Please indicate below the tool you have analyzed for Homework 3, and past the code you have realized.
Your code must include all queries and indexes (if any), and possibly the script you have used to populate the database (or any description of the process you have used to populate the database). If you have interacted with the database system through an external programming language, e.g., Pyhton, insert your functions/program.
===
In this homework we have used the tool **Neo4j**.

```
// Create clubs
LOAD CSV WITH HEADERS FROM 'file:///clubs.csv' AS row
FIELDTERMINATOR ';'
CREATE (c:Club {Name: row.pretty_name,
                id: row.club_id,
                MarketValue: toFloat(row.total_market_value),
                League: row.domestic_competition_id});

// Create players
LOAD CSV WITH HEADERS FROM 'file:///players.csv' AS row
FIELDTERMINATOR ';'
CREATE (p:Player {Name: row.pretty_name,
                id: row.player_id,
                MarketValue: toFloat(row.market_value_in_gbp),
                Country: row.country_of_citizenship,
                Position: row.position, Club: row.current_club_id,
                LastSeason : row.last_season});
```

##
# SQL code used to create the file *Relationship_Player_Club.csv* used to implement the corresponding relationship.

SELECT id, clubs_id
FROM players;
##

```
// Relationship Players - Club
LOAD CSV WITH HEADERS FROM "file:///Relationship_Player_Club.csv" AS row
//look up the two nodes we want to connect up
MATCH (p:Player {id: row.id}), (c:Club {id: row.clubs_id})
//now create a relationship between them
CREATE (p)-[:PLAYER_OF]->(c);
```

```
// Create games
LOAD CSV WITH HEADERS FROM 'file:///games.csv' AS row
FIELDTERMINATOR ';'
CREATE (g:Game {id: row.game_id,
                HomeClub: row.home_club_id,
                AwayClub: row.away_club_id,
                Competition: row.competition_code,
                Round: row.round,
                Stadium: row.stadium,
                Attendance: toInteger(row.attendance),
                Season : row.season});
```

```
############################################################################
# SQL code used to create the file Relationship_Games_Club_Home.csv used to implement
the corresponding relationship.

SELECT id, home_club_id, home_club_goals
FROM games;
############################################################################
```

```
// Relationship Home Club - Games
LOAD CSV WITH HEADERS FROM "file:///Relationship_Games_Club_Home.csv" AS row
//look up the two nodes we want to connect up
MATCH (c:Club {id: row.home_club_id}), (g:Game {id: row.id})
//now create a relationship between them
CREATE (c)-[:PLAYED_AT_HOME {goal: toInteger(row.home_club_goals)}]->(g);
```

```
############################################################################
# SQL code used to create the file Relationship_Games_Club_Away.csv used to implement
the corresponding relationship.

SELECT id, away_club_id, away_club_goals
FROM games;
############################################################################
```

```
// Relationship Away Club - Games
LOAD CSV WITH HEADERS FROM "file:///Relationship_Games_Club_Away.csv" AS row
//look up the two nodes we want to connect up
MATCH (c:Club {id: row.away_club_id}), (g:Game {id: row.id})
//now create a relationship between them
CREATE (c)-[:PLAYED_AWAY {goal: toInteger(row.away_club_goals)}]->(g);
```

```
// Relationship Player - Games
LOAD CSV WITH HEADERS FROM "file:///appearances.csv" AS row
//look up the two nodes we want to connect up
MATCH (p:Player {id: row.players_id}), (g:Game {id: row.games_id})
//now create a relationship between them
CREATE (p)-[:PLAYED_IN {goals: toInteger(row.goals),
                        assists: toInteger(row.assists),
                        minutes_played: toInteger(row.minutes_played),
                        yellow_cards: toInteger(row.yellow_cards),
                        red_cards: toInteger(row.red_cards)}]->(g);


// Schema visualization
CALL db.schema.visualization()

// Add indexes and constraints
CREATE INDEX FOR (p:Player) ON (p.id);

CREATE INDEX FOR (c:Club) ON (c.id);

CREATE INDEX FOR (g:Game) ON (g.id);

CREATE CONSTRAINT PlayersMustHaveNames FOR (p:Player)
REQUIRE p.Name IS NOT NULL;

CREATE CONSTRAINT ClubsMustHaveNames FOR (c:Club)
REQUIRE c.Name IS NOT NULL;

CREATE CONSTRAINT MatchPlayedAtHome FOR ()-[r:PLAYED_AT_HOME]-()
REQUIRE r.goal IS NOT NULL;

CREATE CONSTRAINT MatchPlayedAway FOR ()-[r:PLAYED_AWAY]-()
REQUIRE r.goal IS NOT NULL;

CREATE CONSTRAINT constraint_played_in ON ()-[r:PLAYED_IN]-()
ASSERT EXISTS (r.minutes_played);

// Queries

// Query 1
// Player with the highest market value
MATCH (p:Player)
WITH MAX(p.MarketValue) AS MaxMarketValue
MATCH (p:Player)-[:PLAYER_OF]->(c:Club)
WHERE p.MarketValue = MaxMarketValue
RETURN p, c;
```

```
// Query 2
// Italian clubs whose players have average market value
// greater than 10 million
MATCH (p:Player {LastSeason: "2021"}) -[r:PLAYER_OF]-> (c:Club {League: "IT1"})
WITH AVG(p.MarketValue) AS Avg_MarketValue, c.Name AS Club
WHERE Avg_MarketValue > 1000000
RETURN Club, Avg_MarketValue
ORDER BY Avg_MarketValue DESC;

// Query 3
// Players that received more than 4 red cards
MATCH (p:Player)-[r:PLAYED_IN]->(g:Game)
WITH SUM(r.red_cards) AS SumRedCards, p.id as Id, p.Name as Name
WHERE SumRedCards > 4
RETURN Id, Name, SumRedCards;

// Query 4
// Italian players with the more appearances in Champions League
MATCH (p:Player {Country: "Italy"})-[r:PLAYED_IN]->(g:Game {Competition: "CL"})
RETURN p.Name, COUNT(r) AS CL_Appearances
ORDER BY CL_Appearances DESC;

// Query 4 - To visualize the graph
MATCH (p:Player {Country: "Italy"})-[r:PLAYED_IN]->(g:Game {Competition: "CL"})
RETURN p, g;

// Query 5
// Players who have scored in the Roma Lazio derbies since 2014
MATCH (p:Player)-[r:PLAYED_IN]->(g:Game)
WHERE (g.HomeClub = '12' AND g.AwayClub = '398')
        OR (g.HomeClub = '398' AND g.AwayClub = '12')
RETURN p.Name, SUM(toInteger(r.goals)) AS Derby_Goal
ORDER BY Derby_Goal DESC LIMIT 10;

// Query 6
// Match with the largest number of spectators
MATCH (g:Game)
WITH MAX(g.Attendance) AS MaxAttendance
MATCH (p:Player)-[r:PLAYED_IN]->(g:Game)
WHERE g.Attendance = MaxAttendance AND r.goals > 0
RETURN p, g;
```

```
// Query 7
// Match with the maximum number of goals
MATCH (c1:Club)-[r1:PLAYED_AT_HOME]->(g:Game)<-[r2:PLAYED_AWAY]-(c2:Club)
WITH MAX(r1.goal + r2.goal) AS MaxNumberGoal
MATCH (c1:Club)-[r1:PLAYED_AT_HOME]->(g:Game)<-[r2:PLAYED_AWAY]-(c2:Club)
WHERE (r1.goal + r2.goal) = MaxNumberGoal
RETURN c1, c2;

// Query 8
// Players who scored more goals in the Champions League finals
MATCH (p:Player)-[r:PLAYED_IN]->(g:Game {Competition: "CL", Round: "Final"})
WHERE r.goals > 0
RETURN p.Name, SUM(r.goals) AS CL_Final_Goals
ORDER BY CL_Final_Goals DESC;

// Query 9
// Player who scored more goals in a single match
MATCH (p:Player)-[r:PLAYED_IN]->(g:Game)
WITH MAX(r.goals) AS MaxGoal
MATCH (p:Player)-[r:PLAYED_IN]->(g:Game)
WHERE r.goals = MaxGoal
RETURN p, g;

// Query 10 - Exists
// Italian defenders who have played against Lionel Messi
MATCH (p:Player {Country: "Italy", Position : "Defender"})
WHERE EXISTS ( (p)-[:PLAYED_IN]->(:Game)<-[:PLAYED_IN]- (:Player {Name: 'Lionel
Messi'}) )
RETURN p.Name AS Player;

// Query 11 - Path
// Return all the Player nodes that can reach a Game node which represents
// a Champions League final in two steps
MATCH q = (p:Player {Position: "Goalkeeper", LastSeason: "2021"})-[*2]-(g:Game
{Competition: "CL", Round: "Final"})
RETURN nodes(q);
```