For this challenge you will base your code on the ActiveRecord code we've been working on together. Within the challenge I may refer to ActiveRecord as AR.

Your code should be in a folder structure as follows:

- db (folder)
  - development.sqlite3
- models (folder)
  - One ruby file per ActiveRecord class.
- ar.rb (The ActiveRecord connection file we looked through in class.)
- challenge_create.rb
- challenge_read.rb
- challenge_update.rb
- challenge_destroy.rb
- challenge_faker.rb
- challenge_faker_read.rb

Starter files can be cloned to your VM using:

`git clone https://github.com/StungEye-RRC/Active-Record-Challenge.git`

You will need to create some of the challenge_*.rb files.

Helpful Resources:

- Active Record Without Rails (Source code from the ActiveRecord lecture)
- Active Record Query Interface
- Active Record Associations
- Active Record Validations and Callbacks

The Challenge (should you choose to accept it.)

Part 1 - Deep in the CRUD

- **IMPORTANT:** Start by creating a back-up of the db/development.sqlite3 file so that you can restore the db anytime you want by replacing the .sqlite3 file with the backup.

- In the challenge_read.rb file:
  - Use the Product class to find (any) product from the database.
  - Inspect the Product object you have retrieved.
    - In a comment within your product.rb file, record all the columns that exist in the products table.
  - Based on the columns you find, can you determine if the products table has an association with any other tables? If so, what table?
  - Use ActiveRecord to find and print out:
    - Total number of products
    - The names of all products above $10 with names that begin with the letter C.
    - Total number of products with a low stock quantity. (Low is defined as less than 5 in stock.)

- Modify the Product model class:
  - Add a validation to this class such that a product must have all columns (other than foreign keys, the id, or datetime column) filled out before it can be saved to the db.
  - Research AR validations and add a validation that will ensure that all product names are unique and longer than 3 characters.

- In the challenge_create.rb file:
  - Create three new products using the three different ways to create new AR objects.

    - Ensure that all three new products are persisted to the database, without validations errors.
    - Create a Product object that is missing some required columns.
    - Attempt to save this product and print all the AR errors which are generated.

- In the challenge_update.rb file:
    - Find all products with a stock quantity greater than 40.
    - Add one to the stock quantity of each of these products and then save these changes to the database.

- In the challenge_delete.rb file:
    - Find one of the products you created in challenge_create.rb and delete it from the database.

As you discovered in the first part of this challenge, the products table has a one to many association with another table (categories):

- Add to the challenge_read.rb file:
    - Find the name of the category associated with one of the products you have found. (You should do this without referencing the products foreign key value. Use the product's "belongs to" association instead.)
    - Find a specific category and use it to build and persist a new product associated with this category. (You should do this without manually setting the products foreign key. Look at the end of this example file to see how to build an ActiveRecord object by way of an "has many" association.)
    - Find a specific category and then use it to locate all the the associated products over a certain price.

Part 2 - Faker Faker

- Add the "faker" gem to your project Gemfile like this:

```
gem 'faker', :git => 'https://github.com/stympy/faker.git', :branch => 'main'
```

- From the command prompt in your challenge folder you now need to run:

```
bundle install
```

- Add the following to line 7 of 'ar.rb':

```
require 'faker'
```

- Look through the Faker Docs to see what this gem can do. Be sure to read through the "usage" example.

- In the challenge_faker.rb file:
  - In a loop user Faker to generate 10 new categories.
  - Within this same loop use the newly created and saved category objects to generate 10 new products for each category. The name, description, price and quantity of these 10 products should also be generated using faker.

- In the challenge_faker_read.rb file:
  - Find all the categories in the database (including those that you added using Faker).
  - Display all category names and their associated products (name and price) in a nicely formatted list.