

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
```

```
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(
    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)
```

```
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)
  }
```

|Flight_data|

```

    }
  }
}

```

```

// 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),1)
  }.map(list => list(0))
  .filter(rec => rec.year != "Year")
  .filter(rec => rec.cancelled == "0")
  .filter(rec => rec.origin == "ORD")
}

val data_2007tmp = prepFlightDelays("/Users/gkrishnan/Downloads/2007.csv.bz2")
val data_2007 = data_2007tmp.map(rec => rec.gen_features._2)
val data_2008 = prepFlightDelays("/Users/gkrishnan/Downlaods/2008.csv.bz2").map(rec => rec.gen

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

```

```

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
defined class DelayRec
prepFlightDelays: (infile: String)org.apache.spark.rdd.RDD[DelayRec]
data_2007tmp: org.apache.spark.rdd.RDD[DelayRec] = MapPartitionsRDD[24] at filter at <console>:
114
data_2007: org.apache.spark.rdd.RDD[Array[Double]] = MapPartitionsRDD[25] at map at <console>:
109
data_2008: org.apache.spark.rdd.RDD[Array[Double]] = MapPartitionsRDD[33] at map at <console>:
107
Warning: there was one deprecation warning; re-run with -deprecation for details
17.0,25.0,4.0,0.0,0.0,719.0,10.0
12.0,1.0,24.0,7.0,15.0,925.0,13.0

```

Flight_data

Took 19 sec. Last updated by anonymous at February 02 2017, 8:51:45 PM.

Flight_data



default

```

%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 57 sec. Last updated by anonymous at February 02 2017, 8:53:38 PM.

%sqlFINISHED ▶ ⌵ 📖 ⚙

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





hour	delay
12	ok
13	ok
20	delayed
10	ok
19	ok
	ok



15	Flight_data	▶ ⌵ 📖 ✂ 📄 ⬇ 📄	🗑	🔍	delayed	📄 ⚙ 🔒 default ▼
21					ok	
8					ok	

Took 52 sec. Last updated by anonymous at February 02 2017, 8:55:05 PM.

READY ▶ ⌵ 📖 ⚙