School of Computer Science

COMP47470

Project 2
Coding Assessment

Teaching Assistant:	Patrick Cormac English
Coordinator:	Dr Anthony Ventresque
Date:	Tuesday 9th November, 2021
Total Number of Pages:	5

General Instructions

- Some of the following problems may require a sequence of operations to solve - it is not necessary to design a single operation that will generate the answer, although you should provide an answer that best represents your learning to date - as an example, a very inefficient answer will be acceptable, but may not receive the same grade as an answer that creates a solution in half the operations!
- This project is largely based on the material covered in the lab sessions. However, you may desire additional functionality (such as Java string manipulation functions for Hadoop). We would encourage external research in this regard - the documentation for Scala/Java etc. will be helpful for you. As an example, Scala functionality may be explored **here**. The MapReduce documentation can be found **here** and the latest GraphX docs are **here .**
- We ask you to hand in an archive (zip or tar.gz) of your solution: code/scripts, README.txt file describing how to run your programs, and a short pdf report of your work.
- The report should not be longer than 10 pages (this is not a hard constraint though).
- The report must be submitted as a PDF, but stylistically you may use whichever software you like to prepare it (e.g. Word, Google Docs, LaTeX) as long as any code included can be copy-pasted!
- Your answer in the report should include the following:
  - The answer to the question if requested
  - The code you used - either include a snippet (if the code is short) or reference an attached script/txt file. You may use scala scripting for this assignment, but a text file including **all lines of code you used (NB - running them sequentially must replicate your work)** will be accepted.
  - A brief explanation of what your code is doing/how it operates. This can be very short (one sentence) as long as it demonstrates an understanding of each portion of your code. For extensive scala commands (e.g. with multiple functions strung together) please make sure you explain what each portion is doing.
- The breakdown of marks for the project will be as follows:
  - Exercise 1 (Bash & Data Management): 30%
  - Exercise 2 (Hadoop Graph Processing): 20%
  - Exercise 3 (Spark): 20%
  - Exercise 4 (GraphX): 30%
- **Due date: 12/12/2021**

1 Bash & Data Management

Dataset: Hyphen-delineated with Format: [INDEX, Name, age, country, height, hair colour] For this section, you have been provided with an incorrectly formatted dataset. The first half of this section will involve using Bash commands/scripts to correct the issues in the data. You will then write two Bash scripts to insert the cleaned data into both an SQL and MongoDB database, before performing a number of operations on the data there.

```
wget --no-check-certificate
↪ 'https://docs.google.com/uc?export=download&id=1FDqa2hJ9rPhLG1YEcvHlfUliLym1ul4u'
↪ -O bashdm.csv
```

Dictionary file for Q4 (format: [INDEX&CODE&NAME]):

```
wget --no-check-certificate
↪ 'https://docs.google.com/uc?export=download&id=1h62yHjiRXMElbusQdCOM6WO_kgr-sQiv'
↪ -O dictionary.csv
```

1.1 Clean-up Tasks:

NB - for this section, there are multiple tools that may satisfy the required correction. You may use whichever tool you feel fits best - we will evaluate on the correctness of your output

1. There is an error in the name column - every name starts with an erroneous string "#]". Write a command/script to remove this string from the name column.
2. Currently the database is delineated by a "-" character. Write a command/script to convert this file into a comma delineated file. **NB - there are several triple-barelled names in the dataset, in the format "first-second-thirdname". These must not be split into separate columns. You may address these in a separate preparatory step if you wish**
3. There are two columns in this dataset with non-useful values. Write a script/command that identifies the columns that do not change, and removes them from the output.
4. Lastly, the country names in this dataset have been incorrectly added as country codes. Using the dictionary file provided (**Link above**) find and replace each country code with its correct country name.

NB - the completion of the above tasks is desired, but if you progress to the section below without fully cleaning the dataset, you will only be penalised once for the omission. Please note that your commands below etc. may have to be altered somewhat to accommodate the erroneous data

1.2 Data management Tasks

Using your cleaned dataset from above, complete the following tasks:

1. Create 2 Bash scripts, one for SQL and one for MongoDB, that inserts your cleaned records into the respective database software. For SQL, you should generate the table within your Bash script. **NB - please perform this the way shown in the labs. Do not use mysqlimport etc.**

2. (SQL) Find the average height per country
3. (SQL) Find the maximum height per hair colour
4. (MongoDB) Write a short script that adds a new characteristic to each person - an ID number. You may generate this number however you like (e.g. a counter) within your script.
5. (MongoDB) Find the name of the person with the lowest value for height.

2 Simple Hadoop Graph Processing

In this section you will run Hadoop MapReduce jobs on a dataset of harbours and routes. You may use the Hadoop environment from previous projects/labs. The data is in the following format: [Harbour, Harbour Number, Route, Route Number]. Harbour names are separated by a "-", route names by a "_".

```
wget --no-check-certificate
↪ 'https://docs.google.com/uc?export=download&id=1SrurEuPrw04S6afIPXl1WKuL73JRuULT'
↪ -O hadoop.csv
```

Using a MapReduce script, complete the following graph exploration tasks. You may "merge" the outputs of multiple MapReduce tasks with further MapReduce tasks. You may also create transformed outputs using MapReduce that generate your final answers (e.g. your final output does not need to be generated from the original bacon.csv file). Remember - your MapReduce tasks can take multiple inputs! You may present multiple mapreduce outputs as your answer, if the answer is split across them (e.g. for Q4 below - you may find A-B and B-C separately, although a better answer might have them in a single output).

1. Create a list of the number of routes that connect to each harbour
2. Route "Wolfsbane_Nine" is associated with only one harbour - what is it?
3. What harbours are connected by route Carnation_Sixty-seven(No.1223).
4. Which harbours fielded emergency routes - these are routes whose route number begins with "911"
5. "Midnightblue-Epsilon" is connected to two other harbours by a route- what are they? As a hint, consider how you could compare multiple MapReduce outputs with different Map conditions to find the link. For this question you do not have to combine your separate operations into one script.

2.1 Notes:

- This section can be completed by modifying the "WordCount" example provided to you in Lab 4. Please look at the solutions document for that lab to orient yourself. You will be editing the WordCount file and compiling it inside your docker container - this will limit your debugging potential so it may be preferable to test some changes in a separate Java environment before inputting them (see the last point below).

- The majority of modifications will be made in the TokenizerMapper class (hint: look at the context.write() and word.set() functions) and the IntSumReducer class (take a look at the code after "public void reduce").
- Please remember to look at the Hadoop documentation - many of the functions will not accept strings/integers, and will instead require the creation of new variables of the appropriate type - e.g. in the context.write() function in the TokenizerMapper class, the number "one" is not an integer or a string - it is an "IntWritable" - if we wanted to modify this write to include a different integer (using 7 as an example), we would have to make a new IntWritable as so: "IntWritable t1 = new IntWritable(7);" and include this in the write: "context.write(word, t1);. Similarly the "key" variable in IntSumReducer is not a string, but a "Text" item that can be constructed similarly. This information is available in the code, but might require a close reading!
- The following resource may be useful to you: MapReduce Tutorial
- You may find it helpful to create a smaller dataset to probe functionality on! Alternatively, you might require a Java IDE to test some specific functionality - if you do not have Java on your system, there are some basic implementations available online - see here

3 Spark

```
wget --no-check-certificate
↪ 'https://docs.google.com/uc?export=download&id=1Kay7TkrEr-3u1Q34ObOuEhhibpwAaMPj'
↪ -O spark.csv
```

In this section, you will be using a dataset of restauraunts and reviews. The dataset can be found here (Format: [INDEX, Name, Region, Number of reviews, Review Text] . Load the dataset into Spark, and address the following tasks:

1. Determine the number of records the dataset has in total.
2. Find the restaurant with the highest number of reviews.
3. Determine the restaurant with the longest name.
4. Find the number of reviews for each region.
5. Determine the most frequently occurring term in the review column that doesn't include one of the following stop words: "A", "The", "of".

4 GraphX

For this section, you will be assessing the same harbour data as in the Hadoop Graph Processing section, but we have also provided a mirrored version to allow you to treat this as an undirected graph. You may use either version of the data as appropriate. The following documentation will be very useful for you: **GraphOps Documentation** , **GraphX coding guide**. For all of the following, your answer need not be a single command - you may create outputs and perform follow-up operations on those outputs, but GraphX should

be the "driver" in your answers - e.g. an answer that exclusively uses SparkSQL unions (and no GraphX operations) to make links would be undesirable. Download the data using the following code (remember to make it all one line!). You may use whichever data is preferable for any given question!

For the mirrored dataset:

```
wget --no-check-certificate
↪ 'https://docs.google.com/uc?export=download&id=19Uubzr_jcXGiVse5EJCOmBpLNEHH7YXY'
↪ -O hadoop_mirrored.csv
```

For a dataset of edges:

```
wget --no-check-certificate
↪ 'https://docs.google.com/uc?export=download&id=1QmpOehpXOWGyUZ6AlNpmBrVxY9tpOLmj'
↪ -O hadoop_mirrored.csv
```

1. Import the data and create a graph representing the data
2. Generate an array of each harbour's connected routes - consult the spark documentation to identify the most suitable method for this. Alternatively, you may use Spark commands to generate this information.
3. Which harbour(s) is/are served by route "Porium_Thirty-one"?
4. Which harbour has the most routes associated with it. If there are multiple harbours with the same number of routes, list them all.
5. Which harbour is connected to the most other harbours - for this, you can transform your earlier outputs, or use a new GraphX method. If there are multiple harbours with the same number of connections, list them all. You may find the above edge data useful!