Advanced Pandas

Created by Natawut Nupairoj, Department of Computer Engineering, Chulalongkorn University

Pandas is one of the most popular tools in Python for data analytics. It contains data structures and data manipulation tools designed to make data cleaning and analysis fast and easy.

In this tutorial, we will play with a dataset from kaggle to demonstrate Pandas' basic operations. The dataset is Trending YouTube Video Statistics. For simplicity, we will work with only US dataset (USvideos.csv and US_category_id.json).

We will continue our lessons for advanced pandas. This includes data wrangling and groupby operations.

```
import pandas as pd
import numpy as np
```

Youtube Trending Data Exploration

Downloading data files from shared drive (optional for Colab)

To simplify data retrieval process on Colab, we heck if we are in the Colab environment and download data files from a shared drive and save them in folder "data".

For those using jupyter notebook on the local computer, you can read data directly assuming you save data in the folder "data".

```
import sys
IN_COLAB = 'google.colab' in sys.modules
if IN_COLAB:
    !wget https://github.com/kaopanboonyuen/2110446_DataScience_2021s2/raw/mair
    !tar -xzvf data.tgz
```

```
--2022-01-10 12:34:45--  https://github.com/kaopanboonyuen/2110446_DataScie
Resolving github.com (github.com)... 192.30.255.113
Connecting to github.com (github.com)|192.30.255.113|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/kaopanboonyuen/2110446_DataScie
--2022-01-10 12:34:45--  https://raw.githubusercontent.com/kaopanboonyuen/2
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199
HTTP request sent, awaiting response... 200 OK
Length: 45477462 (43M) [application/octet-stream]
Saving to: ‘data.tgz’

data.tgz            100%[===================>]  43.37M   270MB/s    in 0.2s

2022-01-10 12:34:45 (270 MB/s) - ‘data.tgz’ saved [45477462/45477462]

data/
data/._GB_category_id.json
data/GB_category_id.json
data/._US_category_id.json
data/US_category_id.json
data/._USvideos.csv
data/USvideos.csv
data/._GBvideos.csv
data/GBvideos.csv
```

Read input from a data file into dataframe

```
vdo_df = pd.read_csv('data/USvideos.csv')
```

Remove Duplicates

Dataframe may contain some duplicate rows.

```
vdo_df.drop_duplicates(inplace=True)
```

```
vdo_df.shape
```

```
(40901, 16)
```

Additional Data Preparation

```
vdo_df['trending_dt'] = pd.to_datetime(vdo_df.trending_date, format='%y.%d.%m',
```

Advanced Pandas Operations

When we deal with complex data and analysis, we usually have to to perform data wrangling. In addition, groupby operations are usually required.

Data Wrangling

Data contained in pandas objects can be combined together in a number of ways:

- *pandas.merge* connects rows in DataFrames based on one or more keys. This is similar to SQL *join* operations
- *pandas.concat* concatenates or 'stacks' together objects along an axis.

How each category trending in term of number of videos?

To answer this question, we will need to get category information from *US_category_id.json* file.

```
vdo_df
```

	video_id	trending_date	title	channel_title	category_id	
0	2kyS6SvSYSE	17.14.11	WE WANT TO TALK ABOUT OUR MARRIAGE	CaseyNeistat	22	1
1	1ZAPwfrtAFY	17.14.11	The Trump Presidency: Last Week	LastWeekTonight	24	1

			Tonight with J...			1
2	5qpjK5DgCt4	17.14.11	Racist Superman I Rudy Mancuso, King Bach & Le...	Rudy Mancuso	23	1
3	puqaWrEC7tY	17.14.11	Nickelback Lyrics: Real or Fake?	Good Mythical Morning	24	1
4	d380meD0W0M	17.14.11	I Dare You: GOING BALD!?	nigahiga	24	1
...	...	...	...	...	...	
40944	BZt0qjTWNhw	18.14.06	The Cat Who Caught the Laser	AaronsAnimals	15	1
40945	1h7KV2sjUWY	18.14.06	True Facts : Ant Mutualism	zefrank1	22	1
40946	D6Oy4LfoqsU	18.14.06	I GAVE SAFIYA NYGAARD A PERFECT HAIR MAKEOVER ...	Brad Mondo	24	1
40947	oV0zkMe1K8s	18.14.06	How Black Panther Should Have Ended	How It Should Have Ended	1	1
40948	ooyjaVdt-jA	18.14.06	Official Call of Duty®: Black Ops 4 — Multipla...	Call of Duty	20	1

40901 rows × 17 columns

```
cat_df = pd.read_json('data/US_category_id.json')
```

```
cat_df
```

	kind	etag	
0	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
1	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
2	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
3	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
4	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
5	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
6	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
7	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
8	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
9	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
10	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
11	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
12	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
		"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-	

13	youtube#videoCategoryListResponse	Fi-emsQJv...	'youtube#vi
14	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJv...	'youtube#vi
15	youtube#videoCategoryListResponse	"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-	'youtube#vi

US_category_id.json file seems to be more complicated than expected. We will have to write a customized reader to get specific data from this json file.

```
import json
```

```
with open('data/US_category_id.json') as fd:
    cat = json.load(fd)
```

```
cat
```


```
{'etag': '"m2yskBQFythfE4irbTIeOgYYfBU/S730Ilt-Fi-emsQJvJAAShlR6hM"',
 'items': [{'etag':
'"m2yskBQFythfE4irbTIeOgYYfBU/Xy1mB4_yLrHy_BmKmPBggty2mZQ"',
   'id': '1',
   'kind': 'youtube#videoCategory',
   'snippet': {'assignable': True,
    'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Film & Animation'}},
  {'etag': '"m2yskBQFythfE4irbTIeOgYYfBU/UZ1oLIIz2dxIhO45ZTFR3a3NyTA"',
   'id': '2',
   'kind': 'youtube#videoCategory',
   'snippet': {'assignable': True,
    'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Autos & Vehicles'}},
  {'etag': '"m2yskBQFythfE4irbTIeOgYYfBU/nqRIq97-xe5XRZTxbknKFVe5Lmg"',
   'id': '10',
   'kind': 'youtube#videoCategory',
   'snippet': {'assignable': True,
    'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Music'}},
  {'etag': '"m2yskBQFythfE4irbTIeOgYYfBU/HwXKamM1Q20q9BN-oBJavSGkfDI"',
   'id': '15',
   'kind': 'youtube#videoCategory',
   'snippet': {'assignable': True,
    'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
    'title': 'Pets & Animals'}},
  {'etag': '"m2yskBQFythfE4irbTIeOgYYfBU/9GQMSRjrZdHeb1OEM1XVQ9zbGec"',
   'id': '17',
   'kind': 'youtube#videoCategory',
   'snippet': {'assignable': True,
```

```
          'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
          'title': 'Sports'}},
        {'etag': '"m2yskBQFythfE4irbTIeOgYYfBU/FJwVpGCVZ1yiJrqZbpqe68Sy_OE"',
         'id': '18',
         'kind': 'youtube#videoCategory',
         'snippet': {'assignable': False,
          'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
          'title': 'Short Movies'}},
        {'etag': '"m2yskBQFythfE4irbTIeOgYYfBU/M-3iD9dwK7YJCafRf_DkLN8CouA"',
         'id': '19',
         'kind': 'youtube#videoCategory',
         'snippet': {'assignable': True,
          'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
          'title': 'Travel & Events'}},
        {'etag': '"m2yskBQFythfE4irbTIeOgYYfBU/WmA0qYEfjWsAoyJFSw2zinhn2wM"',
         'id': '20',
         'kind': 'youtube#videoCategory',
         'snippet': {'assignable': True,
          'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
          'title': 'Gaming'}},
        {'etag': '"m2yskBQFythfE4irbTIeOgYYfBU/EapFaGYG7K0StIXVf8aba249tdM"',
         'id': '21',
         'kind': 'youtube#videoCategory',
         'snippet': {'assignable': False,
          'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
          'title': 'Videoblogging'}},
        {'etag': '"m2yskBQFythfE4irbTIeOgYYfBU/xId8RX7vRN8rqkbYZbNIytUQDRo"',
         'id': '22',
         'kind': 'youtube#videoCategory',
```

Extract only the id and snippet->title to be used for mapping

```
cat_list = []
for d in cat['items']:
    cat_list.append((int(d['id']), d['snippet']['title']))
```

```
cat_list
```

```
[(1, 'Film & Animation'),
 (2, 'Autos & Vehicles'),
 (10, 'Music'),
 (15, 'Pets & Animals'),
 (17, 'Sports'),
 (18, 'Short Movies'),
 (19, 'Travel & Events'),
 (20, 'Gaming'),
 (21, 'Videoblogging'),
 (22, 'People & Blogs'),
 (23, 'Comedy'),
 (24, 'Entertainment'),
 (25, 'News & Politics'),
 (26, 'Howto & Style'),
 (27, 'Education'),
 (28, 'Science & Technology'),
 (29, 'Nonprofits & Activism'),
 (30, 'Movies'),
 (31, 'Anime/Animation'),
 (32, 'Action/Adventure'),
 (33, 'Classics'),
 (34, 'Comedy'),
 (35, 'Documentary'),
 (36, 'Drama'),
 (37, 'Family'),
 (38, 'Foreign'),
 (39, 'Horror'),
 (40, 'Sci-Fi/Fantasy'),
 (41, 'Thriller'),
 (42, 'Shorts'),
 (43, 'Shows'),
 (44, 'Trailers')]
```

We can create a new dataframe from a list (or dict, etc.)

```
cat_df = pd.DataFrame(cat_list, columns=['id', 'category'])
```

```
cat_df
```

	id	category
0	1	Film & Animation
1	2	Autos & Vehicles
2	10	Music
3	15	Pets & Animals

4	17	Sports
5	18	Short Movies
6	19	Travel & Events
7	20	Gaming
8	21	Videoblogging
9	22	People & Blogs
10	23	Comedy
11	24	Entertainment
12	25	News & Politics
13	26	Howto & Style
14	27	Education
15	28	Science & Technology
16	29	Nonprofits & Activism
17	30	Movies
18	31	Anime/Animation
19	32	Action/Adventure
20	33	Classics
21	34	Comedy
22	35	Documentary
23	36	Drama
24	37	Family
25	38	Foreign
26	39	Horror
27	40	Sci-Fi/Fantasy
28	41	Thriller
29	42	Shorts
30	43	Shows
31	44	Trailers

```
vdo_df_withcat = vdo_df.merge(cat_df, left_on='category_id', right_on='id')
```

```
vdo_df_withcat.columns
```

```
Index(['video_id', 'trending_date', 'title', 'channel_title',
'category_id',
       'publish_time', 'tags', 'views', 'likes', 'dislikes',
'comment_count',
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description', 'trending_dt', 'id',
       'category'],
      dtype='object')
```

```
vdo_df_withcat[['title', 'category_id', 'category']]
```

	title	category_id	category
0	WE WANT TO TALK ABOUT OUR MARRIAGE	22	People & Blogs
1	Me-O Cats Commercial	22	People & Blogs
2	AFFAIRS, EX BOYFRIENDS, $18MILLION NET WORTH -...	22	People & Blogs
3	BLIND(folded) CAKE DECORATING CONTEST (with Mo...	22	People & Blogs
4	Wearing Online Dollar Store Makeup For A Week	22	People & Blogs
...	...	...	...
40896	Game of Zones - S5:E5: The Isle of Van Gundy	43	Shows
40897	Game of Zones - S5:E5: The Isle of Van Gundy	43	Shows
40898	Game of Zones - S5:E5: The Isle of Van Gundy	43	Shows
40899	Game of Zones - S5:E5: The Isle of Van Gundy	43	Shows

```
vdo_df_withcat.category.value_counts()
```

```
Entertainment            9944
Music                    6467
Howto & Style            4142
Comedy                   3453
People & Blogs           3208
News & Politics          2485
Science & Technology     2397
Film & Animation         2343
Sports                   2172
Education                1655
Pets & Animals            920
Gaming                    816
Travel & Events           401
Autos & Vehicles          384
Shows                      57
Nonprofits & Activism      57
Name: category, dtype: int64
```

Merge function arguments

- **left**
  DataFrame to be merged on the left side.
- **right**
  DataFrame to be merged on the right side.
- **how**
  One of 'inner', 'outer', 'left', or 'right'; defaults to 'inner'.
- **on**
  Column names to join on. Must be found in both DataFrame objects. If not specified and no other join keys given, will use the intersection of the column names in left and right as the join keys.
- **left_on**
  Columns in left DataFrame to use as join keys.
- **right_on**
  Analogous to left_on for left DataFrame.
- **left_index**
  Use row index in left as its join key (or keys, if a MultiIndex).
- **right_index**
  Analogous to left_index.
- **sort**
  Sort merged data lexicographically by join keys; True by default (disable to get better performance in some cases on large datasets).
- **suffixes**
  Tuple of string values to append to column names in case of overlap; defaults to ('_x', '_y') (e.g., if 'data' in both DataFrame objects, would appear as 'data_x' and 'data_y' in result).
- **copy**
  If False, avoid copying data into resulting data structure in some exceptional cases; by default always copies.
- **indicator**
  Adds a special column _merge that indicates the source of each row; values will be 'left_only', 'right_only', or 'both' based on the origin of the joined data in each row.

In addition to merge function, we can also perform *concatenation* to combine 2 dataframes into one. This is useful for merging other data. For example, if we want to combine data from US and GB together, we can use concat.

```
gb_vdo_df = pd.read_csv('data/GBvideos.csv')
```

```
gb_vdo_df.shape
```

```
(38916, 16)
```

```
vdo_df.shape
```

```
(40901, 17)
```

```
pd.concat([vdo_df, gb_vdo_df], ignore_index=True, sort=True)
```

	category_id	channel_title	comment_count	comments_disabled	
0	22	CaseyNeistat	15954	False	SHA https://v
1	24	LastWeekTonight	12703	False	presid
2	23	Rudy Mancuso	8181	False	WA VIDE
3	24	Good Mythical Morning	2146	False	Today
4	24	nigahiga	17518	False	I I s
...	...	...	...	...	

79812	10	EnriqueIglesiasVEVO	9933	False	NE MIAM
79813	10	Jacob Sartorius	24330	False	THE C M
79814	10	Anne-Marie	19988	False	Ge HER
79815	24	Eurovision Song Contest	26766	False	Eleni
79816	10	SuperDuperKyle	1423	False	Debu

79817 rows × 17 columns

concat function arguments

- **objs**
  List or dict of pandas objects to be concatenated; this is the only required argument
- **axis**
  Axis to concatenate along; defaults to 0 (along rows)
- **join**
  Either 'inner' or 'outer' ('outer' by default); whether to intersection (inner) or union (outer) together indexes along the other axes
- **join_axes**
  Specific indexes to use for the other n–1 axes instead of performing union/intersection logic
- **keys**
  Values to associate with objects being concatenated, forming a hierarchical index along the concatenation axis; can either be a list or array of arbitrary values, an array of tuples, or a list of arrays (if multiple-level arrays passed in levels)
- **levels**
  Specific indexes to use as hierarchical index level or levels if keys passed
- **names**
  Names for created hierarchical levels if keys and/or levels passed
- **verify_integrity**
  Check new axis in concatenated object for duplicates and raise exception if so; by default (False) allows duplicates
- **ignore_index**
  Do not preserve indexes along concatenation axis, instead producing a new range(total_length) index

Aggregation and Group Operations

Categorizing a dataset and applying a function to each group, whether an aggregation or transformation, is often a critical component of a data analysis workflow. After loading, merging, and preparing a dataset, you may need to compute group statistics or possibly pivot tables for reporting or visualization purposes. pandas provides a flexible groupby interface, enabling you to slice, dice, and summarize datasets in a natural way.

Group Operation Mechanics

Group operations can be described using the concepts of *split-apply-combine*.

- **The first stage** - data contained in a pandas object, whether a Series, DataFrame, or otherwise, is split into groups based on one or more keys that you provide. The splitting is performed on a particular axis of an object.
- **The second stage** - a function is applied to each group, producing a new value.
- **The third stage** - the results of all those function applications are combined into a result object.

```
vdo_df_groupby_cat = vdo_df_withcat.groupby('category')
```

```
vdo_df_groupby_cat
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fe1d4406490>
```

```
vdo_df_groupby_cat.views
```

```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x7fe1d4171910>
```

```
vdo_df_groupby_cat.views.sum()
```

```
category
Autos & Vehicles           520690717
Comedy                    5111266590
Education                 1180175828
Entertainment            20561101882
Film & Animation          7267792432
Gaming                    2127799781
Howto & Style             4071011870
Music                    40126286541
News & Politics           1473090484
Nonprofits & Activism      168941392
People & Blogs            4910004664
Pets & Animals             764651989
Science & Technology      3473462753
Shows                       51501058
Sports                    4403213872
Travel & Events            343100609
Name: views, dtype: int64
```

```
vdo_df_groupby_cat.views.mean()
```

```
category
Autos & Vehicles         1.355965e+06
Comedy                   1.480239e+06
Education                7.130972e+05
Entertainment            2.067689e+06
Film & Animation         3.101917e+06
Gaming                   2.607598e+06
Howto & Style            9.828614e+05
Music                    6.204776e+06
News & Politics          5.927930e+05
Nonprofits & Activism    2.963884e+06
People & Blogs           1.530550e+06
Pets & Animals           8.311435e+05
Science & Technology     1.449088e+06
Shows                    9.035273e+05
Sports                   2.027262e+06
Travel & Events          8.556125e+05
Name: views, dtype: float64
```

```
vdo_df_groupby_cat.views.describe()
```

category	count	mean	std	min	25%	50%	
Autos & Vehicles	384.0	1.355965e+06	3.373464e+06	2860.0	104652.75	406278.0	107
Comedy	3453.0	1.480239e+06	2.010867e+06	1807.0	351261.00	976989.0	187
Education	1655.0	7.130972e+05	8.795104e+05	773.0	248987.50	419343.0	77
Entertainment	9944.0	2.067689e+06	5.821201e+06	798.0	272989.75	733789.5	173
Film & Animation	2343.0	3.101917e+06	5.572027e+06	943.0	302744.50	1274578.0	320
Gaming	816.0	2.607598e+06	3.144565e+06	1237.0	530424.00	1488158.0	320
Howto & Style	4142.0	9.828614e+05	1.929645e+06	1107.0	215273.00	502356.0	109
Music	6467.0	6.204776e+06	1.546525e+07	1591.0	382560.00	1434324.0	495
News & Politics	2485.0	5.927930e+05	1.119342e+06	549.0	50735.00	244014.0	68
Nonprofits & Activism	57.0	2.963884e+06	7.131112e+06	1456.0	11453.00	73649.0	31
People & Blogs	3208.0	1.530550e+06	3.460309e+06	884.0	202430.00	598419.0	168
Pets & Animals	920.0	8.311435e+05	1.102091e+06	3393.0	185072.25	444501.5	94
Science & Technology	2397.0	1.449088e+06	3.446784e+06	983.0	238839.00	582247.0	136

Accessing groups in groupby

```
for name, group in vdo_df_groupby_cat:
    print(name)
    print('----')
    print(type(group))
    print(group.columns)
```

```
<class 'pandas.core.frame.DataFrame'>
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id'
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description', 'trending_dt', 'id',
       'category'],
```

```
      dtype='object')
Howto & Style
----
<class 'pandas.core.frame.DataFrame'>
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id'
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description', 'trending_dt', 'id',
       'category'],
      dtype='object')
Music
----
<class 'pandas.core.frame.DataFrame'>
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id'
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description', 'trending_dt', 'id',
       'category'],
      dtype='object')
News & Politics
----
<class 'pandas.core.frame.DataFrame'>
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id'
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description', 'trending_dt', 'id',
       'category'],
      dtype='object')
Nonprofits & Activism
----
<class 'pandas.core.frame.DataFrame'>
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id'
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description', 'trending_dt', 'id',
       'category'],
      dtype='object')
People & Blogs
----
<class 'pandas.core.frame.DataFrame'>
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id'
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description', 'trending_dt', 'id',
       'category'],
      dtype='object')
Pets & Animals
----
<class 'pandas.core.frame.DataFrame'>
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id'
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description', 'trending_dt', 'id',
```

```
vdo_df_groupby_cat.get_group('Comedy')
```

	video_id	trending_date	title	channel_title	category_id	
13152	5qpjK5DgCt4	17.14.11	Racist Superman | Rudy Mancuso, King Bach & Le...	Rudy Mancuso	23	1
13153	ZAQs-ctOqXQ	17.14.11	One Change That Would Make Pacific Rim a Classic	Cracked	23	1
13154	lZ68j2J_GOM	17.14.11	Using Other People's Showers	Gus Johnson	23	1
13155	dQvIbulWCM4	17.14.11	Celebrities on Thanksgiving 2017!	Niki and Gabi	23	1
13156	t4YAyT4ihIQ	17.14.11	Getting My Driver's License | Lele Pons	Lele Pons	23	1
...	...	...	...	...	...	
16600	w8AJBlmLMZM	18.14.06	So This is Basically Fire Emblem	JelloApocalypse	23	2
16601	fYUXvrF85uQ	18.14.06	Guess the movie in 4 words! (YIAY #418)	jacksfilms	23	2
16602	p1jllz43xZE	18.14.06	Ellie Kemper and Mindy Kaling Reminisce About ...	Late Night with Seth Meyers	23	2
16603	JNyZ49q4vrU	18.14.06	Buying Used Things 2	Domics	23	2
			Mindy Kaling Is Mad She	The Tonight		

16604	amtC28yfYCM	18.14.06	Wasn't Invited to the ...	Show Starring Jimmy Fallon	23	2

3453 rows × 19 columns

Like statistics of “Music” category over time?

```
music_cat = vdo_df_withcat[vdo_df_withcat.category == 'Music']
```

```
music_groupby_trending_date = music_cat.groupby('trending_dt')
```

```
music_groupby_trending_date
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fe1d41aba50>
```

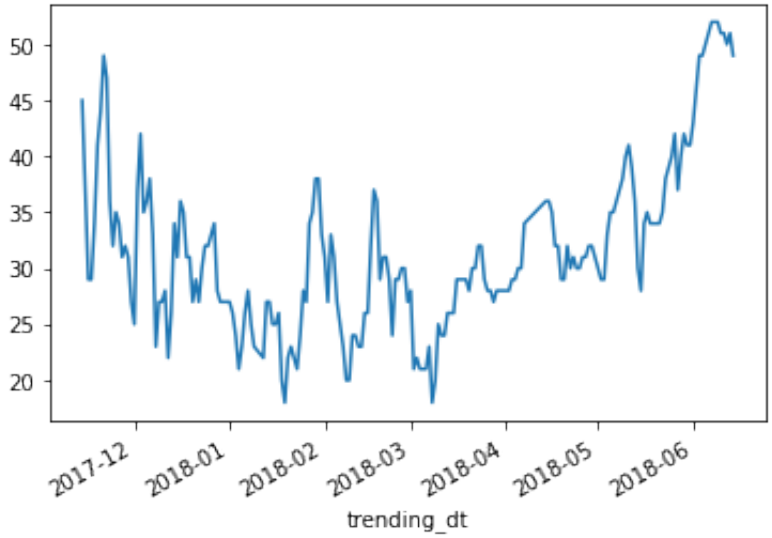
```
music_count_bydate = music_groupby_trending_date.video_id.count()
```

```
music_count_bydate
```

```
trending_dt
2017-11-14 00:00:00+00:00    45
2017-11-15 00:00:00+00:00    37
2017-11-16 00:00:00+00:00    29
2017-11-17 00:00:00+00:00    29
2017-11-18 00:00:00+00:00    34
                             ..
2018-06-10 00:00:00+00:00    51
2018-06-11 00:00:00+00:00    51
2018-06-12 00:00:00+00:00    50
2018-06-13 00:00:00+00:00    51
2018-06-14 00:00:00+00:00    49
Name: video_id, Length: 205, dtype: int64
```

```
music_count_bydate.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe1dbcab4d0>
```




```
music_groupby_trending_date.likes.sum()
```

```
trending_dt
2017-11-14 00:00:00+00:00     4722174
2017-11-15 00:00:00+00:00     4170707
2017-11-16 00:00:00+00:00     4146265
2017-11-17 00:00:00+00:00     3363007
2017-11-18 00:00:00+00:00     4419657
                               ...
2018-06-10 00:00:00+00:00    14986557
2018-06-11 00:00:00+00:00    15109828
2018-06-12 00:00:00+00:00    15583857
2018-06-13 00:00:00+00:00    16278888
2018-06-14 00:00:00+00:00    15954098
Name: likes, Length: 205, dtype: int64
```

```
music_groupby_trending_date.likes.sum().plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe1d4023cd0>
```