GEORGETOWN UNIVERSITY
The Graduate School *of* Arts & Sciences
Master of Science in Data Science & Analytics

DataMorph: Data Engineering AI Agent

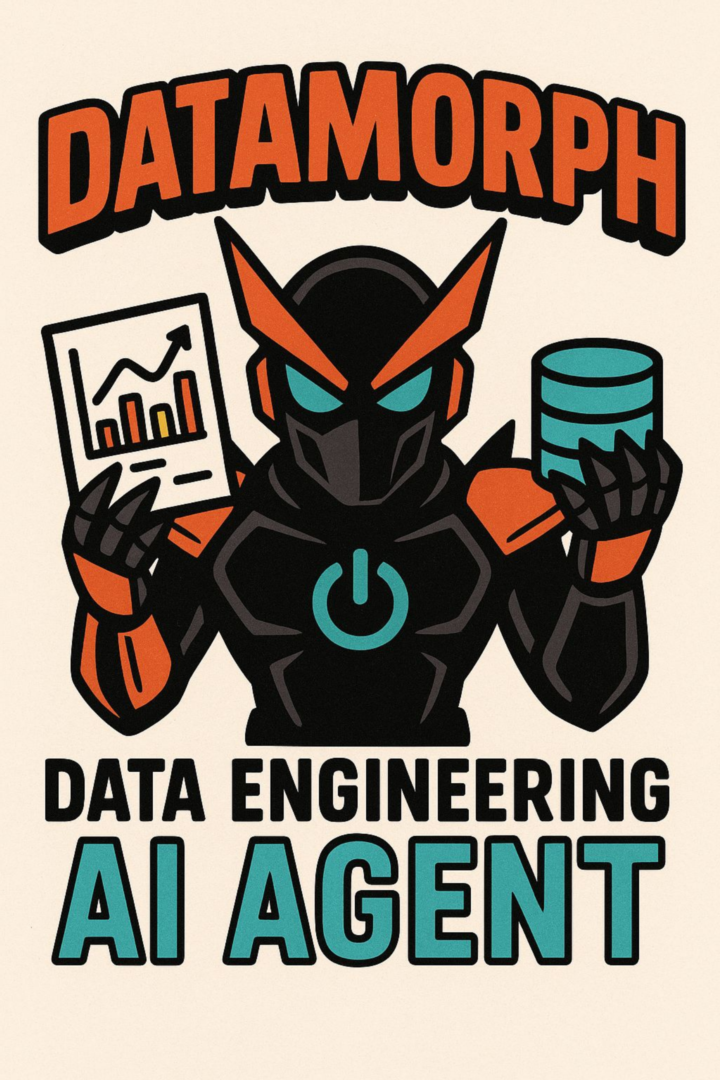
Location : Edward B. Bunn S.J. Intercultural Center 107

04/29/2025 at 11:30 AM

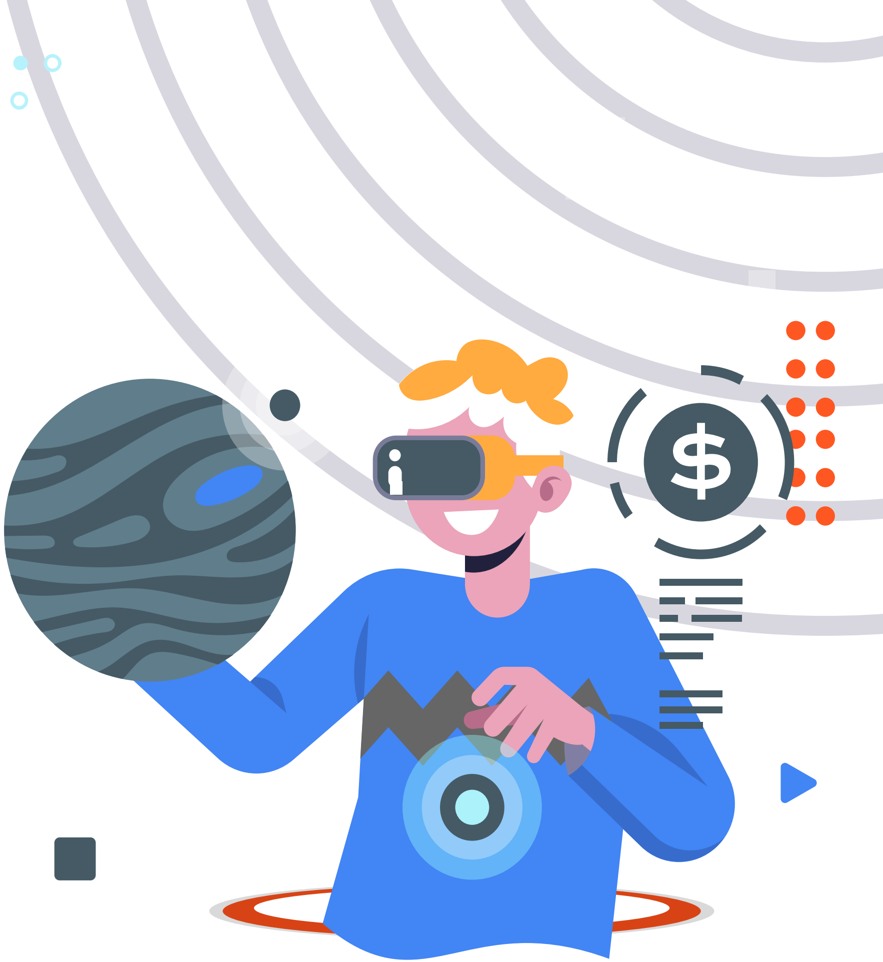
Team No. 15
- **Sai Prerana Mandalika**
- **Pranav Patil**
- **Zenan Wang**

Contents

1. Understanding ETL and its Challenges
2. Motivation: Why GenAI?
3. Our Approach: ETL with GenAI
4. Solution Architecture: Agents, RAG, Prompts
5. Demo: Streamlit App
6. Results and Metrics
7. Conclusion: Solving ETL with DataMorph

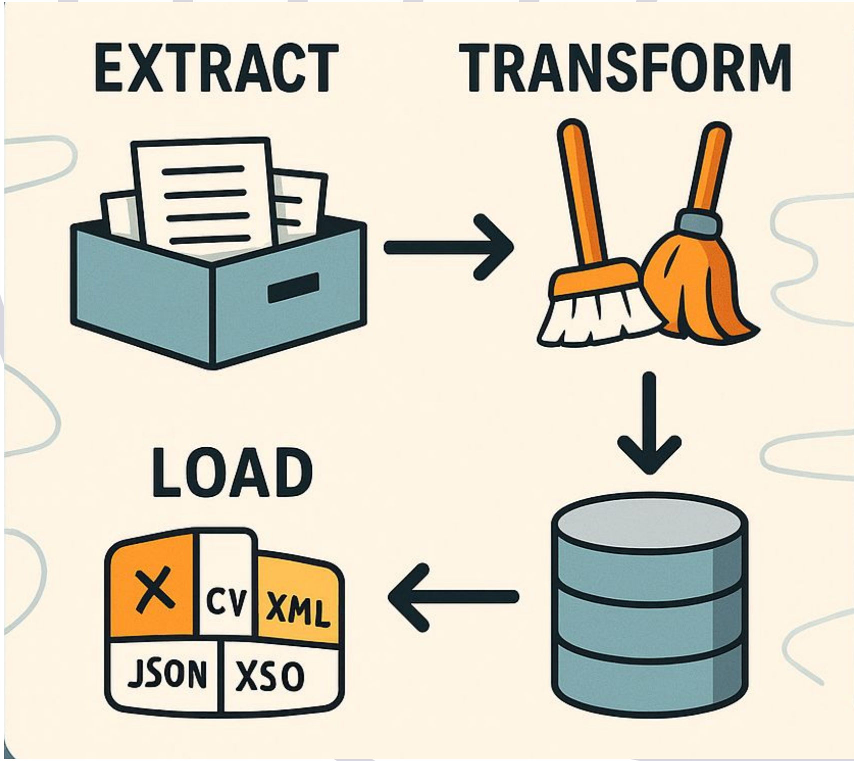


Understanding ETL and its Challenges

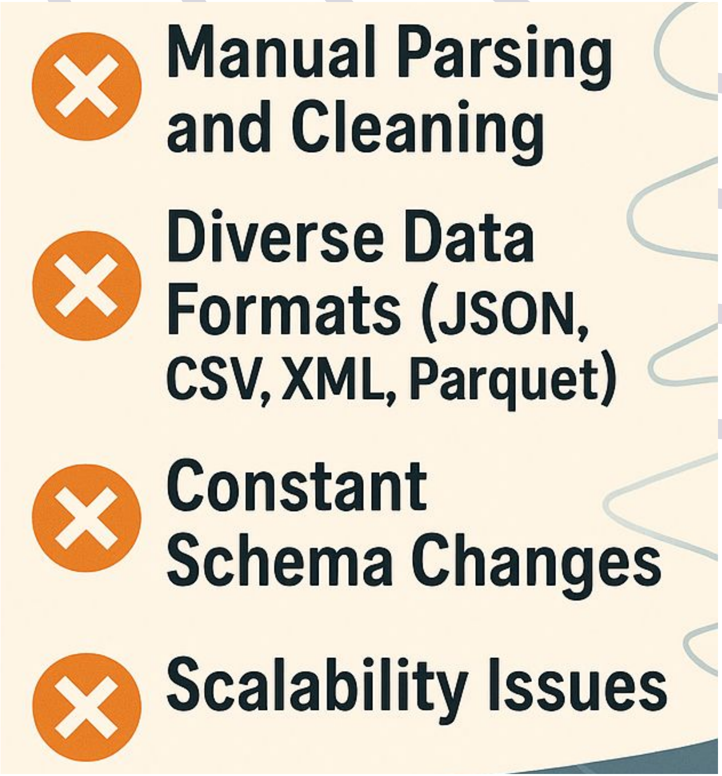


What is ETL?

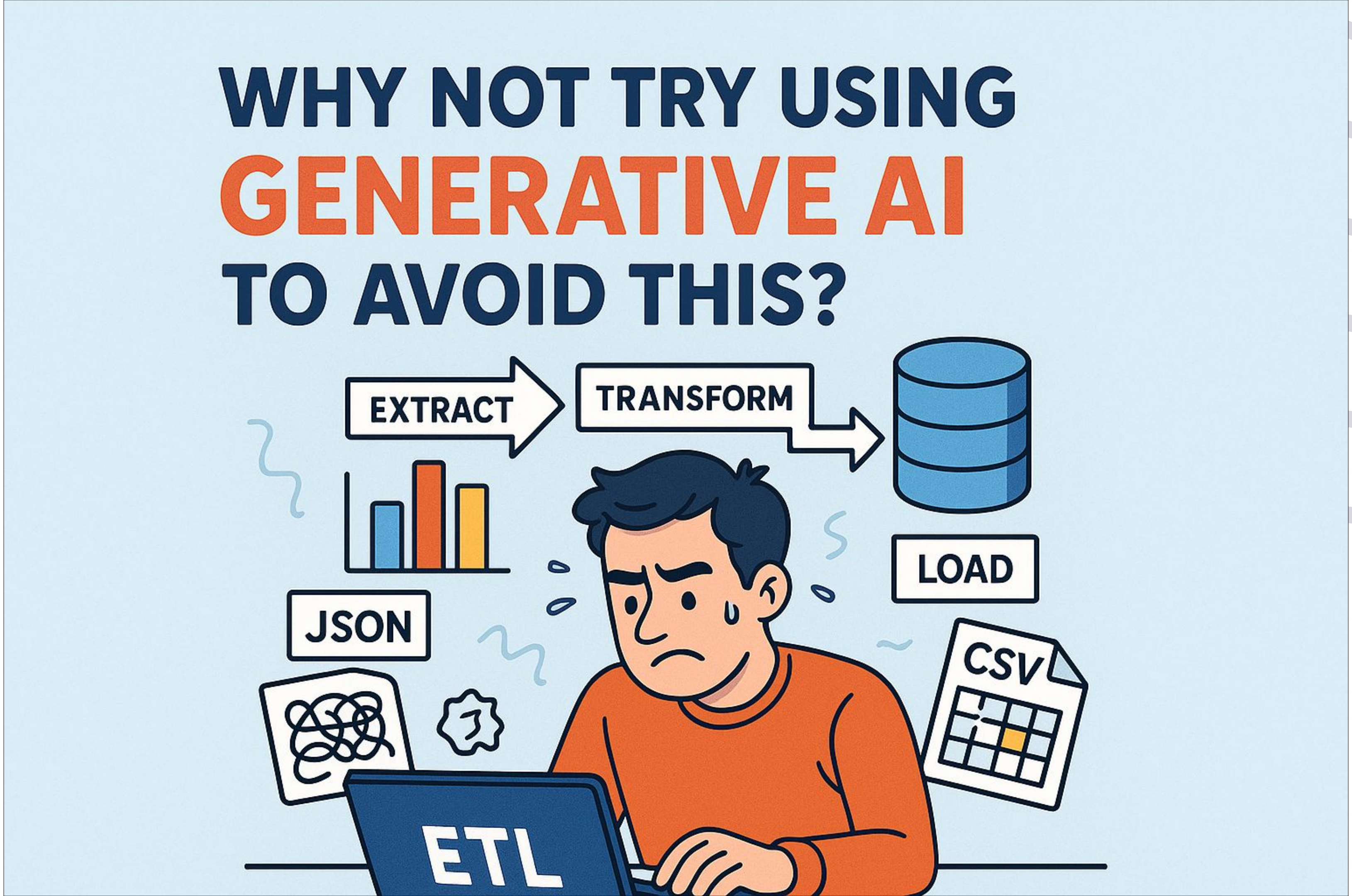
- **Extract: Pulling raw data from multiple sources (databases, APIs, files, etc.)**
- **Transform: Cleaning, standardizing, and reshaping the data**
- **Load: Storing the processed data into a target system (like a data warehouse)**



ETL is essential — but traditional methods are breaking under modern data needs.



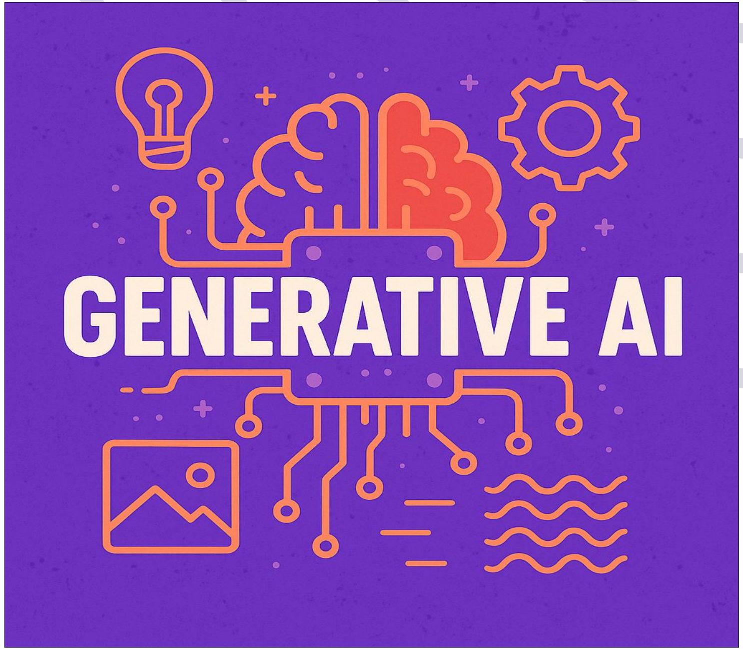
- ☒ Manual Parsing and Cleaning
- ☒ Diverse Data Formats (JSON, CSV, XML, Parquet)
- ☒ Constant Schema Changes
- ☒ Scalability Issues

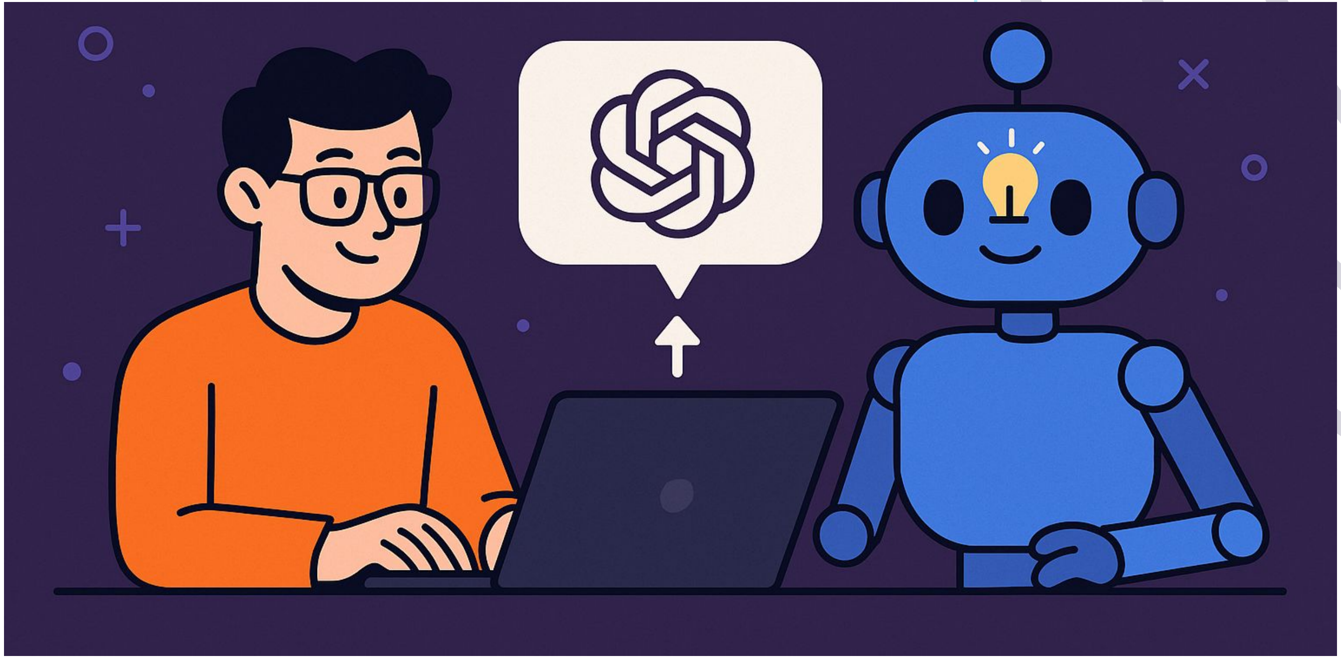


WHY NOT TRY USING GENERATIVE AI TO AVOID THIS?

Why GenAI for Data Engineering?

- Dynamic Schema Detection
- Automated Code Generation
- Intelligent Validation and Transformation
- Learning from Patterns
- Scalability and Efficiency

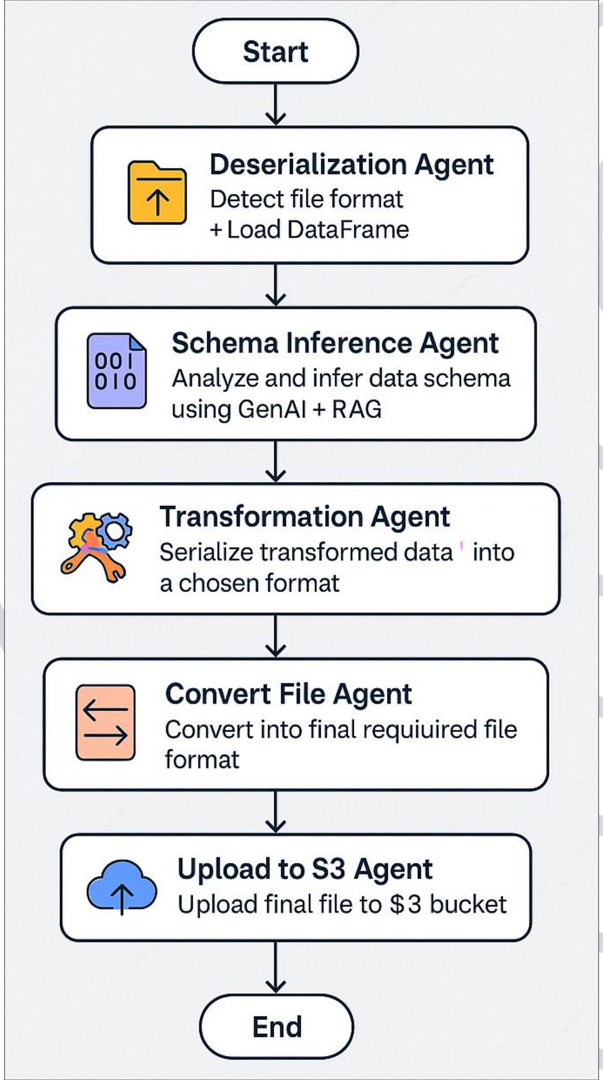




How did we use Generative AI to achieve this?

Automating the Entire ETL Pipeline — from File Detection to Final Storage — with GenAI Intelligence.

Workflow of our proposed method



Start

Deserialization Agent
Detect file format
+ Load DataFrame

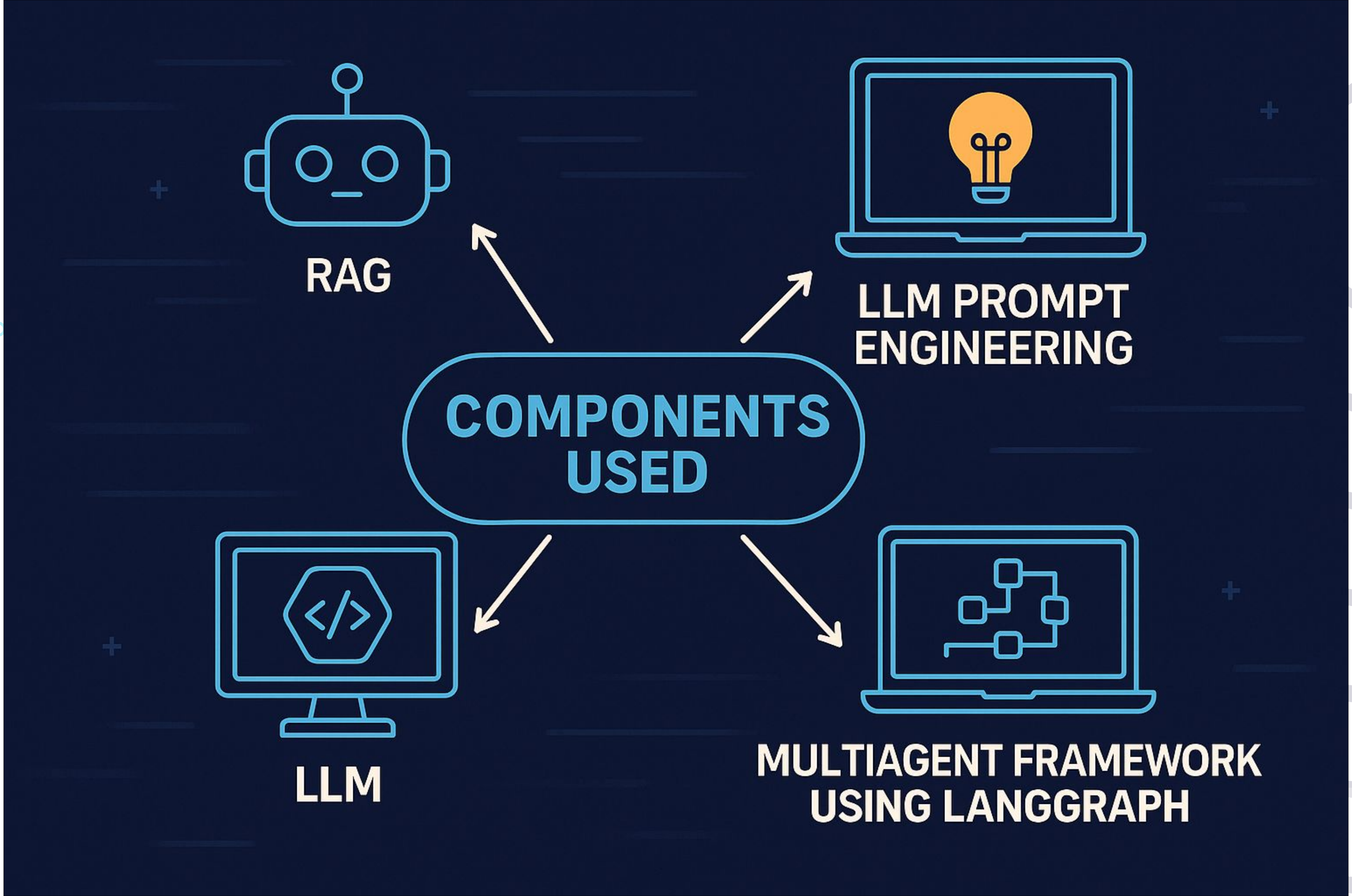
Schema Inference Agent
Analyze and infer data schema
using GenAI + RAG

Transformation Agent
Serialize transformed data ' into
a chosen format

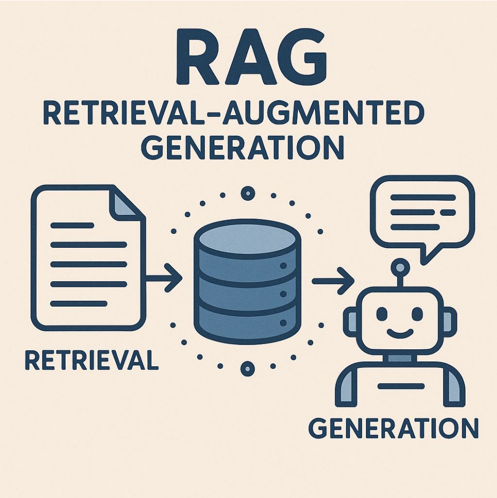
Convert File Agent
Convert into final requiuired file
format

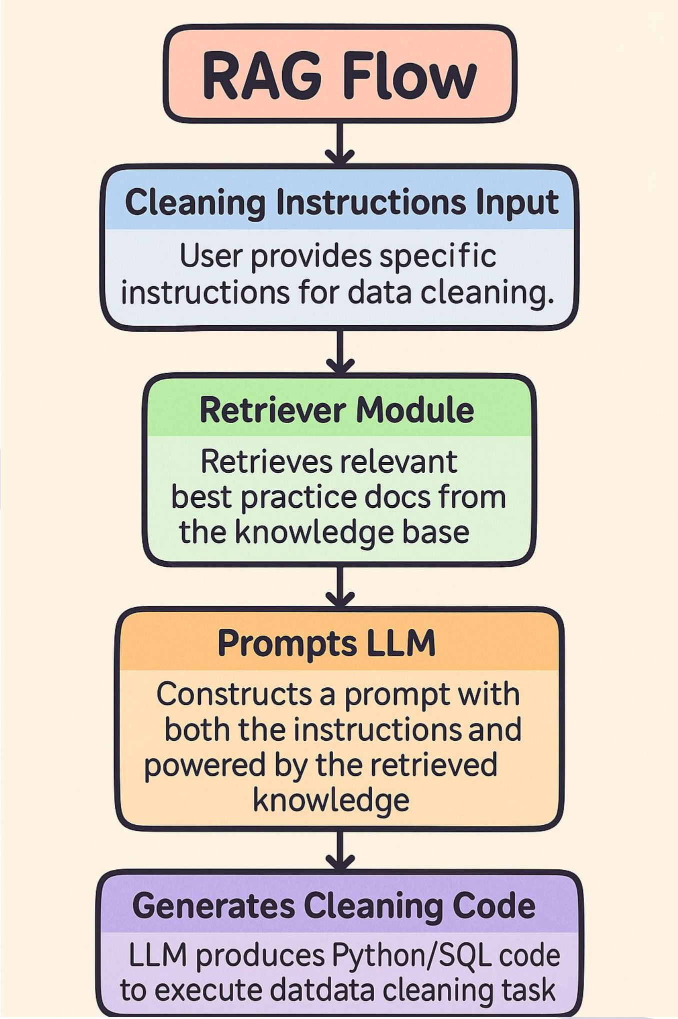
Upload to S3 Agent
Upload final file to $3 bucket

End


RAG
LLM PROMPT ENGINEERING
COMPONENTS USED
LLM
MULTIAGENT FRAMEWORK USING LANGGRAPH

Workflow of our RAG Implementation





RAG Flow

Cleaning Instructions Input

User provides specific instructions for data cleaning.

Retriever Module

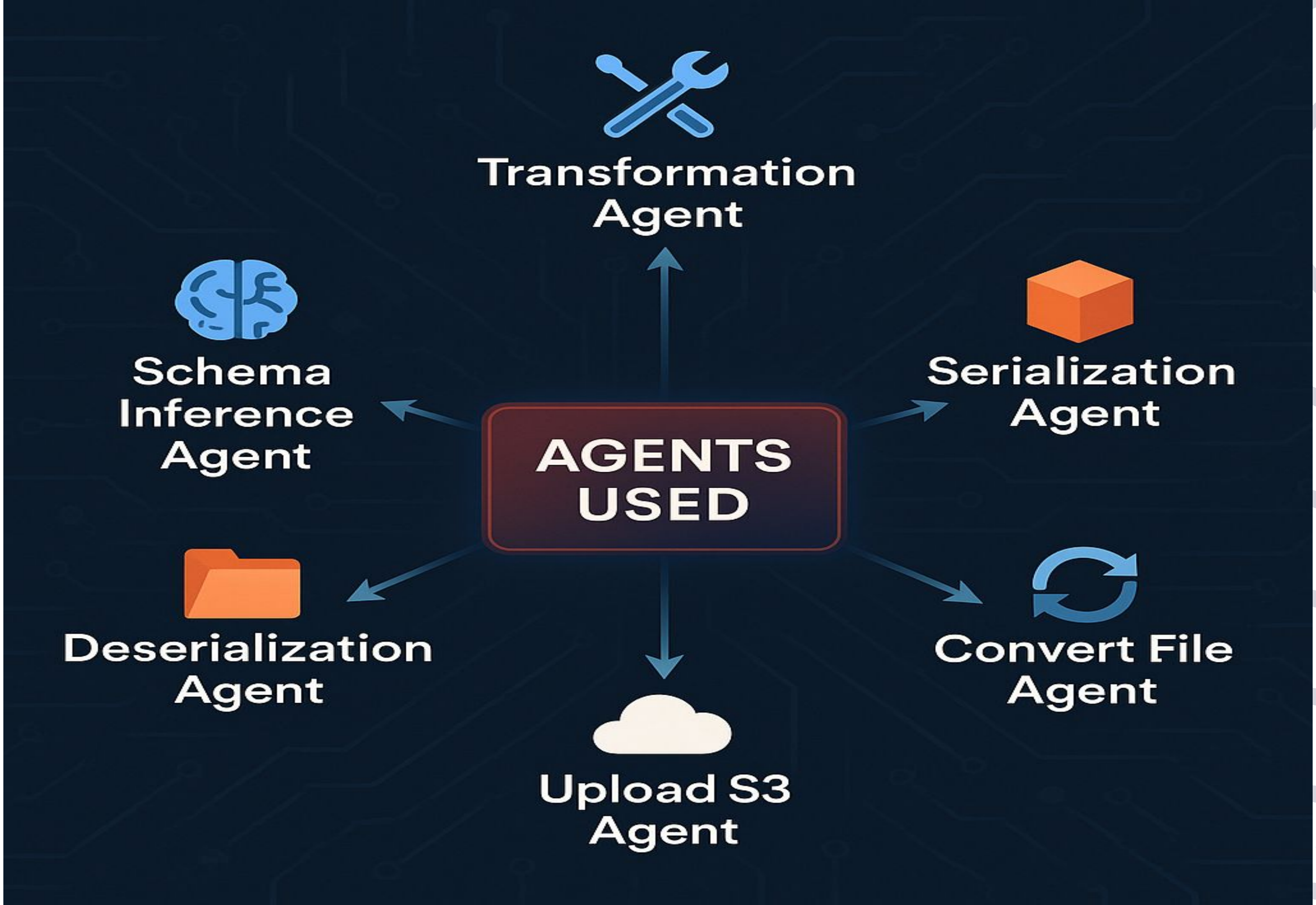
Retrieves relevant best practice docs from the knowledge base

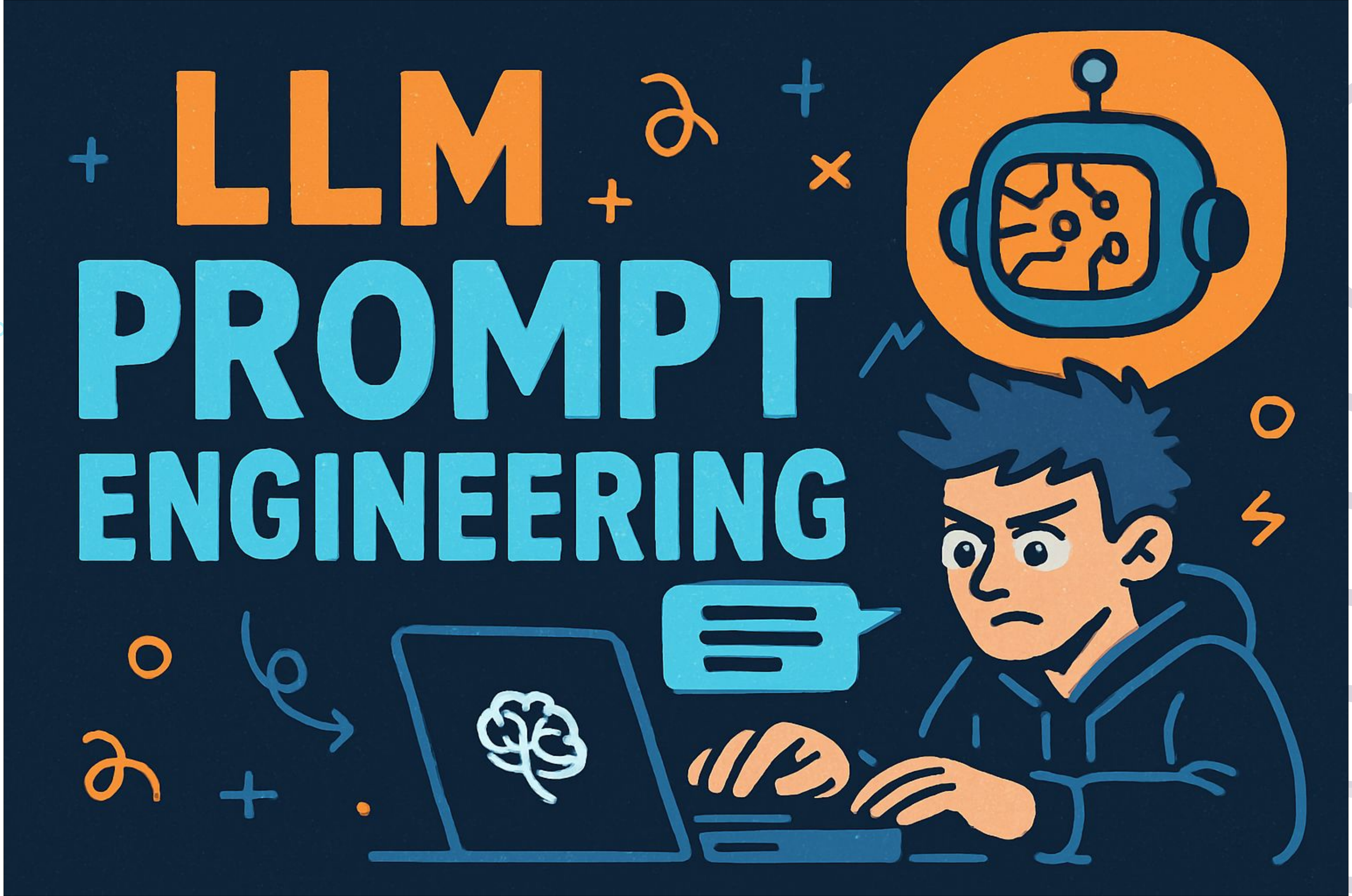
Prompts LLM

Constructs a prompt with both the instructions and powered by the retrieved knowledge

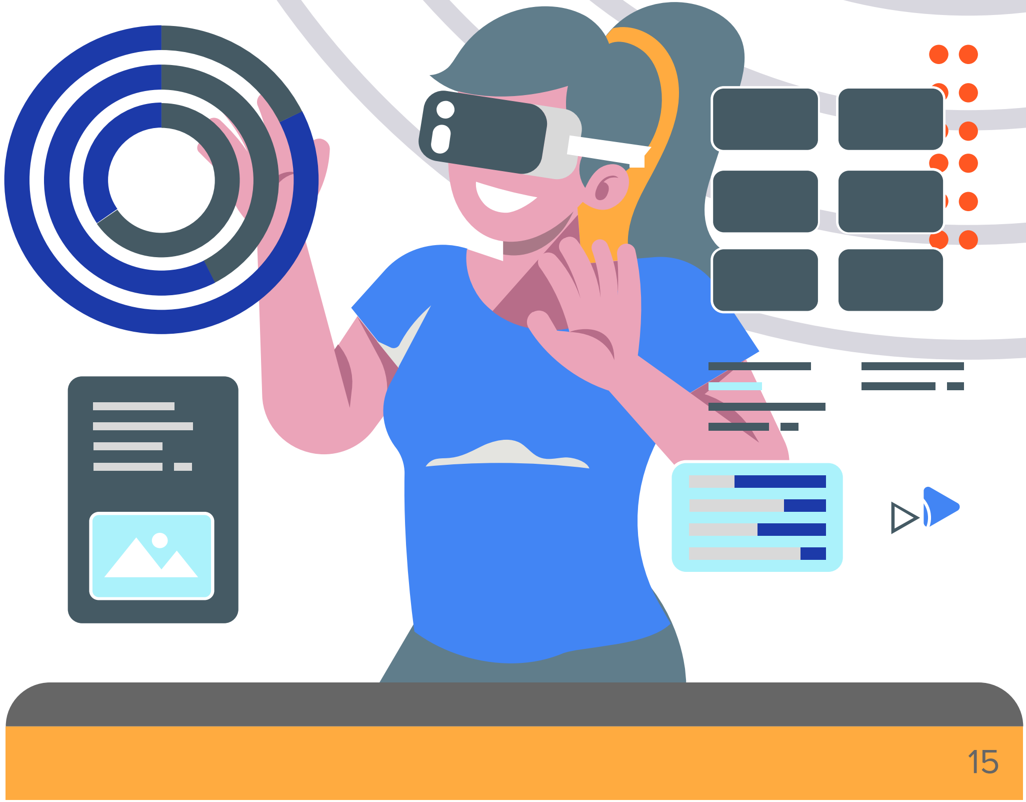
Generates Cleaning Code

LLM produces Python/SQL code to execute datdata cleaning task


AGENTS USED
Transformation Agent
Schema Inference Agent
Serialization Agent
Deserialization Agent
Convert File Agent
Upload S3 Agent


LLM
PROMPT
ENGINEERING

DEMO



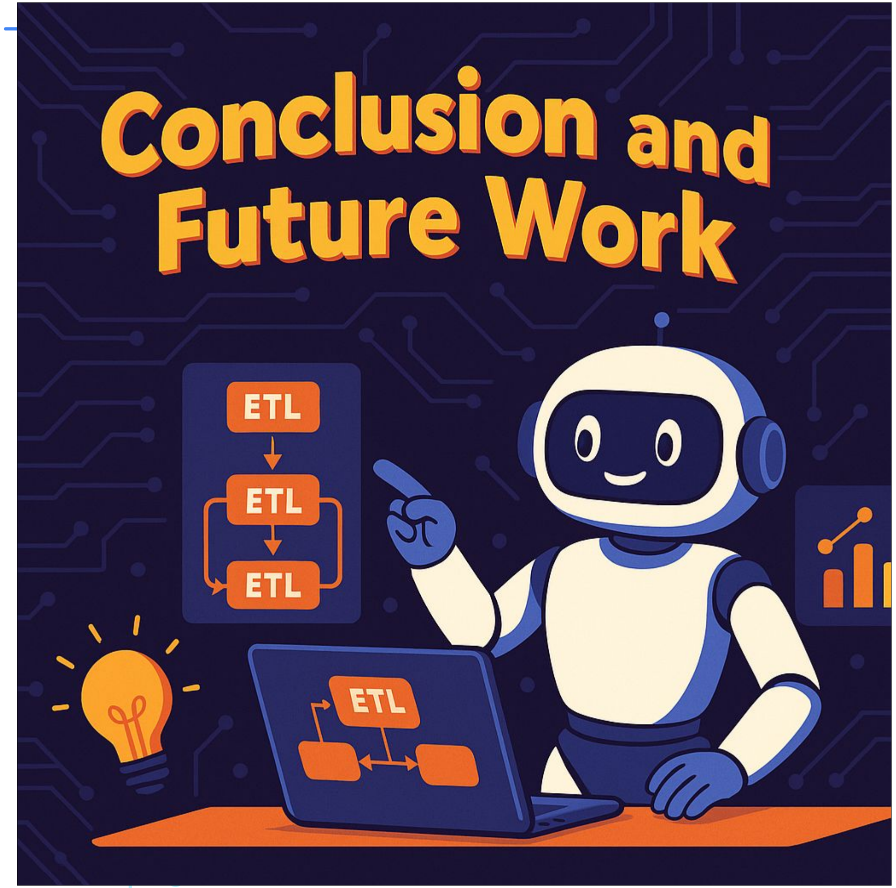
EVALUATION METRICS

Evaluation Metrics

Faithfulness as evaluation metric? Not that accurate.

So..

- **To check the performance, we tried for 50 queries and found the average latency for every agent to between 2 to 3 seconds**
- **For the reliability, we tried 50 queries for which 38 of them followed the correct ETL flow, with accurate data processing results.**



- Advanced data analysis and reporting
- Multi file functionality for Merging two data tables, splitting (multi file functionality)
- Comparing two databases.
- Add Gaurdrails for PII masking (if processing real user files).
- Make sure it works better for XML, paraquet and other formats.

References

- https://github.com/langchain-ai/langgraph
- https://medium.com/totalenergies-digital-factory/advancing-data-engineering-with-generative-ai-cb8c6c3b1b1e
- https://www.cognizant.com/us/en/insights/insights-blog/how-gen-ai-will-forever-change-data-engineering-wf1807301
- https://youtu.be/T23Bs75F7ZQ?feature=shared

Thank you!