README.md

Spark basic operations

Following code deals with spark data manipulation. Following code is written python and uses pyspark with python 3.5.2 as python backend.

Creating local clusters

For executing the a local cluster is created. All the code is submitted to the master using spark submit interface.

1 Basic operations on RDD

1.1 List to RDD

Given two RDD the following code will load the data from the list to the spark RDD The data is mapped as key value pair and the value is stored as key and index in the list is stored as value. This has been done using parallelize command.

```
from pyspark.sql import SparkSession

session = SparkSession.builder.appName("RDDBasics").getOrCreate()
sc = session.sparkContext
sc.setLogLevel("ERROR")

"""Create RDD"""
a = ["spark", "rdd", "python",
    "context", "create", "class"]
b = ["operation", "apache", "scala", "lambda",
    "parallel", "partition"]
rdd_a = sc.parallelize((value, key) for key, value in enumerate(a))
rdd_b = sc.parallelize((value, key) for key, value in enumerate(b))
```

1.2 Spark RDD joins

Follwing code will run right and full outer join on the created lists.

```
"""right outer join"""
"""Collect should not be used on real data set."""
result = rdd_a.rightOuterJoin(rdd_b)
print(result.collect())
"""full outer join"""
result = rdd_a.fullOuterJoin(rdd_b)
print(result.collect())
```

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

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

The first output is from right outer join. The right outer join will take all values from the right RDD. In above case all data from rdd_b is provided as output. The output is a tuple with key as key of rdd_b and value is a tuple as (rdd_a_value, rdd_b_value). Places where the key of rdd_b is not present in rdd_a the value is None.

Second output is from full outer join. Similar to right outer join but now the values of both rdd_a and rdd_b keys are included.

1.3 Map reduce

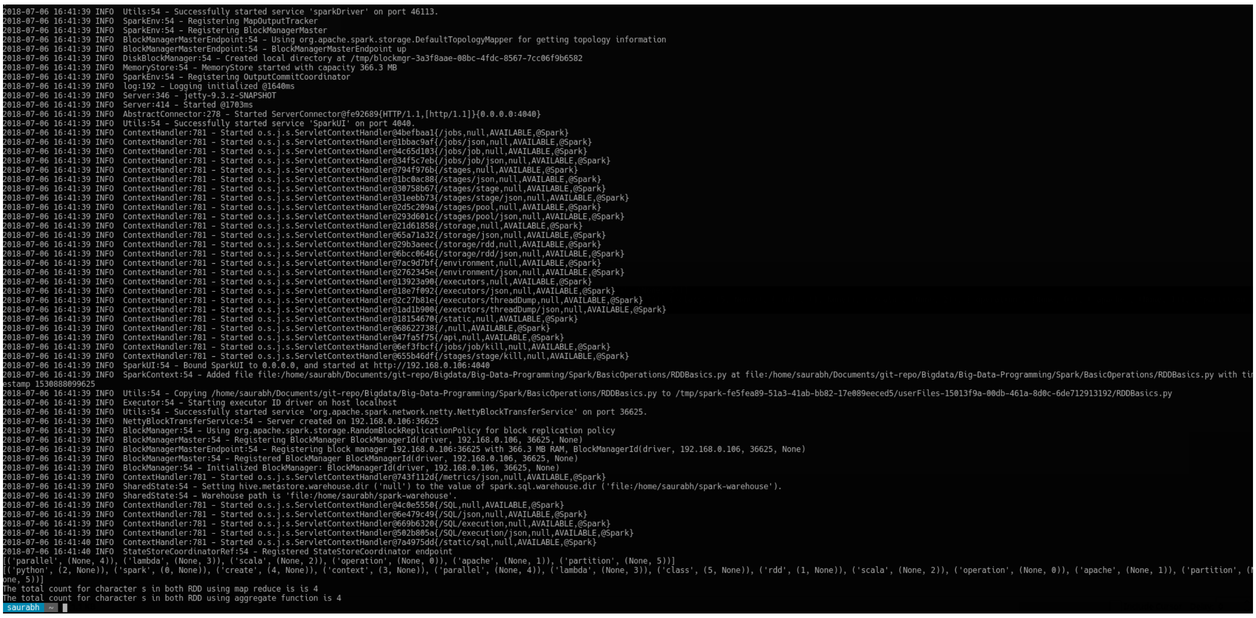
Following code uses map reduce to calculate occurance of character 's'

```
"""map-reduce"""
a_count = rdd_a.map(lambda pair: pair[0].lower().count('s')).reduce(lambda x,
                                                                     y: x+y)
b_count = rdd_b.map(lambda pair: pair[0].lower().count('s')).reduce(lambda x,
                                                                     y: x+y)

print("The total count for character s in both RDD using map reduce is is " +
      str(a_count + b_count))
```

1.4 Aggregate function

Following code does the same instead of map reduce it uses aggregate functions.

```
a_count = rdd_a.aggregate((0, 0), lambda x, y: (1, y[0].lower().count('s') +
                                                x[1]), lambda x, y: (1, y[1] +
                                                                     x[1]))

b_count = rdd_b.aggregate((0, 0), lambda x, y: (1, y[0].lower().count('s') +
                                                x[1]), lambda x, y: (1, y[1] +
                                                                     x[1]))
print("The total count for character s in both RDD using aggregate" +
      " function is " + str(a_count[1] + b_count[1]))
```



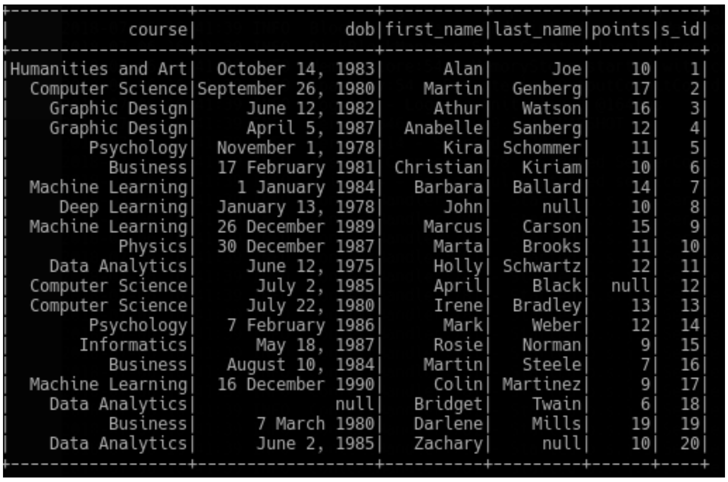
```
2018-07-06 16:41:39 INFO  Utils:54 - Successfully started service 'sparkDriver' on port 46113.
2018-07-06 16:41:39 INFO  SparkEnv:54 - Registering MapOutputTracker
2018-07-06 16:41:39 INFO  SparkEnv:54 - Registering BlockManagerMaster
2018-07-06 16:41:39 INFO  BlockManagerMasterEndpoint:54 - Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
2018-07-06 16:41:39 INFO  BlockManagerMasterEndpoint:54 - BlockManagerMasterEndpoint up
2018-07-06 16:41:39 INFO  DiskBlockManager:54 - Created local directory at /tmp/blockmgr-3a3f8aae-08bc-4fdc-8567-7cc06f9b6582
2018-07-06 16:41:39 INFO  MemoryStore:54 - MemoryStore started with capacity 366.3 MB
2018-07-06 16:41:39 INFO  SparkEnv:54 - Registering OutputCommitCoordinator
2018-07-06 16:41:39 INFO  log:192 - Logging initialized @1640ms
2018-07-06 16:41:39 INFO  Server:346 - jetty-9.3.z-SNAPSHOT
2018-07-06 16:41:39 INFO  Server:414 - Started @1703ms
2018-07-06 16:41:39 INFO  AbstractConnector:278 - Started ServerConnector@fe92689{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2018-07-06 16:41:39 INFO  Utils:54 - Successfully started service 'SparkUI' on port 4040.
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@4befbaa1{/jobs,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@1bbac9af{/jobs/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@4c65d103{/jobs/job,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@34f5c7eb{/jobs/job/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@794f976b{/stages,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@1bc0ac88{/stages/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@30758b67{/stages/stage,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@31eebb73{/stages/stage/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@2d5c209a{/stages/pool,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@293d601c{/stages/pool/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@21d61858{/storage,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@65a71a32{/storage/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@29b3aeec{/storage/rdd,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@6bcc0646{/storage/rdd/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@7ac9d7bf{/environment,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@2762345e{/environment/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@13923a90{/executors,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@18e7f092{/executors/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@2c27b81e{/executors/threadDump,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@1ad1b900{/executors/threadDump/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@18154670{/static,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@68622738{/,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@47fa5f75{/api,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@6ef3fbcf{/jobs/job/kill,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@655b46df{/stages/stage/kill,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  SparkUI:54 - Bound SparkUI to 0.0.0.0, and started at http://192.168.0.106:4040
2018-07-06 16:41:39 INFO  SparkContext:54 - Added file file:/home/saurabh/Documents/git-repo/Bigdata/Big-Data-Programming/Spark/BasicOperations/RDDBasics.py at file:/home/saurabh/Documents/git-repo/Bigdata/Big-Data-Programming/Spark/BasicOperations/RDDBasics.py with ti
estamp 1530888099625
2018-07-06 16:41:39 INFO  Utils:54 - Copying /home/saurabh/Documents/git-repo/Bigdata/Big-Data-Programming/Spark/BasicOperations/RDDBasics.py to /tmp/spark-fe5fea89-51a3-41ab-bb82-17e089eeced5/userFiles-15013f9a-00db-461a-8d0c-6de712913192/RDDBasics.py
2018-07-06 16:41:39 INFO  Executor:54 - Starting executor ID driver on host localhost
2018-07-06 16:41:39 INFO  Utils:54 - Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 36625.
2018-07-06 16:41:39 INFO  NettyBlockTransferService:54 - Server created on 192.168.0.106:36625
2018-07-06 16:41:39 INFO  BlockManager:54 - Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
2018-07-06 16:41:39 INFO  BlockManagerMaster:54 - Registering BlockManager BlockManagerId(driver, 192.168.0.106, 36625, None)
2018-07-06 16:41:39 INFO  BlockManagerMasterEndpoint:54 - Registering block manager 192.168.0.106:36625 with 366.3 MB RAM, BlockManagerId(driver, 192.168.0.106, 36625, None)
2018-07-06 16:41:39 INFO  BlockManagerMaster:54 - Registered BlockManager BlockManagerId(driver, 192.168.0.106, 36625, None)
2018-07-06 16:41:39 INFO  BlockManager:54 - Initialized BlockManager: BlockManagerId(driver, 192.168.0.106, 36625, None)
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@743f112d{/metrics/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  SharedState:54 - Setting hive.metastore.warehouse.dir ('null') to the value of spark.sql.warehouse.dir ('file:/home/saurabh/spark-warehouse').
2018-07-06 16:41:39 INFO  SharedState:54 - Warehouse path is 'file:/home/saurabh/spark-warehouse'.
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@4c0e5550{/SQL,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@6e479c49{/SQL/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@669b6320{/SQL/execution,null,AVAILABLE,@Spark}
2018-07-06 16:41:39 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@502b805a{/SQL/execution/json,null,AVAILABLE,@Spark}
2018-07-06 16:41:40 INFO  ContextHandler:781 - Started o.s.j.s.ServletContextHandler@7a4975dd{/static/sql,null,AVAILABLE,@Spark}
2018-07-06 16:41:40 INFO  StateStoreCoordinatorRef:54 - Registered StateStoreCoordinator endpoint
[('parallel', (None, 4)), ('lambda', (None, 3)), ('scala', (None, 2)), ('operation', (None, 0)), ('apache', (None, 1)), ('partition', (None, 5))]
[('python', (2, None)), ('spark', (0, None)), ('create', (4, None)), ('context', (3, None)), ('parallel', (None, 4)), ('lambda', (None, 3)), ('class', (5, None)), ('rdd', (1, None)), ('scala', (None, 2)), ('operation', (None, 0)), ('apache', (None, 1)), ('partition', (N
one, 5))]
The total count for character s in both RDD using map reduce is is 4
The total count for character s in both RDD using aggregate function is 4
 saurabh  ~  
```

2 Dataframe basic operations

2.1 Load json

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import date_format, coalesce, to_date,\
                                  current_timestamp, datediff, stddev, mean,\
                                  when

import matplotlib.pyplot as plt

session = SparkSession.builder.appName("DataFrameBasics").getOrCreate()
sc = session.sparkContext
sc.setLogLevel("ERROR")

"""Using custom schema so that points can
be loaded as double"""

df = session.read.json("/home/saurabh/python-spark/Spark/" +
                              "Getting_Started/Data/students.json")
df.show()
```

The above code will read json file and load it as spark dataframe. Following is the output of the above code.



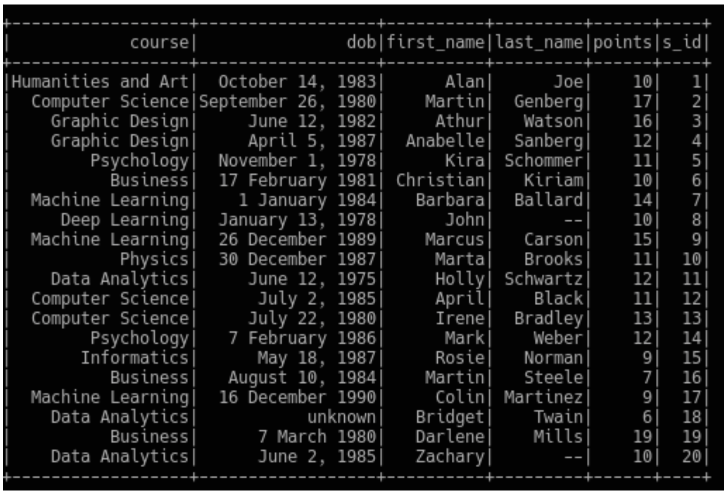
```
+------------------+------------------+----------+---------+------+----+
|            course|               dob|first_name|last_name|points|s_id|
+------------------+------------------+----------+---------+------+----+
|Humanities and Art|  October 14, 1983|      Alan|      Joe|    10|   1|
|  Computer 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|
+------------------+------------------+----------+---------+------+----+
```

The data contains some null values. The next step is to remove null values removal.

2.2 Null value replacement

```
df = df.na.fill(df.groupBy().mean('points').collect()[0][0], "points")
df = df.na.fill("unknown", "dob")
df = df.na.fill("--", "last_name")
df.show()
```

The above code results into following output

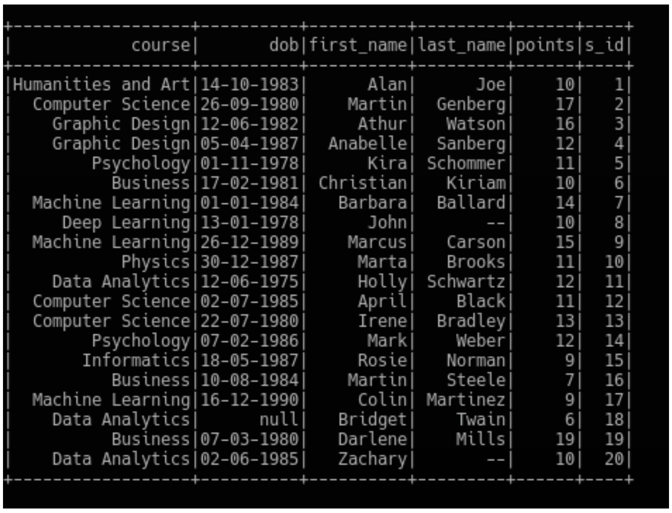


```
+------------------+------------------+----------+---------+------+----+
|            course|               dob|first_name|last_name|points|s_id|
+------------------+------------------+----------+---------+------+----+
|Humanities and Art|  October 14, 1983|      Alan|      Joe|    10|   1|
|  Computer 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|
+------------------+------------------+----------+---------+------+----+
```

2.3 Date manipulation

2.3.1 Date Formatting

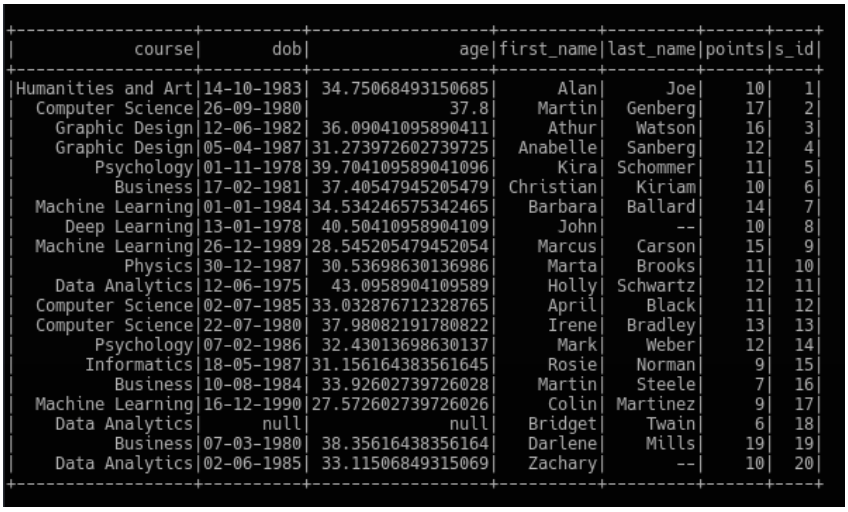
```
"""user defined functions"""


def parseDate(col, formats=("MMM dd, yyyy", "dd MMM yyyy")):
    """coalesce - Retun first non null argument
    * converts the list to parameters"""
    return coalesce(*[to_date(col, f) for f in formats])


def calculateAge(col):
    return datediff(current_timestamp(), to_date(col, "dd-MM-yyyy")) / 365


df = df.select("course", date_format(parseDate("dob"),
                                     "dd-MM-yyyy").alias("dob"),
               "first_name", "last_name", "points", "s_id")
df.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|      null|   Bridget|    Twain|     6|  18|
|          Business|07-03-1980|   Darlene|    Mills|    19|  19|
|    Data Analytics|02-06-1985|   Zachary|       --|    10|  20|
+------------------+----------+----------+---------+------+----+
```

2.3.2 Age calculation

```
df = df.select("course", "dob", calculateAge("dob").alias("age"),
               "first_name", "last_name", "points", "s_id")

df.show()
```

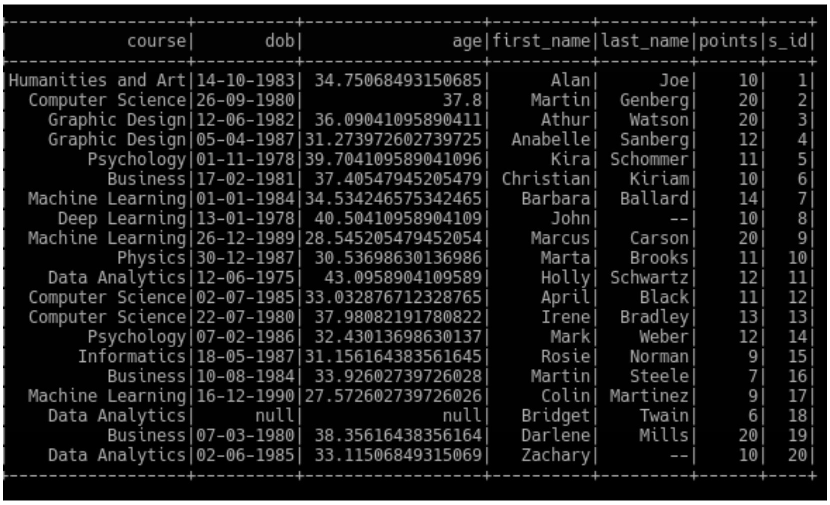


```
+------------------+----------+------------------+----------+---------+------+----+
|            course|       dob|               age|first_name|last_name|points|s_id|
+------------------+----------+------------------+----------+---------+------+----+
|Humanities and Art|14-10-1983| 34.75068493150685|      Alan|      Joe|    10|   1|
|  Computer Science|26-09-1980|              37.8|    Martin|  Genberg|    17|   2|
|    Graphic Design|12-06-1982| 36.09041095890411|     Athur|   Watson|    16|   3|
|    Graphic Design|05-04-1987|31.273972602739725|  Anabelle|  Sanberg|    12|   4|
|        Psychology|01-11-1978|39.704109589041096|      Kira| Schommer|    11|   5|
|          Business|17-02-1981| 37.40547945205479| Christian|   Kiriam|    10|   6|
|  Machine Learning|01-01-1984|34.534246575342465|   Barbara|  Ballard|    14|   7|
|     Deep Learning|13-01-1978| 40.50410958904109|      John|       --|    10|   8|
|  Machine Learning|26-12-1989|28.545205479452054|    Marcus|   Carson|    15|   9|
|           Physics|30-12-1987| 30.53698630136986|     Marta|   Brooks|    11|  10|
|    Data Analytics|12-06-1975|  43.0958904109589|     Holly| Schwartz|    12|  11|
|  Computer Science|02-07-1985|33.032876712328765|     April|    Black|    11|  12|
|  Computer Science|22-07-1980| 37.98082191780822|     Irene|  Bradley|    13|  13|
|        Psychology|07-02-1986| 32.43013698630137|      Mark|    Weber|    12|  14|
|       Informatics|18-05-1987|31.156164383561645|     Rosie|   Norman|     9|  15|
|          Business|10-08-1984| 33.92602739726028|    Martin|   Steele|     7|  16|
|  Machine Learning|16-12-1990|27.572602739726026|     Colin| Martinez|     9|  17|
|    Data Analytics|      null|              null|   Bridget|    Twain|     6|  18|
|          Business|07-03-1980| 38.35616438356164|   Darlene|    Mills|    19|  19|
|    Data Analytics|02-06-1985| 33.11506849315069|   Zachary|       --|    10|  20|
+------------------+----------+------------------+----------+---------+------+----+
```

2.4 Stastical functions

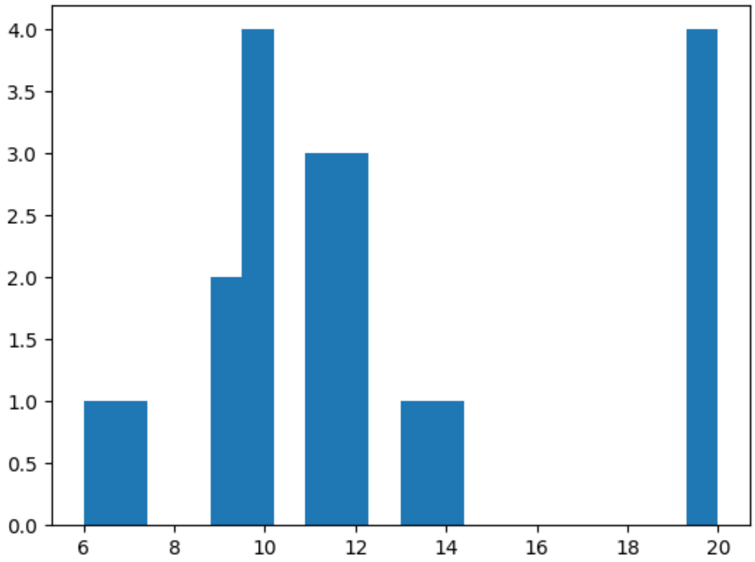
```
df_stats = df.select((stddev("points") + mean("points")).alias('one_std_dev'))
val = df_stats.collect()[0][0]

df = df.withColumn("points", when(df.points > val, 20).otherwise(df.points))
df.show()
```

```
hist = df.select('points').rdd.flatMap(lambda x: x).collect()
print(hist)
plt.hist(hist, bins = 20)
plt.show()
```



```
+------------------+----------+------------------+----------+---------+------+----+
|            course|       dob|               age|first_name|last_name|points|s_id|
+------------------+----------+------------------+----------+---------+------+----+
|Humanities and Art|14-10-1983| 34.75068493150685|      Alan|      Joe|    10|   1|
|  Computer Science|26-09-1980|              37.8|    Martin|  Genberg|    20|   2|
|    Graphic Design|12-06-1982| 36.09041095890411|     Athur|   Watson|    20|   3|
|    Graphic Design|05-04-1987|31.273972602739725|  Anabelle|  Sanberg|    12|   4|
|        Psychology|01-11-1978|39.704109589041096|      Kira| Schommer|    11|   5|
|          Business|17-02-1981| 37.40547945205479| Christian|   Kiriam|    10|   6|
|  Machine Learning|01-01-1984|34.534246575342465|   Barbara|  Ballard|    14|   7|
|     Deep Learning|13-01-1978| 40.50410958904109|      John|       --|    10|   8|
|  Machine Learning|26-12-1989|28.545205479452054|    Marcus|   Carson|    20|   9|
|           Physics|30-12-1987| 30.53698630136986|     Marta|   Brooks|    11|  10|
|    Data Analytics|12-06-1975|  43.0958904109589|     Holly| Schwartz|    12|  11|
|  Computer Science|02-07-1985|33.032876712328765|     April|    Black|    11|  12|
|  Computer Science|22-07-1980| 37.98082191780822|     Irene|  Bradley|    13|  13|
|        Psychology|07-02-1986| 32.43013698630137|      Mark|    Weber|    12|  14|
|       Informatics|18-05-1987|31.156164383561645|     Rosie|   Norman|     9|  15|
|          Business|10-08-1984| 33.92602739726028|    Martin|   Steele|     7|  16|
|  Machine Learning|16-12-1990|27.572602739726026|     Colin| Martinez|     9|  17|
|    Data Analytics|      null|              null|   Bridget|    Twain|     6|  18|
|          Business|07-03-1980| 38.35616438356164|   Darlene|    Mills|    20|  19|
|    Data Analytics|02-06-1985| 33.11506849315069|   Zachary|       --|    10|  20|
+------------------+----------+------------------+----------+---------+------+----+
```

Distribution of score

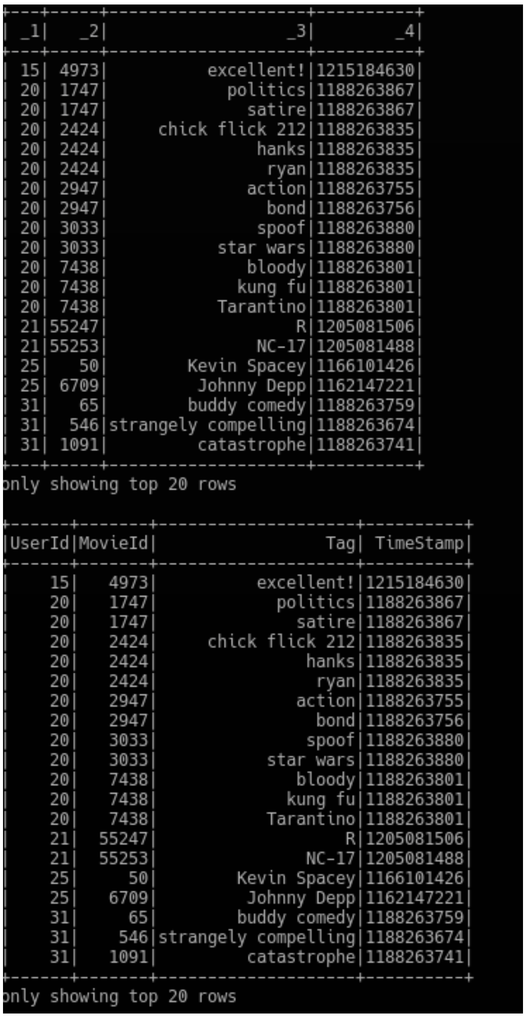


3 Recommender system dataset

For this section Movie lens data set is used.

3.1 Load Data in Dataframe

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import lag, when, isnull, sum, count, lit, avg, stddev
from pyspark.sql.window import Window

session = SparkSession.builder.appName("TaggingSession").getOrCreate()
sc = session.sparkContext
sc.setLogLevel("ERROR")

df = sc.textFile("/home/saurabh/Documents/git-repo/Bigdata/Big-Data-Programming/"+
"Spark/BasicOperations/Data/tags.dat").map(
lambda x: x.split('::')).map(
lambda x: [int(x[0]), int(x[1]), x[2], int(x[3])]).toDF()

df.show()

df = df.select(df._1.alias("UserId"), df._2.alias("MovieId"),
               df._3.alias("Tag"), df._4.alias("TimeStamp"))
df.show()
```

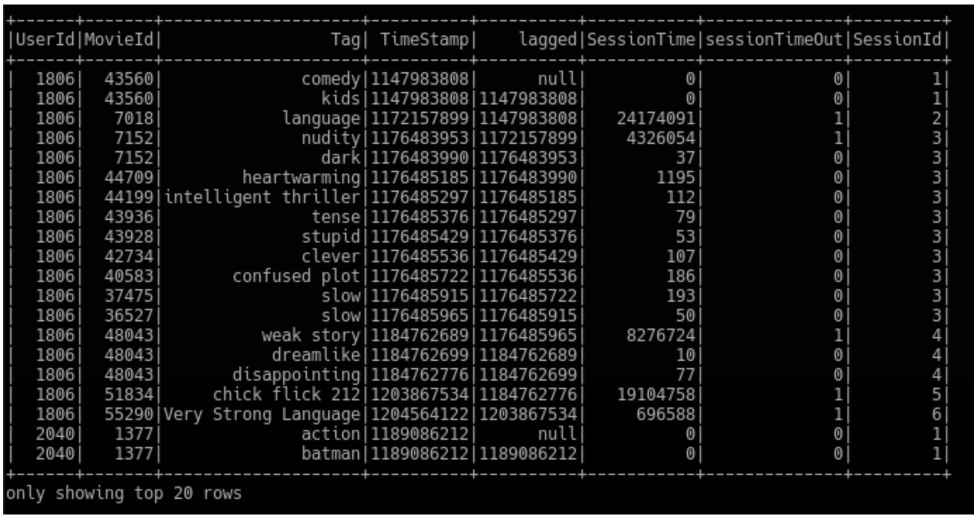


```
+---+-----+--------------------+----------+
| _1|   _2|                  _3|        _4|
+---+-----+--------------------+----------+
| 15| 4973|          excellent!|1215184630|
| 20| 1747|            politics|1188263867|
| 20| 1747|              satire|1188263867|
| 20| 2424|     chick flick 212|1188263835|
| 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

+------+-------+--------------------+----------+
|UserId|MovieId|                 Tag| TimeStamp|
+------+-------+--------------------+----------+
|    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
```

3.2 Tag session

Following code will tag user for session. If user remains inactive for 30 minutes. It is considered as new session.

```
window_partition = Window.partitionBy('UserId').orderBy(['UserId', 'TimeStamp'])

df = df.withColumn("lagged", lag(df.TimeStamp).over(window_partition))
df.show()
df = df.withColumn("SessionTime",
  when(isnull(df.TimeStamp - df.lagged), 0).otherwise(df.TimeStamp - df.lagged))


df = df.withColumn("sessionTimeOut",
      when(df.SessionTime > (30 * 60), 1).otherwise(0))
df.show()

window_partition = Window.partitionBy("UserId").orderBy('TimeStamp')
df = df.withColumn("SessionId", sum(df.sessionTimeOut).over(window_partition))
df.orderBy('MovieId').show()

df = df.withColumn("SessionId", df.SessionId + lit(1))
df.show()
```

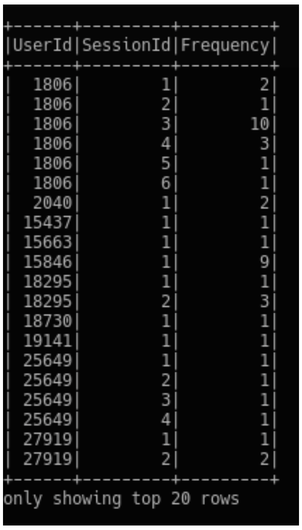


```
+------+-------+--------------------+----------+----------+-----------+--------------+---------+
|UserId|MovieId|                 Tag| TimeStamp|    lagged|SessionTime|sessionTimeOut|SessionId|
+------+-------+--------------------+----------+----------+-----------+--------------+---------+
|  1806|  43560|              comedy|1147983808|      null|          0|             0|        1|
|  1806|  43560|                kids|1147983808|1147983808|          0|             0|        1|
|  1806|   7018|            language|1172157899|1147983808|   24174091|             1|        2|
|  1806|   7152|              nudity|1176483953|1172157899|    4326054|             1|        3|
|  1806|   7152|                dark|1176483990|1176483953|         37|             0|        3|
|  1806|  44709|        heartwarming|1176485185|1176483990|       1195|             0|        3|
|  1806|  44199|intelligent thriller|1176485297|1176485185|        112|             0|        3|
|  1806|  43936|               tense|1176485376|1176485297|         79|             0|        3|
|  1806|  43928|              stupid|1176485429|1176485376|         53|             0|        3|
|  1806|  42734|              clever|1176485536|1176485429|        107|             0|        3|
|  1806|  40583|       confused plot|1176485722|1176485536|        186|             0|        3|
|  1806|  37475|                slow|1176485915|1176485722|        193|             0|        3|
|  1806|  36527|                slow|1176485965|1176485915|         50|             0|        3|
|  1806|  48043|          weak story|1184762689|1176485965|    8276724|             1|        4|
|  1806|  48043|           dreamlike|1184762699|1184762689|         10|             0|        4|
|  1806|  48043|       disappointing|1184762776|1184762699|         77|             0|        4|
|  1806|  51834|     chick flick 212|1203867534|1184762776|   19104758|             1|        5|
|  1806|  55290|Very Strong Language|1204564122|1203867534|     696588|             1|        6|
|  2040|   1377|              action|1189086212|      null|          0|             0|        1|
|  2040|   1377|              batman|1189086212|1189086212|          0|             0|        1|
+------+-------+--------------------+----------+----------+-----------+--------------+---------+
only showing top 20 rows
```

3.3 Session Stats

3.3.1 Calculate Frequency

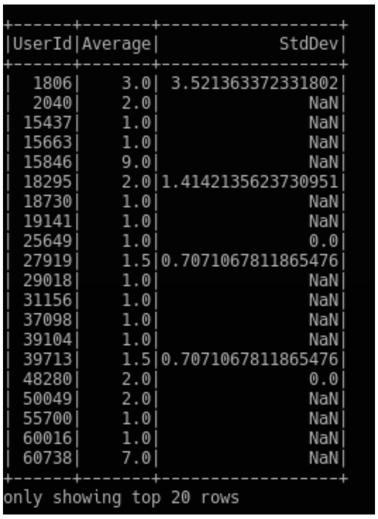
```
tagging_frequency = df.groupBy(['UserId',
                        'SessionId']).agg(count('SessionId').alias('Frequency'))
tagging_frequency.show()
```



```
+------+---------+---------+
|UserId|SessionId|Frequency|
+------+---------+---------+
|  1806|        1|        2|
|  1806|        2|        1|
|  1806|        3|       10|
|  1806|        4|        3|
|  1806|        5|        1|
|  1806|        6|        1|
|  2040|        1|        2|
| 15437|        1|        1|
| 15663|        1|        1|
| 15846|        1|        9|
| 18295|        1|        1|
| 18295|        2|        3|
| 18730|        1|        1|
| 19141|        1|        1|
| 25649|        1|        1|
| 25649|        2|        1|
| 25649|        3|        1|
| 25649|        4|        1|
| 27919|        1|        1|
| 27919|        2|        2|
+------+---------+---------+
only showing top 20 rows
```

3.3.2 Avg and Std user frequency for each user

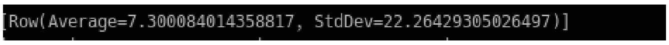
```
stat = tagging_frequency.groupBy(['UserId']
            ).agg(avg('Frequency').alias('Average'),
             stddev('Frequency').alias('StdDev'))
stat.show()
```



```
+------+-------+------------------+
|UserId|Average|            StdDev|
+------+-------+------------------+
|  1806|    3.0| 3.521363372331802|
|  2040|    2.0|               NaN|
| 15437|    1.0|               NaN|
| 15663|    1.0|               NaN|
| 15846|    9.0|               NaN|
| 18295|    2.0|1.4142135623730951|
| 18730|    1.0|               NaN|
| 19141|    1.0|               NaN|
| 25649|    1.0|               0.0|
| 27919|    1.5|0.7071067811865476|
| 29018|    1.0|               NaN|
| 31156|    1.0|               NaN|
| 37098|    1.0|               NaN|
| 39104|    1.0|               NaN|
| 39713|    1.5|0.7071067811865476|
| 48280|    2.0|               0.0|
| 50049|    2.0|               NaN|
| 55700|    1.0|               NaN|
| 60016|    1.0|               NaN|
| 60738|    7.0|               NaN|
+------+-------+------------------+
only showing top 20 rows
```

3.3.3 Avg and Std user frequency across users

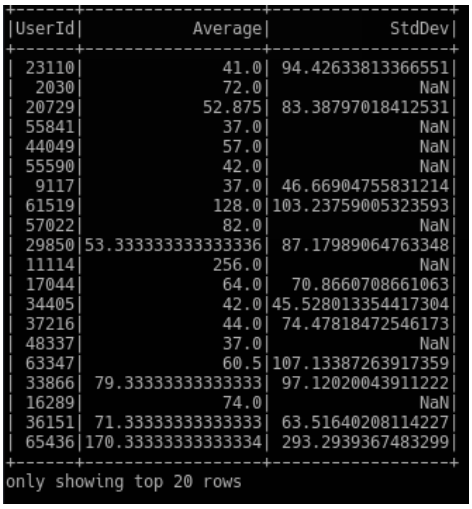
```
stat = tagging_frequency.groupBy().agg(
      avg('Frequency').alias('Average'),
       stddev('Frequency').alias('StdDev')).collect()
print(stat)
```



```
[Row(Average=7.300084014358817, StdDev=22.26429305026497)]
```

3.3.4 Users with mean more that three standard deviation

```
stat_a.filter(stat_a['Average'] > 2 * stat[0][0] + stat[0][1]).show()
```



```
+------+------------------+------------------+
|UserId|           Average|            StdDev|
+------+------------------+------------------+
| 23110|              41.0| 94.42633813366551|
|  2030|              72.0|               NaN|
| 20729|            52.875| 83.38797018412531|
| 55841|              37.0|               NaN|
| 44049|              57.0|               NaN|
| 55590|              42.0|               NaN|
|  9117|              37.0| 46.66904755831214|
| 61519|             128.0|103.23759005323593|
| 57022|              82.0|               NaN|
| 29850|53.333333333333336| 87.17989064763348|
| 11114|             256.0|               NaN|
| 17044|              64.0|  70.8660708661063|
| 34405|              42.0|45.528013354417304|
| 37216|              44.0| 74.47818472546173|
| 48337|              37.0|               NaN|
| 63347|              60.5|107.13387263917359|
| 33866| 79.33333333333333| 97.12020043911222|
| 16289|              74.0|               NaN|
| 36151| 71.33333333333333| 63.51640208114227|
| 65436|170.33333333333334| 293.2939367483299|
+------+------------------+------------------+
only showing top 20 rows
```