Distributed data analytics (2016-2017)
Information Systems and Machine Learning Lab
University of Hildesheim

Exercise Sheet 7

Mario Rodríguez Ruiz

June 1, 2017

Contents

List of Figures

List of Tables

1 Explain your system

	Home System
Machine	Asus Notebook ROG G60Jx
Operating System	Windows 10 Pro 64-bit
CPU	Intel Core i7 720QM @1.60GHz
Number of cores	4
Number of threads	8
RAM	16GB @665MHz (9-9-9-24)
Programming language version Python	v3.6.1
Programming language version Java	v1.8
Hadoop	v2.6.0
Spark	v2.1.1

Table 1.1: My system

2 Basic Resilient Distributed Dataset (RDD)

2.1 Create two RDD objects of a, b and do the following tasks. Words should be remained in the results of join operations.

2.1.1 Perform rightOuterJoin and fullOuterJoin operations between a and b. Briefly explain your solution.

– **rightOuterJoin()** –

Perform a right outer join of self and other.

```
rightOuterJoin(self, other, numPartitions=None)
```

For each element **(k, w)** in *other*, the resulting RDD will either contain all pairs **(k, (v, w))** for **v** in this, or the *pair* **(k, (None, w))** if no elements in *self* have key **k**.

```python
# Imports the PySpark libraries
from pyspark import SparkConf, SparkContext

# The 'os' library allows us to read the environment variable SPARK_HOME defined in the IDE environment
import os

# Configure the Spark context to give a name to the application
sparkConf = SparkConf().setAppName("ex71")
sc = SparkContext(conf = sparkConf)

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

c1 = sc.parallelize(a).map(lambda a: (a, 1))
```

```
c2 = sc.parallelize(b).map(lambda a: (a, 1))

for i in c1.rightOuterJoin(c2).collect(): print (i)
```

The Figure 2.1 shows the execution result of the previous code.

```
Console   Progress
<terminated> ex7-1.py [C:\Users\Mario\AppData\Local\Programs\Python\Python36\python.exe]
[Stage 0:>                                                          (0 + 0) / 16]
[Stage 0:>                                                          (0 + 8) / 16]
[Stage 0:=======>                                                   (2 + 8) / 16]
[Stage 0:==============================>                            (8 + 8) / 16]
[Stage 0:=================================================>        (13 + 3) / 16]
[Stage 1:>                                                          (0 + 8) / 16]
[Stage 1:======================>                                    (6 + 9) / 16]
[Stage 1:==============================>                            (8 + 8) / 16]
                                                                        ('parallel', (None, 1))
('lambda', (None, 1))
('scala', (None, 1))
('operation', (None, 1))
('apache', (None, 1))
('partition', (None, 1))
```

Figure 2.1: Perform a right outer join of self and other

– **fullOuterJoin()** –

Perform a right outer join of self and other.

```
fullOuterJoin(other, numPartitions=None)
```

For each element **(k, v)** in *self*, the resulting RDD will either contain all pairs **(k, (v, w))** for **w** in *other*, or the *pair* **(k, (v, None))** if no elements in *other* have key **k**.

```
# Imports the PySpark libraries
from pyspark import SparkConf, SparkContext

# The 'os' library allows us to read the environment variable SPARK_HOME defined in the IDE environment
import os

# Configure the Spark context to give a name to the application
sparkConf = SparkConf().setAppName("ex7-1-2")
sc = SparkContext(conf = sparkConf)
a = ["spark", "rdd", "python", "context", "create", "class"]
b = ["operation", "apache", "scala", "lambda","parallel","partition"]

c1 = sc.parallelize(a).map(lambda a: (a, 1))
c2 = sc.parallelize(b).map(lambda a: (a, 1))

for i in c1.fullOuterJoin(c2).collect(): print (i)
```

Similarly, for each element **(k, w)** in *other*, the resulting RDD will either contain all *pairs* **(k, (v, w))** for **v** in *self*, or the *pair* **(k, (None, w))** if no elements in *self* have key **k**.

The Figure 2.2 shows the execution result of the previous code.

```
Console   Progress
<terminated> ex7-1-2.py [C:\Users\Mario\AppData\Local\Programs\Python\Python36\python.exe]
[Stage 1:============================>                          (8 + 8) / 16]
[Stage 1:================================>                      (9 + 7) / 16]
[Stage 1:==============================================================(16 + 0) / 16]
                                                                  ('python', (1, None))
('spark', (1, None))
('context', (1, None))
('create', (1, None))
('parallel', (None, 1))
('lambda', (None, 1))
('class', (1, None))
('rdd', (1, None))
('scala', (None, 1))
('operation', (None, 1))
('apache', (None, 1))
('partition', (None, 1))
```

Figure 2.2: Perform a right outer join of self and other

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

```python
# Imports the PySpark libraries
from pyspark import SparkConf, SparkContext

# The 'os' library allows us to read the environment variable SPARK_HOME defined in the IDE environment
import os

# Configure the Spark context to give a name to the application
sparkConf = SparkConf().setAppName("ex7-1-2")
sc = SparkContext(conf = sparkConf)

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

c1 = sc.parallelize(a)
c2 = sc.parallelize(b)
```

– **How many times the character 's' appears in a** –

```python
char = 's'
# how many times the character "s" appears in a
```

```
var = c1.flatMap(lambda x: x).map(lambda x: (x==char, 1)).
    reduceByKey(lambda x, y: x + y).collect()

print("\nHow many times the character 's' appears in 'a':\t",
    var[1][1])
```

– **How many times the character 's' appears in b** –

```
char = 's'
# how many times the character "s" appears in b
var = c2.flatMap(lambda x: x).map(lambda x: (x==char, 1)).
    reduceByKey(lambda x, y: x + y).collect()

print("\nHow many times the character 's' appears in 'b':\t",
    var[1][1])
```

– **How many times the character 's' appears in all a and b** –

```
char = 's'
# how many times the character "s" appears in all a and b
c3 = c1.union(c2)
var = c3.flatMap(lambda x: x).map(lambda x: (x==char, 1)).
    reduceByKey(lambda x, y: x + y).collect()
```

```
Console   Progress
<terminated> ex7-1-2.py [C:\Users\Mario\AppData\Local\Programs\Python\Python36\python.exe]

How many times the character 's' appears in 'a':        3


[Stage 2:>                                                          (0 + 8) / 8]
[Stage 3:>                                                          (0 + 8) / 8]

How many times the character 's' appears in 'b':        1


[Stage 4:>                                                         (0 + 8) / 16]
[Stage 4:===>                                                      (1 + 8) / 16]
[Stage 4:==============================>                           (8 + 8) / 16]
[Stage 4:=================================================>       (13 + 3) / 16]
[Stage 5:>                                                         (0 + 8) / 16]
[Stage 5:==========================>                               (7 + 8) / 16]
[Stage 5:==============================>                           (8 + 8) / 16]
[Stage 5:=====================================================>   (14 + 2) / 16]

How many times the character 's' appears in all 'a' and 'b':     4
```

Figure 2.3: How many times the character 's'

3 DataFrame API and Spark SQL

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

```python
import json
# Imports the PySpark libraries
from pyspark import SparkConf, SparkContext
from pyspark.sql import SQLContext

# The 'os' library allows us to read the environment variable SPARK_HOME defined in the IDE environment
import os

# Configure the Spark context to give a name to the application
sparkConf = SparkConf().setAppName("ex7-1-2")
sc = SparkContext(conf = sparkConf)
sqlContext = SQLContext(sc)

data = sqlContext.read.json("../datasets/students.json")
data.show()
```

```
Console  Progress
<terminated> ex7-2.py [C:\Users\Mario\AppData\Local\Programs\Python\Python36\python.exe]
+------------------+------------------+----------+---------+------+----+
|            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|
+------------------+------------------+----------+---------+------+----+
```

Figure 3.1: DataFrame initial

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

```
from pyspark.sql.functions import col,when, mean
[...]
mean_points = round(data.select([mean('points')]).head()[0],
    3)

data.show()

ch_null_points = data.withColumn("points",
            when(col("points").isNull(), mean_points).
            otherwise(col("points")))
ch_null_points.show()
```

```
Console  Progress
<terminated> ex7-2.py [C:\Users\Mario\AppData\Local\Programs\Python\Python36\python.exe]
+------------------+------------------+----------+---------+------+----+
|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|     null|  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|11.737|  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|              null|   Bridget|    Twain|   6.0|  18|
|          Business|      7 March 1980|   Darlene|    Mills|  19.0|  19|
|    Data Analytics|      June 2, 1985|   Zachary|     null|  10.0|  20|
+------------------+------------------+----------+---------+------+----+
```

Figure 3.2: Replace the null value(s) in column 'points'

3.2 Replace the null value(s) in column dob and column last_name by "unknown" and "- -" respec- tively

```
from pyspark.sql.functions import col,when, mean
[...]
ch_null_dob = ch_null_points.withColumn("dob",
              when(col("dob").isNull(), "unknown").
```

```
                otherwise(col("dob")))

ch_null_lastName = ch_null_dob.withColumn("last_name",
                when(col("last_name").isNull(), "--").
                otherwise(col("last_name")))
                ch_null_lastName.show()
```

```
Console  Progress
<terminated> ex7-2.py [C:\Users\Mario\AppData\Local\Programs\Python\Python36\python.exe]
+------------------+------------------+----------+---------+------+----+
|            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|11.737|  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|
+------------------+------------------+----------+---------+------+----+
```

Figure 3.3: Replace the null value(s) in column 'dob' and column 'last_name'

3.3 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

```
import json, re
# Imports the PySpark libraries
from pyspark import SparkConf, SparkContext
from pyspark.sql.functions import col,when, mean, udf
from pyspark.sql.types import StringType
[...]
# Returns the number of the month that corresponds to a
   character string
```

```python
def monthToNumber(string):
  m = {
  'jan': 1,
  'feb': 2,
  'mar': 3,
  'apr': 4,
  'may': 5,
  'jun': 6,
  'jul': 7,
  'aug': 8,
  'sep': 9,
  'oct': 10,
  'nov': 11,
  'dec': 12
  }
  s = string.strip()[:3].lower()
  out = m[s]
  return out

# Changes the way a date is represented (DD-MM-YYYY)
def dateFormatChange(dat):
  # Return all non-overlapping matches of pattern in string, as a list of strings
  line = re.findall(r"[\w']+", dat)
  if "unknown" not in line:
    for i in line:
      # If the string is a digit
      # otherwise it is the month
      if i.isdigit():
        # If the number of elements of the digit is greater
        # than two is that it is a year, otherwise it is the day
        if len(i)>2:
          y = i
        else:
          d = i
      else:
        m = str(monthToNumber(i))
        line = d + "-" + m + "-" + y
  else:
    line = ''.join(line)
  return line

def newDateValues(value):
  return dateFormatChange(value)

udfValueToCategory = udf(newDateValues, StringType())
```

```
            new_dates = ch_null_lastName.withColumn("dob",
                udfValueToCategory("dob"))
new_dates.show()
```

```
Console   Progress
<terminated> ex7-2.py [C:\Users\Mario\AppData\Local\Programs\Python\Python36\python.exe]
+------------------+----------+----------+---------+------+----+
|            course|       dob|first_name|last_name|points|s_id|
+------------------+----------+----------+---------+------+----+
|Humanities and Art|14-10-1983|      Alan|      Joe|  10.0|   1|
|  Computer Science| 26-9-1980|    Martin|  Genberg|  17.0|   2|
|    Graphic Design| 12-6-1982|     Athur|   Watson|  16.0|   3|
|    Graphic Design|  5-4-1987|  Anabelle|  Sanberg|  12.0|   4|
|        Psychology| 1-11-1978|      Kira| Schommer|  11.0|   5|
|          Business| 17-2-1981| Christian|   Kiriam|  10.0|   6|
|  Machine Learning|  1-1-1984|   Barbara|  Ballard|  14.0|   7|
|     Deep Learning| 13-1-1978|      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| 12-6-1975|     Holly| Schwartz|  12.0|  11|
|  Computer Science|  2-7-1985|     April|    Black|11.737|  12|
|  Computer Science| 22-7-1980|     Irene|  Bradley|  13.0|  13|
|        Psychology|  7-2-1986|      Mark|    Weber|  12.0|  14|
|       Informatics| 18-5-1987|     Rosie|   Norman|   9.0|  15|
|          Business| 10-8-1984|    Martin|   Steele|   7.0|  16|
|  Machine Learning|16-12-1990|     Colin| Martinez|   9.0|  17|
|    Data Analytics|   unknown|   Bridget|    Twain|   6.0|  18|
|          Business|  7-3-1980|   Darlene|    Mills|  19.0|  19|
|    Data Analytics|  2-6-1985|   Zachary|       --|  10.0|  20|
+------------------+----------+----------+---------+------+----+
```

Figure 3.4: Convert all the dates into DD-MM-YYYY format

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

```
from pyspark.sql.functions import udf, datediff, to_date, lit,
    unix_timestamp
[...]
date_fmt = '%Y-%m-%d'
# Current date
today = datetime.today()
today = today.strftime(date_fmt)

# Converts the days into years and returns them as integers
def ageValues(value):
  if(value == None):
    return value
  else:
    return int(value/365)
```

```python

udfValue = udf(ageValues, IntegerType())

# Difference of days between the current date and the student's date
age = new_dates.withColumn("age",
        datediff(to_date(lit(today)),
        to_date(unix_timestamp('dob', "dd-MM-yyyy").cast("timestamp"))))

# Update age to years
age = age.withColumn("age", udfValue("age"))
```

Console Progress

<terminated> ex7-2.py [C:\Users\Mario\AppData\Local\Programs\Python\Python36\python.exe]

```
+------------------+----------+----------+---------+------+----+----+
|            course|       dob|first_name|last_name|points|s_id| age|
+------------------+----------+----------+---------+------+----+----+
|Humanities and Art|14-10-1983|      Alan|      Joe|  10.0|   1|  33|
|  Computer Science| 26-9-1980|    Martin|  Genberg|  17.0|   2|  36|
|    Graphic Design| 12-6-1982|     Athur|   Watson|  16.0|   3|  34|
|    Graphic Design|  5-4-1987|  Anabelle|  Sanberg|  12.0|   4|  30|
|        Psychology| 1-11-1978|      Kira| Schommer|  11.0|   5|  38|
|          Business| 17-2-1981| Christian|   Kiriam|  10.0|   6|  36|
|  Machine Learning|  1-1-1984|   Barbara|  Ballard|  14.0|   7|  33|
|     Deep Learning| 13-1-1978|      John|       --|  10.0|   8|  39|
|  Machine Learning|26-12-1989|    Marcus|   Carson|  15.0|   9|  27|
|           Physics|30-12-1987|     Marta|   Brooks|  11.0|  10|  29|
|    Data Analytics| 12-6-1975|     Holly| Schwartz|  12.0|  11|  41|
|  Computer Science|  2-7-1985|     April|    Black|11.737|  12|  31|
|  Computer Science| 22-7-1980|     Irene|  Bradley|  13.0|  13|  36|
|        Psychology|  7-2-1986|      Mark|    Weber|  12.0|  14|  31|
|       Informatics| 18-5-1987|     Rosie|   Norman|   9.0|  15|  30|
|          Business| 10-8-1984|    Martin|   Steele|   7.0|  16|  32|
|  Machine Learning|16-12-1990|     Colin| Martinez|   9.0|  17|  26|
|    Data Analytics|   unknown|   Bridget|    Twain|   6.0|  18|null|
|          Business|  7-3-1980|   Darlene|    Mills|  19.0|  19|  37|
|    Data Analytics|  2-6-1985|   Zachary|       --|  10.0|  20|  32|
+------------------+----------+----------+---------+------+----+----+
```

Figure 3.5: Calculate the current age of all students

3.5 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

```
# Calculation of the standard deviation
sdv = round(age.select([stddev('points')]).head()[0], 3)
# Calculation of the standard average or mean
mean_points = round(age.select([mean('points')]).head()[0], 3)
# if his point is larger than 1 standard deviation of all points,
# then we update his current point to20,
dev = age.withColumn("points",
      when(col("points")> (mean_points+sdv+1), 20).
      otherwise(col("points")))
```

Console Progress

<terminated> ex7-2.py [C:\Users\Mario\AppData\Local\Programs\Python\Python36\python.exe]

```
|            course|       dob|first_name|last_name|points|s_id| age|
+------------------+----------+----------+---------+------+----+----+
|Humanities and Art|14-10-1983|      Alan|      Joe|  10.0|   1|  33|
|  Computer Science| 26-9-1980|    Martin|  Genberg|  20.0|   2|  36|
|    Graphic Design| 12-6-1982|     Athur|   Watson|  20.0|   3|  34|
|    Graphic Design|  5-4-1987|  Anabelle|  Sanberg|  12.0|   4|  30|
|        Psychology| 1-11-1978|      Kira| Schommer|  11.0|   5|  38|
|          Business| 17-2-1981| Christian|   Kiriam|  10.0|   6|  36|
|  Machine Learning|  1-1-1984|   Barbara|  Ballard|  14.0|   7|  33|
|     Deep Learning| 13-1-1978|      John|       --|  10.0|   8|  39|
|  Machine Learning|26-12-1989|    Marcus|   Carson|  15.0|   9|  27|
|           Physics|30-12-1987|     Marta|   Brooks|  11.0|  10|  29|
|    Data Analytics| 12-6-1975|     Holly| Schwartz|  12.0|  11|  41|
|  Computer Science|  2-7-1985|     April|    Black|11.737|  12|  31|
|  Computer Science| 22-7-1980|     Irene|  Bradley|  13.0|  13|  36|
|        Psychology|  7-2-1986|      Mark|    Weber|  12.0|  14|  31|
|       Informatics| 18-5-1987|     Rosie|   Norman|   9.0|  15|  30|
|          Business| 10-8-1984|    Martin|   Steele|   7.0|  16|  32|
|  Machine Learning|16-12-1990|     Colin| Martinez|   9.0|  17|  26|
|    Data Analytics|   unknown|   Bridget|    Twain|   6.0|  18|null|
|          Business|  7-3-1980|   Darlene|    Mills|  20.0|  19|  37|
|    Data Analytics|  2-6-1985|   Zachary|       --|  10.0|  20|  32|
+------------------+----------+----------+---------+------+----+----+
```

Figure 3.6: Calculate the standard deviation

3.6 Create a histogram on the new points created in the task 5

```
import matplotlib.pyplot as plt
[...]
```

```
sqlContext.registerDataFrameAsTable(dev, "bb")
columns_num = [2, 4]
# Extract the columns that interest (points and names)
df2 = dev.select(*(dev.columns[i] for i in columns_num))

# Compute histogram
def histogram(par):
  points = list(map(lambda x: x[0], par))
  name = map(lambda x: x[1], par)
  plt.xlabel('Points')
  plt.ylabel('Names')
  plt.title('Task-6 Histogram')
  plt.barh(range(len(points)), points, color = 'blue')
  plt.yticks(range(len(points)), name)
  plt.show()

# Put in RDD format to work with the histogram
par = df2.rdd.map(lambda x: (x[1], x[0]))
histogram(par.take(20))
```

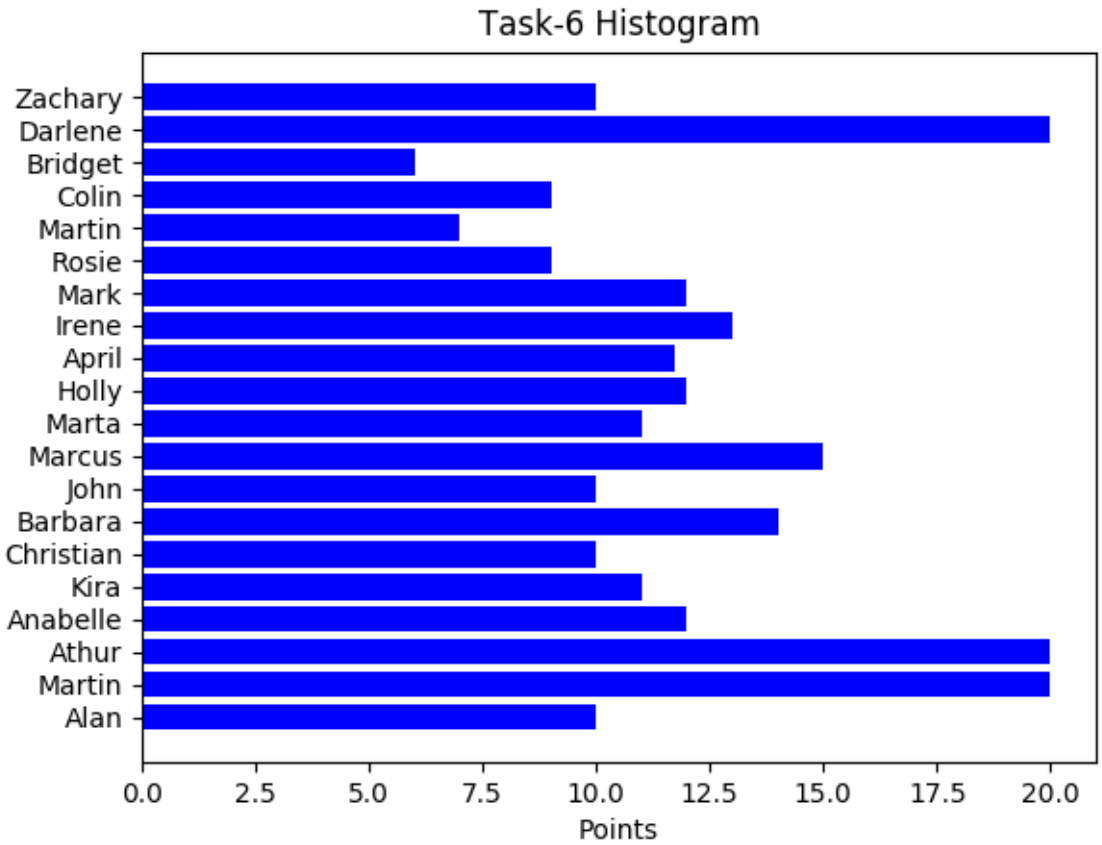

Figure 3.7: Histogram on the new points

4 BONUS: Querying and Updating Documents

4.1 Use dataset students.txt for this exercise. Complete several tasks using basic querying, updating, and inserting mongoDB statements.

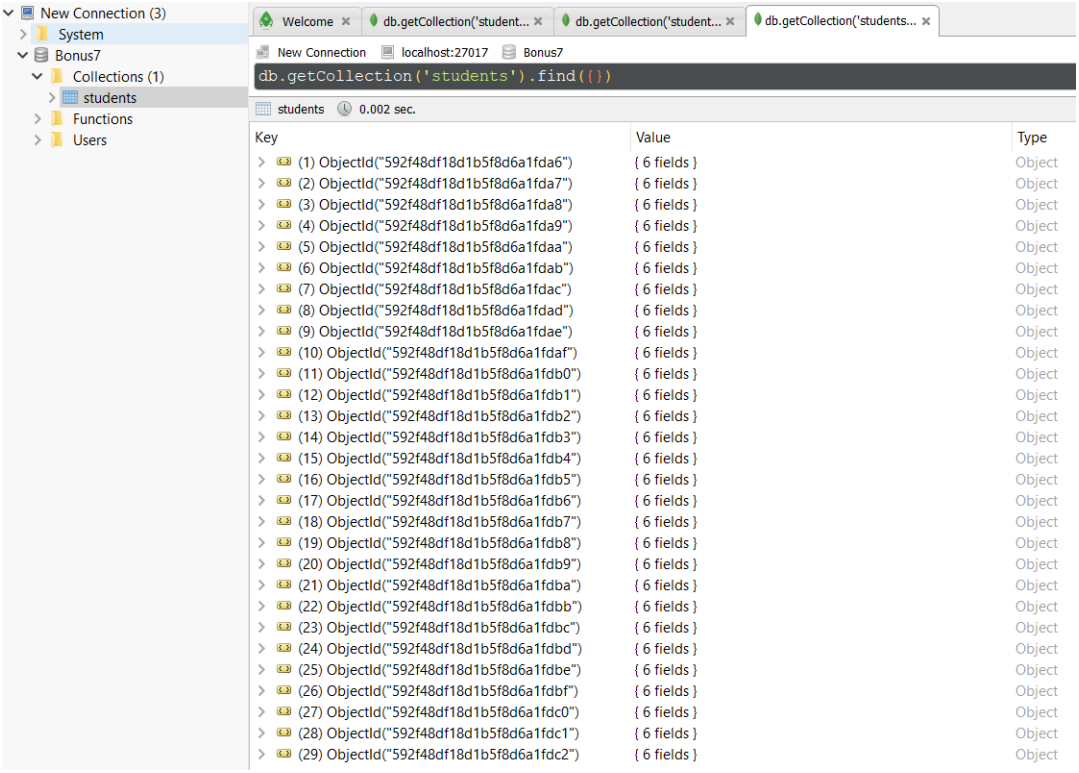

Figure 4.1: Students collections

4.1.1 Display only students' last name and students' course gpas. For the course gpas, only display 5 elements starting at the 5th index in the course gpas array

```
from pymongo import MongoClient
client = MongoClient('mongodb://localhost:27017/')
[...]
db = client['Bonus7']
collection = db['students']
cur = collection.find({}, {"name.last":1, "course_gpas":{"
    $slice": [5, 5]}})

for doc in cur:
  print(doc)
```

```
Console  Progress
<terminated> ex1.py [C:\Users\Mario\Anaconda3\python.exe]
{'_id': ObjectId('592f48df18d1b5f8d6a1fda6'), 'name': {'last': 'Holland'}, 'course_gpas': [3.49, 3.96, 2.22, 2.24, 2.9]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fda7'), 'name': {'last': 'Mills'}, 'course_gpas': [3.49, 3.02, 2.91, 3.74, 2.95]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fda8'), 'name': {'last': 'Cotter'}, 'course_gpas': [2.51, 3.31, 2.5, 3.99, 3.13]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fda9'), 'name': {'last': 'Aldwell'}, 'course_gpas': [2.56, 2.82, 3.9, 3.4, 2.55]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdaa'), 'name': {'last': 'Walker'}, 'course_gpas': [2.74, 3.29, 3.79, 2.79, 3.79]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdab'), 'name': {'last': 'Barrett'}, 'course_gpas': [3.23, 3.94, 3.86, 2.48, 3.91]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdac'), 'name': {'last': 'Edwards'}, 'course_gpas': [2.88, 3.41]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdad'), 'name': {'last': 'Caldwell'}, 'course_gpas': [3.29, 2.94, 3.43, 2.97, 3.6]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdae'), 'name': {'last': 'Roberson'}, 'course_gpas': [3.98, 2.5, 2.95, 3.59, 3.05]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdaf'), 'name': {'last': 'Bass'}, 'course_gpas': [2.37, 3.85, 2.14, 2.89, 2.94]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdb0'), 'name': {'last': 'Reyes'}, 'course_gpas': [2.69, 3.48, 3.79, 2.35, 2.67]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdb1'), 'name': {'last': 'Lynch'}, 'course_gpas': [3.73, 2.74, 3.2, 3.66, 3.89]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdb2'), 'name': {'last': 'Hampton'}, 'course_gpas': [2.24, 2.03, 3.56, 2.36, 2.48]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdb3'), 'name': {'last': 'Maxwell'}, 'course_gpas': [3.91, 2.62, 3.34, 3.84, 3.17]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdb4'), 'name': {'last': 'Arnold'}, 'course_gpas': [2.26, 2.43, 2.64, 3.69, 3.87]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdb5'), 'name': {'last': 'Mills'}, 'course_gpas': [3.03, 2.66, 3.77, 3.38, 3.97]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdb6'), 'name': {'last': 'Martinez'}, 'course_gpas': [3.55, 2.81, 3.99, 2.71, 3.13]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdb7'), 'name': {'last': 'Ford'}, 'course_gpas': [2.22, 3.23, 3.6, 3.85, 2.09]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdb8'), 'name': {'last': 'Washington'}, 'course_gpas': [2.34, 3.05, 3.57]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdb9'), 'name': {'last': 'Steele'}, 'course_gpas': [2.63, 3.61, 2.88, 3.18, 2.77]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdba'), 'name': {'last': 'Norman'}, 'course_gpas': [3.52, 2.55, 3.84, 3.33, 3.85]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdbb'), 'name': {'last': 'Bradley'}, 'course_gpas': [2.81, 2.42, 3.12, 3.55, 3.44]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdbc'), 'name': {'last': 'Weber'}, 'course_gpas': [3.61, 2.41, 2.26, 2.98, 3.16]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdbd'), 'name': {'last': 'Black'}, 'course_gpas': [3.78, 3.05, 2.33, 3.05, 3.69]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdbe'), 'name': {'last': 'Schwartz'}, 'course_gpas': [3.02, 3.53, 2.14, 3.36, 2.5]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdbf'), 'name': {'last': 'Brooks'}, 'course_gpas': [2.79, 3.87, 2.34, 3.3, 2.42]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdc0'), 'name': {'last': 'Carson'}, 'course_gpas': [3.66, 2.42, 3.74, 2.66, 2.11]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdc1'), 'name': {'last': 'Henderson'}, 'course_gpas': [3.69, 3.58, 2.35, 3.11, 3.8]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdc2'), 'name': {'last': 'Ballard'}, 'course_gpas': [3.08, 3.95, 2.11, 3.82, 3.9]}
```

Figure 4.2: Display only students' last name and students' course gpas

4.1.2 Display only students' last name and students' SAT test scores

```python
from pymongo import MongoClient
client = MongoClient('mongodb://localhost:27017/')
[...]
db = client['Bonus7']
collection = db['students']
cur = collection.find({"test_scores":
  {'$elemMatch':{"test": "SAT", "score":{"$gt":700} }}},
    {"name.last":1, "test_scores":
    {"$slice": -1}}).sort("test_scores.score",-1)

for doc in cur:
  print(doc)
```

```
Console  Progress
<terminated> ex1.py [C:\Users\Mario\Anaconda3\python.exe]
{'_id': ObjectId('592f48df18d1b5f8d6a1fdba'), 'name': {'last': 'Norman'}, 'test_scores': [{'test': 'SAT', 'score': 786}]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdaa'), 'name': {'last': 'Walker'}, 'test_scores': [{'test': 'SAT', 'score': 783}]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdbc'), 'name': {'last': 'Weber'}, 'test_scores': [{'test': 'SAT', 'score': 766}]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdbd'), 'name': {'last': 'Black'}, 'test_scores': [{'test': 'SAT', 'score': 763}]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fda7'), 'name': {'last': 'Mills'}, 'test_scores': [{'test': 'SAT', 'score': 759}]}
{'_id': ObjectId('592f48df18d1b5f8d6a1fdb9'), 'name': {'last': 'Steele'}, 'test_scores': [{'test': 'SAT', 'score': 718}]}
```

Figure 4.3: Display only students' last name and students' SAT test scores

4.1.3 Add new empty list called evaluations:[] to all students using one mongoDB command

```
# Add new empty list called evaluations:[]
collection.updateMany({}, { "$set" : {"evaluation":[]} })
# Add "eval comment": "This student is very clever" in "evaluation"
collection.updateMany({},{ "$push" : {"evaluation.0.eval_comment":"This student is very clever"}}
)
# Add "eval comment": "This student always submits exercises ontime" in "evaluation"
collection.updateMany({}, {"$push": {"evaluation.0.eval_comment":"This student always submits exercises ontime"}}
)
```

The version of pymongo has given me quite a few problems with the updateMany function, so I had to perform those three functions using the MongoDB console as Figure 4.4 shows.

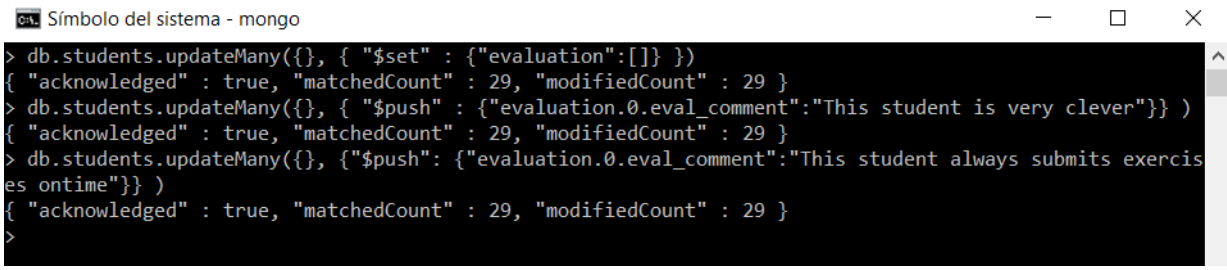

Figure 4.4: MongoDB Shell

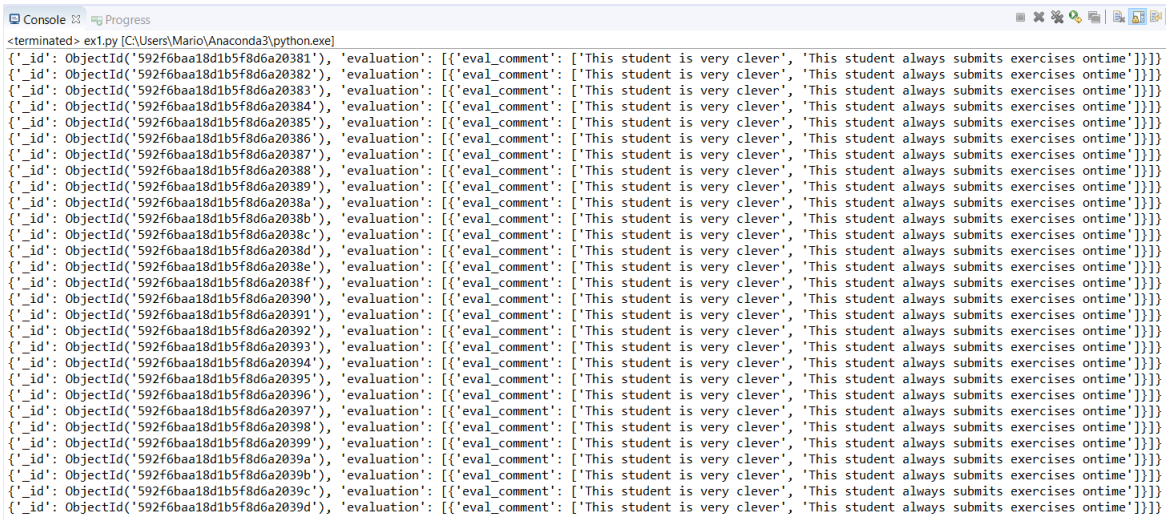

Figure 4.5: Add new empty list called evaluations:[ ]