Assignment Title:

Coursework 2

Coursework Type:

Individual

Module Name:

STW220CT: Data and Information Retrieval

Intake:

MARCH 2020

Submitted By

CU ID: 10272453

College ID: 180072

Name: Sayyed Abrar Akhtar

Table of Contents

Task 1: MongoDB

Games Data

Name	Publisher	Released	Rating	Country	Address	Player name	Goal score
Hays Wise	KOEI Co., Ltd.	April 5, 1990	99	USA	694 Hewes Street	Derrick	705
						Tim	379
						Bryan	810
Ape Escape	EIOI Co., Ltd.	August 5, 1990	44	France	795 Borinquen Pl	Alpha	200
						Alan	500
						Jordan	290
Digi Laus	DIGITALUS Co., Ltd	September 5, 1990	50	Italy	154 Arlington Avenu	Alpha	300
						Aubrey	200
Danone	Danone Co. , Ltd	August 11, 1990	66	France	897 Borinquen Pl	Gabriel	400
						Paul	100
						Arthur	200
						Victor	120

1. Create a database named <<YourName_Games>> and a collection named <<YourName_CoventryID_Games>> and insert the above data.

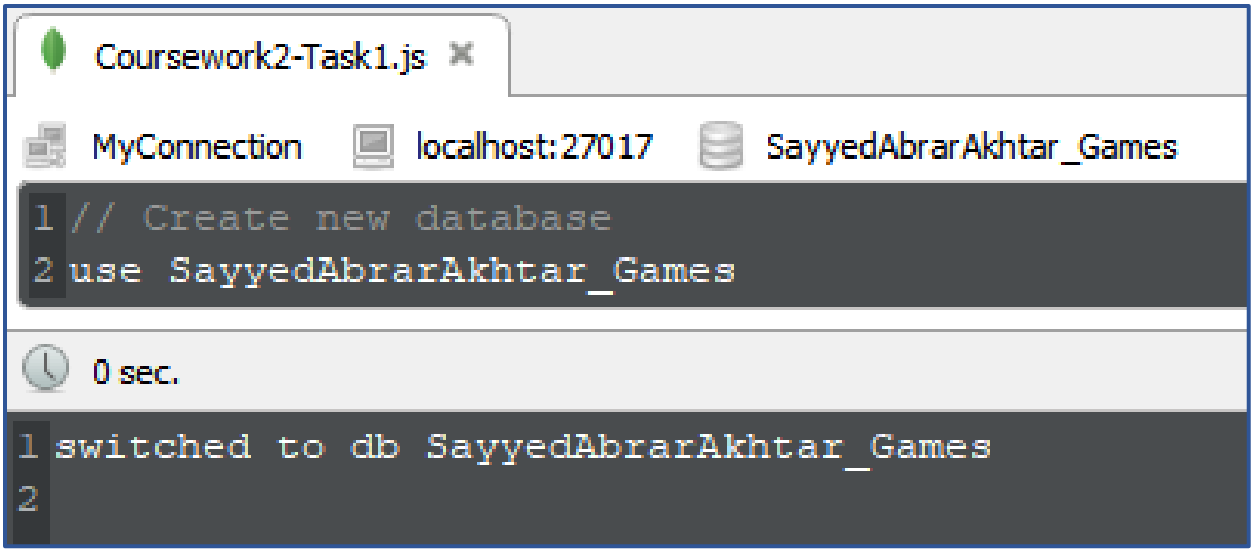


Figure 1: Query to create new database

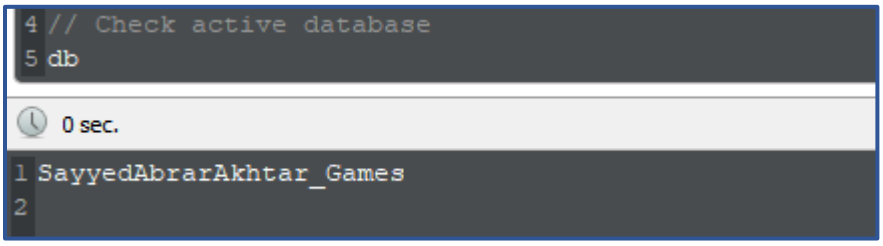


Figure 2: Database created

```
7 // Create collection
8 db.createCollection("SayyedAbrarAkhtar_10272453_Games")
```

0.249 sec.

Key	Value	Type
(1)	{ 1 field }	Object
ok	1.0	Double

Figure 3: Query to create collection

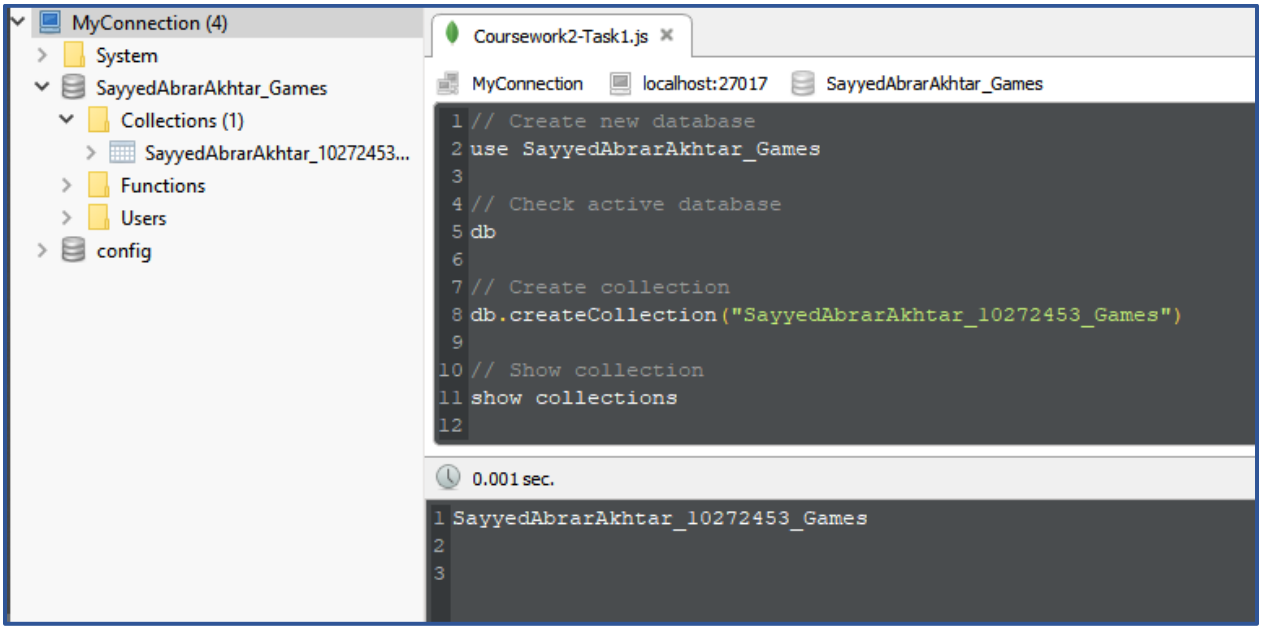


Figure 4: Collection created

Coursework2-Task1.js

MyConnection localhost:27017 SayyedAbrarAkhtar_Games

```
// Insert data
db.getCollection("SayyedAbrarAkhtar_10272453_Games").insert({
    Name: "Hays Wise",
    Publisher: "KOEI Co., Ltd.",
    Released: "April 5, 1990",
    Rating: NumberInt(99),
    Country: "USA",
    Address: "694 Hewes Street",
    PlayerDetails: [
        {Player_name: "Derrick", Goal_score: NumberInt(705)},
        {Player_name: "Tim", Goal_score: NumberInt(379)},
        {Player_name: "Bryan", Goal_score: NumberInt(810)}
    ]
});
```

0.001 sec.

```
Inserted 1 record(s) in 1ms
```

Figure 5: Query to insert first record

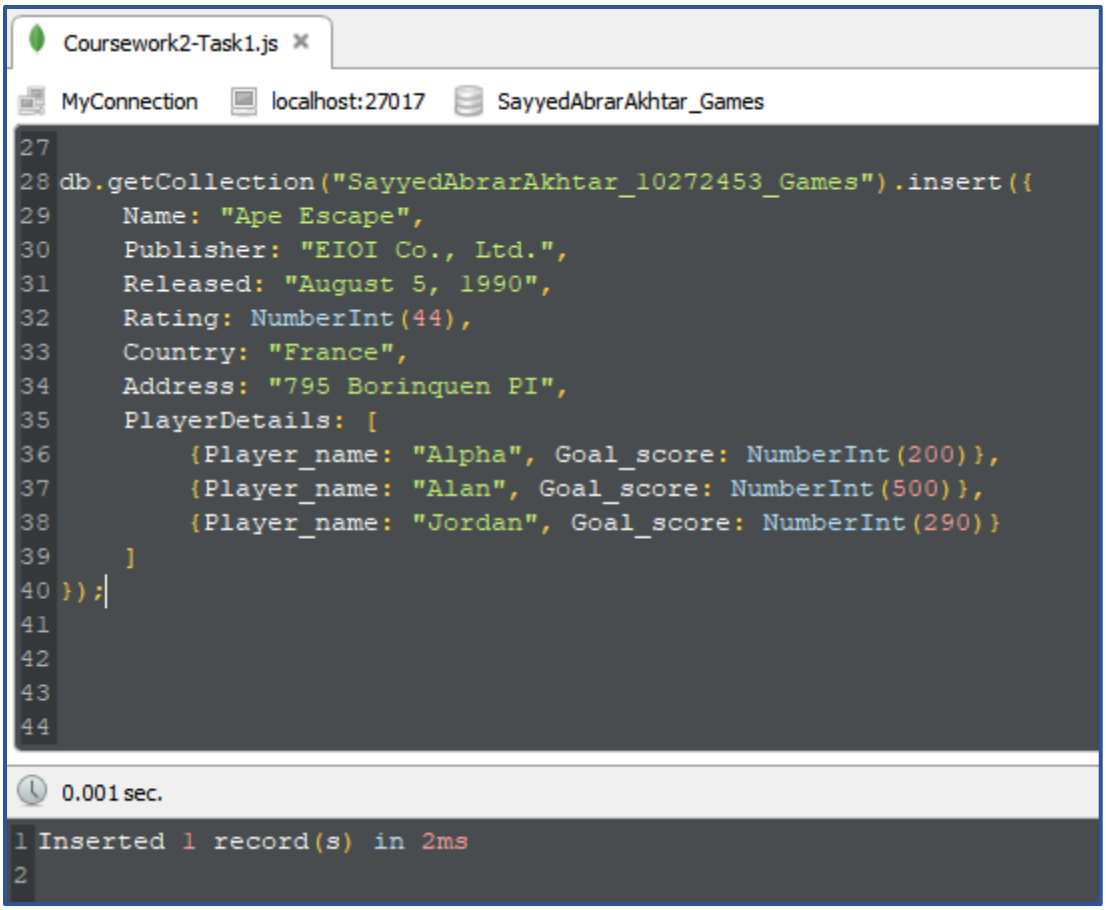


Figure 6: Query to insert second record

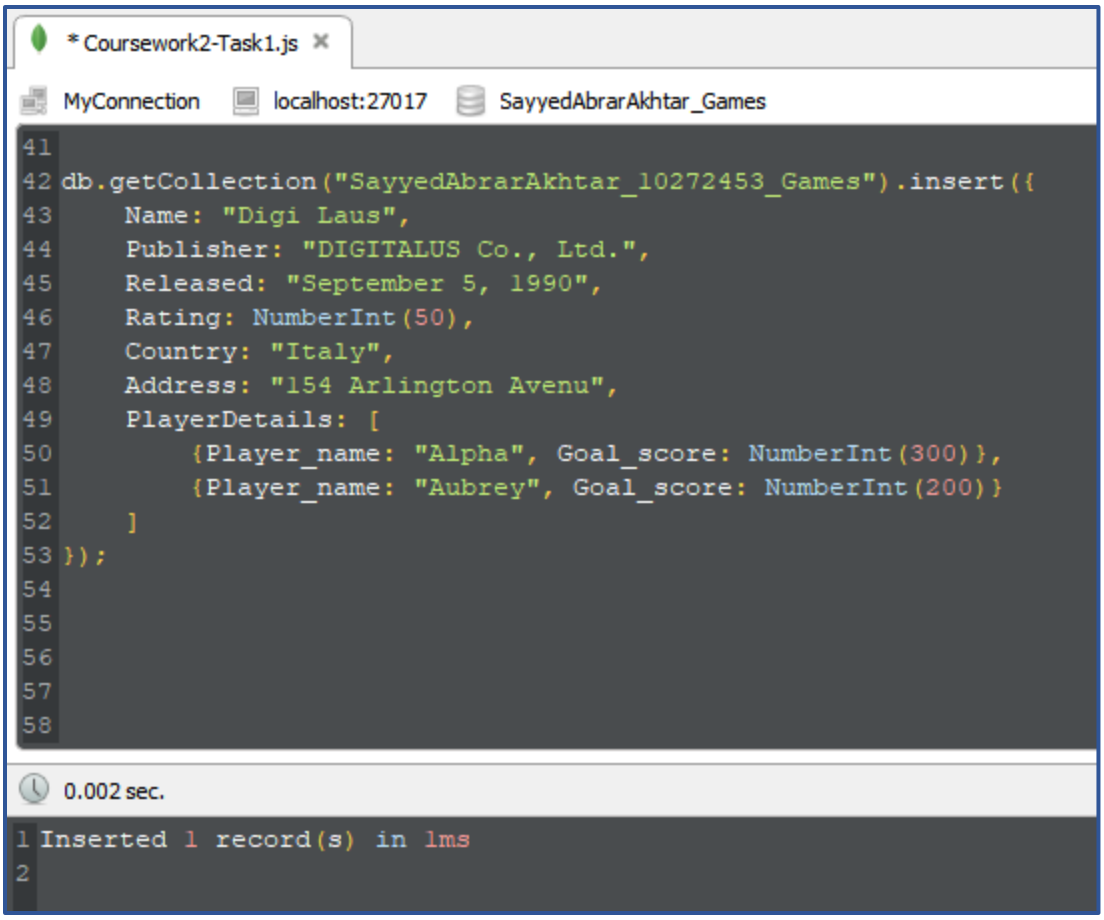


Figure 7: Query to insert Third record

Coursework2-Task1.js

MyConnection | localhost:27017 | SayyedAbrarAkhtar_Games

```
54
55 db.getCollection("SayyedAbrarAkhtar_10272453_Games").insert({
56     Name: "Danone",
57     Publisher: "Danone Co., Ltd.",
58     Released: "August 11, 1990",
59     Rating: NumberInt(66),
60     Country: "France",
61     Address: "897 Borinquen PI",
62     PlayerDetails: [
63         {Player_name: "Gabriel", Goal_score: NumberInt(400)},
64         {Player_name: "Paul", Goal_score: NumberInt(100)},
65         {Player_name: "Arthur", Goal_score: NumberInt(200)},
66         {Player_name: "Victor", Goal_score: NumberInt(120)}
67     ]
68 });
69
70
71
```

0.001 sec.

```
1 Inserted 1 record(s) in 2ms
2
```

Figure 8: Query to insert fourth record

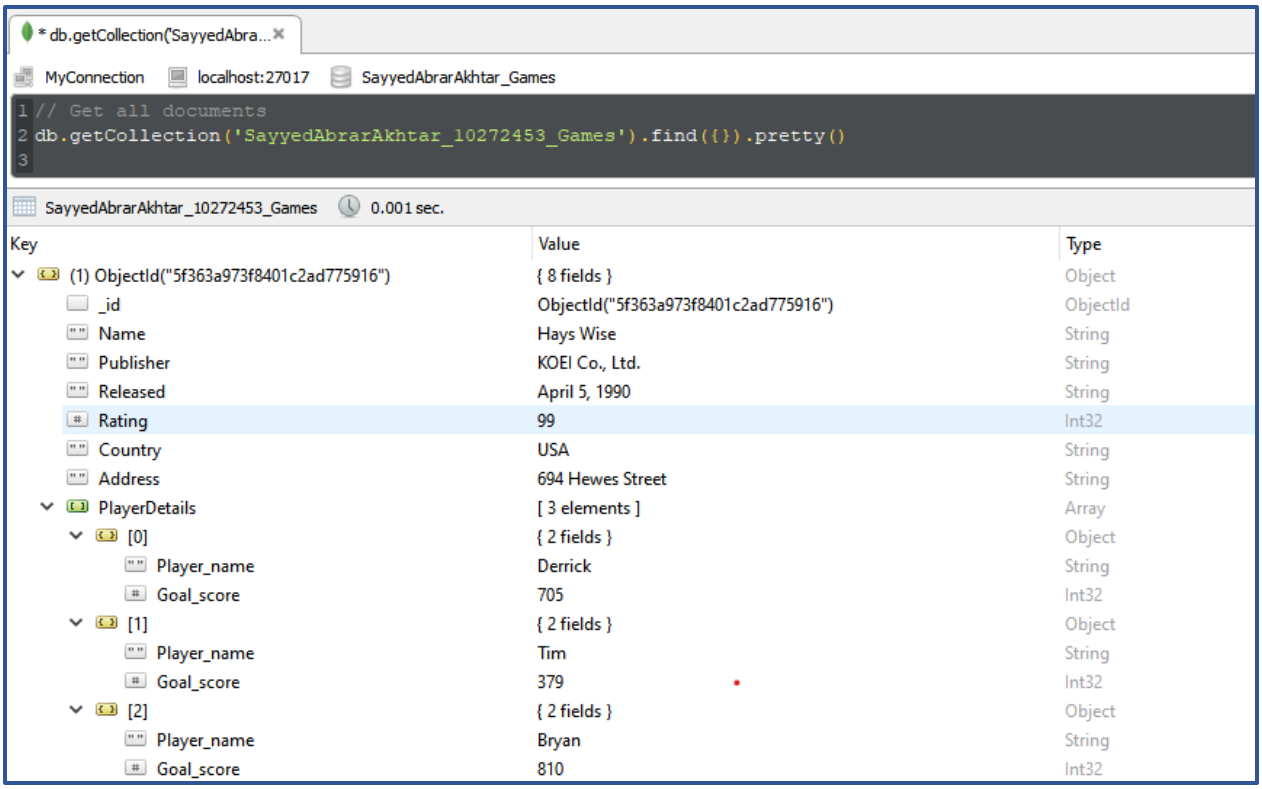


Figure 9: First record inserted

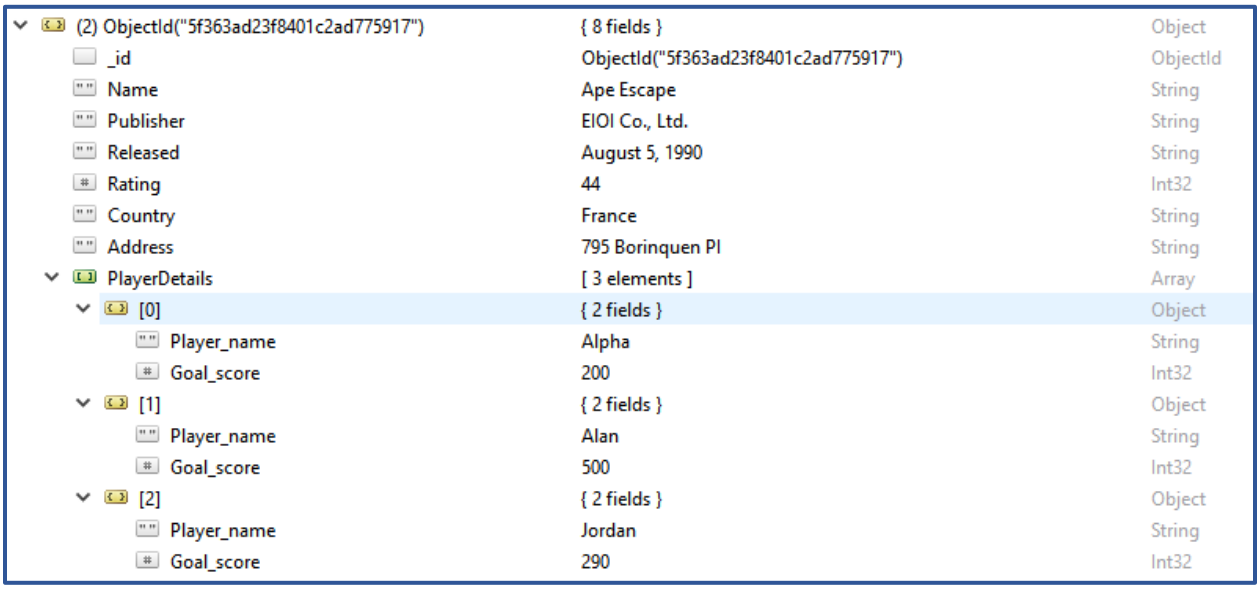
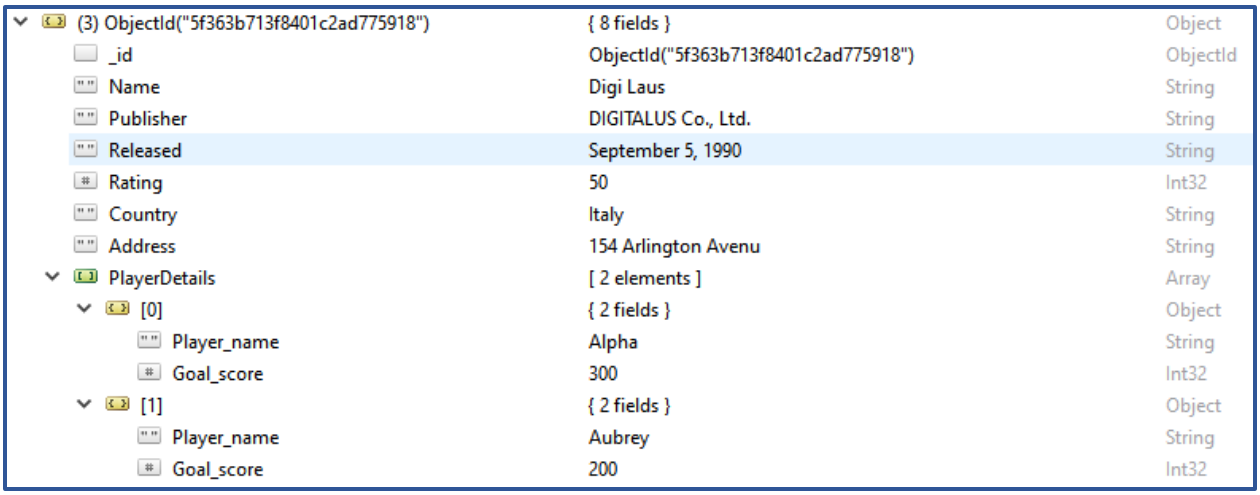
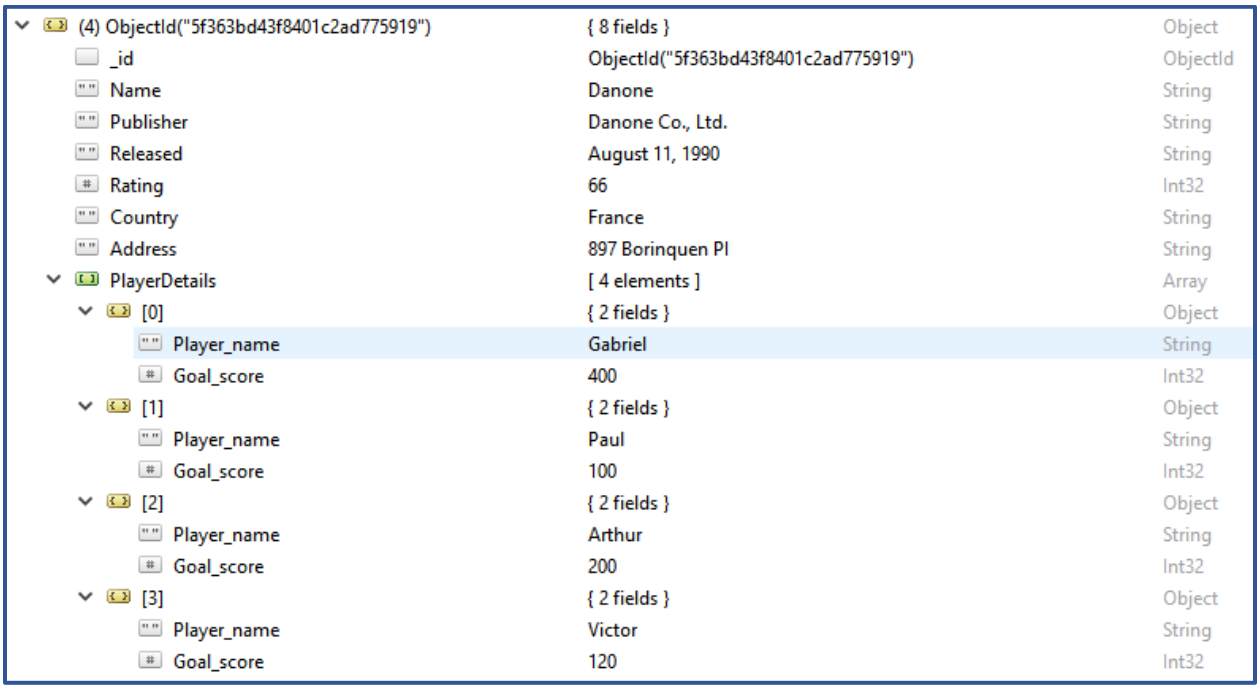


Figure 10: Second record inserted



(3) ObjectId("5f363b713f8401c2ad775918")	{ 8 fields }	Object
_id	ObjectId("5f363b713f8401c2ad775918")	ObjectId
Name	Digi Laus	String
Publisher	DIGITALUS Co., Ltd.	String
Released	September 5, 1990	String
Rating	50	Int32
Country	Italy	String
Address	154 Arlington Avenu	String
PlayerDetails	[2 elements]	Array
[0]	{ 2 fields }	Object
Player_name	Alpha	String
Goal_score	300	Int32
[1]	{ 2 fields }	Object
Player_name	Aubrey	String
Goal_score	200	Int32

Figure 11: Third record inserted



(4) ObjectId("5f363bd43f8401c2ad775919")	{ 8 fields }	Object
_id	ObjectId("5f363bd43f8401c2ad775919")	ObjectId
Name	Danone	String
Publisher	Danone Co., Ltd.	String
Released	August 11, 1990	String
Rating	66	Int32
Country	France	String
Address	897 Borinquen Pl	String
PlayerDetails	[4 elements]	Array
[0]	{ 2 fields }	Object
Player_name	Gabriel	String
Goal_score	400	Int32
[1]	{ 2 fields }	Object
Player_name	Paul	String
Goal_score	100	Int32
[2]	{ 2 fields }	Object
Player_name	Arthur	String
Goal_score	200	Int32
[3]	{ 2 fields }	Object
Player_name	Victor	String
Goal_score	120	Int32

Figure 12: Fourth record inserted

2. What is Map-Reduce? Explain the working of map-reduce with an example.

Before the introduction of Map-Reduce, traditional approach was used for parallel and distributed processing. But traditional approach has some challenges associated with it. Some of them are:

- While processing if any function is interrupted whole process gets delayed.
- There is no management of any part of data failure.
- There was no way of dividing work into equal part to ensure machine is not overloaded.
- If working on any part of data fails whole calculation will be failed.

To overcome these challenges Map-Reduce framework is developed. It allows parallel processing solving the challenges like fault tolerance, reliability, and so on. Map-Reduce is a data processing paradigm that allows distributed and parallel processing of large datasets into useful aggregated results.

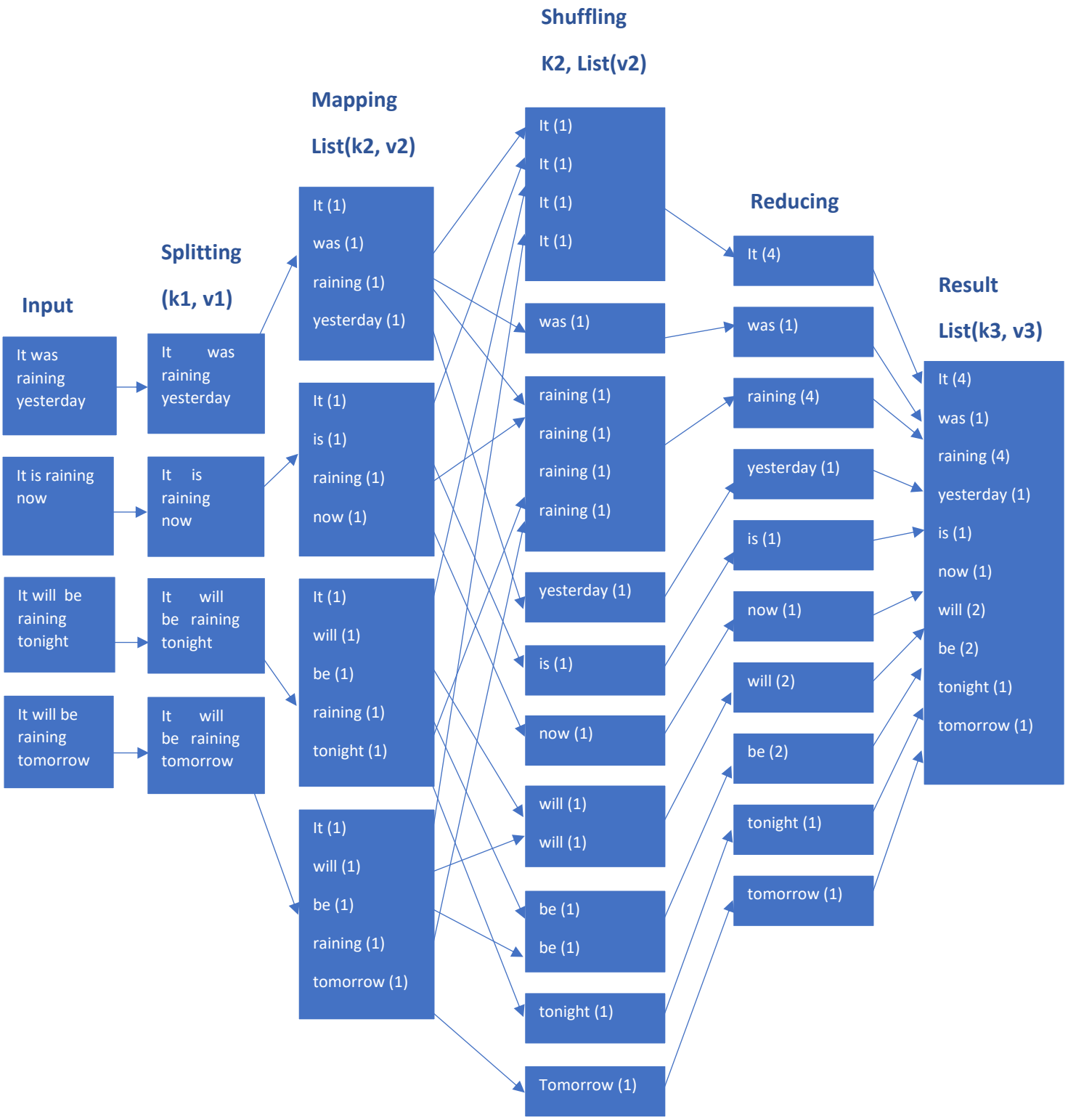
In Map-Reduce, Map is a task and Reduce is another task. Once the mapping is completed, reducing begins. In the mapping phase, block of data is read and processed to output key value pairs. The output key value pair data is served as input in the reducing phase. The reducer outputs the result aggregating those key-value pairs into key value pairs.

Example:

Map-reduce framework can be used for counting words in a paragraph or sentence. The sentences are:

- It was raining yesterday
- It is raining now
- It will be raining tonight
- It will be raining tomorrow

In order to count the words map- reduce framework input the above sentences, passes through splitting, mapping, shuffling, reducing and finally output.



The above diagram shows the map-reduce process for counting words in sentences.

Input: Sentences are provided as input.

Splitting: The input is divided into chunks as shown in the diagram above. It distributes the work among all the nodes.

Mapping: Each word is tokenized and because they occur once in itself each token is hardcoded by value one. Thus, this makes a key value pair where key is the word itself and value (1).

Shuffling: In this stage partition process begins as well as sorting and shuffling starts to send all the list with same key to the matching reducer. It has a key and list of value matching to the key.

Reducing: In this stage reducer count the number of ones in the list as value and provides the final output where key remains the same.

Final result: Thus, the output key value pair is returned by the reducer.

3. Write a reduce function that calculates the total score for each team with the publisher name and count the number of players in each team.

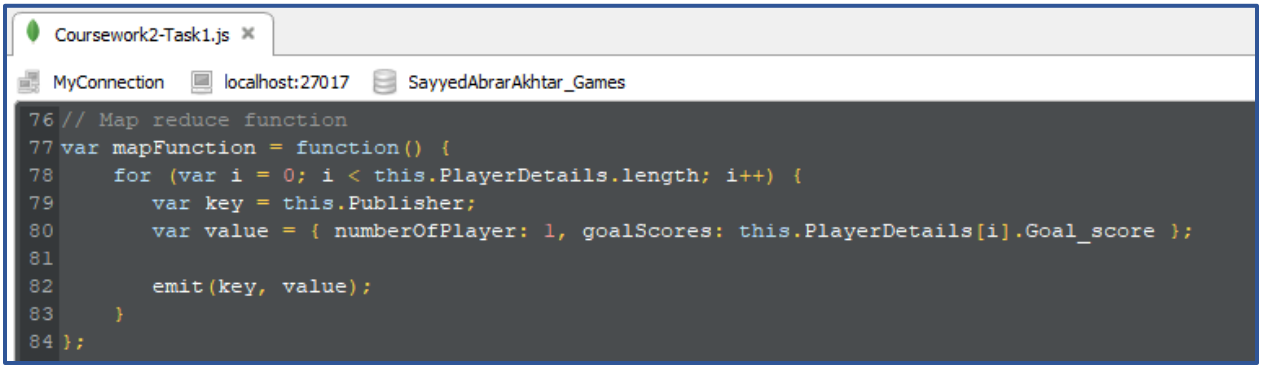


Figure 13:Query to create Map function

```
var reduceFunction = function(keyPublisher, countObjectValues) {
    reducedValues = { numberOfPlayer: 0, goalScores: 0 };

    for (var i = 0; i < countObjectValues.length; i++) {
        reducedValues.numberOfPlayer += countObjectValues[i].numberOfPlayer;
        reducedValues.goalScores += countObjectValues[i].goalScores;
    }

    return reducedValues;
};
```

Figure 14: Query to create Reduce Function

```
98
99 db.SayyedAbrarAkhtar_10272453_Games.mapReduce(
100    mapFunction,
101    reduceFunction,
102    {
103      out: "map_reduce_result"
104      |
105    }
106 ).find().sort( { _id: -1 } );
107
```

Figure 15: Applied Map and Reduce function to calculate total score for each team, number of players and publisher name

robo3t

map_reduce_result 1.32 sec.

Key	Value	Type
(1) KOEI Co., Ltd.	{ 2 fields }	Object
_id	KOEI Co., Ltd.	String
value	{ 2 fields }	Object
numberOfPlayer	3.0	Double
goalScores	1894.0	Double
(2) EIOI Co., Ltd.	{ 2 fields }	Object
_id	EIOI Co., Ltd.	String
value	{ 2 fields }	Object
numberOfPlayer	3.0	Double
goalScores	990.0	Double
(3) Danone Co., Ltd.	{ 2 fields }	Object
_id	Danone Co., Ltd.	String
value	{ 2 fields }	Object
numberOfPlayer	4.0	Double
goalScores	820.0	Double
(4) DIGITALUS Co., Ltd.	{ 2 fields }	Object
_id	DIGITALUS Co., Ltd.	String
value	{ 2 fields }	Object
numberOfPlayer	2.0	Double
goalScores	500.0	Double

Figure 16: Map-reduce output total score, number of players and publisher name

4. Count the number of players in Hays Wise.

Welcome | Coursework2-Task1.js

MyConnection | localhost:27017 | SayyedAbrarAkhtar_Games

```
104
105 // Count number of players in Hays Wise
106 db.SayyedAbrarAkhtar_10272453_Games.aggregate([
107     {
108         $match: {
109             "Name": "Hays Wise"
110         }
111     },
112     {
113         $project:{
114             "Number of Players": {$size: "$PlayerDetails"}
115         }
116
117     }
118 ]);
119
```

Figure 17: Query to count number of players in Hays Wise

SayyedAbrarAkhtar_10272453_Games | 0.002 sec.

Key	Value	Type
(1) ObjectId("5f3ccf07aeaa4d05e62d4ddc")	{ 2 fields }	Object
_id	ObjectId("5f3ccf07aeaa4d05e62d4ddc")	ObjectId
Number of Players	3	Int32

Figure 18: Displayed total number of players in Hays Wise

5. Remove the player “Alpha" from Ape Escape.

Coursework2-Task1.js

MyConnection | localhost:27017 | SayyedAbrarAkhtar_Games

```
126
127 db.SayyedAbrarAkhtar_10272453_Games.update(
128     {
129         "Name":"Ape Escape"
130     },
131     {
132         $pull: {
133             "PlayerDetails":{"Player_name": "Alpha"}
134         }
135     }
136 );
137
138
139
140
141
142
143
```

0.019 sec.

```
1 Updated 1 existing record(s) in 3ms
2
```

Figure 19: Query to remove Alpha from Ape Escape

SayyedAbrarAkhtar_10272453_Games | 0.002 sec.

Key	Value	Type
(1) ObjectId("5f363ad23f8401c2ad775917")	{ 8 fields }	Object
_id	ObjectId("5f363ad23f8401c2ad775917")	ObjectId
Name	Ape Escape	String
Publisher	EIOI Co., Ltd.	String
Released	August 5, 1990	String
Rating	44	Int32
Country	France	String
Address	795 Borinquen Pl	String
PlayerDetails	[2 elements]	Array
[0]	{ 2 fields }	Object
Player_name	Alan	String
Goal_score	500	Int32
[1]	{ 2 fields }	Object
Player_name	Jordan	String
Goal_score	290	Int32

Figure 20: Alpha removed from Ape Escape

6. Update the player name “Jordan” to “Michael” and score to 300.

Coursework2-Task1.js

MyConnection | localhost:27017 | SayyedAbrarAkhtar_Games

```
// Update player name Jordan to Michael and score to 300
db.SayyedAbrarAkhtar_10272453_Games.update(
    {
        "PlayerDetails.Player_name":"Jordan"
    },
    {
        $set:{
            "PlayerDetails.$.Player_name":"Michael",
            "PlayerDetails.$.Goal_score":NumberInt(300)
        }
    }
);

db.SayyedAbrarAkhtar_10272453_Games.find(
    { "PlayerDetails.Player_name":"Michael" }
);
```

Figure 21: Query to update player Jordan to Michael and score to 300

SayyedAbrarAkhtar_10272453_Games 0.001 sec.

Key	Value	Type
(1) ObjectId("5f363ad23f8401c2ad775917")	{ 8 fields }	Object
_id	ObjectId("5f363ad23f8401c2ad775917")	ObjectId
Name	Ape Escape	String
Publisher	ElOI Co., Ltd.	String
Released	August 5, 1990	String
Rating	44	Int32
Country	France	String
Address	795 Borinquen Pl	String
PlayerDetails	[2 elements]	Array
[0]	{ 2 fields }	Object
Player_name	Alan	String
Goal_score	500	Int32
[1]	{ 2 fields }	Object
Player_name	Michael	String
Goal_score	300	Int32

Figure 22: Jordan updated to Michael and Score to 300

7. Show all the number of players with their publisher name.

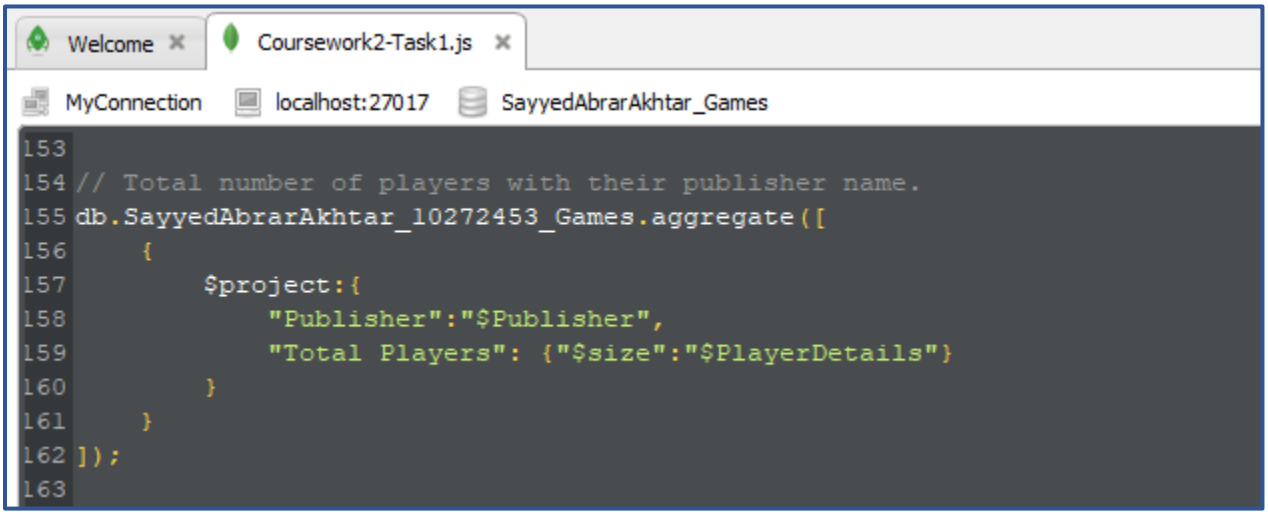


Figure 23: Query to display all players with publisher name

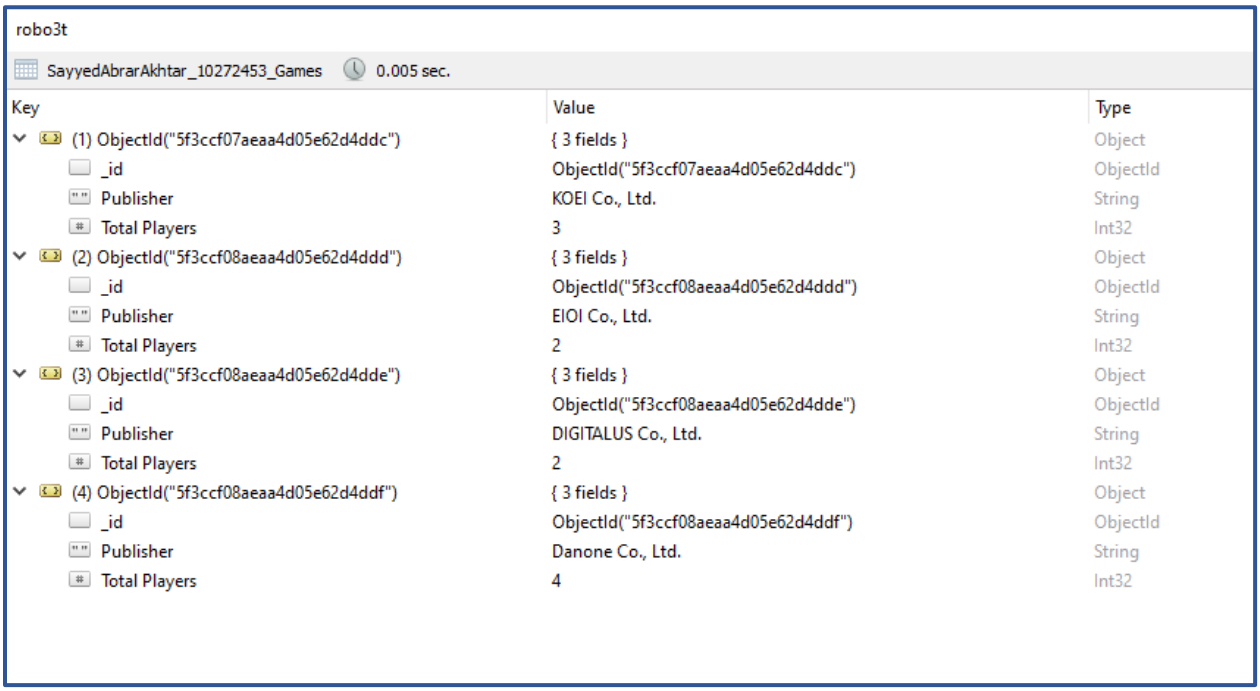


Figure 24: Shown all player with publisher name

8. Show total goals scored by each country name.

Welcome | Coursework2-Task1.js

MyConnection | localhost:27017 | SayyedAbrarAkhtar_Games

```
164 // Total goal scores by each country name.
165 db.SayyedAbrarAkhtar_10272453_Games.aggregate([
166     {
167         $unwind:"$PlayerDetails"
168     },
169     {
170         $group:{
171             _id:"$Country",
172             "Goal_score":{$sum : "$PlayerDetails.Goal_score"}
173         }
174     },
175     {
176         $project:{
177             "Total Goal Scored": "$Goal_score"
178         }
179     }
180 ]);
181
```

Figure 25: Query to show total goal scored by each country

robo3t

SayyedAbrarAkhtar_10272453_Games | 0.002 sec.

Key	Value	Type
(1) France	{ 2 fields }	Object
_id	France	String
Total Goal Scored	1620	Int32
(2) Italy	{ 2 fields }	Object
_id	Italy	String
Total Goal Scored	500	Int32
(3) USA	{ 2 fields }	Object
_id	USA	String
Total Goal Scored	1894	Int32

Figure 26: Displayed total goal scored by each country

Task 2: Development of a graph database for a given dataset.

Find "**Task_2_PremierLeauge_2019.csv**" file from the Moodle. The dataset contains information about the **English Premier League (EPL)** matches. You are expected to design and create a graph database to visualize the dataset and to answer the following queries:

1. Create a Data Model diagram for "Task_2_PremierLeague_2019.csv" dataset.

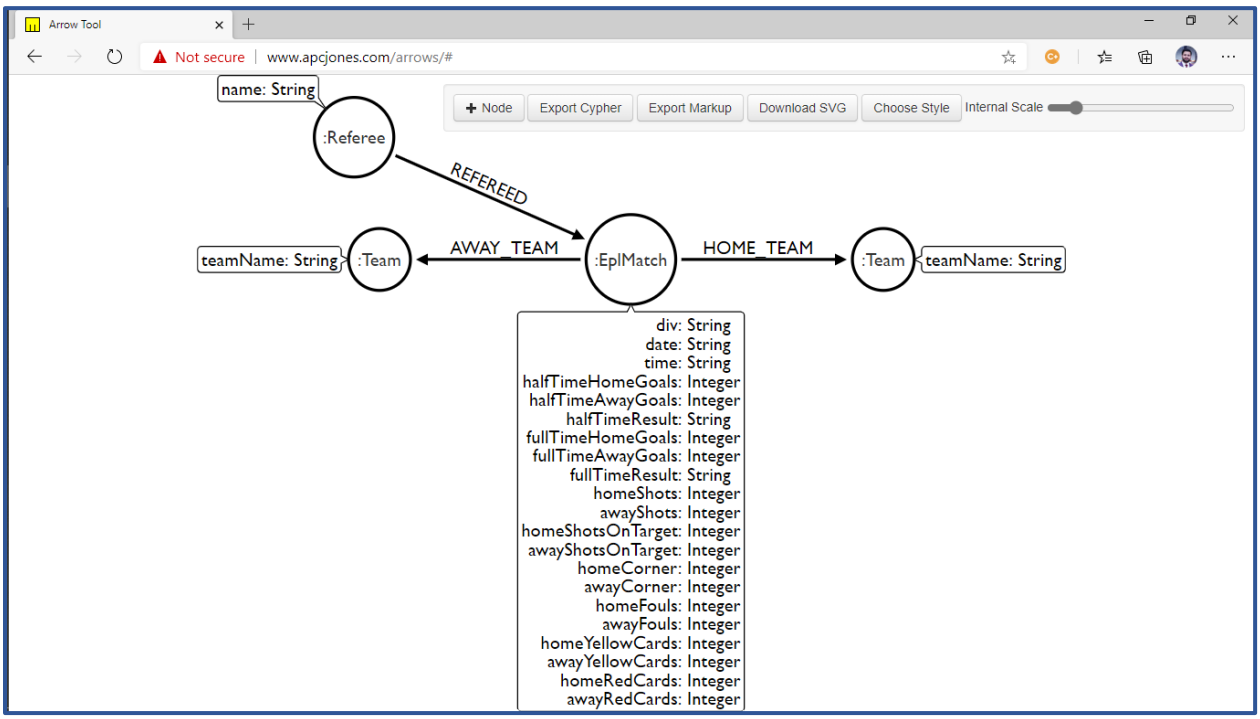


Figure 27: Data Model diagram of "Task_2_PremierLeague_2019.csv" dataset

2. Create nodes and relationship according to the Data Model which you have created in question no. 1.

```
// Query to create Nodes and Relationships
LOAD CSV WITH HEADERS FROM 'file:///Task_2_PremierLeauge_2019.csv' AS row
// Create Team nodes
MERGE(homeTeam:Team{teamName: row.`HomeTeam`})
MERGE(awayTeam:Team{teamName: row.`AwayTeam`})

// Create EplMatch nodes
MERGE(matches:EplMatch{div:row.`Div`, date:row.`Date`,
time: row.`Time`,
halfTimeHomeGoals: toInteger(row.`HTHG`),
halfTimeAwayGoals: toInteger(row.`HTAG`),
halfTimeResult: row.`HTR`,
fullTimeHomeGoals: toInteger(row.`FTHG`),
fullTimeAwayGoals: toInteger(row.`FTAG`),
fullTimeResult: row.`FTR`,
homeShots: toInteger(row.`HS`),
awayShots: toInteger(row.`AS`),
homeShotsOnTarget: toInteger(row.`HST`),
awayShotsOnTarget: toInteger(row.`AST`),
homeCorner: toInteger(row.`HC`),
awayCorner: toInteger(row.`AC`),
homeFouls: toInteger(row.`HF`),
awayFouls: toInteger(row.`AF`),
homeYellowCards: toInteger(row.`HY`),
awayYellowCards: toInteger(row.`AY`),
homeRedCards: toInteger(row.`HR`),
awayRedCards: toInteger(row.`AR`)
})

// Create relationship between EplMatch and Team nodes
MERGE (matches)-[:HOME_TEAM]->(homeTeam)
MERGE (matches)-[:AWAY_TEAM]->(awayTeam)

// Create Referee nodes
MERGE(referee:Referee{name: row.`Referee`})

// Create relationship between Referee and EplMatch
MERGE (referee)-[:REFEREED]->(matches)
```

Figure 28: Query to create nodes and relationship

Figure 29: Query executed and created 330 nodes and 864 relationships

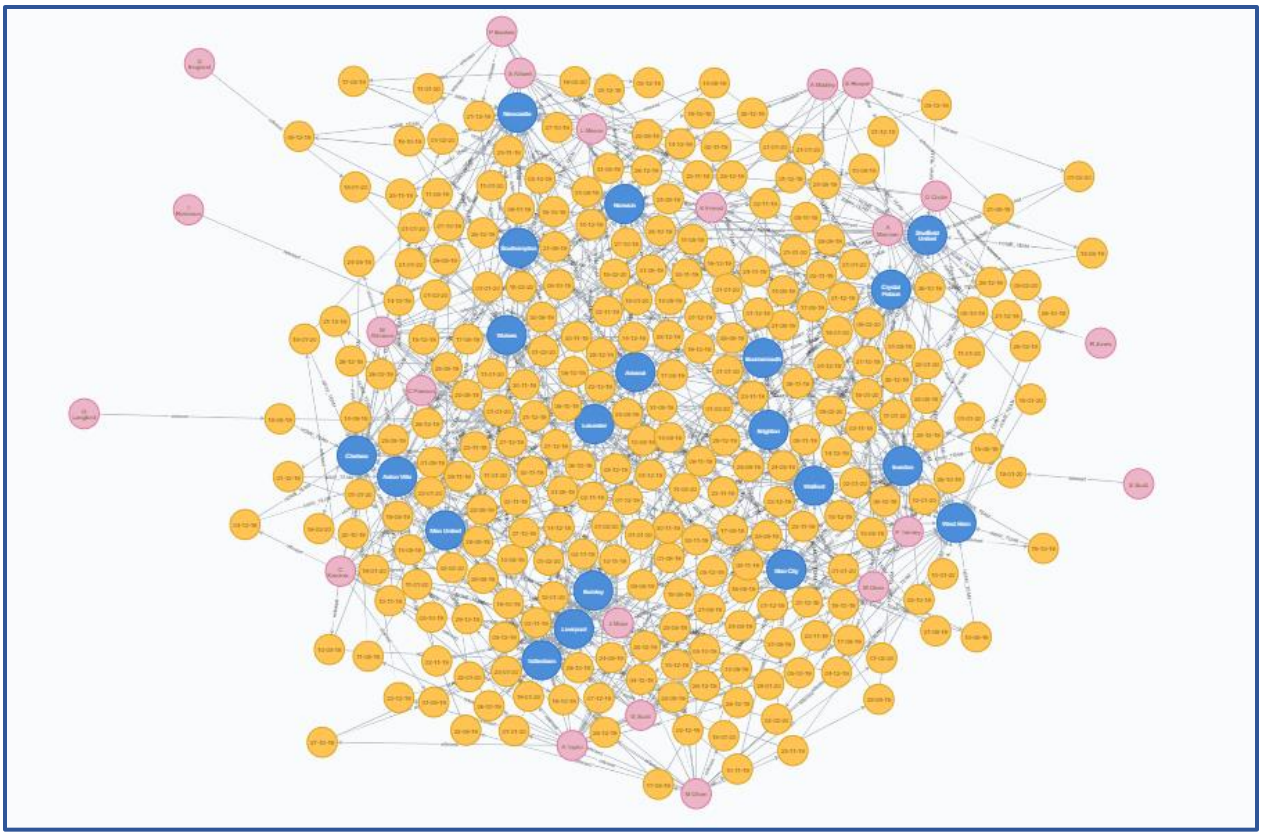


Figure 30: Graph of created nodes and relationships

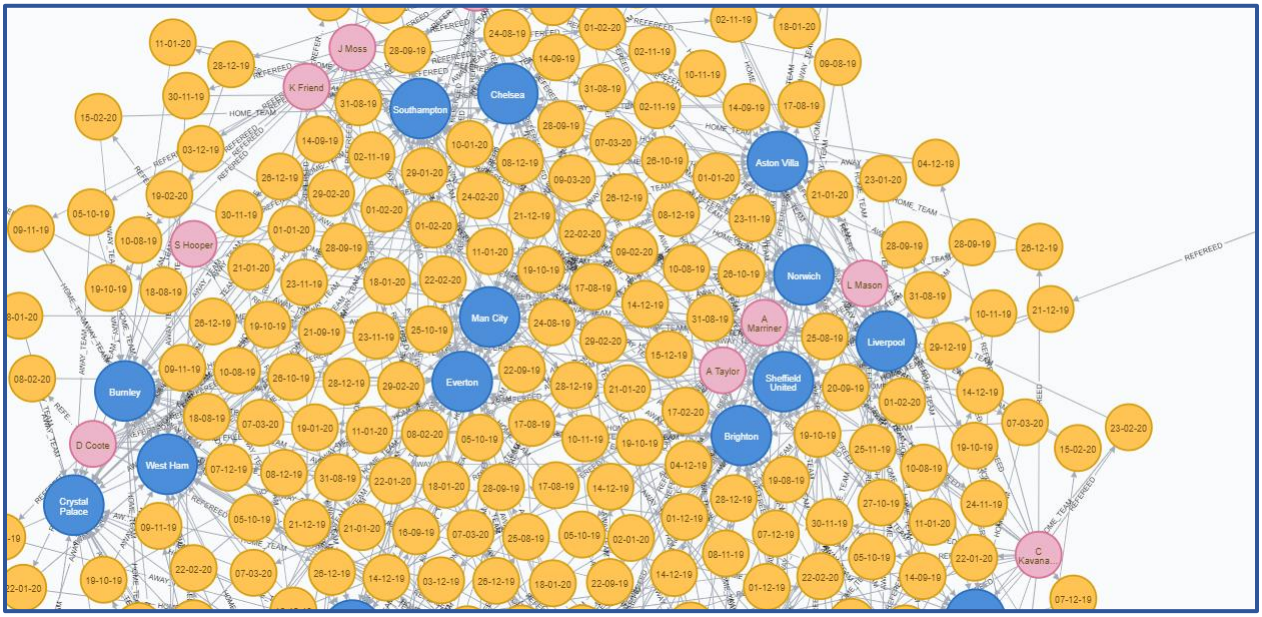


Figure 31: Zoom in view of graph

3. Show all the EPL team involved in the season.

```
// Cypher query to display all teams
MATCH (team:Team)
RETURN DISTINCT team as Team_involved_in_the_season
```

Figure 32: Cypher query to show all EPL teams

```
neo4j$ MATCH (team:Team) RETURN DISTINCT team as Team_involved_in_the_season
```

Graph | Table | Text | Code

"Team_involved_in_the_season"
{"teamName":"Liverpool"}
{"teamName":"West Ham"}
{"teamName":"Bournemouth"}
{"teamName":"Burnley"}
{"teamName":"Crystal Palace"}
{"teamName":"Watford"}
{"teamName":"Tottenham"}
{"teamName":"Leicester"}
{"teamName":"Newcastle"}
{"teamName":"Man United"}
{"teamName":"Arsenal"}

Figure 33: All EPL teams

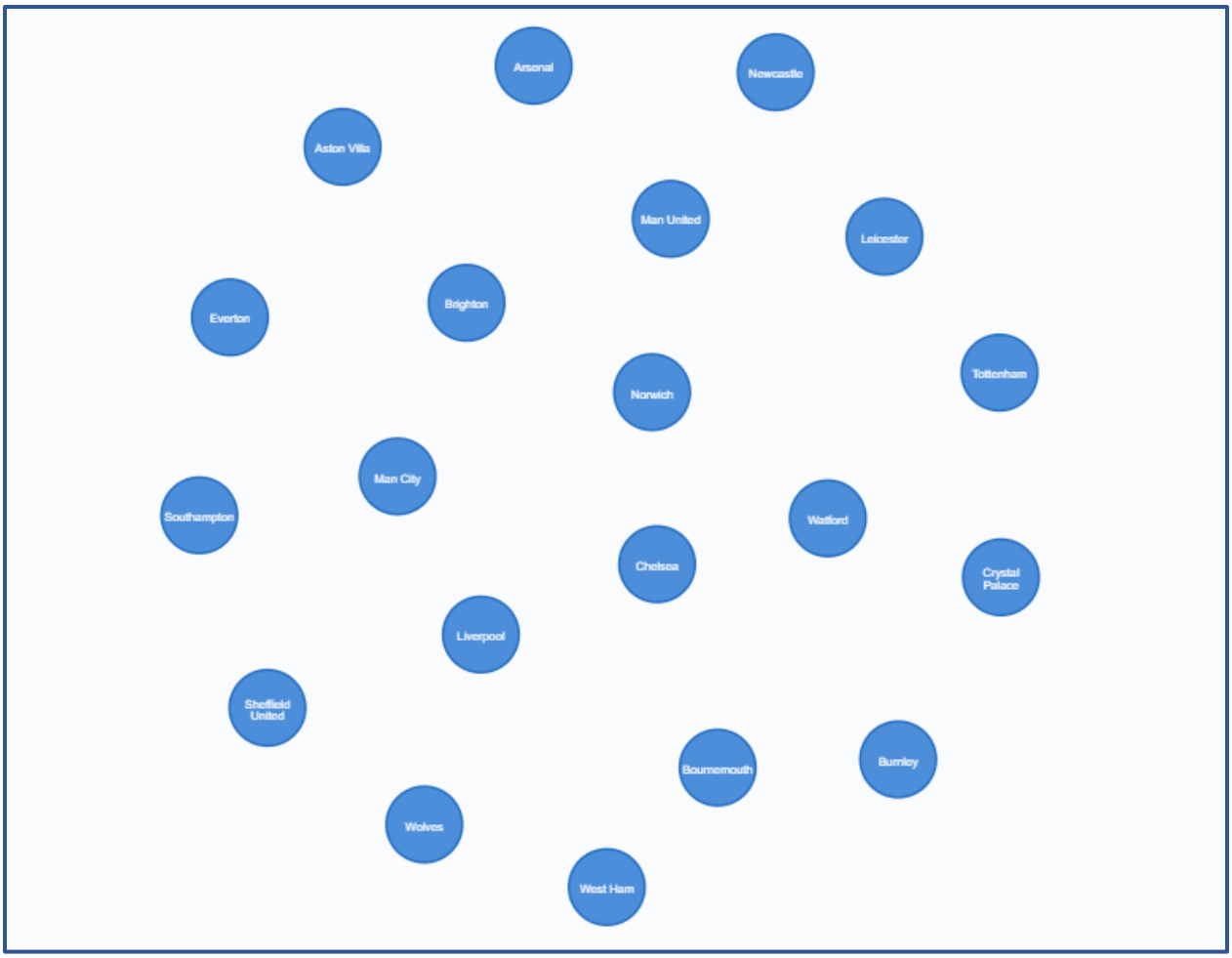


Figure 34: All EPL teams Nodes

4. Count all the matches refereed by each referee.

```
// 4. All the matches refereed by each referee
MATCH(ref:Referee)-[:REFEREED]→(matches:EplMatch)
RETURN ref.name as Referee_Name, COUNT(matches) as All_Matches_Refereed
```

Figure 35: Cypher query to show matches refereed by each referee

neo4j$ MATCH(ref:Referee)-[:REFEREED]→(matches:EplMatch) RETURN ref.…

Table | Text | Code

Referee_Name	All_Matches_Refereed
"M Oliver"	23
"M Dean"	22
"K Friend"	20
"G Scott"	14
"J Moss"	20
"C Pawson"	15
"C Kavanagh"	20
"A Marriner"	17
"M Atkinson"	25
"A Taylor"	23
"L Mason"	13
"S Attwell"	15
"D Coote"	11
"O Langford"	1
"P Tierney"	20
"A Madley"	9
"P Bankes"	7
"S Hooper"	9
"T Robinson"	1
"R Jones"	1
"S Scott"	1
"D England"	1

Figure 36: Number of matches refereed by each referee

5. Who refereed the most matches?

```
MATCH(matches:EplMatch)--(ref:Referee)
WITH ref.name AS Referee, COUNT(matches) AS Number_of_matches_refereed
WITH max(Number_of_matches_refereed) AS Max_refereed_matches
MATCH(matches:EplMatch)--(ref:Referee)
WITH ref.name AS Referee, COUNT(matches) AS Number_of_matches_refereed, Max_refereed_matches
WHERE Number_of_matches_refereed = Max_refereed_matches
RETURN Referee AS Referee, Number of matches refereed AS Refereed max matches
```

Figure 37: Cypher query to show who refereed the most matches

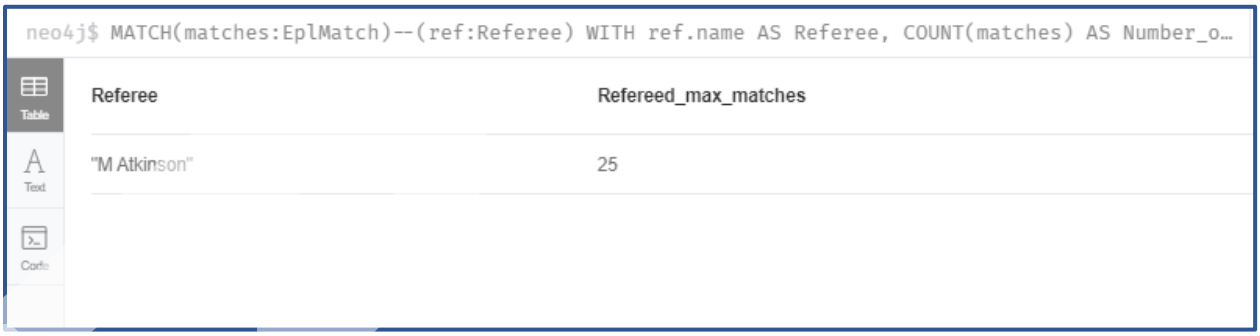


Figure 38: Referee who refereed the most matches

6. How many matches “Arsenal” won as the away team?

```
// 6. Arsenal won matches as away team
MATCH(matches:EplMatch)-[:AWAY_TEAM]→(awayTeam:Team)
WHERE awayTeam.teamName = "Arsenal" AND matches.fullTimeResult = "A"
RETURN awayTeam.teamName AS Team, COUNT(*) AS Matches_won_as_Away_Team
```

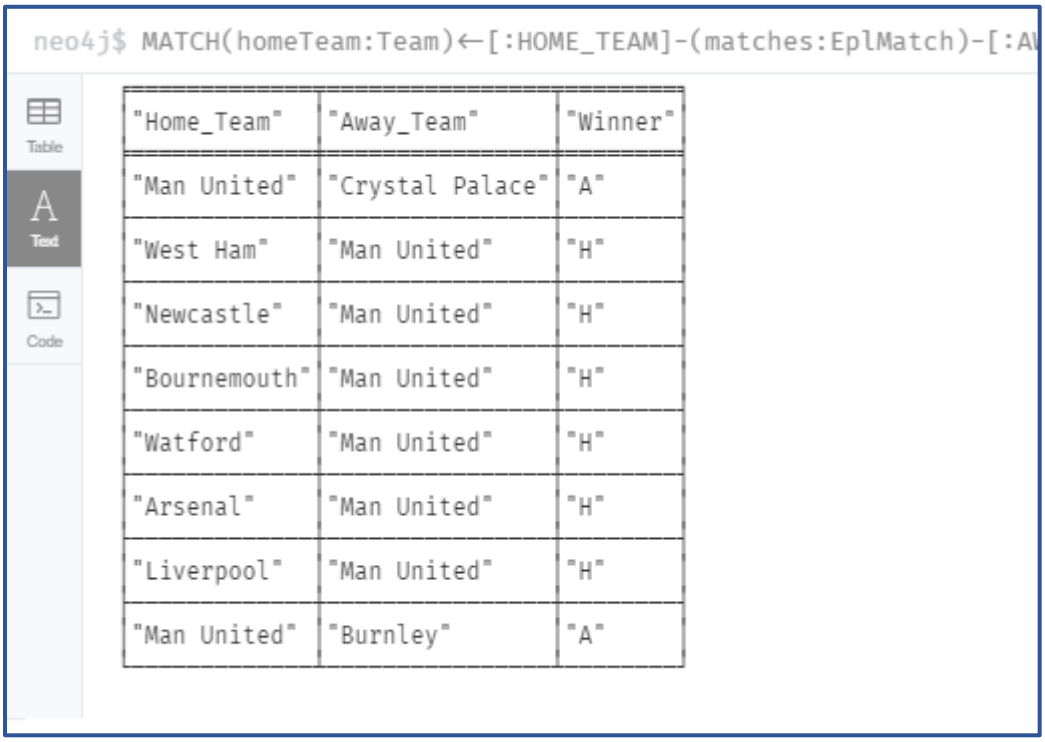
Figure 39: Cypher query to show matches "Arsenal" won as away team

Figure 40: Matches "Arsenal" won as away team

7. Display all the matches that “Man United” lost.

```
// 7. Display all the matches that “Man United” lost.
MATCH(homeTeam:Team)←[:HOME_TEAM]-(matches:EplMatch)-[:AWAY_TEAM]→(awayTeam:Team)
WHERE (awayTeam.teamName = "Man United" AND matches.fullTimeResult = "H") OR (homeTeam.teamName = "Man United" AND
matches.fullTimeResult = "A")
RETURN homeTeam.teamName AS Home_Team, awayTeam.teamName AS Away_Team, matches.fullTimeResult As Winner
```

Figure 41: Cypher query to show matches that "Man United' lost.



"Home_Team"	"Away_Team"	"Winner"
"Man United"	"Crystal Palace"	"A"
"West Ham"	"Man United"	"H"
"Newcastle"	"Man United"	"H"
"Bournemouth"	"Man United"	"H"
"Watford"	"Man United"	"H"
"Arsenal"	"Man United"	"H"
"Liverpool"	"Man United"	"H"
"Man United"	"Burnley"	"A"

Figure 42: Matches that "Man United' lost.

8. Display all the matches that “Liverpool” won but were down in the first half.

```
MATCH(homeTeam:Team)←[:HOME_TEAM]-(matches:EplMatch)-[:AWAY_TEAM]→(awayTeam:Team)
WHERE (homeTeam.teamName = "Liverpool" AND matches.halfTimeResult = "A" AND matches.fullTimeResult = "H") OR
(awayTeam.teamName = "Liverpool" AND matches.halfTimeResult = "H" AND matches.fullTimeResult = "A")
RETURN matches.date AS Match_date, matches.time AS Time, homeTeam.teamName AS Home_Team, awayTeam.teamName AS Away_Team,
matches.halfTimeResult As Half_Time_Winner, matches.fullTimeResult As Full_Time_Winner
```

Figure 43: Cypher query to show all matches that "Liverpool" won but were down in first half

Figure 44: All matches that "Liverpool" won but were down in first half

Task 3:

"Column-oriented storage in a database system are more suitable for analytical reporting than the row-oriented database." Justify this statement with suitable example.

With different memory hierarchy the latency cost related to it also differs. It determines various ways for storing and sharing data. Especially for analytical reporting or any other analytical application the orientation and alignment of data on the storage device can impact significantly.

In a row-oriented database all the values related to a specific row are stored consecutively in memory. Mostly row-oriented design is employed on traditional database. These designs are suitable for applications that works on manipulating any records related to a limited number of transactions at a time, like transaction processing applications.

On the other hand, big data analytics applications work on a large dataset. It performs scanning, aggregating, and summarizing of large datasets. To get a small set of columns, using multiway join and accessing complete rows will increase the execution time. Using a row-oriented design, to get a required attribute, complete records must be read which means it goes through more data than is needed to complete the request. Row-oriented design is usually misaligned with memory systems and as a result of increased access latencies. As well as, row-oriented design does not enable joins and aggregation required to execute analytical queries with desired level of performance.

That's the reason a column-oriented database is more preferred for analytical reporting. It helps to reduce the impacts of data latency. In column-oriented database, values for each column are stored separately which allows to access specific column values efficiently. (Loshin 2013)

The simplicity of the columnar approach provides many benefits over row-oriented database:

- Access performance:
  Example: SELECT SUM(INCOME) FROM WORKER;
  In column-oriented database looping over all the income values in a column and adding is faster by using single instruction multiple data (SIMD) techniques to do perform a parallel sum operation. Whereas the same operation will be slow in row-oriented database.
  Whereas row-oriented database has limitations in diverse queries.
- Join and aggregation:
  Example: SELECT COUNT(*) FROM WORKER WHERE INCOME >= 100000 AND INCOME <= 200000;
  Parallel multiple unit processing, accessing and aggregating different columns increases the performance.
  Whereas in row-oriented database the need for streaming entire records to find the join attributes hampers the performance severely.
- Compression:
  Due to columnar layout of data, compression is easy to apply maintaining the performance without a significant decrease in storage.
  Whereas in row-oriented database it is very difficult to apply compression that increases performance.

- Data loading speed:
  Example: SELECT SALES_ID, SHIPPING_ADDRESS FROM SALES_ACC WHERE LOCATION = 'KTM';
  Data is segregated by column in column-oriented database which allows the system to load columns in parallel using multiple threads.
  Whereas in row-oriented database, all of the data values are stored together which prevent parallel processing.
  (Why Are Column-Oriented Database More Suitable For Analytics Than The Traditional Row-Oriented Database? - Quora 2020)

References

1. Loshin, D. (2013) *Big Data Analytics*. Amsterdam: Elsevier/Morgan Kaufmann
2. *Why Are Column-Oriented Database More Suitable For Analytics Than The Traditional Row-Oriented Database? - Quora* (2020) available from <https://www.quora.com/Why-are-column-oriented-database-more-suitable-for-analytics-than-the-traditional-row-oriented-database> [10 July 2020]