**GLOBAL BANKING ANALYSIS**

**Problem Statement**

Global Bank is undergoing a digital transformation to enhance its fraud detection capabilities, improve customer segmentation, and optimize its reporting systems. As a data engineer, your task is to design and implement a robust data pipeline that can handle real-time transaction data from multiple sources, process it efficiently, and prepare it for advanced analytics and reporting.

Your specific objectives are:

Design and implement a scalable data ingestion system that can handle 100 to 500 transactions per second from 10 to 50 global branches and digital channels.

Implement a Delta Lake architecture with Bronze, Silver, and Gold layers:

Bronze: Raw ingestion of streaming transaction data

Silver: Cleansed and enriched data, joining transactions with customer and branch information

Gold: Aggregated data for reporting, fraud detection results, and customer segments

Create a data processing pipeline that cleanses, transforms, and enriches the transaction data with customer and branch information.

Implement a real-time fraud detection system based on rule-based flagging and transaction pattern analysis.

Develop a customer segmentation system based on transaction behavior and account information.

Create an automated reporting system that generates daily, weekly, and monthly summaries of transaction volumes, potential fraud cases, and customer segment distributions.

Ensure data quality, implement schema evolution capabilities, and manage data retention policies across all layers.

Optimize Delta Lake tables for both streaming writes and batch reads to enhance overall system performance.

Implement conceptual multi-tenant isolation within the same Databricks workspace to support different teams or departments.

Optimize the data storage and retrieval process to support quick analytics queries and real-time dashboards.

Create a real-time dashboard showing transaction volumes across different channels

Visualize the geographical distribution of transactions and potential fraud cases

Build interactive charts for customer segment analysis

Design a time series visualization for transaction patterns and anomalies

**Challenges and Limitations**

During this project, several challenges and limitations were encountered due to the use of the Databricks community edition, primarily resulting from restrictions imposed by Nuvepro Labs.

* **Nuvepro Labs Restrictions:** Due to restrictions, I was unable to use Azure Data Factory (ADF) and faced login issues, necessitating the use of the community edition of Databricks.
* **Lack of Automation:** The community edition does not support automated pipelines or job clusters, preventing workflow automation and requiring manual data management.
* **Resource Constraints:** Limited resources in the community edition led to out-of-memory issues during aggregation processes in the Gold layer, impacting the efficiency of handling large datasets.
* **Pipeline Automation:** Due to restrictions on creating workflows, pipeline automation was not feasible, resulting in a more manual and time-consuming process.

Despite these challenges, the project successfully implemented an end-to-end streaming process using the Medallion architecture in Databricks and connected to Power BI for real-time business insights.

**Approach**:

I have generated the data for customers and branches using batch processing and for transactions using streaming data generation code. This data is stored in the Hive metastore under the schema raw\_data.

We have implemented the Medallion architecture, comprising Bronze, Silver, and Gold layers:

* **Bronze Layer:** This layer involves ingesting raw data from the source, capturing it in its original form for initial storage and historical reference.
* **Silver Layer:** In this layer, the raw data undergoes cleansing and transformation. We apply various data quality checks and enrichment processes to ensure the data is accurate, consistent, and reliable.
* **Gold Layer:** This final layer contains business-ready, aggregated data. It is optimized for analytics and reporting, providing a comprehensive view for business insights.

We connect Databricks directly to Power BI to visualize and present these business insights effectively. This integration allows us to create interactive dashboards and reports, offering a dynamic and real-time view of key performance indicators and trends.

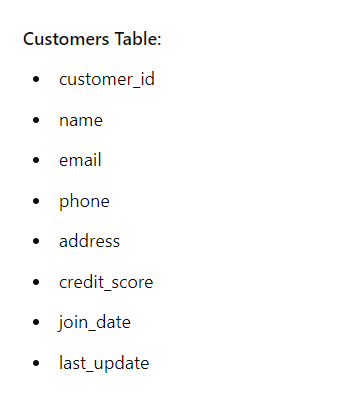
Additionally, we have designed and implemented an end-to-end streaming process for real-time data handling. This approach ensures continuous data ingestion, processing, and analysis, enabling immediate insights and timely decision-making.

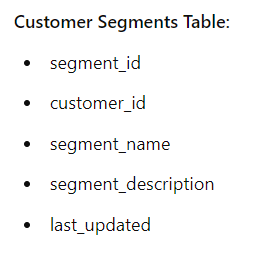
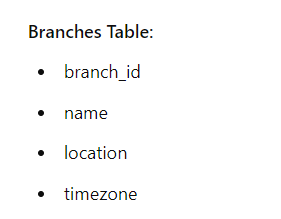
**Architecural Diagram**

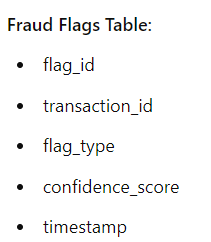


* **Batch Data (Customers and Branches):** Generated using batch processing and stored in the Hive metastore under the raw\_data schema.
* **Streaming Data (Transactions):** Generated using streaming techniques and also stored under the raw\_data schema.
* **Medallion Architecture Implementation:**
  + **Bronze Layer:** Ingests and stores raw data.
  + **Silver Layer:** Cleanses and transforms the data.
  + **Gold Layer:** Aggregates the data for business readiness.
* **Power BI Integration:** Connects to the Gold layer to provide real-time business insights through interactive dashboards and reports.

**Tables, Schema**





**Medallion architecture**

**Data Pipeline Description**

1. **Ingest Data to Bronze Layer:**

* Raw data from the Faker API is ingested into the Bronze layer through spark streaming.

1. **Cleansing and Transformation in Silver Layer:**

* Data is cleansed and transformed, moving from the Bronze layer to the Silver layer through both batch & spark streaming.

Cleansing Rules:

* Drops rows with any null values.
* Ensures columns are of the correct data type based on the schema.
* Removes duplicate rows.
* Trims whitespace and removes special characters from text columns.
* Handles outliers using the IQR method.
* Ensures data consistency and checks for non-null and non-NaN values**.**

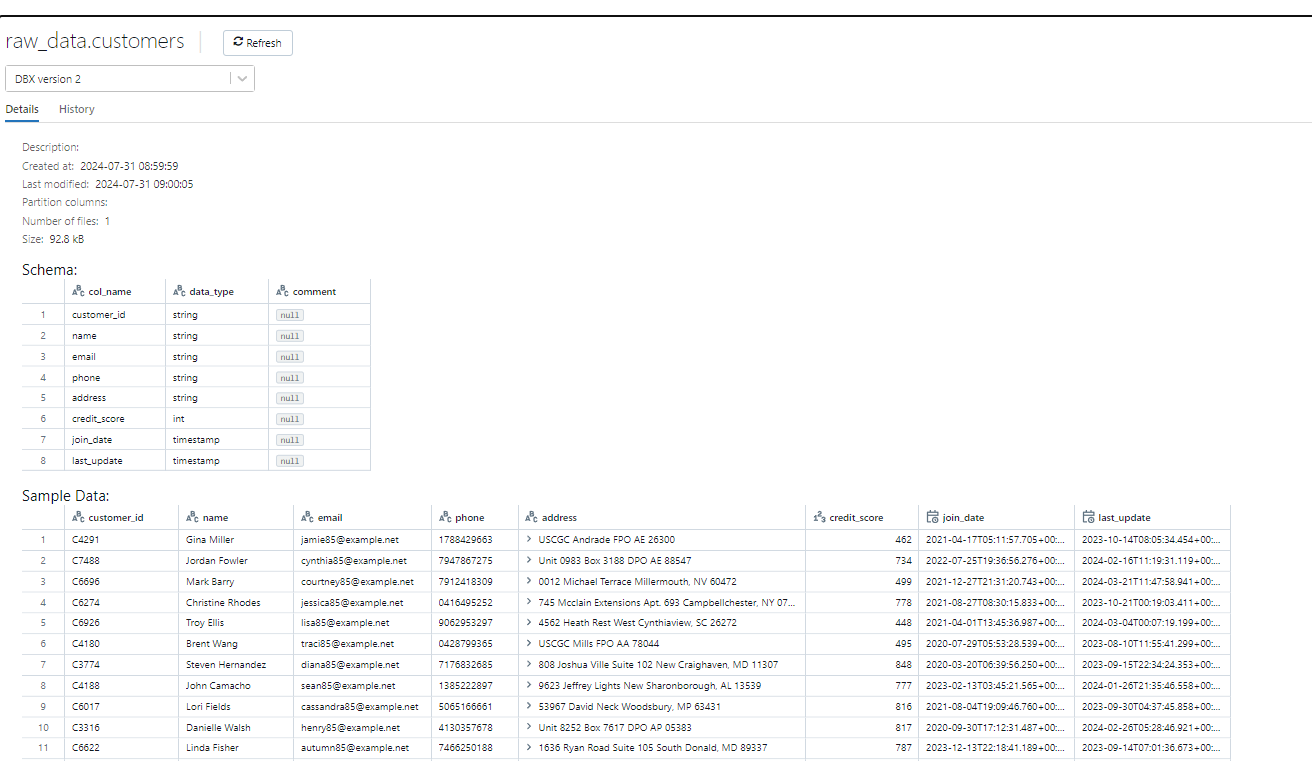
1. **Aggregation and Analysis in Gold Layer:**

* Data is aggregated and prepared for business-level use in the Gold layer through spark streaming.

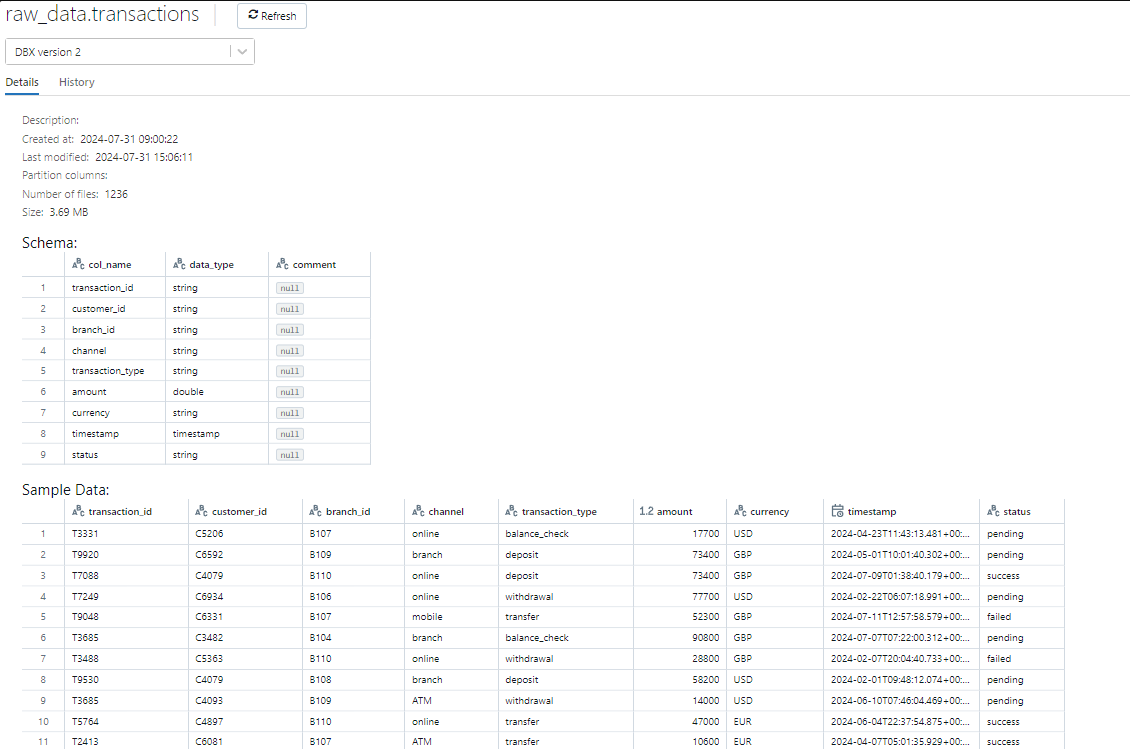
**Raw Data Generation and Ingestion**

1. **Setup:**
   * Installed the Faker library for generating realistic fake data.
   * Defined schemas for transactions, customers, and branches tables.
2. **Static Data Generation:**
   * Generated 1000 entries for customers and 10 entries for branches.
   * Converted to Spark DataFrames and saved as Delta tables in the Hive metastore (raw\_data schema).
3. **Streaming Data Simulation:**
   * Created a function to generate and append new transactions data every 5 seconds.
   * Continuously wrote the streaming transactions data to the raw\_data.transactions table.

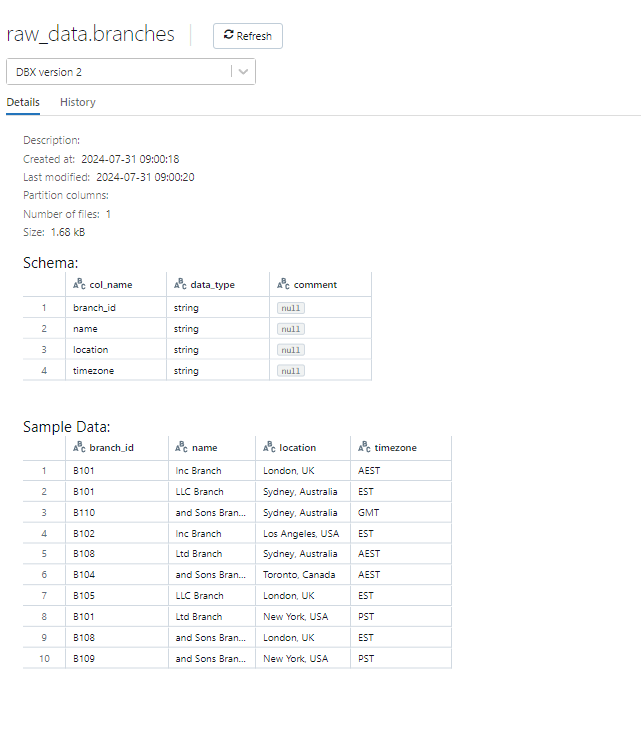
**Customer Table**

****

**Transaction Table**

****

**Branch Table**

****

**Ingest Raw data to Bronze layer**

### Ingesting Raw Data to Bronze Layer

#### Batch Data Processing

* **Function:** process\_and\_save\_batch\_data()
  + **Read:** Load data from raw\_data.customers and raw\_data.branches.
  + **Save:** Write data to Bronze layer as Delta tables Bronze\_layer.customers and Bronze\_layer.branches.
  + **Mode:** Overwrite.
  + **Outcome:** Batch data processed and loaded to Bronze layer.

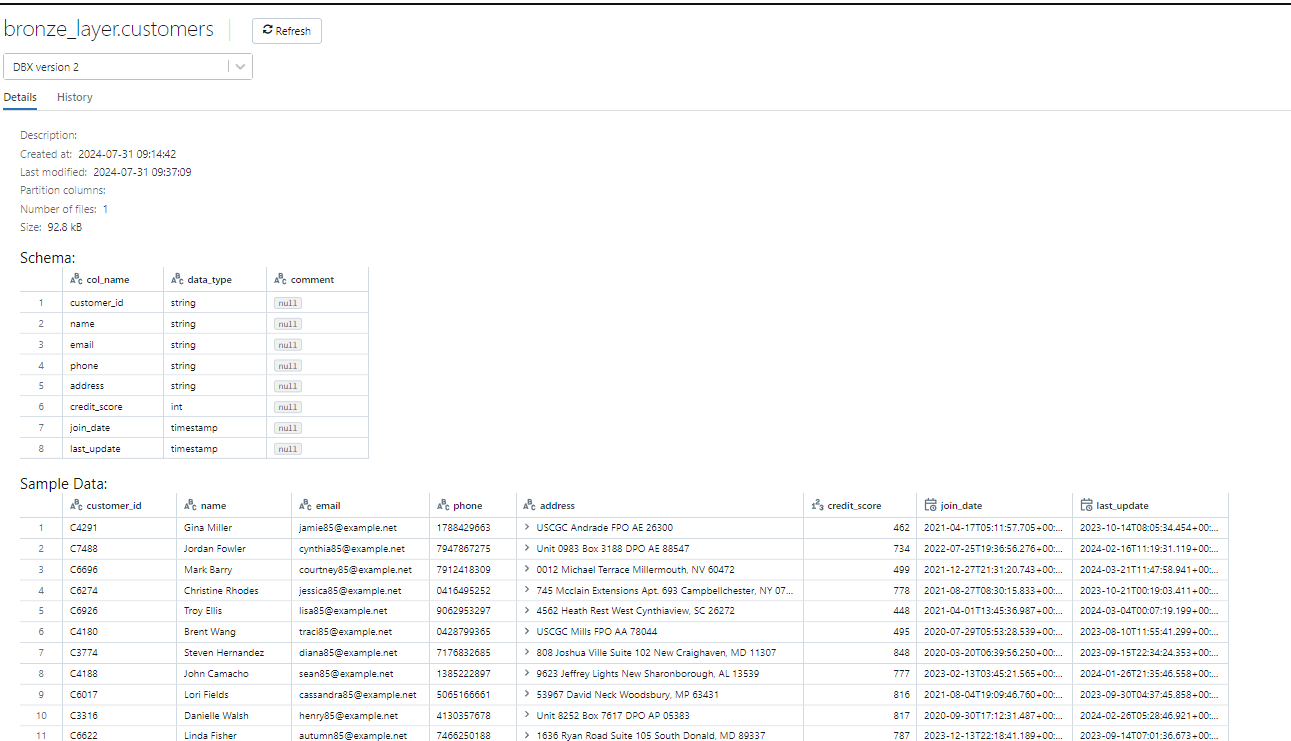
#### Streaming Data Processing

* **Function:** ingest\_transactions()
  + **Read:** Stream data from raw\_data.transactions.
  + **Write:** Save streaming data to Bronze layer Delta table Bronze\_layer.transactions.
  + **Checkpoint:** Bronze\_layer.checkpoint\_1.
  + **Outcome:** Streaming initiated for transaction data.

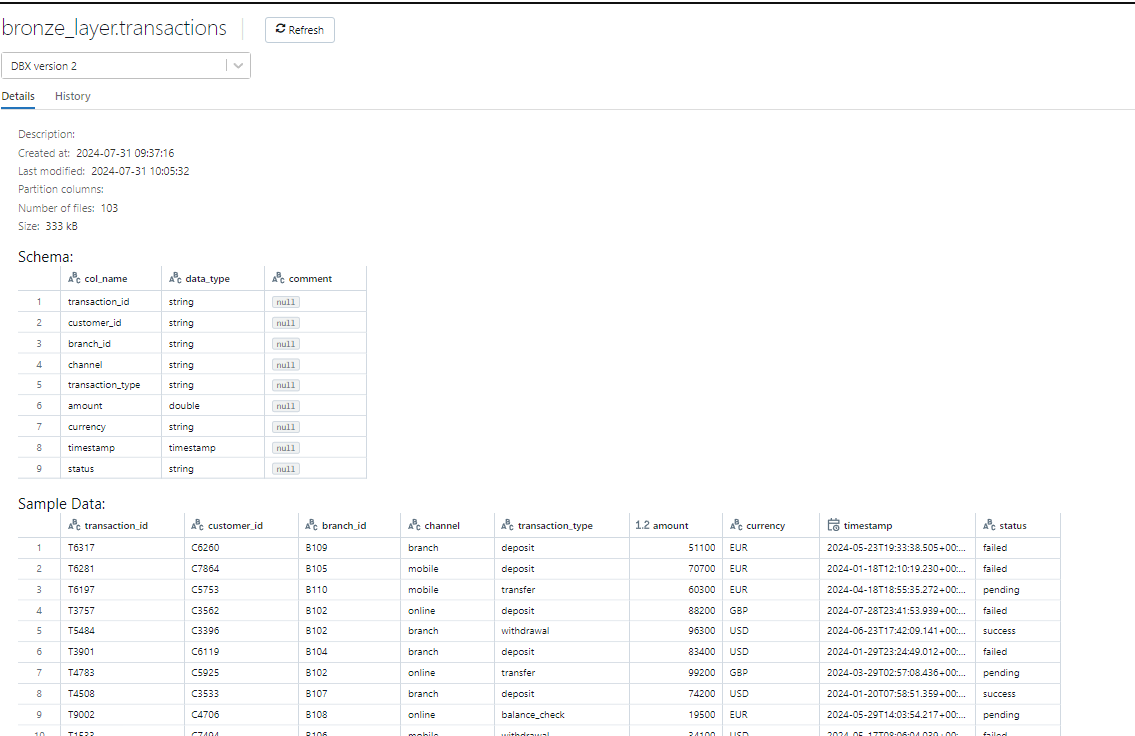
#### Execution

* **Batch Processing:** Run process\_and\_save\_batch\_data().
* **Streaming:** Run ingest\_transactions(), with handling for interruptions.

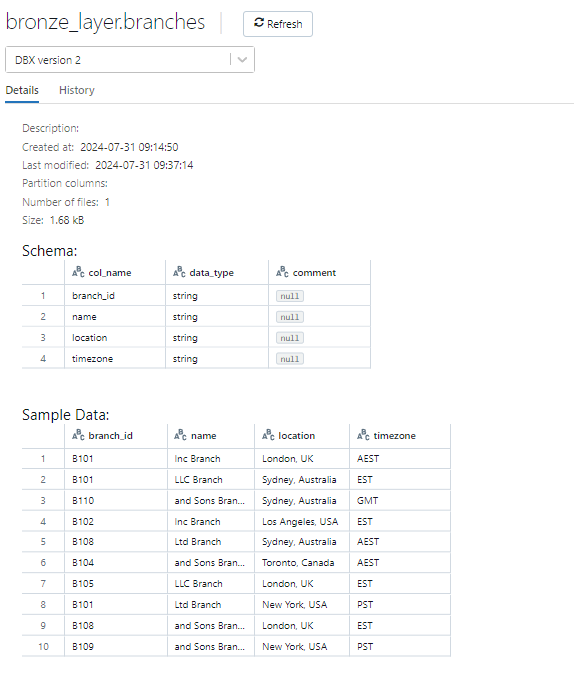
**Customer Table**

****

**Transaction table**

****

**Branch table**

****

**Cleansing and load to Silver Layer**

### Cleansing and Loading Data to Silver Layer

#### Data Cleansing Functions

* **Handle Missing Values:** Drop rows with null values.
* **Convert Data Types:** Cast columns to appropriate types based on schema.
* **Format Timestamps:** Standardize date formats.
* **Remove Duplicates:** Eliminate duplicate rows.
* **Clean Text Data:** Trim and sanitize text columns.
* **Handle Outliers:** Filter out extreme values in numerical columns.
* **Data Quality Checks:** Ensure no nulls or NaNs in critical columns.

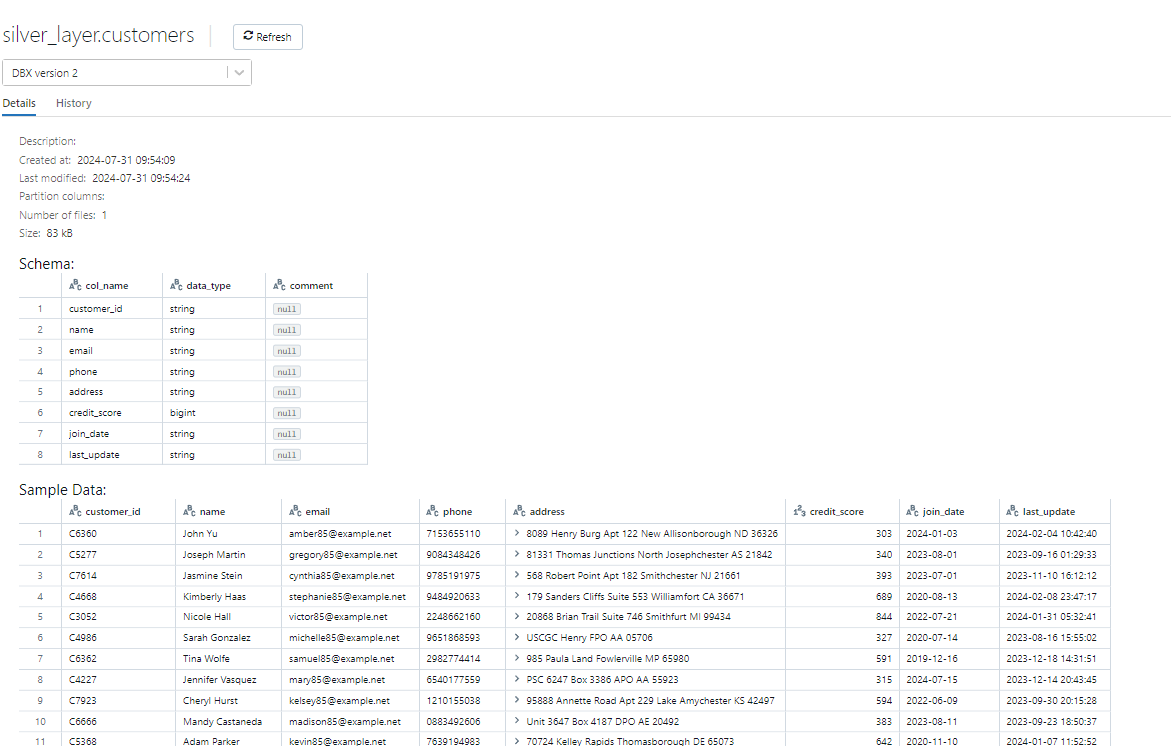
#### Data Processing

* **Transactions (Streaming)**
  + **Read:** From Bronze\_layer.transactions.
  + **Cleanse:** Apply all cleansing functions.
  + **Write:** Save to Silver\_layer.transactions with checkpoint Silver\_layer.checkpoint\_1.
  + **Status:** Streaming initiated.
* **Customers (Batch)**
  + **Read:** From Bronze\_layer.customers.
  + **Cleanse:** Apply all cleansing functions.
  + **Write:** Save to Silver\_layer.customers (overwrite).
* **Branches (Batch)**
  + **Read:** From Bronze\_layer.branches.
  + **Cleanse:** Apply all cleansing functions.
  + **Write:** Save to Silver\_layer.branches (overwrite).

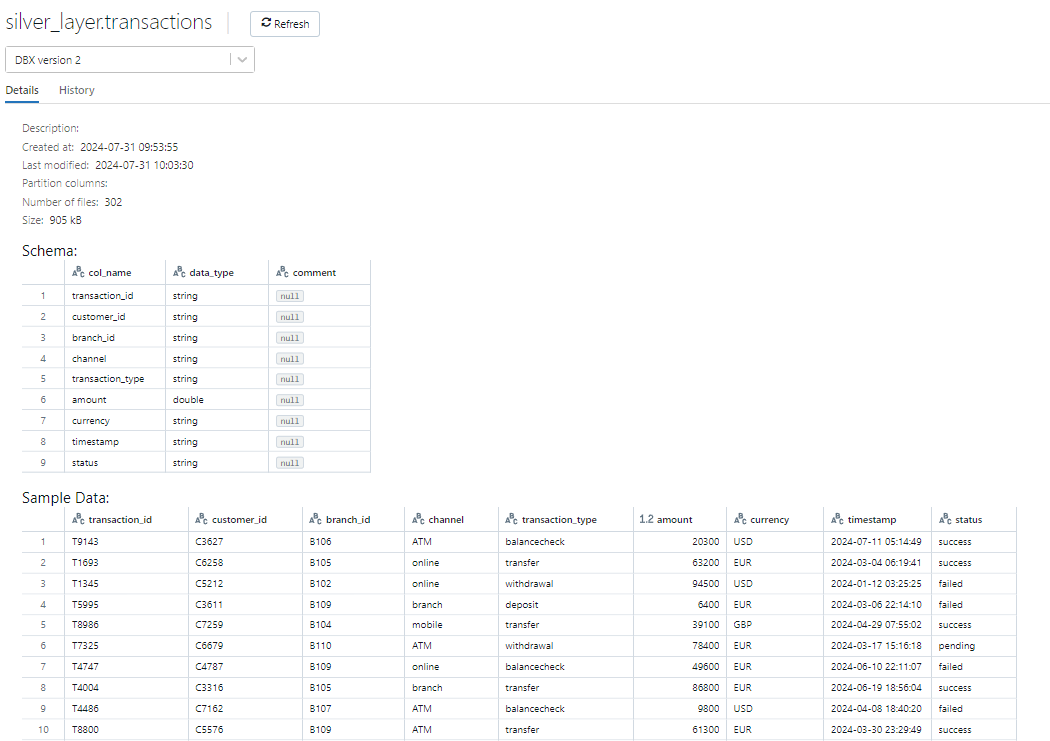
#### Execution

* **Start Processing:** Run process\_and\_cleanse\_data().
* **Handle Interruptions:** Ensure streaming can be interrupted gracefully.

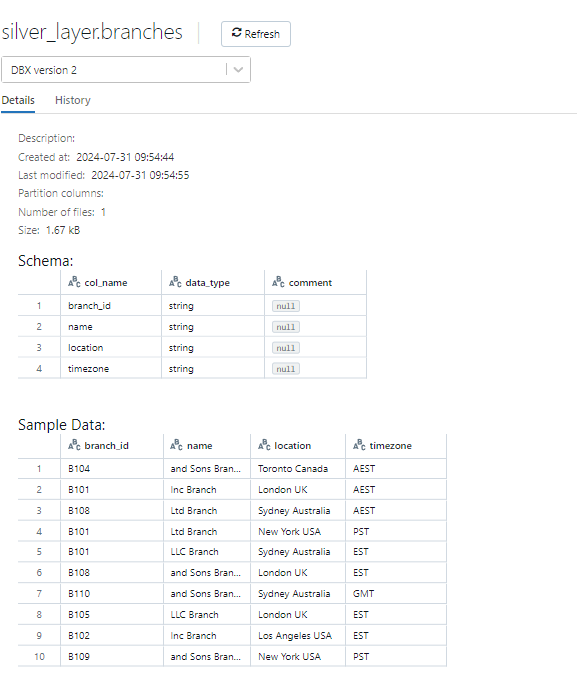
**Customer Table**

****

**Transactions Table**

****

**Branch Table**

****

### Aggregation and Creation of Fraud Flags and Customer Segments

#### Fraud Flags Table

* **Transformation Logic:**
  + **Flag Type:** Identify flags based on amount, channel, and transaction\_type.
  + **Confidence Score:** Assign scores based on flag type.
  + **Generate Flag ID:** Unique ID for each flag using transaction ID.
* **Output:** Create fraud\_flags table with columns flag\_id, transaction\_id, flag\_type, confidence\_score, and timestamp.

#### Customer Segments Table

* **Transformation Logic:**
  + **High Value Customers:** Customers with transaction volume > $100,000.
  + **New Users:** Customers who joined in the last 30 days.
  + **Combine Segments:** Union high value and new users.
  + **Generate Segment ID:** Unique ID for each segment.
* **Output:** Create customer\_segments table with columns customer\_id, segment\_name, segment\_description, segment\_id, and last\_updated.

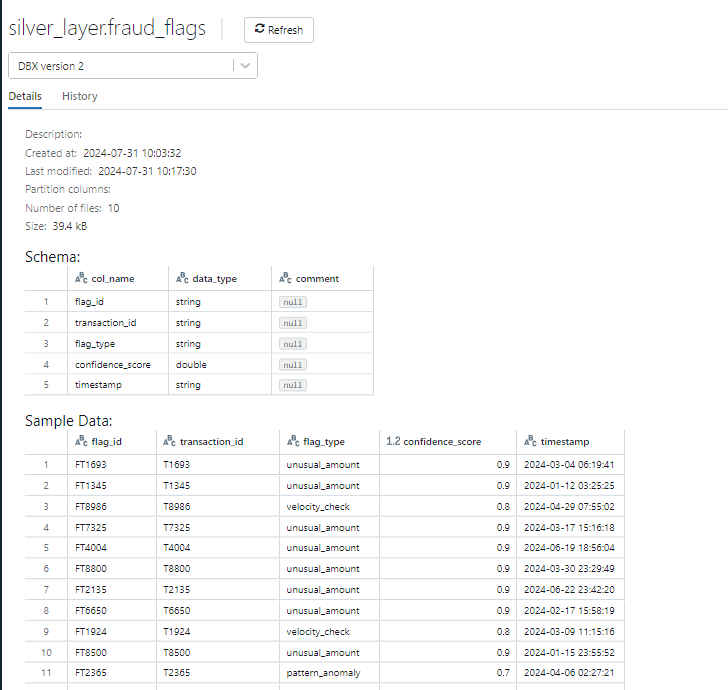
#### Streaming Data Ingestion

* **Fraud Flags:**
  + **Source:** Silver\_layer.transactions
  + **Write:** To Silver\_layer.fraud\_flags with checkpoint fraud\_flag\_1.
  + **Output Mode:** Append.
* **Customer Segments:**
  + **Sources:** Silver\_layer.transactions and Silver\_layer.customers
  + **Write:** To Silver\_layer.customer\_segments with checkpoint customer\_segments\_1.
  + **Output Mode:** Complete.

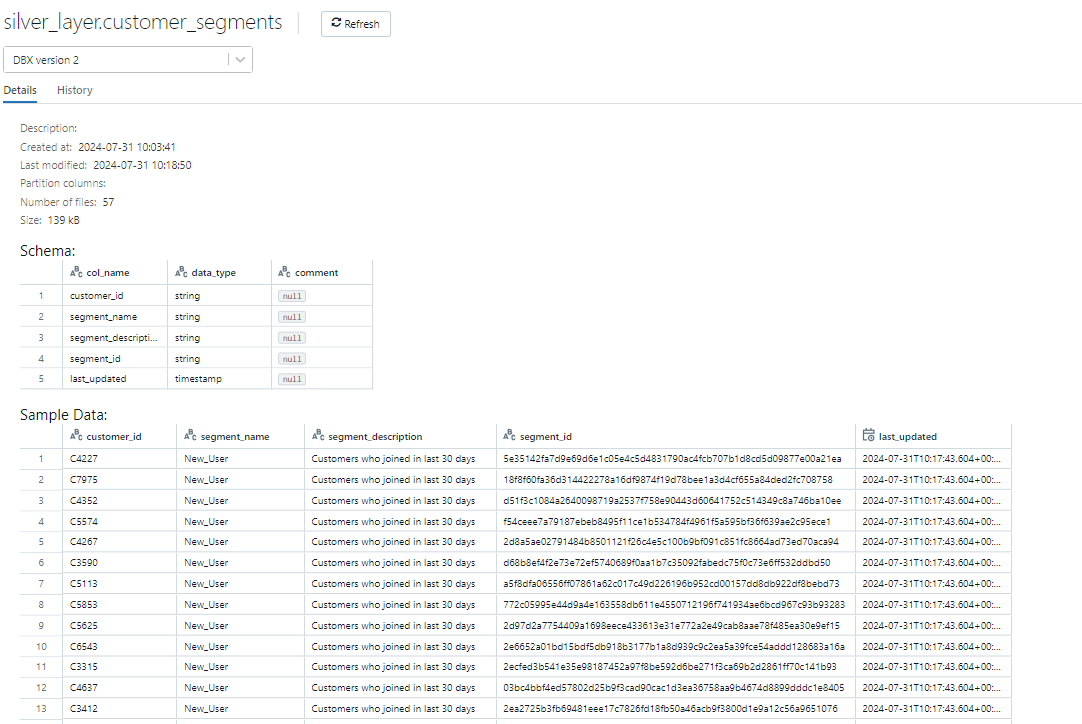
#### Execution

* **Start Streaming:** Run functions ingest\_fraud\_flags() and ingest\_customer\_segments().
* **Await Termination:** Ensure streaming processes continue until manually interrupted.

**Fraud\_flags**

****

**Customer\_segements**

****

### Aggregation and Storage of Gold Layer Data

#### Aggregations

1. **Total Transaction Amount by Branch**
   * **Operation:** Sum of transaction amounts grouped by branch\_id.
   * **Output Table:** Gold\_layer.Tot\_trans\_amt\_by\_branch.
2. **Number of Transactions by Channel and Status**
   * **Operation:** Count of transactions grouped by channel and status.
   * **Output Table:** Gold\_layer.trans\_count\_by\_branch\_and\_chnl.
3. **Highest Transaction Amount by Branch and Customer**
   * **Operation:** Maximum transaction amount grouped by branch\_id and customer\_id.
   * **Output Table:** Gold\_layer.high\_trans\_by\_branch\_cus.
4. **Monthly Transaction Count and Total Amount**
   * **Operation:** Count and sum of transactions grouped by month.
   * **Output Table:** Gold\_layer.monthly\_trans.
5. **Transaction Volumes by Channel**
   * **Operation:** Count of transactions grouped by channel.
   * **Output Table:** Gold\_layer.trans\_volumes.
6. **Transaction Geographic Distribution by Branch Location**
   * **Operation:** Count of transactions grouped by branch location.
   * **Output Table:** Gold\_layer.trans\_geo\_distinct.
7. **Transaction Patterns by Month**
   * **Operation:** Count and sum of transactions grouped by month and transaction\_type.
   * **Output Table:** Gold\_layer.trans\_patterns.

#### Storage

* **Format:** Delta tables.
* **Checkpoint Locations:** Specific to each aggregation for fault tolerance.