**SQL**

1. Intro
   1. Structured Query Language
   2. SQL and noSQL databases and key:value pairs
   3. AirTables, navacat, and other tools are GUI for databases (and importing into csv)
2. After setting up a rails app...
   1. Terminal: rails dbconsole
      1. This gets you into the database console
   2. Configure the database
      1. Terminal: .mode column
      2. Terminal: .headers on
   3. Create the table
      1. Note, each line is it’s own line (return)
      2. Terminal: create table people (

id integer primary key,

name varchar(64) NOT NULL,

age integer NOT NULL,

mood varchar(64));

* 1. Insert data into the table...
     1. Terminal: insert into people (name, age, mood)

values (“andy”, 36, “happy”);

* 1. View the data:
     1. Terminal: select \* from people;
        1. This means “give me all of the information from all people”
     2. Terminal: select name from people

order by name asc;

* + - 1. Displays only names from people, sorts alphabetically, but not smart with caps...
    1. Terminal: select lower(name) from people

order by lower(name) asc;

* + - 1. Displays only names of people, all lowercase & alphabetically
    1. Terminal: select name, age from people

where age > 25;

* + - 1. Displays name and age of all entries who are 26 years old (or older)
    1. Terminal: select count(mood), mood, age from people

where mood=”happy”

group by age

order by count(mood) desc;

* + - 1. Displays a count, mood, and age of people; who have a mood “happy”, grouped by age, ordered descending based on the count
  1. Modify the Data:
     1. Rows, via Terminal: update people

set mood=”hungry”

where name=”andy”;

* + 1. Delete Rows: delete from people

where name=”andy”

* + 1. Prevent duplicate entries from having the same name (terminal):
       1. create unique index name\_the\_index\_something on people(name);
    2. Drop the table: drop table people;

1. Object Relational Mapper (ORM)
   1. Intro
      1. ORM’s help protect your database vs SQL injection
      2. Does the SQL for you...
      3. Database tables are also called “relations”
      4. ActiveRecord Query Interface Guide: <http://guides.rubyonrails.org/active_record_querying.html>
      5. ActiveRecord Validation Guide: http://guides.rubyonrails.org/active\_record\_validations.html
   2. Note: reload!
      1. Use ‘reload!’ in console if you change the model and need to see the new data
   3. After setting up the rails app...
      1. Build the model (of the database)
         1. Terminal: rails g model Person name:string:uniq age:integer mood:string
            1. Basically, generate a model named person, entries will have a name (unique string), age (integer), and mood (string)
            2. Model Names are singular and uppercase, the table created is plural and lowercase
      2. Go to db => migrate => your file
         1. This is the skeleton of the db you will create when you run it
         2. You can also edit this file
            1. add ‘, null: false’ after t.string :name and t.integer :age, add ‘, default: “meh”’ after t.string :mood
            2. This makes :name and :age required, and “meh” the default input for :mood if it’s not given
      3. Launch that shit
         1. Terminal: rails db:migrate
      4. Add a Gem - “hirb”
         1. In gemfile, insert a line: gem ‘hirb’
         2. Run bundle
      5. Enable “hirb”
         1. Terminal: rails c
         2. Terminal: Hirb.enable
      6. Add data
         1. Now you can write ruby code (with ORM) that turns it into SQL
            1. Ex: Person.create(name:”some guy”, mood:”bored”, age:22)
            2. You will get a commit transaction with a pretty table if it works...
      7. View data
         1. Terminal: Person.all
         2. Or: Person.order(name: :asc).select(:name)
         3. Or: Person.where(“age > 25”).select(:name, :age)
         4. Or: Person.where(mood: “happy”).group(:age).count
      8. Modify data
         1. Modify: Person.where(name: "andy").update\_all(mood: "hungry")
         2. Delete: Person.where(name: "andy").delete\_all
      9. Model Validation - business logic for your table, when violated display messages for users
         1. Ex inside app/models/person.rb:

class Person < ApplicationRecord

validates :name, presence: true, uniqueness: true

validates :age, :mood, presence: true

end

* + - 1. Try these:

Person.create(mood: "anonymous")

Person.create.errors.full\_messages