Q1

val rdd1 = sc.textFile("/FileStore/tables/soc\_LiveJournal1Adj-2d179.txt")

val users = rdd1.map(x=>x.split("\t")).filter(x=>x.length == 2).map(x=> (x(0),x(1).split(",").toList))

val mutual = users.flatMap(x=>x.\_2.map(y=>((math.min(x.\_1.toInt,y.toInt),math.max(x.\_1.toInt,y.toInt)),x.\_2))).reduceByKey(\_.intersect(\_))

val mutual1 = mutual.map(x=> (x.\_1.toString().replace("(","").replace(")","").replace(",","\t"),x.\_2.mkString(",")))

val mutual2 = mutual1.filter(x=>((x.\_1 == "0\t4") ||(x.\_1 == "20\t22939")||(x.\_1 == "1\t29826")||(x.\_1 == "6222\t19272")||(x.\_1 == "28041\t28056") )).map(x=>(x.\_1+"\t"+x.\_2)).collect().mkString("\n")

dbutils.fs.put("/FileStore/tables/checkmic.txt", mutual2)

Q2)

import org.apache.spark.sql.functions.{concat, lit, broadcast}

val count\_mutual = mutual.map(x=>(x.\_1.toString(),x.\_2.length))

val test = count\_mutual.take(1)

val reverse\_sorted = count\_mutual.sortBy(-\_.\_2)

reverse\_sorted.take(10)

val top\_ten\_friends = sc.parallelize(reverse\_sorted.take(10))

val arranged\_pairs1 = top\_ten\_friends.map(x=>(x.\_1.replace("(",""),x.\_2))

val arranged\_pairs2 = arranged\_pairs1.map(x=>(x.\_1.replace(")",""),x.\_2))

val finalvals = arranged\_pairs2.map(x=>(x.\_1.replace(",","").toString(),x.\_2))

val rdduserdata = sc.textFile("/FileStore/tables/userdata.txt")

val userdata = rdduserdata.map(x=>x.split(",")).filter(\_.length==10).map(x=>(x(0),x(1),x(2),x(3)))

//userdata.take(10)

val left = finalvals.toDF("UserId","Mutual Num")

val right = userdata.toDF("UserId1","First Name1","Last Name1","Address1")

val right2 = userdata.toDF("UserId2","First Name2","Last Name2","Address2")

val combinedata = right.crossJoin(right2)

val restructuredata = combinedata.withColumn("UserId", concat(combinedata("UserId1"),lit(""), combinedata("UserId2"))).drop("UserId2").drop("UserId1")

val final\_result = restructuredata.join(broadcast(left), "UserId")

val finalrdd = final\_result.select("Mutual Num","First Name1","Last Name1","Address1","First Name2","Last Name2", "Address2").rdd

finalrdd.take(1)

val finalvalues = finalrdd.map(x=>x.toString().replace("[","").replace("]","").replace(",","\t")).collect().mkString("\n")

dbutils.fs.put("/FileStore/tables/checkmic2.txt", finalvalues)

Q3 -1a

val v1 = sc.textFile("/FileStore/tables/ratings.csv")

val v2 = v1.map(x => (x.split(","))).filter(x=>(x(1).toString() != "movieId" && x(2) != "rating" )).collect()

val v3 = v2.map(x=> (x(1),x(2).toDouble))

val v4 = sc.parallelize(v3)

val v5 = v4.toDF("MovieId","Rating")

val v6 = v5.groupBy("MovieId").avg().drop("avg(MovieId)")

display(v6)

Q3-1b

val v7 = v6.sort("avg(Rating)").toDF("MovieId","Average Rating")

val v17 = spark.read.format("csv").option("header", "true").load("/FileStore/tables/movies1.csv")

val v8 = v7.join(v17, v7("MovieId") === v17("movieId"))

display(v8.take(10))

Q3-2

val v10 = sc.textFile("/FileStore/tables/tags.csv")

val v11 = v10.map(x=>x.split(",")).filter(x=>(x(2) == "action")).map(x=>(x(1).toInt,x(2)))

val v12 = v11.toDF("MovieId2","Tag")

val v13 = v5.join(v12, v5("MovieId") === v12("MovieId2")).drop("MovieId2")// v5 is the table with only movieid and ratings

val v14 = v13.groupBy("MovieId").avg()

val vf = v14.join(v12, v12("MovieId2") === v14("MovieId")).drop("MovieId2").drop("avg(MovieId)")

display(vf)

Q3-3

val v15 = spark.read.format("csv").option("header", "true").load("/FileStore/tables/movies1.csv")

val vc = v15.filter($"genres".contains("Thrill"))

val v18 = vc.join(vf, vc("movieId") === vf("MovieId")).drop("MovieId")

display(v18)