```
!pip install pyspark==3.0.1
```

```
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
```

```
transRDD = spark.sparkContext.textFile("trans.txt")
custRDD = spark.sparkContext.textFile("cust.txt")
```

```
#cau 1: Show IDs and number of transactions of each customer
transRDD.map(lambda line: line.split(',')).map(lambda array: (array[2],'1')).reduceByKey(lambda a,b : int(a)+int(b)).collect()
#transRDD.map(lambda x : x.split(",")).map(lambda x : (x[2],1)).collect()
```

```
[('4000004', 5),
 ('4000007', 6),
 ('4000008', 10),
 ('4000001', 8),
 ('4000002', 6),
 ('4000003', 3),
 ('4000005', 5),
 ('4000006', 5),
 ('4000009', 6),
 ('4000010', 6)]
```

```
#cau 2: Show IDs and number of transactions of each customer, sorted by customer ID
transRDD.map(lambda line: line.split(',')).map(lambda array: (array[2],'1')).reduceByKey(lambda a,b : int(a)+int(b)).sortBy(lambda x: x[1]).collect()
```

```
[('4000003', 3),
 ('4000004', 5),
 ('4000005', 5),
 ('4000006', 5),
 ('4000007', 6),
 ('4000002', 6),
 ('4000009', 6),
 ('4000010', 6),
 ('4000001', 8),
 ('4000008', 10)]
```

```
#câu 3: Show IDs and total cost of transactions of each customer, sorted by total cost
transRDD.map(lambda x: x.split(",")).map(lambda x: (x[2],x[3])).reduceByKey(lambda x1,x2:float(x1)+float(x2) ).sortBy(lambda x: x[1]).collect()
```

```
[('4000005', 325.15),
 ('4000004', 337.06),
 ('4000010', 447.09000000000003),
 ('4000009', 457.83),
 ('4000003', 527.5899999999999),
 ('4000006', 539.3800000000001),
 ('4000001', 651.0500000000001),
 ('4000007', 699.55),
 ('4000002', 706.97),
 ('4000008', 859.42)]
```

```
#câu 4: Show ID, number of transactions, and total cost for each customer, sorted by customer ID
transRDD.map(lambda x: x.split(",")).map(lambda x: (x[2],(1,x[3]))).reduceByKey(lambda x1,x2: (  x1[0]+x2[0],   float(x1[1]) + float(x2[1])   )).sortBy(lambda x: x[1][1] ).collect()
```

```
[('4000005', (5, 325.15)),
 ('4000004', (5, 337.06)),
 ('4000010', (6, 447.09000000000003)),
 ('4000009', (6, 457.83)),
 ('4000003', (3, 527.5899999999999)),
 ('4000006', (5, 539.3800000000001)),
 ('4000001', (8, 651.0500000000001)),
 ('4000007', (6, 699.55)),
 ('4000002', (6, 706.97)),
 ('4000008', (10, 859.42))]
```

```
#câu 5: Show name, number of transactions, and total cost for each customer, sorted by totall cost
custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1])).collect()


transRDD.map(lambda x: x.split(",")).map(lambda x: (x[2],(1,x[3]))).reduceByKey(lambda x1,x2: (  x1[0]+x2[0],   float(x1[1]) + float(x2[1])   )).join( custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1])) ).map(lambda x: (x[1][1], x[1][0][0] , x[1][0][1]  )    ).collect()
```

```
[('Gretchen', 5, 337.06),
 ('Elsie', 6, 699.55),
 ('Sherri', 3, 527.5899999999999),
 ('Malcolm', 6, 457.83),
 ('Hazel', 10, 859.42),
 ('Kristina', 8, 651.0500000000001),
 ('Paige', 6, 706.97),
 ('Karen', 5, 325.15),
 ('Patrick', 5, 539.3800000000001),
 ('Dolores', 6, 447.09000000000003)]
```

```
#6 Show name, game types played by each customer

transRDD.map(lambda x: x.split(",")).map(lambda x:(x[2],x[4])).distinct().reduceByKey(lambda x1,x2: x1+";" + x2).join(custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1]))).map(lambda x: (x[1][1]   , x[1][0])).collect()
```

```
[('Elsie', 'Team Sports;Exercise & Fitness;Outdoor Recreation'),
 ('Gretchen', 'Indoor Games;Water Sports;Outdoor Recreation'),
 ('Sherri', 'Gymnastics;Outdoor Recreation;Water Sports'),
 ('Malcolm',
  'Gymnastics;Combat Sports;Outdoor Play Equipment;Indoor Games;Water Sports'),
# 7 Show ID, name, game types of all players who play 5 or more game types
transRDD.map(lambda x: x.split(",")).map(lambda x:(x[2],x[4])).distinct().reduceByKey(lambda x1,x2: x1 + ";" +x2).map(lambda x: (x[0], x[1].split(";")  )).filter(lambda x: len(x[1])>4).join(custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1]))).map(lambda x: (x[1][1],x[1][0])).collect()
```

```
     [('Malcolm',
       ['Gymnastics',
        'Combat Sports',
        'Outdoor Play Equipment',
        'Indoor Games',
        'Water Sports']),
      ('Hazel'
```

```
#8 Show name of all distinct players of each game types

transRDD.map(lambda x: x.split(",")).map(lambda x:(x[2],x[4])).distinct().join(custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],x[1]))).map(lambda x: (x[1][0],x[1][1])).reduceByKey(lambda x1,x2: x1+";"+x2).collect()
```

```
     [('Water Sports', 'Gretchen;Sherri;Malcolm;Hazel;Patrick;Kristina;Paige'),
      ('Winter Sports', 'Patrick;Kristina'),
      ('Gymnastics', 'Sherri;Malcolm;Dolores;Kristina'),
      ('Team Sports', 'Elsie;Hazel;Dolores;Paige;Karen'),
      ('Exercise & Fitness', 'Elsie;Dolores;Kristina;Paige;Karen')
```

```
#9 Show all game types which don't have player under 40

custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],float(x[3]))).collect()
```

```
     [('4000001', 55.0),
      ('4000002', 74.0),
      ('4000003', 34.0),
      ('4000004', 66.0),
      ('4000005', 74.0),
      ('4000006', 42.0),
      ('4000007', 43.0),
      ('4000008', 63.0),
      ('4000009', 39.0),
      ('4000010', 60.0)]
```

```
transRDD.map(lambda x: x.split(",")).map(lambda x:(x[2],x[4])).distinct().join(custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],float(x[3])))).map(lambda x: x[1]).reduceByKey(min).filter(lambda x:x[1]>40).collect()
```

```
     [('Winter Sports', 42.0),
      ('Team Sports', 43.0),
      ('Exercise & Fitness', 43.0),
      ('Games', 60.0),
      ('Puzzles', 74.0),
      ('Air Sports', 74.0),
      ('Jumping', 42.0)]
```

```
#10 show average age of players of all gametypes
transRDD.map(lambda x: x.split(",")).map(lambda x:(x[2],x[4])).distinct().join(custRDD.map(lambda x: x.split(",")).map(lambda x: (x[0],float(x[3])))).map(lambda x: (x[1][0], (1,x[1][1]))).reduceByKey(lambda x1,x2:  (x1[0]+x2[0] ,x1[1]+x2[1]  )       ).map(lambda x: (x[0] , x[1][1]/x[1][0]   ) ).collect()
```

```
     [('Water Sports', 53.285714285714285),
      ('Winter Sports', 48.5),
      ('Gymnastics', 47.0),
      ('Team Sports', 62.8),
      ('Exercise & Fitness', 61.2),
      ('Indoor Games', 52.5),
      ('Outdoor Play Equipment', 54.5),
```