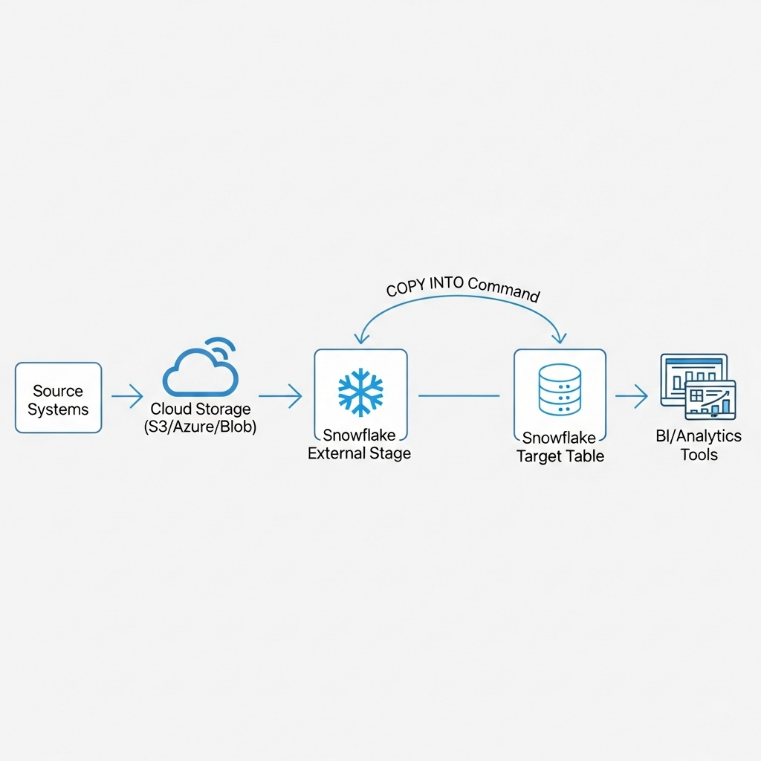
**Copy-into-external-stages:**

This project designs and implements a robust, scalable, and automated data ingestion pipeline for loading data from cloud storage into the Snowflake Data Cloud. The solution leverages **Snowpark Python API** for programmatic control and orchestration, and Snowflake's **External Stages** coupled with the COPY INTO command for high-performance, fault-tolerant data loading.

The primary data source is a cloud object storage (e.g., AWS S3, Azure Blob Storage), where source systems deposit files in various formats (e.g., CSV, JSON, Parquet). The pipeline automatically detects, validates, and loads these files into structured Snowflake tables, ready for analytics and reporting.



**Objectives**

The key objectives of this project are:

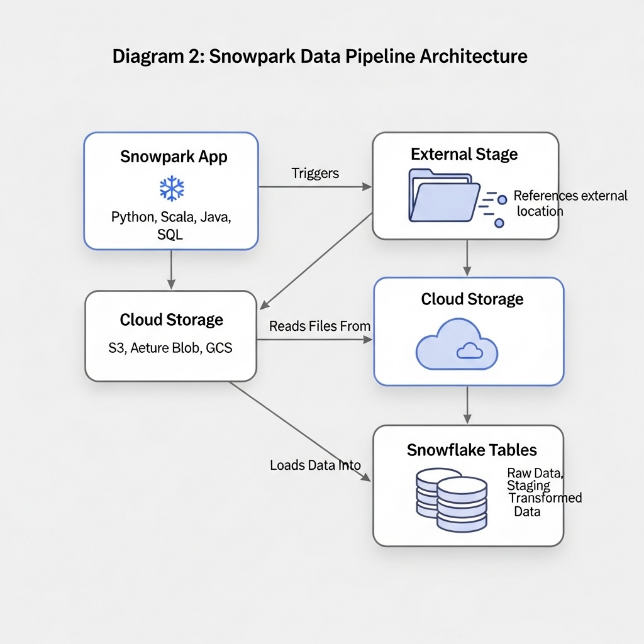
* **To Automate Data Ingestion:** Eliminate manual, error-prone processes for loading data from cloud storage.
* **To Ensure Data Integrity and Reliability:** Implement a pipeline that is idempotent (can be rerun safely), handles errors gracefully, and provides clear load statuses.
* **To Leverage Modern Data Stack Tools:** Utilize the Snowpark Python API for a code-first, DevOps-friendly approach to data engineering.
* **To Create a Scalable and Performant Solution:** Use Snowflake's bulk loading capabilities via the COPY INTO command to efficiently process large volumes of data.
* **To Provide Clear Monitoring and Logging:** Track the history of data loads, success rates, and any failures for operational oversight.

**System Design**

**1. Architecture**

The system follows a modular, event-driven architecture. The core components are:

1. **Cloud Storage Bucket:** The landing zone for all source data files. Folders can be structured by data source and date .
2. **Snowflake External Stage:** A named Snowflake object that stores the connection information and credentials to access the Cloud Storage Bucket, acting as a secure bridge.
3. **Snowpark Python Application:** The orchestration engine. This application:
   * Connects to Snowflake.
   * Executes the COPY INTO command for specified target tables.
   * Checks the load status and logs results.
   * Can be scheduled via an external scheduler (e.g., Apache Airflow, Prefect, cron) or a Snowflake task.
4. **Snowflake Target Tables:** The final destination for the validated and loaded data.
5. **Metadata / Logging Table:** A table within Snowflake to record the history and outcome of every COPY INTO execution.



**2. Technology Stack**

| Component | Technology | Justification |
| --- | --- | --- |
| **Data Warehouse** | Snowflake | Scalable, cloud-native data platform with separation of storage and compute. |
| **Data Processing** | Snowpark Python API | Provides a DataFrame and programming interface for complex data logic within a secure Snowflake runtime. |
| **Cloud Storage** | AWS S3 | Highly durable and scalable object storage. (Could be Azure Blob Storage or Google Cloud Storage). |
| **Orchestration** | Apache Airflow | For scheduling, monitoring, and managing the workflow dependencies. (Alternatively, Prefect, Dagster). |
| **Security** | Snowflake Key Pair Authentication & Storage IAM Roles | Secure, programmatic access without storing passwords. |

**Implementation**

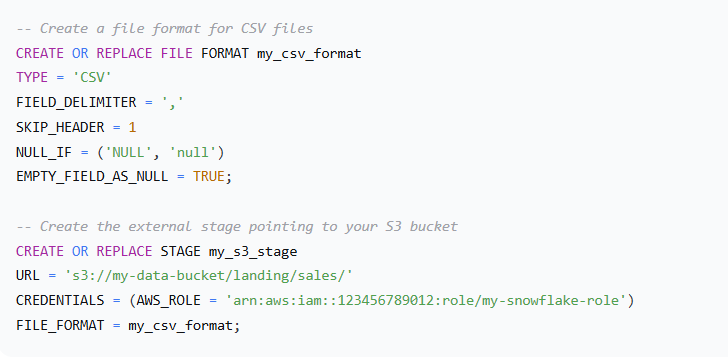
**1. Prerequisites and Setup**

1. **Snowflake Account:** With appropriate privileges (CREATE STAGE, CREATE TABLE, USAGE on warehouse and database).
2. **Cloud Storage:** An S3 bucket with an IAM role/policy granting read access to Snowflake.
3. **Python Environment:** Python 3.8+ with the snowflake-snowpark-python library installed.

**2. Key Configuration Steps**

**Step 1: Create an External Stage in Snowflake**

This establishes the secure link to your cloud storage.



**Step 2: Create the Target and Logging Tables**

sql

**

**3. Core Snowpark Python Application**

The following Python script is the heart of the ingestion pipeline.

****

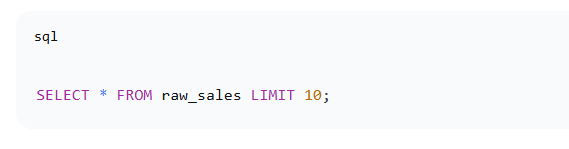
****

**Results and Validation**

**1. Output and Monitoring**

Upon successful execution, the data from the file in the external stage is loaded into the raw\_sales table. The copy\_history table provides a complete audit trail.

**Query to Check Loaded Data:**



**Query to Check Load History:**

