Folder Structure:

Hadoop → README, Installation_and_configuration_instructions, hadoophw

SPARK→ README, Installation_and_configuration_instructions, sparkhw

I suggest you to go through README files under each of the above folders to understand sub folder structures and the commands and instructions needed to compile, execute the corresponding programs.

WordCount:

Input: Two files. File1.txt, File2.txt.

Goal: To Count the number of files each word appears in.

Design Description:

We initially get the file name through following command:

FileSplit fileSplit = (FileSplit) context.getInputSplit();

String filename = fileSplit.getPath().getName(); → yields filename

Mapper:

As long as there are words in input file we do the following:

1) Filter out all non-alphabetic strings and convert every token to lower case.
2) Store every token as key and value as its filename**.(word, filename)**

Reducer:

Here we create a file Array to store the unique values (filenames) of each key.

So if a word appears twice in a single file, file Array stores the file name only once.

Thus at the end of the reducer we have output like **(word, fileArray size)**

Inputs:

File1.txt → Orlando, Miami, Orlando

File2.txt → Orlando, Miami, Melbourne

Output:

Orlando 2

Miami 2

Melbourne 1

Review:

Input: review.data file

Goal: To find all the reviews with more than 100 words and group the result by “overall”, and save the result to file.

Design Description:

1) We read the input file review.data.
2) We then filter out the ‘reviewTexts’ which have length greater than 100.
3) For the filtered output we map the corresponding ‘Overall’ to ‘asin’ and ‘reviewText’
4) We then sort the reviews in descending order, so that higher ratings are at the top.
5) We then save the output.

Output:

Please check the output file at this path /spark/sparkhw/ouput1/part-00001

Sample Output:

(5.0, (u'0802132758', u"Ever wonder whether you were the hero of your life or just a minor character in someone else's?Stoppard explores this premise brilliantly in this revisiting of Shakespeare's Hamlet, this time seen through the eyes of two bit-players in that great drama: Rosencrantz & Guildenstern, hapless drifters caught in a whirl of fate and trying to figure out their place in the universe.With dialogue that cracks like a whip and a truly dizzying plot, we follow these two curious men attempt to make sense of the strange dramatic world into which they've been thrust. As this play weaves into & back out of Shakespeare's original, we are treated to a dazzling display of slapstick existentialism.Stoppard has created a contemporary classic that is every bit as entertaining as it is disturbing"))

Input: meta.data file

Goal: Calculate the average number of words in each review for all reviews on products belonging to “Music” category.

Design Description:

1) We read the meta.data input file.
2) Filter the category which is equal to “Music”.
3) Map “asin” and “music” and then later map “asin” to lenofwords(“reviewText”) and then join these.
4) Save the output in a file.
5) Sum all the words and divide by total number of joined records yields us the average.

Output:

Please check the files at D:\spring 2018\dic\hw3\hw3\spark\sparkhw\output2\part-0000*

SampleOutput: The output is the sum of all words lengths/ total_num_of_records in those files.(834438/9307=89.657)