Wistia Video Analytics (Data Engineering Academy)

1. Introduction and Business Goals

- **Scope:**
    - Designing the entire system architecture: ingestion, storage, processing, and reporting
    - Authenticating and ingesting data from the Wistia Stats API (both media and visitor level)
    - Handling pagination and incremental data pulls
    - Running the pipeline in production mode for 7 days
    - Implementing CI/CD using GitHub.
    - Documenting decisions, assumptions, and tradeoffs
- **Business Objectives:** The marketing team uses Wistia to track video engagement across Facebook and YouTube. We aim to:
- Collect media-level and visitor-level analytics from Wistia's Stats API
- Build an automated data pipeline to ingest and analyze performance
- Use these insights to improve marketing strategies

This project simulates a real-world data engineering assignment with full responsibility placed on the student team to design, implement, and operate the system.

---

2. Requirements

This section details the specific requirements the solution must meet.

- You must design the architecture yourself and present it to SME for approval
- DO NOT USE DBT FOR TRANSFORMATION.
- DO NOT USE ANYTHING APART FROM AWS/Azure.
- You must use GitHub for version control and CI/CD
- You must use Python for API ingestion and PySpark for data transformation.

The pipeline should run for 7 days.

Functional Requirements:

- Design your own architecture for ingestion, processing, and storage

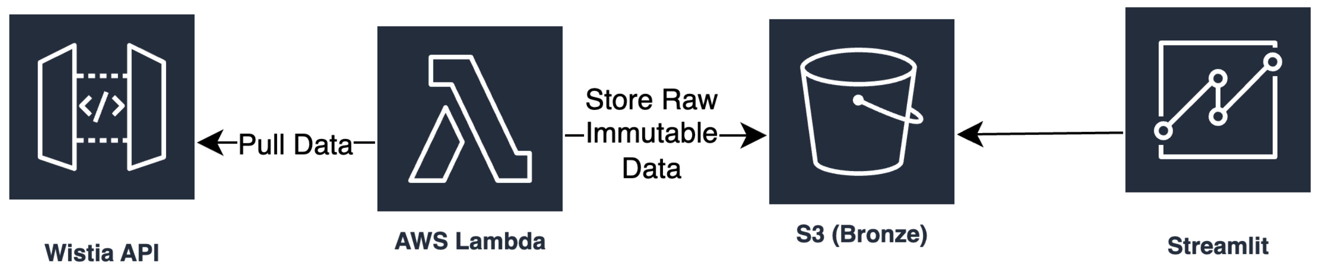
- Authenticate to Wistia Stats API using token-based Basic Auth
- Extract media metadata (title, ID, hashed_id, created_at, etc.)
- Extract engagement metrics (plays, play rate, watch time, etc.)
- Extract visitor-level data (IP, engagement events)
- Implement pagination to fetch all pages of results
- Implement incremental ingestion based on created_at/updated_at
- Run this pipeline in "production mode" for 7 consecutive days
- Implement a CI/CD pipeline using GitHub Actions or equivalent
- Store results in a structured data model (DWH or cloud database)
- Create final reports or dashboards for insights (optional)
- Submit a GitHub repo with documentation, pipeline code, CI/CD setup, and instructions

---

3. Architectural Design

This is the high-level overview of the system.

High-Level Diagram:



Technology Stack

Data Source

Wistia API

Architecture

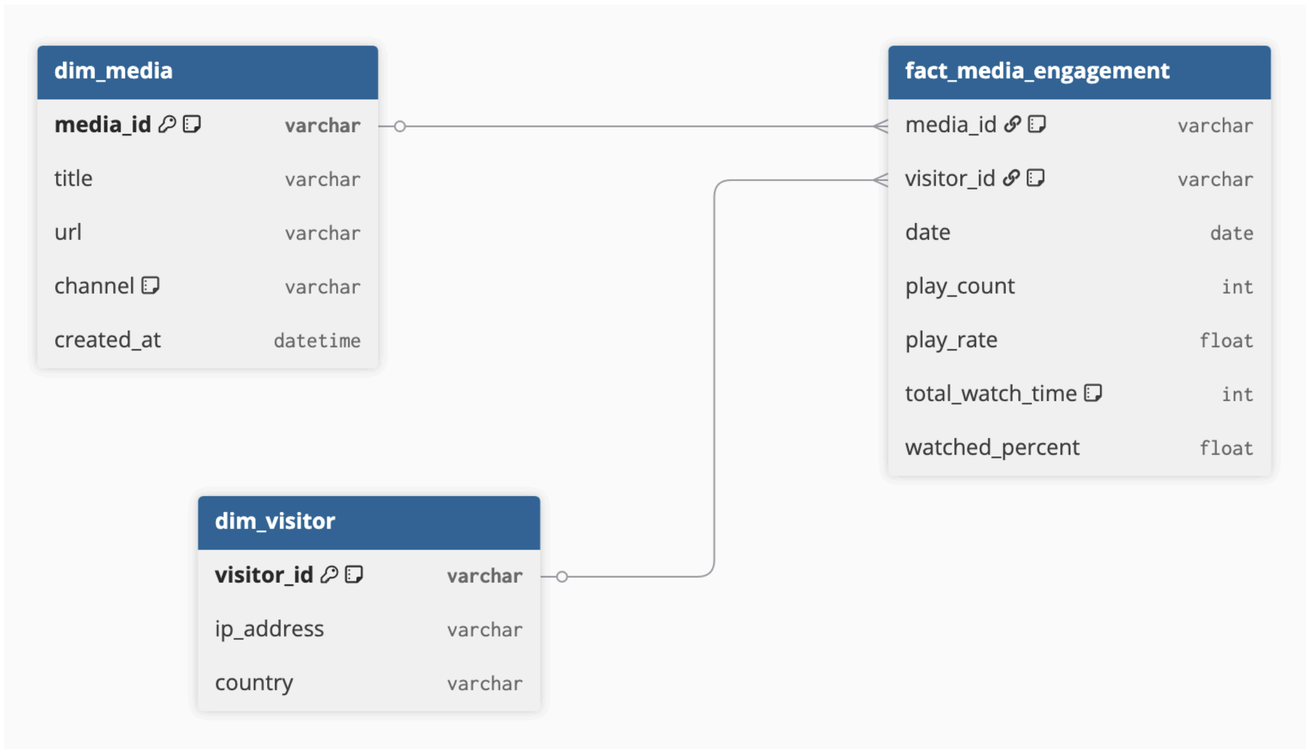
Bronze Stage

Lambda → S3

Dashboards (Streamlit) Then we can hit S3 directly with Streamlit on the views to create BI dashboards. (Optionally)

4. Data Model

- dim_media → https://docs.wistia.com/reference/get_stats-medias-mediaid
- dim_visitor → https://docs.wistia.com/reference/get_stats-visitors and
- fct_media_engagement → https://docs.wistia.com/reference/get_stats-medias-mediaid-engagement



Deliverables

Area	Criteria
Architecture	Clear, scalable, modular design
Data Quality	Correct use of pagination, incremental logic, and schema
Engineering	Effective error handling, retries, logging
CI/CD	Working CI/CD for deployment or validation
Documentation	README + architecture diagram + setup instructions