Analyze social media data

Importing Libraries

In [138…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing the Dataset

In [140…
```
df = pd.read_csv(r"C:\Users\Meharaj\Downloads\top_insta_influencers_data.csv")
```

In [141…
```
df
```

Out[141…

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate
0	1	cristiano	92	3.3k	475.8m	8.7m	1.39%
1	2	kyliejenner	91	6.9k	366.2m	8.3m	1.62%
2	3	leomessi	90	0.89k	357.3m	6.8m	1.24%
3	4	selenagomez	93	1.8k	342.7m	6.2m	0.97%
4	5	therock	91	6.8k	334.1m	1.9m	0.20%
...	...	...	...	...	...	...	...
195	196	iambeckyg	71	2.3k	33.2m	623.8k	1.40%
196	197	nancyajram	81	3.8k	33.2m	390.4k	0.64%
197	198	luansantana	79	0.77k	33.2m	193.3k	0.26%
198	199	nickjonas	78	2.3k	33.0m	719.6k	1.42%
199	200	raisa6690	80	4.2k	32.8m	232.2k	0.30%

200 rows × 10 columns

Data Description

In [143…
```
df.shape
```

Out[143…
```
(200, 10)
```

In [144…
```
df.columns
```

Out[144…
```
Index(['rank', 'channel_info', 'influence_score', 'posts', 'followers',
       'avg_likes', '60_day_eng_rate', 'new_post_avg_like', 'total_likes',
       'country'],
      dtype='object')
```

In [145…
```
df.dtypes
```

Out[145…
```
rank                 int64
channel_info        object
influence_score      int64
posts               object
followers           object
avg_likes           object
60_day_eng_rate     object
new_post_avg_like   object
total_likes         object
country             object
dtype: object
```

In [146…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   rank               200 non-null    int64 
 1   channel_info       200 non-null    object
 2   influence_score    200 non-null    int64 
 3   posts              200 non-null    object
 4   followers          200 non-null    object
 5   avg_likes          200 non-null    object
 6   60_day_eng_rate    200 non-null    object
 7   new_post_avg_like  200 non-null    object
 8   total_likes        200 non-null    object
 9   country            138 non-null    object
dtypes: int64(2), object(8)
memory usage: 15.8+ KB
```

In [147…
```
df.describe()
```

Out[147…

	rank	influence_score
count	200.000000	200.000000
mean	100.500000	81.820000
std	57.879185	8.878159
min	1.000000	22.000000
25%	50.750000	80.000000
50%	100.500000	84.000000
75%	150.250000	86.000000
max	200.000000	93.000000

Data cleaning

duplicate

In [150…
```
duplicates = df.duplicated().sum()
duplicates
```

Out[150…
```
0
```

Missing Values

In [152…
```
df.isna().sum()
```

Out[152…
```
rank                 0
channel_info         0
influence_score      0
posts                0
followers            0
avg_likes            0
60_day_eng_rate      0
new_post_avg_like    0
total_likes          0
country             62
dtype: int64
```

In [153…
```
df["country"].unique
```

Out[153…
```
<bound method Series.unique of 0              Spain
1      United States
2                NaN
3      United States
4      United States
           ...
195    United States
196           France
197           Brazil
198    United States
199        Indonesia
Name: country, Length: 200, dtype: object>
```

In [154…
```
df["country"].isna().sum()
```

Out[154…
```
62
```

In [155…
```
df[df["country"].isna()]
```

Out[155…

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rat
2	3	leomessi	90	0.89k	357.3m	6.8m	1.24%
15	16	virat.kohli	87	1.4k	211.8m	3.5m	0.96%
18	19	mileycyrus	89	1.2k	181.5m	1.3m	0.51%
20	21	katyperry	92	2.0k	170.3m	715.0k	0.16%
26	27	kingjames	86	2.3k	130.9m	2.1m	0.92%
…	…	…	…	…	…	…	.
179	180	sachintendulkar	76	1.0k	35.3m	800.4k	2.03%
182	183	lunamaya	83	4.1k	34.8m	145.4k	0.17%
183	184	toni.kr8s	83	0.94k	34.7m	597.2k	1.68%
185	186	paollaoliveirareal	84	4.6k	34.7m	367.8k	0.57%
187	188	adidasoriginals	83	0.15k	34.2m	136.2k	0.20%

62 rows × 10 columns

In [156…
```
df["country"].mode()
```

Out[156…
```
0    United States
Name: country, dtype: object
```

In [157…
```
df["country"].fillna("United States",inplace = True)
```

In [158…
```
df
```

Out[158…

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate
0	1	cristiano	92	3.3k	475.8m	8.7m	1.39%
1	2	kyliejenner	91	6.9k	366.2m	8.3m	1.62%
2	3	leomessi	90	0.89k	357.3m	6.8m	1.24%
3	4	selenagomez	93	1.8k	342.7m	6.2m	0.97%
4	5	therock	91	6.8k	334.1m	1.9m	0.20%
...	...	...	...	...	...	...	...
195	196	iambeckyg	71	2.3k	33.2m	623.8k	1.40%
196	197	nancyajram	81	3.8k	33.2m	390.4k	0.64%
197	198	luansantana	79	0.77k	33.2m	193.3k	0.26%
198	199	nickjonas	78	2.3k	33.0m	719.6k	1.42%
199	200	raisa6690	80	4.2k	32.8m	232.2k	0.30%

200 rows × 10 columns

In [159…

```
df.isna().sum()
```

Out[159…

```
rank                0
channel_info        0
influence_score     0
posts               0
followers           0
avg_likes           0
60_day_eng_rate     0
new_post_avg_like   0
total_likes         0
country             0
dtype: int64
```

In [160…

```
def convert_to_numeric(value):
    if isinstance(value, str):
        value = value.lower().replace(',', '')
        if 'k' in value:
            return float(value.replace('k', '')) * 1e3
        elif 'm' in value:
            return float(value.replace('m', '')) * 1e6
        elif 'b' in value:
            return float(value.replace('b', '')) * 1e9
    return pd.to_numeric(value, errors='coerce')

convert_to_numeric("5.3m")
```

Out[160… 5300000.0

In [161…
```
numeric_cols = ['posts', 'followers', 'avg_likes', 'new_post_avg_like', 'total_l
```

In [162…
```
for col in numeric_cols:
    df[col] = df[col].apply(convert_to_numeric)
```

In [163…
```
df[numeric_cols].head()
```

Out[163…

	posts	followers	avg_likes	new_post_avg_like	total_likes
0	3300.0	475800000.0	8700000.0	6500000.0	2.900000e+10
1	6900.0	366200000.0	8300000.0	5900000.0	5.740000e+10
2	890.0	357300000.0	6800000.0	4400000.0	6.000000e+09
3	1800.0	342700000.0	6200000.0	3300000.0	1.150000e+10
4	6800.0	334100000.0	1900000.0	665300.0	1.250000e+10

Convert Engagement Rate to Proper Format

In [165…
```
df['60_day_eng_rate'] = pd.to_numeric(df['60_day_eng_rate'], errors='coerce') /
```

In [166…
```
df['60_day_eng_rate'].head()
```

Out[166…
```
0   NaN
1   NaN
2   NaN
3   NaN
4   NaN
Name: 60_day_eng_rate, dtype: float64
```

In [167…
```
df['60_day_eng_rate'].unique()
```

Out[167… array([nan])

In [168…
```
df.drop(columns=['60_day_eng_rate'], inplace=True)
```

In [169…
```
df
```

Out[169…

	rank	channel_info	influence_score	posts	followers	avg_likes	new_post_avg_
0	1	cristiano	92	3300.0	475800000.0	8700000.0	65000
1	2	kyliejenner	91	6900.0	366200000.0	8300000.0	59000
2	3	leomessi	90	890.0	357300000.0	6800000.0	44000
3	4	selenagomez	93	1800.0	342700000.0	6200000.0	33000
4	5	therock	91	6800.0	334100000.0	1900000.0	6653
…	…	…	…	…	…	…	
195	196	iambeckyg	71	2300.0	33200000.0	623800.0	4647
196	197	nancyajram	81	3800.0	33200000.0	390400.0	2080
197	198	luansantana	79	770.0	33200000.0	193300.0	826
198	199	nickjonas	78	2300.0	33000000.0	719600.0	4677
199	200	raisa6690	80	4200.0	32800000.0	232200.0	974

200 rows × 9 columns

In [170…

```
df.head()
```

Out[170…

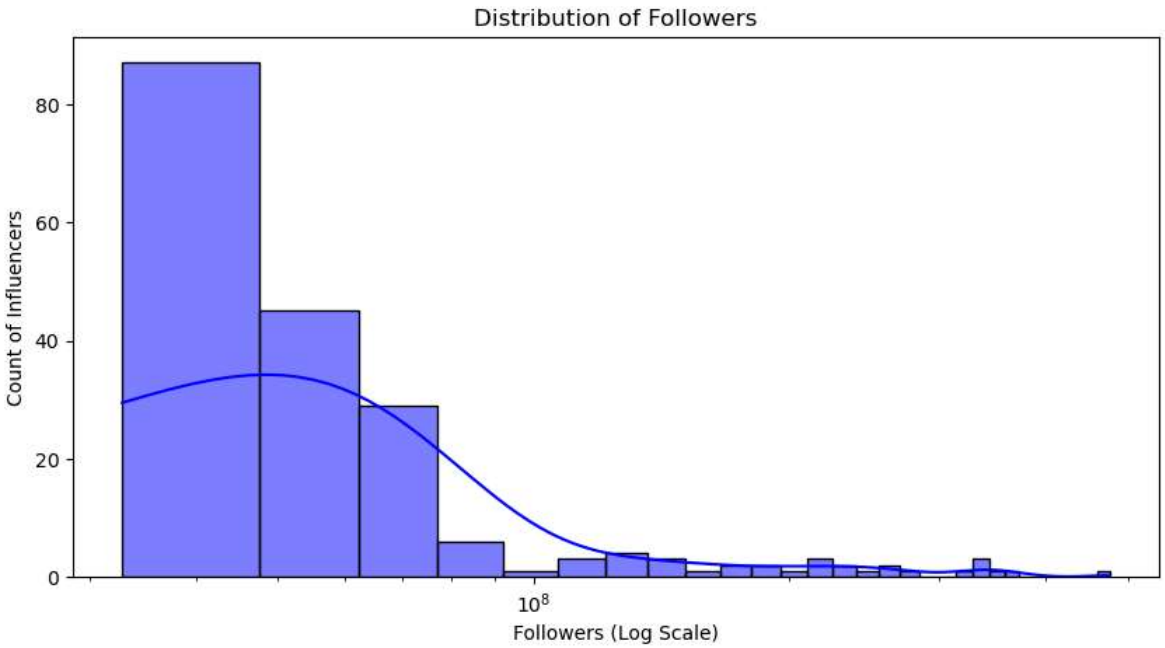
	rank	channel_info	influence_score	posts	followers	avg_likes	new_post_avg_lik
0	1	cristiano	92	3300.0	475800000.0	8700000.0	6500000
1	2	kyliejenner	91	6900.0	366200000.0	8300000.0	5900000
2	3	leomessi	90	890.0	357300000.0	6800000.0	4400000
3	4	selenagomez	93	1800.0	342700000.0	6200000.0	3300000
4	5	therock	91	6800.0	334100000.0	1900000.0	665300

Univariate Analysis

In [172…

```
plt.figure(figsize=(10, 5))
sns.histplot(df['followers'], bins=30, kde=True, color='blue')
plt.xlabel("Followers (Log Scale)")
plt.ylabel("Count of Influencers")
plt.title("Distribution of Followers")
```

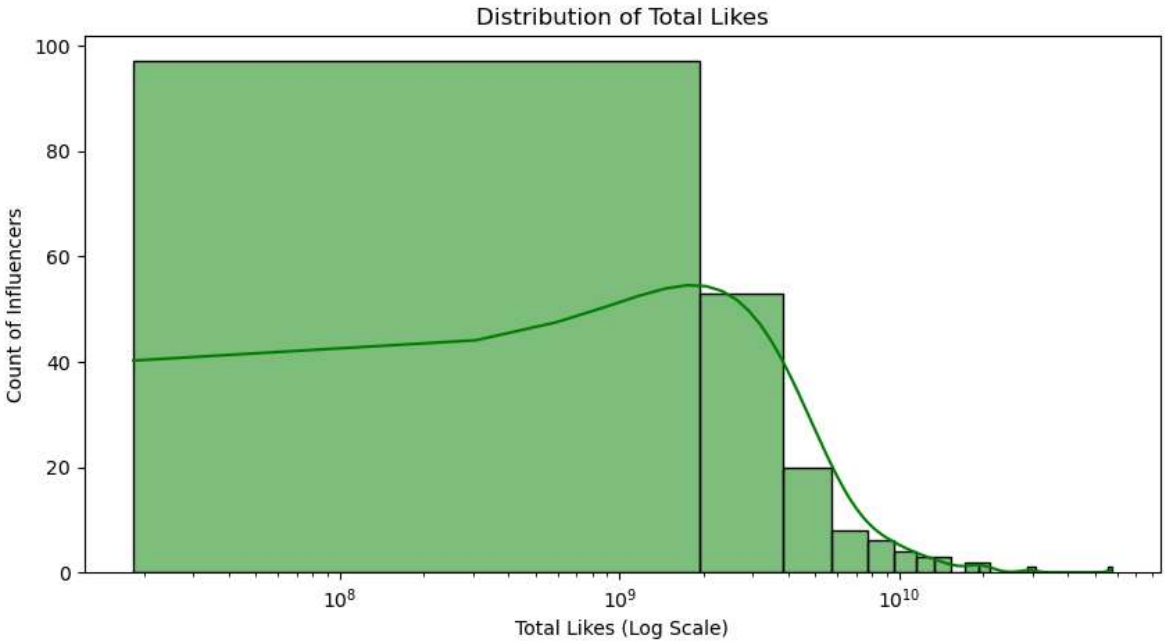
```
plt.xscale('log')
plt.show()
```



The distribution is right-skewed, meaning a small number of influencers dominate in terms of followers.

In [174…

```
plt.figure(figsize=(10, 5))
sns.histplot(df['total_likes'], bins=30, kde=True, color='green')
plt.xlabel("Total Likes (Log Scale)")
plt.ylabel("Count of Influencers")
plt.title("Distribution of Total Likes")
plt.xscale('log')
plt.show()
```

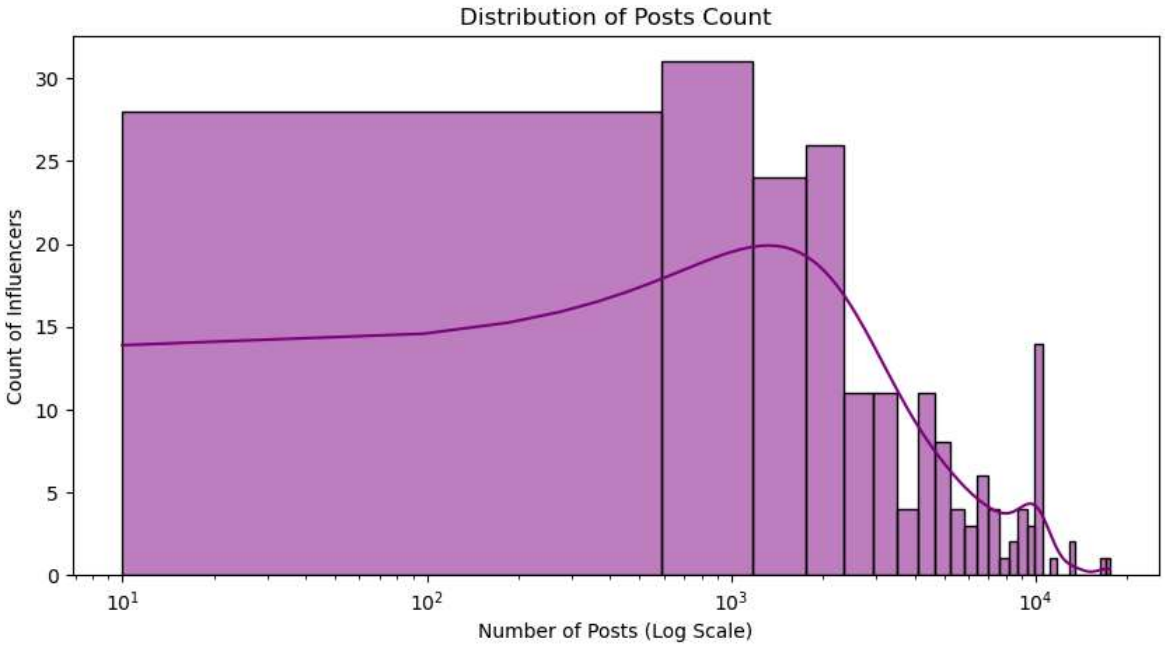


In [175…

```
# while a few have billions of likes. Again, we see a right-skewed distribution,
```

In [176…

```
plt.figure(figsize=(10, 5))
sns.histplot(df['posts'], bins=30, kde=True, color='purple')
plt.xlabel("Number of Posts (Log Scale)")
plt.ylabel("Count of Influencers")
plt.title("Distribution of Posts Count")
plt.xscale('log')  # Log scale for better visibility
plt.show()
```

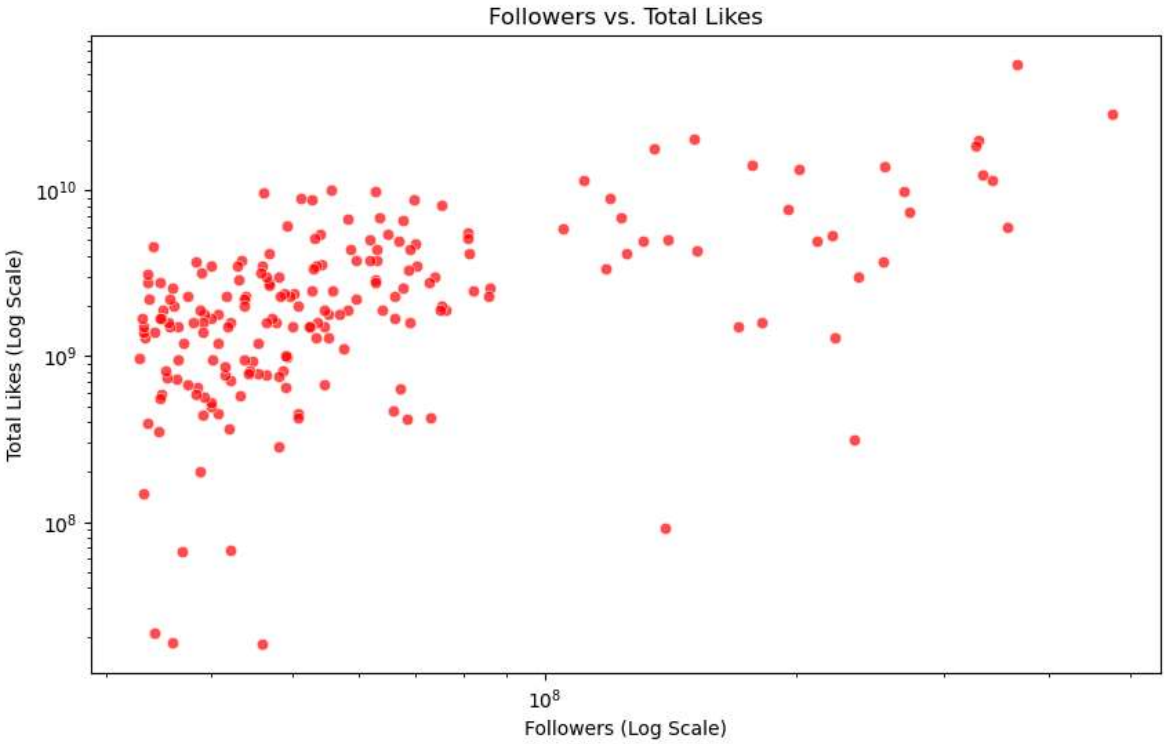


Bivariate Analysis

Followers vs. Total Likes

In [179…

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df['followers'], y=df['total_likes'], alpha=0.7, color='red')
plt.xscale('log')
plt.yscale('log')
plt.xlabel("Followers (Log Scale)")
plt.ylabel("Total Likes (Log Scale)")
plt.title("Followers vs. Total Likes")
plt.show()
```

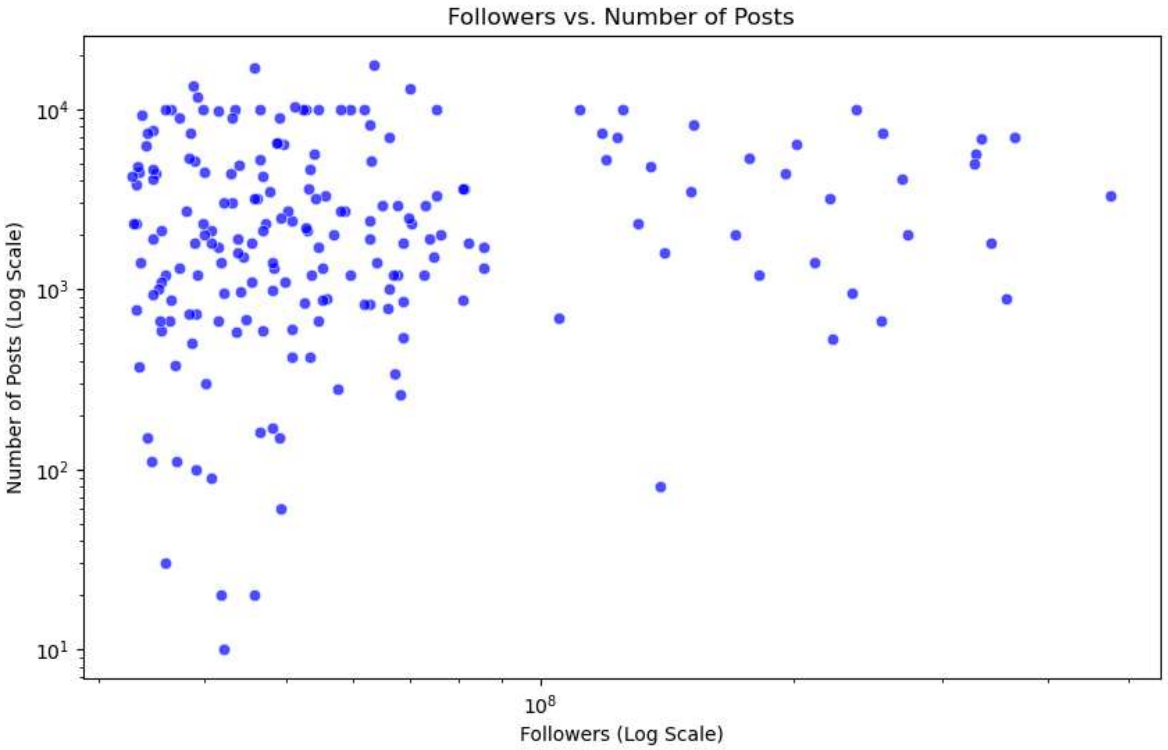


a positive correlation between followers and total likes more followers generally mean more total likes. However, some influencers with fewer followers still get high engagement!

Followers vs. Number of Posts

In [182…

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df['followers'], y=df['posts'], alpha=0.7, color='blue')
plt.xscale('log')
plt.yscale('log')
plt.xlabel("Followers (Log Scale)")
plt.ylabel("Number of Posts (Log Scale)")
plt.title("Followers vs. Number of Posts")
plt.show()
```

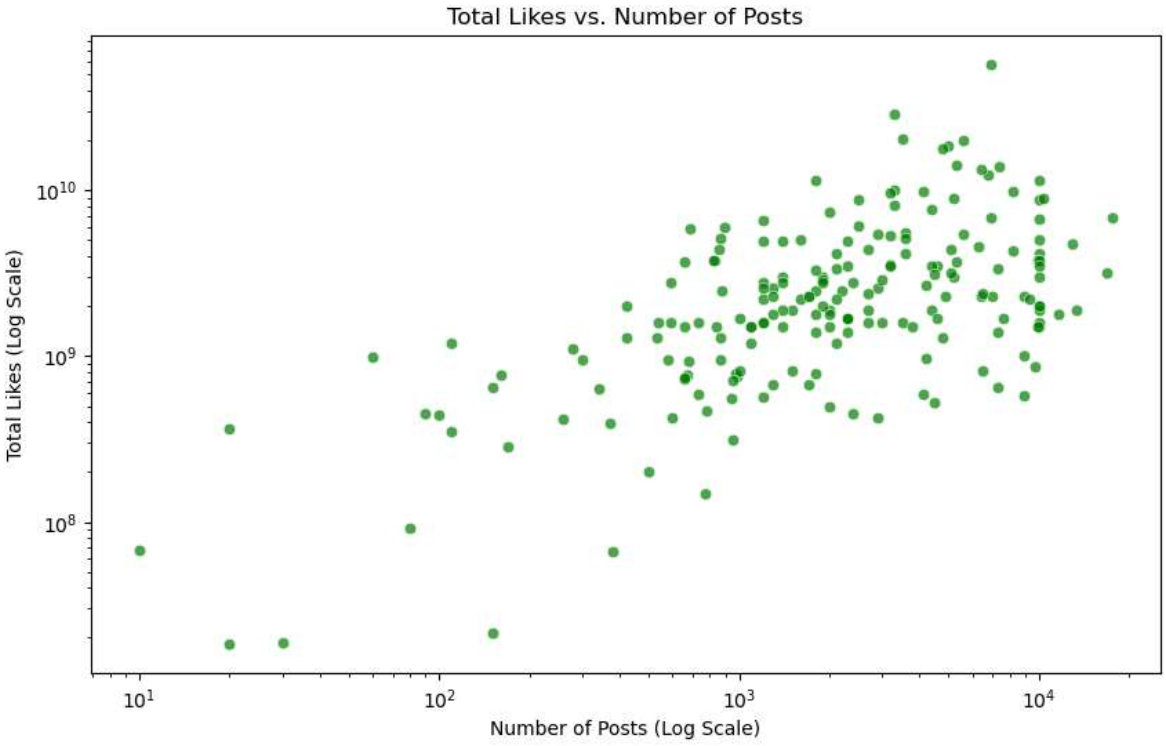


no strong correlation between followers and number of posts—some influencers gain millions of followers with fewer posts

Total Likes vs. Number of Posts

In [185…

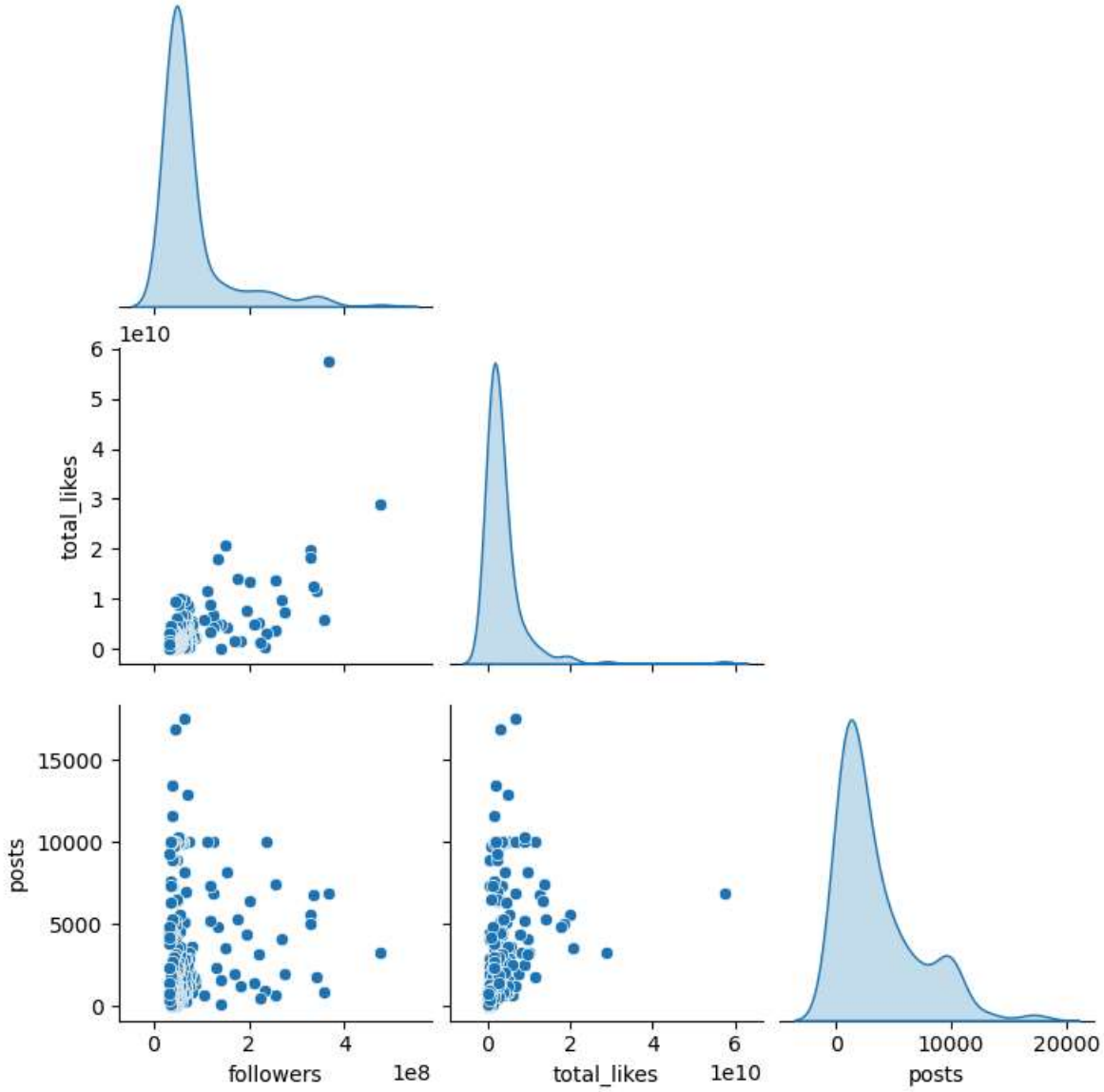
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df['posts'], y=df['total_likes'], alpha=0.7, color='green')
plt.xscale('log')
plt.yscale('log')
plt.xlabel("Number of Posts (Log Scale)")
plt.ylabel("Total Likes (Log Scale)")
plt.title("Total Likes vs. Number of Posts")
plt.show()
```



Multivariate Analysis

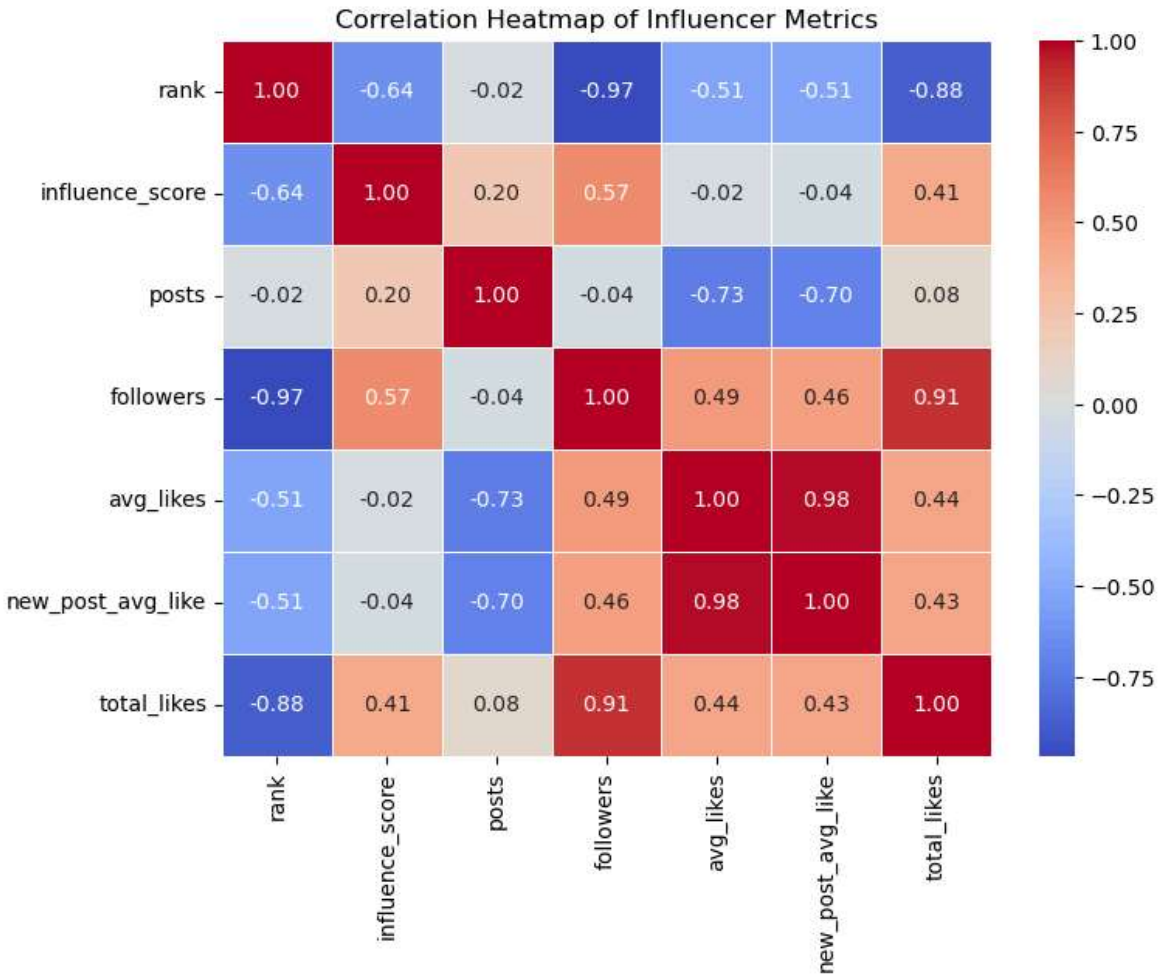
In [187…

```
sns.pairplot(df[['followers', 'total_likes', 'posts']], diag_kind='kde', corner=
plt.show()
```



In [188…
```
numeric_df = df_cleaned.select_dtypes(include=['number'])
```

In [189…
```
correlation_matrix = numeric_df.corr().corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidt
plt.title("Correlation Heatmap of Influencer Metrics")
plt.show()
```



+1 = Strong positive correlation (e.g., followers and total likes)

-1 = Strong negative correlation.

0 = No significant relationship.

Dataset Summary

200 influencers ranked by influence score.

Followers range: 32.8M – 475.8M.

Engagement Rate (last 60 days): Varies from 0.0001% to 0.26%.

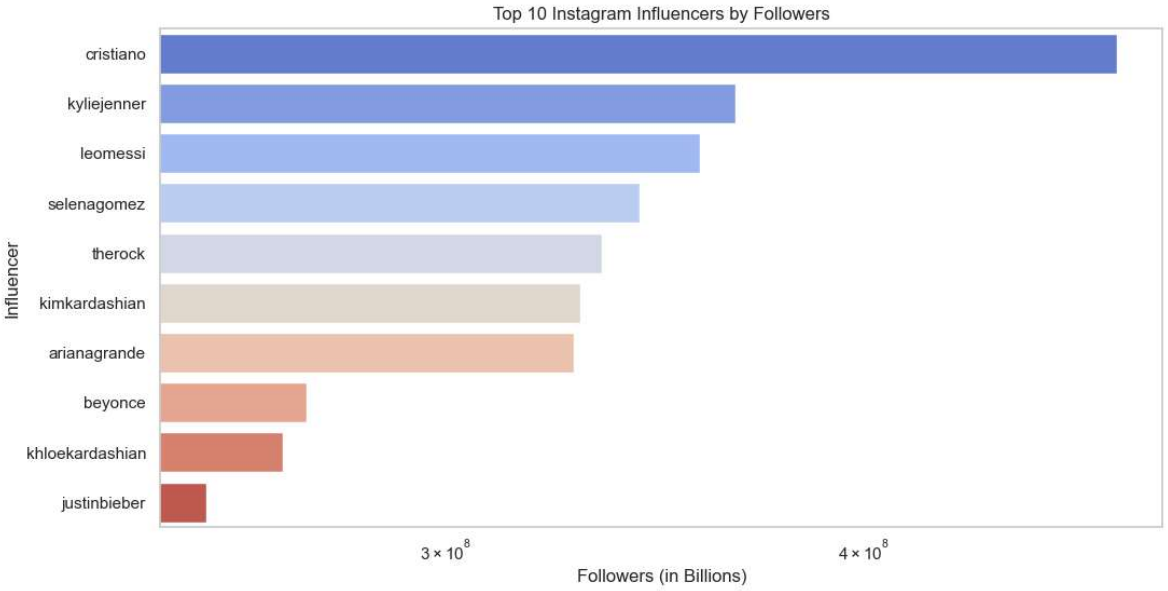
Total Likes: Up to 57.4 billion for top influencers.

Some missing country data (e.g., Lionel Messi). -->

In [230…

```
sns.set(style="whitegrid")

top_followers = df.nlargest(10, 'followers')

plt.figure(figsize=(12, 6))
sns.barplot(x=top_followers['followers'], y=top_followers['channel_info'], palet
plt.xlabel("Followers (in Billions)")
plt.ylabel("Influencer")
plt.title("Top 10 Instagram Influencers by Followers")
plt.xscale('log')  # Log scale to handle large differences in follower counts
plt.show()
```

```
C:\Users\Meharaj\AppData\Local\Temp\ipykernel_27336\1708835156.py:6: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=top_followers['followers'], y=top_followers['channel_info'], pale
tte='coolwarm')
```

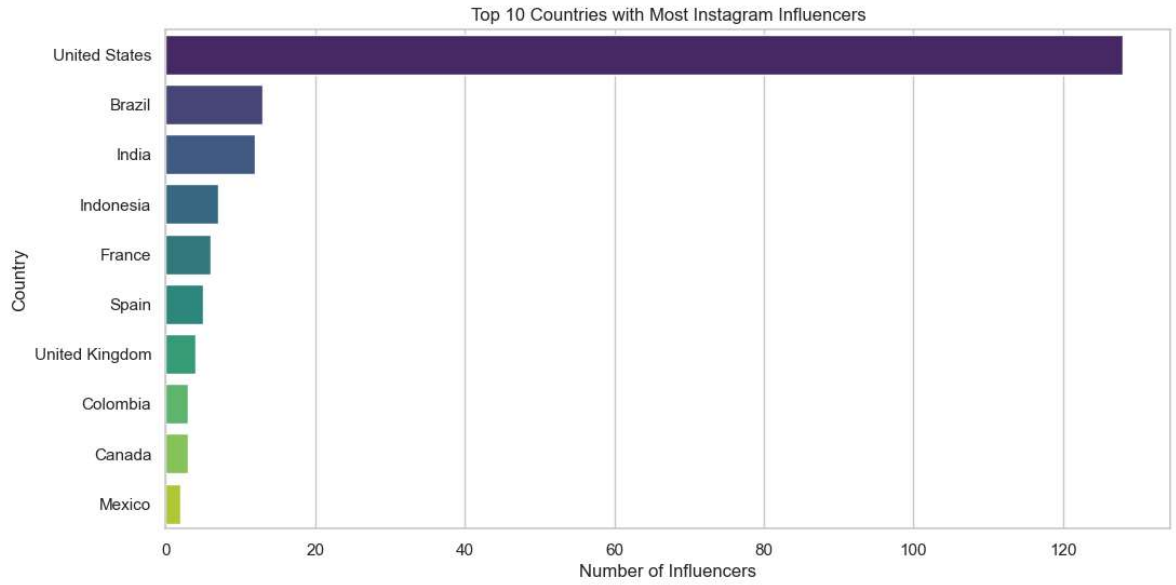


Top countries with the most influencers

In [237…

```
top_countries = df['country'].value_counts().head(10)


plt.figure(figsize=(12, 6))
sns.barplot(x=top_countries.values, y=top_countries.index, palette='viridis')
plt.xlabel("Number of Influencers")
plt.ylabel("Country")
plt.title("Top 10 Countries with Most Instagram Influencers")
plt.show()
```

```
C:\Users\Meharaj\AppData\Local\Temp\ipykernel_27336\2830689138.py:5: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.barplot(x=top_countries.values, y=top_countries.index, palette='viridis')
```



In [ ]: