Spark / Scala Exercise 2

Create a spark application written in scala that processes the data, and stores the entire output as specified below per the requirements and submission guidelines. Pay close attention to performance. Assume that the same code will be run on a much larger input dataset.

Overview

When processing large amounts of data over time it's important to process only the new data in incremental batches for performance and cost efficiency. This exercise is designed to simulate processing incremental and full batches.

The input data contains company attributes from many data sources. Each company will have a set containing one attribute value per attribute type. Each attribute has a numeric attribute type indicating the type of data in the attribute. For example, attribute_type 49 stores the industry of the company. Each unique source has a numeric source id. In every input batch, data is loaded from a subset of sources and may only cover part of the companies in the dataset. In other words, each input batch contains parts of the final company records you will produce in the output.

Requirements

Project

- Create a Scala SBT project
- Use the Spark **Scala API** and Dataframes/Datasets
    - Do not use Spark SQL with a sql string!
    - **If you write** `spark.sql` ("select ….") **you are doing it wrong!!**

- **This application should be close to production quality, with consideration of performance, ease of use, well organized and clean. Comment as necessary. Remove extra debug code or comment it out to avoid extra processing.**

Processing

Application Parameters

inputPath: String
Full path to the input data for each run.

outputPath: String

Full path to the output location where data will be written.

batchId: Long
The batch id to process

lastFullBatchId: Option[Long]
The batch id indicating the most recent full batch available, None for first run.

createFullBatch: Boolean
If true, then create a full batch by combining previous full batch plus incremental batches since.
If false, only create an incremental batch output with the changes in the current batch.

Attribute Rules

- For each company, take the set of all attributes with highest probability. There will be only one attribute value for each attribute_type.
- If probabilities are the same, then take the attribute value from the newer batch, followed by the lower source id as a tie-breaker

Run 1

- Pass parameters inputPath=”..input/run_1”, *batchId=495*, *lastFullBatchId=None, createFullBatch= true*.
- Since createFullBatch is true and lastFullBatchId is None, read in all input batches up to and including 495
- Produce a single batch output, applying the attribute rules
- Output format is Parquet and stored in the output directory provided. The output batch should be stored in its own partition with batch id 495 (same as input structure)

Run 2

- Pass parameters inputPath=”..input/run_2”, *batchId=597*, *lastFullBatchId=495, createFullBatch= false*.
- Read input batch *597* and the previous full batch output 495 you produced in run 1
- Process batch 597 using the same rules you’ve already implemented and produce only the incremental batch changes as follows:
    - Include the **set of attributes** for only the companies that have **at least one** change.
    - Store the incremental output in a partition, using batch id 597.

Run 3

- Pass parameters inputPath=”..input/run_3”, *batchId=648*, *lastFullBatchId=495, createFullBatch= true*.
- Read outputs for 495 (run 1) and incremental 597 (run 2) and combine with all data in this run to produce a full output with batch_id 648 in the output folder

Data

All data for the runs is available here. The input batches are split into 3 groups for clarity and to simulate multiple runs. The format is parquet files under each partition.

Input

input/run_1:
batch_id=73
batch_id=75
batch_id=76
...
batch_id=495

input/run_2:
batch_id=508
batch_id=514
…
batch_id=597

input/run_3:
batch_id=601
batch_id=605
…
batch_id=648

Output

output/batch_id=495
output/batch_id=597
output/batch_id=648

Schema

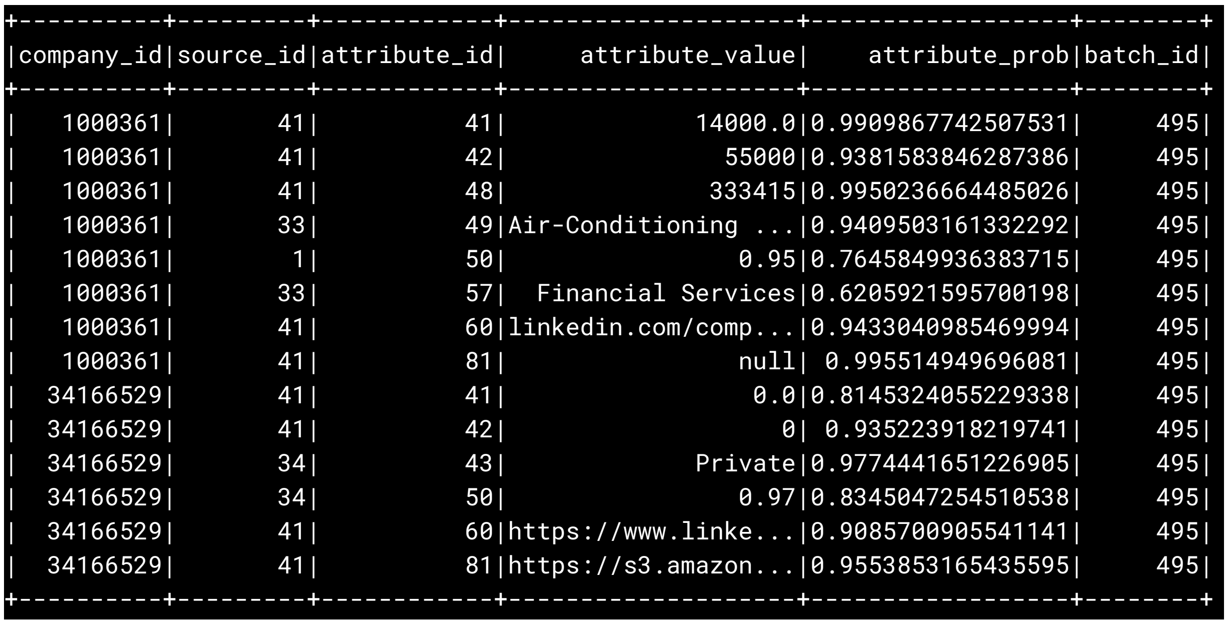
The input and output data schema should be the same:

```
batch_id: long
company_id: Integer
source_id: integer
```

attribute_type: integer
attribute_prob: double
attribute_value: string

Example Partial Output

Here is what the output will look like in the first run for 2 of the 1000 companies. Order of the columns doesn't matter since the schema is stored in parquet.



```
+----------+---------+------------+--------------------+------------------+--------+
|company_id|source_id|attribute_id|     attribute_value|    attribute_prob|batch_id|
+----------+---------+------------+--------------------+------------------+--------+
|   1000361|       41|          41|             14000.0|0.9909867742507531|     495|
|   1000361|       41|          42|               55000|0.9381583846287386|     495|
|   1000361|       41|          48|              333415|0.9950236664485026|     495|
|   1000361|       33|          49|Air-Conditioning ...|0.9409503161332292|     495|
|   1000361|        1|          50|                0.95|0.7645849936383715|     495|
|   1000361|       33|          57|  Financial Services|0.6205921595700198|     495|
|   1000361|       41|          60|linkedin.com/comp...|0.9433040985469994|     495|
|   1000361|       41|          81|                null| 0.995514949696081|     495|
|  34166529|       41|          41|                 0.0|0.8145324055229338|     495|
|  34166529|       41|          42|                   0| 0.935223918219741|     495|
|  34166529|       34|          43|             Private|0.9774441651226905|     495|
|  34166529|       34|          50|                0.97|0.8345047254510538|     495|
|  34166529|       41|          60|https://www.linke...|0.9085700905541141|     495|
|  34166529|       41|          81|https://s3.amazon...|0.9553853165435595|     495|
+----------+---------+------------+--------------------+------------------+--------+
```

Submission Guidelines

Please submit the entire scala sbt project with all code necessary to build and run the app. You can bundle it as a zip, tarball or github link. Also include a copy of the output folders that is generated from running the app.