Setting

set language to korean on matplotlib

In [ ]:

```
!apt-get update -qq
!apt-get install fonts-nanum* -qq
```

In [1]:

```
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm
fm._rebuild()
plt.rc('font', family='NanumBarunGothic')
```

In [ ]:

```
for fontInfo in fm.fontManager.ttflist:
    if 'Nanum' in fontInfo.name:
        print(fontInfo.name+" = "+fontInfo.fname)
```

set disply plotly chart

In [ ]:

```
!pip install -U kaleido
```

Data Analysis and Visualization

In [3]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:

```
file_path = "/content/drive/MyDrive/ㄱ□ㅇㄹ□ㄱㅍ□ㅇ□ㄹ/4. ㅍ□ㄹ□ㅇ□ㄴㅂ□ㄷ□ㅇ/assignment2_analysis/pretest_data.csv"
```

EDA

In [5]:

```
data = pd.read_csv(file_path, parse_dates=['published_date', 'on_trending_date', 'off_trending_date'], infer_date
time_format=True)
data.head()
```

Out[5]:

	video_id	channel_id	published_date	category_name	duration	tags	description	on_trending_date	off_trendi
0	V-0db	CH49ta0	2021-07-01	Entertainment	PT8M20S	SiriusXM|Sirius XM|Sirius|SXM|BIGHIT|빅히트|방탄소년단...	BTS performs their hit songs ‘Dynamite’ and ‘B...	2021-07-03	202
1	V-1XL	CHZVD--	2021-06-24	Entertainment	PT9M17S	치킨불냉면|치킨|불냉면|냉면	영상에 나오는 캐릭터의 이름은 파도비 입니다. 고양이가 아니라 파란 도깨비입니다. ...	2021-06-26	202
2	V-4fa	CH9w-h_	2021-07-17	Entertainment	PT7M39S	NaN	거세 구형, 성 충동 제거를 위한 엄벌 치료 VS 인권 보호해야고민 끝에 내린 강요...	2021-07-19	202
3	V-5ip	CHUQVGX	2021-06-02	Sports	PT6M40S	News Network|SBS SPORTSMUG|SPORTSMUG|스포츠머그|축구|...	세계 최초 9회 연속 올림픽 본선 진출! 그동안 한국 축구의 역사를 써내려 간 올림...	2021-06-04	202
4	V-5jn	CHhI3EX	2021-07-06	Sports	PT11M27S	이천수|심판도전기|축구심판	찾아 뵐 심판분들이 이제 18명정도 남았네요	2021-07-08	202

5 rows × 25 columns

column	description
video_id	영상의 비디오 아이디
channel_id	영상이 업로드 되어있는 채널 아이디
published_date	영상이 유튜브에 업로드된 날짜
category_name	영상/채널의 카테고리
duration	영상 길이 (PT1H13M25S==1시간13분25초)
tags	영상에 사용된 해시태그
description	영상부연설명
on_trending_date	인기 동영상에서 처음 포착된 날짜
off_trending_date	인기 동영상에서 사라진 날짜
on_rank	인기 동영상에서 처음 기록된 순위
off_rank	인기 동영상에서 사라지기 전 기록된 순위
on_views	인기 동영상에서 처음 기록된 조회수
off_views	인기 동영상에서 사라지기전 기록된 조회수
on_likes	인기 동영상에서 처음 기록된 좋아요수
off_likes	인기 동영상에서 사라지기전 기록된 좋아요수
on_dislikes	인기 동영상에서 처음 기록된 싫어요수
off_dislikes	인기 동영상에서 사라지기전 기록된 싫어요수
on_comments	인기 동영상에서 처음 기록된 댓글수
off_comments	인기 동영상에서 사라지기전 기록된 댓글수
on_channel_subscribers	인기 동영상에서 처음 기록된 채널의 구독자수
off_channel_subscribers	인기 동영상에서 사라지기전 기록된 채널의 구독자수
on_channel_total_vies	인기 동영상에서 처음 기록된 채널의 전체 비디오 조회수의 합
off_channel_total_vies	인기 동영상에서 사라지기전 기록된 채널의 전체 비디오 조회수의 합
on_channel_total_videos	인기 동영상에서 처음 기록된 채널의 비디오 개수
off*channel*)total_vidios	인기 동영상에서 사라지기전 기록된 채널의 비디오 개수

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2644 entries, 0 to 2643
Data columns (total 25 columns):
 #   Column                   Non-Null Count  Dtype         
---  ------                   --------------  -----         
 0   video_id                 2644 non-null   object        
 1   channel_id               2644 non-null   object        
 2   published_date           2644 non-null   datetime64[ns]
 3   category_name            2644 non-null   object        
 4   duration                 2644 non-null   object        
 5   tags                     2274 non-null   object        
 6   description              2604 non-null   object        
 7   on_trending_date         2644 non-null   datetime64[ns]
 8   off_trending_date        2644 non-null   datetime64[ns]
 9   on_rank                  2644 non-null   int64         
 10  off_rank                 2644 non-null   int64         
 11  on_views                 2644 non-null   int64         
 12  off_views                2644 non-null   int64         
 13  on_likes                 2644 non-null   int64         
 14  off_likes                2644 non-null   int64         
 15  on_dislikes              2644 non-null   int64         
 16  off_dislikes             2644 non-null   int64         
 17  on_comments              2644 non-null   int64         
 18  off_comments             2644 non-null   int64         
 19  on_channel_subscribers   2644 non-null   int64         
 20  off_channel_subscribers  2644 non-null   int64         
 21  on_channel_total_views   2644 non-null   int64         
 22  off_channel_total_views  2644 non-null   int64         
 23  on_channel_total_videos  2644 non-null   int64         
 24  off_channel_total_videos  2644 non-null   int64         
dtypes: datetime64[ns](3), int64(16), object(6)
memory usage: 516.5+ KB
```

make tags data to list.

In [7]:

```
data["tags_list"] = data.tags.str.split('|')
```

In [8]:

```
data.head()
```

Out[8]:

	video_id	channel_id	published_date	category_name	duration	tags	description	on_trending_date	off_tren
0	V-0db	CH49ta0	2021-07-01	Entertainment	PT8M20S	SiriusXM|Sirius XM|Sirius|SXM|BIGHIT|빅히트|방탄소년단...	BTS performs their hit songs ‘Dynamite’ and ‘B...	2021-07-03	2
1	V-1XL	CHZVD--	2021-06-24	Entertainment	PT9M17S	치킨불냉면|치킨|불냉면|냉면	영상에 나오는 캐릭터의 이름은 파도비 입니다. 고양이가 아니라 파란 도깨비입니다. ...	2021-06-26	2
2	V-4fa	CH9w-h_	2021-07-17	Entertainment	PT7M39S	NaN	거세 구형, 성 충동 제거를 위한 엄벌 치료 VS 인권 보호해야고민 끝에 내린 강요...	2021-07-19	2
3	V-5ip	CHUQVGX	2021-06-02	Sports	PT6M40S	News Network|SBS SPORTSMUG|SPORTSMUG|스포츠머그|축구|...	세계 최초 9회 연속 올림픽 본선 진출! 그동안 한국 축구의 역사를 써내려 간 올림...	2021-06-04	2
4	V-5jn	CHHl3EX	2021-07-06	Sports	PT11M27S	이천수|심판도전기|축구심판	찾아 뵐 심판분들이 이제 18명정도 남았네요	2021-07-08	2

5 rows × 26 columns

Q1. 데이터 타입별 시각화

- 전체기간 카테고리->채널->비디오 개수
- 월별 카테고리->채널->비디오 개수
- 월별 TOP10 채널 (분류 기준은 비디오 개수)
- 주별 TOP5 채널 (분류 기준은 비디오 개수)
- 월별 카테고리별 태그 키워드 순위

The number of channels for each category over the entire period

In [9]:

```
group_table = data.groupby(['category_name'])['channel_id'].count().to_frame()
group_table = group_table.sort_values(['channel_id'], ascending=False)
```

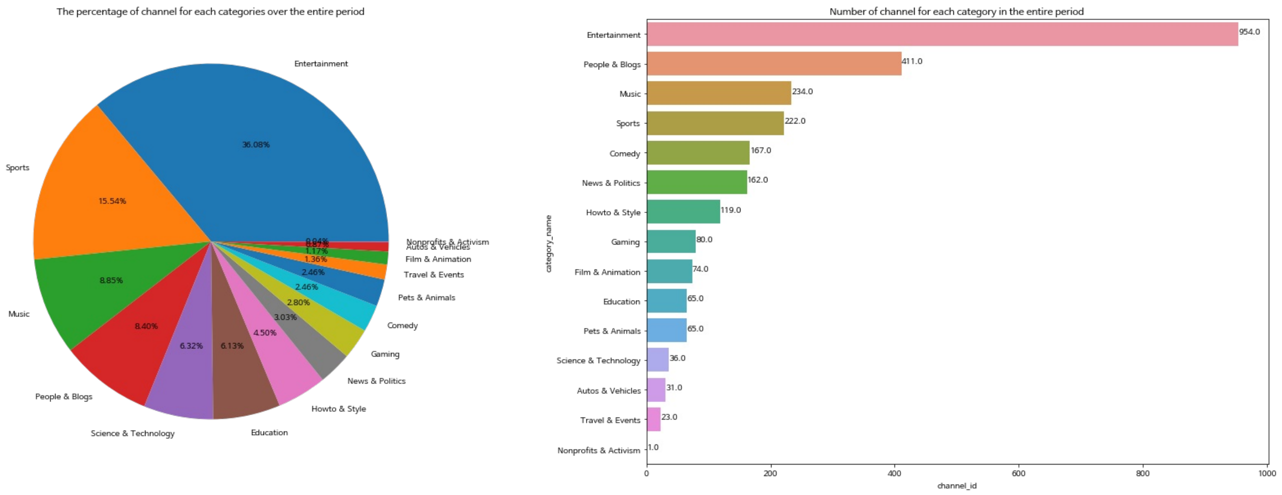
In [10]:

```
def show_values(axs, orient="v", space=.01):
    def _single(ax):
        if orient == "v":
            for p in ax.patches:
                _x = p.get_x() + p.get_width() / 2
                _y = p.get_y() + p.get_height() + (p.get_height()*0.01)
                value = '{:.1f}'.format(p.get_height())
                ax.text(_x, _y, value, ha="center")
        elif orient == "h":
            for p in ax.patches:
                _x = p.get_x() + p.get_width() + float(space)
                _y = p.get_y() + p.get_height() - (p.get_height()*0.5)
                value = '{:.1f}'.format(p.get_width())
                ax.text(_x, _y, value, ha="left")

    if isinstance(axs, np.ndarray):
        for idx, ax in np.ndenumerate(axs):
            _single(ax)
    else:
        _single(axs)
```

In [11]:

```
fig,axes = plt.subplots(1, 2, figsize=(30,10))

category_name = data.category_name.unique()
axes[0].pie(group_table,
        labels=category_name,
        autopct='%1.2f%%',
        startangle=0)
axes[0].set_title("The percentage of channel for each categories over the entire period")

axes[1] = sns.barplot(x='channel_id', y=group_table.index, data=group_table)
show_values(axes[1], "h")
axes[1].set_title("Number of channel for each category in the entire period")

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: MatplotlibDeprecationWarning: Non-1D
inputs to pie() are currently squeeze()d, but this behavior is deprecated since 3.1 and will be remo
ved in 3.3; pass a 1D array instead.
  import sys
```



Over the entire period, The categories with many channels are in the order of **Entertainment, People & Blogs, Music, Sports**.

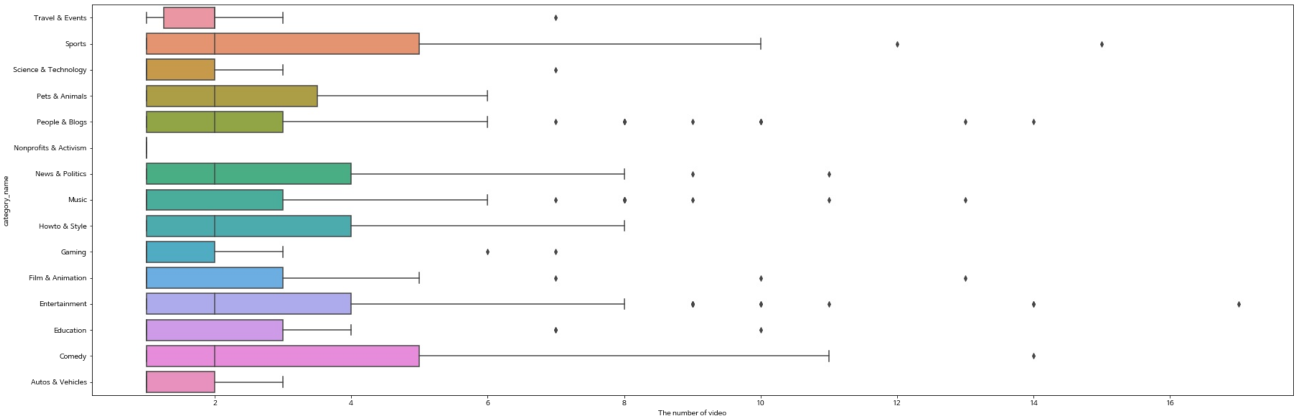
The number of videos on the channel over the entire period (by category)

In [12]:

```
group_table = data.groupby(['category_name', 'channel_id'], as_index=False)['video_id'].count()
group_table = group_table.sort_values(['category_name','video_id'], ascending=False)
```

In [13]:

```
fig,axes = plt.subplots(figsize=(30,10))
sns.boxplot(x="video_id", y="category_name",
              data=group_table)
plt.xlabel("The number of video")
plt.show()
```



In [14]:

```
group_table.groupby('category_name')['video_id'].agg(['max', 'min', 'mean']).sort_values('mean', ascending=False)
```

Out[14]:

	max	min	mean
category_name			
Sports	15	1	3.363636
Comedy	14	1	3.340000
Entertainment	17	1	3.057692
News & Politics	11	1	2.892857
Film & Animation	13	1	2.740741
Howto & Style	8	1	2.704545
People & Blogs	14	1	2.475904
Education	10	1	2.407407
Pets & Animals	6	1	2.407407
Music	13	1	2.387755
Travel & Events	7	1	2.300000
Science & Technology	7	1	1.800000
Gaming	7	1	1.702128
Autos & Vehicles	3	1	1.631579
Nonprofits & Activism	1	1	1.000000

Sports had the highest average number of videos, and **Entertainment** had the highest number of videos.

The channel with the most videos in each category

In [15]:

```
group_table = data.groupby(['category_name', 'channel_id'], as_index=False)['video_id'].count()
group_table = group_table.groupby(['category_name'], as_index=False)['channel_id', 'video_id'].max()
group_table.sort_values('video_id', ascending=False)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Indexing with multipl
e keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```

Out[15]:

	category_name	channel_id	video_id
3	Entertainment	CHzt24f	17
13	Sports	CHvMwK5	15
1	Comedy	CHzjiRW	14
10	People & Blogs	CHzhyI5	14
4	Film & Animation	CHz2Kbg	13
7	Music	CHzxXBQ	13
8	News & Politics	CHzz58-	11
2	Education	CHznImS	10
6	Howto & Style	CHzIOIS	8
5	Gaming	CHzgNzU	7
12	Science & Technology	CHrBpV_	7
14	Travel & Events	CHsLoTw	7
11	Pets & Animals	CHzjDgV	6
0	Autos & Vehicles	CHyF14S	3
9	Nonprofits & Activism	CHSsWdU	1

The number of channels in each category per month

In [16]:

```
data_datetime_index = data.set_index('published_date')
```

In [17]:

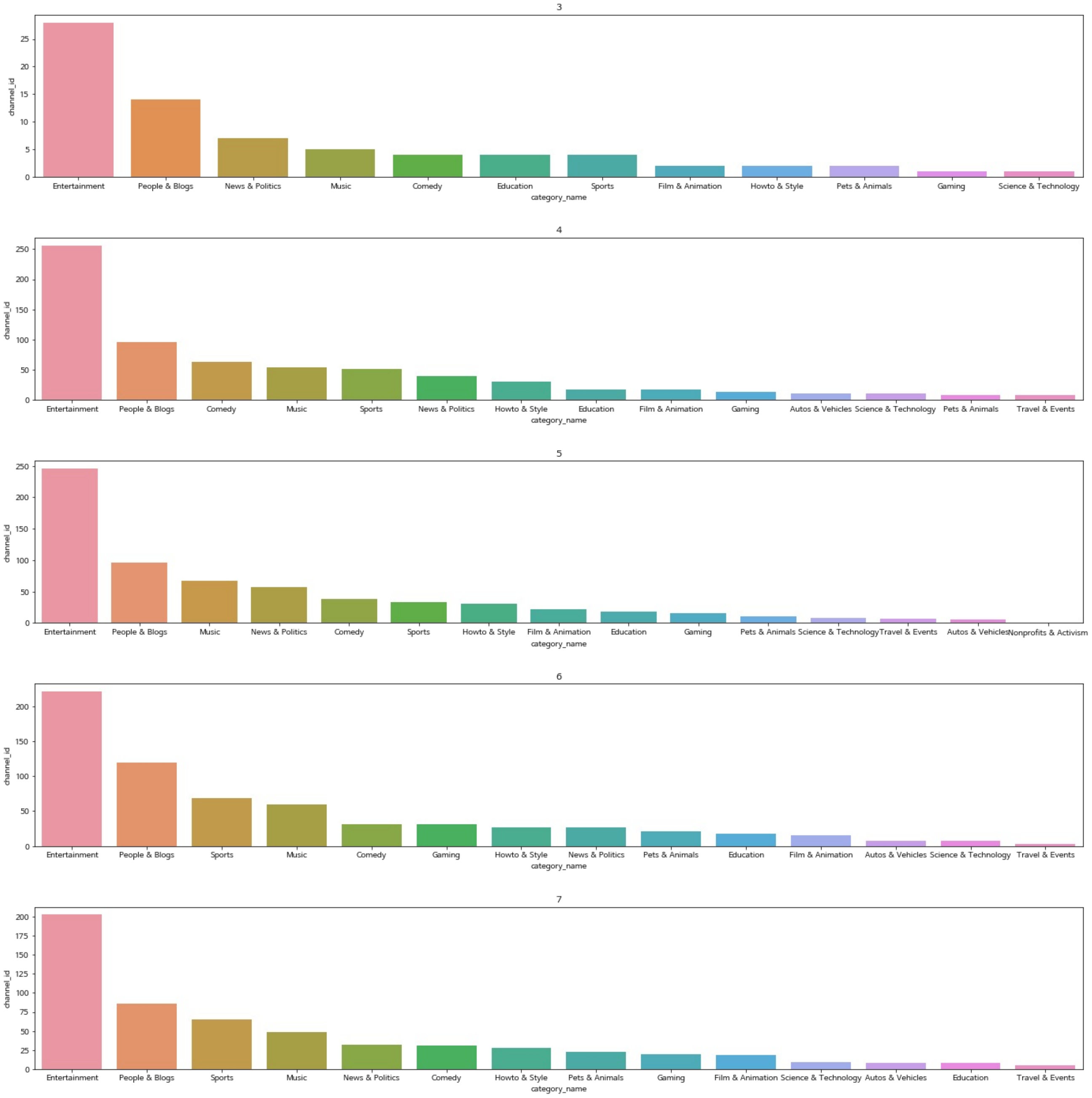
```
data_datetime_index_group = data_datetime_index.groupby(by=[data_datetime_index.index.month,'category_name'])['ch
annel_id'].count().reset_index()
```

In [18]:

```
monthes = data_datetime_index_group['published_date'].unique()
monthes.sort()
```

In [19]:

```
fig, axes = plt.subplots(nrows=len(monthes), ncols=1, figsize=(20, 20))

for idx, ax_y in enumerate(axes):
    temp_month_data = data_datetime_index_group.loc[(data_datetime_index_group['published_date']==monthes[idx])]
    temp_month_data = temp_month_data.sort_values(['channel_id'], ascending=False)
    sns.barplot(x=temp_month_data['category_name'], y=temp_month_data['channel_id'], ax=ax_y)
    ax_y.set_title(monthes[idx])

fig.tight_layout(pad=3.0)

plt.show()
```



Category with a lot of channels per month (Top5)

In [20]:

```
group_top5_data = {'3':data_datetime_index_group.loc[(data_datetime_index_group['published_date']==3)].nlargest(5
, 'channel_id')['category_name'].to_list(),
 '4':data_datetime_index_group.loc[(data_datetime_index_group['published_date']==4)].nlargest(5, 'channel_id')['c
ategory_name'].to_list(),
 '5':data_datetime_index_group.loc[(data_datetime_index_group['published_date']==5)].nlargest(5, 'channel_id')['c
ategory_name'].to_list(),
 '6':data_datetime_index_group.loc[(data_datetime_index_group['published_date']==6)].nlargest(5, 'channel_id')['c
ategory_name'].to_list(),
 '7':data_datetime_index_group.loc[(data_datetime_index_group['published_date']==7)].nlargest(5, 'channel_id')['c
ategory_name'].to_list()}
```

In [21]:

```
group_table_top5 = pd.DataFrame(group_top5_data)
group_table_top5
```

Out[21]:

	3	4	5	6	7
0	Entertainment	Entertainment	Entertainment	Entertainment	Entertainment
1	People & Blogs	People & Blogs	People & Blogs	People & Blogs	People & Blogs
2	News & Politics	Comedy	Music	Sports	Sports
3	Music	Music	News & Politics	Music	Music
4	Comedy	Sports	Comedy	Comedy	News & Politics

In common, **Entertainment** and **People & Blogs** have the largest number of channels.

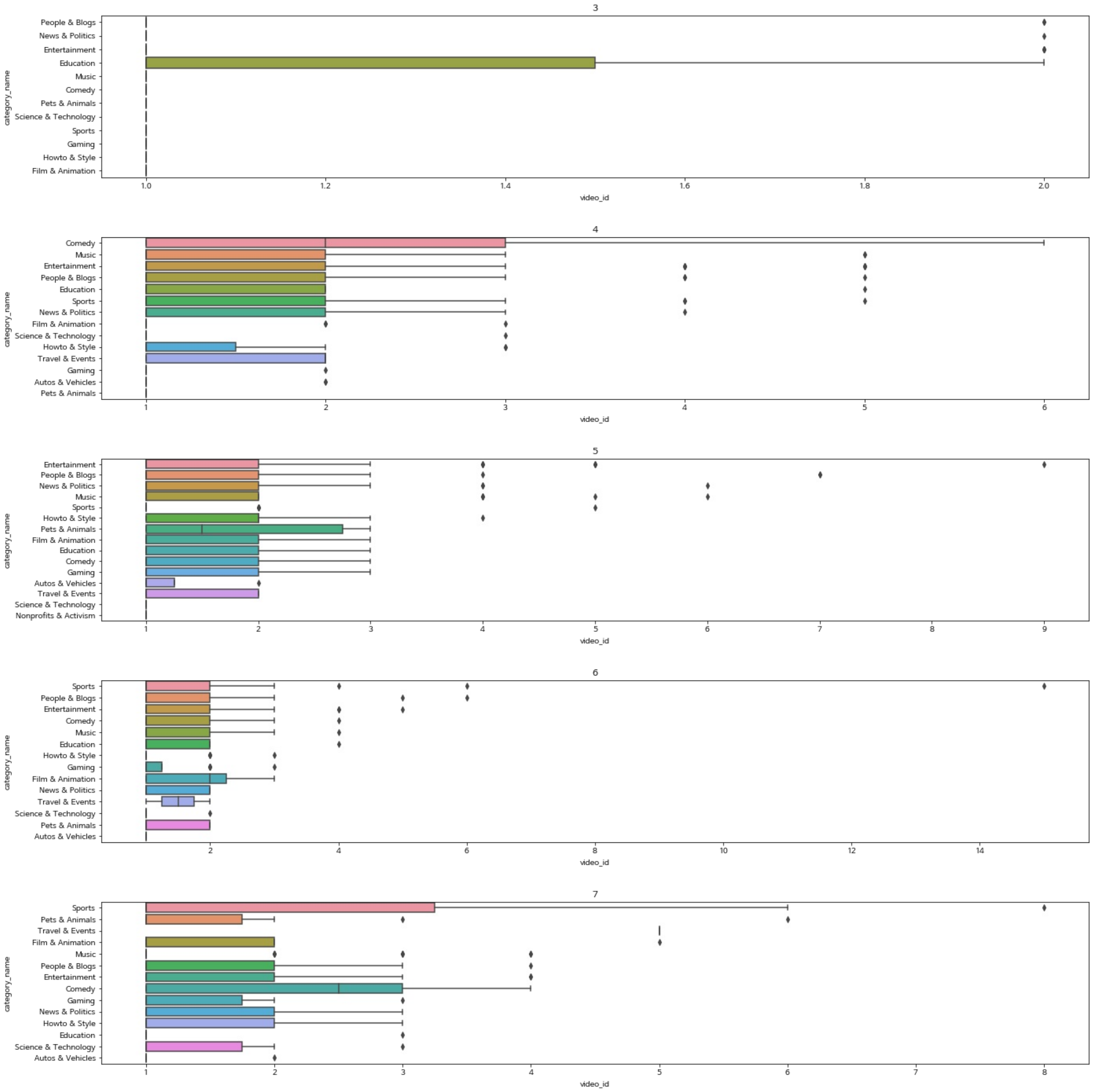
The number of videos on the channel per month (by category)

In [22]:

```
data_datetime_index_group = data_datetime_index.groupby(by=[data_datetime_index.index.month,'category_name','chan
nel_id'])['video_id'].count().reset_index()
```

In [23]:

```
fig, axes = plt.subplots(nrows=len(monthes), ncols=1, figsize=(20, 20))

for idx, ax_y in enumerate(axes):
    temp_month_data = data_datetime_index_group.loc[(data_datetime_index_group['published_date']==monthes[idx])]
    temp_month_data = temp_month_data.sort_values(['video_id'], ascending=False)
    sns.boxplot(x="video_id", y="category_name",
              data=temp_month_data, ax=ax_y)
    ax_y.set_title(monthes[idx])

fig.tight_layout(pad=3.0)

plt.show()
```



The Number of video on March (Max, Min, Mean)

In [24]:

```
group_temp = data_datetime_index_group.loc[(data_datetime_index_group['published_date']==3)]
group_temp.groupby('category_name')['video_id'].agg(['max', 'min', 'mean']).sort_values('mean', ascending=False)
```

Out[24]:

	max	min	mean
category_name			
Education	2	1	1.333333
Entertainment	2	1	1.166667
News & Politics	2	1	1.166667
People & Blogs	2	1	1.166667
Comedy	1	1	1.000000
Film & Animation	1	1	1.000000
Gaming	1	1	1.000000
Howto & Style	1	1	1.000000
Music	1	1	1.000000
Pets & Animals	1	1	1.000000
Science & Technology	1	1	1.000000
Sports	1	1	1.000000

The Number of video on April (Max, Min, Mean)

In [25]:

```
group_temp = data_datetime_index_group.loc[(data_datetime_index_group['published_date']==4)]
group_temp.groupby('category_name')['video_id'].agg(['max', 'min', 'mean']).sort_values('mean', ascending=False)
```

Out[25]:

	max	min	mean
category_name			
Comedy	6	1	2.172414
Music	5	1	1.741935
Education	5	1	1.700000
Entertainment	5	1	1.630573
News & Politics	4	1	1.625000
Travel & Events	2	1	1.600000
Sports	5	1	1.593750
People & Blogs	5	1	1.573770
Howto & Style	3	1	1.347826
Film & Animation	3	1	1.307692
Autos & Vehicles	2	1	1.222222
Science & Technology	3	1	1.222222
Gaming	2	1	1.083333
Pets & Animals	1	1	1.000000

The Number of video on May (Max, Min, Mean)

In [26]:

```
group_temp = data_datetime_index_group.loc[(data_datetime_index_group['published_date']==5)]
group_temp.groupby('category_name')['video_id'].agg(['max', 'min', 'mean']).sort_values('mean', ascending=False)
```

Out[26]:

	max	min	mean
category_name			
Pets & Animals	3	1	1.833333
Howto & Style	4	1	1.823529
News & Politics	6	1	1.781250
Music	6	1	1.717949
Film & Animation	3	1	1.692308
People & Blogs	7	1	1.655172
Education	3	1	1.636364
Entertainment	9	1	1.607843
Gaming	3	1	1.454545
Travel & Events	2	1	1.400000
Comedy	3	1	1.357143
Sports	5	1	1.320000
Autos & Vehicles	2	1	1.250000
Nonprofits & Activism	1	1	1.000000
Science & Technology	1	1	1.000000

The Number of video on June (Max, Min, Mean)

In [27]:

```
group_temp = data_datetime_index_group.loc[(data_datetime_index_group['published_date']==6)]
group_temp.groupby('category_name')['video_id'].agg(['max', 'min', 'mean']).sort_values('mean', ascending=False)
```

Out[27]:

	max	min	mean
category_name			
Sports	15	1	2.156250
Film & Animation	3	1	1.875000
Comedy	4	1	1.631579
Entertainment	5	1	1.601449
Music	4	1	1.594595
People & Blogs	6	1	1.506329
Education	4	1	1.500000
Travel & Events	2	1	1.500000
Pets & Animals	2	1	1.312500
Gaming	3	1	1.291667
Howto & Style	3	1	1.285714
News & Politics	2	1	1.285714
Science & Technology	2	1	1.166667
Autos & Vehicles	1	1	1.000000

The Number of video on July (Max, Min, Mean)

In [28]:

```
group_temp = data_datetime_index_group.loc[(data_datetime_index_group['published_date']==7)]
group_table_max_min_mean = pd.DataFrame()
group_table_max_min_mean['max'] = group_temp.groupby('category_name')['video_id'].max()
group_table_max_min_mean['min'] = group_temp.groupby('category_name')['video_id'].min()
group_table_max_min_mean['mean'] = group_temp.groupby('category_name')['video_id'].mean()
group_table_max_min_mean.sort_values('mean', ascending=False)
```

Out[28]:

	max	min	mean
category_name			
Travel & Events	5	5	5.000000
Sports	8	1	2.321429
Comedy	4	1	2.214286
News & Politics	3	1	1.684211
Pets & Animals	6	1	1.642857
Film & Animation	5	1	1.636364
Entertainment	4	1	1.573643
People & Blogs	4	1	1.535714
Science & Technology	3	1	1.500000
Howto & Style	3	1	1.473684
Music	4	1	1.441176
Gaming	3	1	1.357143
Education	3	1	1.333333
Autos & Vehicles	2	1	1.142857

Top 10 channel on each month

In [29]:

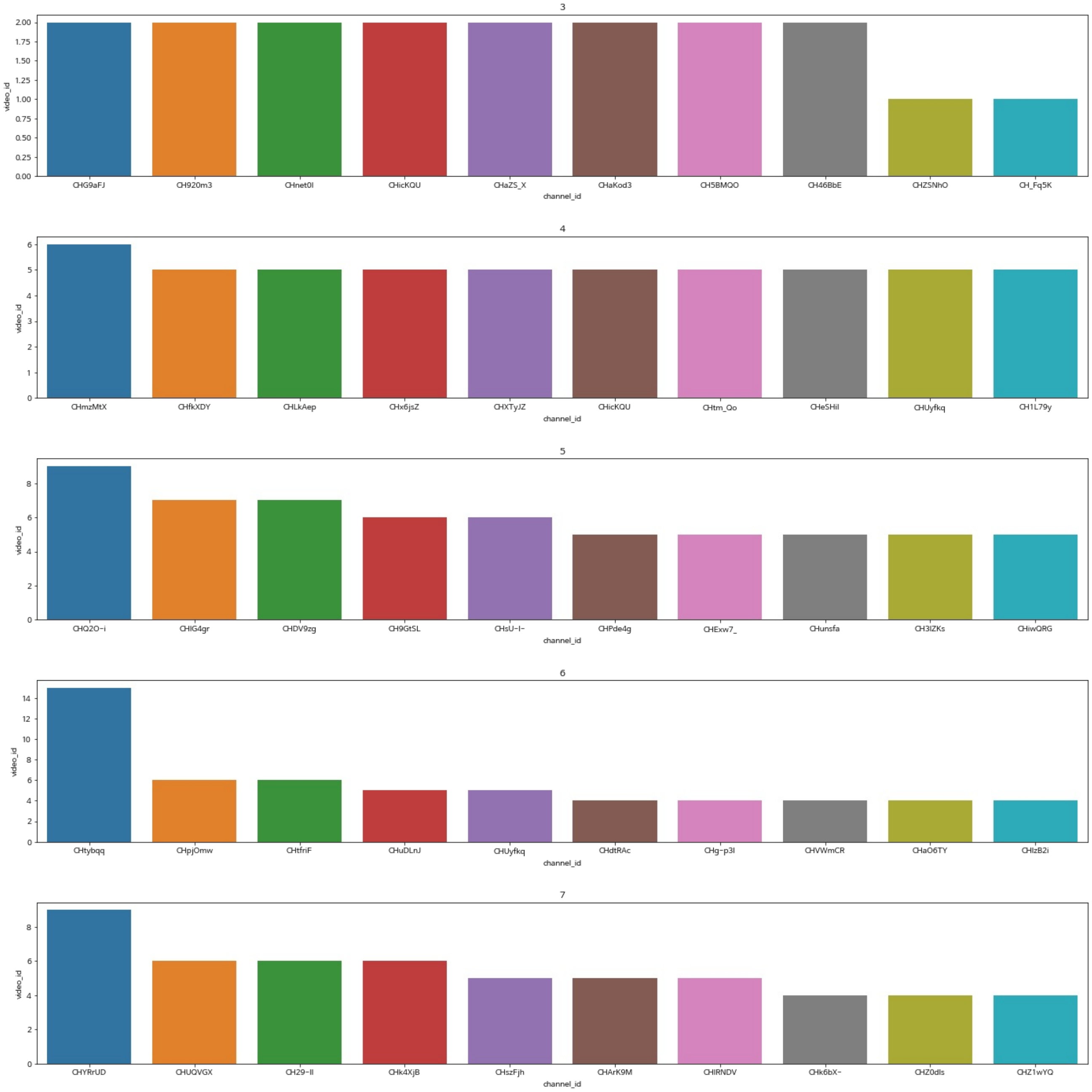
```
top10 = lambda x: x.sort_values(by='video_id', ascending=False)[:10]
```

In [30]:

```
group_temp = data_datetime_index.groupby(by=[data_datetime_index.index.month,'channel_id'])['video_id'].count().t
o_frame().reset_index()
group_temp = group_temp.groupby('published_date').apply(top10).reset_index(drop=True)
```

In [31]:

```
fig, axes = plt.subplots(nrows=len(monthes), ncols=1, figsize=(20, 20))

for idx, ax_y in enumerate(axes):
    temp_month_data = group_temp.loc[(group_temp['published_date']==monthes[idx])]
    temp_month_data = temp_month_data.sort_values(['video_id'], ascending=False)
    sns.barplot(x=temp_month_data['channel_id'], y=temp_month_data['video_id'], ax=ax_y)
    ax_y.set_title(monthes[idx])

fig.tight_layout(pad=3.0)

plt.show()
```



TOP 5 channel on week

In [32]:

```
top5 = lambda x: x.sort_values(by='video_id', ascending=False)[:5]
```

In [33]:

```
data_datetime_index_group = data_datetime_index.groupby(by=[data_datetime_index.index.week, 'channel_id'])['video_id'].count().reset_index()
data_datetime_index_group = data_datetime_index_group.groupby('published_date').apply(top5).reset_index(drop=True)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: weekofyear and week h
ave been deprecated, please use DatetimeIndex.isocalendar().week instead, which returns a Series.  T
o exactly reproduce the behavior of week and weekofyear and return an Index, you may call pd.Int64In
dex(idx.isocalendar().week)
  """Entry point for launching an IPython kernel.
```

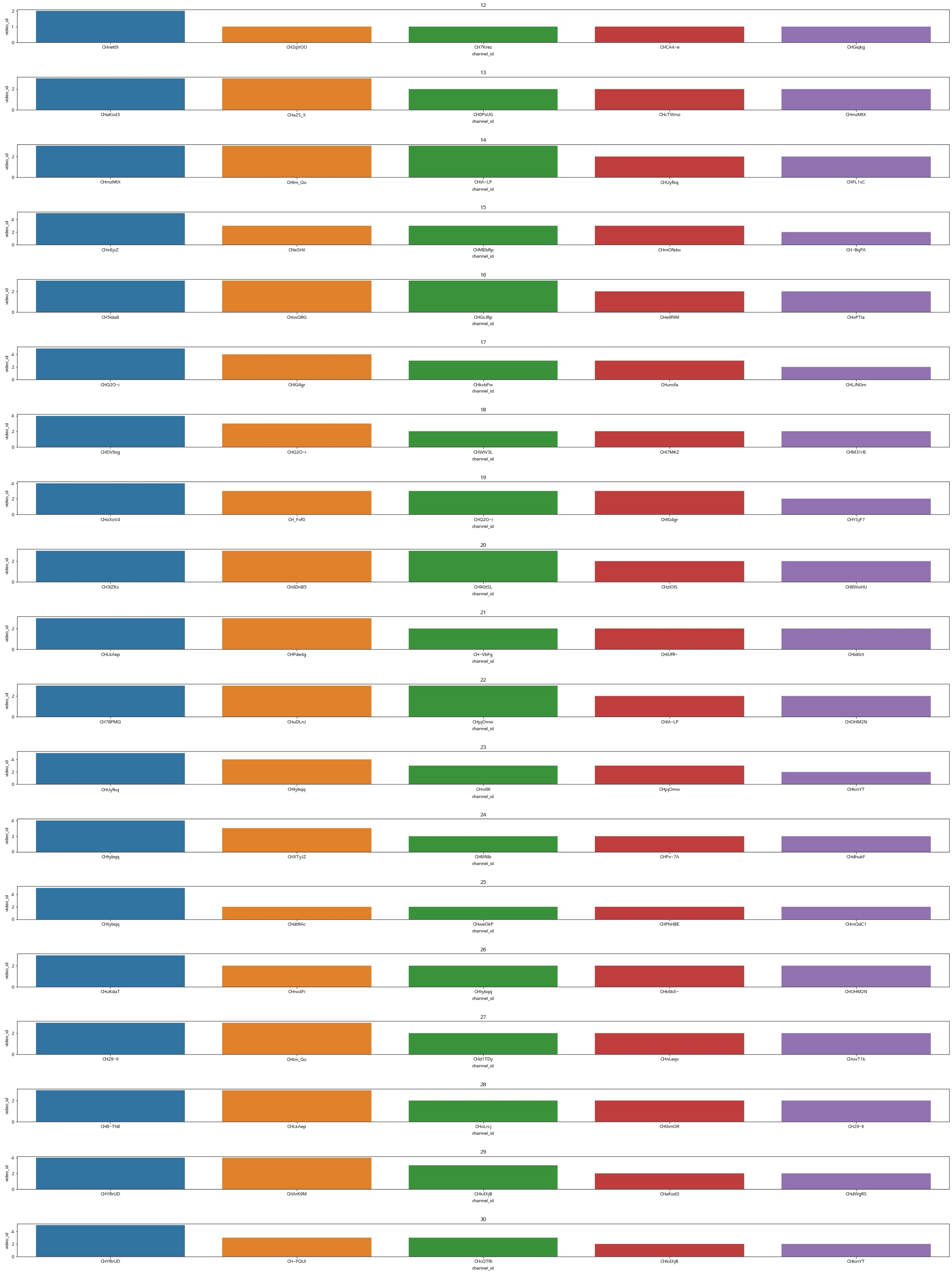
In [34]:

```
weekly = data_datetime_index.index.week.unique()
weekly = weekly.sort_values()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: weekofyear and week h
ave been deprecated, please use DatetimeIndex.isocalendar().week instead, which returns a Series.  T
o exactly reproduce the behavior of week and weekofyear and return an Index, you may call pd.Int64In
dex(idx.isocalendar().week)
  """Entry point for launching an IPython kernel.
```

In [35]:

```
fig, axes = plt.subplots(nrows=len(weekly), ncols=1, figsize=(30, 40))

for idx, ax_y in enumerate(axes):
    temp_month_data = data_datetime_index_group.loc[(data_datetime_index_group['published_date']==weekly[idx])]
    temp_month_data = temp_month_data.sort_values(['video_id'], ascending=False)
    sns.barplot(x=temp_month_data['channel_id'], y=temp_month_data['video_id'], ax=ax_y)
    ax_y.set_title(weekly[idx])

fig.tight_layout(pad=3.0)

plt.show()
```


12
video_id
0
1
2
CHnet0I
CH2qVOO
CH7Krez
CHCA4-e
CHGiqkg
channel_id
13
video_id
0
2
CHaKod3
CHaZS_X
CH0PsUG
CHcTWmz
CHmzMtX
channel_id
14
video_id
0
2
CHmzMtX
CHtm_Qo
CHIA-LP
CHUyfkq
CHFL1sC
channel_id
15
video_id
0
2
4
CHx6jsZ
CHeSHil
CHMEbRp
CHmONdw
CH-BqPA
channel_id
16
video_id
0
2
CH5lda8
CHiwQRG
CHGsJRp
CHe9f9M
CHoPTIa
channel_id
17
video_id
0
2
4
CHQ2O-i
CHIG4gr
CHkxbPw
CHunsfa
CHLJNGm
channel_id
18
video_id
0
2
4
CHDV9zg
CHQ2O-i
CHWIV3L
CHI7MKZ
CHM31rB
channel_id
19
video_id
0
2
4
CHoXoV4
CH_Fxf0
CHQ2O-i
CHIG4gr
CHYSjF7
channel_id
20
video_id
0
2
CH3IZKs
CH4DnB5
CH9GtSL
CHzlOIS
CH8WoHU
channel_id
21
video_id
0
2
CHLkAep
CHPde4g
CH-VbFg
CHIUfR-
CHId0ct
channel_id
22
video_id
0
2
CH78PMQ
CHuDLnJ
CHpjOmw
CHIA-LP
CHOHM2N
channel_id
23
video_id
0
2
4
CHUyfkq
CHtybqq
CHvil9I
CHpjOmw
CHkinYT
channel_id
24
video_id
0
2
4
CHtybqq
CHXTyJZ
CH6INIb
CHPx-7A
CHdhukF
channel_id
25
video_id
0
2
4
CHtybqq
CHdtRAc
CHweOkP
CHPhHBE
CHmQdC1
channel_id
26
video_id
0
2
CHuKdaT
CHnx4Fi
CHtybqq
CHk6bX-
CHOHM2N
channel_id
27
video_id
0
2
CH29-II
CHtm_Qo
CHd1TDy
CHnLeqv
CHoxT1k
channel_id
28
video_id
0
2
CH8-Th8
CHLkAep
CHoLrcj
CH0imOR
CH29-II
channel_id
29
video_id
0
2
4
CHYRrUD
CHArK9M
CHk4XjB
CHaKod3
CHdWgRS
channel_id
30
video_id
0
2
4
CHYRrUD
CH-FQUI
CHcQTRi
CHk4XjB
CHkinYT
channel_id

In [36]:

```
total_summarize_data = pd.DataFrame(columns=weekly)
for week in weekly:
    total_summarize_data[week] = data_datetime_index_group.loc[(data_datetime_index_group['published_date']==week
), 'channel_id'].values
total_summarize_data.reset_index(drop=True)
```

Out[36]:

published_date	12	13	14	15	16	17	18	19	20	21	22
0	CHnet0I	CHaKod3	CHmzMtX	CHx6jsZ	CH5Ida8	CHQ2O-i	CHDV9zg	CHoXoV4	CH3IZKs	CHLkAep	CH78PMQ
1	CH2qVOO	CHaZS_X	CHtm_Qo	CHeSHil	CHiwQRG	CHIG4gr	CHQ2O-i	CH_Fxf0	CH4DnB5	CHPde4g	CHuDLnJ
2	CH7Krez	CH0PsUG	CHIA-LP	CHMEbRp	CHGsJRp	CHkxbPw	CHWIV3L	CHQ2O-i	CH9GtSL	CH-VbFg	CHpjOmw
3	CHCA4-e	CHcTWmz	CHUyfkq	CHmONdw	CHe9f9M	CHunsfa	CHI7MKZ	CHIG4gr	CHzIOIS	CHIUfR-	CHIA-LP
4	CHGiqkg	CHmzMtX	CHFL1sC	CH-BqPA	CHoPTIa	CHLJNGm	CHM31rB	CHYSjF7	CH8WoHU	CHId0ct	CHOHM2N

The ranking of tag keywords by each category

In [37]:

```
data_date_index = data.set_index('published_date')
```

In [38]:

```
group_tag_by_monthly = data_date_index.groupby([data_datetime_index.index.month, 'category_name'])['tags_list'].s
um().to_frame()
```

In [39]:

```
import re
def preprocess(text_list):
    return list(map(lambda x: re.sub('([^\w가-힣 ])', '', x), text_list))
```

In [40]:

```
group_tag_by_monthly['clean_tags'] = group_tag_by_monthly['tags_list'].apply(preprocess)
```

In [41]:

```
from collections import Counter

group_tag_by_monthly['clean_tags_counter'] = group_tag_by_monthly['clean_tags'].apply(Counter)
```

In [42]:

```
data_rank = group_tag_by_monthly['clean_tags_counter'].apply(pd.Series).stack().reset_index().groupby(['published
_date', 'category_name', 'level_2']).sum().reset_index()
```

In [43]:

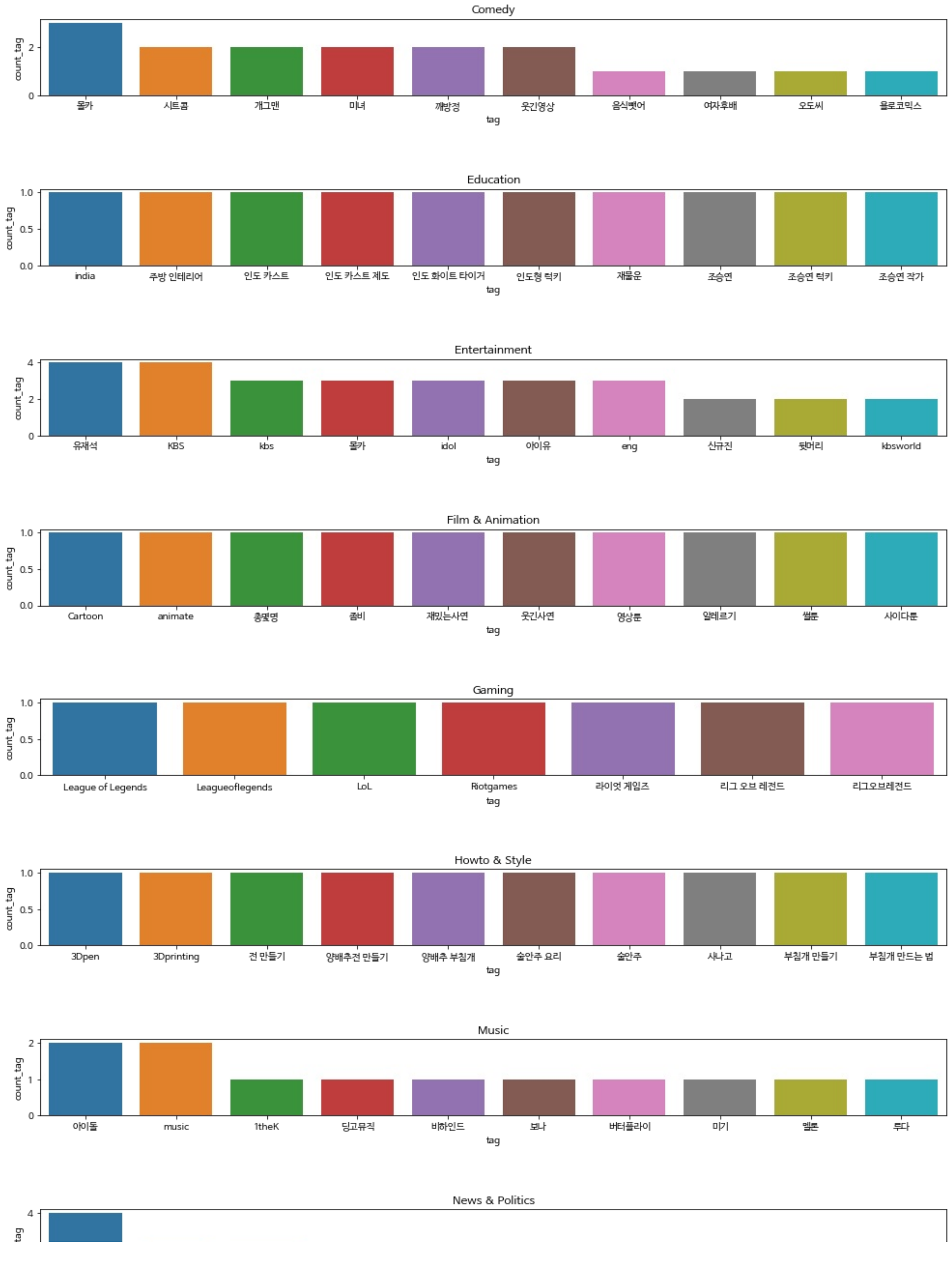
```
def tag_rank_monthly(month):
    categories = data_rank.loc[(data_rank['published_date']==month),'category_name'].unique()
    fig, axes = plt.subplots(nrows=len(categories), ncols=1, figsize=(15,30))
    for idx, ax in enumerate(axes):
        temp =  data_rank.loc[((data_rank['published_date']==month) & (data_rank['category_name']==categories[idx
]))]
        temp = temp.rename(columns={"level_2" : "tag", 0:"count_tag"})
        temp = temp.sort_values('count_tag', ascending=False)[:10]
        temp['count_tag'] = temp['count_tag'].astype(int)

        if len(temp) == 0:
            continue
        sns.barplot(x="tag", y='count_tag', data=temp, ax=ax)
        ax.set_title(categories[idx])

    fig.tight_layout(pad=5.0)
    plt.show()
```

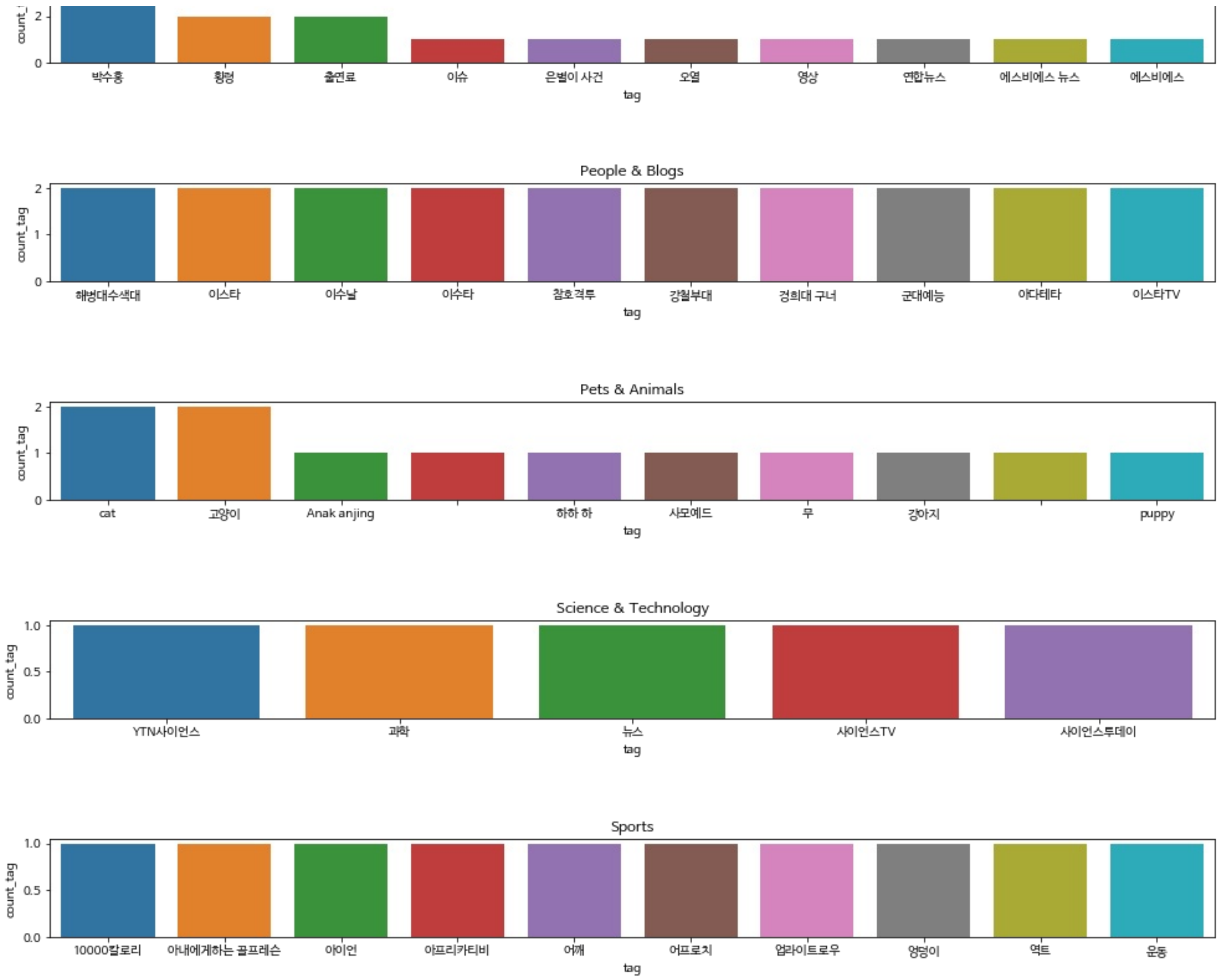
March

In [44]:

```
tag_rank_monthly(3)
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning: Glyph
2346 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning: Glyph
2354 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning: Glyph
2348 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning: Glyph
2346 missing from current font.
  font.set_text(s, 0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning: Glyph
2354 missing from current font.
  font.set_text(s, 0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning: Glyph
2348 missing from current font.
  font.set_text(s, 0, flags=flags)
```

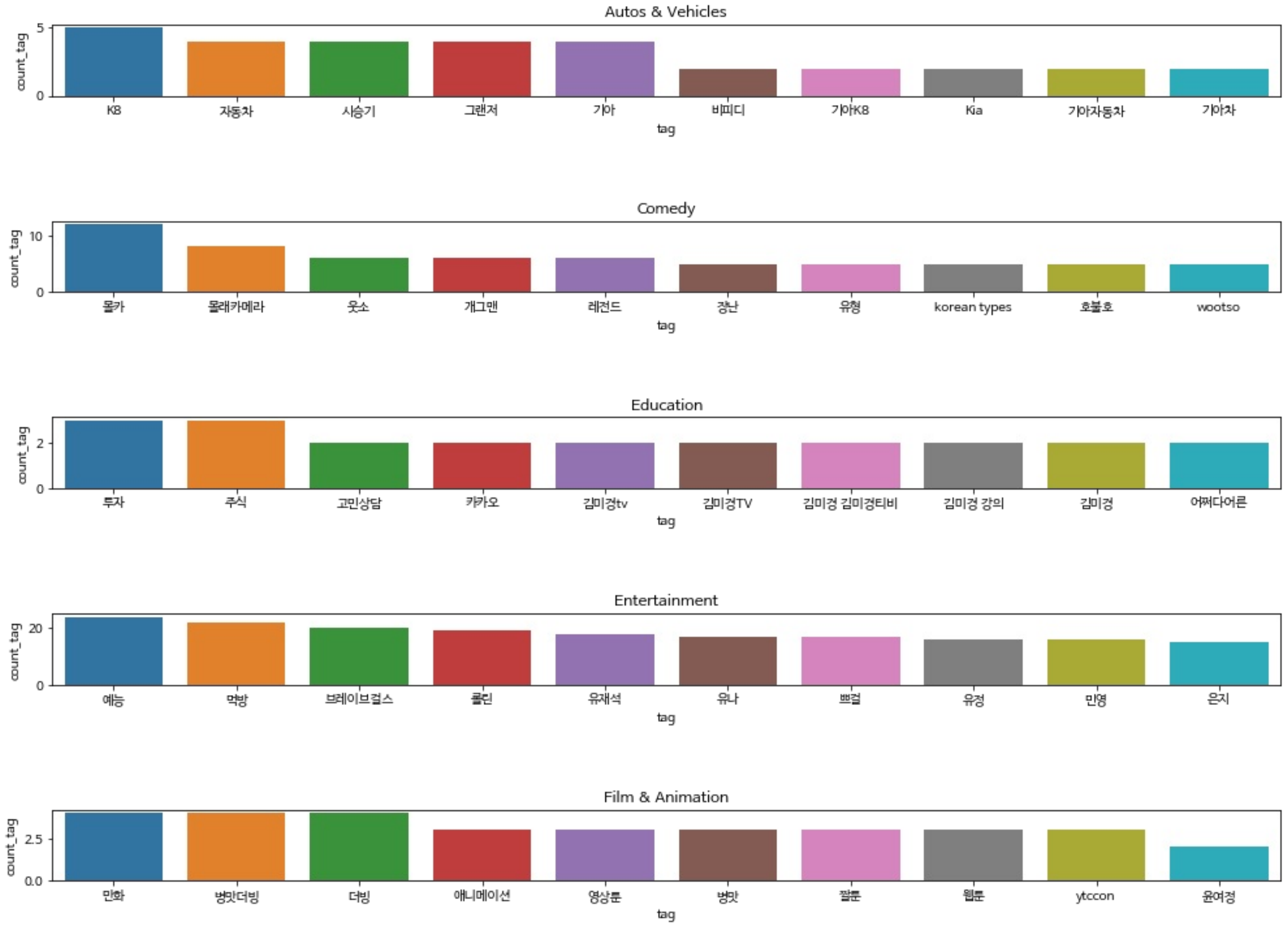


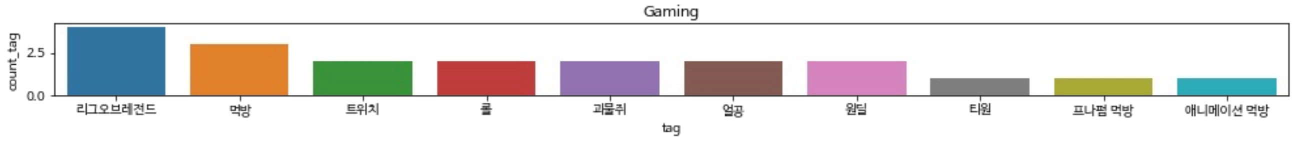


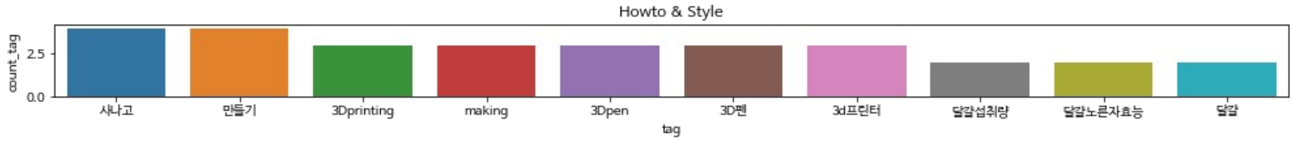
April

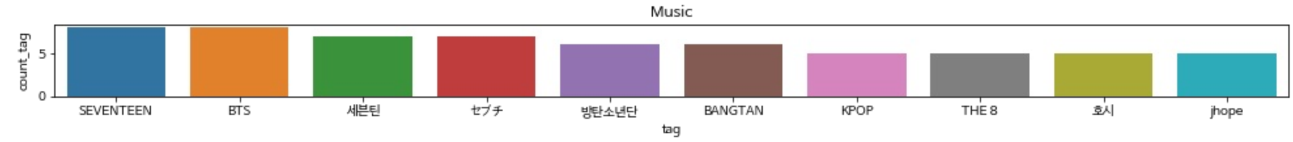
In [45]:

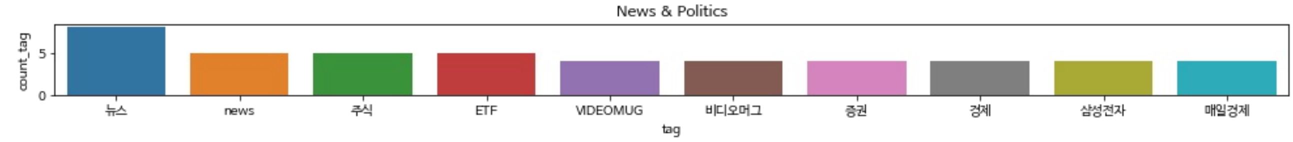
```
tag_rank_monthly(4)
```

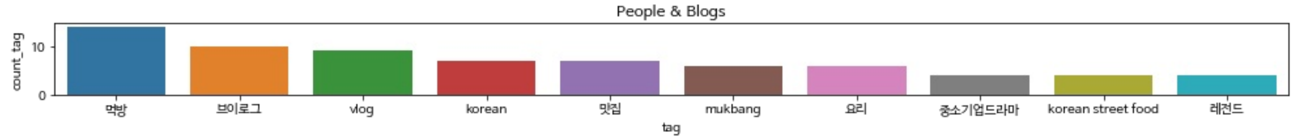


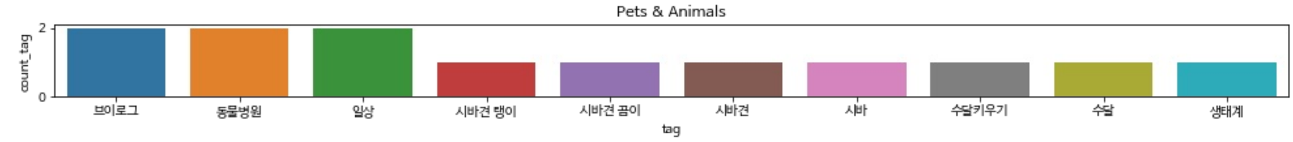

Gaming

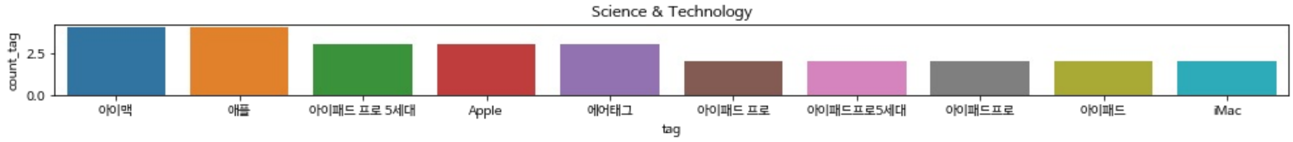

Howto & Style

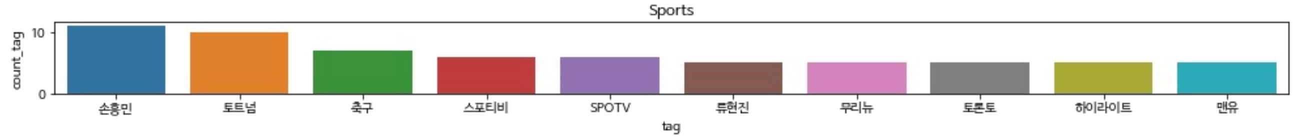

Music

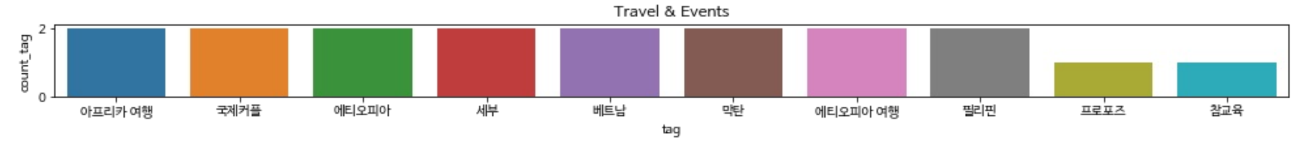

News & Politics


People & Blogs


Pets & Animals


Science & Technology

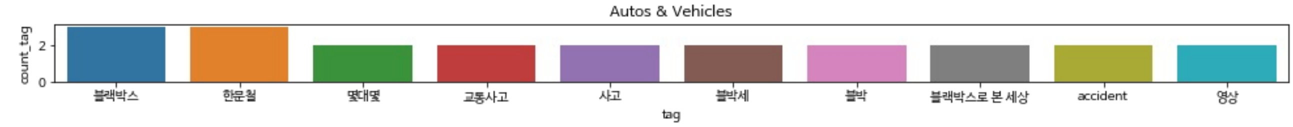

Sports

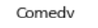

Travel & Events

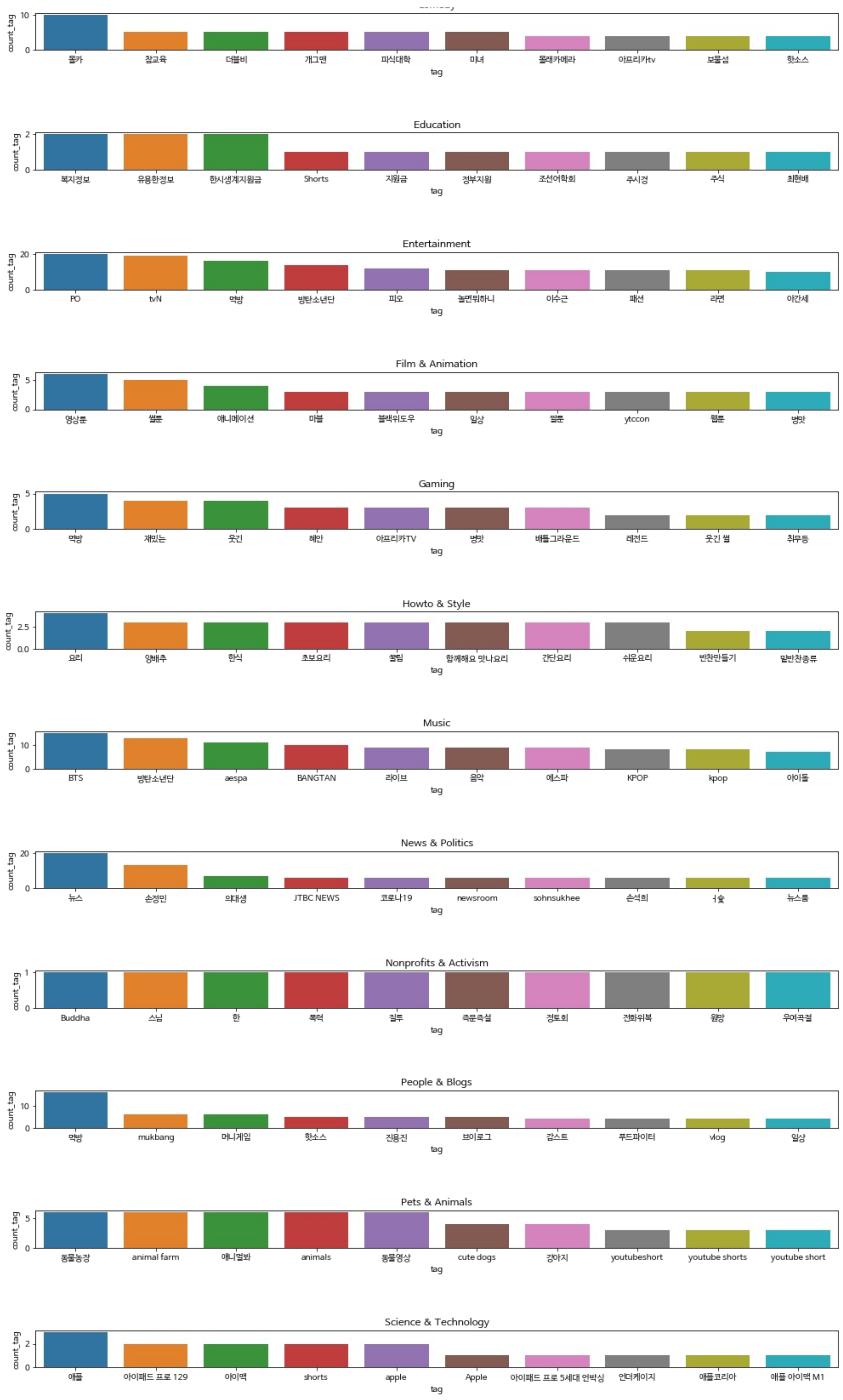
May

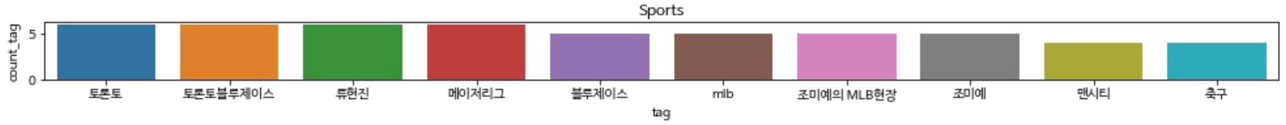
In [46]:

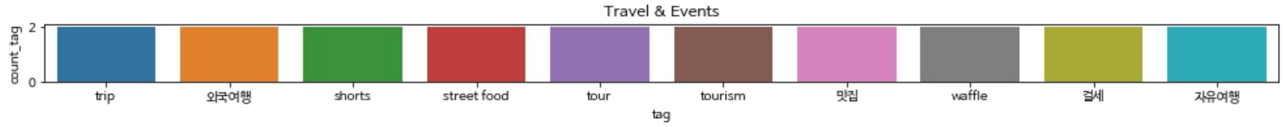
```
tag_rank_monthly(5)
```


Autos & Vehicles


Comedy


count_tag
tag
10
0
몰카
참교육
더블비
개그맨
피식대학
미녀
몰래카메라
아프리카tv
보물섬
핫소스
Education
2
0
복지정보
유용한정보
한시생계지원금
Shorts
지원금
정부지원
조선어학회
주시경
주식
최현배
Entertainment
20
0
PO
tvN
먹방
방탄소년단
피오
놀면뭐하니
이수근
패션
라면
야간세
Film & Animation
5
0
영상툰
썰툰
애니메이션
마블
블랙위도우
일상
짤툰
ytccon
웹툰
병맛
Gaming
5
0
먹방
재밌는
웃긴
헤안
아프리카TV
병맛
배틀그라운드
레전드
웃긴 썰
취두등
Howto & Style
2.5
0.0
요리
양배추
한식
초보요리
꿀팁
함께해요 맛나요리
간단요리
쉬운요리
반찬만들기
밑반찬종류
Music
10
0
BTS
방탄소년단
aespa
BANGTAN
라이브
음악
에스파
KPOP
kpop
아이돌
News & Politics
20
0
뉴스
손정민
의대생
JTBC NEWS
코로나19
newsroom
sohnsukhee
손석희
ㅓ울
뉴스룸
Nonprofits & Activism
1
0
Buddha
스님
한
폭력
질투
즉문즉설
정토회
전화위복
원망
우여곡절
People & Blogs
10
0
먹방
mukbang
머니게임
핫소스
진용진
브이로그
감스트
푸드파이터
vlog
일상
Pets & Animals
5
0
동물농장
animal farm
애니멀봐
animals
동물영상
cute dogs
강아지
youtubeshort
youtube shorts
youtube short
Science & Technology
2
0
애플
아이패드 프로 129
아이맥
shorts
apple
Apple
아이패드 프로 5세대 언박싱
언더케이지
애플코리아
애플 아이맥 M1

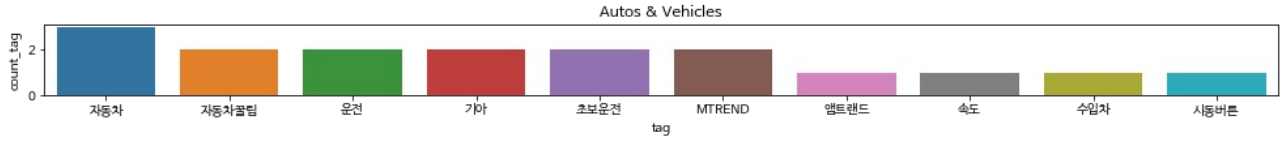


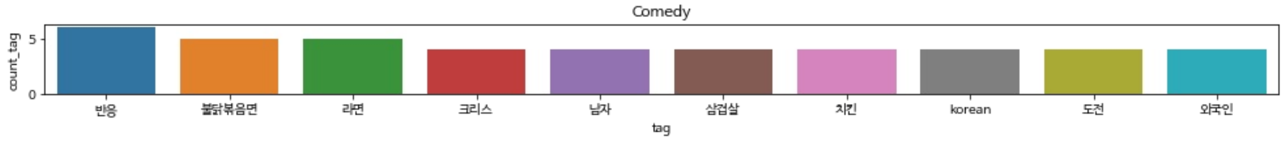


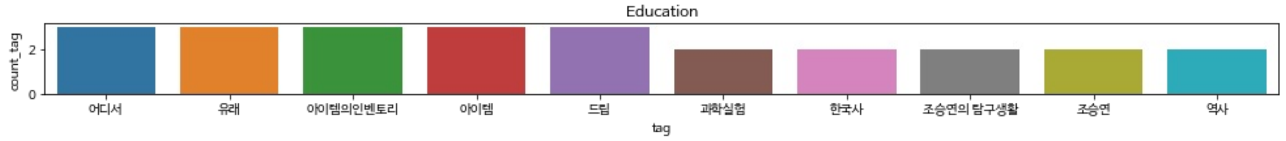
June

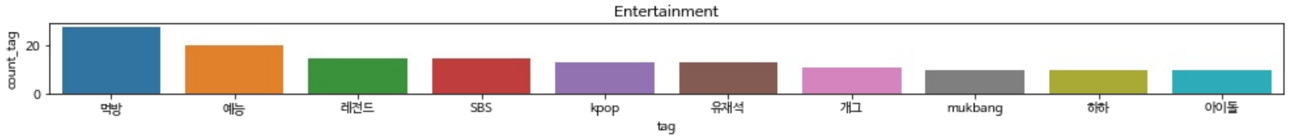
In [47]:

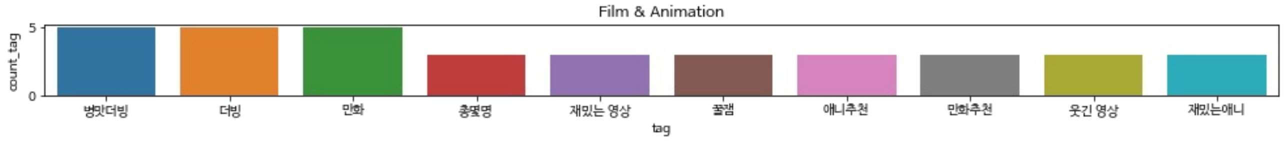
```
tag_rank_monthly(6)
```

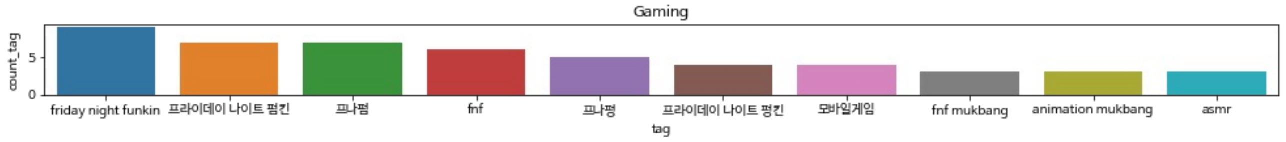


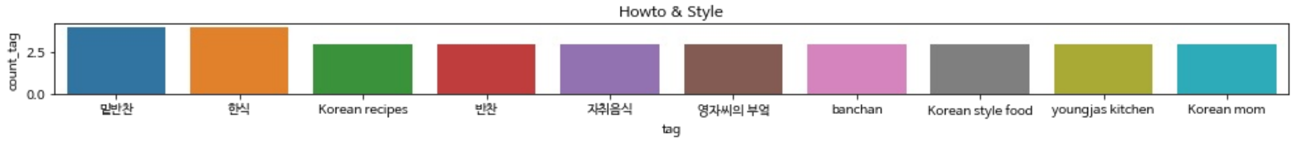


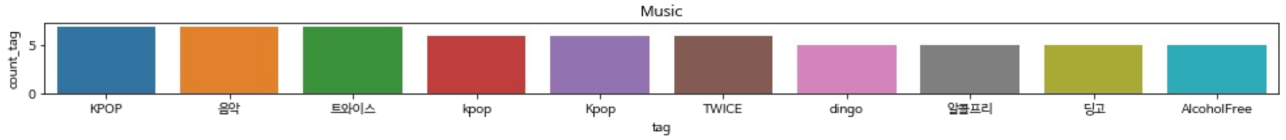


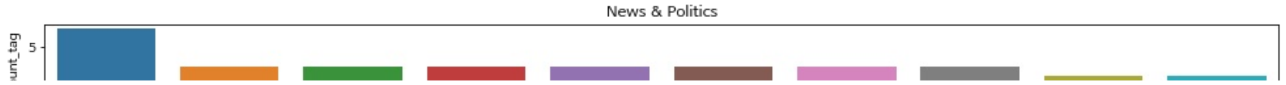


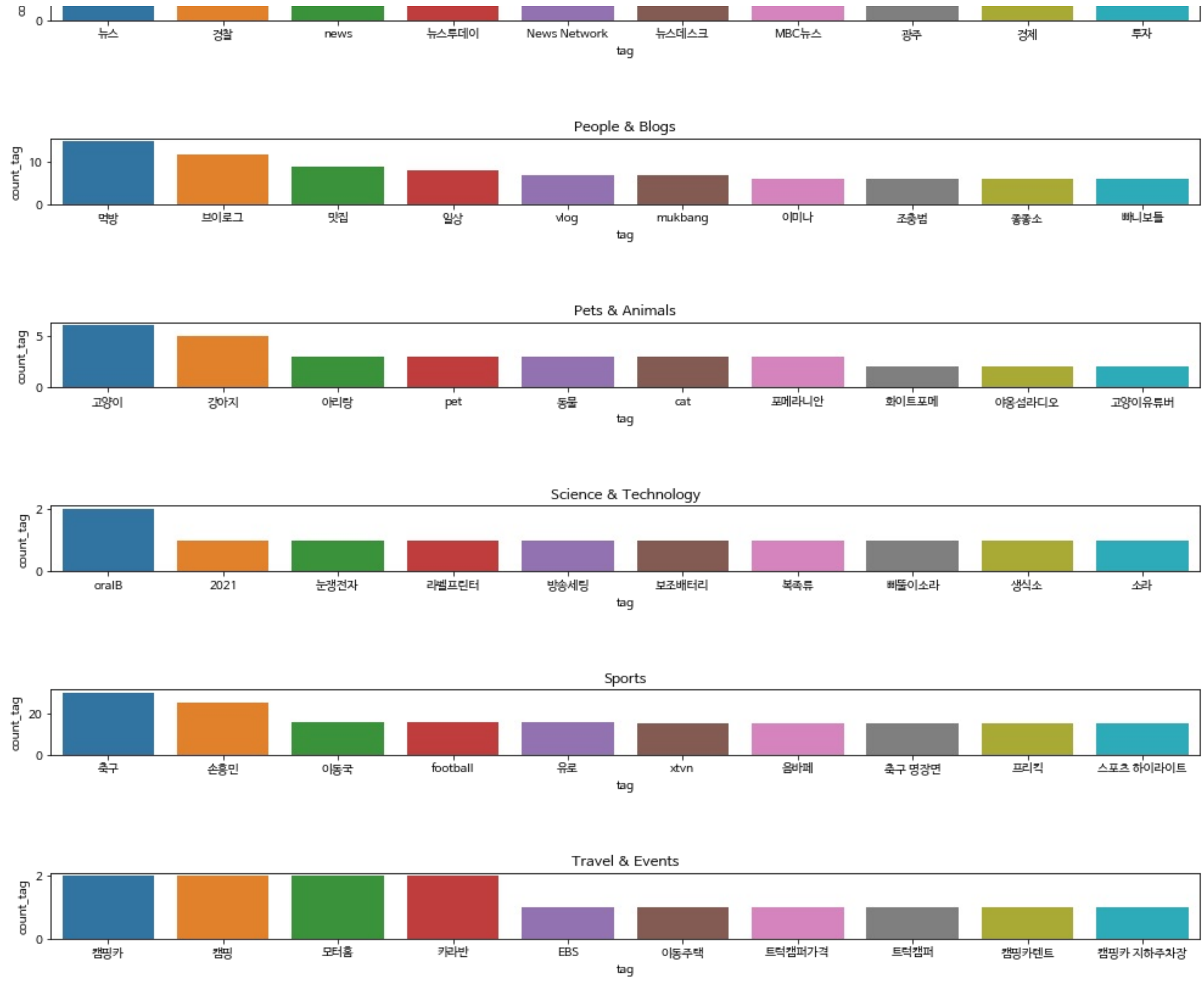








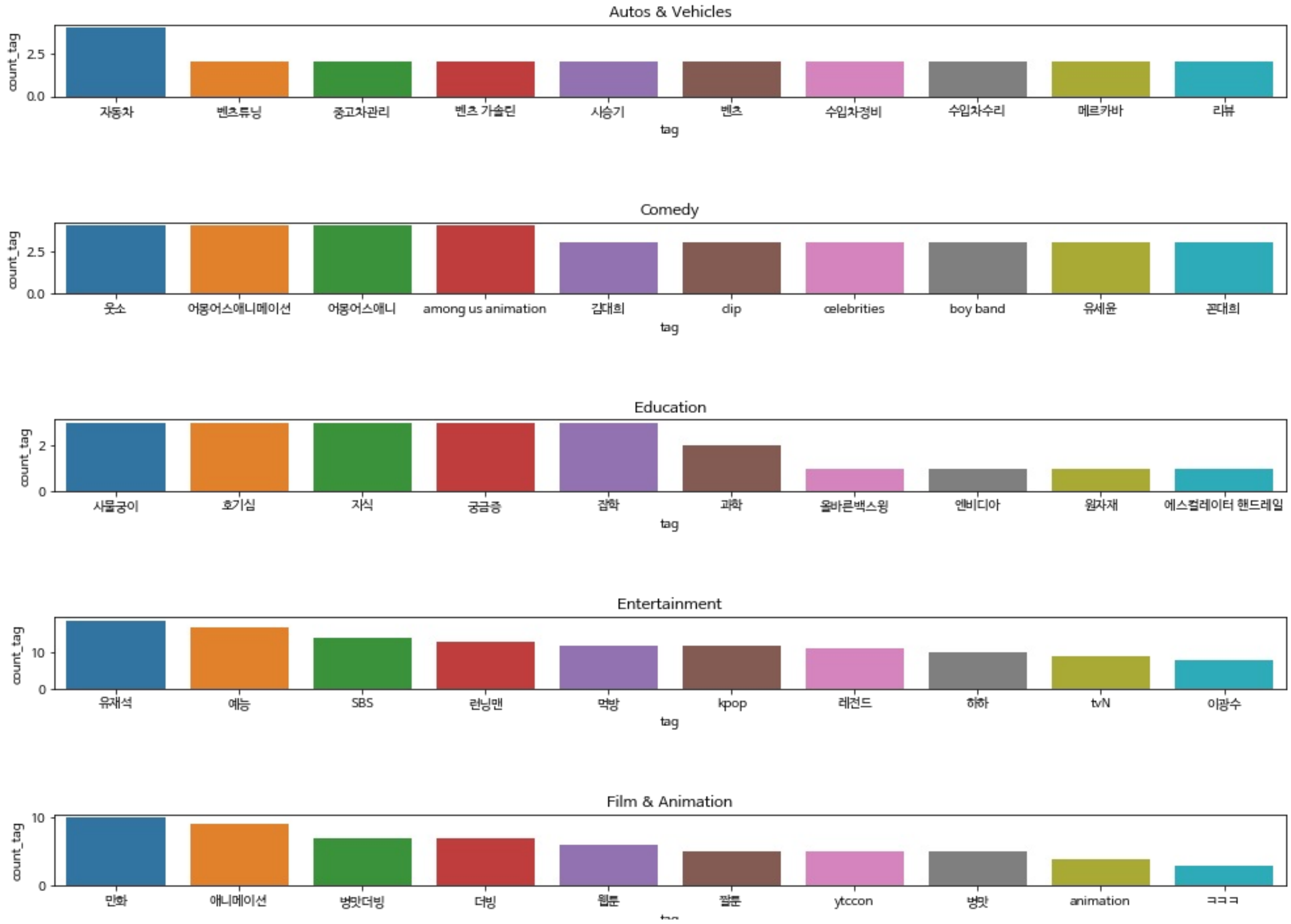


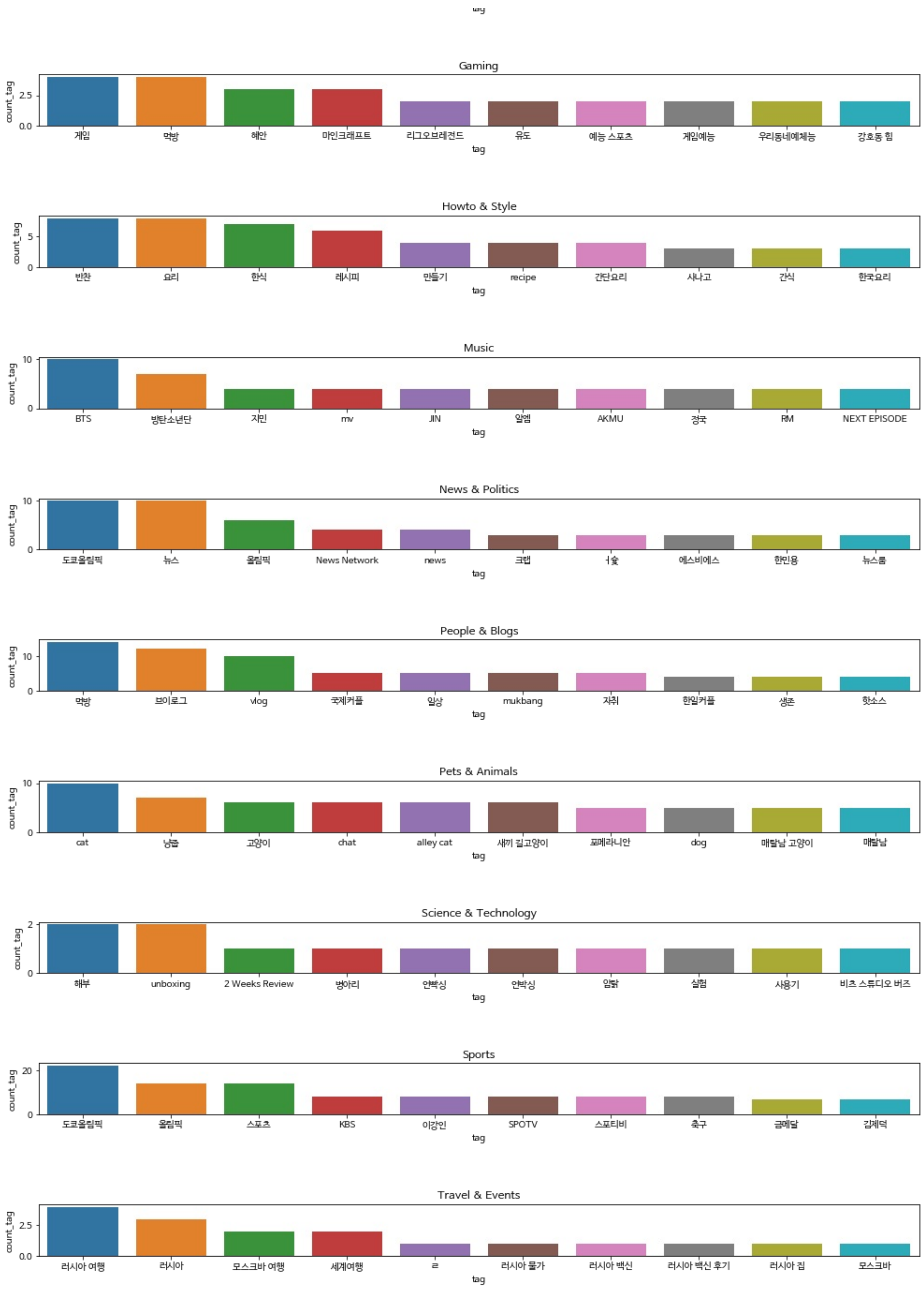


July

In [48]:

```
tag_rank_monthly(7)
```





Q2. 각각의 비디오는 시청자의 호응도(**engagement**)를 판단할수 있는 객관적인 지표들이 있음

ex) views, likes, dislikes, comments,...
비디오를 인기 동영상 기준에 부합하도록 분류할수 있는 새로운 지표를 개발하고 이 지표를 사용하여 engagement 와 어떤 상관관계가 있는지 설명하시오.

What determines if a video is ranked on Trending

- View count
- How quickly the video is generating views (i.e. “temperature”)
- Where views are coming from, including outside of YouTube
- The age of the video
- How the video performs compared to other recent uploads from the same channel

Analysis of objective indicator (views, likes, dislikes, comments...)

In [49]:

```
data[['on_views', 'on_likes', 'on_dislikes', 'on_comments']].describe().apply(lambda x:x.apply("{0:.5f}".format))
```

Out[49]:

	on_views	on_likes	on_dislikes	on_comments
count	2644.00000	2644.00000	2644.00000	2644.00000
mean	953481.91188	54658.12557	534.27988	9032.08548
std	3200374.58035	323933.71177	2070.64349	115532.13114
min	0.00000	0.00000	0.00000	0.00000
25%	235352.00000	4208.50000	86.00000	511.00000
50%	427924.50000	7851.00000	159.00000	1157.00000
75%	826880.00000	16883.50000	315.00000	2720.00000
max	97276666.00000	8097173.00000	37349.00000	4625133.00000

On average, if it exceeds 900,000 views, it will be on the trending video.

The average period of time it took to be trending video

In [50]:

```
(data['on_trending_date'] - data['published_date']).describe()
```

Out[50]:

```
count                          2644
mean     2 days 01:31:29.863842662
std      0 days 11:46:00.709350644
min                 1 days 00:00:00
25%                 2 days 00:00:00
50%                 2 days 00:00:00
75%                 2 days 00:00:00
max                 6 days 00:00:00
dtype: object
```

The Fastest video is on trending in just one day, but on average, it takes two days.

Change when there are several popular videos on the same channel

(Based on the channel with the highest number of videos per month)

In [51]:

```
data_date_index = data.set_index('published_date')
data_date_index = data_date_index.groupby([data_datetime_index.index.month, 'channel_id'])['video_id'].count().so
rt_values(ascending=False).to_frame().reset_index()
data_date_index.iloc[0]
```

Out[51]:

```
published_date          6
channel_id        CHtybqq
video_id               15
Name: 0, dtype: object
```

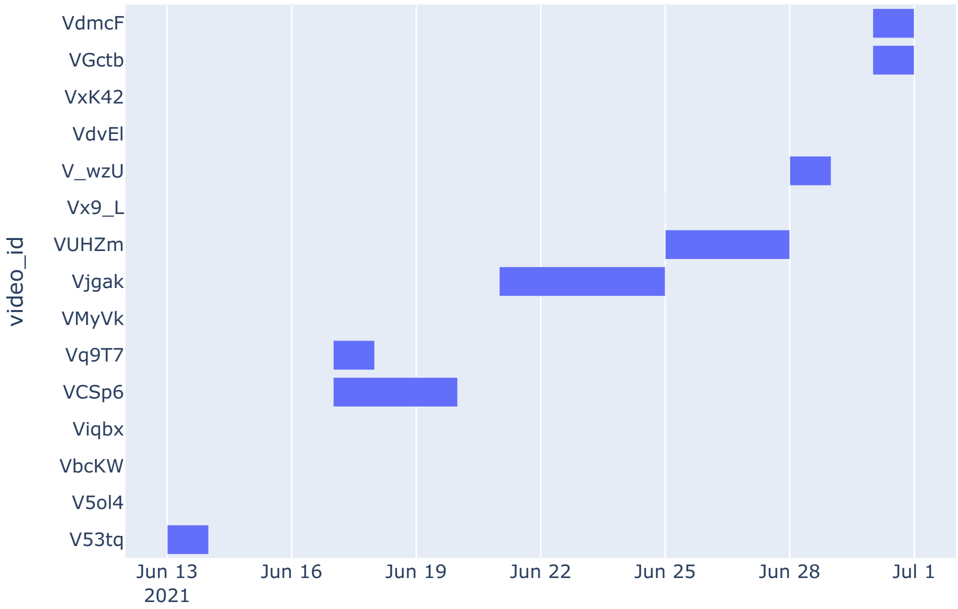
The data to be checked is a channel in which a total of 15 videos were included in trending videos during June.

In [52]:

```
temp = data.loc[(data_datetime_index.index.month==data_date_index.iloc[0].published_date)&(data['channel_id']==da
ta_date_index.iloc[0].channel_id)].sort_values('on_trending_date')
```

In [53]:

```
import plotly.express as px
fig = px.timeline(temp, x_start="on_trending_date", x_end="off_trending_date", y="video_id")
fig.show(renderer="svg")
```



In one channel, it was expected that the period in the trending video would not overlap, but in some cases, it was not.

Analysis of the characteristics of videos that have been in trending videos for a long time

(100 videos that have a long period of time in trending videos)

In [54]:

```
import datetime
data_gap = data
data_gap['gap_trending_date']=(data['off_trending_date']-data['on_trending_date']).dt.days
data_gap = data_gap.sort_values('gap_trending_date', ascending=False)[:100]
```

In [55]:

```
temp = data_gap.groupby(['category_name'])['gap_trending_date'].agg(['mean', 'max'])
temp.sort_values('max', ascending=False)
```

Out[55]:

```
                     mean   max
category_name
            Music  4.888889   7
           Comedy  4.250000   5
        Education  4.333333   5
    Entertainment  4.205882   5
 Film & Animation  4.166667   5
  News & Politics  4.200000   5
   People & Blogs  4.250000   5
   Pets & Animals  4.500000   5
           Sports  4.333333   5
           Gaming  4.000000   4
    Howto & Style  4.000000   4
  Travel & Events  4.000000   4
```

Music is the longest, but on average, Comedy is the longest.

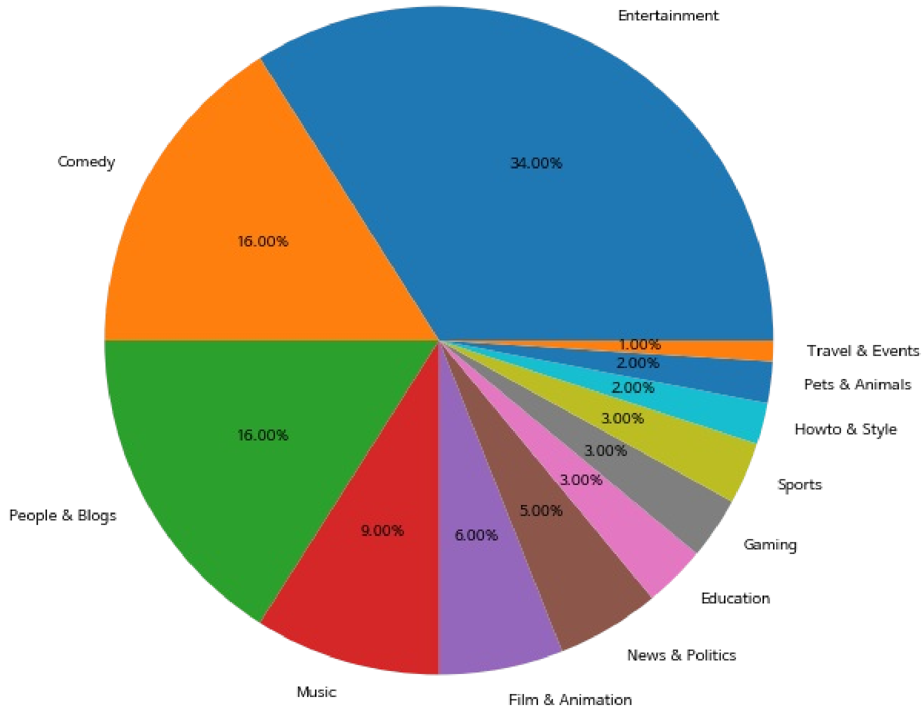
Category distribution

The number of categories of videos that have been on trending video for a long time.

In [56]:

```
fig,ax = plt.subplots(figsize=(10,10))

temp = data_gap.groupby('category_name')['channel_id'].count().to_frame()
temp = temp.sort_values('channel_id', ascending=False)
ax.pie(temp,
        labels=temp.index,
        autopct='%1.2f%%')

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: MatplotlibDeprecationWarning:

Non-1D inputs to pie() are currently squeeze()d, but this behavior is deprecated since 3.1 and will
be removed in 3.3; pass a 1D array instead.
```



Entertainment > Comedy > People & Blogs ...

In [57]:

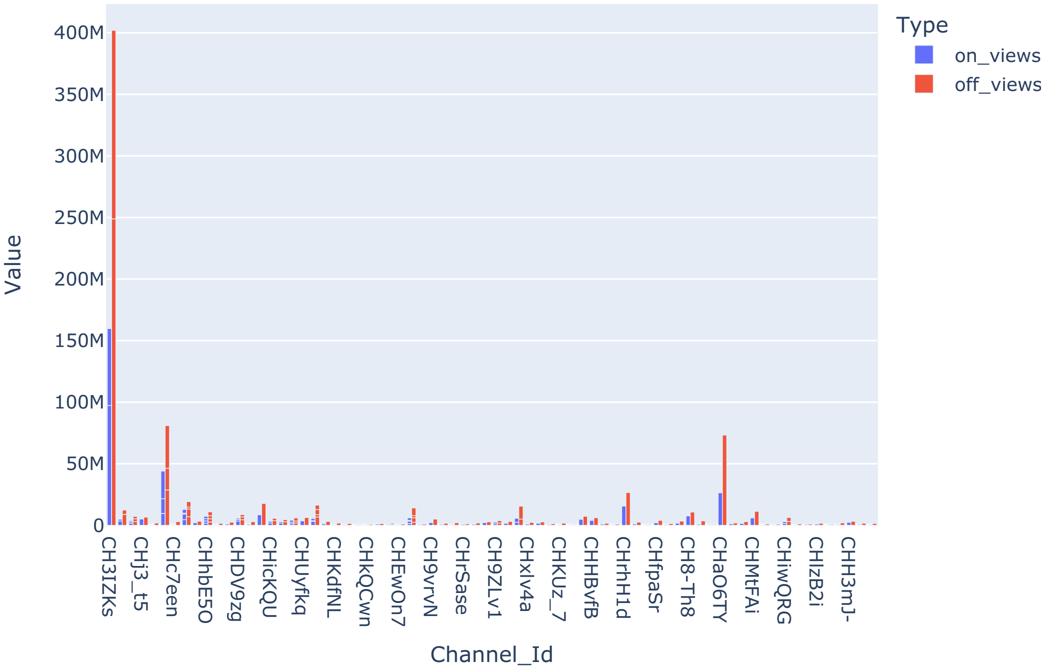
```
def gap_visualization(df, start_column, end_column):
    temp = data_gap[['channel_id', start_column, end_column]].melt(id_vars='channel_id', var_name='type', value_n
ame='value').rename(columns=str.title)
    fig = px.bar(temp, x='Channel_Id', y='Value',barmode="group", color="Type")
    fig.show(renderer="svg")
```

The difference between on trending and off trending.

Difference views

In [58]:

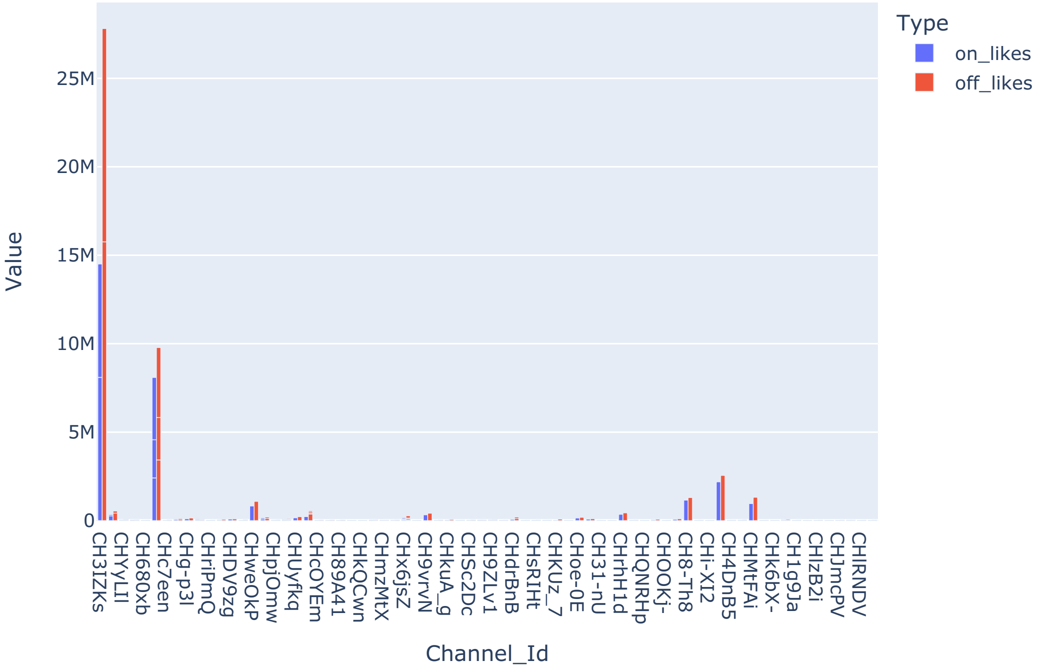
```
gap_visualization(data_gap, 'on_views', 'off_views')
```



Difference likes

In [59]:

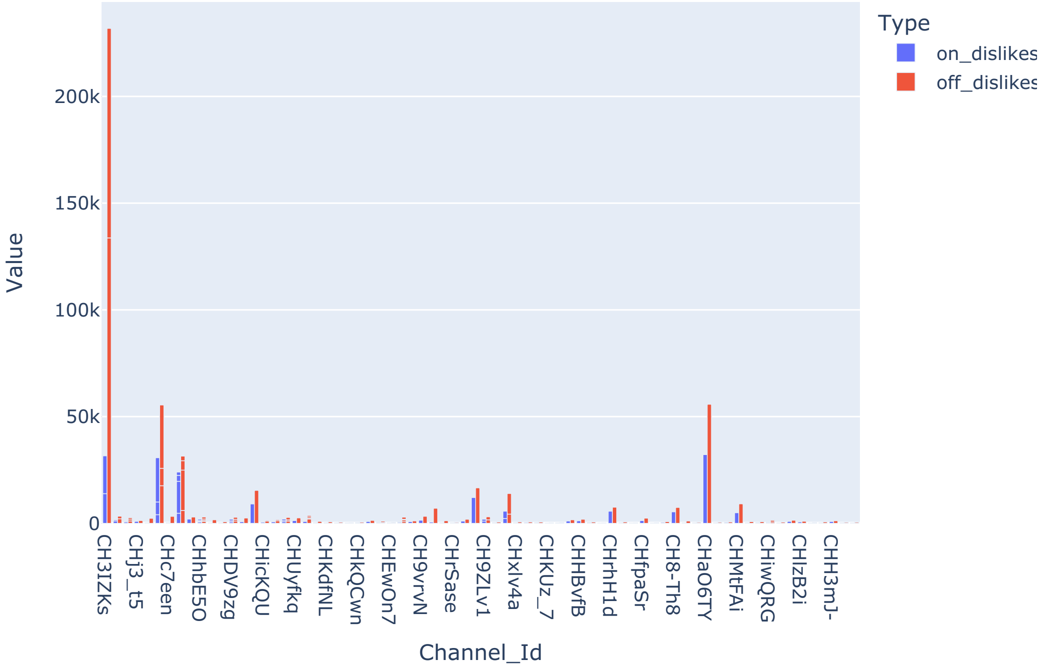
```
gap_visualization(data_gap, 'on_likes', 'off_likes')
```



Difference dislikes

In [60]:

```
gap_visualization(data_gap, 'on_dislikes', 'off_dislikes')
```



There are many factors influencing trending videos, but an additional helpful indicator is how long they have stayed in trending videos. I expect that the long period of stay on trending videios will have a large difference between indicators (e.g. views, likes, comments).