Graph and Modern Databases Project Report

Author: ILKER OZTURK

Student Number: 001187380

Date: 30 March 2022

Authorisation: Module Leaders

Distribution: All students

VM Number: COMP1835-nosql-135

COMP1835-graphdb135

School of Computing and Mathematical Sciences

Table of Contents

1. PART A – Peer Review 5
1.1 SUMMARY 5
1.2 Findings from the paper and their significance 5
1.3 Strengths and Weakness of the paper 6
2.Major issues 6
Presentation related problems 6
2.2 Ethical issues 6
3.Minor issues 7
3.1 Clarity of paper 7
3.2 About the Referencing and citation 7
3.3 Numerical, factual and unit errors 7
3.4 Appropriateness and sufficiency of tables and figures 7
4.Critical analysis of presentation and style 7
5.Critical analysis on referencing 8
6. Recommendation from my perspective 8
2. PART B - Implementation 9
1.JSON Implementation 9
1.1 JSON Definition 9
1.2 JSON Data Structure of My Implementation 9
2. Redis Definition and Coursework Implementation 10
2.1 Redis Definition 10
2.2 Coursework implementation 10
3. Cassandra definition and Coursework Implementation 20
3.1 Coursework Implementation 20
4. MongoDB definition and Coursework Implementation 23
4.1 Coursework Implementation 23
5. NEO4J Definition and Coursework Implementation 32
5.1 Coursework Implementation 32
Appendices 43
Appendix A – JSON Schema 43
Appendix B – JSON Data 45
Bibliography 50

Figure 1:JSON VALIDATION MESSAGE 10
Figure 2:first4 players 11
Figure 3:second 3 player 12
Figure 4:Last 3 players 13
Figure 5:special command 14
Figure 6:Data Structure of Redis 14
Figure 7:RETRIEVE TRANSFER VALUE OF PLAYER1 14
Figure 8:Delete birthday field 15
Figure 9:Length of player4 15
Figure 10:Values of player5 15
Figure 11:Change the values of player3 16
Figure 12:get specific fields from player3 16
Figure 13:get the key values of player2 16
Figure 14:player22 creation 16
Figure 15:Player21 creation 17
Figure 16: zunionStore command 17
Figure 17:Unified set of 2 players 17
Figure 18:Total Number of fields 17
Figure 19:player24 18
Figure 20:PLAYER25 18
Figure 21:Difference between two sets 18
Figure 22:creation of player26 18
Figure 23:creation of player27 18
Figure 24:creation of player28 19
Figure 25:third index of player28 19
Figure 26:length of player26 19
Figure 27:first 10 fields of player27 19
Figure 28:Cassandra data 20
Figure 29:More than 2 kids 20
Figure 30:query2 cassandra 21
Figure 31:query3 CASSANDRA 21
Figure 32:query4 cassandra 21
Figure 33:query5 cassandra 21
Figure 34:query6 cassandra 22
Figure 35:query7 cassandra 22
Figure 36:Query8 cassandra 22
Figure 37:query9 cassandra 22
Figure 38:query10 cassandra 23
Figure 39:Data Structure MongoDB 24
Figure 40:Query mongo db 24
Figure 41:query2 mongo dB 25
Figure 42:query3 mongo db 26
Figure 43:query4 mongo db 27
Figure 44:quey5 mongo db 28
Figure 45:query6 mongo db 29
Figure 46:Query7 mongo db 29
Figure 47:query8 mongo db 30
Figure 48:query9 mongo db 31

Figure 49:query10 mongo db 32
Figure 50:NEO4J data structure 33
Figure 51:Query 1 neo4j 34
Figure 52:Query2 NEO4J 35
Figure 53:Query3 neo4j 36
Figure 54:query4 neo4j 37
Figure 55:query5 neo4j 38
Figure 56:query6 neo4j 38
Figure 57:query7 neo4j 39
Figure 58:query8 neo4j 40
Figure 59:query9 neo4j 41
Figure 60:query10 neo4j 42

1. PART A – Peer Review

In this section, there is a peer review of the academic document comparing SQL and NOSQL databases. Peer Review consists of 5 parts.

- Summary
- Major Issues
- Presentation and style comments
- Reference comments

1.1 SUMMARY

First, I kindly recommend that this article should not be published because it is very lacking in terms of academic language, needs more examples, explanations are not very detailed, and cannot express its views properly with academic citations. Apart from the abstract as well as the reference parts, there are five major key points which are related to the particular paper based on the comparison between SQL as well as NoSQL. These five points are included within the terms of the introduction, relational databases along with the strengths and weaknesses. Furthermore, other points are based upon the elaborated description of the NoSQL databases and the paper has been concluded within the conclusion part. The abstract part of the paper has been discussed with the summary of the whole paper which mainly focuses on the comparison between the SQL as well as the NoSQL databases. Abstract has majorly been focused on the relational database along with its importance for creating and maintaining the data elements. Apart from that, the abstract part has also been focused on the huge flexibility of the different data element-based categories. Furthermore, the abstract of the paper is related to the advantages as well as the disadvantages of the NoSQL as well as the SQL. Moreover, within the introduction part, there are many aspects and significance related to the data. At first the introduction part has introduced the definition of data and then it signifies that data is the most important asset within every possible commercial venturing area. Along with this, taking off an ecommerce website can be effective for identifying the importance and databases. Moreover, it would observe the importance of the arrangement of different data values within the systematic way. The main idea of the chosen paper is related to the data transaction process which keeps different records of data by updating the values of data. After introducing the NoSQL data, it can be said that keeping record of data is low time consuming and it needs advanced techniques for storing the data.

The next point is based on the relational database. Moreover, the relational database, an industrial standard, is a part of databases that can store the data and it is also able to access the data points. Furthermore, relational databases are related to the relational model. The relational database part is also related to the core concepts of the database structure. The relational database observes the core concepts of the normalisation which avoids the repetition of databases. Furthermore, it can also be said that there are some types of referencing which can be evaluated by the implementation of relational databases. Apart from that, some findings as well as strengths and weaknesses would be illustrated within the following parts.

1.2 Findings from the paper and their significance

Findings are related to each different five points. The findings from the introduction part is there are some differences between SQL as well as NoSQL. Another finding from the introduction part is related to the data which is the most important part within the recent days. This is a simple example related to the ecommerce website. From the help of databases, people can be tracked also. It also finds the databases related to the data transaction. Finally, it has also been evaluated the core differences between SQL and NoSQL.

1.3 Strengths and Weakness of the paper

1.3.1 Strengths of the paper

- The structure of the paper is good and well explained
- The paper also contains an abstract part which is really appreciable
- The paper has a glimpse of the referencing style of the Greenwich University of Harvard which is a positive part of the paper. This is because it would avoid academic misconduct.
- The paper is full of information which can be considered as a good writing approach
- Total paper has been written within an academic language which helps to carry more marks.

1.3.2 Weakness of the paper

- There are no tables and figures within the whole paper demonstrating the key aspects
- There is not any proper intext citation as per the reference list. The writer needs to be focused on the last name of the author and the published year instead of writing only the serial number of the references.
- The formatting of the paper is not proper and attractive
- The paper did not follow the Harvard formatting style
- The paper has a lack of description related to the SQL where the paper has contained the brief and compacted description of the NoSQL.

2.Major issues

Presentation related problems

Based on the paper, related to the comparative study of SQL as well as NoSQL, it has been recognised that the presentation related issues have been recognised as the major problem of the paper. The presentation problems of the paper are related to the lack of evidence of any figures and tables. Moreover, the presentation of the paper is not attractive and not also well formatted. Furthermore, the presentation problems are related to the improper formatting of Harvard. Finally, it has thereby been established that the presentation related problem is based upon the improper intext citation.

2.2 Ethical issues

Apart from the presentation related problems, there are also the ethical issues which have hampered the marks. The root causes related to the poor acquiring of marks is basically based upon the evidence of academic misconduct. These are related to the use of poor formatting style and the poor intext citation. Finally, it has thereby been recognised that the ethical issues are needed to be mitigated for acquiring the highest number and rank.

3.Minor issues

3.1 Clarity of paper

The paper has been written within the academic as well as in terms of formal languages. Furthermore, there is no instance of writing repetitive words which can make the paper dull as well as monotonous. Moreover, the writing style of the paper is straightforward as well as linked to the point. The paper is also plagiarism free as well as written within a simple language. This is because there is enough clarity within the chosen paper related to the comparison between the SQL and NoSQL.

3.2 About the Referencing and citation

The referencing style is related to the Greenwich University is Harvard style. Within the chosen paper, there is a total number of 12 references within the list of References. It has also been recognised that there are text citations after each 100 to 120 words, which carries the clarity of the paper. As well as it also ensures the enrichment of the chosen paper. Conclusively, it has thereby been established that the citation can be better which may assure the better enrichment of the paper.

3.3 Numerical, factual and unit errors

This chosen paper is the amalgamation of the different types of information. However, there is not any numerical values or quantitative data within the paper. Though the paper only contains the qualitative information within the paper. This is because there is no chance of the errors in terms of numerical, factual and the unit errors.

3.4 Appropriateness and sufficiency of tables and figures

There are not any glimpses which are related to the tables or figures within the chosen paper. If the researcher adds some tables and figures, it would be more informative as well as attractive too.

4.Critical analysis of presentation and style

Based on the study, it can be concluded that the review has been presented in an organised and detailed manner. This review gives importance to every aspect that is related to effectively and efficiency of the study. The structure of the review gives a detailed analysis of the concept of relational database, but not many details are given about the strengths and weaknesses of the given topic as in other studies. This lack of information can make the readers lose interest in carrying out further study. As opined by Walker (2005), due to the lack of information about a study, the researchers are unable to carry out any further research on the topic. This not only disrupts the existing study, but also hampers the spread of knowledge regarding the research area. In this review, a descriptive overview regarding relational databases has been given, unlike other studies that are conducted on the given topic. This helps an individual in getting a conceptual knowledge about the topic that will help in conducting further research regarding the topic. According to Crooks and Alibali (2014), conceptual knowledge about a given topic helps the researchers in getting a correct viewpoint regarding the subject matter that helps in carrying out the study in an effective and efficient manner. Through this review, the researchers get a concrete overview regarding every aspect of the relational database, which helps in the simplification of the research work. Through the conceptual knowledge gained about the relational database, the researcher will additionally get a correct overview regarding other aspects related to the study.

5.Critical analysis on referencing

The referencing style followed in this review gives the viewers a glimpse of Harvard, which gives the study a formal look as compared to other researchers done on the same topic. The reference style gives the researchers an overview regarding the look of a formal study conducted on a given topic. This study contains wide ranges of references taken from journals, news articles and other internet sources. As opined by Feinstein (2008), adequate and accurate referencing makes the journal more informative and organised in the eyes of the viewers. This helps the researchers in not only getting an updated knowledge regarding the study but also aids in smoothing out the flow of research work. The referencing style followed in the research work gives the readers an example about the structure that is needed to be followed for preparing a formal study on a given topic. In the views of Feinstein (2008), adequate and updated referencing in a study helps the readers in getting detailed and correct knowledge regarding aspects related to the given topic. This eases out the flow of work that is to be conducted on the given topic and helps to prepare the research in a timely manner. This further helps an individual in carrying an in-depth and organised research on the given topic by following adequate reference to the previous studies conducted on the same subject matter in a timely manner. Therefore, adequate and updated referencing is very critical for the acceptance of a research work in the eyes of the researchers.

6. Recommendation from my perspective

- It can be recommended that some informative and subject linked tables and figures could be added.
- Moreover, it has been recommended that the intext citation can be modified within an improved way.
- Ethical issues can be mitigated by doing the proper Harvard referencing and formatting.
- Finally, it has thereby been recommended that some numerical data related to SQL and NoSQL can be added for the enrichment of the paper.

Based on the above analysis or the PEER report, it can be concluded that the provided research study was related to the comparative analysis of the SQL as well as the NoSQL databases. Moreover, the relational database has also been recognised within the paper. The paper has both the issues (in terms of ethical, presentation related and many others) and some strong points also. If the researcher can write the paper by avoiding those issues and implementing the recommendations, it would ensure a better paper related to the comparative study of SQL and NoSQL.

2. PART B - Implementation

1.JSON Implementation

1.1 JSON Definition

JSON is a structured data storage format that is frequently used to transfer data between a server and a user. With similar capabilities, a JSON file is a simpler and lighter alternative to XML. These formats operate together to allow stored data to be loaded asynchronously, allowing a server to change its content without having to refresh the page. JSON is also simpler and lighter alternative to XML. Another advantage of JSON is regardless of the programming language, JSON can be used to exchange data between different platforms. Curly brackets are used to start and finish a JSON object. It can have two or more key-values, each separated by a comma. The value can have different data types in a JSON object as follows:

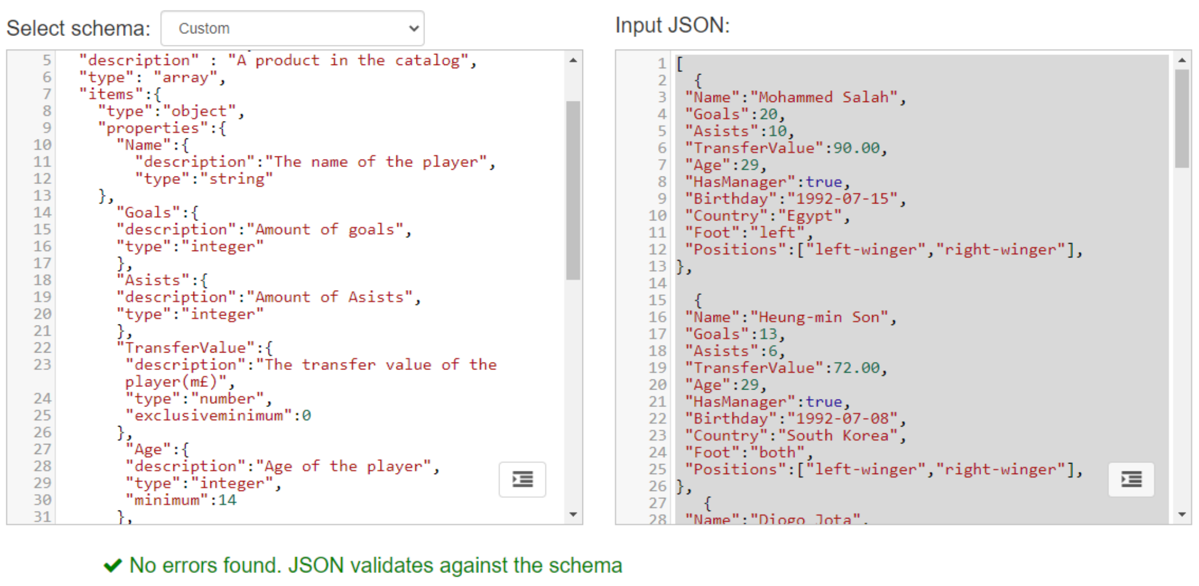
- String
- Number
- Float
- Array
- Object
- Boolean
- Null

The last definition I want to specify regarding JSON is the schema structure. JSON schema is the architecture used to define a standard format for the data in the JSON structure, and it is in the JSON format itself, so it has its own JSON schema.

1.2 JSON Data Structure of My Implementation

In the JSON data I created, I kept the data of the 10 best football players playing in the English Premier League. JSON data is kept as objects inside the array. I kept the values using all the data types used in JSON.

- "Name" : Name of the player as string value
- "Goals": Total goal player scored as number
- "Assists": Total assists of player as number
- "TransferValue": Market value of the player as float
- "Age" : Age of the player as number
- "HasManager": if player has a manager company or not as Boolean
- "Birthday": Birthday of player as string
- "Country": Country of player as string
- "Foot": Foot of player as string
- "Positions": Positions of player as array.


Select schema: Custom

```
  "description" : "A product in the catalog",
  "type": "array",
  "items":{
    "type":"object",
    "properties":{
      "Name":{
        "description":"The name of the player",
        "type":"string"
    },
      "Goals":{
      "description":"Amount of goals",
      "type":"integer"
      },
      "Asists":{
      "description":"Amount of Asists",
      "type":"integer"
      },
      "TransferValue":{
       "description":"The transfer value of the
       player(m£)",
       "type":"number",
       "exclusiveminimum":0
      },
       "Age":{
       "description":"Age of the player",
       "type":"integer",
       "minimum":14
      },
```

Input JSON:

```
[
  {
 "Name":"Mohammed Salah",
 "Goals":20,
 "Asists":10,
 "TransferValue":90.00,
 "Age":29,
 "HasManager":true,
 "Birthday":"1992-07-15",
 "Country":"Egypt",
 "Foot":"left",
 "Positions":["left-winger","right-winger"],
},

  {
 "Name":"Heung-min Son",
 "Goals":13,
 "Asists":6,
 "TransferValue":72.00,
 "Age":29,
 "HasManager":true,
 "Birthday":"1992-07-08",
 "Country":"South Korea",
 "Foot":"both",
 "Positions":["left-winger","right-winger"],
},
   {
 "Name":"Diogo Jota",
```

✔ No errors found. JSON validates against the schema

FIGURE 1:JSON VALIDATION MESSAGE

The JSON schema and all data held in JSON are shared in the appendix as specified in the coursework.

2. Redis Definition and Coursework Implementation

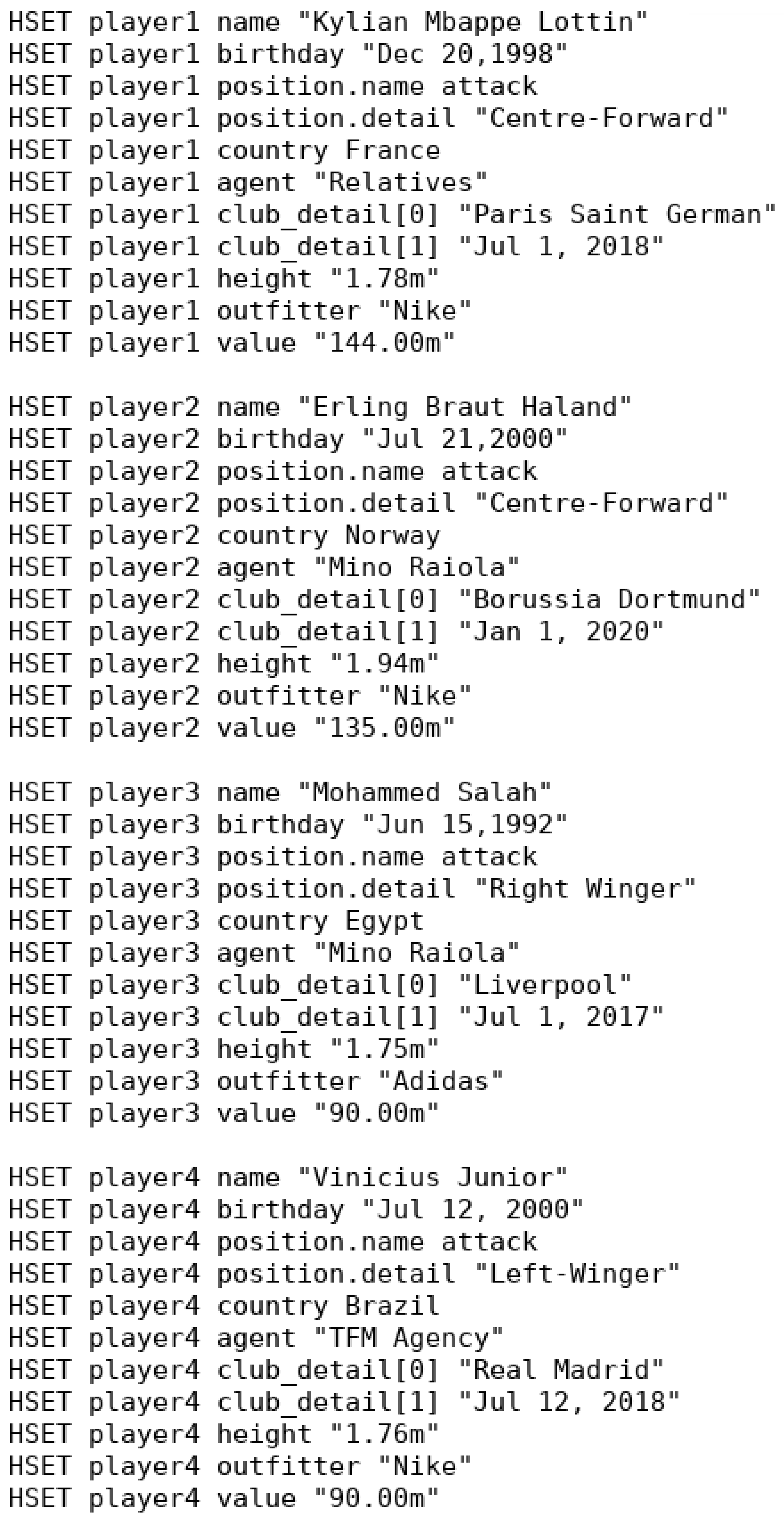
2.1 Redis Definition

Redis is an open source in-memory data store that is commonly used to construct non-relational key-value databases and caches. Redis isn't just a key-value server. One of the main differences from the other options is the ability of Redis to store and use high-level data structures (lists, maps, sets). We use the data cache procedure when traditional database servers are unable to read and write data or when performance is degraded. The goal of caching is to reduce the strain on the background application or database and to get to the data as quickly and cheaply as possible. Redis plays a key role in this, allowing items to be retrieved in milliseconds or less.

2.2 Coursework implementation

2.2.1 Hash Data types

I recorded the detailed information of the top 10 players of the English league as a hash set. Different scenarios were run on this dataset for different scenarios related to the players and the dataset. I created 10 players in a .txt file.



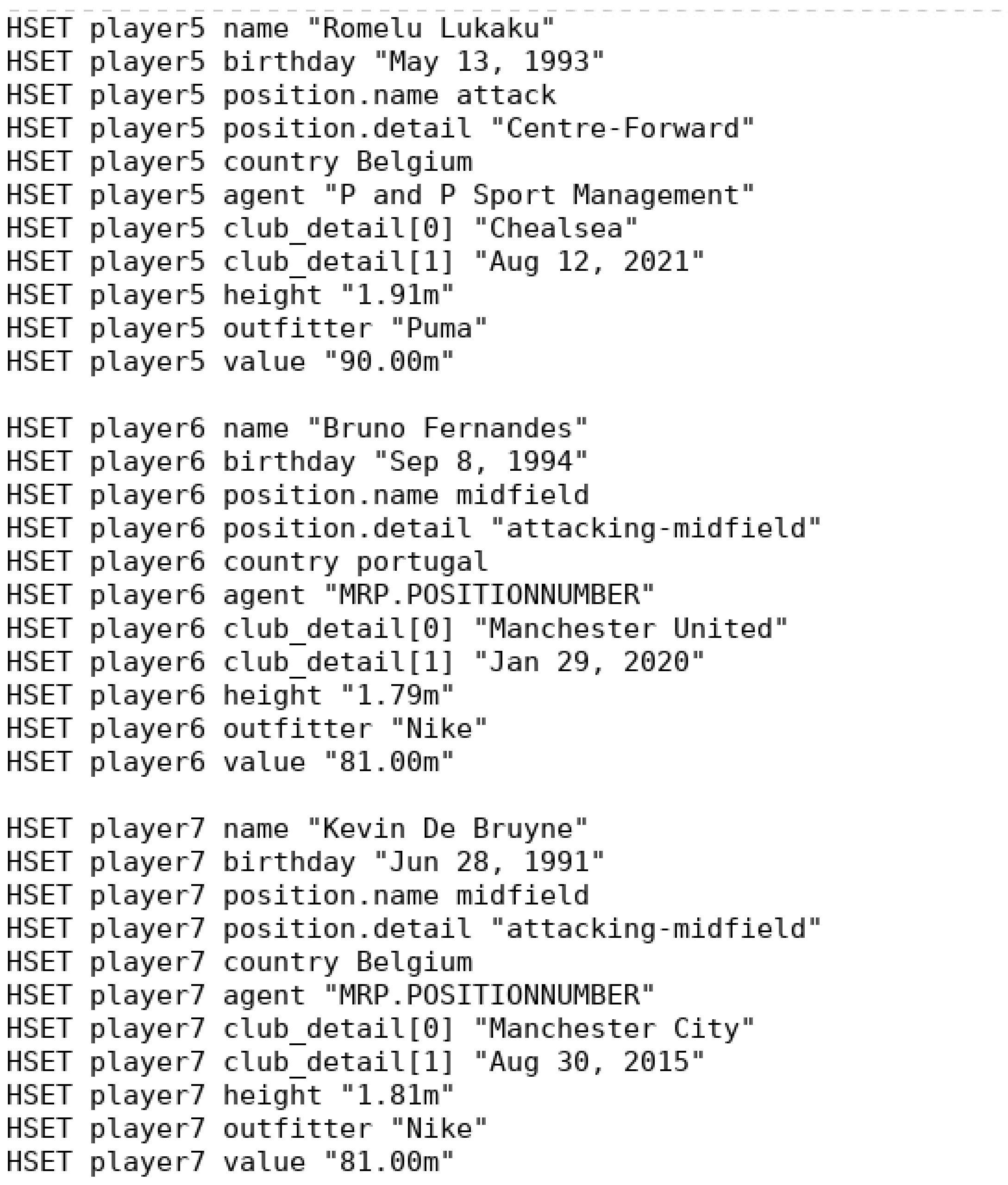
```
HSET player1 name "Kylian Mbappe Lottin"
HSET player1 birthday "Dec 20,1998"
HSET player1 position.name attack
HSET player1 position.detail "Centre-Forward"
HSET player1 country France
HSET player1 agent "Relatives"
HSET player1 club_detail[0] "Paris Saint German"
HSET player1 club_detail[1] "Jul 1, 2018"
HSET player1 height "1.78m"
HSET player1 outfitter "Nike"
HSET player1 value "144.00m"

HSET player2 name "Erling Braut Haland"
HSET player2 birthday "Jul 21,2000"
HSET player2 position.name attack
HSET player2 position.detail "Centre-Forward"
HSET player2 country Norway
HSET player2 agent "Mino Raiola"
HSET player2 club_detail[0] "Borussia Dortmund"
HSET player2 club_detail[1] "Jan 1, 2020"
HSET player2 height "1.94m"
HSET player2 outfitter "Nike"
HSET player2 value "135.00m"

HSET player3 name "Mohammed Salah"
HSET player3 birthday "Jun 15,1992"
HSET player3 position.name attack
HSET player3 position.detail "Right Winger"
HSET player3 country Egypt
HSET player3 agent "Mino Raiola"
HSET player3 club_detail[0] "Liverpool"
HSET player3 club_detail[1] "Jul 1, 2017"
HSET player3 height "1.75m"
HSET player3 outfitter "Adidas"
HSET player3 value "90.00m"

HSET player4 name "Vinicius Junior"
HSET player4 birthday "Jul 12, 2000"
HSET player4 position.name attack
HSET player4 position.detail "Left-Winger"
HSET player4 country Brazil
HSET player4 agent "TFM Agency"
HSET player4 club_detail[0] "Real Madrid"
HSET player4 club_detail[1] "Jul 12, 2018"
HSET player4 height "1.76m"
HSET player4 outfitter "Nike"
HSET player4 value "90.00m"
```

FIGURE 2:FIRST4 PLAYERS

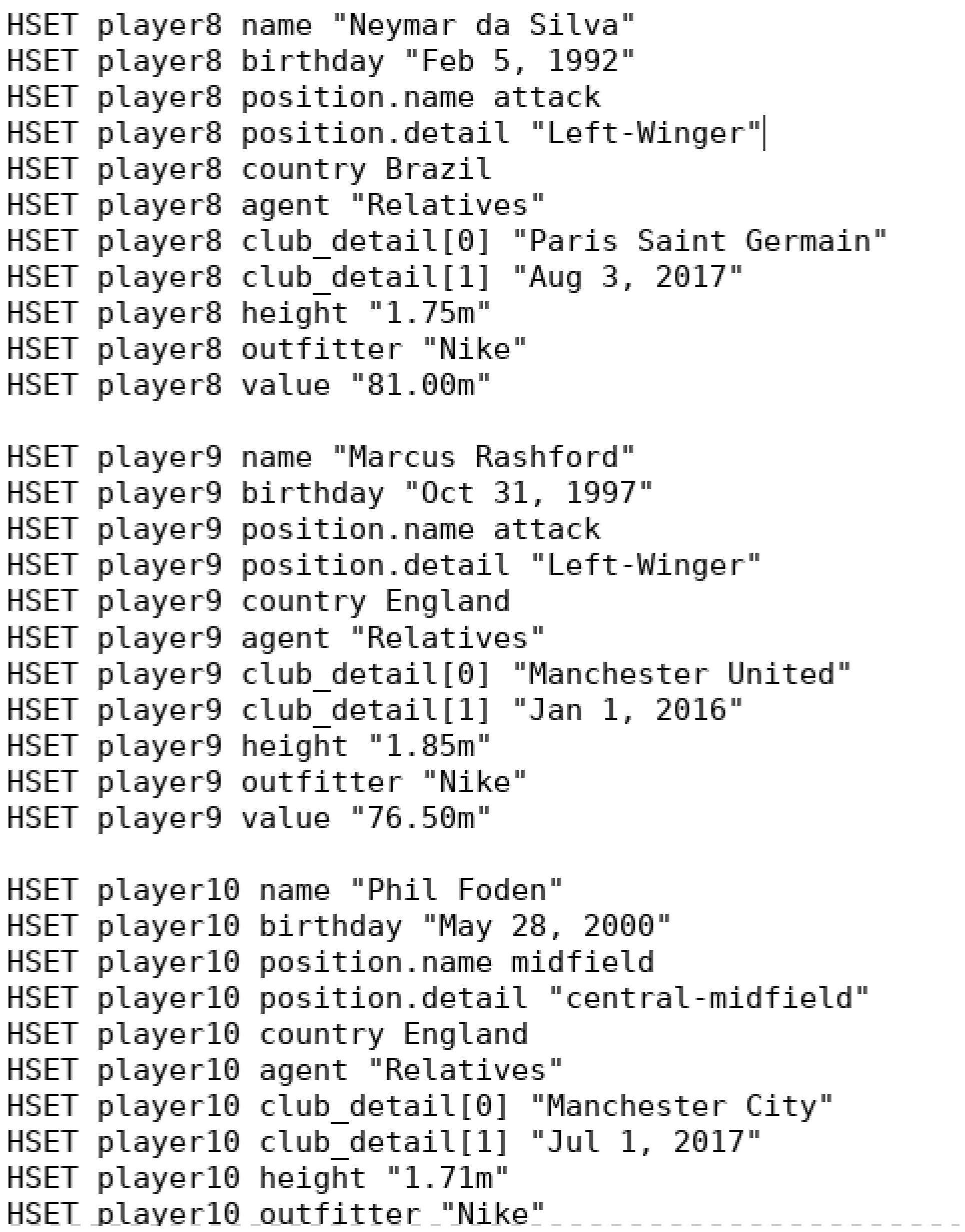


```
HSET player5 name "Romelu Lukaku"
HSET player5 birthday "May 13, 1993"
HSET player5 position.name attack
HSET player5 position.detail "Centre-Forward"
HSET player5 country Belgium
HSET player5 agent "P and P Sport Management"
HSET player5 club_detail[0] "Chealsea"
HSET player5 club_detail[1] "Aug 12, 2021"
HSET player5 height "1.91m"
HSET player5 outfitter "Puma"
HSET player5 value "90.00m"

HSET player6 name "Bruno Fernandes"
HSET player6 birthday "Sep 8, 1994"
HSET player6 position.name midfield
HSET player6 position.detail "attacking-midfield"
HSET player6 country portugal
HSET player6 agent "MRP.POSITIONNUMBER"
HSET player6 club_detail[0] "Manchester United"
HSET player6 club_detail[1] "Jan 29, 2020"
HSET player6 height "1.79m"
HSET player6 outfitter "Nike"
HSET player6 value "81.00m"

HSET player7 name "Kevin De Bruyne"
HSET player7 birthday "Jun 28, 1991"
HSET player7 position.name midfield
HSET player7 position.detail "attacking-midfield"
HSET player7 country Belgium
HSET player7 agent "MRP.POSITIONNUMBER"
HSET player7 club_detail[0] "Manchester City"
HSET player7 club_detail[1] "Aug 30, 2015"
HSET player7 height "1.81m"
HSET player7 outfitter "Nike"
HSET player7 value "81.00m"
```

FIGURE 3:SECOND 3 PLAYER

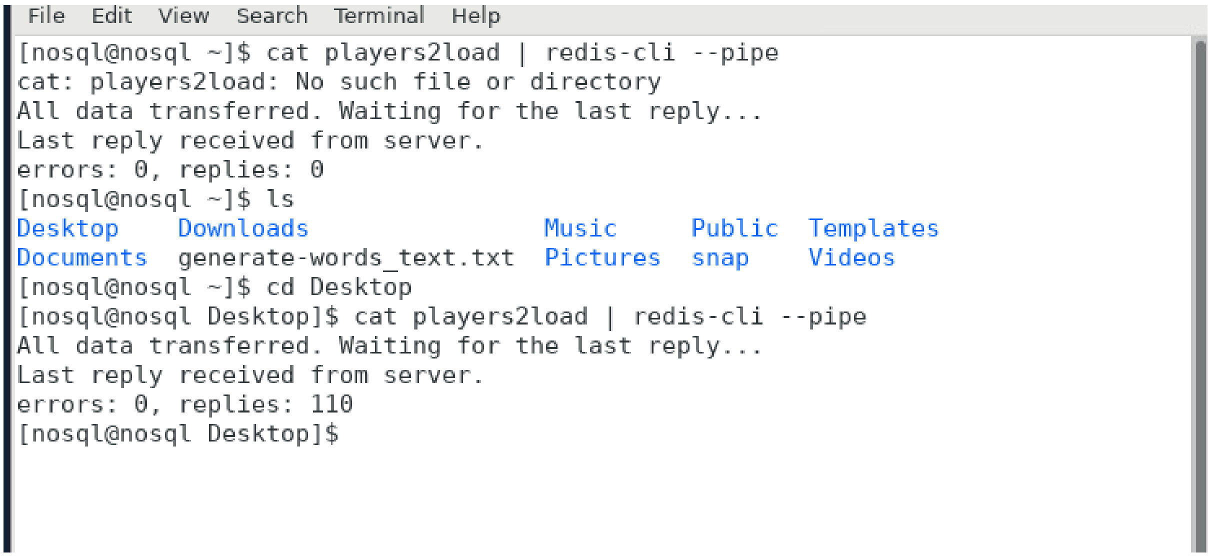

```
HSET player8 name "Neymar da Silva"
HSET player8 birthday "Feb 5, 1992"
HSET player8 position.name attack
HSET player8 position.detail "Left-Winger"
HSET player8 country Brazil
HSET player8 agent "Relatives"
HSET player8 club_detail[0] "Paris Saint Germain"
HSET player8 club_detail[1] "Aug 3, 2017"
HSET player8 height "1.75m"
HSET player8 outfitter "Nike"
HSET player8 value "81.00m"

HSET player9 name "Marcus Rashford"
HSET player9 birthday "Oct 31, 1997"
HSET player9 position.name attack
HSET player9 position.detail "Left-Winger"
HSET player9 country England
HSET player9 agent "Relatives"
HSET player9 club_detail[0] "Manchester United"
HSET player9 club_detail[1] "Jan 1, 2016"
HSET player9 height "1.85m"
HSET player9 outfitter "Nike"
HSET player9 value "76.50m"

HSET player10 name "Phil Foden"
HSET player10 birthday "May 28, 2000"
HSET player10 position.name midfield
HSET player10 position.detail "central-midfield"
HSET player10 country England
HSET player10 agent "Relatives"
HSET player10 club_detail[0] "Manchester City"
HSET player10 club_detail[1] "Jul 1, 2017"
HSET player10 height "1.71m"
HSET player10 outfitter "Nike"
```

FIGURE 4:LAST 3 PLAYERS

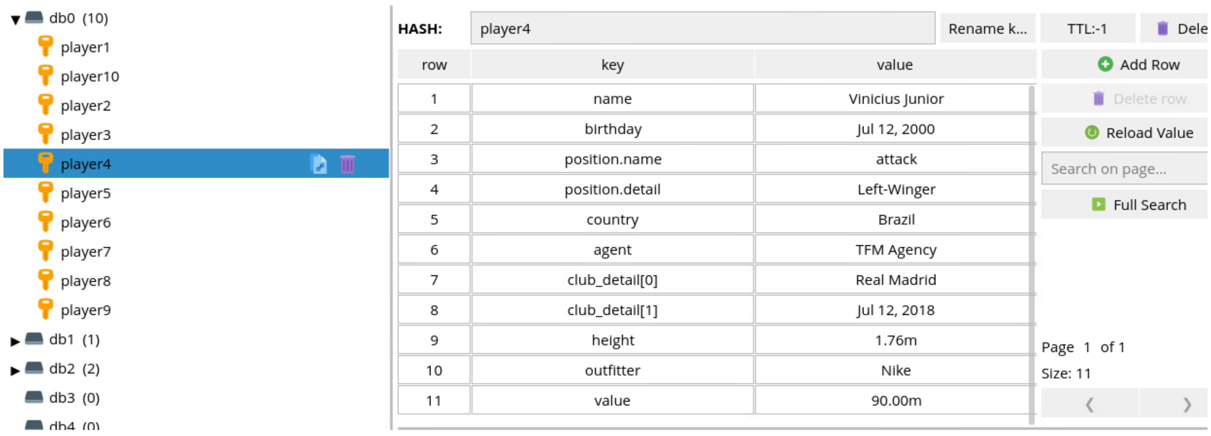
I used "cat players2load | redis-cli –pipe command to save all players to redis.



```
[nosql@nosql ~]$ cat players2load | redis-cli --pipe
cat: players2load: No such file or directory
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 0
[nosql@nosql ~]$ ls
Desktop    Downloads               Music     Public  Templates
Documents  generate-words_text.txt Pictures  snap    Videos
[nosql@nosql ~]$ cd Desktop
[nosql@nosql Desktop]$ cat players2load | redis-cli --pipe
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 110
[nosql@nosql Desktop]$
```

FIGURE 5:SPECIAL COMMAND

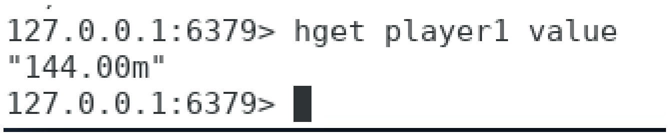
After that, I checked the general structure from GUI for REDIS.



row	key	value
1	name	Vinicius Junior
2	birthday	Jul 12, 2000
3	position.name	attack
4	position.detail	Left-Winger
5	country	Brazil
6	agent	TFM Agency
7	club_detail[0]	Real Madrid
8	club_detail[1]	Jul 12, 2018
9	height	1.76m
10	outfitter	Nike
11	value	90.00m

FIGURE 6:DATA STRUCTURE OF REDIS

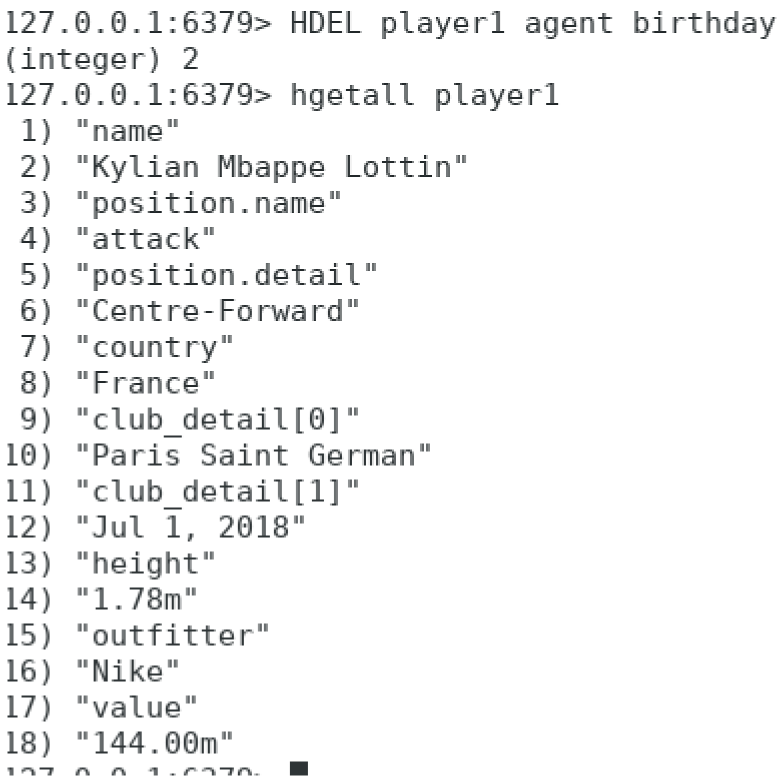
Case 1: Retrieve the value of player1 from database.



```
127.0.0.1:6379> hget player1 value
"144.00m"
127.0.0.1:6379>
```

FIGURE 7:RETRIEVE TRANSFER VALUE OF PLAYER1

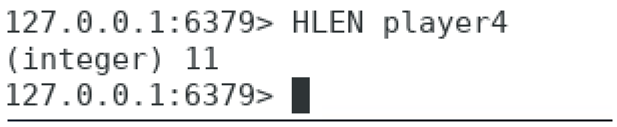
Case2: delete the birthday of player1 and check if the field is deleted.



```
127.0.0.1:6379> HDEL player1 agent birthday
(integer) 2
127.0.0.1:6379> hgetall player1
 1) "name"
 2) "Kylian Mbappe Lottin"
 3) "position.name"
 4) "attack"
 5) "position.detail"
 6) "Centre-Forward"
 7) "country"
 8) "France"
 9) "club_detail[0]"
10) "Paris Saint German"
11) "club_detail[1]"
12) "Jul 1, 2018"
13) "height"
14) "1.78m"
15) "outfitter"
16) "Nike"
17) "value"
18) "144.00m"
```

FIGURE 8:DELETE BIRTHDAY FIELD

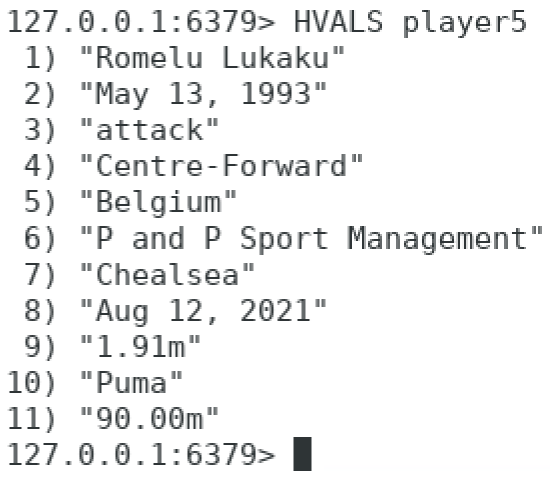
Case3: Check the length of player4



```
127.0.0.1:6379> HLEN player4
(integer) 11
127.0.0.1:6379>
```

FIGURE 9:LENGTH OF PLAYER4

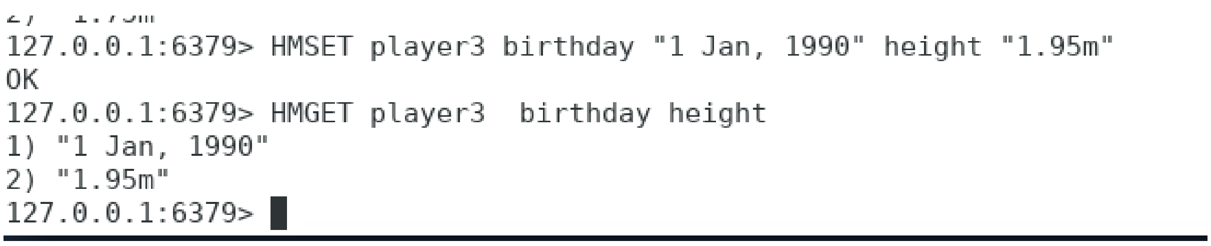
Case4: Show only values of player5



```
127.0.0.1:6379> HVALS player5
 1) "Romelu Lukaku"
 2) "May 13, 1993"
 3) "attack"
 4) "Centre-Forward"
 5) "Belgium"
 6) "P and P Sport Management"
 7) "Chealsea"
 8) "Aug 12, 2021"
 9) "1.91m"
10) "Puma"
11) "90.00m"
127.0.0.1:6379>
```

FIGURE 10:VALUES OF PLAYER5

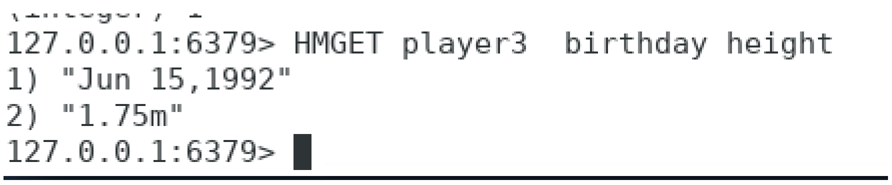
Case5: Change the values of player3



```
127.0.0.1:6379> HMSET player3 birthday "1 Jan, 1990" height "1.95m"
OK
127.0.0.1:6379> HMGET player3  birthday height
1) "1 Jan, 1990"
2) "1.95m"
127.0.0.1:6379> █
```

FIGURE 11:CHANGE THE VALUES OF PLAYER3

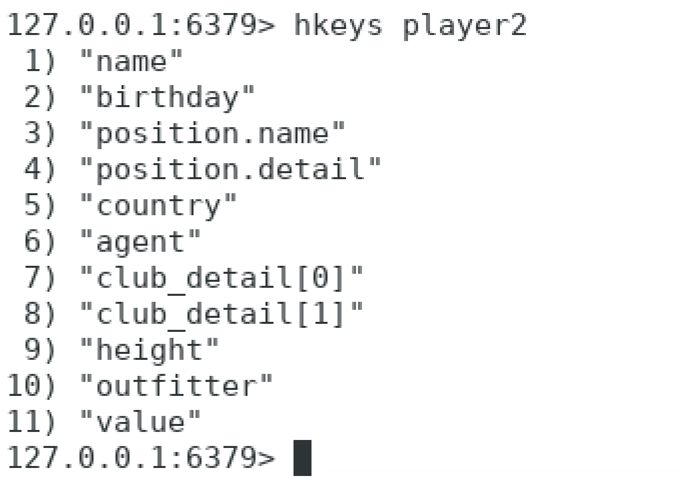
Case6: get only birthday and height values of player3



```
127.0.0.1:6379> HMGET player3  birthday height
1) "Jun 15,1992"
2) "1.75m"
127.0.0.1:6379> █
```

FIGURE 12:GET SPECIFIC FIELDS FROM PLAYER3

Case 6: Get all the key values of player2



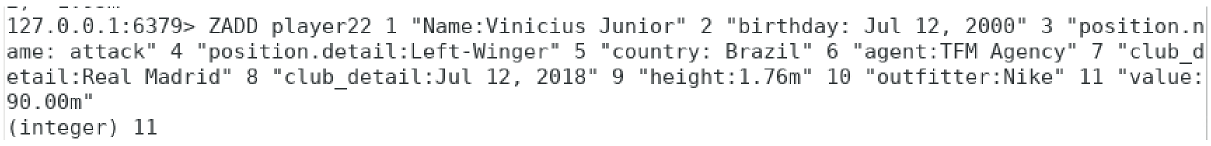
```
127.0.0.1:6379> hkeys player2
 1) "name"
 2) "birthday"
 3) "position.name"
 4) "position.detail"
 5) "country"
 6) "agent"
 7) "club_detail[0]"
 8) "club_detail[1]"
 9) "height"
10) "outfitter"
11) "value"
127.0.0.1:6379> █
```

FIGURE 13:GET THE KEY VALUES OF PLAYER2

2.2.2 Sorted Sets

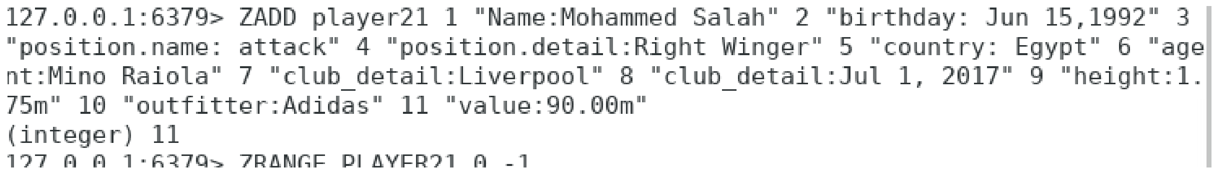
Case7: Find the union of player21 and player22.

First, I created 2 sets.



```
127.0.0.1:6379> ZADD player22 1 "Name:Vinicius Junior" 2 "birthday: Jul 12, 2000" 3 "position.n
ame: attack" 4 "position.detail:Left-Winger" 5 "country: Brazil" 6 "agent:TFM Agency" 7 "club_d
etail:Real Madrid" 8 "club_detail:Jul 12, 2018" 9 "height:1.76m" 10 "outfitter:Nike" 11 "value:
90.00m"
(integer) 11
```

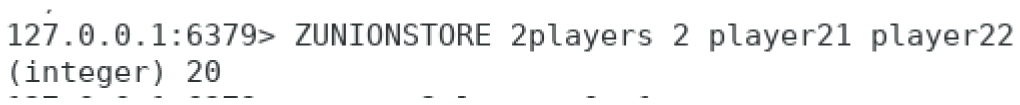
FIGURE 14:PLAYER22 CREATION



```
127.0.0.1:6379> ZADD player21 1 "Name:Mohammed Salah" 2 "birthday: Jun 15,1992" 3
"position.name: attack" 4 "position.detail:Right Winger" 5 "country: Egypt" 6 "age
nt:Mino Raiola" 7 "club_detail:Liverpool" 8 "club_detail:Jul 1, 2017" 9 "height:1.
75m" 10 "outfitter:Adidas" 11 "value:90.00m"
(integer) 11
127.0.0.1:6379> ZRANGE PLAYER21 0 -1
```

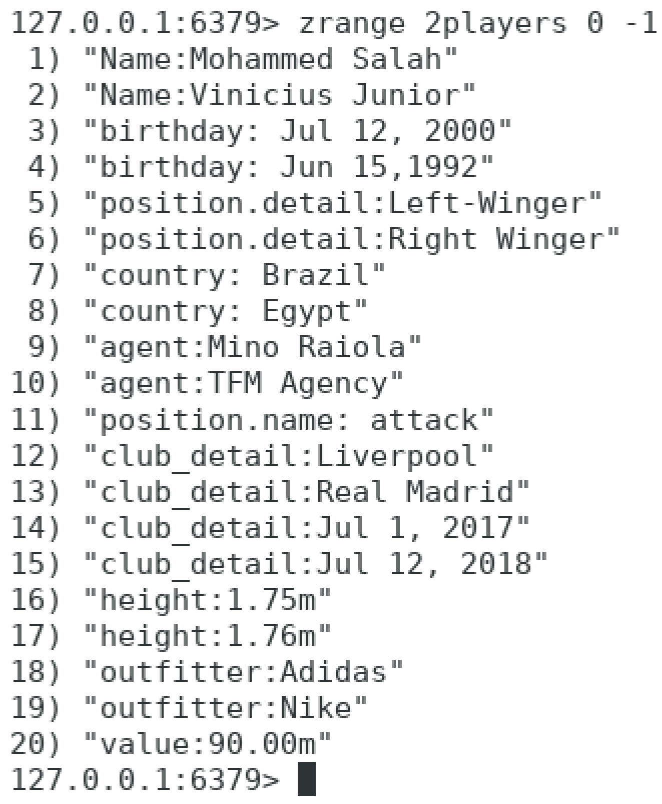
FIGURE 15:PLAYER21 CREATION

I used "ZUNIONSTORE" command to union both player data.



```
127.0.0.1:6379> ZUNIONSTORE 2players 2 player21 player22
(integer) 20
```

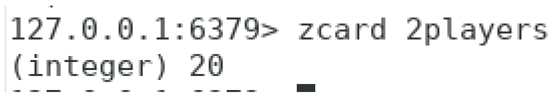
FIGURE 16: ZUNIONSTORE COMMAND



```
127.0.0.1:6379> zrange 2players 0 -1
 1) "Name:Mohammed Salah"
 2) "Name:Vinicius Junior"
 3) "birthday: Jul 12, 2000"
 4) "birthday: Jun 15,1992"
 5) "position.detail:Left-Winger"
 6) "position.detail:Right Winger"
 7) "country: Brazil"
 8) "country: Egypt"
 9) "agent:Mino Raiola"
10) "agent:TFM Agency"
11) "position.name: attack"
12) "club_detail:Liverpool"
13) "club_detail:Real Madrid"
14) "club_detail:Jul 1, 2017"
15) "club_detail:Jul 12, 2018"
16) "height:1.75m"
17) "height:1.76m"
18) "outfitter:Adidas"
19) "outfitter:Nike"
20) "value:90.00m"
127.0.0.1:6379> 
```

FIGURE 17:UNIFIED SET OF 2 PLAYERS

Case8: Find the total number of unified set's length.



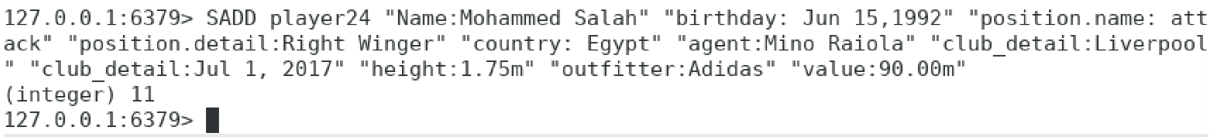
```
127.0.0.1:6379> zcard 2players
(integer) 20
```

FIGURE 18:TOTAL NUMBER OF FIELDS

2.2.3 Sets

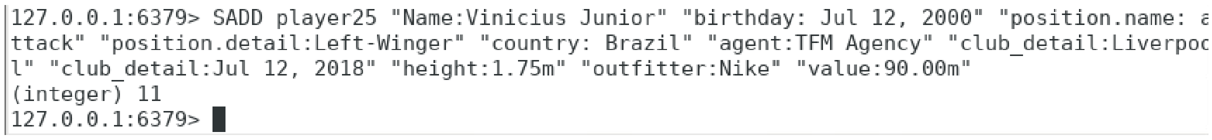
Case9: Find the different fields of 2 players using sets.

First, I added two sets (player25 and player24)



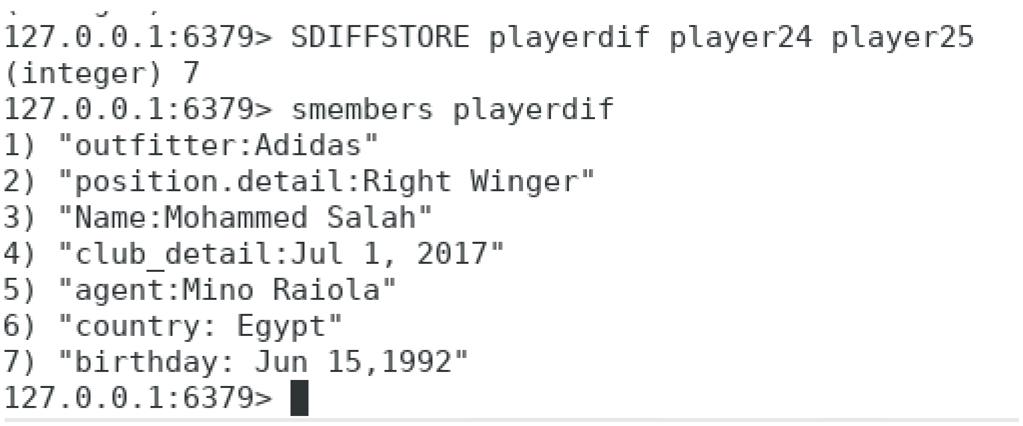
```
127.0.0.1:6379> SADD player24 "Name:Mohammed Salah" "birthday: Jun 15,1992" "position.name: att
ack" "position.detail:Right Winger" "country: Egypt" "agent:Mino Raiola" "club_detail:Liverpool
" "club_detail:Jul 1, 2017" "height:1.75m" "outfitter:Adidas" "value:90.00m"
(integer) 11
127.0.0.1:6379>
```

FIGURE 19:PLAYER24



```
127.0.0.1:6379> SADD player25 "Name:Vinicius Junior" "birthday: Jul 12, 2000" "position.name: a
ttack" "position.detail:Left-Winger" "country: Brazil" "agent:TFM Agency" "club_detail:Liverpoo
l" "club_detail:Jul 12, 2018" "height:1.75m" "outfitter:Nike" "value:90.00m"
(integer) 11
127.0.0.1:6379>
```

FIGURE 20:PLAYER25

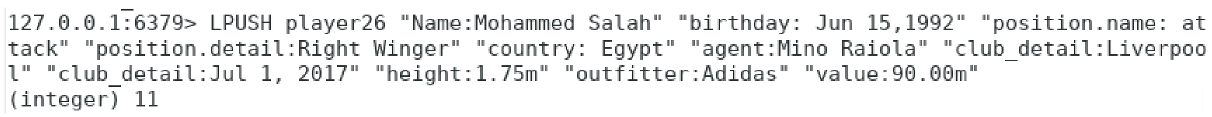


```
127.0.0.1:6379> SDIFFSTORE playerdif player24 player25
(integer) 7
127.0.0.1:6379> smembers playerdif
1) "outfitter:Adidas"
2) "position.detail:Right Winger"
3) "Name:Mohammed Salah"
4) "club_detail:Jul 1, 2017"
5) "agent:Mino Raiola"
6) "country: Egypt"
7) "birthday: Jun 15,1992"
127.0.0.1:6379>
```

FIGURE 21:DIFFERENCE BETWEEN TWO SETS

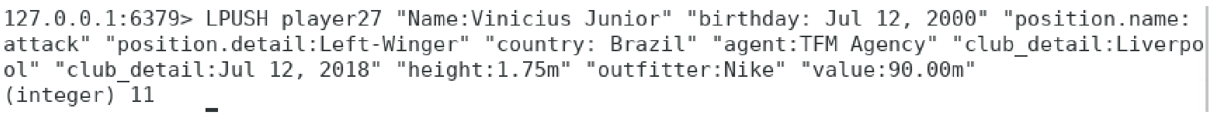
2.2.4 Lists

First, I added 3 players(player26,player27,player28) as List.



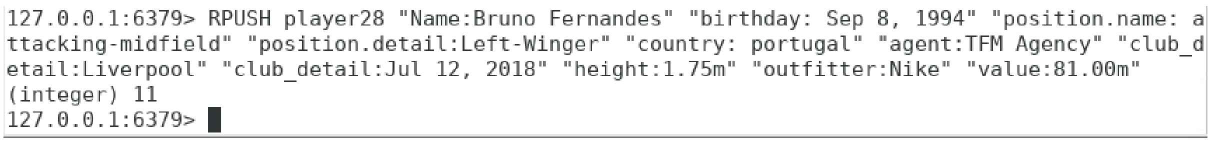
```
127.0.0.1:6379> LPUSH player26 "Name:Mohammed Salah" "birthday: Jun 15,1992" "position.name: at
tack" "position.detail:Right Winger" "country: Egypt" "agent:Mino Raiola" "club_detail:Liverpoo
l" "club_detail:Jul 1, 2017" "height:1.75m" "outfitter:Adidas" "value:90.00m"
(integer) 11
```

FIGURE 22:CREATION OF PLAYER26



```
127.0.0.1:6379> LPUSH player27 "Name:Vinicius Junior" "birthday: Jul 12, 2000" "position.name:
attack" "position.detail:Left-Winger" "country: Brazil" "agent:TFM Agency" "club_detail:Liverpo
ol" "club_detail:Jul 12, 2018" "height:1.75m" "outfitter:Nike" "value:90.00m"
(integer) 11
```

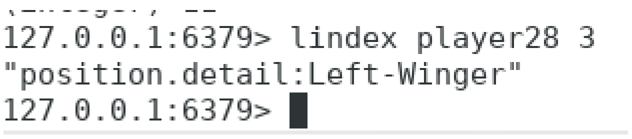
FIGURE 23:CREATION OF PLAYER27



```
127.0.0.1:6379> RPUSH player28 "Name:Bruno Fernandes" "birthday: Sep 8, 1994" "position.name: a
ttacking-midfield" "position.detail:Left-Winger" "country: portugal" "agent:TFM Agency" "club_d
etail:Liverpool" "club_detail:Jul 12, 2018" "height:1.75m" "outfitter:Nike" "value:81.00m"
(integer) 11
127.0.0.1:6379>
```

FIGURE 24:CREATION OF PLAYER28

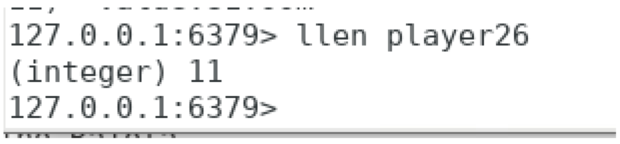
Case10: show the third index value of player28.



```
127.0.0.1:6379> lindex player28 3
"position.detail:Left-Winger"
127.0.0.1:6379>
```

FIGURE 25:THIRD INDEX OF PLAYER28

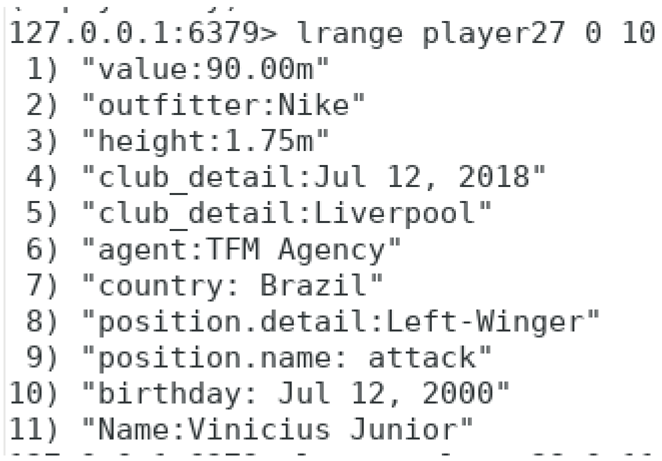
Case11: Show the length of player26



```
127.0.0.1:6379> llen player26
(integer) 11
127.0.0.1:6379>
```

FIGURE 26:LENGTH OF PLAYER26

Case12: Show the first 10 fields of player27



```
127.0.0.1:6379> lrange player27 0 10
 1) "value:90.00m"
 2) "outfitter:Nike"
 3) "height:1.75m"
 4) "club_detail:Jul 12, 2018"
 5) "club_detail:Liverpool"
 6) "agent:TFM Agency"
 7) "country: Brazil"
 8) "position.detail:Left-Winger"
 9) "position.name: attack"
10) "birthday: Jul 12, 2000"
11) "Name:Vinicius Junior"
```

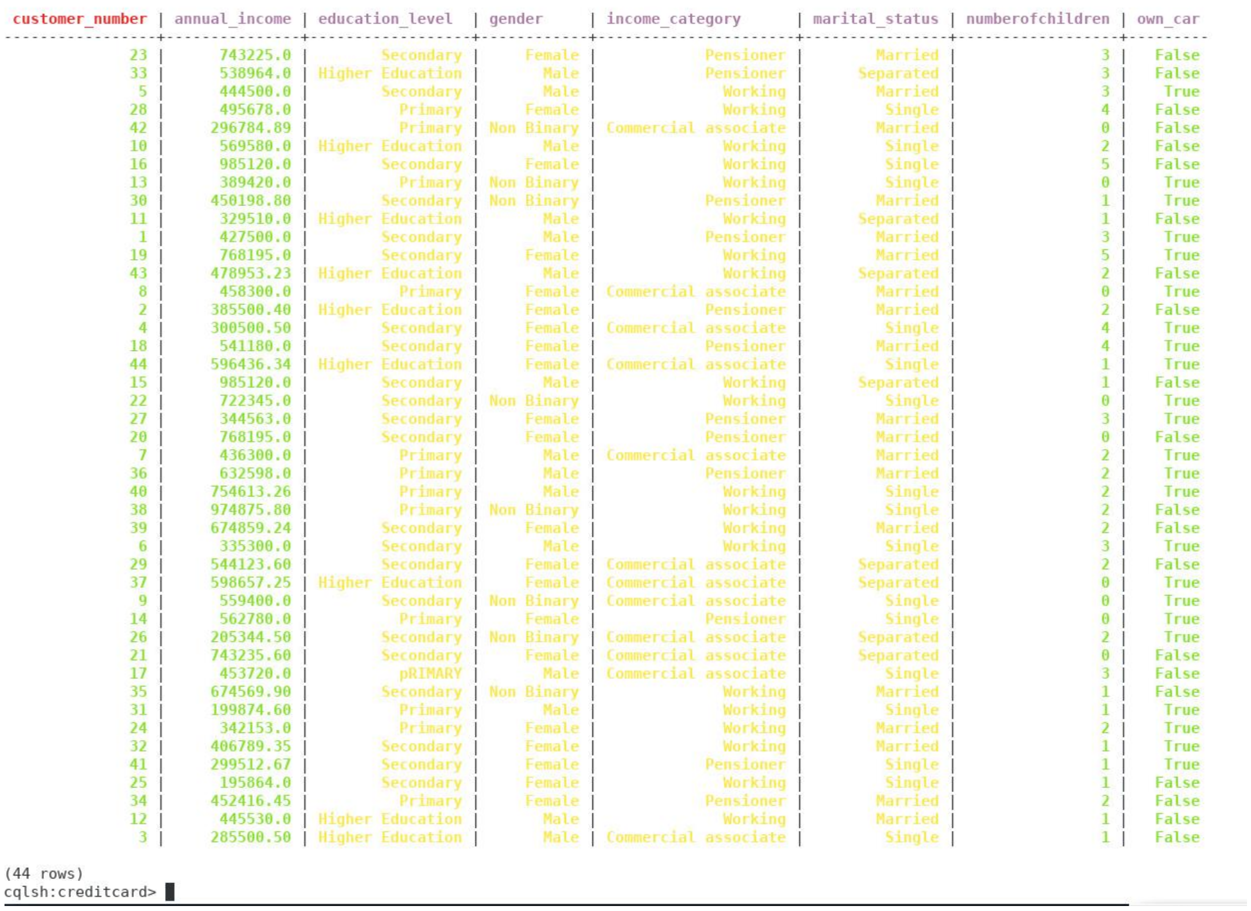
FIGURE 27:FIRST 10 FIELDS OF PLAYER27

3. Cassandra definition and Coursework Implementation

Apache Cassandra is simply a NoSQL database. In comparison to relational databases, it features a more flexible structure and data model. It's especially well-suited to storing non-relational data. Apache Cassandra's distributed structure is its most appealing feature. Data is kept in different components of this structure based on the criteria set in a cluster.

3.1 Coursework Implementation

I keep the data of 44 customers used for credit purchase. Here I used all the different data types kept in Cassandra.

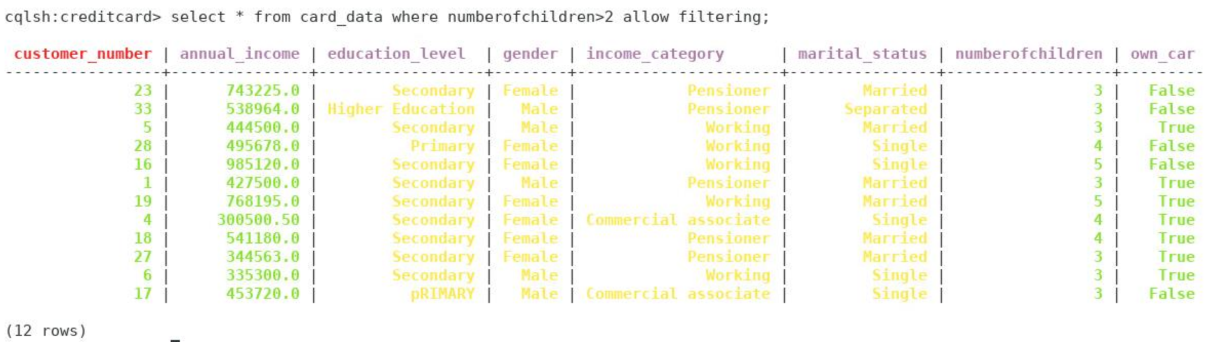


```
 customer_number | annual_income | education_level  | gender     | income_category      | marital_status | numberofchildren | own_car
-----------------+---------------+------------------+------------+----------------------+----------------+------------------+---------
              23 |      743225.0 |        Secondary |     Female |            Pensioner |        Married |                3 |   False
              33 |      538964.0 | Higher Education |       Male |            Pensioner |      Separated |                3 |   False
               5 |      444500.0 |        Secondary |       Male |              Working |        Married |                3 |    True
              28 |      495678.0 |          Primary |     Female |              Working |         Single |                4 |   False
              42 |     296784.89 |          Primary | Non Binary | Commercial associate |        Married |                0 |   False
              10 |      569580.0 | Higher Education |       Male |              Working |         Single |                2 |   False
              16 |      985120.0 |        Secondary |     Female |              Working |         Single |                5 |   False
              13 |      389420.0 |          Primary | Non Binary |              Working |         Single |                0 |    True
              30 |     450198.80 |        Secondary | Non Binary |            Pensioner |        Married |                1 |    True
              11 |      329510.0 | Higher Education |       Male |              Working |      Separated |                1 |   False
               1 |      427500.0 |        Secondary |       Male |            Pensioner |        Married |                3 |    True
              19 |      768195.0 |        Secondary |     Female |              Working |        Married |                5 |    True
              43 |     478953.23 | Higher Education |       Male |              Working |      Separated |                2 |   False
               8 |      458300.0 |          Primary |     Female | Commercial associate |        Married |                0 |    True
               2 |     385500.40 | Higher Education |     Female |            Pensioner |        Married |                2 |   False
               4 |     300500.50 |        Secondary |     Female | Commercial associate |         Single |                4 |    True
              18 |      541180.0 |        Secondary |     Female |            Pensioner |        Married |                4 |    True
              44 |     596436.34 | Higher Education |     Female | Commercial associate |         Single |                1 |    True
              15 |      985120.0 |        Secondary |       Male |              Working |      Separated |                1 |   False
              22 |      722345.0 |        Secondary | Non Binary |              Working |         Single |                0 |    True
              27 |      344563.0 |        Secondary |     Female |            Pensioner |        Married |                3 |    True
              20 |      768195.0 |        Secondary |     Female |            Pensioner |        Married |                0 |   False
               7 |      436300.0 |          Primary |       Male | Commercial associate |        Married |                2 |    True
              36 |      632598.0 |          Primary |       Male |            Pensioner |        Married |                2 |    True
              40 |     754613.26 |          Primary |       Male |              Working |         Single |                2 |    True
              38 |     974875.80 |          Primary | Non Binary |              Working |         Single |                2 |   False
              39 |     674859.24 |        Secondary |     Female |              Working |        Married |                2 |   False
               6 |      335300.0 |        Secondary |       Male |              Working |         Single |                3 |    True
              29 |     544123.60 |        Secondary |     Female | Commercial associate |      Separated |                2 |   False
              37 |     598657.25 | Higher Education |     Female | Commercial associate |      Separated |                0 |    True
               9 |      559400.0 |        Secondary | Non Binary | Commercial associate |         Single |                0 |    True
              14 |      562780.0 |          Primary |     Female |            Pensioner |         Single |                0 |    True
              26 |     205344.50 |        Secondary | Non Binary | Commercial associate |      Separated |                2 |    True
              21 |     743235.60 |        Secondary |     Female | Commercial associate |      Separated |                0 |   False
              17 |      453720.0 |          pRIMARY |       Male | Commercial associate |         Single |                3 |   False
              35 |     674569.90 |        Secondary | Non Binary |              Working |        Married |                1 |   False
              31 |     199874.60 |          Primary |       Male |              Working |         Single |                1 |    True
              24 |      342153.0 |          Primary |     Female |              Working |        Married |                2 |    True
              32 |     406789.35 |        Secondary |     Female |              Working |        Married |                1 |    True
              41 |     299512.67 |        Secondary |     Female |            Pensioner |         Single |                1 |    True
              25 |      195864.0 |        Secondary |     Female |              Working |         Single |                1 |   False
              34 |     452416.45 |          Primary |     Female |            Pensioner |        Married |                2 |   False
              12 |      445530.0 | Higher Education |       Male |              Working |        Married |                1 |   False
               3 |     285500.50 | Higher Education |       Male | Commercial associate |         Single |                1 |   False

(44 rows)
cqlsh:creditcard> █
```

FIGURE 28:CASSANDRA DATA

CASE 1: Show all customers who have more than 2 kids.



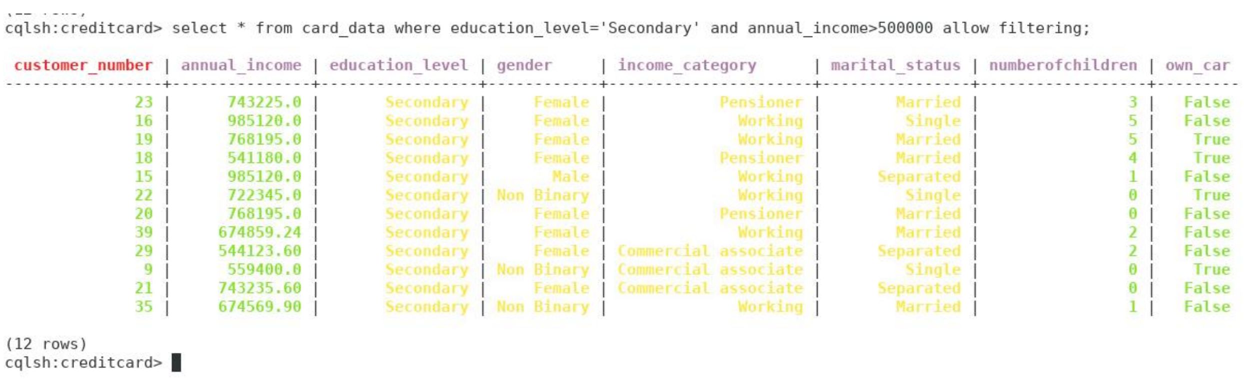
```
cqlsh:creditcard> select * from card_data where numberofchildren>2 allow filtering;

 customer_number | annual_income | education_level  | gender | income_category      | marital_status | numberofchildren | own_car
-----------------+---------------+------------------+--------+----------------------+----------------+------------------+---------
              23 |      743225.0 |        Secondary | Female |            Pensioner |        Married |                3 |   False
              33 |      538964.0 | Higher Education |   Male |            Pensioner |      Separated |                3 |   False
               5 |      444500.0 |        Secondary |   Male |              Working |        Married |                3 |    True
              28 |      495678.0 |          Primary | Female |              Working |         Single |                4 |   False
              16 |      985120.0 |        Secondary | Female |              Working |         Single |                5 |   False
               1 |      427500.0 |        Secondary |   Male |            Pensioner |        Married |                3 |    True
              19 |      768195.0 |        Secondary | Female |              Working |        Married |                5 |    True
               4 |     300500.50 |        Secondary | Female | Commercial associate |         Single |                4 |    True
              18 |      541180.0 |        Secondary | Female |            Pensioner |        Married |                4 |    True
              27 |      344563.0 |        Secondary | Female |            Pensioner |        Married |                3 |    True
               6 |      335300.0 |        Secondary |   Male |              Working |         Single |                3 |    True
              17 |      453720.0 |          pRIMARY |   Male | Commercial associate |         Single |                3 |   False

(12 rows)
```

FIGURE 29:MORE THAN 2 KIDS

Case 2: Show all customers whose education level is Secondary and annual income is greater than 500000.



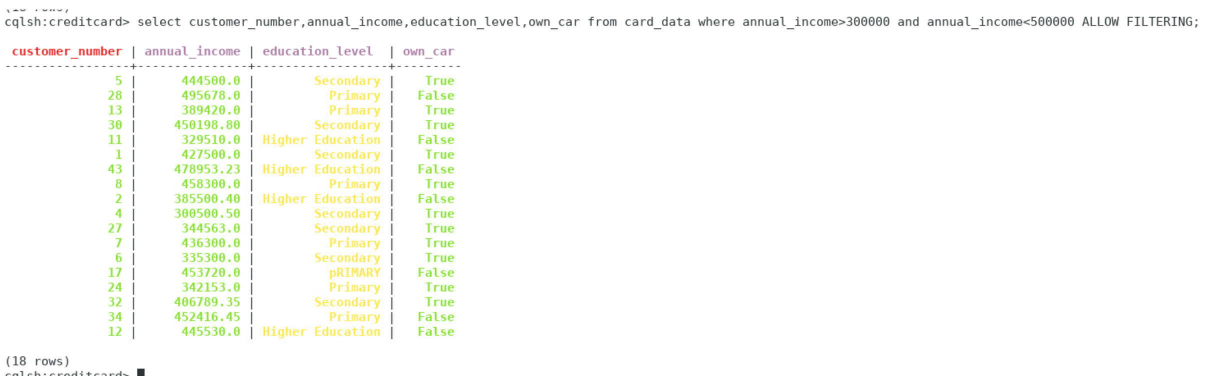
```
cqlsh:creditcard> select * from card_data where education_level='Secondary' and annual_income>500000 allow filtering;

 customer_number | annual_income | education_level | gender     | income_category      | marital_status | numberofchildren | own_car
-----------------+---------------+-----------------+------------+----------------------+----------------+------------------+---------
              23 |      743225.0 |       Secondary |     Female |            Pensioner |        Married |                3 |   False
              16 |      985120.0 |       Secondary |     Female |              Working |         Single |                5 |   False
              19 |      768195.0 |       Secondary |     Female |              Working |        Married |                5 |    True
              18 |      541180.0 |       Secondary |     Female |            Pensioner |        Married |                4 |    True
              15 |      985120.0 |       Secondary |       Male |              Working |      Separated |                1 |   False
              22 |      722345.0 |       Secondary | Non Binary |              Working |         Single |                0 |    True
              20 |      768195.0 |       Secondary |     Female |            Pensioner |        Married |                0 |   False
              39 |     674859.24 |       Secondary |     Female |              Working |        Married |                2 |   False
              29 |     544123.60 |       Secondary |     Female | Commercial associate |      Separated |                2 |   False
               9 |      559400.0 |       Secondary | Non Binary | Commercial associate |         Single |                0 |    True
              21 |     743235.60 |       Secondary |     Female | Commercial associate |      Separated |                0 |   False
              35 |     674569.90 |       Secondary | Non Binary |              Working |        Married |                1 |   False

(12 rows)
cqlsh:creditcard> █
```

FIGURE 30:QUERY2 CASSANDRA

Case 3: Show customer data who has annual income between 30000 and 500000



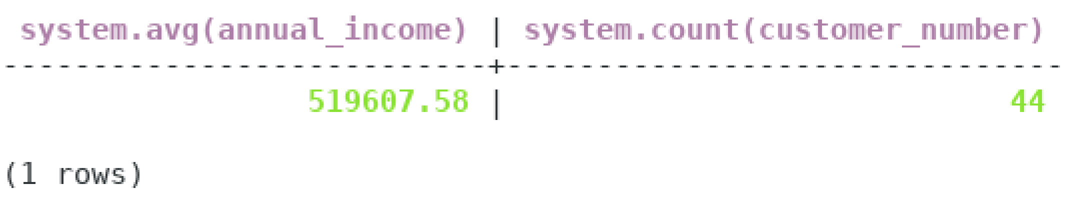
```
cqlsh:creditcard> select customer_number,annual_income,education_level,own_car from card_data where annual_income>300000 and annual_income<500000 ALLOW FILTERING;

 customer_number | annual_income | education_level  | own_car
-----------------+---------------+------------------+---------
               5 |      444500.0 |        Secondary |    True
              28 |      495678.0 |          Primary |   False
              13 |      389420.0 |          Primary |    True
              30 |     450198.80 |        Secondary |    True
              11 |      329510.0 | Higher Education |   False
               1 |      427500.0 |        Secondary |    True
              43 |     478953.23 | Higher Education |   False
               8 |      458300.0 |          Primary |    True
               2 |     385500.40 | Higher Education |   False
               4 |     300500.50 |        Secondary |    True
              27 |      344563.0 |        Secondary |    True
               7 |      436300.0 |          Primary |    True
               6 |      335300.0 |        Secondary |    True
              17 |      453720.0 |          pRIMARY |   False
              24 |      342153.0 |          Primary |    True
              32 |     406789.35 |        Secondary |    True
              34 |     452416.45 |          Primary |   False
              12 |      445530.0 | Higher Education |   False

(18 rows)
```

FIGURE 31:QUERY3 CASSANDRA

Case4: Show average income of total customers

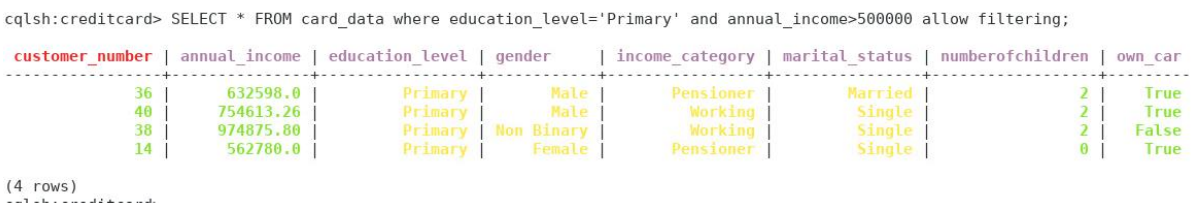


```
 system.avg(annual_income) | system.count(customer_number)
---------------------------+-------------------------------
                 519607.58 |                            44

(1 rows)
```

FIGURE 32:QUERY4 CASSANDRA

Case 5: Show customer data who have primary level education and have annual income more than 500000.



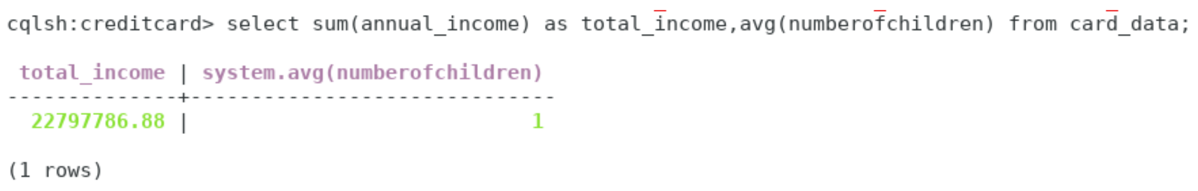
```
cqlsh:creditcard> SELECT * FROM card_data where education_level='Primary' and annual_income>500000 allow filtering;

 customer_number | annual_income | education_level | gender     | income_category | marital_status | numberofchildren | own_car
-----------------+---------------+-----------------+------------+-----------------+----------------+------------------+---------
              36 |      632598.0 |         Primary |       Male |       Pensioner |        Married |                2 |    True
              40 |     754613.26 |         Primary |       Male |         Working |         Single |                2 |    True
              38 |     974875.80 |         Primary | Non Binary |         Working |         Single |                2 |   False
              14 |      562780.0 |         Primary |     Female |       Pensioner |         Single |                0 |    True

(4 rows)
```

FIGURE 33:QUERY5 CASSANDRA

Case 6: show total annual incomes of customers' and average number of kid.



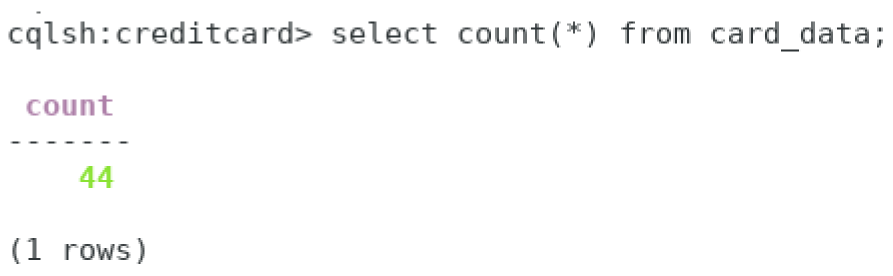
```
cqlsh:creditcard> select sum(annual_income) as total_income,avg(numberofchildren) from card_data;

 total_income | system.avg(numberofchildren)
--------------+------------------------------
  22797786.88 |                            1

(1 rows)
```

FIGURE 34:QUERY6 CASSANDRA

Case 7: Show the number of customers



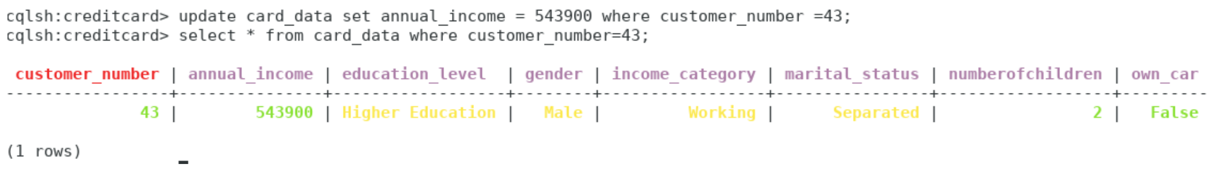
```
cqlsh:creditcard> select count(*) from card_data;

 count
-------
    44

(1 rows)
```

FIGURE 35:QUERY7 CASSANDRA

Case 8: Update customer number 43, change annual income to 543900



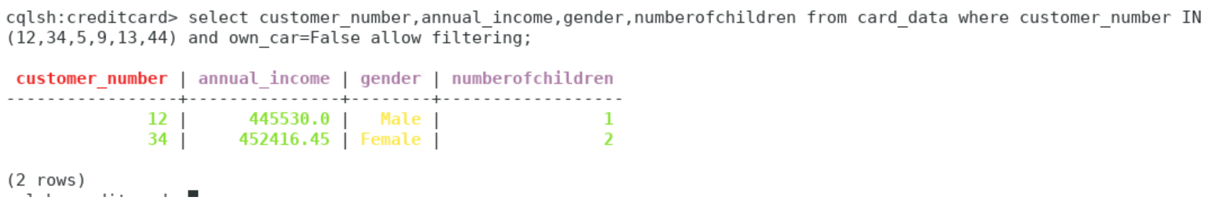
```
cqlsh:creditcard> update card_data set annual_income = 543900 where customer_number =43;
cqlsh:creditcard> select * from card_data where customer_number=43;

 customer_number | annual_income | education_level  | gender | income_category | marital_status | numberofchildren | own_car
-----------------+---------------+------------------+--------+-----------------+----------------+------------------+---------
              43 |        543900 | Higher Education |   Male |         Working |      Separated |                2 |   False

(1 rows)
```

FIGURE 36:QUERY8 CASSANDRA

Case 9: show only customers who have customer id (12,34,5,9,13,44) and own_car=False



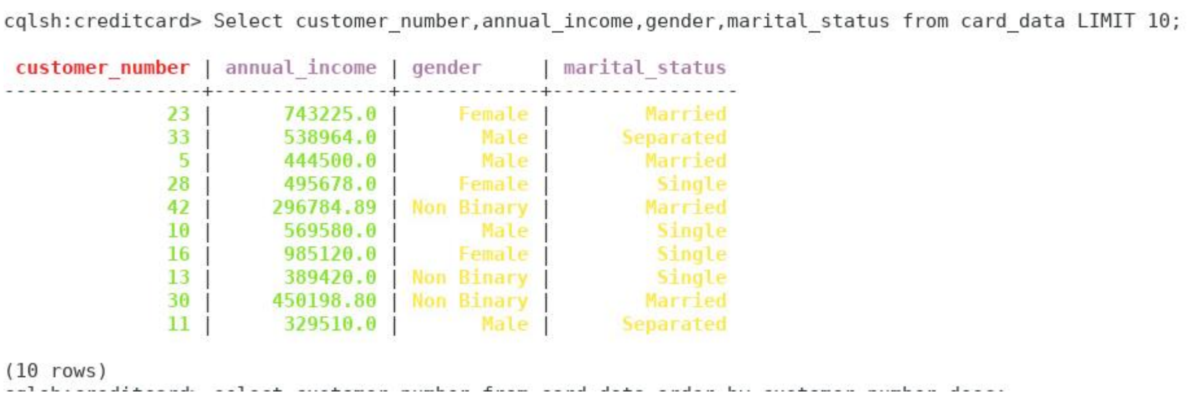
```
cqlsh:creditcard> select customer_number,annual_income,gender,numberofchildren from card_data where customer_number IN
(12,34,5,9,13,44) and own_car=False allow filtering;

 customer_number | annual_income | gender | numberofchildren
-----------------+---------------+--------+------------------
              12 |      445530.0 |   Male |                1
              34 |     452416.45 | Female |                2

(2 rows)
```

FIGURE 37:QUERY9 CASSANDRA

CASE 10: SHOW ONLY 10 CUSTOMERS



```
cqlsh:creditcard> Select customer_number,annual_income,gender,marital_status from card_data LIMIT 10;

 customer_number | annual_income | gender     | marital_status
-----------------+---------------+------------+----------------
              23 |      743225.0 |     Female |        Married
              33 |      538964.0 |       Male |      Separated
               5 |      444500.0 |       Male |        Married
              28 |      495678.0 |     Female |         Single
              42 |     296784.89 | Non Binary |        Married
              10 |      569580.0 |       Male |         Single
              16 |      985120.0 |     Female |         Single
              13 |      389420.0 | Non Binary |         Single
              30 |     450198.80 | Non Binary |        Married
              11 |      329510.0 |       Male |      Separated

(10 rows)
```

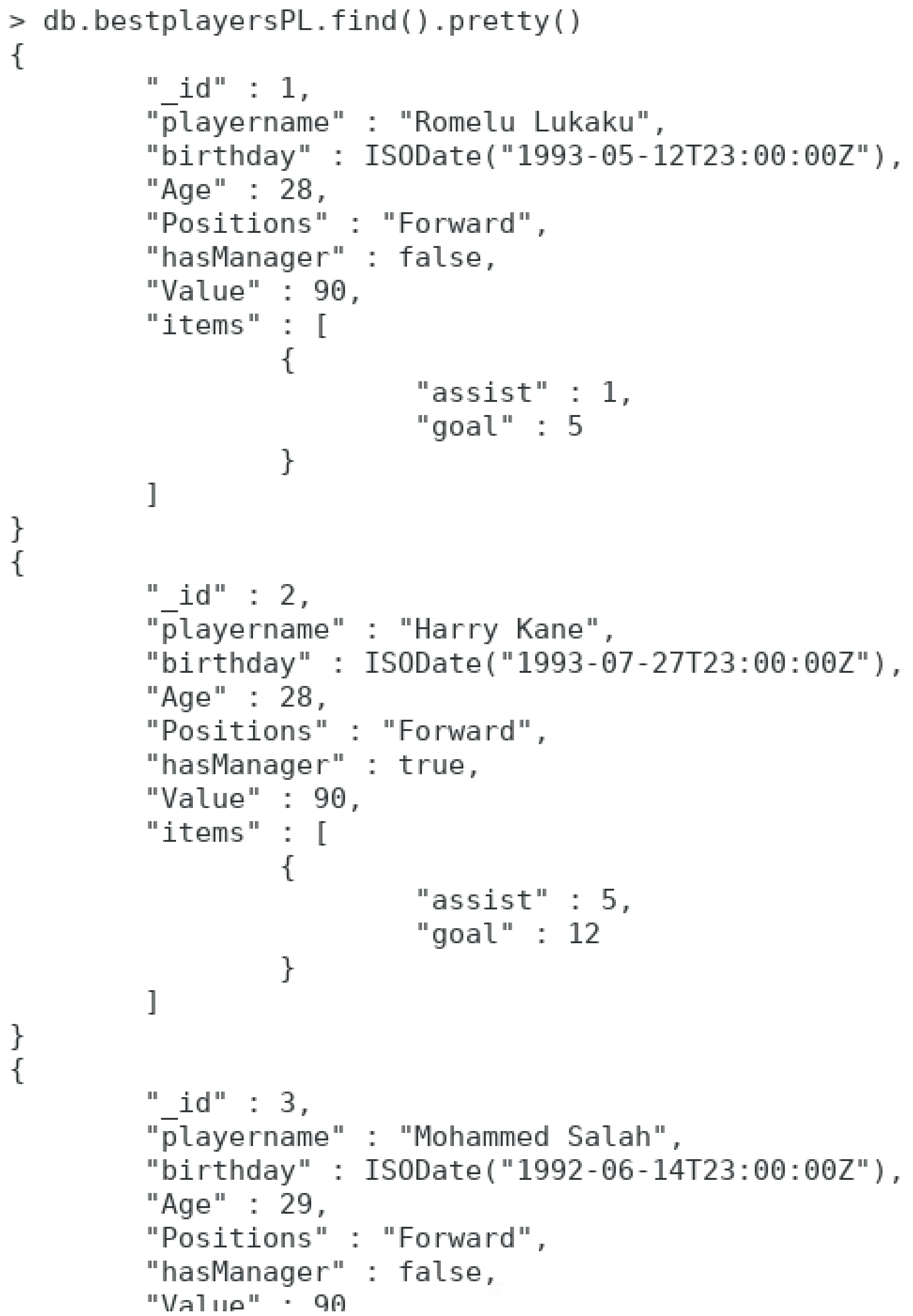
FIGURE 38:QUERY10 CASSANDRA

4. MongoDB definition and Coursework Implementation

Each record in MongoDB is represented as a document. These documents are saved in the JSON format. The table structure is collection here, the row structure is document, and the column structure is field, as is familiar to anyone who have worked with relational databases before.

4.1 Coursework Implementation

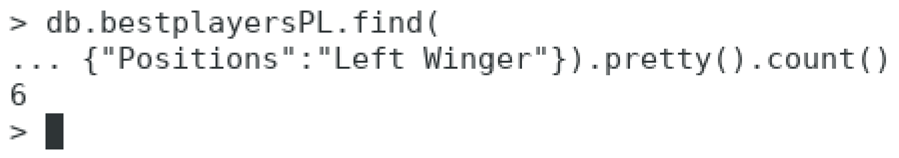
I saved the data of the 30 most valuable football players in the English premier league using all data types in mongo DB.



```
> db.bestplayersPL.find().pretty()
{
        "_id" : 1,
        "playername" : "Romelu Lukaku",
        "birthday" : ISODate("1993-05-12T23:00:00Z"),
        "Age" : 28,
        "Positions" : "Forward",
        "hasManager" : false,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 1,
                        "goal" : 5
                }
        ]
}
{
        "_id" : 2,
        "playername" : "Harry Kane",
        "birthday" : ISODate("1993-07-27T23:00:00Z"),
        "Age" : 28,
        "Positions" : "Forward",
        "hasManager" : true,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 5,
                        "goal" : 12
                }
        ]
}
{
        "_id" : 3,
        "playername" : "Mohammed Salah",
        "birthday" : ISODate("1992-06-14T23:00:00Z"),
        "Age" : 29,
        "Positions" : "Forward",
        "hasManager" : false,
        "Value" : 90
```

FIGURE 39:DATA STRUCTURE MONGODB

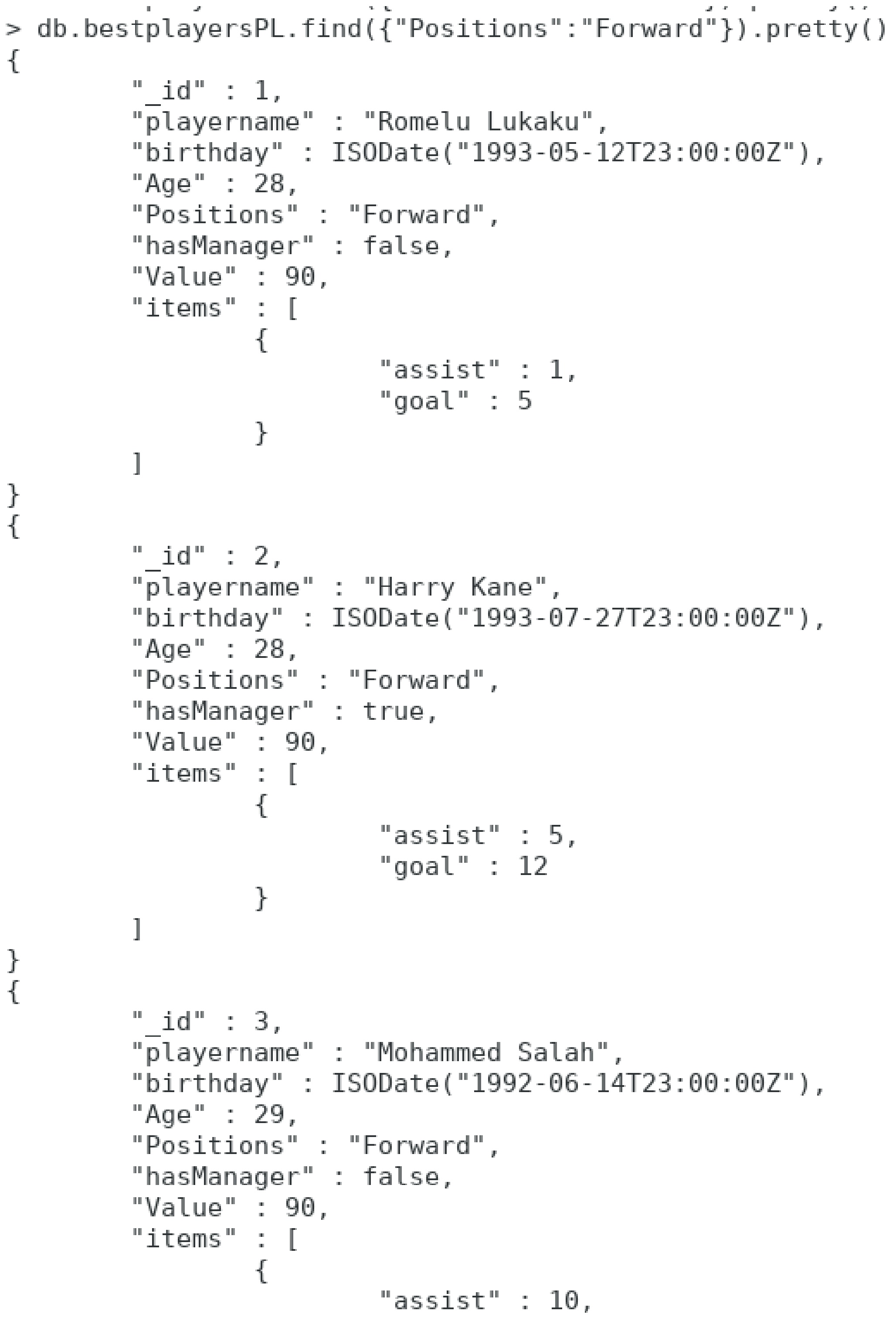
Case 1: Find how many left winger players in database



```
> db.bestplayersPL.find(
... {"Positions":"Left Winger"}).pretty().count()
6
> 
```

FIGURE 40:QUERY MONGO DB

Case 2: List the Forward players



```
> db.bestplayersPL.find({"Positions":"Forward"}).pretty()
{
        "_id" : 1,
        "playername" : "Romelu Lukaku",
        "birthday" : ISODate("1993-05-12T23:00:00Z"),
        "Age" : 28,
        "Positions" : "Forward",
        "hasManager" : false,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 1,
                        "goal" : 5
                }
        ]
}
{
        "_id" : 2,
        "playername" : "Harry Kane",
        "birthday" : ISODate("1993-07-27T23:00:00Z"),
        "Age" : 28,
        "Positions" : "Forward",
        "hasManager" : true,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 5,
                        "goal" : 12
                }
        ]
}
{
        "_id" : 3,
        "playername" : "Mohammed Salah",
        "birthday" : ISODate("1992-06-14T23:00:00Z"),
        "Age" : 29,
        "Positions" : "Forward",
        "hasManager" : false,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 10,
```

FIGURE 41:QUERY2 MONGO DB

Case 3: Find the player whose market value is greater than 85 million

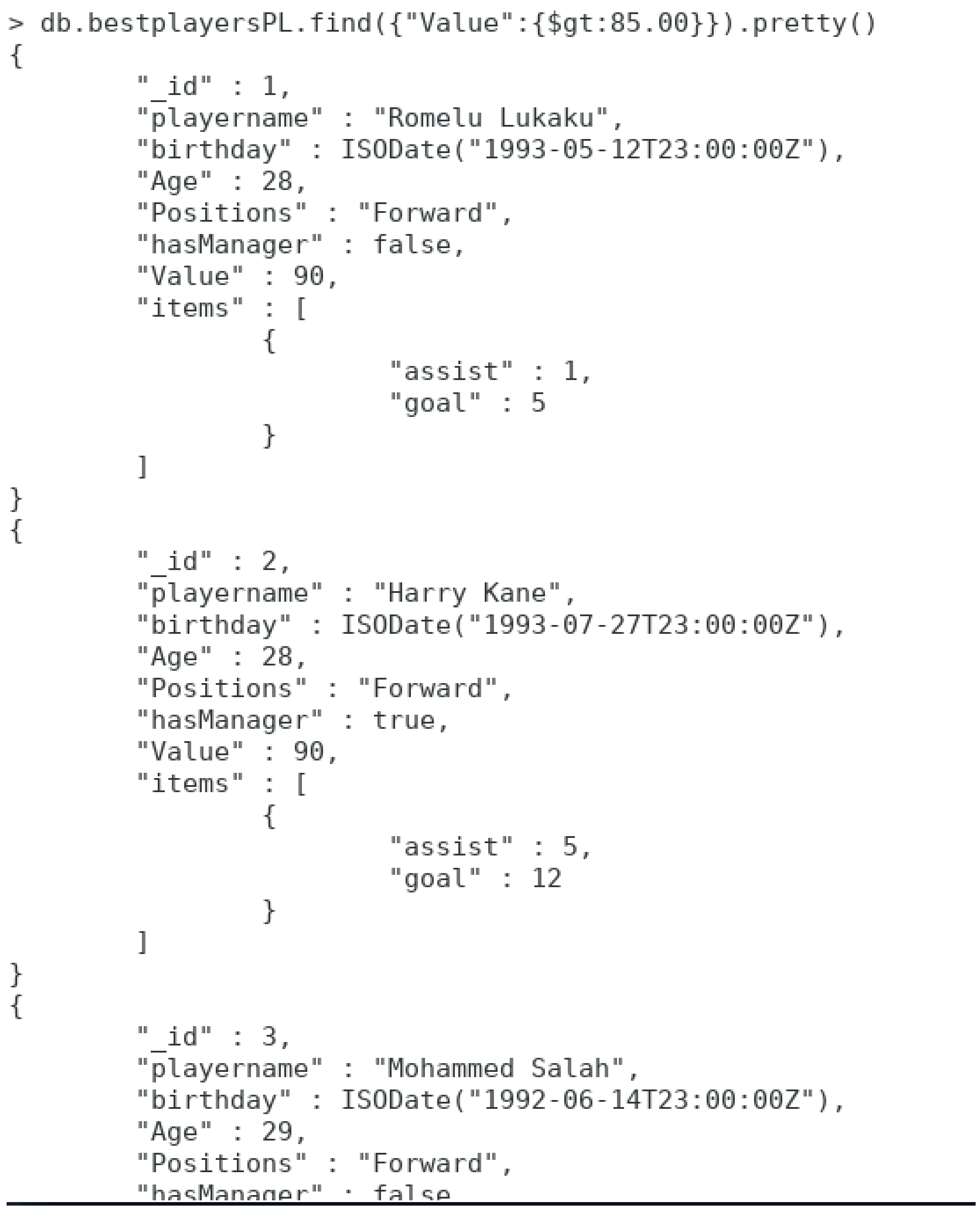
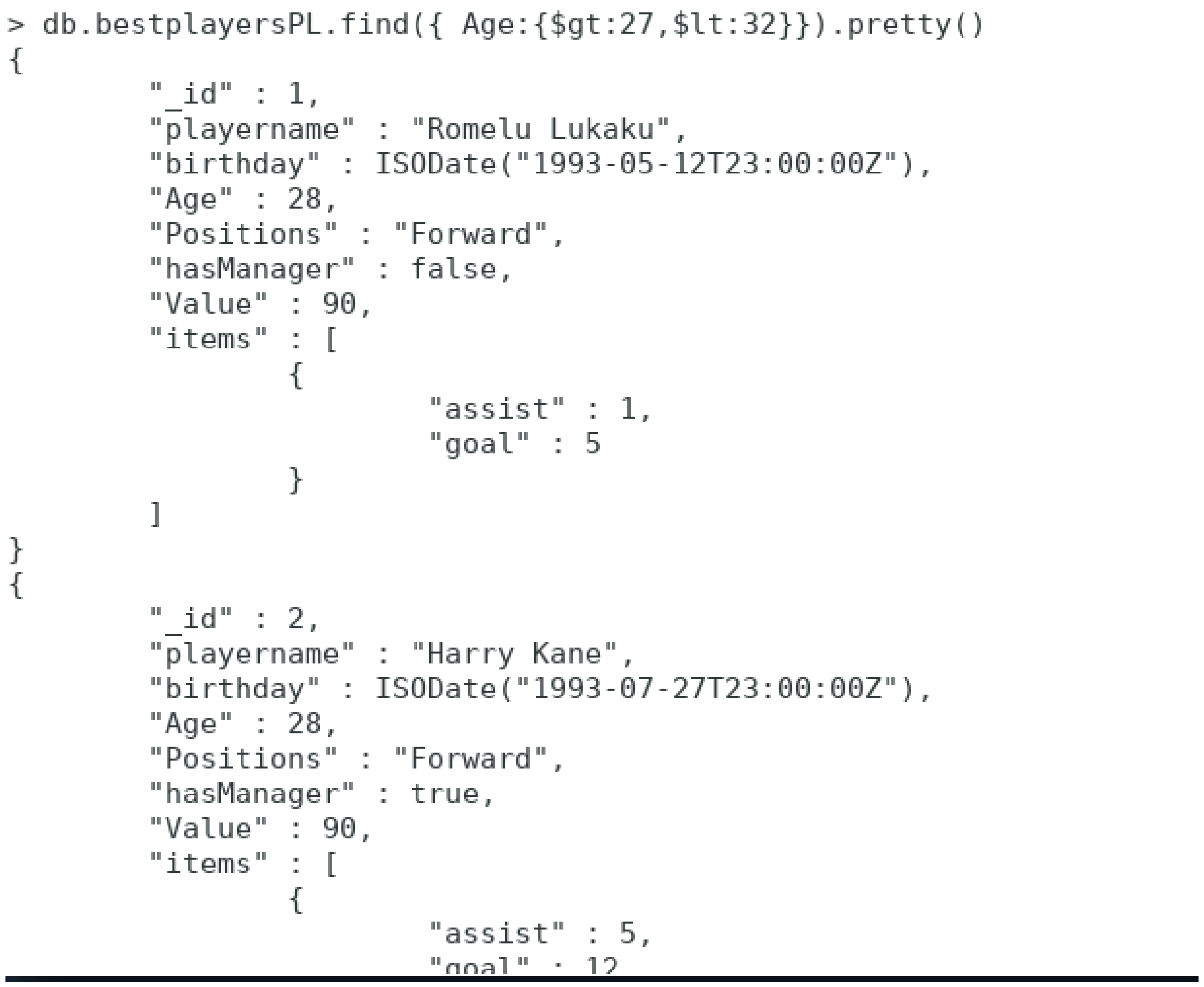

```
> db.bestplayersPL.find({"Value":{$gt:85.00}}).pretty()
{
        "_id" : 1,
        "playername" : "Romelu Lukaku",
        "birthday" : ISODate("1993-05-12T23:00:00Z"),
        "Age" : 28,
        "Positions" : "Forward",
        "hasManager" : false,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 1,
                        "goal" : 5
                }
        ]
}
{
        "_id" : 2,
        "playername" : "Harry Kane",
        "birthday" : ISODate("1993-07-27T23:00:00Z"),
        "Age" : 28,
        "Positions" : "Forward",
        "hasManager" : true,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 5,
                        "goal" : 12
                }
        ]
}
{
        "_id" : 3,
        "playername" : "Mohammed Salah",
        "birthday" : ISODate("1992-06-14T23:00:00Z"),
        "Age" : 29,
        "Positions" : "Forward",
        "hasManager" : false
```

FIGURE 42:QUERY3 MONGO DB

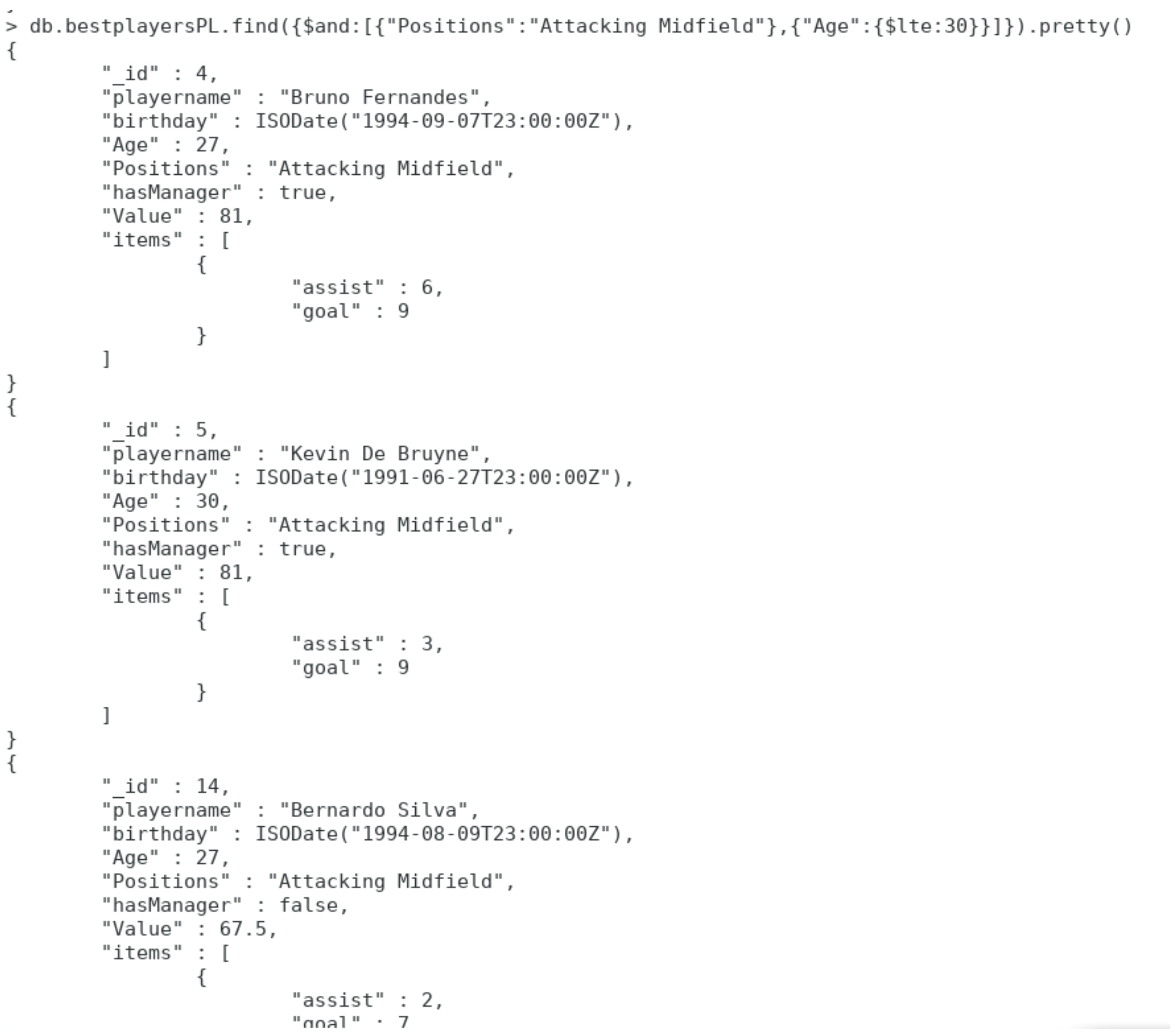
Case 4: Find the players whose age is between 27 and 32



```
> db.bestplayersPL.find({ Age:{$gt:27,$lt:32}}).pretty()
{
        "_id" : 1,
        "playername" : "Romelu Lukaku",
        "birthday" : ISODate("1993-05-12T23:00:00Z"),
        "Age" : 28,
        "Positions" : "Forward",
        "hasManager" : false,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 1,
                        "goal" : 5
                }
        ]
}
{
        "_id" : 2,
        "playername" : "Harry Kane",
        "birthday" : ISODate("1993-07-27T23:00:00Z"),
        "Age" : 28,
        "Positions" : "Forward",
        "hasManager" : true,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 5,
                        "goal" : 12
```

FIGURE 43:QUERY4 MONGO DB

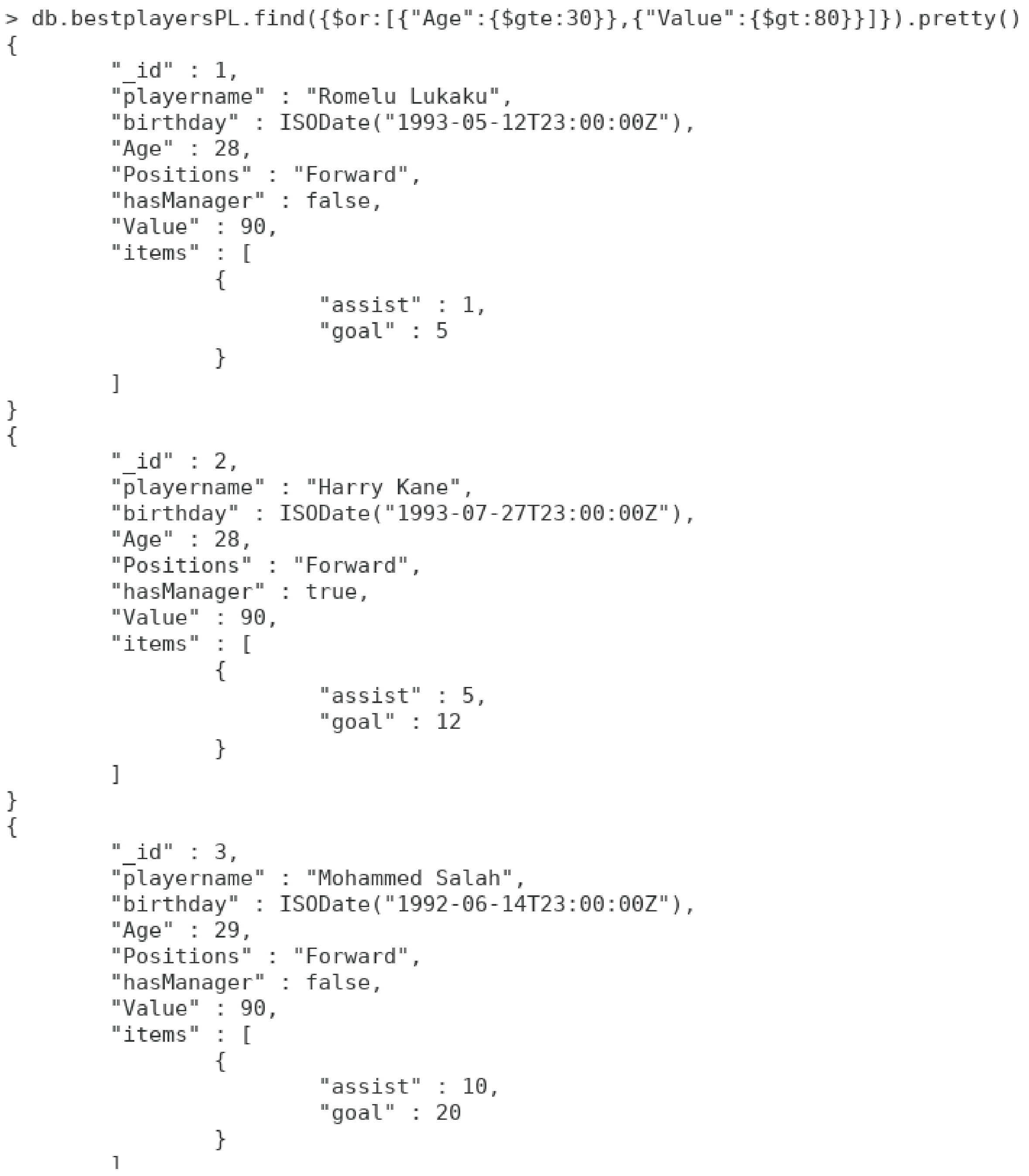
Case 5: Find the attacking midfield players whose age is equal and less than 30



```
> db.bestplayersPL.find({$and:[{"Positions":"Attacking Midfield"},{"Age":{$lte:30}}]}).pretty()
{
        "_id" : 4,
        "playername" : "Bruno Fernandes",
        "birthday" : ISODate("1994-09-07T23:00:00Z"),
        "Age" : 27,
        "Positions" : "Attacking Midfield",
        "hasManager" : true,
        "Value" : 81,
        "items" : [
                {
                        "assist" : 6,
                        "goal" : 9
                }
        ]
}
{
        "_id" : 5,
        "playername" : "Kevin De Bruyne",
        "birthday" : ISODate("1991-06-27T23:00:00Z"),
        "Age" : 30,
        "Positions" : "Attacking Midfield",
        "hasManager" : true,
        "Value" : 81,
        "items" : [
                {
                        "assist" : 3,
                        "goal" : 9
                }
        ]
}
{
        "_id" : 14,
        "playername" : "Bernardo Silva",
        "birthday" : ISODate("1994-08-09T23:00:00Z"),
        "Age" : 27,
        "Positions" : "Attacking Midfield",
        "hasManager" : false,
        "Value" : 67.5,
        "items" : [
                {
                        "assist" : 2,
                        "goal" : 7
```

FIGURE 44:QUEY5 MONGO DB

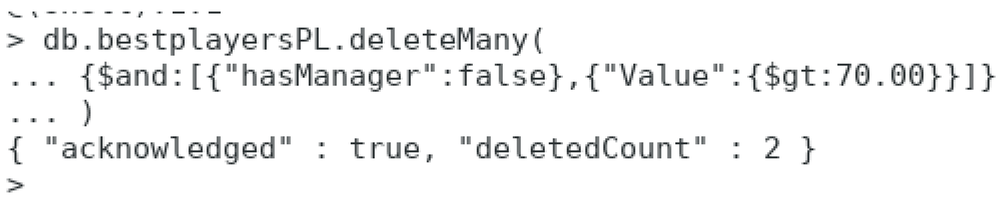
Case 6: Find the players whose market value is greater than 80 million and older than 30 years old.



```
> db.bestplayersPL.find({$or:[{"Age":{$gte:30}},{"Value":{$gt:80}}]}).pretty()
{
        "_id" : 1,
        "playername" : "Romelu Lukaku",
        "birthday" : ISODate("1993-05-12T23:00:00Z"),
        "Age" : 28,
        "Positions" : "Forward",
        "hasManager" : false,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 1,
                        "goal" : 5
                }
        ]
}
{
        "_id" : 2,
        "playername" : "Harry Kane",
        "birthday" : ISODate("1993-07-27T23:00:00Z"),
        "Age" : 28,
        "Positions" : "Forward",
        "hasManager" : true,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 5,
                        "goal" : 12
                }
        ]
}
{
        "_id" : 3,
        "playername" : "Mohammed Salah",
        "birthday" : ISODate("1992-06-14T23:00:00Z"),
        "Age" : 29,
        "Positions" : "Forward",
        "hasManager" : false,
        "Value" : 90,
        "items" : [
                {
                        "assist" : 10,
                        "goal" : 20
                }
        ]
```

FIGURE 45:QUERY6 MONGO DB

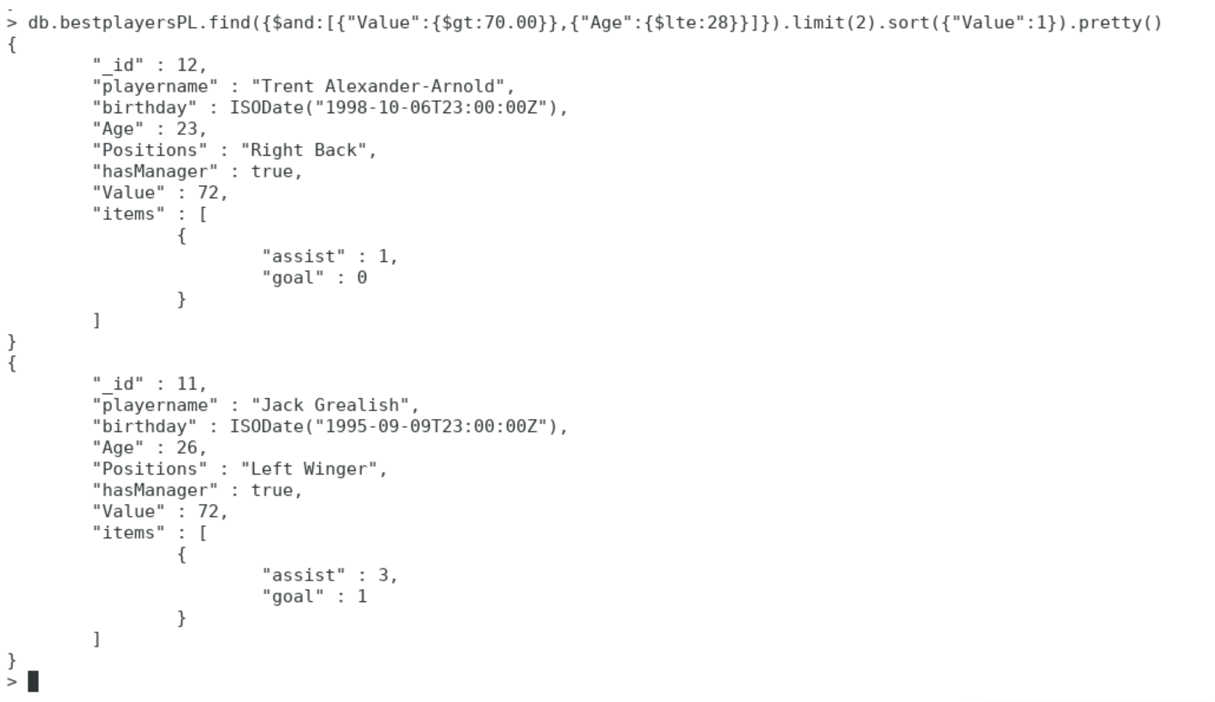
Case 7: Delete the players who hasn't got a manager and market value is greater than 70 million.



```
> db.bestplayersPL.deleteMany(
... {$and:[{"hasManager":false},{"Value":{$gt:70.00}}]}
... )
{ "acknowledged" : true, "deletedCount" : 2 }
>
```

FIGURE 46:QUERY7 MONGO DB

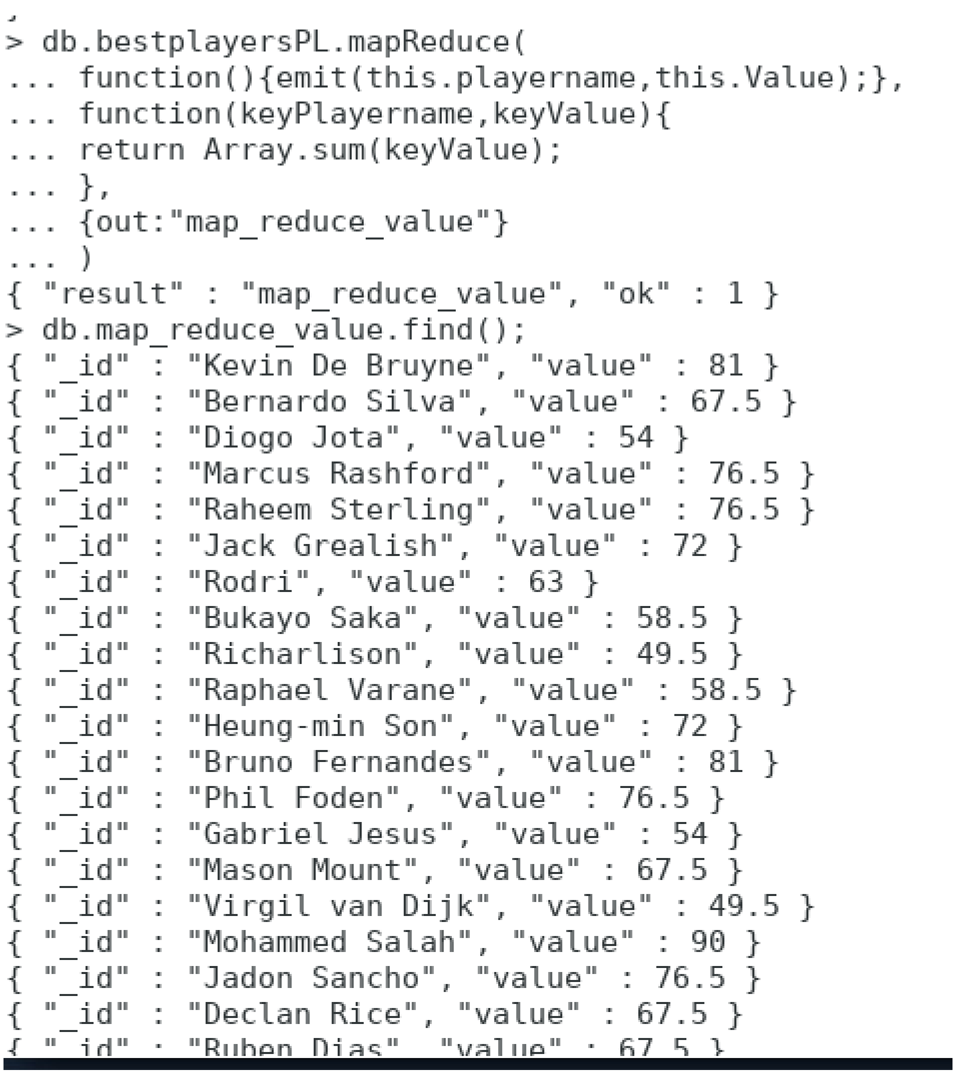
Case 8: Find the players who have more than 70 million market value and less than equal 28 years old. Sort it by market value.



```
> db.bestplayersPL.find({$and:[{"Value":{$gt:70.00}},{"Age":{$lte:28}}]}).limit(2).sort({"Value":1}).pretty()
{
        "_id" : 12,
        "playername" : "Trent Alexander-Arnold",
        "birthday" : ISODate("1998-10-06T23:00:00Z"),
        "Age" : 23,
        "Positions" : "Right Back",
        "hasManager" : true,
        "Value" : 72,
        "items" : [
                {
                        "assist" : 1,
                        "goal" : 0
                }
        ]
}
{
        "_id" : 11,
        "playername" : "Jack Grealish",
        "birthday" : ISODate("1995-09-09T23:00:00Z"),
        "Age" : 26,
        "Positions" : "Left Winger",
        "hasManager" : true,
        "Value" : 72,
        "items" : [
                {
                        "assist" : 3,
                        "goal" : 1
                }
        ]
}
>
```

FIGURE 47:QUERY8 MONGO DB

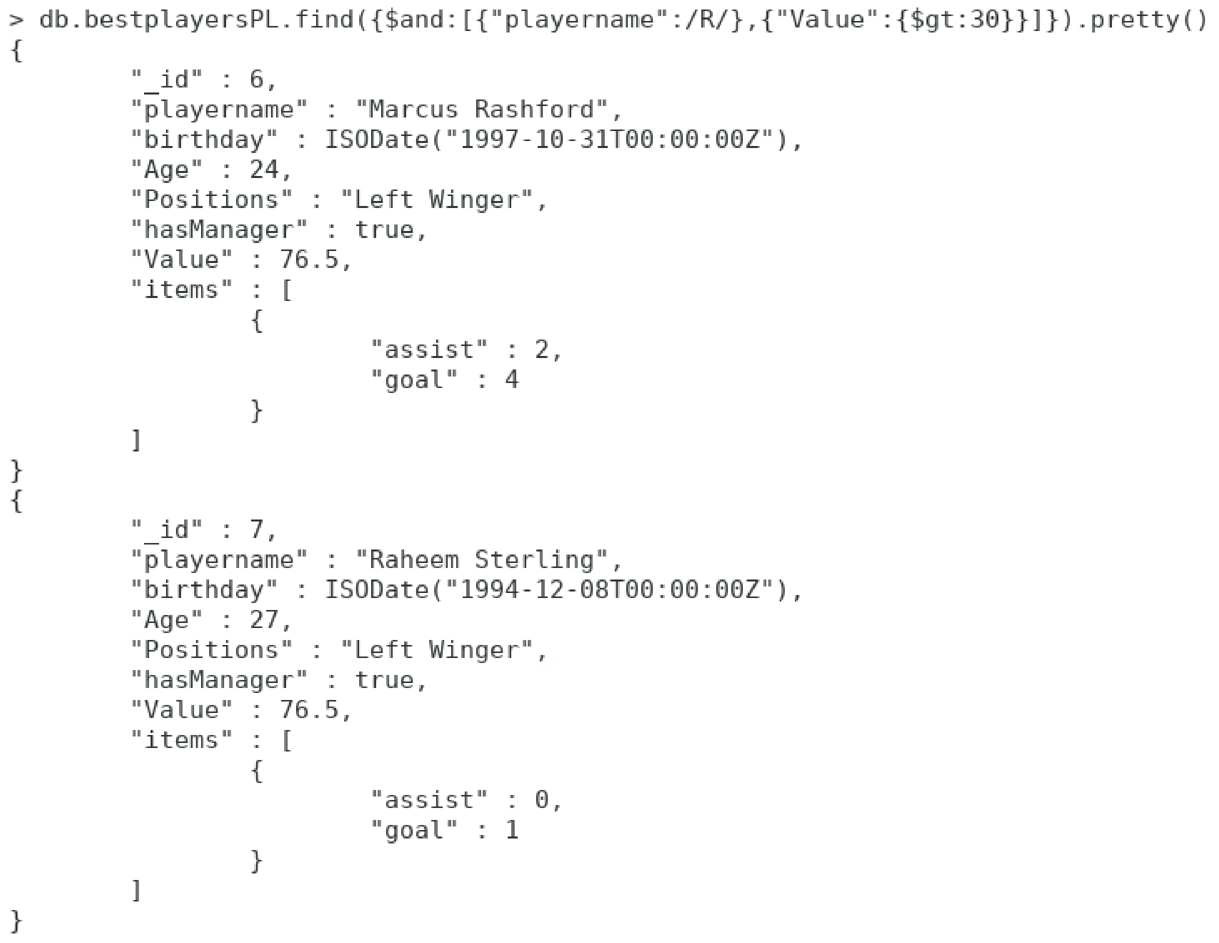
Case 9: Find the player and total sum of each player. (I used mapreduce)



```
,
> db.bestplayersPL.mapReduce(
... function(){emit(this.playername,this.Value);},
... function(keyPlayername,keyValue){
... return Array.sum(keyValue);
... },
... {out:"map_reduce_value"}
... )
{ "result" : "map_reduce_value", "ok" : 1 }
> db.map_reduce_value.find();
{ "_id" : "Kevin De Bruyne", "value" : 81 }
{ "_id" : "Bernardo Silva", "value" : 67.5 }
{ "_id" : "Diogo Jota", "value" : 54 }
{ "_id" : "Marcus Rashford", "value" : 76.5 }
{ "_id" : "Raheem Sterling", "value" : 76.5 }
{ "_id" : "Jack Grealish", "value" : 72 }
{ "_id" : "Rodri", "value" : 63 }
{ "_id" : "Bukayo Saka", "value" : 58.5 }
{ "_id" : "Richarlison", "value" : 49.5 }
{ "_id" : "Raphael Varane", "value" : 58.5 }
{ "_id" : "Heung-min Son", "value" : 72 }
{ "_id" : "Bruno Fernandes", "value" : 81 }
{ "_id" : "Phil Foden", "value" : 76.5 }
{ "_id" : "Gabriel Jesus", "value" : 54 }
{ "_id" : "Mason Mount", "value" : 67.5 }
{ "_id" : "Virgil van Dijk", "value" : 49.5 }
{ "_id" : "Mohammed Salah", "value" : 90 }
{ "_id" : "Jadon Sancho", "value" : 76.5 }
{ "_id" : "Declan Rice", "value" : 67.5 }
{ "_id" : "Ruben Dias", "value" : 67.5 }
```

FIGURE 48:QUERY9 MONGO DB

Case 10: Find the player’s market value is greater than 30 million and name or surname start with letter ‘R’.



```
> db.bestplayersPL.find({$and:[{"playername":/R/},{"Value":{$gt:30}}]}).pretty()
{
        "_id" : 6,
        "playername" : "Marcus Rashford",
        "birthday" : ISODate("1997-10-31T00:00:00Z"),
        "Age" : 24,
        "Positions" : "Left Winger",
        "hasManager" : true,
        "Value" : 76.5,
        "items" : [
                {
                        "assist" : 2,
                        "goal" : 4
                }
        ]
}
{
        "_id" : 7,
        "playername" : "Raheem Sterling",
        "birthday" : ISODate("1994-12-08T00:00:00Z"),
        "Age" : 27,
        "Positions" : "Left Winger",
        "hasManager" : true,
        "Value" : 76.5,
        "items" : [
                {
                        "assist" : 0,
                        "goal" : 1
                }
        ]
}
```

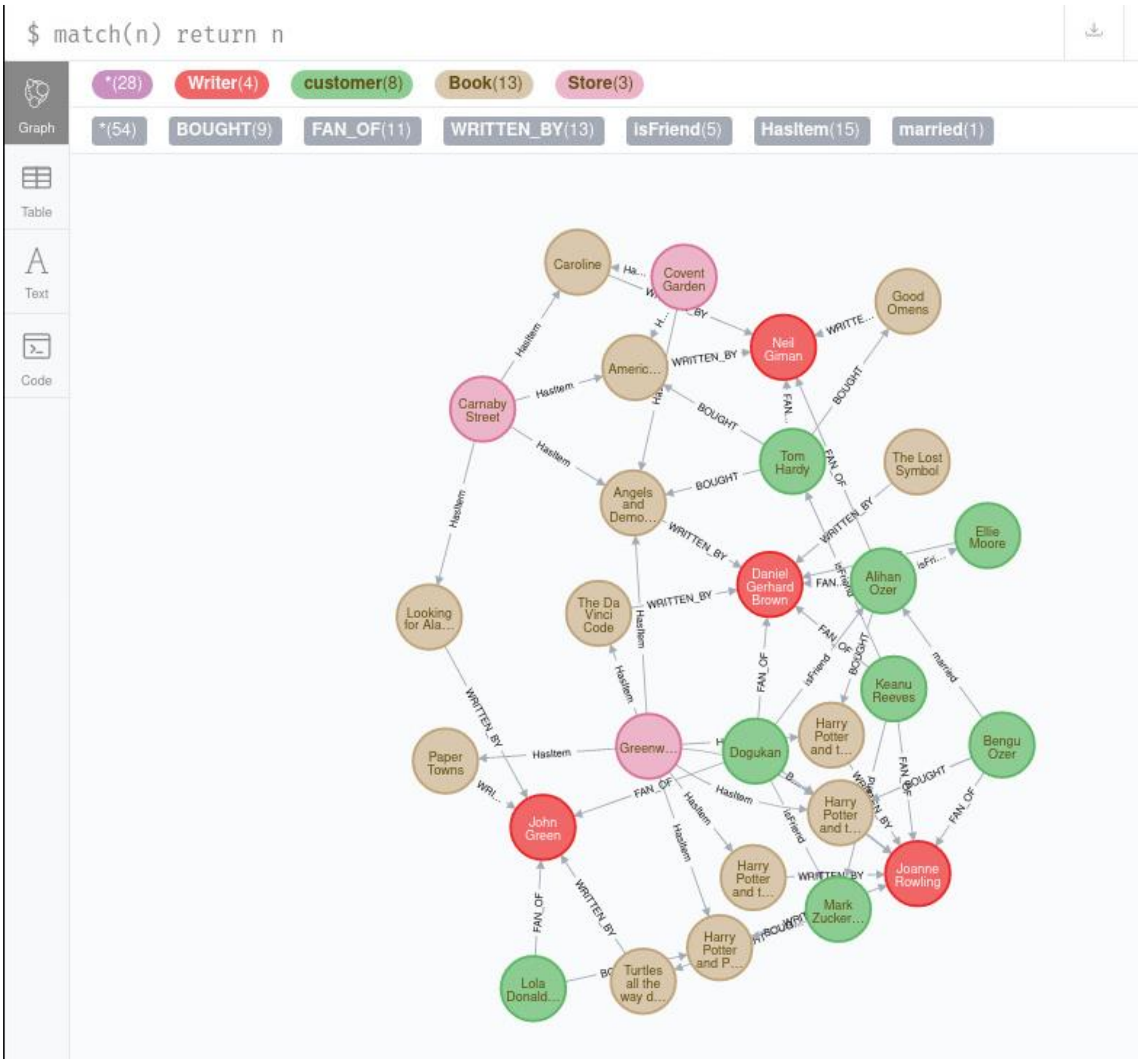
FIGURE 49:QUERY10 MONGO DB

5. NEO4J Definition and Coursework Implementation

Neo4j is a graphical database management system. It is an ACID compliant transactional database with native graphics storage and processing. Neo4j has 3 main components: Nodes, Relationship, Properties

5.1 Coursework Implementation

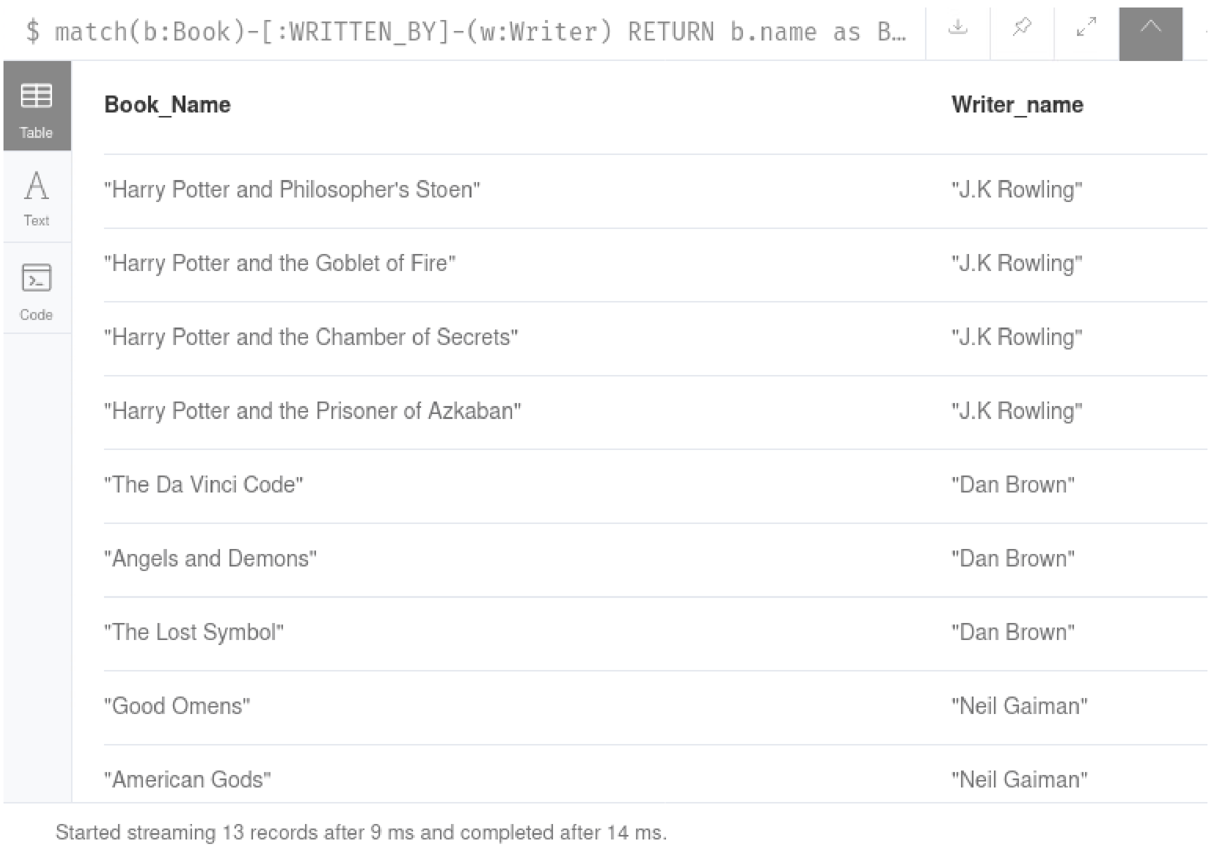
In the coursework database, I recorded the relationships between 8 customers, 13 books, 4 authors and 3 shops.



```
$ match(n) return n
```

FIGURE 50:NEO4J DATA STRUCTURE

Case 1: Show all the books and writers



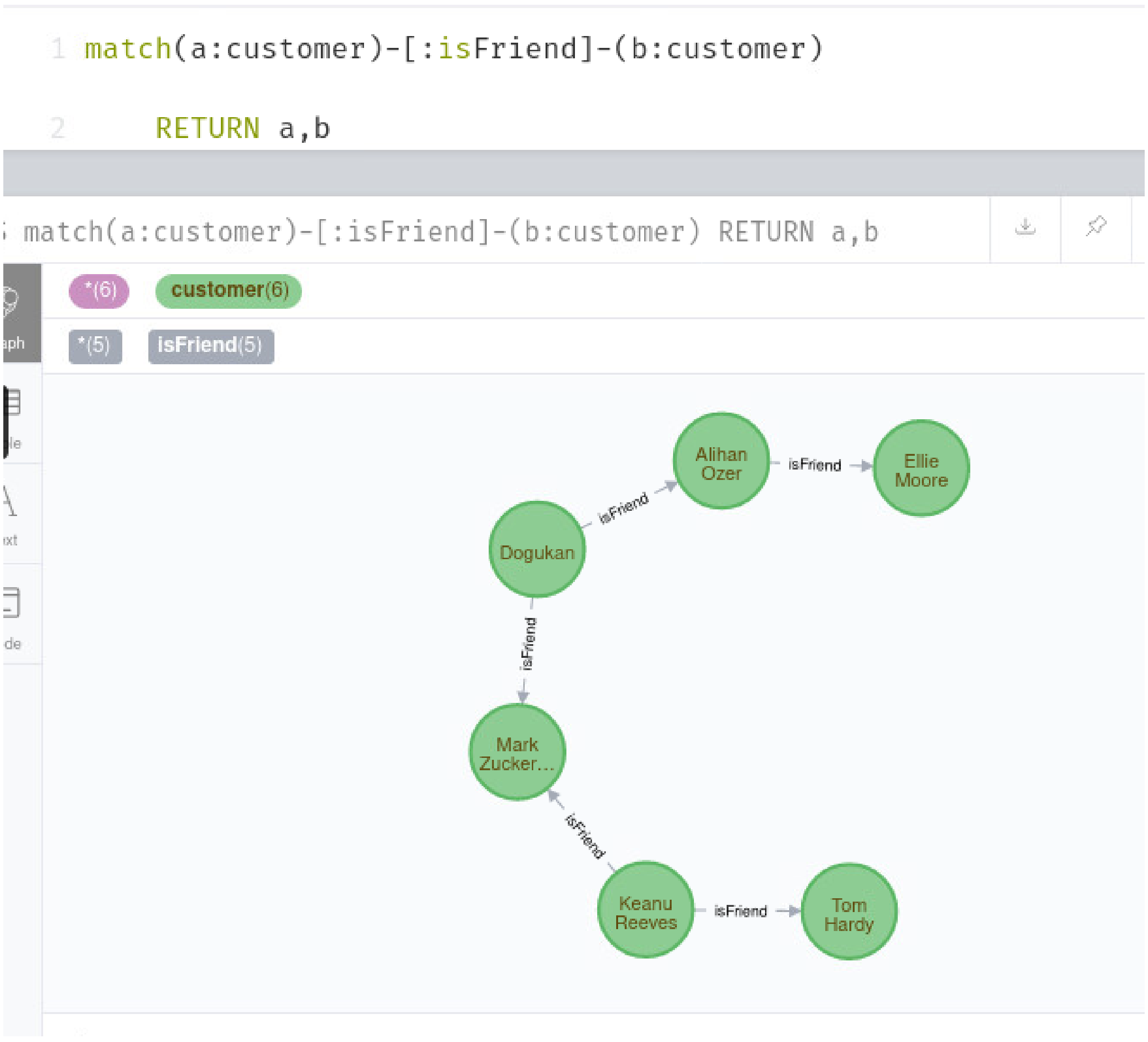
```
$ match(b:Book)-[:WRITTEN_BY]-(w:Writer) RETURN b.name as B…
```

Book_Name	Writer_name
"Harry Potter and Philosopher's Stoen"	"J.K Rowling"
"Harry Potter and the Goblet of Fire"	"J.K Rowling"
"Harry Potter and the Chamber of Secrets"	"J.K Rowling"
"Harry Potter and the Prisoner of Azkaban"	"J.K Rowling"
"The Da Vinci Code"	"Dan Brown"
"Angels and Demons"	"Dan Brown"
"The Lost Symbol"	"Dan Brown"
"Good Omens"	"Neil Gaiman"
"American Gods"	"Neil Gaiman"

Started streaming 13 records after 9 ms and completed after 14 ms.

FIGURE 51:QUERY 1 NEO4J

Case 2: Find all the friendship relations between customers

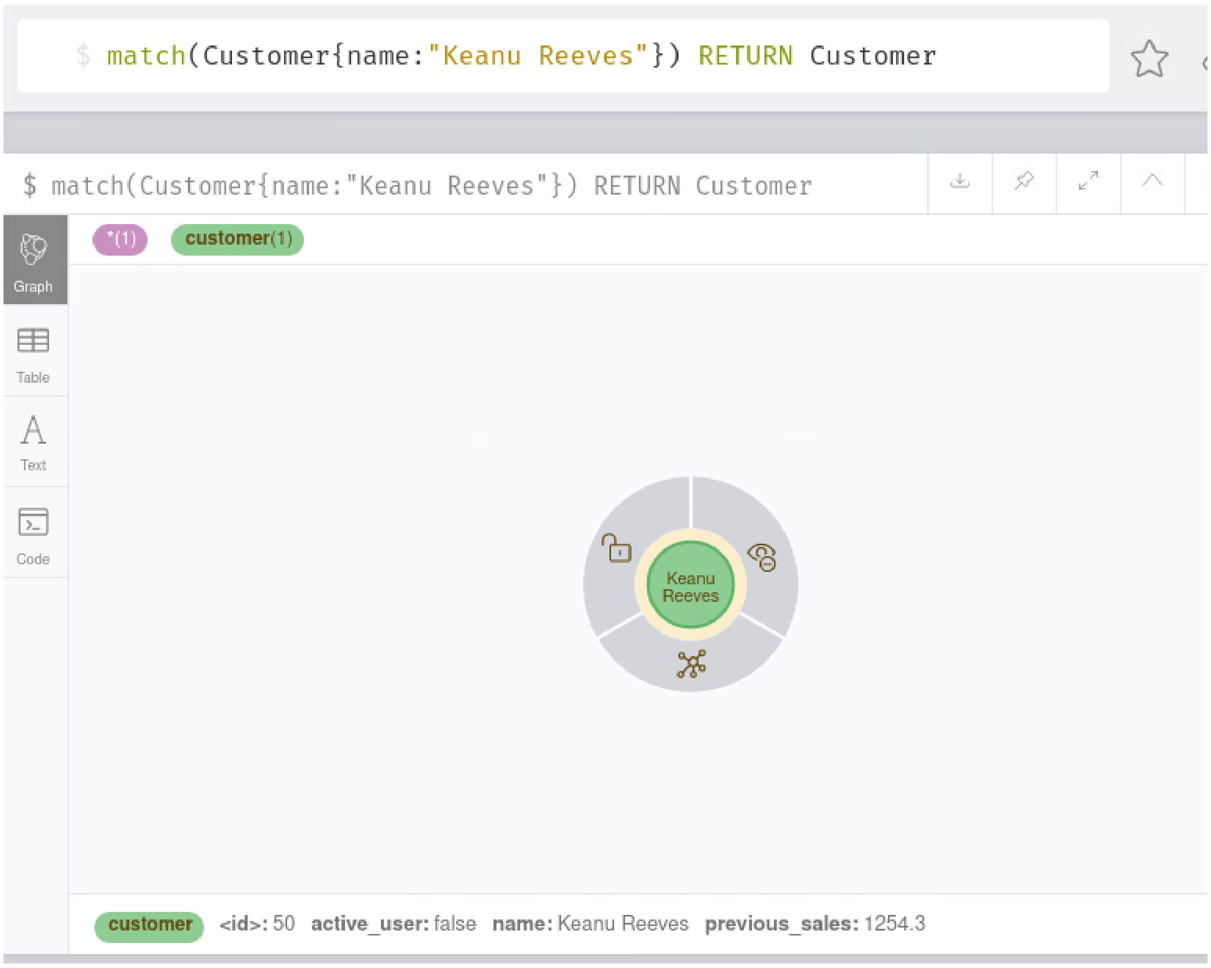


```
match(a:customer)-[:isFriend]-(b:customer)
    RETURN a,b
```

```
match(a:customer)-[:isFriend]-(b:customer) RETURN a,b
```

FIGURE 52:QUERY2 NEO4J

Case 3: Show the data of customer Keanu Reeves.



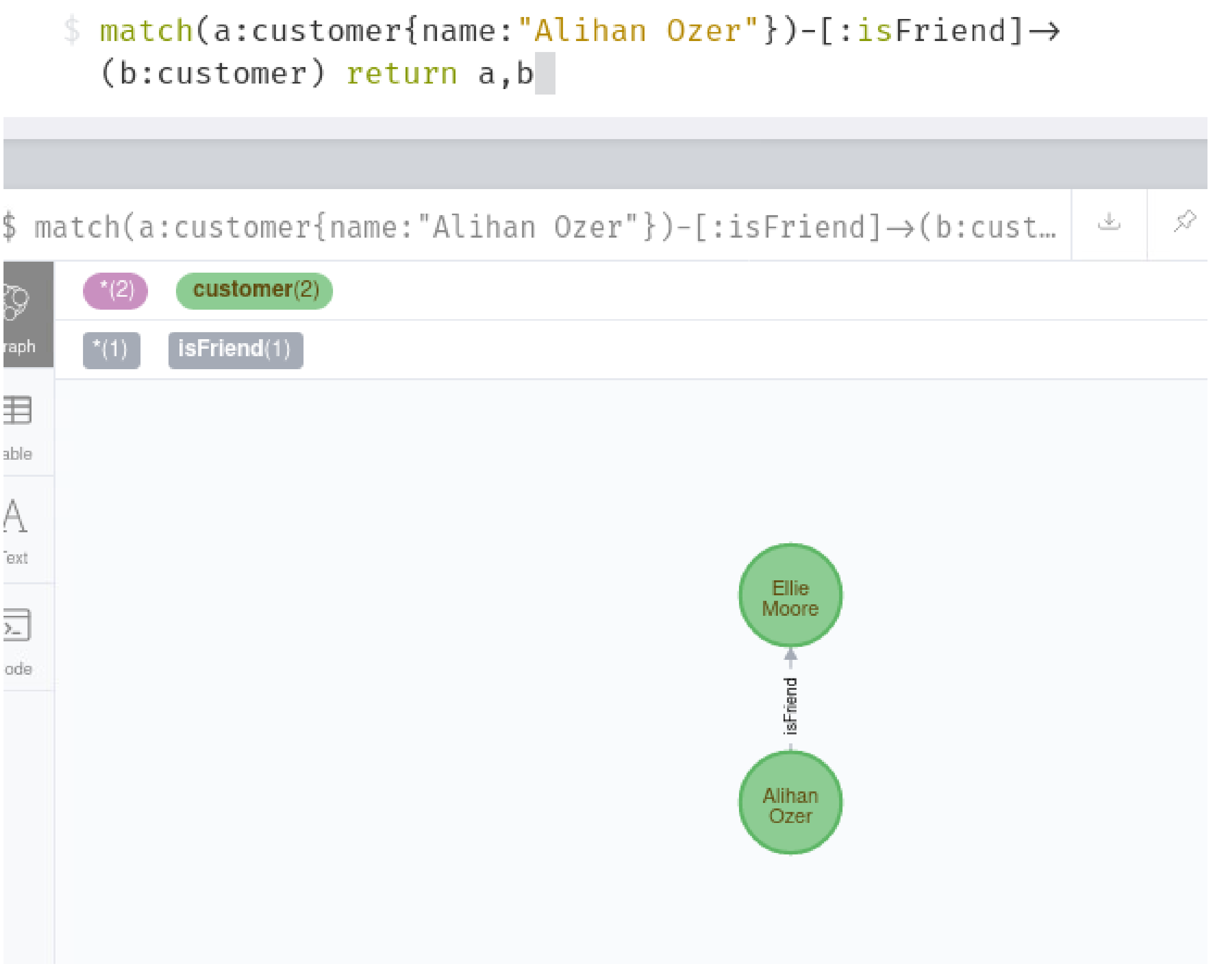
```
$ match(Customer{name:"Keanu Reeves"}) RETURN Customer
```

```
$ match(Customer{name:"Keanu Reeves"}) RETURN Customer
```

customer <id>: 50 active_user: false name: Keanu Reeves previous_sales: 1254.3

FIGURE 53:QUERY3 NEO4J

Case 4: Show the friends of Alihan Ozer.

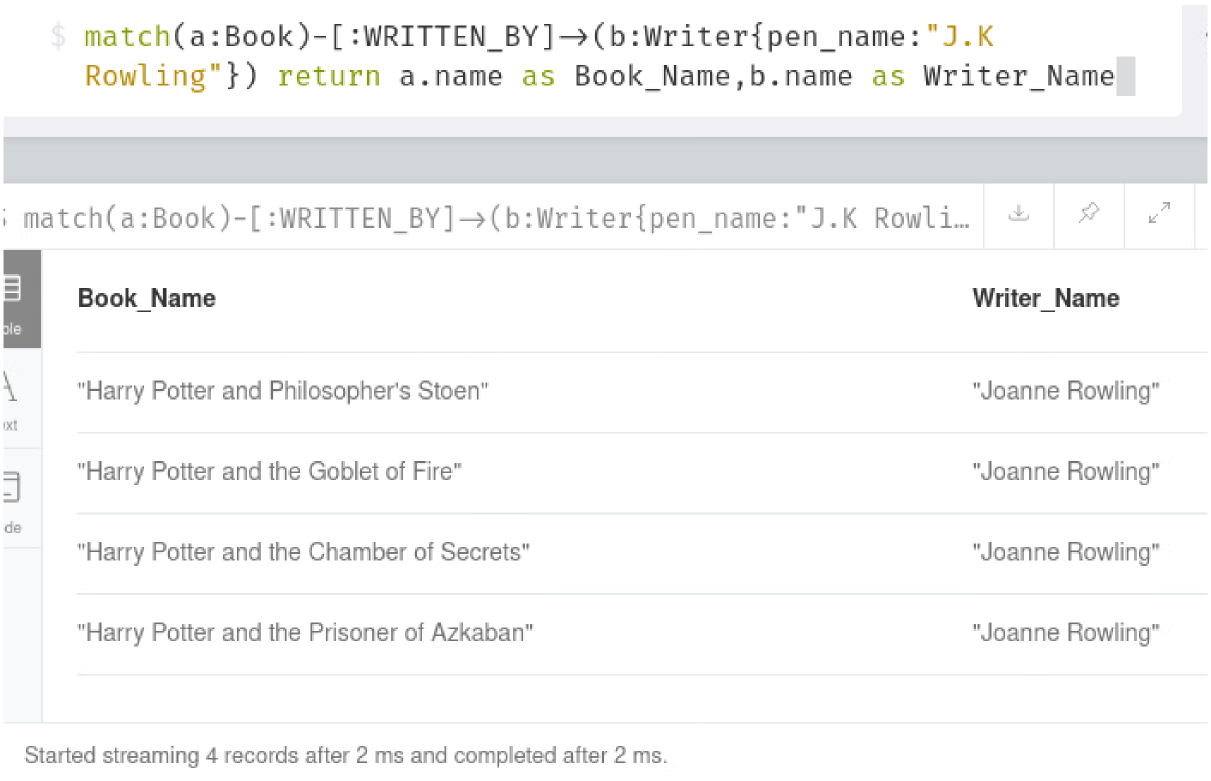


```
$ match(a:customer{name:"Alihan Ozer"})-[:isFriend]→
  (b:customer) return a,b
```

```
$ match(a:customer{name:"Alihan Ozer"})-[:isFriend]→(b:cust…
```

FIGURE 54:QUERY4 NEO4J

Case 5: Show the books of “J.K Rowling”



```
$ match(a:Book)-[:WRITTEN_BY]→(b:Writer{pen_name:"J.K
  Rowling"}) return a.name as Book_Name,b.name as Writer_Name
```

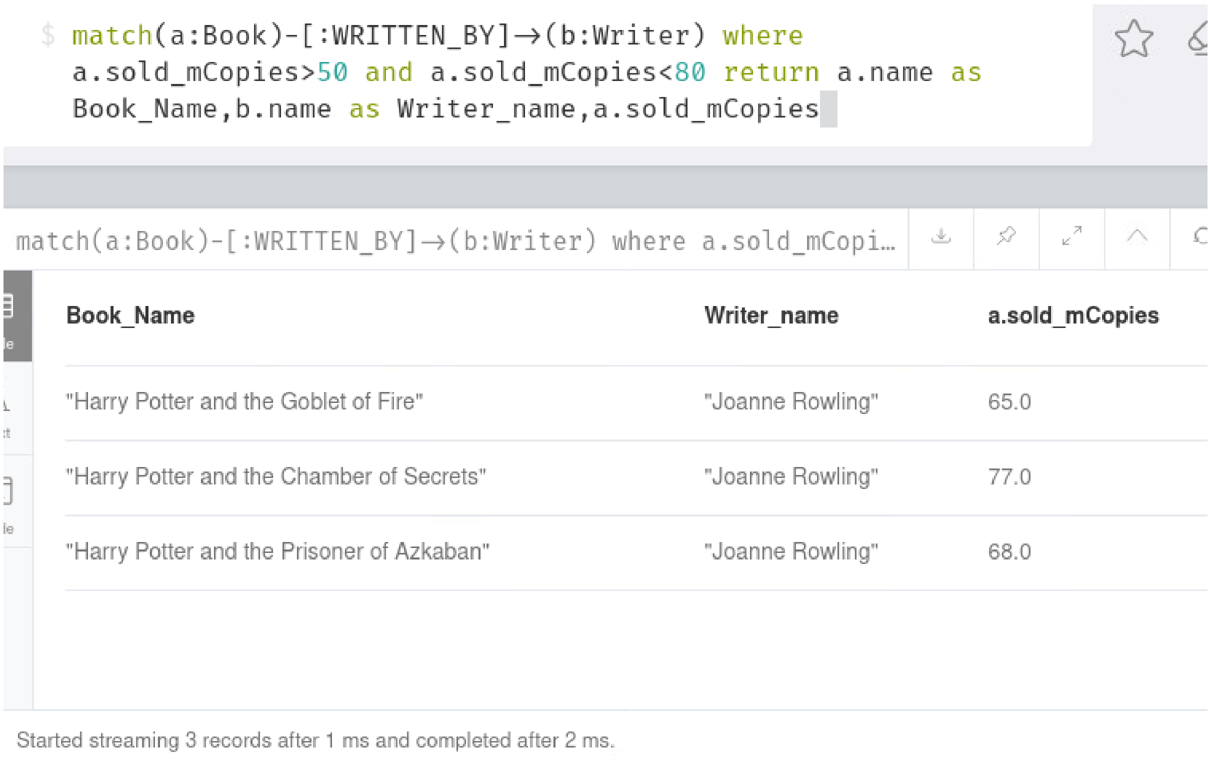
```
match(a:Book)-[:WRITTEN_BY]→(b:Writer{pen_name:"J.K Rowli…
```

Book_Name	Writer_Name
"Harry Potter and Philosopher's Stoen"	"Joanne Rowling"
"Harry Potter and the Goblet of Fire"	"Joanne Rowling"
"Harry Potter and the Chamber of Secrets"	"Joanne Rowling"
"Harry Potter and the Prisoner of Azkaban"	"Joanne Rowling"

Started streaming 4 records after 2 ms and completed after 2 ms.

FIGURE 55:QUERY5 NEO4J

Case 6: show the Books and writers that sold between 50 and 80 million copies.



```
$ match(a:Book)-[:WRITTEN_BY]→(b:Writer) where
  a.sold_mCopies>50 and a.sold_mCopies<80 return a.name as
  Book_Name,b.name as Writer_name,a.sold_mCopies
```

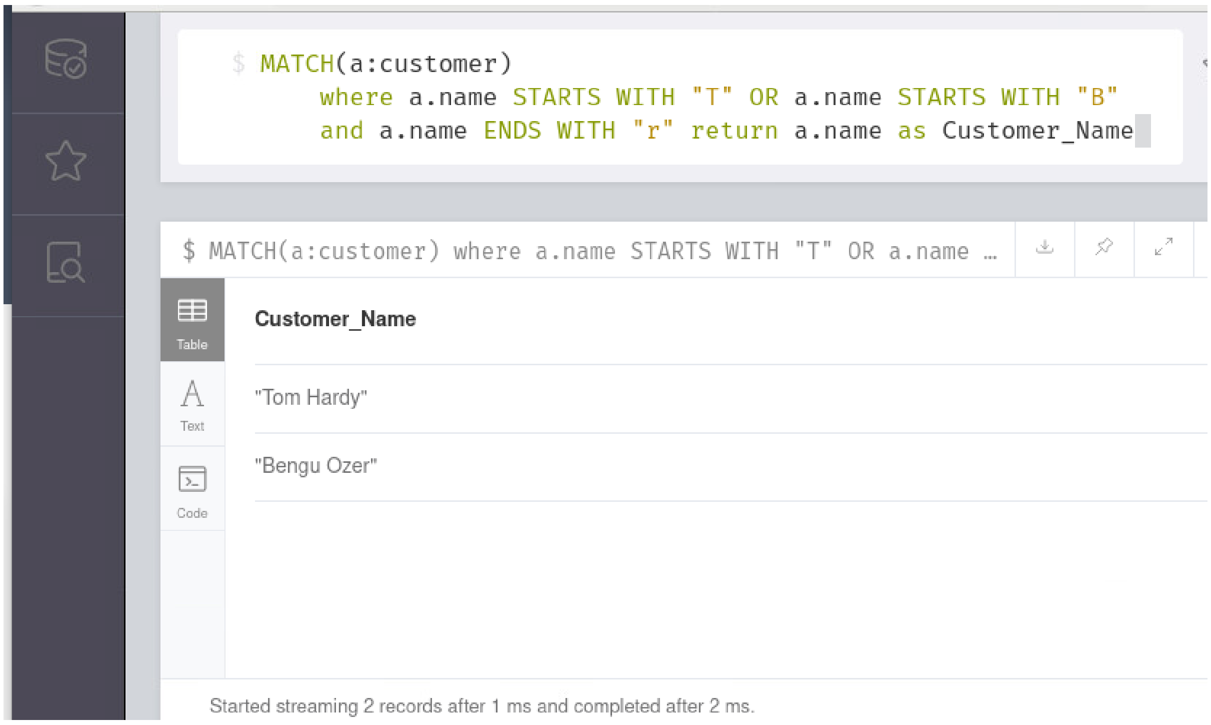
```
match(a:Book)-[:WRITTEN_BY]→(b:Writer) where a.sold_mCopi…
```

Book_Name	Writer_name	a.sold_mCopies
"Harry Potter and the Goblet of Fire"	"Joanne Rowling"	65.0
"Harry Potter and the Chamber of Secrets"	"Joanne Rowling"	77.0
"Harry Potter and the Prisoner of Azkaban"	"Joanne Rowling"	68.0

Started streaming 3 records after 1 ms and completed after 2 ms.

FIGURE 56:QUERY6 NEO4J

Case 7: Find the customers that their name starts with 'T' OR ends with 'r'.



```
$ MATCH(a:customer)
      where a.name STARTS WITH "T" OR a.name STARTS WITH "B"
      and a.name ENDS WITH "r" return a.name as Customer_Name
```

```
$ MATCH(a:customer) where a.name STARTS WITH "T" OR a.name …
```

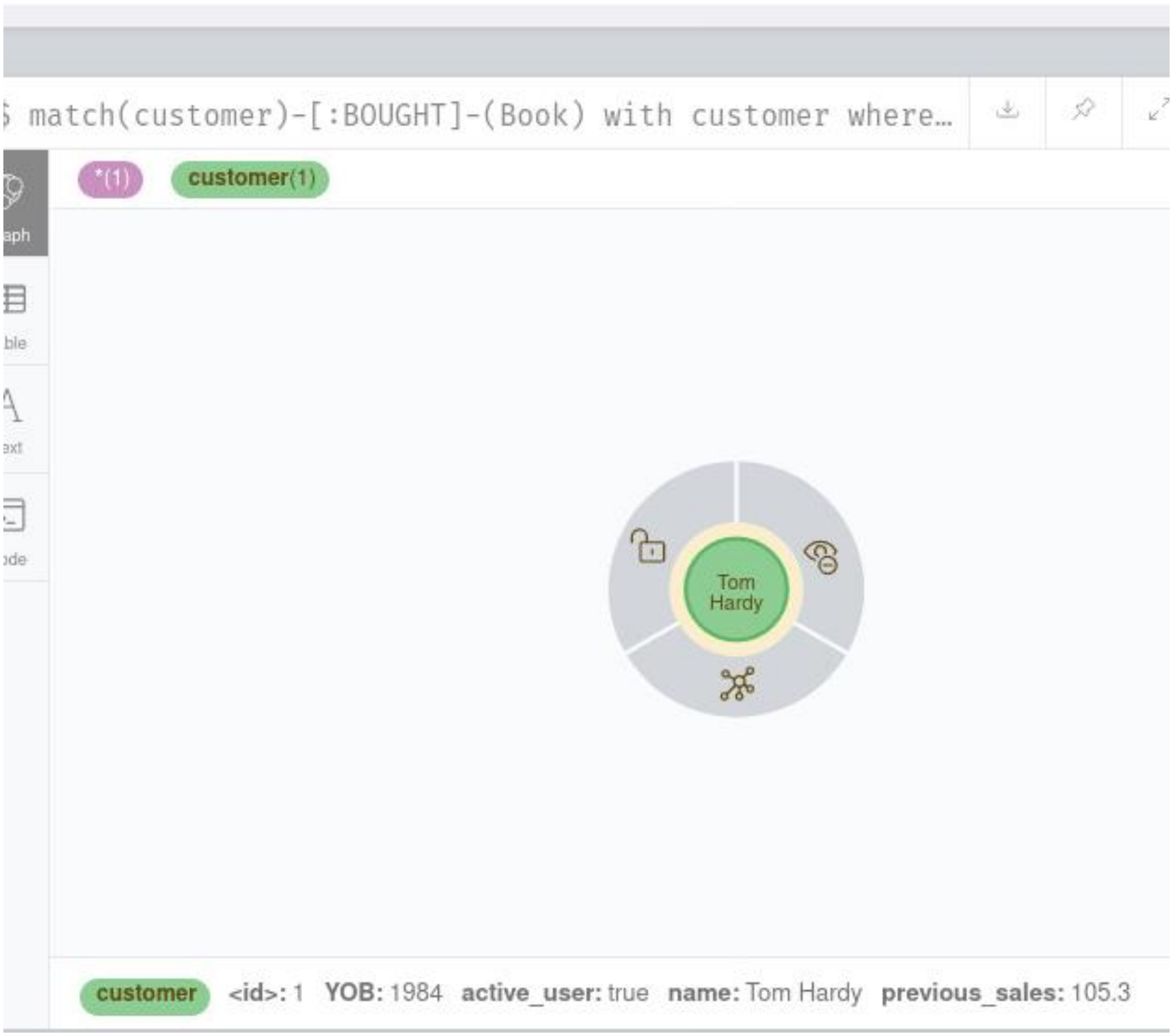
Customer_Name
"Tom Hardy"
"Bengu Ozer"

```
Started streaming 2 records after 1 ms and completed after 2 ms.
```

FIGURE 57:QUERY7 NEO4J

Case 8: Add a field – year of birth – to customer whose name is "Tom Hardy" and nought a book.

```
$ match(customer)-[:BOUGHT]-(Book) with customer
  where customer.name="Tom Hardy"
          SET customer.YOB=1984
      RETURN customer
```

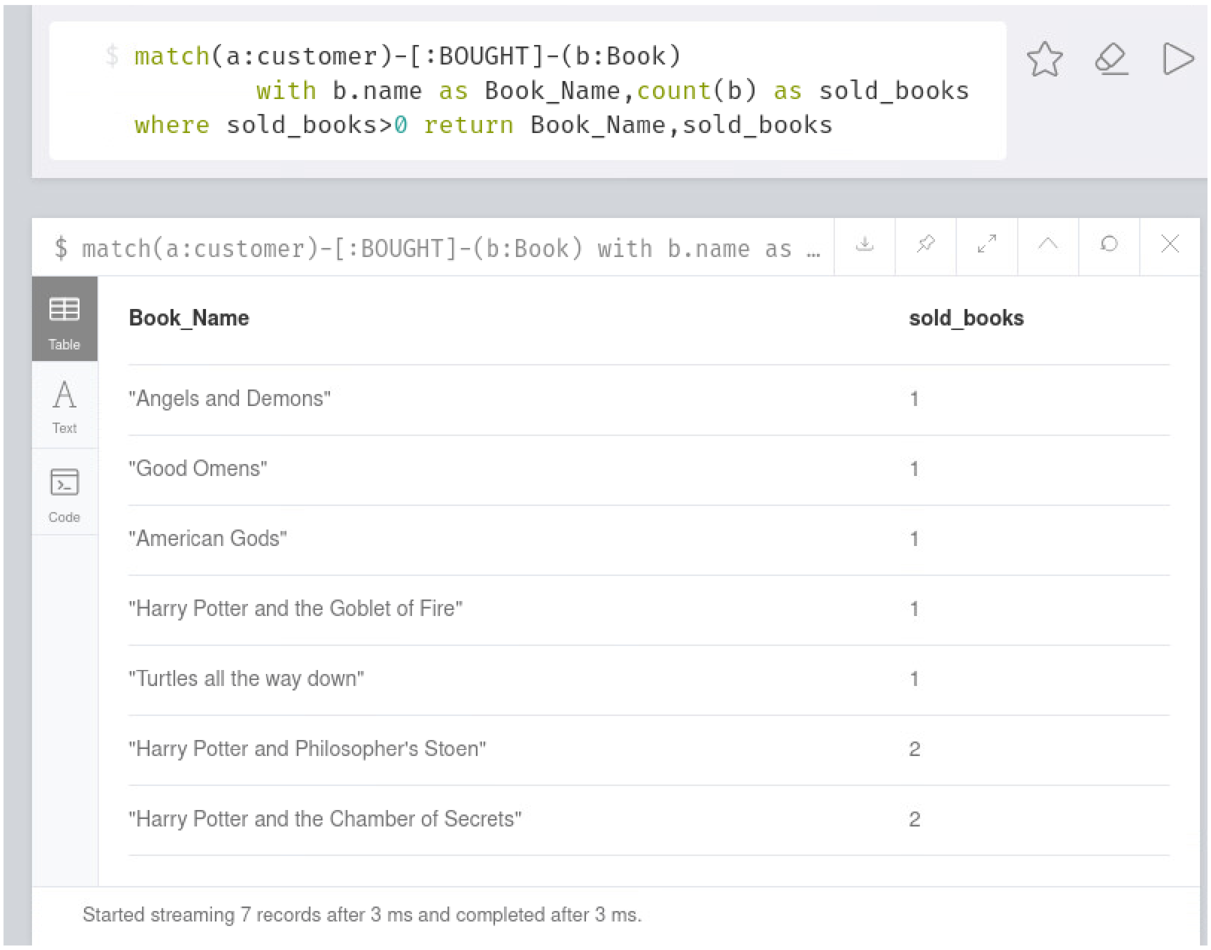


```
$ match(customer)-[:BOUGHT]-(Book) with customer where…
```

```
customer  <id>: 1  YOB: 1984  active_user: true  name: Tom Hardy  previous_sales: 105.3
```

FIGURE 58:QUERY8 NEO4J

Case 9: Total number of each sold books with amounts and their names.



```
$ match(a:customer)-[:BOUGHT]-(b:Book)
          with b.name as Book_Name,count(b) as sold_books
  where sold_books>0 return Book_Name,sold_books
```

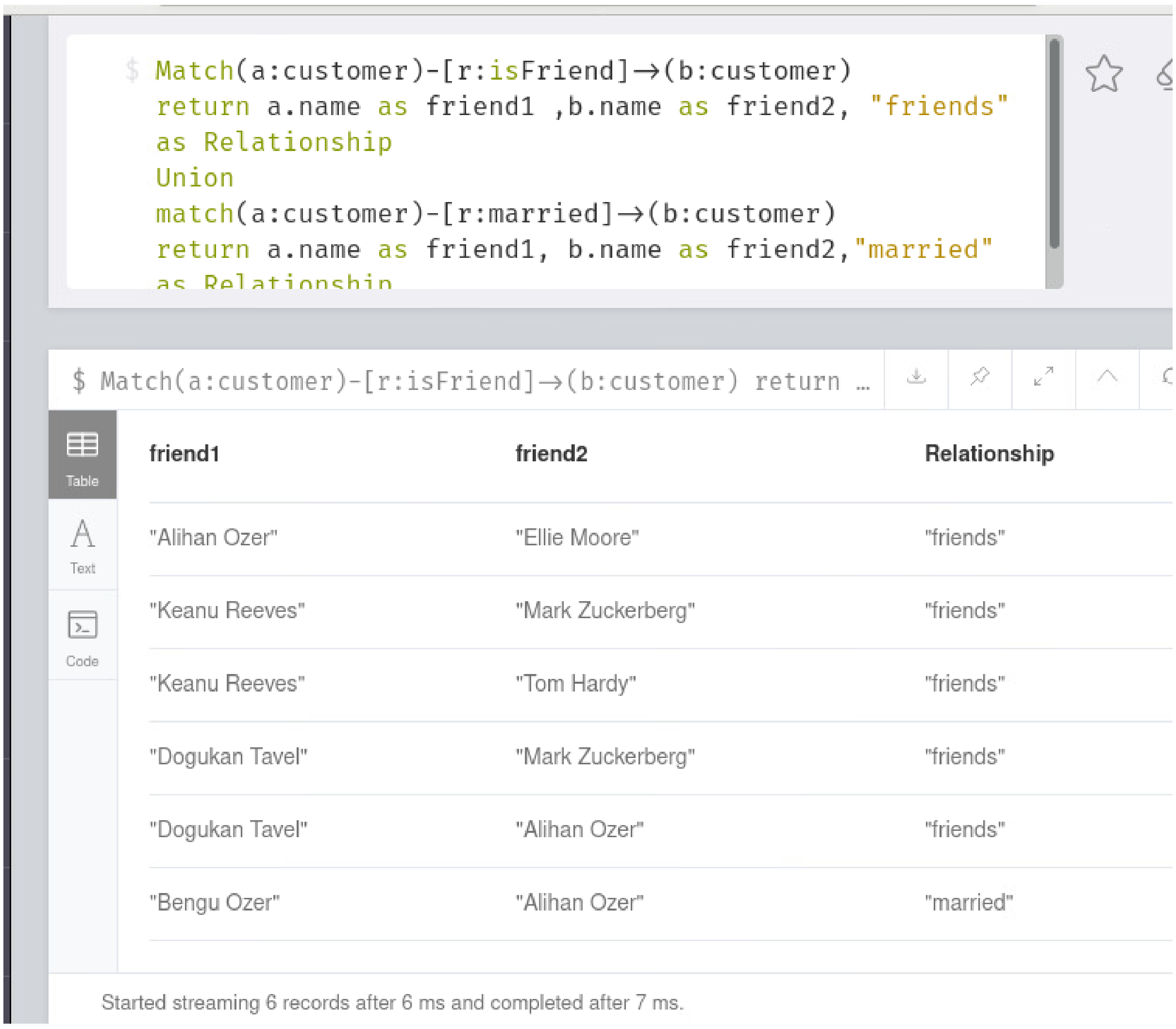
```
$ match(a:customer)-[:BOUGHT]-(b:Book) with b.name as …
```

Book_Name	sold_books
"Angels and Demons"	1
"Good Omens"	1
"American Gods"	1
"Harry Potter and the Goblet of Fire"	1
"Turtles all the way down"	1
"Harry Potter and Philosopher's Stoen"	2
"Harry Potter and the Chamber of Secrets"	2

```
Started streaming 7 records after 3 ms and completed after 3 ms.
```

FIGURE 59:QUERY9 NEO4J

Case 10: Find the Relationships between customers.



```
$ Match(a:customer)-[r:isFriend]→(b:customer)
  return a.name as friend1 ,b.name as friend2, "friends"
  as Relationship
  Union
  match(a:customer)-[r:married]→(b:customer)
  return a.name as friend1, b.name as friend2,"married"
  as Relationship
```

```
$ Match(a:customer)-[r:isFriend]→(b:customer) return …
```

friend1	friend2	Relationship
"Alihan Ozer"	"Ellie Moore"	"friends"
"Keanu Reeves"	"Mark Zuckerberg"	"friends"
"Keanu Reeves"	"Tom Hardy"	"friends"
"Dogukan Tavel"	"Mark Zuckerberg"	"friends"
"Dogukan Tavel"	"Alihan Ozer"	"friends"
"Bengu Ozer"	"Alihan Ozer"	"married"

```
Started streaming 6 records after 6 ms and completed after 7 ms.
```

FIGURE 60:QUERY10 NEO4J

Appendices

Appendix A – JSON Schema

```
{
 "$schema": "https://json-schema.org/draft/2019-09/schema",
 "$id": "http://example.com/product.schema.json",
 "title": "Product",
 "description" : "A product in the catalog",
 "type": "array",
 "items":{
   "type":"object",
   "properties":{
     "Name":{
       "description":"The name of the player",
       "type":"string"
   },
     "Goals":{
     "description":"Amount of goals",
     "type":"integer"
     },
     "Asists":{
     "description":"Amount of Asists",
     "type":"integer"
     },
     "TransferValue":{
      "description":"The transfer value of the player",
      "type":"number",
      "exclusiveminimum":0
```

```
  },
   "Age":{
   "description":"Age of the player",
   "type":"integer",
   "minimum":14
  },
   "HasManager":{
   "description":"player has a manager company or not",
   "type":"boolean"
  },
   "Birthday":{
   "description":"Birthday of the player",
   "type":"string",
   "format": "date"
  },
     "Country":{
    "description":"Country of the player",
    "type":"string"
 },
   "Foot":{
    "description":"Most efficient foot of the player",
    "type":"string"
 },
   "Positions":{
    "description":"Possible positions of player",
    "type":"array",
    "items":{
      "type":"string"
      },
      "minItems":1,
      "maxItems":4,
        "uniqueItems":true
   },
 },
     "required":["Name","Goals","Age","TransferValue"]
   }
 }
```

Appendix B – JSON Data

```
[
 {
 "Name":"Mohammed Salah",
 "Goals":20,
 "Asists":10,
 "TransferValue":90.00,
 "Age":29,
 "HasManager":true,
 "Birthday":"1992-07-15",
 "Country":"Egypt",
 "Foot":"left",
 "Positions":["left-winger","right-winger"],
},

 {
 "Name":"Heung-min Son",
 "Goals":13,
 "Asists":6,
 "TransferValue":72.00,
 "Age":29,
 "HasManager":true,
 "Birthday":"1992-07-08",
 "Country":"South Korea",
 "Foot":"both",
 "Positions":["left-winger","right-winger"],
},
  {
 "Name":"Diogo Jota",
 "Goals":13,
 "Asists":26,
 "TransferValue":54.00,
 "Age":25,
 "HasManager":true,
 "Birthday":"1996-12-04",
 "Country":"Portugal",
 "Foot":"right",
 "Positions":["attack","centre","forward"],
},
   {
 "Name":"Harry Kane",
 "Goals":12,
 "Asists":28,
 "TransferValue":90.00,
 "Age":29,
 "HasManager":true,
 "Birthday":"1993-07-28",
 "Country":"England",
 "Foot":"both",
 "Positions":["attack","centre","forward"],
},
   {
 "Name":"Cristiano Ronaldo",
 "Goals":12,
 "Asists":24,
 "TransferValue":31.50,
 "Age":37,
 "HasManager":true,
 "Birthday":"1985-02-05",
 "Country":"Portugal",
 "Foot":"both",
 "Positions":["attack","centre","forward"],
},
   {
 "Name":"Sadio Mané",
 "Goals":12,
 "Asists":26,
 "TransferValue":72.00,
 "Age":29,
 "HasManager":true,
 "Birthday":"1992-04-10",
 "Country":"Senegal",
 "Foot":"both",
 "Positions":["attack","left-winger"],
},
 {
 "Name":"Ivan Toney",
 "Goals":11,
 "Asists":26,
 "TransferValue":28.80,
 "Age":29,
 "HasManager":true,
 "Birthday":"1996-03-16",
 "Country":"England",
 "Foot":"right",
 "Positions":["attack","centre","forward"],
},
 {
 "Name":"Riyad Mahrez",
 "Goals":10,
 "Asists":22,
 "TransferValue":36.00,
 "Age":31,
 "HasManager":false,
 "Birthday":"1991-02-21",
 "Country":"France",
 "Foot":"left",
 "Positions":["attack","right-winger"],
},
 {
 "Name":"Raheem Sterling",
 "Goals":10,
 "Asists":23,
 "TransferValue":76.50,
 "Age":27,
 "HasManager":false,
 "Birthday":"1994-12-08",
 "Country":"England",
 "Foot":"right",
 "Positions":["attack","left-winger"],
},
  {
 "Name":"Jamie Vardy",
 "Goals":10,
 "Asists":18,
 "TransferValue":4.50,
 "Age":35,
 "HasManager":false,
 "Birthday":"1987-01-11",
 "Country":"England",
 "Foot":"right",
 "Positions":["attack","centre","forward"],
},

 ]
```

Bibliography

Sahatqija, K., Ajdari, J., Zenuni, X., Raufi, B. and Ismaili, F., 2018, May. Comparison between relational and NOSQL databases. In 2018 41st international convention on information and communication technology, electronics and microelectronics (MIPRO) (pp. 0216-0221). IEEE.

Sareen, P. and Kumar, P., 2015. NoSQL Database and its comparison with SQL Database. International Journal of Computer Science & Communication Networks, 5(5), pp.293-298.

Kumar, K.S. and Mohanavalli, S., 2017, January. A performance comparison of document oriented NoSQL databases. In 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP) (pp. 1-6). IEEE.

Moniruzzaman, A.B.M. and Hossain, S.A., 2013. Nosql database: New era of databases for big data analytics-classification, characteristics and comparison. arXiv preprint arXiv:1307.0191.

Venkatraman, S., Fahd, K., Kaspi, S. and Venkatraman, R., 2016. SQL versus NoSQL movement with big data analytics. International Journal of Information Technology and Computer Science, 8(12), pp.59-66.