Intro to SQL
Databases
Databases

Data can be stored in a database, a program that knows how to store, modify, and retrieve data.

A **relational** database stores each kind of data in a table. Each table has columns and rows.

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Row 2</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Row 3</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Example tables

### users:

<table>
<thead>
<tr>
<th>id</th>
<th>nickname</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sal</td>
<td>California</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>New York</td>
</tr>
<tr>
<td>3</td>
<td>MayLi</td>
<td>Washington</td>
</tr>
</tbody>
</table>

### badges:

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Guru</td>
<td>1000</td>
</tr>
<tr>
<td>2</td>
<td>Oracle</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>Wizard</td>
<td>500</td>
</tr>
</tbody>
</table>
## Relating Tables

### users:
<table>
<thead>
<tr>
<th>id</th>
<th>nickname</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sal</td>
<td>California</td>
</tr>
<tr>
<td>2</td>
<td>John</td>
<td>New York</td>
</tr>
<tr>
<td>3</td>
<td>MayLi</td>
<td>Washington</td>
</tr>
</tbody>
</table>

### badges:
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Guru</td>
<td>1000</td>
</tr>
<tr>
<td>2</td>
<td>Oracle</td>
<td>2000</td>
</tr>
<tr>
<td>3</td>
<td>Wizard</td>
<td>500</td>
</tr>
</tbody>
</table>

### user_badges:
<table>
<thead>
<tr>
<th>user_id</th>
<th>badge_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
SQL
Structured Query Language

Originally called SEQUEL!
(Structured English QUEry Language)

The most popular language for interacting with relational databases. Commands like...

- CREATE TABLE ...
- INSERT INTO ...
- UPDATE ...
- SELECT * FROM ...
CREATE TABLE users (id INTEGER PRIMARY KEY,
    nickname TEXT, location TEXT);

INSERT INTO users VALUES (1, "Sal", "California");
INSERT INTO users VALUES (2, "John", "New York");
INSERT INTO users VALUES (3, "MayLi", "Indonesia");

SELECT * FROM users;

Try it on code.cs61a.org or khanacademy.org

Documentation: SQLite.org: CREATE TABLE, SQLite.org: INSERT, KA: CREATE TABLE, KA: INSERT
Update column values of existing rows

```sql
CREATE TABLE users (id INTEGER PRIMARY KEY,
nickname TEXT, location TEXT);

INSERT INTO users VALUES (1, "Sal", "California");
INSERT INTO users VALUES (2, "John", "New York");
INSERT INTO users VALUES (3, "MayLi", "Indonesia");

UPDATE users SET nickname = "Joan" WHERE id = 2;
```

Documentation: KA: UPDATE
DELETE

Delete entire rows that match criteria

```sql
CREATE TABLE users (id INTEGER PRIMARY KEY,
nickname TEXT, location TEXT);

INSERT INTO users VALUES (1, "Sal", "California");
INSERT INTO users VALUES (2, "John", "New York");
INSERT INTO users VALUES (3, "MayLi", "Indonesia");

DELETE FROM users WHERE id = 3;
```

Documentation: **KA: DELETE**
SELECT

Most of the complexity and power of SQL is in the `SELECT` statement.

```
SELECT nickname FROM users;

SELECT nickname, location FROM users;

SELECT nickname, location FROM users ORDER BY nickname;

SELECT nickname FROM users WHERE location = "California";
```

Documentation: SQLite.org: SELECT,
KA: SELECT, KA: SELECT WHERE, KA: ORDER BY,
Aggregate functions

Aggregates column values in some way:

**MAX/MIN, SUM, AVG, COUNT**

```sql
CREATE TABLE groceries (id INTEGER PRIMARY KEY,
    name TEXT, quantity INTEGER, aisle INTEGER);
INSERT INTO groceries VALUES (1, "Bananas", 56, 7);
INSERT INTO groceries VALUES (2, "Peanut Butter", 1, 2);
INSERT INTO groceries VALUES (3, "Dark Chocolate Bars", 2, 2);
INSERT INTO groceries VALUES (4, "Ice cream", 1, 12);
INSERT INTO groceries VALUES (5, "Cherries", 6, 2);
INSERT INTO groceries VALUES (6, "Chocolate syrup", 1, 4);

SELECT MAX(quantity) FROM groceries;
```

Documentation: SQLite.org: Aggregate functions,
KA: SELECT with aggregate
GROUP BY

Groups rows that share the same column value

| CREATE TABLE groceries (id INTEGER PRIMARY KEY,  |
| name TEXT, quantity INTEGER, aisle INTEGER); |
| INSERT INTO groceries VALUES (1, "Bananas", 56, 7); |
| INSERT INTO groceries VALUES (2, "Peanut Butter", 1, 2); |
| INSERT INTO groceries VALUES (3, "Dark Chocolate Bars", 2, 2); |
| INSERT INTO groceries VALUES (4, "Ice cream", 1, 12); |
| INSERT INTO groceries VALUES (5, "Cherries", 6, 2); |
| INSERT INTO groceries VALUES (6, "Chocolate syrup", 1, 4); |

SELECT aisle, SUM(quantity) FROM groceries GROUP BY aisle;

Documentation: SQLite.org: Generating results,
KA: SELECT with GROUP BY
JOIN

Generates results from two related tables that share a column.

CREATE TABLE students (id INTEGER PRIMARY KEY, first_name TEXT, last_name TEXT, email TEXT, phone TEXT, birthdate TEXT);

INSERT INTO students (first_name, last_name, email, phone, birthdate) VALUES ('Peter', 'Rabbit', 'peter@rabbit.com', '555-6666', '2002-06-24');
INSERT INTO students (first_name, last_name, email, phone, birthdate) VALUES ('Alice', 'Wonderland', 'alice@wonderland.com', '555-4444', '2002-07-04');

CREATE TABLE student_projects (id INTEGER PRIMARY KEY, student_id INTEGER, title TEXT);

INSERT INTO student_projects (student_id, title) VALUES (1, 'Carrotapault');

SELECT students.first_name, students.last_name, student_projects.title
FROM students
JOIN student_projects
ON students.id = student_projects.student_id;

Documentation: KA: JOIN
Where is SQL used?

1) Application data storage and 2) data analysis
Demo App: Native or Not?

flask-db-example.pamelafox2.repl.co
Demo App: 61A Merch

flask-db-example-1.pamelafox2.repl.co/

Includes a transaction:

```
BEGIN;

UPDATE products SET quantity = quantity - 1
    WHERE id = 1;

INSERT INTO orders (customer, product_id)
    VALUES ("Animesh", 1);

COMMIT;
```
Further learning

There's lots we didn't cover today! **IN, HAVING, CASE, ALTER TABLE**, more **JOIN**s, transactions, security, etc.

Online learning:

- Khan Academy SQL course
- Kaggle: Intro to SQL
- Stanford's Database courses
- Coursera: Exploring data with BigQuery

Berkeley classes:

- **CS 186**: Intro to Database Systems
- **Data 100**: Principles and Techniques of Data Science
- **CS194/INFO290**: Data Engineering