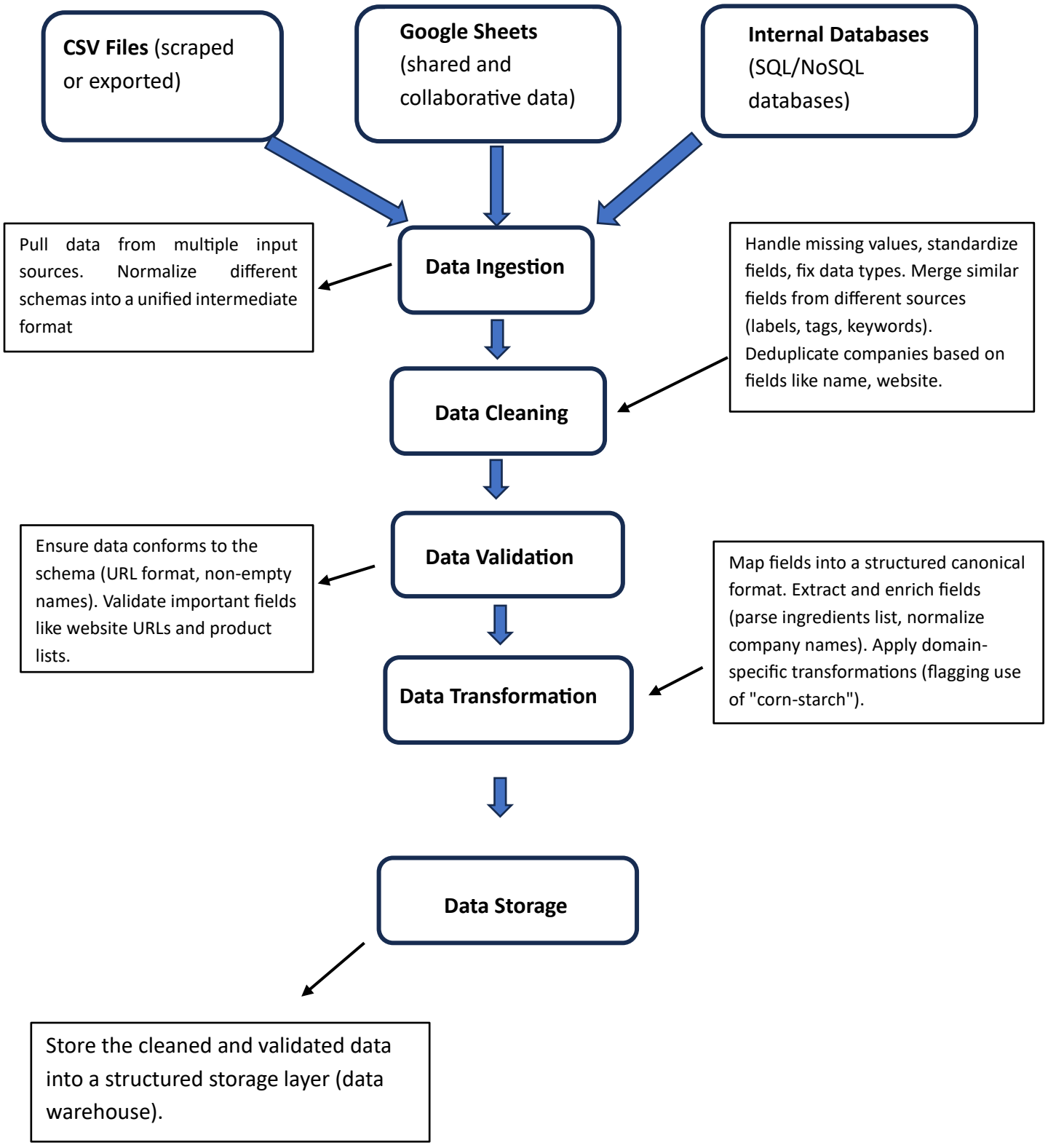
# Conceptual Data Pipeline Flowchart (with Written Explanation)



# Infrastructure Choice and Trade-offs

**Amazon S3 (Storage Layer):** Input data (input_data.json) is uploaded daily to an S3 bucket. Its raw data storage, cost-effective, and durable.

**Why I Chose it:** It is cheap, reliable storage for batch uploads

**Trade-offs:** It does not have compute power; hence it needs EC2 or Lambda for processing

**Amazon EC2 (Processing Layer):** A lightweight EC2 instance (t3.small) runs a daily cron job. The job pulls data from S3, runs the Python script**(this_script.py)** to clean and score leads.

**Why I Chose it: It gives f**ull control over environment to schedule and run cron job

**Trade-offs:** Needs maintenance such as (patches, updates and restarts)

**AWS Lambda (Automation):** Trigger (Lambda) and monitors S3 for new files. It can also be used to triggers the processing job if you want a serverless, event-driven alternative to EC2.

**Why I Chose it:** It is Serverless and it scales automatically.

**Trade-offs:** It has a Cold starts and timeout limits (~15 min).

**Amazon RDS (Relational Database Storage):** Processed, cleaned data can be saved into an RDS (PostgreSQL/MySQL) database. It is easy query able and its integration well with BI tools (Power BI).

**Why I Chose it:** It is a Structured, query able storage for clean data.

**Trade-offs:** Managed service, and incurs daily costs

**Amazon CloudWatch (Monitoring):** Monitor logs and metrics from EC2 instances, S3, and RDS or Lambda functions. It can alert for failures or processing anomalies.

**Why I Chose it: It has m**onitoring and alerting built-in. It also integrates well with most AWS infrastructures

**Trade-offs**: It can incur some slight additional costs.

# AWS-Based Infrastructure Architecture Diagram of Data Pipeline