# SQL Social-Network Modification Exercises

Bookmark this page

Exercise due May 11, 2022 00:52 IST

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:

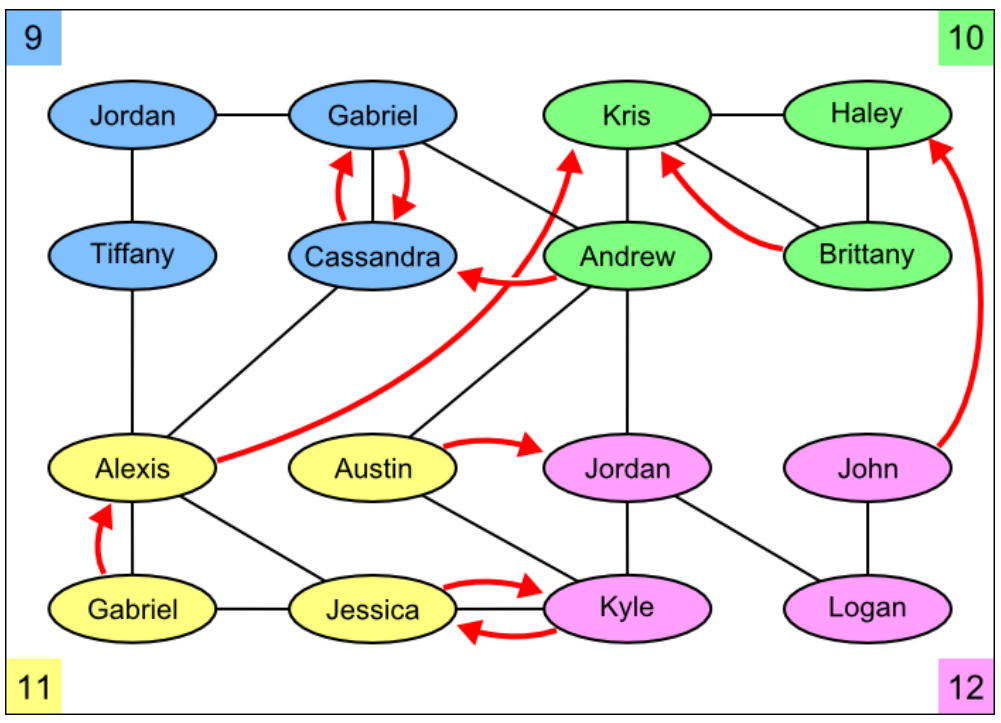
Highschooler ( ID, name, grade )
English: There is a high school student with unique *ID* and a given *first name* in a certain *grade*.

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.

Your modifications will run over a small data set conforming to the schema. View the database. (You can also download the schema and data.)

For your convenience, here is a graph showing the various connections between the people in our database. 9th graders are blue, 10th graders are green, 11th graders are yellow, and 12th graders are purple. Undirected black edges indicate friendships, and directed red edges indicate that one person likes another person.




**Instructions:** You are to write each of the following data modification commands using SQL. Our back-end runs each modification using SQLite on the original state of the sample database. It then performs a query over the modified database to check whether your command made the correct modification, and restores the database to its original state.

Hide Notes

restores the database to its original state.

You may perform these exercises as many times as you like, so we strongly encourage you to keep working with them until you complete the exercises with full credit.

---

## Q1

1/1 point (graded)

It's time for the seniors to graduate. Remove all 12th graders from Highschooler.

**Note:** Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
1 DELETE FROM Highschooler
2 where grade = 12;
```

Press ESC then TAB or click outside of the code editor to exit

Submit

Correct

---

## Q2

0/1 points (graded)

If two students A and B are friends, and A likes B but not vice-versa, remove the Likes tuple.

**Note:** Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
1 DELETE from Likes
2 where ID1 in
3 (SELECT f.ID1 FROM Friend as f
4 JOIN Likes as l
5 on f.ID1 = l.ID1 and f.ID2 = l.ID2
6 where f.ID1 not in (select ID2 from Likes as l1 where
7
```

Press ESC then TAB or click outside of the code editor to exit

Submit

Correct

To check your data modification statement, we ran the following query after your modification:

```
            SELECT H1.name,
```

Hide Notes

```
       H1.grade,
       H2.name,
       H2.grade
FROM Likes L,
     Highschooler H1,
     Highschooler H2
WHERE L.ID1 = H1.ID
  AND L.ID2 = H2.ID
ORDER BY H1.name,
         H1.grade;
```

Your Query Result:

| | | | |
|---|---|---|---|
| Alexis | 11 | Kris | 10 |
| Andrew | 10 | Cassandra | 9 |
| Austin | 11 | Jordan | 12 |
| Cassandra | 9 | Gabriel | 9 |
| Gabriel | 9 | Cassandra | 9 |
| Jessica | 11 | Kyle | 12 |
| John | 12 | Haley | 10 |
| Kyle | 12 | Jessica | 11 |

Expected Query Result:

| | | | |
|---|---|---|---|
| Alexis | 11 | Kris | 10 |
| Andrew | 10 | Cassandra | 9 |
| Austin | 11 | Jordan | 12 |
| Cassandra | 9 | Gabriel | 9 |
| Gabriel | 9 | Cassandra | 9 |
| Jessica | 11 | Kyle | 12 |
| John | 12 | Haley | 10 |
| Kyle | 12 | Jessica | 11 |

---

## Q3

0/1 points (graded)

For all cases where A is friends with B, and B is friends with C, add a new friendship for the pair A and C. Do not add duplicate friendships, friendships that already exist, or friendships with oneself. (This one is a bit challenging; congratulations if you get it right.)

**Note:** Your queries are executed using SQLite, so you must conform to the SQL constructs supported by SQLite.

```
insert into Friend
select distinct F1.ID1, F2.ID2
from Friend F1, Friend F2
where F1.ID2 = F2.ID1 and F1.ID1<>F2.ID2
      and F1.ID1 not in (select F3.ID1 from Friend F3
```

Press ESC then TAB or click outside of the code editor to exit

Submit

Correct

To check your data modification statement, we ran the following query after your modification:

```
            SELECT ID,
       name,
       grade,
  (SELECT count(*)
   FROM Friend
   WHERE id1 = H.id)
FROM Highschooler H
ORDER BY ID;
```

Your Query Result:

| | | | |
|---|---|---|---|
| 1025 | John | 12 | 2 |
| 1101 | Haley | 10 | 3 |
| 1247 | Alexis | 11 | 7 |
| 1304 | Jordan | 12 | 8 |
| 1316 | Austin | 11 | 6 |
| 1381 | Tiffany | 9 | 6 |
| 1468 | Kris | 10 | 6 |
| 1501 | Jessica | 11 | 7 |
| 1510 | Jordan | 9 | 5 |
| 1641 | Brittany | 10 | 3 |
| 1661 | Logan | 12 | 4 |
| 1689 | Gabriel | 9 | 8 |
| 1709 | Cassandra | 9 | 7 |
| 1782 | Andrew | 10 | 10 |
| 1911 | Gabriel | 11 | 5 |
| 1934 | Kyle | 12 | 7 |

Expected Query Result:

Previous Next

Affiliates
edX for Business
Open edX
Careers
News

## Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap

## Connect

Blog
Contact Us
Help Center
Media Kit



Hide Notes