Course 4 – Project

Design a batch ETL job using HDFS and Hive

Objective

The objectives of this project can be categorized as

- Design a full batch data pipeline
- How to use Hive to prepare raw data for data transformation
- How to use partitioning (sharding) in Hive

Data set

The data set is the one that you analyzed in Course 1 and it is STM GTFS data.

Problem statement

We get the information of STM every day and need to run an ETL pipeline to enrich data for reporting and analysis purpose in once a day batch job.

Pipeline schema

Normally, the schema used in a data pipeline are populated once it is installed on an environment. That includes databases and tables as well. Hence, you need to create all the tables and databases you need prior running the application.

Assignment requirements

1. Data pipeline installation
Create a directory on HDFS for staging area called **/user/[GROUP]/[YOUR NAME]/project4/** where **[GROUP]** is the program. Ask the instructor if you don't have it and **[YOUR NAME]** is a nickname of your choice with only lowercase letters.
Create a directory for each source table called **/user/[GROUP]/[YOUR NAME]/project4/[TABLE NAME]** where **[TABLE NAME]** is from the following list • trips • calendar_dates • routes
Create a database called **[GROUP]_[YOUR NAME]** in Hive. If you already have one, just use and don't try to create multiple databases.
Create staging tables called **ext_[TABLE NAME]**. Staging tables are external tables that point to staging directory of each source. • trips • calendar_dates • routes Note that the **LOCATION** for each table is HDFS path under staging directory.
Create a managed table called **enriched_trip** with Parquet encoding and partitioned by `wheelchair_accessible`

2. Extract data from STM to staging area
Download the data set of STM GTFS from http://stm.info/sites/default/files/gtfs/gtfs_stm.zip
Put extracted version into **/user/[GROUP]/[YOUR NAME]/project4/[TABLE NAME]** path on HDFS where **[TABLE NAME]** here is the name of file without extension. We just need the following tables • trips • calendar_dates • routes
3. Data pipeline
Enrich *trips* with *calendar dates* and *routes* and write it to the **enriched_trip** table. You could take any of the following options: 1. Implement a Scala/Java application and run the SQL query using Hive JDBC 2. Use the project of course 2 and 3 and customize it to write the files under proper partition folder on HDFS. Then use MSCK REPAIR command to recover partitions. The latter can be done using Hive JDBC.

Deliverables

Your application should be idempotent. It means, every time you run the application, you get the same result. It has to first check if the items exist or not. If yes, it will clear them up. An "item" here could refer to a file, folder, table, and etc.

Bonus

- Optimized JOIN to get enriched trips information with explanation in demo session
- Automating schema deployment (step 1)
- Automating staging phase (step 2)

Evaluation

1. Basic implementation **%80**
   Manual implementation of part 1 and part 2 and running SQL queries of enrichment through Hive JDBC
2. Advanced implementation **%10**
   Manual implementation of part 1 and part 2 and populating data through HDFS
3. Bonus section **%10**

Schema

Trip

Field Name	Data Type
trip_id	Integer
service_id	String
route_id	String
trip_headsign	String
wheelchair_accessible	Boolean

Calendar Date

Field Name	Data Type
service_id	String
date	String
exception_type	Integer

Route

Field Name	Data Type
route_id	Integer
route_long_name	String
route_color	String

Enriched Trip

Field Name	Data Type
trip_id	Integer
service_id	String
route_id	String
trip_headsign	String
wheelchair_accessible	Boolean
date	String
exception_type	Integer
route_long_name	String
route_color	String