Database Final Project

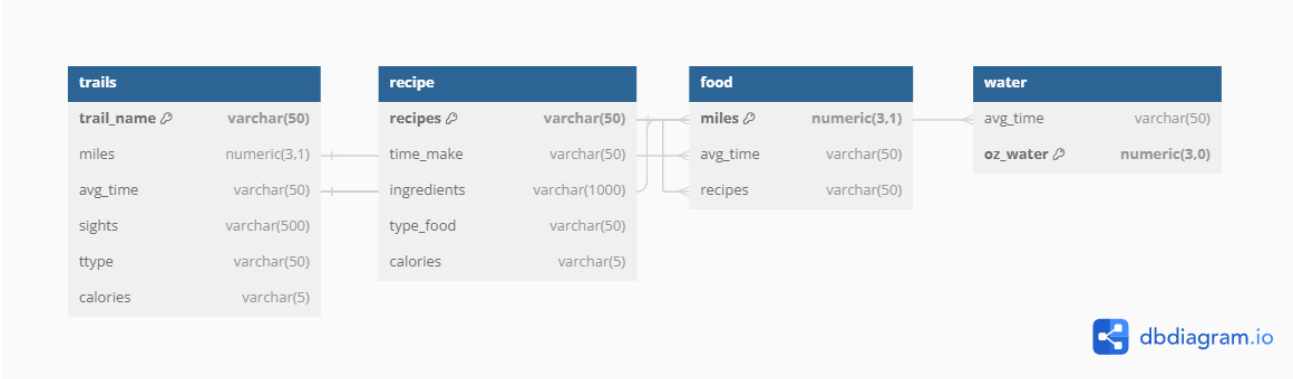
Ashleigh Frank

EPPS 3654

Introduction

The database consists of trails and food to eat after a hike. The audience could be the general public but more specifically people looking for healthy activities while eating the right food after or before a long hike. Cardio requires different types of food to be eaten after activity in comparison to bulking or weight training. High carb and high protein foods provide the best nutrients to eat for hiking. Consumption of food can be either before or after a hike which is dependent on an individual's personal preference. Water consumption for hikes is based on how long the hike is. On average, every 30 minutes, 8 ounces of water is used to replenish the body. The average time of a hike is dependent on the speed of a person and their walking speed. Therefore, average times cannot always be 100% accurate due to variances in a person's speed.

The database will consist of a list of trails that will tell you how many miles the trail is, the average time, the sights you might see on the trail, the trail type and calories expended on a hike. Based on the miles of the trail or the average time, recipes and the food type will appear. Food type will show what the food primary contains such as protein, fiber, carbs, fruits or vegetables. The third relation will be the recipes which show the recipes, the time it takes to make, the ingredients and the calories of the food. The last relation will be the water relation which shows how many ounces of water to drink after the hike.



This is the entity relationship table to show the connections the relations have with each other. The trail_name is the primary key in the trails relation. The recipes is the primary key in the recipe relation, miles is the primary key in the food relation and oz_water is the primary key in the water relation. The foreign keys in food are avg_time which references trails avg_time, recipes references recipe's entity, recipes, and miles which references trails miles. The water relation's avg_time references trails avg_time. The method used will be data manipulation language or data definition language.

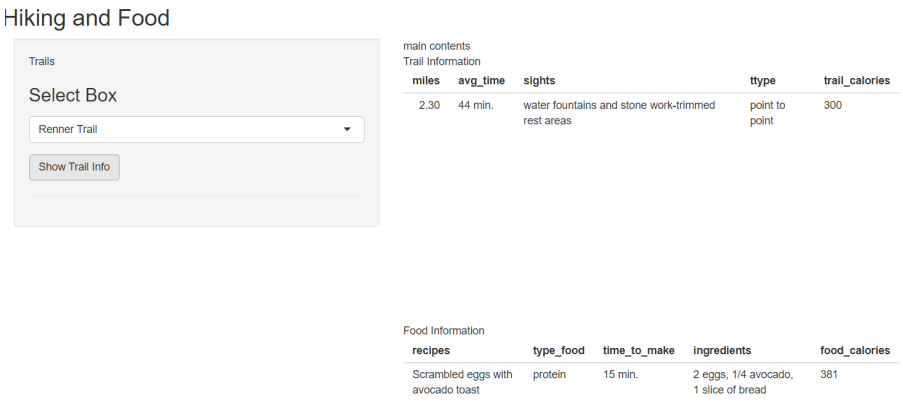
Database

In the beginning creation of the database, the trails are inputted in and their information through using the information that is posted on alltrails.com. The website holds the description of the trail, average time, and the trail route. The calories were calculated based on the miles of a hike and I used the weight of a 150 pound person to put in the amount of calories that would be expended on a hike. For every mile, a 150 pound person would burn 50 to 60 calories with a walking pace of 3 to 4 mph. I looked online for various articles about the best food to eat for a hike. Most of the food types are carbs and protein or a mixture with fruits. I would place the information about the food, ingredients needed and the calories into the recipe relation and the food relation would have the recipe name so that the miles and average time of the hike can reference the recipe relation. One article I found mentioned about water and the general rule of water consumption and I calculated the ounces of water based on the average time of the hike.

```
INSERT INTO trails (trail_name, miles, avg_time, sights, ttype, calories)
VALUES
    ('Renner Trail', 2.3, '44 min.', 'water fountains and stone work-trimmed rest areas', 'point to point', '300'),
    ('Huck Finn', 1.8, '34 min.', 'Bridge crossing, quiet', 'Out & back', '180'),
    ('UTD Trail', 4.4, '83 min.', 'University trails, popular, explore UTD', 'Loop', '440'),
    ('Harry Moss', 1.6, '30 min.', 'forested area, hills. can connect to other trails', 'Loop', '150'),
    ('Cottonwood park perimeter', 0.8, '15 min.', 'easy trail, around a community park', 'Loop', '80'),
    ('Glenville Trail', 4.5, '78 min.', 'completes series of other trails, easy trial', 'Out & back', '400');

INSERT INTO recipe (recipes, time_make, ingredients, type_food, calories)
VALUES
    ('overnight oats', 'overnight, 8 hrs', 'combine 1/2 cup rolled oats, 1 tablespoon ground flaxseed, 2/3 cups milk, 1/3 cup pla
    ('Cheese and Crackers', '5 min.', 'Package of cheese and package of crackers', 'protein & carbs', '104'),
    ('Trail Mix', '5 min.', 'cashews, peanuts or other nuts, raisins or dried cranberries, m&m, pretzels', 'protein & carbs', '36
    ('Peanut Butter and Banana Sandwich', '5 min.', '2 slices bread, peanut butter and bananas', 'Protein and fruit', '300'),
    ('Scrambled eggs with avocado toast', '15 min.', '2 eggs, 1/4 avocado, 1 slice of bread', 'protein', '381');

INSERT INTO food (miles, avg_time, recipes)
VALUES
    (1.6, '30 min.', 'Cheese and Crackers'),
    (4.4, '83 min.', 'overnight oats'),
    (4.5, '78 min.', 'Trail Mix'),
    (1.8, '34 min.', 'Cheese and Crackers'),
    (0.8, '15 min.', 'Peanut Butter and Banana Sandwich'),
    (2.3, '44 min.', 'Scrambled eggs with avocado toast');

INSERT INTO water (avg_time, oz_water)
VALUES
    ('44 min.', 12),
    ('30 min.', 8),
    ('78 min.', 16),
    ('83 min.', 24),
    ('34 min.', 9),
    ('15 min.', 4);
```

Application- Shiny & RStudio

The application contained a sidebar that can be closed to expand information that was selected. I used the drop down selection tool to hold the trails that a user would select and an action button that needed to be pressed before displaying information about the trail. This was done so that information wasn't automatically placed before they picked a trail. There are three cards that display information about the trail, recipe trail and water consumption card. The trail information card displays the trails' miles, average time to complete, trail type, and how many calories burned. The food information card displays the recipe, food type, time to make, ingredients and the calories. Water information card displays the amount of ounces needed for a hike.



I found calculating the calories based on a 150 pound person to be very ambiguous and difficult to accurately display how many calories a person would burn on a hike especially with different weights for a person. For example, a 120 pound person would burn on average 65 calories per mile while a 150 pound person would expend around 50. Without knowing the weights of a user, I would have to put a disclaimer that the calories output would be based on a 150 pound person and couldn't be completely accurate. The disclaimer was at the bottom of the application and said "Calorie output is dependent on weight. Calorie output is based on 150 pound(LB) weight. Water intake is based on time. General rule is every 30 min. is equal to 8 ounces. The times displayed for a hike are the average hike completion time and will not be completely accurate for everyone due to differences in speed."

Food Information

recipes	type_food	time_to_make	ingredients	food_calories
Scrambled eggs with avocado toast	protein	15 min.	2 eggs, 1/4 avocado, 1 slice of bread	381

Water Information

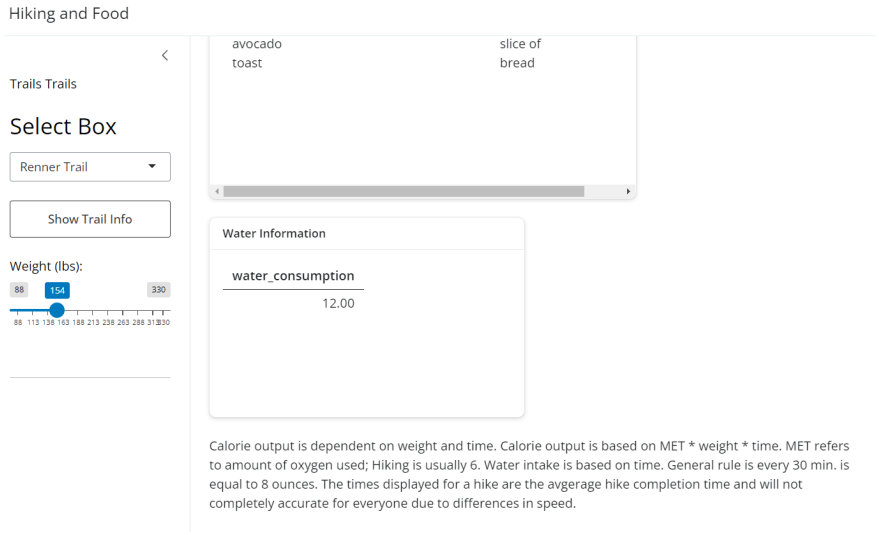
water_consumption
12.00

Calorie output is dependent on weight. Calorie output is based on 150 pound(LB) weight. Water intake is based on time. General rule is every 30 min. is equal to 8 ounces. The times displayed for a hike are the average hike completion time and will not be completely accurate for everyone due to differences in speed.

In comparison to how the water table is ambiguous, this method was far more ambiguous, and I changed it for people to place input on their weights, and the calories burned will display on the application.

When changing the application for the input of the user's weight, I had to change how the calories were inputted using an equation to determine the calories burned. The calorie output is based on weight, how long the exercise is and the amount of oxygen used. The equation used was: calories burned per minute = (MET * weight in kg * 3.5)/200. The weight input had to be converted to kg by multiplying the pounds by 0.4536. MET is the Metabolic Equivalent of Task which is a measure of oxygen used and hiking is around 6 MET to 7 MET. The time of the trail was converted to hours in the application then was multiplied by the equation to find the calories burned for the entire session. This way of calculating the calories becomes more accurate for the individual because it allows user input on the weight, but it isn't 100% accurate because the ability to tell how much oxygen someone uses is different based on hills, if they are carrying a

backpack and how much that weighs. I included a disclaimer of how the calories burned are calculated and information about the water consumption with its general rule of every 30 minutes, 8 ounces of water is best to be consumed. It is as follows: “Calorie output is dependent on weight and time. Calorie output is based on MET * weight * time. MET refers to the amount of oxygen used; Hiking is usually 6. Water intake is based on time. General rule is every 30 min. is equal to 8 ounces. The times displayed for a hike are the average hike completion time and will not be completely accurate for everyone due to differences in speed.” Weight input is done through a slider, and I kept the drop down selection button as well as the action button that will display all the information once it was clicked.



In R-Studio the packages that were downloaded were shiny, bslib, DBI and RPostgres. Bslib helps to create the cards that are on the application. On the ui side, it is a page sidebar with a main panel that holds four cards.

```
# Define UI
ui <- page_sidebar(
  title = "Hiking and Food",
  sidebar = sidebar("Trails",
                    'Trails',
                    selectInput("select", label = h3("Select Box"),
                                choices = c("Renner Trail" = 'Renner Trail', "Huck Finn" = 'Huck Finn', "UTD Trail" = 'UTD Trail', "Harry Moss" = 'Harry Moss', "Cottonwood park perimeter" = 'Cottonwood park perimeter'
                                selected = 'Renner Trail'),
                    actionButton("showInfoBtn", "Show Trail Info"),
                    sliderInput("weightInput", label = "Weight (lbs):", min = 88, max = 330, value = 154),
                    hr()
  ),
  mainPanel(
    "main contents",
    fluidRow(
      column(8,
             card(height = 350,
                  full_screen = TRUE,
                  card_header("Trail Information"),
                  div(id = "trailTableDiv", class = "scrollable-table", tableOutput("trailTable"))
             )
      ),
      column(8,
             card(height = 350,
                  full_screen = TRUE,
                  card_header("Food Information"),
                  div(id = "foodTableDiv", class = "scrollable-table", tableOutput("foodTable"))
             )
      ),
      column(6,
             card(height = 250,
                  full_screen = TRUE,
                  card_header("Water Information"),
                  div(id = "waterTableDiv", class = "scrollable-table", tableOutput("waterTable"))
             )
      ),
      column(12,
             p("Calorie output is dependent on weight and time. Calorie output is based on MET * weight * time. MET refers to amount of oxygen used; Hiking is usually 6. Water intake is based on time. General rule is
    )
  )
)
```

The server logic renders three different tables for the first three cards. There are three different SQL statements: (SELECT t.miles, t.avg_time, t.sights, t.ttype FROM trails t WHERE t.trail_name = '", input$select, "'), (SELECT f.recipes, r.type_food, rec.time_make AS time_to_make, rec.ingredients, rec.calories AS food_calories FROM trails t JOIN food f ON t.miles = f.miles AND t.avg_time = f.avg_time JOIN recipe r ON f.recipes = r.recipes JOIN (SELECT recipes, ingredients, calories, time_make FROM recipe) rec ON f.recipes = rec.recipes WHERE t.trail_name = '", input$select, "'), and (SELECT w.oz_water AS water_consumption FROM trails t JOIN water w ON t.avg_time = w.avg_time WHERE t.trail_name = '", input$select, "'). This allows for each table to only pull from it's SQL statement to place on the application card. The server logic contains the equation to take the weight input from the user and use it as to create the calories output.

```r
# Define server logic
server <- function(input, output) {
  # Define MET value (based on kg) and calorie calculation function
  met_value_kg <- 6  # MET value for hiking based on kg
  lbs_to_kg <- 0.453592  # Conversion factor from lbs to kg

  calculateCalories <- function(weight_lbs, time_hours) {
    weight_kg <- weight_lbs * lbs_to_kg  # Convert weight from lbs to kg
    calories_burned <- met_value_kg * weight_kg * time_hours
    return(calories_burned)
  }

  # Inside the render function for trail information
  output$trailTable <- renderTable({
    req(input$showInfoBtn) # action button click

    # Fetch weight from input
    weight <- input$weightInput

    # Connect to the database and fetch trail data including weight
    conn <- dbConnect(RPostgres::Postgres(), dbname = 'projecttest',
                      host = '127.0.0.1', port = 5432,
                      user = 'postgres', password = 'DaLlas00!')

    postgres_sql <- paste0("SELECT t.miles, t.avg_time, t.sights, t.ttype
                           FROM trails t
                           WHERE t.trail_name = '", input$select, "'")

    trail_data <- dbGetQuery(conn, postgres_sql)
    dbDisconnect(conn)

    # Calculate calories burned using weight input and trail time
    time_hours <- as.numeric(gsub("[^0-9.]", "", trail_data$avg_time)) / 60  # Convert avg_time to hours
    trail_data$calories_burned <- calculateCalories(weight, time_hours)

    return(trail_data)
  })
```

```r
output$foodTable <- renderTable({
  req(input$showInfoBtn) # action button click
  # Postgres connection
  postgres_conn <- dbConnect(RPostgres::Postgres(), dbname = 'projecttest',
                             host = '127.0.0.1', port = 5432,
                             user = 'postgres', password = 'DaLlas00!')

  # SQL query to fetch food information
  postgres_sql <- paste0("SELECT f.recipes, r.type_food, rec.time_make AS time_to_make, rec.ingredients, rec.calories AS food_calories
    FROM trails t
    JOIN food f ON t.miles = f.miles AND t.avg_time = f.avg_time
    JOIN recipe r ON f.recipes = r.recipes
    JOIN (SELECT recipes, ingredients, calories, time_make FROM recipe) rec ON f.recipes = rec.recipes
    WHERE t.trail_name = '", input$select, "'")

  # Execute SQL query and fetch data
  conn <- postgres_conn
  food_data <- dbGetQuery(conn, postgres_sql)

  # Close the database connection
  on.exit(dbDisconnect(conn), add = TRUE)

  return(food_data)  # Return the food data frame
})

output$waterTable <- renderTable({
  req(input$showInfoBtn) # action button click
  # Postgres connection
  postgres_conn <- dbConnect(RPostgres::Postgres(), dbname = 'projecttest',
                             host = '127.0.0.1', port = 5432,
                             user = 'postgres', password = 'DaLlas00!')

  # SQL query to fetch water information
  postgres_sql <- paste0("SELECT w.oz_water AS water_consumption
    FROM trails t
    JOIN water w ON t.avg_time = w.avg_time
    WHERE t.trail_name = '", input$select, "'")

  # Execute SQL query and fetch data
  conn <- postgres_conn
  water_data <- dbGetQuery(conn, postgres_sql)

  # Close the database connection
  on.exit(dbDisconnect(conn), add = TRUE)

  return(water_data)  # Return the water data frame
})
```

Using RPostgres allows the application to reference the database through Postgres and pg admin. The photos above show the code I used to locally run the database. During the implementation process of the project I used SQLite and the application is posted on shinyapps.io which is posted through my GitHub account. The URL link to the application is https://ashfrank1.shinyapps.io/ProjectApplication/ and the URL to my GitHub repository is https://github.com/ashfrank1/ashfrank1 and the website is https://ashfrank1.github.io/ashfrank1/ .

Full Study Expectations

With a full study or more time, I would hope to add more trails, and recipes. Additionally, I was hoping to add instructions on how to make certain food. Some recipes are simple like sandwiches and cheese and crackers which do not need the instructions but some might be new to user's, like overnight oats, and adding in instructions can allow users to have all information

regarding food. Implementing it would look something like showing the recipe first then clicking a button to receive the information on how to make it. Since there are many different ways to show instructions, I feel having an action button to show the information would be the best, so that the page and its cards are not crowded. Another thing was to add more recipes that will be shown on the food information card. This would just have more rows on the table card and allow for users to scroll the different types of food they would like and can pick from there. I would consider allowing user input on allergens, food avoidance and foods they love so that the food becomes more tailored to the individual. This requires more food options in the database and the ability to exclude certain types or allergens. Another thing is adding in geospatial location so that users can choose a location or their current location and show the nearest trails. In hopes that the database can encompass a larger area, such as all of Dallas and surrounding cities, this would allow increased visits for the application and be easier for the user to choose if they want to travel to a hike or stay in their area for a quick hike.

Since learning about the best types of food, arguments for eating before or after a hike, and understanding calories burned and water consumption, I thought about holding an informational section that can state all this information. Users could be willing to understand their body and the exercise they are partaking in as well as best habits. Some information that can be included would be the benefits of eating before or after a hike. Eating before a hike can help store up energy, maintain endurance and prevent fatigue during a hike while eating after a hike can help replenish lost nutrients, promote muscle recovery and prevent dehydration and fatigue after the hike. The decision to eat before or after a hike is up to the individual but both contain its own set of pros and cons. The main factors that should be considered are the length of the hike and the intensity. The best food to eat for a hike is carbohydrates, protein, healthy fats

and hydration rich foods. The food to avoid is high-fiber, high-fat, spicy and processed food as they can cause stomach discomfort and create feelings of lethargy. Understanding food and when to eat can help foster the individual to make healthy decisions for themselves and foster a learning environment for the hobby they want to get into.

References

AllTrails. (2019). *AllTrails: Trail Guides & Maps for Hiking, Camping, and Running | AllTrails*. AllTrails.com. https://www.alltrails.com/

Thad. (2023, May 18). *What To Eat Before, During, and After A Hike*. The Fun Outdoors. https://thefunoutdoors.com/hiking/what-to-eat-before-a-hike/

What to eat after a long walk | Sensible foods to eat after hiking - Walking Academy. (2021, March 12). https://walkingacademy.com/what-to-eat-after-a-long-walk-sensible-foods-to-eat-after-hiking/