ECE 656 Winter 2023 Project Report

NBA Database

Mareena Francis
m22franc@uwaterloo.ca
20985107

Zeyi Yu
z27yu@uwaterloo.ca
21044384

1. Introduction

1.1. Background

The National Basketball Association (NBA) is a global sports phenomenon, encompassing 30 teams and over 500 registered players. Millions of fans worldwide follow the games, statistics, and individual player performances, reflecting the growing interest in the sport. However, despite the abundance of data generated by the NBA, public databases that collect and organize game or player information are often flawed or incomplete. They tend to focus on a specific field of interest, such as player information or scores only, which limits their usefulness for comprehensive analysis.

In light of these limitations, our project aims to create a unified, informative, and user-friendly NBA database and CLI application that combines data from multiple sources. By integrating 3 separate databases—NBA salary & coach, player info & game logs, and a master database containing game plays, home and away team info, and more—we seek to offer a more comprehensive and accessible tool for analyzing and querying NBA-related data.

1.2. Objective

This consolidated database will contain a wealth of information, including game scores, team plays, player and coach details, team information, and much more. To achieve this, we will begin by cleaning and preprocessing the datasets, removing redundant entries or incorrect information, and normalizing data types and relations. Our next step will be to design and implement an efficient relational database, complete with an Entity-Relationship (ER) model, tables, attributes, and relations that accurately represent the data. By defining primary and foreign keys and implementing necessary constraints, we will ensure data correctness and enable seamless querying.

One of the major challenges we will face during the implementation of this project is the handling of large volumes of data from various sources. This includes dealing with inconsistencies, missing values, and duplicates, which may require extensive data cleaning and preprocessing. Additionally, we will need to consider the scalability and performance of our database design, as new data will continuously be added to the database as new seasons and games unfold.

To address these challenges, we will employ advanced data preprocessing techniques, such as data imputation, outlier detection, and normalization, ensuring that our database remains accurate, consistent, and up-to-date. We will also optimize our database design and query processing, taking advantage of indexing, partitioning, and caching mechanisms to improve performance and minimize response times.

Once the database design is complete, we will populate the tables with data from the selected datasets, combining them into a single, comprehensive source of NBA information spanning the years 1996 to 2021. Our database will provide a historical perspective on the league's development, allowing users to analyze trends and patterns in player performance, team dynamics, and game outcomes.

With the database in place, we will turn our attention to the development of a user-friendly CLI application that allows users to perform various operations on the database, such as fetching game, player, and score information, as well as comparing player performances across different seasons. Our CLI application will be designed with ease of use in mind, featuring a minimal hinting mechanism that guides users through the query process and enables them to create complex, tailored queries with ease.

Beyond the basic querying functionality, our CLI application will also offer advanced analytical capabilities, such as the ability to compare the performance of players and teams across different seasons, identify patterns and trends in game outcomes, and generate customized reports based on user-defined criteria. By providing a versatile and powerful analytical tool, we aim to empower fans, analysts, and other stakeholders with the insights they need to better understand and appreciate the world of professional basketball.

Moreover, we will ensure that our database and CLI application adhere to best practices in data security and privacy. This includes implementing appropriate access controls and authentication mechanisms to prevent unauthorized access and protect the integrity of the data. We will also maintain regular backups and employ disaster recovery strategies to minimize the risk of data loss and ensure the continued availability of our service.

In summary, our project will deliver a comprehensive, reliable, and user-friendly NBA database and CLI application that combines data from multiple sources and provides advanced analytical capabilities. By doing so, we aim to bridge the existing gaps in the public domain and offer a valuable resource for fans, analysts, and other stakeholders interested in the NBA.

1.3. Scope of the project

Key features of our project include:

Integration of multiple datasets (NBA salary & coach, player info & game logs, and a master database containing game plays, home and away team info, etc.) into a single, comprehensive source of NBA information.

Rigorous data cleaning, preprocessing, and normalization to ensure the accuracy and consistency of the data.

Design and implementation of an efficient, scalable, and user-friendly relational database, complete with an Entity-Relationship (ER) model, tables, attributes, and relations.

Development of a versatile and powerful CLI application that allows users to perform various operations on the database, such as fetching game, player, and score information, as well as comparing player performances across different seasons.

Advanced analytical capabilities, including the ability to compare the performance of players and teams across different seasons, identify patterns and trends in game outcomes, and generate customized reports based on user-defined criteria.

Adherence to best practices in data security and privacy, including implementing appropriate access controls and authentication mechanisms, as well as maintaining regular backups and employing disaster recovery strategies.

By successfully executing our project, we anticipate a significant impact on the way NBA data is accessed, analyzed, and utilized. Our comprehensive database and user-friendly CLI application will empower users with the insights they need to better understand and appreciate the world of professional basketball. Additionally, our project may serve as a blueprint for similar initiatives in other sports, paving the way for more accessible, reliable, and informative sports databases in the future.

In conclusion, the NBA Comprehensive Database and CLI Application project aims to revolutionize the way fans, analysts, and other stakeholders interact with NBA data. By consolidating multiple datasets, implementing a robust and efficient relational database, and offering a powerful and user-friendly CLI application, we hope to deliver a valuable resource that promotes a deeper understanding and appreciation of the NBA. As we continue to develop and refine our project, we are excited about the potential it holds for enhancing the world of professional basketball and inspiring similar initiatives in other sports.

2. Dataset Resources

The dataset resources for our NBA Comprehensive Database and CLI Application project are primarily sourced from Kaggle and various other online platforms. By leveraging these rich sources of information, we aim to create a comprehensive and accurate database that covers different aspects of the NBA, including player and coach information, game logs, and salary data.

Kaggle is a well-known platform for data scientists and machine learning practitioners that provides a wealth of datasets, competitions, and learning resources. It is a valuable resource for obtaining high-quality and up-to-date data for our project. Our team has identified and selected several relevant datasets available on Kaggle that pertain to the NBA. These datasets contain essential information on player and coach profiles, game logs, and salary data, which will be used to build our comprehensive NBA database.

In addition to Kaggle, we have also explored other online platforms and websites to gather additional data related to the NBA. These sources include official NBA websites, sports news websites, and public

APIs that provide access to sports data. By collecting data from various sources, we aim to create a more complete and diverse dataset, which will enable us to offer a richer and more insightful analysis of the NBA.

Once we have gathered the data from these sources, we will perform rigorous preprocessing and cleaning to ensure its accuracy and consistency. This process will involve removing redundant entries, correcting any errors, and normalizing data types across the different tables. Furthermore, we will merge the various datasets into a single, unified database, establishing relationships between columns using primary and foreign keys.

In summary, the dataset resources for our NBA Comprehensive Database and CLI Application project are primarily sourced from Kaggle and various other online platforms. By leveraging these rich sources of information and carefully preprocessing and cleaning the data, we aim to create a comprehensive, accurate, and up-to-date database that covers different aspects of the NBA. By regularly updating our database and ensuring the quality and accuracy of our data sources, we will provide our users with a reliable and insightful resource for exploring and analyzing the world of professional basketball.

3. Data cleaning and preprocessing

The CSV files for all the three datasets are downloaded for preprocessing. Based on the project plan, there is one major dataset and a large subset of attributes and data are coming from the major dataset which is NBA Play-by-Play Data 2015-2021. The initial step was to remove the irrelevant attributes and data from the collection and keep only the data that is relevant for the project. Below are the manipulations we performed for each of the datasets both manually and also by using SQL. For achieving this, we divided the CSV files into multiple files with each of the CSN files containing required attributes.

a. Dataset 1 : NBA Play-by-Play Data 2015-2021 [1]

This is the major dataset we are using which contains the NBA data during the period between 2015 and 2021. The whole dataset contains information regarding the Games, Player and Coaches information during the years. The dataset also contains information about various game plays such as Shots, Fauls, Jumpballs, Turnovers, Rebounds, Violations and Free throws. All these data were stored in separate CSV files based on the season and there were a total of 6 CSV files containing the whole data. In the first step, we created separate CSV files for Game, Shots, Turnovers, Jumpballs, Rebounds and Fauls and combined the files into a single CSV file which contains the information from 2015 to 2021. In addition to that, we fetched only the relevant attributes and removed the attributes which had empty values on almost all records.

b. Dataset 2 : NBA data from 1996-2021 [2]

The second dataset we chose contains the NBA information throughout the years of 1996 to 2021.This dataset has unique information such as salaries and coaches which are not there in the

major dataset we selected. Inorder to create a database with all these data, we have picked the two files for Salaries and Coaches to include in the database. All the irrelevant attributes and attributes with empty values in most of the rows were removed.

c. Dataset 3 : NBA Database [3]

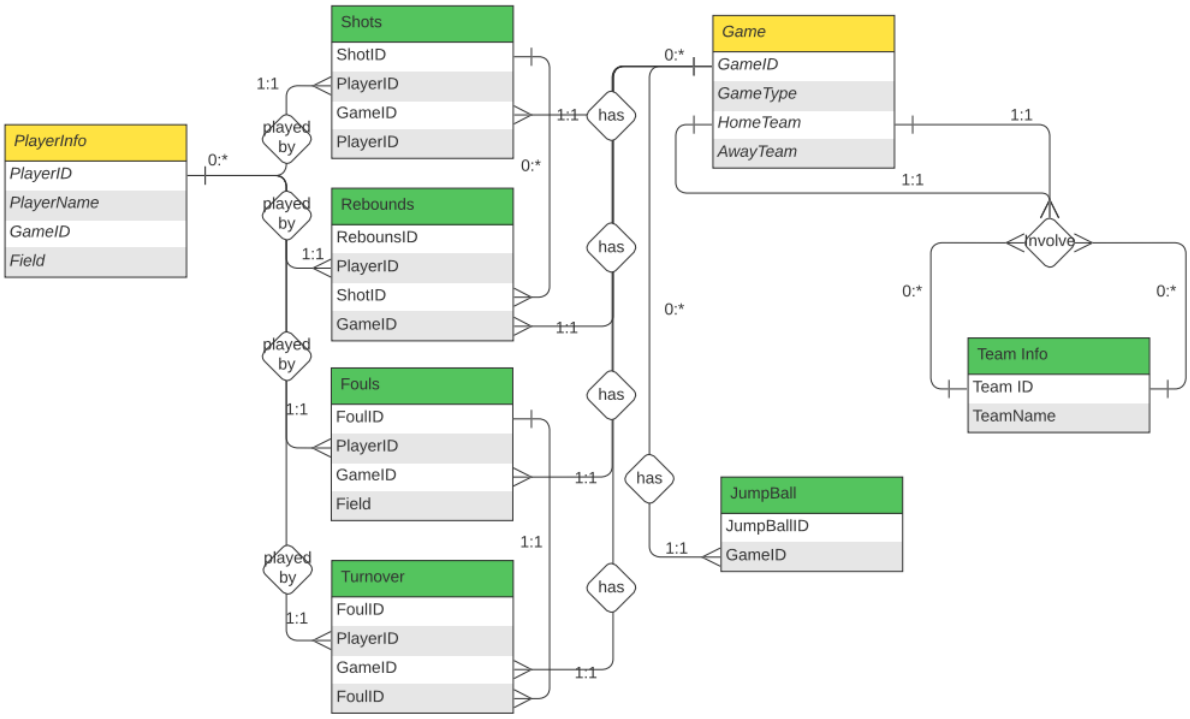
The chosen NBA Database contains the information similar to the major dataset and it also contains additional information related to different Teams and Players which is relevant to the database and client application we implemented. All the unwanted attributes and attributes with empty values in most of the rows were discarded.

After completing the manipulation of CSV files manually, we loaded the data to respective tables for further data cleaning and preprocessing. All the game plays including Shots, Fauls, Jumpballs, Rebounds and Turnovers have a GameID column which is a foreign key to the Game table. We have loaded the GameID columns in all of these tables with corresponding GameID values. Similarly, the Players table has a column named teamID which indicates which team the respective player belongs to. The teamID in the Players table is a foreign key to the teamID in the Teams table. We have loaded the respective teamID values to the Players table from the Teams table.

4. Database Design

4.1. ER Model

In our entity-relationship model, we identified several entities and their respective attributes. Additionally, we discussed merging three NBA databases. To provide a more detailed explanation and emphasize the entity relation part, we will examine relationships, cardinalities, primary and foreign keys, and design tradeoffs in our ER model.



Relationships:

In the ER diagram, there are multiple relationships. The Player table is connected to Shots, Fouls, Rebounds, and TurnOvers tables through a weak 'played by' relationship. This relationship is connected to these tables via a strong 1:1 relation. The Shots, Fouls, Rebounds, and TurnOvers tables are further connected to a 'has block' relationship on a strong basis, connecting to the Game table as a weak 0:* relationship. The JumpBalls table is isolated from the Players table but connected in the same fashion with the Game table. The Teams table is connected to the Games table in a strong relationship, utilizing both HomeTeams and AwayTeams attributes in the Game table.

Cardinalities:

For the 'played by' relationship, a given player could participate in many shots, fouls, rebounds, and turnovers but must be involved in at least one event in a game. This results in a 1:* cardinality on the left side of the relationship. In contrast, every shot, foul, rebound, or turnover must involve at least one player, resulting in a 1:1 cardinality on the right side of the relationship. The 'has block' relationship has a 1:1 relation on the left side since every event (shot, foul, rebound, turnover) must be associated with at least one game, and a 0:* relation on the right side since a game may have zero or more such events.

Primary and Foreign Keys:

The primary keys for the entities in our model are as follows: TeamID for Teams, PlayerID for Players, GameID for Game, ShotID for Shots, FoulID for Fouls, ReboundID for Rebounds, TurnoverID for TurnOvers, and JumpballID for JumpBalls. Foreign keys include GameID in Shots, Fouls, Rebounds, TurnOvers, and JumpBalls tables, connecting them to the Game table. The Players table is referenced by Shooter, Assister, Blocker, FreeThrowShooter, Fouler, Fouled, Rebounder, TurnoverPlayer, JumpballAwayPlayer, and JumpballHomePlayer in other tables. Finally, the Teams table is referenced by AwayTeam and HomeTeam in the Game table.

Design tradeoffs in ER Diagram:

In designing the database, we encountered tradeoffs. Our model prioritizes performance, clear categorization, and usability over storage space. The ER model is designed to handle complex relationships between entities, but synthetic IDs in some tables may lead to possible confusion or increased storage space requirements. Despite these tradeoffs, the ER model maintains the integrity of the original data, while improving categorization and overall performance. The separation of datetime attributes into date and time attributes enhances the user interface's intuitiveness and ensures atomicity.

Overall, our entity-relationship model provides a robust structure for analyzing and storing the NBA data, considering relationships, cardinalities, primary and foreign keys, and design tradeoffs. This comprehensive model allows for better categorization, performance, scalability, and usability, ultimately enabling users to analyze NBA data efficiently and effectively.

4.2. Relational Model

4.2.1. Table creation and data loading

We have a total of 10 tables which contain the Games, Plays , Players , Teams, Coaches and Salary information. After creating the CSV files for all the tables, we have executed the create table scripts with the required attributes and data types. Primary keys and foreign keys are also specified in the script. Below is the script used for creating Shots tables and rest tables are also created using similar scripts. For Shots relation, there are few attributes has empty values in many of the records. We have used the LOAD command in such a way that it will insert the value as NULL for those attributes.

```
create table Shots (ShotID int primary key,
                    GameID int,
                    ShotType varchar(100),
              Shooter varchar(100),
                    Assister varchar(100),
                    Blocker varchar(50),
              ShotOutcome varchar(20),
              foreign key(GameID) references Game(GameID)
              );

LOAD DATA INFILE
'/var/lib/mysql-files/Group07/Shots.csv' ignore INTO TABLE Shots
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
(ShotID,GameID,@ShotType,@Shooter,@Assister,@Blocker,@ShotOutcome)
       set
       ShotType = IF(@ShotType = '' , NULL, @ShotType),
       Shooter = IF(@Shooter = 'NA' , NULL, @Shooter),
     Assister = IF(@Assister = 'NA' , NULL, @Assister),
     Blocker = IF(@Blocker = 'NA' , NULL, @Blocker),
     ShotOutcome = IF(@ShotOutcome = 'NA' , NULL, @ShotOutcome)
IGNORE 1 LINES;
```

4.2.2. Primary keys and Foreign keys

Necessary primary keys and foreign keys were added to the tables. Primary keys are added to all the relations and use the attribute with integer type for the purpose of saving space. GameID table has the GameID attribute as the primary key which uniquely identifies all the records. Other relations also have corresponding primary keys which uniquely identify the records. Added the foreign key constraints in order to maintain data integrity. Below are the various foreign keys added to the relations.

- Shots, Jumpballs, Turnovers, Fauls and Rebounds are the different game play types involved. All these types are associated with a particular Game. We have the Game table with GameID attribute as primary key. All the play types are created with a GameID attribute which is a foreign key referring GameID in Game relation.
- Players relation is created with all the information including PlayerID, name, city and other relevant details. In addition to this, teamID attribute is added to the relation which tells which team the player belongs to. This teamID attribute in Players relation refers to the primary key attribute teamID in Teams relation.

4.2.3. Index creation and performance

In this section, we identified what indexes should be created in order to enable efficient querying. Initially we performed a few query operations without adding any additional indexes. The only indexes existing were the primary and foreign key indexes. Below are the execution times for the queries without having explicit indexes created.

```
(actual time=2.218..2.219 rows=1 loops=1)
```

```
(actual time=0.340..0.340 rows=1 loops=1)
```

In order to improve the performance, we have created indexes on relations which are involved in complex queries. Below are the execution times for the queries after adding indexes. It shows that there is a decrease in the execution times when comparing the respective execution times before execution which is shown above.

```
(actual time=1.826..1.827 rows=1 loops=1)
```

```
(actual time=0.287..0.287 rows=1 loops=1)
```

5. Client application

The primary objective of our project is to provide users with an efficient and user-friendly platform to interact with NBA data, specifically focusing on querying and modifying the data in a way that caters to customers in the domain. To achieve this, we have designed a client application with a Python GUI that offers a range of functionalities, including access to team, player, game, play, and coach information, the ability to append new players to the database, and predefined queries for retrieving relevant information.

Ideal Client Requirements

An ideal client for our project should meet the following requirements:

User Interface: The client should have an intuitive and visually appealing user interface that enables users to navigate and interact with the data effortlessly.

Query Functionality: The client should offer a wide range of query options, allowing users to retrieve specific data points and generate custom reports based on their requirements.

Data Modification: The client should enable users to modify the data, such as adding new players, updating player information, and adjusting team statistics.

Data Validation: The client should include robust error handling and data validation features to ensure the integrity of the data and minimize the risk of incorrect or incomplete data entry.

Performance: The client should be capable of handling large datasets and multiple simultaneous users without experiencing performance issues or crashes.

Actual Client Proposed

To address the ideal client requirements, we have proposed a client application featuring a Python GUI with clickable buttons that trigger specific functions to access or modify the data. The following functions are available to users:

- ➢ Retrieve top 5 highest or lowest salaries for a specified year.
- ➢ Calculate shot percentages, total shots, and rebounds for a specified season.
- ➢ Obtain information on nationalities, fouls, coaches, games played, and turnovers.
- ➢ Access detailed data on rebounds and turnover-related statistics.

In addition to the above functions, the client allows users to interact with the data in a more customized manner, offering flexibility in querying and modifying the data to suit their specific needs. The client application also includes error handling and data validation features to maintain data integrity and minimize the risk of data corruption.

Actual Client Implemented

The implemented client consists of the Python GUI with clickable buttons to execute the predefined queries mentioned in the proposal. Upon clicking a button, the GUI returns the requested information, allowing users to quickly access essential data.

The GUI is designed with simplicity and ease of use in mind, providing users with a clear and organized layout that enables them to interact with the data effortlessly. The client application also supports data modification, allowing users to add new players, update player information, and adjust team statistics as needed.

Test Cases Necessary to Validate the Client

To ensure the client's reliability and accuracy, several test cases should be conducted:

Functionality Testing: Test each function by executing the corresponding button and verifying that the returned data is correct and relevant. This includes testing the query options, data modification features, and error handling capabilities.

Performance Testing: Check the client's ability to handle large datasets and ensure that it does not crash or freeze. Additionally, evaluate the client's performance when multiple users access the application simultaneously, ensuring that it can handle the increased load without compromising its responsiveness.

Data Validation Testing: Test the client's error handling capabilities when the input data is incomplete or contains errors. Verify that the application provides appropriate error messages and prevents users from submitting incorrect or incomplete data.

Security Testing: Evaluate the client's security measures, such as data encryption, user authentication, and access control, to ensure that the application protects the data from unauthorized access or tampering.

Client Functions and Features

In this section, we will discuss the various functions and features of the client application in detail.

5.1 Query Functions

The client application provides a range of query functions that allow users to access specific data points and generate custom reports based on their requirements. The query functions include:

1. Retrieve top 5 highest salaries for a specified year:
   Example: Top 5 highest salaries in 1996
   Table: Salaries
   Year: 1996
   Attribute: Salary

   Query:
   SELECT * FROM salaries where SeasonStartyear=1996 ORDER BY salary ASC LIMIT 5;

2. Retrieve top 5 lowest salaries for a specified year:
   Example: Top 5 lowest salaries in 1996
   Table: Salaries
   Year: 1996
   Attribute: Salary

   Query:
   SELECT * FROM salaries where SeasonStartyear=1996 ORDER BY salary DESC LIMIT 5;

3. Retrieve top 5 highest salaries for other specified years (2001, 2006, 2011, 2016):

   Queries:

   SELECT * FROM salaries where SeasonStartyear=2001 ORDER BY salary DESC LIMIT 5;
   SELECT * FROM salaries where SeasonStartyear=2006 ORDER BY salary DESC LIMIT 5;
   SELECT * FROM salaries where SeasonStartyear=2011 ORDER BY salary DESC LIMIT 5;
   SELECT * FROM salaries where SeasonStartyear=2016 ORDER BY salary DESC LIMIT 5;

4. Calculate the total number of rebounds made
   Table: Rebounds
   Attribute: ReboundType

   Query: Select count(*) as Rebounds from Rebounds where ReboundType='defensive' or ReboundType='offensive';

5. Find the total number of distinct nationalities of players in the history of the NBA
   Table: Players

Attribute: Country

Query: Select count(distinct(country)) as Nationalities from Players;

6. Find the player who played the dirtiest:
   Table: Fouls
   Attribute: Fouler

   Query: SELECT my_column, COUNT(*) as count FROM my_table GROUP BY my_column ORDER BY count DESC LIMIT 1

7. Find the player who had the most fouls:
   Table: Fouls
   Attribute: Fouled

   Query: SELECT Fouled, COUNT(*) as count FROM Fouls where Fouled !=" GROUP BY Fouled ORDER BY count DESC LIMIT 1;

8. Find the most occurred type of foul:
   Table: Fouls
   Attribute: FoulType

   Query: SELECT FoulType, COUNT(*) as count FROM Fouls where FoulType !=" GROUP BY FoulType ORDER BY count DESC LIMIT 1;

9. Find the coach who worked the longest:
   Table: Coaches
   Attribute: CoachName

   Query: SELECT CoachName FROM Coaches GROUP BY CoachName ORDER BY count(*) DESC LIMIT 1;

10. Calculate the number of games the Los Angeles Lakers played from 1996 to 2020:
    Table: Game

    Query: Select count(*) as No_of_Games from Game where HomeTeam='Los Angeles Lakers' or AwayTeam='Los Angeles Lakers';

11. Find the player who got the most rebounds:
    Table: Rebounds
    Attribute: Rebounder

    Query: select Rebounder, count(*) as count from Rebounds where Rebounder != " and Rebounder != 'Team' group by Rebounder order by count DESC limit 1;

12. Find the player who got the most offensive rebounds:

Table: Rebounds
Attribute: Rebounder
Condition: Rebound Type = offensive

Query: Select Rebounder, count(*) as count from Rebounds where ReboundType='defensive' and Rebounder != 'Team' group by Rebounder order by count DESC limit 1;

13. Find the player who got the most defensive rebounds:
Table: Rebounds
Attribute: Rebounder

Query : Select Rebounder, count(*) as count from Rebounds where ReboundType='defensive' and Rebounder != 'Team' group by Rebounder order by count DESC limit 1;

14. Find the top reason for turnovers:
Table: Turnovers
Attribute: TurnoverType

Query: SELECT TurnoverType, COUNT(*) as count FROM Turnovers where TurnoverType !='' GROUP BY TurnoverType ORDER BY count DESC LIMIT 5;

15. Find the top cause of turnovers:
Table: Turnovers
Attribute: TurnoverCause

Query: SELECT TurnoverCause, COUNT(*) as count FROM Turnovers where TurnoverCause !='' GROUP BY TurnoverCause ORDER BY count DESC LIMIT 5;

16. Find the top 5 players with the most turnovers:
Table: Turnovers
Attribute: TurnoverPlayer

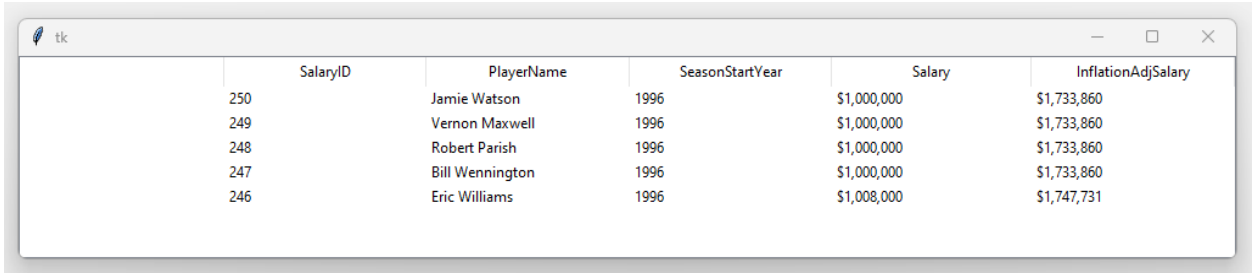
Query: SELECT TurnoverPlayer, COUNT(*) as count FROM Turnovers GROUP BY TurnoverPlayer ORDER BY count DESC LIMIT 5;

17. Find the top 5 players who caused the most turnovers:
Table: Turnovers
Attribute: TurnoverCauser

Query: SELECT TurnoverCauser, COUNT(*) as count FROM Turnovers GROUP BY TurnoverCauser ORDER BY count DESC LIMIT 5

These query functions enable users to access a wide range of data points and generate insightful reports based on their specific requirements. The client application can perform these queries efficiently and present the data in a user-friendly format.

5.2. Application User Interface



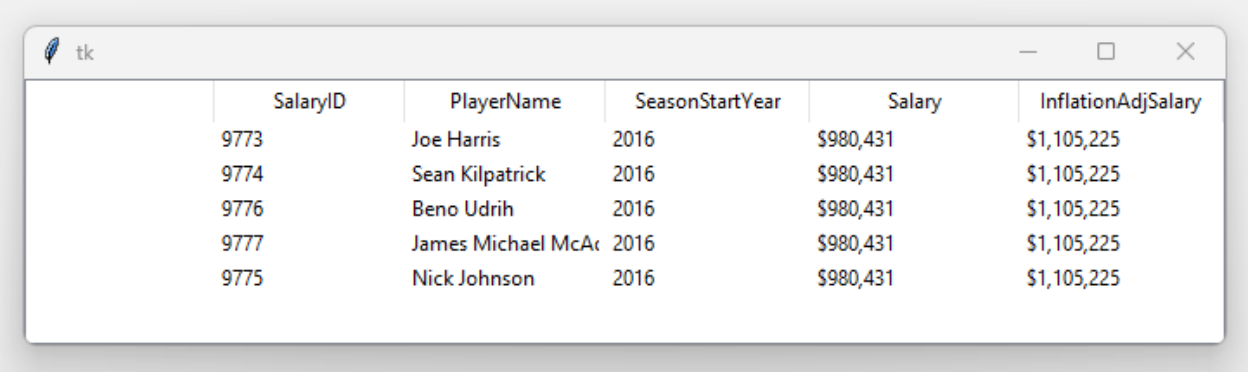
	SalaryID	PlayerName	SeasonStartYear	Salary	InflationAdjSalary
	250	Jamie Watson	1996	$1,000,000	$1,733,860
	249	Vernon Maxwell	1996	$1,000,000	$1,733,860
	248	Robert Parish	1996	$1,000,000	$1,733,860
	247	Bill Wennington	1996	$1,000,000	$1,733,860
	246	Eric Williams	1996	$1,008,000	$1,747,731

5.3. Test plan

Below are a few test cases we used for validating the results:

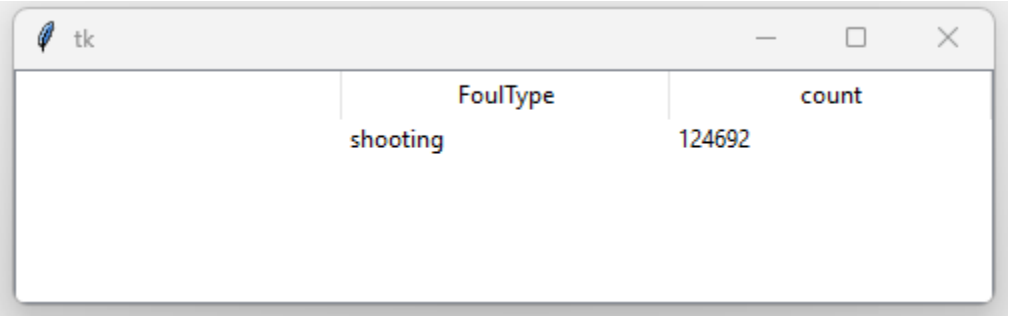
Test case 1 :

Step 1 - Run the user application
Step 2 - Click on the button ‘Top 5 highest Salary in year 2016’
Step 3 - Below output should be displayed



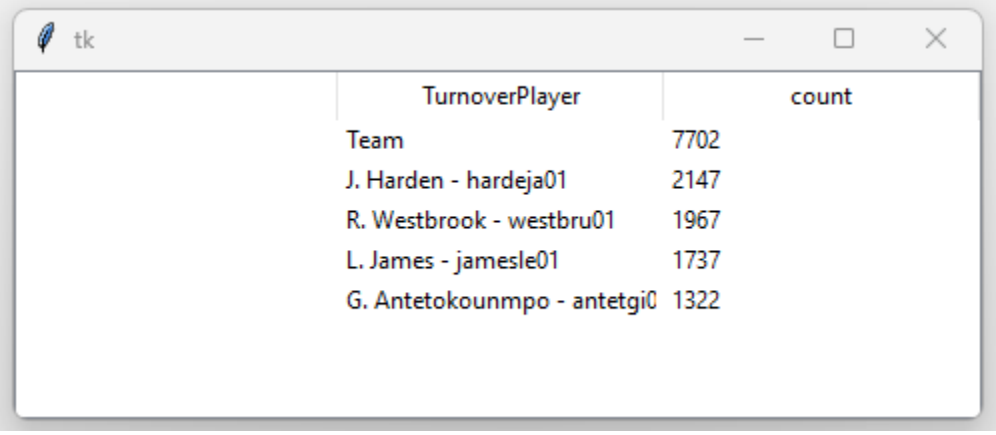
Test case 2 :

Step 1 - Run the user application
Step 2 - Click on the button 'The most occurred type of foul'
Step 3 - Below output should be displayed



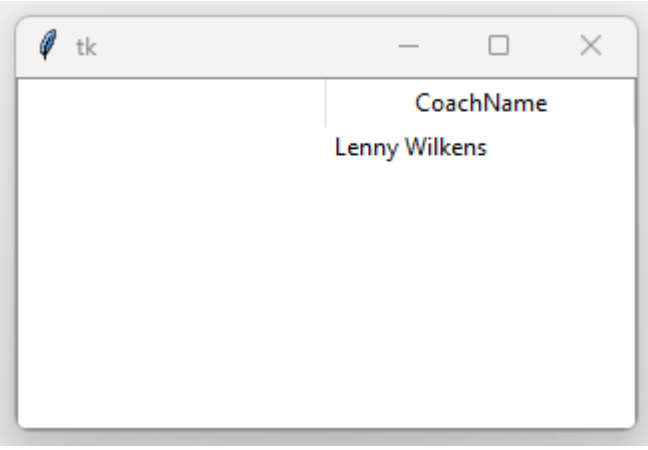
Test case 3 :

Step 1 - Run the user application
Step 2 - Click on the button 'Top 5 turnover player
Step 3 - Below output should be displayed



Test case 4 :

Step 1 - Run the user application
Step 2 - Click on the button ‘Coach who worked the longest’
Step 3 - Below output should be displayed



6. Data Mining

The NBA database has attributes associated with Shots in Game which includes type of shot and shot outcomes which has two possibilities: make or miss. For the data mining part, we are trying to predict the shot out come to be make or miss using Linear Regression Classifier.

i. Loading data

In the initial step, we have created the CSV file with the required attributes for performing Logistic Regression and simplified the columns by providing the column names to pandas readr.csv() function

ii. Selecting feature and Splitting Data

We have divided the columns into two sets: independent variables and dependent variables where the ShotOutCome is the target variable and other attributes such as GameType, Quarter, SecondsLeft, Shot distance, and ShotType as the feature variables. To access the model performance, we have divided the dataset into a training set and test set. The Shots dataset is split into two parts with a ratio of 75:25 where 75% of data will be used for model training and 25% for model testing.

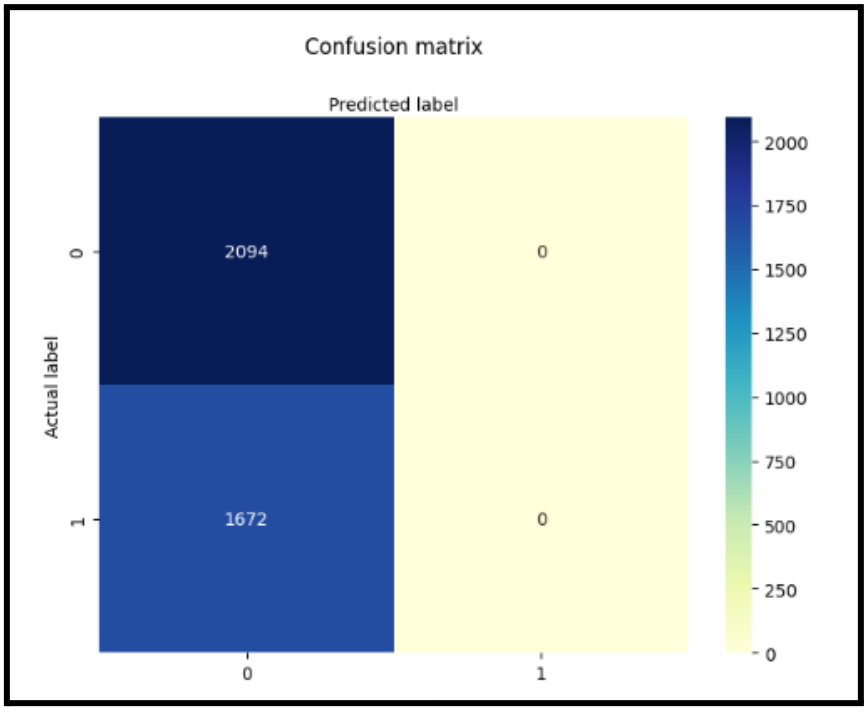
iii. Model development and prediction

We used the Logistic Regression module and created a Logistic Regression CLassifier object using the Logistic Regression function. Then we fit our data model on the training set and performed prediction on the test set.

iv. Model validation using Confusion Matrix

Confusion matrix is used for evaluating the performance of a classification data model which shows the prediction including the class-wise summed up results of the number of correct and incorrect predictions. Below are the investigation results in the form of confusion matrix and a visualized chart.

	precision	recall	f1-score	support
miss	0.56	1.00	0.71	2094
make	0.00	0.00	0.00	1672
accuracy			0.56	3766
macro avg	0.28	0.50	0.36	3766
weighted avg	0.31	0.56	0.40	3766



The results show an accuracy of 56% with which the model is able to predict whether a shot outcome is a make or a miss.

7. Conclusion

In conclusion, this project has successfully designed a database for the NBA dataset and implemented a client application that performs various operations on the database. By carefully designing the ER model and the relational model, we were able to take into account data analysis, implementation procedures, and the possible operations that can be performed by the client using the application. Data integrity was also considered during the data loading process, and suitable constraints were put in place to ensure its preservation.

The development of this comprehensive database solution for NBA data has potential for numerous applications, including analysis of team and player performance, prediction of game outcomes, and the development of new strategies. It could also contribute to enhancing fan engagement through the creation of interactive and informative platforms. The addition of indexes to the necessary attributes greatly improved the performance of complex join operations, demonstrating the importance of well-designed indexes in efficient query execution.

As we have also applied data mining towards the Shots data set using Logistic Regression. We used the Shots for training the model and used a fraction of the data as a test set. The model we created was able to predict whether the shot outcome is a make or a miss with an accuracy of 56%.

In the future, this database and its accompanying client application could be expanded to incorporate additional NBA datasets and new attributes, enabling even more in-depth analysis and insights into the world of professional basketball. As the NBA continues to grow in popularity and complexity, the need for robust and efficient data management solutions like the one developed in this project will only increase. By providing a solid foundation for future development and expansion, this project has demonstrated the potential to significantly contribute to the ongoing evolution of basketball analytics and the NBA as a whole.

8. References

1. NBA Play-by-Play Data 2015-2021, https://www.kaggle.com/datasets/schmadam97/nba-playbyplay-data-20182019?select=NBA_PBP_2015-16.csv
2. NBA data from 1996-2021, https://www.kaggle.com/datasets/patrickhallila1994/nba-data-from-basketball-reference?select=salaries.csv
3. NBA Database, https://www.kaggle.com/datasets/wyattowalsh/basketball