NAME: BISMAY PARIJA
ROLL NO.: 20CS30067

---

Question 1:

```
scala> val ratings = sc.textFile("data/data/data/ratings.dat").map(_.split("::"))
ratings: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[33] at map at <console>:23

scala> ratings.count()
res9: Long = 1000209
```

We employ **sc.textFile(#file-path)** to read the data file, and we utilize the **split** function to dissect the individual columns. To determine the count of entries, we invoke the **count** action.

Question 2:

(a)

```
scala> val movies = sc.textFile("data/data/data/movies.dat").map(_.split("::"))
movies: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[36] at map at <console>:23

scala> val users = sc.textFile("data/data/data/users.dat").map(_.split("::"))
users: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[39] at map at <console>:23

scala> movies.count()
res10: Long = 3883

scala> users.count()
res11: Long = 6040
```

We utilize **sc.textFile(#file-path)** to read the data file and employ the **count** action to determine the total number of entries.

(b)

```
scala> val comedies = movies.filter(_(2).contains("Comedy"))
comedies: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[40] at filter at <console>:23

scala> comedies.count()
res12: Long = 1200
```

We employ the **filter** transformation to single out the comedy films by checking for their presence.

Question 3:

(c)

```
scala>  val ratingsByMovieID = ratings.map(i => (i(1),i(0)))
ratingsByMovieID: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[44] at map at <console>:23

scala>  val movies_new = movies.map(i => (i(0),i(1)))
movies_new: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[45] at map at <console>:23

scala>  val combined = ratingsByMovieID.join(movies_new)
combined: org.apache.spark.rdd.RDD[(String, (String, String))] = MapPartitionsRDD[48] at join at <console>:24

scala>  val ratingsByMovieID = ratings.map(i => (i(1),i(0)))
ratingsByMovieID: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[49] at map at <console>:23

scala>  val movies_new = movies.map(i => (i(0),i(1)))
movies_new: org.apache.spark.rdd.RDD[(String, String)] = MapPartitionsRDD[50] at map at <console>:23

scala>  val combined = ratingsByMovieID.join(movies_new).groupByKey()
combined: org.apache.spark.rdd.RDD[(String, Iterable[(String, String)])] = MapPartitionsRDD[54] at groupByKey at <console>:24
```

```
scala> val maxRatedComedy = combined.reduce((a,b) => { if (a._2.size > b._2.size) a else b})
maxRatedComedy: (String, Iterable[(String, String)]) = (2858,CompactBuffer((2,American Beauty (1999)), (3,American Beauty (1999)), (5,American Beauty (1999)), (6,Amer
ican Beauty (1999)), (8,American Beauty (1999)), (9,American Beauty (1999)), (10,American Beauty (1999)), (11,American Beauty (1999)), (14,American Beauty (1999)), (1
5,American Beauty (1999)), (17,American Beauty (1999)), (18,American Beauty (1999)), (19,American Beauty (1999)), (20,American Beauty (1999)), (22,American Beauty (19
99)), (23,American Beauty (1999)), (24,American Beauty (1999)), (26,American Beauty (1999)), (27,American Beauty (1999)), (28,American Beauty (1999)), (30,American Be
auty (1999)), (32,American Beauty (1999)), (33,American Beauty (1999)), (34,American Beauty (1999)), (35,American Beauty (1999)), (36...

scala> maxRatedComedy._2.size
res13: Int = 3428
```

We transform the ratings RDD by setting MovieID as the primary key. Afterward, we perform a join operation between the ratings RDD and the movies RDD. Subsequently, we group the data by the MovieID key, resulting in an iterable containing ratings for each distinct movie. To identify the movie with the highest count of ratings, we employ the filter action.