Youtube Analysis project

```
!pip3 install --upgrade google-api-python-client google-auth-
```

- `google-api-python-client`: This package allows Python developers to interact with Google APIs, including the YouTube Data API, which you'll likely use to retrieve data from YouTube. The `google-api-python-client` is essential for making requests to the YouTube Data API.
- `google-auth-httplib2` and `google-auth-oauthlib`: These packages provide authentication support for accessing Google APIs securely. They handle tasks such as obtaining OAuth 2.0 credentials and managing access tokens. `google-auth-httplib2` and `google-auth-oauthlib` are required for handling authentication, ensuring that your requests to the YouTube Data API are authorized and secure.

```
from googleapiclient.discovery import build
from google_auth_oauthlib.flow import InstalledAppFlow
from google.auth.transport.requests import Request
```

`from google_auth_oauthlib.flow import InstalledAppFlow`: This line imports the `InstalledAppFlow` class from the `google_auth_oauthlib.flow` module. The `InstalledAppFlow` class is used to perform the OAuth 2.0 authorization flow for installed applications.

`from google.auth.transport.requests import Request`: This line imports the `Request` class from the `google.auth.transport.requests` module. The `Request` class is used to make HTTP requests with the appropriate credentials.

- These imports are necessary to access the functionalities provided by the `google-api-python-client` and `google-auth-oauthlib` packages.
- The `build` function is crucial for creating a service object to interact with the YouTube Data API.
- The `InstalledAppFlow` class is essential for handling the OAuth 2.0 authorization flow required to access Google APIs securely.

- The `Request` class is necessary for making HTTP requests with the appropriate authentication credentials.

```
def search_videos(query):
  youtube = build('youtube','v3', developerKey="AIzaSyC7mUpUo
  request = youtube.search().list(part='id', type='video', q=
  response = request.execute()
  return response
```

- This function encapsulates the process of searching for videos on YouTube based on a given query, abstracting away the details of interacting with the YouTube Data API.
- The `build` function is used to create a service object, which is necessary for making requests to the YouTube Data API.
- The `search().list()` method constructs a search request with specific parameters, including the search query (`q`), and executes it to retrieve search results.

```
query= 'flower'
results = search_videos(query)
print(results)
```

```
def get_channel_details(channel_id):
  youtube = build('youtube','v3',developerKey="AIzaSyC7mUpUoZ
  request = youtube.channels().list(id=channel_id,part='snipp
  response =request.execute()
  return response

channel_id = 'UCRXx_pbQ7XhEPdZb4eUNOrQ'
details = get_channel_details(channel_id)
print(details)

# Retrieve information about the channel with the ID UCRXx_pb
channel = get_channel_details("UCRXx_pbQ7XhEPdZb4eUNOrQ")
```

```
# Print the channel name
print("Channel Name: ", channel["items"][0]["snippet"]["title

# Print the subscriber count
print("Subscriber Count: ", channel["items"][0]["statistics"]

# Print the view count
print("View Count: ", channel["items"][0]["statistics"]["view
```

This code snippet demonstrates how to use the `get_channel_details` function to retrieve details about a YouTube channel based on its channel ID.

```
def get_playlist_details(playlist_id):
  youtube = build('youtube','v3',developerKey="AIzaSyC7mUpUoZ
  request = youtube.playlists().list(part='snippet',id=playli
  response =request.execute()
  return response

playlist_id = 'PLbZYzh0g-r0uShgaf3RBy1NbmhJ8MAqiu'
details = get_playlist_details(playlist_id)
print(details)

# Print the playlist name
print("Playlist Name: ", playlist["items"][0]["snippet"]["tit

#Retrieve information about the playlist with the ID PLbZYzh0
playlist = get_playlist_details("PLbZYzh0g-r0uShgaf3RBy1NbmhJ

# Print the number of videos in the playlist
print("Number of Videos: ", playlist["items"][0])
```

```
def get_video_details(video_id):
    youtube = build('youtube','v3',developerKey="AIzaSyC7mUpU
    request = youtube.videos().list(part='snippet,statistics'
    response = request.execute()
    video = response['items'][0]
```

```
    if 'statistics' not in video:
        video['statistics'] = {'viewCount': 0, 'likeCount': 0
    else:
        if 'dislikeCount' not in video['statistics']:
            video['statistics']['dislikeCount'] = 0
    return video

video_id = 'NU2So9mJc2E'
video = get_video_details(video_id)
print("Video Details: ", video)
print("Views: ", video['statistics']['viewCount'])
print("Likes: ", video['statistics']['likeCount'])
print("Dislikes: ", video['statistics']['dislikeCount'])
print("Favorites: ", video['statistics']['favoriteCount'])
print("Comments: ", video['statistics']['commentCount'])
```

```
resource = build('youtube','v3',developerKey="AIzaSyC7mUpUoZk
request = resource.commentThreads().list(part = 'snippet', vi
response= request.execute()

# Retreiving Comments from the video.
items = response['items']
for item in items:
    print("Comment: ", item['snippet']['topLevelComment']['sn
```