Task 1.

Assume that the spark dataframe is df with the columns: UserId, ReviewTime, ReviewText, ReviewRating, NumberOfProductBought.

```
% now create udf functions to use the built-in functions

@udf("String")

def getSentiment_udf(ReviewText):

        return getSentiment(ReviewText)

@udf("String")

def getUserCountry_udf(UserId):

        return getUserCountry(UserId)

@udf("String")

def getUserCity_udf(UserId):

        return getUserCity(UserId)

@udf("Integer")

def getYear _udf(UserId):

        return getYear(ReviewTime)

@udf("Integer")

def getMonth _udf(UserId):

        return getMonth(ReviewTime)

@udf("Integer")

def getDay _udf(UserId):

        return getDay(ReviewTime)


% now add the new columns using the udf functions as follows

df = df.withColumn("SentimentPolarity", getSentiment_udf("ReviewText")
```

df = df.withColumn(“UserCountry”, getUserCountry_udf(“UserId”)

….

Task 2.

There are two ways:

1. using dataframe groupBy
   dfNumProductsBought = df.select(“NumberOfProductBought”)

   dfNumProductsBought.groupBy().sum().collect()[0][0]

2. using rdd
   dfNumProductsBought.rdd.map(lambda x: (1,x[0])).reduceByKey(lambda x,y: x + y).collect()[0][1]

Task 3.

1. The average overall ReviewRating per user
   from pyspark.sql import functions as F
   df.agg(F.mean("ReviewRating"), F.count("ReviewRating")).collect()[0][0]
2. The total number of Reviews by:
   a. Overall sentiment polarity
      i. df.groupBy(“SentimentPolarity).count().show()
      ii. df.groupBy(“UserCountry”).count().show(),
      df.groupBy(“UserId”).count().show()