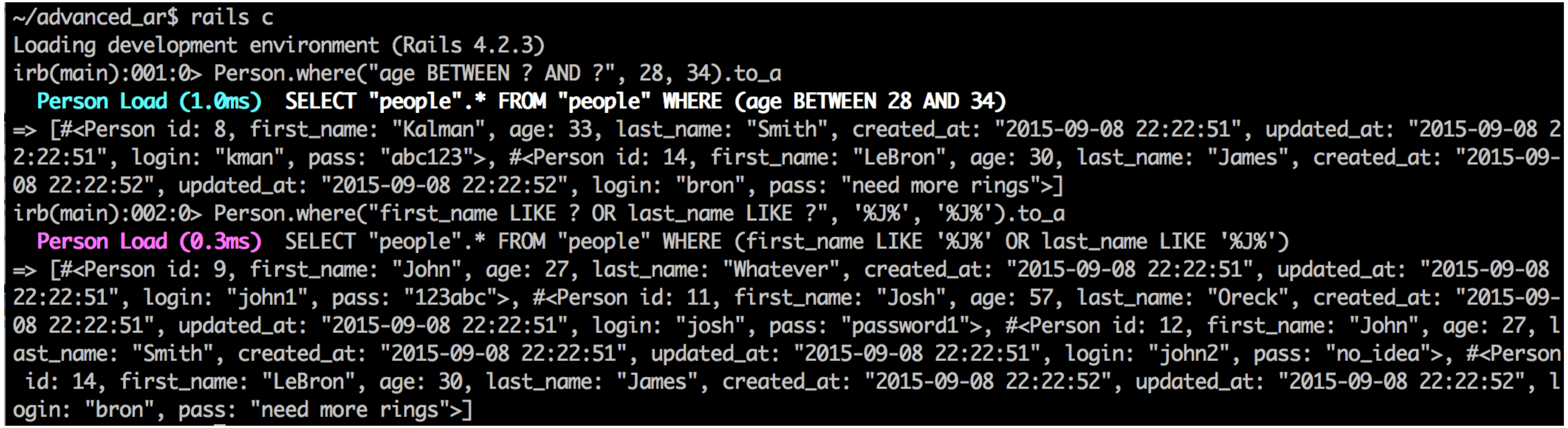
In this lecture, we will discuss…

- ✧ Two alternatives to directly specifying SQL literals
  - Array Condition Syntax
  - Hash Condition Syntax

Array Syntax

- Lets you specify SQL fragment with `?` followed by values (parameters)
- "Automagically" performs conversions on the input values and escapes strings in the SQL
- Immune to SQL injection
- Similar to a PreparedStatement in Java

Array Condition Syntax



```
~/advanced_ar$ rails c
Loading development environment (Rails 4.2.3)
irb(main):001:0> Person.where("age BETWEEN ? AND ?", 28, 34).to_a
  Person Load (1.0ms)  SELECT "people".* FROM "people" WHERE (age BETWEEN 28 AND 34)
=> [#<Person id: 8, first_name: "Kalman", age: 33, last_name: "Smith", created_at: "2015-09-08 22:22:51", updated_at: "2015-09-08 2
2:22:51", login: "kman", pass: "abc123">, #<Person id: 14, first_name: "LeBron", age: 30, last_name: "James", created_at: "2015-09-
08 22:22:52", updated_at: "2015-09-08 22:22:52", login: "bron", pass: "need more rings">]
irb(main):002:0> Person.where("first_name LIKE ? OR last_name LIKE ?", '%J%', '%J%').to_a
  Person Load (0.3ms)  SELECT "people".* FROM "people" WHERE (first_name LIKE '%J%' OR last_name LIKE '%J%')
=> [#<Person id: 9, first_name: "John", age: 27, last_name: "Whatever", created_at: "2015-09-08 22:22:51", updated_at: "2015-09-08
22:22:51", login: "john1", pass: "123abc">, #<Person id: 11, first_name: "Josh", age: 57, last_name: "Oreck", created_at: "2015-09-
08 22:22:51", updated_at: "2015-09-08 22:22:51", login: "josh", pass: "password1">, #<Person id: 12, first_name: "John", age: 27, l
ast_name: "Smith", created_at: "2015-09-08 22:22:51", updated_at: "2015-09-08 22:22:51", login: "john2", pass: "no_idea">, #<Person
 id: 14, first_name: "LeBron", age: 30, last_name: "James", created_at: "2015-09-08 22:22:52", updated_at: "2015-09-08 22:22:52", l
ogin: "bron", pass: "need more rings">]
```

Array Condition Syntax issues

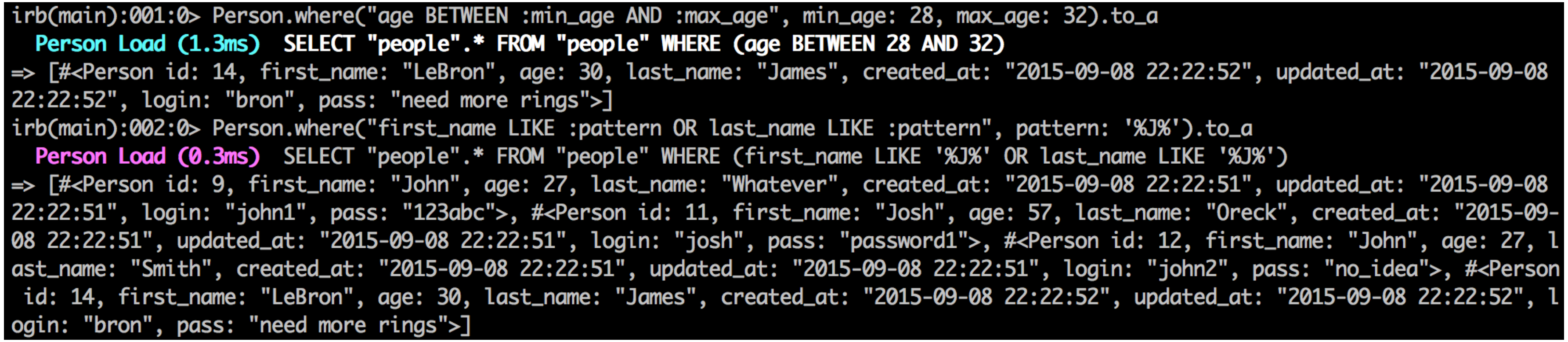
✧ Array Condition Syntax is "SQL Injection safe" and easy to use, but there are now two (small) problems:

1. You have to keep track of the order of parameters "hiding" behind the "`?`"

2. If you have *n* "`?`" – you need to pass in *n* values, even if they are a reference to the same value

Hash Condition Syntax

- Instead of "?", you specify symbols which map to the values in the hash passed in as a second parameter



```
irb(main):001:0> Person.where("age BETWEEN :min_age AND :max_age", min_age: 28, max_age: 32).to_a
  Person Load (1.3ms)  SELECT "people".* FROM "people" WHERE (age BETWEEN 28 AND 32)
=> [#<Person id: 14, first_name: "LeBron", age: 30, last_name: "James", created_at: "2015-09-08 22:22:52", updated_at: "2015-09-08
22:22:52", login: "bron", pass: "need more rings">]
irb(main):002:0> Person.where("first_name LIKE :pattern OR last_name LIKE :pattern", pattern: '%J%').to_a
  Person Load (0.3ms)  SELECT "people".* FROM "people" WHERE (first_name LIKE '%J%' OR last_name LIKE '%J%')
=> [#<Person id: 9, first_name: "John", age: 27, last_name: "Whatever", created_at: "2015-09-08 22:22:51", updated_at: "2015-09-08
22:22:51", login: "john1", pass: "123abc">, #<Person id: 11, first_name: "Josh", age: 57, last_name: "Oreck", created_at: "2015-09-
08 22:22:51", updated_at: "2015-09-08 22:22:51", login: "josh", pass: "password1">, #<Person id: 12, first_name: "John", age: 27, l
ast_name: "Smith", created_at: "2015-09-08 22:22:51", updated_at: "2015-09-08 22:22:51", login: "john2", pass: "no_idea">, #<Person
 id: 14, first_name: "LeBron", age: 30, last_name: "James", created_at: "2015-09-08 22:22:52", updated_at: "2015-09-08 22:22:52", l
ogin: "bron", pass: "need more rings">]
```

Summary

- ✧ Always use either the Array or Hash Condition Syntax to avoid SQL injection
- ✧ Hash syntax seems more intuitive to most people

What’s Next?

- ✧ One-to-One Association