Social Network Stanford Exercise:

Students at your hometown high school have decided to organize their social network using databases. So far, they have collected information about sixteen students in four grades, 9-12. Here's the schema:

Highschooler (ID, name, grade)

English: There is a high school student with unique ID and a given first name in a certain grade.

Friend (ID1, ID2)

English: The student with ID1 is friends with the student with ID2. Friendship is mutual, so if (123, 456) is in the Friend table, so is (456, 123).

Likes (ID1, ID2)

English: The student with ID1 likes the student with ID2. Liking someone is not necessarily mutual, so if (123, 456) is in the Likes table, there is no guarantee that (456, 123) is also present.

Q1. Find the names of all students who are friends with someone named Gabriel.

```
SELECT h.name
FROM highschooler h
WHERE h.id in (SELECT f.id1 FROM friend f WHERE f.id2 in (SELECT id FROM highschooler WHERE name='Gabriel')) or
      h.id in (SELECT f.id2 FROM friend f WHERE f.id1 in (SELECT id FROM highschooler WHERE name='Gabriel'));
```

Q2. For every student who likes someone 2 or more grades younger than themselves, return that student's name and grade, and the name and grade of the student they like.

```
SELECT h1.name, h1.grade, h2.name, h2.grade
FROM likes l
LEFT JOIN highschooler h1 ON l.id1=h1.id
LEFT JOIN highschooler h2 ON l.id2=h2.id
WHERE h1.grade-h2.grade>=2;
```

Q3. For every pair of students who both like each other, return the name and grade of both students. Include each pair only once, with the two names in alphabetical order.

```
SELECT h1.name, h1.grade, h2.name, h2.grade
FROM likes l, likes l1
LEFT JOIN highschooler h1 ON l.id1=h1.id
LEFT JOIN highschooler h2 ON l.id2=h2.id
WHERE (l.id1=l1.id2) and (l1.id1=l.id2) and (h1.name<h2.name);
```

Q4. Find all students who do not appear in the Likes table (as a student who likes or is liked) and return their names and grades. Sort by grade, then by name within each grade.

```
SELECT h.name, h.grade
FROM highschooler h
WHERE h.id not in (SELECT l.id1 FROM likes l) and
      h.id not in (SELECT l.id2 FROM likes l);
```

Q5. For every situation where student A likes student B, but we have no information about whom B likes (that is, B does not appear as an ID1 in the Likes table), return A and B's names and grades.

```
SELECT h1.name, h1.grade, h2.name, h2.grade
FROM likes l
LEFT JOIN highschooler h1 ON l.id1=h1.id
LEFT JOIN highschooler h2 ON l.id2=h2.id
WHERE l.id2 not in (SELECT id1 from likes);
```

Q6. Find names and grades of students who only have friends in the same grade. Return the result sorted by grade, then by name within each grade.

```
SELECT h.name, h.grade
FROM highschooler h
WHERE h.id not in (SELECT f.id1
                   FROM friend f
                   LEFT JOIN highschooler h1 ON f.id1=h1.id
                   LEFT JOIN highschooler h2 ON f.id2=h2.id
                   WHERE h1.grade<>h2.grade)
ORDER BY h.grade, h.name;
```

Q7. For each student A who likes a student B where the two are not friends, find if they have a friend C in common (who can introduce them!). For all such trios, return the name and grade of A, B, and C

```
SELECT DISTINCT H1.name, H1.grade, H2.name, H2.grade, H3.name, H3.grade
FROM Highschooler H1, Highschooler H2, Highschooler H3, Likes L, Friend F1, Friend F2
WHERE (H1.ID = L.ID1 AND H2.ID = L.ID2) AND H2.ID NOT IN (
  SELECT ID2
  FROM Friend
  WHERE ID1 = H1.ID
) AND (H1.ID = F1.ID1 AND H3.ID = F1.ID2) AND (H2.ID = F2.ID1 AND H3.ID = F2.ID2);
```

Q8. Find the difference between the number of students in the school and the number of different first names.

```
SELECT COUNT(h.id)-COUNT(DISTINCT h.name) as difference
FROM highschooler h;
```

Q9. Find the name and grade of all students who are liked by more than one other student.

```
SELECT h.name, h.grade
FROM highschooler h
WHERE h.id in (SELECT id2 FROM likes GROUP BY id2 HAVING COUNT(id2)>1);
```

Q10. For every situation where student A likes student B, but student B likes a different student C, return the names and grades of A, B, and C.

```
SELECT h1.name,h1.grade,h2.name,h2.grade,h3.name,h3.grade
FROM likes l1, likes l2
LEFT JOIN highschooler h1 ON  l1.id1=h1.id
LEFT JOIN highschooler h2 ON  l1.id2=h2.id
LEFT JOIN highschooler h3 ON  l2.id2=h3.id
WHERE (l1.id2=l2.id1) and (l1.id1 != l2.id2);
```

Q11. Find those students for whom all of their friends are in different grades from themselves. Return the students' names and grades.

```
SELECT h.name, h.grade
FROM highschooler h
WHERE h.id not in (SELECT f.id1
                   FROM friend f
                   LEFT JOIN highschooler h1 ON f.id1=h1.id
                   LEFT JOIN highschooler h2 ON f.id2=h2.id
                   WHERE h1.grade = h2.grade);
```

Q12. What is the average number of friends per student? (Your result should be just one number.)

```
SELECT AVG(count)
FROM (
  SELECT COUNT(*) AS count
  FROM Friend
  GROUP BY ID1
);
```

Q13. Find the number of students who are either friends with Cassandra or are friends of friends of Cassandra. Do not count Cassandra, even though technically she is a friend of a friend.

```
SELECT COUNT(*)
FROM Friend
WHERE ID1 IN (
  SELECT ID2
  FROM Friend
  WHERE ID1 IN (
    SELECT ID
    FROM Highschooler
    WHERE name = 'Cassandra'
  )
);
```

Q14. Find the name and grade of the student(s) with the greatest number of friends.

```
SELECT h.name, h.grade
FROM highschooler h
WHERE h.id in (SELECT f1.id1
               FROM friend f1
               GROUP BY f1.id1
               HAVING COUNT(f1.id2) = (SELECT MAX(t.count_) FROM (SELECT COUNT(f.id2) as count_ FROM friend f GROUP BY f.id1) t));
```