UDACITY

Logout

‹ Return to Classroom

Fyyur: Artist Booking Site

REVIEW | CODE REVIEW 7 | HISTORY

▸ app.py 4

▾ Models.py 2

```

#----------------------------------------------------------------------------#
# Imports
#----------------------------------------------------------------------------#

import json
import dateutil.parser
import babel
from flask import Flask, render_template, request, Response, flash, redirect, url_for, jsonify, abort
from datetime import datetime
from flask_moment import Moment
from flask_sqlalchemy import SQLAlchemy
import logging
from logging import Formatter, FileHandler
from flask_wtf import Form
from forms import *
import sys as sys
from flask_migrate import Migrate

#from sqlalchemy import  create_engine
from sqlalchemy_utils import database_exists, create_database
from sqlalchemy_utils.functions import database_exists

from config import *

#engine = create_engine(SQLALCHEMY_DATABASE_URI)
if not database_exists(SQLALCHEMY_DATABASE_URI):
    create_database(SQLALCHEMY_DATABASE_URI)

app = Flask(__name__)
db = SQLAlchemy(app)
moment = Moment(app)
app.config.from_object('config')
migrate = Migrate(app,db)


#----------------------------------------------------------------------------#
# Models.
#----------------------------------------------------------------------------#


class Venue(db.Model):
    __tablename__ = 'Venue'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String, nullable=False)
    city = db.Column(db.String(120),nullable=False)
    state = db.Column(db.String(120),nullable=False)
    address = db.Column(db.String(120),nullable=False)
    phone = db.Column(db.String(120))
    image_link = db.Column(db.String(500))
    facebook_link = db.Column(db.String(120))
    genres = db.Column("genres", db.ARRAY(db.String()), nullable=False)

    website_link = db.Column(db.String(120))
    seeking_talent=db.Column(db.Boolean, nullable=False, default=False)
```

Rate this review

START

```
    seeking_description= db.Column(db.String(120))
    shows = db.relationship('Show', backref='venue', lazy=True)
```

REQUIRED

> ⚠ There is a problem with this implementation

One of the rubric points specifically asks for using the `JOIN` strategy - and when you implement the `lazy` engine, you will e differently.

I want to highlight that I understand what you did above, and this does achieve the same result. Still, the usage of explicit `j` through SQLAlchemy helpers must be implemented in order to meet the requirement.

```
    def __repr__(self):
      return f'<Venue {self.id} name: {self.name}>'


    # TODO: implement any missing fields, as a database migration using Flask-Migrate

class Artist(db.Model):
    __tablename__ = 'Artist'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String, nullable=False)
    city = db.Column(db.String(120),nullable=False)
    state = db.Column(db.String(120),nullable=False)
    phone = db.Column(db.String(120))
    image_link = db.Column(db.String(500))
    facebook_link = db.Column(db.String(120))
    website_link = db.Column(db.String(120))
    genres =db.Column("genres", db.ARRAY(db.String()), nullable=False)
    seeking_venue=db.Column(db.Boolean, nullable=False, default=False)
    seeking_description= db.Column(db.String(120))
    shows = db.relationship('Show', backref='artist', lazy=True)

    def __repr__(self):
      return f'<Artist {self.id} name: {self.name}>'

class Show(db.Model):
    __tablename__ = 'Show'
    id = db.Column(db.Integer, primary_key=True)
    artist_id = db.Column(db.Integer, db.ForeignKey('Artist.id'), nullable=False)
    venue_id = db.Column(db.Integer, db.ForeignKey('Venue.id'), nullable=False)
```

AWESOME

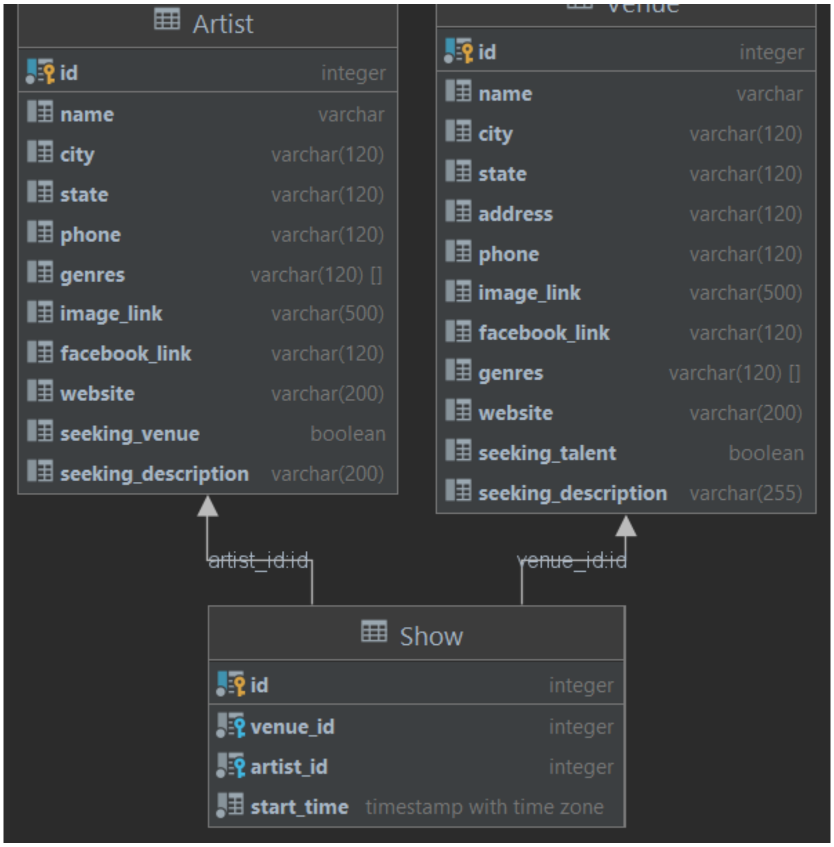
Your application implements the `Show` class as required.

This strategy establishes the correct relationship between the application entities; implementing that model ensure that you

1. List all shows from a given Artist.
2. List all shows from a given Venue.
3. Associate Artist and Venue information from a given Show.

Take a look at the model below that corroborates with this pattern.

is review
ART



Artist	
id	integer
name	varchar
city	varchar(120)
state	varchar(120)
phone	varchar(120)
genres	varchar(120) []
image_link	varchar(500)
facebook_link	varchar(120)
website	varchar(200)
seeking_venue	boolean
seeking_description	varchar(200)

Venue	
id	integer
name	varchar
city	varchar(120)
state	varchar(120)
address	varchar(120)
phone	varchar(120)
image_link	varchar(500)
facebook_link	varchar(120)
genres	varchar(120) []
website	varchar(200)
seeking_talent	boolean
seeking_description	varchar(255)

Show	
id	integer
venue_id	integer
artist_id	integer
start_time	timestamp with time zone

As an engineer, you need to be aware this is not the only way of associating data, and each technology presents benefits an crucial to be able to consider different approaches for different problems; with that in mind, I suggest the following articles:

- SQL Data modeling and different table association strategies
- Data relationship - NoSQL vs SQL

```
  start_time = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)

  def __repr__(self):
   return f'<Show {self.id}, Artist {self.artist_id}, Venue {self.venue_id}>'


  # TODO: implement any missing fields, as a database migration using Flask-Migrate

# TODO Implement Show and Artist models, and complete all model relationships and properties, as a da

```

▸ config.py 1

▸ fabfile.py

▸ forms.py

▸ migrations/env.py

▸ migrations/README

▸ migrations/versions/092c81e0ae06_initial_database_creation.py

▸ migrations/versions/c7d504feae69_added_all_models.py

▸ migrations/versions/dc594c07605c_added_website_to_artists.py

▸ migrations/versions/e4a166de4cb9_added_fields_to_artist.py

Rate this review

START

▸ migrations/versions/[illegible]_added_fields_to_artist.py

▸ README.md

▸ requirements.txt

▸ static/css/bootstrap.css

▸ static/css/bootstrap.min.css

▸ static/css/bootstrap.min.js

▸ static/css/bootstrap-theme.css

▸ static/css/bootstrap-theme.min.css

▸ static/css/layout.forms.css

▸ static/css/layout.main.css

▸ static/css/main.css

▸ static/css/main.quickfix.css

▸ static/css/main.responsive.css

▸ static/js/libs/bootstrap-3.1.1.min.js

▸ static/js/libs/jquery-1.11.1.min.js

▸ static/js/libs/modernizr-2.8.2.min.js

▸ static/js/libs/moment.min.js

▸ static/js/libs/respond-1.4.2.min.js

▸ static/js/plugins.js

▸ static/js/script.js

▸ templates/errors/404.html

▸ templates/errors/500.html

▸ templates/forms/edit_artist.html

▸ templates/forms/edit_venue.html

▸ templates/forms/new_artist.html

▸ templates/forms/new_show.html

▸ templates/forms/new_venue.html

Rate this review

START

▸ templates/layouts/form.html

▸ templates/layouts/form.html

▸ templates/layouts/main.html

▸ templates/pages/artists.html

▸ templates/pages/home.css

▸ templates/pages/home.html

▸ templates/pages/search_artists.html

▸ templates/pages/search_venues.html

▸ templates/pages/show.html

▸ templates/pages/show_artist.html

▸ templates/pages/show_venue.html

▸ templates/pages/shows.html

▸ templates/pages/venues.html

RETURN TO PATH

Rate this review

START