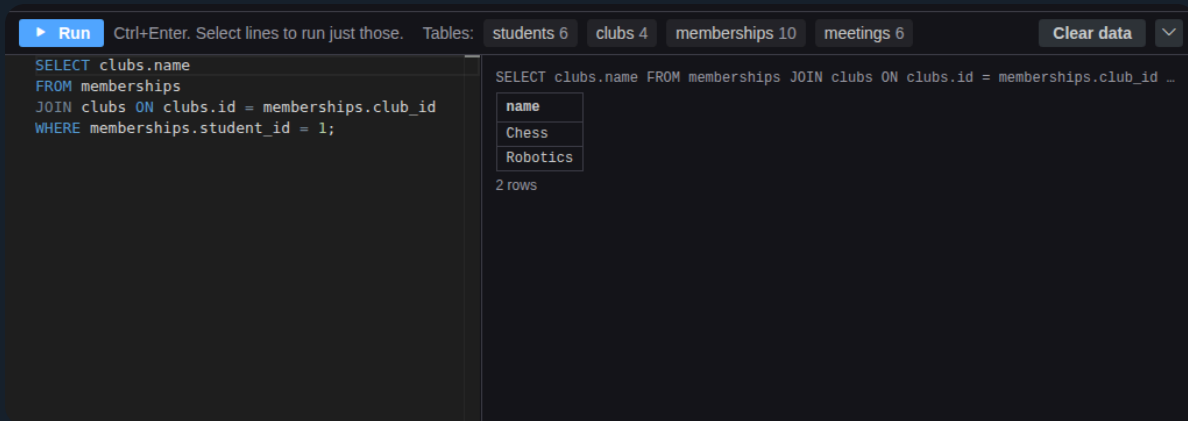


mytekDev

Database Designer: Ask a Database a Question

Grades 7-12 · 45 minutes

Computer science



Slides, rubric and answer key

Print and go

No student accounts

Real SQL in the browser

What's included

- A one-page overview: objectives, essential question, vocabulary, standards and assessment
- A ten-minute walkthrough to rehearse before class, and background in plain words for the teacher
- A 45-minute lesson sequence: warm-up, direct instruction, guided practice, independent practice, closure
- Slides for the board: a PowerPoint file that also opens in Google Slides
- A student worksheet with a QR code
- A 3-question exit ticket, two to a page, with an answer key
- A rubric: 3 criteria, 4 levels, scored out of 12
- Differentiation for English learners, students with IEPs or 504 plans, and students who need more support or more challenge
- Cross-curricular extensions, and notes for teaching it in grades 5-6 and in a college course
- Standards: CSTA 2017

Database Designer: Ask a Database a Question

Grades 7-12 · 45 minutes · Computer science

Students open the database behind a school's clubs: students, clubs, the memberships that link them, and meetings, already filled with data. They ask it questions in SQL, writing their prediction before every Run: who is in grade 10, which room each club meets in, and which clubs Ana belongs to. The memberships table answers that last one only in numbers, so students decode it by hand, then write a JOIN that does it for them. They finish by adding a membership for a club that does not exist and watching the database refuse it. By the end, students can read a diagram, write a SELECT with WHERE and ORDER BY, and explain what a JOIN and a foreign key are for.

At a glance

ESSENTIAL QUESTION

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

OBJECTIVES

Students will be able to:

- read a database diagram: tables, columns, primary keys and the lines between tables
- write SELECT statements with WHERE and ORDER BY, predicting each answer before running it
- use a JOIN to turn the ids in a linking table into names
- explain how a foreign key stops data that points at nothing

VOCABULARY

database, table, SQL, query, primary key, foreign key, JOIN

STANDARDS

3A-DA-10, 2-DA-08

TIME AND MATERIALS

45 minutes, grades 7-12.

- One computer or Chromebook per student or pair
- A projector
- The worksheet and exit ticket, one copy each per student
- The rubric, one copy per student if you grade the project

PREP

Make copies of the worksheet and exit ticket. Before class, open mytekdev.com/try/drawdb?lesson=ask-a-database-a-question&ref=tpt-drawdb-1 once on a student computer, open the SQL Query bar and press Run on one query, to check that your school's filter allows it (about ten minutes, in Before you teach). No account is needed.

ASSESSMENT

- Formative: the questions under What to look for, asked while students work, and the exit ticket (answer key included).
- Summative: the worksheet's prediction table and the student's own JOIN query, scored with the rubric out of 12.

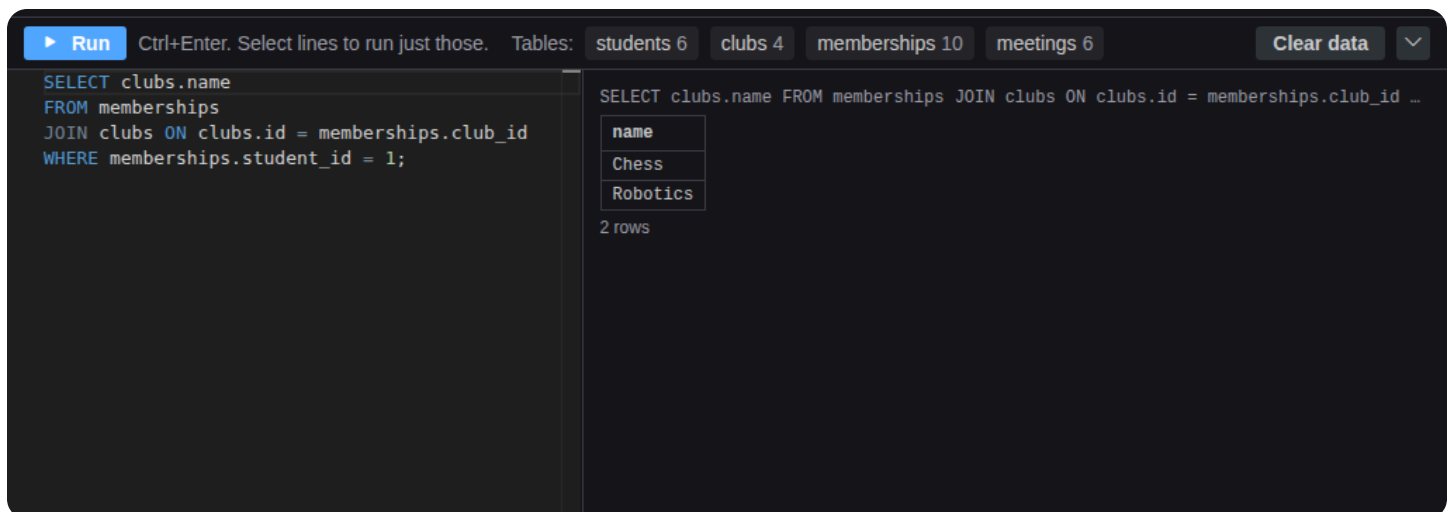
TECHNOLOGY AND PRIVACY

Runs in any browser, Chromebooks included, with nothing to install. The queries run in a real database engine, SQLite, inside the browser, so nothing is sent to a server. Students need no account and no email: the link works signed out, and nothing they make is stored on our server unless they choose to save it to an account. If your school blocks new sites, ask IT to allow mytekdev.com.

Before you teach: do it once yourself

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

1. **Open the lesson.** Go to mytekdev.com/try/drawdb?lesson=ask-a-database-a-question&ref=tpt-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.

Background 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.

STUDENTS 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.
Is this a real database?	Yes: the queries run in SQLite, the database engine inside most phones and browsers.

Lesson sequence

MINUTES	STAGE	WHAT YOU DO	WHAT STUDENTS DO
0 to 6	Warm-up Hook	Show the title, essential question and Today I can slides (about 1 minute). Then have students open the link or scan the QR code on the worksheet. The app opens on a diagram of four tables: students, clubs, memberships and meetings, with lines between them. A short guide runs along the bottom; tell students to follow it. It has them open the SQL Query bar, press the students button to see its six rows, and change that query to <code>SELECT name FROM students WHERE grade = 9;</code> and run it. Ask: "How would you find out which clubs Ana is in?" Take a few ideas.	Open the database, follow the guide, and answer the question.

MINUTES	STAGE	WHAT YOU DO	WHAT STUDENTS DO
6 to 14	Direct instruction (I do) Model it: read the diagram, then ask	Show the vocabulary and Three words of SQL. On the projector, point to the diagram: each box is a table, each line in it a column, and the key symbol marks the primary key, id. The line from memberships to students says memberships.student_id holds a student's id; the 1 and the n at its ends say one student can have many memberships. In the panel, run <code>SELECT * FROM students;</code> (6 rows), then the guide's <code>SELECT name FROM students WHERE grade = 9;</code> (Ana Lopez and Dev Patel) and ask which part picked those two rows. Change 9 to 10, and before pressing Run, ask the class to predict: Ben Carter and Fatima Noor. Then, predicting first, <code>SELECT name, room FROM clubs ORDER BY name;</code> (Art, Chess, Debate, Robotics). Say: the buttons after Tables: in the bar above the editor are the tables, with how many rows each holds; pressing one replaces what is in the editor with its SELECT and runs it. It's working when: Ctrl+Enter runs the query too. Several statements separated by semicolons run in order, and each shows its own result.	Predict each answer out loud, then watch it run, and fill in worksheet step 2.
14 to 24	Guided practice (We do) Together: which clubs is Ana in?	Ana's id is 1. Everyone runs <code>SELECT * FROM memberships WHERE student_id = 1;</code> Two rows come back, with club_id 1 and 2. Numbers, not names: ask students to look them up by hand in clubs (1 is Chess, 2 is Robotics). Then show the JOIN: <code>SELECT clubs.name FROM memberships JOIN clubs ON clubs.id = memberships.club_id WHERE memberships.student_id = 1;</code> It returns Chess and Robotics. Read it aloud: take each membership, find the club whose id matches its club_id, and show that club's name. This is why memberships exists: a student can be in many clubs and a club has many students, so neither table can hold the list, and the linking table holds one row per pairing. Watch for: Names must match the diagram exactly: memberships, club_id. A misspelled name gives no such table or no such column, with a hint under it.	Decode Ana's memberships by hand, then type the JOIN and run it.

MINUTES	STAGE	WHAT YOU DO	WHAT STUDENTS DO
24 to 38	Independent practice (You do) Your questions, and a broken key	Students work through the worksheet's prediction table, writing each answer before running it: Robotics' meetings in date order, and a JOIN of their own that lists everyone in Robotics (Ana Lopez, Ben Carter and Fatima Noor). Then: <code>INSERT INTO memberships (student_id, club_id, role) VALUES (4, 7, 'member');</code> The answer is Statement 1: FOREIGN KEY constraint failed, because there is no club 7. Students fix it with club 3, run it, and check Dev's clubs with a JOIN: Chess and Art. Walk the room and ask the questions under What to look for. Watch for: A join of students and clubs returns two columns both called name. Writing students.name AS student, clubs.name AS club gives them different headings. Watch for: Rows added with INSERT stay with the project. If a student's data gets confusing, Clear data empties every table, and opening the lesson link in a new tab starts over with the original rows. Reloading the same tab keeps the changes.	Predict and run each query, write a JOIN of their own, and break and fix a foreign key.
38 to 45	Closure Explain it, and exit ticket	Show the last slide and ask: "The database refused Dev's membership in club 7. Why is that a good thing?" (A membership in a club that does not exist would be data that points at nothing; the foreign key, the line in the diagram, makes the database refuse it.) Ask the essential question again, then hand out the exit ticket.	Answer out loud, then answer the exit ticket on their own.

What to look for

Ask these while students work. A student 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 student who can explain why a linking table holds ids understands the idea every relational database is built on.

Cross-curricular extensions

- 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.

Vocabulary

database	An organized collection of data that a program can search and change.
table	One kind of thing in a database, like students or clubs. Each row is one of them; each column is one fact about it.
SQL	The language used to ask a database questions and change its data.
query	One question asked in SQL, such as SELECT name FROM students;
primary key	A column whose value is different in every row, usually id, so each row can be found exactly.
foreign key	A column that holds another table's primary key, like club_id in memberships. The database refuses a value that matches no row.
JOIN	Part of a query that matches rows from two tables, for example memberships to clubs by club id.

Differentiation

ENGLISH LEARNERS

- Post the vocabulary with a picture for each word: a table as a grid, a key on the id column, a line between two tables.
- Give the SQL keywords on a card with their meaning: SELECT (which columns), FROM (which table), WHERE (which rows).
- Let students record answers by copying the rows from the screen.

STUDENTS WHO NEED MORE SUPPORT

- Use the table buttons first: each writes a working SELECT to change one part at a time.
- Give a sentence starter: "I predict the answer will be ... because ..."
- Check in at minute 30: is the student writing a prediction before every Run?

STUDENTS WITH IEPS OR 504 PLANS

- Where the IEP allows a modified assignment, do the guided JOIN together and skip writing a JOIN of their own.
- Give the queries printed, so students edit one word instead of typing whole statements.
- Pair students: one types, the other reads the result aloud.
- Read the exit ticket aloud, or let students answer by pointing at the result on their screen.

STUDENTS READY FOR MORE

- 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.

Adapting for other grades

Younger students: For grades 5 and 6, run the SELECT queries together and use the table buttons, do the JOIN as a class demonstration, and score only the first two rubric rows (out of 8; 6 or more meets the objectives).

Older students: For a college course, add GROUP BY and HAVING, write the three-table join, and compare an inner join with a LEFT JOIN that also lists students in no club.

Student worksheet

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**

```
INSERT INTO memberships  
(student_id, club_id, role) VALUES  
(4, 7, 'member'); (Dev into club 7)
```

1. Write your JOIN that lists everyone in Robotics:

2. Why did the database refuse club 7?

Exit ticket

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?

Exit ticket

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?

Answer key: exit ticket

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.
3	It refuses the row (FOREIGN KEY constraint failed)	Full credit for refusing and a reason: the membership would point at a club that does not exist.

Rubric

Score each row from 1 to 4.

Name: _____

CRITERIA	4 EXCEEDS	3 MEETS	2 APPROACHING	1 BEGINNING
Predicts, then runs	Every prediction was written before Run and has a result, and the student explains the one that surprised them most.	Every prediction was written before Run and has a result.	Results are written, but some predictions are missing or were written after.	Queries were run, but nothing is written down.
Writes SELECT queries	Writes queries with WHERE and ORDER BY that answer new questions, without help.	Writes queries with WHERE and ORDER BY that answer the worksheet's questions.	Changes a given query, but cannot yet write one.	Cannot yet run a query.
Explains JOINS and keys	Writes a working JOIN of their own and explains why the foreign key refused club 7.	Writes a working JOIN with the worksheet's help and says what a foreign key is for.	Runs the given JOIN but cannot explain what it matches.	Cannot yet say what a JOIN or a foreign key does.

Total: ____ / 12. A score of 9 or more meets the objectives.

Standards

These CSTA 2017 standards describe storing, querying and connecting data. Students read a database's structure and write SQL that answers questions across linked tables.

FRAMEWORK	CODE	GRADES	STANDARD	HOW THE LESSON MEETS IT
CSTA 2017	3A-DA-10	9-10	Evaluate the tradeoffs in how data elements are organized and where data is stored.	In part: students explain why memberships stores pairs of ids in its own table instead of a list of names in each row (exit ticket question 2). Lesson 2 weighs the tradeoffs in full.
CSTA 2017	2-DA-08	6-8	Collect data using computational tools and transform the data to make it more useful and reliable.	Students use SQL to pull out only the rows that answer a question, use a JOIN to turn a table of ids into names, and see a foreign key keep a membership in a missing club out of the data.

Checked October 2026 against: CSTA 2017 progression chart, CSTA 2017 to 2026 crosswalk.

Optional: save student work in a free class

The whole lesson runs at the link on the worksheet, with no accounts and no sign-up. Work stays in the browser tab until it is closed.

To keep students' work, and see every student's database in one place, make a free class. It takes about ten minutes, once, and covers every lesson in this series: mytekdev.com/tools/class-setup

Students in a class open mytekdev.com/tools/database-designer/lessons/ask-a-database-a-question/start instead of the worksheet link. Each one gets their own copy of the database, with its data, and the same link reopens it next time. A student who started at the worksheet link and then signs up keeps what they made.

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

Thank you for downloading this lesson. The plan, the handouts and the art were made by mytekDev.

You may:

- print and copy it for your students
- change it for your own classes
- post it on your class website or learning management system
- share this file with other teachers, whole and unchanged, with this page included

You may not:

- sell it, or put it in a product you sell
- share a changed copy
- claim it as your own

The app this lesson uses is free and runs in the browser. Students need no account. It is at mytekdev.com.

Questions, or found a mistake? Ask on the Teachers Pay Teachers Q&A for this resource, or write to us at mytekdev.com/contact.

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.

© 2026 mytekDev. All rights reserved.