Microsoft

T-SQL → Spark SQL & PySpark Code Converter Tool

Individual Intern Project, Summer 2024

Contributors:
Ray Wang *with grateful guidance from*
Numair Ul-Ghani, Brian Lui

July 22nd, 2024

Context

For Microsoft Internally and Our Customers

Problem

How can we rapidly enable the conversion of T-SQL into PySpark and Spark SQL to **save time** and **ensure correctness?**

Opportunity

Can we better equip team members with custom tools using LLMs like gpt-4o to automate this process?

"Helping customers quickly convert their T-SQL is a capability we'd love to offer!" – TSP/CSA Data & AI

Key Outcomes	Less friction in migrations	Lower Spark SQL & PySpark barrier	Increased uptake of Fabric/Databricks

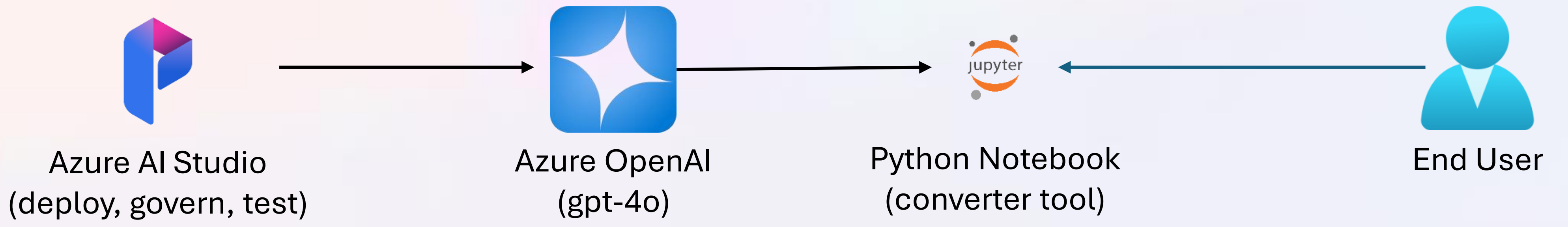
Conversion Steps and Customized Components

How is this different from plain old gpt-4o?

Code blocks in a shareable Jupyter Notebook:

1. Install dependencies and import environment variables
2. Ingest T-SQL code
   A. Supports single string paste and bulk .sql file. formats
3. Validate T-SQL for correctness
4. List dependencies on tables, views, and other database objects
5. Define custom conversion logic (tuning for Databricks / Fabric)
   A. Few-shot prompting and custom context reminders of syntactical differences
   B. Detailed, guard railed system message with temperature set to 0.1
6. Translation to Spark SQL
   A. Works best for basic T-SQL / dialect conversion for DDLs, etc.
7. Translation to PySpark
   A. Works best for stored procedures / complex logic)

Technical Resources and Requirements



Azure:

- Active Azure subscription and Azure OpenAI resource with deployed gpt-4o model, and sufficient tokens-per-minute provisioned to avoid rate limiting (recommended: 130K TPM)

Local computer-side:

- Visual Studio Code (or another IDE with Jupyter Notebook support)
- Python 3.11
- Git Bash to pull the repository (optional, can also download as .zip)
  - .ipynb notebook that can execute in any Jupyter environment
  - .env file to enable easy linking with Azure resources (template included)

Benchmarking

	Key Constraints	Integrability	Accuracy and Performance
This Converter (custom gpt-4o)	**Context length: 128K tokens** (*supports many stored procs)*	Supports **multiple statements** & **.sql files**	1st / Best (output requires no debugging most of the time)
ChatGPT Plus (gpt-4o)	**Context length: 4K tokens** at a time *(prone to truncation)*	Requires manual pasting in of SQL extracts	2nd / Okay (output valid some of the time, data not secured)
Databricks Assistant (gpt-4)	**Context length: 500 tokens** *(usually stops part-way during periods of high demand)*	Automatically debugs and can offer conversions	2nd / Okay (helpful for debugging, but insufficient for conversion)
Copilot (gpt-4o)	**Context length: ~2K chars** (script usually doesn't fit)	Requires manual pasting in of SQL extracts	3rd / Poor (lazily truncates logic)

Microsoft

Success Story: Major Public Transit Agency

Context

Agency is migrating from Synapse to Databricks with CSA assistance.

Action

Converted DDLs, DMLs, and Stored Procs to Spark SQL and PySpark with the code converter and Databricks Assistant debugging

Result

Completed main ETL procedures in 4 weeks with just 1 CSA + intern

Q&A