

**Final Project Written Submission****Team Name: Team Socios**

Vinay Kumar Thadaka (ASU ID: 1223433926)

Saketh Velidimalla (ASU ID: 1225618875)

Sharan Kumar Mogili (ASU ID: 1225885780)

Department of Information Technology, Arizona State University

IFT 530: Advanced Database Management Systems

**Project Title: La Liga Database Management System**

Dr. Robert Rucker

December 4<sup>th</sup>, 2022

## Table of Contents

Section 1.....	3
Synopsis .....	3
Introduction.....	4
Expected value of the Project.....	5
Questions to be answered.....	6
Section 2.....	7
Job Flow Diagram.....	7
ORM diagram .....	8
Section 3.....	8
Relational Diagram .....	8
Cardinality Ratio.....	9
Design Decisions .....	9
Section 4.....	10
Database Description .....	10
Data in the tables.....	11
Constraints on the Database.....	20
Section 5.....	21
Stored Procedures .....	21
Triggers .....	25
Queries .....	26
Section 6 - Couchbase (NoSQL).....	31
Introduction.....	31
Buckets.....	31
Documents in the bucket.....	34
Analytics Service .....	36
Queries .....	36
Section 7.....	39
Summary .....	39
Conclusion .....	40
References.....	42

## Section 1

### Synopsis

The topic of interest for our team is Sports, and we have decided to go with football as our choice of sport particularly focusing on the Spanish football league also called as La Liga. The ORM diagram is generated using the CSDP stages. Based on the ORM diagram, the SQL code is constructed, which will build tables and populate the data in them. We will be creating two stored procedures and triggers to handle the data in our database.

The entities are Player, Game, Team, Manager, Stadium, SquadPlayer and SubstitutePlayer. Entities such as Game and Team have many to many-to-many relationship whereas the relationship between Team, Game and PlaysGameOnDate is a ternary one.

We created the social information database using this outline. We also used these materials to shape cans in Couchbase to focus on the JSON structure. The JSON structure makes it easier to address information in a fixed manner without having to use many tables joins. By using the relational database mentioned above, we aim to answer questions like –

- Which team is the best performing team based on number of wins?
- Which team is the poorly performing team based on number of wins?
- Which teams have performed the best as home and away teams?
- For a team that does well in games, what is the average height and weight of the team players?

We used saved systems and triggers to aid us in our investigation. The put away methodology helps us answer some of our questions, such as which group performs well as the host group and which group performs well as the away group. They also help us determine which group is the most solid and which is the most vulnerable based on the number of wins. CouchBase was used to investigate the NoSQL functionality. We also used CouchBase's inquiry administrations and investigation administrations.

## Introduction

We have chosen Sports as it gives us a wide range of options to choose from and the members of our team are sports enthusiasts. Almost all the countries today participate in different kinds of sports and as a result there is a large amount of data being generated. We have chosen one such sport i.e., Football, to create the database.

Football is one of the sports which has a huge popularity and following. As a result of which, winning is a game is a matter of great pride for the teams. Due to this popularity, there is a constant pressure of forming teams with the best performing players who can lead the team to victory.

The data gathered from the past games, attributes of teams and players can be utilized to gain an understanding of what factors contribute to a team's good performance. Querying the database can give results that would be helpful in making important decisions about the teams.

The database that we will be implementing here will be a Football database to perform multiple operations to demonstrate our knowledge on database management.

Massive informational collections are used in modern training to nurture winning methods for both individual competitors and organizations. Expert group mentors, for example, can use information science to create hyper-customized competitor matches and different designs for each game they play. As a result, the group's strategies remain mysterious while being viable. The information gathered from previous games, qualities of groups, and players can be used to gain an understanding of what components contribute to a group's good presentation. To meet our needs, the data collection includes diverse substances displayed in several tables with varying quality. The data base contains a few substances that are displayed in several tables with diverse attributes, allowing us to perform different data set tasks to the information.

Different entities that we've chosen for this project are Manager, Player, Team, Game and Stadium.

There is a many-to-many relationship between Game and Group. Subtypes of Player include SquadPlayer and SubstitutePlayer.

We used Put away Techniques, UDFs, and Triggers to aid us in our investigation. We used saved systems to answer some of our questions, for example, which group is the most solid and most fragile in the organization based on the number of wins. We also attempted to determine which group performs well as the host group and which group performs well as the away group. Put away Strategy has also been used to determine a team's chances of dominating a match based on its previous game history.

For a game whose area isn't referenced, we used a Trigger to define the default incentive for Arena as the home ground of the group playing as host group. Another Trigger is also used, which converts the country names to title case.

### **Expected Value of the Project**

This venture, which focuses on basketball data analysis, can be used by sports teams to generate huge suspicions. These assumptions can be used to improve a group's presentation, devise a powerful gaming technique, and underline the group's weak points. This task's storage techniques, UDFs, and triggers are dynamic. New investigations can be drawn as new information becomes available. This will be valuable for an extended period. We can also modify the data set to record new traits in existing tables, create new tables, and remove section ascribes that are no longer required.

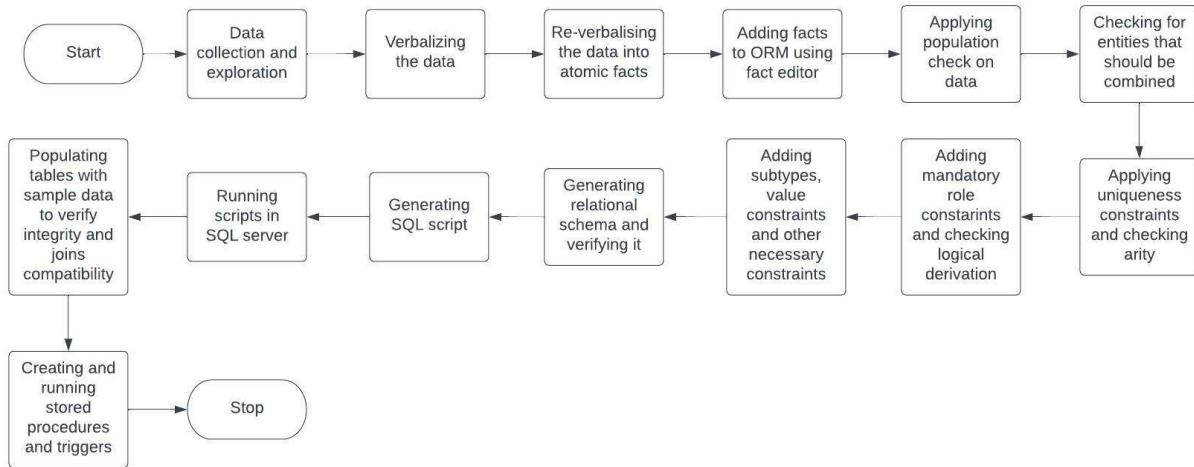
## Questions to be answered

We are aiming to find solutions to various questions using the data collection, such as -

- Which group is the most successful in terms of number of wins?
- Which group is underperforming in terms of the number of wins?
- Which group has performed better as a host group?
- Taking aside exhibitions into account, which group is the best?
- What is the average level and weight of cooperative persons for a most grounded group based on the number of wins?

## Section 2

### Job Flow Diagram



### ORM Diagram

The ORM diagram is built based on the knowledge we have about the entities and the relationships between them. The ORM model is built using the 7 CSDP steps. We have different entities in the ORM diagram such as Player, Game, Team, Manager and Stadium.

The entities have different attributes which are defined in the diagram. The different relationships between the entities are also defined in the diagram. The entity player has a many to one relationship with the entity team. The SquadPlayer and SubstitutePlayer are subsets of the entity Player. The entities Team and Game have a ternary relationship with the field PlaysGameOnDate. The other entities in the diagram are Manager and Stadium.

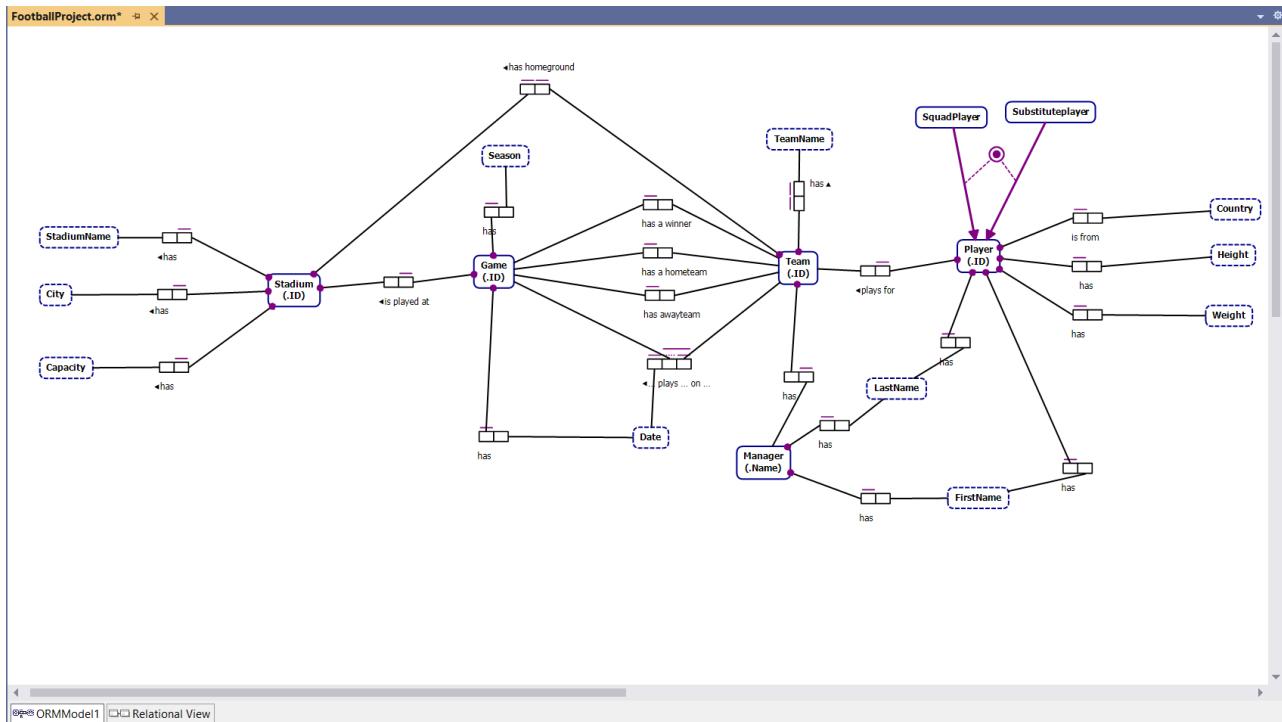


Figure 1: ORM diagram for La Liga Database

### Section 3

#### Relational Schema

NORMA programming is used to obtain the social chart from the ORM outline. This depicts the relationship between the substances based on the situation for data set display. It depicts what attributes the tables will have, as well as the necessary and unfamiliar keys of individual tables, as well as the information types of each property. This gives the client a clear picture of how the information base plan should seem. As can be seen from the outline, each table includes an ID/Name column that serves as the primary key. Every important key is unique and cannot be duplicated. The vital key is used to identify each outstanding entry in the table. When necessary, the unfamiliar keys assist in joining the various tables. The unfamiliar keys are essentially the essential keys in another table

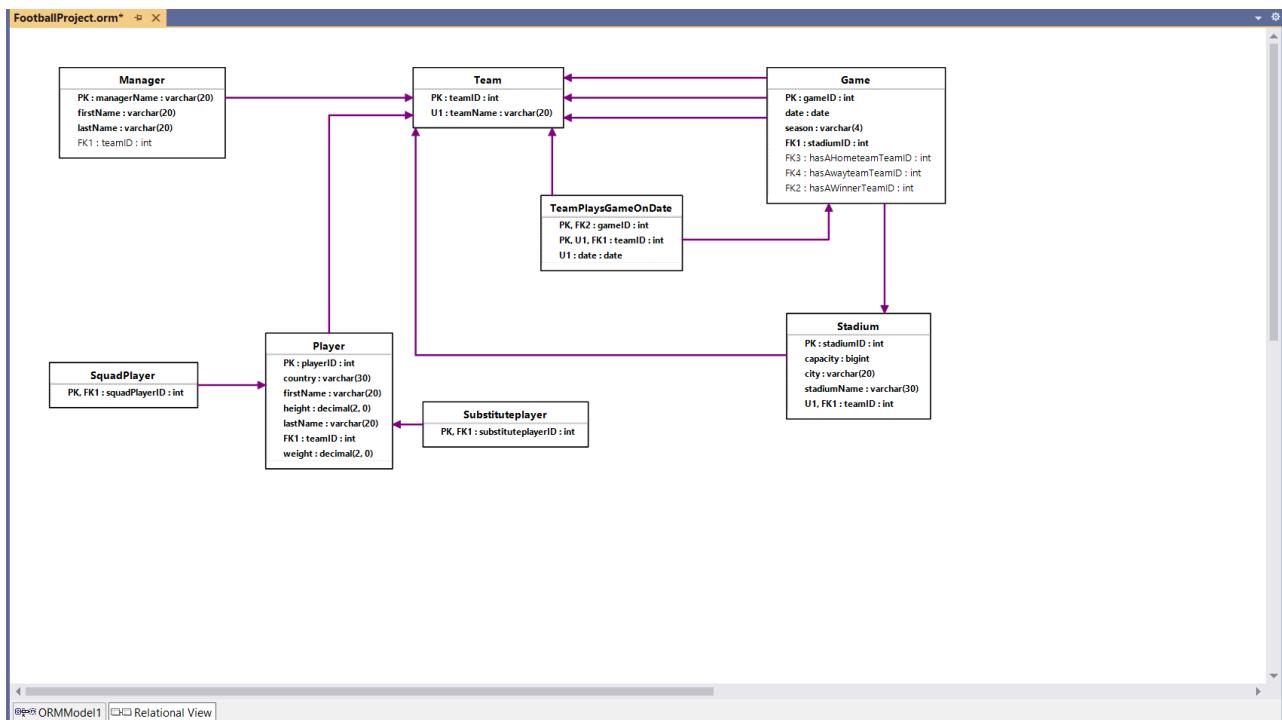


Figure 2: Relational Diagram

## Cardinality Ratio

Entity 1	Entity 2	Relationship
Player	Team	1:1
SquadPlayer	Player	1:1
SubstitutePlayer	Player	1:1
TeamPlaysGameOnDate	Team	1:M
Game	Team	1:M
Game	Stadium	1:1
Manager	Team	1:1

## Design Decisions

- A Player can be a SubstitutePlayer or a SquadPlayer, therefore a 1:1 relationship.
- A Player may play only for one Team, hence 1:1 relationship.
- On a given date many teams can play games, which is why 1:M relationship.
- A game can only be played between two teams, hence 1:M relationship.
- On a given date a game can be played at one stadium, hence 1:1 relationship.
- A Manager can manage only one team, hence 1:1 relationship.

## Section 4

### Database Description

The database has 8 tables which are –

Player, Team, Game, Manager, Stadium, TeamPlaysGameOnDate, SquadPlayer and SubstitutePlayer.

- Player has Primary Key as playerId and Foreign Key as teamId
- Team has Primary Key as teamId
- Manager has Primary Key as ManagerName and Foreign Key as teamId
- Game has has PrimaryKey as gameId and
  - i. Foreignkey1 as hasHometeamTeamId
  - ii. Foreignkey2 as hasAwayteamTeamId
  - iii. Foreignkey3 as hasAWinnerTeamId
  - iv. Foreignkey4 as stadiumId
- Stadium has Primary Key as stadiumId and Foreign Key as teamId
- SquadPlayer has both primary key(PK) and ForeignKey(FK1) as SquadPlayerId which is integer data type.
- SubstitutePlayer has both primary key(PK) and ForeignKey(FK1) as substitutePlayerId which is integer data type.
- TeamPlaysGameOnDate has Primary Key as date and
  - i. Foreignkey1 as teamId
  - ii. Foreignkey2 as gameId

## Data in Tables

### Team Table –

*DDL Statements:*

```
CREATE TABLE Team(
    teamId int IDENTITY (1, 1) NOT NULL,
    teamName nvarchar(20) NOT NULL,
    CONSTRAINT Team_PK PRIMARY KEY(teamId),
    CONSTRAINT Team_UC UNIQUE(teamName)
)
```

*DML Statements:*

```
insert into Team(teamName) values ('FC Barcelona');
insert into Team(teamName) values ('Sevilla FC');
insert into Team(teamName) values ('Real Madrid');
insert into Team(teamName) values ('Athletic Club');
insert into Team(teamName) values ('Villarreal CF');
insert into Team(teamName) values ('Atletico De Madrid');
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure. Two queries are running in the center pane: 'SQLQuery2.sql' and 'SQLQuery1.sql'. The 'Results' tab for 'SQLQuery1.sql' displays the following data:

teamId	teamName
1	FC Barcelona
2	Sevilla FC
3	Real Madrid
4	Athletic Club
5	Villarreal CF
6	Atletico De Madrid

The Properties window on the right shows connection details like connection name, state, and session tracing ID. The status bar at the bottom indicates the query was executed successfully.

## Game Table –

*DDL Statements:*

```
CREATE TABLE Game(
    gameId int IDENTITY (1, 1) NOT NULL,
    "date" date NOT NULL,
    season nchar(4) NOT NULL,
    stadiumId int NOT NULL,
    hasAwayteamTeamId int,
    hasAWinnerTeamId int,
    hasHometeamTeamId int,
    CONSTRAINT Game_PK PRIMARY KEY(gameId)
)
```

*DML Statements:*

```
SET IDENTITY_INSERT Game ON
insert into Game(gameId, date, season, stadiumId, hasHometeamTeamId,hasAwayteamTeamId,
hasAWinnerTeamId) values (1, '2021-01-01',
'2021', 2, 2, 1, 2)
insert into Game(gameId, date, season, stadiumId, hasHometeamTeamId,hasAwayteamTeamId,
hasAWinnerTeamId) values (2, '2021-01-15',
'2021', 3, 3, 4, 4)
insert into Game(gameId, date, season, stadiumId, hasHometeamTeamId,hasAwayteamTeamId,
hasAWinnerTeamId) values (3, '2021-01-30',
'2021', 6, 6, 5, 6)
insert into Game(gameId, date, season, stadiumId, hasHometeamTeamId,hasAwayteamTeamId,
hasAWinnerTeamId) values (4, '2021-02-01',
'2021', 2, 2, 4, 2)
insert into Game(gameId, date, season, stadiumId, hasHometeamTeamId,hasAwayteamTeamId,
hasAWinnerTeamId) values (5, '2021-02-15',
'2021', 4, 4, 6, 4)
insert into Game(gameId, date, season, stadiumId, hasHometeamTeamId,hasAwayteamTeamId,
hasAWinnerTeamId) values (6, '2021-02-28',
'2021', 6, 6, 2, 2)
```

```

SET IDENTITY_INSERT Game ON
Insert into Game(gameId, date, season, stadiumId, hasHomeTeamId, hasAwayTeamId, hasWinnerTeamId) values (1, '2021-01-01', '2021', 2, 2, 1, 2)
Insert into Game(gameId, date, season, stadiumId, hasHomeTeamId, hasAwayTeamId, hasWinnerTeamId) values (2, '2021-01-15', '2021', 3, 3, 4, 4)
Insert into Game(gameId, date, season, stadiumId, hasHomeTeamId, hasAwayTeamId, hasWinnerTeamId) values (3, '2021-01-30', '2021', 6, 6, 5, 6)
Insert into Game(gameId, date, season, stadiumId, hasHomeTeamId, hasAwayTeamId, hasWinnerTeamId) values (4, '2021-02-01', '2021', 2, 2, 4, 2)
Insert into Game(gameId, date, season, stadiumId, hasHomeTeamId, hasAwayTeamId, hasWinnerTeamId) values (5, '2021-02-15', '2021', 4, 4, 6, 4)
Insert into Game(gameId, date, season, stadiumId, hasHomeTeamId, hasAwayTeamId, hasWinnerTeamId) values (6, '2021-02-28', '2021', 6, 6, 2, 2)

select * from Game

```

gameId	date	season	stadiumId	hasHomeTeamId	hasAwayTeamId	hasWinnerTeamId
1	2021-01-01	2021	2	1	2	2
2	2021-01-15	2021	3	4	4	3
3	2021-01-30	2021	6	5	6	6
4	2021-02-01	2021	2	4	2	2
5	2021-02-15	2021	4	6	4	4
6	2021-02-28	2021	6	2	2	6

Query executed successfully.

## Player Table –

*DDL Statements:*

`CREATE TABLE Player`

```

(
    playerId int IDENTITY (1, 1) NOT NULL,
    firstName nvarchar(20) NOT NULL,
    height decimal(2,0) NOT NULL,
    lastName nvarchar(20) NOT NULL,
    country nvarchar(30) NOT NULL,
    teamId int NOT NULL,
    weight decimal(2,0) NOT NULL,
    CONSTRAINT Player_PK PRIMARY KEY(playerId)
)

```

*DML Statements:*

```

insert into Player(firstName, lastName, height, weight, country, teamId) values ('Ronald', 'Araujo', 6.4, 75, 'Uruguay', 1)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Pablo', 'Gavi', 5.8, 72, 'Spain', 1)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Pedro Gonzales', 'Lopez', 6.2, 78, 'Spain', 1)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Ivan', 'Rakitic', 6.4, 82, 'Croatia', 2)

```

```

insert into Player(firstName, lastName, height, weight, country, teamId) values ('Yassine', 'Bounou', 6.6, 81, 'Morocco', 2)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Marcos', 'Acuna', 6.7, 84, 'Argentina', 2)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Karim', 'Benzema', 6.5, 75, 'France', 3)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Luka', 'Modric', 6.8, 81, 'Croatia', 3)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Toni', 'Kroos', 6.4, 77, 'Germany', 3)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Inaki', 'Williams', 6.7, 82, 'Ghana', 4)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Nico', 'Williams', 6.0, 74, 'Spain', 4)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Unai', 'Simon', 6.0, 76, 'Spain', 4)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Gerard', 'Moreno', 6.5, 83, 'Spain', 5)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Giovani', 'Lo Celso', 6.5, 81, 'Argentina', 5)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Juan', 'Foyth', 6.0, 79, 'Argentina', 5)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Joao', 'Felix', 6.4, 80, 'Portugal', 6)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Antoine', 'Griezmann', 6.4, 78, 'France', 6)
insert into Player(firstName, lastName, height, weight, country, teamId) values ('Rodrigo', 'De Paul', 6.9, 85, 'Argentina', 6)

```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. There are four tabs open in the query editor:

- SQLQuery1.sql - DESKTOP-26HAOS\SQLEXPRESS.FP (SA (55))
- SQLQuery2.sql - DE...PRESS.FP (SA (53))
- SQLQuery4.sql - DE...PRESS.FP (SA (54))
- SQLQuery3.sql - DE...PRESS.FP (SA (52))

The results tab displays the following data:

playerId	firstName	height	lastName	country	teamId	weight
1	Ronald	6	Araujo	Uruguay	1	75
2	Pablo	6	Gavi	Spain	1	72
3	Pedro Gonzales	6	Lopez	Spain	1	78
4	Ivan	6	Rekic	Croatia	2	82
5	Yassine	7	Bounou	Morocco	2	81
6	Marcos	7	Acuna	Argentina	2	84
7	Karim	7	Benzema	France	3	75
8	Luka	7	Modric	Croatia	3	81
9	Toni	6	Kroos	Germany	3	77
10	Inaki	7	Williams	Ghana	4	82
11	Nico	6	Williams	Spain	4	74
12	Unai	6	Simon	Spain	4	76
13	Gerard	7	Moreno	Spain	5	83
14	Giovani	7	Lo Celso	Argentina	5	81
15	Juan	6	Foyth	Argentina	5	79

The status bar at the bottom indicates "Query executed successfully." and shows system information like "DESKTOP-26HAOS\SQLEXPRESS - SA (55) FP 00:00:00 | 18 rows".

## Manager Table –

*DDL Statements:*

```
CREATE TABLE Manager(
    ManagerName nvarchar(20) NOT
    NULL, firstName nvarchar(20) NOT
    NULL, lastName nvarchar(20) NOT
    NULL, teamId int,
    CONSTRAINT Manager_PK PRIMARY KEY(ManagerName)
)
```

*DML Statements:*

```
insert into Manager(ManagerName, firstName, lastName, teamId) values
('XH','Xavier','Hernandez', 1)
insert into Manager(ManagerName, firstName, lastName, teamId) values ('JL','Julan','Lopetegui',
2)
insert into Manager(ManagerName, firstName, lastName, teamId) values ('CA','Carlo','Ancelotti',
3)
insert into Manager(ManagerName, firstName, lastName, teamId) values ('EV','Ernesto','Valverde',
4)
insert into Manager(ManagerName, firstName, lastName, teamId) values ('UE','Unai','Emery', 5)
insert into Manager(ManagerName, firstName, lastName, teamId) values ('DS','Diego','Simeone',
6)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer pane on the left shows a database named 'DESKTOP-26HAOS\SQL'. The central pane displays a query window with the following code and results:

```
CREATE TABLE Manager (
    ManagerName nvarchar(20) NOT NULL, firstName nvarchar(20) NOT NULL, lastName nvarchar(20) NOT NULL, teamId int,
    CONSTRAINT Manager_PK PRIMARY KEY(ManagerName)
)

insert into Manager(ManagerName, firstName, lastName, teamId) values ('XH','Xavier','Hernandez', 1)
insert into Manager(ManagerName, firstName, lastName, teamId) values ('JL','Julan','Lopetegui', 2)
insert into Manager(ManagerName, firstName, lastName, teamId) values ('CA','Carlo','Ancelotti', 3)
insert into Manager(ManagerName, firstName, lastName, teamId) values ('EV','Ernesto','Valverde', 4)
insert into Manager(ManagerName, firstName, lastName, teamId) values ('UE','Unai','Emery', 5)
insert into Manager(ManagerName, firstName, lastName, teamId) values ('DS','Diego','Simeone', 6)

Select * from Manager;
```

The Results pane shows the following data:

ManagerName	firstName	lastName	teamId
CA	Carlo	Ancelotti	3
DS	Diego	Simeone	6
EV	Ernesto	Valverde	4
JL	Julan	Lopetegui	2
UE	Unai	Emery	5
XH	Xavier	Hernandez	1

The Properties pane on the right provides connection details:

- Current connection parameters: Aggregate Status, Connection failure, Elapsed time 00:00:00.116, Finish time 11/29/2022 8:39:40 PM, Name DESKTOP-26HAOS\SQL, Rows returned 6, Start time 11/29/2022 8:39:40 PM, State Open.
- Connection: Connection name DESKTOP-26HAOS\SQL, Connection elapse 00:00:00.116, Connection finish t 11/29/2022 8:39:40 PM, Connection rows n 6, Connection start b 11/29/2022 8:39:40 PM, Connection state Open, Display name DESKTOP-26HAOS\SQL, Login name SA, Server name DESKTOP-26HAOS\SQL, Server version 11.0.3156, Session Tracing ID, SPID 56.

The status bar at the bottom indicates: DESKTOP-26HAOS\SQLEXPRESS - SA (56) | FP | 00:00:00 | 6 rows | Ln 13 | Col 1 | Ch 1 | INS | 12°C Clear | 8:39 PM | 11/29/2022.

## Stadium Table –

*DDL Statements:*

```
CREATE TABLE Stadium(
    stadiumId int IDENTITY (1, 1) NOT NULL,
    capacity as stadiumID + 10000,
    city nvarchar(20) NOT NULL,
    stadiumName nvarchar(30) NOT NULL,
    teamId int NOT NULL,
    CONSTRAINT Stadium_PK PRIMARY KEY(stadiumId),
    CONSTRAINT Stadium_UC UNIQUE(teamId)
)
```

*DML Statements:*

```
insert into Stadium(city, stadiumName, teamId) values ('Barcelona', 'Camp Nuo', 1)
insert into Stadium(city, stadiumName, teamId) values ('Sevilla', 'Ramon Sanchez-Pizjuan', 2)
insert into Stadium(city, stadiumName, teamId) values ('Madrid', 'Santiago Berneba', 3)
insert into Stadium(city, stadiumName, teamId) values ('Bilbao', 'San Mames', 4)
insert into Stadium(city, stadiumName, teamId) values ('Villareal', 'Estadio de la ceramica', 5)
insert into Stadium(city, stadiumName, teamId) values ('Madrid', 'Wanda Metropolitana', 6)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer pane on the left shows the database structure. The central pane contains four query windows. The first window contains the DDL for creating the Stadium table. The second and third windows contain the DML statements for inserting data. The fourth window shows the results of the SELECT statement, displaying six rows of stadium information. The Properties pane on the right shows connection details like name, state, and session tracing ID.

```
CREATE TABLE Stadium (
    stadiumId int IDENTITY (1, 1) NOT NULL,
    capacity as stadiumID + 10000,
    city nvarchar(20) NOT NULL,
    stadiumName nvarchar(30) NOT NULL,
    teamId int NOT NULL,
    CONSTRAINT Stadium_PK PRIMARY KEY(stadiumId),
    CONSTRAINT Stadium_UC UNIQUE(teamId)
)

insert into Stadium(city, stadiumName, teamId) values ('Barcelona', 'Camp Nuo', 1)
insert into Stadium(city, stadiumName, teamId) values ('Sevilla', 'Ramon Sanchez-Pizjuan', 2)
insert into Stadium(city, stadiumName, teamId) values ('Madrid', 'Santiago Berneba', 3)
insert into Stadium(city, stadiumName, teamId) values ('Bilbao', 'San Mames', 4)
insert into Stadium(city, stadiumName, teamId) values ('Villareal', 'Estadio de la ceramica', 5)
insert into Stadium(city, stadiumName, teamId) values ('Madrid', 'Wanda Metropolitana', 6)

select * from Stadium;
```

stadiumId	capacity	city	stadiumName	teamId
1	10001	Barcelona	Camp Nuo	1
2	10002	Seville	Ramon Sanchez-Pizjuan	2
3	10003	Madrid	Santiago Berneba	3
4	10004	Bilbao	San Mames	4
5	10005	Villareal	Estadio de la ceramica	5
6	10006	Madrid	Wanda Metropolitana	6

Query executed successfully.

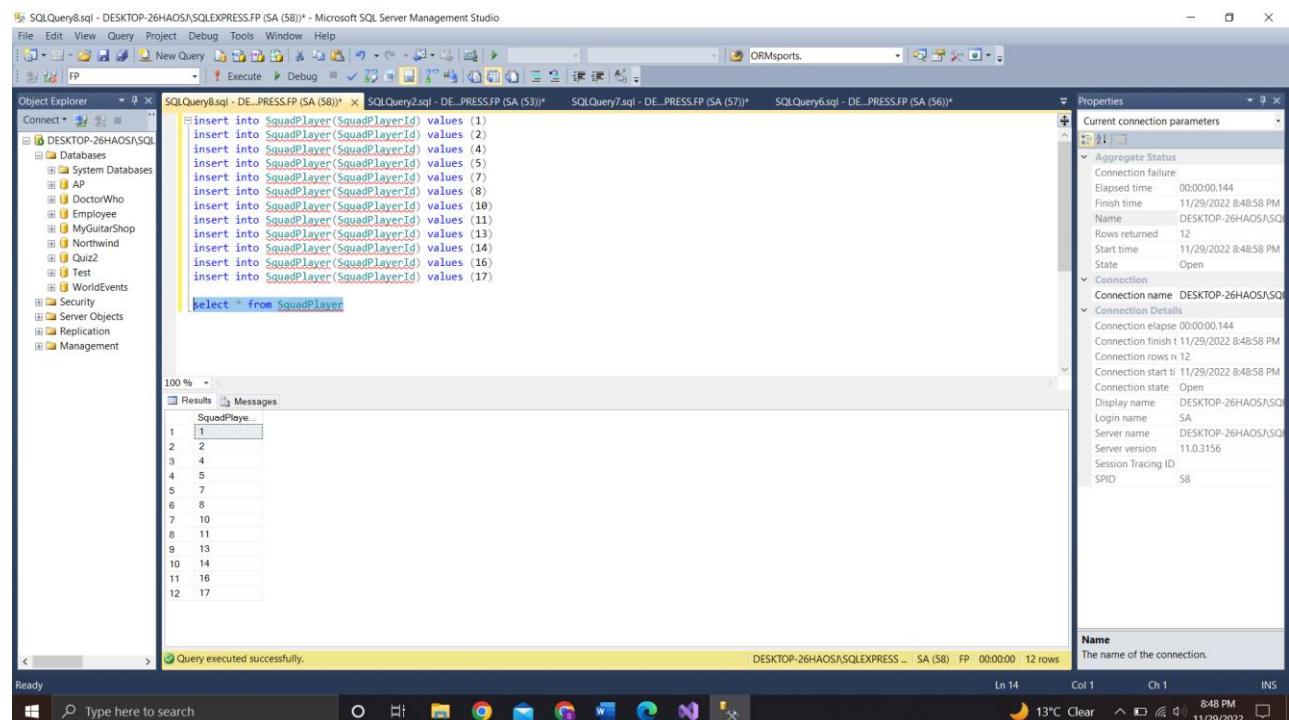
## Squad Player Table –

### *DDL Statements:*

```
CREATE TABLE SquadPlayer(
    SquadPlayerId int NOT NULL,
    CONSTRAINT SquadPlayer_PK PRIMARY KEY(SquadPlayerId)
)
GO
```

### *DML Statements:*

```
insert into SquadPlayer(SquadPlayerId) values (1)
insert into SquadPlayer(SquadPlayerId) values (2)
insert into SquadPlayer(SquadPlayerId) values (4)
insert into SquadPlayer(SquadPlayerId) values (5)
insert into SquadPlayer(SquadPlayerId) values (7)
insert into SquadPlayer(SquadPlayerId) values (8)
insert into SquadPlayer(SquadPlayerId) values (10)
insert into SquadPlayer(SquadPlayerId) values (11)
insert into SquadPlayer(SquadPlayerId) values (13)
insert into SquadPlayer(SquadPlayerId) values (14)
insert into SquadPlayer(SquadPlayerId) values (16)
insert into SquadPlayer(SquadPlayerId) values (17)
```



## Substitute Player Table –

*DDL Statements:*

```
CREATE TABLE SubstitutePlayer(
    substitutePlayerId int NOT NULL,
    CONSTRAINT SubstitutePlayer_PK PRIMARY KEY(substitutePlayerId)
)
```

*DML Statements:*

```
insert into SubstitutePlayer(substitutePlayerId) values (3)
insert into SubstitutePlayer(substitutePlayerId) values (6)
insert into SubstitutePlayer(substitutePlayerId) values (9)
insert into SubstitutePlayer(substitutePlayerId) values (12)
insert into SubstitutePlayer(substitutePlayerId) values (15)
insert into SubstitutePlayer(substitutePlayerId) values (18)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'DESKTOP-26HAOS\SQL' is selected. In the center pane, a query window displays the creation of the 'SubstitutePlayer' table and the insertion of six rows of data. The Properties pane on the right shows connection details for the current session. The status bar at the bottom indicates the query was executed successfully.

```
CREATE TABLE SubstitutePlayer (
    substitutePlayerId int NOT NULL,
    CONSTRAINT SubstitutePlayer_PK PRIMARY KEY(substitutePlayerId)
)

insert into SubstitutePlayer(substitutePlayerId) values (3)
insert into SubstitutePlayer(substitutePlayerId) values (6)
insert into SubstitutePlayer(substitutePlayerId) values (9)
insert into SubstitutePlayer(substitutePlayerId) values (12)
insert into SubstitutePlayer(substitutePlayerId) values (15)
insert into SubstitutePlayer(substitutePlayerId) values (18)

select * from SubstitutePlayer
```

Properties pane (partial view):

- Current connection parameters
- Aggregate Status: Connection failure
  - Elapsed time: 00:00:00.064
  - Finish time: 11/29/2022 8:50:16 PM
  - Name: DESKTOP-26HAOS\SQL
  - Rows returned: 6
  - Start time: 11/29/2022 8:50:16 PM
  - State: Open
- Connection
  - Connection name: DESKTOP-26HAOS\SQL
  - Connection Details
    - Connection elapse: 00:00:00.064
    - Connection finish: 11/29/2022 8:50:16 PM
    - Connection rows: 6
    - Connection start: 11/29/2022 8:50:16 PM
    - Connection state: Open
    - Display name: DESKTOP-26HAOS\SQL
    - Login name: SA
    - Server name: DESKTOP-26HAOS\SQL
    - Server version: 11.0.3156
    - Session Tracing ID: SPID: 58

Status bar: Query executed successfully. DESKTOP-26HAOS\SQLExpress - SA (58) FP 00:00:00 6 rows. Ln 13 Col 1 Ch 1 INS. 13°C Clear 8:50 PM 11/29/2022.

## TeamPlaysGameOnDate Table –

*DDL Statements:*

```
CREATE TABLE TeamPlaysGameOnDate(
```

```
    "date" date NOT NULL,
    teamId int NOT NULL,
    gameId int NOT NULL,
    CONSTRAINT TeamPlaysGameOnDate_PK PRIMARY KEY("date", teamId),
    CONSTRAINT TeamPlaysGameOnDate_UC UNIQUE(gameId, teamId)
```

```
)
```

*DML Statements:*

```
insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-01-01', 2, 1)
insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-01-15', 3, 2)
insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-01-30', 6, 3)
insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-02-01', 2, 4)
insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-02-15', 4, 5)
insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-02-28', 6, 6)
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'DE...PRESS.FP (SA (59))' is selected. In the center pane, a query window displays the creation of the 'TeamPlaysGameOnDate' table and the insertion of six rows of data. The Properties pane on the right shows connection details like connection name, elapsed time, and rows returned. The status bar at the bottom indicates the query was executed successfully.

```
--CREATE TABLE TeamPlaysGameOnDate (
--    "date" date NOT NULL,
--    teamId int NOT NULL,
--    gameId int NOT NULL,
--    CONSTRAINT TeamPlaysGameOnDate_PK PRIMARY KEY("date", teamId),
--    CONSTRAINT TeamPlaysGameOnDate_UC UNIQUE(gameId, teamId)

insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-01-01', 2, 1)
insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-01-15', 3, 2)
insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-01-30', 6, 3)
insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-02-01', 2, 4)
insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-02-15', 4, 5)
insert into TeamPlaysGameOnDate(date, teamId, gameId) values ('2021-02-28', 6, 6)
```

	date	teamId	gameId
1	2021-01-01	2	1
2	2021-01-15	3	2
3	2021-01-30	6	3
4	2021-02-01	2	4
5	2021-02-15	4	5
6	2021-02-28	6	6

## Constraints on the Database

```
ALTER TABLE Game ADD CONSTRAINT Game_FK1 FOREIGN KEY
(hasHometeamTeamId) REFERENCES Team (teamId) ON DELETE NO ACTION ON UPDATE
NO ACTION
GO
```

```
ALTER TABLE Game ADD CONSTRAINT Game_FK2 FOREIGN KEY
(hasAwayteamTeamId) REFERENCES Team (teamId) ON DELETE NO ACTION ON UPDATE
NO ACTION
GO
```

```
ALTER TABLE Game ADD CONSTRAINT Game_FK3 FOREIGN KEY
(hasAWinnerTeamId) REFERENCES Team (teamId) ON DELETE NO ACTION ON UPDATE
NO ACTION
GO
```

```
ALTER TABLE Game ADD CONSTRAINT Game_FK4 FOREIGN KEY
(stadiumId) REFERENCES Stadium (stadiumId) ON DELETE NO ACTION ON UPDATE NO
ACTION
GO
```

```
ALTER TABLE Manager ADD CONSTRAINT Manager_FK FOREIGN KEY (teamId)
REFERENCES Team (teamId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
```

```
ALTER TABLE Stadium ADD CONSTRAINT Stadium_FK FOREIGN KEY (teamId)
REFERENCES Team (teamId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
```

```
ALTER TABLE Player ADD CONSTRAINT Player_FK FOREIGN KEY (teamId)
REFERENCES Team (teamId) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
```

```
ALTER TABLE SquadPlayer ADD CONSTRAINT SquadPlayer_FK FOREIGN KEY
(SquadPlayerId) REFERENCES Player (playerId) ON DELETE NO ACTION ON UPDATE NO
ACTION
GO
```

```
ALTER TABLE SubstitutePlayer ADD CONSTRAINT SubstitutePlayer_FK FOREIGN KEY
(substitutePlayerId) REFERENCES Player (playerId) ON DELETE NO ACTION ON UPDATE
NO ACTION
GO
```

```
ALTER TABLE TeamPlaysGameOnDate ADD CONSTRAINT TeamPlaysGameOnDate_FK1
FOREIGN KEY (teamId) REFERENCES Team (teamId) ON DELETE NO ACTION ON
UPDATE NO ACTION
GO
```

```
ALTER TABLE TeamPlaysGameOnDate ADD CONSTRAINT TeamPlaysGameOnDate_FK2
FOREIGN KEY (gameId) REFERENCES Game (gameId) ON DELETE NO ACTION ON
UPDATE NO ACTION
GO
```

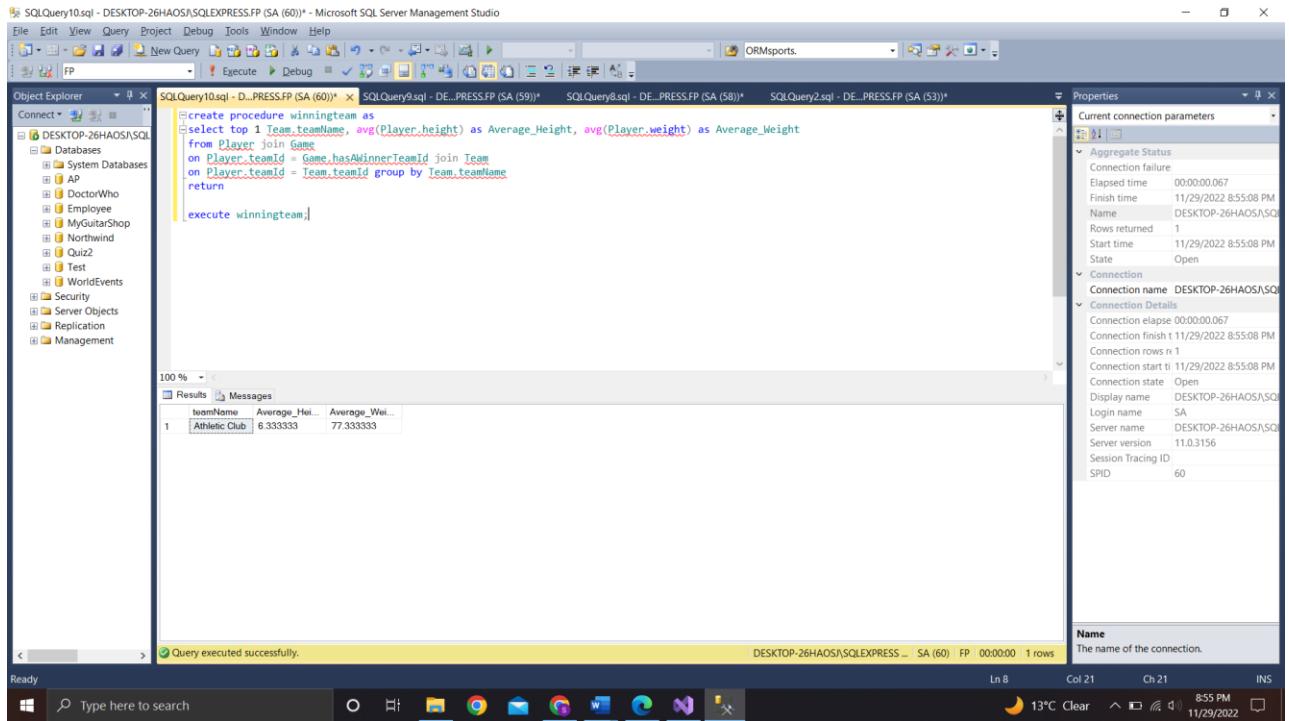
## Section 5

### Stored Procedures

1. Procedure to fetch the average height and weight of players in the best team based on the number of wins.

```
create procedure winningteam as
select top 1 Team.teamName, avg(Player.height) as Average_Height, avg(Player.weight) as
Average_Weight
from Player join Game
on Player.teamId = Game.hasAWinnerTeamId join Team
on Player.teamId = Team.teamId group by Team.teamName
return

execute winningteam;
```



## 2. Stored procedure to predict the winner in future games, based on previous games.

```

create procedure PredictWinner as
select Team.teamId as TID, Team.teamName as Probable_Winner, Game.hasHometeamTeamId as
PlayadasHomeTeam,
Game.hasAwayteamTeamId as PlayadasAwayTeam
from Team join Game on Team.teamId =
Game.hasAwayteamTeamId
or Team.teamId = Game.hasHometeamTeamId or Team.teamId = Game.hasAWinnerTeamId
where Team.teamName = any (select Team.teamName from Team join Game
on (Team.teamId = Game.hasAwayteamTeamId or Team.teamId = Game.hasHometeamTeamId)
and Team.teamId = Game.hasAWinnerTeamId
group by Team.teamName, Game.hasAwayteamTeamId, Game.hasHometeamTeamId,
Game.hasAWinnerTeamId) group by Team.teamId, Team.teamName,
Game.hasHometeamTeamId, Game.hasAwayteamTeamId
  
```

```

SQLQuery10.sql - DESKTOP-26HAOS\SQLEXPRESS.FP (SA (60)) - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query Execute Debug
Object Explorer
SQLQuery10.sql - DESKTOP-26HAOS\SQLEXPRESS.FP (SA (60))*
SQLQuery9.sql - DE...PRESS.FP (SA (59))*
SQLQuery8.sql - DE...PRESS.FP (SA (58))*
SQLQuery2.sql - DE...PRESS.FP (SA (53))*
ORMsports.
Properties
Current connection parameters
Aggregate Status
Connection failure
Elapsed time 00:00:00.313
Finish time 11/29/2022 8:56:39 PM
Name DESKTOP-26HAOS\SQL
Rows returned 9
Start time 11/29/2022 8:56:39 PM
State Open
Connection
Connection name DESKTOP-26HAOS\SQL
Connection Details
Connection elapse 00:00:00.313
Connection finish t 11/29/2022 8:56:39 PM
Connection rows n 9
Connection start t 11/29/2022 8:56:39 PM
Connection state Open
Display name DESKTOP-26HAOS\SQL
Login name SA
Server name DESKTOP-26HAOS\SQL
Server version 11.0.3156
Session Tracing ID
SPID 60
Name
The name of the connection.

Results Messages
TID Probable_Winner PlayedasHomeTe... PlayedasAwayTe...
1 2 Sevilla FC 2 1
2 2 Sevilla FC 2 4
3 2 Sevilla FC 6 2
4 4 Athletic Club 2 4
5 4 Athletic Club 3 4
6 4 Athletic Club 4 6
7 6 Atletico De Madrid 4 6
8 6 Atletico De Madrid 6 2
9 6 Atletico De Madrid 6 5

Query executed successfully.
DESKTOP-26HAOS\SQLEXPRESS... SA (60) FP 00:00:00 9 rows
Ln 12 Col 19 Ch 19 INS
Ready
Type here to search
13°C Clear 8:56 PM 11/29/2022

```

3. Procedure to get the number of wins of teams as home and away teams.

```

create procedure homeawaywinner as
select t.teamId, t.teamName,COUNT(t.teamId) as HOME_WINS from Team as t join Game
as g on t.teamId = g.hasAWinnerTeamId and t.teamId = g.hasHometeamTeamId
group by t.teamId,t.teamName
select t.teamId, t.teamName,COUNT(t.teamId) as AWAY_WINS from Team as t join Game
as g on t.teamId = g.hasAWinnerTeamId and t.teamId = g.hasAwayteamTeamId
group by t.teamId,t.teamName

```

```
execute homeawaywinner;
```

SQLQuery10.sql - DESKTOP-26HAOS\SQLEXPRESS.FP (SA (60)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

Object Explorer

SQLQuery10.sql - D...PRESS.FP (SA (60))\*

```

create procedure homeawaywinner as
select t.teamId, t.teamName,COUNT(t.teamId) as HOME_WINS from Team as t join Game as g on t.teamId = g.hasHomeTeamId and t.teamId = g.hasAwayTeamId
group by t.teamId,t.teamName
select t.teamId, t.teamName,COUNT(t.teamId) as AWAY_WINS from Team as t join Game as g on t.teamId = g.hasHomeTeamId and t.teamId = g.hasAwayTeamId
group by t.teamId,t.teamName
execute homeawaywinner;

```

Properties

Current connection parameters

Aggregate Status

Connection failure

Elapsed time 00:00:00.151

Finish time 11/29/2022 8:57:16 PM

Name DESKTOP-26HAOS\SO

Rows returned 5

Start time 11/29/2022 8:57:16 PM

State Open

Connection

Connection name DESKTOP-26HAOS\SO

Connection Details

Connection elapse 00:00:00.151

Connection finish 11/29/2022 8:57:16 PM

Connection rows n 5

Connection start ti 11/29/2022 8:57:16 PM

Connection state Open

Display name DESKTOP-26HAOS\SO

Login name SA

Server name DESKTOP-26HAOS\SO

Server version 11.0.3156

Session Tracing ID 60

Results Messages

teamId	teamName	HOME_WINS
1	Athletic Club	1
2	Atletico De Madrid	1
3	Seville FC	2

teamId	teamName	AWAY_WINS
1	Athletic Club	1
2	Seville FC	1

Query executed successfully.

DESKTOP-26HAOS\SQLEXPRESS... SA (60) FP 00:00:00 5 rows

Ready Type here to search

Ln 8 Col 1 Ch 1 INS

13°C Clear 8:57 PM 11/29/2022

## Triggers

- Trigger to capitalize the first letter in the country name

```
create trigger capitalize_country on Player after insert,update
as
update Player set country= UPPER(left(country,1)) + SUBSTRING(country,2,LEN(country))
where country in (select country from inserted);
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'DESKTOP-26HAOS\SQL' is selected. In the center pane, a query window displays the creation of a trigger:

```
create trigger capitalize_country on Player after insert,update
as
update Player set country= UPPER(left(country,1)) + SUBSTRING(country,2,LEN(country))
where country in (select country from inserted);
```

Below the trigger definition, an 'insert into Player...' statement is shown, followed by a 'select \* from Player' command. The results pane shows a table of player data, including columns: playerId, firstName, lastName, height, weight, country, teamId, and weight. The data includes rows for Yessine Bounou, Marcos Acuna, Karim Benzema, Luka Modric, Toni Kroos, Iñaki Williams, Nico Williams, Unai Simon, Gerard Moreno, Giovanni Lo Celso, Juan Foyth, Joao Felix, Antoine Griezmann, Rodrigo De Paul, and Gerard Pique.

The status bar at the bottom indicates the query was executed successfully on 'DESKTOP-26HAOS\SQL' at 9:00:05 PM on 11/29/2022, with 19 rows affected.

- Trigger to audit the entered game details

```
create trigger Audit_game on Game after insert as
declare @gameId int; declare @action varchar(50);
select @gameId = i.gameId from inserted as i; set @action = 'Inserted record at'
insert into Game_Audit (gameId, action, entrytime) values (@gameId, @action, GETDATE());
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query window with the following trigger creation script:

```

create trigger Audit_game on Game after insert as
declare @gameId int;
declare @action varchar(50);
select @gameId = i.gameId from inserted as i;
set @action = 'Inserted record at';
insert into Game_Audit(gameId, action, entrytime) values (@gameId, @action, GETDATE());

SET IDENTITY_INSERT Game on
insert into Game(gameId, date, season, stadiumId, hasHomeTeamId, hasAwayTeamId, hasWinnerTeamId) values (7, '2021-06-03', '2021', 1, 2, 3, 4)
select * from Game_Audit;

```

The results pane shows one row of data:

gameId	action	entrytime
1	Inserted record at	2022-11-29

The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-26HAOS\SQLEXPRESS ... SA (60) FP 00:00:00 1 rows".

## Section 5.1

### Queries

- Which team is the best performing team based on number of wins?

```

select top 1 Team.teamName as StrongestTeam, count(Game.hasAWinnerTeamId) as TotalWins from Team join Game
on Team.teamId = Game.hasAWinnerTeamId group by Team.teamName,
Game.hasAWinnerTeamId order by TotalWins Desc;

```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, there is a query window with the following SQL statement:

```

select top 1 Team.teamName as StrongestTeam, count(Game.hasAWinnerTeamId) as TotalWins from Team join Game
on Team.teamId = Game.hasAWinnerTeamId group by Team.teamName, Game.hasAWinnerTeamId order by TotalWins Desc;

```

The results pane shows one row of data:

StrongestTeam	TotalWi...
Sevilla FC	3

The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-26HAOS\SQLEXPRESS ... SA (60) FP 00:00:00 1 rows".

- Which team is the poorly performing team based on number of wins?

```
select top 1 Team.teamName as WeakestTeam, count(Game.hasAWinnerTeamId) as TotalWins from Team join Game on Team.teamId = Game.hasAWinnerTeamId group by Team.teamName, Game.hasAWinnerTeamId order by TotalWins Asc;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window contains the following SQL code:

```
select top 1 Team.teamName as WeakestTeam, count(Game.hasAWinnerTeamId) as TotalWins from Team join Game on Team.teamId = Game.hasAWinnerTeamId group by Team.teamName, Game.hasAWinnerTeamId order by TotalWins Asc;
```

The results pane shows a single row of data:

WeakestTeam	TotalWins
Atletico De Madrid	1

The status bar at the bottom indicates "Query executed successfully." and "1 rows". The properties pane on the right shows connection details for the current session.

- Which team has performed the best as a home team?

```
select top 1 t.teamId, t.teamName,COUNT(t.teamId) as HOME_WINS from Team as t join Game as g on t.teamId = g.hasAWinnerTeamId and t.teamId = g.hasHometeamTeamId group by t.teamId,t.teamName
```

SQLQuery14.sql - DESKTOP-26HAOS\SQLEXPRESS.FP (SA (60)) - Microsoft SQL Server Management Studio

File Edit View Query Project Debug Tools Window Help

Object Explorer

Connect > DESKTOP-26HAOS\SQLEXPRESS (60) > Databases > System Databases > AP > DoctorWho > Employee > FP > Database Diagrams > Tables > System Tables > FileTables > dbo.Game > dbo.Manager > dbo.Player > dbo.SquadPlayer > dbo.Stadium > dbo.SubstitutePlayer > dbo.Team > dbo.TeamPlaysGameOn > Views > Synonyms > Programmability > Service Broker > Storage > Security > MyGuitarShop > Northwind > Quiz2 > Test > WorldEvents > Security > Server Objects > Replication > Management

SQLQuery14.sql - DE...PRESS.FP (SA (60)) \* SQLQuery9.sql - DE...PRESS.FP (SA (59)) \* SQLQuery8.sql - DE...PRESS.FP (SA (58)) \* SQLQuery2.sql - DE...PRESS.FP (SA (53)) \*

```
select top 1 t.teamId, t.teamName, COUNT(t.teamId) as HOME_WINS from Team
as t join Game as g on t.teamId = g.hasAWinnerTeamId and t.teamId = g.hasHomeTeamTeamId
group by t.teamId,t.teamName
```

Properties

Current connection parameters

Aggregate Status

Connection failure

- Elapsed time 00:00:00.71
- Finish time 11/29/2022 9:21:09 PM
- Name DESKTOP-26HAOS\SQLEXPRESS.FP (SA (60))
- Rows returned 1
- Start time 11/29/2022 9:21:09 PM
- State Open

Connection

Connection name DESKTOP-26HAOS\SQLEXPRESS.FP (SA (60))

Connection Details

- Connection elapse 00:00:00.71
- Connection finish 11/29/2022 9:21:09 PM
- Connection rows n 1
- Connection start 11/29/2022 9:21:09 PM
- Connection state Open
- Display name DESKTOP-26HAOS\SQLEXPRESS.FP (SA (60))
- Login name SA
- Server name DESKTOP-26HAOS\SQLEXPRESS.FP
- Server version 11.0.3156
- Session Tracing ID
- SPID 60

Results

teamId	teamName	HOME_WINS
1	Athletic Club	1

Messages

Query executed successfully.

DESKTOP-26HAOS\SQLEXPRESS ... SA (60) FP 00:00:00 1 rows

Ready Type here to search

Ln 3 Col 29 Ch 29 INS

13°C, Clear 9:21 PM 11/29/2022

- Considering the away performances of teams, which team is the best?

```
select top 1 t.teamId, t.teamName,COUNT(t.teamId) as AWAY_WINS from Team
as t join Game as g on t.teamId = g.hasAWinnerTeamId and t.teamId =
g.hasAwayteamTeamId
group by t.teamId,t.teamName
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The query window contains the following SQL code:

```
select top 1 t.teamId, t.teamName,COUNT(t.teamId) as AWAY_WINS from Team
as t join Game as g on t.teamId = g.hasAWinnerTeamId and t.teamId =
g.hasAwayteamTeamId
group by t.teamId,t.teamName
```

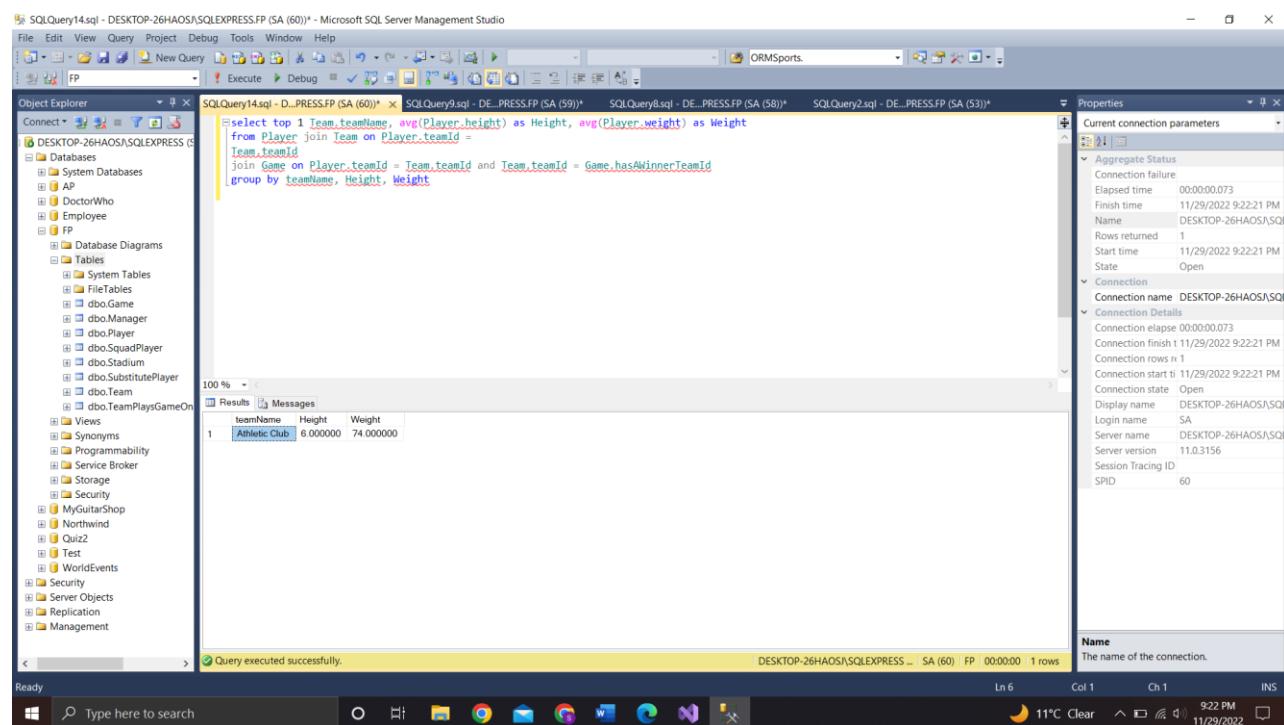
The results pane displays a single row of data:

teamId	teamName	AWAY_WINS
1	Athletic Club	1

The Properties pane on the right shows connection details, including the connection name, state, and server information.

- For a strongest team based on the number of wins, what is the average height and weight of the team players?

```
select top 1 Team.teamName, avg(Player.height) as Height, avg(Player.weight) as Weight
from Player join Team on Player.teamId =
Team.teamId
join Game on Player.teamId = Team.teamId and Team.teamId = Game.hasAWinnerTeamId
group by teamName, Height, Weight
```



## Section 6 - Couchbase (NoSQL)

### Introduction:

We have two buckets named Game and Team in the NoSQL database. The Game bucket maintains all information about the game such as the date it was played on, the season, the stadium ID, team IDs of the teams that played as home and away teams and the winning team ID. The Team bucket contains information such as the team Id, Name of the team, its home stadium (Id), and team players. In the instance of nested structures, we employed NoSQL to simplify queries and eliminate several table joins. The player names in the Team bucket are from the Player table and we used nested structures while loading the Team bucket in order to access the data. This is a highly helpful feature of NoSQL since it allows us to have nested structures that eliminate the need for separate tables and allows data to be stored in the same bucket as a subset of it.

### Buckets:

#### Game Bucket:

The screenshot shows the Couchbase Web UI interface. The left sidebar has navigation links like Dashboard, Servers, Buckets (which is selected), Backup, XDCR, Security, Settings, Logs, Documents, Query, Indexes, Search, Analytics, Eventing, and Views. The main content area is titled "SharanKumarMogili > Buckets". It displays the "Game" bucket details. The bucket type is Couchbase, with a Bucket RAM Quota of 150MB and a Cluster RAM Quota of 8.41GB. It has 1 replica and 1 server node. The conflict resolution is Sequence Number, compaction is Not active, compression is Passive, and the storage backend is CouchStore. The minimum durability level is none. The bucket is 100% resident with 0 ops/sec. RAM used is 30.8MB / 150MB, and disk used is 8.06MB. There are tabs for Documents and Scopes & Collections. Below the bucket details, there are memory and disk usage charts. A warning message at the bottom says "Warning: At least two servers with the data service are required to provide replication." The bottom of the screen shows the Windows taskbar with various pinned icons and system status.

#### JSON Document:

```
insert into Game (key, value) values ("1",
{ "gameId": "1",
"date": "2021-01-01",
```

```
"season": "2021",
"stadiumId": "2",
"hasHometeamTeamId": "2",
"hasAwayteamTeamId": "2",
"hasAWinnerteamId" : "2"
}),("2",
{ "gameId": "2",
"date": "2021-01-15",
"season": "2021",
"stadiumId": "3",
"hasHometeamTeamId": "3",
"hasAwayteamTeamId": "4",
"hasAWinnerteamId" : "4"
}),("3",
{ "gameId": "3",
"date": "2021-01-30",
"season": "2021",
"stadiumId": "6",
"hasHometeamTeamId": "6",
"hasAwayteamTeamId": "5",
"hasAWinnerteamId" : "6"
}),("4",
{ "gameId": "4",
"date": "2021-02-01",
"season": "2021",
"stadiumId": "2",
"hasHometeamTeamId": "2",
"hasAwayteamTeamId": "4",
"hasAWinnerteamId" : "2"
}),("5",
{ "gameId": "5",
"date": "2021-02-15",
"season": "2021",
"stadiumId": "4",
"hasHometeamTeamId": "4",
"hasAwayteamTeamId": "6",
"hasAWinnerteamId" : "4"
}),("6",
{ "gameId": "6",
"date": "2021-02-28",
"season": "2021",
"stadiumId": "6",
"hasHometeamTeamId": "6",
```

```
"hasAwayteamTeamId": "2" ,
"hasAWinnerteamId" : "2"
})
```

### Team Bucket:

The screenshot shows the Couchbase Server interface. On the left, a sidebar lists various management categories like Dashboard, Servers, Buckets, Backup, XDCR, Security, Settings, Logs, Documents, Query, Indexes, Search, Analytics, Eventing, and Views. The 'Buckets' category is currently selected. The main panel displays a table of buckets, with the 'Team' bucket highlighted. Below the table, detailed configuration information for the 'Team' bucket is shown, including its type (Couchbase), RAM quota (150MB), cluster RAM quota (8.41GB), replicas (1), server nodes (1), and ejection method (Value-Only). It also shows conflict resolution (Sequence Number), compaction status (Not active), compression (Passive), storage backend (CouchStore), and minimum durability level (none). A warning message at the bottom of the table area states: "Warning: At least two servers with the data service are required to provide replication." The bottom right corner of the screen shows the Windows taskbar with the date and time (11/29/2022, 9:26 PM) and weather (11°C, Clear).

name	items	resident	ops/sec	RAM used/quota	disk used	Documents	Scopes & Collections
Game	0	100%	0	30.8MiB / 150MiB	8.06MiB	Documents	Scopes & Collections
Sharan	15	100%	0	21.6MiB / 1.95GiB	16.1MiB	Documents	Scopes & Collections
<b>Team</b>	0	100%	0	30.8MiB / 150MiB	8.06MiB	Documents	Scopes & Collections
<b>Customer</b>	8	100%	0	23.5MiB / 2GiB	16.1MiB	Documents	Scopes & Collections
	9	100%	0	23.5MiB / 1.95GiB	16.1MiB	Documents	Scopes & Collections

### JSON Document:

```
insert into Team (key, value) values ("1",
{ "TeamId":"1",
"TeamName":"Fc Barcelona", "Stadiumid" : "1",
"TeamPlayers": { "Player 1":"Ronald Araujo","Player 2":"Pablo Gavi","Player 3":"Pedro
Gonzalez Lopez" }
}),("2",
{ "TeamId":"2",
"TeamName":"Athletic Club", "Stadiumid" : "2",
"TeamPlayers": { "Player 1":"Inaki Williams","Player 2":"Nico Williams","Player 3":"Unai
Simon" }
}),("3",
{ "TeamId":"3",
"TeamName":"Real Madrid", "Stadiumid" : "3",
"TeamPlayers": { "Player 1":"Luka Modric","Player 2":"Toni Kroos","Player 3":"Benzema" }
}),("4",
{ "TeamId":"4",
"TeamName":"Villareal FC", "Stadiumid" : "4",
"TeamPlayers": { "Player 1":"Gerard Moreno","Player 2":"Giovanni Lo Celso","Player 3":"Juan
Foyth" } })
```

```

}),("5",
{ "TeamId":"5",
"TeamName":"Athletico Madrid", "Stadiumid" : "5",
"TeamPlayers": { "Player 1":"Joao Felix","Player 2":"Antoinne Griezzman","Player 3":"Rodrigo De Paul"}
}),("6",
{ "TeamId":"6",
"TeamName":"Sevilla FC", "Stadiumid" : "6",
"TeamPlayers": { "Player 1":"Ivan Rakitic","Player 2":"Yassine Buono","Player 3":"Marcos Acuna"}
})
}

```

## Documents in bucket

	id	Document Content
	1	{"date":"2021-01-01","gameId":1,"hasAWinnerteamId":2,"hasAwayteamTeamId":2,"hasHometeamTeamId":2,"season":2021,"stadiumId":2}
	2	{"date":"2021-01-15","gameId":2,"hasAWinnerteamId":4,"hasAwayteamTeamId":4,"hasHometeamTeamId":3,"season":2021,"stadiumId":3}
	3	{"date":"2021-01-30","gameId":3,"hasAWinnerteamId":6,"hasAwayteamTeamId":5,"hasHometeamTeamId":6,"season":2021,"stadiumId":6}
	4	{"date":"2021-02-01","gameId":4,"hasAWinnerteamId":2,"hasAwayteamTeamId":4,"hasHometeamTeamId":2,"season":2021,"stadiumId":2}
	5	{"date":"2021-02-15","gameId":5,"hasAWinnerteamId":4,"hasAwayteamTeamId":6,"hasHometeamTeamId":4,"season":2021,"stadiumId":4}
	6	{"date":"2021-02-28","gameId":6,"hasAWinnerteamId":2,"hasAwayteamTeamId":2,"hasHometeamTeamId":6,"season":2021,"stadiumId":6}

The screenshot shows a web browser window with multiple tabs open. The active tab is titled 'localhost:8091/ui/index.html#/docs/editor?commonBucket=Sharan&scenarioZoom=minute&scenario=bcrok0uu6'. The interface is a document management system with a sidebar on the left containing navigation links like Dashboard, Servers, Buckets, Backup, XDCR, Security, Settings, Logs, Documents, Query, Indexes, Search, Analytics, Eventing, and Views. The main area has a search bar at the top with fields for Keyspace, bucket, scope, collection, Limit (set to 10), Offset (set to 0), Document ID (optional...), show range, NIQL WHERE (no indexes available...), and a 'Retrieve Docs' button. Below the search bar, it says '6 Results for Team,\_default,\_default, limit: 10, offset: 0'. There is a toggle switch for 'enable field editing' and buttons for '< prev batch' and 'next batch >'. A table lists six documents, each with an id (1 through 6) and a JSON representation of its contents. The JSON for each document describes a team with its stadium ID, team ID, team name, and team players.

	id	Content
1	1	{"Stadiumid": "1", "Teamid": "1", "TeamName": "Fc Barcelona", "TeamPlayers": {"Player 1": "Ronald Araujo", "Player 2": "Pablo Gavi", "Player 3": "Pedro Gonzalez Lopez"}}
2	2	{"Stadiumid": "2", "Teamid": "2", "TeamName": "Athletic Club", "TeamPlayers": {"Player 1": "Inaki Williams", "Player 2": "Nico Williams", "Player 3": "Unai Simon"}}
3	3	{"Stadiumid": "3", "Teamid": "3", "TeamName": "Real Madrid", "TeamPlayers": {"Player 1": "Luka Modric", "Player 2": "Toni Kroos", "Player 3": "Benzema"}}
4	4	{"Stadiumid": "4", "Teamid": "4", "TeamName": "Villareal FC", "TeamPlayers": {"Player 1": "Gerard Moreno", "Player 2": "Giovanni Lo Celso", "Player 3": "Juan Foyth"}}
5	5	{"Stadiumid": "5", "Teamid": "5", "TeamName": "Athletico Madrid", "TeamPlayers": {"Player 1": "Joao Felix", "Player 2": "Antoine Griezmann", "Player 3": "Rodrigo De Paul"}}
6	6	{"Stadiumid": "6", "Teamid": "6", "TeamName": "Sevilla FC", "TeamPlayers": {"Player 1": "Ivan Rakitic", "Player 2": "Yassine Buono", "Player 3": "Marcos Acuna"}}



## Analytics Service

Analytics Scopes, Links, & Collections

Map From Data Service

Default + remote link

Local cb local + collection

- Game
- Team
- customers
- orders
- skmCustomers\_sharan
- skmOrders\_sharan

Hide empty analytics scopes

## Queries

1. Query to find out the player names who belongs to a certain team played in a particular stadium.

```
FROM Game AS g JOIN Team AS t  
ON g.stadiumId = t.Stadiumid  
WHERE g.stadiumId = "4"  
SELECT t.TeamPlayers
```

The screenshot shows the Apache Ignite Workbench interface running in a browser window. The URL is `localhost:8091/ui/index.html#/cbas/workbench?commonBucket=Sharan&scenarioZoom=minute&scenario=bcr0k0uu6`. The main area displays a query editor with the following SQL code:

```
1 FROM Game AS g JOIN Team AS t ON g.stadiumId = t.stadiumid WHERE g.stadiumId = "4" SELECT t.TeamPlayers
```

The results are shown in JSON format:

```
1 [
2   {
3     "TeamPlayers": [
4       "Player 1": "Gerard Moreno",
5       "Player 2": "Giovanni Lo Celso",
6       "Player 3": "Juan Foyth"
7     ]
8   }
9 ]
```

On the right side, there's a sidebar titled "Analytics Scopes, Links, & Collections" which lists various scopes like Game, Team, customers, orders, and specific scopes for Sharan users.

## 2. Query to find out all the names of players in all the teams

from Team as t

```
SELECT t.TeamId, t.TeamName ,t.TeamPlayers, g.stadiumId
```

The screenshot shows the Apache CouchDB Analytics interface. On the left, there's a sidebar with various navigation options like Dashboard, Servers, Buckets, Backup, XDCR, Security, Settings, Logs, Documents, Query, Indexes, Search, Analytics (which is selected), Eventing, and Views. The main area has tabs for Workbench and Monitor, with Workbench being active. In the Workbench tab, the Query Editor contains the following code:

```
1 from Team as t
2 SELECT t.TeamId, t.TeamName ,t.TeamPlayers, g.stadiumId
```

Below the editor, there are two buttons: Execute and Explain. The Execute button is highlighted with a green checkmark and the text "success". The Explain button is greyed out. To the right of the editor, there's a status bar showing "elapsed: 91.90ms | execution: 76.76ms | docs scanned: 6 | docs returned: 6 | size: 914 bytes". There are also buttons for "format" and "query context".

The Query Results section displays the JSON output of the query. The results are as follows:

```
1 [
2   {
3     "TeamId": "2",
4     "TeamName": "Athletic Club",
5     "TeamPlayers": [
6       "Player 1": "Inaki Williams",
7       "Player 2": "Nico Williams",
8       "Player 3": "Unai Simon"
9     ]
10   },
11   {
12     "TeamId": "3",
13     "TeamName": "Real Madrid",
14     "TeamPlayers": [
15       "Player 1": "Luka Modric",
16       "Player 2": "Toni Kroos",
17       "Player 3": "Karim Benzema"
18     ]
19   }
]
```

Below the results, there are tabs for JSON, Table, Chart, Plan, and Plan Text. The JSON tab is selected. To the right of the results, there's a panel titled "Analytics Scopes, Links, & Collections" which lists "Default" and "Local" scopes. The "Default" scope includes "Game", "Team", "customers", "orders", "skmCustomers\_sharan", and "skmOrders\_sharan". The "Local" scope includes "cb local" and a "+ collection" option.

## Section 7

### **Summary**

We were able to obtain some useful analytics after properly constructing the ORM Diagram, Tables, and Stored Procedures. Using the number of wins each team has had in past games, we can predict who is likely to win in the future games and who is highly unlikely to win.

Based on the performance of the teams against each other in prior matches, we may also estimate which side will win in a given game. This provides insight into how teams should prepare for upcoming matchups as well as the flaws they face. We've also been able to anticipate whether a team will do better at home or away. This prediction will assist us in determining whether any team will have a home-field advantage. This will help the other team, allowing them to plan properly.

We also attempted to determine what characteristics of a player contribute to a good player and a strong team. This will aid in team building and determining a number for the sought-after player.

The tables containing the appropriate data in relational database management systems are related to one another, so the required data is fetched from the preceding tables, avoiding data redundancy.

Data access is privileged, which means the database administrator has the authority to provide data access to specific personnel, ensuring the data's security. However, the database uses tables with rows and columns, which necessitate a lot of physical capacity, which is one of the database's drawbacks.

To underscore its capacity to manage massive amounts of rapidly changing, unstructured data in ways that a relational (SQL) database with rows and tables cannot, NoSQL databases are frequently referred to as ‘non-relational’, ‘NoSQL DBs’, or ‘non-SQL’.

Because of its JSON structure, which allows us to have layered structures, NoSQL has helped us evaluate data quickly. We've used this functionality to collect information about players who were

part of a team in a specific game. CouchDB is a trustworthy and dependable NoSQL community member. CouchDB takes a truly decentralized approach to data storage, based on the assumption that networks are inherently unstable, and that hardware failure is inevitable.

CouchDB, which is small enough to fit in your pocket but powerful enough to run a business, supports a wide range of deployment options. These non-relational, or table-less, NoSQL databases differ from SQL databases in that they are not relational. They make management easier while keeping a high level of responsiveness to new data in this way.

NoSQL has gained popularity as a result of its increased elasticity and scalability over previous types of databases. It was designed to work in a variety of settings, including low-cost hardware.

Despite the fact that NoSQL has expanded at a great pace, due to the technology's newness, community support is limited. It doesn't have a common platform, like SQL, which makes it difficult to propagate. During migration, this was a source of concern. The database industry can only come together through standardization. Interface and interoperability are another difficulty that NoSQL faces, and they need to be addressed as soon as possible.

## Conclusion

Even though much has been analyzed and evaluated, this data exploration has resulted in a slew of new questions, the answers to which can be sought after further investigation. We can currently predict which team will win, but perhaps with new data and more information, we can try to determine what factors are facilitating the win and what advantage is assisting them. It could be the location, the performance of the players, or the overall strategy of the team. We can also use our analysis to determine what the best aspect of each team and player is that contributes to their success. We may also try to figure out how we can help a weaker team improve their strategy by identifying their weak points.

Because the NoSQL structure eliminates the need for tables and table joins, there is a lot of room

to improve our analysis. This allows us to have attributes and sub-attributes in a nested structure, allowing us to include any related features in the future for our analysis without having to touch our SQL database model.

## References

Halpin, T. (2015, March 27). *Object-Role Modeling Fundamentals: A Practical Guide to Data Modeling with ORM* (First). Technics Publications.

Murach, J. (2016, July 8). *Murach's SQL Server 2016 for Developers*(p410-447). Mike Murach & Associates.

Kareem, H. F. (2021, September 18). *La Liga 2020\_2021 dataset*. Kaggle. Retrieved October 23, 2022, from <https://www.kaggle.com/datasets/hamdallak/la-liga-2020-2021-dataset>