Advanced Join in Spark

Verify input data

Ensure the following 6 files are available in the HDFS input/ folder by running the following command in the terminal (not in PySpark!):

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

Containing the following 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

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

Read shows files

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

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

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

Write a function that splits and parses each line of the dataset.

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

Use the following function to transform the input RDD:

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

Check that the show_views RDD is as expected.

Read channel files

genchan files contains show names and channel, read them into Spark with a pattern matching, see the '?' character 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

Use the join transformation to join the 2 datasets using the show name as the key.

Join the datasets in any order.

```
joined_dataset = <INSERT_CODE_HERE>
```

Extract channel as key

Find the total viewers by channel, 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)
```

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

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 the analysis, so copy the results back to the Driver with collect:

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