INFO 7250 – Engineering Bigdata Systems

Big data Analysis on flight data using Hadoop MapReduce, PIG, HIVE and MongoDB

By

Venkata Lakshmi Mahitha, Mandalapu
NUID: 001387807

Table of Contents

1. Introduction

1.1. About Dataset:

The flight dataset consists of flight arrival and departure for all commercial flights within the USA, from October 1987 to April 2008. This is a large dataset: there are nearly 120 million records in total and takes up to 1.6GB of space compressed and 12GB when uncompressed. The dataset has many columns so that wide range of analysis can be performed. The dataset is available on the following link: http://stat-computing.org/dataexpo/2009/the-data.html.

The columns and description of each of the column is as below:

Column No	Name	Description
1	Year	1987-2008
2	Month	1-12
3	DayofMonth	1-31
4	DayOfWeek	1 (Monday) - 7 (Sunday)
5	DepTime	actual departure time (local, hhmm)
6	CRSDepTime	scheduled departure time (local, hhmm)
7	ArrTime	actual arrival time (local, hhmm)
8	CRSArrTime	scheduled arrival time (local, hhmm)
9	UniqueCarrier	unique carrier code
10	FlightNum	flight number
11	TailNum	plane tail number
12	ActualElapsedTime	in minutes
13	CRSElapsedTime	in minutes
14	AirTime	in minutes
15	ArrDelay	arrival delay, in minutes
16	DepDelay	departure delay, in minutes
17	Origin	origin IATA airport code
18	Dest	destination IATA airport code
19	Distance	in miles
20	TaxiIn	taxi in time, in minutes
21	TaxiOut	taxi out time in minutes
22	Cancelled	was the flight cancelled?
23	CancellationCode	reason for cancellation (A = carrier, B = weather, C = NAS, D = security)
24	Diverted	1 = yes, 0 = no
25	CarrierDelay	in minutes
26	WeatherDelay	in minutes
27	NASDelay	in minutes
28	SecurityDelay	in minutes
29	LateAircraftDelay	in minutes

1.2. Frameworks used for analysis:

Following are the frameworks used for different analysis performed:

- Hadoop Mapreduce: Different Algorithms like Join patterns, filtering patterns, Average, minimum, Top10, Secondary sorting, Inverting index etc are used for the analysis
- PIG
- HIVE
- MongoDB

2. Analysis using Hadoop MapReduce

2.1. Copy data from local to HDFS:

From the dataset link mentioned above all year's data from 1987 to 2008 which are in .bz2 format are downloaded manually and are unzipped using linux commands as below:

```
(base) Maheswaras-MacBook-Pro:data maheswararaogunturi$ cd /MAHI_NEU/EBD/Project/data/
(base) Maheswaras-MacBook-Pro:data maheswararaogunturi$ ls
1987.csv.bz2    1989.csv.bz2    1991.csv.bz2    1993.csv.bz2    1995.csv.bz2    1997.csv.bz2    1999.csv.bz2    2001.csv.bz2    2003.csv.bz2    2005.csv.bz2    2007.csv.bz2
1988.csv.bz2    1990.csv.bz2    1992.csv.bz2    1994.csv.bz2    1996.csv.bz2    1998.csv.bz2    2000.csv.bz2    2002.csv.bz2    2004.csv.bz2    2006.csv.bz2    2008.csv.bz2
(base) Maheswaras-MacBook-Pro:data maheswararaogunturi$
(base) Maheswaras-MacBook-Pro:data maheswararaogunturi$ bzip2 -d *.bz2
(base) Maheswaras-MacBook-Pro:data maheswararaogunturi$
(base) Maheswaras-MacBook-Pro:data maheswararaogunturi$ ls
1987.csv        1989.csv        1991.csv        1993.csv        1995.csv        1997.csv        1999.csv        2001.csv        2003.csv        2005.csv        2007.csv
1988.csv        1990.csv        1992.csv        1994.csv        1996.csv        1998.csv        2000.csv        2002.csv        2004.csv        2006.csv        2008.csv
(base) Maheswaras-MacBook-Pro:data maheswararaogunturi$
```

After extracting the data to a local folder, it is copied to HDFS path using Hadoop commands as below:

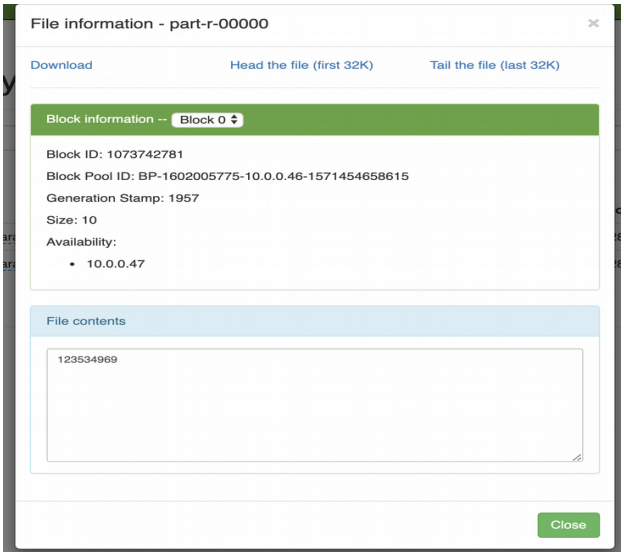
```
hadoop                  hdfs                    mapred                  oom-listener            yarn
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop fs -mkdir /flightData
2019-11-29 21:42:52,484 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop fs -copyFromLocal /MAHI_NEU/EBD/Project/data/* /flightData
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop fs -ls /flightData
2019-11-29 23:20:09,778 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 22 items
-rw-r--r--   1 maheswararaogunturi supergroup  127162942 2019-11-29 21:43 /flightData/1987.csv
-rw-r--r--   1 maheswararaogunturi supergroup  501039472 2019-11-29 21:43 /flightData/1988.csv
-rw-r--r--   1 maheswararaogunturi supergroup  486518821 2019-11-29 21:43 /flightData/1989.csv
-rw-r--r--   1 maheswararaogunturi supergroup  509194687 2019-11-29 21:43 /flightData/1990.csv
-rw-r--r--   1 maheswararaogunturi supergroup  491210093 2019-11-29 21:43 /flightData/1991.csv
-rw-r--r--   1 maheswararaogunturi supergroup  492313731 2019-11-29 21:43 /flightData/1992.csv
-rw-r--r--   1 maheswararaogunturi supergroup  490753652 2019-11-29 21:43 /flightData/1993.csv
-rw-r--r--   1 maheswararaogunturi supergroup  501558665 2019-11-29 21:43 /flightData/1994.csv
-rw-r--r--   1 maheswararaogunturi supergroup  530751568 2019-11-29 21:43 /flightData/1995.csv
-rw-r--r--   1 maheswararaogunturi supergroup  533922363 2019-11-29 21:43 /flightData/1996.csv
-rw-r--r--   1 maheswararaogunturi supergroup  540347861 2019-11-29 21:43 /flightData/1997.csv
-rw-r--r--   1 maheswararaogunturi supergroup  538432875 2019-11-29 21:43 /flightData/1998.csv
-rw-r--r--   1 maheswararaogunturi supergroup  552926022 2019-11-29 21:43 /flightData/1999.csv
-rw-r--r--   1 maheswararaogunturi supergroup  570151613 2019-11-29 21:43 /flightData/2000.csv
-rw-r--r--   1 maheswararaogunturi supergroup  600411462 2019-11-29 21:43 /flightData/2001.csv
-rw-r--r--   1 maheswararaogunturi supergroup  530507013 2019-11-29 21:43 /flightData/2002.csv
-rw-r--r--   1 maheswararaogunturi supergroup  626745242 2019-11-29 21:43 /flightData/2003.csv
-rw-r--r--   1 maheswararaogunturi supergroup  669879113 2019-11-29 21:43 /flightData/2004.csv
-rw-r--r--   1 maheswararaogunturi supergroup  671027265 2019-11-29 21:43 /flightData/2005.csv
-rw-r--r--   1 maheswararaogunturi supergroup  672068096 2019-11-29 21:43 /flightData/2006.csv
-rw-r--r--   1 maheswararaogunturi supergroup  702878193 2019-11-29 21:43 /flightData/2007.csv
-rw-r--r--   1 maheswararaogunturi supergroup  689413344 2019-11-29 21:43 /flightData/2008.csv
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$
```

2.2. Total number of flight travels from 1987 to 2008 in USA (Count Algorithm):

The total number of trips made over 22 years or in other words the total count of the data is calculated as below:

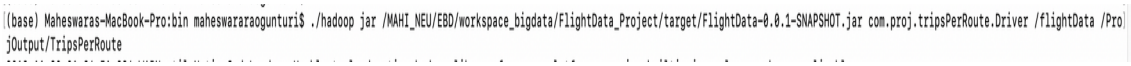
```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.numOfFlightTravels.Driver /flightData
 /ProjOutput/TotalTravelsCount
```

(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.numOfFlightTravels.Driver /flightData /ProjOutput/TotalTravelsCount

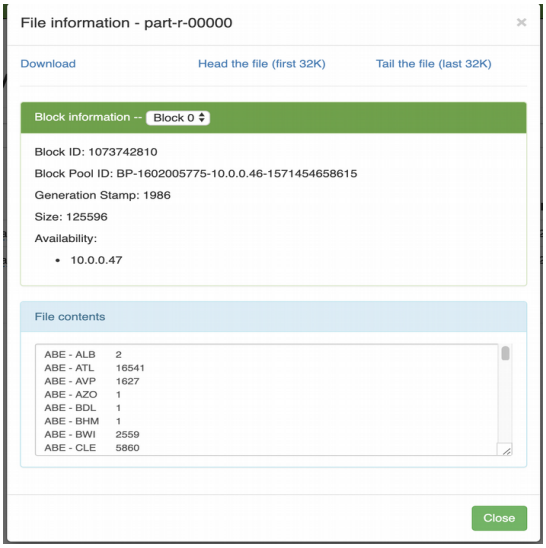


2.3. Number of trips in each Route: (Count Algorithm):

The total number of trips between each origin and destination route is calculated as below:



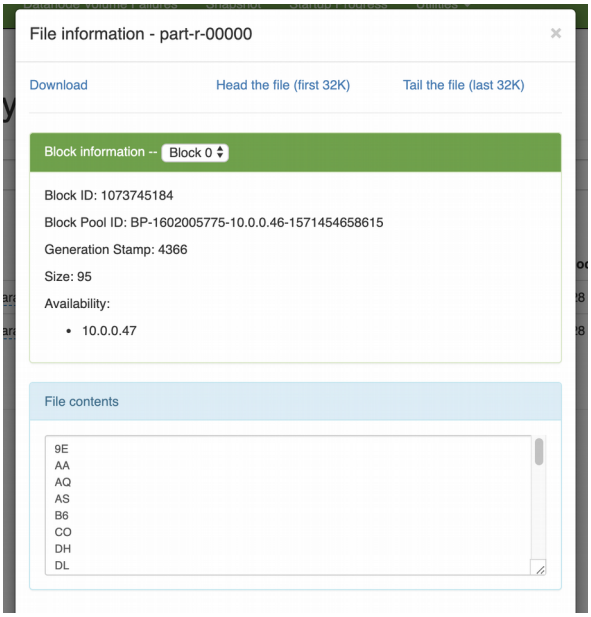
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./□adoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.tripsPerRoute.Driver /flightData /ProjOutput/TripsPerRoute



2.4. Finding distinct Unique Carriers (Distinct Filter Pattern Algorithm):

Each unique carrier has many flights in the data. Distinct unique carriers are obtained as below:

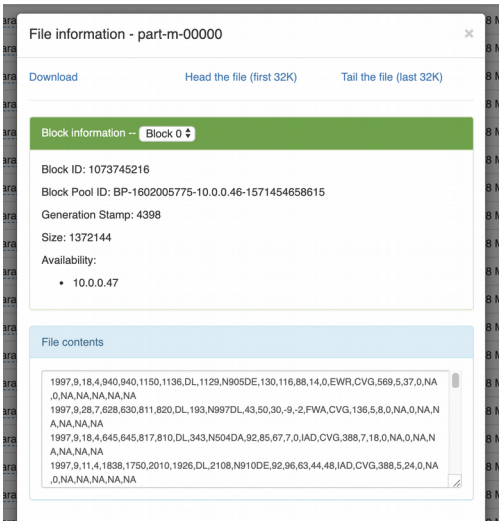
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.distinctCarriers.Driver /flightData /ProjOutput/DistinctUniqueCarriers



2.5. Simple Random Sampling (Filtering pattern Algorithm):

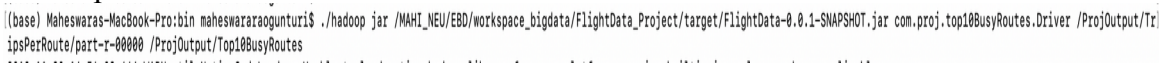
Simple random sampling of entire data set into a subset of our larger data set in which each record has an equal probability of being selected is performed as below:

(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.srSampling.Driver /flightData /ProjOutput/SRSampling



2.6. Top 10 Busiest Routes (Top10 Algorithm):

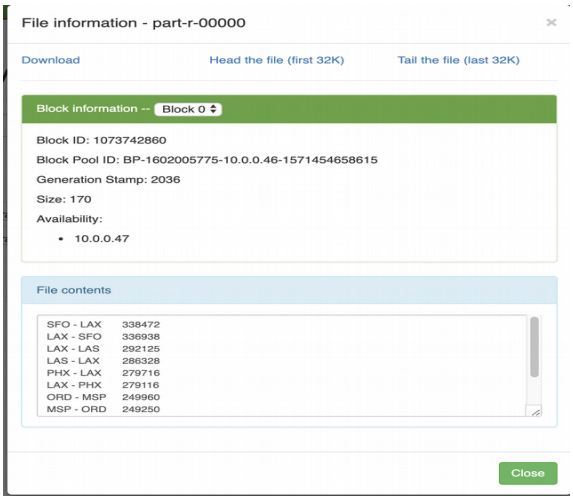
The Top 10 busiest routes are calculated as below:



(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.top10BusyRoutes.Driver /ProjOutput/TripsPerRoute/part-r-00000 /ProjOutput/Top10BusyRoutes

The time taken for the analysis is calculated so that performance can be analyzed in the later stages:

```
                Virtual memory (bytes) snapshot=0
                Total committed heap usage (bytes)=689438720
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=125596
        File Output Format Counters
                Bytes Written=170
time taken = 575.252
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$
```



2.7. Partitioning based on year (Data Organization Algorithm):

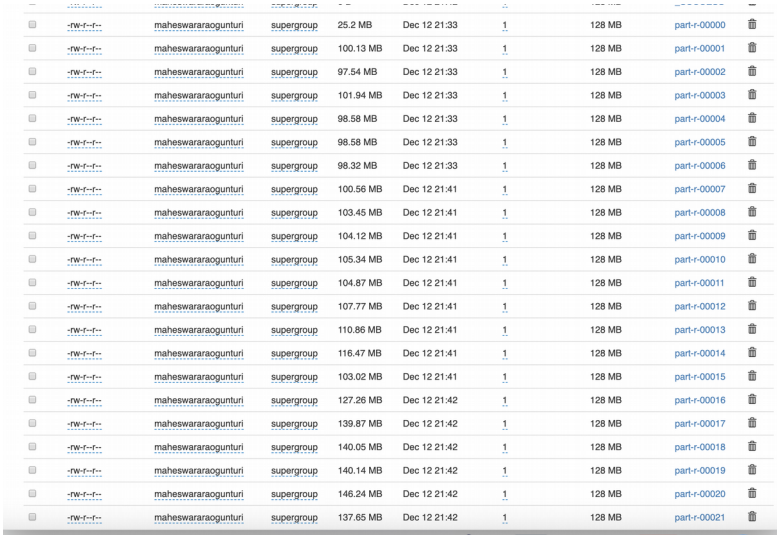
Using this Algorithm, we get the data divided on the basis of years. Here we have partitioned into 22 years and the output looks as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar
 com.proj.partitioningByYear.Driver /flightData /ProjOutput/PartitioningByYear
```

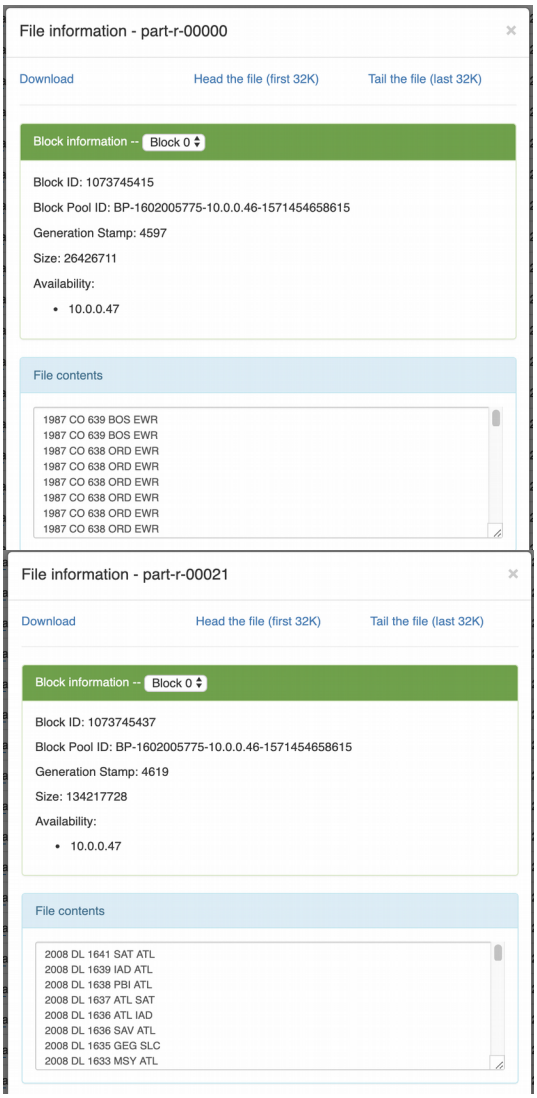
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.partitioningByYear.Driver /flightData /ProjOutput/PartitioningByYear

```
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=12029517197
        File Output Format Counters
                Bytes Written=2524951576
time taken = 879.26
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$
```

We can see the output is partitioned into 22 parts as there are 22 years and each part has data of different years:



-rw-r--r--	maheswararaogunturi	supergroup	25.2 MB	Dec 12 21:33	1	128 MB	part-r-00000
-rw-r--r--	maheswararaogunturi	supergroup	100.13 MB	Dec 12 21:33	1	128 MB	part-r-00001
-rw-r--r--	maheswararaogunturi	supergroup	97.54 MB	Dec 12 21:33	1	128 MB	part-r-00002
-rw-r--r--	maheswararaogunturi	supergroup	101.94 MB	Dec 12 21:33	1	128 MB	part-r-00003
-rw-r--r--	maheswararaogunturi	supergroup	98.58 MB	Dec 12 21:33	1	128 MB	part-r-00004
-rw-r--r--	maheswararaogunturi	supergroup	98.58 MB	Dec 12 21:33	1	128 MB	part-r-00005
-rw-r--r--	maheswararaogunturi	supergroup	98.32 MB	Dec 12 21:33	1	128 MB	part-r-00006
-rw-r--r--	maheswararaogunturi	supergroup	100.56 MB	Dec 12 21:41	1	128 MB	part-r-00007
-rw-r--r--	maheswararaogunturi	supergroup	103.45 MB	Dec 12 21:41	1	128 MB	part-r-00008
-rw-r--r--	maheswararaogunturi	supergroup	104.12 MB	Dec 12 21:41	1	128 MB	part-r-00009
-rw-r--r--	maheswararaogunturi	supergroup	105.34 MB	Dec 12 21:41	1	128 MB	part-r-00010
-rw-r--r--	maheswararaogunturi	supergroup	104.87 MB	Dec 12 21:41	1	128 MB	part-r-00011
-rw-r--r--	maheswararaogunturi	supergroup	107.77 MB	Dec 12 21:41	1	128 MB	part-r-00012
-rw-r--r--	maheswararaogunturi	supergroup	110.86 MB	Dec 12 21:41	1	128 MB	part-r-00013
-rw-r--r--	maheswararaogunturi	supergroup	116.47 MB	Dec 12 21:41	1	128 MB	part-r-00014
-rw-r--r--	maheswararaogunturi	supergroup	103.02 MB	Dec 12 21:41	1	128 MB	part-r-00015
-rw-r--r--	maheswararaogunturi	supergroup	127.26 MB	Dec 12 21:42	1	128 MB	part-r-00016
-rw-r--r--	maheswararaogunturi	supergroup	139.87 MB	Dec 12 21:42	1	128 MB	part-r-00017
-rw-r--r--	maheswararaogunturi	supergroup	140.05 MB	Dec 12 21:42	1	128 MB	part-r-00018
-rw-r--r--	maheswararaogunturi	supergroup	140.14 MB	Dec 12 21:42	1	128 MB	part-r-00019
-rw-r--r--	maheswararaogunturi	supergroup	146.24 MB	Dec 12 21:42	1	128 MB	part-r-00020
-rw-r--r--	maheswararaogunturi	supergroup	137.65 MB	Dec 12 21:42	1	128 MB	part-r-00021



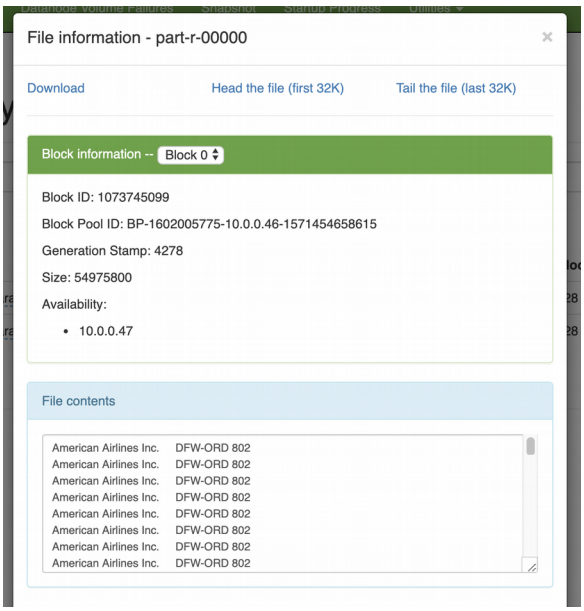
2.8. Join Unique Carrier code with its name (Reduce-side Join Pattern):

Here we are trying to replace each unique carrier code with its name in the data by using Join technique. Each carrier code is joined with carrier name and displayed

with source, destination and distance between them. We give two datasets, one having carrier code and other having carrier name as input and is as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.srcDestDistCarriersJoinName.Driver /joindata /flightData /ProjOutput/innerJoin
```

(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.srcDestDistCarriersJoinName.Driver /joindata /flightData /ProjOutput/innerJoin

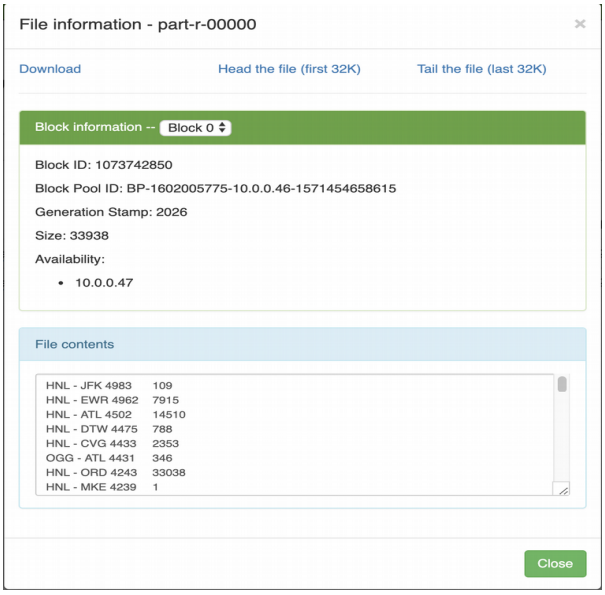


2.9. Trips per route sorted by distance (Secondary sort Algorithm):

Number of trips in each route sorted based on distance travelled in the route is calculated as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.secsortTripsByDistance.Driver /flightData /ProjOutput/TripsByDistance
```

(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.secsortTripsByDistance.Driver /flightData /ProjOutput/TripsByDistance

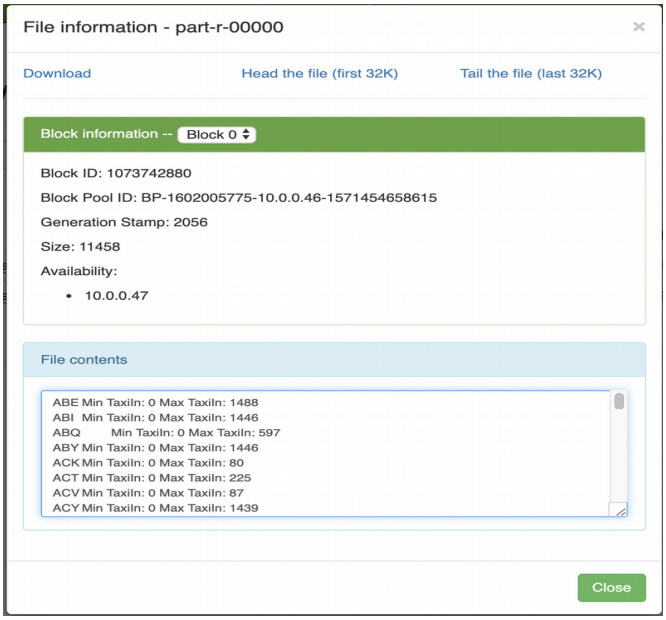


2.10. Calculating minimum and maximum taxin/taxiout time (MinMaxTuple Algorithm):

The maximum and minimum time for taxiin to origin airport is calculated as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.minMaxTaxiInToOrigin.Driver /flightDa
ta /ProjOutput/MinMaxTaxiInToOrigin
```

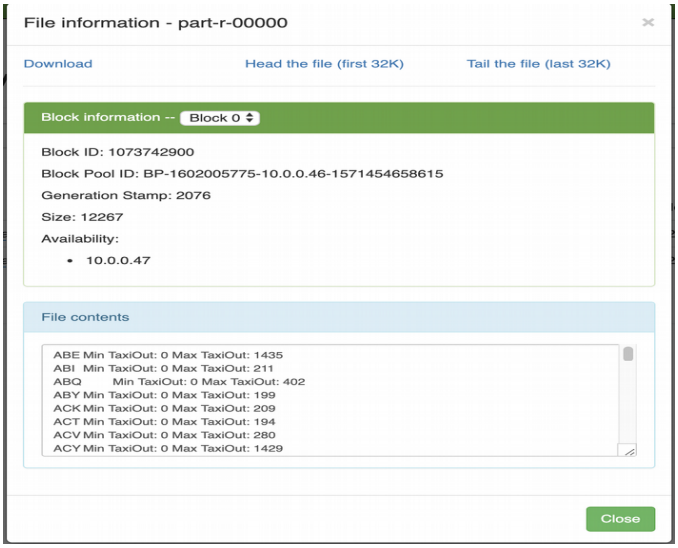
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.minMaxTaxiInToOrigin.Driver /flightData /ProjOutput/MinMaxTaxiInToOrigin



The maximum and minimum time for taxiout for destination airport is calculated as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.minMaxTaxiOutFromDest.Driver /flightD
ata /ProjOutput/MinMaxTaxiOutFromDest
```

(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.minMaxTaxiOutFromDest.Driver /flightData /ProjOutput/MinMaxTaxiOutFromDest

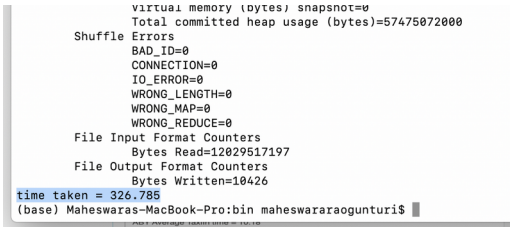


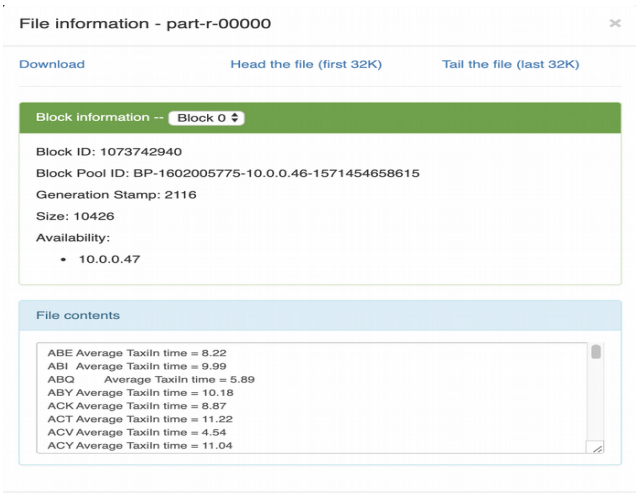
2.11. Average Taxiin/Taxiout time (Average Algorithm):

The average taxiin time is calculated as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.averageTaxiInTimeToOrigin.Driver /flightData /ProjOutput/AvgTaxiInTime
```

(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.averageTaxiInTimeToOrigin.Driver /flightData /ProjOutput/AvgTaxiInTime

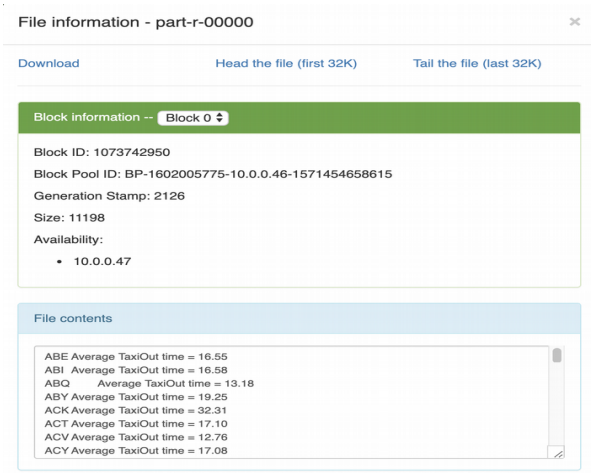




The average taxiout time is calculated as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.averageTaxiOutTimeFromDest.Driver /flightData /ProjOutput/AvgTaxiOutTime
```

(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.averageTaxiOutTimeFromDest.Driver /flightData /ProjOutput/AvgTaxiOutTime



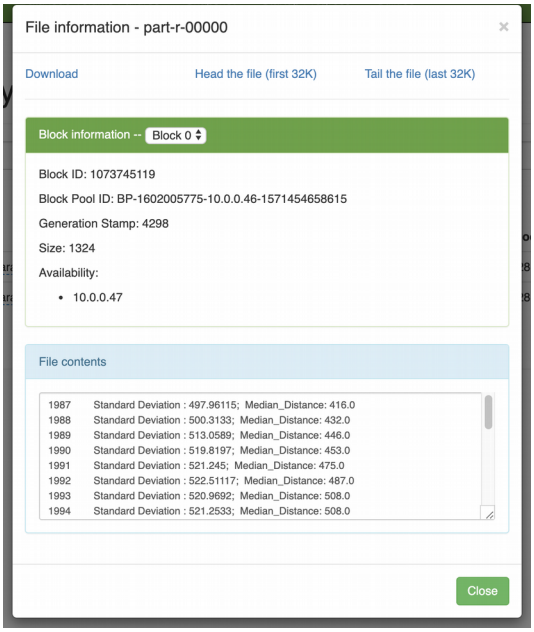
2.12. Median and standard deviation of distance travelled (Median-StdDev Algorithm):

The median and standard deviation of distance travelled over 22 years is analysed and calculated as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.distanceMedianSD.Driver /flightData /ProjOutput/MedianStdDevDistance
```

(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar

/MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.distanceMedianSD.Driver /flightData /ProjOutput/MedianStdDevDistance

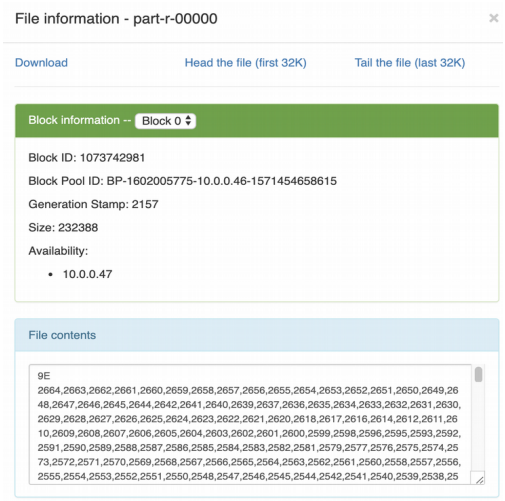


2.13. Flight number indexed by Unique carrier (Inverted Index Algorithm):

Each flight number is indexed by the carrier line code it belongs to and is calculated as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.flightNumsUnderCarrier.Driver /flightData /ProjOutput/InvertedIndexByCarrier
```

(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.flightNumsUnderCarrier.Driver /flightData /ProjOutput/InvertedIndexByCarrier

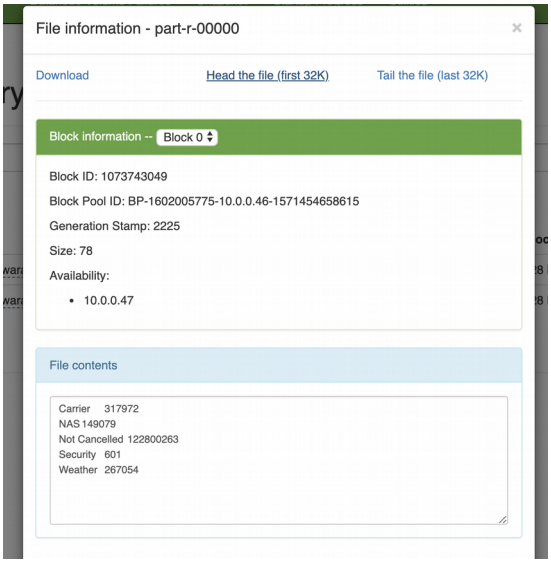


2.14. Flights per each cancellation reason:

The number of flights cancelled per each cancellation reason is calculated as below:

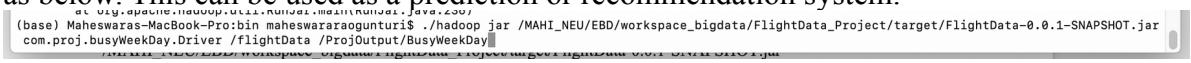
```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.numFlightsCancelledPerReason.Driver /flightData /ProjOutput/CancelCount
```

(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.numFlightsCancelledPerReason.Driver /flightData /ProjOutput/CancelCount

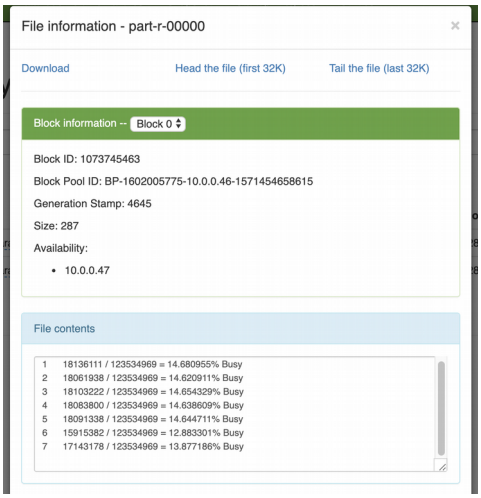


2.15. Busy day of week (Recommendation/Prediction system):

The busiest day of week and percentage of flights travelled on the day is analyzed as below. This can be used as a prediction or recommendation system.



(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./hadoop jar /MAHI_NEU/EBD/workspace_bigdata/FlightData_Project/target/FlightData-0.0.1-SNAPSHOT.jar com.proj.busyWeekDay.Driver /flightData /ProjOutput/BusyWeekDay



The busy day of week observed was 1(i.e Monday), that is immediately after the weekend which is quite possible.

3. Analysis using PIG:

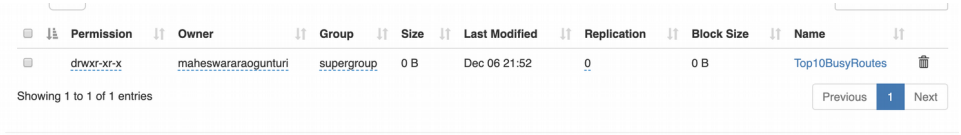
3.1. Top10 busy routes:

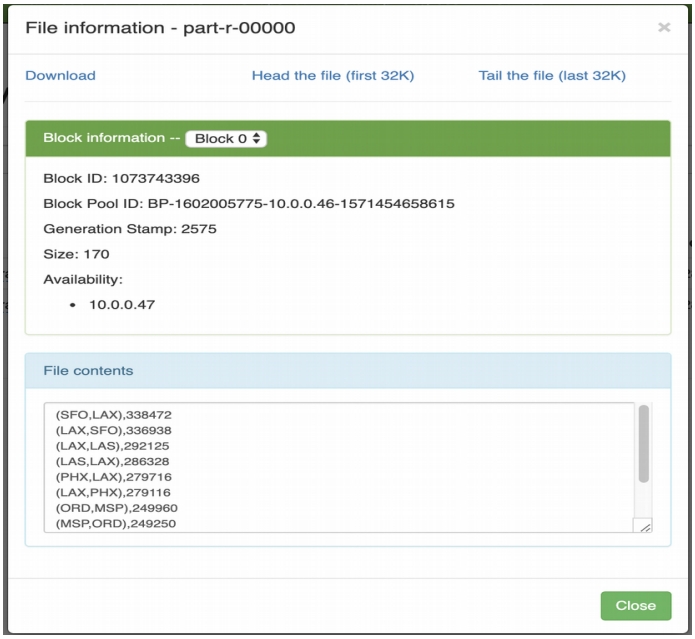
The top 10 busy routes using pig is calculated as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ pwd
/MAHI_NEU/EBD/packages/pig-0.17.0/bin
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./pig -x mapreduce /MAHI_NEU/EBD/Project/PigScripts/Top10BusyRoutes.pig
```

```
2019-12-06 21:52:46,757 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus
=SUCCEEDED. Redirecting to job history server
2019-12-06 21:52:46,769 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2019-12-06 21:52:46,786 [main] INFO  org.apache.pig.Main - Pig script completed in 8 minutes, 53 seconds and 486 milliseconds (533486 ms)
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$
```

The time taken is also noted. As soon as the script is run, the output is stored in HDFS path as below:





The pig script for this is as below:

```
-- TOP10 BUSY ROUTES
data = LOAD '/flightData' USING PigStorage(',') AS
        (year: int, month: int, day: int, dayweek: int,
        deptime: int, crsdeptime: int, arrtime: int, crsarrtime: int,
        carrier: chararray, flightnum: int, tailnum: chararray,
        actelaptime: int, crselaptime: int, airtime: int,
        arrdelay: int, depdelay: int,
        origin: chararray, dest: chararray, dist: int,
        taxiin: int, taxiout: int,
        cancelled: chararray, cancelcode: chararray, diverted: int,
        carrdelay: int, weadelay: int, nasdelay: int, secdelay: int, lateacdelay: int);


trip = FOREACH data GENERATE origin, dest;

grouped = GROUP trip by (origin, dest);

route = FOREACH grouped GENERATE group, COUNT(trip) as nooftrips;

sorted = ORDER route BY nooftrips DESC;

top10 = LIMIT sorted 10;

STORE top10 INTO '/PIGOUTPUT/Top10BusyRoutes' USING PigStorage(',');
~
~
```

3.2. Proportion of early arrived flights:

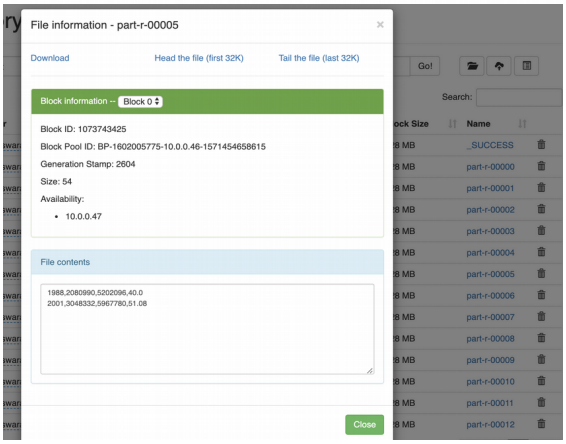
The percentage of flights that arrived early than expected are as below. Also find the pig script:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ pwd
/MAHI_NEU/EBD/packages/pig-0.17.0/bin
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./pig -x mapreduce /MAHI_NEU/EBD/Project/PigScripts/EarlyArrFlightsPercent.pig

=SUCCEEDED. Redirecting to job history server
2019-12-06 23:01:35,795 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning DI
VIDE_BY_ZERO 1 time(s).
2019-12-06 23:01:35,795 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning FI
ELD_DISCARDED_TYPE_CONVERSION_FAILED 2587573 time(s).
2019-12-06 23:01:35,795 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2019-12-06 23:01:35,818 [main] INFO  org.apache.pig.Main - Pig script completed in 8 minutes, 19 seconds and 474 milliseconds (499474 ms)
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$
```

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 06 23:01	0	0 B	EarlyArrFlightsPercent
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 06 21:52	0	0 B	Top10BusyRoutes

Showing 1 to 2 of 2 entries

Previous 1 Next



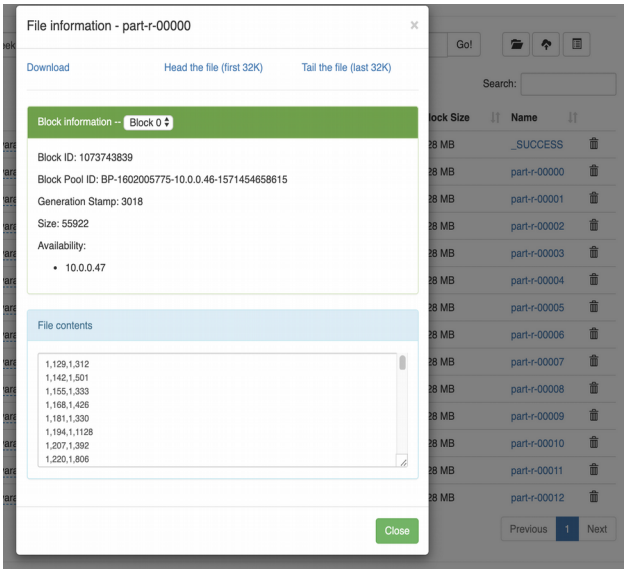
```
-- Percentage of flights that arrived earlier than expected per year
data = LOAD '/flightData' USING PigStorage(',') AS
        (year: int, month: int, day: int, dayweek: int,
        deptime: int, crsdeptime: int, arrtime: int, crsarrtime: int,
        carrier: chararray, flightnum: int, tailnum: chararray,
        actelaptime: int, crselaptime: int, airtime: int,
        arrdelay: int, depdelay: int,
        origin: chararray, dest: chararray, dist: int,
        taxiin: int, taxiout: int,
        cancelled: chararray, cancelcode: chararray, diverted: int,
        carrdelay: int, weadelay: int, nasdelay: int, secdelay: int, lateacdelay: int);

flights = FOREACH data GENERATE year, arrdelay;

grouped = GROUP flights BY year;

calc_perc = FOREACH grouped {
                earlyarr = FILTER flights BY (arrdelay<0); -- negative means arrived early
                GENERATE group, COUNT(earlyarr) AS early, COUNT(flights) AS total,
                                ROUND_TO((float) COUNT(earlyarr)/COUNT(flights) * 100, 2) AS percentage;
            }


STORE calc_perc INTO '/PIGOUTPUT/EarlyArrFlightsPercent' USING PigStorage(',');
~
~
```

3.3. Minimum Maximum departure delay:

The minimum and maximum departure delay time for each flight in month is analyzed for span of 22 years as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ pwd
/MAHI_NEU/EBD/packages/pig-0.17.0/bin
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./pig -x mapreduce /MAHI_NEU/EBD/Project/PigScripts/MinMaxDelayPerDay.pig

2019-12-07 13:11:57,513 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning FI
ELD_DISCARDED_TYPE_CONVERSION_FAILED 2302202 time(s).
2019-12-07 13:11:57,513 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2019-12-07 13:11:57,531 [main] INFO  org.apache.pig.Main - Pig script completed in 8 minutes, 14 seconds and 444 milliseconds (494444 ms)
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$
```

drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 07 11:23	0	0 B	FamousCarriers	
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 07 13:11	0	0 B	MinMaxDelayPerDayWeek	
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 06 21:52	0	0 B	Top10BusyRoutes	



```
-- Min Max departure delay time for each flight in month analysed for span of 22 years
data = LOAD '/flightData' USING PigStorage(',') AS
        (year: int, month: int, day: int, dayweek: int,
        deptime: int, crsdeptime: int, arrtime: int, crsarrtime: int,
        carrier: chararray, flightnum: int, tailnum: chararray,
        actelaptime: int, crselaptime: int, airtime: int,
        arrdelay: int, depdelay: int,
        origin: chararray, dest: chararray, dist: int,
        taxiin: int, taxiout: int,
        cancelled: chararray, cancelcode: chararray, diverted: int,
        carrdelay: int, weadelay: int, nasdelay: int, secdelay: int, lateacdelay: int);

flights = FOREACH data GENERATE dayweek, flightnum, depdelay;

delayedflights = FILTER flights BY (depdelay>0);

grouped = GROUP delayedflights by (dayweek,flightnum);

minmax = FOREACH grouped GENERATE FLATTEN(group), MIN(delayedflights.depdelay), MAX(delayedflights.depdelay);

STORE minmax INTO '/PIGOUTPUT/MinMaxDelayPerDayWeek' USING PigStorage(',');
~
~
```

3.4. Famous Carriers:

The famous carriers analyzed over span of 22 years is as below:

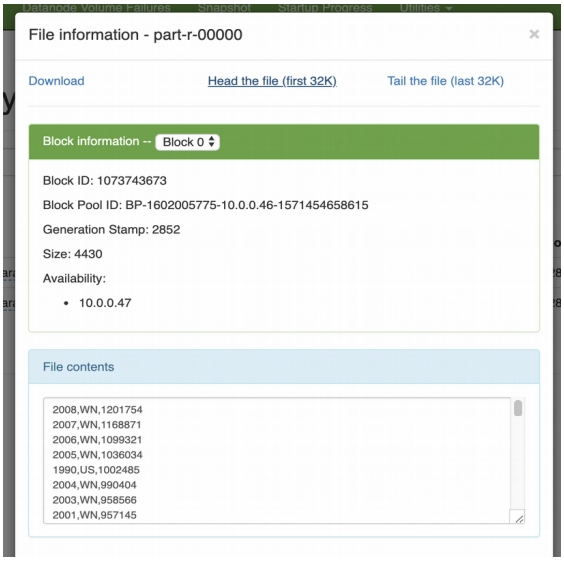
```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ pwd
/MAHI_NEU/EBD/packages/pig-0.17.0/bin
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./pig -x mapreduce /MAHI_NEU/EBD/Project/PigScripts/FamousCarriers.pig
```

```
=SUCCEEDED. Redirecting to job history server
2019-12-07 11:23:21,943 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning FI
ELD_DISCARDED_TYPE_CONVERSION_FAILED 22 time(s).
2019-12-07 11:23:21,944 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2019-12-07 11:23:21,968 [main] INFO  org.apache.pig.Main - Pig script completed in 8 minutes, 15 seconds and 820 milliseconds (495820 ms)
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$
```

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 07 00:34	0	0 B	AvgDelayPerDay
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 06 23:01	0	0 B	EarlyArrFlightsPercent
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 07 11:23	0	0 B	FamousCarriers
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 06 21:52	0	0 B	Top10BusyRoutes

Showing 1 to 4 of 4 entries

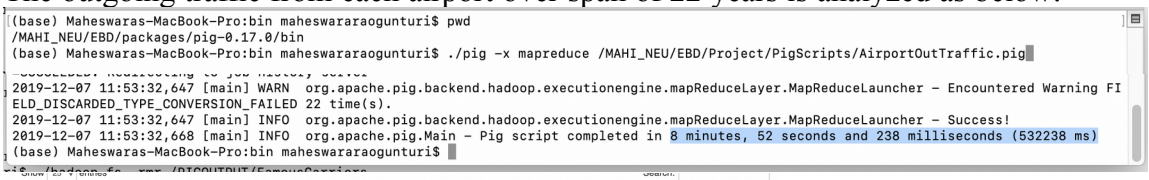
Previous 1 Next

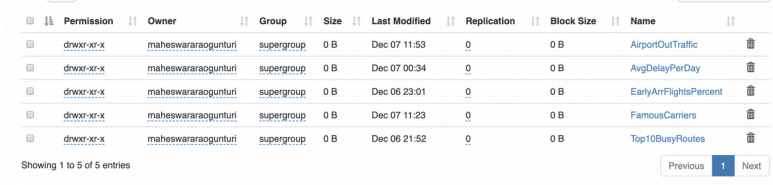


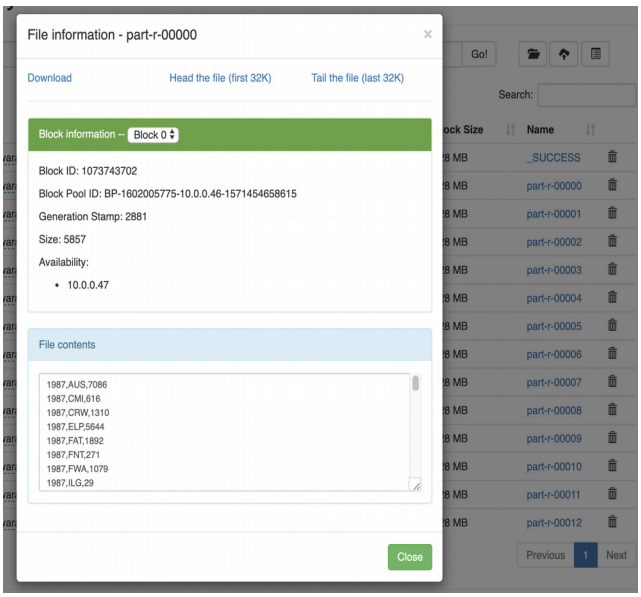
```
-- Famous Carriers analyzed over span of 22 years
data = LOAD '/flightData' USING PigStorage(',') AS
        (year: int, month: int, day: int, dayweek: int,
        deptime: int, crsdeptime: int, arrtime: int, crsarrtime: int,
        carrier: chararray, flightnum: int, tailnum: chararray,
        actelaptime: int, crselaptime: int, airtime: int,
        arrdelay: int, depdelay: int,
        origin: chararray, dest: chararray, dist: int,
        taxiin: int, taxiout: int,
        cancelled: chararray, cancelcode: chararray, diverted: int,
        carrdelay: int, weadelay: int, nasdelay: int, secdelay: int, lateacdelay: int);

flights = FOREACH data GENERATE year, carrier;

grouped = GROUP flights by (year,carrier);

fam = FOREACH grouped GENERATE FLATTEN(group), COUNT(flights) as count;

sortedfam = ORDER fam BY count DESC;

STORE sortedfam INTO '/PIGOUTPUT/FamousCarriers' USING PigStorage(',');
~
~
~
```

3.5. Outgoing Traffic from Airport:

The outgoing traffic from each airport over span of 22 years is analyzed as below:







```
-- Analyzing outgoing traffic from an airport per day of week
data = LOAD '/flightData' USING PigStorage(',') AS
        (year: int, month: int, day: int, dayweek: int,
        deptime: int, crsdeptime: int, arrtime: int, crsarrtime: int,
        carrier: chararray, flightnum: int, tailnum: chararray,
        actelaptime: int, crselaptime: int, airtime: int,
        arrdelay: int, depdelay: int,
        origin: chararray, dest: chararray, dist: int,
        taxiin: int, taxiout: int,
        cancelled: chararray, cancelcode: chararray, diverted: int,
        carrdelay: int, weadelay: int, nasdelay: int, secdelay: int, lateacdelay: int);

flights = FOREACH data GENERATE year, dest;

grouped = GROUP flights by (year, dest);

traff = FOREACH grouped GENERATE FLATTEN(group), COUNT(flights) as count;

STORE traff INTO '/PIGOUTPUT/AirportOutTraffic' USING PigStorage(',');
~
~
~
```

3.6. Average taxiin time:

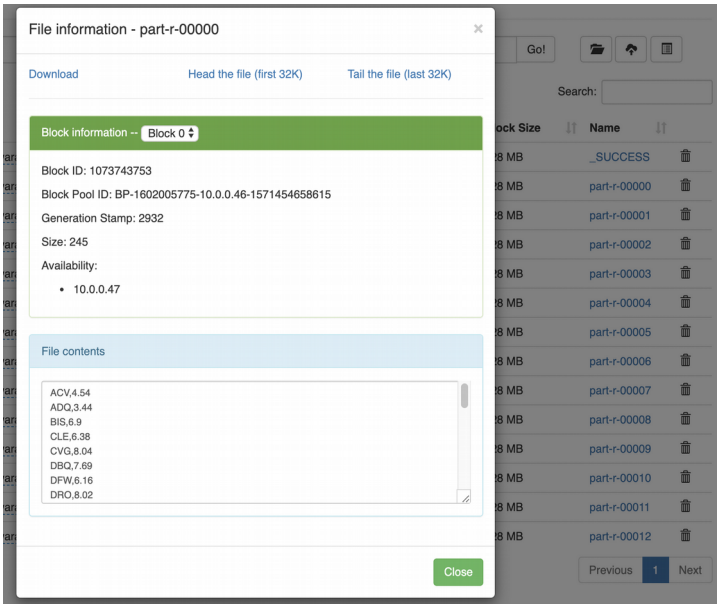
The average taxiin time using pig is calculated as below:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ pwd
/MAHI_NEU/EBD/packages/pig-0.17.0/bin
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ ./pig -x mapreduce /MAHI_NEU/EBD/Project/PigScripts/AvgTaxiinTime.pig
2019-12-07 12:20:07,709 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning FI
ELD_DISCARDED_TYPE_CONVERSION_FAILED 37397317 time(s).
2019-12-07 12:20:07,710 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2019-12-07 12:20:07,728 [main] INFO  org.apache.pig.Main - Pig script completed in 8 minutes, 38 seconds and 890 milliseconds (518890 ms)
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$
```

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 07 00:34	0	0 B	AvgDelayPerDay
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 07 12:20	0	0 B	AvgTaxiinTime
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 06 23:01	0	0 B	EarlyArrFlightsPercent
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 07 11:23	0	0 B	FamousCarriers
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 06 21:52	0	0 B	Top10BusyRoutes

Showing 1 to 5 of 5 entries

Previous 1 Next



```
-- Avg Taxiin time to airport
data = LOAD '/flightData' USING PigStorage(',') AS
        (year: int, month: int, day: int, dayweek: int,
        deptime: int, crsdeptime: int, arrtime: int, crsarrtime: int,
        carrier: chararray, flightnum: int, tailnum: chararray,
        actelaptime: int, crselaptime: int, airtime: int,
        arrdelay: int, depdelay: int,
        origin: chararray, dest: chararray, dist: int,
        taxiin: int, taxiout: int,
        cancelled: chararray, cancelcode: chararray, diverted: int,
        carrdelay: int, weadelay: int, nasdelay: int, secdelay: int, lateacdelay: int);

flights = FOREACH data GENERATE origin, taxiin;

grouped = GROUP flights by origin;

average = FOREACH grouped GENERATE FLATTEN(group), ROUND_TO(AVG(flights.taxiin),2);

STORE average INTO '/PIGOUTPUT/AvgTaxiinTime' USING PigStorage(',');
~
~
```

4. Analysis using HIVE

4.1. Creating Schema:

To load the data to Hive, we have to first create database(schema), table and then load the data into it as below:

CREATE DATABASE AirlineDB;

```
hive>
    > CREATE DATABASE AirlineDB;
```

USE AirlineDB;

```
2019-12-07 15:45:31,067 INFO  [
hive> USE AirlineDB;
2019-12-07 15:45:43,828 INFO  [
```

CREATE EXTERNAL TABLE flightData(Year INT, Month INT, DayofMonth INT, DayOfWeek INT, DepTime INT, CRSDepTime INT, ArrTime INT, CRSArrTime INT, UniqueCarrier String, FlightNum INT, TailNum String, ActualElapsedTime INT, CRSElapsedTime INT, AirTime INT, ArrDelay INT, DepDelay INT, Origin String, Dest String, Distance INT, TaxiIn INT, TaxiOut INT, Cancelled INT, CancellationCode String, Diverted String, CarrierDelay INT, WeatherDelay INT, NASDelay INT, SecurityDelay INT, LateAircraftDelay INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

```
hive>
    > CREATE EXTERNAL TABLE flightData(Year INT, Month INT, DayofMonth INT, DayOfWeek INT, DepTime INT, CRSDepTime INT, ArrTime INT, CRSArrTime INT, UniqueCarrier String, FlightNum INT, Tai
lNum String, ActualElapsedTime INT, CRSElapsedTime INT, AirTime INT, ArrDelay INT, DepDelay INT, Origin String, Dest String, Distance INT, TaxiIn INT, TaxiOut INT, Cancelled INT, Cancellati
onCode String, Diverted String, CarrierDelay INT, WeatherDelay INT, NASDelay INT, SecurityDelay INT, LateAircraftDelay INT ) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
2019-12-07 15:48:31,310 INFO  [main] conf.HiveConf: Using the default value passed in for log id: 28f778b8-8f82-44bf-8ebb-d666be1c91f8
```

LOAD DATA INPATH '/flightData' OVERWRITE INTO TABLE flightData;

```
2019-12-07 15:48:37,627 INFO  [28f778b8-8f82-44bf-8ebb-d666be1c91f8 main] conf.HiveConf: Using the d
2019-12-07 15:48:37,627 INFO  [28f778b8-8f82-44bf-8ebb-d666be1c91f8 main] session.SessionState: Rese
hive>
    > LOAD DATA INPATH '/flightData' OVERWRITE INTO TABLE flightData;
2019-12-07 15:49:31,800 INFO  [main] conf.HiveConf: Using the default value passed in for log id: 28
```

4.2. Total number of flight trips from 1987 to 2008:

The command and output is as below:

select count(*) from flightData where AirTime > 500;

```
2019-12-07 15:49:32,354 INFO  [28f778b8-8f82-44bf-8ebb-d666be1c91f8 main] session.SessionState
hive> select count(*) from flightData where AirTime > 500;
2019-12-07 15:50:38,635 INFO  [main] conf.HiveConf: Using the default value passed in for log
```

```
2019-12-07 15:53:12,360 INFO  [28f778b8-8f82-44bf-8ebb-d666be1c91f8 main] mapred.File
2019-12-07 15:53:12,365 INFO  [28f778b8-8f82-44bf-8ebb-d666be1c91f8 main] sasl.SaslDa
2019-12-07 15:53:12,382 INFO  [28f778b8-8f82-44bf-8ebb-d666be1c91f8 main] exec.ListSi
30711
Time taken: 153.705 seconds, Fetched: 1 row(s)
2019-12-07 15:53:12,385 INFO  [28f778b8-8f82-44bf-8ebb-d666be1c91f8 main] CliDriver:
2019-12-07 15:53:12,385 INFO  [28f778b8-8f82-44bf-8ebb-d666be1c91f8 main] conf.HiveCo
2019-12-07 15:53:12,385 INFO  [28f778b8-8f82-44bf-8ebb-d666be1c91f8 main] session.Ses
hive>
```

4.3. Early arrived flights:

Flights that started late but arrived early are analyzed as below:

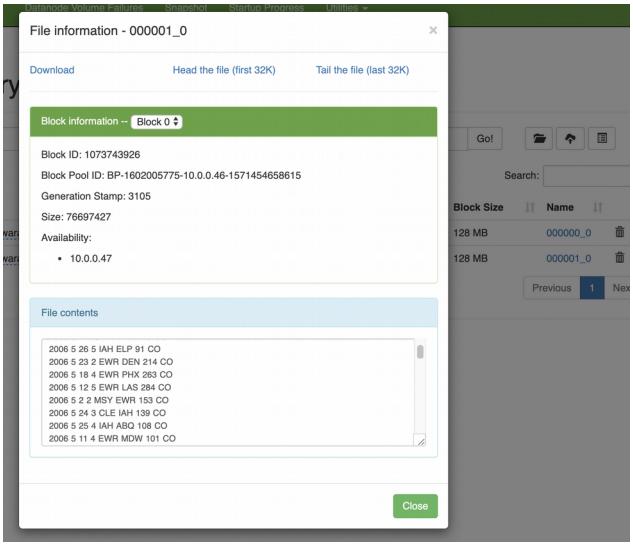
INSERT OVERWRITE DIRECTORY '/HIVEOUTPUT/ArrDepAnalysis'
SELECT
Year,Month,DayofMonth,DayOfWeek,Origin,Dest,AirTime,UniqueCarrier FROM flightData where DepTime>CRSDepTime and ArrTime<=CRSArrTime;

```
hive>
    > INSERT OVERWRITE DIRECTORY '/HIVEOUTPUT/ArrDepAnalysis' SELECT Year,Month,DayofMonth,DayOfWeek,Origin,Dest,AirTime,Un
iqueCarrier FROM flightData where DepTime>CRSDepTime and ArrTime<=CRSArrTime;
2019-12-07 16:07:38,295 INFO  [main] conf.HiveConf: Using the default value passed in for log id: 28f778b8-8f82-44bf-8ebb-d
```

The output is stored in HDFS path as below:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 07 16:11	0	0 B	ArrDepAnalysis

Showing 1 to 1 of 1 entries Previous 1 Next

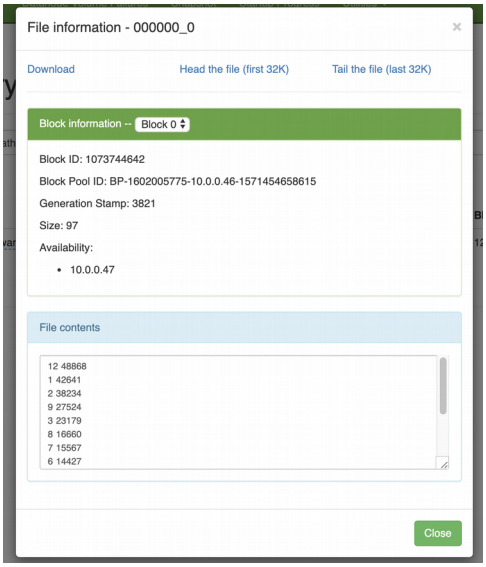


4.4. Month analysis of cancellation due to bad weather:

The month wise analysis with number of flights cancelled with reason bad weather is as follows:

```
hive> INSERT OVERWRITE DIRECTORY '/HIVEOUTPUT/MonthCancReasonWeather' SELECT month, COUNT(Cancelled) as count FROM flightData W
HERE Cancelled=1 AND CancellatioCode='B' GROUP BY month ORDER BY count;
```

drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 07 16:11	0	0 B	ArrDepAnalysis
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 07 18:39	0	0 B	CarrierFlightsCanc
drwxr-xr-x	maheswararaogunturi	supergroup	0 B	Dec 07 19:03	0	0 B	MonthCancReasonWeather

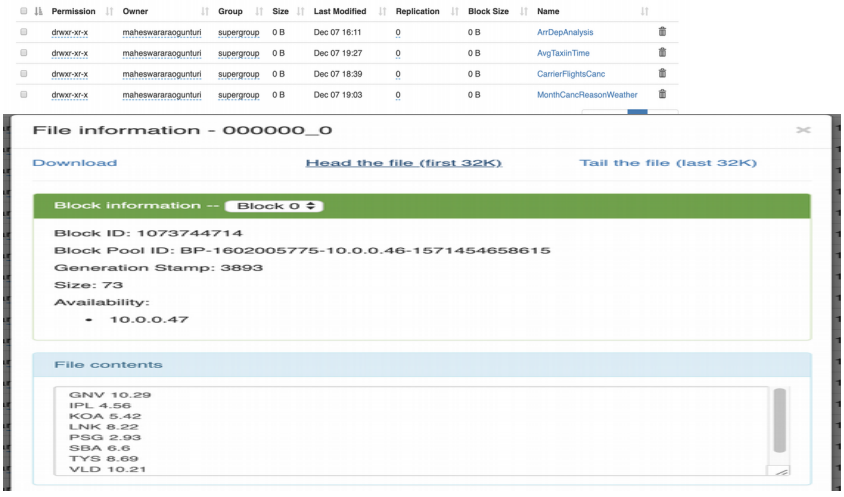


4.5. Average Taxiin time:

The average taxiin time to airport is calculated as below:

INSERT OVERWRITE DIRECTORY '/HIVEOUTPUT/AvgTaxiinTime' SELECT Origin, ROUND(AVG(TaxiIn),2) as avg FROM flightData GROUP BY Origin;

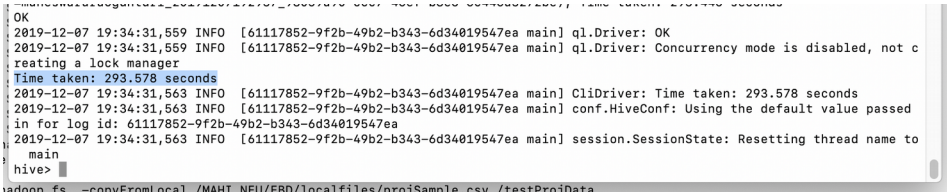
```
hive> INSERT OVERWRITE DIRECTORY '/HIVEOUTPUT/AvgTaxiinTime' SELECT Origin, ROUND(AVG(TaxiIn),2) as avg FROM flightData
 GROUP BY Origin;
OK
2019-12-07 19:27:09,048 INFO  [61117852-9f2b-49b2-b343-6d34019547ea main] ql.Driver: OK
2019-12-07 19:27:09,048 INFO  [61117852-9f2b-49b2-b343-6d34019547ea main] ql.Driver: Concurrency mode is disabled, not c
reating a lock manager
Time taken: 291.807 seconds
2019-12-07 19:27:09,068 INFO  [61117852-9f2b-49b2-b343-6d34019547ea main] CliDriver: Time taken: 291.807 seconds
2019-12-07 19:27:09,068 INFO  [61117852-9f2b-49b2-b343-6d34019547ea main] conf.HiveConf: Using the default value passed
in for log id: 61117852-9f2b-49b2-b343-6d34019547ea
2019-12-07 19:27:09,068 INFO  [61117852-9f2b-49b2-b343-6d34019547ea main] session.SessionState: Resetting thread name to
  main
hive>
```

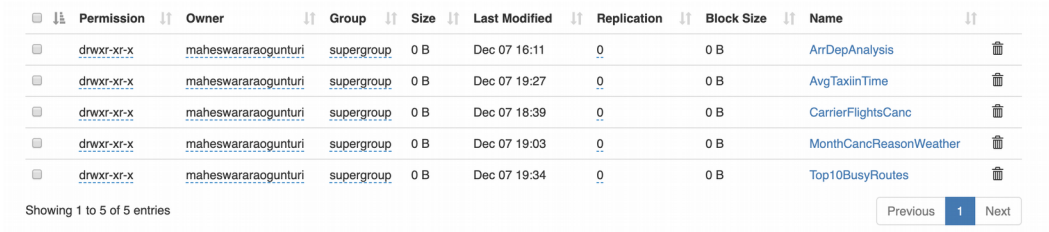


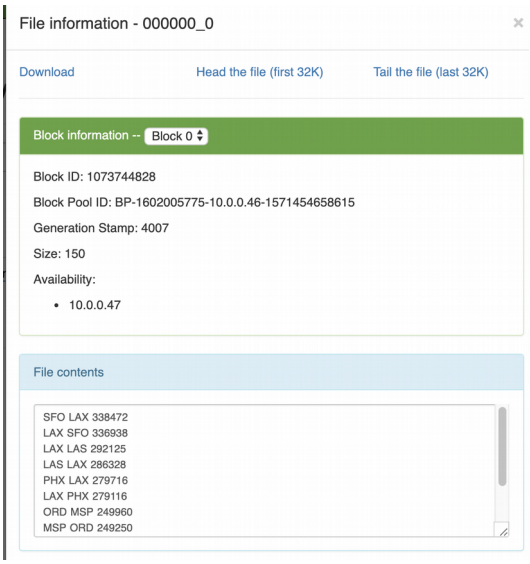
4.6. Top 10 Busy routes:

The top 10 busy routes are calculated as below:

```
hive> INSERT OVERWRITE DIRECTORY '/HIVEOUTPUT/Top10BusyRoutes' SELECT Origin, Dest, count(*) as count FROM flightData GR
OUP BY Origin, Dest ORDER BY count DESC LIMIT 10;
```







5. Analysis using MongoDB:

5.1. Import data:

First we have to import the data to MongoDB. This is done easily by running a script file:

```
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$ /MAHI_NEU/EBD/Project/flightDataImport.sh
Processing /MAHI_NEU/EBD/Project/data/1987.csv file...
/MAHI_NEU/EBD/Project/flightDataImport.sh: line 7: MONGODB_HOME: command not found
-rw-r--r--  1 maheswararaogunturi  staff  127162942 Nov 29 18:51 /MAHI_NEU/EBD/Project/data/1987.csv
2019-12-07T23:33:05.982-0500    connected to: mongodb://localhost/
2019-12-07T23:33:08.985-0500    [##......................] flightdb.flight      13.6MB/121MB (11.2%)
2019-12-07T23:33:11.985-0500    [#####...................] flightdb.flight      27.6MB/121MB (22.7%)
2019-12-07T23:33:14.987-0500    [########................] flightdb.flight      41.3MB/121MB (34.1%)
2019-12-07T23:33:17.983-0500    [###########.............] flightdb.flight      54.9MB/121MB (45.3%)
2019-12-07T23:33:20.983-0500    [##############..........] flightdb.flight      68.5MB/121MB (56.5%)
2019-12-07T23:33:23.985-0500    [#################.......] flightdb.flight      82.2MB/121MB (67.7%)
2019-12-07T23:33:26.982-0500    [####################....] flightdb.flight      95.6MB/121MB (78.8%)
```

```
2019-12-08T00:27:32.694-0500    [######################...] flightdb.flight      598MB/657MB (91.0%)
2019-12-08T00:27:35.692-0500    [######################..] flightdb.flight      612MB/657MB (93.1%)
2019-12-08T00:27:38.693-0500    [######################..] flightdb.flight      626MB/657MB (95.2%)
2019-12-08T00:27:41.691-0500    [######################.] flightdb.flight      640MB/657MB (97.3%)
2019-12-08T00:27:44.694-0500    [######################.] flightdb.flight      654MB/657MB (99.5%)
2019-12-08T00:27:45.351-0500    [########################] flightdb.flight      657MB/657MB (100.0%)
2019-12-08T00:27:45.351-0500    7009728 document(s) imported successfully. 0 document(s) failed to import.
(base) Maheswaras-MacBook-Pro:bin maheswararaogunturi$
```

```
> show dbs
admin      0.000GB
config     0.000GB
flightdb  10.862GB
local      0.000GB
> 
> use flightdb
switched to db flightdb
> show collections
flight
> 
```

All the data is stored in flightdb database and in flight collection in the form of documents:


```
> db.flight.findOne()
{
        "_id" : ObjectId("5dec7d02ff94dd191dccc034"),
        "Year" : 1987,
        "Month" : 10,
        "DayofMonth" : 18,
        "DayOfWeek" : 7,
        "DepTime" : 729,
        "CRSDepTime" : 730,
        "ArrTime" : 847,
        "CRSArrTime" : 849,
        "UniqueCarrier" : "PS",
        "FlightNum" : 1451,
        "TailNum" : "NA",
        "ActualElapsedTime" : 78,
        "CRSElapsedTime" : 79,
        "AirTime" : "NA",
        "ArrDelay" : -2,
        "DepDelay" : -1,
        "Origin" : "SAN",
        "Dest" : "SFO",
        "Distance" : 447,
        "TaxiIn" : "NA",
        "TaxiOut" : "NA",
        "Cancelled" : 0,
        "CancellationCode" : "NA",
        "Diverted" : 0,
        "CarrierDelay" : "NA",
        "WeatherDelay" : "NA",
        "NASDelay" : "NA",
        "SecurityDelay" : "NA",
        "LateAircraftDelay" : "NA"
}
> 
```


5.2. Trips made in each route:

The number of trips made in each route is calculated using mapreduce in MongoDB and is as follows and the time taken is also noted:

```
> mapperRouteTrips
function() {
  var key = {Route: {Origin:this.Origin,Dest:this.Dest}};
  var value = {Count:1};
  emit(key,value);
 }
> 
```

```
> reducerRouteTrips
function(key,values){
 var Count  = 0;
 values.forEach(function(value){
                                Count += value['Count'];
                               }
             );
 return {Count:Count};
}
> 
```

```
> db.flight.mapReduce(mapperRouteTrips, reducerRouteTrips, {out:"NumRouteTrips"})
{
        "result" : "NumRouteTrips",
        "timeMillis" : 4854918,
        "counts" : {
                "input" : 123534969,
                "emit" : 123534969,
                "reduce" : 6063263,
                "output" : 8607
        },
        "ok" : 1
}
> 
```

The output is saved as a new collection and is as below:

```
> show collections
NumRouteTrips
flight
> 
> db.NumRouteTrips.find().limit(10)
{ "_id" : { "Route" : { "Origin" : "ABE", "Dest" : "ALB" } }, "value" : { "Count" : 2 } }
{ "_id" : { "Route" : { "Origin" : "ABE", "Dest" : "ATL" } }, "value" : { "Count" : 16541 } }
{ "_id" : { "Route" : { "Origin" : "ABE", "Dest" : "AVP" } }, "value" : { "Count" : 1627 } }
{ "_id" : { "Route" : { "Origin" : "ABE", "Dest" : "AZO" } }, "value" : { "Count" : 1 } }
{ "_id" : { "Route" : { "Origin" : "ABE", "Dest" : "BDL" } }, "value" : { "Count" : 1 } }
{ "_id" : { "Route" : { "Origin" : "ABE", "Dest" : "BHM" } }, "value" : { "Count" : 1 } }
{ "_id" : { "Route" : { "Origin" : "ABE", "Dest" : "BWI" } }, "value" : { "Count" : 2559 } }
{ "_id" : { "Route" : { "Origin" : "ABE", "Dest" : "CLE" } }, "value" : { "Count" : 5860 } }
{ "_id" : { "Route" : { "Origin" : "ABE", "Dest" : "CLT" } }, "value" : { "Count" : 7261 } }
{ "_id" : { "Route" : { "Origin" : "ABE", "Dest" : "CVG" } }, "value" : { "Count" : 6881 } }
> 
```

5.3. Average Taxiin time:

The average taxiin time is calculated using MongoDB mapreduce as follows:

```
> mapperAvgTaxiIn
function() {
 var value = {count:1,
              TaxiIn:parseInt(this.TaxiIn)};
 if(this.TaxiIn != "NA") emit({Airport:this.Origin}, value);
}
>
> reducerAvgTaxiIn
function(key,values) {
 reducedVal = {count:0, TaxiIn:0.0};
 for(var idx=0; idx<values.length; idx++) {
           reducedVal.count += values[idx].count;
           reducedVal.TaxiIn += values[idx].TaxiIn;
 }
 return reducedVal;
}
>
> finalizeAvgTaxiIn
function(key,reducedVal) {
  reducedVal.avg = reducedVal.TaxiIn/reducedVal.count;
  return reducedVal;
}
> 
```

```
> db.flight.mapReduce(mapperAvgTaxiIn, reducerAvgTaxiIn, {out: "AverageTaxiIn", finalize: finalizeAvgTaxiIn})
{
        "result" : "AverageTaxiIn",
        "timeMillis" : 1342106,
        "counts" : {
                "input" : 123534969,
                "emit" : 86137674,
                "reduce" : 3256576,
                "output" : 335
        },
        "ok" : 1
}
```

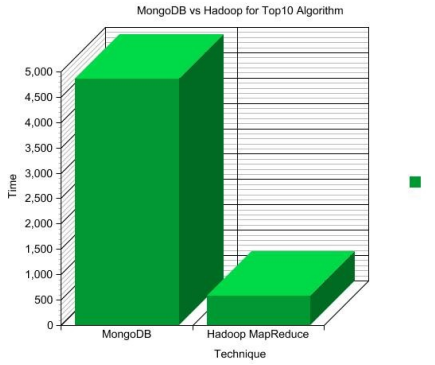
```
> show collections
AverageTaxiIn
NumRouteTrips
flight
> db.AverageTaxiIn.find().limit(10)
{ "_id" : { "Airport" : "ABE" }, "value" : { "count" : 78423, "TaxiIn" : 644254, "avg" : 8.215115463575737 } }
{ "_id" : { "Airport" : "ABI" }, "value" : { "count" : 21667, "TaxiIn" : 216393, "avg" : 9.987215581298749 } }
{ "_id" : { "Airport" : "ABQ" }, "value" : { "count" : 524829, "TaxiIn" : 3093375, "avg" : 5.8940626375448 } }
{ "_id" : { "Airport" : "ABY" }, "value" : { "count" : 8018, "TaxiIn" : 81641, "avg" : 10.182215016213519 } }
{ "_id" : { "Airport" : "ACK" }, "value" : { "count" : 1733, "TaxiIn" : 15368, "avg" : 8.867859203693017 } }
{ "_id" : { "Airport" : "ACT" }, "value" : { "count" : 21007, "TaxiIn" : 235731, "avg" : 11.221545199219308 } }
{ "_id" : { "Airport" : "ACV" }, "value" : { "count" : 23277, "TaxiIn" : 105581, "avg" : 4.535850839884865 } }
{ "_id" : { "Airport" : "ACY" }, "value" : { "count" : 3878, "TaxiIn" : 42827, "avg" : 11.043579164517793 } }
{ "_id" : { "Airport" : "ADK" }, "value" : { "count" : 579, "TaxiIn" : 1873, "avg" : 3.234887737478411 } }
{ "_id" : { "Airport" : "ADQ" }, "value" : { "count" : 9461, "TaxiIn" : 32525, "avg" : 3.4377972730155375 } }
> 
```

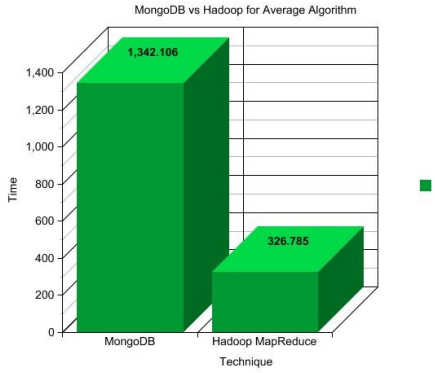
6. Performance Analysis

As we have performed analysis on various frameworks, we can do a performance analysis.
We can compare each of the framework with two of the algorithms (top 10 ang average taxiin). The times taken for these algorithms by different frameworks are as below:

Algorithm	Hadoop MapReduce(sec)	PIG(sec)	HIVE(sec)	MongoDB(sec)
Top10	575.252	533	293.578	4854.918
Average	326.785	518	291.807	1342.106

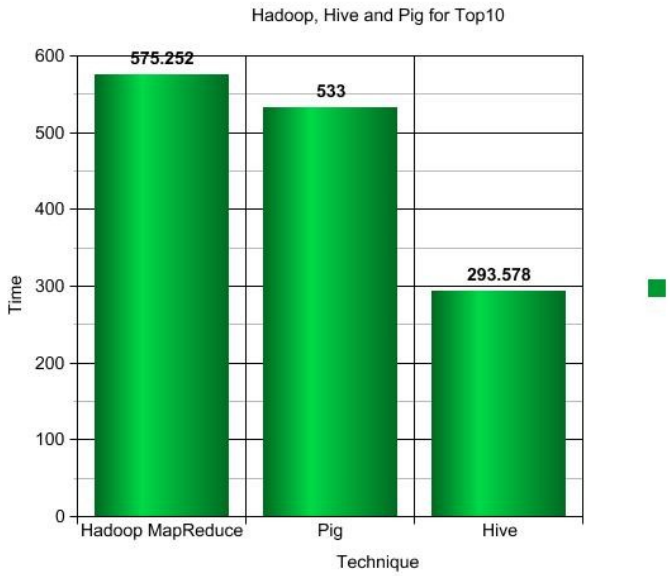
Now let's see their difference graphically.
Firstly, we will compare MongoDB and Hadoop MapReduce frameworks for two of the algorithms we used.

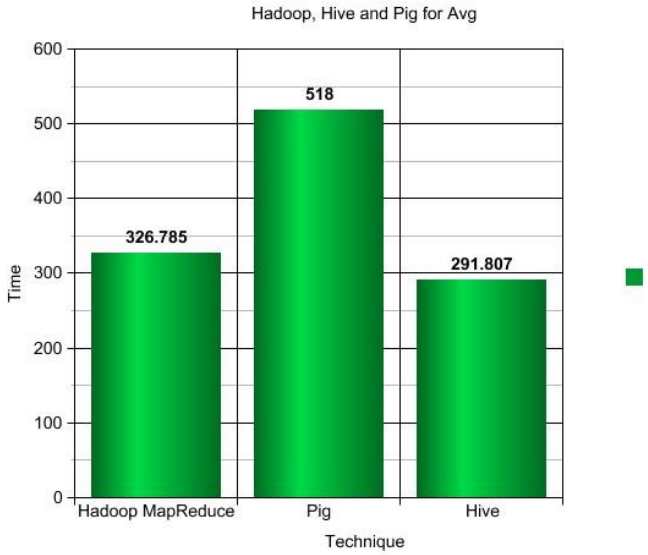




There is a huge difference between the times taken by MongoDB and Hadoop MapReduce techniques. It is obvious that Hadoop is much faster than MongoDB is analysis.

Now similarly, let's compare the times taken for Hadoop MapReduce, Pig and hive for these 2 algorithms graphically:





Among these three, Hive gives the best performance.

7. References

- All Year Data: http://stat-computing.org/dataexpo/2009/the-data.html
- Carrier csv: http://stat-computing.org/dataexpo/2009/supplemental-data.html
- For graphs: https://nces.ed.gov/nceskids/createagraph/default.aspx
- All Lecture slides

8. Appendix

8.1. Total number of flight travels from 1987 to 2008 in USA:

```
package com.proj.numOfFlightTravels;

import java.io.IOException;

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

public class TotalCountMapper extends
Mapper<LongWritable,Text,NullWritable,IntWritable>  {

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

            if(key.get()>0) {
                    context.write(NullWritable.get(), new IntWritable(1));
            }
    }
}



package com.proj.numOfFlightTravels;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.mapreduce.Reducer;
```

```
public class TotalCountReducer extends
Reducer<NullWritable,IntWritable,NullWritable,IntWritable> {
    IntWritable count = new IntWritable();
    int sum = 0;
    @Override
    public void reduce(NullWritable key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {
            for(IntWritable val : values) {
                    sum += val.get();
            }
            count.set(sum);
            context.write(key, count);
    }

}


package com.proj.numOfFlightTravels;

import java.io.IOException;

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.io.NullWritable;
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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class Driver {

    public static void main(String[] args) {
    Configuration conf = new Configuration();
    try {
                    Job job = Job.getInstance(conf, "Total number of flight
journeys from 1987 to 2008");
                    job.setJarByClass(Driver.class);


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

                    //set inputformat and outputformat
                    job.setInputFormatClass(TextInputFormat.class);
                    job.setOutputFormatClass(TextOutputFormat.class);
```

```
                        //set the output key and value types
                        job.setOutputKeyClass(NullWritable.class);
                        job.setOutputValueClass(IntWritable.class);

                        //set mappers and reducers classes
                        job.setMapperClass(TotalCountMapper.class);
                        job.setReducerClass(TotalCountReducer.class);

                        //set the input and output args
                        FileInputFormat.addInputPath(job, new Path(args[0]));
                        FileOutputFormat.setOutputPath(job, new Path(args[1]));

                        //deletes if o/p path already exist
                        FileSystem fs = FileSystem.get(conf);
                        fs.delete(new Path(args[1]),true);

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

                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (ClassNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

}
```

8.2. Number of trips in each Route:

```
package com.proj.tripsPerRoute;

import java.io.IOException;

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

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

    Text route = new Text();
    IntWritable one = new IntWritable(1);
```

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

		if(key.get()>0) {
			String line = value.toString();
			String[] tokens = line.split(",");
			route.set(tokens[16] + " - " + tokens[17]);
			context.write(route, one);
		}
	}

}


package com.proj.tripsPerRoute;

import java.io.IOException;

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

public class TripsPerRouteReducer extends
Reducer<Text,IntWritable,Text,IntWritable> {
    IntWritable count = new IntWritable();

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
		int sum = 0;
		for(IntWritable val : values) {
			sum += val.get();
		}
		count.set(sum);
		context.write(key, count);
    }

}


package com.proj.tripsPerRoute;

import java.io.IOException;

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.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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


public class Driver {

    public static void main(String[] args) {
    Configuration conf = new Configuration();
    try {
                        Job job = Job.getInstance(conf, "Total number of trips in each
route from 1987 to 2008");
                        job.setJarByClass(Driver.class);


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

                        //set inputformat and outputformat
                        job.setInputFormatClass(TextInputFormat.class);
                        job.setOutputFormatClass(TextOutputFormat.class);


                        //set the output key and value types
                        job.setOutputKeyClass(Text.class);
                        job.setOutputValueClass(IntWritable.class);

                        //set mappers and reducers classes
                        job.setMapperClass(TripsPerRouteMapper.class);
                        job.setReducerClass(TripsPerRouteReducer.class);

                        //set the input and output args
                        FileInputFormat.addInputPath(job, new Path(args[0]));
                        FileOutputFormat.setOutputPath(job, new Path(args[1]));

                        //deletes if o/p path already exist
                        FileSystem fs = FileSystem.get(conf);
                        fs.delete(new Path(args[1]),true);

                        //System.exit(job.waitForCompletion(true)? 0:1);
                        long start = System.currentTimeMillis();
                        if(job.waitForCompletion(true)) {
                                long end = System.currentTimeMillis();
                            float timeTaken = (end - start)/1000F;
                            System.out.println("time taken =
"+Float.toString(timeTaken));
                                System.exit(0);
                        }else {
                                System.exit(1);
```

```
                }

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
    }

}
```

8.3. Finding distinct Unique Carriers:

```
package com.proj.distinctCarriers;

import java.io.IOException;

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

public class DistinctCarrierMapper extends
Mapper<LongWritable,Text,Text,NullWritable>  {

    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        if(key.get()>0) {
            String line = value.toString();
            String[] tokens = line.split(",");
            String carrier = tokens[8];

            context.write(new Text(carrier), NullWritable.get());
        }
    }

}
```

```
package com.proj.distinctCarriers;

import java.io.IOException;

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

```
public class DistinctCarrierReducer extends
Reducer<Text,NullWritable,Text,NullWritable> {

    public void reduce(Text key, Iterable<NullWritable> values,Context context)
throws IOException, InterruptedException {
            context.write(key, NullWritable.get());
    }

}


package com.proj.distinctCarriers;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


public class Driver {
   public static void main( String[] args )
   {
    Configuration conf = new Configuration();
    try {
                    Job job = Job.getInstance(conf, "Distinct Carriers");
                    job.setJarByClass(Driver.class);


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

                    //set inputformat and outputformat
                    job.setInputFormatClass(TextInputFormat.class);
                    job.setOutputFormatClass(TextOutputFormat.class);


                    //set the output key and value types
                    job.setOutputKeyClass(Text.class);
                    job.setOutputValueClass(NullWritable.class);

                    //set mappers and reducers classes
```

```
                        job.setMapperClass(DistinctCarrierMapper.class);
                        job.setCombinerClass(DistinctCarrierReducer.class);
                        job.setReducerClass(DistinctCarrierReducer.class);

                        //set the input and output args
                        FileInputFormat.addInputPath(job, new Path(args[0]));
                        FileOutputFormat.setOutputPath(job, new Path(args[1]));

                        //deletes if o/p path already exist
                        FileSystem fs = FileSystem.get(conf);
                        fs.delete(new Path(args[1]),true);

                        //System.exit(job.waitForCompletion(true)? 0:1);
                        long start = System.currentTimeMillis();
                        if(job.waitForCompletion(true)) {
                                long end = System.currentTimeMillis();
                            float timeTaken = (end - start)/1000F;
                            System.out.println("time taken =
"+Float.toString(timeTaken));
                                System.exit(0);
                        }else {
                                System.exit(1);
                        }

                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (ClassNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
    }
}
```

8.4. Simple Random Sampling:

```
package com.proj.srSampling;

import java.io.IOException;
import java.util.Random;

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

```
public class SRSMapper extends Mapper<LongWritable,Text,NullWritable,Text>
{

    private Random rands = new Random();
    private Double percentage;

    protected void setup(Context context) {
            String strPercentage =
context.getConfiguration().get("filter.percentage");
            percentage = Double.parseDouble(strPercentage) / 100.0;
    }

    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            if(rands.nextDouble() < percentage) {
                    context.write(NullWritable.get(), value);
            }
    }

}

package com.proj.srSampling;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


public class Driver {
   public static void main( String[] args )
   {
    Configuration conf = new Configuration();
    try {
                    Job job = Job.getInstance(conf, "SRS sampling");
                    job.setJarByClass(Driver.class);

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

                    //set inputformat and outputformat
                    job.setInputFormatClass(TextInputFormat.class);
```

```
                job.setOutputFormatClass(TextOutputFormat.class);


                //set mappers and reducers classes
                job.setMapperClass(SRSMapper.class);
                job.setNumReduceTasks(0);

                job.getConfiguration().set("filter.percentage", "1");
                //set the input and output args
                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));

                //deletes if o/p path already exist
                FileSystem fs = FileSystem.get(conf);
                fs.delete(new Path(args[1]),true);

                //System.exit(job.waitForCompletion(true)? 0:1);
                long start = System.currentTimeMillis();
                if(job.waitForCompletion(true)) {
                        long end = System.currentTimeMillis();
                    float timeTaken = (end - start)/1000F;
                    System.out.println("time taken =
"+Float.toString(timeTaken));
                         System.exit(0);
                }else {
                        System.exit(1);
                }

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
       }
    }
```

8.5. Top 10 Busiest Routes:

```
package com.proj.top10BusyRoutes;

import java.io.IOException;
import java.util.Map;
import java.util.TreeMap;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.mapreduce.Mapper;

public class Top10BusyRoutesMapper extends
Mapper<LongWritable,Text,Text,LongWritable>  {

    private TreeMap<Long,  String> topTenMap ;

    public void setup(Context context) {
            topTenMap = new TreeMap<Long, String>();
    }

    public void map(LongWritable key, Text value, Context context) {
            String line = value.toString();
            String[] tokens = line.split("\\t");
            String route = tokens[0];
            Long tripsCount = Long.parseLong(tokens[1]);
            topTenMap.put(tripsCount, route);

            if(topTenMap.size() > 10) {
                    topTenMap.remove(topTenMap.firstKey());
            }
    }

    public void cleanup(Context context) throws IOException,
InterruptedException {
            for(Map.Entry<Long, String> m : topTenMap.entrySet()) {
                    context.write(new Text(m.getValue()), new
LongWritable(m.getKey()));
            }
    }
}

package com.proj.top10BusyRoutes;

import java.io.IOException;
import java.util.Map;
import java.util.TreeMap;

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

public class Top10BusyRoutesReducer extends
Reducer<Text,LongWritable,Text,LongWritable> {

private TreeMap<Long,  String> topTenMap ;

    public void setup(Context context) {
            topTenMap = new TreeMap<Long, String>();
    }
```

```
    @Override
    public void reduce(Text key, Iterable<LongWritable> values,
                    Reducer<Text,LongWritable,Text,LongWritable>.Context
context) {
            String route  =  key.toString();
            long tripsCount = 0;
            for(LongWritable val : values) {
                    tripsCount = val.get();
            }
            topTenMap.put(tripsCount, route);

            if(topTenMap.size() > 10) {
                    topTenMap.remove(topTenMap.firstKey());
            }
    }

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

            for(Map.Entry<Long, String> m :
topTenMap.descendingMap().entrySet()) {
                    context.write(new Text(m.getValue()), new
LongWritable(m.getKey()));
            }
    }
}


package com.proj.top10BusyRoutes;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class Driver {
   public static void main( String[] args )
   {
    Configuration conf = new Configuration();
    try {
                    Job job = Job.getInstance(conf, "Top 10 busy routes");
```

```
                job.setJarByClass(Driver.class);


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

                //set inputformat and outputformat
                job.setInputFormatClass(TextInputFormat.class);
                job.setOutputFormatClass(TextOutputFormat.class);


                //set the output key and value types
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(LongWritable.class);

                //set mappers and reducers classes
                job.setMapperClass(Top10BusyRoutesMapper.class);
                job.setReducerClass(Top10BusyRoutesReducer.class);

                //set the input and output args
                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));

                //deletes if o/p path already exist
                FileSystem fs = FileSystem.get(conf);
                fs.delete(new Path(args[1]),true);

                //System.exit(job.waitForCompletion(true)? 0:1);
                long start = System.currentTimeMillis();
                if(job.waitForCompletion(true)) {
                        long end = System.currentTimeMillis();
                    float timeTaken = (end - start)/1000F;
                    System.out.println("time taken =
"+Float.toString(timeTaken));
                        System.exit(0);
                }else {
                        System.exit(1);
                }

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
```

```
}
```

8.6. Partitioning based on year:

```
package com.proj.partitioningByYear;

import org.apache.hadoop.conf.Configurable;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Partitioner;

public class YearPartitioner extends Partitioner<IntWritable, Text> implements
Configurable {

    private static final String MIN_LAST_ACCESS_DATE_YEAR =
"min.last.access.date.year";
    private Configuration conf = null;
    private int minLastAccessDateYear = 0;

    public Configuration getConf() {
        // TODO Auto-generated method stub
        return conf;
    }

    public void setConf(Configuration conf) {
        // TODO Auto-generated method stub
        this.conf = conf;
        minLastAccessDateYear =
conf.getInt(MIN_LAST_ACCESS_DATE_YEAR, 0);
    }

    @Override
    public int getPartition(IntWritable key, Text value, int numPartitions) {
        // TODO Auto-generated method stub
        return key.get()-minLastAccessDateYear;
    }

    public static void setMinLastAccessDate(Job job,int minLastAccessDateYear)
{
        job.getConfiguration().setInt(MIN_LAST_ACCESS_DATE_YEAR,
minLastAccessDateYear);
    }

}


package com.proj.partitioningByYear;

import java.io.IOException;
```

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

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

    private  IntWritable yrMapKey = new IntWritable();

    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            if(key.get()>0) {
                    String line = value.toString();
                    String[] tokens = line.split(",");

                    int year = Integer.parseInt(tokens[0]);
                    String row = tokens[0] + " " + tokens[8] + " " + tokens[9] + " "
+
                            tokens[16] + " " + tokens[17] ;

                    yrMapKey.set(year);

                    context.write(yrMapKey, new Text(row));
            }
    }

}


package com.proj.partitioningByYear;

import java.io.IOException;

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

public class YearPartitionReducer extends Reducer<IntWritable, Text, Text,
NullWritable> {

   protected void reduce(IntWritable key, Iterable<Text> values,
                   Context context) throws IOException, InterruptedException {
      for (Text t : values) {
         context.write(t, NullWritable.get());
      }
   }
}
```

```
package com.proj.partitioningByYear;

import java.io.IOException;

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.io.NullWritable;
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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


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


      Configuration conf = new Configuration();
      // Create a new Job
      Job job = Job.getInstance(conf,"Partitioning By Year");
      job.setJarByClass(Driver.class);

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


      job.setInputFormatClass(TextInputFormat.class);
      job.setOutputFormatClass(TextOutputFormat.class);

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


      job.setMapperClass(YearPartitionMapper.class);
      // Set custom partitioner and min last access date
      job.setPartitionerClass(YearPartitioner.class);
      YearPartitioner.setMinLastAccessDate(job, 1987);

      job.setNumReduceTasks(22);

      job.setReducerClass(YearPartitionReducer.class);

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

```
        //deletes if o/p path already exist
                FileSystem fs = FileSystem.get(conf);
                fs.delete(new Path(args[1]),true);

        // Submit the job, then poll for progress until the job is complete
        //System.exit(job.waitForCompletion(true)?0:1);
        long start = System.currentTimeMillis();
                if(job.waitForCompletion(true)) {
                        long end = System.currentTimeMillis();
                    float timeTaken = (end - start)/1000F;
                    System.out.println("time taken = "+Float.toString(timeTaken));
                        System.exit(0);
                }else {
                        System.exit(1);
                }

    }
}
```

8.7. Join Unique Carrier code with its name:

```
package com.proj.srcDestDistCarriersJoinName;

import java.io.IOException;

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

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

    Text code = new Text();

    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
                if(key.get()>0) {
                        String line = value.toString();
                        line = line.replace("\"", "");
                        String[] tokens = line.split(",");
                        code.set(tokens[0]);
                        String outValue= "A"+tokens[1];
                        context.write(code, new Text(outValue));
                }
    }

}


package com.proj.srcDestDistCarriersJoinName;
```

```
import java.io.IOException;

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

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

Text code = new Text();

    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            if(key.get()>0) {
                    String line = value.toString();
                    String[] tokens = line.split(",");
                    code.set(tokens[8]);
                    String outValue= "B"+tokens[16]+"-"+tokens[17]+"
"+tokens[18];
                    context.write(code, new Text(outValue));
            }
    }

}


package com.proj.srcDestDistCarriersJoinName;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;

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

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

   private Text tmp = new Text();
   private ArrayList<Text> listA = new ArrayList<Text>();
   private ArrayList<Text> listB = new ArrayList<Text>();
   private String joinType = null;

   public void setup(Context context) throws IOException, InterruptedException {
      joinType = context.getConfiguration().get("join.type");
   }

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

```
        while (values.iterator().hasNext()) {
            tmp = values.iterator().next();

            if (tmp.charAt(0) == 'A') {
                listA.add(new Text(tmp.toString().substring(1)));
            } else if (tmp.charAt(0) == 'B') {
                listB.add(new Text(tmp.toString().substring(1)));
            }
        }

        executeJoinLogic(context);
    }

    private void executeJoinLogic(Context context) throws IOException,
InterruptedException {
        if(joinType.equals("inner")){
            if(!listA.isEmpty() && !listB.isEmpty()){
                for(Text textA:listA){
                    for(Text textB:listB){
                        context.write(textA,textB);
                    }
                }
            }
        }
    }


}


package com.proj.srcDestDistCarriersJoinName;

import java.io.IOException;

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.MultipleInputs;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


public class Driver {

    public static void main(String[] args) {
```

```
    Configuration conf = new Configuration();
    try {
                    Job job = Job.getInstance(conf, "Reduceside Join - Inner join");
                    job.setJarByClass(Driver.class);

                    MultipleInputs.addInputPath(job, new Path(args[0]),
                TextInputFormat.class, CarrierMapper.class);
            MultipleInputs.addInputPath(job, new Path(args[1]),
                TextInputFormat.class, FlightMapper.class);
            job.getConfiguration().set("join.type", "inner");

            job.setReducerClass(SrcDestByCarrierReducer.class);

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

            job.setOutputFormatClass(TextOutputFormat.class);
                    FileOutputFormat.setOutputPath(job, new Path(args[2]));
//          TextOutputFormat.setOutputPath(job, new Path(args[2]));

                    //deletes if o/p path already exist
                    FileSystem fs = FileSystem.get(conf);
                    fs.delete(new Path(args[2]),true);

//                  System.exit(job.waitForCompletion(true)? 0:1);
                    long start = System.currentTimeMillis();
                    if(job.waitForCompletion(true)) {
                            long end = System.currentTimeMillis();
                        float timeTaken = (end - start)/1000F;
                        System.out.println("time taken =
"+Float.toString(timeTaken));
                             System.exit(0);
                    }else {
                            System.exit(1);
                    }

            } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            } catch (ClassNotFoundException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }
      }

}
```

8.8. Trips per route sorted by distance:

```
package com.proj.secsortTripsByDistance;

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

import org.apache.hadoop.io.WritableComparable;

public class CompositeKeyWritable implements WritableComparable{

    String route;
    Long distance;

   public CompositeKeyWritable() {

   }

    public CompositeKeyWritable(String route, Long distance) {
            super();
            this.route = route;
            this.distance = distance;
    }

    public String getRoute() {
            return route;
    }

    public void setRoute(String route) {
            this.route = route;
    }

    public Long getDistance() {
            return distance;
    }

    public void setDistance(Long distance) {
            this.distance = distance;
    }

    public void readFields(DataInput in) throws IOException {
            // TODO Auto-generated method stub
            route = in.readUTF();
            distance = in.readLong();
    }

    public void write(DataOutput out) throws IOException {
            // TODO Auto-generated method stub
            out.writeUTF(route);
```

```
            out.writeLong(distance);
    }

    public int compareTo(Object o) {
      String thisValue = this.getRoute();
      String thatValue = ((CompositeKeyWritable)o).getRoute();
      int result = thisValue.compareTo(thatValue);
      return (result < 0 ? -1 : (result == 0 ? 0 : 1));
    }


    @Override
    public String toString() {
            // TODO Auto-generated method stub
            return route + " " + distance;
    }

}


package com.proj.secsortTripsByDistance;

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


public class NaturalKeyGroupComparator extends WritableComparator{

    public NaturalKeyGroupComparator() {
            super(CompositeKeyWritable.class, true);
    }

    public int compare(WritableComparable a, WritableComparable b) {
            CompositeKeyWritable ck1 = (CompositeKeyWritable)a;
            CompositeKeyWritable ck2  =  (CompositeKeyWritable)b;

            int result = ck1.getDistance().compareTo(ck2.getDistance());
            return  result;
    }

}


package com.proj.secsortTripsByDistance;

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

public class NaturalKeyPartitioner extends
Partitioner<CompositeKeyWritable,IntWritable>{
```

```
    @Override
    public int getPartition(CompositeKeyWritable key, IntWritable val, int
numPartitions) {
            // TODO Auto-generated method stub
            return key.getDistance().hashCode() % numPartitions;
    }

}
```

```
package com.proj.secsortTripsByDistance;

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

public class SecondarySortComparator extends WritableComparator {

    public SecondarySortComparator() {
            super(CompositeKeyWritable.class, true);
    }

    public int compare(WritableComparable a, WritableComparable b) {
            CompositeKeyWritable ck1 = (CompositeKeyWritable)a;
            CompositeKeyWritable ck2  =  (CompositeKeyWritable)b;

            int result = ck1.getDistance().compareTo(ck2.getDistance());
            return -1 * result;
    }

}
```

```
package com.proj.secsortTripsByDistance;

import java.io.IOException;

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

public class TripsByDistanceMapper extends Mapper<LongWritable, Text,
CompositeKeyWritable, IntWritable>  {

    IntWritable one = new IntWritable(1);
    String route = "";
    Long distance;
    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
```

```
		if(key.get() > 0) {
			String line = value.toString();
			String[] tokens = line.split(",");
			route = tokens[16] + " - " + tokens[17];
			try {
				distance = Long.parseLong(tokens[18]);
			}catch(NumberFormatException e) {
				return;
			}
			CompositeKeyWritable ck = new
CompositeKeyWritable(route, distance);
			context.write(ck, one);
		}
	}
}


package com.proj.secsortTripsByDistance;

import java.io.IOException;

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

public class TripsByDistanceReducer extends Reducer<CompositeKeyWritable,
IntWritable, Text, IntWritable> {

	Text k = new Text();
	protected void reduce(CompositeKeyWritable key, Iterable<IntWritable>
values,
			Reducer<CompositeKeyWritable, IntWritable, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
		int sum = 0;
		// TODO Auto-generated method stub
		for (IntWritable val : values) {
			sum += val.get();
		}
		IntWritable count = new IntWritable(sum);
		k.set(key.getRoute() + " " + key.getDistance());
		context.write(k, count);
	}

}


package com.proj.secsortTripsByDistance;

import java.io.IOException;
```

```
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.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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


public class Driver {
   public static void main( String[] args )
   {
    Configuration conf = new Configuration();
    try {
                        Job job = Job.getInstance(conf, "Secondary sort");
                        job.setJarByClass(Driver.class);

                        job.setPartitionerClass(NaturalKeyPartitioner.class);

    job.setGroupingComparatorClass(NaturalKeyGroupComparator.class);
                        job.setSortComparatorClass(SecondarySortComparator.class);

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

                        //set inputformat and outputformat
                        job.setInputFormatClass(TextInputFormat.class);
                        job.setOutputFormatClass(TextOutputFormat.class);


                        //set the output key and value types
                        job.setOutputKeyClass(Text.class);
                        job.setOutputValueClass(IntWritable.class);

                        //set mappers and reducers classes
                        job.setMapperClass(TripsByDistanceMapper.class);
                        job.setReducerClass(TripsByDistanceReducer.class);

                        //set the input and output args
                        FileInputFormat.addInputPath(job, new Path(args[0]));
                        FileOutputFormat.setOutputPath(job, new Path(args[1]));

                        //deletes if o/p path already exist
                        FileSystem fs = FileSystem.get(conf);
                        fs.delete(new Path(args[1]),true);
```

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

        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
    }
}
```

8.9. Calculating minimum and maximum taxin/taxiout time:

```
package com.proj.minMaxTaxiInToOrigin;

import java.io.IOException;

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

public class MinMaxTaxiInMapper extends Mapper<LongWritable, Text, Text,
MinMaxTaxiInTuple>  {

    Text origin = new Text();
    MinMaxTaxiInTuple tuple = new MinMaxTaxiInTuple();

    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        if(key.get()>0) {
                String line = value.toString();
                String[] tokens = line.split(",");
                long taxiIn;
                try {
                        if(tokens[19]=="NA") return;
                        else {
                                taxiIn =  Long.parseLong(tokens[19]);
                        }
                }catch(NumberFormatException e) {
                        return;
                }
                tuple.setMaxTaxiIn(taxiIn);
                tuple.setMinTaxiIn(taxiIn);

                origin.set(tokens[16]);
                context.write(origin, tuple);
```

```
		}
	}

}
```

```
package com.proj.minMaxTaxiInToOrigin;

import java.io.IOException;

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

public class MinMaxTaxiInReducer extends
Reducer<Text,MinMaxTaxiInTuple,Text,MinMaxTaxiInTuple> {

	@Override
	public void reduce(Text key, Iterable<MinMaxTaxiInTuple> values, Context
context) throws IOException, InterruptedException {
		long minTaxiIn = Long.MAX_VALUE;
		long maxTaxiIn = Long.MIN_VALUE;

		for(MinMaxTaxiInTuple val : values) {
			if(val.getMinTaxiIn() < minTaxiIn)
				minTaxiIn = val.getMinTaxiIn();
			if(val.getMaxTaxiIn() > maxTaxiIn)
				maxTaxiIn = val.getMaxTaxiIn();
		}
		MinMaxTaxiInTuple tup = new MinMaxTaxiInTuple();
		tup.setMaxTaxiIn(maxTaxiIn);
		tup.setMinTaxiIn(minTaxiIn);
		context.write(key, tup);
	}

}
```

```
package com.proj.minMaxTaxiInToOrigin;

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

import org.apache.hadoop.io.WritableComparable;

public class MinMaxTaxiInTuple implements WritableComparable{

	long minTaxiIn = Long.MIN_VALUE;
	long maxTaxiIn = Long.MAX_VALUE;

	public long getMinTaxiIn() {
```

```
		return minTaxiIn;
	}

	public void setMinTaxiIn(long minTaxiIn) {
		this.minTaxiIn = minTaxiIn;
	}

	public long getMaxTaxiIn() {
		return maxTaxiIn;
	}

	public void setMaxTaxiIn(long maxTaxiIn) {
		this.maxTaxiIn = maxTaxiIn;
	}


	public void readFields(DataInput in) throws IOException {
		// TODO Auto-generated method stub
		minTaxiIn = in.readLong();
		maxTaxiIn = in.readLong();
	}

	public void write(DataOutput out) throws IOException {
		// TODO Auto-generated method stub
		out.writeLong(minTaxiIn);
		out.writeLong(maxTaxiIn);
	}

	public int compareTo(Object o) {
		// TODO Auto-generated method stub
		return 0;
	}

	@Override
	public String toString() {
		// TODO Auto-generated method stub
		return "Min TaxiIn: " + minTaxiIn + " Max TaxiIn: " + maxTaxiIn ;
	}

}


package com.proj.minMaxTaxiInToOrigin;

import java.io.IOException;

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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class Driver {
   public static void main( String[] args )
   {
    Configuration conf = new Configuration();
    try {
                        Job job = Job.getInstance(conf, "Min Max TaxiIn time to origin
airport");
                        job.setJarByClass(Driver.class);


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

                        //set inputformat and outputformat
                        job.setInputFormatClass(TextInputFormat.class);
                        job.setOutputFormatClass(TextOutputFormat.class);


                        //set the output key and value types
                        job.setOutputKeyClass(Text.class);
                        job.setOutputValueClass(MinMaxTaxiInTuple.class);

                        //set mappers and reducers classes
                        job.setMapperClass(MinMaxTaxiInMapper.class);
                        job.setReducerClass(MinMaxTaxiInReducer.class);

                        //set the input and output args
                        FileInputFormat.addInputPath(job, new Path(args[0]));
                        FileOutputFormat.setOutputPath(job, new Path(args[1]));

                        //deletes if o/p path already exist
                        FileSystem fs = FileSystem.get(conf);
                        fs.delete(new Path(args[1]),true);

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

                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (ClassNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
```

```
                    e.printStackTrace();
                }
            }
        }
```

8.10. Average Taxiin/Taxiout time:

```
package com.proj.averageTaxiInTimeToOrigin;

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

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


public class AverageMapper extends Mapper<LongWritable, Text, Text,
CountAverageWritable> {

    Text origin = new Text();
    CountAverageWritable countAvg = new CountAverageWritable();

    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        if(key.get()>0) {
            String line = value.toString();
            String[] tokens = line.split(",");

            long taxiIn = 0;
            try{
                if(tokens[19]=="NA")
                    taxiIn = 0;
                else
                    taxiIn = Long.parseLong(tokens[19]);
            }catch(NumberFormatException e) {
                return;
            }
            countAvg.setAverage(taxiIn);
            countAvg.setCount(1);

            origin.set(tokens[16]);
            context.write(origin, countAvg);
        }
    }
```

```
}


package com.proj.averageTaxiInTimeToOrigin;

import java.io.IOException;

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


public class AverageReducer extends Reducer<Text, CountAverageWritable, Text,
CountAverageWritable>{

    CountAverageWritable result = new CountAverageWritable();

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

            double sum = 0.0;
            long count = 0;

            for(CountAverageWritable  val : values) {
                sum += val.getCount() * val.getAverage();
                count += val.getCount();
            }

            result.setCount(count);
            result.setAverage(sum/count);

            context.write(key, result);

    }

}


package com.proj.averageTaxiInTimeToOrigin;

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

import org.apache.hadoop.io.Writable;

public class CountAverageWritable implements Writable {

    long count = 0;
```

```
    double average = 0.0;

    public long getCount() {
            return count;
    }

    public void setCount(long count) {
            this.count = count;
    }

    public double getAverage() {
            return average;
    }

    public void setAverage(double average) {
            this.average = average;
    }

    public void readFields(DataInput in) throws IOException {
            // TODO Auto-generated method stub
            count =  in.readLong();
            average = in.readDouble();
    }

    public void write(DataOutput out) throws IOException {
            // TODO Auto-generated method stub
            out.writeLong(count);
            out.writeDouble(average);
    }

    public String toString() {
            String avg = (String) String.format("%.2f", average);
            return "Average TaxiIn time = " + avg;
    }

}


package com.proj.averageTaxiInTimeToOrigin;

import java.io.IOException;

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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


/**
 * Hello world!
 *
 */
public class Driver
{
   public static void main( String[] args )
   {
     Configuration conf = new Configuration();
     try {
                        Job job = Job.getInstance(conf, "Avg Taxi in time");
                        job.setJarByClass(Driver.class);


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

                        //set inputformat and outputformat
                        job.setInputFormatClass(TextInputFormat.class);
                        job.setOutputFormatClass(TextOutputFormat.class);


                        //set the output key and value types
                        job.setOutputKeyClass(Text.class);
                        job.setOutputValueClass(CountAverageWritable.class);

                        //set mappers and reducers classes
                        job.setMapperClass(AverageMapper.class);
                        job.setCombinerClass(AverageReducer.class);
                        job.setReducerClass(AverageReducer.class);

                        //set the input and output args
                        FileInputFormat.addInputPath(job, new Path(args[0]));
                        FileOutputFormat.setOutputPath(job, new Path(args[1]));

                        //deletes if o/p path already exist
                        FileSystem fs = FileSystem.get(conf);
                        fs.delete(new Path(args[1]),true);

                        //System.exit(job.waitForCompletion(true)? 0:1);
                        long start = System.currentTimeMillis();
                        if(job.waitForCompletion(true)) {
                                long end = System.currentTimeMillis();
                            float timeTaken = (end - start)/1000F;
                            System.out.println("time taken =
"+Float.toString(timeTaken));
                                System.exit(0);
```

```
                }else {
                        System.exit(1);
                }

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
    }
}
```

8.11. Median and standard deviation of distance travelled:

```
package com.proj.distanceMedianSD;

import java.io.IOException;

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

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

    private Text year = new Text();
    private LongWritable dist = new LongWritable();

    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            if(key.get()>0) {
                String line = value.toString();
        String[] tokens= line.split(",");

        year.set(tokens[0]);
        if(tokens[18].equalsIgnoreCase("NA")) {
            return;
        }
        try {
            dist.set(Long.parseLong(tokens[18]));
        } catch(NumberFormatException e) {
            return;
        }
```

```
            context.write(year, dist);
                }
        }

}


package com.proj.distanceMedianSD;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;

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

public class DistanceReducer extends Reducer<Text,
LongWritable,Text,MedianStdDevTuple> {

    MedianStdDevTuple tuple = new MedianStdDevTuple();
    ArrayList<Float> list = new ArrayList<Float>();

    public void reduce(Text key, Iterable<LongWritable> values, Context context)
throws IOException, InterruptedException {
            float sum =0;
        float count=0;
        list.clear();
        tuple.setMedianDist(0);

        for(LongWritable val : values) {
            list.add((float)val.get());
            sum += (float)val.get();
            ++count;
        }

        Collections.sort(list);

        if(count%2==0){
            tuple.setMedianDist((list.get((int)count/2-1) +
list.get((int)count/2))/2.0f);
        }else {
            tuple.setMedianDist(list.get((int)count/2));
        }

        float mean = sum/count;
        float sumOfSquares =0.0f;
        for(Float f :list){
            sumOfSquares += (f-mean) * (f-mean);
        }
```

```
            tuple.setStdDevDist((float)Math.sqrt(sumOfSquares/(count-1)));
            context.write(key, tuple);

        }

    }


    package com.proj.distanceMedianSD;

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

    import org.apache.hadoop.io.Writable;

    public class MedianStdDevTuple implements Writable {

        private float stdDevDist;
       private float medianDist;

       public MedianStdDevTuple() {
        super();
       }

        public float getStdDevDist() {
                return stdDevDist;
        }

        public void setStdDevDist(float stdDevDist) {
                this.stdDevDist = stdDevDist;
        }

        public float getMedianDist() {
                return medianDist;
        }

        public void setMedianDist(float medianDist) {
                this.medianDist = medianDist;
        }

        public void readFields(DataInput in) throws IOException {
                // TODO Auto-generated method stub
                stdDevDist=in.readFloat();
          medianDist=in.readFloat();
        }

        public void write(DataOutput out) throws IOException {
                // TODO Auto-generated method stub
```

```
            out.writeFloat(stdDevDist);
      out.writeFloat(medianDist);
     }

     @Override
   public String toString() {
      return "Standard Deviation : "+ stdDevDist+";  Median_Distance: " +
medianDist;
   }

}


package com.proj.distanceMedianSD;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


public class Driver {
   public static void main( String[] args )
   {
    Configuration conf = new Configuration();
    try {
                    Job job = Job.getInstance(conf, "Median std deviation for
distance");
                    job.setJarByClass(Driver.class);


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

                    //set inputformat and outputformat
                    job.setInputFormatClass(TextInputFormat.class);
                    job.setOutputFormatClass(TextOutputFormat.class);


                    //set the output key and value types
                    job.setOutputKeyClass(Text.class);
                    job.setOutputValueClass(MedianStdDevTuple.class);
```

```
                        //set mappers and reducers classes
                        job.setMapperClass(DistanceMapper.class);
                        job.setReducerClass(DistanceReducer.class);

                        //set the input and output args
                        FileInputFormat.addInputPath(job, new Path(args[0]));
                        FileOutputFormat.setOutputPath(job, new Path(args[1]));

                        //deletes if o/p path already exist
                        FileSystem fs = FileSystem.get(conf);
                        fs.delete(new Path(args[1]),true);

                        //System.exit(job.waitForCompletion(true)? 0:1);
                        long start = System.currentTimeMillis();
                        if(job.waitForCompletion(true)) {
                                long end = System.currentTimeMillis();
                            float timeTaken = (end - start)/1000F;
                            System.out.println("time taken =
"+Float.toString(timeTaken));
                                 System.exit(0);
                        }else {
                                System.exit(1);
                        }

                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (ClassNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
    }
}
```

8.12. Flight number indexed by Unique carrier:

```
package com.proj.flightNumsUnderCarrier;

import java.io.IOException;

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

```
public class IndexByCarrierMapper extends Mapper<LongWritable, Text, Text,
Text> {

    @Override
   protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            if(key.get()>0) {
                    String line = value.toString();
          String[] tokens= line.split(",");

          context.write(new Text(tokens[8]), new Text(tokens[9]));
            }
    }

}


package com.proj.flightNumsUnderCarrier;

import java.io.IOException;
import java.util.ArrayList;

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

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

    private Text result = new Text();
    ArrayList<String> list = new ArrayList<String>(); // To insert only unique
flight numbers

    public void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
            list.clear();
            StringBuilder sb = new StringBuilder();
            for(Text carrier :  values) {
                    if(!list.contains(carrier.toString()))
                            list.add(carrier.toString());
            }
            boolean first = true;
            for(String carr : list) {
                    if(first)
                            first = false;
                    else
                            sb.append(",");
                    sb.append(carr.toString());
            }
            result.set(sb.toString());
            context.write(key, result);
    }
```

```
}


package com.proj.flightNumsUnderCarrier;

import java.io.IOException;

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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;


public class Driver {
   public static void main( String[] args )
   {
    Configuration conf = new Configuration();
    try {
                        Job job = Job.getInstance(conf, "Flights under a carrier line");
                        job.setJarByClass(Driver.class);


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

                        //set inputformat and outputformat
                        job.setInputFormatClass(TextInputFormat.class);
                        job.setOutputFormatClass(TextOutputFormat.class);


                        //set the output key and value types
                        job.setOutputKeyClass(Text.class);
                        job.setOutputValueClass(Text.class);

                        //set mappers and reducers classes
                        job.setMapperClass(IndexByCarrierMapper.class);
                        job.setReducerClass(IndexByCarrierReducer.class);

                        //set the input and output args
                        FileInputFormat.addInputPath(job, new Path(args[0]));
                        FileOutputFormat.setOutputPath(job, new Path(args[1]));

                        //deletes if o/p path already exist
                        FileSystem fs = FileSystem.get(conf);
```

```
                fs.delete(new Path(args[1]),true);

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

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
    }
}
```

8.13. Flights per each cancellation reason:

```
package com.proj.numFlightsCancelledPerReason;


import java.io.IOException;

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

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


    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        if(key.get()>0) {
            String line = value.toString();
        String[] tokens= line.split(",");

        String cancelCode = tokens[22];
        String code = "";
        if(cancelCode.isEmpty() || cancelCode == null || tokens[21]=="0")
            code = "Not Cancelled";
        else if(cancelCode.equals("A"))
            code = "Carrier";
        else if(cancelCode.equals("B"))
            code = "Weather";
        else if(cancelCode.equals("C"))
            code = "NAS";
        else if(cancelCode.equals("D"))
```

```
            code = "Security";
        else
            code = "Not Cancelled";
        context.write(new Text(code), new LongWritable(1));
            }
    }



}


package com.proj.numFlightsCancelledPerReason;

import java.io.IOException;

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

public class CountCancelledReducer extends Reducer<Text, LongWritable, Text,
LongWritable> {
    LongWritable count = new LongWritable();

    @Override
    public void reduce(Text key, Iterable<LongWritable> values, Context context)
throws IOException, InterruptedException {
            long sum = 0;
            for(LongWritable val : values) {
                    sum += val.get();
            }
            count.set(sum);
            context.write(key, count);
    }

}

package com.proj.numFlightsCancelledPerReason;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class Driver {
    public static void main(String[] args) {

            Configuration conf = new Configuration();
            try {
                    Job job = Job.getInstance(conf, "Total number of flights
cancelled for each reason from 1987 to 2008");
                    job.setJarByClass(Driver.class);


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

                    //set inputformat and outputformat
                    job.setInputFormatClass(TextInputFormat.class);
                    job.setOutputFormatClass(TextOutputFormat.class);


                    //set the output key and value types
                    job.setOutputKeyClass(Text.class);
                    job.setOutputValueClass(LongWritable.class);

                    //set mappers and reducers classes
                    job.setMapperClass(CountCancelledMapper.class);
                    job.setReducerClass(CountCancelledReducer.class);

                    //set the input and output args
                    FileInputFormat.addInputPath(job, new Path(args[0]));
                    FileOutputFormat.setOutputPath(job, new Path(args[1]));

                    //deletes if o/p path already exist
                    FileSystem fs = FileSystem.get(conf);
                    fs.delete(new Path(args[1]),true);

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

            } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            } catch (ClassNotFoundException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }

    }
```

```
}
```

8.14. Busy day of week:

```
package com.proj.busyWeekDay;

import java.io.IOException;

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

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

    public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            if(key.get()>0) {
                    String line = value.toString();
                    String[] tokens = line.split(",");
                    String weekDay = tokens[3];
                    context.write(new Text(weekDay), new IntWritable(1));
            }
    }

}


package com.proj.busyWeekDay;

import java.io.IOException;

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

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

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
            int sum=0;
      for(IntWritable a:values){
         sum+=a.get();
      }

      context.write(key,new IntWritable(sum));
    }
```

```
}


package com.proj.busyWeekDay;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

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

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

     int total_count =0;
   HashMap<String,IntWritable> map;
   @Override
   protected void setup(Context context) throws IOException,
InterruptedException {
       map = new HashMap<String, IntWritable>();
   }

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

   @Override
   protected void cleanup(Context context) throws IOException,
InterruptedException {
       for(Map.Entry <String,IntWritable> entry: map.entrySet()){
          int v = entry.getValue().get();
          float percent = (float) v/total_count;
          percent = percent*100;
          context.write(new Text(entry.getKey()), new Text(entry.getValue() + " / " +
total_count + " = " +
                  percent+"% Busy"));
       }
   }

}
```

```
package com.proj.busyWeekDay;

import java.io.IOException;

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.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 org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class Driver {
    public static void main(String[] args) {
    Configuration conf = new Configuration();
    try {
                        Job job = Job.getInstance(conf, "Most busy day of weeek");
                        job.setJarByClass(Driver.class);
                        job.setMapOutputKeyClass(Text.class);
                        job.setMapOutputValueClass(IntWritable.class);

                        //set inputformat and outputformat
                        job.setInputFormatClass(TextInputFormat.class);
                        job.setOutputFormatClass(TextOutputFormat.class);


                        //set the output key and value types
                        job.setOutputKeyClass(Text.class);
                        job.setOutputValueClass(Text.class);

                        //set mappers and reducers classes
                        job.setMapperClass(BusyWeekDayMapper.class);
                        job.setCombinerClass(BusyWeekDayCombiner.class);
                        job.setReducerClass(BusyWeekDayReducer.class);
                        job.setNumReduceTasks(1);

                        //set the input and output args
                        FileInputFormat.addInputPath(job, new Path(args[0]));
                        FileOutputFormat.setOutputPath(job, new Path(args[1]));

                        //deletes if o/p path already exist
                        FileSystem fs = FileSystem.get(conf);
                        fs.delete(new Path(args[1]),true);

                        //System.exit(job.waitForCompletion(true)? 0:1);
                        long start = System.currentTimeMillis();
                        if(job.waitForCompletion(true)) {
```

```
                        long end = System.currentTimeMillis();
                    float timeTaken = (end - start)/1000F;
                    System.out.println("time taken =
"+Float.toString(timeTaken));
                        System.exit(0);
                }else {
                        System.exit(1);
                }

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
    }

}
```