# Abstract:

In recent years, there has been a tremendous increase in the number of air passengers. This increasing figures have enhanced problems, ultimately causing flight delays. Delay in flights is recognized not only the usual factor but also a bothersome experience offered to customers. Delay causes effects leading from passengers monetary value to dis-satisfactions. Additionally, it also affects the business value and status of the aviation industry. Flight delays can be caused due to many diverse factors, tending from adverse weather conditions, technical problem, restriction in air-traffic and many more. These concerned parameter makes it important in carrying out analyzes based on past trends of delays which can be further divided into daily, monthly and yearly intervals. For this project, U.S. airlines and airports data is been utilized. This dataset contains recent data from the years 2015-2019 having daily records of airlines and airports, arrival and departure delays, with cancelled and diverted flights. This project makes use of big data technologies like Hadoop MapReduce, Apache Hive, Apache Pig, Hadoop Distributed File System (HDFS),HBASE and MySQL. These technologies would be helpful in determining delay problems faced by the aviation industry where an appropriate decision can be implemented and furthermore would be benefiting customers and industrialists of the aviation domain.Apart from the aformentioned analysis.few other analysis have been performed which are discussed in detail in the upcoming sections.

# Dataset Description:

The **Flight  Reporting Carrier On-Time Performance (1987-present)** is a vast dataset comprising of size around 5GB(approx).However,due to machine constraints I have chosen a smaller(600,000 records approx) subset from 2015-2020 years.It is csv file comprising of 6lacs record and 30 columns.The following is the description of the dataset

1.Year : 2015-2020
2.Month : 1-12
3.DAY_OF_MONTH : 1-31
4.DAY_OF_WEEK : 1 (Monday) - 7 (Sunday)
5.OP_UNIQUE_CARRIER:Unique Carrier Code. When the same code has been used by multiple carriers, a numeric suffix is used for earlier users, for example, PA, PA(1), PA(2). Use this field for analysis across a range of years.
6.TAIL_NUM : Tail Number
7.OP_CARRIER_FL_NUM : Flight Number
8.ORIGIN : Origin Airport
9.DEST : Destination Airport
10.CRS_DEPT_TIME : CRS Departure Time (local time: hhmm)
11.DEP_TIME : Actual Departure Time (local time: hhmm)
12.DEP_DELAY : Difference in minutes between scheduled and actual departure time. Early departures show negative numbers.
13.TAXI_OUT : Taxi Out Time, in Minutes
14.TAXI_IN : Taxi In Time, in Minutes
15.CRS_ARR_TIME : CRS Arrival Time (local time: hhmm)
16.ARR_TIME :
17.ARR_DELAY : Difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers.
18.CANCELLED : Cancelled Flight Indicator (1=Yes)
19.CANCELLATION_CODE : Specifies The Reason For Cancellation
20.DIVERTED : Diverted Flight Indicator (1=Yes)
21.CRS_ELAPSED_TIME : CRS Elapsed Time of Flight, in Minutes
22.ACTUAL_ELAPSED_TIME : Elapsed Time of Flight, in Minutes
23.AIR_TIME : Flight Time, in Minutes
24.DISTANCE : Distance between airports (miles)
25.CARRIER_DELAY : Carrier Delay, in Minutes
26.WEATHER_DELAY : Weather Delay, in Minutes
27.NAS_DELAY : National Air System Delay, in Minutes
28.SECURITY_DELAY : Security Delay, in Minutes
29.LATE_AIRCRAFT_DELAY : Late Aircraft Delay, in Minutes

The other dataset used was “carrier.csv” consists of two colums  one for carrier code and other for description.

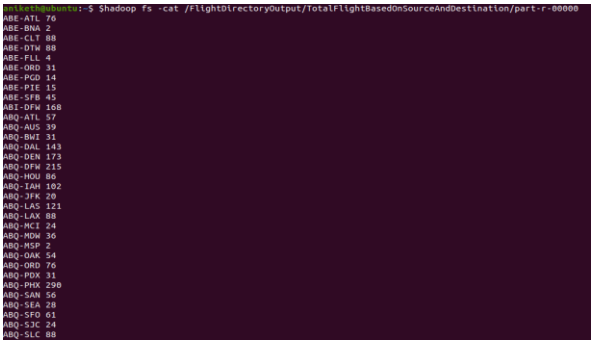
# Dataset Resources:

http://stat-computing.org/dataexpo/2009/the-data.html
https://www.transtats.bts.gov/Fields.asp?Table_ID=236

# Hadoop MapReduce Analysis:

1)Total Flights flown between source and destination:

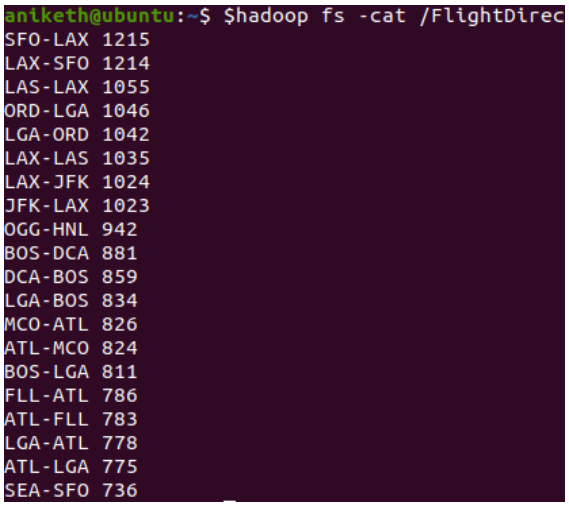
This is a simple map reduce program to calculate the number of flights flown between source and destination.




Business value captured:The airline company can get an idea as to which source-destination have demand and increase number of airplanes accordingly.Air traffic is also monitored based on this data.

2)Top 30 source destination pair:

I used the TopN filtering  pattern to get the top 30 source destination pairs as discussed in the lectures by the professor.




Business value added:Airline comapnies can know which regions are the popular and can increase their capacity accordingly.For example in the above result we have San Francisco to Los Angeles as most traversed route.

3)Number of flights cancelled between 2015-2020:

```
aniketh@ubuntu:~$ $hadoop fs -cat /FlightDirectoryOutput/DelayedCount/part-r-00000
82285
```

Business value added:The transport department can take this data and introspect as why the flights have been cancelled.Basically it just gives a quick total of how many flights were cancelled in five years.

4)Different Airline companies and the number of times they were flown:
I used the map reduce program to calculate the number of airline companies based on the "OP_UNIQUE_CARRIER" column

```
aniketh@ubuntu:~$ $hadoop fs  -cat /FlightDirectoryOutput/UniqueCarrierCount/part-r-00000
9E      23068
AA      76276
AS      21110
B6      24709
DL      80067
EV      10564
F9      12467
G4      7774
HA      7035
MQ      26200
NK      17477
OH      24309
OO      71160
```

Business value added : This data can be used to analyse which airline comapnies are popular among customers.

5)Airline companies full name:

I used the output of the above file and carrier.csv file and performed an Reduced-Side Join pattern to perform inner join.The foreign key in this case is the "OP_UNIQUE_CARRIER".

```
aniketh@ubuntu:~$ $hadoop fs -cat /FlightDirectoryOutput/Uniqu
Pinnacle Airlines Inc.  23068
American Airlines Inc.  76276
Alaska Airlines Inc.    21110
JetBlue Airways 24709
Delta Air Lines Inc.    80067
Atlantic Southeast Airlines     10564
Frontier Airlines Inc.  12467
Allegiant Air   7774
Hawaiian Airlines Inc.  7035
American Eagle Airlines Inc.    26200
Spirit Air Lines        17477
Comair Inc.     24309
Skywest Airlines Inc.   71160
United Air Lines Inc.   48401
Southwest Airlines Co.  109770
Mesa Airlines Inc.      17836
"Midwest Airline        29123
```

6)Flight count based on year:

This map-reduce program  calculates the number of airline carriers on an yearly basis.

```
aniketh@ubuntu:~$ $hadoop fs -cat /FlightDir
2015    65767
2016    33936
2017    37852
2018    19500
2019    35981
2020    414310
```

Business value added:This analysis gives the number of flights scheduled for travel on yearly basis.We can get measure the air travel demand metric

7)Number of flights delayed per year:

This program calculates the flights delayed per year.

```
aniketh@ubuntu:~$ $hadoop fs -cat /FlightDirectoryOut
2015    30230
2016    12653
2017    17509
2018    11560
2019    16183
2020    223538
```

Business value added:This analysis gives the number of flights delayed on yearly basis.It gives a quick glance of delayed count.For example we can see that for 2020 more flights were delayed when compared to previous years.Probably this might be due to pandemic!!!

8)Flights Cancelled Per Year:

This program calculates the number of flights cancelled per year.

```
aniketh@ubuntu:~$ $hadoop fs -cat /FlightDirectoryOutput/Canc
2015    840
2016    563
2017    181
2018    579
2019    180
2020    4585
```

Business value added:This analysis gives the number of flights delayed on yearly basis.It gives a quick glance of cancelled count.

9)Percentage of Flights delayed per year:

I used a custom writable class to stream tuples and calculated the percentage based on (DelayedFlightsNumber/TotalFlightsNumber)*100.
I also used combiner for optimisation.

```
aniketh@ubuntu:~$ $hadoop fs -cat /FlightDirectoryOutput/RatioOfDelayedFlightsPerTotalFligh
2015    Number of Flights=65767, Number of Delayed Flights=7894, Delay Percentage=12.00
2016    Number of Flights=33936, Number of Delayed Flights=3803, Delay Percentage=11.21
2017    Number of Flights=37852, Number of Delayed Flights=4098, Delay Percentage=10.83
2018    Number of Flights=19500, Number of Delayed Flights=3021, Delay Percentage=15.49
2019    Number of Flights=35981, Number of Delayed Flights=4060, Delay Percentage=11.28
2020    Number of Flights=414310, Number of Delayed Flights=56258, Delay Percentage=13.58
```

Business value added:This analysis gives the percentage of flights delayed per year against the total flight count.The airline company can use this data and look for ways to reduce the delay and also can estimate the loss incurred due to delay.Also it shows that 2018 has the highest delay percentage.

10)Percentage of Flights delayed based on day of the week:

This program is same as previous one but it calculates delay% based on day of week.

```
aniketh@ubuntu:~$ $hadoop fs -cat /FlightDirectoryOutput/TotalFlightsByDayOfWeekAndDelayedRa
Friday  Flight Count=103812, Delayed Fligth Count=15670, Delay Percentage=15.09
Monday  Flight Count=82389, Delayed Fligth Count=10518, Delay Percentage=12.77
Saturday        Flight Count=65520, Delayed Fligth Count=13037, Delay Percentage=19.90
Sunday  Flight Count=78506, Delayed Fligth Count=10197, Delay Percentage=12.99
Thursday        Flight Count=103676, Delayed Fligth Count=13607, Delay Percentage=13.12
Tuesday Flight Count=76644, Delayed Fligth Count=7152, Delay Percentage=9.33
Wednesday       Flight Count=96799, Delayed Fligth Count=8953, Delay Percentage=9.25
```

Business value added:This analysis gives the percentage of flights delayed on a particular week day against the total flight count.The airline company can use this data and analyse as to why on particular day there is more delay than compared to other weekdays.It is safe to assume that on Saturday we have  most delay % because of weekend and more people travel and hence,more air traffic.

11)Percentage of Flights delayed based on carrier:
This program is same as previous one but it calculates delay% based on airline companies(unique carrier code).

```
aniketh@ubuntu:~$ $hadoop fs -cat /FlightDirectoryOutput/TotalFlightsDelayedByCarri
9E      Flight Count=23068, Delayed FLight Count=2442, Delay Percentage=10.59
AA      Flight Count=76276, Delayed FLight Count=10864, Delay Percentage=14.24
AS      Flight Count=21110, Delayed FLight Count=4230, Delay Percentage=20.04
B6      Flight Count=24709, Delayed FLight Count=3538, Delay Percentage=14.32
DL      Flight Count=80067, Delayed FLight Count=8055, Delay Percentage=10.06
EV      Flight Count=10564, Delayed FLight Count=1547, Delay Percentage=14.64
F9      Flight Count=12467, Delayed FLight Count=1731, Delay Percentage=13.88
G4      Flight Count=7774, Delayed FLight Count=1484, Delay Percentage=19.09
HA      Flight Count=7035, Delayed FLight Count=634, Delay Percentage=9.01
MQ      Flight Count=26200, Delayed FLight Count=4818, Delay Percentage=18.39
NK      Flight Count=17477, Delayed FLight Count=2271, Delay Percentage=12.99
OH      Flight Count=24309, Delayed FLight Count=4979, Delay Percentage=20.48
OO      Flight Count=71160, Delayed FLight Count=10395, Delay Percentage=14.61
UA      Flight Count=48401, Delayed FLight Count=6019, Delay Percentage=12.44
WN      Flight Count=109770, Delayed FLight Count=9822, Delay Percentage=8.95
YV      Flight Count=17836, Delayed FLight Count=3016, Delay Percentage=16.91
YX      Flight Count=29123, Delayed FLight Count=3289, Delay Percentage=11.29
```

Business value added:This analysis gives the percentage of flights delayed based on airline company against the total flight count.From the above data the airline company Hawain Airlines(HA) has the least delay.

12)Inverted index for all source and destination

I used an inverted index summarization pattern to map all the destinations to their source.

```
aniketh@ubuntu:~$ $hadoop fs -cat /FlightDirectoryOutput/InvertedIndexRealFlight/part-r-0000
ABE:    ORD BNA DTW FLL ATL CLT PGD PIE SFB
ABI:    DFW
ABQ:    ORD SAN PHX MDW LAX PDX SLC MSP DEN SJC DAL SEA JFK HOU IAH MCI DFW BWI OAK ATL SFO
ABR:    MSP
ABY:    ATL
ACT:    DFW
ACV:    LAX DEN SFO
ACY:    TPA MYR FLL PBI RSW MCO
ADK:    ANC
ADQ:    ANC
AEX:    IAH DFW ATL
AGS:    DCA CLT DFW ATL
ALB:    ORD MDW EWR DCA TPA DTW FLL CLT MSP PGD PIE IAD SFB BWI RSW MCO ATL
ALO:    ORD
AMA:    DEN DAL HOU IAH DFW LAS
ANC:    ORD PHX LAX PDX OME HNL ADK MSP DEN SEA ADQ OGG FAI BET JNU SCC CDV OTZ KOA
APN:    DTW PLN
ASE:    ORD PHX LAX SLC MSP DEN IAH DFW ATL SFO
ATL:    JAX GTR HNL LNK STL CLT XNA SLC STT SDF PNS DFW FSD STX LFT CMH FSM OMA EYW MIA LGA
BDL GNV VLD PHF ABQ BUF ASE ONT PHL ABY SFO BUR BMI MBS PHX RDU DAB MKE COS ISP PIA MSN MSP
ELM ELP TRI GPT ILM CHO MLI FNT JFK CHS FWA RNO BWI PBI MLU MDT AUS CID DSM SHV MDW DCA ROA
BHM DTW GRR SAT SAV CRW SJC TTN TLH SRQ HDN CSG VPS BQK EWN SBN SJU MOB EWR GSO EGE PDX GSP
```

Business value added:This data gives the holistic view of the all the sources mapped to their destinations from 2015-2020.

13)Average flying distance and flying air time based on  carrier:

This map reduce program uses the numerical summarization techniques discussed in the class calculate the average time and average flying time of a airline carrier.I used a combiner class to optimise the performance.

```
aniketh@ubuntu:~$ $hadoop fs -cat /FlightDirectoryOutput/AverageFlyingDistancePerCarrier/part-r-00000
9E      AverageCount{Flight Count=22955,Total Time in Air=2227816,Total Distance=10151549,Average Air Time=97.05,Average Distance=442.24}
AA      AverageCount{Flight Count=75472,Total Time in Air=12239404,Total Distance=74931452,Average Air Time=162.17,Average Distance=992.84}
AS      AverageCount{Flight Count=20783,Total Time in Air=4205628,Total Distance=27305462,Average Air Time=202.36,Average Distance=1313.84}
B6      AverageCount{Flight Count=24621,Total Time in Air=4344111,Total Distance=27666805,Average Air Time=176.44,Average Distance=1123.71}
DL      AverageCount{Flight Count=79928,Total Time in Air=11899389,Total Distance=72646392,Average Air Time=148.88,Average Distance=908.90}
EV      AverageCount{Flight Count=10376,Total Time in Air=1136556,Total Distance=5090483,Average Air Time=109.54,Average Distance=490.60}
F9      AverageCount{Flight Count=12312,Total Time in Air=2032663,Total Distance=12504291,Average Air Time=165.10,Average Distance=1015.62}
G4      AverageCount{Flight Count=7663,Total Time in Air=1082558,Total Distance=6809350,Average Air Time=141.27,Average Distance=888.60}
HA      AverageCount{Flight Count=7015,Total Time in Air=880590,Total Distance=5638984,Average Air Time=125.53,Average Distance=803.85}
MQ      AverageCount{Flight Count=25284,Total Time in Air=2580866,Total Distance=11567451,Average Air Time=102.08,Average Distance=457.50}
NK      AverageCount{Flight Count=17333,Total Time in Air=2809376,Total Distance=17638564,Average Air Time=162.08,Average Distance=1017.63}
OH      AverageCount{Flight Count=23999,Total Time in Air=2338817,Total Distance=9874677,Average Air Time=97.45,Average Distance=411.46}
OO      AverageCount{Flight Count=69501,Total Time in Air=7152449,Total Distance=33940244,Average Air Time=102.91,Average Distance=488.34}
UA      AverageCount{Flight Count=48019,Total Time in Air=8892859,Total Distance=57761659,Average Air Time=185.19,Average Distance=1202.89}
WN      AverageCount{Flight Count=107708,Total Time in Air=12827570,Total Distance=78146978,Average Air Time=119.10,Average Distance=725.54}
YV      AverageCount{Flight Count=17473,Total Time in Air=1989447,Total Distance=10352920,Average Air Time=113.86,Average Distance=592.51}
YX      AverageCount{Flight Count=28826,Total Time in Air=3483195,Total Distance=17201779,Average Air Time=120.84,Average Distance=596.75}
```

Business value added:This analysis gives the average flying distance,average airtime by a airline company.We can analyse which airline carrier travelled
long distance and is famous among customers for long distance travel.

# Analysis using Hive:

Creating database:

```
aniketh@ubuntu:~$ hive
Hive Session ID = c22db9c5-7656-4957-86c1-b177449ddb2f

Logging initialized using configuration in jar:file:/usr/local/bin/apa
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new dri
nual loading of the driver class is generally unnecessary.
Hive-on-MR is deprecated in Hive 2 and may not be available in the fut
ases.
Hive Session ID = adcb6f28-3ef4-40c0-a6d7-d942b1b9c014
hive> create schema FlightSystem
    > ;
OK
Time taken: 3.887 seconds
hive> use FlightSystem;
OK
Time taken: 0.229 seconds
```

Creating tables:

```
hive> CREATE EXTERNAL TABLE flightdata(YEAR INT,MONTH INT,DAY_OF_MONTH INT,DAY_OF_WEEK INT,OP_UNIQUE_CARRIER STRING,TAIL_NUM STRING,OP_CARRIER_FL_NUM INT,ORIGIN STRING,
    > DEST STRING,CRS_DEP_TIME INT,DEP_TIME INT,DEP_DELAY INT,TAXI_OUT INT,TAXI_IN INT,CRS_ARR_TIME INT,ARR_TIME INT,ARR_DELAY INT,CANCELLED INT,CANCELLATION_CODE STRIN
G,
    > DIVERTED STRING,CRS_ELAPSED_TIME INT,ACTUAL_ELAPSED_TIME INT,AIR_TIME INT,DISTANCE INT,CARRIER_DELAY INT,WEATHER_DELAY INT,NAS_DELAY INT,SECURITY_DELAY INT,
    > LATE_AIRCRAFT_DELAY INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' tblproperties("skip.header.line.count"="1") ;
OK
Time taken: 1.224 seconds
```

```
hive> create table carriersdata(OP_UNIQUE_CARRIER STRING,DESCRIPTION STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' tblproperties("skip.
header.line.count"="1");
OK
Time taken: 0.945 seconds
```

Mapping our csv file from HDFS to Hive:

```
hive> LOAD DATA INPATH '/FlightDirectory' OVERWRITE INTO TABLE carriersdata;
Loading data to table flightsystem.carriersdata
OK
Time taken: 2.155 seconds
```

## Queries:

1)This is a basic query to find the count of all the records.

select count(*) from flightdata;

```
hive> select count(*) from flightdata;
Query ID = aniketh_20201207222545_bab9965c-63cd-4210-b6b6-acbce60a2502
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
```

```
Ended Job = job_1607399075569_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 22.88 sec
Total MapReduce CPU Time Spent: 22 seconds 880 msec
OK
607346
```

2)Number of flights which have been delayed by more than 20minutes.

select count(*) from flightdata where ARR_DELAY>20;

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 31.88
Total MapReduce CPU Time Spent: 31 seconds 880 msec
OK
65843
```

3)Flights cancelled and their source and destination

select ORIGIN AS SOURCE,DEST AS DESTINATION from flightdata where CANCELLED=1;

```
hive> select ORIGIN AS SOURCE,DEST AS DESTINATION from flightdata where CANCELLED=1;
OK
PDX     LAS
SNA     OAK
DAL     PHX
DAL     SAT
MSY     HOU
ONT     PHX
PHX     ONT
SAN     SFO
SFO     LAX
SFO     SAN
SJU     BWI
```

4)Total cancelled flights by year in ascending order

select YEAR,COUNT(*) as COUNT FROM flightdata where CANCELLED=1 GROUP BY YEAR ORDER BY COUNT;

```
hive> select YEAR,COUNT(*) as COUNT FROM flightdata where CANCELLED=1 GROUP BY YEAR ORDER BY COUNT;
Query ID = aniketh_20201207230118_de6aa660-ffdd-416a-b5f6-654563b390e5
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
```

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 28.69 sec
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 16.28 sec
Total MapReduce CPU Time Spent: 44 seconds 970 msec
OK
2019    180
2017    181
2016    563
2018    579
2015    840
2020    4585
```

5)Airline flight company names:

SELECT DISTINCT DESCRIPTION FROM flightdata INNER JOIN carriersdata on flightdata.OP_UNIQUE_CARRIER=carriersdata.OP_UNIQUE_CARRIER;

```
hive> SELECT DISTINCT DESCRIPTION FROM flightdata INNER JOIN carriersdata on flightdata.OP_UNIQUE_CARRIER=carriersdata.OP_UNIQUE_CARRIER
Query ID = aniketh_20201207233130_225d1658-9477-4e4d-a0ca-4bdfdf401858
Total jobs = 1
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
```

```
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 32.41 sec   HD
Total MapReduce CPU Time Spent: 32 seconds 410 msec
OK
"Midwest Airline
Alaska Airlines Inc.
Allegiant Air
American Airlines Inc.
American Eagle Airlines Inc.
Atlantic Southeast Airlines
Comair Inc.
Delta Air Lines Inc.
Frontier Airlines Inc.
Hawaiian Airlines Inc.
JetBlue Airways
Mesa Airlines Inc.
Pinnacle Airlines Inc.
Skywest Airlines Inc.
Southwest Airlines Co.
Spirit Air Lines
United Air Lines Inc.
```

6)Average distance travelled:

SELECT AVG(DISTANCE) from flightdata;

```
hive> SELECT AVG(DISTANCE) from flightdata;
Query ID = aniketh_20201207233853_dc5606e5-8fe9-4361-8d7d-f4829e200171
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
```

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3
Total MapReduce CPU Time Spent: 30 seconds 640 msec
OK
798.0223414659848
Time taken: 126.521 seconds, Fetched: 1 row(s)
```

7)Total Distance travelled by carrier:

SELECT DESCRIPTION,AVG(DISTANCE) FROM flightdata INNER JOIN carriersdata on flightdata.OP_UNIQUE_CARRIER=carriersdata.OP_UNIQUE_CARRIER GROUP BY OP_UNIQUE_CARRIER;

```
MapReduce Total cumulative CPU time: 24 seconds 680 m
Ended Job = job_1607399075569_0019
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 24
Total MapReduce CPU Time Spent: 24 seconds 680 msec
OK
2015    B6      6025252
2015    DL      5261951
2015    MQ      6935770
2015    OO      3125459
2015    UA      2575625
2015    WN      15221151
2015    YX      5215473
2016    AA      6560975
2016    HA      2955130
2016    MQ      1071551
2016    WN      11653583
2016    YV      2603221
2017    9E      2777832
2017    AA      2589411
2017    DL      12033401
```

# Analysis Using Pig:

1)All the flights cancelled for a source-destination

```
aniketh@ubuntu:~/Desktop/PigScripts$ pig -x local 1CancelledFlightsSrcDest.pig
2020-12-08 12:24:38,936 [main] INFO  org.apache.pig.Main - Apache Pig version 0.12.0-
2020-12-08 12:24:38,939 [main] INFO  org.apache.pig.Main - Logging error messages to:
2020-12-08 12:24:42,765 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup
2020-12-08 12:24:43,310 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation
2020-12-08 12:24:43,312 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation
2020-12-08 12:24:43,316 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HE
```

```
--/FlightDirectory/712352293_T_ONTIME_REPORTING.csv
--/PigOutputDirectory/1.CancelledFlights

--Load the data
RAW_DATA = LOAD '/home/aniketh/Desktop/PigScripts/712352293_T_ONTIME_REPORTING.csv' USING PigStorage(',') AS
        (year:int,month:int,dayofmonth:int,dayofweek: int,
        opuniquecarrier:chararray,tailnum:chararray,opcarrierflightnum:int,
        origin:chararray,destination:chararray,crsdeptime:int,deptime:int,
        depdelay:int,taxiout:int,taxiin:int,crsarrtime:int,arrtime:int,
        arrdelay:int,cancelled:int,cancellationcode:chararray,diverted:chararray,
        crselapsedtime:int,actualelapsedtime:int,airtime:int,distance:int,carrierdelay:int,weatherdelay:int,
        nasdelay:int,securitydelay:int,lateaircraftdelay:int);
--Selecting cancelled source-destination cancelled data
CANCELLEDFLIGHTS = FOREACH RAW_DATA GENERATE origin AS s, destination AS d,cancelled as c;
FILTERCANCELELDSRCDEST =FILTER CANCELLEDFLIGHTS BY c==1;
--Storing the results is HDFS
STORE FILTERCANCELELDSRCDEST INTO 'home/aniketh/Desktop/PigScripts/PigOutputFiles/1CancelledFlightsSrcDest' USING PigStorage(',');
```

```
PDX,LAS,1
SNA,OAK,1
DAL,PHX,1
DAL,SAT,1
MSY,HOU,1
ONT,PHX,1
PHX,ONT,1
SAN,SFO,1
SFO,LAX,1
SFO,SAN,1
SJU,BWI,1
SMF,LAX,1
STL,DAL,1
ATL,BNA,1
BNA,ATL,1
BUR,SJC,1
BUR,SMF,1
```

2)Frequency of Destination:
This scripts generates number of times each destination has been visited on an yearly basis.

```
aniketh@ubuntu:~$ pig -x local /home/aniketh/Desktop/PigScripts/2MostFrequentedDestinations.pig
2020-12-08 15:04:07,898 [main] INFO  org.apache.pig.Main - Apache Pig version 0.12.0-cdh5.3.2 (rexporte
2020-12-08 15:04:07,901 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/aniketh/pig
2020-12-08 15:04:10,962 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/aniketh
2020-12-08 15:04:11,240 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name
2020-12-08 15:04:11,242 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.trac
2020-12-08 15:04:11,252 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - C
```

```
2020-12-08 15:04:52,414 [main] INFO  org.apache.pig.tools.pigstats.SimplePigStats - Script Statistics:

HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.10.1  0.12.0-cdh5.3.2 aniketh 2020-12-08 15:04:15     2020-12-08 15:04:52     GROUP_BY

Success!

Job Stats (time in seconds):
JobId   Alias   Feature Outputs
job_local313007052_0001 FREQUENCY,FREQUENCYOUTPUT,GROUPBYYEAR,RAW_DATA  GROUP_BY,COMBINER       file:///home/aniketh/home/aniketh/Desktop/Pi
tFrequentedPlacesBYYear,

Input(s):
Successfully read records from: "/home/aniketh/Desktop/PigScripts/712352293_T_ONTIME_REPORTING.csv"

Output(s):
Successfully stored records in: "file:///home/aniketh/home/aniketh/Desktop/PigScripts/PigOutputFiles/2MostFrequentedPlacesBYYear"
```

```
--Load the data
RAW_DATA = LOAD '/home/aniketh/Desktop/PigScripts/712352293_T_ONTIME_REPORTING.csv' USING PigStorage(',') AS
        (year:int,month:int,dayofmonth:int,dayofweek: int,
        opuniquecarrier:chararray,tailnum:chararray,opcarrierflightnum:int,
        origin:chararray,destination:chararray,crsdeptime:int,deptime:int,
        depdelay:int,taxiout:int,taxiin:int,crsarrtime:int,arrtime:int,
        arrdelay:int,cancelled:int,cancellationcode:chararray,diverted:chararray,
        crselapsedtime:int,actualelapsedtime:int,airtime:int,distance:int,carrierdelay:int,weatherdelay:int,
        nasdelay:int,securitydelay:int,lateaircraftdelay:int);
--Selecting cancelled source-destination cancelled data
FREQUENCY = FOREACH RAW_DATA GENERATE year as y,destination AS d;
GROUPBYYEAR = GROUP FREQUENCY BY (y,d);
FREQUENCYOUTPUT = FOREACH GROUPBYYEAR GENERATE FLATTEN(group),(COUNT(FREQUENCY)) AS FREQUENTEDDESTINATIONS;
--Storing the results is HDFS
STORE FREQUENCYOUTPUT INTO 'home/aniketh/Desktop/PigScripts/PigOutputFiles/2FrequencyPlacesBYYear' USING PigStorage(',');
```

```
2015,ABE,42
2015,ABI,58
2015,ABQ,240
2015,ABR,6
2015,ABY,6
2015,ACT,59
2015,ACV,17
2015,AEX,59
2015,AGS,37
2015,ALB,97
2015,AMA,37
2015,ANC,8
2015,APN,6
2015,ASE,99
2015,ATL,2622
2015,ATW,17
2015,ATY,6
2015,AUS,586
2015,AVL,92
2015,AVP,34
2015,AZO,41
2015,BDL,246
```

3)Airline popularity-Based on Number of times a carrier is travelled:

```
aniketh@ubuntu:~$ pig -x local /home/aniketh/Desktop/PigScripts/3AIrlinePopularity.pig
2020-12-08 15:27:40,171 [main] INFO  org.apache.pig.Main - Apache Pig version 0.12.0-cdh5.3.2 (rexported) c
2020-12-08 15:27:40,174 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/aniketh/pig_160
2020-12-08 15:27:43,035 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/aniketh/.pi
2020-12-08 15:27:43,269 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is
2020-12-08 15:27:43,270 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker
2020-12-08 15:27:43,274 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Conne
2020-12-08 15:27:45,834 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the s
2020-12-08 15:27:46,039 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES
mnRewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInserter, MergeFi
achFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimpli
2020-12-08 15:27:46,149 [main] INFO  org.apache.pig.newplan.logical.rules.ColumnPruneVisitor - Columns prun
2, $13, $14, $15, $16, $17, $18, $19, $20, $21, $22, $23, $24, $25, $26, $27, $28
2020-12-08 15:27:46,210 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.textoutputfo
```

```
HadoopVersion   PigVersion      UserId  StartedAt       FinishedAt      Features
2.10.1  0.12.0-cdh5.3.2 aniketh 2020-12-08 15:27:47     2020-12-08 15:28:31     GROU

Success!

Job Stats (time in seconds):
JobId   Alias   Feature Outputs
job_local1748688903_0001        AIRLINESFREQUENCY,GROUPBYAIRLINES,POPULARITYOUTPUT,F
ipts/PigOutputFiles/3AIRLINEPOPULARITY,

Input(s):
Successfully read records from: "/home/aniketh/Desktop/PigScripts/712352293_T_ONTIME

Output(s):
Successfully stored records in: "file:///home/aniketh/home/aniketh/Desktop/PigScript
```

```
--Load the data
RAW_DATA = LOAD '/home/aniketh/Desktop/PigScripts/712352293_T_ONTIME_REPORTING.csv' USING PigStorage(',') AS
       (year:int,month:int,dayofmonth:int,dayofweek: int,
       opuniquecarrier:chararray,tailnum:chararray,opcarrierflightnum:int,
       origin:chararray,destination:chararray,crsdeptime:int,deptime:int,
       depdelay:int,taxiout:int,taxiin:int,crsarrtime:int,arrtime:int,
       arrdelay:int,cancelled:int,cancellationcode:chararray,diverted:chararray,
       crselapsedtime:int,actualelapsedtime:int,airtime:int,distance:int,carrierdelay:int,weatherdelay:int,
       nasdelay:int,securitydelay:int,lateaircraftdelay:int);
--Selecting airline frequency
AIRLINESFREQUENCY = FOREACH RAW_DATA GENERATE month as m,opuniquecarrier AS c;
GROUPBYAIRLINES = GROUP AIRLINESFREQUENCY BY (m,c);
POPULARITYOUTPUT = FOREACH GROUPBYAIRLINES GENERATE FLATTEN(group),(COUNT(AIRLINESFREQUENCY)) AS AIRLINEPOPULRITY;
--Storing the results is HDFS
STORE POPULARITYOUTPUT INTO 'home/aniketh/Desktop/PigScripts/PigOutputFiles/3AIRLINEPOPULARITY' USING PigStorage(',');
```

```
1,9E,23068
1,AA,76276
1,AS,21110
1,B6,24709
1,DL,80067
1,EV,10564
1,F9,12467
1,G4,7774
1,HA,7035
1,MQ,26200
1,NK,17477
1,OH,24309
1,OO,71160
1,UA,48401
1,WN,109770
1,YV,17836
1,YX,29123
```

4)Time from airport:
This pig script executes the time travelled by a customer from the airport based on the criteria (taxiout-taxiin) time.The travel time greater 30 minutes is considered to be far.

```
aniketh@ubuntu:~$ pig -x local /home/aniketh/Desktop/PigScripts/4FarFromAirport.pig
2020-12-08 16:07:33,526 [main] INFO  org.apache.pig.Main - Apache Pig version 0.12.0-cdh5.3.2 (rexported)
2020-12-08 16:07:33,529 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/aniketh/pig_16
2020-12-08 16:07:35,102 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/aniketh/.p
2020-12-08 16:07:35,242 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is
2020-12-08 16:07:35,243 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker
2020-12-08 16:07:35,245 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Conn
2020-12-08 16:07:36,689 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the
2020-12-08 16:07:36,868 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULE
mnRewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInserter, MergeF
achFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimpl
2020-12-08 16:07:36,965 [main] INFO  org.apache.pig.newplan.logical.rules.ColumnPruneVisitor - Columns pru
```

```
Success!

Job Stats (time in seconds):
JobId   Alias   Feature Outputs
job_local933836547_0001 FAROUTPUT,GROUPBYCLAUSE,RAW_DATA,TEMP,TIMEFROMAIRPORT   GROUP_
LINEFAROUTPUT,

Input(s):
Successfully read records from: "/home/aniketh/Desktop/PigScripts/712352293_T_ONTIME_R

Output(s):
Successfully stored records in: "file:///home/aniketh/home/aniketh/Desktop/PigScripts/
```

```
--Load the data
RAW_DATA = LOAD '/home/aniketh/Desktop/PigScripts/712352293_T_ONTIME_REPORTING.csv' USING PigStorage(',') AS
        (year:int,month:int,dayofmonth:int,dayofweek: int,
        opuniquecarrier:chararray,tailnum:chararray,opcarrierflightnum:int,
        origin:chararray,destination:chararray,crsdeptime:int,deptime:int,
        depdelay:int,taxiout:int,taxiin:int,crsarrtime:int,arrtime:int,
        arrdelay:int,cancelled:int,cancellationcode:chararray,diverted:chararray,
        crselapsedtime:int,actualelapsedtime:int,airtime:int,distance:int,carrierdelay:int,weatherdelay:int,
        nasdelay:int,securitydelay:int,lateaircraftdelay:int);
--Selecting airline frequency
TIMEFROMAIRPORT = FOREACH RAW_DATA GENERATE destination as d,(taxiout-taxiin) as t;
GROUPBYCLAUSE = GROUP TIMEFROMAIRPORT BY d;

FAROUTPUT = FOREACH GROUPBYCLAUSE { TEMP = FILTER TIMEFROMAIRPORT BY ((t)>=30) ;
                                    GENERATE group,COUNT(TEMP) AS FAR;
                                  };
--Storing the results is HDFS
STORE FAROUTPUT INTO 'home/aniketh/Desktop/PigScripts/PigOutputFiles/4AIRLINEFAROUTPUT' USING PigStorage(',');
```

```
ABE,31
ABQ,56
ANC,62
ABI,16
ABR,12
ABY,2
ACT,10
ACV,17
ACY,3
ADK,0
ADQ,1
AEX,17
AGS,12
ALB,72
ALO,7
AMA,10
```

5)Airlines Delay Count By Year(Based on Carrier Delay or Weather Delay):

This script calculates the number of times an airline has been delayed based on carrier delay or weather delay.

```
aniketh@ubuntu:~$ pig -x local /home/aniketh/Desktop/PigScripts/5AirportsDifferentDelays.pig
2020-12-08 16:57:25,036 [main] INFO  org.apache.pig.Main - Apache Pig version 0.12.0-cdh5.3.2 (rexported) compiled Fe
2020-12-08 16:57:25,040 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/aniketh/pig_160747544501
2020-12-08 16:57:28,253 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/aniketh/.pigbootup no
2020-12-08 16:57:28,579 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated
2020-12-08 16:57:28,581 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is depreca
2020-12-08 16:57:28,587 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to h
2020-12-08 16:57:31,348 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GRO
2020-12-08 16:57:32,545 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[
mnRewrite, GroupByConstParallelSetter, ImplicitSplitInserter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, Merg
achFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter], RULES_DISABLED=[FilterLogicExpressionSimplifier, Part
2020-12-08 16:57:32,899 [main] INFO  org.apache.pig.newplan.logical.rules.ColumnPruneVisitor - Columns pruned for RAW
, $12, $13, $14, $15, $16, $17, $18, $19, $20, $21, $22, $23
2020-12-08 16:57:33,741 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File c
2020-12-08 16:57:33,904 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer
2020-12-08 16:57:33,905 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer
2020-12-08 16:57:34,311 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - session.id is deprecated. Ins
```

```
Success!

Job Stats (time in seconds):
JobId   Alias   Feature Outputs
job_local430417348_0002 DELAYOUTPUT,DELAYS,GROUPBYCLAUSE,RAW_DATA,T
ERENTDELAYSOUTPUT,

Input(s):
Successfully read records from: "/home/aniketh/Desktop/PigScripts/7

Output(s):
Successfully stored records in: "file:///home/aniketh/home/aniketh/

Job DAG:
job_local430417348_0002
```

```
--Load the data
RAW_DATA = LOAD '/home/aniketh/Desktop/PigScripts/712352293_T_ONTIME_REPORTING.csv' USING PigStorage(',') AS
        (year:int,month:int,dayofmonth:int,dayofweek: int,
        opuniquecarrier:chararray,tailnum:chararray,opcarrierflightnum:int,
        origin:chararray,destination:chararray,crsdeptime:int,deptime:int,
        depdelay:int,taxiout:int,taxiin:int,crsarrtime:int,arrtime:int,
        arrdelay:int,cancelled:int,cancellationcode:chararray,diverted:chararray,
        crselapsedtime:int,actualelapsedtime:int,airtime:int,distance:int,carrierdelay:int,weatherdelay:int,
        nasdelay:int,securitydelay:int,lateaircraftdelay:int);
--Selecting airline frequency
DELAYS = FOREACH RAW_DATA GENERATE year as y,carrierdelay as cdelay,weatherdelay as wdelay,nasdelay as ndelay
                      ,securitydelay as sdelay,lateaircraftdelay as ladelay;

GROUPBYCLAUSE = GROUP DELAYS BY y;

DELAYOUTPUT = FOREACH GROUPBYCLAUSE { TEMP = FILTER DELAYS BY (cdelay==1 OR wdelay==1) ;
                                      GENERATE flatten(group),COUNT(TEMP) AS DELAYCOUNT;
                                      };

STORE DELAYOUTPUT INTO 'home/aniketh/Desktop/PigScripts/PigOutputFiles/5AIRLINEDIFFERENTDELAYSOUTPUT' USING PigStorage(',');
```

```
2015,184
2016,90
2017,99
2018,26
2019,91
2020,213
```

# HBASE Analysis:

create 'flightsystem','flight_data'

```
hbase(main):001:0> describe 'flightsystem'
Table flightsystem is ENABLED
flightsystem
COLUMN FAMILIES DESCRIPTION
{NAME => 'flight_data', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => 'NONE'
, TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

1 row(s)

QUOTAS
0 row(s)
Took 4.2168 seconds
```

alter 'flightsystem',{NAME=>'flight_data',VERSIONS=>'700000'}

```
hbase(main):002:0> alter 'flightsystem',{NAME=>'flight_data',VERSIONS=>'700000'}
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.6589 seconds
hbase(main):003:0> describe 'flightsystem'
Table flightsystem is ENABLED
flightsystem
COLUMN FAMILIES DESCRIPTION
{NAME => 'flight_data', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '700000', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', COMPRESSION => '
NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

1 row(s)

QUOTAS
```

Loading csv file to hbase from hdfs:

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=',' -Dimporttsv.columns='HBASE_ROW_KEY,
flight_data:year, flight_data:month, flight_data:dayofmonth ,flight_data:dayofweek ,
flight_data:opuniquecarrier ,flight_data:tailnum ,flight_data:opcarrierflightnum ,flight_data:origin ,
flight_data:destination ,flight_data:crsdeptime ,flight_data:deptime ,flight_data:depdelay ,
flight_data:taxiout ,flight_data:taxiin ,flight_data:crsarrtime ,flight_data:arrtime ,
flight_data:arrdelay ,flight_data:cancelled ,flight_data:cancellationcode ,flight_data:diverted ,
flight_data:crselapsedtime ,flight_data:actualelapsedtime ,flight_data:airtime ,flight_data:distance ,
flight_data:carrierdelay ,flight_data:weatherdelay ,flight_data:nasdelay ,flight_data:securitydelay ,
flight_data: lateaircraftdelay' flightsystem /FlightDirectory/712352293_T_ONTIME_REPORTING.csv

```
aniketh@ubuntu:~$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=',' -Dimporttsv.columns='HBASE_ROW_KEY,
> flight_data:year, flight_data:month, flight_data:dayofmonth ,flight_data:dayofweek ,
> flight_data:opuniquecarrier ,flight_data:tailnum ,flight_data:opcarrierflightnum ,flight_data:origin ,
> flight_data:destination ,flight_data:crsdeptime ,flight_data:deptime ,flight_data:depdelay ,
> flight_data:taxiout ,flight_data:taxiin ,flight_data:crsarrtime ,flight_data:arrtime ,
> flight_data:arrdelay ,flight_data:cancelled ,flight_data:cancellationcode ,flight_data:diverted ,
> flight_data:crselapsedtime ,flight_data:actualelapsedtime ,flight_data:airtime ,flight_data:distance ,
> flight_data:carrierdelay ,flight_data:weatherdelay ,flight_data:nasdelay ,flight_data:securitydelay ,
> flight_data: lateaircraftdelay' flightsystem /FlightDirectory/712352293_T_ONTIME_REPORTING.csv
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/bin/hadoop-2.10.1/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticL
SLF4J: Found binding in [jar:file:/home/aniketh/Downloads/hbase-2.3.3-bin/hbase-2.3.3/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30
Binder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
```

```
aniketh@ubuntu:~$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=',' -Dimporttsv.columns='HBASE_ROW_KEY,
> flight_data:year, flight_data:month, flight_data:dayofmonth ,flight_data:dayofweek ,
> flight_data:opuniquecarrier ,flight_data:tailnum ,flight_data:opcarrierflightnum ,flight_data:origin ,
> flight_data:destination ,flight_data:crsdeptime ,flight_data:deptime ,flight_data:depdelay ,
> flight_data:taxiout ,flight_data:taxiin ,flight_data:crsarrtime ,flight_data:arrtime ,
> flight_data:arrdelay ,flight_data:cancelled ,flight_data:cancellationcode ,flight_data:diverted ,
> flight_data:crselapsedtime ,flight_data:actualelapsedtime ,flight_data:airtime ,flight_data:distance ,
> flight_data:carrierdelay ,flight_data:weatherdelay ,flight_data:nasdelay ,flight_data:securitydelay ,
> flight_data: lateaircraftdelay' flightsystem /FlightDirectory/712352293_T_ONTIME_REPORTING.csv
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/bin/hadoop-2.10.1/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticL
SLF4J: Found binding in [jar:file:/home/aniketh/Downloads/hbase-2.3.3-bin/hbase-2.3.3/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30
Binder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
```

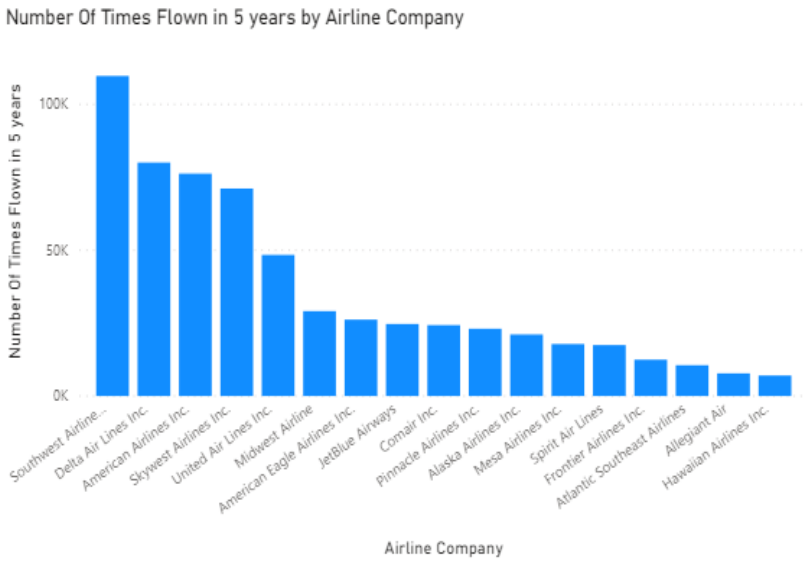
```
2020-12-09 14:37:26,089 INFO  [main] mapreduce.Job: Job job_1607553033653_0001 completed successfully
2020-12-09 14:37:28,121 INFO  [main] mapreduce.Job: bytes read=55610039
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=2
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Job Counters
                Launched map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=265526
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=265526
                Total vcore-milliseconds taken by all map tasks=265526
                Total megabyte-milliseconds taken by all map tasks=271898624
        Map-Reduce Framework
                Map input records=607347
                Map output records=607347
                Input split bytes=135
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
```

File successfully updated to database in hbase:

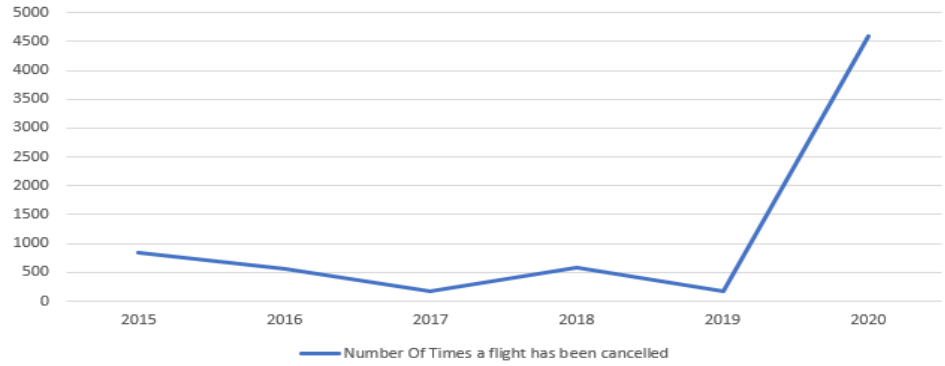
```
hbase(main):008:0> scan 'flightsystem'
ROW                                     COLUMN+CELL
 2015                                   column=flight_data:actualelapsedtime, timestamp=2020-12-09T14:52:03.240, value=129
 2015                                   column=flight_data:airtime, timestamp=2020-12-09T14:52:03.240, value=990
 2015                                   column=flight_data:arrdelay, timestamp=2020-12-09T14:52:03.240, value=0
 2015                                   column=flight_data:arrtime, timestamp=2020-12-09T14:52:03.240, value=-3
 2015                                   column=flight_data:cancellationcode, timestamp=2020-12-09T14:52:03.240, value=0
 2015                                   column=flight_data:cancelled, timestamp=2020-12-09T14:52:03.240, value=
 2015                                   column=flight_data:carrierdelay, timestamp=2020-12-09T14:52:03.240, value=
 2015                                   column=flight_data:crsarrtime, timestamp=2020-12-09T14:52:03.240, value=1707
 2015                                   column=flight_data:crsdeptime, timestamp=2020-12-09T14:52:03.240, value=1342
 2015                                   column=flight_data:crselapsedtime, timestamp=2020-12-09T14:52:03.240, value=145
 2015                                   column=flight_data:dayofmonth, timestamp=2020-12-09T14:52:03.240, value=4
 2015                                   column=flight_data:dayofweek, timestamp=2020-12-09T14:52:03.240, value=WN
```

# Visualisation Reports:

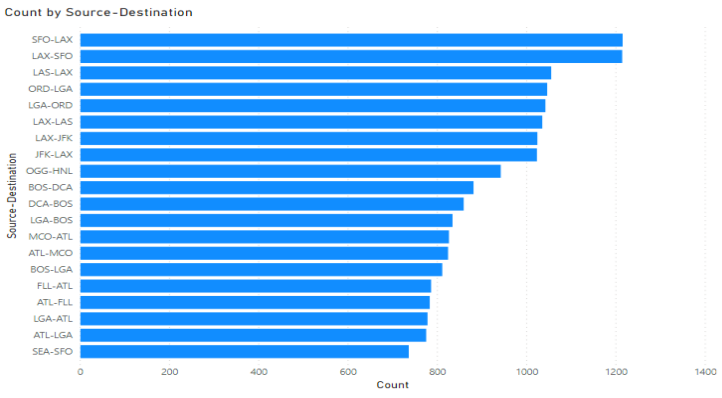
1)AirlineFlightCount:



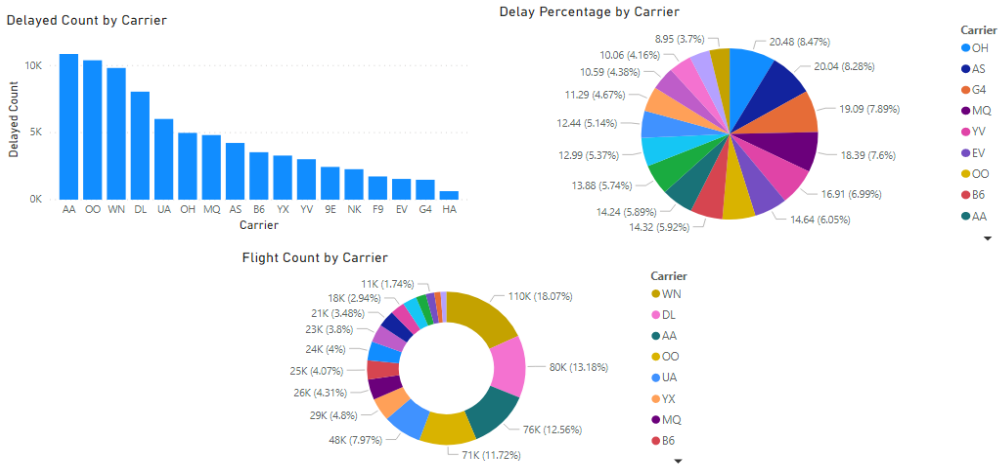

2)Number of times a flight got cancelled based on year




3)Top30SourceDestinations



4)Average Delay Percentage:



# Appendix:

1) Total Flights flown between source and destination.

Main:

```
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 Main {
   public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException{
      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf);
      job.setJarByClass(Main.class);
      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));
      job.setInputFormatClass(TextInputFormat.class);
      job.setOutputFormatClass(TextOutputFormat.class);
      job.setMapperClass(SourceDestinationMapper.class);
      job.setReducerClass(SourceDestinationReducer.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);
      System.exit(job.waitForCompletion(true)?0:1);
   }
}
```

Mapper:

```
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 SourceDestinationMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

   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[7]+"-" + tokens[8];
      word.set(orig-dest);
      context.write(word,one);
   }
```

```
}
```

Reducer:

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

import java.io.IOException;

public class SourceDestinationReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    int sum=0;
    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {


        for(IntWritable v: values){
            sum += v.get();
        }
        context.write(key, new IntWritable(sum));

    }

}
```

2) Top 30 source destination pair.

Main:

```
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 Main {
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException{
        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf);
        job.setJarByClass(TopNMapper.class);

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

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

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

        job.setMapperClass(TopNMapper.class);
```

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

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

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

    }
}
```

Mapper:

```
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 Top30Mapper extends Mapper<Object, Text, NullWritable, Text> {
    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()>20)
            countMap.remove(countMap.firstKey());
    }
    @Override
    protected void cleanup(Context context) throws IOException, InterruptedException {
        for(Text t: countMap.values())
            context.write(NullWritable.get(),t);
    }
}
```

Reducer:

```
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 Top30Reducer 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);
  }
}
```

3)Number of flights cancelled between 2015-2020:
Main:

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


      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf);
      job.setJarByClass(DelayedMapper.class);

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

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

      job.setMapperClass(DelayedMapperCount.class);
      job.setReducerClass(DelayedReducerCount.class);

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

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

   }
}
```

Mapper:

```
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 DelayedMapperCount extends Mapper<LongWritable, Text, NullWritable, IntWritable> {

    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(",");
        //ByPassing First Line
        if(values[0].equals("Year"))return;
        try {
            int delay = Integer.parseInt(values[16]);

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

    }
}
```

Reducer:

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

import java.io.IOException;

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

    @Override
    protected void reduce(NullWritable key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
        }
        context.write(key, new IntWritable(sum));
    }

}
```

4)Airline companies & the number of times they were flown.

Main:

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

      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf);
      job.setJarByClass(CarrierMapper.class);

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

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

      job.setMapperClass(UniqueCarrierCountMapper.class);
      job.setReducerClass(UniqueCarrierCountReducer.class);

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

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

   }
}
```

Mapper:

```
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 UniqueCarrierCountMapper  extends Mapper<LongWritable, Text, Text, IntWritable> {

   @Override
   protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
      Text word = new Text();
      IntWritable one = new IntWritable(1);

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

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

      String carrier = tokens[4];
      word.set(carrier);
      context.write(word,one);
   }


}
```

Reducer:

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

```
import java.io.IOException;

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

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

}
```

5)Airline companies full name-Inner Join:
Main:

```
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);
      job.setJarByClass(FlightMain.class);
      job.setReducerClass(FlightReducer.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(Text.class);

      //Telling the system which mapper gets which files
      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);

   }
}
```

Mapper:

```java
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> {



   @Override
   protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
       Text word = new Text();
       IntWritable one = new IntWritable(1);

       String line = value.toString();
       String[] tokens= line.split("\t");
       //Bypass the firstline
       if(tokens[0].equals("OP_UNIQUE_CARRIER"))return;
       String newKey = tokens[0];
       word.set(newKey);
       //Flag the value for reducer
       String outValue= "B"+tokens[1];
       context.write(word,new Text(outValue));
   }

}

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> {



   @Override
   protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
       Text word = new Text();
       IntWritable one = new IntWritable(1);

       String line = value.toString();
       String[] tokens= line.split(",");
       if(tokens[0].equals("OP_UNIQUE_CARRIER"))
          return;
       String newKey = tokens[0];
       word.set(newKey);
       //FLag the value for reducer
       String outValue= "A"+tokens[1];
       context.write(word,new Text(outValue));
   }
```

```
}
```

Reducer:

```
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="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)));
            }
        }
         //Executes join logic
        executeJoinLogic(context);
    }

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

6)Flight count based on year

Main:

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

      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf);
      job.setJarByClass(YearMain.class);

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

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

      job.setMapperClass(YearMapper.class);
      job.setReducerClass(YearReducer.class);

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

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

   }
}
```

Mapper:

```
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> {

   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);
    }


}
```

Reducer:

```
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> {

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

}
```

7)Number of flights delayed per year:

Main:

```
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();
        Job job = Job.getInstance(conf);
        job.setJarByClass(DelayedMain.class);

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

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

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

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

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

    }
}
```

Mapper:

```
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> {
    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[12]);
            String year = values[0];

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

    }
}
```

Reducer:

```
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> {

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
```

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

}
```

8)Flights Cancelled Per Year:
Main:

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

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);
        job.setJarByClass(CancelledMain.class);

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

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

        job.setMapperClass(CancelledMapper.class);
        job.setReducerClass(CancelledReducer.class);

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

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

    }
}
```

Mapper:

```
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 CancelledMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    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[17];
            String year = values[0];

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

Reducer:

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

import java.io.IOException;

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

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

}
```

9)Percentage of Flights delayed per year

Main:

```
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();
        Job job = Job.getInstance(conf);
        job.setJarByClass(DelayYearMain.class);

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

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

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

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

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

    }
}
```

Mapper:

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

import java.io.IOException;


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

    private DelayCountTuple tuple = new DelayCountTuple();

    @Override
    protected void map(k 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[16]);
            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);
    }
}
```

Reducer:

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

```
        int total=0;
        int delayedTotal=0;
        for(DelayCountTuple d: values){
            total += d.getFlightCount();
            delayedTotal +=d.getDelayedFlightCount();
        }
        double percent = ((double)delayedTotal/total)*100;
        result.setDelayedFlightCount(delayedTotal);
        result.setFlightCount(total);
        result.setDelayPercent(percent);
        context.write(key,result);
    }
}
```

Writable:

```
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  "Number of Flights=" + flightCount +
                ", Number of Delayed Flights=" + delayedFlightCount +
                ", Delay Percentage=" + 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();

    }
}
```

10)Percentage of Flights delayed based on day of the week:

Main:

```
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 DelayDayOfWeekMain {

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

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

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

        job.setMapperClass(DelayDayOfWeekMapper.class);
        job.setCombinerClass(DelayDayOfWeekReducer.class);
        job.setReducerClass(DelayDayOfWeekReducer.class);

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

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

Mapper:

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

```
import java.io.IOException;


public class DelayDayOfWeekMapper 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;
      if(tokens[3].equals("DAY_OF_WEEK"))
         return;
      int dayOfWeek=Integer.parseInt(tokens[3]);
      if(dayOfWeek>8)
         return;
      String day = days[dayOfWeek];//DayOfWeek


      try {
         int delay = Integer.parseInt(tokens[16]);


         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);
   }
}
```

Reducer:

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

import java.io.IOException;

public class DelayDayOfWeekReducer 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);
    }
}
```

Writable:

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

import java.io.IOException;

public class DelayDayOfWeekReducer 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);
    }
}
```

11)Percentage of Flights delayed based on carrier

Main:

```
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();
        Job job = Job.getInstance(conf);
        job.setJarByClass(DelayCarrierMain.class);

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

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

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

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

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

    }
}
```

Mapper:

```
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;
        if(tokens[4].equals("OP_UNIQUE_CARRIER"))return;
        String carrier = tokens[4];

        try {
            int delay = Integer.parseInt(tokens[16]);
```

```
            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);
    }
}
```

Reducer:

```
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);
    }
}
```

Writable:

```
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  "Flight Count=" + flightCount +
                ", Delayed FLight Count=" + delayedFlightCount +
                ", Delay Percentage=" + 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();

    }
}
```

12Inverted index for all source and destination

Main:

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

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

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

      job.setMapperClass(InvertedIndexMapper.class);
      job.setReducerClass(InvertedIndexReducer.class);

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

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

   }
}
```

Mapper:

```
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 InvertedIndexMapper 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[7];
      String dest = tokens[8];
      context.write(new Text(src+":"),new Text(dest));
   }
```

```
}
```

Reducer:

```
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 InvertedIndexReducer extends Reducer<Text, Text, Text, Text> {

   Set<String> set = new HashSet<>();
   @Override
   protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
      set.clear();
      StringBuilder sb = new StringBuilder("");
      for(Text v: values){
         set.add(v.toString());
      }
      for(String v: set){
         sb.append(v);
         sb.append(" ");
      }
      context.write(key, new Text(sb.toString()));
   }

}
```

13) Average flying distance and flying air time based on carrier
Main:

```
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();
      Job job = Job.getInstance(conf);
      job.setJarByClass(AverageMain.class);

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

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

      job.setMapOutputKeyClass(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);

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

    }
}
```

Mapper:

```
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[4];
        int distance=0;
        int flightTime =0;

        try {
            distance = Integer.parseInt(tokens[23]);//Distance

            flightTime = Integer.parseInt(tokens[21]);//Actual Elapsed Time
        } catch (Exception e){
            return;
        }
        tuple.setFlightCount(1);
        tuple.setDistCount(distance);
        tuple.setAirTime(flightTime);

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

Reducer:

```
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 avg: values){
                        totalFlight += avg.getFlightCount();
                        totalDist += avg.getDistCount();
                        totalAirTime += avg.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);
        }
}
```

Combiner:

```
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 avg: values){
            totalFlight += avg.getFlightCount();
            totalDist += avg.getDistCount();
            totalAirTime += avg.getAirTime();
        }

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

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

```
}
```

Writable:

```
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 "AverageCount{" +
                "Flight Count=" + flightCount +
                ",Total Time in Air=" + airTime +
                ",Total Distance=" + distCount +
                ",Average Air Time=" + String.format("%.2f", averageAirTime)+
                ",Average Distance=" + String.format("%.2f", averageDist) +
                '}';
    }

    @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();

    }
}
```

## Resources:


https://www.researchgate.net/publication/339104000_Big_data_analytic_on_aviation_trends_in_US_and_determining_possible_enhancements_for_flight_delays

https://blog.datumize.com/9-incredible-ways-data-analytics-is-transforming-airlines

https://www.transtats.bts.gov/Fields.asp?Table_ID=236

https://www.investopedia.com/articles/investing/061015/how-southwest-different-other-airlines.asp#:~:text=Besides%20motivated%20employees%20and%20great,as%20Delta%20and%20American%20Airlines.