Phase 3 SQLAlchemy Code Challenge: Restaurants

New Attempt

Due No Due Date **Points** 15 **Submitting** a website url

Phase 3 Code Challenge: Restaurants

For this assignment, we'll be working with a restaurant review domain.

We have three models: `Restaurant`, `Review`, and `Customer`.

For our purposes, a `Restaurant` has many `Review`s, a `Customer` has many `Review`s, and a `Review` belongs to a `Restaurant` and to a `Customer`.

`Restaurant` - `Customer` is a many to many relationship.

Note: You should draw your domain on paper or on a whiteboard _before you start coding_. Remember to identify a single source of truth for your data.

Topics

- SQLAlchemy Migrations
- SQLAlchemy Relationships
- Class and Instance Methods
- SQLAlchemy Querying

***

Instructions

Build out all of the methods listed in the deliverables.

The methods are listed in a suggested order, but you can feel free to tackle the ones you think are easiest. Be careful: some of the later methods rely on earlier ones.

Remember! This code challenge does not have tests. You cannot run `pytest`.

You'll need to create your own sample instances so that you can try out your code on your own. Make sure your relationships and methods work in the console before submitting.

You are also encouraged to use the `seeds.py` file to create sample data to test your models and relationships.

Writing error-free code is more important than completing all of the deliverables listed - prioritize writing methods that work over writing more methods that don't work. You should test your code in the console as you write.

Similarly, messy code that works is better than clean code that doesn't. First, prioritize getting things working. Then, if there is time at the end, refactor your code to adhere to best practices.

Before you submit! Save and run your code to verify that it works as you expect. If you have any methods that are not working yet, feel free to leave comments describing your progress.

What You Need to Have

You need to have migrations and models for the initial `Restaurant` and `Customer` models, and seed data for some `Restaurant`s and `Customer`s.

The schema currently looks like this:

restaurants table

Column	Type
name	String
price	Integer

customers Table

Column	Type
first_name	String
last_name	String

You will need to create the migration for the `reviews` table using the attributes specified in the deliverables below.

Deliverables

Write the following methods in the classes. Feel free to build out any helper methods if needed.

Remember: SQLAlchemy give your classes access to a lot of methods already! Keep in mind what methods SQLAlchemy gives you access to on each of your

classes when you're approaching the deliverables below.

Migrations

Before working on the rest of the deliverables, you will need to create a migration for all tables.

- A `**Review**` belongs to a `**Restaurant**`, and a `**Review**` also belongs to a `**Customer**`. In your migration, create any columns your `**reviews**` table will

need to establish these relationships.

- The `**reviews**` table should also have: - A `**star_rating**` column that stores an integer.

After creating the `**reviews**` table using a migration, use the `**seeds.py**` file to create instances of all your classes so you can test your code.

****Once you've set up your tables****, work on building out the following deliverables.

Object Relationship Methods

Use SQLAlchemy query methods where appropriate.

Review

- `**Review customer()**`

- should return the `Customer` instance for this review

- `**Review restaurant()**`

- should return the `Restaurant` instance for this review

Restaurant

- **`Restaurant reviews()`**
  - returns a collection of all the reviews for the `Restaurant`
- **`Restaurant customers()`**
  - returns a collection of all the customers who reviewed the `Restaurant`

Customer

- **`Customer reviews()`**
  - should return a collection of all the reviews that the `Customer` has left
- **`Customer restaurants()`**
  - should return a collection of all the restaurants that the `Customer` has reviewed

Check that these methods work before proceeding. For example, you should be able to call **`session.query(Customer).first().restaurants`** and see a list

of the restaurants for the first customer in the database based on your seed data; and **`session.query(Review).first().customer`** should return the customer for the first review in the database.

Aggregate and Relationship Methods

Customer

- **`Customer full_name()`**
  - returns the full name of the customer, with the first name and the last name concatenated, Western style.
- **`Customer favorite_restaurant()`**

- returns the restaurant instance that has the highest star rating from this customer

- **`Customer add_review(restaurant, rating)`**

- takes a `restaurant` (an instance of the `Restaurant` class) and a rating

- creates a new review for the restaurant with the given `restaurant_id`

- **`Customer delete_reviews(restaurant)`**

- takes a `restaurant` (an instance of the `Restaurant` class) and

- removes ****all**** their reviews for this restaurant

- you will have to delete rows from the `reviews` table to get this to work!

Review

- **`Review full_review()`**

- should return a string formatted as follows:

Review for {insert restaurant name} by {insert customer's full name}: {insert review star_rating} stars.

Restaurant

- **`Restaurant fanciest()`**, this method should be a class method`

- returns _*one*_ restaurant instance for the restaurant that has the highest price

- **`Restaurant all_reviews()`**

- should return an list of strings with all the reviews for this restaurant

formatted as follows:

[

"Review for {insert restaurant name} by {insert customer's full name}: {insert review star_rating} stars.",

"Review for {insert restaurant name} by {insert customer's full name}: {insert review star_rating} stars.",

]