Name: Troy Ascher

**Table of Contents**

[Project Direction Overview 3](#_Toc78702578)

[Use Cases and Fields 3](#_Toc78702579)

[Use Case 1: Account Creation for users. 3](#_Toc78702580)

[Use Case 2: Account Creation for Doctors and Researchers. 4](#_Toc78702581)

[Use Case 3: Creating a Journal Entry 5](#_Toc78702582)

[Use Case 4: Reading Previous Journal Entries 6](#_Toc78702583)

[Use Case 5: User takes ‘default’ Health and Wellness Survey 7](#_Toc78702584)

[Use Case 6: User Creates a Custom Survey 7](#_Toc78702585)

[Use Case 7: User Loads a Previously Saved Custom Survey Template 8](#_Toc78702586)

[Use Case 8: Deleting Questions from the Health and Wellness Survey 9](#_Toc78702587)

[Use Case 9: Resetting a Password 9](#_Toc78702588)

[Use Case 10: Tracking Food 9](#_Toc78702589)

[Use Case 11: Tracking Exercise 10](#_Toc78702590)

[Use Case 12: User Changes Pseudonym 11](#_Toc78702591)

[Use Case 13: Researcher Creates a Research Project 11](#_Toc78702592)

[Use Case 14: Researcher Adds Users and Patients to Project. 12](#_Toc78702593)

[Use Case 15: User Tracks Weight. 13](#_Toc78702594)

[Structural Database Rules 14](#_Toc78702595)

[Conceptual Entity-Relationship Diagram 18](#_Toc78702596)

[Full DBMS Physical ERD 23](#_Toc78702597)

[Adding Attributes 25](#_Toc78702598)

[Normalization 35](#_Toc78702599)

[SQL Script, Table and Sequence Creation 36](#_Toc78702600)

[History Table and Trigger 37](#_Toc78702601)

[Index Identification and Creations 42](#_Toc78702602)

[Primary Keys 42](#_Toc78702603)

[Foreign Keys 42](#_Toc78702604)

[Unique Indexes 44](#_Toc78702605)

[Index Creation 45](#_Toc78702606)

[Stored Procedure Execution and Explanations 46](#_Toc78702607)

[Use Case 1: Account Creation for users. 46](#_Toc78702608)

[Use Case 2: Account Creation for Doctors and Researchers. 48](#_Toc78702609)

[Use Case 3: Creating a Journal Entry 51](#_Toc78702610)

[Inserting Data 53](#_Toc78702611)

[Question Identification and Explanations 55](#_Toc78702612)

[Query Executions and Explanations 56](#_Toc78702613)

[Data Visualizations 60](#_Toc78702614)

[Summary and Reflection 63](#_Toc78702615)

[Change Log 64](#_Toc78702616)

# Project Direction Overview

My database will support an online health and wellness app that contains a journal, a food and exercise tracker, and that allows users to design surveys to track health and wellness data. Journals are static and contain the same fields for every entry, but surveys and the food and exercise trackers are fully dynamic and can be fully customized to track any questions, food items, and activities. The health and wellness app will also allow help doctors treat their patients by creating patient and doctor groupings. In addition, researchers will be able to create research projects that users can be a part of. In addition to these groups, any person that wants to use the system to write or publish their daily diary or journal while tracking health and wellness data can do so.

Originally, I was interested in this project because I was tracking my own health and wellness data in a paper journal over the past few years and I’d found that doing analysis of it is tedious. I’d explored the idea of using Python to create a journal in a CSV file, but after working through the modules in CS669 I’m fully committed to using a database to start storing this data. In fact, I would like to implement this program in reality to see if others would also find it useful to track their data.

# Use Cases and Fields

I started my design by building Use cases and identifying key fields.

## Use Case 1: Account Creation for users.

Steps:

1. User logs onto website.
2. User clicks, “Create Account”.
3. Website loads the “Account Creation” form.
4. User enters a username, password, and their email address.
5. Account is created and the data is stored in the database along with the date.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| UserName | This will store a VARCHAR field of the user’s chosen username. | This is needed so that the user can publish their journals under a pseudonym. |
| EmailAddress | This will store the user’s unique email address. | Email is needed so that the user’s password can be reset and for logging into the system |
| Password | This field will store the user’s chosen password. | Password is needed so that journal entries and data can be restricted to only the user. |
| UserCreationDate | The is the date that the user created their account on the website. | Creation date is needed so that emails can be sent on the user’s anniversary that will include analytical data such as a word cloud of their most used words. |

*Table 1.1 Fields for Use Case 1.*

## Use Case 2: Account Creation for Doctors and Researchers.

Steps:

1. User logs onto website.
2. User clicks, “Create Account”.
3. Website loads the “Account Creation” form.
4. User enters a username, password, and their email address.
5. User chooses account type from list: ’Doctor’, or ‘Researcher’.
6. Account is created and the data is stored in the database along with the date.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| UserName | This will store a VARCHAR field of the user’s chosen username. | This is needed to log in to the website and so that the user can publish their journals under a pseudonym. |
| EmailAddress | This will store the user’s unique email address. | Email is needed so that the user’s password can be reset. |
| Password | This field will store the user’s chosen password. | Password is needed so that journal entries and data can be restricted to only the user. |
| UserCreationDate | The is the date that the user created their account on the website. | Creation date is needed so that emails can be sent on the user’s anniversary that will include analytical data such as a word cloud of their most used words. |
| FirstName | This will store the user’s first name. | This is needed so that user’s can identify their doctor or the creator of the research project, and also to add a level of accountability to the doctor’s and researcher’s using the system. |
| LastName | This will store the user’s last name. | This is needed so that user’s can identify their doctor or reseracher, and also to add a level of accountability to the doctor’s and researcher’s using the system. |
| Organization | This will store the organization (University, hospital) the user is associated with. | This is needed for analysis, so that the number of users associated with a particular organization is summable. |

*Table 1.2 Fields for Use Case 2.*

## Use Case 3: Creating a Journal Entry

Steps:

1. User opens the journal homepage on the website or APP.
2. User selects Make a Journal Entry and is taken to the journaling page.
3. User enters a title into the “JournalTitle” field.
4. User enters their journal entry into the JournalEntry field, checks a checkbox to indicate if the entry is public or private, and submits the entry.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| UserName | The user’s chosen UserName | This field will auto-populate with the user name so that journal entries get saved to the correct user account in the database. |
| Pseudonym | This stores the name that the user will publish (privately or publicaly) their journal entry under. | This is needed to add a layer of anonymity to the journal, and so that the user can change their published name without causing security issues (like if they were changing their Username often). |
| JournalEntry | This field stores the user’s journal entry. | This field is needed to record and store the journal entries of the user. |
| DailyMood | This stores a mood chosen by the user to represent their mood for the journal entry or day. | This field is needed for search functions and for analysis. Users can search previous entries by mood, and can analyze their mood based on other data saved in the database. |
| JournalTitle | This stores a user generated title for the journal entry being recorded. | This field will be published when the user shares their journal entries and will display when the user searches through previous entries. |
| JournalDate | This is the date that the journal is recorded. | This field will allow users to search by date for journal entries and will also allow for analysis on the frequency of journaling. |
| PublicEntry | This is a binary field that stores if a field is meant to be public or private. | This field will default to private, but will allow users to mark entries as public so that the entries will show up in searches by other users. |

*Table 1.3 Fields for Use Case 3.*

## Use Case 4: Reading Previous Journal Entries

Steps:

1. User goes to the Journal homepage.
2. User selects “Read Previous Entries”.
3. User chooses between their own entries or public entries.
4. User enters search terms or date ranges and submits the search.
5. A list of entries matching the criteria is returned.
6. User selects the entry they want to read.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| JournalEntry | This field stores the user’s journal entry. | This field will be returned |
| Pseudonym | The user’s chosen Pseudonym. | This will allow the user to search for journal entries by pseudonym. It defaults to the current user’s user name, but other text can be entered to search for other user’s public entries. |
| JournalTitle | This stores a user generated title for the journal entry being recorded. | This field will be published when the user shares their journal entries and will display when the user searches through previous entries. |
| JournalDate | This is the date that the journal is recorded. | This field will allow users to search by date for journal entries and will also allow for analysis on the frequency of journaling. |
| PublicEntry | This is a binary field that stores if a field is meant to be public or private. | This field is required because only entries that are marked public will be returned in the search results if the user searches for other user’s entries. |
| DailyMood | This stores a mood chosen by the user to represent their mood for the journal entry or day. | This field is needed so that the user can search for entries with a particular “mood” that they want to read. |

*Table 1.4 Fields for Use Case 4*

*.*

## 

## Use Case 5: User takes ‘default’ Health and Wellness Survey

Steps:

1. User goes to survey page
2. User selects “default” from survey selection dropdown that saves user’s surveys
3. User enters data
4. User clicks “Save”
5. Answers are saved into the database.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| SurveyName | This stores the name of the Survey being taken. In this case, the survey name is ‘default’. | This is needed because each user can have multiple surveys saved. They can select which survey to take from a drop down on the survey page. |
| QuestionText | This stores the text of the question being asked. | This is needed so user’s know which question to answer. |
| AnswerText | This stores the answer that the user inputs. | This is needed so that historic answers can be viewed and analyzed. |
| AnswerDate | This stores the date that answer was/is submitted. | This is needed for future analysis. |

*Table 1.5 Fields for Use Case 5.*

## 

## Use Case 6: User Creates a Custom Survey

Steps:

1. User goes to the “Health and Wellness” page and selects, “Complete today’s wellness survey”.
2. On the survey page user clicks “Create Survey Template” button.
3. User enters a name for the survey.
4. User clicks ‘+’ to add a question.
5. User selects the type of question from a drop down.
6. User enters question text. (Enhancement, survey starts to auto-fill based on questions already stored in the DB).
7. User selects the order that this question will appear in on the survey from a dropdown containing the number of questions on the survey. This dropdown defaults to the next available integer in chronological order.
8. User repeats for up to 15 questions (researchers can add 25 questions).
9. When finished, user clicks “save”.
10. Survey is saved.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| QuestionID | This stores a PKID for each question. | This is needed so that custom questions can be stored on their own table and multiple questions can be saved by the same user. |
| QuestionType | This stores the type of question that the user is saving, eg. Likert, Text, numeric. | This is needed so that the database knows how to store the answer in the database. |
| QuestionUnit | This optional field stores the unit of measure for the question. | This is needed in case the user wants to track measurable units such as body temperature, blood pressure, etc. |
| AnswerText | This stores the answer that the user inputs. | This is needed so that historic answers can be viewed and analyzed. |
| DefaultQuestion | This stores a Y/N value indicating if this question should be included by default for the user. | This is needed so that the user does not have to continually add custom questions to their survey. |
| SurveyName | This field stores a name for the survey. | This is needed so that the user can easily select the survey from a list of available surveys. |
| PublicSurvey | This stores a BOOL 1/0 if the user wants to make their survey template available to other users. | This is needed so that the user can choose if they want to make their survey template public or not. |

*Table 1.6 Fields for Use Case 6.*

## Use Case 7: User Loads a Previously Saved Custom Survey Template

Steps:

1. User goes to the survey page.
2. User selects a previously created survey from the drop-down menu.
3. Survey questions load
4. User completes questions.
5. User clicks submit and the answers are saved to the database.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| SurveyName | This field stores a name for the survey. | This is needed so that the user can easily select the survey from a list of available surveys. |

*Table 1.7 Fields for Use Case 7.*

## Use Case 8: Deleting Questions from the Health and Wellness Survey

Steps:

1. User goes to the “Health and Wellness” page.
2. *User selects the survey they want to edit from the dropdown.*
3. On the survey page user clicks “Edit Survey” button.
4. User clicks “-” button
5. User confirms choice.
6. Question is still available to be used on the survey, but won’t show up on the user’s survey until they edit their survey and re-add it.

## Use Case 9: Resetting a Password

Steps:

1. User goes to Homepage.
2. User clicks the “forget password” button.
3. User is prompted for their username and email address.
4. If the email address and username match a link is sent to the user’s email address with a link to create a new password.
5. User enters a new password, which is updated into the database.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| UserName | This will store a VARCHAR field of the user’s chosen username. | This is needed to determine which user is resetting their password. |
| EmailAddress | This will store the user’s unique email address. | This is needed as an added layer of security when requesting the password reset and also so that a password reset link can be generated for the user. |
| Password | This field will store the user’s chosen password. | This is needed so that the newly created password can be saved into the database. |
| DateCreated | This field will store the date that password is created. | This is needed so that the system can periodically prompt for new passwords, and so that the system knows which password is the current password. |

*Table 1.8 Fields for Use Case 9.*

## Use Case 10: Tracking Food

Steps:

1. User goes to “Health Tracker” on the website/app.
2. User clicks the “Track Food” button.
3. User types *or selects* the name of the food, the portion size (numeric), the time they ate the food, and optionally can estimate the number of calories.
4. User clicks “Submit” and the items are stored in the database.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| Food | This stores the name of the food the user is entering into the food tracker. | The name of the food is needed so that the user can identify trends and habits in the food they are eating. |
| PortionSize | This stores a numberic portion size. Eg. 1, 1.5, 2, .5. | This is needed so that users can track if they are over/under eating on a regular basis. |
| EatTime | This stores the time the user had the meal. It is a user entered date/time. | This is needed for the user to do analysis on their eating patterns and compare it with other data points such as weight, mood, or physical and mental outlook. |
| Calories | This stores an integer that is an estimate of the number of calories the food item was. | This is needed so that user can have a detailed tracker of the calories they are intaking each day. |

*Table 1.9 Fields for Use Case 10.*

## Use Case 11: Tracking Exercise

Steps:

1. User goes to “Health Tracker” on the website/app.
2. User clicks the “Enter Exercise” button.
3. User types *or selects* the name of the exercise completed, the duration of the exercise, the intensity of the exercise, the time it was completed, and optionally can estimate the number of calories burned.
4. User clicks “Submit” and the workout is stored in the database.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| ExerciseType | This stores the name of the exercise being entered. | The name of the excercise is needed so that the user can identify trends and habits in their workout habits. |
| Intensity | This stores a numeral from 1 – Least Intense to 5 – Most Intense. The description will be handled on the application side. | This is needed so that users can track if intensity variation affects other attributes like mood, weight, etc. |
| Duration | This stores the number of minutes that the activity lasted. | This is needed so the user can track how much time they spend working out. |
| CaloriesBurned | This stores an integer that is an estimate of the number of calories the exercise burned. | This is needed so that user can have an estimate of their burned calories, which will be useful to compare to the calories consumed. |

*Table 1.10 Fields for Use Case 11.*

## Use Case 12: User Changes Pseudonym

Steps:

1. User goes to “Account Settings”.
2. User clicks “Change Pseudonym”.
3. User is prompted for their username and password.
4. If the username and password are correct, user is prompted to enter a new Pseudonym.
5. User enters new Pseudonym, which is associated with user’s journal entries and will display on all entries going forward.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| AccountID | This will be a PKID associated with the user’s account. | This is needed so that all user actions can be associated with an unchanging ID, while still allowing the user to change their username over time. |
| Pseudonym | This will store a VARCHAR field of the user’s chosen Pseudonym | This is needed to store the user’s new Pseudonym. |
| Password | This field will store the user’s chosen password. | This is needed to verify the user’s identity. |
| DateCreated | This field will store the date that Pseudonym is created. | This is needed so that the most recent UserName for each user is used. |

*Table 1.11 Fields for Use Case 12.*

## Use Case 13: Researcher Creates a Research Project

Steps:

1. Researcher go to the “Research” page.
2. Researcher selects “Start New Project”
3. Research names project.
4. Researcher marks themself or a different user(s) as the project head.
5. Researcher adds the researcher IDs for other researchers on the project.
6. Researcher clicks “submit”.
7. Project is created in the database.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| ProjectId | This will store the PKID for each project. | This is needed so that each project can be identified. |
| ProjectName | This stores the name of the project. | This is needed to make it easier for researchers to identify the project. |
| ProjectStartDate | This identifies the date the project starts. | This is needed to identify when the project started for analysis purposes. |
| ActiveProject | This identifies if a project is active, or not. | This is needed for analysis purposes. |

*Table 1.12 Fields for Use Case 13.*

## Use Case 14: Researcher Adds Users and Patients to Project.

Steps:

1. Researcher go to the “Research” page.
2. Researcher selects their active project.
3. Researcher clicks “Add Survey Recipients”.
4. Researcher adds the usernames of the user’s who will take the survey(s).
5. Researcher clicks submit.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| ProjectId | This will store the PKID for each project. | This is needed so that each project can be identified. |
| ProjectName | This stores the name of the project. | This is needed to make it easier for researchers to identify the project. |
| ProjectParticipant | This identifies is an account is a participant (research subject) of a project. | This is needed to attach accounts with Projects. |

*Table 1.13 Fields for Use Case 14: Researcher Adds Users and Patients to Project.*

## Use Case 15: User Tracks Weight.

Steps:

1. User goes to the Health Tracker.
2. User enters weight.
3. User enters date.
4. User clicks submit.
5. The data is stored in the user’s account.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| Weight | This will store the user’s weight. | This is needed so the user can track their weight over time and compare it to food intake and exercise. |

*Table 1.14 Fields for Use Case 15: User tracks weight.*

# Structural Database Rules

I’ve broken down my database and business rules in this section. Each rule is in **bold**, with the supporting reasoning in normal typeface.

**Database Rules:**

1. **An account is one or more of three types: Doctor, Researcher, or Patient.**

I decided that one of my target markets will be mental health doctors and their patients. To assist with this, I created account sub-types. Doctor’s will be able to read journals, trackers, and surveys of any account that they are assigned to on the JournalPermission table. I also created the researcher sub-type. Researchers will be able to access large amounts of survey and journal data. Patients are similar to the standard account, but contain a treatment start date and have a relationship that gives doctors and researchers permission to view their journal entries and/or other health and wellness information.

1. **Every Account can author multiple Journal entries; every Journal Entry is authored by one Account.**

Each account (contingent on account type) can write multiple journal entries. Each journal entry must be associated with an account, and only one account can be the author of each entry.

1. **Each Account can complete multiple Surveys; each Survey can be completed by one Account.**

This is straightforward. An account can complete many surveys over time, but each survey instance can only be associated with one account, on the physical ERD this is broken up into multiple tables since the answers to surveys are saved in the UserAnswers entity.

1. **Each survey contains survey questions; each question may be contained in multiple surveys.**

Each template will be associated with questions saved in the Questions entity.

1. **Each survey question may have many answers; each answer must have only one survey question.**

Answers will be saved in the SurveyAnswer table and will be associated with a particular survey instance, as well as an individual survey question (see RULE 8).

1. **Each survey saves survey answers; each survey answer is associated with a survey.**

Answers will be saved in the SurveyAnswer table and will be associated with a particular survey instance, as well as an individual survey question.

1. **Each Question has one Question Type; each Question Type might be associated with many questions.**

Each question will have a type such as “free form”, “Likert Scale (1-7)”, “Likert Scale (1 -5 ), etc. These types will assist with storing answers correctly, and also with the UX design for the website.

1. **Each Question could have one unit of measure; each Unit of Measure could be associated with multiple Questions.**

Each question will have a unit of measure so that during analysis the integrity of data can be ensured. Knowing the units (eg pounds vs kilograms) will allow researchers to easily convert data for uniformity.

1. **Each Question can have 1 Question Category; each Question category may belong to multiple Questions.**

Categories are for analysis and help to group questions. Categories could be “mental health”, “physical activity”, or anything else that question creator decides makes sense.

1. **Each user can record multiple entries in a food tracker; each food tracker entry is recorded by one user.**

Users will enter their daily food intake into the food tracker. Each entry will be associated with a user.

1. **Each health tracker entry can have multiple food items; each food item can be included in multiple food tracker entries.**

The food tracker food items will be saved into a food items table so that food items can be searched and selected when the user completes tracker.

1. **Each health tracker entry can have multiple Exercise entries; each Exercise can be included in multiple health tracker entries.**

The exercise tracker activities will be saved into a exercise table so that exercises can be searched and selected when the user completes tracker.

1. **Each Researcher can be associated with multiple projects; each project may be associated with multiple researchers.**

Researchers can create “projects” that are associated with different survey templates. These projects will also have many patients associated with them which will allow the researchers to gather the same dataset from many different users.

1. **Each Research Project may be associated with multiple Surveys; each Survey may be associated with multiple Research Projects.**

Survey projects can have many surveys associated with them. For instance, one project may have two different surveys that they give. One goes to a control group, the other goes to a group with different demographics. Likewise, since survey templates can be re-used, the survey template may be used in multiple projects.

1. **Each Research Project may be associated with multiple Patients; each Patient may be associated with multiple Research Projects.**

Each Project will have one to many patients that are participants. Being a participant means that the researchers will have access to the patient’s surveys, journals, and food and exercise trackers.

1. **Each Doctor may have permission for multiple Patients; each Patient may give multiple Doctors permission.**

Doctor’s have multiple patients. Likewise, patients can have more than one doctor. When a relationship is present, the doctor will have access to their patient’s journal, and other health and wellness data.

1. **Each WeightHistory record is associated with one Account; an Account may have multiple WeightHistory records.**

This is where changes to Answer.weight will be recorded.

**Business Rules:**

1. **Users can change their Pseudonym up to three(3) times every rolling year.**

Historic pseudonyms will be tied to a user’s journal entries, so limiting the number of changes will keep the availability of popular names high, while also allowing users more than one attempt to make a change each year.

1. **Users can save survey templates for themselves or they can mark them public and they can be shared.**

Templates are a grouping of survey questions that can be used over and over.

1. **‘Normal accounts and Patient account types can have from 1 to 15 questions on their survey templates.**
2. **‘Researcher’ account types can have from 1 to 30 questions on their survey templates.**
3. **Normal accounts can only see their own survey answers.**

Answers to survey questions are stored so that users can easily access the data at a later date, but the data is only available for the answers that each user gives.

1. **Researchers can see survey responses from any user that is a participant on a project they belong to.**
2. **Users can see any journal entry marked “Public”.**

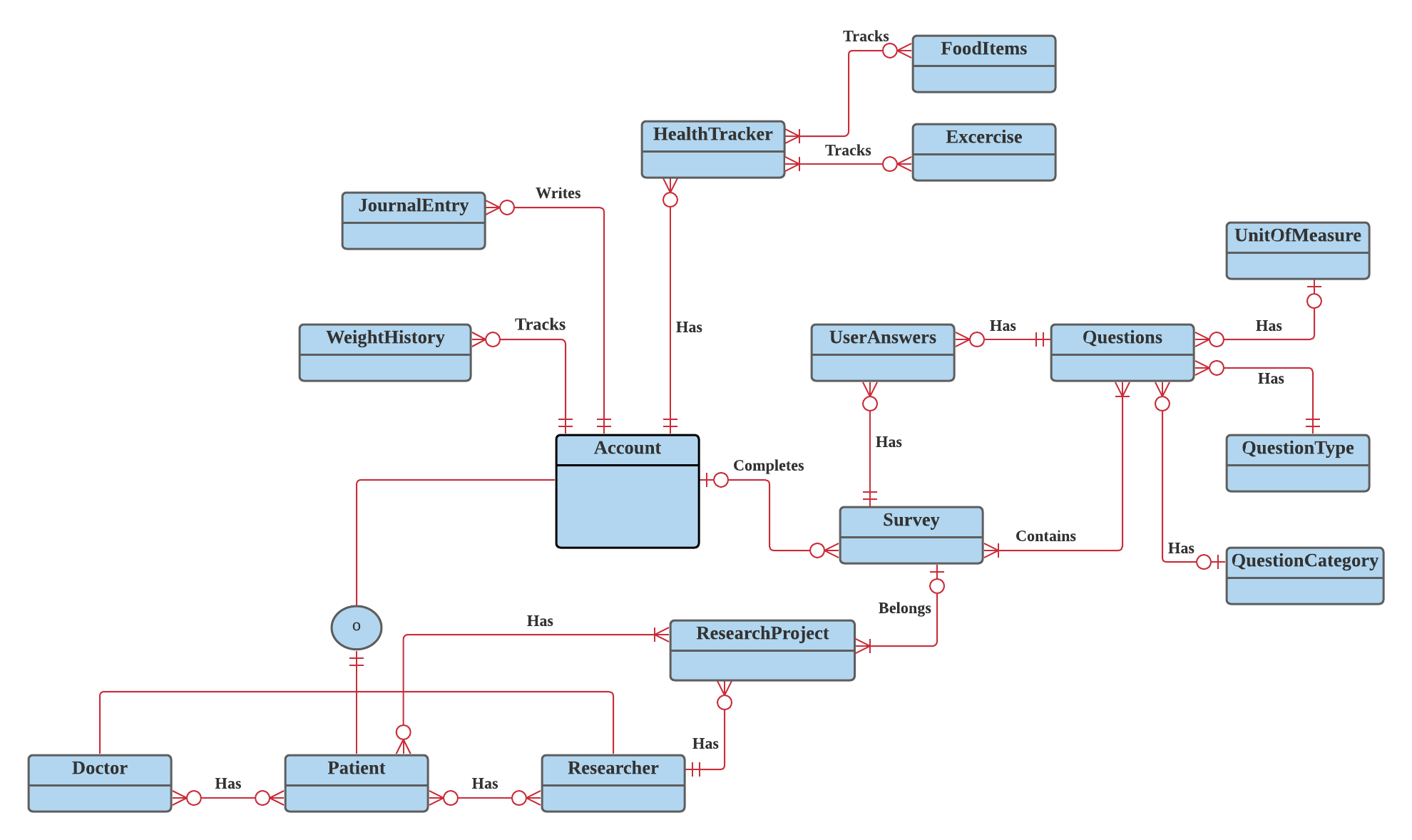
Users can search through and read public journal entries.

1. **“Doctor” account types can see journals and survey data from any of their patients, unless it is marked “hidden” by the patient.**
2. **Only Researchers can participate in Research Projects.**

Since account sub-types are overlap, Doctor’s can create a researcher account to participate in projects.

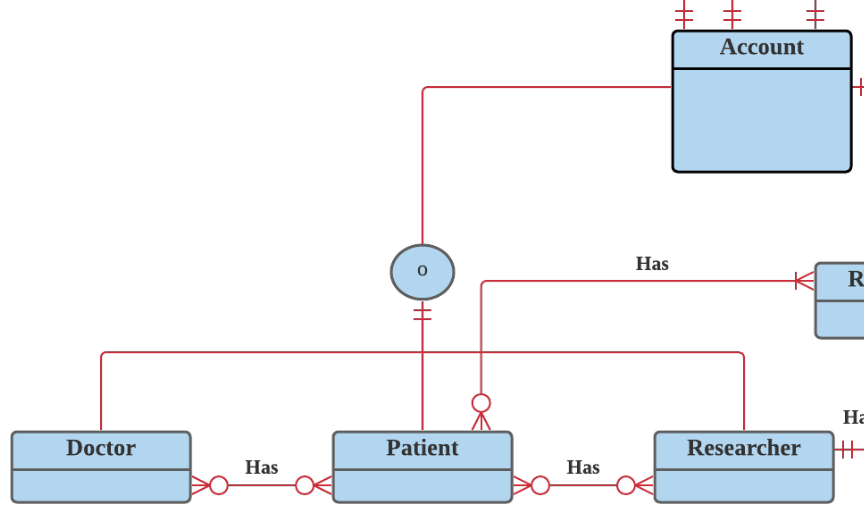
# Conceptual Entity-Relationship Diagram

In this section I present the initial Entity-Relationship Diagram (ERD) for my database. I start by sharing the entire diagram, and then break it up into sections and highlight the details from each section.



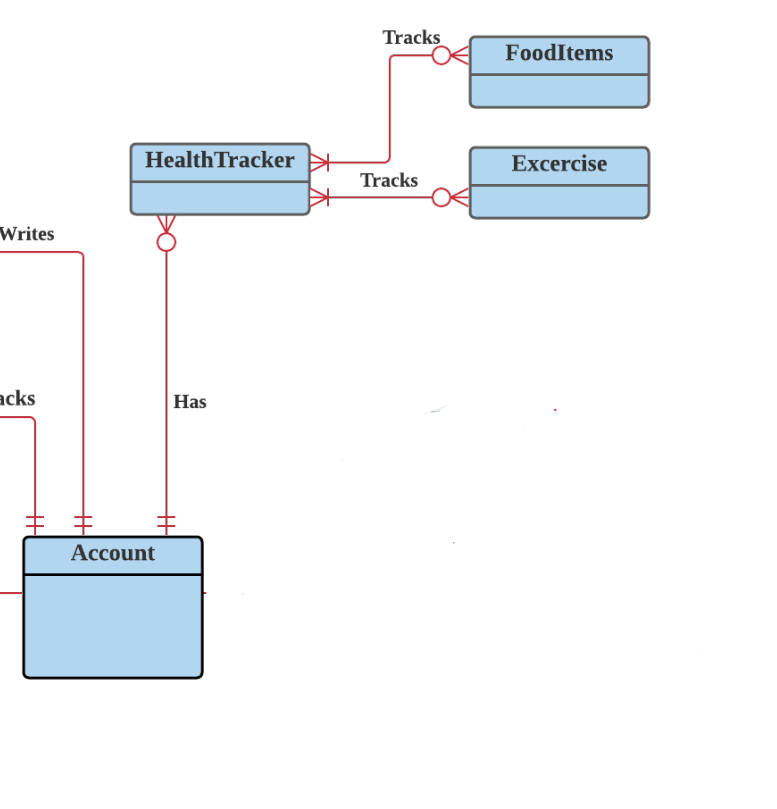
*Figure 1.1 The full ERD for the journaling, health and wellness tracking app/website. Updated 4/19/2021.*

This ERD (**figure 1.1**) can be divided into four sections. The first contains account super-type and sub-type information. The second section is the health tracker and items, the third is the journal entries, and the four is the logic surrounding the survey design.



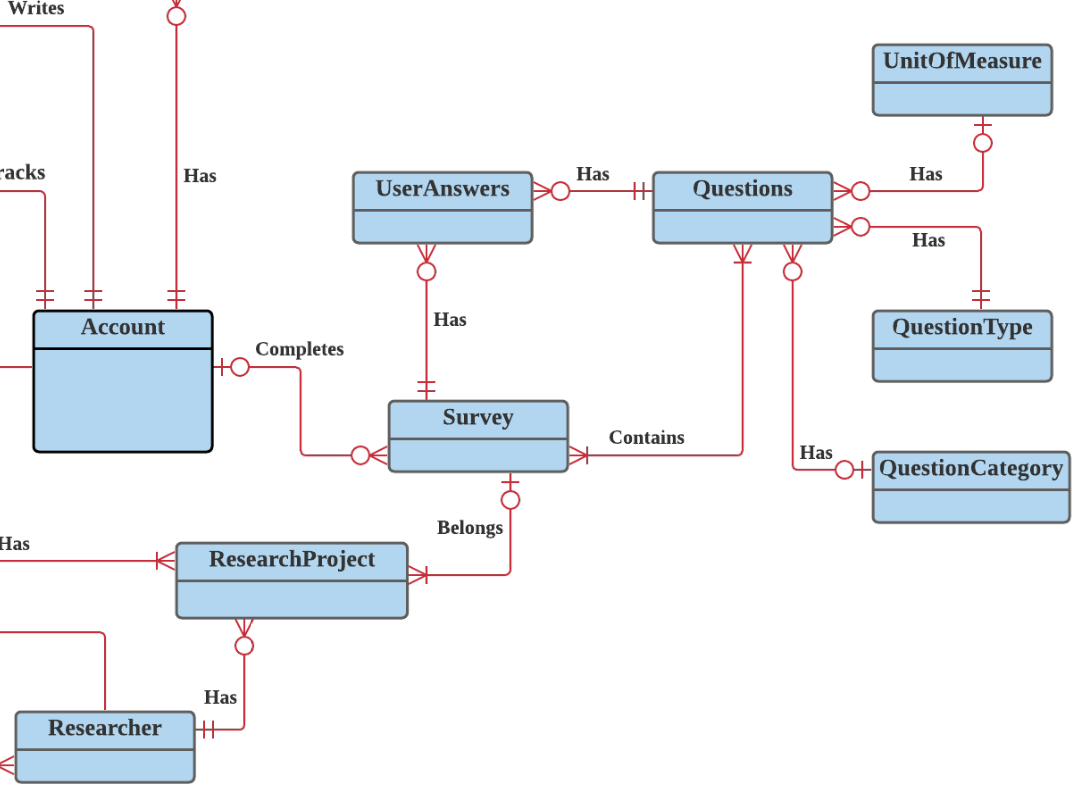
*Figure 1.2 Super and Sub-account types.*

**Figure 1.2** shows the logic associated with each account. Each account will contain user information. Each account can be a ‘Doctor’, ‘Researcher’, or ‘Patient’ sub-type, with the possibility that more sub-types and functionality will be added in the future. Each is associated with one active account.



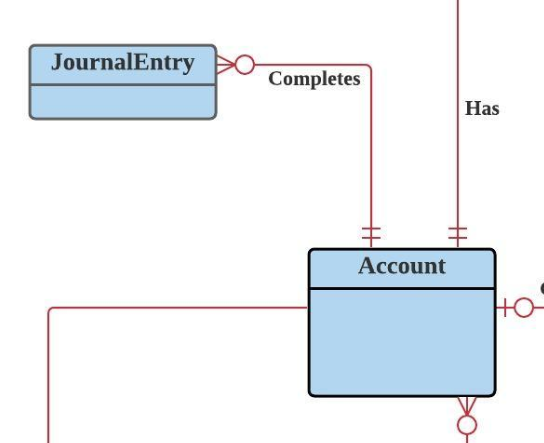
*Figure 1.3 The Health Tracker logic.*

**Figure 1.3** shows the portion of the ERD pertaining to the health tracker. Each user can track their food intake, weight, and exercise in the health tracker. Individual food items and exercise types are saved on the FoodItems and Exericise tables, respectively. Users can record multiple entries in the health tracker, each of which is associated with that particular user.



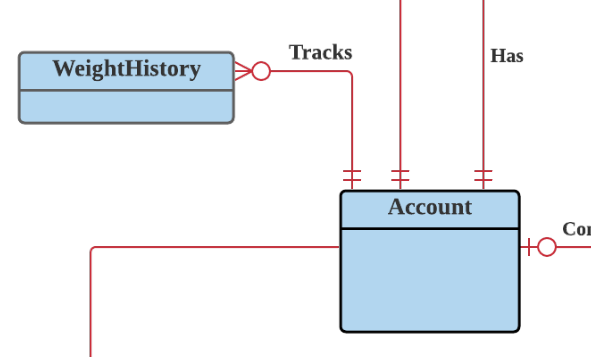
*Figure1.4 The Survey logic in the ERD.*

**Figure 1.4** pertains to the survey. Each user can record their health and wellness data through surveys. They can edit the default survey and give it a name for future use. When the user takes the survey, they will select which template (default or a named survey) they want to take. The template will be linked to specific questions on the SurveyQuestions table. When the user submits their survey, the question number, userID and their answer will be saved into the SurveyAnswers table. Each answer will need to be associated with a unique user, and also a question ID.



*Figure 1.5 ERD showing journal logic.*

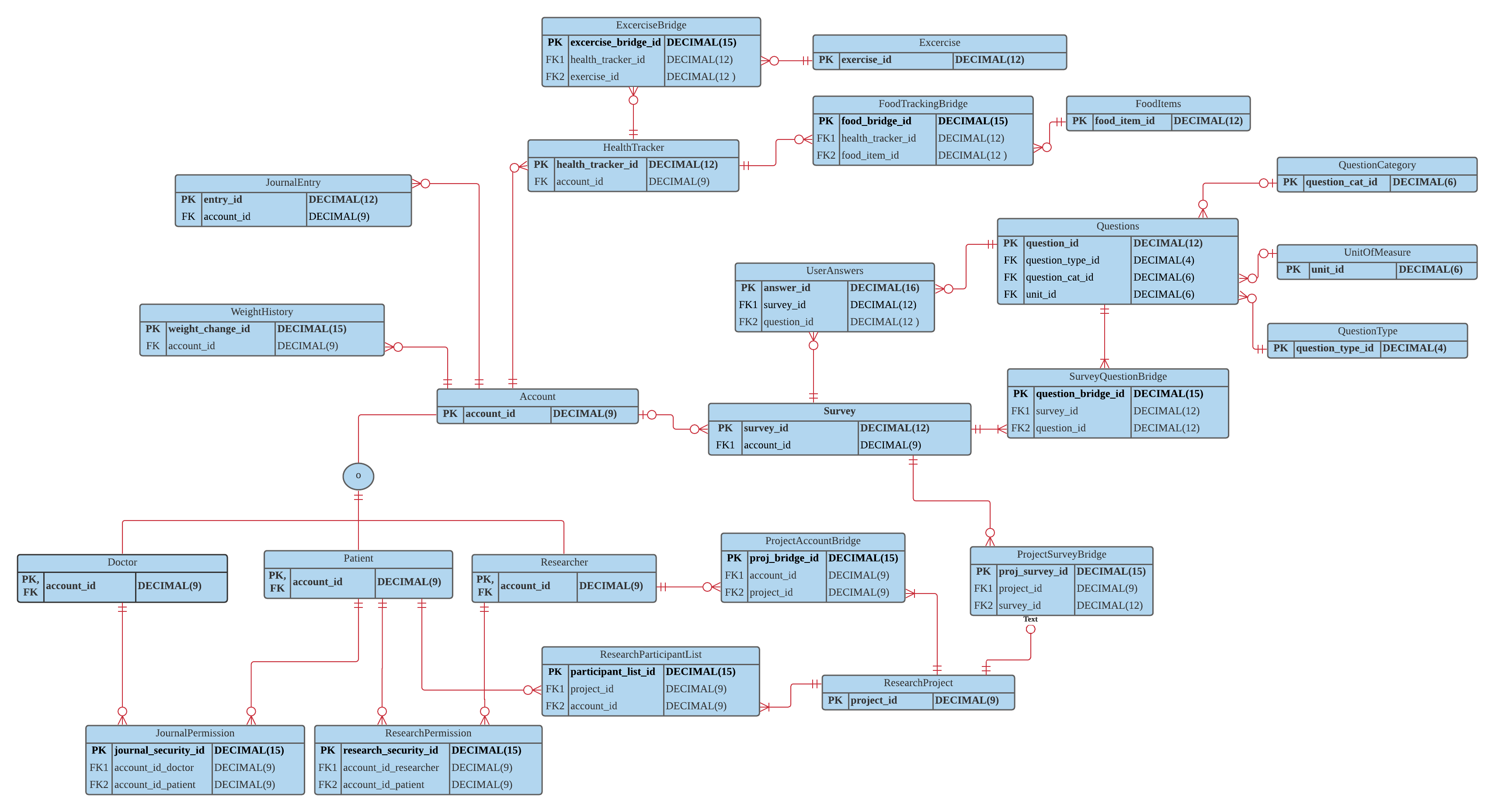
The final portion of the ERD is shown in figure 1.5 and pertains to the user making journal entries. Journal entries will be saved on the JournalEntry table. Each entry will be linked to one user, and the entries can be made public or private. The JournalEntry table can also be searched by accounts of ‘Reader’ type that want to read other user’s entries that are marked public.



*Figure 1.5.2 ERD showing WeightHistory and Account relationship.*

# Full DBMS Physical ERD

I started the process of creating a DMBS physical ERD following the steps outlined in the iteration 3 walkthrough. The first stage of my physical ERD is shown in Figure 1.6, with primary and foreign keys for each entity included.

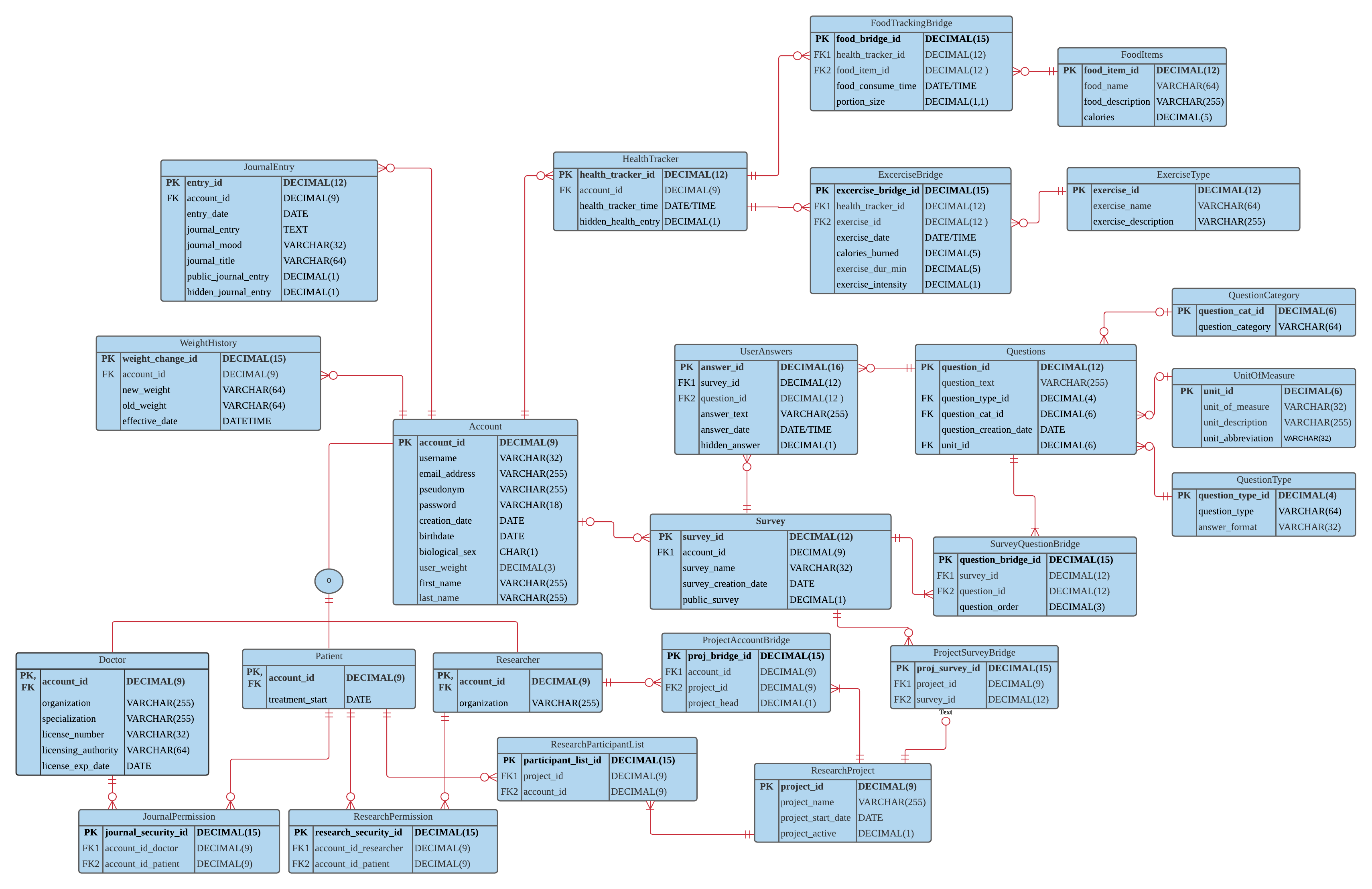
**

*Figure 1.6 The initial physical ERD, before including all fields. Updated 4/19/2021.*

You’ll notice some entities that were not in the conceptual ERD to assist with many-to-many relationships.

* The first is the FoodTrackingBridge, which includes the health\_tracker\_id and the food\_item\_id as foreign keys.
* The second is the SurveyQuestionBridge, which includes the survey\_id and question\_id as foreign keys.
* I added a ResearchParticipationList to link ResearchProjects with Accounts.
* Finally, I broke up the PermissionTable into two separated tables, 1. JournalPermission and 2. ResearchPermission. Each is a link between Doctor and Patient or Researcher and Patient, respectively.

Adding Attributes

*Figure 1.7 The final DBMS physical ERD for iteration 3. Updated 4/19/2021.*

The next step that I took was to go through the exercise that Lecturer Mansur used in his example. I mapped out each entity and field and included the format and justification for the format. This process helped me identify a few changes in my ERD, and I went back and changed the name and formats of a few different fields. The final version is included in the ERD in Figure 1.7, as well as in TABLE 1.12 below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Account** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| Account | birthdate | DATE | This will be a DATE format so that any changes in the DBMS vendor in the future will not cause issues. This will also help communication across systems that may be integrated with this schema. |
| Account | creation\_date | DATE | This will be a DATE format so that any changes in the DBMS vendor in the future will not cause issues. This will also help communication across systems that may be integrated with this schema. |
| Account | email\_address | VARCHAR(255) | This is the user's email address. It can be up to 255 characters in length. |
| Account | first\_name | VARCHAR(255) | This is the user's first name. We wil store up to 255 characters of it. |
| Account | last\_name | VARCHAR(255) | This is the user's last name. We wil store up to 255 characters of it. |
| Account | weight | DECIMAL(3,2) | This is the user's weight. No person can weigh more than 999 pounds, so we will store 3 places, plus 2 decimals for user's that want to be exact. |
| Account | username | VARCHAR(64) | This is the username that will show up associated with each journal entry, survey, and food tracking entry. Also, the user can use it to sign-in to their account. I choose 64 characters. |
| Account | pseudonym | VARCHAR(255) | This is the psuedonym that the user will publish their journals under. It can be changed periodically. It is VARCHAR(255) to leave room for phrases or sentences that some users may want to use as a psuedonym. |
| Account | password | VARCHAR(18) | I choose VARCHAR so that characters and numbers can both be used. A length of 18 will allow for a very secure password. |
| Account | biological\_sex | CHAR(1) | This will be either 'M' for male, 'F' for female, 'N' for "Prefer not to answer". |
| Account | account\_id | DECIMAL(9) | account\_id is 9 digits because that will for a large user-base if the website/APP becomes popular over time. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Doctor** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| Doctor | organization | VARCHAR255 | This is the hospital or organization that doctor belongs to. |
| Doctor | specialization | VARCHAR(255) | This is the doctor's specialization, kept for analysis purposes. 255 characters will be sufficient to document the speciaility. |
| Doctor | licensing\_authority | VARCHAR(64) | This is the licensing authority. |
| Doctor | license\_exp\_date | DATE | This is the date the doctor's license expires. |
| Doctor | license\_number | VARCHAR(32) | This is the doctor's license ID. It is 32 digits and VARCHAR to allow for a variety of ID types and sizes as licensing agencies don't use a uniform ID outside of the USA. |
| Doctor | account\_id | DECIMAL(9) | account\_id is 9 digits because that will for a large user-base if the website/APP becomes popular over time. |

|  |  |  |  |
| --- | --- | --- | --- |
| **FoodItems** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| FoodItems | food\_name | VARCHAR(64) | This will save the name of the food being tracked. 64 characters will give users flexibility with the name, while keeping storage to a minimum. |
| FoodItems | food\_description | VARCHAR(255) | This is a description attached to each food item. Allowing for 255 characters will give users the freedom to fill in details without compromising storage space. |
| FoodItems | food\_hidden | DECIMAL(1) | This is a binary 0/1, If this is marked 1, then the emtry is hidden from everyone other than the author, even if other user's have permission to read the author's entries (such as a doctor or researcher). |
| FoodItems | food\_item\_id | DECIMAL(12) | This number will identify the food item. 12 digits will allow many, many food items to be saved. |

|  |  |  |  |
| --- | --- | --- | --- |
| **FoodTrackingBridge** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| FoodTrackingBridge | food\_consume\_time | DATE/TIME | This will save the date and time that the user enters for each food item being consumed. |
| FoodTrackingBridge | calories | DECIMAL(5) | This will store the number of calories for each food item tracked. 5 digits keeps the calories within the realm of what is possible for calorie count per item. |
| FoodTrackingBridge | portion\_size | DECIMAL(1,1) | This is a decimal that allows users to track portion size as a multiple of 1. For instance, 1.5 size portion size, 2x portion size, .5x portion size. |
| FoodTrackingBridge | food\_item\_id | DECIMAL(12) | This number will identify the food item. 12 digits will allow many, many food items to be saved. |
| FoodTrackingBridge | food\_tracker\_id | DECIMAL(12) | This is a unique number that identifies each food tracking entry. |
| FoodTrackingBridge | food\_bridge\_id | DECIMAL(15) | This is the primary key for the FoodTrackingBridge. It is 15 digits to allow for all the combinations of users and food items that may be created. |

|  |  |  |  |
| --- | --- | --- | --- |
| **HealthTracker** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| HealthTracker | health\_tracker\_id | DECIMAL(12) | This is a unique number that identifies each health tracking entry. |
| HealthTracker | food\_tracker\_time | DATE/TIME | This will save the date and time that the entry is saved. |
| HealthTracker | account\_id | DECIMAL(9) | account\_id is 9 digits because that will for a large user-base if the website/APP becomes popular over time. |
| HealthTracker | hidden\_health\_entry | DECIMAL(1) | This is a binary 0/1, If this is marked 1, then the entryis hidden from everyone other than the author, even if other user's have permission to read the author's entries (such as a doctor or researcher). |

|  |  |  |  |
| --- | --- | --- | --- |
| **JournalEntry** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| JournalEntry | entry\_date | DATE | This will be a DATE format so that any changes in the DBMS vendor in the future will not cause issues. This will also help communication across systems that may be integrated with this schema. |
| JournalEntry | entry\_mood | VARCHAR(32) | This is the author's mood when making the entry, or the general mood of the writing. VARCHAR(32) will allow users to be creative with their mood while also minimizing storage needs. |
| JournalEntry | journal\_title | VARCHAR(64) | This is a title given to the journal entry by the author. 64 characters will allow users to be creative with their mood while also minimizing storage needs. |
| JournalEntry | account\_id | DECIMAL(9) | account\_id is 9 digits because that will for a large user-base if the website/APP becomes popular over time. |
| JournalEntry | hidden\_journal\_entry | DECIMAL(1) | This is a binary 0/1, If this is marked 1, then the journal entry is hidden from everyone other than the author, even if other user's have permission to read the author's entries (such as a doctor or researcher). |
| JournalEntry | journal\_entry | TEXT | This is the actual journal entry each user will enter. I choose TEXT so that entries can be comprehensive, and so that formatting can be saved. |
| JournalEntry | public\_journal\_entry | DECIMAL(1) | This is a binary 0/1, If this is marked 1, then the journal entry is made public and any other user can read it. If it is a 0, then only the author and the people the author has given explicit permission to (such as their doctor) can read this. |
| JournalEntry | entry\_id | DECIMAL(12) | This is a synthetic key that is 12 digits long. 12 digits will allow for an expansive number of unique entries. |

|  |  |  |  |
| --- | --- | --- | --- |
| **JournalPermission** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| JournalPermission | account\_id\_doctor | DECIMAL(9) | This is the account ID of the doctor that is receiving permission to read a patient's journal entries. |
| JournalPermission | account\_id\_patient | DECIMAL(9) | This is the accoun ID of the patient who is giving permission to a doctor to read their journal entries. |
| JournalPermission | journal\_security\_id | DECIMAL(15) | This is the PKID for the JournalPermission table. 15 digits will allow for all possible combinations of accounts now and in the future. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Patient** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| Patient | account\_id | DECIMAL(9) | account\_id is 9 digits because that will for a large user-base if the website/APP becomes popular over time. |
| Patient | treatment\_start | DATE | This is the date that the patient created a patient account. Their "treatment" start date. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Account Bridge** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| ProjectAccountBridge | project\_head | DECIMAL(1) | This is a binary 0/1. If it is a 1 then the account\_id on this record is the head (or a head) of the project. |
| ProjectAccountBridge | account\_id | DECIMAL(9) | This is the user's account ID. |
| ProjectAccountBridge | project\_id | DECIMAL(9) | This is the project ID. |
| ProjectAccountBridge | proj\_bridge\_id | DECIMAL(15) | This is the PKID for the ProjectAccountBridge. 15 digits will allow for all combinations of account and project ID now and in the future. |

|  |  |  |  |
| --- | --- | --- | --- |
| **ProjectSurveyBridge** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| ProjectSurveyBridge | project\_id | DECIMAL(9) | This is the project ID. |
| ProjectSurveyBridge | survey\_id | DECIMAL(12) | This is the Survey ID. |
| ProjectSurveyBridge | proj\_survey\_id | DECIMAL(15) | This is the PKID for the ProjectSurveyBridge. 15 Digits will allow for all cominbations of projects and survey IDs. |

|  |  |  |  |
| --- | --- | --- | --- |
| **QuestionCategory** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| QuestionCategory | question\_category | VARCHAR(64) | This is the type of category the question is. Examples include, "Health", "Weight", "Mental Health", "Spiritual Health", etc. |
| QuestionCategory | category\_description | VARCHAR(255) | This is a description of the question category. 255 characters will be sufficient to describe the category. |
| QuestionCategory | question\_cat\_id | DECIMAL(6) | This is the PKID for the question category. 6 digits is sufficient to remain unique for all types of categories. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Questions** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| Questions | question\_text | VARCHAR(255) | This will store the question being asked. 255 characters will allow for most reasonably phrased questions to be asked. |
| Questions | question\_cat\_id | VARCHAR(32) | This will be a place to save the category this question falls into such as "health", "Physical attributes", etc. 32 characters will be sufficient to indicate the category. |
| Questions | question\_type\_id | DECIMAL(4) | This will store the PKID for the QuestionType table. Four digits will be enough to store any possible question type. |
| Questions | unit\_id | DECIMAL(6) | This links to the UnitOfMeasure table. 6 digits is sufficient for the types of units that can be measured . |
| Questions | question\_id | DECIMAL(12) | This will store the question ID assigned to each question. 12 Digits will allow for the creation of many unique questions. |

|  |  |  |  |
| --- | --- | --- | --- |
| **QuestionType** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| QuestionType | question\_type | VARCHAR(64) | This is the type of question being asked, eg. "Likert 7", "Likert 5", "free-form", etc. This is 64 characters because that will allow user to be succinct, but also give freedom to be descriptive. |
| QuestionType | answer\_format | VARCHAR(32) | This is the type of question being asked, eg. "Likert 7", "Likert 5", "free-form", etc. This is 64 characters because that will allow user to be succinct, but also give freedom to be descriptive. |
| QuestionType | question\_type\_id | DECIMAL(4) | This will store the PKID for the QuestionType table. Four digits will be enough to store any possible question type. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Researcher** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| Researcher | account\_id | DECIMAL(9) | account\_id is 9 digits because that will for a large user-base if the website/APP becomes popular over time. |
| Researcher | Organization | VARCHAR(255) | This is VARCHAR(255) to allow for long organization names. |

|  |  |  |  |
| --- | --- | --- | --- |
| **ResearchParticipantList** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| ResearchParticipantList | project\_id | DECIMAL(9) | This is the ID number of the project. 9 digits will allow for all projects to be saved now and in the future without having to modify this field. |
| ResearchParticipantList | account\_id | DECIMAL(9) | This is the user's account ID. It is 9 digits to allow for many unique users to create accounts over time. |
| ResearchParticipantList | participant\_list\_id | DECIMAL(15) | This is the PKID for the ResearchParticipantList. 15 digits will allow a unique ID for all future cominations of project and account IDs. |

|  |  |  |  |
| --- | --- | --- | --- |
| **ResearchPermission** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| ResearchPermission | account\_id\_researcher | DECIMAL(9) | This is the account that has permission to read other user's accounts (for example this is a doctor account). This will be an account\_id, so the format is DECIMAL(9). |
| ResearchPermission | account\_id\_participant | DECIMAL(9) | This will be the account that is giving permission to other accounts to view their entries. This will be an account\_id, so the format is DECIMAL(9). |
| ResearchPermission | research\_security\_id | DECIMAL(15) | This is the PKID for the ResearchPermission table. 15 digits will allow for all possible combinations of accounts now and in the future. |

|  |  |  |  |
| --- | --- | --- | --- |
| **ResearchProject** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| ResearchProject | project\_name | VARCHAR(255) | This is the name of the project. 255 characters will allow for a robust name to be saved. |
| ResearchProject | project\_start\_date | DATE | This is the date the project starts. |
| ResearchProject | project\_active | DECIMAL(1) | This is a binary 0/1. If it is a 1, then the project is still active. |
| ResearchProject | project\_id | DECIMAL(9) | This is the ID number of the project. 9 digits will allow for all projects to be saved now and in the future without having to modify this field. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Survey** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| Survey | survey\_name | VARCHAR(64) | This is an optional title given to each survey by the author. 64 characters will allow users to be creative with their mood while also minimizing storage needs. |
| Survey | account\_id | DECIMAL(9) | account\_id is 9 digits because that will for a large user-base if the website/APP becomes popular over time. |
| Survey | survey\_id | DECIMAL(16) | This will store the identification # of each survey. Each survey consists of a list of questions that is assigned to a user. Answers are stored in the UserAnswers. I choose DECIMAL(16) so that many surveys can be taken over time if the user-base grows to a large number. |

|  |  |  |  |
| --- | --- | --- | --- |
| **SurveyQuestionBridge** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| SurveyQuestionBridge | question\_order | DECIMAL(3) | This is the order that the question will appear in on the user's survey. It is 3 digits because surveys should never be more than 1000 questions long. |
| SurveyQuestionBridge | question\_id | DECIMAL(12) | This will store the question ID assigned to each question. 12 Digits will allow for the creation of many unique questions. |
| SurveyQuestionBridge | survey\_id | DECIMAL(16) | This will store the identification # of each survey. Each survey consists of a list of questions that is assigned to a user. Answers are stored in the UserAnswers. I choose DECIMAL(16) so that many surveys can be taken over time if the user-base grows to a large number. |
| SurveyQuestionBridge | question\_bridge\_id | DECIMAL(15) | This is the PKID for the SurveyQuestionBridge. It is 15 digits to allow for many combinations of questions and survey IDs. |

|  |  |  |  |
| --- | --- | --- | --- |
| **UnitOfMeasure** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| UnitOfMeasure | unit\_of\_measure | VARCHAR(32) | This is where the unit of measurement is inidicated. 32 characters is sufficient for indicating the unit type. |
| UnitOfMeasure | unit\_description | VARCHAR(255) | This is a description of the unit of measure. 255 characters allows users to give additional details about that particular unit of measurement. |
| UnitOfMeasure | unit\_id | DECIMAL(6) | This is the PKID for the UnitOfMeasure entity. It is 6 digits because that is sufficient for all the unique types of measurements. |
| UnitOfMeasure | unit\_abbreviation | VARCHAR(32) | This stores the common abbreviation for the unit of measure, which will be used when filling out surveys. This is 32 characters because that is sufficient to for an abbreviation. |

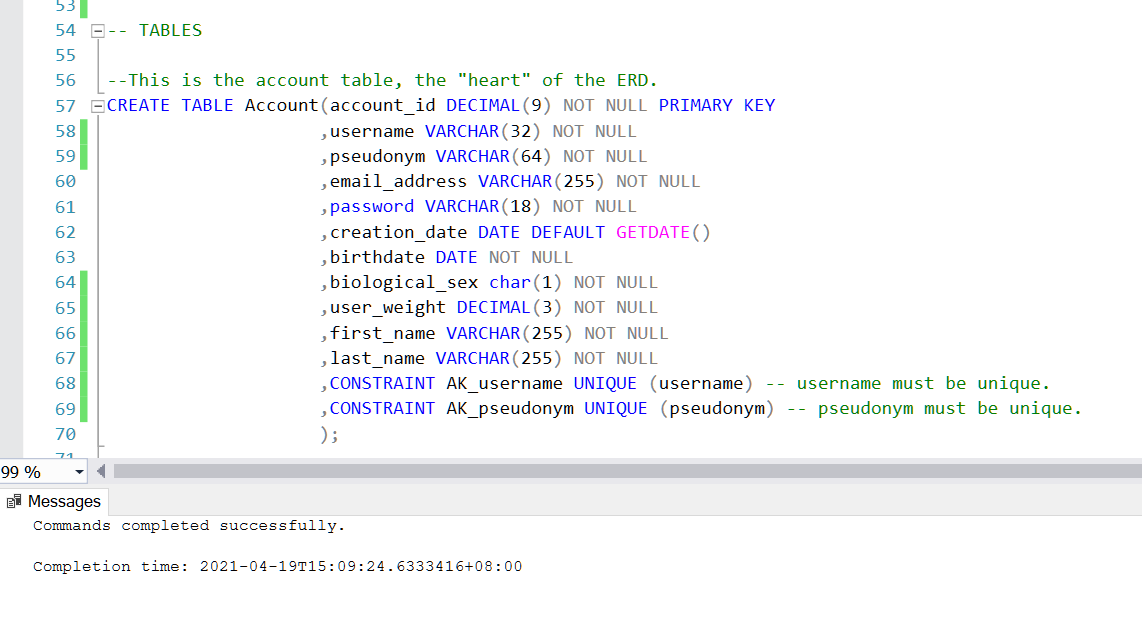
|  |  |  |  |
| --- | --- | --- | --- |
| **UserAnswers** | | | |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| UserAnswers | answer\_date | DATE | This is the date that the answer is saved into the system. This will be a DATE format so that any changes in the DBMS vendor in the future will not cause issues. This will also help communication across systems that may be integrated with this schema. |
| UserAnswers | answer\_text | VARCHAR(255) | This will store the user's response to the survey question. I choose 255 characters because this should be sufficient for all types of survey questions. |
| UserAnswers | answer\_id | DECIMAL(16) | This will store the ID assigned to each answer. 16 Digits will allow for many unique answers to be stored. |
| UserAnswers | hidden\_answer | DECIMAL(1) | This is a binary 0/1, If this is marked 1, then the answer is hidden from everyone other than the author, even if other user's have permission to read the author's entries (such as a doctor or researcher). |
| UserAnswers | question\_id | DECIMAL(12) | This will store the question ID assigned to each question. 12 Digits will allow for the creation of many unique questions. |
| UserAnswers | survey\_id | DECIMAL(16) | This will store the identification # of each survey. Each survey consists of a list of questions that is assigned to a user. Answers are stored in the UserAnswers. I choose DECIMAL(16) so that many surveys can be taken over time if the user-base grows to a large number. |

*Table 1.14 Entities and their fields.*

## Normalization

The final process that I went through was to look for dependencies and redundancies in the physical ERD. I went through each table in the previous section and verified that there are no redundancies. I did not identify any redundancies that exist in the current version of the ERD (figure 1.7). Going forward from this point, it should be possible to create the tables and relationships in SQL Server using the physical ERD as mapped.

SQL Script, Table and Sequence Creation



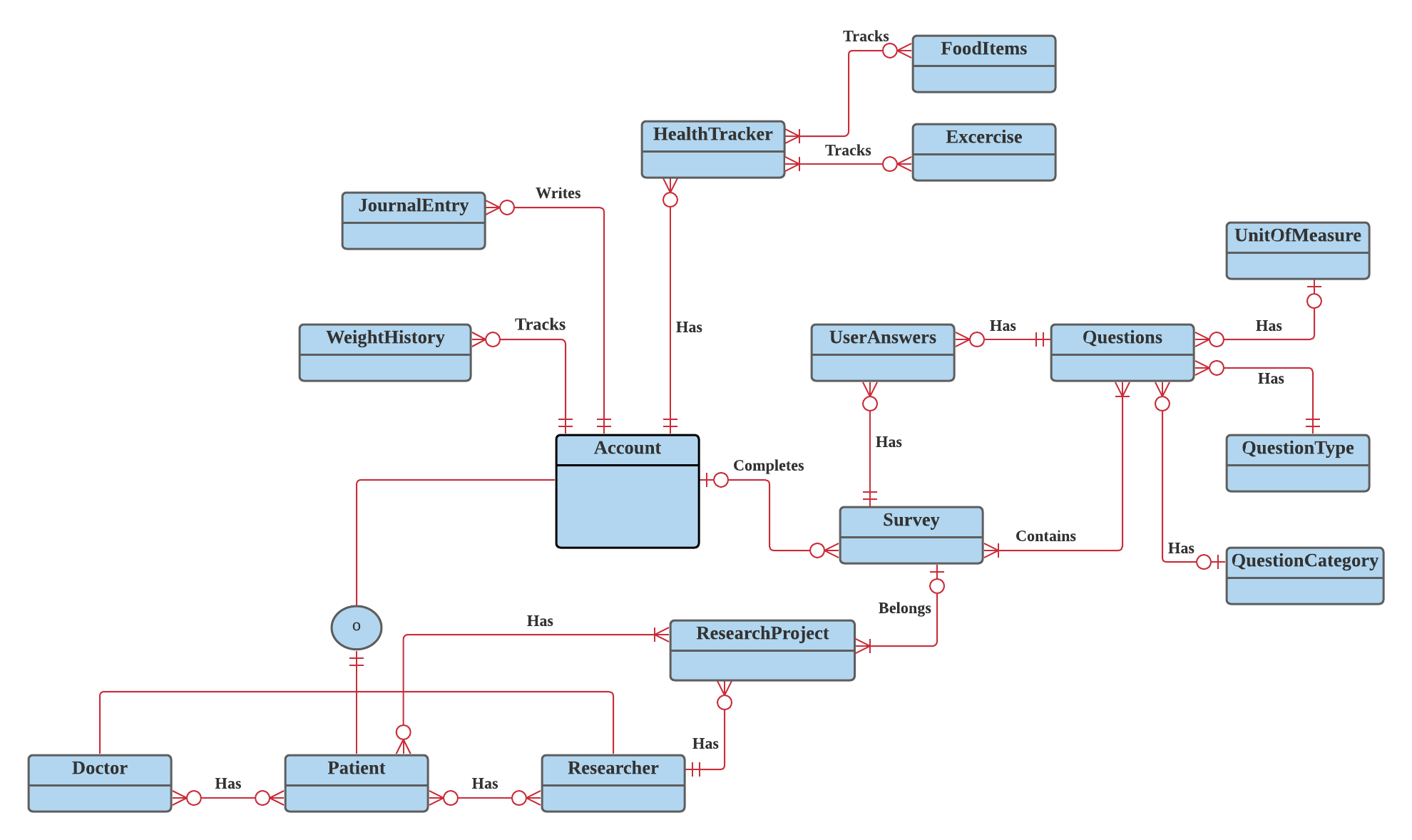
*Figure 1.8 Creating my tables and sequences using CS669\_TableCreation\_Ascher.sql*

# History Table and Trigger

In this section, I identify a field to create a history table and trigger off of.

One field that many users may be interested in tracking is the weight field on the Account entity.

To assist with tracking these changes, I’ve added a WeightHistory table that includes the fields laid out in Table 1.14 and Figure 1.14. The conceptual ERD is updated in Figure 1.9.

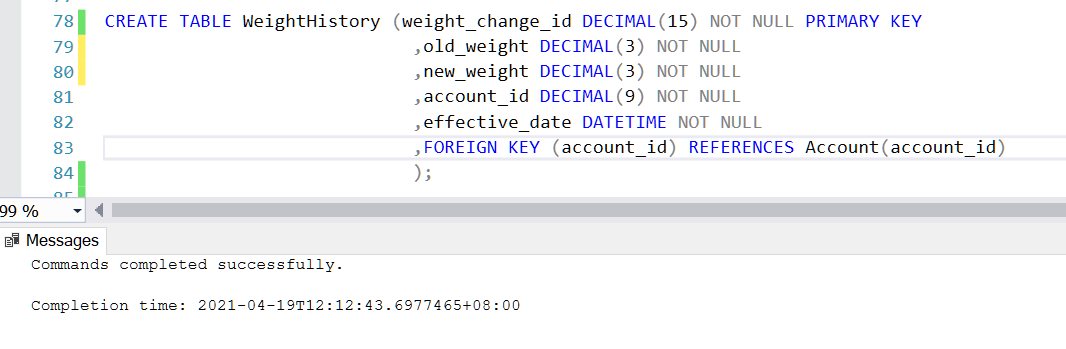


*Figure 1.9 Updated Conceptual ERD.*

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| weight\_change\_id | This is the PKID for the username History table. It is DECIMAL(15) to allow for the possibility of many users making many changes over time. |
| old\_weight | This is the old weight associated with the user. It is DECIMAL (3). |
| new\_weight | This is the new weight associated with the user. It is DECIMAL (3). |
| account\_id | This is the account\_id associated with the change. It is DECIMAL(9) just like account\_id on the Account table. |
| effective\_date | This is the DATETIME that the change takes place. |

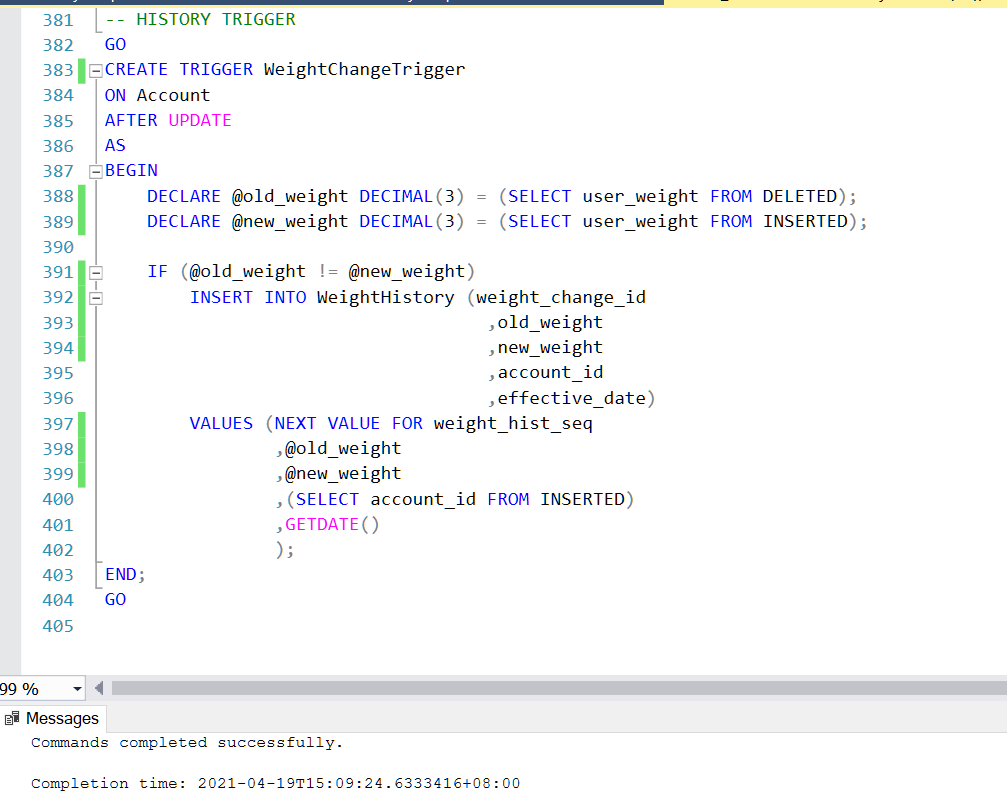
*TABLE 1.15 The fields for the UsernameHistory entity.*

I added the following SQL into the SQL script in order to create the table.



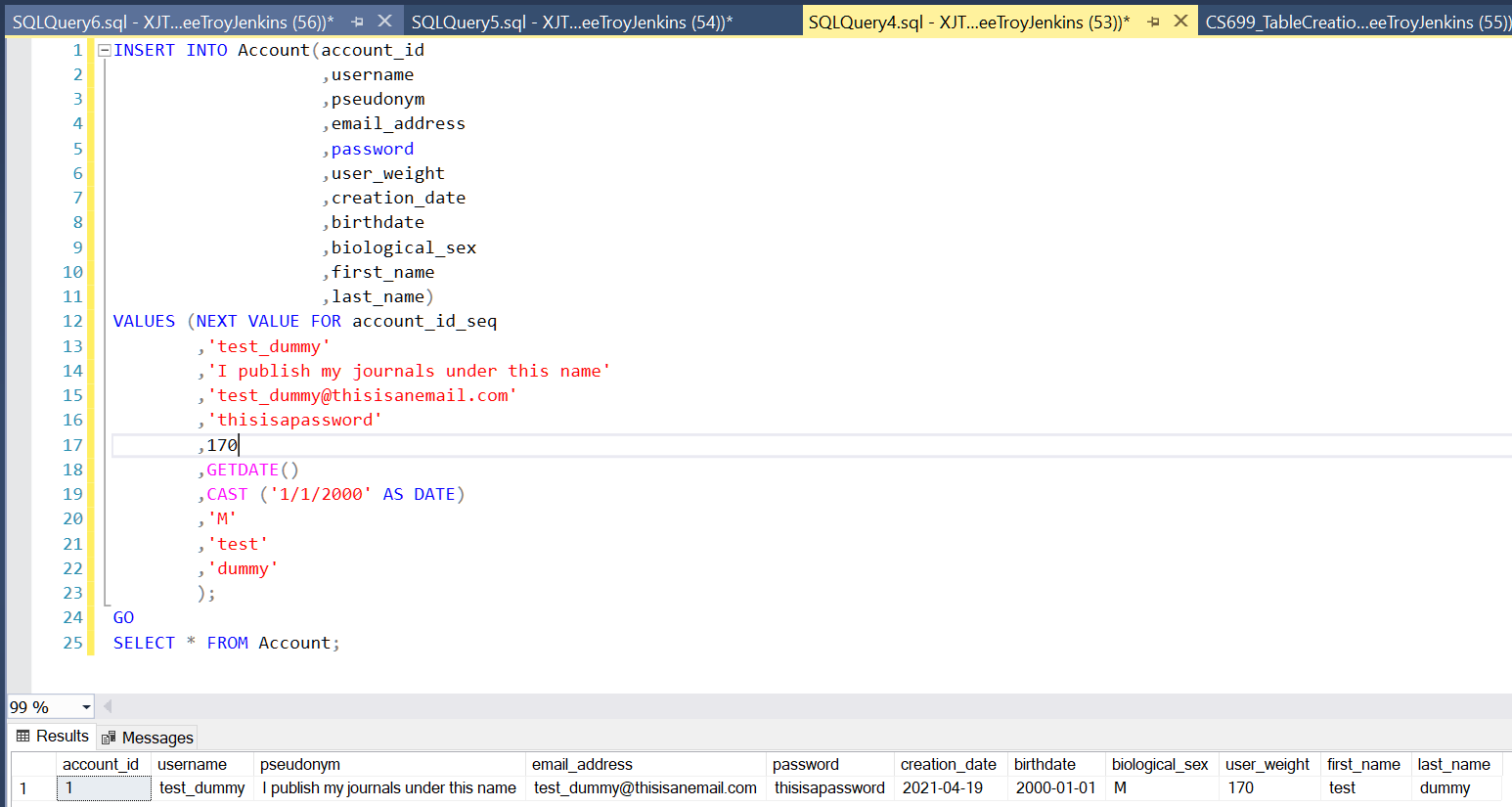
*Figure 1.10 SQL creating the WeightHistory entity.*

After adding the table, I added a trigger that would capture any changes to the weight field on the Account table.

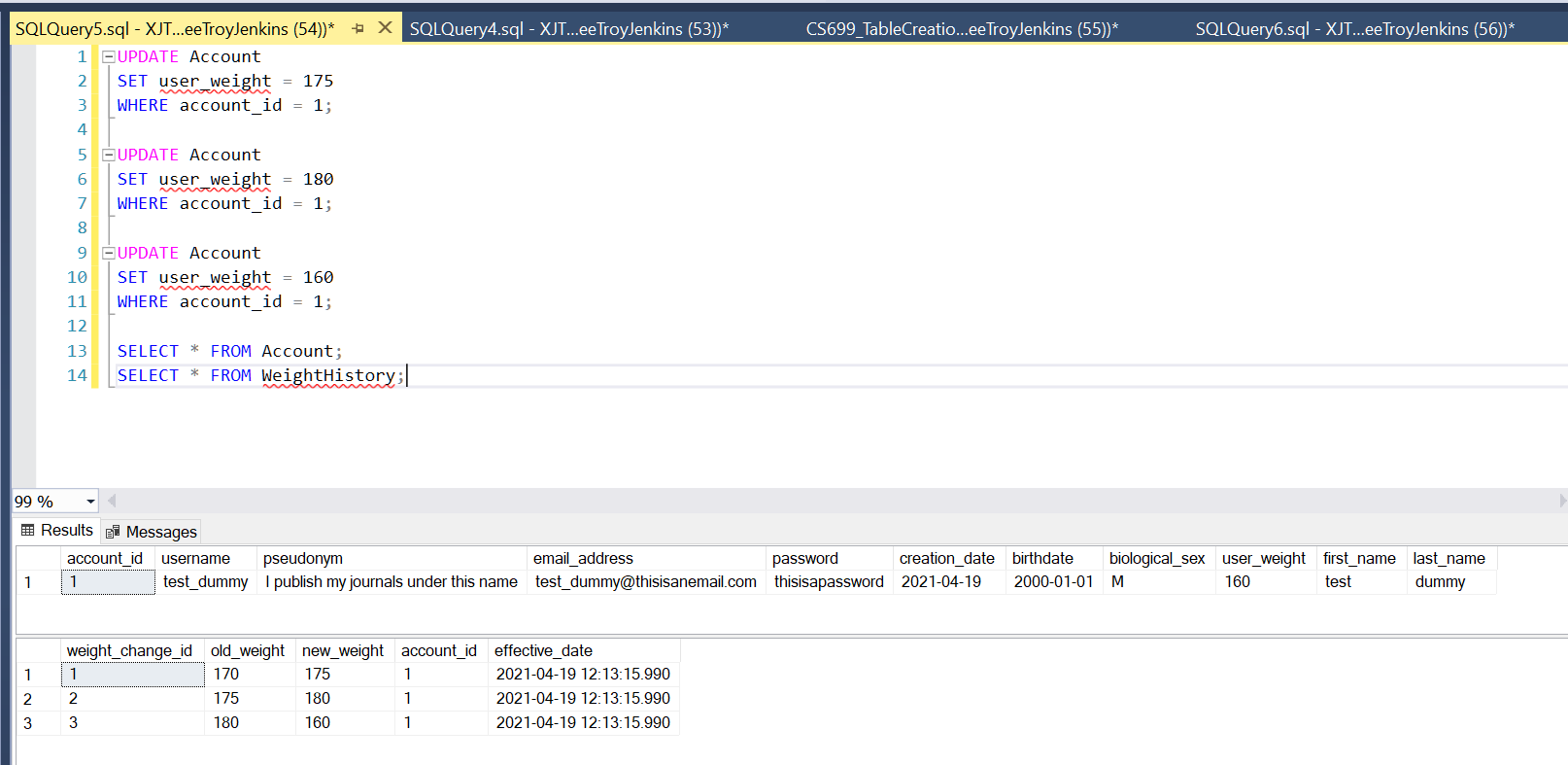


*Figure 1.11 SQL creating the WeightHistory trigger.*

I tested the trigger by adding a user and changing their weight threetimes. My expected result was that the last change to the username would reflect on the Account table while the three changes to the weight would also be saved into the WeightHistory table. As you can see in Figure 1.12, the test was successful. This indicates that the trigger works correctly.

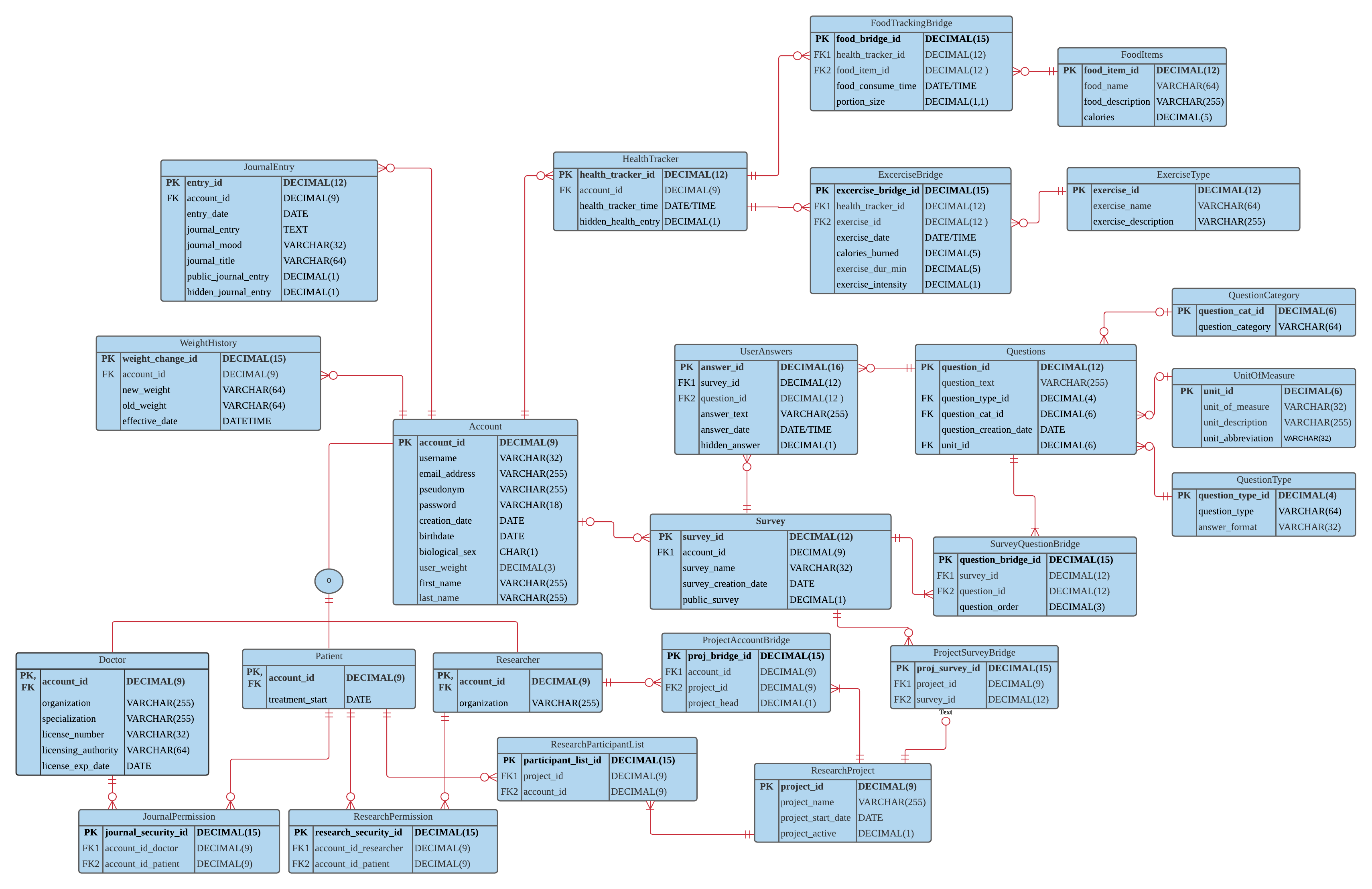


*Figure 1.12 Creation of the first account in the WellnessTracker Database.*



*Figure 1.13 Testing the WeightChangeTrigger.*

Finally, after adding the WeightHistory table to the DBMS Physical ERD, I’ve included a final version here in Figure 1.14.



*Figure 1.14 The updated DBMS Physical ERD.*

# Index Identification and Creations

During my analysis to determine which fields to index, I identified the following primary and foreign key fields.

Primary Keys

* Account.account\_id
* JournallPermission.journal\_security\_id
* Doctor.account\_id
* FoodItems.food\_item\_id
* HealthTracker.health\_tracker\_id
* FoodTrackingBridge.food\_bridge\_id
* ExcerciseBridge.health\_tracker\_id
* ExcerciseBridge.excercise\_id
* JournalEntry.entry\_id
* Patient.account\_id
* ProjectAccountBridge.proj\_bridge\_id
* ProjectSurveyBridge.proj\_survey\_id
* QuestionCategory.question\_cat\_id
* Questions.question\_id
* QuestionType.question\_type\_id
* Researcher.account\_id
* ResearchParticipantList.participant\_list\_id
* ResearchPermission.research\_security\_id
* ResearchProject.project\_id
* Survey.survey\_id
* SurveyQuestionBridge.question\_bridge\_id
* UnitOfMeasure.unit\_id
* UserAnswers.answer\_id
* WeightHistory.weight\_change\_id

Foreign Keys

I then identified the foreign keys in my table, and evaluated if they should be indexed with a unique or non-unique index.

|  |  |  |
| --- | --- | --- |
| **FieldName** | **Unique or Duplicating** | **Description** |
| JournalPermission.account\_id\_doctor | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| JournalPermission.account\_id\_patient | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| HealthTracker.account\_id | Not Unique | This account ID will be assigned to each health tracker entry. It will repeat every time the user creates an entry. |
| FoodTrackingBridge.food\_item\_id | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| FoodTrackingBridge.food\_tracker\_id | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| ExcerciseBridge.health\_tracker\_id | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| ExcerciseBridge.excercise\_id | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| JournalEntry.account\_id | Not Unique | This account ID will be assigned to each journal entry. It will repeat every time the user creates an entry. |
| ProjectAccountBridge.account\_id | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| ProjectAccountBridge.project\_id | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| ProjectSurveyBridge.project\_id | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| ProjectSurveyBridge.survey\_id | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| Questions.question\_cat\_id | Not Unique | This id will repeat many times on the Questions table. It is not unique. |
| Questions.unit\_id | Not Unique | This value will repeat multiple times on the Questions table. |
| ResearchParticipantList.project\_id | Not Unique | This value will repeat for every participant that is a part of the project. It is not unique. |
| ResearchParticipantList.account\_id | Not Unique | This value may repeat if an account is a part of multiple projects. It is not unique. |
| ResearchPermission.account\_id\_researcher | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| ResearchPermission.account\_id\_patient | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| Survey.account\_id | Not Unique | This value will repeat every time a user takes a survey. It is not unique. |
| SurveyQuestionBridge.question\_id | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| SurveyQuestionBridge.survey\_id | Not Unique | This is an FK on a bridging table, and may be repeated many times. It is not unique. |
| UserAnswers.question\_id | Not Unique | Questions can be repeated if users take the same survey on multiple days, and also used on multiple surveys. It is not unique. |
| UserAnswers.survey\_id | Not Unique | This value will repeat on the UserAnswers multiple times. It is not unique. |
| WeightHistory.account\_id | Not Unique | This value will repeat on the WeightHistory table as many times as a user updates their weight. |

*Table 1.16 Foreign Keys identified for indexes.*

Unique Indexes

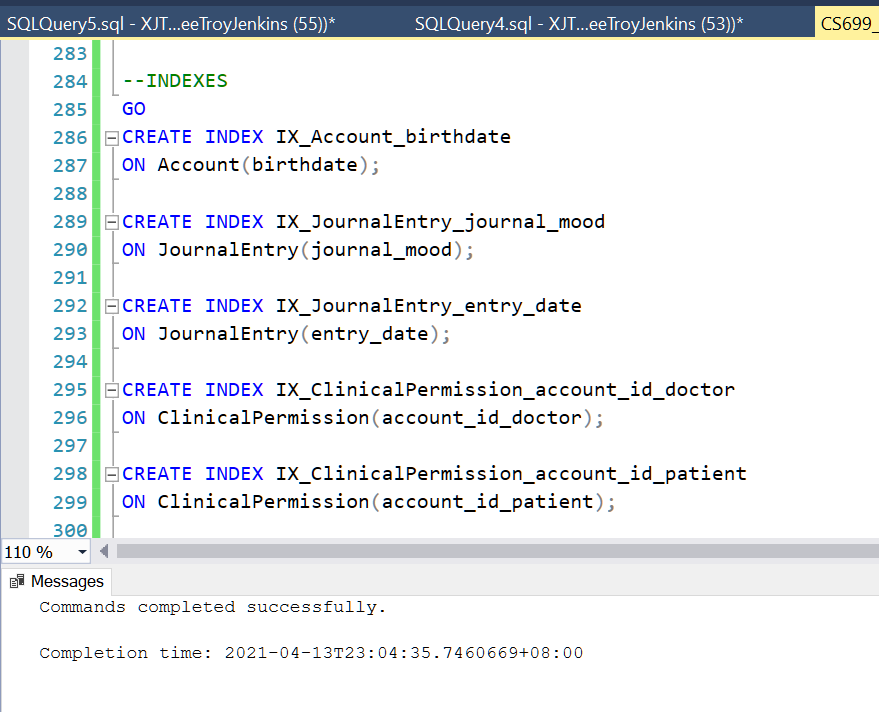
In addition to the Foreign Keys, I identified many fields that would make good candidates for indexes. I choose the following three because I think they will be some of the most common fields used in the WHERE phrase in queries. Account.birthdate will be used when researchers are searching for data from participants of a particular age or age range. JournalEntry.entry\_date will be used in queries that parse journal entries by date. For instance, many users that are choosing to read public entries will use the date function to search for entries. Finally, JournalEntry.entry\_mood is a good candidate for indexing because doctors can search for the mood of their patients and compare it to other data points that are available (such as date).

|  |  |  |
| --- | --- | --- |
| **FieldName** | **Unique or Duplicating** | **Description** |
| Account.birthdate | Y - Not Unique | Many users may share a birthdate, so this index is not unique. |
| JournalEntry.entry\_date | Y - Not Unique | Multiple users may make journal entries on the same date, so this index will not be unique. |
| JournalEntry.entry\_mood | Y - Not Unique | Users may choose the same mood when saving their journal entry. It is not unique. |

*Table 1.17 Fields that have been identified for indexes.*

Index Creation

I created indexes for all of the foreign keys as well as the three fields I identified in Table 1.15. The full SQL is in my SQL script, but here is screenshot of the three uniquely chosen fields and the first couple foreign key indexes.



*Figure 1.15 A screenshot of some of the indexes created for the Health and Wellness database.*

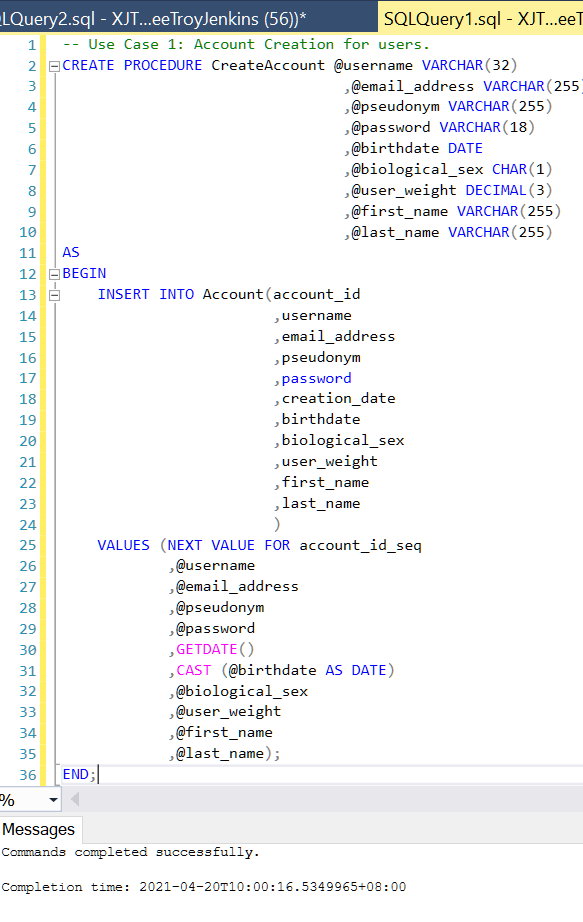
# Stored Procedure Execution and Explanations

In this section, I created procedures to execute my first three use cases. They are broken down by case, and include before and after pictures of each table.

## Use Case 1: Account Creation for users.

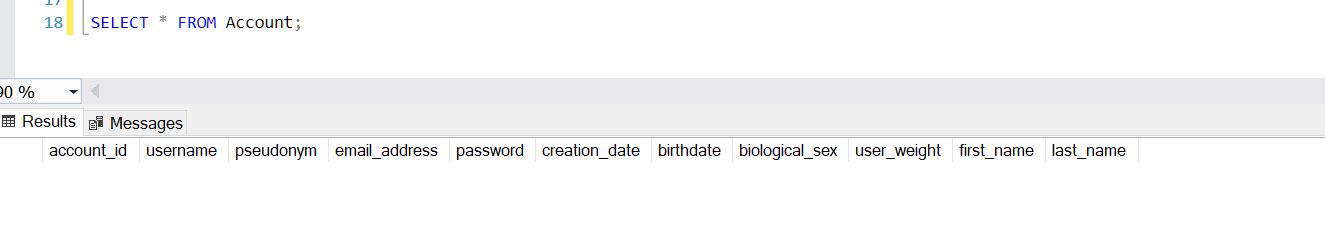
Steps:

1. User logs onto website.
2. User clicks, “Create Account”.
3. Website loads the “Account Creation” form.
4. User enters a username, password, and their email address.
5. Account is created and the data is stored in the database along with the date.



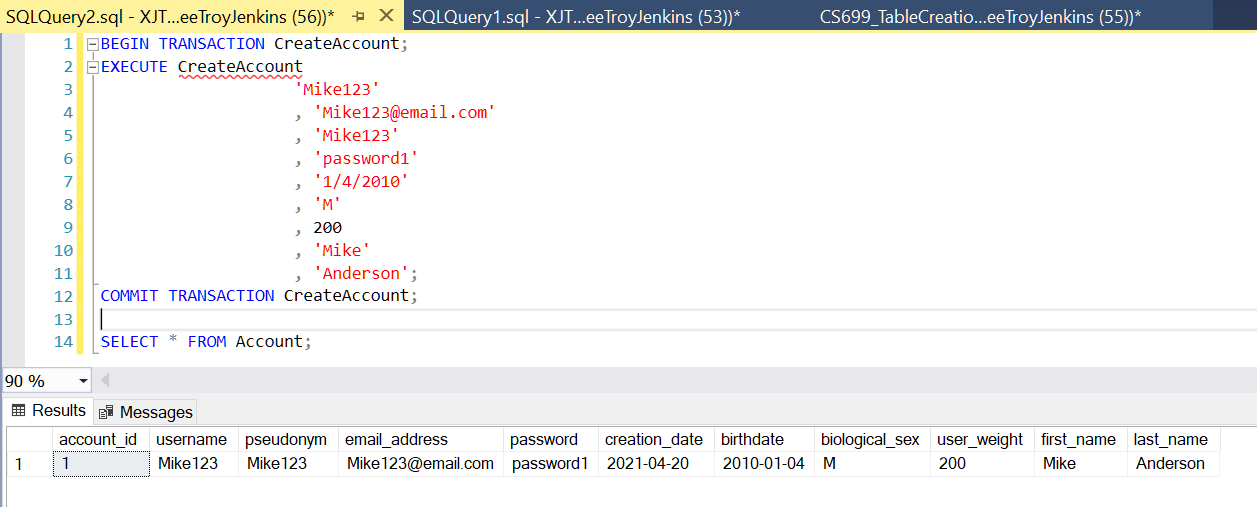
*Figure 1.16 the* CreateAccount *procedure.*

Here you can see that the procedure has been created successfully. All fields are manually entered except account\_id and creation\_date. Account\_id is generated using the account\_id\_seq, and the creation\_date is populated using the GETDATE() command.



*Figure 1.17 the* Account *table before executing the procedure.*

Figure 1.17 shows that the Account table is blank.



*Figure 1.18 The* Account *table after executing the procedure.*

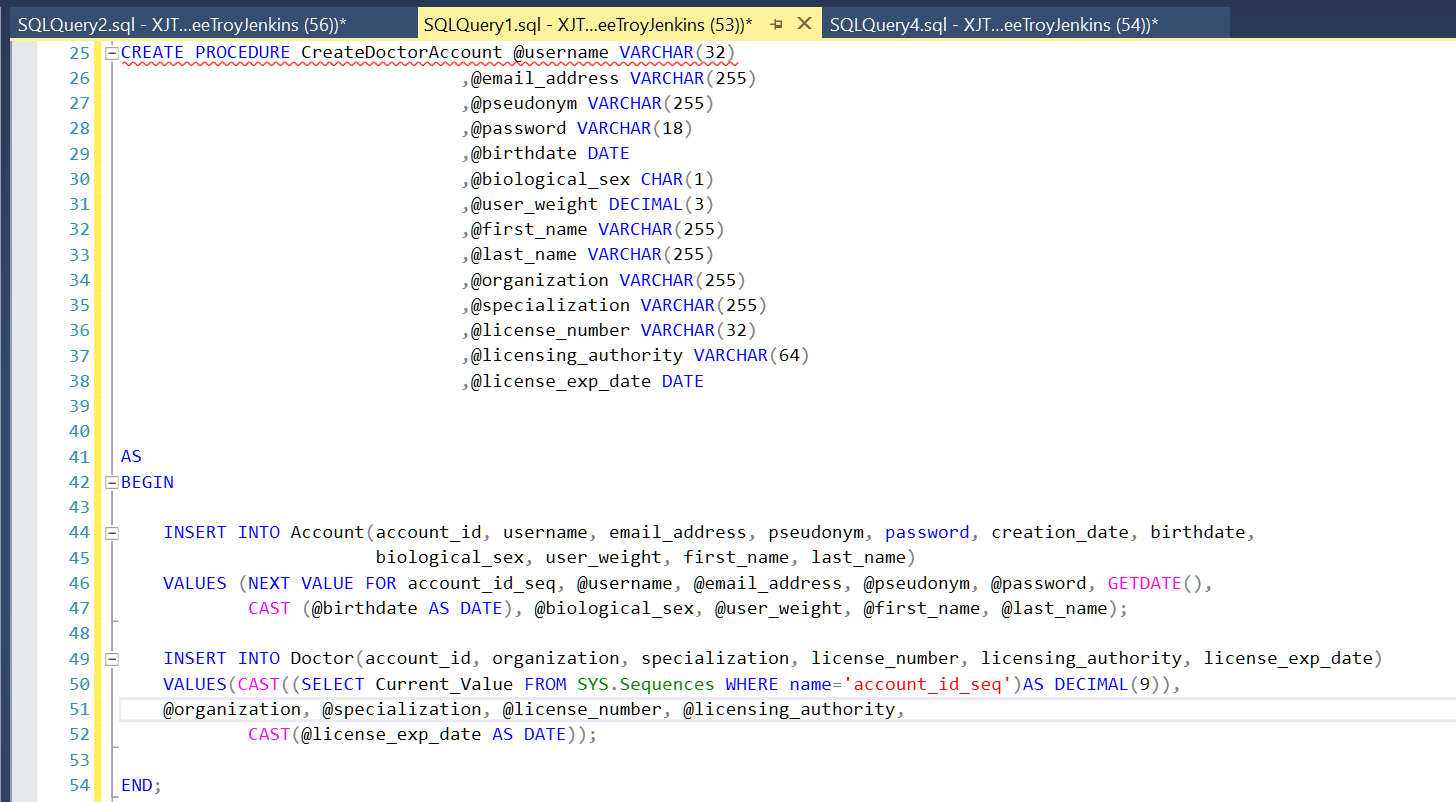
After the CreateAccount procedure is executed, the user “Mike123” is created. Figure 1.18 shows the account table with all fields populated for this user. Date and account\_id were populated successfully.

## Use Case 2: Account Creation for Doctors and Researchers.

Steps:

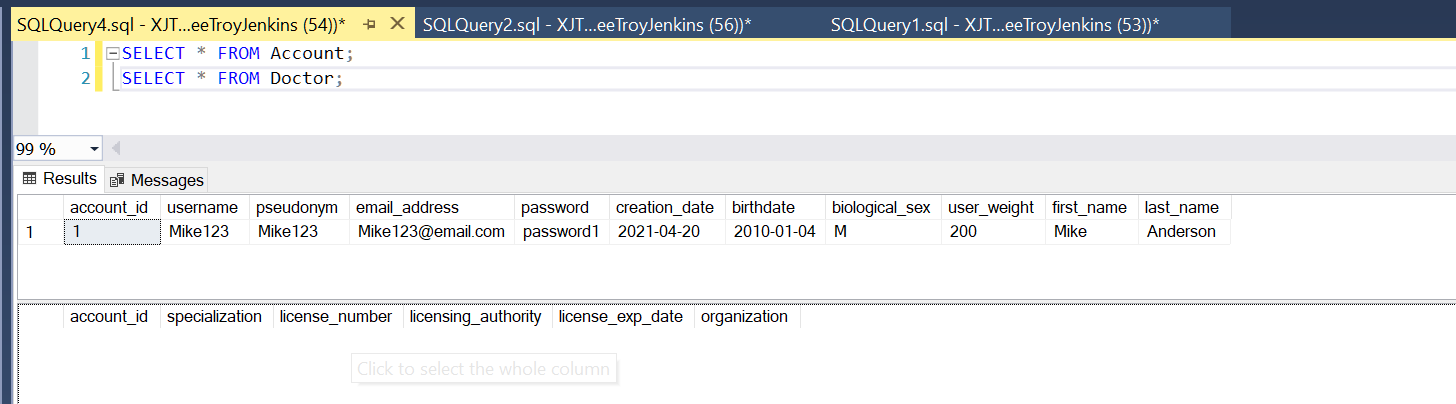
1. User logs onto website.
2. User clicks, “Create Account”.
3. Website loads the “Account Creation” form.
4. User enters a username, password, and their email address.
5. User selects check box for either ’Doctor’ or ‘Researcher’.
6. Account is created and the data is stored in the database along with the date.

For Use Case 2, the CreateDoctorAccount procedure is created and executed. This procedure was a little more complicated because the account\_id\_seq value is used twice. I used NEXT VALUE FOR on the first insertion, and then wrote a sub-query to pull the current value on the second insertion. Figures 1.19 – 1.21 show the creation and execution of the procedure to add values to the Doctor and Account tables.



*Figure 1.19 Creating the procedure.*

Figure 1.20 shows that the Doctor entity is blank.

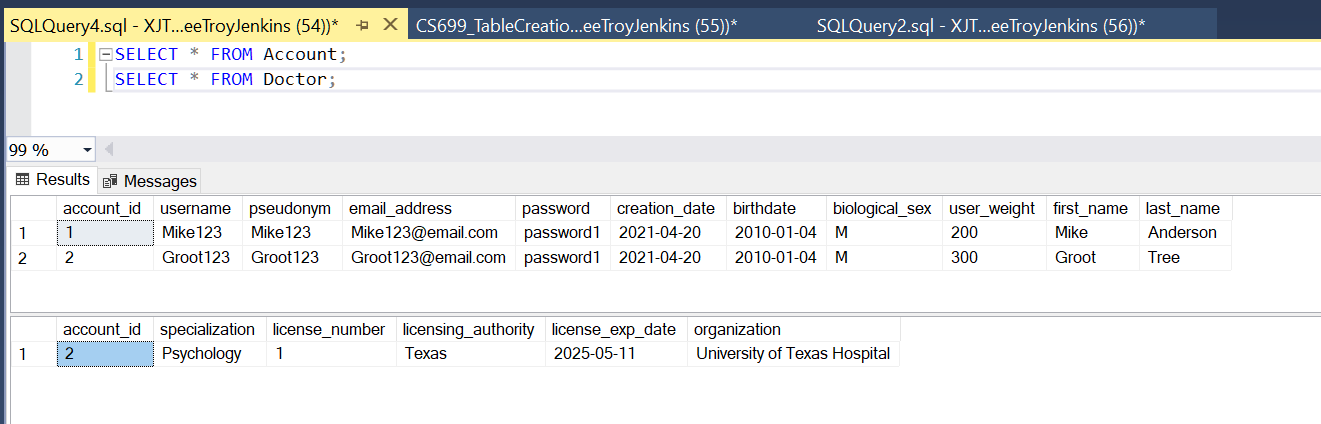


*Figure 1.20 The* Account *and* Doctor *table before executing the procedure.*



*Figure 1.21 executing the procedure.*

After the procedure is executed (Figure 1.21) the Account and Doctor tables update correctly.



*Figure 1.22 The* Account *and* Doctor *table after executing the procedure.*

## Use Case 3: Creating a Journal Entry

Steps:

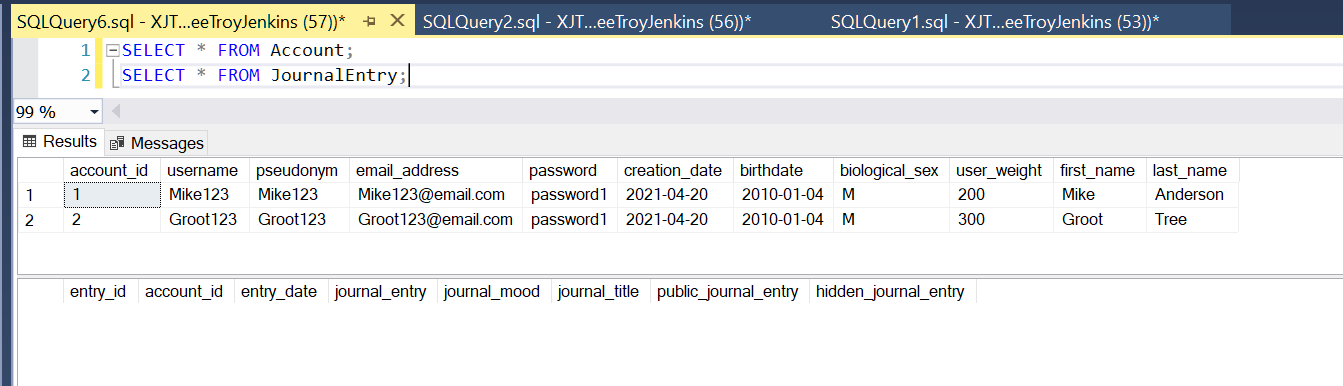
1. User opens the journal homepage on the website or APP.
2. User selects Make a Journal Entry and is taken to the journaling page.
3. User enters a title into the “JournalTitle” field.
4. User enters their journal entry into the JournalEntry field, checks a checkbox to indicate if the entry is public or private, and submits the entry.

To update journal entries I created the AddJournalEntry procedure. This procedure will execute when users submit their journal entry on the website. All fields are manually entered except for Journal.entry\_id and Journal.entry\_date. Those fields are populated using a sequence, and the GETDATE command, respectively.

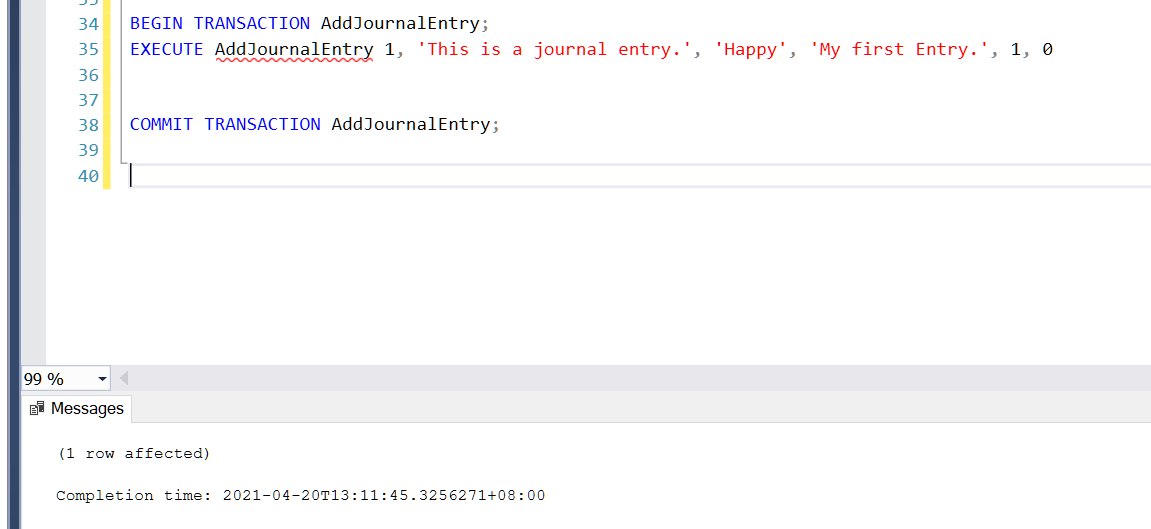
Figures 1.22 – 1.25 show the creation and execution of the procedure.



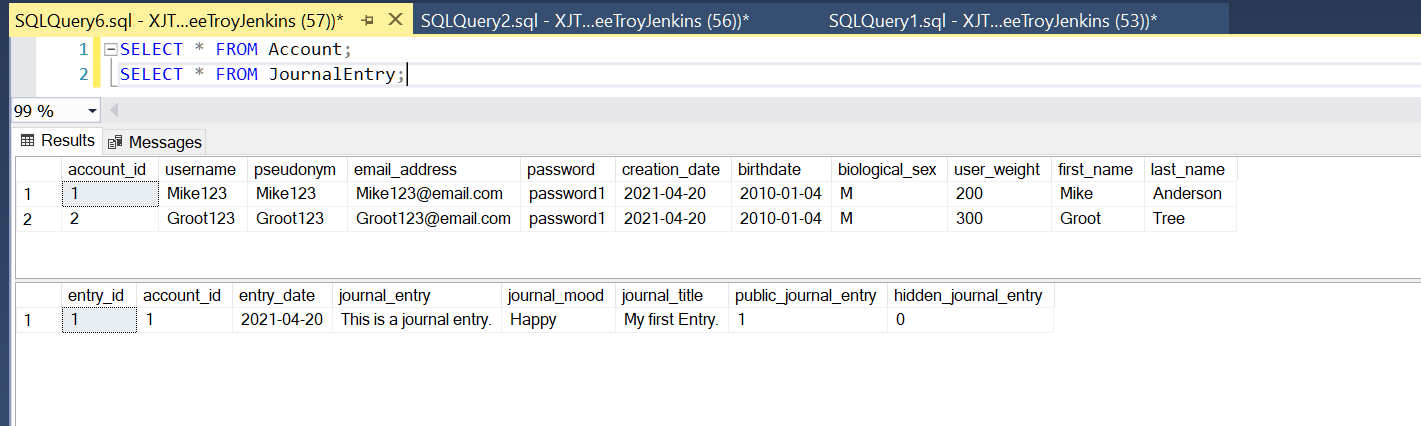
*Figure 1.23 Creating the* AddJournalEntry *procedure.*



*Figure 1.24 The* Account *and* Doctor *table before executing* AddJournalEntry*.*



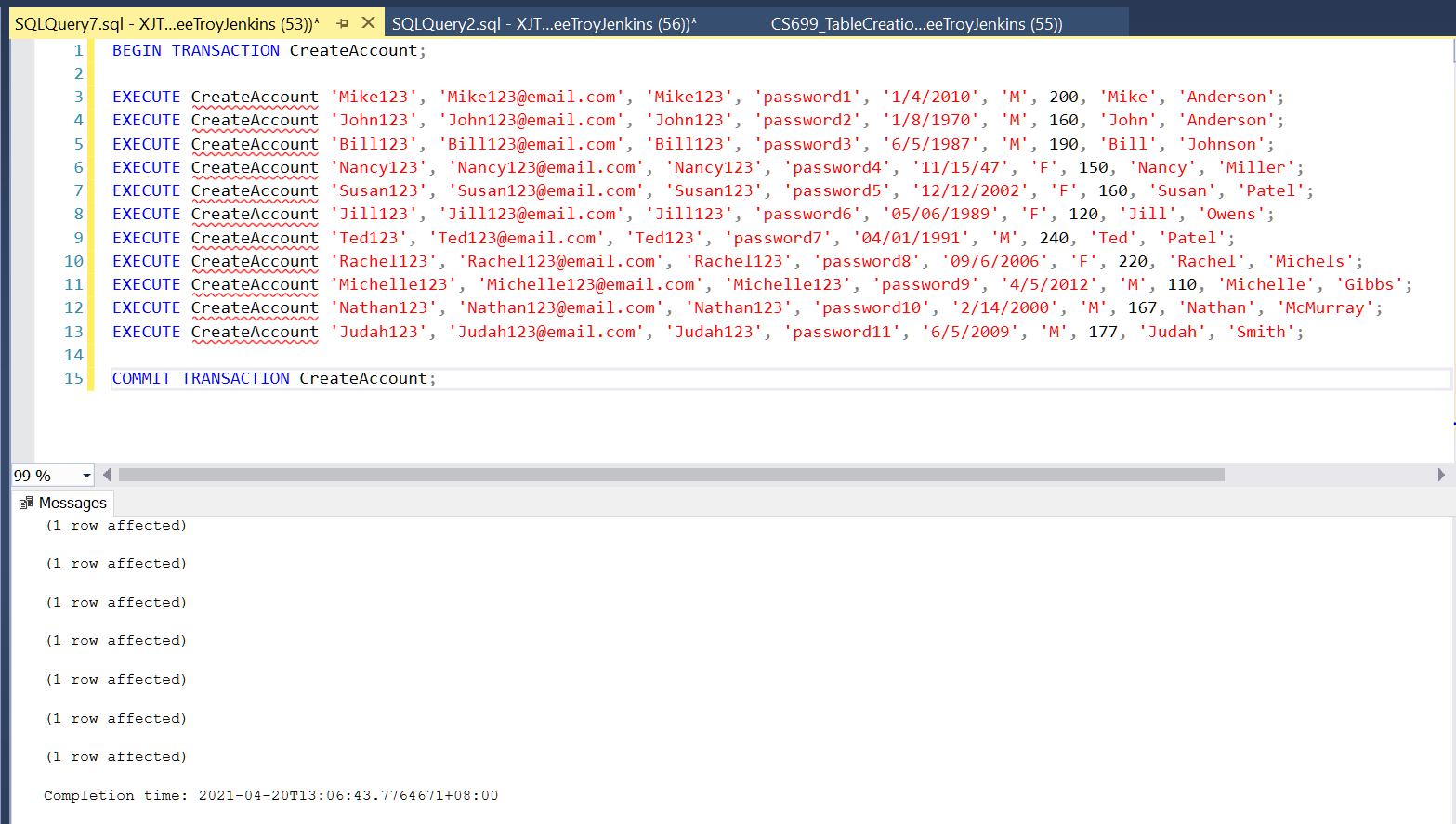
*Figure 1.25 executing* AddJournalEntry*.*



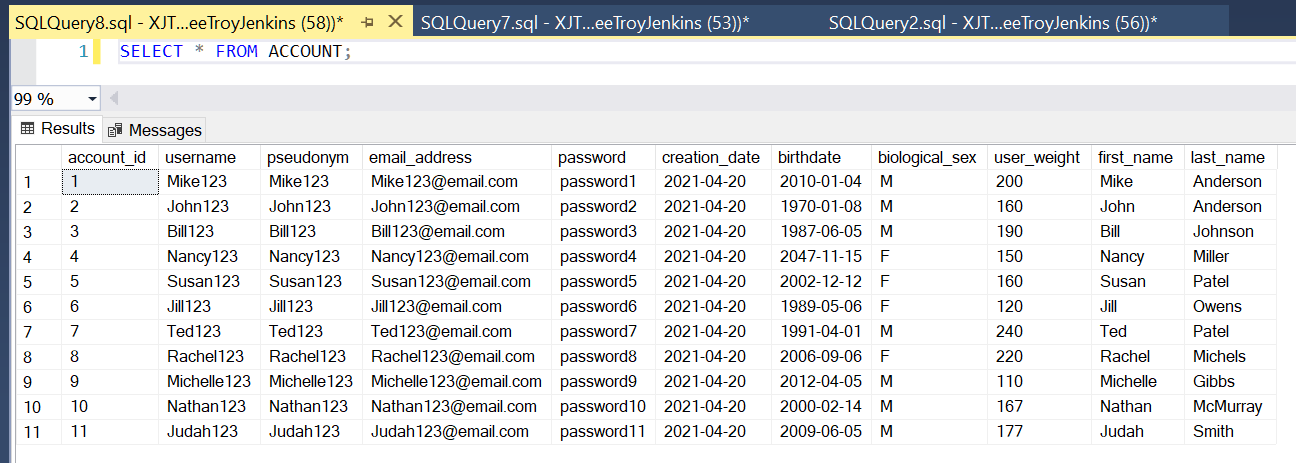
*Figure 1.26 The* Account *and* JournalEntry *tables after executing* AddJournalEntry*.*

Inserting Data

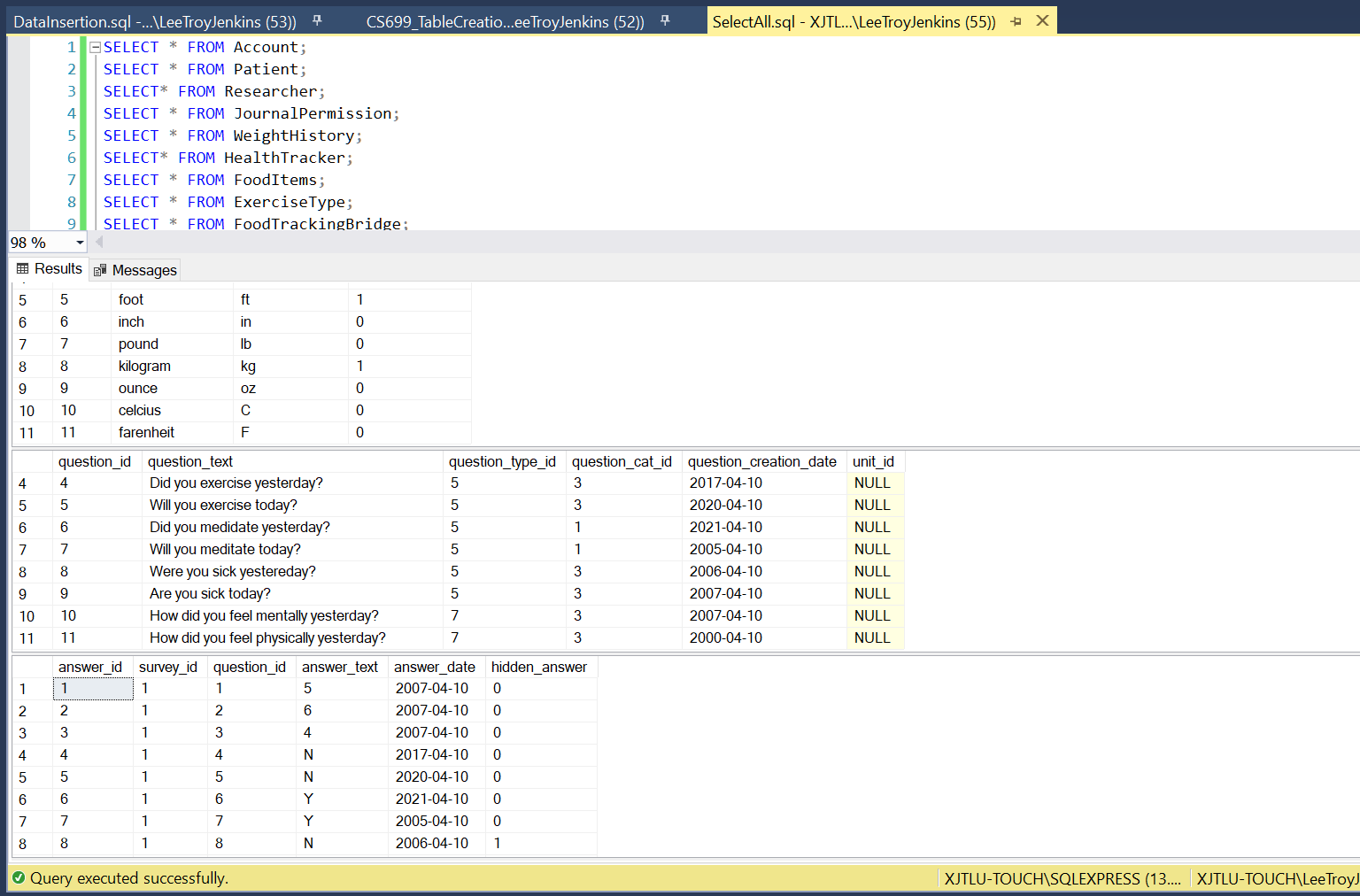
After the initial three procedures were created, I continued on and created a procedure for every table that isn’t a linking/bridging table in the database. I’ve included a few screenshots here as a demonstration, but the full SQL for the procedures is in the SQL table creation file, while the actual insertions are in the datainsertion.sql file.



*Figure 1.27 Inserting data using the CreateAccount procedure.*



*Figure 1.28 The* Account *table after insertion.*



*Figure 1.29 A few tables after running the* DataInsertion.sql *file.*

# Question Identification and Explanations

In this section I identify four questions that are useful to me as the owner/operator of the company using this database or useful to the end users as they use the system.

1. **How often are users entering their weight?**

My first question is, “How often are users entering their weight?” and will analyze data off the WeightHistory table. This is a useful metric to track because entering weight change takes very few clicks and is straight forward for the end-user. Therefore, it should be indicative of how often users are using the health and wellness tracker overall. If the number is low, further analysis can be done to discover why this functionality is not being utilized. If the number is high, further action can be taken to streamline the process so that even more users will come to our site to track their data.

I will use a query that pulls off the WeightHistory table and groups on month to see how many entries are being entered per month.

1. **How many accounts are Doctor, how many are Patient, how many are Researchers, and how many do not have a subtype?**

My second question is how many accounts are of the Doctor sub-type, how many are Researchers, how many are Patients, and how many accounts don’t have a sub-type? This question will be useful as we analyze how our system is used, and by whom. Based on the information we glean from this data, we can improve the system to capture more users and to increase the usage of our current users.

1. **How are users utilizing the survey functionality?**

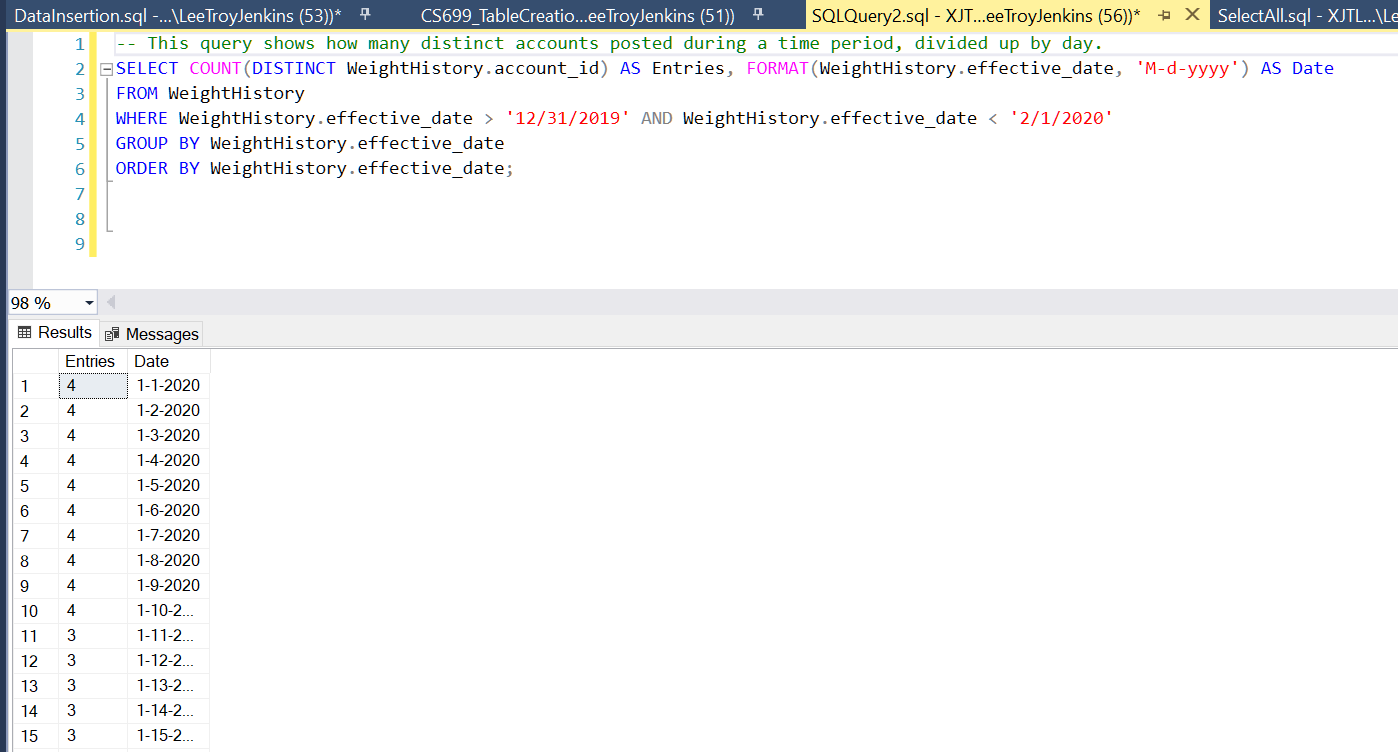
The third question is “What surveys are users creating?” This will pull survey questions and answers for users and sort them by account\_id and survey\_id. This query could also be used with a account\_id in the WHERE clause to create a view so users can view their past survey data and it builds a foundation that a more complicated query for Researchers can be built on to pull survey data from all of their project participants (Patients).

1. **Which journal entries have been authored by my patients?**

My fourth question will be useful for our end-users, specifically doctor. The question is, “Which journal entries belong to my patients, and how do I access those?” To answer this, I will create a view that pulls a list of all journal entries sorted by the author’s account ID that the doctor has access to read. Another view could easily be built changing out Researchers for Doctors.

# Query Executions and Explanations

Question 1: How often are users entering their weight?



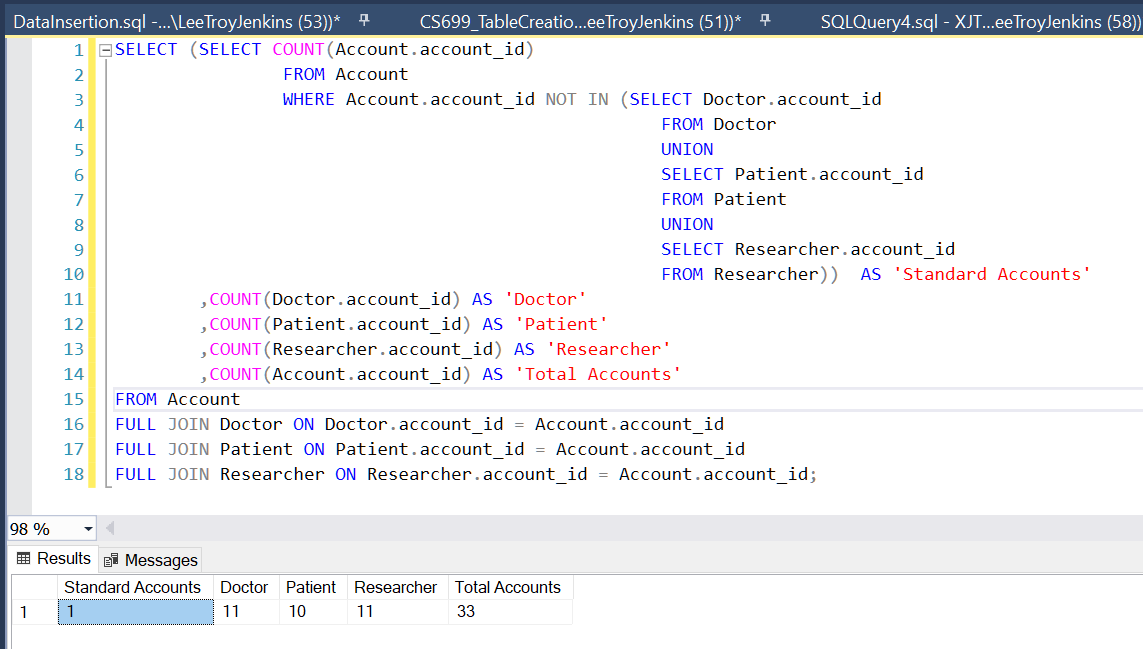
*Figure 1.30 A query showing the number of weight changes made by distinct accounts, per day.*

The query in Figure 1.30 shows the number of weight changes by distinct account on the WeightHistory table, per day. I select two columns: the first is a count aggregate on the user’s account ID, the second is the effective date formatted so that the time data is not included in the results or used in the ORDER BY function. I use DISTINCT when aggregating COUNT so that only one change per day counts per account.

The rows are filtered so that only the month of January, 2020 is returned, and then the results are grouped by day.

The final result shows the number of times distinct accounts tracked their weight each day in January.

**Question 2: How many accounts are Doctor, how many are Patient, how many are Researchers, and how many do not have a subtype?**



*Figure 1.31 A query that returns the total account number, as well as each account type.*

Figure 1.31 shows a query that calculates the total number of accounts and the number of account types. The query uses the COUNT function to calculate the number of accounts in each sub-type table (Doctor, Patient, Researcher). It also counts the number of records on the Account table to get the number of total accounts.

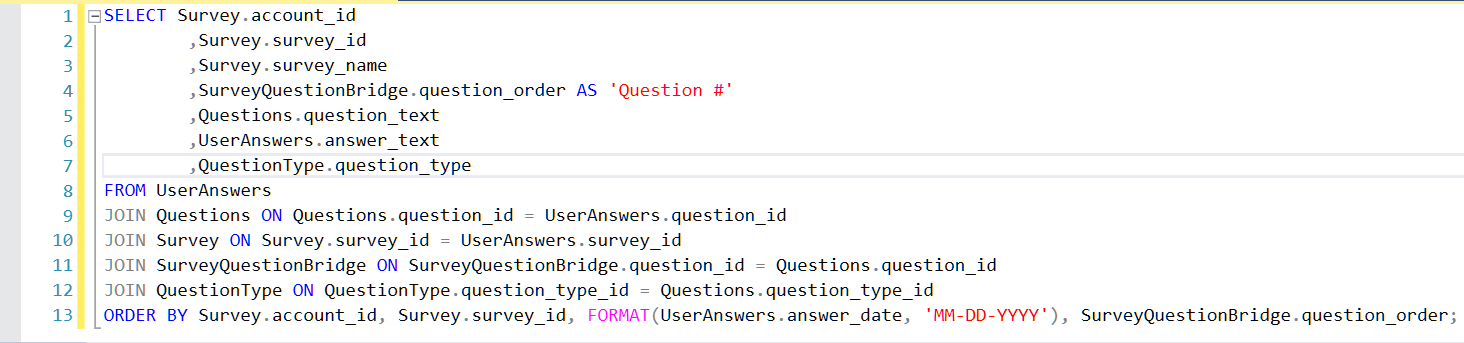
To calculate the number of accounts that aren’t a sub-type, I used subqueries in the SELECT statement.

The innermost subquery uses UNION to return a column of all account IDs that are on the Doctor, Patient, or Researcher tables. The query one step out from that uses the COUNT aggregate to count how many account IDs are on the Account table that aren’t in that list, which is the number of accounts that don’t have a subtype, or what I call “Standard Accounts” in the query results.

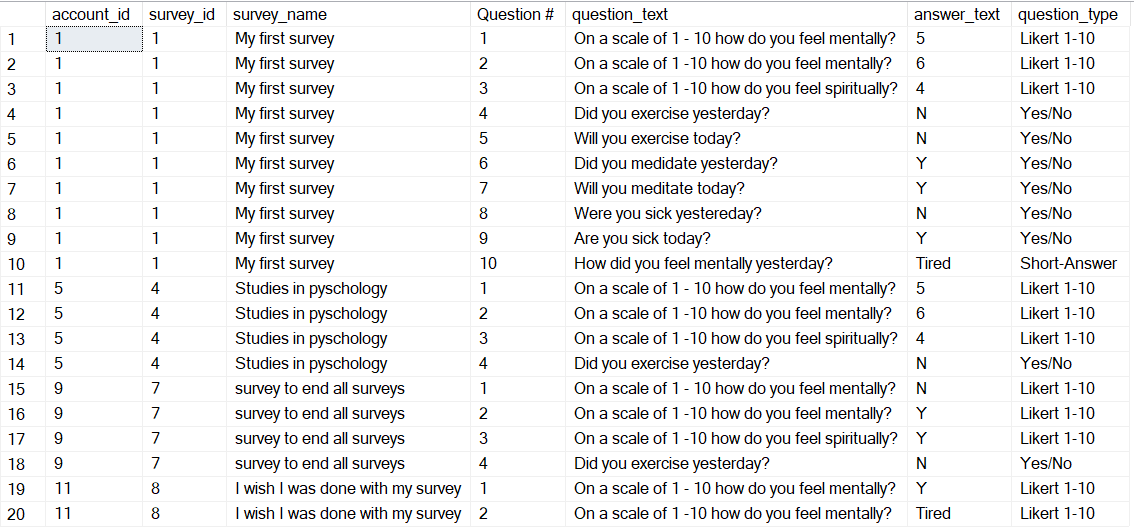
**Question 3: How are users utilizing the survey functionality?**

The system allows users to create survey templates that are made up of dynamic questions, meaning they can create their own questions and add or remove them from their survey templates. In order to efficiently store this data, the questions, answers, and survey template data are all stored on separate tables. This allows questions and templates to be used multiple times, and by multiple users.

This query is useful for anyone that wants to view previous survey data, because it recreates the surveys that users have taken. The returned data is ordered by account id, then by the survey id, the date, and the question order. Figure 1.32 shows the query, while Figure 1.33 shows the returned results.



*Figure 1.32 A query that combines SurveyTemplate with Questions and UserAnswers to give a snapshot of the survey’s that users are creating and taking.*

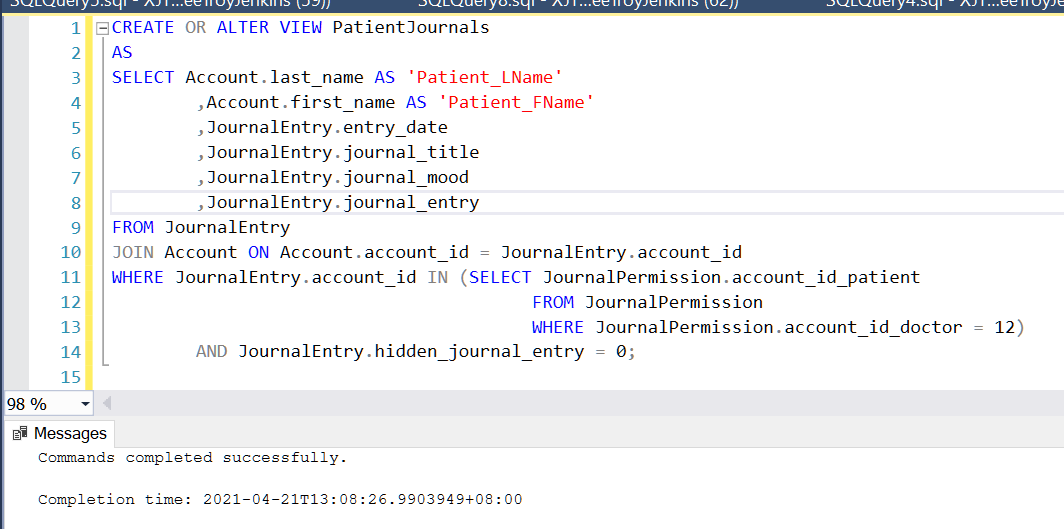
**

*Figure 1.33 A query that combines SurveyTemplate with Questions and UserAnswers to give a snapshot of the survey’s that users are creating and taking.*

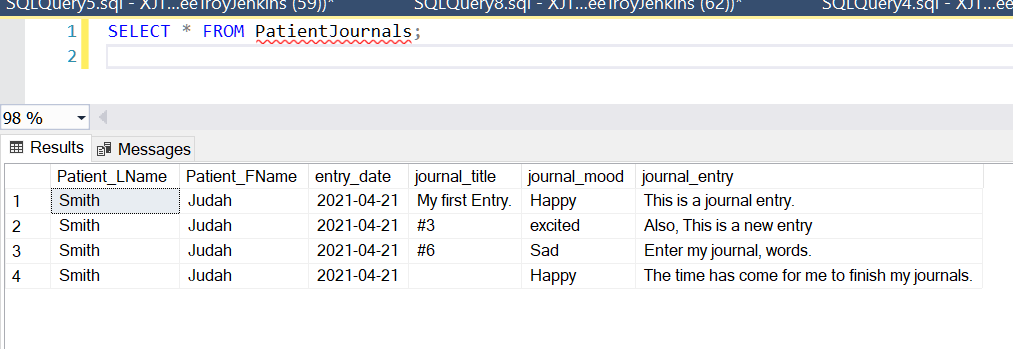
**Question 4: Which journal entries have been authored by my patients?**

This final question is useful for Doctors that want to view their patient’s journal entries.

The query creates a view for the doctor (in this case the doctor with the account ID 12). The view pulls the patient’s name and journal entry data by joining JournalEntry to Account via the account id. The results are filtered to entries that weren’t marked “hidden” by the journal entry author, and through the use of a subquery that returns a list of patient ID numbers that a particular doctor has permission to view (available on the JournalPermission table).



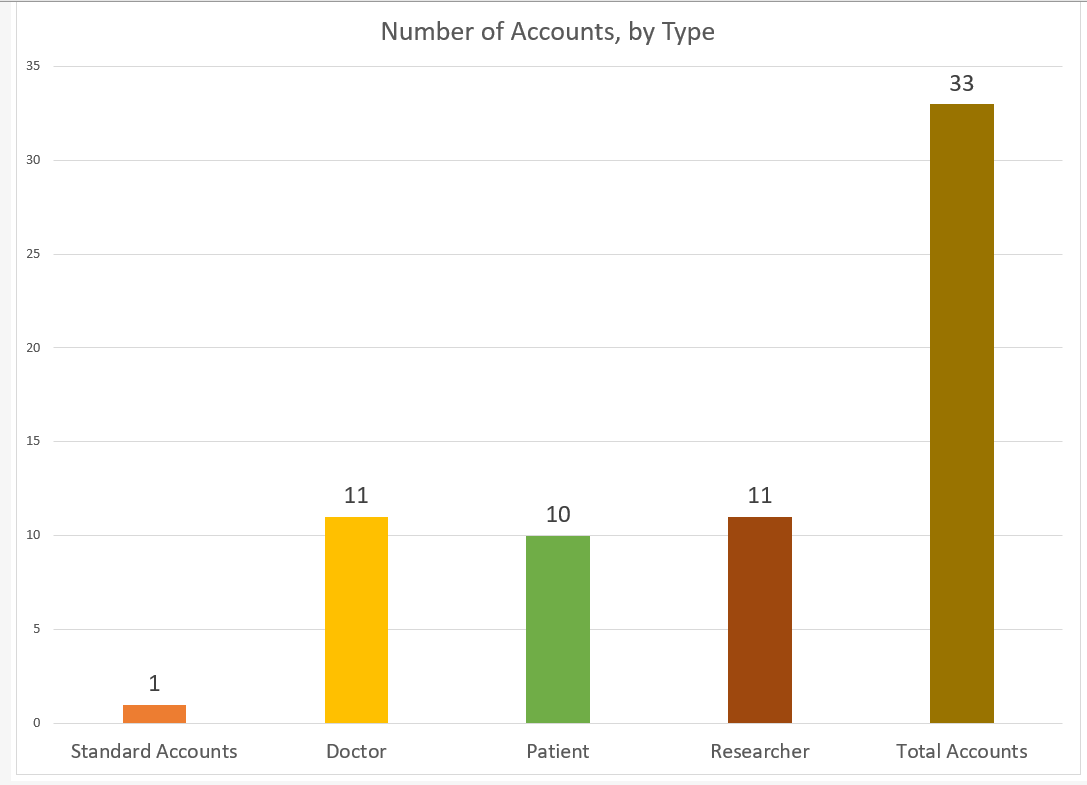
*Figure 1.34 A view for Doctors to view their patient’s journal entries.*



*Figure 1.36 Selecting all records using the PatientJournals view.*

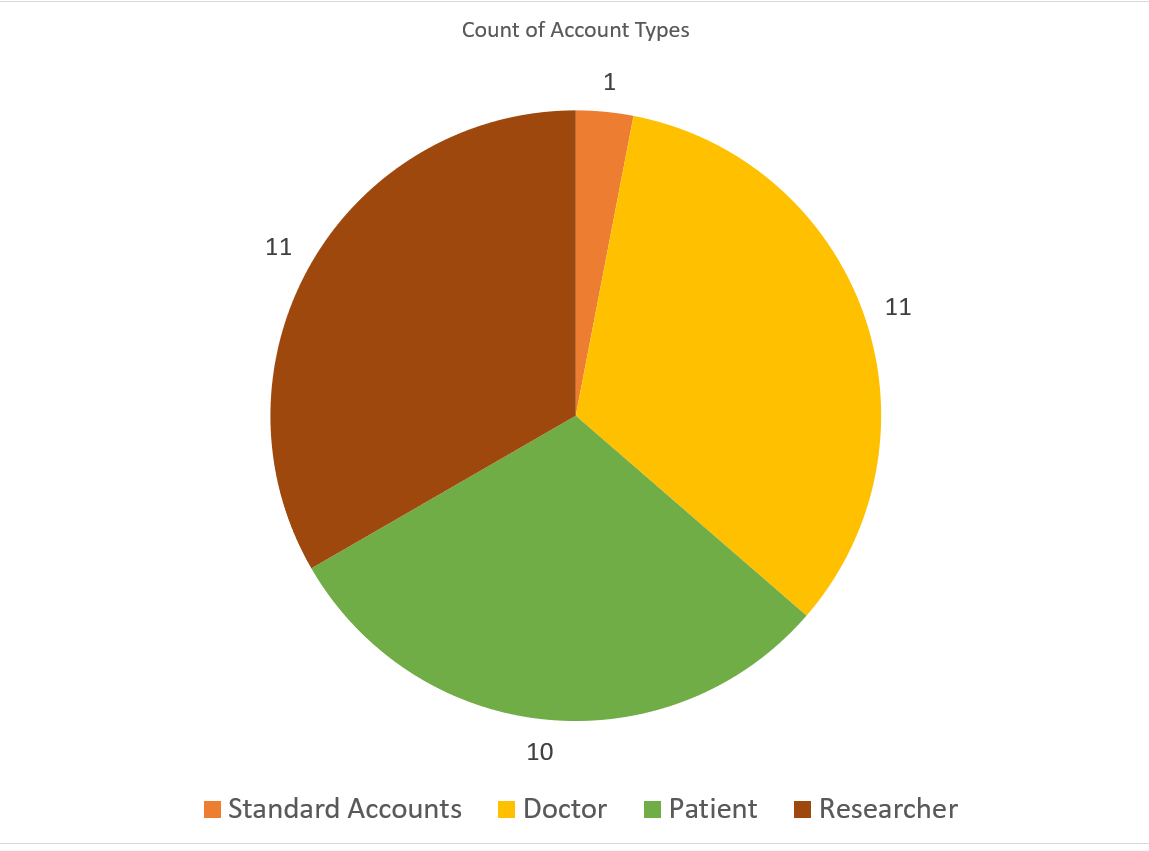
# Data Visualizations

One of the first questions that I would like to know and answer to is, “Who is using the system?” This question can be answered by using the query built for Question 2 in the previous section that shows user numbers by account type.



*Figure 1.37 The number of accounts by type. Barchart.*

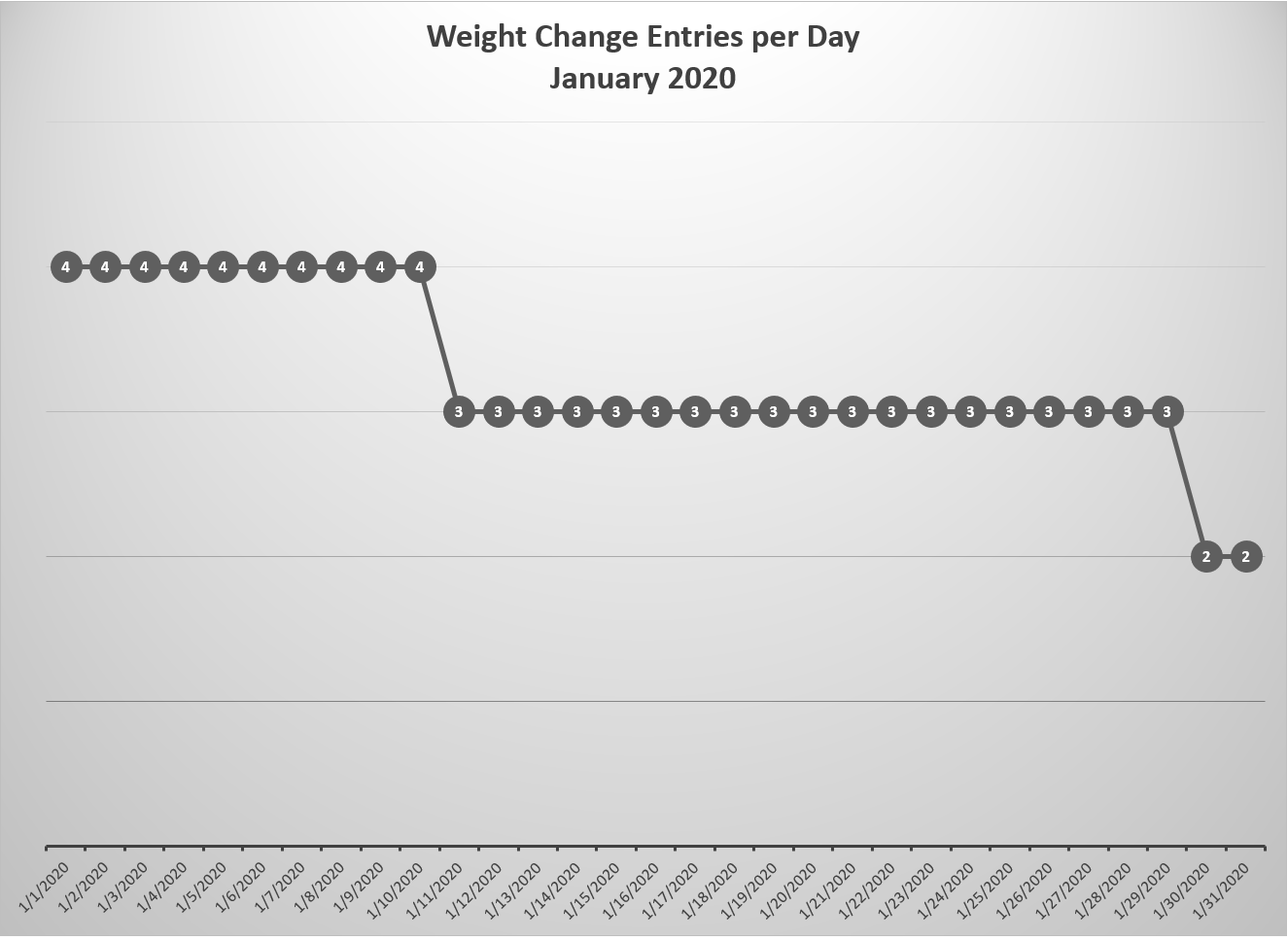
This figure shows the number of accounts in the system. The right hand bar “Total Accounts” shows that there are 33 users with accounts in the system. Looking at Figure 1.38 below, I can see that most accounts belong to a sub-group, and that it is split fairly evening into thirds between Doctor, Patient, and Researchers.



*Figure 1.38 The number of accounts by type. Pie chart.*

From this information, I ask myself two questions. 1. Why are regular account users so low and what can I do to increase usage for that type of user; and 2. How can I further capitalize on my success with Doctors, Patients, and Researchers? Should I increase functionality for those types of accounts, or perhaps add greater analysis capabilities for those users.

These questions lead me to my next question, which is how are users using the system right now? One query that I would run after asking myself this is the query from Question 1 in the previous section which determines how often users are entering their weight changes. Weight change is one of the easiest metrics to track for users, so I’m assuming that utilization will be high.



*Figure 1.39 Weight change entries by date.*

Viewing the plot, I notice that weight tracking usage starts high at the beginning of the month, but trends negative as January progresses. One assumption that I can make from this data is that it is possible that users track their weight more around the new year, but lose interest over time. Taking this into account, I would explore if there are options that could increase usage over time, such as sending out an email reminder to users that have indicated they would like a reminder to track their data. Another option could be to create another sub-type for people trying to manage their weight, and build functionality where they can interact with each other and encourage one another towards their weight goals.

# Summary and Reflection

This database is built to support a journaling and health & wellness website/app that will allow users to save journals, track food intake and exercise activities, and take dynamic health and wellness surveys. Between these three entities, data such as daily mood, weight, physical and mental health, exercise, and the number of hours slept will be stored in the database and can be used for analysis by end-users of the health and wellness APP over time. In addition, the database will allow users to share their data with their doctors or to be analyzed by researchers. Users will also be able to read journal entries that have been marked public by other users, or share their own journals to the public.

Through the use of the design techniques I learned in this class, my DBMS Physical ERD contains all of the tables necessary for each functional area of the Health and Wellness tracker to work properly. The survey functionality is supported by the Survey, Questions, and UserAnswers tables which allow survey questions to be created, used, and re-used by users. Survey questions can be saved in a particular grouping and order as a template on the Survey table, and can be re-assembled for analysis in the future. The journal functionality contains the JournalEntry table where users share their journal entries. The HealthTracker table links to the Exercise and FoodItems table to help users track their eating and exercising activities. Finally, the Patient, Doctor, and Researcher sub-types connect users into functional groups for specialized activities such as sharing journals (patient to doctor) or sharing surveys (patient to researcher). Seeing the design come together on the final ERD made me quite happy.

I’ve also used the skills I’ve learned in this class to create my tables, index important fields, build sequences, add procedures, and create triggers. After inserting data into the tables I identified interesting questions that I answered by building queries and views using SQL. I took the data from my queries and used it to do an analysis on parts of the database.

Overall, I’m very happy with my project. This is my first exposure to SQL, and I’m surprised with the amount of knowledge I’ve gained in the last five weeks. One of the most satisfying moments of the moment was running the script to insert data into all of my tables using the procedures I developed, and not getting any error messages after the execution. I am planning to implement my database in the real world, and I’m excited that I was able to build such a solid foundation in such a short period of time. I’m looking forward to continuing to work on it, and to using it myself in the future.

# Change Log

I added this change log so that I can easily keep track of the changes I am making over the course of this project.

|  |  |  |  |
| --- | --- | --- | --- |
| **Iteration** | **Location** | **Change** | **Date** |
| #5 | Use Case | * Added “case 15: User tracks Weight” * Added “weight” to HealthTracker entity. | 4/19/2021 |
| #5 | Full DBMS Physical ERD | * Changed FoodTracker to HealthTracker. * Changed “food\_tracker\_id” to “health\_tracker\_id”. * Added ExcerciseTypes entity. * Added ExcerciseBridge entity. * Added inedexes for ExcerciseBridge and Exercise FKs. | 4/19/2021 |
| #4 | Index Identification and Creation | Added section to document. | 4/13/2021 |
| #4 | History Table and Trigger | Added section to document. | 4/13/2021 |
| #4 | Summary and Reflection | Updated Summary and Reflect. |  |
| #4 | Full DBMS Physical ERD | * Updated Physical ERD * Updated Attribute table with new tables, attributes, and reasonings. | 4/12/2021 |
| #4 | Conceptual-Entity Relationship Diagram | Updated the Conceptual ERD to include new entities and relationships, including figures and descriptions.   * Added Doctor-Patient Relationship. * Added Project entity. * Added Researcher – Project relationship. * Added Project – Account relationship. * Added QuestionCategory entity. * Added QuestionCategory – Questions relationship. * Added QuestionType entity. * Added QuestionType – Questions relationship. * Added UnitofMeasure entity. * Added UnitofMeasure – Questions relationship. * Changed QuestionBank entity name to Questions. * Changed AnswerBank entity name to UserAnswers. * Changed OrganizationBridge to DoctorOrgBridge. * Added ResearchOrgBridge | 4/12/2021 |
| #4 | Structural Database Rules | * Database Rules   + Updated rule 1 to correctly reflect account sub-types.   + Added clarification to Rule 3.   + Added Rule 5   + Added Rule 7   + Added Rule 8   + Added Rule 9   + Added Rule 12   + Added Rule 13   + Added Rule 14   + Added Rule 15   + Added Rule 16 * Business Rules   + Removed character limit rule on journal entries. Journal entries will be saved as TEXT, not VARCHAR.   + Added Rule 10   + Added Rule 8 | 4/12/2021 |
| #4 | Use Cases and Fields | * Updated Use Case 1 to reflect modified account sub-types: Doctor, Researcher, Patient, or Casual User. * Added Use Case 2: Account Creation for Doctors and Researchers. * Changed “Entering Health and Wellness Data” to “User Case 5: User takes ‘default’ Health and Wellness Survey” and redefined case. * Modified “Adding Custom Fields to Health and Wellness Survey” to “User Creates a Custom Survey” and created new steps and fields. * Simplified “Use Case 8: Deleting Questions from the Health and Wellness Survey” * Added “Use Case 11: Researcher Creates a Research Project”. * Added “User Case 12: Researcher Adds Users and Patients to Project.”. | 4/12/2021 |
| #3 | Conceptual Entity-Relationship Diagram | Updated Figures 1.1 – 1.6 | 4/6/2021 |
| #3 | Conceptual Entity-Relationship Diagram | Updated the conceptual ERD based on issues found during the DBMS Physical ERD design and normalization process. Removed password entity, changed ACCOUNT and USER significance.  Changed Entity names.  SurveyAnswers becomes UserAnswers  SurveyQuestions becomes QuestionBank  Deleted:  UserPassword  Changed ACCOUNT subtypes:  Doctor  Researcher  Journaler | 4/6/2021 |
| #2 | Change Log | Added Change Log. | 3/29/2021 |
| #2 | Use Cases and Fields – Use Case 4 | Removed *UnitStandard* field. | 3/29/2021 |
| #2 | Use Cases and Fields – Use Case 7 | Added *DateCreated* to the password table. | 3/29/2021 |
| #2 | Use Cases and Fields – Use Case 1 | *User chooses account type from list of ‘Journaler’, ‘Researcher’, ‘Reader’, or a combination of these.* | 3/29/2021 |
| #2 | Use Cases and Fields – Use Case 1 | Added *AccountType* field. | 3/29/2021 |
| #2 | Structural Database Rules | Added and completed this section for iteration #2. | 3/29/2021 |
| #2 | Conceptual Entity-Relationship Diagram | Added and completed this section for iteration #2. | 3/29/2021 |
| #2 | Use Cases and Fields – User Case 8: Tracking Food – Step 3 | User types***or selects*** the name of the food, the portion size (numeric), the time they ate the food, and optionally can estimate the number of calories. (Change in **BOLD**). | 3/29/2021 |
| #2 | Use Cases and Fields – User Case 5: Adding Custom Fields to Health and Wellness Survey – SurveyName | Added this field to use case so that survey templates can be created and shared. | 3/29/2021 |
| #2 | Use Cases and Fields – User Case 5: Adding Custom Fields to Health and Wellness Survey – PublicSurvey | Added this field to use case so that survey templates can be created and shared. | 3/29/2021 |
| #2 | Use Cases and Fields – User Case 6: Deleting Questions from the Health and Wellness Survey | Added Step 2. *User selects the survey they want to edit from the dropdown.* | 3/29/2021 |
| #2 | Use Cases and Fields – User Case 9: User Changes User Name | Added this use case. | 3/29/2021 |
| #2 | Summary and Reflections | Made changes to the reflection. | 3/29/2021 |
| # 1 | Project Direction Overview, Use Cases, and Summary and Reflection. | Initial document creation. | 3/22/2021 |