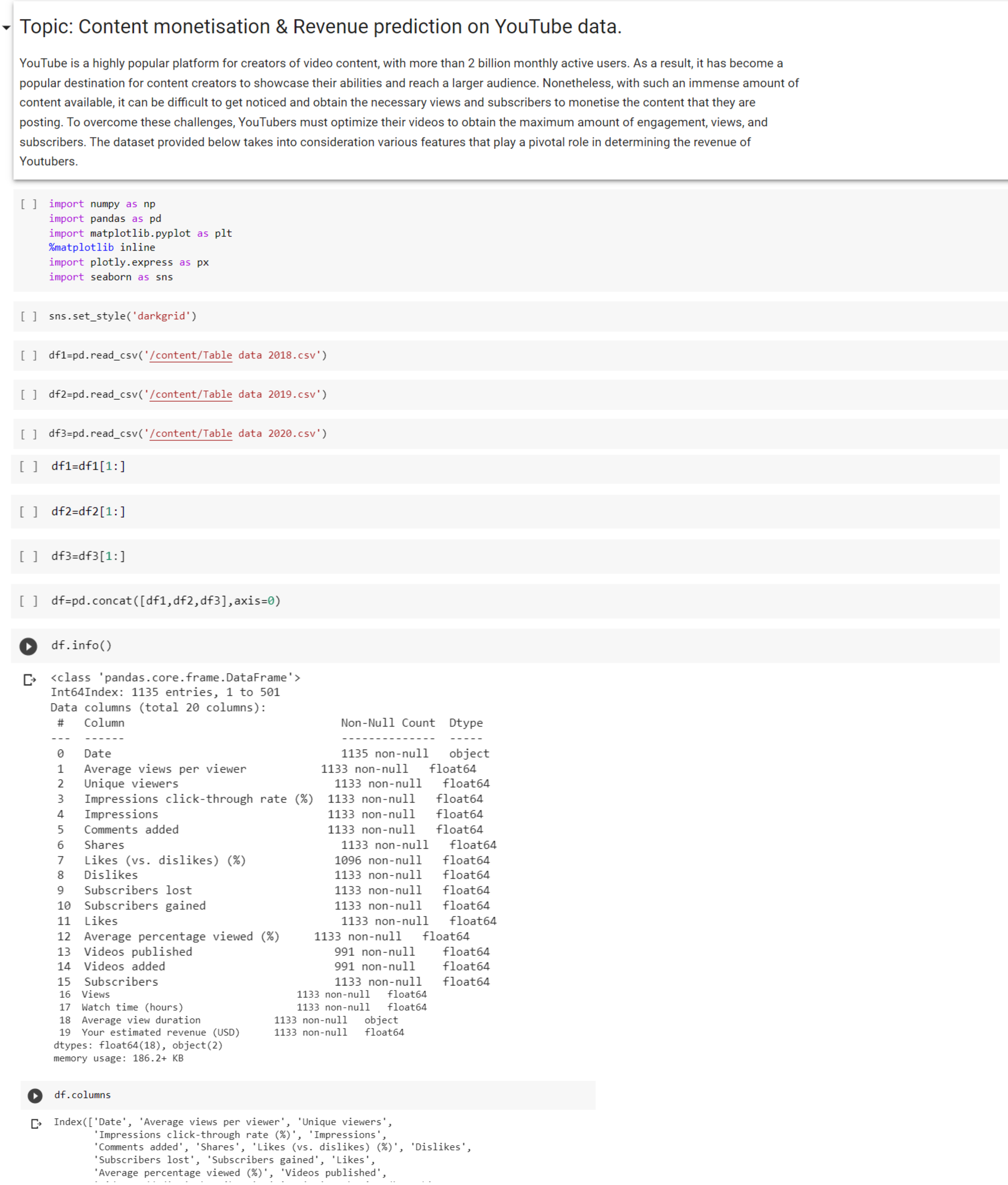
Bhuvi Ghosh
60009210191

Colab link:

https://colab.research.google.com/drive/1XK2f3vBINhJI9nW5d2VMgrIy4TygpHRK?usp=sharing



Topic: Content monetisation & Revenue prediction on YouTube data.

YouTube is a highly popular platform for creators of video content, with more than 2 billion monthly active users. As a result, it has become a popular destination for content creators to showcase their abilities and reach a larger audience. Nonetheless, with such an immense amount of content available, it can be difficult to get noticed and obtain the necessary views and subscribers to monetise the content that they are posting. To overcome these challenges, YouTubers must optimize their videos to obtain the maximum amount of engagement, views, and subscribers. The dataset provided below takes into consideration various features that play a pivotal role in determining the revenue of Youtubers.

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    %matplotlib inline
    import plotly.express as px
    import seaborn as sns
```

```
[ ] sns.set_style('darkgrid')
```

```
[ ] df1=pd.read_csv('/content/Table data 2018.csv')
```

```
[ ] df2=pd.read_csv('/content/Table data 2019.csv')
```

```
[ ] df3=pd.read_csv('/content/Table data 2020.csv')
```

```
[ ] df1=df1[1:]
```

```
[ ] df2=df2[1:]
```

```
[ ] df3=df3[1:]
```

```
[ ] df=pd.concat([df1,df2,df3],axis=0)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1135 entries, 1 to 501
Data columns (total 20 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Date                                1135 non-null   object
 1   Average views per viewer            1133 non-null   float64
 2   Unique viewers                      1133 non-null   float64
 3   Impressions click-through rate (%)  1133 non-null   float64
 4   Impressions                         1133 non-null   float64
 5   Comments added                      1133 non-null   float64
 6   Shares                              1133 non-null   float64
 7   Likes (vs. dislikes) (%)            1096 non-null   float64
 8   Dislikes                            1133 non-null   float64
 9   Subscribers lost                    1133 non-null   float64
 10  Subscribers gained                  1133 non-null   float64
 11  Likes                               1133 non-null   float64
 12  Average percentage viewed (%)       1133 non-null   float64
 13  Videos published                    991 non-null    float64
 14  Videos added                        991 non-null    float64
 15  Subscribers                         1133 non-null   float64
 16  Views                               1133 non-null   float64
 17  Watch time (hours)                  1133 non-null   float64
 18  Average view duration               1133 non-null   object
 19  Your estimated revenue (USD)        1133 non-null   float64
dtypes: float64(18), object(2)
memory usage: 186.2+ KB
```

```
df.columns
```

```
Index(['Date', 'Average views per viewer', 'Unique viewers',
       'Impressions click-through rate (%)', 'Impressions',
       'Comments added', 'Shares', 'Likes (vs. dislikes) (%)', 'Dislikes',
       'Subscribers lost', 'Subscribers gained', 'Likes',
       'Average percentage viewed (%)', 'Videos published',
columns=df.select_dtypes(include=['int','float']).columns
columns=columns.tolist()
```

Dropping duplicate values:

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

Checking is null values exist & dropping them in case they are in a minor percentage:

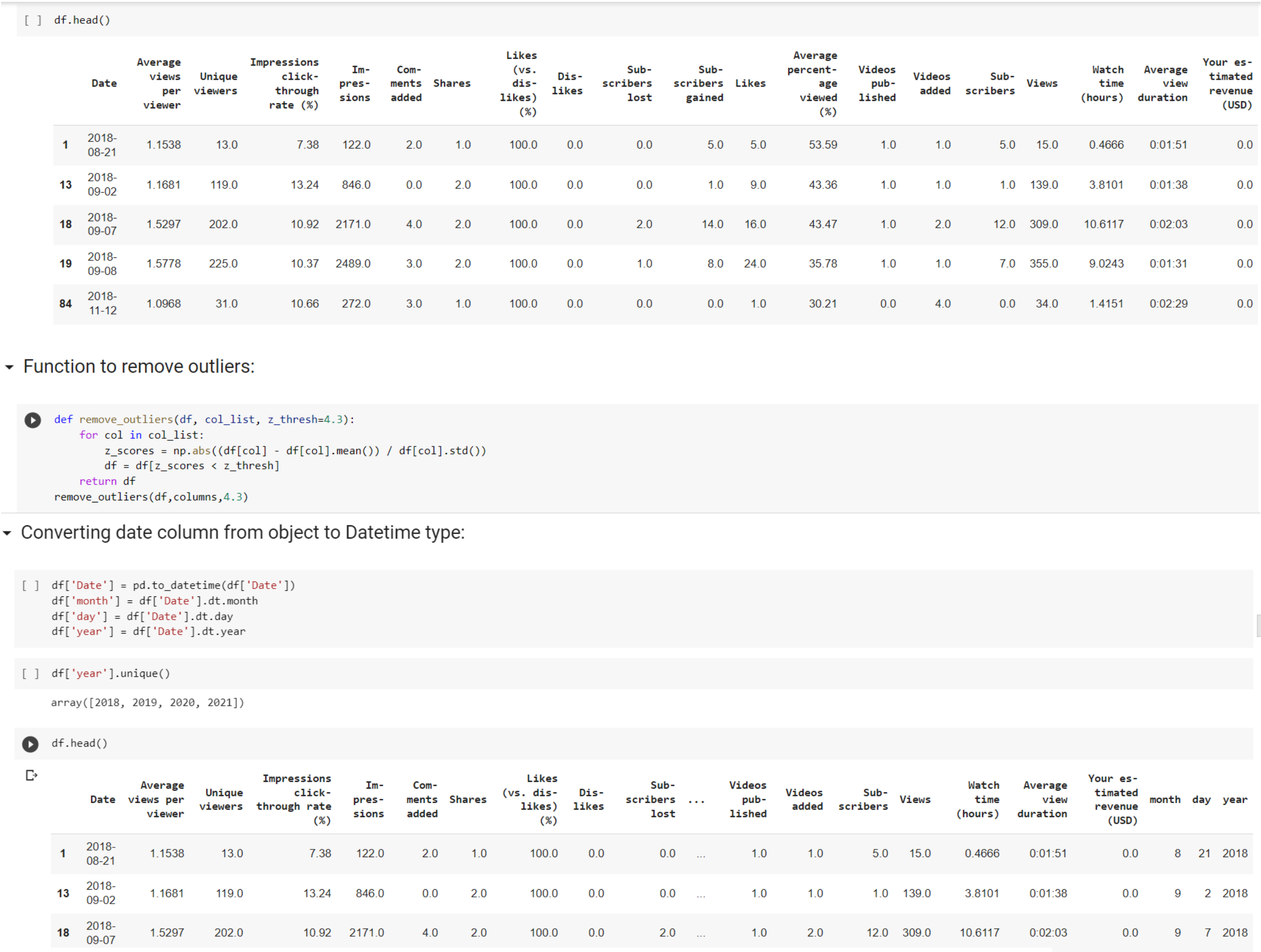
```
df.isnull().sum()/len(df)
```

```
Date                                  0.000000
Average views per viewer              0.000882
Unique viewers                        0.000882
Impressions click-through rate (%)    0.000882
Impressions                           0.000882
Comments added                        0.000882
Shares                                0.000882
Likes (vs. dislikes) (%)              0.033510
Dislikes                              0.000882
Subscribers lost                      0.000882
Subscribers gained                    0.000882
Likes                                 0.000882
Average percentage viewed (%)         0.000882
Videos published                      0.126102
Videos added                          0.126102
Subscribers                           0.000882
Views                                 0.000882
Watch time (hours)                    0.000882
Average view duration                 0.000882
Your estimated revenue (USD)          0.000882
dtype: float64
```

```
df.dropna(inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 988 entries, 1 to 500
Data columns (total 20 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Date                                988 non-null    object 
 1   Average views per viewer            988 non-null    float64
 2   Unique viewers                      988 non-null    float64
 3   Impressions click-through rate (%)  988 non-null    float64
 4   Impressions                         988 non-null    float64
 5   Comments added                      988 non-null    float64
 6   Shares                              988 non-null    float64
 7   Likes (vs. dislikes) (%)            988 non-null    float64
 8   Dislikes                            988 non-null    float64
 9   Subscribers lost                    988 non-null    float64
 10  Subscribers gained                  988 non-null    float64
 11  Likes                               988 non-null    float64
 12  Average percentage viewed (%)       988 non-null    float64
 13  Videos published                    988 non-null    float64
 14  Videos added                        988 non-null    float64
 15  Subscribers                         988 non-null    float64
 16  Views                               988 non-null    float64
 17  Watch time (hours)                  988 non-null    float64
 18  Average view duration               988 non-null    object 
 19  Your estimated revenue (USD)        988 non-null    float64
dtypes: float64(18), object(2)
```



```
[ ] df.head()
```

	Date	Average views per viewer	Unique viewers	Impressions click-through rate (%)	Impressions	Comments added	Shares	Likes (vs. dislikes) (%)	Dislikes	Subscribers lost	Subscribers gained	Likes	Average percentage viewed (%)	Videos published	Videos added	Subscribers	Views	Watch time (hours)	Average view duration	Your estimated revenue (USD)
1	2018-08-21	1.1538	13.0	7.38	122.0	2.0	1.0	100.0	0.0	0.0	5.0	5.0	53.59	1.0	1.0	5.0	15.0	0.4666	0:01:51	0.0
13	2018-09-02	1.1681	119.0	13.24	846.0	0.0	2.0	100.0	0.0	0.0	1.0	9.0	43.36	1.0	1.0	1.0	139.0	3.8101	0:01:38	0.0
18	2018-09-07	1.5297	202.0	10.92	2171.0	4.0	2.0	100.0	0.0	2.0	14.0	16.0	43.47	1.0	2.0	12.0	309.0	10.6117	0:02:03	0.0
19	2018-09-08	1.5778	225.0	10.37	2489.0	3.0	2.0	100.0	0.0	1.0	8.0	24.0	35.78	1.0	1.0	7.0	355.0	9.0243	0:01:31	0.0
84	2018-11-12	1.0968	31.0	10.66	272.0	3.0	1.0	100.0	0.0	0.0	0.0	1.0	30.21	0.0	4.0	0.0	34.0	1.4151	0:02:29	0.0

Function to remove outliers:

```
def remove_outliers(df, col_list, z_thresh=4.3):
    for col in col_list:
        z_scores = np.abs((df[col] - df[col].mean()) / df[col].std())
        df = df[z_scores < z_thresh]
    return df
remove_outliers(df,columns,4.3)
```

Converting date column from object to Datetime type:

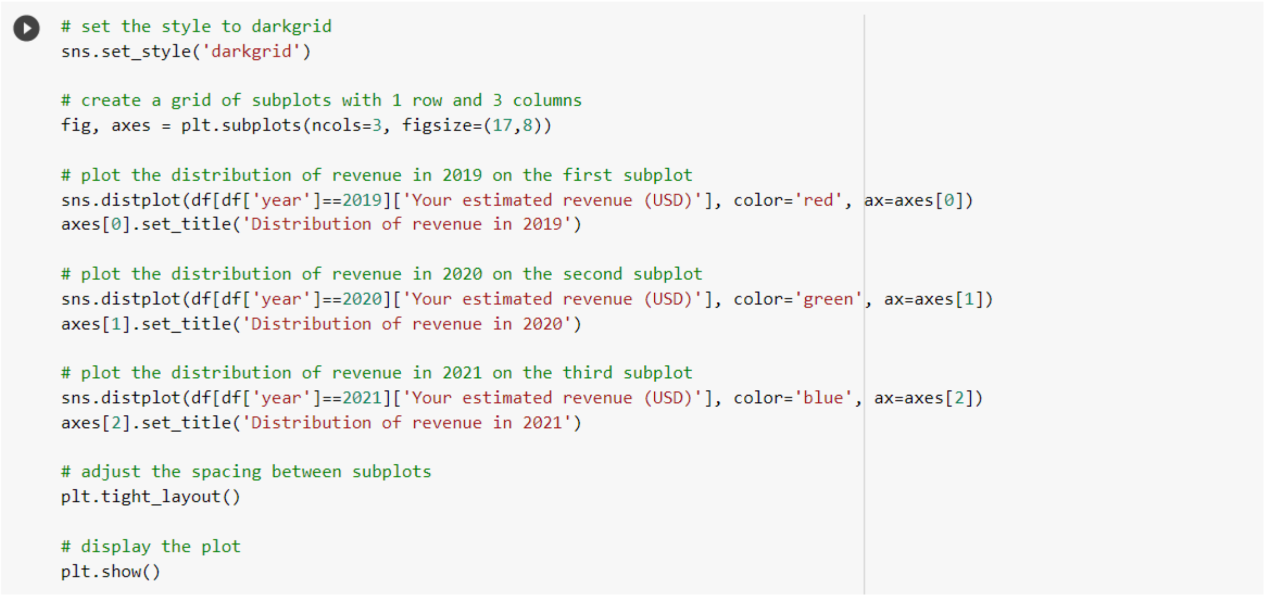
```
[ ] df['Date'] = pd.to_datetime(df['Date'])
    df['month'] = df['Date'].dt.month
    df['day'] = df['Date'].dt.day
    df['year'] = df['Date'].dt.year
```

```
[ ] df['year'].unique()

    array([2018, 2019, 2020, 2021])
```

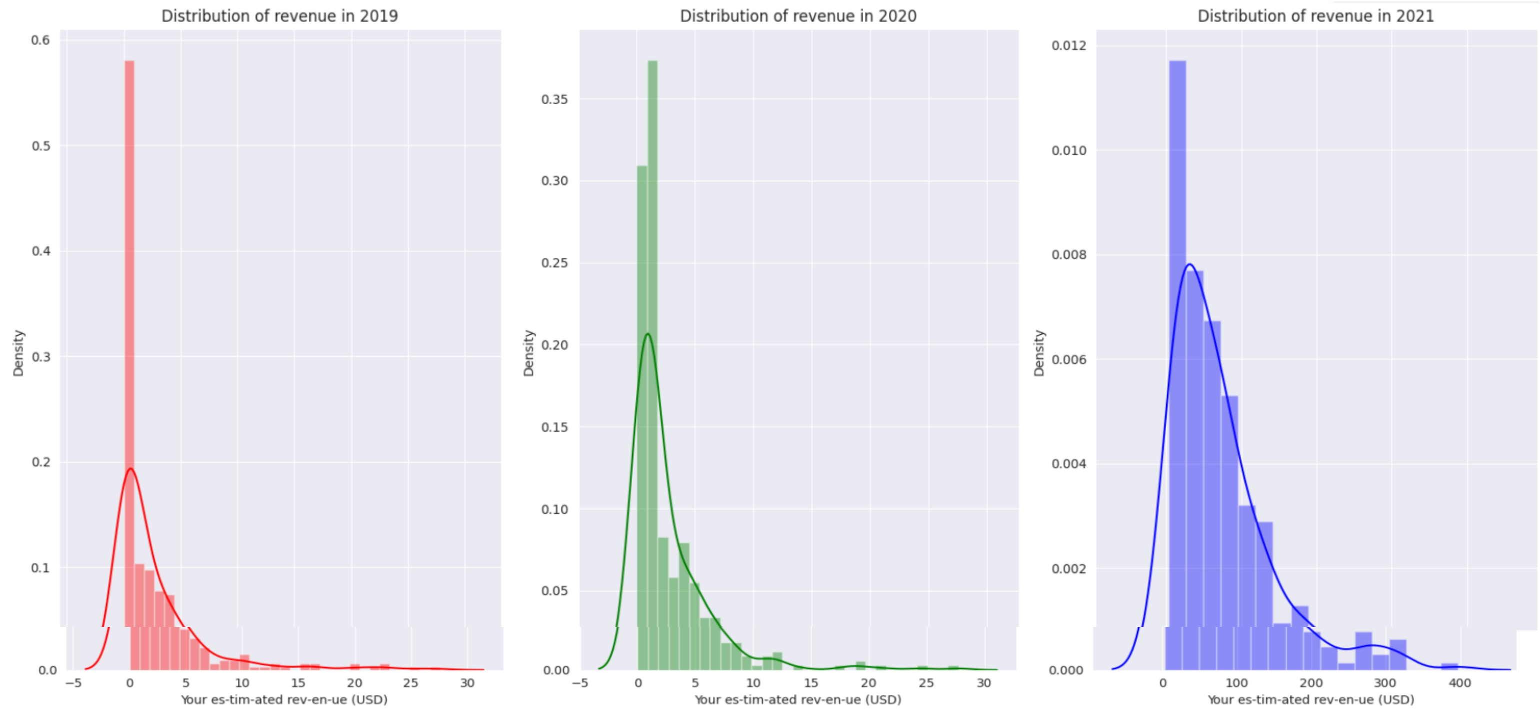
```
df.head()
```

	Date	Average views per viewer	Unique viewers	Impressions click-through rate (%)	Impressions	Comments added	Shares	Likes (vs. dislikes) (%)	Dislikes	Subscribers lost	...	Videos published	Videos added	Subscribers	Views	Watch time (hours)	Average view duration	Your estimated revenue (USD)	month	day	year
1	2018-08-21	1.1538	13.0	7.38	122.0	2.0	1.0	100.0	0.0	0.0	...	1.0	1.0	5.0	15.0	0.4666	0:01:51	0.0	8	21	2018
13	2018-09-02	1.1681	119.0	13.24	846.0	0.0	2.0	100.0	0.0	0.0	...	1.0	1.0	1.0	139.0	3.8101	0:01:38	0.0	9	2	2018
18	2018-09-07	1.5297	202.0	10.92	2171.0	4.0	2.0	100.0	0.0	2.0	...	1.0	2.0	12.0	309.0	10.6117	0:02:03	0.0	9	7	2018

Checking the distribution of the output variable:



```
# set the style to darkgrid
sns.set_style('darkgrid')

# create a grid of subplots with 1 row and 3 columns
fig, axes = plt.subplots(ncols=3, figsize=(17,8))

# plot the distribution of revenue in 2019 on the first subplot
sns.distplot(df[df['year']==2019]['Your estimated revenue (USD)'], color='red', ax=axes[0])
axes[0].set_title('Distribution of revenue in 2019')

# plot the distribution of revenue in 2020 on the second subplot
sns.distplot(df[df['year']==2020]['Your estimated revenue (USD)'], color='green', ax=axes[1])
axes[1].set_title('Distribution of revenue in 2020')

# plot the distribution of revenue in 2021 on the third subplot
sns.distplot(df[df['year']==2021]['Your estimated revenue (USD)'], color='blue', ax=axes[2])
axes[2].set_title('Distribution of revenue in 2021')

# adjust the spacing between subplots
plt.tight_layout()

# display the plot
plt.show()
```

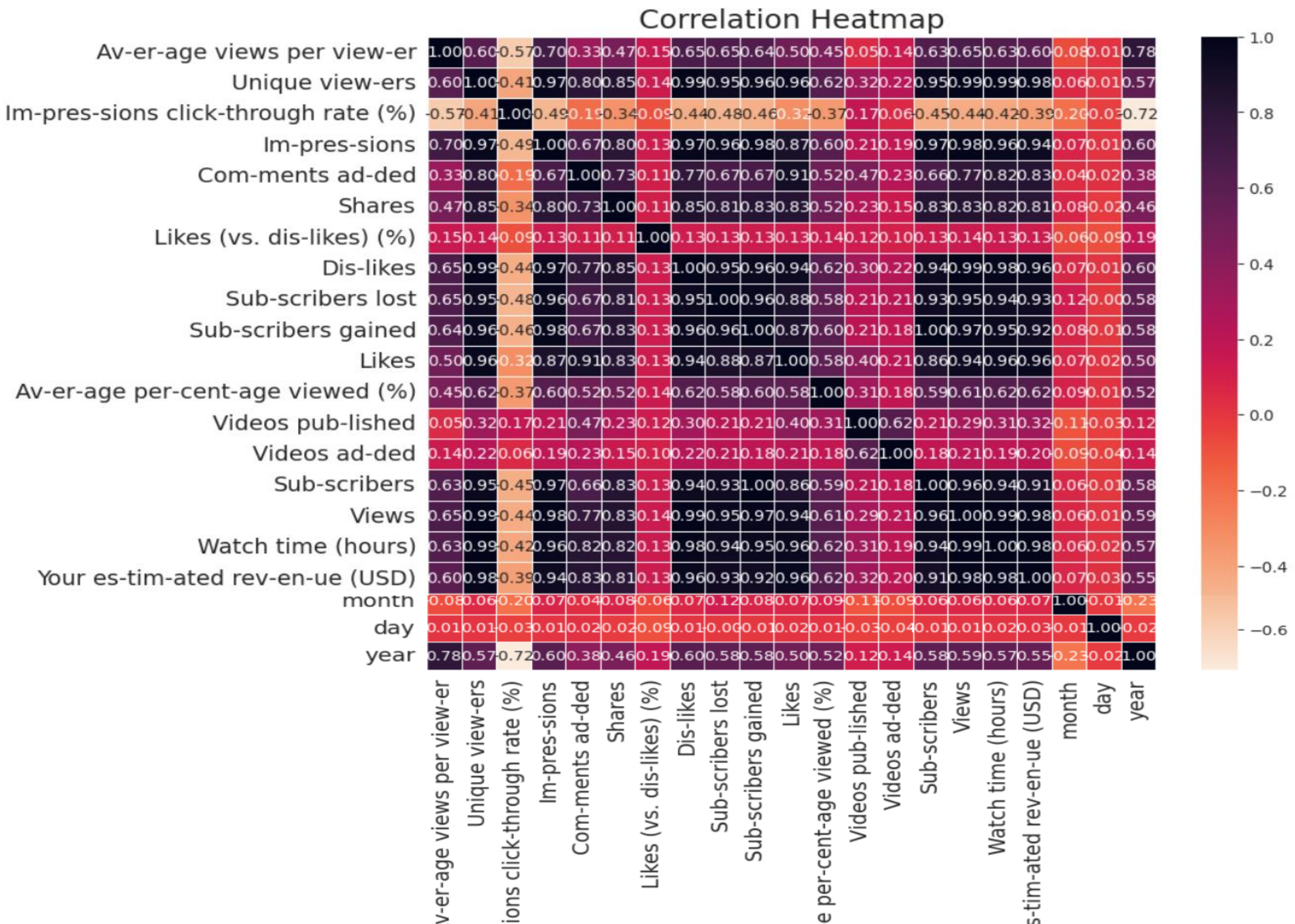


It is right skewed for the three given years.

Finding correlation between the variables:

```
[ ]  fig, ax = plt.subplots(figsize=(10, 8))
     sns.heatmap(df.corr(), cmap="rocket_r", annot=True, linewidths=0.5, fmt=".2f", ax=ax)
     ax.set_title("Correlation Heatmap", fontsize=18)
     ax.tick_params(labelsize=14)
     sns.despine(ax=ax, top=True, bottom=True)
     plt.show()
```

```
<ipython-input-669-e50ef237404e>:2: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence
```

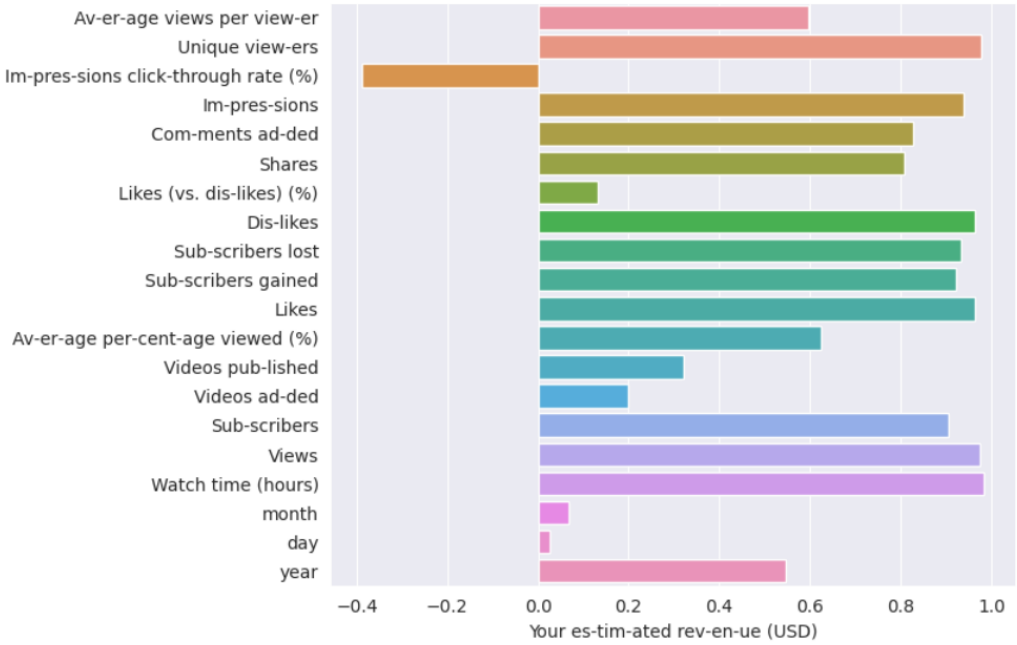


```
[ ] corr_matrix = df.corr()
    corr_values = corr_matrix['Your estimated revenue (USD)'].drop('Your estimated revenue (USD)')
```

```
<ipython-input-670-57ea96d97aac>:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to si
```

Finding the correlation between the target attribute & the other attributes:

```
plt.figure(figsize=(7,6))
sns.barplot(y=corr_values.index,x=corr_values)
plt.show()
```



```
[ ] def get_correlated_attributes(df, target_var, threshold):
        # get the correlation matrix
        corr_matrix = df.corr()
        # get the correlation values for the target variable
        corr_values = corr_matrix[target_var].drop(target_var)
        # filter the attributes by the correlation threshold
        correlated_attrs = corr_values[corr_values >= threshold].index
        # return the list of correlated attributes
        return correlated_attrs
```

```
[ ] z_pos=get_correlated_attributes(df,'Your estimated revenue (USD)',0.6)
```

```
<ipython-input-672-054fbbdc8730>:3: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify t
```

```
[ ] print(z_pos)
```

```
Index(['Unique viewers', 'Impressions', 'Comments added', 'Shares',
       'Dislikes', 'Subscribers lost', 'Subscribers gained', 'Likes',
       'Average percentage viewed (%)', 'Subscribers', 'Views',
       'Watch time (hours)'],
      dtype='object')
```

```
def get_correlated_attributes(df, target_var, threshold):
    # get the correlation matrix
    corr_matrix = df.corr()
    # get the correlation values for the target variable
    corr_values = corr_matrix[target_var].drop(target_var)
```

```
[ ]     # filter the attributes by the correlation threshold
        correlated_attrs = corr_values[corr_values <= threshold].index
        # return the list of correlated attributes
        return correlated_attrs
```

```
[ ] z_neg=get_correlated_attributes(df,'Your estimated revenue (USD)',-0.6)
```

```
<ipython-input-675-659e8b10d090>:3: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of
```

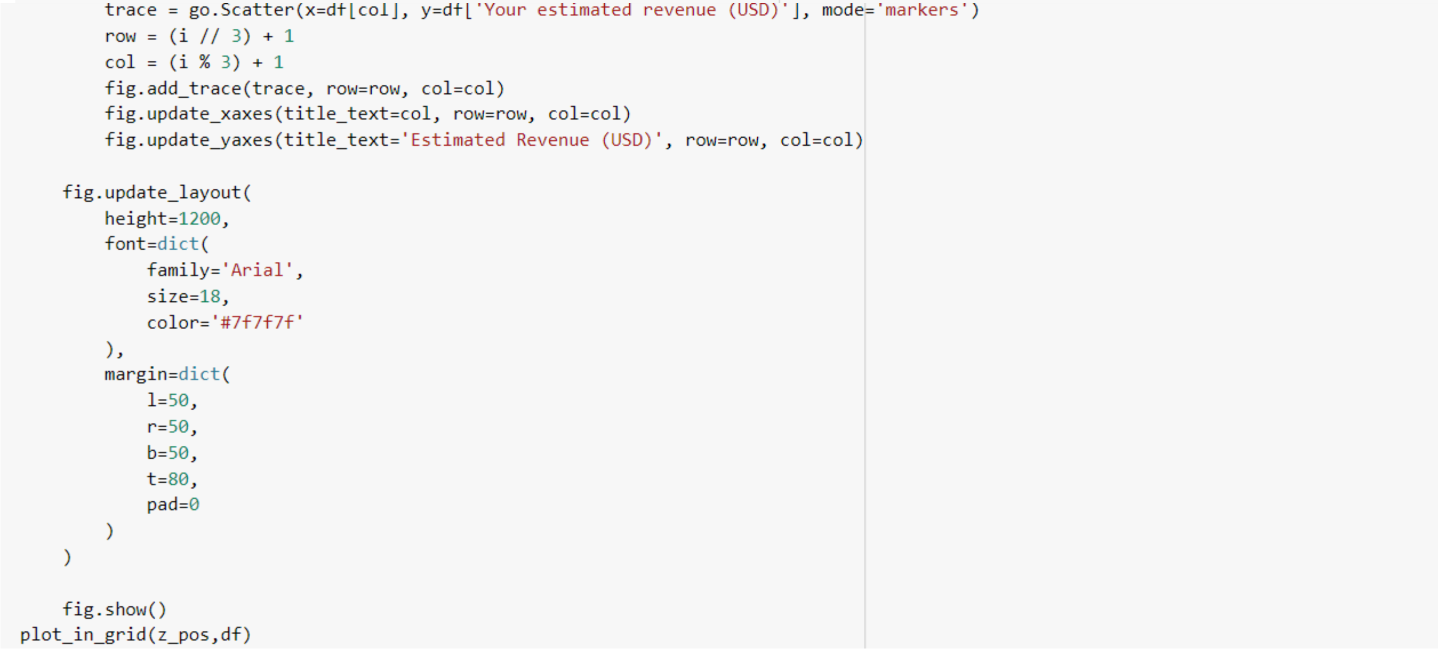
```
[ ] z_neg
```

```
Index([], dtype='object')
```

```
[ ] import plotly.graph_objects as go
    from plotly.subplots import make_subplots
    fig = go.Figure(data=go.Scatter(x=df['Shares'], y=df['Your estimated revenue (USD)'], mode='markers'))
```

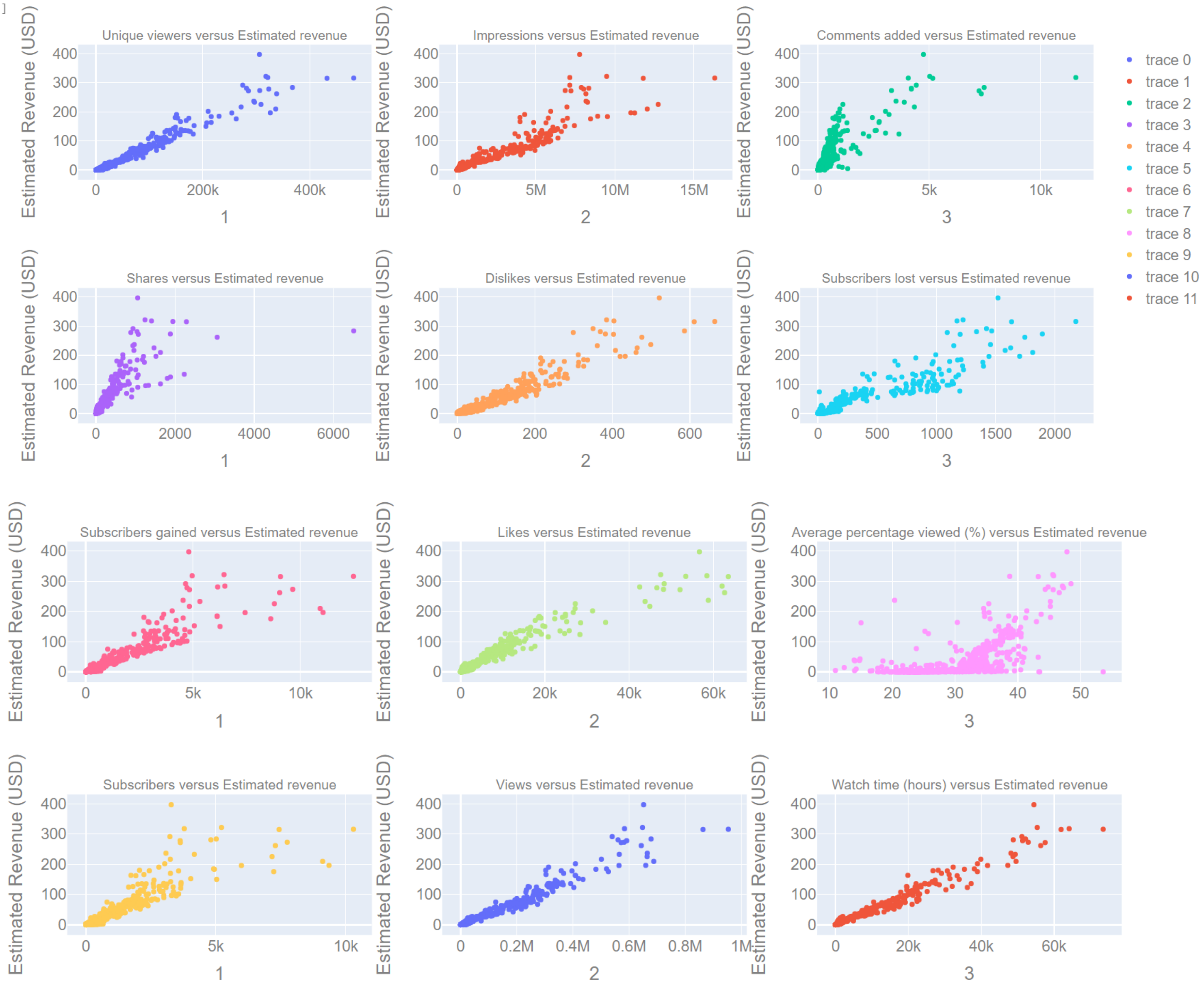
Is the target variable linearly dependent on the most important features? Let's check out how these features affect the target variable.

```
def plot_in_grid(z, df):
    fig = make_subplots(rows=4, cols=3, subplot_titles=z + ' versus Estimated revenue')
```



```
        trace = go.Scatter(x=df[col], y=df['Your estimated revenue (USD)'], mode='markers')
        row = (i // 3) + 1
        col = (i % 3) + 1
        fig.add_trace(trace, row=row, col=col)
        fig.update_xaxes(title_text=col, row=row, col=col)
        fig.update_yaxes(title_text='Estimated Revenue (USD)', row=row, col=col)

    fig.update_layout(
        height=1200,
        font=dict(
            family='Arial',
            size=18,
            color='#7f7f7f'
        ),
        margin=dict(
            l=50,
            r=50,
            b=50,
            t=80,
            pad=0
        )
    )

    fig.show()
plot_in_grid(z_pos,df)
```

[ ]



Unique viewers versus Estimated revenue

Impressions versus Estimated revenue

Comments added versus Estimated revenue

Shares versus Estimated revenue

Dislikes versus Estimated revenue

Subscribers lost versus Estimated revenue

Subscribers gained versus Estimated revenue

Likes versus Estimated revenue

Average percentage viewed (%) versus Estimated revenue

Subscribers versus Estimated revenue

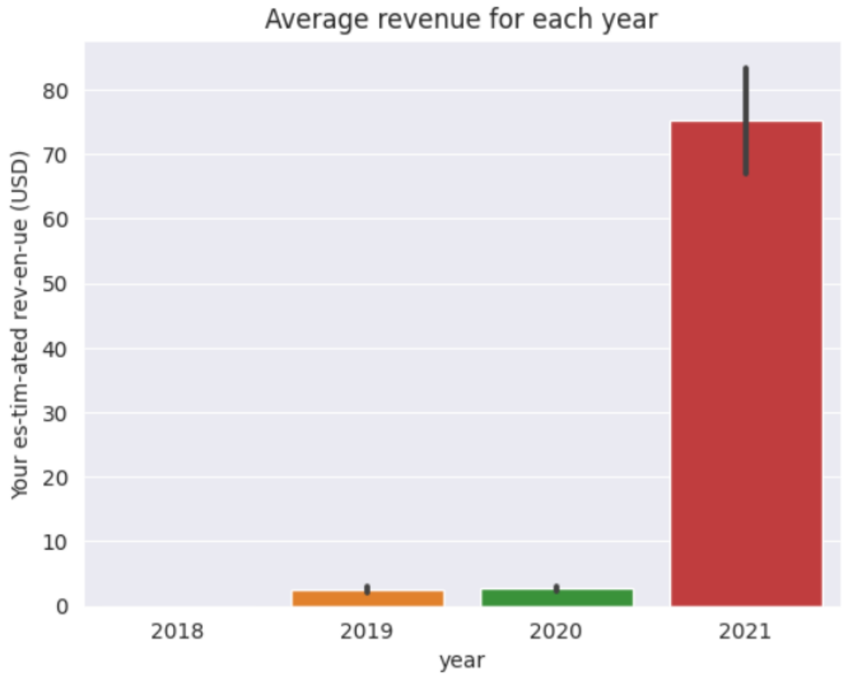
Views versus Estimated revenue

Watch time (hours) versus Estimated revenue

The revenue linearly increases with an increase in unique viewers, subscribers, subscribers gained, likes,dislikees, views, shares & impressions. However there is a non-linear change in the estimated revenue with respected to the percentage viewed going upto 400 illion dollars.

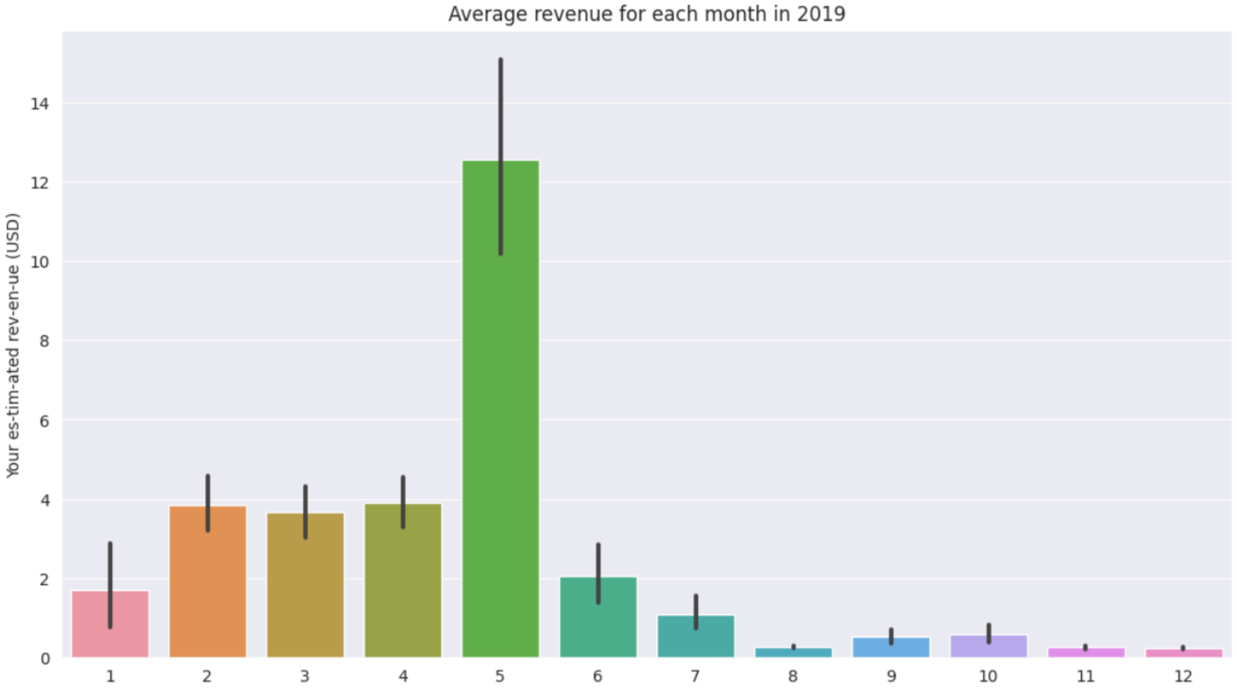
Average revenue per year:

```
[ ]  sns.barplot(x='year',y='Your estimated revenue (USD)',data=df)
     plt.title('Average revenue for each year')
     plt.show()
```



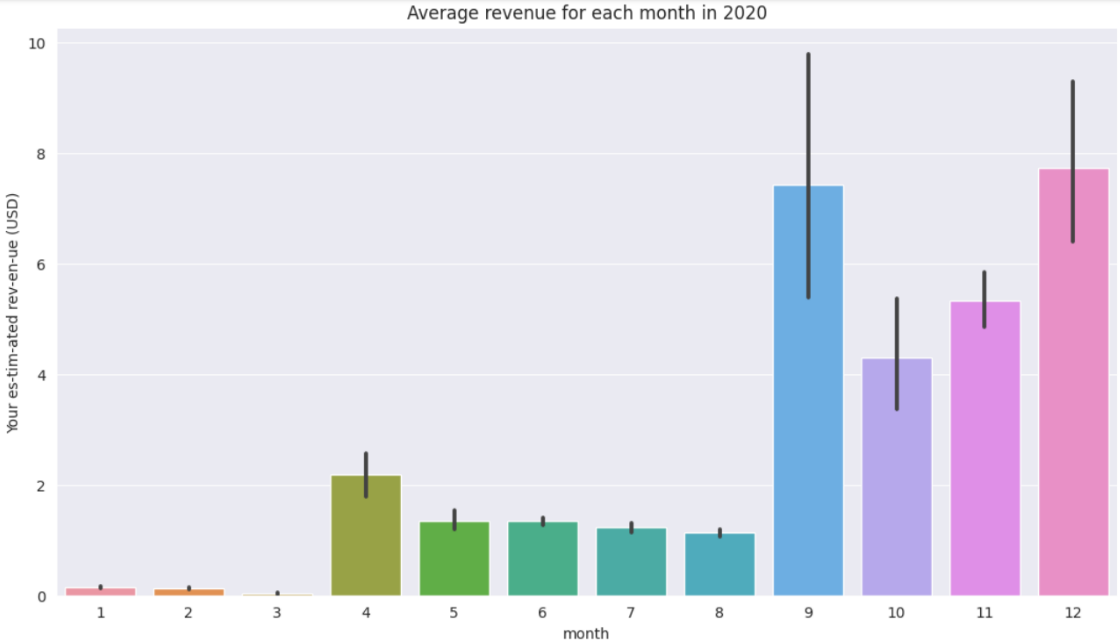
```
df1=df[df['year']==2021]
df2=df[df['year']==2020]
df3=df[df['year']==2018]
df4=df[df['year']==2019]
```

```
plt.figure(figsize=(13,7))
sns.barplot(x='month',y='Your estimated revenue (USD)',data=df4)
plt.title('Average revenue for each month in 2019')
plt.show()
```



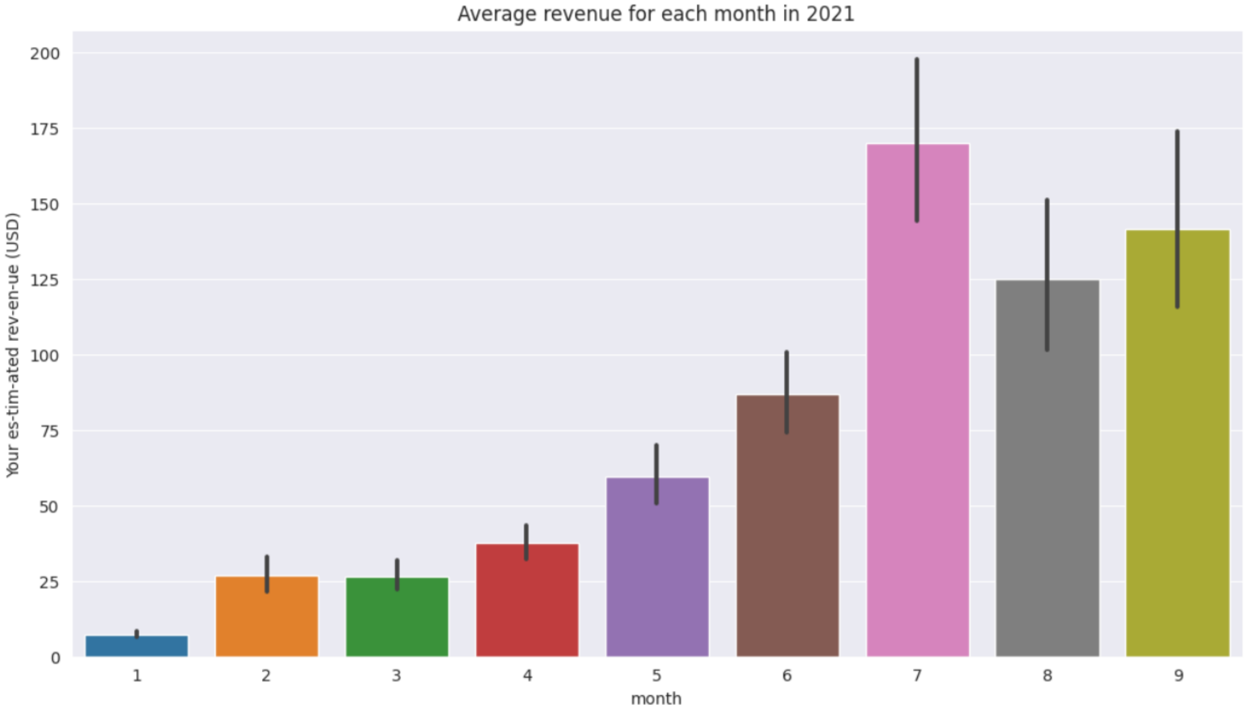
December had the maximum revenue earning during the year 2020 & March saw the least amount of earning.

```
plt.figure(figsize=(13,7))
sns.barplot(x='month',y='Your estimated revenue (USD)',data=df1)
plt.title('Average revenue for each month in 2021')
plt.show()
```



May had the maximum revenue earning during the year 2019 & December saw the least amount of earning.

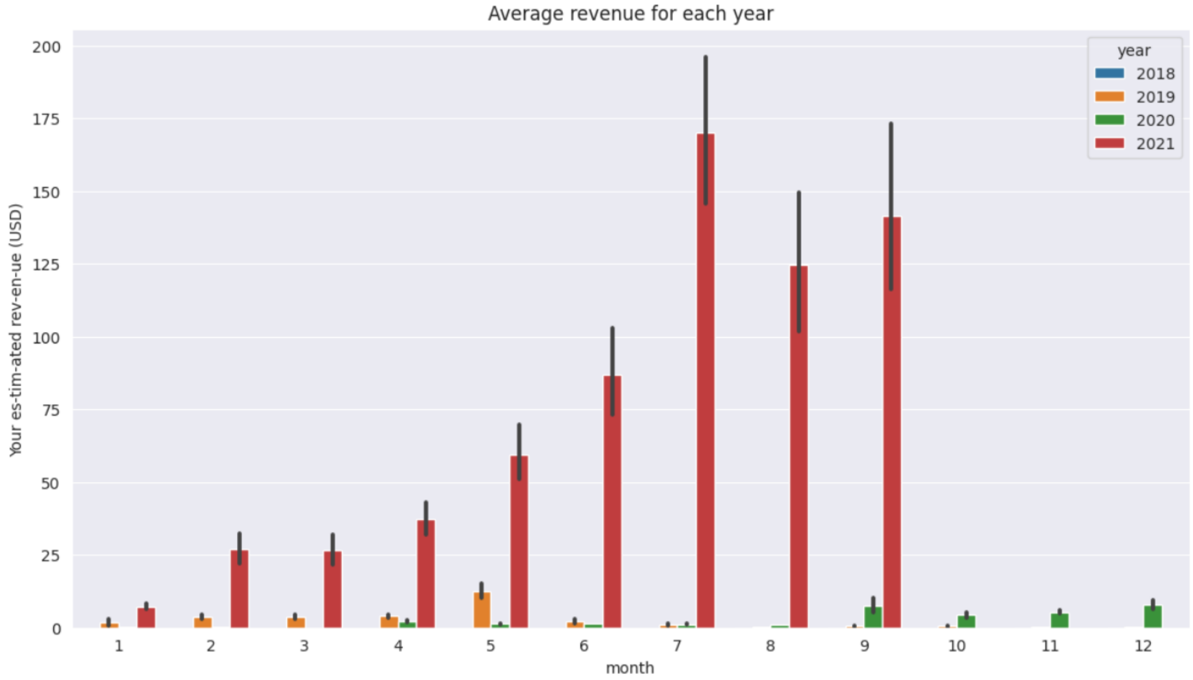
```
[ ] plt.figure(figsize=(13,7))
    sns.barplot(x='month',y='Your estimated revenue (USD)',data=df2)
    plt.title('Average revenue for each month in 2020')
    plt.show()
```



July had the maximum revenue earning during the year 2021 & January saw the least amount of earning.

Monthly comparison of average revenue of the years:

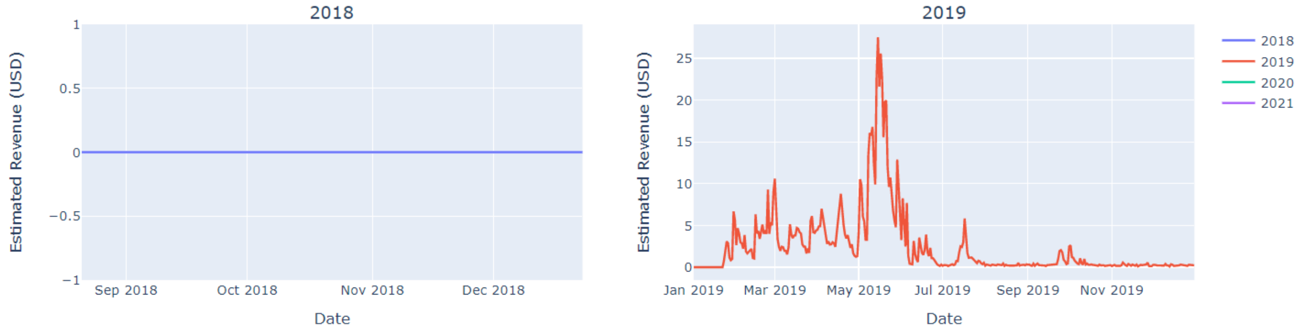
```
[ ]  plt.figure(figsize=(13,7))
     sns.barplot(x='month',y='Your estimated revenue (USD)',hue='year',data=df)
     plt.title('Average revenue for each year')
     plt.show()
```

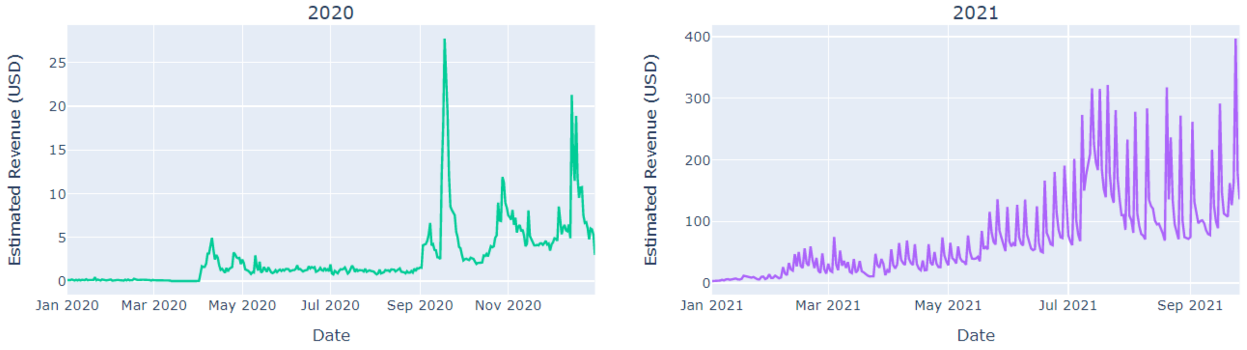


Revenue trends over the years:

```
[ ]  def plot_in_grid(df1, df2, df3, df4):
         fig = make_subplots(rows=2, cols=2,
                             subplot_titles=('2018','2019','2020','2021'))

         fig.add_trace(go.Scatter(x=df3['Date'], y=df3['Your estimated revenue (USD)'],
                                  mode='lines', name='2018'), row=1, col=1)

         fig.add_trace(go.Scatter(x=df4['Date'], y=df4['Your estimated revenue (USD)'],
                                  mode='lines', name='2019'), row=1, col=2)

         fig.add_trace(go.Scatter(x=df2['Date'], y=df2['Your estimated revenue (USD)'],
                                  mode='lines', name='2020'), row=2, col=1)

         fig.add_trace(go.Scatter(x=df1['Date'], y=df1['Your estimated revenue (USD)'],
                                  mode='lines', name='2021'), row=2, col=2)

         fig.update_xaxes(title_text='Date', row=1, col=1)
         fig.update_xaxes(title_text='Date', row=1, col=2)
         fig.update_xaxes(title_text='Date', row=2, col=1)
         fig.update_xaxes(title_text='Date', row=2, col=2)

         fig.update_yaxes(title_text='Estimated Revenue (USD)', row=1, col=1)
         fig.update_yaxes(title_text='Estimated Revenue (USD)', row=1, col=2)
         fig.update_yaxes(title_text='Estimated Revenue (USD)', row=2, col=1)
         fig.update_yaxes(title_text='Estimated Revenue (USD)', row=2, col=2)

         fig.update_layout(height=800, width=1200, showlegend=True)

         fig.show()
```

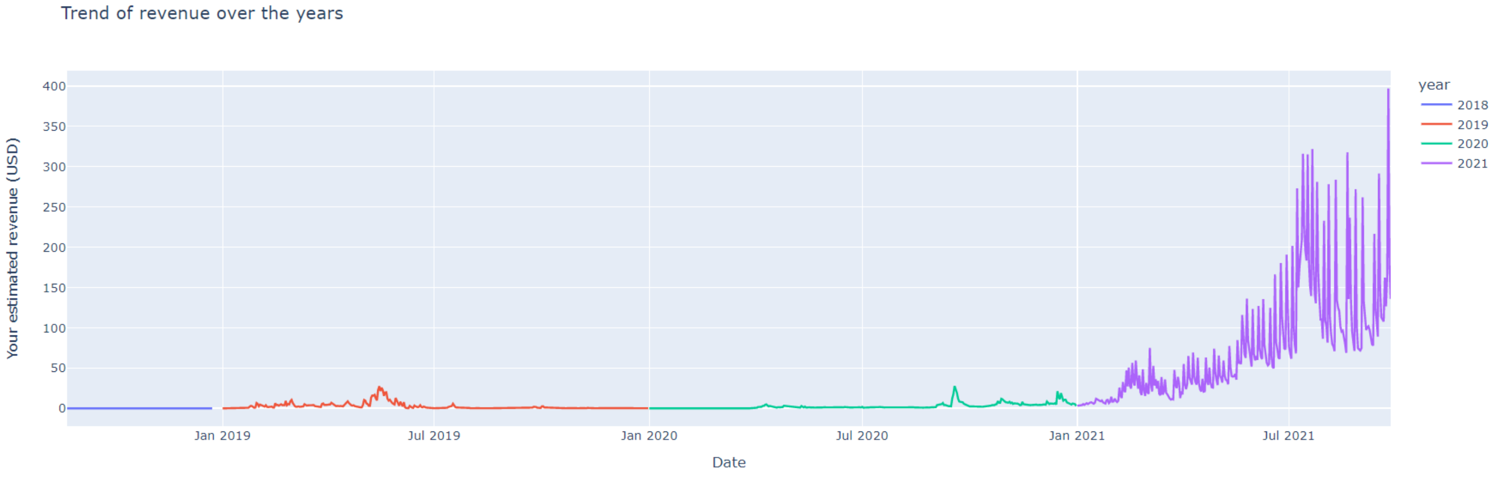
```
plot_in_grid(df1,df2,df3,df4)
```





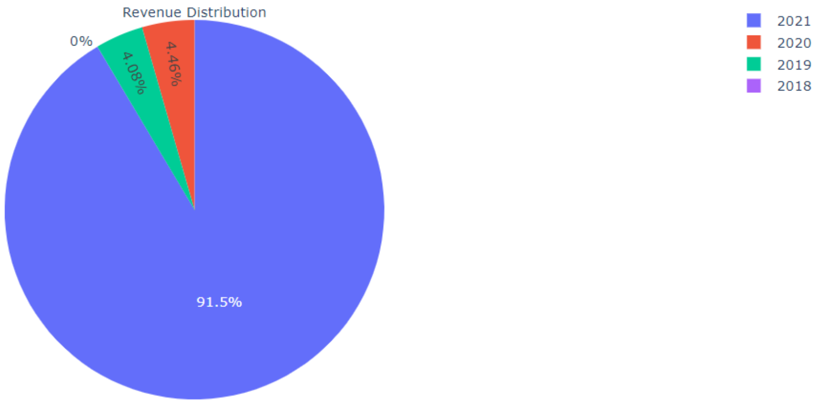
The above plots show the trend of revenue over the years. There is a spike in the mid 2019.In 2020, the revenue saw a considerable increase from September to December. In 2021 there is a constant increase in revenue with time.

```
[ ]  px.line(df, x='Date', y='Your estimated revenue (USD)',title='Trend of revenue over the years',color='year')
```



Revenue distribution over the years:

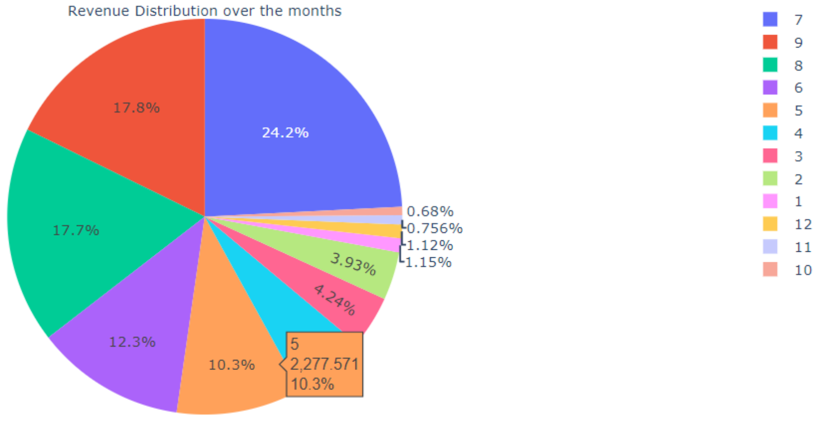
```
fig = go.Figure(data=[go.Pie(labels=df['year'], values=df['Your estimated revenue (USD)'],title='Revenue Distribution')])
fig.show()
```



Maximum revenue generation has occured in 2021 & it contributes to 91.5% of the total revenue generated.

Distribution of revenue over the months:

```
fig = go.Figure(data=[go.Pie(labels=df['month'], values=df['Your estimated revenue (USD)'],title='Revenue Distribution over the months')])
fig.show()
```



From the pie chart, it can be inferred that maximum amount of revenue has been generated in July & September over the years.November & December have the least revenue generation.