**YouTube API Integration and Data Pipeline Documentation**

**Overview**

This documentation provides an overview and setup guide for the YouTube API Integration and Data Pipeline project. The project involves fetching data from the YouTube API, processing it, storing it in a SQLite database, and creating endpoints to access and visualize the data every hour using Flask and Streamlit.

## Files

### 1. creating\_schema.py

* **Purpose:** Creates SQLite database tables to store raw video data and processed insights.
* **Functionality:**
  + Defines the schema for two tables: raw\_videos and processed\_insights.
  + Establishes a connection to the SQLite database (youtube\_channel\_data.db).
  + Creates the raw\_videos table to store raw video data fetched from the YouTube API.
  + Creates the processed\_insights table to store processed insights derived from the raw video data.

### 2. Flask\_app.py

* **Purpose:** Implements a Flask application to serve as the backend API for fetching and processing YouTube data.
* **Functionality:**
  + Imports necessary libraries for working with Flask, SQLite, Google API, and data processing.
  + Defines functions to interact with the SQLite database (get\_db\_connection, fetch\_data).
  + Implements functions to fetch video details from the YouTube API (fetch\_video\_details), extract keywords, and calculate sentiment.
  + Defines API endpoints to retrieve raw video data, insights, view count trends, and generate a word cloud.

### 3. Streamlit\_app.py

* **Purpose:** Implements a Streamlit web application for visualizing YouTube data fetched through the Flask API.
* **Functionality:**
  + Imports Streamlit, requests, pandas, matplotlib, and PIL libraries.
  + Defines a dictionary of YouTube channels and their corresponding IDs.
  + Provides a user interface to select a channel, plot view count trends, view raw video data, insights, and generate a word cloud.
  + Sends requests to the Flask API endpoints for fetching data and displaying it in the Streamlit app.

### 4. Youtube\_channel\_data.db

* **Purpose:** SQLite database file to store raw video data and processed insights.

## Setup Guide

### Requirements.txt

* Python 3.x
* Flask
* Streamlit
* pandas
* matplotlib
* PIL
* requests
* googleapiclient
* textblob
* wordcloud

### Steps

1. **Clone the Repository:** Clone the project repository from the source.
2. **Install Dependencies:** Install the required Python packages using pip:

pip install -r requirements.txt

1. **Obtain API Key:** Obtain a YouTube Data API key from the Google Cloud Console.
2. **Update API Key:** Replace the placeholder API key (api\_key) in flask\_app.py with your API key obtained in the previous step.
3. **Run the Flask App:** Run the Flask application (flask\_app.py) using the following command:

python flask\_app.py

1. **Run the Streamlit App:** Open a new terminal window and run the Streamlit application (streamlit\_app.py) using the following command:

streamlit run streamlit\_app.py

1. **Interact with the App:** Access the Streamlit app through the provided URL and interact with the user interface to visualize YouTube data.

## Conclusion

This concludes the documentation for the YouTube API Integration and Data Pipeline project. The setup guide provided above should help users set up the project locally and interact with the Flask and Streamlit applications to visualize YouTube data effectively.

An alternative approach to the YouTube API Integration and Data Visualization project. The project involves populating a SQLite database with YouTube data using the YouTube API and creating a Streamlit-based visualization dashboard to analyse the data.

**File: populate\_db.py**

* **Purpose:** This script fetches data from the YouTube API and populates a SQLite database with raw video data and processed insights.
* **Functionality:**
  + Establishes a connection to the SQLite database (youtube\_channel\_data.db).
  + Defines functions to extract keywords, fetch video details, and insert data into the database.
  + Fetches video details for specified channels using the YouTube API and inserts them into the database.
  + Extracts keywords from video titles and descriptions, calculates sentiment, and inserts processed insights into the database.

**File: visualization.py**

* **Purpose:** This script generates a Streamlit-based dashboard to visualize and analyze the YouTube data populated in the SQLite database.
* **Functionality:**
  + Imports necessary libraries for visualization, SQLite interaction, and Streamlit.
  + Defines functions to fetch data from the database, calculate trends in view counts over time, extract top keywords, analyze audience engagement metrics, and extract additional insights.
  + Displays a user interface using Streamlit to select a channel, visualize trends in view counts over time, display top keywords in a word cloud, analyze audience engagement metrics, and showcase additional insights such as top trending videos and average engagement rates.
  + Utilizes Matplotlib, Seaborn, and WordCloud for data visualization within the Streamlit app.

### Steps

1. **Clone the Repository:** Clone the project repository from the source.
2. **Install Dependencies:** Install the required Python packages using pip:

pip install -r requirements.txt

1. **Set API Key:** Ensure that your YouTube Data API key is correctly set in the populate\_db.py file (api\_key variable).
2. **Populate Database:** Run the populate\_db.py script to populate the SQLite database with YouTube data:

python populate\_db.py

1. **Run the Streamlit App:** Open a new terminal window and run the Streamlit application (visualization.py) using the following command:

streamlit run visualization.py

1. **Interact with the Dashboard:** Access the Streamlit dashboard through the provided URL and interact with the user interface to visualize and analyze the YouTube data effectively.

## Conclusion

This concludes the documentation for the alternative approach to the YouTube API Integration and Data Visualization project. The provided setup guide should assist users in setting up the project locally and interacting with the Streamlit-based visualization dashboard to gain insights into YouTube channel analytics.

**Deployment on Azure**

1. Begin by establishing a new directory for the deployment process.
2. Ensure that ports are properly exposed to facilitate communication.
3. Configure endpoints within the config file, incorporating the desired port and implementing a health route for added safety.
4. Create a Docker file equipped with essential operating system specifications such as Bookworm, and include Python for necessary functionalities.
5. Verify that your code adheres to the PEP 8 configuration standards for optimal readability and maintainability.

Top of Form

Bottom of Form