YouTubeStandardization

May 7, 2025

0.0.1 Clean and standardize YouTube trending datasets from different countries

0.1 Load & Inspect Dataset:Global_YouTube_Statistics

[3]:
```python
import pandas as pd

# Load the Global YouTube Statistics dataset
df = pd.read_csv(r"C:\Users\91961\Downloads\Global_YouTube_Statistics.csv",␣
 ↪encoding="latin1")

# Show column names & first few rows
print(df.head())
print(df.info())

# Check for missing values
print(df.isnull().sum())
```

```
   rank                    Youtuber  subscribers   video views  \
0     1                    T-Series    245000000  2.280000e+11   
1     2              YouTube Movies    170000000  0.000000e+00   
2     3                     MrBeast    166000000  2.836884e+10   
3     4  Cocomelon - Nursery Rhymes    162000000  1.640000e+11   
4     5                   SET India    159000000  1.480000e+11   

           category                       Title  uploads        Country  \
0             Music                    T-Series    20082          India   
1  Film & Animation               youtubemovies        1  United States   
2     Entertainment                     MrBeast      741  United States   
3         Education  Cocomelon - Nursery Rhymes      966  United States   
4             Shows                   SET India   116536          India   

  Abbreviation   channel_type  …  subscribers_for_last_30_days  \
0           IN          Music  …                     2000000.0   
1           US          Games  …                           NaN   
2           US  Entertainment  …                     8000000.0   
3           US      Education  …                     1000000.0   
4           IN  Entertainment  …                     1000000.0   

   created_year  created_month  created_date  \
0       2006.0            Mar          13.0
1       2006.0            Mar           5.0
2       2012.0            Feb          20.0
3       2006.0            Sep           1.0
4       2006.0            Sep          20.0

   Gross tertiary education enrollment (%)    Population  Unemployment rate  \
0                                     28.1  1.366418e+09               5.36
1                                     88.2  3.282395e+08              14.70
2                                     88.2  3.282395e+08              14.70
3                                     88.2  3.282395e+08              14.70
4                                     28.1  1.366418e+09               5.36

   Urban_population   Latitude  Longitude
0       471031528.0  20.593684  78.962880
1       270663028.0  37.090240 -95.712891
2       270663028.0  37.090240 -95.712891
3       270663028.0  37.090240 -95.712891
4       471031528.0  20.593684  78.962880

[5 rows x 28 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 995 entries, 0 to 994
Data columns (total 28 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   rank                                     995 non-null    int64
 1   Youtuber                                 995 non-null    object
 2   subscribers                              995 non-null    int64
 3   video views                              995 non-null    float64
 4   category                                 949 non-null    object
 5   Title                                    995 non-null    object
 6   uploads                                  995 non-null    int64
 7   Country                                  873 non-null    object
 8   Abbreviation                             873 non-null    object
 9   channel_type                             965 non-null    object
 10  video_views_rank                         994 non-null    float64
 11  country_rank                             879 non-null    float64
 12  channel_type_rank                        962 non-null    float64
 13  video_views_for_the_last_30_days         939 non-null    float64
 14  lowest_monthly_earnings                  995 non-null    float64
 15  highest_monthly_earnings                 995 non-null    float64
 16  lowest_yearly_earnings                   995 non-null    float64
 17  highest_yearly_earnings                  995 non-null    float64
 18  subscribers_for_last_30_days             658 non-null    float64
 19  created_year                             990 non-null    float64
 20  created_month                            990 non-null    object
 21  created_date                             990 non-null    float64
 22  Gross tertiary education enrollment (%)  872 non-null    float64
 23  Population                               872 non-null    float64
 24  Unemployment rate                        872 non-null    float64
 25  Urban_population                         872 non-null    float64
 26  Latitude                                 872 non-null    float64
 27  Longitude                                872 non-null    float64
dtypes: float64(18), int64(3), object(7)
memory usage: 217.8+ KB
None
rank                                         0
Youtuber                                     0
subscribers                                  0
video views                                  0
category                                    46
Title                                        0
uploads                                      0
Country                                    122
Abbreviation                               122
channel_type                                30
video_views_rank                             1
country_rank                               116
channel_type_rank                           33
video_views_for_the_last_30_days            56
lowest_monthly_earnings                      0
highest_monthly_earnings                     0
lowest_yearly_earnings                       0
highest_yearly_earnings                      0
subscribers_for_last_30_days               337
created_year                                 5
created_month                                5
created_date                                 5
Gross tertiary education enrollment (%)    123
Population                                 123
Unemployment rate                          123
Urban_population                           123
Latitude                                   123
Longitude                                  123
dtype: int64
```

0.1.1 Observations from the Dataset

Country and Abbreviation have 122 missing values, which need filling or standardization. category has 46 missing values, which we'll handle appropriately. created_date is stored as float, so we need to convert it to a proper date format. Several columns like subscribers_for_last_30_days, video_views_for_the_last_30_days, Population, and Urban_population also have missing values.

0.1.2 Cleaning & Standardizing Country Names

0.2 Fill Missing Country Names Using Abbreviations

1 Some missing countries might have valid Abbreviation, so we can fill those gaps:

```
[9]: df.loc[df["Country"].isnull(), "Country"] = df["Abbreviation"]
```

1.0.1 Identify Which Countries Are Still Missing

Run this code to check which rows still have missing country names

```
[11]: print(df["Country"].isnull().sum())
```

```
122
```

```
[13]: print(df[df["Country"].isnull()])
```

```
     rank                                            Youtuber  subscribers  \
5       6                                               Music    119000000
12     13                                              Gaming     93600000
14     15                                           Goldmines     86900000
38     39  LooLoo Kids - Nursery Rhymes and Children's Songs     54000000
48     49                                             Badabun     46800000
..    …                                                   …          …
958   959                                        Troom Troom PT     12500000
967   968                                 Troom Troom Indonesia     12500000
972   973                                      Hero Movies 2023     12400000
985   986                                                  TKOR     12400000
986   987                                             ANNA KOVA     12400000

      video views          category  \
5    0.000000e+00               NaN
12   0.000000e+00               NaN
14   2.411823e+10  Film & Animation
38   3.231243e+10             Music
48   1.939805e+10     Entertainment
..            …                 …
958  4.384178e+09     Howto & Style
967  5.379684e+09    People & Blogs
972  1.689091e+09    People & Blogs
985  3.392023e+09         Education
986  1.395959e+10    People & Blogs

                                                  Title  uploads Country  \
5                                                 Music        0     NaN
12                                               Gaming        0     NaN
14                                            goldmines        1     NaN
```

```
38   LooLoo Kids - Nursery Rhymes and Children's ï¿½       11     NaN
48                                           badabun        1     NaN
..                                               …       …     …
958                                   Troom Troom PT     2738     NaN
967                              TROOM TROOM INDONESIA        8     NaN
972                                  Hero Movies 2023      689     NaN
985                                             TKoR        0     NaN
986                                         annakova        1     NaN

    Abbreviation channel_type  …  subscribers_for_last_30_days  \
5            NaN        Music  …                           NaN
12           NaN        Games  …                           NaN
14           NaN        Music  …                           NaN
38           NaN          NaN  …                           NaN
48           NaN        Music  …                          75.0
..             …            …  …                             …
958          NaN        Howto  …                           NaN
967          NaN       People  …                           NaN
972          NaN       People  …                           NaN
985          NaN       People  …                           NaN
986          NaN         Film  …                           NaN

     created_year  created_month  created_date  \
5          2013.0            Sep          24.0
12         2013.0            Dec          15.0
14         2006.0            Aug          15.0
38         2016.0            Nov          29.0
48         2007.0            Jul          21.0
..            …             …            …
958        2015.0            Apr          19.0
967        2020.0            Jul          29.0
972        2017.0            Feb          22.0
985        2006.0            Aug          16.0
986        2006.0            Jun          18.0

     Gross tertiary education enrollment (%)  Population  Unemployment rate  \
5                                        NaN         NaN                NaN
12                                       NaN         NaN                NaN
14                                       NaN         NaN                NaN
38                                       NaN         NaN                NaN
48                                       NaN         NaN                NaN
..                                       …          …                …
958                                      NaN         NaN                NaN
967                                      NaN         NaN                NaN
972                                      NaN         NaN                NaN
985                                      NaN         NaN                NaN
986                                      NaN         NaN                NaN
```

```
     Urban_population  Latitude  Longitude
5                 NaN       NaN        NaN
12                NaN       NaN        NaN
14                NaN       NaN        NaN
38                NaN       NaN        NaN
48                NaN       NaN        NaN
..                ...       ...        ...
958               NaN       NaN        NaN
967               NaN       NaN        NaN
972               NaN       NaN        NaN
985               NaN       NaN        NaN
986               NaN       NaN        NaN

[122 rows x 28 columns]
```

[ ]:
```
Identify Patterns in Missing Country Data
From your output, we observe:  Some missing values belong to categories like␣
 ↪"Music" or "Gaming."
 Abbreviation is also missing in many cases, meaning we can't rely on it to␣
 ↪fill country names.
Let's first count missing values in the Abbreviation column:
```

[15]:
```
print(df["Abbreviation"].isnull().sum())
```

```
122
```

1.0.2 Assign Country Values Using Known Patterns

Since certain YouTubers are well-known in specific countries, we'll manually map them where possible:

[17]:
```
# Assign country values based on known creators
df.loc[df["Title"].str.contains("T-Series", case=False, na=False), "Country"] =␣
 ↪"India"
df.loc[df["Title"].str.contains("MrBeast", case=False, na=False), "Country"] =␣
 ↪"United States"
df.loc[df["Title"].str.contains("SET India", case=False, na=False), "Country"]␣
 ↪= "India"
df.loc[df["Title"].str.contains("Gaming", case=False, na=False), "Country"] =␣
 ↪"Various"
df.loc[df["Title"].str.contains("Music", case=False, na=False), "Country"] =␣
 ↪"Various"
df.loc[df["Title"].str.contains("YouTube Movies", case=False, na=False),␣
 ↪"Country"] = "United States"
```

1.0.3 Assign Unknown for Remaining Missing Values

If some rows still don't have a clear country, we temporarily fill them with "Unknown" so we can manually review later:

[21]:
```
df.loc[df["Country"].isnull(), "Country"] = "Unknown"
```

[23]:
```
print(df["Country"].isnull().sum())
```

```
0
```

1.0.4 all missing country values are handled, your dataset is clean and standardized for country-wise analysis.

Standardizing Category Names ## Since some category values are missing or inconsistent, we'll ensure proper formatting: 1 Check for missing categories

[25]:
```
print(df["category"].isnull().sum())
```

```
46
```

2 2 Fill missing values with "Unknown" or "Other" if needed

[29]:
```
df.loc[df["category"].isnull(), "category"] = "Unknown"
```

[31]:
```
print(df["category"].isnull().sum())
```

```
0
```

2.0.1 Standardize capitalization & spacing

[34]:
```
df["category"] = df["category"].str.strip().str.title()  # Makes "music" →␣
  ↪"Music"
```

2.0.2 Verify changes

[36]:
```
print(df["category"].unique())  # Check consistency
```

```
['Music' 'Film & Animation' 'Entertainment' 'Education' 'Shows' 'Unknown'
 'People & Blogs' 'Gaming' 'Sports' 'Howto & Style' 'News & Politics'
 'Comedy' 'Trailers' 'Nonprofits & Activism' 'Science & Technology'
 'Movies' 'Pets & Animals' 'Autos & Vehicles' 'Travel & Events']
```

2.0.3 Standardizing Date Formats

Since created_date and trending_date are stored as floats, let's convert them into proper date formats for consistency. ## Step 1.1: Convert created_date to DateTime

[48]:
```
print(df["created_date"].head(20))  # Check first 20 values
```

```
0     1970-01-01 00:00:00.000000013
1     1970-01-01 00:00:00.000000005
2     1970-01-01 00:00:00.000000020
3     1970-01-01 00:00:00.000000001
4     1970-01-01 00:00:00.000000020
```

```
5     1970-01-01 00:00:00.000000024
6     1970-01-01 00:00:00.000000012
7     1970-01-01 00:00:00.000000029
8     1970-01-01 00:00:00.000000014
9     1970-01-01 00:00:00.000000023
10    1970-01-01 00:00:00.000000012
11    1970-01-01 00:00:00.000000011
12    1970-01-01 00:00:00.000000015
13    1970-01-01 00:00:00.000000029
14    1970-01-01 00:00:00.000000015
15    1970-01-01 00:00:00.000000004
16    1970-01-01 00:00:00.000000027
17    1970-01-01 00:00:00.000000017
18    1970-01-01 00:00:00.000000030
19    1970-01-01 00:00:00.000000015
Name: created_date, dtype: datetime64[ns]
```

2.0.4 This confirms that the original date conversion was incorrect, likely due to the float values being misinterpreted as timestamps, causing them to default to 1970-01-01.

2.0.5 Reconstruct created_date Using created_year & created_month

2.0.6 Since we have separate columns for year and month, we can correctly rebuild the date:

[52]:
```
df["created_date_fixed"] = pd.to_datetime(
    df["created_year"].astype(str) + "-" + df["created_month"].astype(str) +␣
↪"-01",
    format="%Y-%b-%d",  # Explicitly define format
    errors="coerce"
)
```

2.0.7 What This Does:

Ensures created_year is treated as a 4-digit year (YYYY). created_month is treated as a month abbreviation (Jan, Feb, etc.). Day is set to 01 by default for consistency.

2.0.8 Verify Correct Date Formatting

[54]:
```
print(df[["created_date_fixed"]].head())
print(df.info())  # Ensure created_date_fixed is datetime64[ns]
```

```
  created_date_fixed
0                NaT
1                NaT
2                NaT
3                NaT
4                NaT
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 995 entries, 0 to 994
Data columns (total 29 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   rank                                     995 non-null    int64
 1   Youtuber                                 995 non-null    object
 2   subscribers                              995 non-null    int64
 3   video views                              995 non-null    float64
 4   category                                 995 non-null    object
 5   Title                                    995 non-null    object
 6   uploads                                  995 non-null    int64
 7   Country                                  995 non-null    object
 8   Abbreviation                             873 non-null    object
 9   channel_type                             965 non-null    object
 10  video_views_rank                         994 non-null    float64
 11  country_rank                             879 non-null    float64
 12  channel_type_rank                        962 non-null    float64
 13  video_views_for_the_last_30_days         939 non-null    float64
 14  lowest_monthly_earnings                  995 non-null    float64
 15  highest_monthly_earnings                 995 non-null    float64
 16  lowest_yearly_earnings                   995 non-null    float64
 17  highest_yearly_earnings                  995 non-null    float64
 18  subscribers_for_last_30_days             658 non-null    float64
 19  created_year                             990 non-null    float64
 20  created_month                            990 non-null    object
 21  created_date                             990 non-null    datetime64[ns]
 22  Gross tertiary education enrollment (%)  872 non-null    float64
 23  Population                               872 non-null    float64
 24  Unemployment rate                        872 non-null    float64
 25  Urban_population                         872 non-null    float64
 26  Latitude                                 872 non-null    float64
 27  Longitude                                872 non-null    float64
 28  created_date_fixed                       0 non-null      datetime64[ns]
dtypes: datetime64[ns](2), float64(17), int64(3), object(7)
memory usage: 225.6+ KB
None
```

[42]:
```
print(df.columns)
```

```
Index(['rank', 'Youtuber', 'subscribers', 'video views', 'category', 'Title',
       'uploads', 'Country', 'Abbreviation', 'channel_type',
       'video_views_rank', 'country_rank', 'channel_type_rank',
       'video_views_for_the_last_30_days', 'lowest_monthly_earnings',
       'highest_monthly_earnings', 'lowest_yearly_earnings',
       'highest_yearly_earnings', 'subscribers_for_last_30_days',
       'created_year', 'created_month', 'created_date',
       'Gross tertiary education enrollment (%)', 'Population',
       'Unemployment rate', 'Urban_population', 'Latitude', 'Longitude'],
      dtype='object')
```

2.0.9 Check created_year and created_month Formats

[56]:
```
print(df[["created_year", "created_month"]].head(20))
```

```
    created_year created_month
0         2006.0           Mar
1         2006.0           Mar
2         2012.0           Feb
3         2006.0           Sep
4         2006.0           Sep
5         2013.0           Sep
6         2015.0           May
7         2010.0           Apr
8         2016.0           Jan
9         2018.0           Apr
10        2014.0           Mar
11        2007.0           May
12        2013.0           Dec
13        2016.0           Jun
14        2006.0           Aug
15        2007.0           Aug
16        2020.0           Jul
17        2012.0           Dec
18        2006.0           Jan
19        2007.0           Jan
```

2.0.10 Convert created_year fom Float to Integer

[58]:
```
df["created_year"] = df["created_year"].astype("Int64")  # Converts to integer␣
 ↪format
```

2.0.11 Convert created_month to Numeric Format

[61]:
```
month_mapping = {
    "Jan": "01", "Feb": "02", "Mar": "03", "Apr": "04", "May": "05", "Jun":␣
 ↪"06",
    "Jul": "07", "Aug": "08", "Sep": "09", "Oct": "10", "Nov": "11", "Dec": "12"
}

df["created_month"] = df["created_month"].replace(month_mapping)
```

3 Reconstruct created_date Properly

[63]:
```
df["created_date_fixed"] = pd.to_datetime(
    df["created_year"].astype(str) + "-" + df["created_month"].astype(str) +␣
 ↪"-01",
    format="%Y-%m-%d",
    errors="coerce"
)
```

3.1 Check if created_date_fixed contains valid dates

[65]:
```
print(df[["created_date_fixed"]].head())
```

```
  created_date_fixed
0         2006-03-01
1         2006-03-01
2         2012-02-01
3         2006-09-01
4         2006-09-01
```

[69]:
```
### Since created_date_fixed is now accurate, let's finalize the cleanup:
#   Drop the old incorrect created_date column
df.drop(columns=["created_date"], inplace=True)


#   Rename created_date_fixed to created_date for consistency
df.rename(columns={"created_date_fixed": "created_date"}, inplace=True)


#   Verify the final date format
print(df[["created_date"]].head())
print(df.info())  # Ensure `created_date` is now in datetime format
```

```
  created_date
0   2006-03-01
1   2006-03-01
2   2012-02-01
3   2006-09-01
4   2006-09-01
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 995 entries, 0 to 994
Data columns (total 28 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   rank                                     995 non-null    int64
 1   Youtuber                                 995 non-null    object
 2   subscribers                              995 non-null    int64
 3   video views                              995 non-null    float64
 4   category                                 995 non-null    object
 5   Title                                    995 non-null    object
 6   uploads                                  995 non-null    int64
 7   Country                                  995 non-null    object
 8   Abbreviation                             873 non-null    object
 9   channel_type                             965 non-null    object
 10  video_views_rank                         994 non-null    float64
 11  country_rank                             879 non-null    float64
 12  channel_type_rank                        962 non-null    float64
 13  video_views_for_the_last_30_days         939 non-null    float64
 14  lowest_monthly_earnings                  995 non-null    float64
 15  highest_monthly_earnings                 995 non-null    float64
 16  lowest_yearly_earnings                   995 non-null    float64
 17  highest_yearly_earnings                  995 non-null    float64
 18  subscribers_for_last_30_days             658 non-null    float64
 19  created_year                             990 non-null    Int64
 20  created_month                            990 non-null    object
 21  Gross tertiary education enrollment (%)  872 non-null    float64
 22  Population                               872 non-null    float64
 23  Unemployment rate                        872 non-null    float64
 24  Urban_population                         872 non-null    float64
 25  Latitude                                 872 non-null    float64
 26  Longitude                                872 non-null    float64
 27  created_date                             990 non-null    datetime64[ns]
dtypes: Int64(1), datetime64[ns](1), float64(16), int64(3), object(7)
memory usage: 218.8+ KB
None
```

3.1.1 Check for Duplicate Entries

[5]:
```
print(df.duplicated().sum())  # Total number of duplicate rows
```

```
0
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:
```

```

[ ]:
```

```

[ ]:
```

```

[ ]:
```

```