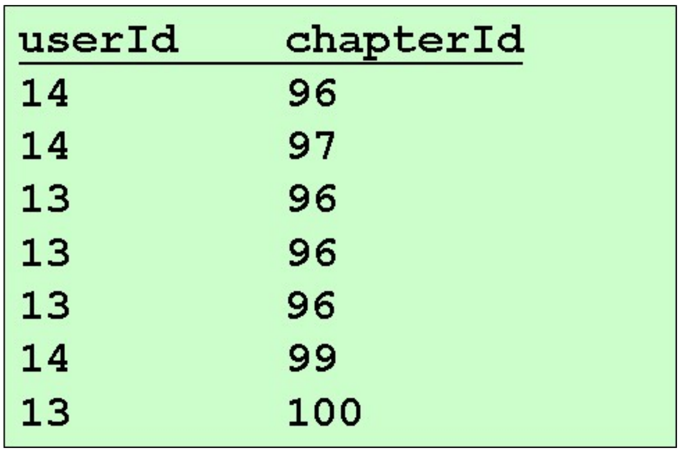
Ranking Video Training Courses by Number of Views

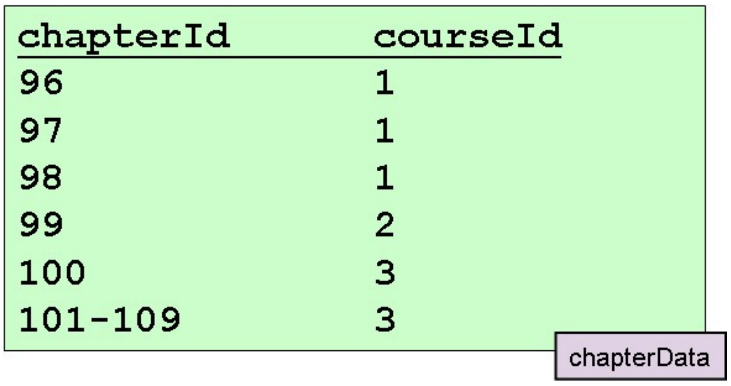
You are producing an Apache Spark "report" for a video training company. They have dumped a set of raw data from a database which is capturing the number of views each video has.

Here is the raw data. First of all, there is a dump containing "views" - there are two columns as follows:



userId	chapterId
14	96
14	97
13	96
13	96
13	96
14	99
13	100

What are chapterIds? We have a separate dump file containing the following:



chapterId	courseId
96	1
97	1
98	1
99	2
100	3
101-109	3

chapterData

There is another dump file (titles.csv) containing courseIds against their titles - we will use this at the end to print the results...

Exercise 1: as a warm up, build an RDD containing a key of courseId together with the number of chapters on that course.

hint: You can do this using a map to pair as follows:

courseId	count
1	1
1	1
1	1
2	1
3	1
3	1

(etc, lots more 3's!)

Throughout this guide we'll give you a guide to the "expected" data. For this warmup you should get (although not necessarily in this order!)

courseId	chapters
1	3
3	10
2	1

Exercise 2: Your job is to produce a ranking chart detailing which are the most popular courses by score.

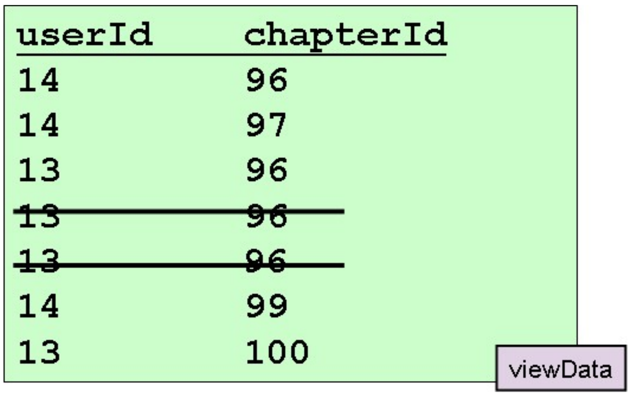
Business Rules:

We think that if a user sticks it through most of the course, that's more deserving of "points" than if someone bails out just a quarter way through the course. So we've cooked up the following scoring system:

- *If a user watches more than 90% of the course, the course gets 10 points*
- *If a user watches > 50% but <90% , it scores 4*
- *If a user watches > 25% but < 50% it scores 2*
- *Less than 25% is no score*

Step 1: Removing Duplicate Views

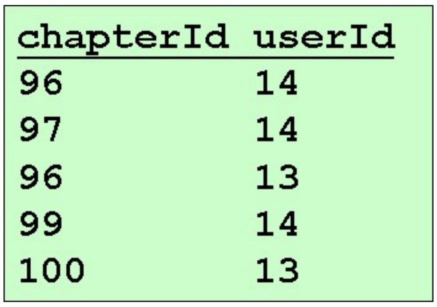
Let's analyse our test data...the same user watching the same chapter doesn't count, so we can call distinct to remove duplications...



userId	chapterId
14	96
14	97
13	96
~~13~~	~~96~~
~~13~~	~~96~~
14	99
13	100

Step 2: Joining to get Course Id in the RDD

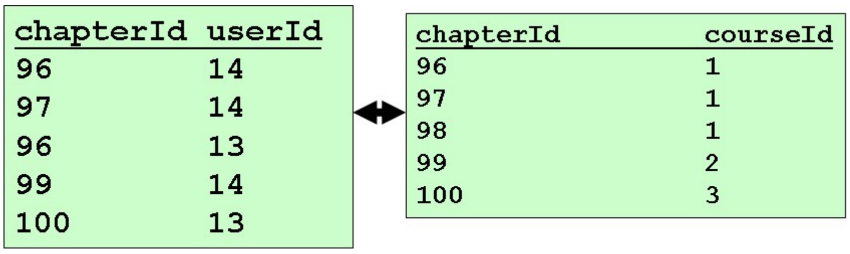
This isn't very meaningful as it is - we know for example that user 14 has watched 96 and 97, but are they from the same course, or are they different courses???? We need to join the data together. As the common column in both RDDs is the chapterId, we need both Rdds to be keyed on that.



chapterId	userId
96	14
97	14
96	13
99	14
100	13

You need to do a map to switch the key and the value.

Now that you have two RDDs, each keyed on chapterId, you can join them together. Here's the input...



chapterId	userId
96	14
97	14
96	13
99	14
100	13

chapterId	courseId
96	1
97	1
98	1
99	2
100	3

And now after joining you will have:

chapterId	(userId,courseId)
96	(14, 1)
97	(14, 1)
96	(13, 1)
99	(14, 2)
100	(13, 3)

Step 3: Drop the Chapter Id

As each "row" in the RDD now represents "a chapter from this course has been viewed", the chapterIds are no longer relevant. You can get rid of them at this point - this will avoid dealing with tricky nested tuples later on. At the same time, given we're counting how many chapters of a course each user watched, we will be counting shortly, so we can drop a "1" in for each user/course combination.

Expected RDD after this step:

(userId,courseId)	count
(14, 1)	1
(14, 1)	1
(13, 1)	1
(14, 2)	1
(13, 3)	1

Step 4 - Count Views for User/Course

We can now run a reduce as usual...

(userId,courseId)	views
(14, 1)	2
(13, 1)	1
(14, 2)	1
(13, 3)	1

We can read these results as "user 14 watched 2 chapters of 1; user 13 watched 1 chapter of course 1; user 14 watched 2 chapters of course 2. etc".

Step 5 - Drop the userId

The user id is no longer relevant. We've needed it up until now to ensure that for example, 10 users watching 1 chapter doesn't count the same as 1 user watching 10 chapters! Another map transformation should get you...

courseId	views
1	2
1	1
2	1
3	1

To rephrase the previous step, we now read this as "someone watched 2 chapters of course 1. Somebody *different* watched 1 chapter of course 1; someone watched 1 chapter of course 2, etc".

Step 6: of how many chapters?

The scoring is based on what percentage of the total course they watched. So we will need to get the total number of chapters for each course into our RDD:

courseId	(views, of)
1	(2,3)
1	(1,3)
2	(1,1)
3	(1,10)

"Someone watched 2 chapters of 3. Somebody *different* watched 1 chapter of 3".

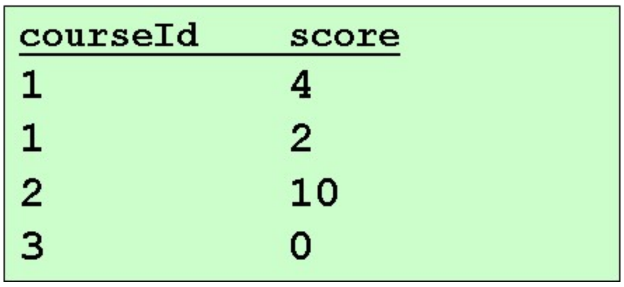
Step 7: Convert to percentages

This should be an easy mapping. Be careful of Java datatypes here!

courseId	percent
1	0.66667
1	0.33333
2	1.0
3	0.1

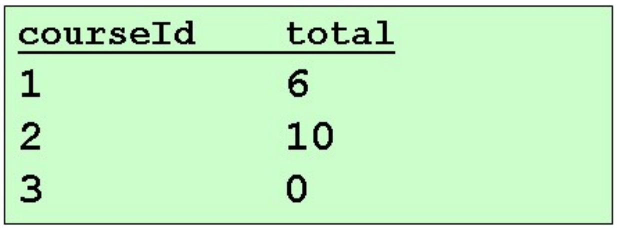
Step 8: Convert to scores

As described earlier, percentages are converted to scores:



courseId	score
1	4
1	2
2	10
3	0

Step 9: Add up the total scores



courseId	total
1	6
2	10
3	0

Exercise 3: this is optional but as a final touch, we can get the titles using a final (small data) join. You can also sort by total score (descending) to get the results in a pleasant format.

Exercise 4: once you've got the output looking "nice", it's more fun to run the job against some real data - change the value for testMode to false and it will read the data from disk (using some anonymized real viewing figures from VirtualPairProgrammers.com, over a short period from last year).