FINAL PROJECT BIG DATA ENGINEERING (INFO7250)

NAME - SHALINI CHANDRA (NUID - 001062801)

CONTENTS

Overview about the Dataset

I am using the data on **Airline On-Time Statistics and Delay Causes from**. The data is available from year 1987 to 2008. It has many columns which can be helpful in analysis. Due to huge volume of data, I am taking only 3 years data from 2006 to 2008

Dataset Description: (Airline Dataset 2006 - 2008)

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

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

Variable descriptions

	Name	Description
1	Year	2006-2008
2	Month	1-12
3	DayofMonth	1-31
4	DayOfWeek	1 (Monday) - 7 (Sunday)
5	DepTime	actual departure time (local, hhmm)
6	CRSDepTime	scheduled departure time (local, hhmm)
7	ArrTime	actual arrival time (local, hhmm)
8	CRSArrTime	scheduled arrival time (local, hhmm)
9	UniqueCarrier	unique carrier code
10	FlightNum	flight number
11	TailNum	plane tail number
12	ActualElapsedTime	in minutes
13	CRSElapsedTime	in minutes
14	AirTime	in minutes
15	ArrDelay	arrival delay, in minutes
16	DepDelay	departure delay, in minutes
17	Origin	origin IATA airport code

18	Dest	destination IATA airport code
19	Distance	in miles
20	TaxiIn	taxi in time, in minutes
21	TaxiOut	taxi out time in minutes
22	Cancelled	was the flight cancelled?
23	CancellationCode	reason for cancellation (A = carrier, B = weather, C = NAS, D = security)
24	Diverted	1 = yes, 0 = no
25	CarrierDelay	in minutes
26	WeatherDelay	in minutes
27	NASDelay	in minutes
28	SecurityDelay	in minutes
29	LateAircraftDelay	in minutes

- **Loading Data to HDFS** (Size - 2.5 GB)
  - Shell script for converting bz2 file bz2 file is (a TAR archive, compressed with a Burrows-Wheeler (BZ2) compression algorithm, along with Run-Length Encoding (RLE) for better compression) to csv and combining all data
  - Due to large volume of data, I took dataset year from 2006 -2008

combined

```
for i in {2001..2008}; do
  echo "removing header start for " $i
  bzip2 -dk /home/shalini/Desktop/BIgDataProject/Data/$i.csv.bz2
  echo "removing header done for " $i
done
echo "now combining all files"
cat /home/shalini/Desktop/BIgDataProject/Data/*csv > /home/shalini/Desktop/BIgDataProject/combined.csv

done

```


```
shalini@ubuntu:~/Desktop/BIgDataProject$ ./loadAirlineData.sh
removing header start for  2001
removing header done for  2001
removing header start for  2002
removing header done for  2002
removing header start for  2003
removing header done for  2003
removing header start for  2004
removing header done for  2004
removing header start for  2005
removing header done for  2005
removing header start for  2006
removing header done for  2006
removing header start for  2007
removing header done for  2007
removing header start for  2008
removing header done for  2008
now combining all files
cat: write error: No space left on device
./loadAirlineData.sh: line 9: syntax error near unexpected token `done'
./loadAirlineData.sh: line 9: `done'
shalini@ubuntu:~/Desktop/BIgDataProject$ ./loadAirlineData.sh
now combining all files
now combining all files
now combining all files
```

Data

```
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /airlinedata |head
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-11-23 18:44:39,291 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueCarrier,FlightNum,TailNum,ActualElapsedTime,CRSElapsedTime,AirTime,ArrDelay,DepDelay,Origin,
Dest,Distance,TaxiIn,TaxiOut,Cancelled,CancellationCode,Diverted,CarrierDelay,WeatherDelay,NASDelay,SecurityDelay,LateAircraftDelay
2006,1,11,3,743,745,1024,1018,US,343,N657AW,281,273,223,6,-2,ATL,PHX,1587,45,13,0,,0,0,0,0,0,0
2006,1,11,3,1053,1053,1313,1318,US,613,N834AW,260,265,214,-5,0,ATL,PHX,1587,27,19,0,,0,0,0,0,0,0
2006,1,11,3,1915,1915,2110,2133,US,617,N605AW,235,258,220,-23,0,ATL,PHX,1587,4,11,0,,0,0,0,0,0,0
2006,1,11,3,1753,1755,1925,1933,US,300,N312AW,152,158,126,-8,-2,AUS,PHX,872,16,10,0,,0,0,0,0,0,0
2006,1,11,3,824,832,1015,1015,US,765,N309AW,171,163,132,0,-8,AUS,PHX,872,27,12,0,,0,0,0,0,0,0
2006,1,11,3,627,630,834,832,US,295,N733UW,127,122,108,2,-3,BDL,CLT,644,6,13,0,,0,0,0,0,0,0
2006,1,11,3,825,820,1041,1021,US,349,N177UW,136,121,111,20,5,BDL,CLT,644,4,21,0,,0,0,0,20,0,0
2006,1,11,3,942,945,1155,1148,US,356,N404US,133,123,121,7,-3,BDL,CLT,644,4,8,0,,0,0,0,0,0,0
2006,1,11,3,1239,1245,1438,1445,US,775,N722UW,119,120,103,-7,-6,BDL,CLT,644,4,12,0,,0,0,0,0,0,0
cat: Unable to write to output stream.
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$
```

MAPREDUCE ON HADOOP ANALYSIS

Number of rows in the dataset (Time taken – 1 min)

Source Code

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

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

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


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


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

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


        job.setMapperClass(RowsCountMapper.class);
        job.setReducerClass(RowsCountReducer.class);
```

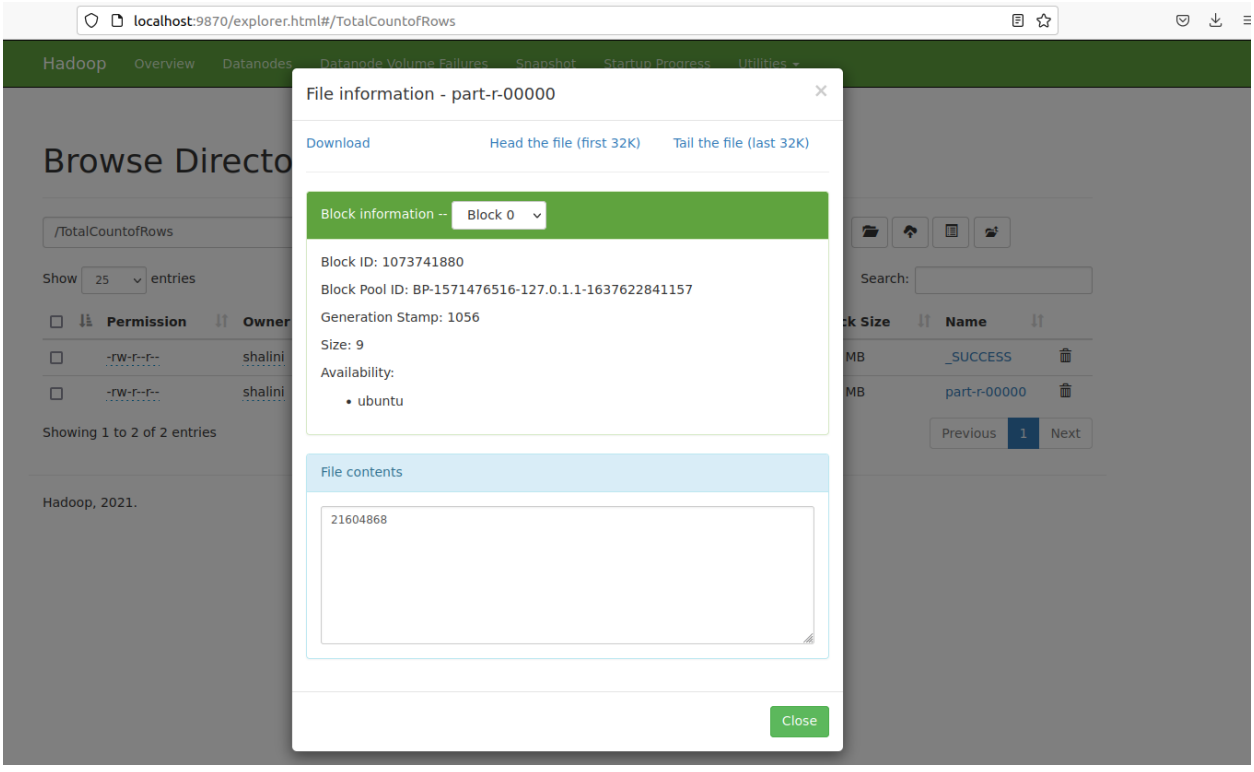
```java
import java.io.IOException;
/**
 *
 * @author shali
 */
public class RowsCountMapper extends Mapper<LongWritable, Text, NullWritable, IntWritable> {

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

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


        context.write(NullWritable.get(),one);
    }

}
```

```java
/**
 *
 * @author shali
 */
public class RowsCountReducer extends Reducer<NullWritable, IntWritable, NullWritable, IntWritable> {

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

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

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

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


}
```

```
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/sbin$ ./start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as shalini in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ubuntu]
2021-11-30 16:33:48,198 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting resourcemanager
Starting nodemanagers
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/sbin$ cd bun/
bash: cd: bun/: No such file or directory
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/sbin$ cd ..
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1$ cd bin/
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop jar ~/Desktop/Rows_Count-1.0-SNAPSHOT.jar com.mycompany.rows_count.RowsCount_MR /airlinedata /TotalCountofRows
2021-11-30 16:37:41,925 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2021-11-30 16:37:42,904 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2021-11-30 16:37:42,979 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2021-11-30 16:37:42,979 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2021-11-30 16:37:43,125 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application
with ToolRunner to remedy this.
2021-11-30 16:37:43,259 INFO input.FileInputFormat: Total input files to process : 1
2021-11-30 16:37:43,421 INFO mapreduce.JobSubmitter: number of splits:16
2021-11-30 16:37:43,565 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1828170516_0001
2021-11-30 16:37:43,565 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-11-30 16:37:43,725 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2021-11-30 16:37:43,725 INFO mapreduce.Job: Running job: job_local1828170516_0001
2021-11-30 16:37:43,730 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2021-11-30 16:37:43,738 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2021-11-30 16:37:43,738 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
2021-11-30 16:37:43,741 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2021-11-30 16:37:43,805 INFO mapred.LocalJobRunner: Waiting for map tasks
2021-11-30 16:37:43,806 INFO mapred.LocalJobRunner: Starting task: attempt_local1828170516_0001_m_000000_0
2021-11-30 16:37:43,829 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2021-11-30 16:37:43,832 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
```

Output

Total number of rows in the dataset **21604868**




```
                Bytes Written=9
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /TotalCountofRows/part-r-00000 |head
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-11-30 16:41:20,779 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
21604868
```

- **Unique Carrier Count in the dataset (Time taken – 2 mins)**

Code

```
 */
public class CarrierMain {
    public static void main(String[] args) throws IOException, InterruptedException, ClassNot


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

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


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

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

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

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

```
 */
public class CarrierMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

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

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

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

        String carrier = tokens[8];
        word.set(carrier);
        context.write(word,one);
    }
```

```
 */
public class CarrierReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

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

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

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

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


}
```

Output

```
                Bytes Written=217
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /UniqueCarrierCount/part-r-00000
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-12-08 18:03:12,297 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your p
9E      521059
AA      1882339
AQ      89547
AS      470691
B6      543273
CO      930995
DL      1433906
EV      839952
F9      283703
FL      762488
HA      170174
MQ      1581275
NW      1195058
OH      709493
OO      1713150
TZ      19602
UA      1439525
US      1443880
WN      3469946
XE      1250753
YV      854056
```

- **Origin Airport Count in the dataset (Time taken – 1 min)**

CODE (Main, Mapper, Reducer)

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


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

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

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

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

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

    job.setMapperClass(Airport_Count_Mapper.class);
    job.setCombinerClass(Airport_Count_Reducer.class);
    job.setReducerClass(Airport_Count_Reducer.class);

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

vcompany.biodata_final_project.Airport_Count_Main › main ›

```
 *
 * @author shali
 */
public class Airport_Count_Mapper extends Mapper<LongWritable, Text, Text, IntWritable> {

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

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

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

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


}
```

```java
 *
 * @author shali
 */
public class Airport_Count_Reducer extends Reducer<Text, IntWritable, Text, IntWritable> {

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

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

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

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

}
```

Jar file on HDFS

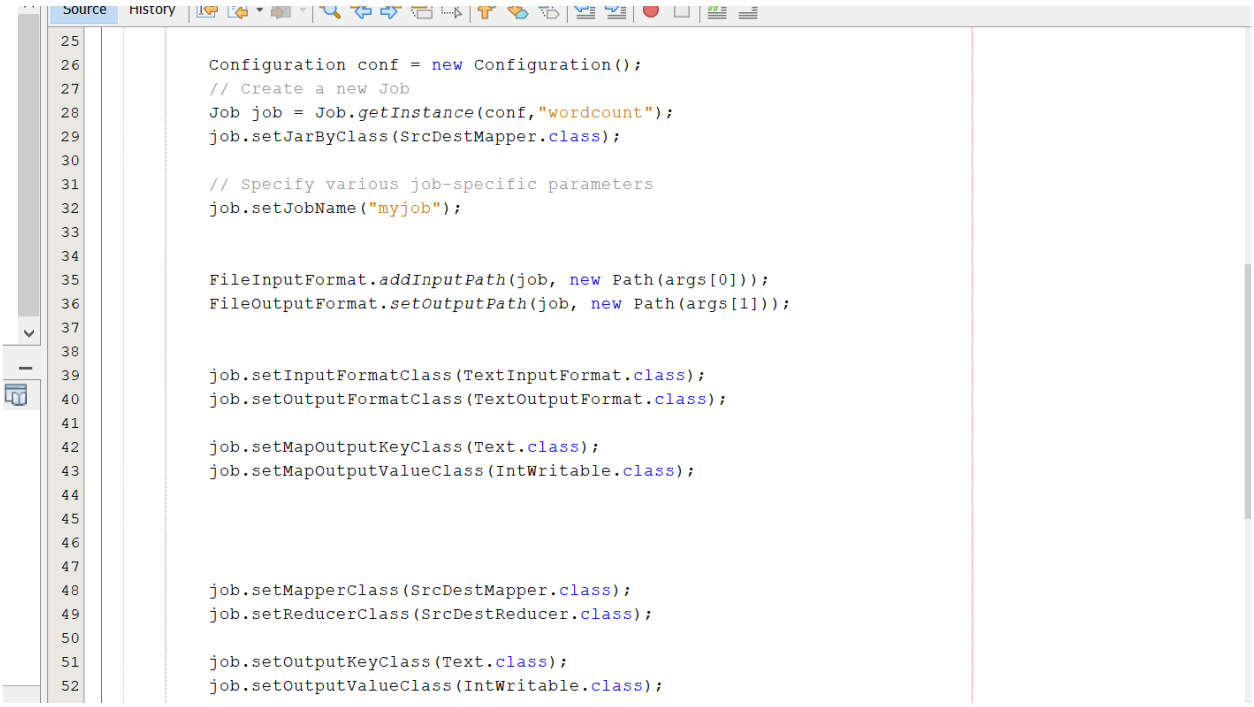
```
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop jar ~/Desktop/BigData_Final_Project-1.0-SNAPSHOT.jar com.mycompany.bigdata_final_project.Airport_Count_Main /ai
rlinedata /airlinecount
2021-11-23 18:48:57,779 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2021-11-23 18:48:59,479 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2021-11-23 18:48:59,786 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2021-11-23 18:48:59,786 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2021-11-23 18:49:00,238 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application
with ToolRunner to remedy this.
2021-11-23 18:49:00,498 INFO input.FileInputFormat: Total input files to process : 1
2021-11-23 18:49:00,749 INFO mapreduce.JobSubmitter: number of splits:16
2021-11-23 18:49:01,348 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local54258680_0001
2021-11-23 18:49:01,349 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-11-23 18:49:01,920 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2021-11-23 18:49:01,922 INFO mapreduce.Job: Running job: job_local54258680_0001
2021-11-23 18:49:01,958 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2021-11-23 18:49:02,011 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2021-11-23 18:49:02,011 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
2021-11-23 18:49:02,043 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2021-11-23 18:49:02,563 INFO mapred.LocalJobRunner: Waiting for map tasks
2021-11-23 18:49:02,566 INFO mapred.LocalJobRunner: Starting task: attempt_local54258680_0001_m_000000_0
2021-11-23 18:49:02,634 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2021-11-23 18:49:02,638 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
2021-11-23 18:49:02,716 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2021-11-23 18:49:02,736 INFO mapred.MapTask: Processing split: hdfs://127.0.0.1:9000/airlinedata:0+134217728
2021-11-23 18:49:03,025 INFO mapreduce.Job: Job job_local54258680_0001 running in uber mode : false
2021-11-23 18:49:03,026 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2021-11-23 18:49:03,027 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2021-11-23 18:49:03,027 INFO mapred.MapTask: soft limit at 83886080
2021-11-23 18:49:03,028 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2021-11-23 18:49:03,028 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
```

Output

```
                Bytes Written=3614
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat  /airlinecount/part-r-00000 |head
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-11-23 18:53:21,911 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
ABE     15696
ABI     8341
ABQ     119571
ABY     3776
ACK     1006
ACT     6800
ACV     11250
ACY     1616
ADK     309
ADQ     1999
```

- **Source and Destination Pairs count (Time taken – 1 min)**

MR Code



```
25
26          Configuration conf = new Configuration();
27          // Create a new Job
28          Job job = Job.getInstance(conf,"wordcount");
29          job.setJarByClass(SrcDestMapper.class);
30
31          // Specify various job-specific parameters
32          job.setJobName("myjob");
33
34
35          FileInputFormat.addInputPath(job, new Path(args[0]));
36          FileOutputFormat.setOutputPath(job, new Path(args[1]));
37
38
39          job.setInputFormatClass(TextInputFormat.class);
40          job.setOutputFormatClass(TextOutputFormat.class);
41
42          job.setMapOutputKeyClass(Text.class);
43          job.setMapOutputValueClass(IntWritable.class);
44
45
46
47
48          job.setMapperClass(SrcDestMapper.class);
49          job.setReducerClass(SrcDestReducer.class);
50
51          job.setOutputKeyClass(Text.class);
52          job.setOutputValueClass(IntWritable.class);
```

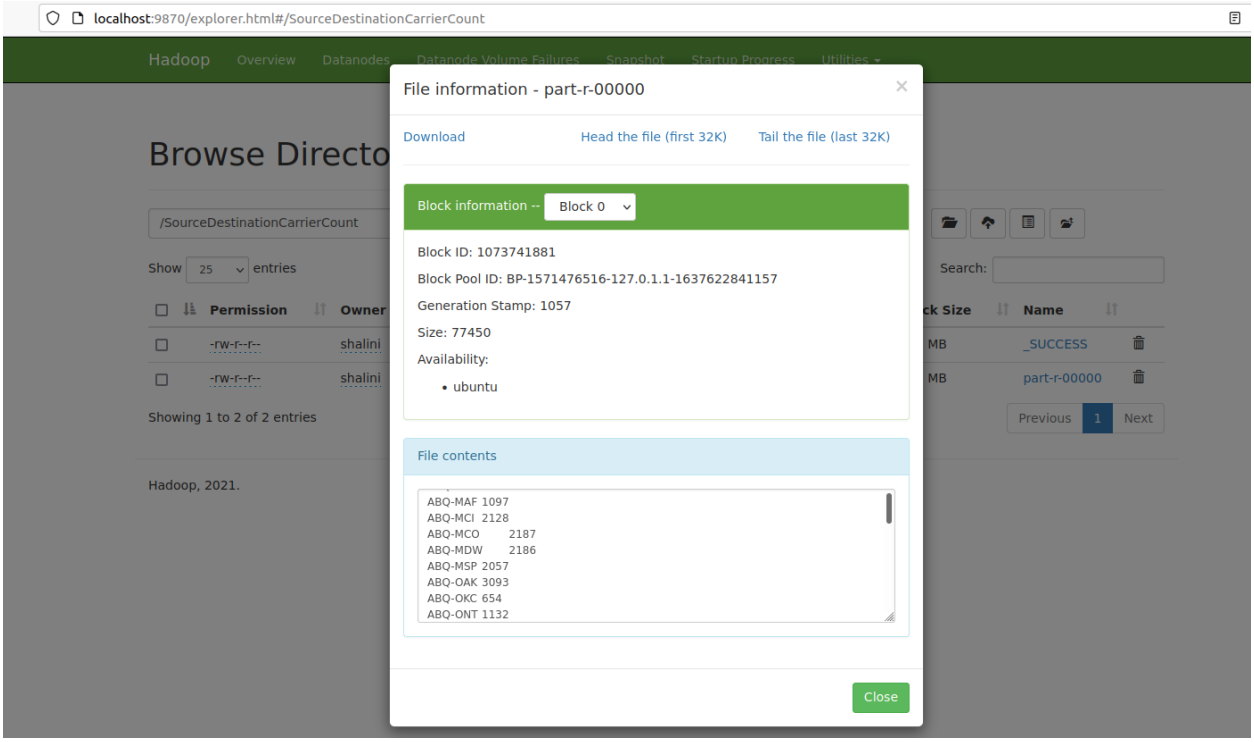
```
/**
 *
 * @author shali
 */
public class SrcDestMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

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

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

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


}
```

```java
import java.io.IOException;
/**
 *
 * @author shali
 */
public class SrcDestReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

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

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

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

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


}
```

Running JAR on HDFS

```
21604868
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop jar ~/Desktop/Source_Desti_Pairs_Count-1.0-SNAPSHOT.jar com.mycompany.source_desti_pairs_count.SrcDestMR /airli
nedata /SourceDestinationCarrierCount
2021-11-30 16:45:38,997 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2021-11-30 16:45:39,603 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2021-11-30 16:45:39,756 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2021-11-30 16:45:39,757 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2021-11-30 16:45:39,980 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application
with ToolRunner to remedy this.
2021-11-30 16:45:40,097 INFO input.FileInputFormat: Total input files to process : 1
2021-11-30 16:45:40,247 INFO mapreduce.JobSubmitter: number of splits:16
2021-11-30 16:45:40,559 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local54590920_0001
2021-11-30 16:45:40,560 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-11-30 16:45:40,727 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2021-11-30 16:45:40,728 INFO mapreduce.Job: Running job: job_local54590920_0001
2021-11-30 16:45:40,735 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2021-11-30 16:45:40,746 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2021-11-30 16:45:40,746 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
2021-11-30 16:45:40,747 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2021-11-30 16:45:40,929 INFO mapred.LocalJobRunner: Waiting for map tasks
2021-11-30 16:45:40,930 INFO mapred.LocalJobRunner: Starting task: attempt_local54590920_0001_m_000000_0
2021-11-30 16:45:40,967 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2021-11-30 16:45:40,968 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
2021-11-30 16:45:41,016 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2021-11-30 16:45:41,031 INFO mapred.MapTask: Processing split: hdfs://127.0.0.1:9000/airlinedata:0+134217728
2021-11-30 16:45:41,111 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2021-11-30 16:45:41,111 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2021-11-30 16:45:41,111 INFO mapred.MapTask: soft limit at 83886080
2021-11-30 16:45:41,112 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2021-11-30 16:45:41,112 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2021-11-30 16:45:41,117 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2021-11-30 16:45:41,734 INFO mapreduce.Job: Job job_local54590920_0001 running in uber mode : false
2021-11-30 16:45:41,735 INFO mapreduce.Job:  map 0% reduce 0%
2021-11-30 16:45:43,972 INFO mapred.LocalJobRunner:
2021-11-30 16:45:43,973 INFO mapred.MapTask: Starting flush of map output
2021-11-30 16:45:43,973 INFO mapred.MapTask: Spilling map output
2021-11-30 16:45:43,973 INFO mapred.MapTask: bufstart = 0; bufend = 17187192; bufvoid = 104857600
2021-11-30 16:45:43,973 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 20485336(81941344); length = 5729061/6553600
2021-11-30 16:45:45,002 INFO mapred.MapTask: Finished spill 0
2021-11-30 16:45:45,017 INFO mapred.Task: Task:attempt_local54590920_0001_m_000000_0 is done. And is in the process of committing
2021-11-30 16:45:45,028 INFO mapred.LocalJobRunner: map
2021-11-30 16:45:45,028 INFO mapred.Task: Task 'attempt_local54590920_0001_m_000000_0' done.
2021-11-30 16:45:45,037 INFO mapred.Task: Final Counters for attempt_local54590920_0001_m_000000_0: Counters: 23
```

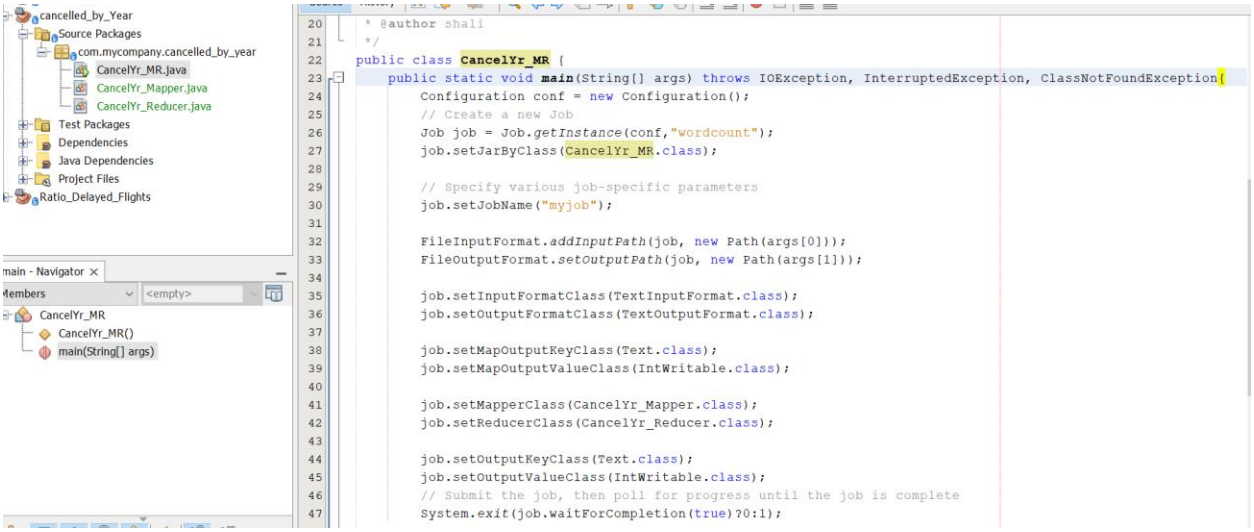
Output


localhost:9870/explorer.html#/SourceDestinationCarrierCount
Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directo
/SourceDestinationCarrierCount
Show 25 entries
Search:
Permission
Owner
Name
-rw-r--r--
shalini
_SUCCESS
part-r-00000
Showing 1 to 2 of 2 entries
Previous
1
Next
Hadoop, 2021.
File information - part-r-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information --
Block 0
Block ID: 1073741881
Block Pool ID: BP-1571476516-127.0.1.1-1637622841157
Generation Stamp: 1057
Size: 77450
Availability:
ubuntu
File contents
ABQ-MAF 1097
ABQ-MCI 2128
ABQ-MCO 2187
ABQ-MDW 2186
ABQ-MSP 2057
ABQ-OAK 3093
ABQ-OKC 654
ABQ-ONT 1132
Close

```
ABE-BHM 1
ABE-BWI 1
ABE-CLE 2971
ABE-CLT 1190
ABE-CVG 1989
ABE-DTW 2009
ABE-FWA 1
ABE-JFK 10
ABE-LGA 14
ABE-ORD 4497
ABE-PHL 2
ABE-SBN 1
ABI-DFW 8340
ABI-LAX 1
ABQ-AMA 1681
ABQ-ATL 3465
ABQ-AUS 847
ABQ-BWI 1276
ABQ-CLE 35
ABQ-CVG 1295
ABQ-DAL 7891
ABQ-DEN 10142
ABQ-DFW 8655
ABQ-ELP 2757
ABQ-EWR 557
ABQ-GJT 2
ABQ-HOU 3054
ABQ-IAD 847
ABQ-IAH 6861
ABQ-LAS 7199
ABQ-LAX 7319
ABQ-LBB 1094
ABQ-MAF 1097
ABQ-MCI 2128
ABQ-MCO 2187
ABQ-MDW 2186
ABQ-MSP 2057
ABQ-OAK 3093
ABQ-OKC 654
ABQ-ONT 1132
ABQ-ORD 2188
ABQ-PDX 1095
ABQ-PHX 16588
ABQ-PSP 1
ABQ-SAN 3174
ABQ-SAT 831
ABQ-SEA 1659
```

- **Cancelled flights per year (Time taken – 2 mins 4 secs)**

Code



```java
package com.mycompany.cancelled_by_year;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

/**
 *
 * @author shali
 */
public class CancelYr_Reducer extends Reducer<Text, IntWritable, Text, IntWritable> {

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

        int sum=0;
        for(IntWritable v: values){
            sum += v.get();
        }

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


    }
```

Running JAR ON HDFS

```
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop jar ~/Desktop/cancelled_by_Year-1.0-SNAPSHOT.jar com.mycompany.cancelled_by_year.CancelYr_MR /airlinedata /canc
elledFlightYearly
2021-11-28 14:46:02,798 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2021-11-28 14:46:05,031 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2021-11-28 14:46:05,453 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2021-11-28 14:46:05,453 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2021-11-28 14:46:05,952 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application
with ToolRunner to remedy this.
2021-11-28 14:46:06,243 INFO input.FileInputFormat: Total input files to process : 1
2021-11-28 14:46:06,654 INFO mapreduce.JobSubmitter: number of splits:16
2021-11-28 14:46:07,513 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local812642352_0001
2021-11-28 14:46:07,513 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-11-28 14:46:08,036 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2021-11-28 14:46:08,038 INFO mapreduce.Job: Running job: job_local812642352_0001
2021-11-28 14:46:08,053 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2021-11-28 14:46:08,111 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2021-11-28 14:46:08,112 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
2021-11-28 14:46:08,117 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2021-11-28 14:46:08,323 INFO mapred.LocalJobRunner: Waiting for map tasks
2021-11-28 14:46:08,327 INFO mapred.LocalJobRunner: Starting task: attempt_local812642352_0001_m_000000_0
2021-11-28 14:46:08,414 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2021-11-28 14:46:08,418 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: fals
e
2021-11-28 14:46:08,509 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2021-11-28 14:46:08,519 INFO mapred.MapTask: Processing split: hdfs://127.0.0.1:9000/airlinedata:0+134217728
2021-11-28 14:46:08,744 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2021-11-28 14:46:08,745 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2021-11-28 14:46:08,745 INFO mapred.MapTask: soft limit at 83886080
2021-11-28 14:46:08,745 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2021-11-28 14:46:08,746 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2021-11-28 14:46:08,777 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2021-11-28 14:46:09,099 INFO mapreduce.Job: Job job_local812642352_0001 running in uber mode : false
2021-11-28 14:46:09,101 INFO mapreduce.Job:  map 0% reduce 0%
2021-11-28 14:46:18,962 INFO mapred.LocalJobRunner:
2021-11-28 14:46:18,963 INFO mapred.MapTask: Starting flush of map output
2021-11-28 14:46:18,963 INFO mapred.MapTask: Spilling map output
2021-11-28 14:46:18,964 INFO mapred.MapTask: bufstart = 0; bufend = 227016; bufvoid = 104857600
2021-11-28 14:46:18,964 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 26113504(104454016); length = 100893/6553600
2021-11-28 14:46:19,163 INFO mapred.MapTask: Finished spill 0
2021-11-28 14:46:19,220 INFO mapred.Task: Task:attempt_local812642352_0001_m_000000_0 is done. And is in the process of committing
2021-11-28 14:46:19,251 INFO mapred.LocalJobRunner: map
2021-11-28 14:46:19,256 INFO mapred.Task: Task 'attempt_local812642352_0001_m_000000_0' done.
```

Output

```
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /cancelledFlightYearly/part-r-00000 |head
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-11-28 14:49:10,690 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2006    121934
2007    160748
2008    137434
```

Most of the flights cancelled in 2007

- **Flight Delayed count yearly (2 mins)**
  MR Code

DelayMR.java | Delay_Mapper.java | DelayYearMapper.java | DelayYearReducer.java | Delay_Reducer.java

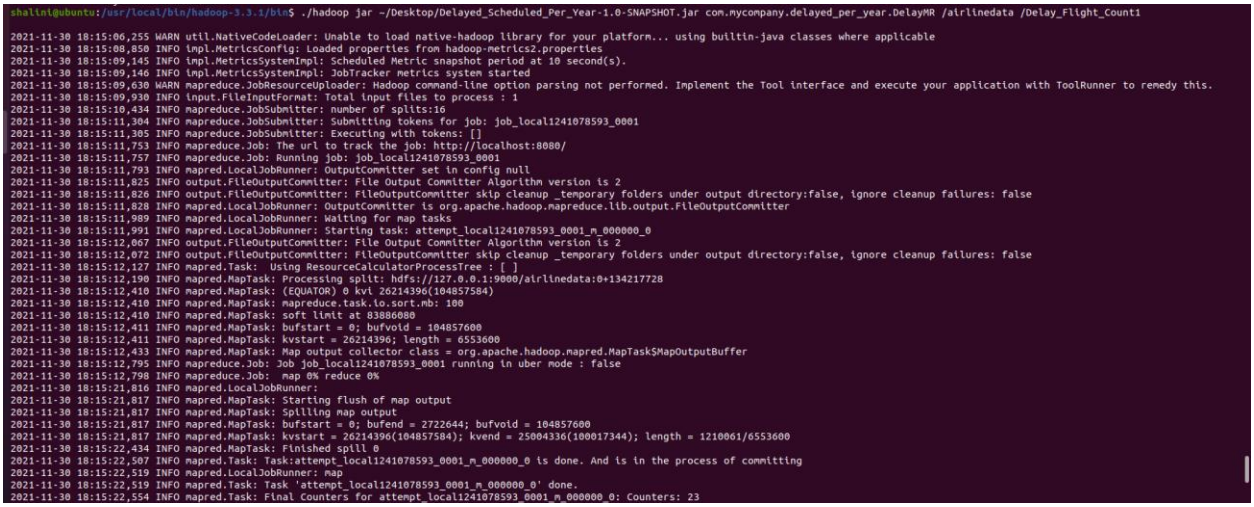
Source | History

```
 * @author shali
 */
public class DelayMR {
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundExcept
        Configuration conf = new Configuration();
        // Create a new Job
        Job job = Job.getInstance(conf,"wordcount");
        job.setJarByClass(DelayMR.class);

        // Specify various job-specific parameters
        job.setJobName("myjob");
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

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

        job.setMapperClass(Delay_Mapper.class);
        job.setReducerClass(Delay_Reducer.class);

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

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

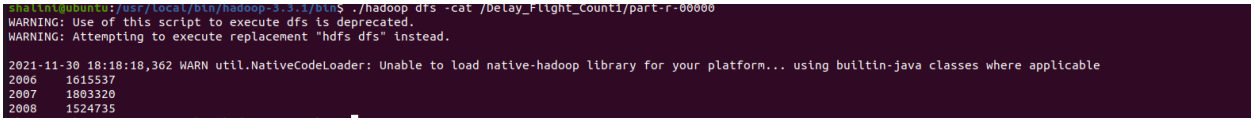
```
/**
 *
 * @author shali
 */
public class Delay_Mapper extends Mapper<LongWritable, Text, Text, IntWritable> {

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

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

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

    }
}
```

```
 *
 * @author shali
 */
public class Delay_Reducer extends Reducer<Text, IntWritable, Text, IntWritable> {

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

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

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

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

}
```

Jar file on HDFS



Output



From above we can see that 2007 has more delayed flights i.e., 15 minutes late as per scheduled time

- **Ratio of Delayed flights per year to total flights (Time taken-2 mins 15 secs)**

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

Delayed flight Count:

Percentage of departure delayed flights: Total Flight Count/ Departure Delayed Flight count = **(19690422/123534970) * 100 = 15.94 %**

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

Code (Main, Tuples, Mapper, Reducer)

```
Source  History
 * @author shali
 */
public class DelayYearMR {

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

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

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

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

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

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

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

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

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

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

    String year = tokens[0];


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


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

    tuple.setFlightCount(1);

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

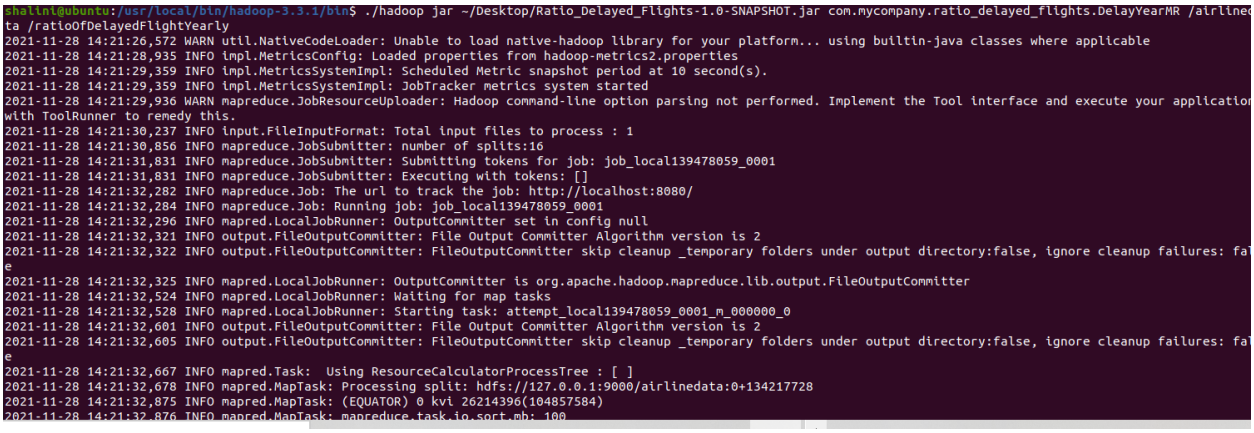
```java
package com.mycompany.ratio_delayed_flights;
import org.apache.hadoop.io.Writable;

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


/**
 *
 * @author shali
 */
public class DelayCountTuple implements Writable {

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

    public int getFlightCount() {
        return totalFlightCount;
    }

    public void setFlightCount(int flightCount) {
        this.totalFlightCount = 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  "totalFlightCount=" + totalFlightCount +
                ", delayedFlightCount=" + delayedFlightCount +
                ", delayedFlightPercentage=" + String.format("%.2f", delayPercent);
    }

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

    }

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

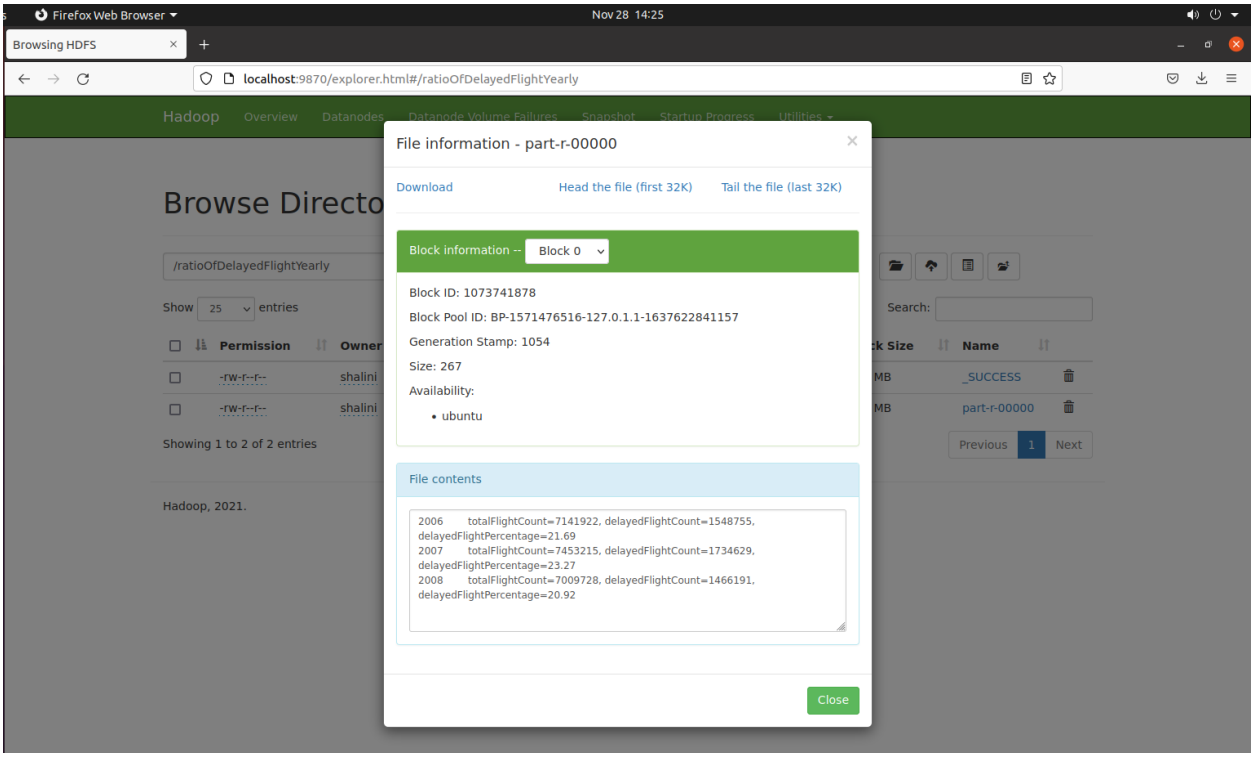
```
        totalFlightCount = dataInput.readInt();
        delayedFlightCount = dataInput.readInt();
        delayPercent = dataInput.readDouble();

    }
}
```

```java
*/
public class DelayYearReducer extends Reducer<Text, DelayCountTuple, Text, DelayCountTuple> {

    private DelayCountTuple res= new DelayCountTuple();

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

        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);
    }
```

Running Jar file on hadoop



Output



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

```
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /ratioOfDelayedFlightYearly/part-r-00000 |head
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-11-28 14:26:38,822 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2006    totalFlightCount=7141922, delayedFlightCount=1548755, delayedFlightPercentage=21.69
2007    totalFlightCount=7453215, delayedFlightCount=1734629, delayedFlightPercentage=23.27
2008    totalFlightCount=7009728, delayedFlightCount=1466191, delayedFlightPercentage=20.92
```

It shows that 2008 is best time to fly as it's has less delayd flights than the year 2007 and 2008

Year with most delay in 2007(23.27%)

- **Ratio of Delayed flights per month to total flights (Time – 1 min 20 secs)**

Code


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

    private String [] days ={"","1-January","2-February","3-March","4-April","5-May","6June","7-July","8-Aug
    private DelayCountTuple tuple = new DelayCountTuple();

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

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

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


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


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


```java
public class DelayCountTuple implements Writable {

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

    public int getFlightCount() {
        return totalFlightCount;
    }

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

    public int getDelayedFlightCount() {
        return delayedFlightCount;
    }

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

    public double getDelayPercent() {
        return delayPercent;
    }
```

```java
public void setDelayPercent(double delayPercent) {
    this.delayPercent = delayPercent;
}
@Override
public String toString() {
    return  "totalFlightCount=" + totalFlightCount +
            ", delayedFlightCount=" + delayedFlightCount +
            ", delayedFlightPercentage=" + String.format("%.2f", delayPercent);
}

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

}

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

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

}
```

Output

```
                Bytes Written=1108
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /DelayMonthCount/part-r-00000
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-12-08 16:55:44,811 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
1-January       totalFlightCount=1808611, delayedFlightCount=394645, delayedFlightPercentage=21.82
10-October      totalFlightCount=1797915, delayedFlightCount=340108, delayedFlightPercentage=18.92
11-November     totalFlightCount=1714618, delayedFlightCount=308659, delayedFlightPercentage=18.00
12-December     totalFlightCount=1763855, delayedFlightCount=504274, delayedFlightPercentage=28.59
2-February      totalFlightCount=1666087, delayedFlightCount=416716, delayedFlightPercentage=25.01
3-March totalFlightCount=1860516, delayedFlightCount=428150, delayedFlightPercentage=23.01
4-April totalFlightCount=1798125, delayedFlightCount=361555, delayedFlightPercentage=20.11
5-May   totalFlightCount=1840821, delayedFlightCount=357823, delayedFlightPercentage=19.44
6June   totalFlightCount=1836260, delayedFlightCount=478740, delayedFlightPercentage=26.07
7-July  totalFlightCount=1897735, delayedFlightCount=454300, delayedFlightPercentage=23.94
8-August        totalFlightCount=1894290, delayedFlightCount=416057, delayedFlightPercentage=21.96
9-September     totalFlightCount=1726032, delayedFlightCount=288548, delayedFlightPercentage=16.72
```

We can infer from this data that best months to fly was September with least delay- 16.72 %
Other good months were-May, October and Nov with delay – 18%
Worst month was December- 28% flights delayed. It may be due to big holidays season in December.

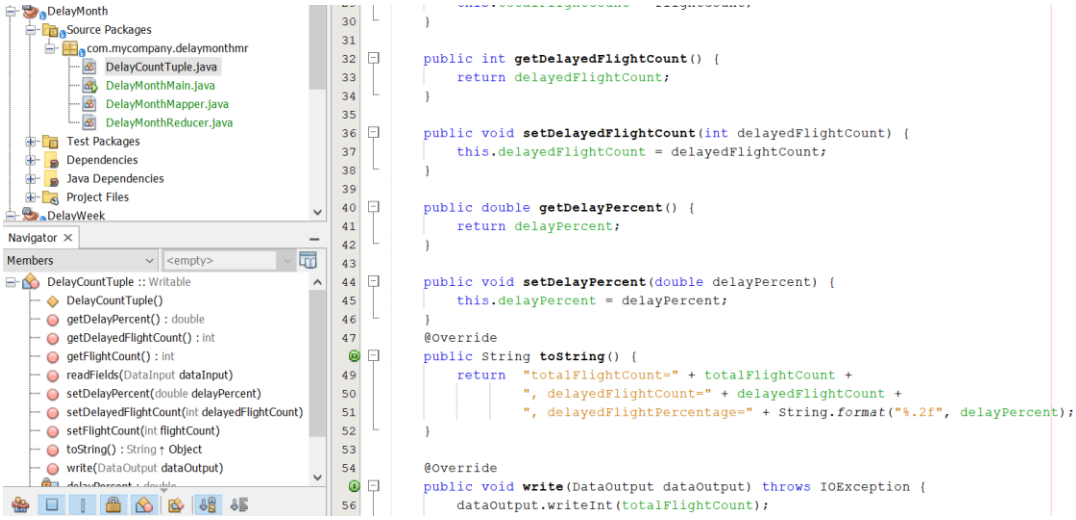
- **Ratio of Delayed flights per day to total flights (Time – 2 mins 30 secs)**

Code

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

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


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


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

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


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

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

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

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

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

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

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

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


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


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

        tuple.setFlightCount(1);
```

```
 */
public class DelayDayReducer extends Reducer<Text, DelayCountTuple, Text, DelayCountTuple> {

    private DelayCountTuple res= new DelayCountTuple();

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

        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);
    }

}
```



Running JAR on HDFS

```
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop jar ~/Desktop/DelayWeek-1.0-SNAPSHOT.jar com.mycompany.delayweek.DelayDayMR /airlinedata /DelayWeekCount
2021-12-08 17:08:14,238 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2021-12-08 17:08:16,181 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2021-12-08 17:08:16,606 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2021-12-08 17:08:16,606 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2021-12-08 17:08:17,126 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy th
2021-12-08 17:08:17,368 INFO input.FileInputFormat: Total input files to process : 1
2021-12-08 17:08:17,639 INFO mapreduce.JobSubmitter: number of splits:16
2021-12-08 17:08:18,331 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local32425814_0001
2021-12-08 17:08:18,331 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-12-08 17:08:18,777 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2021-12-08 17:08:18,779 INFO mapreduce.Job: Running job: job_local32425814_0001
2021-12-08 17:08:18,797 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2021-12-08 17:08:18,820 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2021-12-08 17:08:18,820 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2021-12-08 17:08:18,822 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2021-12-08 17:08:18,992 INFO mapred.LocalJobRunner: Waiting for map tasks
2021-12-08 17:08:18,997 INFO mapred.LocalJobRunner: Starting task: attempt_local32425814_0001_m_000000_0
2021-12-08 17:08:19,098 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2021-12-08 17:08:19,100 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2021-12-08 17:08:19,197 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2021-12-08 17:08:19,207 INFO mapred.MapTask: Processing split: hdfs://127.0.0.1:9000/airlinedata:0+134217728
2021-12-08 17:08:19,427 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2021-12-08 17:08:19,428 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2021-12-08 17:08:19,428 INFO mapred.MapTask: soft limit at 83886080
2021-12-08 17:08:19,428 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2021-12-08 17:08:19,428 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2021-12-08 17:08:19,447 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2021-12-08 17:08:19,794 INFO mapreduce.Job: Job job_local32425814_0001 running in uber mode : false
2021-12-08 17:08:19,796 INFO mapreduce.Job:  map 0% reduce 0%
2021-12-08 17:08:30,157 INFO mapred.LocalJobRunner:
2021-12-08 17:08:30,161 INFO mapred.MapTask: Starting flush of map output
2021-12-08 17:08:30,162 INFO mapred.MapTask: Spilling map output
2021-12-08 17:08:30,162 INFO mapred.MapTask: bufstart = 0; bufend = 34522548; bufvoid = 104857600
2021-12-08 17:08:30,162 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 20485336(81941344); length = 5729061/6553600
2021-12-08 17:08:31,144 INFO mapred.LocalJobRunner: map > sort
2021-12-08 17:08:31,857 INFO mapreduce.Job:  map 4% reduce 0%
2021-12-08 17:08:31,902 INFO mapred.MapTask: Finished spill 0
```

Output

```
                Bytes Written=638
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /DelayWeekCount/part-r-00000
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-12-08 17:12:33,342 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Friday  totalFlightCount=3193461, delayedFlightCount=811409, delayedFlightPercentage=25.41
Monday  totalFlightCount=3196729, delayedFlightCount=708488, delayedFlightPercentage=22.16
Saturday        totalFlightCount=2690405, delayedFlightCount=504583, delayedFlightPercentage=18.75
Sunday  totalFlightCount=3028951, delayedFlightCount=675364, delayedFlightPercentage=22.30
Thursday        totalFlightCount=3182911, delayedFlightCount=759270, delayedFlightPercentage=23.85
Tuesday totalFlightCount=3140933, delayedFlightCount=626378, delayedFlightPercentage=19.94
Wednesday       totalFlightCount=3171475, delayedFlightCount=664083, delayedFlightPercentage=20.94
```

From this data we can infer that maximum delay is on Thursday and Fridays that is when weekends are starting .
Best day to fly are when the weekends ends like Saturday, Tuesday

PIG Analysis

- ## Loading Data into Pig

```
grunt> RAW_DATA = LOAD 'hdfs://127.0.0.1:9000/airlinedata' USING PigStorage(',') AS
>> (year: int, month: int, day: int, dow: int,
>> dtime: int, sdtime: int, arrtime: int, satime: int,
>> carrier: chararray, fn: int, tn: chararray,
>> etime: int, setime: int, airtime: int,
>> adelay: int, ddelay: int,
>> scode: chararray, dcode: chararray, dist: int,
>> tintime: int, touttime: int,
>> cancel: chararray, cancelcode: chararray, diverted: int,
>> cdelay: int, wdelay: int, ndelay: int, sdelay: int, latedelay: int);
```

- ## Monthly Carriers Count (runtime – 3 mins)

```
grunt> CARRIER_DATA = FOREACH RAW_DATA GENERATE month AS m, carrier AS cname;
grunt> GROUP_CARRIERS = GROUP CARRIER_DATA BY (m,cname);
grunt> COUNT_CARRIERS = FOREACH GROUP_CARRIERS GENERATE FLATTEN(group), (COUNT(CARRIER_DATA)) AS popularity;
grunt>
grunt> STORE COUNT_CARRIERS INTO '/PIG-OUTPUT-FULL/Q2/COUNT_CARRIERS' USING PigStorage(',');
2021-12-05 08:52:07,563 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY
2021-12-05 08:52:07,842 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2021-12-05 08:52:07,863 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalculat
timizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimitOptimizer, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushDownForEachFlatten, PushUp
```

Output

```
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /PIG-OUTPUT-FULL/Q1/COUNT_CARRIERS/part-r-00000 |head
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-12-05 09:42:33,717 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
1,9E,43910
1,B6,42859
1,DL,122310
1,NW,100207
1,TZ,1945
1,UA,119672
1,US,125871
1,WN,288071
2,AQ,7289
2,CO,71460
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /PIG-OUTPUT-FULL/Q1/COUNT_CARRIERS/part-r-00001 |head
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-12-05 09:42:41,545 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
1,AA,162141
1,AS,38460
1,F9,22396
1,OH,64057
1,XE,103919
1,YV,73202
2,9E,40915
2,B6,41172
2,DL,112267
2,NW,93526
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /PIG-OUTPUT-FULL/Q1/COUNT_CARRIERS/part-r-00002 |head
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-12-05 09:42:47,273 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
1,AQ,7903
1,CO,75745
1,EV,67976
1,FL,59616
1,HA,13441
1,MQ,134418
1,OO,140492
2,AA,148055
2,AS,35277
2,F9,20590
```

- ## Top Monthly Inbound (runtime – 3 mins)

```
grunt> INBOUND = FOREACH RAW_DATA GENERATE month AS m, dcode AS d;
grunt> -- group by month, then ID (sorted)
grunt> GROUP_INBOUND = GROUP INBOUND BY (m,d);
grunt> -- aggregate over the group, flatten group, such that output relation has 3 fields
grunt> COUNT_INBOUND = FOREACH GROUP_INBOUND GENERATE FLATTEN(group), COUNT(INBOUND) AS count;
grunt> -- aggregate over months only
grunt> GROUP_COUNT_INBOUND = GROUP COUNT_INBOUND BY m;
grunt> -- now apply UDF to compute top k (k=20)
grunt> topMonthlyInbound = FOREACH GROUP_COUNT_INBOUND {
>>     result = TOP(20, 2, COUNT_INBOUND);
>>     GENERATE FLATTEN(result);
>> }
2021-12-03 11:05:44,506 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (G1 Old Gen) of size 104
003200
grunt> --dump topMonthlyInbound
grunt> STORE topMonthlyInbound INTO '/PIG-OUTPUT-FULL/Q1/INBOUND-TOP' USING PigStorage(',');
2021-12-03 11:07:05,668 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.textoutputformat.separator is
2021-12-03 11:07:05,702 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY
2021-12-03 11:07:05,775 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not g
```

Output

```
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /PIG-OUTPUT-FULL/Q1/INBOUND-TOP/part-r-00000
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-12-05 09:49:28,592 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
1,BWI,26614
1,CVG,30194
1,JFK,28766
1,SFO,33215
1,CLT,31191
1,BOS,31700
1,MCO,31437
1,MSP,34667
1,DTW,39385
1,LGA,31778
1,IAH,51491
1,PHX,54084
1,SLC,35974
1,EWR,38062
1,DEN,57510
1,ATL,99421
1,LAX,57752
1,ORD,92603
1,DFW,74430
1,LAS,45322
2,PHL,24387
2,JFK,26679
2,CLT,28188
2,CVG,26698
2,MCO,29756
2,BOS,29925
2,PHX,49330
2,MSP,32094
2,LGA,29332
2,SFO,30497
2,DEN,53739
2,EWR,35099
2,SLC,33067
2,LAX,52659
2,DFW,67765
2,ORD,85125
2,DTW,36520
2,LAS,41552
2,IAH,47857
2,ATL,92561
3,PHL,27373
3,JFK,30021
```

- **Top Monthly Outbound (runtime – 1 min 57 sec)**

```
grunt> OUTBOUND = FOREACH RAW_DATA GENERATE month AS m, scode AS s;
grunt> GROUP_OUTBOUND = GROUP OUTBOUND BY (m,s);
grunt> COUNT_OUTBOUND = FOREACH GROUP_OUTBOUND GENERATE FLATTEN(group), COUNT(OUTBOUND) AS count;
grunt> GROUP_COUNT_OUTBOUND = GROUP COUNT_OUTBOUND BY m;
grunt> topMonthlyOutbound = FOREACH GROUP_COUNT_OUTBOUND {
>>     result = TOP(20, 2, COUNT_OUTBOUND);
>>     GENERATE FLATTEN(result);
>> }
grunt> STORE topMonthlyOutbound INTO '/PIG-OUTPUT-FULL/Q1/OUTBOUND-TOP' USING PigStorage(',');
2021-12-03 16:49:40,595 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY
2021-12-03 16:49:40,816 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2021-12-03 16:49:40,825 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalculato
timizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimitOptimizer, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushDownForEachFlatten, PushUpF
rter]}
```

Output

```
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /PIG-OUTPUT-FULL/Q1/OUTBOUND-TOP/part-r-00000
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-12-05 09:51:11,042 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
1,BWI,26619
1,JFK,28741
1,CLT,31196
1,CVG,30175
1,LGA,31771
1,MCO,31468
1,BOS,31682
1,MSP,34626
1,SFO,33213
1,IAH,51503
1,DTW,39349
1,SLC,36077
1,ATL,99494
1,LAS,45354
1,DEN,57536
1,DFW,74390
1,LAX,57704
1,PHX,54075
1,EWR,38051
1,ORD,92596
2,PHL,24387
2,CVG,26694
2,JFK,26714
2,MCO,29743
2,LGA,29337
2,CLT,28190
2,SFO,30511
2,BOS,29926
2,MSP,32112
2,DTW,36553
2,DFW,67744
2,ATL,92741
2,IAH,47879
2,LAS,41533
2,DEN,53755
2,PHX,49332
2,EWR,35110
2,SLC,33048
2,ORD,85117
2,LAX,52628
3,PHL,27359
3,CVG,29437
```

- **Top Monthly Traffic (runtime – 2 mins 15 sec)**

```
grunt> UNION_TRAFFIC = UNION COUNT_INBOUND, COUNT_OUTBOUND;
grunt> GROUP_UNION_TRAFFIC = GROUP UNION_TRAFFIC BY (m,d);
grunt> TOTAL_TRAFFIC = FOREACH GROUP_UNION_TRAFFIC GENERATE FLATTEN(group) AS (m,code), SUM(UNION_TRAFFIC.count) AS total;
grunt> TOTAL_MONTHLY = GROUP TOTAL_TRAFFIC BY m;
grunt>
grunt> topMonthlyTraffic = FOREACH TOTAL_MONTHLY {
>>     result = TOP(20, 2, TOTAL_TRAFFIC);
>>     GENERATE FLATTEN(result) AS (month, iata, traffic);
>> }
grunt> STORE topMonthlyTraffic INTO '/PIG-OUTPUT-FULL/Q1/MONTHLY-TRAFFIC-TOP/' USING PigStorage(',');
2021-12-03 17:01:06,708 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY,UNION
2021-12-03 17:01:06,776 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2021-12-03 17:01:06,777 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, ConstantCalculator,
timizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimitOptimizer, PartitionFilterOptimizer, PredicatePushdownOptimizer, PushDownForEachFlatten, PushUpFil
```

Output

```
shalini@ubuntu:/usr/local/bin/hadoop-3.3.1/bin$ ./hadoop dfs -cat /PIG-OUTPUT-FULL/Q1/MONTHLY-TRAFFIC-TOP/part-r-00000
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2021-12-05 09:52:32,723 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java clas
1,BWI,53233
1,JFK,57507
1,CVG,60369
1,MCO,62905
1,SFO,66428
1,CLT,62387
1,BOS,63382
1,MSP,69293
1,LGA,63549
1,SLC,72051
1,DTW,78734
1,LAS,90676
1,ORD,185199
1,DFW,148820
1,DEN,115046
1,PHX,108159
1,IAH,102994
1,ATL,198915
1,EWR,76113
1,LAX,115456
2,PHL,48774
2,CVG,53392
2,JFK,53393
2,BOS,59851
2,LGA,58669
2,MCO,59499
2,CLT,56378
2,DTW,73073
2,SLC,66115
2,SFO,61008
2,EWR,70209
2,DEN,107494
2,IAH,95736
2,MSP,64206
2,LAS,83085
2,ATL,185302
2,PHX,98662
2,LAX,105287
```

HIVE ANALYSIS

CREATE DATABASE AIRLINE

```
hive> create schema airline;
OK
Time taken: 1.511 seconds
hive> use airline;
OK
Time taken: 0.084 seconds
```

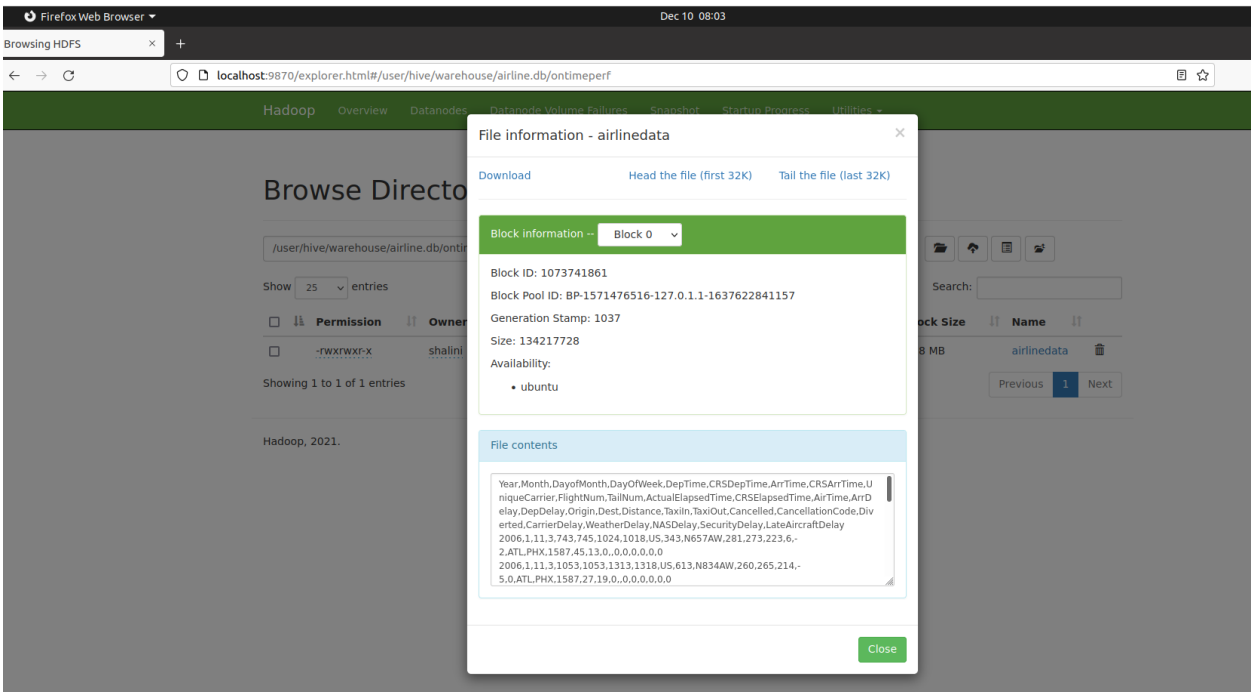
CREATED TABLE WITH COLUMN NAMES AND DATA TYPES

```
hive> create table airline.onTimePerf (year int, month int, DayofMonth INT,DayOfWeek INT,DepTime INT,CRSDepTime INT,ArrTime INT,CRSArrTime INT,UniqueCarrier String,FlightNum INT,TailNum String,ActualElapsedTime INT,CRSElapsedTime INT,AirTime INT,ArrDelay INT,DepDelay INT,Origin String,Dest String,Distance INT,TaxiIn INT,TaxiOut INT,Cancelled INT,CancellationCode String,Diverted String,CarrierDelay INT,WeatherDelay INT,NASDelay INT,SecurityDelay INT, LateAircraftDelay INT)ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
OK
Time taken: 1.85 seconds
hive>
```

LOADING DATA INTO TABLE

```
hive> load data inpath 'hdfs://127.0.0.1:9000/airlinedata' into table  airline.onTimePerf;
Loading data to table airline.ontimeperf
OK
Time taken: 3.32 seconds
hive>
```

FILE CHECK



- **Flight late departure and arrival on time (Time taken – 100 secs)**

```
hive> INSERT OVERWRITE DIRECTORY 'HiveOutput/StartedlateReachedOntime'
    > select
    > Year,Month,DayofMonth,Origin,Dest,AirTime,Distance,TaxiIn,TaxiOut
    > from ontimeperf where DepTime>CRSDepTime and
    > ArrTime<=CRSArrTime;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Cor
Query ID = shalini_20211210085301_3dc4466a-1cbb-4530-92bd-8f792c602fb7
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2021-12-10 08:53:08,213 Stage-1 map = 0%,  reduce = 0%
2021-12-10 08:53:20,521 Stage-1 map = 6%,  reduce = 0%
```

Output

```
2006 1 11 BDL PHL 52 196 10 10
2006 1 11 BOS DCA 79 399 3 11
2006 1 11 BOS LAS 314 2381 4 27
2006 1 11 BOS LAS 312 2381 8 19
2006 1 11 BOS PHL 62 280 5 14
2006 1 11 BOS PIT 81 496 3 14
2006 1 11 BUF CLT 82 546 11 9
2006 1 11 BWI CLT 59 361 5 9
```

- **Flight departure and arrival on time (Time taken – 100 secs)**

```
INSERT OVERWRITE DIRECTORY 'HiveOutput/Dep_Arrived_Ontime'
select
Year,Month,DayofMonth,Origin,Dest,AirTime,Distance,TaxiIn,TaxiOut
from ontimeperf where DepTime<=CRSDepTime and
ArrTime<=CRSArrTime;
```

2007 3 24 SFO JFK 301 2586 4 16
2007 3 3 JFK SFO 323 2586 6 34
2007 3 4 JFK SFO 348 2586 5 16
2007 3 8 JFK SFO 342 2586 6 22
2007 3 9 JFK SFO 345 2586 4 33
2007 3 10 JFK SFO 343 2586 6 33
2007 3 11 JFK SFO 339 2586 5 17
2007 3 12 JFK SFO 331 2586 9 35

- **Flight Travelled greater than 1000 miles (Time – 15.86 secs)**

```
hive> INSERT OVERWRITE DIRECTORY 'HiveOutput/FlightTravelledgreaterthan1000'
    > select count(*) from onTimePerf where Distance > 1000;
```

File contents

```
4815308
```

- **Flight Travelled less than 1000 miles (Time - 17 secs)**

```
hive> INSERT OVERWRITE DIRECTORY 'HiveOutput/FlightTravelledlessthan1000'
    > select count(*) from onTimePerf where Distance < 1000;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future ver
```

File contents

```
16789557
```

- **Count of Flight Travelled having arrival and departure delay greater than 30 mins(Time - 17 secs)**

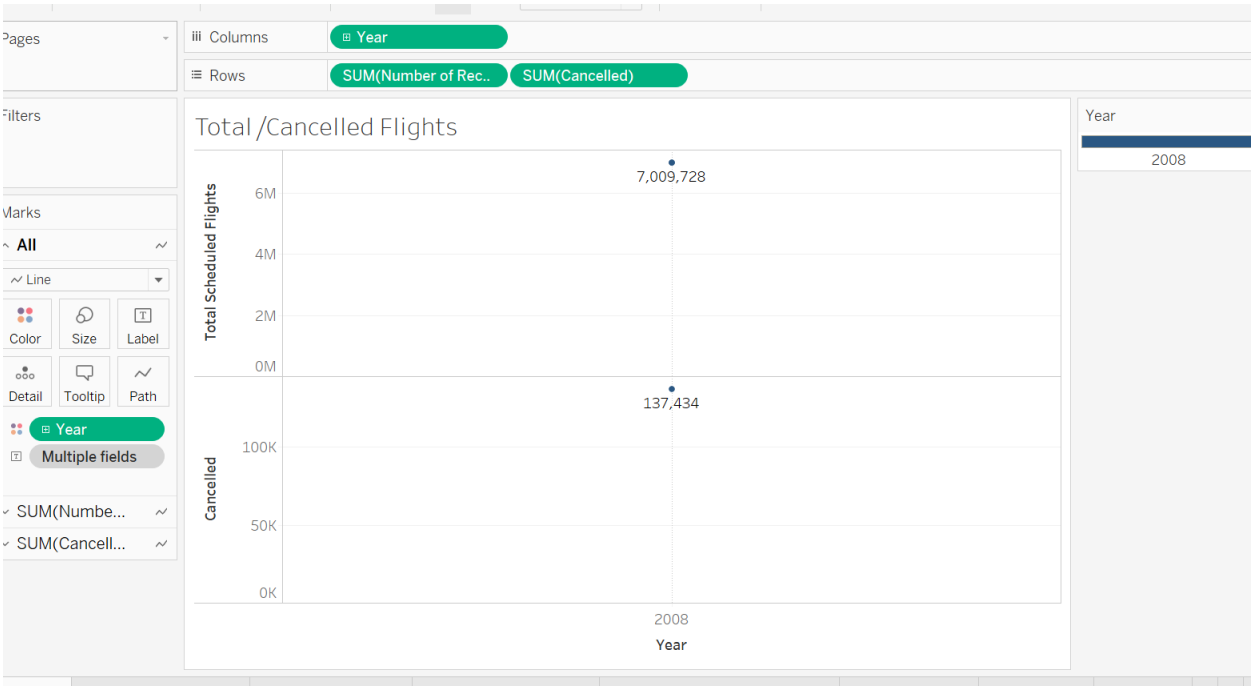
```
hive> INSERT OVERWRITE DIRECTORY 'HiveOutput/ArrDepartureDelaymorethan30mins'
    > select count(*) from onTimePerf where (ArrDelay + DepDelay) > 30 ;
```

File contents

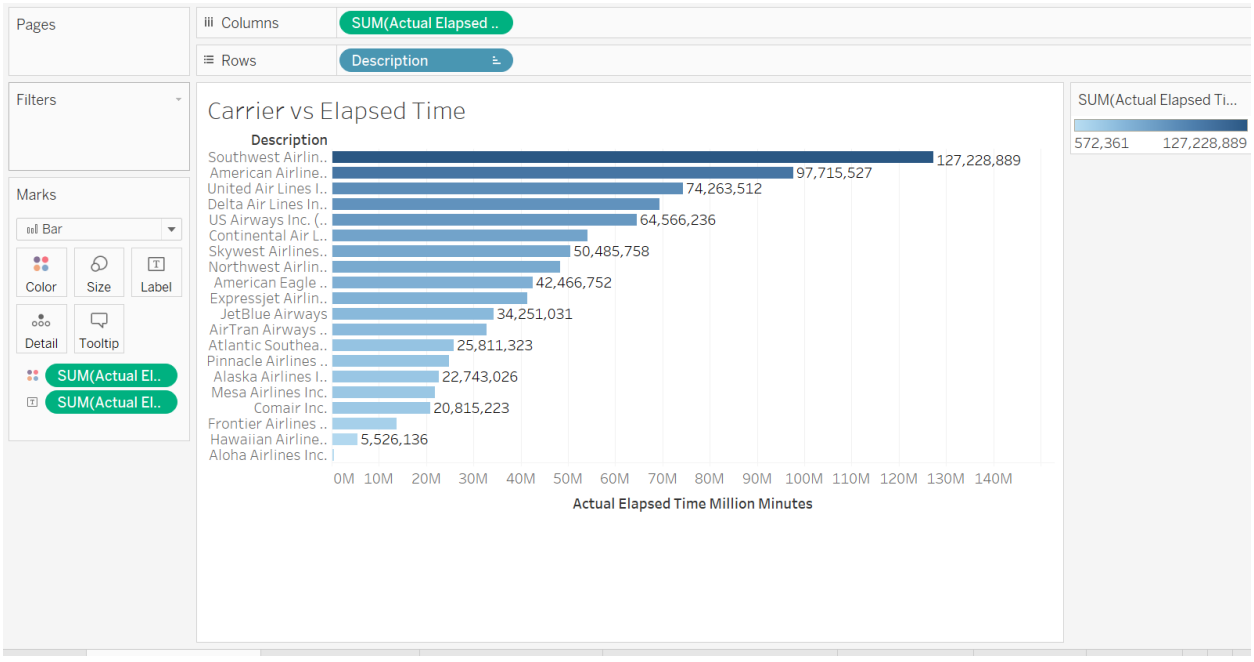
```
4320780
```

TABLEAU ANALYSIS

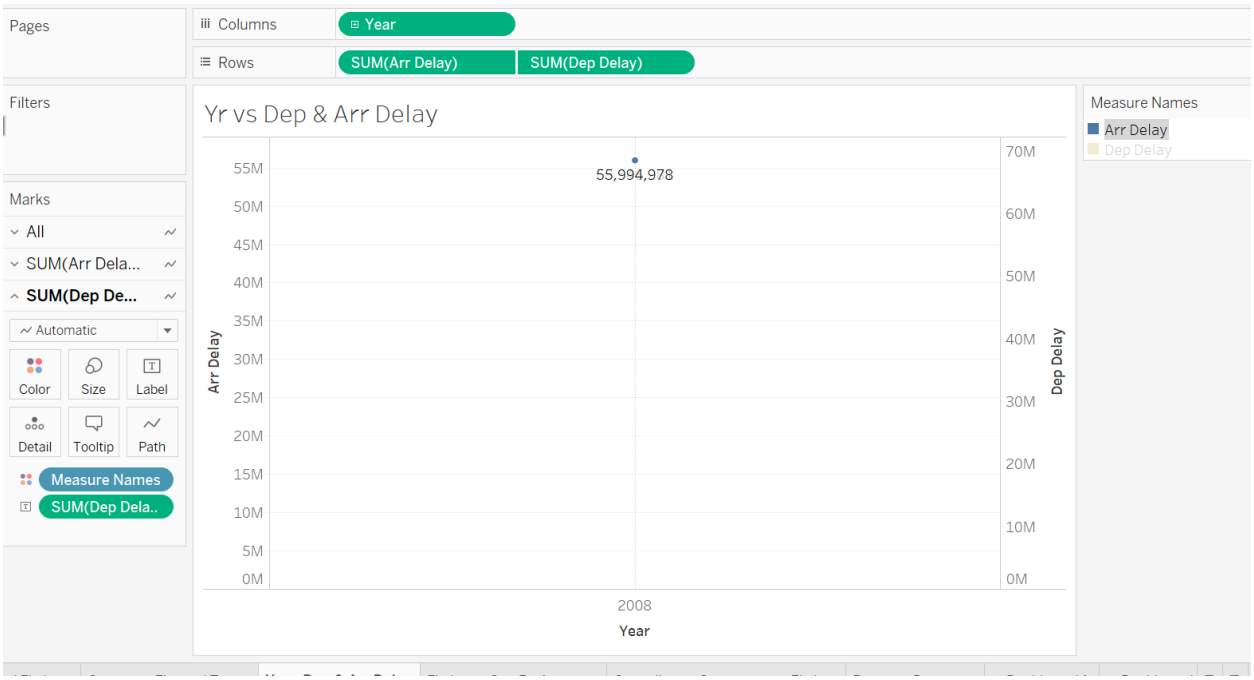
- **Total Flights vs Cancelled Flights**



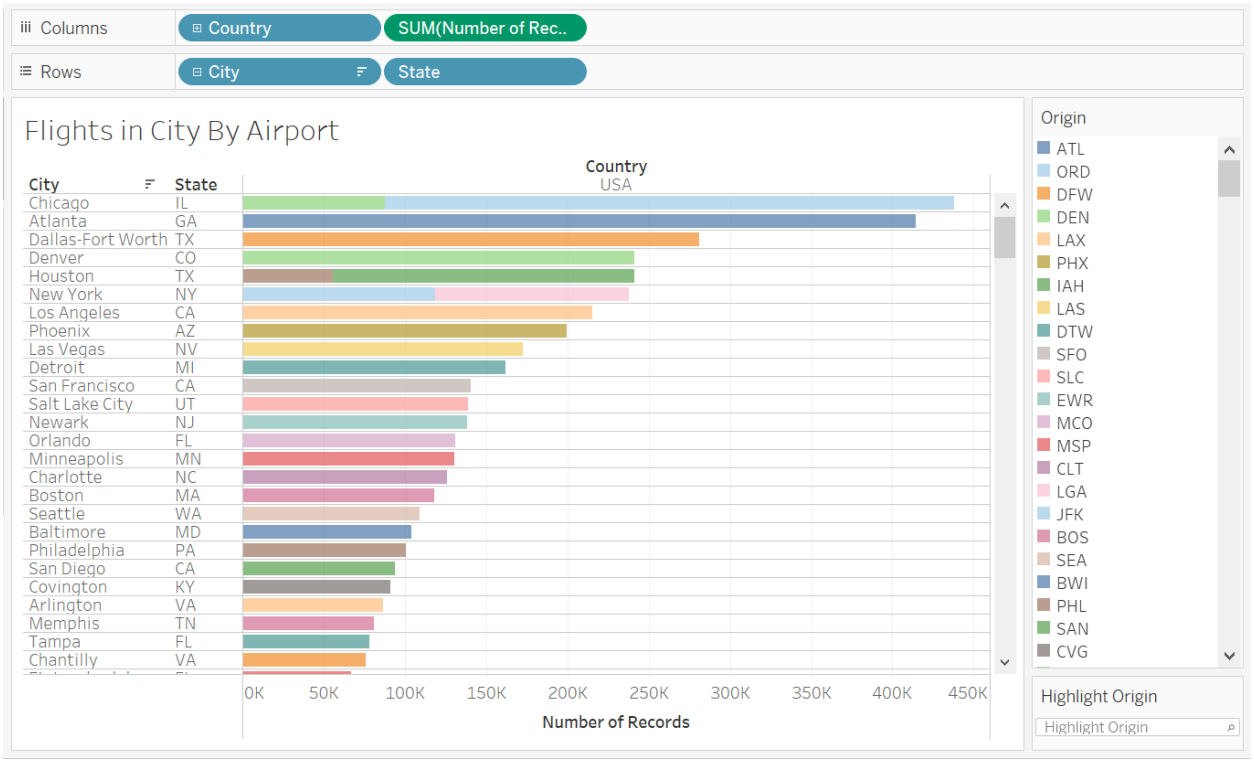
- **CARRIER VS ELAPSED TIMES**



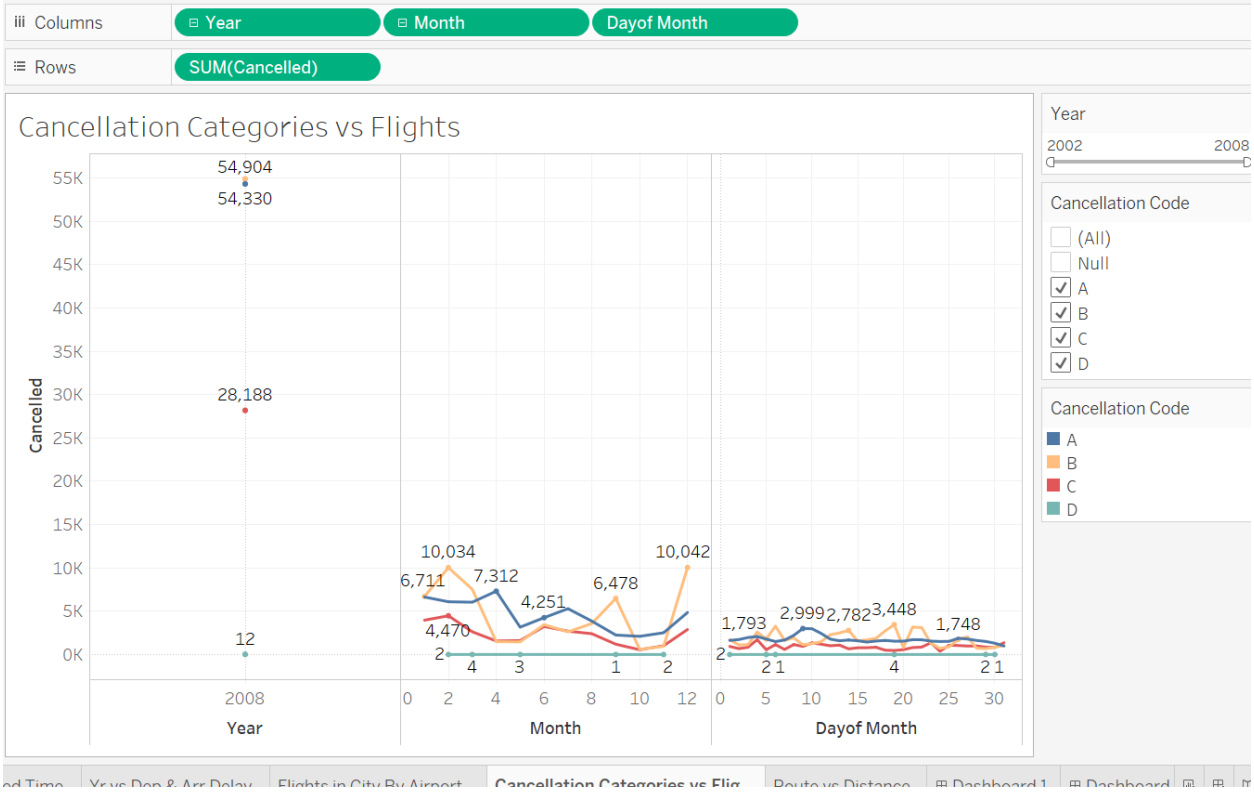
- **Arr vs Dep Delay**



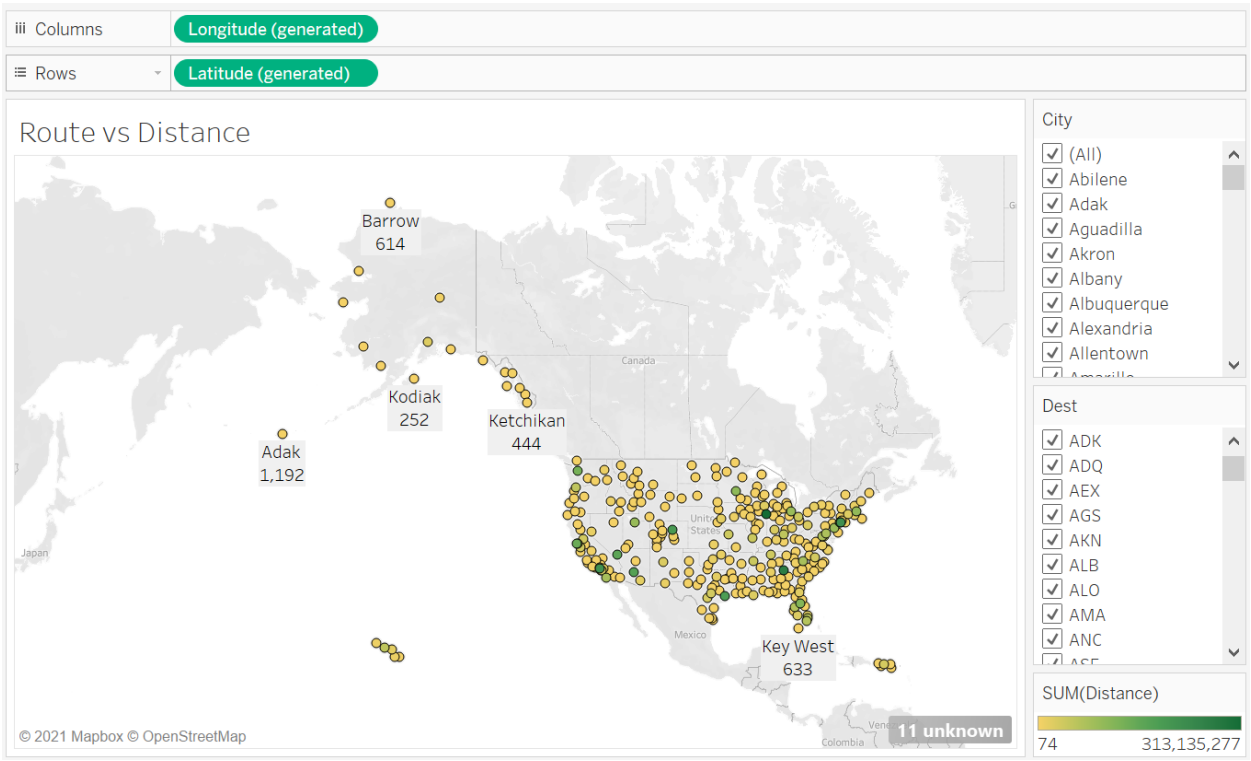
- **Number of Flight by City and State Airport**



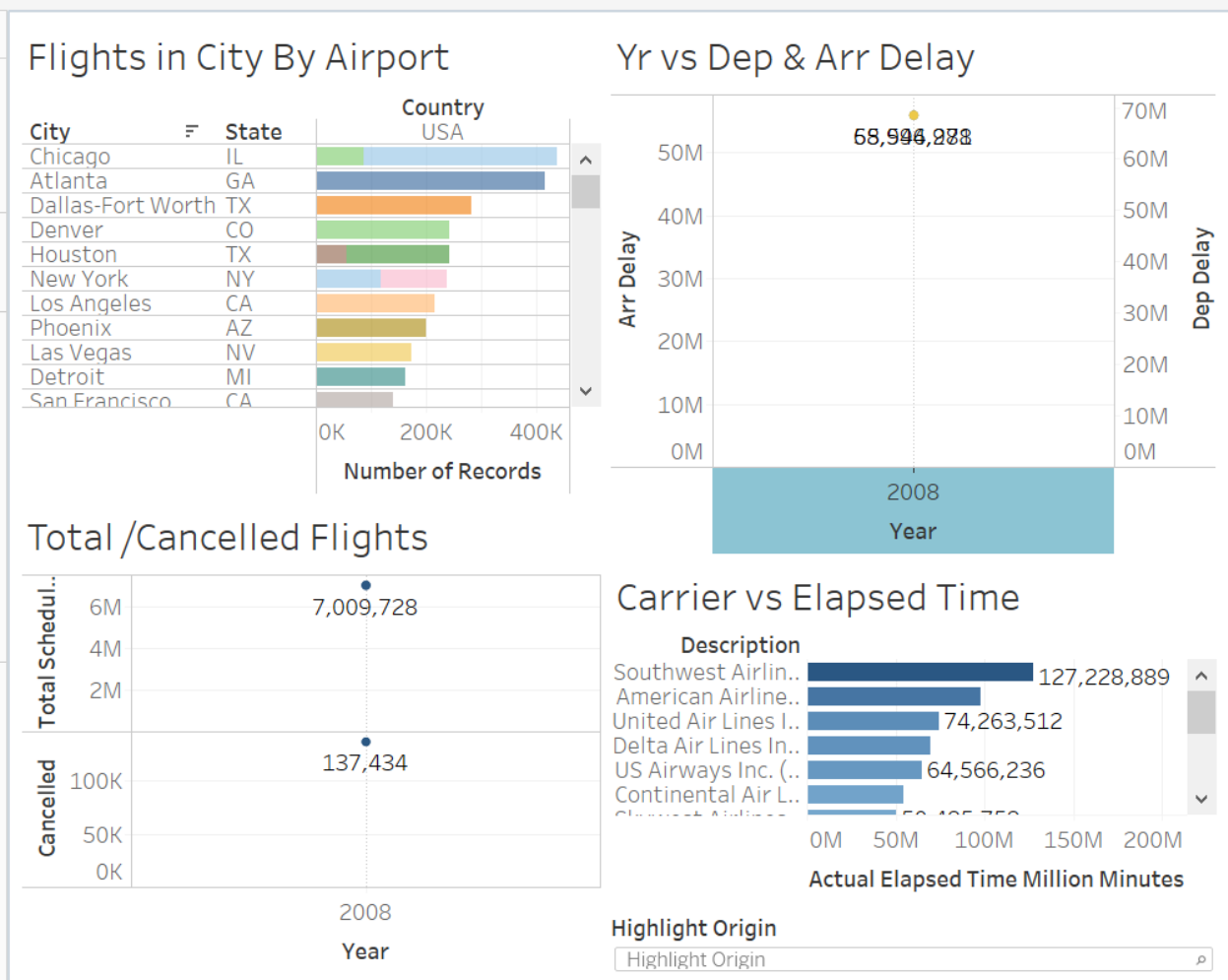
- **Cancellation categories vs flights**



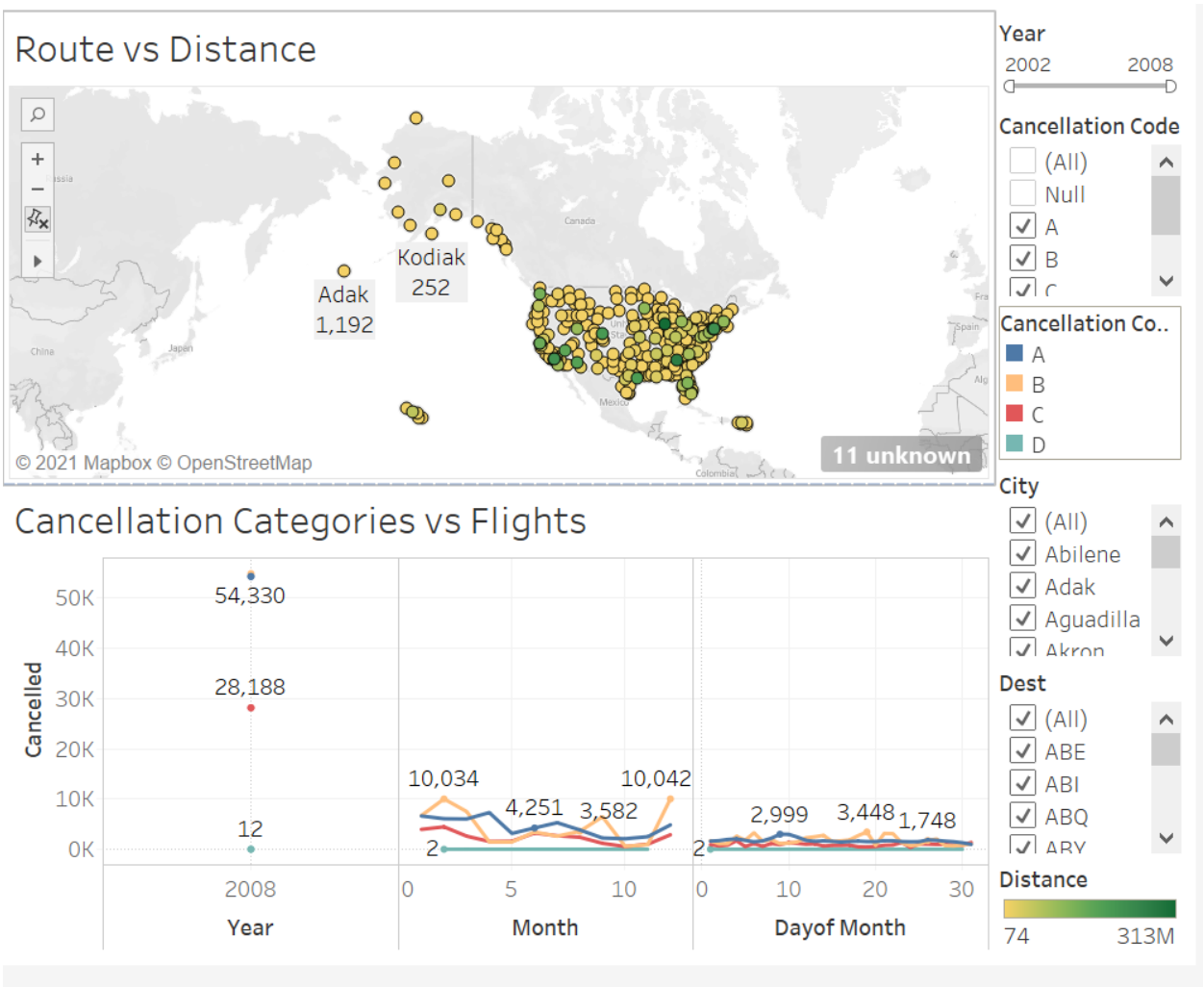
- **Route vs Distance**



- **Dashboard 1**



- **DASHBOARD 2**


Route vs Distance
Adak
1,192
Kodiak
252
11 unknown
© 2021 Mapbox © OpenStreetMap
Cancellation Categories vs Flights
Cancelled
50K
40K
30K
20K
10K
0K
54,330
28,188
12
10,034
4,251
3,582
10,042
2
2,999
3,448
1,748
2
2008
Year
0
5
10
Month
0
10
20
30
Dayof Month
Year
2002
2008
Cancellation Code
(All)
Null
A
B
C
Cancellation Co..
A
B
C
D
City
(All)
Abilene
Adak
Aguadilla
Akron
Dest
(All)
ABE
ABI
ABQ
ARV
Distance
74
313M

REFERENCES

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

APPENDIX

1. Rows Count

```
package com.mycompany.bigdata_final_project;
import java.io.IOException;
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;

/**
 *
 * @author shali
 */
public class Airport_Count_Main {
   public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException{


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

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

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

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

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

      job.setMapperClass(Airport_Count_Mapper.class);
      job.setCombinerClass(Airport_Count_Reducer.class);
      job.setReducerClass(Airport_Count_Reducer.class);
```

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

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

    }


}package com.mycompany.bigdata_final_project;
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;

/**
 *
 * @author shali
 */
public class Airport_Count_Mapper extends Mapper<LongWritable, Text, Text, IntWritable> {

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

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

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

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



}package com.mycompany.bigdata_final_project;
```

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

import java.io.IOException;

/**
 *
 * @author shali
 */
public class Airport_Count_Reducer extends Reducer<Text, IntWritable, Text, IntWritable> {

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

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

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

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

}
```

2 . Unique Carrier Count

```
package com.mycompany.unique_carrier_count;

import java.io.IOException;

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;
/**
 *
 * @author shali
 */
public class CarrierMain {
    public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException{


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

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



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



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

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




        job.setMapperClass(CarrierMapper.class);
        job.setReducerClass(CarrierReducer.class);

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

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

    }
}
package com.mycompany.unique_carrier_count;

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

import java.io.IOException;
/**
 *
 * @author shali
```

```
 */
public class CarrierMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

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

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

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

      String carrier = tokens[8];
      word.set(carrier);
      context.write(word,one);
   }



}
package com.mycompany.unique_carrier_count;

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

```
import java.io.IOException;
/**
 *
 * @author shali
 */
public class CarrierMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

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

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

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

        String carrier = tokens[8];
        word.set(carrier);
        context.write(word,one);
    }

}
```

3. Rows Count

```
package com.mycompany.rows_count;
import org.apache.hadoop.io.IntWritable;
```

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

import java.io.IOException;
/**
 *
 * @author shali
 */
public class RowsCountMapper extends Mapper<LongWritable, Text, NullWritable, IntWritable> {

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

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



      context.write(NullWritable.get(),one);
   }

}package com.mycompany.rows_count;

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

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


import java.io.IOException;


/**
 *
 * @author shali
 */
public class RowsCountReducer extends Reducer<NullWritable, IntWritable, NullWritable, IntWritable> {


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


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


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


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



}package com.mycompany.rows_count;
```

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

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
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;

/**
 *
 * @author shali
 */
public class RowsCount_MR{
   public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException{



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

      // Specify various job-specific parameters
```

```
        job.setJobName("myjob");



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



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

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



        job.setMapperClass(RowsCountMapper.class);
        job.setReducerClass(RowsCountReducer.class);

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

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

    }


}
```

4. Source Destination Pairs

```
package com.mycompany.source_desti_pairs_count;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

/**
 *
 * @author shali
 */
public class SrcDestMR {
   public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException{



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

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


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


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

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




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

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

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

    }
}
```

```
package com.mycompany.source_desti_pairs_count;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;


import java.io.IOException;









/**

 *

 * @author shali

 */

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


   //    hadoop datatype

   Text word = new Text();

   IntWritable one = new IntWritable(1);


   @Override

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


       String line = value.toString();

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

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



}package com.mycompany.source_desti_pairs_count;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
/**
 *
 * @author shali
 */
public class SrcDestReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

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

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

```
        }

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

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

}
```

5. CANCELLED FLIGHT YEARLY

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


/**
 *
```

```
 * @author shali
 */
public class CancelYr_MR {
    public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException{


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

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


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


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

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

```
        job.setMapperClass(CancelYr_Mapper.class);
        job.setReducerClass(CancelYr_Reducer.class);

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

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

    }


} package com.mycompany.cancelled_by_year;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

/**
 *
```

```
 * @author shali
 */
public class CancelYr_Mapper extends Mapper<LongWritable, Text, Text,
IntWritable> {


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


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


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


   }
```

```
} package com.mycompany.cancelled_by_year;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;


import java.io.IOException;


/**

 *

 * @author shali

 */

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


   // just like in mongoDB values is iterable

   @Override

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


      int sum=0;

      for(IntWritable v: values){

         sum += v.get();

      }
```

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


    }

}
```

6. DELAYED FLIGHT YEARLY

```
package com.mycompany.delayed_per_year;
import java.io.IOException;
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;

/**
 *
 * @author shali
 */
```

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


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

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


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


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

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

```
        job.setMapperClass(Delay_Mapper.class);
        job.setReducerClass(Delay_Reducer.class);

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

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

    }
} package com.mycompany.delayed_per_year;

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

import java.io.IOException;

/**
 *
 * @author shali
 */
public class Delay_Mapper extends Mapper<LongWritable, Text, Text,
IntWritable> {
```

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


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


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


    }
} package com.mycompany.delayed_per_year;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
```

```
import java.io.IOException;


/**
 *
 * @author shali
 */
public class Delay_Reducer extends Reducer<Text, IntWritable, Text, IntWritable>
{


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


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


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


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

```
    }

}
```

7. RATIO OF DELAYED FLIGHT yearly

```
package com.mycompany.ratio_delayed_flights;
import org.apache.hadoop.io.Writable;

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



/**
 *
 * @author shali
 */
public class DelayCountTuple implements Writable {

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

    public int getFlightCount() {
        return totalFlightCount;
```

```
}

public void setFlightCount(int flightCount) {
   this.totalFlightCount = 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  "totalFlightCount=" + totalFlightCount +
```

```
            ", delayedFlightCount=" + delayedFlightCount +
            ", delayedFlightPercentage=" + String.format("%.2f", delayPercent);
    }

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

    }

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

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

    }
} package com.mycompany.ratio_delayed_flights;
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;

/**
 *
 * @author shali
 */
public class DelayYearMR {

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


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

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

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


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

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


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

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

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

  }
```

```
} package com.mycompany.ratio_delayed_flights;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;


import java.io.IOException;


/**

 *

 * @author shali

 */

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


   private DelayCountTuple tuple = new DelayCountTuple();


   @Override

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

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


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


      String year = tokens[0];
```

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


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

        tuple.setFlightCount(1);

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

} package com.mycompany.ratio_delayed_flights;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
/**
```

```
 *
 * @author shali
 */
public class DelayYearReducer extends Reducer<Text, DelayCountTuple, Text, DelayCountTuple> {

    private DelayCountTuple res= new DelayCountTuple();

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

        int total=0;
        int delayedTotal=0;

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

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

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

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

}
```

8. RATIO OF DELAYED FLIGHT Monthly

```
package com.mycompany.delaymonthmr;
import java.io.IOException;
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;
/**
 *
 * @author shali
 */
public class DelayMonthMain {
    public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException{
```

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

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


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


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

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


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

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

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

    }

} package com.mycompany.delaymonthmr;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

/**
 *
 * @author shali
 */
public class DelayMonthMapper extends Mapper<Object, Text, Text,
DelayCountTuple> {

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

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

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

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

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

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

        tuple.setFlightCount(1);
```

```
        context.write(new Text(month),tuple);
    }
} package com.mycompany.delaymonthmr;


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

import java.io.IOException;

/**
 *
 * @author shali
 */
public class DelayMonthReducer extends Reducer<Text, DelayCountTuple, Text,
DelayCountTuple> {

    private DelayCountTuple res= new DelayCountTuple();

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

        int total=0;
```

```
        int delayedTotal=0;

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

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

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

        context.write(key,res);
    }

} package com.mycompany.delaymonthmr;
import org.apache.hadoop.io.Writable;


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

```
/**
 *
 * @author shali
 */
public class DelayCountTuple implements Writable {

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

    public int getFlightCount() {
        return totalFlightCount;
    }

    public void setFlightCount(int flightCount) {
        this.totalFlightCount = 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  "totalFlightCount=" + totalFlightCount +
            ", delayedFlightCount=" + delayedFlightCount +
            ", delayedFlightPercentage=" + String.format("%.2f", delayPercent);
}

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

}
```

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

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

    }
}
```

9. RATIO OF DELAYED FLIGHT Daily

```
package com.mycompany.delayweek;
import java.io.IOException;
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;
```

```
/**
 *
 * @author shali
 */
public class DelayDayMR {public static void main(String[] args) throws
IOException, InterruptedException, ClassNotFoundException{


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

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


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


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

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



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

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

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

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

import java.io.IOException;

/**
 *
 * @author shali
 */
```

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


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


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


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


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



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



         if (delay > 15) {
            tuple.setDelayedFlightCount(1);
         } else {
            tuple.setDelayedFlightCount(0);
```

```
        }
      }catch (Exception e){
        tuple.setDelayedFlightCount(0);
      }

      tuple.setFlightCount(1);

      context.write(new Text(day),tuple);
   }
} package com.mycompany.delayweek;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

/**
 *
 * @author shali
 */
public class DelayDayReducer extends Reducer<Text, DelayCountTuple, Text,
DelayCountTuple> {

   private DelayCountTuple res= new DelayCountTuple();

   @Override
```

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

      int total=0;
      int delayedTotal=0;

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

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

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

      context.write(key,res);
   }

} package com.mycompany.delayweek;
import org.apache.hadoop.io.Writable;


import java.io.DataInput;
```

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


/**
 *
 * @author shali
 */
public class DelayCountTuple implements Writable {

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

    public int getFlightCount() {
        return totalFlightCount;
    }

    public void setFlightCount(int flightCount) {
        this.totalFlightCount = 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  "totalFlightCount=" + totalFlightCount +
         ", delayedFlightCount=" + delayedFlightCount +
         ", delayedFlightPercentage=" + String.format("%.2f", delayPercent);
}

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

```
        dataOutput.writeDouble(delayPercent);

    }

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

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

    }
}
```