**Data Pipeline Assignment Report**

**Project: Serverless ETL Pipeline (Dual API to Snowflake)**

**1. Objective**

This report documents the design and execution of a serverless ETL (Extract, Transform, Load) pipeline. The pipeline extracts data from two separate public APIs with different data formats (JSON and CSV), transforms all data into a unified JSON format, stores the results in an AWS S3 bucket, and loads them into a Snowflake data warehouse for analysis.

**2. Data Sources**

This project utilized two distinct public data sources, as defined in the config.json file:

**Source 1**: User Data (JSON)

   API: <https://jsonplaceholder.typicode.com/users>

   Format: JSON

   Description: This API provides a list of 10 fake user objects. The data is already in a well-structured JSON format, including nested objects for address and company. This data was used directly without transformation.

**Source 2**: S&P 500 Company Data (CSV)

   API: <https://raw.githubusercontent.com/datasets/s-and-p-500-companies/main/data/constituents.csv>

   Format: CSV

   Description: This URL provides a raw CSV text file containing a list of S&P 500 companies and their financial details (Symbol, Name, Sector, Market Cap, etc.). This data required an in-memory transformation before storage.

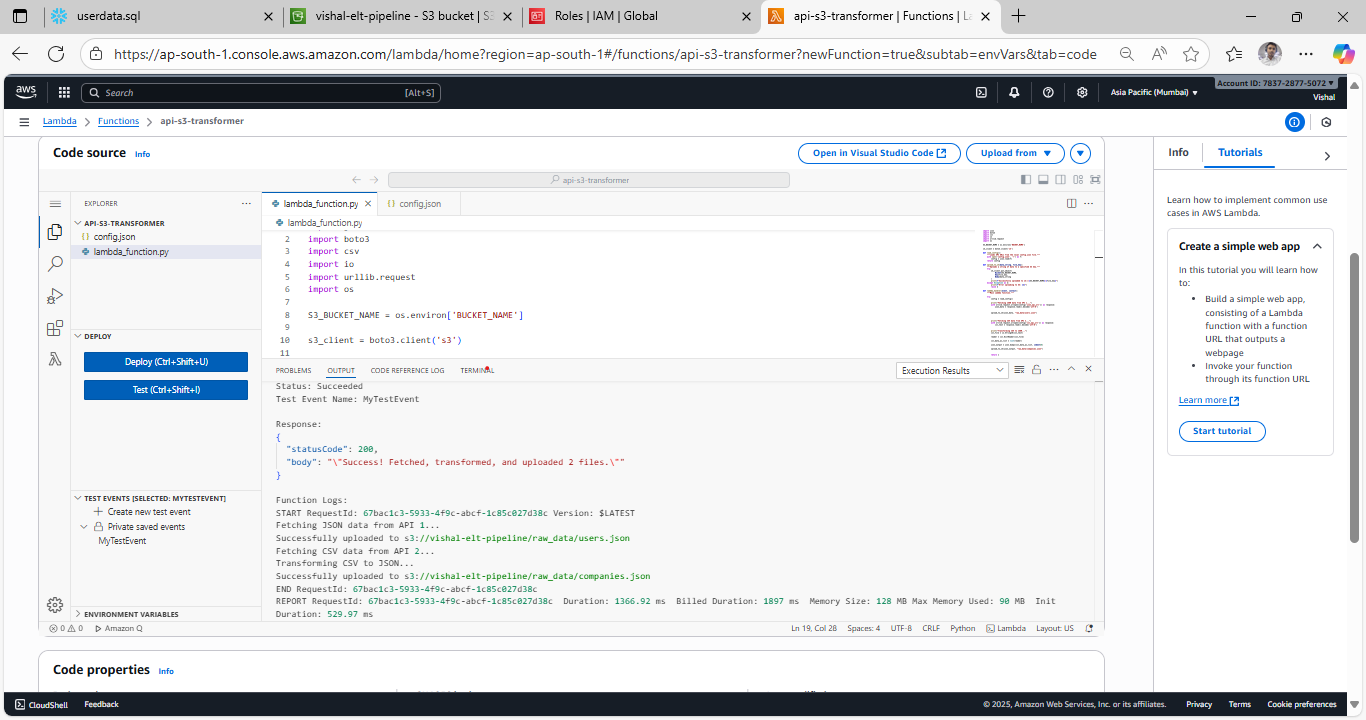
**3. Project Workflow**

The pipeline operates in three main stages, orchestrated by a single AWS Lambda function:

* Extract (AWS Lambda):
* The api-s3-transformer Lambda function is triggered.
* It first reads the config.json file (packaged within the function) to get the URLs for the two data sources.
* Using Python's built-in urllib.request library, it calls both API endpoints and fetches the raw data into memory.
* Transform (AWS Lambda):
* User Data (JSON): The data from API 1 is already in the target JSON format, so no transformation is needed.
* Company Data (CSV): The raw CSV text from API 2 is processed. Python's csv.DictReader is used to read the text as a list of dictionaries (one dictionary per company). The json.dumps function then converts this list into a properly formatted JSON string.
* This step ensures that both datasets are in a unified JSON format before leaving Lambda.
* Load (S3 & Snowflake):
* S3 Storage: The Lambda function uses the boto3 library to upload both JSON objects (users.json and companies.json) to the s3://vishal-elt-pipeline/raw\_data/ bucket.
* Snowflake Load: In Snowflake, COPY INTO commands are used to pull data from S3. A STORAGE INTEGRATION (S3\_LOAD\_INT) securely connects Snowflake to the S3 bucket. A FILE\_FORMAT (JSON\_FILE\_FORMAT) with STRIP\_OUTER\_ARRAY = TRUE is used to correctly parse the JSON lists, loading each user and company as a separate row in the RAW\_USERS and RAW\_COMPANIES tables.
* Snowflake Transform: Finally, SQL VIEWs (VW\_USERS and VW\_COMPANIES) are created in the ANALYTICS schema. These views parse the raw VARIANT data into clean, typed, and queryable columns.

**4. Execution Screenshots**

**Screenshot 1**: AWS Lambda Function Execution



**Screenshot 2**: S3 Bucket Data in JSON Format

A screenshot of a computer

AI-generated content may be incorrect.

**Screenshot 3**: Snowflake Showing Transformed Data

A screenshot of a computer

AI-generated content may be incorrect.