USE CASE STUDY REPORT

Group No.: Group 24

Student Name: Charles Schatmeyer

Executive Summary: CompeteNow is a platform designed to democratize and simplify the process of running and taking part in competitive events by creating a centralized, online hub. As of now, it can be extremely difficult to organize a sporting event without significant volunteer hours, or the backing of large academic or nation-wide institutions. That leaves a hole that CompeteNow fills, letting everyone get involved with the amazing world of competition.

The database was designed to consider all aspects of a sporting event, including the players, spectators, teams, venues, managers, and more. These aspects were identified, then mapped out onto EER and UML diagrams. These were then translated to a relational model, with attention placed to primary and foreign keys. The model was fully implemented in MySQL, and partially implemented in MongoDB. The MySQL implementation is additionally connected to a prototype python application, allowing for database querying and data visualization.

The CompeteNow platform is far from a public release build, but the potential in the framework is present. The database can store and control all the critical pieces of a competition infrastructure. Next steps would include additions to the model to account for more edge-case competition scenarios, and a further build-out of the user interface.

I. Introduction

Competition is an essential component to the human experience. Throughout history, in times of famine and fortune, we have always had time for competing against one another. Competition is an incredibly important outlet for the natural human drive to improve and be the best, and through these outlets a community, or even a family, can be formed, in players and spectators alike.

Despite this all, competitive events are extremely difficult to organize, and are often left only to academic institutions or billion-dollar sports leagues. There is simply too much to manage, from booking venues, making teams, recruiting players, and managing spectators. With that said, a centralized, online space to enable the creation and management of competition could solve these issues, and democratize the ability to run and participate in sporting events.

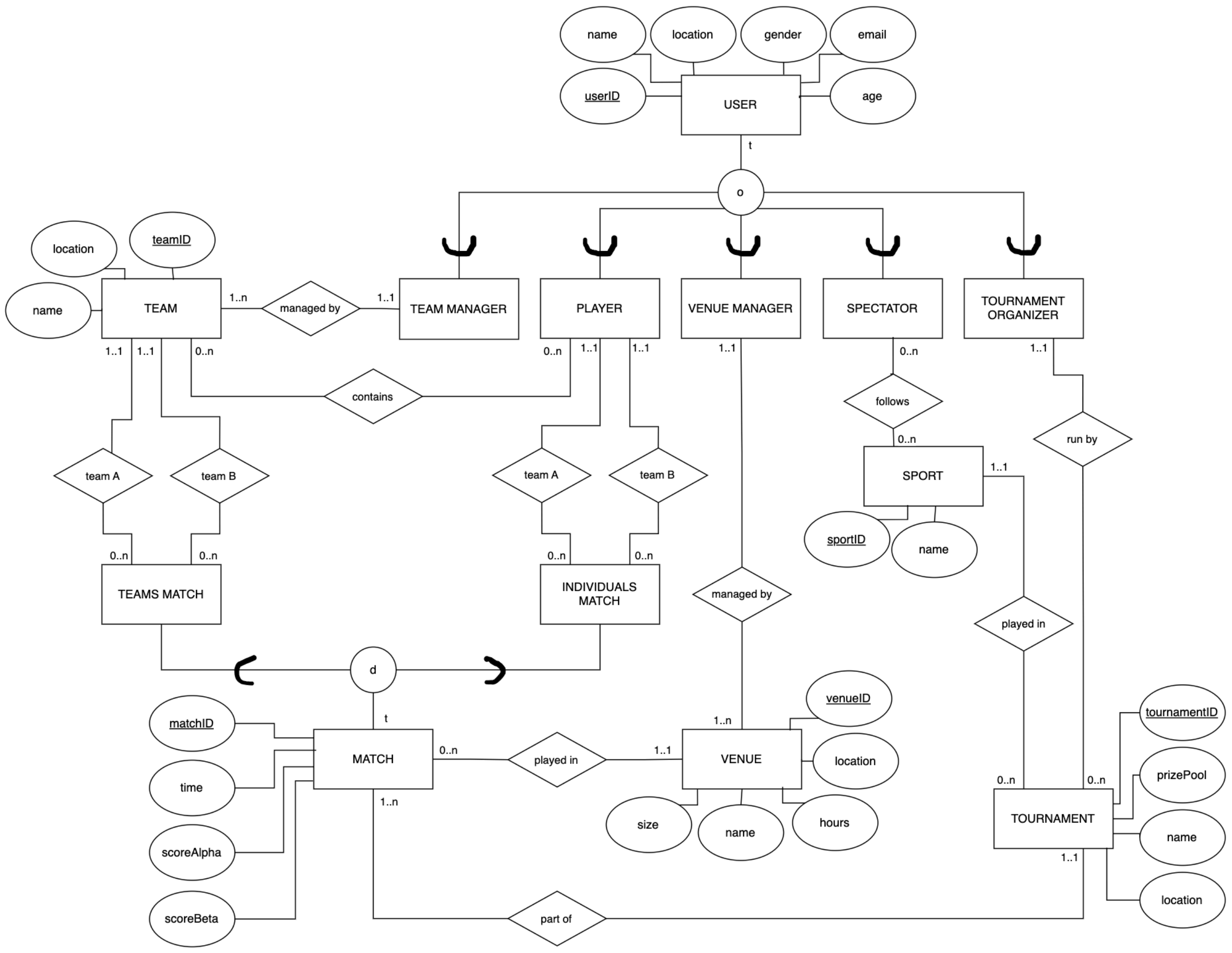
We envision a tournament organization platform, named CompeteNow, to give a space to plan and run matches and tournaments, giving consideration to players, teams, tournament organizers, venue managers, and spectators. Our central guiding vision is to provide everyone with a platform to join competition without needing to perform at the highest echelons of professional play.

For this concept to work, we will need to give any user that signs up the ability to be one or many of a Player, a Team Manager, a Venue Manager, a Tournament Organizer, and a Spectator. Players will be able to join tournaments, join teams, and play in individuals matches. Team Managers control teams, made of up players, which are able to also join tournaments and play in teams matches. Matches will be part of tournaments, have a score and time, and play in a venue. Venues will be controlled by venue managers, and have specific hours and a location. Finally, spectators will be able to follow their favorite sport and see times for tournaments and matches.

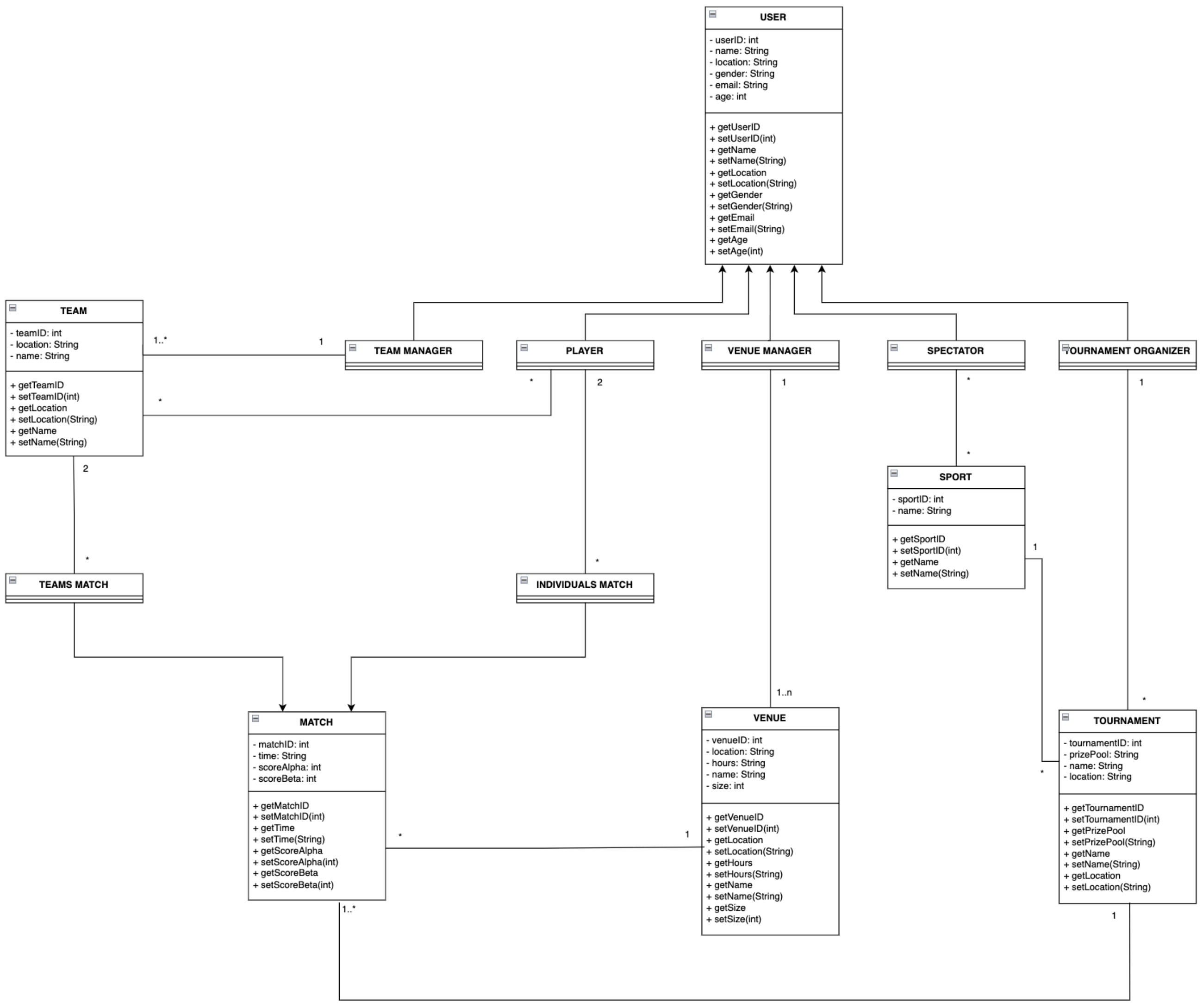
We see CompeteNow becoming a central hub for competition, and all the events and discussion around it, at the non-professional or academic level. With this platform created, we'll enable more people than ever to take part in their favorite sports, and in doing so, win glory, friends, and memorable experiences.

II. Conceptual Data Modeling

1. EER Model



2. UML Model



III. Mapping Conceptual Model to Relational Model

Relational Model (Note that Primary Key is Underlined and *Foreign Key* is Italicized):

User(UserID, name, location, gender, email, age)

TeamManager(*UserID*)

Player(*UserID*)

VenueManager(*UserID*)

Spectator(*UserID*)

TournamentOrganizer(*UserID*)

Venue(VenueID, size, name, hours, location, *VenueManager-UserID*)

Sport(SportID, name)

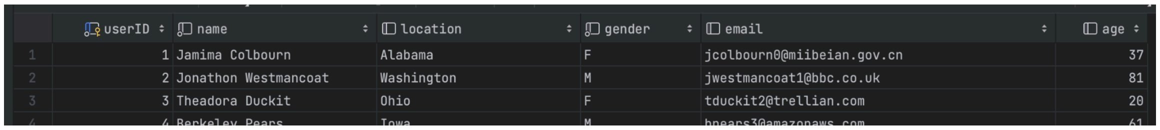
SpectatorFollows(*SportID*, *Spectator-UserID*)

Tournament(<u>TournamentID</u>, prize, name, location, *TournamentOrganizer-UserID, SportID*)
Team(<u>TeamID</u>, name, location, *TeamManager-UserID*)
TeamPlayer(<u>*TeamID*</u>, <u>*Player-UserID*</u>)
Match(<u>MatchID</u>, time, scoreAlpha, scoreBeta, *VenueID, TournamentID*)
TeamsMatch(<u>*TeamID[Alpha]*</u>, <u>*TeamID[Beta]*</u>)
IndividualsMatch(<u>*Player-UserID[Alpha]*</u>, <u>*Player-UserID[Beta]*</u>)

IV. Implementation of Relation Model via MySQL and NoSQL

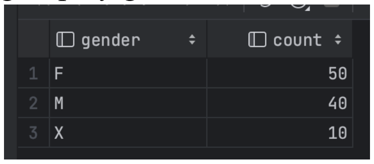
The database was implemented in MySQL, and the following eight queries were performed.

Q1: Simple - Get all information from User
select * from user;



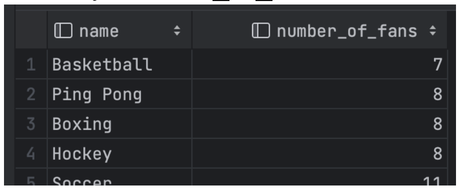
	userID	name	location	gender	email	age
1	1	Jamima Colbourn	Alabama	F	jcolbourn0@miibeian.gov.cn	37
2	2	Jonathon Westmancoat	Washington	M	jwestmancoat1@bbc.co.uk	81
3	3	Theadora Duckit	Ohio	F	tduckit2@trellian.com	20
4	4	Berkeley Pears	Iowa	M	bpears3@amazonaws.com	61

Q2: Aggregate - Get the count of all users in each gender
select gender, COUNT(*) as count
from user
group by gender;



	gender	count
1	F	50
2	M	40
3	X	10

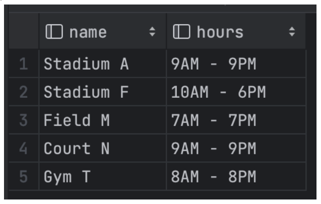
Q3: Inner/Outer Join - Get the count of the number of fans of sport
select name, COUNT(*) as number_of_fans
from sport s
right outer join spectator_follows sf
on s.sportID = sf.sport
group by name
order by number_of_fans;



	name	number_of_fans
1	Basketball	7
2	Ping Pong	8
3	Boxing	8
4	Hockey	8
5	Soccer	11

Q4: Nested - Get the name and hours of each venue used in Championship 1

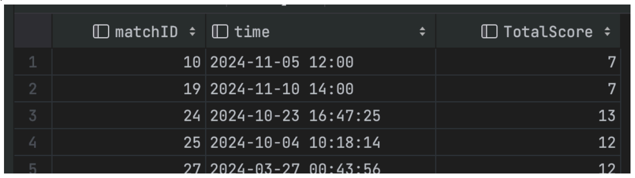
```
select name, hours
from venue
where venueID in (
    select venueID
    from match_gen
    where tournamentID in (
        select tournamentID
        from tournament
        where name = 'Championship 1'
    )
);
```



	name	hours
1	Stadium A	9AM - 9PM
2	Stadium F	10AM - 6PM
3	Field M	7AM - 7PM
4	Court N	9AM - 9PM
5	Gym T	8AM - 8PM

Q5: Correlated - Get all matches with a total score over the average total score

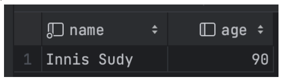
```
select matchID, time, scoreAlpha + scoreBeta as TotalScore
from match_gen
where scoreAlpha + scoreAlpha > (
    select avg(scoreAlpha + scoreBeta)
    from match_gen
);
```



	matchID	time	TotalScore
1	10	2024-11-05 12:00	7
2	19	2024-11-10 14:00	7
3	24	2024-10-23 16:47:25	13
4	25	2024-10-04 10:18:14	12
5	27	2024-03-27 00:43:56	12

Q6: >=ALL/>ANY/Exists/Not Exists - Get the user(s) with the highest age

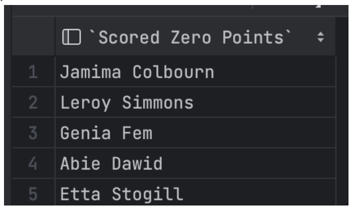
```
select name, age
from user
where age >= ALL (
    select age
    from user
);
```



	name	age
1	Innis Sudy	90

Q7: Union - Get all players that once scored 0 points as player Alpha or Beta

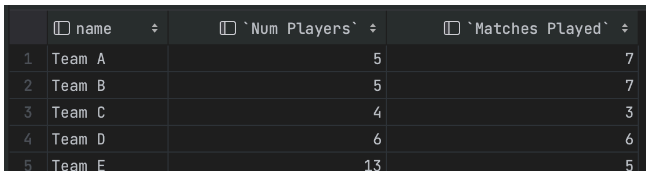
```
select u.name as "Scored Zero Points"
from user u
join player p on u.userID = p.playerID
where p.playerID in (
    select playerAlpha
    from individuals_match
    where matchID in (
        select matchID
        from match_gen
        where scoreAlpha = 0
    )
)
union
select u.name as "Scored Zero Points"
from user u
join player p on u.userID = p.playerID
where p.playerID in (
    select playerBeta
    from individuals_match
    where matchID in (
        select matchID
        from match_gen
        where scoreBeta = 0
    )
);
```



	`Scored Zero Points`
1	Jamima Colbourn
2	Leroy Simmons
3	Genia Fem
4	Abie Dawid
5	Etta Stogill

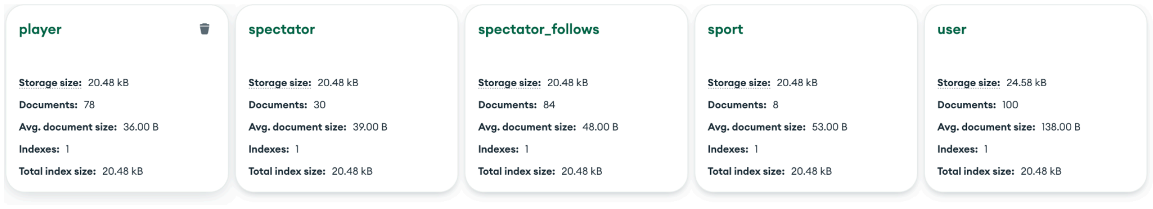
Q8: Sub-queries (Select, From) - Get names, number of players, and # of matches played of all teams

```
select t.name,
	(select count(tp.playerID) from team_player tp where tp.teamID = t.teamID group by
t.teamID) as 'Num Players',
	(select count(*) from teams_match tm where tm.teamBeta = t.teamID or
tm.teamAlpha = t.teamID) as 'Matches Played'
from team t;
```



	name	`Num Players`	`Matches Played`
1	Team A	5	7
2	Team B	5	7
3	Team C	4	3
4	Team D	6	6
5	Team E	13	5

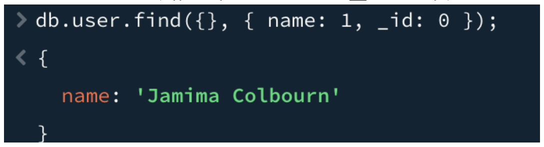
The database was partially implemented in MongoDB. The user, sport, player, spectator, and spectator_follows tables were included, shown below in MongoDB Compass.



	player	spectator	spectator_follows	sport	user
Storage size:	20.48 kB	20.48 kB	20.48 kB	20.48 kB	24.58 kB
Documents:	78	30	84	8	100
Avg. document size:	36.00 B	39.00 B	48.00 B	53.00 B	138.00 B
Indexes:	1	1	1	1	1
Total index size:	20.48 kB	20.48 kB	20.48 kB	20.48 kB	20.48 kB

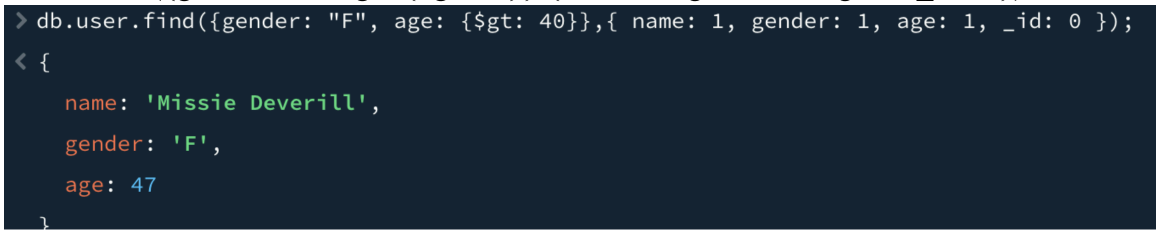
The following three queries were performed:

Simple
db.user.find({}, { name: 1, _id: 0 });



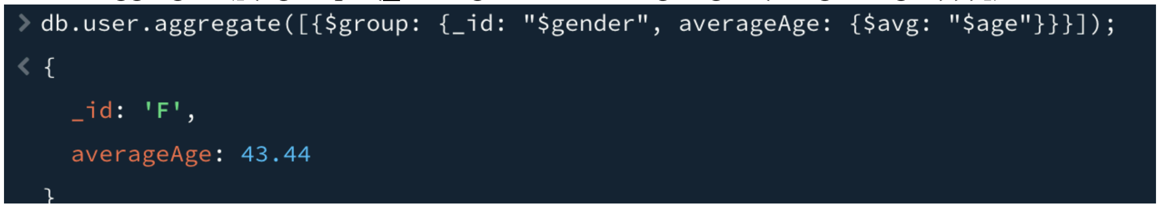
```
> db.user.find({}, { name: 1, _id: 0 });
< {
    name: 'Jamima Colbourn'
  }
```

Multiple Conditions
db.user.find({gender: "F", age: {$gt: 40}},{ name: 1, gender: 1, age: 1, _id: 0 });



```
> db.user.find({gender: "F", age: {$gt: 40}},{ name: 1, gender: 1, age: 1, _id: 0 });
< {
    name: 'Missie Deverill',
    gender: 'F',
    age: 47
  }
```

Aggregate
db.user.aggregate([{$group: {_id: "$gender", averageAge: {$avg: "$age"}}}]);

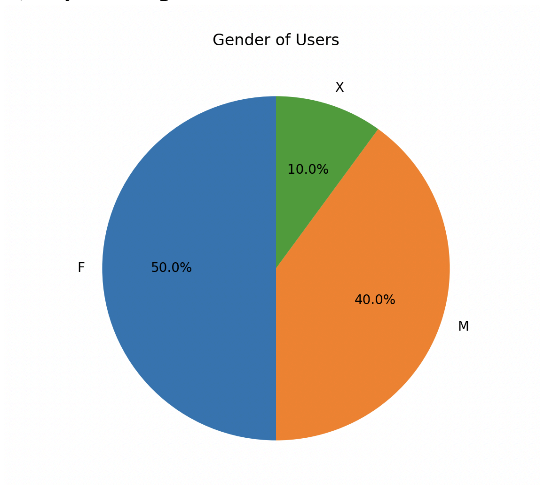


```
> db.user.aggregate([{$group: {_id: "$gender", averageAge: {$avg: "$age"}}}]);
< {
    _id: 'F',
    averageAge: 43.44
  }
```

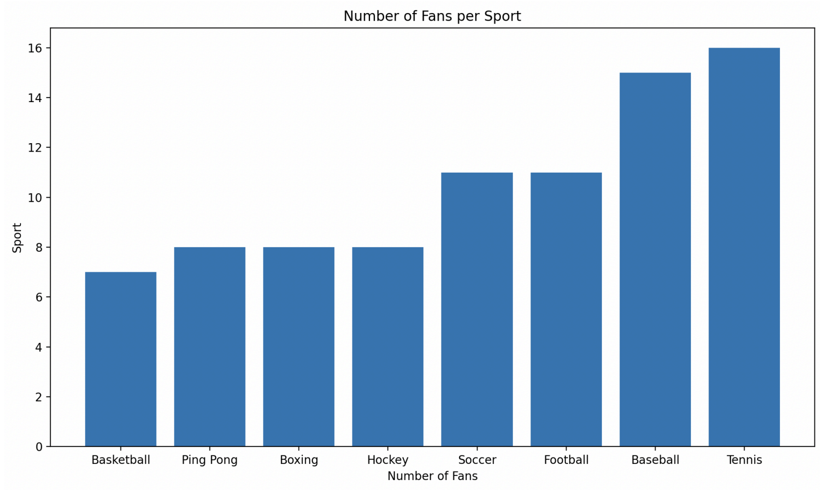
V. Database Access via R or Python

The MySQL Database can be accessed through a python application. The app uses mysql.connector to connect to the database, python's built in input function for user control, and matplotlib for graphing query output. All eight queries shown in Section IV of this report are available to run, with queries 2, 3, and 8 providing graphical outputs.

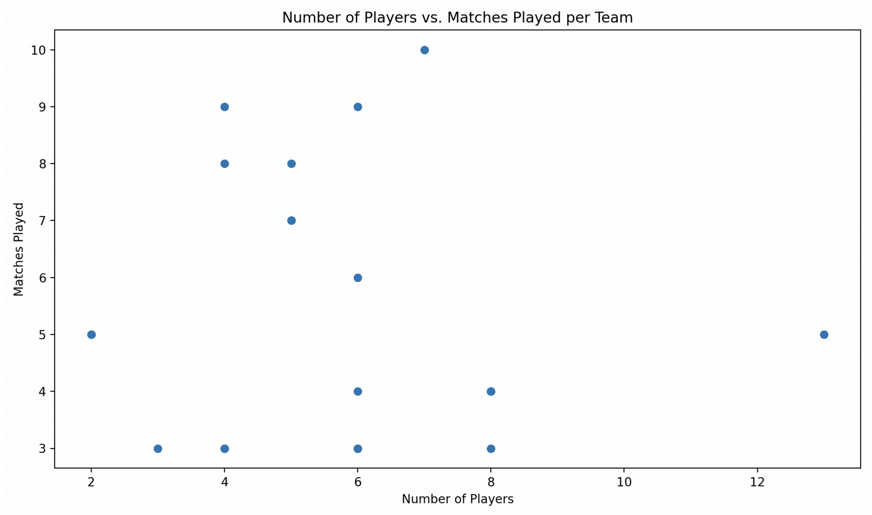
Query 2 Output Pie Chart:



Query 3 Output Bar Chart:



Query 8 Output Scatter Plot:



VII. Summary and recommendation

The CompeteNow MySQL Database is a strong, considered foundation. The platform allows for anyone to start running their own events and tournaments, or just join in on the fun as a player or spectator. The design of CompeteNow gives correct acknowledgement to all the pieces involved in a successful competition event, including matches, venues, teams, spectators, and more, ensuring everything is accounted for, and it is a smooth process for the everyman to get involved with.

There are some notable shortcomings of this implementation. For one, some important connections can be clunky. Performing any connection between player and match or tournament requires going through several tables, which can be detrimental when calculating win-rate or other important stats. Additionally, the sport table only connects to tournament, making it impossible to filter venues or teams by the specific sport. It does allow for more freedom in team and venue management, though that may be seen as a poor tradeoff in a complete implementation. The bigger shortcoming of the platform comes in its simplicity. There are many edge cases the platform does not cover as-is, such as competitive matches like bowling or golf that are not strictly one versus one. If progress is to move forward on this application, features should be slowly rolled out and community feedback should be heavily monitored, such that the platform stays true to its intentions of being the one-stop-shop for creating and participating in the amazing world of competition.