Assignment: Advanced Join in Spark

Make sure first you were able to complete the "Setup PySpark on the Cloudera VM" tutorial in lesson 1 of this module.

Verify input data

In this lesson you will use the data you generated in the second part of the programming assignment of module 4 lesson 2. Make sure the 6 files are available in the HDFS input/ folder by running this command in the terminal (not in PySpark!):

```
hdfs dfs -ls input/
```

Should contain the 6 files:

```
input/join2_genchanA.txt
input/join2_genchanB.txt
input/join2_genchanC.txt
input/join2_gennumA.txt
input/join2_gennumB.txt
input/join2_gennumC.txt
```

Goal of the programming assignment

gennum files contain show names and their viewers, genchan files contain show names and their channel. We want to find out the total number of viewer across all shows for the channel BAT.

Read shows files

gennum files contains show names and number of viewers, you can read into Spark all of them with a pattern matching, see the ? which will match either A, B or C:

```
show_views_file = sc.textFile("input/join2_gennum?.txt")
```

Remember you can check what Spark is doing by copying some elements of an RDD back to the driver:

```
show_views_file.take(2)
```

will return the first 2 elements of the dataset:

```
[u'Hourly_Sports,21', u'PostModern_Show,38']
```

Parse shows files

Next you need to write a function that splits and parses each line of the dataset.

```
def split_show_views(line):
    <INSERT_CODE_HERE>
    return (show, views)
```

Then you can use this function to transform the input RDD:

```
show_views = show_views_file.<INSERT_CODE_HERE>(split_show_views)
```

By now you should know how to check that the show_views RDD is how you expect.

Read channel files

genchan files contains show names and channel, you can read into Spark all of them with a pattern matching, see the ? which will match either A, B or C:

```
show_channel_file = sc.textFile("input/join2_genchan?.txt")
```

Parse channel files

Write a function to parse each line of the dataset:

```
def split_show_channel(line):
    <INSERT_CODE_HERE>
    return (show, channel)
```

Use it to parse the channel files:

```
show_channel = show_channel_file.<INSERT_CODE_HERE>
```

Join the 2 datasets

At this point you should use the join transformation to join the 2 dataset using the show name as the key.

You can join the datasets in any order, as long as you are consistent, both are fine.

```
joined_dataset = <INSERT_CODE_HERE>
```

Extract channel as key

You want to find the total viewers by channel, so you need to create an RDD with the channel as key and all the viewer counts, whichever is the show.

```
def extract_channel_views(show_views_channel):
    <INSERT_CODE_HERE>
    return (channel, views)
```

Now you can apply this function to the joined dataset to create an RDD of channel and views:

```
channel_views = joined_dataset.<INSERT_CODE_HERE>(extract_channel_views)
```

Sum across all channels

Finally, we need to sum all of the viewers for each channel:

```
def some_function(a, b):
    <INSERT_CODE_HERE>
    return some_result
```

This is the final stage of your analysis, so you can copy the results back to the Driver with collect:

```
channel_views.<INSERT_CODE_HERE>(some_function).collect()
```

Submit one line for grading

Finally, you need to create a text file with just one line for submission.

From the Cloudera VM, open the text editor from Applications > Accessories > gedit Text Editor.

Paste 1 single number into gedit, the number of viewers for the BAT channel, with no punctuation, spaces, commas. Just the digits of the number.

In gedit, click on the Save button and save it in the default folder (/home/cloudera) with the name bat_viewers.txt

Open now the browser within the Cloudera VM, login to coursera, and upload that file for grading.