Gym Database

Author: Tyler Gigot

Course: IST 659

Term: Fall 2020

Background

Weightlifting has always interested me since an early age. Throughout highschool and college I had done some resistance training, however, it was not until I was 21 years old that I fully committed to myself to weight training. At that point, I finally started going to the gym on a consistent basis. But I was not seeing the physical results that I wanted.

I signed up for personal training with a local trainer on my college campus. We worked together to build a weightlifting program. I was putting in significant amounts of effort and time and still, I was not seeing the results that I wanted. One day, an old friend who I had not talked to for a while reached out to me.

He was a psychology major with intentions of going to grad school for neuroscience. What was meant to be a quick catch up call turned into a lengthy discussion about weightlifting. He inquired about what I was doing for a weightlifting program. He explained some of the research that he has done on the subject. He shared a lot of information that was new to me.

I adopted his recommendations shortly after. The results were nothing short of phenomonal. I was losing weight and gaining muscle at the same time. About a year later I had shaved off 50 pounds and my lifts were up. My friends and family could not believe the transformation that had taken place. They were all wondering how I did it.

To sum it up in a few words, less is more. The program revolves around the concept of minimalism. Go to the gym no more than three days per week. Never workout two days in a row. Perform only the most essential compound movements. Use reverse pyramid style training. Take a minimum of 3 minutes of rest between each set. This style of training can be referred to as the leangains method.

The database that I will be implementing for this project is meant to be designed to supplement the leangains program. It is not designed to be an all-purpose, flexible workout tracking database. It is designed for the purpose of following a strict leangains program and tracking the

results. The business rules reflect the leangains program and the client is expected to be compliant with the leangains program.

In the subsequent section, I will be presenting the project through the lens of a personal trainer who has just signed on a new client and will be deploying the leangains programs for that client. As such, we will be using the database to track the data related to the lifting and nutrition of this client. As a disclaimer, the data that you see in the database is actually real data that I am tracking for myself but will be pretending that it is for a client.

Project Summary

You are a client who is interested in losing weight, building muscle, and achieving a chiseled, athletic-looking physical appearance. As a matter of fact, you have been working towards this goal for several years. Although there has been some noticeable progression over the years, you have not gotten the results that you would like.

After setting up an initial interview with you as a prospective client, I manage to convince you to be my client. You sign a 6 month contract and agree to follow my instructions and guidance. We will be using the leangains method for weightlifting.

Supplemental to the program itself, the leangains database is going to serve as the tool through which all data related to exercise and diet will be stored in. At the end of each day, you will take a moment to enter the data from that day into the database. There are several reasons as to why the database will be instrumental throughout this process.

One of the most important things about fitness is keeping track of your results. Traditionally, hand written notes in a notebook with pencil and paper are carried around. This is far from ideal. It becomes inefficient and takes up extra time at the gym. Not to mention, carrying around the notebook is a hassle when trying to workout. To make matters worse, the pencil may get lost or water can get spilled on the notebook. Sometimes people walking around accidently step on the notebook if its on the ground.

The database eliminates the nuisance of carrying around a notebook. There is still going to have be some manual entry of data, but this can be done directly on a phone through an application for your convenience. From there, analysis can be done and progress reports can be pulled out of the database at any time.

Lastly, with all of the logic of the leangains program being coded into the database, this should help enforce the boundaries of the training regiment. Although technically speaking, there is nothing stopping you from deviating from the training program, by having a database with constraints, and certain business rules that must be adhered to, it will help on the basis of accountability. The business rules section will get into some more specifics about that.

After the 6 month contract is completed and all of the data has been entered into the database, we will do some analysis to measure how the progress has been. Here are some questions that we might want to think about answering at that time:

Data Questions

- How did your 1 rep max of each lift change over time
- How much body weight did you lose or gain
- How much did the size of your muscle groups increase or decrease by
- How much money did you spend on average per day on food
- What kinds of foods did you eat the most

Stakeholders

The main stakeholder of this database project will be myself. My goal is to implement this database to help me better track my nutrition and fitness and hopefully draw some meaningful conclusions that give me insight into my results. I am actually a certified personal trainer, so maybe I will be able to use this down the road with a client, friend, or family member who is interested in following my program.

Business Rules

1. Congratulations on signing your contract with me as your personal trainer. I am looking forward to working with you. As such, you have been entered as a client in the database and your status has been deemed 'active'. When the 6 month contract is up, your status will be changed to 'inactive' and you will not be able to enter data into the database unless you renew.

2. This is a weightlifting database that is designed to follow the leangains program. As such, cardiovascular activity is prohibited from being documented in this database. In fact, it is required that you refrain from doing any cardiovascular activity altogether, at least in the short term. We can think about introducing some strategic cardio later on but either way that would not be captured in this database.

3. You cannot go to the gym two days in a row. Every workout must be separated by at least one full day of rest. This ensures that your central nervous system is recharged, you are able to exert maximum effort, and the risk of overtraining is reduced.

4. You must perform a minimum of 2 or a maximum of 3 workouts each week. The maximum of 3 workouts per week will be enforced by the database. Although the minimum of 2 workouts per week will not be enforced by the database, you are still expected to comply with that rule.

5. Your 3 workout session rotation must continue to be consistent in terms of the order. Every third session will have the same lifts performed. For example, if you decide that you will be hitting bench, chins, and bicep on workout 1, you must also do those same lifts in workout 4, 7, 10, and so on. Logically, there must be 6 full days between any given exercise.

6. You cannot exceed 3 sets of a particular lift in a particular workout, and you must perform a minimum of at least 2 sets per lift in a particular workout. The maximum of 3 sets per lift per workout will be enforced by the database. Although the minimum of 2 sets per lift per workout will not be enforced by the database, you are still expected to

comply with the rule. This is because one set has shown to be not enough volume to be effective while 4 sets or more has shown to be too much volume to be effective. As a result of this rule, the set number attribute in the database is only allowed to contain the values of 1, 2, or 3.

7. You cannot take a particular measurement for a particular muscle group multiple times in one day. For example, you could not measure your bicep twice in one day. This can only be done once per day per muscle group.
8. If you are doing measurements it must be either in centimeters if the size of a muscle group is being measured or pounds if weight is being measured but it cannot be any other unit of measurement. You will have to convert it to one of these two if the unit of measurement is different before you enter the value into the database.
9. You cannot split a workout into multiple gyms on a single day. For example, you cannot do half of a workout in one gym and the other half in another. An entire workout must take place in a single gym on a single day. Although not advised (for reasons out of the scope of this project), it is permitted to do workouts at different locations on different days.

Glossary

- CLIENT: An individual who is actively taking part in, or has previously took part in, a leangains weightlifting program and is currently using, or did use at a previous time, the leangains database for keeping track of their progression.
  - Client Name: a composite attribute which consists of the first name and the last name by which the client provides when they sign up for the program.
  - Birth Date: The birth date by which the client provides on their paperwork when they sign up for the program. Must include the day, month, and year.
  - Status: can be one of two possible values, “active” or “inactive”. An active client is one who is currently participating in the program. An inactive client is somebody who was at one time following the program but is no longer active.

- MEASUREMENT: a measurement can be taken by a client at a particular point in time. Again, only one measurement per category can be taken per day.
  - Measurement Name: the name that describes what is being measured
  - Recorded Value: numeric value of what the measurement was at the time of when it was taken.
  - Unit of Measurement: used to specify what the unit is for the recorded value. There are only two options that are allowed in this attribute and that is either “lb” (for pounds) or “cm” (for centimeter). If the measurement was taken in another unit it must be converted prior to entering into the database.
  - Date Taken On: the date for which a particular measurement for a particular client was taken on. Must include the day, month, and year.
  - Time Taken On: the approximate time in military time for which a particular measurement for a particular client was taken on.
- LIFT: a movement performed in a gym during a workout, that typically includes resistance, and impacts one or more muscle groups.
  - Lift Name: the name of the lift that is being performed.
- SET: a set is a collection of repetitions of a particular lift on a particular day performed at maximal effort, typically with resistance added.
  - Set Number: the set number is a number between 1 and 3. It cannot be anything other than these values as described in the business rules section.
  - Resistance: the amount of weight that is added onto a lift for a particular set measured in pounds and only in pounds. The weight must be converted into pounds before the value is entered into the database. For bodyweight movements (i.e. chin-ups), the resistance will be your average body weight measurement from that week + additional resistance added.
  - Repetitions: a single reptition is one full concentric and eccentric portion of a movement completed by a client. The reptition attribute is allowed to be any number that is not zero. You must perform a minimum of at least one reptition on a given set.

The recommended rep range is generally 6-12 with the reverse pyramid style training, it is understandable that in some cases a set may fall outside of these parameters but this should only occur on an exception basis.

  - RPE: stands for ratings of perceived exertion. This is a subjective measurement on a scale from 1 to 10 of how difficult you feel that a given set was directly after performing it, 10 being the most difficult and 1 being the least difficult. It is not a required attribute in the database but it could be helpful for insight later on so is recommended that you enter it as much as possible.

- WORKOUT: a workout is a collection of sets. A workout takes place on a single day at a single gym for a single client. The minimum number of sets that make up a workout is 6, and the maximum number of sets that make up a workout is 9, in respect of the other business rules that have already been established.
  - Workout Date: the calendar date that the workout was performed on. Must include the day, month, and year.
  - Workout Start Time: this is the approximate time at which the first set of the workout was performed.
  - Workout End Time: this is the approximate time at which the last set of the workout was completed.
  - Duration: this is the approximate duration of the workout, in minutes, calculated by taking the workout end time minus the workout start time.
- GYM: a gym is a facility that has the necessary equipment that can be used by a client to perform lifts. It should be noted that this does not necessarily have to be a commercial gym. For example, you could have weightlifting equipment at your home, and if the equipment is sufficient to perform the lifts then it would be considered a gym in this database.
  - Gym Name: the name of the gym. If it is a public gym then typically it will be the same as what you would see when you look it up in Google. If it is a private or home gym then the client has individual discretion over what to use as the name, however, it must remain consistent throughout the database. For example, you could not enter

the value of “home gym” one week, and then “home” the next. These are technically the same and as such the names must remain the same from workout to workout.
    - City: the city of where the gym is located
    - State: the state of where the gym is located
- NUTRITION LOG: a diary that contains records of all of the food and/or beverage items that were consumed by the client on a particular date. Anything that has calories should be entered into the nutrition log.
    - Date: the calendar date of which the nutrition was recorded by the client. Must include the day, month, and year.
- FOOD ITEM: an article of food or beverage that has calories and that is entered into the nutrition log. It is encouraged to consume food and beverage that is purchased at a store, rather than going out to eat, however, it is recognized that you will go out to eat at times as well. In the event that you do go out to eat, the food item should be entered clearly such as “Chickentenders_BaysideSupperClub”. You will have to make an estimate of the characteristics of the food item.
    - Food item Name: this is the name that describes the food item. There is some flexibility in this attribute, but a client must enter this as consistently as possible. Food items do not always come from the same store, and are not always the same brand, but it is advised to simply put the food item itself. For example, “Pineapple” would be preferred over “Dole Pineapple”.
    - Serving Size: this is the serving size of a particular food item from the nutrition label. This value will include both the size and the unit of measure in the same entry. For example, “4 ounces” may be entered for a serving size of ground beef. If a food item does not have a nutrition label, for example this is the case with pineapple and bananas, then the client should benchmark via Google search to find a suggested serving size as well as the subsequent nutritional characteristics.
    - Calories Per Serving: this is the amount of calories per one serving of the food item as stated on the nutrition label.

- Fat Per Serving: this is the amount of fat per one serving of the food item as stated on the nutrition label measured in grams
- Carbohydrate Per Serving: this is the amount of carbs per one serving of the food item as stated on the nutrition label measured in grams
- Protein Per Serving: this Is the amount of protein per one serving of the food item as stated on the nutrition label measured in grams
- Price Per Serving: this is the estimated cost of one serving of a food item based on data from outside of this database. Typically the client will use their prior knowledge of what they have paid for the food item or they can look up what the food item costs.
- Food Category: this is the category for which the food item would be considered according to the food pyramid.

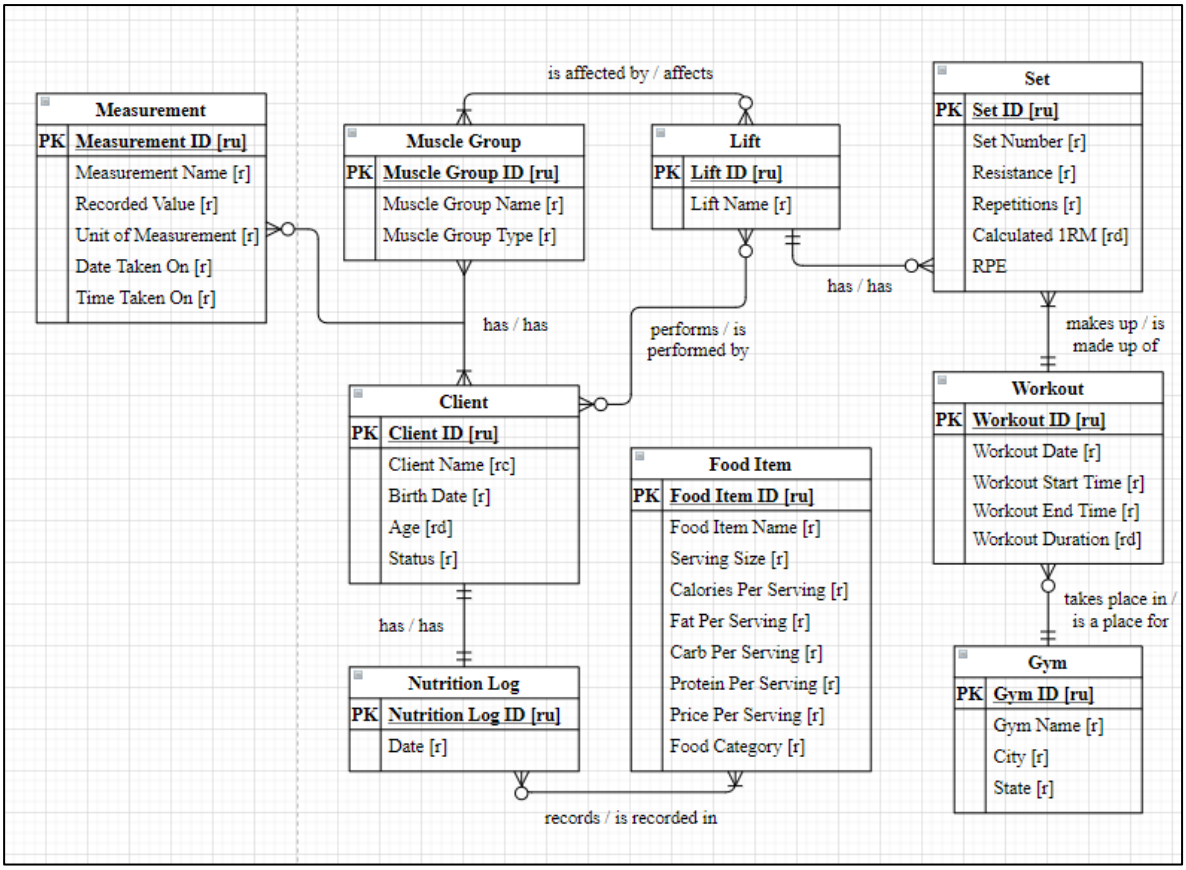
Conceptual Model

Entity	Attribute
Client	**Client ID [ru]*** **Client Name [rc]** **Birth Date [r]** **Status [r]**
Measurement	**Measurement ID [ru]** * **Measurement Name [r]** **Recorded Value [r]** **Unit of Measurement [r]** **Date Taken On [r]** **Time Taken On [r]**
Lift	**Lift ID [ru] *** **Lift Name [r]**
Set	**Set ID [ru] *** **Set number [r]** **Resistance [r]** **Repetitions [r]** RPE
Workout:	**Workout ID [ru]*** **Workout Date [r]** **Workout Start Time [r]** **Workout End Time [r]** **Workout Duration [r]**
Gym	**Gym ID [ru] *** **Gym Name [r]** **City [r]** **State [r]**
Nutrition Log	**Nutrition Log ID [ru] *** **Date [r]** **Servings Consumed [r]**
Food Item	**Food Item ID [ru] *** **Food item name [r]** **Serving Size [r]** **Calories Per Serving [r]** **Fat per seving [r]** **Carb per serving [r]** **Protein Per Serving [r]** **Price Per Serving [r]** **Food category [r]**

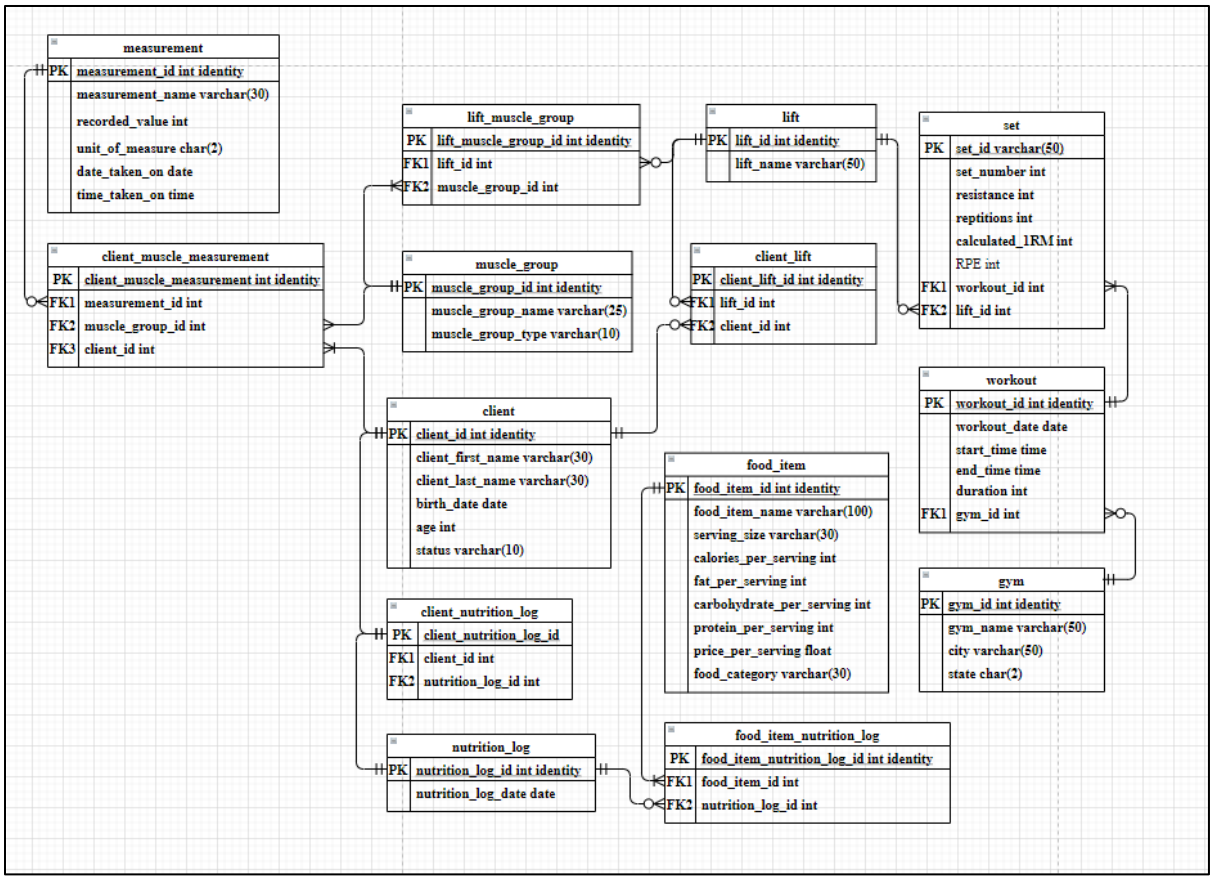
Relationships

A client takes zero to many measurements, measurements are taken by zero to many clients
A client performs zero to many lifts, a lift is performed by zero to many clients
A set has one and only one lift, a lift has zero to many sets
A workout is made up of many sets, a set makes up one and only one workout
A workout takes place in one and only one gym, a gym is a place for zero to many workouts
A client has one and only one nutrition log, a nutrition log has one and only one client
A nutrition log records one to many food items, a food item is recorded in zero to many nutrition logs

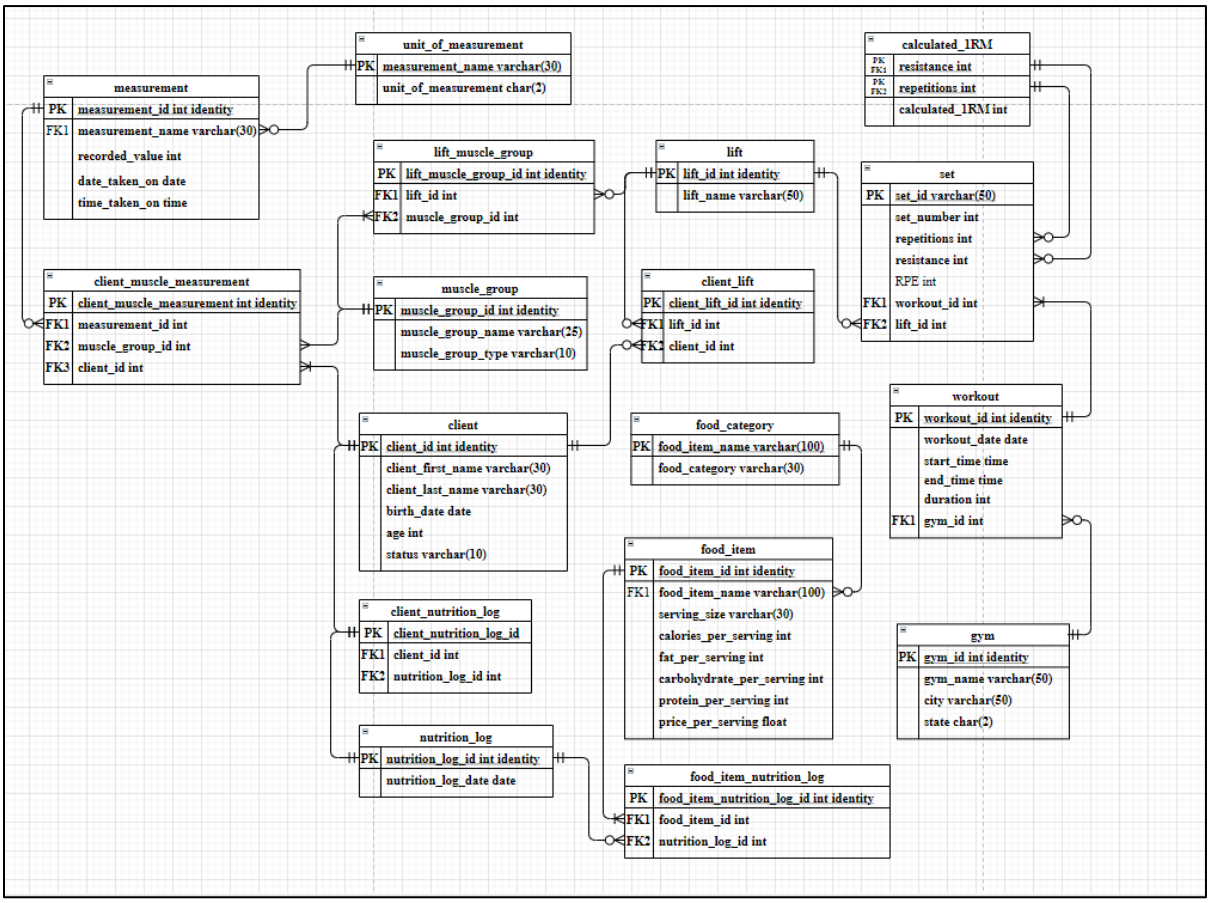
Conceptual Model



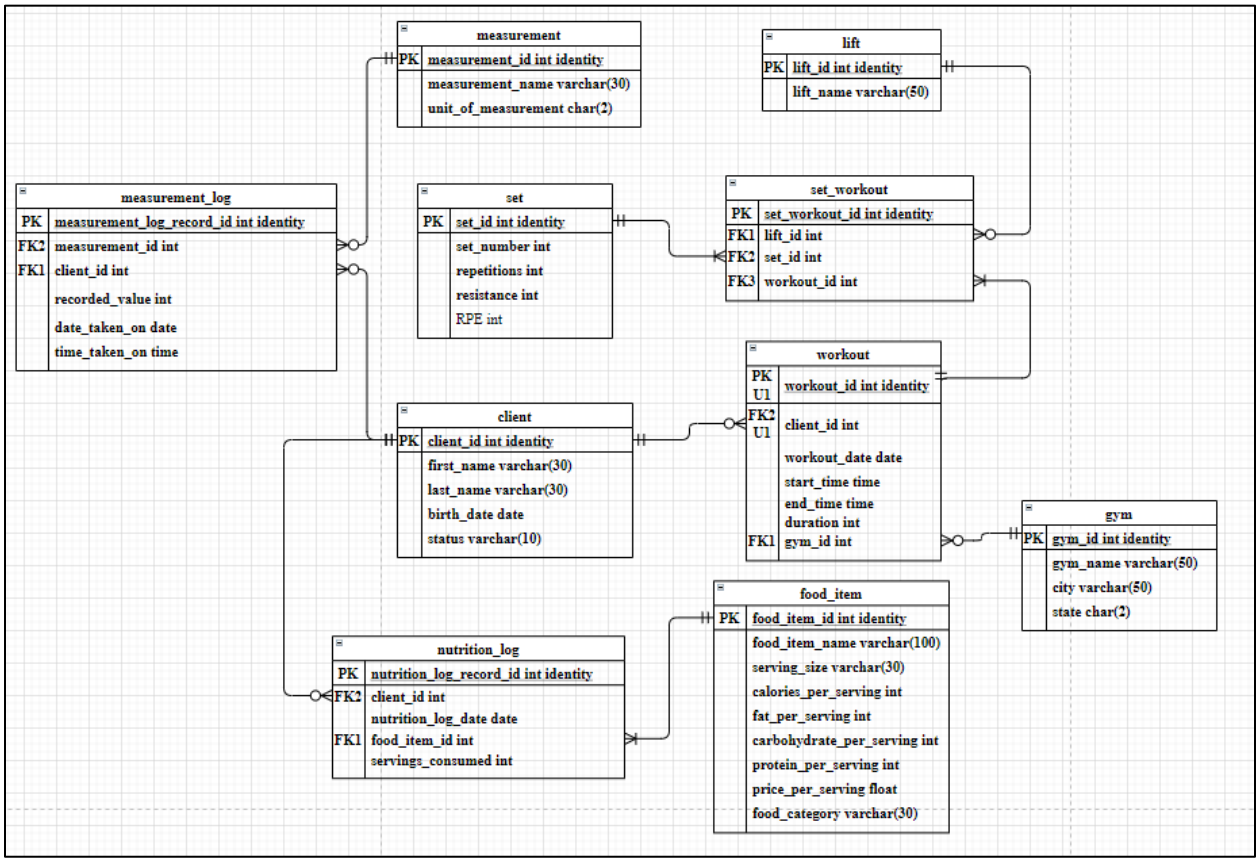
Logical Model



Normalized Model



Revised Normalized Model



Physical Database Design

```
-- optional wipe tables clause
drop table if exists set_workout_T
drop table if exists lift_T
drop table if exists set_T
drop table if exists workout_T
drop table if exists gym_T
drop table if exists measurement_log_T
drop table if exists measurement_T
drop table if exists nutrition_log_T
drop table if exists food_item_T
drop table if exists client_T
drop procedure if exists addworkout
drop procedure if exists addset
drop view if exists bench_progression_V
drop view if exists weekly_cals_V


-- start create client table
-- COMPLIES WITH BUSINESS RULE 1 --
create table client_T (
client_id int identity,
first_name varchar(30) not null,
last_name varchar(30) not null,
birth_date date not null,
status varchar(10) not null,
constraint PK_client_T primary key (client_id),
constraint CK_client_T check (status in ('active', 'inactive')))
go -- end create client table


-- start create food item table
create table food_item_T (
food_item_id int identity,
food_item_name varchar(100) not null,
serving_size varchar(30) not null,
calories_per_serving int not null,
fat_per_serving int not null,
carbohydrate_per_serving int not null,
protein_per_serving int not null,
price_per_serving float not null,
food_category varchar(30) not null,
constraint PK_food_item_T primary key (food_item_id))
go -- end create food item table


-- start create nutrition log table
create table nutrition_log_T (
nutrition_log_record_id int identity,
client_id int not null,
nutrition_log_date date not null,
food_item_id int not null,
servings_consumed float not null,
constraint PK_nutrition_log_T primary key (nutrition_log_record_id),
constraint FK1_nutrition_log_T foreign key (food_item_id) references food_item_T(food_item_id),
constraint FK2_nutrition_log_T foreign key (client_id) references client_T(client_id),
constraint U1_nutrition_log_T unique (nutrition_log_date, food_item_id, servings_consumed))
go -- end create nutrition log table


-- start create measurement table
-- COMPLIES WITH BUSINESS RULE 8 --
create table measurement_T (
measurement_id int identity,
measurement_name varchar(30) not null,
unit_of_measurement char(2) not null,
constraint PK_measurement_T primary key (measurement_id),
constraint C1_measurement_T check (unit_of_measurement in ('cm', 'lb')))
```

```
go -- end create measurement table


-- start create measurement log table
--  COMPLIES WITH BUSINESS RULE 7 --
create table measurement_log_T (
measurement_log_record_id int identity,
client_id int not null,
measurement_id int not null,
recorded_value float not null,
date_taken_on date not null,
time_taken_on time not null,
constraint PK_measurement_log_T primary key (measurement_log_record_id),
constraint FK1_measurement_log_T foreign key (measurement_id) references
measurement_T(measurement_id),
constraint FK2_measurement_log_T foreign key (client_id) references client_T(client_id),
constraint U1_measurement_log_T unique (client_id, measurement_id, date_taken_on))
go -- end create measurement log table


-- start create gym table
create table gym_T (
gym_id int identity,
gym_name varchar(50) not null,
city varchar(50) not null,
state char(2) not null,
constraint PK_gym_T primary key (gym_id))
go -- end create gym table


-- start create workout table
-- COMPLIANT WITH BUSINESS RULE 9 --
create table workout_T (
workout_id int identity,
client_id int not null,
workout_date date not null,
start_time time not null,
end_time time not null,
duration int not null,
gym_id int not null,
constraint PK_workout_T primary key (workout_id),
constraint FK1_workout_T foreign key (gym_id) references gym_T(gym_id),
constraint FK2_workout_T foreign key (client_id) references client_T(client_id),
constraint U1_workout_T unique (client_id, workout_date))
go -- end create workout table


-- start create set table
-- COMPLIES WITH BUSINESS RULE 6 --
create table set_T (
set_id int identity,
set_number int not null,
repetitions int not null,
resistance int not null,
RPE int not null,
constraint PK_set_T primary key (set_id),
constraint C1_set_T check (set_number in (1, 2, 3)),
constraint C2_set_T check (repetitions > 0),
constraint C3_set_T check (RPE in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)))
go -- end create set table



-- start create lift table
-- COMPLIES WITH BUSINESS RULE 2 --
create table lift_T (
lift_id int identity,
lift_name varchar(50) not null,
constraint PK_lift_T primary key (lift_id),
constraint C1_lift_T check (lift_name not like ('%cardio%')))
go -- end create lift table
```

```
-- start create set workout table
create table set_workout_T (
set_workout_id int identity,
lift_id int not null,
set_id int not null,
workout_id int not null,
constraint PK_set_workout_T primary key (set_workout_id),
constraint FK1_set_workout_T foreign key (lift_id) references lift_T(lift_id),
constraint FK2_set_workout_T foreign key (set_id) references set_T(set_id),
constraint FK3_set_workout_T foreign key (workout_id) references workout_T(workout_id))
go -- end create set workout table


/*-------------------------STORED PROCEDURE FOR ENTERING A WORKOUT-------------------------*/
                            --COMPLIES WITH BUSINESS RULES 1,3,4--


/* start create stored procedure for entering a workout into the database*/
    create procedure addworkout (
    @client_id      int,
    @workout_date   date,
    @start_time     time,
    @end_time       time,
    @duration       int,
    @gym_id         int)
    as begin
/* store the last workout date for the client as a variable in the procedure*/
    declare         @lastworkout date
    set             @lastworkout = (
    select          max(workout_date)
    from            workout_T
    where           client_id = @client_id)
/* store # of workouts over last 7 days for the client as a variable in the procedure*/
    declare         @numworkout int
    set             @numworkout = (
    select          count(workout_id)
    from            workout_T
    where           client_id = 1
    and             workout_date >= dateadd(day, -7, @workout_date))
/* store the status of the client as a variable in the procedure*/
    declare         @status varchar(10)
    set             @status = (
    select          [status]
    from            client_T
    where           client_id = @client_id)
/*refer back to the lastworkout date to ensure that it complies with business rules*/
    if @workout_date > dateadd(day, 1, @lastworkout) -- BUSINESS RULE 3
    and @numworkout < 3 -- BUSINESS RULE 4
    and @status = 'active' -- BUSINESS RULE 1
    begin
    insert into workout_T (client_id, workout_date, start_time, end_time, duration, gym_id)
    values (@client_id, @workout_date, @start_time, @end_time, @duration, @gym_id)
    end
    else
    print 'Date does not work'
    end
go -- end create stored procedure for entering a workout into the database


/*-----------------------------STORED PROCEDURE FOR ENTERING A SET-----------------------------*/
                            --COMPLIES WITH BUSINESS RULES 1,5,6--


/* start create stored procedure for entering a set into the database*/
    create procedure addset (
    @client_id      int,
    @workout_id     int,
    @set_number     int,
    @reptitions     int,
    @resistance     int,
    @RPE            int,
```

```
    @lift_id        int)
    as begin
/* store the last lift date for the client as a variable in the procedure*/
    declare         @lastlift date
    set             @lastlift = (
    select          max(workout_date) from
    (select         set_T.set_id, set_workout_T.lift_id, workout_T.workout_date,
client_T.client_id
    from            set_T
    join            set_workout_T on set_workout_T.set_id = set_T.set_id
    join            workout_T on workout_T.workout_id = set_workout_T.workout_id
    join            client_T on client_T.client_id = workout_T.client_id) sub
    where           client_id = @client_id
    and             lift_id = @lift_id)
/* store the workout date as looked up by the workout id parameter as a variable in the
procedure*/
    declare         @workout_date date
    set             @workout_date = (
    select          workout_T.workout_date
    from            workout_T
    where           workout_T.workout_id = @workout_id)
/* store the set_id as the max of set id's +1 as a variable in the procedure*/
    declare         @set_id int
    set             @set_id = (
    select          max(set_id) +1 from
    (select         set_T.set_id, set_workout_T.lift_id, workout_T.workout_date,
client_T.client_id
    from            set_T
    join            set_workout_T on set_workout_T.set_id = set_T.set_id
    join            workout_T on workout_T.workout_id = set_workout_T.workout_id
    join            client_T on client_T.client_id = workout_T.client_id) sub)
/* store the status of the client as a variable in the procedure*/
    declare         @status varchar(10)
    set             @status = (
    select          [status]
    from            client_T
    where           client_id = @client_id)
/* store the current number of sets that are in for a lift on a day as a variable in the
procedure*/
    declare         @numsets int
    set             @numsets = (
    select          count(set_id) from
    (select         set_T.set_id, set_workout_T.lift_id, workout_T.workout_id,
workout_T.workout_date, client_T.client_id
    from            set_T
    join            set_workout_T on set_workout_T.set_id = set_T.set_id
    join            workout_T on workout_T.workout_id = set_workout_T.workout_id
    join            client_T on client_T.client_id = workout_T.client_id) sub
    where           client_id = @client_id
    and             lift_id = @lift_id
    and             workout_id = @workout_id)
/*refer back to the lastlift date to ensure that it complies with business rules*/
    if @workout_date >= dateadd(day, 6, @lastlift) -- BUSINESS RULE 5
    and @status = 'active' -- BUSINESS RULE 1
    and @numsets < 3  -- BUSINESS RULE 6
    begin
    begin transaction
    insert into set_T (set_number, repetitions, resistance, RPE)
    values (@set_number, @reptitions, @resistance, @RPE)
    insert into set_workout_T (lift_id, set_id, workout_id)
    values (@lift_id, @set_id, @workout_id)
    commit transaction
    begin transaction
    end
    else
    print 'Date does not work'
    end
go -- end create stored procedure for entering a set into the database
```

```
-- start create bench press progression view
create view bench_progression_V as
select top 1000 client_T.first_name+' '+client_T.last_name as clientname,
set_T.set_id, lift_T.lift_name, set_T.repetitions, set_T.resistance, set_T.RPE,
workout_T.workout_date,
cast(set_T.resistance / (1.0278 - 0.0278 * set_T.repetitions) as int) as estimated_1rm
from set_T
join set_workout_T on set_workout_T.set_id = set_T.set_id
join workout_T on workout_T.workout_id = set_workout_T.workout_id
join client_T on client_T.client_id = workout_T.client_id
join lift_T on lift_T.lift_id = set_workout_T.lift_id
where set_T.set_number = 1
and set_workout_T.lift_id = 1
order by workout_date asc
go -- end create bench press progression view


-------------------------------------------------------------------------------
-------------------------- start avg cals by week view ------------------------
create view weekly_cals_V as                                                  --
select datepart(year, nutrition_log_date) as log_year,                        --
datepart(week, nutrition_log_date) as log_week,                               --
sum(total_calories)/7 as avg_daily_cals from                                  --
------------------------------ start sub query 1 ---------------------------------
/**/ (select nutrition_log_T.nutrition_log_date, nutrition_log_T.food_item_id,  --
/**/ nutrition_log_T.servings_consumed, food_item_T.calories_per_serving,       --
/**/ servings_consumed*calories_per_serving as total_calories                  --
/**/ from nutrition_log_T                                                      --
/**/ join food_item_T on food_item_T.food_item_id = nutrition_log_T.food_item_id--
/**/ where nutrition_log_T.client_id =1                                        --
/**/ group by nutrition_log_T.nutrition_log_date, nutrition_log_T.food_item_id, --
/**/ nutrition_log_T.servings_consumed,food_item_T.calories_per_serving) sub    --
------------------------------ start sub query 1 ---------------------------------
group by datepart(year, nutrition_log_date), datepart(week, nutrition_log_date) --
-------------------------- end avg cals by day view ---------------------------
-------------------------------------------------------------------------------
```

Physical Database Design

```
-- start add records to client table
insert into client_T (first_name, last_name, birth_date, status) values
('tyler', 'gigot', '02-20-1995', 'active')
-- end add records to client table


-- start add records to food item table
insert into food_item_T (food_item_name, serving_size,
calories_per_serving, fat_per_serving, carbohydrate_per_serving,
protein_per_serving, price_per_serving, food_category) values
('french onion soup' , '1 cup' , 210 , 10 , 18 , 15 , 0.5 , 'other'),
('cliff bar chunky peanutbutter' , '1 bar' , 260 , 7 , 41 , 11 , 1.5 , 'grain'),
('beef regular 93%' , '4 oz' , 160 , 8 , 0 , 23 , 1.35 , 'meat'),
('mission flour tortilla' , '1 tortilla' , 90 , 4 , 32 , 6 , 0.36 , 'grain'),
('grapes' , '100 grams' , 70 , 0 , 20 , 0 , 0.75 , 'fruit')
-- end add records to food item table


-- start add records to measurement table
insert into measurement_T (measurement_name, unit_of_measurement) values
('weight', 'lb'),
('left bicep', 'cm'),
('right bicep', 'cm'),
('chest', 'cm'),
('waist', 'cm')
-- end add records to measurement table
```

```
-- start add records to measurement log table
insert into measurement_log_T (client_id, measurement_id,
recorded_value, date_taken_on, time_taken_on) values
(1, 1, 167.9, '10-05-2020', '07:00:00 AM'),
(1, 2, 37, '10-05-2020', '07:00:00 AM'),
(1, 3, 37, '10-05-2020', '07:00:00 AM'),
(1, 4, 103.4, '10-05-2020', '07:00:00 AM'),
(1, 5, 79.7, '10-05-2020', '07:00:00 AM'),
(1, 1, 166.4, '10-12-2020', '07:00:00 AM')
-- end add records to measurement log table


-- start add records to gym table
insert into gym_T (gym_name, city, state) values
('western raquet', 'green bay', 'wi'),
('home', 'green bay', 'wi'),
('big tex', 'austin', 'tx'),
('golds gym', 'austin', 'tx')
-- end add records to gym table


-- start add records to workout table
insert into workout_T (client_id, workout_date, start_time, end_time, duration, gym_id) values
(1, '10-07-2020', '05:00:00 PM', '05:45:00 PM', 45, 1),
(1, '10-10-2020', '04:45:00 PM', '05:20:00 PM', 35, 1),
(1, '10-12-2020', '04:30:00 PM', '05:05:00 PM', 35, 1),
(1, '10-14-2020', '06:30:00 PM', '07:15:00 PM', 45, 1)
-- end add records to workout table


--start add records to lift table
insert into lift_T (lift_name) values
('flat dumbell bench'),
('neutral grip chinup'),
('standing dumbell hammer curl'),
('conventional deadlift'),
('barbell overhead press'),
('rope pushdown'),
('seal row'),
('walking dumbell lunge'),
('barbell calve raise')
--end add records to lift table
```

Data Manipulation

```
/*-------------------------------------------------------------------------------------------------------*/
/*                              DEMONSTRATE UNSUCCESSFUL UPDATE ATTEMPTS                                 */
/*-------------------------------------------------------------------------------------------------------*/


/*-----------------VIOLATION OF BUSINESS RULE 1: a client must be 'active' or 'inactive'------------*/
insert into client_T (first_name, last_name, birth_date, status) values
('Nick', 'Budsworth', '4-20-1970', 'not active')


/*-----------------VIOLATION OF BUSINESS RULE 2: a client cannot enter cardio as a lift-------------*/
insert into lift_T (lift_name) values ('cardio')


/*-----------VIOLATION OF BUSINESS RULE 9: a client can't have two workouts on the same date--------*/
insert into workout_T (client_id, workout_date, start_time, end_time, duration, gym_id) values
(1, '12-02-2020', '18:30', '19:05', 35, 1)
```

```
/*---------------VIOLATION OF BUSINESS RULE 3: a client can't workout two days in a row-------------*/
-- from the query result you can see that this record was not added to the table
declare @return_value1 int
exec @return_value1 = addworkout
1, '12-01-2020', '18:30', '19:05', 35, 1
select 'Return Value' = @return_value1
select * from workout_T where client_id = 1 and workout_date = '12-01-2020'


/*---------------VIOLATION OF BUSINESS RULE 5: a client can't do a lift out of rotation-------------*/


-- first add a workout to the database so the set can be linked to it
declare @return_value2 int
exec @return_value2 = addworkout
1, '12-08-2020', '18:30', '19:05', 35, 1
select 'Return Value' = @return_value2
select * from workout_T where client_id = 1 and workout_date = '12-08-2020'
-- from the query result you can see that this record was added to the table


-- now try to add a set of bench when bench was also done on the last workout
declare @return_value3 int
exec @return_value3 = addset
1, 25, 1, 10, 200, 8, 1
select 'Return Value' = @return_value3
select  set_T.set_id, set_workout_T.lift_id, lift_T.lift_name, workout_T.workout_id,
workout_T.workout_date, client_T.client_id
from set_T
join set_workout_T on set_workout_T.set_id = set_T.set_id
join workout_T on workout_T.workout_id = set_workout_T.workout_id
join client_T on client_T.client_id = workout_T.client_id
join lift_T on lift_T.lift_id = set_workout_T.lift_id
order by workout_date desc
-- from the query result you can see that this record was not added to the table


/*-------------------------------------------------------------------------------------------------------*/
/*                       DEMONSTRATE SUCCESSFUL UPDATES, INSERTS, AND DELETES                            */
/*-------------------------------------------------------------------------------------------------------*/


-- delete the workout that was just added
delete workout_T where workout_id = 25
select * from workout_T order by workout_date desc
-- from the query result you can see that this record was deleted from the table


-- correcting end time and duration for workout 5
update workout_T set end_time ='17:00' where workout_id = 5
update workout_T set duration = 30 where workout_id = 5
select * from workout_T where workout_id = 5
-- from the query result you can see that this record was updated in the table


-- pull up some tables to assist in the subsequent entry of data
select * from food_item_T order by food_item_name asc
select * from nutrition_log_T
select nutrition_log_T.nutrition_log_date, nutrition_log_T.food_item_id, food_item_T.food_item_name,
servings_consumed, calories_per_serving
from nutrition_log_T
join food_item_T on food_item_T.food_item_id = nutrition_log_T.food_item_id
order by nutrition_log_date desc


-- log a couple of days worth of nutrition data
insert into food_item_T (food_item_name, serving_size, calories_per_serving, fat_per_serving,
carbohydrate_per_serving, protein_per_serving, price_per_serving, food_category) values
('neils elk', '4 oz', 150, 5, 0, 21, 0, 'meat'),
```

```
('organic white wine generic', '6 oz', 120, 0, 20, 0, 2.50, 'alcohol'),
('salsa generic', '1 serving', 10, 0, 2, 1, .10, 'vegetable')


insert into nutrition_log_T (client_id, nutrition_log_date, food_item_id, servings_consumed) values
(1, '2020-12-07', 29, 6),
(1, '2020-12-07', 9, 2),
(1, '2020-12-07', 58, 4),
(1, '2020-12-07', 79, 1),
(1, '2020-12-07', 18, 1),
(1, '2020-12-07', 10, 2),
(1, '2020-12-07', 50, 1.5),
(1, '2020-12-07', 24, 5),
(1, '2020-12-07', 83, 4),
(1, '2020-12-08', 79, 1),
(1, '2020-12-08', 18, 1),
(1, '2020-12-08', 62, 6),
(1, '2020-12-08', 74, 2),
(1, '2020-12-08', 82, 3),
(1, '2020-12-08', 51, 3),
(1, '2020-12-08', 56, 1),
(1, '2020-12-08', 84, 1),
(1, '2020-12-09', 18, 1),
(1, '2020-12-09', 79, 1),
(1, '2020-12-09', 29, 6),
(1, '2020-12-09', 9, 4),
(1, '2020-12-09', 8, 1),
(1, '2020-12-09', 67, 8),
(1, '2020-12-09', 85, 1),
(1, '2020-12-09', 74, 1),
(1, '2020-12-09', 48, 2),
(1, '2020-12-09', 58, 6),
(1, '2020-12-10', 29, 6),
(1, '2020-12-10', 8, 1),
(1, '2020-12-10', 9, 2),
(1, '2020-12-10', 70, 5),
(1, '2020-12-10', 67, 4),
(1, '2020-12-10', 58, 4),
(1, '2020-12-10', 77, 0.5)
```

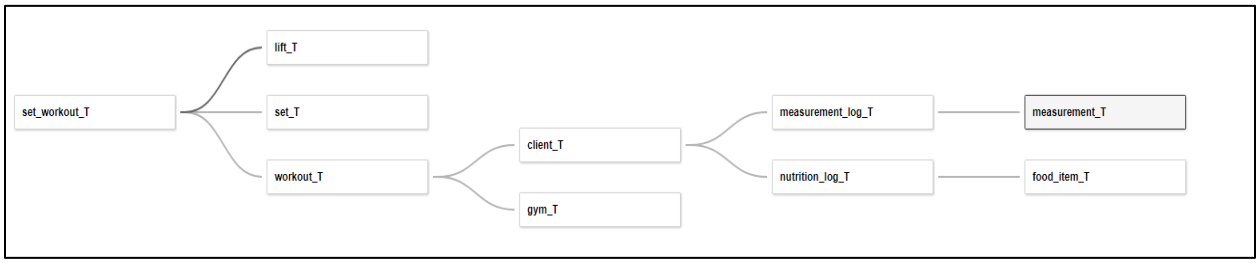
Answering Data Questions

```
-- 1. How did your 1 rep max of each lift change over time

/* Logic: pull a table that shows each lift by date and the
estimated 1RM, used a generic formula to calculate the 1RM.
This is only for first sets, since that is the set that reflects
true strength to the best degree when using reverse pyramid style*/

select top 1000 client_T.first_name+' '+client_T.last_name as clientname,
set_T.set_id, lift_T.lift_name, set_T.repetitions, set_T.resistance, set_T.RPE, workout_T.workout_date,
cast(set_T.resistance / (1.0278 - 0.0278 * set_T.repetitions) as int) as estimated_1rm
from set_T
join set_workout_T on set_workout_T.set_id = set_T.set_id
join workout_T on workout_T.workout_id = set_workout_T.workout_id
join client_T on client_T.client_id = workout_T.client_id
join lift_T on lift_T.lift_id = set_workout_T.lift_id
where set_T.set_number = 1


-- 2. How much body weight did you lose or gain

/*pull a table to show body weight measurements by date. further analysis
on this will be explored in the front end tool.*/
```

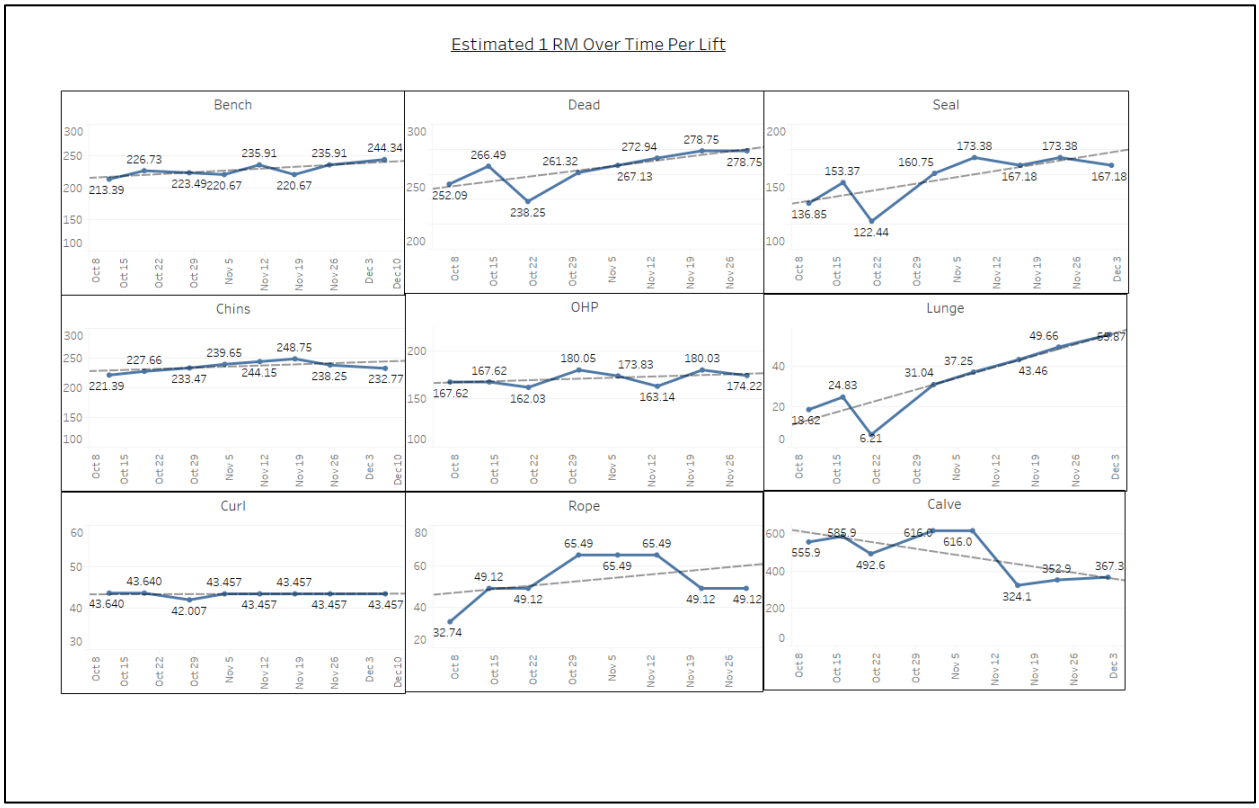
```sql
select  measurement_log_T.date_taken_on, measurement_log_T.recorded_value
from measurement_log_T
join measurement_T on measurement_T.measurement_id = measurement_log_T.measurement_id
where client_id = 1 and measurement_log_T.measurement_id = 1


-- 3. How much did the size of your muscle groups increase or decrease by

/*pull a table to show the measurements by date. Combined the recorded value and
unit of measurement fields into one for the sake of aesthetics*/

select  measurement_T.measurement_name, measurement_log_T.date_taken_on,
concat(cast(measurement_log_T.recorded_value as varchar), ' ',
cast(measurement_T.unit_of_measurement as char)) as recordedvalue
from measurement_log_T
join measurement_T on measurement_T.measurement_id = measurement_log_T.measurement_id
where client_id = 1 and measurement_log_T.measurement_id <> 1
order by measurement_T.measurement_name, measurement_log_T.date_taken_on


-- 4. How much money did you spend on average per day on food

/*used a sub query approach to pull together a table that shows
the total spend per day calculated by using the cost for food items
according to the data that is stored in the database*/

select top 1000 nutrition_log_date, sum(totalspend) as totalspend from
(select top 1000 nutrition_log_T.nutrition_log_date, nutrition_log_T.food_item_id,
food_item_T.food_item_name, nutrition_log_T.servings_consumed,
food_item_T.price_per_serving,
sum(food_item_T.price_per_serving * servings_consumed) as totalspend
from nutrition_log_T
join food_item_T on food_item_T.food_item_id = nutrition_log_T.food_item_id
where client_id = 1
group by nutrition_log_T.nutrition_log_date, nutrition_log_T.food_item_id,
food_item_T.food_item_name, nutrition_log_T.servings_consumed,
food_item_T.price_per_serving
order by nutrition_log_T.nutrition_log_date) sub
group by nutrition_log_date order by nutrition_log_date


-- 5. What kinds of foods did you eat the most

/*Pulled a table that shows both the count of how many times a particular
food category was logged as well as the the total cals and most importantly
the percentage of the total amount that the food category makes up in aggregate*/

select food_category,
sum(logcount) as logcount,
sum(totalcals) as totalcals,
sum(totalcals)/
(select sum(nutrition_log_T.servings_consumed *
food_item_T.calories_per_serving)
from nutrition_log_T
join food_item_T on food_item_T.food_item_id = nutrition_log_T.food_item_id)
as percenttotal from
        (select food_item_t.food_category,
        count(food_item_T.food_category) as logcount,
        sum(nutrition_log_T.servings_consumed * food_item_T.calories_per_serving) as totalcals
        from nutrition_log_T
        join food_item_T on food_item_T.food_item_id = nutrition_log_T.food_item_id
        group by food_item_T.food_category) sub
group by food_category
order by percenttotal desc
```

Implementation

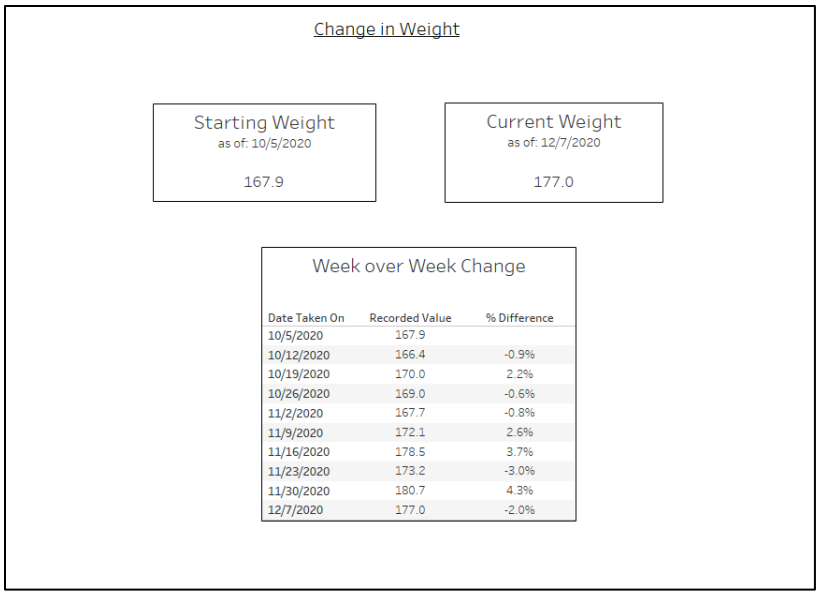
Used Tableau as the tool of my choice and built a dashboard for each business question. Here is the data model in Tableau.



Business question 1: How much did your 1 rep max of each lift change over time?

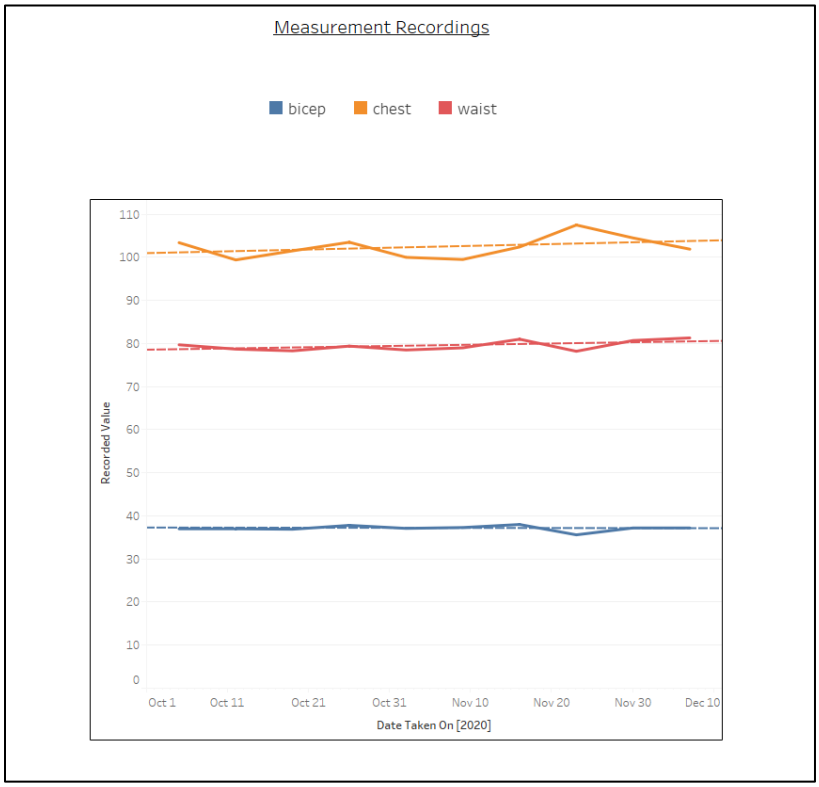


Business question 2: How much body weight did you lose or gain?



Date Taken On	Recorded Value	% Difference
10/5/2020	167.9	
10/12/2020	166.4	-0.9%
10/19/2020	170.0	2.2%
10/26/2020	169.0	-0.6%
11/2/2020	167.7	-0.8%
11/9/2020	172.1	2.6%
11/16/2020	178.5	3.7%
11/23/2020	173.2	-3.0%
11/30/2020	180.7	4.3%
12/7/2020	177.0	-2.0%

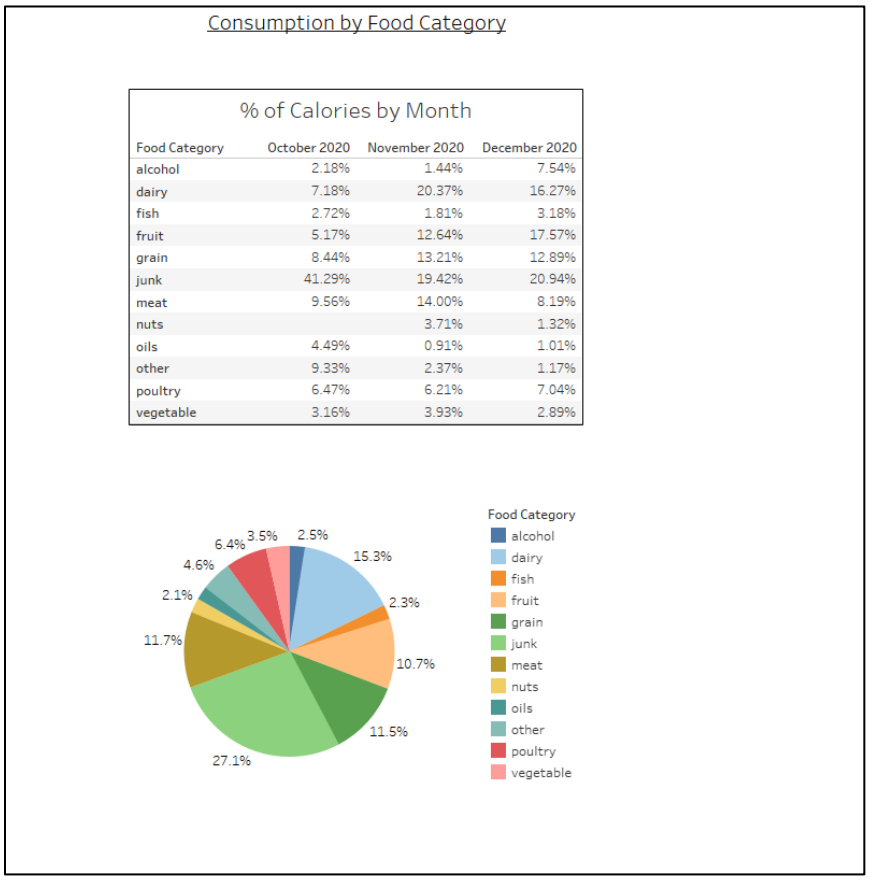
Business question 3: How much did the size of your muscle groups increase or decrease by?



Business question 4: How much money did you spend on average per day on food?

Nutrition Log Date	Food Item Name	Price Per Serving	Servings Consumed	totalspend
Null	salsa generic	$0.10		
	Total	$0.10		
10/7/2020	beef regula..	$1.35	2.0	$2.70
	cliff bar chu..	$1.50	2.0	$3.00
	french onio..	$0.50	3.0	$1.50
	grapes	$0.75	1.5	$1.13
	mission flo..	$0.36	2.0	$0.72
	siracha	$0.03	10.0	$0.30
	Total	$4.49	20.5	$9.35
10/8/2020	cliff bar chu..	$1.50	1.0	$1.50
	french onio..	$0.50	4.0	$2.00
	lasagna	$3.50	2.0	$7.00
	Total	$5.50	7.0	$10.50
10/9/2020	apple	$0.64	1.5	$0.96
	asparagus	$0.50	2.0	$1.00
	cedar plank..	$3.60	2.5	$9.00
	green pepp..	$0.10	1.0	$0.10
	homemade ..	$0.40	5.0	$2.00
	olive oil	$0.20	0.5	$0.10
	raw onion	$0.10	1.0	$0.10
	regular jum..	$0.18	8.0	$1.44
	Total	$5.72	21.5	$14.70
10/10/2020	boneless sk..	$1.07	6.0	$6.42
	breakfast s..	$1.44	1.0	$1.44
	green pepp..	$0.10	1.0	$0.10
	olive oil	$0.20	0.5	$0.10
	raw onion	$0.10	1.0	$0.10

Business question 5: What kinds of food did you eat the most?



% of Calories by Month

Food Category	October 2020	November 2020	December 2020
alcohol	2.18%	1.44%	7.54%
dairy	7.18%	20.37%	16.27%
fish	2.72%	1.81%	3.18%
fruit	5.17%	12.64%	17.57%
grain	8.44%	13.21%	12.89%
junk	41.29%	19.42%	20.94%
meat	9.56%	14.00%	8.19%
nuts		3.71%	1.32%
oils	4.49%	0.91%	1.01%
other	9.33%	2.37%	1.17%
poultry	6.47%	6.21%	7.04%
vegetable	3.16%	3.93%	2.89%

Reflection

Overall, I am very pleased with how the database turned out and am excited to actually use the database for personal use. Prior to this, I was tracking all of my workout information in a notebook.

At the beginning of the project, I thought that all of the steps would flow in a linear way. As it turns out, this was not the case. There were times where I had to jump around between business rules, definitions, modeling, and design. It was not a straight line.

There was certainly some trial and error along the way. I kept the original conceptual, logical, and normalized models in this document for reference, while adding a fourth which is the revised normalized model. You may notice that there were some changes.

At one point I went back and made a structural change after already having coded the tables. I identified an issue with associative entities. This caused me to also have to change some object names and data types in the data definition language. Another call out was that I got rid of the muscle group entity altogether. After thinking about it some more, it didn't make sense to keep this. The measurement data should be sufficient enough.

Before I coded anything in SQL, I mapped out everything in Excel. This was very helpful because it helped me get a feel for the structure and data. I had to manually enter all of my data into Excel though which took a while. From there, I used excel formulas to create the SQL code.

Creating the stored procedures was the most challenging task, but a very important one because they ensure that the data being entered conforms with the business rules. It took me a while to figure out how to get it this work. Although the stored procedures function properly, there is still some tuning needed. I would like to show an error message if the rule is violated, but I could not figure out how to do so. For the sake of time I decided to move on.

A key takeaway for me is to always keep things as lean as possible. This is something that I would want to consider if ever doing this again. Just like my workout philosophy is that less is more, I would argue the same for a database.

Looking forward, I would really like to create my own personal mobile or web application that provides a user interface for entering data, ideally on my phone. This is a tall task though, because not only would there be more SQL code required, but I am no expert with application development. But I think that would be a great learning experience to continue to develop my skills in data science.

Again, I am very happy with how the database turned out, I put a lot of attention to detail into creating it to ensure that it will operate as I want it to. It is nice that I will continue to use this database beyond the course because that means that it is practical.