Florida Polytechnic University

COP 3710

Fall 2019

Group Se7en Project

Luiz Gustavo Malpele

Elijah Luckey

Gunnar Sundberg

Thomas Sawyer

December 4th, 2019

0.1 Table of Contents

1 Overview

1.1 Introduction

Our project is focused on modeling the operations of a rehab program and using the power of data to drive better decision-making, which, in turn, improves the health of clients and the community. Using our database, it is possible to compute graduation rates, length of time in program, the number of classes attended, and other client data.

To better understand how our database can be utilized, think of the daily operations of a rehab program and its clients. A person seeking rehabilitation would send in an application through our system. Clients will then be accepted based on the organization's policies and background checks. Once a client is accepted into the program, they become a resident - living on the organization's campus. They are assigned a mentor (member/employee of the organization) and a room number – where the newly accepted resident will stay while in the program. While in the program, clients take classes to progress toward graduation. If the individual does well in the program, they will graduate from the program after completing the required classes (in this case 16). If the individual does not fit well after becoming a resident, there is an option to mark a resident as ejected while maintaining the client's data for further analysis.

1.2 Data Modeling

Using the business rules established in the overview, we constructed an ERD to plan our database. We went through several iterations, even making changes to our model late in the semester after discussing normalization in class.

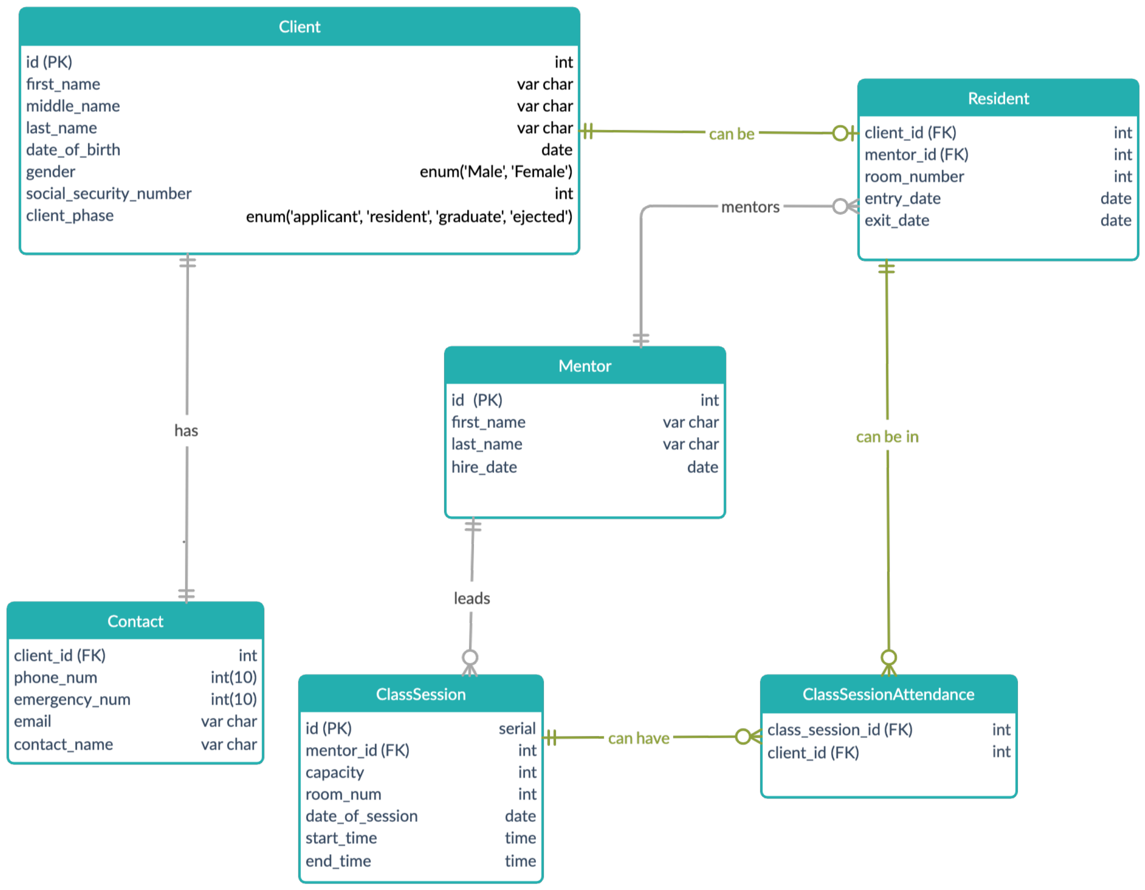


Figure 1: ERD

The relationships between attributes can be shown using dependency diagrams for the tables. Our database meets industry-standard 3NF, with extremely performant queries and no data anomalies.

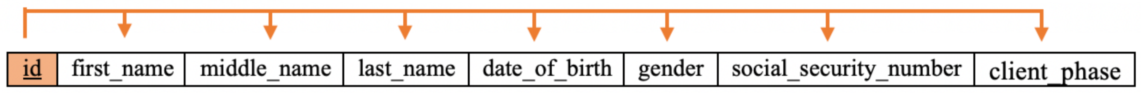


Figure 2: Client Dependency Diagram

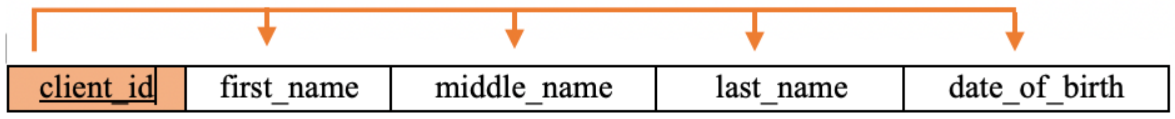


Figure 3: Contact Dependency Diagram

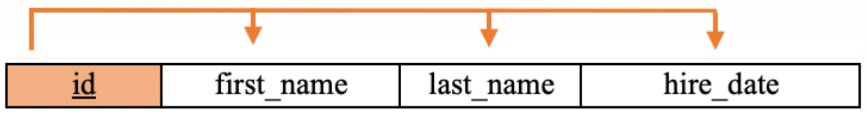


Figure 4: Mentor Dependency Diagram

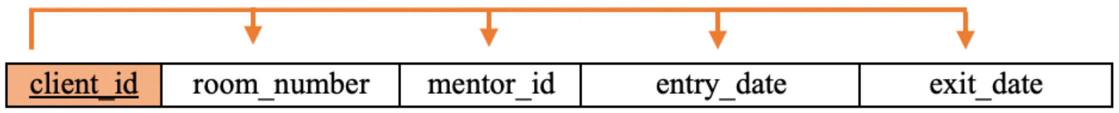


Figure 5: Resident Dependency Diagram

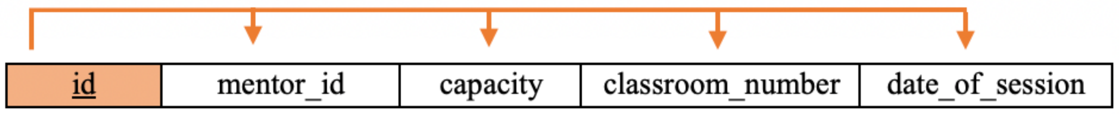


Figure 6: ClassSession Dependency Diagram

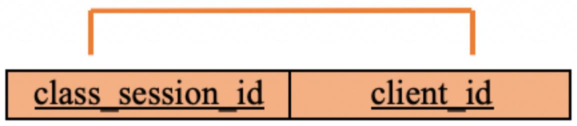


Figure 7: ClassSessionAttendance Dependency Diagram

1.3 Data Acquisition

Every table in our database uses human data, and as such, it was necessary that we generate mock data. To accomplish this, we used an online tool called Mockaroo to generate csv files. We then used SQL scripts to add data from the csv files to the database. Mockaroo provides parameters for each data type to ensure the data is reasonable. Beyond Mockaroo's generation parameters, it was necessary to perform manual checks of the data, which was one of the most time-consuming elements of the project. We used queries to find inconsistent data and delete or update it. We left some duplicates in our data to demonstrate our database's ability to stop duplication and anomalies. The SQL scripts and updated csv files have been included with the project submission.

1.4 Conventions

The design of the database was determined by the team involved. We made careful use of naming conventions for ease of use when working with our database.

1.4.1 Database Name

The database naming is simple, we simply use a variation of `CamelCase` to create the name `RehabProgram`.

1.4.2 Table Names

For table names we use `PascalCase`, which is a subset of the `CamelCase` convention, where each word in the title of the table is a new capital letter. For example `ClassSessionAttendance` does not contain underscores (_) to make it read `Class_Session_Attendance` or `class_session_attendance`.

When choosing names, it must directly correlate to the nature of the data stored in the table. The `Client` table will not include details of any classes. This lowers the normal form (not ideal) of the system, and ultimately makes it inefficient.

1.4.3 Attribute Names

As table names use `PascalCase`, the attributes will use `snake_case`, always separating words with underscores (_). This was decided so that any interfaces with tables/attributes, the user will distinctly be aware of the attributes and tables.

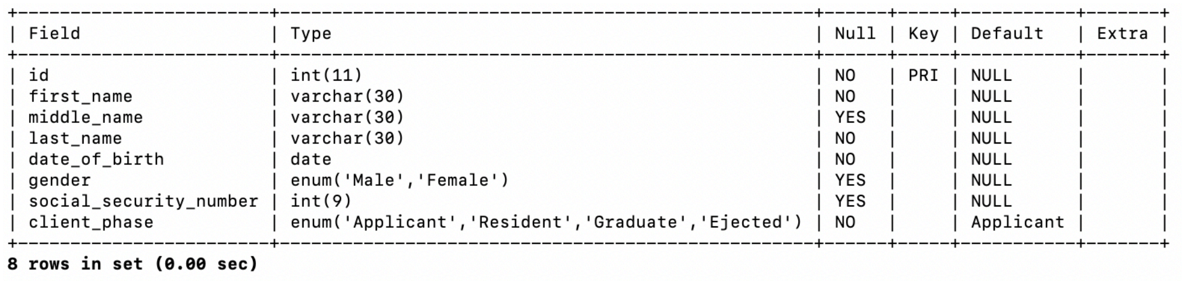
The naming of the attributes inherits the idea that the table (a subject) is *implied* when referencing an attribute. For example, the `Client` table has an attribute for ID's. `Client.client_id` contains redundancy in the reference name. `Client.id` is an ideal reference for an attribute that operates as an ID. Likewise goes for other attributes.

Acronyms are always spelled out. "Date of birth" will become `date_of_birth`, not `dob`. This is decided since acronyms are ambiguous and not always deterministic. Another example, "social security number" could easily become `ssn`, but even this remains unclear if users are not native English speakers.

1.5 Structure

As described in the Data Modeling section, the database includes 6 tables: `Client`, `Contact`, `Mentor`, `Resident`, `ClassSession`, and `ClassSessionAttendance`.

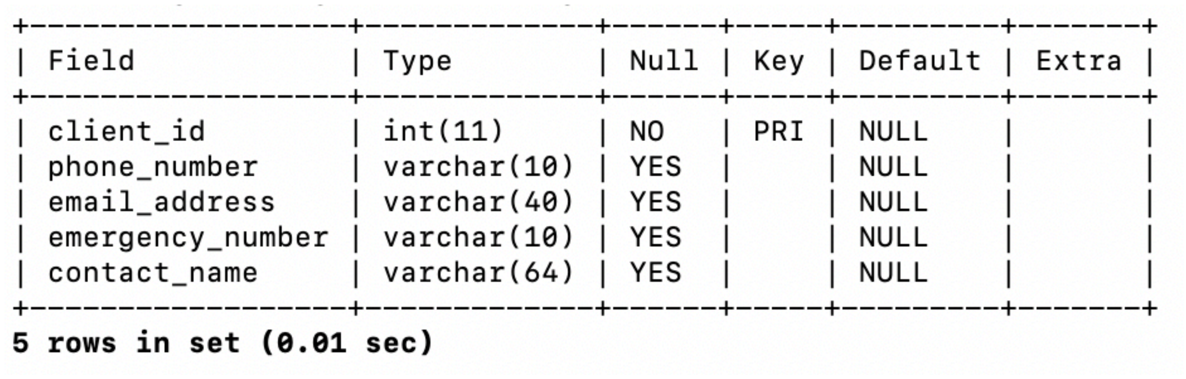
1.5.1 Client



```
+------------------------+---------------------------------------------------+------+-----+-----------+-------+
| Field                  | Type                                              | Null | Key | Default   | Extra |
+------------------------+---------------------------------------------------+------+-----+-----------+-------+
| id                     | int(11)                                           | NO   | PRI | NULL      |       |
| first_name             | varchar(30)                                       | NO   |     | NULL      |       |
| middle_name            | varchar(30)                                       | YES  |     | NULL      |       |
| last_name              | varchar(30)                                       | NO   |     | NULL      |       |
| date_of_birth          | date                                              | NO   |     | NULL      |       |
| gender                 | enum('Male','Female')                             | YES  |     | NULL      |       |
| social_security_number | int(9)                                            | YES  |     | NULL      |       |
| client_phase           | enum('Applicant','Resident','Graduate','Ejected') | NO   |     | Applicant |       |
+------------------------+---------------------------------------------------+------+-----+-----------+-------+
8 rows in set (0.00 sec)
```

Figure 8: describe Client

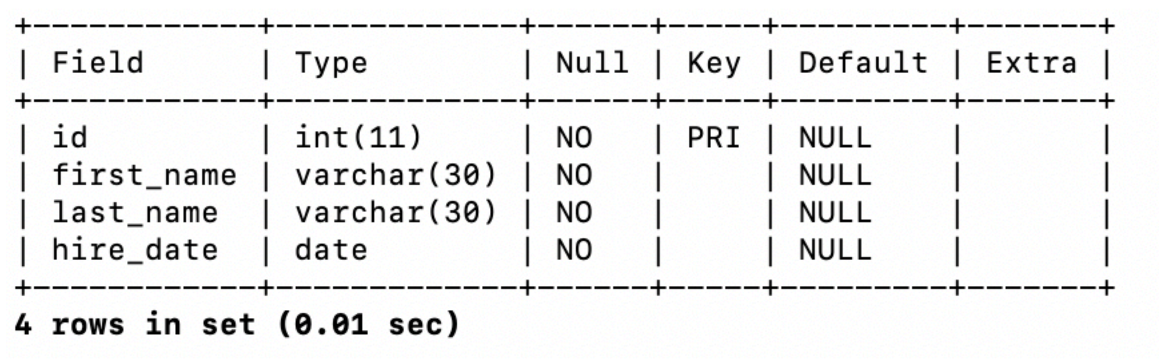
1.5.2 Contact



```
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| client_id        | int(11)     | NO   | PRI | NULL    |       |
| phone_number     | varchar(10) | YES  |     | NULL    |       |
| email_address    | varchar(40) | YES  |     | NULL    |       |
| emergency_number | varchar(10) | YES  |     | NULL    |       |
| contact_name     | varchar(64) | YES  |     | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Figure 9: describe Contact

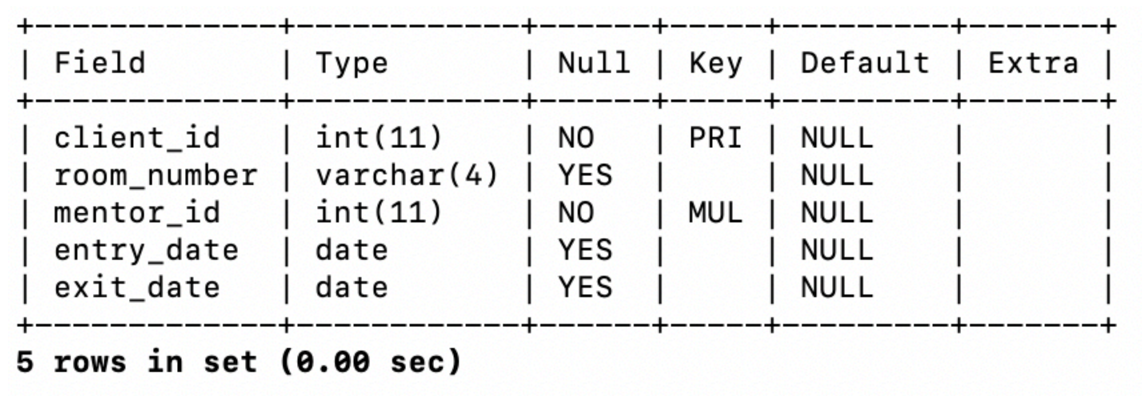
1.5.3 Mentor



```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int(11)     | NO   | PRI | NULL    |       |
| first_name | varchar(30) | NO   |     | NULL    |       |
| last_name  | varchar(30) | NO   |     | NULL    |       |
| hire_date  | date        | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Figure 10: describe Mentor

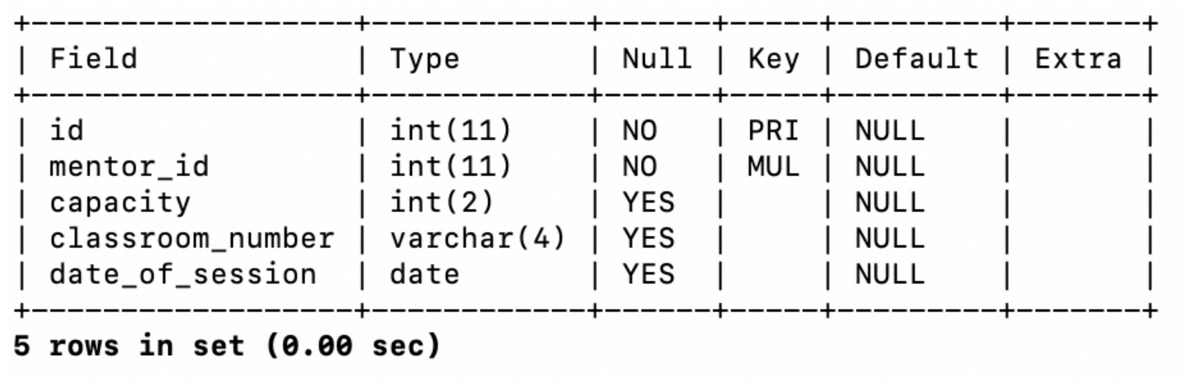
1.5.4 Resident



```
+-------------+------------+------+-----+---------+-------+
| Field       | Type       | Null | Key | Default | Extra |
+-------------+------------+------+-----+---------+-------+
| client_id   | int(11)    | NO   | PRI | NULL    |       |
| room_number | varchar(4) | YES  |     | NULL    |       |
| mentor_id   | int(11)    | NO   | MUL | NULL    |       |
| entry_date  | date       | YES  |     | NULL    |       |
| exit_date   | date       | YES  |     | NULL    |       |
+-------------+------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Figure 11: describe Resident

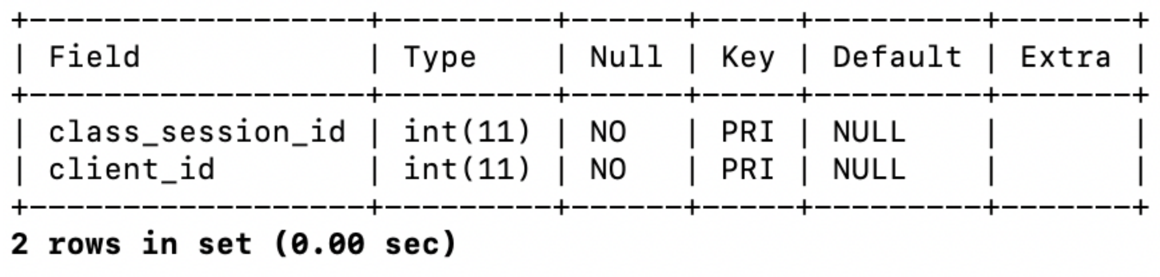
1.5.5 ClassSession



```
+------------------+------------+------+-----+---------+-------+
| Field            | Type       | Null | Key | Default | Extra |
+------------------+------------+------+-----+---------+-------+
| id               | int(11)    | NO   | PRI | NULL    |       |
| mentor_id        | int(11)    | NO   | MUL | NULL    |       |
| capacity         | int(2)     | YES  |     | NULL    |       |
| classroom_number | varchar(4) | YES  |     | NULL    |       |
| date_of_session  | date       | YES  |     | NULL    |       |
+------------------+------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Figure 12: describe ClassSession

1.5.6 ClassSessionAttendance



```
+-------------------+---------+------+-----+---------+-------+
| Field             | Type    | Null | Key | Default | Extra |
+-------------------+---------+------+-----+---------+-------+
| class_session_id  | int(11) | NO   | PRI | NULL    |       |
| client_id         | int(11) | NO   | PRI | NULL    |       |
+-------------------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Figure 13: describe ClassSessionAttendance

1.5.7 Data Dictionary

2 Setup and Usage

Our final database is available to use on ember as `gsundberg3513-RehabProgram`. Permissions have been granted to user `msamarah`. If you have access issues, email Gunnar. Instructions for recreating the database locally are also included below.

2.1 Setup

2.1.1 Prerequisites

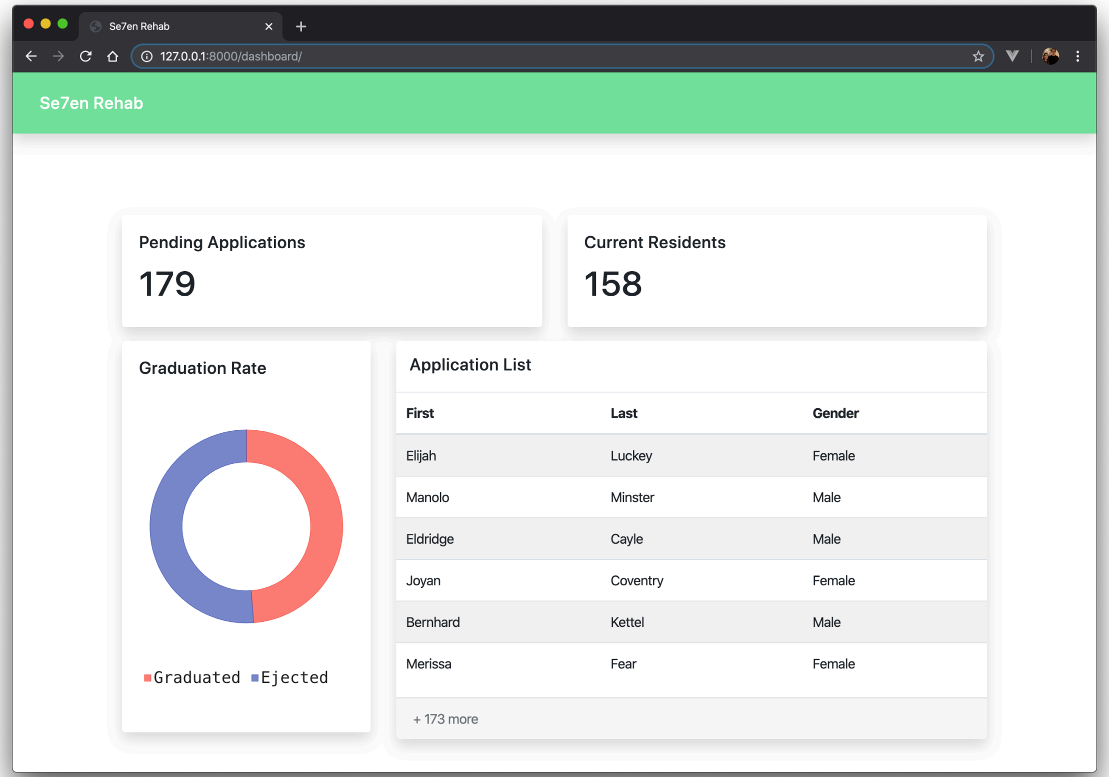
It is recommended that tests be performed on the live version in ember, but for some use cases it may be necessary to run the database locally. For instance, in order to use the bundled demo frontend web application, the database must run on your local machine due to restrictions on ember. For creating and running the database, all that is needed is `mariadb`. To run the web app, `python3` and `pipenv` are necessary.

2.1.2 Database Quickstart

- To create the database and define the tables, run `tables.sql` located in `src`.
- In order to insert the data (located in the `data` folder), move each of the csv files into your home directory.
- Then run `loaddata.sql`, also located in `src`. Note: You should receive a warning that some data was skipped. This is normal. As explained earlier, our database is built to withstand duplication attempts.

2.1.3 Demo Web Application

To demonstrate a practical application of our database, we made a simple dashboard application to display some basic information pertinent to organization operations.



Se7en Rehab

Pending Applications
179

Current Residents
158

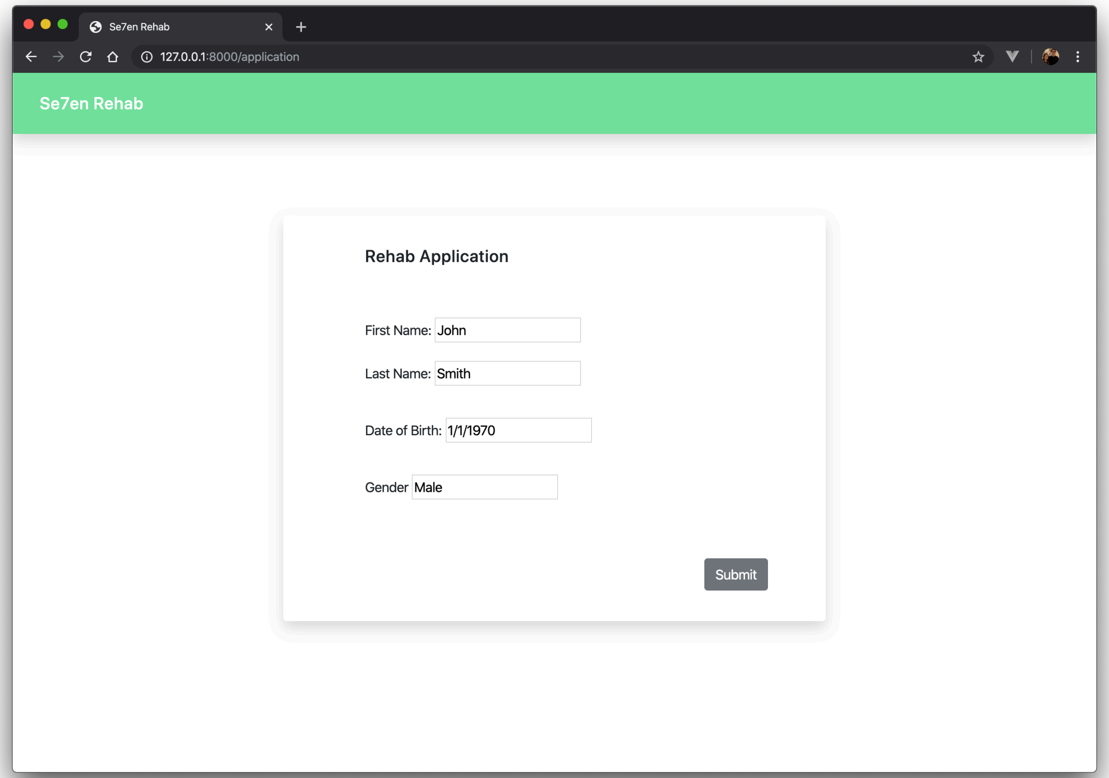
Application List

First	Last	Gender
Elijah	Luckey	Female
Manolo	Minster	Male
Eldridge	Cayle	Male
Joyan	Coventry	Female
Bernhard	Kettel	Male
Merissa	Fear	Female

+ 173 more

Figure 14: GUI Dashboard

In addition to the dashboard, there is also an application form for prospective clients. Submitting this form will actually update the database. Try submitting an application yourself. Note that the form does not have data validation capabilities, so stick to what is specified by the data dictionary.



Rehab Application

First Name: John

Last Name: Smith

Date of Birth: 1/1/1970

Gender Male

Figure 15: GUI Application Form

2.1.4 Demo Web Application Quick Start

Note that the database must be named “RehabProgram” created by user “root” with password “password”.

- Navigate to the `demo_app` folder and run `pipenv install`. This will create a virtual environment with all necessary dependencies.
- Next, run `pipenv shell` to open a shell using your newly created virtual environment.
- From the shell, navigate to the `demo_app` folder, then run `python3 manage.py makemigrations` followed by `python3 manage.py migrate --fake webapp`.
- Lastly, run `python3 manage.py runserver`.

- Open your browser. The dashboard should be accessible at `http://127.0.0.1/dashboard` and the application at `http://127.0.0.1/application`.

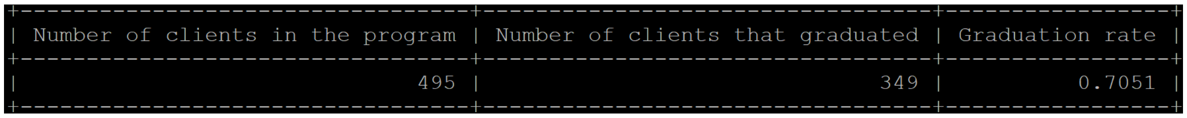
2.2 Usage and Sample Queries

2.2.1 Graduation Rate

SQL:

```
select
  count(Client.id) as "Number of clients in the program",
  Sum(
    Case
      When Client.client_phase = 'Graduate' Then 1
      Else 0
    End
  ) as "Number of clients that graduated",
  Sum(
    Case
      When Client.client_phase = 'Graduate' Then 1
      Else 0
    End
  ) / count(Client.id) as "Graduation rate"
from Client
where
  client_phase = 'Ejected'
  or client_phase = 'Graduate';
```

Output:



```
+---------------------------------+---------------------------------+-----------------+
| Number of clients in the program | Number of clients that graduated | Graduation rate |
+---------------------------------+---------------------------------+-----------------+
|                             495 |                             349 |          0.7051 |
+---------------------------------+---------------------------------+-----------------+
```

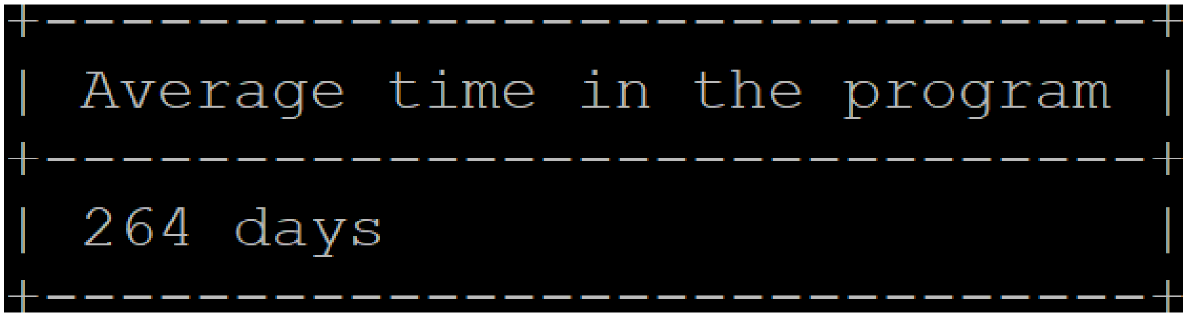
Figure 16: Graduation Rate

2.2.2 Average Length of Stay

SQL:

```sql
select
  concat(
    round(
      Sum(datediff(Resident.exit_date, Resident.entry_date)) / count(Client.id)
    ),
    " days"
  ) as "Average time in the program"
from Client,
  Resident
where
  Client.id = Resident.client_id
  and client_phase = 'Ejected'
  or client_phase = 'Graduate';
```

Output:



```
+-----------------------------+
| Average time in the program |
+-----------------------------+
| 264 days                    |
+-----------------------------+
```

Figure 17: Graduation Rate

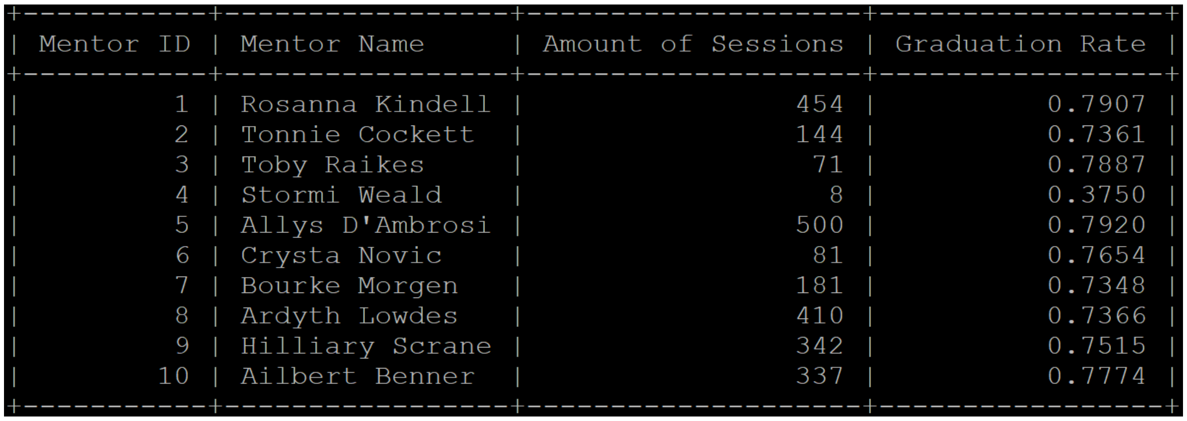
2.2.3 Graduation Rate by Mentor

SQL:

```sql
select
    Mentor.id as "Mentor ID",
    concat(Mentor.first_name, " ", Mentor.last_name) as "Mentor Name",
    count(ClassSessionAttendance.class_session_id) as "Amount of Sessions",
    Sum(
    Case
      When Client.client_phase = 'Graduate' Then 1
      Else 0
    End
  ) / (
    Sum(
      Case
        When Client.client_phase = 'Ejected' Then 1
        Else 0
      End
    ) + Sum(
      Case
        When Client.client_phase = 'Graduate' Then 1
        Else 0
      End
    )
  ) as "Graduation Rate"
from ClassSessionAttendance,
    ClassSession,
    Mentor,
    Client
```

```
where
  ClassSession.id = ClassSessionAttendance.class_session_id
    and ClassSessionAttendance.client_id = Client.id
    and ClassSession.mentor_id = Mentor.id
group by
  Mentor.id
order by
  Mentor.id
limit
  10;
```

Output:



```
+-----------+-----------------+--------------------+------------------+
| Mentor ID | Mentor Name     | Amount of Sessions | Graduation Rate  |
+-----------+-----------------+--------------------+------------------+
|         1 | Rosanna Kindell |                454 |           0.7907 |
|         2 | Tonnie Cockett  |                144 |           0.7361 |
|         3 | Toby Raikes     |                 71 |           0.7887 |
|         4 | Stormi Weald    |                  8 |           0.3750 |
|         5 | Allys D'Ambrosi |                500 |           0.7920 |
|         6 | Crysta Novic    |                 81 |           0.7654 |
|         7 | Bourke Morgen   |                181 |           0.7348 |
|         8 | Ardyth Lowdes   |                410 |           0.7366 |
|         9 | Hilliary Scrane |                342 |           0.7515 |
|        10 | Ailbert Benner  |                337 |           0.7774 |
+-----------+-----------------+--------------------+------------------+
```

Figure 18: Graduation Rate by Mentor

3 Conclusions and Team Contributions

3.1 Conclusions

The database designed succeeded in not only provided the *fictional* rehab program with a digital way of "upgrading" their operations, but also is able to provide impressive insights to the organization.

The first query provided, "Graduation Rate", enables the organization to see a good and *powerful* overview of their organization. The following query, "Average Length of Stay", shares with them about how long a client remains in the program. This can be extremely useful for calculating a "cost-of-living" expense, and how long the organization can expect an incoming client to stay. The "Graduation Rate by Mentor" query is great for seeing the most successful mentors in the program. This serve to be important so the organization can hone-in on the successful mentor's methods and techniques in order to improve organization-wide operations.

This database serves as an excellent means to boost the organizations operations on so many more levels. The power of relational databases lies in a *sharp* design and powerful queries. Through this, we were able to achieve it.

3.2 Contributions

3.2.1 Luiz Gustavo Malpele

- Team leader
- Task delegation and team submissions
- Contributions to Data Dictionary
- Contributions to ER Diagram
- Query design

3.2.2 Elijah Luckey

- Created team communication protocols
- Project topic idea
- Github page
- DDL including all table design
- Data modeling lead

3.2.3 Gunnar Sundberg

- Contributions to ER Diagram
- DDL contributions
- Data generation and checking
- Demo Web Application
- Final report

3.2.4 Thomas Sawyer

- Team logo
- IDEA Expo presentation
- Document editing
- Integrity of data