Lab7 : Spark Exercises

Dealing with massive amounts of data often requires parallelization and cluster computing; Apache Spark is an industry-standard for doing just that. In this lab, we introduce the basics of PySpark, Spark's Python API for this. Apache Spark is an open-source, general-purpose distributed computing system used for big data analytics. Spark is able to complete jobs substantially faster than previous big data tools (i.e. Apache Hadoop) because of its in-memory caching and optimized query execution. Spark provides development APIs in Python, Java, Scala, and R. On top of the main computing framework, Spark provides machine learning, SQL, graph analysis, and streaming libraries.

In this lab, you will get to explore a bit of Apache Spark. In particular, you will learn about transforming RDDs and actions on RDDs as well as manipulating DataFrames in PySpark.

Installation:

- Check if Java 11 is installed
  ```
  java -version
  ```
  If not install by using the below commands or any other way you wish
  ```
  sudo apt update
  sudo apt install default-jdk
  java -version
  ```

- Now install pyspark
  ```
  pip3 install pyspark
  ```

Resources:

- Dataframes
- RDD
- http://users.csc.calpoly.edu/~dekhtyar/369-Winter2019/papers/pyspark.pdf
- https://spark.apache.org/docs/1.6.1/sql-programming-guide.html

Part A: Clustering using Spark

In this part you will write Spark code to process a groceries basket dataset in parallel to construct clusters in such a way that entities in one cluster are more closely related, i.e., similar to each other than entities in other clusters.

Input: Dataset containing the list of items purchased by customers in a single bill

Tasks:

Create a file named **`q1.py`** and do the following tasks.

a. Create an RDD of (item, item) pairs for each row in the dataset. There should not be any trivial pairs such as (cheese,cheese).
b. Using the RDD from (a), create an RDD with ((item1, item2), count) where count is the total number of bills that (item1, item2) both occur in, for all item pairs (item1, item2). **Save this dataset as count.csv** in the same folder**.**
c. Using the dataset from (b) **print top 5 (item1, item2) pairs** on the basis of count.

Note: you should consider unordered pairs of items, i.e., (A, B) is the same as (B, A).

Part B: Log Analysis

One of the most common uses of Spark is analyzing and processing log files. In this part, you will put Spark to good use for analysis web server data
DataSet : Access Log Data Set use access log file

Here is an extract of what the log looks like:

31.56.96.51 - - [22/Jan/2019:03:56:16 +0330] "GET /image/60844/productModel/200x200 HTTP/1.1" 200 5667 "https://www.zanbil.ir/m/filter/b113" "Mozilla/5.0 (Linux; Android 6.0; ALE-L21 Build/HuaweiALE-L21) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.158 Mobile Safari/537.36" "-"

31.56.96.51 - - [22/Jan/2019:03:56:16 +0330] "GET /image/61474/productModel/200x200 HTTP/1.1" 200 5379 "https://www.zanbil.ir/m/filter/b113" "Mozilla/5.0 (Linux; Android 6.0; ALE-L21 Build/HuaweiALE-L21) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.158 Mobile Safari/537.36" "-"

Each part of this log entry is described below.

- 127.0.0.1 - - This is the IP address (or host name, if available) of the client (remote host) which made the request to the server.

- - - The "-" in the output indicates that the requested piece of information (user identity from remote machine) is not available.

- [01/Aug/1995:00:00:01 -0400] The time that the server finished processing the request. The format is: [day/month/year:hour:minute:second timezone]
    - ####day = 2 digits
    - ####month = 3 letters
    - ####year = 4 digits
    - ####hour = 2 digits
    - ####minute = 2 digits
    - ####second = 2 digits
    - ####zone = (+ | -) 4 digits

- "GET /images/launch-logo.gif HTTP/1.0" This is the first line of the request string from the client. It consists of three components: the request method (e.g., GET, POST, etc.), the endpoint (a Uniform Resource Identifier), and the client protocol version.

- 200 This is the status code that the server sends back to the client. This information is very valuable, because it reveals whether the request resulted in a successful response (codes beginning in 2), a redirection (codes beginning in 3), an error caused by the client (codes beginning in 4), or an error in the server (codes beginning in 5). The full list of

possible status codes can be found in the HTTP specification (RFC 2616 section 10).

- 1839 This indicates the size of the object returned to the client, not including the response headers. If no content was returned to the client, this value will be "-" (or sometimes 0).
- "https://www.zanbil.ir/m/filter/b113" indicates the referer (a misspelling of referrer) header contains the address of the previous web page from which a link to the currently requested page was followed.
- Last Entry displays the information about the device which issued the requests

Tasks:

Create a file named q2.py and do the following tasks.

A. Loading the log file

Download the log file and write a function to load it in an RDD.

B. Parsing data using regex and Spark

Divide the single string log into multiple columns given below and truncate remaining data.

a. Remote Host
b. Request Timestamp
c. Request Method (ex GET,POST, DELETE)
d. Response Code
e. Response Length

C. Cleaning data

Remove rows containing null fields and print the count of such rows containing null fields
Print number of bad rows as "Number of bad Rows : <count>"

D. Data Analysis

Perform the following analysis using the filtered dataset from task C

a. HTTP status analysis

Print the count of number of request group by status code.

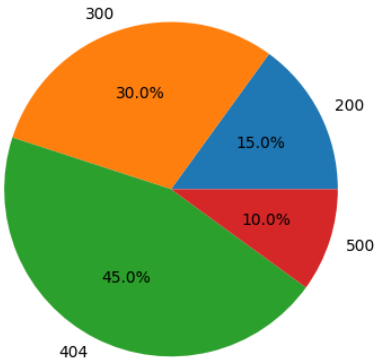
Output format :

```
HTTP status analysis:
status     count
200   11330
206   52
302   3498
304   658
404   251
```

b. HTTP status analysis with matplotlib

Create a pie chart using matplotlib on the basis of http status analysis.You can do that by creating a two map functions in which first map function will create a array that corresponds to the status code and second map function will creates a array that corresponds to the (count of requests with that status code divided by total number of request in the dataset).

Output format:



c. **Frequent Hosts**
Print the number of requests each host issued using group-by for each host.

Output format:

```
Frequent Hosts:
host  count
10.131.2.1 1626
10.128.2.1 4257
10.130.2.1 4056
10.131.0.1 4198
10.129.2.1 1652
```

d. **Find the number of unique hosts.**
Using reduce by calculating the number of unique hosts in the dataset.

Output format

```
Unique hosts:
1000
```

e. **Find the number of unique hosts daily.**

Calculate the number of unique hosts in the entire log on a day-by-day basis. Print the list sorted by increasing day of the month which includes the day of the month and the associated number of unique hosts for that day.

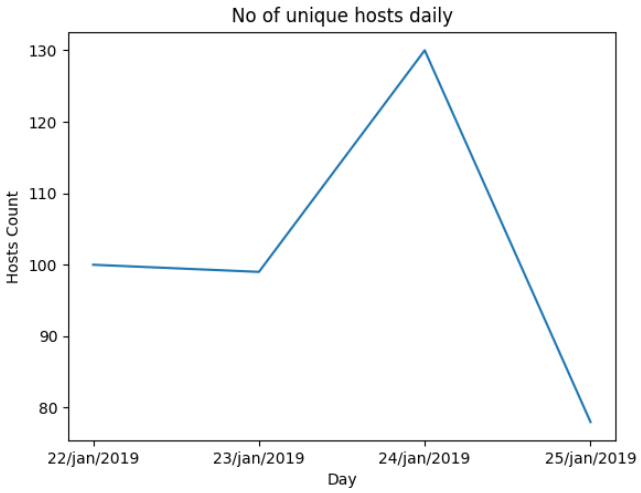
Output format:

```
Unique hosts per day:
day   hosts
22/Jan/2019     151
```

f. **Visualizing the Number of Unique Daily Hosts**

Using the dataset from task e, create a line graph using matplotlib. X-axis denotes the date from the access log and y-axis will give the number of unique requests issued on that day.

Output format:



g. **Which client HTTP requests lead to FAILURE(SERVER/CLIENT) ?**

Find the top 5 clients whose request to the web server either leads to server failures or client failures.Server failure is denoted by response codes between 400-499 and client failure is denoted by response code b/w 500-599. Print the clients in sorted order of failed requests

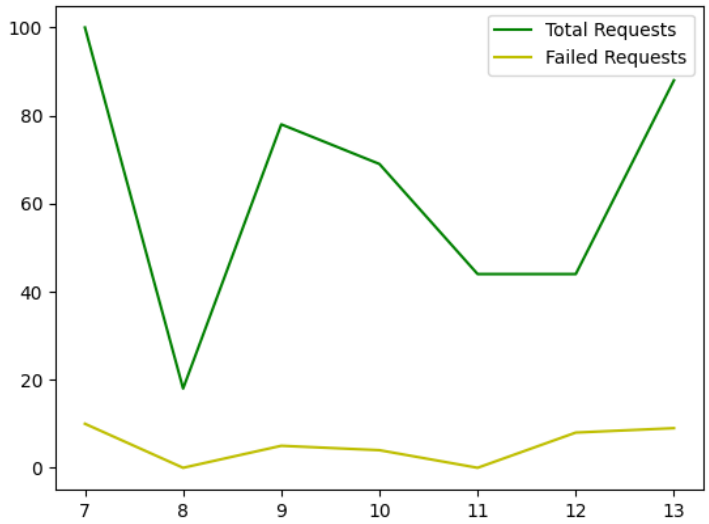
Output format :

```
Failed HTTP Clients:
10.131.2.1
10.131.2.5
10.135.85.1
10.215.25.2
```

`10.131.2.2`

h. Visualize the requests issued during each hour.

Plot the number of requests issued each hour and failed requests due to errors in that hour. Create multiple line graphs for day 22/JAN/2019 from the access log where line 1 will denote the total number of requests issued hourly and line 2 will denote the number of requests failed with status code greater than > 399.



i. What is the most active hour of day?

For each day in the dataset find the most active hour of day i.e. maximum no of requests issued in that hour

.

Output format:

```
Active Hours:
day   hour
22/Jan/2019 03:00
```

j. Response length statistics

Find the average, minimum and maximum size of response returned by the web server.

Output format :

```
Response length statistics:
Minimum length  100
Maximum length  200
Average length  150
```

Submission Instructions:

- Make a folder named **<rollnumber1>-<rollnumber2>** containing **q1.py**, **q2.py**
- Your submission should not contain any other files
- **Zip** the folder and upload it to the **moodle**
- ***Before submitting, check all the file names and run your code one more time to ensure that everything is working fine as we'll only run the python files described above via a script and ANY naming errors will result in you not receiving any marks.***

Grading Rubric

Exercise	Marks
Part A	20
Part B - Task A	5
Part B - Task B	15
Part B - Task C	10
Part B - Task D	50 (5 for each sub part)
Total	100