Kinnar Rajeshkumar Kansara

NUID: 001388213

Congestion in the sky

INFO7250 Engineering of Big-Data Systems Final Project Report

Contents

[1. Overview 3](#_Toc27236143)

[2. Dataset Details 4](#_Toc27236144)

[2.1. Variable Descriptions 4](#_Toc27236145)

[3. Analysis of Flights using Hadoop MapReduce 5](#_Toc27236146)

[3.1. Getting top 10 source destination airport pairs 5](#_Toc27236147)

[3.2. Unique Carriers Names with Flights Count 5](#_Toc27236148)

[3.3. Year wise flight delay (> 15 minutes) and cancellation. Counts and Ratio 6](#_Toc27236149)

[3.4. Flight delay and cancellation by carrier including carrier names 6](#_Toc27236150)

[3.5. Date wise flight delay (> 15 minutes) and cancellation. Counts and Ratio 7](#_Toc27236151)

[3.6. Average distance covered and airtime done by each carrier 9](#_Toc27236152)

[3.7. Recommendation System using RMS (Root Mean Square) 10](#_Toc27236153)

[4. Analysis of flights using Pig on Hadoop 12](#_Toc27236154)

[4.1. Top 10 cities by total traffic of flights 12](#_Toc27236155)

[4.2. Popular carriers 15](#_Toc27236156)

[4.3. Most busy routes (descending order) 16](#_Toc27236157)

[5. Analysis on Flights using Hive on Hadoop 17](#_Toc27236158)

[5.1. Create FlightSchema and setup the parameters 17](#_Toc27236159)

[5.2. Create table flights and load data from hdfs path 17](#_Toc27236160)

[5.3. Create table airports and load data from hdfs path 18](#_Toc27236161)

[5.4. Create table carriers and load data from hdfs path 18](#_Toc27236162)

[5.5. Find flights which travelled more than 500 airmiles 18](#_Toc27236163)

[5.6. Find flights which arrive and depart on time 18](#_Toc27236164)

[5.7. Get count of flights for each carrier 19](#_Toc27236165)

[5.8. Find origin and destination pairs from DEN airport from 2008 data 20](#_Toc27236166)

[6. Graphical Representation and Analysis 22](#_Toc27236167)

[6.1. % of flight departures delayed > 15 minutes – daily basis 22](#_Toc27236168)

[6.2. % of flight cancelled – daily basis 24](#_Toc27236169)

[6.3. Flight trend on daily basis 25](#_Toc27236170)

[6.4. Delayed flights trend on daily basis 27](#_Toc27236171)

[6.5. Flight delay history by year 27](#_Toc27236172)

[6.6. Total flights vs delayed flights for each carrier 28](#_Toc27236173)

[6.7. Tabular view of top 10 most popular source destination pairs 28](#_Toc27236174)

[7. Lessons Learned & tips for travellers 29](#_Toc27236175)

[8. Challenges 29](#_Toc27236176)

[9. Future Scope 29](#_Toc27236177)

[10. References 29](#_Toc27236178)

[11. Appendix 30](#_Toc27236179)

[11.1. Getting top 10 source destination airport pairs 30](#_Toc27236180)

[11.2. Unique Carriers Names with Flights Count 34](#_Toc27236181)

[11.3. Year wise flight delay (> 15 minutes) and cancellation. Counts and Ratio 39](#_Toc27236182)

[11.4. Flight delay and cancellation by carrier including carrier names 43](#_Toc27236183)

[11.5. Date wise flight delay (> 15 minutes) and cancellation. Counts and Ratio 50](#_Toc27236184)

[11.6. Average distance covered and airtime done by each carrier 55](#_Toc27236185)

[11.7. Recommendation System using RMS (Root Mean Square) 60](#_Toc27236186)

# Overview

The dataset I have chosen is Flight Arrival Data from stat-computing.org (http://stat-computing.org/dataexpo/2009/the-data.html)

The dataset has various columns as described in the following sections. The main reason to select this dataset is the number of columns, data distribution by years and other small datasets which give information about carriers, airports and aircrafts. This will help me in performing MapReduce in efficient manner with the help of Java MapReduce, Hive and Pig.

In this project, the data I am using is of 10 years which is 1999-2008. Due to local disk space constraints, I avoided the previous 10 years’ data. By going through the csv pattern, I observed that I can use 1999-2008 data for proper analysis. All csv files combined makes up 12.6 GB of disk space.

Project Code Repository URL: <https://github.com/kinnarrk/INFO7250BigData>

# Dataset Details

## Variable Descriptions

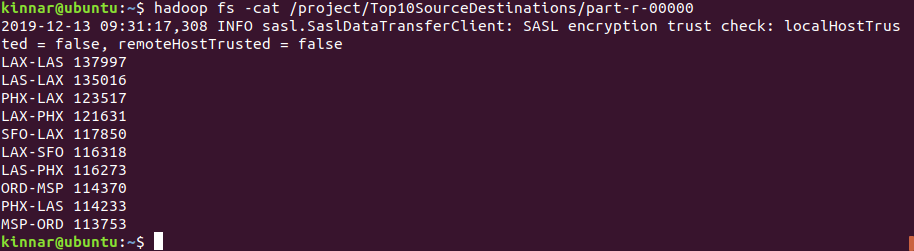
|  |  |  |
| --- | --- | --- |
| # | Name | Description |
| 1 | Year | 1987-2008 |
| 2 | Month | 1-12 |
| 3 | DayofMonth | 1-31 |
| 4 | DayOfWeek | 1 (Monday) - 7 (Sunday) |
| 5 | DepTime | actual departure time (local, hhmm) |
| 6 | CRSDepTime | scheduled departure time (local, hhmm) |
| 7 | ArrTime | actual arrival time (local, hhmm) |
| 8 | CRSArrTime | scheduled arrival time (local, hhmm) |
| 9 | UniqueCarrier | [unique carrier code](http://stat-computing.org/dataexpo/2009/supplemental-data.html) |
| 10 | FlightNum | flight number |
| 11 | TailNum | plane tail number |
| 12 | ActualElapsedTime | in minutes |
| 13 | CRSElapsedTime | in minutes |
| 14 | AirTime | in minutes |
| 15 | ArrDelay | arrival delay, in minutes |
| 16 | DepDelay | departure delay, in minutes |
| 17 | Origin | origin [IATA airport code](http://stat-computing.org/dataexpo/2009/supplemental-data.html) |
| 18 | Dest | destination [IATA airport code](http://stat-computing.org/dataexpo/2009/supplemental-data.html) |
| 19 | Distance | in miles |
| 20 | TaxiIn | taxi in time, in minutes |
| 21 | TaxiOut | taxi out time in minutes |
| 22 | Cancelled | was the flight cancelled? |
| 23 | CancellationCode | reason for cancellation (A = carrier, B = weather, C = NAS, D = security) |
| 24 | Diverted | 1 = yes, 0 = no |
| 25 | CarrierDelay | in minutes |
| 26 | WeatherDelay | in minutes |
| 27 | NASDelay | in minutes |
| 28 | SecurityDelay | in minutes |
| 29 | LateAircraftDelay | in minutes |

# Analysis of Flights using Hadoop MapReduce

## Getting top 10 source destination airport pairs

This analysis gives us the basic understanding about busiest source and destination pairs. This gives us the idea about how flight booking trend is and if we want to avoid rush by choosing alternate source or destination airport. Job chaining has been carried out to get this information.

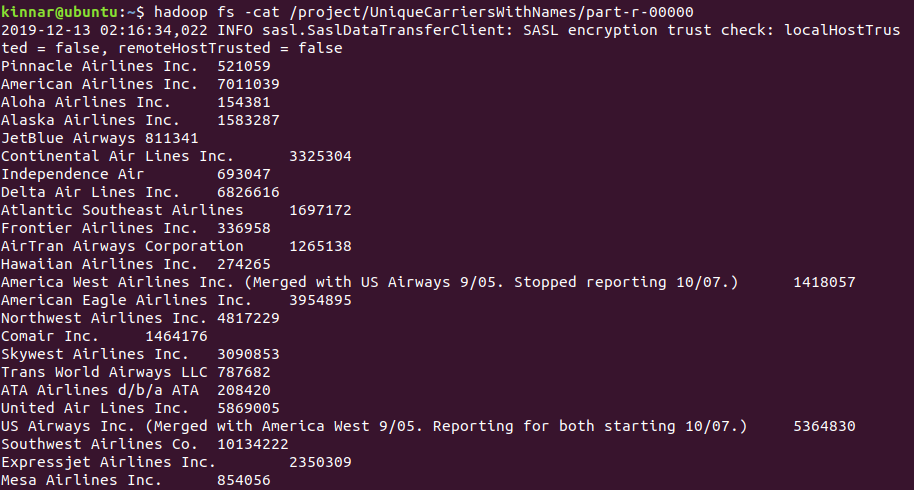
hadoop jar ~/Desktop/proj\_jars/finalproject.jar com.kinnar.bigdataproject.top10\_busy\_airports.TopNApp /project/input/years /project/ Top10SourceDestinations



## Unique Carriers Names with Flights Count

By doing this, we can get all the carriers with their names from different file which is carriers.csv. I have performed reducer side join to achieve this. This is divided in two parts: first get the flight count of each carrier and then perform join to get the carrier names.

hadoop jar ~/Desktop/proj\_jars/finalproject.jar com.kinnar.bigdataproject.unique\_carrier\_names.CarriersApp /project/input/years /project

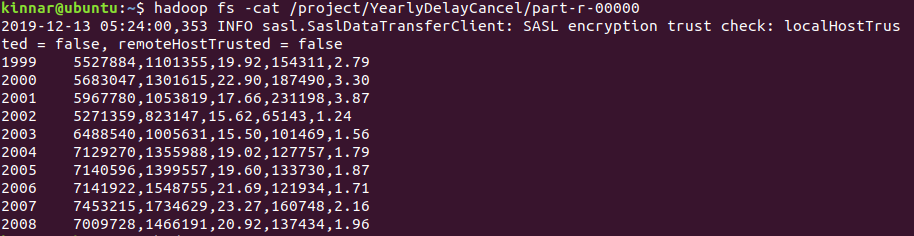


## Year wise flight delay (> 15 minutes) and cancellation. Counts and Ratio

This will give us the idea about the historical data. It gives us information that which years were good and worse for the travelling. By combining weather conditions, we can do better analytics

hadoop jar ~/Desktop/proj\_jars/finalproject.jar com.kinnar.bigdataproject.yearly\_delay.YearlyDelayApp /project/input/years /project/YearlyDelayCancel

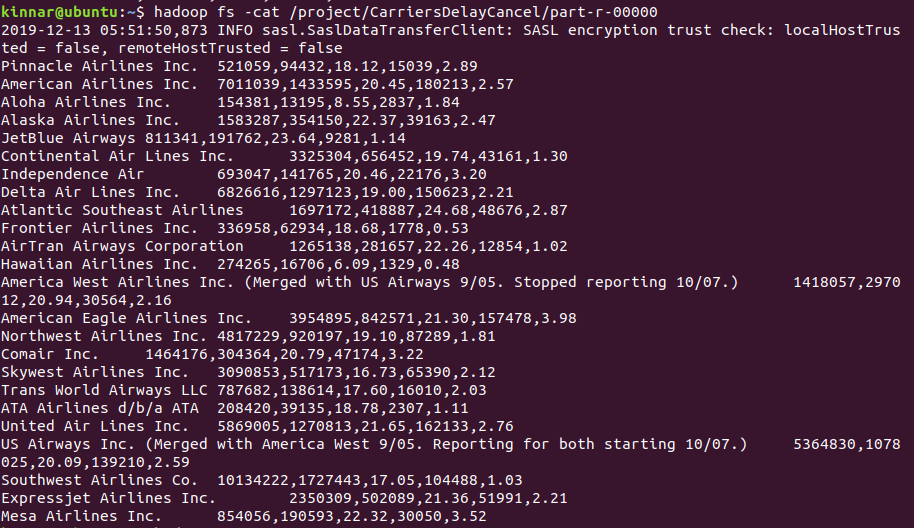
The fields contained in the value part are: flightsCount, delayedFlightsCount, delayPercentage, canceledFlightsCount, canceledPercentage. The labels are removed for the sake of easy extraction while doing graphical representation.



## Flight delay and cancellation by carrier including carrier names

This is important to analyse which carrier creates more delay and gives us important information about choosing the flights which are on time. For this analysis, showing carrier name is important so I have carried out reducer side join. Also, job chaining is needed for the analysis so after chaining, join has been carried out to show carrier information.

hadoop jar ~/Desktop/proj\_jars/finalproject.jar com.kinnar.bigdataproject.carrier\_delay\_cancel.CarrierDelayCancelApp /project/input/years /project/input/carriers.csv /project/CarriersDelayCancel

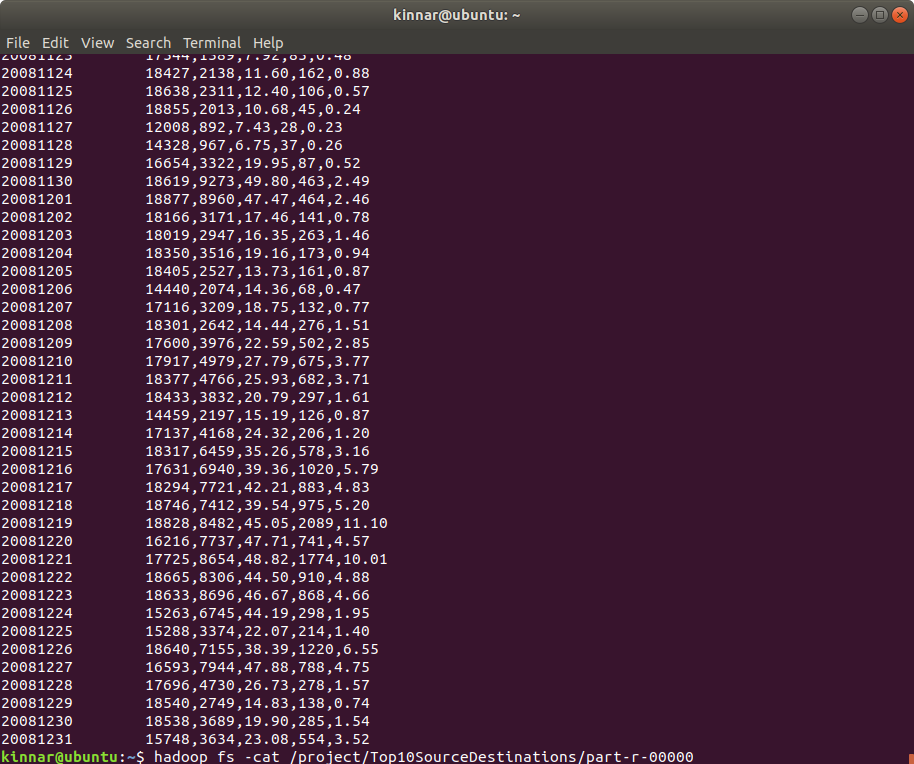
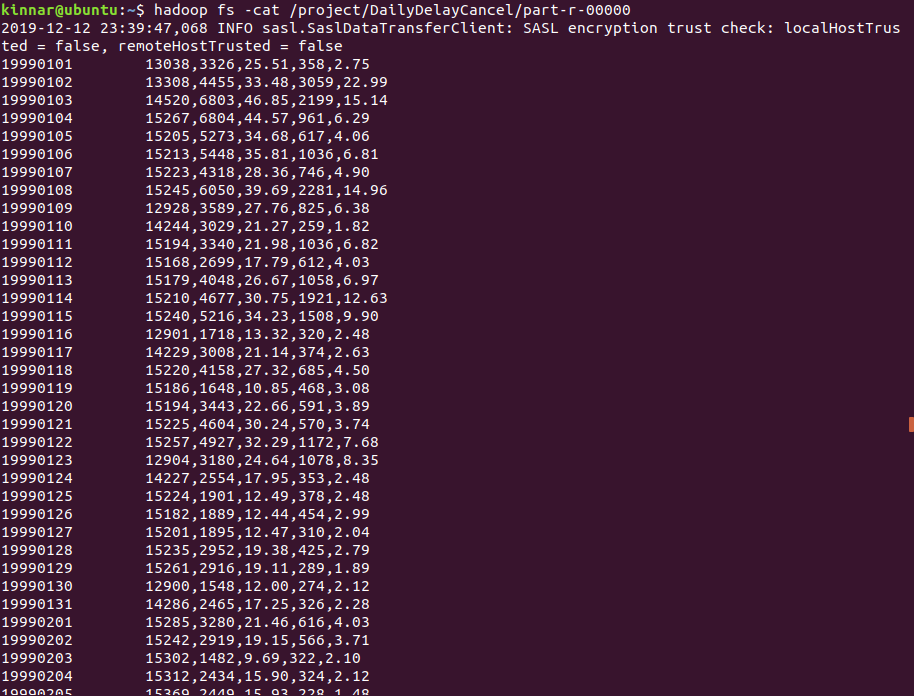


## Date wise flight delay (> 15 minutes) and cancellation. Counts and Ratio

This MapReduce is one of the most important part of the analysis. For graphical representation, this was the key to see so many patterns in the flight delay and cancellation. This gives us clear picture about the historical data and how we can take a good decision about when to travel and what time should be avoided. Although, this MapReduce is not so difficult to carry out, but the data it can represent is very important. Graphical representation will be explained in the later part of the report.

hadoop jar ~/Desktop/proj\_jars/finalproject.jar com.kinnar.bigdataproject.daily\_delay\_cancel.DailyDelayCancelApp /project/input/years /project/DailyDelayCancel

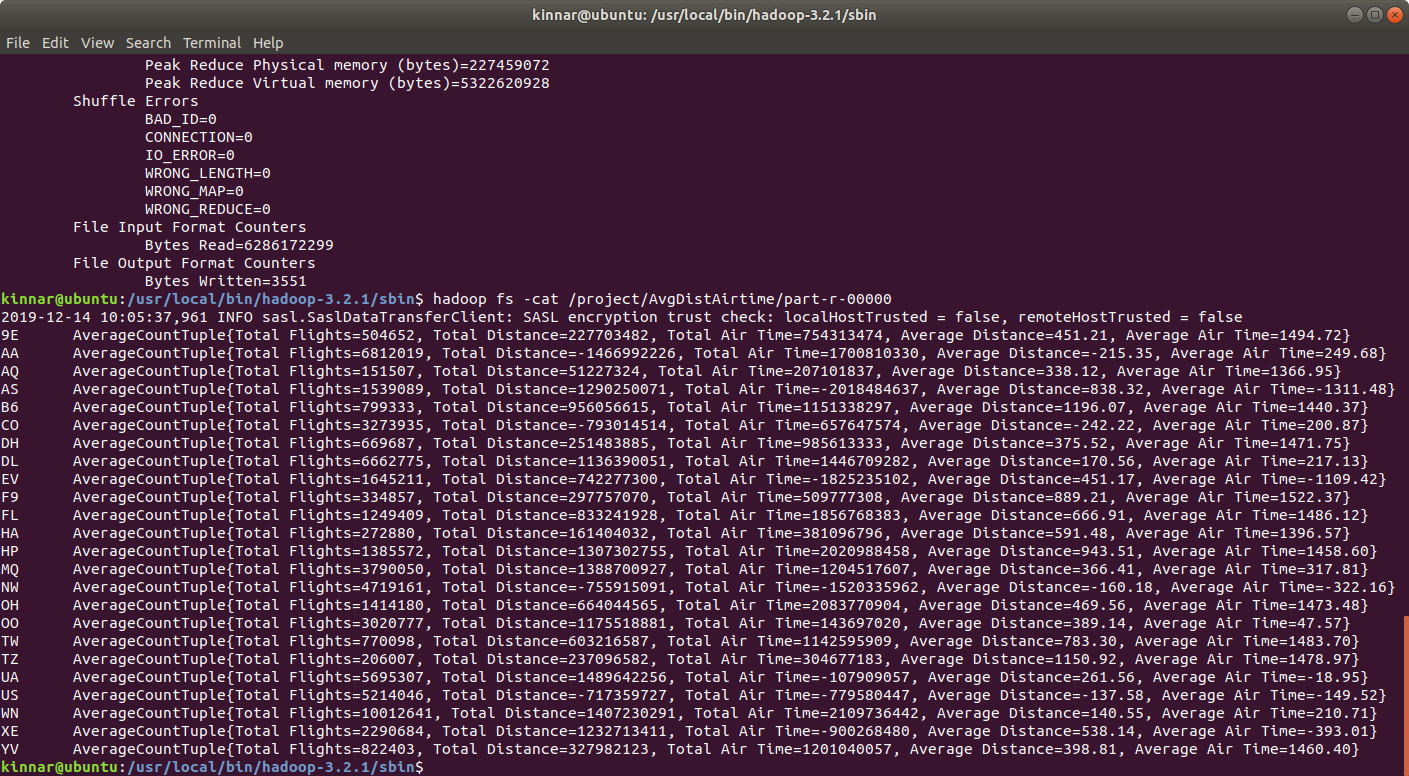
Same as year wise details, the fields contained in the value part are: flightsCount, delayedFlightsCount, delayPercentage, canceledFlightsCount, canceledPercentage. The labels are removed for the sake of easy extraction while doing graphical representation.



## Average distance covered and airtime done by each carrier

This analysis is useful for viewing carrier reach and capacity. We can segregate the carriers by their ranks using this analysis. This information combined with delay and cancellation details gives us the overall reliability and quality of the airline service.

hadoop jar ~/Desktop/proj\_jars/finalproject.jar com.kinnar.bigdataproject.avg\_dist\_carrier.AverageMain /project/input/years /project/AvgDistAirtime

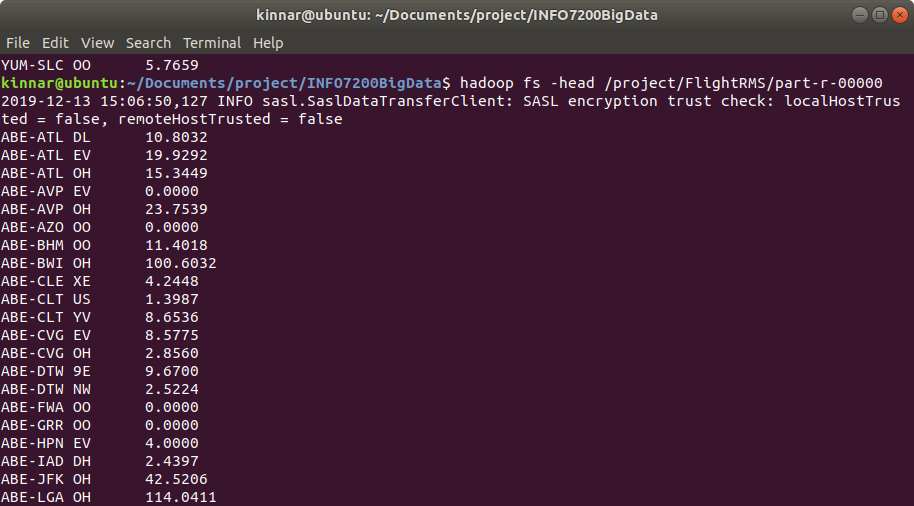


## Recommendation System using RMS (Root Mean Square)

This process is divided in two parts: First we need to calculate average of arrival and departure delay for each source destination pair for each carrier. After that, second part is to calculate RMS value for average of arrival and departure delay.

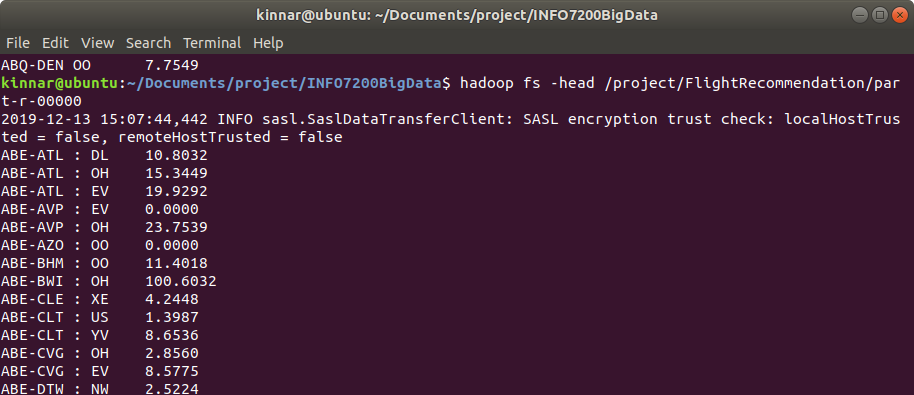
Once this is done, we have to sort the result in ascending order by the RMS value. This will give us the recommendation for choosing carrier for source and destination pair.

hadoop jar ~/Desktop/proj\_jars/finalproject.jar com.kinnar.bigdataproject.rms\_carrier.RMSMain /project/input/years /project/FlightRMS



*Getting RMS values*

hadoop jar ~/Desktop/proj\_jars/finalproject.jar com.kinnar.bigdataproject.recommendation\_sys.SecondarySortDriver /project/FlightRMS /project/FlightRecommendation



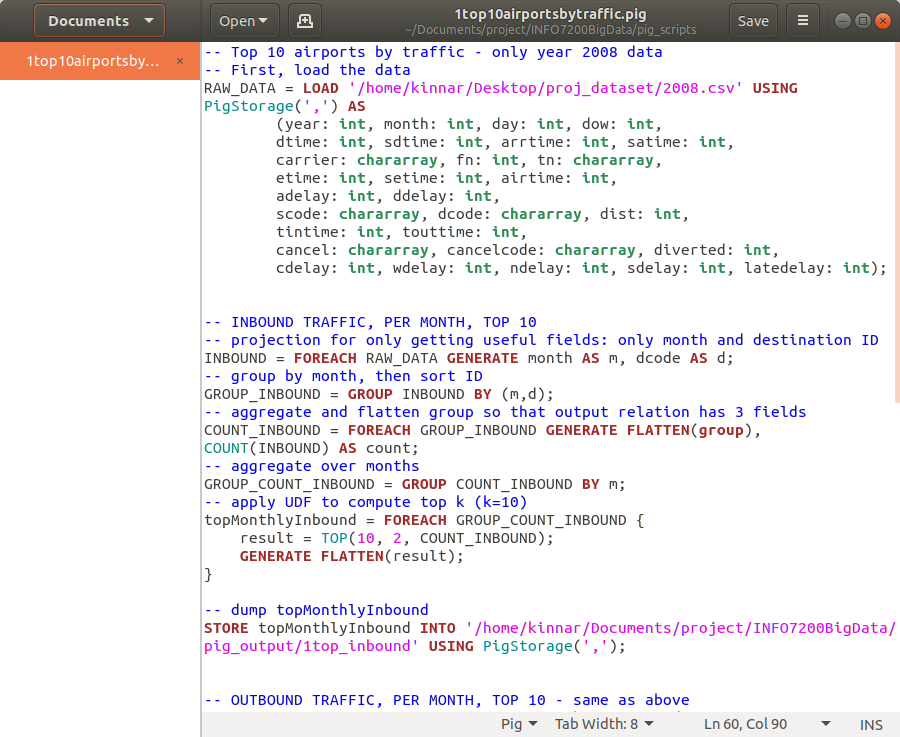
*Get the recommended carrier for source-destination pair*

# Analysis of flights using Pig on Hadoop

## Top 10 cities by total traffic of flights

Here, this analysis would be very difficult to achieve by Java MapReduce. Pig makes it really easy and less time consuming. It also executes really fast and it can leverage the facility to run on the local mode using –x local switch.

This analysis gives us the information about the busies cities by traffic. For each airport, we are finding number of inbound, outbound and all flights.

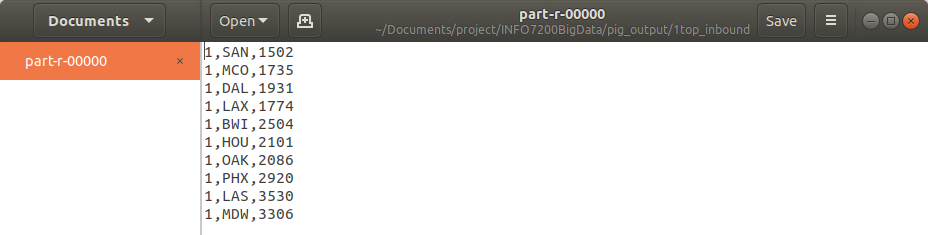


Output:

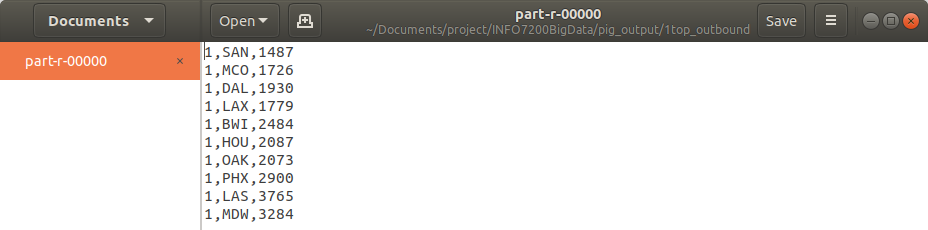
Top 10 inbound



Top 10 outbound

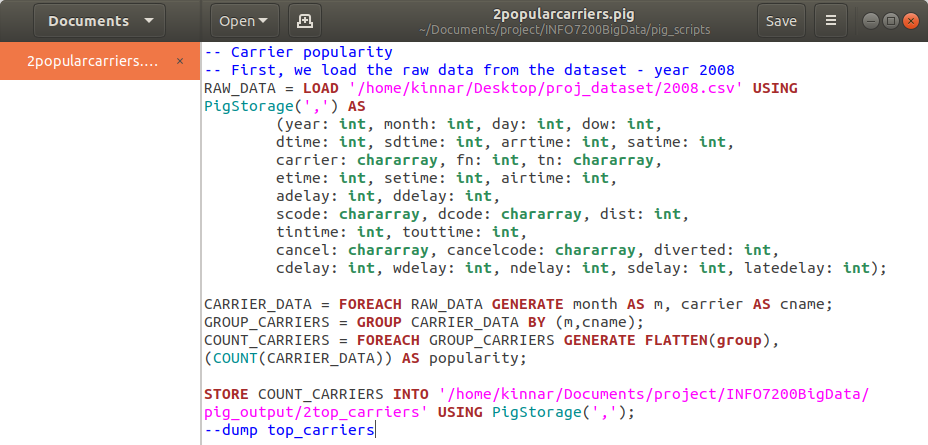


Top 10 monthly

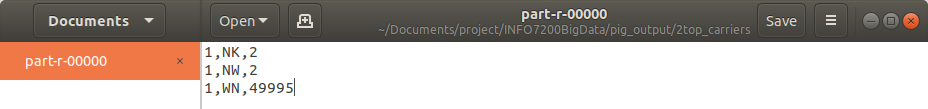


## Popular carriers

Let’s calculate the volume of each carrier by total flights of a year. Carrier ranking is carried out by their median volume value over 10 years’ span.



Output: The data output is very less because I took 2008 year’s data. It would give better result if I use 10 years data.

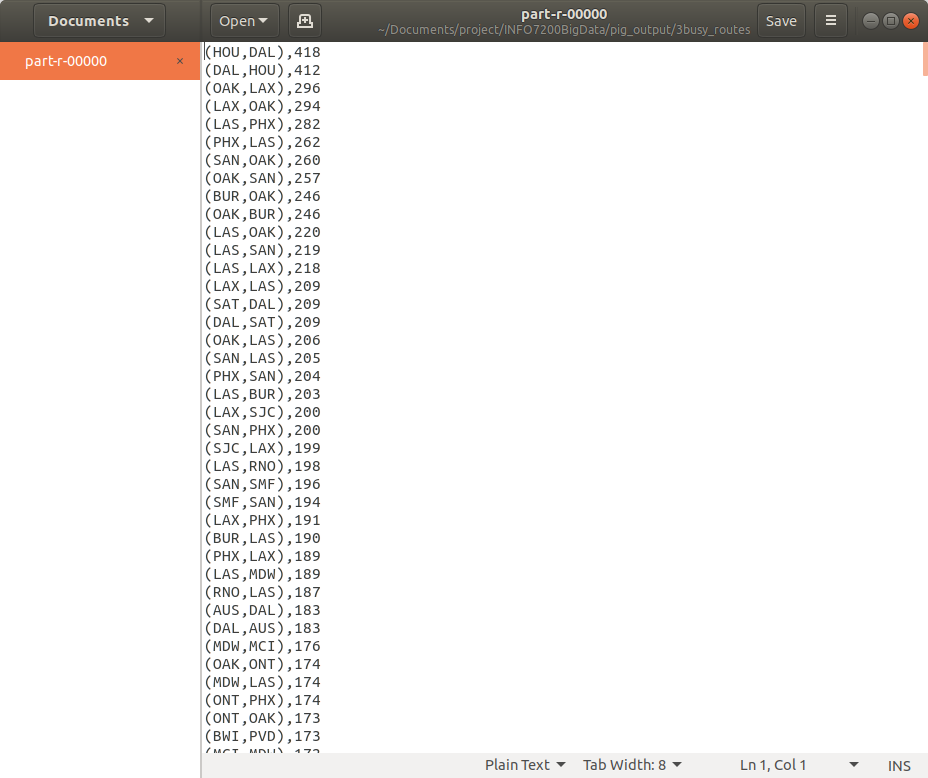


## Most busy routes (descending order)

In this part of analysis, the idea is to create a frequency table for unordered pair (m, n) in which they are the unique pairs of airport codes and it will give us one of the most important information that which airport pairs are busiest.



Output: We are sorting them in descending order so we can check the busiest and popular pair of airports which in turn are source and destinations



# Analysis on Flights using Hive on Hadoop

Hive makes it easy to do analysis since it supports SQL which is one of the most familiar languages for data extraction and manipulation.

Following are the commands used for executing some tasks on Hive. They are according to the below steps:

### Create FlightSchema and setup the parameters

create schema FlightSchema;

use FlightSchema;

SET hive.exec.dynamic.partition = true;

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

### Create table flights and load data from hdfs path

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

LOAD DATA INPATH '/project/input/flights.csv' OVERWRITE INTO TABLE flights;

### Create table airports and load data from hdfs path

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

LOAD DATA INPATH '/project/input/airports.csv' OVERWRITE INTO TABLE airports;

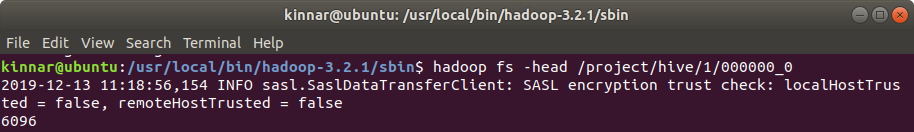
### Create table carriers and load data from hdfs path

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

LOAD DATA INPATH '/project/input/carriers.csv' OVERWRITE INTO TABLE carriers;

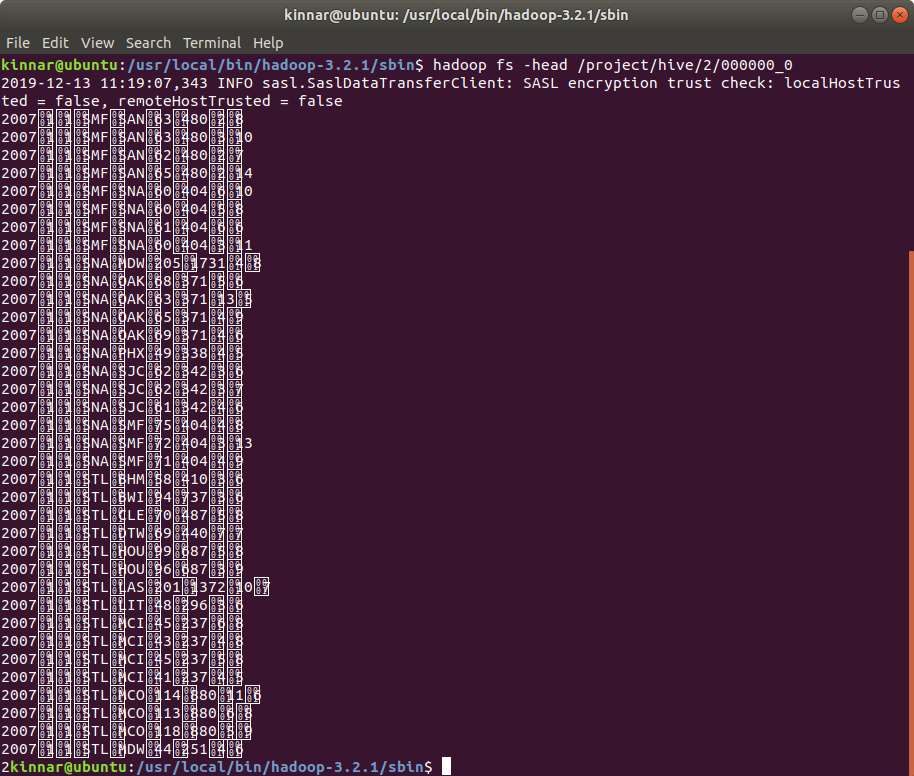
### Find flights which travelled more than 500 airmiles

INSERT OVERWRITE DIRECTORY '/project/hive/1' select count(\*) from flights where AirTime > 500;



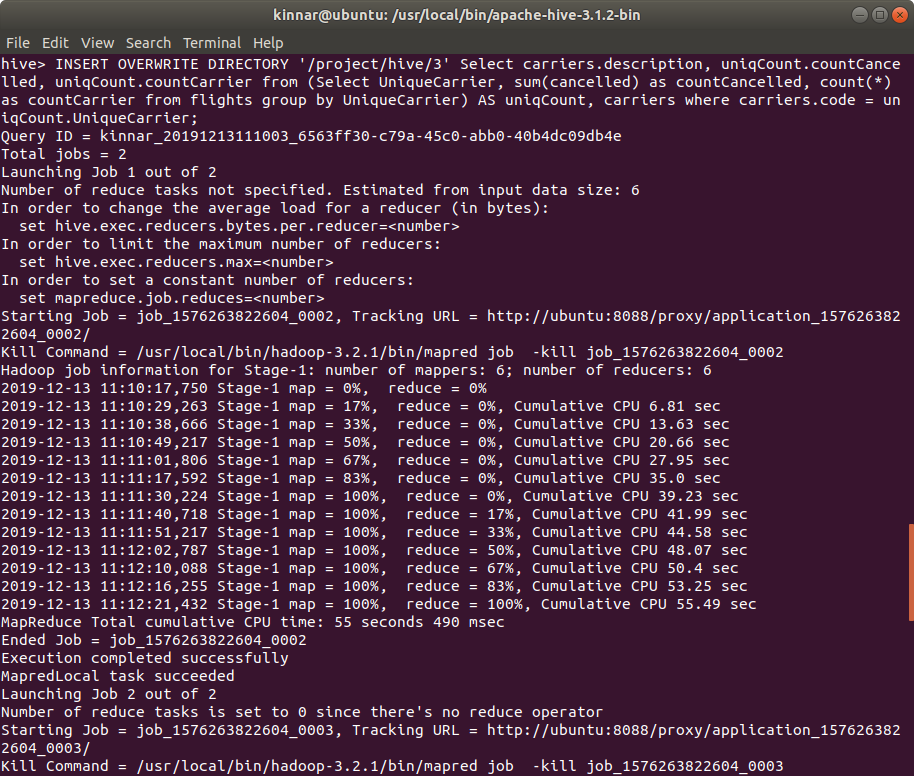
### Find flights which arrive and depart on time

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



### Get count of flights for each carrier

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



### Find origin and destination pairs from DEN airport from 2008 data

INSERT OVERWRITE DIRECTORY '/project/hive/4'

select f.Origin, f.Dest, count(\*) cnt

FROM airports a

JOIN flights f ON (a.Iata = f.Origin)

JOIN airports b ON (b.Iata = f.Dest)

WHERE f.Origin = 'DEN' AND f.Year = 2018

GROUP BY f.Origin, f.Dest;

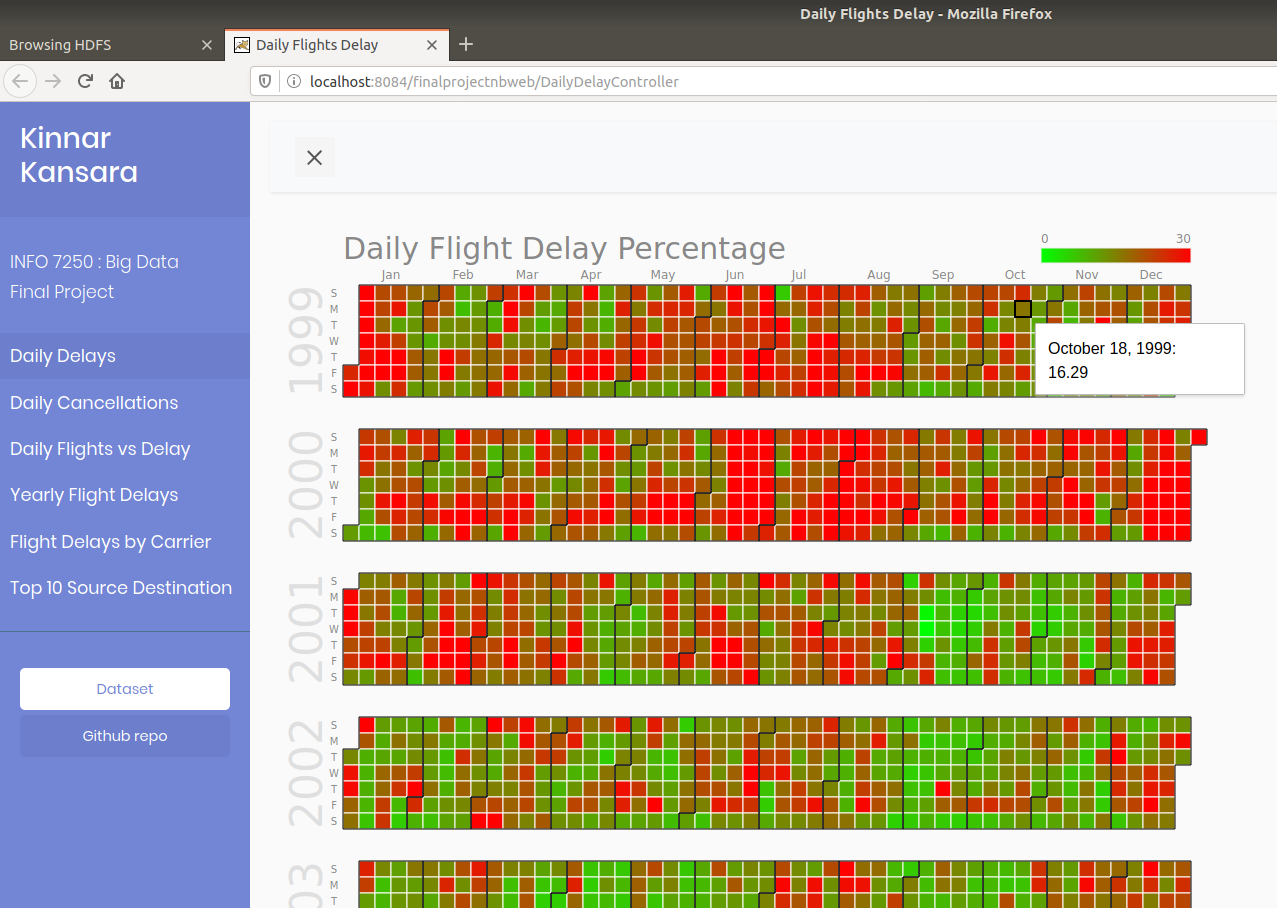


# Graphical Representation and Analysis

We can do many meaningful analyses from the graphical representation rather than the textual data. I will show you how impactful the graphical representation can be. This web application source code can be found from the github repo url mentioned in the overview section.

## % of flight departures delayed > 15 minutes – daily basis

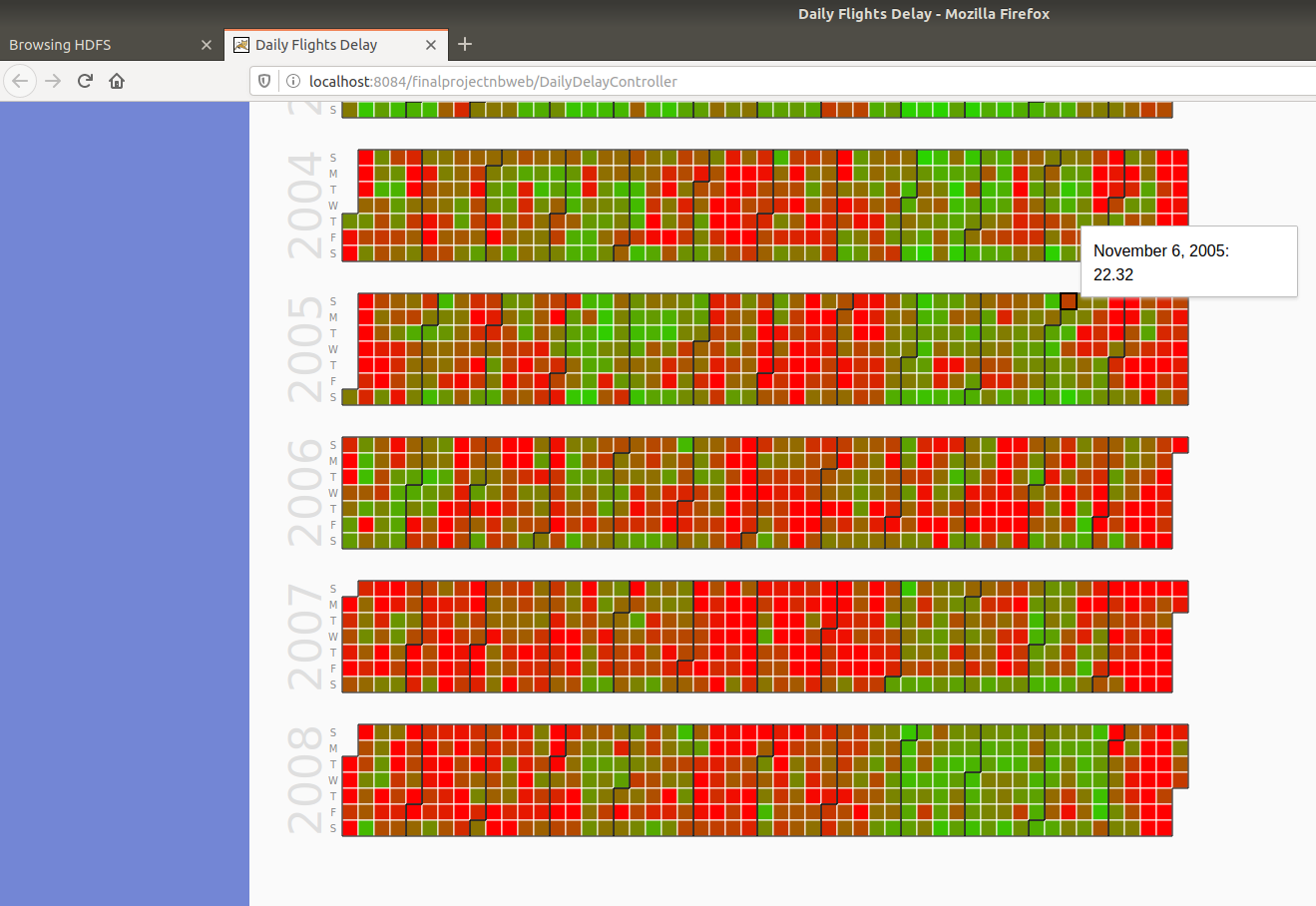
Not good time to fly during holidays



Winter ends in April

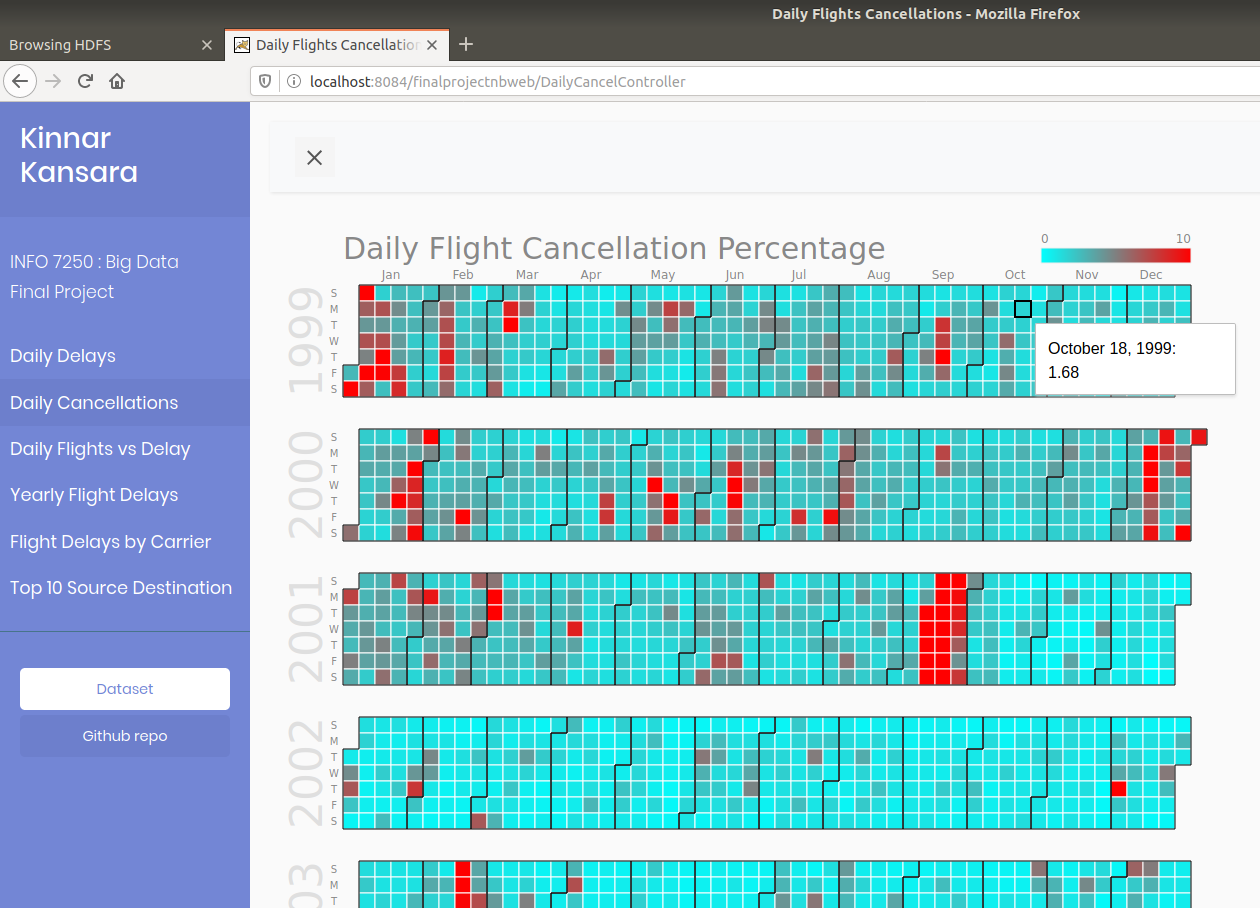
Sep and Oct are best time to fly

Summers are bad time to fly



## % of flight cancelled – daily basis

9/11 changed EVERYTHING



Winter leads to cancellations

Fly in Sep-Oct to avoid cancellations

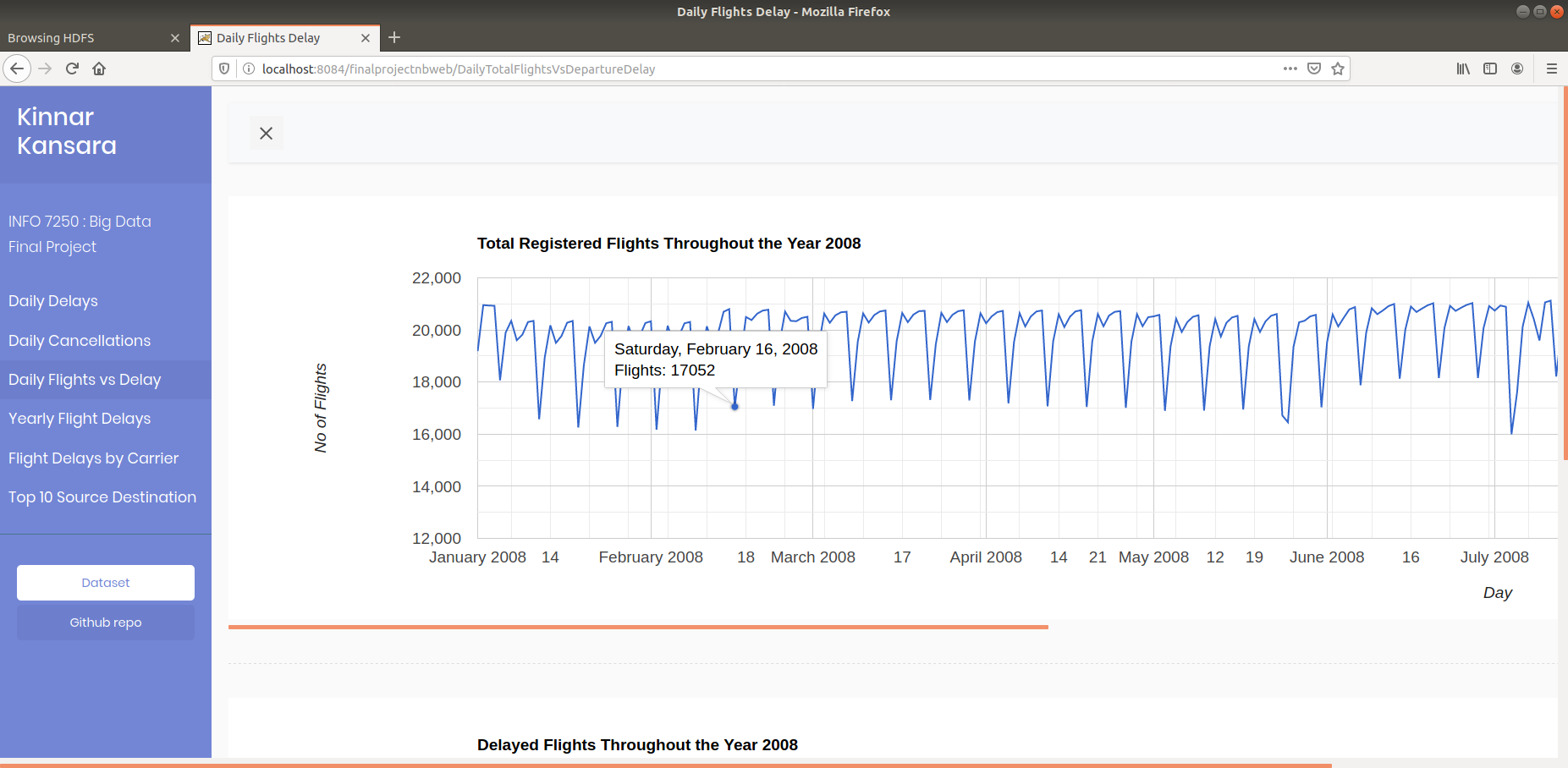


## Flight trend on daily basis

Weekly cycles.

Few flights: Saturday and Sunday

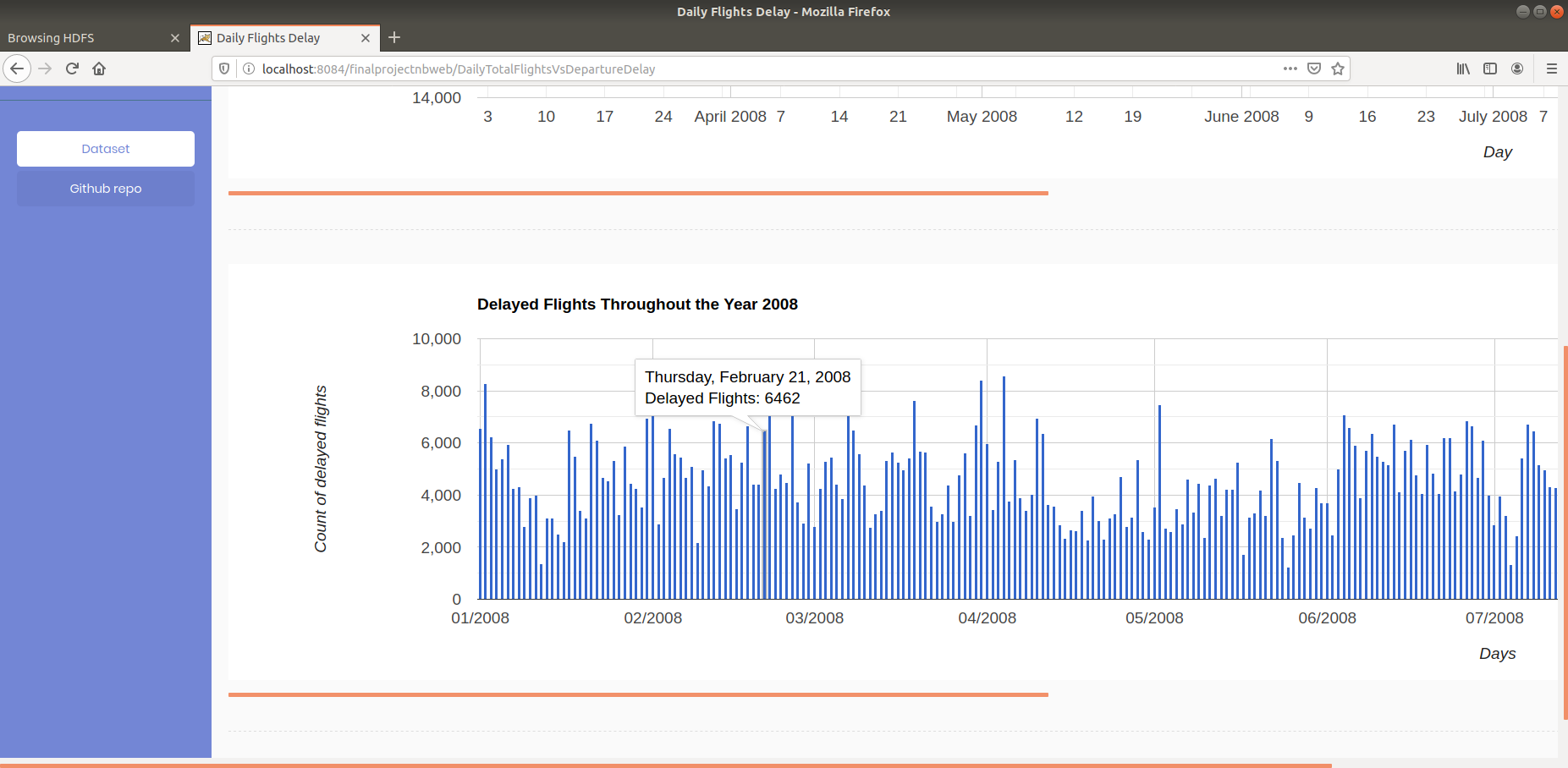
Most flights: Monday, Thursday and Friday



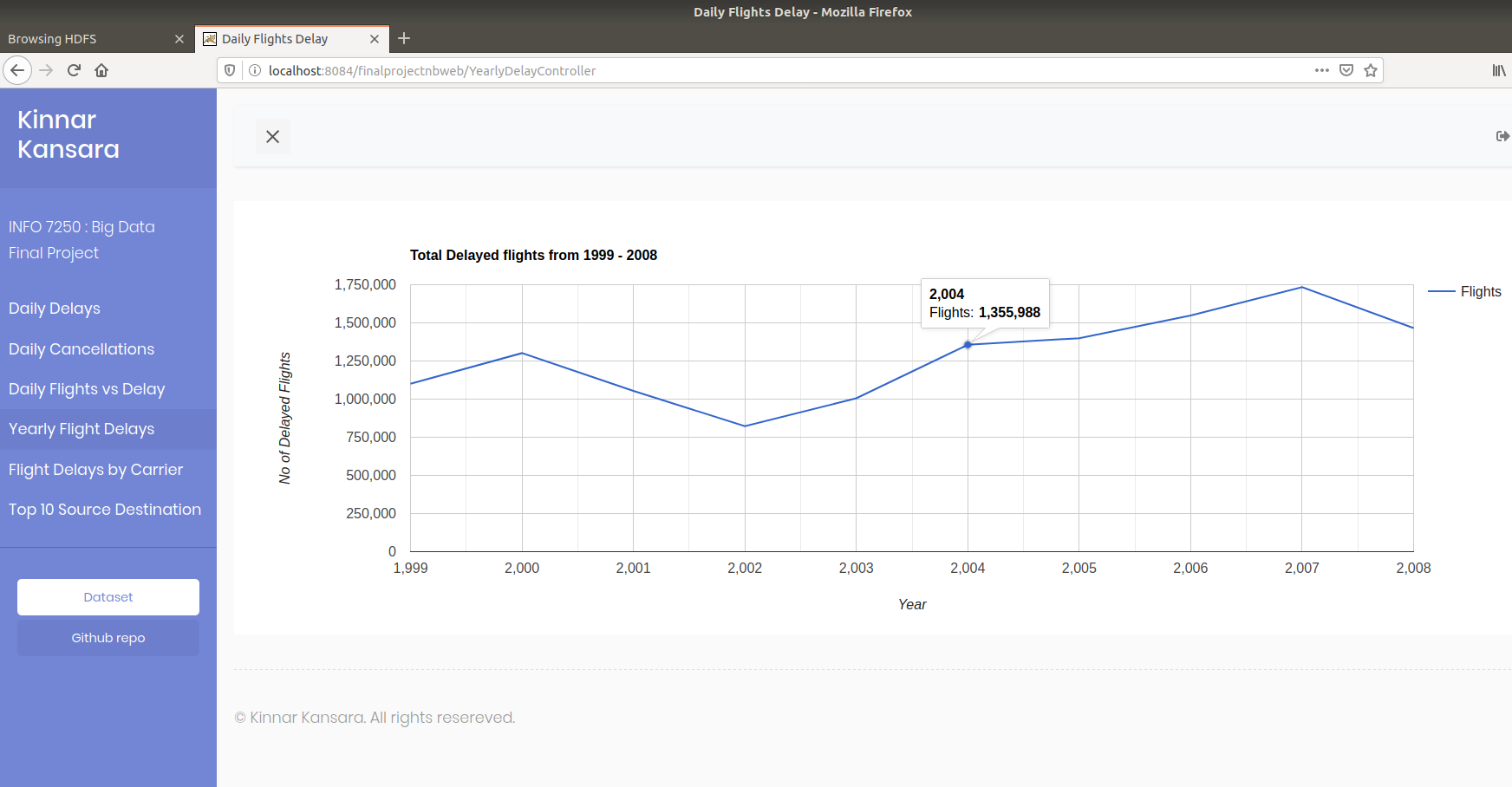
Economic recession lead to fewer flights



## Delayed flights trend on daily basis



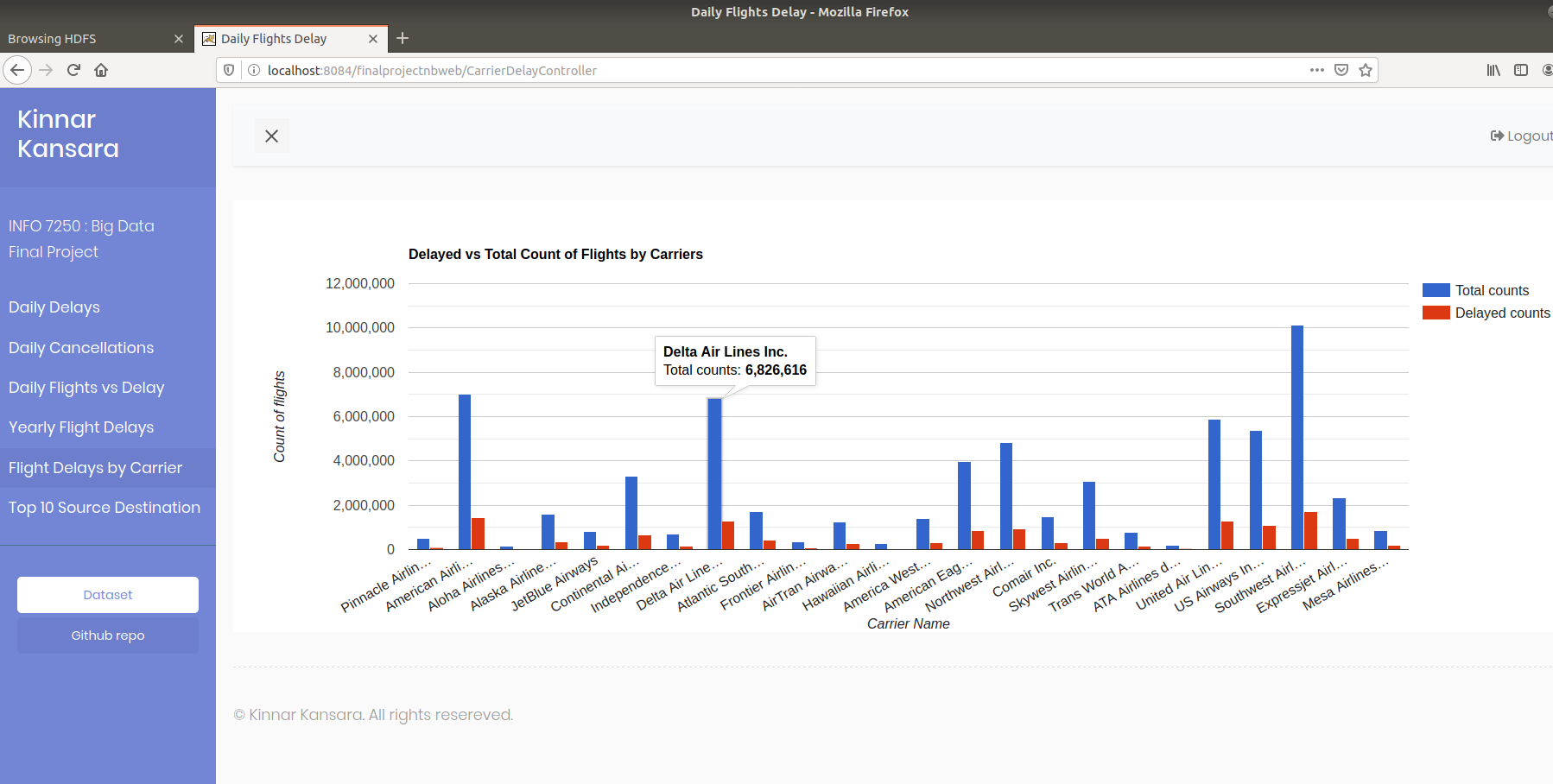
## Flight delay history by year



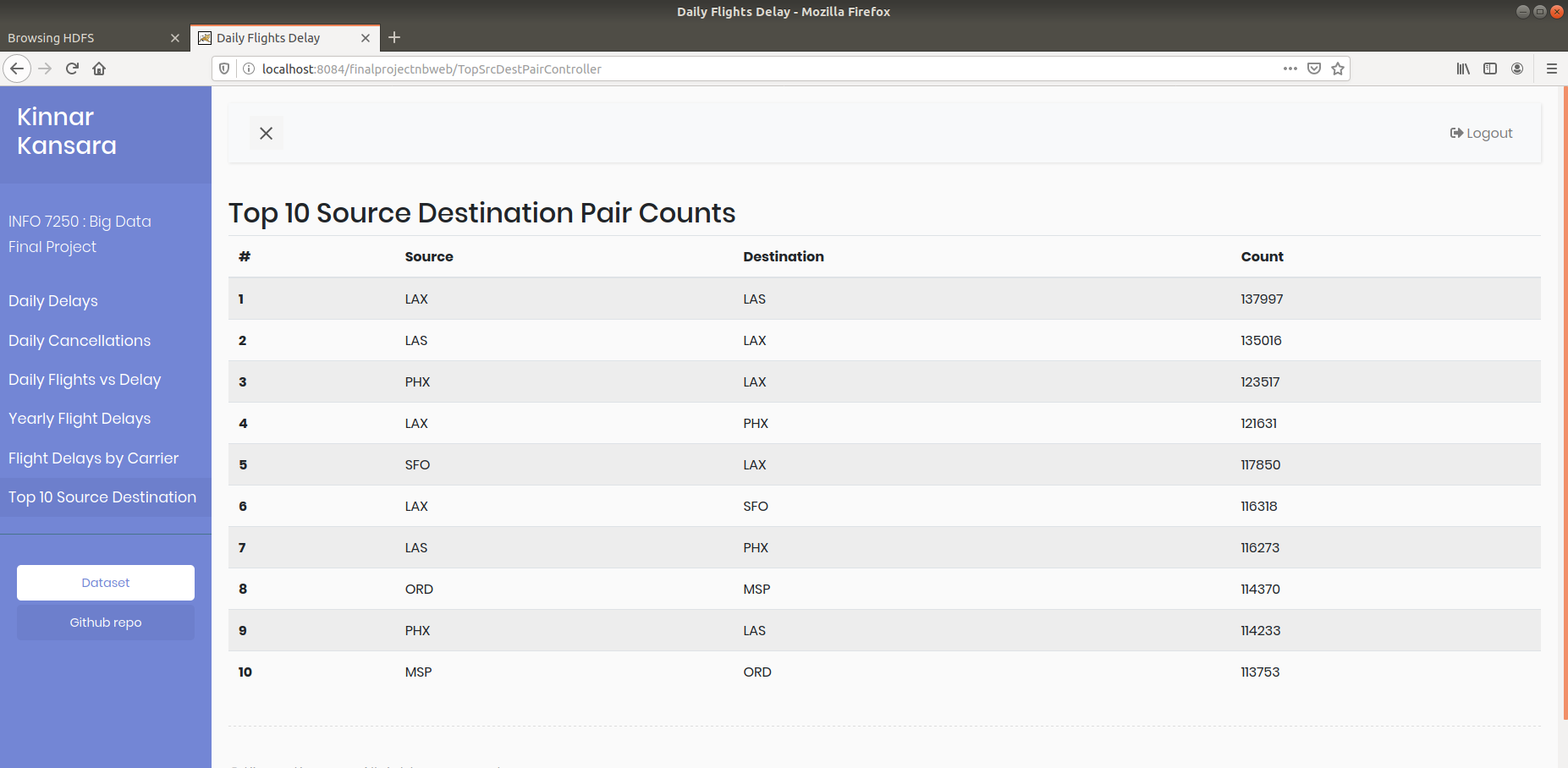
## Total flights vs delayed flights for each carrier

Delta and United Air are among the airlines which cause the most delay so try avoiding them

While airlines like Aloha, Hawaiian and Southwest have superior on time performance



## Tabular view of top 10 most popular source destination pairs



# Lessons Learned & tips for travellers

* Avoid flying during holidays and summer to avoid delays and cancellations due to huge rush
* Fly in April, May, September and October
* Watch the weather!
* Avoid busy airports like JFK, Newark, Chicago which are causing consistent delays
* Use carriers like Southwest, Aloha and Hawaiian with better on-time performance
* Avoid flights which depart at peak hours like 5 to 7 pm.
* Some factors like 9/11 cause tremendous impact on the aviation industry

# Challenges

* I could incorporate the weather details with the analysis but due to inconsistent weather and flight data could not lead to a valuable analysis

# Future Scope

* More analysis with graphical representations can lead to better timing performances
* I could do better analysis with aircrafts like how old are they, what models generally create delays, etc.

# References

https://www.oreilly.com/library/view/data-algorithms/9781491906170/?ar

https://learning.oreilly.com/library/view/mapreduce-design-patterns/9781449341954/

<http://stat-computing.org/dataexpo/2009/posters/wicklin-allison.pdf>

<https://developers.google.com/chart/interactive/docs>

<http://timepasstechies.com/category/programming/data-analytics/hive/>

<http://hadoopilluminated.com/hadoop_illuminated/Public_Bigdata_Sets.html>

# Appendix

## Getting top 10 source destination airport pairs







## Unique Carriers Names with Flights Count











## Year wise flight delay (> 15 minutes) and cancellation. Counts and Ratio









## Flight delay and cancellation by carrier including carrier names















## Date wise flight delay (> 15 minutes) and cancellation. Counts and Ratio











## Average distance covered and airtime done by each carrier











## Recommendation System using RMS (Root Mean Square)

















