Scopes with has_many and belongs_to

Defining Scopes on the Association

```
class User < ActiveRecord::Base
    has_many :posts,
             -> {where('created_at > ?', Time.current - 1.year)}
end
```

Design Note: In general, I'd recommend not scoping associations directly like this (aka "default scope"). This is because this can sometimes be a source of confusion. Imagine someone doing `@user.posts` and wondering why older posts are not returned. Consider this instead:

```
class User < ActiveRecord::Base
    has_many :posts
    has_many :recent_posts,
             -> {where('created_at > ?', Time.current - 1.year)},
             class_name: 'Post'
end
```

This way, doing `@user.posts` will return all posts, and `@user.recent_posts` will return only ones which have been created in the past year.

Defining a Scope that References Attributes of the Association

```
class User < ActiveRecord::Base
  has_many :posts
  scope :with_pending_posts,
        -> {joins(:posts).where('posts.pending = true')}
end
```

With the above, you can do `User.with_pending_posts`, which will return all users in the database with pending posts. If you've already defined a scope in the association model, you can absolutely make use of this. In our example above, if we had a scope defined in the Post model like:

```
class Post < ActiveRecord::Base
  belongs_to :user
  scope :pending, -> { where(pending: True) }
end
```

You can then do:

```
class User < ActiveRecord::Base
  has_many :posts
  scope :with_pending_posts,
        -> {joins(:posts).merge(Post.pending)}
end
```

For the full article, visit: https://ducktypelabs.com/using-scope-with-associations/

Do you want to implement a scope not covered by the examples above? Email me at sidk@ducktypelabs.com and let's see if we can't figure out a way to do it!