Interface Design

Our project uses a command-line interface (CLI) with an orange basketball-themed colour scheme, since we're working with NBA data. When the program is run, the user is greeted with a welcome message that briefly describes the database and what information is available to them. The welcome message instructs the user to type `help` to see a list of available commands. All query results are printed using dynamically sized columns with orange headers that are underlined for readability.

The help menu is divided into three sections. The first section lists the simple queries, which display entire tables. These simple queries are used as reference tools for getting the input needed for complex queries. For example, if a complex query requires a GameID, the user should run the game table query to find the GameID of interest, then use that value as input for the complex query. The second section lists the complex queries. Finally, the third section lists system-level commands like clearing or repopulating the database, clearing the screen, or exiting the program. Each command in the help menu includes the command the user must enter, along with any parameters, and a short description of what the command does.

The interface is implemented in Python. We created a `DatabaseManager` class, which serves as the central manager for all database operations. It manages the connection by reading the configuration file and runs SQL scripts to create tables and views, as well as clear the entire database. It also orchestrates data loading and querying through its `DataLoader` and `QueryManager` instance variables. The `DataLoader` loads and validates all CSV files into Python using `pandas`, then inserts the data into the SQL Server using prepared statements through `pymssql`. The `QueryManager` class handles all queries sent to SQL Server, again using prepared statements to prevent SQL injection. Finally, we created an `Interface` class that handles user interaction. It displays the welcome message, help menu, and neatly formatted query results. Additionally, it parses user input and sends commands to the `DatabaseManager`.

Diagrams

This section shows diagrams illustrating how the interface will look.

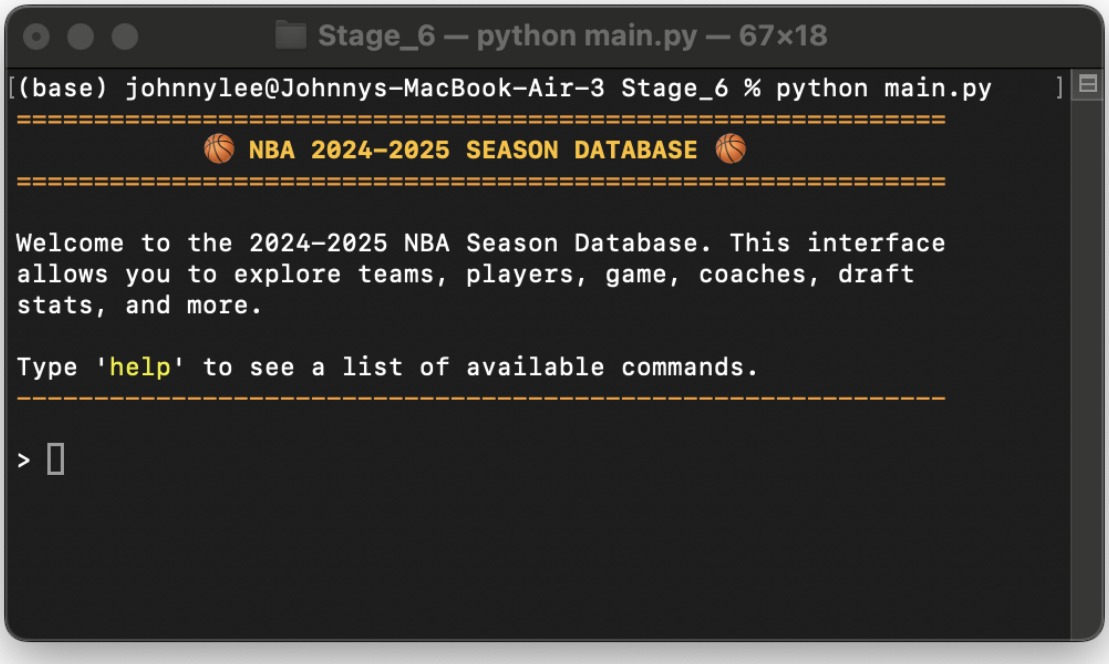


Figure 1: Welcome Message Screenshot

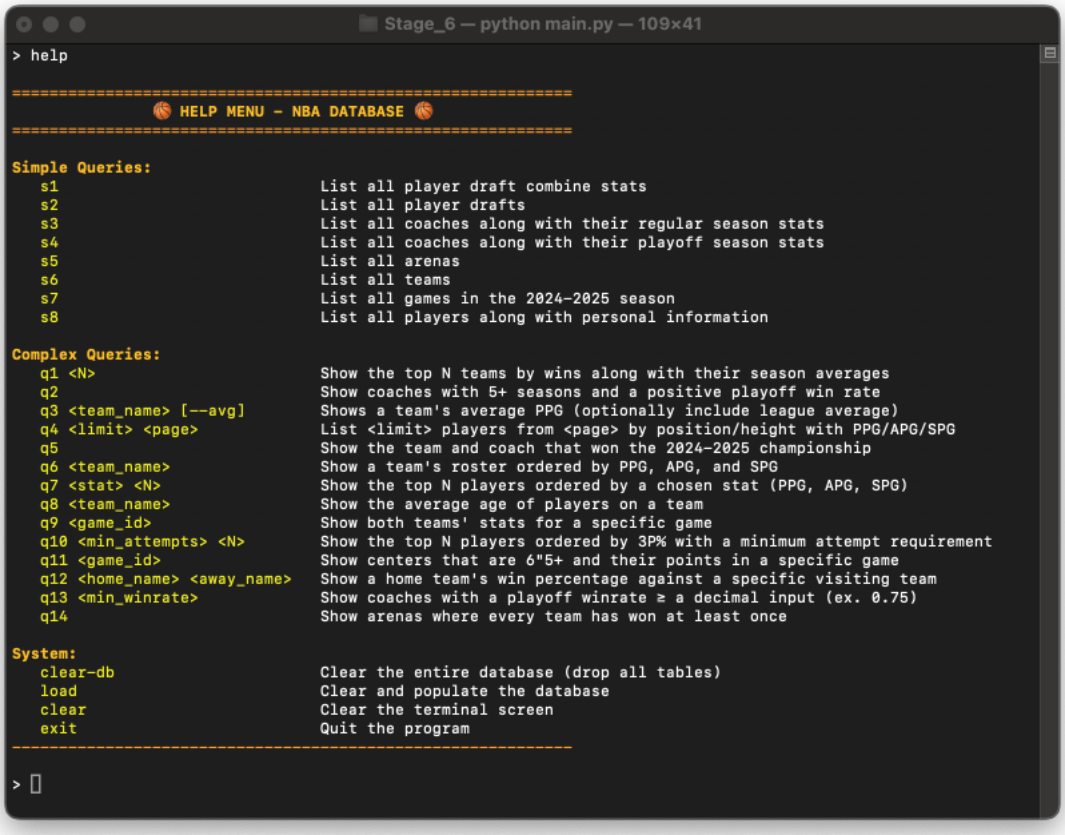


Figure 2: Help Menu Screenshot

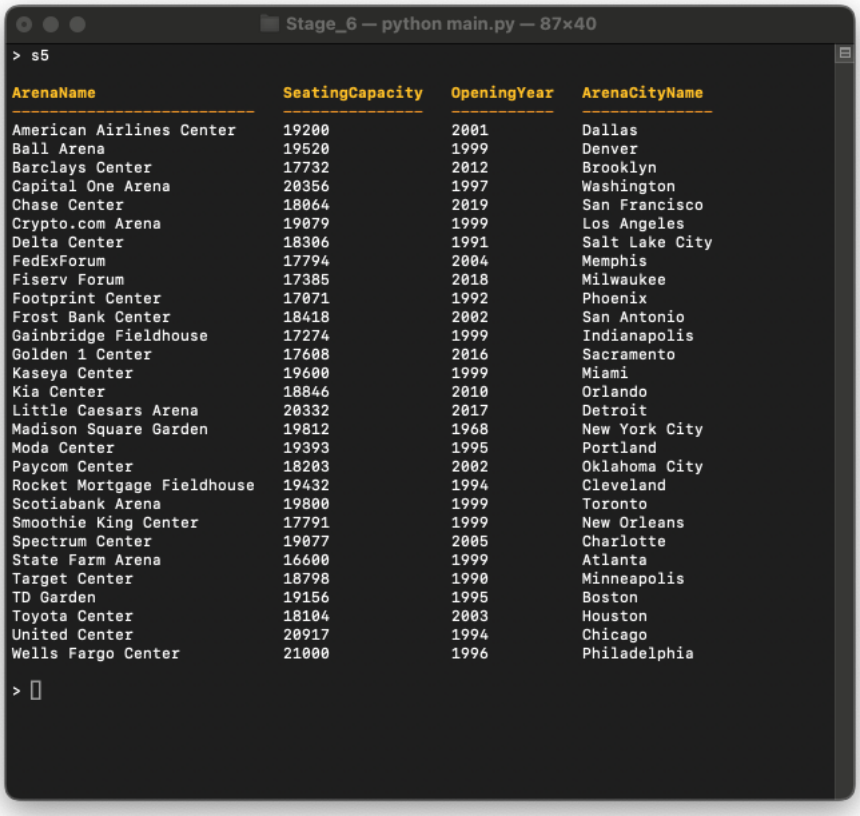
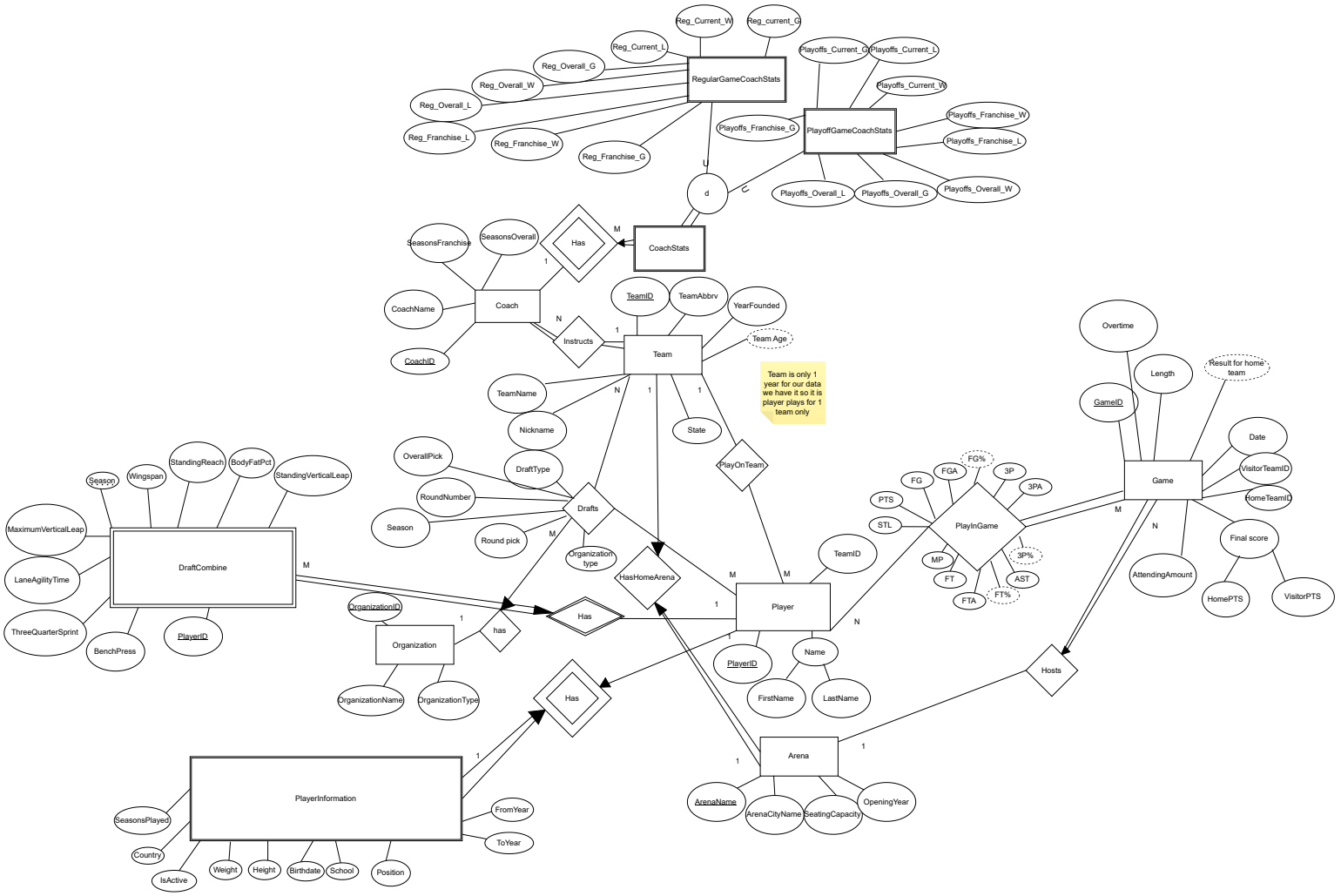


Figure 3: Query Results Screenshot


Reg_Current_W
Reg_current_G
Reg_Current_L
Reg_Overall_G
Reg_Overall_W
Reg_Overall_L
Reg_Franchise_L
Reg_Franchise_W
Reg_Franchise_G
RegularGameCoachStats
Playoffs_Current_G
Playoffs_Current_L
Playoffs_Current_W
Playoffs_Franchise_G
PlayoffGameCoachStats
Playoffs_Franchise_W
Playoffs_Franchise_L
Playoffs_Overall_L
Playoffs_Overall_G
Playoffs_Overall_W
U
d
c
CoachStats
Has
M
1
SeasonsFranchise
SeasonsOverall
Coach
CoachName
CoachID
N
Instructs
1
TeamID
TeamAbbrv
YearFounded
Team Age
Team
Team is only 1 year for our data we have it so it is player plays for 1 team only
TeamName
Nickname
N
1
1
State
PlayOnTeam
OverallPick
DraftType
RoundNumber
Season
Round pick
Drafts
M
Organization type
HasHomeArena
M
M
TeamID
Player
1
N
PlayerID
Name
FirstName
LastName
Season
Wingspan
StandingReach
BodyFatPct
StandingVerticalLeap
MaximumVerticalLeap
LaneAgilityTime
ThreeQuarterSprint
BenchPress
DraftCombine
PlayerID
M
OrganizationID
1
has
Organization
OrganizationName
OrganizationType
Has
Has
1
PlayerInformation
SeasonsPlayed
Country
IsActive
Weight
Height
Birthdate
School
Position
FromYear
ToYear
1
Arena
1
ArenaName
ArenaCityName
SeatingCapacity
OpeningYear
FG%
FGA
3P
FG
3PA
PTS
STL
PlayInGame
MP
3P%
FT
AST
FTA
FT%
Overtime
Length
Result for home team
GameID
Game
Date
VisitorTeamID
HomeTeamID
M
N
Final score
AttendingAmount
HomePTS
VisitorPTS
Hosts