

Flight Delay Analysis



Poojith Shankar Shetty
001821661
4/26/2019

AGENDA

- Introduction
- Dataset Overview
- List of Analysis
- Representation using Tableau
- Analysis Outputs
- Technologies and Patterns used
- Appendix
- Additional Screenshots

INTRODUCTION

- Flight Cancellation Analysis
 - Based on Airlines, month, State etc.
 - Reason for the Delay.
- Flight Delay Analysis
 - Time of Delay
 - Based on Airlines, month and State
 - Reason for the Delay
- Dataset Link
 - https://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time
- Flight Distance Travelled
 - Details about how much distance the different flight travelled

DATA SET OVERVIEW

Columns of Dataset:

| YEAR | ORIGIN_CITY_NAME | DEP_DELAY_NEW | CRS_ELAPSED_TIME |
|-------------------|------------------|-------------------|---------------------|
| QUARTER | ORIGIN_STATE_ABR | TAXI_OUT | ACTUAL_ELAPSED_TIME |
| MONTH | ORIGIN_STATE_NM | WHEELS_ON | AIR_TIME |
| DAY_OF_MONTH | DEST_AIRPORT_ID | TAXI_IN | FLIGHTS |
| DAY_OF_WEEK | DEST | CRS_ARR_TIME | DISTANCE |
| FL_DATE | DEST_CITY_NAME | ARR_TIME | DISTANCE_GROUP |
| OP_UNIQUE_CARRIER | DEST_STATE_ABR | ARR_DELAY | CARRIER_DELAY |
| TAIL_NUM | DEST_STATE_NM | ARR_DELAY_NEW | WEATHER_DELAY |
| OP_CARRIER_FL_NUM | CRS_DEP_TIME | CANCELLED | NAS_DELAY |
| ORIGIN_AIRPORT_ID | DEP_TIME | CANCELLATION_CODE | SECURITY_DELAY |
| ORIGIN | DEP_DELAY | DIVERTED | LATE_AIRCRAFT_DELAY |

DATA SET OVERVIEW

- Cancellation Code: These are the reasons for Flight cancellation
 - A – Carrier
 - B – Weather
 - C – National Air System
 - D – Security
- All the time in the data set are in Minutes.(Delay related columns).

LIST OF ANALYSIS

1. Used PutMerge to Merge the Data.
2. Used AWS Cloud EMR instance for running MapReduce Jobs.
3. Used counting with Counters to determine Number of Flights in different Distance Groups
4. Used Memory-Conscious Median and Standard Deviation of Distance travelled based on Airliner used separate Combiner to simplify the input to Reducer.
5. Calculated the each Cancellation of Flight based on Airliner each month.
6. Calculated the each Delay of Flight based on Airliner each month.
7. Did Secondary Sorting of the States and the available Airports in that state to know the list of Airports Present in US.(Partitioner is used in this case)
8. Calculated Average Delay by the airliner each month used Custom Writable class for this technique.

LIST OF ANALYSIS

9. Calculated Total Average Flight Delay by different Airliner.
10. Used Shuffling Technique to shuffle the Data.
11. Calculated Total Cancelled Flights based on Different Reason.
12. Calculated Total Delay of Flight each month
13. Determined Top 10 flight data based on longest Distance.
14. Determined Delay Median based on Airliner
15. Determined Distinct Airline in the Dataset.
16. Separated the data based on Cancellation Reason by using Binning Technique.

LIST OF ANALYSIS

17. Used Bloom Filter to Filter the data based on the Airport input.
18. Used Filtered output using Chaining and determined the Flights from the Airport using Inverted Index Method.
19. Based on the Carrier Names from a different Carrier CSV found the Full name of the Flights present in the original Data.
20. Determined the Minimum and Maximum Distance travelled by flight based on each Airliner.
21. Calculated Flight cancelled statewise in each month to determine which state has most flight cancelled.
22. Determined Flight delayed state wise in each month.
23. Found Top 10 Airline with Most Flights using Pig.
24. Filtered the Latest Data based on Year using Pig.
25. Joined Airline Data based on Airline Name using Pig.

Analysis using Tableau

Pages

iii Columns

Airline_Name 

Rows

SUM(Count)

Filters

 Month Airline_Name 

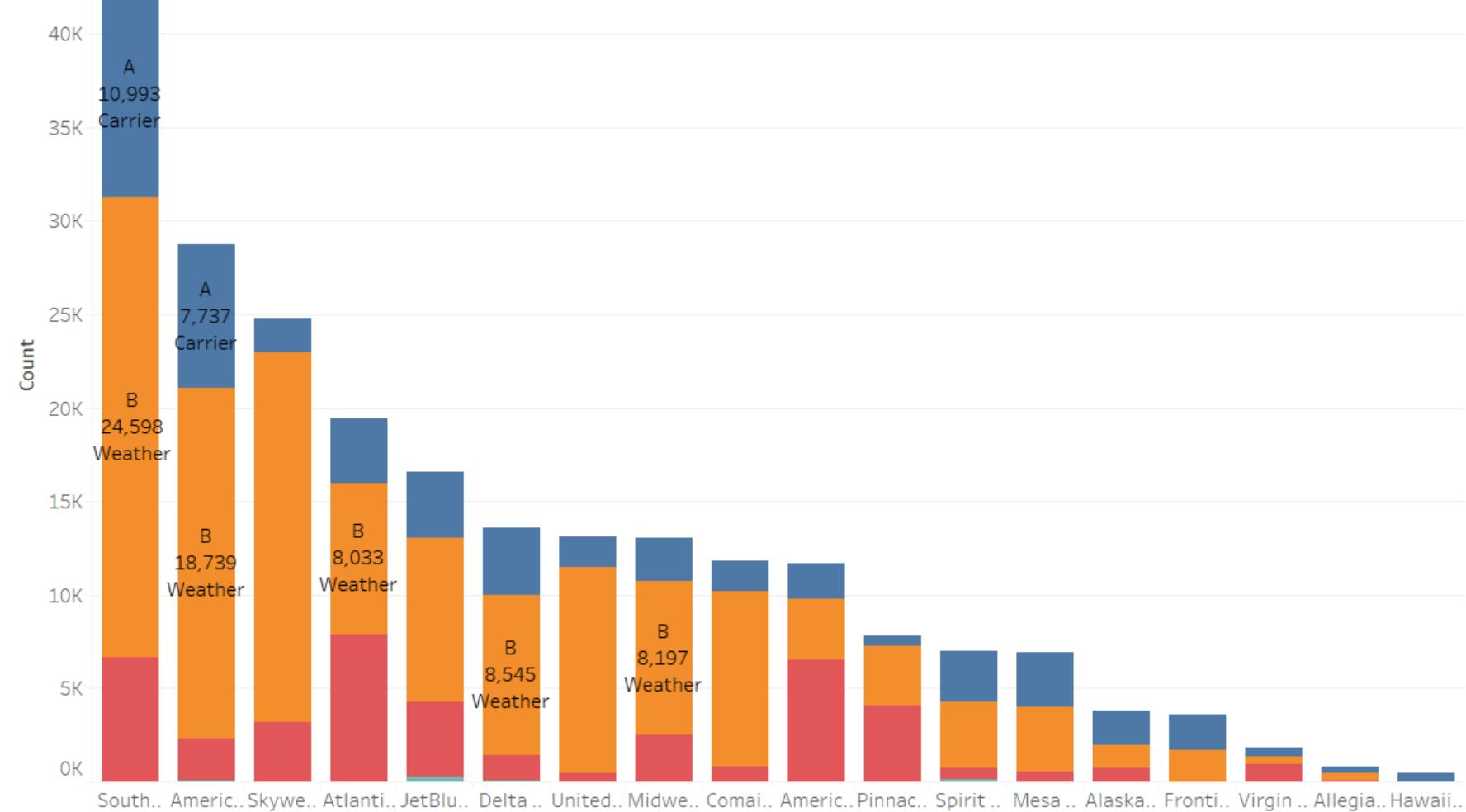
Marks

Automatic

 Color  Size  Label Detail  Tooltip Canc_Reason  Canc_Reason  SUM(Count)  Description

Cancellation Count By Airline

Airline_Name



Airline_Name

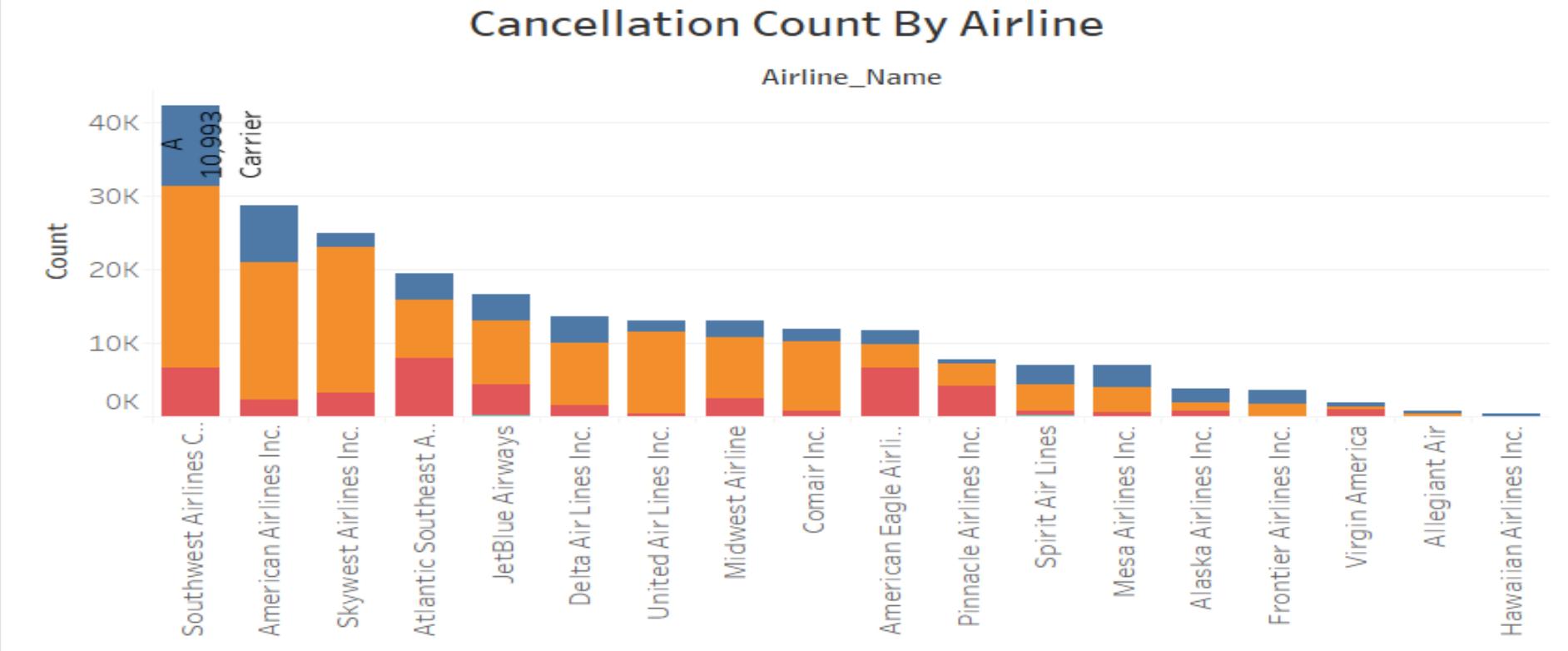
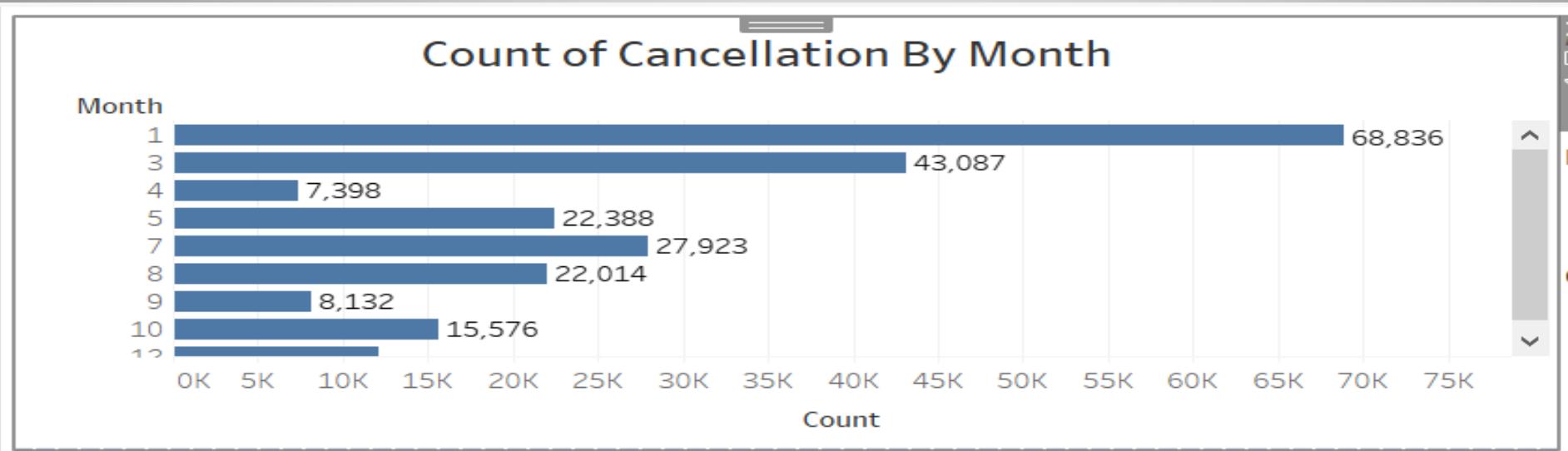
- (All)
- Alaska Airlines Inc.
- Allegiant Air
- American Airlines...
- American Eagle A...
- Atlantic Southea...
- Comair Inc.
- Delta Air Lines Inc.
- Frontier Airlines I...
- Hawaiian Airlines...
- JetBlue Airways
- Mesa Airlines Inc.
- Midwest Airline
- Pinnacle Airlines I...
- Southwest Airlines

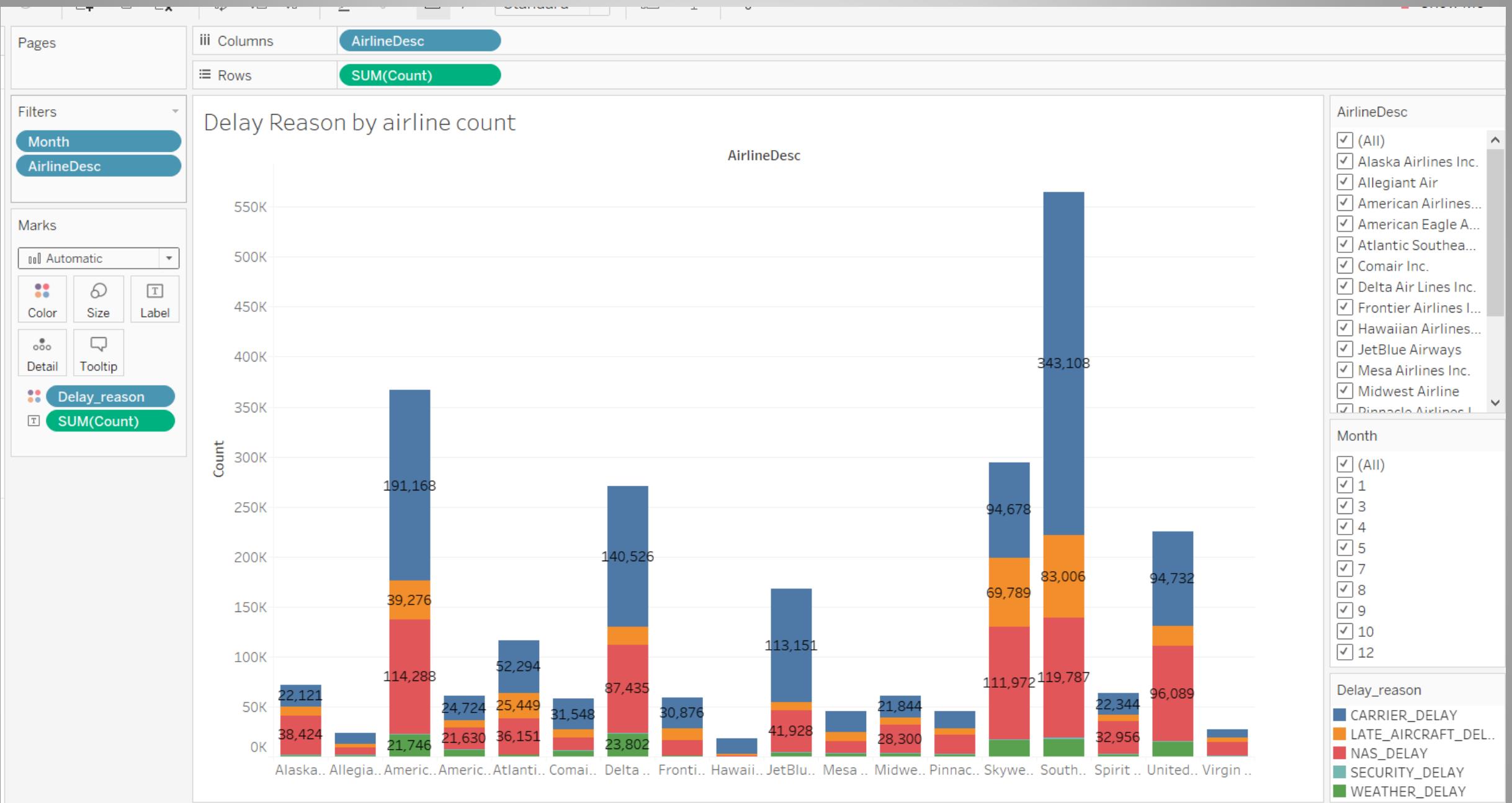
Month

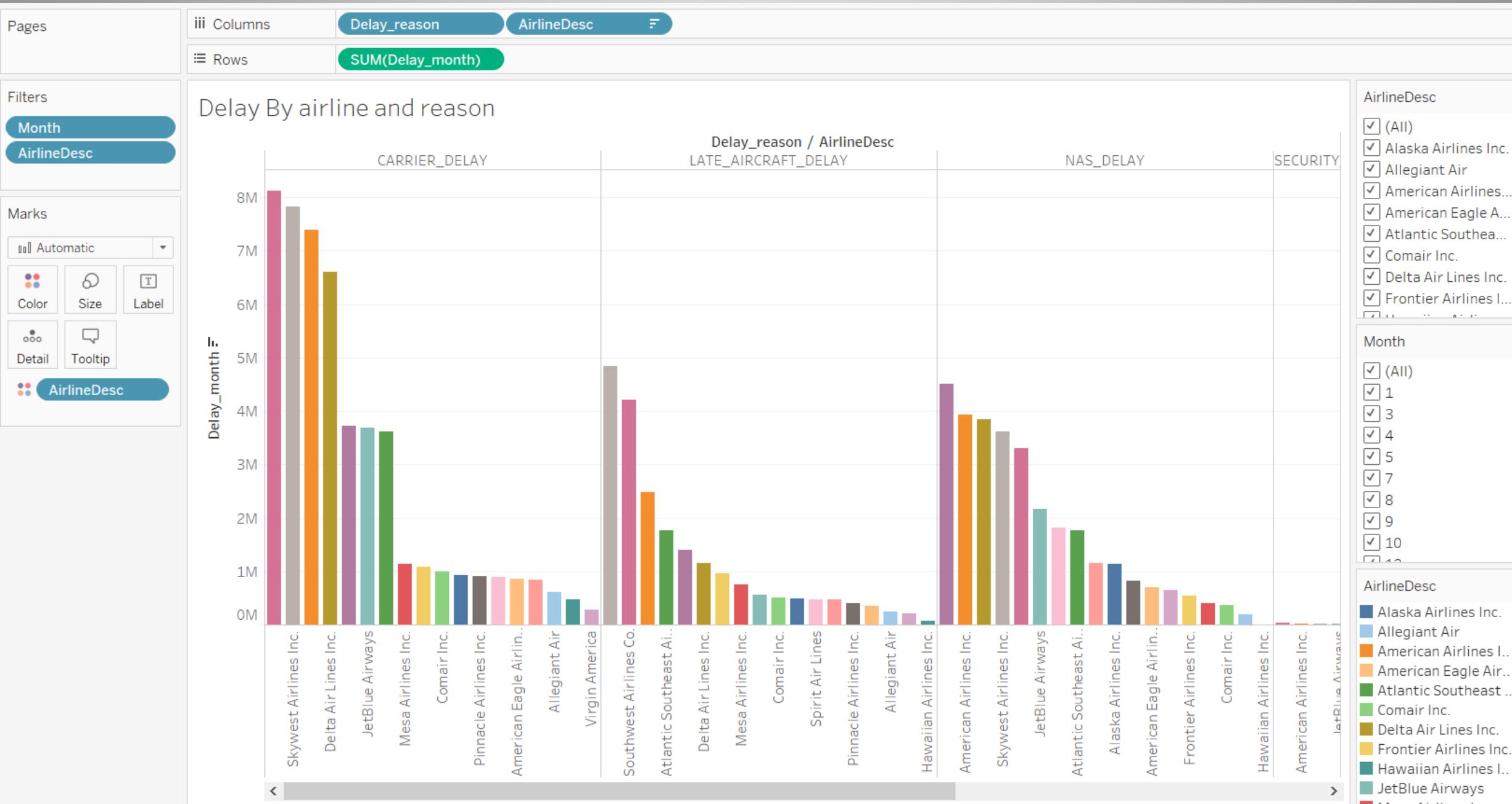
- (All)
- 1
- 3
- 4
- 5
- 7
- 8
- 9
- 10
- 12

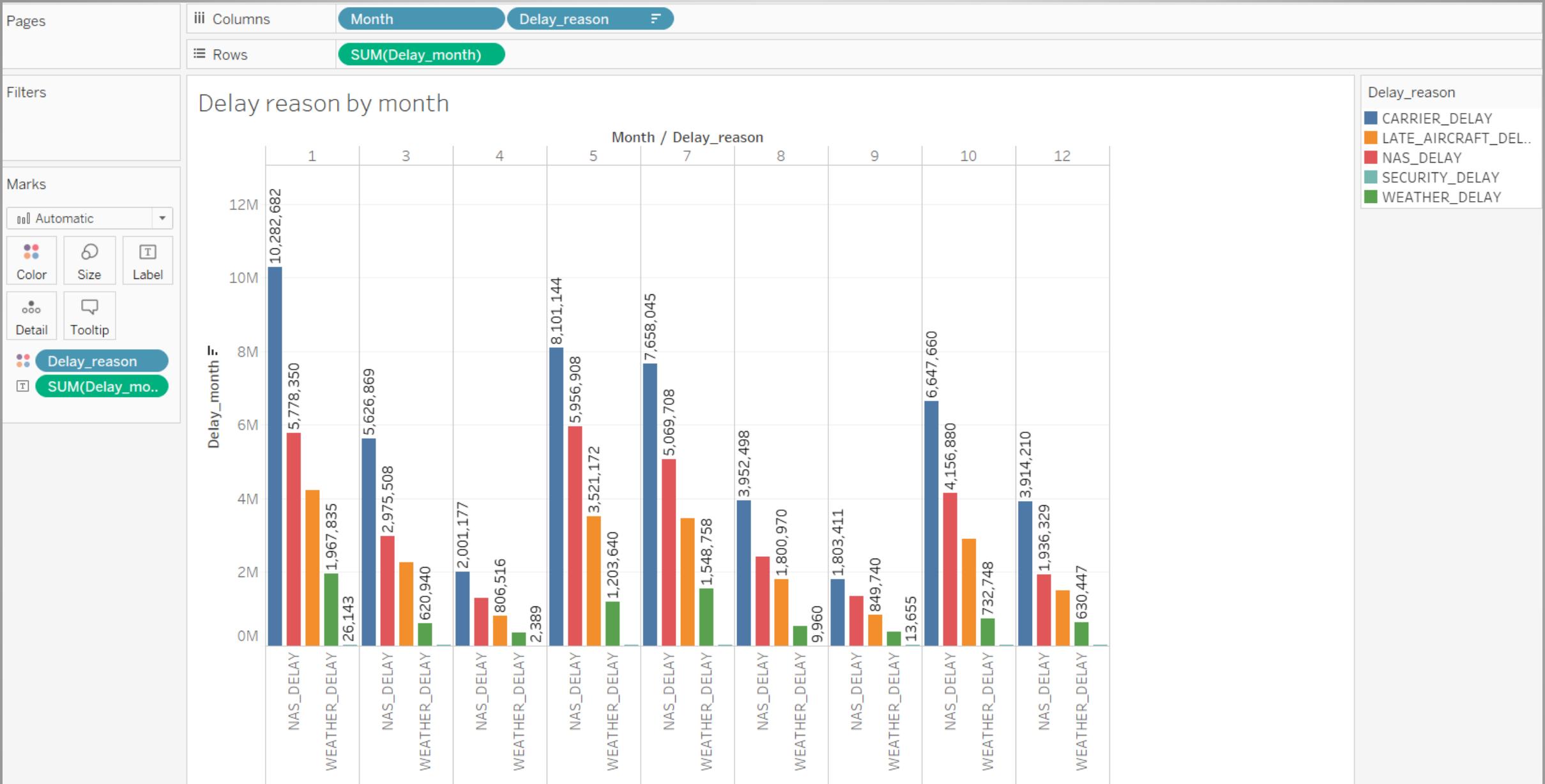
Canc_Reason

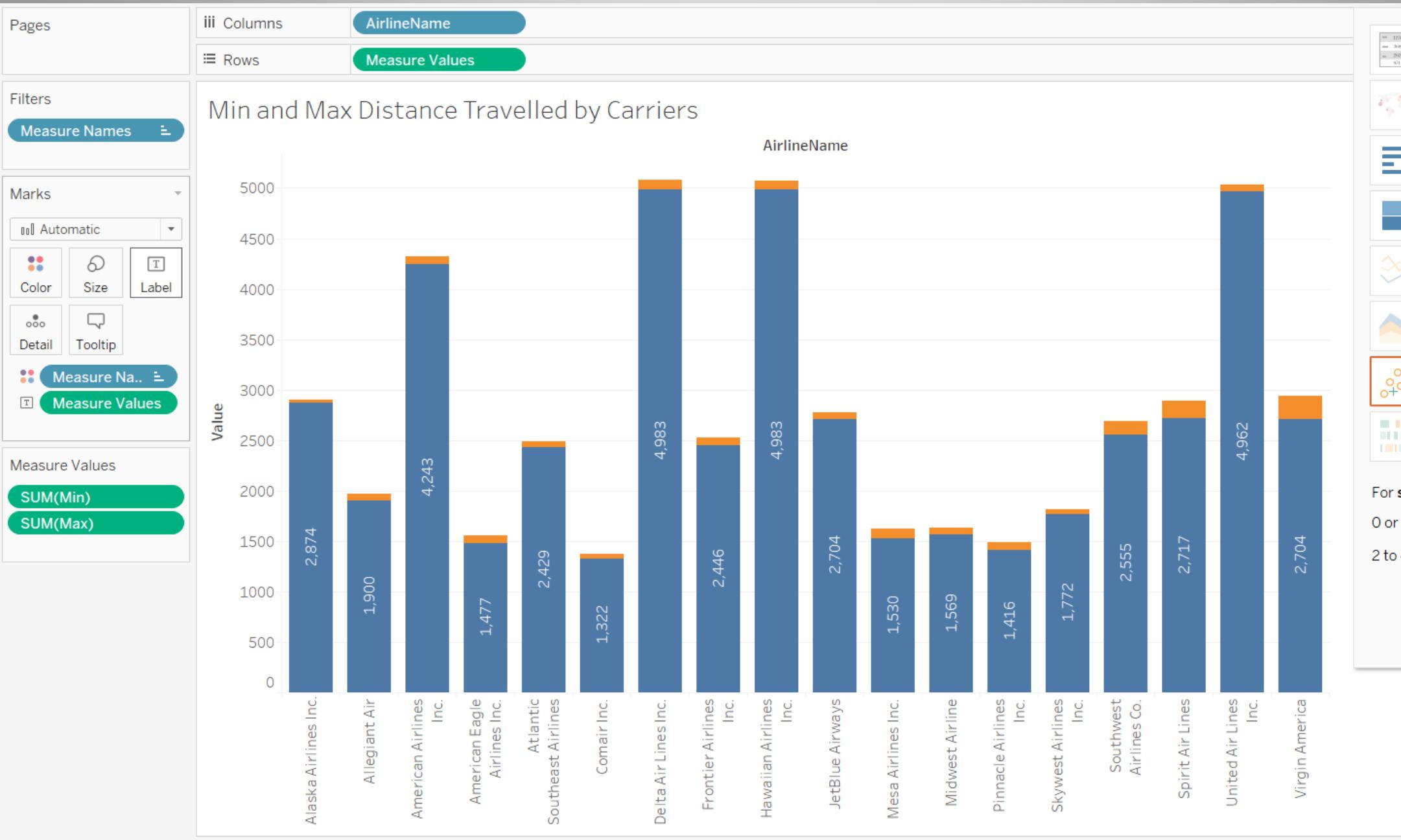
- A
- B
- C
- D

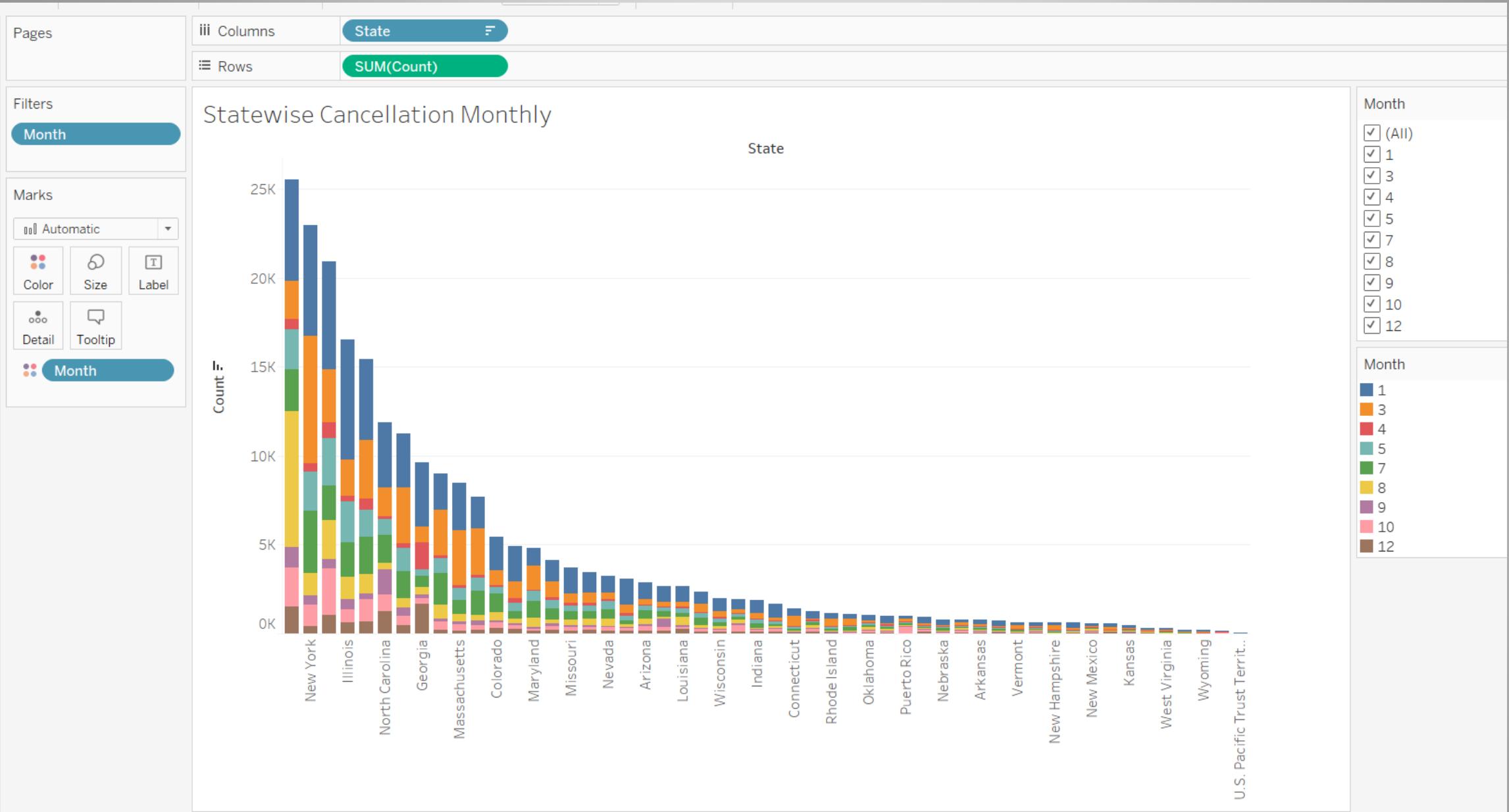




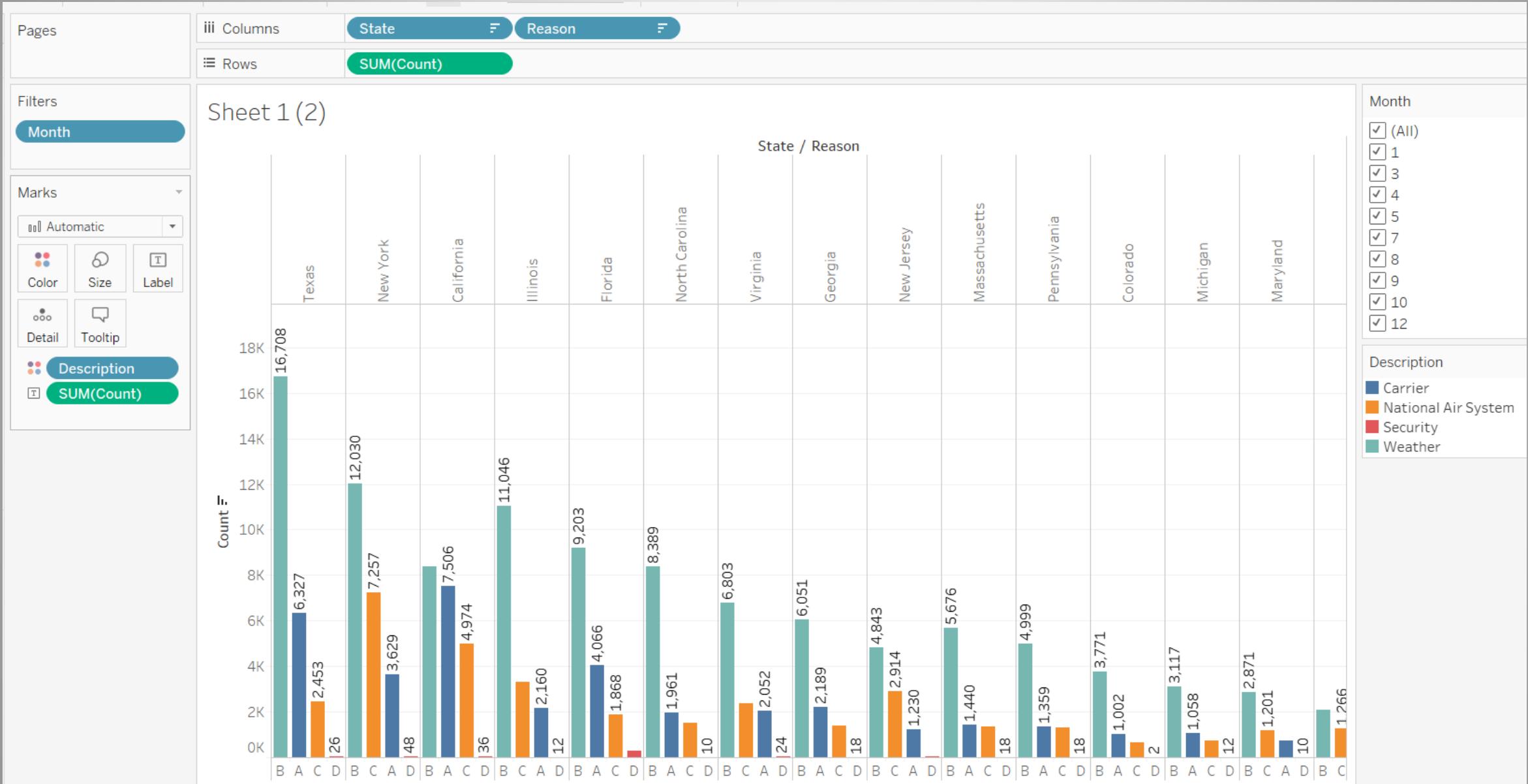


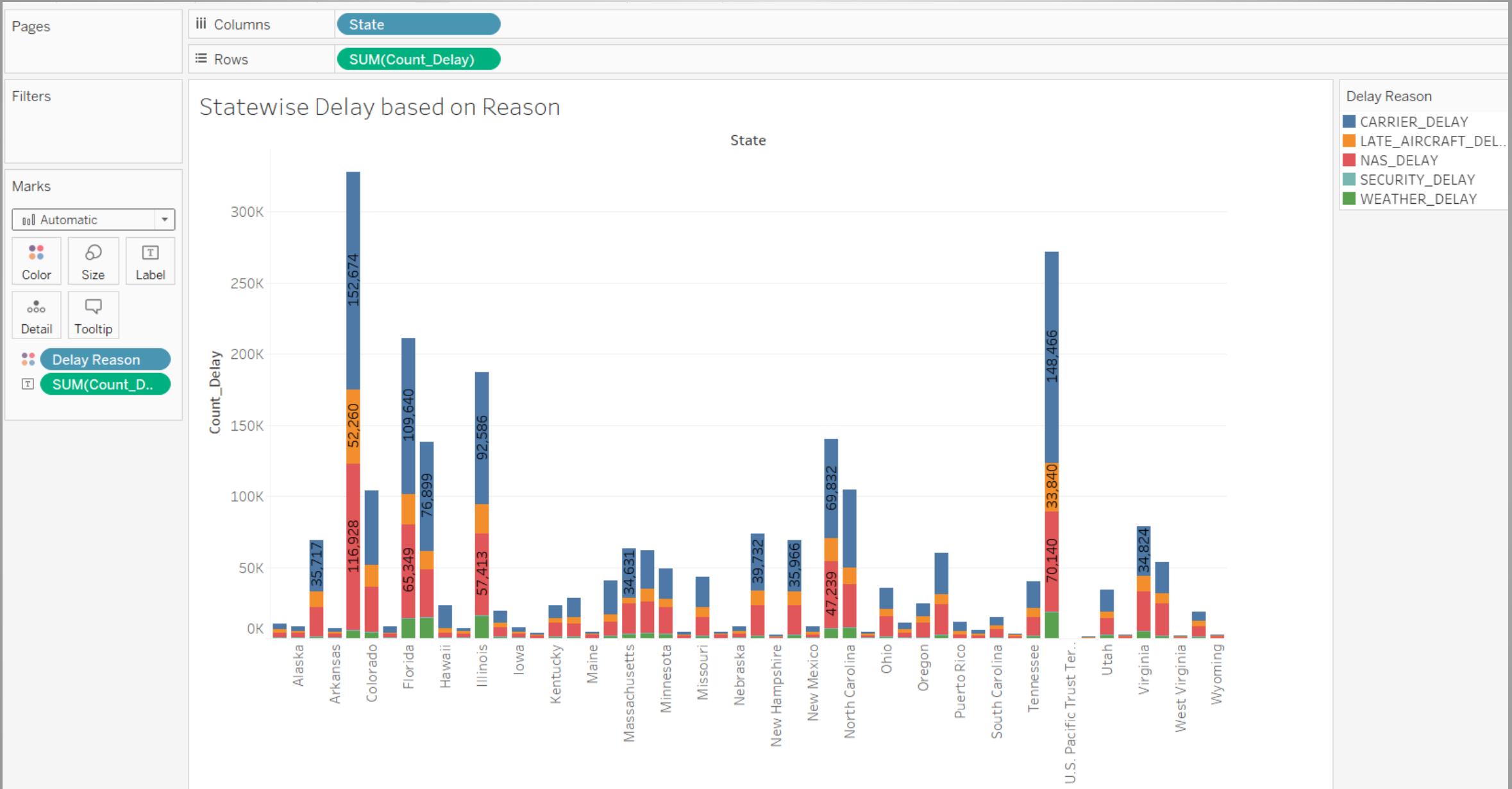


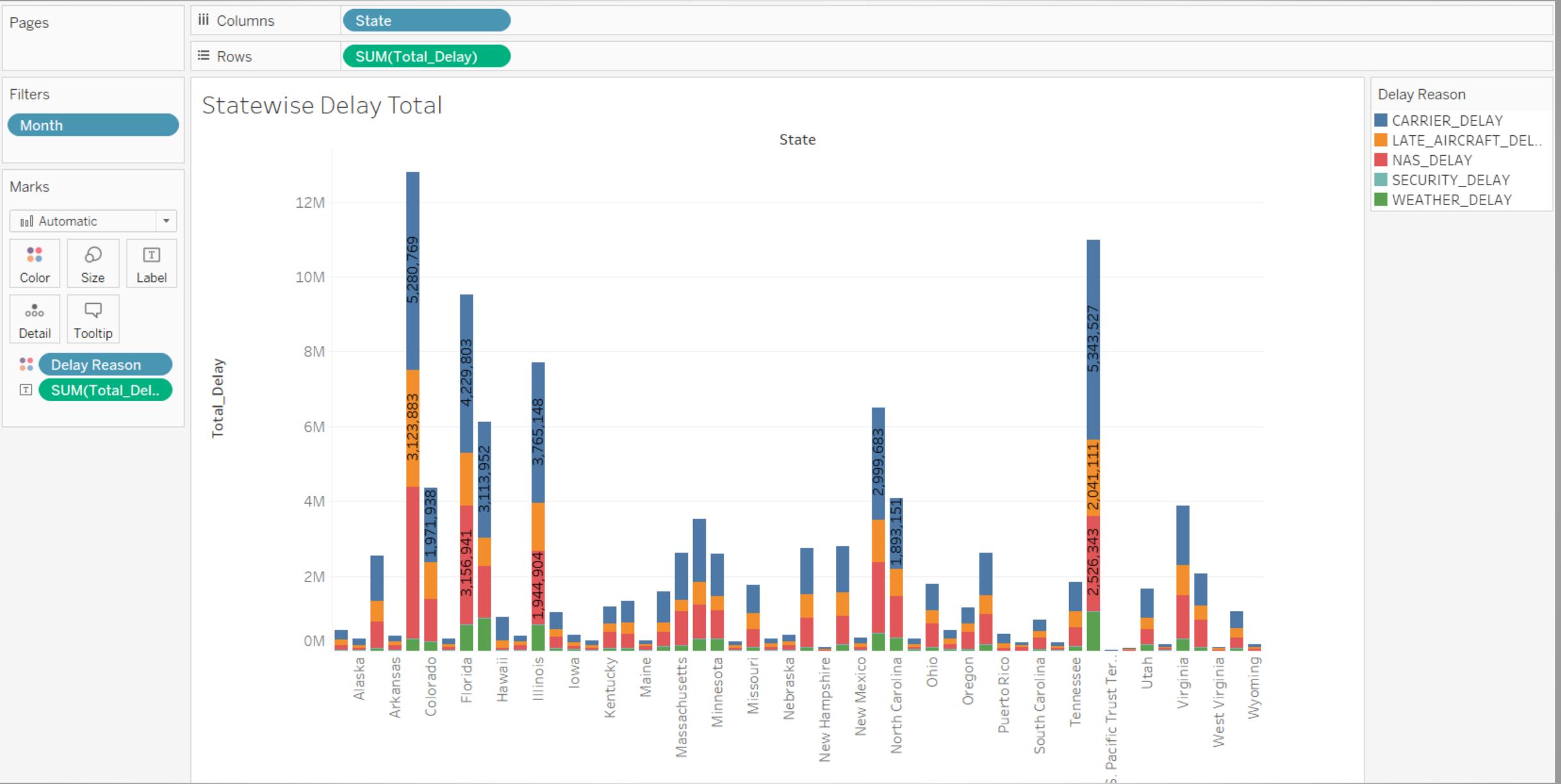




STATEWISE BASED CANCELLATION COUNT







ANALYSIS OUTPUTS

PutMerge:

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

FlightDelay OMerge Merge

MergeFiles

Project

ir.java TopKReducer.java TopKMapper.java DelayCountMapper.java App.java OTopDistanceValues/part-r-00000 OMSDMedian/part-r-00000 MergeFiles.java

The file is too large: 121.25 MB. Showing a read-only preview of the first 2.56 MB.

YEAR,QUARTER,MONTH,DAY_OF_MONTH,DAY_OF_WEEK,FL_DATE,OP_UNIQUE_CARRIER,TAIL_NUM,OP_CARRIER_FL_NUM,ORIGIN_AIRPORT_ID,ORIGIN,ORIGIN_CITY_NAME,ORIGIN_STATE_ABR,ORIGIN_STATE_NM,DEST_AIRPORT_ID,DEST

2018,1,3,1,4,2018-03-01,F9,N201FR,1680,12889,LAS,"Las Vegas, NV",NV,Nevada,13204,MCO,"Orlando, FL",FL,Florida,30,221,111,111,18,1011,8,756,1019,143,143,0,,0,266,298,272,1,2039,9,20,0,32,0,91

2018,1,3,1,4,2018-03-01,F9,N201FR,681,13204,MCO,"Orlando, FL",FL,Florida,11292,DEN,"Denver, CO",CO,Colorado,900,1108,128,128,12,1302,8,1115,1310,115,115,0,,0,255,242,222,1,1546,7,4,0,0,0,111

2018,1,3,1,4,2018-03-01,F9,N201FR,681,11292,DEN,"Denver, CO",CO,Colorado,14747,SEA,"Seattle, WA",WA,Washington,1215,1358,103,103,13,1534,6,1413,1540,87,87,0,,0,178,162,143,1,1024,5,0,0,0,0,87

2018,1,3,1,4,2018-03-01,F9,N201FR,144,14747,SEA,"Seattle, WA",WA,Washington,11292,DEN,"Denver, CO",CO,Colorado,1514,1621,67,67,11,1940,8,1855,1948,53,53,0,,0,161,147,128,1,1024,5,0,0,0,0,53

2018,1,3,1,4,2018-03-01,F9,N201FR,122,11292,DFW,"Dallas/Fort Worth, TX",TX,Texas,1955,2035,40,40,9,2307,10,2252,2317,25,25,0,,0,117,102,83,1,641,3,0,0,0,0,25

2018,1,3,1,4,2018-03-01,F9,N202FR,1138,12889,LAS,"Las Vegas, NV",NV,Nevada,10423,AUS,"Austin, TX",TX,Texas,845,836,-9,0,14,1306,10,1331,1316,-15,0,0,,0,166,160,136,1,1090,5,,,,

2018,1,3,1,4,2018-03-01,F9,N202FR,1141,10423,AUS,"Austin, TX",TX,Texas,12889,LAS,"Las Vegas, NV",NV,Nevada,1421,1416,-5,0,12,1505,6,1525,1511,-14,0,0,,0,184,175,157,1,1090,5,,,,

2018,1,3,1,4,2018-03-01,F9,N202FR,1106,12889,LAS,"Las Vegas, NV",NV,Nevada,12266,IAH,"Houston, TX",TX,Texas,1615,1611,-4,0,31,2108,7,2117,2115,-2,0,0,,0,182,184,146,1,1222,5,,,,

2018,1,3,1,4,2018-03-01,F9,N202FR,1111,12266,IAH,"Houston, TX",TX,Texas,12889,LAS,"Las Vegas, NV",NV,Nevada,2207,2200,-7,0,20,2314,7,2342,2321,-21,0,0,,0,215,201,174,1,1222,5,,,,

2018,1,3,1,4,2018-03-01,F9,N203FR,1236,13487,MSP,"Minneapolis, MN",MN,Minnesota,15304,TPA,"Tampa, FL",FL,Florida,704,700,-4,0,50,1132,8,1120,1140,20,20,0,,0,196,220,162,1,1306,6,0,0,20,0,0

2018,1,3,1,4,2018-03-01,F9,N203FR,1201,15304,TPA,"Tampa, FL",FL,Florida,12339,IND,"Indianapolis, IN",IN,Indiana,1210,1225,15,15,13,1431,10,1430,1441,11,11,0,,0,140,136,113,1,837,4,,,,

2018,1,3,1,4,2018-03-01,F9,N203FR,1204,12339,IND,"Indianapolis, IN",IN,Indiana,15304,TPA,"Tampa, FL",FL,Florida,1520,1622,62,62,16,1831,6,1733,1837,64,64,0,,0,133,135,113,1,837,4,56,0,8,0,0

2018,1,3,1,4,2018-03-01,F9,N203FR,1209,15304,TPA,"Tampa, FL",FL,Florida,13342,MKE,"Milwaukee, WI",WI,Wisconsin,1825,1939,74,74,17,2125,5,2017,2130,73,73,0,,0,172,171,149,1,1075,5,16,0,0,57

2018,1,3,1,4,2018-03-01,F9,N203FR,1099,13342,MKE,"Milwaukee, WI",WI,Wisconsin,12889,LAS,"Las Vegas, NV",NV,Nevada,2107,2214,67,67,19,6,7,2259,13,74,74,0,,0,232,239,213,1,1524,7,0,0,7,0,67

2018,1,3,1,4,2018-03-01,F9,N205FR,1977,13204,MCO,"Orlando, FL",FL,Florida,12339,IND,"Indianapolis, IN",IN,Indiana,935,935,0,0,13,1149,7,1207,1156,-11,0,0,,0,152,141,121,1,829,4,,,,

2018,1,3,1,4,2018-03-01,F9,N205FR,1977,12339,IND,"Indianapolis, IN",IN,Indiana,12889,LAS,"Las Vegas, NV",NV,Nevada,1259,1306,7,7,34,1424,9,1414,1433,19,19,0,,0,255,267,224,1,1590,7,7,0,12,0,0

"YEAR","QUARTER","MONTH","DAY_OF_MONTH","DAY_OF_WEEK","FL_DATE","OP_UNIQUE_CARRIER","TAIL_NUM","OP_CARRIER_FL_NUM","ORIGIN_AIRPORT_ID","ORIGIN","ORIGIN_CITY_NAME","ORIGIN_STATE_ABR","ORIGIN_STA

2017,2,4,1,6,2017-04-01,"DL","N3755D","2",12478,"JFK","New York, NY","NY","New York",14679,"SAN","San Diego, CA","CA","California",1925,"1925",0.00,0.00,36.00,"2216",4.00,"2240",2220,-20.00

2017,2,4,1,6,2017-04-01,"DL","N37415","4",12889,"LAS","Las Vegas, NV",NV,"Nevada",12478,"JFK","New York, NY","NY","New York",1024,"1129",65.00,65.00,13.00,"1912",8.00,"1829","1920",51.00,51

2017,2,4,1,6,2017-04-01,"DL","N3743H","7",12889,"LAS","Las Vegas, NV",NV,"Nevada",12892,"LAX","Los Angeles, CA",CA,"California",0802,"0801",-1.00,0.00,12.00,"0852",10.00,"0919","0902",-17

2017,2,4,1,6,2017-04-01,"DL","N910DN","11",13487,"MSP","Minneapolis, MN",MN,"Minnesota",11292,DEN,"Denver, CO",CO,"Colorado",1800,"1800",0.00,0.00,13.00,"1855",9.00,"1920","1904",-16.00

2017,2,4,1,6,2017-04-01,"DL","N658DL","15",10397,"ATL","Atlanta, GA",GA,"Georgia",14683,"SAT","San Antonio, TX",TX,"Texas",1835,"1832",-3.00,0.00,16.00,"1953",6.00,"2011","1959",-12.00,0.0

2017,2,4,1,6,2017-04-01,"DL","N591NW","16",13204,"MCO","Orlando, FL",FL,"Florida",11433,"DTW","Detroit, MI",MI,"Michigan",1400,"1358",-2.00,0.00,15.00,"1617",7.00,"1642",1624,-18.00,0.0

2017,2,4,1,6,2017-04-01,"DL","N592NW","17",11433,"DTW","Detroit, MI",MI,"Michigan",13204,"MCO","Orlando, FL",FL,"Florida",1517,"1522",5.00,5.00,12.00,"1741",11.00,"1753","1752",-1.00,0.00

2017,2,4,1,6,2017-04-01,"DL","N048AT","19",11433,"DTW","Detroit, MI",MI,"Michigan",13108,"MCT","Kansas City, MO",MO,"Missouri",2018,"2000",9.00,0.00,15.00,"2050",5.00,"2122","2104",-18.00

AIRLINE DELAY:

Calculated Average Delay by the airliner each month used Custom Writable class for this technique.

The screenshot shows a Java project structure and a file browser interface.

Java Project Structure:

- Project: OTopDistancevalues
- src/main/java:
 - AirlineCancel7
 - AirlineDelay7
 - AverageFlightDelay
 - AvgDelayMonthly
 - AirlineTextPair
 - AverageFlightDelayByMonth
 - DistinctFlight

Code Editor (Java Class Definition):

```
public class AirlineTextPair implements WritableComparable {
    // Airline's UniqueCarrier
    Text airLineName;
    // Airlines fly Month
    Text month;

    /**
     * constructor
     */
    public AirlineTextPair() {
        this.airLineName = new Text();
        this.month = new Text();
    }
}
```

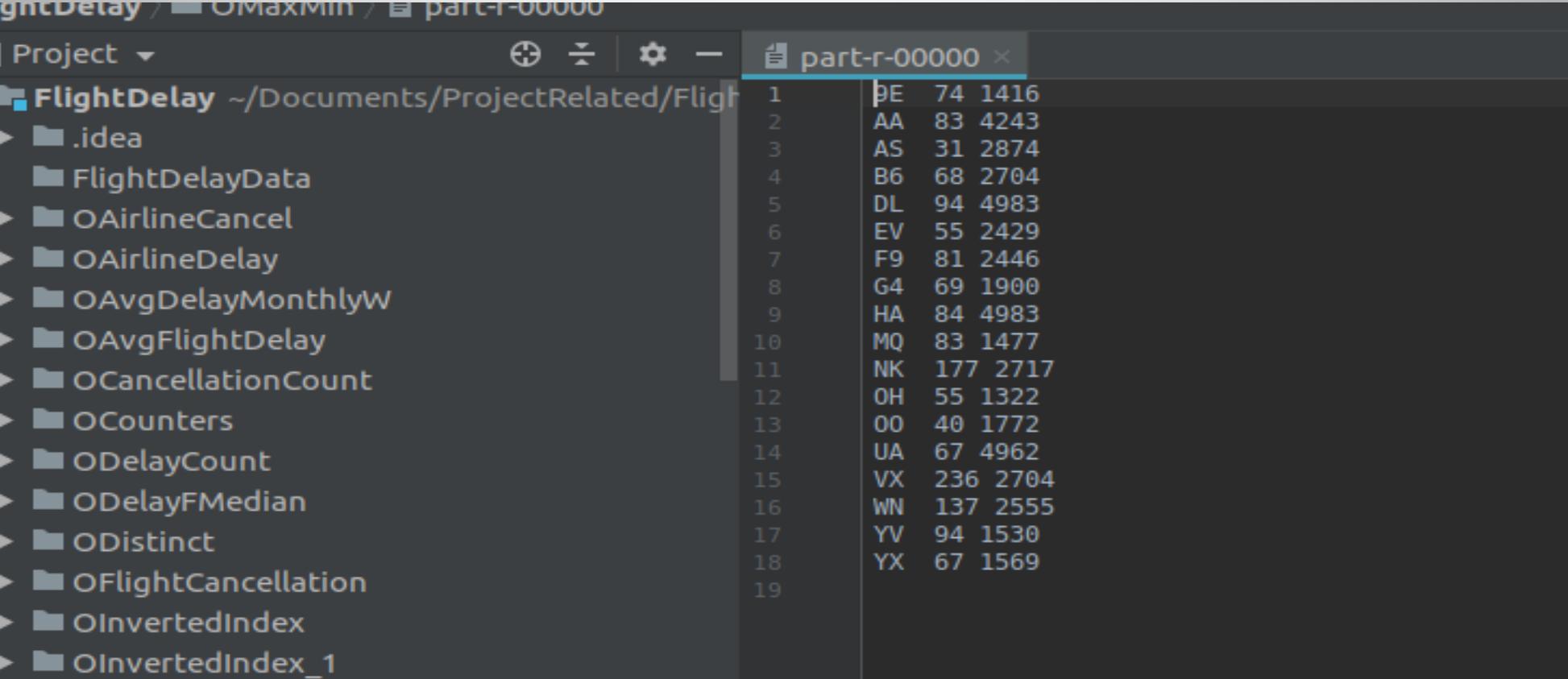
File Browser (HDFS Partitions):

FlightDelay / OAvgDelayMonthlyW / part-r-00000

| Index | Value |
|-------|--|
| 1 | 9E, (1,1473), (3,1479), (5,1467), (7,1448), (9,1469), (10,1472), (12,1470) |
| 2 | AA, (1,1491), (3,1482), (4,1483), (5,1474), (7,1451), (8,1469), (9,1461), (10,1478), (12,1478) |
| 3 | AS, (1,1475), (3,1488), (4,1457), (5,1459), (7,1448), (8,1450), (9,1483), (10,1481), (12,1460) |
| 4 | BG, (1,1484), (3,1496), (4,1484), (5,1483), (7,1469), (8,1450), (9,1470), (10,1465), (12,1455) |
| 5 | DL, (1,1486), (3,1485), (4,1476), (5,1476), (7,1465), (8,1480), (9,1487), (10,1490), (12,1493) |
| 6 | EV, (1,1471), (3,1466), (4,1465), (5,1458), (7,1466), (8,1459), (9,1458), (10,1466), (12,1469) |
| 7 | F9, (1,1451), (3,1452), (4,1454), (5,1428), (7,1412), (8,1432), (9,1426), (10,1436), (12,1454) |
| 8 | GU, (1,1472), (3,1473), (4,1472), (5,1472), (7,1472), (8,1472), (9,1472), (10,1472), (12,1453) |
| 9 | HA, (1,1418), (3,1409), (4,1378), (5,1390), (7,1411), (8,1409), (9,1407), (10,1406), (12,1426) |
| 10 | MQ, (1,1464), (3,1462), (5,1455), (7,1458), (9,1456), (10,1458), (12,1459) |
| 11 | NK, (1,1465), (3,1464), (4,1461), (5,1442), (7,1426), (8,1432), (9,1448), (10,1456), (12,1464) |
| 12 | OH, (1,1474), (3,1489), (4,1469), (5,1469), (7,1466), (8,1460), (9,1465), (10,1464), (12,1454) |
| 13 | OO, (1,1475), (3,1467), (4,1462), (5,1460), (7,1455), (8,1453), (9,1456), (10,1457), (12,1466) |
| 14 | UA, (1,1482), (3,1473), (4,1463), (5,1455), (7,1413), (8,1420), (9,1455), (10,1466), (12,1472) |
| 15 | VX, (1,1541), (3,1519), (4,1538), (5,1534), (7,1529), (8,1529), (10,1516), (12,1514) |
| 16 | WN, (1,1471), (3,1471), (4,1471), (5,1471), (7,1471), (8,1471), (9,1483), (10,1484), (12,1465) |
| 17 | YV, (1,1494), (3,1474), (4,1474), (5,1477), (7,1473), (8,1484), (9,1486), (10,1486), (12,1486) |
| 18 | YY, (1,1490), (3,1485), (5,1471), (7,1461), (9,1469), (10,1476), (12,1478) |
| 19 | |

MIN-MAX DELAY:

Determined the Minimum and Maximum Distance travelled by flight based on each Airliner.



The screenshot shows a terminal window with the title bar "FlightDelay / MaxMin / part-r-00000". The left pane displays a project structure under "FlightDelay" with various sub-directories and files. The right pane shows the content of the file "part-r-00000" which contains 19 rows of data, each consisting of four columns: a number from 1 to 19, followed by three letters, a number, and another number. The data is as follows:

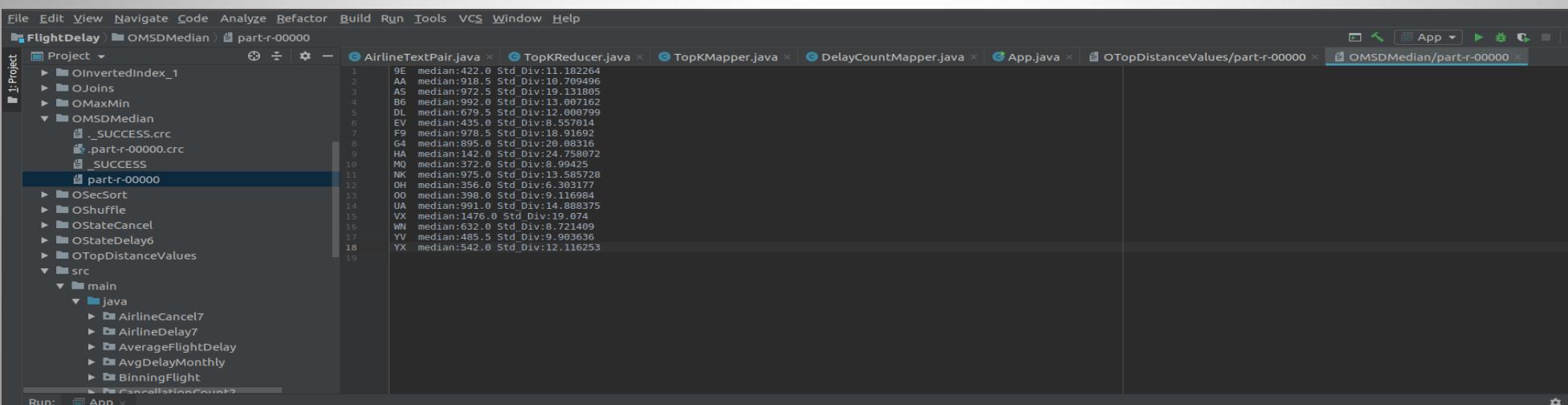
| 1 | PE | 74 | 1416 |
|----|----|-----|------|
| 2 | AA | 83 | 4243 |
| 3 | AS | 31 | 2874 |
| 4 | B6 | 68 | 2704 |
| 5 | DL | 94 | 4983 |
| 6 | EV | 55 | 2429 |
| 7 | F9 | 81 | 2446 |
| 8 | G4 | 69 | 1900 |
| 9 | HA | 84 | 4983 |
| 10 | MQ | 83 | 1477 |
| 11 | NK | 177 | 2717 |
| 12 | OH | 55 | 1322 |
| 13 | OO | 40 | 1772 |
| 14 | UA | 67 | 4962 |
| 15 | VX | 236 | 2704 |
| 16 | WN | 137 | 2555 |
| 17 | YV | 94 | 1530 |
| 18 | YX | 67 | 1569 |
| 19 | | | |

MEDIAN

Determined Delay Median based on Airliner

```
protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
    Integer median=0;
    //int sum = 0;
    //int count = 0;

    /Integer temp= (int) Float.parseFloat(key.toString());
    try {
        for (Text value:values) {
            Integer temp = (int) Float.parseFloat(value.toString());
            if (temp != 0){
                array.add(temp);
            }
            //context.write(new Text(key), NullWritable.get());
        }
        Collections.sort(array);
        int len=array.size();
        if(array.size()%2==0){
            // median= array.get((int) len/2 - 1)+array.get((int) len/2);
            median= (array.get((int) len/2 - 1)+array.get((int) len/2)) /2;
        }
        else{
            median= array.get(len/2);
        }
        context.write(key,new Text( string: median+""));
    }
}
```



MEMORY CONSCIOUS MEDIAN

Used Memory-Conscious Median and Standard Deviation of Distance travelled based on Airliner used separate Combiner to simplify the input to Reducer.

The screenshot shows an IDE interface with two main sections: a code editor and a terminal window.

Code Editor: The code editor displays a Java class named `myCombiner` which extends `Reducer<Text, SortedMapWritable, Text, SortedMapWritable>`. The implementation of the `reduce` method is shown, where it iterates over the values (which are `SortedMapWritable` objects) for a given key. It then processes each entry in these maps to calculate the sum of values. If the count is not null, it adds the current value to the sum; if it is null, it sets the sum to the current value. Finally, it writes the result back to the context.

```
► AirlineDelay7
► AverageFlightDelay
► AvgDelayMonthly
► BinningFlight
► CancellationCount2
► CountersDistanceGroup
► DelayCount3
► DelayFMedian
► DistanceMax
► Distinct
▼ FlightCarrierDistanceMedian
  Driver
  MedianStdDevTuple
  myCombiner
  MyMapper
  MyReducer
► InverteBloomChain
3   import ...
4
5   public class myCombiner extends Reducer<Text, SortedMapWritable, Text, SortedMapWritable> {
6       @Override
7       protected void reduce(Text key, Iterable<SortedMapWritable> values, Context context) throws IOException, InterruptedException {
8           // int sum=0;
9           SortedMapWritable map = new SortedMapWritable();
10          for(SortedMapWritable v:values) {
11              for(Object entry : v.entrySet() ) {
12                  if(entry instanceof Map.Entry) {
13                      Map.Entry entry1 = (Map.Entry) entry;
14                      IntWritable count = (IntWritable) map.get(entry1.getKey());
15                      FloatWritable mapKey = ((FloatWritable) entry1.getKey());
16                      Integer mapValue = ((IntWritable) entry1.getValue()).get();
17                      if (count != null) {
18                          map.put(mapKey, new IntWritable( count.get() + ((IntWritable) entry1.getValue()).get()));
19                      } else {
20                          map.put(mapKey, new IntWritable(((IntWritable) entry1.getValue()).get()));
21                      }
22                  }
23              }
24          }
25          v.clear();
26      }
27      context.write(key,map);
28  }
29
30 }
```

Terminal Window: The terminal window shows the output of a command, likely `hadoop jar`, displaying a list of airline codes and their corresponding median and standard deviation values.

| Airline | median | Std Div |
|---------|--------|------------|
| 9E | 422.0 | 11.182264 |
| AA | 918.5 | 10.709496 |
| AS | 972.5 | 19.131805 |
| B6 | 992.0 | 13.007162 |
| DL | 679.5 | 12.0000799 |
| EV | 437.0 | 18.55104 |
| F9 | 781.5 | 18.91602 |
| G4 | 895.0 | 20.98316 |
| HA | 142.0 | 24.758072 |
| MQ | 372.0 | 8.99425 |
| NK | 975.0 | 13.585728 |
| OH | 356.0 | 6.303177 |
| OO | 398.0 | 9.116984 |
| UA | 991.0 | 14.888375 |
| VX | 1476.0 | 19.074 |
| WN | 632.0 | 8.721409 |
| YY | 485.5 | 9.903636 |
| YX | 542.0 | 12.116253 |

COUNTING WITH COUNTERS

Used counting with Counters to determine Number of Flights in different Distance Groups

The screenshot shows an IDE interface with a file tree on the left and a code editor on the right.

File Tree:

- .idea
- FlightDelayData
- OAirlineCancel
- OAirlineDelay
- OAvgDelayMonthlyW
- OAvgFlightDelay
- OCancellationCount
- OCounters
- OFlightCancellation
- src
- main
- java
- AirlineCancel7
- AirlineDelay7
- AverageFlightDelay
- AvgDelayMonthly
- BinningFlight
- CancellationCount2
- CountersDistanceGroup

Code Editor (Driver.java):

```
24 job.setMapperClass(MyMapper.class);
25
26
27
28
29
30     job.setOutputKeyClass(NullWritable.class);
31     job.setOutputValueClass(NullWritable.class);
32
33
34
35
36     int code = job.waitForCompletion( verbose: true)? 1 : 0;
37     if (code==1) {
38         System.out.print("the counter output:\n");
39         for(Counter counter: job.get_counters().getGroup( groupName: "DistanceCounter")) {
40             // System.out.println(counter.getDisplayName() +":\t" +counter.getValue());
41             // System.out.println("the counter values is "+counter.getValue());
42         }
43     }
44 }
45 }
46
47 }
```

Run Log:

```
File Output Format Counters
Bytes Written=8
the counter output:
1: 1731883
10: 385284
11: 282962
2: 3289052
3: 2669652
4: 2092525
5: 1431105
6: 604557
7: 631685
8: 325164
9: 227472
Process finished with exit code 0
```

INVERTED INDEX:

Used Filtered output and determined the Flights from the Airport using Inverted Index Method.

The screenshot shows a Java IDE interface with the following details:

- File Bar:** File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
- Title Bar:** FlightDelay > OInvertedIndex > part-r-00000
- Toolbar:** Includes icons for New, Open, Save, Cut, Copy, Paste, Find, and others.
- Project Explorer:** Shows the project structure under "1: Project".
 - FlightDelay
 - OAvgFlightDelay
 - OCancellationCount
 - OCounters
 - ODelayCount
 - ODelayFMedian
 - ODistinct
 - OFlightCancellation
 - OInvertedIndex
 - _SUCCESS.crc
 - .part-r-00000.crc
 - _SUCCESS
 - part-r-00000
 - OInvertedIndex_1
 - OJoins
 - OMaxMin
 - OMSDMedian
 - OSecSort
 - OShuffle
 - OStateCancel
- Code Editor:** Displays a portion of the "part-r-00000" file content.

```
BOS YV,9E,AA,AS,B6,DL,EV,MQ,NK,00,UA,WN,YX,VX,YV,YX,9E,AS,B6,DL,EV,MQ,NK,00,UA,WN,AA,,AS,B6,DL,EV,NK,00,UA,VX,WN,YV,YX,9E,AA,,9E,AA,AS,B6,DL,EV,MQ,NK,00,UA,YV,YX,WN,,00,AS,B6,DL,EV,NK,UA,VX,WN,AA  
JFK HA,MQ,OH,AS,DL,YX,VX,AA,9E,B6,,OH,00,HA,MQ,VX,YX,AA,AS,DL,B6,9E,,AS,VX,DL,HA,B6,AA,,AS,HA,OH,00,MQ,YX,9E,AA,B6,DL,,AS,HA,MQ,OH,YX,00,9E,AA,B6,DL,,
```
- Toolbars:** Includes MCDMedian, Find, Replace, and other development tools.

SECONDARY SORTING:

Did Secondary Sorting of the States and the available Airports in that state to know the list of Airports Present in US.(Partitioner is used in this case)

The screenshot shows a Java IDE interface with a dark theme. The top menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The title bar indicates the project is "FlightDelay > OSecSort > part-r-00000".

The left sidebar displays the project structure under "Project":

- ODelayCount
- ODelayFMedian
- ODistinct
- OFlightCancellation
- OInvertedIndex
- OInvertedIndex_1
- OJoins
- OMaxMin
- OMSDMedian
- OSecSort** (selected)
 - _.SUCCESS.crc
 - .part-r-00000crc
 - _SUCCESS
 - part-r-00000** (selected)
- OShuffle
- OStateCancel
- OStateDelay6
- src** (selected)
 - main
 - java** (selected)
 - AirlineCancel7
 - AirlineDelay7

JOINS:

Based on the Carrier Names from a different Carrier CSV joined the Full name of the Flights present in the original Data.

| Code | Description |
|------|--|
| 02Q | Titan Airways |
| 04Q | Tradewind Aviation |
| 05Q | Comlux Aviation, AG |
| 06Q | Master Top Linhas Aereas Ltd. |
| 07Q | Flair Airlines Ltd. |
| 09Q | Swift Air, LLC |
| 0BQ | DCA |
| 0CQ | ACM AIR CHARTER GmbH |
| 0FQ | Maine Aviation Aircraft Charter, LLC |
| 0GQ | Inter Island Airways, d/b/a Inter Island Air |
| 0HQ | Polar Airlines de Mexico d/b/a Nova Air |
| 0J | JetClub AG |
| 0JQ | Vision Airlines |
| 0KQ | Mokulele Flight Services, Inc. |
| 0LQ | Metropix UK, LLP |
| 0MQ | Multi-Aero, Inc. d/b/a Air Choice One |
| 0Q | Flying Service N.V. |
| 16 | PSA Airlines Inc. |
| 17 | Piedmont Airlines |
| 1I | Sky Trek Int'l Airlines |
| 2E | Smokey Bay Air Inc. |
| 2F | Frontier Flying Service |
| 2M | Midway Express Airlines |
| 2O | Island Air Service |
| 2R | Regal Air |
| 2T | Canada 3000 Airlines Ltd. |
| 2U | Valley Air Express Inc. |
| 37 | Zeal 320 |
| 3C | Regions Air, Inc. |
| 3F | Pacific Airways, Inc. |
| 3M | Gulfstream Int |
| 3Z | Tatonduk Flying Service |
| 4B | Olson Air Service |
| 4E | Tanana Air Service |

The screenshot shows a file explorer window titled "FlightDelay > OJoins > part-r-00000". The left pane displays a project structure with several subfolders: OAvgFlightDelay, OCancellationCount, OCounters, ODelayCount, ODelayFMedian, ODistinct, OFlightCancellation, OInvertedIndex, OInvertedIndex_1, and OJoins. The "part-r-00000" file is currently selected. To the right of the project structure, two text files are displayed side-by-side:

ODelayCount/part-r-00000

| | | |
|----|----|------------------------------|
| 1 | 9E | Pinnacle Airlines Inc. |
| 2 | AA | American Airlines Inc. |
| 3 | AS | Alaska Airlines Inc. |
| 4 | B6 | JetBlue Airways |
| 5 | DL | Delta Air Lines Inc. |
| 6 | EV | Atlantic Southeast Airlines |
| 7 | F9 | Frontier Airlines Inc. |
| 8 | G4 | Allegiant Air |
| 9 | HA | Hawaiian Airlines Inc. |
| 10 | MQ | American Eagle Airlines Inc. |
| 11 | NK | Spirit Air Lines |
| 12 | OH | Comair Inc. |
| 13 | OO | Skywest Airlines Inc. |
| 14 | UA | United Air Lines Inc. |
| 15 | VX | Virgin America |
| 16 | WN | Southwest Airlines Co. |
| 17 | YV | Mesa Airlines Inc. |
| 18 | YX | Midwest Airline |
| 19 | | |

ODelayFMedian/part-r-00000

| | | |
|----|----|------------------------------|
| 1 | 9E | Pinnacle Airlines Inc. |
| 2 | AA | American Airlines Inc. |
| 3 | AS | Alaska Airlines Inc. |
| 4 | B6 | JetBlue Airways |
| 5 | DL | Delta Air Lines Inc. |
| 6 | EV | Atlantic Southeast Airlines |
| 7 | F9 | Frontier Airlines Inc. |
| 8 | G4 | Allegiant Air |
| 9 | HA | Hawaiian Airlines Inc. |
| 10 | MQ | American Eagle Airlines Inc. |
| 11 | NK | Spirit Air Lines |
| 12 | OH | Comair Inc. |
| 13 | OO | Skywest Airlines Inc. |
| 14 | UA | United Air Lines Inc. |
| 15 | VX | Virgin America |
| 16 | WN | Southwest Airlines Co. |
| 17 | YV | Mesa Airlines Inc. |
| 18 | YX | Midwest Airline |
| 19 | | |

BLOOM FILTER:

Used Bloom Filter to Filter the data based on the Airport and used Chaining

```
private BloomFilter<String> origin;

Funnel<String> p = (from, into) -> {
    into.putString(from,Charsets.UTF_8);
};

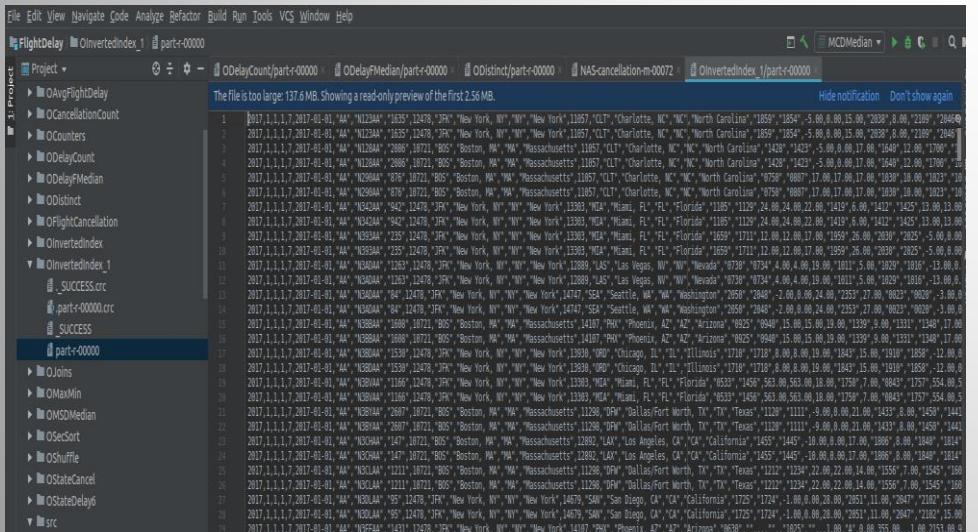
@Override
protected void setup(Mapper<LongWritable, Text, Text, NullWritable>.Context context) throws IOException, InterruptedException {

    this.origin = BloomFilter.create(p, expectedInsertions: 300000, falsePositiveProbability: 0.1);

    String p1 = "BOS";
    String p2 = "JFK";
    // Person p2 = new Person("Jamie", "Scott");

    ArrayList<String> originList = new ArrayList<->();
    originList.add(p1);
    originList.add(p2);

    for (String ps : originList) {
        origin.put(ps);
    }
}
```



```
job.setOutputFormatClass(TextOutputFormat.class);
job.setMapOutputValueClass(NullWritable.class);
job.setMapOutputKeyClass(Text.class);
job.setInputFormatClass(TextInputFormat.class);

// job.setNumReduceTasks(1);

Boolean a=job.waitForCompletion( verbose: true);

if(a==true)
{
    Job job2 = Job.getInstance();

    job2.setJarByClass(Driver.class);

    // job2.setGroupingComparatorClass(GroupComparator.class);
    // job2.setSortComparatorClass(SecondarySortComparator.class);
    //job2.setPartitionerClass(KeyPartition.class);

    TextInputFormat.addInputPath(job2, new Path( pathString: args[1]+"_1"));

    Path outDir = new Path(args[1]);
}
```

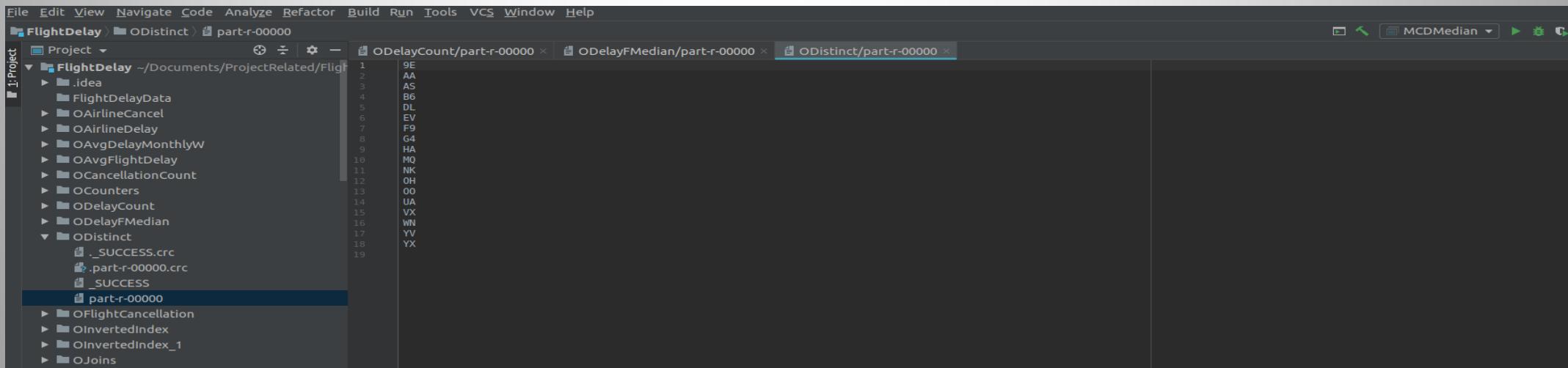
DISTINCT:

Determined Distinct Airline in the Dataset. (Used NLineInputFormat)

```
FileInputFormat.addInputPath(job2, new Path(args[0]));
Path outDir1 = new Path(args[1]);
FileOutputFormat.setOutputPath(job2, outDir1);

job2.setMapperClass(DistinctMapper.class);
job2.setReducerClass(DistinctReducer.class);
job2.setCombinerClass(DistinctReducer.class);

job2.setOutputFormatClass(TextOutputFormat.class);
job2.setMapOutputValueClass(NullWritable.class);
job2.setMapOutputKeyClass(Text.class);
job2.setInputFormatClass(NLineInputFormat.class);
NLineInputFormat.setNumLinesPerSplit(job2, numLines: 10000);
job2.setNumReduceTasks(1);
```



TOP K:

Determined Top 10 flight data based on longest Distance.

```
1 2018,1,3,1,4,2018-03-01,F9,N203FR,1204,12339,IND,"Indianapolis, IN",IN,Indiana,15304,TPA,"Tampa, FL",FL,Florida,1520,1622,62,62,16,1831,6,1733,1837,64,64,0,,0,133,135,113,1,837,4,56,0,8,0,0  
2 2018,1,3,1,4,2018-03-01,F9,N201FR,144,14747,SEA,"Seattle, WA",WA,Washington,11292,DEN,"Denver, CO",CO,Colorado,1514,1621,67,67,11,1940,8,1855,1948,53,53,0,,0,161,147,128,1,1024,5,0,0,0,0,53  
3 2018,1,3,1,4,2018-03-01,F9,N203FR,1209,15304,TPA,"Tampa, FL",FL,Florida,13342,MKE,"Milwaukee, WI",WI,Wisconsin,1825,1939,74,74,17,2125,5,2017,2130,73,73,0,,0,172,171,149,1,1075,5,16,0,0,0,57  
4 2018,1,3,1,4,2018-03-01,F9,N202FR,1141,10423,AUS,"Austin, TX",TX,Texas,12889,LAS,"Las Vegas, NV",NV,Nevada,1421,1416,-5,0,12,1505,6,1525,1511,-14,0,0,,0,184,175,157,1,1090,5,....  
5 2018,1,3,1,4,2018-03-01,F9,N202FR,1111,12266,IAH,"Houston, TX",TX,Texas,12889,LAS,"Las Vegas, NV",NV,Nevada,2207,2200,-7,0,20,2314,7,2342,2321,-21,0,0,,0,215,201,174,1,1222,5,....  
6 2018,1,3,1,4,2018-03-01,F9,N203FR,1236,13487,MSP,"Minneapolis, MN",MN,Minnesota,15304,TPA,"Tampa, FL",FL,Florida,704,700,-4,0,50,1132,8,1120,1140,20,20,0,,0,196,220,162,1,1306,6,0,0,20,0,0  
7 2018,1,3,1,4,2018-03-01,F9,N203FR,1099,13342,MKE,"Milwaukee, WI",WI,Wisconsin,12889,LAS,"Las Vegas, NV",NV,Nevada,2107,2214,67,67,19,6,7,2259,13,74,74,0,,0,232,239,213,1,1524,7,0,0,7,0,67  
8 2018,1,3,1,4,2018-03-01,F9,N201FR,681,13204,MCO,"Orlando, FL",FL,Florida,11292,DEN,"Denver, CO",CO,Colorado,900,1108,128,128,12,1302,8,1115,1310,115,115,0,,0,255,242,222,1,1546,7,4,0,0,0,111  
9 2018,1,3,1,4,2018-03-01,F9,N205FR,1977,12339,IND,"Indianapolis, IN",IN,Indiana,12889,LAS,"Las Vegas, NV",NV,Nevada,1259,1306,7,7,34,1424,9,1414,1433,19,19,0,,0,255,267,224,1,1590,7,7,0,12,0,0  
10 2018,1,3,1,4,2018-03-01,F9,N201FR,1680,12889,LAS,"Las Vegas, NV",NV,Nevada,13204,MCO,"Orlando, FL",FL,Florida,30,221,111,111,18,1011,8,756,1019,143,143,0,,0,266,298,272,1,2039,9,20,0,32,0,91  
11
```

SHUFFLING:

Used Shuffling Technique to shuffle the Data.

The file is too large: 3.54 GB. Showing a read-only preview of the first 2.56 MB.

| Line Number | Content Preview |
|-------------|---|
| 1 | 2018,2,5,20,7,2018-05-20,"YX","N434YX","4426",13303,"MIA","Miami, FL","FL","Florida",10693,"BNA","Nashville, TN","TN","Tennessee","1530","1528",-2.00,0.00,18.00,"1634",84.00,"1655", |
| 2 | 2018,1,3,19,1,2018-03-19,"B6","N637JB","500","DAB", "Daytona Beach, FL","FL","Florida",12478,"JFK","New York, NY","NY","New York","1132","1127",-5.00,0.00,13.00,"1340",3.00,"14 |
| 3 | 2017,2,5,2,2,2017-05-02,"AA","N391AA","5",11298,"DFW","Dallas/Fort Worth, TX","TX","Texas",12173,"HNL","Honolulu, HI","HI","Hawaii","1110","1219",69.00,60.00,18.00,"1519",3.00,"1444 |
| 4 | 2017,4,10,17,2,2017-10-17,"UA","N24211","696",12264,"IAD","Washington, DC","VA","Virginia",11292,"DEN","Denver, CO","CO","Colorado",1445,"1457",12.00,12.00,13.00,"1626",9.00,"1632 |
| 5 | 2018,3,9,18,2,2018-09-18,"DL","N866DN","2255",13487,"MSP","Minneapolis, MN","MN","Minnesota",13342,"MKE","Milwaukee, WI","WI","Wisconsin","2225","2224",-1.00,0.00,12.00,"2322",3.00, |
| 6 | 2017,4,10,7,6,2017-10-07,"OO","N9085W","5217",11292,"DEN","Denver, CO","CO","Colorado",11637,"FAR","Fargo, ND","ND","North Dakota","1525","1518",-7.00,0.00,13.00,"1753",3.00,"1820" |
| 7 | 2018,1,1,17,2,2018-01-07,"WN","N402WN","4346",14869,"SLC","Salt Lake City, UT","UT","Utah",11292,"DEN","Denver, CO","CO","Colorado",1540,"1603",23.00,23.00,13.00,"1712",5.00,"1700 |
| 8 | 2017,1,1,17,2,2017-01-17,"WN","N7847A","125",14771,"SFO","San Francisco, CA","CA","California",12889,"LAS","Las Vegas, NV","NV","Nevada",1345,"1341",-4.00,0.00,10.00,"1458",5.00," |
| 9 | 2017,3,7,10,1,2017-07-10,"AA","N3AMAA","279",12478,"JFK","New York, NY","NY","New York",13204,"MCO","Orlando, FL","FL","Florida",1559,"1728",89.00,89.00,"",,"1905",",,1.00,"C", |
| 10 | 2018,2,5,10,4,2018-05-10,"WN","N421LV","5814",14771,"SFO","San Francisco, CA","CA","California",12892,"LAX","Los Angeles, CA","CA","California",1810,"1808",-2.00,0.00,29.00,"1929" |
| 11 | 2018,4,10,23,2,2018-10-23,"AA","N5240W","475",13930,"ORD","Chicago, IL","IL","Illinois",14107,"PHX","Phoenix, AZ","AZ","Arizona",0710,"0717",7.00,7.00,11.00,"0848",4.00,"0901",08 |
| 12 | 2018,1,1,23,2,2018-01-23,"VX","N853VA","1165",11618,"EWR","Newark, NJ","NJ","New Jersey",12892,"LAX","Los Angeles, CA","CA","California",1230,"1222",-8.00,0.00,19.00,"1506",10.00, |
| 13 | 2017,2,5,3,3,2017-05-03,"AA","N982AA","607",12889,"LAS","Las Vegas, NV","NV","Nevada",14100,"PHL","Philadelphia, PA","PA","Pennsylvania",0615,"0612",-3.00,0.00,14.00,"1334",5.00," |
| 14 | 2017,1,1,7,6,2017-01-07,"NK","N603NK","219",13930,"ORD","Chicago, IL","IL","Illinois",10397,"ATL","Atlanta, GA","GA","Georgia",1446,"1542",56.00,56.00,16.00,"1821",12.00,"1745",1 |
| 15 | 2018,1,3,22,4,2018-03-22,"AA","N156AN","2774",11298,"DFW","Dallas/Fort Worth, TX","TX","Texas",14679,"SAN","San Diego, CA","CA","California",1655,"1649",-6.00,0.00,18.00,"1753",3 |
| 16 | 2018,2,5,1,2,2018-05-01,"DL","N939DN","2465",10529,"BDL","Hartford, CT","CT","Connecticut",10397,"ATL","Atlanta, GA","GA","Georgia",1540,"1544",4.00,4.00,10.00,"1748",4.00,"1820" |
| 17 | 2017,4,10,7,6,2017-10-07,"B6","N519JB","697",10721,"BOS","Boston, MA","MA","Massachusetts",10397,"ATL","Atlanta, GA","GA","Georgia",1355,"1345",-10.00,0.00,15.00,"1615",9.00,"1644 |
| 18 | 2017,3,7,4,2,2017-07-04,"AS","N265AK","461",12266,"IAH","Houston, TX","TX","Texas",14747,"SEA","Seattle, WA","WA","Washington",1745,"1733",-12.00,0.00,12.00,"1954",16.00,"2019",2 |
| 19 | 2018,2,5,12,6,2018-05-12,"AA","N924US","1743",14107,"PHX","Phoenix, AZ","AZ","Arizona",15304,"TPA","Tampa, FL","FL","Florida",0901,"0856",-5.00,0.00,20.00,"1540",9.00,"1604",1549 |
| 20 | 2018,4,10,23,2,2018-10-23,"AA","N357PV","1524",11298,"DFW","Dallas/Fort Worth, TX","TX","Texas",14908,"SNA","Santa Ana, CA","CA","California",1105,"1103",-2.00,0.00,21.00,"1204",2 |
| 21 | 2017,2,5,13,6,2017-05-13,"WN","N8530W","4928",14679,"SAN","San Diego, CA","CA","California",10397,"ATL","Atlanta, GA","GA","Georgia",1110,"1124",14.00,14.00,12.00,"1830",7.00,"182 |
| 22 | 2018,1,1,1,1,2018-01-01,"OO","N693BR",3490,14747,"SEA","Seattle, WA","WA","Washington",11638,"FAT","Fresno, CA","CA","California",1855,"1851",-4.00,0.00,15.00,"2046",4.00,"2105" |
| 23 | 2017,3,8,17,4,2017-08-17,"WN","N7724A","1451",14122,"PIT","Pittsburgh, PA","PA","Pennsylvania",12191,"HOU","Houston, TX","TX","Texas",1130,"1132",2.00,2.00,9.00,"1315",6.00,"1330 |
| 24 | 2018,2,5,17,4,2018-05-17,"WN","N420WN","1732",13342,"MKE","Milwaukee, WI","WI","Wisconsin",12953,"LGA","New York, NY","NY","New York",0620,"0617",-3.00,0.00,18.00,"0913",11.00,"09 |
| 25 | 2019,1,1,18,5,2019-01-18,"UA","N588UA","510",12892,"LAX","Los Angeles, CA","CA","California",11618,"EWR","Newark, NJ","NJ","New Jersey",1315,"1309",-6.00,0.00,17.00,"2107",6.00,"2 |
| 26 | 2017,4,12,1,5,2017-12-01,"EV","N758EV","5393",11433,"DTW","Detroit, MI","MI","Michigan",14524,"RIC","Richmond, VA","VA","Virginia",1540,"1537",-3.00,0.00,19.00,"1766",11.00,"1732" |
| 27 | 2018,1,1,27,6,2018-01-27,"VX","N365VA","1592",14771,"SFO","San Francisco, CA","CA","California",14262,"PSB","Palm Springs, CA","CA","California",1330,0.00,0.00,14.00,"1446" |
| 28 | 2018,2,5,10,4,2018-05-10,"AS","N633VA","1756",14747,"SEA","Seattle, WA","WA","Washington",14771,"SFO","San Francisco, CA","CA","California",1855,"1859",4.00,4.00,14.00,"2047",6.00 |

BINNING:

Separated the data based on Cancellation Reason by using Binning Technique.

The screenshot shows a software interface for data processing, likely Apache Flink, with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Project Tree:** FlightDelay > OFlightCancellation > NAS-cancellation-m-00072. The tree includes nodes for Project, OAvgFlightDelay, OCancellationCount, OCounters, ODelayCount, ODelayFMedian, ODistinct, and OFlightCancellation, along with several .crc files.
- Table View:** The main area displays a table with 32 rows of data, each containing 13 columns of flight cancellation information. The columns include:
 - Date: 2017, 1, 1, 6, 5, 2017-01-06, DL, N967DL, 1198, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 11433, DTW, Detroit, MI, MI, Michigan, 1346, ..., 1651, ..., 1.00, C, 0.00, 185.00, ..., 1.
 - Flight ID: 2017, 1, 1, 6, 5, 2017-01-06, DL, N399DA, 1427, 10397, ATL, Atlanta, GA, GA, Georgia, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 2050, ..., 2248, ..., 1.00, C, 0.00, 118.00, ..., 1.0
 - Flight ID: 2017, 1, 1, 6, 5, 2017-01-06, DL, N302DN, 1465, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 10397, ATL, Atlanta, GA, GA, Georgia, 1345, ..., 1545, ..., 1.00, C, 0.00, 120.00, ..., 1.0
 - Flight ID: 2017, 1, 1, 6, 5, 2017-01-06, DL, N307DX, 1726, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 10397, ATL, Atlanta, GA, GA, Georgia, 1700, ..., 1900, ..., 1.00, C, 0.00, 120.00, ..., 1.0
 - Flight ID: 2017, 1, 1, 6, 5, 2017-01-06, DL, N830DN, 1897, 10397, ATL, Atlanta, GA, GA, Georgia, 11066, CMH, Columbus, OH, OH, Ohio, 2116, 2119, 3.00, 3.00, 2250, ..., 1.00, C, 0.00, 94.00, ..., 1.
 - Flight ID: 2017, 1, 1, 6, 5, 2017-01-06, DL, N943DL, 2065, 12478, JFK, New York, NY, NY, New York, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 1559, ..., 1925, ..., 1.00, C, 0.00, 206.00, ..., 1.
 - Flight ID: 2017, 1, 1, 6, 5, 2017-01-06, DL, N991AT, 2068, 12953, LGA, New York, NY, NY, New York, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 1700, ..., 2018, ..., 1.00, C, 0.00, 198.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N360NB, 46, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 12478, JFK, New York, NY, NY, New York, 1213, ..., 1508, ..., 1.00, C, 0.00, 175.00, ..., 1.0
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N399DA, 464, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 12478, JFK, New York, NY, NY, New York, 1500, ..., 1758, ..., 1.00, C, 0.00, 178.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N9380L, 957, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 11193, CVG, Cincinnati, OH, KY, Kentucky, 1245, ..., 1524, ..., 1.00, C, 0.00, 159.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N581NW, 1085, 11433, DTW, Detroit, MI, MI, Michigan, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 0850, ..., 1203, ..., 1.00, C, 0.00, 193.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N6704Z, 1102, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 14747, SEA, Seattle, WA, WA, Washington, 1200, ..., 1559, ..., 1.00, C, 0.00, 419.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N581NW, 1198, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 11433, DTW, Detroit, MI, MI, Michigan, 1330, ..., 1636, ..., 1.00, C, 0.00, 186.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N304DN, 1465, 10397, ATL, Atlanta, GA, GA, Georgia, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 1105, ..., 1258, ..., 1.00, C, 0.00, 113.00, ..., 1.0
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N304DN, 1465, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 10397, ATL, Atlanta, GA, GA, Georgia, 1355, ..., 1553, ..., 1.00, C, 0.00, 118.00, ..., 1.0
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N833DN, 1604, 11433, DTW, Detroit, MI, MI, Michigan, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 0720, ..., 1033, ..., 1.00, C, 0.00, 193.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N833DN, 1604, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 11433, DTW, Detroit, MI, MI, Michigan, 1135, ..., 1435, ..., 1.00, C, 0.00, 180.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N921DN, 1717, 13487, MSP, Minneapolis, MN, MN, Minnesota, 13930, ORD, Chicago, IL, IL, Illinois, 1515, ..., 1641, ..., 1.00, C, 0.00, 86.00, ..., 1.00
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N930DL, 1879, 12953, LGA, New York, NY, NY, New York, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 0700, ..., 1021, ..., 1.00, C, 0.00, 201.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N3767, 1940, 10397, ATL, Atlanta, GA, GA, Georgia, 12953, LGA, New York, NY, NY, New York, 1045, 1836, 471.00, 471.00, ..., 1257, ..., 1.00, C, 0.00, 132.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N991AT, 1970, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 10397, ATL, Atlanta, GA, GA, Georgia, 1550, ..., 1749, ..., 1.00, C, 0.00, 119.00, ..., 1.0
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N924DN, 2012, 12953, LGA, New York, NY, NY, New York, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 0840, ..., 1203, ..., 1.00, C, 0.00, 203.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N311DN, 2026, 10397, ATL, Atlanta, GA, GA, Georgia, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 1002, ..., 1155, ..., 1.00, C, 0.00, 113.00, ..., 1.0
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N311DN, 2026, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 10397, ATL, Atlanta, GA, GA, Georgia, 1256, ..., 1455, ..., 1.00, C, 0.00, 119.00, ..., 1.0
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N930DL, 2112, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 12953, LGA, New York, NY, NY, New York, 1101, ..., 1405, ..., 1.00, C, 0.00, 184.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N863DN, 2182, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 13487, MSP, Minneapolis, MN, MN, Minnesota, 1330, ..., 1631, ..., 1.00, C, 0.00, 241.00
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N863DN, 2182, 13487, MSP, Minneapolis, MN, MN, Minnesota, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 0725, ..., 1212, ..., 1.00, C, 0.00, 227.00
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N399DA, 2185, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 12478, JFK, New York, NY, NY, New York, 0730, ..., 1017, ..., 1.00, C, 0.00, 167.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N305DN, 2226, 10397, ATL, Atlanta, GA, GA, Georgia, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 0800, ..., 0946, ..., 1.00, C, 0.00, 106.00, ..., 1.0
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N305DN, 2226, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 10397, ATL, Atlanta, GA, GA, Georgia, 1045, ..., 1240, ..., 1.00, C, 0.00, 115.00, ..., 1.0
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N938DL, 2234, 11193, CVG, Cincinnati, OH, KY, Kentucky, 11697, FLL, Fort Lauderdale, FL, FL, Florida, 0915, ..., 1158, ..., 1.00, C, 0.00, 163.00, ..., 1.
 - Flight ID: 2017, 1, 1, 7, 6, 2017-01-07, DL, N822DN, 2351, 12980, LAS, Las Vegas, NV, NV, Nevada, 10397, ATL, Atlanta, GA, GA, Georgia, 1223, ..., 1015, ..., 1.00, C, 0.00, 232.00, 1.00, 1747.

PIG: TOP K

Found Top 10 Airline with Most Flights using Pig.

```
flights = LOAD 'apr17.csv' using PigStorage(',');
grouped = GROUP flights BY $6;
summed = FOREACH grouped GENERATE group, COUNT(flights) AS cntd;
sorted = ORDER summed BY cntd DESC;
top25 = LIMIT sorted 10;
Dump top25;
STORE top25 into 'top10Carriers';
```

| WN | 112223 |
|----|--------|
| DL | 77046 |
| AA | 73656 |
| OO | 56210 |
| UA | 45541 |
| EV | 30789 |
| B6 | 25191 |
| AS | 15261 |
| NK | 12544 |
| F9 | 7801 |

PIG: JOIN

Joined Airline csv and unique Airline names using Pig.

```
uniqueid = LOAD 'distinct' using PigStorage(',') as (uid);
carriers = LOAD 'carriers.csv' using PigStorage(',') as (id,name);
joined = JOIN uniqueid BY uid, carriers BY id;
summed = FOREACH joined GENERATE $0,$2;
STORE summed into 'joinoutput';
```

```
"9E"      "Pinnacle Airlines Inc."
"AA"      "American Airlines Inc."
"AS"      "Alaska Airlines Inc."
"B6"      "JetBlue Airways"
"DL"      "Delta Air Lines Inc."
"EV"      "Atlantic Southeast Airlines"
"F9"      "Frontier Airlines Inc."
"G4"      "Allegiant Air"
"HA"      "Hawaiian Airlines Inc."
"MQ"      "American Eagle Airlines Inc."
"NK"      "Spirit Air Lines"
"OH"      "Comair Inc."
"OO"      "Skywest Airlines Inc."
"UA"      "United Air Lines Inc."
"VX"      "Virgin America"
"WN"      "Southwest Airlines Co."
"YV"      "Mesa Airlines Inc."
"YX"      "Midwest Airline
```

PIG FILTER AND MERGE:

Filtered the Latest Data based on Year using Pig.

```
flights = LOAD 'Merge.csv' using PigStorage(',');
filter1 = FILTER flights BY $0 > 2017;
filter2 = FOREACH filter1 GENERATE $0,$1,$2,$3,$4,$5,$6,$7,$8,$9, 'LatestData';
STORE filter2 into 'LatestData';
```

part-00000 /usr/local/pig-0.17.0/bin/Late

| | | | | | | | | | | |
|------|---|---|---|---|------------|----|--------|------|-------|-------------|
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N201FR | 1680 | 12889 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N201FR | 681 | 13204 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N201FR | 681 | 11292 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N201FR | 144 | 14747 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N201FR | 122 | 11292 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N202FR | 1138 | 12889 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N202FR | 1141 | 10423 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N202FR | 1106 | 12889 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N202FR | 1111 | 12266 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N203FR | 1236 | 13487 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N203FR | 1201 | 15304 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N203FR | 1204 | 12339 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N203FR | 1209 | 15304 | Latest Data |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N203FR | 1099 | 13342 | Latest Data |

AWS CLOUD

Used AWS Cloud EMR instance for running MapReduce Jobs.

The screenshot shows the AWS EMR Cluster details page for 'FlightCluster'. The cluster status is 'Waiting'. The 'Steps' tab is selected, showing one step named 'Custom JAR' with ID 's-2KLGKR4N9VF0Z'. The step status is 'Completed', started at 2019-04-24 19:45 (UTC-4), and took 4 minutes. The log files link is 'View logs'. The step details include:

- JAR location: s3://flightmapreduce/jar/FlightDelay.jar
- Main class: None
- Arguments: AirlineCancel7.Driver s3://flightmapreduce/input s3://flightmapreduce/output/file
- Action on failure: Continue

The screenshot shows the AWS S3 object details page for 'job_1556143697838_0001-1556149535584-hadoop-FlightDelay.jar'. The object was last modified on Apr 24, 2019, at 7:58:17 PM GMT-0400. The Etag is ee493fe385f7a1d2785881c5bcd44eb4. The storage class is Standard. The server-side encryption is AES-256. The size is 21.2 KB. The key is logs/_266838CXTTD1/hadoop-mapreduce/history/2019/04/24/000000/job_1556143697838_0001-1556149535584-hadoop-FlightDelay.jar-1556149810763-60-1-SUCCEEDED-default-1556149546461.jhist.gz. The object URL is https://s3.amazonaws.com/flightmapreduce/logs/_266838CXTTD1/hadoop-mapreduce/history/2019/04/24/000000/job_1556143697838_0001-1556149535584-hadoop-FlightDelay.jar-1556149810763-60-1-SUCCEEDED-default-1556149546461.jhist.gz.

The screenshot shows a terminal window displaying the contents of a file named 'parc-r-00000'. The file contains a large amount of data, likely a histogram or summary of flight delay statistics, structured in a grid format. The columns represent categories like '9E', 'A', 'B', and 'C', and the rows represent numerical values ranging from 1 to 13. The file is located in the directory '/Downloads'.

| 9E | A | 1 | 13 | |
|----|---|----|------|--|
| 9E | A | 10 | 12 | |
| 9E | A | 12 | 23 | |
| 9E | A | 3 | 494 | |
| 9E | A | 5 | 8 | |
| 9E | A | 9 | 7 | |
| 9E | B | 1 | 364 | |
| 9E | B | 10 | 106 | |
| 9E | B | 12 | 28 | |
| 9E | B | 3 | 2086 | |
| 9E | B | 5 | 356 | |
| 9E | B | 7 | 12 | |
| 9E | B | 9 | 36 | |
| 9E | C | 1 | 1077 | |
| 9E | C | 10 | 60 | |
| 9E | C | 12 | 65 | |
| 9E | C | 3 | 486 | |
| 9E | C | 5 | 84 | |
| 9E | C | 7 | 1302 | |
| 9E | C | 9 | 109 | |
| AA | A | 1 | 1099 | |
| AA | A | 10 | 642 | |
| AA | A | 12 | 553 | |
| AA | A | 3 | 621 | |
| AA | A | 4 | 274 | |
| AA | A | 5 | 1180 | |
| AA | A | 7 | 2335 | |
| AA | A | 8 | 609 | |
| AA | A | 9 | 33 | |
| AA | B | 1 | 3671 | |
| AA | B | 10 | 1168 | |
| AA | B | 12 | 1132 | |
| AA | B | 3 | 5525 | |
| AA | B | 4 | 356 | |
| AA | B | 5 | 1322 | |
| AA | B | 7 | 2290 | |
| AA | B | 8 | 974 | |
| AA | B | 9 | 1021 | |
| AA | C | 1 | 214 | |
| AA | C | 10 | 266 | |
| AA | C | 12 | 13 | |
| AA | C | 3 | 57 | |

TECHNOLOGIES AND PATTERNS USED

- Max Min
- Count
- Average
- Median
- Memory Conscious Median
- Inverted Index
- Counting with Counters
- Input Format
- Custom Combiner
- Partitioner

- Bloom Filter
- Top K
- Distinct
- Joins
- Shuffling
- Binning
- Chaining
- AWS Cloud
- PIG Top K
- PIG Filter and Merge
- PIG Joins

Appendix

Code Snippet:

1. Airline Cancellation:

```
package AirlineCancel7;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

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

        job.setMapperClass(CancelCountMapper.class);
        job.setCombinerClass(CancelCountReducer.class);
        job.setReducerClass(CancelCountReducer.class);

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

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

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

job.setNumReduceTasks(1);

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

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

// Submit the job, then poll for progress until the job is complete
job.waitForCompletion(true);
}

}

package AirlineCancel7;

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

import java.io.IOException;
```

```
public class CancelCountMapper extends Mapper<LongWritable, Text, Text, Text> {  
    private static final IntWritable one = new IntWritable(1);  
  
    @Override  
    protected void map(LongWritable key, Text value, Context context)  
        throws IOException, InterruptedException {  
        String line = value.toString();  
        String t= "";  
        String[] tokens = line.split(",");  
        // for(String s : tokens) {  
        if(!tokens[0].contains("YEAR") && tokens.length >33 && !tokens[32].equalsIgnoreCase("0")  
            && !tokens[33].equalsIgnoreCase(" ") &&  
            !tokens[33].replace("\n", "").trim().equalsIgnoreCase("")) {  
  
            context.write(new  
                Text(tokens[6].replace("\n", "").trim()+"\t"+tokens[33].replace("\n", "").trim()+"\t"+tokens[2]), new  
                Text(1+""));  
        }  
        //}  
    }  
  
    @Override  
    protected void cleanup(Context context)  
        throws IOException, InterruptedException {  
        // TODO Auto-generated method stub  
        super.cleanup(context);  
    }  
  
    @Override
```

```

public void run(Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.run(context);
}

@Override
protected void setup(Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.setup(context);
}

}

```

2. Airline Cancellation:

```

package AirlineCancel7;

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

import java.io.IOException;

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

    @Override
    protected void cleanup(Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.cleanup(context);
}

```

```
}

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

    int sum = 0;
    int count=0;
    for (Text value : values) {
        String a[]={value.toString().split(" ");}
        // sum += Float.parseFloat(a[0]);
        count+=Integer.parseInt(a[0]);
    }

    context.write(key, new Text(count+""));
}

@Override
public void run(Context arg0)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.run(arg0);
}

@Override
protected void setup(Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.setup(context);
```

```
}
```

```
}
```

3. Airline Delay:

```
package AirlineDelay7;
```

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

import java.io.IOException;

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

        job.setMapperClass(DelayCountMapper.class);
        job.setCombinerClass(DelayCountReducer.class);
        job.setReducerClass(DelayCountReducer.class);

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

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

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

job.setNumReduceTasks(1);

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

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

// Submit the job, then poll for progress until the job is complete
job.waitForCompletion(true);
}

}

package AirlineDelay7;

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

import java.io.IOException;
```

```
public class DelayCountMapper extends Mapper<LongWritable, Text, Text, Text> {  
    private static final IntWritable one = new IntWritable(1);  
  
    @Override  
    protected void map(LongWritable key, Text value, Context context)  
        throws IOException, InterruptedException {  
        String line = value.toString();  
        String t= "";  
        String[] tokens = line.split(",");  
        Integer delay=0;  
        // for(String s : tokens) {  
        if(!tokens[0].contains("YEAR") && tokens.length >45 && !tokens[41].equalsIgnoreCase("")  
            && !tokens[41].equalsIgnoreCase(" ") && !tokens[41].equalsIgnoreCase("NA")) {  
            if(Float.parseFloat(tokens[41]) >0 ){  
                t="CARRIER_DELAY";  
                delay=(int)Float.parseFloat(tokens[41]);  
            }  
            else if(Float.parseFloat(tokens[42]) >0 ){  
                t="WEATHER_DELAY";  
                delay=(int)Float.parseFloat(tokens[42]);  
            }  
            else if(Float.parseFloat(tokens[43]) >0 ){  
                t="NAS_DELAY";  
                delay=(int)Float.parseFloat(tokens[43]);  
            }  
            else if (Float.parseFloat(tokens[44]) >0 ){  
                t="SECURITY_DELAY";  
                delay=(int)Float.parseFloat(tokens[44]);  
            }  
        }  
    }  
}
```

```
        }

    else{
        t="LATE_AIRCRAFT_DELAY";
        delay=(int)Float.parseFloat(tokens[45]);
    }

    context.write(new Text(tokens[6].replace("", "")+"\t"+t+"\t"+tokens[2]), new
Text(delay.toString()+"\t"+1));
}

//}

}
```

```
@Override
protected void cleanup(Context context)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.cleanup(context);
}
```

```
@Override
public void run(Context context)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.run(context);
}
```

```
@Override
protected void setup(Context context)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
```

```
super.setup(context);
```

```
}
```

```
}
```

4. Airline delay State:

```
package AirlineDelay7;
```

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

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

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

```
public class DelayCountReducer extends Reducer<Text, Text, Text, Text> {
```

```
    @Override
```

```
    protected void cleanup(Context context)
```

```
        throws IOException, InterruptedException {
```

```
        // TODO Auto-generated method stub
```

```
        super.cleanup(context);
```

```
}
```

```
    @Override
```

```
    protected void reduce(Text key, Iterable<Text> values,
```

```
        Context context) throws IOException, InterruptedException {
```

```
        int sum = 0;
```

```
        int count=0;
```

```
        for (Text value : values) {
```

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

```

        sum += Float.parseFloat(a[0]);
        count+=Integer.parseInt(a[1]);

    }

    context.write(key, new Text(sum+"\t"+count));

}

@Override
public void run(Context arg0)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.run(arg0);

}

@Override
protected void setup(Context context)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.setup(context);

}

}

```

5. Avg Flight Delay:

```

package AverageFlightDelay;

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

```

```
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
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.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

public class AvgFlightDelay {
    public static class MapDelay extends Mapper<LongWritable, Text, Text, DelayAveragingPair> {
        private Text outText = new Text();
        private Map<String,DelayAveragingPair> pairMap = new HashMap<String,DelayAveragingPair>();
        private DelayAveragingPair pair = new DelayAveragingPair();

        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
            String line = value.toString();
            String[] columnValues = line.split(",");
            if(columnValues.length > 24) {
                if (!columnValues[22].contains("DEP_DELAY") && !columnValues[24].contains("TAXI_OUT") &&
                !columnValues[24].equalsIgnoreCase("NA") && columnValues[24] != null &&
                !columnValues[24].equalsIgnoreCase("") ) {
                    outText.set(columnValues[6]);
                }
            }
        }
    }
}
```

```

int delayTime = (int) Float.parseFloat(columnValues[24]);
DelayAveragingPair pair = pairMap.get(columnValues[6]);
if (pair == null) {
    pair = new DelayAveragingPair();
    pairMap.put(columnValues[6], pair);
}
int delay = pair.getDelay().get() + delayTime;
int count = pair.getCount().get() + 1;
pair.set(delay, count);
}

}

@Override
protected void cleanup(Context context) throws IOException, InterruptedException {
    Set<String> keys = pairMap.keySet();
    Text keyText = new Text();
    for (String key : keys) {
        keyText.set(key.replace("\r\n", "\n").trim());
        context.write(keyText, pairMap.get(key));
    }
}

public static class Reduce extends Reducer<Text, DelayAveragingPair, Text, IntWritable> {
    public void reduce(Text key, Iterable<DelayAveragingPair> values
                      , Context context)
        throws IOException, InterruptedException {
        int sum = 0;

```

```

int count = 0;
for (DelayAveragingPair val : values) {
    sum += val.getDelay().get();
    count += val.getCount().get();
}
context.write(key, new IntWritable(sum/count));
}
}

```

6. Flight Average Delay

```

public static class DelayAveragingPair implements Writable,
WritableComparable<DelayAveragingPair> {

    private IntWritable delay;
    private IntWritable count;

    public DelayAveragingPair() {
        set(new IntWritable(0), new IntWritable(0));
    }

    public DelayAveragingPair(int delay, int count) {
        set(new IntWritable(delay), new IntWritable(count));
    }

    public void set(int delay, int count){
        this.delay.set(delay);
        this.count.set(count);
    }

    public void set(IntWritable delay, IntWritable count) {
        this.delay = delay;
    }
}

```

```
    this.count = count;
}

public void write(DataOutput out) throws IOException {
    delay.write(out);
    count.write(out);
}

public void readFields(DataInput in) throws IOException {
    delay.readFields(in);
    count.readFields(in);
}

public int compareTo(DelayAveragingPair other) {
    int compareVal = this.delay.compareTo(other.getDelay());
    if (compareVal != 0) {
        return compareVal;
    }
    return this.count.compareTo(other.getCount());
}

public static DelayAveragingPair read(DataInput in) throws IOException {
    DelayAveragingPair averagingPair = new DelayAveragingPair();
    averagingPair.readFields(in);
    return averagingPair;
}
```

```
@Override  
public boolean equals(Object o) {  
    if (this == o) return true;  
    if (o == null || getClass() != o.getClass()) return false;  
    DelayAveragingPair that = (DelayAveragingPair) o;  
  
    if (!count.equals(that.count)) return false;  
    if (!delay.equals(that.delay)) return false;  
  
    return true;  
}
```

```
@Override  
public int hashCode() {  
    int result = delay.hashCode();  
    result = 163 * result + count.hashCode();  
    return result;  
}
```

```
@Override  
public String toString() {  
    return "DelayAveragingPair{" +  
        "delay=" + delay +  
        ", count=" + count +  
        '}';  
}
```

```
public IntWritable getDelay() {
```

```
    return delay;
}

public IntWritable getCount() {
    return count;
}

}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

    Job job = new Job(conf, "flight count");
    job.setJarByClass(AvgFlightDelay.class);

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

    job.setMapperClass(MapDelay.class);
    job.setReducerClass(Reduce.class);

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

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

    job.waitForCompletion(true);
}
```

7. Average Delay by Month

```
package AvgDelayMonthly;

import au.com.bytecode.opencsv.CSVParser;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

import java.io.IOException;

public class AverageFlightDelayByMonth {
    public static class AverageFlightDelayByMonthMapper extends
        Mapper<Object, Text, AirlineTextPair, Text> {

        // initialize CSVParser as comma separated values
        private CSVParser csvParser = new CSVParser(',', "'");

        /**
         * Key : Byte offset from where the line is being read
         * value : string representing the entire line of flight data
    }
}
```

```

        */

    public void map(Object offset, Text value, Context context)
        throws IOException, InterruptedException {

        // Parse the input line
        String[] line = this.csvParser.parseLine(value.toString());
        AirlineTextPair key = new AirlineTextPair();
        Text arrDelayMinutes = new Text();

        if (line.length > 0 && isValidEntry(line) && !line[2].contains("MONTH") &&
        !line[24].equalsIgnoreCase("")) {
            // set key as AirlineTextPair
            key.setAirLineName(line[6].replace("\",\"", "").trim());
            key.setMonth(line[2]);

            // set value as arrDelayMinutes
            arrDelayMinutes.set(line[24]);

            context.write(key, arrDelayMinutes);
        }
    }

    /**
     * Function determines the validity of the input record
     *
     * @param record : line record representing flight details
     * @return true iff entry contain all required data fields
     *         false otherwise
     */

```

```
private boolean isValidEntry(String[] record) {  
  
    if (record == null || record.length == 0) {  
        return false;  
    }  
  
    // If any of required field is missing, we'll ignore the record  
    if (record[0].equalsIgnoreCase("") ) {  
        return false;  
    }  
  
    //  
    if(! record[0].equals("2008")){  
        //  
        return false;  
    }  
  
    // Whether flight was cancelled  
    if (record[32].equals("1")) {  
        return false;  
    }  
  
    return true;  
}  
  
/**  
 * Reduce class  
 */  
public static class AverageFlightDelayByMonthReducer extends  
    Reducer<AirlineTextPair, Text, Text, Text> {
```

```

/**
 * Reduce function will be called per Airline's Carrier Name and
 * months will be in increasing sorted order
 * Key : AirlineTextPair which implements WritableComparable
 * values : sorted delay in increasing order by month for given key airline
 */

public void reduce(AirlineTextPair key, Iterable<Text> values,
                    Context context) throws IOException, InterruptedException {
    double totalDelay = 0;
    int totalFligths = 0;
    int currMonth = 1;
    StringBuilder output = new StringBuilder();
    output.append(key.getAirLineName().toString());

    for(Text value : values){
        // Check whether month has changed for the given flight
        // On change append data to output string and clear the counters
        if(currMonth != Integer.parseInt(key.getMonth().toString())){
            int avgDelay = (int)Math.ceil(totalDelay/(double)totalFligths);
            output.append(", (" + currMonth + "," + avgDelay + ")");
            totalDelay = 0;
            totalFligths = 0;
            currMonth = Integer.parseInt(key.getMonth().toString());
        }
        totalDelay += Double.parseDouble(value.toString());
        totalFligths++;
    }
}

```

```

// Append last valid data of the current flight

int avgDelay = (int)Math.ceil(totalDelay/(double)totalFligths);
output.append(", (" + currMonth + "," + avgDelay + ")");

// Write down remaining months as 0 delay
while(currMonth < 12)
{
    currMonth++;
    output.append(", (" + currMonth + ", 0 )");
}

// Output the formatted string in the file
context.write(new Text(), new Text(output.toString()));

}

/**
 * AirlinePartitioner will partition data based on the hashCode of the
 * Airline's UniqueCarrier
 */
public static class AirlinePartitioner extends
    Partitioner<AirlineTextPair, Text> {
    /**
     * Based on the configured number of reducer, this will
     * partition the data approximately evenly based on number of
     * unique carrier names
     */
    @Override

```

```

        public int getPartition(AirlineTextPair key, Text value,
                               int numPartitions) {
            // multiply by 127 to perform some mixing
            return Math.abs(key.hashCode() * 127) % numPartitions;
    }

}

/***
 * KeyComparator first sorts based on the unique carrier name and
 * then sorts months in increasing order using compareTo method of the
 * AirlineTextPair class
 */
public static class KeyComparator extends WritableComparator {
    protected KeyComparator() {
        super(AirlineTextPair.class, true);
    }

    @Override
    public int compare(WritableComparable w1, WritableComparable w2) {
        AirlineTextPair airline1 = (AirlineTextPair) w1;
        AirlineTextPair airline2 = (AirlineTextPair) w2;
        return airline1.compareTo(airline2);
    }
}

/***
 * GroupComparator ignores the month files of the key so that
 * single reduce function call receives all flight data for given
 * unique carrier name
*/

```

```

        */

public static class GroupComparator extends WritableComparator {
    protected GroupComparator() {
        super(AirlineTextPair.class, true);
    }

    @Override
    public int compare(WritableComparable w1, WritableComparable w2) {
        AirlineTextPair airline1 = (AirlineTextPair) w1;
        AirlineTextPair airline2 = (AirlineTextPair) w2;
        return airline1.compare(airline2);
    }
}

/**
 * @param args
 */
public static void main(String[] args) throws Exception {

    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args)
        .getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Usage: AverageFlightDelayByMonth <in> <out>");
        System.exit(2);
    }
    Job job = new Job(conf, "Average monthly flight delay");
    job.setJarByClass(AverageFlightDelayByMonth.class);
    job.setMapperClass(AverageFlightDelayByMonthMapper.class);
}

```

```

        job.setReducerClass(AverageFlightDelayByMonthReducer.class);
        job.setOutputKeyClass(AirlineTextPair.class);
        job.setOutputValueClass(Text.class);
        job.setPartitionerClass(AirlinePartitioner.class);
        job.setSortComparatorClass(KeyComparator.class);
        job.setGroupingComparatorClass(GroupComparator.class);
        FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }

}

package AvgDelayMonthly;

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

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

/**
 * AirlineTextPair implements the WritableComparable
 * This class contains useful functions used for keyComparator and
 * Grouping comparator
 */
public class AirlineTextPair implements WritableComparable {
    // Airline's UniqueCarrier
}

```

```
Text airLineName;  
// Airlines fly Month  
Text month;  
  
/**  
 * constructor  
 */  
public AirlineTextPair() {  
    this.airLineName = new Text();  
    this.month = new Text();  
}  
  
/**  
 * constructor  
 * @param airLineName  
 * @param month  
 */  
public AirlineTextPair(Text airLineName, Text month) {  
    this.airLineName = airLineName;  
    this.month = month;  
}  
  
/**  
 * set airline's unique carrier name  
 * @param airLineName  
 */  
public void setAirLineName(String airLineName) {  
    this.airLineName.set(airLineName.getBytes());  
}
```

```
/**  
 * set the month of the flight fly  
 * @param month  
 */  
  
public void setMonth(String month) {  
    this.month.set(month.getBytes());  
}  
  
/**  
 * get airline's unique carrier name  
 */  
  
public Text getAirLineName() {  
    return this.airLineName;  
}  
  
/**  
 * get the month of the flight fly  
 */  
  
public Text getMonth() {  
    return this.month;  
}  
  
/**  
 * overrider the write method to support write operation  
 */  
  
public void write(DataOutput out) throws IOException {  
    this.airLineName.write(out);  
    this.month.write(out);
```

```
}

/**
 * overrider readFiled method to support reading fields
 */
public void readFields(DataInput in) throws IOException {
    if (this.airLineName == null)
        this.airLineName = new Text();

    if (this.month == null)
        this.month = new Text();

    this.airLineName.readFields(in);
    this.month.readFields(in);
}

/**
 * Sort first by airline name and then by month in increasing order
 */
public int compareTo(Object object) {
    AirlineTextPair ip2 = (AirlineTextPair) object;
    int cmp = getAirLineName().compareTo(ip2.getAirLineName());
    if (cmp != 0) {
        return cmp;
    }
    int m1 = Integer.parseInt(getMonth().toString());
    int m2 = Integer.parseInt(ip2.getMonth().toString());
    return m1 == m2 ? 0 : (m1 < m2 ? -1 : 1);
}
```

```

/**
 * provide comparator for airline name for grouping comparator
 */
public int compare(Object object){
    AirlineTextPair airline2 = (AirlineTextPair) object;
    return getAirLineName().compareTo(airline2.getAirLineName());
}

/**
 * use this hashCode for partitioning
 */
public int hashCode() {
    return this.airLineName.hashCode();
}

public boolean equals(Object o) {
    AirlineTextPair p = (AirlineTextPair) o;
    return this.airLineName.equals(p.getAirLineName());
}

public Text toText(){
    return new Text("(" + this.airLineName.toString() + "," + this.month.toString() + ")");
}
}

8. Binning
package BinningFlight;

```

```
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.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.MultipleOutputs;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

public class FlightCancellation
{

    public static class TMapper extends Mapper<Object, Text, Text, NullWritable>
    {

        private MultipleOutputs<Text, NullWritable> mos=null;

        @Override
        protected void setup(Context context) throws IOException, InterruptedException
        {
            mos = new MultipleOutputs(context);
        }

        @Override
```

```

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

    String row[] = value.toString().split(",");
    if(!row[32].equalsIgnoreCase("0") && !row[33].equalsIgnoreCase("\\"\\\"")){
        String cancellationCode = row[33];
        // System.out.println(cancellationCode);
        if(cancellationCode.contains("A"))
            mos.write("bins", value, NullWritable.get(),"Carrier-cancellation");
        if(cancellationCode.contains("B"))
            mos.write("bins", value, NullWritable.get(),"Weather-cancellation");
        if(cancellationCode.contains("C"))
            mos.write("bins", value, NullWritable.get(),"NAS-cancellation");
        if(cancellationCode.contains("D"))
            mos.write("bins", value, NullWritable.get(),"Security-cancellation");
    }
}

@Override
protected void cleanup(Context context) throws IOException, InterruptedException
{
    mos.close();
}

```

```
}

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

    MultipleOutputs.addNamedOutput(job, "bins", TextOutputFormat.class, Text.class,
IntWritable.class);

    MultipleOutputs.setCountersEnabled(job, true);
    job.setNumReduceTasks(0);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(NullWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}

}
```

9. Flight Cancellation Count

```
package CancellationCount2;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

public class Driver {

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

        job.setMapperClass(CancellationCountMapper.class);
        job.setCombinerClass(CancellationCountReducer.class);
        job.setReducerClass(CancellationCountReducer.class);

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

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

job.setNumReduceTasks(1);

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

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

// Submit the job, then poll for progress until the job is complete
job.waitForCompletion(true);
}

}

package CancellationCount2;

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

import java.io.IOException;

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

```
private static final IntWritable one = new IntWritable(1);

@Override
protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
IntWritable>.Context context)
    throws IOException, InterruptedException {
    String line = value.toString();

    String[] tokens = line.split(",");
    // for(String s : tokens) {
        if(!tokens[0].equalsIgnoreCase("YEAR") && !tokens[32].equalsIgnoreCase("0") &&
!tokens[33].equalsIgnoreCase(""))
            context.write(new Text(tokens[33].replace("\\"","").trim()), one);
    //}
}

@Override
protected void cleanup(Mapper<LongWritable, Text, Text, IntWritable>.Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.cleanup(context);
}

@Override
public void run(Mapper<LongWritable, Text, Text, IntWritable>.Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.run(context);
```

```
}

@Override
protected void setup(Mapper<LongWritable, Text, Text, IntWritable>.Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.setup(context);
}

}

package CancellationCount2;

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

import java.io.IOException;

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

    @Override
    protected void cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.cleanup(context);
    }

    @Override
```

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

    int sum = 0;

    for (IntWritable value : values) {
        sum += value.get();
    }

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

@Override
public void run(Reducer<Text, IntWritable, Text, IntWritable>.Context arg0)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.run(arg0);
}

@Override
protected void setup(Reducer<Text, IntWritable, Text, IntWritable>.Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.setup(context);
}

}
```

10. Counting with Counters

```
package CountersDistanceGroup;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.mapreduce.Counter;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class Driver {

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

    {
        Job job = Job.getInstance();

        job.setJarByClass(Driver.class);

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

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

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

int code = job.waitForCompletion(true)? 1 : 0;
if (code==1) {
    System.out.print("the counter output:\n");
    for(Counter counter: job.get_counters().getGroup("DistanceCounter")) {
        //
        System.out.println(counter.getDisplayName()+"\t"+counter.getValue());
        // System.out.println("the counter values is "+counter.getValue());
    }
}

}

}

}

package CountersDistanceGroup;

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

import java.io.IOException;
import java.util.Arrays;
```

```
import java.util.HashSet;

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

    @Override

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

        String distances[] = {"1","2","3","4","5","6","7", "8","9","10","11","12"};

        HashSet<String> distanceHashSet= new HashSet<String>(Arrays.asList(distances));

        String input = value.toString();

        String splitString[] = input.split(",");

        if(splitString.length > 39 && !value.toString().contains("YEAR")) {

            String distanceGroup = splitString[40].replace("\\" , "").trim();

            // System.out.print(month);

            if (distanceHashSet.contains(distanceGroup) && splitString[40] != "grade") {

                // System.out.print(month);

                context.getCounter("DistanceCounter", distanceGroup).increment(1);

            } else {

                context.getCounter("DistanceCounter", "unknown").increment(1);

            }

        }

    }

}
```

11. Delay Count based on reason

```
package DelayCount3;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

public class Driver {

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

        job.setMapperClass(DelayCountMapper.class);
        job.setCombinerClass(DelayCountReducer.class);
        job.setReducerClass(DelayCountReducer.class);

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

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

job.setNumReduceTasks(1);

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

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

// Submit the job, then poll for progress until the job is complete
job.waitForCompletion(true);
}

}

package DelayCount3;

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

import java.io.IOException;

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

```
private static final IntWritable one = new IntWritable(1);

@Override
protected void map(LongWritable key, Text value, Context context)
    throws IOException, InterruptedException {
    String line = value.toString();
    String t= "";
    String[] tokens = line.split(",");
    Integer delay=0;
    // for(String s : tokens) {
    if(!tokens[0].contains("YEAR") && tokens.length >45 && !tokens[41].equalsIgnoreCase("") )
        && !tokens[41].equalsIgnoreCase(" ") && !tokens[41].equalsIgnoreCase("NA")) {
        if(Float.parseFloat(tokens[41]) >0 ){
            t="CARRIER_DELAY";
            delay=(int)Float.parseFloat(tokens[41]);
        }
        else if(Float.parseFloat(tokens[42]) >0 ){
            t="WEATHER_DELAY";
            delay=(int)Float.parseFloat(tokens[42]);
        }
        else if(Float.parseFloat(tokens[43]) >0 ){
            t="NAS_DELAY";
            delay=(int)Float.parseFloat(tokens[43]);
        }
        else if (Float.parseFloat(tokens[44]) >0 ){
            t="SECURITY_DELAY";
            delay=(int)Float.parseFloat(tokens[44]);
        }
    else{

```

```
t="LATE_AIRCRAFT_DELAY";
delay=(int)Float.parseFloat(tokens[45]);
}

    context.write(new Text(t+" "+tokens[2]), new Text(delay.toString()+" "+1));
}

//}

}

@Override
protected void cleanup(Context context)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.cleanup(context);
}

@Override
public void run(Context context)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.run(context);
}

@Override
protected void setup(Context context)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.setup(context);
}
```

```
}
```

```
package DelayCount3;

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

import java.io.IOException;

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

    @Override
    protected void cleanup(Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.cleanup(context);
    }

    @Override
    protected void reduce(Text key, Iterable<Text> values,
        Context context) throws IOException, InterruptedException {
        int sum = 0;
        int count=0;
        for (Text value : values) {
            String a[]=value.toString().split(" ");
            sum += Float.parseFloat(a[0]);
            count+=Integer.parseInt(a[1]);
        }
    }
}
```

```

    }

    context.write(key, new Text(sum+" "+count));

}

@Override
public void run(Context arg0)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.run(arg0);

}

@Override
protected void setup(Context context)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.setup(context);

}

}

```

12. Delay Median using ArrayList

```

package DelayFMedian;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.SortedMapWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.lib.IdentityReducer;

```

```
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 DelayFMedian.FilterMapper;

import java.io.IOException;

public class Driver {
    public static void main( String[] args ) throws IOException, ClassNotFoundException, InterruptedException
    {
        Job job = Job.getInstance();

        job.setJarByClass(Driver.class);

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

        job.setMapperClass(FilterMapper.class);
        job.setReducerClass(FilterReducer.class);
        // job.setCombinerClass(myCombiner.class);

        //job.setOutputFormatClass(TextOutputFormat.class);
        // job.setMapOutputValueClass(SortedMapWritable.class);
        // job.setMapOutputKeyClass(Text.class);
```

```
//    job.setInputFormatClass(TextInputFormat.class);
//    job.setOutputKeyClass(Text.class);
//    job.setOutputValueClass(MedianStdDevTuple.class);

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

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

job.setNumReduceTasks(1);

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

FileSystem fs = FileSystem.get(job.getConfiguration());

job.waitForCompletion(true);

}

}

package DelayFMedian;

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

import java.io.IOException;

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

```

@Override
protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
    try {
        String split[]=value.toString().split(",");
        // IntWritable one = new IntWritable(1);
        String symbol = split[0];
        if(!symbol.contains("YEAR") && split.length>20 && !split[31].equalsIgnoreCase("0") &&
!split[31].equalsIgnoreCase("") && !split[31].equalsIgnoreCase(""));
            context.write(new Text(split[6]),new Text(split[31]));
    }
}

} catch(Exception ex) {
    System.out.println("Exception in mapper"+ex);
}
}

}

package DelayFMedian;

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

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

public class FilterReducer extends Reducer<Text, Text, Text, Text> {
    ArrayList<Integer> array= new ArrayList<Integer>();

```

```

@Override
protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
    Integer median=0;
    //int sum = 0;
    //int count = 0;

    //Integer temp= (int) Float.parseFloat(key.toString());
    try {
        for (Text value:values) {

            Integer temp = (int) Float.parseFloat(value.toString());
            if (temp != 0){
                array.add(temp);
            }
            //context.write(new Text(key), NullWritable.get());
        }
        Collections.sort(array);
        int len=array.size();
        if(array.size()%2==0){
            // median= array.get((int) len/2 - 1)+array.get((int) len/2);
            median= (array.get((int) len/2 - 1)+array.get((int) len/2)) /2;
        }
        else{
            median= array.get(len/2);
        }

        context.write(key,new Text(median+""));

    }
}

```

```

    }

    catch (Exception e){

        System.out.println(key+e.getMessage());

    }

}

}

```

13. Find Max and Min distance

```

package DistanceMax;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class Driver {

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

        Job job = Job.getInstance();

        job.setJarByClass(Driver.class);

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

        Path outDir = new Path(args[1]);
    }
}

```

```
    FileOutputFormat.setOutputPath(job, outDir);

    job.setMapperClass(MyMapper.class);
    job.setReducerClass(MyReducer.class);
    job.setCombinerClass(MyReducer.class);

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

    job.setNumReduceTasks(1);

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

    job.waitForCompletion(true);
}

}

package DistanceMax;

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

import java.io.IOException;

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

```
    @Override  
  
    protected void map(LongWritable key, Text value, Context context) throws IOException,  
    InterruptedException {  
  
        Text symbol;  
  
        String line = value.toString();  
  
        String[] fields = line.split(",");  
  
        if (fields.length > 20) {  
  
            if (!line.contains("YEAR")) {  
  
                symbol = new Text(fields[6].replace("\\"", "").trim());  
  
                context.write(symbol, new Text(fields[39]+"\t"+fields[39]));  
  
            }  
  
        }  
  
    }  
  
}
```

```
package DistanceMax;  
  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Reducer;  
  
import java.io.IOException;  
  
public class MyReducer extends Reducer<Text,Text,Text,Text> {  
  
    @Override  
  
    protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException,  
    InterruptedException {  
  
        Integer currentMaxRateValue;  
  
        Integer currentMinRateValue;
```

```

Integer maxRateValue = 0;
Integer minRateValue = Integer.MAX_VALUE;

for (Text value : values) {
    String a[] = value.toString().split("\t");
    currentMaxRateValue = (int) Float.parseFloat(a[1].trim().replace("\"", "")).trim());
    currentMinRateValue = (int) Float.parseFloat(a[0].trim().replace("\"", "")).trim());

    if (currentMaxRateValue > maxRateValue) {
        maxRateValue = currentMaxRateValue;
    }
    if (currentMinRateValue < minRateValue) {
        minRateValue = currentMinRateValue;
    }
}
context.write(key, new Text(minRateValue+"\t"+maxRateValue));
}
}

```

14. Distinct:

```

package Distinct;

import SecSortPartitionerWritable.*;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;

```

```
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.NLineInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

public class Driver
{
    public static void main( String[] args ) throws IOException, ClassNotFoundException,
InterruptedException
    {
        //    Job job = Job.getInstance();
        //
        //    job.setJarByClass(Driver.class);
        //
        //    job.setGroupingComparatorClass(GroupComparator.class);
        //    job.setSortComparatorClass(SecodarySortComparator.class);
        //    job.setPartitionerClass(KeyPartition.class);
        //
        //    FileInputFormat.addInputPath(job, new Path(args[0]));
        //    Path outDir = new Path(args[1]);
        //    FileOutputFormat.setOutputPath(job, outDir);
        //
        //    job.setMapperClass(MyMapper.class);
        //    job.setReducerClass(MyReducer.class);
        //
        //    job.setNumReduceTasks(1);
    }
}
```

```
//  
//    job.setOutputKeyClass(CompositeKeyWritable.class);  
//    job.setOutputValueClass(NullWritable.class);  
  
//  
//    FileSystem fs = FileSystem.get(job.getConfiguration());  
//    if(fs.exists(outDir)) {  
//        fs.delete(outDir, true);  
//    }  
  
//  
//    Boolean a=job.waitForCompletion(true);  
//    if(a==true)  
//    {  
        Job job2 = Job.getInstance();
```

```
        job2.setJarByClass(InverteBloomChain.Driver.class);
```

```
        FileInputFormat.addInputPath(job2, new Path(args[0]));  
        Path outDir1 = new Path(args[1]);  
        FileOutputFormat.setOutputPath(job2, outDir1);
```

```
        job2.setMapperClass(DistinctMapper.class);  
        job2.setReducerClass(DistinctReducer.class);  
        job2.setCombinerClass(DistinctReducer.class);
```

```
        job2.setOutputFormatClass(TextOutputFormat.class);  
        job2.setMapOutputValueClass(NullWritable.class);  
        job2.setMapOutputKeyClass(Text.class);  
        job2.setInputFormatClass(NLineInputFormat.class);
```

```

NLineInputFormat.setNumLinesPerSplit(job2, 10000);
job2.setNumReduceTasks(1);

job2.waitForCompletion(true);
// }

}

package Distinct;

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

import java.io.IOException;

public class DistinctMapper extends Mapper<LongWritable, Text, Text, NullWritable> {
    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException,
    InterruptedException {
        try {
            String input = value.toString();

            String splitString[] = input.split(",");
            if(splitString.length>10 && !input.contains("YEAR")) {
                String origin = splitString[6].replace("\r\n","");
                context.write(new Text(origin), NullWritable.get());
            }
        }
    }
}

```

```

        }

    } catch(Exception e) {
        System.out.println(e);
    }
}

}

package Distinct;

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

import java.io.IOException;

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

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

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

```

15. Flight Carrier Distance Memory Conscious Median:

```

package FlightCarrierDistanceMedian;

import org.apache.hadoop.fs.Path;

```

```
import org.apache.hadoop.io.SortedMapWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class Driver {
    public static void main( String[] args ) throws IOException, ClassNotFoundException,
InterruptedException
    {
        Job job = Job.getInstance();

        job.setJarByClass(Driver.class);

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

        job.setMapperClass(MyMapper.class);
        job.setReducerClass(MyReducer.class);
        job.setCombinerClass(myCombiner.class);

        //job.setOutputFormatClass(TextOutputFormat.class);
        job.setMapOutputValueClass(SortedMapWritable.class);
        job.setMapOutputKeyClass(Text.class);
        job.setInputFormatClass(TextInputFormat.class);
```

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

        job.setNumReduceTasks(1);

        job.waitForCompletion(true);
    }
}
```

```
package FlightCarrierDistanceMedian;

import org.apache.hadoop.io.Writable;

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

public class MedianStdDevTuple implements Writable {

    private float median;
    private float stdDev;

    public MedianStdDevTuple(float median, float stdDev) {
        this.median = median;
        this.stdDev = stdDev;
    }

    public float getMedian() {
        return median;
    }
}
```

```
public void setMedian(float median) {  
    this.median = median;  
}  
  
public float getStdDev() {  
    return stdDev;  
}  
  
public void setStdDev(float stdDev) {  
    this.stdDev = stdDev;  
}  
  
public void write(DataOutput out) throws IOException {  
    out.writeFloat(this.median);  
    out.writeFloat(this.stdDev);  
}  
  
@Override  
public String toString() {  
  
    return "median:"+ this.median +" Std_Div:"+this.stdDev;  
}  
  
public void readFields(DataInput in) throws IOException {  
    this.median=in.readFloat();  
    this.stdDev=in.readFloat();  
}
```

```

public MedianStdDevTuple() {
}

}

package FlightCarrierDistanceMedian;

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

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

public class myCombiner extends Reducer<Text, SortedMapWritable, Text, SortedMapWritable> {

    @Override
    protected void reduce(Text key, Iterable<SortedMapWritable> values, Context context) throws
    IOException, InterruptedException {
        // int sum=0;
        SortedMapWritable map = new SortedMapWritable();
        for(SortedMapWritable v:values) {
            for(Object entry : v.entrySet() ) {
                if(entry instanceof Map.Entry) {
                    Map.Entry entry1 = (Map.Entry) entry;
                    IntWritable count = (IntWritable) map.get(entry1.getKey());
                    FloatWritable mapKey = ((FloatWritable) entry1.getKey());
                    Integer mapValue = ((IntWritable) entry1.getValue()).get();
                    if (count != null) {
                        map.put(mapKey, new IntWritable(count.get() + ((IntWritable) entry1.getValue()).get()));
                    } else {
                }
            }
        }
    }
}

```

```
        map.put(mapKey, new IntWritable(((IntWritable) entry1.getValue()).get()));

    }

}

v.clear();

}

context.write(key,map);

}

}
```

```
package FlightCarrierDistanceMedian;

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

import java.io.IOException;

public class MyMapper extends Mapper<LongWritable, Text, Text, SortedMapWritable> {

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        try {
            String split[]=value.toString().split(",");
            IntWritable one = new IntWritable(1);
            String symbol = split[6];
            if(!split[0].contains("YEAR") && split.length>30 && !split[31].equalsIgnoreCase("0") &&
            !split[31].equalsIgnoreCase("\\"0\\\"")) {
```

```

        Float avg = Float.parseFloat(split[39]);
        SortedMapWritable map = new SortedMapWritable();
        map.put(new FloatWritable(avg),one);
        context.write(new Text(symbol.replace("\n","")),map);
    }

} catch(Exception ex) {
    System.out.println("Exception in mapper"+ex);
}
}

package FlightCarrierDistanceMedian;

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

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

public class MyReducer extends Reducer<Text, SortedMapWritable, Text, MedianStdDevTuple> {
    @Override
    protected void reduce(Text key, Iterable<SortedMapWritable> values, Context context) throws
    IOException, InterruptedException {
        //int sum = 0;
        //int count = 0;
        MedianStdDevTuple result = new MedianStdDevTuple();
    }
}

```

```
TreeMap<Float, Integer> finalMap= new TreeMap<Float, Integer>();  
  
try{  
    Float sum = 0f;  
    Integer totalCount = 0;  
    for(SortedMapWritable v:values) {  
        for(Object entry: v.entrySet() ) {  
            if(entry instanceof Map.Entry) {  
                Map.Entry entry1 = (Map.Entry) entry;  
                Float mapKey = ((FloatWritable) entry1.getKey()).get();  
                Integer mapValue = ((IntWritable) entry1.getValue()).get();  
                Integer count = finalMap.get(mapKey);  
  
                sum = sum + mapValue * mapKey;  
                totalCount = totalCount + mapValue;  
  
                if (count != null) {  
                    finalMap.put(mapKey, mapValue + count);  
                } else {  
                    finalMap.put(mapKey, mapValue);  
                }  
            }  
        }  
        v.clear();  
    }  
    Integer middleIndex = totalCount/2;  
    int prev=0;
```

```

int current = 0;
Float prevKey = 0f;
Float median=0f;

for(Map.Entry<Float,Integer> entry: finalMap.entrySet()) {
    current = entry.getValue();
    if(middleIndex > prev && middleIndex<=(current+prev)) {
        if(totalCount%2==0) {
            median = (entry.getKey()+prevKey)/2;
        } else {
            median = entry.getKey();
        }
        break;
    }
    prev= current+prev;
    prevKey = entry.getKey();
}

result.setMedian(median);

Float mean = sum/totalCount;

Float sumOfSquares=0.f;
for(Map.Entry<Float,Integer> entry:finalMap.entrySet()) {
    sumOfSquares = sumOfSquares + (entry.getKey()-mean)*(entry.getKey()-mean);
}
Float stdDev = (float) Math.sqrt(sumOfSquares/(totalCount-1));

```

```

        result.setStdDev(stdDev);

        context.write(key,result);

    } catch(Exception e) {
        System.out.println("exception in reducer "+ e);
    }
}
}

```

16. Bloom Filter

Inverted Bloom Chain:

```

package InverteBloomChain;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

public class Driver {

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

```

```
{  
    Job job = Job.getInstance();  
  
    job.setJarByClass(Driver.class);  
  
    FileInputFormat.addInputPath(job, new Path(args[0]));  
    Path outDir = new Path(args[1]+"_1");  
    FileOutputFormat.setOutputPath(job, outDir);  
  
    job.setMapperClass(FilterMapper.class);  
    // job.setReducerClass(MyReducer.class);  
    // job.setCombinerClass(MyReducer.class);  
  
    job.setOutputFormatClass(TextOutputFormat.class);  
    job.setMapOutputValueClass(NullWritable.class);  
    job.setMapOutputKeyClass(Text.class);  
    job.setInputFormatClass(TextInputFormat.class);  
  
    // job.setNumReduceTasks(1);  
  
    Boolean a=job.waitForCompletion(true);  
  
    if(a==true)  
    {  
        Job job2 = Job.getInstance();  
  
        job2.setJarByClass(Driver.class);
```

```
// job2.setGroupingComparatorClass(GroupComparator.class);
// job2.setSortComparatorClass(SecodarySortComparator.class);
//job2.setPartitionerClass(KeyPartition.class);

FileInputFormat.addInputPath(job2, new Path(args[1]+"_1"));

Path outDir1 = new Path(args[1]);
FileOutputFormat.setOutputPath(job2, outDir1);

job2.setMapperClass(MyMapper.class);
job2.setReducerClass(MyReducer.class);
job2.setCombinerClass(MyReducer.class);

job2.setOutputFormatClass(TextOutputFormat.class);
job2.setMapOutputValueClass(Text.class);
job2.setMapOutputKeyClass(Text.class);
job2.setInputFormatClassTextInputFormat.class);

job2.setNumReduceTasks(1);

// job2.setOutputKeyClass(CompositeKeyWritable.class);
// job2.setOutputValueClass(NullWritable.class);

job2.waitForCompletion(true);

}

}

}

package InverteBloomChain;
```

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

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

import org.apache.hadoop.io.NullWritable;

import com.google.common.base.Charsets;
import com.google.common.hash.BloomFilter;
import com.google.common.hash.Funnel;
import com.google.common.hash.Sink;

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

    private BloomFilter<String> origin;

    Funnel<String> p = new Funnel<String>() {

        public void funnel(String from, Sink into) {

            into.putString(from, Charsets.UTF_8);

        }
    };

    @Override
```

```
protected void setup(Mapper<LongWritable, Text, Text, NullWritable>.Context context) throws
IOException, InterruptedException {

    this.origin = BloomFilter.create(p, 300000, 0.1);

    String p1 = "BOS";
    String p2 = "JFK";
    // Person p2 = new Person("Jamie", "Scott");

    ArrayList<String> originList = new ArrayList<String>();
    originList.add(p1);
    originList.add(p2);

    for (String ps : originList) {
        origin.put(ps);
    }

}

@Override
protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
    try {
        String input = value.toString();

        String splitString[] = input.split(",");
        String origins = splitString[10].replace("\\"","").trim();
        //System.out.println(origins);
        //String values[] = value.toString().split(",");
    }
}
```

```

//Person p = new Person(values[1], values[2]);

if (origin.mightContain(origins) && !splitString[0].contains("YEAR")) {
    context.write(value, NullWritable.get());
}

//context.write(new Text(origin.replace("\n", "").trim()), new
Text(splitString[6].replace("\n", "").trim()));

} catch(Exception e) {
    System.out.println(e);
}

}

@Override
protected void cleanup(Context context) throws IOException, InterruptedException
{
    super.cleanup(context);
}
}

```

17. Inverted Index

```

package InverteBloomChain;

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

import java.io.IOException;

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

```

```
    @Override  
  
    protected void map(LongWritable key, Text value, Context context) throws IOException,  
    InterruptedException {  
  
        try {  
  
            String input = value.toString();  
  
            String splitString[] = input.split(",");  
            String origin = splitString[10];  
  
            context.write(new Text(origin.replace("\n", "").trim()), new  
Text(splitString[6].replace("\n", "").trim()));  
  
        } catch(Exception e) {  
            System.out.println(e);  
        }  
    }  
}
```

```
package InverteBloomChain;  
  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Reducer;  
  
import java.io.IOException;  
  
public class MyReducer extends Reducer<Text, Text, Text, Text> {  
  
    @Override  
  
    protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException,  
    InterruptedException {  
  
        String concatString = "";
```

```

for(Text s:values) {
    if(!concatString.contains(s.toString())) {
        concatString = s.toString() + "," + concatString;
    }
}

context.write(key,new Text(concatString));
}
}

```

18. Join:

```

package Join;

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

import java.io.IOException;

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

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String[] fields = value.toString().split(",");
        Text t1 = new Text(fields[0].replace("\r\n", "").trim());
        String val = "M"+fields[1].replace("\r\n", "").trim();
        Text t2 = new Text(val);
        context.write(t1, t2);
    }
}

```

```
        context.write(t1,t2);

    }

}

package Join;

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

import java.io.IOException;

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

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException,
    InterruptedException {
        String[] fields = value.toString().split(" ");
        Text t1 = new Text(fields[0].replace("\r\n", "").trim());
        String val = "R"+fields[0].replace("\r\n", "").trim();
        Text t2 = new Text(val);
        context.write(t1,t2);
    }
}

package Join;

import org.apache.hadoop.fs.FileSystem;
```

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class JoinPattern {

    public static void main(String[] args){

        try {
            Job job = Job.getInstance();
            job.setJarByClass(JoinPattern.class);

            job.getConfiguration().set("type",args[3]);

            MultipleInputs.addInputPath(job, new Path(args[1]),TextInputFormat.class, DataMapper.class);
            MultipleInputs.addInputPath(job,new Path(args[0]), TextInputFormat.class, AirlineMapper.class);

            //Reducer
            job.setReducerClass(JoinReducer.class);
            job.setOutputFormatClass(TextOutputFormat.class);

            //Number of Reducers
            job.setNumReduceTasks(1);

            //Specify Key value
        }
    }
}
```

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

//Specify Output Path
Path output = new Path(args[2]);
FileOutputFormat.setOutputPath(job, output);

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

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

}catch(Exception ex){
    ex.printStackTrace();
}
}

package Join;

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

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

public class JoinReducer extends Reducer<Text, Text, Text, Text> {
```

```

String joinType=null;

Text emptyText = new Text("");
Text temp = new Text();
ArrayList<Text> r = new ArrayList<Text>();
ArrayList<Text> m = new ArrayList<Text>();

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

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

    for(Text val: values){
        temp=val;
        if(val.toString().charAt(0)=='R') r.add(new Text(temp.toString().substring(1)));
        else if(val.toString().charAt(0)=='M') m.add(new Text(temp.toString().substring(1)));
    }
    executeJoin(context);
}

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

    if(joinType.equalsIgnoreCase("inner")){

```

```

if(!r.isEmpty() && !m.isEmpty()){

    for(Text ra: r){

        for(Text mo: m){

            context.write(ra,mo);

        }

    }

}

else if(joinType.equalsIgnoreCase("leftouter")){

    for(Text ra: r){

        if(!m.isEmpty()){

            for(Text mo : m){

                context.write(ra,mo);

            }

        }

        else context.write(ra,emptyText);

    }

}

else if(joinType.equalsIgnoreCase("rightouter")){

    for(Text mo: m){

        if(!r.isEmpty()){

            for(Text ra: r){

                context.write(ra,mo);

            }

        }

        else context.write(emptyText,mo);

    }

}

```

```

    }

else if(joinType.equalsIgnoreCase("fullouter")){
    if(!r.isEmpty()){
        for(Text ra:r){
            if(!m.isEmpty()){
                for(Text mo: m){
                    context.write(ra,mo);
                }
            }else context.write(ra,emptyText);
        }
    }
    else {
        for(Text mo: m){
            context.write(emptyText,mo);
        }
    }
}

else if(joinType.equalsIgnoreCase("antijoin")){
    if(r.isEmpty()^m.isEmpty()){
        for(Text ra: r){
            context.write(ra,emptyText);
        }
        for(Text mo: m){
            context.write(emptyText,mo);
        }
    }
}

```

```
 }  
 }
```

19. PutMerge:

```
package PutMerge;  
  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.*;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
  
public class MergeFiles {  
  
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {  
  
        //Hadoop File System instance  
        Configuration conf = new Configuration();  
        FileSystem hdfs = FileSystem.get(conf);  
  
        //local File System instance  
        FileSystem local = FileSystem.getLocal(conf);  
  
        //Path to the directory in local file System  
        Path inputDir = new Path(args[0]);  
        Path hdfsFile = new Path(args[1]);
```

```
try {

    //List of Files Names from FileStatus
    FileStatus[] inputFiles = local.listStatus(inputDir);

    //Writing to hdfs using OutputStream
    FSDataOutputStream out = hdfs.create(hdfsFile);

    //Reading the input files using FSDataInputStream
    for (int i = 0; i < inputFiles.length; i++) {

        FSDataInputStream in = local.open(inputFiles[i].getPath());

        // BufferedReader br = new BufferedReader(new InputStreamReader(in, UTF8));
        //int lineCount = 0;
        byte buffer[] = new byte[256];
        int bytesRead = 0;

        while ((bytesRead = in.read(buffer)) > 0 ) {

            out.write(buffer, 0, bytesRead);
        }

        in.close();
    }

    out.close();

} catch (IOException e) {
    e.printStackTrace();
}

}
```

20. Secondary Sort:

```
package SecSortPartitionerWritable;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

public class Driver
{
    public static void main( String[] args ) throws IOException, ClassNotFoundException,
    InterruptedException
    {
        Job job = Job.getInstance();

        job.setJarByClass(Driver.class);

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

        FileInputFormat.addInputPath(job, new Path(args[0]));
    }
}
```

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

job.setMapperClass(MyMapper.class);
job.setReducerClass(MyReducer.class);

job.setNumReduceTasks(1);

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

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

Boolean a=job.waitForCompletion(true);

}

}

package SecSortPartitionerWritable;

import org.apache.hadoop.io.WritableComparable;

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

```
public class CompositeKeyWritable implements WritableComparable<CompositeKeyWritable> {

    private String state;
    private String airportId;

    public CompositeKeyWritable() {
        super();
    }

    public CompositeKeyWritable(String state, String id) {
        super();
        this.state = state;
        this.airportId = id;
    }

    public void write(DataOutput out) throws IOException {
        out.writeUTF(state);
        out.writeUTF(airportId);
    }

    public void readFields(DataInput in) throws IOException {
        state = in.readUTF();
        airportId = in.readUTF();
    }

    public int compareTo(CompositeKeyWritable o) {
        int result = this.state.compareTo(o.state);

```

```
        if (result == 0) {

            return this.airportId.compareTo(o.airportId);

        }

        return result;

    }

}

public String getState() {

    return state;

}

public void setState(String state) {

    this.state = state;

}

public String getAirportId() {

    return airportId;

}

public void setAirportId(String airportId) {

    this.airportId = airportId;

}

@Override

public String toString() {

    return state + "," + airportId;

}

}
```

```
package SecSortPartitionerWritable;

import org.apache.hadoop.io.WritableComparator;

public class GroupComparator extends WritableComparator {

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

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

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

        return ckw1.getState().compareTo(ckw2.getState());
    }
}

package SecSortPartitionerWritable;

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

public class KeyPartition extends Partitioner<CompositeKeyWritable, NullWritable> {
```

```
    @Override
    public int getPartition(CompositeKeyWritable key, NullWritable value, int numPartitions) {
        return key.getState().hashCode()%numPartitions;
    }

}

package SecSortPartitionerWritable;

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

import java.io.IOException;

public class MyMapper extends Mapper<LongWritable, Text, CompositeKeyWritable, NullWritable> {

    @Override
    protected void map(LongWritable key, Text value,
                       Mapper<LongWritable, Text, CompositeKeyWritable, NullWritable>.Context context)
            throws IOException, InterruptedException {
        String line = value.toString();

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

```
if(!value.toString().contains("YEAR")) {  
    try {  
  
        state = tokens[14].replace("\'", "").trim();  
        airportId = tokens[10].replace("\'", "").trim();  
  
    } catch (Exception e) {  
  
    }  
  
    if (state != null && airportId != null) {  
  
        CompositeKeyWritable outKey = new CompositeKeyWritable(state, airportId);  
  
        context.write(outKey, NullWritable.get());  
  
    }  
}  
  
}  
  
package SecSortPartitionerWritable;  
  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.mapreduce.Reducer;  
  
import java.io.IOException;
```

```
public class MyReducer extends Reducer<CompositeKeyWritable, NullWritable, CompositeKeyWritable,
NullWritable> {

    @Override
    protected void reduce(CompositeKeyWritable key, Iterable<NullWritable> values,
        Reducer<CompositeKeyWritable, NullWritable, CompositeKeyWritable,
NullWritable>.Context context)
        throws IOException, InterruptedException {
        //for (NullWritable v : values) {
        //    context.write(key, NullWritable.get());
        //}
    }
}

package SecSortPartitionerWritable;

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

public class SecodarySortComparator extends WritableComparator {
    protected SecodarySortComparator() {
        super(CompositeKeyWritable.class, true);
    }
    @Override
```

```

public int compare(WritableComparable a, WritableComparable b) {

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

    int result = ckw1.getState().compareTo(ckw2.getState());

    if (result == 0) {
        return ckw1.getAirportId().compareTo(ckw2.getAirportId());
    }

    return result;
}

}

```

21. Shuffle:

```

package ShuffleData;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

```

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

import java.io.IOException;

public class Driver {

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

        Job job = Job.getInstance();

        job.setJarByClass(Driver.class);

        job.setMapperClass(ShuffleMapper.class);

        job.setReducerClass(ShuffleReducer.class);

        job.setInputFormatClass(TextInputFormat.class);

        job.setOutputFormatClass(TextOutputFormat.class);

        job.setOutputKeyClass(IntWritable.class);

        job.setOutputValueClass(Text.class);

        job.setNumReduceTasks(1);

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

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

        FileSystem fs = FileSystem.get(job.getConfiguration());

        job.waitForCompletion(true);

    }

}
```

```
}
```

```
package ShuffleData;
```

```
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;
```

```
import java.util.Random;
```

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

```
    private static final IntWritable one = new IntWritable(1);
```

```
    private Random random= new Random();
```

```
    @Override
```

```
    protected void map(LongWritable key, Text value, Context context)
```

```
        throws IOException, InterruptedException {
```

```
        String line = value.toString();
```

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

```
        if(tokens.length > 15) {
```

```
            context.write(new IntWritable(random.nextInt()), value);
```

```
        }
```

```
}
```

```
    @Override
```

```
    protected void cleanup(Context context)
```

```
    throws IOException, InterruptedException {  
        // TODO Auto-generated method stub  
        super.cleanup(context);  
    }  
  
    @Override  
    public void run(Context context)  
        throws IOException, InterruptedException {  
        // TODO Auto-generated method stub  
        super.run(context);  
    }  
  
    @Override  
    protected void setup(Context context)  
        throws IOException, InterruptedException {  
        // TODO Auto-generated method stub  
        super.setup(context);  
    }  
  
}
```

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

```
import java.io.IOException;

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

    @Override
    protected void cleanup(Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.cleanup(context);
    }

    @Override
    protected void reduce(IntWritable key, Iterable<Text> values,
        Context context) throws IOException, InterruptedException {
        for (Text val: values) {
            // context.write(new IntWritable(),val);
            context.write(val,NullWritable.get());
        }
    }

    @Override
    public void run(Context arg0)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.run(arg0);
    }

    @Override
    protected void setup(Context context)
```

```
    throws IOException, InterruptedException {  
  
    // TODO Auto-generated method stub  
  
    super.setup(context);  
  
}  
  
}
```

22. State Cancellation:

```
package StateCancellationCount6;
```

```
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import java.io.IOException;
```

```
public class Driver {  
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {  
        Job job = Job.getInstance();  
        job.setJarByClass(Driver.class);  
        job.setMapperClass(CancelCountMapper.class);  
    }  
}
```

```
job.setCombinerClass(CancelCountReducer.class);
job.setReducerClass(CancelCountReducer.class);

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

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

job.setNumReduceTasks(1);

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

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

// Submit the job, then poll for progress until the job is complete
job.waitForCompletion(true);
}

}

package StateCancellationCount6;

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

```

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

import java.io.IOException;

public class CancelCountMapper extends Mapper<LongWritable, Text, Text, Text> {
    private static final IntWritable one = new IntWritable(1);

    @Override
    protected void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
        String line = value.toString();
        String t= "";
        String[] tokens = line.split(",");
        // for(String s : tokens) {
        if(!tokens[0].contains("YEAR") && tokens.length >33 && !tokens[32].equalsIgnoreCase("0")
            && !tokens[33].equalsIgnoreCase(" ") &&
        !tokens[33].replace("\"\"", "").trim().equalsIgnoreCase("")) {
            context.write(new
Text(tokens[14].replace("\"\"", "").trim()+"\t"+tokens[33].replace("\"\"", "").trim()+"\t"+tokens[2]), new
Text(1+""));
        }
        //}
    }

    @Override
    protected void cleanup(Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
    }
}

```

```
        super.cleanup(context);

    }

    @Override
    public void run(Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.run(context);
    }

    @Override
    protected void setup(Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.setup(context);
    }

}

package StateCancellationCount6;

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

import java.io.IOException;

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

    @Override
```

```
protected void cleanup(Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.cleanup(context);
}

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

    int sum = 0;
    int count=0;
    for (Text value : values) {
        String a[]=value.toString().split(" ");
        // sum += Float.parseFloat(a[0]);
        count+=Integer.parseInt(a[0]);
    }

    context.write(key, new Text(count+""));
}

@Override
public void run(Context arg0)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.run(arg0);
}

@Override
```

```
protected void setup(Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.setup(context);
}

}
```

23 State Delay Count:

```
package StateDelayCount6;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

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

```
job.setMapperClass(DelayCountMapper.class);
job.setCombinerClass(DelayCountReducer.class);
job.setReducerClass(DelayCountReducer.class);

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

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

job.setNumReduceTasks(1);

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

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

// Submit the job, then poll for progress until the job is complete
job.waitForCompletion(true);
}

}

package StateDelayCount;

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

```

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

import java.io.IOException;

public class DelayCountMapper extends Mapper<LongWritable, Text, Text, Text> {
    private static final IntWritable one = new IntWritable(1);

    @Override
    protected void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
        String line = value.toString();
        String t= "";
        String[] tokens = line.split(",");
        Integer delay=0;
        // for(String s : tokens) {
        if(!tokens[0].contains("YEAR") && tokens.length >45 && !tokens[41].equalsIgnoreCase("") )
            && !tokens[41].equalsIgnoreCase(" ") && !tokens[41].equalsIgnoreCase("NA")) {
            if(Float.parseFloat(tokens[41]) >0 ){
                t="CARRIER_DELAY";
                delay=(int)Float.parseFloat(tokens[41]);
            }
            else if(Float.parseFloat(tokens[42]) >0 ){
                t="WEATHER_DELAY";
                delay=(int)Float.parseFloat(tokens[42]);
            }
            else if(Float.parseFloat(tokens[43]) >0 ){
                t="NAS_DELAY";
            }
        }
    }
}

```

```

        delay=(int)Float.parseFloat(tokens[43]);
    }

    else if (Float.parseFloat(tokens[44]) >0 ){
        t="SECURITY_DELAY";
        delay=(int)Float.parseFloat(tokens[44]);
    }

    else{
        t="LATE_AIRCRAFT_DELAY";
        delay=(int)Float.parseFloat(tokens[45]);
    }

    context.write(new Text(tokens[14].replace("\n","")+"\t"+t+"\t"+tokens[2]), new
Text(delay.toString()+"\t"+1));
}

//}

}

@Override
protected void cleanup(Context context)
throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.cleanup(context);
}

@Override
public void run(Context context)
throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.run(context);
}

```

```
    @Override
    protected void setup(Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.setup(context);
    }

}

package StateDelayCount6;

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

import java.io.IOException;

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

    @Override
    protected void cleanup(Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.cleanup(context);
    }

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

```

int sum = 0;
int count=0;
for (Text value : values) {
    String a[]=value.toString().split("\t");
    sum += Float.parseFloat(a[0]);
    count+=Integer.parseInt(a[1]);
}

context.write(key, new Text(sum+"\t"+count));
}

@Override
public void run(Context arg0)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.run(arg0);
}

@Override
protected void setup(Context context)
    throws IOException, InterruptedException {
// TODO Auto-generated method stub
super.setup(context);
}

}

```

24. Top 10 flight Distance:

```
package Top10FlightDistance;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args ) throws IOException, ClassNotFoundException,
InterruptedException
    {
        Job job = Job.getInstance();
        job.setJarByClass(App.class);

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

```
Path input = new Path(args[0]);
Path output = new Path(args[1]);

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

FileInputFormat.addInputPath(job, input);
FileOutputFormat.setOutputPath(job, output);

job.setNumReduceTasks(1);

job.setMapperClass(TopKMapper.class);
job.setReducerClass(TopKReducer.class);

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

job.waitForCompletion(true);
}

}

package Top10FlightDistance;

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

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

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

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

    private static final int K = 10;
    private TreeMap<Integer, Text> tMap;

    @Override
    protected void setup(Mapper<LongWritable, Text, NullWritable, Text>.Context context)
        throws IOException, InterruptedException {
        this.tMap = new TreeMap<Integer, Text>();
    }

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, NullWritable,
    Text>.Context context)
        throws IOException, InterruptedException {

        String[] tokens = value.toString().split(",");
        try {
            String score = tokens[39].replace("\'", "").trim();
            tMap.put(Integer.parseInt(score), new Text(value));
        } catch (Exception ex) {
    }
}
```

```
    }

    if (tMap.size() > K) {
        tMap.remove(tMap.firstKey());
    }
}

@Override
protected void cleanup(Mapper<LongWritable, Text, NullWritable, Text>.Context context)
    throws IOException, InterruptedException {

    for (Text t : tMap.values()) {
        context.write(NullWritable.get(), t);
    }
}

package Top10FlightDistance;

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

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

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

    private static final int K = 10;

    private TreeMap<Integer, Text> tMap;

    @Override
    protected void setup(Reducer<NullWritable, Text, NullWritable, Text>.Context context)
        throws IOException, InterruptedException {

        this.tMap = new TreeMap<Integer, Text>();
    }

    @Override
    protected void reduce(NullWritable arg0, Iterable<Text> values,
        Reducer<NullWritable, Text, NullWritable, Text>.Context arg2) throws IOException,
    InterruptedException {

        for (Text t : values) {
            String[] tokens = t.toString().split(",");
            tMap.put(Integer.parseInt(tokens[39].replace("\\" , "").trim()), new Text(t));

            if (tMap.size() > K) {
                tMap.remove(tMap.firstKey());
            }
        }
    }

    @Override
```

```

protected void cleanup(Reducer<NullWritable, Text, NullWritable, Text>.Context context)
    throws IOException, InterruptedException {

    for(Text t : tMap.values()) {
        context.write(NullWritable.get(), t);
    }
}

}

```

Pig:

25. Join:

```

uniqueid = LOAD 'distinct' using PigStorage(',') as (uid);
carriers = LOAD 'carriers.csv' using PigStorage(',') as (id,name);
joined = JOIN uniqueid BY uid, carriers BY id;
summed = FOREACH joined GENERATE $0,$2;
STORE summed into 'joinoutput';

```

26. Filter Merge:

```

flights = LOAD 'Merge.csv' using PigStorage(',');
filter1 = FILTER flights BY $0 > 2017;
filter2 = FOREACH filter1 GENERATE $0,$1,$2,$3,$4,$5,$6,$7,$8,$9,'LatestData';
STORE filter2 into 'LatestData';

```

27. Topk:

```

flights = LOAD 'apr17.csv' using PigStorage(',');
grouped = GROUP flights BY $6;
summed = FOREACH grouped GENERATE group, COUNT(flights) AS cntd;

```

```
sorted = ORDER summed BY cntd DESC;  
top25 = LIMIT sorted 10;  
Dump top25;  
STORE top25 into 'top10Carriers';
```

Additional Screenshots for reference:

AWS:

The screenshot shows the AWS EMR console interface. On the left, there's a sidebar with navigation links: Clusters, Security configurations, VPC subnets, Events, Notebooks, Help, and What's new. The main area has a header with a message about using the AWS Glue Data Catalog as an external Hive metastore. Below this is a 'Create cluster' button and a 'Clusters' section. A table lists one cluster: FlightCluster, which is currently in the 'Waiting Cluster ready' state. The table includes columns for Name, ID, Status, Creation time (UTC-4), Elapsed time, and Normalized instance hours. Under the 'Summary' section, it shows Master public DNS: ec2-34-201-245-127.compute-1.amazonaws.com, Termination protection: Off, and Tags: View All / Edit. The 'Hardware' section indicates Master: Running 1 m4.large, Core: Running 2 m4.large, and Task: -. At the bottom, there are links for View cluster details and View monitoring details.

This screenshot shows the 'Steps' tab for the FlightCluster in the AWS EMR console. The cluster status is listed as 'Waiting Cluster ready after last step completed'. The 'Steps' tab is selected, showing a table with one step: s-2KLGKR4N9VFOZ, which is a Custom JAR step. The step status is Completed, started at 2019-04-24 19:45 (UTC-4), and took 4 minutes. It has a 'View logs' link under Actions. The table has columns for ID, Name, Status, Start time (UTC-4), Elapsed time, Log files, and Actions. Below the table, it says 'Action on failure: Continue'. There are also buttons for Add step, Clone step, and Cancel step.

AWS Services Resource Groups

Amazon S3 > flightmapreduce > output > file > part-r-00000

part-r-00000 Latest version ▾

Overview Properties Permissions Select from

Open Download Download as Make public Copy path

Owner
shetty.poo

Last modified
Apr 24, 2019 7:50:11 PM GMT-0400

Etag
06bee8edc21074a4f2ff8f21c71d312a

Storage class
Standard

Server-side encryption
None

Size
4.6 KB

Key
output/file/part-r-00000

Object URL
<https://s3.amazonaws.com/flightmapreduce/output/file/part-r-00000>

Amazon S3 > flightmapreduce > output > file

Overview

Type a prefix and press Enter to search. Press ESC to clear.

Upload Create folder Download Actions ▾

US East (N. Virginia) ⚙ Viewing 1 to 2

| Name | Last modified | Size | Storage class |
|--------------|----------------------------------|--------|---------------|
| _SUCCESS | Apr 24, 2019 7:50:11 PM GMT-0400 | 0 B | Standard |
| part-r-00000 | Apr 24, 2019 7:50:11 PM GMT-0400 | 4.6 KB | Standard |

Viewing 1 to 2

Amazon S3 > flightmapreduce > jar

Overview

Type a prefix and press Enter to search. Press ESC to clear.

[Upload](#) [+ Create folder](#) [Download](#) [Actions](#)

US East (N. Virginia) [Edit](#)

Viewing 1 to 1

| Name | Last modified | Size | Storage class |
|-----------------|----------------------------------|---------|---------------|
| FlightDelay.jar | Apr 24, 2019 7:38:28 PM GMT-0400 | 77.5 KB | Standard |

Viewing 1 to 1

Amazon S3 > flightmapreduce > input

Overview

Type a prefix and press Enter to search. Press ESC to clear.

[Upload](#) [+ Create folder](#) [Download](#) [Actions](#)

US East (N. Virginia) [Edit](#)

Viewing 1 to 24

| Name | Last modified | Size | Storage class |
|-----------|----------------------------------|----------|---------------|
| apr17.csv | Apr 24, 2019 6:50:09 PM GMT-0400 | 115.6 MB | Standard |
| apr18.csv | Apr 24, 2019 6:50:09 PM GMT-0400 | 150.4 MB | Standard |
| aug17.csv | Apr 24, 2019 6:50:09 PM GMT-0400 | 126.0 MB | Standard |
| aug18.csv | Apr 24, 2019 6:50:10 PM GMT-0400 | 159.9 MB | Standard |
| dec17.csv | Apr 24, 2019 6:50:10 PM GMT-0400 | 115.0 MB | Standard |
| dec18.csv | Apr 24, 2019 6:51:41 PM GMT-0400 | 147.4 MB | Standard |
| feb17.csv | Apr 24, 2019 6:51:51 PM GMT-0400 | 111.2 MB | Standard |
| feb18.csv | Apr 24, 2019 6:52:11 PM GMT-0400 | 140.3 MB | Standard |
| jan17.csv | Apr 24, 2019 6:52:18 PM GMT-0400 | 111.2 MB | Standard |

Amazon S3 > ... > 000000 > job_1556143697838_0001-1556149535584-hadoop-FlightDelay.jar-1556149810763-60-1-SUCCEEDED-default-1556149546461.jhist.gz

job_1556143697838_0001-1556149535584-hadoop-FlightDelay.jar-1556149810763-60-1-SUCCEEDED-default-1556149546461.jhist.gz Latest version ▾

Overview Properties Permissions Select from

[Open](#) [Download](#) [Download as](#) [Make public](#) [Copy path](#)

Owner
shetty.poo

Last modified
Apr 24, 2019 7:58:17 PM GMT-0400

ETag
ee493fe3857fa1d2785881c5bcd44eb4

Storage class
Standard

Server-side encryption
AES-256

Size
21.2 KB

Key
logslj-266838CXTTDT1/hadoop-mapreduce/history/2019/04/24/000000/job_1556143697838_0001-1556149535584-hadoop-FlightDelay.jar-1556149810763-60-1-SUCCEEDED-default-1556149546461.jhist.gz

Object URL
https://s3.amazonaws.com/flightmapreduce/logs/j-266838CXTTDT1/hadoop-mapreduce/history/2019/04/24/000000/job_1556143697838_0001-1556149535584-hadoop-FlightDelay.jar-1556149810763-60-1-SUCCEEDED-default-1556149546461.jhist.gz

Open ▾

part-r-00000
~/Downloads

| | | | |
|----|---|----|------|
| 9E | A | 1 | 13 |
| 9E | A | 10 | 12 |
| 9E | A | 12 | 23 |
| 9E | A | 3 | 494 |
| 9E | A | 5 | 8 |
| 9E | A | 9 | 7 |
| 9E | B | 1 | 364 |
| 9E | B | 10 | 106 |
| 9E | B | 12 | 28 |
| 9E | B | 3 | 2086 |
| 9E | B | 5 | 350 |
| 9E | B | 7 | 12 |
| 9E | B | 9 | 36 |
| 9E | C | 1 | 1077 |
| 9E | C | 10 | 60 |
| 9E | C | 12 | 65 |
| 9E | C | 3 | 486 |
| 9E | C | 5 | 84 |
| 9E | C | 7 | 1302 |
| 9E | C | 9 | 100 |
| AA | A | 1 | 1008 |
| AA | A | 10 | 642 |
| AA | A | 12 | 553 |
| AA | A | 3 | 621 |
| AA | A | 4 | 274 |
| AA | A | 5 | 1180 |
| AA | A | 7 | 2335 |
| AA | A | 8 | 608 |
| AA | A | 9 | 333 |
| AA | B | 1 | 3671 |
| AA | B | 10 | 1168 |
| AA | B | 12 | 1132 |
| AA | B | 3 | 5525 |
| AA | B | 4 | 356 |
| AA | B | 5 | 1322 |
| AA | B | 7 | 2290 |
| AA | B | 8 | 974 |
| AA | B | 9 | 1021 |
| AA | C | 1 | 214 |
| AA | C | 10 | 266 |
| AA | C | 12 | 13 |
| AA | C | 3 | 57 |

Counting with Counters:

The screenshot shows an IDE interface with the following details:

- Project Structure:**
 - Root: .idea, FlightDelayData, OAirlineCancel, OAirlineDelay, OAvgDelayMonthlyW, OAvgFlightDelay, OCancellationCount, OCounters, OFlightCancellation.
 - src/main/java:
 - AirlineCancel7
 - AirlineDelay7
 - AverageFlightDelay
 - AvgDelayMonthly
 - BinningFlight
 - CancellationCount2
 - CountersDistanceGroup
 - Driver
- Code Editor:** Shows the `Driver` class with the `main()` method. The code uses a Job object to set mapper and output classes, and then waits for completion to print counter values.
- Run Tab:** Shows the output of the `CountingCounters` task. It displays the following counter output:

| Counter Name | Value |
|--------------|-------|
| 1: 1731883 | |
| 10: 385284 | |
| 11: 282962 | |
| 2: 3285052 | |
| 3: 2669652 | |
| 4: 2092525 | |
| 5: 1431105 | |
| 6: 604557 | |
| 7: 631685 | |
| 8: 325164 | |
| 9: 227472 | |

Median Based on Flight Carrier SD: MCD Combiner used

The screenshot shows a terminal window with the following output:

```

1  9E median:422.0 Std_Div:11.182264
2  AA median:918.5 Std_Div:10.709496
3  AS median:972.5 Std_Div:19.131805
4  B6 median:992.0 Std_Div:13.007162
5  DL median:679.5 Std_Div:12.000799
6  EV median:435.0 Std_Div:8.557014
7  F9 median:978.5 Std_Div:18.91692
8  G4 median:895.0 Std_Div:20.08316
9  HA median:142.0 Std_Div:24.758072
10 MQ median:372.0 Std_Div:8.99425
11 NK median:975.0 Std_Div:13.585728
12 OH median:356.0 Std_Div:6.303177
13 OO median:398.0 Std_Div:9.116984
14 UA median:991.0 Std_Div:14.888375
15 VX median:1476.0 Std_Div:19.074
16 WN median:632.0 Std_Div:8.721409
17 YV median:485.5 Std_Div:9.903636
18 YX median:542.0 Std_Div:12.116253
19

```

Airline Cancelled each month count:

FlightDelay

```

1 9E A 1 21
2 9E A 10 12
3 9E A 12 23
4 9E A 3 494
5 9E A 5 8
6 9E A 9 7
7 9E B 1 556
8 9E B 10 106
9 9E B 12 28
10 9E B 3 2086
11 9E B 5 350
12 9E B 7 12
13 9E B 9 36
14 9E C 1 1990
15 9E C 10 60
16 9E C 12 65
17 9E C 3 486
18 9E C 5 84
19 9E C 7 1302
20 9E C 9 100
21 AA A 1 1191
22 AA A 10 642
23 AA A 12 553
24 AA A 3 621
25 AA A 4 274
26 AA A 5 1180
27 AA A 7 2335
28 AA A 8 608
29 AA A 9 333
30 AA B 1 4951
31 AA B 10 1168

```

Airline Delay each month count:

FlightDelay

```

1 9E CARRIER DELAY 1 224732 3876
2 9E CARRIER DELAY 10 105540 2476
3 9E CARRIER DELAY 12 78842 1506
4 9E CARRIER DELAY 3 159678 3172
5 9E CARRIER DELAY 5 145310 2577
6 9E CARRIER DELAY 9 130424 32976
7 9E CARRIER DELAY 9 55757 1108
8 9E LATE AIRCRAFT DELAY 1 102361 1522
9 9E LATE AIRCRAFT DELAY 10 61784 1016
10 9E LATE AIRCRAFT DELAY 12 31276 451
11 9E LATE AIRCRAFT DELAY 5 62896 1004
12 9E LATE AIRCRAFT DELAY 5 11974 204
13 9E LATE AIRCRAFT DELAY 7 69076 1008
14 9E LATE AIRCRAFT DELAY 9 30153 498
15 9E NAS DELAY 1 168787 3904
16 9E NAS DELAY 10 108246 3666
17 9E NAS DELAY 12 1014 1157
18 9E NAS DELAY 3 81396 242
19 9E NAS DELAY 5 118902 2714
20 9E NAS DELAY 7 215602 3690
21 9E NAS DELAY 9 76986 1513
22 9E SECURITY DELAY 1 10472 3
23 9E SECURITY DELAY 10 66 2
24 9E SECURITY DELAY 12 229 4
25 9E SECURITY DELAY 3 140 6
26 9E SECURITY DELAY 5 28 2
27 9E SECURITY DELAY 7 4 2
28 9E SECURITY DELAY 9 10005 875
29 9E WEATHER DELAY 10 15658 202
30 9E WEATHER DELAY 12 21401 252
31 9E WEATHER DELAY 3 16784 290
32 9E WEATHER DELAY 5 10277 178
Run: MCDMedian

```

Avg Delay each month:

FlightDelay

```

1 9E, (1,1473), (3,1479), (5,1467), (7,1448), (9,1469), (10,1472), (12,1470)
2 AA, (1,1491), (3,1482), (4,1483), (5,1474), (7,1451), (8,1469), (9,1461), (10,1478), (12,1478)
3 AS, (1,1475), (3,1458), (4,1452), (5,1459), (7,1448), (8,1458), (9,1483), (10,1481), (12,1464)
4 DU, (1,1471), (3,1464), (4,1459), (5,1452), (7,1448), (8,1458), (9,1483), (10,1481), (12,1459)
5 EV, (1,1471), (3,1466), (4,1465), (5,1458), (7,1460), (8,1459), (9,1458), (10,1466), (12,1459)
6 F9, (1,1451), (3,1452), (4,1454), (5,1428), (7,1412), (8,1432), (9,1426), (10,1436), (12,1459)
7 G4, (1,1580), (3,1528), (5,1581), (7,1569), (9,1589), (10,1512), (12,1522)
8 HA, (1,1464), (3,1464), (4,1464), (5,1459), (7,1459), (8,1459), (9,1459), (10,1466), (12,1426)
9 MO, (1,1464), (3,1462), (4,1455), (5,1458), (7,1458), (8,1456), (10,1458), (12,1459)
10 NK, (1,1465), (3,1464), (4,1467), (5,1442), (7,1426), (8,1432), (9,1448), (10,1456), (12,1464)
11 OH, (1,1474), (3,1474), (4,1469), (5,1469), (7,1474), (9,1466), (10,1471), (12,1454)
12 OO, (1,1464), (3,1464), (4,1464), (5,1459), (7,1459), (8,1459), (9,1459), (10,1457), (12,1466)
13 UH, (1,1462), (3,1473), (4,1463), (5,1451), (7,1413), (8,1429), (9,1455), (10,1466), (12,1472)
14 VX, (1,1541), (3,1519), (4,1530), (5,1534), (7,1529), (8,1529), (10,1516), (12,1514)
15 WH, (1,1494), (3,1455), (4,1466), (5,1455), (7,1426), (8,1454), (9,1485), (10,1484), (12,1465)
16 YK, (1,1480), (3,1485), (4,1471), (5,1471), (7,1461), (8,1469), (10,1476), (12,1478)

```

Avg Flight Delay:

```
1  * @param median
2  * @param stdDev
3  * @param min
4  * @param max
5  */
6  public MedianStdDevTuple(double median, double stdDev, double min, double max) {
7      this.m = median;
8      this.s = stdDev;
9      this.mn = min;
10     this.mx = max;
11 }
12 
13 /**
14  * @return standard deviation
15  */
16 public double getStdDev() {
17     return s;
18 }
19 
20 /**
21  * @param val
22  * @return true if val is between min and max
23  */
24 public boolean isBetween(double val) {
25     return val > mn && val < mx;
26 }
```

Total Cancellation Count by reason:

The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure under 'FlightDelay' with several sub-directories and files listed. The right side shows the code editor with a file named 'part-r-00000' containing the following Java code:

```
1  ** 13439911
2  *A* 50130
3  *B* 134112
4  *C* 134112
5  *CRS_ELAPSED_TIME* 25
6  *D* 614
7  *D* 614
```

Delay Count Total each month:

The screenshot shows a Jupyter Notebook interface with the following structure:

- FlightDelay** -> **ODelayCount** -> **part-r-00000**
- Project**: .idea
- FlightDelayData**
- OAirlineCancel**
- OAirlineDelay**
- OAvgDelayMonthlyW**
- OAvgFlightDelay**
- OCancellationCount**
- OCounters**
- ODelayCount**
 - _SUCCESS.crc
 - part-r-00000.crc
 - _SUCCESS**
 - part-r-00000**
- ODelayFMedian**
- ODistinct**
- OflightCancellation**
- OnvertedIndex**
- OnvertedIndex_1**
- OJoins**
- OMaxMin**
- QMSMedian**

The notebook cell for **part-r-00000** displays the following Python code and its output:

```
1 CARRIER DELAY 1 102872882 248332
2 CARRIER DELAY 10 6647660 163924
3 CARRIER DELAY 12 3014210 102412
4 CARRIER DELAY 3 5626869 149331
5 CARRIER DELAY 4 2001177 44433
6 LATE AIRCRAFT DELAY 1 102872882 248332
7 CARRIER DELAY 7 7658045 196291
8 CARRIER DELAY 8 3952498 105062
9 CARRIER DELAY 9 1803411 41599
10 LATE AIRCRAFT DELAY 1 4218030 65962
11 LATE AIRCRAFT DELAY 19 2911846 46777
12 LATE AIRCRAFT DELAY 12 102872882 248331
13 LATE AIRCRAFT DELAY 3 2263545 38009
14 LATE AIRCRAFT DELAY 4 806516 12244
15 LATE AIRCRAFT DELAY 5 3521172 55172
16 LATE AIRCRAFT DELAY 7 3468117 52478
17 LATE AIRCRAFT DELAY 17 102872882 248330
18 LATE AIRCRAFT DELAY 9 849746 13159
19 NAS DELAY 1 5778350 157291
20 NAS DELAY 10 4156888 117590
21 NAS DELAY 12 1936329 57205
22 NAS DELAY 13 1976030 88181
23 NAS DELAY 4 1209532 113335
24 NAS DELAY 5 9595698 141258
25 NAS DELAY 7 5067076 118317
26 NAS DELAY 8 2144644 64472
27 NAS DELAY 9 1343252 33686
28 SECURITY DELAY 1 20378 764
29 SECURITY DELAY 10 18378 532
30 SECURITY DELAY 12 12948 442
31 SECURITY DELAY 3 15893 473
32 SECURITY DELAY 7 788 77
```

Delay Median based on Airline:

```
FlightDelay ~/Documents/ProjectRelated/FlightDelay
FlightDelayData
OAirlineCancel
OAirlineDelay
OAvgDelayMonthlyW
OAvgFlightDelay
OCancellationCount
OCounters
ODelayCount
ODelayFMedian
  _SUCCESS.crc
  part-r-00000.crc
    part-r-00000
    _SUCCESS
    part-r-00000
  ODistinct
  OFlightCancellation
  OInvertedIndex
  OInvertedIndex 1

ODelayCount/part-r-00000
  1 "9E" 21
  2 "AA" 16
  3 "AS" 15
  4 "B6" 17
  5 "DL" 16
  6 "EV" 17
  7 "F9" 17
  8 "G4" 17
  9 "HA" 17
  10 "KQ" 17
  11 "NK" 17
  12 "OH" 17
  13 "OO" 17
  14 "UA" 17
  15 "VX" 17
  16 "WN" 16
  17 "YY" 16
  18 "YX" 16
```

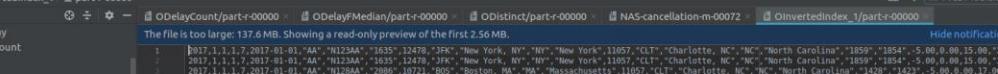
Distinct Airlines:

The screenshot shows the Apache Flink IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The left sidebar displays the project structure under 'Project' with nodes like FlightDelay, _Idea, FlightDelayData, OAirlineCancel, OAirlineDelay, OAvgDelayMonthlyW, OAvgFlightDelay, OCancellationCount, OCounters, ODelayCount, ODelayCount, ODelayMedian, ODelayFMedian, and oDistinct. Below these are sub-nodes such as _SUCCESS.crc, .part-r-00000.crc, and _SUCCESS. A specific node, 'part-r-00000', is selected and highlighted in blue. The main workspace shows a table with 19 rows and 2 columns. The first column contains integers from 1 to 19, and the second column contains airline codes: 9E, AA, AS, BB, DL, EV, FA, g4, HA, MO, NW, OH, OO, UA, VX, WH, YV, and YX.

Binning based on cancellation reason separate files:

Bloom Filter Inverted Index and Cjhaining: Based on BOS and JFK flights

Temporary Bloom Filter Data:



The screenshot shows the Microsoft Visual Studio Code interface. The top navigation bar includes 'File', 'Edit', 'View', 'Navigate', 'Code', 'Analyze', 'Refactor', 'Build', 'Run', 'Tools', 'VCS', 'Window', and 'Help'. A search bar at the top right contains the query 'FlightDelay'. The left sidebar shows a project structure for 'FlightDelay' with files like 'OFlightDelay', 'OFlightCancellation', 'OFlightCount', 'ODelayCount', 'ODelayMedian', 'ODistinct', 'OFlightCancellation', 'OInvertedIndex', 'OInvertedIndex_1', 'OSUCCESS', 'part-00000.crc', 'SUCCESS', and 'part-00000'. The main editor area displays a preview of a large file: 'The file is too large: 137.6 MB. Showing a read-only preview of the first 2.56 MB.' Below this, several tabs are open: 'ODelayCount/part-00000', 'ODelayMedian/part-00000', 'ODistinct/part-00000', 'NAS-cancellation-m-00072', 'OInvertedIndex_1/part-00000', and 'MCDMedian'. The status bar at the bottom shows 'MCDMedian' and other system information.

Joins:

```
FlightDelay 0 Joins part-r-00000
Project - ODelayCount/part-r-00000 ODelayFMedian/part-r-00000 ODistinct/part-r-00000 NAS-cancellation-m-00072 InvertedIndex_1/part-r-00000 JJoins/part-r-00000
  ▶ OAvgFlightDelay
  ▶ OCancellationCount
  ▶ OCounters
  ▶ ODelayCount
  ▶ ODelayFMedian
  ▶ ODistinct
  ▶ OFlightCancellation
  ▶ OInvertedIndex
  ▶ OInvertedIndex_1
  ▶ OJoins
    ▫ _SUCCESS.crc
    ▫ .part-r-00000.crc
    ▫ _SUCCESS
    ▫ part-r-00000
  ▶ OMaxMin
  ▶ OMSDMedian
  ▶ OSecSort
  ▶ OShuffle
  ▶ OStateCancel
  ▶ OStateDelayed
  ▶ src
```

Based on Airline Min and Max distance travelled:

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

FlightDelay > OMaxMin > part-r-00000

Project

- ODelayCount
- ODelayFMedian
- ODistinct
- OFlightCancellation
- OInvertedIndex
- OInvertedIndex_1
- OJoins
- OMaxMin
 - .SUCCESS.crc
 - .part-r-00000.crc
 - _SUCCESS
 - part-r-00000
- OMSDMedian
- OSecSort
- OShuffle
- OStateCancel
- OStateDelay6
- src
 - main
 - java
 - AirlineCancel7

ODelayCount/part-r-00000 x ODelayFMedian/part-r-00000 x ODistinct/part-r-00000 x NAS-cancellation-m-00072 x

```

1 JE 74 1416
2 AA 83 4243
3 AS 31 2874
4 B6 68 2794
5 DL 94 4983
6 EV 55 2429
7 F9 81 2446
8 G4 69 1980
9 HA 84 4983
10 MQ 83 1477
11 NK 177 2717
12 OH 55 1322
13 OO 40 1772
14 UA 67 4962
15 VX 236 2704
16 WN 137 2555
17 YV 94 1530
18 YX 67 1569
19

```

MCD Median of Distance:

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

FlightDelay > OMSDMedian > part-r-00000

Project

- ODelayCount
- ODelayFMedian
- ODistinct
- OFlightCancellation
- OInvertedIndex
- OInvertedIndex_1
- OJoins
- OMaxMin
- OMSDMedian
 - .SUCCESS.crc
 - .part-r-00000.crc
 - _SUCCESS
 - part-r-00000
- OSecSort
- OShuffle
- OStateCancel
- OStateDelay6
- src
 - main
 - java
 - AirlineCancel7

ODelayFMedian/part-r-00000 x ODistinct/part-r-00000 x NAS-cancellation-m-00072 x

```

1 GE median:422.0 Std_Div:11.182264
2 AA median:919.5 Std_Div:10.709496
3 AS median:972.5 Std_Div:19.131805
4 B6 median:992.0 Std_Div:13.007162
5 DL median:679.5 Std_Div:12.000799
6 EV median:435.0 Std_Div:8.557014
7 F9 median:978.5 Std_Div:18.91692
8 G4 median:895.0 Std_Div:20.08316
9 HA median:142.0 Std_Div:24.758072
10 MQ median:372.0 Std_Div:8.99425
11 NK median:975.0 Std_Div:13.585728
12 OH median:356.0 Std_Div:6.303177
13 OO median:398.0 Std_Div:9.116984
14 UA median:991.0 Std_Div:14.888375
15 VX median:1476.0 Std_Div:19.074
16 WN median:632.0 Std_Div:8.721499
17 YV median:485.5 Std_Div:9.903636
18 YX median:542.0 Std_Div:12.116253
19

```

Sec sort of all State and Airports in state:

Screenshot of the IntelliJ IDEA IDE showing the FlightDelay project structure and the content of part-r-00000.

Project Tree:

- FlightDelay
- OSecSort
- part-r-00000
- _SUCCESS.crc
- .part-r-00000.crc
- _SUCCESS
- part-r-00000
- Oshuffle
- OStateCancel
- OStateDelay6
- src
- main
- java
- AirlineCancel7
- AirlineDelay7

Content of part-r-00000:

| Line Number | Value |
|-------------|-------------|
| 1 | Alabama,BHM |
| 2 | Alabama,DHN |
| 3 | Alabama,HSV |
| 4 | Alabama,MGM |
| 5 | Alabama,MOB |
| 6 | Alaska,ADK |
| 7 | Alaska,ADQ |
| 8 | Alaska,AKN |
| 9 | Alaska,ANC |
| 10 | Alaska,BET |
| 11 | Alaska,BRW |
| 12 | Alaska,CDV |
| 13 | Alaska,DLG |
| 14 | Alaska,FAI |
| 15 | Alaska,GST |
| 16 | Alaska,JNU |
| 17 | Alaska,KTN |
| 18 | Alaska,OME |
| 19 | Alaska,OTZ |
| 20 | Alaska,PSG |
| 21 | Alaska,SCC |
| 22 | Alaska,SIT |
| 23 | Alaska,TKI |
| 24 | Alaska,WRG |
| 25 | Alaska,YAK |
| 26 | Arizona,AZA |
| 27 | Arizona,FLG |
| 28 | Arizona,IFP |
| 29 | Arizona,PHX |
| 30 | Arizona,PRC |
| 31 | Arizona,TUS |
| 32 | Arizona,YUM |

Shuffle Data:

Screenshot of the IntelliJ IDEA IDE showing the FlightDelay project structure and the content of part-r-00000.

Project Tree:

- FlightDelay
- Oshuffle
- part-r-00000
- _SUCCESS.crc
- .part-r-00000.crc
- _SUCCESS
- part-r-00000
- OdelayCount
- OdelayMedian
- ODistinct
- OFlightCancellation
- OInvertedIndex
- OInvertedIndex_1
- OJoins
- OMaxMin
- OMSDMedian
- OSSort
- OSecSort
- src
- main
- java

Content of part-r-00000:

The file is too large: 3.54 GB. Showing a read-only preview of the first 2.56 MB.

| Line Number | Value |
|-------------|---|
| 1 | 2018-2-5,20,7,2018-05-20,"YY","N434YK","4426",1383,"MIA","Miami, FL","FL","Florida",10693,"BNA","Nashville, TN","TN","Tennessee",1530,"1528",-2,0,0,0,18,00,"1634",84,00,"1655" |
| 2 | 2018-1,3,19,1,2018-03-19,"B6","N637JB","500",1125,"DAB","Daytona Beach, FL","FL","Florida",12478,"JFK","New York, NY","NY","New York",1132,"1127",-5,0,0,0,13,00,"1340",3,00,"1442" |
| 3 | 2017-2,5,2,2,2017-05-02,"AA","N391AA","5",11298,"DFW","Dallas/Fort Worth, TX","TX","Texas",12173,"HNL","Honolulu, HI","HI","Hawaii",1110,"1219",69,00,69,00,18,00,"1519",3,00,"1442" |
| 4 | 2017-4,16,17,2,2017-10-17,"UA","N24211","698",11264,"IAD","Washington, DC","VA","Virginia",11292,"DEN","Denver, CO","CO","Colorado",1445,"1457",12,00,12,00,13,00,"1626",9,00,"1710" |
| 5 | 2017-4,16,17,2,2017-10-17,"UA","N24211","698",11264,"IAD","Washington, DC","VA","Virginia",11292,"DEN","Denver, CO","CO","Colorado",1445,"1457",12,00,12,00,13,00,"1626",9,00,"1710" |
| 6 | 2017-4,16,7,6,2017-10-07,"000","N08859","5217",11292,"DEN","Denver, CO","CO","Colorado",11637,"FAR","Fargo, ND","ND","North Dakota",1525,"1518",-7,00,0,0,13,00,"1753",3,00,"1820" |
| 7 | 2018-1,1,7,7,2018-01-07,"WN","N402WN","4346",14869,"SLC","Salt Lake City, UT","UT","Utah",11292,"DEN","Denver, CO","CO","Colorado",1540,"1603",23,00,23,13,00,"1712",5,00,"1700" |
| 8 | 2017-1,1,17,2,2017-01-17,"WN","N7847A","125",14771,"SFO","San Francisco, CA","CA","California",12889,"LAS","Las Vegas, NV","NV","Nevada",1340,"1341",4,00,0,0,10,00,"1458",5,00,"1500" |
| 9 | 2018-2,5,19,1,2018-03-19,"B6","N637JB","500",1125,"DAB","Daytona Beach, FL","FL","Florida",12478,"JFK","New York, NY","NY","New York",1132,"1127",-5,0,0,0,13,00,"1340",3,00,"1442" |
| 10 | 2018-2,5,19,4,2018-05-10,"WN","N4211V","5814",14771,"SFO","San Francisco, CA","CA","California",12892,"LAS","Las Vegas, NV","NV","Nevada",1340,"1341",4,00,0,0,10,00,"1458",5,00,"1500" |
| 11 | 2018-4,16,23,2,2018-10-23,"AA","N524UN","475",13930,"ORD","Chicago, IL","IL","Illinois",14107,"PHX","Phoenix, AZ","AZ","Arizona",8710,"0717",7,00,7,00,11,00,"0848",4,00,"0901","08" |
| 12 | 2018-1,1,23,2,2018-01-23,"VX","N833VA","1165",111618,"EWR"," Newark, NJ","NJ","New Jersey",12892,"LAX","Los Angeles, CA","CA","California",1230,"1222",-8,00,0,0,19,00,"1506",10,00,"1000" |
| 13 | 2017-7,5,3,2,2017-05-03,"AA","N982AA","697",12889,"LAX","Los Angeles, CA","CA","California",1230,"1222",3,00,0,0,19,00,"1506",10,00,"1000" |
| 14 | 2018-1,1,23,2,2018-01-23,"VX","N833VA","1165",111618,"EWR"," Newark, NJ","NJ","New Jersey",12892,"LAX","Los Angeles, CA","CA","California",1230,"1222",3,00,0,0,19,00,"1506",10,00,"1000" |
| 15 | 2018-1,3,22,4,2018-03-22,"AA","N564W","2774",11298,"DFW","Dallas/Fort Worth, TX","TX","Texas",14673,"SAN","San Diego, CA","CA","California",1655,"1640",-6,00,0,0,18,00,"1753",3,00,"1820" |
| 16 | 2018-2,5,1,2,2018-05-01,"OL","N0390W","2465",10529,"BOL","Hartford, CT","CT","Connecticut",10397,"ATL","Atlanta, GA","GA","Georgia",1540,"1544",4,00,4,0,10,00,"1748",4,00,"1820" |
| 17 | 2017-4,16,7,6,2017-10-07,"B6","N637JB","697",10721,"BOS","Boston, MA","MA","Massachusetts",10397,"ATL","Atlanta, GA","GA","Georgia",1354,"1355",10,00,0,0,15,00,"1619",9,00,"1644" |
| 18 | 2018-2,5,19,1,2018-03-19,"B6","N637JB","500",1125,"DAB","Daytona Beach, FL","FL","Florida",12478,"JFK","New York, NY","NY","New York",1132,"1127",-5,0,0,0,13,00,"1340",3,00,"1442" |
| 19 | 2018-2,5,12,6,2018-05-12,"AA","N24211","4745",14745,"IAD","Washington, DC","VA","Virginia",11292,"DEN","Denver, CO","CO","Colorado",1445,"1457",12,00,12,00,13,00,"1626",9,00,"1710" |
| 20 | 2018-4,16,23,2,2018-10-23,"AA","N537PV","1524",11298,"DFW","Dallas/Fort Worth, TX","TX","Texas",14980,"SNA","Santa Ana, CA","CA","California",1105,"1103",10,00,0,0,20,00,"1540",9,00,"1604",1548 |
| 21 | 2017-2,5,13,6,2017-05-13,"WN","N833VA","4928",14679,"SAM","San Diego, CA","CA","California",10937,"ATL","Atlanta, GA","GA","Georgia",1110,"1112",14,00,14,00,12,00,"1830",7,00,"1820" |
| 22 | 2018-1,1,1,2018-01-01,"000","N6938R","3491",14745,"IAD","Washington, DC","VA","Virginia",11292,"DEN","Denver, CO","CO","Colorado",1445,"1457",12,00,12,00,13,00,"1626",9,00,"1710" |
| 23 | 2018-1,1,1,2018-01-17,"VX","N833VA","1165",111618,"EWR"," Newark, NJ","NJ","New Jersey",12892,"LAX","Los Angeles, CA","CA","California",1230,"1222",3,00,0,0,19,00,"1506",10,00,"1000" |
| 24 | 2018-2,5,17,4,2018-05-17,"WN","N4208W","1732",13342,"MKE","Milwaukee, WI","WI","Wisconsin",12933,"LGA","New York, NY","NY","New York",0620,"0617",-3,00,0,0,18,00,"0913",11,00,"09" |
| 25 | 2019-1,1,18,5,2019-01-18,"UA","N658WA","510",12892,"LAX","Los Angeles, CA","CA","California",11618,"EWR"," Newark, NJ","NJ","New Jersey",1115,"1309",-6,00,0,0,18,00,"2107",6,00,"2107" |
| 26 | 2017-4,12,1,5,2017-12-01,"EV","N7848V","5393",11433,"DTW","Detroit, MI","MI","Michigan",14524,"RIC","Richmond, VA","VA","Virginia",1540,"1537",-3,00,0,0,19,00,"1766",11,00,"1732" |
| 27 | 2018-1,1,27,6,2018-01-27,"VX","N658WA","510",12892,"LAX","Los Angeles, CA","CA","California",11618,"EWR"," Newark, NJ","NJ","New Jersey",1115,"1309",-6,00,0,0,18,00,"2107",6,00,"2107" |
| 28 | 2018-2,5,10,4,2018-05-10,"AS","N653VA","1756",16747,"SEA","Seattle, WA","WA","Washington",14771,"SFO","San Francisco, CA","CA","California",1055,"1059",4,00,4,0,14,00,"2047",6,00,"2047" |

Statewise Cancelled Flights Monthly count:

FlightDelay

OStateCancel

part-r-00000

Project

- ODelayCount
- ODelayFMedian
- ODDistinct
- OFlightCancellation
- OInvertedIndex
- OInvertedIndex_1
- OJoins
- OMaxMin
- OMSDMedian
- OSSort
- OShuffle
- OStateCancel
 - _SUCCESS.crc
 - .part-r-00000.crc
 - _SUCCESS
 - part-r-00000
- OStateDelay6
- src
 - main
 - java
 - AirlineCancel7

Oshuffle/part-r-00000

OStateCancel/part-r-00000

| 1 | Alabama A | 1 | 68 |
|----|-----------|----|-----|
| 2 | Alabama A | 10 | 54 |
| 3 | Alabama A | 12 | 24 |
| 4 | Alabama A | 3 | 23 |
| 5 | Alabama A | 4 | 25 |
| 6 | Alabama A | 5 | 24 |
| 7 | Alabama A | 7 | 32 |
| 8 | Alabama A | 8 | 8 |
| 9 | Alabama A | 9 | 17 |
| 10 | Alabama A | 1 | 294 |
| 11 | Alabama B | 10 | 46 |
| 12 | Alabama B | 12 | 89 |
| 13 | Alabama B | 3 | 57 |
| 14 | Alabama B | 4 | 27 |
| 15 | Alabama B | 5 | 28 |
| 16 | Alabama B | 7 | 55 |
| 17 | Alabama B | 8 | 164 |
| 18 | Alabama B | 9 | 49 |
| 19 | Alabama C | 1 | 41 |
| 20 | Alabama C | 10 | 8 |
| 21 | Alabama C | 12 | 25 |
| 22 | Alabama C | 3 | 14 |
| 23 | Alabama C | 4 | 9 |
| 24 | Alabama C | 5 | 24 |
| 25 | Alabama C | 7 | 45 |
| 26 | Alabama C | 8 | 22 |
| 27 | Alabama C | 9 | 9 |
| 28 | Alaska A | 1 | 90 |
| 29 | Alaska A | 10 | 64 |
| 30 | Alaska A | 12 | 40 |

Statewise delayed flights:

FlightDelay [-~/Documents/ProjectsRelated/FlightDelay] - .../OStateDelay6/part-r-00000 [F]

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

FlightDelay

OStateDelay6

part-r-00000

Project

- ODelayCount
- ODelayFMedian
- ODDistinct
- OFlightCancellation
- OInvertedIndex
- OInvertedIndex_1
- OJoins
- OMaxMin
- OMSDMedian
- OSSort
- OShuffle
- OStateCancel
- OStateDelay6
 - _SUCCESS.crc
 - .part-r-00000.crc
 - _SUCCESS
 - part-r-00000
- src
 - main
 - java
 - AirlineCancel7

Oshuffle/part-r-00000

OStateCancel/part-r-00000

OStateDelay6/part-r-00000

| 1 | Alabama CARRIER DELAY | 1 | 55878 862 |
|----|-----------------------------|----|-----------|
| 2 | Alabama CARRIER DELAY | 10 | 30210 536 |
| 3 | Alabama CARRIER DELAY | 12 | 16173 375 |
| 4 | Alabama CARRIER DELAY | 3 | 25930 493 |
| 5 | Alabama CARRIER DELAY | 4 | 8805 117 |
| 6 | Alabama CARRIER DELAY | 5 | 43096 778 |
| 7 | Alabama CARRIER DELAY | 7 | 28090 682 |
| 8 | Alabama CARRIER DELAY | 8 | 10482 204 |
| 9 | Alabama CARRIER DELAY | 9 | 11384 218 |
| 10 | Alabama LATE AIRCRAFT DELAY | 1 | 34208 560 |
| 11 | Alabama LATE AIRCRAFT DELAY | 10 | 20704 290 |
| 12 | Alabama LATE AIRCRAFT DELAY | 12 | 12509 193 |
| 13 | Alabama LATE AIRCRAFT DELAY | 3 | 17893 293 |
| 14 | Alabama LATE AIRCRAFT DELAY | 4 | 7225 68 |
| 15 | Alabama LATE AIRCRAFT DELAY | 5 | 22788 334 |
| 16 | Alabama LATE AIRCRAFT DELAY | 7 | 25322 403 |
| 17 | Alabama LATE AIRCRAFT DELAY | 8 | 9210 154 |
| 18 | Alabama LATE AIRCRAFT DELAY | 9 | 8622 117 |
| 19 | Alabama NAS DELAY | 1 | 21131 593 |
| 20 | Alabama NAS DELAY | 10 | 17718 538 |
| 21 | Alabama NAS DELAY | 12 | 12765 312 |
| 22 | Alabama NAS DELAY | 3 | 10050 390 |
| 23 | Alabama NAS DELAY | 4 | 4429 87 |
| 24 | Alabama NAS DELAY | 5 | 22574 534 |
| 25 | Alabama NAS DELAY | 7 | 23248 557 |
| 26 | Alabama NAS DELAY | 8 | 4882 182 |
| 27 | Alabama NAS DELAY | 9 | 7791 190 |
| 28 | Alabama SECURITY DELAY | 1 | 2 1 |
| 29 | Alabama SECURITY DELAY | 12 | 16 1 |
| 30 | Alabama SECURITY DELAY | 7 | 18 2 |
| 31 | Alabama SECURITY DELAY | 9 | 3 1 |
| 32 | Alabama WEATHER DELAY | 1 | 6147 01 |

Top10 airline with most flights Pig:

Open

topk.pig

/usr/local/pig-0.17.0/bin/top10Carriers

part-r-00000

| WN | |
|----|--------|
| WN | 112223 |
| DL | 77046 |
| AA | 73656 |
| OO | 56210 |
| UA | 45541 |
| EV | 30789 |
| B6 | 25191 |
| AS | 15261 |
| NK | 12544 |
| F9 | 7801 |

```

topk.pig

flights = LOAD 'apr17.csv' using PigStorage(',');
grouped = GROUP flights BY $6;
summed = FOREACH grouped GENERATE group, COUNT(flights) AS cntd;
sorted = ORDER summed BY cntd DESC;
top25 = LIMIT sorted 10;
Dump top25;
STORE top25 into 'top10Carriers';

```

Top Distance values:

```

Google Chrome - FlightDelay - OTopDistanceValues part-r-00000
FlightDelay - OTopDistanceValues part-r-00000
Project ▾
  ▷ AirlineTextPair.java
  ▷ TopKReducer.java
  ▷ TopMapper.java
  ▷ DelayCountMapper.java
  ▷ App.java
  ▷ part-r-00000
  ▷ _SUCCESS
  ▷ .part-r-00000.crc
  ▷ _SUCCESS5
  ▷ part-r-00000
src
  ▷ main
    ▷ java
      ▷ AirlineCancel7
      ▷ AirlineDelay7
      ▷ AverageFlightDelay
      ▷ AvgDelayMonthly
      ▷ BinningFlight
      ▷ PerClassificationPoints
Run: App

```

FlightDelay - OTopDistanceValues part-r-00000

```

2018-1-3-14, 2018-03-01, F9,N263FR,1141,18423,AUS,"Austin, TX",TX,Texas,12889,LAS,"Las Vegas, NV",Nevada,1421,1416,-5,0,12,1585,6,1525,1511,-14,0,0,0,184,175,157,1,1890,5,...,
2018-1-3-14, 2018-03-01, F9,N263FR,144,14747,SEA,"Seattle, WA",WA,Washington,11292,DEN,"Denver, CO",CO,Colorado,1514,1621,67,67,11,1848,6,1855,1848,53,53,0,0,163,167,128,1,1824,5,0,0,0,53
2018-1-3-14, 2018-03-01, F9,N263FR,1299,15304,TPA,"Tampa, FL",FL,Florida,13342,MKE,"Milwaukee, WI",WI,Wisconsin,1825,1939,74,74,2125,5,2017,2130,73,73,0,0,172,171,149,1,1875,5,16,8,0,0,57
2018-1-3-14, 2018-03-01, F9,N263FR,1141,18423,AUS,"Austin, TX",TX,Texas,12889,LAS,"Las Vegas, NV",Nevada,1421,1416,-5,0,12,1585,6,1525,1511,-14,0,0,0,184,175,157,1,1890,5,...,
2018-1-3-14, 2018-03-01, F9,N263FR,1296,13487,MSP,"Minneapolis-St. Paul, MN",MN,Minnesota,15304,TPA,"Tampa, FL",FL,Florida,784,700,-4,0,50,112,8,1120,1140,20,20,0,0,150,220,152,1,1596,6,0,0,26,0,0
2018-1-3-14, 2018-03-01, F9,N263FR,1099,13342,MKE,"Milwaukee, WI",WI,Wisconsin,12889,LAS,"Las Vegas, NV",Nevada,2107,2214,67,67,19,6,7,2759,13,74,74,0,0,232,239,213,1,1524,7,0,0,7,0,67
2018-1-3-14, 2018-03-01, F9,N263FR,681,13284,MKO,"Orlando, FL",FL,Florida,11292,DEN,"Denver, CO",CO,Colorado,980,1108,128,128,1302,8,1115,1310,115,115,0,0,255,242,221,1,1546,7,4,0,0,0,111
2018-1-3-14, 2018-03-01, F9,N263FR,1977,12359,IND,"Indianapolis, IN",IN,Indiana,12889,LAS,"Las Vegas, NV",Nevada,1259,1386,7,7,34,1424,9,1414,1433,19,19,0,0,255,267,224,1,1596,7,7,0,12,0,0
2018-1-3-14, 2018-03-01, F9,N263FR,1680,12889,LAS,"Las Vegas, NV",Nevada,12104,MKO,"Orlando, FL",FL,Florida,30,221,111,111,18,181,8,786,1089,142,147,0,0,266,246,272,1,2039,8,24,0,32,0,31

```

MC Median of Airliner distance:

```

FlightDelay - OMSMedian part-r-00000
FlightDelay - OMSMedian part-r-00000
Project ▾
  ▷ AirlineTextPair.java
  ▷ TopKReducer.java
  ▷ TopMapper.java
  ▷ DelayCountMapper.java
  ▷ App.java
  ▷ OTopDistanceValues/part-r-00000
  ▷ OMSMedian/part-r-00000
  ▷ _SUCCESS
  ▷ .part-r-00000.crc
  ▷ _SUCCESS
  ▷ part-r-00000
src
  ▷ main
    ▷ java
      ▷ AirlineCancel7
      ▷ AirlineDelay7
      ▷ AverageFlightDelay
      ▷ AvgDelayMonthly
      ▷ BinningFlight
      ▷ PerClassificationPoints
Run: App

```

FlightDelay - OMSMedian part-r-00000

```

2018-1-3-14, 2018-03-01, F9,N263FR,1141,18423,AUS,"Austin, TX",TX,Texas,12889,LAS,"Las Vegas, NV",Nevada,1421,1416,-5,0,12,1585,6,1525,1511,-14,0,0,0,184,175,157,1,1890,5,...,
2018-1-3-14, 2018-03-01, F9,N263FR,144,14747,SEA,"Seattle, WA",WA,Washington,11292,DEN,"Denver, CO",CO,Colorado,1514,1621,67,67,11,1848,6,1855,1848,53,53,0,0,163,167,128,1,1824,5,0,0,0,53
2018-1-3-14, 2018-03-01, F9,N263FR,1299,15305,TPA,"Tampa, FL",FL,Florida,13342,MKE,"Milwaukee, WI",WI,Wisconsin,1825,1939,74,74,2125,5,2017,2130,73,73,0,0,172,171,149,1,1875,5,16,8,0,0,57
2018-1-3-14, 2018-03-01, F9,N263FR,1141,18423,AUS,"Austin, TX",TX,Texas,12889,LAS,"Las Vegas, NV",Nevada,1421,1416,-5,0,12,1585,6,1525,1511,-14,0,0,0,184,175,157,1,1890,5,...,
2018-1-3-14, 2018-03-01, F9,N263FR,1296,13487,MSP,"Minneapolis-St. Paul, MN",MN,Minnesota,15305,TPA,"Tampa, FL",FL,Florida,784,700,-4,0,50,112,8,1120,1140,20,20,0,0,150,220,152,1,1596,6,0,0,26,0,0
2018-1-3-14, 2018-03-01, F9,N263FR,1099,13342,MKE,"Milwaukee, WI",WI,Wisconsin,12889,LAS,"Las Vegas, NV",Nevada,2107,2214,67,67,19,6,7,2759,13,74,74,0,0,232,239,213,1,1524,7,0,0,7,0,67
2018-1-3-14, 2018-03-01, F9,N263FR,681,13284,MKO,"Orlando, FL",FL,Florida,11292,DEN,"Denver, CO",CO,Colorado,980,1108,128,128,1302,8,1115,1310,115,115,0,0,255,242,221,1,1546,7,4,0,0,0,111
2018-1-3-14, 2018-03-01, F9,N263FR,1977,12359,IND,"Indianapolis, IN",IN,Indiana,12889,LAS,"Las Vegas, NV",Nevada,1259,1386,7,7,34,1424,9,1414,1433,19,19,0,0,255,267,224,1,1596,7,7,0,12,0,0
2018-1-3-14, 2018-03-01, F9,N263FR,1680,12889,LAS,"Las Vegas, NV",Nevada,12104,MKO,"Orlando, FL",FL,Florida,30,221,111,111,18,181,8,786,1089,142,147,0,0,266,246,272,1,2039,8,24,0,32,0,31

```

Merge: Example with small file

Pig:

Latest Data:

| | | | | | | | | | | | | | | | part-n-00000 | /usr/local/pig-0.17.0/bin/Lat |
|------|---|---|---|---|------------|----|--------|------|-------|--------|------|--|--|--|--------------|-------------------------------|
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N201FR | 1680 | 12889 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N201FR | 681 | 13204 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N201FR | 681 | 11292 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N201FR | 144 | 14747 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N201FR | 122 | 11292 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N202FR | 1138 | 12889 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N202FR | 1141 | 10423 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N202FR | 1106 | 12889 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N202FR | 1111 | 12266 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N203FR | 1236 | 13487 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N203FR | 1201 | 15304 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N203FR | 1204 | 12339 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N203FR | 1209 | 15304 | Latest | Data | | | | | |
| 2018 | 1 | 3 | 1 | 4 | 2018-03-01 | F9 | N203FR | 1099 | 13342 | Latest | Data | | | | | |

Pig Top 10:



| | Open ▾ |  |
|----|--------|---|
| WN | 112223 | |
| DL | 77046 | |
| AA | 73656 | |
| OO | 56210 | |
| UA | 45541 | |
| EV | 30789 | |
| B6 | 25191 | |
| AS | 15261 | |
| NK | 12544 | |
| F9 | 7801 | |