Activity_Course 3 TikTok project lab

August 26, 2023

1 TikTok Project

The goal is to explore the dataset and create visualizations.

Part 1: Imports, links, and loading

Part 2: Data Exploration * Data cleaning

Part 3: Build visualizations

Part 4: Evaluate and share results

1.0.1 Imports, links, and loading

```
[1]: # Import packages for data manipulation
     import pandas as pd
     import numpy as np

     # Import packages for data visualization
     import matplotlib.pyplot as plt
     import seaborn as sb
```

```
[2]: # Load dataset into dataframe
     data = pd.read_csv("tiktok_dataset.csv")
```

1.0.2 Data exploration and cleaning

```
[3]: # Display and examine the first few rows of the dataframe
     data.head(5)
```

```
[3]:    # claim_status    video_id  video_duration_sec  \
     0  1        claim  7017666017                  59
     1  2        claim  4014381136                  32
     2  3        claim  9859838091                  31
     3  4        claim  1866847991                  25
     4  5        claim  7105231098                  19
```

```
                            video_transcription_text verified_status  \
0  someone shared with me that drone deliveries a…    not verified
1  someone shared with me that there are more mic…    not verified
2  someone shared with me that american industria…    not verified
3  someone shared with me that the metro of st. p…    not verified
4  someone shared with me that the number of busi…    not verified

  author_ban_status  video_view_count  video_like_count  video_share_count  \
0      under review          343296.0           19425.0              241.0
1            active          140877.0           77355.0            19034.0
2            active          902185.0           97690.0             2858.0
3            active          437506.0          239954.0            34812.0
4            active           56167.0           34987.0             4110.0

   video_download_count  video_comment_count
0                   1.0                  0.0
1                1161.0                684.0
2                 833.0                329.0
3                1234.0                584.0
4                 547.0                152.0
```

[4]:
```
# Get the size of the data
np.size(data)
```

[4]:
```
232584
```

[5]:
```
# Get the shape of the data
np.shape(data)
```

[5]:
```
(19382, 12)
```

[6]:
```
# Get basic information about the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19382 entries, 0 to 19381
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   #                         19382 non-null  int64
 1   claim_status              19084 non-null  object
 2   video_id                  19382 non-null  int64
 3   video_duration_sec        19382 non-null  int64
 4   video_transcription_text  19084 non-null  object
 5   verified_status           19382 non-null  object
 6   author_ban_status         19382 non-null  object
 7   video_view_count          19084 non-null  float64
 8   video_like_count          19084 non-null  float64
 9   video_share_count         19084 non-null  float64
 10  video_download_count      19084 non-null  float64
 11  video_comment_count       19084 non-null  float64
dtypes: float64(5), int64(3), object(4)
memory usage: 1.8+ MB
```

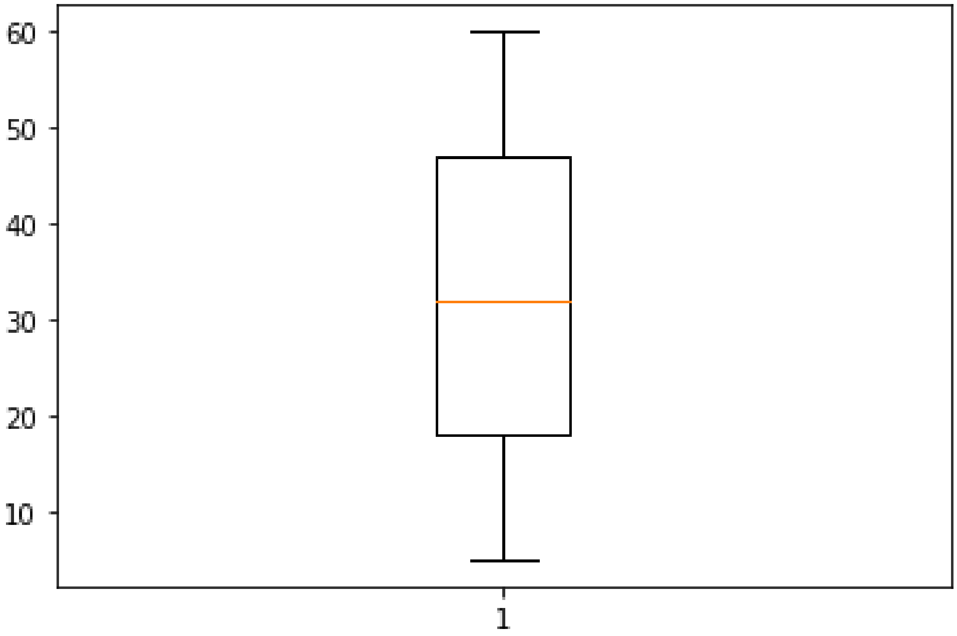
[7]:
```
# Generate a table of descriptive statistics
data.describe()
```

[7]:
```
                  #      video_id  video_duration_sec  video_view_count  \
count  19382.000000  1.938200e+04        19382.000000      19084.000000
mean    9691.500000  5.627454e+09           32.421732     254708.558688
std     5595.245794  2.536440e+09           16.229967     322893.280814
min        1.000000  1.234959e+09            5.000000         20.000000
25%     4846.250000  3.430417e+09           18.000000       4942.500000
50%     9691.500000  5.618664e+09           32.000000       9954.500000
75%    14536.750000  7.843960e+09           47.000000     504327.000000
max    19382.000000  9.999873e+09           60.000000     999817.000000

       video_like_count  video_share_count  video_download_count  \
count      19084.000000       19084.000000          19084.000000
mean       84304.636030       16735.248323           1049.429627
std       133420.546814       32036.174350           2004.299894
min            0.000000           0.000000              0.000000
25%          810.750000         115.000000              7.000000
50%         3403.500000         717.000000             46.000000
75%       125020.000000       18222.000000           1156.250000
max       657830.000000      256130.000000          14994.000000

       video_comment_count
count         19084.000000
mean            349.312146
std             799.638865
min               0.000000
25%               1.000000
50%               9.000000
75%             292.000000
max            9599.000000
```

1.0.3 Build visualizations

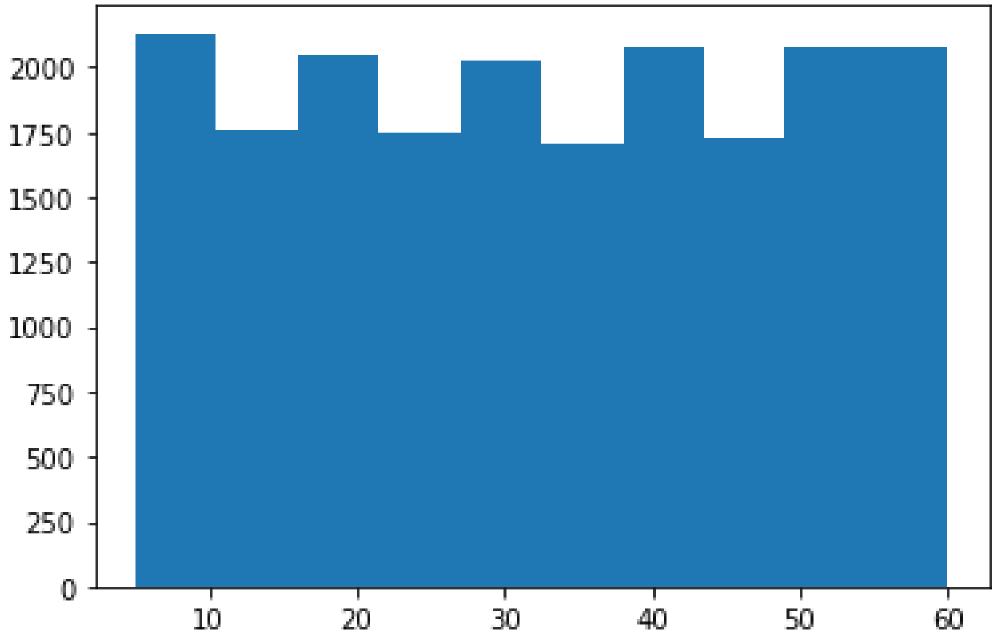
[8]:
```
# Create a boxplot to visualize distribution of `video_duration_sec`
plt.boxplot(data.video_duration_sec.dropna())
```

```
[8]: {'whiskers': [<matplotlib.lines.Line2D at 0x7f2b8c0b6290>,
       <matplotlib.lines.Line2D at 0x7f2b8c04a850>],
      'caps': [<matplotlib.lines.Line2D at 0x7f2b8c04ad90>,
       <matplotlib.lines.Line2D at 0x7f2b8c058310>],
      'boxes': [<matplotlib.lines.Line2D at 0x7f2b8c040cd0>],
      'medians': [<matplotlib.lines.Line2D at 0x7f2b8c058890>],
      'fliers': [<matplotlib.lines.Line2D at 0x7f2b8c040290>],
      'means': []}
```



```
[9]: # Create a histogram
     plt.hist(data.video_duration_sec.dropna())
     #The distribution of video lengths is surprisingly uniform.
```

```
[9]: (array([2131., 1761., 2048., 1749., 2030., 1704., 2080., 1723., 2078.,
             2078.]),
      array([ 5. , 10.5, 16. , 21.5, 27. , 32.5, 38. , 43.5, 49. , 54.5, 60. ]),
      <a list of 10 Patch objects>)
```

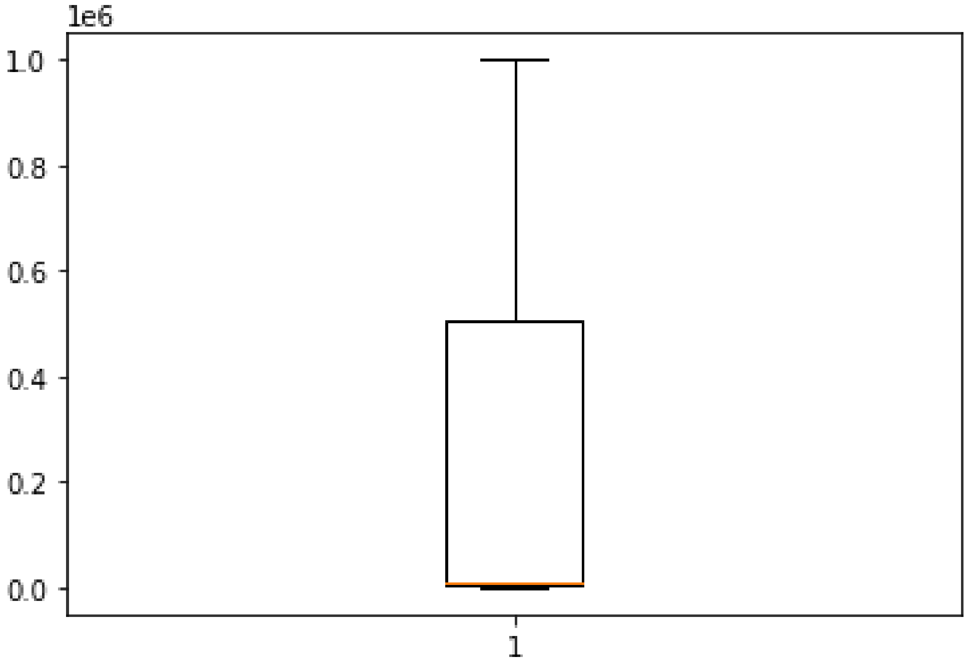


[10]:

```
# Create a boxplot to visualize distribution of `video_view_count`
plt.boxplot(data.video_view_count.dropna())
```

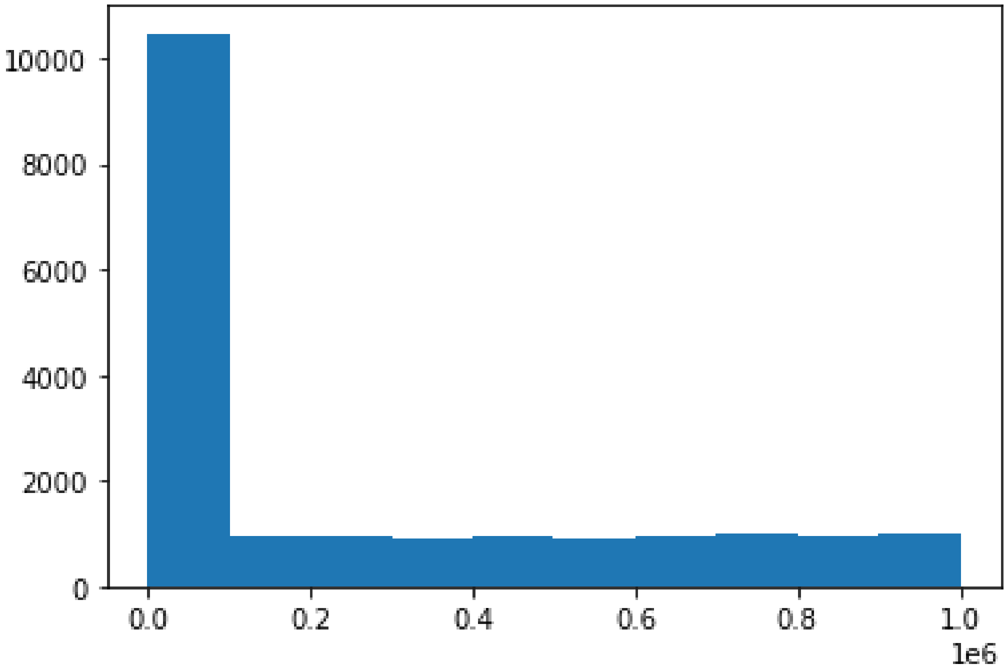
[10]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f2b8beff810>,
  <matplotlib.lines.Line2D at 0x7f2b8beffd50>],
 'caps': [<matplotlib.lines.Line2D at 0x7f2b8bf062d0>,
  <matplotlib.lines.Line2D at 0x7f2b8bf06810>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f2b8beff250>],
 'medians': [<matplotlib.lines.Line2D at 0x7f2b8bf06d90>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f2b8bf0e310>],
 'means': []}
```



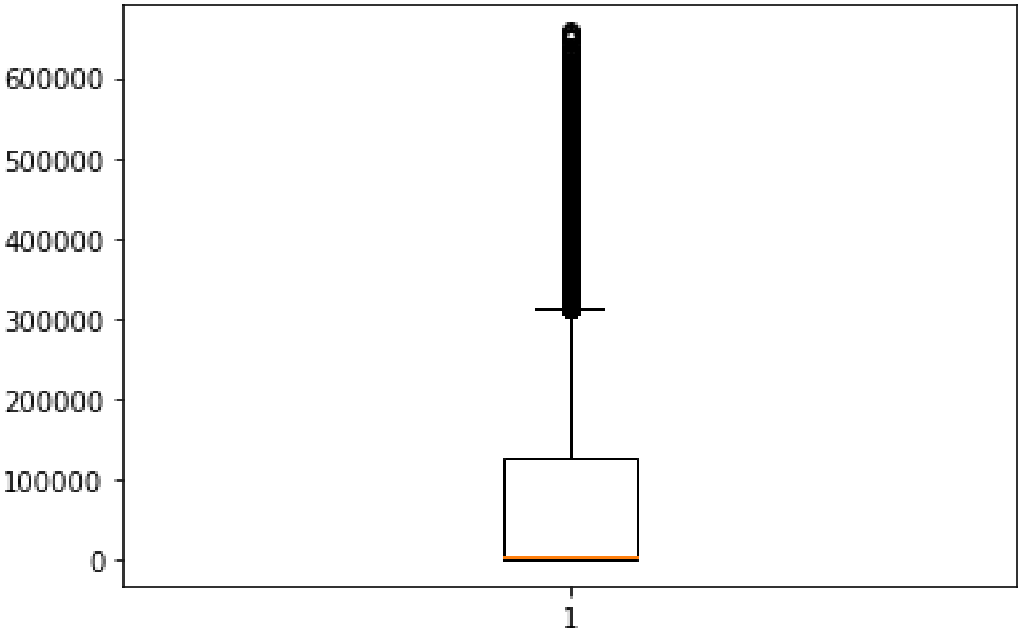
```
[11]: # Create a histogram
      plt.hist(data.video_view_count.dropna())
      #Video view counts are extremely skewed but become surprisingly uniform outside␣
       ↪of the category that we would consider viral videos.
```

```
[11]: (array([10475.,   961.,   944.,   913.,   972.,   920.,   965.,   991.,
                949.,   994.]),
       array([2.000000e+01, 9.999970e+04, 1.999794e+05, 2.999591e+05,
              3.999388e+05, 4.999185e+05, 5.998982e+05, 6.998779e+05,
              7.998576e+05, 8.998373e+05, 9.998170e+05]),
       <a list of 10 Patch objects>)
```



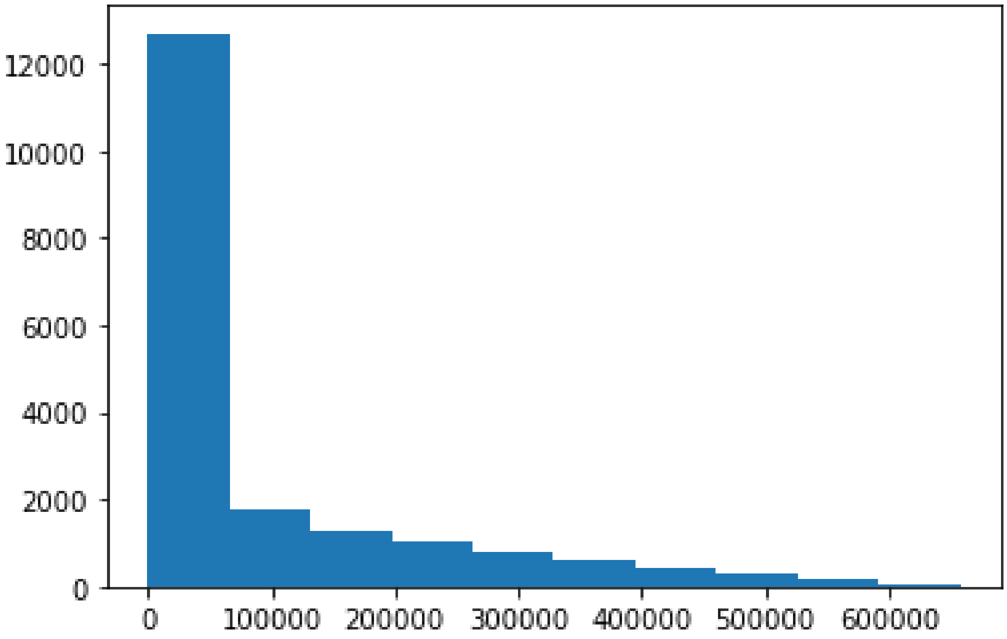
```
[12]: # Create a boxplot to visualize distribution of `video_like_count`
      plt.boxplot(data.video_like_count.dropna())
```

```
[12]: {'whiskers': [<matplotlib.lines.Line2D at 0x7f2b8be72dd0>,
        <matplotlib.lines.Line2D at 0x7f2b8bdf7350>],
       'caps': [<matplotlib.lines.Line2D at 0x7f2b8bdf7890>,
        <matplotlib.lines.Line2D at 0x7f2b8bdf7dd0>],
       'boxes': [<matplotlib.lines.Line2D at 0x7f2b8be72810>],
       'medians': [<matplotlib.lines.Line2D at 0x7f2b8bdff390>],
       'fliers': [<matplotlib.lines.Line2D at 0x7f2b8bdff8d0>],
       'means': []}
```



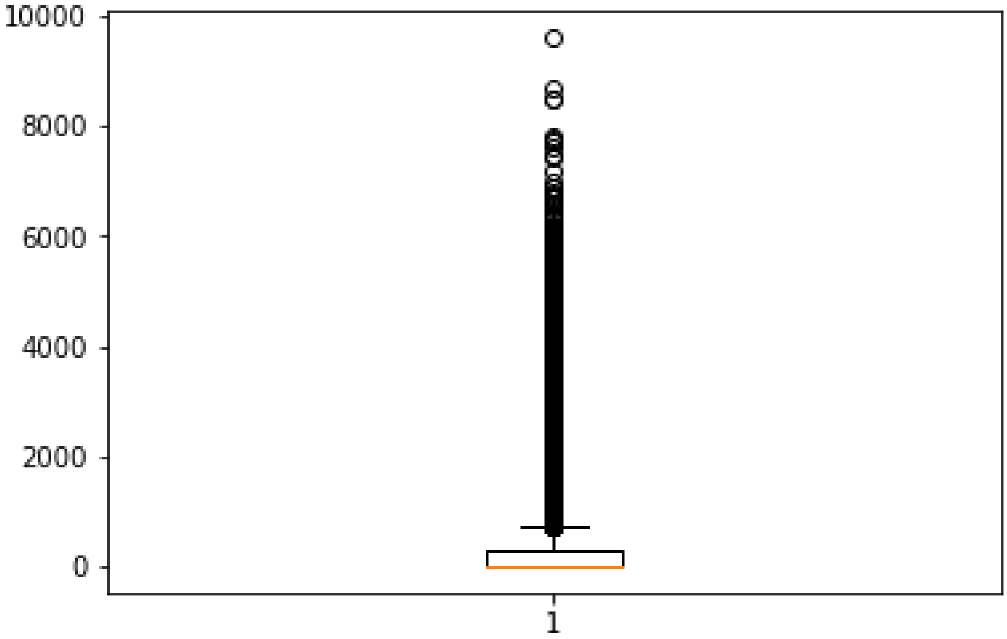
[13]:
```
# Create a histogram
plt.hist(data.video_like_count.dropna())
#Video like counts are extremely skewedc
```

[13]:
```
(array([12694.,  1775.,  1288.,  1035.,   763.,   580.,   426.,   296.,
          163.,    64.]),
 array([     0.,  65783., 131566., 197349., 263132., 328915., 394698.,
        460481., 526264., 592047., 657830.]),
 <a list of 10 Patch objects>)
```



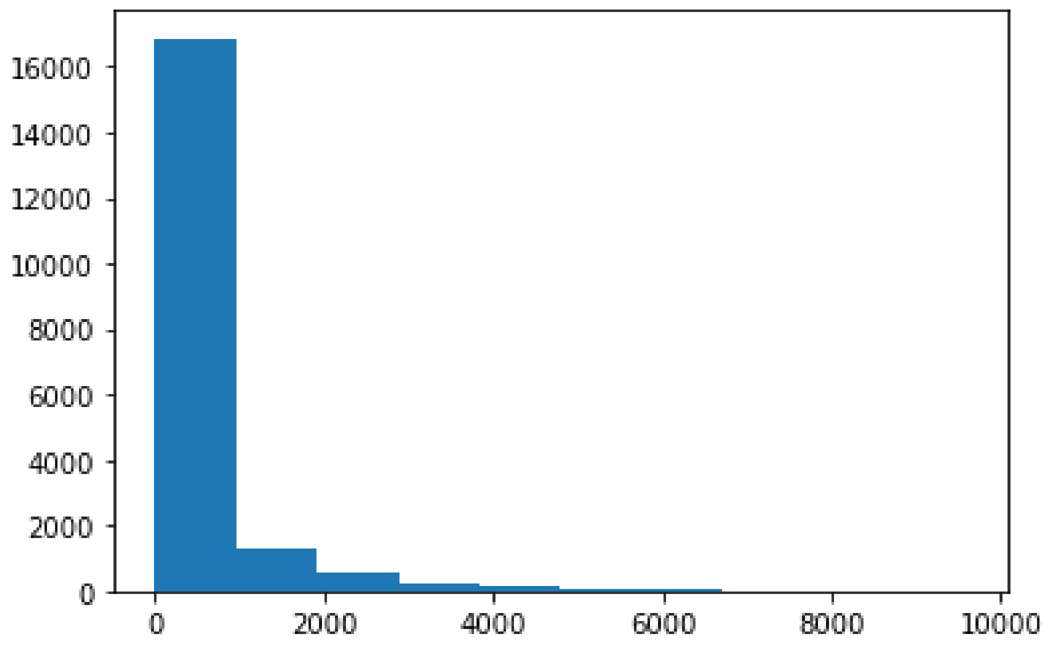
[14]:
```
# Create a boxplot to visualize distribution of `video_comment_count`
plt.boxplot(data.video_comment_count.dropna())
```

[14]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f2b8bd6e410>,
  <matplotlib.lines.Line2D at 0x7f2b8bd6e950>],
 'caps': [<matplotlib.lines.Line2D at 0x7f2b8bd6ee90>,
  <matplotlib.lines.Line2D at 0x7f2b8bcf5410>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f2b8bd68e10>],
 'medians': [<matplotlib.lines.Line2D at 0x7f2b8bcf5990>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f2b8bcf5ed0>],
 'means': []}
```



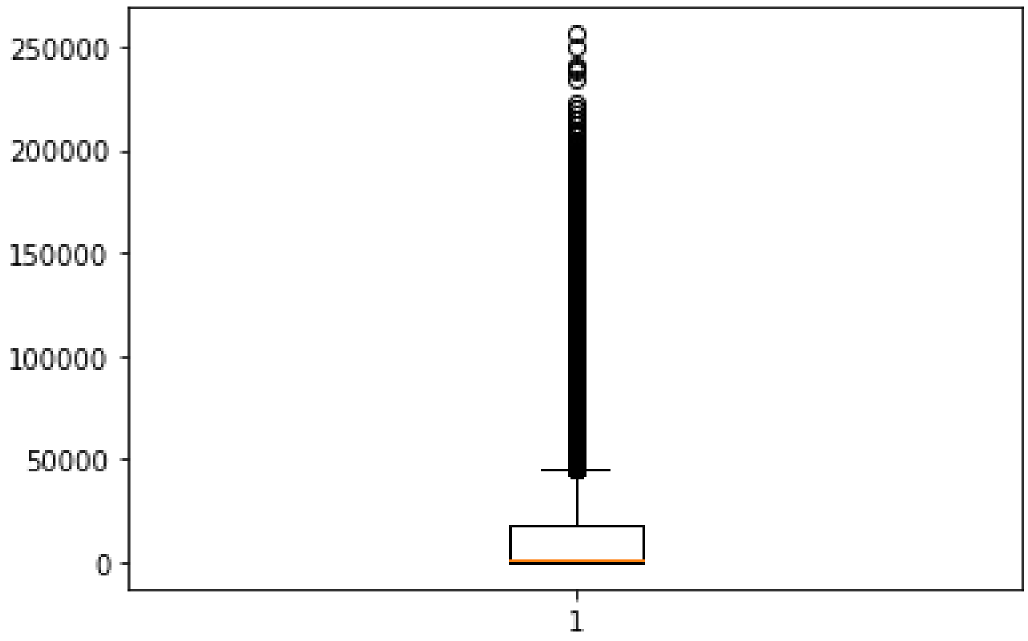
```
[15]: # Create a histogram
      plt.hist(data.video_comment_count.dropna())
      #Video comment counts are extremely skewed and contain a significant number of␣
       ↪flagged fliers.
      #This data might look significantly different with the outliers removed.
```

```
[15]: (array([1.6875e+04, 1.2510e+03, 5.2500e+02, 2.0900e+02, 1.1900e+02,
              5.5000e+01, 3.4000e+01, 9.0000e+00, 5.0000e+00, 2.0000e+00]),
       array([   0. ,  959.9, 1919.8, 2879.7, 3839.6, 4799.5, 5759.4, 6719.3,
              7679.2, 8639.1, 9599. ]),
       <a list of 10 Patch objects>)
```



[16]:
```
# Create a boxplot to visualize distribution of `video_share_count`
plt.boxplot(data.video_share_count.dropna())
```

[16]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f2b8bc4f450>,
  <matplotlib.lines.Line2D at 0x7f2b8bc4f990>],
 'caps': [<matplotlib.lines.Line2D at 0x7f2b8bc4fed0>,
  <matplotlib.lines.Line2D at 0x7f2b8bc57450>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f2b8bc49e50>],
 'medians': [<matplotlib.lines.Line2D at 0x7f2b8bc579d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f2b8bc57f10>],
 'means': []}
```

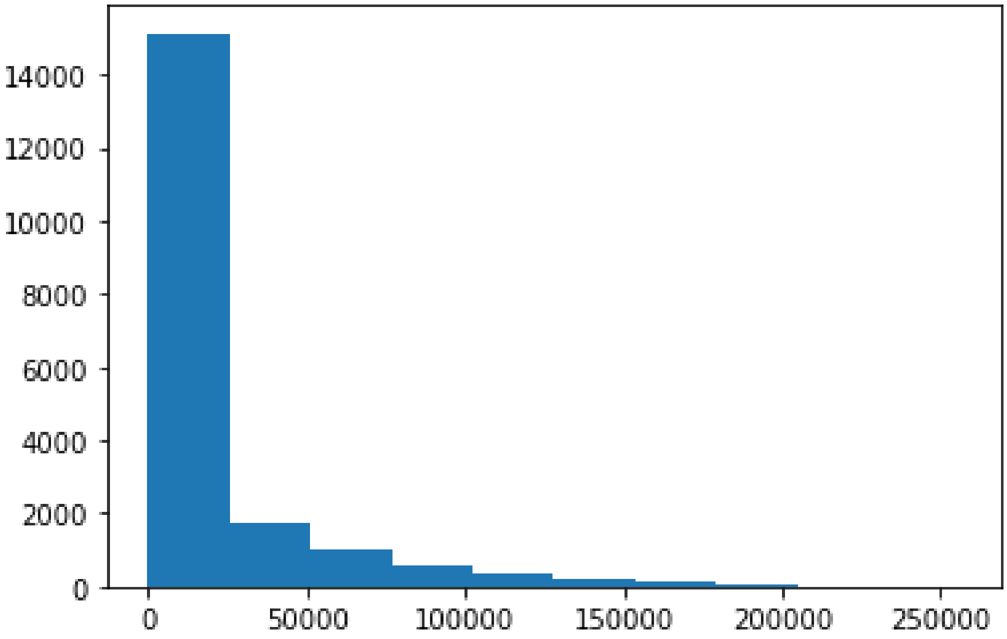


[17]:

```
# Create a histogram
plt.hist(data.video_share_count.dropna())
#Video share counts are extremely skewed and contain a significant number of␣
 ↪flagged fliers.
#This data might look significantly different with the outliers removed.
```

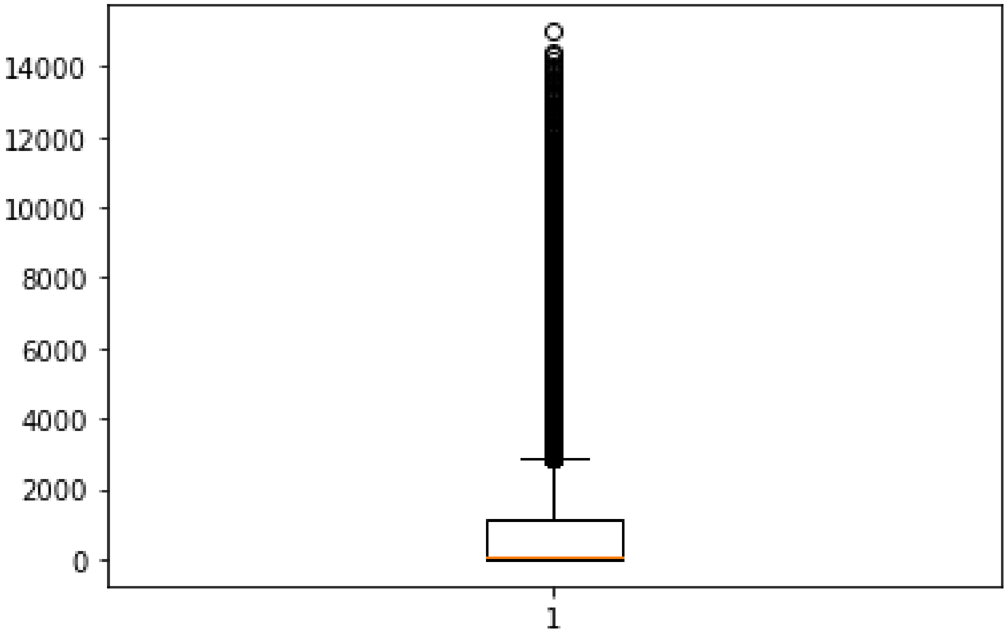
[17]:

```
(array([1.5127e+04, 1.7460e+03, 9.8600e+02, 5.4000e+02, 3.2900e+02,
        1.8800e+02, 9.7000e+01, 5.0000e+01, 1.5000e+01, 6.0000e+00]),
 array([     0.,  25613.,  51226.,  76839., 102452., 128065., 153678.,
        179291., 204904., 230517., 256130.]),
 <a list of 10 Patch objects>)
```



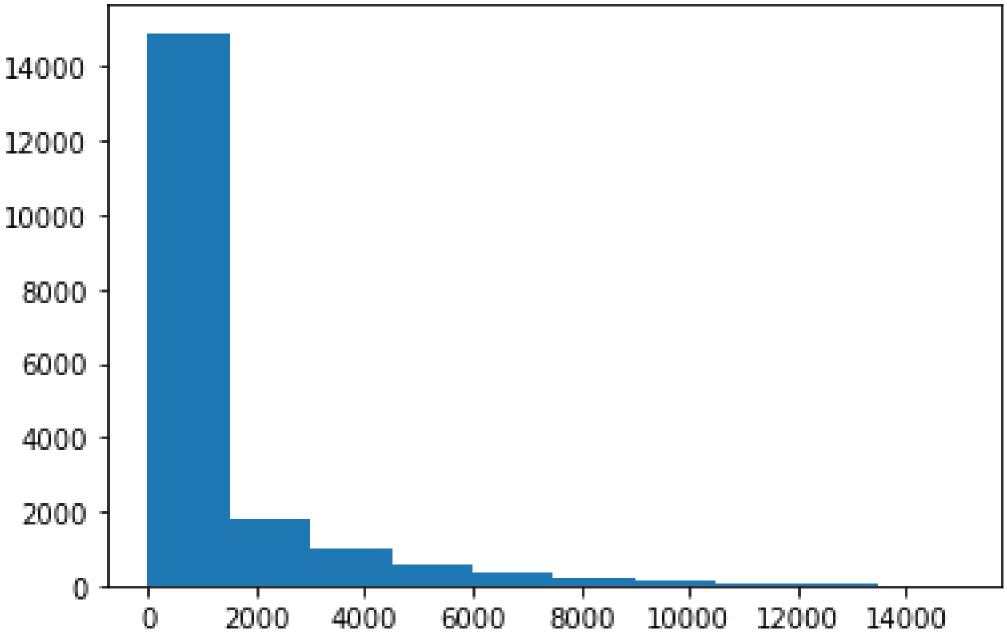
```
[18]: # Create a boxplot to visualize distribution of `video_download_count`
      plt.boxplot(data.video_download_count.dropna())
```

```
[18]: {'whiskers': [<matplotlib.lines.Line2D at 0x7f2b8bb46450>,
        <matplotlib.lines.Line2D at 0x7f2b8bb46990>],
       'caps': [<matplotlib.lines.Line2D at 0x7f2b8bb46ed0>,
        <matplotlib.lines.Line2D at 0x7f2b8bb4d450>],
       'boxes': [<matplotlib.lines.Line2D at 0x7f2b8bb41e50>],
       'medians': [<matplotlib.lines.Line2D at 0x7f2b8bb4d9d0>],
       'fliers': [<matplotlib.lines.Line2D at 0x7f2b8bb4df10>],
       'means': []}
```



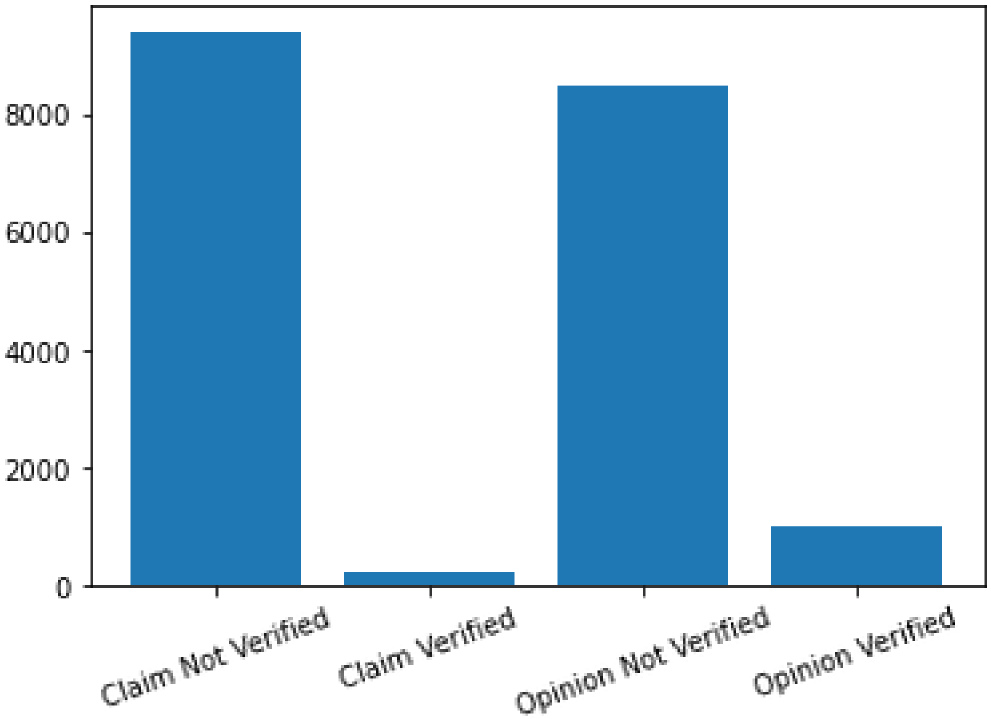
[19]:
```
# Create a histogram
plt.hist(data.video_download_count.dropna())
#Video download counts are extremely skewed and contain a significant number of␣
↪flagged fliers.
#This data might look significantly different with the outliers removed.
```

[19]:
```
(array([1.4921e+04, 1.8020e+03, 9.9500e+02, 5.7800e+02, 3.5500e+02,
        1.9800e+02, 1.2700e+02, 6.3000e+01, 3.3000e+01, 1.2000e+01]),
 array([    0. ,  1499.4,  2998.8,  4498.2,  5997.6,  7497. ,  8996.4,
        10495.8, 11995.2, 13494.6, 14994. ]),
 <a list of 10 Patch objects>)
```



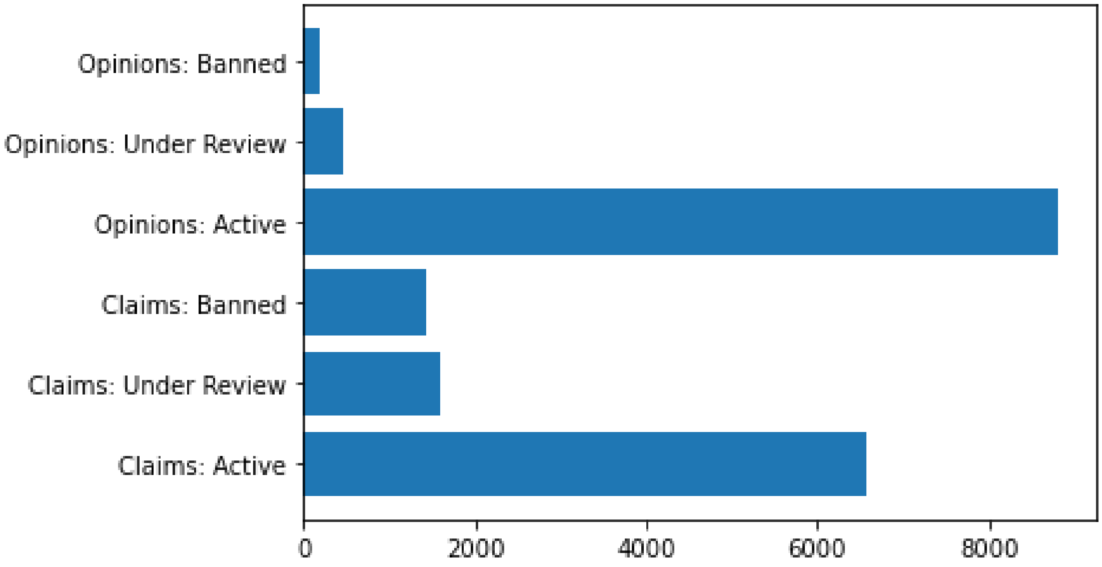
```
[20]: # Create a histogram with four bars: one for each combination of claim status␣
       ↪and verification status.
      data.groupby(['claim_status','verified_status']).size()
      CV = {'Claims_Verified':['Claim Not Verified','Claim Verified', 'Opinion Not␣
       ↪Verified', 'Opinion Verified'], 'Count':[9399,209,8485,991]}
      #claims vs verification

      plt.bar(CV['Claims_Verified'],CV['Count'])
      plt.xticks(CV['Claims_Verified'],rotation=20)
      #There are far more unverified authors than verified authors.
      #Verified authors are many times more likely to post opinions than claims.
      #Unverified authors could be somewhat more likely to post claims than opinions␣
       ↪but it's close so a hypothesis test would be necessary to prove it.
```

```
[20]: ([<matplotlib.axis.XTick at 0x7f2b8ba994d0>,
        <matplotlib.axis.XTick at 0x7f2b8ba9fc10>,
        <matplotlib.axis.XTick at 0x7f2b8bb41a50>,
        <matplotlib.axis.XTick at 0x7f2b8ba52dd0>],
       <a list of 4 Text major ticklabel objects>)
```



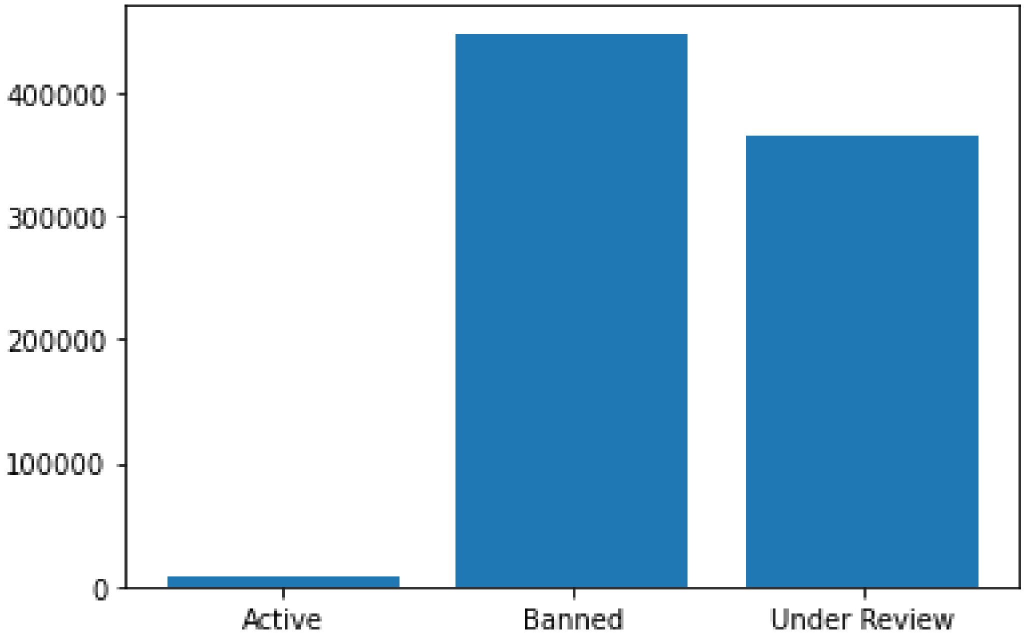
[21]:
```
# Create a histogram for each claim status and each author ban status.
data.groupby(['claim_status','author_ban_status']).size()

CB = {'Claims_Vs_Bans':['Claims: Active','Claims: Under Review', 'Claims:␣
↪Banned', 'Opinions: Active', 'Opinions: Under Review','Opinions: Banned'],␣
↪'Count':[6566,1603,1439,8817,463,196]}
#claims vs verification

plt.barh(CB['Claims_Vs_Bans'],CB['Count'])
#There are many times more authors making claims in the banned category than␣
↪the active category.
#There are noticeably more active authors providing opinions than making claims.
#A confidence interval would be necessary to prove this relationship is␣
↪significant, but it appears significant.
#There are also noticeably more authors under review with claims than with␣
↪opinions.
```

[21]:
```
<BarContainer object of 6 artists>
```



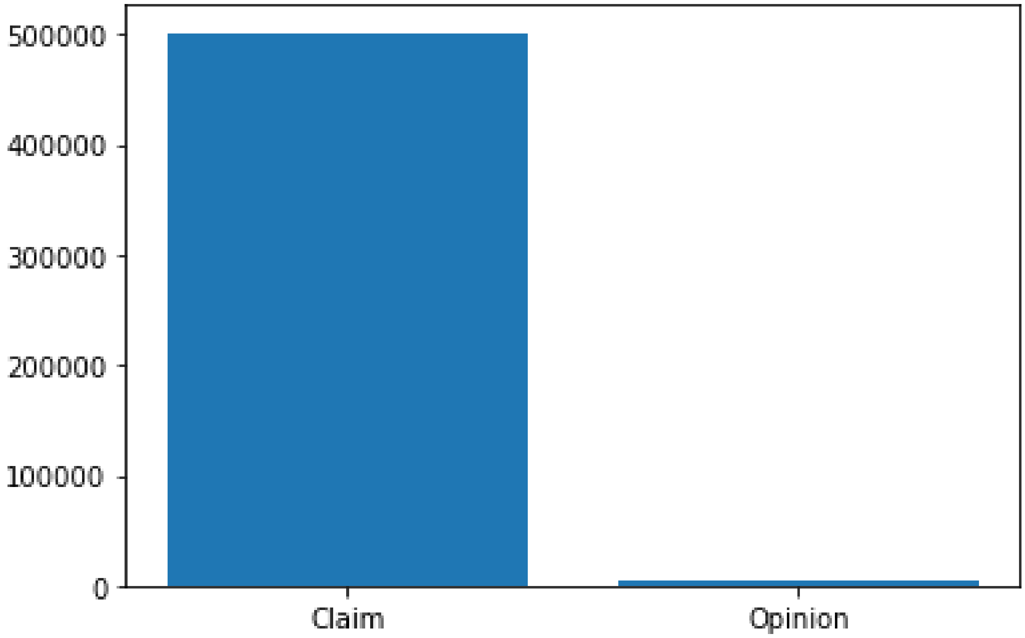
[22]:
```
#Create a bar plot with bars for each the median video views of each author ban␣
↪status.
data.groupby(['author_ban_status']).median()

BSV = {'Views_by_Ban_Status':['Active','Banned','Under Review'], 'Count':
↪[8616,448201,365245.5]}
#Ban Status Views
plt.bar(BSV['Views_by_Ban_Status'],BSV['Count'])
#View counts for non-active authors are drastically larger than for active␣
↪authors.
#Based on this insight, views might be a good indicator of claim status.
```

[22]:
```
<BarContainer object of 3 artists>
```



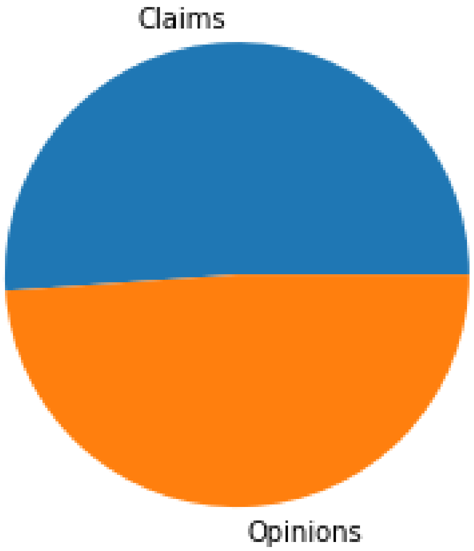
```
[23]: # Calculate the median view count for claim status.
      data.groupby(['claim_status']).median()

      CSV = {'Views_by_Claim_Status':['Claim','Opinion'], 'Count':[501555,4953]}
      #Ban Status Views
      plt.bar(CSV['Views_by_Claim_Status'],CSV['Count'])
      #The views of claims are astronomically larger than the views of opinions.
```

```
[23]: <BarContainer object of 2 artists>
```



[24]:
```
# Create a pie graph the depicts the proportions of total views for claim␣
↪videos and total views for opinion videos.
data.groupby(['claim_status']).size()
labels = 'Claims','Opinions'
sizes = [9908,9476]
plt.pie(sizes,labels=labels)
#While the total views for opinions and claims are similar
#The median of claims dwarfs that of opinions indicating that these two groups␣
↪might be skewed in opposite directions.
```

[24]:
```
([<matplotlib.patches.Wedge at 0x7f2b8b8c9c10>,
  <matplotlib.patches.Wedge at 0x7f2b8b9127d0>],
 [Text(-0.03850029594001087, 1.0993260331732946, 'Claims'),
  Text(0.03850029594001073, -1.0993260331732946, 'Opinions')])
```



Question: What do you notice about the overall view count for claim status?

1.0.4 Determine outliers

The ultimate objective of the TikTok project is to build a model that predicts whether a video is a claim or opinion.

Commonly some outliers might be 1.5 * IQR above the 3rd quartile.

The data is heavily skewed to the right so the outlier threshold is better represented by calculating the median value instead of the 3rd quartile for each variable and then adding 1.5 * IQR. This results in a threshold that is much lower than it would be if the 3rd quartile was used.

[25]:

```
import scipy.stats as sp

    #total number of values in the column:
#columncount = np.size(data['video_view_count'])
    #total of the values in the column:
#total = np.nansum(data['video_view_count'])
#print('Total:',total)
    #median of each column ignoring nans:
#med = np.nanmedian(data['video_view_count'])
#print('Total:',total,'Median:',med)
    #IQR:
    #q3:
#q3 = np.nanpercentile(data['video_view_count'], 75)
    #q1:
#q1 = np.nanpercentile(data['video_view_count'], 25)
```

```
#iqr = q3 - q1
#print('Total:',total,'Median:',med,"IQR:",iqr)
    #The value of the first likely outlier:
#outlierlimit = med + 1.5 * iqr
#print('Total:',total,'Median:',med,"IQR:",iqr,"Outlier Limit:",outlierlimit)
    #Percentile of the first likely outlier:
#outlierpercentile = sp.percentileofscore(data['video_view_count'],␣
 ↪outlierlimit)
#print('Total:',total,'Median:',med,"IQR:",iqr,"Outlier Limit:
 ↪",outlierlimit,"Outlier Percentile:",outlierpercentile)
    #Datapoint of the first likely outlier:
#outliervalue = columncount * outlierpercentile / 100
#print('Total:',total,'Median:',med,"IQR:",iqr,"Outlier Limit:
 ↪",outlierlimit,"Outlier Value:",outliervalue)
    #Number of points that are likely to be outliers:
#totaloutliers = columncount - outliervalue
#print('Total:',total,'Median:',med,"IQR:",iqr,"Outlier Limit:
 ↪",outlierlimit,"Outlier Value:",outliervalue,"Total Outliers:",totaloutliers)

from pandas.api.types import is_numeric_dtype
for column in data:
    if is_numeric_dtype(data[column]) is True:
        print('Number of outliers,',column,':',np.size(data[column]) - np.
 ↪size(data[column]) * sp.percentileofscore(data[column], (np.
 ↪nanmedian(data[column]) + 1.5 * ( np.nanpercentile(data[column], 75) - np.
 ↪nanpercentile(data[column], 25) ) ) ) / 100 )
```

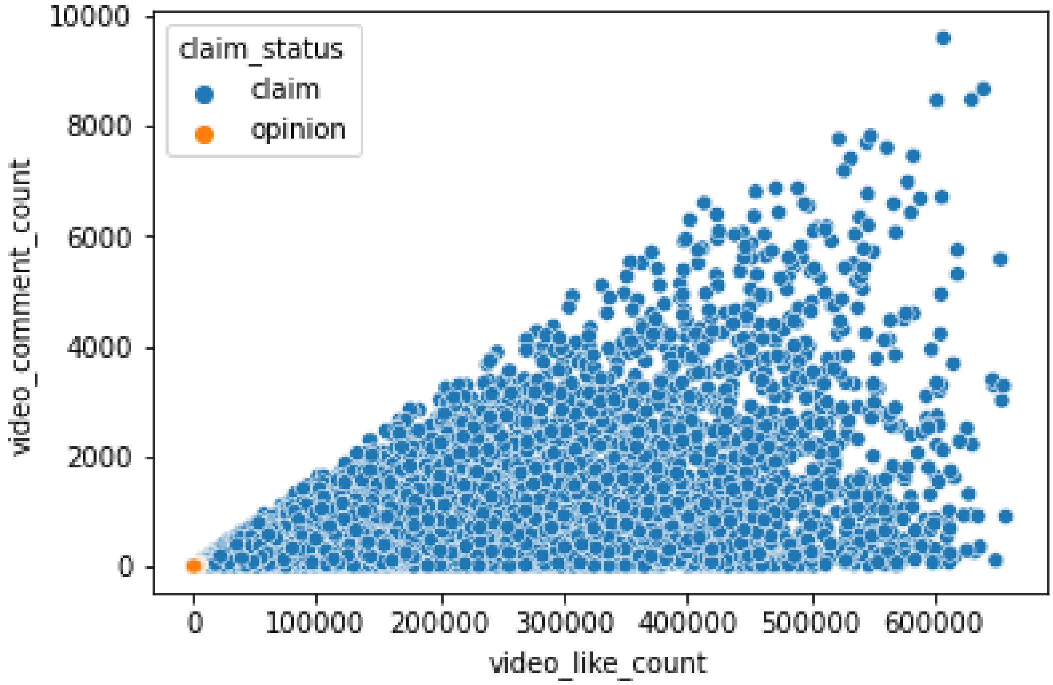
```
Number of outliers, # : 0.0
Number of outliers, video_id : 0.0
Number of outliers, video_duration_sec : 0.0
Number of outliers, video_view_count : 2641.0
Number of outliers, video_like_count : 3766.0
Number of outliers, video_share_count : 4030.0
Number of outliers, video_download_count : 4031.0
Number of outliers, video_comment_count : 4180.0
```

[26]:

```
# Create a scatterplot of `video_like_count` versus `video_comment_count`␣
 ↪according to 'claim_status'
sb.scatterplot(data=data, x=data.video_like_count, y=data.video_comment_count,␣
 ↪hue=data.claim_status)
```

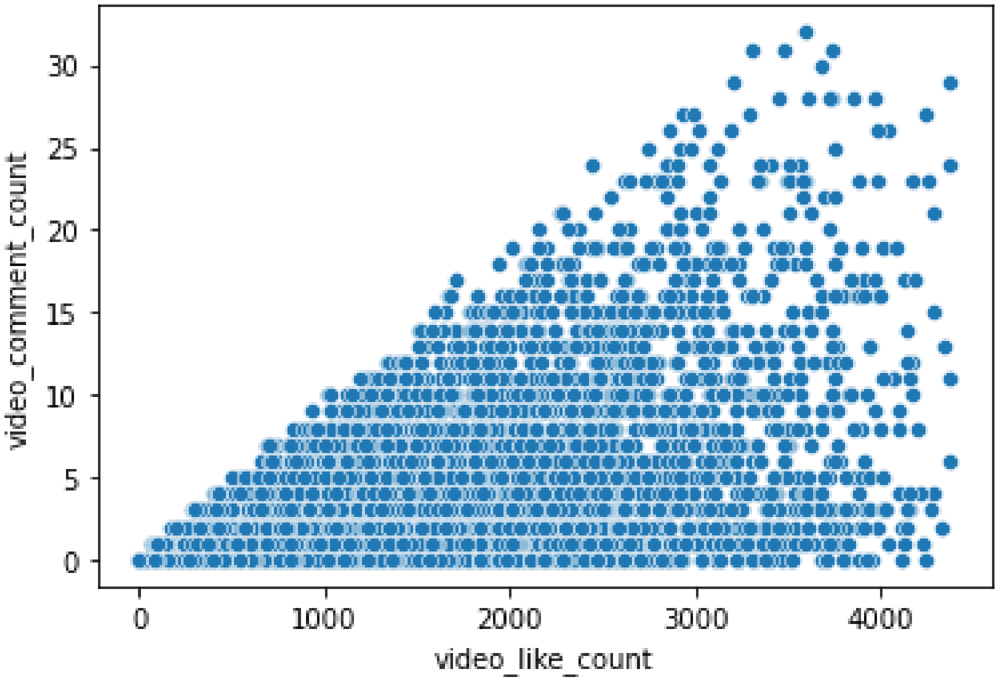
[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2b8b8f0690>
```



[27]:
```
# Create a scatterplot of `video_like_count` versus `video_comment_count` for␣
↪opinions only
opinion_mask = data[data.claim_status == 'opinion']
sb.scatterplot(data=data, x=opinion_mask.video_like_count, y=opinion_mask.
↪video_comment_count)
```

[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2b8b8aa490>
```



Create a scatterplot in Tableau Public. https://public.tableau.com/views/TikTokEDA_16890178963150/TikTokE
US&:display_count=n&:origin=viz_share_link

1.0.5 Results and evaluation

Every engagement characteristic of these videos is heavily skewed to the right and each category has between 2000-4000 outliers that could be removed in order to make the data more reliable. This is very significant as even 2000/19382 values is over 10% of all of the data values in this dataset. The number of opinions is very small among this dataset and almost solely occurs among active users.