**Fahad Fiaz – (303141) – G2**

## System Info:

| | |
|---|---|
| Processor | i7-5500U , 2.40GHz |
| Cores | 4 |
| Operating system | Windows 64 Bit |
| Ram | 8GB |
| Programming Language | Python 3.7.7 |

# Q1:

# Part A):

Common configuration for Spark:

In [1]:
```
import findspark
import matplotlib.pyplot as plt
findspark.init()
findspark.find()
import pyspark
findspark.find()
```

Out[1]:
```
'D:\\spark-2.4.6-bin-hadoop2.7'
```

In [2]:
```
from pyspark import SparkContext, SparkConf
import numpy as np
import pandas as pd
from pyspark.sql.functions import mean , stddev, col, to_date, date_format
import pyspark.sql.functions as F
from time import strptime
import datetime
from ast import literal_eval


from pyspark.sql import SparkSession
conf = pyspark.SparkConf().setAppName('SparkApp').setMaster('local')
sc = pyspark.SparkContext(conf=conf)
spark=SparkSession(sc)
```

1) Right outer joins return all the data from the right RDD and only matched ones of left RDD, whereas for full outer join it joins both the RDDs and returns data from both of them. Since we need pair RDD ((key, value) form) to apply join operation so for each pair I have taken key as word and added dummy value as index of word in list before applying join operation.

```
In [3]: #Part a : 1)
a = ["spark", "rdd", "python", "context", "create", "class"]
b = ["operation", "apache", "scala", "lambda","parallel","partition"]

new_a=map(lambda x: (x[1],x[0]), enumerate(a))
new_b=map(lambda x: (x[1],x[0]), enumerate(b))
rdd_a=sc.parallelize(new_a)
rdd_b=sc.parallelize(new_b)
rightOuterJoin = rdd_a.leftOuterJoin(rdd_b).collect()
fullOuterJoin = rdd_a.fullOuterJoin(rdd_b).collect()
```

```
In [4]: rightOuterJoin

Out[4]: [('python', (2, None)),
         ('class', (5, None)),
         ('spark', (0, None)),
         ('rdd', (1, None)),
         ('context', (3, None)),
         ('create', (4, None))]
```

```
In [5]: fullOuterJoin

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

2) First I concatenated two rdds, then apply flat map which convert each word into characters list and give use us flat list, then filtered character “s”, and map it to count the number of times “s” appear, then used reduce function which return the total count of “s” in both rdds.

```
In [6]: #Part a : 2)
rdd_a=sc.parallelize(a)
rdd_b=sc.parallelize(b)
rdd_a.union(rdd_b).flatMap(lambda x: x).filter(lambda x: x == 's').map(lambda x: (x,1)).reduceByKey(lambda x,y: x+y).collect()

Out[6]: [('s', 4)]
```

3) First I concatenated two rdds and then used aggregation function which iterates over entire concatenated RDD and increase the value of count by 1 if character “s” is found.

```
]: rdd_a.union(rdd_b).flatMap(lambda x: x).filter(lambda x: x == 's').aggregate(0,(lambda i, value: i + 1),(lambda i, j: (i + j)))
]: 4
```

# Partb:

1,) First mean value is calculated using mean() function and then null values is replaced by using fillna() function.

2) The null values in dob column is replaced with “unknown” and null value in last_name is replaced with “--” by using fillna() function.

| course | dob | first_name | last_name | points | s_id |
|---|---|---|---|---|---|
| Humanities and Art | October 14, 1983 | Alan | Joe | 10.0 | 1 |
| Computer Science | September 26, 1980 | Martin | Genberg | 17.0 | 2 |
| Graphic Design | June 12, 1982 | Athur | Watson | 16.0 | 3 |
| Graphic Design | April 5, 1987 | Anabelle | Sanberg | 12.0 | 4 |
| Psychology | November 1, 1978 | Kira | Schommer | 11.0 | 5 |
| Business | 17 February 1981 | Christian | Kiriam | 10.0 | 6 |
| Machine Learning | 1 January 1984 | Barbara | Ballard | 14.0 | 7 |
| Deep Learning | January 13, 1978 | John | -- | 10.0 | 8 |
| Machine Learning | 26 December 1989 | Marcus | Carson | 15.0 | 9 |
| Physics | 30 December 1987 | Marta | Brooks | 11.0 | 10 |
| Data Analytics | June 12, 1975 | Holly | Schwartz | 12.0 | 11 |
| Computer Science | July 2, 1985 | April | Black | 11.736842105263158 | 12 |
| Computer Science | July 22, 1980 | Irene | Bradley | 13.0 | 13 |
| Psychology | 7 February 1986 | Mark | Weber | 12.0 | 14 |
| Informatics | May 18, 1987 | Rosie | Norman | 9.0 | 15 |
| Business | August 10, 1984 | Martin | Steele | 7.0 | 16 |
| Machine Learning | 16 December 1990 | Colin | Martinez | 9.0 | 17 |
| Data Analytics | unknown | Bridget | Twain | 6.0 | 18 |
| Business | 7 March 1980 | Darlene | Mills | 19.0 | 19 |
| Data Analytics | June 2, 1985 | Zachary | -- | 10.0 | 20 |

3) For this task I wrote user defined function which takes input each row of “dob” and calculate date according to task. First it split the date and get list with 3 indexes. Then for each index it checks if it month and convert month to specific format mentioned in task. Do this same for day and year. Finally it appends the find date and replaces it with previous row value.

| course | dob | first_name | last_name | points | s_id |
|---|---|---|---|---|---|
| Humanities and Art | 14-10-1983 | Alan | Joe | 10.0 | 1 |
| Computer Science | 26-09-1980 | Martin | Genberg | 17.0 | 2 |
| Graphic Design | 12-06-1982 | Athur | Watson | 16.0 | 3 |
| Graphic Design | 05-04-1987 | Anabelle | Sanberg | 12.0 | 4 |
| Psychology | 01-11-1978 | Kira | Schommer | 11.0 | 5 |
| Business | 17-02-1981 | Christian | Kiriam | 10.0 | 6 |
| Machine Learning | 01-01-1984 | Barbara | Ballard | 14.0 | 7 |
| Deep Learning | 13-01-1978 | John | -- | 10.0 | 8 |
| Machine Learning | 26-12-1989 | Marcus | Carson | 15.0 | 9 |
| Physics | 30-12-1987 | Marta | Brooks | 11.0 | 10 |
| Data Analytics | 12-06-1975 | Holly | Schwartz | 12.0 | 11 |
| Computer Science | 02-07-1985 | April | Black | 11.736842105263158 | 12 |
| Computer Science | 22-07-1980 | Irene | Bradley | 13.0 | 13 |
| Psychology | 07-02-1986 | Mark | Weber | 12.0 | 14 |
| Informatics | 18-05-1987 | Rosie | Norman | 9.0 | 15 |
| Business | 10-08-1984 | Martin | Steele | 7.0 | 16 |
| Machine Learning | 16-12-1990 | Colin | Martinez | 9.0 | 17 |
| Data Analytics | unknown | Bridget | Twain | 6.0 | 18 |
| Business | 07-03-1980 | Darlene | Mills | 19.0 | 19 |
| Data Analytics | 02-06-1985 | Zachary | -- | 10.0 | 20 |

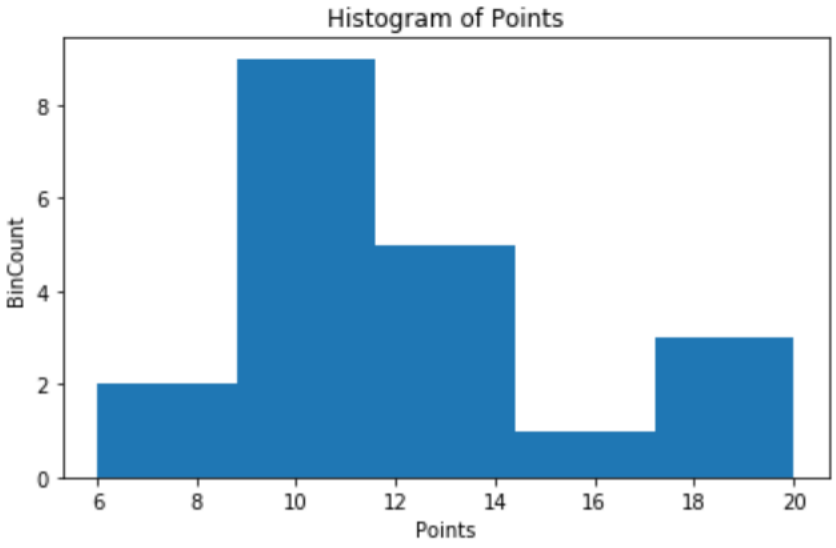
4) Calculate_age calculate age by subtracting current date from dob column value

| course | dob | first_name | last_name | points | s_id | age |
|---|---|---|---|---|---|---|
| Humanities and Art | 14-10-1983 | Alan | Joe | 10.0 | 1 | 36 |
| Computer Science | 26-09-1980 | Martin | Genberg | 17.0 | 2 | 39 |
| Graphic Design | 12-06-1982 | Athur | Watson | 16.0 | 3 | 37 |
| Graphic Design | 05-04-1987 | Anabelle | Sanberg | 12.0 | 4 | 33 |
| Psychology | 01-11-1978 | Kira | Schommer | 11.0 | 5 | 42 |
| Business | 17-02-1981 | Christian | Kiriam | 10.0 | 6 | 39 |
| Machine Learning | 01-01-1984 | Barbara | Ballard | 14.0 | 7 | 36 |
| Deep Learning | 13-01-1978 | John | -- | 10.0 | 8 | 42 |
| Machine Learning | 26-12-1989 | Marcus | Carson | 15.0 | 9 | 30 |
| Physics | 30-12-1987 | Marta | Brooks | 11.0 | 10 | 32 |
| Data Analytics | 12-06-1975 | Holly | Schwartz | 12.0 | 11 | 44 |
| Computer Science | 02-07-1985 | April | Black | 11.736842105263158 | 12 | 35 |
| Computer Science | 22-07-1980 | Irene | Bradley | 13.0 | 13 | 39 |
| Psychology | 07-02-1986 | Mark | Weber | 12.0 | 14 | 34 |
| Informatics | 18-05-1987 | Rosie | Norman | 9.0 | 15 | 33 |
| Business | 10-08-1984 | Martin | Steele | 7.0 | 16 | 35 |
| Machine Learning | 16-12-1990 | Colin | Martinez | 9.0 | 17 | 29 |
| Data Analytics | unknown | Bridget | Twain | 6.0 | 18 | unknown |
| Business | 07-03-1980 | Darlene | Mills | 19.0 | 19 | 40 |
| Data Analytics | 02-06-1985 | Zachary | -- | 10.0 | 20 | 35 |

5) First we find standard deviation of point's col. Then my user define function check if point value is greater than 1 standard deviation of all points, if yes it replaces point value to 20 else point value remains same

| course | dob | first_name | last_name | points | s_id | age |
|---|---|---|---|---|---|---|
| Humanities and Art | 14-10-1983 | Alan | Joe | 10.0 | 1 | 36 |
| Computer Science | 26-09-1980 | Martin | Genberg | 20 | 2 | 39 |
| Graphic Design | 12-06-1982 | Athur | Watson | 20 | 3 | 37 |
| Graphic Design | 05-04-1987 | Anabelle | Sanberg | 12.0 | 4 | 33 |
| Psychology | 01-11-1978 | Kira | Schommer | 11.0 | 5 | 42 |
| Business | 17-02-1981 | Christian | Kiriam | 10.0 | 6 | 39 |
| Machine Learning | 01-01-1984 | Barbara | Ballard | 14.0 | 7 | 36 |
| Deep Learning | 13-01-1978 | John | -- | 10.0 | 8 | 42 |
| Machine Learning | 26-12-1989 | Marcus | Carson | 15.0 | 9 | 30 |
| Physics | 30-12-1987 | Marta | Brooks | 11.0 | 10 | 32 |
| Data Analytics | 12-06-1975 | Holly | Schwartz | 12.0 | 11 | 44 |
| Computer Science | 02-07-1985 | April | Black | 11.736842105263158 | 12 | 35 |
| Computer Science | 22-07-1980 | Irene | Bradley | 13.0 | 13 | 39 |
| Psychology | 07-02-1986 | Mark | Weber | 12.0 | 14 | 34 |
| Informatics | 18-05-1987 | Rosie | Norman | 9.0 | 15 | 33 |
| Business | 10-08-1984 | Martin | Steele | 7.0 | 16 | 35 |
| Machine Learning | 16-12-1990 | Colin | Martinez | 9.0 | 17 | 29 |
| Data Analytics | unknown | Bridget | Twain | 6.0 | 18 | unknown |
| Business | 07-03-1980 | Darlene | Mills | 20 | 19 | 40 |
| Data Analytics | 02-06-1985 | Zachary | -- | 10.0 | 20 | 35 |

6) Histogram:



# Q2:

For this task I have run my code by taking 1st 100 rows of dataset. Since, my code works fine for these rows, then it should work fine on all dataset.

1) For this task I have written 2 functions.

1ST function takes timestamp as input. Timestamp is key value pair as key = user_id and value = timestamp. This first check if user exist in user_duration list or not. If not then it add that user to this list along with time 0 as user is just starting his session. Then if user already exist in user_duration list, it check calculate total time of user session until new use session has started and save it as user_id,user_session_time .

2nd function takes user_id,session_time as input and check if user spent less than 30 minutes or not. If yes, then it considered as a termination of the tagging session. Then total tagging sessions of user is calculated and saved.

I have just attached part of ouput. Full output available in HTML files attached

```
[(15, 1),
 (20, 1),
 (20, 1),
 (20, 1),
 (20, 1),
 (20, 1),
 (20, 1),
 (20, 1),
 (20, 1),
 (20, 1),
 (20, 1),
 (20, 1),
 (20, 1),
 (20, 1),
 (21, 1),
 (21, 1),
 (25, 1),
 (25, 1),
 (31, 1),
 (31, 1),
 (31, 1),
 (31, 1),
 (31, 1),
 (32, 1),
 (39, 1),
 (39, 1),
 (39, 1),
```

2) Then frequency of tagging for each user session is calculated by checking it the user session exist in dictionary or not. If it exist then we increase the count.

```
def calculate_frequency(tag_session):
    dic={}
    for i in range(len(tag_session)):
        if (tag_session[i][0],tag_session[i][1]) in dic:
            dic[(tag_session[i][0],tag_session[i][1])] += 1
        else:
            dic[(tag_session[i][0],tag_session[i][1])] =1
    return dic

frequency_list=calculate_frequency(tags_session)
print(frequency_list)
```

```
{(15, 1): 1, (20, 1): 12, (21, 1): 2, (25, 1): 2, (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): 6, (109, 3): 4, (109, 4): 3, (109, 5): 1, (127, 1): 26}
```

3) Then each_and_across_user  function perform the last 3 mentioned tasks. Output is as follows:

```
Each User Mean:  {15: 1.0, 20: 1.0, 21: 1.0, 25: 1.0, 31: 1.0, 32: 1.0, 39: 1.0, 48: 1.0, 49: 1.0, 75: 1.0, 78: 1.0, 109: 3.0,
127: 1.0}
Each User Sd:  {15: 0.0, 20: 0.0, 21: 0.0, 25: 0.0, 31: 0.0, 32: 0.0, 39: 0.0, 48: 0.0, 49: 0.0, 75: 0.0, 78: 0.0, 109: 1.41421
35623730951, 127: 0.0}
Each Across User Mean:  1.588235294117647
Each Across User Sd:  1.1910856900774465
Each Final Data:  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 4, 5, 1]
```



**All  running outputs in Exercise1.html and Exercise2.html**