**FINAL EVALUATION PROJECT REPORT**

**"COMPREHENSIVE ETL PIPELINE CAPSTONE: PARSING, TRANSFORMING, AGGREGATING, AND LOADING NESTED TWITTER DATA"**

**PREPARED BY:**

**ASSOCIATE ID: 2381869**

**ASSOCIATE NAME: ALAN BENO**

**GUIDED BY:**

**Trainer Coach**

**Karthick Selvam Harismitha Srinivasan**

**Contractor Jr.Executive-HR**

**CPO L&D GenC Training CPO L&D GenC Training**

**1. Introduction**

**1.1 Overview**

This capstone project, **“Comprehensive ETL Pipeline: Parsing, Transforming, Aggregating, and Loading Nested Twitter Data”**, demonstrates building an efficient ETL pipeline by:

* Ingesting nested JSON data from Twitter feeds.
* Parsing URLs and hashtags using custom UDFs.
* Aggregating daily statistics of trending domains and hashtags.
* Joining data with historical malicious user records.
* Loading structured results into a database for analytics.

**1.2 Objectives**

* Automate the ETL process for parsing, transforming, and loading Twitter data.
* Handle complex nested JSON data to create queryable tables.
* Aggregate daily trends of hashtags and tweeted domains.
* Filter malicious users using historical lookup tables.
* Develop expertise in Spark, SQL, and cloud-based data engineering pipelines.

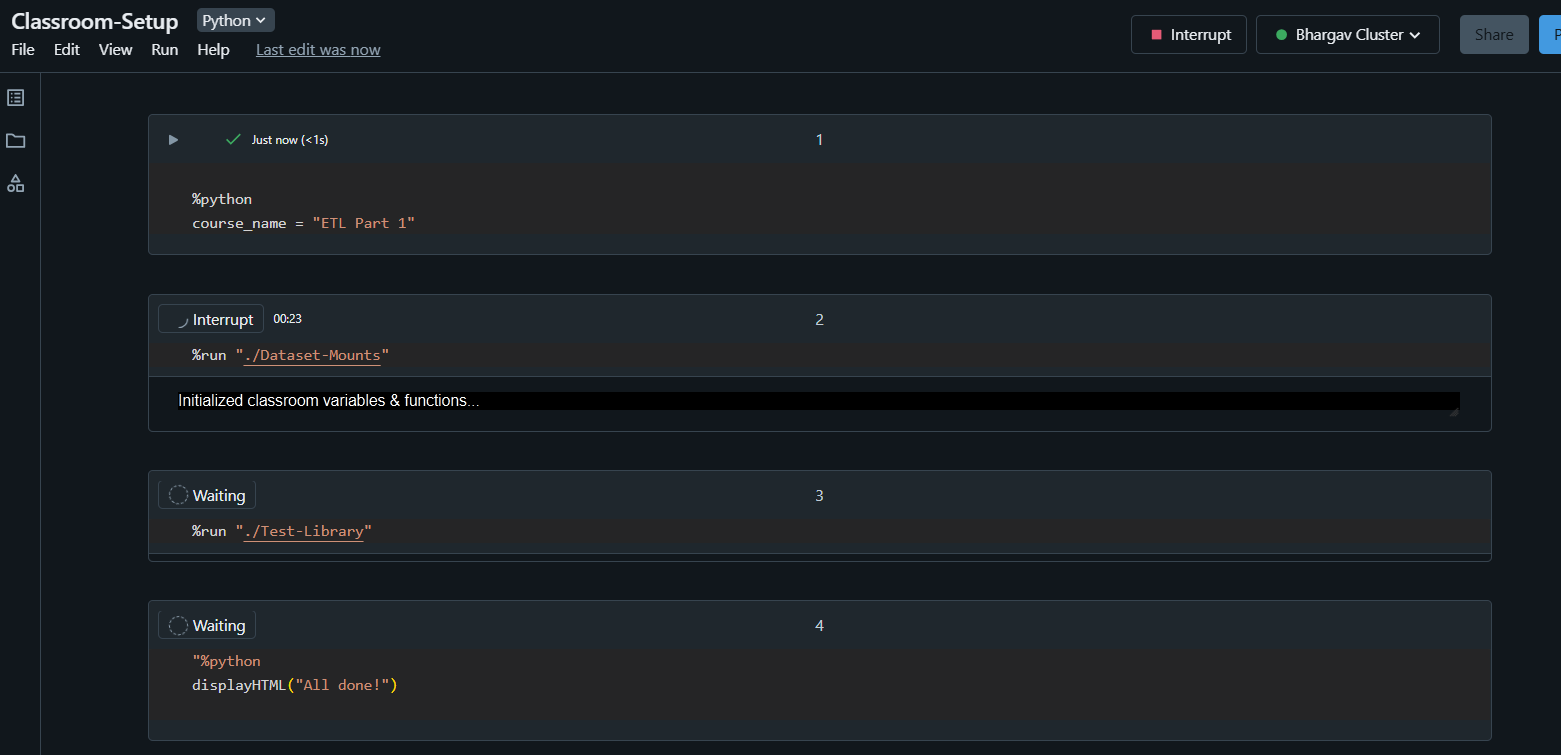
This project consists of two parts,

* Part-1 Parsing Nested Data
* Part –2 Custom transformations, Aggregating and Loading

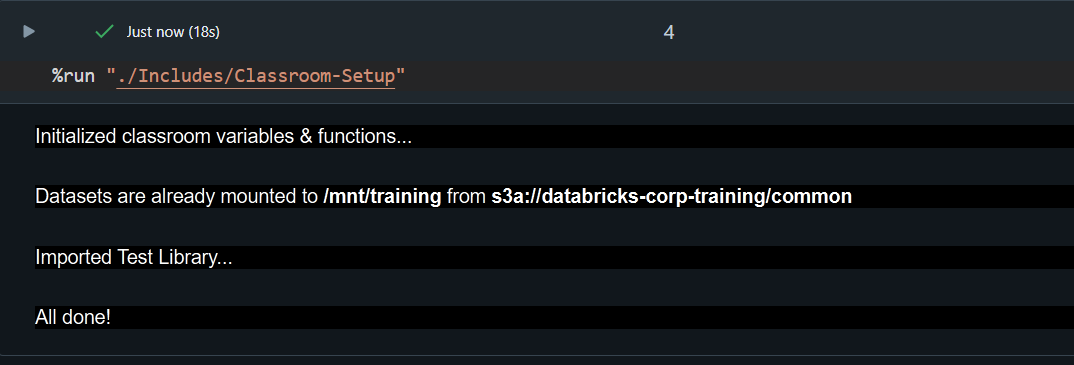
**PART –1 Parsing Nested Data**

In this part we are going to ingest JSON data using Databricks File System (DBFS), define and apply a schema, parse the fields, and save the cleaned results back to DBFS.

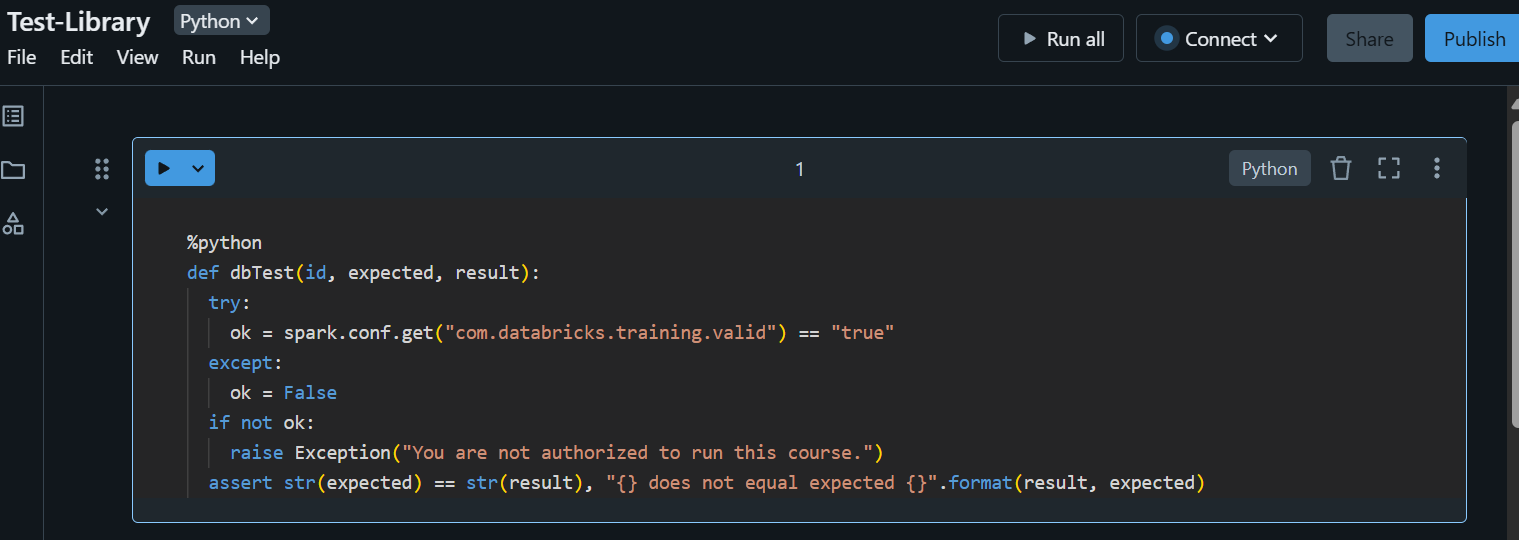
**Step 1: Run the cell of Classroom setup**



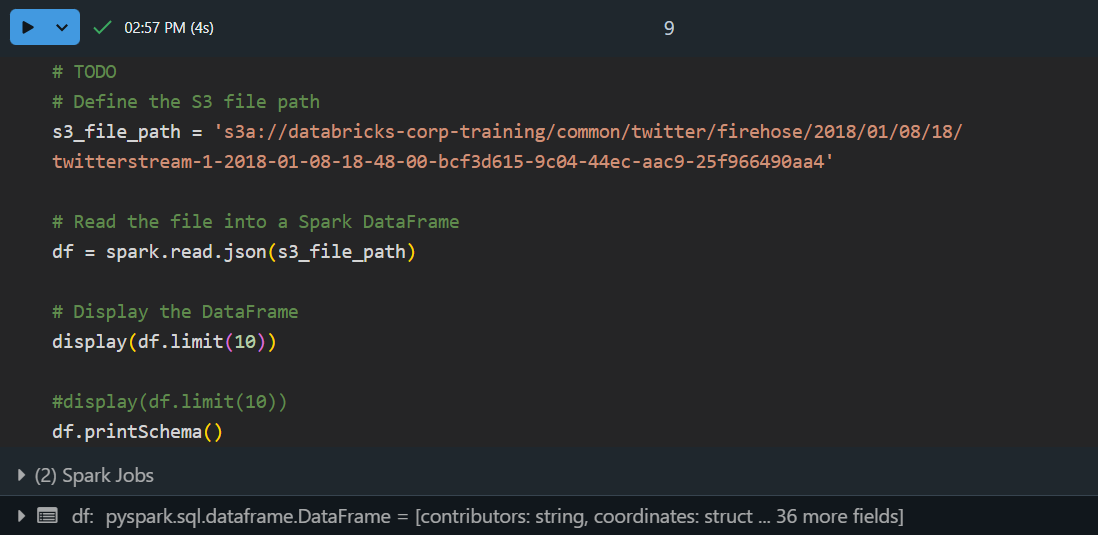
The %run ". /Includes/Classroom-Setup" command initializes the environment by configuring paths, importing dependencies, and preparing datasets f5or the notebook.



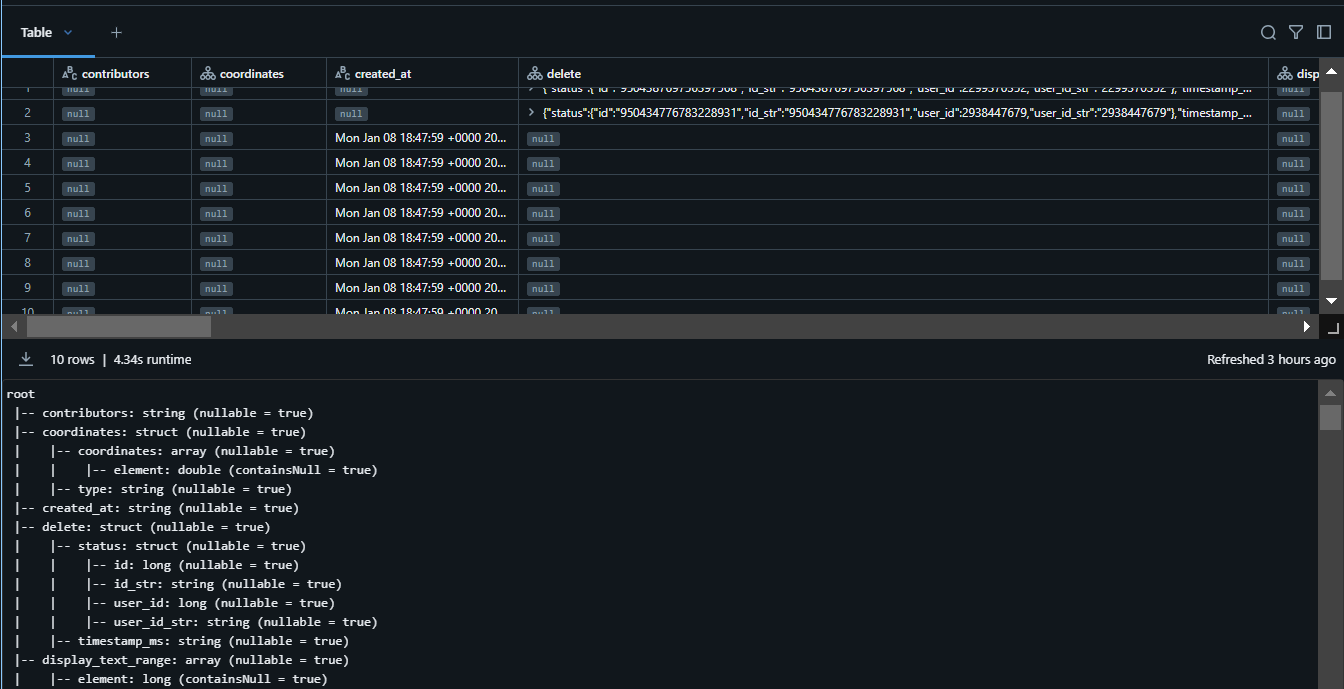
**User Defined Function for Testing**

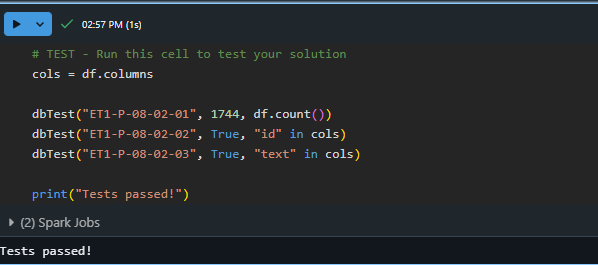


**Step 2: Extracting & Exploring data**



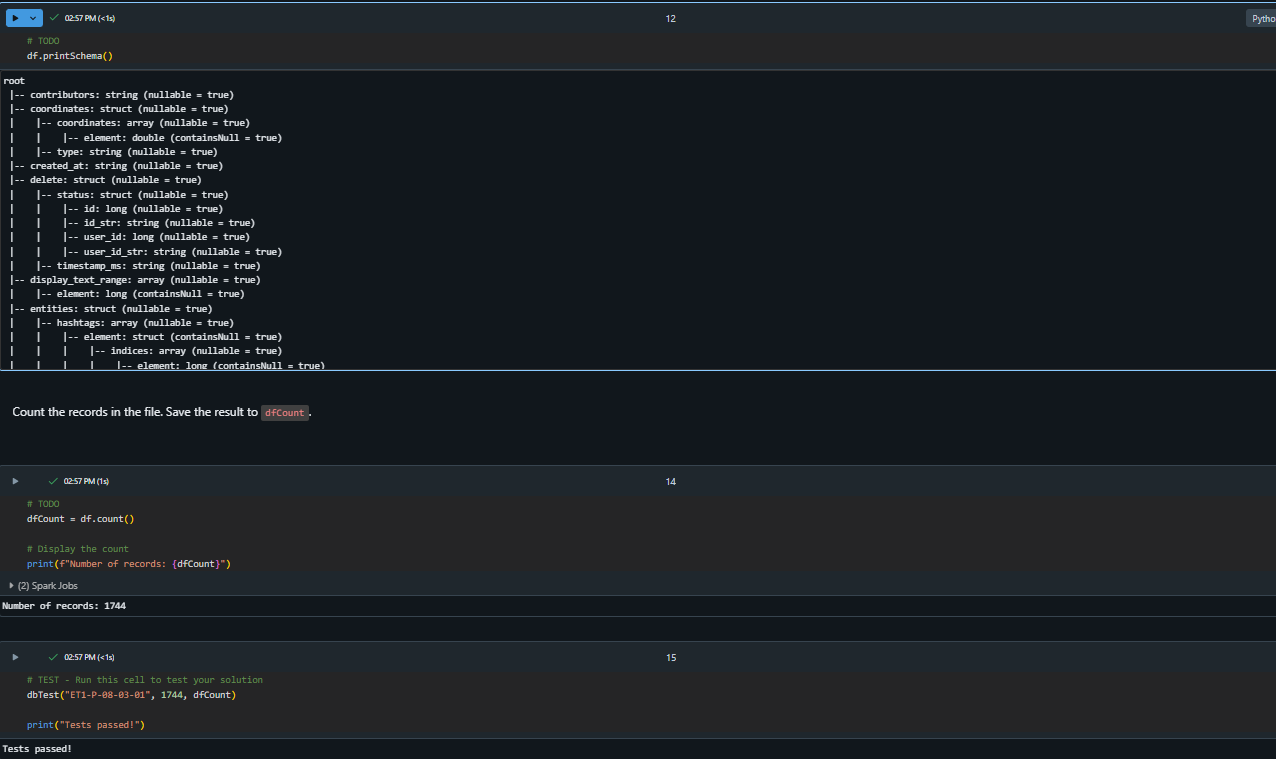
* **Defines the S3 file path** to locate the dataset.
* **Loads the JSON data** into a Spark DataFrame for processing.
* **Previews the first 10 rows** to check the data content.
* **Prints the schema** to understand the structure and format of the data for further transformations.



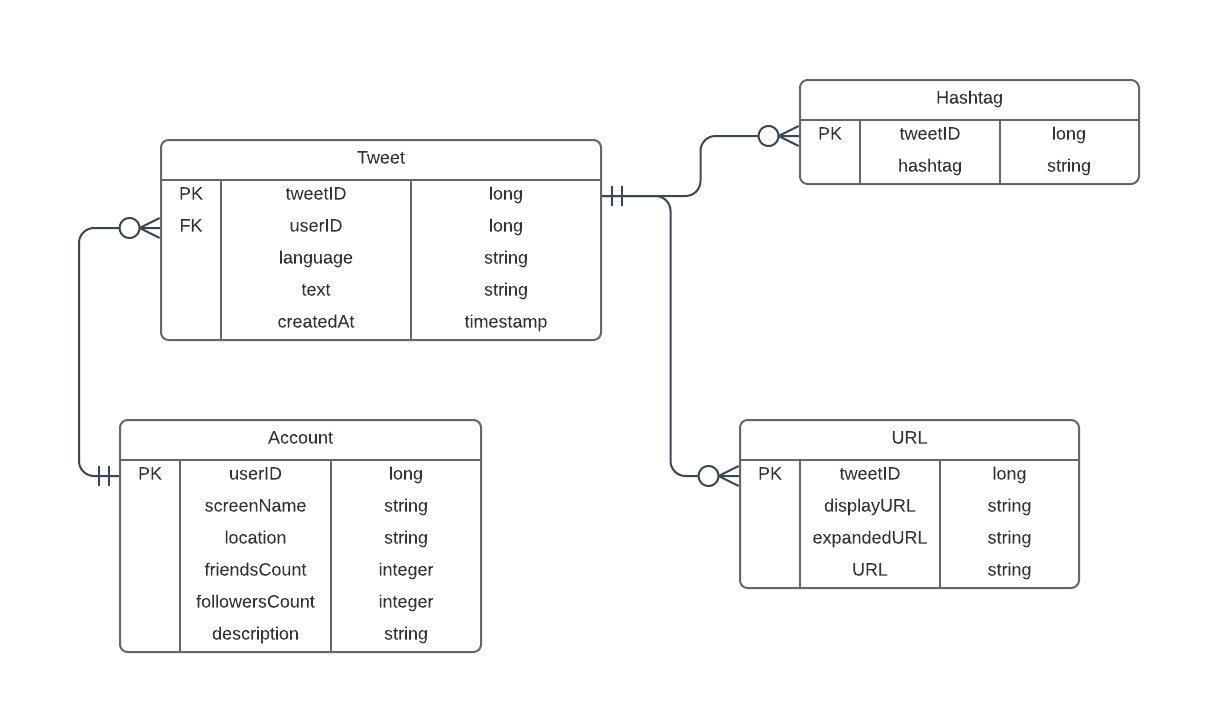


All test cases passed

* Then check the schema, check the count of the records and finally validate the test cases.



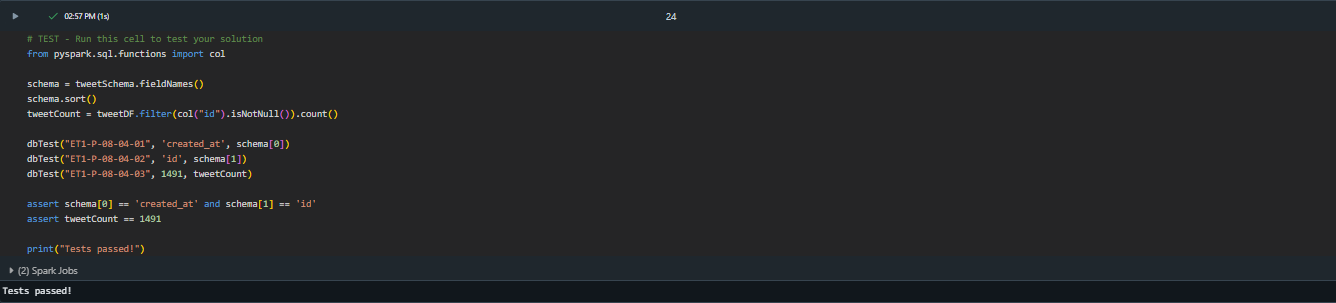
**Step 3: Defining and Applying a Schema**



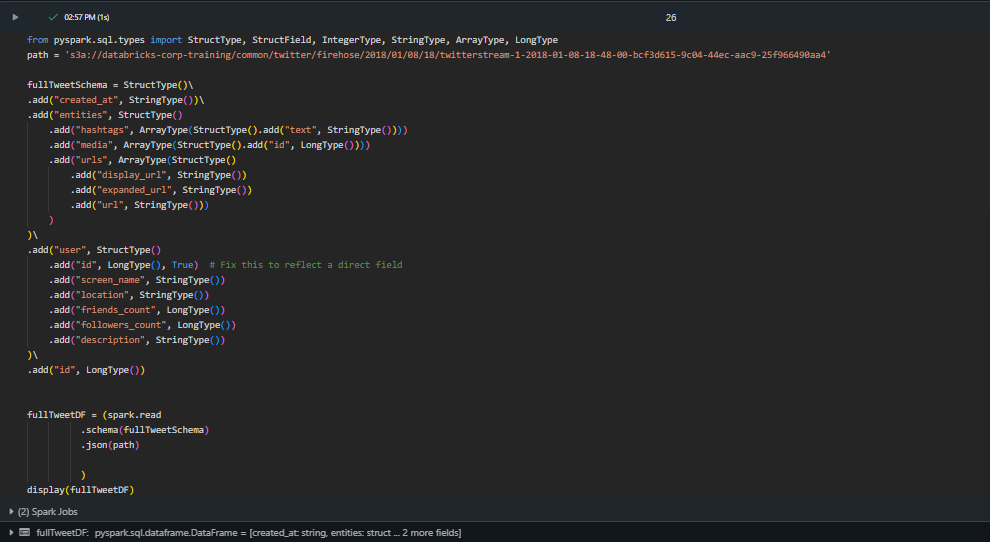
* Create Schema for tweet table

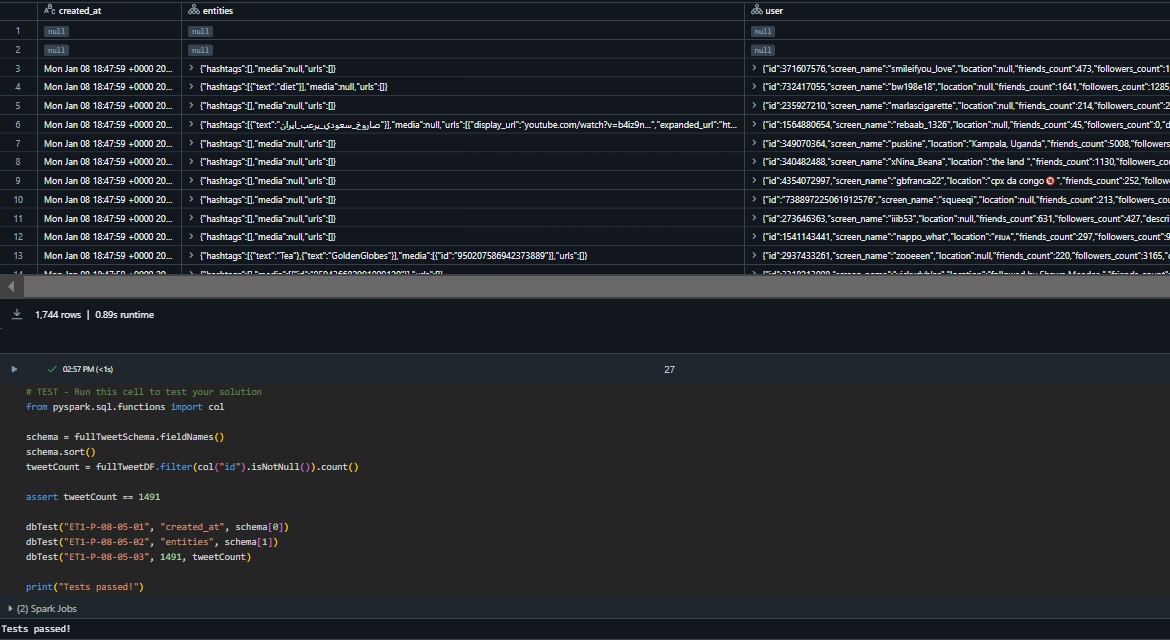




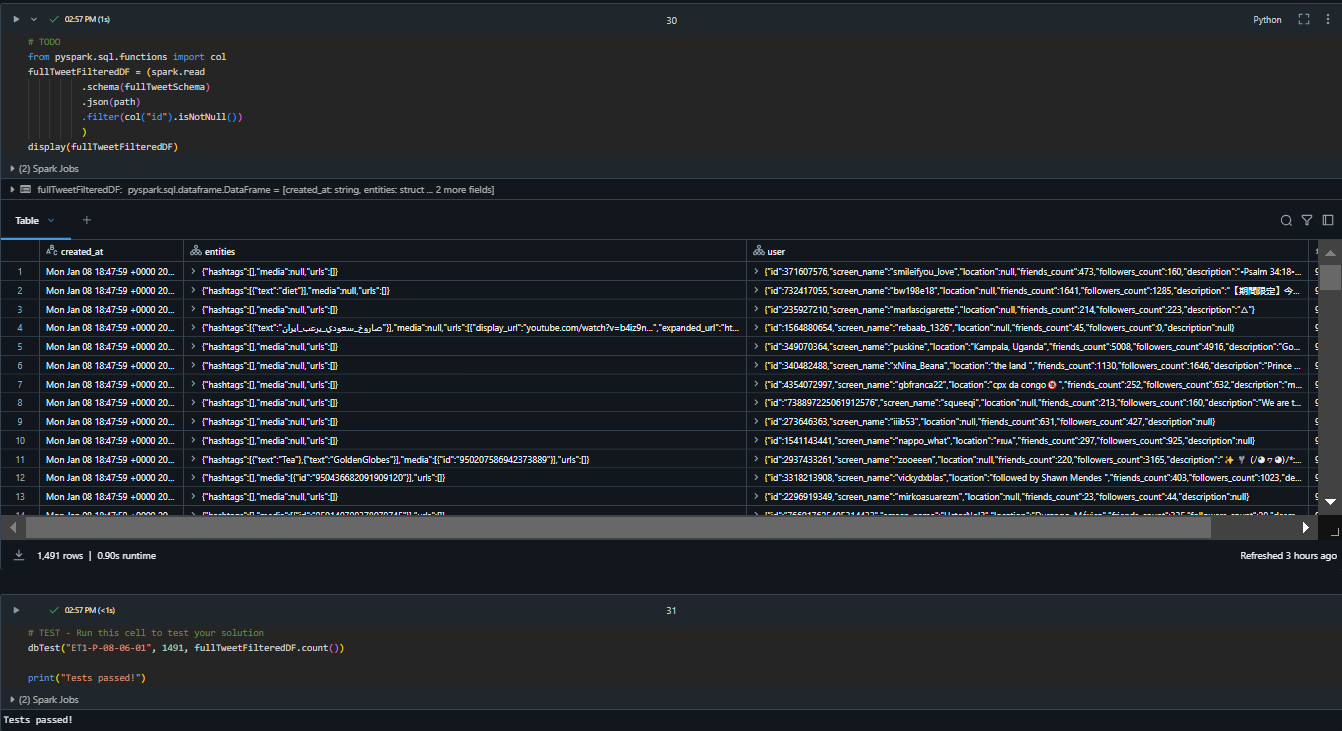
 Run the test cases for the tweet table

* Create the Schema for remaining tables

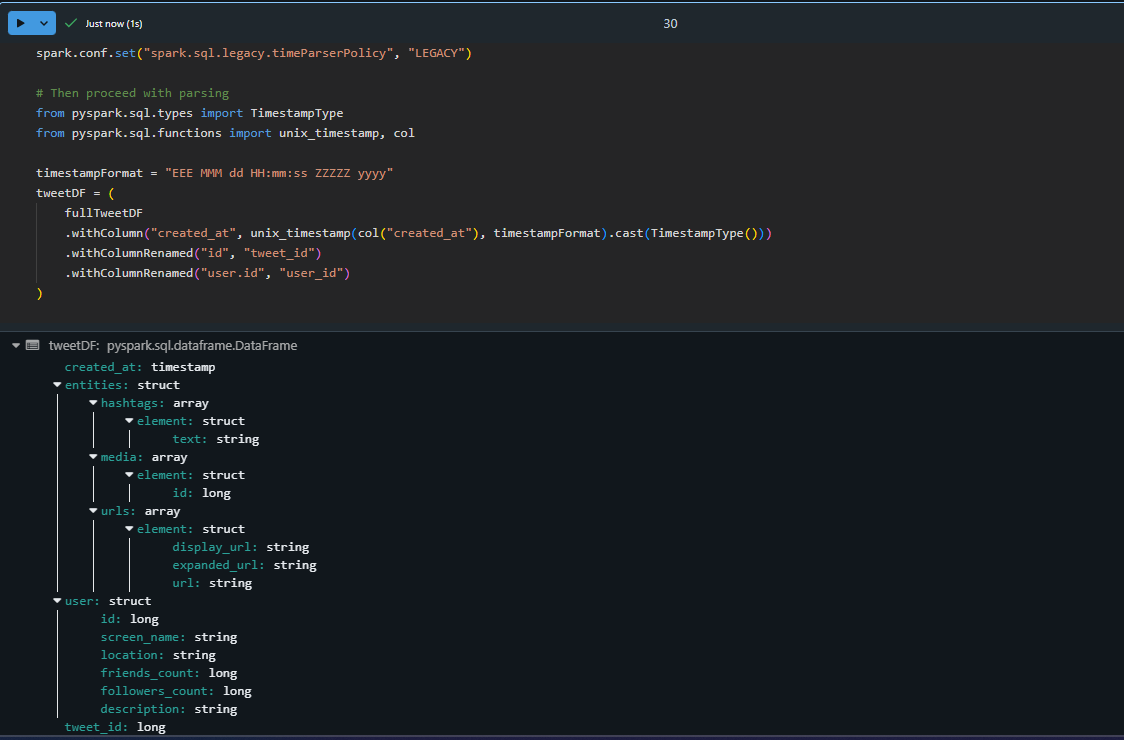


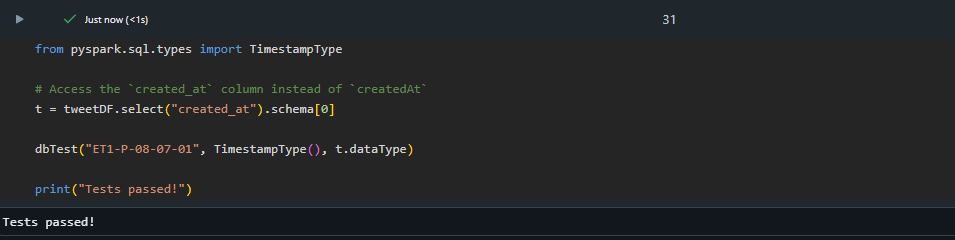


**Step 4: Creating the tables & Filtering Values**

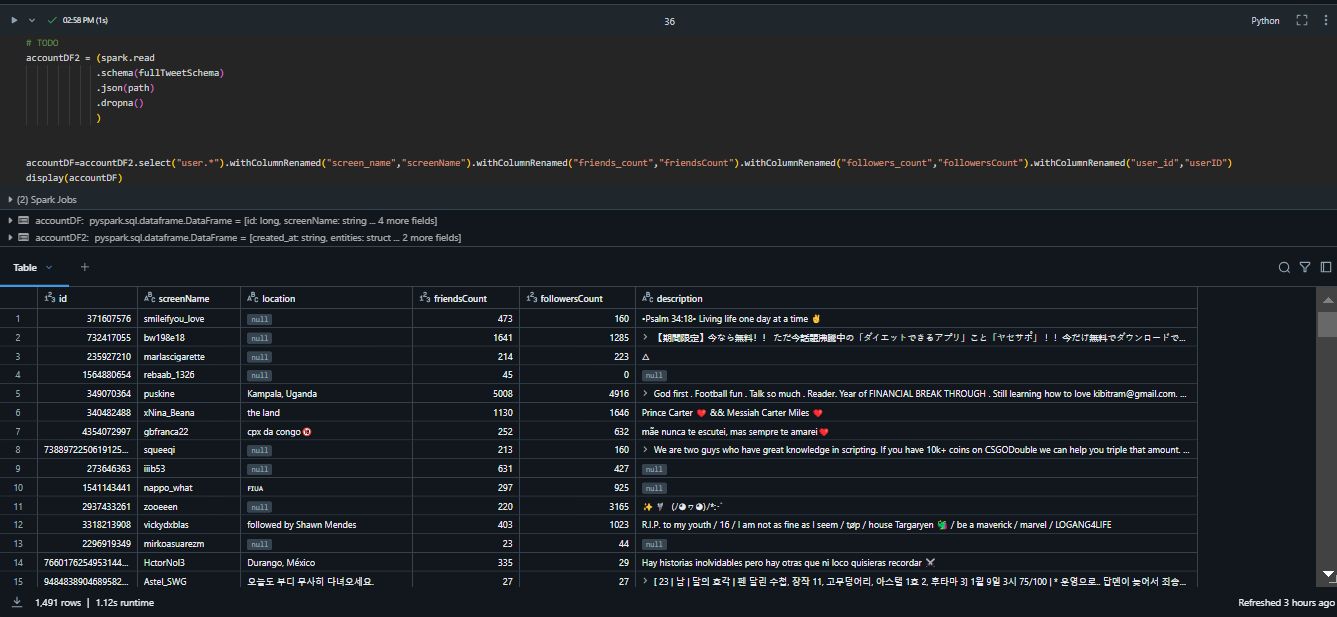
 Filtered the Null Values all test cases are passed

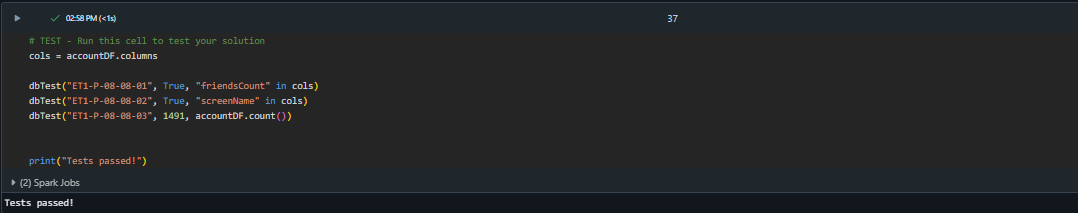
* Create the tweet table using timestamp datatype for created\_at



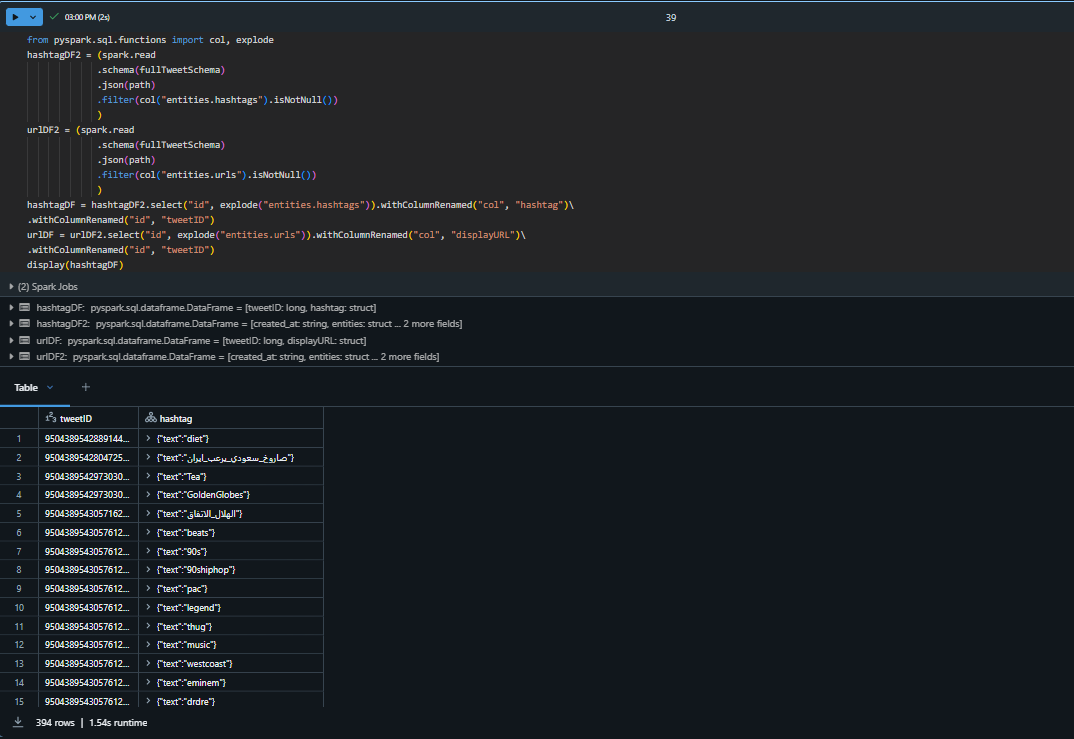
All test cases are passed and created \_at is created with timestamp datatype

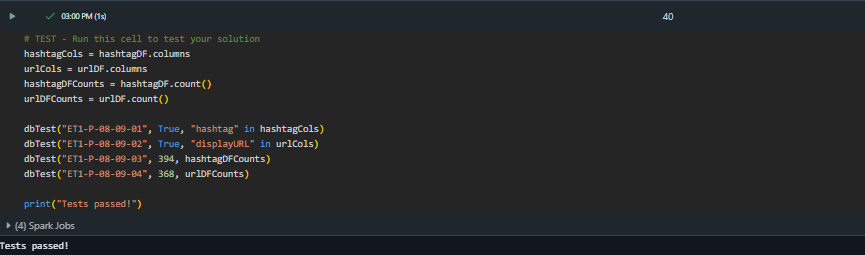
* Create the accounting table

-

 All test cases are passed without Null Values

* Create the tables for hashtags and URLS using Explode

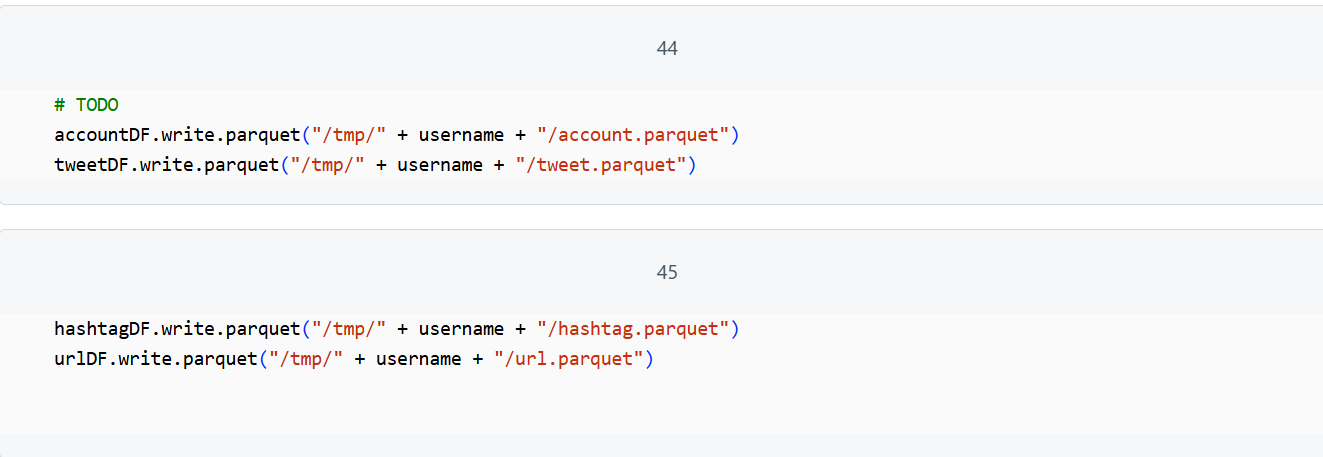


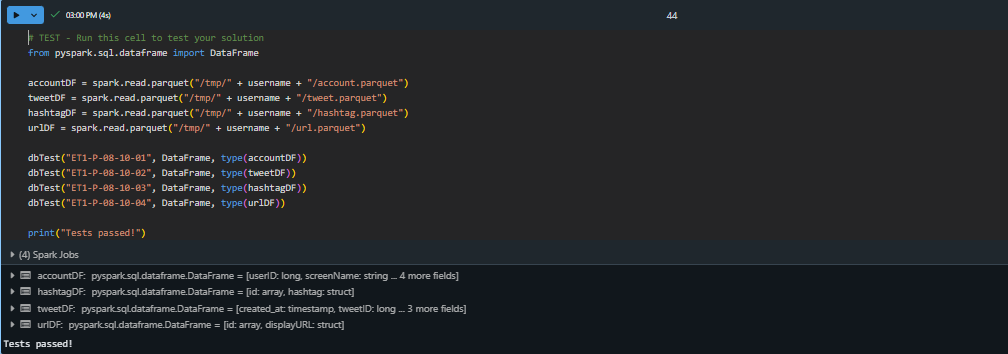


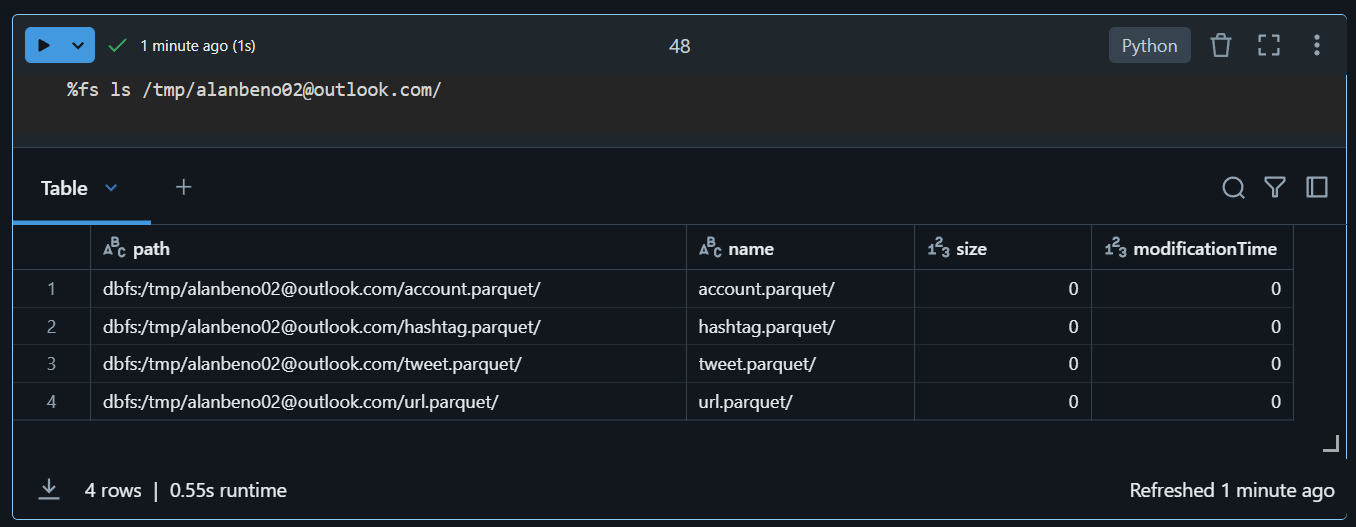
All test cases are passed if hashtags and URLS count is valid or not

**Step 5: Loading the data**

We are **loading the data** as the final step of the ETL (Extract, Transform, Load) process. The goal of loading and saving the transformed Data Frames is to persist the cleaned, well-structured data for future analysis, reporting, or sharing, while ensuring it’s organized, accessible, and performant.



All test cases are passed



Parquet files in dbfs

**PART –2 Custom Transformations, Aggregating & Loading**

The goal of this part is to populate aggregate tables using Twitter data. In the process, we write custom User Defined Functions (UDFs), aggregate daily most trafficked domains, join new records to a lookup table, and load to a target database.

**Step 1: Run the cell of Classroom setup**

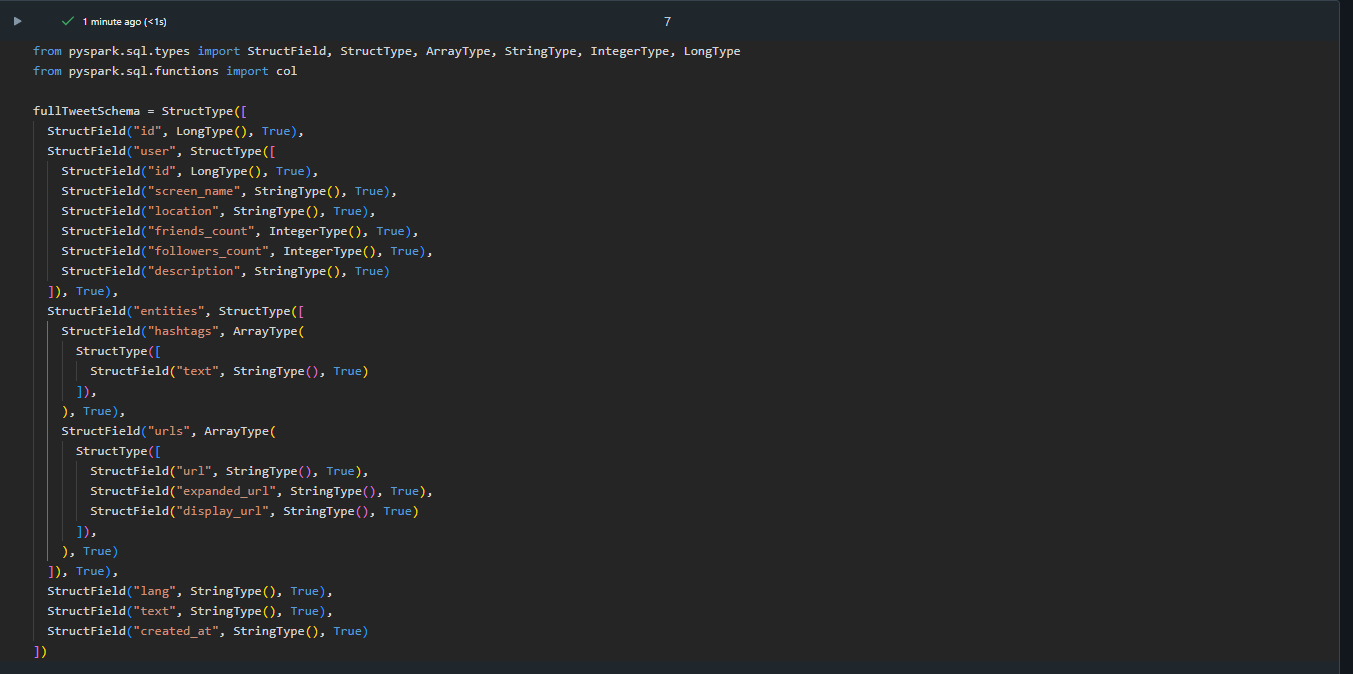
The %run ". /Includes/Classroom-Setup" command initializes the environment by configuring paths, importing dependencies, and preparing datasets for the notebook.



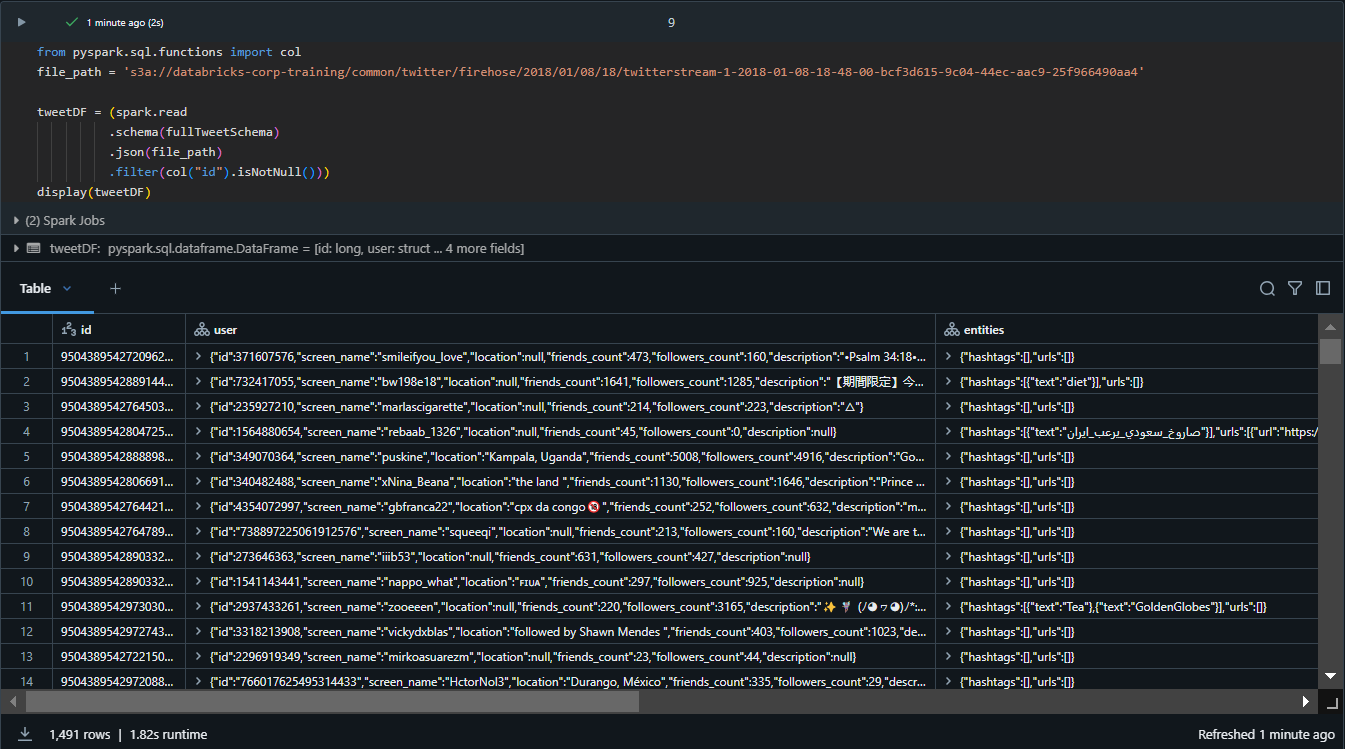
**Step 2: Parse tweeted URLs**

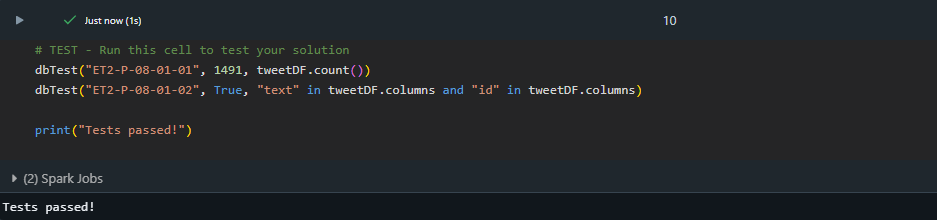
**IMPORT AND EXPLORE**

* Create the Schema that we have done in ETL part –1 of all tables.



* Import the data located at the S3 account.



 Run the testcases

* **WRITE A UDF TO PARSE URLS**

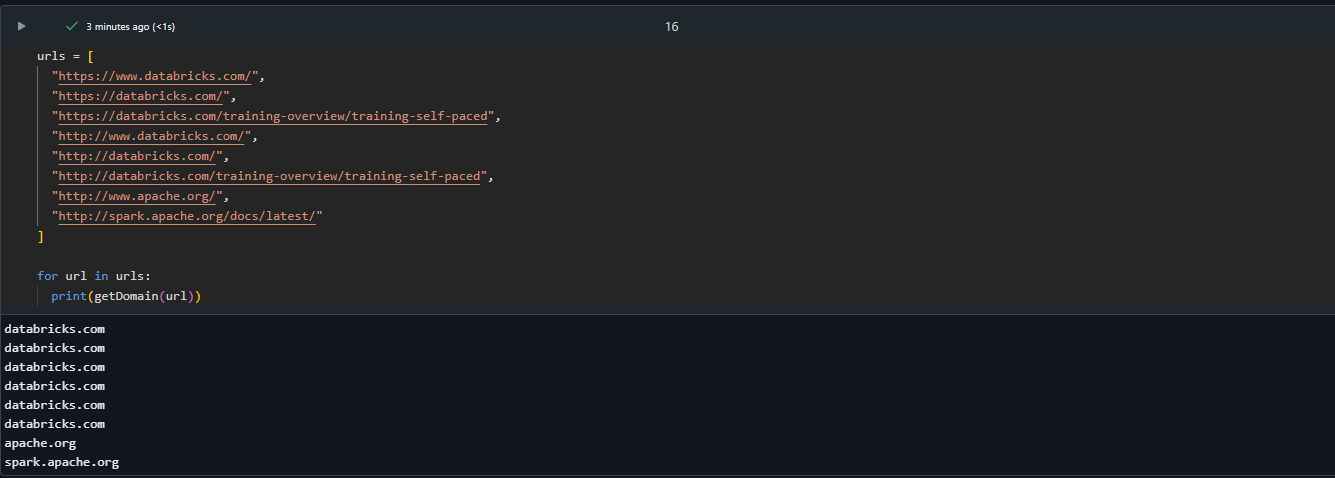
Writing a UDF to parse URLs helps automate the extraction of specific components (like domain names) from URLs for efficient data analysis and processing.

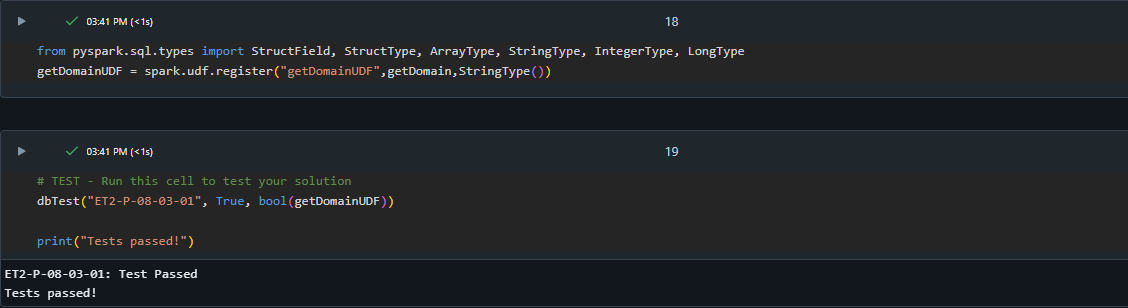


UDFs are written and all testcases are passed

* **TEST & REGISTER THE UDF**

Now that the function works with a single URL, confirm that it works on different URL formats.

 Working of Different URLs

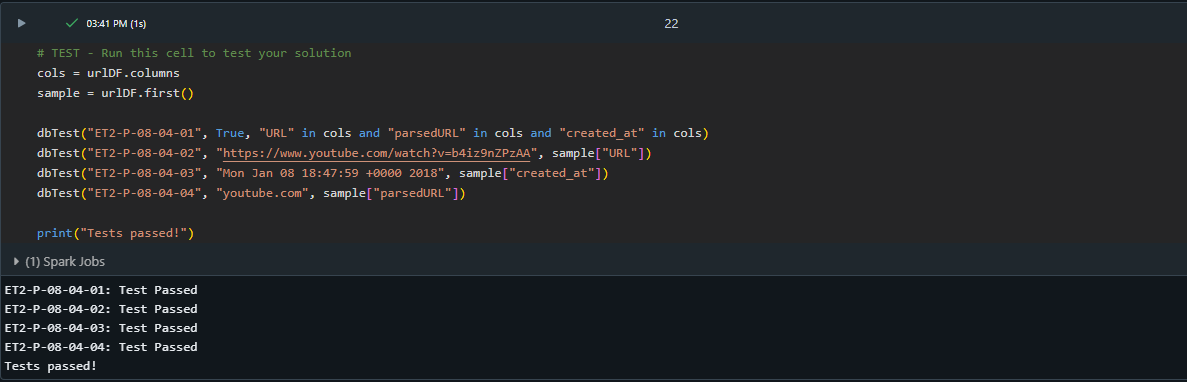
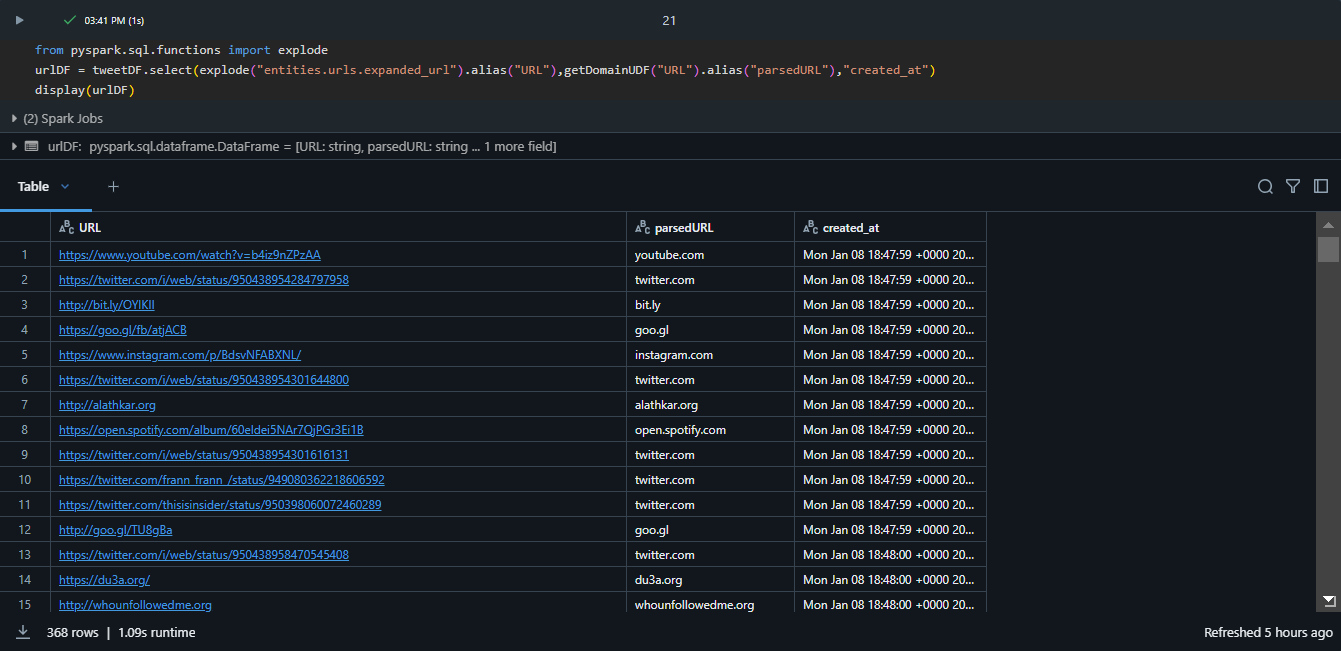
 Registering the UDF as getDomainUDF & test cases are passed

* **APPLY THE UDF**

We apply a UDF to perform custom data transformations, like extracting specific components (e.g., domain names) from a dataset efficiently and consistently.

Here we created a dataframe called **urlDF** that has three columns:

* URL: The URL's from tweetDF (located in entities.urls.expanded\_url)
* parsedURL: The UDF applied to the column URL
* Created\_at

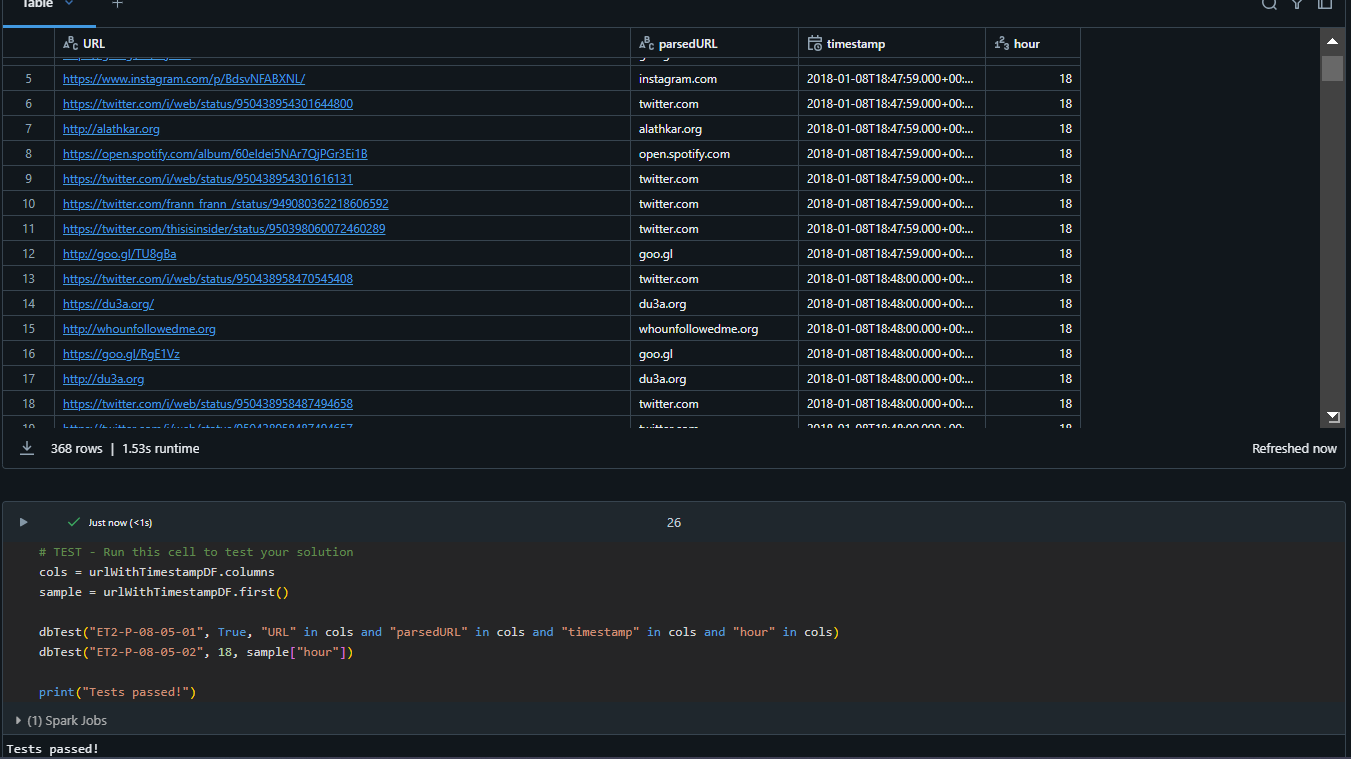
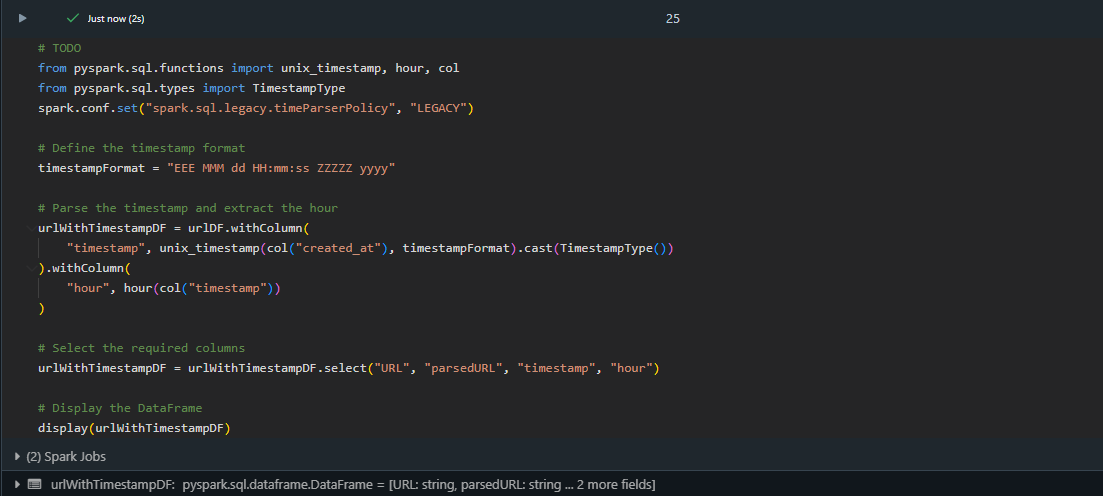
Test cases passed

**Step 4: COMPUTE AGGREGATE STATISTICS**

* **PARSE THE TIMESTAMP**

Create a dataframe urlWithTimestampDF that includes the following columns:

* URL
* parsedURL
* timestamp
* Hour

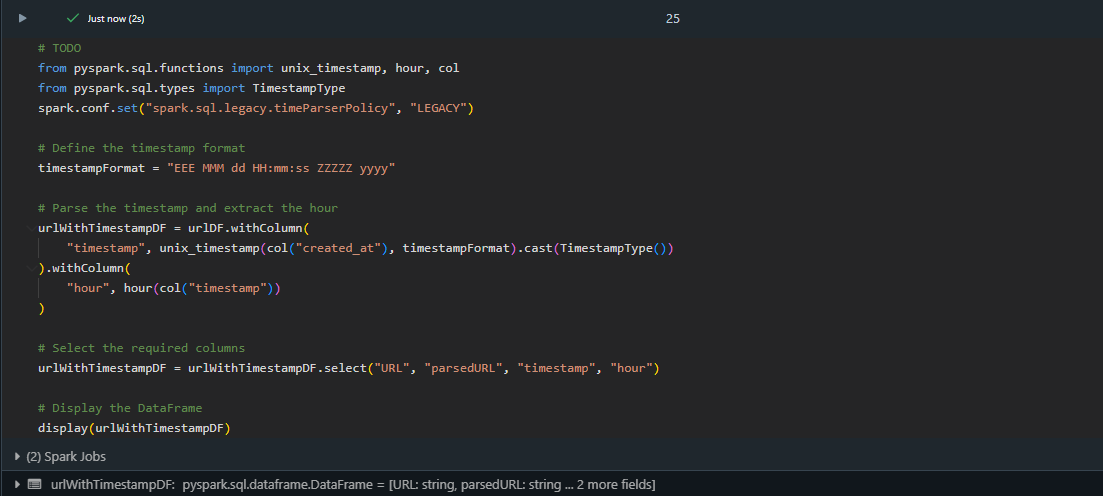
Output & All test cases are passed

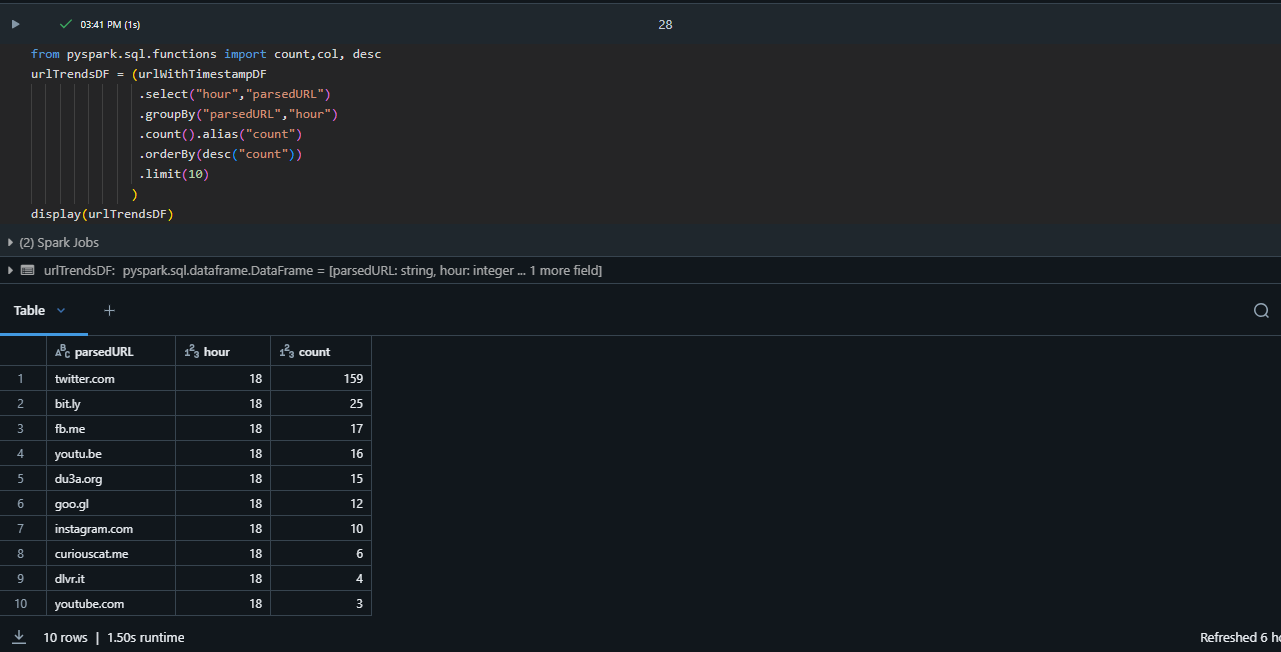
* **CALCULATE TRENDING URLS**
* Here We are calculating top trending 10 URLs by hour.
* Create a dataframe urlTrendDF that looks at the top 10 hourly counts of domain names includes the following columns:

- hour

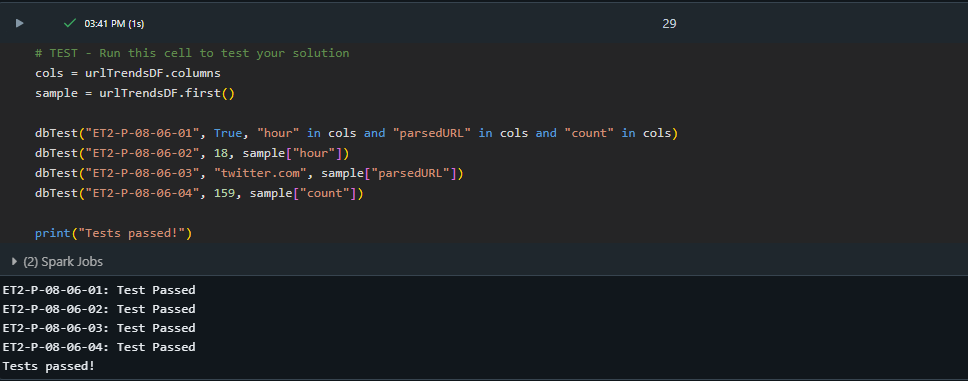
- parsedURL

- count



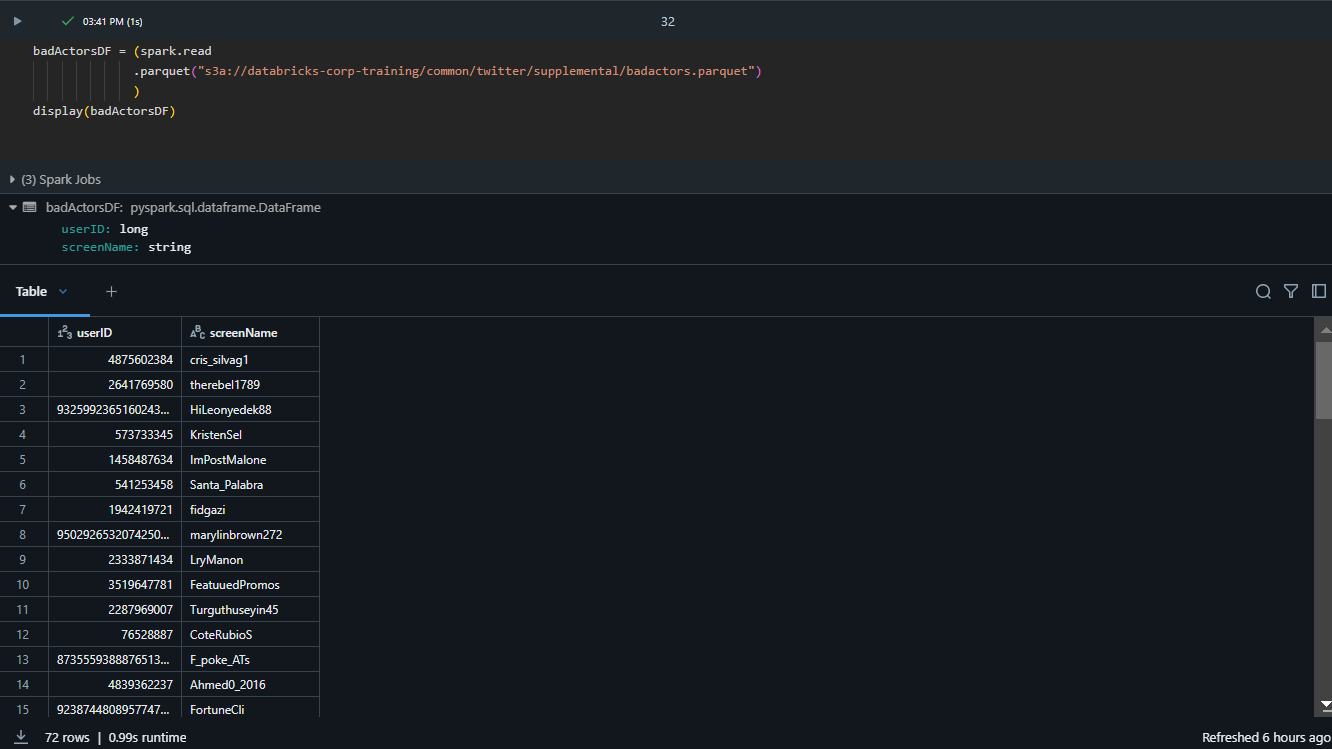


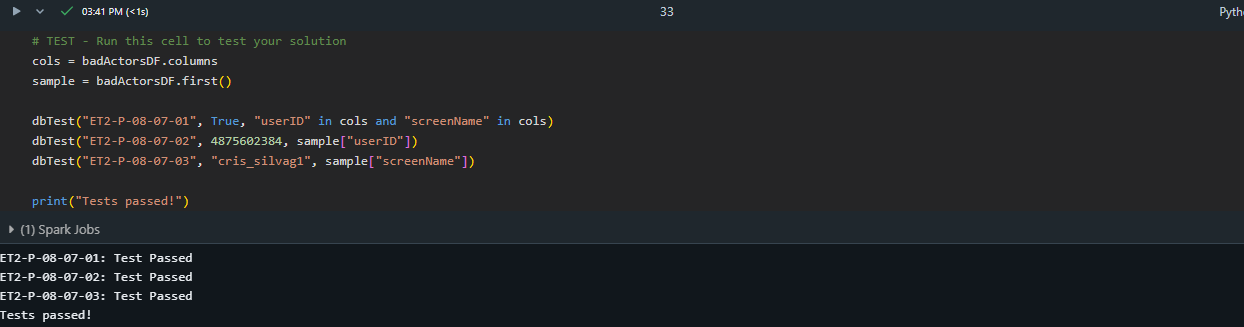
* Here the result should sort **hour** in ascending order & **count** is in descending order.

All test cases passed

**Step 5: JOIN NEW DATA**

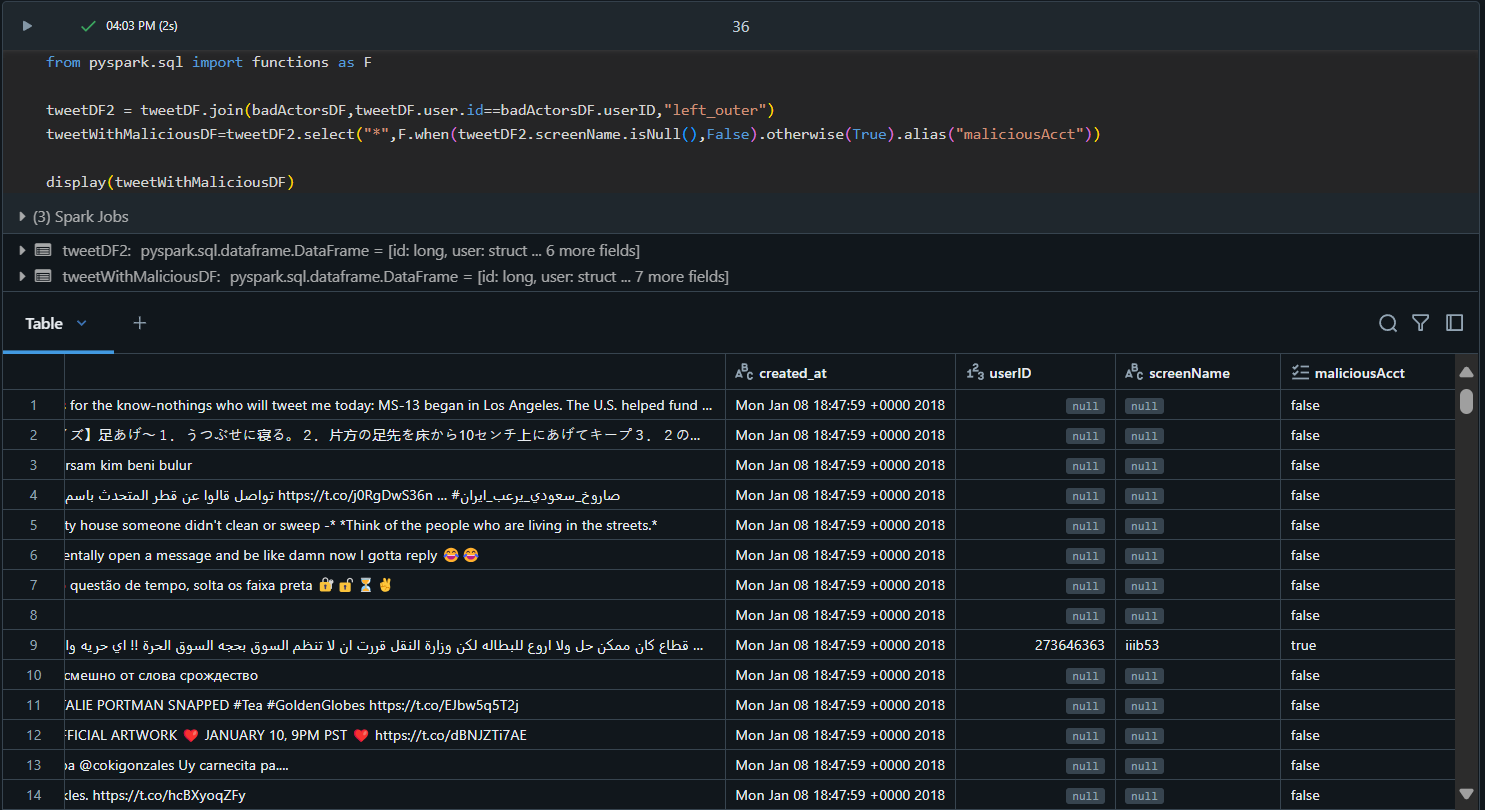
* By joining new data, we can filter out bad users.
* **IMPORT TABLE FOR BAD ACTORS**

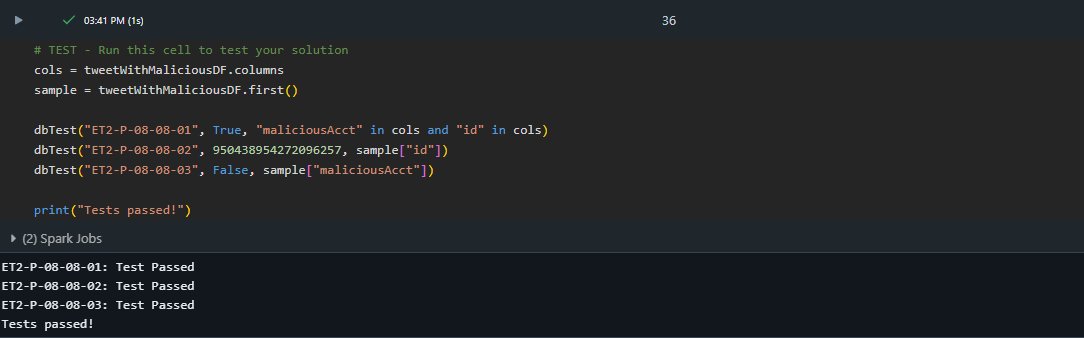


All test cases passed

* **ADD A COLUMN FOR BAD ACTORS**

Add a new column to tweetDF called maliciousAcct with true if the user is in badActorsDF. Save the results to tweetWithMaliciousDF. Remember to do a left join of the malicious accounts on tweetDF.

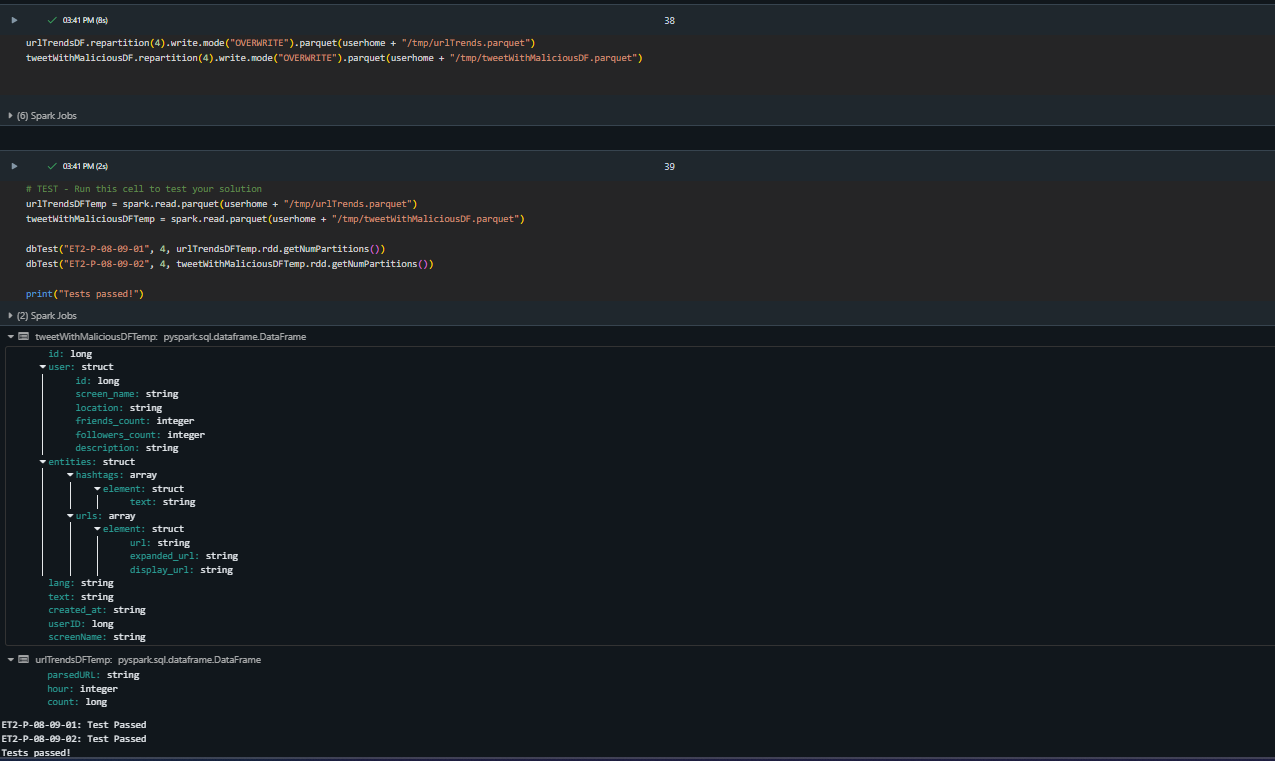


All test cases passed

**STEP: 8 LOAD RECORDS**

Transform your two Data Frames to 4 partitions and save the results to the following endpoints:

|  |  |
| --- | --- |
| **DataFrame** | **Endpoint** |
| urlTrendsDF | userhome + /tmp/urlTrends.parquet |
| tweetWithMaliciousDF | userhome + /tmp/tweetWithMaliciousDF.parquet |



## **CONCLUSION**

The ETL capstone projects (Parts 1 and 2) guide us through the complete data pipeline process. In **Part 1**, the focus is on extracting raw JSON Twitter data, applying schemas to structure it, and saving the cleaned output for further use. **Part 2** advances this by using custom transformations, aggregating data like trending hashtags and domains, joining with historical data (e.g., malicious users), and loading results into a database. Together, these projects build a strong foundation in handling raw data, transforming it into actionable insights, and creating ready-to-use datasets for analytics and decision-making. They comprehensively enhance real-world data engineering skills.