```
import pandas as pd

file_name = 'coc_clans_dataset.csv'

# Loading the dataset into a pandas DataFrame
df = pd.read_csv(file_name)

print("Dataset preview:")
print(df.head())

# print("\nDataset info:")
# print(df.info())

## Handling missing data

# Excluding 'clan_location' from missing data analysis
columns_to_check = df.columns.difference(['clan_location'])

# Dropping rows with missing data in the specified columns
cleaned_df = df.dropna(subset=columns_to_check)

# Printing the updated dataset information
print(f"Original number of rows: {df.shape[0]}")
print(f"Number of rows after removing missing data:
{cleaned_df.shape[0]}")

from pymongo import MongoClient

# Connecting to MongoDB
client = MongoClient("mongodb://localhost:27017/")
db = client['clash_of_clans_db']
collection = db['clans']

# Dropping the collection (removing all data) (the data was being
multiplied each time, so dropping the collection had to be included)
collection.drop()

print("Collection dropped. Now inserting the cleaned data.")

# Converting the cleaned DataFrame to dictionary format
data_dict = cleaned_df.to_dict(orient='records')

# Inserting the data into MongoDB
collection.insert_many(data_dict)

print("Clash of Clans dataset successfully uploaded to MongoDB!")

## MongoDB Schema Design
## Just as a design, not a coding part

{
```

```
    "clan_name": "string",                          // Name of the clan
    "clan_type": "string",                          // Type of clan (e.g.,
open, closed)
    "clan_location": "string",                      // Clan's location
(e.g., International)
    "isFamilyFriendly": "boolean",                  // Whether the clan is
family-friendly
    "clan_level": "integer",                        // Level of the clan
    "clan_points": "integer",                       // Total clan points
    "clan_builder_base_points": "integer",          // Points from the
builder base
    "required_trophies": "integer",                 // Minimum trophies
required to join
    "war_frequency": "string",                      // Frequency of wars
(e.g., once a week)
    "war_stats": {                                  // Embedded document
for war statistics
        "war_win_streak": "integer",                // Current win streak
in wars
        "war_wins": "integer",                      // Total number of
wars won
        "war_ties": "integer",                      // Total number of
ties in wars
        "war_losses": "integer",                    // Total number of
wars lost
        "clan_war_league": "string"                 // Current war league
(e.g., Unranked)
    },
    "num_members": "integer",                       // Number of members
in the clan
    "requirements": {                               // Embedded document
for joining requirements
        "required_builder_base_trophies": "integer",  // Trophies
needed for builder base
        "required_townhall_level": "integer"          // Minimum town
hall level
    },
    "capital": {                                    // Embedded document
for clan capital
        "clan_capital_hall_level": "integer",       // Clan capital hall
level
        "clan_capital_points": "integer",           // Points in the clan
capital
        "capital_league": "string"                  // Capital league tier
(e.g., Unranked)
    },
    "mean_stats": {                                 // Embedded document
for member averages
        "mean_member_level": "integer",             // Average level of
```

```
members
        "mean_member_trophies": "integer"       // Average trophies of
members
    }
}

## Queries on the dataset

total_records = collection.count_documents({})
print("The total records are", total_records)

# Query 1: Clan Type Percentages

# Pipeline for Clan Type Percentages
pipeline_clan_type_percentage = [
    {'$group': {'_id': '$clan_type', 'count': {'$sum': 1}}},
    {'$project': {
        '_id': 1,
        'count': 1,
        'percentage': {'$multiply': [{'$divide': ['$count',
total_records]}, 100]}
    }},
    {'$sort': {'percentage': -1}}
]

# Results for Clan Type Percentages
results_clan_type_percentage =
collection.aggregate(pipeline_clan_type_percentage)

print("Results for Query 1: Clan Type Percentages\n")

for result in results_clan_type_percentage:
    print(f"Clan Type: {result['_id']}, Count: {result['count']},
Percentage: {result['percentage']:.1f}%")

# Query 2: Clan Location Percentages (Top 15 Countries)

# Pipeline for Clan Location Percentages (Top 15 Countries)
pipeline_clan_location_percentage = [
    {'$group': {'_id': '$clan_location', 'count': {'$sum': 1}}},
    {'$match': {'_id': {'$ne': None}}},
    {'$project': {
        '_id': 1,
        'count': 1,
        'percentage': {'$multiply': [{'$divide': ['$count',
total_records]}, 100]}
    }},
    {'$sort': {'percentage': -1}},
    {'$limit': 15}
]
```

```
# Results for Clan Location Percentages (Top 15 Countries)
results_clan_location_percentage =
collection.aggregate(pipeline_clan_location_percentage)

print("Results for Query 2: Clan Location Percentages (Top 15
Countries)\n")

for result in results_clan_location_percentage:
    print(f"Clan Location: {result['_id']}, Count: {result['count']},
Percentage: {result['percentage']:.1f}%")

# Query 3: Family Friendly Percentages

# Pipeline for Family Friendly Percentages
pipeline_family_friendly_percentage = [
    {'$group': {'_id': '$isFamilyFriendly', 'count': {'$sum': 1}}},
    {'$project': {
        '_id': 1,
        'count': 1,
        'percentage': {'$multiply': [{'$divide': ['$count',
total_records]}, 100]}
    }}
]

# Results for Family Friendly Percentages
results_family_friendly_percentage =
collection.aggregate(pipeline_family_friendly_percentage)

print("Results for Query 3: Family Friendly Percentages\n")

for result in results_family_friendly_percentage:
    print(f"Family Friendly: {result['_id']}, Count:
{result['count']}, Percentage: {result['percentage']:.1f}%")

# Query 4: Clan Level Counts (Ascending) and Top Clan

# Pipeline for Clan Level Counts (Ascending)
pipeline_clan_level_counts = [
    {'$group': {'_id': '$clan_level', 'count': {'$sum': 1}}},
    {'$sort': {'_id': 1}}
]

# Results for Clan Level Counts (Ascending)
results_clan_level_counts =
collection.aggregate(pipeline_clan_level_counts)

print("Results for Query 4: Clan Level Counts (Ascending)\n")

for result in results_clan_level_counts:
    print(f"Clan Level: {result['_id']}, Count: {result['count']}")
```

```
# Pipeline for Top Clan with Highest Level
pipeline_top_clan = [
    {'$sort': {'clan_level': -1}},
    {'$limit': 1}
]

# Results for Top Clan with Highest Level
result_top_clan = collection.aggregate(pipeline_top_clan)

print("\nResults for Query 4: Top Clan with Highest Level\n")

for result in result_top_clan:
    print(f"Top Clan with Highest Level: {result['clan_name']}, Level:
{result['clan_level']}")

# Query 5: Clan Points (Range and Top 5 Clans)

range_size = 3000

# Pipeline for Clan Points Range
pipeline_clan_points_range = [
    {'$match': {'clan_points': {'$ne': None}}},
    {'$group': {
        '_id': '$clan_name',
        'clan_points': {'$first': '$clan_points'}
    }},
    {'$project': {'clan_points_range': {
        '$floor': {'$divide': ['$clan_points', range_size]}}}},
    {'$group': {'_id': '$clan_points_range', 'count': {'$sum': 1}}},
    {'$sort': {'_id': 1}}
]

# Results for Clan Points Range
results_clan_points_range =
list(collection.aggregate(pipeline_clan_points_range))

print("Results for Query 5: Clan Points Range\n")

for result in results_clan_points_range:
    lower_range = result['_id'] * range_size
    upper_range = (result['_id'] + 1) * range_size
    print(f"Clan Points Range: {lower_range} - {upper_range}, Count:
{result['count']}")

top_5_clans = list(collection.aggregate([
    {'$match': {'clan_points': {'$ne': None}}},
    {'$group': {'_id': '$clan_name', 'clan_points': {'$first':
'$clan_points'}}},
    {'$sort': {'clan_points': -1}},
    {'$limit': 5}
```

```
]))

print("\nResults for Query 5: Top 5 Clans by Clan Points\n")

for clan in top_5_clans:
    print(f"Top Clan: {clan['_id']}, Points: {clan['clan_points']}")

# Query 6: Clan Builder Points (Range) and Top 5 Clans

range_size_builder = 3000

# Pipeline for Clan Builder Points Range
pipeline_clan_builder_points_range = [
    {'$match': {'clan_builder_base_points': {'$ne': None}}},
    {'$group': {
        '_id': {
            'clan_name': '$clan_name',
            'builder_points_range': {
                '$floor': {'$divide': ['$clan_builder_base_points',
range_size_builder]}
            }
        },
        'count': {'$sum': 1}
    }},
    {'$group': {
        '_id': '$_id.builder_points_range',
        'count': {'$sum': 1}
    }},
    {'$sort': {'_id': 1}}
]

# Results for Clan Builder Points Range
results_clan_builder_points_range =
collection.aggregate(pipeline_clan_builder_points_range)

print("Results for Query 6: Clan Builder Points Range\n")

for result in results_clan_builder_points_range:
    lower_range = result['_id'] * range_size_builder
    upper_range = (result['_id'] + 1) * range_size_builder
    print(f"Builder Points Range: {lower_range} - {upper_range},
Count: {result['count']}")

top_5_clans_builder_points = list(collection.aggregate([
    {'$match': {'clan_builder_base_points': {'$ne': None}}},
    {'$group': {
        '_id': '$clan_name',
        'clan_builder_base_points': {'$max':
'$clan_builder_base_points'}
    }},
```

```
    {'$sort': {'clan_builder_base_points': -1}},
    {'$limit': 5}
]))

print("\nResults for Query 6: Top 5 Clans by Builder Points\n")

for clan in top_5_clans_builder_points:
    print(f"Top Clan: {clan['_id']}, Builder Points:
{clan['clan_builder_base_points']}")

# Query 7: Required Trophies (Counts and Percentages)

# Pipeline for Required Trophies Counts and Percentages
pipeline_required_trophies = [
    {'$match': {'$expr': {'$eq': [{'$mod': ["$required_trophies",
100]}, 0]}}},
    {'$group': {'_id': '$required_trophies', 'count': {'$sum': 1}}},
    {'$match': {'_id': {'$lte': 5500}}},
    {'$project': {
        '_id': 1,
        'count': 1,
        'percentage': {'$multiply': [{'$divide': ['$count',
total_records]}, 100]}
    }},
    {'$sort': {'_id': 1}}
]

# Results for Required Trophies Counts and Percentages
results_required_trophies =
collection.aggregate(pipeline_required_trophies)

print("Results for Query 7: Required Trophies (Counts and
Percentages)\n")

for result in results_required_trophies:
    print(f"Required Trophies: {result['_id']}, Count:
{result['count']}, Percentage: {result['percentage']:.1f}%")

# Query 8: War Frequency (Counts and Percentages)

# Pipeline for War Frequency Counts and Percentages with Descending
Order
pipeline_war_frequency = [
    {'$group': {'_id': '$war_frequency', 'count': {'$sum': 1}}},
    {'$project': {
        '_id': 1,
        'count': 1,
        'percentage': {'$multiply': [{'$divide': ['$count',
total_records]}, 100]}
    }},
```

```
    {'$sort': {'count': -1}}
]

# Results for War Frequency Counts and Percentages
results_war_frequency = collection.aggregate(pipeline_war_frequency)

print("Results for Query 8: War Frequency (Counts and Percentages)\n")

for result in results_war_frequency:
    print(f"War Frequency: {result['_id']}, Count: {result['count']},
Percentage: {result['percentage']:.1f}%")

# Query 9: War Win Streak (Counts and Top 5 Clans)

# Pipeline for War Win Streak Counts
pipeline_war_win_streak = [
    {'$group': {'_id': '$war_win_streak', 'count': {'$sum': 1}}},
    {'$sort': {'_id': -1}}
]

# Results for War Win Streak Counts
results_war_win_streak = collection.aggregate(pipeline_war_win_streak)

print("Results for Query 9: War Win Streak (Counts and Top 5 Clans)\
n")

for result in results_war_win_streak:
    print(f"War Win Streak: {result['_id']}, Count:
{result['count']}")

top_5_clans_by_streak = collection.find().sort('war_win_streak', -
1).limit(5)

print("\nTop 5 Clans by War Win Streak:\n")
for clan in top_5_clans_by_streak:
    print(f"Top Clan: {clan['clan_name']}, War Win Streak:
{clan['war_win_streak']}")

# Query 10: War Wins (Range and Top 5 Clans)

range_size_wins = 100
num_bins = 14

# Pipeline for War Wins Range
pipeline_war_wins_range = [
    {'$group': {'_id': '$clan_name', 'war_wins': {'$first':
'$war_wins'}}},
    {'$project': {'war_wins_range': {'$floor': {'$divide':
['$war_wins', range_size_wins]}}}},
    {'$group': {'_id': '$war_wins_range', 'count': {'$sum': 1}}},
    {'$sort': {'_id': 1}}
```

```
]

# Results for War Wins Range
print("Results for Query 10: War Wins (Range and Top 5 Clans)\n")
results_war_wins_range = collection.aggregate(pipeline_war_wins_range)
for result in results_war_wins_range:
    lower_range = result['_id'] * range_size_wins
    upper_range = (result['_id'] + 1) * range_size_wins
    print(f"War Wins Range: {lower_range} - {upper_range}, Count:
{result['count']}")

# Pipeline for Top 5 Clans by War Wins
pipeline_top_5_war_wins = [
    {'$sort': {'war_wins': -1}},
    {'$group': {'_id': '$clan_name', 'war_wins': {'$first':
'$war_wins'}}},
    {'$sort': {'war_wins': -1}},
    {'$limit': 5},
    {'$project': {'clan_name': '$_id', 'war_wins': 1, '_id': 0}}
]

# Results for Top 5 Clans by War Wins
results_top_5_war_wins = collection.aggregate(pipeline_top_5_war_wins)
print("\nTop 5 Clans by War Wins:")
for result in results_top_5_war_wins:
    print(f"Top Clan: {result['clan_name']}, War Wins:
{result['war_wins']}")

# Query 11: War Ties (Counts)

# Pipeline for War Ties Counts
pipeline_war_ties_counts = [
    {'$group': {'_id': '$war_ties', 'count': {'$sum': 1}}},
    {'$sort': {'_id': 1}}
]

# Results for War Ties Counts
print("Results for Query 11: War Ties (Counts)\n")

results_war_ties_counts =
collection.aggregate(pipeline_war_ties_counts)
for result in results_war_ties_counts:
    print(f"War Ties: {result['_id']}, Count: {result['count']}")

# Query 12: War Losses (Counts)

# Pipeline for War Losses Counts
pipeline_war_losses_counts = [
    {'$group': {'_id': '$war_losses', 'count': {'$sum': 1}}},
    {'$sort': {'_id': 1}}
```

```
]

# Results for War Losses Counts
print("Results for Query 12: War Losses (Counts)\n")

results_war_losses_counts =
collection.aggregate(pipeline_war_losses_counts)
for result in results_war_losses_counts:
    print(f"War Losses: {result['_id']}, Count: {result['count']}")

# Query 13: Win Percentage Range (10% Bins) and Count of Clans with
Specific Conditions

# Pipeline: Filter clans with at least 10 total games (wins + losses)
pipeline_win_percentage_bins = [
    {'$match': {
        '$and': [
            {'war_wins': {'$gte': 0}},
            {'war_losses': {'$gte': 0}},
            {'$expr': {'$gte': [{'$add': ['$war_wins',
'$war_losses']}, 10]}}
        ]
    }},

    # Pipeline: Calculating win percentage
    {'$project': {
        'clan_name': 1,
        'win_percentage': {
            '$cond': {
                'if': {'$gt': [{'$add': ['$war_wins', '$war_losses']},
0]},
                'then': {
                    '$multiply': [
                        {'$divide': ['$war_wins', {'$add':
['$war_wins', '$war_losses']}]}, 100
                    ]
                },
                'else': 0  # For 0 wins and 0 losses, the percentage
was set to 0
            }
        }
    }},

    # Pipeline: Creating bins for each 10% range of win percentage
    {'$project': {
        'clan_name': 1,
        'win_percentage': 1,
        'win_percentage_bin': {
            '$cond': {
                'if': {'$gte': ['$win_percentage', 100]},
```

```
                'then': 9,
                'else': {'$floor': {'$divide': ['$win_percentage',
10]}}
            }
        }
    }},

    # Pipeline: Grouping by bins and counting the number of clans in
each bin
    {'$group': {
        '_id': '$win_percentage_bin',
        'count': {'$sum': 1}
    }},

    # Pipeline: Sorting by bins
    {'$sort': {'_id': 1}}
]

# Results: Output bins and their counts
print("Results for Query 13: Win Percentage Range (10% Bins) and Count
of Clans with Specific Conditions\n")

results_win_percentage_bins =
collection.aggregate(pipeline_win_percentage_bins)
for result in results_win_percentage_bins:
    lower_range = result['_id'] * 10
    upper_range = (result['_id'] + 1) * 10 if result['_id'] < 9 else
100
    print(f"Win Percentage Range: {lower_range}% - {upper_range}%,
Count: {result['count']}")

# Query 14: Clan War League (Counts and Percentages) with Custom Order
(Excluding Unranked)

# Custom order for Clan War League rankings
war_league_order = [
    "Unranked", "Bronze League III", "Bronze League II", "Bronze
League I",
    "Silver League III", "Silver League II", "Silver League I",
    "Gold League III", "Gold League II", "Gold League I",
    "Crystal League III", "Crystal League II", "Crystal League I",
    "Master League III", "Master League II", "Master League I",
    "Champion League III", "Champion League II", "Champion League I"
]

# Pipeline for Clan War League Counts and Percentages with Custom
Order (Excluding Unranked)
pipeline_war_league_counts_exclude_unranked = [
    {'$match': {'clan_war_league': {'$ne': 'Unranked'}}},
    {'$group': {'_id': '$clan_war_league', 'count': {'$sum': 1}}},
```

```
    {'$addFields': {
        'league_rank': {'$indexOfArray': [war_league_order, '$_id']}
    }},
    {'$sort': {'league_rank': 1}},
    {'$project': {
        '_id': 1,
        'count': 1,
        'percentage': {'$multiply': [{'$divide': ['$count',
total_records]}, 100]}
    }}
]

print("Results for Query 14: Clan War League (Counts and Percentages)
with Custom Order (Excluding Unranked)\n")

results_war_league_counts_exclude_unranked =
collection.aggregate(pipeline_war_league_counts_exclude_unranked)
for result in results_war_league_counts_exclude_unranked:
    print(f"Clan War League: {result['_id']}, Count:
{result['count']}, Percentage: {result['percentage']:.1f}%")

# Query 15: Number of Members (Counts and Percentages)

# Pipeline for Number of Members Counts and Percentages
pipeline_num_members_counts = [
    {'$group': {'_id': '$num_members', 'count': {'$sum': 1}}},
    {'$project': {
        '_id': 1,
        'count': 1,
        'percentage': {'$multiply': [{'$divide': ['$count',
total_records]}, 100]}
    }},
    {'$sort': {'_id': 1}}
]

print("Results for Query 15: Number of Members (Counts and
Percentages)\n")

results_num_members_counts =
collection.aggregate(pipeline_num_members_counts)
for result in results_num_members_counts:
    print(f"Number of Members: {result['_id']}, Count:
{result['count']}, Percentage: {result['percentage']:.1f}%")

# Query 16: Requirements (Counts and Percentages)

# Pipeline for Requirements (Required Builder Base Trophies)
pipeline_requirements_builder = [
    {'$group': {'_id': '$required_builder_base_trophies', 'count':
{'$sum': 1}}},
```

```
    {'$match': {'_id': {'$mod': [100, 0]}}},
    {'$project': {
        '_id': 1,
        'count': 1,
        'percentage': {'$multiply': [{'$divide': ['$count',
total_records]}, 100]}
    }},
    {'$sort': {'_id': 1}}
]

print("Results for Query 16: Requirements (Ordered by Required Builder
Base Trophies)\n")

results_requirements_builder =
collection.aggregate(pipeline_requirements_builder)
for result in results_requirements_builder:
    print(f"Requirements: {result['_id']}, Count: {result['count']},
Percentage: {result['percentage']:.1f}%")

# Pipeline for Required Townhall Level
pipeline_requirements_townhall = [
    {'$group': {'_id': '$required_townhall_level', 'count': {'$sum':
1}}},
    {'$project': {
        '_id': 1,
        'count': 1,
        'percentage': {'$multiply': [{'$divide': ['$count',
total_records]}, 100]}
    }},
    {'$sort': {'_id': 1}}
]

print("\nResults for Query 16: Required Townhall Level (Ordered by
Townhall Level)\n")

results_requirements_townhall =
collection.aggregate(pipeline_requirements_townhall)
for result in results_requirements_townhall:
    print(f"Townhall Level: {result['_id']}, Count: {result['count']},
Percentage: {result['percentage']:.1f}%")

# Query 17: Clan Capital Hall Level (Counts, Percentages)

# Pipeline for Clan Capital Hall Level (Counts and Percentages)
pipeline_capital_hall = [
    {'$group': {'_id': '$clan_capital_hall_level', 'count': {'$sum':
1}}},
    {'$project': {
        '_id': 1,
        'count': 1,
```

```
        'percentage': {'$multiply': [{'$divide': ['$count',
total_records]}, 100]}
    }},
    {'$sort': {'_id': 1}}
]

print("Results for Query 17: Clan Capital Hall Level (Counts and
Percentages)\n")

results_capital_hall = collection.aggregate(pipeline_capital_hall)
for result in results_capital_hall:
    print(f"Capital Hall Level: {result['_id']}, Count:
{result['count']}, Percentage: {result['percentage']:.1f}%")

# Query 18: Clan Capital Points (Range and Top 5 Clans)

range_size_capital = 300
num_bins = 21

# Pipeline to group clan capital points into bins and count
pipeline_capital_points = [
    {'$project': {'capital_points_bin': {
        '$floor': {'$divide': ['$clan_capital_points',
range_size_capital]}}
    }},
    {'$group': {'_id': '$capital_points_bin', 'count': {'$sum': 1}}},
    {'$sort': {'_id': 1}}
]

print("Results for Query 18: Clan Capital Points (Range and Top 5
Clans)\n")

results_capital_points = collection.aggregate(pipeline_capital_points)

for result in results_capital_points:
    bin_start = result['_id'] * range_size_capital
    bin_end = bin_start + range_size_capital
    print(f"Capital Points Range: {bin_start} - {bin_end}, Count:
{result['count']}")

top_5_clans_capital = collection.find().sort('clan_capital_points', -
1).limit(5)
print("\nTop 5 Clans by Capital Points:")
for clan in top_5_clans_capital:
    print(f"Top Clan: {clan['clan_name']}, Capital Points:
{clan['clan_capital_points']}")

# Query 19: Clan Capital League (Counts and Percentages)

# Defining the correct league order excluding "Unranked"
league_order = [
```

```
    "Bronze League III", "Bronze League II", "Bronze League I",
    "Silver League III", "Silver League II", "Silver League I",
    "Gold League III", "Gold League II", "Gold League I",
    "Crystal League III", "Crystal League II", "Crystal League I",
    "Master League III", "Master League II", "Master League I",
    "Champion League III", "Champion League II", "Champion League I",
    "Titan League III", "Titan League II", "Titan League I",
    "Legend League"
]

# Aggregating to get counts and percentages
pipeline_capital_league = [
    {'$group': {'_id': '$capital_league', 'count': {'$sum': 1}}},
    {'$project': {
        '_id': 1,
        'count': 1,
        'percentage': {'$multiply': [{'$divide': ['$count',
total_records]}, 100]}
    }},
    {'$sort': {'_id': 1}}
]

print("Results for Query 19: Clan Capital League (Counts and
Percentages)(excluding 'Unranked')\n")

results_capital_league = collection.aggregate(pipeline_capital_league)

capital_league_counts = {result['_id']: result for result in
results_capital_league}
sorted_results_capital_league = [capital_league_counts[league] for
league in league_order if league in capital_league_counts]

for result in sorted_results_capital_league:
    print(f"Capital League: {result['_id']}, Count: {result['count']},
Percentage: {result['percentage']:.1f}%")

# Query 20: Mean Member Level (Range and Top 5 Clans)

# Pipeline to calculate counts based on mean member level
pipeline_mean_member_level = [
    {'$match': {'num_members': {'$gte': 5}}},
    {'$group': {'_id': '$mean_member_level', 'count': {'$sum': 1}}},
    {'$sort': {'_id': 1}}
]

# Aggregating to get the counts for each mean member level
print("Results for Query 20: Mean Member Level (Range and Top 5
Clans)\n")

results_mean_member_level =
```

```
collection.aggregate(pipeline_mean_member_level)
for result in results_mean_member_level:
    print(f"Mean Member Level: {result['_id']}, Count:
{result['count']}")

top_5_clans_by_member_level = collection.find({'num_members': {'$gte':
5}}).sort('mean_member_level', -1).limit(5)
print("\nTop 5 Clans by Mean Member Level:\n")
for clan in top_5_clans_by_member_level:
    print(f"Top Clan: {clan['clan_name']}, Mean Member Level:
{clan['mean_member_level']}")

# Query 21: Mean Member Trophies (Range and Top 5 Clans)

# Pipeline to calculate counts based on mean member trophies
pipeline_mean_member_trophies = [
    {'$match': {'num_members': {'$gte': 5}}},  # Filter clans with at
least 5 members
    {'$project': {'trophies_range': {'$floor': {'$divide':
['$mean_member_trophies', 300]}}}},
    {'$group': {'_id': '$trophies_range', 'count': {'$sum': 1}}},
    {'$sort': {'_id': 1}}
]

# Aggregating to get the counts for each trophy range
print("Results for Query 21: Mean Member Trophies (Range and Top 5
Clans)\n")

results_mean_member_trophies =
collection.aggregate(pipeline_mean_member_trophies)
for result in results_mean_member_trophies:
    min_trophies = result['_id'] * 300
    max_trophies = (result['_id'] + 1) * 300
    print(f"Mean Member Trophies Range: {min_trophies} -
{max_trophies}, Count: {result['count']}")

top_5_clans_by_member_trophies = collection.find({'num_members':
{'$gte': 5}}).sort('mean_member_trophies', -1).limit(5)
print("\nTop 5 Clans by Mean Member Trophies:")
for clan in top_5_clans_by_member_trophies:
    print(f"Top Clan: {clan['clan_name']}, Mean Member Trophies:
{clan['mean_member_trophies']}")

## Data Visualization

# Visualization of Query 1: Clan Type Percentages
import matplotlib.pyplot as plt

# Ensuring the results are fully retrieved from the aggregation
pipeline
results_clan_type_percentage =
```

```
list(collection.aggregate(pipeline_clan_type_percentage))

labels = [result['_id'] for result in results_clan_type_percentage]
percentages = [result['percentage'] for result in
results_clan_type_percentage]

# Plotting
plt.figure(figsize=(10, 6))
plt.barh(labels, percentages, color='skyblue')
plt.title('Visualization of Query 1: Clan Type Percentages')
plt.xlabel('Percentage (%)')
plt.ylabel('Clan Type')
plt.tight_layout()
plt.show()

# Visualization of Query 2: Clan Location Percentages (Top 15
Countries)
import matplotlib.pyplot as plt

results_clan_location_percentage =
list(collection.aggregate(pipeline_clan_location_percentage))

locations = [str(result['_id']) if result['_id'] else 'Unknown' for
result in results_clan_location_percentage]
percentages = [result['percentage'] for result in
results_clan_location_percentage]

plt.figure(figsize=(10, 6))
plt.barh(locations, percentages, color='lightcoral')
plt.title('Visualization of Query 2: Clan Location Percentages (Top 15
Countries)')
plt.xlabel('Percentage')
plt.ylabel('Clan Location')
plt.tight_layout()
plt.show()

# Visualization of Query 3: Family Friendly Percentages
import matplotlib.pyplot as plt

results_family_friendly_percentage =
list(collection.aggregate(pipeline_family_friendly_percentage))

labels = [str(result['_id']) if result['_id'] is not None else
'Unknown' for result in results_family_friendly_percentage]
percentages = [result['percentage'] for result in
results_family_friendly_percentage]

plt.figure(figsize=(8, 5))
plt.bar(labels, percentages, color='lightseagreen')
plt.title('Visualization of Query 3: Family Friendly Percentages')
```

```
plt.xlabel('Family Friendly')
plt.ylabel('Percentage')
plt.tight_layout()
plt.show()

# Visualization of Query 4: Clan Level Counts (Logarithmic Scale on X-Axis)

results_clan_level_counts =
list(collection.aggregate(pipeline_clan_level_counts))

levels = [result['_id'] for result in results_clan_level_counts]
counts = [result['count'] for result in results_clan_level_counts]

sorted_levels_counts = sorted(zip(levels, counts), key=lambda x: x[0])
sorted_levels = [x[0] for x in sorted_levels_counts]
sorted_counts = [x[1] for x in sorted_levels_counts]

plt.figure(figsize=(10, 7))
bars = plt.barh(sorted_levels, sorted_counts, color='darkred')
plt.xscale('log')
plt.title('Visualization of Query 4: Clan Level Counts (Logarithmic
Scale on X-Axis)')
plt.xlabel('Counts (Log Scale)')
plt.ylabel('Clan Level')

# Adding the count labels next to the bars
for bar in bars:
    width = bar.get_width()
    plt.text(width + max(sorted_counts) * 0.02, bar.get_y() +
bar.get_height()/2,
             f"{int(width)}", va='center', fontsize=10)

plt.tight_layout()
plt.show()

# Visualization of Query 5: Clan Points Range (Logarithmic Scale on X-Axis)

results_clan_points_range =
list(collection.aggregate(pipeline_clan_points_range))

ranges_all = [f"{result['_id'] * range_size} - {(result['_id'] + 1) *
range_size}" for result in results_clan_points_range]
counts_all = [result['count'] for result in results_clan_points_range]

plt.figure(figsize=(10, 6))
bars = plt.barh(ranges_all, counts_all, color='grey')
plt.xscale('log')
plt.title('Clan Points Range Distribution (Logarithmic Scale on X-
Axis)')
```

```
plt.xlabel('Counts (Log Scale)')
plt.ylabel('Clan Points Range')

for bar in bars:
    width = bar.get_width()
    plt.text(width + max(counts_all) * 0.02, bar.get_y() +
bar.get_height()/2,
             f"{int(width)}", va='center', fontsize=10)

plt.tight_layout()
plt.show()

# Visualization of Query 6: Clan Builder Points Range (Logarithmic
Scale on X-Axis)

results_clan_builder_points_range =
list(collection.aggregate(pipeline_clan_builder_points_range))

ranges_all = [f"{result['_id']*range_size_builder}-{(result['_id']
+1)*range_size_builder}" for result in
results_clan_builder_points_range]
counts_all = [result['count'] for result in
results_clan_builder_points_range]

plt.figure(figsize=(10, 6))
bars = plt.barh(ranges_all, counts_all, color='lightskyblue')
plt.xscale('log')
plt.title('Clan Builder Points Range Distribution (Logarithmic Scale
on X-Axis)')
plt.xlabel('Counts (Log Scale)')
plt.ylabel('Builder Points Range')

for bar in bars:
    width = bar.get_width()
    plt.text(width + max(counts_all) * 0.02, bar.get_y() +
bar.get_height()/2,
             f"{int(width)}", va='center', fontsize=10)

plt.tight_layout()
plt.show()

# Visualization of Query 8: War Frequency (Counts and Percentages)

results_war_frequency =
list(collection.aggregate(pipeline_war_frequency))

labels = [result['_id'] for result in results_war_frequency]
counts = [result['count'] for result in results_war_frequency]
percentages = [result['percentage'] for result in
results_war_frequency]
```

```
plt.figure(figsize=(10, 6))
bars = plt.barh(labels, percentages, color='yellow')
plt.title('War Frequency Percentages')
plt.xlabel('Percentage (%)')
plt.ylabel('War Frequency')

for bar, percentage in zip(bars, percentages):
    width = bar.get_width()
    plt.text(width + max(percentages) * 0.02, bar.get_y() +
bar.get_height()/2,
             f"{percentage:.1f}%", va='center', fontsize=10)

plt.tight_layout()
plt.show()

# Visualization of Query 10: War Wins Range Distribution (Logarithmic
Scale on X-Axis)

results_war_wins_range =
list(collection.aggregate(pipeline_war_wins_range))

ranges_all = [f"{result['_id'] * range_size_wins}-{(result['_id'] + 1)
* range_size_wins}" for result in results_war_wins_range]
counts_all = [result['count'] for result in results_war_wins_range]

plt.figure(figsize=(10, 6))
bars = plt.barh(ranges_all, counts_all, color='lightseagreen')
plt.xscale('log')
plt.title('War Wins Range Distribution (Logarithmic Scale on X-Axis)')
plt.xlabel('Counts (Log Scale)')
plt.ylabel('War Wins Range')

for bar, count in zip(bars, counts_all):
    width = bar.get_width()
    plt.text(width + max(counts_all) * 0.02, bar.get_y() +
bar.get_height()/2,
             f"{count}", va='center', fontsize=10)

plt.tight_layout()
plt.show()

# Visualization of Query 13: Win Percentage Range Distribution, at
least 10 total games (wins + losses)

results_win_percentage_bins =
list(collection.aggregate(pipeline_win_percentage_bins))

ranges = [
    f"{result['_id'] * 10}%-{(result['_id'] + 1) * 10}%" if
result['_id'] < 9 else "90%-100%"
    for result in results_win_percentage_bins
```

```
]
counts = [result['count'] for result in results_win_percentage_bins]

plt.figure(figsize=(10, 6))
bars = plt.bar(ranges, counts, color='steelblue')
plt.title('Visualization of Query 13: Win Percentage Range
Distribution, at least 10 total games (wins + losses)')
plt.xlabel('Win Percentage Range')
plt.ylabel('Count')
plt.xticks(rotation=45)

for bar, count in zip(bars, counts):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height + 1, f"{count}",
ha='center', va='bottom', fontsize=10)

plt.tight_layout()
plt.show()

# Visualization of Query 14: Clan War League Counts and Percentages
(Excluding Unranked)

results_war_league_counts_exclude_unranked =
list(collection.aggregate(pipeline_war_league_counts_exclude_unranked)
)

leagues = [result['_id'] for result in
results_war_league_counts_exclude_unranked]
counts = [result['count'] for result in
results_war_league_counts_exclude_unranked]
percentages = [result['percentage'] for result in
results_war_league_counts_exclude_unranked]

plt.figure(figsize=(10, 6))
plt.bar(leagues, percentages, color='mediumseagreen')
plt.title('Visualization of Query 14: Clan War League Counts and
Percentages (Excluding Unranked)')
plt.xlabel('Clan War League')
plt.ylabel('Percentage (%)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Visualization of Query 15: Number of Members Counts (Logarithmic
Scale on X-Axis)

results_num_members_counts =
list(collection.aggregate(pipeline_num_members_counts))

num_members = [result['_id'] for result in results_num_members_counts]
```

```
counts = [result['count'] for result in results_num_members_counts]

plt.figure(figsize=(10, 8))
bars = plt.barh(num_members, counts, color='cornflowerblue')
plt.xscale('log')
plt.title('Visualization of Query 15: Number of Members Counts
(Logarithmic Scale on X-Axis)')
plt.xlabel('Counts (Log Scale)')
plt.ylabel('Number of Members')
plt.xticks(rotation=45)

for bar, count in zip(bars, counts):
    width = bar.get_width()
    plt.text(width + max(counts) * 0.01, bar.get_y() +
bar.get_height()/2,
             f"{count}", va='center', fontsize=10)

plt.tight_layout()
plt.show()

# Visualization of Query 16: Requirements (Counts with Logarithmic
Scale on X-Axis)

# For Required Builder Base Trophies
results_requirements_builder =
list(collection.aggregate(pipeline_requirements_builder))

ranges_builder = [f"{result['_id']}" for result in
results_requirements_builder]
counts_builder = [max(result['count'], 1) for result in
results_requirements_builder]  # Replace zero counts with 1 for log
scale

plt.figure(figsize=(12, 9))
bars_builder = plt.barh(ranges_builder, counts_builder,
color='lightcoral')
plt.xscale('log')
plt.title('Visualization of Query 16: Required Builder Base Trophies
(Counts with Log Scale)')
plt.ylabel('Required Builder Base Trophies')
plt.xlabel('Counts')

for bar, count in zip(bars_builder, counts_builder):
    width = bar.get_width()
    plt.text(width + max(counts_builder) * 0.01, bar.get_y() +
bar.get_height() / 2,
             f"{count}", va='center', fontsize=10)

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

```
# For Required Townhall Level
results_requirements_townhall =
list(collection.aggregate(pipeline_requirements_townhall))

townhall_levels = [f"TH {result['_id']}" for result in
results_requirements_townhall]
counts_townhall = [max(result['count'], 1) for result in
results_requirements_townhall]  # Replace zero counts with 1 for log
scale

plt.figure(figsize=(10, 6))
bars_townhall = plt.barh(townhall_levels, counts_townhall,
color='mediumseagreen')
plt.xscale('log')
plt.title('Visualization of Query 16: Required Townhall Level (Counts
with Log Scale)')
plt.ylabel('Required Townhall Level')
plt.xlabel('Counts')

for bar, count in zip(bars_townhall, counts_townhall):
    width = bar.get_width()
    plt.text(width + max(counts_townhall) * 0.01, bar.get_y() +
bar.get_height() / 2,
             f"{count}", va='center', fontsize=10)

plt.tight_layout()
plt.show()

# Visualization of Query 17: Clan Capital Hall Level (Counts with
Logarithmic Scale on X-Axis)

results_capital_hall =
list(collection.aggregate(pipeline_capital_hall))

capital_hall_levels = [result['_id'] for result in
results_capital_hall]
counts_capital_hall = [max(result['count'], 1) for result in
results_capital_hall]  # Replace zero counts with 1 for log scale

plt.figure(figsize=(10, 6))
bars_capital_hall = plt.barh(capital_hall_levels, counts_capital_hall,
color='darkgrey')
plt.xscale('log')
plt.title('Visualization of Query 17: Clan Capital Hall Level (Counts
with Log Scale on X-Axis)')
plt.xlabel('Counts (Log Scale)')
plt.ylabel('Clan Capital Hall Level')

for bar, count in zip(bars_capital_hall, counts_capital_hall):
```

```
    width = bar.get_width()
    plt.text(width + max(counts_capital_hall) * 0.01, bar.get_y() +
bar.get_height() / 2,
             f"{count}", va='center', fontsize=10)

plt.tight_layout()
plt.show()

# Visualization of Query 19: Clan Capital League (Counts Displayed)

leagues = [result['_id'] for result in sorted_results_capital_league]
counts = [result['count'] for result in sorted_results_capital_league]

plt.figure(figsize=(12, 8))
bars = plt.barh(leagues, counts, color='mediumorchid')

for bar, count in zip(bars, counts):
    width = bar.get_width()
    plt.text(width + max(counts) * 0.005, bar.get_y() +
bar.get_height()/2,
             f"{count}", va='center', fontsize=9)

plt.title('Visualization of Query 19: Clan Capital League (Counts
Displayed)', fontsize=14)
plt.xlabel('Count', fontsize=12)
plt.ylabel('Clan Capital League', fontsize=12)
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()

# Visualization of Query 21: Mean Member Trophies (Range and Top 5
Clans)

results_mean_member_trophies =
list(collection.aggregate(pipeline_mean_member_trophies))

bin_ranges = [f"{result['_id'] * 300}-{(result['_id'] + 1) * 300}" for
result in results_mean_member_trophies]
counts = [result['count'] for result in results_mean_member_trophies]

plt.figure(figsize=(12, 7))
bars = plt.bar(bin_ranges, counts, color='goldenrod')

for bar, count in zip(bars, counts):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height + max(counts) *
0.01,
             f"{count}", ha='center', va='bottom', fontsize=9)

plt.title('Visualization of Query 21: Mean Member Trophies (Range and
Top 5 Clans)', fontsize=14)
```

```
plt.xlabel('Mean Member Trophies Range', fontsize=12)
plt.ylabel('Count of Clans', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```