Logotipo

Descrição gerada automaticamente

*Projeto final do curso de Python*

Kevelin Suellen de Souza Oliveira

Turma: 58875

Este projeto tem por finalidade extrair dados da API do YouTube para analisar e comparar canais na plataforma. O objetivo é reunir informações relevantes sobre o desempenho dos canais e dos vídeos, possibilitando uma análise detalhada e comparativa.Foi necessário a instalação das bibliotecas Pandas, google-api-python-client.

Configuração Inicial

O projeto começou com a criação de uma conta no Google Cloud Platform, o que permitiu obter a chave de API necessária para acessar a API do YouTube Data API v3. Esta chave é fundamental para autenticar as requisições e obter os dados desejados da plataforma.

Com a chave da API, foi desenvolvido um script Python utilizando a biblioteca googleapiclient para interagir com a API do YouTube. O código foi estruturado para extrair diferentes conjuntos de dados dos canais e vídeos.

Com o código é possível coletar dados sobre canais, como o nome do canal, descrição, data de publicação, país, número de inscritos, total de visualizações, total de vídeos e o ID da playlist de uploads.

Para cada canal, é extraido os IDs dos vídeos a partir da playlist de uploads.

Com os IDs dos vídeos, é possível obter detalhes sobre cada vídeo, incluindo o título, descrição, tags, data de publicação, duração, categoria, visualizações, curtidas e comentários.

E ainda, realizar uma análise das atividades dos canais, calculando a frequência de uploads mensais e o engajamento médio (visualizações, curtidas e comentários).

from googleapiclient.discovery import build

import pandas as pd

# Substitua 'YOUR\_API\_KEY' pela sua chave de API do YouTube

api\_key = 'AIzaSyBf6vTVrP0lMvpVjb6CqrGeNdo8p6hYtKE'

# Construa o serviço da API do YouTube

youtube = build('youtube', 'v3', developerKey=api\_key)

def get\_channel\_stats(youtube, channel\_ids):

    all\_data = []

    request = youtube.channels().list(

                part='snippet,contentDetails,statistics',

                id=','.join(channel\_ids))

    response = request.execute()

    for item in response['items']:

        data = {

            'channel\_id': item['id'],

            'channel\_name': item['snippet']['title'],

            'description': item['snippet'].get('description', 'N/A'),

            'published\_at': item['snippet'].get('publishedAt', 'N/A'),

            'country': item['snippet'].get('country', 'N/A'),

            'subscribers': int(item['statistics'].get('subscriberCount', 0)),

            'views': int(item['statistics'].get('viewCount', 0)),

            'total\_videos': int(item['statistics'].get('videoCount', 0)),

            'playlist\_id': item['contentDetails']['relatedPlaylists']['uploads']

        }

        all\_data.append(data)

    return pd.DataFrame(all\_data)

def get\_video\_ids(youtube, playlist\_id):

    video\_ids = []

    request = youtube.playlistItems().list(

                part='contentDetails',

                playlistId=playlist\_id,

                maxResults=50)

    response = request.execute()

    for item in response['items']:

        video\_ids.append(item['contentDetails']['videoId'])

    next\_page\_token = response.get('nextPageToken')

    while next\_page\_token:

        request = youtube.playlistItems().list(

                    part='contentDetails',

                    playlistId=playlist\_id,

                    maxResults=50,

                    pageToken=next\_page\_token)

        response = request.execute()

        for item in response['items']:

            video\_ids.append(item['contentDetails']['videoId'])

        next\_page\_token = response.get('nextPageToken')

    return video\_ids

def get\_video\_details(youtube, video\_ids, channel\_name):

    all\_video\_stats = []

    for i in range(0, len(video\_ids), 50):

        request = youtube.videos().list(

                    part='snippet,statistics,contentDetails',

                    id=','.join(video\_ids[i:i+50]))

        response = request.execute()

        for video in response['items']:

            video\_stats = {

                'channel\_name': channel\_name,

                'video\_id': video['id'],

                'title': video['snippet']['title'],

                'description': video['snippet'].get('description', 'N/A'),

                'tags': ', '.join(video['snippet'].get('tags', [])),  # Converte lista de tags para string

                'published\_at': video['snippet']['publishedAt'],

                'duration': video['contentDetails']['duration'],

                'category\_id': video['snippet']['categoryId'],

                'views': int(video['statistics'].get('viewCount', 0)),

                'likes': int(video['statistics'].get('likeCount', 0)),

                'comments': int(video['statistics'].get('commentCount', 0))

            }

            all\_video\_stats.append(video\_stats)

    return pd.DataFrame(all\_video\_stats)

def get\_channel\_activity(youtube, playlist\_id):

    video\_details = get\_video\_details(youtube, get\_video\_ids(youtube, playlist\_id), "")

    # Convert published\_at to datetime

    video\_details['published\_at'] = pd.to\_datetime(video\_details['published\_at'])

    # Calculate monthly upload frequency

    video\_details['month'] = video\_details['published\_at'].dt.to\_period('M')

    monthly\_uploads = video\_details.groupby(['channel\_name', 'month']).size().reset\_index(name='uploads')

    # Calculate average engagement

    avg\_engagement = video\_details[['views', 'likes', 'comments']].mean().reset\_index()

    avg\_engagement.columns = ['metric', 'average']

    return monthly\_uploads, avg\_engagement

# IDs de canais para análise (substitua pelos IDs dos canais desejados)

channel\_ids = ['UCy9xDEfMD8jrXU3NatFLLtw', 'UCNWlx\_0XxpUkci6PnCwH\_dQ', 'UC5qJINQ80-zvtGQ4fq3kSRQ']

# Obter estatísticas dos canais

channel\_stats = get\_channel\_stats(youtube, channel\_ids)

# Inicializar uma lista para coletar detalhes dos vídeos de todos os canais

all\_video\_details = []

all\_channel\_activities = []

# Obter IDs dos vídeos e detalhes dos vídeos de cada canal

for index, row in channel\_stats.iterrows():

    playlist\_id = row['playlist\_id']

    channel\_name = row['channel\_name']

    video\_ids = get\_video\_ids(youtube, playlist\_id)

    video\_details = get\_video\_details(youtube, video\_ids, channel\_name)

    all\_video\_details.append(video\_details)

    # Obter atividade do canal

    monthly\_uploads, avg\_engagement = get\_channel\_activity(youtube, playlist\_id)

    monthly\_uploads['channel\_name'] = channel\_name

    avg\_engagement['channel\_name'] = channel\_name

    all\_channel\_activities.append((monthly\_uploads, avg\_engagement))

# Combinar todos os detalhes dos vídeos em um único DataFrame

all\_video\_details\_df = pd.concat(all\_video\_details, ignore\_index=True)

# Combinar atividades dos canais

monthly\_uploads\_df = pd.concat([x[0] for x in all\_channel\_activities], ignore\_index=True)

avg\_engagement\_df = pd.concat([x[1] for x in all\_channel\_activities], ignore\_index=True)

# Exibir os dados dos canais

print(channel\_stats)

# Exibir os detalhes dos vídeos

print(all\_video\_details\_df)

# Exibir os uploads mensais

print(monthly\_uploads\_df)

# Exibir o engajamento médio

print(avg\_engagement\_df)

# Salvar os dados em arquivos Excel

channel\_stats.to\_excel('channel\_stats.xlsx', index=False)

all\_video\_details\_df.to\_excel('video\_details.xlsx', index=False)

monthly\_uploads\_df.to\_excel('monthly\_uploads.xlsx', index=False)

avg\_engagement\_df.to\_excel('avg\_engagement.xlsx', index=False)

print("Dados exportados para 'channel\_stats.xlsx', 'video\_details.xlsx', 'monthly\_uploads.xlsx' e 'avg\_engagement.xlsx'.")

Para extrair o channel\_ids foi necessário a utilização de um código:

from googleapiclient.discovery import build

# Substitua 'YOUR\_API\_KEY' pela sua chave de API do YouTube

api\_key = 'AIzaSyBf6vTVrP0lMvpVjb6CqrGeNdo8p6hYtKE'

# Construa o serviço da API do YouTube

youtube = build('youtube', 'v3', developerKey=api\_key)

def get\_channel\_id(youtube, custom\_url):

    request = youtube.search().list(

                part='snippet',

                q=custom\_url,

                type='channel',

                maxResults=1)

    response = request.execute()

    for item in response['items']:

        return item['snippet']['channelId']

    return None

# URL personalizada do canal

custom\_url = 'https://www.youtube.com/@depoisdosquinze'

# Obter ID do canal

channel\_id = get\_channel\_id(youtube, custom\_url)

print(f"ID do canal: {channel\_id}")

**Geração de Relatórios**

Os dados extraídos foram organizados e salvos tabelas distintas para facilitar a análise:

* **channel\_stats.xlsx**: Contém estatísticas gerais dos canais, como número de inscritos, visualizações totais e total de vídeos.
* **video\_details.xlsx**: Inclui informações detalhadas sobre os vídeos, como título, descrição, tags, data de publicação, duração, e métricas de engajamento (visualizações, curtidas, comentários).
* **monthly\_uploads.xlsx:** Apresenta a frequência de uploads mensais para cada canal.
* **avg\_engagement.xlsx:** Mostra a média de engajamento dos vídeos, calculando a média de visualizações, curtidas e comentários.

**Resultados e Conclusões**

Com base nas tabelas geradas, será possível realizar análises comparativas entre canais e vídeos, identificando padrões de desempenho e características comuns. Essa análise irá fornecer uma visão detalhada sobre a presença e atividade dos canais no YouTube, oferecendo insights valiosos para estratégias de conteúdo e engajamento.

https://github.com/kevelincode/Projeto-final-Python.git