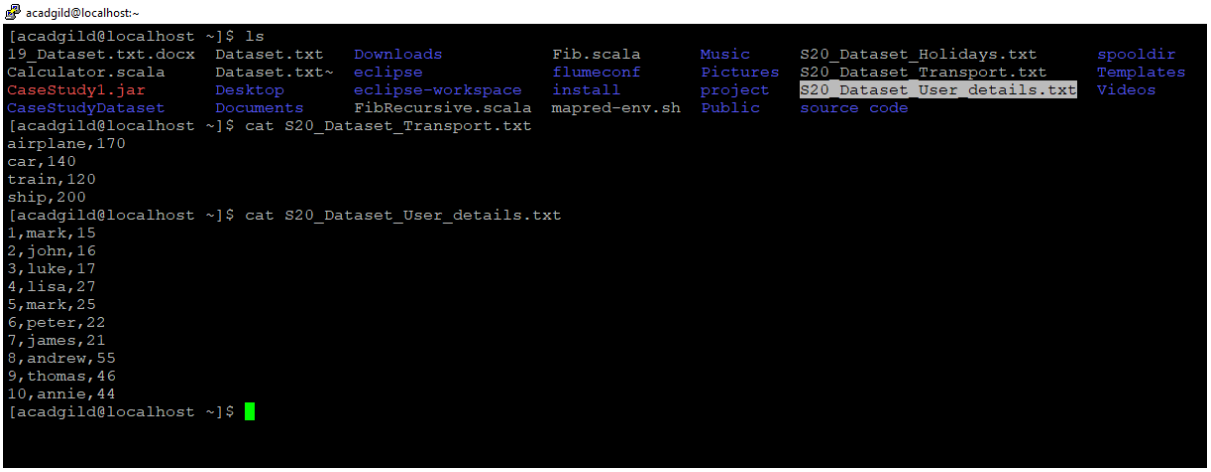
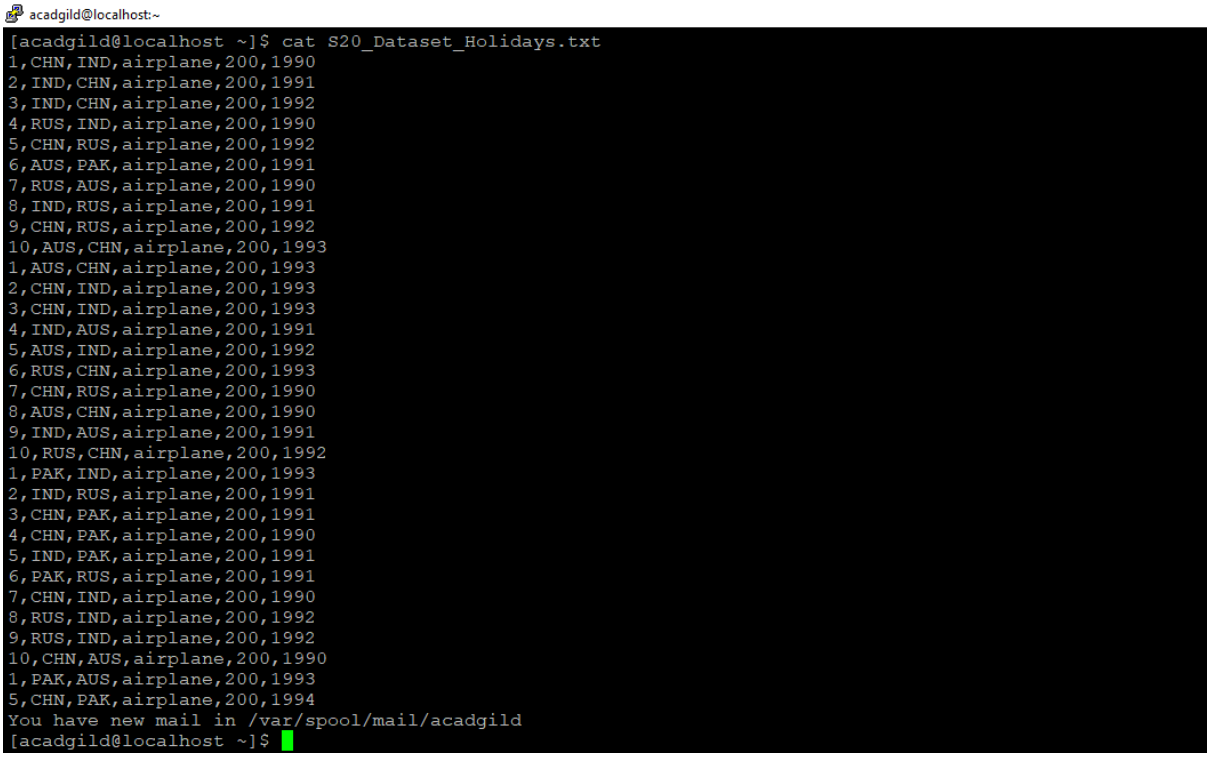
ASSIGNMENT

Task1

a- Load the dataset to hadoop.

➔ Command used: hadoop fs –put s20





b-> Creating the BaseRDD and loading the textfile using sc(sparkContext)

➔ Command used: val BaseRDD = sc.textFile("s20/s20_Dataset_Holidays.txt")

➔ BaseRDD.collect().foreach(println)

```
acadgild@localhost:~
scala> sc
res0: org.apache.spark.SparkContext = org.apache.spark.SparkContext@6313b441

scala> val BaseRDD = sc.textFile("s20/S20_Dataset_Holidays.txt")
BaseRDD: org.apache.spark.rdd.RDD[String] = s20/S20_Dataset_Holidays.txt MapPartitionsRDD[1] at textFile at <console>:24

scala> BaseRDD.collect().foreach(println)
1,CHN,IND,airplane,200,1990
2,IND,CHN,airplane,200,1991
3,IND,CHN,airplane,200,1992
4,RUS,IND,airplane,200,1990
5,CHN,RUS,airplane,200,1992
6,AUS,PAK,airplane,200,1991
7,RUS,AUS,airplane,200,1990
8,IND,RUS,airplane,200,1991
9,CHN,RUS,airplane,200,1992
10,AUS,CHN,airplane,200,1993
1,AUS,CHN,airplane,200,1993
2,CHN,IND,airplane,200,1993
3,CHN,IND,airplane,200,1993
4,IND,AUS,airplane,200,1991
5,AUS,IND,airplane,200,1992
6,RUS,CHN,airplane,200,1993
7,CHN,RUS,airplane,200,1990
8,AUS,CHN,airplane,200,1990
9,IND,AUS,airplane,200,1991
10,RUS,CHN,airplane,200,1992
1,PAK,IND,airplane,200,1993
2,IND,RUS,airplane,200,1991
3,CHN,PAK,airplane,200,1991
4,CHN,PAK,airplane,200,1990
5,IND,PAK,airplane,200,1991
6,PAK,RUS,airplane,200,1991
7,CHN,IND,airplane,200,1990
8,RUS,IND,airplane,200,1992
9,RUS,IND,airplane,200,1992
10,CHN,AUS,airplane,200,1990
1,PAK,AUS,airplane,200,1993
5,CHN,PAK,airplane,200,1994

scala>
```

1.1 What is the distribution of the total number of air-travellers per year

➔ Command used:

➔ val splitRDD = BaseRDD.map(x=>(x.split(",")(5).toInt,1))

➔ val countSplit = splitRDD.reduceByKey((x,y)=>(x+y))

➔ countSplit.foreach(println)

```
acadgild@localhost:~

scala> val splitRDD = BaseRDD.map(x=>(x.split(",")(5).toInt,1))
splitRDD: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[2] at map at <console>:26

scala> val countSplit = splitRDD.reduceByKey((x,y)=>(x+y))
countSplit: org.apache.spark.rdd.RDD[(Int, Int)] = ShuffledRDD[3] at reduceByKey at <console>:28

scala> countSplit.foreach(println)
(1994,1)
(1992,7)
(1990,8)
(1991,9)
(1993,7)

scala>
```

1.2 What is the total air distance covered by each user per year

➔ Command used:

➔ val SplitRDD = BaseRDD.map(x=>((x.split(“,”)(0),x.split(“,”)(5)),x.split(“,”)(4).toInt))

➔ val distRDD = SplitRDD.reduceByKey((x,y)=>(x+y))

➔ distRDD.foreach(println)

acadgild@localhost:~

```
scala> val SplitRDD = BaseRDD.map(x=>((x.split(",")(0),x.split(",")(5)),x.split(",")(4).toInt))
SplitRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = MapPartitionsRDD[4] at map at <console>:26

scala> val distRDD = SplitRDD.reduceByKey((x,y)=> (x+y))
distRDD: org.apache.spark.rdd.RDD[((String, String), Int)] = ShuffledRDD[5] at reduceByKey at <console>:28

scala> distRDD.foreach(println)
((3,1992),200)
((3,1993),200)
((5,1991),200)
((6,1991),400)
((10,1993),200)
((5,1992),400)
((8,1991),200)
((8,1990),200)
((1,1993),600)
((5,1994),200)
((2,1993),200)
((2,1991),400)
((4,1990),400)
((10,1992),200)
((3,1991),200)
((1,1990),200)
((10,1990),200)
((6,1993),200)
((9,1992),400)
((8,1992),200)
((7,1990),600)
((9,1991),200)
((4,1991),200)

scala>
```

1.3 Which user has travelled the largest distance till date

➔ Command used:

➔ val userRDD = BaseRDD.map(x=>(x.split(“,”)(0),x.split(“,”)(4).toInt))

➔ val totalDistRDD = userRDD.reduceByKey((x,y=>(x+y))

➔ val maxRDD = totalDistRDD.takeOrdered(1)

➔ maxRDD.foreach(println)

acadgild@localhost:~

```
scala> val userRDD = BaseRDD.map(x=>(x.split(",")(0),x.split(",")(4).toInt))
userRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[6] at map at <console>:26

scala> val totalDistRDD = userRDD.reduceByKey((x,y)=> (x+y))
totalDistRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[7] at reduceByKey at <console>:28

scala> val maxRDD = totalDistRDD.takeOrdered(1)
maxRDD: Array[(String, Int)] = Array((1,800))

scala> maxRDD.foreach(println)
(1,800)

scala>
```

1.4 What is the most preferred destination for all users

➔ Command used:

➔ val DestRDD = BaseRDD.map(x=>(x.split(“,”)(2),1))

➔ val destReduceRDD = DestRDD.reduceByKey((x,y)=>(x+y))

➔ val maxDestRDD = destReduceRDD.takeOrdered(1)(Ordering[Int].reverse.on(_._2))

➔ maxDestRDD.foreach(println)

acadgild@localhost:~

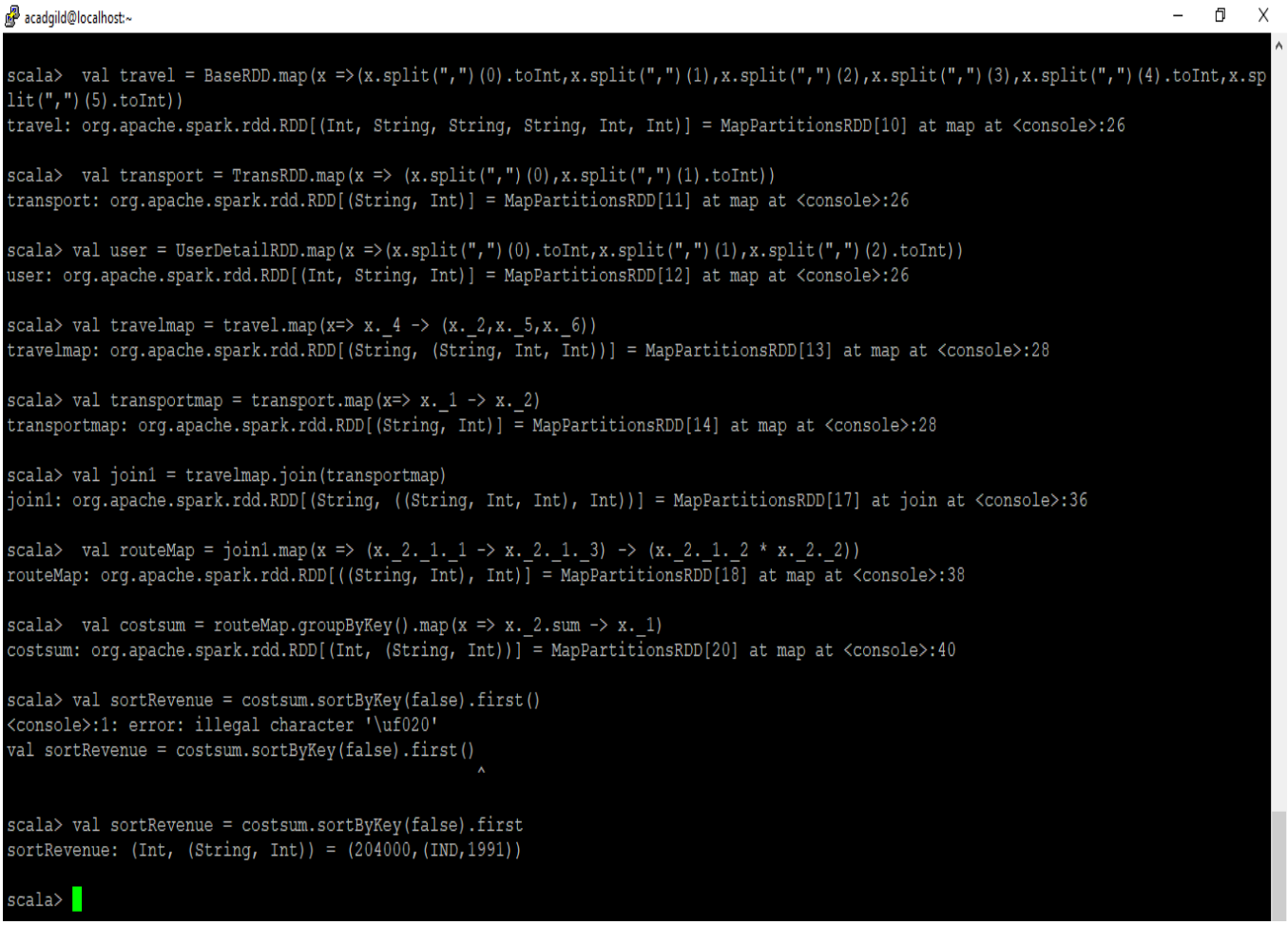
```
scala> val DestRDD = BaseRDD.map(x=>(x.split(",")(2),1))
DestRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[9] at map at <console>:26

scala> val destReduceRDD = DestRDD.reduceByKey((x,y)=>(x+y))
destReduceRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[10] at reduceByKey at <console>:28

scala> val maxDestRDD = destReduceRDD.takeOrdered(1)
maxDestRDD: Array[(String, Int)] = Array((AUS,5))

scala> maxDestRDD.foreach(println)
(AUS,5)

scala> val maxDestRDD = destReduceRDD.takeOrdered(1)(Ordering[Int].rever
se.on(_._2))
maxDestRDD: Array[(String, Int)] = Array((IND,9))

scala> maxDestRDD.foreach(println)
(IND,9)

scala>
```

1.5 Which route is generating the most revenue per year

-> Load the data to RDD

acadgild@localhost:~

```
scala> val BaseRDD = sc.textFile("s20/S20_Dataset_Holidays.txt")
BaseRDD: org.apache.spark.rdd.RDD[String] = s20/S20_Dataset_Holidays.txt MapPart
itionsRDD[5] at textFile at <console>:24

scala> val TransRDD = sc.textFile("s20/S20_Dataset_Transport.txt")
TransRDD: org.apache.spark.rdd.RDD[String] = s20/S20_Dataset_Transport.txt MapPa
rtitionsRDD[7] at textFile at <console>:24

scala> TransRDD.collect().foreach(println)
airplane,170
car,140
train,120
ship,200

scala> val UserDetailRDD = sc.textFile("s20/S20_Dataset_User_details.txt")
UserDetailRDD: org.apache.spark.rdd.RDD[String] = s20/S20_Dataset_User_details.t
xt MapPartitionsRDD[9] at textFile at <console>:24

scala> UserDetailRDD.collct().foreach(println)
<console>:27: error: value collct is not a member of org.apache.spark.rdd.RDD[St
ring]
       UserDetailRDD.collct().foreach(println)
                     ^

scala> UserDetailRDD.collect().foreach(println)
1,mark,15
2,john,16
3,luke,17
4,lisa,27
5,mark,25
6,peter,22
7,james,21
8,andrew,55
9,thomas,46
10,annie,44
```



1.6 What is the total amount spent by every user on air-travel per year

Sol: Below is the code used:-

```
val userMap = travel.map(x => x._4 -> (x._1,x._5,x._6))

val amtMap = userMap.join(transportmap)

val spendMap = amtMap.map(x => (x._2._1._1, x._2._1._3) -> (x._2._1._2 * x._2._2))

val total = spendMap.groupByKey().map(x => x._1 -> x._2.sum)

total.foreach(println)
```

```
acadgild@localhost:~

scala> val userMap = travel.map(x => x._4 -> (x._1,x._5,x._6))
userMap: org.apache.spark.rdd.RDD[(String, (Int, Int, Int))] = MapPartitionsRDD[22] at map at <console>:28

scala> val amtMap = userMap.join(transportmap)
amtMap: org.apache.spark.rdd.RDD[(String, ((Int, Int, Int), Int))] = MapPartitionsRDD[25] at join at <console>:36

scala> val spendMap = amtMap.map(x => (x._2._1._1, x._2._1._3) -> (x._2._1._2 * x._2._2))
spendMap: org.apache.spark.rdd.RDD[((Int, Int), Int)] = MapPartitionsRDD[26] at map at <console>:38

scala> val total = spendMap.groupByKey().map(x => x._1 -> x._2.sum)
total: org.apache.spark.rdd.RDD[((Int, Int), Int)] = MapPartitionsRDD[28] at map at <console>:40

scala> total.foreach(println)
((2,1993),34000)
((6,1993),34000)
((10,1993),34000)
((10,1992),34000)
((2,1991),68000)
((4,1990),68000)
((10,1990),34000)
((5,1992),68000)
((4,1991),34000)
((1,1993),102000)
((9,1992),68000)
((5,1991),34000)
((3,1993),34000)
((1,1990),34000)
((8,1990),34000)
((7,1990),102000)
((6,1991),68000)
((5,1994),34000)
((3,1991),34000)
((9,1991),34000)
((3,1992),34000)
((8,1991),34000)
((8,1992),34000)

scala>
```

1.7 Considering age groups of < 20 , 20-35, 35 > ,Which age group is travelling the most every year

Sol: val AgeMap = user.map(x => x._1 -> {if(x._3<20) "20" else if(x._3>35) "35" else "20-35" })

val UIDMap = travel.map(x => x._1 -> 1)

val joinMap2 = joinMap.map(x => x._2._1 -> x._2._2)

val groupKey = joinMap2.groupByKey.map(x => x._1 -> x._2.sum)

val maxVal = groupKey.sortBy(x => -x._2).first()

```
acadgild@localhost:~

scala> val AgeMap = user.map(x => x._1 -> {if(x._3<20) "20" else if(x._3>35) "35" else "20-35" })
AgeMap: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[29] at map at <console>:28

scala> val UIDMap = travel.map(x => x._1 -> 1)
UIDMap: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[30] at map at <console>:28

scala> val joinMap = AgeMap.join(UIDMap)
<console>:1: error: illegal character '\uf020'
val joinMap = AgeMap.join(UIDMap)
                          ^

scala> val joinMap = AgeMap.join(UIDMap)
joinMap: org.apache.spark.rdd.RDD[(Int, (String, Int))] = MapPartitionsRDD[33] at join at <console>:36

scala>  val joinMap2 = joinMap.map(x => x._2._1 -> x._2._2)
joinMap2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[34] at map at <console>:38

scala> val groupKey = joinMap2.groupByKey.map(x => x._1 -> x._2.sum)
groupKey: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[36] at map at <console>:40

scala> val maxVal = groupKey.sortBy(x => -x._2).first()
maxVal: (String, Int) = (20-35,13)

scala>
```