**DATA ENGINEERING PIPELINE**

# **OBJECTIVE**

To convert the data from JSON files into a tabular structure that is more manageable and can be utilized for various analytical tasks and dashboard creation.

# **POSSIBLE SOLUTIONS**

### **SOLUTION 1 (Fixed Output Schema):**

* Gather requirements from the end user (e.g., Analytics/Dashboard) to determine the specific data needed for their use case.
* Assess the source data to identify the relevant information to be processed and delivered as output, aligning with the user's requirements.
* Develop a Data Cleaning and Extraction strategy to establish a standardized schema for the output dataset, ensuring clarity and usability for the end user.
* Implement a robust data pipeline script to efficiently process and deliver the required data to the end user.
* Determine the frequency and scheduling mechanism for running the pipeline, and deploy it with necessary dependencies in the designated environment.

**SOLUTION 2 (EAV Data Model):**

* Utilize the "Entity – Attribute – Value" Data Model to encompass all potential attributes within the JSON data format, facilitating the transformation of diverse data into a tabular structure.
* This model is particularly suitable for scenarios where data attributes are undefined and subject to frequent additions.
* Conduct a thorough analysis of the data to determine the appropriate mapping of entities and attributes for tabular representation.
* Devise a Data Extraction approach to accurately extract the necessary values and attributes from each source file.
* Develop a robust data pipeline script to effectively process and convert all available data fields and values into a structured tabular format.
* Determine the frequency and scheduling mechanism for running the pipeline, and deploy it with necessary dependencies in the designated environment.

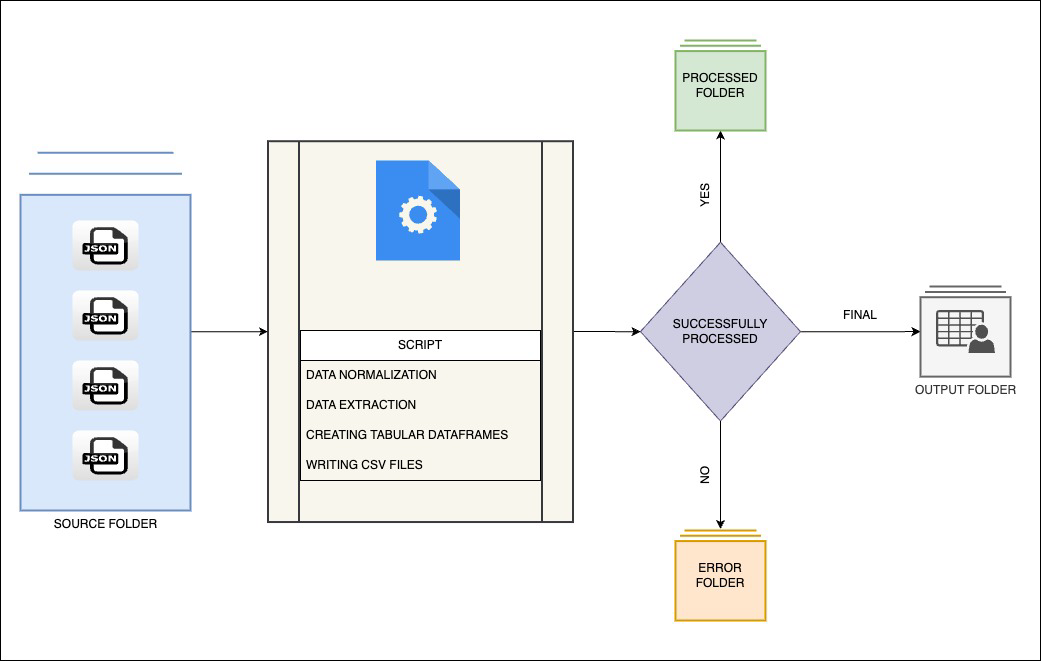
# **SELECTED SOLUTION**

**Solution 2 (EAV DATA MODEL)**

**Reasons:**

* The specific use case and required data are not predefined.
* Facilitates future scalability by allowing for the addition of new attributes without altering the underlying data model.
* Provides adaptability to evolving data requirements and changing business needs.
* Supports a wide range of data types and structures, ensuring compatibility with diverse datasets.
* Enhances data exploration and discovery by enabling the dynamic selection and analysis of attributes.
* Enables efficient storage and retrieval of sparse or irregular data, optimizing resource utilization.
* Facilitates data governance and management by maintaining a centralized repository of all attributes and values.
* Promotes consistency and standardization across different data sources and domains.

# **HIGH LEVEL ARCHITECTURE DESIGN**

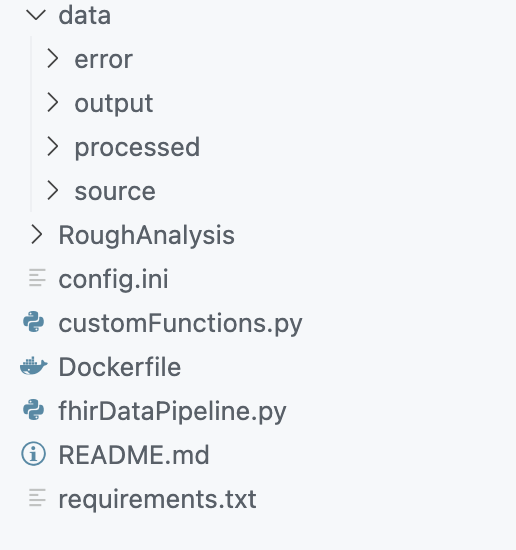


# **COMPONENTS**

**Script**

* Developed in Python programming language.
* Incorporates fundamental data analysis libraries like NumPy, Pandas, and JSON.
* Displays progress through TQDM.
* Implements exception handling.
* Logging functionality is activated.
* Includes custom functions for data normalization.
* Configurable via a config file.

**Folder Structure**

****

**Key Takeaways**

**Data Volume Concerns**

* Storing output data as a single CSV file is not advisable due to its excessive size.
* Managing such large files can be challenging and inefficient.

**EAV Model Challenges**

* The Entity-Attribute-Value (EAV) model results in an abundance of attributes for a single entity.
* This can complicate querying and extracting the necessary data.

**Trigger-Based Pipeline**

* To minimize delays, the pipeline should be triggered upon source file arrival rather than running in batch mode.

**Analytical Database for Output**

* Consider saving output data in an analytical database like Google Big Query.
* This enables flexible querying and enhances data visualization capabilities.

**Further Enhancements**

**Selective Data Fields**

* Discuss and identify the essential data fields needed for visualization.
* Extract and store only necessary data for deeper analysis.

**Reconsider EAV for Visualization**

* Evaluate whether the EAV model is suitable for data visualization purposes.
* Alternative approaches may be more effective.

**Structured Schema**

* Develop a well-defined schema to organize data in a tabular format.
* Avoid dumping all fields and values into a few cluttered columns.

**Data Quality**

* Engage in discussions to understand the expected values within the data.
* Identify the permissible range, constraints, and any specific business rules associated with each value.
* Determine the data types (e.g., numeric, string, date) for each attribute.
* Integrate data quality checks into the pipeline.
* Validate data against predefined rules (e.g., uniqueness, referential integrity).
* Detect anomalies, missing values, and outliers.

**Data Governance**

* Establish governance policies (e.g., data ownership, access controls).
* Define data lineage and metadata management.
* Monitor compliance with data standards and regulations.