Name: Vaishnavi Manthena NetID: vm504
Assignment: CS417 Project 3 Report

IMPLEMENTATION

RedditPhotoImpact.java

RDD	Format of each RDD row	Transformation function for next RDD
lines	Each line of in the input csv	Map to pair
instanceImpact	(image_id, num_of_upvotes+num_of_downvotes + num_of_comments)	reduceByKey [to add the values of each distinct key]
photoImpact	(image_id, total impact of this image)	1) mapToPair [to swap key and value] 2) sortByKey [to sort in descending order] 3) mapToPair [swap key, value again to get original format]
Photo_impact_sorted [This is the final RDD from which I collect and print out results]	Same as above but keys are sorted in descending order of 'total impact'	

RedditHourImpact.java

RDD	Format of each RDD row	Transformation function for next RDD
lines	Each line of in the input csv	mapToPair
est_hours	(hour of America/New_york time zone, num_of_upvotes+num_of_downvotes + num_of_comments) So, basically Key: hour Value: impact of this repost	reduceBykey to sum up all values for a key so that we get total impact for a given hour.
hour_impact	(hour, total Impact)	sortByKey
hour_impact_sorted [final RDD from which to collect result and print]	Same as above but rows are sorted in ascending order with respect to the keys (hours)	

NetflixMovieAverage.java

RDD	Format of each RDD row	Transformation that created this rdd
lines	Each line of in the input csv	
rating	Key: movie id Value: single rating for this movie	lines.mapToPair
totalRating	Key: movie id Value: Sum of all ratings for this movie	Rating.reduceByKey (to sum up values corresponding to the same key)
Counter	Key: movie id Value: 1	lines.mapToPair
counts	Key: movie id Value: total number of ratings for this movie	counter.reduceByKey (to sum up values corresponding to the same key)
avgRating	Key: movie_id Value: avg rating for this movie	(totalRating.join(counts)).mapValues(x -> ((x._1() * 1.0) / x._2())) So, basically using the values from totalRating and Counts for each key (movie_id) to get its average rating.
avgRating_sorted [final RDD from which to collect the result and print]	Same as above but the rows are sorted in descending order of value (average rating)	1) mapToPair [to swap key and value] 2) sortByKey [to sort in descending order] 3) mapToPair [swap key, value again to get original format]

NetflixGraphGenerate.java

RDD	Format of each RDD row	Transformation that created this rdd
lines	Each line of in the input csv	
rating	Key: (movie, rating) Value: customer_id	lines.mapToPair
commonCustomers	Key: (movie, rating) Value: Iterable of all customer_ids	Rating.groupByKey
commonCustomersLines	Each row of RDD is an iterable of customer ID's. So, basically each row of RDD stores a list of customers who should have an edge between each other.	commonCustomers.map()
connections	Each RDD is a pair: (customer A, customer B)	commonCustomersLines.flatMap I used a helper function 'toPairs' to implement this. Basic Idea: From each line of commonCustomersLines get all the customer pairs that can be generated and map each pair to its own line in a new RDD.
sorted_connections	Similar to above RDD except this time I make sure that customer A < customer B. If not, then change the line from (customer A, customer B) to (customer B, customer A)	Connections.map
conn_count	Key: (customer A, customer B) where customer A < customer B Value: 1	sorted_connections.mapToPair
conn_total	Key: (customer A, customer B) where customer A < customer B Value: Total number of edges between customers A and B	conn_count.reduceByKey
conn_total_sorted [final RDD from which to collect the result and print]	Same as above but rows are sorted in descending order of value (# of edges)	Similar to previous cases where I sorted by value

In the description when I say the number of edges between two customers, I mean the weight of the edge between them.

PROJECT RESULTS

Reddit Data Set	Most Impactful PhotoID	Sum of all comments and votes (PhotoImpact)
Small	0	73988
Medium	222	165877
Large	1437	192896

Reddit Data Set	Most Impactful hour (EST)	Sum of all comments and votes (Hour Impact)
Small	1	73634
Medium	21	1270862
Large	20	15057971

Movies which have highest average rating in the large data set:

Movie ID	Movie Name	Year	Avg Rating rounded to 2 decimal places
14961	Lord of the Rings: The Return of the King: Extended Edition	2003	4.72
7230	The Lord of the Rings: The Fellowship of the Ring: Extended Edition	2001	4.72
7057	Lord of the Rings: The Two Towers: Extended Edition	2002	4.70

REFLECTION

I believe the hardest part of the project was getting used to the new syntax and dealing with errors like the datatype mismatch errors (list vs Iterable, etc ..). The sample WordCount example helped a lot in getting started. Also, online resources like stackoverflow helped a lot in fixing the errors. Also, my own trial and error to figure out potential operations/transformations was useful.

I would say the second hardest part was testing. I figured that using python notebooks and dataframes was the most straightforward way to check my solutions. It was hard to get a

scheme to test NetflixGraphGenerate so I just checked my output for my own simple inputs I generated.