In [24]:
```
from googleapiclient.discovery import build
import pandas as pd
import googleapiclient.discovery
from IPython.display import JSON
import itertools

import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
%matplotlib inline
import seaborn as sb
import imageio
import isodate

#NLP
from wordcloud import WordCloud, STOPWORDS
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

In [2]:
```
api_key = ['']
```

In [3]:
```
channel_names = [
    "Alex The Analyst",
    "Corey Schafer",
    "Ken Jee",
    "Mo Chen",
    "Luke Barousse",
    "Data Professor",
    "Tech With Tim",
    "Data Science Jay",
    "Nicholas Renotte",
    "StatQuest with Josh Starmer"
]

# Build YouTube API service
youtube = build('youtube', 'v3', developerKey=api_key)

# Dictionary to store channel names and their IDs
channel_ids = {}

for channel_name in channel_names:
    # Search for the channel
    request = youtube.search().list(
        part='snippet',
        q=channel_name,
        type='channel',
        maxResults=1
    )
    response = request.execute()

    # Extract and store the channel ID
    if response['items']:
        channel_id = response['items'][0]['id']['channelId']
        channel_ids[channel_name] = channel_id
        print(f"Channel: {channel_name} | ID: {channel_id}")
    else:
        print(f"Channel: {channel_name} not found.")

# Print all channel IDs
print(channel_ids)
```

```
Channel: Alex The Analyst | ID: UC7cs8q-gJRlGwj4A8OmCmXg
Channel: Corey Schafer | ID: UCCezIgC97PvUuR4_gbFUs5g
Channel: Ken Jee | ID: UCiT9RITQ9PW6BhXK0y2jaeg
Channel: Mo Chen | ID: UCDybamfye5An6p-j1t2YMsg
Channel: Luke Barousse | ID: UCLLw7jmFsvfIVaUFsLs8mlQ
Channel: Data Professor | ID: UCV8e2g4IWQqK71bbzGDEI4Q
Channel: Tech With Tim | ID: UC4JX40jDee_tINbkjycV4Sg
Channel: Data Science Jay | ID: UCcQx1UnmorvmSEZef4X7-6g
Channel: Nicholas Renotte | ID: UCHXa4OpASJEwrHrLeIzw7Yg
Channel: StatQuest with Josh Starmer | ID: UCtYLUTtgS3k1Fg4y5tAhLbw
{'Alex The Analyst': 'UC7cs8q-gJRlGwj4A8OmCmXg', 'Corey Schafer': 'UCCezIgC97PvUuR
4_gbFUs5g', 'Ken Jee': 'UCiT9RITQ9PW6BhXK0y2jaeg', 'Mo Chen': 'UCDybamfye5An6p-j1t
2YMsg', 'Luke Barousse': 'UCLLw7jmFsvfIVaUFsLs8mlQ', 'Data Professor': 'UCV8e2g4IW
QqK71bbzGDEI4Q', 'Tech With Tim': 'UC4JX40jDee_tINbkjycV4Sg', 'Data Science Jay':
'UCcQx1UnmorvmSEZef4X7-6g', 'Nicholas Renotte': 'UCHXa4OpASJEwrHrLeIzw7Yg', 'StatQ
uest with Josh Starmer': 'UCtYLUTtgS3k1Fg4y5tAhLbw'}
```

In [4]:
```
youtube_channel_ids = [
    "UC7cs8q-gJRlGwj4A8OmCmXg",
    "UCCezIgC97PvUuR4_gbFUs5g",
    "UCiT9RITQ9PW6BhXK0y2jaeg",
    "UCDybamfye5An6p-j1t2YMsg",
    "UCLLw7jmFsvfIVaUFsLs8mlQ",
    "UCV8e2g4IWQqK71bbzGDEI4Q",
    "UC4JX40jDee_tINbkjycV4Sg",
    "UCcQx1UnmorvmSEZef4X7-6g",
    "UCHXa4OpASJEwrHrLeIzw7Yg",
    "UCtYLUTtgS3k1Fg4y5tAhLbw"
]

print(youtube_channel_ids)
```

```
['UC7cs8q-gJRlGwj4A8OmCmXg', 'UCCezIgC97PvUuR4_gbFUs5g', 'UCiT9RITQ9PW6BhXK0y2jae
g', 'UCDybamfye5An6p-j1t2YMsg', 'UCLLw7jmFsvfIVaUFsLs8mlQ', 'UCV8e2g4IWQqK71bbzGDE
I4Q', 'UC4JX40jDee_tINbkjycV4Sg', 'UCcQx1UnmorvmSEZef4X7-6g', 'UCHXa4OpASJEwrHrLeI
zw7Yg', 'UCtYLUTtgS3k1Fg4y5tAhLbw']
```

In [5]:
```
api_service_name = "youtube"
api_version = "v3"
youtube = googleapiclient.discovery.build(
    api_service_name, api_version, developerKey=api_key)
```

In [6]:
```
def get_channel_stats(youtube, channel_ids):
    """
    Function: Gather interested channel stats from youtube creator's channel page

    INPUT:
    youtube - build object from googleapiclient.discovery
    channel_ids - (list) list of channel ids to be analyzed

    OUTPUT:
    all_data - (pandas dataframe) dataframe that consists of the following columns:
    """
    all_data = []

    request = youtube.channels().list(
        part="snippet,contentDetails,statistics",
        id=','.join(channel_ids)
    )
    response = request.execute()

    #loop through items
    for item in response['items']:
        data = {'channelName': item['snippet']['title'],
                'publishDate': item['snippet']['publishedAt'],
                'subscribers': item['statistics']['subscriberCount'],
                'views': item['statistics']['viewCount'],
                'totalVideos': item['statistics']['videoCount'],
                'playlistId': item['contentDetails']['relatedPlaylists']['uploads']
        }
        all_data.append(data)
    all_data = pd.DataFrame(all_data)

    return(all_data)
```

In [7]:
```
def get_videos_ids(youtube, playlist_id):
    """
    Function: Gather videoIds from channel.

    INPUT:
    youtube - Get credentials and create an API client/Initialise a Youtube API ser
    playlist_ids - (list) list of playlist ids to be analyzed.

    OUTPUT:
    video_ids - (list) list of dictionary that contains all videoId for channel.
    """
    video_ids = []

    request = youtube.playlistItems().list(
        part="snippet, contentDetails",
        playlistId= playlist_id,
        maxResults = 50
    )

    response = request.execute()

    for item in response['items']:
        data = {
                'videoId': item['contentDetails']['videoId']
               }
        video_ids.append(data)

    next_page_token = response.get('nextPageToken')
    while next_page_token is not None:
        request = youtube.playlistItems().list(
            part="snippet, contentDetails",
            playlistId= playlist_id,
            maxResults = 50,
            pageToken = next_page_token
            )
        response = request.execute()

        for item in response['items']:
            data = {
                    'videoId': item['contentDetails']['videoId']
               }
            video_ids.append(data)
        next_page_token = response.get('nextPageToken')

    return video_ids
```

In [8]:
```
def get_video_details(youtube, video_ids):
    """
    Function: Gather interested information from videos and store in dataframe.

    INPUT:
    youtube - Get credentials and create an API client/Initialise a Youtube API ser
    video_ids - (list) list of video ids.

    OUTPUT:
    video_df - (pandas dataframe) dataframe of video statistics. Includes columns:
                 'channelTitle', 'title', 'description', 'tags', 'publishedAt',
                 'viewCount', 'likeCount', 'favouriteCount', 'commentCount',
                 'duration', 'definition', 'caption'%
    """
    all_video_info = []
    for i in range(0,len(video_ids), 50):
        request = youtube.videos().list(
            part="snippet,contentDetails,statistics",
            id= ','.join(video_ids[i:i+50])
        )
        response = request.execute()

        for video in response['items']:
            # create dictionary of stats I want to keep
            stats_keep = {'snippet': ['channelTitle', 'title', 'description', 'tags
                          'statistics': ['viewCount', 'likeCount', 'favouriteCount'
                          'contentDetails': ['duration', 'definition', 'caption']
                         }

            # empty dictionary to keep track of keys and values
            video_info = {}
            video_info['video_id'] = video['id']

            # extract values and append them into empty dictionary
            for k in stats_keep.keys():
                for v in stats_keep[k]:
                    try:
                        video_info[v] = video[k][v]
                    except:
                        video_info[v] = None

            all_video_info.append(video_info)
        video_df = pd.DataFrame(all_video_info)
    return video_df
```

In [9]:
```
def get_videos_comments(youtube, video_ids):
    """
    Function: Gather comments from videos and store in dataframe.

    INPUT:
    youtube - Get credentials and create an API client/Initialise a Youtube API ser
    video_ids - (list) list of video ids.

    OUTPUT:
    all_comments_df - (pandas dataframe) dataframe of comments. Each video has a ma
                        10 comments which are compiled in a list.
    """
    all_comments = []

    for video_id in video_ids:
        try:
            request = youtube.commentThreads().list(
                part="snippet,replies",
                videoId=video_id
            )
            response = request.execute()
            # help https://developers.google.com/youtube/v3/docs/commentThreads?hl=
            video_comments = [comment['snippet']['topLevelComment']['snippet']['tex
                              for comment in response['items'][0:10]]
            video_comments_info = {'video_id': video_id, 'comments': video_comments
            all_comments.append(video_comments_info)
        except:
            # dealing with errors
            pass

    all_comments_df = pd.DataFrame(all_comments)
    return all_comments_df
```

In [10]:
```
channel_stats = get_channel_stats(youtube, youtube_channel_ids)
channel_stats
```

Out[10]:

	channelName	publishDate	subscribers	views	totalVideos	playlistI
0	StatQuest with Josh Starmer	2011-05-24T01:52:48Z	1330000	74805212	285	UUtYLUTtgS3k1Fg4y5tAhLb
1	Alex The Analyst	2020-01-08T05:04:24.970712Z	973000	45262286	345	UU7cs8q-gJRlGwj4A8OmCmX
2	Luke Barousse	2020-08-03T09:02:41.213077Z	498000	24854675	163	UULLw7jmFsvfIVaUFsLs8ml
3	Mo Chen	2022-12-25T20:25:38.187653Z	148000	5642948	215	UUDybamfye5An6p-j1t2YMs
4	Data Professor	2019-08-17T15:59:56Z	200000	7160298	353	UUV8e2g4IWQqK71bbzGDEI4
5	Jay Feng	2019-11-19T19:16:30.516571Z	52000	3487091	420	UUcQx1UnmorvmSEZef4X7-6
6	Corey Schafer	2006-05-31T22:49:22Z	1400000	100442533	239	UUCezIgC97PvUuR4_gbFUs5
7	Ken Jee	2014-02-28T14:58:24Z	266000	9309046	288	UUiT9RITQ9PW6BhXK0y2jae
8	Tech With Tim	2014-04-23T01:57:10Z	1670000	163496137	1314	UU4JX40jDee_tINbkjycV4S
9	Nicholas Renotte	2019-01-26T22:31:46Z	295000	20586312	308	UUHXa4OpASJEwrHrLeIzw7Y

In [11]:
```
playlist_ids = list(channel_stats.playlistId.unique()) # convert all unique calues
video_ids_list = []

# loop to get video ids from all interested channels
for playlist_id in playlist_ids:
    video_ids = get_videos_ids(youtube, playlist_id)
    video_ids_list.append(video_ids)
video_ids_list
```

```
  {'videoId': 'QdXF69-EGEI'},
  {'videoId': 'ZTt9gsGcdDo'},
  {'videoId': 'Qf06XDYXCXI'},
  {'videoId': 'rC9vw2dSpQo'},
  {'videoId': 'Ka04Dj7DxGk'},
  {'videoId': 'bQ5BoolX9Ag'},
  {'videoId': 'zxQyTK8quyY'},
  {'videoId': '8ZcccMzTz7Y'},
  {'videoId': 'YaQEUgIr4Mk'},
  {'videoId': 'PSs6nxngL6k'},
  {'videoId': '953NHzFtGHc'},
  {'videoId': '02zO75hHpZQ'},
  {'videoId': 'L8HKweZIOmg'},
  {'videoId': 'y8xRw76i1qY'},
  {'videoId': 'LS6VX7noVWY'},
  {'videoId': 'ZYDN25N5WhQ'},
  {'videoId': 'ccjrsxXmfnw'},
  {'videoId': 'bv9agba7blc'},
  {'videoId': 'oZ9SrkF_-LE'},
```

In [12]:
```
# chain all lists into one giant list
video_ids_list_clean= list(itertools.chain(*video_ids_list))
# only get video id value(str) and put into list
video_ids_list_clean = [d['videoId'] for d in video_ids_list_clean]
```

In [13]:
```
video_ids = video_ids_list_clean
video_df = get_video_details(youtube, video_ids)
video_df
```

Out[13]:

e	title	description	tags	publishedAt	viewCount	likeCount	favou
t ı r	Encoder-Only Transformers (like BERT) for RAG,...	Encoder-Only Transformers are the backbone for...	[Josh Starmer, StatQuest, Machine Learning, BE...	2024-11-18T05:00:11Z	24006	864	
t ı r	Luis Serrano + Josh Starmer Q&A Livestream!!!	Join me, Luis Serrano http://www.youtube.com/c...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-10-10T04:04:08Z	4892	113	
t ı r	Human Stories in AI: Nana Janashia@TechWorld W...	In this episode we have special guest Nana Jan...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-09-09T04:00:17Z	6353	127	
t ı r	A few more lessons from my Pop!	Since September 4th is Global Frank Starmer Da...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-09-04T04:00:02Z	6770	306	
t ı r	Human Stories in AI: Abbas Merchant@Matics Ana...	In this episode we have special guest Abbas Me...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-07-29T04:00:38Z	6368	132	
.	...	...	...	...	...	...	
s e	Generating Credentials - Build An Image Classi...	Tired of struggling to build an image classifi...	[image classification, python, ibm, visual rec...	2019-01-29T21:29:54Z	1624	16	
s e	Installing Watson Developer Cloud - Build An I...	Tired of struggling to build an image classifi...	[visual recognition, image classification, pyt...	2019-01-29T21:29:50Z	1502	12	
s e	General Image Classification - Build An Image ...	Tired of struggling to build an image classifi...	[watson, ibm, visual recognition, image classi...	2019-01-29T21:29:47Z	2006	17	
s e	Food Image Classification - Build An Image Cla...	Tired of struggling to build an image classifi...	[python, image classification, watson, visual ...	2019-01-29T21:29:44Z	2831	19	
s e	Face Detection - Build An Image Classifier wit...	Tired of struggling to build an image classifi...	[python, ibm, watson, image classification, vi...	2019-01-29T21:29:41Z	4115	38	

In [14]:
```
all_comments_df = get_videos_comments(youtube, video_ids)
all_comments_df
```

Out[14]:

	video_id	comments
0	GDN649X_acE	[Support StatQuest by buying my books The Stat...
1	qJrmQe8TOTw	[Josh is so humble, but a genius :). Thanks so...
2	DkmfIQRDyXc	[Amazing job!, never stop making videos, or el...
3	0QOm7Sn5uwQ	[Support StatQuest by buying my book The StatQ...
4	wIGOnM6Cf_E	[Actually your clips are good, not only chasin...
...	...	...
3922	JjuCGJyZacE	[Hello,In IBM cloud I can't find visual recogn...
3923	PPq79Q51Ya4	[Hey bro im just getting started in these but ...
3924	oKDkwZkzX48	[I had some problems - looks like this might b...
3925	FgZsV09npbs	[Very helpful my friend. I WOULD kindly ask a ...
3926	z16aNdvgAog	[Sir please do make RASA playlist. Your teachi...

3927 rows × 2 columns

Data Pre-Processing

In [15]:
```
video_df.isnull().any()
```

Out[15]:
```
video_id          False
channelTitle      False
title             False
description       False
tags               True
publishedAt       False
viewCount         False
likeCount          True
favouriteCount     True
commentCount       True
duration          False
definition        False
caption           False
dtype: bool
```

In [16]:
```
video_df.dtypes
```
Out[16]:
```
video_id          object
channelTitle      object
title             object
description       object
tags              object
publishedAt       object
viewCount         object
likeCount         object
favouriteCount    object
commentCount      object
duration          object
definition        object
caption           object
dtype: object
```

In [17]:
```
video_df.describe()
```
Out[17]:

	video_id	channelTitle	title	description	tags	publishedAt	viewCount	likeCount
count	3936	3936	3936	3936	3541	3936	3936	3930
unique	3936	10	3926	3419	2779	3923	3814	2308
top	GDN649X_acE	Tech With Tim	Linear Regression, Clearly Explained!!!		[tech with tim]	2019-02-11T08:16:02Z	0	3
freq	1	1315	2	388	278	3	8	15

In [18]:
```
all_comments_df.isnull().sum()
```
Out[18]:
```
video_id    0
comments    0
dtype: int64
```

In [19]:
```
num_cols = ['viewCount','likeCount','favouriteCount', 'commentCount']
video_df[num_cols] = video_df[num_cols].apply(pd.to_numeric, errors = 'coerce', axi

#Check
video_df.dtypes
```

Out[19]:
```
video_id           object
channelTitle       object
title              object
description        object
tags               object
publishedAt        object
viewCount         float64
likeCount         float64
favouriteCount    float64
commentCount      float64
duration           object
definition         object
caption            object
dtype: object
```

```
In [20]: video_df
```

Out[20]:

	video_id	channelTitle	title	description	tags	publis
0	GDN649X_acE	StatQuest with Josh Starmer	Encoder-Only Transformers (like BERT) for RAG,...	Encoder-Only Transformers are the backbone for...	[Josh Starmer, StatQuest, Machine Learning, BE...	20 18T05:
1	qJrmQe8TOTw	StatQuest with Josh Starmer	Luis Serrano + Josh Starmer Q&A Livestream!!!	Join me, Luis Serrano http://www.youtube.com/c...	[Josh Starmer, StatQuest, Machine Learning, St...	20 10T04:
2	DkmflQRDyXc	StatQuest with Josh Starmer	Human Stories in AI: Nana Janashia@TechWorld W...	In this episode we have special guest Nana Jan...	[Josh Starmer, StatQuest, Machine Learning, St...	20 09T04:
3	0QOm7Sn5uwQ	StatQuest with Josh Starmer	A few more lessons from my Pop!	Since September 4th is Global Frank Starmer Da...	[Josh Starmer, StatQuest, Machine Learning, St...	20 04T04:
4	wIGOnM6Cf_E	StatQuest with Josh Starmer	Human Stories in AI: Abbas Merchant@Matics Ana...	In this episode we have special guest Abbas Me...	[Josh Starmer, StatQuest, Machine Learning, St...	20 29T04:
...	...	...	...	...	...	
3931	JjuCGJyZacE	Nicholas Renotte	Generating Credentials - Build An Image Classi...	Tired of struggling to build an image classifi...	[image classification, python, ibm, visual rec...	20 29T21:
3932	PPq79Q51Ya4	Nicholas Renotte	Installing Watson Developer Cloud - Build An I...	Tired of struggling to build an image classifi...	[visual recognition, image classification, pyt...	20 29T21:
3933	oKDkwZkzX48	Nicholas Renotte	General Image Classification - Build An Image ...	Tired of struggling to build an image classifi...	[watson, ibm, visual recognition, image classi...	20 29T21:
3934	FgZsV09npbs	Nicholas Renotte	Food Image Classification - Build An Image Cla...	Tired of struggling to build an image classifi...	[python, image classification, watson, visual ...	20 29T21:
3935	z16aNdvgAog	Nicholas Renotte	Face Detection - Build An Image Classifier wit...	Tired of struggling to build an image classifi...	[python, ibm, watson, image classification, vi...	20 29T21:

3936 rows × 13 columns

In [21]:
```
video_df['publishedAt'] = pd.to_datetime(video_df['publishedAt'], format="%Y-%m-%dT
#Check
video_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3936 entries, 0 to 3935
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   video_id        3936 non-null   object
 1   channelTitle    3936 non-null   object
 2   title           3936 non-null   object
 3   description     3936 non-null   object
 4   tags            3541 non-null   object
 5   publishedAt     3936 non-null   datetime64[ns]
 6   viewCount       3936 non-null   float64
 7   likeCount       3930 non-null   float64
 8   favouriteCount  0 non-null      float64
 9   commentCount    3935 non-null   float64
 10  duration        3936 non-null   object
 11  definition      3936 non-null   object
 12  caption         3936 non-null   object
dtypes: datetime64[ns](1), float64(4), object(8)
memory usage: 399.9+ KB
```

In [22]:
```
# Help: https://stackoverflow.com/questions/29096381/num-day-to-name-day-with-panda
video_df['publishedDayName'] = video_df['publishedAt'].dt.day_name()
video_df['publishedMonthName'] = video_df['publishedAt'].dt.month_name()

#Check
video_df.head(1)
```

Out[22]:

	video_id	channelTitle	title	description	tags	publishedAt	viewCount	likeCoun
0	GDN649X_acE	StatQuest with Josh Starmer	Encoder-Only Transformers (like BERT) for RAG,...	Encoder-Only Transformers are the backbone for...	[Josh Starmer, StatQuest, Machine Learning, BE...	2024-11-18 05:00:11	24006.0	864.(

In [25]:
```
# convert duration column to seconds with isodate
# help: https://stackoverflow.com/questions/16742381/how-to-convert-youtube-api-dur
# time delta help: https://pandas.pydata.org/docs/user_guide/timedeltas.html

video_df['durationSec'] = video_df['duration'].apply(lambda x:isodate.parse_duratio
video_df['durationSec'] = video_df['durationSec'].astype('timedelta64[s]')
#check
video_df.head(1)
```

Out[25]:

	video_id	channelTitle	title	description	tags	publishedAt	viewCount	likeCoun
0	GDN649X_acE	StatQuest with Josh Starmer	Encoder-Only Transformers (like BERT) for RAG,...	Encoder-Only Transformers are the backbone for...	[Josh Starmer, StatQuest, Machine Learning, BE...	2024-11-18 05:00:11	24006.0	864.(

In [26]:
```
# len(video_df['tags'][2195]) - produced a number
# len(video_df['tags'][0]) - produced a Nonetype error; must address
video_df['tagsCount'] = video_df['tags'].apply(lambda x: 0 if x is None else len(x)

#check
video_df.tail()
```

Out[26]:

	video_id	channelTitle	title	description	tags	publishedAt	viewCount	likeC
3931	JjuCGJyZacE	Nicholas Renotte	Generating Credentials - Build An Image Classi...	Tired of struggling to build an image classifi...	[image classification, python, ibm, visual rec...	2019-01-29 21:29:54	1624.0	
3932	PPq79Q51Ya4	Nicholas Renotte	Installing Watson Developer Cloud - Build An I...	Tired of struggling to build an image classifi...	[visual recognition, image classification, pyt...	2019-01-29 21:29:50	1502.0	
3933	oKDkwZkzX48	Nicholas Renotte	General Image Classification - Build An Image ...	Tired of struggling to build an image classifi...	[watson, ibm, visual recognition, image classi...	2019-01-29 21:29:47	2006.0	
3934	FgZsV09npbs	Nicholas Renotte	Food Image Classification - Build An Image Cla...	Tired of struggling to build an image classifi...	[python, image classification, watson, visual ...	2019-01-29 21:29:44	2831.0	
3935	z16aNdvgAog	Nicholas Renotte	Face Detection - Build An Image Classifier wit...	Tired of struggling to build an image classifi...	[python, ibm, watson, image classification, vi...	2019-01-29 21:29:41	4115.0	

In [27]:
```
days_ordered = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
days_ordered_var = pd.api.types.CategoricalDtype(ordered = True, categories = days_
video_df.publishedDayName = video_df.publishedDayName.astype(days_ordered_var)
```

In [28]:
```
months_ordered = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
          'August', 'September', 'October', 'November', 'December']
months_ordered_var = pd.api.types.CategoricalDtype(ordered = True, categories = mon
video_df.publishedMonthName = video_df.publishedMonthName.astype(months_ordered_var
```

In [29]:
```
video_df.drop(columns = ['favouriteCount'], inplace = True)
# check
video_df.head()
```

Out[29]:

	video_id	channelTitle	title	description	tags	publishedAt
0	GDN649X_acE	StatQuest with Josh Starmer	Encoder-Only Transformers (like BERT) for RAG,...	Encoder-Only Transformers are the backbone for...	[Josh Starmer, StatQuest, Machine Learning, BE...	2024-11-18 05:00:11
1	qJrmQe8TOTw	StatQuest with Josh Starmer	Luis Serrano + Josh Starmer Q&A Livestream!!!	Join me, Luis Serrano http://www.youtube.com/c...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-10-10 04:04:08
2	DkmfIQRDyXc	StatQuest with Josh Starmer	Human Stories in AI: Nana Janashia@TechWorld W...	In this episode we have special guest Nana Jan...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-09-09 04:00:17
3	0QOm7Sn5uwQ	StatQuest with Josh Starmer	A few more lessons from my Pop!	Since September 4th is Global Frank Starmer Da...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-09-04 04:00:00
4	wIGOnM6Cf_E	StatQuest with Josh Starmer	Human Stories in AI: Abbas Merchant@Matics Ana...	In this episode we have special guest Abbas Me...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-07-29 04:00:38

In [30]:
```
video_df['title_length'] = video_df['title'].apply(lambda x: len(x))
video_df.head()
```

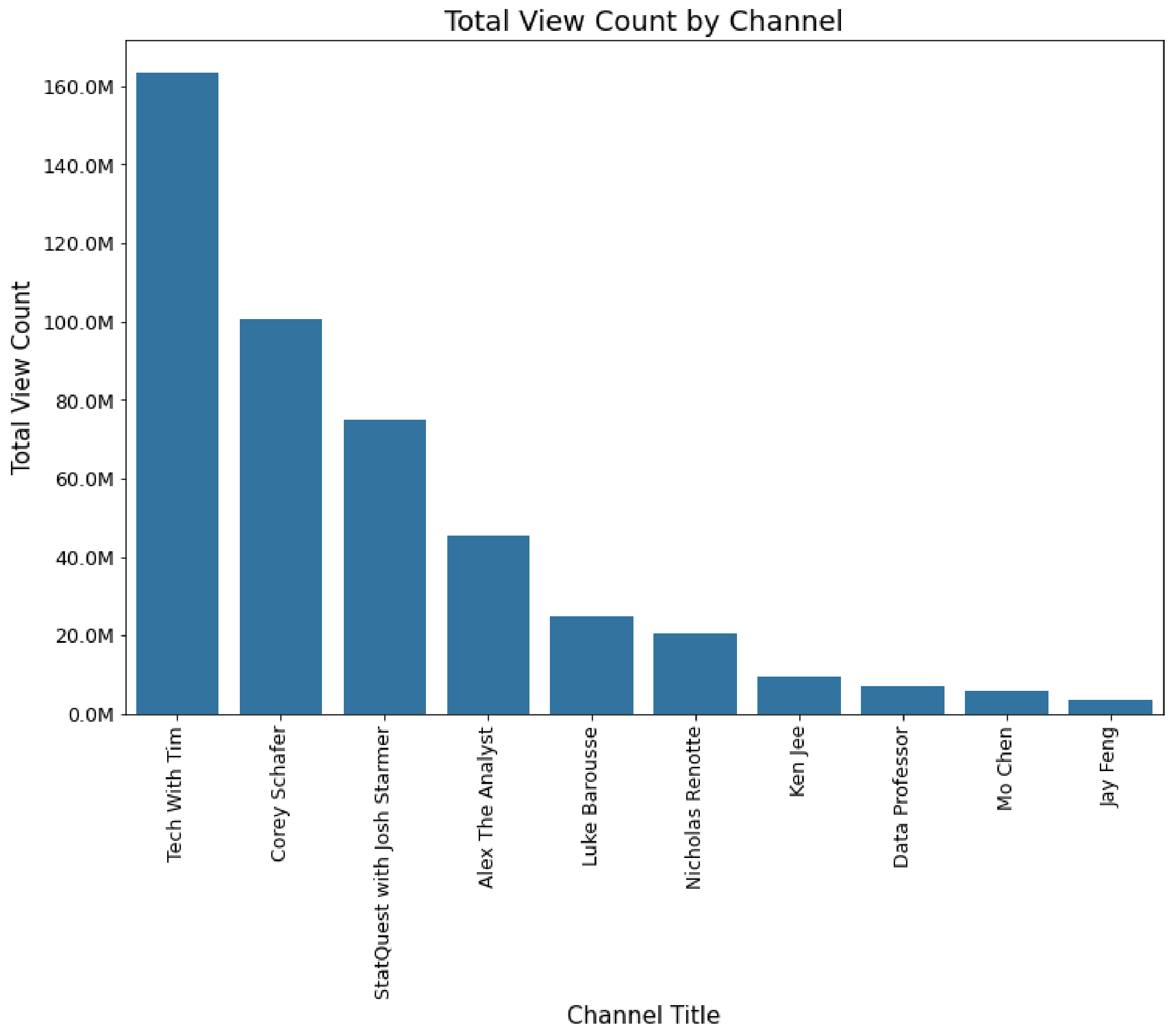
Out[30]:

	video_id	channelTitle	title	description	tags	publishedAt
0	GDN649X_acE	StatQuest with Josh Starmer	Encoder-Only Transformers (like BERT) for RAG,...	Encoder-Only Transformers are the backbone for...	[Josh Starmer, StatQuest, Machine Learning, BE...	2024-11-18 05:00:11
1	qJrmQe8TOTw	StatQuest with Josh Starmer	Luis Serrano + Josh Starmer Q&A Livestream!!!	Join me, Luis Serrano http://www.youtube.com/c...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-10-10 04:04:08
2	DkmflQRDyXc	StatQuest with Josh Starmer	Human Stories in AI: Nana Janashia@TechWorld W...	In this episode we have special guest Nana Jan...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-09-09 04:00:17
3	0QOm7Sn5uwQ	StatQuest with Josh Starmer	A few more lessons from my Pop!	Since September 4th is Global Frank Starmer Da...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-09-04 04:00:00
4	wIGOnM6Cf_E	StatQuest with Josh Starmer	Human Stories in AI: Abbas Merchant@Matics Ana...	In this episode we have special guest Abbas Me...	[Josh Starmer, StatQuest, Machine Learning, St...	2024-07-29 04:00:38

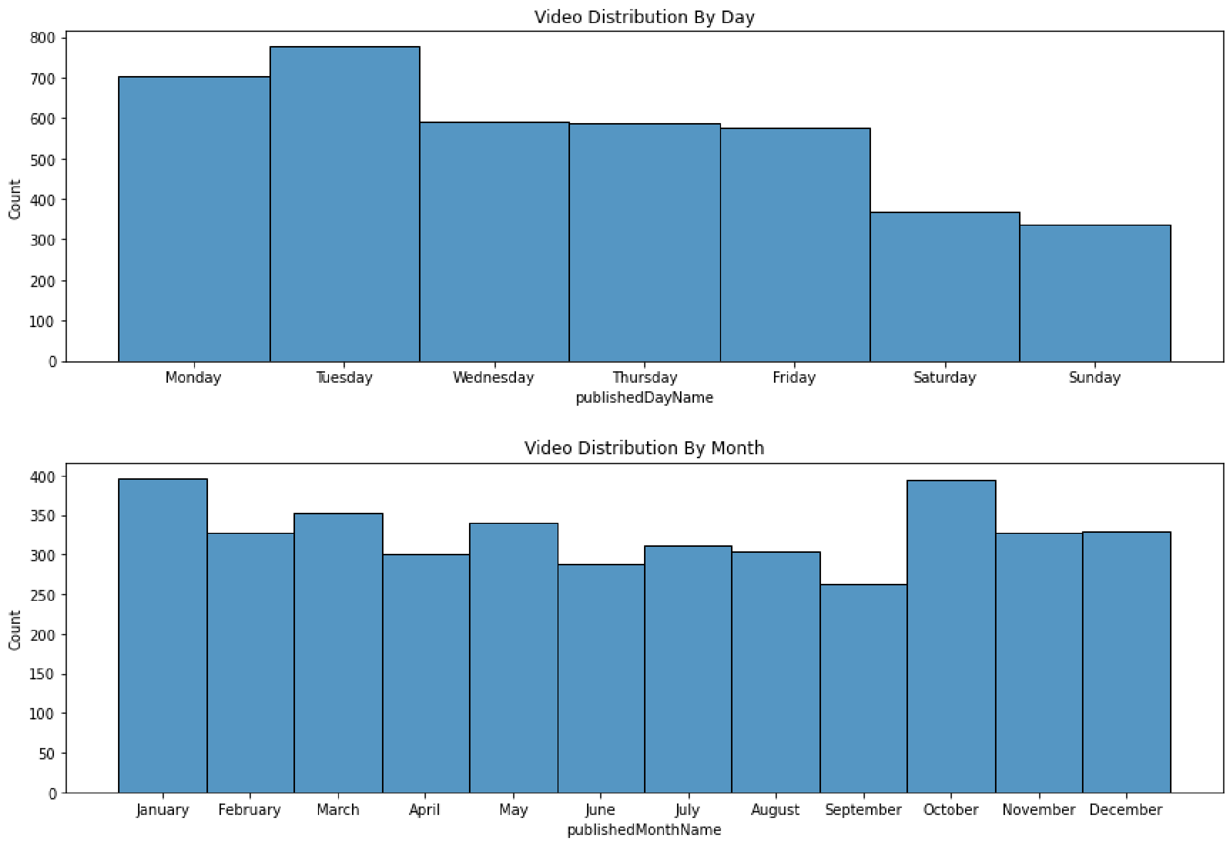
Exploratory Data Analysis

In [32]:
```
def millions(x, pos):
    return '%1.1fM' % (x*1e-6)

formatter = FuncFormatter(millions)
```

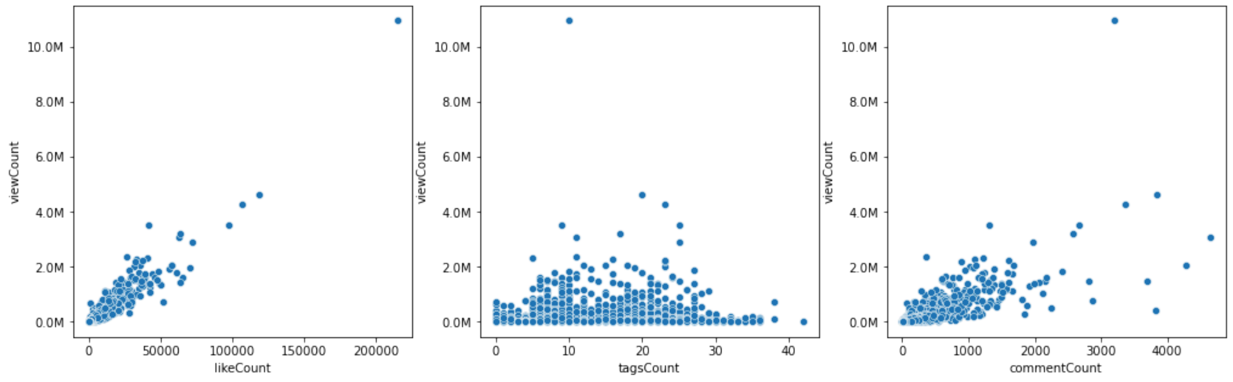
In [33]:
```
base_color = sb.color_palette()[0]
plt.figure(figsize = (12, 8))
ax = sb.barplot(x = 'channelTitle', y = 'viewCount',
                data = video_df.groupby('channelTitle')['viewCount'].sum().sort_val
               color = base_color)
ax.yaxis.set_major_formatter(formatter)

plt.title('Total View Count by Channel', fontsize = 18)
plt.xticks(rotation = 90, fontsize = 12.5)
plt.yticks(fontsize = 12.5)
plt.xlabel('Channel Title', fontsize = 15)
plt.ylabel('Total View Count', fontsize = 15);
```



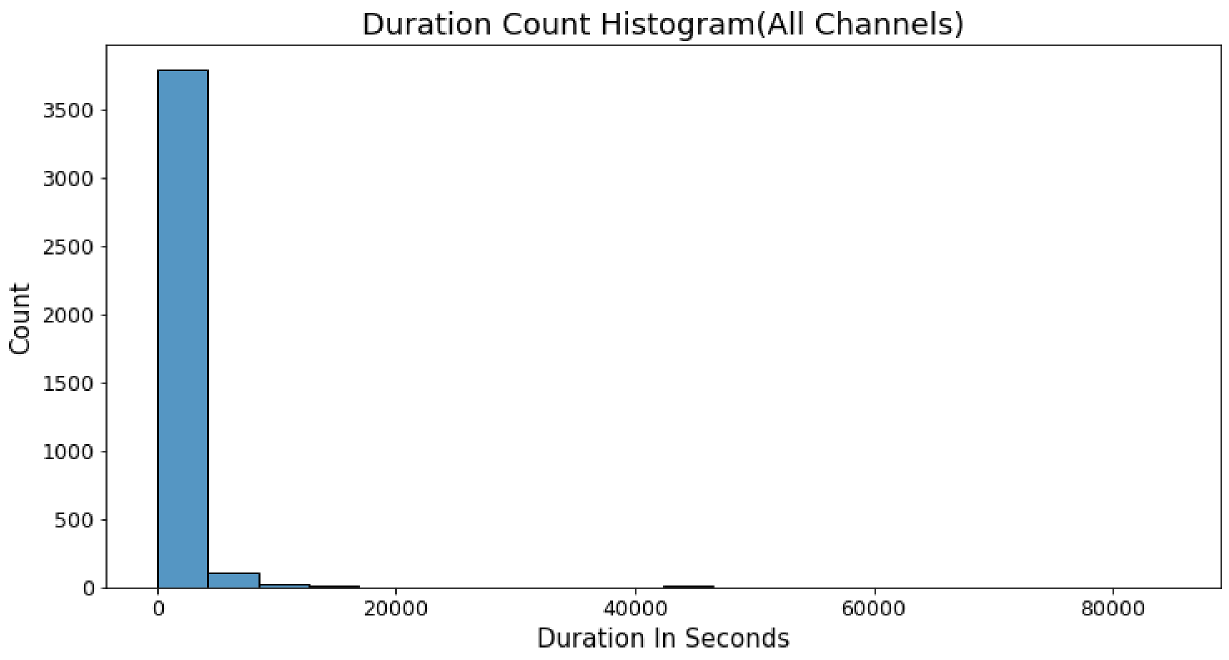
In [34]:
```
fig, ax = plt.subplots(2, 2, figsize = [12, 8])
fig.tight_layout(h_pad=5)

plt.subplot(2, 1, 1)
sb.histplot(data = video_df, x = "publishedDayName")
plt.title('Video Distribution By Day')

plt.subplot(2, 1, 2)
sb.histplot(data = video_df, x = "publishedMonthName")
plt.title('Video Distribution By Month');
```



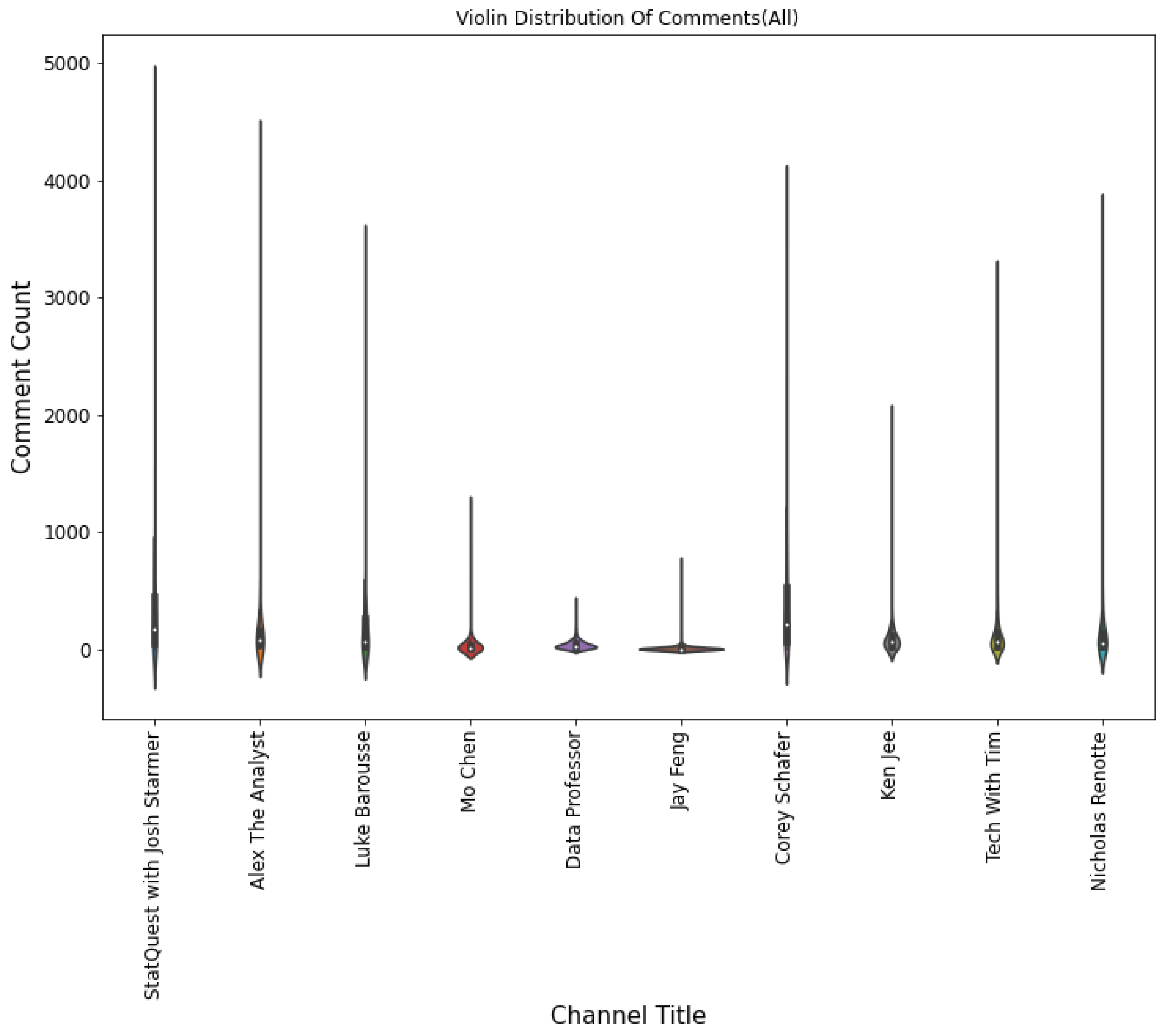
In [35]:
```
fig, ax = plt.subplots(1,3, figsize = [17, 5])

sb.scatterplot(data = video_df, x = 'likeCount', y = 'viewCount', ax = ax[0])
sb.scatterplot(data = video_df, x = 'tagsCount', y = 'viewCount', ax = ax[1])
sb.scatterplot(data = video_df, x = 'commentCount', y = 'viewCount', ax = ax[2])

# convert scietific notations on y-axis to millions
for i in range(3):
    ax[i].yaxis.set_major_formatter(formatter);
```



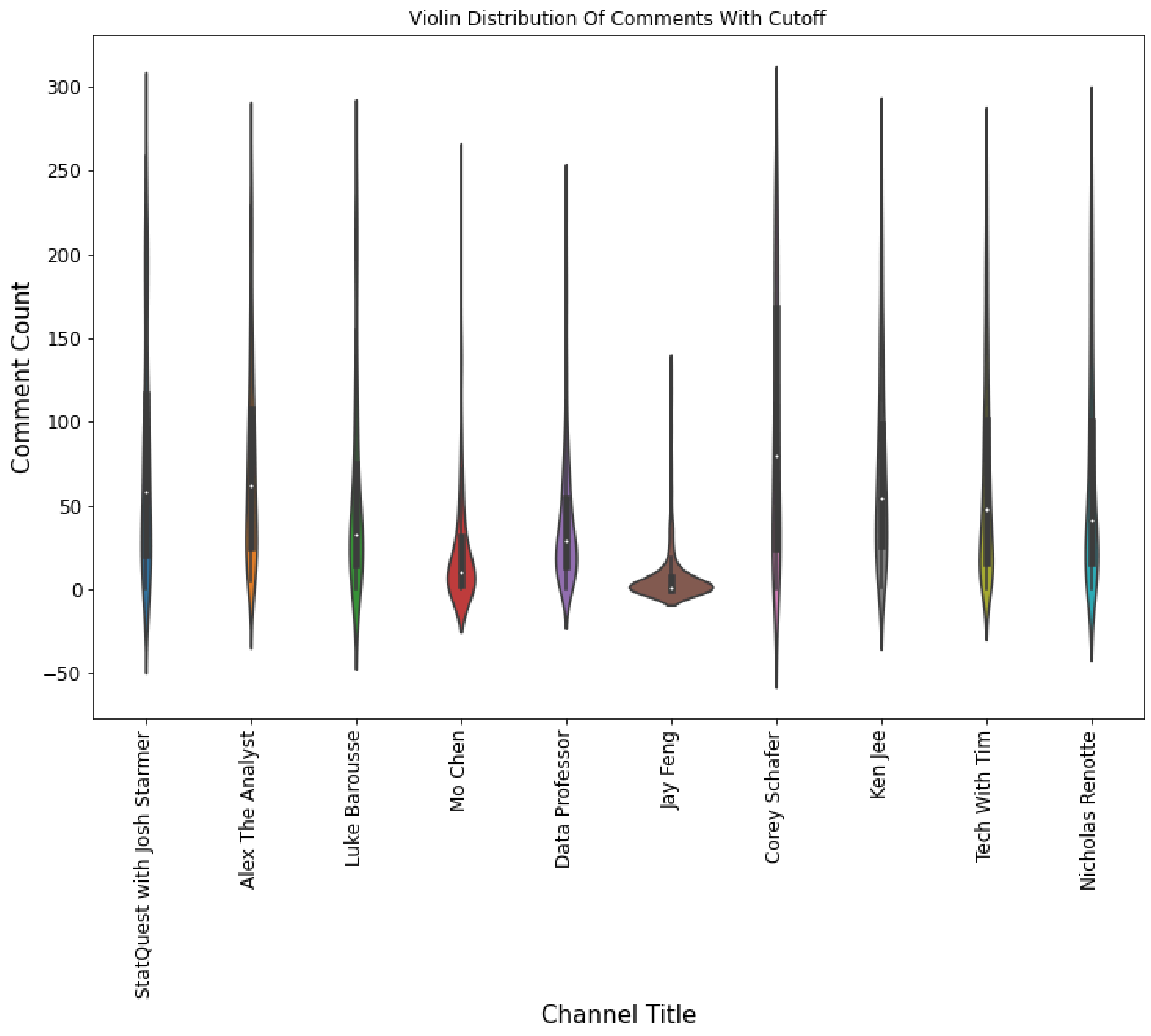
In [42]:
```
plt.figure(figsize = (12, 6))
sb.histplot(data=video_df, x="durationSec", bins = 20)

plt.title('Duration Count Histogram(All Channels)', fontsize = 18)
plt.xticks(fontsize = 12.5)
plt.yticks(fontsize = 12.5)
plt.xlabel('Duration In Seconds', fontsize = 15)
plt.ylabel('Count', fontsize = 15);
```



In [43]:
```
fig = plt.figure(figsize = (12, 8))
ax = sb.violinplot(data = video_df, x = 'channelTitle', y = 'commentCount')

plt.xticks(rotation = 90, fontsize = 12)
plt.yticks(fontsize = 12)
plt.xlabel("Channel Title", fontsize = 15)
plt.ylabel("Comment Count", fontsize = 15)
plt.title("Violin Distribution Of Comments(All)");
```

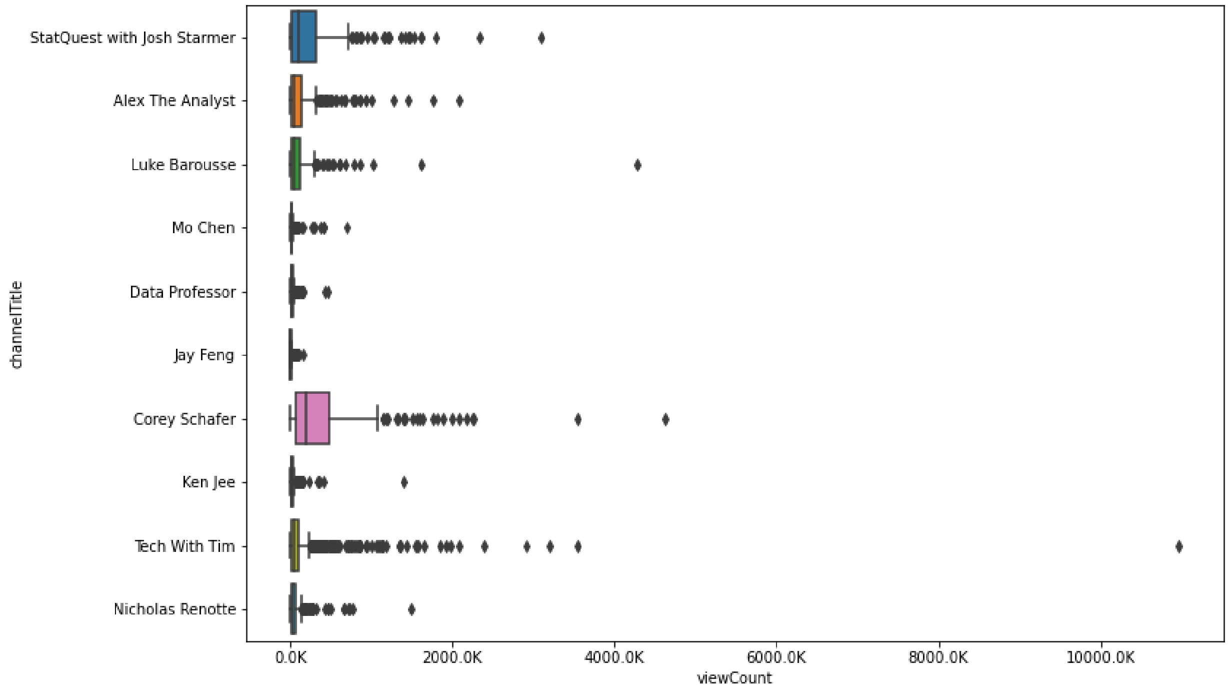


In [44]:
```
# establish cutoff point, get rid of outliers for better viewing
double_Q3 = (video_df['commentCount'].quantile(0.75))*2

#plot
fig = plt.figure(figsize = (12, 8))
ax = sb.violinplot(data = video_df[video_df.commentCount < double_Q3], x = 'channel
plt.xticks(rotation = 90, fontsize = 12)
plt.yticks(fontsize = 12)
plt.xlabel("Channel Title", fontsize = 15)
plt.ylabel("Comment Count", fontsize = 15)
plt.title("Violin Distribution Of Comments With Cutoff");
```

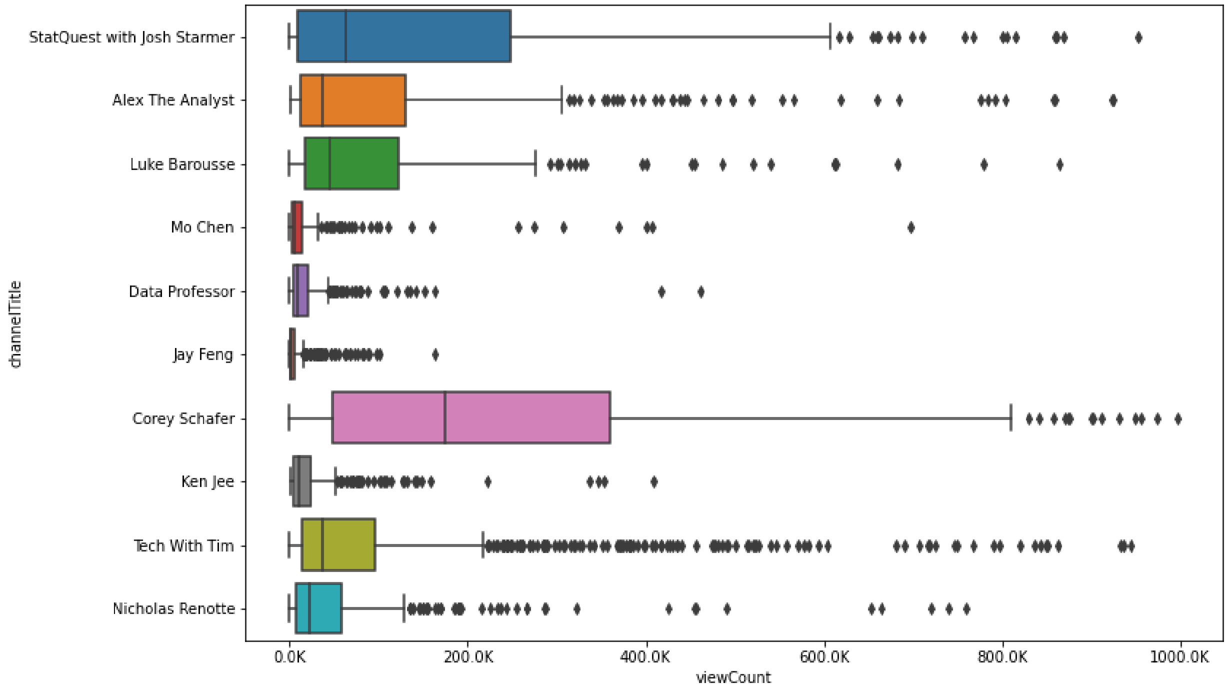


In [45]:
```
# function similair to the million() function, but for thousands
def thousands(x, pos):
    """
    INPUT:
    x: numerical value
    pos: tick position

    OUTPUT: formatted string of % (x*1e-3) with K to represent thousands
    """
    return '%1.1fK' % (x*1e-3)

formatter = FuncFormatter(thousands)
```

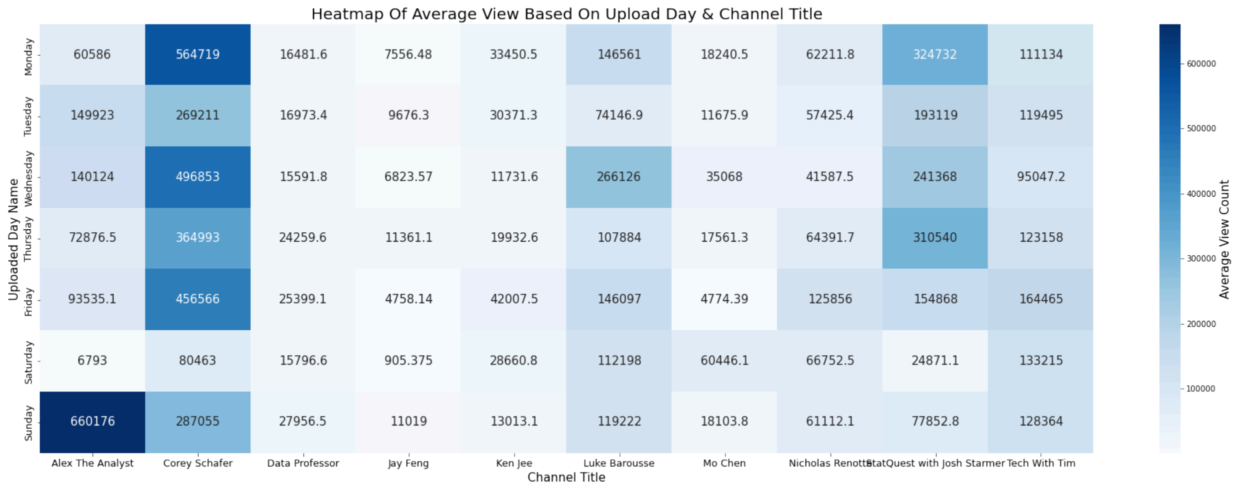
In [46]:
```
#plot
fig = plt.figure(figsize = (12, 8))
ax = sb.boxplot(data = video_df, y = 'channelTitle', x = 'viewCount')
ax.xaxis.set_major_formatter(formatter)
```



In [49]:
```
#plot
cutoff = 1000000

fig = plt.figure(figsize = (12, 8))
ax = sb.boxplot(data = video_df[video_df.viewCount < cutoff], y = 'channelTitle', x
ax.xaxis.set_major_formatter(formatter)
```

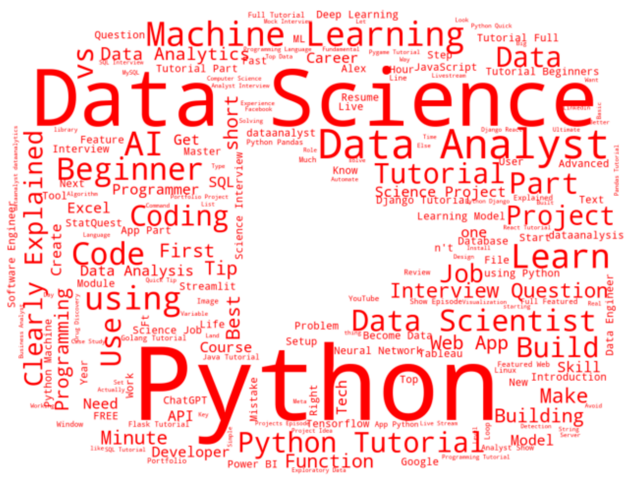


In [50]:
```
channel_view_means = video_df.groupby(['channelTitle', 'publishedDayName'])['viewCo
channel_view_means = channel_view_means.pivot(columns = 'channelTitle', index = 'pu

fig = plt.figure(figsize = (30,10))
ax = sb.heatmap(channel_view_means, annot = True, cmap='Blues',
                fmt='g', annot_kws = {'size': 15}, cbar_kws = {'label': 'Average Vi

plt.title('Heatmap Of Average View Based On Upload Day & Channel Title',
         fontsize = 20)
plt.xticks(rotation = 360, fontsize = 12.5)
plt.yticks(fontsize = 12.5)
plt.xlabel("Channel Title", fontsize = 15)
plt.ylabel("Uploaded Day Name", fontsize = 15)

# setting color bar fontsize
# help: https://stackoverflow.com/questions/48586738/seaborn-heatmap-colorbar-label
ax.figure.axes[-1].yaxis.label.set_size(15);
```



In [51]:
```
# funciton for plotting wordcloud
# word cloud help: https://www.youtube.com/watch?v=f1TJXu5H8ZM
def plot_youtube_cloud(data):
    """
    INPUT:
    data: Input data structure.
    title: (str) Title of Word Cloud

    OUTPUT:
    No Output
    """
    # import youtube outline
    image = imageio.imread('youtube_image.png')
    wordcloud = WordCloud(background_color = 'white',
                          mask = image,
                          stopwords = stop_words,
                          max_words = 200,
                          max_font_size = 70,
                          scale = 3,
                          random_state =1,
                          # coloring help: https://github.com/amueller/word_cloud/is
                          color_func=lambda *args, **kwargs: "red").generate(str(dat
    fig = plt.figure(1, figsize = (20,20), dpi=80)
    plt.axis('off')

    plt.imshow(wordcloud)
    plt.show()
```

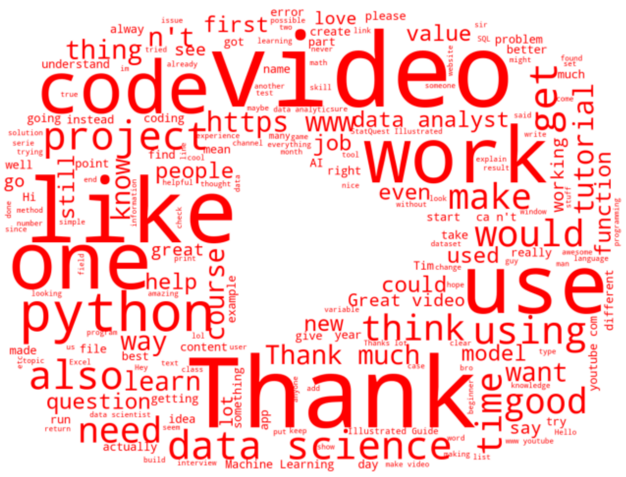
In [53]:
```
# setting up stop words
stop_words = set(stopwords.words('english'))
# word tokenization help: https://stackabuse.com/removing-stop-words-from-strings-i
video_df['title_no_stopwords'] = video_df['title'].apply(lambda x: [w for w in word

# append all words 'title_no_stopwords' into single list
all_words = list([a for b in video_df['title_no_stopwords'].tolist() for a in b])
# join all words together in list into single string
all_words_str = ' '.join(all_words)
plot_youtube_cloud(all_words_str)
```



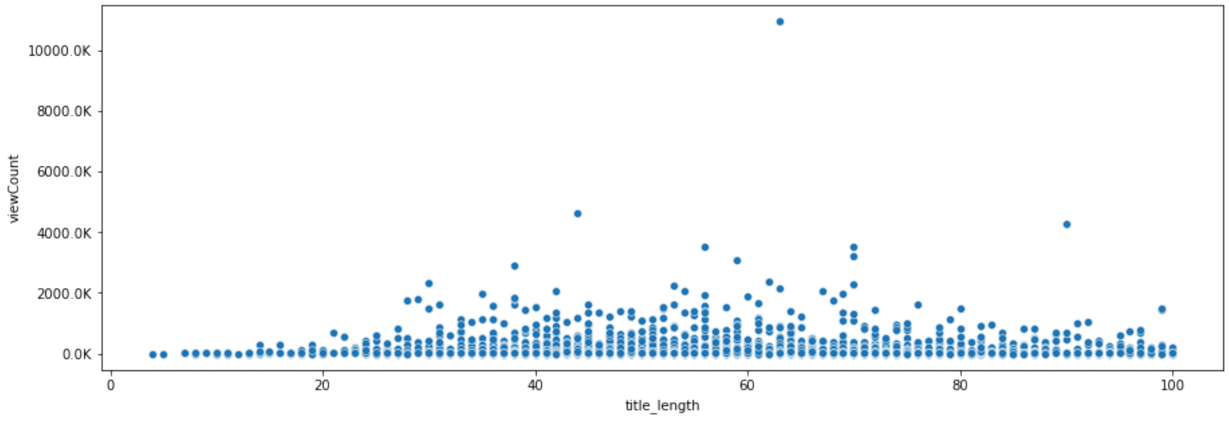
In [54]:
```
# join list of comments into a single string
for i in range(len(all_comments_df['comments'])):
    all_comments_df['comments'][i] = " ".join(all_comments_df['comments'][i])
all_comments_df['comments'].head()
```

Out[54]:
```
0    Support StatQuest by buying my books The StatQ...
1    Josh is so humble, but a genius :). Thanks so ...
2    Amazing job! never stop making videos, or else...
3    Support StatQuest by buying my book The StatQu...
4    Actually your clips are good, not only chasing...
Name: comments, dtype: object
```

In [56]:
```
# new column for words not in stopwords
all_comments_df['comments_no_stopwords'] = all_comments_df['comments'].apply(lambda

# append all words 'title_no_stopwords' into single list
all_words = list([a for b in all_comments_df['comments_no_stopwords'].tolist() for
# join all words together in list into single string
all_words_str = ' '.join(all_words)
plot_youtube_cloud(all_words_str)
```



In [57]:
```
fig, ax = plt.subplots(figsize = [15, 5])
sb.scatterplot(data = video_df, x = 'title_length', y = 'viewCount')
ax.yaxis.set_major_formatter(formatter);
```



In [ ]: