Question-1:

Logic:

1a) Loaded the ratings file and transformed the strings to Rating class(which consists of 4 fields i.e user_Id, movie_id, rating and timestamp) using map, and saved in the RDD named ratings.

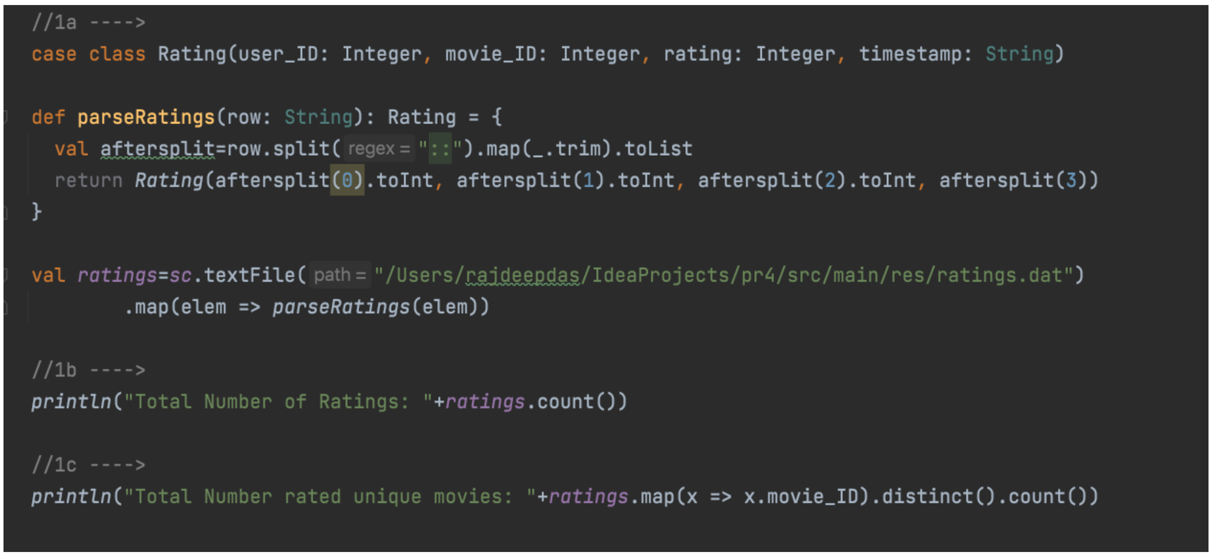
b) Using the count() method counted the total numbers of ratings in the RDD.

c) Using the map function changed the Ratings to movie_ids and after that counted the total number of unique movie ids using the distinct function

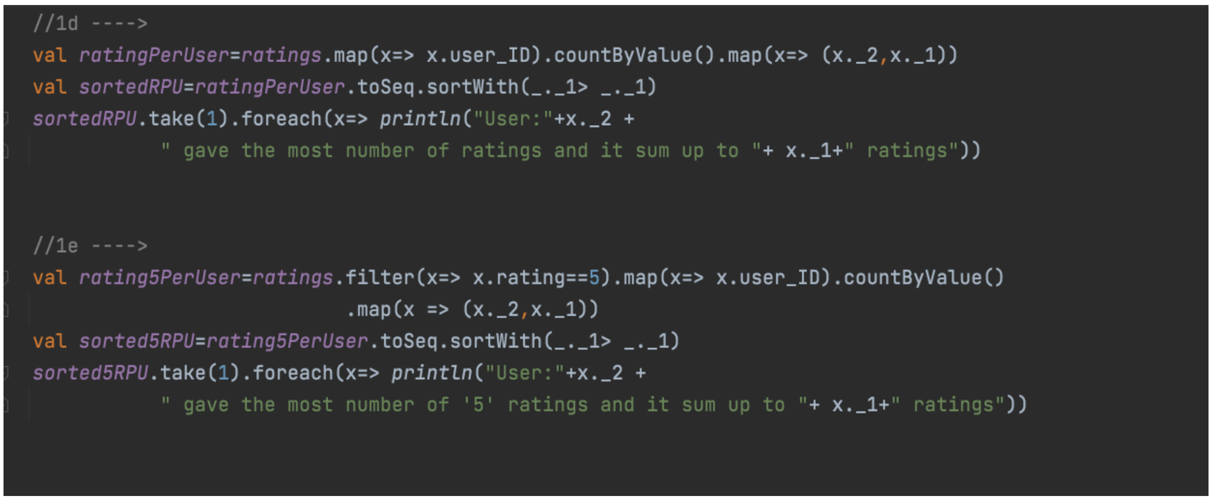
d) First using map transformed the ratings to movie id , then using countByValue(), for each movie_id, counted the number of times each movie has been rated . Then again swapped (movie_id, No. of ratings of movie_id) to (No. of ratings of movie_id, movie_id). Then sorted it in decreasing order. So the first element in the sequence is our answer.

e) First filtered the ratings with rating=5 and then used the same logic as of 1d problem.

Code Snippet:



```
//1a ---->
case class Rating(user_ID: Integer, movie_ID: Integer, rating: Integer, timestamp: String)

def parseRatings(row: String): Rating = {
  val aftersplit=row.split( regex = "::").map(_.trim).toList
  return Rating(aftersplit(0).toInt, aftersplit(1).toInt, aftersplit(2).toInt, aftersplit(3))
}

val ratings=sc.textFile( path = "/Users/rajdeepdas/IdeaProjects/pr4/src/main/res/ratings.dat")
        .map(elem => parseRatings(elem))

//1b ---->
println("Total Number of Ratings: "+ratings.count())

//1c ---->
println("Total Number rated unique movies: "+ratings.map(x => x.movie_ID).distinct().count())
```



```
//1d ---->
val ratingPerUser=ratings.map(x=> x.user_ID).countByValue().map(x=> (x._2,x._1))
val sortedRPU=ratingPerUser.toSeq.sortWith(_._1> _._1)
sortedRPU.take(1).foreach(x=> println("User:"+x._2 +
          " gave the most number of ratings and it sum up to "+ x._1+" ratings"))


//1e ---->
val rating5PerUser=ratings.filter(x=> x.rating==5).map(x=> x.user_ID).countByValue()
                              .map(x => (x._2,x._1))
val sorted5RPU=rating5PerUser.toSeq.sortWith(_._1> _._1)
sorted5RPU.take(1).foreach(x=> println("User:"+x._2 +
          " gave the most number of '5' ratings and it sum up to "+ x._1+" ratings"))
```

Output:

1a) Nothing

b) Total Number of Ratings: **1000209**

c) Total Number rated unique movies: **3706**

d) User:**4169** gave the most number of ratings and it sum up to **2314** ratings

e) User:**4277** gave the most number of '5' ratings and it sum up to **571** Ratings

Question-2:

2a) Loaded the movies and users file into respective RDD and used the count method to determine the number of records in each RDD

b) Splitted the strings in movies RDD to get the genre of each movie and we filtered out all the movies having “Comedy” genre and after that counted total number of such movies.

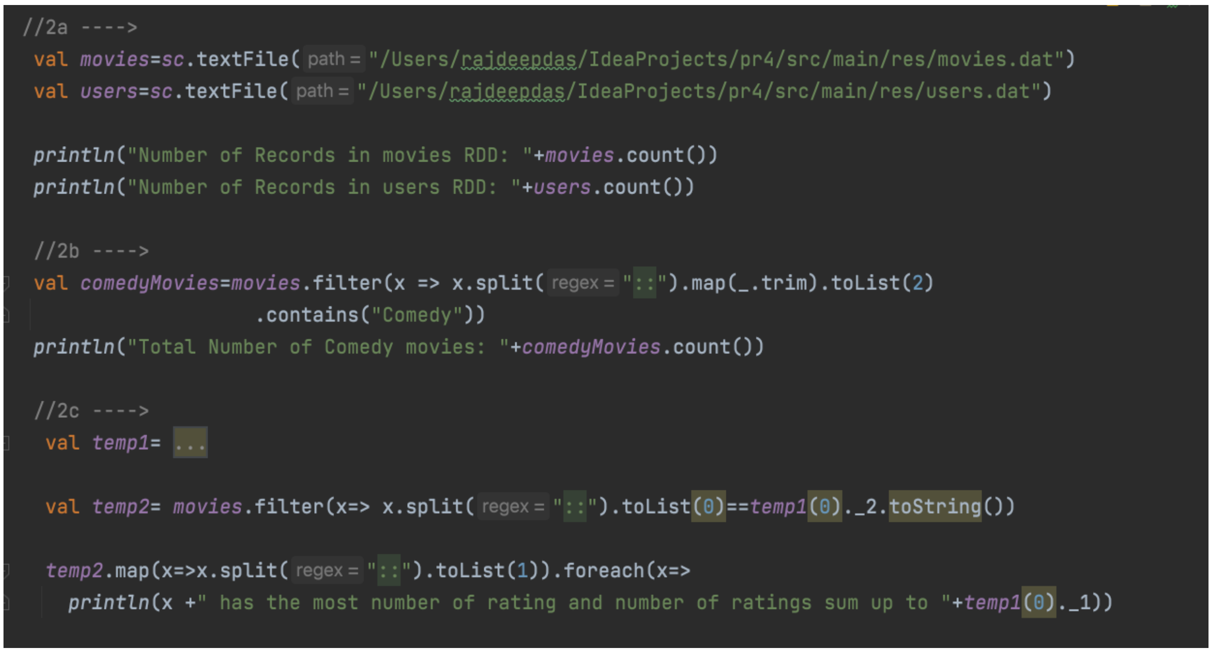
c) First joined the comedyMovies(RDD which only contain comedy movies) and users RDD using movies_id key. Then transformed each element in the RDD into movie_id and counted the number of times each movie_id is rated. Then sorted the sequence in decreasing order and picked the first one which is our answer.

e) First filtered out all the rating with movie_IDs 2858, 356, 2329 from the ratings RDD, then mapped the ratings to user_id and finally counted the total number of distinct user ids.

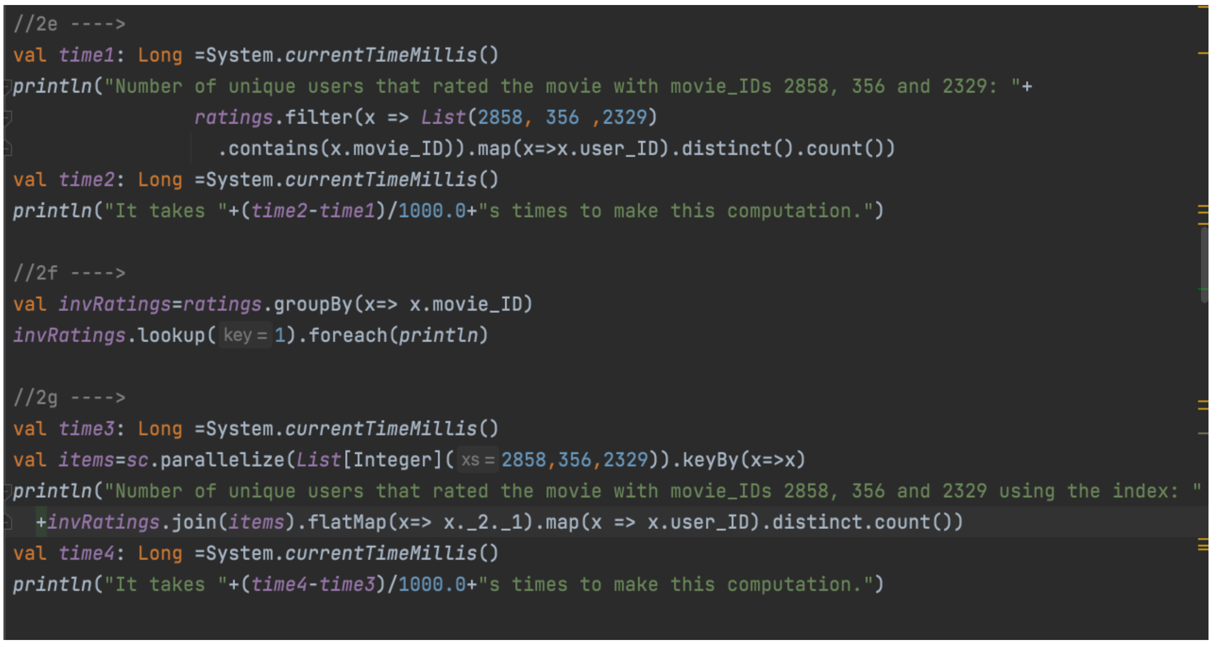
f) used groupBy() to create an inverted index on ratings using movie_ID field.

g) Used the inverted index created in above question f to find out all the user_ids that follows the given constraints and then counted total unique user ids.

Code Snippet:



```
//2a ---->
 val movies=sc.textFile( path = "/Users/rajdeepdas/IdeaProjects/pr4/src/main/res/movies.dat")
 val users=sc.textFile( path = "/Users/rajdeepdas/IdeaProjects/pr4/src/main/res/users.dat")

 println("Number of Records in movies RDD: "+movies.count())
 println("Number of Records in users RDD: "+users.count())

 //2b ---->
 val comedyMovies=movies.filter(x => x.split( regex = "::").map(_.trim).toList(2)
                    .contains("Comedy"))
 println("Total Number of Comedy movies: "+comedyMovies.count())

 //2c ---->
  val temp1= ...

  val temp2= movies.filter(x=> x.split( regex = "::").toList(0)==temp1(0)._2.toString())

  temp2.map(x=>x.split( regex = "::").toList(1)).foreach(x=>
    println(x +" has the most number of rating and number of ratings sum up to "+temp1(0)._1))
```



```
//2e ---->
val time1: Long =System.currentTimeMillis()
println("Number of unique users that rated the movie with movie_IDs 2858, 356 and 2329: "+
                ratings.filter(x => List(2858, 356 ,2329)
                  .contains(x.movie_ID)).map(x=>x.user_ID).distinct().count())
val time2: Long =System.currentTimeMillis()
println("It takes "+(time2-time1)/1000.0+"s times to make this computation.")

//2f ---->
val invRatings=ratings.groupBy(x=> x.movie_ID)
invRatings.lookup( key = 1).foreach(println)

//2g ---->
val time3: Long =System.currentTimeMillis()
val items=sc.parallelize(List[Integer]( xs = 2858,356,2329)).keyBy(x=>x)
println("Number of unique users that rated the movie with movie_IDs 2858, 356 and 2329 using the index: "
  +invRatings.join(items).flatMap(x=> x._2._1).map(x => x.user_ID).distinct.count())
val time4: Long =System.currentTimeMillis()
println("It takes "+(time4-time3)/1000.0+"s times to make this computation.")
```

Output:

2a) Number of Records in **movies** RDD: **3883**
Number of Records in **users** RDD: **6040**

b)Total Number of Comedy movies: **1200**

c) **American Beauty (1999)** has the most number of rating and number of ratings sum up to **3428**

e) Number of unique users that rated the movie with movie_IDs 2858, 356 and 2329: **4213**
It takes **1.293s** times to make this computation.

f) CompactBuffer(Rating(1,1,5,978824268), Rating(6,1,4,978237008), Rating(8,1,4,978233496), Rating(9,1,5,978225952), Rating(10,1,5,978226474), Rating(18,1,4,978154768)..................

g) Number of unique users that rated the movie with movie_IDs 2858, 356 and 2329 using the index: **4213**
It takes **3.923s** times to make this computation.

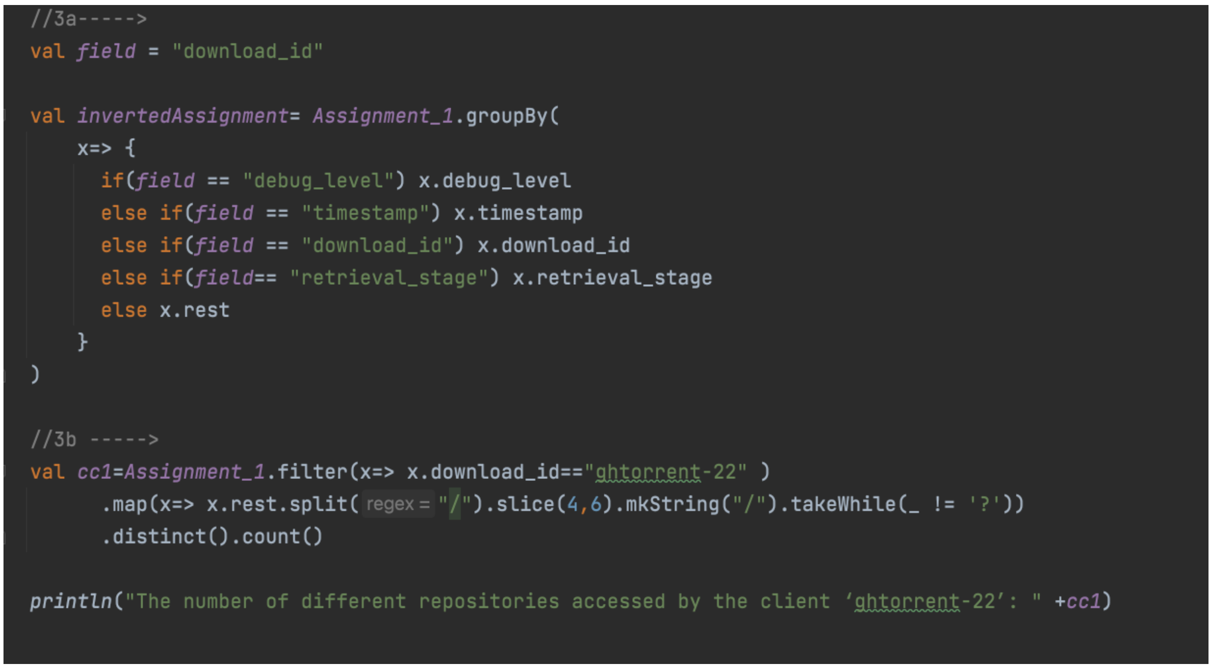
Question-3:

3a) created a function that creates inverted indexes based on the given field using if-else statements

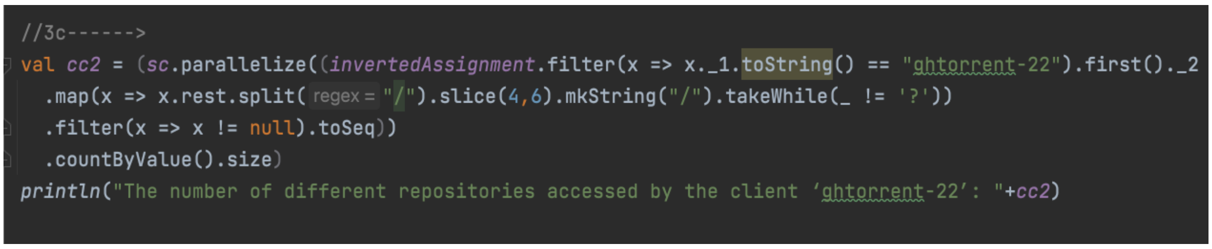
b) Filtered out all the datas from Assignment_1 RDD that have “ghtorrent-22” as its download id. Then counted total number of unique repositories.

c) Used the inverted index created in the question 3a and filtered out all the datas that have “ghtorrent-22” as its download id. Then counted total number of distinct repositories

Code Snippet:



```
//3a----->
val field = "download_id"

val invertedAssignment= Assignment_1.groupBy(
    x=> {
      if(field == "debug_level") x.debug_level
      else if(field == "timestamp") x.timestamp
      else if(field == "download_id") x.download_id
      else if(field== "retrieval_stage") x.retrieval_stage
      else x.rest
    }
)

//3b ----->
val cc1=Assignment_1.filter(x=> x.download_id=="ghtorrent-22" )
      .map(x=> x.rest.split( regex = "/").slice(4,6).mkString("/").takeWhile(_ != '?'))
      .distinct().count()

println("The number of different repositories accessed by the client ‘ghtorrent-22’: " +cc1)
```



```
//3c------>
val cc2 = (sc.parallelize((invertedAssignment.filter(x => x._1.toString() == "ghtorrent-22").first()._2
  .map(x => x.rest.split( regex = "/").slice(4,6).mkString("/").takeWhile(_ != '?'))
  .filter(x => x != null).toSeq))
  .countByValue().size)
println("The number of different repositories accessed by the client ‘ghtorrent-22’: "+cc2)
```

Output:

3a) Nothing

b) The number of different repositories accessed by the client ‘ghtorrent-22’: 4601

c) The number of different repositories accessed by the client ‘ghtorrent-22’: 4601

<u>FULL CODE LINK</u>:
https://drive.google.com/file/d/1TvLX3E3KcJWnesTt3GwpwRlzBgAtUCJU/view?usp=sharing