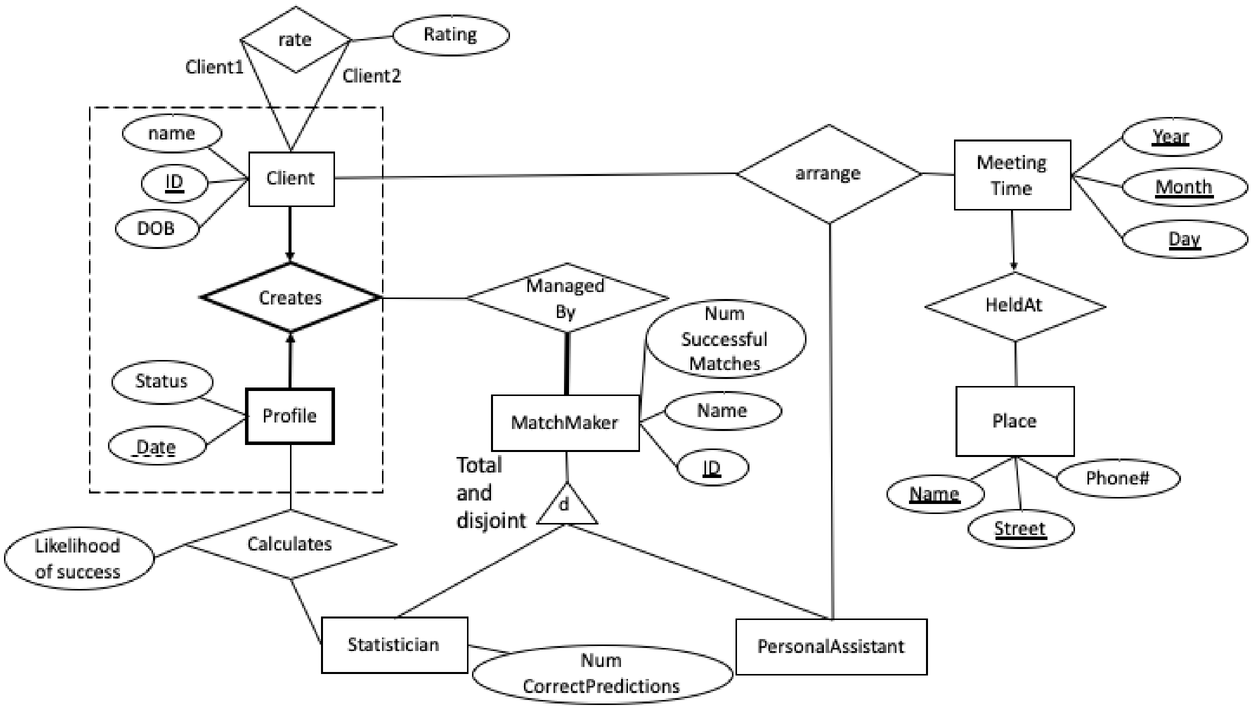
Evaluate the ER Diagram

While grading assignments, the CPSC 368 TAs decided to consider alternate employment options and worked together to create a matchmaking app where users can sign up for accounts and the TAs will play matchmaker. The first draft of their ER diagram is below.

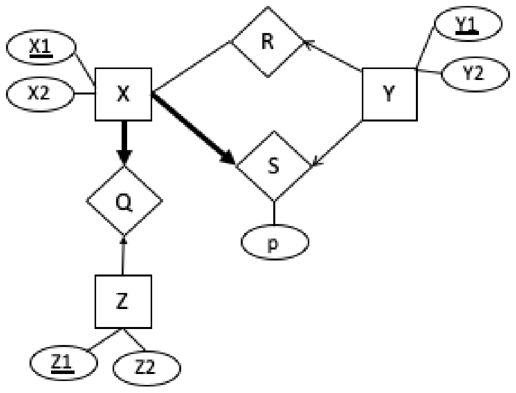


Answer the following questions based on the ER diagram above. Read the following statements and decide if each statement is true or false. Answers that are unclear or unreadable will be considered as incorrect.

Statement	True or False
Multiple MatchMakers can manage a client's profile.	True
A client can rate another client multiple times and the history of all ratings can be stored.	False
A meeting can be arranged without a personal assistant.	False
Different clients can create profiles on the same date.	True
The likelihood of success will be used for arranging the meeting.	False

Translate an ER Diagram into a Schema

Write relational schemas for the E-R diagram below. State any assumptions that you make – but your assumptions cannot contradict the facts given. Clearly identify all primary keys by underlining and foreign keys by circling. If any attribute is required to be not null or unique, be sure to clearly state that too. **No SQL DDL is required.**

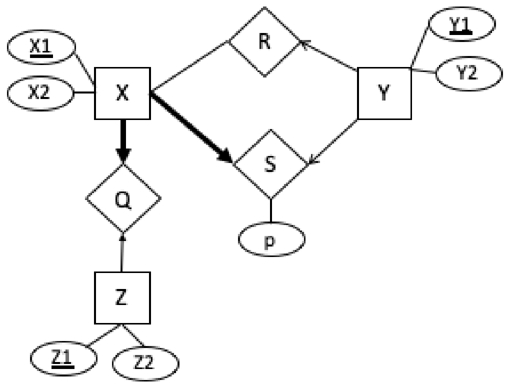


Primary keys are underlined and foreign keys are bolded.

- Z(<u>Z1</u>, Z2)
- XQS(<u>X11</u>, X2, **Z1**, **Y1**, p), Z1 and Y1 cannot be null
- YR(<u>Y1</u>, Y2, **X1**)

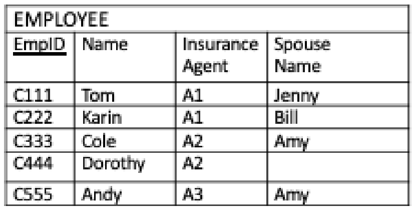
Writing SQL DDL

Translate the following ER diagram into relations. Write the SQL DDL statement for the relation that contains the relationship S from the ER diagram. Be sure to capture its required participation constraints (if any). Choose any data type you want for the attributes.



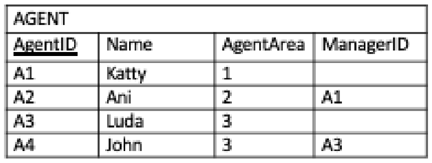
```
CREATE TABLE XQS (
  X1 INTEGER PRIMARY KEY,
  X2 INTEGER,
  Z1 INTEGER NOT NULL,
  Y1 INTEGER NOT NULL,
  p INTEGER,
  FOREIGN KEY(Z1) REFERENCES Z
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY(Y1) REFERENCES Y
    ON DELETE CASCADE
    ON UPDATE CASCADE
)
```

Referential Integrity



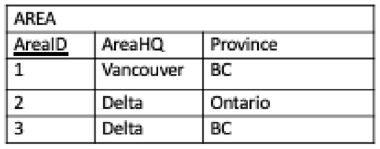
EMPLOYEE

EmplD	Name	Insurance Agent	Spouse Name
C111	Tom	A1	Jenny
C222	Karin	A1	Bill
C333	Cole	A2	Amy
C444	Dorothy	A2	
C555	Andy	A3	Amy



AGENT

AgentID	Name	AgentArea	ManagerID
A1	Katty	1	
A2	Ani	2	A1
A3	Luda	3	
A4	John	3	A3



AREA

AreaID	AreaHQ	Province
1	Vancouver	BC
2	Delta	Ontario
3	Delta	BC

In the relational instance above:

- Agent.ManagerID is a foreign key that references Agent.AgentID and has a DELETE NO ACTION specification.
- Employee.InsuranceAgent references Agent.AgentID and has a DELETE CASCADE specification.

Question: Is it true that you can delete the second record in the Agent table?

A. True
B. False
C. It depends

True. The DELETE NO ACTION option on the self-referencing integrity constraints means you cannot delete an Agent tuple that is being referred to by another Agent tuple (e.g., you can't delete the A1 tuple). The DELETE CASCADE option between Agent and Employee means if you delete a tuple in Agent, any tuple in Employee that refers to Agent will be deleted.

SQL Schema

Consider the following schema for a hotel:

1. Hotel(branchID, city, country, maxCapacity)

2. Rooms(bID, floorNumber, roomNumber, type, capacity, smokingAllowed, area)
    - bID refers to branchID in Hotel
    - capacity represents the maximum capacity a room can hold

3. Amenities(amenityID, description)

4. HotelAmenities(bID, amenityID)
    - amenityID references Amenities
    - bID references Hotel

5. Customers(passportNumber, firstName, lastName, nationality, dateOfBirth)

6. HotelClubMember(pNum)
    - pNum references passportNumber in Customers

7. Reservation(id, bID, pNum, startDate, endDate, numGuests, smokingRequested)
    - bID refers to branchID in Hotel
    - pNum refers to passportNumber in Customers

8. HotelStay(rID, bID, floorNum, roomNum)
    - rID refers to id in Reservation
    - bID, floorNum and roomNum refer to bID, floorNumber and roomNumber in Rooms

9. Employees(id, firstName, lastName, occupation, startDate, endDate)

10. EmployeeCheckInLocations(eID, beaconID, date, time)
    - This relation holds information about when and where an employee has checked in
    - eID refers to id in Employees
    - beaconID refers to id in CheckInBeacons

11. CheckInBeacons(id, bID, floorNumber, direction)
    - This relation represents information about electronic check in points where employees have to tap their badge against to prove they were at a particular location at a particular time
    - bID refers to branchID in Hotel

Hotel Club Members in Vancouver

For each hotel in Vancouver, find the total number of hotel club members who have stayed there.

```
SELECT h.branchID, COUNT(r.pNum)
FROM Hotel h, Reservation r, HotelClubMember hcm
WHERE h.branchID = r.id AND r.pNum = hcm.pNum AND h.city = “Vancouver”
GROUP BY h.branchID
```

Popular Types of Rooms

Find the nationality that most frequently stays in a smoking room.

```
CREATE VIEW num_reservations_by_nationality AS
  SELECT c.nationality AS nationality, COUNT(r.pNum) AS count
  FROM Reservation r, Customers c
  WHERE r.pNum = c.passportNumber AND r.smokingRequested = "yes"
  GROUP BY c.nationality

SELECT nationality
FROM num_reservations_by_nationality
WHERE count = (SELECT Max(count) FROM num_reservations_by_nationality)
```

Fix the Error

Due to a system glitch, the number of guests per reservation was incorrectly stored. Every reservation that has more than three guests recorded one additional guest than they have. For example, a reservation that says there were four guests actually should have three guests. A reservation that says there are three guests should stay at three guests.

Write an UPDATE statement that would update the reservation so that every booking has the correct number of guests.

```
UPDATE Reservation
SET numGuests = numGuests -1
WHERE numGuests > 3
```

Booked Out Hotel

Return hotels where all rooms have been reserved.

```
SELECT *
FROM Hotel h
WHERE NOT EXISTS( (SELECT r.bID, r.floorNumber, r.roomNumber
          FROM Rooms r)

        EXCEPT

          (SELECT r2.bID, r2.floorNumber, r2.roomNumber
          FROM Reservation r1, rooms r2
          WHERE h.branchID = r1.bID and r2.bID = r1.bID)
        )
```

All Amenities

Return Canadian hotels that have all amenities. Do not use the EXPECT operator.

```
SELECT h.branchID
FROM Hotel h
WHERE h.country = "Canada" AND NOT EXISTS (SELECT *
                    FROM Amenities a
                    WHERE NOT EXISTS (SELECT ha.bID
                              FROM HotelAmenities ha
                              WHERE h.branchID = ha.bID AND a.amenityID =
                                    ha.amenityID))
```