UE21CS343AB2 Big Data Assignment 2

Working with Spark and Kafka

This is the third assignment for the UE21CS343AB2 Big Data course. The assignment consists of two tasks. The first task focuses on working with Apache Spark and the second task focuses on working with Apache Kafka.

The files required for the assignment can be found here.

Assignment Objectives and Outcomes

1. Get familiar with Apache Spark and learn about the speed and efficiency of Spark.
2. Work with Apache Kafka and understand how a Producer - Consumer model works.

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 submied 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.**

Submission Deadline

24th October 2023, 11:59 PM

The portal uptimes will be communicated to you in due course. Please do not wait till the last minute to submit your code since wait times may be long.

Submission Link

Common submission link for the portal: https://www.bigdata-pesu.tech/

The time durations for which the portal will be up will be communicated to you via the Whatsapp group.

Submission Guidelines

You will need to make the following changes to your Python scripts before submitting them.

1. Include the following shebang at the top of your Python scripts.

```
#!/usr/bin/env python3
```

2. Convert your files to executable using the following command.

```
chmod +x *.py
```

3. Convert line breaks in DOS format to Unix format (this is necessary if you are coding on Windows without which your code will not run our portal).

```
dos2unix *.py
```

Software/Languages to be used:

1. Python `3.10.x`
2. Hadoop `v3.3.3`
3. Apache Spark `v3.3.0`
4. Apache Kafka `v3.3.1`

Additionally, the following Python libraries are required:

1. `pyspark==3.3.0`
2. `kafka-python==2.0.2`

No other libraries are allowed.

Task 1 - Spark

Problem Statement

Given the dataset, ascertain the ten states exhibiting the highest incidence of criminal activities, as well as identifying the presiding judge overseeing the greatest number of criminal cases.

Dataset

A drive link with the datasets specific to each task is provided.

Task 1.1

Utilize the follwing files cases_2012.csv,cases_2013.csv,cases_2014.csv and cases_state_key.csv, to extract and determine the top 10 states exhibiting the highest crime rates over the three-year period. You may find these files here

Input characteristics

The metadata for cases files can be found here.

The file cases_state_key.csv has the following attributes:

1. year: Holds value for the year
2. state_code: This holds an integer value to represent a state
3. state_name: Consists of the name of the state.

Output characteristics

The output for this sub-task should be a list of states, sorted in descending order based on their crime rates.

Task 1.2

Utilize the follwing files case_2012.csv, case_2013.csv, case_2014.csv, judges_clean.csv, judge_case_merge_key.csv and acts_section.csv in order to identify the judge who has presided over the highest number of criminal cases.

Input Characteristics

The metadata for judges_clean.csv and acts_section.csv can be found at Judges_clean and acts_section.

The judge_case_merge_key.csv has the following attributes:

1. ddl_case_id: Holds the case ID.
2. ddl_filing_judge_id: Holds the filing judge ID.
3. ddl_decision_judge_id: Holds the decision judge ID.

Output Characteristics

The output for this sub-task should be the Judge ID who has presided over the most number of criminal cases.

Final Output for Task 1

The final output of the task should be a tuple of outputs from both the tasks. The output will thus have a tuple of two elements:

1. List of states
2. Judge ID

How to Run your code

You are required to load the dataset into a Spark RDD or DataFrame and perform the required operations on it. You can use the spark-submit command to run your code. The command should be as follows:

```
$SPARKHOME/bin/spark-submit spark-solution.py <input_paths> <output_path>
```

Give the input paths in the following file order:

1. case_2012.csv
2. case_2013.csv
3. case_2014.csv
4. cases_state_key.csv
5. judges_clean.csv
6. judge_case_merge_key.csv
7. acts_section.csv

Important notes

1. The input dataset filename and output filenameshould not be hardcoded. The filenames should be passed as command line arguments.
2. Loops are not allowed. You must use Spark's RDDs and DataFrames to solve the problem statement.
3. The filename of the code should be spark-solution.py .
4. Usage of direct file interaction commands such as python's open() is not allowed to load the data, It's only allowed to write the final output. You must use Spark's RDDs and DataFrames to solve the problem statement.
5. Make sure to always fetch the available SparkContext / SparkSession object using the getOrCreate() method instead of creating a new one. This will prevent any errors while attempting to create a new SparkContext/SparkSession object to connect to the Spark cluster.
6. Print out the tuple (final result) to the file given in output_path.

Sample output

The sample output file is provided here. You can use the diff command to check if your output is correct.

```
diff solution.txt your-solution.txt
```

Task 2 - Kafka

Dataset

You will be working with the a social media dataset. The dataset is artificailly generated. You can find the required files here. The rows in the dataset are of three types, based on the social media action they are describing.

1. `Like` - Describes when a user likes a post of another user. Format: [like @user_liking @user_who_posted post_id]
2. `Share` - Describes when a user shares the post of another user. Format: [share @user_sharing @user_who_posted post_id @user_shared_to_1 @user_shared_to_2 ...]
3. `Comment` - Describes when a user comments on the post of another user. Format: [comment @user_commenting @user_who_posted post_id "comment enclosed in double quotes"] NOTE: Post_id is unique to each user, but not unique across users.

Problem Statement

Using the dataset provided to you, generate output files for 3 different clients based on their requirements. Client 1: Wants all the comments which have been made on a particular user's account. Client 2: Wants a count of the number of likes for every post made by a user Client 3: Wants to calculate the popularity of a user, based on the number of likes, comments, and shares on their profile.

Description

Use the student-dataset.txt file to prepare the desired outputs for each client. You are required to use Kafka's producer and consumer APIs to solve the problem statement.

A sample input of the dataset is as follows. Consider the scenario

Instagram user2 made a post (with id 1) . user1 liked the post, commented “So beautiful”, and shared it with three other users: user3, user4, and user 5

Instagram user3 made a post (with id 1). user1 liked that post, and shared it with one other user: user5. user2 liked the post and commented "Brilliant" on it.

Instagram user2 made a post (with id 2). user1 liked that post, commented “Stunning” and shared it with seven other users: user3, user4, user5, user6, user7, user8, user9

```
comment @user1 @user2 1 “So beautiful”
like @user1 @user2 1
share @user1 @user2 1 @user3 @user4 @user5
like @user1 @user3 1
share @user1 @user3 1 @user5
like @user2 @user3 1
comment @user2 @user3 1 "Brilliant"
comment @user1 @user2 2 “Stunning”
like @user1 @user2 2
share @user1 @user2 2 @user3 @user4 @user5 @user6 @user7 @user8 @user9
```

Towards this, you will have to write three consumer files, one for each client.

Client1

List down all the comments received on posts for all users.

Output format after running consumer1.py, for client1: Print the json.dumps of the dictionary in the following format.

```
{
    "@username1" : [
        "comment1",
        "comment2"
    ],
    "@username2" : [
        "comment1",
        "comment2"
    ]
    ...
}
```

Sample output after running consumer1.py, for client1:

```
{
    "@user2": [
        "So beautiful",
        "Stunning"
    ],
    "@user3": [
        "Brilliant"
    ]
}
```

Client2

List down the number of likes received on different posts for each user. Output format after running consumer2.py, for client2: Print the json.dumps of a dictionary contianing the number of likes per post of all the users in the following format:

```
{
    "@username1" : {
        "post-id-1" : no_of_likes,
        "post-id-2" : no_of_likes
    },
    ...
}
```

Sample output after running consumer2.py, for client2:

```
{
    "@user2": {
        "1": 1,
        "2": 1
    },
    "@user3": {
        "1": 2
    }
}
```

Client3

Calculate the popularity of a user based on the number of likes, shares and comments on the user's profile.

Formula to calculate the popularity: Popularity=(number of likes + 20*(number of people shared to) + 5*(number of comments))/1000

Output format: Print the json.dumps of a dictionary of the following format.

```
{
    "@username_1": popularity,
    "@username_2": popularity,
    ...
}
```

Sample output:

```
{
    "@user2": 0.212,
    "@user3": 0.027
}
```

The logic for the producer and consumer is up to you. The input to the producer file will be streamed through the standard input.

You are required to use the `kafka-python` library to solve the problem statement. You should have four files, one for the producer and three for the three different consumers. The producer should be named `kafka-producer.py` and the consumers should be named `kafka-consumer1.py`, `kafka-consumer2.py` and `kafka-consumer3.py`.

It is recommended for you to use three topics to solve the assignment. All three topic names will be passed as command line arguments to all four files, and you may make use of them as required.

To test your code, run the consumer files first in three seperate terminals and then the producer file in a fourth separate terminal.

```
python3 kafka-consumer1.py topicName1 topicName2 topicName3 > output1.json
```

```
python3 kafka-consumer2.py topicName1 topicName2 topicName3 > output2.json
```

```
python3 kafka-consumer3.py topicName1 topicName2 topicName3 > output3.json
```

```
cat student_dataset.txt | python3 kafka-producer.py topicName1 topicName2 topicName3
```

Important!!

1. The topic name should not be hardcoded. Three topic names should be passed as a command line argument for both the producer and consumer files.
2. There is a special line in the end of the input file named `EOF`. This is done so that the consumer knows when to stop reading from the topic. You must not include this line in the output.
3. Usage of direct file interaction commands such as python's `open()` is not allowed. You must use Kafka's producer and consumer APIs to solve the problem statement.
4. **Sort the output in ascending order of the usernames.** .

Sample Input and Output

The sample input file is provided here. The sample output files are provided here. You can use the `diff` command to check if your output is correct.

```
diff <(jq --sort-keys . output1.json) <(jq --sort-keys . sample_output1.json)
```