Project Title	**Top Instagram Influencers Data (Cleaned)**
Tools	ML, Python, SQL, Excel
Domain	Data Analyst,
Project Difficulties level	intermediate

Dataset : Dataset is available in the given link. You can download it at your convenience.

Click here to download data set

About Dataset

Instagram is an American photo and video sharing social networking service founded in 2010 by Kevin Systrom and Mike Krieger, and later acquired by Facebook Inc.. The app allows users to upload media that can be edited with filters and organized by hashtags and geographical tagging. Posts can be shared publicly or with preapproved followers. Users can browse other users' content by tag and location, view trending content, like photos, and follow other users to add their content to a personal feed.

Instagram network is very much used to influence people (the users followers) in a particular way for a specific issue - which can impact the order in some ways.

About this file

In this file, basically there are 10 attributes. It has been ordered on basis of the rank which has been decided on basis of "followers".

rank: Rank of the Influencer on basis of number of followers they have
channel_info: Username of the Instagrammer
influence score: Influence score of the users. It is calculated on basis of mentions, importance and popularity
posts: Number of posts they have made so far
followers: Number of followers of the user
avg_likes: Average likes on instagrammer posts (total likes/ total posts)
60_day_eng_rate: Last 60 days engagement rate of instagrammer as faction of engagements they have done so far
new_post_avg_like: Average likes they have on new posts
total Likes: Total likes the user has got on their posts. (in Billion)
country: Country or region of origin of the user.

Example: You can get the basic idea how you can create a project from here

Project: Analysis and Prediction on Top Instagram Influencers Data

This project will analyze and predict insights on Instagram influencers using a dataset containing information on influencer ranking, engagement metrics, followers, likes, and more. This project targets a more advanced audience with 5 years of experience, so it will involve detailed exploratory data analysis (EDA), visualizations, and a machine learning model to predict key engagement metrics.

Dataset Overview

Here's an overview of the columns we'll be working with:

rank: Influencer rank

channel_info: Instagram handle or channel information

influence_score: Calculated influence score based on engagement and followers

posts: Total number of posts made by the influencer

followers: Number of followers

avg_likes: Average likes per post

60_day_eng_rate: Engagement rate over the past 60 days

new_post_avg_like: Average likes on recent posts

total_likes: Cumulative likes on all posts

country: Influencer's country

Step 1: Data Loading and Preprocessing

```
import pandas as pd
import numpy as np

# Load the dataset
df = pd.read_csv('top_instagram_influencers.csv')

# Quick inspection of data
```

```
print(df.info())
print(df.describe())

# Drop any duplicate rows if present
df.drop_duplicates(inplace=True)

# Handle missing values
# Fill missing numerical values with median, and categorical
with mode
for column in df.columns:
    if df[column].dtype == 'object':
        df[column].fillna(df[column].mode()[0], inplace=True)
    else:
        df[column].fillna(df[column].median(), inplace=True)

# Convert necessary columns to appropriate data types
df['followers'] = df['followers'].astype(int)
df['posts'] = df['posts'].astype(int)
df['total_likes'] = df['total_likes'].astype(int)
```

Step 2: Exploratory Data Analysis (EDA)

EDA is critical for uncovering patterns, trends, and relationships in the data.

1. Summary Statistics

Generate a summary of key metrics to understand distributions.

```
# Display summary statistics for numeric columns
print(df[['influence_score', 'followers', 'avg_likes',
'60_day_eng_rate', 'new_post_avg_like']].describe())
```

2. Relationship between Followers and Engagement

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 6))
sns.scatterplot(data=df, x='followers', y='60_day_eng_rate',
hue='country', alpha=0.7)
plt.title('Followers vs 60-Day Engagement Rate')
plt.xlabel('Number of Followers')
plt.ylabel('60-Day Engagement Rate (%)')
plt.legend(title='Country')
plt.show()
```

3. Distribution of Influence Score

```
plt.figure(figsize=(10, 5))
sns.histplot(df['influence_score'], bins=30, kde=True)
```

```python
plt.title('Distribution of Influence Score')
plt.xlabel('Influence Score')
plt.show()
```

4. Most Active Countries

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

plt.figure(figsize=(10, 5))
sns.barplot(x=top_countries.index, y=top_countries.values,
palette="viridis")
plt.title('Top 10 Countries by Number of Influencers')
plt.xlabel('Country')
plt.ylabel('Number of Influencers')
plt.show()
```

Step 3: Feature Engineering

To improve our model's performance, let's create a few additional features that might be helpful predictors.

```python
# Creating engagement-related features
df['like_follower_ratio'] = df['total_likes'] / df['followers']
df['post_follower_ratio'] = df['posts'] / df['followers']
df['avg_likes_ratio'] = df['avg_likes'] / df['followers']
```

Step 4: Model Building

Objective: Predict `influence_score` using other variables.

Split the data into training and testing sets

Scale the features for optimal performance

Train a regression model to predict the influence score

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Define feature columns and target variable
X = df[['followers', 'avg_likes', '60_day_eng_rate',
'new_post_avg_like', 'like_follower_ratio',
'post_follower_ratio']]
y = df['influence_score']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Standardize features
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and train a Random Forest Regressor
model = RandomForestRegressor(n_estimators=100,
random_state=42)
model.fit(X_train_scaled, y_train)

# Predictions and evaluation
y_pred = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')
```

Step 5: Model Interpretation and Feature Importance

```
# Display feature importances
feature_importances = pd.Series(model.feature_importances_,
index=X.columns)
feature_importances.sort_values().plot(kind='barh',
title='Feature Importance')
plt.show()
```

Step 6: Visualizing Predictions

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.plot([y.min(), y.max()], [y.min(), y.max()], '--',
color='red')
plt.xlabel('True Influence Score')
plt.ylabel('Predicted Influence Score')
plt.title('True vs Predicted Influence Score')
plt.show()
```

Step 7: Final Observations and Summary

Top Influential Factors: Feature importance analysis indicates which features most influence the `influence_score`.

Model Performance: With the achieved R^2 score, assess the accuracy of the model in predicting an influencer's influence score based on follower metrics.

Business Insights: Using insights on top-engaging influencers by country and engagement rates, businesses can strategize influencer collaborations for marketing.

Sample Code and output

```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing
Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory
(/kaggle/working/) that gets preserved as output when you create
```

```
a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/top-instagram-influencers-data-cleaned/top_insta_influencers_data.csv
```

Lets Learn Pandas

Pandas is a Library which is used to do operations on datasets or dataframe which basically is a 2D format of data which means set od rows and columns

Here we are taking a dictionary in which there is an array of names and array of marks and we are simply making a dataframe out of it to understand basics of pandas

In [2]:

```
dictionary = { 'Names': ["Alice", "Oggy", "Modi ji", "SRK", "Tony Stark"], "Marks": [10,15,20,17,18] }
df = pd.DataFrame(dictionary)
df.head()
```

Out[2]:

	Names	Marks
0	Alice	10
1	Oggy	15
2	Modi ji	20
3	SRK	17
4	Tony Stark	18

In [3]:

```
print(df['Names'])
```

```
0          Alice
1           Oggy
2        Modi ji
```

```
3           SRK
4    Tony Stark
Name: Names, dtype: object
```

In [4]:

```
print(df.loc[3])
```

```
Names     SRK
Marks      17
Name: 3, dtype: object
```

Earlier we took a very small and simple dataframe out of a self made dictionary of elements now lets take any real time dataset

We usually import dataset from any source we like and give the pd.read_csv function the path to access the dataset

In [5]:

```
insta_df =
pd.read_csv('/kaggle/input/top-instagram-influencers-data-cleaned/top_insta_influencers_data.csv')
print('Dataset is ready to use')
```

```
Dataset is ready to use
```

In [6]:

```
insta_df.head(25)
```

Out[6]:

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	country
0	1	cristiano	92	3.3k	475.8m	8.7m	1.39%	6.5m	29.0b	Spain
1	2	kyliejenner	91	6.9k	366.2m	8.3m	1.62%	5.9m	57.4b	United States
2	3	leomessi	90	0.89	357.3m	6.8m	1.24%	4.4m	6.0b	NaN

				k						
3	4	selenagomez	93	1.8k	342.7m	6.2m	0.97%	3.3m	11.5b	United States
4	5	therock	91	6.8k	334.1m	1.9m	0.20%	665.3k	12.5b	United States
5	6	kimkardashian	91	5.6k	329.2m	3.5m	0.88%	2.9m	19.9b	United States
6	7	arianagrande	92	5.0k	327.7m	3.7m	1.20%	3.9m	18.4b	United States

7	8	beyonce	92	2.0k	272.8m	3.6m	0.76%	2.0m	7.4b	United States
8	9	khloekardashian	89	4.1k	268.3m	2.4m	0.35%	926.9k	9.8b	United States
9	10	justinbieber	91	7.4k	254.5m	1.9m	0.59%	1.5m	13.9b	Canada
10	11	kendalljenner	90	0.66k	254.0m	5.5m	2.04%	5.1m	3.7b	United States
11	12	natgeo	91	10.0k	237.0m	302.2k	0.07%	159.3k	3.0b	United States

12	13	nike	90	0.95k	234.1m	329.0k	0.08%	181.8k	313.6m	United States
13	14	taylorswift	91	0.53k	222.2m	2.4m	1.01%	2.3m	1.3b	United States
14	15	jlo	89	3.2k	220.4m	1.7m	0.62%	1.4m	5.3b	United States
15	16	virat.kohli	87	1.4k	211.8m	3.5m	0.96%	2.0m	4.9b	NaN
16	17	nickiminaj	90	6.4k	201.6m	2.1m	0.53%	1.0m	13.5b	United States

17	18	kourtney kardash	89	4.4k	195.2m	1.8m	0.67%	1.3m	7.7b	United States
18	19	mileycyrus	89	1.2k	181.5m	1.3m	0.51%	913.6k	1.6b	NaN
19	20	neymarjr	90	5.3k	177.1m	2.7m	1.09%	1.9m	14.1b	Brazil
20	21	katyperry	92	2.0k	170.3m	715.0k	0.16%	265.1k	1.5b	NaN
21	22	kevinhart4real	88	8.2k	152.0m	522.0k	0.08%	115.2k	4.3b	United States
22	23	zendaya	87	3.5k	150.7m	5.8m	3.17%	4.8m	20.6b	United Stat

										es
23	24	iamcardib	75	1.6k	140.5m	3.1m	1.10%	1.5m	5.0b	United States
24	25	ddlovato	88	0.08k	139.1m	1.1m	0.27%	363.4k	91.3m	United States

In [7]:

```
insta_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 [8]:

```
insta_df.shape
```

Out[8]:

```
(200, 10)
```

Lets Do some Data Visualization

For visualizing data we use the two very famous libraries Matplotlib and Seaborn

Lets import them and plot some nice visualizations

In [9]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [10]:

```
country = insta_df['country'].value_counts()
```

In [11]:

```
country
```

Out[11]:

```
country
United States          66
Brazil                 13
India                  12
Indonesia               7
France                  6
Spain                   5
United Kingdom          4
Colombia                3
Canada                  3
Mexico                  2
Turkey                  2
```

```
Netherlands              2
Switzerland              1
Germany                  1
Czech Republic           1
British Virgin Islands   1
Sweden                   1
Australia                1
Anguilla                 1
CÃ´te d'Ivoire           1
Puerto Rico              1
United Arab Emirates     1
Italy                    1
Uruguay                  1
Russia                   1

Name: count, dtype: int64
```

In [12]:

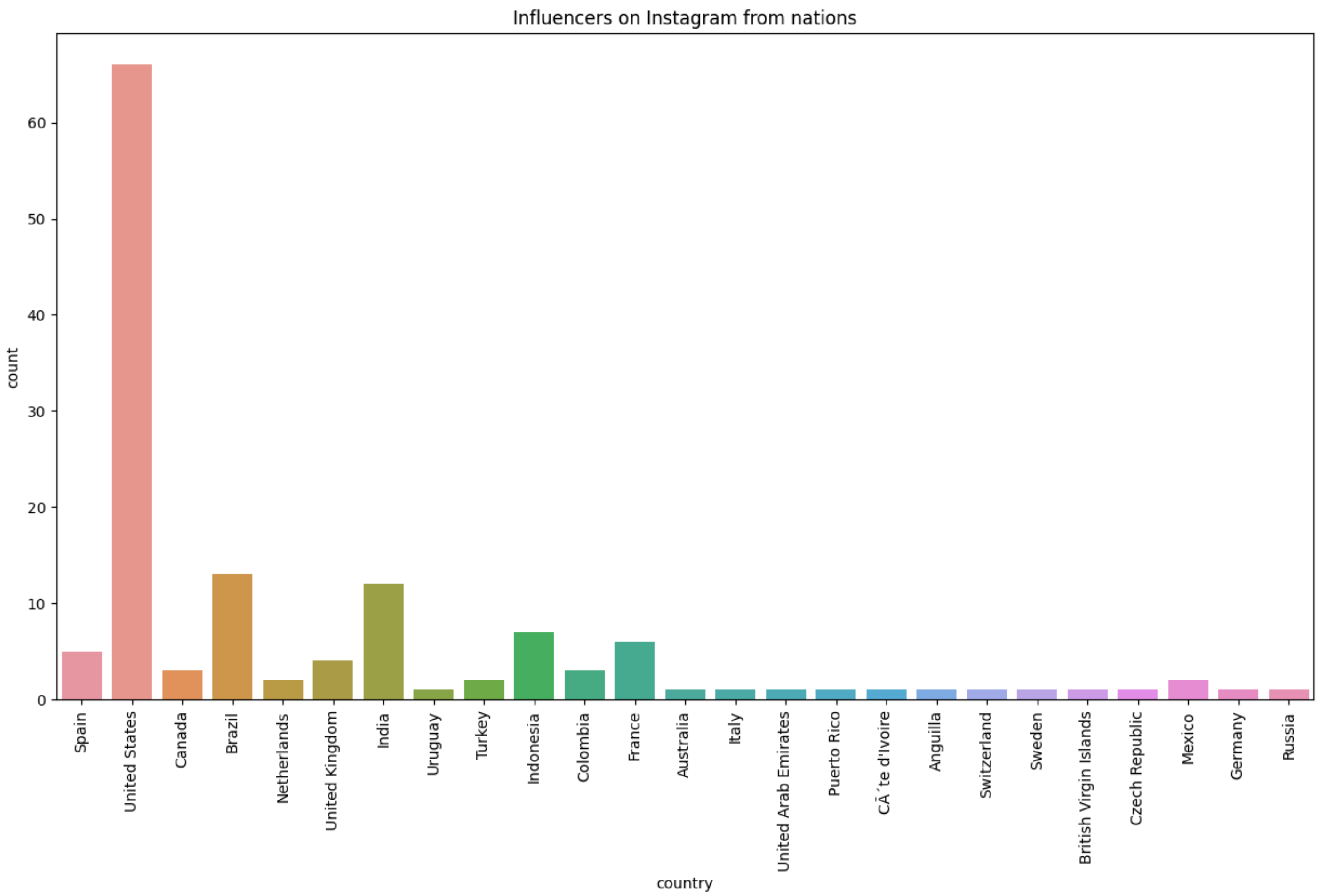
```
plt.figure(figsize=(15,8))
plt.title('Influencers on Instagram from nations')
sns.countplot(x=insta_df["country"])
plt.xticks(rotation=90)
```

Out[12]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13,
```

```
14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24]),
 [Text(0, 0, 'Spain'),
  Text(1, 0, 'United States'),
  Text(2, 0, 'Canada'),
  Text(3, 0, 'Brazil'),
  Text(4, 0, 'Netherlands'),
  Text(5, 0, 'United Kingdom'),
  Text(6, 0, 'India'),
  Text(7, 0, 'Uruguay'),
  Text(8, 0, 'Turkey'),
  Text(9, 0, 'Indonesia'),
  Text(10, 0, 'Colombia'),
  Text(11, 0, 'France'),
  Text(12, 0, 'Australia'),
  Text(13, 0, 'Italy'),
  Text(14, 0, 'United Arab Emirates'),
  Text(15, 0, 'Puerto Rico'),
  Text(16, 0, "CÃ´te d'Ivoire"),
  Text(17, 0, 'Anguilla'),
  Text(18, 0, 'Switzerland'),
  Text(19, 0, 'Sweden'),
  Text(20, 0, 'British Virgin Islands'),
  Text(21, 0, 'Czech Republic'),
  Text(22, 0, 'Mexico'),
```

```
Text(23, 0, 'Germany'),

Text(24, 0, 'Russia')])
```



Processing Data

As we saw that in the datafram we have alot of different values and we need mostly numeric values to make a nice plot so lets process the data

In [13]:

```
insta_df.duplicated().sum()
```

Out[13]:

```
0
```

In [14]:

```
insta_df.describe()
```

Out[14]:

	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

In [15]:

```
# Selecting specific columns to view
insta_df[['channel_info', 'followers', '60_day_eng_rate']]
```

Out[15]:

	channel_info	followers	60_day_eng_rate
0	cristiano	475.8m	1.39%
1	kyliejenner	366.2m	1.62%
2	leomessi	357.3m	1.24%
3	selenagomez	342.7m	0.97%
4	therock	334.1m	0.20%
...	...	...	...

195	iambeckyg	33.2m	1.40%
196	nancyajram	33.2m	0.64%
197	luansantana	33.2m	0.26%
198	nickjonas	33.0m	1.42%
199	raisa6690	32.8m	0.30%

200 rows × 3 columns

In [16]:

```
# # Filtering influencers with over 1 million followers
# high_follower_df = insta_df[insta_df['followers'] > 1_000_000]


# # Filtering influencers with an engagement rate above 5%
```

```
# high_engagement_df = insta_df[insta_df['engagement_rate'] > 5]
```

In [17]:

```
# We got a Type Error now lets fix this
```

Making string values such as 2M into numeric like 2000000

as we know M stands for million and K stands for thousand and so on so we wrote a regex(regular expression that replaces M with 6 zeros, k with 3 zeros and so on)

We are taking all the columns which have these kind of symbols and at once converting them

In [18]:

```
replace = {'b': 'e9', 'm': 'e6', 'k': 'e3', '%': ''}
convert_column = ['total_likes', 'posts', 'followers',
'avg_likes', '60_day_eng_rate', 'new_post_avg_like']
insta_df[convert_column] =
insta_df[convert_column].replace(replace,
regex=True).astype(float)
insta_df[convert_column]
```

Out[18]:

	total_likes	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like
0	2.900000e+10	3300.0	475800000.0	8700000.0	1.39	6500000.0
1	5.740000e+10	6900.0	366200000.0	8300000.0	1.62	5900000.0
2	6.000000e+09	890.0	357300000.0	6800000.0	1.24	4400000.0
3	1.150000e+10	1800.0	342700000.0	6200000.0	0.97	3300000.0
4	1.250000e+10	6800.0	334100000.0	1900000.0	0.20	665300.0
...	...	...	...	...	...	...

195	1.400000e+09	2300.0	33200000.0	623800.0	1.40	464700.0
196	1.500000e+09	3800.0	33200000.0	390400.0	0.64	208000.0
197	1.492000e+08	770.0	33200000.0	193300.0	0.26	82600.0
198	1.700000e+09	2300.0	33000000.0	719600.0	1.42	467700.0
199	9.691000e+08	4200.0	32800000.0	232200.0	0.30	97400.0

200 rows × 6 columns

In [19]:

```
# Filtering influencers with over 1 million followers
high_follower_df = insta_df[insta_df['followers'] > 1_000_000]


high_follower_df
```

Out[19]:

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	country
0	1	cristiano	92	3300.0	475800000.0	8700000.0	1.39	6500000.0	2.900000e+10	Spain
1	2	kyliejenner	91	6900.0	366200000.0	8300000.0	1.62	5900000.0	5.740000e+10	United States
2	3	leomessi	90	890.0	357300000.0	6800000.0	1.24	4400000.0	6.000000e+09	NaN
3	4	selena	93	18	34270	6200	0.97	3300000.	1.1500	Unit

		gomez		00.0	0000.0	000.0		0	00e+10	ed States
4	5	therock	91	6800.0	334100000.0	1900000.0	0.20	665300.0	1.250000e+10	United States
...	...	...	...	...	...	...	...	...	...	...
195	196	iambeckyg	71	2300.0	33200000.0	623800.0	1.40	464700.0	1.400000e+09	United States
196	197	nancyajram	81	3800.0	33200000.0	390400.0	0.64	208000.0	1.500000e+09	France
19	19	luansa	79	77	33200	1933	0.26	82600.0	1.492000e+0	Braz

7	8	ntana		0.0	000.0	00.0			8	il
198	199	nickjonas	78	2300.0	33000000.0	719600.0	1.42	467700.0	1.700000e+09	United States
199	200	raisa6690	80	4200.0	32800000.0	232200.0	0.30	97400.0	9.691000e+08	Indonesia

200 rows × 10 columns

In [20]:

```
# Filtering influencers with an engagement rate above 5%
high_engagement_df = insta_df[insta_df['60_day_eng_rate'] > 5]

high_engagement_df
```

Out[20]:

	ra	channel	influence_sco	po	follow	avg_l	60_day_eng_ra	new_post_avg_lik	total_li	cou

	nk	_info	re	sts	ers	ikes	te	e	kes	ntry
32	33	billieeilish	73	690.0	105200000.0	8500000.0	5.02	5200000.0	5.900000e+09	NaN
38	39	lalalalisa_m	70	870.0	80900000.0	5800000.0	9.00	7200000.0	5.100000e+09	NaN
49	50	jennierubyjane	76	860.0	68900000.0	5100000.0	8.36	5700000.0	4.400000e+09	NaN
53	54	tomholland2013	77	1200.0	67700000.0	5400000.0	10.83	7300000.0	6.600000e+09	NaN
56	57	bts.bighitofficial	78	1200.0	66900000.0	4100000.0	5.40	3600000.0	4.900000e+09	Uruguay

64	65	sooyaaa__	82	830.0	62900000.0	4500000.0	9.43	5900000.0	3.800000e+09	NaN
69	70	roses_are_rosie	82	820.0	61800000.0	4600000.0	9.72	6000000.0	3.800000e+09	NaN
75	76	milliebobbybrown	80	280.0	57600000.0	4000000.0	8.63	5000000.0	1.100000e+09	United States
78	79	karolg	83	3300.0	55600000.0	3100000.0	10.25	5700000.0	1.010000e+10	India
83	84	zacefron	86	660.0	54500000.0	2300000.0	8.18	4400000.0	1.500000e+09	United States

102	103	thv	83	60.0	49300000.0	15400000.0	25.80	12600000.0	9.874000e+08	NaN
114	115	harrystyles	57	590.0	46900000.0	4700000.0	6.38	2900000.0	2.800000e+09	United States
118	119	zayn	82	160.0	46500000.0	4700000.0	8.81	4000000.0	7.735000e+08	United States
120	121	travisscott	78	3200.0	46200000.0	3000000.0	5.71	2600000.0	9.600000e+09	United States
138	139	badbunnypr	83	10.0	42100000.0	3700000.0	13.09	5400000.0	6.750000e+07	NaN

140	141	j.m	83	20.0	41900000.0	14200000.0	26.41	11000000.0	3.681000e+08	NaN
156	157	georginagio	74	730.0	39100000.0	2200000.0	8.56	3300000.0	1.600000e+09	NaN
177	178	kimberly.loaiza	78	590.0	35500000.0	2600000.0	5.23	1800000.0	1.600000e+09	Mexico

In [21]:

```
top10_er = insta_df.drop(["rank",
"influence_score","posts","avg_likes","new_post_avg_like",
"total_likes","country"], axis = 1)
top10_er.head(10)
```

Out[21]:

	channel_i	follower	60_day_en

	nfo	s	g_rate
0	cristiano	475800000.0	1.39
1	kyliejenner	366200000.0	1.62
2	leomessi	357300000.0	1.24
3	selenagomez	342700000.0	0.97
4	therock	334100000.0	0.20
5	kimkardashian	329200000.0	0.88

6	arianagrande	327700000.0	1.20
7	beyonce	272800000.0	0.76
8	khloekardashian	268300000.0	0.35
9	justinbieber	254500000.0	0.59

In [22]:

```
replace = {'b': 'e9', 'm': 'e6', 'k': 'e3', '%': ''}

converted_data = top10_er['60_day_eng_rate'].replace(replace, regex=True).astype(float)
converted_data.head()
```

Out[22]:

```
0    1.39
```

```
1    1.62
2    1.24
3    0.97
4    0.20
Name: 60_day_eng_rate, dtype: float64
```

In [23]:

```python
# Check for missing values
insta_df.isnull().sum()

# Fill missing engagement rates with the average engagement rate
insta_df['60_day_eng_rate'].fillna(insta_df['60_day_eng_rate'].
mean(), inplace=True)

# Drop rows with any missing values
insta_df.dropna(inplace=True)



print("Data Cleaned")
```

```
Data Cleaned


/tmp/ipykernel_18/1562330709.py:5: FutureWarning: A value is
trying to be set on a copy of a DataFrame or Series through
```

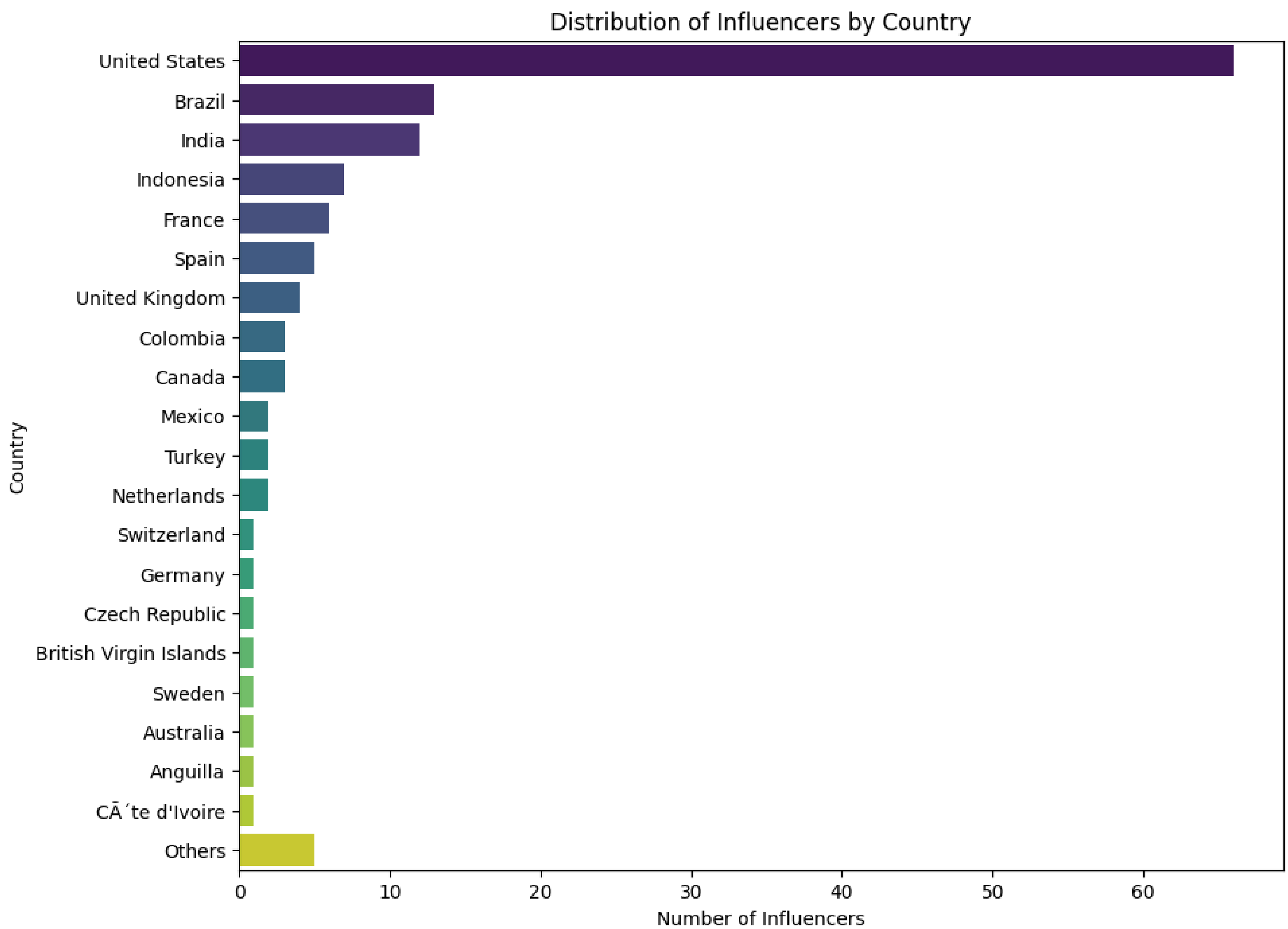
```
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method
will never work because the intermediate object on which we are
setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)',
try using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace
on the original object.



insta_df['60_day_eng_rate'].fillna(insta_df['60_day_eng_rate'].
mean(), inplace=True)
```

In [24]:

```
country= insta_df['country'].value_counts()[:20].to_list()
name_countries =
insta_df['country'].value_counts().index[:20].to_list()

name_countries.append("Others")
max20 = sum(country)
others = len(insta_df) - max20
```

```
country.append(others)


plt.figure(figsize=(10, 8))
sns.barplot(x=country, y=name_countries, palette='viridis')
plt.title('Distribution of Influencers by Country')
plt.xlabel('Number of Influencers')
plt.ylabel('Country')
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1765: FutureWarning: unique with argument that is not not a Series, Index, ExtensionArray, or np.ndarray is deprecated and will raise in a future version.
  order = pd.unique(vector)
```

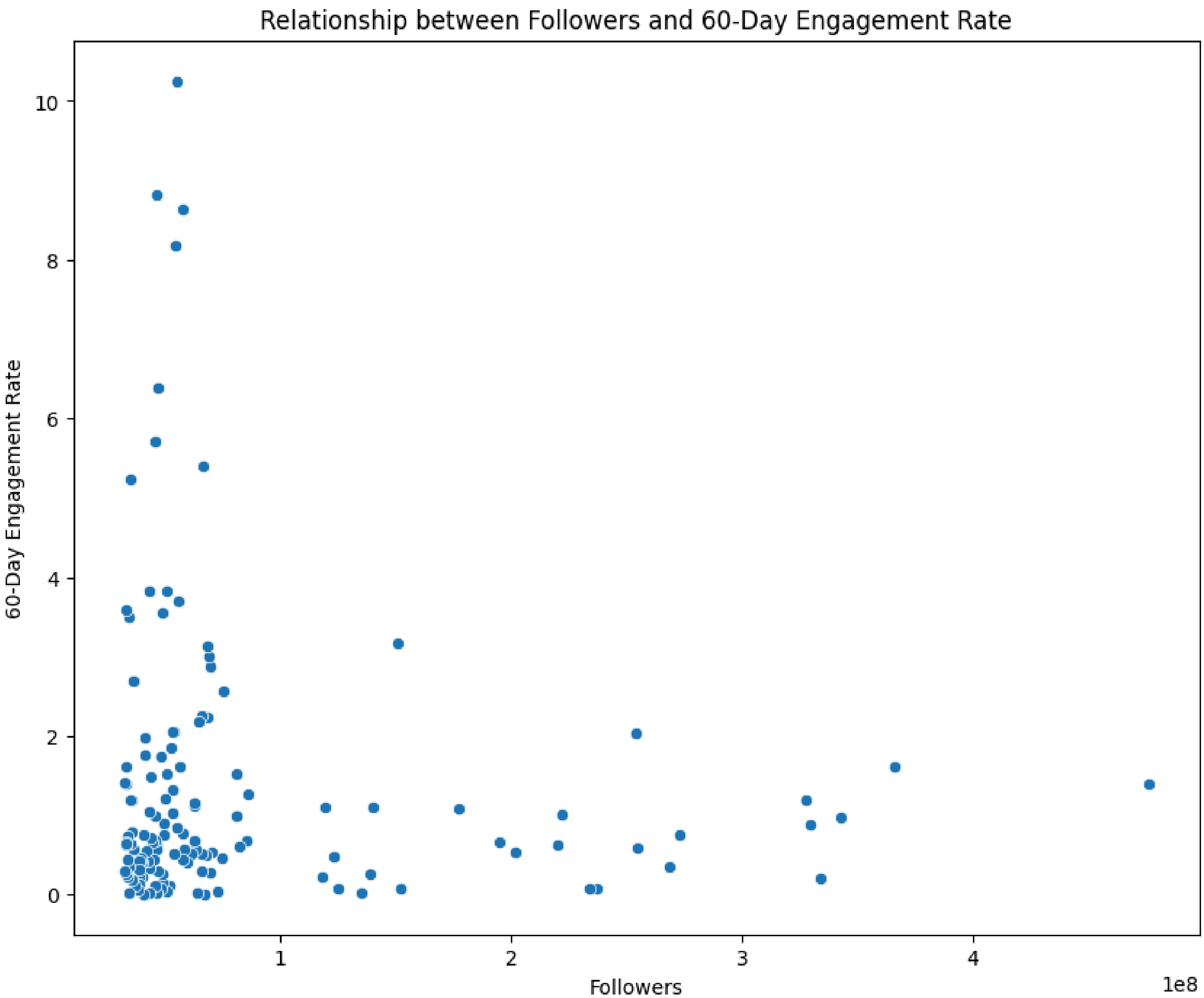


We were trying to see which influencer has high engagement rate so here is a relationship between followers and engagement rate as you can see on x axis we have followers and on y axis we have engagement rate

In [25]:

```
plt.figure(figsize=(10, 8))
sns.scatterplot(x='followers', y='60_day_eng_rate',
data=insta_df)
plt.title('Relationship between Followers and 60-Day Engagement
Rate')
plt.xlabel('Followers')
```

```
plt.ylabel('60-Day Engagement Rate')
plt.show()
```

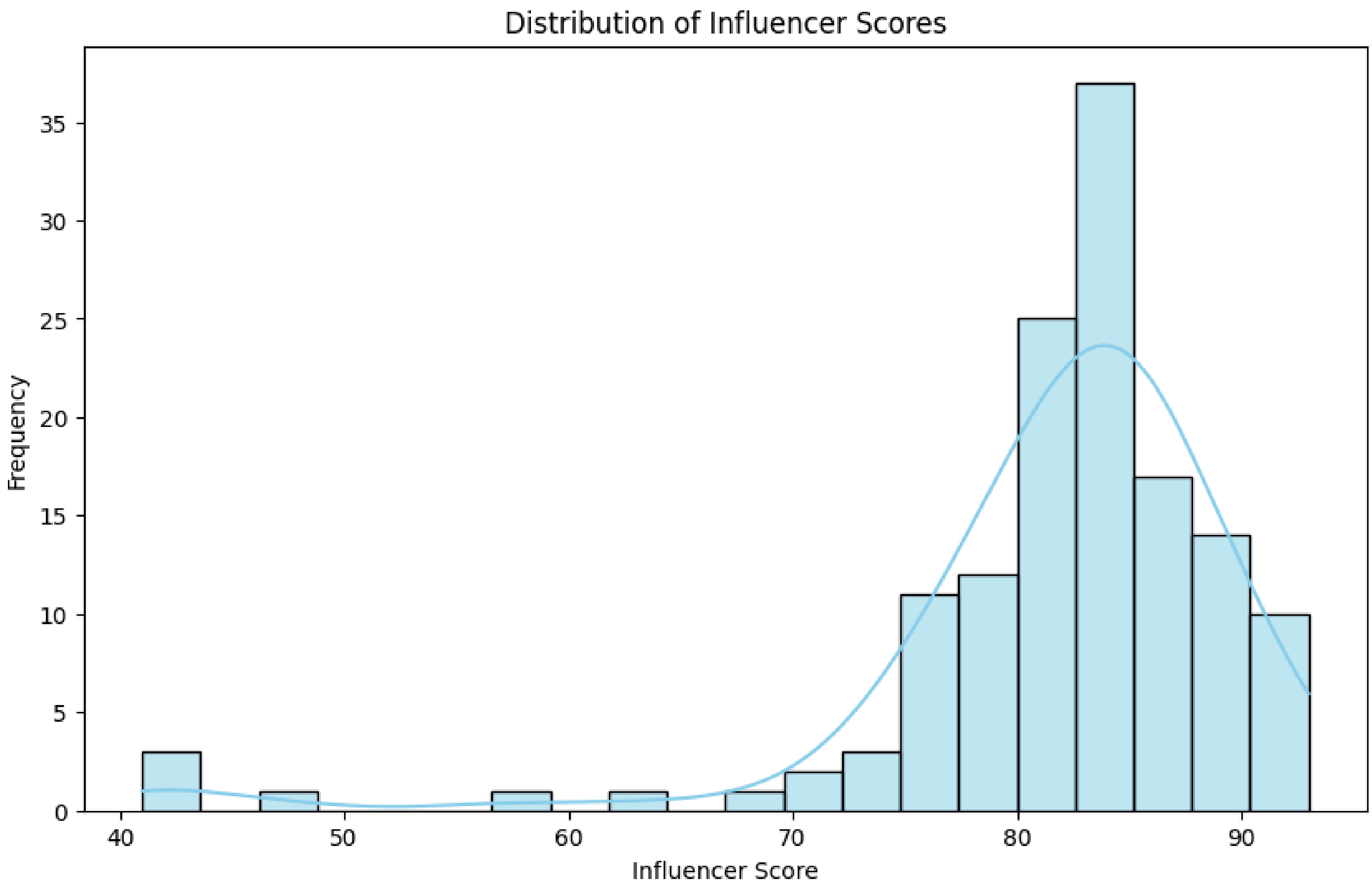


In [26]:

```
plt.figure(figsize=(10, 6))
sns.histplot(insta_df['influence_score'], kde=True,
color='skyblue')
```

```python
plt.title('Distribution of Influencer Scores')
plt.xlabel('Influencer Score')
plt.ylabel('Frequency')
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [ ]:

To make predictions with this data, we'll need to select a target variable. For example, let's predict *engagement rate* based on features like *followers*, *category*, and *country*. Here's a concise approach, including data preparation, model selection, training, and making predictions.

Steps for Prediction:

Data Preparation

Feature Selection and Encoding

Splitting the Data

Model Selection and Training

Making Predictions

Let’s go through this in code using a basic regression model

In [27]:

```
# Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder
```

In [28]:

```
# Feature Selection and Encoding
# Selecting features and encoding categorical variables
features = insta_df[['followers', 'influence_score',
'country']]
target = insta_df['60_day_eng_rate']
```

In [29]:

```
# Encoding categorical variables
encoder = LabelEncoder()
# features['category_encoded'] =
encoder.fit_transform(features['category'])
features['country_encoded'] =
encoder.fit_transform(features['country'])

print("Encoding completed")
```

```
Encoding completed


/tmp/ipykernel_18/177526071.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a
DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy
  features['country_encoded'] =
encoder.fit_transform(features['country'])
```

In [30]:

```
features = features[['followers', 'influence_score',
'country_encoded']]

features
```

Out[30]:

	followers	influence_score	country_encoded
0	4758000000.0	92	17
1	3662000000.0	91	23
3	3427000000.0	93	23

4	334100000.0	91	23
5	329200000.0	91	23
...	...	...	...
195	33200000.0	71	23
196	33200000.0	81	8
197	33200000.0	79	2
198	33000000.0	78	23

199	32800000.0	80	11

138 rows × 3 columns

In [31]:

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features,
target, test_size=0.1, random_state=42)

print("TTS Done")
```

```
TTS Done
```

In [32]:

```
# Linear Regression Model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

print("Training Completed")
```

```
Training Completed
```

In [33]:

```
# Random Forest Model
rf_model = RandomForestRegressor(n_estimators=100,
random_state=42)
rf_model.fit(X_train, y_train)

print("Training Completed")
```

```
Training Completed
```

In [34]:

```
# Making Predictions and Comparing Models
lr_preds = lr_model.predict(X_test)
rf_preds = rf_model.predict(X_test)
```

In [35]:

```
lr_preds
```

Out[35]:

```
array([1.37028635, 1.30255527, 1.01603073, 1.49674945,
1.12804654,
       1.37867911, 1.38950008, 1.32829439, 1.45159326,
1.39365706,

       1.40583913, 1.28780069, 1.42115805, 1.54741186])
```

In [ ]:

In [36]:

```
# Evaluate the models
# lr_mse = mean_squared_error(y_test, lr_predictions)
# rf_mse = mean_squared_error(y_test, rf_predictions)

# # Print out the results
# print("Linear Regression MSE:", lr_mse)
# print("Random Forest MSE:", rf_mse)
```

In [ ]:

In [37]:

```
# # Plotting Actual vs Predicted Engagement Rate
# plt.figure(figsize=(10, 5))
# sns.scatterplot(x=y_test, y=y_pred, color='blue')
# plt.plot([y_test.min(), y_test.max()], [y_test.min(),
y_test.max()], 'r--')  # Line for perfect predictions
# plt.xlabel("Actual Engagement Rate")
# plt.ylabel("Predicted Engagement Rate")
# plt.title("Actual vs Predicted Engagement Rate")
# plt.show()

# # Partial Dependency Plots for each feature
# fig, axes = plt.subplots(1, 3, figsize=(18, 5))
# sns.scatterplot(ax=axes[0], x=X_test['followers'], y=y_test,
label="Actual", color='blue')
# sns.lineplot(ax=axes[0], x=X_test['followers'],
y=lr_model.predict(X_test), label="Predicted", color='red')
# axes[0].set_title("Followers vs Engagement Rate")
# axes[0].set_xlabel("Followers")
# axes[0].set_ylabel("Engagement Rate")

# sns.scatterplot(ax=axes[1], x=X_test['influence_score'],
y=y_test, label="Actual", color='blue')
# sns.lineplot(ax=axes[1], x=X_test['influence_score'],
y=lr_model.predict(X_test), label="Predicted", color='red')
# axes[1].set_title("Influence Score vs Engagement Rate")
```

```
# axes[1].set_xlabel("Influence Score")
# axes[1].set_ylabel("Engagement Rate")

# sns.scatterplot(ax=axes[2], x=X_test['country_encoded'],
y=y_test, label="Actual", color='blue')
# sns.lineplot(ax=axes[2], x=X_test['country_encoded'],
y=lr_model.predict(X_test), label="Predicted", color='red')
# axes[2].set_title("Country Encoded vs Engagement Rate")
# axes[2].set_xlabel("Country Encoded")
# axes[2].set_ylabel("Engagement Rate")

# plt.tight_layout()
# plt.show()
```

In [38]:

```
# Create a DataFrame with test features and predictions
predictions_df = X_test.copy()
predictions_df['Actual Engagement Rate'] = y_test.values
predictions_df['Predicted Engagement Rate'] = lr_preds

# Display the first few rows of the predictions DataFrame
predictions_df.head()
```

Out[38]:

	followers	influence_score	country_encoded	Actual Engagement Rate	Predicted Engagement Rate
122	45900000.0	80	15	1.00	1.370286
149	40000000.0	85	17	0.41	1.302555
13	222200000.0	91	23	1.01	1.016031
33	85900000.0	74	22	1.26	1.496749
85	53900000.0	86	2	0.51	1.128047

In [39]:

###

Now to perform classification on this dataset we can do it by transforming it into a classification problem. For example, you could classify influencers based on their engagement rate into categories like "Low," "Medium," or "High." Here's how to proceed:

Define Classes: Convert the engagement rate into categorical classes. Train-Test Split: Use followers, influence score, and encoded country as features. Build and Train a Classifier: A Random Forest Classifier can work well for this. Evaluate the Model: Check the classification accuracy.

In [40]:

```
insta_df.head()
```

Out[40]:

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	country
0	1	cristiano	92	3300.0	475800000.0	8700000.0	1.39	6500000.0	2.900000e+10	Spain

1	2	kyliejenner	91	6900.0	366200000.0	8300000.0	1.62	5900000.0	5.740000e+10	United States
3	4	selenagomez	93	1800.0	342700000.0	6200000.0	0.97	3300000.0	1.150000e+10	United States
4	5	therock	91	6800.0	334100000.0	1900000.0	0.20	665300.0	1.250000e+10	United States
5	6	kimkardashian	91	5600.0	329200000.0	3500000.0	0.88	2900000.0	1.990000e+10	United States

In [41]:

```
bins = [0, 1, 3, insta_df['60_day_eng_rate'].max()]
labels = ['Low', 'Medium', 'High']
```

```python
insta_df['engagement_rate_class'] =
pd.cut(insta_df['60_day_eng_rate'], bins=bins, labels=labels)

insta_df.head(20)
```

Out[41]:

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	country	engagement_rate_class
0	1	cristiano	92	3300.0	47580000 0.0	870000 0.0	1.39	650000 0.0	2.900 000e +10	Spain	Medium
1	2	kyliejenner	91	6900.0	36620000 0.0	830000 0.0	1.62	590000 0.0	5.740 000e +10	United States	Medium

3	4	selena gomez	93	1800.0	34270000 0.0	6200000 0.0	0.97	3300000 0.0	1.150000e+10	United States	Low
4	5	therock	91	6800.0	33410000 0.0	1900000 0.0	0.20	665300.0	1.250000e+10	United States	Low
5	6	kimkardashian	91	5600.0	32920000 0.0	3500000 0.0	0.88	2900000 0.0	1.990000e+10	United States	Low
6	7	ariana grand	92	5000	32770000	370000	1.20	3900000 0.0	1.840 000e	United	Medium

		e		.0	0.0	0.0			+10	States	
7	8	beyonce	92	2000.0	27280000 0.0	3600000 0.0	0.76	2000000.0	7.400000e+09	United States	Low
8	9	khloekardashian	89	4100.0	26830000 0.0	2400000 0.0	0.35	926900.0	9.800000e+09	United States	Low
9	10	justinbieber	91	7400.0	25450000 0.0	1900000 0.0	0.59	1500000.0	1.390000e+10	Canada	Low
1	1	kendal	90	660.	25400000	550000	2.04	510000	3.700000e	Unite	Medium

0	1	ljenner		0	0.0	0.0		0.0	+09	d States	
11	12	natgeo	91	10000.0	23700000 0.0	302200.0	0.07	159300.0	3.000000e+09	United States	Low
12	13	nike	90	950.0	23410000 0.0	329000.0	0.08	181800.0	3.136000e+08	United States	Low
13	14	taylorswift	91	530.0	22220000 0.0	2400000.0	1.01	2300000.0	1.300000e+09	United Stat	Medium

										es	
14	15	jlo	89	3200.0	220400000.0	1700000.0	0.62	1400000.0	5.300000e+09	United States	Low
16	17	nickiminaj	90	6400.0	201600000.0	2100000.0	0.53	1000000.0	1.350000e+10	United States	Low
17	18	kourtneykardash	89	4400.0	195200000.0	1800000.0	0.67	1300000.0	7.700000e+09	United States	Low

19	20	neymarjr	90	5300.0	17710000 0.0	2700000 0.0	1.09	1900000 0.0	1.410000e +10	Brazil	Medium
21	22	kevinhart4real	88	8200.0	15200000 0.0	522000.0	0.08	115200.0	4.300000e +09	United States	Low
22	23	zendaya	87	3500.0	15070000 0.0	5800000 0.0	3.17	4800000 0.0	2.060000e +10	United States	High
23	24	iamcardib	75	1600.0	14050000 0.0	3100000 0.0	1.10	1500000 0.0	5.000000e +09	United Stat	Medium

										es	

In [42]:

```python
high_engagement_df = insta_df[insta_df['engagement_rate_class'] == "High"]
high_engagement_df
```

Out[42]:

	rank	channel_info	influence_score	posts	followers	avg_likes	60_day_eng_rate	new_post_avg_like	total_likes	country	engagement_rate_class
22	23	zendaya	87	3500.0	150700000.0	5800000.0	3.17	4800000.0	2.060000e+10	United States	High
50	51	narendramodi	85	540.	68900000	290000	3.01	2000000.0	1.600000e	India	High

				0	.0	0.0			+09		
51	52	aliaabhatt	82	1800.0	68700000.0	1800000.0	3.14	2100000.0	3.300000e+09	India	High
56	57	bts.bighitofficial	78	1200.0	66900000.0	4100000.0	5.40	3600000.0	4.900000e+09	Uruguay	High
75	76	milliebobbybrown	80	280.0	57600000.0	4000000.0	8.63	5000000.0	1.100000e+09	United States	High
77	78	chrishemsworth	86	880.0	55900000.0	2800000.0	3.69	2100000.0	2.500000e+09	Australia	High

78	79	karolg	83	3300.0	55600000.0	3100000.0	10.25	5700000.0	1.010000e+10	India	High
83	84	zacefron	86	660.0	54500000.0	2300000.0	8.18	4400000.0	1.500000e+09	United States	High
97	98	adele	84	420.0	50700000.0	4700000.0	3.82	1900000.0	2.000000e+09	United States	High
103	104	lelepons	81	2500.0	49200000.0	2400000.0	3.55	1700000.0	6.100000e+09	United State	High

										s	
114	115	harrystyles	57	590.0	46900000.0	4700000.0	6.38	2900000.0	2.800000e+09	United States	High
118	119	zayn	82	160.0	46500000.0	4700000.0	8.81	4000000.0	7.735000e+08	United States	High
120	121	travisscott	78	3200.0	46200000.0	3000000.0	5.71	2600000.0	9.600000e+09	United States	High

132	133	hrithikroshan	85	580.0	43700000.0	1600000.0	3.82	1600000.0	9.499000e+08	CÃ´te d'Ivoire	High
177	178	kimberly.loaiza	78	590.0	35500000.0	2600000.0	5.23	1800000.0	1.600000e+09	Mexico	High
184	185	danna paola	68	1900.0	34700000.0	1500000.0	3.49	1200000.0	2.800000e+09	Mexico	High
191	192	danbilzerian	84	1400.0	33600000.0	2000000.0	3.58	1200000.0	2.800000e+09	Canada	High

In [43]:

```
# Encode the 'country' column
encoder = LabelEncoder()
insta_df['country_encoded'] =
```

```
encoder.fit_transform(insta_df['country'])
```

In [44]:

```
# Select features and target for classification
features = insta_df[['followers', 'influence_score',
'country_encoded']]
target = insta_df['engagement_rate_class']
```

In [45]:

```
# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(features,
target, test_size=0.2, random_state=42)
print("TTS Completed")
```

```
TTS Completed
```

In [46]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report,
accuracy_score
```

In [47]:

```
# Build and train the Random Forest Classifier
classifier = RandomForestClassifier(n_estimators=100,
random_state=42)
classifier.fit(X_train, y_train)

print("Model Trained")
```

```
Model Trained
```

In [48]:

```
# Make predictions on the test set
y_pred = classifier.predict(X_test)

y_pred
```

Out[48]:

```
array(['Low', 'Low', 'Low', 'Low', 'Low', 'Low', 'Low', 'Low',
'Low',
       'Low', 'Low', 'Low', 'Low', 'Medium', 'Low', 'Medium',
```

```
'Low',
       'Low', 'Low', 'Low', 'Low', 'Low', 'Low', 'Low', 'Low',
'Low',
       'Low', 'Low'], dtype=object)
```

In [49]:

```python
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:" + str(float(accuracy)*100) + "%")
print("\nClassification Report:\n",
classification_report(y_test, y_pred))

print("Ideal f1: >0.7")
print("Ideal recall =: 1.0")
```

```
Accuracy:50.0%

Classification Report:
               precision    recall  f1-score   support

        High       0.00      0.00      0.00         4
         Low       0.50      1.00      0.67        13
      Medium       0.50      0.09      0.15        11

    accuracy                           0.50        28
```

```
   macro avg       0.33      0.36      0.27        28
weighted avg       0.43      0.50      0.37        28

Ideal f1: >0.7
Ideal recall =: 1.0


/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classi
fication.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classi
fication.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.10/site-packages/sklearn/metrics/_classi
fication.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

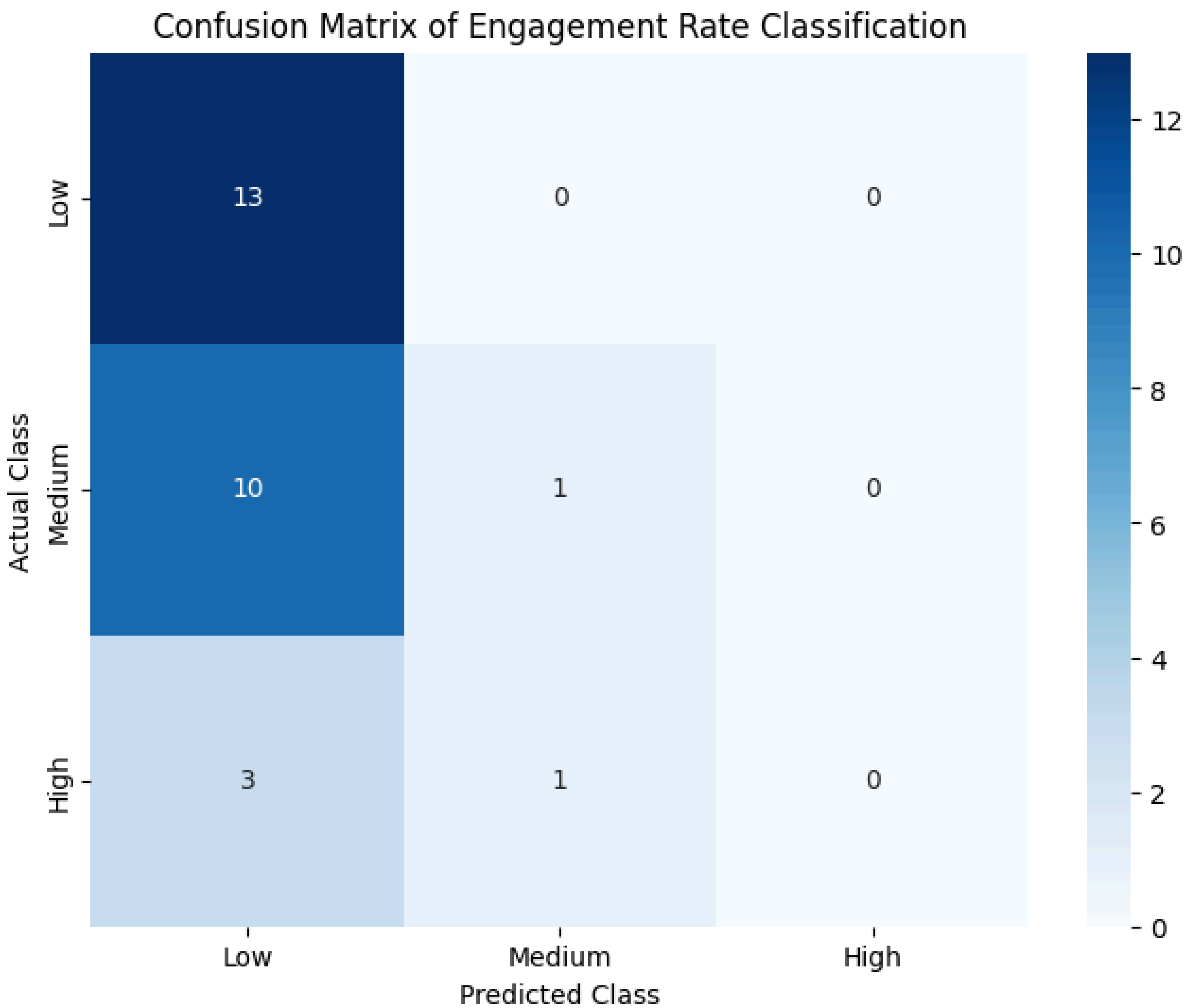
In [50]:

```
from sklearn.metrics import confusion_matrix
```

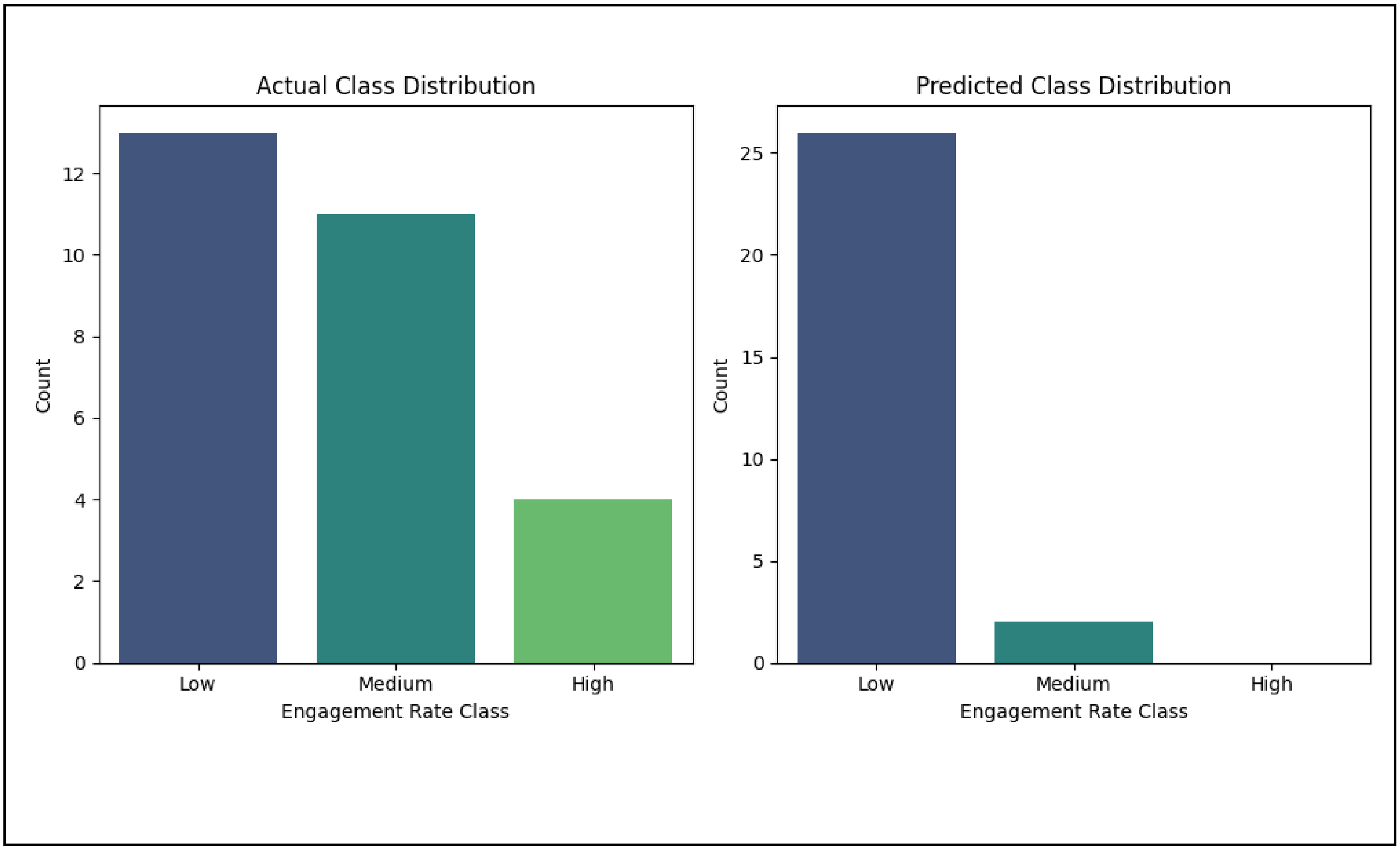
In [51]:

```
# Generate the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred, labels=labels)

# Plotting the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted Class')
plt.ylabel('Actual Class')
plt.title('Confusion Matrix of Engagement Rate Classification')
plt.show()

# Bar plot for the distribution of actual vs. predicted classes
plt.figure(figsize=(10, 5))

# Plot actual class distribution
plt.subplot(1, 2, 1)
sns.countplot(x=y_test, order=labels, palette="viridis")
```

```python
plt.title("Actual Class Distribution")
plt.xlabel("Engagement Rate Class")
plt.ylabel("Count")

# Plot predicted class distribution
plt.subplot(1, 2, 2)
sns.countplot(x=y_pred, order=labels, palette="viridis")
plt.title("Predicted Class Distribution")
plt.xlabel("Engagement Rate Class")
plt.ylabel("Count")

plt.tight_layout()
plt.show()
```




```
/opt/conda/lib/python3.10/site-packages/seaborn/categorical.py:
641: FutureWarning: The default of observed=False is deprecated
and will be changed to True in a future version of pandas. Pass
observed=False to retain current behavior or observed=True to
adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```



[Reference link]

You can practice and get experience from here for sql project

SQL Project: Analysis of Top Instagram Influencers Data

This project is designed for someone with approximately five years of experience in data analysis and SQL, aiming to dive into exploratory data analysis (EDA), data transformations, and visualizations for insight extraction on Instagram influencers.

Dataset Schema

rank: Rank of the influencer

channel_info: Name or handle of the influencer

influence_score: Score reflecting influencer's overall impact

posts: Number of posts by the influencer

followers: Total follower count

avg_likes: Average number of likes per post

60_day_eng_rate: Engagement rate over the past 60 days

new_post_avg_like: Average likes on recent posts

total_likes: Total likes across all posts

country: Country of the influencer

Step 1: Database Setup and Data Insertion

First, create the database and import the data. We will use SQL for data manipulation, and the project may also involve Python for visualization.

Sql code

```sql
-- Create the database
CREATE DATABASE InstagramInfluencers;
USE InstagramInfluencers;

-- Create the table
CREATE TABLE influencers (
    rank INT PRIMARY KEY,
    channel_info VARCHAR(100),
    influence_score DECIMAL(5, 2),
    posts INT,
    followers BIGINT,
    avg_likes INT,
    sixty_day_eng_rate DECIMAL(4, 2),
    new_post_avg_like INT,
    total_likes BIGINT,
    country VARCHAR(50)
);
```

Insert Data

Load the data into this table from a CSV file or directly insert sample data if testing manually.

Sql code

```sql
–  Sample insert
INSERT INTO influencers (rank, channel_info, influence_score,
```

```
posts, followers, avg_likes, sixty_day_eng_rate,
new_post_avg_like, total_likes, country)
VALUES
(1, 'influencer1', 85.50, 500, 1000000, 20000, 4.50, 21000,
10000000, 'USA'),
(2, 'influencer2', 80.25, 450, 950000, 18000, 4.20, 18500,
8500000, 'UK');
```

Step 2: Exploratory Data Analysis (EDA)

1. Distribution of Followers

sql code

```sql
-- Find the distribution of followers
SELECT
    country,
    AVG(followers) AS avg_followers,
    MAX(followers) AS max_followers,
    MIN(followers) AS min_followers
FROM influencers
GROUP BY country
ORDER BY avg_followers DESC;
```

This query helps to understand which countries have the most followed influencers.

2. Top Influencers by Influence Score

Sql code

```sql
-- Retrieve top influencers based on influence_score
SELECT
    rank,
    channel_info,
    influence_score,
    followers,
    avg_likes
FROM influencers
ORDER BY influence_score DESC
LIMIT 10;
```

This identifies the highest-ranking influencers by influence score, giving insights into their audience reach and engagement.

3. Engagement Rate Analysis

Sql code

```sql
-- Calculate engagement rate statistics
SELECT
    AVG(sixty_day_eng_rate) AS avg_eng_rate,
    MAX(sixty_day_eng_rate) AS max_eng_rate,
    MIN(sixty_day_eng_rate) AS min_eng_rate
FROM influencers;
```

This query provides average, maximum, and minimum engagement rates to assess overall engagement trends among top influencers.

4. Influencers with High Engagement but Low Followers

Sql code

```sql
-- Identify influencers with high engagement but relatively low followers
SELECT
    channel_info,
    followers,
    sixty_day_eng_rate
FROM influencers
WHERE followers < 500000
AND sixty_day_eng_rate > 5
ORDER BY sixty_day_eng_rate DESC;
```

This finds influencers who, despite a lower follower count, have high engagement rates, making them potential high-return micro-influencers.

Step 3: Visualization and Analysis with Python

After querying, we can use Python with libraries like `matplotlib` and `seaborn` to visualize the data.

Example Visualization: Distribution of Followers by Country

Python code

```python
import pandas as pd
import seaborn as sns
```

```
import matplotlib.pyplot as plt

# Load data into pandas DataFrame
data = pd.read_sql_query("SELECT country, AVG(followers) AS
avg_followers FROM influencers GROUP BY country", connection)

# Plot the data
plt.figure(figsize=(12, 6))
sns.barplot(data=data, x="country", y="avg_followers")
plt.xticks(rotation=45)
plt.title("Average Followers by Country")
plt.xlabel("Country")
plt.ylabel("Average Followers")
plt.show()
```

Visualization: Engagement vs Follower Count

Python code

```
# Query for engagement vs followers
data = pd.read_sql_query("SELECT followers, sixty_day_eng_rate
FROM influencers", connection)

# Scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x="followers",
y="sixty_day_eng_rate", hue="country")
```

```
plt.title("Engagement Rate vs Followers")
plt.xlabel("Followers")
plt.ylabel("60-Day Engagement Rate")
plt.show()
```

Step 4: Advanced Analysis

1. Growth Potential (New Post Like Analysis)

Sql code

```sql
-- Find influencers whose new post average likes have increased by 10% over average likes
SELECT
    channel_info,
    avg_likes,
    new_post_avg_like
FROM influencers
WHERE new_post_avg_like > avg_likes * 1.1
ORDER BY new_post_avg_like DESC;
```

This finds influencers experiencing growth in engagement, useful for targeting those with increasing influence.

2. Country-wise Average Influence Score

Sql code

```sql
-- Find average influence score by country
```

```sql
SELECT
    country,
    AVG(influence_score) AS avg_influence_score
FROM influencers
GROUP BY country
ORDER BY avg_influence_score DESC;
```

Final Report Summary

Influencer Rankings and Distribution:

Analyzed top influencers, their reach, and engagement patterns.

Country-based Trends:

Identified which countries have the most active influencer bases and higher follower counts.

Micro-Influencers with High Engagement:

Discovered lower-followed influencers with significant engagement.

Growth Indicators:

Showcased influencers with rising engagement rates on recent posts.

Reference link