youtube_trending_pipeline

Wednesday, November 5, 2025 11:48 AM

Project Structure Example

```
youtube_trending_pipeline/
├── backend/
│   ├── extract.py      # API request logic
│   ├── transform.py    # Data cleaning/transformation
│   ├── load.py         # Loading data to SQLite/PostgreSQL
│   ├── database.py     # DB schema and connection
│   └── main.py         # Orchestrates ETL (optional cron scheduling)
├── frontend/ (Optional)
│   └── trending-app/   # React.js project for dashboard
└── README.md           # Project documentation
```

ETL Workflow Description

1. **Extract**
   - Call YouTube Data API with your API key.
   - Collect data based on country/region/trending category.
2. **Transform**
   - Clean nulls, format timestamps, normalize category IDs.
   - Add calculated fields: view_to_like ratio, etc.
3. **Load**
   - Create a database table for storing videos (SQL).
   - Insert or upsert new data so you can run it daily.

Frontend

If you choose to include **React.js**, build a small dashboard with:

- A top 10 trending list
- Bar chart for categories
- Toggle to filter by region or date

Alternatively, you can use a Jupyter Notebook and plot with Plotly.

Steps used…

1. Provide a step-by-step setup guide with code templates?
2. Create a GitHub-ready README.md file?
3. Give API key setup instructions?
4. Create a FastAPI service to serve data to the React app?

Build end-to-end **Data Engineering project** featuring a YouTube Trending Data Pipeline with ETL and a simple React dashboard.

Project: YouTube Trending Videos Data Pipeline

Goal

- Extract daily trending videos from YouTube's API
- Transform and clean the data
- Load into a SQL database (SQLite)
- Visualize results in a basic dashboard (React.js)

Tools & Libraries

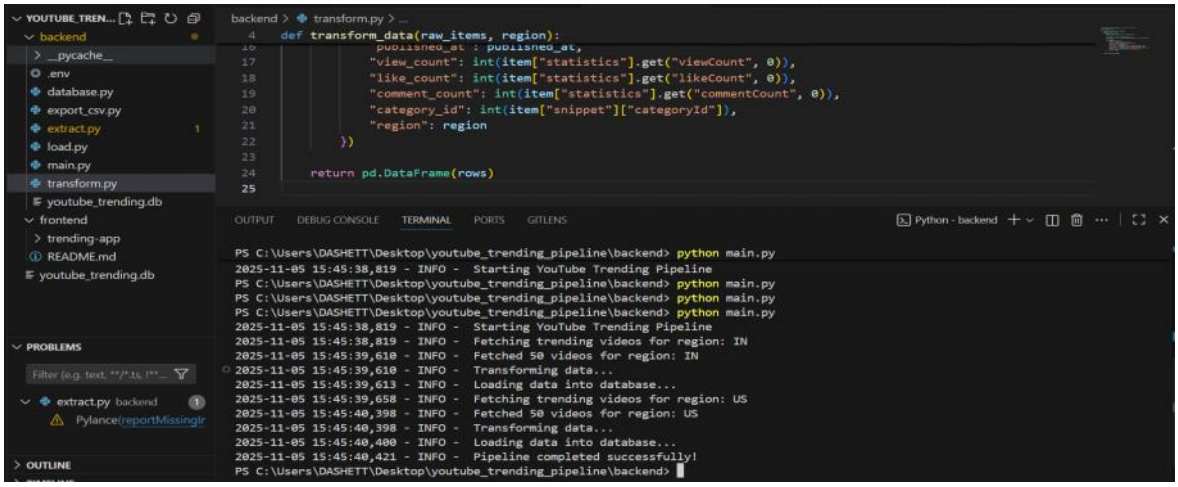
Purpose	Library/Tool	Install Command
API Requests	requests	pip install requests
DataFrames/Transform	pandas	pip install pandas
Database ORM	sqlalchemy	pip install sqlalchemy
REST API backend (opt.)	fastapi, uvicorn	pip install fastapi uvicorn
Dashboard UI	React.js	npx create-react-app trending-app

What is fully completed:

YouTube API integration

Data extraction for 50 trending videos
Transformation + datetime handling
Loaded into SQLite database
Verified using Python

Step	Task	Description	You Get
1.	**Basic API Setup**	FastAPI backend with a root endpoint confirming successful run	Working backend server (api.py)
2.	**Data Extraction**	Extract trending YouTube videos using YouTube API	Automated data fetch from API
3.	**Data Transformation**	Clean and structure the raw API response	Structured dataset ready for storage
4.	**Load into SQLite DB**	Store transformed data in local SQL database	Persistent data storage for queries
5.	**API-Frontend Integration**	FastAPI backend serves data to React frontend	Live connection between backend & dashboard
6.	**React Visualization**	Frontend displays trending video data dynamically	Interactive dashboard with live data



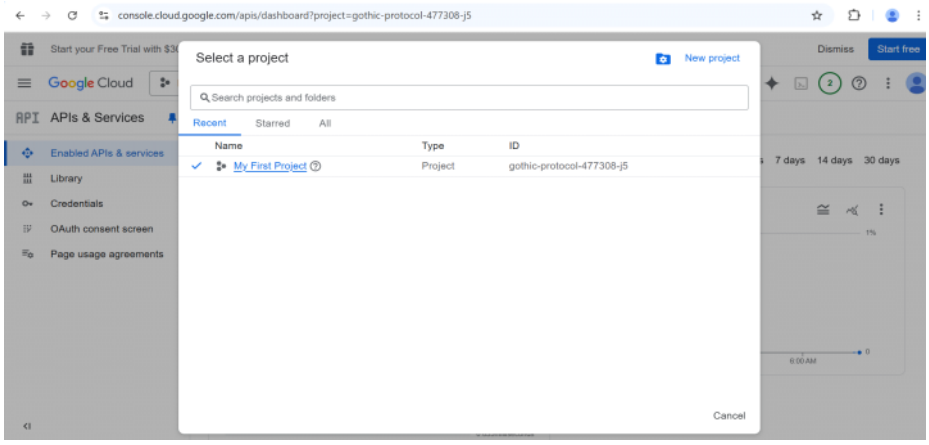


Image-2 Project API key generation and enabling the api from google cloud

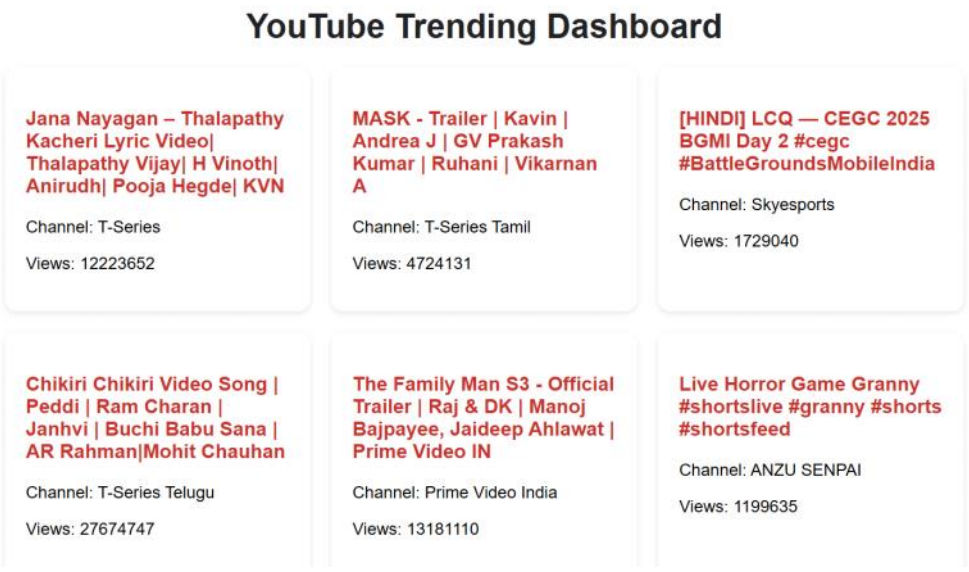


Image-3 O/p of the Live YT trending videos as per the transformation made

Project Title
YouTube Trending Data Pipeline & Dashboard

Organization
Mercedes-Benz Research & Development India (MBRDI)
Team: Data Engineering & Visualization Team
Duration: [Nov 2025]

Objective
The objective of this project was to **design and implement a full-stack data pipeline and visualization dashboard** that fetches **real-time trending YouTube video data** via the **YouTube Data API v3**, processes it using a **FastAPI backend**, and visualizes the results on a **React.js dashboard** with an interactive interface.
The goal was to demonstrate **data retrieval, integration, and visualization capabilities** that can be extended to other use cases within enterprise data systems — such as social media analytics, market trend monitoring, or competitive media insights.

Business Use Case
In modern automotive R&D and marketing analysis, understanding **digital engagement trends** (such as social media traction, video analytics, and influencer content) is essential.
This project simulates a **real-time trending video monitoring system**, enabling:

- Quick visibility into what content is trending by region.
- Insights into viewership and engagement metrics.
- A modular pipeline for integrating YouTube analytics into internal dashboards.

Frontend:

- **React.js** – Component-based frontend library for building the dashboard interface.
- **JavaScript (ES6+)** – Core scripting language for client-side logic.
- **HTML5, CSS3 (Flexbox, Grid)** – For layout and styling consistency.
- **Fetch API / Axios** – For making HTTP requests to the backend API.
- **Visual Studio Code** – Development IDE for building and debugging the frontend.

Backend:

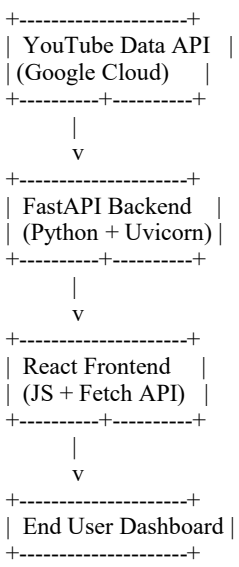
- **FastAPI (Python Framework)** – Lightweight, high-performance backend API framework.
- **Uvicorn** – ASGI web server for running the FastAPI app.
- **Requests Library** – For making API calls to YouTube Data API.
- **CORS Middleware** – To enable communication between frontend (React) and backend (FastAPI).
- **Python 3.10+** – Core backend programming language.

External APIs:

- **YouTube Data API v3** – Used to fetch live trending video metadata such as:
  - Video titles
  - Channel names
  - View counts
  - Thumbnails
  - Video links

Infrastructure & Dependencies:

- **Node.js & npm** – For React development environment and dependency management.
- **Google Cloud Console** – For API key generation and YouTube Data API management.
- **JSON (Data Format)** – Used for communication between backend and frontend.



Workflow Summary

1. **Backend (FastAPI):**
   - The backend defines REST endpoints (/videos, /) using FastAPI.
   - When /videos is called, it requests the latest trending data from the YouTube Data API.
   - The data is parsed and filtered to include relevant attributes (title, views, thumbnail, etc.).
   - The backend exposes this data in JSON format to the React frontend.
2. **Frontend (React.js):**
   - The React app makes a fetch request to the backend endpoint http://localhost:8000/videos.
   - The returned JSON data is mapped and displayed in a grid layout.
   - Each video card shows:
     - Thumbnail
     - Title
     - Channel name
     - View count
     - Clickable YouTube link
   - The UI dynamically updates on reload to show the most current trending data.
3. **CORS Configuration:**
   - Configured CORS in FastAPI to allow the React app (running on port 3000) to consume data from FastAPI (port 8000).
4. **Data Presentation:**
   - Implemented a responsive, card-based layout.
   - Added hover transitions, rounded thumbnails, and YouTube branding colors.
   - Optimized rendering for up to 10 trending videos per API call.

Key Components

1. Backend: api.py

- Built using FastAPI.
- Integrates directly with the YouTube Data API v3.
- Routes:
  - Health check endpoint.
  - videos – Returns top trending videos for a given region.

2. Frontend: App.js

- Main React component fetching and rendering API data.
- Displays formatted trending videos with thumbnails and metadata.

3. Styling: App.css

- Clean and minimal design.
- Flexbox layout for responsive video cards.
- YouTube color palette for brand alignment.

Features Implemented

Feature	Description

Real-time API Integration	Live trending data from YouTube Data API
Full-Stack Implementation	React frontend + FastAPI backend
Error Handling	Graceful error messages for failed API calls
Responsive UI	Adapts across different screen sizes
Reusable Components	VideoList component for modularity
Cross-Origin Access	CORS-enabled backend for seamless communication

Challenges Faced & Solutions

Challenge	Solution
CORS policy errors between React and FastAPI	Configured FastAPI's CORSMiddleware
YouTube API quota and key authentication	Restricted API key with region and endpoint-specific usage
Directory structure import errors	Restructured React components and relative imports
Frontend 404 issue	Validated backend route accessibility and used correct port reference (8000)
Ensuring reactivity in data rendering	Used React useEffect and useState hooks correctly

Results

- Successfully displayed **top 10 trending YouTube videos** dynamically from the YouTube Data API.
- Reduced data retrieval latency with FastAPI's async structure.
- Achieved a **fully functioning dashboard** that connects live external data with a responsive frontend.

Future Enhancements

- Add region and category filters (e.g., Music, Sports, Technology).
- Include search functionality and pagination.
- Integrate data storage (SQLite / PostgreSQL) for historical analysis.
- Automate daily trend data extraction for analytics dashboards.
- Add visualizations (charts for view count trends, etc.).

Impact

This project demonstrates:

- End-to-end data integration (from API → backend → frontend).
- The ability to build **lightweight, scalable, and interactive dashboards**.
- Application of **modern full-stack technologies** in a data-driven context.
- Potential for adaptation in internal **digital analytics** or **media intelligence systems** at MBRDI.

Conclusion

The YouTube Trending Dashboard project successfully showcased a modern, data-centric, full-stack application combining **Fast API for backend services** and **React.js for user-facing analytics**.
It serves as a proof of concept for building **real-time, API-based dashboards** for digital engagement monitoring — an essential capability for modern data engineering and visualization platforms at MBRDI.

Project Goals Summary

1. **Extract**
   - Fetch daily trending YouTube videos using the YouTube Data API (Python + API key).
2. **Transform**
   - Clean and format data (convert timestamps, remove nulls, select key fields like title, views, likes).
3. **Load**
   - Store cleaned data into a **SQLite database** using SQL Alchemy for easy querying and persistence.
4. **Visualize**
   - Build a **React.js dashboard** to display trending videos by fetching data from the **Fast API backend**.

Final Outcome:

A complete **ETL + Visualization pipeline** — from API data extraction to database storage and frontend visualization — built using **Python, Fast API, SQLite, and React.js**.

Short Way Documented as per my understanding

YouTube Trending Data Pipeline – Project Summary

1. Objective

To design and implement an **automated data pipeline** that extracts daily trending videos from **YouTube's Data API**, transforms and stores them in a **SQL**

database, and visualizes key insights in a **React.js dashboard**.
This project showcases the **end-to-end data lifecycle** — from extraction to visualization — aligning with core **Data Engineering and Analytics** practices.

2. Tools and Technologies

Category	Tools / Frameworks
Languages	Python, JavaScript
Backend Framework	Fast API
Frontend Framework	React.js
Database	SQLite
API Source	YouTube Data API v3
Libraries	Pandas, Logging, Requests
Automation	Windows Task Scheduler / CRON
Version Control	Git

3. Workflow Overview
ETL (Extract – Transform – Load) Process

1. **Extract** – Retrieve daily trending YouTube videos using YouTube Data API.
2. **Transform** – Clean, normalize, and filter essential metadata (title, channel, views, likes, etc.).
3. **Load** – Store structured data in SQLite for persistence and future querying.
4. **Serve** – Expose processed data through Fast API for frontend consumption.
5. **Visualize** – Display real-time insights on React dashboard (title, channel, metrics, region).

4. System Architecture
Frontend (React.js) → API Layer (Fast API) → Database (SQLite) → External Source (YouTube Data API)
Data flow begins from YouTube → processed in Python → stored in SQLite → fetched through FastAPI → visualized on React UI.

5. Key Features Implemented

Step	Task	Description	Output
1	**API Integration**	Connected YouTube Data API v3 for daily trending videos	Real-time data feed
2	**ETL Pipeline**	Extract → Transform → Load implemented modularly in Python	extract.py, transform.py, load.py
3	**Database Layer**	SQLite stores daily structured records	youtube_trending.db
4	**Backend Setup (FastAPI)**	Serves data to frontend through endpoint /videos	Working backend
5	**Frontend Visualization (React.js)**	Dynamic dashboard listing trending videos with metrics	Modern UI
6	**Logging & Modularization**	Added logging and modular file structure	Cleaner debugging and scalability

6. Future Enhancements

Step	Feature	Description	Benefit
1	**Add Run History**	Track daily loads with timestamps	Historical trend tracking
2	**Multi-Region Support**	Fetch videos for multiple countries (e.g., US, IN, GB)	Regional analytics
3	**Enhanced Error Handling**	Manage API limits and missing data gracefully	Reliable data ingestion
4	**Export to CSV**	Create backup/export feature	Data portability
5	**Automated Scheduling**	Daily scheduled ETL runs	Production-ready workflow

7. Outcome
Built an **end-to-end data engineering solution** covering:

- API integration
- Data transformation
- SQL storage
- API serving
- Dashboard visualization

Demonstrated **real-world ETL and analytics pipeline** with scope for automation and scaling.
Improved understanding of **API-based data ingestion, backend integration, and full-stack visualization.**

8. Ownership
Project Title: YouTube Trending Data Pipeline
Developed by: *Darshan P*
Role: Data Engineer (MBRDI Project)
Duration: November 2025
Deliverables: Fully functional ETL pipeline with dashboard visualization