**Part 1**: Use the following schema to answer the questions 1-3 below. This schema refers to a gaming organization. There can be any number of players per game. Each game and each tournament has one winner; if the game or tournament is ongoing, the winner is null. The gameWinner and tournamentWinner attributes are foreign keys for memberID.

Member (memberID, memberName, joinDate)

Game (gameID, gameType, gameDate, gameWinner, tournamentName)

MemberGame (memberID, gameID)

Tournament (tournamentName, tournamentWinner)

1. Write SQL queries to generate the following tables:
   1. The total number of ongoing tournaments.
      1. Select count(tournamentWinner) from Tournament where tournamentWinner=null;
   2. The average number of players per game of type Power Grid.
      1. Select count(memberID)/count(gameType) from MemberGame natural join Game where gameType= “PowerGrid”;
   3. The names (no duplicates) of all players Abigail Baker beat in any game.
      1. Select distinct memberName from Member natural join MemberGame natural join Game where memberName!= “Abigail Baker” and gameWinner =(Select memberID from Member where memberName= “Abigail Baker”);
   4. Each tournament name with the number of games in that tournament won by the tournament winner.
      1. Select tournamentName, count(gameID) as numWinGame from Tournament Natural join Game natural join memberGame where tournamentWinner=gameWinner group by tournamentName;
   5. A list of game IDs and game types sorted by game type then game id.
      1. Select gameID, gameType from Game order by gameType ASC, gameID ASC;
   6. A list of player IDs of all players who have played a game with the date of the first game played by that player.
      1. Select memberID, min(gameDate) from MemberGame natural join Game group by memberID;
2. Write a description of what each of the following queries does:
   1. SELECT \*

FROM Member LEFT OUTER JOIN Tournament ON memberID = tournamentWinner;

Dispaly all information from member table and tournament table where member’s memberID are same as tournament table’s tournamentWinner.

* 1. UPDATE Member

SET memberName = “Devon Chance”

WHERE memberID = 100154;

Modify memberID to 100154 where memberName is Devon Chance from Member table.

* 1. SELECT memberID, gameID

FROM Member NATURAL JOIN MemberGame NATURAL JOIN Game

WHERE joinDate < “2021-01-01” AND gameDate >= “2021-01-01”

Display memberID and gameID who joinDate is before 2021-01-01 and which game date is 2021-01-01 or after.

1. Write two functionally different relational algebra statements equivalent to the query in 2c.
   1. Pi memberID,gameID (sigma joinDate< “2021-01-01” ^ gameDate>= “2021-01-01” (Member NAJ MemberGame NAJ Game))
   2. Pi memberID,gameID (Membergame NAJ(sigma joinDate < “2021-01-01” (Member)) NAJ (sigma gameDate>= “2021-01-01”(Game)))

**Part 2**: Use the following relations to answer question 4.

R0 R1

|  |  |  |
| --- | --- | --- |
| a | b | c |
| 1 | 2 | 2 |
| 2 | 1 | 2 |
| 3 | 1 | 2 |

|  |  |  |
| --- | --- | --- |
| b | c | d |
| 1 | 2 | 3 |
| 2 | 4 | 7 |

1. For each of the following relational algebra expressions, draw the resultant relation in table form and write an equivalent expression in tuple relational calculus.

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | D |
| 1 | 2 | 2 | Null |
| 2 | 1 | 2 | 3 |
| 3 | 1 | 2 | 3 |
| Null | 2 | 4 | 7 |

{f,s|R0(f)^R1(s)^f.b=s.b^f.c=s.c}

|  |  |
| --- | --- |
| B | C |
| 2 | 2 |
| 1 | 2 |
| 1 | 2 |

{f.b,f.c|R0(f)^R1(f)}

**Bonus**: Which of your answers to question 3 will most likely be processed faster? Justify your answer.

Answer b will be processed faster because it is like ab+ac to become a(b+c). The calculation method will reduce the calculation steps.