📘 Project Title : YouTube Trending Video Analytics

• **Intern Name Chatap Nilesh Chandrakant**
• **Company Name: ELEVATE LAB**
• **Domain of Internship: Data Analytics**
• **Internship Duration: 45 Days**
• **Project Mentor / Supervisor: HR-Team Elevate Labs**

🧾 Abstract :

This project focuses on analyzing YouTube trending video data to understand viewing patterns and audience engagement across different regions. Using Python, the data is cleaned, stored, and analyzed to extract insights such as top categories and sentiment trends. Visualization tools such as Matplotlib and Plotly are used to present data interactively. The project concludes with a dashboard that provides a complete analytical overview of YouTube's trending ecosystem.

🎯 Objective :

- To identify trends in YouTube videos across different regions.
- To analyze the sentiment of video titles and tags.
- To visualize engagement metrics such as views, likes, and comments.
- To create an interactive dashboard for data insights.

🧰 Tools and Technologies Used :

Tool / Library	Purpose
Python	Programming Language
Pandas, NumPy	Data Cleaning & Analysis
SQLite3	Database Storage
Matplotlib, Plotly	Data Visualization

Tool / Library	Purpose
Dash	Web-based Dashboard

⚙️ Steps Involved :

1. Data Collection: Load datasets (India and USA YouTube Trending data).
2. Data Cleaning: Handle missing data, standardize dates, and remove duplicates.
3. Feature Engineering: Calculate trending duration and sentiment scores.
4. Database Storage: Store cleaned data in SQLite database.
5. Visualization: Create visual insights and plots.
6. Dashboard: Build an interactive dashboard with Dash and Plotly.

</> CODE :

```
import os
import sqlite3
from datetime import datetime, timedelta

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from dash import Dash, dcc, html
import plotly.express as px

# Define file paths
CLEAN_CSV_PATH = "Cleaned_Youtube_Trending_sample.csv"
DB_PATH = "youtube_trending_sample.db"
FIG1_PATH = "avg_views_by_category_region.png"
FIG2_PATH = "sentiment_distribution_region.png"
FIG3_PATH = "trending_duration_ts.png"

# File names for real datasets
US_CSV = "USvideos.csv"
IN_CSV = "INvideos.csv"
```

```
# Check if real YouTube datasets exist
use_real_files = os.path.exists(US_CSV) and os.path.exists(IN_CSV)

# Function to create sample YouTube-like dataset
def make_sample_df(region, start_id=0):
    np.random.seed(0 if region == "USA" else 1)
    n = 10
    ids = [f"{region[:2]}vid{i+start_id}" for i in range(n)]
    titles = [
        "Amazing music video release",
        "Top 10 funny moments",
        "Breaking news: market update",
        "Epic gaming highlights",
        "Healthy recipe tutorial",
        "Emotional short film",
        "Live concert full show",
        "Tech review: new phone",
        "Daily vlog: travel diary",
        "Sports match highlights"
    ]
    channels = [f"{region}channel{i%5}" for i in range(n)]
    categories = [10, 24, 25, 20, 26, 1, 10, 28, 22, 17]
    base_publish = datetime(2025, 10, 1) if region == "USA" else datetime(2025, 9, 20)
    publish_times = [(base_publish + timedelta(days=int(i*2))).isoformat() for i in range(n)]
    trending_dates = [(base_publish + timedelta(days=int(i*2) + np.random.randint(0,
5))).isoformat() for i in range(n)]
    views = np.random.randint(10000, 2000000, size=n)
    likes = (views * np.random.uniform(0.01, 0.05, size=n)).astype(int)
    comment_count = (likes * np.random.uniform(0.1, 0.5, size=n)).astype(int)
    tags = [
        "music|pop", "funny|comedy", "news|finance", "gaming|highlights", "cooking|recipe",
        "shortfilm|emotional", "concert|live", "tech|review", "vlog|travel", "sports|highlights"
    ]
    df = pd.DataFrame({
        "video_id": ids,
        "title": titles,
        "channel_title": channels,
        "category_id": categories,
        "publish_time": publish_times,
        "trending_date": trending_dates,
        "views": views,
        "likes": likes,
        "comment_count": comment_count,
        "tags": tags
    })
    df["region"] = region
    return df
```

```
# Load data: real or sample
if use_real_files:
    us_df = pd.read_csv(US_CSV)
    in_df = pd.read_csv(IN_CSV)
    us_df["region"] = "USA"
    in_df["region"] = "India"
else:
    print("No real data found — generating sample data.")
    us_df = make_sample_df("USA", start_id=0)
    in_df = make_sample_df("India", start_id=100)

# Combine US and India datasets
df = pd.concat([us_df, in_df], ignore_index=True)

# Ensure required columns exist
expected_cols = ["video_id", "title", "channel_title", "category_id", "publish_time",
                 "trending_date", "views", "likes", "comment_count", "tags", "region"]
for c in expected_cols:
    if c not in df.columns:
        df[c] = pd.NA

# Convert time columns to datetime
df["publish_time"] = pd.to_datetime(df["publish_time"], errors="coerce")
df["trending_date"] = pd.to_datetime(df["trending_date"], errors="coerce")

# Add extra useful columns
df["publish_date"] = df["publish_time"].dt.date
df["trending_duration_days"] = (df["trending_date"] - df["publish_time"]).dt.days

# Remove duplicates and missing titles/views
df.drop_duplicates(subset=["video_id", "region"], inplace=True)
df = df.dropna(subset=["title", "views"])

# Reorder columns for readability
cols_order = ["video_id", "title", "channel_title", "category_id", "region",
              "publish_time", "publish_date", "trending_date", "trending_duration_days",
              "views", "likes", "comment_count", "tags"]
df = df[[c for c in cols_order if c in df.columns]]

# Define simple sentiment word sets
positive_words = {"amazing", "top", "funny", "epic", "healthy", "emotional", "live", "new", "best",
"full"}
negative_words = {"breaking", "warning", "scandal", "crash", "lost"}

# Function to calculate basic sentiment score
def simple_sentiment(text):
    if pd.isna(text) or not str(text).strip():
        return 0.0
    txt = str(text).lower()
    score = 0
    for w in positive_words:
        if w in txt:
            score += 1
    for w in negative_words:
        if w in txt:
            score -= 1
    tokens = max(1, len(txt.split()))
    return score / tokens

# Apply sentiment analysis on titles and tags
df["title_sentiment_score"] = df["title"].apply(simple_sentiment)
df["tags_sentiment_score"] = df["tags"].apply(simple_sentiment)

# Function to label sentiment categories
def label_sentiment(s):
    if s > 0.02:
        return "Positive"
    elif s < -0.02:
        return "Negative"
    else:
        return "Neutral"

# Add sentiment labels to DataFrame
df["title_sentiment_label"] = df["title_sentiment_score"].apply(label_sentiment)
df["tags_sentiment_label"] = df["tags_sentiment_score"].apply(label_sentiment)

# Save cleaned dataset and write to SQLite database
df.to_csv(CLEAN_CSV_PATH, index=False)
conn = sqlite3.connect(DB_PATH)
df.to_sql("youtube_data", conn, if_exists="replace", index=False)

# Query average views and count by category and region
query = """
SELECT category_id, region, ROUND(AVG(views), 0) AS avg_views, COUNT(video_id) AS
total_videos
FROM youtube_data
GROUP BY category_id, region
ORDER BY avg_views DESC;
"""
rank_df = pd.read_sql_query(query, conn)
conn.close()

# Create a bar plot: average views by category and region
pivot = rank_df.pivot(index="category_id", columns="region", values="avg_views").fillna(0)
pivot.plot(kind="bar", figsize=(10, 5))
```

```
plt.title("Average Views by Category and Region")
plt.xlabel("Category ID")
plt.ylabel("Average Views")
plt.tight_layout()
plt.savefig(FIG1_PATH)
plt.close()

# Bar chart for sentiment distribution by region
sent_counts = df.groupby(["region", "title_sentiment_label"]).size().unstack(fill_value=0)
sent_counts.plot(kind="bar", figsize=(7, 4))
plt.title("Title Sentiment Distribution by Region")
plt.xlabel("Region")
plt.ylabel("Count")
plt.tight_layout()
plt.savefig(FIG2_PATH)
plt.close()

# Line chart for trending duration trend over time
df["publish_date_dt"] = pd.to_datetime(df["publish_date"])
duration_ts = df.groupby(["region",
"publish_date_dt"])["trending_duration_days"].mean().reset_index()

for region in duration_ts["region"].unique():
  sub = duration_ts[duration_ts["region"] == region].sort_values("publish_date_dt")
  plt.plot(sub["publish_date_dt"], sub["trending_duration_days"], marker='o', label=region)
plt.title("Average Trending Duration Over Time")
plt.xlabel("Publish Date")
plt.ylabel("Trending Duration (days)")
plt.legend()
plt.tight_layout()
plt.savefig(FIG3_PATH)
plt.close()

# Create Dash web application
app = Dash(__name__)

# Plotly bar chart for average views
fig1 = px.bar(
  rank_df, x="category_id", y="avg_views", color="region",
  barmode="group", title="Average Views by Category and Region"
)

# Plotly histogram for sentiment distribution
fig2 = px.histogram(
  df, x="title_sentiment_label", color="region",
  title="Sentiment Distribution by Region"
)
```

```
# Plotly line chart for trending duration
fig3 = px.line(
    duration_ts, x="publish_date_dt", y="trending_duration_days", color="region",
    title="Average Trending Duration Over Time"
)

# Define dashboard layout
app.layout = html.Div([
    html.H1("🎬 YouTube Trending Video Analytics Dashboard", style={'textAlign': 'center'}),
    dcc.Graph(figure=fig1),
    dcc.Graph(figure=fig2),
    dcc.Graph(figure=fig3)
])

# Run the Dash app
if __name__ == "__main__":
    app.run(debug=True, port=8050)
```

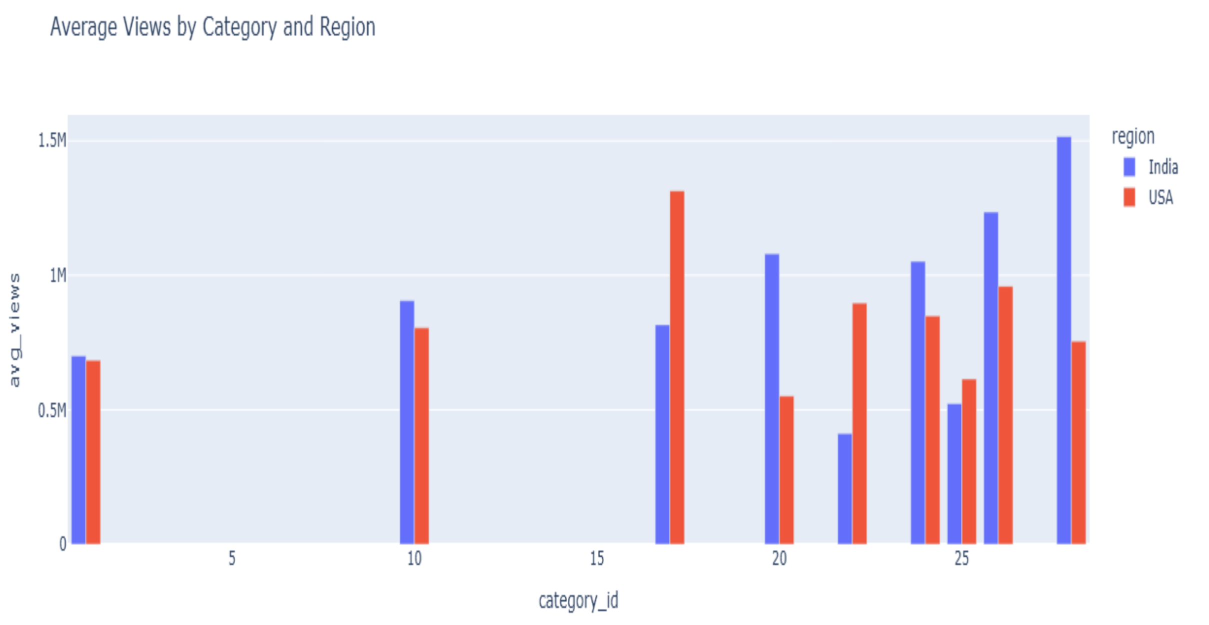
📊 Output and Visualization

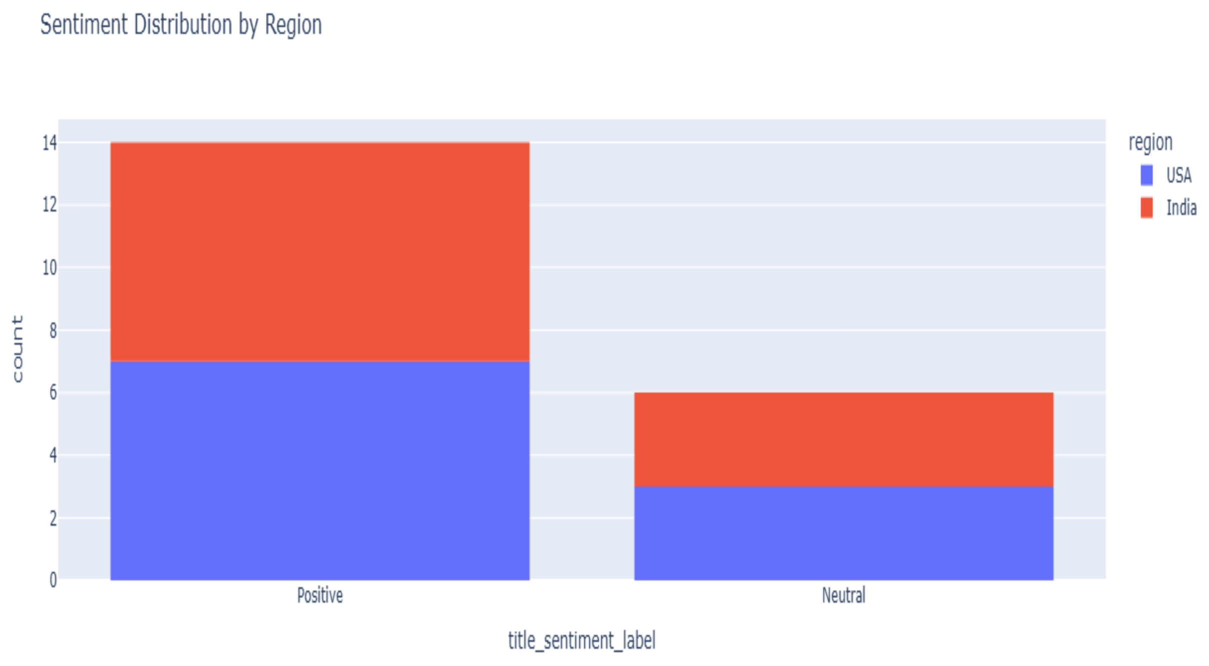
1. **Average Views by Category and Region**
   (Insert the image: avg_views_by_category_region.png)
2. **Sentiment Distribution by Region**
   (Insert the image: sentiment_distribution_region.png)
3. **Average Trending Duration Over Time**
   (Insert the image: trending_duration_ts.png)

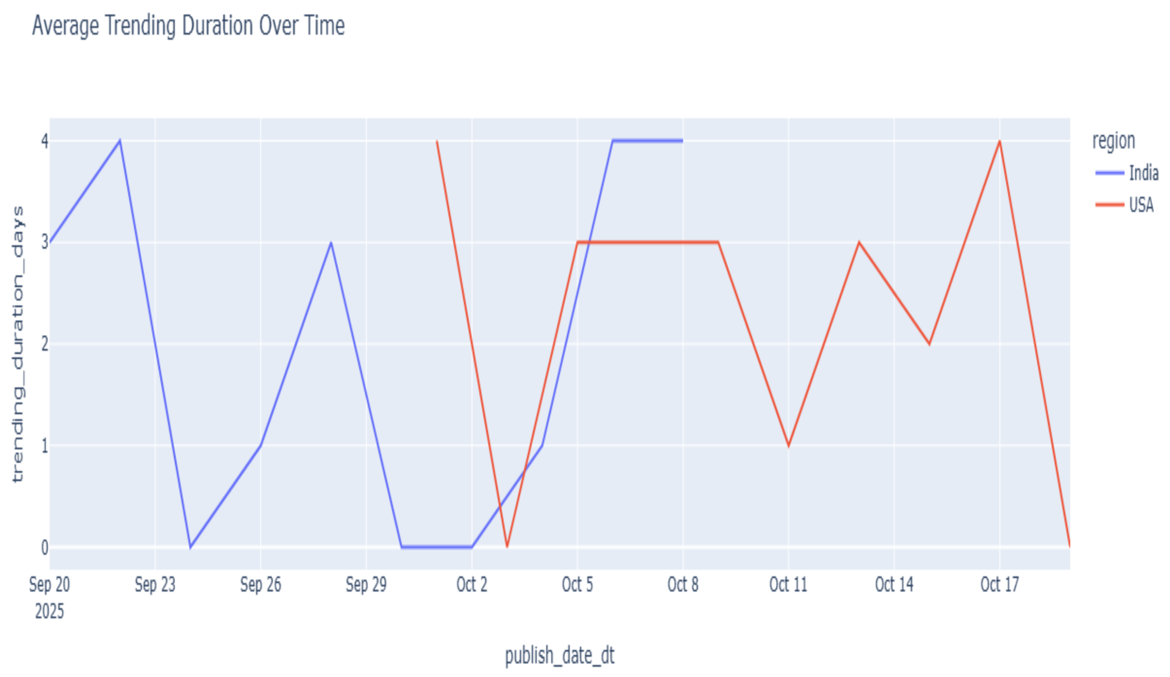
📈 Output :

```
No real data found — generating sample data.

Dash is running on http://127.0.0.1:8050/

 * Serving Flask app 'project'

 * Debug mode: on        No real data found — generating sample data.
```

🎬 YouTube Trending Video Analytics Dashboard







🚀 Future Scope :

- Integrate YouTube API for real-time data.
- Use Machine Learning for advanced sentiment prediction.
- Expand analysis to other platforms.
- Deploy the dashboard online (e.g., Render, Heroku).

🧩 Conclusion :

The project successfully demonstrates the process of analyzing YouTube's trending video data using Python, SQL, and visualization tools. It provides valuable insights into content trends, sentiment, and audience behavior.

📚 References :

1. YouTube Trending Datasets (Kaggle)
2. Python Official Documentation
3. Plotly & Dash Documentation
4. SQLite Documentation