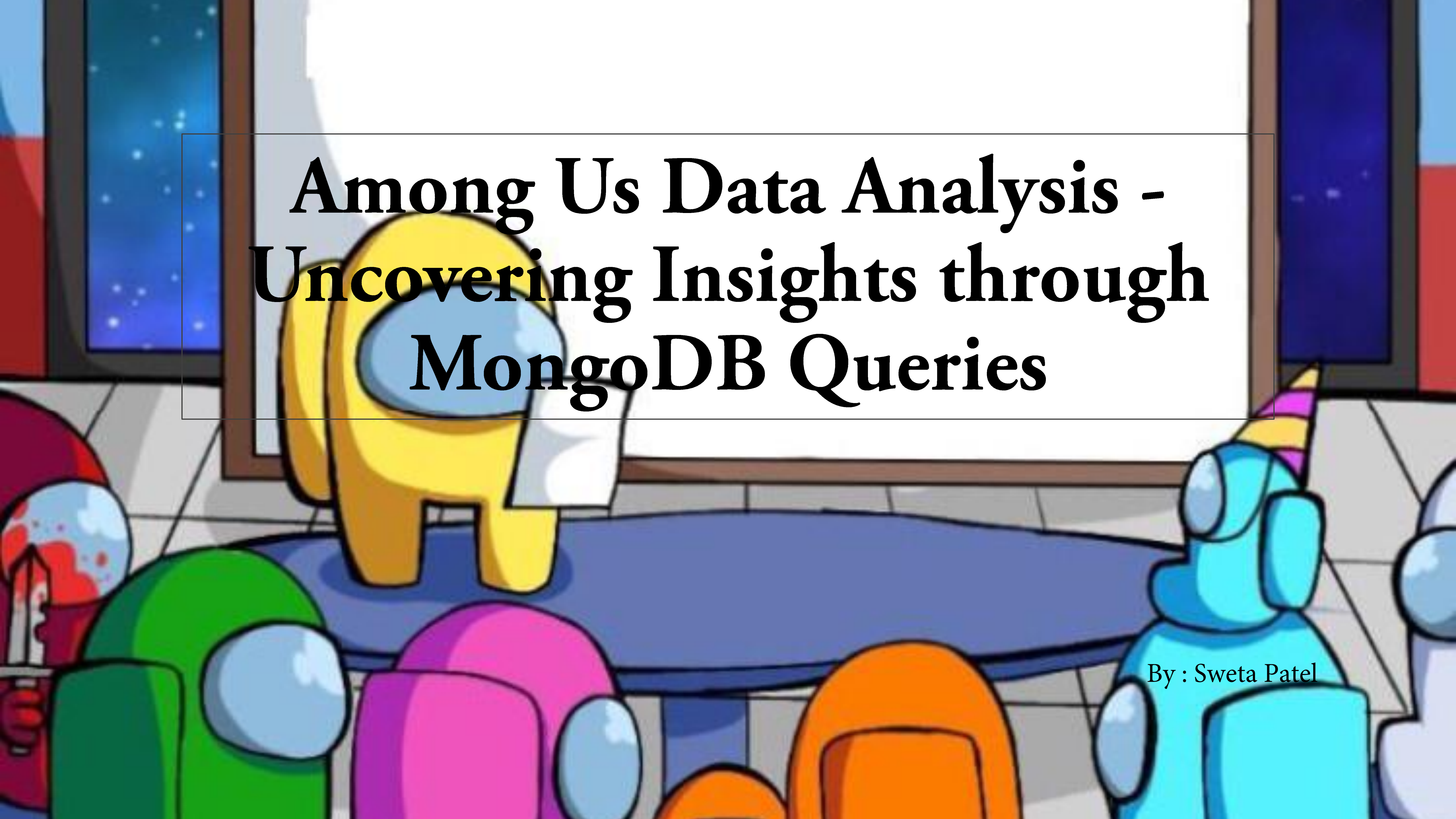


Among Us Data Analysis - Uncovering Insights through MongoDB Queries

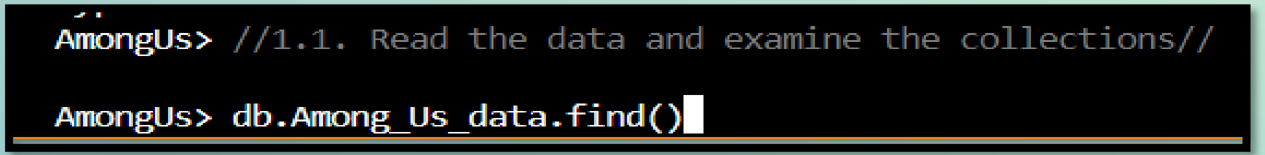
By : Sweta Patel

Introduction to Among Us Data Analysis Content

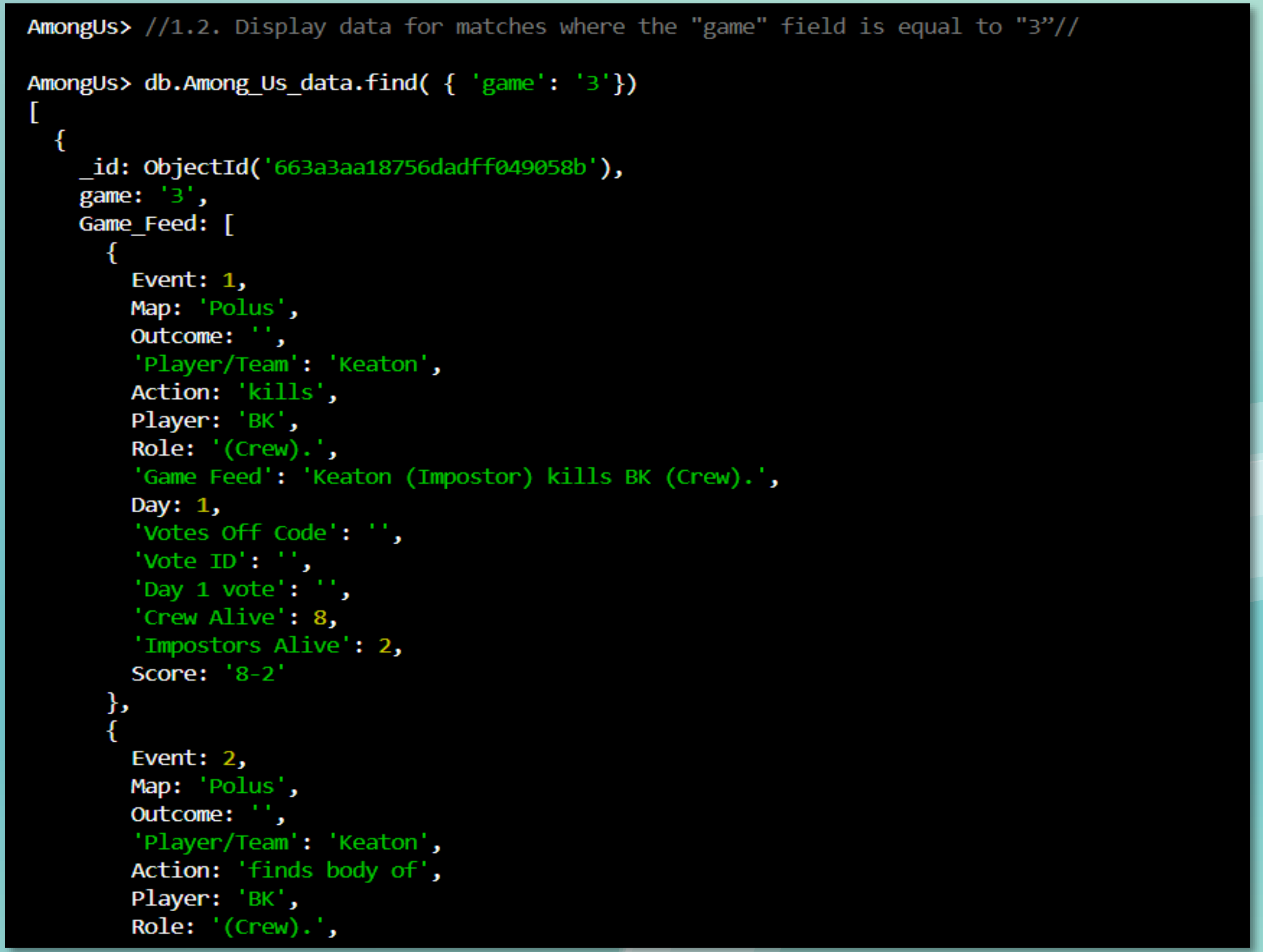
The code db.Among_Us_data.find() is used to retrieve all documents within the collection named "Among_Us_data" in the database. This code queries the database to find and display all data stored in that particular collection.

To present it in brief, this code simply retrieves all the data in the "Among_Us_data" collection, which can then be used for further analysis, visualization, or manipulation as needed.

This code is querying the Among_Us_data collection to find and display data for matches where the "game" field is equal to "3". The find() method is used to retrieve documents from the collection that match the specified query criteria, in this case, the game field being equal to 3. The result will show all the data entries that meet this condition.

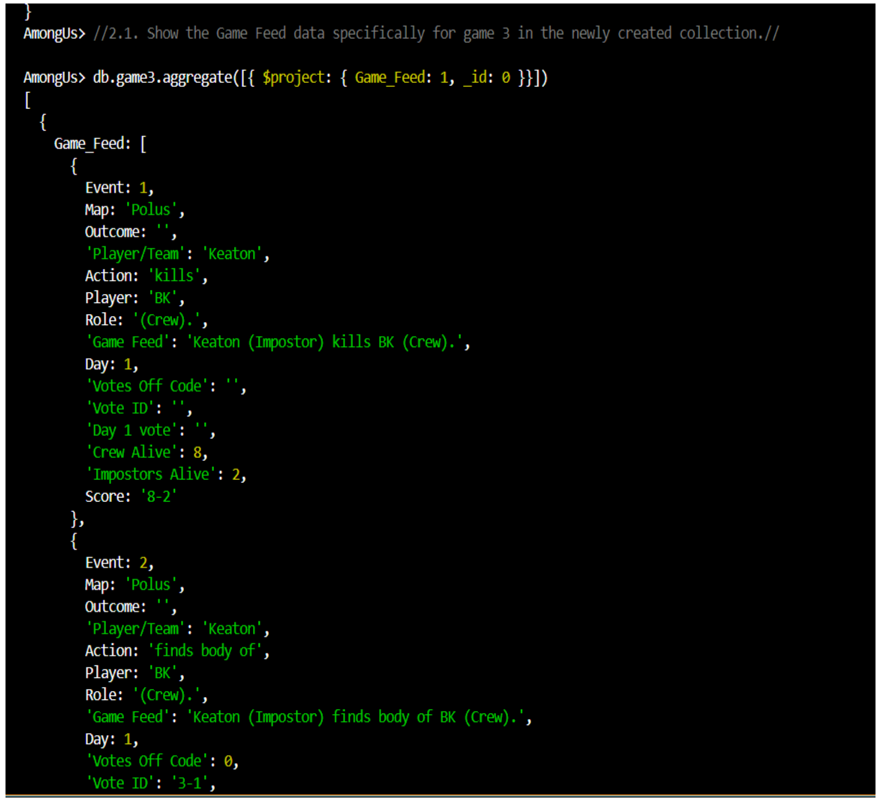


```
AmongUs> //1.1. Read the data and examine the collections//

AmongUs> db.Among_Us_data.find()
```

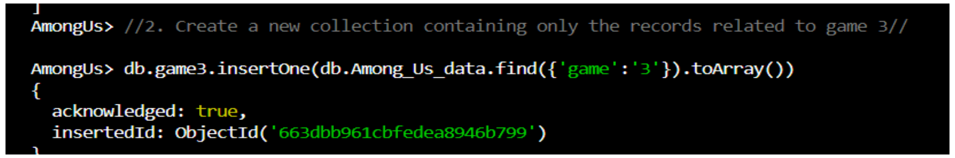


```
AmongUs> //1.2. Display data for matches where the "game" field is equal to "3"//

AmongUs> db.Among_Us_data.find( { 'game': '3'})
[
  {
    _id: ObjectId('663a3aa18756dadff049058b'),
    game: '3',
    Game_Feed: [
      {
        Event: 1,
        Map: 'Polus',
        Outcome: '',
        'Player/Team': 'Keaton',
        Action: 'kills',
        Player: 'BK',
        Role: '(Crew).',
        'Game Feed': 'Keaton (Impostor) kills BK (Crew).',
        Day: 1,
        'Votes Off Code': '',
        'Vote ID': '',
        'Day 1 vote': '',
        'Crew Alive': 8,
        'Impostors Alive': 2,
        Score: '8-2'
      },
      {
        Event: 2,
        Map: 'Polus',
        Outcome: '',
        'Player/Team': 'Keaton',
        Action: 'finds body of',
        Player: 'BK',
        Role: '(Crew).',
```

Game 3 Data Analysis Content



```
}
AmongUs> //2.1. Show the Game Feed data specifically for game 3 in the newly created collection.//

AmongUs> db.game3.aggregate([{ $project: { Game_Feed: 1, _id: 0 }}])
[
  {
    Game_Feed: [
      {
        Event: 1,
        Map: 'Polus',
        Outcome: '',
        'Player/Team': 'Keaton',
        Action: 'kills',
        Player: 'BK',
        Role: '(Crew).',
        'Game Feed': 'Keaton (Impostor) kills BK (Crew).',
        Day: 1,
        'Votes Off Code': '',
        'Vote ID': '',
        'Day 1 vote': '',
        'Crew Alive': 8,
        'Impostors Alive': 2,
        Score: '8-2'
      },
      {
        Event: 2,
        Map: 'Polus',
        Outcome: '',
        'Player/Team': 'Keaton',
        Action: 'finds body of',
        Player: 'BK',
        Role: '(Crew).',
        'Game Feed': 'Keaton (Impostor) finds body of BK (Crew).',
        Day: 1,
        'Votes Off Code': 0,
        'Vote ID': '3-1',
```



```
]
AmongUs> //2. Create a new collection containing only the records related to game 3//

AmongUs> db.game3.insertOne(db.Among_Us_data.find({'game':'3'}).toArray())
{
  acknowledged: true,
  insertedId: ObjectId('663dbb961cbfedea8946b799')
}
```

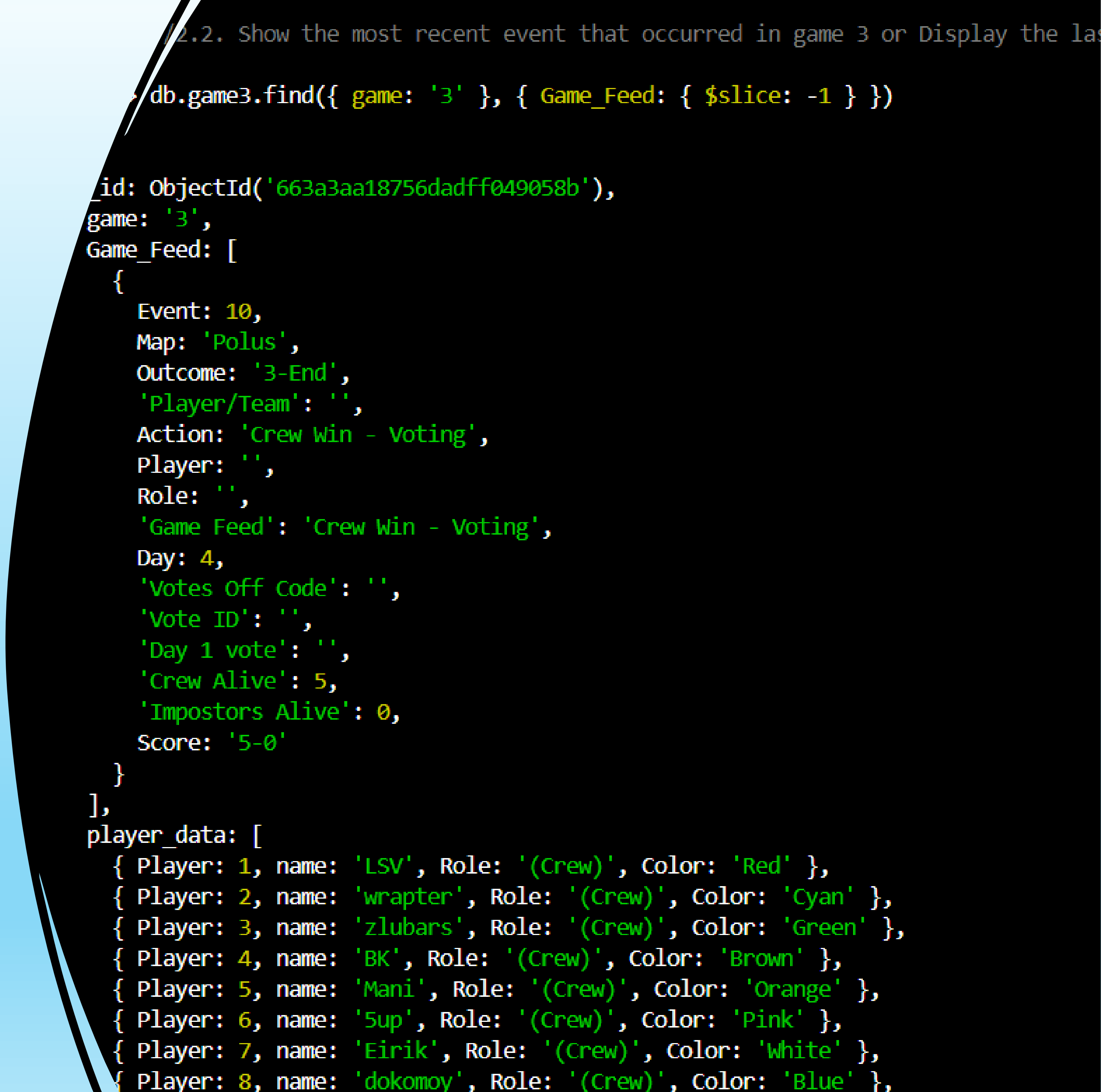
- The code provided is storing the result of a find query on the "Among_Us_data" collection for records with the field "game" equal to '3' into a new collection named "game3". The insertOne method is used to insert the results of the find query into the new collection.

- Following that, the code uses the aggregate method to project only the "Game_Feed" field from the documents in the "game3" collection. The $project operator is used to include or exclude specific fields in the output. In this case, the only field being included in the output is "Game_Feed" using { Game_Feed: 1, _id: 0 }. The _id field is explicitly excluded from the output using _id: 0. This ensures that only the "Game_Feed" field is displayed in the results.

- Overall, the code first creates a new collection with records related to game 3, and then retrieves and displays only the "Game_Feed" data for game 3 in the newly created collection.

Game 3 Data Analysis Content

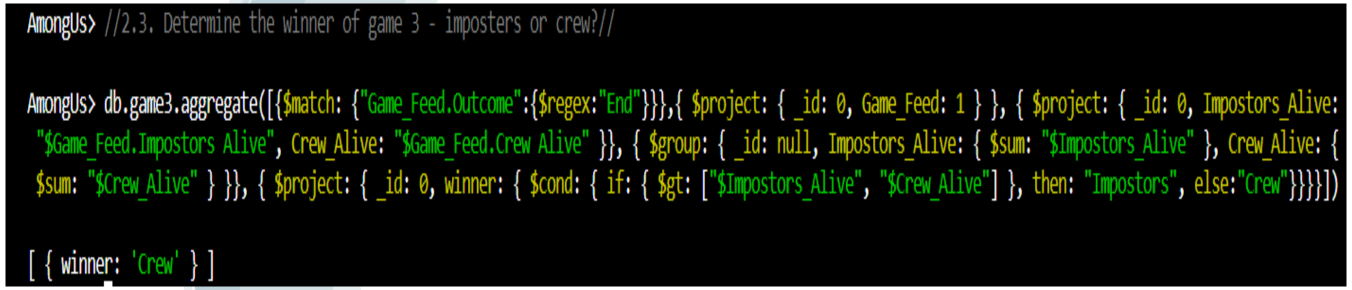
- This code uses the MongoDB query function `find` to retrieve data from the collection `game3`. It specifies the condition `{ game: '3' }` to filter out data related to game 3.
- The second parameter of the `find` function is an object that includes `{ Game_Feed: { $slice: -1 } }`. This parameter specifies that we want to return only the last item in the `Game_Feed` array of events for game 3.
- As a result, this code will display the most recent event that occurred in game 3.



```
/2.2. Show the most recent event that occurred in game 3 or Display the las

db.game3.find({ game: '3' }, { Game_Feed: { $slice: -1 } })

_id: ObjectId('663a3aa18756dadff049058b'),
game: '3',
Game_Feed: [
  {
    Event: 10,
    Map: 'Polus',
    Outcome: '3-End',
    'Player/Team': '',
    Action: 'Crew Win - Voting',
    Player: '',
    Role: '',
    'Game Feed': 'Crew Win - Voting',
    Day: 4,
    'Votes Off Code': '',
    'Vote ID': '',
    'Day 1 vote': '',
    'Crew Alive': 5,
    'Impostors Alive': 0,
    Score: '5-0'
  }
],
player_data: [
  { Player: 1, name: 'LSV', Role: '(Crew)', Color: 'Red' },
  { Player: 2, name: 'wrapter', Role: '(Crew)', Color: 'Cyan' },
  { Player: 3, name: 'zlubars', Role: '(Crew)', Color: 'Green' },
  { Player: 4, name: 'BK', Role: '(Crew)', Color: 'Brown' },
  { Player: 5, name: 'Mani', Role: '(Crew)', Color: 'Orange' },
  { Player: 6, name: '5up', Role: '(Crew)', Color: 'Pink' },
  { Player: 7, name: 'Eirik', Role: '(Crew)', Color: 'White' },
  { Player: 8, name: 'dokomoy', Role: '(Crew)', Color: 'Blue' },
```

Determine the Winner of Game 3 (Impostor or Crew)

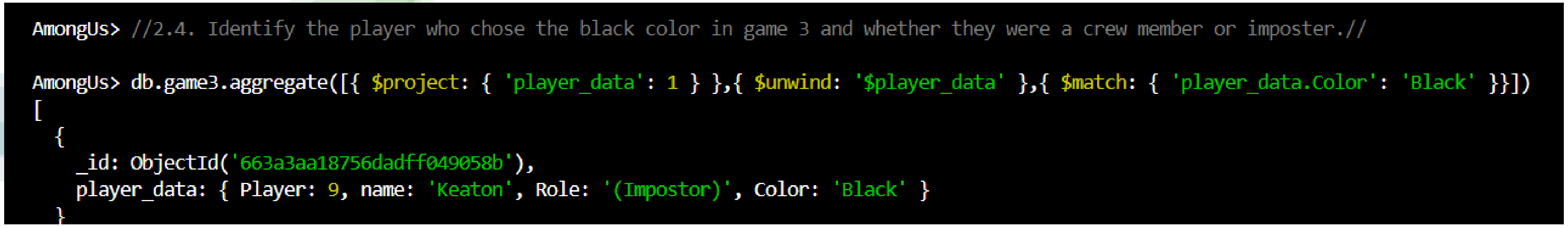
The code is using the aggregation framework in MongoDB to determine the winner of game 3 by matching the games with the outcome containing "End", projecting the Impostors and Crew alive counts, grouping the counts, and then deciding the winner based on the higher count of Impostors or Crew alive.



```
AmongUs> //2.3. Determine the winner of game 3 - imposters or crew?//

AmongUs> db.game3.aggregate([{$match: {"Game_Feed.Outcome":{$regex:"End"}}},{ $project: { _id: 0, Game_Feed: 1 } }, { $project: { _id: 0, Impostors_Alive:
 "$Game_Feed.Impostors Alive", Crew_Alive: "$Game_Feed.Crew Alive" }}, { $group: { _id: null, Impostors_Alive: { $sum: "$Impostors_Alive" }, Crew_Alive: {
 $sum: "$Crew_Alive" } }}, { $project: { _id: 0, winner: { $cond: { if: { $gt: ["$Impostors_Alive", "$Crew_Alive"] }, then: "Impostors", else:"Crew"}}}}])

[ { winner: 'Crew' } ]
```

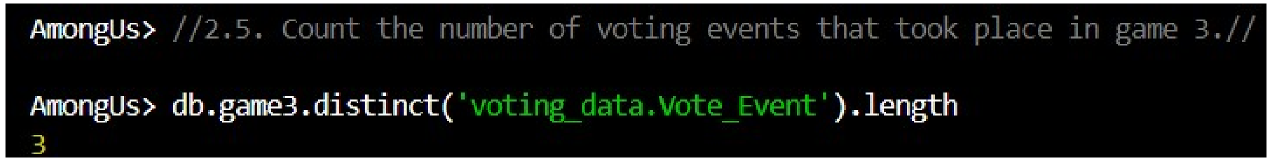
The result of the code is that the winner of game 3 is "Crew".

Detail the queries to identify players who chose black color, count voting events, and display voting events in game 3



```
AmongUs> //2.4. Identify the player who chose the black color in game 3 and whether they were a crew member or imposter.//

AmongUs> db.game3.aggregate([{ $project: { 'player_data': 1 } },{ $unwind: '$player_data' },{ $match: { 'player_data.Color': 'Black' }}])
[
  {
    _id: ObjectId('663a3aa18756dadff049058b'),
    player_data: { Player: 9, name: 'Keaton', Role: '(Impostor)', Color: 'Black' }
  }
```

The aggregation pipeline begins by projecting only the 'player_data' field from the game 3 data. Then, the data is unwound to separate each player's data into its own document. Next, a match stage is applied to filter for only player data with the color 'Black'. The result shows that in game 3, the player 'Keaton' chose the color Black and was an Impostor.



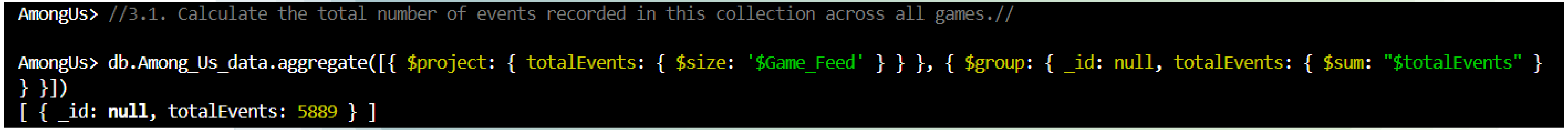
```
AmongUs> //2.5. Count the number of voting events that took place in game 3.//

AmongUs> db.game3.distinct('voting_data.Vote_Event').length
3
```

The aggregation pipeline starts by matching for game 3 data.
The data is unwound to separate each voting event into its own document. Then, a group stage is used to count the total number of voting events in game 3. The result shows that there were a total of 3 voting events in game 3.

These queries provide specific insights into the player who chose the Black color and the total number of voting events that occurred in game 3, which can be useful for analyzing gameplay strategies and interactions in Among Us.)

Total Events Analysis for all games

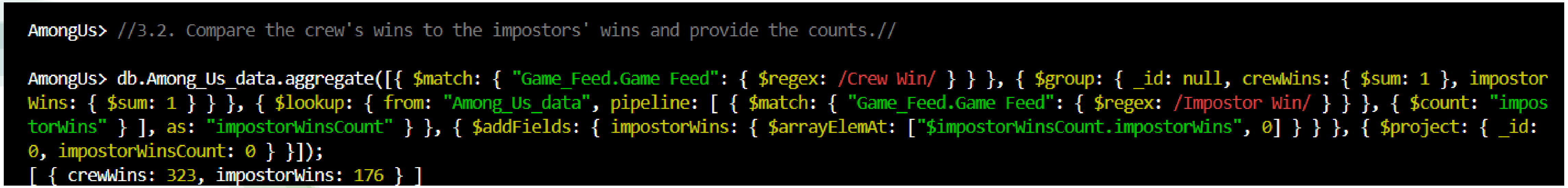
The aggregate method in MongoDB is used to perform aggregation operations on the data.
The aggregation pipeline consists of two stages: a. $project stage: This stage projects a new field called totalEvents with the size of the Game_Feed array in each document. b. $group stage: This stage groups all the documents together and calculates the sum of the totalEvents field across all documents.
The result of the aggregation operation is a single document with the _id value set to null and the totalEvents field containing the total number of events recorded in the collection across all games, which is 5889. This means that there were a total of 5889 events recorded in the collection across all games.



```
AmongUs> //3.1. Calculate the total number of events recorded in this collection across all games.//

AmongUs> db.Among_Us_data.aggregate([{ $project: { totalEvents: { $size: '$Game_Feed' } } }, { $group: { _id: null, totalEvents: { $sum: "$totalEvents" }
} }])
[ { _id: null, totalEvents: 5889 } ]
```

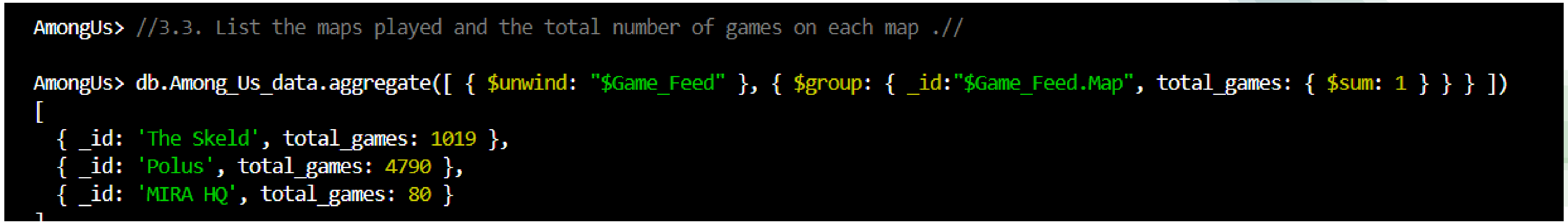
Showcase the analysis comparing crew wins to impostor wins and provide the counts

The provided aggregate query is designed to compare the number of wins between the crew and impostors in the Among Us dataset. The first stage of the query matches all game feeds that contain the term "Crew Win" and then groups the results to count the total number of crew wins. Similarly, it also counts the total number of impostor wins by matching game feeds with "Impostor Win" and grouping the results.
In the final stages of the query, a lookup operation is performed to retrieve the count of impostor wins from the Among Us dataset. After retrieving the count, the query adds this count to the existing impostor wins field. The project stage is used to remove unnecessary fields and structure the final output.
The query results show that the crew has won 323 times, while the impostors have won 176 times in the Among Us games based on the dataset analyzed.



```
AmongUs> //3.2. Compare the crew's wins to the impostors' wins and provide the counts.//

AmongUs> db.Among_Us_data.aggregate([{ $match: { "Game_Feed.Game Feed": { $regex: /Crew Win/ } } }, { $group: { _id: null, crewWins: { $sum: 1 }, impostor
Wins: { $sum: 1 } } }, { $lookup: { from: "Among_Us_data", pipeline: [ { $match: { "Game_Feed.Game Feed": { $regex: /Impostor Win/ } } }, { $count: "impos
torWins" } ], as: "impostorWinsCount" } }, { $addFields: { impostorWins: { $arrayElemAt: ["$impostorWinsCount.impostorWins", 0] } } }, { $project: { _id:
0, impostorWinsCount: 0 } }]);
[ { crewWins: 323, impostorWins: 176 } ]
```

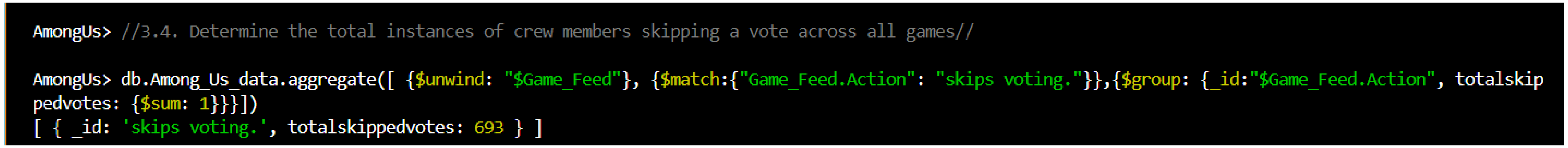
Display the query results listing the maps played in the games and the total number of games on each map.



```
AmongUs> //3.3. List the maps played and the total number of games on each map .//

AmongUs> db.Among_Us_data.aggregate([ { $unwind: "$Game_Feed" }, { $group: { _id:"$Game_Feed.Map", total_games: { $sum: 1 } } } ])
[
  { _id: 'The Skeld', total_games: 1019 },
  { _id: 'Polus', total_games: 4790 },
  { _id: 'MIRA HQ', total_games: 80 }
```

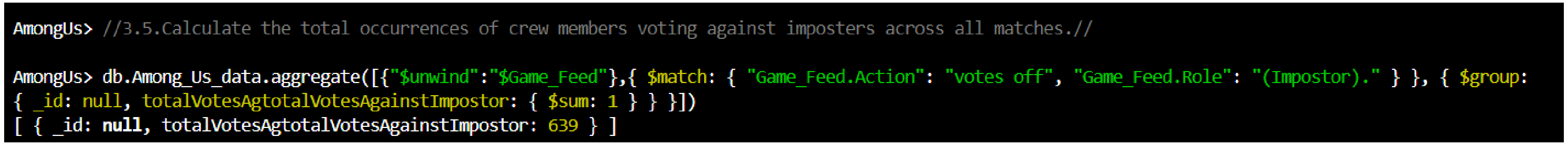
The code provided uses the aggregate function in MongoDB to first unwind the "Game_Feed" array in the data, allowing us to access each individual game. Then, it groups the games by the "Map" field and calculates the total number of games played on each map using the $sum function.

Explain the query to determine the total instances of crew members skipping votes and voting against impostors across all games



```
AmongUs> //3.4. Determine the total instances of crew members skipping a vote across all games//

AmongUs> db.Among_Us_data.aggregate([ {$unwind: "$Game_Feed"}, {$match:{"Game_Feed.Action": "skips voting."}},{$group: {_id:"$Game_Feed.Action", totalskip
pedvotes: {$sum: 1}}}])
[ { _id: 'skips voting.', totalskippedvotes: 693 } ]
```

Among all games analyzed, there were 693 instances of crew members skipping a vote. This data was gathered by unwinding the 'Game_Feed' array, filtering for actions labeled as 'skips voting,' and then aggregating the results to calculate the total number of skipped votes."



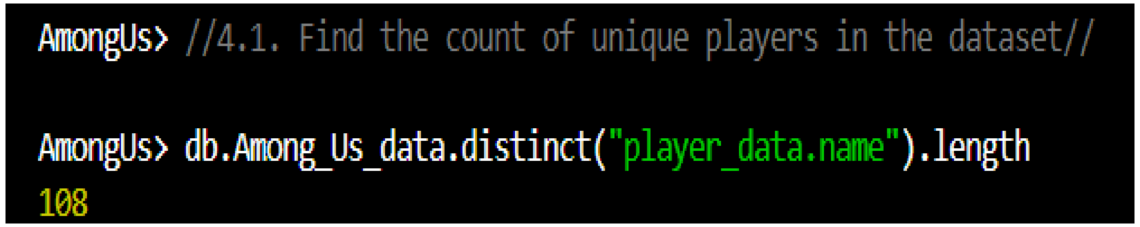
```
AmongUs> //3.5.Calculate the total occurrences of crew members voting against imposters across all matches.//

AmongUs> db.Among_Us_data.aggregate([{"$unwind":"$Game_Feed"},{ $match: { "Game_Feed.Action": "votes off", "Game_Feed.Role": "(Impostor)." } }, { $group:
{ _id: null, totalVotesAgtotalVotesAgainstImpostor: { $sum: 1 } } }])
[ { _id: null, totalVotesAgtotalVotesAgainstImpostor: 639 } ]
```

The $group stage aggregates the filtered documents based on a specified field, in this case, the "votes off" action and "(Impostor)" role from the Game_Feed array. By using the $sum operator, the stage calculates the total number of occurrences where crew members have voted off impostors in the game. This stage helps to consolidate the data and provides a singular count of crew members actively participating in identifying and voting off impostors. In this case, the result means that in the data set, crew members voted against imposters a total of 639 times

Discuss the query to find the count of unique players

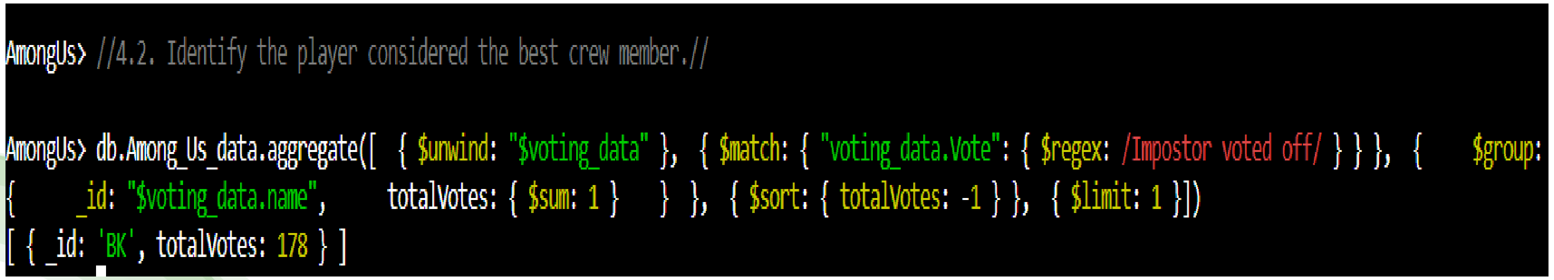
- The code using the distinct method on the Player field in the Among_Us_data collection. This method returns an array of unique values for the specified field. By calling the length property on this array, we can find out the count of unique players in the dataset.

- In this case, the result of the code execution is 108, which means that there are 108 unique players in the dataset. This information can be useful for analyzing player engagement, diversity, and other metrics in the context of the Among Us game.



```
AmongUs> //4.1. Find the count of unique players in the dataset//

AmongUs> db.Among_Us_data.distinct("player_data.name").length
108
```

Identify the player considered the best crew member



```
AmongUs> //4.2. Identify the player considered the best crew member.//

AmongUs> db.Among_Us_data.aggregate([  { $unwind: "$voting_data" },  { $match: { "voting_data.Vote": { $regex: /Impostor voted off/ } } },  {     $group:
{      _id: "$voting_data.name",      totalVotes: { $sum: 1 }     }  },  { $sort: { totalVotes: -1 } },  { $limit: 1 }])
[ { _id: 'BK', totalVotes: 178 } ]
```

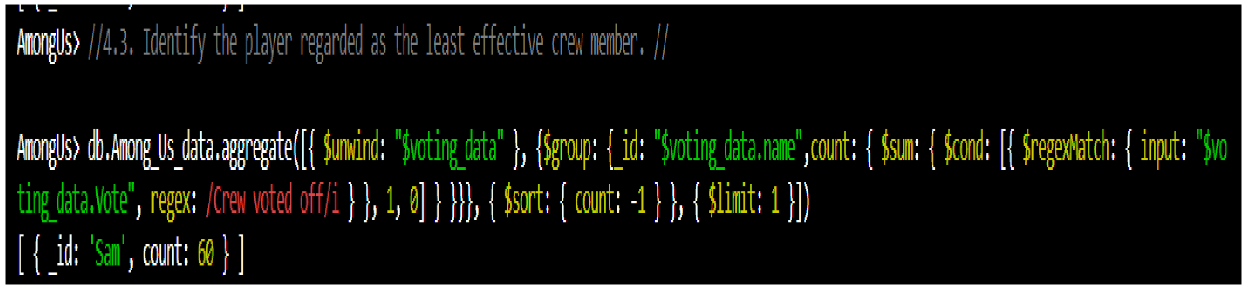
- The code provided uses the aggregate function to unwind the "voting_data" array, match only the documents where the vote is for an impostor to be voted off, group the data by crew member name, calculate the total number of votes each crew member received, sort the results in descending order based on total votes, and limit the results to only the top crew member with the highest number of votes.
- The results of the aggregation show that the crew member with the most votes for impostor to be voted off is designated as 'BK' with a total of 178 votes.

Identify the player regarded as the least effective crew member

- This code is using the aggregation framework in MongoDB to analyze voting data from the game Among Us. It first uses the $unwind operator to deconstruct the array field "voting_data" which contains voting information for each player. Then it groups the data by the name of the player and calculates the count of times they were voted off by the crew.

- The $regexMatch operator is used within the $cond operator to check if the vote was against a crew member. If it matches, it counts as 1, otherwise 0.

- The results show that the player "Sam" was voted off 60 times by the crew, making them the least effective crew member in the game

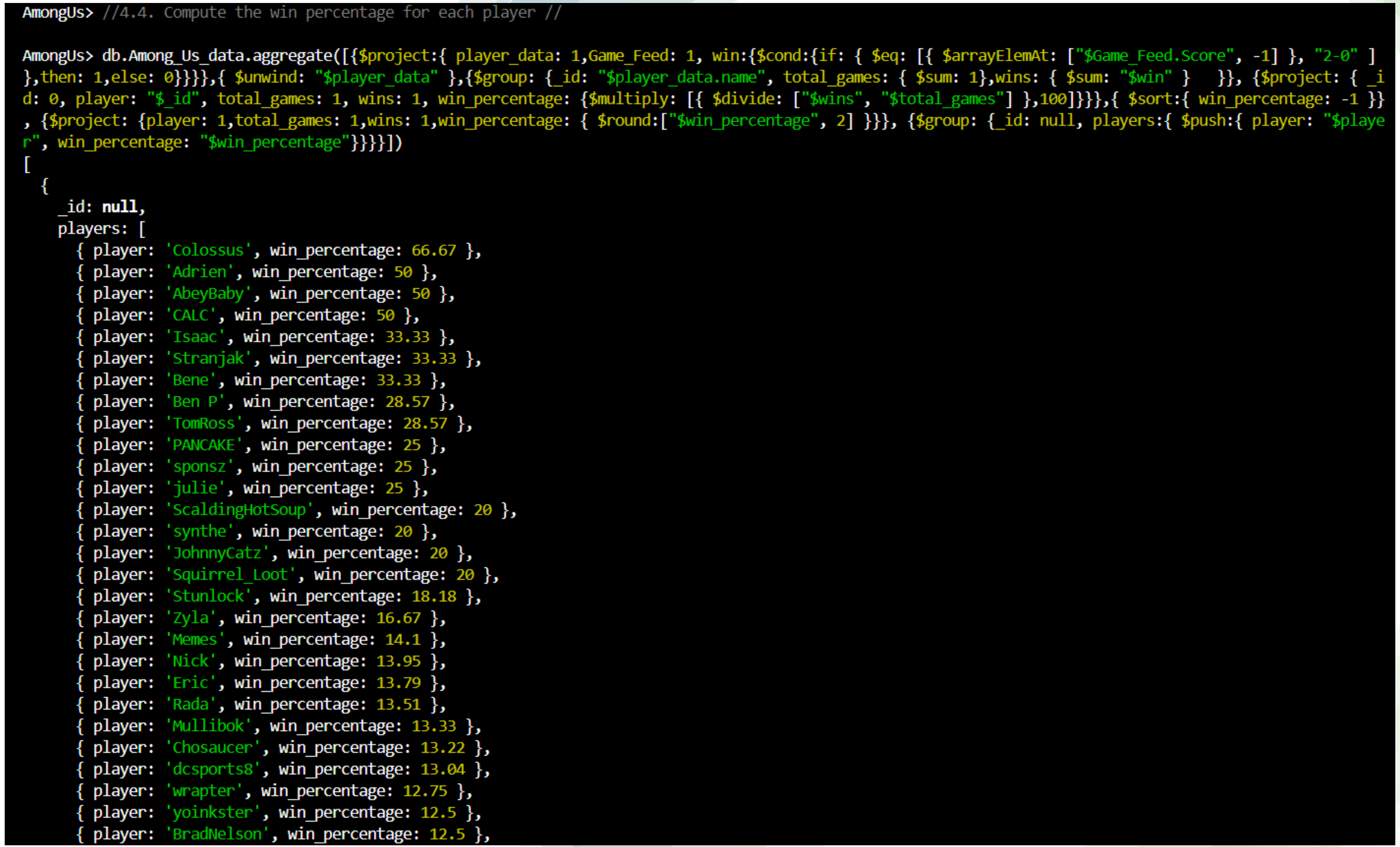


```
AmongUs> //4.3. Identify the player regarded as the least effective crew member. //

AmongUs> db.Among_Us_data.aggregate([{ $unwind: "$voting_data" }, {$group: {_id: "$voting_data.name",count: { $sum: { $cond: [{ $regexMatch: { input: "$vo
ting_data.Vote", regex: /Crew voted off/i } }, 1, 0] } }}}, { $sort: { count: -1 } }, { $limit: 1 }])
[ { _id: 'Sam', count: 60 } ]
```

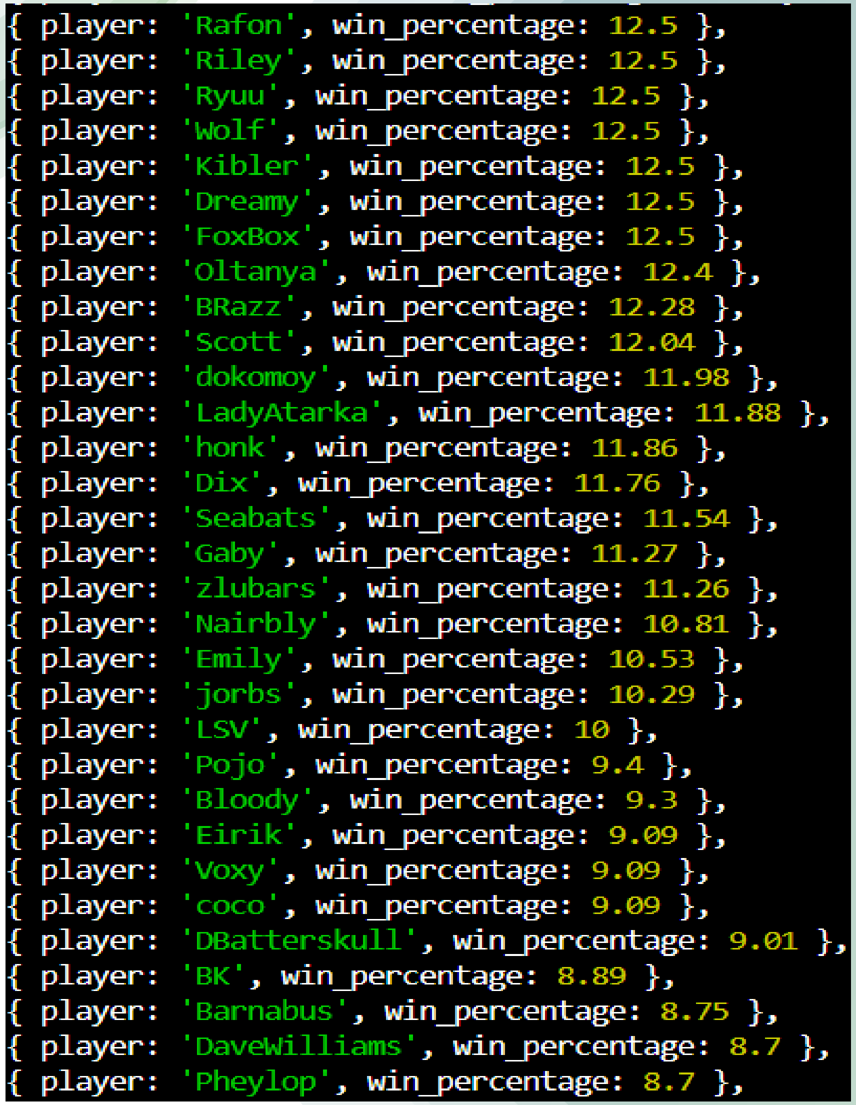
Compute the win percentage for each player

- The code first projects the player data, game feed, and calculates a win variable based on the game score. It then unwinds the player data, groups the players by name, calculates the total games played and total wins for each player. It projects the player name and win percentage, sorts the results in descending order by win percentage, and finally displays the top players with their respective win percentages.

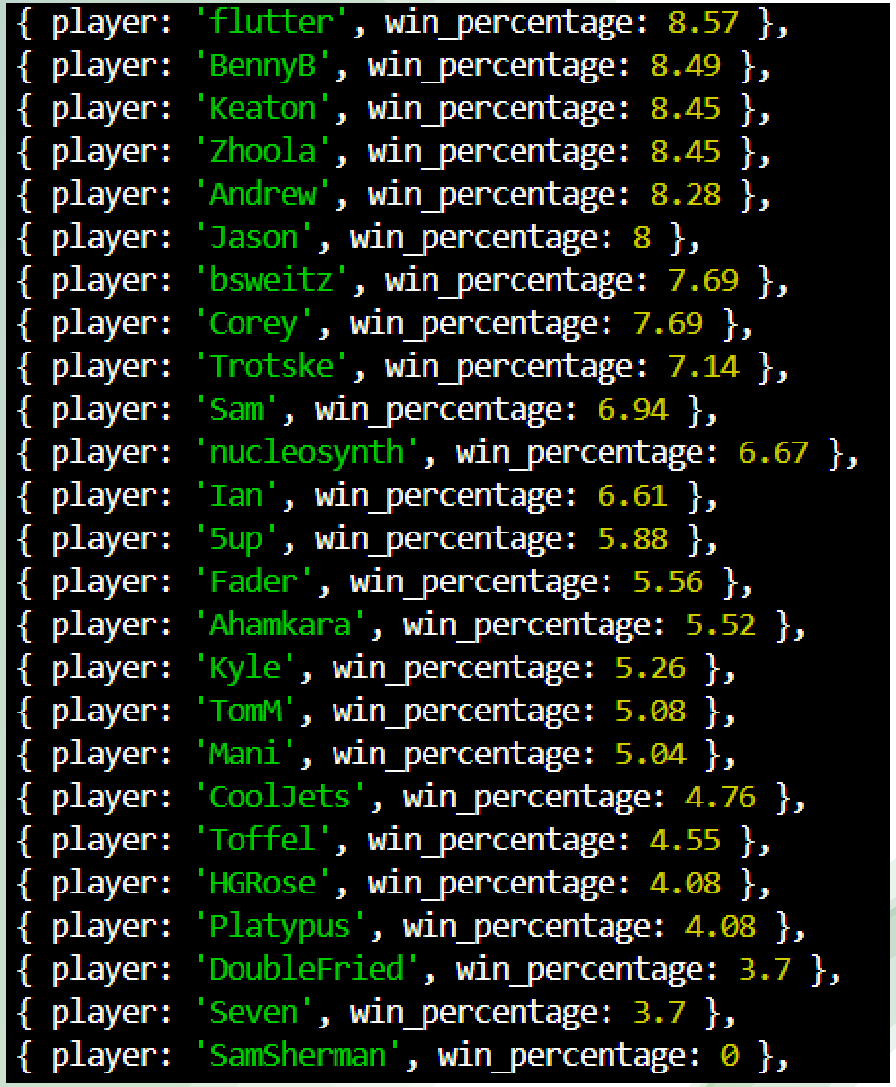
- The results show the win percentage for each player, with the top player having a win percentage of 66.67% and the lowest player having a win percentage of 3.7%. This analysis helps determine the success rate of each player in Among Us games.



```
AmongUs> //4.4. Compute the win percentage for each player //

AmongUs> db.Among_Us_data.aggregate([{$project:{ player_data: 1,Game_Feed: 1, win:{$cond:{if: { $eq: [{ $arrayElemAt: ["$Game_Feed.Score", -1] }, "2-0" ]
},then: 1,else: 0}}}},{ $unwind: "$player_data" },{$group: {_id: "$player_data.name", total_games: { $sum: 1},wins: { $sum: "$win" }   }}, {$project: { _i
d: 0, player: "$_id", total_games: 1, wins: 1, win_percentage: {$multiply: [{ $divide: ["$wins", "$total_games"] },100]}}},{ $sort:{ win_percentage: -1 }}
, {$project: {player: 1,total_games: 1,wins: 1,win_percentage: { $round:["$win_percentage", 2] }}}, {$group: {_id: null, players:{ $push:{ player: "$playe
r", win_percentage: "$win_percentage"}}}}])
[
  {
    _id: null,
    players: [
      { player: 'Colossus', win_percentage: 66.67 },
      { player: 'Adrien', win_percentage: 50 },
      { player: 'AbeyBaby', win_percentage: 50 },
      { player: 'CALC', win_percentage: 50 },
      { player: 'Isaac', win_percentage: 33.33 },
      { player: 'Stranjak', win_percentage: 33.33 },
      { player: 'Bene', win_percentage: 33.33 },
      { player: 'Ben P', win_percentage: 28.57 },
      { player: 'TomRoss', win_percentage: 28.57 },
      { player: 'PANCAKE', win_percentage: 25 },
      { player: 'sponsz', win_percentage: 25 },
      { player: 'julie', win_percentage: 25 },
      { player: 'ScaldingHotSoup', win_percentage: 20 },
      { player: 'synthe', win_percentage: 20 },
      { player: 'JohnnyCatz', win_percentage: 20 },
      { player: 'Squirrel_Loot', win_percentage: 20 },
      { player: 'Stunlock', win_percentage: 18.18 },
      { player: 'Zyla', win_percentage: 16.67 },
      { player: 'Memes', win_percentage: 14.1 },
      { player: 'Nick', win_percentage: 13.95 },
      { player: 'Eric', win_percentage: 13.79 },
      { player: 'Rada', win_percentage: 13.51 },
      { player: 'Mullibok', win_percentage: 13.33 },
      { player: 'Chosaucer', win_percentage: 13.22 },
      { player: 'dcsports8', win_percentage: 13.04 },
      { player: 'wrapter', win_percentage: 12.75 },
      { player: 'yoinkster', win_percentage: 12.5 },
      { player: 'BradNelson', win_percentage: 12.5 },
```



```
{ player: 'Rafon', win_percentage: 12.5 },
{ player: 'Riley', win_percentage: 12.5 },
{ player: 'Ryuu', win_percentage: 12.5 },
{ player: 'Wolf', win_percentage: 12.5 },
{ player: 'Kibler', win_percentage: 12.5 },
{ player: 'Dreamy', win_percentage: 12.5 },
{ player: 'FoxBox', win_percentage: 12.5 },
{ player: 'Oltanya', win_percentage: 12.4 },
{ player: 'BRazz', win_percentage: 12.28 },
{ player: 'Scott', win_percentage: 12.04 },
{ player: 'dokomoy', win_percentage: 11.98 },
{ player: 'LadyAtarka', win_percentage: 11.88 },
{ player: 'honk', win_percentage: 11.86 },
{ player: 'Dix', win_percentage: 11.76 },
{ player: 'Seabats', win_percentage: 11.54 },
{ player: 'Gaby', win_percentage: 11.27 },
{ player: 'zlubars', win_percentage: 11.26 },
{ player: 'Nairbly', win_percentage: 10.81 },
{ player: 'Emily', win_percentage: 10.53 },
{ player: 'jorbs', win_percentage: 10.29 },
{ player: 'LSV', win_percentage: 10 },
{ player: 'Pojo', win_percentage: 9.4 },
{ player: 'Bloody', win_percentage: 9.3 },
{ player: 'Eirik', win_percentage: 9.09 },
{ player: 'Voxy', win_percentage: 9.09 },
{ player: 'coco', win_percentage: 9.09 },
{ player: 'DBatterskull', win_percentage: 9.01 },
{ player: 'BK', win_percentage: 8.89 },
{ player: 'Barnabus', win_percentage: 8.75 },
{ player: 'DaveWilliams', win_percentage: 8.7 },
{ player: 'Pheylop', win_percentage: 8.7 },
```



```
{ player: 'flutter', win_percentage: 8.57 },
{ player: 'BennyB', win_percentage: 8.49 },
{ player: 'Keaton', win_percentage: 8.45 },
{ player: 'Zhoola', win_percentage: 8.45 },
{ player: 'Andrew', win_percentage: 8.28 },
{ player: 'Jason', win_percentage: 8 },
{ player: 'bsweitz', win_percentage: 7.69 },
{ player: 'Corey', win_percentage: 7.69 },
{ player: 'Trotske', win_percentage: 7.14 },
{ player: 'Sam', win_percentage: 6.94 },
{ player: 'nucleosynth', win_percentage: 6.67 },
{ player: 'Ian', win_percentage: 6.61 },
{ player: '5up', win_percentage: 5.88 },
{ player: 'Fader', win_percentage: 5.56 },
{ player: 'Ahamkara', win_percentage: 5.52 },
{ player: 'Kyle', win_percentage: 5.26 },
{ player: 'TomM', win_percentage: 5.08 },
{ player: 'Mani', win_percentage: 5.04 },
{ player: 'CoolJets', win_percentage: 4.76 },
{ player: 'Toffel', win_percentage: 4.55 },
{ player: 'HGRose', win_percentage: 4.08 },
{ player: 'Platypus', win_percentage: 4.08 },
{ player: 'DoubleFried', win_percentage: 3.7 },
{ player: 'Seven', win_percentage: 3.7 },
{ player: 'SamSherman', win_percentage: 0 },
```