★ SQL Video ★

What is a Database?

General Definition

- Any collection of related information
  - Phone Book ✓
  - Shopping list ✓
  - Todo list
  - Your 5 best friends
  - Facebook's User Base
- Databases can be stored in different ways
  - On paper
  - In your mind
  - On a computer
  - This powerpoint
  - Comments Section

Computers + Databases = <3

Amazon.com

- Keeps track of Products, Reviews, Purchase Orders, Credit Cards, Users, Media, etc
- Trillions of pieces of information need to be stored and readily available
- Information is extremely valuable and critical to Amazon.com's functioning
- Security is essential, Amazon stores peoples personal information
  - Credit card #, SSN, Address, phone
- Information is stored on a computer

vs

Shopping List

- Keeps track of consumer products that need to be purchased
- 10-20 pieces of information need to be stored and readily available
- Information is for convenience sake only and not necessary for shopping
- Security is not important
- Information is stored on a piece of paper, or even just in someone's memory

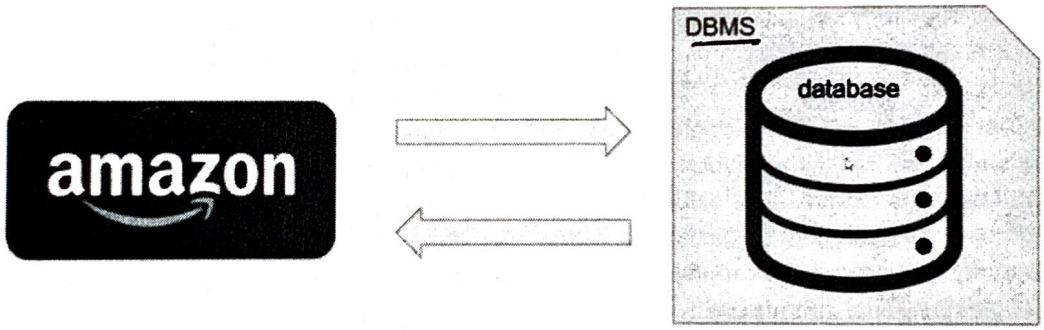
Computers are great at keeping track of large amounts of information!
Database doesn't have to be on a computer

Database Management Systems (DBMS)

- A special software program that helps users create and maintain a database
  - Makes it easy to manage large amounts of information
  - Handles Security ✓
  - Backups ✓
  - Importing/exporting data ✓
  - Concurrency
  - Interacts with software applications ✓
  - Programming Languages ~ like python

Giraffe Academy

Amazon.com Database Diagram



Amazon.com will interact with the DBMS in order to create, read, update and delete information

Giraffe Academy

DBMS isn't the actual database its the software application that is creating, reading, updating, and deleting.

C.R.U.D

4-main option your going to be doing with a database

Create Read Update Delete

-Retrieve

Creating new Database entries
Reading
Updating
Deleting

Two Types of Databases

Relational Databases (SQL)

- Organize data into one or more tables
  - Each table has columns and rows
  - A unique key identifies each row

Non-Relational (noSQL / not just SQL)

- Organize data is anything but a traditional table
  - Key-value stores
  - Documents (JSON, XML, etc)
  - Graphs
  - Flexible Tables

most popular

like a excel spreadsheet?

Relational Databases (SQL)

How you would store data in a relational database.

Student Table

*ID #	Name	Major
1	Jack	Biology
2	Kate	Sociology
3	Claire	English
4	John	Chemistry

Users Table

*Username	Password	Email
jsmith22	wordpass	...
catlover45	apple223	...
gamerkid	...	...
giraffe	...	...

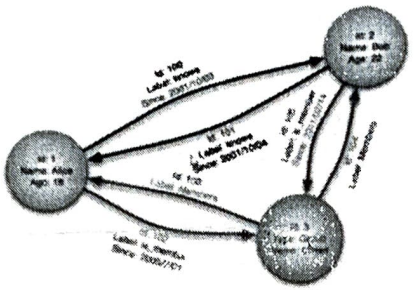
Relational Databases (SQL)

- Relational Database Management Systems (RDBMS)
  - Help users create and maintain a relational database
    - mySQL, Oracle, postgreSQL, mariaDB, etc.
- Structured Query Language (SQL)
  - Standardized language for interacting with RDBMS
  - Used to to perform C.R.U.D operations, as well as other administrative tasks (user management, security, backup, etc).
  - Used to define tables and structures
  - SQL code used on one RDBMS is not always portable to another without modification.

Non-Relational Databases (noSQL / not just SQL)

```
[{
    "_id": 1345,
    "name": "Jack",
    "major": "Biology"
}, {
    "_id": 2267,
    "name": "Kate",
    "major": "Sociology"
}, {
    "_id": 2453,
    "name": "Claire",
    "major": "English"
}, {
    "_id": 1957,
    "name": "John",
    "major": "Chemistry"
}]
```

Document
JSON, BLOB, XML, etc..)



Stores data in non-tables

Graph
Relational nodes

Key	Value
"xyz"	*string*
"abc"	*JSON*
"pqr"	*BLOB*
"lmno"	*etc...*

Key-Value Hash
Keys are mapped to values
(strings, json, blob, etc..)

Giraffe Academy

Non-Relational Databases (noSQL / not just SQL)

- Non-Relational Database Management Systems (NRDBMS)
  - Help users create and maintain a non-relational database
    - mongoDB, dynamoDB, apache cassandra, firebase, etc

- Implementation Specific
  - Any non-relational database falls under this category, so there's no set language standard.
  - Most NRDBMS will implement their own language for performing C.R.U.D and administrative operations on the database.

Database Queries

Queries are requests made to the database management system for specific information

As the database's structure become more and more complex, it becomes more difficult to get the specific pieces of information we want.

Asking for something.

A google search is a query

Wrap Up

- Database is any collection of related information
- Computer are great for storing databases
- Database Management Systems (DBMS) make it easy to create, maintain and secure a database.
- DBMS allow you to perform the C.R.U.D operations and other administrative tasks
- Two types of Databases, Relational & Non-Relational
- Relational databases use SQL and store data in tables with rows and columns
- Non-Relational data store data using other data structures

Tables and Keys

Student

You see how Jack is the same name?

student_id	name	major
1	Jack	Biology
2	Kate	Sociology
3	Claire	English
4	Jack	Biology
5	Mike	Comp. Sci

student id is a Primary Key

- The Primary Key is unique in each row, it's unique & Unique,

---

User

email	password	date_created	Type
fakemail@fake.co	shivers1	1999-05-11	Admin
fakemail112@fake.co	wordpass	2001-03-15	Free
rsmith@fake.co	redRoad23	2010-09-05	Free
jdoe@fake.co	passw0rd	2008-06-25	Premium
jhalpert@fake.co	557df32d	2003-07-22	Free

Key: Primary Key

Attribute

Foreign Key

Employee

emp_id	first_name	last_name	birth_date	sex	salary
100	Jan	Levinson	1961-05-11	F	110,000
101	Michael	Scott	1964-03-15	M	75,000
102	Josh	Porter	1969-09-05	M	78,000
103	Angela	Martin	1971-06-25	F	63,000
104	Andy	Bernard	1973-07-22	M	65,000

Employee id

Surrogate key: Primary Key that has no mapping to the real world

Employee

emp_ssn	first_name	last_name	birth_date	sex	salary
123456789	Jan	Levinson	1961-05-11	F	110,000
555667777	Michael	Scott	1964-03-15	M	75,000
8886665555	Josh	Porter	1969-09-05	M	78,000
111332467	Angela	Martin	1971-06-25	F	63,000
99857463	Andy	Bernard	1973-07-22	M	65,000

Natural Key: Primary Key that has a mapping/purpose in the real world

Employee

emp_id	first_name	last_name	birth_date	sex	salary	branch_id
100	Jan	Levinson	1961-05-11	F	110,000	1
101	Michael	Scott	1964-03-15	M	75,000	2
102	Josh	Porter	1969-09-05	M	78,000	3
103	Angela	Martin	1971-06-25	F	63,000	2
104	Andy	Bernard	1973-07-22	M	65,000	3

Foreign Key: A Key that will link us to another Database table (branch id), stores primary key in another table

- A foreign key stores the primary key of a row of in another database table.

Branch

branch_id	branch_name	mgr_id
2	Scranton	101
3	Stamford	102
1	Corporate	108

- A Foreign Key is a primary key inside another table.

-Jan Levinson branch id is 1 so she is in the Corporate branch

- Foreign Key can find relationships between tables

Another Foreign Key that connects Branch to foreign table the Employee table.

Michael Scott is the manager of the Scranton Branch!

Employee

Foreign Key

emp_id	first_name	last_name	birth_date	sex	salary	branch_id	super_id
100	Jan	Levinson	1961-05-11	F	110,000	1	NULL
101	Michael	Scott	1964-03-15	M	75,000	2	100
102	Josh	Porter	1969-09-05	M	78,000	3	100
103	Angela	Martin	1971-06-25	F	63,000	2	101
104	Andy	Bernard	1973-07-22	M	65,000	3	101

hes a boss !

super_id is going to define who is the supervisor

Angela Martin supervisor id is 101, which means her supervisor is Michael Scott

Employee

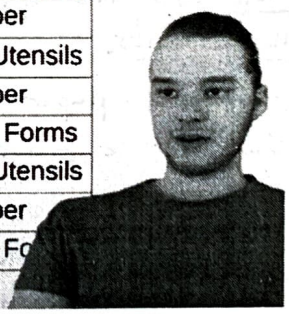
emp_id	first_name	last_name	birth_date	sex	salary	branch_id	super_id
100	Jan	Levinson	1961-05-11	F	110,000	1	NULL
101	Michael	Scott	1964-03-15	M	75,000	2	100
102	Josh	Porter	1969-09-05	M	78,000	3	100
103	Angela	Martin	1971-06-25	F	63,000	2	101
104	Andy	Bernard	1973-07-22	M	65,000	3	101

Branch

branch_id	branch_name	mgr_id
2	Scranton	101
3	Stamford	102
1	Corporate	108

Branch Supplier

branch_id	supplier_name	supply_type
2	Hammer Mill	Paper
2	Uni-ball	Writing Utensils
3	Patriot Paper	Paper
2	J.T. Forms & Labels	Custom Forms
3	Uni-ball	Writing Utensils
3	Hammer Mill	Paper
3	Stamford Lables	Custom Fo



You notice that the primary key consist of two columns (**branch id** and **supplier name**) and this is what we call a **Composite Key**. A **Composite Key** is basically a key that needs two attributes.

The reason we need the **Composite Key** is because the **supplier name** doesn't uniquely identify each row and the **branch id** doesn't uniquely identify each row. Only together can they uniquely identify each row.

needs to uniquely identify !

"Foreign Key"

Employee

emp_id	first_name	last_name	birth_date	sex	salary	branch_id	super_id
100	Jan	Levinson	1961-05-11	F	110,000	1	NULL
101	Michael	Scott	1964-03-15	M	75,000	2	100
102	Josh	Porter	1969-09-05	M	78,000	3	100
103	Angela	Martin	1971-06-25	F	63,000	2	101
104	Andy	Bernard	1973-07-22	M	65,000	3	101

Branch

branch_id	branch_name	mgr_id
2	Scranton	101
3	Stamford	102
1	Corporate	108

Client

client_id	client_name	branch_id
400	Dunmore Highschool	2
401	Lackawana Country	2
402	FedEx	3
403	John Daly Law, LLC	3
404	Scranton Whitepages	2

refer to the Employee in the Employee table

refer to a client in the Client table

Works_With

emp_id	client_id	total_sales
107	400	55,000
101	401	267,000
105	402	22,500
104	403	5,000
105	403	12,000
107	404	33,000

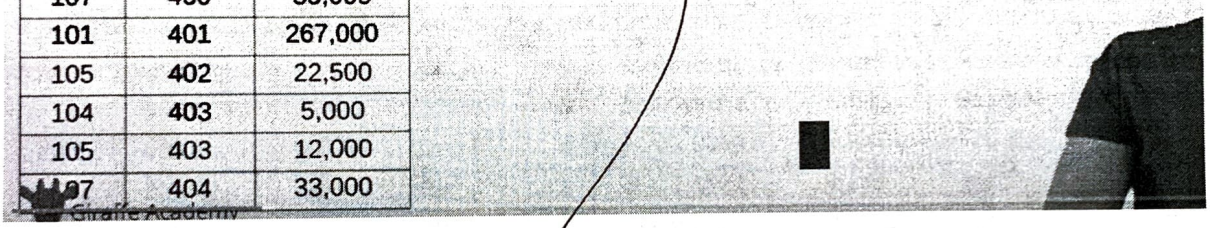


Table is defining the relationship b/c the employees and the clients.

- How much paper an employee sells to a specific client.

Lackawana Country is going to buy products from the Scranton Branch.

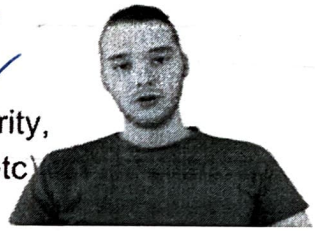
☆ ☆

in the Works_with table

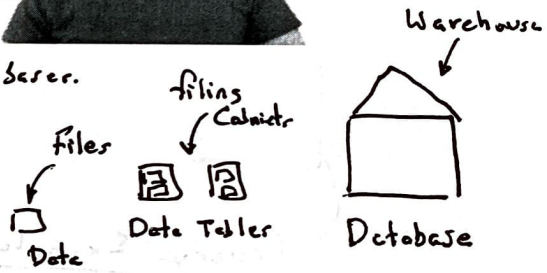
emp_id and client_id are composite Keys, special type of composite Key b/c both of these columns are foriegn Keys, both of the foriegn Keys make up the primary Key of the table.

Structured Query Language (SQL)

- SQL is a language used for interacting with Relational Database Management Systems (RDBMS)
  - You can use SQL to get the RDBMS to do things for you
    - Create, retrieve, update & delete data ✓
    - Create & manage databases ✓
    - Design & create database tables ✓
    - Perform administration tasks (security, user management, import/export, etc)



- language that communicates with databases.



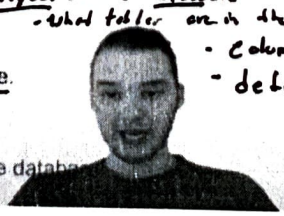
Structured Query Language (SQL)

- SQL implementations vary between systems
  - Not all RDBMS' follow the SQL standard to a 'T'
  - The concepts are the same but the implementation may vary ★

Structured Query Language (SQL)

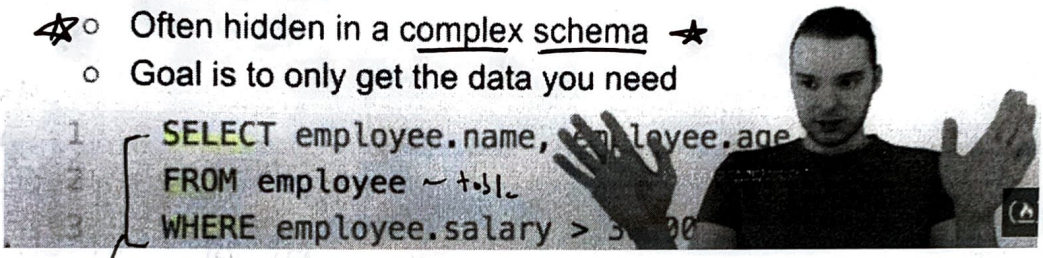
- SQL is actually a hybrid language, it's basically 4 types of languages in one
  - **Data Query Language (DQL)**
    - Used to query the database for information. ~ write Query in SQL to get data
    - Get information that is already stored there
  - **Data Definition Language (DDL)**
    - Used for defining database schemas. ~ "overall layout of the database" - what tables are in the database, - Columns - define database
  - **Data Control Language (DCL)**
    - Used for controlling access to the data in the database.
    - User & permissions management
  - **Data Manipulation Language (DML)**
    - Used for inserting, updating and deleting data from the database

Ask Queries for information



Queries

- A query is a set of instructions given to the RDBMS (written is SQL) that tell the RDBMS what information you want it to retrieve for you
  - TONS of data in a DB
  - Often hidden in a complex schema
  - Goal is to only get the data you need



Query Example!

Tell Query what you want!

Logging into MySQL and creating a Database
Password: Notoriousbig1

Using the terminal ↓ to create a "database"

```
Last login: Sat May 30 12:16:01 on ttys000
(base) Devins-MacBook-Pro-2:~ devinpowers$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database practice;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Create database "insert name of db";

★ name you want to call the database ★

In PopSQL

[My_SQL is a database management system] ★

Manage Connections

New Connection

Giraffe

Nickname *
Practice

Type *
MySQL

Hostname *
localhost

Port *
3306

Database *
SQL_Practice

Username
root

Password
••••••••••••

Make this my default connection

~~SQL Management manages those databases~~

SQL is a language to fetch, update and insert data in the database.

Pop SQL is a modern SQL editor makes it easy to write queries, organize them in folders, visualize your data, and collaborate with your team.

- Pop SQL is going to connect to mySQL database we created!

Creating Tables

Data Types that we will work with, most common, there is a few others:

```
INT                     -- Whole Numbers
DECIMAL(M,N)            -- Decimal Numbers - Exact Value
VARCHAR(l)              -- String of text of length l
BLOB                    -- Binary Large Object, Stores large data
DATE                    -- 'YYYY-MM-DD'
TIMESTAMP               -- 'YYYY-MM-DD HH:MM:SS' - used for recording events
```

```
-- Creating tables
CREATE TABLE student (
  student_id INT PRIMARY KEY,
  name VARCHAR(40),
  major VARCHAR(40)
  -- PRIMARY KEY(student_id)
);

DESCRIBE student;
DROP TABLE student;
ALTER TABLE student ADD gpa DECIMAL;
ALTER TABLE student DROP COLUMN gpa;
```

name of table

letting table know that this is the primary key

How to create a table

need to give data type

length, # of characters you want to store

Close with semi-colon ; indicates end of query or just a delimiter

or primary key like this!

display table

drop table

drop column from table!

add column to table

This is going to create this table setup.

- We can add student name and major later!

Notes:

Primary Key: Uniquely identify each row

student_id	name	major
1	Jack	Biology
2	Kate	Sociology
3	Claire	English
4	Jack	Biology
5	Mike	Comp. Sci

When we Run 'DESCRIBE student'

Success
3 rows

Explore | SQL | Data | Chart | Export

Field	Type	Null	Key	Default	Extra
student_id	int	NO	PRI	NULL	
name	varchar(40)	YES		NULL	
major	varchar(40)	YES		NULL	

Drop Table

If we highlight and run DROP TABLE student, it will drop the table. When we try to run DESCRIBE student this is returned:

Failed

Explore | SQL | Data | Chart | Export

```
ER_NO_SUCH_TABLE: Table 'sql_practice.student' doesn't exist
```

Modify the Table

```
ALTER TABLE student ADD gpa DECIMAL;
```

Then once we run DESCRIBE student again,

Success
4 rows

Explore | SQL | Data | Chart | Export

Field	Type	Null	Key	Default	Extra
student_id	int	NO	PRI	NULL	
name	varchar(40)	YES		NULL	
major	varchar(40)	YES		NULL	
gpa	decimal(10,0)	YES		NULL	

Add spc

The column for gpa was added

We can also drop a specific column:

```
ALTER TABLE student DROP COLUMN gpa;
```

Then once we run DESCRIBE student again,

Success
3 rows

Explore SQL Data Chart Export

Field	Type	Null	Key	Default	Extra
student_id	int	NO	PRI	NULL	
name	varchar(40)	YES		NULL	
major	varchar(40)	YES		NULL	

The column for gpa was removed

Inserting Data

name of table that we want to insert into!

To insert data into table

```
INSERT INTO student VALUES(1, 'Jack', 'Biology');
INSERT INTO student VALUES(2, 'Kate', 'Sociology');
INSERT INTO student(student_id, name) VALUES(3, 'Claire');
INSERT INTO student VALUES(4, 'Jack', 'Biology');
INSERT INTO student VALUES(5, 'Mike', 'Computer Science');
```

When you highlight a row:

```
INSERT INTO student VALUES(1, 'Jack', 'Biology');
```

Then run it:

student_id (primary key), name, major

Success | Explore | SQL | Data | Chart | Export | %

Rows affected: 1

- We can see how this

Highlight this:

This isn't always necessary

```
SELECT * FROM student;
```

~ Graph all info from table

student_id	name	major
1	Jack	Biology

- We can see this table is made

What if we didn't know the Students major?

inserting what we know and the values we know!

Table

```
INSERT INTO student(student_id, name) VALUES(3, 'Claire');
```

student_id	name	major
1	Jack	Biology
2	Kate	Sociology
3	Claire	NULL

- When you don't enter something in, NULL is placed in place
- You cant enter duplicate entry: e.g. same primary key

```
INSERT INTO student(student_id, name) VALUES(3, 'Claire');
```

Then this would appear once you ran:

Failed

Explore | SQL | Data | Chart | Export

```
ER_DUP_ENTRY: Duplicate entry '3' for key 'student.PRIMARY'
```

Constraints

```
INSERT INTO student VALUES(1, 'Jack', 'Biology');

INSERT INTO student VALUES(2, 'Kate', 'Sociology');

INSERT INTO student(student_id, name) VALUES(3, 'Claire');

INSERT INTO student VALUES(4, 'Jack', 'Biology');

INSERT INTO student VALUES(5, 'Mike', 'Computer Science');
```

NOT ~~B~~ NULL
UNIQUE

student_id	name	major
1	Jack	Biology
2	Kate	Sociology
3	Claire	NULL
4	Jack	Biology
5	Mike	Computer Science

NOT NULL and UNIQUE

- Not Null means that the name can't be empty
- Unique means each word must be unique in the column
- ★ Control the data that gets put on the table ★

```
CREATE TABLE student (
  student_id INT ,
  name VARCHAR(20) NOT NULL,
  major VARCHAR(20) UNIQUE,
  PRIMARY KEY(student_id)
);
```

name can't be empty

each word must be unique.

```
INSERT INTO student VALUES(3, NULL, 'Chemistry')
```

When you run it:

Failed | Explain | SQL | Edit | Chart | Export | 2:57 PM

```
ER_BAD_NULL_ERROR: Column 'name' cannot be null
```

```
INSERT INTO student VALUES(1, 'Jack', 'Biology');
INSERT INTO student VALUES(2, 'Kate', 'Sociology');
INSERT INTO student VALUES(3, NULL, 'Chemistry')
INSERT INTO student VALUES(4, 'Jack', 'Biology');
```

Failed | Explore | SQL | Data | Chart | Export

```
ER_DUP_ENTRY: Duplicate entry 'Biology' for key 'student.major'
```

Default Values:

★ DEFAULT ★

```
CREATE TABLE student (
  student_id INT ,
  name VARCHAR(20) ,
  major VARCHAR(20) DEFAULT 'undecided',
  PRIMARY KEY(student_id)
);
```

if we don't enter in a major it will default the value to 'undecided'

```
INSERT INTO student(student_id, name) VALUES(1,'Jack');
```

student_id	name	major
1	Jack	undecided

↑ default

Auto Increment the Primary Key:

- no need to add student_id anymore, it automatically increase it.

```
CREATE TABLE student (
  student_id INT AUTO_INCREMENT ,
  name VARCHAR(20) ,
  major VARCHAR(20) DEFAULT 'undecided',
  PRIMARY KEY(student_id)
);
```

```
INSERT INTO student(name, major) VALUES('Jack', 'Biology');

INSERT INTO student(name, major) VALUES('Kate', 'Sociology');
```

student_id	name	major
2	Kate	Sociology
1	Jack	Biology

Update & Delete

```
CREATE TABLE student (
  student_id INT,
  name VARCHAR(20) ,
  major VARCHAR(20) ,
  PRIMARY KEY(student_id)
);
INSERT INTO student VALUES(1,'Jack', 'Biology');
INSERT INTO student VALUES(2, 'Kate', 'Sociology');
INSERT INTO student VALUES(3, 'Claire', 'Chemistry');
INSERT INTO student VALUES(4, 'Jack', 'Biology');
INSERT INTO student VALUES(5, 'Mike', 'Computer Science');
```

student_id	name	major
1	Jack	Biology
2	Kate	Sociology
3	Claire	Chemistry
4	Jack	Biology
5	Mike	Computer Science

SELECT * FROM student;

highlight and run anytime you want to display the table.

Let's say we want to update Biology to Bio for all students with that major:

Write this querying

```
UPDATE student
SET major = 'Bio'
WHERE major = 'Biology';
```

student_id	name	major
1	Jack	Bio
2	Kate	Sociology
3	Claire	Chemistry
4	Jack	Bio
5	Mike	Computer Science

Another Example:

Table

Change major

```
UPDATE student
SET major = 'Comp Sci'
WHERE major = 'Computer Science';
```

student_id	name	major
1	Jack	Bio
2	Kate	Sociology
3	Claire	Chemistry
4	Jack	Bio
5	Mike	Comp Sci

Another Example:

```
UPDATE student
SET major = 'Comp Sci'
WHERE student_id =4;
```

student_id	name	major
1	Jack	Bio
2	Kate	Sociology
3	Claire	Chemistry
4	Jack	Comp Sci
5	Mike	Comp Sci

Another Example (using 'or'):

```
UPDATE student
SET major = 'Biochemistry'
WHERE major = 'Bio' or major = 'Chemistry';
```

student_id	name	major
1	Jack	Biochemistry
2	Kate	Sociology
3	Claire	Biochemistry
4	Jack	Comp Sci
5	Mike	Comp Sci

Another Example (change multiple columns):

```
UPDATE student
SET name = 'Tom', major = 'undecided'
WHERE student_id = 1;
```

student_id	name	major
1	Tom	undecided
2	Kate	Sociology
3	Claire	Biochemistry
4	Jack	Comp Sci
5	Mike	Comp Sci.

Another example (the where is optional)

```
UPDATE student
SET Major = 'undecided';
```

student_id	name	major
1	Tom	undecided
2	Kate	undecided
3	Claire	undecided
4	Jack	undecided
5	Mike	undecided

Delete Rows

name of table

```
DELETE FROM student
WHERE student_id = 5;
```

student_id	name	major
1	Tom	undecided
2	Kate	undecided
3	Claire	undecided
4	Jack	undecided

Another Example:

```
DELETE FROM student
WHERE name = 'Tom' AND major = 'undecided';
```

using "and"

student_id	name	major
2	Kate	undecided
3	Claire	undecided
4	Jack	undecided

Another Example (we can delete all the rows):

```
DELETE FROM student;
```

Success

Explore | SQL | Data | Chart | Export

No results found

Basics Queries

~ Writing Instructions to retrieve what we want!

```
SELECT *
FROM student LIMIT 100;
```

Example:

name of column

```
SELECT name
FROM student LIMIT 100;
```

~ semi-colon

table we want info from

Note "Limit" is written since my mac sucks

name
Jack
Kate
Claire
Jack
Mike

Example:

```
SELECT name, major
FROM student LIMIT 100;
```

~ name and major

name	major
Jack	Biology
Kate	Sociology
Claire	Chemistry
Jack	Biology
Mike	Computer Science

Example:

"pre-pennel"

complex

```
SELECT student.name, student.major
FROM student LIMIT 100;
```

name	major
Jack	Biology
Kate	Sociology
Claire	Chemistry
Jack	Biology
Mike	Computer Science

student.name

↑ ↑ ↑ Column we want returned

default period

name of table were looking at

Example (Alphabetical order)

```
SELECT student.name, student.major
FROM student
ORDER BY name
LIMIT 100;
```

ORDER BY

name	major
Claire	Chemistry
Jack	Biology
Jack	Biology
Kate	Sociology
Mike	Computer Science

Example (Descending order)

```
SELECT student.name, student.major
FROM student
ORDER BY name DESC
LIMIT 100;
```

name	major
Mike	Computer Science
Kate	Sociology
Jack	Biology
Jack	Biology
Claire	Chemistry

Example

```
SELECT *
FROM student
ORDER BY student_id DESC
LIMIT 100;
```

student_id	name	major
5	Mike	Computer Science
4	Jack	Biology
3	Claire	Chemistry
2	Kate	Sociology
1	Jack	Biology

Example:

```
SELECT *
FROM student
ORDER BY major, student_id
LIMIT 100;
```

order by major, then student_id alphabetically!

student_id	name	major
1	Jack	Biology
4	Jack	Biology
3	Claire	Chemistry
5	Mike	Computer Science
2	Kate	Sociology

Filtering

Example:

```
SELECT *
FROM student
WHERE major = 'Biology'
```

student_id	na...	major
1	Jack	Biology
4	Jack	Biology

returned student_id and name of those with major = Biology

Example:

```
SELECT *
FROM student
WHERE major = 'Biology' AND student_id > 1;
```

student_id	na...	major
4	Jack	Biology

★ comparison operators <, >, <=, >=, =, <>, AND, OR ★

-- <- comment in SQL

IN Keyword

```
SELECT *
FROM student
WHERE name IN ('Claire', 'Kate', 'Mike');
```

student_id	name	major
2	Kate	Sociology
3	Claire	Chemistry
5	Mike	Computer Science

Example:

```
SELECT *
FROM student
WHERE major IN ('Biology', 'Chemistry');
```

student_id	name	major
1	Jack	Biology
3	Claire	Chemistry
4	Jack	Biology

~ found all student_id, name when their major was biology or chemistry

Example:

```
SELECT *
FROM student
WHERE major IN ('Biology', 'Chemistry') AND student_id > 2
LIMIT 100;
```

student_id	name	major
3	Claire	Chemistry
4	Jack	Biology

~ Using both IN and AND and > operator