

```
In [ ]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Data Cleaning and Transformation

Global YouTube Statistics 2023

This dataset can be openly accessed via [Kaggle.com](https://www.kaggle.com)

The dataset is a collection of the most subscribed channels on youtube in 2023 it has comprehensive details on top creators' subscriber counts, video views, upload frequency, country of origin, earnings, and more.

In this jupyter notebook I'll have a go at cleaning and transforming the data. This is part of my portfolio on data analysis.

```
In [ ]: # Read the dataset from a csv file and load it into a pandas dataframe.
# Please notice, the accompanying metadata doesn't specify the proper encoding
# "UnicodeDecodeError: 'utf-8' codec can't decode byte 0xfd in position 6
# Hence I opted for an alternatively common encoding: latin-1. This reads

df = pd.read_csv("Global YouTube Statistics.csv", encoding="latin-1")
```

Getting an overview

```
In [ ]: # Show the first five rows of the dataframe with their columns.
# Transposed to prevent truncation of columns by screen width limitations
print("df.head().T")
display(df.head().T)

# statistical description of numerical columns
# Transposed to prevent truncation of columns by screen width limitations
print("df.describe().T")
display(df.describe().T)

# Show datatypes for the dataframe's columns
print("df.dtypes")
display(df.dtypes)

# Show the sum of missing values per column
print("df.isna().sum()")
display(df.isna().sum())
```

```
df.head().T
```

	0	1	2	
rank	1	2	3	
Youtuber	T-Series	YouTube Movies	MrBeast	C
subscribers	245000000	170000000	166000000	
video views	228000000000.0	0.0	28368841870.0	1640
category	Music	Film & Animation	Entertainment	
Title	T-Series	youtubemovies	MrBeast	C
uploads	20082	1	741	
Country	India	United States	United States	U
Abbreviation	IN	US	US	
channel_type	Music	Games	Entertainment	
video_views_rank	1.0	4055159.0	48.0	
country_rank	1.0	7670.0	1.0	
channel_type_rank	1.0	7423.0	1.0	
video_views_for_the_last_30_days	2258000000.0	12.0	1348000000.0	19
lowest_monthly_earnings	564600.0	0.0	337000.0	
highest_monthly_earnings	9000000.0	0.05	5400000.0	
lowest_yearly_earnings	6800000.0	0.04	4000000.0	
highest_yearly_earnings	108400000.0	0.58	64700000.0	
subscribers_for_last_30_days	2000000.0	NaN	8000000.0	
created_year	2006.0	2006.0	2012.0	
created_month	Mar	Mar	Feb	
created_date	13.0	5.0	20.0	
Gross tertiary education enrollment (%)	28.1	88.2	88.2	
Population	1366417754.0	328239523.0	328239523.0	3
Unemployment rate	5.36	14.7	14.7	
Urban_population	471031528.0	270663028.0	270663028.0	2
Latitude	20.593684	37.09024	37.09024	

		0	1	2	
	Longitude	78.96288	-95.712891	-95.712891	
df.describe().T					
		count	mean	std	min
	rank	995.0	4.980000e+02	2.873761e+02	1.000000e+00
	subscribers	995.0	2.298241e+07	1.752611e+07	1.230000e+07
	video views	995.0	1.103954e+10	1.411084e+10	0.000000e+00
	uploads	995.0	9.187126e+03	3.415135e+04	0.000000e+00
	video_views_rank	994.0	5.542489e+05	1.362782e+06	1.000000e+00
	country_rank	879.0	3.860535e+02	1.232245e+03	1.000000e+00
	channel_type_rank	962.0	7.457193e+02	1.944387e+03	1.000000e+00
	video_views_for_the_last_30_days	939.0	1.756103e+08	4.163782e+08	1.000000e+00
	lowest_monthly_earnings	995.0	3.688615e+04	7.185872e+04	0.000000e+00
	highest_monthly_earnings	995.0	5.898078e+05	1.148622e+06	0.000000e+00
	lowest_yearly_earnings	995.0	4.422574e+05	8.612161e+05	0.000000e+00
	highest_yearly_earnings	995.0	7.081814e+06	1.379704e+07	0.000000e+00
	subscribers_for_last_30_days	658.0	3.490791e+05	6.143554e+05	1.000000e+00
	created_year	990.0	2.012630e+03	4.512503e+00	1.970000e+03
	created_date	990.0	1.574646e+01	8.777520e+00	1.000000e+00
	Gross tertiary education enrollment (%)	872.0	6.362775e+01	2.610689e+01	7.600000e+00
	Population	872.0	4.303873e+08	4.727947e+08	2.025060e+05
	Unemployment rate	872.0	9.279278e+00	4.888354e+00	7.500000e-01
	Urban_population	872.0	2.242150e+08	1.546874e+08	3.558800e+04
	Latitude	872.0	2.663278e+01	2.056053e+01	-3.841610e+01
	Longitude	872.0	-1.412815e+01	8.476081e+01	-1.721046e+02

df.dtypes

rank	int64
Youtuber	object
subscribers	int64
video views	float64
category	object
Title	object
uploads	int64
Country	object
Abbreviation	object
channel_type	object
video_views_rank	float64
country_rank	float64
channel_type_rank	float64
video_views_for_the_last_30_days	float64
lowest_monthly_earnings	float64
highest_monthly_earnings	float64
lowest_yearly_earnings	float64
highest_yearly_earnings	float64
subscribers_for_last_30_days	float64
created_year	float64
created_month	object
created_date	float64
Gross tertiary education enrollment (%)	float64
Population	float64
Unemployment rate	float64
Urban_population	float64
Latitude	float64
Longitude	float64
dtype: object	
df.isna().sum()	
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	

Subscribers count

Since this database holds the most subscribed youtube channels the minimal value for subscribers is in the millions.

For ease of readability I transform the Series to multiples of 1 million.

```
In [ ]: # statistical description of subscribers count
print(df['subscribers'].describe())
display(df["subscribers"].describe())

# Transform subscribers Series
df["subscribers_mil"] = (df["subscribers"]/1_000_000).round().astype("int64")

# display newly calculated Series
print("Newly created Series of subscriber counts in multiples of 1 million")
display(df["subscribers_mil"].describe())

# Show datatypes
print("Datatype of df['subscribers']")
display(df["subscribers"].dtype)
print("Datatype of df['subscribers_mil']")
display(df["subscribers_mil"].dtype)
```

```
df['subscribers'].describe()
count      9.950000e+02
mean       2.298241e+07
std        1.752611e+07
min        1.230000e+07
25%        1.450000e+07
50%        1.770000e+07
75%        2.460000e+07
max        2.450000e+08
Name: subscribers, dtype: float64
Newly created Series of subscriber counts in multiples of 1 million
count      995.000000
mean       22.962814
std        17.544815
min        12.000000
25%        14.000000
50%        18.000000
75%        25.000000
max        245.000000
Name: subscribers_mil, dtype: float64
Datatype of df['subscribers']
dtype('int64')
Datatype of df['subscribers_mil']
dtype('int64')
```

Video Views

For easier downstream calculations and since video views are supposed to be whole numbers, I transform Video views count to int64.

For ease of readability I also transform the Series to multiples of 1 million.

```
In [ ]: # original data description
print("Description of original data")
display(df["video views"].describe())

# Does the column contain NaN values?
print("Does the Series contain NaN entries? No")
display(df["video views"].isna().sum())
# no

# transform video views Series
df["video views mil"] = (df["video views"]/1_000_000).round().astype('int64')
df["video views"] = df["video views"].astype('int64')

# display newly calculated Series
print("Description of newly created Series 'video views mil'")
display(df["video views mil"].describe())

# Notice the discrepancy of dtypes reported with .df.describe() and df.dtype
# The .describe() method provides summary statistics for a Series, and it
print("Datatype of 'video views mil'")
display(df["video views mil"].dtype)
print("Datatype of 'video views '")
display(df["video views"].dtype)
```

Description of original data

```
count      9.950000e+02
mean       1.103954e+10
std        1.411084e+10
min        0.000000e+00
25%        4.288145e+09
50%        7.760820e+09
75%        1.355470e+10
max        2.280000e+11
```

Name: video views, dtype: float64

Does the Series contain NaN entries? No

0

Description of newly created Series 'video views mil'

```
count      995.000000
mean       11039.540704
std        14110.846997
min         0.000000
25%        4288.500000
50%        7761.000000
75%        13555.000000
max        228000.000000
```

Name: video views mil, dtype: float64

Datatype of 'video views mil'

dtype('int64')

Datatype of 'video views '

dtype('int64')

Identify faulty entries

The minimal video views is 0, which seems not feasible for the most subscribed channels.

By identifying the channels with the lowest video views, we can get more information.

The 10 channels with the lowest video views seem to be either youtube specific channels, that have different system of counting views or errors in the database. Youtube Movies for example is a service to buy and rent movies, which are not openly accessible like normal youtube videos.

"Popular on YouTube" and "Minecraft - Topic" are not independent channels, but rather platforms that point to other channels within their topics scope.

Other channels seem to be errors in the database, as the channels "Happy Lives" and "ýýýýý" with rather low video views (below 1 million) can not be found on youtube as of September 2023.

```
In [ ]: # Identify channels with lowest video views count
print("channels with the lowest amount of video views")
display(df.nsmallest(15, 'video views mil'))

# How many channels have NaN video views?
print("channels listed with NaN video views")
display(df[df['video views'].isna()])
# 0
```

channels with the lowest amount of video views

	rank	Youtuber	subscribers	video views	category	Title	uploads	
	1	2	YouTube Movies	170000000	0	Film & Animation	youtubemovies	1
	5	6	Music	119000000	0	NaN	Music	0
	12	13	Gaming	93600000	0	NaN	Gaming	0
	18	19	Sports	75000000	0	NaN	sports	3
	102	103	News	36300000	0	NaN	News	0
	173	174	Popular on YouTube	29300000	0	NaN	Popular on Youtube	3
	286	287	Happy Lives	23200000	2634	Science & Technology	Happy Lives	1
	360	361	Minecraft - Topic	20900000	0	NaN	Minecraft - Topic	0
	592	593	Live	16100000	0	NaN	Live	0
	700	701	ýýýýýý	14900000	439098	People & Blogs	ýýýýýý	1
	902	903	Calon Sarjana	13000000	10664585	Entertainment	Calon Sarjana	29
	852	853	Vibhu 96	13400000	20563378	NaN	Vibhu 96	256
	912	913	Matheus Yurley	12900000	140022442	Entertainment	Matheus Yurley	69
	950	951	Wolfoo Family	12500000	161254021	NaN	Wolfoo Family	61
	424	425	Manoj parihar	19300000	264228052	NaN	Manoj parihar	335

15 rows × 30 columns

◀		▶
channels listed with NaN video views		

rank	Youtuber	subscribers	video views	category	Title	uploads	Country	Abbreviatic

0 rows × 30 columns

◀		▶
---	--	---

Remove faulty entries (video views)

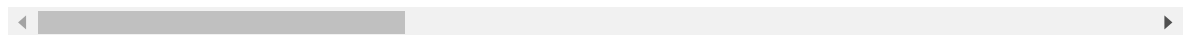
As the two channels mentioned above are not accessible on Youtube I'll remove them from the dataframe.

```
In [ ]: # Drop faulty channels by their index
df.drop([286, 700], inplace=True)

# Show that the specific indices have been dropped from the dataframe
display(df.iloc[285: 287])
display(df.iloc[698: 700])
```

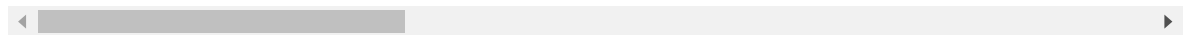
	rank	Youtuber	subscribers	video views	category	Title	uploads	Country
285	286	Sesame Street	23300000	22471357411	Entertainment	Sesame Street	3657	U S
287	288	Lady Gaga	23200000	15751661213	People & Blogs	Lady Gaga	172	U S

2 rows × 30 columns



	rank	Youtuber	subscribers	video views	category	Title	uploads	Country
699	700	ZutiGang	14900000	15913320995	People & Blogs	ZutiGang	1363	N
701	702	TazerCraft	14900000	5956193599	Gaming	TazerCraft	4175	Br

2 rows × 30 columns



Category

```
In [ ]: # How many channels fall under each category?
print("How many channels fall under each category?")
display(df["category"].value_counts())

# How many channels don't list a category?
print("How many channels don't list a category?")
display(df["category"].isna().sum())
```

How many channels fall under each category?

```
category
Entertainment      241
Music              202
People & Blogs     131
Gaming             94
Comedy             69
Film & Animation   46
Education          45
Howto & Style      40
News & Politics    26
Science & Technology 16
Shows             13
Sports            11
Pets & Animals     4
Trailers           2
Nonprofits & Activism 2
Movies            2
Autos & Vehicles   2
Travel & Events    1
Name: count, dtype: int64
How many channels don't list a category?
46
```

If you the reader, want to explore the dataset, you can change "Sports" below to any of the categories listed above and explore.

If instead you are interested in the channels without a proper category, remove the # symbol preceeding the last line.

```
In [ ]: # Display channels by specific category
df.loc[df["category"] == "Sports"]

# Display channels without a category
#df.loc[df["category"].isna()]
```

Out[]:

	rank	Youtuber	subscribers	video views	category	Title	uploa	
	11	12	WWE	96000000	77428473662	Sports	WWE	701
	28	29	Dude Perfect	59500000	16241549158	Sports	Dude Perfect	3
	368	369	NBA	20700000	12624879732	Sports	NBA	479
	423	424	FIFA	19400000	5529131886	Sports	FIFA	107
	478	479	How Ridiculous	18000000	9601137077	Sports	How Ridiculous	6
	567	568	UFC - Ultimate Fighting Championship	16400000	7135820721	Sports	UFC - Ultimate Fighting Championship	146
	646	647	FC Barcelona	15300000	2656528205	Sports	FC Barcelona	109
	790	791	F2Freestylers - Ultimate Soccer Skills Channel	14100000	3280481927	Sports	F2Freestylers - Ultimate Soccer Skills Channel½	7
	833	834	DALLMYD	13600000	1948925559	Sports	DALLMYD	4
	913	914	gymvirtual	12900000	2509752944	Sports	gymvirtual	15
	990	991	Natan por Aİz	12300000	9029609749	Sports	Natan por Aİz	12

11 rows × 30 columns



Uploads

```
In [ ]: # Display the channels with the most uploads
print("Display the channels with the most uploads")
display(df.nlargest(10, 'uploads'))

# It seems that news & politics channels upload the most videos

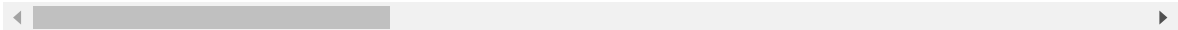
# Display the channels with the fewest uploads
print("Display the channels with the fewest uploads")
display(df.nsmallest(10, 'uploads'))

# There are a few channels listed with 0 uploads, but non-zero video views
```

Display the channels with the most uploads

	rank	Youtuber	subscribers	video	views	category	Title	uploads
95	96	ABP NEWS	37000000	13102611515		People & Blogs	ABP NEWS	301308
857	858	GMA Integrated News	13400000	9569814790		News & Politics	GMA Integrated News	296272
747	748	TV9 Bharatvarsh	14500000	10303519926		People & Blogs	TV9 Bharatvarsh	293516
33	34	Aaj Tak	57600000	25307753534		News & Politics	Aaj Tak	283775
107	108	IndiaTV	35500000	16105023749		News & Politics	IndiaTV	273255
689	690	KOMPASTV	15000000	11827310821		News & Politics	KOMPASTV	269050
586	587	Thairath Online	16200000	14563841315		News & Politics	Thairath Online	244899
502	503	News 24	17700000	8396875537		News & Politics	News 24	211620
673	674	ABS-CBN News	15100000	10489367372		News & Politics	ABS-CBN News	209520
84	85	TEDx Talks	38600000	7339333120		Nonprofits & Activism	TEDx Talks	200933

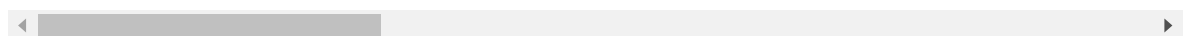
10 rows × 30 columns



Display the channels with the fewest uploads

	rank	Youtuber	subscribers	video	views	category	Title
	5	6	Music	119000000	0	NaN	Music
	12	13	Gaming	93600000	0	NaN	Gaming
	57	58	BRIGHT SIDE	44500000	10708531817	Howto & Style	brightside
	73	74	Luisito Comunica	40600000	8670473639	Comedy	Luis Arturo Villar Sudek
	102	103	News	36300000	0	NaN	News
	113	114	T-Series Apna Punjab	34600000	21306315429	Music	T- Series Apna Punjab
	149	150	Luis Fonsi	31400000	15176762479	Entertainment	luisfonsi
	166	167	Frost Diamond	30100000	7277493940	Gaming	frostdiamond
	180	181	Aditya Music India	28500000	25857994495	Music	Aditya Music
	190	191	Sandeep Maheshwari	27800000	2303069221	People & Blogs	Sandeepmaheshwari

10 rows × 30 columns



There seems to be an error in the database with the uploads Series, as some channels are listed with 0 uploads, but non-zero video views. This can't be easily resolved by calculation of other Series.

As such, I'll change the uploads column for channels with non-zero video views to -1 to indicate the discrepancy, while keeping the int64 datatype of the Series.

```
In [ ]: # Change "uploads" from 0 to -1
df.loc[df["uploads"] == 0, "uploads"] = -1

# Descriptive statistics of "uploads" Series
print("descriptive statistics of 'uploads', notice that I changed values")
display(df["uploads"].describe())

# Display the channels with exactly -1 uploads
print("Show channels with newly set 'uploads' of -1")
display(df.loc[df["uploads"] == -1])
```

descriptive statistics of 'uploads', notice that I changed values from 0 to -1, as indicated by the min value

```
count      993.000000
mean       9205.584089
std        34183.279928
min         -1.000000
25%        196.000000
50%        733.000000
75%       2717.000000
max       301308.000000
Name: uploads, dtype: float64
Show channels with newly set 'uploads' of -1
```

	rank	Youtuber	subscribers	video views	category	
	5	6	Music	119000000	0	NaN
	12	13	Gaming	93600000	0	NaN
	57	58	BRIGHT SIDE	44500000	10708531817	Howto & Style
	73	74	Luisito Comunica	40600000	8670473639	Comedy
	102	103	News	36300000	0	NaN
	113	114	T-Series Apna Punjab	34600000	21306315429	Music
	149	150	Luis Fonsi	31400000	15176762479	Entertainment
	166	167	Frost Diamond	30100000	7277493940	Gaming
	180	181	Aditya Music India	28500000	25857994495	Music
	190	191	Sandeep Maheshwari	27800000	2303069221	People & Blogs
	217	218	1MILLION Dance Studio	26100000	7886440199	Entertainment
	226	227	Fede Vigevani	25600000	7962725960	Howto & Style
	236	237	Chris Brown	25200000	15520569496	Music
	273	274	FaZe Rug	23700000	7451792132	Gaming
	299	300	Alan Becker	22900000	5380073627	Film & Animation
	340	341	YOLO	21400000	1573058816	Comedy
	360	361	Minecraft - Topic	20900000	0	NaN
	377	378	Linkin Park	20400000	13397000296	Music
	386	387	Family GamesTV	20200000	7066711323	Entertainment
	394	395	Robin Hood Gamer	20100000	10366850490	Entertainment
	440	441	Masha e o Urso	18900000	8301731337	Shows
	463	464	Dhar Mann	18400000	11280732382	People & Blogs
	468	469	Good Mythical Morning	18300000	8798044678	Entertainment

	rank	Youtuber	subscribers	video views	category	
476	477	Ajay Sharma	18100000	12249828886	Entertainment	Ajay Sha
508	509	The Game Theorists	17600000	3752347262	Gaming	TheGameThec
544	545	Doggy Doggy Cartoons	16800000	6518418501	Entertainment	Doggy Do Carto
554	555	Werever2morro	16600000	2798273962	Entertainment	werever2m
592	593	Live	16100000	0	NaN	
600	601	La Rosa de Guadalupe	16100000	9642146451	Entertainment	larosadeguada
604	605	Enes Batur	16000000	9786595271	Gaming	enest
606	607	yyyyyyyyyy	15900000	1845329502	People & Blogs	Kung Fu F
629	630	Super Senya	15500000	5070970714	Entertainment	Super S
634	635	Mr DegrEE	15500000	12880388253	Science & Technology	MrDe
680	681	TheRichest	15000000	2730879024	Education	Theric
710	711	Major Lazer Official	14800000	9383431376	Music	MajorLazerOf
735	736	LEGENDA FUNK	14500000	2440718089	Music	LegendaF
762	763	Harry Styles	14400000	5689224452	People & Blogs	harrys
777	778	Crazy Frog	14200000	7946322061	Music	Crazy
853	854	Blockbuster Movies	13400000	2650061211	Entertainment	Blockbuster Mo
877	878	Ei Nerd	13200000	3568392223	Entertainment	Ei
951	952	Ja Mill	12500000	1302818088	People & Blogs	j
983	984	MoniLina	12400000	16086808918	Comedy	MoniLinaFa
985	986	TKOR	12400000	3392022527	Education	-

43 rows × 30 columns

Country

```
In [ ]: # Display the top 5 countries with the most registered Youtube channels
print("Display the top 5 countries with the most registered Youtube channels")
display(df["Country"].value_counts().head())

# How many channels don't have an entry in the "Country" column?
print("How many channels don't have an entry in the 'Country' column?")
display(df["Country"].isna().sum())
```

Display the top 5 countries with the most registered Youtube channels

Country

United States 312

India 168

Brazil 62

United Kingdom 43

Mexico 33

Name: count, dtype: int64

How many channels don't have an entry in the 'Country' column?

122

Video views rank

```
In [ ]: # Are there NaN values in video views rank?
display(df["video_views_rank"].isna().sum())

# Identify the channel without a ranking
display(df.loc[df["video_views_rank"].isna()], ["Title", "video_views_rank"])

# Drop the LegendaFUNK channel
df.drop(735, inplace = True)

# Show that the specific index 735 has been dropped from the dataframe
df.iloc[732:734]
```

1

	Title	video_views_rank
735	LegendaFUNK	NaN

Out[]:

	rank	Youtuber	subscribers	video views	category	Title	uploads	Co
734	735	Noman Official	14600000	5525773746	Comedy	Noman Official	560	
736	737	Like Nastya Stories	14500000	6944967581	Entertainment	Like Nastya Stories	479	U s

2 rows × 30 columns



Country rank

```
In [ ]: # Are there NaN values in country ranks?
display(df["country_rank"].isna().sum())

# Identify the channel without a ranking
display(df.loc[df["country_rank"].isna(), ["Title", "Country", "country_r

# Are there entries with a country but without a ranking?
display(df.loc[(df["Country"].notna()) &
               (df["country_rank"].isna())])

# No, the missing "country_rank" stems solely from the missing entry in t
```

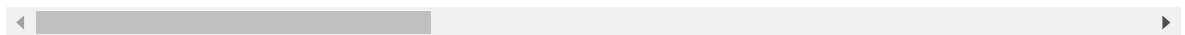
115

	Title	Country	country_rank
5	Music	NaN	NaN
12	Gaming	NaN	NaN
14	goldmines	NaN	NaN
38	LooLoo Kids - Nursery Rhymes and Children's ĩ½	NaN	NaN
48	badabun	NaN	NaN
...
958	Troom Troom PT	NaN	NaN
967	TROOM TROOM INDONESIA	NaN	NaN
972	Hero Movies 2023	NaN	NaN
985	TKoR	NaN	NaN
986	annakova	NaN	NaN

115 rows × 3 columns

rank	Youtuber	subscribers	video views	category	Title	uploads	Country	Abbreviat
------	----------	-------------	----------------	----------	-------	---------	---------	-----------

0 rows × 30 columns



channel_type & channel_type_rank

```
In [ ]: # Display the different types of channels listed
display(df["channel_type"].value_counts())

# Are there NaN values in channel_type?
display(df["channel_type"].isna().sum())

# Statistical description of the channel_type_rank column
display(df["channel_type_rank"].describe())

# Are there NaN entries in the channel_type_rank column?
display(df["channel_type_rank"].isna().sum())

# Are there entries with a channel type but without a channel_type_rank?
display(df.loc[(df["channel_type"].notna()) &
               (df["channel_type_rank"].isna()),
               ["Title", "channel_type", "channel_type_rank"]])

# Yes, there are 2 channels from the Music and Comedy type, that don't ha
# Let's drop these 2 channels
```

```
df.drop([5, 73], inplace=True)

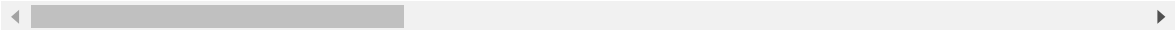
# Show that the specific indices have been dropped from the dataframe
display(df.iloc[4:6])
display(df.iloc[71:73])
```

channel_type
Entertainment 303
Music 215
People 101
Games 98
Comedy 51
Education 49
Film 42
Howto 36
News 29
Tech 17
Sports 13
Autos 3
Animals 3
Nonprofit 2
Name: count, dtype: int64
30
count 960.000000
mean 747.170833
std 1946.151925
min 1.000000
25% 27.000000
50% 65.500000
75% 140.000000
max 7741.000000
Name: channel_type_rank, dtype: float64
32

	Title	channel_type	channel_type_rank
5		Music	NaN
73	Luis Arturo Villar Sudek	Comedy	NaN

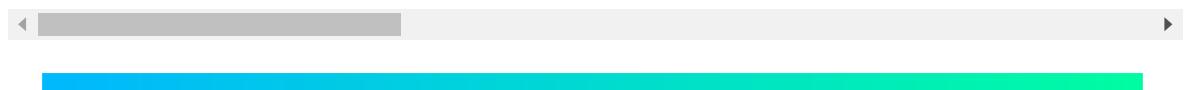
	rank	Youtuber	subscribers	video views	category	Title	uploads	Country	A
4	5	SET India	159000000	148000000000	Shows	SET India	116536	India	
6	7	ýýý Kids Diana Show	112000000	93247040539	People & Blogs	ýýý Kids Diana Show	1111	United States	

2 rows × 30 columns



	rank	Youtuber	subscribers	video views	category	Title	uploads
72	73	Little Baby Bum - Nursery Rhymes & Kids Songs	40900000	39450824833	Education	Little Baby Bum - Nursery Rhymes & Kids Songs	2423
74	75	elrubiusOMG	40400000	7410536668	Gaming	elrubiusOMG	703

2 rows × 30 columns



video_views_for_the_last_30_days

```
In [ ]: # Statistical description of the channel_type_rank column
display(df["video_views_for_the_last_30_days"].describe())

# Are there NaN entries in the channel_type_rank column?
display(df["video_views_for_the_last_30_days"].isna().sum())

# Does the missing entry in video_views_for_the_last_30_days correspond to
display(df.loc[df["video_views_for_the_last_30_days"].isna() & df["video_views"] != 0])
# no

# Are there channels with higher view counts for the last 30 days than the total
video_views_mismatch = df.loc[(df["video_views_for_the_last_30_days"] > df["video_views"]) &
                               (df["video_views"] != 0)]
display(video_views_mismatch.T)

# Calculate the differences in the transposed DataFrame
differences = video_views_mismatch.T.loc[:, "video_views_for_the_last_30_days"]

# Rename the columns for clarity
differences.columns = ["video_views", "video_views difference"]

# Print the calculated differences
print(differences)

# The Difference between the video views for the last 30 days and the total
# We'll drop these three entries.
df.drop([455, 902, 950], inplace=True)
```

```
count      9.370000e+02
mean       1.689531e+08
std        3.601665e+08
min        1.000000e+00
25%        2.026300e+07
50%        6.408500e+07
75%        1.685970e+08
max        6.148000e+09
Name: video_views_for_the_last_30_days, dtype: float64
53
```

rank	Youtuber	subscribers	video views	category	Title	uploads	Country	Abbreviatic
0 rows × 30 columns								
<div><div></div></div>								

	455	902	950
rank	456	903	951
Youtuber	Dan-Sa / Daniel Saboya	Calon Sarjana	Wolfoo Family
subscribers	18500000	13000000	12500000
video views	2908120896	10664585	161254021
category	Music	Entertainment	NaN
Title	Dan-Sa / Daniel Saboya	Calon Sarjana	Wolfoo Family
uploads	1329	29	61
Country	Brazil	Indonesia	United States
Abbreviation	BR	ID	US
channel_type	Music	Entertainment	Film
video_views_rank	45213.0	772571.0	81750.0
country_rank	24.0	31.0	175.0
channel_type_rank	102.0	166.0	46.0
video_views_for_the_last_30_days	6148000000.0	2292000000.0	757789000.0
lowest_monthly_earnings	0.0	0.0	0.0
highest_monthly_earnings	0.0	0.0	0.0
lowest_yearly_earnings	0.0	0.0	0.0
highest_yearly_earnings	0.0	0.0	0.0
subscribers_for_last_30_days	100000.0	300000.0	NaN
created_year	2007.0	2016.0	2019.0
created_month	Sep	Jan	Oct
created_date	22.0	20.0	21.0
Gross tertiary education enrollment (%)	51.3	36.3	88.2
Population	212559417.0	270203917.0	328239523.0
Unemployment rate	12.08	4.69	14.7
Urban_population	183241641.0	151509724.0	270663028.0
Latitude	-14.235004	-0.789275	37.09024
Longitude	-51.92528	113.921327	-95.712891
subscribers_mil	18	13	12
video views mil	2908	11	161

	video views	video views	difference
455	NaN		-3239879104.0
902	NaN		-2281335415.0
950	NaN		-596534979.0

Earnings

```
In [ ]: # Statistical description of four earnings columns
display(df[["lowest_monthly_earnings", "highest_monthly_earnings", "lowest_yearly_earnings", "highest_yearly_earnings"]].describe())

# Are there NaN entries in any of the earnings columns?
display(df[["lowest_monthly_earnings", "highest_monthly_earnings", "lowest_yearly_earnings", "highest_yearly_earnings"]].isna().sum())
# no
```

	lowest_monthly_earnings	highest_monthly_earnings	lowest_yearly_earnings	highest_yearly_earnings
count	987.000000	9.870000e+02	9.870000e+02	9.870000e+02
mean	37185.122128	5.945884e+05	4.458420e+05	4.458420e+05
std	72072.488523	1.152039e+06	8.637773e+05	8.637773e+05
min	0.000000	0.000000e+00	0.000000e+00	0.000000e+00
25%	2900.000000	4.585000e+04	3.440000e+04	3.440000e+04
50%	13500.000000	2.165000e+05	1.624000e+05	1.624000e+05
75%	38250.000000	6.122000e+05	4.591500e+05	4.591500e+05
max	850900.000000	1.360000e+07	1.020000e+07	1.020000e+07

```
lowest_monthly_earnings    0
highest_monthly_earnings    0
lowest_yearly_earnings      0
highest_yearly_earnings      0
dtype: int64
```

subscribers_for_last_30_days

```
In [ ]: # Statistical description of four earnings columns
display(df[["subscribers_for_last_30_days"]].describe())

# Are there NaN entries in any of the earnings columns?
display(df[["subscribers_for_last_30_days"]].isna().sum())
# yes, 332 entries

# Does subscribers_for_last_30_days surpass the total subscribers count?
# Notice the conversion to float64. As "subscribers_for_last_30_days" column is float64, the comparison will be float64 vs int64.
display(df.loc[df[["subscribers_for_last_30_days"]].astype(float) > df[["subscribers"]].astype(float)]).describe()
# no
```



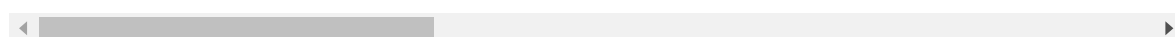
```

count      6.550000e+02
mean       3.499146e+05
std        6.156053e+05
min        1.000000e+00
25%        1.000000e+05
50%        2.000000e+05
75%        4.000000e+05
max         8.000000e+06
Name: subscribers_for_last_30_days, dtype: float64
332

```

rank	Youtuber	subscribers	video views	category	Title	uploads	Country	Abbreviat
------	----------	-------------	----------------	----------	-------	---------	---------	-----------

0 rows × 30 columns



Channel Creation

```

In [ ]: # Statistical description of creation year date column
display(df[["created_year", "created_month", "created_date"]].describe())

# Are there NaN entries in any of the creation columns?
display(df[["created_year", "created_month", "created_date"]].isna().sum())
# yes, 4 entries in each column

# Drop entries created before the launch of youtube (2005)
df.drop(df[df['created_year'] < 2005].index, inplace=True)

# Drop entries without a proper creation entry
df.dropna(subset=['created_year', "created_month", "created_date"], inplace=True)

# Is the minimal value 2005?
display(df["created_year"].describe())
# yes

# Are there NaN entries in any of the creation columns?
display(df[["created_year", "created_month", "created_date"]].isna().sum())
# no longer any NaN

```

	created_year	created_date
count	983.000000	983.000000
mean	2012.629705	15.725331
std	4.514653	8.798250
min	1970.000000	1.000000
25%	2009.000000	8.000000
50%	2013.000000	16.000000
75%	2016.000000	23.000000
max	2022.000000	31.000000

```

created_year      4
created_month     4
created_date      4
dtype: int64
count      982.000000
mean      2012.673116
std        4.306796
min       2005.000000
25%       2009.000000
50%       2013.000000
75%       2016.000000
max       2022.000000
Name: created_year, dtype: float64
created_year      0
created_month     0
created_date      0
dtype: int64

```

Gross tertiary education enrollment (%)

```

In [ ]: # Statistical description of Gross tertiary education enrollment (%)
display(df["Gross tertiary education enrollment (%)"].describe())
# The max value is 113%

# Are there NaN entries in the Gross tertiary education enrollment column?
display(df["Gross tertiary education enrollment (%)"].isna().sum())
#yes, 120 entries

# At first I thought a Gross tertiary education enrollment percentage > 100% was impossible
# "A high GER generally indicates a high degree of participation, whether it is in primary, secondary or tertiary education" (https://uis.unesco.org/en/glossary-term/gross-enrolment-ratio)

```

```
count      862.000000
mean       63.522622
std        26.107115
min         7.600000
25%        36.300000
50%        68.000000
75%        88.200000
max       113.100000
Name: Gross tertiary education enrollment (%), dtype: float64
120
```



Population

```
In [ ]: # Statistical description of Population and Urban_population columns
display(df[["Population", "Urban_population"]].describe())

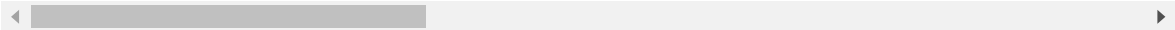
# Are there NaN entries in any of the Population columns?
display(df[["Population", "Urban_population"]].isna().sum())
# yes, 120 entries in each column

# Does the urban population exceed the total poluation?
display(df.loc[df["Population"]< df["Urban_population"]])
# no
```

	Population	Urban_population
count	8.620000e+02	8.620000e+02
mean	4.327068e+08	2.246899e+08
std	4.749152e+08	1.552588e+08
min	2.025060e+05	3.558800e+04
25%	8.313280e+07	5.590832e+07
50%	3.282395e+08	2.706630e+08
75%	3.282395e+08	2.706630e+08
max	1.397715e+09	8.429340e+08
Population	120	
Urban_population	120	
dtype:	int64	

rank	Youtuber	subscribers	video views	category	Title	uploads	Country	Abbreviatic
------	----------	-------------	-------------	----------	-------	---------	---------	-------------

0 rows × 30 columns



Unemployment rate

```
In [ ]: # Statistical description of Unemployment rate column
display(df["Unemployment rate"].describe())

# Are there NaN entries in the Unemployment rate column?
display(df["Unemployment rate"].isna().sum())
# yes, 120 entries
```

```
count      862.000000
mean         9.269838
std          4.889803
min          0.750000
25%          5.360000
50%          8.950000
75%         14.700000
max         14.720000
Name: Unemployment rate, dtype: float64
120
```