Project 2: Analyzing SUPERCELL Data

For this project I have decided to look at some video game data form mobile platforms. I choose the company SUPERCELL, as they created multiple popular mobile games with million of users. The three games I found API data and analyzed were *Clash of Clans*, *Clash Royale*, and *Brawl Stars*. All the APIs were public and easy to manage. However for many of the API steps, I had to use the AI resource a lot to basically create the entire steps as reading the documentation from the API itself was very difficult to understand. However most of the analysis and choices that were done on this project were soley by me. I did not have AI help analyze, create any of the solutions, or write any explanation on what this data tells. As a forewarning, although these games were created by the same company, they are entirely three different games, so much of the data did not overlap. In this analysis I focused mainly on rankings, clans/clubs in the game, and member count.

```
import pandas as pd
import requests
import plotly.express as px

# How do I read in data with an API and token key?

coc_api_token = "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiIsImtpZCI6IjI4YTMxOGY3LTAwMDAtYTFlYi03ZmExLTJjNzQzM2M2Y2NhNSJ9.eyJpc3MiOiJzdXBlcmNlbGwiLCJhdWQiOiJzdXBlcmNlbGw6Z2FtZWFwaSIsImp0aSI6IjVmMWQ1NTFiLWI2OTItNDI3ZC05ZjIxLWI2MjhiNjQwNzc4MSIsImlhdCI6MTczMTQzNTgzNywic3ViIjoiZGV2ZWxvcGVy
headers_coc = {
    "Authorization": f"Bearer {coc_api_token}"
}
url_coc = "https://api.clashofclans.com/v1/clans"

cr_api_token = "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiIsImtpZCI6IjI4YTMxOGY3LTAwMDAtYTFlYi03ZmExLTJjNzQzM2M2Y2NhNSJ9.eyJpc3MiOiJzdXBlcmNlbGwiLCJhdWQiOiJzdXBlcmNlbGw6Z2FtZWFwaSIsImp0aSI6ImNiMjE2ZTM3LWU1OGQtNGM2MS04YTBmLTk3NmYzYTQ2MzNlMSIsImlhdCI6MTczMTQzNjYyMSwic3ViIjoiZGV2ZWxvcGVy
headers_cr = {
    "Authorization": f"Bearer {cr_api_token}"
}
url_cr = "https://api.clashroyale.com/v1/clans"
```

The first thing I did here was create the URL and header to each API. The intial set up took the most amount of time in this project as I didn't know how to set up or where to look first. Luckily in this analysis, I didn't need to use a time.sleep because the all three APIs were in JSON, making the read in much easier. Additionally, on the actual documentation of these APIs it was very helpful to pull the links from the API. The only thing I needed from the AI was the headers.

```
# Question for AI: How do I 500 clans from Clash of Clans API and Clash Royale API?

# Function to get 500 clans by clan level for Clash of Clans
def get_500_clans_by_level():
    params_coc = {
        "limit": 500,  # Request only the top 500 clans
        "minMembers": 10  # Add a valid filter: minimum number of members
    }

    # Make the request for Clash of Clans
    response_coc = requests.get(url_coc, headers=headers_coc, params=params_coc)

    data_coc = response_coc.json().get("items", [])
    return data_coc

# Function to get 500 clans by clan score for Clash Royale
def get_500_clans_by_score():
    params_cr = {
        "limit": 500,  # Request only the top 100 clans
        "minMembers": 10  # Add a valid filter: minimum number of members
    }

    # Make the request for Clash Royale
    response_cr = requests.get(url_cr, headers=headers_cr, params=params_cr)

    data_cr = response_cr.json().get("items", [])
    return data_cr

# Fetch 500 Clash of Clans clans by clan level
get_500_coc = get_500_clans_by_level()

# Fetch 500 Clash Royale clans by clan score
get_500_cr = get_500_clans_by_score()

# Convert the data for Clash of Clans to DataFrame
df_coc = pd.json_normalize(get_500_coc)

# Convert the data for Clash Royale to DataFrame
df_cr = pd.json_normalize(get_500_cr)

# Display the top 500 clans from both games
print("Clash of Clans DataFrame Length:", len(df_coc))
print("Clash Royale DataFrame Length:", len(df_cr))
```

```
Clash of Clans DataFrame Length: 500
Clash Royale DataFrame Length: 500
```

So above, a lot of the original AI code was wrong, so I did simplify and correct it. When I intially got it to work, one of the things AI suggested was to put a minimum number of clan members, so there are thousands of clans with just 1 or 2 members each. I limited this to only 500 clans each. 10 members is a good indication at a clubs health, as most active clans have active leaders and co-leaders kick out inactive members.

Explanation of Clans and Clubs in the Games

In both Clash of Clans and Clash Royale groups of player can join a clan. Clans can be beneficial to all players, giving them additional troops and support. Additionally in both games there are events specifically for Clans, that all members participate to earn better rewards. In Brawl Stars clubs serve a similar purpose, but they aren't all impactful to the overall game like Clash of Clans and Clash Royale.

```
print(df_coc.columns)
print(df_cr.columns)
```

```
Index(['tag', 'name', 'type', 'isFamilyFriendly', 'clanLevel', 'clanPoints',
       'clanBuilderBasePoints', 'clanCapitalPoints', 'requiredTrophies',
       'warFrequency', 'warWinStreak', 'warWins', 'isWarLogPublic', 'members',
       'labels', 'requiredBuilderBaseTrophies', 'requiredTownhallLevel',
       'location.id', 'location.name', 'location.isCountry',
       'location.countryCode', 'badgeUrls.small', 'badgeUrls.large',
       'badgeUrls.medium', 'capitalLeague.id', 'capitalLeague.name',
       'warLeague.id', 'warLeague.name', 'warTies', 'warLosses',
       'chatLanguage.id', 'chatLanguage.name', 'chatLanguage.languageCode'],
      dtype='object')
Index(['tag', 'name', 'type', 'badgeId', 'clanScore', 'clanWarTrophies',
       'requiredTrophies', 'donationsPerWeek', 'clanChestLevel',
       'clanChestMaxLevel', 'members', 'location.id', 'location.name',
       'location.isCountry', 'location.countryCode'],
      dtype='object')
```

From here we can see that tag, name, type, requiredTrophies, location id, location name, location isCountry, and country code are all shared between these data sets.

```
# Only keeping data that was here
df_coc_clans = df_coc[['tag', 'name', 'type', 'location.name', 'location.id',
       'location.isCountry', 'location.countryCode', 'requiredTrophies', 'members']]
df_coc_clans.columns = ['tag', 'name', 'type', 'location', 'loc_id', 'country_tf', 'country_code', 'requiredTrophies', 'members']
df_coc_clans['game'] = 'Clash of Clans'  # Adding the game source column

# Normalize the common columns for Clash Royale (CR) and add a game source column
df_cr_clans = df_cr[['tag', 'name', 'type', 'location.name', 'location.id',
       'location.isCountry', 'location.countryCode', 'requiredTrophies', 'members']]
df_cr_clans.columns = ['tag', 'name', 'type', 'location', 'loc_id', 'country_tf', 'country_code', 'requiredTrophies', 'members']
df_cr_clans['game'] = 'Clash Royale'  # Adding the game source column

# Concat the two DataFrames
combined_df = pd.DataFrame(pd.concat([df_coc_clans, df_cr_clans], ignore_index = True))
combined_df

# Warning are annoying here.
```

```
<ipython-input-46-3583e6c1cc66>:5: 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/indexing.html#returning-a-view-versus-a-copy

<ipython-input-46-3583e6c1cc66>:11: 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/indexing.html#returning-a-view-versus-a-copy
```

	tag	name	type	location	loc_id	country_tf	country_code	requiredTrophies	members	game
0	#RYUR02R9	[Δπ¶€£$] wings	inviteOnly	Philippines	32000185.0	True	PH	0	13	Clash of Clans
1	#2GYRQQVRG	___	inviteOnly	Honduras	32000109.0	True	HN	0	15	Clash of Clans
2	#2GJUUR8RJ	_boom_panot_	closed	NaN	NaN	NaN	NaN	0	11	Clash of Clans
3	#2YU2GLJGY	`MBAH JENGGOT`	inviteOnly	Indonesia	32000114.0	True	ID	0	10	Clash of Clans
4	#2RUGLU0RR	bebelove	open	Philippines	32000185.0	True	PH	5500	15	Clash of Clans
...	...	...	...	...	...	...	...	...	...	...
995	#G8Y8RUG9	STREET WORKOUT	closed	South Korea	57000216.0	True	KR	0	11	Clash Royale
996	#PYUURYP9	Strongers men	open	Peru	57000184.0	True	PE	5000	15	Clash Royale
997	#QRC2Q809	Stupid Town	inviteOnly	International	57000006.0	False	NaN	0	16	Clash Royale
998	#QRQJYR2L	suomen bobot	open	International	57000006.0	False	NaN	0	10	Clash Royale
999	#QLCJJ9YG	Superbiam	open	International	57000006.0	False	NaN	0	20	Clash Royale

1000 rows × 10 columns

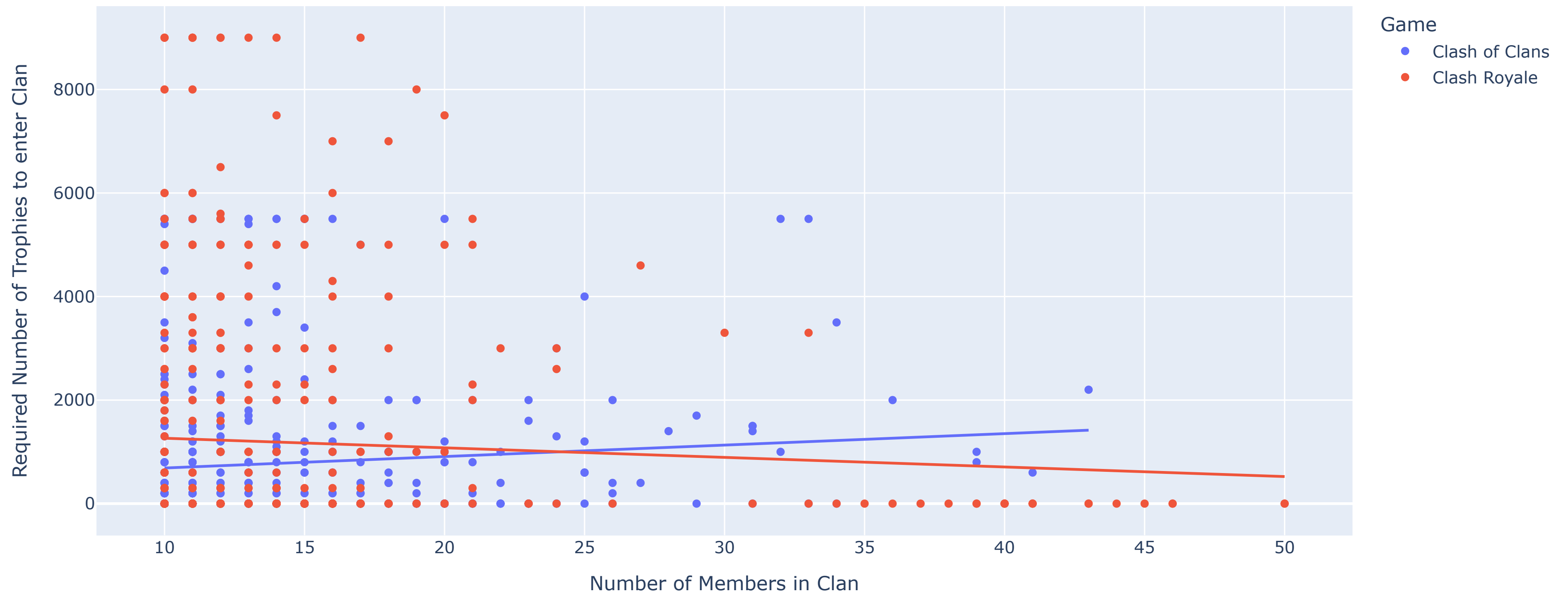
Next steps: Generate code with combined_df | View recommended plots | New interactive sheet

In the block of code above. I combined all similar data. Now all of this data happened to spatial data like location of clans and if they were a country or not. There were lots of missing data especially when it came to country code. But this table was very much usable. The missing values should not play a big role in this analysis. Seeing as the locations of each clan is listed and there are no NA values there. Additionally in this quick analysis between the two games I wanted to analyze clan size to required trophie to enter. Something both games keeps track of.

I wanted to see if the required number of trophies was correlated with the number of members a clan had. I predict the higher number of members less trophies are required to join a clan. This is because clans with a lower entry requirement will have more members able to join.

```
fig = px.scatter(combined_df,
             x = "members",
             y = "requiredTrophies",
             color = "game",
             labels=dict(members="Number of Members in Clan",
                     requiredTrophies="Required Number of Trophies to enter Clan",
                     game = "Game"),
             trendline="ols")

fig.show()
```

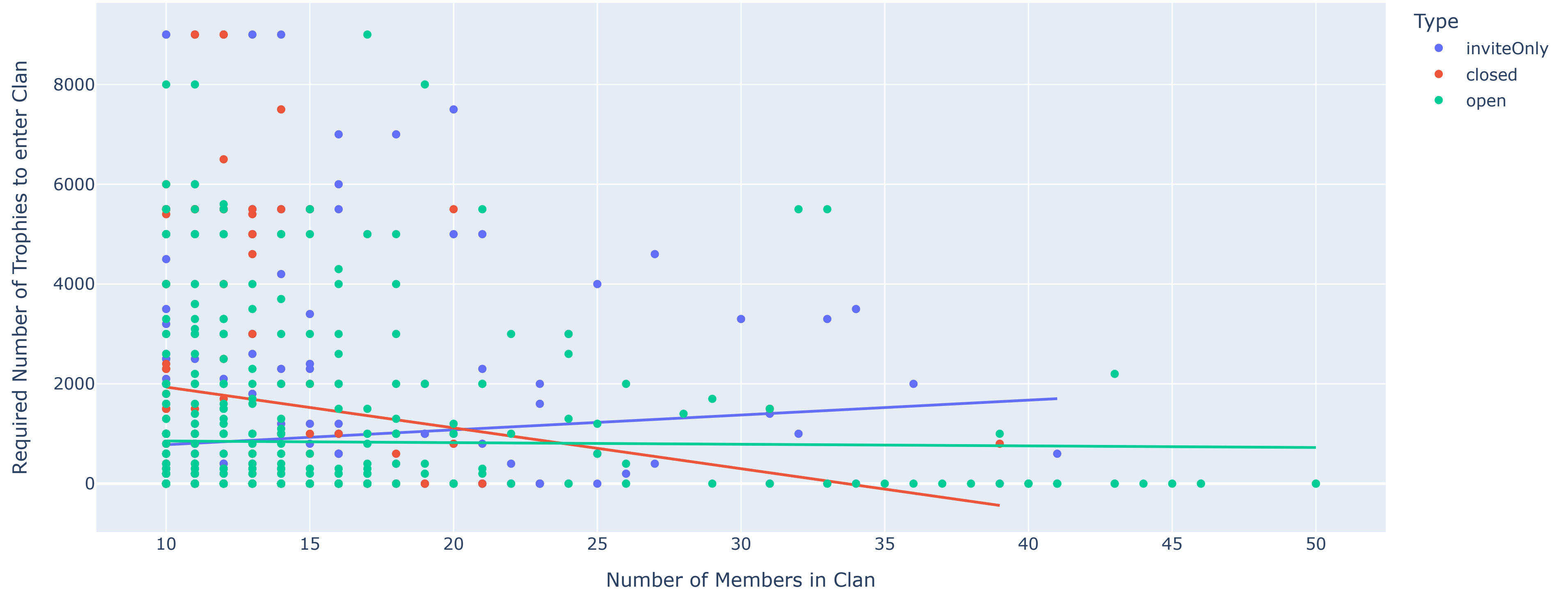


Graph is pretty clear that this is exactly what happened for Clash Royale. It makes sense as in Clash Royale trophy count isn't as relevant as other factors. On the other hand it was a little bit surprising to see Clash of Clans have a positive correlation. But taking a moment to think this also does make sense as well. Here trophy count and maintaining rate is much more important in Clash of Clans. As the entire clan relies on each individuals trophy counts where as Clash Royale that total trophy count doesn't affect much in the rankings. The OLS for this graph doesn't really do this reasoning any justice. There are many more factors are in play here. But this data is not really quantifiabile in the data.

Maybe type of clan may be more telling of something? Not the game?

```
fig = px.scatter(combined_df,
             x = "members",
             y = "requiredTrophies",
             color = "type",
             labels=dict(members="Number of Members in Clan",
                     requiredTrophies="Required Number of Trophies to enter Clan",
                     type = "Type"),
             trendline="ols")

fig.show()
```



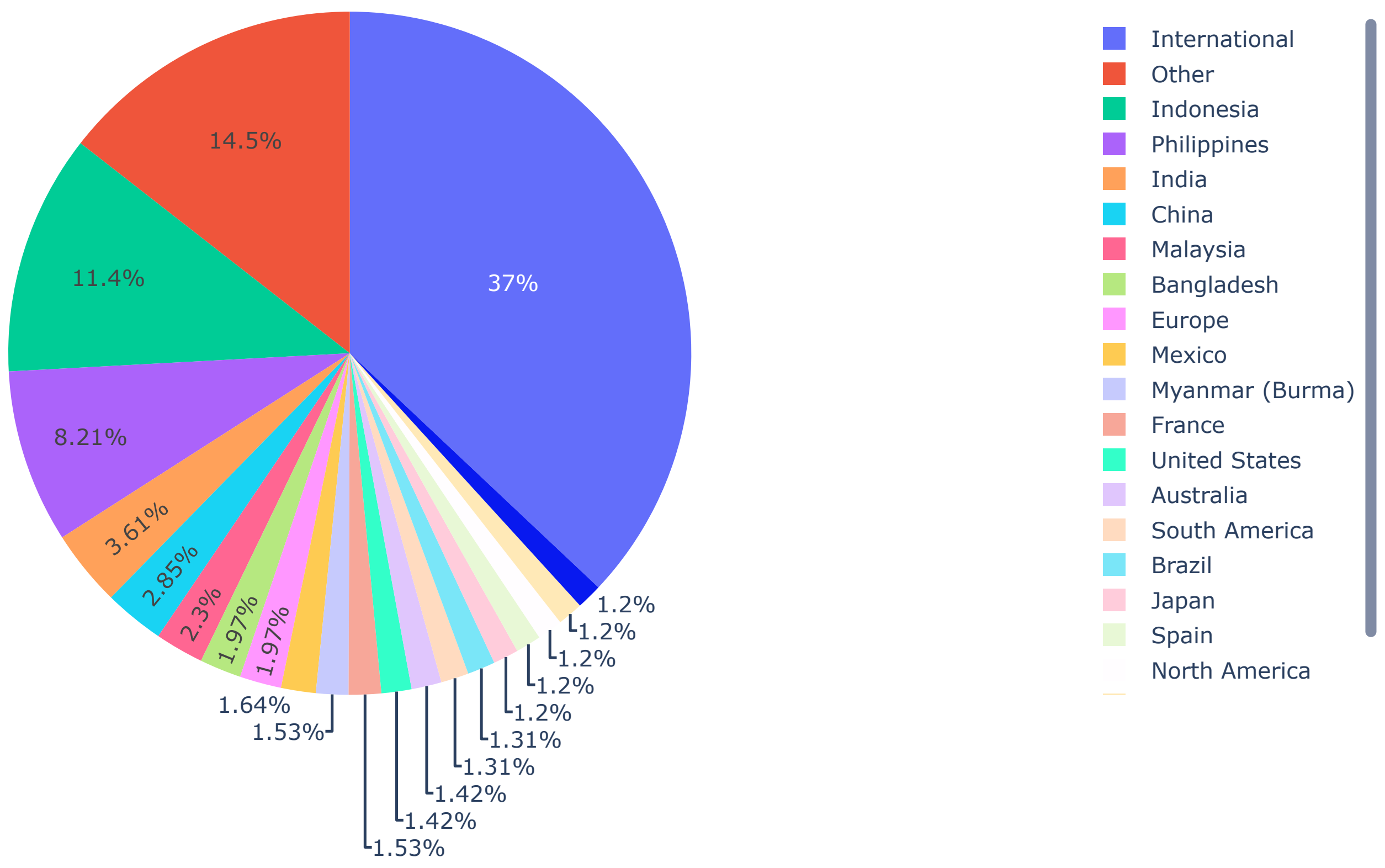
I guess. Closed clans have less members, while open clans do. Surprisingly, invite only clans may have a predicted increase in trophies as the number of members join. This may be because certain clans are picky on who gets in and may invite players whose trophy counts are already really high.

```
country_count = combined_df['location'].value_counts().reset_index()
country_count.columns = ['location', 'count']

# How do I do an if else for countries that have a count less than 10 label them as other? AI help here.
country_count['location'] = ['Other' if count < 10 else location for location, count in zip(country_count['location'], country_count['count'])]

fig = px.pie(country_count,
           values = "count",
           names = "location")

fig.show()
```

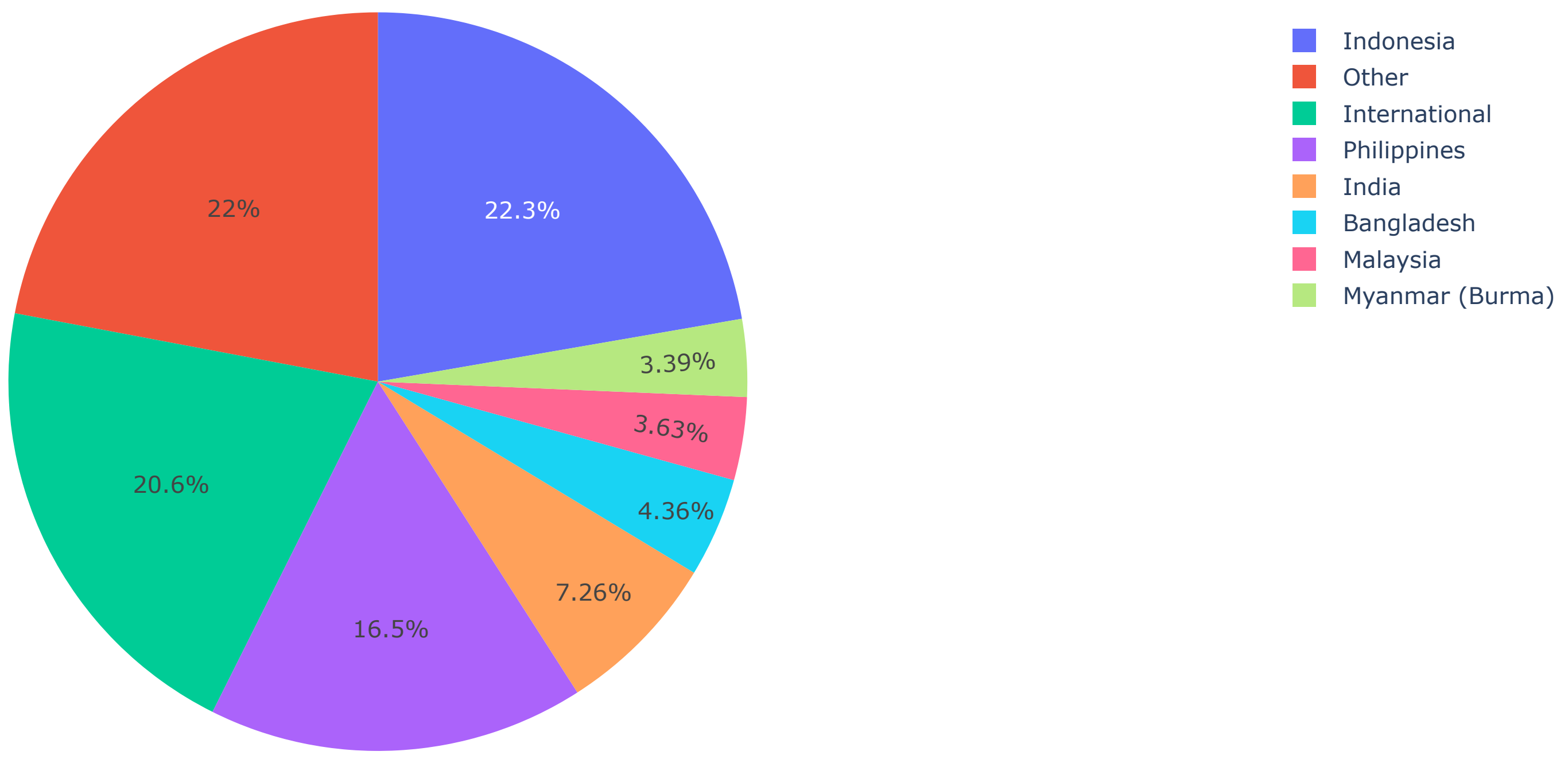


A quick demographic for players of both Clash of Clans and Clash Royale. Here I kept any location above a count of 10 since a lot of countries have 1 player each in the data set. Surprisingly a majority of players with a specific country are found in Indonesia and Phillipines. While a majority of players report being an international player (did not specify) or are from a country with 9 or less players.

```
country_count_coc = df_coc_clans['location'].value_counts().reset_index()
country_count_coc.columns = ['location', 'count']
country_count_coc['location'] = ['Other' if count < 10 else location for location, count in zip(country_count_coc['location'], country_count_coc['count'])]

fig = px.pie(country_count_coc,
             values = "count",
             names = "location")

fig.show()
```

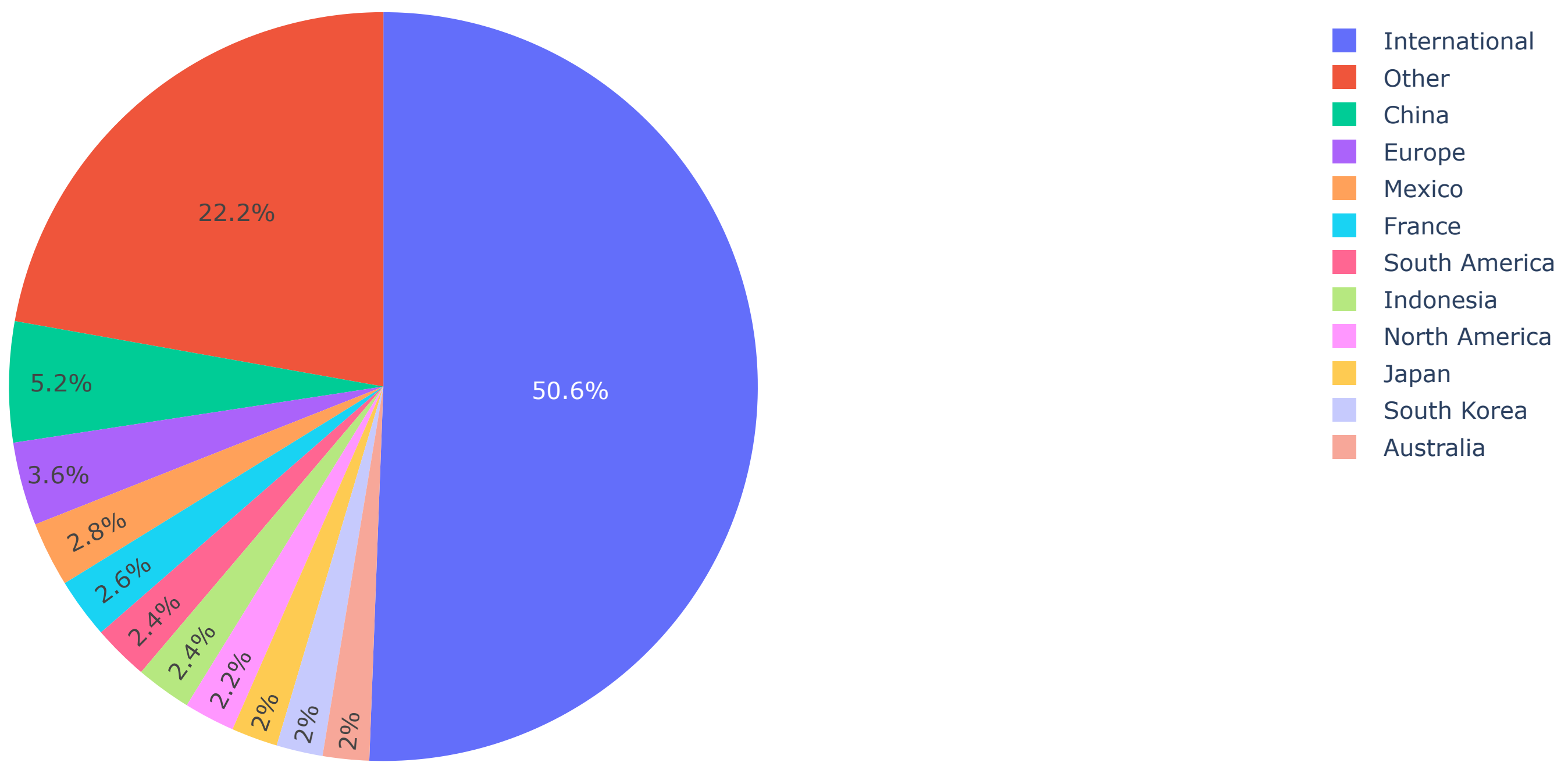


I also created a count for Clash of Clans only. Here it's clear that most players are Indonesian, international, and Philippines. While a majority of other players are from other smaller player count countries like India, Bangladesh, or Malaysia. Notice how the trend with SuperCell games is that they are mostly from SouthEastern Asian countries?

```
country_count_cr = df_cr_clans['location'].value_counts().reset_index()
country_count_cr.columns = ['location', 'count']
country_count_cr['location'] = ['Other' if count < 10 else location for location, count in zip(country_count_cr['location'], country_count_cr['count'])]

fig = px.pie(country_count_cr,
             values = "count",
             names = "location")

fig.show()
```



Here Clash Royale is much less diverse in where a player comes from. Most of the are international (or did not tell country). With the rest coming from countries with 9 or less players. This is followed by China, Europe, Mexico, France, etc. I think this was expected since Clash Royale actually has an E-sports league, so players from more international countries appear more.

Brawl Stars

The next API I used was the Brawl Stars API. Same process here. But the Brawl Stars API didn't have the clan JSON directory like the Clash of Clans or Clash Royale ones did. The closest they had was a global ranking. So I used that instead. For Clash of Clans and Clash Royale I used the international rankings since they could give me something close to real clan data that would match a "global" ranking.

Note: The max number of members is 50 in Clash of Clans and Clash Royale while the max number of members is 30 in Brawl Stars.

Same process here

```
bs_api_token = "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiIsImtpZCI6IjI4YTMxOGY3LTAwMDAtYTFlYi03ZmExLTJjNzQzM2M2Y2NhNSJ9.eyJpc3MiOiJzdXBlcmNlbGwiLCJhdWQiOiJzdXBlcmNlbGw6Z2FtZWFwaSIsImp0aSI6IjM2MDk3ZDk3LTc4NTctNDJlOS1iYjY1LWY3ODY1MjI1NWFkYSIsImlhdCI6MTczMTQ0MTc3OCwic3ViIjoiZGV2ZWxvcGVy
headers_bs = {
    "Authorization": f"Bearer {bs_api_token}"
}
url_bs = "https://api.brawlstars.com/v1/rankings/global/clubs"
```

```
# Function to get the top 200 clubs for Brawl Stars
def get_top_200_rank_clubs_bs():
    params_bs = {
        "limit": 200,  # Request only the top 200 clubs
    }

    # Make the request for Brawl Stars
    response_bs = requests.get(url_bs, headers=headers_bs, params=params_bs)

    data_bs = response_bs.json().get("items", [])
    return data_bs

# Fetch top 200 Brawl Stars Players by club level
top_200_bs = get_top_200_rank_clubs_bs()

df_top_bs = pd.json_normalize(top_200_bs)
df_top_bs["game"] = "Brawl Stars"
df_top_bs
```

	tag	name	badgeId	trophies	rank	memberCount	game
0	#2CJ8YYJV9	Thunder ⚡	8000053	2799265	1	30	Brawl Stars
1	#2LGYPUPV0	Poland Esports	8000053	2704433	2	30	Brawl Stars
2	#2VRPY2CYV	North London	8000007	2641519	3	30	Brawl Stars
3	#2UU8VPPR9	Old Friends	8000022	2638262	4	30	Brawl Stars
4	#2UG0JP0RP	ILY ESPORTS	8000029	2636948	5	30	Brawl Stars
...	...	...	...	...	...	...	...
195	#2CVQ8JJ92	Goats	8000016	2334478	196	30	Brawl Stars
196	#2YP8R2GLP	Regions Team	8000049	2333844	197	30	Brawl Stars
197	#2GGLC22C0	NO RISKnoFun PL	8000029	2333064	198	30	Brawl Stars
198	#28U982	RealDutch	8000029	2332954	199	28	Brawl Stars
199	#2J090GVUY	Wojownicy Mocy	8000049	2332221	200	30	Brawl Stars

200 rows × 7 columns

Next steps: Generate code with `df_top_bs` View recommended plots New interactive sheet

Not as much information here. But still maybe member count and rank is correlated somehow.

```
url_test = "https://api.clashofclans.com/v1/locations"

# Function to find the "International" location
def find_international_location():
    params_test = {
        "limit": 10,
    }

    # Make the request to the Clash of Clans API
    response_test = requests.get(url_test, headers=headers_coc, params=params_test)
    data_test = response_test.json().get("items", [])

    # Filter the data to find the location that says "International"
    for location in data_test:
      if "International" in location.get("name", ""):
        return location

# Find the "International" location
international_location = find_international_location()
print("Found International Location:", international_location)
```

```
Found International Location: {'id': 32000006, 'name': 'International', 'isCountry': False}
```

Had to find the international location ID for the URL below to find international clan rankings.

```
url_coc = "https://api.clashofclans.com/v1/locations/32000006/rankings/clans"

# Function to get the top international 200 clans for Clash of Clans
def get_top_200_rank_coc():
    params_coc = {
        "limit": 200,  # Request only the top 200 international clans
    }

    # Make the request for Clash Of Clans
    response_coc = requests.get(url_coc, headers=headers_coc, params=params_coc)

    data_coc = response_coc.json().get("items", [])
    return data_coc

# Fetch top 200 Clash of Clans Clans
top_200_coc = get_top_200_rank_coc()

df_top_coc = pd.json_normalize(top_200_coc)
df_top_coc["game"] = "Clash of Clans"
df_top_coc
```

	tag	name	clanLevel	members	clanPoints	rank	previousRank	location.id	location.name	location.isCountry	badgeUrls.small	badgeUrls.large	badgeUrls.medium	game
0	#22R8990Q8	Kings Nation™	24	50	56748	1	1	32000006	International	False	https://api-assets.clashofclans.com/badges/70/...	https://api-assets.clashofclans.com/badges/512...	https://api-assets.clashofclans.com/badges/200...	Clash of Clans
1	#PQ2VL90Y	(OuTlaWs'Team:)	21	47	56411	2	2	32000006	International	False	https://api-assets.clashofclans.com/badges/70/...	https://api-assets.clashofclans.com/badges/512...	https://api-assets.clashofclans.com/badges/200...	Clash of Clans
2	#PC9QYGCU	BAX flower city	20	50	56382	3	5	32000006	International	False	https://api-assets.clashofclans.com/badges/70/...	https://api-assets.clashofclans.com/badges/512...	https://api-assets.clashofclans.com/badges/200...	Clash of Clans
3	#C0YLCCJJ	Royal Family™	24	48	56296	4	3	32000006	International	False	https://api-assets.clashofclans.com/badges/70/...	https://api-assets.clashofclans.com/badges/512...	https://api-assets.clashofclans.com/badges/200...	Clash of Clans
4	#2PQVR9YGQ	كلان العرب	27	50	56233	5	-1	32000006	International	False	https://api-assets.clashofclans.com/badges/70/...	https://api-assets.clashofclans.com/badges/512...	https://api-assets.clashofclans.com/badges/200...	Clash of Clans
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
195	#2LYYPVRR	These Boots	27	47	50514	196	172	32000006	International	False	https://api-assets.clashofclans.com/badges/70/...	https://api-assets.clashofclans.com/badges/512...	https://api-assets.clashofclans.com/badges/200...	Clash of Clans
196	#299QPGUUQ	霳领潮流•太陽神	24	43	50508	197	139	32000006	International	False	https://api-assets.clashofclans.com/badges/70/...	https://api-assets.clashofclans.com/badges/512...	https://api-assets.clashofclans.com/badges/200...	Clash of Clans
197	#GCCJVQJV	INDO REBELLION	32	45	50499	198	205	32000006	International	False	https://api-assets.clashofclans.com/badges/70/...	https://api-assets.clashofclans.com/badges/512...	https://api-assets.clashofclans.com/badges/200...	Clash of Clans
198	#P92JGYGJ	NEW ERA	30	48	50487	199	215	32000006	International	False	https://api-assets.clashofclans.com/badges/70/...	https://api-assets.clashofclans.com/badges/512...	https://api-assets.clashofclans.com/badges/200...	Clash of Clans
199	#2YUR2QYV0	Active Brothers	22	47	50483	200	286	32000006	International	False	https://api-assets.clashofclans.com/badges/70/...	https://api-assets.clashofclans.com/badges/512...	https://api-assets.clashofclans.com/badges/200...	Clash of Clans

200 rows × 14 columns

Next steps: Generate code with `df_top_coc` View recommended plots New interactive sheet

```
url_test_cr = "https://api.clashroyale.com/v1/locations"

# Function to find the "International" location
def find_international_location():
    params_test = {
        "limit": 100,
    }

    # Make the request to the Clash of Clans API
    response_test = requests.get(url_test_cr, headers=headers_cr, params=params_test)
    data_test = response_test.json().get("items", [])

    # Filter the data to find the location that says "International"
    for location in data_test:
      if "International" in location.get("name", ""):
        return location

# Find the "International" location
international_location = find_international_location()
print("Found International Location:", international_location)
```

```
Found International Location: {'id': 57000006, 'name': 'International', 'isCountry': False}
```

Also had to find the international location ID for the URL below to find international clan rankings.

```
url_cr = "https://api.clashroyale.com/v1/locations/57000006/rankings/clans"

# Function to get the top international 200 clans for Clash Royale
def get_top_200_rank_cr():
    params_cr = {
        "limit": 200,  # Request only the top 200 international clans
    }

    # Make the request for Clash Royale
    response_cr = requests.get(url_cr, headers=headers_cr, params=params_cr)

    data_cr = response_cr.json().get("items", [])
    return data_cr

# Fetch top 200 Clash Royale Clans
```

```
top_200_cr = get_top_200_rank_cr()

df_top_cr = pd.json_normalize(top_200_cr)
df_top_cr["game"] = "Clash Royale"
df_top_cr
```

	tag	name	rank	previousRank	clanScore	members	badgeId	location.id	location.name	location.isCountry	game
0	#QGPC0Q0Q	WARSHIP ACADEMY	1	2	90000	50	16000024	57000006	International	False	Clash Royale
1	#QY0CCGJJ	50 Green Jits	2	3	90000	50	16000154	57000006	International	False	Clash Royale
2	#GYCPQJL8	TikTok Live	3	4	89971	50	16000125	57000006	International	False	Clash Royale
3	#Y0JVRULJ	Masters mx	4	10	89947	50	16000022	57000006	International	False	Clash Royale
4	#G0CVYJQU	bankdup	5	6	89866	50	16000147	57000006	International	False	Clash Royale
...	...	...	...	...	...	...	...	...	...	...	...
195	#P908GQ0R	● Helskrim ○	196	176	86347	48	16000005	57000006	International	False	Clash Royale
196	#Q8JQ2U0G	No War	197	201	86344	48	16000107	57000006	International	False	Clash Royale
197	#2J9RGLJ	ButterMyLobster	198	158	86339	46	16000168	57000006	International	False	Clash Royale
198	#8PR82C2P	Hit $quad 2	199	202	86335	50	16000028	57000006	International	False	Clash Royale
199	#GJY8RPCQ	the legend	200	231	86332	50	16000128	57000006	International	False	Clash Royale

200 rows × 11 columns

Next steps: Generate code with `df_top_cr` View recommended plots New interactive sheet

```
df_combined = pd.concat([df_top_bs, df_top_coc, df_top_cr], ignore_index=True)
df_combined["memberCount"] = df_combined["memberCount"].fillna(df_combined["members"])
df_combined = df_combined.drop(columns=["badgeId", "badgeUrls.small", "badgeUrls.large", "badgeUrls.medium", "members", "clanLevel", "clanPoints", "clanScore", "previousRank", "location.id", "location.name", "location.isCountry", "trophies"], inplace=False)
df_combined
```

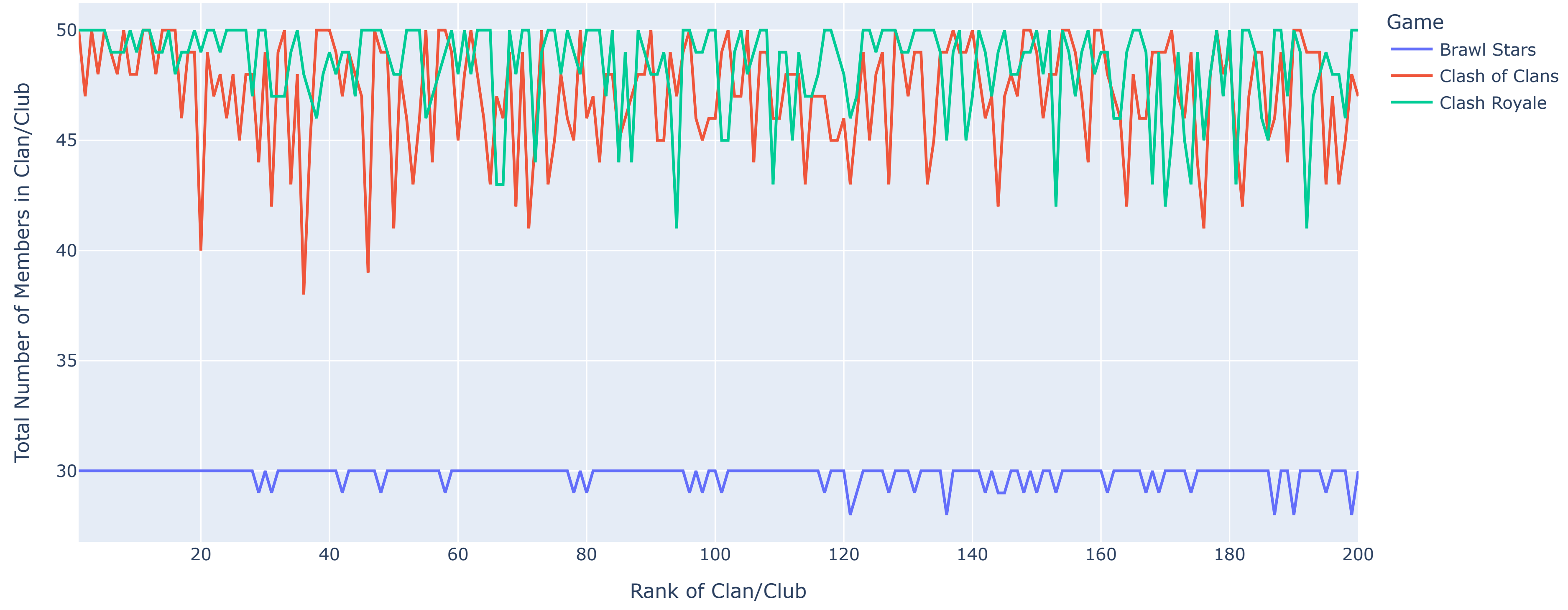
	tag	name	rank	memberCount	game
0	#2CJ8YYJV9	Thunder ⚡	1	30.0	Brawl Stars
1	#2LGYPUPV0	Poland Esports	2	30.0	Brawl Stars
2	#2VRPY2CYV	North London	3	30.0	Brawl Stars
3	#2UU8VPPR9	Old Friends	4	30.0	Brawl Stars
4	#2UG0JP0RP	ILY ESPORTS	5	30.0	Brawl Stars
...	...	...	...	...	...
595	#P908GQ0R	● Helskrim ○	196	48.0	Clash Royale
596	#Q8JQ2U0G	No War	197	48.0	Clash Royale
597	#2J9RGLJ	ButterMyLobster	198	46.0	Clash Royale
598	#8PR82C2P	Hit $quad 2	199	50.0	Clash Royale
599	#GJY8RPCQ	the legend	200	50.0	Clash Royale

600 rows × 5 columns

Next steps: Generate code with `df_combined` View recommended plots New interactive sheet

```
fig = px.line(df_combined,
             x = "rank",
             y = "memberCount",
             color = "game",
             labels=dict(rank="Rank of Clan/Club",
                        memberCount="Total Number of Members in Clan/Club",
                        game = "Game"))

fig.show()
```



```
df_combined.groupby("game")["memberCount"].mean()
```

	memberCount
game	
Brawl Stars	29.825
Clash Royale	48.395
Clash of Clans	47.190

dtype: float64

I know the second analysis wasn't as exciting as the first analysis. But what can be seen here is that all top ranking clans/clubs have a max number of members. What's more interesting is that in Clash of Clans, some clans that are high ranking have a varied number of members, dipping as low as 38 members. This may not seem that little, but in these games, having a clan member is very valuable. So missing even 1 member can change ranking drastically. The averages suggest that this difference isn't that much different. But I still expect the member count for Clash of Clans to be lower than Clash Royale because of the importance of clans in Clash of Clans.

Clarifying Questions for the end

Why did I do this and not other data?

I originally wanted to do music data. But I didn't find that too interesting and the API I chose to do it in was complicated and not as clear to me to use as the SUPERCELL data. So I thought about what API and data I could use, could get the most amount of data from the most amount of sources. I came across the SUPERCELL data in the JSON API github repo, and I thought it was super interesting to look at individuals statistics. I'm dissappointed that there could be no merging. That's one weakness of my project. But I thought about more the real world implications of this data. In reality, it's hard to find data that matches perfectly where we could use the pandas merge function. The data sets created for the purpose of that merge function and I think finding wild random data that matches is impossible without tons of data manipulating to be all on the same scale. I know it wouldn't be too difficult from what I already completed here. But still I want to show that I am capable of doing what's asked.

What are the sources of your data sets? What format is each data set in (CSV, JSON, shapefile, etc) For the API, be sure to specify the API (provide a link) and document what endpoints/parameters you used to make the request.

I used three APIs from SUPERCELL. All public and easily accessible.

BrawlStars: https://developer.brawlstars.com/#/

Clash Royale: https://developer.clashroyale.com/#/

Clash of Clans: https://developer.clashofclans.com/#/

Above I used specifically the rankings and clan parts of the JSON. Again the website had easy access and was readable.

For each source, investigate how the data was collected For each source, what are the observational units and how were they selected for the data set? For each source, what are the variables, and how they were measured? For each source, are there any missing observations or values? What role might missing values play in this context?

Each source was collected directly from SUPERCELL (again the company who created and maintains the game). It's constantly update with new player information, what clubs/clans they join, what in game events occurs, rankings, tournaments, etc. The variables I choose for each observation was mainly region, clan, and member counts. These are interesting numbers to me because when I play the game, some clans are raelly strict on joining and some are not as strict. So analyzing these sources would be really interesting to me into exactly what are the effects of being stricter with clan/club entry and what factors and aspects of the game I may have overlooked and not have seen the first time.

Document your process of cleaning/processing the data from each source before joining For the data from each source, which observations did you choose to include, and why? For the data from each source, which variables did you choose to include, and why? For each source, did you transform or create any new variables - if so, how and why? Did you reshape the data (e.g., melt or stack) - if so, how and why? Did you summarize the data before joining (e.g., groupby) - if so, how and why? If JSON data, describe the hierarchy and what data you used from different levels in the hierarchy. If HTML scraping, describe the process for extracting the relevant information from the pages

Up above. I'll answer some ones I didn't during the process, but I didn't have to clean as much before the joining. I think the hardest aspect of joining was the set up from the API. Finding which group I should narrow it down to and where and how to retreieve the data from the very top was the process. I didn't reshape that much but it was just summarising. And no I did not summarize before joining. When I hear summarizing I think we lose the whole picture and get a surface level model or simplification of something. So I wanted to keep as much original data when joining.

The data was in a JSON format and the format for all three sources were all something like individual players, clans/clubs, locations and rankings, events, brawlers, locations, tournaments, and events. I choose to target clans/clubs, locations and rankings. Those could be more generalized over the three/two and I found those the most interesting. Mainly I wanted to look at an overall average of clubs, maybe asking if being in a club is being that much beneficial. From this analysis it appears not that significant. But it is something.

What did you use as the key? Did you need to process the key in order to join the data sets? Were there any observations in any of the data sets that you were not able to merge? Which ones and why? What effect will this have?

I did not have a key for this data set. I concatened instead. I think this is a cheaper way out. But I knew the games were uch different and not all users play the same game under the same ID. So merging was not an option.

A lot of observations in the dataset could not be merged. I eliminated around 75% of them. Whatever could be merged was more user and clan related data like location, rankings, types of clans, number of members, something general like that. The effect I saw is that my results are super general and don't tell much. They do answer my sort-of question of are clans picky about members and who in this world is the main demographic of playing Clash of Clans and Clash Royale.

Document your process of creating your final data frame

Which observations did you select and why?

I capped the observations at 500 at first, with a minimum of 10 members in each clan. To attempt to diferentiate active and inactive clans and to not overwhelm the analysis.

Which variables did you select and why?

I selected above tag, name, type, requiredTrophies, location id, location name, location isCountry, and country code. Clash of Clans and Clash Royale share these variables and answered my demographic question.

Did you transform or create any new variables - if so, how and why?

I only created a "game" variable to check which games each row cam from.

Did you reshape the data (e.g., melt or stack) - if so, how and why?

I only unstacked one group by to summarize average member counts.

Did you summarize the data (e.g., groupby) - if so, how and why?

I summarize average member counts. Was interested if the best clans/clubs had the highest counts and see which game had more or less members.

I learned from this project that most successful clans/clubs are full or are at near capacity. That most Clash Royale players randomly selected are international and did not disclose their locations, while Clash of Clans players are more likely to be from South East Asia. From the Brawl Stars data and the other two I found that clans/clubs that are in the top ranks also tend to be vey active, have near capacity, and flucatute in members by a few.

Citations: OpenAI. ChatGPT. 2024. https://chat.openai.com

Google DeepMind. Gemini. 2024.

Todd Motto. Public APIs. GitHub, n.d. https://github.com/toddmotto/public-apis.

Supercell. Brawl Stars API Documentation. n.d. https://developer.brawlstars.com/#/.