\*All queries are done from the rails console, may differ slightly when done in rails

\*Can test out all of these queries in the blogs2 project in Rails\_models, it uses the same models

**QUERIES FROM CONSOLE**

**Models worked with in the examples**

* User - first\_name, last\_name, email

Can own several blogs, and write several posts and messages

* Blog - name, description

Owned by one or more users, and can have several posts and messages

* Owner - user\_id, blog\_id

User and Blog have a many to many relationship, so the Owner is the intermediary table

* Post - title, content, user\_id, blog\_id

Post is owned by one user and one blog. Can have many messages

* Message - message, post\_id, user\_id

Messages are owned by one user and own post (and through that post, a blog)

**Personal Notes**

* capitalization matters, all the models are singular by themselves, meaning User.first works, but user.first would error out, this changes when it’s not first, User.first.blogs is correct
* when you retrieve a single row (ex: User.first), it’s an instance, but if you get more than one back as a result, it is delivered as an array of instances/objects (array of objects in rails at least)
* Pay attention to when you write them as plural or singular. You always write singular when it’s a query for a single model (ex: User.first, Blog.all). It gets more complicated when you query one table through another. It’s easiest to think about how you write their model relations in the models folder inside the app folder of your project. So, if I want to find what blogs are owned by the first user, I would do this: User.first.blogs the blogs is plural because one user can own multiple blogs (has\_many in the model file). Even if that user only has one blog, you still write it plural. If I were to write User.first.blog it would error out. Similarly, if I wanted to see all the posts for the first user, since one user can have multiple posts, I would type: User.first.posts. However, if I wanted to see which user owned the first post, I would type Post.first.user, this is because each post is only owned by one user (belongs\_to on the Post model file). Think about whether you wrote has\_many or belongs\_to when you’re thinking about whether it needs to be plural or singular

**Basic Queries**

* **User.first** - finds the first user
* **User.last** - finds the last user
* **User.find(1)** - finds the user with an id of 1. will error out in the console if there is no user with an id of 1
* **test = User.all -** creates a variable called test and puts all of the users into it
* **User.where(:first\_name => ‘Bradly’) -** Finds any user with the first name of ‘Bradly’
* **User.where( [1,3]) -** finds any users within the array given, so any user between and including the id of one and three

**Creating**

* **User.create(:first\_name => 'Jerrod', :last\_name => 'Quintana', :email => 'j@j.com') -** creates a new user, the format is slightly different in the rails project, it’s formatted as an associative array, basically the :first\_name would be changed to “first\_name” =>…. ex: {"name"=>"Lightbulb", "description"=>"used for lighting", "pricing"=>"5.00"}
* **Post.create(user: User.first, blog: Blog.find(2), title: ‘title’, content: ‘content’) -** creating a new user, But since it is owned by a user and blog, you need to include those, but never reference the actual id or foreign key. For example: (user: User.first) is good, it puts the entire instance there, but (user\_id: 1) is bad. Also, notice that you don’t need to put the colon : mark before each column, like I did in the User.create, you can do it two ways. Way one: (:first\_name => ‘Jerrod’) or (first\_name: ‘Jerrod’). Either a colon first, followed by =>, or just a colon inbetween the column name and value.
* **Owner.create(user: User.first, blog: Blog.last) -** creating a row in the owners table, since it is just an intermediary table between users and blogs due to them having a many to many relationship, all I need to do is give it a user\_id and blog\_id. this is so that I can make the first user be one of the owners of the last blog. Another way of doing this is shown below
* **User.first.owners.create(blog: Blog.last) -** this is another way of doing the query above, making the first user be one of the owners of the last blog. Except this way, I first query owner rows that are related to the first user, or owner rows that have that the user\_id of the first User. Then I create a new one, which automatically puts the first user in the user\_id column, so all I have to do is specify the blog. Make sure to make ‘owners’ plural, since one user can have many blogs through owners (thus specified in the model file in the rails project).

**Updating**

**Relational Queries - Queries Through Multiple Models That Are Connected**

**Deleting/Destroying**

**Creating Multiple rows at the same time with the Faker Gem**