Let's test your knowledge about Python, data structures, NumPy, and pandas.

We are going to createa a dummy dataset for some Youtubers regarding their channals. We will create data including:

1. Decide the size of the dataset by using a SIZE constant variable
2. ChannelID (`Y0001, Y0002,...`)
3. Number of subscribers
4. Number of videos
5. Number of total views
6. Category (`['music','news','gaming','food','travel']`)
7. Language (`['English','Spanish','Japanese','Franch','Russia','Chinese']`)
8. Type(`['Corperate','Goverment','NGO','Individual']`)

Then we will play with the dummy dataset with our knowledge of pandas, including:

1. Access
2. Sampling
3. Filtering
4. Aggregation
5. Manipulation

Setup environment

Let's import random, Numpy, and pandas

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

Dummy dataset generation

Setup SIZE

```
# We make this variable SIZE because it supposed to be constant for the runtime.
SIZE = 10000
```

Generate ChannelID

```
ids = ['Y{0:04d}'.format(x) for x in range(SIZE)]
ids[:5], ids[-5:]
```

```
    (['Y0000', 'Y0001', 'Y0002', 'Y0003', 'Y0004'],
     ['Y9995', 'Y9996', 'Y9997', 'Y9998', 'Y9999'])
```

Generate # of subscribers

We first simulate a normal distribution

```
subs = np.random.normal(100000, 50000, (SIZE))
subs[:5], subs[-5:]
```

```
(array([ 28045.65114534, 112407.63315093,  75610.76598577,  68566.13035803,
        154427.69953221]),
 array([119716.26919506, 111044.52008844,  63854.96432585, 233726.38844679,
         66319.49650509]))
```

▾ We convert the float numbers to integer

```
subs = subs.astype(int)
subs[:5], subs[-5:]
```

```
(array([ 28045, 112407,  75610,  68566, 154427]),
 array([119716, 111044,  63854, 233726,  66319]))
```

▾ We check if there are non-positive numbers of subscribers

```
subs[subs <= 0 ]
```

```
array([-43555, -10281,  -9595,  -9578, -31035, -16535, -26981,   -228,
        -8651, -11262,  -4140,  -7238,  -8909, -30504,   -352,  -1969,
        -6037, -20941,  -3692, -29614, -24630, -20008, -11979, -42748,
        -3704,  -1762, -13142, -15088,  -2583, -15622, -11416, -47707,
       -10122, -10234, -36979,  -1200, -21980, -31153, -32859, -15957,
       -62227,  -3793, -12368,  -3275,  -4564, -40634, -36394,  -5238,
        -7393, -58930, -33036,  -7504,  -5827, -31303, -28492, -18510,
       -18404, -10121, -10687, -33655,  -1446, -10605, -44041,  -8288,
       -12448, -23102,  -8137,  -6630, -36058, -67239, -28889, -35601,
       -64242, -18212, -41231,  -7935,  -7460, -25507,  -3606, -28889,
       -13148,  -9799, -16342,  -3244, -17594, -16562,  -3315,  -7712,
       -22337, -14823, -74174, -61623, -20826,  -4478,   -751, -51497,
        -8895,  -1238,   -671,  -2369,  -4539,  -5003, -20124,  -1109,
       -36848,  -9377, -29740, -11504,   -872,  -7431,  -1976,  -3418,
       -52816, -14244, -19265, -21366,  -4352, -22007,  -2803, -36957,
       -24975, -19583, -34099,  -4300,  -1676,  -9936,  -2713, -10280,
        -7408, -22221, -43022,  -5841,  -2058,  -3612,  -9231,  -4218,
       -20115, -71255, -20533, -34103, -58353, -31651, -43687, -19968,
        -2633,  -9376, -13424,  -4648, -34092, -20428, -11563, -25526,
        -7475, -15464,  -4244, -37158, -24598, -10461, -12078, -12484,
        -7941,  -9496, -46764,  -2953,  -7706, -15757, -12330,  -7315,
       -15160, -33276, -16303, -23248, -31327, -19299, -13196, -40630,
        -8535,  -2588,  -1191, -18142, -20926,  -6107,  -8816, -19274,
       -47385, -66442, -41380, -34599, -14872, -25928,   -278, -18912,
       -43631, -40395, -53495, -49751, -10702,  -8777,  -3902, -23689,
       -39780, -18228, -21412,  -2807,   -561, -31575, -13888, -12623,
        -1383, -49558,   -215, -17362,  -2396, -34213,  -2704,  -4379,
        -4687, -12341, -21761, -25723, -50481, -21887,  -1316, -18339,
       -13852, -29685, -20372,  -8933,  -7702, -71676,  -3248, -37297,
       -10876,  -5547, -42003,  -4571,  -7150, -25219, -25145, -10334,
        -7466,  -1286, -36384, -27059, -27509,  -8768, -18319, -29419,
        -3898, -38358, -20016, -19863, -25943, -10858,  -6447,  -1623,
       -35629, -46525, -26406,  -5532])
```

▾ We set them to be 1

```
subs[subs <= 0 ] = 1
subs[subs <= 0 ]
```

```
array([], dtype=int64)
```

▾ Generate number of videos

We use a uniform distribution this time (just for practice)

```
nvideos = [np.random.randint(1, 100) for i in range(SIZE)]
nvideos[:5], nvideos[-5:]
```

```
([61, 8, 89, 17, 92], [68, 19, 66, 11, 69])
```

We simulate the number of views

To make it real, we will use the # of subscribers and # of videos as factors to get the # of total views

```
views = [ int(x*1.5*np.random.random() + y*2*np.random.random() + np.random.randint(-1000, 1000))
for x, y in zip(subs, nvideos)]
views[:5], views[-5:]
```

```
([34515, 137830, 75561, 14567, 172224], [5922, 58326, 78399, 78299, 3725])
```

```
views = np.array(views)
views
```

```
array([ 34515, 137830,  75561, ...,  78399,  78299,   3725])
```

```
views[views < 0]
```

```
array([-736,  -28, -520, -780, -181,  -86, -477, -469, -715, -968, -590,
       -500, -227, -514, -349, -361, -307, -748,  -26, -155, -207, -388,
       -950, -543, -182, -821, -647, -787, -646, -691, -305, -140, -734,
        -75,  -67, -190, -635, -783, -706, -235,  -35, -474,  -30,  -76,
       -660, -448, -126,  -60, -735, -356, -739, -756, -880,  -79, -664,
       -493, -194, -431, -865, -695, -768, -876, -113,  -53, -577, -119,
       -860,  -65, -684, -582, -525,  -76, -265, -224, -562, -742, -717,
       -334, -528, -108, -506, -628, -724, -603, -914, -526, -351, -552,
        -76, -900, -562, -116, -781, -884, -700, -537, -152,  -78, -912,
       -245,  -70, -746, -770, -560, -771, -215, -926, -244, -814, -506,
       -679, -779, -406, -178, -761, -875, -847, -547, -238, -243, -704,
       -233, -804, -542, -353, -561,  -38, -421, -797, -449, -707, -853,
       -212, -309, -730, -649, -262, -791, -379, -950,  -66,  -72, -859,
       -497, -376, -111])
```

```
views[views < 0] = 0
views[views < 0]
```

```
array([], dtype=int64)
```

Category

> Category (`['music','news','gaming','food','travel']`)

```
category = np.random.choice(['music','news','gaming','food','travel'], SIZE)
category[:20]
```

```
array(['food', 'food', 'food', 'food', 'gaming', 'food', 'news', 'food',
       'food', 'news', 'music', 'music', 'food', 'gaming', 'news',
       'travel', 'travel', 'news', 'gaming', 'gaming'], dtype='<U6')
```

Language

> Language (`['English','Spanish','Japanese','Franch','Russia','Chinese']`)

```
language = np.random.choice(['English','Spanish','Japanese','Franch','Russia','Chinese'],
                            SIZE, p=[0.5, 0.1, 0.1, 0.1, 0.1, 0.1])
language[:20]
```

```
array(['Russia', 'English', 'Franch', 'English', 'English', 'English',
       'English', 'Franch', 'English', 'English', 'English', 'English',
       'English', 'English', 'English', 'Russia', 'Russia', 'English',
       'Chinese', 'Russia'], dtype='<U8')
```

Type

> Type(`['Corperate','Goverment','NGO','Individual']`)

```
tp = np.random.choice(['Corperate','Goverment','NGO','Individual'], SIZE, p=[0.1, 0.2, 0.3, 0.4])
tp[:20]
```

```
array(['Individual', 'Goverment', 'Goverment', 'Individual', 'NGO',
       'Individual', 'NGO', 'Goverment', 'Corperate', 'Individual',
       'Goverment', 'NGO', 'NGO', 'Goverment', 'Corperate', 'Goverment',
       'Individual', 'NGO', 'NGO', 'NGO'], dtype='<U10')
```

Now we have all attributes, let's put them into a dataframe

```
df = pd.DataFrame({'ChannelID': ids,
                   'subs': subs,
                   'nvideos': nvideos,
                   'views': views,
                   'Category': category,
                   'Language':language,
                   'Type': tp},
                  index = ids)
df.head()
```

	ChannelID	subs	nvideos	views	Category	Language	Type
Y0000	Y0000	28045	61	34515	food	Russia	Individual
Y0001	Y0001	112407	8	137830	food	English	Goverment
Y0002	Y0002	75610	89	75561	food	Franch	Goverment
Y0003	Y0003	68566	17	14567	food	English	Individual
Y0004	Y0004	154427	92	172224	gaming	English	NGO

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10000 entries, Y0000 to Y9999
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   ChannelID  10000 non-null  object
 1   subs       10000 non-null  int64 
 2   nvideos    10000 non-null  int64 
 3   views      10000 non-null  int64 
 4   Category   10000 non-null  object
 5   Language   10000 non-null  object
 6   Type       10000 non-null  object
dtypes: int64(3), object(4)
memory usage: 625.0+ KB
```

Let's save the dummy dataset to `youtube_channels.csv`

```
df.to_csv('/content/youtube_channels.csv', index=False)
```

Let's play with the dataframe a little bit

Sampling

Select certain rows

```
df_sub1 = df.loc[:'Y0100']
df_sub1
```

	ChannelID	subs	nvideos	views	Category	Language	Type
Y0000	Y0000	28045	61	34515	food	Russia	Individual
Y0001	Y0001	112407	8	137830	food	English	Goverment
Y0002	Y0002	75610	89	75561	food	Franch	Goverment
Y0003	Y0003	68566	17	14567	food	English	Individual
Y0004	Y0004	154427	92	172224	gaming	English	NGO
...	...	...	...	...	...	...	...
Y0096	Y0096	72594	18	8188	news	Franch	Individual
Y0097	Y0097	125410	90	152805	travel	English	Individual
Y0098	Y0098	137602	73	676	music	English	Individual
Y0099	Y0099	119988	34	52080	news	English	NGO
Y0100	Y0100	105222	19	69281	gaming	English	Individual

101 rows × 7 columns

Select certain rows and columns

```
df_sub2 = df.loc[:'Y1000', ['subs','views']]
df_sub2
```

	subs	views
Y0000	28045	34515
Y0001	112407	137830
Y0002	75610	75561

Select randome rows

Y0004	154427	172224					

```
df_sub3 = df.iloc[random.sample(range(0, SIZE), 100)]
df_sub3
```

	ChannelID	subs	nvideos	views	Category	Language	Type
Y7934	Y7934	1	58	479	news	English	NGO
Y4974	Y4974	48515	47	25066	food	Russia	NGO
Y8898	Y8898	99912	73	1385	food	Russia	Individual
Y1580	Y1580	37341	13	34180	gaming	Chinese	Goverment
Y4149	Y4149	173878	19	226348	music	Japanese	NGO
...	...	...	...	...	...	...	...
Y8747	Y8747	101132	80	150711	news	Russia	Individual
Y6329	Y6329	114735	80	170020	food	English	Individual
Y0242	Y0242	178802	55	220842	news	Russia	Individual
Y2615	Y2615	219678	1	254137	music	English	Corperate
Y2270	Y2270	84831	39	102813	music	English	Individual

100 rows × 7 columns

Select random rows with selected columns

```
df_sub4 = df.iloc[random.sample(range(0, SIZE), 100)][['subs','views']]
df_sub4
```

	subs	views
Y7633	79223	40161
Y7965	77167	40551
Y4786	28367	35095
Y1991	179690	176777
Y1831	41955	4277
...	...	...
Y8427	93260	4679
Y6860	185962	266192
Y1589	166517	67476
Y3098	64708	2307
Y4297	90637	55938

100 rows × 2 columns

Filtering

Super Popular Channel

```
df_superp = df[df['subs'] > 300000]
df_superp
```

	ChannelID	subs	nvideos	views	Category	Language	Type

Popular Channel

```
df_p = df[df['subs'] > 100000]
df_p
```

	ChannelID	subs	nvideos	views	Category	Language	Type
Y0001	Y0001	112407	8	137830	food	English	Goverment
Y0004	Y0004	154427	92	172224	gaming	English	NGO
Y0005	Y0005	171466	61	183685	food	English	Individual
Y0008	Y0008	142673	87	154982	food	English	Corperate
Y0013	Y0013	112532	20	74012	gaming	English	Goverment
...	...	...	...	...	...	...	...
Y9993	Y9993	149002	3	140254	gaming	Japanese	Goverment
Y9994	Y9994	100716	70	90956	news	Japanese	NGO
Y9995	Y9995	119716	68	5922	food	English	Goverment
Y9996	Y9996	111044	19	58326	music	Franch	Individual
Y9998	Y9998	233726	11	78299	gaming	English	NGO

4954 rows × 7 columns

Start up channel

```
df_begin = df[df['subs'] < 100]
df_begin
```

	ChannelID	subs	nvideos	views	Category	Language	Type
Y0123	Y0123	1	40	0	gaming	English	Individual
Y0189	Y0189	1	17	0	music	English	Individual

Popular English Channel

```
df_EnglishP = df[(df['subs'] > 100000) & (df['Language'] == 'English')]
df_EnglishP
```

	ChannelID	subs	nvideos	views	Category	Language	Type
Y0001	Y0001	112407	8	137830	food	English	Goverment
Y0004	Y0004	154427	92	172224	gaming	English	NGO
Y0005	Y0005	171466	61	183685	food	English	Individual
Y0008	Y0008	142673	87	154982	food	English	Corperate
Y0013	Y0013	112532	20	74012	gaming	English	Goverment
...	...	...	...	...	...	...	...
Y9979	Y9979	135756	35	83628	gaming	English	NGO
Y9984	Y9984	152586	41	8405	travel	English	NGO
Y9985	Y9985	115796	8	146382	music	English	Individual
Y9995	Y9995	119716	68	5922	food	English	Goverment
Y9998	Y9998	233726	11	78299	gaming	English	NGO

2508 rows × 7 columns

Gaming channel with many videos

```
df_gaming_nv = df[(df['Category'] == 'gaming') & (df['nvideos'] > 90)]
df_gaming_nv
```

	ChannelID	subs	nvideos	views	Category	Language	Type
Y0004	Y0004	154427	92	172224	gaming	English	NGO
Y0039	Y0039	114186	96	145534	gaming	English	Individual
Y0043	Y0043	83225	99	2881	gaming	English	Individual
Y0075	Y0075	54639	95	70018	gaming	Franch	Individual
Y0299	Y0299	119074	99	79042	gaming	English	Individual
...	...	...	...	...	...	...	...
Y9750	Y9750	101044	96	148319	gaming	English	NGO
Y9784	Y9784	100741	91	83593	gaming	Franch	Individual
Y9793	Y9793	83465	99	0	gaming	English	NGO
Y9924	Y9924	71820	92	106613	gaming	English	NGO
Y9987	Y9987	72528	91	28345	gaming	English	Goverment

207 rows × 7 columns

Non-Corperate and News channel

```
df_nc_news = df[(df['Category'] == 'news') & (df['Type'] != 'Corperate')]
df_nc_news
```

	ChannelID	subs	nvideos	views	Category	Language	Type
Y0006	Y0006	25407	30	29163	news	English	NGO
Y0009	Y0009	42078	66	42632	news	English	Individual
Y0017	Y0017	182756	86	150412	news	English	NGO
Y0027	Y0027	87038	54	70244	news	English	Individual
Y0050	Y0050	93176	96	43607	news	Franch	NGO
...	...	...	...	...	...	...	...
Y9964	Y9964	32626	86	14193	news	Spanish	Individual
Y9966	Y9966	81409	78	90670	news	Japanese	Individual
Y9969	Y9969	73792	19	105253	news	Spanish	Individual
Y9994	Y9994	100716	70	90956	news	Japanese	NGO
Y9997	Y9997	63854	66	78399	news	English	Individual

1792 rows × 7 columns

Top Subs **and** Top nvideos

```
df_subs_and_nvideos = df[(df['subs'] > 200000) & (df['nvideos'] > 90)]
df_subs_and_nvideos
```

	ChannelID	subs	nvideos	views	Category	Language	Type
Y0457	Y0457	200578	98	179438	music	English	NGO
Y1234	Y1234	270322	91	246792	music	English	Individual
Y1752	Y1752	252441	97	94488	music	Spanish	NGO
Y1966	Y1966	239748	95	234882	gaming	English	Goverment
Y2118	Y2118	203434	94	251128	gaming	Russia	Goverment
Y2264	Y2264	209237	99	214449	news	English	Individual
Y2369	Y2369	215407	91	222893	travel	English	Individual
Y2655	Y2655	216743	98	294638	gaming	Chinese	NGO
Y2788	Y2788	253750	95	41430	food	Russia	Individual

Top Subs **or** Top nvideos

Y3638	Y3638	203695	92	105152	travel	English	Individual

```
df_subs_or_nvideos = df[(df['subs'] > 200000) | (df['nvideos'] > 90)]
df_subs_or_nvideos
```

	ChannelID	subs	nvideos	views	Category	Language	Type
Y0004	Y0004	154427	92	172224	gaming	English	NGO
Y0010	Y0010	94659	92	19164	music	English	Goverment
Y0018	Y0018	214402	24	213773	gaming	Chinese	NGO
Y0034	Y0034	77831	95	30652	food	English	NGO
Y0039	Y0039	114186	96	145534	gaming	English	Individual
...	...	...	...	...	...	...	...
Y9955	Y9955	87258	96	52558	travel	Franch	Goverment
Y9973	Y9973	120056	98	109683	food	English	NGO
Y9977	Y9977	131093	94	186946	travel	English	Individual
Y9987	Y9987	72528	91	28345	gaming	English	Goverment
Y9998	Y9998	233726	11	78299	gaming	English	NGO

1098 rows × 7 columns

Aggregation

```
byCategory = df.groupby('Category')
byCategory.sum()
```

```
<ipython-input-31-1bf2018b2b87>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum i
  byCategory.sum()
```

```
byCategory.mean()
```

```
<ipython-input-32-4fe0643856b0>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean
  byCategory.mean()
```

Category	subs	nvideos	views
food	100040.159226	49.306548	75921.875992
gaming	99108.617032	50.172311	76012.443725
music	100346.642495	50.544625	75803.384888
news	99632.784855	50.510532	75957.311936
travel	99516.782587	49.083582	74892.036816

```
byLanguage = df.groupby('Language')
byLanguage.sum()
```

```
<ipython-input-33-7ff598080e9a>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum i
  byLanguage.sum()
```

Language	subs	nvideos	views
Chinese	104324228	53327	79330991
English	496591607	250622	376477909
Franch	101118223	49236	76222840
Japanese	106116938	51277	83271965
Russia	92832378	46971	72219624
Spanish	96287775	47765	69644309

```
byLanguage.mean()
```

```
<ipython-input-34-b9813ba23235>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean
  byLanguage.mean()
```

Language	subs	nvideos	views
Chinese	99831.797129	51.030622	75914.823923
English	99697.170648	50.315599	75582.796426
Franch	101320.864729	49.334669	76375.591182
Japanese	101450.227533	49.021989	79609.909178
Russia	97308.572327	49.235849	75701.911950
Spanish	98655.507172	48.939549	71356.873975

```
byLanType = df.groupby(['Language', 'Type'])
byLanType.describe()
```

		subs								nvideos	
		count	mean	std	min	25%	50%	75%	max	count	mean
Language	Type										
Chinese	Corperate	110.0	99457.336364	50733.441208	1.0	62248.00	98675.5	135087.50	237038.0	110.0	46.336364
	Goverment	237.0	102804.552743	52212.092706	1.0	70381.00	100310.0	134932.00	263176.0	237.0	52.801688
	Individual	406.0	95629.667488	50165.313076	1.0	58861.00	96165.5	132292.75	247324.0	406.0	51.647783
	NGO	292.0	103402.729452	51112.394733	1.0	69207.50	101904.5	135814.75	261685.0	292.0	50.503425
English	Corperate	509.0	96385.499018	51289.569872	1.0	59867.00	94118.0	130254.00	286875.0	509.0	48.263261
	Goverment	942.0	100659.650743	48636.473989	1.0	66303.00	102239.0	135400.00	239748.0	942.0	51.132696
	Individual	2042.0	99512.581293	49236.312216	1.0	66363.00	100693.0	132786.75	282862.0	2042.0	51.214985
	NGO	1488.0	100473.995968	49121.816322	1.0	66272.75	100893.0	132323.75	260881.0	1488.0	49.266129
Franch	Corperate	115.0	95224.339130	47015.224298	1.0	63358.50	93194.0	128991.00	255294.0	115.0	51.347826
	Goverment	184.0	106364.016304	49287.658376	1.0	68470.25	105015.0	143614.25	224179.0	184.0	47.043478
	Individual	402.0	102360.087065	48107.306405	1.0	69731.00	99148.5	136601.75	251861.0	402.0	51.221393
	NGO	297.0	99150.471380	49538.230429	1.0	61748.00	98632.0	134409.00	251957.0	297.0	47.420875
Japanese	Corperate	99.0	107543.101010	48240.195506	1.0	70032.00	101998.0	142026.00	222586.0	99.0	49.929293
	Goverment	213.0	99929.657277	45414.076484	1.0	70109.00	100804.0	131386.00	215839.0	213.0	50.938967
	Individual	437.0	100754.350114	46178.322297	1.0	67055.00	98685.0	132725.00	245272.0	437.0	49.114416
	NGO	297.0	101533.680135	50876.264972	1.0	65658.00	99841.0	137068.00	234421.0	297.0	47.208754
Russia	Corperate	78.0	97518.435897	42844.391905	1.0	68983.75	92531.5	125842.50	198685.0	78.0	49.666667
	Goverment	189.0	100941.333333	49441.830860	1.0	70928.00	99725.0	131467.00	235418.0	189.0	47.952381
	Individual	378.0	96058.542328	48338.538666	1.0	61912.00	95125.0	131080.50	253750.0	378.0	47.878307
	NGO	309.0	96562.779935	49256.079654	1.0	63280.00	98793.0	128026.00	247106.0	309.0	51.572816
Spanish	Corperate	93.0	92909.806452	46227.832268	1.0	64779.00	92387.0	117625.00	224218.0	93.0	49.225806
	Goverment	188.0	101609.914894	47198.701891	1.0	64861.75	102255.0	139183.75	199215.0	188.0	46.526596
	Individual	399.0	100184.558897	49745.512587	1.0	66689.50	100733.0	136009.50	235658.0	399.0	49.518797
	NGO	296.0	96523.175676	49759.531255	1.0	61615.50	94835.0	131885.50	252441.0	296.0	49.601351

24 rows × 24 columns

```
byLanType.describe()[['subs','views']]
```

		subs							views	
		count	mean	std	min	25%	50%	75%	max	count	mean
Language	Type										
Chinese	Corperate	110.0	99457.336364	50733.441208	1.0	62248.00	98675.5	135087.50	237038.0	110.0	71285.6818
	Goverment	237.0	102804.552743	52212.092706	1.0	70381.00	100310.0	134932.00	263176.0	237.0	74188.1476
	Individual	406.0	95629.667488	50165.313076	1.0	58861.00	96165.5	132292.75	247324.0	406.0	75170.8990
	NGO	292.0	103402.729452	51112.394733	1.0	69207.50	101904.5	135814.75	261685.0	292.0	80094.4863
English	Corperate	509.0	96385.499018	51289.569872	1.0	59867.00	94118.0	130254.00	286875.0	509.0	71423.1139
	Goverment	942.0	100659.650743	48636.473989	1.0	66303.00	102239.0	135400.00	239748.0	942.0	75625.1740
	Individual	2042.0	99512.581293	49236.312216	1.0	66363.00	100693.0	132786.75	282862.0	2042.0	75803.8878
	NGO	1488.0	100473.995968	49121.816322	1.0	66272.75	100893.0	132323.75	260881.0	1488.0	76675.4643
Franch	Corperate	115.0	95224.339130	47015.224298	1.0	63358.50	93194.0	128991.00	255294.0	115.0	68803.6869
	Goverment	184.0	106364.016304	49287.658376	1.0	68470.25	105015.0	143614.25	224179.0	184.0	78024.0326
	Individual	402.0	102360.087065	48107.306405	1.0	69731.00	99148.5	136601.75	251861.0	402.0	76677.9104
	NGO	297.0	99150.471380	49538.230429	1.0	61748.00	98632.0	134409.00	251957.0	297.0	77877.0168
Japanese	Corperate	99.0	107543.101010	48240.195506	1.0	70032.00	101998.0	142026.00	222586.0	99.0	87365.0404
	Goverment	213.0	99929.657277	45414.076484	1.0	70109.00	100804.0	131386.00	215839.0	213.0	83696.6901
	Individual	437.0	100754.350114	46178.322297	1.0	67055.00	98685.0	132725.00	245272.0	437.0	80041.7002

```
byLanType.describe().loc['English',['subs','views']]
```

	subs								views		
	count	mean	std	min	25%	50%	75%	max	count	mean	std
Type											
Corperate	509.0	96385.499018	51289.569872	1.0	59867.00	94118.0	130254.00	286875.0	509.0	71423.113949	60978.1
Goverment	942.0	100659.650743	48636.473989	1.0	66303.00	102239.0	135400.00	239748.0	942.0	75625.174098	60702.5
Individual	2042.0	99512.581293	49236.312216	1.0	66363.00	100693.0	132786.75	282862.0	2042.0	75803.887855	61317.2
NGO	1488.0	100473.995968	49121.816322	1.0	66272.75	100893.0	132323.75	260881.0	1488.0	76675.464382	60287.7

Manipulation

```
df['subpervideo'] = df['subs']/df['nvideos']
df['subpervideo'].describe()
```

```
count     10000.000000
mean       5261.265431
std       13647.291270
min           0.010101
25%        1160.077077
50%        1996.801082
75%        3985.608989
max      235418.000000
Name: subpervideo, dtype: float64
```

```
df['viewspervideo'] = df['views']/df['nvideos']
df['viewspervideo'].describe()
```

```
count     10000.000000
mean       3984.010792
std       11965.410055
min           0.000000
25%         533.549567
50%        1348.045304
75%        3020.480612
max      254193.000000
Name: viewspervideo, dtype: float64
```

```
df['viewspersub'] = df['views']/df['subs']
df['viewspersub'].describe()
```

```
count    10000.000000
mean         8.448969
std         73.398496
min          0.000000
25%          0.376409
50%          0.766664
75%          1.140467
max       1134.000000
Name: viewspersub, dtype: float64
```

```
df.head(10)
```

	ChannelID	subs	nvideos	views	Category	Language	Type	subpervideo	viewspervideo	viewspersub
Y0000	Y0000	28045	61	34515	food	Russia	Individual	459.754098	565.819672	1.230701
Y0001	Y0001	112407	8	137830	food	English	Goverment	14050.875000	17228.750000	1.226169
Y0002	Y0002	75610	89	75561	food	Franch	Goverment	849.550562	849.000000	0.999352
Y0003	Y0003	68566	17	14567	food	English	Individual	4033.294118	856.882353	0.212452
Y0004	Y0004	154427	92	172224	gaming	English	NGO	1678.554348	1872.000000	1.115245
Y0005	Y0005	171466	61	183685	food	English	Individual	2810.918033	3011.229508	1.071262
Y0006	Y0006	25407	30	29163	news	English	NGO	846.900000	972.100000	1.147833
Y0007	Y0007	90808	74	133107	food	Franch	Goverment	1227.135135	1798.743243	1.465807
Y0008	Y0008	142673	87	154982	food	English	Corperate	1639.919540	1781.402299	1.086274
Y0009	Y0009	42078	66	42632	news	English	Individual	637.545455	645.939394	1.013166

Congratulations! You completed our course: Data Wrangling -- Fundamentals