Project Title	**YouTube Data Harvesting and Warehousing using SQL, MongoDB and Streamlit**
Skills take away From This Project	**Python scripting, Data Collection, MongoDB, Streamlit, API integration, Data Management using MongoDB (Atlas) and SQL**
Domain	**Social Media**

Problem Statement:

The problem statement is to create a Streamlit application that allows users to access and analyze data from multiple YouTube channels. The application should have the following features:

1. Ability to input a YouTube channel ID and retrieve all the relevant data (Channel name, subscribers, total video count, playlist ID, video ID, likes, dislikes, comments of each video) using Google API.
2. Option to store the data in a MongoDB database as a data lake.
3. Ability to collect data for up to 10 different YouTube channels and store them in the data lake by clicking a button.
4. Option to select a channel name and migrate its data from the data lake to a SQL database as tables.
5. Ability to search and retrieve data from the SQL database using different search options, including joining tables to get channel details.

Approach:

1. **Set up a Streamlit app:** Streamlit is a great choice for building data visualization and analysis tools quickly and easily. You can use Streamlit to create a simple UI where users can enter a YouTube channel ID, view the channel details, and select channels to migrate to the data warehouse.
2. **Connect to the YouTube API:** You'll need to use the YouTube API to retrieve channel and video data. You can use the Google API client library for Python to make requests to the API.
3. **Store data in a MongoDB data lake:** Once you retrieve the data from the YouTube API, you can store it in a MongoDB data lake. MongoDB is a great choice for a data lake because it can handle unstructured and semi-structured data easily.

4. **Migrate data to a SQL data warehouse:** After you've collected data for multiple channels, you can migrate it to a SQL data warehouse. You can use a SQL database such as MySQL or PostgreSQL for this.
5. **Query the SQL data warehouse:** You can use SQL queries to join the tables in the SQL data warehouse and retrieve data for specific channels based on user input. You can use a Python SQL library such as SQLAlchemy to interact with the SQL database.
6. **Display data in the Streamlit app:** Finally, you can display the retrieved data in the Streamlit app. You can use Streamlit's data visualization features to create charts and graphs to help users analyze the data.

Overall, this approach involves building a simple UI with Streamlit, retrieving data from the YouTube API, storing it in a MongoDB data lake, migrating it to a SQL data warehouse, querying the data warehouse with SQL, and displaying the data in the Streamlit app.

Example Data Extraction from Youtube to MongoDB :


```
{
   "Channel_Name": {
      "Channel_Name": "Example Channel",
      "Channel_Id": "UC1234567890",
      "Subscription_Count": 10000,
      "Channel_Views": 1000000,
      "Channel_Description": "This is an example channel.",
      "Playlist_Id": "PL1234567890"
   },
   "Video_Id_1": {
      "Video_Id": "V1234567890",
      "Video_Name": "Example Video 1",
      "Video_Description": "This is an example video.",
      "Tags": ["example", "video"],
      "PublishedAt": "2022-01-01T00:00:00Z",
      "View_Count": 1000,
      "Like_Count": 100,
      "Dislike_Count": 10,
      "Favorite_Count": 5,
      "Comment_Count": 20,
      "Duration": "00:05:00",
```

```
        "Thumbnail": "https://example.com/thumbnail.jpg",
        "Caption_Status": "Available",
        "Comments": {
          "Comment_Id_1": {
            "Comment_Id": "C1234567890",
            "Comment_Text": "This is a comment.",
            "Comment_Author": "Example User",
            "Comment_PublishedAt": "2022-01-01T00:01:00Z"
          },
          "Comment_Id_2": {
            "Comment_Id": "C2345678901",
            "Comment_Text": "This is another comment.",
            "Comment_Author": "Another User",
            "Comment_PublishedAt": "2022-01-01T00:02:00Z"
          }
        }
    },
    "Video_Id_2": {
        "Video_Id": "V2345678901",
        "Video_Name": "Example Video 2",
        "Video_Description": "This is another example video.",
        "Tags": ["example", "video"],
        "PublishedAt": "2022-01-02T00:00:00Z",
        "View_Count": 2000,
        "Like_Count": 200,
        "Dislike_Count": 20,
        "Favorite_Count": 10,
        "Comment_Count": 30,
        "Duration": "00:10:00",
        "Thumbnail": "https://example.com/thumbnail.jpg",
        "Caption_Status": "Not Available",
        "Comments": {}
    }
}
```

Example SQL Tables:

Table: Channel

Column Name	Data Type	Description
channel_id	VARCHAR(255)	Unique identifier for the channel
channel_name	VARCHAR(255)	Name of the channel
channel_type	VARCHAR(255)	Type of the channel
channel_views	INT	Total number of views for the channel
channel_description	TEXT	Description of the channel
channel_status	VARCHAR(255)	Status of the channel

Table: Playlist

Column Name	Data Type	Description
playlist_id	VARCHAR(255)	Unique identifier for the playlist
channel_id	VARCHAR(255)	Foreign key referencing the channel table
playlist_name	VARCHAR(255)	Name of the playlist

Table: Comment

Column Name	Data Type	Description
comment_id	VARCHAR(255)	Unique identifier for the comment
video_id	VARCHAR(255)	Foreign key referencing the video table
comment_text	TEXT	Text of the comment
comment_author	VARCHAR(255)	Name of the comment author
comment_published_date	DATETIME	Date and time when the comment was published

Table: Video

Column Name	Data Type	Description
video_id	VARCHAR(255)	Unique identifier for the video
playlist_id	VARCHAR(255)	Foreign key referencing the playlist table
video_name	VARCHAR(255)	Name of the video
video_description	TEXT	Description of the video
published_date	DATETIME	Date and time when the video was published
view_count	INT	Total number of views for the video
like_count	INT	Total number of likes for the video
dislike_count	INT	Total number of dislikes for the video
favorite_count	INT	Total number of times the video has been marked as a favorite
comment_count	INT	Total number of comments on the video
duration	INT	Duration of the video in seconds
thumbnail	VARCHAR(255)	URL of the thumbnail for the video
caption_status	VARCHAR(255)	Status of the video caption

Reference :

Orientation Link:	English Recoding link	Tamil Recording Link
Streamlit Doc:	https://docs.streamlit.io/library/api-reference	
MongoDB Recordings:	Topic: MongoDB - Skill Enhancement Session	
Youtube API Reference:	https://developers.google.com/youtube/v3/getting-started	API Data Collection Reference Colab

SQL Query Output need to displayed as table in Streamlit Application:

1. What are the names of all the videos and their corresponding channels?
2. Which channels have the most number of videos, and how many videos do they have?
3. What are the top 10 most viewed videos and their respective channels?
4. How many comments were made on each video, and what are their corresponding video names?
5. Which videos have the highest number of likes, and what are their corresponding channel names?
6. What is the total number of likes and dislikes for each video, and what are their corresponding video names?
7. What is the total number of views for each channel, and what are their corresponding channel names?
8. What are the names of all the channels that have published videos in the year 2022?
9. What is the average duration of all videos in each channel, and what are their corresponding channel names?
10. Which videos have the highest number of comments, and what are their corresponding channel names?

Results:

This project aims to develop a user-friendly Streamlit application that utilizes the Google API to extract information on a YouTube channel, stores it in a MongoDB database, migrates it to a SQL data warehouse, and enables users to search for channel details and join tables to view data in the Streamlit app.

Project Evaluation metrics:

- You are supposed to write a code in a modular fashion (**in functional blocks**)
- Maintainable: It can be maintained, even as your codebase grows.
- Portable: It works the same in every environment (operating system)
- You have to maintain your code on **GitHub**.(Mandatory)
- You have to keep your **GitHub** repo public so that anyone can check your code.(Mandatory)
- Proper readme file you have to maintain for any project development(Mandatory)
- You should include basic workflow and execution of the entire project in the readme file on **GitHub**
- Follow the coding standards: https://www.python.org/dev/peps/pep-0008/
- You need to Create a Demo video of your working model and post in **LinkedIn**(Mandatory)