**Sweat Signal**

**WEEK 1- Architecture, Data Models & Secure Ingestion Pipelines**

SweatSignal is a privacy-first wellness analytics platform designed to allow yoga, therapy, and somatic wellness practitioners to collect anonymous client feedback via QR codes, analyze emotional impact using machine learning, and publish transparent public impact dashboards and reports.

This project was built using Azure Blob Storage,Azure Databricks (PySpark),Azure SQL,DatabaseFlask Web App,Python ML,Automated PDF Reporting.

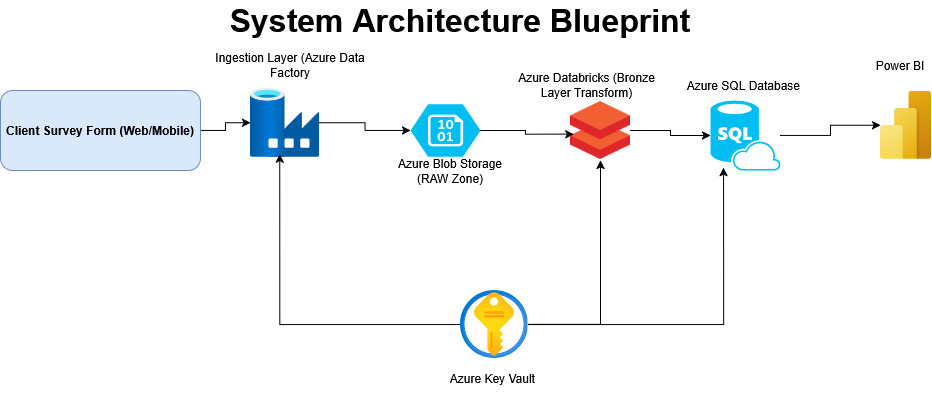
The objective of Week 1 was to design and implement the Azure data backbone for an ethical, anonymous wellness feedback system. This phase focused on system architecture design, secure ingestion of anonymous feedback, dimensional data modeling, and initial data transformation using PySpark.

The Technologies Used were Azure Blob Storage Raw data landing zone,Azure Databricks (Apache Spark) – Data transformation layer,Azure SQL Database Structured analytical storage,PySpark – Data cleansing and transformation

1. System Architecture Blueprint

Front-end (form) → Azure Data Factory → Azure Blob Storage → Azure

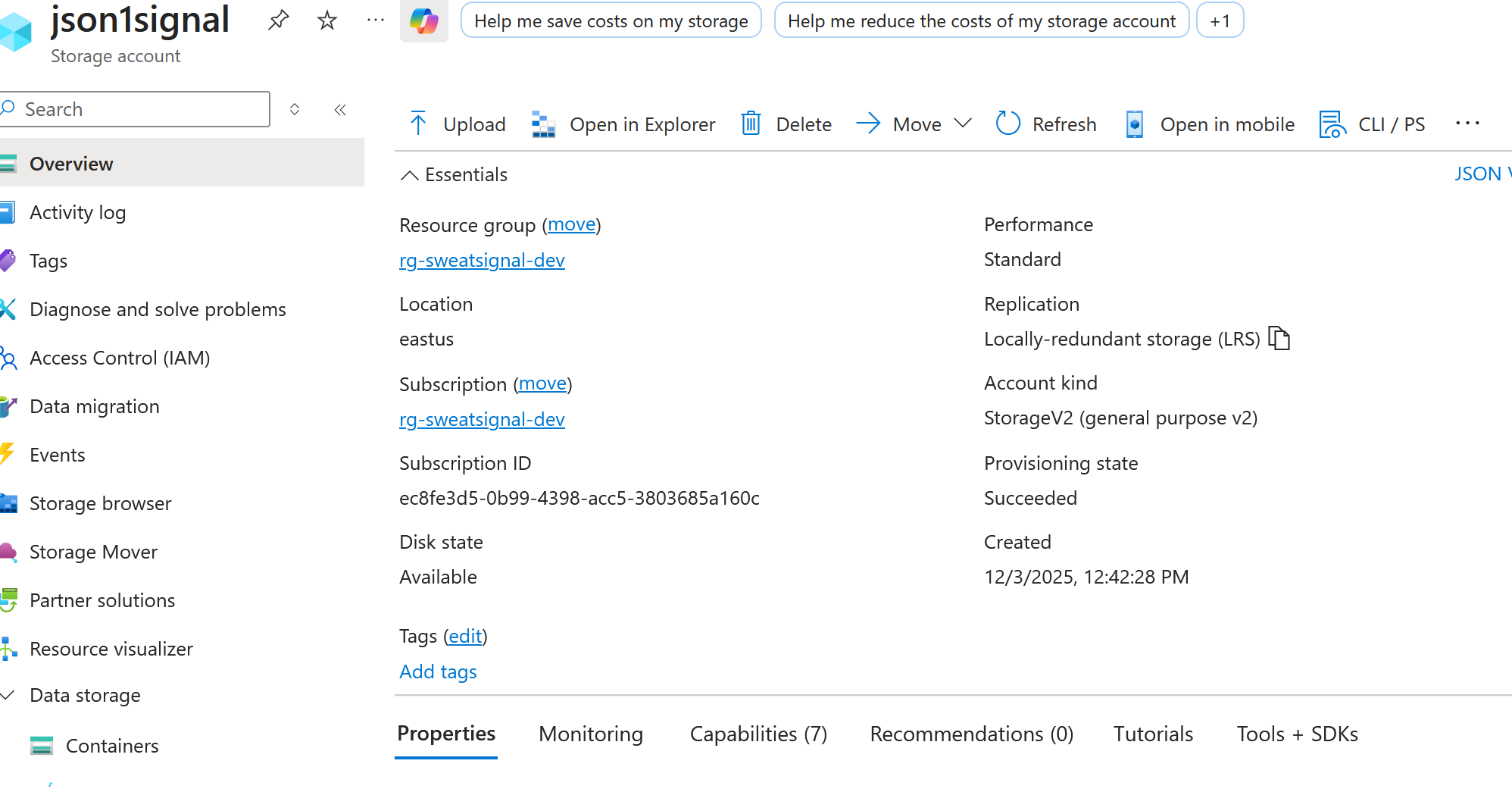
Databricks → Azure SQL Database

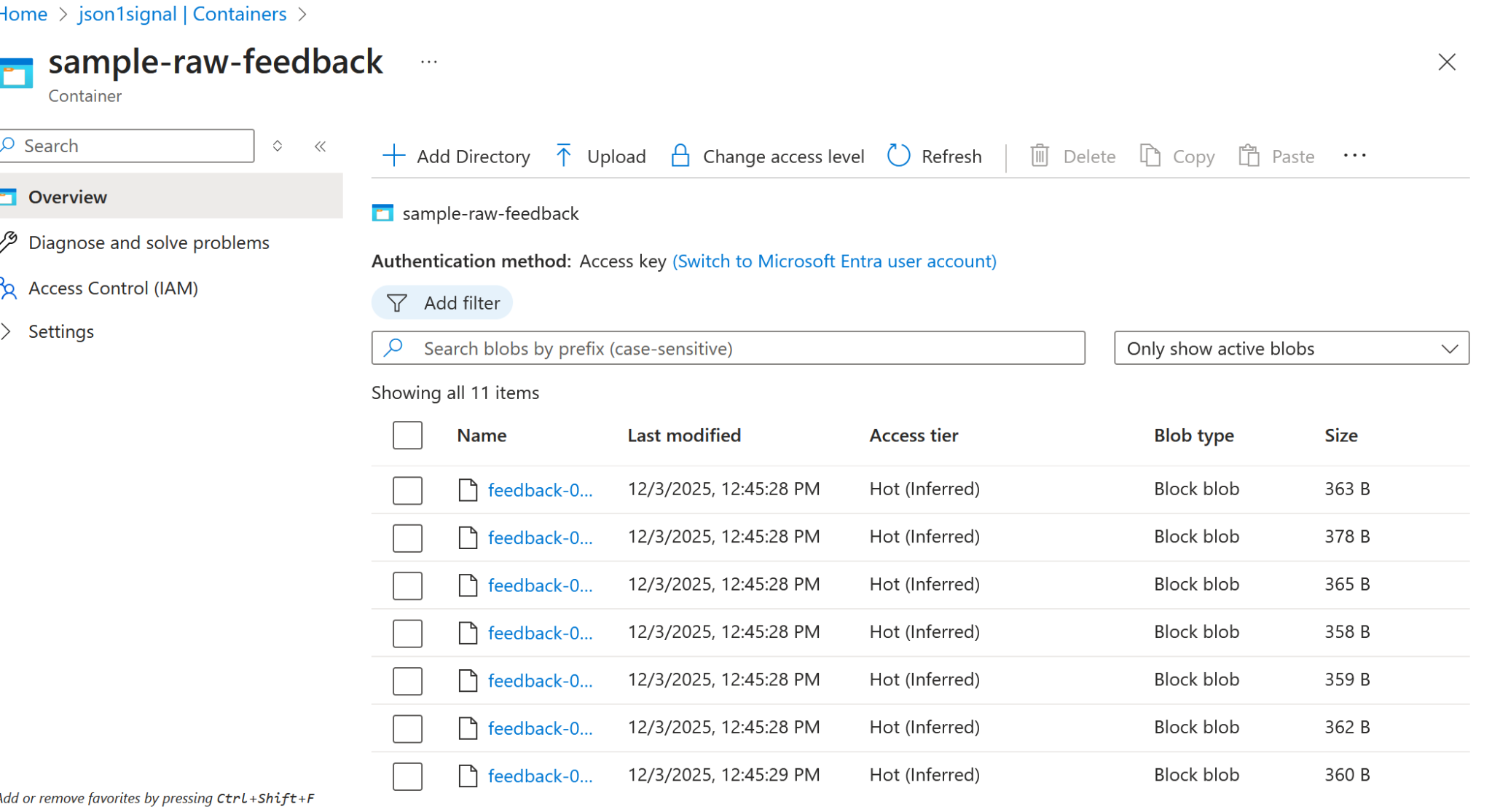


2. Build Anonymous Feedback Ingestion Flow (ADF)

Use Azure Data Factory to ingest JSON survey submissions into Blob Storage.

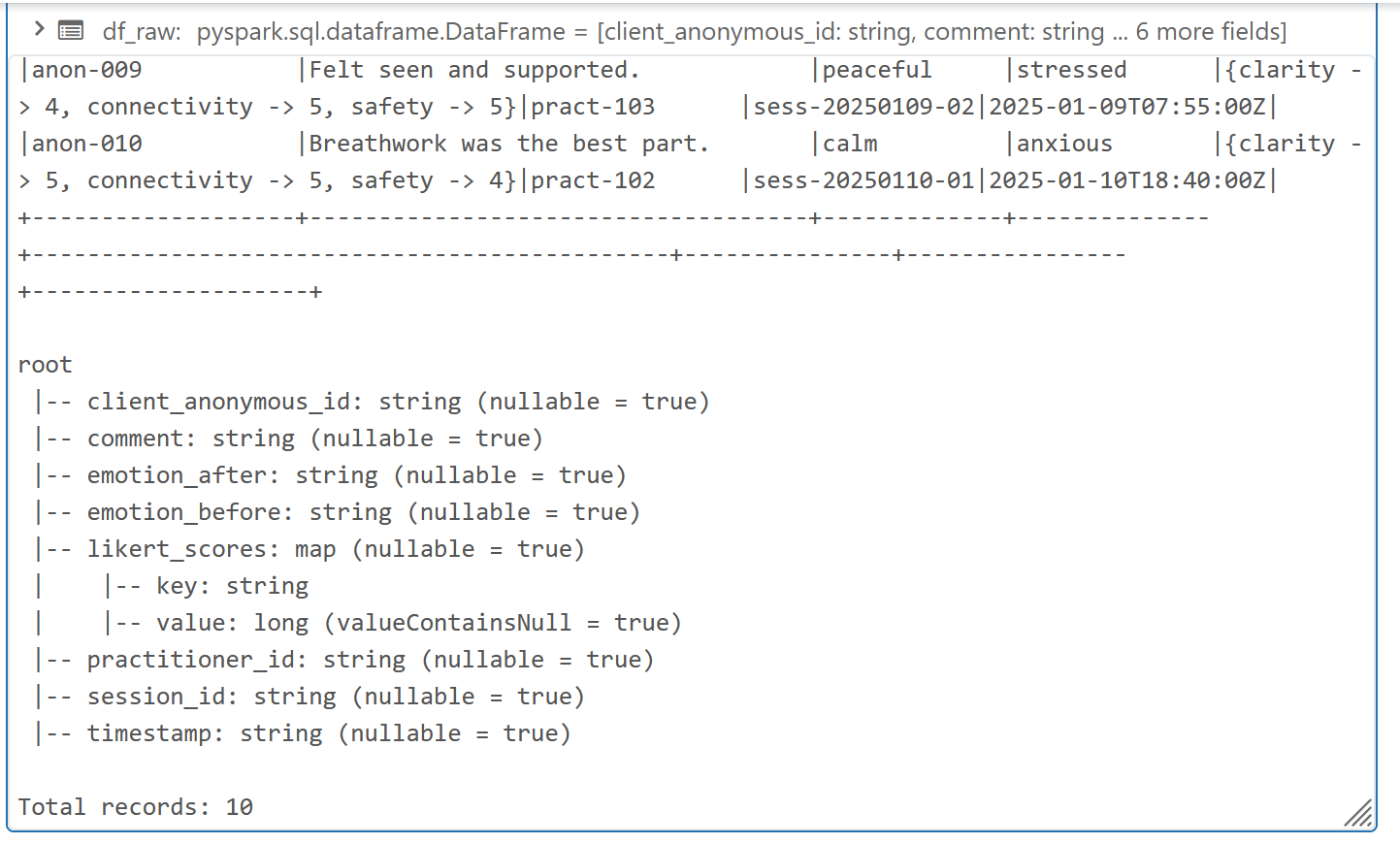
The blob storage and container for raw data below:

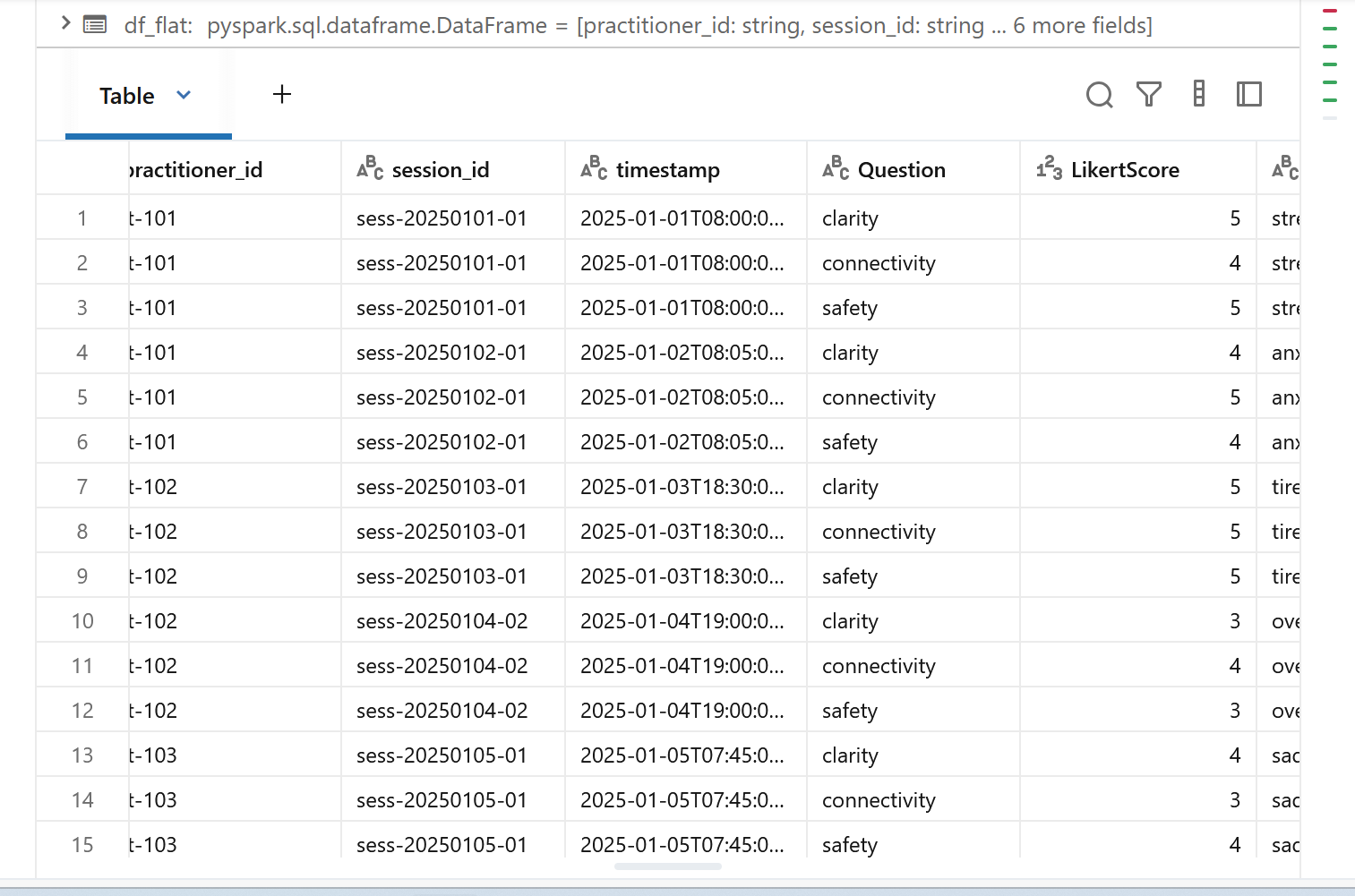




A sample of 10 records were created to Use to ingest JSON survey submissions into Blob Storage, cleaned parameterized pipelines to accept practitioner\_id, session\_id, timestamp,

Likert/emotion,score free-text comments. Below are the screenshots of the notebook of cleaned data.





After the data is cleaned, the next step is to create the dimensions and fact tables.

3. Create Dimensional Model (Synapse / Azure SQL)

• Dimensions:

o Practitioner\_Dim

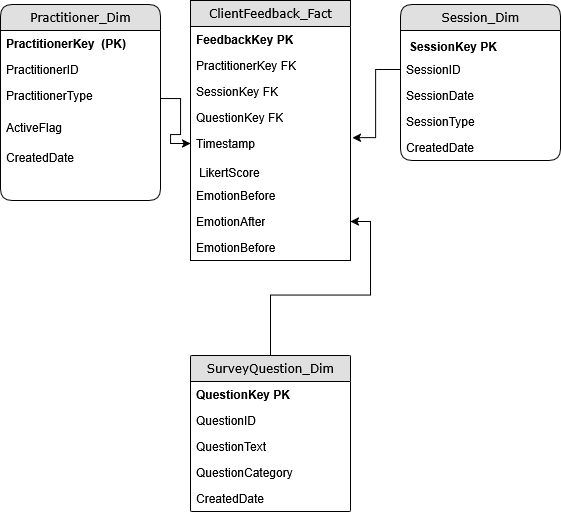
o Session\_Dim

o SurveyQuestion\_Dim

• Fact:

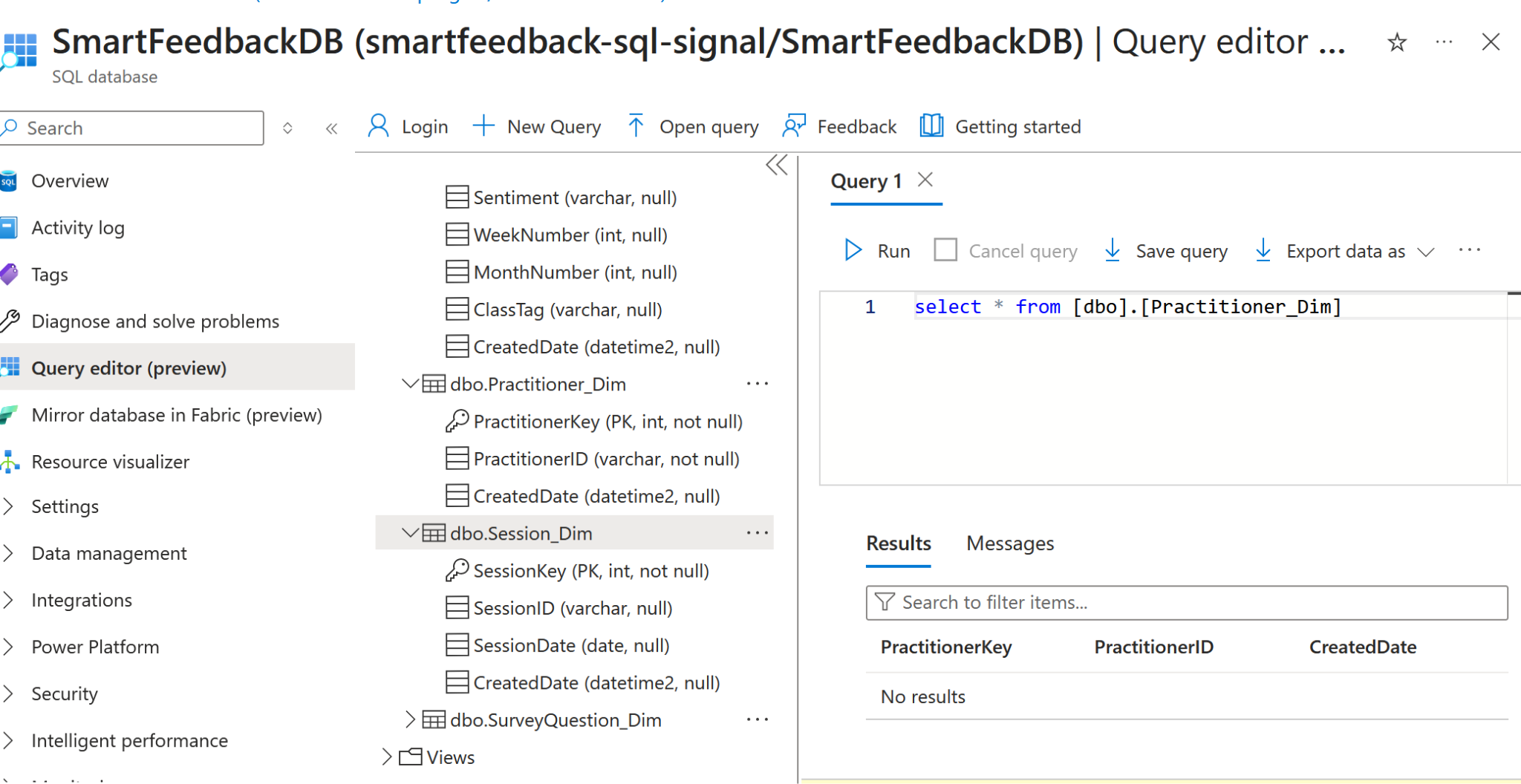
o ClientFeedback\_Fact (anonymous)

**Conceptual data model**

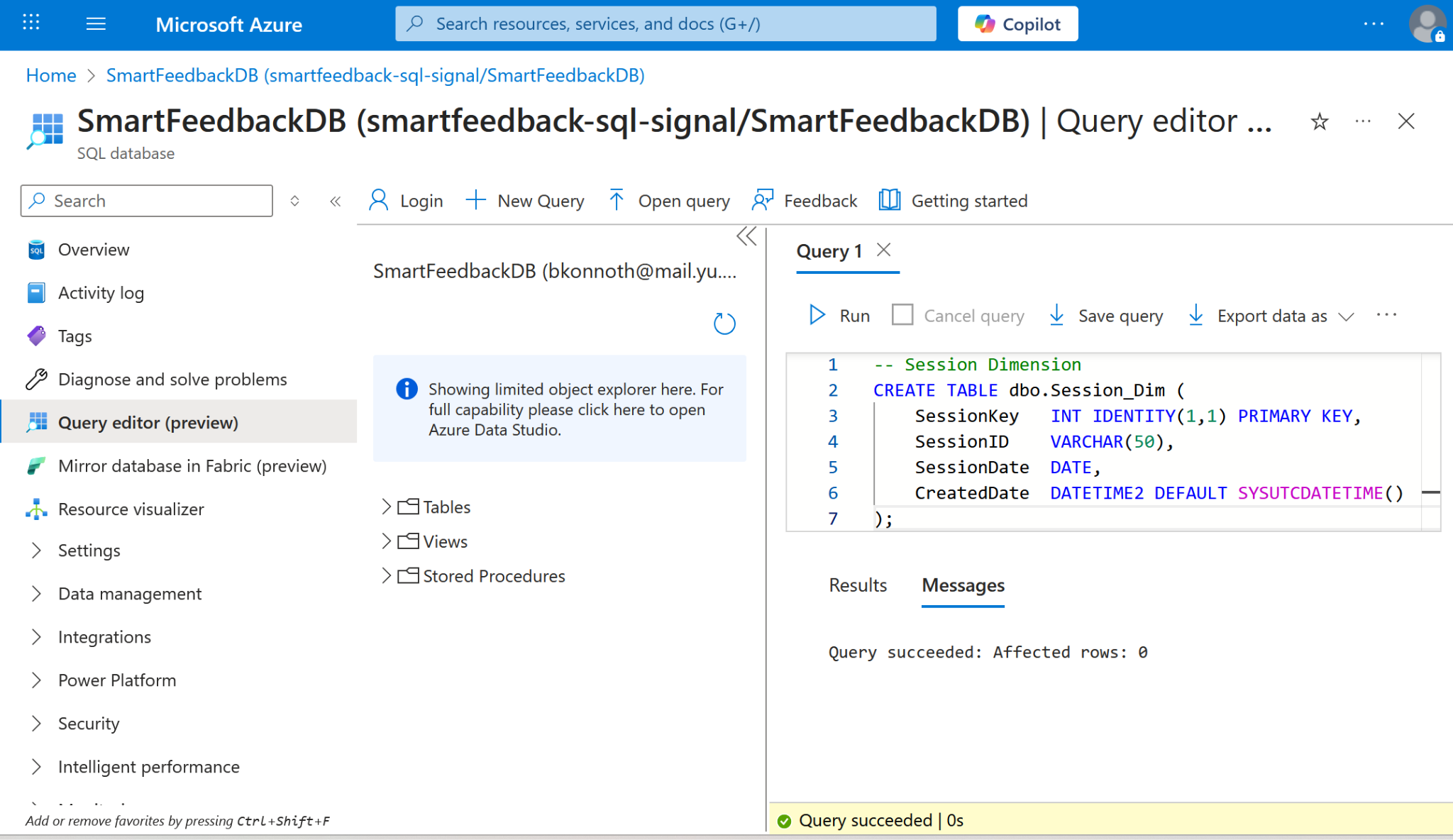
****

Below are the screenshots of the dimensions and fact tables which were created in Azure SQL.

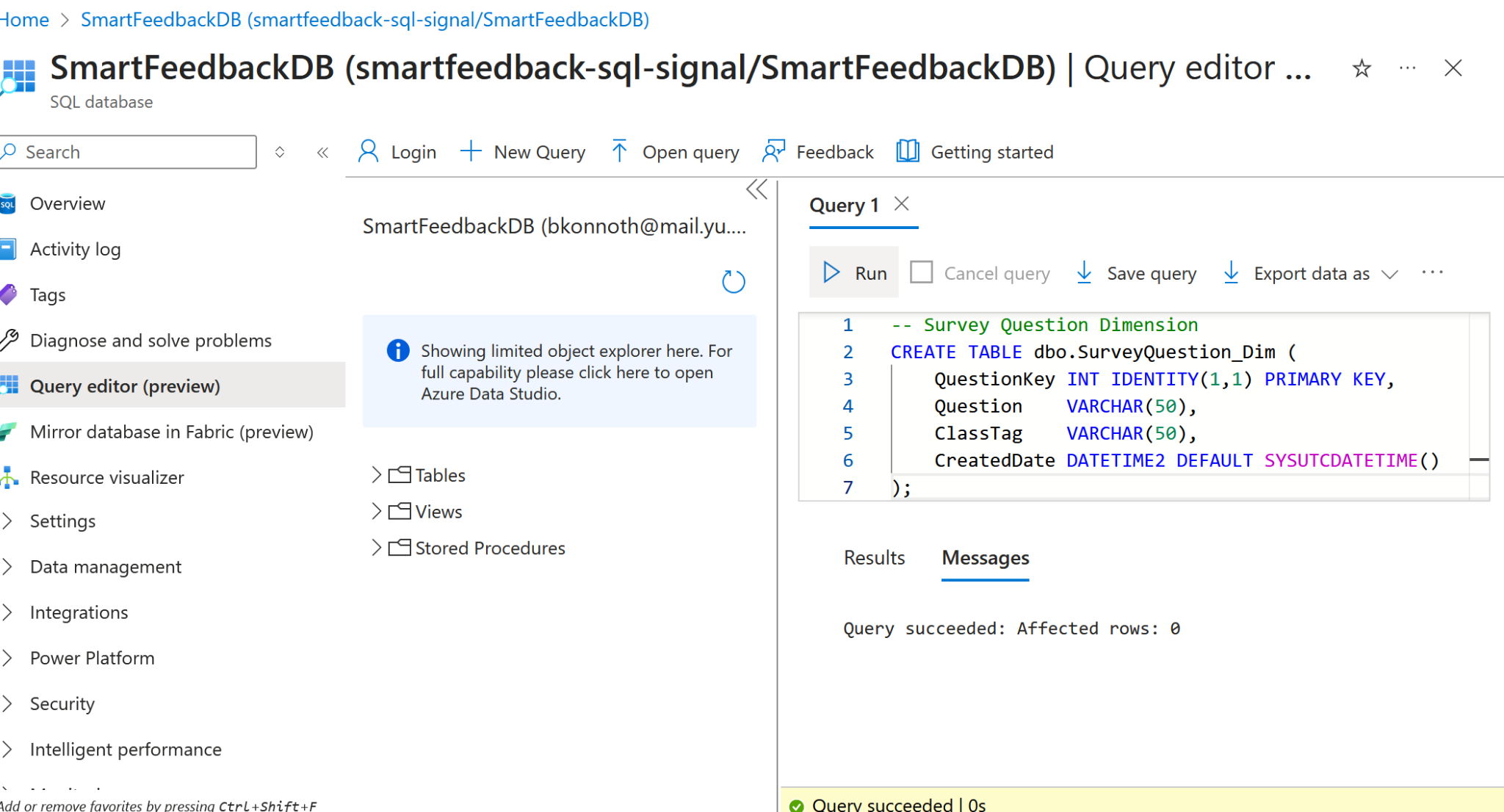
* Practitioner\_dimension



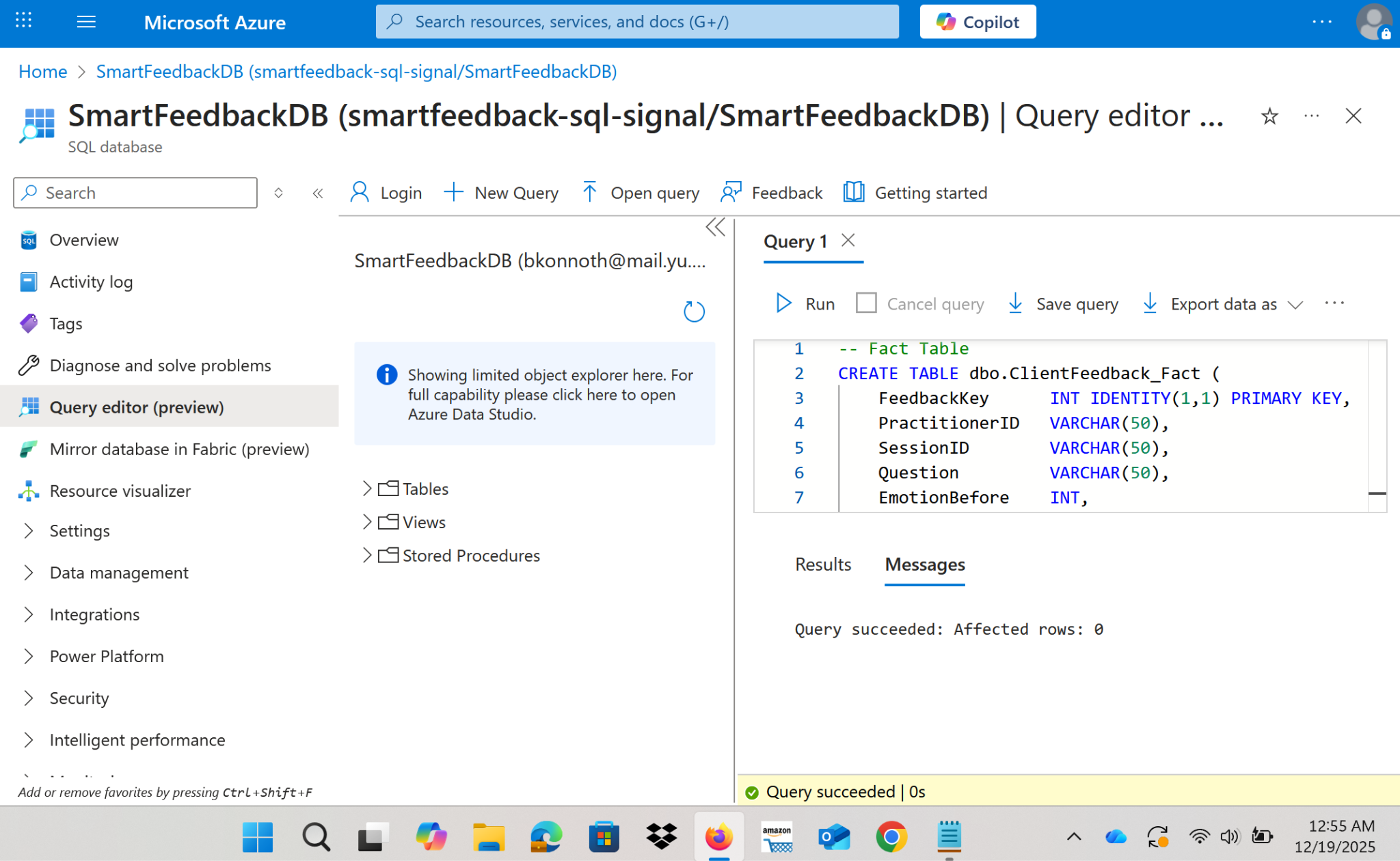
* Session\_dimension

****

* Survey\_Question\_feedback\_Dimension



* Client\_feeback\_Fact



The next process in the project was to create likert score clean the data of no null values and so on.

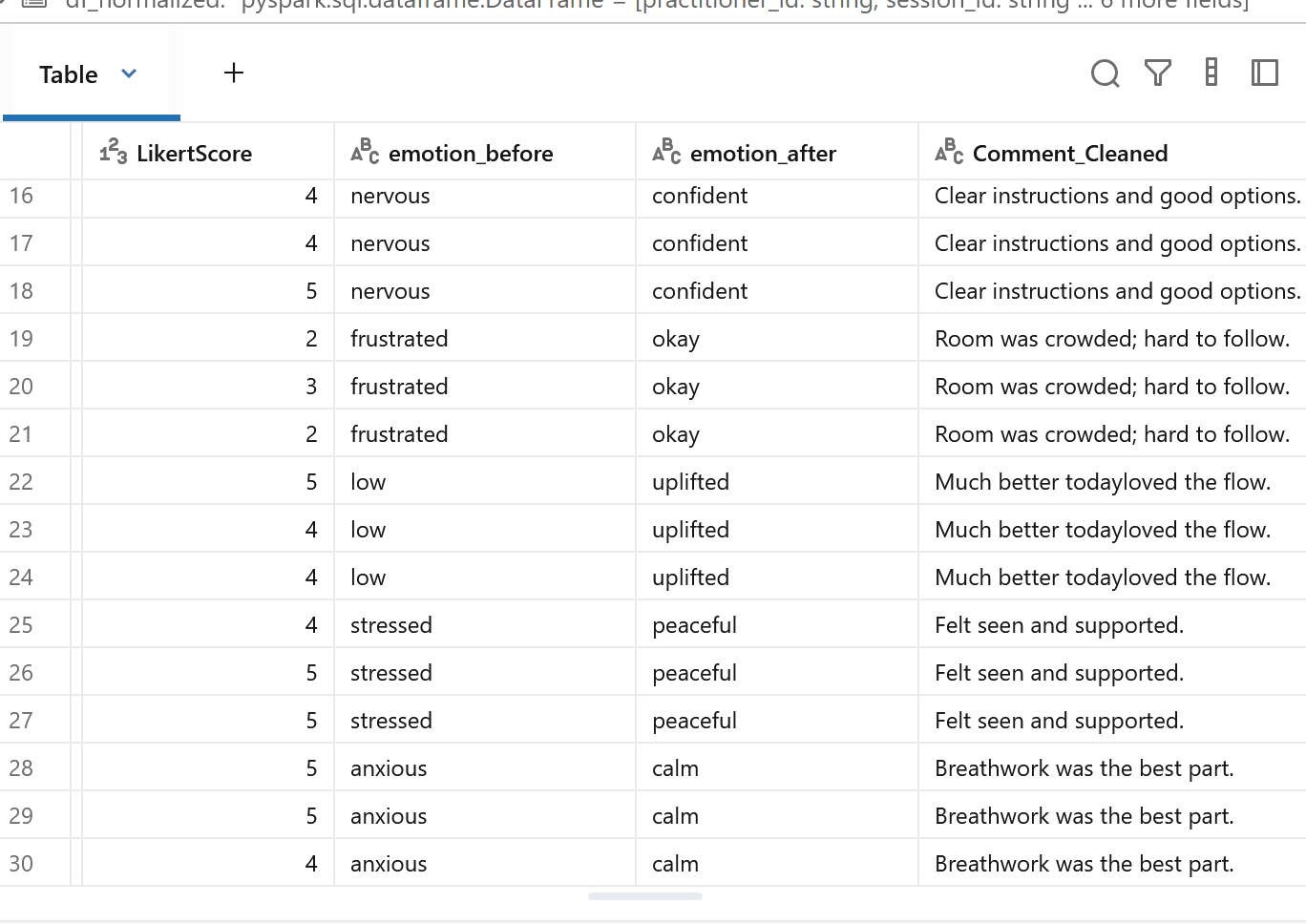
4. Databricks Notebook (PySpark) for Transformations

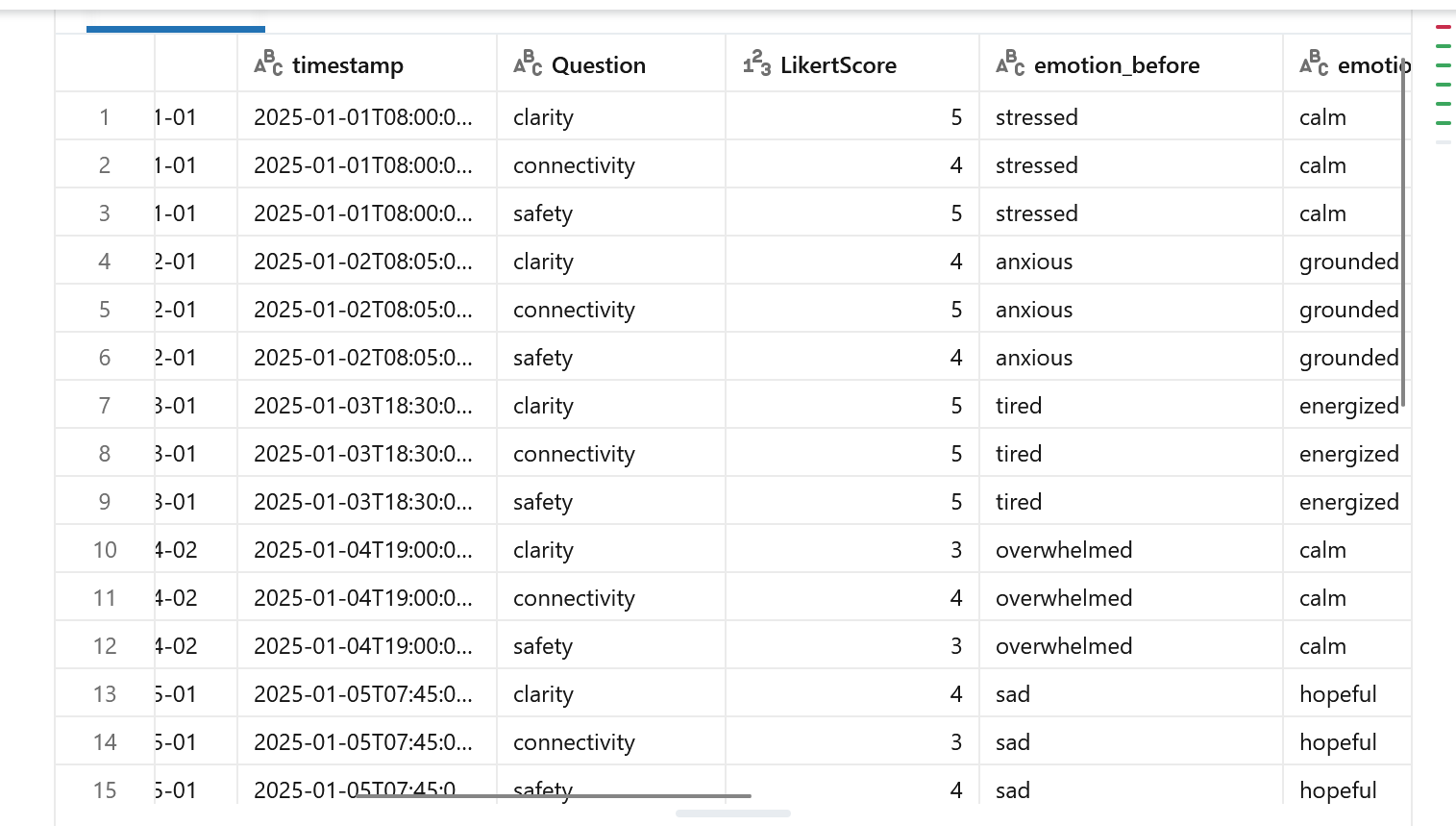
• Clean emoji-based inputs, null values, and normalize Likert scores.

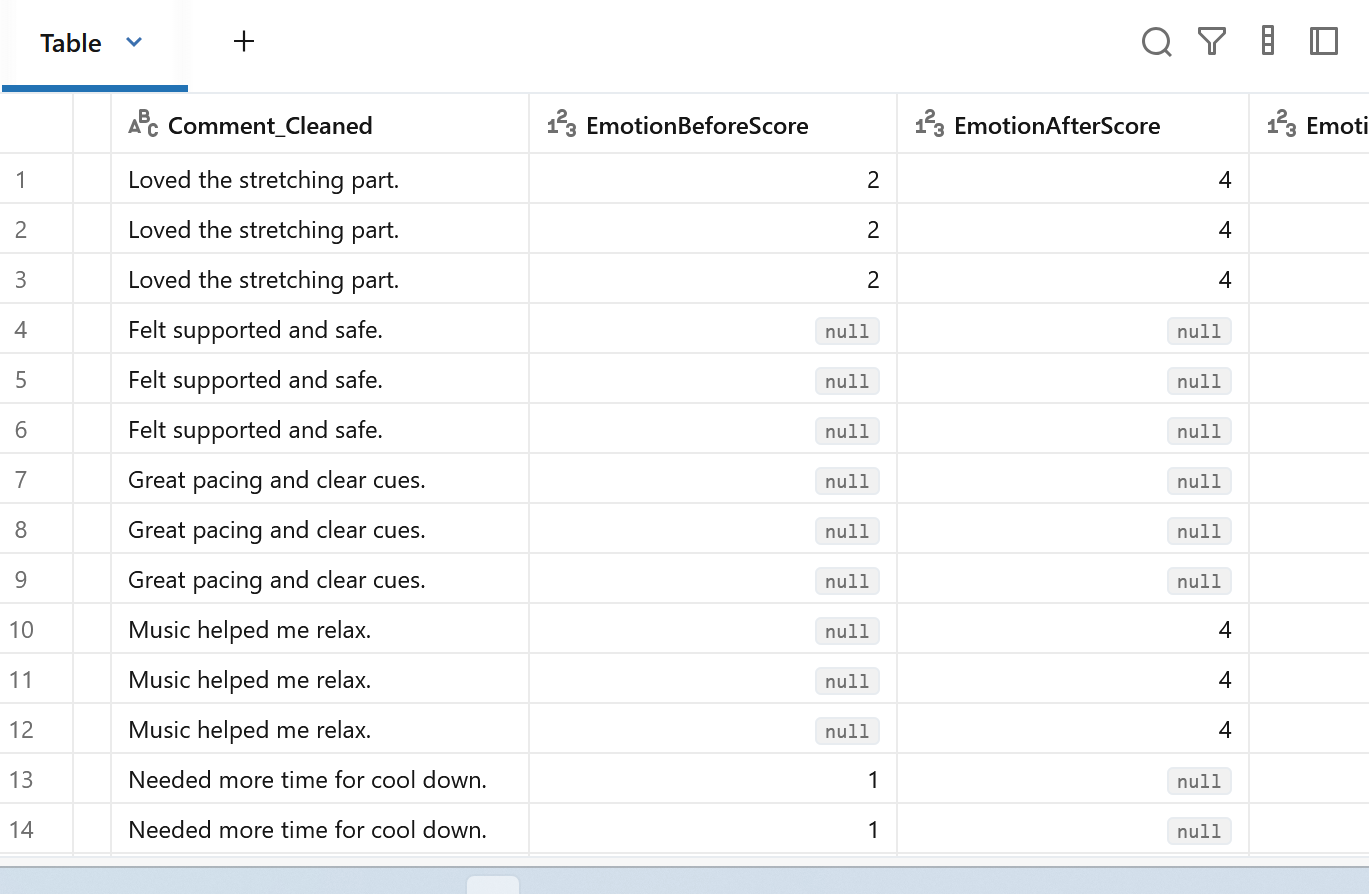
• Create a derived field:

emotional delta (client’s emotional shift after class).

Below is the Databricks notebook output.





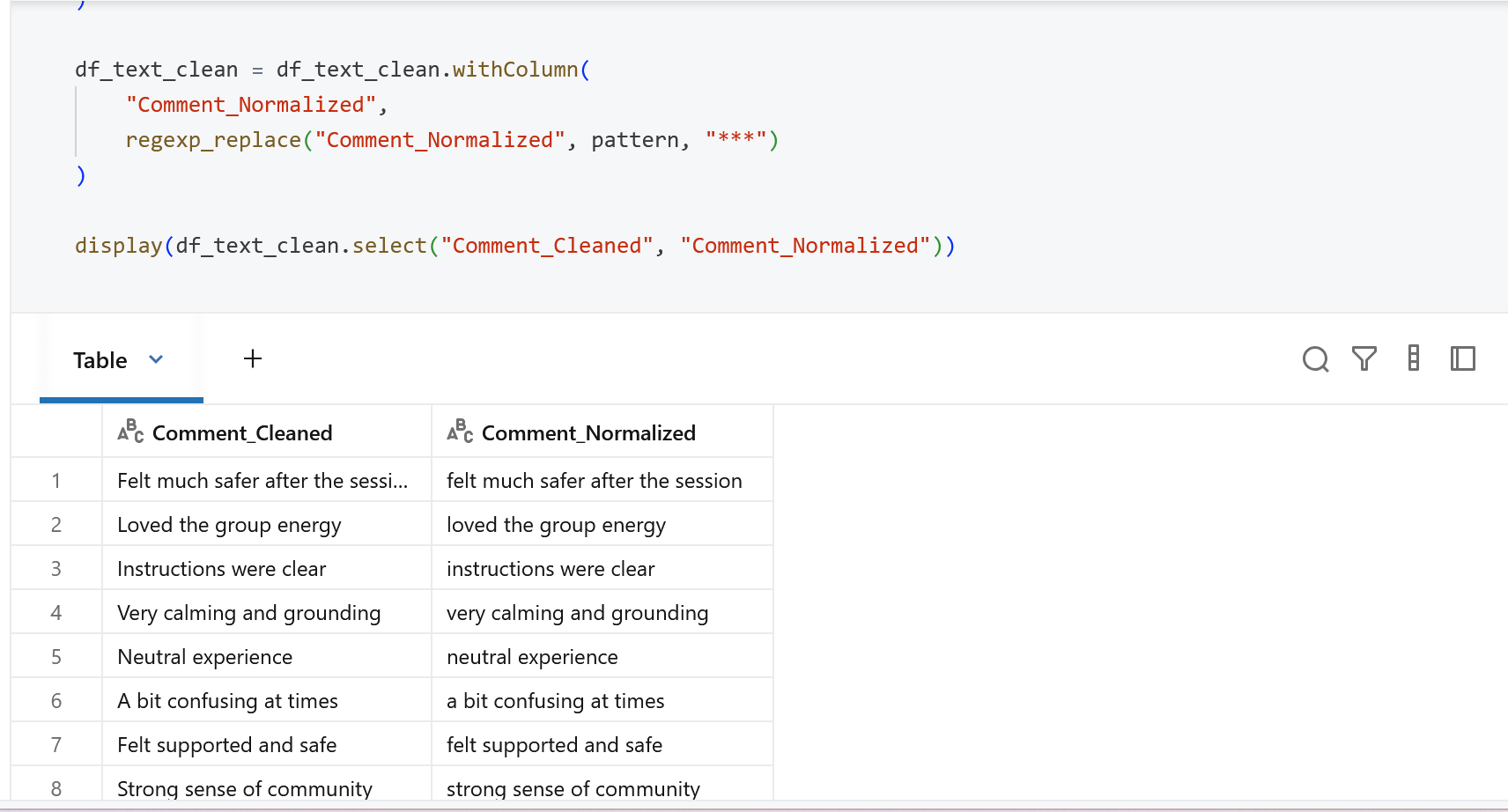


**WEEK 2 — Analytics Layer, Sentiment Modeling**

1. Data Transformation with Databricks (PySpark)Handle text normalization, profanity filtering (optional), tokenization. Add week/month rollups.Tag classes by “Intensity”, “Community-building”, etc.

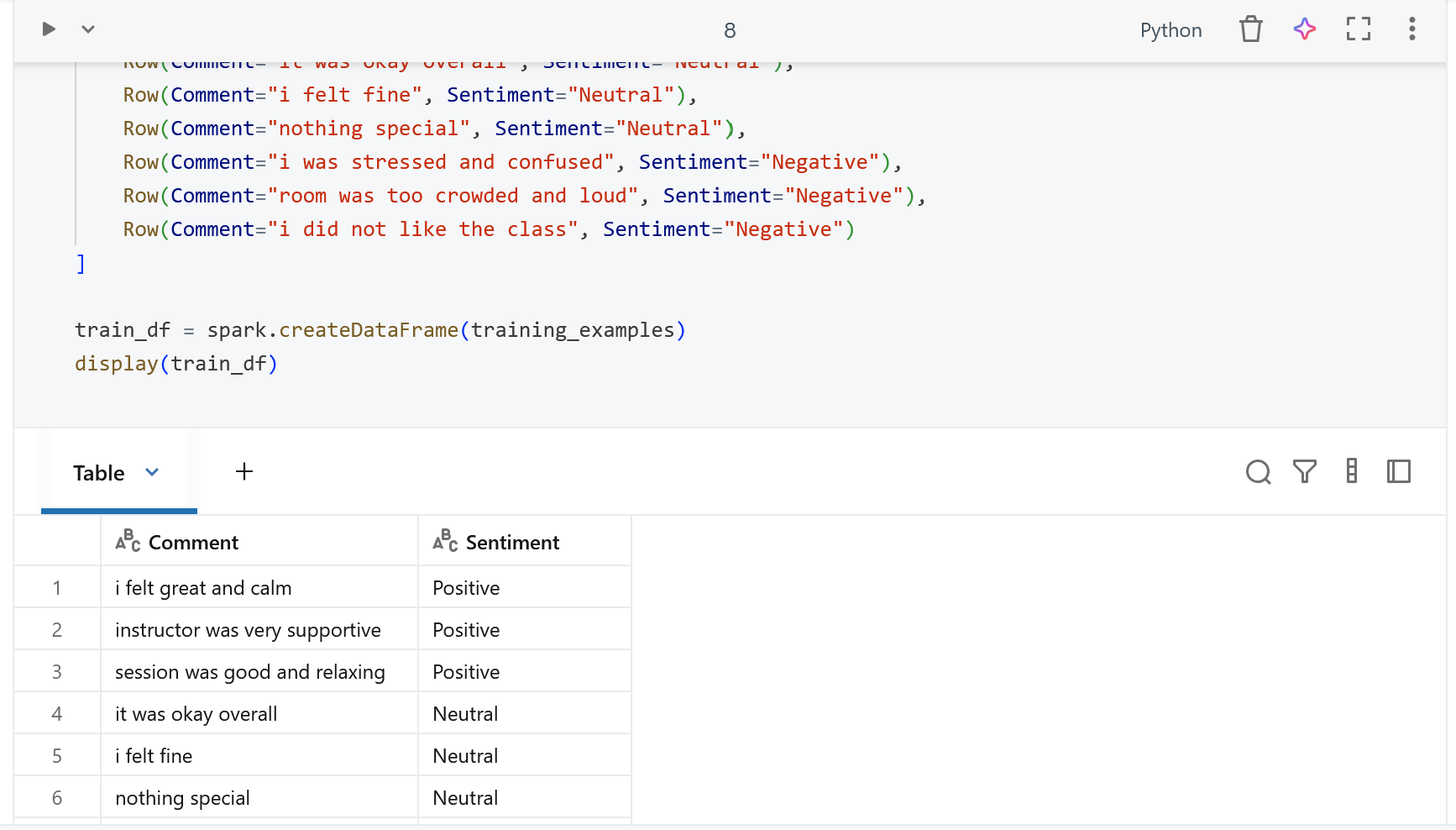
In this I used a **controlled dev dataset** representing real studio feedback coming from Azure blob storage to Databricks notebook. This notebook job run will clean the data, do sentiment modeling and train a model. Finally the cleaned dataset will be passed to azure sql database for further proceeding.

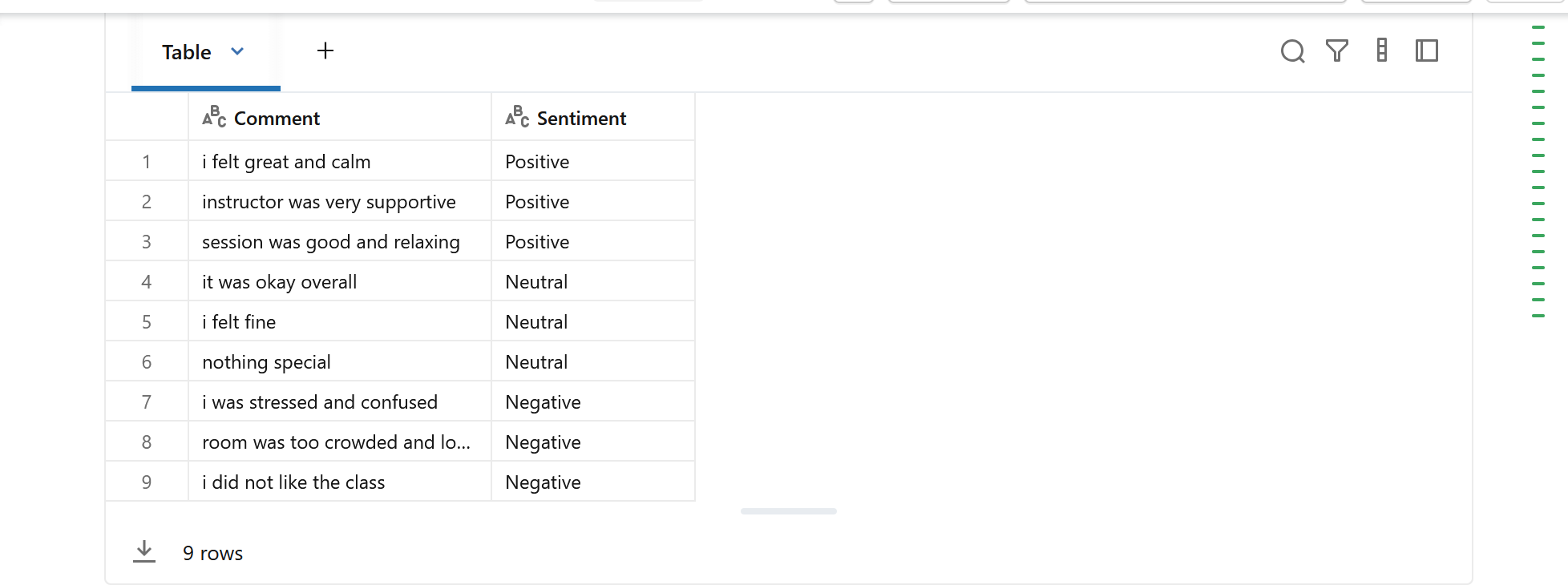
Below are the screenshots:



2. Build Lightweight Sentiment Analysis

Using skills listed under supervised learning (Random Forest, Logistic Regression)Train a 3-class model Positive,Neutral,Constructive and Use Scikit-learn inside Databricks.





The above two images are the sentimental model and text normalized and cleaned.

3. Create Star Schema Tables

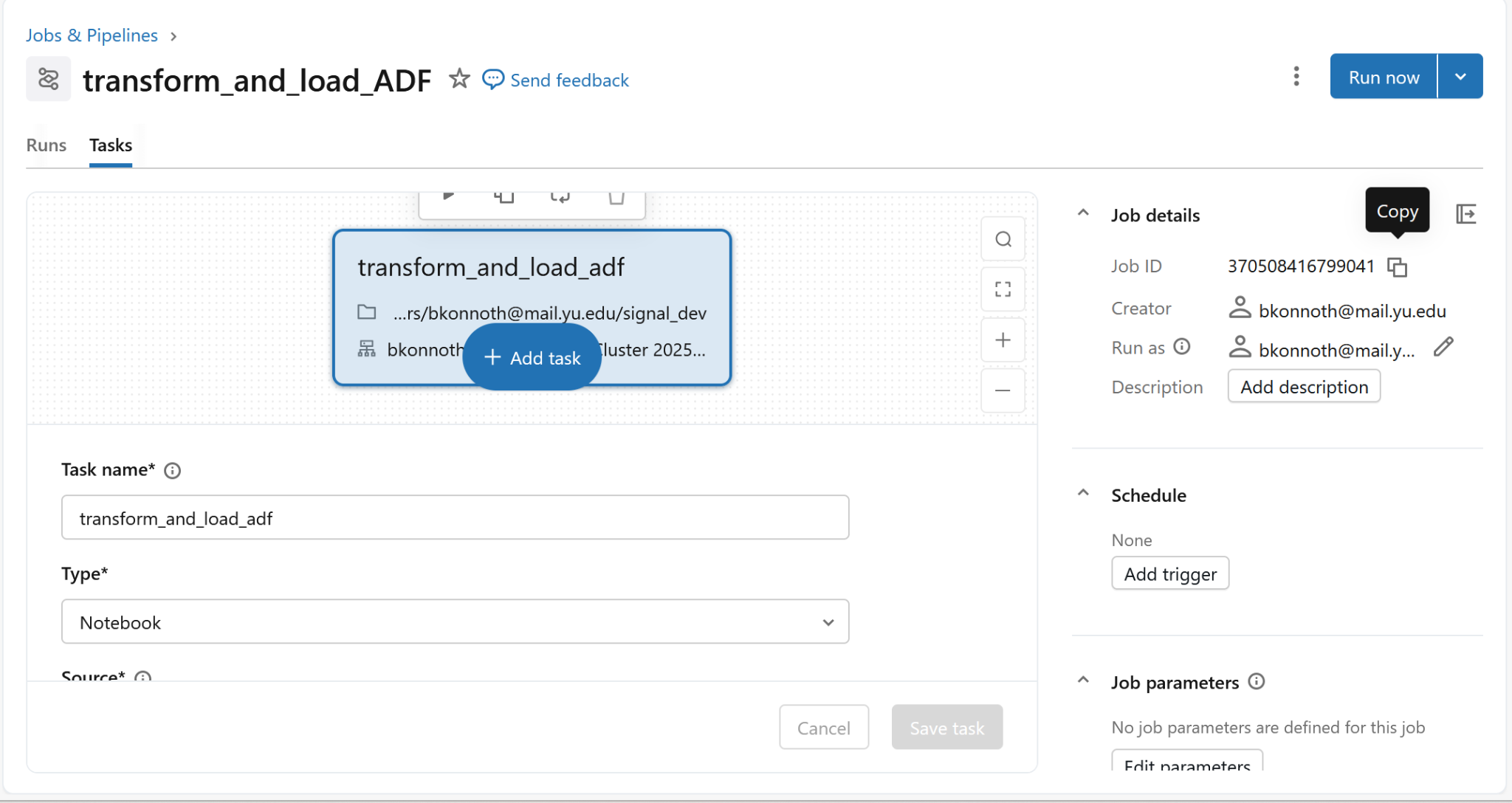
• Load transformed data into Azure SQL / Synapse,Build reporting-friendly views:

o vw\_EmotionalStateTrends

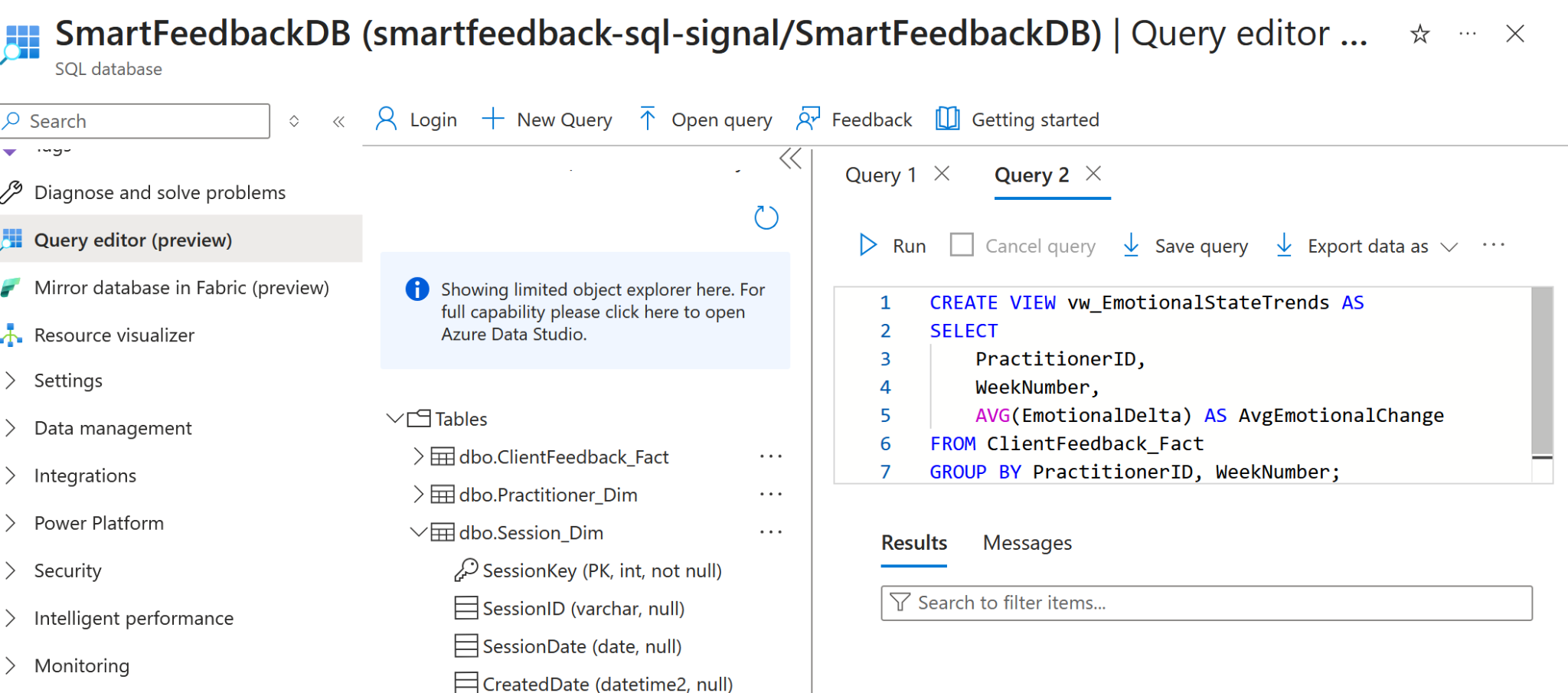
o vw\_SentimentRollups

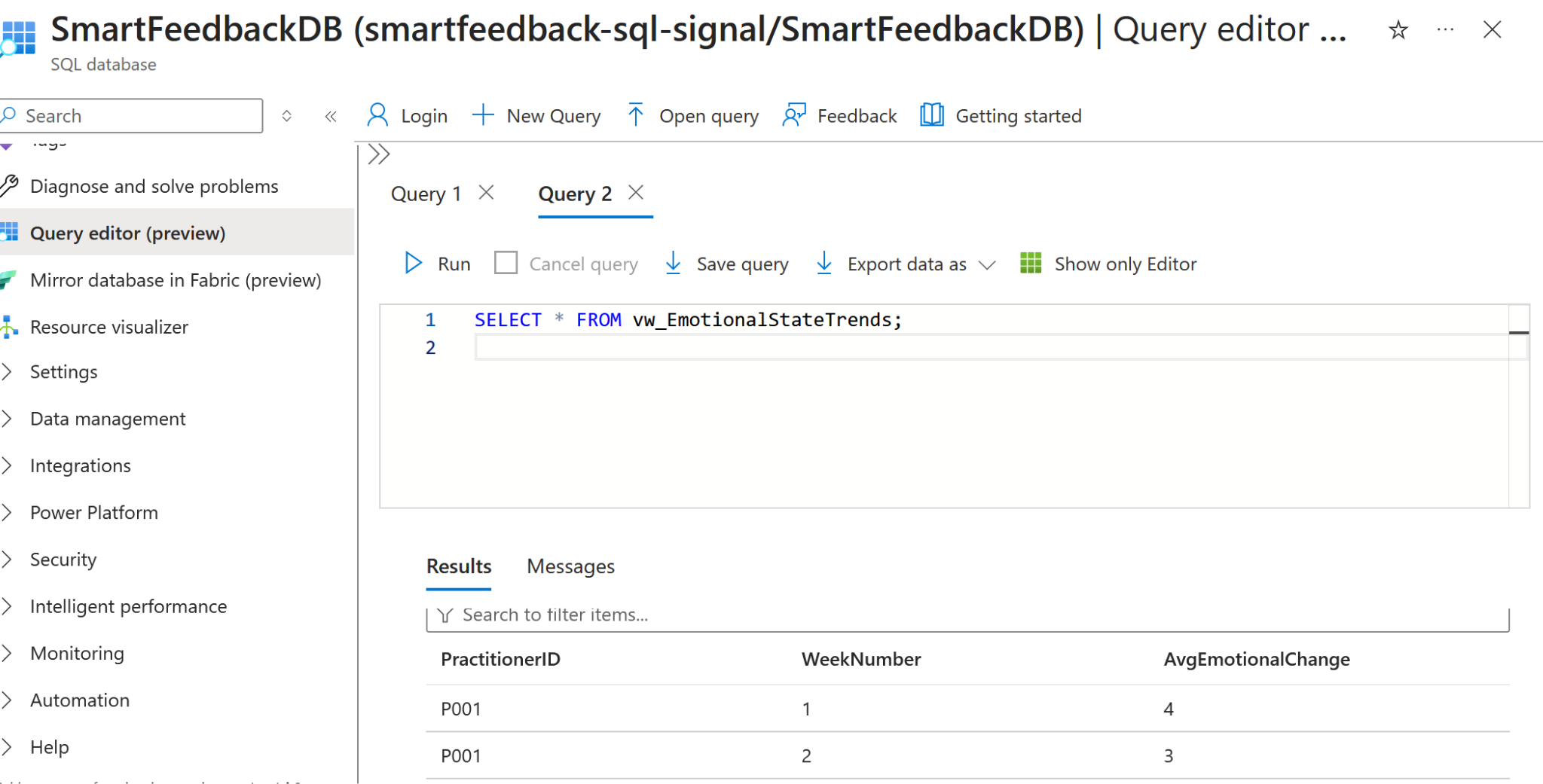
o vw\_PractitionerPublicImpact

For loading data I created a databricks job pipeline when it ran it push the cleaned data to Azure Sql.

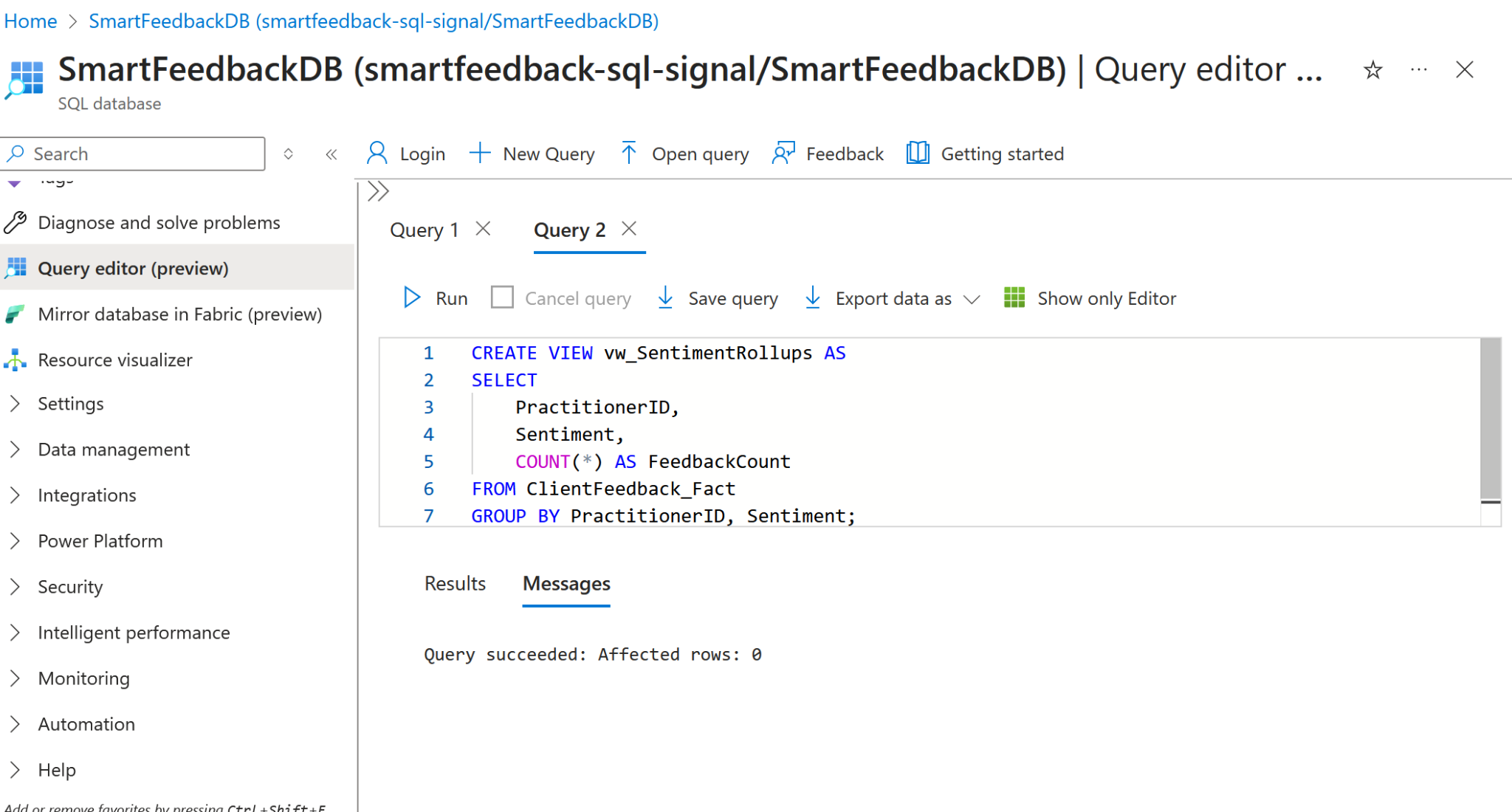


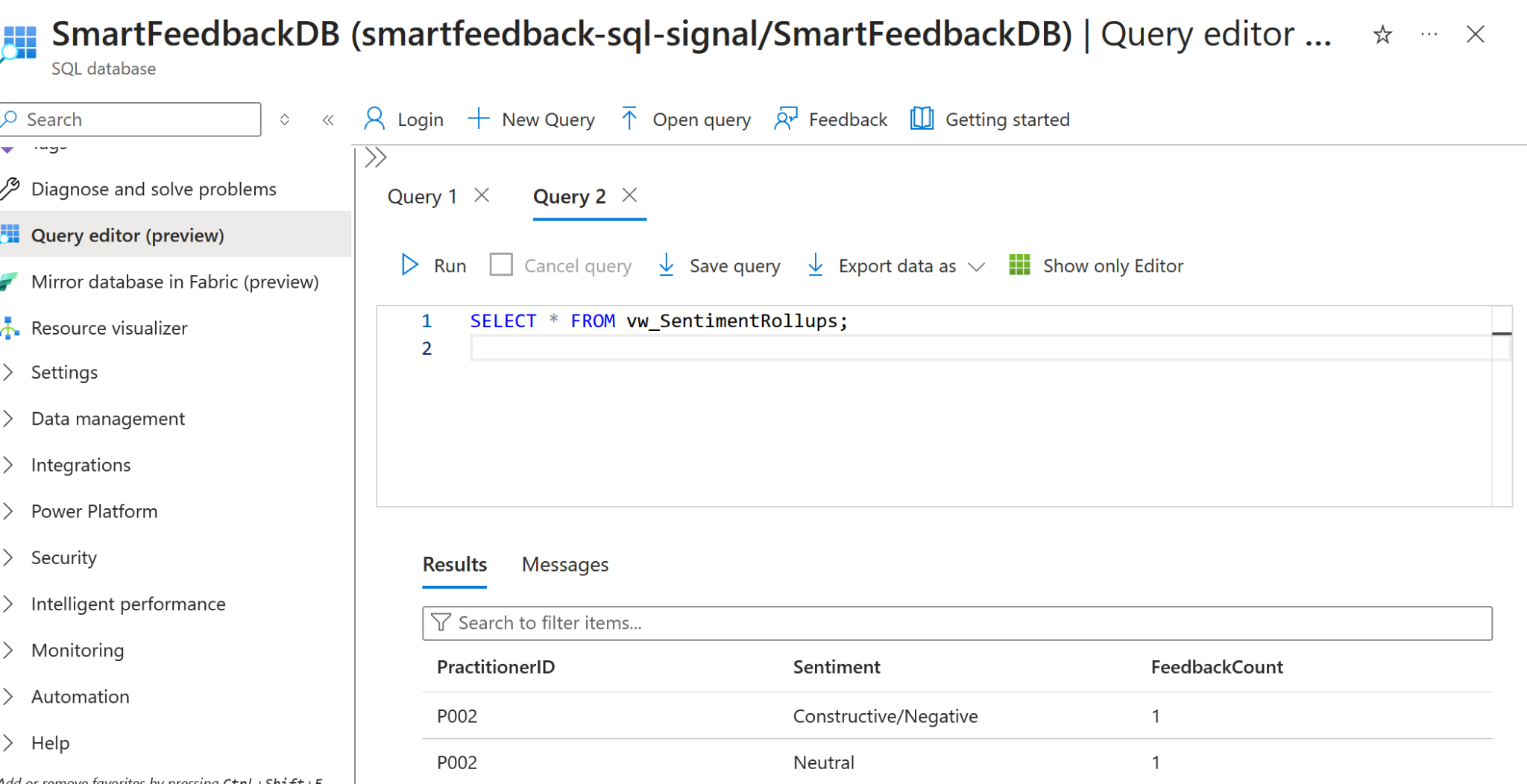
The reporting views were created and results were gathered in the Azure Sql database.



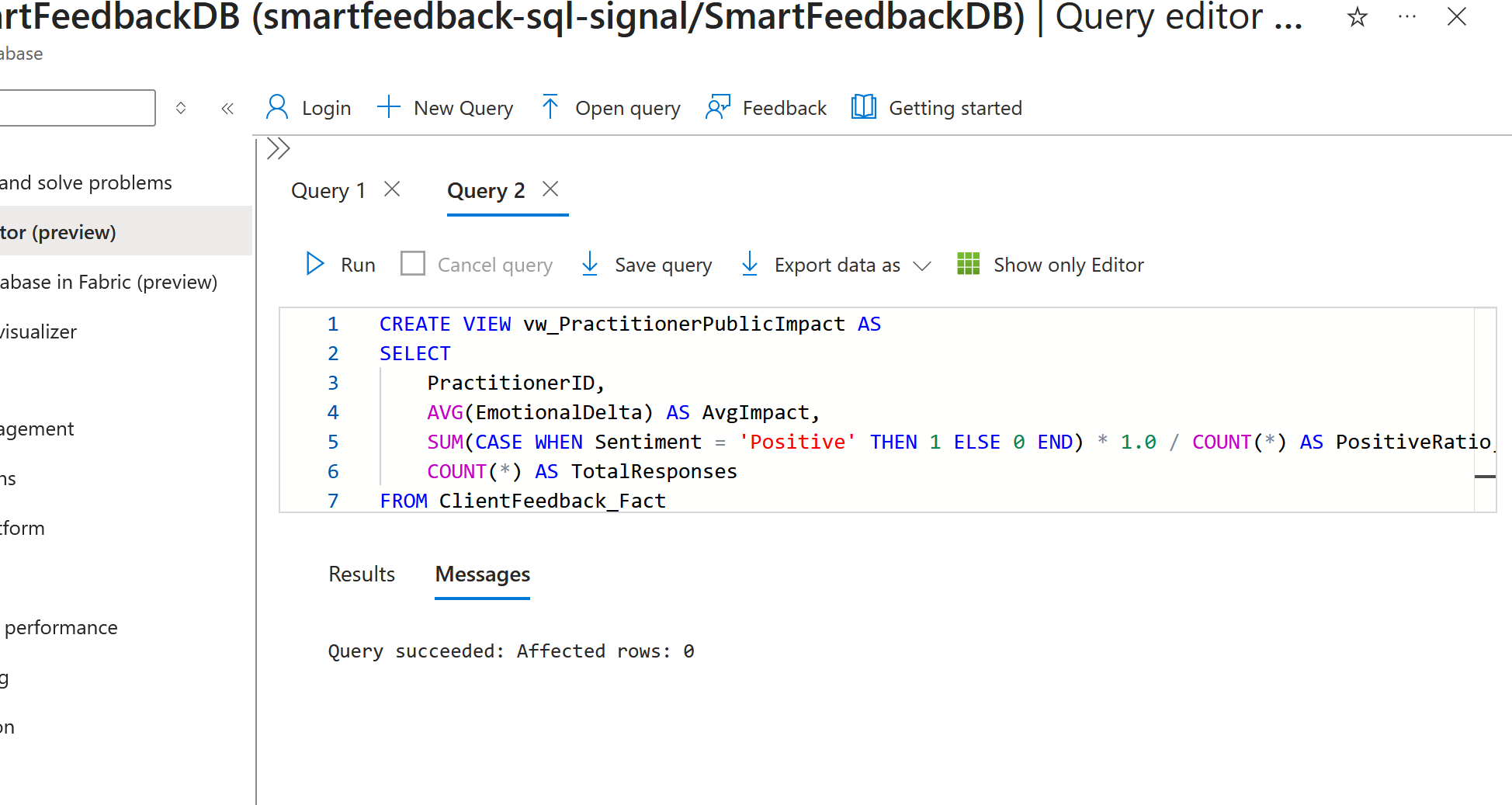


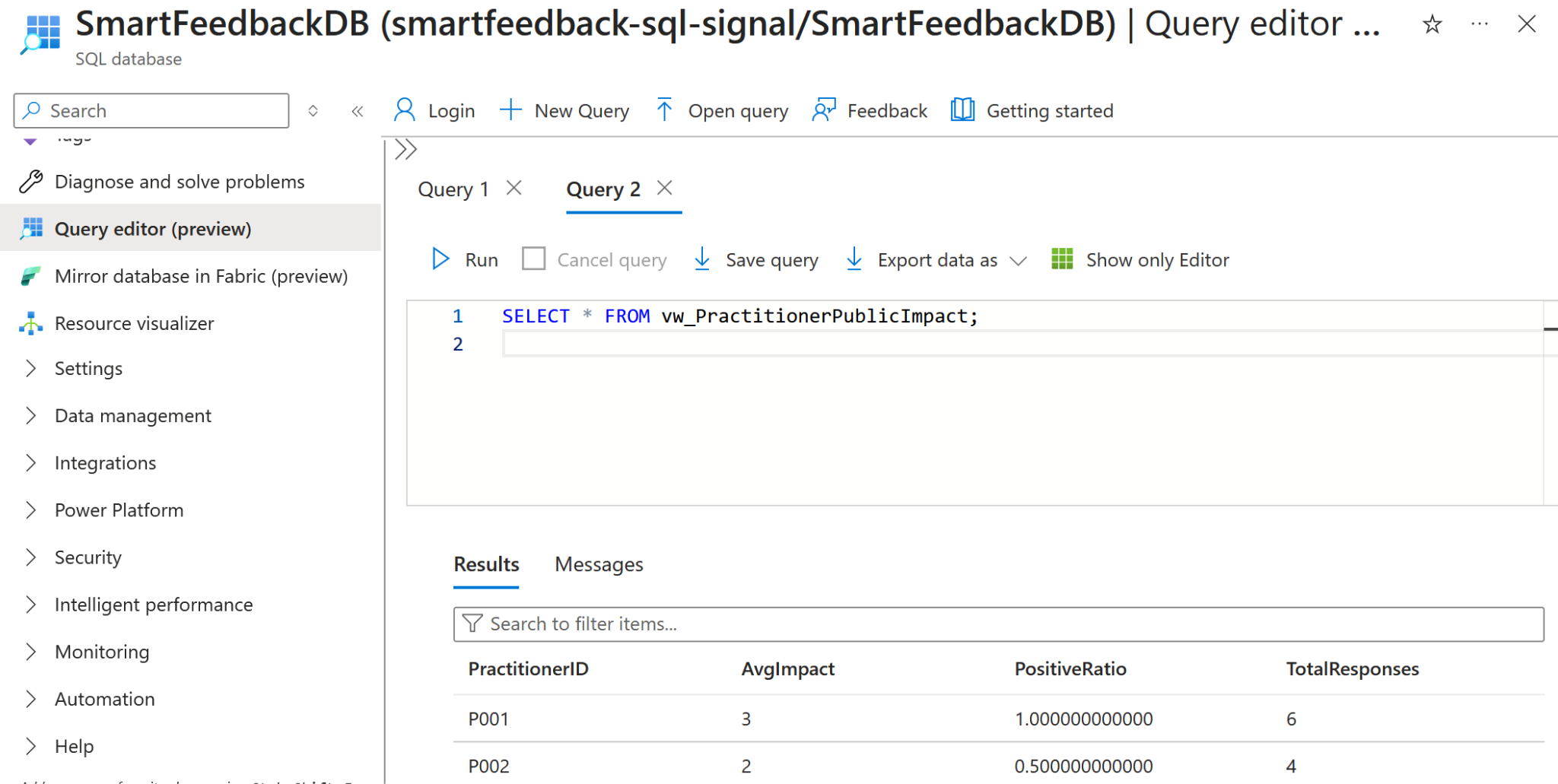
This shows average emotional change per practitioner per week.





Shows how many Positive / Neutral / Negative responses each practitioner has.





This shows the public profiles,PDF impact reports and positive ratio of each id.

**WEEK 3 Web App Layer, Public Impact Page,QR Codes**

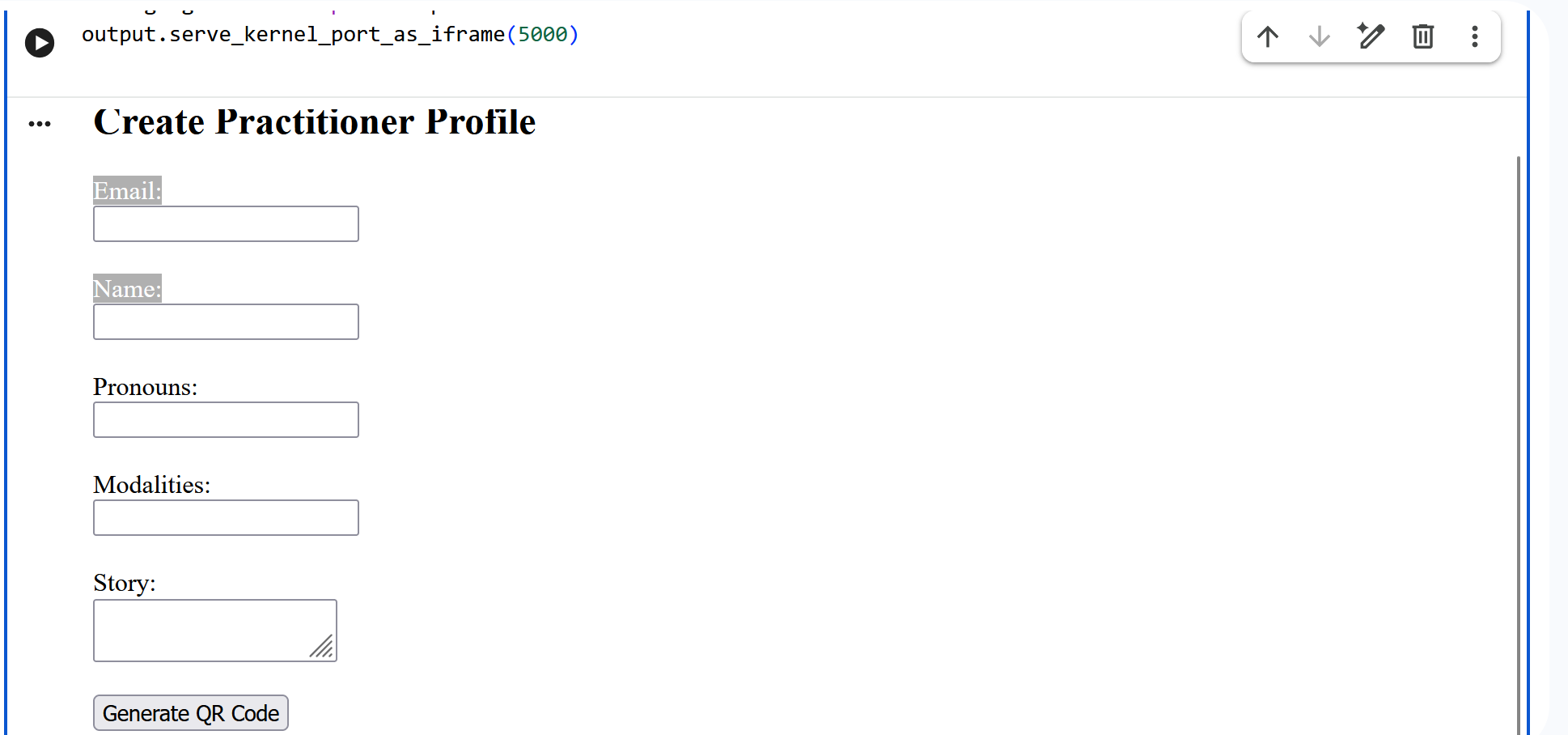
Week-3 focuses on converting the analytics pipeline into a human-usable platform. The goal was to allow wellness practitioners to collect feedback via QR codes and show transparent impact dashboards to the public. So basically it will pull the views from the azure sql database and turn it into insights here.

1. Build the Micro Web App (Lightweight Framework)

A Flask-based micro web application was built to allow wellness practitioners to Register with email,Create a public profile,Generate a unique QR code,Receive anonymous client feedbackThe form collects:Name,Pronouns,Wellness modalities  
Personal story.

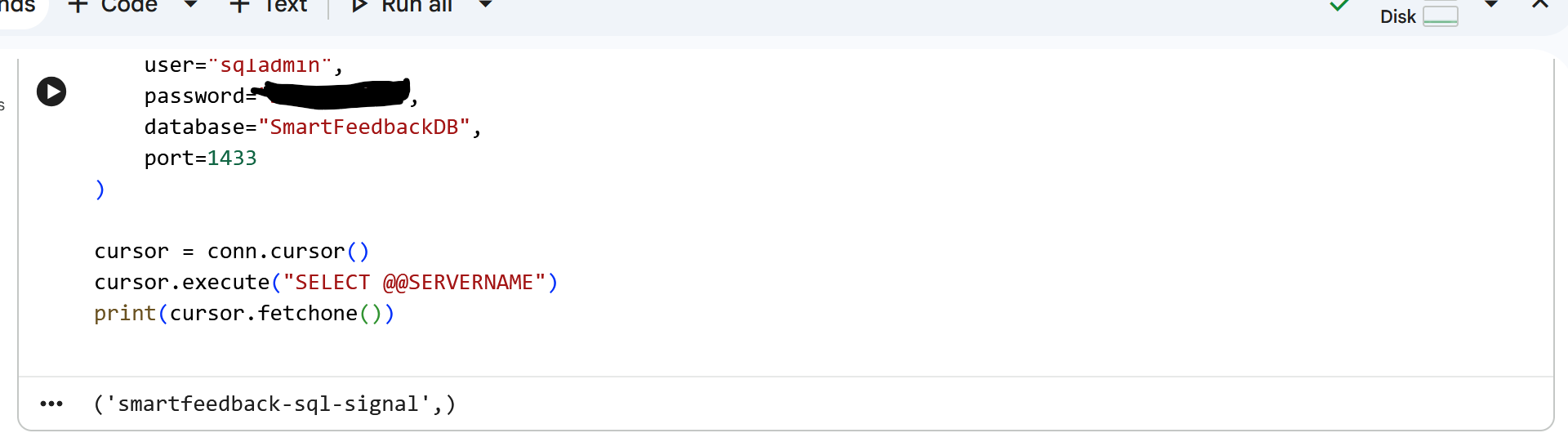
Each practitioner is assigned a unique practitioner id which becomes their permanent public link.

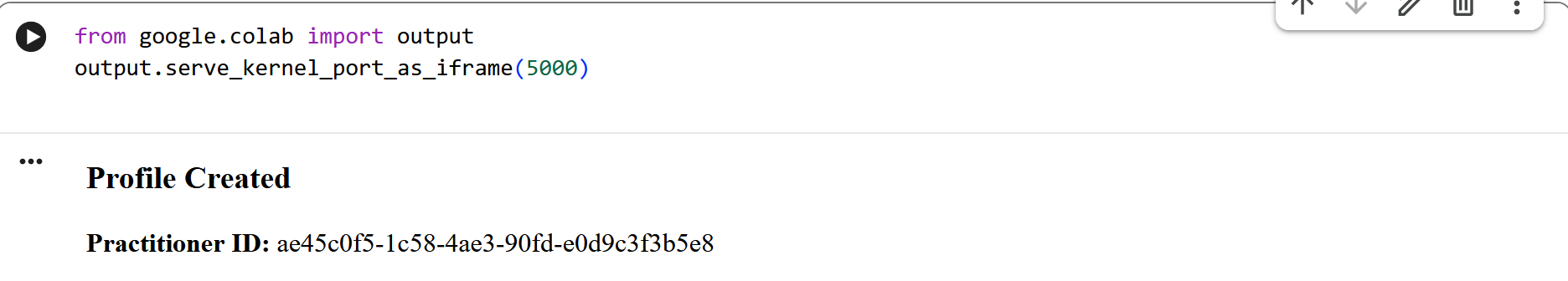
Below is the screenshot of the “Create Practitioner Profile” form.with QR code generation button.



2. Connect Web App → Azure SQL

After creating a profile, a **QR code** is generated then QR code links to Clients scanning the QR code are taken to a feedback form where they submit Session ID,Question (Safety / Connection / Clarity),Emotion before,Emotion after,Comment.

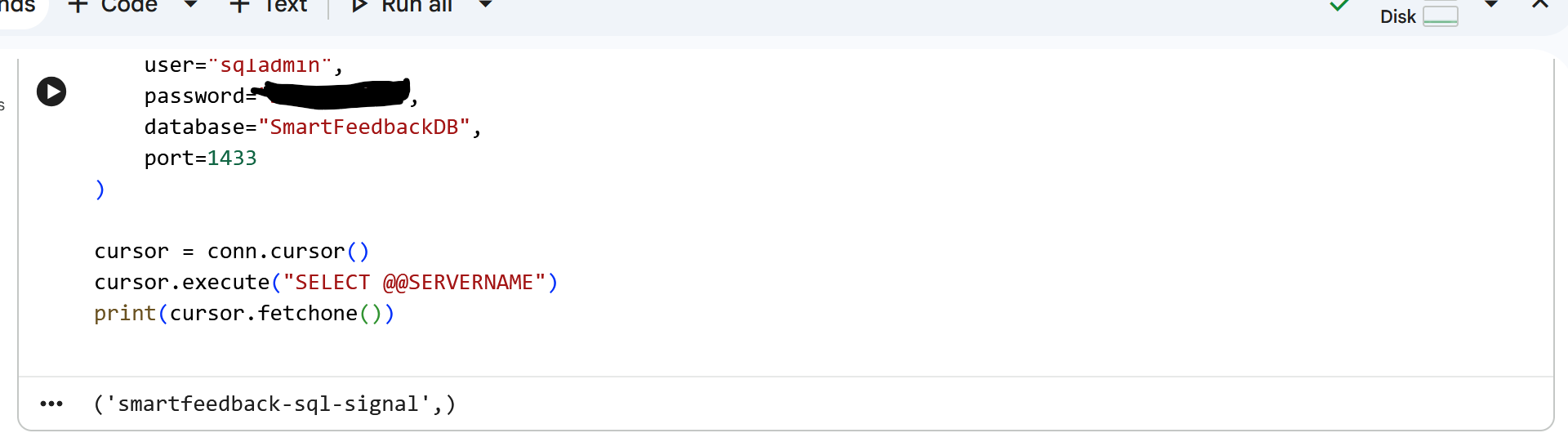


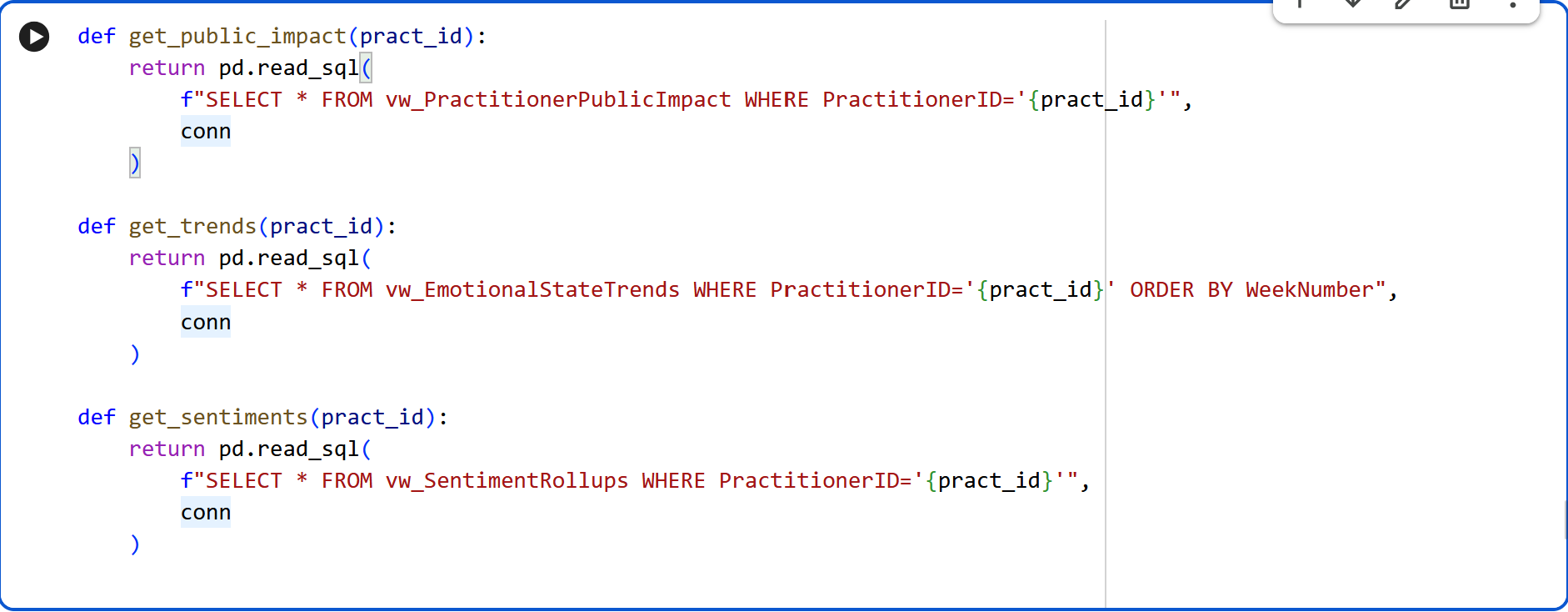




3. Build Public Practitioner Impact Page

The Azure database was connected and the reporting views were gathered from there for the analytics impact page.



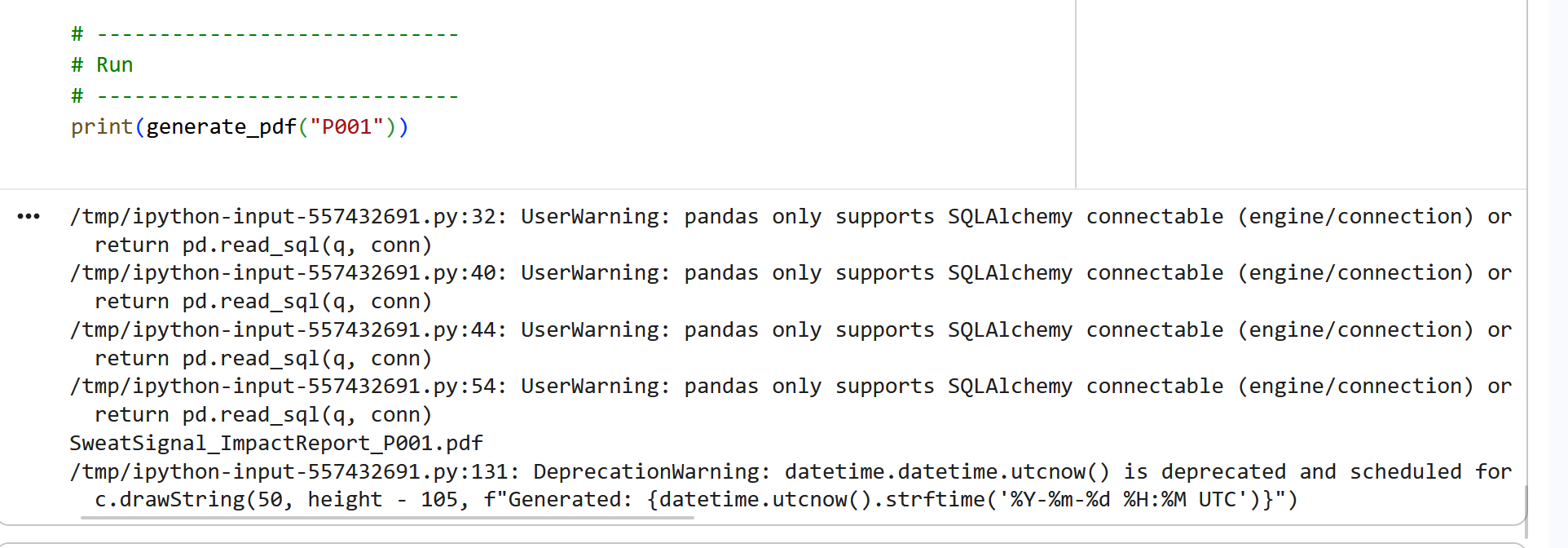


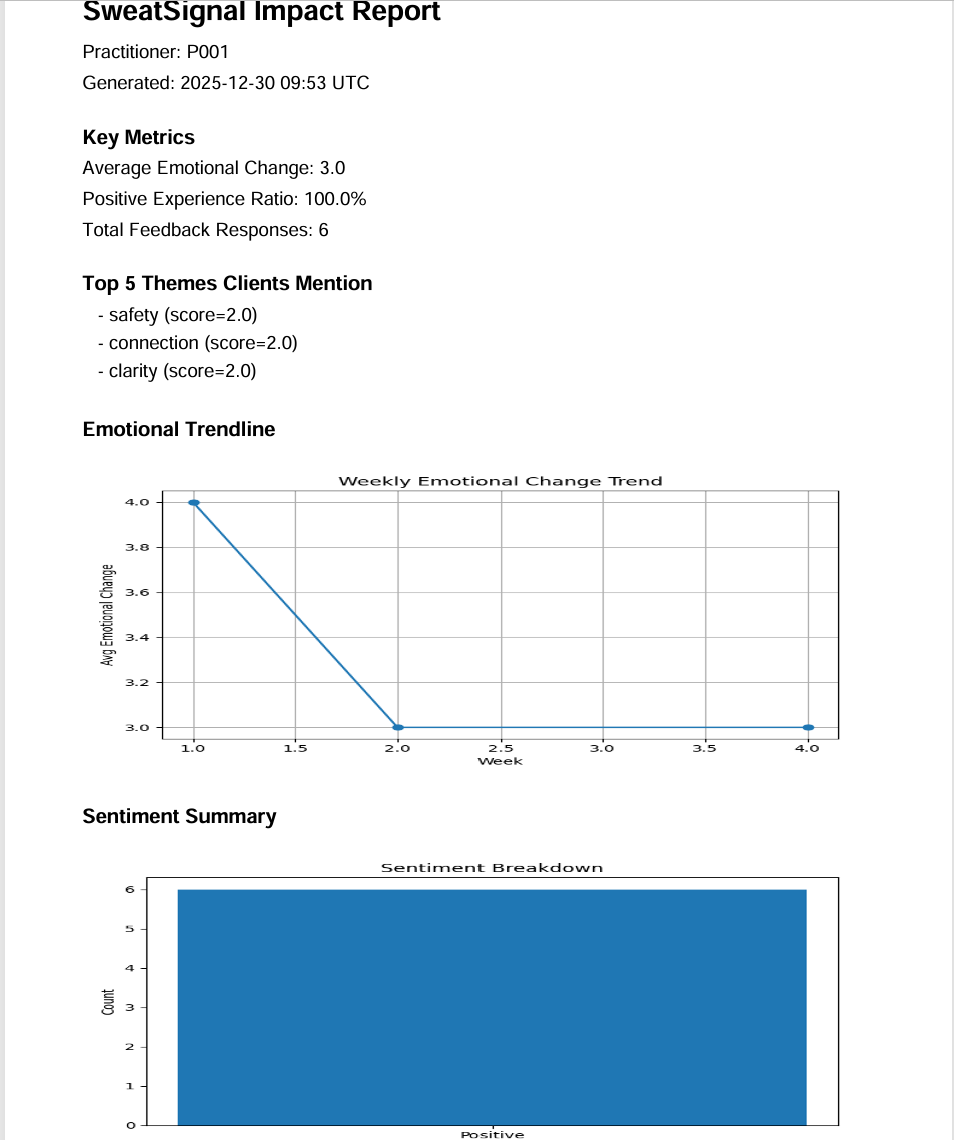
In conclusion I have built a full web-based feedback system with QR codes, practitioner profiles, anonymous submission pipelines, and Azure-connected ingestion. with full SQL-backed analytics ready for rendering in the next iteration.

**WEEK 4 — PDF Reporting Engine,App Polishing,Documentation**

Built an automated PDF Impact Report Generator in Python that pulls practitioner and session insights directly from SQL, generates branded matplotlib charts, and compiles a clear narrative summary of outcomes. The report includes a sentiment summary, an emotional trendline, and a “Top 5 themes clients mention about you” section, then exports a consistent pdf named a **SweatSignal\_ImpactReport\_PractitionerName.pdf**.

Below are the screenshots of the pdf generated and reports in it.

****

****