Healthcare Data Analysis Incremental Load in Cassandra Using PySpark

Assignment

You are required to build a Spark application that will process a daily CSV file from a HDFS folder and perform certain transformations on it, and then store the transformed data in a Cassandra table.

Here are the detailed steps:

1. The Spark application should run daily and pick the CSV file from the source folder in HDFS. The file will be named in this format: **health_data_<YYYYMMDD>.csv**, where **<YYYYMMDD>** is the date.

2. The application should perform data validation and data cleaning steps. **For example,** it should validate if all mandatory columns (like **patient_id**, **age**, **diagnosis_code**, etc.) are present and are in the correct format. It should also handle missing or inconsistent data appropriately.

3. If the data validation is successful, the Spark job should perform certain transformations, such as:

    a. **Disease Gender Ratio:** Calculate the gender ratio for each disease. This will help in identifying if a particular disease is more prevalent in a particular gender. For example, for each diagnosis_code, you could calculate the ratio of male to female patients.

b. **Most Common Diseases:** Find the top 3 most common diseases in the dataset. This will help in identifying the most prevalent diseases.

c. **Age Category:** Create age categories. For example, you can divide age into groups like '30-40', '41-50', '51-60', '61-70' and so forth. Then, calculate the number of patients in each age category for each disease. This can help understand the age distribution of different diseases.

d. **Flag for senior patients:** Flag patients who are senior citizens (typically, age >= 60 years). This might be useful information since senior citizens might require special healthcare attention or follow-up.

e. **Disease trend over the week:** If you have more than a week's data, you can calculate the number of cases of each disease for each day of the week to understand if there's a trend (more cases diagnosed on particular days).

4. The transformed data should then be loaded into a Cassandra table in upsert mode **(Separate table for each type of transformation).** This would involve creating stage and target tables, where the stage table will be used for the raw data and then upserted to the target table with transformed data. **Think about the correct attributes which can work like a key for upsert operation.**

5. If the entire process is successful, the input file should be moved to an **archive folder** in HDFS.

6. All Spark best practices and optimizations should be followed, such as partitioning the data properly, using broadcast variables for small lookups, avoiding shuffles, etc.

7. **Data backfilling** should be taken care of, so if any day's processing fails, the system should be able to process the failed day's data along with the current day's data.

Dataset - You can use **mock_data_generator.py** script to generate multiple csv files for each day, dates you can decide.

Each file will have these fields: **patient_id, age, gender, diagnosis_code, diagnosis_description, diagnosis_date**