Cheatography

Rails 5 ActiveRecord Queries Cheat Sheet

by dwapi via cheatography.com/43665/cs/12982/

Active Record Queries			
Find:	clients = Client.find([1, 10])	SELECT * FROM clients WHERE (clients.id IN (1,10))	
Find By:	Client.find_by! first_name: 'Nobody'	SELECT * FROM clients WHERE (clients.first_name = 'Nobody') LIMIT 1	find_by! will raise an error if no record is fo # => ActiveRecord::RecordNotFound
Passing Params	Client.where("orders_count = ? AND locked = ?", params[:orders], params[:locked])		
Passing Params as Hash	Client.where("created_at >= :start_date AND created_at <= :end_date", {start_date: params[:start_date], end_date: params[:end_date]})}		
Between	Client.where(created_at: (Time.now.midnight - 1.day)..Time.now.midnight)	SELECT FROM clients WHERE(clients.created_at BETWEEN '2008-12-21 00:00:00' AND '2008-12-22 00:00:00')	
Subsets: Find using SQL **IN**	Client.where(orders_count: [1,3,5])	SELECT * FROM clients WHERE (clients.orders_count IN (1,3,5))	
Not:	Client.where.not(locked: true)	SELECT * FROM clients WHERE (clients.locked != 1)	
Distinct:	Client.select(:name).distinct	SELECT DISTINCT name FROM clients	
Limit	Client.limit(5)	SELECT * FROM clients LIMIT 5	
Take:	client = Client.take(2)	SELECT * FROM clients LIMIT 2	Returns record without any implicit orderin Returns nil if no record is found.
Offset:	Client.limit(5).offset(30)	SELECT * FROM clients LIMIT 5 OFFSET 30	

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Active Record Queries (cont)		
Find or Create a New Object	Client.find_or_create_by!(first_name: 'Andy')	SELECT * FROM clients WHERE(clients.first_name = 'Andy') LIMIT 1 BEGIN INSERT INTO clients(created_at, first_name, update-d_at) VALUES ('2011-08-30 05:22:57', 'Andy', '2011-08-30 05:22:57') COMMIT
Find or Initialize a New Object	nick = Client.find_or_initialize_by(first_name: 'Nick')	
Find by SQL	Client.find_by_sql("SELECT * FROM clients INNER JOIN orders ON clients.id = orders.client_id ORDER BY clients.created_at desc")	
Exists?	Client.exists?(1)	
	Client.exists?(name: ['John', 'Sergei']	
	Client.where(first_name: 'Ryan').exists?	
Count	Client.count	SELECT count(*) AS count_all FROM clients
Average	Client.average("orders_count")	
Minimum and Maximum	Client.minimum("age")	minimum/maximum value of a field
Sum	Client.sum("orders_count")	sum of a field
Ordering Results	Client.order(:orders_count, created_at: :desc)	
	Client.order("orders_count ASC, created_at DESC")	

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Active Record Queries (cont)

Chaining ORDER BY	Client.order("orders_count ASC").order("created_at DESC")	SELECT * FROM clients ORDER BY orders_count ASC, created_at DESC

Joining Tables

 # Join Through Defined Associations (Inner Join):

Category.joins(:articles)

Article.joins(:category, :comments)

```
SELECT articles.* FROM articles   INNER JOIN categories ON articl es.c at ego r
 ts.a rt icle_id = articl es.id
```

 # Outer Joins:

Author.left_outer_joins(:posts).distinct.select('authors.*, COUNT(posts.*) AS posts_count').group('authors.id')

```
SELECT DISTINCT authors.*, COUNT( posts.*) AS posts_ count FROM " aut hor s" LEFT OUTER JOIN posts ON posts
```

 # N + 1 queries problem: Always use .includes()

Article.includes(:category, :comments).where(comments: { visible: true })

```
Includes will decide between INNER JOIN eager_load (LOJ) or  Seperate Queries
```

 # Join Using Raw SQL:

Author.joins("INNER JOIN posts ON posts.author_id = authors.id AND posts.published = 't'")

```
SELECT authors.* FROM authors INNER JOIN posts ON posts.a ut hor_id = authors.id AND posts.p ub lished =
```

 # Retrieving filtered data from multiple tables: If you want to call order multiple times, subsequent orders will be appended to the first.

Person .select('people.id, people.name, comments.text') .joins(:comments) .where('comments.created_at > ?', 1.week.ago)

```
SELECT people.id, people.name, commen ts.text FROM people INNER JOIN comments   ON commen ts.p er son_id
01'
```

 # Retrieving specific data from multiple tables:

Person .select('people.id, people.name, companies.name') .joins(:company) .find_by('people.name' => 'John') # this should be the last

```
SELECT people.id, people.name, compan ies.name FROM people INNER JOIN companies   ON compan ies.pe rson_i
```

Group By and Having

 # Group By: Find a collection of the dates on which orders were created.

Order.select("date(created_at) as ordered_date, sum(price) as total_price").group("date(created_at)")

```
SELECT date(c rea ted_at) as ordere d_date, sum(price) as total_ price FROM orders GROUP BY date(c rea te
```

 # Total of grouped items: To get the total of grouped items on a single query, call count after the group.

Order.group(:status).count

```
SELECT COUNT (*) AS count_all, status AS status FROM " ord ers " GROUP BY status)
```

=> { 'awaiting_approval' => 7, 'paid' => 12 }

 # Having: SQL uses the HAVING clause to specify conditions on the GROUP BY fields. You can add the HAVING clause to the SQL fired by the M
method to the find.

Order.select("date(created_at) as ordered_date, sum(price) as total_price"). group("date(created_at)").having("sum(price) > ?", 100)

```
SELECT date(c rea ted_at) as ordere d_date, sum(price) as total_ price FROM orders GROUP BY date(c rea te
 100
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Pluck

Client.where(active: true).pluck(:id)	SELECT id FROM clients WHERE active = 1	# => [1, 2, 3]
Client.distinct.pluck(:role)	SELECT DISTINCT role FROM clients	# => ['admin', 'member', 'guest']
Client.pluck(:id, :name)	SELECT clients.id, clients.name FROM clients	# => [[1, 'David'], [2, 'Jeremy'], [3, 'Jose']]

pluck can be used to query single or multiple columns from the underlying table of a model.

pluck makes it possible to replace code like Client.select(:id).map { |c| c.id }

Unlike select, pluck directly converts a database result into a Ruby Array, without constructing ActiveRecord objects

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com