Project 3

For this project you'll have a group. Those have been automatically assigned and you can find your groups's information on the assignment link **You should not discuss the project with anyone outside your group. If you are stuck please contact Dr. Post or the TA, Khuzaima Hameed.** If you run into issues with your partners, let me know immediately. I can't help if it is the day before the due date and you tell me you haven't heard from them. You'll be stuck doing it yourself then!

This project is meant to assess your ability to understand streaming data concepts and how to use spark to handle streaming data. You'll write a .py file with answers to the question prompts below.

- **Both group members should submit the files.**
- **When submitting, please leave a note about how working with your group went.**
- You should each be doing the entire project as a collaboration. Repeat: You should not split the project up and only do certain parts yourself. **If you have no part in writing one section, you will receive a 0 for that part of the project.**

Goals

For this, we'll write a python script to send `.csv` files to a folder. You'll have pyspark monitoring that folder, reading in the data, transforming it, and writing the result out to a file. - The data we'll use comes from the UCI machine learning repository. The study was about trying to detect heavy drinking during a bar crawl based on alcohol readings and accelerometer data from cell phones. We'll simply consider the accelerometer data for this example. + This data has five columns: time (in unicode seconds), pid (person id), x, y, and z information from the accelerometer. + We'll want to take in these values, convert the x, y, and z components to a magnitude (I think that makes sense to do here...)

Set up for Creating Files

- Read in the `all_accelerometer_data_pids_13.csv` file
- There are lots of records per user id. Let's just concern ourselves with person SA0297 and PC6771. Create two data frames, one for person SA0297's data and one for person PC6771's data.
- Set up a for loop to write 500 values at a time (not randomly, from first line) for SA0297 to a .csv file in a folder for that person's data. Similarly output values for PC6771 to another folder. This should be done in the same loop. The loop should then delay for 20 seconds after writing to the files. You should not run the loop yet!

Reading a Stream

- We're going to read in two streams (via two separate queries) for this part.
- Setup the schema and create an input stream from the `csv` folder for SA0297
- Setup the schema and create another input stream from the `csv` folder for PC6771

Transform/Aggregation Step

- Now, for each stream we'll do a basic transformation of the x, y, and z coordinates into a magnitude. This can be done via
$$mag = \sqrt{x^2 + y^2 + z^2}$$
- All of the necessary functions are available from the `spark.sql.functions` submodule. I did this using the `.select()` method with the transformations done within `.select(transform here)` but there are multiple ways to do this.
- We want to keep the time and pid columns, this new column, and drop the original x, y, and z columns. This was easy to incorporate within the `.select()` above!

Writing the Streams

- Lastly, we'll write each stream out to their own csv file(s).
  - Use the `csv` output `format`
  - Use the `append` `outputMode`
  - You'll need to include an option for `checkpointlocation`.
    - The syntax is `.option("checkpointlocation","path")`
  - Lastly, start each query via the `.start()` method

Leave these queries running.

Now open a python console and submit the necessary code to run the loop that outputs data.

Let this all run for about 5 minutes. Stop both queries using the `.stop()` method.

Now What?

- The output is in `.csv` files for each user but not very easy to deal with because it is split up
- Using code from here, we can read in all the pieces and output each to their own single `.csv` file.

```
# Use PySpark to read in all "part" files
allfiles =  spark.read.option("header","false").csv("/destination_path/part-*.csv")
# Output as CSV file
allfiles \
.coalesce(1) \
.write.format("csv") \
.option("header", "false") \
.save("/destination_path/single_csv_file/")
```

You should turn in these `.csv` files (two of them) and include the code to make them (and everything else above!) in the .py file.

File to Submit

Upload the `.py` file with all the code (separated into clear sections with **extensive commenting**) to the assignment link along with the two `.csv` files.

Rubric for Grading (total = 100 points)

Item	Points	Notes
Creation of Data to Write to .csv files	15	Worth either 0, 3, 6, . . . , 15
Reading streams	20	Worth either 0, 4, 8, . . . , 20
Transforming streams	35	Worth either 0, 5, . . . , 35
Writing streams	20	Worth either 0, 4, 8, . . . , 20
Combining .csv parts	10	Worth either 0, 5, or 10

Notes on grading:

- For each item in the rubric, your grade will be lowered one level for each each error (syntax, logical, or other) in the code and for each required item that is missing or lacking a description.
- **You should use Good Programming Practices when coding (see wolfware). If you do not follow GPP you can lose up to 25 points on the project.**

Be sure to include comments describing what you are doing, even when not explicitly asked for! Points will be deducted from appropriate sections as appropriate.