# YouTube Trending Video Analytics – Project Pack

\*\*Objective:\*\* Uncover patterns in trending videos by analyzing YouTube datasets across regions.

\*\*Tools:\*\* Python (pandas, NumPy, Matplotlib, Seaborn, NLTK/VADER), SQL (SQLite/Postgres/BigQuery), Tableau

---

## 1) Project Plan (End-to-End)

\*\*A. Data Sourcing\*\*

\* Use Kaggle’s “Trending YouTube Video Statistics” dataset or YouTube API (v3) if you prefer fresh data.

\* Countries commonly available: US, GB, CA, DE, FR, IN, JP, KR, MX, RU (varies by source). Each country has a CSV with daily snapshots + a shared `category\_id` mapping JSON.

\*\*B. Data Ingestion & Standardization\*\*

\* Standardize column names and types across regions (e.g., `publish\_time` to UTC datetime, `views` as int64).

\* Parse JSON columns (`category\_id` mapping) and join to attach human-readable category names.

\* Normalize tags (split on `|`, lower-case, strip whitespace).

\* Create a unified fact table with a `region` column.

\*\*C. Cleaning Rules\*\*

\* Remove rows with missing critical fields (`video\_id`, `title`, `views`).

\* Deduplicate by `video\_id + trending\_date + region`.

\* Handle outliers: Winsorize top 0.5% of `views`/`likes`/`comments` per region for visualization stability.

\* Convert booleans: `comments\_disabled`, `ratings\_disabled`, `video\_error\_or\_removed` to True/False.

\*\*D. Feature Engineering\*\*

\* Trending duration for each `video\_id` in a region = number of unique `trending\_date` rows.

\* Engagement rate = (`likes` + `comment\_count`) / `views`.

\* Like ratio = `likes`/( `likes`+`dislikes` ) if dislikes available (legacy datasets only).

\* Upload weekday/hour (from `publish\_time`), trending weekday (from `trending\_date`).

\* Title length, tag count, presence of ALL-CAPS, presence of emojis.

\*\*E. Sentiment Analysis (Titles & Tags)\*\*

\* Use \*\*VADER\*\* (rule-based, good for short text). Compute compound scores for `title` and for the top-N tags (joined as a single string per video).

\* Bucket compound into \*\*Negative\*\* (≤ -0.05), \*\*Neutral\*\* (-0.05 to 0.05), \*\*Positive\*\* (≥ 0.05).

\*\*F. SQL Analytics\*\*

\* Use SQL to create category rankings by average views/engagement, region-wise comparisons, and trend duration distributions.

\*\*G. Visualization\*\*

\* Python (Matplotlib/Seaborn) for EDA and time-series sanity checks.

\* Tableau for interactive dashboards: genre popularity, sentiment distributions, region comparisons, trending duration.

\*\*H. Storytelling & Report\*\*

\* Focus on how categories perform across regions, how sentiment correlates with views and engagement, and timing patterns (publish hour/day vs. trending likelihood).

---

## 2) Suggested Folder Structure

```

yt-trending-analytics/

├─ data/

│ ├─ raw/ # per-country CSVs + category JSONs

│ ├─ interim/ # cleaned per-country CSVs

│ └─ processed/ # unified dataset(s)

├─ notebooks/

│ ├─ 01\_eda.ipynb

│ ├─ 02\_clean\_standardize.ipynb

│ ├─ 03\_sentiment\_features.ipynb

│ └─ 04\_time\_series\_trending\_duration.ipynb

├─ sql/

│ ├─ schema.sql

│ ├─ transforms.sql

│ └─ analysis.sql

├─ src/

│ ├─ config.py

│ ├─ ingest.py

│ ├─ clean\_standardize.py

│ ├─ build\_features.py

│ └─ utils.py

├─ tableau/

│ └─ workbook.twbx # to be created in Tableau

├─ reports/

│ └─ final\_report.docx # export target

└─ README.md

```

---

## 3) Data Schema (Analytical)

\*\*Table: `videos\_trending` (grain: video x trending\\_date x region)\*\*

\* `video\_id` TEXT

\* `trending\_date` DATE (YYYY-MM-DD)

\* `publish\_time` TIMESTAMP (UTC)

\* `region` TEXT (ISO country code)

\* `title` TEXT

\* `channel\_title` TEXT

\* `category\_id` INT

\* `category\_name` TEXT

\* `tags` TEXT (pipe-delimited in raw; normalized to array in SQL engines that support it)

\* `views` BIGINT

\* `likes` BIGINT

\* `dislikes` BIGINT (may be missing in newer datasets)

\* `comment\_count` BIGINT

\* `comments\_disabled` BOOLEAN

\* `ratings\_disabled` BOOLEAN

\* `video\_error\_or\_removed` BOOLEAN

\* `description` TEXT

\*\*Derived Table: `videos\_daily\_features`\*\*

\* All of the above plus engineered columns:

\* `title\_len`, `tag\_count`, `has\_caps`, `has\_emoji`

\* `publish\_hour`, `publish\_weekday`, `trending\_weekday`

\* `engagement\_rate`

\* `vader\_title`, `vader\_tags`, `sentiment\_bucket\_title`, `sentiment\_bucket\_tags`

\*\*Aggregate Table: `videos\_agg` (grain: video x region)\*\*

\* `video\_id`, `region`, `category\_name`

\* `first\_trending\_date`, `last\_trending\_date`, `trending\_days`

\* `avg\_views`, `median\_views`, `avg\_engagement\_rate`

\* `max\_views\_day` (views on top trending day)

\* `vader\_title`, `vader\_tags`, `sentiment\_bucket\_title`, `sentiment\_bucket\_tags`

---

## 4) SQL – Schema & Transforms (example in PostgreSQL)

\*\*`schema.sql`\*\*

```sql

CREATE TABLE IF NOT EXISTS videos\_trending (

video\_id TEXT,

trending\_date DATE,

publish\_time TIMESTAMP,

region TEXT,

title TEXT,

channel\_title TEXT,

category\_id INT,

category\_name TEXT,

tags TEXT,

views BIGINT,

likes BIGINT,

dislikes BIGINT,

comment\_count BIGINT,

comments\_disabled BOOLEAN,

ratings\_disabled BOOLEAN,

video\_error\_or\_removed BOOLEAN,

description TEXT

);

```

\*\*`transforms.sql`\*\*

```sql

-- 1) Trending duration per video per region

CREATE MATERIALIZED VIEW IF NOT EXISTS mv\_trending\_duration AS

SELECT

video\_id,

region,

MIN(trending\_date) AS first\_trending\_date,

MAX(trending\_date) AS last\_trending\_date,

COUNT(DISTINCT trending\_date) AS trending\_days

FROM videos\_trending

GROUP BY 1,2;

-- 2) Category rank by average views (per region)

CREATE MATERIALIZED VIEW IF NOT EXISTS mv\_category\_avg\_views AS

SELECT

region,

category\_name,

AVG(views)::BIGINT AS avg\_views,

RANK() OVER (PARTITION BY region ORDER BY AVG(views) DESC) AS rank\_in\_region

FROM videos\_trending

GROUP BY 1,2;

-- 3) Engagement rate per row

CREATE MATERIALIZED VIEW IF NOT EXISTS mv\_daily\_features AS

SELECT

\*,

CASE WHEN views > 0 THEN (COALESCE(likes,0) + COALESCE(comment\_count,0))::FLOAT / views ELSE NULL END AS engagement\_rate,

EXTRACT(HOUR FROM publish\_time) AS publish\_hour,

TO\_CHAR(publish\_time, 'Dy') AS publish\_weekday,

TO\_CHAR(trending\_date, 'Dy') AS trending\_weekday

FROM videos\_trending;

-- 4) Aggregates by video & region

CREATE MATERIALIZED VIEW IF NOT EXISTS mv\_video\_agg AS

SELECT

vt.video\_id,

vt.region,

MAX(vt.category\_name) AS category\_name,

MIN(vt.trending\_date) AS first\_trending\_date,

MAX(vt.trending\_date) AS last\_trending\_date,

COUNT(DISTINCT vt.trending\_date) AS trending\_days,

AVG(vt.views)::BIGINT AS avg\_views,

PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY vt.views) AS median\_views,

MAX(vt.views) AS max\_views\_day

FROM videos\_trending vt

GROUP BY 1,2;

```

\*\*`analysis.sql`\*\*

```sql

-- A) Rank categories by average views (global)

SELECT category\_name,

AVG(views) AS avg\_views,

RANK() OVER (ORDER BY AVG(views) DESC) AS global\_rank

FROM videos\_trending

GROUP BY category\_name;

-- B) Region-wise comparison: top 5 categories by avg views in each region

WITH cat\_rank AS (

SELECT region, category\_name, AVG(views) AS avg\_views,

RANK() OVER (PARTITION BY region ORDER BY AVG(views) DESC) AS r

FROM videos\_trending

GROUP BY region, category\_name

)

SELECT \* FROM cat\_rank WHERE r <= 5 ORDER BY region, r;

-- C) Trending duration distribution per region

SELECT region,

NTILE(10) OVER (PARTITION BY region ORDER BY trending\_days) AS decile,

AVG(trending\_days) AS avg\_days

FROM mv\_trending\_duration;

-- D) Correlate sentiment buckets (after you add columns) with avg views

-- Assuming columns sentiment\_bucket\_title exist in videos\_trending or a joined table

SELECT region, sentiment\_bucket\_title, AVG(views) AS avg\_views

FROM videos\_trending

GROUP BY region, sentiment\_bucket\_title

ORDER BY region, avg\_views DESC;

```

---

## 5) Python – Cleaning & Sentiment (ready-to-run snippets)

\*\*Install\*\*

```bash

pip install pandas numpy matplotlib seaborn nltk vaderSentiment pyarrow

python -m nltk.downloader vader\_lexicon

```

\*\*`src/clean\_standardize.py`\*\*

```python

import pandas as pd

import numpy as np

from pathlib import Path

RENAME\_MAP = {

'video\_id': 'video\_id',

'trending\_date': 'trending\_date',

'publish\_time': 'publish\_time',

'channel\_title': 'channel\_title',

'category\_id': 'category\_id',

'tags': 'tags',

'views': 'views',

'likes': 'likes',

'dislikes': 'dislikes',

'comment\_count': 'comment\_count',

'comments\_disabled': 'comments\_disabled',

'ratings\_disabled': 'ratings\_disabled',

'video\_error\_or\_removed': 'video\_error\_or\_removed',

'description': 'description',

'title': 'title'

}

DTYPES = {

'views': 'Int64',

'likes': 'Int64',

'dislikes': 'Int64',

'comment\_count': 'Int64',

}

def load\_and\_clean(file\_path: str, region: str) -> pd.DataFrame:

df = pd.read\_csv(file\_path)

df = df.rename(columns=RENAME\_MAP)

df['region'] = region

# parse dates

df['trending\_date'] = pd.to\_datetime(df['trending\_date'], errors='coerce')

df['publish\_time'] = pd.to\_datetime(df['publish\_time'], errors='coerce', utc=True)

# normalize tags

df['tags'] = df['tags'].fillna('')

df['tags\_norm'] = df['tags'].str.replace('"', '', regex=False).str.lower()

# feature helpers

df['title\_len'] = df['title'].fillna('').str.len()

df['tag\_count'] = df['tags\_norm'].apply(lambda x: 0 if x in ('[none]', '') else len(x.split('|')))

df['has\_caps'] = df['title'].fillna('').str.contains(r'[A-Z]{3,}', regex=True)

df['has\_emoji'] = df['title'].fillna('').apply(lambda s: any(ord(c) > 10000 for c in s))

# cast numerics

for c in ['views','likes','dislikes','comment\_count']:

if c in df.columns:

df[c] = pd.to\_numeric(df[c], errors='coerce')

# drop dupes

df = df.drop\_duplicates(subset=['video\_id','trending\_date','region'])

return df

def unify\_to\_parquet(raw\_dir='data/raw', out\_dir='data/processed/unified.parquet'):

raw = Path(raw\_dir)

frames = []

for csv\_file in raw.glob('\*.csv'):

region = csv\_file.stem.split('\_')[-1].upper() # e.g., US from ...\_US.csv

frames.append(load\_and\_clean(str(csv\_file), region))

df = pd.concat(frames, ignore\_index=True)

# compute engagement rate & time parts

df['engagement\_rate'] = np.where(df['views'] > 0, (df['likes'].fillna(0) + df['comment\_count'].fillna(0)) / df['views'], np.nan)

df['publish\_hour'] = df['publish\_time'].dt.hour

df['publish\_weekday'] = df['publish\_time'].dt.day\_name()

df['trending\_weekday'] = df['trending\_date'].dt.day\_name()

Path(out\_dir).parent.mkdir(parents=True, exist\_ok=True)

df.to\_parquet(out\_dir, index=False)

print(f"Wrote {out\_dir} with {len(df):,} rows")

if \_\_name\_\_ == '\_\_main\_\_':

unify\_to\_parquet()

```

\*\*`src/build\_features.py`\*\* (VADER sentiment)

```python

import pandas as pd

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

from pathlib import Path

SENT\_THRESH\_NEG = -0.05

SENT\_THRESH\_POS = 0.05

def bucketize(compound: float) -> str:

if compound is None:

return 'Neutral'

if compound <= SENT\_THRESH\_NEG:

return 'Negative'

if compound >= SENT\_THRESH\_POS:

return 'Positive'

return 'Neutral'

def add\_sentiment(parquet\_in='data/processed/unified.parquet', parquet\_out='data/processed/with\_sentiment.parquet'):

df = pd.read\_parquet(parquet\_in)

an = SentimentIntensityAnalyzer()

df['vader\_title'] = df['title'].fillna('').apply(lambda s: an.polarity\_scores(s)['compound'])

df['tags\_join'] = df['tags\_norm'].str.replace('|', ' ', regex=False)

df['vader\_tags'] = df['tags\_join'].fillna('').apply(lambda s: an.polarity\_scores(s)['compound'])

df['sentiment\_bucket\_title'] = df['vader\_title'].apply(bucketize)

df['sentiment\_bucket\_tags'] = df['vader\_tags'].apply(bucketize)

Path(parquet\_out).parent.mkdir(parents=True, exist\_ok=True)

df.to\_parquet(parquet\_out, index=False)

print(f"Wrote {parquet\_out} with sentiment columns")

if \_\_name\_\_ == '\_\_main\_\_':

add\_sentiment()

```

\*\*`notebooks/04\_time\_series\_trending\_duration.ipynb` (logic sketch)\*\*

```python

# Compute trending duration per video per region

import pandas as pd

import matplotlib.pyplot as plt

# load

df = pd.read\_parquet('data/processed/with\_sentiment.parquet')

# duration

duration = (

df.groupby(['video\_id','region'])['trending\_date']

.nunique()

.reset\_index(name='trending\_days')

)

# plot distribution per region

ax = duration.boxplot(by='region', column=['trending\_days'])

plt.title('Trending Duration by Region')

plt.suptitle('')

plt.xlabel('Region')

plt.ylabel('Days trending')

plt.show()

```

\*\*Python – Region-wise Comparison Visuals (examples)\*\*

```python

import pandas as pd

import matplotlib.pyplot as plt

agg = (df

.groupby(['region','category\_name'])['views']

.mean()

.reset\_index(name='avg\_views'))

# top categories per region

top = agg.sort\_values(['region','avg\_views'], ascending=[True, False])

for region, sub in top.groupby('region'):

head = sub.head(10)

head.plot.bar(x='category\_name', y='avg\_views', title=f'Top Categories by Avg Views – {region}')

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()

```

---

## 6) Tableau Dashboard Spec

\*\*Workbook Pages\*\*

1. \*\*Overview (KPIs)\*\*

\* Cards: Total Videos, Unique Channels, Median Views (per region), Avg Trending Days

\* Map: Regions with total trending videos

\* Bar: Top Categories by Avg Views (filter by region)

2. \*\*Sentiment & Performance\*\*

\* Histogram / Violin: Distribution of VADER title scores

\* Stacked bar: Sentiment bucket share by region

\* Scatter: VADER title vs. Avg Views (log scale option), colored by category

3. \*\*Trending Duration\*\*

\* Box plot: Trending days by region

\* Line: Count of videos trending by date (time-series)

\* Heatmap: Publish hour (x) vs. Avg Views (y) by region

4. \*\*Region Comparison\*\*

\* Small multiples: Top 5 categories per region

\* Toggle: Normalize by population (optional, if you add external data)

\*\*Filters & Interactivity\*\*

\* Region multi-select, Category multi-select, Date range, Sentiment bucket

\* Highlighter for `channel\_title`

\*\*Data Sources in Tableau\*\*

\* `videos\_agg` for high-level, `videos\_trending` for row-level time series, `videos\_daily\_features` for hour/weekday.

---

## 7) Data Storytelling Template (Narrative Arc)

\*\*Executive Summary (½ page)\*\*

\* 3–5 bullets on key findings: top genres globally, standout regions, sentiment patterns, timing insights.

\*\*Methodology (1–2 pages)\*\*

\* Datasets & coverage period, cleaning & standardization, sentiment method (VADER), SQL logic for category ranking & trending duration.

\*\*Findings (3–5 pages)\*\*

\* \*\*Genre Popularity:\*\* Which categories dominate globally? Any region-specific favorites?

\* \*\*Sentiment vs. Performance:\*\* Do positive/negative titles correlate with higher views/engagement? Note caveats.

\* \*\*Timing Effects:\*\* Publish hour/weekday patterns and the path to trending; how long do videos stay trending by region?

\* \*\*Regional Contrasts:\*\* Side-by-side charts and takeaways; call out anomalies (e.g., a category strong in one region but weak elsewhere).

\*\*Recommendations (½–1 page)\*\*

\* Content strategy: optimal categories, tone (title/tag sentiment), and posting windows by region.

\*\*Limitations & Next Steps (½ page)\*\*

\* Dataset biases (not all videos; shifting YouTube policies like hidden dislikes), alternative sentiment models (transformers), adding channel-level controls.

\*\*Appendix\*\*

\* Table dictionary, SQL snippets, feature definitions.

---

## 8) Deliverables Checklist

\* [ ] \*\*Cleaned & unified dataset\*\* (`with\_sentiment.parquet`).

\* [ ] \*\*SQL views / queries\*\* for category ranking and trending duration.

\* [ ] \*\*Python EDA visuals\*\* (PNG exports) for time series and distributions.

\* [ ] \*\*Tableau dashboard\*\* with four pages above.

\* [ ] \*\*Final report\*\* following storytelling template.

---

## 9) Evaluation Metrics (for the project)

\* Reproducibility: `README.md` + scripts can rebuild processed data from raw.

\* Data Quality: % missing, duplicate rate, schema conformity.

\* Insight Quality: Specific, actionable, supported by visuals.

\* Dashboard Usability: Interactive filters, clear legends, performance (<3s per interaction on sample hardware).

---

## 10) Ready-to-use Snippets for the Report

\*\*Abstract (sample)\*\*

> This study analyzes trending YouTube videos across multiple regions to uncover genre popularity, sentiment dynamics, and timing effects. After standardizing country-level datasets and enriching them with VADER sentiment on titles and tags, we rank categories by average views using SQL and model trending duration as a time-based signal. Our dashboards reveal strong regional heterogeneity and suggest practical publishing strategies.

\*\*Method – Sentiment (sample)\*\*

> We apply the VADER lexicon to video titles and tag strings. We treat compound scores ≥ 0.05 as Positive, ≤ -0.05 as Negative, and the remainder as Neutral. This rule-based method is optimized for short, informal text and is robust to emojis and capitalization.

\*\*Limitations (sample)\*\*

> Dislike counts are not consistently available post-2021, and “trending” reflects platform curation rather than absolute popularity. Future work should consider transformer-based sentiment and causal controls for channel size.

---

## 11) Tips & Pitfalls

\* Always attach `category\_name` via the JSON mapping; `category\_id` alone isn’t interpretable.

\* Convert `publish\_time` to UTC and derive local time if you want per-country hour-of-day comparisons.

\* Watch for the string `"[none]"` in tags; treat as missing.

\* Winsorize extreme outliers to keep charts readable; note it in the methodology.

\* Cache interim outputs (Parquet) to speed up iteration.

---

## 12) Next Steps You Can Take Now

1. Download per-country CSVs to `data/raw/`.

2. Run `python src/clean\_standardize.py` then `python src/build\_features.py`.

3. Load `with\_sentiment.parquet` into your SQL engine (or use DuckDB for simplicity) and execute `sql/transforms.sql` then `sql/analysis.sql`.

4. Connect Tableau to the processed data and build the four dashboard pages using the spec.

5. Draft the report with the provided storytelling template and paste in the key visuals.