

Introduction to SQL



BEW 1.2

Warm-Up - My Favorite Mistake (8 minutes)

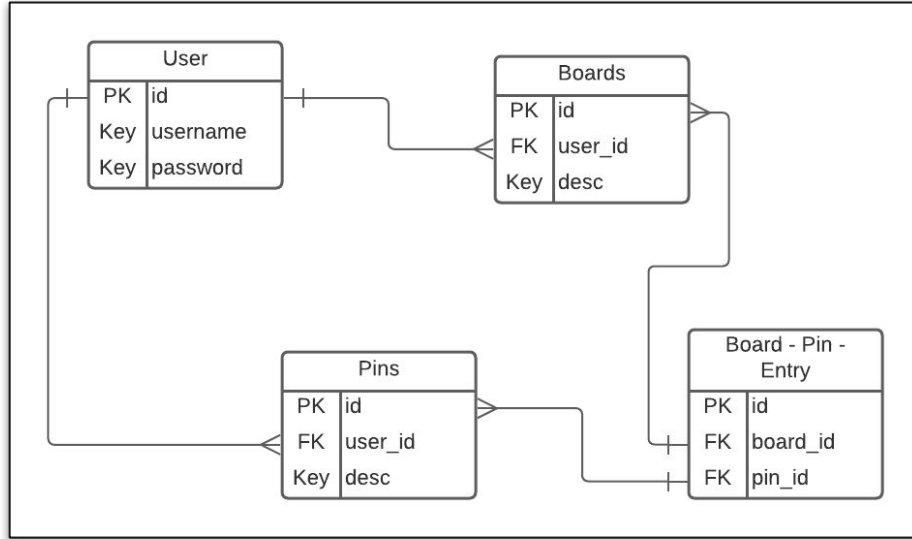
In a group of 3, discuss each of the following **3 submissions** and answer:

1. What did this person do **right**?
2. What **misconception** did this person have?

Go to the next slides (3, 4, 5) to view the submissions.

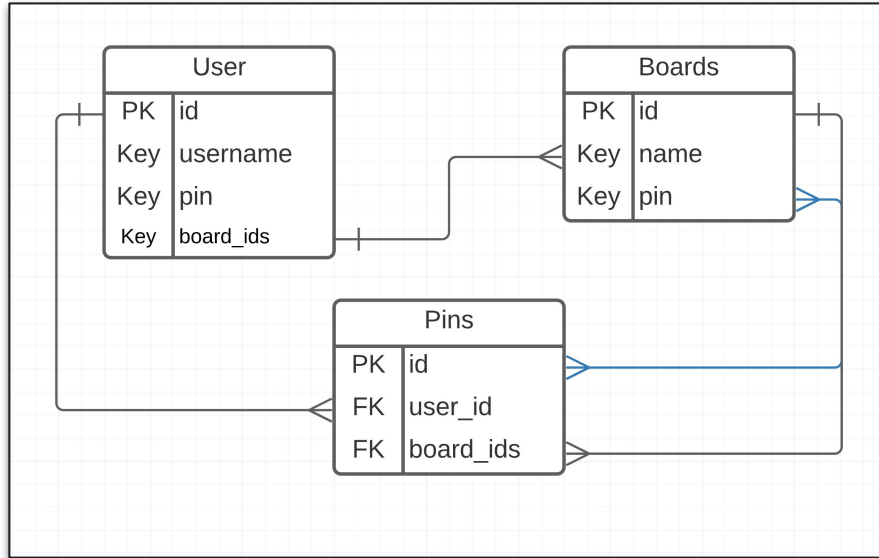
Warm-Up - My Favorite Mistake

Submission 1: Pinterest Diagram



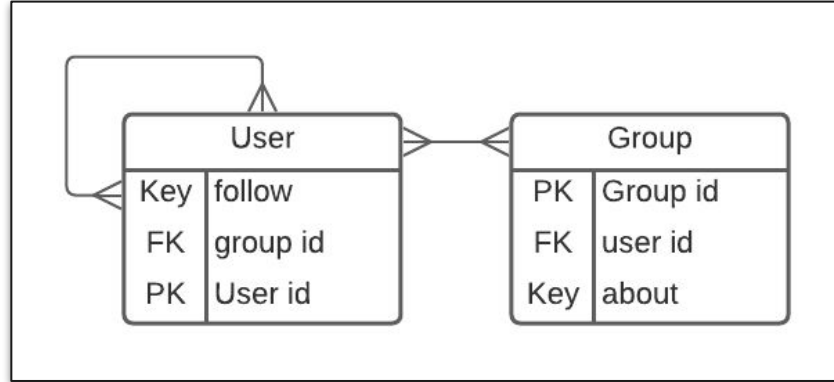
Warm-Up - My Favorite Mistake

Submission 2: Pinterest Diagram



Warm-Up - My Favorite Mistake

Submission 3: Facebook diagram



- Learning Outcomes
- What is SQL?
- Making Queries
- Activity: Select Star SQL
- **BREAK**
- SQL Joins
- Activity
- Wrap-Up

By the end of today, you should be able to...

1. **Write** SELECT queries in SQL to retrieve a subset of data from a table.
2. **Define** a SQL table, then add, delete, and modify data using sqlite code.
3. **Explain** why joins are useful, and **use** an inner join to query a SQL table.

What is SQL?

What are SQL databases?

SQL (or, "sequel") stands for **Structured Query Language**. It is a programming language that is used to directly communicate with a database.



```
INSERT INTO Customers  
(first_name, last_name)  
VALUES ('Mary',  
       'Bright')
```



OK

What is SQL?

It can be helpful to remember that **everything in SQL is stored in tables.**

Unlike in NoSQL, we can't store arbitrary objects - only keys and values.

For example, this might be our **Users** table:

id	first_name	last_name	birth_date
30015	Jane	Programmer	2/5/1997
30016	I.M.	Hacker	8/13/2000
30017	Joe	Maker	5/23/1992
...

SQL also isn't just one language. There are many different "dialects" of SQL, each of which has some minor differences. These include:

- MySQL
- Oracle
- PostgreSQL
- SQLite

The differences aren't too important, though - it's more important to understand the ***concepts*** so that you can apply them to any scenario.

SQL:

- Pre-defined structure
- Data is predictable - You always know what columns you will get.
- Table based
- Can associate data in multiple tables using joins

MongoDB (NoSQL):

- Unstructured - Can insert any data, as long as it's in Python dictionary format
- Data is unpredictable - A document can contain any key/values.
- Document based
- Harder to associate data from multiple tables

Making Queries

Select Star Sql (25 minutes)

With a partner, read and do the exercises in [Beazley's Last Statement \(Chapter 1\)](#) of Select Star SQL.

Chapters 2, 3, and 4 are optional - but you can continue on and do them as stretch challenges!

The Select Statement

We can use the **SELECT** statement to query our database table for data matching one or more constraints.

```
SELECT * FROM fruits LIMIT 3
```



id	name	price	qty
1	apple	0.99	10
2	banana	1.29	20
3	pear	1.99	5

We can use a **WHERE** clause to specify which items we are searching for.

```
SELECT name, price, qty FROM fruits WHERE price  
< 1.00 LIMIT 3
```



name	price	qty
apple	0.99	10
grape	0.02	50
orange	0.99	20

Break - 10 min

For the next few SQL terms, we'll look at this [Books & Authors Table](#) example.

[Repl.it link](#)

Take a few minutes to read the code and see what you notice about the queries being performed.

We can use **CREATE TABLE** to set up a table:

```
CREATE TABLE Books (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    title VARCHAR(50) NOT NULL  
);
```

We set the `id` field to be the **primary key**.

We can also set a field to be a **foreign key**:

```
CREATE TABLE BooksAuthors (  
    author_id INTEGER NOT NULL,  
    book_id  INTEGER NOT NULL,  
    FOREIGN KEY (author_id) REFERENCES Authors(id),  
    FOREIGN KEY (book_id) REFERENCES Books(id)  
);
```

Once we have a table, we can **insert** data:

```
INSERT INTO Authors  
  (name, country)  
VALUES  
  ('J.D. Salinger', 'USA'),  
  ('F. Scott. Fitzgerald', 'USA'),  
  ('Jane Austen', 'UK'),
```

Table Joins

What is a Table Join?

Let's say we want to query for all **Book** entries that were written by **Author** with **id=1**.

How would we do that? The **Book** table doesn't contain any references to **Author**.

Books	
id	title
1	Catcher in the Rye
2	Nine Stories
3	Franny and Zooey

BooksAuthors	
book_id	author_id
1	1
2	1
3	1

Authors	
id	name
1	J.D. Salinger
2	F. Scott Fitzgerald
3	Jane Austen

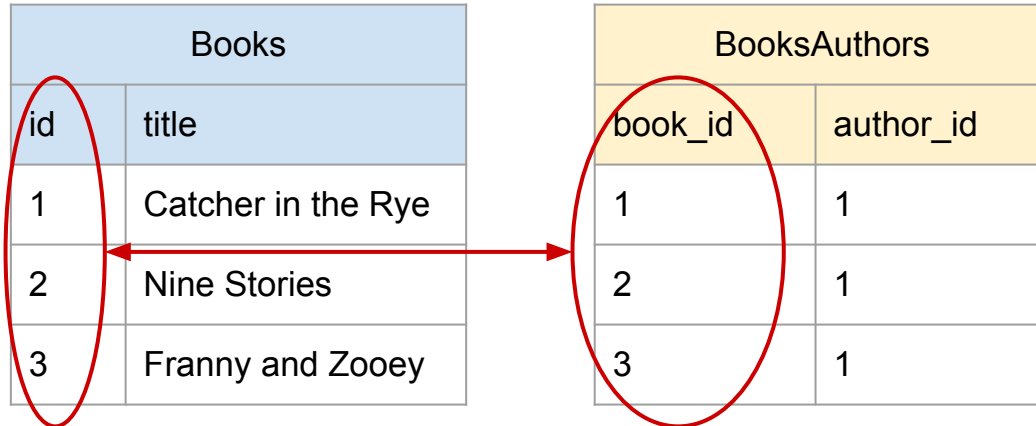
We need to **join** two tables together to get some data from each of the tables.

Our resulting data should look like:

Books + BooksAuthors		
id	title	author_id
1	Catcher in the Rye	1
2	Nine Stories	1
3	Franny and Zooey	1

We can use the **JOIN** keyword to join two tables. We need to tell SQL how to join them - we need to join "on" a certain column.

```
SELECT * FROM Books  
JOIN BooksAuthors ON Books.id = BooksAuthors.book_id  
WHERE BooksAuthors.author_id = 1;
```



Books	
id	title
1	Catcher in the Rye
2	Nine Stories
3	Franny and Zooey

BooksAuthors	
book_id	author_id
1	1
2	1
3	1

Activity (10 minutes)

Using the [example given](#), write a table join to **find all Authors who authored the book with id=7.**

[Repl.It Link](#)

What if we want to get a list of all book titles, along with their author names?

We can't join the **Books** and **Authors** tables together, because they don't reference each other.

We can do a **three-table join** instead, to join both tables to the **BooksAuthors** table:

```
SELECT Books.title, Authors.name FROM BooksAuthors  
JOIN Books ON Books.id = BooksAuthors.book_id  
JOIN Authors ON Authors.id = BooksAuthors.author_id;
```

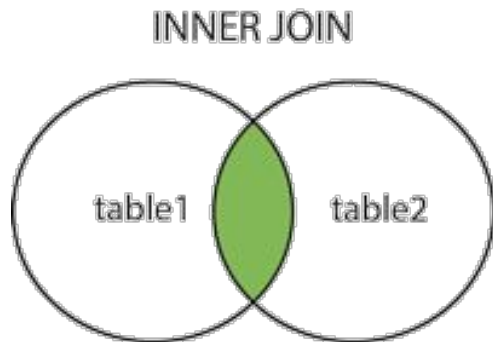
Activity (10 minutes)

Using [this SQL Fiddle](#), write a three-table join to get a list of students alongside the courses they are taking. Use the previous slide as an example.

[Repl.It Link](#)

By default, the **JOIN** keyword performs an "**inner join**" - that is, if we are querying two tables and joining on a certain field, it will ignore any data that isn't in both tables.

Let's see how this works with an example.



Let's say we have the following data:

Users	
id	name
1	Jen Hacker
2	Joe Maker
3	Maya Programmer

BlogPosts	
title	author_id
"Top 7 Programming Languages"	1
"Reasons to Use MySQL"	1
"Migrating Your Database"	2

What happens if we do an inner join on `author_id`?

Maya Programmer gets left out as she hasn't written any blog posts!

An inner join would give us this resulting dataset:

```
SELECT Users.id, Users.name, BlogPosts.title FROM Users  
JOIN BlogPosts ON Users.id = BlogPosts.author_id;
```

Users.id	Users.name	title
1	Jen Hacker	"Top 7 Programming Languages"
1	Jen Hacker	"Reasons to Use MySQL"
2	Joe Maker	"Migrating Your Database"

No Maya Programmer :(

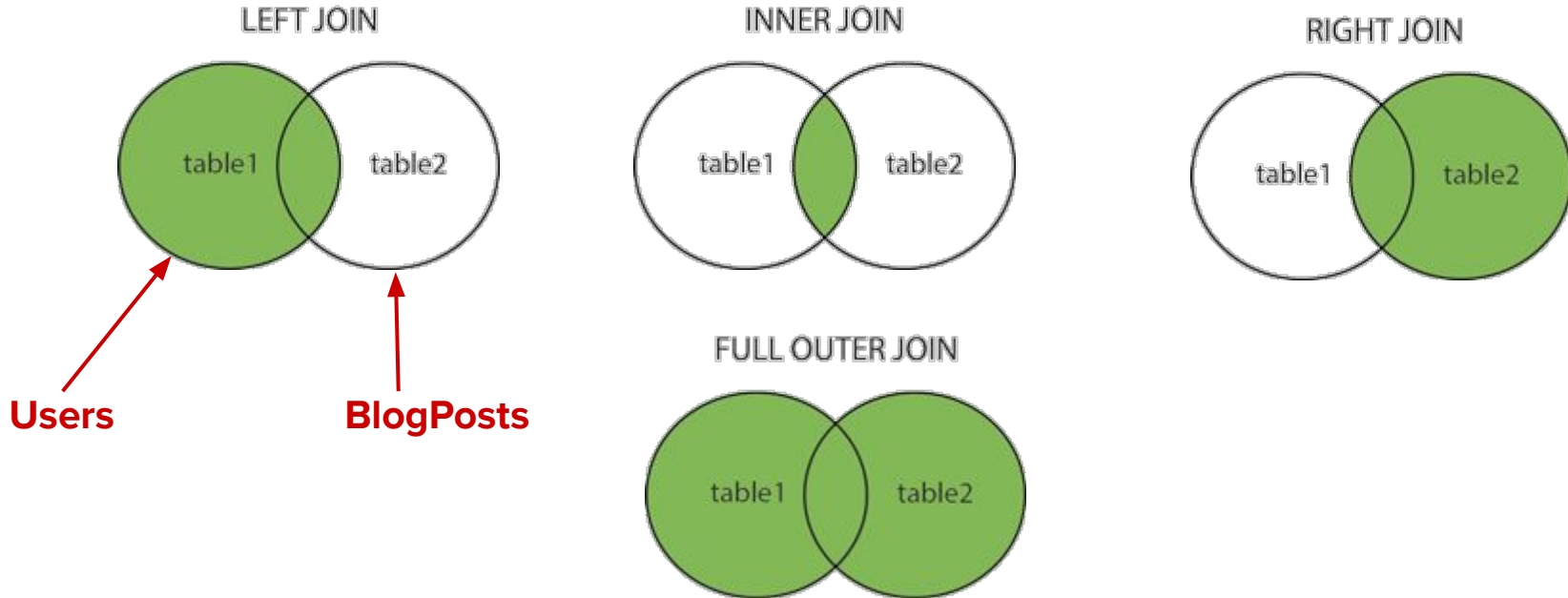
A **left join** would give us this resulting dataset:

```
SELECT Users.id, Users.name, BlogPosts.title FROM Users  
LEFT JOIN BlogPosts ON Users.id = BlogPosts.author_id;
```

Users.id	Users.name	title
1	Jen Hacker	"Top 7 Programming Languages"
1	Jen Hacker	"Reasons to Use MySQL"
2	Joe Maker	"Migrating Your Database"
3	Maya Programmer	(null)

Types of Joins

There are **four types of joins** in all. The left & inner join are most common.



Which type of join would be most appropriate for the following scenarios:

- We want to show a summary of all books with their authors. If a book doesn't have an author (e.g. it was published anonymously), we want to show it as well.

Left join - include empty values

- We want to get a list of all students with the courses they are enrolled in. If a student doesn't have any enrollments, we don't want to include them.

Inner join - don't include empty values

Wrap-Up

Homework

With a partner, start on today's [homework](#).