

Database Designer: Ask a Database a Question

Ages 12-18 · about 45 minutes, in one sitting

Your child opens a real database of school clubs and asks it questions in SQL, the language most databases use. They say what the answer will be before every run, find out which clubs one member belongs to, first by hand and then with a JOIN, and see the database refuse data that points at nothing.

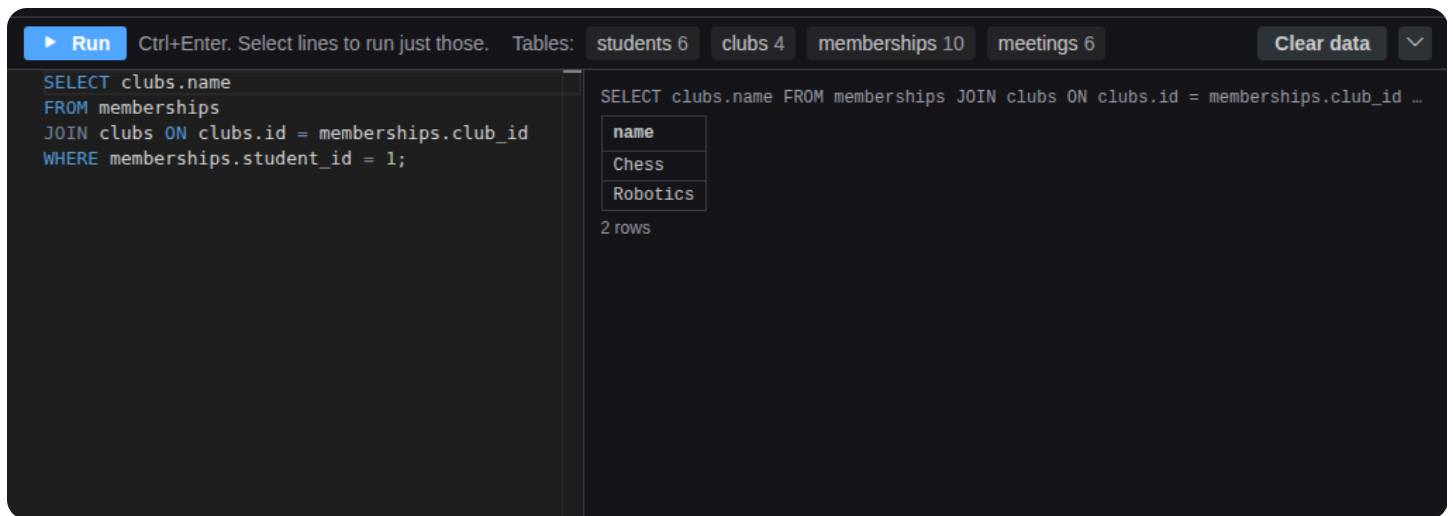
What you need

- A computer or Chromebook with a current browser.
- The lesson link: mytekdev.com/try/drawdb?lesson=ask-a-database-a-question&ref=fam-drawdb-1. No account, no email and nothing to install.
- Optional: the activity sheet at the end of this plan, printed, and a pencil, for writing each prediction down first.

Before you start: try it once yourself

Run the lesson's queries yourself once before you start, about ten minutes.

1. **Open the lesson.** Go to mytekdev.com/try/drawdb?lesson=ask-a-database-a-question&ref=fam-drawdb-1. The diagram has four tables. Press the SQL Query bar at the bottom of the diagram.
2. **See a table.** Press the `students` button after Tables: in the bar above the editor. It writes `SELECT * FROM "students";` and shows 6 rows.
3. **Ask with WHERE.** Type `SELECT name FROM students WHERE grade = 9;` and press Run: Ana Lopez and Dev Patel.
4. **Follow the numbers.** Run `SELECT * FROM memberships WHERE student_id = 1;`: club_id 1 and 2. Then the JOIN: `SELECT clubs.name FROM memberships JOIN clubs ON clubs.id = memberships.club_id WHERE memberships.student_id = 1;`: Chess and Robotics.
5. **Break a key.** Run `INSERT INTO memberships (student_id, club_id, role) VALUES (4, 7, 'member');`: Statement 1: FOREIGN KEY constraint failed. Change 7 to 3 and run it: 1 row changed.



The JOIN turns Ana's two club ids into names.

Every Run builds the database again from the diagram, with the rows saved in the project, so a change to the diagram shows up at once. Rows added with INSERT are saved with the project. Clear data empties every table, and opening the lesson link in a new tab starts over; reloading the same tab keeps the changes.

The ideas, in plain words

A database keeps each kind of thing in its own table: `students` in one, `clubs` in another. Each row is one person or one club, and every row has an id that no other row in that table has, its primary key. SQL is how you ask: SELECT names the columns you want, FROM names the table, WHERE keeps only the rows that match, and ORDER BY sorts them.

Facts that connect two things live in a table of their own. A person can be in many clubs, and a club has many members, so the database keeps a `memberships` table with one row per pairing: person 1 is in club 2. That table holds only ids, which is why reading it by hand means looking each id up. A JOIN does the looking up: it matches each membership to the club whose id equals its `club_id`, so the answer comes back as names.

The lines in the diagram are foreign keys: `memberships.club_id` must match some `clubs.id`. The database enforces that, so a membership in a club that does not exist is refused. In this database the lines are also set to delete a club's memberships when the club is deleted; that rule is part of the design, and the diagram is where it was decided. The queries run in SQLite, a real database engine, inside the browser.

YOUR CHILD MIGHT ASK	A GOOD ANSWER
Why not just put the club names in the <code>students</code> table?	One person can be in several clubs. One column cannot hold a list, and a list in one cell cannot be searched or checked.
Why are there two name columns in my result?	Both tables have a column called name. Write <code>students.name AS student</code> to give it its own heading.
Why did it say no such column?	The name in the query must match the diagram exactly, including underscores.

YOUR CHILD MIGHT ASK

A GOOD ANSWER

Is this a real database?

Yes: the queries run in SQLite, the database engine inside most phones and browsers.

Do it together

1. Open mytekdev.com/try/drawdb?lesson=ask-a-database-a-question&ref=fam-drawdb-1 together. The diagram shows four tables: `students`, `clubs`, `memberships` and `meetings`. A short guide runs along the bottom; follow it together. It has your child open the SQL Query bar, press the `students` button, and run a first query.

You'll know it worked when: The `students` button shows six rows, from Ana Lopez to Fatima Noor, and the first query returns Ana Lopez and Dev Patel.

2. Read the diagram together: each box is a table, each line in it a column, and the key symbol marks id, the primary key. Then your child predicts and runs `SELECT name FROM students WHERE grade = 10;` and `SELECT name, room FROM clubs ORDER BY name;`

If you get stuck: Names have to match the diagram exactly. A misspelling gives no such table or no such column, with a hint under it.

You'll know it worked when: Ben Carter and Fatima Noor; then Art, Chess, Debate and Robotics with their rooms.

3. Which clubs is Ana in? Run `SELECT * FROM memberships WHERE student_id = 1;` and look up each `club_id` in the clubs table by hand.

You'll know it worked when: Club 1 is Chess and club 2 is Robotics.

4. Now let the database do it: `SELECT clubs.name FROM memberships JOIN clubs ON clubs.id = memberships.club_id WHERE memberships.student_id = 1;`

You'll know it worked when: Chess and Robotics, as names this time.

5. Break a rule on purpose: `INSERT INTO memberships (student_id, club_id, role) VALUES (4, 7, 'member');` Then change 7 to 3 and run it again.

You'll know it worked when: The first try says FOREIGN KEY constraint failed, because there is no club 7. With club 3 it works, and Dev is now in Chess and Art.

6. Finish with one question: why is it good that the database refused club 7?

You'll know it worked when: Your child explains that the membership would have pointed at a club that does not exist, and the foreign key prevents that.

Talk about it

Ask these while you work together. A child who has understood the lesson can answer them without help:

- Before you press Run: what will this query return?
- Which part of your query picks the rows, and which picks the columns?
- Why does memberships hold numbers instead of names?
- In your JOIN, which two columns have to match?
- Why did the database refuse club 7?

The third one is the real test. A child who can explain why a linking table holds ids understands the idea every relational database is built on.

For younger and older kids

Younger: For ages 10 and 11, use the table buttons and run the SELECT queries side by side, and do the JOIN together.

Older: For older teens, add a JOIN of three tables (`students` , `memberships` and `clubs`) and a COUNT of members per club with GROUP BY.

Keep going

- Count each club's members: `SELECT clubs.name, COUNT(*) AS members FROM memberships JOIN clubs ON clubs.id = memberships.club_id GROUP BY clubs.name;` (Chess and Robotics have 3, Art and Debate 2, before any changes.)
- Join three tables: list every person's name next to every club they are in.
- The line from memberships to clubs is set to delete memberships when their club is deleted. Predict what `DELETE FROM clubs WHERE name = 'Chess';` does to memberships, then run `SELECT COUNT(*) FROM memberships;` before and after.

Bring it into other subjects

- Math: COUNT(*) and GROUP BY give the number of members in each club. Make a bar chart of the result by hand.
- Data science: write a query that answers a question a club advisor might ask, such as which `grade` has the most club members.

- Computer science: open File, Export SQL, SQLite. Find the CREATE TABLE for memberships and the FOREIGN KEY lines that came from the diagram's lines.
- Careers: database administrators and data analysts write queries like these every day. Find one job listing that asks for SQL.

For your records

Ready to copy into an hour log, a portfolio or a transcript.

LESSON	Database Designer: Ask a Database a Question
SUBJECTS	Computer science
TIME	45 minutes
WHAT WE DID	Queried a real SQLite database of school clubs: wrote SELECT statements with WHERE and ORDER BY, predicted each answer first, used a JOIN to match a linking table to names, and saw a foreign key refuse data that pointed at nothing.
SKILLS PRACTICED	SQL SELECT, WHERE and ORDER BY; JOINs across tables; primary and foreign keys; predicting before running

Activity sheet (optional)

Name: _____ Date: _____



Open the app: scan the QR code, or go to mytekdev.com/try/drawdb?lesson=ask-a-database-a-question

Essential question: How do you ask a database a question, and how does it connect facts stored in different tables?

- 1. **Open the panel.** Press the SQL Query bar under the diagram. The buttons are the tables.
- 2. **Read the diagram.** The primary key of every table is _____. The memberships table links _____ to _____.
- 3. **Predict, then run.** Write your prediction before pressing Run every time.

QUERY	I PREDICT	THE DATABASE SAYS
SELECT name FROM students WHERE grade = 10;		
SELECT name, room FROM clubs ORDER BY name; (first row)		
SELECT * FROM memberships WHERE student_id = 1; (club ids)		
My query: Robotics' meetings, in date order		
The JOIN for Ana's clubs		
My JOIN: everyone in Robotics		

QUERY	I PREDICT	THE DATABASE SAYS
<pre>INSERT INTO memberships (student_id, club_id, role) VALUES (4, 7, 'member'); (Dev into club 7)</pre>		

1. Write your JOIN that lists everyone in Robotics:

2. Why did the database refuse club 7?

Quick check

Name: _____

1. In `SELECT name FROM students WHERE grade = 9;` which part picks the rows? (A) `SELECT name`, (B) `FROM students`, (C) `WHERE grade = 9`.

2. Why does the memberships table hold `student_id` and `club_id` instead of names? (A) numbers are smaller, (B) one person can be in many clubs and a club has many members, so the pairings get their own table, (C) names are not allowed in databases.

3. A new row in memberships says `club_id` 12, and there is no club 12. What does the database do, and why is that useful?

Answers

Measured in the app on the lesson's data: `grade` 9 is Ana Lopez and Dev Patel (and in the same six rows, `grade` 10, the worksheet's first row, is Ben Carter and Fatima Noor); clubs by name are Art, Chess, Debate and Robotics; Ana's memberships are `club_id` 1 and 2, which the JOIN shows as Chess and Robotics; Robotics (`club` 2) meets on 2026-10-08 (Build the drivetrain) and 2026-10-15 (Program the arm); Robotics' members are Ana Lopez, Ben Carter and Fatima Noor; inserting club 7 gives FOREIGN KEY constraint failed, and with club 3 Dev's clubs are Chess and Art.

QUESTION	ANSWER	WHY
1	C	WHERE keeps only the rows that match. SELECT picks the columns and FROM picks the table.
2	B	Pairings are many to many, so neither table can hold the list; memberships holds one row per pairing.

QUESTION	ANSWER	WHY
3	It refuses the row (FOREIGN KEY constraint failed)	A good answer says it refuses the row and why: the membership would point at a club that does not exist.

Want the whole subject taught?

This is one lesson to do together. If you would rather have the whole subject taught, our Computer Science 1 course teaches databases and SQL as part of a full course, with an instructor, live and recorded classes, graded work and credit.

If your state or umbrella school asks, this lesson covers CSTA 2017 3A-DA-10, CSTA 2017 2-DA-08.

Database Designer is built on drawDB by the drawDB authors, open source under the GNU Affero General Public License 3.0. Its queries run in SQLite through sql.js, open source under the MIT License.

Terms of use

This lesson was made by mytekDev. You may print and copy it for the children you teach at home, and change it for them. You may not sell it, share it with other families, or post it publicly.

The app this lesson uses is free and runs in the browser, with nothing to install and no account needed. It is at mytekdev.com.

© 2026 mytekDev. All rights reserved.