# **Data Migration and Transformation**

**1. Introduction**

The ETL (Extract, Transform, Load) pipeline documented here is designed to retrieve JSON files from a remote source, process the data contained within these files, and then load the transformed data into a SQL Server database. Additionally, the pipeline includes a step to copy the extracted data to an Azure Data Lake Storage Gen2 for archival and backup purposes. This pipeline aims to automate the extraction and loading process while ensuring data integrity, reliability, and scalability.

**2. Pipeline Components**

The ETL pipeline consists of the following main components:

a. Data Source

* The data source is a remote location hosting JSON files containing structured data.
* In this documentation, we assume the data source is the SEC (U.S. Securities and Exchange Commission) website, which provides financial reports in XBRL format.

b. Extractor

* The extractor component retrieves JSON files from the data source.
* It utilizes HTTP requests to download the files and extract their contents using standard compression formats like ZIP.
* Technologies used: Python requests library for HTTP requests, zipfile library for ZIP file extraction.

c. Transformer

* The transformer component processes the extracted JSON data to prepare it for loading into the target SQL Server database.
* This involves tasks such as cleaning, filtering, restructuring, and enriching the data as needed.
* Technologies used: Pandas library for data manipulation and transformation.

d. Data Lake

* The Data Lake component copies the extracted JSON files to an Azure Data Lake Storage Gen2 for archival and backup purposes.
* This provides a centralized and scalable storage solution for long-term retention of raw data.
* Technologies used: Azure Data Lake Storage Gen2 for storage, Databricks File System (DBFS) for file manipulation.

e. Loader

* The loader component inserts the transformed data into the SQL Server database.
* It establishes a connection to the database and executes SQL commands or uses an ORM (Object-Relational Mapping) library like SQLAlchemy to handle data insertion.
* Technologies used: SQLAlchemy for database connectivity and data insertion.

**3. Pipeline Workflow**

The workflow of the ETL pipeline is as follows:

1. **Setup**:
   * Install necessary dependencies, such as ODBC drivers, Python libraries like SQLAlchemy, and Azure SDK for Python.
   * Configure the environment and ensure access to the data source, Data Lake, and target database.
2. **Data Extraction**:
   * Use HTTP requests to download JSON files from the remote source.
   * Extract the contents of downloaded ZIP files to obtain the JSON data.
3. **Data Transformation**:
   * Read the JSON data and convert it into Pandas DataFrames for processing.
   * Cleanse, transform, and enrich the data to meet the requirements of the target database schema.
   * Handle any missing or invalid data values.
4. **Data Lake Archival**:
   * Copy the extracted JSON files to Azure Data Lake Storage Gen2 for archival and backup.
   * Organize the files in the Data Lake according to a suitable folder structure.
5. **Data Loading**:
   * Establish a connection to the SQL Server database using SQLAlchemy or a similar library.
   * Insert the transformed data into the appropriate tables in the database.
   * Implement strategies for efficient data loading, such as batch processing and bulk inserts.
6. **Testing and Validation**:
   * Perform thorough testing of the pipeline with sample data to ensure correctness and reliability.
   * Validate the loaded data against predefined criteria and business rules.

**4. Conclusion**

The documented ETL pipeline automates the extraction, transformation, loading, and archival of JSON data from a remote source to a SQL Server database and Azure Data Lake Storage Gen2.

**Technologies Used:**

* Databricks: For executing the ETL pipeline within a managed environment.
* Python: For scripting and data manipulation tasks.
* Pandas: For data transformation and manipulation.
* SQLAlchemy: For database connectivity and data insertion.
* Azure Data Lake Storage Gen2: For scalable storage and archival of extracted data.
* Azure Databricks File System (DBFS): For file manipulation within the Databricks environment.
* Python requests library: For HTTP requests to retrieve JSON files.
* zipfile library: For extraction of ZIP files.
* Azure SDK for Python: For interacting with Azure services.
* SQL Server: For storing transformed data.