**Project Delivery Documentation**

**project\_twitter\_listener.py**

This code uses the Twitter API to search for tweets containing the keyword "premier league" in English language, within a specific time frame (last 10 minutes). It retrieves up to 1000 tweets matching the search criteria and sends each tweet's relevant data (including user data) through a socket connection to a client.

The code first authenticates the user's access to the Twitter API by retrieving a bearer token from the environment variables. It then creates headers and a URL with query parameters for the search, specifying the keyword, start and end times, and other fields to be included in the response.

The code then connects to the Twitter API endpoint with the provided URL and headers, retrieves the JSON response, and parses it to extract the relevant data for each tweet and its author. The code sends the data for each tweet to the client through the socket connection.

Finally, the socket connection is closed.

**project\_spark\_application.py**

This code is written in Python using the PySpark library for processing structured data. It uses SparkSession to connect to a stream of data from a socket, reads it, and converts it into a DataFrame. It also defines a schema for the JSON data and applies it to the DataFrame.

The code then applies several functions from the PySpark SQL library to manipulate the DataFrame. It uses the from\_json function to parse the JSON data, the selectExpr function to select specific columns, and the withColumn function to create new columns based on user-defined functions (UDFs).

In particular, the code defines three UDFs:

extract\_hashtags to extract hashtags from the text column

remove\_urls to remove URLs from the text column

extract\_clubs to extract words from the text column that match a predefined list of words

Finally, the code uses the size function to count the number of hashtags, words, and mentioned clubs in each row, and selects a subset of columns to display as the output.

**hive\_script.hql**

This is a HiveQL script that creates external tables to store Twitter data in a partitioned manner and inserts the data into these tables.

The script starts by setting the partition mode to non-strict and creating a new database called Twitter\_Project. Then, it creates an external table called twitter\_landing\_data to store the landing data in a partitioned manner. The table has columns to store various Twitter metrics such as hashtags, words, mentioned clubs, retweet count, reply count, like count, quote count, impression count, user information such as user ID, username, user description, name, user verification, user creation date, and user metrics such as user followers count, user following count, and user tweet count.

The data is partitioned by year, month, day, and hour and is stored in the Parquet format. The data is stored in the location /home/itversity/twitter-landing-data\_test. The MSCK REPAIR TABLE command is used to repair the table's partitions.

Next, the script creates four new tables: tweet\_text, tweet\_metrics, user\_info, and user\_metrics. These tables are also partitioned by year, month, day, and hour and are stored in the Parquet format.

The tweet\_text table stores columns related to tweet text, such as ID, text, words, words count, hashtags, hashtags count, and mentioned clubs. The data for this table is inserted from the twitter\_landing\_data table.

The tweet\_metrics table stores columns related to tweet metrics such as retweet count, reply count, like count, quote count, and impression count. The data for this table is also inserted from the twitter\_landing\_data table.

The user\_info table stores columns related to user information such as user ID, username, user description, name, user verification, and user creation date. The data for this table is also inserted from the twitter\_landing\_data table.

The user\_metrics table stores columns related to user metrics such as user followers count, user following count, and user tweet count. The data for this table is also inserted from the twitter\_landing\_data table.

**SparkSQLApp.py**

The code you provided looks like a PySpark script to process Twitter data and create an aggregated table in Hive. Here is a brief summary of the code:

The script starts by importing the necessary libraries, including pyspark.sql and pyspark.sql.functions.

The script creates a SparkSession object with the name "twitter-processed-data".

The script loads the dimensions from the Hive tables using spark.table.

The script joins the dimensions to create the fact table.

The script aggregates the fact table to create multiple aggregations using groupBy and agg.

The script saves the aggregated table to Hive as a processed table using write.mode("overwrite").saveAsTable.

The script saves the aggregated table as Parquet files to /home/itversity/twitter-processed-data/fact\_table\_processed using write.mode("overwrite").parquet.

**twitter\_pipline\_script.py**

This is a Python script that executes a series of commands in the terminal to run a pipeline that collects Twitter data, processes it using Spark, and stores the results in Hive and HDFS.

Here's what the script does:

Starts a Twitter stream listener using a Python script called project\_twitter\_listener.py

Starts a Spark application using a Python script called project\_spark\_application.py

Creates dimension tables in Hive using a Hive script called hive\_script.hql

Runs a Spark job using a Python script called SparkSQLApp.py