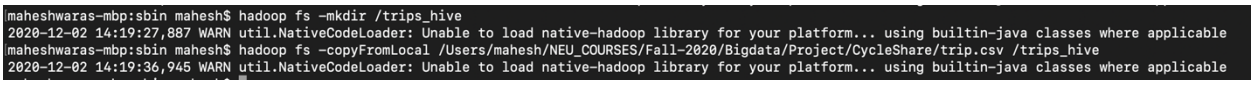
```
maheshwaras-mbp:sbin mahesh$ hadoop jar /Users/mahesh/NEU_COURSES/Fall-2020/Bigdata/Project/JavaProgram/out/artifacts/JavaProgram_jar/JavaProgram.jar Q9CountMemershipByGender.DriverClass /FinalProject/CycleShare/trip.csv /Q9
2020-12-14 22:44:06,805 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2020-12-14 22:44:07,439 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2020-12-14 22:44:07,706 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2020-12-14 22:44:07,719 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/mahesh/.staging/job_1607988050774_0028
2020-12-14 22:44:08,493 INFO input.FileInputFormat: Total input files to process : 1
2020-12-14 22:44:08,541 INFO mapreduce.JobSubmitter: number of splits:1
2020-12-14 22:44:08,647 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1607988050774_0028
2020-12-14 22:44:08,647 INFO mapreduce.JobSubmitter: Executing with tokens: []
2020-12-14 22:44:08,794 INFO conf.Configuration: resource-types.xml not found
2020-12-14 22:44:08,794 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2020-12-14 22:44:08,858 INFO impl.YarnClientImpl: Submitted application application_1607988050774_0028
2020-12-14 22:44:08,898 INFO mapreduce.Job: The url to track the job: http://maheshwaras-mbp.lan:8088/proxy/application_1607988050774_0028/
2020-12-14 22:44:08,899 INFO mapreduce.Job: Running job: job_1607988050774_0028
2020-12-14 22:44:17,015 INFO mapreduce.Job: Job job_1607988050774_0028 running in uber mode : false
2020-12-14 22:44:17,016 INFO mapreduce.Job:  map 0% reduce 0%
2020-12-14 22:44:21,066 INFO mapreduce.Job:  map 100% reduce 0%
2020-12-14 22:44:25,089 INFO mapreduce.Job:  map 100% reduce 100%
2020-12-14 22:44:26,101 INFO mapreduce.Job: Job job_1607988050774_0028 completed successfully
2020-12-14 22:44:26,178 INFO mapreduce.Job: Counters: 52
        File System Counters
                FILE: Number of bytes read=6
                FILE: Number of bytes written=526537
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=47662463
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=2160
                Total time spent by all reduces in occupied slots (ms)=1620
                Total time spent by all map tasks (ms)=2160
                Total time spent by all reduce tasks (ms)=1620
                Total vcore-milliseconds taken by all map tasks=2160
                Total vcore-milliseconds taken by all reduce tasks=1620
                Total megabyte-milliseconds taken by all map tasks=2211840
                Total megabyte-milliseconds taken by all reduce tasks=1658880
        Map-Reduce Framework
                Map input records=286859
                Map output records=0
                Map output bytes=0
                Map output materialized bytes=6
                Input split bytes=119
                Combine input records=0
                Combine output records=0
```

```
                Reduce input groups=0
                Reduce shuffle bytes=6
                Reduce input records=0
                Reduce output records=0
                Spilled Records=0
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=74
                CPU time spent (ms)=0
                Physical memory (bytes) snapshot=0
                Virtual memory (bytes) snapshot=0
                Total committed heap usage (bytes)=692060160
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        gender
                Female=37562
                Male=140564
        File Input Format Counters
                Bytes Read=47662344
        File Output Format Counters
                Bytes Written=0
Female  37562
Male    140564
```

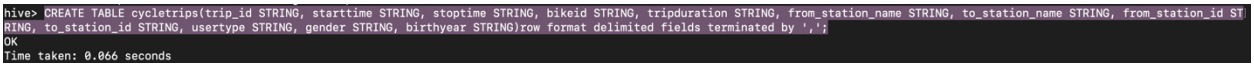
Hive Analysis

Starting the hive server
hive --service metastore &
hive

Create a directory in HDFS
```
hadoop fs –mkdir /trips_hive
hadoop fs –copyFromLocal /Users/mahesh/NEU_COURSES/Fall–2020/Bigdata/Project/CycleShare/trip.csv /trips_hive
```



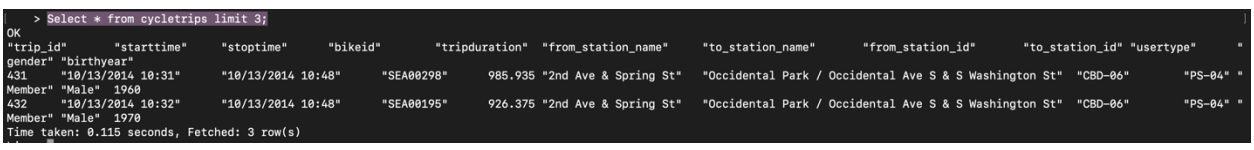
Create table 'cycletrips' in hive
```
hive> CREATE TABLE cycletrips(trip_id STRING, starttime STRING, stoptime STRING, bikeid STRING, tripduration STRING, from_station_name STRING, to_station_name STRING, from_station_id STRING, to_station_id STRING, usertype STRING, gender STRING, birthyear STRING)row format delimited fields terminated by ',';
```



Load trips.csv file into cycletrips table
```
LOAD DATA INPATH 'hdfs://localhost:9000/trips_hive' OVERWRITE INTO TABLE cycletrips;
```

```
hive> LOAD DATA INPATH 'hdfs://localhost:9000/trips_hive' OVERWRITE INTO TABLE cycletrips;
Loading data to table default.cycletrips
OK
Time taken: 0.211 seconds
hive> DESCRIBE cycletrips;
OK
trip_id                 string
starttime               string
stoptime                string
bikeid                  string
tripduration            string
from_station_name       string
to_station_name         string
from_station_id         string
to_station_id           string
usertype                string
gender                  string
birthyear               string
Time taken: 0.068 seconds, Fetched: 12 row(s)
```

```
Select * from cycletrips limit 3;
```



Analysis-9: Count of all the trips by station

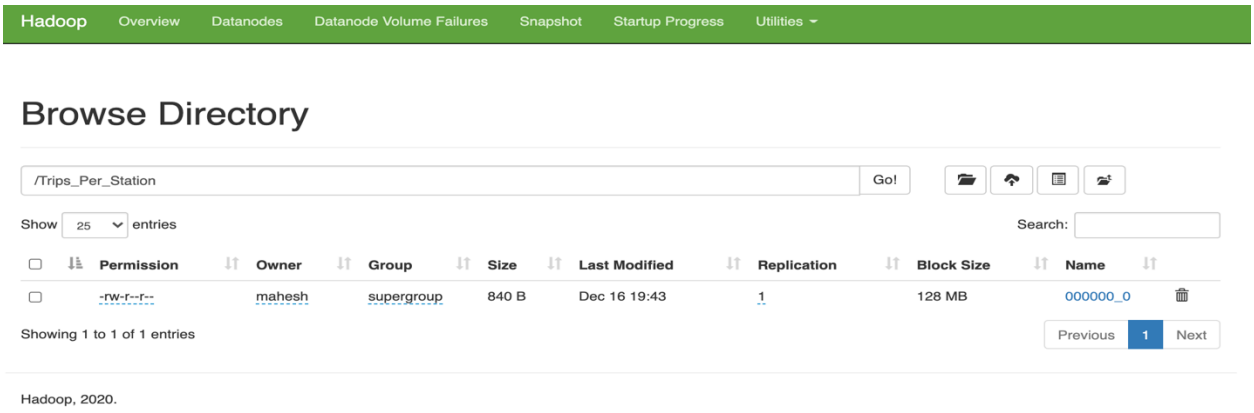
```
Select from_station_id, count(*) from cycletrips WHERE from_station_id is not NULL AND from_station_id <> '' AND length(from_station_id) > 0 GROUP BY from_station_id;
```

```
hive> Select from_station_id, count(*) from cycletrips WHERE from_station_id is not NULL AND from_station_id <> '' AND length(from_station_id) > 0 GROUP BY from_station_id;
Query ID = mahesh_20201202144042_531430c4-80a7-42dc-88f8-a86c72271052
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1606887595230_0004, Tracking URL = http://maheshwaras-mbp.lan:8088/proxy/application_1606887595230_0004/
Kill Command = /usr/local/Cellar/hadoop/3.3.0/libexec/bin/mapred job  -kill job_1606887595230_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-12-02 14:40:49,525 Stage-1 map = 0%,  reduce = 0%
2020-12-02 14:40:54,639 Stage-1 map = 100%,  reduce = 0%
2020-12-02 14:40:59,753 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_1606887595230_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   HDFS Read: 47678821 HDFS Write: 1671 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
"8D OPS 02"     2
"BT-01" 10934
"BT-03" 8168
"BT-04" 4684
"BT-05" 6270
"CBD-03"        5555
"CBD-04"        2693
"CBD-05"        5626
"CBD-06"        5464
"CBD-07"        3632
"CBD-13"        10049
"CD-01" 958
"CH-01" 6768
"CH-02" 9639
"CH-03" 6576
"CH-05" 7680
"CH-06" 3837
"CH-07" 11392
"CH-08" 9468
"CH-09" 5698
"CH-12" 5548
"CH-15" 6322
"CH-16" 1288
"DPD-01"        5161
"DPD-03"        1657
"EL-01" 3580
"EL-03" 5747
"EL-05" 3645
"FH-01" 3282
"FH-04" 4077
"ID-04" 2732
```

Load the output into HDFS

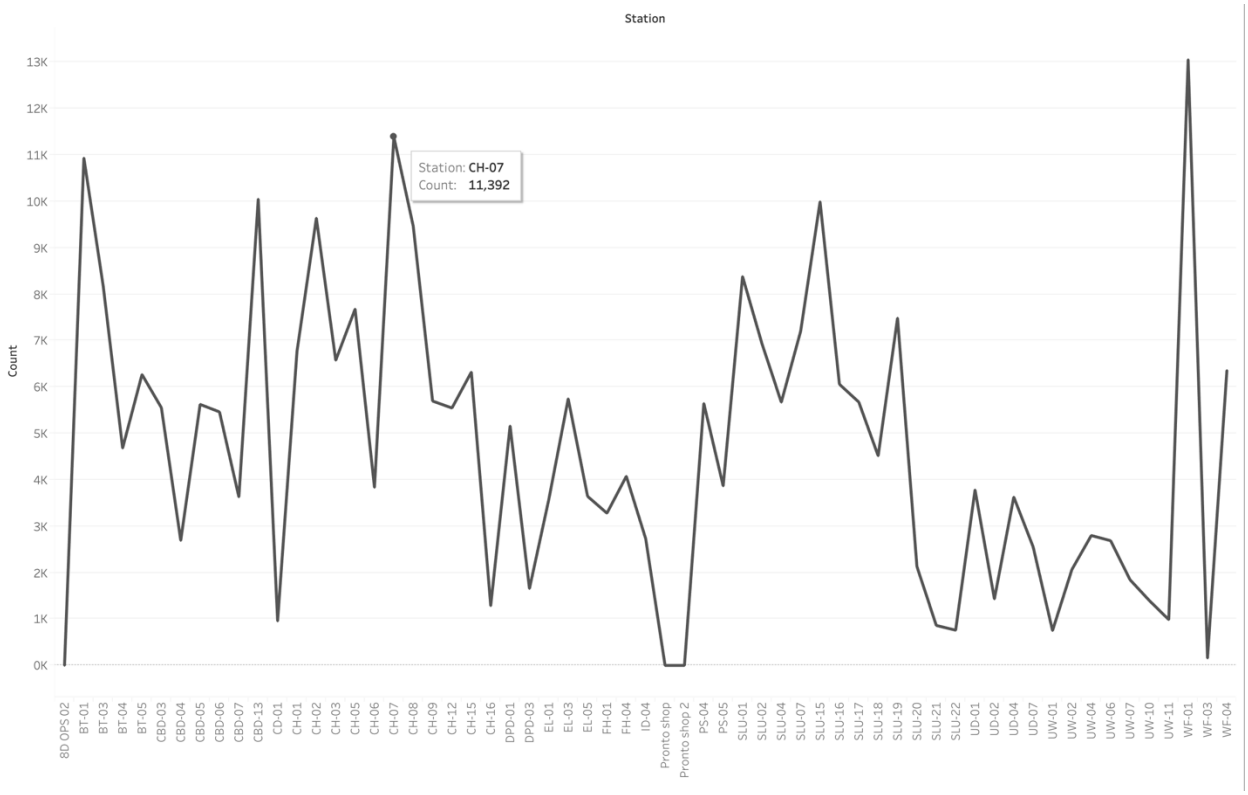
```
Insert Overwrite Directory 'hdfs://localhost:9000/Trips_Per_Station' Select from_station_id, count(*) from cycletrips WHERE from_station_id is not NULL AND from_station_id <> '' AND length(from_station_id) > 0 GROUP BY from_station_id;
```

```
hive> Insert Overwrite Directory 'hdfs://localhost:9000/Trips_Per_Station' Select from_station_id, count(*) from cycletrips WHERE from_station_id is not NULL AND from_station_id <> '' A
ND length(from_station_id) > 0 GROUP BY from_station_id;
Query ID = mahesh_20201202144537_13ec4b7c-4615-403e-bff4-78992af6bddd
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1606887595230_0005, Tracking URL = http://maheshwaras-mbp.lan:8088/proxy/application_1606887595230_0005/
Kill Command = /usr/local/Cellar/hadoop/3.3.0/libexec/bin/mapred job  -kill job_1606887595230_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-12-02 14:45:42,949 Stage-1 map = 0%,  reduce = 0%
2020-12-02 14:45:48,072 Stage-1 map = 100%,  reduce = 0%
2020-12-02 14:45:53,180 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_1606887595230_0005
Moving data to directory hdfs://localhost:9000/Trips_Per_Station
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   HDFS Read: 47678368 HDFS Write: 840 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 17.15 seconds
```




```
maheshwaras-mbp:sbin mahesh$ hadoop fs -cat /Trips_Per_Station/000000_0
2020-12-02 14:47:17,357 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
"8D OPS 02"2
"BT-01"10934
"BT-03"8168
"BT-04"4684
"BT-05"6270
"CBD-03"5555
"CBD-04"2693
"CBD-05"5626
"CBD-06"5464
"CBD-07"3632
"CBD-13"10049
"CD-01"958
"CH-01"6768
"CH-02"9639
"CH-03"6576
"CH-05"7680
"CH-06"3837
"CH-07"11392
"CH-08"9468
"CH-09"5698
"CH-12"5548
"CH-15"6322
"CH-16"1288
"DPD-01"5161
"DPD-03"1657
"EL-01"3580
"EL-03"5747
"EL-05"3645
"FH-01"3282
"FH-04"4077
"ID-04"2732
"PS-04"5647
"PS-05"3868
"Pronto shop 2"2
"Pronto shop"1
"SLU-01"8382
"SLU-02"6937
"SLU-04"5673
"SLU-07"7206
"SLU-15"9994
"SLU-16"6060
"SLU-17"5676
"SLU-18"4518
"SLU-19"7486
"SLU-20"2134
"SLU-21"863
"SLU-22"761
"UD-01"3784
"UD-02"1434
"UD-04"3630
"UD-07"2555
```

~~~~~~~~Representation of results in Tableau



Analysis-10: Top 5 busiest hours of the day

Create table test to store the timestamp

```
Create Table test as Select starttime from cycletrips;
```

Create table datetime to extract time from the timestamp

```
Create table datetime as Select
LTRIM(SUBSTRING_INDEX(SUBSTRING(starttime,12,4),":",1)) as hour from test;
```

```
hive> Create table datetime as Select LTRIM(SUBSTRING_INDEX(SUBSTRING(starttime,12,4),":",1)) as hour from cycletrips;
Query ID = mahesh_20201203220337_81a85a11-1051-41fb-b625-4cc8f5e63090
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1607059866948_0001, Tracking URL = http://maheshwaras-mbp.lan:8088/proxy/application_1607059866948_0001/
Kill Command = /usr/local/Cellar/hadoop/3.3.0/libexec/bin/mapred job  -kill job_1607059866948_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2020-12-03 22:03:48,581 Stage-1 map = 0%,  reduce = 0%
2020-12-03 22:03:54,798 Stage-1 map = 100%,  reduce = 0%
Ended Job = job_1607059866948_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/.hive-staging_hive_2020-12-03_22-03-37_146_552756235035558283O-1/-ext-10002
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/datetime
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   HDFS Read: 47669173 HDFS Write: 725670 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 19.286 seconds
hive> show tables;
OK
cycletrips
datetime
Time taken: 0.042 seconds, Fetched: 2 row(s)
hive>
```

```
SELECT hour, count(*) as use from datetime group by hour order by use desc
LIMIT 5;
```

```
hive> select hour, count(*) as use from datetime group by hour order by use desc LIMIT 5;
Query ID = mahesh_20201216195938_b8de0126-59e2-4b4c-b07d-ea21e997fe34
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1608175553045_0006, Tracking URL = http://maheshwaras-mbp.lan:8088/proxy/application_1608175553045_0006/
Kill Command = /usr/local/Cellar/hadoop/3.3.0/libexec/bin/mapred job  -kill job_1608175553045_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-12-16 19:59:44,967 Stage-1 map = 0%,  reduce = 0%
2020-12-16 19:59:50,075 Stage-1 map = 100%,  reduce = 0%
2020-12-16 19:59:55,178 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_1608175553045_0006
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1608175553045_0007, Tracking URL = http://maheshwaras-mbp.lan:8088/proxy/application_1608175553045_0007/
Kill Command = /usr/local/Cellar/hadoop/3.3.0/libexec/bin/mapred job  -kill job_1608175553045_0007
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-12-16 20:00:06,783 Stage-2 map = 0%,  reduce = 0%
2020-12-16 20:00:10,873 Stage-2 map = 100%,  reduce = 0%
2020-12-16 20:00:15,978 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_1608175553045_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   HDFS Read: 737228 HDFS Write: 659 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   HDFS Read: 8276 HDFS Write: 189 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
8       23341
17      22025
9       19565
16      18017
7       16671
```

Pig Analysis

Run Pig in local mode
pig -x local

Analysis-11: Total number of trips that lasted more than 30mins (1800) in each station

1. Define the CSVLoader and upload the trips dataset from HDFS using CSVLoader.

```
DEFINE CSVLoader org.apache.pig.piggybank.storage.CSVLoader();

trips = LOAD 'hdfs://localhost:9000/cycleShare_pig/CycleShare/trip.csv' USING
CSVLoader AS (trip_id:INT, starttime:CHARARRAY, stoptime:CHARARRAY, bikeid:
CHARARRAY, tripduration:DOUBLE, from_station_name:CHARARRAY,
to_station_name:CHARARRAY, from_station_id:CHARARRAY,
to_station_id:CHARARRAY, usertype:CHARARRAY, gender:CHARARRAY,
birthyear:INT);
```

2. Filtering the trips file

```
trips = FOREACH trips GENERATE tripduration, from_station_id;
duration = FILTER trips BY tripduration >= 1800;
```

3. Grouping the filtered trips by from_station_id

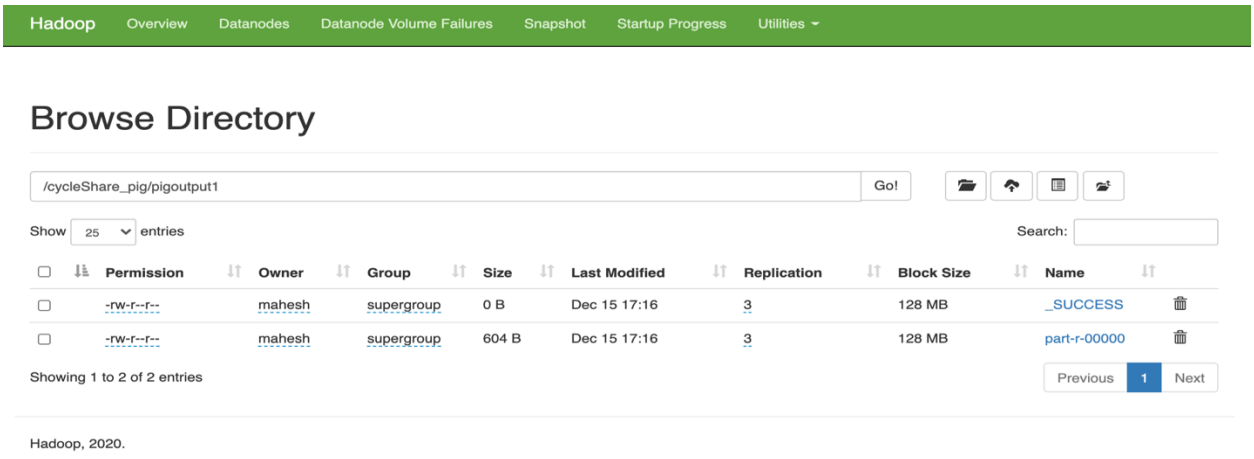
```
grouped = GROUP duration BY from_station_id;

counted = FOREACH grouped GENERATE group, COUNT(duration) as cnt;

sorted = ORDER counted BY cnt DESC;
```

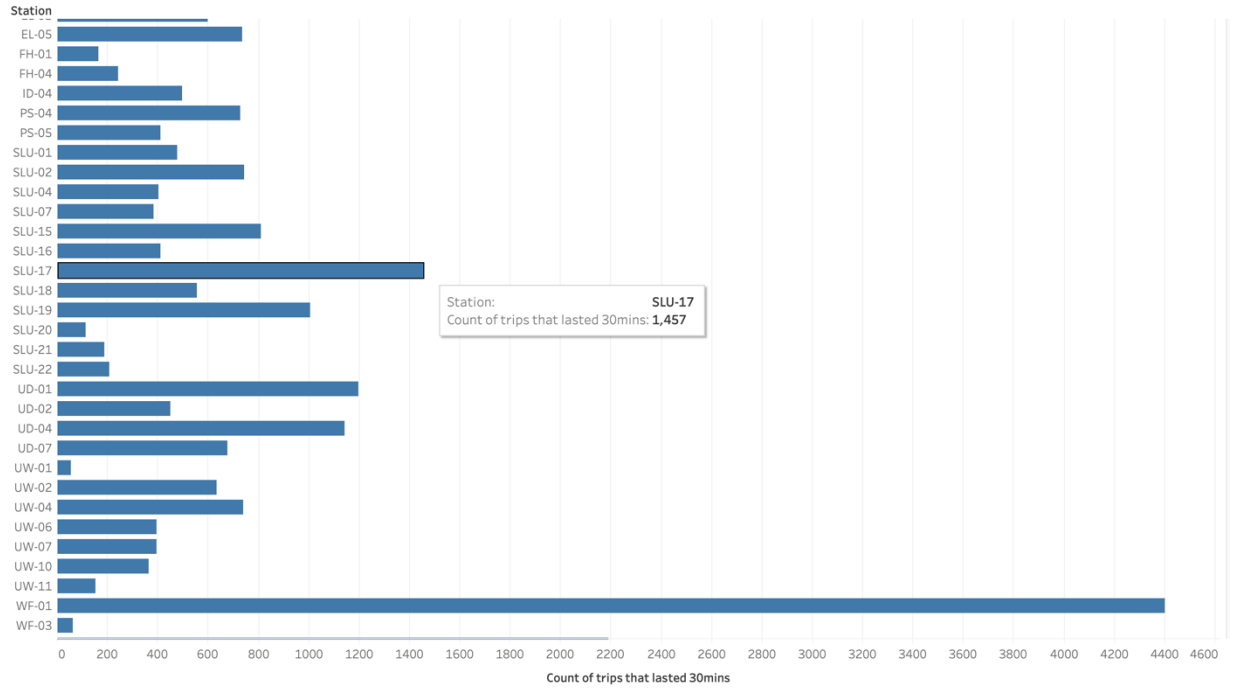
4. Storing the output file to HDFS

```
Store sorted into 'hdfs://localhost:9000/cycleShare_pig/pigoutput1' USING
PigStorage(',');
```



```
(WF-01,4400)
(WF-04,2192)
(BT-01,1811)
(SLU-17,1457)
(UD-01,1198)
(UD-04,1141)
(SLU-19,1004)
(CBD-13,865)
(SLU-15,809)
(SLU-02,744)
(UW-04,740)
(EL-05,737)
(CBD-05,731)
(PS-04,729)
(BT-03,712)
(BT-05,702)
(UD-07,677)
(EL-01,640)
(UW-02,635)
(EL-03,597)
(CBD-06,555)
(SLU-18,555)
(CH-02,542)
(CBD-03,513)
(CH-08,513)
(ID-04,497)
(SLU-01,476)
(UD-02,451)
(DPD-01,442)
(DPD-03,442)
(SLU-16,411)
(PS-05,410)
(SLU-04,403)
(UW-07,396)
(UW-06,395)
(CH-07,393)
(SLU-07,382)
(CH-05,381)
(CH-15,371)
(BT-04,367)
(UW-10,364)
(CH-09,347)
(CBD-07,301)
(CH-03,281)
(CH-12,259)
(FH-04,244)
(CBD-04,235)
(CH-01,224)
(SLU-22,208)
(SLU-21,189)
(FH-01,163)
(UW-11,154)
```

~~~~~~~~Representation of results in Tableau



Analysis-12: Total number of trips in a given weather conditions

Performed the **Join** operation between the trips and weather dataset to get the count of all the trips in a range of temperatures.

1. Define the CSVLoader and upload the trips dataset from HDFS using CSVLoader.

```
DEFINE CSVLoader org.apache.pig.piggybank.storage.CSVLoader();

trips = LOAD 'hdfs://localhost:9000/cycleShare_pig/CycleShare/trip.csv' USING
CSVLoader AS (trip_id:INT, starttime:CHARARRAY, stoptime:CHARARRAY, bikeid:
CHARARRAY, tripduration:DOUBLE, from_station_name:CHARARRAY,
to_station_name:CHARARRAY, from_station_id:CHARARRAY,
to_station_id:CHARARRAY, usertype:CHARARRAY, gender:CHARARRAY,
birthyear:INT);
```

2. upload the weather data set from HDFS

```
weather = LOAD 'hdfs://localhost:9000/cycleShare_pig/CycleShare/weather.csv'
USING CSVLoader AS(Date:CHARARRAY, Max_Temperature_F:INT,
Mean_Temperature_F:INT, Min_TemperatureF:INT, Max_Dew_Point_F:INT,
MeanDew_Point_F:INT, Min_Dewpoint_F:INT, Max_Humidity:INT, Mean_Humidity:INT,
Min_Humidity:INT, Max_Sea_Level_Pressure_In:DOUBLE,
Mean_Sea_Level_Pressure_In:DOUBLE, Min_Sea_Level_Pressure_In:DOUBLE,
Max_Visibility_Miles:INT, Mean_Visibility_Miles:INT,
Min_Visibility_Miles:INT, Max_Wind_Speed_MPH:INT, Mean_Wind_Speed_MPH:INT,
Max_Gust_Speed_MPH:INT, Precipitation_In:DOUBLE, Events:CHARARRAY);
```

3. Filter the trips file to get the date

```
splitdata1 = FOREACH trips GENERATE FLATTEN(STRSPLIT(starttime, ' '));

all_dates = FOREACH splitdata1 GENERATE $0 as date;
```

4. Group the filtered data based on all_dates

```
all_dates_grouped = GROUP all_dates BY date;

all_dates_count = FOREACH all_dates_grouped GENERATE FLATTEN(group) as
(date), COUNT(all_dates) as use;
```

5. Filter the weather file to get the mean temperature

```
splitweather = FOREACH weather GENERATE Date, Mean_Temperature_F;
```

6. Performing the join operations

```
joined = JOIN all_dates BY date, splitweather BY Date;
```

7. Filtering the joined data

```
filtered = FOREACH joined GENERATE all_dates::date as date,
splitweather::Mean_Temperature_F as temp;
```

8. Calculating a bin id for each range of temperature

```
filtered_bin = FOREACH  filtered GENERATE date, temp, (temp – 10)*10/(110 –
9) as temp_bin;
```

```
filtered_band = FOREACH filtered_bin GENERATE date, temp, temp_bin,
temp_bin*101/10 + 10 as band_min, temp_bin*101/10 + 20 as band_max;
```

9. Grouping the temp

```
temp_group = GROUP filtered_band BY (temp_bin, band_min, band_max);
```

10. Filtering and counting the temps based on the bin id

```
temp_count = FOREACH temp_group GENERATE FLATTEN(group) as (temp_bin,
band_min, band_max), COUNT(filtered_band) as use;
```

```
(2,30,40,10719)
(3,40,50,63180)
(4,50,60,95795)
(5,60,70,78918)
(6,70,80,37586)
(7,80,90,550)
```

Appendix

Analysis-1: Number of trips by month-year

Mapper class

```
package Q2NumberOfTripsByMonth;
import org.apache.hadoop.io.IntWritable;
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, IntWritable> {
    private String[] months = {"January", "February", "March", "April", "May",
"June", "July", "August", "September",
   "October", "November", "December"};

    protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        String[] fields = value.toString().split(",");
        String date = fields[1].substring(1);
        if (!fields[1].contains("starttime")) {
            Integer month = Integer.parseInt(date.split("/")[0]);
            String [] dateParts = date.split("/");
            try {
                context.write(new Text(months[month-1] + "-" + dateParts[2].split("
")[0]), new IntWritable(1));
            } catch (IOException e) {
            }
        }
    }
}
```

Reducer Class

```
package Q2NumberOfTripsByMonth;
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 <Text, IntWritable, Text, IntWritable> {

    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
        int count = 0;
        for(IntWritable val: values) {
            count += val.get();
        }
        context.write(key, new IntWritable(count));
    }
}
```

Driver class

```
package Q2NumberOfTripsByMonth;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;

public class DriverClass {
    public static void main (String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "NumberOfTripsByMonth");
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.setJarByClass(DriverClass.class);
        job.setInputFormatClass(TextInputFormat.class);
        Path outDir = new Path(args[1]);
        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(outDir)) {
            fs.delete(outDir, true);
        }

        job.setMapperClass(MapperClass.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);
        job.setReducerClass(ReducerClass.class);
        job.setNumReduceTasks(1);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Analysis-2: Min, Max and Average duration of trips from each station

Custom Writable class

```
package Q3MinMaxAvgTripDuration;

public class TripWritable implements Writable {
    private double minTrip;
    private double maxTrip;
    private double avgTrip;
    private long count;
    public TripWritable() {
        super();
    }
    public TripWritable (double minTrip, double maxTrip, double avgTrip, long
count) {
        super();
        this.minTrip = minTrip;
        this.maxTrip = maxTrip;
        this.avgTrip = avgTrip;
        this.count = count;
    }
    public double getMinTrip() {return minTrip; }
    public void setMinTrip(double minTrip) {this.minTrip = minTrip;}
    public double getMaxTrip() {return maxTrip;}
    public void setMaxTrip(double maxTrip) {this.maxTrip = maxTrip;}
    public double getAvgTrip() {return avgTrip;}
    public void setAvgTrip(double avgTrip) {this.avgTrip = avgTrip;}
    public long getCount() {return count;}
   public void setCount(long count) {this.count = count;}
    public void write(DataOutput out) throws IOException {
        out.writeDouble(minTrip);
        out.writeDouble(maxTrip);
        out.writeDouble(avgTrip);
        out.writeLong(count);
    }
    public void readFields(DataInput in) throws IOException {
        minTrip = in.readDouble();
        maxTrip = in.readDouble();
        avgTrip = in.readDouble();
        count = in.readLong();
    }
   @Override
    public String toString() {
        return  "Min-Trip-Duration: " + minTrip + ", " +
                "Max-Trip-Duration: " + maxTrip + ", " +
                "Avg-Trip-Duration: " + avgTrip;
     }
}
```

Mapper class

```
package Q3MinMaxAvgTripDuration;
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, TripWritable> {

    Text stationKey = new Text();
    TripWritable tuple = new TripWritable();

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

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

        String duration = tokens[4];

        if(!tokens[7].equals("from_station_id") &&
!duration.contains("tripduration") && duration != null) {

            stationKey.set(tokens[7].substring(1, tokens[7].length()-1));

            double tripDuration = Double.parseDouble(duration);
            tuple.setMinTrip(tripDuration);
            tuple.setMaxTrip(tripDuration);
            tuple.setAvgTrip(tripDuration);
            tuple.setCount(1);

            context.write(stationKey, tuple);
        }
    }
}
```

Reducer Class

```
package Q3MinMaxAvgTripDuration;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

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

    TripWritable output = new TripWritable();

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

        double minTripDuration = Integer.MAX_VALUE;
        double maxTripDuration = Integer.MIN_VALUE;
        long count = 0;
        double sumOfDurations = 0;

        for(TripWritable val: values) {
            sumOfDurations += val.getAvgTrip() * val.getCount();
            count += val.getCount();

            if(val.getMinTrip() < minTripDuration) {
                minTripDuration = val.getMinTrip();
            }

            if(val.getMaxTrip() > maxTripDuration) {
                maxTripDuration = val.getMaxTrip();
            }
        }
        output.setMinTrip(minTripDuration);
        output.setMaxTrip(maxTripDuration);
        output.setAvgTrip(sumOfDurations / count);
        output.setCount(count);

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

Driver Class

```
package Q3MinMaxAvgTripDuration;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class DriverClass {

    public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "MinMaxAvgTripDuration");
        job.setJarByClass(DriverClass.class);

        job.setMapperClass(MapperClass.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(TripWritable.class);

        TextInputFormat.addInputPath(job, new Path(args[0]));

        job.setReducerClass(ReducerClass.class);

        job.setNumReduceTasks(1);

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

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

        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(outDir)) {
            fs.delete(outDir, true);
        }
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Analysis-3: Total number of trips from a station in each year

Composite key class

```
package Q4CountTripsPerStationPerYear;

public class YearStationPair implements WritableComparable
<YearStationPair> {
    private String year;
    private String station;
    public YearStationPair() { super();}
    public YearStationPair(String year, String station) {
        super();
        this.year = year;
        this.station = station;
    }

    public String getYear() {return year;}
    public void setYear(String year) {this.year = year;}
    public String getStation() {return station;}
    public void setStation(String station) {this.station = station; }
    public int compareTo(YearStationPair o) {
        int result = this.year.compareTo(o.year);
        if(result == 0) {
            return this.station.compareTo(o.station);
        }
        return result;
    }
    public void write(DataOutput out) throws IOException {
        out.writeUTF(year);
        out.writeUTF(station);
    }
    public void readFields(DataInput in) throws IOException {
        year = in.readUTF();
        station = in.readUTF();
    }
    @Override
    public String toString() {
        return year + "," + station;
    }
}
```

SecSortComparator class

```
package Q4CountTripsPerStationPerYear;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class SecSortComparator extends WritableComparator {

    public SecSortComparator() {
        super(YearStationPair.class, true);
        // super();
    }

    @Override
    public int compare(WritableComparable k1, WritableComparable k2) {
        YearStationPair key1 = (YearStationPair) k1;
        YearStationPair key2 = (YearStationPair) k2;

        int result = key1.compareTo(key2);
        return result;
    }
}
```

SecSortGroupComparator class

```
package Q4CountTripsPerStationPerYear;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class SecSortGroupComparator extends WritableComparator {

    public SecSortGroupComparator() {
        super(YearStationPair.class, true);

    }

    @Override
    public int compare(WritableComparable k1, WritableComparable k2) {
        YearStationPair key1 = (YearStationPair)k1;
        YearStationPair key2 = (YearStationPair)k2;
        return key1.compareTo(key2);
    }
}
```

Mapper class

```
package Q4CountTripsPerStationPerYear;

public class SecSortMapper extends Mapper <LongWritable, Text,
YearStationPair, IntWritable> {

    protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
        String[] tokens = value.toString().split(",");
        String year = null;
        String station = null;

        try {
            year = tokens[1].split("/")[2].substring(0, 4);
            station = tokens[7].substring(1, tokens[7].length() - 1);
        } catch (Exception e) {

        }

        if(year != null && station != null) {
            YearStationPair outkey = new YearStationPair(year, station);
            context.write(outkey, new IntWritable(1));
        }
    }
}
```

Key Partitioner

```
package Q4CountTripsPerStationPerYear;

public class KeyPartitioner extends Partitioner<YearStationPair,
IntWritable> {

    @Override
    public int getPartition(YearStationPair key, IntWritable value, int
numPartitions) {
        return key.getYear().hashCode() % numPartitions;
    }
}
```

Reducer class

```
package Q4CountTripsPerStationPerYear;

public class SecSortReducer extends Reducer<YearStationPair, IntWritable,
YearStationPair, IntWritable> {

    protected void reduce(YearStationPair key, Iterable<IntWritable>
values, Context context) throws IOException, InterruptedException {
        int count = 0;

        for (IntWritable val : values) {
            count += val.get();
        }
        context.write(key, new IntWritable(count));
    }
}
```

Driver Class

```
package Q4CountTripsPerStationPerYear;

public class DriverClass {
    public static void main (String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "tripsPerYearPerStation");
          job.setJarByClass(DriverClass.class);
          job.setGroupingComparatorClass(SecSortGroupComparator.class);
        job.setSortComparatorClass(SecSortComparator.class);
        job.setPartitionerClass(KeyPartitioner.class);

        TextInputFormat.addInputPath(job, new Path(args[0]));
        Path outDir = new Path(args[1]);
        TextOutputFormat.setOutputPath(job, outDir);

        job.setMapperClass(SecSortMapper.class);
        job.setReducerClass(SecSortReducer.class);

        job.setNumReduceTasks(1);

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

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

Analysis-4: To find out the top 5 busy stations by month

Composite key class

```
package Q5Top5BusiestStations;
public class MonthStationPair implements WritableComparable
<MonthStationPair> {
    private String month;
    private String station;
    public MonthStationPair() {super();}
    public MonthStationPair(String month, String staion) {
        super();
        this.month = month;
        this.station = staion;
    }
    public String getMonth() {return month;}

    public void setMonth(String month) {this.month = month;}
    public String getStation() {return station;}
    public void setStation(String station) {this.station  =station;}
    public int compareTo(MonthStationPair o) {
        int result = this.month.compareTo(o.month);
        if(result == 0) {
            return this.station.compareTo(o.station);
        }
        return result;
    }
    public void write(DataOutput out) throws IOException {
        out.writeUTF(month);
        out.writeUTF(station);
    }
    public void readFields(DataInput in) throws IOException {
        month = in.readUTF();
        station = in.readUTF();
    }
    @Override
    public String toString() {
        return month + ", " + station;
    }
}
```

SecSortComparator Class

```
package Q5Top5BusiestStations;
public class SecSortComparator extends WritableComparator {
    public SecSortComparator() {
        super(MonthStationPair.class, true);
    }
    @Override
    public int compare(WritableComparable k1, WritableComparable k2) {
        MonthStationPair key1 = (MonthStationPair) k1;
        MonthStationPair key2 = (MonthStationPair) k2;

        int result = key1.compareTo(key2); // -1, 0, 1
        return result;
    }
}
```

SecSortGroupComparator Class

```
package Q5Top5BusiestStations;
public class SecSortGroupComparator extends WritableComparator {

    public SecSortGroupComparator() {
        super(MonthStationPair.class, true);
    }

    @Override
    public int compare(WritableComparable k1, WritableComparable k2) {
        MonthStationPair key1 = (MonthStationPair)k1;
        MonthStationPair key2 = (MonthStationPair) k2;
        return key1.compareTo(key2);
    }
}
```

Mapper Class

```
package Q5Top5BusiestStations;
public class MapperClass extends Mapper <LongWritable, Text, MonthStationPair,
IntWritable> {
    protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        String[] tokens = value.toString().split(",");

        String  month = null;
        String station = null;
        try {
            month = tokens[1].split("/")[0].substring(1);
            station = tokens[7].substring(1, tokens[7].length() - 1);
        } catch (Exception e) {}
        if(month != null  && station != null) {
            MonthStationPair outkey = new MonthStationPair(month, station);
            context.write(outkey, new IntWritable(1));
        }
    }
}
```

Partitioner Class

```
package Q5Top5BusiestStations;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Partitioner;

public class KeyPartitioner extends Partitioner <MonthStationPair,
IntWritable> {

    @Override
    public int getPartition(MonthStationPair key, IntWritable value, int
numPartitions) {
        return key.getMonth().hashCode() % numPartitions;
    }
}
```

Reducer Class

```
package Q5Top5BusiestStations;
public class ReducerClass extends Reducer <MonthStationPair, IntWritable,
MonthStationPair, IntWritable> {
    protected void reduce(MonthStationPair key, Iterable<IntWritable>
values, Context context) throws IOException, InterruptedException {
        int count = 0;

        for (IntWritable val : values) {
            count += val.get();
        }
        context.write(key, new IntWritable(count));
    }
}
```

Custom Writable class

```
package Q5Top5BusiestStations;
public class StationCountWrittable implements Writable {
    private String station;
    private Integer count;
    public StationCountWrittable() {super();}
    public StationCountWrittable(String station, Integer count) {
        super();
        this.station = station;
        this.count = count;
    }
    public String getStation() {return station;}

    public void setStation(String station) {this.station = station; }

    public Integer getCount() {return count;}
    public void setCount(Integer count) {this.count = count;}
    public void write(DataOutput out) throws IOException {
        out.writeUTF(station);
        out.writeInt(count);
    }
    public void readFields(DataInput in) throws IOException {
        station = in.readUTF();
        count = in.readInt();
    }
    public String toString() {
        return station + ", " + count;
    }
}
```

Mapper class

```
package Q5Top5BusiestStations;
public class Top5Mapper extends Mapper <LongWritable, Text, IntWritable,
StationCountWrittable> {

    protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
        StationCountWrittable outvalue = new StationCountWrittable();
        if (value.toString().contains("starttime")) {
            return;
        }
        String[] tokens = value.toString().split(",");  // month, station
count
        String[] stationCount = tokens[1].substring(1).split("\\s+");
        int month = Integer.parseInt(tokens[0]);
        outvalue.setStation(stationCount[0]);
        outvalue.setCount(Integer.parseInt(stationCount[stationCount.length
- 1]));
        context.write(new IntWritable(month), outvalue);
    }
}
```

Reducer class

```
package Q5Top5BusiestStations;
public class Top5Reducer extends Reducer <IntWritable,
StationCountWrittable, IntWritable, Text> {
    @Override
    protected void reduce(IntWritable key, Iterable<StationCountWrittable>
values, Context context) throws IOException, InterruptedException {
        List <StationCountWrittable> allStations = new
ArrayList<StationCountWrittable>();

        for (StationCountWrittable value : values) {
            allStations.add(new StationCountWrittable(value.getStation(),
value.getCount()));
        }

        int count = 0;
        StringBuilder stations = new StringBuilder();
        stations.append("");

        Collections.sort(allStations, new
Comparator<StationCountWrittable>() {
            public int compare(StationCountWrittable o1,
StationCountWrittable o2) {
                return o2.getCount() - o1.getCount();
            }
        });

        String prefix = "";
        while(count < 5 && count < allStations.size()) {
            StationCountWrittable station = allStations.get(count);
            stations.append(prefix).append(station.getStation());
//          stations.append(prefix).append(station.getStation()).append("
[").append(station.getCount()).append("]");
            prefix = ", ";
            count++;
        }

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

Driver Class

```
package Q5Top5BusiestStations;
public class DriverClass {
    public static void main(String[] args) throws IOException, ClassNotFoundException,
InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Q8SecondarySorting");
          job.setJarByClass(DriverClass.class);
       job.setGroupingComparatorClass(SecSortGroupComparator.class);
        job.setSortComparatorClass(SecSortComparator.class);
        job.setPartitionerClass(KeyPartitioner.class);

        TextInputFormat.addInputPath(job, new Path(args[0]));
        Path outDir = new Path(args[1]);
        TextOutputFormat.setOutputPath(job, outDir);

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

        job.setNumReduceTasks(1);

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

        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(outDir)) {
            fs.delete(outDir, true);
        }
        boolean result = job.waitForCompletion(true);

        if(result)
        {
            Job job1 = Job.getInstance();
            job1.setJarByClass(DriverClass.class);

            job1.setMapperClass(Top5Mapper.class);
            job1.setReducerClass(Top5Reducer.class);

            job1.setMapOutputKeyClass(IntWritable.class);
            job1.setMapOutputValueClass(StationCountWrittable.class);

            job1.setOutputKeyClass(IntWritable.class);
            job1.setOutputValueClass(Text.class);

            // input to this is the output of the first mapper
            TextInputFormat.addInputPath(job1, new Path(args[1]));
            // final output will be stored in the 3rd argument
            Path finalOutputDirectory = new Path(args[2]);

            TextOutputFormat.setOutputPath(job1, finalOutputDirectory);

            FileSystem fs2 = FileSystem.get(job1.getConfiguration());
            if(fs2.exists(finalOutputDirectory))
            {
                fs2.delete(finalOutputDirectory, true);
            }
            job1.waitForCompletion(true);
        }
    }
}
```

Analysis-5: Most active age groups

Composite key class

```
package Q6MostActiveAgeGroup;
public class MemberandAge implements WritableComparable <MemberandAge> {
    private String ageBand;
    public MemberandAge() {super();}
    public MemberandAge(String band) {
        super();
        this.ageBand = band;
    }
    public String getAgeBand() {return ageBand;}
    public void setAgeBand(String ageBand) {this.ageBand = ageBand; }
    public int compareTo(MemberandAge o) {
        return this.ageBand.compareTo(o.getAgeBand());}
    public void write(DataOutput out) throws IOException {
        out.writeUTF(ageBand);
    }
    public void readFields(DataInput in) throws IOException {
        ageBand = in.readUTF();
    }
    @Override
    public String toString() {
        return ageBand;
    }
}
```

Mapper Class

```
package Q6MostActiveAgeGroup;
public class MapperClass extends Mapper <LongWritable, Text, MemberandAge,
IntWritable> {

    MemberandAge tuple = new MemberandAge();

    protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
        String[] tokens = value.toString().split(",");

        String member = tokens[9].substring(1, tokens[9].length()-1);
        if (member.contains("Short-Term Pass Holder")) {
            return;
        }

        try {
            String birthyearString = tokens[11];
            if (!tokens[9].contains("usertype") && member != "" &&
!birthyearString.contains("birthyear") && birthyearString != "") {
                String year = tokens[1].substring(1);
                String[] dateParts = year.split("/");
                int currentyear = Integer.parseInt(dateParts[2].split("
")[0]);
                int birthyear = Integer.parseInt(birthyearString);
                int diff = currentyear - birthyear;
                if (diff < 30) {
                    tuple.setAgeBand("[LT-30)");
                } else if (diff >= 30 && diff < 40) {
                    tuple.setAgeBand("[30-40)");
                } else if (diff >= 40 && diff < 50) {
                    tuple.setAgeBand("[40-50)");
                } else {
                    tuple.setAgeBand("[50+)");
                }
                context.write(tuple, new IntWritable(1));
            }
        } catch (Exception e) {
            // Skipping exceptions caused by data formatting
        }

    }
}
```

Reducer Class

```
package Q6MostActiveAgeGroup;
public class ReducerClass extends Reducer <MemberandAge, IntWritable,
MemberandAge, IntWritable> {
    MemberandAge output = new MemberandAge();

    protected void reduce(MemberandAge key, Iterable <IntWritable> values,
Context context) throws IOException, InterruptedException {
        int count = 0;
        for(IntWritable val : values) {
            count += val.get();
        }
        context.write(key, new IntWritable(count));
    }
}
```

Driver class

```
package Q6MostActiveAgeGroup;
public class DriverClass {
    public static void main(String args[]) throws IOException,
ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "MembershipsBasedOnAgeBands");
        job.setJarByClass(DriverClass.class);

        job.setMapperClass(MapperClass.class);
        job.setMapOutputKeyClass(MemberandAge.class);
        job.setMapOutputValueClass(IntWritable.class);

        TextInputFormat.addInputPath(job, new Path(args[0]));

        job.setReducerClass(ReducerClass.class);

        job.setNumReduceTasks(1);

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

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

        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(outDir)) {
            fs.delete(outDir, true);
        }
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Analysis-6: Number of trips in a day from each station and the corresponding weather on that day

Trip Mapper Class

```
package Q7JoinPattern;
public class TripMapper extends Mapper<LongWritable, Text, Text, Text> {
    private Text outkey = new Text();
    private Text outvalue = new Text();
    @Override
    protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
        String[] tokens = value.toString().split(",");
        String[] datepart = tokens[1].substring(1).split(" ");
        outkey.set(datepart[0]);
        outvalue.set("A" + value.toString());
        context.write(outkey, outvalue);
    }
}
```

Weather Mapper Class

```
package Q7JoinPattern;
public class WeatherMapper extends Mapper <LongWritable, Text, Text, Text>
{
    private Text outkey = new Text();
    private Text outvalue = new Text();

    @Override
    protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
        String[] tokens = value.toString().split(",");

        outkey.set(tokens[0].substring(1, tokens[0].length()-1)); // date
        outvalue.set("B" + value.toString());
        context.write(outkey, outvalue);
    }
}
```

Reducer Class

```
package Q7JoinPattern;
public class ReducerClass extends Reducer<Text, Text, Text, Text> {
    private static final Text EMPTY_TEXT = new Text();
    private List<Text> listA = new ArrayList<Text>();
    private List<Text> listB = new ArrayList<Text>();
    private String joinType = null;
    @Override
    protected void setup(Context context) {
        joinType = context.getConfiguration().get("join.type");
    }

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

        for(Text val : values)
        {
            if(val.charAt(0) == 'A')
                listA.add(new Text(val.toString().substring(1)));
            else if(val.charAt(0) == 'B')
                listB.add(new Text(val.toString().substring(1)));
        }
        executeJoinLogic(context);
    }

    private void executeJoinLogic(Context context) throws IOException,
InterruptedException {
        // if both the lists are not empty, perform two nested for loops
and join each of the values together
        if (joinType.equalsIgnoreCase("inner")) {
            if (!listA.isEmpty() && !listB.isEmpty()) {
                for (Text a : listA) {
                    for (Text b : listB) {
                        context.write(a, b);
                    }
                }
            }
        }
    }
}
```

Driver class

```
package Q7JoinPattern;
public class DriverClass {
    public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);

        job.setJarByClass(DriverClass.class);

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

        MultipleInputs.addInputPath(job, new Path(args[0]),
TextInputFormat.class, TripMapper.class);
        MultipleInputs.addInputPath(job, new Path(args[1]),
CombineTextInputFormat.class,WeatherMapper.class);

        job.setReducerClass(ReducerClass.class);

        job.getConfiguration().set("join.type",args[2]);

        FileOutputFormat.setOutputPath(job, new Path(args[3]));

        FileSystem fs = FileSystem.get(conf);
        fs.delete(new Path(args[3]), true);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Analysis-7: Custom MapReduce algorithm to find the top 10 most busy routes

Composite key class

```
package Q8Top5BusyRoutes;
public class FromToStationPair implements
WritableComparable<FromToStationPair> {
    private String fromStationId;
    private String toStationId;

    public FromToStationPair() {super();}
    public FromToStationPair(String fromStationId, String toStationId) {
        super();
        this.fromStationId = fromStationId;
        this.toStationId = toStationId;
    }
    public String getFromStationId() {return fromStationId;}
    public void setFromStationId(String fromStationId) {
        this.fromStationId = fromStationId;}

    public String getToStationId() {return toStationId;}
    public void setToStationId(String toStationId) {this.toStationId =
toStationId;}
    public int compareTo(FromToStationPair o) {
int result = this.fromStationId.compareTo(o.fromStationId);
        if(result == 0) {
            return this.toStationId.compareTo(o.toStationId);
        }
        return result;
    }
    public void write(DataOutput out) throws IOException {
        out.writeUTF(fromStationId);
        out.writeUTF(toStationId);
    }
    public void readFields(DataInput in) throws IOException {
        fromStationId = in.readUTF();
        toStationId = in.readUTF();
    }
    @Override
    public String toString() {
        return fromStationId + ", " + toStationId;
    }
}
```

SecSortComparator Class

```
package Q8Top5BusyRoutes;
public class SecSortComparator extends WritableComparator {
    public SecSortComparator() {
        super(FromToStationPair.class, true);
    }
    @Override
    public int compare(WritableComparable k1, WritableComparable k2) {
        FromToStationPair key1 = (FromToStationPair) k1;
        FromToStationPair key2 = (FromToStationPair) k2;

        int result = key1.compareTo(key2); // -1, 0, 1
        return result;
    }
}
```

SecSortGroupComparator Class

```
package Q8Top5BusyRoutes;
public class SecSortGroupComparator extends WritableComparator {

    public SecSortGroupComparator() {
        super(FromToStationPair.class, true);
    }

    @Override
    public int compare(WritableComparable k1, WritableComparable k2) {
        FromToStationPair key1 = (FromToStationPair) k1;
        FromToStationPair key2 = (FromToStationPair) k2;
        return key1.compareTo(key2);
    }
}
```

Mapper class

```
package Q8Top5BusyRoutes;
public class SecSortMapper extends Mapper <LongWritable, Text,
FromToStationPair, IntWritable> {

    protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
        String[] tokens = value.toString().split(",");

        String fromstn = null;
        String tostn = null;

        try {
            fromstn = tokens[7].substring(1, tokens[7].length() - 1);
            tostn = tokens[8].substring(1, tokens[7].length()-1);
        } catch (Exception e) {

        }

        if(fromstn != null && tostn != null) {
            FromToStationPair outkey = new FromToStationPair(fromstn,
tostn);
            if (fromstn.compareTo(tostn) > 0) {
                outkey.setToStationId(fromstn);
                outkey.setFromStationId(tostn);
            }
            context.write(outkey, new IntWritable(1));
        }
    }
}
```

Key Partitioner

```
package Q8Top5BusyRoutes;
public class KeyPartitioner extends Partitioner<FromToStationPair,
IntWritable> {
    @Override
    public int getPartition(FromToStationPair key, IntWritable value, int
numPartitions) {
        return key.getFromStationId().hashCode() % numPartitions;
    }
}
```

Reducer Class

```
package Q8Top5BusyRoutes;
public class SecSortReducer extends Reducer <FromToStationPair,
IntWritable, FromToStationPair, IntWritable> {

    protected void reduce(FromToStationPair key, Iterable<IntWritable>
values, Context context) throws IOException, InterruptedException {
        int count = 0;

        for(IntWritable val: values) {
            count += val.get();
        }
        context.write(key, new IntWritable(count));
    }
}
```

Mapper Class

```
package Q8Top5BusyRoutes;
public class Top10Mapper extends Mapper <LongWritable, Text,
FromToStationPair, IntWritable> {

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

        String[] keyVals = value.toString().split(",");

        String fromStation = "";
        String[] keyParts = keyVals[0].split("\\s+");
        if (keyParts.length > 1) {
            String prefix = "";
            for(String keyPart: keyParts) {
                fromStation = fromStation + prefix + keyPart;
                prefix = " ";
            }
        } else {
            fromStation = keyVals[0];
        }

        String[] tokens = keyVals[1].trim().split("\\s+");
        int val = Integer.parseInt(tokens[tokens.length-1]);
        String toStation = "";

        if (tokens.length > 2) {
            String prefix = "";
            for(int i = 0; i < tokens.length - 2; i++) {
                toStation = toStation + prefix + tokens[i];
                prefix = " ";
            }
        } else {
            toStation = tokens[0];
        }

        FromToStationPair outkey = new FromToStationPair();
        outkey.setFromStationId(fromStation);
        outkey.setToStationId(toStation);

        context.write(outkey, new IntWritable(val));
    }
}
```

Reducer Class

```
package Q8Top5BusyRoutes;
public class Top10Reducer extends Reducer <FromToStationPair, IntWritable,
Text, Text> {
    private PriorityQueue<RouteFreq> heap;

    protected void setup(Context context) throws IOException,
InterruptedException {
        heap = new PriorityQueue<RouteFreq>(new Comparator<RouteFreq>() {
            public int compare(RouteFreq o1, RouteFreq o2) {
                return o2.freq - o1.freq;
            }
        });
    }
    @Override
    protected void reduce(FromToStationPair key, Iterable<IntWritable>
values, Context context) throws IOException, InterruptedException {
        int count = 0;

        for(IntWritable val: values) {
            count += val.get();
        }
        System.out.println(" -- Adding route: " + key.getFromStationId() +
" - " + key.getToStationId() + " = " + count);

        heap.add(new RouteFreq(count, new
FromToStationPair(key.getFromStationId(), key.getToStationId())));
    }
    protected void cleanup(Context context) throws IOException,
InterruptedException {
        int count = 0;
        StringBuilder sb = new StringBuilder();
        while(!heap.isEmpty() && count < 10) {
            RouteFreq route = heap.poll();
            sb.append(route.key.getFromStationId()).append(" <->
").append(route.key.getToStationId())
                    .append(":").append(route.freq).append("\n");
            count++;
        }
       context.write(new Text("Top 10 Routes"), new Text(sb.toString()));
    }
   class RouteFreq {
        int freq;
        FromToStationPair key;

        public RouteFreq(int freq, FromToStationPair key) {
            this.freq = freq;
           this.key = key;
        }
    }
}
```

Driver Class

```
package Q8Top5BusyRoutes;
public class DriverClass {
    public static void main (String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "fromTostationPairCount");
        job.setJarByClass(DriverClass.class);
job.setGroupingComparatorClass(SecSortGroupComparator.class);
        job.setSortComparatorClass(SecSortComparator.class);
        job.setPartitionerClass(KeyPartitioner.class);

        TextInputFormat.addInputPath(job, new Path(args[0]));
        Path outDir = new Path(args[1]);
        TextOutputFormat.setOutputPath(job, outDir);

        job.setMapperClass(SecSortMapper.class);
        job.setReducerClass(SecSortReducer.class);

        job.setNumReduceTasks(1);

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

        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(outDir)) {
            fs.delete(outDir, true);
        }
        boolean result = job.waitForCompletion(true);
        if(result)
        {
            Job job1 = Job.getInstance();
            job1.setJarByClass(DriverClass.class);

            job1.setMapperClass(Top10Mapper.class);
            job1.setReducerClass(Top10Reducer.class);

            job1.setMapOutputKeyClass(FromToStationPair.class);
            job1.setMapOutputValueClass(IntWritable.class);

            job1.setOutputKeyClass(FromToStationPair.class);
            job1.setOutputValueClass(IntWritable.class);

            TextInputFormat.addInputPath(job1, new Path(args[1]));
            Path finalOutputDirectory = new Path(args[2]);

            TextOutputFormat.setOutputPath(job1, finalOutputDirectory);

            FileSystem fs2 = FileSystem.get(job1.getConfiguration());
            if(fs2.exists(finalOutputDirectory))
            {
                fs2.delete(finalOutputDirectory, true);
            }
            job1.waitForCompletion(true);
        }
    }
}
```

Analysis-8: Count membership by gender

Mapper Class

```
package Q9CountMemershipByGender;
public class MapperClass extends Mapper<LongWritable, Text, NullWritable,
NullWritable> {

    public static final String GENDER = "gender";

    protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
        String[] tokens = value.toString().split(",");

        String user = tokens[9].substring(1,7);

        if("Member".equals(user) && !tokens[10].contains("gender") &&
!tokens[10].contains("Other")) {
            String gender = tokens[10].substring(1, tokens[10].length()-1);
            try {
                context.getCounter(GENDER, gender).increment(1);
            } catch (Exception e) {
            }
        }
    }
}
```

Driver Class

```
package Q9CountMemershipByGender;
public class DriverClass {
    public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);

        job.setJarByClass(DriverClass.class);
        job.setMapperClass(MapperClass.class);

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

        FileSystem fs = FileSystem.get(conf);
        fs.delete(new Path(args[1]), true);

        int code = job.waitForCompletion(true) ? 0 : 1;

        if(code == 0) {
            for (Counter counter:
job.getCounters().getGroup(MapperClass.GENDER)) {
                System.out.println(counter.getDisplayName()+ "\t" +
counter.getValue());
            }
        }
        fs.get(conf).delete(new Path(args[1]), true);
        System.exit(code);
    }
}
```