Omar Ahmadi
5.2, 5.3, 5.4, 5.9, 19.4, 19.10

Exercise 5.2

Suppliers(*sid:* integer, *sname:* string, *address:* string)
Parts(*pid:* integer, *pname:* string, *color:* string)
Catalog(*sid:* integer, *pid6:* integer, *cost:* real)

1. Find the *pname*s of parts for which there is some supplier.

```
SELECT DISTINCT P.pname
FROM Parts P, Catalog C
WHERE P.pid = C.pid
```

2. Find the *sname*s of suppliers who supply every part.

```
SELECT S.sname
FROM Suppliers S
WHERE NOT EXISTS (( SELECT P.pid
                    FROM Parts P )
                    EXCEPT
                    ( SELECT C.pid
                    FROM Catalog C
                    WHERE C.sid = S.sid ))
```

3. Find the *sname*s of suppliers who supply every red part.

```
SELECT S.sname
FROM Suppliers S
WHERE NOT EXISTS (( SELECT P.pid
                    FROM Parts P
                    WHERE P.color = 'Red' )
                    EXCEPT
                    ( SELECT C.pid
                    FROM Catalog C, Parts P
                    WHERE C.sid = S.sid AND
                    C.pid = P.pid AND P.color = 'Red' ))
```

4. Find the *pname*s of parts supplied by Acme Widget Suppliers and no one else.

```
SELECT P.pname
FROM Parts P, Catalog C, Suppliers S
WHERE P.pid = C.pid AND C.sid = S.sid
AND S.sname = 'Acme Widget Suppliers'
AND NOT EXISTS ( SELECT *
                 FROM Catalog C1, Suppliers S1
                 WHERE P.pid = C1.pid AND C1.sid = S1.sid AND
                 S1.sname <> 'Acme Widget Suppliers' )
```

5. Find the *sid*s of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).

```
SELECT DISTINCT C.sid
FROM Catalog C
WHERE C.cost > ( SELECT AVG (C1.cost)
                 FROM Catalog C1
                 WHERE C1.pid = C.pid )
```

6. For each part, find the *sname* of the supplier who charges the most for that part.

```
SELECT P.pid, S.sname
FROM Parts P, Suppliers S, Catalog C
WHERE C.pid = P.pid
AND C.sid = S.sid
AND C.cost >= ALL (SELECT C1.cost)
                  FROM Catalog C1
                  WHERE C1.pid = P.pid)
```

7. Find the *sid*s of suppliers who supply only red parts.

```
SELECT DISTINCT C.sid
FROM Catalog C
WHERE NOT EXISTS ( SELECT *
                   FROM Parts P
                   WHERE P.pid = C.pid AND P.color <> 'Red' )
```

8. Find the *sid*s of suppliers who supply a red part and a green part.

```
SELECT DISTINCT C.sid
FROM Catalog C, Parts P
WHERE C.pid = P.pid AND P.color = 'Red'
INTERSECT
SELECT DISTINCT C1.sid
FROM Catalog C1, Parts P1
WHERE C1.pid = P1.pid AND P1.color = 'Green'
```

9. Find the *sid*s of suppliers who supply a red part or a green part.

```
SELECT DISTINCT C.sid
FROM Catalog C, Parts P
WHERE C.pid = P.pid AND P.color = 'Red'
UNION
SELECT DISTINCT C1.sid
FROM Catalog C1, Parts P1
WHERE C1.pid = P1.pid AND P1.color = 'Green'
```

10. For every supplier that only supplies green parts, print the name of the supplier and the total number of parts that she supplies.

```
SELECT S.sname, COUNT(*) as PartCount
FROM Suppliers S, Parts P, Catalog C
WHERE P.pid = C.pid AND C.sid = S.sid
GROUP BY S.sname, S.sid
HAVING EVERY (P.color='Green')
```

11. For every supplier that supplies a green part and a red part, print the name and price of the most expensive part that she supplies.

```
SELECT S.sname, MAX(C.cost) as MaxCost
FROM Suppliers S, Parts P, Catalog C
WHERE P.pid = C.pid AND C.sid = S.sid
GROUP BY S.sname, S.sid
HAVING ANY ( P.color='green' ) AND ANY ( P.color = 'red' )
```

Exercise 5.3

Flights(*flno:* integer, *from:* string, *to:* string, *distance:* integer,
departs: time, *arrives:* time, *price:* real)
Aircraft(*aid:* integer, *aname:* string, *cruisingrange:* integer)
Certified(*eid:* integer, *aid:* integer)
Employees(*eid:* integer, *ename:* string, *salary:* integer)

1. Find the names of aircraft such that all pilots certified to operate them have salaries more than $80,000.

```
SELECT DISTINCT A.aname
FROM Aircraft A
WHERE A.Aid IN (SELECT C.aid
                FROM Certified C, Employees E
                WHERE C.eid = E.eid AND
                NOT EXISTS ( SELECT *
                             FROM Employees E1
                             WHERE E1.eid = E.eid AND E1.salary < 80000 ))
```

2. For each pilot who is certified for more than three aircraft, find the *eid* and the maximum *cruisingrange* of the aircraft for which she or he is certified.

```
SELECT C.eid, MAX (A.cruisingrange)
FROM Certified C, Aircraft A
WHERE C.aid = A.aid
GROUP BY C.eid
HAVING COUNT (*) > 3
```

3. Find the names of pilots whose *salary* is less than the price of the cheapest route from Los Angeles to Honolulu.

```
SELECT DISTINCT E.ename
FROM Employees E
WHERE E.salary < ( SELECT MIN (F.price)
FROM Flights F
WHERE F.from = 'Los Angeles' AND F.to = 'Honolulu')
```

4. For all aircraft with *cruisingrange* over 1000 miles, find the name of the aircraft and the average salary of all pilots certified for this aircraft.

```
SELECT Temp.name, Temp.AvgSalary
FROM ( SELECT A.aid, A.aname AS name,
        AVG (E.salary) AS AvgSalary
        FROM Aircraft A, Certified C, Employees E
        WHERE A.aid = C.aid AND C.eid = E.eid AND A.cruisingrange > 1000
        GROUP BY A.aid, A.aname ) AS Temp
```

5. Find the names of pilots certified for some Boeing aircraft.

```
SELECT DISTINCT E.ename
FROM Employees E, Certified C, Aircraft A
WHERE E.eid = C.eid AND C.aid = A.aid AND A.aname LIKE 'Boeing%'
```

6. Find the *aid*s of all aircraft that can be used on routes from Los Angeles to Chicago.

```
SELECT A.aid
FROM Aircraft A
WHERE A.cruisingrange > ( SELECT MIN (F.distance)
                         FROM Flights F
                         WHERE F.from = 'Los Angeles' AND F.to = 'Chicago' )
```

7. Identify the routes that can be piloted by every pilot who makes more than $100,000.

```
SELECT DISTINCT F.from, F.to
FROM Flights F
WHERE NOT EXISTS ( SELECT *
                   FROM Employees E
                   WHERE E.salary > 100000 AND
                   NOT EXISTS (SELECT *
                               FROM Aircraft A, Certified C
                               WHERE A.cruisingrange > F.distance AND E.eid = C.eid
                               AND A.aid = C.aid) )
```

8. Print the *ename*s of pilots who can operate planes with *cruisingrange* greater than 3000 miles but are not certified on any Boeing aircraft.

```
SELECT DISTINCT E.ename
FROM Employees E
WHERE E.eid IN ( ( SELECT C.eid
                   FROM Certified C
                   WHERE EXISTS ( SELECT A.aid
                                  FROM Aircraft A
                                  WHERE A.aid = C.aid AND A.cruisingrange > 3000 )
                   AND NOT EXISTS ( SELECT A1.aid
                                    FROM Aircraft A1
                                    WHERE A1.aid = C.aid
                                    AND A1.aname LIKE 'Boeing%'))
```

9. A customer wants to travel from Madison to New York with no more than two changes of flight. List the choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m.

```
SELECT F.departs
FROM Flights F
WHERE F.flno IN ( ( SELECT F0.flno
                    FROM Flights F0
                    WHERE F0.from = 'Madison' AND F0.to = 'New York'
                          AND F0.arrives < '18:00' )
                    UNION
```

```
( SELECT F0.flno
FROM Flights F0, Flights F1
WHERE F0.from = 'Madison' AND F0.to <> 'New York'
        AND F0.to = F1.from AND F1.to = 'New York'
        AND F1.departs > F0.arrives AND F1.arrives < '18:00' )
UNION
( SELECT F0.flno
FROM Flights F0, Flights F1, Flights F2
WHERE F0.from = 'Madison'
        AND F0.to = F1.from AND F1.to = F2.from
        AND F2.to = 'New York'AND F0.to <> 'New York'
        AND F1.to <> 'New York'AND F1.departs > F0.arrives
        AND F2.departs > F1.arrives AND F2.arrives < '18:00' ))
```

10. Compute the difference between the average salary of a pilot and the average salary of all employees (including pilots).

```
SELECT Temp1.avg - Temp2.avg
FROM (SELECT AVG (E.salary) AS avg
        FROM Employees E
        WHERE E.eid IN (SELECT DISTINCT C.eid
                        FROM Certified C )) AS Temp1,
      (SELECT AVG (E1.salary) AS avg
        FROM Employees E1 ) AS Temp2
```

11. Print the name and salary of every nonpilot whose salary is more than the average salary for pilots.

```
SELECT E.ename, E.salary
FROM Employees E
WHERE E.eid NOT IN ( SELECT DISTINCT C.eid
                     FROM Certified C )
AND E.salary > ( SELECT AVG (E1.salary)
                 FROM Employees E1
                 WHERE E1.eid IN
                     ( SELECT DISTINCT C1.eid
                     FROM Certified C1 ) )
```

12. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles.

```
SELECT E.ename
FROM Employees E, Certified C, Aircraft A
WHERE C.aid = A.aid AND E.eid = C.eid
GROUP BY E.eid, E.ename
HAVING EVERY (A.cruisingrange > 1000)
```

13. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles, but on at least two such aircrafts.

```
SELECT E.ename
FROM Employees E, Certified C, Aircraft A
WHERE C.aid = A.aid AND E.eid = C.eid
GROUP BY E.eid, E.ename
HAVING EVERY (A.cruisingrange > 1000) AND COUNT (*) > 1
```

14. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles and who are certified on some Boeing aircraft.

```
SELECT E.ename
FROM Employees E, Certified C, Aircraft A
WHERE C.aid = A.aid AND E.eid = C.eid
GROUP BY E.eid, E.ename
HAVING EVERY (A.cruisingrange > 1000) AND ANY (A.aname = 'Boeing')
```

Exercise 5.4

Emp(*eid:* integer, *ename:* string, *age:* integer, *salary:* real)
Works(*eid:* integer, *did:* integer, *pct time:* integer)
Dept(*did:* integer, *dname:* string, *budget:* real, *managerid:* integer)

1. Print the names and ages of each employee who works in both the Hardware department and the Software department.

```
SELECT E.ename, E.age
FROM Emp E, Works W1, Works W2, Dept D1, Dept D2
WHERE E.eid = W1.eid AND W1.did = D1.did AND D1.dname = 'Hardware'
AND E.eid = W2.eid AND W2.did = D2.did AND D2.dname = 'Software'
```

2. For each department with more than 20 full-time-equivalent employees (i.e., where the part-time and full-time employees add up to at least that many full-time employees), print the *did* together with the number of employees that work in that department.

```
SELECT W.did, COUNT (W.eid)
FROM Works W
GROUP BY W.did
HAVING 2000 < ( SELECT SUM (W1.pct time)
                FROM Works W1
                WHERE W1.did = W.did )
```

3. Print the name of each employee whose salary exceeds the budget of all of the departments that he or she works in.

```
SELECT E.ename
FROM Emp E
WHERE E.salary > ALL (SELECT D.budget
                      FROM Dept D, Works W
                      WHERE E.eid = W.eid AND D.did = W.did)
```

4. Find the *managerid*s of managers who manage only departments with budgets greater than $1 million.

```
SELECT DISTINCT D.managerid
FROM Dept D
WHERE 1000000 < ALL (SELECT D2.budget
                     FROM Dept D2
                     WHERE D2.managerid = D.managerid )
```

5. Find the *ename*s of managers who manage the departments with the largest budgets.

```
SELECT E.ename
FROM Emp E
WHERE E.eid IN (SELECT D.managerid
                FROM Dept D
                WHERE D.budget >= ALL (SELECT D2.budget
                                       FROM Dept D2))
```

6. If a manager manages more than one department, he or she *controls* the sum of all the budgets for those departments. Find the *managerid*s of managers who control more than $5 million.

```
SELECT D.managerid
FROM Dept D
WHERE 5000000 < (SELECT SUM (D2.budget)
                 FROM Dept D2
                 WHERE D2.managerid = D.managerid )
```

7. Find the *managerid*s of managers who control the largest amounts.

```
SELECT DISTINCT tempD.managerid
FROM (SELECT DISTINCT D.managerid, SUM (D.budget) AS tempBudget
      FROM Dept D
      GROUP BY D.managerid ) AS tempD
WHERE tempD.tempBudget = (SELECT MAX (tempD.tempBudget)
                          FROM tempD)
```

8. Find the *ename*s of managers who manage only departments with budgets larger than $1 million, but at least one department with budget less than $5 million.

```
SELECT M.ename
FROM Emp M
WHERE M.sid IN (SELECT M2.sid
                FROM Emp M2, Dept D
                WHERE M2.sid = D.managerid AND D.budget < 5000000
                EXCEPT
                SELECT M3.sid
                FROM Emp M3, Dept D2
                WHERE M3.sid = D2.managerid AND
                D2.budget <= 5000000)
```

Exercise 5.9

Discuss the strengths and weaknesses of the trigger mechanism. Contrast triggers with other integrity constraints supported by SQL.

A trigger stored procedure that automatically executes when an event occurs in the database server. Advantages of triggers are the ability to create new tables, new views in addition to update, delete, or insertion queries, all based on the result of a query condition. Triggers can also occur before or after a change in the database. The disadvantage with triggers is that they provide added complexity. Integrity constraints can do the same things as triggers without the added complexity.

Exercise 19.4

1. Assume that no record has NULL values. Write an SQL query that checks whether the functional dependency $A \rightarrow B$ holds.

```
SELECT COUNT(*)
FROM R AS R1,R AS R2
WHERE (R1.B ! = R2.B) AND (R1.A = R2.A)
```

2. Assume again that no record has NULL values. Write an SQL assertion that enforces the functional dependency $A \rightarrow B$.

```
CREATE ASSERTION ADeterminesB
CHECK ((SELECT COUNT(*)
        FROM R AS R1,R AS R2
        WHERE (R1.B ! = R2.B) AND (R1.A = R2.A))
        =0)
```

3. Let us now assume that records could have NULL values. Repeat the previous two questions under this assumption.

```
SELECT COUNT(*)
FROM R AS R1,R AS R2
WHERE ((R1.B != R2.B) AND (R1.A = R2.A))
OR ((R1.B is NULL) AND (R2.B is NOT NULL) AND (R1.A = R2.A))

CREATE ASSERTION ADeterminesBNull
CHECK ((SELECT COUNT(*)
        FROM R AS R1,R AS R2
        WHERE ((R1.B != R2.B) AND (R1.A = R2.A)))
                OR ((R1.B is NULL) AND (R2.B is NOT NULL) AND (R1.A = R2.A))
        =0)
```

Exercise 19.10

Suppose you are given a relation *R(A,B,C,D)*. For each of the following sets of FDs, assuming they are the only dependencies that hold for *R*, do the following: (a) Identify the candidate key(s) for *R*. (b) State whether or not the proposed decomposition of *R* into smaller relations is a good decomposition and briefly explain why or why not.

1. $B \rightarrow C$, $D \rightarrow A$; decompose into *BC* and *AD*.
    a) BD
    b) The decomposition is bad because it is lossy
2. $AB \rightarrow C$, $C \rightarrow A$, $C \rightarrow D$; decompose into *ACD* and *BC*.
    a) AB, BC
    b) The decomposition is in BCNF and is lossless because ACD ∩ BC → ACD. It is not dependency preserving because AB → C is not preserved

3. $A \rightarrow BC$, $C \rightarrow AD$; decompose into *ABC* and *AD*.
    a) A, C
    b) The decomposition is in BCNF, but not dependency preserving because C → AD is not preserved
4. $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$; decompose into *AB* and *ACD*.
    a) A
    b) The decomposition is a lossless-join decomposition, but not dependency preserving because B → C is not preserved

5. $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$; decompose into AB, AD and CD.
    a) A
    b) The decomposition is a lossless BCNF, but not dependency preserving