UE20CS322 Big Data Assignment 1

Analysis of Sofia air quality

This is the first assignment for the UE20CS322 Big Data Course at PES University. The assignment consists of 2 tasks and focuses on running MapReduce jobs to analyze air quality of various regions in USA.

The files required for the assignment can be found here.

Assignment Objectives and Outcomes

1. This assignment will help students become familiar with the Map Reduce programming environment and the HDFS.
2. At the end of this assignment, the student will be able to write and debug MapReduce code.

Ethical practices

Please submit original code only. You can discuss your approach with your friends but you must write original code. All solutions must be submitted through the portal. We will perform a plagiarism check on the code and **you will be penalized if your code is found to be plagiarized**.

The Dataset

You will be working with the following set of attributes.

attribute	Description
sensor_id	id of the sensor
location	location number of sensor
lat	latitude of the area covered by sensor
lon	longitude of the area covered by sensor
timestamp	Timestamp when scanner took readings
pressure	Pressure of the area scanned by sensor
temperature	Temperature of the area scanned by sensor
humidity	Humidity of the area

Software/Languages to be used:

1. Python `3.10.x`
2. Hadoop `v3.3.3` only

Tasks Overview:

1. Load the data into HDFS.
2. Create `mapper.py` and `reducer.py` for Task 1 and Task 2
3. Run your code on the sample dataset until you get the right answer
4. Submit the files to the portal

Submission Deadline

27th August, 11:59 PM

Submission Guidelines

You will need to make the following changes to your mapper.py and reducer.py scripts to run them on the portal

- Include the following shebang on the first line of your code
  `#!/usr/bin/env python3`
- Convert your files to an executable
  `chmod +x mapper.py reducer.py`
- Convert line breaks in DOS format to Unix format (this is necessary if you are coding on Windows - your code will not run on our portal otherwise)
  `dos2unix mapper.py reducer.py`

Check out a detailed list of submission guidelines here.

Task Specifications

Task 1

Problem Statement

Find the number of records per date with respect to a set of conditions

Description

Find the number of records per date that satisfy a set of conditions and display them in sorted fashion.
All the following conditions must be satisfied by a record.

Attribute	Description
location	**1700 < location< 2500**
Sensor Id	**< 5000**

Attribute	Description
pressure	**>= 93500.00**
humidity	**>= 72.00**
temperature	**>= 12.00**

Comments

Ignore records which do not satisfy the mentioned conditions. You do not require any command line arguments for this task. Additionally, if any of the required attributes contain `NaN`, ignore the record. Please perform type conversions for the records wherever necessary.

Output Format

Please print each date separated by the number of records as shown in examples

For a subset of 2018 year, this is the expected output

```
2018-01-07 1
2018-01-11 3
2018-01-12 3
2018-01-16 1
2018-01-21 1
2018-01-28 1
2018-01-31 1
```

For subset of 2017 year, this is the expected output:

```
2017-10-01 178
2017-10-02 152
2017-10-03 98
2017-10-04 279
2017-10-05 489
2017-10-06 473
2017-10-07 5
2017-10-08 4
```

```
2017-10-09 1
2017-10-10 8
2017-10-11 203
2017-10-12 71
2017-10-13 76
2017-10-14 838
2017-10-15 279
2017-10-16 1238
2017-10-17 555
2017-10-18 153
2017-10-19 85
2017-10-20 35
2017-10-21 21
2017-10-22 79
2017-10-23 4983
2017-10-24 184
2017-10-25 1
2017-10-26 96
2017-10-27 1034
```

Task 2

Problem Statement

Find the number of records per date with respect to a entered latitude and longitude with a set of conditions

Description

Find the number of records per date where the distance between the start coordinates of the accident and a given pair of coordinates - (`LATITUDE`, `LONGITUDE`) is within `D` and must satisfy a given set of conditions . You will be using Euclidean Distance to find whether the distance calculated is within `D`.
The set of following conditions must be satisfied by a record.

Attribute	Description
humidity	**48 < humidity < 54**

Attribute	Description
Temperature	**20 < Temperature < 24**

Comments

Ignore records which do not satisfy the mentioned conditions. You do require 3 command line arguments for this task, the first is the queried distance, second is the latitude and the third is the longitude. Additionally, if any of the required attributes contain `NaN`, ignore the record. Please perform type conversions for the records wherever necessary.

Output Format

Please print each date separated by the number of records as shown in examples

For a subset of 2018 year, with given latitude=40 and longitude=25 and distance=20 this is the expected output:

```
2018-01-01 59
2018-01-02 96
2018-01-03 79
2018-01-04 90
2018-01-05 91
2018-01-06 146
2018-01-07 188
2018-01-08 6
2018-01-10 1
2018-01-11 67
2018-01-12 15
2018-01-17 7
2018-01-27 42
2018-01-28 56
2018-01-29 38
2018-01-30 33
```

For subset of 2017 year with given latitude=40 and longitude=25 and distance=20, this is the expected output:

```
2017-10-01 123
2017-10-02 65
2017-10-03 29
2017-10-04 706
2017-10-05 831
2017-10-06 1574
2017-10-07 379
2017-10-08 50
2017-10-09 455
2017-10-10 498
2017-10-11 280
2017-10-12 90
2017-10-13 114
2017-10-14 75
2017-10-15 464
2017-10-16 1982
2017-10-17 659
2017-10-18 210
2017-10-19 190
2017-10-20 131
2017-10-21 122
2017-10-22 130
2017-10-26 29
2017-10-27 72
```

Helpful Commands

Running the MapReduce Job without Hadoop

A MapReduce job can also be run without Hadoop. Although slower, this utility helps you debug faster and helps you isolate Hadoop errors from code errors.

```
cat path_to_dataset | python3 mapper.py
[command line arguments] | sort -k 1,1 |
python3 reducer.py [command line arguments] > output.txt
```

Starting Hadoop

If you are running Hadoop for the first time, run

```
hdfs namenode -format
```

Hadoop can be started using the following command.

```
$HADOOP_HOME/sbin/start-all.sh
```

You can view all the Java processes running on your system using `jps`.
After running `jps` you should see the following processes running (in any order) along with their process IDs:

```
DataNode
SecondaryNameNode
Jps
ResourceManager
NameNode
NodeManager
```

HDFS Operations

The `HDFS` supports all file operations and is greatly similar to the file system commands available on Linux.
You can access `HDFS` on command line using `hdfs dfs` and use the `-` prefix before the file system command to execute general Linux file system commands.

Loading a file into HDFS

A file can be loaded into `HDFS` using the following command.

```
hdfs dfs -put path_to_file /hdfs_directory_path
```

Listing files on HDFS

Files can be listed on `HDFS` using

```
hdfs dfs -ls /hdfs_directory_path
```

Similarly, `HDFS` also supports `-mkdir`, `-rm` and more.

Running a MapReduce Job

A MapReduce job can be run using the following command

```
hadoop jar path-to-streaming-jar-file \
-input path_to_input_folder_on_hdfs \
-output path_to_output_folder_on_hdfs \
-mapper absolute_path_to_mapper.py command_line_arguments \
-reducer absolute_path_to_reducer.py command_line_arguments
```