Unit Assessment: Databases

After submitting the assessment, you will see a summary of your performance. While you will not be able to see your performance on individual questions, you are allowed unlimited attempts to complete the assessment.

Some of the questions on this assessment require specific resources. Download the following resources before you get started. There are also several websites used as question resources listed below.

players.csv

matches.csv

schema.txt

dracula.txt

CTA - System Information - List of 'L' Stops (https://data.cityofchicago.org/Transportation/CTA-System-Information-List-of-L-Stops/8pix-ypme)

CTA - Ridership - 'L' Station Entries - Daily Totals (https://data.cityofchicago.org/Transportation/CTA-Ridership-L-Station-Entries-Daily-Totals/5neh-572f)

Census_Data.sqlite

←

Question 1

Using *players.csv*

Your department is participating as a team in a fantasy tennis competition at work, and your boss wants to drive the decisions with data.

In pgAdmin, create a database called “tennis_db” and make a new table called “players.” Create columns to match the headers in the *players.csv* file. Import the CSV into the new table.

What data type is most appropriate for the “player_id” column?

- STRING
- VARCHAR
- [✓ INT] ✓
- FLOAT

5/5

Question 2

Using *matches.csv*

Your boss's favorite player is Serena Williams. They've asked for a list of every match she's won.

Create a new table in tennis_db called “matches." Create columns to match the headers in the *matches.csv* file. Import the CSV into the new table.

Write a query to display all of the matches where Serena Williams won.

How many matches in the dataset has Serena Williams won?

- ○ 41
- ○ 8
- ○ 45
- [✓ ● 37] ✓

5/5

Question 3

Using the *tennis_db* from questions 1 and 2, write a query that returns the count of players for each dominant hand group (i.e. right, left).

What is the distribution of the players' dominant hands?

- ○ 5023 are left handed, 487 are right handed.
- ○ 93 are left handed, 898 are right handed.
- [✓ ● 487 are left handed, 5023 are right handed.] ✓
- ○ 898 are left handed, 1456 are right handed.

5/5

Question 4

(continued from last question)

Still using the *tennis_db*, write a query that returns the number of WINS for each dominant hand.

What is the distribution of wins for each dominant hand?

HINT: This will require you to join two tables together.

- 80 wins for left handed players, 2126 wins for right handed players.
- [✓ 241 wins for left handed players, 2126 wins for right handed players.] ✓
- 2126 wins for left handed players, 241 wins for right handed players.
- 487 wins for left handed players, 5023 wins for right handed players.

5/5

Question 5

You've been hired to create a SQL database for a local gym owner. Based on your meetings with the owner, you've determined that the initial database design should follow *schema.txt*.
The gym owner contacts you to let you know that the gym is going to start offering drop-in classes. The classes are paid for separately, and trainers are paid a percentage commission for the class (commission is determined for each class).

Using https://app.quickdatabasediagrams.com/#/, add two new tables with the following columns to the ERD:

```
Classes
-
Class_ID PK
Trainer_ID
Gym_ID
Class_Name
Commission_Percentage

Class_Attendance
-
Member_ID
Class_ID
```

Add the appropriate data types and foreign key constraints to your ERD.

How many foreign key relationships needed to be added?

- 2
- 6
- [✓ 4] ✓
- 3

5/5

Question 6

(continued from last question)

Using the ERD from the last question, export the ERD to a PostgreSQL query and open the query.

What is the keyword used when adding the foreign key constraints?

- [✓ `ALTER`] ✓
- `CREATE`
- `UPDATE`
- `INSERT`

5/5

Question 7

(continued from last question)

Create a “gym_db” database in pgAdmin. Open the query tool and run the exported ERD query to create all the tables.

The following SQL code is attempting to insert a class into the database, but it returns an error.

```
insert into gym
(gym_id, gym_name, address, city, zipcode)
values (1, 'Average Joe''s Gymnasium', '123 Main St.', 'Springfield', '12345');

insert into classes
(class_id, trainer_id, gym_id, class_name, commission_percentage)
values (1,1,1,'Wrench Dodging',0.1);

insert into trainers
(trainer_id, gym_id, first_name, last_name)
values (1, 1, 'Patches', 'O''Houlihan');
```

What needs to be changed for the code to run correctly?

- [✓ The insert into “trainers” needs to happen before the insert into “classes”] ✓
- The ‘commission_percentage’ value should be a percent, not a decimal
- Foreign key restraints need to be temporarily relaxed
- The IDs need to be changed to unique values

5/5

Question 8

Using:

CTA - System Information - List of 'L' Stops

CTA - Ridership - 'L' Station Entries - Daily Totals

A journalist has contacted you to perform data analysis for an article they're writing about CTA ridership. They want to investigate how the CTA serves the North and South sides of Chicago. They've provided you two datasets with ridership information and station information, but they need to merge them together to connect ridership data with location data.

The list of L stops has multiple stops per station, to account for the direction the trains are heading in. You need to reduce the information down to one line per station.

1. Drop *STOP_ID*, *DIRECTION_ID*, and *STOP_NAME*
2. Compress the boolean columns (*ADA, RED, BLUE, G, BRN, P, Pexp, Y, Pnk, O*) using pandas.groupby().any()
3. Remove duplicate rows
4. Don't worry about parsing Location, we'll do that in the next question

How many rows are in the transformed dataset?

- 204
- [✓ 147] ✗
- 146
- 283

0/5

Question 9

(Continued from the previous question)

The following code will perform the task from the previous question:

```
station_bools =
l_stops_df[['MAP_ID','ADA','RED','BLUE','G','BRN','P','Pexp','Y','Pnk','O']
].groupby('MAP_ID').any()
l_stops_df = l_stops_df.drop(['STOP_ID', 'DIRECTION_ID', 'STOP_NAME',
'ADA','RED','BLUE','G','BRN','P','Pexp','Y','Pnk','O'], axis=1) \
    .merge(station_bools, how='left', left_on='MAP_ID',
right_index=True).drop_duplicates()
```

A journalist has contacted you to perform data analysis for an article they're writing about CTA ridership. They want to investigate how the CTA serves the North and South sides of Chicago. They've provided you two datasets with ridership information and station information, but they need to merge them together to connect ridership data with location data.

The *Location* column is currently stored as a string. Parse the *Location* column into a *Latitude* and *Longitude* column using a regular expression to replace the parentheses and the `pandas.Series().str.split()` method. Convert the now split numbers to numeric data types.

What character needs to be placed before a parenthesis in a regular expression to escape the parenthesis?

- ○ “
- ○ /
- [✓ \ ] ✓
- ○ #

5/5

Question 10

(Continued from the previous question)

The following code will perform the task from the previous question:

```
l_stops_df[['latitude','longitude']] =
l_stops_df['Location'].str.replace('\(|\)','',
regex=True).str.split(',',expand=True).apply(pd.to_numeric)
```

A journalist has contacted you to perform data analysis for an article they’re writing about CTA ridership. They want to investigate how the CTA serves the North and South sides of Chicago. They’ve provided you two datasets with ridership information and station information, but they need to merge them together to connect ridership data with location data.

Merge the ridership dataset and station dataset by *station_id* and *MAP_ID*, respectively.

What type of join is the most appropriate to use for this merge?

- ○ inner
- [✓ left] ✓
- ○ full outer
- ○ right

5/5

Question 11

(Continued from the previous question)

The following code will perform the task from the previous question:

The following code will perform the task from the previous question:
```
df = pd.merge(ridership_df, l_stops_df, how='left', left_on='station_id',
right_on='MAP_ID')
```

A journalist has contacted you to perform data analysis for an article they're writing about CTA ridership. They want to investigate how the CTA serves the North and South sides of Chicago. They've provided you two datasets with ridership information and station information, but they need to merge them together to connect ridership data with location data.

Create a boolean column in the merged dataset to determine if a station is in the South side or not, using the latitude 41.881 as a cutoff. Calculate the mean rides per station for the North side and South side.

What is the mean rides per station for the South side (rounded to the nearest whole number)?

- [✓ 3015] ✓
- 3021
- 2887
- 2008

5/5

Question 12

(Continued from the previous question)

The following code will perform the task from the previous question:
```
df['south_side'] = df['latitude'] < 41.881
df[['south_side','rides']].groupby('south_side').mean()
```

A journalist has contacted you to perform data analysis for an article they're writing about CTA ridership. They want to investigate how the CTA serves the North and South sides of Chicago. They've provided you two datasets with ridership information and station information, but they need to merge them together to connect ridership data with location data.

Convert the *date* column to a datetime data type and create a new column for just the year. Calculate the mean rides per station, grouping by year and side.

In which year was the absolute difference between North side ridership and South side ridership the smallest?

- 2005
- 2009
- 2010
- [✓ 2008] ✓

5/5

Question 13

(Continued from the previous question)

The following code will perform the task from the previous question:

```
rides_by_sides = df[['year','south_side','rides']] \
    .groupby(['year','south_side']).mean() \
    .reset_index() \
    .pivot(index='year',columns='south_side',values='rides')
rides_by_sides.columns = ['north','south']
(rides_by_sides['north'] - rides_by_sides['south']).plot(kind='bar')
```

A journalist has contacted you to perform data analysis for an article they're writing about CTA ridership. They want to investigate how the CTA serves the North and South sides of Chicago. They've provided you two datasets with ridership information and station information, but they need to merge them together to connect ridership data with location data.

Clean up any duplicated columns (e.g. *station_id* & *MAP_ID*, *Location* & *Latitude/Longitude*) in the merged dataset. Open up pgAdmin and create a database called "CTA_DB". Create a schema to match the merged dataset. Load the merged dataset into CTA_DB using `pandas.to_sql()` .

Which column, if any, should be the primary key?

○ `north`

○ `rides`

○ `year`

[✓ ● `station_id`] ✓

5/5

Question 14

Using *Census_Data.sqlite*

The marketing department wants to focus on areas around the country that satisfy certain demographics.

Download *Census_Data.sqlite* and use SQLAlchemy to connect to the database. Use Pandas to import the Census_Data table into a DataFrame.

Out of the cities with populations over 100,000 people, which one has the youngest median age?

[✓ ● College Station, TX] ✗

○ Athens, GA

○ Provo, UT

○ Delray Beach, FL

0/5

Question 15

Consider this code snippet from a Flask app:

```
@app.route("/api/v1.0/users/<id>")
def user(***):
```

What needs to replace `***` in order to get the user's id?

- [✓ `id`] ✓
- `“/api/v1.0/users/<id>”`
- `<id>`
- We can replace the `***` with any word.

5/5

Question 16

The following function returns the index page for a Flask route, but it’s missing the Flask decorator:

```
# MISSING CODE HERE
def index():
    """List all available api routes."""
    return (
        f"Available Routes:<br/>"
        f"/api/v1.0/names<br/>"
        f"/api/v1.0/passengers"
    )
```

What line of code needs to be added above the function to route to the index?

- [✓ `@app.route('/')`] ✓
- `index('/')`
- `app.route('/')`
- `route('/')`

5/5

Question 17

You've been given data on passengers of the Titanic in a SQLite file, and you want to create an API that will serve the data in a JSON format. You've created a bare-bones Flask app that connects to the SQLite database and serves a welcome page. Eventually, you will add two endpoints: one that returns a list of all database and serves a welcome page. Eventually, you will add two endpoints: one that returns a list of all passenger names, and a more detailed endpoint that returns a list of all passengers, including their name, age, and sex. But first, you need a reference to the *passenger* table in the SQLite database.

```
from sqlalchemy.ext.automap import automap_base
from sqlalchemy.orm import Session
from sqlalchemy import create_engine

from flask import Flask, jsonify

engine = create_engine("sqlite:///titanic.sqlite")
Base = automap_base()
Base.prepare(engine, reflect=True)

Passenger = # MISSING CODE HERE

app = Flask(__name__)

@app.route("/")
def welcome():
   """List all available api routes."""
   return (
       f"Available Routes:<br/>"
       f"/api/v1.0/names<br/>"
       f"/api/v1.0/passengers"
   )


if __name__ == '__main__':
   app.run(debug=True)
```

On line 11, what does Passenger need to have assigned to it so that it will refer to the *passenger* table in the SQLite database?

- `sqlite:///titanic.sqlite/passengers`
- `passenger`
- [✓ `Base.classes.passenger`] ✓
- `Base.tables.passenger`

5/5

Question 18

(Continued from the previous question)

Next, create an endpoint to list all of the names of the passengers. Use the route */api/v1.0/names*.

```
from sqlalchemy.ext.automap import automap_base
from sqlalchemy.orm import Session
from sqlalchemy import create_engine

from flask import Flask, jsonify

engine = create_engine("sqlite:///titanic.sqlite")
Base = automap_base()
Base.prepare(engine, reflect=True)

Passenger = Base.classes.passenger

app = Flask(__name__)

@app.route("/")
def welcome():
   """List all available api routes."""
   return (
       f"Available Routes:<br/>"
       f"/api/v1.0/names<br/>"
       f"/api/v1.0/passengers"
   )

### Create an endpoint for names ###

if __name__ == '__main__':
   app.run(debug=True)
```

On line 24, what is the first line you will have to write to make a new route for passenger names?

- `route('/api/v1.0/names')`
- `def names():`
- `f"/api/v1.0/names"`
- [✓ `@app.route('/api/v1.0/names')`] ✓

5/5

Question 19

(Continued from the previous question)

An endpoint has been created for all passenger data, and it connects to the database and converts all of the data into a list of dictionaries. However, the last line of the function is missing.

```
from sqlalchemy.ext.automap import automap_base
from sqlalchemy.orm import Session
from sqlalchemy import create_engine

from flask import Flask, jsonify

engine = create_engine("sqlite:///titanic.sqlite")
Base = automap_base()
Base.prepare(engine, reflect=True)

Passenger = Base.classes.passenger

app = Flask(__name__)

@app.route("/")
def welcome():
   """List all available api routes."""
   return (
       f"Available Routes:<br/>"
       f"/api/v1.0/names<br/>"
       f"/api/v1.0/passengers"
   )

@app.route("/api/v1.0/names")
def names():
   """Return a list of all passenger names"""
   # Query all passengers
   session = Session(engine)
   results = session.query(Passenger.name).all()

   # Convert list of tuples into normal list
   all_names = list(np.ravel(results))

   return jsonify(results)

@app.route("/api/v1.0/passengers")
def passengers():
   """Return a list of passenger data including the name, age, and sex of each passenge
r"""
   # Query all passengers
   session = Session(engine)
   results = session.query(Passenger.name, Passenger.age, Passenger.sex).all()

   # Create a dictionary from the row data and append to a list of all_passengers
   all_passengers = []
   for name, age, sex in results:
       passenger_dict = {}
       passenger_dict["name"] = name
       passenger_dict["age"] = age
       passenger_dict["sex"] = sex
       all_passengers.append(passenger_dict)
```

```
    ### Return list of all passenger data in JSON format ###

if __name__ == '__main__':
    app.run(debug=True)
```

On line 52, what line of code needs to be inserted so that the list of all passenger data will be sent to the end-point in JSON format?

- ○ `return all_passengers`
- ○ `return all_passengers.json()`
- ○ `return json(all_passengers)`
- [✓ `return jsonify(all_passengers)`] ✓

5/5

Question 20

Using *dracula.txt*

Use the following code and replace `p` with a regular expression to find the most common word that follows “vampire” in the text:

```
import pandas as pd
import re

dracula_df = pd.read_csv('dracula.txt', sep='\n', header=None)
dracula_df.columns = ['text']

p = 'YOUR REGULAR EXPRESSION HERE'
dracula_df['text'].str.extractall(p, flags=re.I)[0].value_counts()
```

What is the most common word that follows “vampire” in the text?

- ○ rest
- ○ drink
- [✓ sleep] ✓
- ○ live

5/5

Retake