FINISHED

```
%sh
wget http://stat-computing.org/dataexpo/2009/2007.csv.bz2 -O /tmp/flights_2007.csv.bz2
wget http://stat-computing.org/dataexpo/2009/2008.csv.bz2 -O /tmp/flights_2008.csv.bz2
wget ftp://ftp.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/2007.csv.gz -O /tmp/weather_2007.csv
wget ftp://ftp.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/2008.csv.gz -O /tmp/weather_2008.csv
echo "downloaded"
```

```
bash: wget: command not found
bash: line 1: wget: command not found
bash: line 2: wget: command not found
bash: line 3: wget: command not found
downloaded
```

Took 0 sec. Last updated by anonymous at February 02 2017, 7:31:28 PM. (outdated)

FINISHED

```
%spark

import org.apache.spark.rdd._
import scala.collection.JavaConverters._
import au.com.bytecode.opencsv.CSVReader

import java.io._
import org.joda.time._
import org.joda.time.format._
import org.joda.time.format.DateTimeFormat
import org.joda.time.DateTime
import org.joda.time.Days
```

```
import org.apache.spark.rdd._
import scala.collection.JavaConverters._
import au.com.bytecode.opencsv.CSVReader
import java.io._
import org.joda.time._
import org.joda.time.format._
import org.joda.time.format.DateTimeFormat
import org.joda.time.DateTime
import org.joda.time.Days
```

Took 5 sec. Last updated by anonymous at February 02 2017, 8:28:43 PM.

FINISHED

```
case class DelayRec(year: String,
                    month: String,
                    dayOfMonth: String,
                    dayOfWeek: String,
                    crsDepTime: String,
                    depDelay: String,
                    origin: String,
                    distance: String,
                    cancelled: String) {

    val holidays = List("01/01/2007", "01/15/2007", "02/19/2007", "05/28/2007", "06/07/2007",
      "09/03/2007", "10/08/2007" ,"11/11/2007", "11/22/2007", "12/25/2007",
      "01/01/2008", "01/21/2008", "02/18/2008", "05/22/2008", "05/26/2008", "07/04/2008",
      "09/01/2008", "10/13/2008" ,"11/11/2008", "11/27/2008", "12/25/2008")

    def gen_features: (String, Array[Double]) = {
      val values = Array(
        depDelay.toDouble,
        month.toDouble,
        dayOfMonth.toDouble,
        dayOfWeek.toDouble,
        get_hour(crsDepTime).toDouble,
        distance.toDouble,
        days_from_nearest_holiday(year.toInt, month.toInt, dayOfMonth.toInt)
      )
      new Tuple2(to_date(year.toInt, month.toInt, dayOfMonth.toInt), values)
    }
    def get_hour(depTime: String) : String = "%04d".format(depTime.toInt).take(2)
    def to_date(year: Int, month: Int, day: Int) = "%04d%02d%02d".format(year, month, day)
```

2/2 thurs | Untitled | Untitled | Untitled | Untitled | Untitled | Untitled | Untitled

```
    def days_from_nearest_holiday(year:Int, month:Int, day:Int): Int = {
      val sampleDate = new org.joda.time.DateTime(year, month, day, 0, 0)

      holidays.foldLeft(3000) { (r, c) =>
        val holiday = org.joda.time.format.DateTimeFormat.forPattern("MM/dd/yyyy").parseDateT
        val distance = Math.abs(org.joda.time.Days.daysBetween(holiday, sampleDate).getDays)
        math.min(r, distance)
      }
    }
  }
```

```
defined class DelayRec
```

Took 2 sec. Last updated by anonymous at February 02 2017, 8:29:36 PM.

FINISHED

```
// function to do a preprocessing step for a given file
def prepFlightDelays(infile: String): RDD[DelayRec] = {
    val data = sc.textFile(infile)

    data.map { line =>
      val reader = new CSVReader(new StringReader(line))
      reader.readAll().asScala.toList.map(rec => DelayRec(rec(0),rec(1),rec(2),rec(3),rec(5),r
    }.map(list => list(0))
    .filter(rec => rec.year != "Year")
    .filter(rec => rec.cancelled == "0")
    .filter(rec => rec.origin == "ORD")
}

val data_2007tmp = prepFlightDelays("/Users/datascienceadmin/Downloads/2007.csv")
val data_2007 = data_2007tmp.map(rec => rec.gen_features._2)
val data_2008 = prepFlightDelays("/Users/datascienceadmin/Downloads/2008.csv").map(rec => rec

data_2007tmp.toDF().registerTempTable("data_2007tmp")

data_2007.take(5).map(x => x mkString ",").foreach(println)
```

```
prepFlightDelays: (infile: String)org.apache.spark.rdd.RDD[DelayRec]
data_2007tmp: org.apache.spark.rdd.RDD[DelayRec] = MapPartitionsRDD[55] at filter at <console
>:77
data_2007: org.apache.spark.rdd.RDD[Array[Double]] = MapPartitionsRDD[56] at map at <console>:
71
data_2008: org.apache.spark.rdd.RDD[Array[Double]] = MapPartitionsRDD[64] at map at <console>:
69
warning: there was one deprecation warning; re-run with -deprecation for details
-8.0,1.0,25.0,4.0,11.0,719.0,10.0
41.0,1.0,28.0,7.0,15.0,925.0,13.0
45.0,1.0,29.0,1.0,20.0,316.0,14.0
-9.0,1.0,17.0,3.0,19.0,719.0,2.0
180.0,1.0,12.0,5.0,17.0,316.0,3.0
```

Took 21 sec. Last updated by anonymous at February 02 2017, 8:31:04 PM.

FINISHED

```
%sql
select dayofWeek, case when depDelay > 15 then 'delayed' else 'ok' end , count(1)
from data_2007tmp
group by dayofWeek, case when depDelay > 15 then 'delayed' else 'ok' end
```

dayofWeek	CASE WHEN (CAST(depDelay AS DOUBLE) > CAST(15 AS DOUBLE)) THEN delayed
1	delayed
7	ok
1	ok
6	delayed
2	delayed
3	ok
4	delayed
3	delayed
5	ok

Took 28 sec. Last updated by anonymous at February 02 2017, 8:37:50 PM.

FINISHED

```
%sql

select cast( cast(crsDepTime as int) / 100 as int) as hour,  case when depDelay > 15 then 'del
from  data_2007tmp
group by  cast( cast(crsDepTime as int) / 100 as int),  case when depDelay > 15 then 'delayed
```

hour	delay
12	ok
13	ok
20	delayed
10	ok
19	ok
15	ok
15	delayed
21	ok
8	ok

Took 25 sec. Last updated by anonymous at February 02 2017, 8:43:38 PM.

Zeppelin

READY

2/2 thurs

default