In this exercise, you will need to write the enriched user interaction feedback to HDFS for billing and archiving purposes. Let's see how to write an HDFS sink for a PySpark consumer. For writing the data into file(i.e. Local filesystem, HDFS etc) sink, you must specify the following:

1. **Format:** It can be either of json, csv or parquet.
2. **Output Mode:** The default mode is 'append'. The other options are complete' and 'update'. For streaming use cases, where the data is being written to any kind of file system, the most widely used mode is 'append'.
3. **Path:** The path to the file where the data will be written.
4. **Checkpoint Location:** The path to the checkpoint directory. This checkpoint location has to be a path in an HDFS compatible file system.

This corresponding code segment can be as used for writing data to any file sink:

```
aggregatedDataStream.writeStream \
    .format("csv") \
    .outputMode("append") \
    .option("truncate", "false") \
    .option("path", "path/to/destination/dir") \
    .option("checkpointLocation", "path/to/checkpoint/dir") \
    .trigger(processingTime="1 minute") \
    .start() \
    .awaitTermination()
```

You have been provided with the Cloudera distribution of Spark and the file paths mentioned here will point to HDFS directly. If you write the path as 'op', it will point to the path '/user/root/op' in HDFS.

Notes:

- In case of subsequent runs of your code, you will need to change the checkpoint location and output path before every run.
- In the industry, the parquet format with snappy compression is favoured for Hive sink. However, for this task, it's not mandatory.