Solutions to Problem Set 5

Problem 1.

CREATE TABLE Teams (  
    name CHAR(30) PRIMARY KEY,  
    country CHAR(30),  
    fanLevel CHAR(5) CHECK (fanLevel in ('wild', 'timid')),  
    CHECK (country <> 'Brazil' OR fanLevel = 'wild')  
);

CREATE TABLE Players (  
    name CHAR(30) PRIMARY KEY,  
    country CHAR(30),  
    teamName CHAR(30) REFERENCES Teams(name),  
    salary NUMBER,  
    CHECK (country <> 'Bulgaria' OR name LIKE '%ov'),  
    CHECK (country = (SELECT T.country  
        FROM Teams T  
        WHERE T.name = teamName)  
    OR  
    salary/1.2 > (SELECT max(P.salary)  
        FROM Players P, Teams T  
        WHERE P.teamName = teamName  
        AND T.name = teamName  
        AND T.country = P.country))  
);

CREATE TABLE Games(  
    teamName1 CHAR(30) REFERENCES Teams(name),  
    numGoals1 NUMBER,  
    teamName2 CHAR(30) REFERENCES Teams(name),  
    numGoals2 NUMBER,  
    gameDate DATE,  
    PRIMARY KEY(teamName1, teamName2, date),  
    CHECK (teamName1 <> teamName2)  
);

CREATE TABLE Goals(  
    playerName CHAR(30) REFERENCES Players(name),  
    numGoals NUMBER CHECK (numGoals > 0),  
    goalDate DATE,  
    PRIMARY KEY(playerName, goalDate)  
);
CREATE ASSERTION GoalsCheck CHECK
  (NOT EXISTS
   ((SELECT *
     FROM Games G
     WHERE G.numGoals1 <>
       (SELECT sum(L.numGoals)
        FROM Players P, Goals L
        WHERE P.teamName = G.teamName1
        AND P.name = L.playerName
        AND G.gameDate = L.goalDate))
    UNION
   ((SELECT *
     FROM Games G
     WHERE G.numGoals2 <>
       (SELECT sum(L.numGoals)
        FROM Players P, Goals L
        WHERE P.teamName = G.teamName2
        AND P.name = L.playerName
        AND G.gameDate = L.goalDate))));

CREATE ASSERTION NoPhantomGoals CHECK
  (NOT EXISTS
   ((SELECT *
     FROM Goals L
     WHERE L.goalDate <> ALL
       (SELECT G.gameDate
        FROM Players P, Games G
        WHERE P.name = L.playerName
        AND G.teamName1 = P.teamName))
    UNION
   ((SELECT *
     FROM Goals L
     WHERE L.goalDate <> ALL
       (SELECT G.gameDate
        FROM Players P, Games G
        WHERE P.name = L.playerName
        AND G.teamName2 = P.teamName))));

Problem 2.
CREATE TABLE R (  
a int PRIMARY KEY,  
b int
);
CREATE TABLE S (  
c int REFERENCES R(a),  
d int
);
Initial state: R = {(1,2)}, S = {(1,3)}

Sequence of SQL statements:

1. DELETE FROM R;
2. DELETE FROM S;
3. DELETE FROM R;

Assuming default policy (reject) ON DELETE:

<table>
<thead>
<tr>
<th>After statement</th>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{(1,2)}</td>
<td>{(1,3)}</td>
</tr>
<tr>
<td>2</td>
<td>{(1,2)}</td>
<td>{}</td>
</tr>
<tr>
<td>3</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>

Assuming ON DELETE SET NULL:

<table>
<thead>
<tr>
<th>After statement</th>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{}</td>
<td>{(NULL,3)}</td>
</tr>
<tr>
<td>2</td>
<td>{}</td>
<td>{}</td>
</tr>
<tr>
<td>3</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>

Assuming ON DELETE CASCADE:

<table>
<thead>
<tr>
<th>After statement</th>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{}</td>
<td>{}</td>
</tr>
<tr>
<td>2</td>
<td>{}</td>
<td>{}</td>
</tr>
<tr>
<td>3</td>
<td>{}</td>
<td>{}</td>
</tr>
</tbody>
</table>
Problem 3.

CREATE TRIGGER DoubleFreqMiles
AFTER INSERT ON Flights
REFERENCING
        NEW ROW AS NewTuple
FOR EACH ROW
WHEN (100000 <= ( SELECT SUM(miles)
                  FROM Flights
                  WHERE Flights.airline = NewTuple.airline
                     AND Flights.passenger = NewTuple.passenger))

UPDATE Flights
SET miles = 2*NewTuple.miles
WHERE passenger = NewTuple.passenger
AND airline = NewTuple.airline
AND flightNum = NewTuple.flightNum;

CREATE TRIGGER NoRadicalMileChange
AFTER UPDATE ON Flights
REFERENCING
        NEW ROW AS NewTuple,
        OLD ROW AS OldTuple
FOR EACH ROW
WHEN (NewTuple.miles < 500 OR NewTuple.miles <10*OldTuple.miles)

UPDATE Flights
SET miles = 500
WHERE passenger = NewTuple.passenger
AND airline = NewTuple.airline
AND flightNum = NewTuple.flightNum;