Question

The assignment to be solved using NEO4J.
Need to CSV LOAD FEATURE Need relationships build. Code needs to be basic so it's understandable
Need to build graph data base, database for a given dataset, which will be a CSV file that will be provided.

The dataset contains information about the English Premier League (EPL) matches. I am expected to design and create a graph database to visualise the dataset and to answer the following queries: 1) Display total number of matches played. 2) Display details of all matches involved "Arsenal FC" 3) Display the number of matches "Liverpool FC" has lost. 4) Display top five teams that have best scoring power. 5) Display top five teams that have poorest defending. 6) Display top five teams that have best winning records. 7) Display top five teams with best home winning records. 8) Display top five teams with worst home losing recording. 9) Which teams had most "draw"? 10) display the team with most consecutive "wins".

Solution

```
//creates matches graph
LOAD CSV WITH HEADERS FROM "file:///data.csv"  AS row CREATE ( :Match {status:
row.status, time:row.time,
id:row.id,home:row.home,score_0:`row.score/0`,score_1:`row.score_1`,
away:row.away,link:row.link});

BUILD NODES:
//creates team nodes(main graph)
Match (m:Match) MERGE (:Team {team: m.home});

// in case there is any team only in away column - but there won't be any
Match (m:Match) MERGE (:Team {team: m.away});

BUILD RELATIONSHIPS:
LOAD CSV with headers from "file:///data.csv" as row MATCH (home:TeamCopy {team:
row.home}) MATCH (away:TeamCopy {team: row.away}) MERGE (home)-
[pu:EPL_MATCH]->(away) ON CREATE SET pu.id=toInteger(row.id),pu.status =
row.status,pu.time=row.time,pu.link=row.link,pu.score_0=toInteger(trim(row.`score/0`)),pu.sc
ore_1=toInteger(trim(row.`score/1`));

Trim - is used to remove trailing spaces
toIntger - to convert scores string to int
EPL_MATCH is the relation between teams
```

Home → Away

Each relationship has properties like :

1. Id
2. Time
3. Score/0 - score by home team
4. score/1 - score by away team
5. Link - match link
6. Status - match status

Query 1:

No of matches?

```
MATCH p=()-[r:EPL_MATCH]->() RETURN count(*);

// counting the number of relationships
```

Query 2:

```
// match details of matches involving arsenal fc
MATCH (:Team { team: 'Arsenal FC' })<-[r]->(team)
RETURN r,team
R-match details, team - opposite team
```

Query 3:

Liverpool lost matches

```
MATCH (:Team { team: 'Liverpool FC' })-[r]->(team) where r.score_0 <
r.score_1 with collect({relation:r}) as rows match (team)-[q]->(:Team
{team:'Liverpool FC'}) where q.score_0>q.score_1 with
rows+collect({relation:q}) as allrows unwind allrows as row return
count(row)
```

We are uniting two queries here, loses when it’s home + loses when it’s away.

Query 4:

```
MATCH (n:Team) with collect(n.team) as names unwind names as q1 MATCH
(:Team { team: q1 })-[r]->(team) with
q1,collect({score:r.score_0,team:q1}) as rows match (team)-[q]->(:Team
{team:q1}) with rows+collect({score:q.score_1,team:q1}) as allrows
unwind allrows as row return sum(row["score"]),row["team"] order by
sum(row["score"]) desc limit 5
```

Query 5:

```
MATCH (n:Team) with collect(n.team) as names unwind names as q1 MATCH (:Team { team: q1 })-[r]->(team) with q1,collect({score:r.score_1,team:q1}) as rows match (team)-[q]->(:Team {team:q1}) with rows+collect({score:q.score_0,team:q1}) as allrows unwind allrows as row return sum(row["score"]),row["team"] order by sum(row["score"]) desc limit 5
```

Query 6:

```
MATCH (n:Team) with collect(n.team) as names unwind names as q1 MATCH (:Team { team: q1 })-[r]->(team) where r.score_0>r.score_1 with q1,collect({score:r.score_0-r.score_1,team:q1}) as rows match (team)-[q]->(:Team {team:q1}) where q.score_1>q.score_0 with rows+collect({score:q.score_1-q.score_0,team:q1}) as allrows unwind allrows as row return sum(row["score"]),row["team"] order by sum(row["score"]) desc limit 5
```

Output:

13	"Manchester City FC"
9	"Liverpool FC"
9	"Arsenal FC"
8	"Manchester United FC"
7	"Everton FC"

Query 7:

```
MATCH (n:Team) with collect(n.team) as names unwind names as q1 MATCH (:Team { team: q1 })-[r]->(team) where r.score_0>r.score_1 with q1,collect({score:r.score_0-r.score_1,team:q1}) as rows unwind rows as row return sum(row["score"]),row["team"] order by sum(row["score"]) desc limit 5
```

Output:

7	"Liverpool FC"
7	"Manchester City FC"
5	"Manchester United FC"
4	"Chelsea FC"
4	"Arsenal FC"

Query 8:

MATCH (n:Team) with collect(n.team) as names unwind names as q1 MATCH (:Team { team: q1 })-[r]->(team) where r.score_1>r.score_0 with q1,collect({score:r.score_1-r.score_0,team:q1}) as rows unwind rows as row return sum(row["score"]),row["team"] order by sum(row["score"]) desc limit 5

Output:

7	"Stoke City FC"
5	"Sunderland AFC"
5	"West Ham United FC"
4	"Hull City AFC"
4	"Swansea City AFC"

Query 9:

MATCH (n:Team) with collect(n.team) as names unwind names as q1 MATCH (:Team { team: q1 })-[r]->(team) where r.score_0=r.score_1 with q1,collect({score:r.score_0,team:q1}) as rows match (team)-[q]->(:Team {team:q1}) where q.score_1=q.score_0 with rows+collect({score:q.score_1,team:q1}) as allrows unwind allrows as row return count(row["score"]),row["team"] order by count(row["score"]) desc limit 5

Output:

2	"Middlesbrough FC"
2	"Crystal Palace FC"
2	"West Bromwich Albion FC"
2	"Tottenham Hotspur FC"
2	"Stoke City FC"

Query 10:

Final :

```
MATCH (n:Team) with collect(n.team) as names unwind names as q1 MATCH (:Team { team: q1 })-[r]->(team) with q1, collect(r.score_0-r.score_1) as rows match (team)-[q]->(:Team {team:q1}) with q1, rows+collect(q.score_1-q.score_0) as allrows with q1, reduce(count1=[0,0], m in allrows | case when m>0 then [count1[0]+1,count1[1]] else [0,count1[0]] end) as reduction with q1, reduce(count2=0, m in reduction | case when m>count2 then m else count2 end) as red return red,q1;
```