Simran Kaur

DDA Lab

311443

Exercise 8

Exercise 1: Apache Spark Basics

In [340…
```
import pyspark
from pyspark.sql import *
from pyspark.sql.functions import *
import numpy as np
import matplotlib.pyplot as plt
from ast import literal_eval
from pyspark.sql.functions import *
from pyspark.sql.types import StringType
from pyspark.sql.types import *
```

In [2]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("SparkByExamples.com").getOrCreate()
```

In [3]:
```
spark
```

Out[3]:

SparkSession - hive

SparkContext

Spark UI

Version	v3.1.3
Master	local[*]
AppName	PySparkShell

Part a) Basic Operations on Resilient Distributed Dataset (RDD)

In [32]:
```
from pyspark import SparkContext
```

In [74]:
```
a = ["spark", "rdd", "python", "context", "create", "class"]
b = ["operation", "apache", "scala", "lambda","parallel","partition"]
```

1. Perform rightOuterJoin and fullOuterJoin operations between

a and b. Briefly explain your solution.

In [82]:
```
a_ = []
for i, ele in enumerate(a):
    a_.append((ele, i))
a_
```

Out[82]:
```
[('spark', 0),
 ('rdd', 1),
 ('python', 2),
 ('context', 3),
 ('create', 4),
 ('class', 5)]
```

In [83]:
```
b_ = []
for i, ele in enumerate(b):
    b_.append((ele, i))
b_
```

Out[83]:
```
[('operation', 0),
 ('apache', 1),
 ('scala', 2),
 ('lambda', 3),
 ('parallel', 4),
 ('partition', 5)]
```

In [86]:
```
a_new = sc.parallelize(a_)
b_new = sc.parallelize(b_)
rightOuterJoin = a_new.rightOuterJoin(b_new).collect()
```

In [87]:
```
rightOuterJoin
```

Out[87]:
```
[('lambda', (None, 3)),
 ('scala', (None, 2)),
 ('operation', (None, 0)),
 ('partition', (None, 5)),
 ('parallel', (None, 4)),
 ('apache', (None, 1))]
```

In [88]:
```
fullOuterJoin = a_new.fullOuterJoin(b_new).collect()
fullOuterJoin
```

Out[88]:
```
[('python', (2, None)),
 ('create', (4, None)),
 ('lambda', (None, 3)),
 ('scala', (None, 2)),
 ('operation', (None, 0)),
 ('partition', (None, 5)),
 ('spark', (0, None)),
 ('context', (3, None)),
 ('parallel', (None, 4)),
 ('class', (5, None)),
 ('rdd', (1, None)),
 ('apache', (None, 1))]
```

2. Using map and reduce functions to count how many times the character "s" appears in all a and b.

In [96]:

```
a_rdd = sc.parallelize(a)
b_rdd = sc.parallelize(b)
```

In [90]:
```
def count_s(x):
    count = 0
    for ch in x:
        if ch == 's':
            count += 1
    return count
```

In [92]:
```
a_map = a_rdd.map(count_s)
a_red = a_map.reduce(lambda a, b : a+b)
b_map = b_rdd.map(count_s)
b_red = b_map.reduce(lambda a, b : a+b)
```

In [93]:
```
print(f'Number of times "s" appears in all a and b : {a_red + b_red}')
```

```
Number of times "s" appears in all a and b : 4
```

3. Using aggregate function to count how many times the character "s" appears in all a and b.

In [110…
```
agg = a_rdd.union(b_rdd).flatMap(lambda x: x).filter(lambda x: x == 's'). \
        aggregate(0, (lambda i, value: i + 1) ,(lambda a, b: (a + b)))
```

In [111…
```
agg
```

Out[111… 4

Part b) Basic Operations on DataFrames

Use dataset students.json (download from learnweb) for this exercise. First creating DataFrames from the dataset and do several tasks as follows:

In [331…
```
students = sc.textFile('students.json').collect()
```

In [332…
```
studentDf = spark.read.json(sc.parallelize(students))
studentDf.show()
```

```
+------------------+------------------+----------+---------+------+----+
|            course|               dob|first_name|last_name|points|s_id|
+------------------+------------------+----------+---------+------+----+
|Humanities and Art|  October 14, 1983|      Alan|      Joe|    10|   1|
|  Computer Science|September 26, 1980|    Martin|  Genberg|    17|   2|
|    Graphic Design|     June 12, 1982|     Athur|   Watson|    16|   3|
|    Graphic Design|     April 5, 1987|  Anabelle|  Sanberg|    12|   4|
|        Psychology|  November 1, 1978|      Kira| Schommer|    11|   5|
|          Business|  17 February 1981| Christian|   Kiriam|    10|   6|
|  Machine Learning|    1 January 1984|   Barbara|  Ballard|    14|   7|
|     Deep Learning|  January 13, 1978|      John|     null|    10|   8|
|  Machine Learning|  26 December 1989|    Marcus|   Carson|    15|   9|
```

```
|           Physics|  30 December 1987|     Marta|   Brooks|    11|  10|
|    Data Analytics|     June 12, 1975|     Holly| Schwartz|    12|  11|
|  Computer Science|      July 2, 1985|     April|    Black|  null|  12|
|  Computer Science|     July 22, 1980|     Irene|  Bradley|    13|  13|
|        Psychology|   7 February 1986|      Mark|    Weber|    12|  14|
|       Informatics|      May 18, 1987|     Rosie|   Norman|     9|  15|
|          Business|   August 10, 1984|    Martin|   Steele|     7|  16|
|  Machine Learning|  16 December 1990|     Colin| Martinez|     9|  17|
|    Data Analytics|              null|   Bridget|    Twain|     6|  18|
|          Business|      7 March 1980|   Darlene|    Mills|    19|  19|
|    Data Analytics|      June 2, 1985|   Zachary|     null|    10|  20|
+------------------+------------------+----------+---------+------+----+
```

1. Replace the null value(s) in column points by the mean of all points.

In [334…

```
points_mean = studentDf.agg({'points': 'mean'}).collect()
studentDf = studentDf.na.fill(points_mean[0][0], ['points'])
studentDf.show()
```

```
+------------------+------------------+----------+---------+------+----+
|            course|               dob|first_name|last_name|points|s_id|
+------------------+------------------+----------+---------+------+----+
|Humanities and Art|  October 14, 1983|      Alan|      Joe|    10|   1|
|  Computer Science|September 26, 1980|    Martin|  Genberg|    17|   2|
|    Graphic Design|     June 12, 1982|     Athur|   Watson|    16|   3|
|    Graphic Design|     April 5, 1987|  Anabelle|  Sanberg|    12|   4|
|        Psychology|  November 1, 1978|      Kira| Schommer|    11|   5|
|          Business|  17 February 1981| Christian|   Kiriam|    10|   6|
|  Machine Learning|    1 January 1984|   Barbara|  Ballard|    14|   7|
|     Deep Learning|  January 13, 1978|      John|     null|    10|   8|
|  Machine Learning|  26 December 1989|    Marcus|   Carson|    15|   9|
|           Physics|  30 December 1987|     Marta|   Brooks|    11|  10|
|    Data Analytics|     June 12, 1975|     Holly| Schwartz|    12|  11|
|  Computer Science|      July 2, 1985|     April|    Black|    11|  12|
|  Computer Science|     July 22, 1980|     Irene|  Bradley|    13|  13|
|        Psychology|   7 February 1986|      Mark|    Weber|    12|  14|
|       Informatics|      May 18, 1987|     Rosie|   Norman|     9|  15|
|          Business|   August 10, 1984|    Martin|   Steele|     7|  16|
|  Machine Learning|  16 December 1990|     Colin| Martinez|     9|  17|
|    Data Analytics|              null|   Bridget|    Twain|     6|  18|
|          Business|      7 March 1980|   Darlene|    Mills|    19|  19|
|    Data Analytics|      June 2, 1985|   Zachary|     null|    10|  20|
+------------------+------------------+----------+---------+------+----+
```

2. Replace the null value(s) in column dob and column last name by "unknown" and "--" respectively.

In [335…

```
studentDf = studentDf.na.fill("unknown",["dob"]).na.fill("--",["last_name"])
studentDf.show()
```

```
+------------------+------------------+----------+---------+------+----+
|            course|               dob|first_name|last_name|points|s_id|
+------------------+------------------+----------+---------+------+----+
|Humanities and Art|  October 14, 1983|      Alan|      Joe|    10|   1|
|  Computer Science|September 26, 1980|    Martin|  Genberg|    17|   2|
|    Graphic Design|     June 12, 1982|     Athur|   Watson|    16|   3|
|    Graphic Design|     April 5, 1987|  Anabelle|  Sanberg|    12|   4|
|        Psychology|  November 1, 1978|      Kira| Schommer|    11|   5|
```

```
|          Business|  17 February 1981| Christian|   Kiriam|    10|   6|
|  Machine Learning|    1 January 1984|   Barbara|  Ballard|    14|   7|
|     Deep Learning|  January 13, 1978|      John|       --|    10|   8|
|  Machine Learning|  26 December 1989|    Marcus|   Carson|    15|   9|
|           Physics|  30 December 1987|     Marta|   Brooks|    11|  10|
|    Data Analytics|     June 12, 1975|     Holly| Schwartz|    12|  11|
|  Computer Science|      July 2, 1985|     April|    Black|    11|  12|
|  Computer Science|     July 22, 1980|     Irene|  Bradley|    13|  13|
|        Psychology|   7 February 1986|      Mark|    Weber|    12|  14|
|       Informatics|      May 18, 1987|     Rosie|   Norman|     9|  15|
|          Business|   August 10, 1984|    Martin|   Steele|     7|  16|
|  Machine Learning|  16 December 1990|     Colin| Martinez|     9|  17|
|    Data Analytics|           unknown|   Bridget|    Twain|     6|  18|
|          Business|      7 March 1980|   Darlene|    Mills|    19|  19|
|    Data Analytics|      June 2, 1985|   Zachary|       --|    10|  20|
+------------------+------------------+----------+---------+------+----+
```

3. In the dob column, there exist several formats of dates, e.g. October 14, 1983 and 26 December 1989. Let's convert all the dates into DD-MM-YYYY format where DD, MM and YYYY are two digits for day, two digits for months and four digits for year respectively.

In [336…

```
import dateutil
def formatdate(dateStr):
    try:
        date = dateutil.parser.parse(dateStr)
        formatedDateStr = date.strftime("%d-%m-%Y")
        return formatedDateStr
    except:
        return str("unknown")
udfformatdate = udf(formatdate, StringType())
studentDf = studentDf.withColumn("dob", udfformatdate("dob"))
studentDf.show()
```

```
+------------------+----------+----------+---------+------+----+
|            course|       dob|first_name|last_name|points|s_id|
+------------------+----------+----------+---------+------+----+
|Humanities and Art|14-10-1983|      Alan|      Joe|    10|   1|
|  Computer Science|26-09-1980|    Martin|  Genberg|    17|   2|
|    Graphic Design|12-06-1982|     Athur|   Watson|    16|   3|
|    Graphic Design|05-04-1987|  Anabelle|  Sanberg|    12|   4|
|        Psychology|01-11-1978|      Kira| Schommer|    11|   5|
|          Business|17-02-1981| Christian|   Kiriam|    10|   6|
|  Machine Learning|01-01-1984|   Barbara|  Ballard|    14|   7|
|     Deep Learning|13-01-1978|      John|       --|    10|   8|
|  Machine Learning|26-12-1989|    Marcus|   Carson|    15|   9|
|           Physics|30-12-1987|     Marta|   Brooks|    11|  10|
|    Data Analytics|12-06-1975|     Holly| Schwartz|    12|  11|
|  Computer Science|02-07-1985|     April|    Black|    11|  12|
|  Computer Science|22-07-1980|     Irene|  Bradley|    13|  13|
|        Psychology|07-02-1986|      Mark|    Weber|    12|  14|
|       Informatics|18-05-1987|     Rosie|   Norman|     9|  15|
|          Business|10-08-1984|    Martin|   Steele|     7|  16|
|  Machine Learning|16-12-1990|     Colin| Martinez|     9|  17|
|    Data Analytics|   unknown|   Bridget|    Twain|     6|  18|
|          Business|07-03-1980|   Darlene|    Mills|    19|  19|
|    Data Analytics|02-06-1985|   Zachary|       --|    10|  20|
+------------------+----------+----------+---------+------+----+
```

4. Insert a new column age and calculate the current age of all students.

In [337…

```
from datetime import date
def Age(birth):
    if(birth!='unknown'):
        birthDate = dateutil.parser.parse(birth)
        current = date.today()
        return current.year - birthDate.year - ((current.month, current.day) < (birt
    else:
        return 0
```

In [338…

```
udfAge = udf(Age, StringType())
studentDf = studentDf.withColumn("age", udfAge("dob"))
studentDf.show()
```

```
+------------------+----------+----------+---------+------+----+---+
|            course|       dob|first_name|last_name|points|s_id|age|
+------------------+----------+----------+---------+------+----+---+
|Humanities and Art|14-10-1983|      Alan|      Joe|    10|   1| 38|
|  Computer Science|26-09-1980|    Martin|  Genberg|    17|   2| 41|
|    Graphic Design|12-06-1982|     Athur|   Watson|    16|   3| 39|
|    Graphic Design|05-04-1987|  Anabelle|  Sanberg|    12|   4| 35|
|        Psychology|01-11-1978|      Kira| Schommer|    11|   5| 44|
|          Business|17-02-1981| Christian|   Kiriam|    10|   6| 41|
|  Machine Learning|01-01-1984|   Barbara|  Ballard|    14|   7| 38|
|     Deep Learning|13-01-1978|      John|       --|    10|   8| 44|
|  Machine Learning|26-12-1989|    Marcus|   Carson|    15|   9| 32|
|           Physics|30-12-1987|     Marta|   Brooks|    11|  10| 34|
|    Data Analytics|12-06-1975|     Holly| Schwartz|    12|  11| 46|
|  Computer Science|02-07-1985|     April|    Black|    11|  12| 37|
|  Computer Science|22-07-1980|     Irene|  Bradley|    13|  13| 41|
|        Psychology|07-02-1986|      Mark|    Weber|    12|  14| 36|
|       Informatics|18-05-1987|     Rosie|   Norman|     9|  15| 35|
|          Business|10-08-1984|    Martin|   Steele|     7|  16| 37|
|  Machine Learning|16-12-1990|     Colin| Martinez|     9|  17| 31|
|    Data Analytics|   unknown|   Bridget|    Twain|     6|  18|  0|
|          Business|07-03-1980|   Darlene|    Mills|    19|  19| 42|
|    Data Analytics|02-06-1985|   Zachary|       --|    10|  20| 37|
+------------------+----------+----------+---------+------+----+---+
```

5. Let's consider granting some points for good performed students in the class. For each student, if his point is larger than 1 standard deviation of all points, then we update his current point to 20, which is the maximum.

In [339…

```
mean_points = studentDf.agg({'points': 'mean'}).collect()
std_points = studentDf.agg({'points': 'std'}).collect()
```

In [341…

```
def grant(x):
    if x > (mean_points[0][0] + std_points[0][0]):
        return 20
    else:
        return x
```

In [342…

```
udfgrant = udf(grant, IntegerType())
```

```
studentDf = studentDf.withColumn('points', udfgrant('points'))
```

In [343…

```
studentDf.show()
```

```
+------------------+----------+----------+---------+------+----+---+
|            course|       dob|first_name|last_name|points|s_id|age|
+------------------+----------+----------+---------+------+----+---+
|Humanities and Art|14-10-1983|      Alan|      Joe|    10|   1| 38|
|  Computer Science|26-09-1980|    Martin|  Genberg|    20|   2| 41|
|    Graphic Design|12-06-1982|     Athur|   Watson|    20|   3| 39|
|    Graphic Design|05-04-1987|  Anabelle|  Sanberg|    12|   4| 35|
|        Psychology|01-11-1978|      Kira| Schommer|    11|   5| 44|
|          Business|17-02-1981| Christian|   Kiriam|    10|   6| 41|
|  Machine Learning|01-01-1984|   Barbara|  Ballard|    14|   7| 38|
|     Deep Learning|13-01-1978|      John|       --|    10|   8| 44|
|  Machine Learning|26-12-1989|    Marcus|   Carson|    20|   9| 32|
|           Physics|30-12-1987|     Marta|   Brooks|    11|  10| 34|
|    Data Analytics|12-06-1975|     Holly| Schwartz|    12|  11| 46|
|  Computer Science|02-07-1985|     April|    Black|    11|  12| 37|
|  Computer Science|22-07-1980|     Irene|  Bradley|    13|  13| 41|
|        Psychology|07-02-1986|      Mark|    Weber|    12|  14| 36|
|       Informatics|18-05-1987|     Rosie|   Norman|     9|  15| 35|
|          Business|10-08-1984|    Martin|   Steele|     7|  16| 37|
|  Machine Learning|16-12-1990|     Colin| Martinez|     9|  17| 31|
|    Data Analytics|   unknown|   Bridget|    Twain|     6|  18|  0|
|          Business|07-03-1980|   Darlene|    Mills|    20|  19| 42|
|    Data Analytics|02-06-1985|   Zachary|       --|    10|  20| 37|
+------------------+----------+----------+---------+------+----+---+
```

6. Create a histogram on the new points created in the task 5.

In [360…

```
new_points = sorted([data[0] for data in studentDf.select('points').collect()])
new_points
```

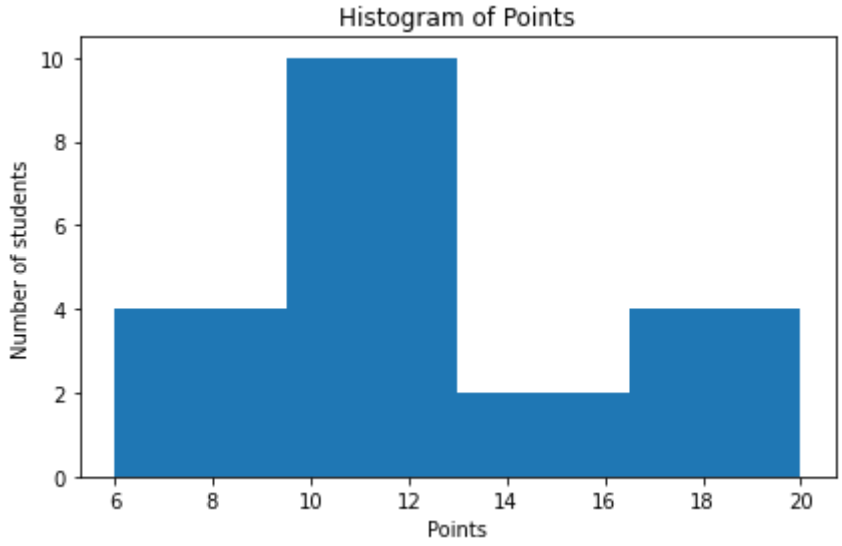
Out[360… `[6, 7, 9, 9, 10, 10, 10, 10, 11, 11, 11, 12, 12, 12, 13, 14, 20, 20, 20, 20]`

In [363…

```
fig, ax = plt.subplots()
num_bins = 4

ax.hist(new_points, num_bins)
ax.set_xlabel('Points')
ax.set_ylabel('Number of students')
ax.set_title(r'Histogram of Points')

fig.tight_layout()
plt.show()
```



Exercise 2: Manipulating Recommender Dataset with Apache Spark

Loading the spark dataframe

In [211…
```
tags = sc.textFile('tags.dat').map(lambda line: line.split("::")).collect()
tag_rdd = sc.parallelize(tags).toDF()
```

In [212…
```
tag_rdd.show()
```

```
+---+-----+--------------------+----------+
| _1|   _2|                  _3|        _4|
+---+-----+--------------------+----------+
| 15| 4973|          excellent!|1215184630|
| 20| 1747|            politics|1188263867|
| 20| 1747|              satire|1188263867|
| 20| 2424|     chick flick 212|1188263835|
| 20| 2424|               hanks|1188263835|
| 20| 2424|                ryan|1188263835|
| 20| 2947|              action|1188263755|
| 20| 2947|                bond|1188263756|
| 20| 3033|               spoof|1188263880|
| 20| 3033|           star wars|1188263880|
| 20| 7438|              bloody|1188263801|
| 20| 7438|            kung fu|1188263801|
| 20| 7438|           Tarantino|1188263801|
| 21|55247|                   R|1205081506|
| 21|55253|               NC-17|1205081488|
| 25|   50|        Kevin Spacey|1166101426|
| 25| 6709|         Johnny Depp|1162147221|
| 31|   65|        buddy comedy|1188263759|
| 31|  546|strangely compelling|1188263674|
| 31| 1091|         catastrophe|1188263741|
+---+-----+--------------------+----------+
only showing top 20 rows
```

In [213…
```
tag_rdd = tag_rdd.withColumnRenamed("_1", "UserID").withColumnRenamed("_2", "MovieID
            withColumnRenamed("_4", "Timestamp")
```

1. A tagging session for a user can be defined as the duration in which he/she generated tagging activities. Typically, an inactive duration of 30 mins is considered as a termination of the tagging session. Your task is to separate out tagging sessions for each user.

In [214…
```
timestamp = [(literal_eval(i.UserID),literal_eval(i.Timestamp)) for i in tag_rdd.col
```

creating a list of tuples contaning userid and timestamp

In [215…
```
timestamp
```

Out[215…
```
[(15, 1215184630),
 (20, 1188263867),
 (20, 1188263867),
 (20, 1188263835),
 (20, 1188263835),
 (20, 1188263835),
 (20, 1188263755),
 (20, 1188263756),
 (20, 1188263880),
 (20, 1188263880),
 (20, 1188263801),
 (20, 1188263801),
 (20, 1188263801),
 (21, 1205081506),
 (21, 1205081488),
 (25, 1166101426),
 (25, 1162147221),
 (31, 1188263759),
 (31, 1188263674),
 (31, 1188263741),
 (31, 1188263707),
 (31, 1188263644),
 (32, 1164735331),
 (39, 1188263791),
 (39, 1188263843),
 (39, 1188263764),
 (39, 1188263782),
 (39, 1188263820),
 (48, 1215135611),
 (48, 1215135517),
 (49, 1188264255),
 (49, 1188264178),
 (49, 1188264095),
 (49, 1188264095),
 (49, 1188264152),
 (49, 1188264256),
 (49, 1188264110),
 (49, 1188264273),
 (49, 1188264273),
 (49, 1188264124),
 (49, 1188264228),
 (49, 1188264226),
 (49, 1188264228),
 (49, 1188264166),
 (49, 1188264167),
 (75, 1162160415),
 (78, 1176691425),
 (109, 1211433235),
```

```
(109, 1165555281),
(109, 1165555288),
(109, 1231122288),
(109, 1165555223),
(109, 1184166988),
(109, 1184166981),
(109, 1165555248),
(109, 1165554873),
(109, 1165555208),
(109, 1165555240),
(109, 1212900981),
(109, 1165555291),
(109, 1211592818),
(109, 1211592812),
(109, 1211592823),
(109, 1165555226),
(109, 1165554764),
(109, 1206627378),
(109, 1165555218),
(109, 1211592803),
(109, 1180487989),
(109, 1180487996),
(109, 1184166348),
(109, 1204754550),
(127, 1188265347),
(127, 1188265347),
(127, 1188265347),
(127, 1188265367),
(127, 1188265367),
(127, 1188265367),
(127, 1188265544),
(127, 1188265544),
(127, 1188265509),
(127, 1188265509),
(127, 1188265463),
(127, 1188265463),
(127, 1188265463),
(127, 1188265447),
(127, 1188265447),
(127, 1188265437),
(127, 1188265437),
(127, 1188265437),
(127, 1188265408),
(127, 1188265408),
(127, 1188265408),
(127, 1188265408),
(127, 1188265408),
(127, 1188265497),
(127, 1188265497),
(127, 1188265497),
(133, 1188265396),
(133, 1188265375),
(133, 1188265375),
(133, 1188265376),
(133, 1188265376),
(146, 1226742764),
(146, 1196517851),
(146, 1213424486),
(146, 1213424434),
(146, 1206782429),
(146, 1210845801),
(146, 1208571447),
(146, 1195728632),
(146, 1198823684),
```

```
(146, 1206782710),
(146, 1195557210),
(146, 1195157483),
(146, 1197724752),
(146, 1204570073),
(146, 1195617357),
(146, 1220879833),
(146, 1221898347),
(146, 1207133904),
(146, 1207128362),
(146, 1204345046),
(146, 1162262859),
(146, 1207043962),
(146, 1196554216),
(146, 1197788078),
(146, 1205300199),
(146, 1207133154),
(146, 1221868010),
(146, 1162264616),
(146, 1195969263),
(146, 1207131682),
(146, 1204569643),
(146, 1195556428),
(146, 1196515683),
(146, 1197720747),
(146, 1210468581),
(146, 1204203536),
(146, 1223701240),
(146, 1223701240),
(146, 1162262502),
(146, 1198456701),
(146, 1196248639),
(146, 1198827455),
(146, 1193266912),
(146, 1195785540),
(146, 1160652719),
(146, 1148728244),
(146, 1208571210),
(146, 1195784881),
(146, 1207133478),
(146, 1206777458),
(146, 1209542902),
(146, 1207189395),
(146, 1207188348),
(146, 1196070162),
(146, 1198456621),
(146, 1198459098),
(146, 1192620381),
(146, 1195554490),
(146, 1207665021),
(146, 1210388699),
(146, 1221984600),
(146, 1162264327),
(146, 1207200137),
(146, 1191750877),
(146, 1195781540),
(146, 1197788347),
(146, 1208760127),
(146, 1212033500),
(146, 1206655868),
(146, 1208761215),
(146, 1162264342),
(146, 1207198999),
(146, 1204430702),
```

```
(146, 1209895664),
(146, 1195618576),
(146, 1222146696),
(146, 1222146714),
(146, 1162262375),
(146, 1208571433),
(146, 1188777367),
(146, 1195728535),
(146, 1208203548),
(146, 1210236373),
(146, 1160652800),
(146, 1207200408),
(146, 1206782587),
(146, 1166969387),
(146, 1166969376),
(146, 1196669479),
(146, 1195731354),
(146, 1197716123),
(146, 1195614195),
(146, 1186715185),
(146, 1211517010),
(146, 1215506567),
(146, 1162261447),
(146, 1188776220),
(146, 1183003380),
(146, 1219722081),
(146, 1207191141),
(146, 1207201664),
(146, 1207131652),
(146, 1193960005),
(146, 1214259710),
(146, 1197804319),
(146, 1195995153),
(146, 1196064245),
(146, 1195617506),
(146, 1223095475),
(146, 1183003519),
(146, 1207670933),
(146, 1207670933),
(146, 1205300149),
(146, 1195613201),
(146, 1207215214),
(146, 1207212226),
(146, 1229369952),
(146, 1196066809),
(146, 1195617545),
(146, 1195782381),
(146, 1196562075),
(146, 1196562064),
(146, 1206783054),
(146, 1207665604),
(146, 1207665604),
(146, 1204202551),
(146, 1204201140),
(146, 1204200458),
(146, 1226742853),
(146, 1226742836),
(146, 1162262050),
(146, 1204201962),
(146, 1206248553),
(146, 1204200822),
(146, 1204202119),
(146, 1220880014),
(146, 1195559992),
```

```
(146, 1162263470),
(146, 1197708700),
(146, 1197708700),
(146, 1204202971),
(146, 1204117008),
(146, 1207665128),
(146, 1207665128),
(146, 1196557973),
(146, 1196557973),
(146, 1186715016),
(146, 1207193821),
(146, 1207044102),
(146, 1207044077),
(146, 1196514488),
(146, 1206251674),
(146, 1219151238),
(146, 1162262254),
(146, 1195736272),
(146, 1197715617),
(146, 1195727751),
(146, 1195969308),
(146, 1195554190),
(146, 1196563428),
(146, 1206782364),
(146, 1162261790),
(146, 1195730091),
(146, 1207666013),
(146, 1207666013),
(146, 1207130668),
(146, 1197724371),
(146, 1207380201),
(146, 1220270316),
(146, 1207666620),
(146, 1204116964),
(146, 1195733699),
(146, 1196065561),
(146, 1197718254),
(146, 1197718237),
(146, 1203547417),
(146, 1207128324),
(146, 1196518353),
(146, 1196248392),
(146, 1205300243),
(146, 1205299268),
(146, 1207201150),
(146, 1217770750),
(146, 1204202332),
(146, 1207193679),
(146, 1196555099),
(146, 1205303463),
(146, 1206777619),
(146, 1160653483),
(146, 1162261432),
(146, 1195786248),
(146, 1195560393),
(146, 1160654269),
(146, 1162262618),
(146, 1198456753),
(146, 1205293518),
(146, 1204203000),
(146, 1148728043),
(146, 1191889433),
(146, 1195783519),
(146, 1206782495),
```

```
(146, 1207199040),
(146, 1212238107),
(146, 1212238107),
(146, 1229309288),
(146, 1204202423),
(146, 1206782750),
(146, 1229309384),
(146, 1204202302),
(146, 1204201935),
(146, 1207211592),
(146, 1197785757),
(146, 1207566692),
(146, 1218252762),
(146, 1218252762),
(146, 1197891212),
(146, 1198458545),
(146, 1196249499),
(146, 1196670258),
(146, 1210473492),
(146, 1208404755),
(146, 1207190398),
(146, 1195783902),
(146, 1195738715),
(146, 1212189099),
(146, 1195780445),
(146, 1207214254),
(146, 1198459716),
(146, 1196824410),
(146, 1195616984),
(146, 1207396314),
(146, 1193547326),
(146, 1192620348),
(146, 1207190867),
(146, 1207566555),
(146, 1195969411),
(146, 1198459019),
(146, 1192620319),
(146, 1198458848),
(146, 1198458893),
(146, 1207195030),
(146, 1195779671),
(146, 1207199675),
(146, 1219722133),
(146, 1198470892),
(146, 1195739211),
(146, 1207191376),
(146, 1206782462),
(146, 1207214067),
(146, 1213606960),
(146, 1210613486),
(146, 1210652572),
(146, 1211422483),
(146, 1207211224),
(146, 1195731312),
(146, 1195779901),
(146, 1186715230),
(146, 1205460575),
(146, 1195879012),
(146, 1223268039),
(146, 1207895341),
(146, 1162262221),
(146, 1203927762),
(146, 1207895334),
(146, 1195149987),
```

```
(146, 1197717364),
(146, 1207199835),
(146, 1197784803),
(146, 1208571487),
(146, 1208571596),
(146, 1223701180),
(146, 1223701180),
(146, 1162264285),
(146, 1196075496),
(146, 1197725059),
(146, 1162264066),
(146, 1211279960),
(146, 1162264062),
(146, 1214193496),
(146, 1195617099),
(146, 1195557225),
(146, 1195731830),
(146, 1195731690),
(146, 1207396777),
(146, 1218620290),
(146, 1218620290),
(146, 1203547466),
(146, 1196517876),
(146, 1220073834),
(146, 1207666259),
(146, 1195617152),
(146, 1195734715),
(146, 1195554581),
(146, 1207212539),
(146, 1190715861),
(146, 1190714591),
(146, 1190714591),
(146, 1206779729),
(146, 1206416856),
(146, 1162264537),
(146, 1205299863),
(146, 1205299854),
(146, 1207902899),
(146, 1192619811),
(146, 1218250457),
(146, 1187605368),
(146, 1206246313),
(146, 1209544621),
(146, 1193546257),
(146, 1197804410),
(146, 1207190828),
(146, 1207216032),
(146, 1196517050),
(146, 1195732786),
(146, 1196516994),
(146, 1207190081),
(146, 1195788584),
(146, 1227823798),
(146, 1226744899),
(146, 1226744905),
(146, 1206924946),
(146, 1206924946),
(146, 1197715885),
(146, 1198461166),
(146, 1222481573),
(146, 1208767625),
(146, 1206246003),
(146, 1203547508),
(146, 1196514607),
```

```
(146, 1205303813),
(146, 1204203518),
(146, 1162264759),
(146, 1195994640),
(146, 1206248841),
(146, 1204609685),
(146, 1206316968),
(146, 1206316968),
(146, 1207200917),
(146, 1207199334),
(146, 1197710245),
(146, 1206217563),
(146, 1204564092),
(146, 1197717038),
(146, 1197788286),
(146, 1197788066),
(146, 1207670991),
(146, 1160651943),
(146, 1196554280),
(146, 1197788069),
(146, 1207189966),
(146, 1162264639),
(146, 1204116426),
(146, 1195612858),
(146, 1189920078),
(146, 1212237079),
(146, 1212237079),
(146, 1204610613),
(146, 1196514172),
(146, 1212236913),
(146, 1162261568),
(146, 1162261942),
(146, 1189919756),
(146, 1162262823),
(146, 1162262730),
(146, 1207986927),
(146, 1207663129),
(146, 1207663129),
(146, 1197604397),
(146, 1207211801),
(146, 1196555257),
(146, 1206246947),
(146, 1207189510),
(146, 1196555178),
(146, 1207199238),
(146, 1209542801),
(146, 1204199200),
(146, 1208760649),
(146, 1195558049),
(146, 1204200985),
(146, 1197715271),
(146, 1197708577),
(146, 1226744980),
(146, 1226744980),
(146, 1226744980),
(146, 1207886721),
(146, 1207130461),
(146, 1207212628),
(146, 1217668500),
(146, 1207395420),
(146, 1208410152),
(146, 1172812063),
(146, 1207195171),
(146, 1157368171),
```

```
(146, 1197787115),
(146, 1207569251),
(146, 1157367995),
(146, 1207132382),
(146, 1162263501),
(146, 1162262874),
(146, 1206925611),
(146, 1196643502),
(146, 1206217612),
(146, 1206217046),
(146, 1207190501),
(146, 1195154384),
(146, 1157368140),
(146, 1207213154),
(146, 1196556330),
(146, 1160654404),
(146, 1196564537),
(146, 1207201266),
(146, 1193961617),
(146, 1223427339),
(146, 1223427339),
(146, 1197787094),
(146, 1207130540),
(146, 1195733372),
(146, 1208235587),
(146, 1198459129),
(146, 1192620391),
(146, 1204610158),
(146, 1204610158),
(146, 1204610158),
(146, 1197604041),
(146, 1193546684),
(146, 1197316888),
(146, 1220750423),
(146, 1195823097),
(146, 1220750423),
(146, 1210316197),
(146, 1196646378),
(146, 1207193574),
(146, 1148728204),
(146, 1205300320),
(146, 1160653346),
(146, 1209679187),
(146, 1195154606),
(146, 1211516372),
(146, 1198457962),
(146, 1207202414),
(146, 1207380825),
(146, 1204203155),
(146, 1195617375),
(146, 1207215901),
(146, 1206072590),
(146, 1199144827),
(146, 1195560102),
(146, 1198470858),
(146, 1197451967),
(146, 1197452122),
(146, 1207128444),
(146, 1221195479),
(146, 1204567317),
(146, 1204567300),
(146, 1211608955),
(146, 1206782224),
(146, 1198458259),
```

```
(146, 1196517109),
(146, 1207379654),
(146, 1195618232),
(146, 1206781078),
(146, 1196562521),
(146, 1204202793),
(146, 1196561908),
(146, 1196558152),
(146, 1196561666),
(146, 1205378970),
(146, 1208659139),
(146, 1163483619),
(146, 1189918560),
(146, 1196515744),
(146, 1148729307),
(146, 1196554260),
(146, 1197788084),
(146, 1218621872),
(146, 1197716665),
(146, 1197808696),
(146, 1204202389),
(146, 1207130277),
(146, 1197807192),
(146, 1197807180),
(146, 1197807180),
(146, 1174643754),
(146, 1208233383),
(146, 1208233441),
(146, 1208760572),
(146, 1225067248),
(146, 1198565440),
(146, 1226826284),
(146, 1206924981),
(146, 1206924981),
(146, 1204116898),
(146, 1204116805),
(146, 1198623964),
(146, 1196516924),
(146, 1206782923),
(146, 1207131109),
(146, 1207201210),
(146, 1205302212),
(146, 1196518535),
(146, 1196518750),
(146, 1195617305),
(146, 1196066831),
(146, 1215597786),
(146, 1215597786),
(146, 1195723529),
(146, 1208570646),
(146, 1195552145),
(146, 1183286548),
(146, 1207369962),
(146, 1226745128),
(146, 1226745128),
(146, 1204202260),
(146, 1196517614),
(146, 1210561491),
(146, 1207133316),
(146, 1162261845),
(146, 1204117964),
(146, 1212152707),
(146, 1207214020),
(146, 1196518037),
```

```
(146, 1204568393),
(146, 1207200877),
(146, 1196556687),
(146, 1208403886),
(146, 1193961644),
(146, 1226742942),
(146, 1218517191),
(146, 1208767675),
(146, 1193960915),
(146, 1206782996),
(146, 1192620611),
(146, 1208204755),
(146, 1206526974),
(146, 1195617338),
(146, 1206779741),
(146, 1220071340),
(146, 1218878201),
(146, 1210059814),
(146, 1195780002),
(146, 1204201297),
(146, 1195788457),
(146, 1218620517),
(146, 1196517451),
(146, 1196517535),
(146, 1197716169),
(146, 1204115272),
(146, 1160653063),
(146, 1204115272),
(146, 1197708523),
(146, 1197708523),
(146, 1201662889),
(146, 1196243730),
(146, 1196795516),
(146, 1169361772),
(146, 1205663169),
(146, 1192621356),
(146, 1196518900),
(146, 1196518900),
(146, 1196518906),
(146, 1196067590),
(146, 1208404064),
(146, 1206782330),
(146, 1217824095),
(146, 1204203180),
(146, 1207665892),
(146, 1196070064),
(146, 1208760987),
(146, 1207199914),
(146, 1197724536),
(146, 1220071371),
(146, 1215822049),
(146, 1215822049),
(146, 1195783152),
(146, 1195783152),
(146, 1210468376),
(146, 1203927737),
(146, 1207130751),
(146, 1197708220),
(146, 1197708285),
(146, 1204955338),
(146, 1208230651),
(146, 1197696927),
(146, 1196070010),
(146, 1195617411),
```

```
(146, 1206780063),
(146, 1207189856),
(146, 1195780611),
(146, 1206925455),
(146, 1206925455),
(146, 1195618357),
(146, 1196554118),
(146, 1197788006),
(146, 1196554205),
(146, 1197788105),
(146, 1196554226),
(146, 1197788093),
(146, 1207215286),
(146, 1208405984),
(146, 1195617224),
(146, 1207476749),
(146, 1206247671),
(146, 1197001276),
(146, 1207896545),
(146, 1204201995),
(146, 1196070106),
(146, 1218250836),
(146, 1204201206),
(146, 1204205041),
(146, 1206317404),
(146, 1197721884),
(146, 1197721884),
(146, 1197721885),
(146, 1198811759),
(146, 1204116510),
(146, 1206346168),
(146, 1206346513),
(146, 1193962020),
(146, 1211521283),
(146, 1162262038),
(146, 1217930379),
(146, 1196558915),
(146, 1210250626),
(146, 1207190194),
(146, 1207200075),
(146, 1207193930),
(146, 1205460268),
(146, 1207191965),
(146, 1197807401),
(146, 1195618071),
(146, 1206782825),
(146, 1207191221),
(146, 1197626772),
(146, 1196070210),
(146, 1208767656),
(146, 1196518126),
(146, 1228452954),
(146, 1228452954),
(146, 1228452954),
(146, 1228452954),
(146, 1206782955),
(146, 1196670432),
(146, 1207212865),
(146, 1197709985),
(146, 1207191477),
(146, 1197806785),
(146, 1197806820),
(146, 1193548116),
(146, 1197806826),
```

```
(146, 1215737765),
(146, 1215737765),
(146, 1206783150),
(146, 1206782202),
(146, 1210967300),
(146, 1197787937),
(146, 1196517388),
(146, 1207213413),
(146, 1195615984),
(146, 1206925489),
(146, 1160654874),
(146, 1198458435),
(146, 1208571423),
(146, 1197724912),
(146, 1196554832),
(146, 1208570965),
(146, 1196554832),
(146, 1193961866),
(146, 1207662951),
(146, 1148728806),
(146, 1157368021),
(146, 1208760079),
(146, 1195733428),
(146, 1208760096),
(146, 1208760096),
(146, 1219269660),
(146, 1219269659),
(146, 1204613530),
(146, 1196562617),
(146, 1196562617),
(146, 1195618705),
(146, 1196562465),
(146, 1196562465),
(146, 1207134096),
(146, 1216589865),
(146, 1216589848),
(146, 1208571333),
(146, 1195149676),
(146, 1162262472),
(146, 1160651996),
(146, 1206239662),
(146, 1205300363),
(146, 1207662749),
(146, 1193961714),
(146, 1198988119),
(146, 1207216475),
(146, 1195618155),
(146, 1195614975),
(146, 1208203627),
(146, 1207665304),
(146, 1208203627),
(146, 1204862529),
(146, 1204862529),
(146, 1196902196),
(146, 1215842233),
(146, 1195733474),
(146, 1192610438),
(146, 1208403546),
(146, 1197720640),
(146, 1207132195),
(146, 1204117097),
(146, 1195618738),
(146, 1195617286),
(146, 1197805470),
```

```
(146, 1209897935),
(146, 1209897935),
(146, 1148728454),
(146, 1208569752),
(146, 1197708430),
(146, 1197708430),
(146, 1204609954),
(146, 1204613114),
(146, 1197720440),
(146, 1209881461),
(146, 1198481658),
(146, 1221427216),
(146, 1162262572),
(146, 1220152545),
(146, 1196516585),
(146, 1204200688),
(146, 1197713953),
(146, 1209785077),
(146, 1197787669),
(146, 1196516001),
(146, 1198919544),
(146, 1210468076),
(146, 1193962092),
(146, 1206316331),
(146, 1198470932),
(146, 1228453019),
(146, 1195150686),
(146, 1206956899),
(146, 1195150721),
(146, 1160654894),
(146, 1204203363),
(146, 1196562328),
(146, 1196562328),
(146, 1195617953),
(146, 1206780857),
(146, 1206780857),
(146, 1206780857),
(146, 1195557402),
(146, 1195732023),
(146, 1183286514),
(146, 1207199623),
(146, 1196563117),
(146, 1196066248),
(146, 1196066248),
(146, 1215493325),
(146, 1215493286),
(146, 1204113619),
(146, 1195733122),
(146, 1225524078),
(146, 1223701291),
(146, 1223701291),
(146, 1207201946),
(146, 1195616867),
(146, 1206924880),
(146, 1207132782),
(146, 1207193501),
(146, 1147948639),
(146, 1206783176),
(146, 1195618821),
(146, 1196843191),
(146, 1195824310),
(146, 1207567365),
(146, 1207215981),
(146, 1205300414),
```

```
(146, 1209679431),
(146, 1209679431),
(146, 1195617022),
(146, 1196065277),
(146, 1217771003),
(146, 1207191708),
(146, 1223787382),
(146, 1196066318),
(146, 1196064918),
(146, 1196065876),
(146, 1148728397),
(146, 1203325527),
(146, 1195556446),
(146, 1207213527),
(146, 1162261469),
(146, 1196516349),
(146, 1207201857),
(146, 1196562198),
(146, 1196562198),
(146, 1207896515),
(146, 1207896678),
(146, 1218251797),
(146, 1197724268),
(146, 1193705107),
(146, 1226287298),
(146, 1197724805),
(146, 1195731964),
(146, 1162261598),
(146, 1196514896),
(146, 1204331069),
(146, 1195727219),
(146, 1156032815),
(146, 1221918773),
(146, 1221918773),
(146, 1195994812),
(146, 1207397143),
(146, 1195618128),
(146, 1208761071),
(146, 1195733535),
(146, 1195727931),
(146, 1195728002),
(146, 1207190680),
(146, 1205377146),
(146, 1196562091),
(146, 1196561972),
(146, 1197787247),
(146, 1209444732),
(146, 1196070289),
(146, 1196562412),
(146, 1196562412),
(146, 1196562412),
(146, 1195823837),
(146, 1148728740),
(146, 1183284698),
(146, 1207199517),
(146, 1219722232),
(146, 1209373931),
(146, 1207193724),
(146, 1199144890),
(146, 1195727144),
(146, 1190715953),
(146, 1207212303),
(146, 1207212303),
(146, 1205305553),
```

```
(146, 1196669397),
(146, 1162261455),
(146, 1195728074),
(146, 1207129860),
(146, 1205378508),
(146, 1163483532),
(146, 1197721431),
(146, 1207211843),
(146, 1207190971),
(146, 1204568279),
(146, 1228172043),
(146, 1148728550),
(146, 1209373078),
(146, 1195615216),
(146, 1228355315),
(146, 1205300486),
(146, 1207216212),
(146, 1207896706),
(146, 1207214632),
(146, 1197700640),
(146, 1197719122),
(146, 1195615044),
(146, 1195154041),
(146, 1162261487),
(146, 1196516318),
(146, 1209897942),
(146, 1209897942),
(146, 1195618555),
(146, 1207664780),
(146, 1207664780),
(146, 1162262650),
(146, 1162262654),
(146, 1196558653),
(146, 1207200947),
(146, 1207370119),
(146, 1197807612),
(146, 1204568205),
(146, 1207211663),
(146, 1198459510),
(146, 1206781227),
(146, 1195618267),
(146, 1197708771),
(146, 1197708771),
(146, 1197802511),
(146, 1208230542),
(146, 1203659233),
(146, 1198459141),
(146, 1192620413),
(146, 1225067287),
(146, 1162261581),
(146, 1196514978),
(146, 1204331202),
(146, 1195727237),
(146, 1207202121),
(146, 1201149756),
(146, 1207129723),
...]
```

Now in order to calculate difference in two tags, for each user its corresponding time stamps are sorted and stored in a dictionary with key as users and values as list of timestamp sorted in ascending order.

In [216…
```
import bisect
user_time = {}
for user, time in timestamp:
    if str(user) not in user_time.keys():
        user_time[str(user)] = [time]
    else:
        bisect.insort(user_time[str(user)], time)
```

To calculate total number of tagging sessions for each user simply difference between two consecutive timestamps is calculated (as the timestamps are sorted) and when the difference exceeds 1800 which is 30 minutes new session gets started.

In [225…
```
tag_session = {}
for user, l_time in user_time.items():
    session = 1
    tags = 1
    if len(l_time) > 1:
        for i in range(1, len(l_time)):
            if (l_time[i] - l_time[i-1]) < 1800:
                tags += 1

            elif (l_time[i] - l_time[i-1]) > 1800:
                session += 1

        if tags == len(l_time):          # if all the tags were done with a differen
            tag_session[user] = 1
        else:
            tag_session[user] = session

    else:                               # for users with just one tag
        tag_session[user] = 1
```

In [226…
```
userid = list(tag_session.keys())
totalSessions = list(tag_session.values())
columns = ['userid', 'totalsessions']
data = [(user, session) for user, session in zip(userid, totalSessions)]
rdd_df = sc.parallelize(data)
userSession = rdd_df.toDF(columns)
userSession.show()
```

```
+------+-------------+
|userid|totalsessions|
+------+-------------+
|    15|            1|
|    20|            1|
|    21|            1|
|    25|            2|
|    31|            1|
|    32|            1|
|    39|            1|
|    48|            1|
|    49|            1|
|    75|            1|
|    78|            1|
|   109|            9|
|   127|            1|
|   133|            1|
|   146|          333|
```

```
|   147|           1|
|   170|           1|
|   175|           2|
|   181|           1|
|   190|           4|
+------+------------+
only showing top 20 rows
```

2. Once you have all the tagging sessions for each user, calculate the frequency of tagging for each user session.

For each user we have to calculate the number of taggings he/she has done in one session. this is just a extension from the last part.

In [234…

```
tag_freq = {}
for user, l_time in user_time.items():
    tags = 1
    session = 1
    tag_freq[user] = []
    if len(l_time) > 1:                        # if atleat two tags are there
        for i in range(1, len(l_time)):

            if (l_time[i] - l_time[i-1]) < 1800:    # add tag for one ongoing sessio
                tags += 1

            elif (l_time[i] - l_time[i-1]) > 1800:       # returns session along wit
                tag_freq[user].append((session, tags))
                session += 1
                tags = 1

        if tags == len(l_time):                          # if no new session was star
            tag_freq[user].append((1, tags))

        else:
            if (l_time[i] - l_time[i-1]) < 1800:            # conditions on the last
                tag_freq[user].append((session, tags))
            elif (l_time[i] - l_time[i-1]) > 1800:
                tag_freq[user].append((session, tags))

    else:
        tag_freq[user].append((1, 1))
```

In [237…

```
new_data = []
new_columns = ['userid', 'session', 'tags']
for user, l_freq in tag_freq.items():
    for session, tags in l_freq:
        new_data.append((user, session, tags))
rdd_newdf = sc.parallelize(new_data)
usertags = rdd_newdf.toDF(new_columns)
usertags.show()
```

```
+------+-------+----+
|userid|session|tags|
+------+-------+----+
|    15|      1|   1|
|    20|      1|  12|
|    21|      1|   2|
|    25|      1|   1|
```

```
|    25|      2|   1|
|    31|      1|   5|
|    32|      1|   1|
|    39|      1|   5|
|    48|      1|   2|
|    49|      1|  15|
|    75|      1|   1|
|    78|      1|   1|
|   109|      1|  11|
|   109|      2|   2|
|   109|      3|   3|
|   109|      4|   1|
|   109|      5|   1|
|   109|      6|   1|
|   109|      7|   4|
|   109|      8|   1|
+------+-------+----+
only showing top 20 rows
```

3. Find a mean and standard deviation of the tagging frequency of each user.

In [239…

```
mean_freq = usertags.groupby('userid').mean('tags')
mean_df = userSession.join(mean_freq, ['userid'])
mean_df.show()
```

```
+------+-------------+---------+
|userid|totalsessions|avg(tags)|
+------+-------------+---------+
| 11563|            1|      1.0|
|  1436|            1|     16.0|
| 17427|            2|      1.5|
|  2136|            1|      1.0|
| 23318|            1|      2.0|
| 28473|            4|      1.0|
|  2904|            1|      1.0|
| 29549|            1|      2.0|
| 32812|            2|      3.0|
| 38271|            1|      1.0|
| 39917|            1|      2.0|
| 42688|            1|      1.0|
| 44446|            1|      1.0|
| 48370|            4|      1.0|
|  5325|            1|      6.0|
| 57051|            1|     13.0|
| 57112|            1|      1.0|
| 57464|            1|      1.0|
| 58744|            1|      1.0|
|  5925|            2|      1.0|
+------+-------------+---------+
only showing top 20 rows
```

In [244…

```
new_df = usertags.join(mean_freq, ['userid'])
new_df.show()
```

```
+------+-------+----+---------+
|userid|session|tags|avg(tags)|
+------+-------+----+---------+
| 11563|      1|   1|      1.0|
|  1436|      1|  16|     16.0|
```

```
| 17427|      1|   2|      1.5|
| 17427|      2|   1|      1.5|
|  2136|      1|   1|      1.0|
| 23318|      1|   2|      2.0|
| 28473|      1|   1|      1.0|
| 28473|      2|   1|      1.0|
| 28473|      3|   1|      1.0|
| 28473|      4|   1|      1.0|
|  2904|      1|   1|      1.0|
| 29549|      1|   2|      2.0|
| 32812|      1|   4|      3.0|
| 32812|      2|   2|      3.0|
| 38271|      1|   1|      1.0|
| 39917|      1|   2|      2.0|
| 42688|      1|   1|      1.0|
| 44446|      1|   1|      1.0|
| 48370|      1|   1|      1.0|
| 48370|      2|   1|      1.0|
+------+-------+----+---------+
only showing top 20 rows
```

In [300…

```
data_collect = new_df.collect()
users = []

# A list of all users
users = [users.append(data[0]) for data in new_df.select('userid').collect() if data
```

In [298…

```
user_std = {}
for user in users:
    s = 0
    c = 0
    for row in data_collect:            # calculating std per user
        if row['userid'] == user:
            c += 1
            s += (row['tags'] - row['avg(tags)'])**2
    user_std[user] = (s/c)**(0.5)
```

In [301…

```
userid = list(user_std.keys())
stdperuser = list(user_std.values())
columns = ['userid', 'standardDeviation']
data2 = [(user, std) for user, std in zip(userid, stdperuser)]
rdd_df2 = sc.parallelize(data2)
userSTD = rdd_df2.toDF(columns)
userSTD.show()
```

```
+------+-----------------+
|userid|standardDeviation|
+------+-----------------+
| 11563|              0.0|
|  1436|              0.0|
| 17427|              0.5|
|  2136|              0.0|
| 23318|              0.0|
| 28473|              0.0|
|  2904|              0.0|
| 29549|              0.0|
| 32812|              1.0|
| 38271|              0.0|
| 39917|              0.0|
| 42688|              0.0|
```

```
| 44446|              0.0|
| 48370|              0.0|
|  5325|              0.0|
| 57051|              0.0|
| 57112|              0.0|
| 57464|              0.0|
| 58744|              0.0|
|  5925|              0.0|
+------+-----------------+
only showing top 20 rows
```

Dataframe contaning user, total sessions, ,mean frequency and standard deviation.

In [323…

```
finaldf = mean_df.join(userSTD, ['userid'])
finaldf.show()
```

```
+------+-------------+---------+-----------------+
|userid|totalsessions|avg(tags)|standardDeviation|
+------+-------------+---------+-----------------+
| 11563|            1|      1.0|              0.0|
|  1436|            1|     16.0|              0.0|
| 17427|            2|      1.5|              0.5|
|  2136|            1|      1.0|              0.0|
| 23318|            1|      2.0|              0.0|
| 28473|            4|      1.0|              0.0|
|  2904|            1|      1.0|              0.0|
| 29549|            1|      2.0|              0.0|
| 32812|            2|      3.0|              1.0|
| 38271|            1|      1.0|              0.0|
| 39917|            1|      2.0|              0.0|
| 42688|            1|      1.0|              0.0|
| 44446|            1|      1.0|              0.0|
| 48370|            4|      1.0|              0.0|
|  5325|            1|      6.0|              0.0|
| 57051|            1|     13.0|              0.0|
| 57112|            1|      1.0|              0.0|
| 57464|            1|      1.0|              0.0|
| 58744|            1|      1.0|              0.0|
|  5925|            2|      1.0|              0.0|
+------+-------------+---------+-----------------+
only showing top 20 rows
```

4. Find a mean and standard deviation of the tagging frequency for across users.

In [313…

```
total_mean = usertags.select(mean("tags")).collect()[0][0]
total_mean
```

Out[313… 7.300084014358817

In [314…

```
total_std = usertags.select(stddev("tags")).collect()[0][0]
total_std
```

Out[314… 22.26429305026497

5. Provide the list of users with a mean tagging frequency within

the two standard deviation from the mean frequency of all users.

This part is bit confusing either standard deviation for entire users is taken or for particular user.

I have done both way. First approach makes more sense though.

In [325…
```
final_data = finaldf.collect()
users2Std = []
for row in final_data:
    if (row['avg(tags)'] > (total_mean - 2*row['standardDeviation'])) and \
        (row['avg(tags)'] < (total_mean + 2*row['standardDeviation'])):
        users2Std.append(row['userid'])
```

In [326…
```
users2Std
```

Out[326…
```
['944',
 '38875',
 '53859',
 '6758',
 '14361',
 '18161',
 '19933',
 '33659',
 '4078',
 '59499',
 '19866',
 '29850',
 '56091',
 '12296',
 '57475',
 '12986',
 '36306',
 '17779',
 '44847',
 '52707',
 '63347',
 '20987',
 '29592',
 '48128',
 '41740',
 '48454',
 '6385',
 '23647',
 '6119',
 '788',
 '13938',
 '37454',
 '40456',
 '40800',
 '43056',
 '10058',
 '59771',
 '17288',
 '42122',
 '17755',
 '25191',
 '60564',
 '27951',
 '60713',
```

```
'67171',
'9595',
'14895',
'3058',
'4341',
'49265',
'26785',
'5637',
'146',
'1510',
'40895',
'59529',
'63507',
'63594',
'69719',
'8683',
'14653',
'29538',
'55865',
'62012',
'69341',
'71149',
'18600',
'36079',
'62847',
'36619',
'21947',
'24263',
'57671',
'10476',
'24517',
'68118',
'18367',
'53192',
'636',
'67795',
'3630',
'38848',
'63198',
'69374',
'22957',
'61274',
'22198',
'70782',
'30849',
'45190',
'57018',
'60316',
'68076',
'30879',
'51027',
'14222',
'22712',
'40930',
'57380',
'6785',
'20897',
'48717',
'62815',
'16155',
'20226',
'23011',
'33891',
'43948',
```

```
'56030',
'63375',
'24221',
'61836',
'32861',
'46085',
'68986',
'7315',
'40174',
'4517',
'66149',
'7815',
'29942',
'34687',
'51603',
'11898',
'13454',
'34680',
'24376',
'51118',
'63458',
'19391',
'46441',
'70743',
'17044',
'24132',
'24592',
'49065',
'55414',
'8041',
'33477',
'5739',
'22609',
'33251',
'65607',
'19018',
'26624',
'68523',
'11368',
'16094',
'34857',
'46586',
'19909',
'58160',
'64363',
'19131',
'25679',
'43603',
'45290',
'64294',
'65293',
'14918',
'66936',
'19885',
'45129',
'67199',
'15087',
'22713',
'39065',
'67066',
'21343',
'9475',
'42059',
'44117',
```

```
'5956',
'10003',
'21522',
'54217',
'35622',
'24929',
'52930',
'68988',
'71509',
'48905',
'57912',
'67687',
'70698',
'4944',
'59092',
'60863',
'6393',
'59401',
'61869',
'15851',
'3301',
'34197',
'3962',
'43113',
'47249',
'4827',
'70760',
'22025',
'12265',
'40737',
'49696',
'52806',
'17541',
'25038',
'25730',
'33943',
'63725',
'59182',
'62880',
'19493',
'4697',
'47384',
'5161',
'36559',
'69002',
'60307',
'13914',
'43792',
'13716',
'15783',
'68263',
'40421',
'32538',
'39033',
'65436',
'65914',
'4022',
'46959',
'1107',
'13403',
'60374',
'68441',
'71497',
'18015',
```

```
'22095',
'47448',
'51649',
'23388',
'25450',
'27808',
'29137',
'5552',
'12273',
'12451',
'23850',
'37419',
'52906',
'22872',
'69388',
'14679',
'190',
'30310',
'33384',
'34405',
'56896',
'70753',
'3304',
'36931',
'11613',
'17312',
'23289',
'40294',
'55840',
'57973',
'59439',
'23858',
'36151',
'1504',
'40916',
'7704',
'69517',
'14599',
'50606',
'24476',
'2643',
'64408',
'13086',
'17909',
'27306',
'34745',
'62055',
'62989',
'64821',
'66129',
'48600',
'63213',
'68516',
'15937',
'18712',
'21767',
'22815',
'67500',
'14423',
'48621',
'41154',
'52510',
'67691',
'30167',
```

```
'37786',
'46804',
'622',
'71331',
'778',
'61327',
'21188',
'31175',
'59468',
'66784',
'23386',
'32828',
'44485',
'59816',
'33866',
'50970',
'57972',
'70974',
'26057',
'3392',
'47438',
'61519',
'9426',
'26116',
'17653',
'38927',
'57962',
'23172',
'24231',
'32736',
'21672',
'10084',
'17647',
'24158',
'62014',
'66892',
'18666',
'23110',
'52983',
'6362',
'15264',
'20729',
'50160',
'5917',
'33235',
'45049',
'69867',
'10674',
'39677',
'63491',
'18093',
'66279',
'8930',
'29442',
'37568',
'41640',
'43699',
'63604',
'11271',
'13014',
'14648',
'48395',
'71037',
'33460',
```

```
'44030',
'70299',
'1806',
'20591',
'4728',
'16949',
'10032',
'21374',
'41130',
'8787',
'41838',
'43017',
'50846',
'19762',
'25425',
'37698',
'68810',
'9688',
'3844',
'61137',
'33784',
'21930',
'46491',
'69820',
'13694',
'16766',
'68637',
'1632',
'1751',
'37216',
'46353',
'66726',
'69660',
'36988',
'38692',
'48006',
'66831',
'8624',
'28892',
'38726',
'29462',
'54650',
'68228',
'67230',
'71420',
'10555',
'21052',
'35821',
'48743',
'56472',
'109',
'22936',
'23032',
'65500',
'13890',
'19620',
'19923',
'51372',
'36784',
'41726',
'64009',
'9316',
'39282',
'64633',
```

```
'68304',
'44041',
'9522',
'1164',
'22302',
'57127',
'25907',
'39935',
'57512',
'36063',
'57035',
'51832',
'66457',
'5002',
'68050',
'15804',
'31170',
'2429',
'34671',
'4737',
'52723',
'55330',
'60158',
'9970',
'25460',
'41178',
'12046',
'49882',
'51954',
'58709',
'65985',
'20281',
'51198',
'9117',
'11744',
'23581',
'51033',
'2456',
'26704',
'43213',
'50733',
'7287',
'25286',
'55398',
'69245',
'14160',
'21272',
'21277',
'14598',
'32011',
'16610',
'40344',
'54740',
'62452',
'2854']
```

In [318…

```
mean_data = mean_df.collect()
users_2std = []
for row in mean_data:
    if (row['avg(tags)'] > (total_mean - 2*total_std)) and (row['avg(tags)'] < (tota
        users_2std.append(row['userid'])
```

In [319…

```
users_2std
```

Out[319…

```
['11563',
 '1436',
 '17427',
 '2136',
 '23318',
 '28473',
 '2904',
 '29549',
 '32812',
 '38271',
 '39917',
 '42688',
 '44446',
 '48370',
 '5325',
 '57051',
 '57112',
 '57464',
 '58744',
 '5925',
 '62646',
 '65495',
 '70078',
 '7252',
 '7711',
 '9030',
 '10272',
 '10309',
 '12888',
 '17835',
 '20141',
 '23113',
 '25821',
 '28039',
 '35438',
 '36538',
 '38900',
 '40512',
 '52364',
 '54729',
 '55388',
 '65298',
 '70542',
 '7743',
 '944',
 '1241',
 '17260',
 '18921',
 '29401',
 '30088',
 '36937',
 '38875',
 '42917',
 '50146',
 '5757',
 '62354',
 '69555',
 '71155',
 '71424',
 '7248',
 '8621',
 '25249',
```

```
'26029',
'2930',
'34958',
'35620',
'42937',
'43040',
'45631',
'47933',
'53859',
'54376',
'55314',
'60661',
'6227',
'66970',
'70120',
'8409',
'16133',
'20435',
'23947',
'24049',
'25122',
'27200',
'27923',
'28180',
'29092',
'3057',
'36243',
'36678',
'36719',
'38047',
'39608',
'39904',
'46033',
'46640',
'46672',
'48534',
'58203',
'62565',
'6348',
'6758',
'70608',
'70981',
'8701',
'11510',
'14361',
'16031',
'17930',
'18161',
'19933',
'24423',
'2477',
'2644',
'27363',
'31661',
'33659',
'35721',
'42337',
'44294',
'5085',
'51259',
'52427',
'54017',
'54129',
'6264',
```

```
'63086',
'66745',
'69551',
'70151',
'10523',
'17946',
'19804',
'2200',
'24708',
'25867',
'28914',
'33880',
'37713',
'38740',
'39689',
'39812',
'4078',
'442',
'44706',
'52882',
'5321',
'55468',
'57050',
'59499',
'60317',
'66128',
'68660',
'71278',
'7685',
'8699',
'8824',
'10563',
'15',
'19866',
'20598',
'2953',
'30846',
'32283',
'36685',
'383',
'40813',
'4856',
'56091',
'65881',
'68818',
'71556',
'11245',
'12296',
'1496',
'17623',
'18059',
'18669',
'19916',
'31179',
'33528',
'38728',
'43593',
'46573',
'49361',
'51761',
'54362',
'57475',
'5859',
'64115',
```

```
'68191',
'70362',
'70368',
'12986',
'15899',
'24641',
'3490',
'36306',
'36667',
'38527',
'40260',
'41829',
'46121',
'55315',
'5829',
'60486',
'67282',
'7625',
'14984',
'16047',
'16389',
'17215',
'17779',
'18933',
'20734',
'21440',
'24171',
'26654',
'26881',
'27238',
'29680',
'31068',
'44847',
'47613',
'5059',
'67306',
'67542',
'71288',
'8286',
'8758',
'9139',
'12941',
'1539',
'17900',
'18114',
'19714',
'20699',
'20987',
'21838',
'24201',
'26855',
'29592',
'29701',
'30139',
'30840',
'44464',
'48128',
'5076',
'53379',
'5475',
'58425',
'58601',
'61855',
'68314',
```

```
'13937',
'3489',
'37396',
'37882',
'39979',
'41740',
'43718',
'48454',
'55992',
'56941',
'58915',
'65463',
'68178',
'69068',
'69469',
'8669',
'12429',
'1910',
'34579',
'36637',
'37553',
'38928',
'42254',
'45938',
'46931',
'52464',
'53558',
'59267',
'62394',
'63060',
'6385',
'6757',
'69590',
'13821',
'15168',
'16779',
'2108',
'21101',
'23647',
'38664',
'39890',
'40603',
'47592',
'47719',
'50354',
'51378',
'51738',
'53641',
'6119',
'61352',
'64215',
'6504',
'68698',
'69489',
'71052',
'788',
'13938',
'1728',
'25656',
'35078',
'37454',
'37657',
'40446',
'40456',
```

```
'43056',
'45562',
'50156',
'50847',
'56568',
'61281',
'63805',
'64344',
'6571',
'70946',
'10058',
'16322',
'16862',
'1915',
'19402',
'25915',
'27313',
'35461',
'40194',
'42908',
'43928',
'46517',
'50011',
'52418',
'54034',
'59771',
'626',
'12917',
'14681',
'16003',
'17461',
'1972',
'23260',
'26508',
'27748',
'33579',
'38263',
'40375',
'41070',
'42191',
'44258',
'46333',
'52881',
'57102',
'59738',
'60036',
'6375',
'68206',
'10191',
'1056',
'10936',
'15702',
'17464',
'19409',
'19783',
'25412',
'25976',
'37025',
'37321',
'42122',
'42747',
'46807',
'46964',
'51461',
```

```
'60598',
'60692',
'66527',
'69930',
'70249',
'70331',
'71301',
'10791',
'1298',
'16149',
'17755',
'25191',
'25359',
'28642',
'33585',
'3386',
'3478',
'35543',
'44484',
'45050',
'47040',
'57315',
'6038',
'60515',
'60564',
'10639',
'11472',
'11608',
'11810',
'16098',
'17364',
'18473',
'20029',
'23157',
'2778',
'27951',
'28011',
'31098',
'36632',
'37217',
'37754',
'39191',
'39609',
'39643',
'42582',
'44468',
'51901',
'52576',
'55620',
'59801',
'60713',
'67171',
'71483',
'880',
'9525',
'9595',
'11255',
'11338',
'12949',
'133',
'14895',
'25606',
'27713',
'28350',
```

```
'29360',
'30225',
'3058',
'34101',
'34783',
'36292',
'37934',
'4341',
'47780',
'52218',
'54087',
'58001',
'64560',
'8194',
'13595',
'19475',
'23087',
'26785',
'27430',
'2842',
'29343',
'31789',
'39986',
'46031',
'53059',
'53279',
'5637',
'57620',
'60229',
'66227',
'66476',
'67889',
'68112',
'69838',
'70066',
'70102',
'7919',
'1282',
'12866',
'13531',
'14924',
'16168',
'25686',
'29477',
'36060',
'44285',
'50721',
'59285',
'61400',
'62600',
'65851',
'66804',
'13029',
'146',
'1510',
'15663',
'19519',
'1988',
'28082',
'36794',
'40895',
'45815',
'47539',
'49913',
```

```
'51448',
'52060',
'54348',
'5621',
'5936',
'59529',
'63507',
'63594',
'65308',
'67816',
'69719',
'69935',
'7587',
'8683',
'9415',
'14653',
'15250',
'16485',
'18718',
'20385',
'22278',
'28508',
'28962',
'29538',
'35051',
'38390',
'38788',
'41258',
'42017',
'47924',
'49334',
'53905',
'55865',
'57175',
'62012',
'65595',
'69341',
'69795',
'71149',
'7530',
'12088',
'15105',
'18600',
'21186',
'21712',
'29435',
'32097',
'3429',
'36079',
'43906',
'46762',
'47002',
'5210',
'62420',
'62847',
'67937',
'7637',
'10926',
'1117',
'11239',
'11824',
'18084',
'19575',
'20321',
```

```
            '20432',
            '20974',
            '25244',
            '25570',
            '27441',
            '35665',
            '36619',
            '45805',
            '57977',
            '60363',
            '65839',
            '68901',
            '69866',
            '894',
            '17384',
            '2014',
            '21184',
            '21947',
            '22125',
            '24263',
            '25635',
            '25741',
            '28001',
            '29421',
            '32308',
            '36655',
            '39746',
            '48076',
            '51301',
            '56561',
            '57671',
            '66863',
            '67418',
            '67465',
            '68363',
            '692',
            '70107',
            '70485',
            '8584',
            '8585',
            '10476',
            '19054',
            '19533',
            '20536',
            '21372',
            '24517',
            '26712',
            '3541',
            '43670',
            '43883',
            '45586',
            '46291',
            '51123',
            '57637',
            '66453',
            '67422',
            '68118',
            '7885',
            '8970',
            '9134',
            '1379',
            '14479',
            '18367',
            '23063',
```

```
'2532',
'34039',
'36875',
'423',
'45987',
'50653',
'53192',
'61969',
'636',
'64370',
'65420',
'67658',
'67795',
'9057',
'13916',
'18304',
'18775',
'18797',
'23221',
'27620',
'31309',
'32293',
'33657',
'3630',
'38847',
'38848',
'47489',
'47768',
'49032',
'4912',
'53610',
'63198',
'66825',
'67993',
'69374',
'70188',
'14036',
'16664',
'17883',
'18293',
'18742',
'22957',
'28734',
'30147',
'35022',
'3512',
'39156',
'3999',
'40837',
'41788',
'43726',
'46185',
'53117',
'61274',
'61377',
'61544',
'62373',
'65869',
'66541',
'66813',
'69155',
'9470',
'11050',
'11554',
```

```
'11555',
'1565',
'1613',
'17363',
'18068',
'21349',
'21513',
'22198',
'28434',
'3251',
'34881',
'37135',
'37762',
'4198',
'45356',
'45495',
'47911',
'50949',
'51692',
'52243',
'5285',
'59219',
'60325',
'66152',
'69172',
'70782',
'71062',
'11162',
'12227',
'13346',
'17394',
'18673',
'21466',
'26987',
'27464',
'30849',
'32299',
'39272',
'39599',
'45190',
'48776',
'53849',
'53902',
'56124',
'56669',
'57018',
'5831',
'60316',
'66190',
'11486',
'11802',
'1457',
'17854',
'2000',
'26528',
'28048',
'3106',
'38375',
'47066',
'47528',
'49935',
'51168',
'5240',
'52591',
```

```
'53702',
'57787',
'66328',
'67524',
'67538',
'68076',
'70645',
'7612',
'10215',
'16754',
'24026',
'28317',
'30879',
'3911',
'51027',
'52648',
'61915',
'67119',
'67594',
'69232',
'14222',
'16234',
'18506',
'30613',
'32021',
'35335',
'38662',
'41822',
'49329',
'528',
'53065',
'59082',
'59358',
'60002',
'6625',
'69480',
'71525',
'7939',
'9469',
'959',
'13151',
'14149',
'14280',
'15974',
'17613',
'18384',
'22712',
'24573',
'31193',
'33285',
'40810',
'40930',
'43102',
'4607',
'56347',
'57380',
'60206',
'67705',
'6785',
'67865',
'71287',
'8777',
'19820',
'20897',
```

```
'24634',
'32294',
'36033',
'48717',
'53042',
'53428',
'54503',
'55491',
'56016',
'57248',
'62795',
'62815',
'65868',
'66136',
'67901',
'11809',
'16155',
'20226',
'23011',
'23228',
'26738',
'29816',
'33780',
'33891',
'35601',
'35703',
'35785',
'36280',
'38119',
'38340',
'43948',
'45555',
'50533',
'54262',
'55693',
'56030',
'60510',
'63375',
'65081',
'21708',
'24221',
'27935',
'28342',
'28496',
'31523',
'3527',
'36203',
'3622',
'37061',
'3860',
'3976',
'40033',
'42201',
'4549',
'49996',
'55382',
'58441',
'59078',
'61836',
'62637',
'69880',
'1010',
'10200',
'10739',
```

```
'25913',
'26674',
'33385',
'36907',
'3849',
'40863',
'42592',
'44287',
'45656',
'4682',
'50503',
'51373',
'58600',
'59462',
'59571',
'63024',
'8148',
'11057',
'1278',
'22089',
'2693',
'28129',
'38889',
'39137',
'40457',
'46085',
'48167',
'5008',
'60182',
'66134',
'68986',
'69646',
'71107',
'13296',
'15879',
'15992',
'267',
'33305',
'34757',
'37168',
'60166',
'63442',
'68465',
'6891',
'7315',
'850',
'13815',
'15178',
'20606',
'21439',
'23466',
'23796',
'25197',
'284',
'28414',
'32512',
'40174',
'4397',
'4517',
'47717',
'54578',
'60963',
'6150',
'66149',
```

```
'7815',
'14246',
'26563',
'29369',
'29508',
'29942',
'30210',
'31327',
'3307',
'33202',
'34687',
'4291',
'46272',
'4650',
'48652',
'50517',
'51603',
'57249',
'57928',
'63542',
'64286',
'65637',
'70742',
'7181',
'9006',
'11229',
'11258',
'11898',
'12691',
'13454',
'13870',
'16425',
'21835',
'34295',
'34680',
'36548',
'43501',
'55193',
'61020',
'61065',
'63116',
'8282',
...]
```