

## SQL CRUD

### CREATE data

```
-- Columns get explicitly listed, then contents
INSERT INTO users (username, first_name, age)
VALUES ('janeqhacker', 'Jane', 37);
```

### READ data

```
-- Get all rows from the "users" table
SELECT * FROM users;

-- Get rows with username janeqhacker
SELECT * FROM users
WHERE username = 'janeqhacker';

-- Get all rows, but only 2 certain columns
SELECT first_name, last_name FROM users;

-- Sort the result set in descending order
SELECT * FROM users
ORDER BY username DESC;

-- Skip 40 rows and return the next 10 rows
SELECT * FROM users
ORDER BY username
LIMIT 10 OFFSET 40;
```

### UPDATE data

```
UPDATE users SET last_name = 'Quacker'
WHERE username = 'janeqhacker';
```

### DELETE data

```
DELETE FROM users
WHERE username = 'janeqhacker';
```

## SQL TABLE MANAGEMENT

```
CREATE TABLE users (
  id INT NOT NULL AUTO_INCREMENT,
  username VARCHAR(63) NOT NULL,
  email VARCHAR(255) NOT NULL,
  first_name VARCHAR(127),
  last_name VARCHAR(127),
  age INT,
  PRIMARY KEY (id)
);

DROP TABLE users; -- Delete a table

-- Adding or deleting columns
ALTER TABLE users ADD COLUMN
  like_count INT;
ALTER TABLE users DROP COLUMN like_count;
```

## POSTGRES COMMANDS

```
\d          -- List tables
\d users    -- Describe table "users"
\q         -- Quit
```

### ADVANCED

```
-- Index -- permits fast lookup (binary search)
-- when column is in WHERE clause (or ORDER BY)
CREATE INDEX email_index
ON users (email);

-- Joins -- getting data from multiple tables
SELECT username, tweet
FROM users, tweets
WHERE username='janeqhacker'
AND users.id=tweets.user_id;

-- Show execution plan for query (good for checking
-- if indices are working as intended)
EXPLAIN
SELECT * FROM users
WHERE username = 'janeqhacker';
```

### JOINS TERMINOLOGY EXAMPLE

Joins combine data from multiple tables. They are categorized into 4 categories, based on how they do the matching of data between the tables. Examples for fetching data from users (the *left table*) and tweets (the *right table*) as below:

**INNER JOIN** Exclude tweetless users, exclude anonymous tweets.

**LEFT JOIN** Include all users, exclude anonymous tweets

**RIGHT JOIN** Exclude tweetless users, include anonymous tweets

**FULL OUTER JOIN** Include tweetless users, include anonymous tweets

### TERMINOLOGY

**SQL** The programming language used for creating and accessing data in a database

**table** One grouping of data, consists of columns (the schema) and rows (the data itself)

**column** Consists of a name and a type

**row** The actual data in the database is in *rows*

**schema** refers to the shape of your data – that is to say, everything but the rows

**database** Refers to one particular application's data in the your database

**index** Essentially, can be “turned on or off” on a column, makes look-ups which filter by that column faster