Lab: Advanced RDD Programming

About This Lab

Objective:	To use the advanced RDD transformations that were covered in a previous lesson

File locations:	HDFS:
	`/tmp/flight.csv`
	`/tmp/carriers.csv`
	`/tmp/plane-data.csv`

Successful outcome:	Find the top 3 airlines with the most flights Find the top 5 most common routes between cities Find the airline with the most delays over 15 mins Find the most common plane for flights over 1500 miles
Before you begin	You should be logged in to your Jupyter notebook

Lab Steps

Perform the following steps:

1. Find the top 3 airlines with the most flights
   a. Create an RDD for `flights.csv`:

```
>>> flightRdd=sc.textFile("/user/root/flights.csv") \
.map(lambda line: line.split(","))
```

   b. This application looks like a word count. As a general rule of thumb, process the minimal amount of data to get the answer. Transform the RDD created above to only get the necessary fields, along with anything else needed for a word count.

```
>>> carrierRdd = flightRdd.map(lambda line: (line[5],1))
>>> carrierRdd.take(1)
```

   c. Reduce the RDD to get the number of flights for each airline
   d. Using `sortByKey`, find the top 3 airlines.

2. Find the top 5 most common routes, between two cities
   a. This application also looks like a word count, but the key is made up of more then one field. Also, there might be more than one airport for each city, make sure to take that into account.
   b. Reuse the `flightRdd` created in **3a**, and create an `airportsRdd` using `airports.csv`:

```
>>> airportsRdd = sc.textFile("/user/root/airports.csv") \
.map(lambda line: line.split(","))
```

c. Create a new RDD using the smallest amount of required data, and join the `airportsRdd` to `flightsRdd`.

   i. Prep the `airportsRdd` and `flightRdd` to only keep what's needed

```
>>> cityRdd = airportsRdd.map(lambda line: (line[0], \ line[2]))
>>> flightOrigDestRdd = flightRdd \
.map(lambda line: (line[12], line[13]))
```

   ii. Join the RDDs to get the correct city, retaining only the required data

d. Map the `citiesRdd` to a new RDD that is then ready to do a `reduceByKey`.

3. **CHALLENGE:** Find the longest departure delay for each airline if its over 15 minutes

   a. This application is similar to a word count, believe it or not.

   b. Filter out all departure delays less then 15 minutes.

   c. Instead of adding together values, compare them to find the longest for each key

      HINT: `max(a,b)` returns the greater of the two values, make sure you're comparing `ints`, the data is read in as a `string` until casted.

4. **CHALLENGE:** Find the most common airplane model for flights over 1500 miles

   NOTE: Not all data is perfect (`plane-data.csv` has some missing values), make sure to filter out airplane model records that don't contain 9 fields after it is split into an array.