Unit 10 SQLAlchemy homework - VSCode

from matplotlib import style

#style.use('fivethirtyeight')

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import datetime as dt

# Python SQL toolkit and Object Relational Mapper

import sqlalchemy

from sqlalchemy.ext.automap import automap\_base

from sqlalchemy.orm import Session

from sqlalchemy import create\_engine, inspect, func

engine = create\_engine("sqlite:///Resources/hawaii.sqlite")

from flask import Flask, jsonify

import datetime as dt

engine = create\_engine("sqlite:///Resources/hawaii.sqlite")

# Create engine.#Use SQLAlchemy create\_engine to connect to your sqlite database.

hawaii\_path = "Resources/hawaii.sqlite"

engine\_hawaii = create\_engine(f"sqlite:///{hawaii\_path}")

#engine\_hawaii = create\_engine(f"sqlite:///Resources/hawaii.sqlite")

hawaii = pd.read\_sql("SELECT \* FROM measurement", engine\_hawaii)

hawaii.head()

hawaii = pd.read\_sql("SELECT \* FROM station", engine\_hawaii)

hawaii.head()

inspector = inspect(engine\_hawaii)

inspector.get\_table\_names()

Base = automap\_base()

Base.prepare(engine\_hawaii, reflect=True)

Base.classes.keys()

measurement = Base.classes.measurement

station = Base.classes.station

# Get a list of column names and types

columns = inspector.get\_columns('measurement')

for c in columns:

    print(c['name'], c["type"])

 # Get a list of column names and types

columns = inspector.get\_columns('station')

for c in columns:

    print(c['name'], c["type"])

    # Flask Setup

app = Flask(\_\_name\_\_)

 # Flask Routes

@app.route("/")

def welcome():

    """List all available API routes."""

    return (

        f"Available Routes:<br/>"

        f"/api/v1.0/about<br/>"

        f"/api/v1.0/precipitation<br/>"

        f"/api/v1.0/stations<br/>"

        f"/api/v1.0/tobs<br/>"

        f"/api/v1.0/temp\_start<br/>"

        f"/api/v1.0/temp-start\_end<br/>"

        f"/api/v1.0/contact"

    )

@app.route("/api/v1.0/about")

def about():

   name = "Flynn LLC"

   location = "Arizona"

   return f"We are {name} and we are located in sunny {location}."

# /api/v1.0/precf"/api/v1.0/precipitation<br/>"precipitation

# Convert the query results to a dictionary using date as the key and prcp as the value.

# Return the JSON representation of your dictionary.

@app.route("/api/v1.0/precipitation")

def precipitation():

    # Create our session (link) from Python to the DB

    session = Session(engine)

    """Return a list of all precipitation"""

    # Query all measurements

    results = session.query(measurement.date, measurement.prcp).all()

    session.close()

    # Convert list of tuples into normal list

    all\_precipitation = list(np.ravel(results))

    return jsonify(all\_precipitation)

# /api/v1.0/stations

# Return a JSON list of stations from the dataset.

@app.route("/api/v1.0/stations")

def stations():

    # Create our session (link) from Python to the DB

    session = Session(engine)

    """Return a list of all stations"""

    # Query all stations

    results = session.query(station.station).all()

    session.close()

    # Convert list of tuples into normal list

    all\_stations = list(np.ravel(results))

    return jsonify(all\_stations)

# Query for the date and precipitation (max, mean, min)

# for all dates in the year

# Sort the result by date

@app.route("/api/v1.0/tobs")

def tobs():

    # Create our session (link) from Python to the DB

    session = Session(engine)

    """Return a list of all tobs"""

    # Query all observations

    results = session.query(measurement.tobs).filter(measurement.station =="USC00519281").all()

    session.close()

    # Convert list of tuples into normal list

    all\_tobs = list(np.ravel(results))

    return jsonify(all\_tobs)

# /api/v1.0/<start>/<end>

# Return a JSON list of the minimum temperature, the average temperature, and the max

# temperature for a given start or start-end range.

@app.route("/api/v1.0/temp/<start>")

def tobs\_start(start=None):

    sel = [func.min(measurement.tobs), func.avg(measurement.tobs), func.max(measurement.tobs)]

    results = session.query(\*sel).filter(measurement.date>=start).all()

    all\_tobs = list(np.ravel(results))

    return jsonify(all\_tobs)

@app.route("/api/v1.0/temp/<start>/<end>")

def tobs\_start\_end(start=None, end=None):

    # Create our session (link) from Python to the DB

    session = Session(engine)

    """Return a list of all tobs"""

    # Query all observations

    sel = [func.min(measurement.tobs), func.avg(measurement.tobs), func.max(measurement.tobs)]

    results = session.query(\*sel).filter(measurement.date>=start).filter(measurement.date<=end).all()

    session.close()

    # Convert list of tuples into normal list

    all\_tobs = list(np.ravel(results))

    return jsonify(all\_tobs)

@app.route("/api/v1.0/contact")

def contact():

   email = "Flynnworking@gmail.com"

   return f"Questions, Comments, Complaints? Send an email to {email}."

if \_\_name\_\_ == "\_\_main\_\_":

 app.run(debug=True)