FINAL PROJECT

INFO7250- Engg Of Big Data

Ankit Yadav

NUID- 001271369

Northeastern University

CONTENTS

0. Overview about the dataset

I am using the data on **Airline On-Time Statistics and Delay Causes from**

http://stat-computing.org/dataexpo/2009/the-data.html

This is dataset containing information about airline schedule with following columns:

Variable descriptions

	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

The reason of selections this data set is that it has many numbers of columns which will enable me to use various MapReduce algorithms studies in the course for different types of analysis.

Also, the data is evenly segregated in yearly basis. So, in case If I can am unable to load complete data in my computer then too I can do the same analysis on small portion of same data more easily.

The complete data was downloaded using following script:

```
Open
# How to unzip files in linux
# bzip2 -d filename.bz2
# Note, that this command will not preserve original archive file.
# To preserve the original archive, add the -k option:
# bzip2 -dk filename.bz2

for i in {1987..2008}; do
  echo "downloading start for " $i
  wget http://stat-computing.org/dataexpo/2009/$i.csv.bz2
  echo "downloading done for " $i
  echo "unzipping start for " $i
  bzip2 -dk $i.csv.bz2
  echo "unzipping done for " $i
  echo "removing header start for " $i
  sed -i "1 d" "$i.csv"
  echo "removing header done for " $i
done
echo "now combining all files"
cat *csv > combined.csv

done
```

1. Analysis of Flight Data using MapReduce on Hadoop

1- Getting total count of all the data:

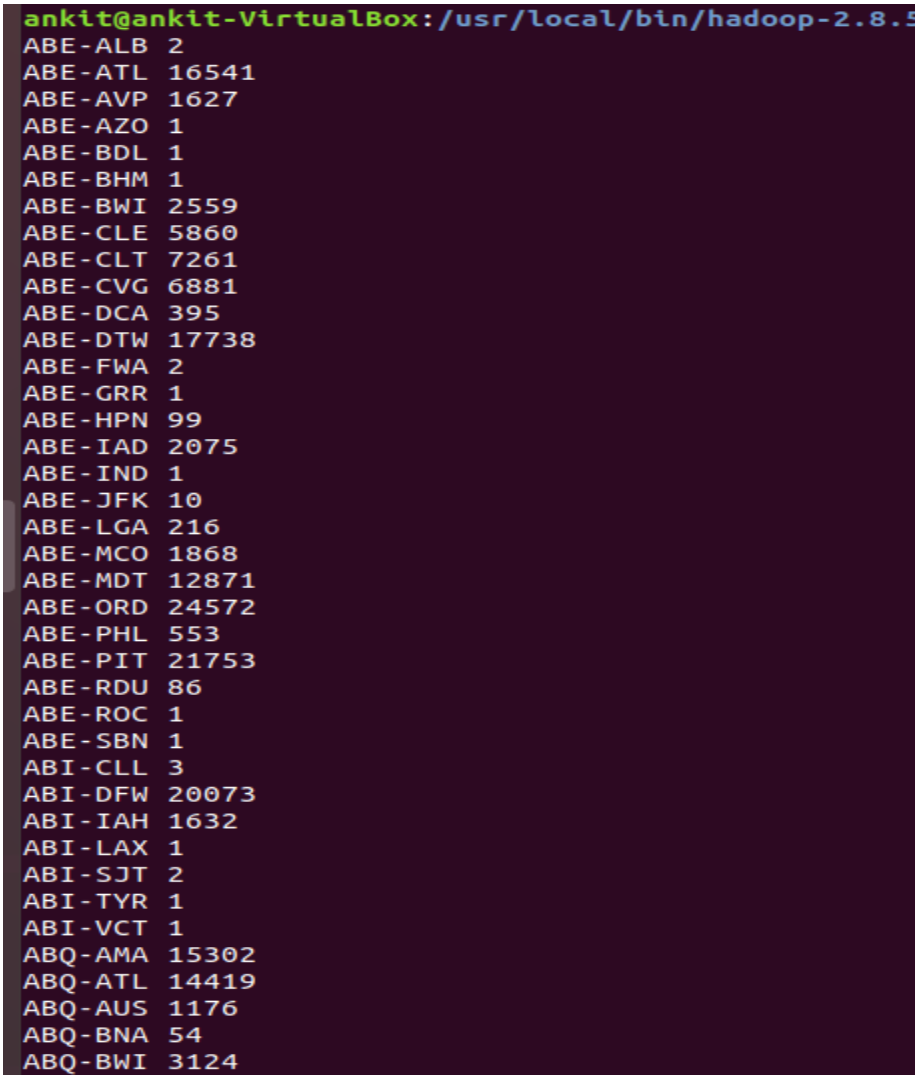
This is a very basic map reduce use case in which we count the whole data to get a sense of how many total records are there:

```
hadoop jar ~/Downloads/ProjectJars/count.jar
hadoop.project.total_count.MRCount /flight-data /FinalProjectMROutput/2-
Total-Data-Count
```

The final count is: **123534970**

2- Getting the total flights from all source destinations pairs in from 1987 to 2008:

This was a huge data and MapReduce made this analysis quite simple and fast:



3: Top 30 source destination pairs

Sorting the above data to get top 30 most busy Source Destination pair:

4: Delay in flight percentage

We considered the delay greater than or equal to 15 minutes as delay . Now we need to count those flights which had delay greater than or equal to 15 minutes.

Delayed flight Count:

Percentage of departure delayed flights: Total Flight Count/ Departure Delayed Flight count

= (19690422/123534970) * 100 = 15.94 %

So, this shows that the actual delay greater that 15 minutes is very less and generally flights depart on time.

Let’s check the same for arrival delay

Percentage of departure delayed flights: Total Flight Count/ Delayed Flight count

= (24627925/123534970) * 100 = 19.9 %

So, the delay in departure and arrival is between 15 to 20 % range.

So, it shows that overall flights are mostly on time from/to all source destination

5- Count of unique carrier’s flights

The data for unique carriers are as follows:

```
ankit@ankit-VirtualBox:/usr/local/bin/
9E      521059
AA      14984647
AQ      154381
AS      2878021
B6      811341
CO      8145788
DH      693047
DL      16547870
EA      919785
EV      1697172
F9      336958
FL      1265138
HA      274265
HP      3636682
ML (1)  70622
MQ      3954895
NW      10292627
OH      1464176
OO      3090853
PA (1)  316167
PI      873957
PS      83617
TW      3757747
TZ      208420
UA      13299817
US      14075530
WN      15976022
XE      2350309
YV      854056
ankit@ankit-VirtualBox:/usr/local/bin/
```

6- Inner Join to get the full name for unique carriers

We did inner join with between two files to get carrier names instead of carrier codes

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /FinalProjectMROutput/7-Inner-Join-Carrier
Pinnacle Airlines Inc.  521059
American Airlines Inc.  14984647
Aloha Airlines Inc.     154381
Alaska Airlines Inc.    2878021
JetBlue Airways 811341
Continental Air Lines Inc.      8145788
Independence Air        693047
Delta Air Lines Inc.    16547870
Eastern Air Lines Inc.  919785
Atlantic Southeast Airlines     1697172
Frontier Airlines Inc.  336958
AirTran Airways Corporation     1265138
Hawaiian Airlines Inc.  274265
America West Airlines Inc. (Merged with US Airways 9/05. Stopped reporting 10/07.)      3636682
Midway Airlines Inc. (1)        70622
American Eagle Airlines Inc.    3954895
Northwest Airlines Inc. 10292627
Comair Inc.     1464176
Skywest Airlines Inc.   3090853
Pan American World Airways (1)  316167
Piedmont Aviation Inc.  873957
Pacific Southwest Airlines      83617
Trans World Airways LLC 3757747
ATA Airlines d/b/a ATA  208420
United Air Lines Inc.   13299817
US Airways Inc. (Merged with America West 9/05. Reporting for both starting 10/07.)     14075530
Southwest Airlines Co.  15976022
Expressjet Airlines Inc.        2350309
Mesa Airlines Inc.      854056
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

7- Getting Flight data by year

```
ankit@ankit-VirtualBox:
1987    1311826
1988    5202096
1989    5041200
1990    5270893
1991    5076925
1992    5092157
1993    5070501
1994    5180048
1995    5327435
1996    5351983
1997    5411843
1998    5384721
1999    5527884
2000    5683047
2001    5967780
2002    5271359
2003    6488540
2004    7129270
2005    7140596
2006    7141922
2007    7453215
2008    7009728
ankit@ankit-VirtualBox:
```

8- Delayed flights per year

In this we will check delayed flights per year(we will count flights as delayed only is the delay time if greater than equal to 15 minutes)

```
                Bytes Written=279
ankit@ankit-VirtualBox:/usr/local/bi
1987    312770
1988    977853
1989    1119466
1990    1019363
1991    833978
1992    838347
1993    861259
1994    881408
1995    1039250
1996    1220045
1997    1083834
1998    1070071
1999    1152725
2000    1356040
2001    1104439
2002    868225
2003    1057804
2004    1421391
2005    1466065
2006    1615537
2007    1803320
2008    1524735
ankit@ankit-VirtualBox:/usr/local/bi
```

9- Cancelled flights by year

```
ankit@ankit-VirtualBox:/usr/local/bi
1987    19685
1988    50163
1989    74165
1990    52458
1991    43505
1992    52836
1993    59845
1994    66740
1995    91905
1996    128536
1997    97763
1998    144509
1999    154311
2000    187490
2001    231198
2002    65143
2003    101469
2004    127757
2005    133730
2006    121934
2007    160748
2008    137434
ankit@ankit-VirtualBox:/usr/local/bi
```

10- Ratio of delayed flights per year to total flights

We can get percentage of delayed flights per year also

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /FinalProjectMROutput
1987    flightCount=1311826, delayedFlightCount=291947, delayPercent=22.26
1988    flightCount=5202096, delayedFlightCount=910460, delayPercent=17.50
1989    flightCount=5041200, delayedFlightCount=1050606, delayPercent=20.84
1990    flightCount=5270893, delayedFlightCount=954609, delayPercent=18.11
1991    flightCount=5076925, delayedFlightCount=777309, delayPercent=15.31
1992    flightCount=5092157, delayedFlightCount=779598, delayPercent=15.31
1993    flightCount=5070501, delayedFlightCount=805674, delayPercent=15.89
1994    flightCount=5180048, delayedFlightCount=825865, delayPercent=15.94
1995    flightCount=5327435, delayedFlightCount=982790, delayPercent=18.45
1996    flightCount=5351983, delayedFlightCount=1161396, delayPercent=21.70
1997    flightCount=5411843, delayedFlightCount=1030159, delayPercent=19.04
1998    flightCount=5384721, delayedFlightCount=1020934, delayPercent=18.96
1999    flightCount=5527884, delayedFlightCount=1101355, delayPercent=19.92
2000    flightCount=5683047, delayedFlightCount=1301615, delayPercent=22.90
2001    flightCount=5967780, delayedFlightCount=1053819, delayPercent=17.66
2002    flightCount=5271359, delayedFlightCount=823147, delayPercent=15.62
2003    flightCount=6488540, delayedFlightCount=1005631, delayPercent=15.50
2004    flightCount=7129270, delayedFlightCount=1355988, delayPercent=19.02
2005    flightCount=7140596, delayedFlightCount=1399557, delayPercent=19.60
2006    flightCount=7141922, delayedFlightCount=1548755, delayPercent=21.69
2007    flightCount=7453215, delayedFlightCount=1734629, delayPercent=23.27
2008    flightCount=7009728, delayedFlightCount=1466191, delayPercent=20.92
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

It shows that the years 1991-1994, 2002,2003 were best years to fly as they had least delayed flights (less than 16%).

Years with most delays were- 1987, 2000, 2007 with more than 22% flights delayed

11- Total flights by day of week and ratio to delayed

Following is the data of total flights , delayed flights and their ratio.

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /FinalProjectMROutput
Friday  flightCount=18091338, delayedFlightCount=4004214, delayPercent=22.13
Monday  flightCount=18136111, delayedFlightCount=3298072, delayPercent=18.19
Saturday        flightCount=15915382, delayedFlightCount=2520933, delayPercent=15.84
Sunday  flightCount=17143178, delayedFlightCount=3151506, delayPercent=18.38
Thursday        flightCount=18083800, delayedFlightCount=3838270, delayPercent=21.22
Tuesday flightCount=18061938, delayedFlightCount=3153109, delayPercent=17.46
Wednesday       flightCount=18103222, delayedFlightCount=3415930, delayPercent=18.87
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

From this data we can infer that maximum delay is on Thursday and Fridays that is when weekends are starting .

Best day to fly are when the weekends ends like Saturday, Sunday or on weekdays

12- Total flights by months of year and ratio to delayed

We can infer from this data that best months to fly was September with least delay- 14 %

Other good months were- April, May and October with delay – 16%

Worst month was December- 24% flights delayed. It may be due to big holidays season in December.

13- Total delayed flights by flight carriers and ratio of delayed flights

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /FinalProjectMROutput/14-Delay_Ratio
9E      flightCount=521059, delayedFlightCount=94432, delayPercent=18.12
AA      flightCount=14984647, delayedFlightCount=2819508, delayPercent=18.82
AQ      flightCount=154381, delayedFlightCount=13195, delayPercent=8.55
AS      flightCount=2878021, delayedFlightCount=593864, delayPercent=20.63
B6      flightCount=811341, delayedFlightCount=191762, delayPercent=23.64
CO      flightCount=8145788, delayedFlightCount=1555471, delayPercent=19.10
DH      flightCount=693047, delayedFlightCount=141765, delayPercent=20.46
DL      flightCount=16547870, delayedFlightCount=3215538, delayPercent=19.43
EA      flightCount=919785, delayedFlightCount=156324, delayPercent=17.00
EV      flightCount=1697172, delayedFlightCount=418887, delayPercent=24.68
F9      flightCount=336958, delayedFlightCount=62934, delayPercent=18.68
FL      flightCount=1265138, delayedFlightCount=281657, delayPercent=22.26
HA      flightCount=274265, delayedFlightCount=16706, delayPercent=6.09
HP      flightCount=3636682, delayedFlightCount=670214, delayPercent=18.43
ML (1)  flightCount=70622, delayedFlightCount=9288, delayPercent=13.15
MQ      flightCount=3954895, delayedFlightCount=842571, delayPercent=21.30
NW      flightCount=10292627, delayedFlightCount=1815983, delayPercent=17.64
OH      flightCount=1464176, delayedFlightCount=304364, delayPercent=20.79
OO      flightCount=3090853, delayedFlightCount=517173, delayPercent=16.73
PA (1)  flightCount=316167, delayedFlightCount=57436, delayPercent=18.17
PI      flightCount=873957, delayedFlightCount=201513, delayPercent=23.06
PS      flightCount=83617, delayedFlightCount=17789, delayPercent=21.27
TW      flightCount=3757747, delayedFlightCount=709233, delayPercent=18.87
TZ      flightCount=208420, delayedFlightCount=39135, delayPercent=18.78
UA      flightCount=13299817, delayedFlightCount=2761933, delayPercent=20.77
US      flightCount=14075530, delayedFlightCount=2615152, delayPercent=18.58
WN      flightCount=15976022, delayedFlightCount=2565525, delayPercent=16.06
XE      flightCount=2350309, delayedFlightCount=502089, delayPercent=21.36
YV      flightCount=854056, delayedFlightCount=190593, delayPercent=22.32
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /FinalProjectMROutput/14.1-Delay_Ratio_Carriers_Name/part-r-00000
Pinnacle Airlines Inc.  flightCount=521059, delayedFlightCount=94432, delayPercent=18.12
American Airlines Inc.  flightCount=14984647, delayedFlightCount=2819508, delayPercent=18.82
Aloha Airlines Inc.     flightCount=154381, delayedFlightCount=13195, delayPercent=8.55
Alaska Airlines Inc.    flightCount=2878021, delayedFlightCount=593864, delayPercent=20.63
JetBlue Airways flightCount=811341, delayedFlightCount=191762, delayPercent=23.64
Continental Air Lines Inc.      flightCount=8145788, delayedFlightCount=1555471, delayPercent=19.10
Independence Air        flightCount=693047, delayedFlightCount=141765, delayPercent=20.46
Delta Air Lines Inc.    flightCount=16547870, delayedFlightCount=3215538, delayPercent=19.43
Eastern Air Lines Inc.  flightCount=919785, delayedFlightCount=156324, delayPercent=17.00
Atlantic Southeast Airlines     flightCount=1697172, delayedFlightCount=418887, delayPercent=24.68
Frontier Airlines Inc.  flightCount=336958, delayedFlightCount=62934, delayPercent=18.68
AirTran Airways Corporation     flightCount=1265138, delayedFlightCount=281657, delayPercent=22.26
Hawaiian Airlines Inc.  flightCount=274265, delayedFlightCount=16706, delayPercent=6.09
America West Airlines Inc. (Merged with US Airways 9/05. Stopped reporting 10/07.)      flightCount=3636682, delayedFlightCount=670214, delayPercent=18.43
Midway Airlines Inc. (1)        flightCount=70622, delayedFlightCount=9288, delayPercent=13.15
American Eagle Airlines Inc.    flightCount=3954895, delayedFlightCount=842571, delayPercent=21.30
Northwest Airlines Inc. flightCount=10292627, delayedFlightCount=1815983, delayPercent=17.64
Comair Inc.     flightCount=1464176, delayedFlightCount=304364, delayPercent=20.79
Skywest Airlines Inc.   flightCount=3090853, delayedFlightCount=517173, delayPercent=16.73
Pan American World Airways (1)  flightCount=316167, delayedFlightCount=57436, delayPercent=18.17
Piedmont Aviation Inc.  flightCount=873957, delayedFlightCount=201513, delayPercent=23.06
Pacific Southwest Airlines      flightCount=83617, delayedFlightCount=17789, delayPercent=21.27
Trans World Airways LLC flightCount=3757747, delayedFlightCount=709233, delayPercent=18.87
ATA Airlines d/b/a ATA  flightCount=208420, delayedFlightCount=39135, delayPercent=18.78
United Air Lines Inc.   flightCount=13299817, delayedFlightCount=2761933, delayPercent=20.77
US Airways Inc. (Merged with America West 9/05. Reporting for both starting 10/07.)     flightCount=14075530, delayedFlightCount=2615152, delayPercent=18.58
Southwest Airlines Co.  flightCount=15976022, delayedFlightCount=2565525, delayPercent=16.06
Expressjet Airlines Inc.        flightCount=2350309, delayedFlightCount=502089, delayPercent=21.36
Mesa Airlines Inc.      flightCount=854056, delayedFlightCount=190593, delayPercent=22.32
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

By using this analysis we can check which carriers are more prone to delays and can plan flights with those carriers who are less prone to delays.

Carriers with least delays- **Hawaiian Airlines, Aloha Airlines** with **6%** and **8%** flights delayed respectively.

Carriers with most delays- **JetBlue Airways, Atlantic Southeast Airlines** with around **24%** flights delayed.

14- Total cancelled flights by flight carriers and ratio of cancelled flights

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /FinalProjectMROutput/15-Cancelle
9E      flightCount=521059, delayedFlightCount=15039, delayPercent=2.89
AA      flightCount=14984647, delayedFlightCount=286889, delayPercent=1.91
AQ      flightCount=154381, delayedFlightCount=2837, delayPercent=1.84
AS      flightCount=2878021, delayedFlightCount=57121, delayPercent=1.98
B6      flightCount=811341, delayedFlightCount=9281, delayPercent=1.14
CO      flightCount=8145788, delayedFlightCount=113064, delayPercent=1.39
DH      flightCount=693047, delayedFlightCount=22176, delayPercent=3.20
DL      flightCount=16547870, delayedFlightCount=258382, delayPercent=1.56
EA      flightCount=919785, delayedFlightCount=28702, delayPercent=3.12
EV      flightCount=1697172, delayedFlightCount=48676, delayPercent=2.87
F9      flightCount=336958, delayedFlightCount=1778, delayPercent=0.53
FL      flightCount=1265138, delayedFlightCount=12854, delayPercent=1.02
HA      flightCount=274265, delayedFlightCount=1329, delayPercent=0.48
HP      flightCount=3636682, delayedFlightCount=55431, delayPercent=1.52
ML (1)  flightCount=70622, delayedFlightCount=1342, delayPercent=1.90
MQ      flightCount=3954895, delayedFlightCount=157478, delayPercent=3.98
NW      flightCount=10292627, delayedFlightCount=214154, delayPercent=2.08
OH      flightCount=1464176, delayedFlightCount=47174, delayPercent=3.22
OO      flightCount=3090853, delayedFlightCount=65390, delayPercent=2.12
PA (1)  flightCount=316167, delayedFlightCount=3521, delayPercent=1.11
PI      flightCount=873957, delayedFlightCount=8910, delayPercent=1.02
PS      flightCount=83617, delayedFlightCount=1151, delayPercent=1.38
TW      flightCount=3757747, delayedFlightCount=69088, delayPercent=1.84
TZ      flightCount=208420, delayedFlightCount=2307, delayPercent=1.11
UA      flightCount=13299817, delayedFlightCount=290506, delayPercent=2.18
US      flightCount=14075530, delayedFlightCount=291650, delayPercent=2.07
WN      flightCount=15976022, delayedFlightCount=155053, delayPercent=0.97
XE      flightCount=2350309, delayedFlightCount=51991, delayPercent=2.21
YV      flightCount=854056, delayedFlightCount=30050, delayPercent=3.52
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

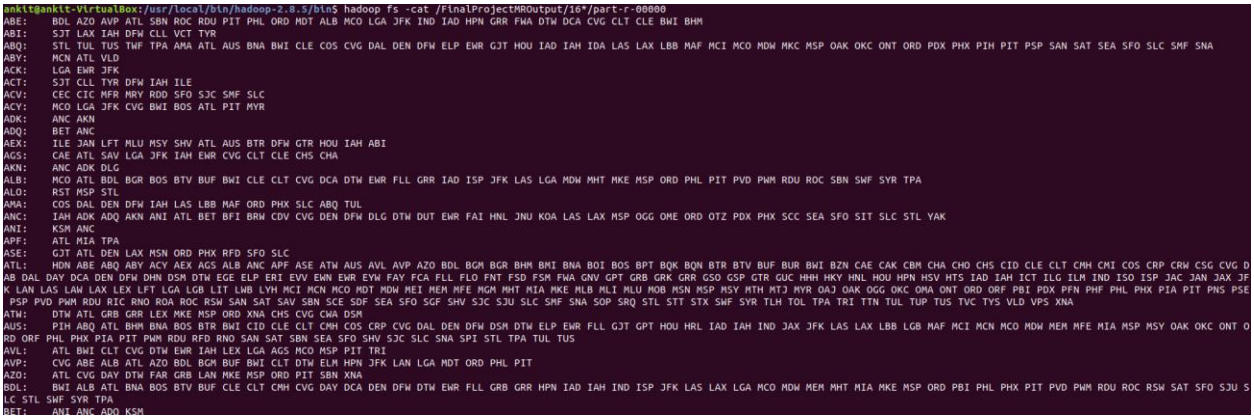
```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /FinalProjectMROutput/15.1-Cancelled_Ratio_Carriers_Name/part-r-00000
Pinnacle Airlines Inc.  flightCount=521059, delayedFlightCount=15039, delayPercent=2.89
American Airlines Inc.  flightCount=14984647, delayedFlightCount=286889, delayPercent=1.91
Aloha Airlines Inc.     flightCount=154381, delayedFlightCount=2837, delayPercent=1.84
Alaska Airlines Inc.    flightCount=2878021, delayedFlightCount=57121, delayPercent=1.98
JetBlue Airways flightCount=811341, delayedFlightCount=9281, delayPercent=1.14
Continental Air Lines Inc.      flightCount=8145788, delayedFlightCount=113064, delayPercent=1.39
Independence Air        flightCount=693047, delayedFlightCount=22176, delayPercent=3.20
Delta Air Lines Inc.    flightCount=16547870, delayedFlightCount=258382, delayPercent=1.56
Eastern Air Lines Inc.  flightCount=919785, delayedFlightCount=28702, delayPercent=3.12
Atlantic Southeast Airlines     flightCount=1697172, delayedFlightCount=48676, delayPercent=2.87
Frontier Airlines Inc.  flightCount=336958, delayedFlightCount=1778, delayPercent=0.53
AirTran Airways Corporation     flightCount=1265138, delayedFlightCount=12854, delayPercent=1.02
Hawaiian Airlines Inc.  flightCount=274265, delayedFlightCount=1329, delayPercent=0.48
America West Airlines Inc. (Merged with US Airways 9/05. Stopped reporting 10/07.)      flightCount=3636682, delayedFlightCount=55431, delayPercent=1.52
Midway Airlines Inc. (1)        flightCount=70622, delayedFlightCount=1342, delayPercent=1.90
American Eagle Airlines Inc.    flightCount=3954895, delayedFlightCount=157478, delayPercent=3.98
Northwest Airlines Inc. flightCount=10292627, delayedFlightCount=214154, delayPercent=2.08
Comair Inc.     flightCount=1464176, delayedFlightCount=47174, delayPercent=3.22
Skywest Airlines Inc.   flightCount=3090853, delayedFlightCount=65390, delayPercent=2.12
Pan American World Airways (1)  flightCount=316167, delayedFlightCount=3521, delayPercent=1.11
Piedmont Aviation Inc.  flightCount=873957, delayedFlightCount=8910, delayPercent=1.02
Pacific Southwest Airlines      flightCount=83617, delayedFlightCount=1151, delayPercent=1.38
Trans World Airways LLC flightCount=3757747, delayedFlightCount=69088, delayPercent=1.84
ATA Airlines d/b/a ATA  flightCount=208420, delayedFlightCount=2307, delayPercent=1.11
United Air Lines Inc.   flightCount=13299817, delayedFlightCount=290506, delayPercent=2.18
US Airways Inc. (Merged with America West 9/05. Reporting for both starting 10/07.)     flightCount=14075530, delayedFlightCount=291650, delayPercent=2.07
Southwest Airlines Co.  flightCount=15976022, delayedFlightCount=155053, delayPercent=0.97
Expressjet Airlines Inc.        flightCount=2350309, delayedFlightCount=51991, delayPercent=2.21
Mesa Airlines Inc.      flightCount=854056, delayedFlightCount=30050, delayPercent=3.52
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

The number of cancelled flights are very less for almost all the carriers less than 4%.

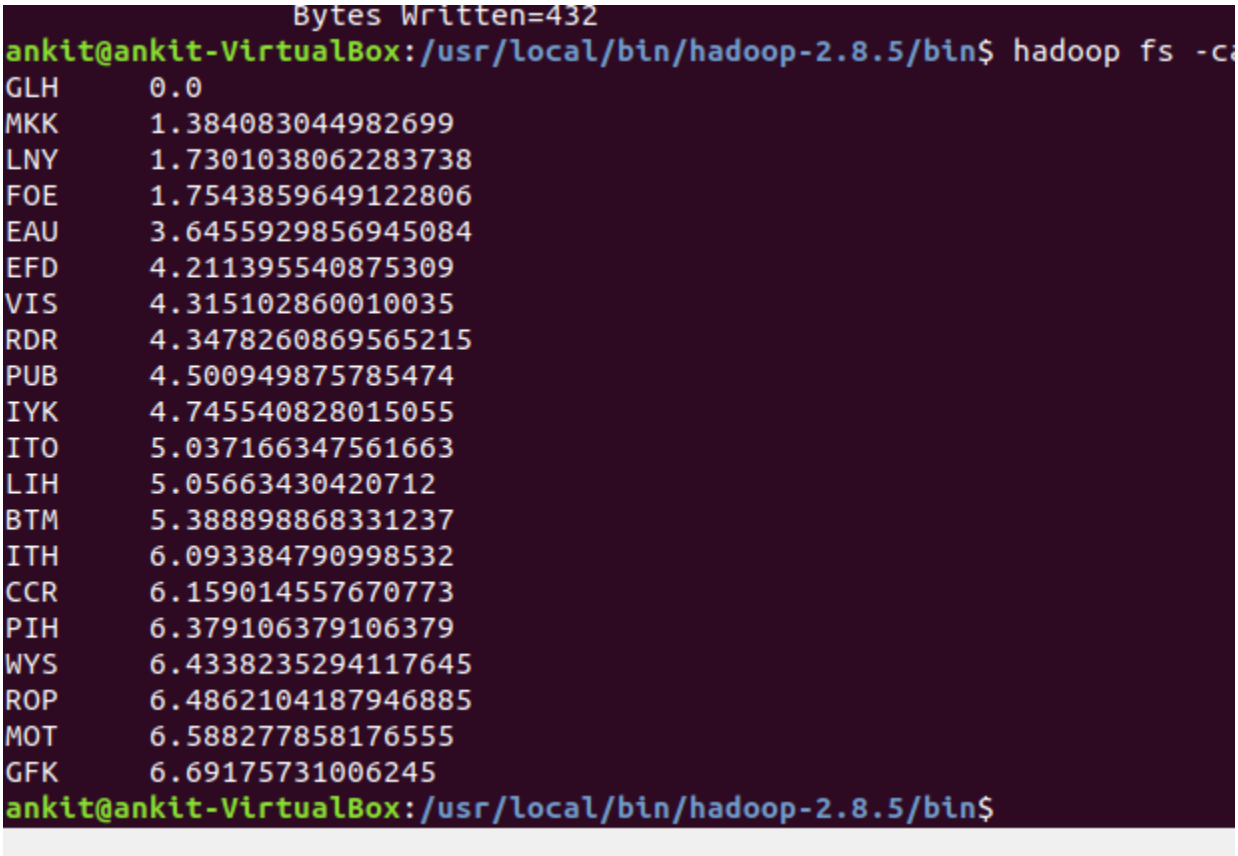
Among them best are **Frontier Airlines, Hawaiian Airlines** with **0.5%** cancelled flights and worst are **American Eagle Airlines, Mesa Airlines** with more than **3.5%** cancelled flights.

15- Inverted index for all source and destination

This data can help to search for all the destination stations from a particular source stations.



16- Top 20 best source station with least departure delayed flight percent



17- Top 20 best destination station with least arrival delayed flight percent

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /FinalProject
BFF     0.0
MKK     1.3888888888888888
LNY     2.0689655172413794
ROP     5.523710265763419
ITO     6.492665484134358
IYK     7.778510217534608
LIH     8.344797820398695
EAU     9.417889256980597
VIS     9.63673057517659
SMX     9.645635263612792
OXR     9.938676252907591
PUB     10.08503655079815
MIB     10.112359550561797
GCN     10.299999999999999
SPN     10.55402656455666
PMD     10.560859188544153
FLG     10.744087011567013
CCR     10.902510744175526
KOA     11.085274322107248
CLD     11.0986073990716
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

18- Delay groups- grouping amount of flights per delay groups

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ h
Between 1 hour and 2 hour       2.93
Between 15 abd 30 minutes       7.33
Between 30 minutes and 1 hour   4.55
Less than 15 Minutes    84.06
More than 2 hours       1.13
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

Between 1 hour and 2 hour	**2.93**
Between 15 and 30 minutes	**7.33**
Between 30 minutes and 1 hour	**4.55**
Less than 15 Minutes	**84.06**
More than 2 hours	**1.13**

19- Recommendation system- Best carrier for a source destination route

I first calculate average arrival delay and average departure delay for each source destination pair for each carrier.

After this I calculate root mean square value for average arrival delay and average departure delay.

$$\textbf{Rms} = \frac{\sqrt{AvgArrivalDelay^2 + AvgDepartureDelay^2}}{}$$

```
LEX-EWR XE      {arrDelay=56340, depDelay=50938, totalFlight=1856, rms=40.9230
LEX-FLL DL      {arrDelay=-15, depDelay=0, totalFlight=1, rms=15.0000}
LEX-FLL OH      {arrDelay=-297, depDelay=580, totalFlight=122, rms=5.3411}
LEX-HTS PI      {arrDelay=-28, depDelay=5, totalFlight=8, rms=3.5554}
LEX-IAH XE      {arrDelay=48057, depDelay=47054, totalFlight=3959, rms=16.9885
LEX-IND OO      {arrDelay=0, depDelay=0, totalFlight=1, rms=0.0000}
LEX-LGA EV      {arrDelay=-57, depDelay=-6, totalFlight=6, rms=9.5525}
LEX-LGA OH      {arrDelay=1594, depDelay=3356, totalFlight=1788, rms=2.0779}
LEX-MCN EV      {arrDelay=0, depDelay=0, totalFlight=1, rms=0.0000}
LEX-MCO EV      {arrDelay=-168, depDelay=-48, totalFlight=14, rms=12.4802}
LEX-MCO OH      {arrDelay=744, depDelay=1081, totalFlight=232, rms=5.6564}
LEX-MEM 9E      {arrDelay=330, depDelay=445, totalFlight=252, rms=2.1984}
LEX-MLI OO      {arrDelay=0, depDelay=0, totalFlight=1, rms=0.0000}
LEX-OKC EV      {arrDelay=0, depDelay=-10, totalFlight=1, rms=10.0000}
LEX-ORD DH      {arrDelay=27300, depDelay=24457, totalFlight=1504, rms=24.3703
LEX-ORD MQ      {arrDelay=19065, depDelay=16857, totalFlight=2478, rms=10.2698
LEX-ORD OO      {arrDelay=50949, depDelay=72095, totalFlight=5561, rms=15.8750
LEX-ORD PI      {arrDelay=9550, depDelay=6869, totalFlight=876, rms=13.4289}
LEX-ORD UA      {arrDelay=353, depDelay=114, totalFlight=30, rms=12.3650}
LEX-PIA OO      {arrDelay=0, depDelay=0, totalFlight=1, rms=0.0000}
LEX-PIT US      {arrDelay=72758, depDelay=70357, totalFlight=16190, rms=6.2515
LEX-ROA PI      {arrDelay=368, depDelay=324, totalFlight=61, rms=8.0378}
LEX-SDF DL      {arrDelay=28080, depDelay=19667, totalFlight=6962, rms=4.9242}
LEX-SDF PI      {arrDelay=-7, depDelay=-7, totalFlight=1, rms=9.8995}
LEX-SDF TW      {arrDelay=2875, depDelay=1595, totalFlight=1056, rms=3.1135}
LEX-SHV EV      {arrDelay=54, depDelay=0, totalFlight=1, rms=54.0000}
LEX-STL TW      {arrDelay=3247, depDelay=1377, totalFlight=838, rms=4.2087}
LEX-TPA OH      {arrDelay=-1284, depDelay=7, totalFlight=122, rms=10.5247}
LEX-TRI PI      {arrDelay=468, depDelay=329, totalFlight=247, rms=2.3161}
LEX-TYS OO      {arrDelay=128, depDelay=245, totalFlight=1, rms=276.4218}
LFT-ATL EV      {arrDelay=75828, depDelay=80410, totalFlight=5427, rms=20.3657
LFT-BTR CO      {arrDelay=-1608, depDelay=2955, totalFlight=1147, rms=2.9330}
LFT-DCA CO      {arrDelay=-73, depDelay=-2, totalFlight=16, rms=4.5642}
LFT-DFW EV      {arrDelay=2742, depDelay=8232, totalFlight=1137, rms=7.6312}
LFT-DFW MQ      {arrDelay=27465, depDelay=38463, totalFlight=5985, rms=7.8968}
LFT-GTR EV      {arrDelay=98, depDelay=0, totalFlight=1, rms=98.0000}
LFT-IAD CO      {arrDelay=-19, depDelay=-12, totalFlight=1, rms=22.4722}
LFT-IAH CO      {arrDelay=40260, depDelay=23538, totalFlight=6929, rms=6.7305}
LFT-IAH XE      {arrDelay=107932, depDelay=72526, totalFlight=17111, rms=7.599
LFT-MCN EV      {arrDelay=61, depDelay=0, totalFlight=1, rms=61.0000}
LFT-MEI EV      {arrDelay=654, depDelay=3560, totalFlight=370, rms=9.7826}
LFT-MGM EV      {arrDelay=126, depDelay=85, totalFlight=1, rms=151.9901}
LFT-MOB XE      {arrDelay=2, depDelay=13, totalFlight=1, rms=13.1529}
LFT-ORF CO      {arrDelay=-14, depDelay=-4, totalFlight=1, rms=14.5602}
LFT-PHL CO      {arrDelay=-7, depDelay=-2, totalFlight=3, rms=2.4267}
LFT-TXK EV      {arrDelay=-19, depDelay=-4, totalFlight=1, rms=19.4165}
LGA-ABE TW      {arrDelay=354, depDelay=306, totalFlight=110, rms=4.2538}
```

After this I sorted the carriers for all source destination pair in ascending order by RMS value.

It gives user a recommendation for choosing a carrier between a source destination pair with least arrival and/or departure delay.

```
File  Edit  View  Search  Terminal  Help
LEX-EWR : XE    40.9230
LEX-FLL : OH    5.3411
LEX-FLL : DL    15.0000
LEX-HTS : PI    3.5554
LEX-IAH : XE    16.9885
LEX-IND : OO    0.0000
LEX-LGA : OH    2.0779
LEX-LGA : EV    9.5525
LEX-MCN : EV    0.0000
LEX-MCO : OH    5.6564
LEX-MCO : EV    12.4802
LEX-MEM : 9E    2.1984
LEX-MLI : OO    0.0000
LEX-OKC : EV    10.0000
LEX-ORD : MQ    10.2698
LEX-ORD : UA    12.3650
LEX-ORD : PI    13.4289
LEX-ORD : OO    15.8750
LEX-ORD : DH    24.3703
LEX-PIA : OO    0.0000
LEX-PIT : US    6.2515
LEX-ROA : PI    8.0378
LEX-SDF : TW    3.1135
LEX-SDF : DL    4.9242
LEX-SDF : PI    9.8995
LEX-SHV : EV    54.0000
LEX-STL : TW    4.2087
LEX-TPA : OH    10.5247
LEX-TRI : PI    2.3161
LEX-TYS : OO    276.4218
LFT-ATL : EV    20.3657
LFT-BTR : CO    2.9330
LFT-DCA : CO    4.5642
LFT-DFW : EV    7.6312
LFT-DFW : MQ    7.8968
LFT-GTR : EV    98.0000
LFT-IAD : CO    22.4722
LFT-IAH : CO    6.7305
LFT-IAH : XE    7.5996
LFT-MCN : EV    61.0000
LFT-MEI : EV    9.7826
LFT-MGM : EV    151.9901
LFT-MOB : XE    13.1529
LFT-ORF : CO    14.5602
LFT-PHL : CO    2.4267
LFT-TXK : EV    19.4165
LGA-ABE : TW    4.2538
```

20- Average flying distance per carrier

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /FinalProjectMROutput/20*/part-r-00000
9E      AverageCountTuple{Total Flights=504652, Total Distance=227703482, Total Air Time=754313474, Average Distance=451.21, Average Air Time=1494.72}
AA      AverageCountTuple{Total Flights=14602906, Total Distance=1043051525, Total Air Time=546884063, Average Distance=71.43, Average Air Time=37.45}
AQ      AverageCountTuple{Total Flights=151507, Total Distance=51227324, Total Air Time=207101837, Average Distance=338.12, Average Air Time=1366.95}
AS      AverageCountTuple{Total Flights=2782295, Total Distance=2092368934, Total Air Time=-180060988, Average Distance=752.03, Average Air Time=-64.72}
B6      AverageCountTuple{Total Flights=799333, Total Distance=956056615, Total Air Time=1151338297, Average Distance=1196.07, Average Air Time=1440.37}
CO      AverageCountTuple{Total Flights=8004389, Total Distance=-1404710022, Total Air Time=-781563387, Average Distance=-175.49, Average Air Time=-97.64}
DH      AverageCountTuple{Total Flights=669687, Total Distance=251483885, Total Air Time=985613333, Average Distance=375.52, Average Air Time=1471.75}
DL      AverageCountTuple{Total Flights=16208118, Total Distance=-1298102575, Total Air Time=-1486892450, Average Distance=-80.09, Average Air Time=-91.74}
EA      AverageCountTuple{Total Flights=881538, Total Distance=536021400, Total Air Time=1339260606, Average Distance=608.05, Average Air Time=1519.23}
EV      AverageCountTuple{Total Flights=1645211, Total Distance=742277300, Total Air Time=-1825235102, Average Distance=451.17, Average Air Time=-1109.42}
F9      AverageCountTuple{Total Flights=334857, Total Distance=297757070, Total Air Time=509777308, Average Distance=889.21, Average Air Time=1522.37}
FL      AverageCountTuple{Total Flights=1249409, Total Distance=833241928, Total Air Time=1856768383, Average Distance=666.91, Average Air Time=1486.12}
HA      AverageCountTuple{Total Flights=272880, Total Distance=161404032, Total Air Time=381096796, Average Distance=591.48, Average Air Time=1396.57}
HP      AverageCountTuple{Total Flights=3572762, Total Distance=-1607195094, Total Air Time=831953310, Average Distance=-449.85, Average Air Time=232.86}
ML (1)  AverageCountTuple{Total Flights=69119, Total Distance=46873702, Total Air Time=104382535, Average Distance=678.16, Average Air Time=1510.19}
MQ      AverageCountTuple{Total Flights=3790050, Total Distance=1388700927, Total Air Time=1204517607, Average Distance=366.41, Average Air Time=317.81}
NW      AverageCountTuple{Total Flights=10047357, Total Distance=-1438027173, Total Air Time=-2112964318, Average Distance=-143.12, Average Air Time=-210.30}
OH      AverageCountTuple{Total Flights=1414180, Total Distance=664044565, Total Air Time=2083770904, Average Distance=469.56, Average Air Time=1473.48}
OO      AverageCountTuple{Total Flights=3020777, Total Distance=1175518881, Total Air Time=143697020, Average Distance=389.14, Average Air Time=47.57}
PA (1)  AverageCountTuple{Total Flights=310361, Total Distance=210996096, Total Air Time=484961259, Average Distance=679.84, Average Air Time=1562.57}
PI      AverageCountTuple{Total Flights=862209, Total Distance=327763510, Total Air Time=1294513853, Average Distance=380.14, Average Air Time=1501.39}
PS      AverageCountTuple{Total Flights=82293, Total Distance=29819392, Total Air Time=121771283, Average Distance=362.36, Average Air Time=1479.73}
TW      AverageCountTuple{Total Flights=3673505, Total Distance=-1614613432, Total Air Time=1236157660, Average Distance=-439.53, Average Air Time=336.51}
TZ      AverageCountTuple{Total Flights=206007, Total Distance=237096582, Total Air Time=304677183, Average Distance=1150.92, Average Air Time=1478.97}
UA      AverageCountTuple{Total Flights=12971049, Total Distance=-964391351, Total Air Time=-2137142143, Average Distance=-74.35, Average Air Time=-164.76}
US      AverageCountTuple{Total Flights=13741778, Total Distance=-637703914, Total Air Time=-844032545, Average Distance=-46.41, Average Air Time=-61.42}
WN      AverageCountTuple{Total Flights=15769974, Total Distance=-584533440, Total Air Time=2061208199, Average Distance=-37.07, Average Air Time=130.70}
XE      AverageCountTuple{Total Flights=2290684, Total Distance=1232713411, Total Air Time=-900268480, Average Distance=538.14, Average Air Time=-393.01}
YV      AverageCountTuple{Total Flights=822403, Total Distance=327982123, Total Air Time=1201040057, Average Distance=398.81, Average Air Time=1460.40}
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /FinalProjectMROutput/20.1*/part-r-00000
Pinnacle Airlines Inc.  AverageCountTuple{Total Flights=504652, Total Distance=227703482, Total Air Time=754313474, Average Distance=451.21, Average Air Time=1494.72}
American Airlines Inc.  AverageCountTuple{Total Flights=14602906, Total Distance=1043051525, Total Air Time=546884063, Average Distance=71.43, Average Air Time=37.45}
Aloha Airlines Inc.     AverageCountTuple{Total Flights=151507, Total Distance=51227324, Total Air Time=207101837, Average Distance=338.12, Average Air Time=1366.95}
Alaska Airlines Inc.    AverageCountTuple{Total Flights=2782295, Total Distance=2092368934, Total Air Time=-180060988, Average Distance=752.03, Average Air Time=-64.72}
JetBlue Airways AverageCountTuple{Total Flights=799333, Total Distance=956056615, Total Air Time=1151338297, Average Distance=1196.07, Average Air Time=1440.37}
Continental Air Lines Inc.      AverageCountTuple{Total Flights=8004389, Total Distance=-1404710022, Total Air Time=-781563387, Average Distance=-175.49, Average Air Time=-97.64}
Independence Air        AverageCountTuple{Total Flights=669687, Total Distance=251483885, Total Air Time=985613333, Average Distance=375.52, Average Air Time=1471.75}
Delta Air Lines Inc.    AverageCountTuple{Total Flights=16208118, Total Distance=-1298102575, Total Air Time=-1486892450, Average Distance=-80.09, Average Air Time=-91.74}
Eastern Air Lines Inc.  AverageCountTuple{Total Flights=881538, Total Distance=536021400, Total Air Time=1339260606, Average Distance=608.05, Average Air Time=1519.23}
Atlantic Southeast Airlines     AverageCountTuple{Total Flights=1645211, Total Distance=742277300, Total Air Time=-1825235102, Average Distance=451.17, Average Air Time=-1109.42}
Frontier Airlines Inc.  AverageCountTuple{Total Flights=334857, Total Distance=297757070, Total Air Time=509777308, Average Distance=889.21, Average Air Time=1522.37}
AirTran Airways Corporation     AverageCountTuple{Total Flights=1249409, Total Distance=833241928, Total Air Time=1856768383, Average Distance=666.91, Average Air Time=1486.12}
Hawaiian Airlines Inc.  AverageCountTuple{Total Flights=272880, Total Distance=161404032, Total Air Time=381096796, Average Distance=591.48, Average Air Time=1396.57}
America West Airlines Inc. (Merged with US Airways 9/05. Stopped reporting 10/07.)      AverageCountTuple{Total Flights=3572762, Total Distance=-1607195094, Total Air Time=831953310, Average Distance=-449.
85, Average Air Time=232.86}
Midway Airlines Inc. (1)        AverageCountTuple{Total Flights=69119, Total Distance=46873702, Total Air Time=104382535, Average Distance=678.16, Average Air Time=1510.19}
American Eagle Airlines Inc.    AverageCountTuple{Total Flights=3790050, Total Distance=1388700927, Total Air Time=1204517607, Average Distance=366.41, Average Air Time=317.81}
Northwest Airlines Inc. AverageCountTuple{Total Flights=10047357, Total Distance=-1438027173, Total Air Time=-2112964318, Average Distance=-143.12, Average Air Time=-210.30}
Comair Inc.     AverageCountTuple{Total Flights=1414180, Total Distance=664044565, Total Air Time=2083770904, Average Distance=469.56, Average Air Time=1473.48}
Skywest Airlines Inc.   AverageCountTuple{Total Flights=3020777, Total Distance=1175518881, Total Air Time=143697020, Average Distance=389.14, Average Air Time=47.57}
Pan American World Airways (1)  AverageCountTuple{Total Flights=310361, Total Distance=210996096, Total Air Time=484961259, Average Distance=679.84, Average Air Time=1562.57}
Piedmont Aviation Inc.  AverageCountTuple{Total Flights=862209, Total Distance=327763510, Total Air Time=1294513853, Average Distance=380.14, Average Air Time=1501.39}
Pacific Southwest Airlines      AverageCountTuple{Total Flights=82293, Total Distance=29819392, Total Air Time=121771283, Average Distance=362.36, Average Air Time=1479.73}
Trans World Airways LLC AverageCountTuple{Total Flights=3673505, Total Distance=-1614613432, Total Air Time=1236157660, Average Distance=-439.53, Average Air Time=336.51}
ATA Airlines d/b/a ATA  AverageCountTuple{Total Flights=206007, Total Distance=237096582, Total Air Time=304677183, Average Distance=1150.92, Average Air Time=1478.97}
United Air Lines Inc.   AverageCountTuple{Total Flights=12971049, Total Distance=-964391351, Total Air Time=-2137142143, Average Distance=-74.35, Average Air Time=-164.76}
US Airways Inc. (Merged with America West 9/05. Reporting for both starting 10/07.)     AverageCountTuple{Total Flights=13741778, Total Distance=-637703914, Total Air Time=-844032545, Average Distance=-46.
41, Average Air Time=-61.42}
Southwest Airlines Co.  AverageCountTuple{Total Flights=15769974, Total Distance=-584533440, Total Air Time=2061208199, Average Distance=-37.07, Average Air Time=130.70}
Expressjet Airlines Inc.        AverageCountTuple{Total Flights=2290684, Total Distance=1232713411, Total Air Time=-900268480, Average Distance=538.14, Average Air Time=-393.01}
Mesa Airlines Inc.      AverageCountTuple{Total Flights=822403, Total Distance=327982123, Total Air Time=1201040057, Average Distance=398.81, Average Air Time=1460.40}
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

21- Using partitioning pattern on the basis of year

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -ls  /FinalProjectMROutput/23*
Found 23 items
-rw-r--r--   1 ankit supergroup          0 2019-08-14 23:50 /FinalProjectMROutput/23-Partitioning_Pattern_Year/_SUCCESS
-rw-r--r--   1 ankit supergroup   21056356 2019-08-14 23:48 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00000
-rw-r--r--   1 ankit supergroup   83522592 2019-08-14 23:48 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00001
-rw-r--r--   1 ankit supergroup   80976532 2019-08-14 23:48 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00002
-rw-r--r--   1 ankit supergroup   84666512 2019-08-14 23:48 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00003
-rw-r--r--   1 ankit supergroup   81772204 2019-08-14 23:48 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00004
-rw-r--r--   1 ankit supergroup   81474512 2019-08-14 23:49 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00005
-rw-r--r--   1 ankit supergroup   81128016 2019-08-14 23:49 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00006
-rw-r--r--   1 ankit supergroup   82880768 2019-08-14 23:49 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00007
-rw-r--r--   1 ankit supergroup   85238960 2019-08-14 23:49 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00008
-rw-r--r--   1 ankit supergroup   85631728 2019-08-14 23:49 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00009
-rw-r--r--   1 ankit supergroup   86589488 2019-08-14 23:49 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00010
-rw-r--r--   1 ankit supergroup   86155536 2019-08-14 23:49 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00011
-rw-r--r--   1 ankit supergroup   88446144 2019-08-14 23:49 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00012
-rw-r--r--   1 ankit supergroup   90928752 2019-08-14 23:49 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00013
-rw-r--r--   1 ankit supergroup   95484480 2019-08-14 23:49 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00014
-rw-r--r--   1 ankit supergroup   84341744 2019-08-14 23:50 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00015
-rw-r--r--   1 ankit supergroup  103816640 2019-08-14 23:50 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00016
-rw-r--r--   1 ankit supergroup  114068320 2019-08-14 23:50 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00017
-rw-r--r--   1 ankit supergroup  114249536 2019-08-14 23:50 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00018
-rw-r--r--   1 ankit supergroup  114270752 2019-08-14 23:50 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00019
-rw-r--r--   1 ankit supergroup  119251440 2019-08-14 23:50 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00020
-rw-r--r--   1 ankit supergroup  112155648 2019-08-14 23:50 /FinalProjectMROutput/23-Partitioning_Pattern_Year/part-r-00021
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

2. Analysis of Flight Data using Apache PIG on Hadoop

Analysis 1: Top 20 cities by total volume of flights

What are the busiest cities by total flight traffic? For each airport code I computed the number of inbound, outbound and all flights.

```
-- First, we load the raw data from a test dataset
RAW_DATA = LOAD '/flight-data' USING PigStorage(',') AS
        (year: int, month: int, day: int, dow: int,
        dtime: int, sdtime: int, arrtime: int, satime: int,
        carrier: chararray, fn: int, tn: chararray,
        etime: int, setime: int, airtime: int,
        adelay: int, ddelay: int,
        scode: chararray, dcode: chararray, dist: int,
        tintime: int, touttime: int,
        cancel: chararray, cancelcode: chararray, diverted: int,
        cdelay: int, wdelay: int, ndelay: int, sdelay: int, latedelay: int);


------------------------------------------------------------
-- INBOUND TRAFFIC, PER IATA AIRPORT CODE, PER MONTH, TOP k
------------------------------------------------------------
-- project, to get rid of unused fields: only month and destination ID
INBOUND = FOREACH RAW_DATA GENERATE month AS m, dcode AS d;
-- group by month, then ID (sorted)
GROUP_INBOUND = GROUP INBOUND BY (m,d);
-- aggregate over the group, flatten group, such that output relation has 3 fields
COUNT_INBOUND = FOREACH GROUP_INBOUND GENERATE FLATTEN(group), COUNT(INBOUND) AS count;
-- aggregate over months only
GROUP_COUNT_INBOUND = GROUP COUNT_INBOUND BY m;
-- now apply UDF to compute top k (k=20)
topMonthlyInbound = FOREACH GROUP_COUNT_INBOUND {
    result = TOP(20, 2, COUNT_INBOUND);
    GENERATE FLATTEN(result);
}

--dump topMonthlyInbound
STORE topMonthlyInbound INTO '/home/ankit/Downloads/output/INBOUND-TOP' USING PigStorage(',');

------------------------------------------------------------
--  OUTBOUND TRAFFIC, PER IATA AIRPORT CODE, PER MONTH, TOP k
------------------------------------------------------------
OUTBOUND = FOREACH RAW_DATA GENERATE month AS m, scode AS s;
GROUP_OUTBOUND = GROUP OUTBOUND BY (m,s);
COUNT_OUTBOUND = FOREACH GROUP_OUTBOUND GENERATE FLATTEN(group), COUNT(OUTBOUND) AS count;
GROUP_COUNT_OUTBOUND = GROUP COUNT_OUTBOUND BY m;
topMonthlyOutbound = FOREACH GROUP_COUNT_OUTBOUND {
    result = TOP(20, 2, COUNT_OUTBOUND);
    GENERATE FLATTEN(result);
}

--dump topMonthlyOutbound
STORE topMonthlyOutbound INTO '/home/ankit/Downloads/output/OUTBOUND-TOP' USING PigStorage(',');
```

OUTPUT: INBOUNT_TOP

```
1,CVG,8669
1,BWI,8889
1,BOS,9726
1,CLT,10745
1,SFO,11560
1,MCO,11063
1,JFK,10039
1,DTW,14381
1,EWR,12470
1,MSP,11825
1,LAS,15292
1,SLC,12364
1,DEN,19482
1,LGA,10298
1,PHX,17693
1,ATL,33881
1,LAX,18964
1,DFW,23874
1,IAH,15527
1,ORD,29936
2,SEA,7977
2,BWI,8214
2,JFK,9545
2,BOS,9430
2,CLT,9996
2,MSP,10848
2,DTW,13387
2,MCO,10705
2,LGA,9741
2,SFO,10817
2,ORD,27987
2,DFW,22231
2,LAS,14282
2,IAH,14834
2,DEN,18667
2,LAX,17491
2,ATL,32357
2,SLC,11681
2,EWR,11616
2,PHX,16595
3,SEA,8729
3,JFK,10349
3,BWI,8858
```

OUTPUT: OUTBOUND TOP

*Q1.pig

```
1,CVG,8659
1,BWI,8883
1,LGA,10300
1,JFK,10023
1,BOS,9717
1,CLT,10752
1,MSP,11810
1,MCO,11070
1,SLC,12401
1,LAX,18945
1,DTW,14373
1,SFO,11573
1,EWR,12467
1,LAS,15292
1,DEN,19477
1,PHX,17695
1,IAH,15534
1,DFW,23861
1,ORD,29936
1,ATL,33906
2,SEA,7978
2,BWI,8217
2,JFK,9555
2,BOS,9426
2,SFO,10815
2,LGA,9750
2,CLT,9995
2,EWR,11614
2,MCO,10701
2,MSP,10853
2,ATL,32378
2,PHX,16602
2,IAH,14839
2,LAS,14280
2,DTW,13397
2,ORD,27972
2,DEN,18660
2,LAX,17482
2,SLC,11688
2,DFW,22223
3,SEA,8735
3,BWI,8860
```

```
-------------------------------------------------------------
-- TOTAL TRAFFIC, PER IATA AIRPORT CODE, PER MONTH, TOP k
-------------------------------------------------------------
UNION_TRAFFIC = UNION COUNT_INBOUND, COUNT_OUTBOUND;
GROUP_UNION_TRAFFIC = GROUP UNION_TRAFFIC BY (m,d);
TOTAL_TRAFFIC = FOREACH GROUP_UNION_TRAFFIC GENERATE FLATTEN(group) AS (m,code), SUM(UNION_TRAFFIC.count) AS total;
TOTAL_MONTHLY = GROUP TOTAL_TRAFFIC BY m;

topMonthlyTraffic = FOREACH TOTAL_MONTHLY {
    result = TOP(20, 2, TOTAL_TRAFFIC);
    GENERATE FLATTEN(result) AS (month, iata, traffic);
}


STORE topMonthlyTraffic INTO '/home/ankit/Downloads/output/MONTHLY-TRAFFIC-TOP/' USING PigStorage(',');

explain -brief -dot -out ./ topMonthlyTraffic
```

Output Monthly Traffic Top

```
1,CVG,17328
1,BWI,17772
1,BOS,19443
1,CLT,21497
1,SFO,23133
1,MCO,22133
1,JFK,20062
1,DTW,28754
1,EWR,24937
1,MSP,23635
1,LAS,30584
1,SLC,24765
1,DEN,38959
1,LGA,20598
1,PHX,35388
1,ATL,67787
1,LAX,37909
1,DFW,47735
1,IAH,31061
1,ORD,59872
2,SEA,15955
2,BWI,16431
2,BOS,18856
2,JFK,19100
2,CLT,19991
2,MSP,21701
2,IAH,29673
2,SLC,23369
2,LGA,19491
2,SFO,21632
2,MCO,21406
2,DEN,37327
2,LAS,28562
2,PHX,33197
2,DFW,44454
2,LAX,34973
2,ATL,64735
2,EWR,23230
2,DTW,26784
2,ORD,55959
3,SEA,17464
3,JFK,20705
3,BWI,17718
3,MCO,24298
```

Analysis 2: Carrier Popularity

Computing the volume -- total flights -- over each year, by carrier. The carriers are ranked by their median volume (over the 10-year span).

```
-- First, we load the raw data from a test dataset
RAW_DATA = LOAD '/flight-data' USING PigStorage(',') AS
        (year: int, month: int, day: int, dow: int,
        dtime: int, sdtime: int, arrtime: int, satime: int,
        carrier: chararray, fn: int, tn: chararray,
        etime: int, setime: int, airtime: int,
        adelay: int, ddelay: int,
        scode: chararray, dcode: chararray, dist: int,
        tintime: int, touttime: int,
        cancel: chararray, cancelcode: chararray, diverted: int,
        cdelay: int, wdelay: int, ndelay: int, sdelay: int, latedelay: int);

CARRIER_DATA = FOREACH RAW_DATA GENERATE month AS m, carrier AS cname;
GROUP_CARRIERS = GROUP CARRIER_DATA BY (m,cname);
COUNT_CARRIERS = FOREACH GROUP_CARRIERS GENERATE FLATTEN(group), (COUNT(CARRIER_DATA)) AS popularity;

STORE COUNT_CARRIERS INTO '/PIG-OUTPUT-FULL/Q2/COUNT_CARRIERS' USING PigStorage(',');
--dump COUNT_CARRIERS
```

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8
1,OH,128287
2,B6,60613
2,EA,78969
2,PA (1),24608
3,EV,144106
3,PS,13054
3,XE,202815
4,FL,103851
5,AS,239718
5,PI,79887
6,AA,1234356
6,DL,1359922
6,TW,309144
8,9E,44296
8,AQ,14326
8,TZ,18157
8,ML (1),6565
9,CO,642141
9,HP,293381
9,MQ,329669
9,YV,67564
10,OO,264649
10,WN,1408459
11,US,1170795
12,DH,48113
12,HA,27482
12,NW,877933
12,UA,1129620
,UniqueCarrier,0
```

Analysis 3: Proportion of Flights Delayed

A flight is delayed if the delay is greater than 15 minutes. I am calculating the fraction of delayed flights per different time limits (hour, day, week, month, year).

```
-- First, we load the raw data from a test dataset
RAW_DATA = LOAD '/flight-data' USING PigStorage(',') AS
        (year: int, month: int, day: int, dow: int,
        dtime: int, sdtime: int, arrtime: int, satime: int,
        carrier: chararray, fn: int, tn: chararray,
        etime: int, setime: int, airtime: int,
        adelay: int, ddelay: int,
        scode: chararray, dcode: chararray, dist: int,
        tintime: int, touttime: int,
        cancel: chararray, cancelcode: chararray, diverted: int,
        cdelay: int, wdelay: int, ndelay: int, sdelay: int, latedelay: int);

-- A flight is delayed if the delay is greater than 15 minutes.
-- delay = arrival time - scheduled arrival time
-- Compute the fraction of delayed flights per different time
-- granularities (hour, day, week, month, year).


-- example: let's focus on a month
-- Foreach month:
-- compute the total number of flights
-- compute delay relation: only those filght with delay > 15 min appear here
-- compute the total number of delayed flights
-- output relation: month, ratio delayed/total


-- project, to get rid of unused fields
A = FOREACH RAW_DATA GENERATE day AS d, dow AS dow, month AS m, (int)(arrtime-satime) AS delay;

-- group by month
B = GROUP A BY (m,dow);

COUNT_TOTAL = FOREACH B {
        C = FILTER A BY (delay >= 15); -- only keep tuples with a delay >= than 15 minutes
        GENERATE group, COUNT(A) AS tot, COUNT(C) AS del, (float) COUNT(C)/COUNT(A) AS frac;
}

dump COUNT_TOTAL;
```

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /PIG*/Q3/COUNT*/part-r-00000
(1,1),80807,20100,0.24874082
(1,2),97298,25886,0.26604864
(1,3),100080,23790,0.23770984
(1,4),102043,30424,0.2981488
(1,5),81940,21736,0.26526728
(1,6),67178,14830,0.2207568
(1,7),76419,19734,0.2582342
(2,1),81504,23688,0.29063603
(2,2),79700,25176,0.31588456
(2,3),80587,22104,0.2742874
(2,4),82158,21580,0.26266462
(2,5),102726,34337,0.3342581
(2,6),66462,14416,0.2169059
(2,7),76099,20952,0.27532557
(3,1),103210,28609,0.27719215
(3,2),81159,19502,0.24029374
(3,3),82307,19835,0.240988
(3,4),82831,19446,0.23476718
(3,5),82936,26168,0.3155204
(3,6),86153,23543,0.27326965
(3,7),97494,27022,0.27716577
(4,1),82463,17261,0.20931812
(4,2),100785,18871,0.18724017
(4,3),102586,18098,0.17641783
(4,4),82799,19901,0.24035314
(4,5),82964,24060,0.2900053
(4,6),68304,14171,0.20746955
(4,7),78225,17847,0.22814956
(5,1),80626,15997,0.19840994
(5,2),79884,16203,0.20283161
(5,3),81264,17249,0.21225882
(5,4),102572,22649,0.22081074
(5,5),102878,28756,0.27951553
(5,6),84493,14940,0.17681938
(5,7),74576,13140,0.1761961
(6,1),104168,26853,0.2577855
(6,2),82160,20815,0.25334713
(6,3),82902,22303,0.26902848
(6,4),83617,25746,0.3079039
(6,5),83930,25300,0.30144167
(6,6),72322,18197,0.25161085
```

Analysis 4: Carrier Delays

Calculating the proportion of delayed flights by carrier, ranked by carrier, at different time (hour, day, week, month year). Again, a flight is delayed if the delay is greater than 15 minutes.

```
-- First, we load the raw data from a test dataset
RAW_DATA = LOAD '/flight-data' USING PigStorage(',') AS
        (year: int, month: int, day: int, dow: int,
        dtime: int, sdtime: int, arrtime: int, satime: int,
        carrier: chararray, fn: int, tn: chararray,
        etime: int, setime: int, airtime: int,
        adelay: int, ddelay: int,
        scode: chararray, dcode: chararray, dist: int,
        tintime: int, touttime: int,
        cancel: chararray, cancelcode: chararray, diverted: int,
        cdelay: int, wdelay: int, ndelay: int, sdelay: int, latedelay: int);

-- A flight is delayed if the delay is greater than 15 minutes.
-- delay = arrival time - scheduled arrival time
-- Compute the fraction of delayed flights per different time
-- granularities (hour, day, week, month, year).


-- example: let's focus on a month
-- Foreach month:
-- compute the total number of flights
-- compute delay relation: only those filght with delay > 15 min appear here
-- compute the total number of delayed flights
-- output relation: month, ratio delayed/total


-- project, to get rid of unused fields
A = FOREACH RAW_DATA GENERATE month AS m, carrier, (int)(arrtime-satime) AS delay;

-- group by carrier
B = GROUP A BY carrier;

COUNT_TOTAL = FOREACH B {
        C = FILTER A BY (delay >= 15); -- only keep tuples with a delay >= than 15 minutes
        GENERATE group, COUNT(A) AS tot, COUNT(C) AS del, (float) COUNT(C)/COUNT(A) AS frac;
}

dump COUNT_TOTAL;
```

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop f
9E,262208,51877,0.19784674
AA,604885,164384,0.27176076
AQ,7800,446,0.05717949
AS,151102,32453,0.21477544
B6,196091,48042,0.2449985
CO,298455,75372,0.2525406
DL,451931,108304,0.23964721
EV,280575,71399,0.25447384
F9,95762,21403,0.22350201
FL,261684,60872,0.23261644
HA,61826,6303,0.101947404
MQ,490693,118471,0.2414361
NW,347652,82584,0.23754789
OH,197607,56372,0.28527328
OO,567159,114613,0.20208266
UA,449515,120020,0.2669989
US,453589,89773,0.19791706
WN,1201754,226880,0.18879072
XE,374510,91481,0.2442685
YV,254930,61532,0.24136822
UniqueCarrier,0,0,
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

Analysis 5: Routes that were most busy

The approach is to create a frequency table for the unordered pair (m,n) where m and n are distinct airport codes which will help in finding the routes that are more busy.

```
-- First, we load the raw data from a test dataset
RAW_DATA = LOAD '/flight-data' USING PigStorage(',') AS
        (year: int, month: int, day: int, dow: int,
        dtime: int, sdtime: int, arrtime: int, satime: int,
        carrier: chararray, fn: int, tn: chararray,
        etime: int, setime: int, airtime: int,
        adelay: int, ddelay: int,
        scode: chararray, dcode: chararray, dist: int,
        tintime: int, touttime: int,
        cancel: chararray, cancelcode: chararray, diverted: int,
        cdelay: int, wdelay: int, ndelay: int, sdelay: int, latedelay: int);

-----------------------------------------------------------------------------
-- APPROACH 1:
-- The idea is to build a frequency table for the unordered pair (i,j) where i and j are distinct airport codes
-- This means we are not interested in any relative counts. In APPROACH 2 we will see how to do this

-- QUESTION: what about the shuffle key space? Is it balanced? How can it be made balanced?
-----------------------------------------------------------------------------


-- project to get rid of unused fields
A = FOREACH RAW_DATA GENERATE scode AS s, dcode AS d;

-- group by (s,d) pair
B = GROUP A by (s,d);

COUNT = FOREACH B GENERATE group, COUNT(A);

dump COUNT;
```

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/
(ABE,ATL),853
(ABE,BHM),1
(ABE,CLE),805
(ABE,CLT),465
(ABE,CVG),247
(ABE,DTW),997
(ABE,JFK),3
(ABE,LGA),9
(ABE,ORD),1425
(ABE,PHL),2
(ABI,DFW),2660
(ABQ,AMA),368
(ABQ,ATL),1067
(ABQ,AUS),433
(ABQ,BWI),546
(ABQ,CLE),12
(ABQ,CVG),260
(ABQ,DAL),3078
(ABQ,DEN),4318
(ABQ,DFW),2888
(ABQ,ELP),799
(ABQ,EWR),167
(ABQ,HOU),1011
(ABQ,IAD),365
(ABQ,IAH),2261
(ABQ,LAS),2402
(ABQ,LAX),2395
(ABQ,LBB),366
(ABQ,MAF),367
(ABQ,MCI),670
(ABQ,MCO),730
(ABQ,MDW),730
(ABQ,MSP),677
(ABQ,OAK),1016
(ABQ,OKC),229
(ABQ,ONT),446
(ABQ,ORD),751
(ABQ,PDX),366
(ABQ,PHX),5265
(ABQ,SAN),1034
(ABQ,SAT),424
(ABQ,SEA),721
```

3. Analysis of Flight Data using Apache HIVE on Hadoop

Creating schema for flight data

create schema AirlineSchema;

use AirlineSchema;

```
hive> create schema AirlineSchema;
OK
Time taken: 1.18 seconds
hive> use AirlineSchema;
OK
Time taken: 0.069 seconds
hive>
```

Creating table to store flight data

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, TailNum String, Ac
tualElapsedTime 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 ',';
OK
Time taken: 0.808 seconds
hive>
```

Set some hive properties

SET hive.exec.dynamic.partition = true;

SET hive.exec.dynamic.partition.mode = nonstrict;

Load flight data from HDFS into table

LOAD DATA INPATH '/flight-data' OVERWRITE INTO TABLE flightData;

```
hive> LOAD DATA INPATH '/flight-data' OVERWRITE INTO TABLE flightdata;
Loading data to table airlineschema.flightdata
OK
Time taken: 0.237 seconds
hive>
```

Create table and load airports data from HDFS

create external table airports (Iata String, aiport String, city String, state String, country String, lat String, longi String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;

LOAD DATA INPATH '/supp-data/airports.csv' OVERWRITE INTO TABLE airports;

```
hive> create external table airports (Iata String, aiport String, city String, state String, country String, lat String, longi String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
OK
Time taken: 0.124 seconds
hive> LOAD DATA INPATH '/supp-data/airports.csv' OVERWRITE INTO TABLE airports;
Loading data to table airlineschema.airports
OK
Time taken: 0.455 seconds
hive>
```

Create table and load carrier's data from HDFS

create external table carriers (Code String, Description String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;

LOAD DATA INPATH '/supp-data/carriers.csv' OVERWRITE INTO TABLE carriers;

```
hive> create external table carriers (Code String, Description String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
OK
Time taken: 0.068 seconds
hive> LOAD DATA INPATH '/supp-data/carriers.csv' OVERWRITE INTO TABLE carriers;
Loading data to table airlineschema.carriers
OK
Time taken: 0.174 seconds
hive>
```

Now, All data is loaded. So, we can proceed to analysis.

1: FLIGHTS THAT TRAVELLED LESS THAN OR MORE THAN 500 AIRTIME

INSERT OVERWRITE DIRECTORY '/HiveMROutput/1.1' select count(*) from flightData where AirTime > 500;

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /HiveMROutput/1.1/000000_0
30711
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

INSERT OVERWRITE DIRECTORY '/HiveMROutput/1.2' select count(*) from flightData where AirTime >= 500;

```
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /HiveMROutput/1.2/000000_0
31632
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$
```

2. COUNT OF ALL THE FLIGHTS THAT WERE ON TIME WHILE ARRIVING AND DEPARTURE

INSERT OVERWRITE DIRECTORY '/HiveMROutput/2' select Year,Month,DayofMonth,Origin,Dest,AirTime,Distance,TaxiIn,TaxiOut from flightData where DepTime<=CRSDepTime and ArrTime<=CRSArrTime;

```
1988  1ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ hadoop fs -cat /HiveMROutput/2/000000_0|head
1987 10 18 SAN SFO N 47 N N
1987 10 21 SAN SFO N 47 N N
1987 10  5 BUR OAK N 25 N N
1987 10  6 BUR OAK N 25 N N
1987 10 11 BUR OAK N 25 N N
1987 10  7 OAK BUR N 25 N N
1987 10 10 LAX SFO N 37 N N
1987 10 17 LAX SFO N 37 N N
1987 10 18 LAX SFO N 37 N N
1987 10 22 LAX SFO N 37 N N
cat: Unable to write to output stream.
ankit@ankit-VirtualBox:/usr/local/bin/hadoop-2.8.5/bin$ ^C
```

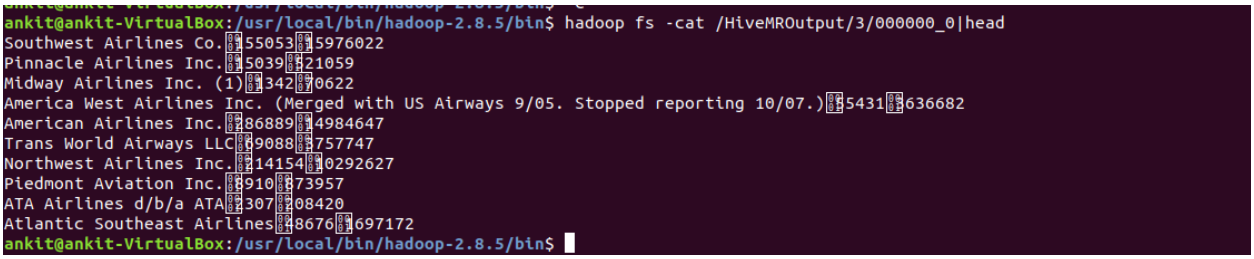
3: COUNT OF ALL THE FLIGHTS THAT TOOK MORE THAN 30 MINS TO DEP AND ARRIVAL DELAY

select count(*) from flightData where ArrDelay + DepDelay >30;

```
MapReduce Total cumulative CPU time: 8 minutes 1 seconds 810 msec
Ended Job = job_1565878533511_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 45  Reduce: 1   Cumulative CPU: 481.81 sec   HDFS Read: 12030058986 HDFS Write: 108 SUCCESS
Total MapReduce CPU Time Spent: 8 minutes 1 seconds 810 msec
OK
20148201
Time taken: 526.069 seconds, Fetched: 1 row(s)
hive>
```

4: COUNT OF FLIGHTS FOR EACH CARRIER

INSERT OVERWRITE DIRECTORY '/HiveMROutput/3' Select carriers.description, uniqCount.countCancelled, uniqCount.countCarrier from (Select UniqueCarrier, sum(cancelled) as countCancelled, count(*) as countCarrier from flightData group by UniqueCarrier) AS uniqCount, carriers where carriers.code = uniqCount.UniqueCarrier;



4. References

1. https://learning.oreilly.com/library/view/mapreduce-design-patterns/9781449341954/
2. https://gitlab.eurecom.fr/yonghui.feng/clouds-lab
3. https://learning.oreilly.com/library/view/data-algorithms/9781491906170/ch01.html
4. http://cs229.stanford.edu/proj2013/MathurNagaoNg-PredictingFlightOnTimePerformance.pdf

5. APPENDIX

The code of this project can be found at GitHub repository for this project at

https://github.com/ankit08015/Engg-Of-Big-Data/tree/master/Final_Project

1. Airport Count

```
package hadoop.project.airport_count;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"airportcount");
        job.setJarByClass(AirportMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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




        job.setMapperClass(AirportMapper.class);
        job.setCombinerClass(AirportReducer.class);
        job.setReducerClass(AirportReducer.class);

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

```
        job.setOutputValueClass(IntWritable.class);

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}
```

```
package hadoop.project.airport_count;

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

import java.io.IOException;

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

    //    hadoop datatype
    Text word = new Text();
    IntWritable one = new IntWritable(1);

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

        String line = value.toString();
        String[] tokens= line.split(",");
        if(tokens[0].equals("Year"))return;

        String src = tokens[16];
        word.set(src);
        context.write(word,one);
    }


}
```

```
package hadoop.project.airport_count;

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

import java.io.IOException;

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

    // just like in mongoDB values is iterable
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
```

```
throws IOException, InterruptedException {

        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
            // can we use this--  Integer.parseInt(v.toString());
        }

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

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }

}
```

2- Average Distance Carrier

```
package hadoop.project.avg_distance_carrier;

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

import java.io.IOException;

public class AverageCombiner extends Reducer<Text, AverageCountTuple, Text,
AverageCountTuple> {

    private AverageCountTuple tuple = new AverageCountTuple();

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

        int totalFlight=0;
        int totalDist=0;
        int totalAirTime =0;



        for(AverageCountTuple dt: values){
            totalFlight += dt.getFlightCount();
            totalDist += dt.getDistCount();
            totalAirTime += dt.getAirTime();

        }

        tuple.setAirTime(totalAirTime);
        tuple.setDistCount(totalDist);
        tuple.setFlightCount(totalFlight);

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

```
    }
}


package hadoop.project.avg_distance_carrier;

import org.apache.hadoop.io.Writable;

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

public class AverageCountTuple implements Writable {

    private int flightCount=0;
    private int distCount=0;
    private int airTime=0;
    private double averageDist =0.0;
    private double averageAirTime=0.0;

    public int getAirTime() {
        return airTime;
    }

    public void setAirTime(int airTime) {
        this.airTime = airTime;
    }

    public int getFlightCount() {
        return flightCount;
    }

    public void setFlightCount(int flightCount) {
        this.flightCount = flightCount;
    }

    public int getDistCount() {
        return distCount;
    }

    public void setDistCount(int distCount) {
        this.distCount = distCount;
    }

    public double getAverageDist() {
        return averageDist;
    }

    public void setAverageDist(double averageDist) {
        this.averageDist = averageDist;
    }

    public double getAverageAirTime() {
        return averageAirTime;
    }
```

```
    public void setAverageAirTime(double averageAirTime) {
        this.averageAirTime = averageAirTime;
    }

    @Override
    public String toString() {
        return "AverageCountTuple{" +
                "Total Flights=" + flightCount +
                ", Total Distance=" + distCount +
                ", Total Air Time=" + airTime +
                ", Average Distance=" + String.format("%.2f", averageDist) +
                ", Average Air Time=" + String.format("%.2f", averageAirTime)+
                '}';
    }

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeInt(flightCount);
        dataOutput.writeInt(distCount);
        dataOutput.writeInt(airTime);
        dataOutput.writeDouble(averageDist);
        dataOutput.writeDouble(averageAirTime);

    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {

        flightCount = dataInput.readInt();
        distCount = dataInput.readInt();
        airTime = dataInput.readInt();
        averageDist = dataInput.readDouble();
        averageAirTime = dataInput.readDouble();

    }
}


package hadoop.project.avg_distance_carrier;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

public class AverageMain {

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

```
ClassNotFoundException{


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(AverageMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(AverageMapper.class);
        job.setCombinerClass(AverageCombiner.class);
        job.setReducerClass(AverageReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}


package hadoop.project.avg_distance_carrier;


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

import java.io.IOException;


public class AverageMapper extends Mapper<Object, Text, Text, AverageCountTuple> {

    private AverageCountTuple tuple = new AverageCountTuple();

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

```
        if(tokens[0].equals("Year"))return;

        String carrier = tokens[8];
        int dist=0;
        int flightTime =0;

        try {
            dist = Integer.parseInt(tokens[18]);

            flightTime = Integer.parseInt(tokens[6]);
        } catch (Exception e){
            return;
        }
        tuple.setFlightCount(1);
        tuple.setDistCount(dist);
        tuple.setAirTime(flightTime);


        context.write(new Text(carrier),tuple);
    }
}


package hadoop.project.avg_distance_carrier;

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

import java.io.IOException;

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

private AverageCountTuple tuple = new AverageCountTuple();

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

        int totalFlight=0;
        int totalDist=0;
        int totalAirTime =0;



        for(AverageCountTuple dt: values){
        totalFlight += dt.getFlightCount();
        totalDist += dt.getDistCount();
        totalAirTime += dt.getAirTime();

        }

        double avgDist = (double)totalDist/totalFlight;
        double avgAirTime = (double)totalAirTime/totalFlight;
```

```
        tuple.setAirTime(totalAirTime);
        tuple.setDistCount(totalDist);
        tuple.setFlightCount(totalFlight);
        tuple.setAverageDist(avgDist);
        tuple.setAverageAirTime(avgAirTime);

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

3 Cancel By Carriers Count

```
package hadoop.project.cancel_by_carriers;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

public class CancelCarrierMain {

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(CancelCarrierMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(CancelCarrierMapper.class);
        job.setCombinerClass(CancelCarrierReducer.class);
```

```
        job.setReducerClass(CancelCarrierReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}
```

```
package hadoop.project.cancel_by_carriers;

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

import java.io.IOException;


public class CancelCarrierMapper extends Mapper<Object, Text, Text, CancelCountTuple>
{

    private CancelCountTuple tuple = new CancelCountTuple();

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

        if(tokens[0].equals("Year"))return;

        String carrier = tokens[8];


        try {
            String can = tokens[21];
            tuple.setCancelFlightCount(Integer.parseInt(can));

        }catch (Exception e){
            tuple.setCancelFlightCount(0);
        }


        tuple.setFlightCount(1);

        context.write(new Text(carrier),tuple);
    }
}
```

```
package hadoop.project.cancel_by_carriers;

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

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

import java.io.IOException;

public class CancelCarrierReducer extends Reducer<Text, CancelCountTuple, Text,
CancelCountTuple> {

    private CancelCountTuple res= new CancelCountTuple();

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

        int total=0;
        int cancelTotal=0;

        for(CancelCountTuple dt: values){
            total += dt.getFlightCount();
            cancelTotal +=dt.getCancelFlightCount();
        }

        double percent = ((double)cancelTotal/total)*100;

        res.setCancelFlightCount(cancelTotal);
        res.setFlightCount(total);
        res.setCancelPercent(percent);

        context.write(key,res);
    }
}


package hadoop.project.cancel_by_carriers;

import org.apache.hadoop.io.Writable;

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

public class CancelCountTuple implements Writable {

    private int flightCount=0;
    private int cancelFlightCount=0;
    private double cancelPercent =0.0;

    public int getFlightCount() {
        return flightCount;
    }

    public void setFlightCount(int flightCount) {
        this.flightCount = flightCount;
    }

    public int getCancelFlightCount() {
```

```
        return cancelFlightCount;
    }

    public void setCancelFlightCount(int cancelFlightCount) {
        this.cancelFlightCount = cancelFlightCount;
    }

    public double getCancelPercent() {
        return cancelPercent;
    }

    public void setCancelPercent(double cancelPercent) {
        this.cancelPercent = cancelPercent;
    }

    @Override
    public String toString() {
        return  "flightCount=" + flightCount +
                ", delayedFlightCount=" + cancelFlightCount +
                ", delayPercent=" + String.format("%.2f", cancelPercent);
    }

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeInt(flightCount);
        dataOutput.writeInt(cancelFlightCount);
        dataOutput.writeDouble(cancelPercent);

    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {

        flightCount = dataInput.readInt();
        cancelFlightCount = dataInput.readInt();
        cancelPercent = dataInput.readDouble();

    }
}
```

4- Cancelled By Year

```
package hadoop.project.cancelled_by_year;

import org.apache.hadoop.conf.Configuration;
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;
```

```
import java.io.IOException;

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(CancMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(CancMapper.class);
        job.setReducerClass(CancReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}


package hadoop.project.cancelled_by_year;

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

import java.io.IOException;

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

    //    hadoop datatype
    Text word = new Text();
    IntWritable one = new IntWritable(1);
```

```
    @Override
    protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        String [] values = value.toString().split(",");
        if(values[0].equals("Year"))return;
        try {
            String can = values[21];
            String year = values[0];

            if(can.equals("1"))
                context.write(new Text(year), one);
        } catch(Exception e){
            return;
        }

    }
}


package hadoop.project.cancelled_by_year;

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

import java.io.IOException;

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

    // just like in mongoDB values is iterable
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {

        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
            // can we use this--  Integer.parseInt(v.toString());
        }

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

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }

}
```

5- Delayed Flights By Careers

```
package hadoop.project.delay_by_carriers;
```

```
import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

public class DelayCarrierMain {

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(DelayCarrierMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(DelayCarrierMapper.class);
        job.setCombinerClass(DelayCarrierReducer.class);
        job.setReducerClass(DelayCarrierReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}


package hadoop.project.delay_by_carriers;

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

```
import java.io.IOException;


public class DelayCarrierMapper extends Mapper<Object, Text, Text, DelayCountTuple> {

    private DelayCountTuple tuple = new DelayCountTuple();

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

        if(tokens[0].equals("Year"))return;

        String carrier = tokens[8];


        try {
            int delay = Integer.parseInt(tokens[14]);


            if (delay > 15) {
                tuple.setDelayedFlightCount(1);
            } else {
                tuple.setDelayedFlightCount(0);
            }
        }catch (Exception e){
            tuple.setDelayedFlightCount(0);
        }

        tuple.setFlightCount(1);

        context.write(new Text(carrier),tuple);
    }
}


package hadoop.project.delay_by_carriers;

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

import java.io.IOException;

public class DelayCarrierReducer extends Reducer<Text, DelayCountTuple, Text,
DelayCountTuple> {

    private DelayCountTuple res= new DelayCountTuple();

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

        int total=0;
```

```
        int delayedTotal=0;

        for(DelayCountTuple dt: values){
            total += dt.getFlightCount();
            delayedTotal +=dt.getDelayedFlightCount();
        }

        double percent = ((double)delayedTotal/total)*100;

        res.setDelayedFlightCount(delayedTotal);
        res.setFlightCount(total);
        res.setDelayPercent(percent);

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

```
package hadoop.project.delay_by_carriers;

import org.apache.hadoop.io.Writable;

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

public class DelayCountTuple implements Writable {

    private int flightCount=0;
    private int delayedFlightCount=0;
    private double delayPercent =0.0;

    public int getFlightCount() {
        return flightCount;
    }

    public void setFlightCount(int flightCount) {
        this.flightCount = flightCount;
    }

    public int getDelayedFlightCount() {
        return delayedFlightCount;
    }

    public void setDelayedFlightCount(int delayedFlightCount) {
        this.delayedFlightCount = delayedFlightCount;
    }

    public double getDelayPercent() {
        return delayPercent;
    }

    public void setDelayPercent(double delayPercent) {
        this.delayPercent = delayPercent;
    }
```

```
    @Override
    public String toString() {
        return  "flightCount=" + flightCount +
                ", delayedFlightCount=" + delayedFlightCount +
                ", delayPercent=" + String.format("%.2f", delayPercent);
    }

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeInt(flightCount);
        dataOutput.writeInt(delayedFlightCount);
        dataOutput.writeDouble(delayPercent);

    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {

        flightCount = dataInput.readInt();
        delayedFlightCount = dataInput.readInt();
        delayPercent = dataInput.readDouble();

    }
}
```

6- Delay Groups

```
package hadoop.project.delay_groups;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
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;

import java.io.IOException;

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(GroupMain.class);
```

```
        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(GroupMapper.class);
        job.setReducerClass(GroupReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}


package hadoop.project.delay_groups;

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

import java.io.IOException;

public class GroupMapper  extends Mapper<Object, Text, Text, IntWritable> {


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

        if(tokens[0].equals("Year"))return;

        int delay =0;

        try {
            delay = Integer.parseInt(tokens[15]);

        }catch (Exception e){
           delay=0;
        }
```

```
        String grpKey="";

        if(delay<15)
            grpKey="Less than 15 Minutes";
        else if(delay>=15 && delay <=30)
            grpKey="Between 15 abd 30 minutes";
        else if(delay>30 && delay<60)
            grpKey="Between 30 minutes and 1 hour";
        else if(delay>=60 && delay<=120)
            grpKey="Between 1 hour and 2 hour";
        else
            grpKey="More than 2 hours";

        context.write(new Text(grpKey),new IntWritable(1));
    }
}


package hadoop.project.delay_groups;

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

import java.io.IOException;

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

    // just like in mongoDB values is iterable
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {

        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
            // can we use this--  Integer.parseInt(v.toString());
        }

        double total = 123534970.0;
        //double total = 1311827;

        double percent= (sum/total)*100;

        String res = String.format("%.2f", percent);

        context.write(key, new Text(res));

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }
```

```
}
```

7- Delay By Month

```
package hadoop.project.delay_month;

import org.apache.hadoop.io.Writable;

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

public class DelayCountTuple implements Writable {

    private int flightCount=0;
    private int delayedFlightCount=0;
    private double delayPercent =0.0;

    public int getFlightCount() {
        return flightCount;
    }

    public void setFlightCount(int flightCount) {
        this.flightCount = flightCount;
    }

    public int getDelayedFlightCount() {
        return delayedFlightCount;
    }

    public void setDelayedFlightCount(int delayedFlightCount) {
        this.delayedFlightCount = delayedFlightCount;
    }

    public double getDelayPercent() {
        return delayPercent;
    }

    public void setDelayPercent(double delayPercent) {
        this.delayPercent = delayPercent;
    }

    @Override
    public String toString() {
        return  "flightCount=" + flightCount +
                ", delayedFlightCount=" + delayedFlightCount +
                ", delayPercent=" + String.format("%.2f", delayPercent);
    }

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeInt(flightCount);
```

```
        dataOutput.writeInt(delayedFlightCount);
        dataOutput.writeDouble(delayPercent);

    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {

        flightCount = dataInput.readInt();
        delayedFlightCount = dataInput.readInt();
        delayPercent = dataInput.readDouble();

    }
}
```

```
package hadoop.project.delay_month;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

public class DelayMonthMain {

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(DelayMonthMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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

```
        job.setMapperClass(DelayMonthMapper.class);
        job.setCombinerClass(DelayMonthReducer.class);
        job.setReducerClass(DelayMonthReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}
```

```
package hadoop.project.delay_month;

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

import java.io.IOException;


public class DelayMonthMapper extends Mapper<Object, Text, Text, DelayCountTuple> {

    private String [] days ={"","1-January","2-February","3-March","4-April","5-
May","6June","7-July","8-August","9-September","10-October","11-November","12-
December"};
    private DelayCountTuple tuple = new DelayCountTuple();

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

        if(tokens[0].equals("Year"))return;

        String month = days[Integer.parseInt(tokens[1])];


        try {
            int delay = Integer.parseInt(tokens[14]);


            if (delay > 15) {
                tuple.setDelayedFlightCount(1);
            } else {
                tuple.setDelayedFlightCount(0);
            }
        }catch (Exception e){
            tuple.setDelayedFlightCount(0);
        }

        tuple.setFlightCount(1);

        context.write(new Text(month),tuple);
```

```
    }
}
```

```
package hadoop.project.delay_month;

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

import java.io.IOException;

public class DelayMonthReducer extends Reducer<Text, DelayCountTuple, Text,
DelayCountTuple> {

    private DelayCountTuple res= new DelayCountTuple();

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

        int total=0;
        int delayedTotal=0;

        for(DelayCountTuple dt: values){
            total += dt.getFlightCount();
            delayedTotal +=dt.getDelayedFlightCount();
        }

        double percent = ((double)delayedTotal/total)*100;

        res.setDelayedFlightCount(delayedTotal);
        res.setFlightCount(total);
        res.setDelayPercent(percent);

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

8- Delay per year

```
package hadoop.project.delay_per_year;

import org.apache.hadoop.conf.Configuration;
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;
```

```
import java.io.IOException;

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(DelayedMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(DelayedMapper.class);
        job.setReducerClass(DelayedReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}


package hadoop.project.delay_per_year;

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;

import java.io.IOException;

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

    //    hadoop datatype
    Text word = new Text();
    IntWritable one = new IntWritable(1);
```

```
    @Override
    protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        String [] values = value.toString().split(",");
        if(values[0].equals("Year"))return;
        try {
            int delay = Integer.parseInt(values[14]);
            String year = values[0];

            if(delay>=15)
                context.write(new Text(year), one);
        } catch(Exception e){
            return;
        }

    }
}


package hadoop.project.delay_per_year;

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

import java.io.IOException;

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

    // just like in mongoDB values is iterable
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {

        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
            // can we use this--  Integer.parseInt(v.toString());
        }

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

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }

}
```

9- Delay ratio by year

```java
package hadoop.project.delay_ratio_year;

import org.apache.hadoop.io.Writable;

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

public class DelayCountTuple implements Writable {

    private int flightCount=0;
    private int delayedFlightCount=0;
    private double delayPercent =0.0;

    public int getFlightCount() {
        return flightCount;
    }

    public void setFlightCount(int flightCount) {
        this.flightCount = flightCount;
    }

    public int getDelayedFlightCount() {
        return delayedFlightCount;
    }

    public void setDelayedFlightCount(int delayedFlightCount) {
        this.delayedFlightCount = delayedFlightCount;
    }

    public double getDelayPercent() {
        return delayPercent;
    }

    public void setDelayPercent(double delayPercent) {
        this.delayPercent = delayPercent;
    }

    @Override
    public String toString() {
        return  "flightCount=" + flightCount +
                ", delayedFlightCount=" + delayedFlightCount +
                ", delayPercent=" + String.format("%.2f", delayPercent);
    }

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeInt(flightCount);
        dataOutput.writeInt(delayedFlightCount);
        dataOutput.writeDouble(delayPercent);

    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {
```

```
        flightCount = dataInput.readInt();
        delayedFlightCount = dataInput.readInt();
        delayPercent = dataInput.readDouble();

    }
}


package hadoop.project.delay_ratio_year;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

public class DelayYearMain {

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(DelayYearMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(DelayYearMapper.class);
        job.setCombinerClass(DelayYearReducer.class);
        job.setReducerClass(DelayYearReducer.class);

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

```
        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}
```

```
package hadoop.project.delay_ratio_year;

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

import java.io.IOException;


public class DelayYearMapper extends Mapper<Object, Text, Text, DelayCountTuple> {

    private DelayCountTuple tuple = new DelayCountTuple();

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

        if(tokens[0].equals("Year"))return;

        String year = tokens[0];


        try {
            int delay = Integer.parseInt(tokens[14]);


            if (delay > 15) {
                tuple.setDelayedFlightCount(1);
            } else {
                tuple.setDelayedFlightCount(0);
            }
        }catch (Exception e){
            tuple.setDelayedFlightCount(0);
        }

        tuple.setFlightCount(1);

        context.write(new Text(year),tuple);
    }
}
```

```
package hadoop.project.delay_ratio_year;

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

```
import java.io.IOException;

public class DelayYearReducer extends Reducer<Text, DelayCountTuple, Text,
DelayCountTuple> {

    private DelayCountTuple res= new DelayCountTuple();

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

        int total=0;
        int delayedTotal=0;

        for(DelayCountTuple dt: values){
            total += dt.getFlightCount();
            delayedTotal +=dt.getDelayedFlightCount();
        }

        double percent = ((double)delayedTotal/total)*100;

        res.setDelayedFlightCount(delayedTotal);
        res.setFlightCount(total);
        res.setDelayPercent(percent);

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

10- All delayed count

```
package hadoop.project.delayed_count;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

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


        Configuration conf = new Configuration();
        // Create a new Job
```

```
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(DelayedMapper.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(DelayedMapper.class);
        job.setReducerClass(DelayedReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}


package hadoop.project.delayed_count;

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;

import java.io.IOException;

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

    //    hadoop datatype
    Text word = new Text();
    IntWritable one = new IntWritable(1);

    @Override
    protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        String [] values = value.toString().split(",");
        if(values[0].equals("Year"))return;
        try {
            int delay = Integer.parseInt(values[14]);
```

```
            if(delay>=15)
                context.write(NullWritable.get(), one);
        } catch(Exception e){
            return;
        }

    }
}
```

```
package hadoop.project.delayed_count;

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

import java.io.IOException;

public class DelayedReducer extends Reducer<NullWritable, IntWritable, NullWritable,
IntWritable> {

    // just like in mongoDB values is iterable
    @Override
    protected void reduce(NullWritable key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {

        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
            // can we use this--  Integer.parseInt(v.toString());
        }

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

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }

}
```

11-Delay by days

```
package hadoop.project.delays_by_days;

import org.apache.hadoop.io.Writable;

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

public class DelayCountTuple implements Writable {

    private int flightCount=0;
    private int delayedFlightCount=0;
```

```
    private double delayPercent =0.0;

    public int getFlightCount() {
        return flightCount;
    }

    public void setFlightCount(int flightCount) {
        this.flightCount = flightCount;
    }

    public int getDelayedFlightCount() {
        return delayedFlightCount;
    }

    public void setDelayedFlightCount(int delayedFlightCount) {
        this.delayedFlightCount = delayedFlightCount;
    }

    public double getDelayPercent() {
        return delayPercent;
    }

    public void setDelayPercent(double delayPercent) {
        this.delayPercent = delayPercent;
    }

    @Override
    public String toString() {
        return  "flightCount=" + flightCount +
                ", delayedFlightCount=" + delayedFlightCount +
                ", delayPercent=" + String.format("%.2f", delayPercent);
    }

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeInt(flightCount);
        dataOutput.writeInt(delayedFlightCount);
        dataOutput.writeDouble(delayPercent);

    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {

        flightCount = dataInput.readInt();
        delayedFlightCount = dataInput.readInt();
        delayPercent = dataInput.readDouble();

    }
}

package hadoop.project.delays_by_days;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

public class DelayDayMain {

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(DelayDayMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(DelayDayMapper.class);
        job.setCombinerClass(DelayDayReducer.class);
        job.setReducerClass(DelayDayReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}

package hadoop.project.delays_by_days;

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

import java.io.IOException;
```

```
public class DelayDayMapper extends Mapper<Object, Text, Text, DelayCountTuple> {

    private String [] days
={"","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"};
    private DelayCountTuple tuple = new DelayCountTuple();

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

        if(tokens[0].equals("Year"))return;

        String day = days[Integer.parseInt(tokens[3])];


        try {
            int delay = Integer.parseInt(tokens[14]);


            if (delay > 15) {
                tuple.setDelayedFlightCount(1);
            } else {
                tuple.setDelayedFlightCount(0);
            }
        }catch (Exception e){
            tuple.setDelayedFlightCount(0);
        }

        tuple.setFlightCount(1);

        context.write(new Text(day),tuple);
    }
}

package hadoop.project.delays_by_days;

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

import java.io.IOException;

public class DelayDayReducer extends Reducer<Text, DelayCountTuple, Text,
DelayCountTuple> {

    private DelayCountTuple res= new DelayCountTuple();

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

        int total=0;
        int delayedTotal=0;
```

```
        for(DelayCountTuple dt: values){
            total += dt.getFlightCount();
            delayedTotal +=dt.getDelayedFlightCount();
        }

        double percent = ((double)delayedTotal/total)*100;

        res.setDelayedFlightCount(delayedTotal);
        res.setFlightCount(total);
        res.setDelayPercent(percent);

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

12 Hierarchical Pattern

```
package hadoop.project.hier_pattern;

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

import java.io.IOException;

public class AirportMapper extends Mapper<Object, Text, Text, Text> {

    private Text outkey = new Text();
    private Text outvalue = new Text();

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

        String[] tokens = value.toString().split(",");
        if (tokens[0].equals("iata"))
            return;

        // The foreign join key is the post ID
        outkey.set(tokens[0]);


        // Flag this record for the reducer and then output
        StringBuffer sb = new StringBuffer();
        sb.append(tokens[1]);
        sb.append(" ");
        sb.append(tokens[2]);
        sb.append(" ");
        sb.append(tokens[3]);
        sb.append(" ");
        sb.append(tokens[4]);

        outvalue.set("P" + sb.toString());
        context.write(outkey, outvalue);
```

```
    }
}
```

```
package hadoop.project.hier_pattern;

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

public class HierMain {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = new Job(conf, "Hierarchy");
        job.setJarByClass(HierMain.class);

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

        MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class,
SrcCarrierMapper.class);

        job.setReducerClass(HierReducer.class);

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

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

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

```
package hadoop.project.hier_pattern;


import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import org.w3c.dom.Attr;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.xml.sax.InputSource;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
```

```
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.IOException;
import java.io.StringReader;
import java.io.StringWriter;
import java.util.ArrayList;
import java.util.List;

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

    private ArrayList<String> comments = new ArrayList<String>();
    private DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    private String post = null;

    @Override
    public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
        // Reset variables
        post = null;
        comments.clear();

        // For each input value
        for (Text t : values) {
            // If this is the post record, store it, minus the flag
            if (t.charAt(0) == 'P') {
                post = t.toString().substring(1).trim();
            } else {
                // Else, it is a comment record. Add it to the list, minus
                // the flag
                comments.add(t.toString().substring(1).trim());
            }
        }

        // If post is not null
        if (post != null) {
            // nest the comments underneath the post element
            String postWithCommentChildren = nestElements(post, comments);
            System.out.println(postWithCommentChildren);
            context.write(new Text(postWithCommentChildren), NullWritable.get());

        }
    }

    private String nestElements(String post, List<String> comments) {
        try {
            // Create the new document to build the XML
            DocumentBuilder bldr = dbf.newDocumentBuilder();
            Document doc = bldr.newDocument();

            // Copy parent node to document
            Element postEl = getXmlElementFromString(post);
            Element toAddPostEl = doc.createElement("Airport");

            // Copy the attributes of the original post element to the new
```

```
            // one
            copyAttributesToElement(postEl.getAttributes(), toAddPostEl);

            // For each comment, copy it to the "post" node
            for (String commentXml : comments) {
                Element commentEl = getXmlElementFromString(commentXml);
                Element toAddCommentEl = doc.createElement("Carriers");

                // Copy the attributes of the original comment element to
                // the new one
                copyAttributesToElement(commentEl.getAttributes(),
                        toAddCommentEl);

                // Add the copied comment to the post element
                toAddPostEl.appendChild(toAddCommentEl);
            }

            // Add the post element to the document
            doc.appendChild(toAddPostEl);

            // Transform the document into a String of XML and return
            return transformDocumentToString(doc);

        } catch (Exception e) {
            return null;
        }
    }

    private Element getXmlElementFromString(String xml) {
        try {
            // Create a new document builder
            DocumentBuilder bldr = dbf.newDocumentBuilder();

            // Parse the XML string and return the first element
            return bldr.parse(new InputSource(new StringReader(xml)))
                    .getDocumentElement();
        } catch (Exception e) {
            return null;
        }
    }

    private void copyAttributesToElement(NamedNodeMap attributes,
                                         Element element) {

        // For each attribute, copy it to the element
        for (int i = 0; i < attributes.getLength(); ++i) {
            Attr toCopy = (Attr) attributes.item(i);
            element.setAttribute(toCopy.getName(), toCopy.getValue());
        }
    }

    private String transformDocumentToString(Document doc) {
        try {
            TransformerFactory tf = TransformerFactory.newInstance();
            Transformer transformer = tf.newTransformer();
```

```
            transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION,
                    "yes");
            StringWriter writer = new StringWriter();
            transformer.transform(new DOMSource(doc), new StreamResult(
                    writer));
            // Replace all new line characters with an empty string to have
            // one record per line.
            return writer.getBuffer().toString().replaceAll("\n|\r", "");
        } catch (Exception e) {
            return null;
        }
    }
}
```

```
package hadoop.project.hier_pattern;

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

import java.io.IOException;

public class SrcCarrierMapper extends Mapper<Object, Text, Text, Text> {
    private Text outkey = new Text();
    private Text outvalue = new Text();

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

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

        // The foreign join key is the post ID
        outkey.set(tokens[0]);

        // Flag this record for the reducer and then output
        outvalue.set("C" + tokens[1]);
        context.write(outkey, outvalue);
    }
}
```

13- Inner Join on Carriers

```
package hadoop.project.inner_join_carriers;

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

import java.io.IOException;

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

    //    hadoop datatype
```

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

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

        String line = value.toString();
        String[] tokens= line.split(",");
        if(tokens[0].equals("Code"))return;
        String newKey = tokens[0];
        word.set(newKey);
        String outValue= "A"+tokens[1];
        context.write(word,new Text(outValue));
    }

}
```

```
package hadoop.project.inner_join_carriers;

import org.apache.hadoop.conf.Configuration;
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.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;

import java.io.IOException;

public class FlightMain {

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


        Configuration conf = new Configuration();
        Job job = new Job(conf, "Reduce-side join");
        job.setJarByClass(FlightMain.class);
        job.setReducerClass(FlightReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        MultipleInputs.addInputPath(job, new Path(args[0]),TextInputFormat.class,
CarrierMapper.class);
        MultipleInputs.addInputPath(job, new Path(args[1]),TextInputFormat.class,
FlightMapper.class);
        Path outputPath = new Path(args[2]);

        FileOutputFormat.setOutputPath(job, outputPath);
        outputPath.getFileSystem(conf).delete(outputPath);
```

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

    }
}
```

```
package hadoop.project.inner_join_carriers;

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

import java.io.IOException;

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

    //    hadoop datatype
    Text word = new Text();
    IntWritable one = new IntWritable(1);

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

        String line = value.toString();
        String[] tokens= line.split("\t");
        if(tokens[0].equals("Code"))return;
        String newKey = tokens[0];
        word.set(newKey);
        String outValue= "B"+tokens[1];
        context.write(word,new Text(outValue));
    }

}
```

```
package hadoop.project.inner_join_carriers;

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

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

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

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

    @Override
    protected void setup(Context context) throws IOException, InterruptedException {
```

```
        //joinType = context.getConfiguration().get("join.type");
        joinType="inner";
    }

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
       listA.clear();
       listB.clear();
        Iterator<Text> itr = values.iterator();
       while(itr.hasNext()){
           tmp= itr.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);
                    }
                }
            }
        }
    }
}
```

14- Inverted Index

```
package hadoop.project.inverted_index;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

public class IndexMain {
```

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(IndexMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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




        job.setMapperClass(IndexMapper.class);

        job.setReducerClass(IndexReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}

package hadoop.project.inverted_index;

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

import java.io.IOException;

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

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

```
        String line = value.toString();
        String[] tokens= line.split(",");
        if(tokens[0].equals("Year"))return;
        String src = tokens[16];
        String dest = tokens[17];

        context.write(new Text(src+":"),new Text(dest));
    }


}
```

```
package hadoop.project.inverted_index;

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

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

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

    // just like in mongoDB values is iterable
    HashSet<String> hs = new HashSet<>();
    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
        hs.clear();
        StringBuffer sb = new StringBuffer("");
        for(Text v: values){
            hs.add(v.toString());
            // can we use this--  Integer.parseInt(v.toString());
        }

        for(String v: hs){
            sb.append(v);
            sb.append(" ");
        }

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

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }

}
```

15- Inverted Index Helper

```
package hadoop.project.inverted_index_count;

import org.apache.hadoop.conf.Configuration;
```

```
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;

import java.io.IOException;

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(IndexMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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




        job.setMapperClass(IndexMapper.class);

        job.setReducerClass(IndexReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}

package hadoop.project.inverted_index_count;

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

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

import java.io.IOException;

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

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

        String line = value.toString();
        String[] tokens= line.split("\t");
        String pair[] = tokens[0].split("-");
        String src = pair[0];
        String dest = pair[1];

        context.write(new Text(src+":"),new Text(dest));
    }


}

package hadoop.project.inverted_index_count;

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

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

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

    // just like in mongoDB values is iterable
    HashSet<String> hs = new HashSet<>();
    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
        hs.clear();
        StringBuffer sb = new StringBuffer("");
//        for(Text v: values){
//            hs.add(v.toString());
//            // can we use this--  Integer.parseInt(v.toString());
//        }

        for(Text v: values){
            sb.append(v.toString());
            sb.append(" ");
        }

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

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }
```

```
}
```

16- Partitioning Pattern on Year

```
package hadoop.project.partitioning_pattern_year;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(PartitionMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


        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(PartitionMapper.class);
        // Set custom partitioner and min last access date
        job.setPartitionerClass(PartitionPartitioner.class);
        PartitionPartitioner.setMinLastAccessDate(job, 1987);

// Last access dates span between 2008-2011, or 4 years
        job.setNumReduceTasks(22);

        job.setReducerClass(PartitionReducer.class);
```

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}
```

```
package hadoop.project.partitioning_pattern_year;

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;

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

public class PartitionMapper extends Mapper<Object, Text, IntWritable, Text> {


    private IntWritable outkey = new IntWritable();

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

        String [] tokens = value.toString().split(",");
        if(tokens[0].equals("Year"))
            return;

        StringBuffer sb = new StringBuffer();
        sb.append(tokens[0]);
        sb.append("\t");
        sb.append(tokens[8]);
        sb.append("\t");
        sb.append(tokens[16]);
        sb.append("\t");
        sb.append(tokens[17]);

        int year = Integer.parseInt(tokens[0]);

        outkey.set(year);

        // Write out the year with the input value
        context.write(outkey, new Text(sb.toString()));
    }
}
```

```
package hadoop.project.partitioning_pattern_year;

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 PartitionPartitioner 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 int getPartition(IntWritable key, Text value, int numPartitions) {
        return key.get() - minLastAccessDateYear;
    }

    public Configuration getConf() {
        return conf;
    }

    public void setConf(Configuration conf) {
        this.conf = conf;
        minLastAccessDateYear = conf.getInt(MIN_LAST_ACCESS_DATE_YEAR, 0);
    }

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

package hadoop.project.partitioning_pattern_year;

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

import java.io.IOException;

public class PartitionReducer 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());
        }
    }
}
```

17- Recommendation System

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package hadoop.project.recommendation_system;

import org.apache.hadoop.io.WritableComparable;

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

/**
 *
 * @author ankit
 */
public class CompositeKey implements WritableComparable<CompositeKey>{

    private String srcDest;
    private String carrierInfo;

    public CompositeKey() {
        super();
    }

    public String getSrcDest() {
        return srcDest;
    }

    public void setSrcDest(String srcDest) {
        this.srcDest = srcDest;
    }

    public String getCarrierInfo() {
        return carrierInfo;
    }

    public void setCarrierInfo(String carrierInfo) {
        this.carrierInfo = carrierInfo;
    }

    public CompositeKey(String srcDest, String carrierInfo) {
        this.srcDest = srcDest;
        this.carrierInfo = carrierInfo;
    }

    @Override
    public void write(DataOutput d) throws IOException {
        d.writeUTF(srcDest);
        d.writeUTF(carrierInfo);
    }
```

```
    @Override
    public void readFields(DataInput di) throws IOException {
        srcDest = di.readUTF();
        carrierInfo = di.readUTF();
    }

    @Override
    public int compareTo(CompositeKey o) {
        int result = this.srcDest.compareTo(o.getSrcDest());
        if(result==0){
            String c1 = this.carrierInfo;
            Double rms1 = Double.parseDouble(c1.split("\t")[1]);

            String c2 = o.getCarrierInfo();
            Double rms2 = Double.parseDouble(c2.split("\t")[1]);
            return rms1.compareTo(rms2);
        }

        return result;
    }

    @Override
    public String toString() {
        return srcDest + " : " + carrierInfo;
    }
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package hadoop.project.recommendation_system;

/**
 *
 * @author ankit
 */
import org.apache.hadoop.io.WritableComparator;

public class GroupComparator extends WritableComparator{

   protected GroupComparator() {
      super(CompositeKey.class, true);
   }

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

      CompositeKey ckw1 = (CompositeKey)a;
      CompositeKey ckw2 = (CompositeKey)b;

      return ckw1.getSrcDest().compareTo(ckw2.getSrcDest());
```

```
   }
}


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package hadoop.project.recommendation_system;

/**
 *
 * @author ankit
 */
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.mapreduce.Partitioner;

public class KeyPartition extends Partitioner<CompositeKey, NullWritable>{

   @Override
   public int getPartition(CompositeKey key, NullWritable value, int numPartitions) {

      return key.getSrcDest().hashCode()%numPartitions;

   }

}


/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package hadoop.project.recommendation_system;

/**
 *
 * @author ankit
 */

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

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;

public class SecondarySortComparator extends WritableComparator {

       private final static SimpleDateFormat frmt = new SimpleDateFormat("yyyy-MM-
dd");
```

```
    protected SecondarySortComparator() {
        super(CompositeKey.class, true);
    }

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



        CompositeKey ck1 = (CompositeKey) a;
        CompositeKey ck2 = (CompositeKey) b;

        int result = ck1.getSrcDest().compareTo(ck2.getSrcDest());

        if (result == 0) {
            String c1 = ck1.getCarrierInfo();
            Double rms1 = Double.parseDouble(c1.split("\t")[1]);

            String c2 = ck2.getCarrierInfo();
            Double rms2 = Double.parseDouble(c2.split("\t")[1]);
            result =  rms1.compareTo(rms2);

        }

        return result;
    }



}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package hadoop.project.recommendation_system;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
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.output.FileOutputFormat;

import java.io.IOException;

/**
 *
 * @author ankit
 */
public class SecondarySortDriver {

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

```
    {
        Job job = Job.getInstance();

        job.setJarByClass(SecondarySortDriver.class);

        job.setGroupingComparatorClass(GroupComparator.class);
        job.setSortComparatorClass(SecondarySortComparator.class);
        job.setPartitionerClass(KeyPartition.class);

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

        job.setMapperClass(SecondarySortMapper.class);
        job.setReducerClass(SecondarySortReducer.class);

        job.setNumReduceTasks(1);

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

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

        job.waitForCompletion(true);
    }
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package hadoop.project.recommendation_system;

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

import java.io.IOException;

/**
 *
 * @author ankit
 */
public class SecondarySortMapper extends Mapper<LongWritable, Text, CompositeKey,
NullWritable>{

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

```
        //To change body of generated methods, choose Tools | Templates.

        String [] tokens = value.toString().split("\t",2);

        try {
            String srcDest = tokens[0];
            String carrInfo = tokens[1];

            CompositeKey coKey = new CompositeKey(srcDest, carrInfo);

            context.write(coKey, NullWritable.get());
        }catch(Exception e){
            e.getStackTrace();
        }

    }


}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package hadoop.project.recommendation_system;

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

import java.io.IOException;

/**
 *
 * @author ankit
 */
public class SecondarySortReducer extends Reducer<CompositeKey, NullWritable,
CompositeKey, NullWritable>{

    @Override
    protected void reduce(CompositeKey key, Iterable<NullWritable> values, Context
context) throws IOException, InterruptedException {
        //To change body of generated methods, choose Tools | Templates.

        for(NullWritable v:values){
            context.write(key, v);
        }
    }

}
```

18- RMS for src and dest delays

```java
package hadoop.project.rms_src_dest_carriers;

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

import java.io.IOException;

public class RMSCombiner extends Reducer<Text, RMSCountTuple, Text, RMSCountTuple> {

    private RMSCountTuple res= new RMSCountTuple();

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

        int total=0;
        int arrDelay=0;
        int depDelay=0;

        for(RMSCountTuple tup : values){
            total +=tup.getTotalFlight();
            arrDelay +=tup.getArrDelay();
            depDelay +=tup.getDepDelay();
        }

        res.setTotalFlight(total);
        res.setArrDelay(arrDelay);
        res.setDepDelay(depDelay);

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

```java
package hadoop.project.rms_src_dest_carriers;

import org.apache.hadoop.io.Writable;

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

public class RMSCountTuple implements Writable {

    private int arrDelay=0;
    private int depDelay=0;
    private int totalFlight=0;
    private double rms  =0.0;

    public int getArrDelay() {
        return arrDelay;
    }

    public void setArrDelay(int arrDelay) {
        this.arrDelay = arrDelay;
```

```
    }

    public int getDepDelay() {
        return depDelay;
    }

    public void setDepDelay(int depDelay) {
        this.depDelay = depDelay;
    }

    public int getTotalFlight() {
        return totalFlight;
    }

    public void setTotalFlight(int totalFlight) {
        this.totalFlight = totalFlight;
    }

    public double getRms() {
        return rms;
    }

    public void setRms(double rms) {
        this.rms = rms;
    }

    @Override
    public String toString() {
        return "{" +
                "arrDelay=" + arrDelay +
                ", depDelay=" + depDelay +
                ", totalFlight=" + totalFlight +
                ", rms=" +  String.format("%.4f", rms)+
                '}';
    }

//
//    @Override
//    public String toString() {
//        return String.format("%.4f", rms);
//    }

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeInt(arrDelay);
        dataOutput.writeInt(depDelay);
        dataOutput.writeInt(totalFlight);
        dataOutput.writeDouble(rms);

    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {

        arrDelay = dataInput.readInt();
```

```
        depDelay = dataInput.readInt();
        totalFlight = dataInput.readInt();
        rms = dataInput.readDouble();

    }
}
package hadoop.project.rms_src_dest_carriers;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

public class RMSMain {

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(RMSMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(RMSMapper.class);
        job.setCombinerClass(RMSCombiner.class);
        job.setReducerClass(RMSReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);
```

```
    }
}

package hadoop.project.rms_src_dest_carriers;

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

import java.io.IOException;


public class RMSMapper extends Mapper<Object, Text, Text, RMSCountTuple> {

    private RMSCountTuple tuple = new RMSCountTuple();

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

        if(tokens[0].equals("Year"))return;

        String src = tokens[16];
        String dest = tokens[17];
        String carrier = tokens[8];

       int arrDelay=0;
       int depDelay=0;



        try {
            arrDelay = Integer.parseInt(tokens[14]);
            depDelay = Integer.parseInt(tokens[15]);
        }catch (Exception e){

        }

        String newKey = src+"-" + dest +"\t"+carrier;

        tuple.setArrDelay(arrDelay);
        tuple.setDepDelay(depDelay);
        tuple.setTotalFlight(1);

        context.write(new Text(newKey),tuple);
    }
}

package hadoop.project.rms_src_dest_carriers;

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

import java.io.IOException;
```

```
public class RMSReducer extends Reducer<Text, RMSCountTuple, Text, RMSCountTuple> {

    private RMSCountTuple res= new RMSCountTuple();

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

        int total=0;
        int arrDelay=0;
        int depDelay=0;

        for(RMSCountTuple tup : values){
            total +=tup.getTotalFlight();
            arrDelay +=tup.getArrDelay();
            depDelay +=tup.getDepDelay();
        }

        double avgArrDelay = (double)arrDelay/total;
        double avgDepDelay = (double)depDelay/total;

        double rms = Math.sqrt((avgArrDelay*avgArrDelay) +
(avgDepDelay*avgDepDelay));

        res.setTotalFlight(total);
        res.setArrDelay(arrDelay);
        res.setDepDelay(depDelay);
        res.setRms(rms);



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

19- SRC Carrier Map

```
package hadoop.project.src_carrier_map;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

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

```
ClassNotFoundException{


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(SrcCarrierMapper.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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




        job.setMapperClass(SrcCarrierMapper.class);
        job.setReducerClass(SrcCarrierReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}

package hadoop.project.src_carrier_map;

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

import java.io.IOException;

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

    //    hadoop datatype
    Text word = new Text();

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

```
        String line = value.toString();
        String[] tokens= line.split(",");
        if(tokens[0].equals("Year"))return;
        String orig = tokens[16];
        String carrier = tokens[8];
        word.set(orig);
        context.write(word,new Text(carrier));
    }


}
```

```
package hadoop.project.src_carrier_map;

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

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

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

    // just like in mongoDB values is iterable
    private HashSet<String> carriers = new HashSet<>();
    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {


        for(Text v: values){
            carriers.add(v.toString());
            // can we use this--  Integer.parseInt(v.toString());
        }
        for(String car: carriers)
        context.write(key, new Text(car));

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }

}
```

21- Src-Dest pair count

```
package hadoop.project.src_dest_count;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(SrcDestMapper.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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




        job.setMapperClass(SrcDestMapper.class);
        job.setReducerClass(SrcDestReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}

package hadoop.project.src_dest_count;

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

import java.io.IOException;
```

```
public class SrcDestMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    //    hadoop datatype
    Text word = new Text();
    IntWritable one = new IntWritable(1);

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

        String line = value.toString();
        String[] tokens= line.split(",");
        if(tokens[0].equals("Year"))return;
        String orig_dest = tokens[16]+"-" + tokens[17];
        word.set(orig_dest);
        context.write(word,one);
    }


}

package hadoop.project.src_dest_count;

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

import java.io.IOException;

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

    // just like in mongoDB values is iterable
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {

        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
            // can we use this--  Integer.parseInt(v.toString());
        }

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

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }

}

package hadoop.project.src_dest_count;

import org.apache.hadoop.io.IntWritable;
```

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

import java.io.IOException;

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

    //    hadoop datatype
    Text word = new Text();
    IntWritable one = new IntWritable(1);

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

        String line = value.toString();
        String[] tokens= line.split(",");
        if(tokens[0].equals("Year"))return;
        String orig_dest = tokens[16]+"-" + tokens[17];
        word.set(orig_dest);
        context.write(word,one);
    }


}
```

```
package hadoop.project.src_dest_count;

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

import java.io.IOException;

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

    // just like in mongoDB values is iterable
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {

        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
            // can we use this--  Integer.parseInt(v.toString());
        }

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

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }
```

```
}
```

22- Top 20 SRC Dest

```
package hadoop.project.top20_src_dest;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

public class DelaySrcMain {

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(DelaySrcMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(DelaySrcMapper.class);
        //job.setCombinerClass(DelaySrcReducer.class);
        job.setReducerClass(DelaySrcReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);
```

```
    }
}
```

```
package hadoop.project.top20_src_dest;


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

import java.io.IOException;


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


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

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

```
package hadoop.project.top20_src_dest;

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

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

public class DelaySrcReducer extends Reducer<NullWritable, Text, NullWritable, Text>
{
    private TreeMap<Double, Text> countMap = new TreeMap<>();

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

            String[] tokens = value.toString().split("\t");
            double perc = Double.parseDouble(tokens[1]);

            countMap.put(perc, new Text(value));


            if (countMap.size() > 20)
                countMap.remove(countMap.lastKey());
        }

        for (Text t : countMap.values())
```

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

23- Top 30 Busy SRC Dest Pair

```
package hadoop.project.top30_busy_src_dest_pair;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(TopNMapper.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(TopNMapper.class);
        job.setReducerClass(TopNReducer.class);

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

```
        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}
```

```
package hadoop.project.top30_busy_src_dest_pair;

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

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

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

    // store a map of number of trips to src-dest pair
    private TreeMap<Integer,Text> countMap = new TreeMap<>();

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

        if(val.length==2){
            String pair = val[0];
            int count = Integer.parseInt(val[1]);
            countMap.put(count,new Text(value));
        }

        if(countMap.size()>30)
            countMap.remove(countMap.firstKey());
    }

    @Override
    protected void cleanup(Context context) throws IOException, InterruptedException
{
        for(Text t: countMap.values())
            context.write(NullWritable.get(),t);
    }
}
```

```
package hadoop.project.top30_busy_src_dest_pair;

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

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

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

```
    private TreeMap<Integer, Text> countMap = new TreeMap<>();

    @Override
    protected void reduce(NullWritable key, Iterable<Text> values, Context context)
throws IOException, InterruptedException {
        for(Text value: values){
            String[] val= value.toString().split("\t");

            if(val.length==2){
                String pair = val[0];
                int count = Integer.parseInt(val[1]);
                countMap.put(count,new Text(value));
            }

            if(countMap.size()>30)
                countMap.remove(countMap.firstKey());
        }

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

24- Top dst

```
package hadoop.project.top_dst;

import org.apache.hadoop.io.Writable;

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

public class DelayCountTuple implements Writable {

    private int flightCount=0;
    private int delayedFlightCount=0;
    private double delayPercent =0.0;

    public int getFlightCount() {
        return flightCount;
    }

    public void setFlightCount(int flightCount) {
        this.flightCount = flightCount;
    }

    public int getDelayedFlightCount() {
        return delayedFlightCount;
    }

    public void setDelayedFlightCount(int delayedFlightCount) {
        this.delayedFlightCount = delayedFlightCount;
    }
```

```
    public double getDelayPercent() {
        return delayPercent;
    }

    public void setDelayPercent(double delayPercent) {
        this.delayPercent = delayPercent;
    }

    @Override
    public String toString() {
        return  ""+delayPercent;
    }

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeInt(flightCount);
        dataOutput.writeInt(delayedFlightCount);
        dataOutput.writeDouble(delayPercent);

    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {

        flightCount = dataInput.readInt();
        delayedFlightCount = dataInput.readInt();
        delayPercent = dataInput.readDouble();

    }
}

package hadoop.project.top_dst;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
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;

import java.io.IOException;

public class DelayDestMain {

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


        Configuration conf = new Configuration();
        // Create a new Job
```

```
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(DelayDestMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(DelayDestMapper.class);
        //job.setCombinerClass(DelaySrcReducer.class);
        job.setReducerClass(DelayDestReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}
```

```
package hadoop.project.top_dst;

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

import java.io.IOException;


public class DelayDestMapper extends Mapper<Object, Text, Text, DelayCountTuple> {

    private DelayCountTuple tuple = new DelayCountTuple();

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

        if(tokens[0].equals("Year"))return;

        String dest = tokens[17];

        try {
            int delay = Integer.parseInt(tokens[14]);
```

```
            if (delay > 15) {
                tuple.setDelayedFlightCount(1);
            } else {
                tuple.setDelayedFlightCount(0);
            }
        }catch (Exception e){
            tuple.setDelayedFlightCount(0);
        }

        tuple.setFlightCount(1);

        context.write(new Text(dest),tuple);
    }
}
```

```
package hadoop.project.top_dst;

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

import java.io.IOException;

public class DelayDestReducer extends Reducer<Text, DelayCountTuple, Text,
DoubleWritable> {

    private DelayCountTuple res= new DelayCountTuple();

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

        int total=0;
        int delayedTotal=0;

        for(DelayCountTuple dt: values){
            total += dt.getFlightCount();
            delayedTotal +=dt.getDelayedFlightCount();
        }

        double percent = ((double)delayedTotal/total)*100;

        res.setDelayedFlightCount(delayedTotal);
        res.setFlightCount(total);
        res.setDelayPercent(percent);

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

25- Top SRC

```java
package hadoop.project.top_src;

import org.apache.hadoop.io.Writable;

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

public class DelayCountTuple implements Writable {

    private int flightCount=0;
    private int delayedFlightCount=0;
    private double delayPercent =0.0;

    public int getFlightCount() {
        return flightCount;
    }

    public void setFlightCount(int flightCount) {
        this.flightCount = flightCount;
    }

    public int getDelayedFlightCount() {
        return delayedFlightCount;
    }

    public void setDelayedFlightCount(int delayedFlightCount) {
        this.delayedFlightCount = delayedFlightCount;
    }

    public double getDelayPercent() {
        return delayPercent;
    }

    public void setDelayPercent(double delayPercent) {
        this.delayPercent = delayPercent;
    }

    @Override
    public String toString() {
        return  ""+delayPercent;
    }

    @Override
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeInt(flightCount);
        dataOutput.writeInt(delayedFlightCount);
        dataOutput.writeDouble(delayPercent);

    }

    @Override
    public void readFields(DataInput dataInput) throws IOException {

        flightCount = dataInput.readInt();
```

```
        delayedFlightCount = dataInput.readInt();
        delayPercent = dataInput.readDouble();

    }
}
```

```
package hadoop.project.top_src;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
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;

import java.io.IOException;

public class DelaySrcMain {

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(DelaySrcMain.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(DelaySrcMapper.class);
        //job.setCombinerClass(DelaySrcReducer.class);
        job.setReducerClass(DelaySrcReducer.class);

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

```
        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}
```

```
package hadoop.project.top_src;

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

import java.io.IOException;


public class DelaySrcMapper extends Mapper<Object, Text, Text, DelayCountTuple> {

    private DelayCountTuple tuple = new DelayCountTuple();

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

        if(tokens[0].equals("Year"))return;

        String src = tokens[16];

        try {
            int delay = Integer.parseInt(tokens[15]);


            if (delay > 15) {
                tuple.setDelayedFlightCount(1);
            } else {
                tuple.setDelayedFlightCount(0);
            }
        }catch (Exception e){
            tuple.setDelayedFlightCount(0);
        }

        tuple.setFlightCount(1);

        context.write(new Text(src),tuple);
    }
}
```

```
package hadoop.project.top_src;

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

import java.io.IOException;
```

```
public class DelaySrcReducer extends Reducer<Text, DelayCountTuple, Text,
DoubleWritable> {

    private DelayCountTuple res= new DelayCountTuple();

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

        int total=0;
        int delayedTotal=0;

        for(DelayCountTuple dt: values){
            total += dt.getFlightCount();
            delayedTotal +=dt.getDelayedFlightCount();
        }

        double percent = ((double)delayedTotal/total)*100;

        res.setDelayedFlightCount(delayedTotal);
        res.setFlightCount(total);
        res.setDelayPercent(percent);

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

26- Total Count

```
package hadoop.project.total_count;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(WordMapper.class);
```

```
        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(WordMapper.class);
        job.setReducerClass(WordReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}
```

```
package hadoop.project.total_count;

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;

import java.io.IOException;

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

    //    hadoop datatype
    Text word = new Text();
    IntWritable one = new IntWritable(1);

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


        context.write(NullWritable.get(),one);
    }


}
```

```
package hadoop.project.total_count;

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

import java.io.IOException;

public class WordReducer extends Reducer<NullWritable, IntWritable, NullWritable,
IntWritable> {

    // just like in mongoDB values is iterable
    @Override
    protected void reduce(NullWritable key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {

        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
            // can we use this--  Integer.parseInt(v.toString());
        }

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

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }

}
```

26- Unique Carrier Count

```
package hadoop.project.unique_carrier_count;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

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


        Configuration conf = new Configuration();
```

```
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(CarrierMapper.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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




        job.setMapperClass(CarrierMapper.class);
        job.setReducerClass(CarrierReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}

package hadoop.project.unique_carrier_count;

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

import java.io.IOException;

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

    //    hadoop datatype
    Text word = new Text();
    IntWritable one = new IntWritable(1);

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

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

```
        if(tokens[0].equals("Year"))return;

        String carrier = tokens[8];
        word.set(carrier);
        context.write(word,one);
    }


}
```

```
package hadoop.project.unique_carrier_count;

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

import java.io.IOException;

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

    // just like in mongoDB values is iterable
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {

        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
            // can we use this--  Integer.parseInt(v.toString());
        }

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

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }

}
```

27 Yearly Data

```
package hadoop.project.yearly_data;

import org.apache.hadoop.conf.Configuration;
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;

import java.io.IOException;

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


        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(MRCount.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");


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


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

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


        job.setMapperClass(YearMapper.class);
        job.setReducerClass(YearReducer.class);

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

        // Submit the job, then poll for progress until the job is complete
        System.exit(job.waitForCompletion(true)?0:1);

    }
}


package hadoop.project.yearly_data;

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;

import java.io.IOException;

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

    //    hadoop datatype
```

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

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

        String line = value.toString();
        String[] tokens= line.split(",");
        if(tokens[0].equals("Year"))return;
        String year = tokens[0];
        word.set(year);
        context.write(word,one);
    }


}
```

```
package hadoop.project.yearly_data;

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

import java.io.IOException;

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

    // just like in mongoDB values is iterable
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {

        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
            // can we use this--  Integer.parseInt(v.toString());
        }

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

        //super.reduce(key, values, context); //To change body of generated methods,
choose Tools | Templates.
    }

}
```