**Group Members:** Allan Genari Gaarden, Tommy Ni, Saahil Vasdev, Joshua Medvinsky

**What is the application area of the database?**

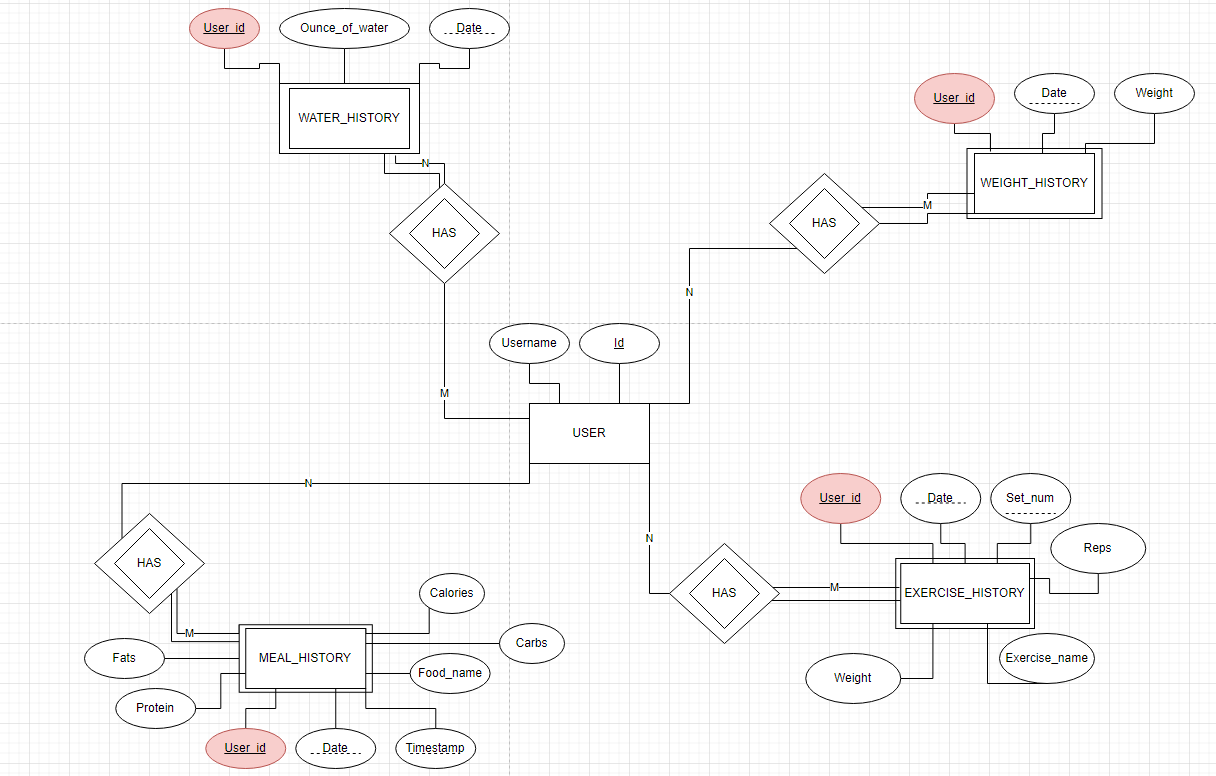
The application area of our database will primarily focus on health tracking. This database will be composed of data that involves water intake history, weight history, exercise history, and meal history.

**Database Requirements**

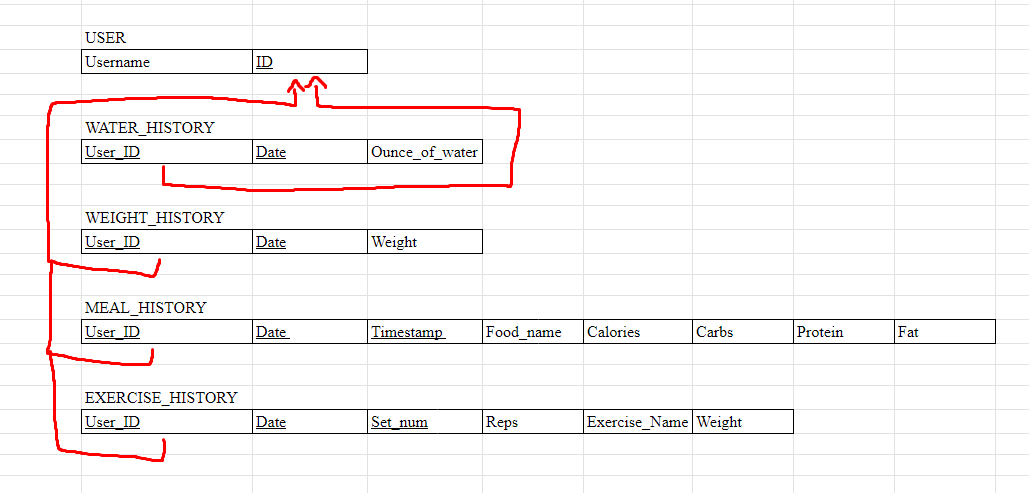
Consider a *health tracker* database where users input their daily exercise, meals, weight, and water intake. The data requirements are summarized as follows:

Each user has an username and an ID that uniquely identifies them. The user has a water history that tracks the ounces of water they drink on a particular day.The user has a weight history that tracks their weight (in pounds) on a particular day. The user has an exercise history that tracks their workout on a particular day which includes: exercise name, number of sets, number of reps, and can track the amount of weight they are lifting (if applicable). The user has a meal history that tracks what they are eating on a particular day which includes: calories, amount of carbohydrates, amount of fats, amount of proteins, food name, timestamp, and date. The user has a water history that tracks the total ounces of water consumed on a given date. The history for a particular user is referenced by the user ID.

**Figure 1 - ER Diagram for Health Tracker Database**



**Figure 2 - Relational Relation**



**SQL Statements**

Note: Due to the time sensitivity of using date(‘now’), which references the current day’s date, some SQL statements may return nothing as the data is “out of date”. Currently, the data only goes up to the date 12/13/2021. However, these queries worked on the day of our testing, 12/13/21.

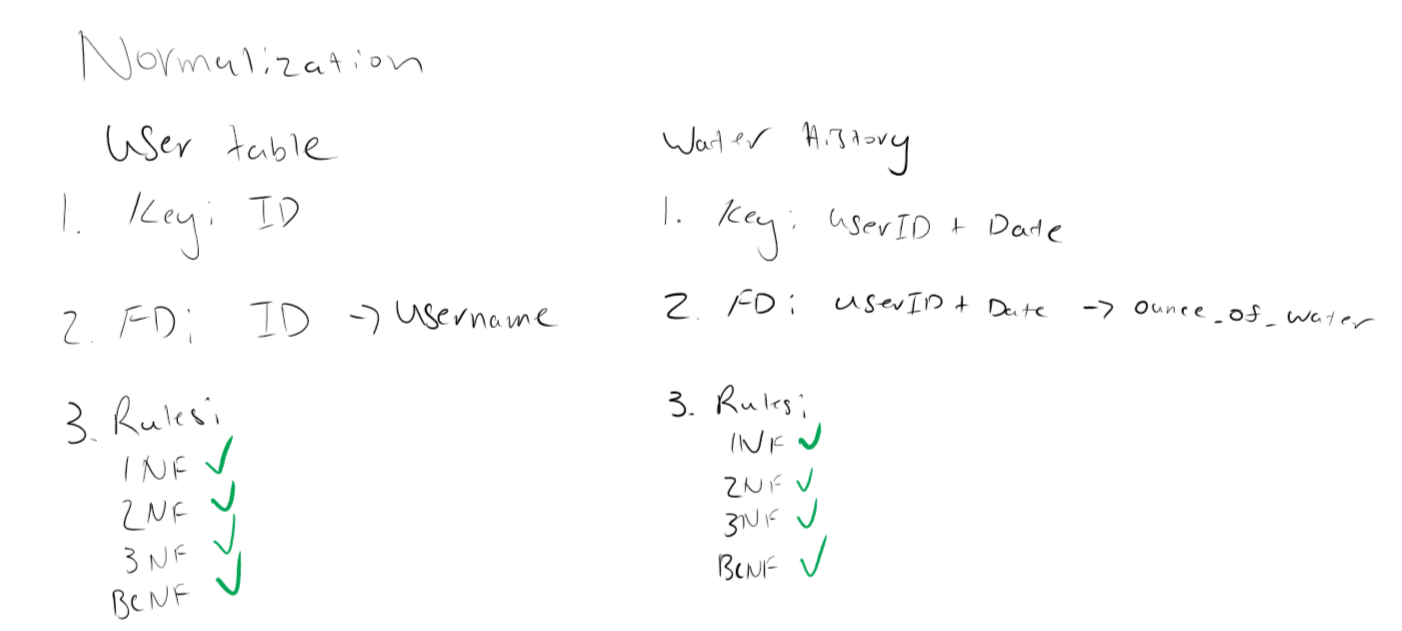
Users are specified in the SQL statements because in the context of using this database, it is assumed that a user querying the database would know their username.

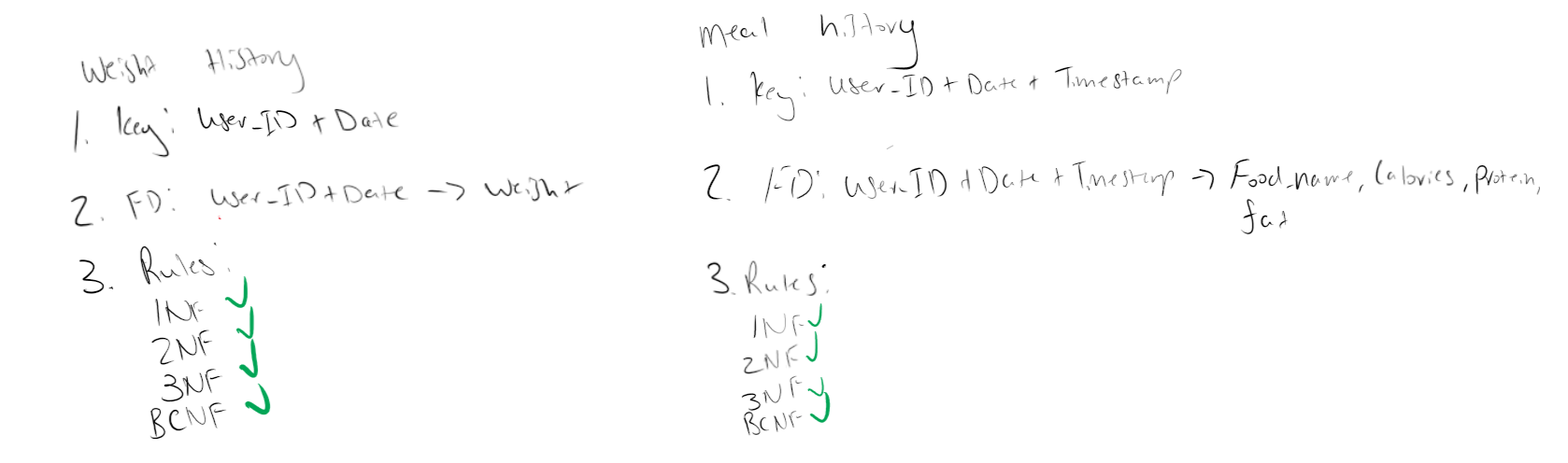
|  |  |
| --- | --- |
| **SQL Statement** | **Purpose** |
| select Date, Exercise\_Name, max(Weight) as Max\_Weight  from EXERCISE\_HISTORY  where User\_Id = (select Id from USER where Username = 'ILoveBamboo')  group by Exercise\_Name; | Returns the max weight for each exercise, and when that max was achieved for a particular user. In this case, the user is ‘ILoveBamboo.’ |
| select avg(Calories) as Average\_Calories, avg(Ounce\_of\_Water) as Average\_Water\_Intake\_in\_Ounce  from MEAL\_HISTORY as mh, WATER\_HISTORY as wh  where mh.User\_Id = (select Id from USER where Username = 'ILovePandas') and  wh.User\_Id = (select Id from USER where Username = 'ILovePandas') and  mh.Date >= Date('now', '-14 day') and  wh.Date >= Date('now', '-14 day'); | Finds the average calories and ounces of water consumed in the past 14 days for a specific user. In this case, the user is ‘ILovePandas’. |
| select  (select Weight from WEIGHT\_HISTORY where User\_Id = (select Id from USER where Username = 'ILovePandas') and Date = date('now'))  -  (select Weight from WEIGHT\_HISTORY where User\_Id = (select Id from USER where Username = 'ILovePandas') and Date = date('now', '-1 month'))  as Weight\_Change\_In\_Last\_Month; | Determine how a user’s weight has changed over the past month. In this case, the user is ‘ILovePandas.’ |
| select sum(protein)  from MEAL\_HISTORY  where User\_Id = (select Id from USER where Username = 'ILovePandas') and Date = date('now', '-2 day'); | Calculates the total grams of protein a user consumed on a specified day. In this case, the user is ‘ILovePandas.’ Protein can be substituted with any other nutrient to find that nutrient’s total as well. |
| select count(distinct date) as Amount\_of\_Time\_Exercised\_in\_the\_Last\_7\_Days  from EXERCISE\_HISTORY  where user\_id = (select id from USER where Username = 'ILovePandas') and Date >= date('now', '-7 day'); | Finds the amount of times a specific user exercised in the last 7 days. In this case, the user is ‘ILovePandas’. |
| select count(Timestamp) as Amount\_of\_Time\_Ate\_2\_Days\_Ago  from MEAL\_HISTORY  where User\_Id = (select Id from USER where Username = 'ILovePandas') and Date = date('now', '-2 day'); | Finds the amount of time a specific user eats, two days ago. In this case the user is ‘ILovePandas’. |
| select Date, Exercise\_Name, max(Weight) as Max\_Weight  from EXERCISE\_HISTORY  where User\_Id = (select Id from USER where Username = 'ILoveBamboo')  group by Date, Exercise\_Name; | For a particular user, get the max weight that they lifted on each exercise on each day that they have an entry, in this case the user is ‘ILoveBamboo’. |

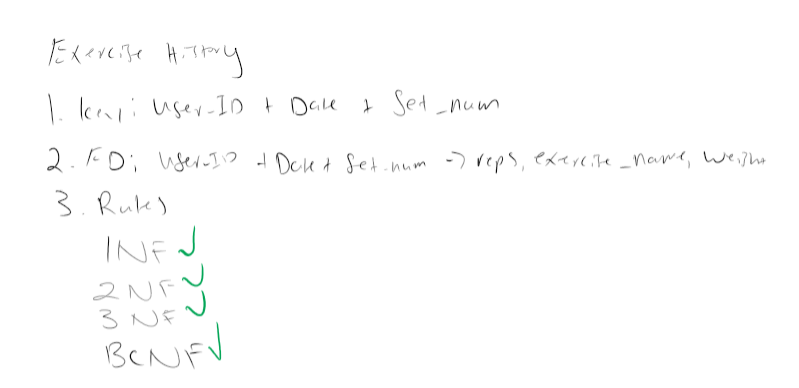
**Normal Form**

All of our relations are in BCNF. Due to our design of the database, each relation would only have one functional dependency. As shown in Figure 3, we have defined all functional dependencies for each relation. After defining all possible keys for each relation, we checked for each normal form with the defined functional dependencies. After checking for each normal form, we conclude that all of our relations are in BCNF.

**Figure 3 - Normalization Checking**







**Project Evaluation:**

Our group felt like our project covered all grounds for a health tracker database. Our group spent a lot of time and effort to ensure that the project criteria would be fulfilled. At the same time, we also took into consideration what the most straight-forward, efficient, and elegant implementation would be when designing our database. This ended up translating into an unconvoluted and very easy to understand database. Something that went wrong from what we originally planned was scrapping a table called TOTAL\_NUTRITION\_HISTORY. We decided to get rid of this table because it wasn’t accomplishing anything the other tables didn’t already cover. Instead, we used select statements that accomplished what the TOTAL\_NUTRITION\_HISTORY table would’ve done. We felt that the SQL statements we wrote were very solid, and the way we organized our tables and keys went right as well. If we had to do this project again, we would follow a consistent naming convention from the start as we had to spend a lot of time at the end correcting this issue.