University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: Apr 5, 2024

Group Number: 40

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Nam Nguyen	89939383	e5n5h	nnguye10@student.ubc.ca
Ubada Raja	99035578	m4p5v	ubada@student.ubc.ca
Hieu Le	76067321	s3b9t	xle@student.ubc.ca

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

a. Project Description

- We've developed a full-stack web application that empowers users to effortlessly track their food intake and exercise routines. By simply inputting the foods they've consumed and the activities they've engaged in, our platform calculates their daily nutritional intake and monitors their progress toward their fitness objectives.

b. A description of how your final schema differed from the schema you turned in

Our final schema has separated Eats_Meal into Eats and Meal as they are many-to-many relationships (suggestion from our T.A).

c. A copy of the schema and screenshots that show what data is present in each relation after the SQL script from item #2 is run

User(UID, UserName, Weight, Height, **GoalID** , **RecommendID**)
Activities (CaloriesPerHour, ActivityName)
Exercise_Activities(Date, **ActivityName**, Hour)
Nutrition(Protein, Fat, Carb, Calories)
FoodNutrition(Protein, Fat, Carb, FoodName)
Record_Hold (RecordID, **UID**, Date, Calories, Protein, Fat)
Goal (GoalID, Calories, Protein, Fat)
Recommendation (RecommendID, **RecommendedFood**)
Vegetarian (**UID**)
Eats (**UID**, **MealName**, Date, Amount)
Meal(Name, MealType)
Meal_Has_Food (**MealName**, **FoodName**)
Favourite_1 (**UID**, FoodName)
Favourite_2 (**FoodName**, TypeFood)
User_Do_Exercise (**UID**, **Date**, **ActivityName**)

TABLE User

a. Project Description

- We've developed a full-stack web application that empowers users to effortlessly track their food intake and exercise routines. By simply inputting the foods they've consumed and the activities they've engaged in, our platform calculates their daily nutritional intake and monitors their progress toward their fitness objectives.

b. A description of how your final schema differed from the schema you turned in
Our final schema has separated Eats_Meal into Eats and Meal as they are many-to-many relationships (suggestion from our T.A).

c. A copy of the schema and screenshots that show what data is present in each relation after the SQL script from item #2 is run

User(UID, UserName, Weight, Height, **GoalID** , **RecommendID**)

Activities (CaloriesPerHour, ActivityName)

Exercise_Activities(Date, **ActivityName**, Hour)

Nutrition(Protein, Fat, Carb, Calories)

FoodNutrition(Protein, Fat, Carb, FoodName)

Record_Hold (RecordID, **UID**, Date, Calories, Protein, Fat)

Goal (GoalID, Calories, Protein, Fat)

Recommendation (RecommendID, **RecommendedFood**)

Vegetarian (**UID**)

Eats (**UID**, **MealName**, Date, Amount)

Meal(Name, MealType)

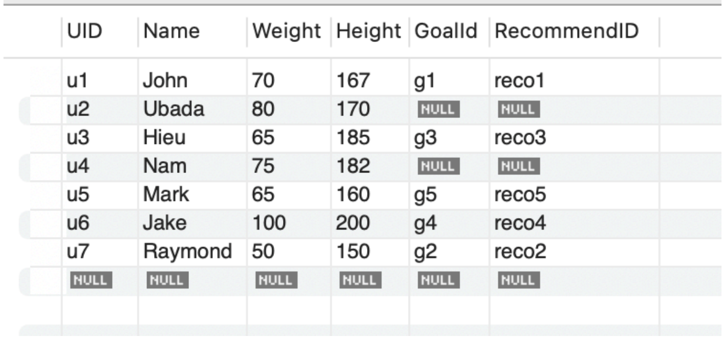
Meal_Has_Food (**MealName**, **FoodName**)

Favourite_1 (**UID**, FoodName)

Favourite_2 (**FoodName**, TypeFood)

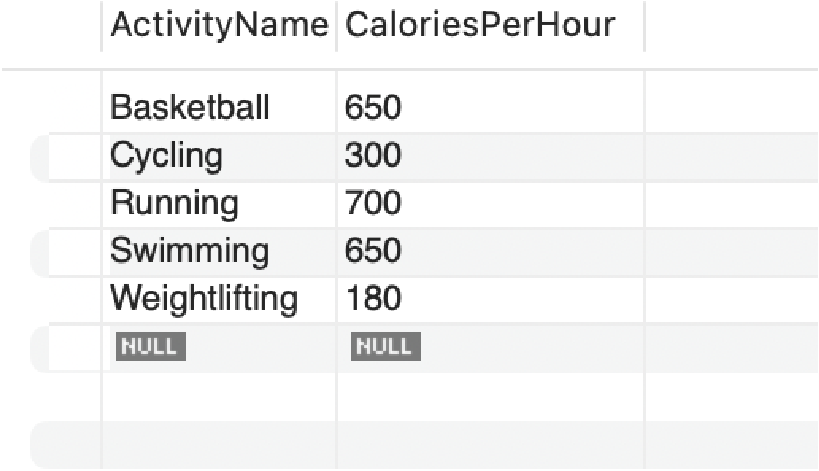
User_Do_Exercise (**UID**, **Date**, **ActivityName**)

TABLE User



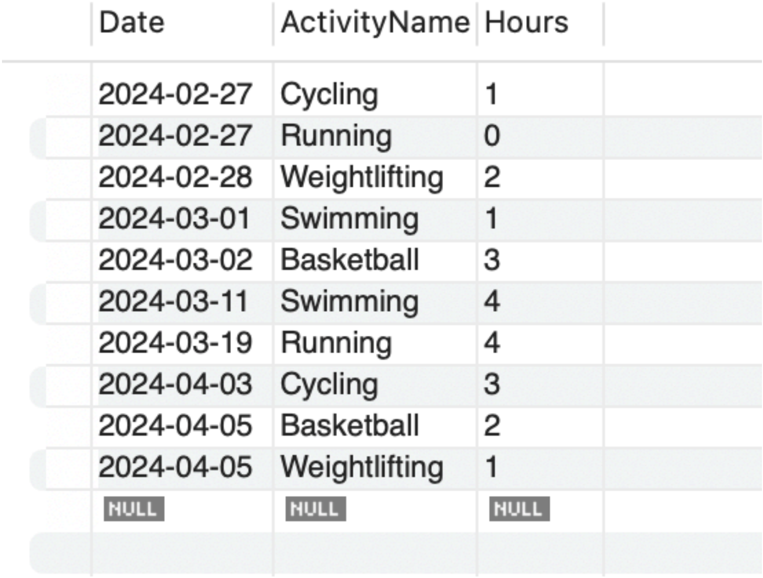
UID	Name	Weight	Height	GoalId	RecommendID
u1	John	70	167	g1	reco1
u2	Ubada	80	170	NULL	NULL
u3	Hieu	65	185	g3	reco3
u4	Nam	75	182	NULL	NULL
u5	Mark	65	160	g5	reco5
u6	Jake	100	200	g4	reco4
u7	Raymond	50	150	g2	reco2
NULL	NULL	NULL	NULL	NULL	NULL

TABLE Activities



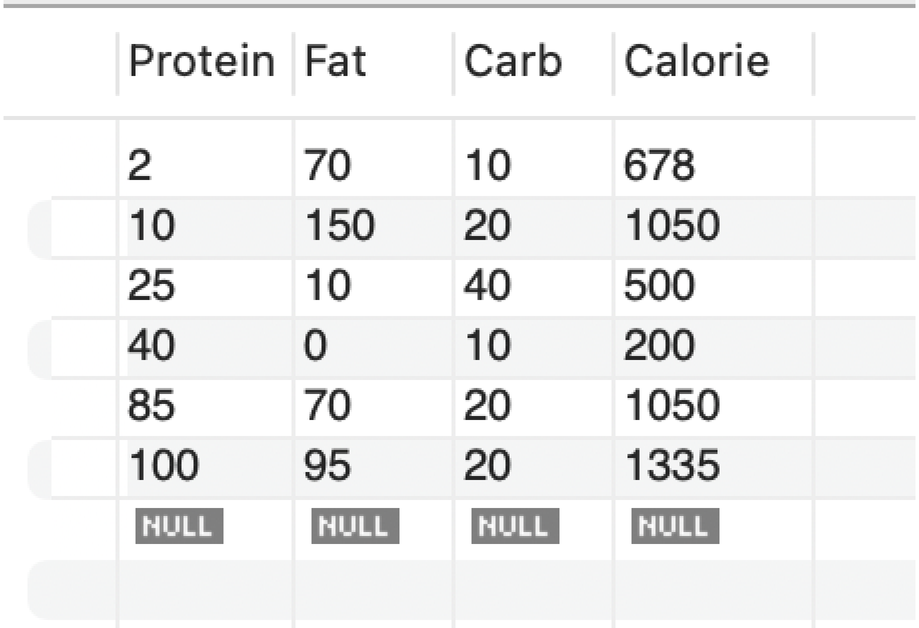
ActivityName	CaloriesPerHour
Basketball	650
Cycling	300
Running	700
Swimming	650
Weightlifting	180
NULL	NULL

TABLE Exercise_Activities



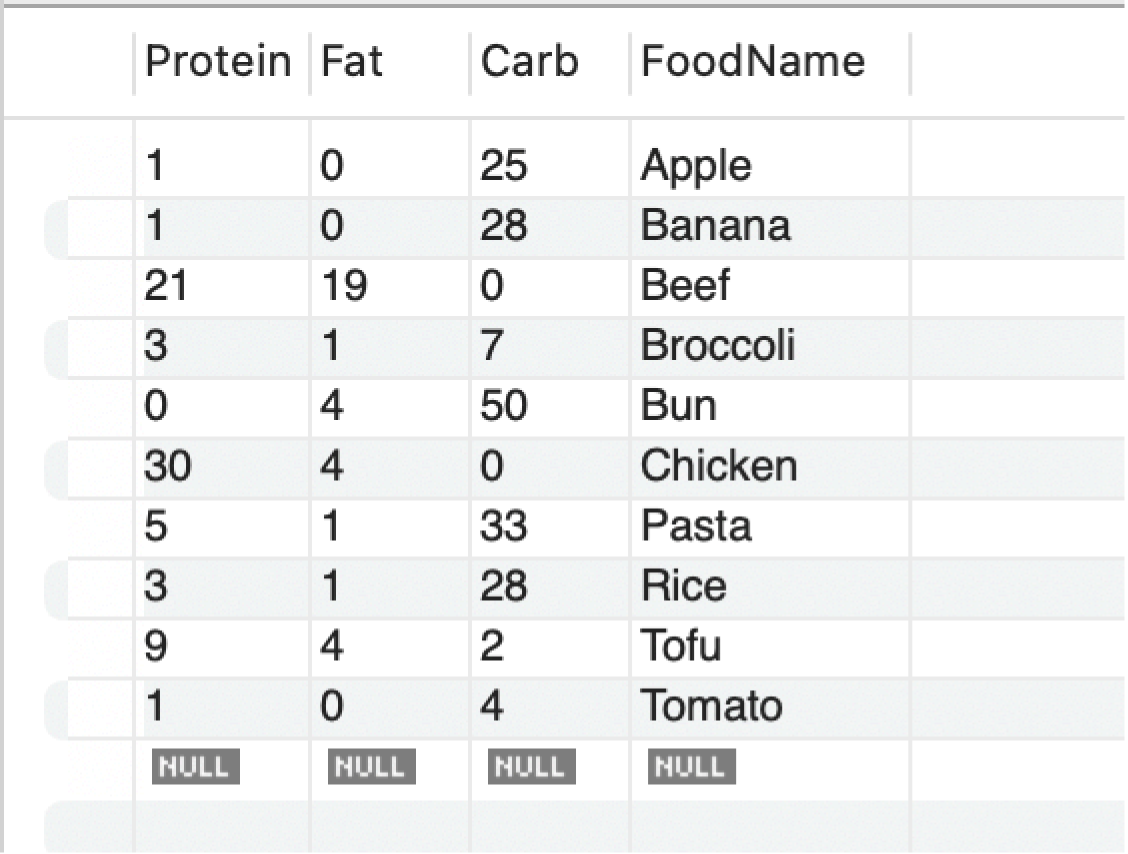
Date	ActivityName	Hours
2024-02-27	Cycling	1
2024-02-27	Running	0
2024-02-28	Weightlifting	2
2024-03-01	Swimming	1
2024-03-02	Basketball	3
2024-03-11	Swimming	4
2024-03-19	Running	4
2024-04-03	Cycling	3
2024-04-05	Basketball	2
2024-04-05	Weightlifting	1
NULL	NULL	NULL

TABLE Nutrition



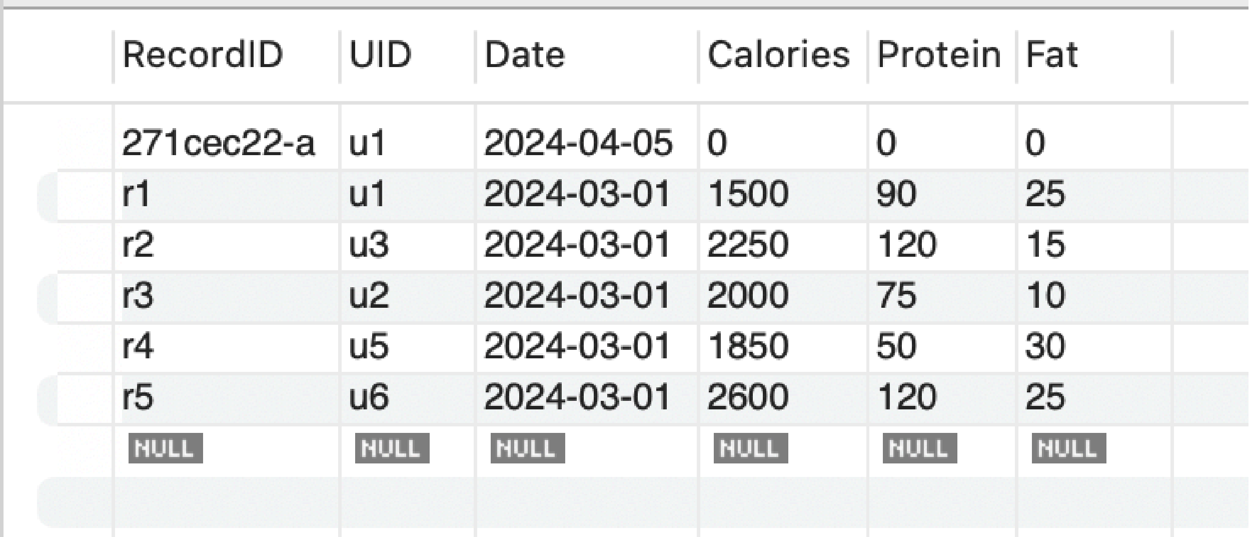
Protein	Fat	Carb	Calorie
2	70	10	678
10	150	20	1050
25	10	40	500
40	0	10	200
85	70	20	1050
100	95	20	1335
NULL	NULL	NULL	NULL

Table FoodNutrition



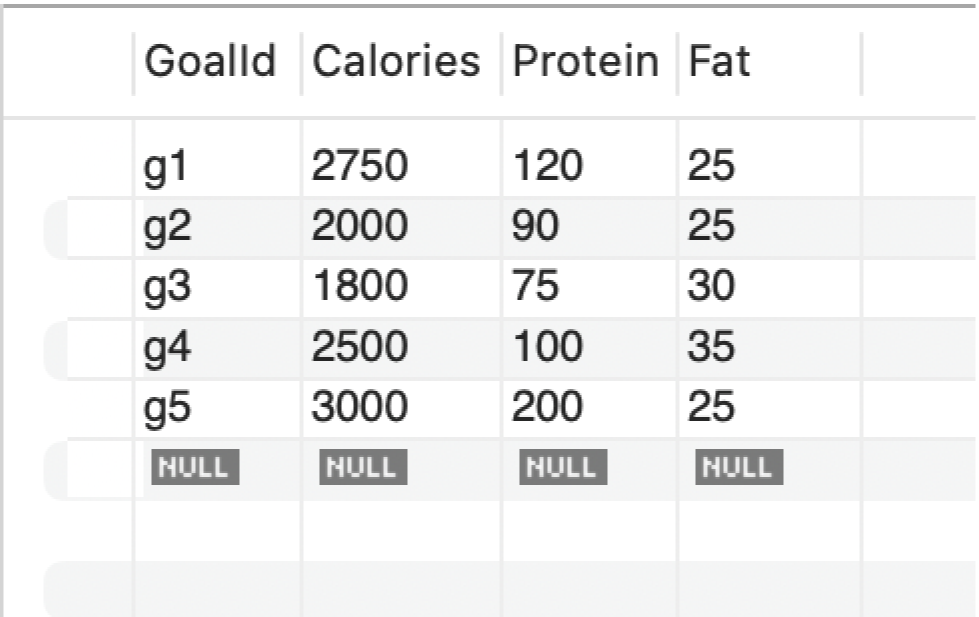
Protein	Fat	Carb	FoodName
1	0	25	Apple
1	0	28	Banana
21	19	0	Beef
3	1	7	Broccoli
0	4	50	Bun
30	4	0	Chicken
5	1	33	Pasta
3	1	28	Rice
9	4	2	Tofu
1	0	4	Tomato
NULL	NULL	NULL	NULL

TABLE Record_Hold



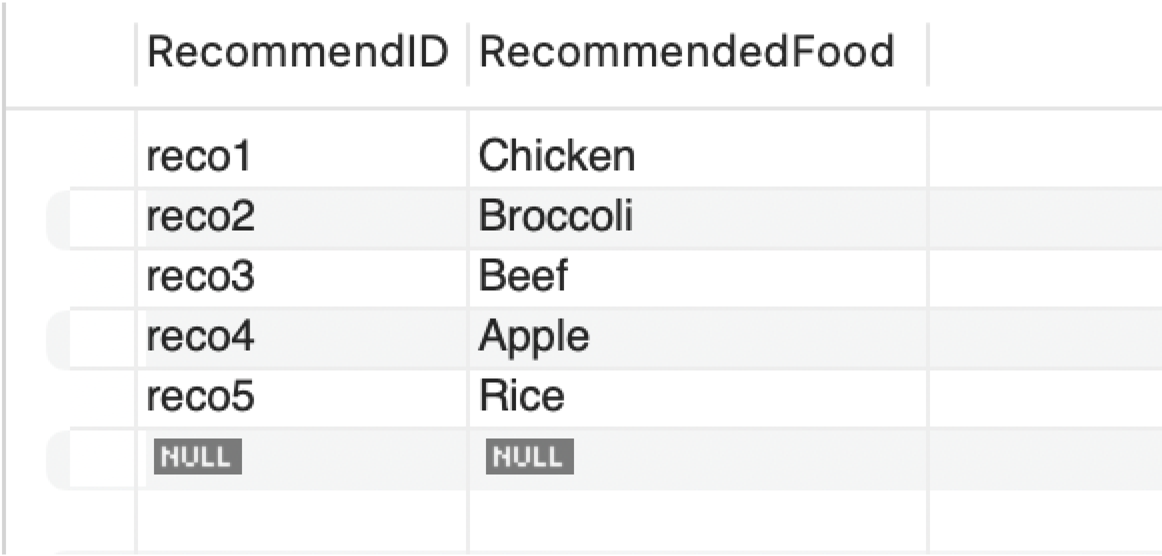
RecordID	UID	Date	Calories	Protein	Fat
271cec22-a	u1	2024-04-05	0	0	0
r1	u1	2024-03-01	1500	90	25
r2	u3	2024-03-01	2250	120	15
r3	u2	2024-03-01	2000	75	10
r4	u5	2024-03-01	1850	50	30
r5	u6	2024-03-01	2600	120	25
NULL	NULL	NULL	NULL	NULL	NULL

TABLE Goal



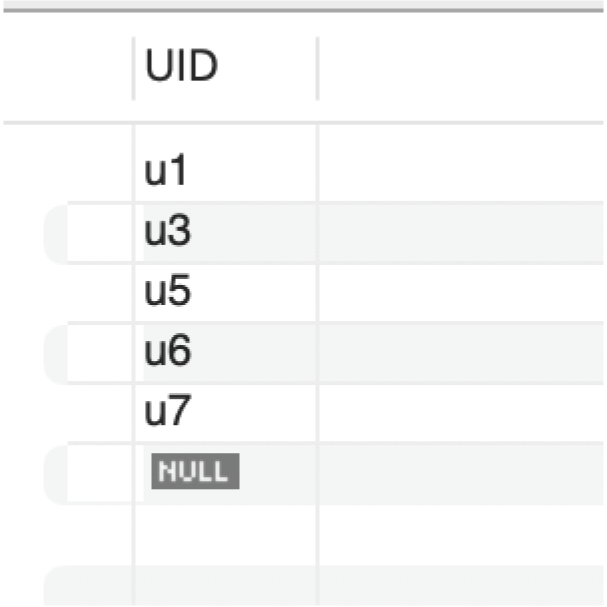
GoalId	Calories	Protein	Fat
g1	2750	120	25
g2	2000	90	25
g3	1800	75	30
g4	2500	100	35
g5	3000	200	25
NULL	NULL	NULL	NULL

TABLE Recommendation



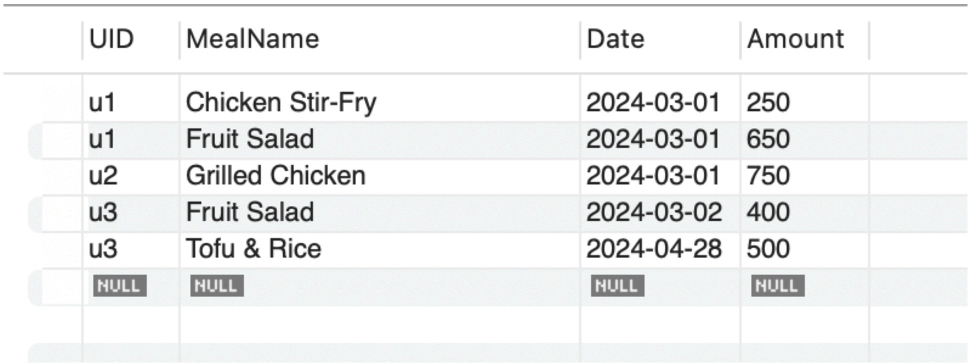
RecommendID	RecommendedFood
reco1	Chicken
reco2	Broccoli
reco3	Beef
reco4	Apple
reco5	Rice
NULL	NULL

TABLE Vegetarian



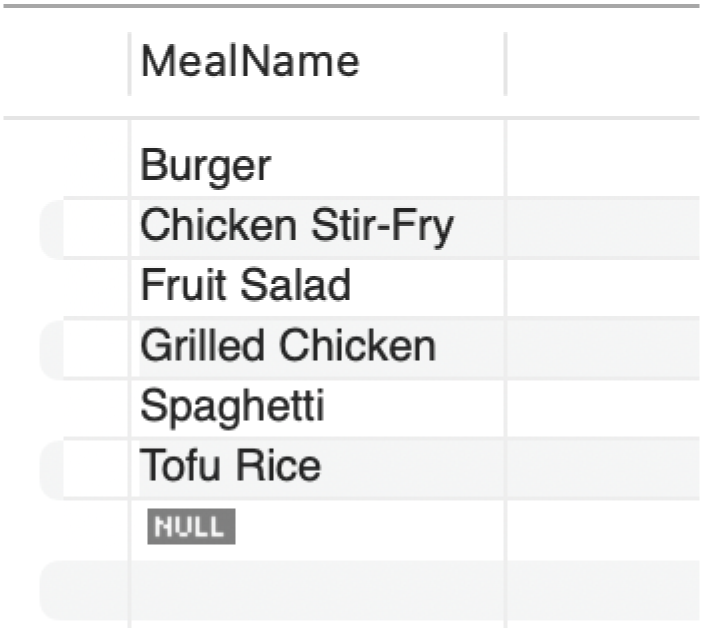
UID
u1
u3
u5
u6
u7
NULL

TABLE Eats



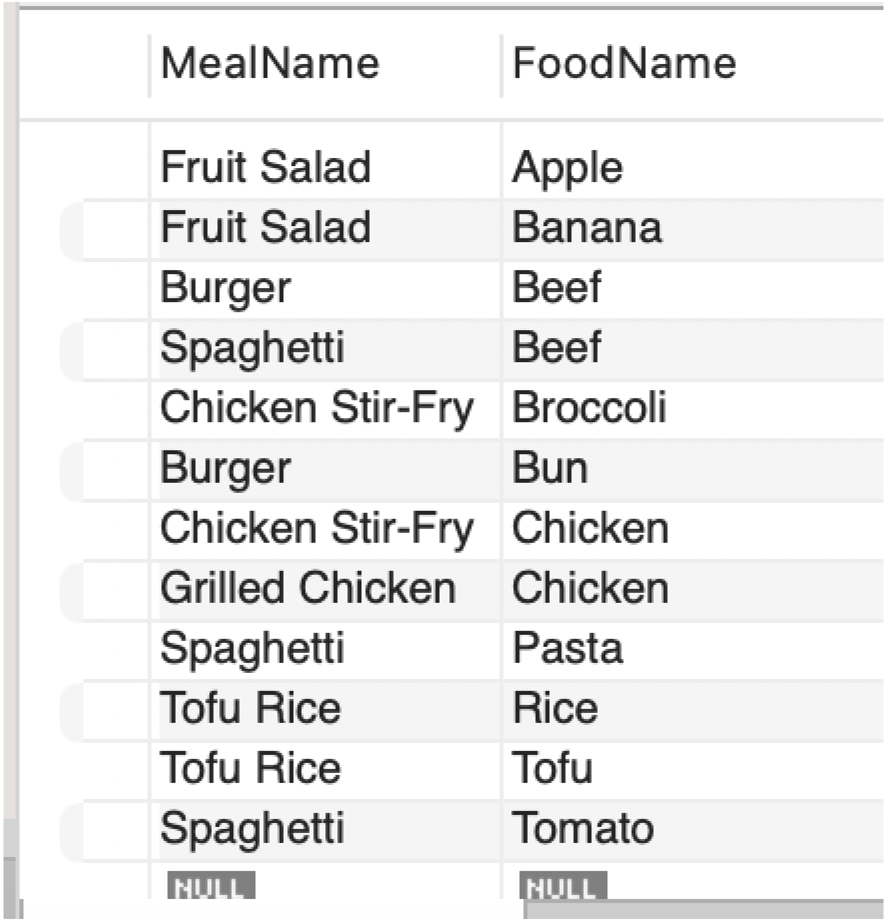
UID	MealName	Date	Amount
u1	Chicken Stir-Fry	2024-03-01	250
u1	Fruit Salad	2024-03-01	650
u2	Grilled Chicken	2024-03-01	750
u3	Fruit Salad	2024-03-02	400
u3	Tofu & Rice	2024-04-28	500
NULL	NULL	NULL	NULL

TABLE Meal



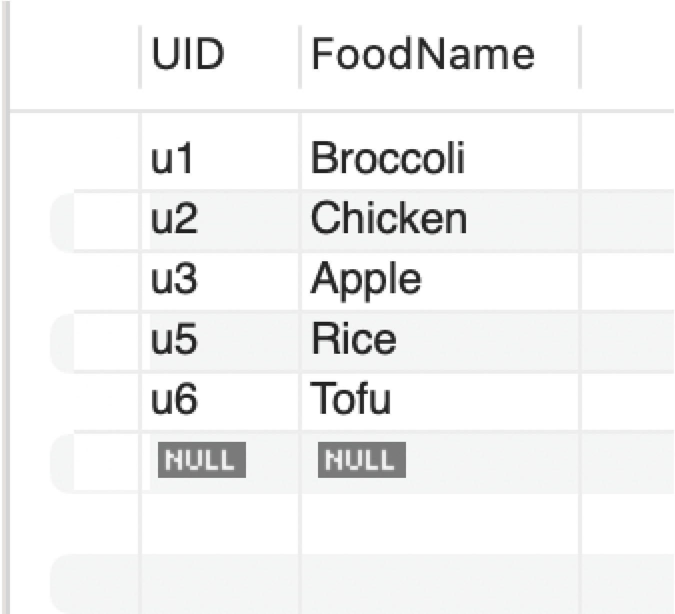
MealName
Burger
Chicken Stir-Fry
Fruit Salad
Grilled Chicken
Spaghetti
Tofu Rice
NULL

TABLE Meal_Has_Food



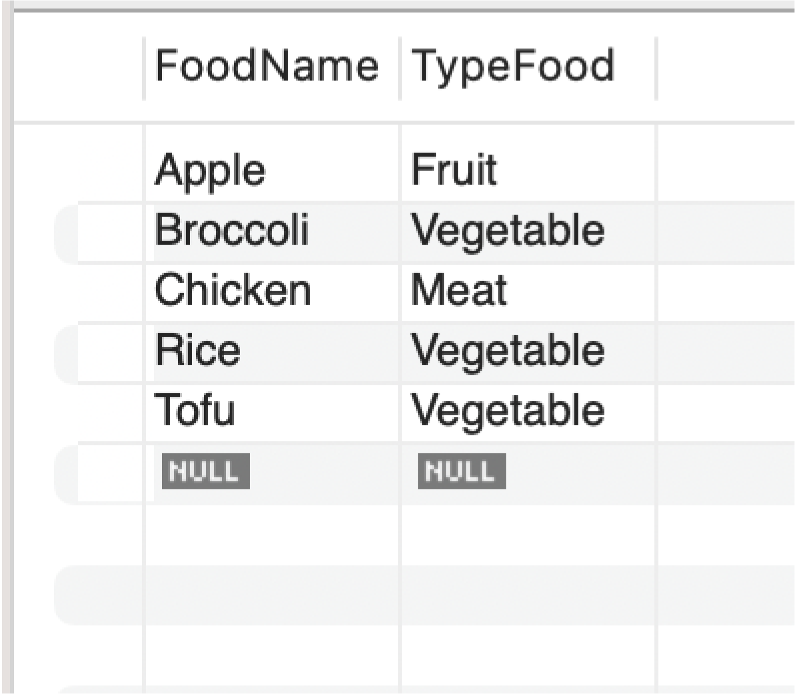
MealName	FoodName
Fruit Salad	Apple
Fruit Salad	Banana
Burger	Beef
Spaghetti	Beef
Chicken Stir-Fry	Broccoli
Burger	Bun
Chicken Stir-Fry	Chicken
Grilled Chicken	Chicken
Spaghetti	Pasta
Tofu Rice	Rice
Tofu Rice	Tofu
Spaghetti	Tomato
NULL	NULL

TABLE Favourite_1



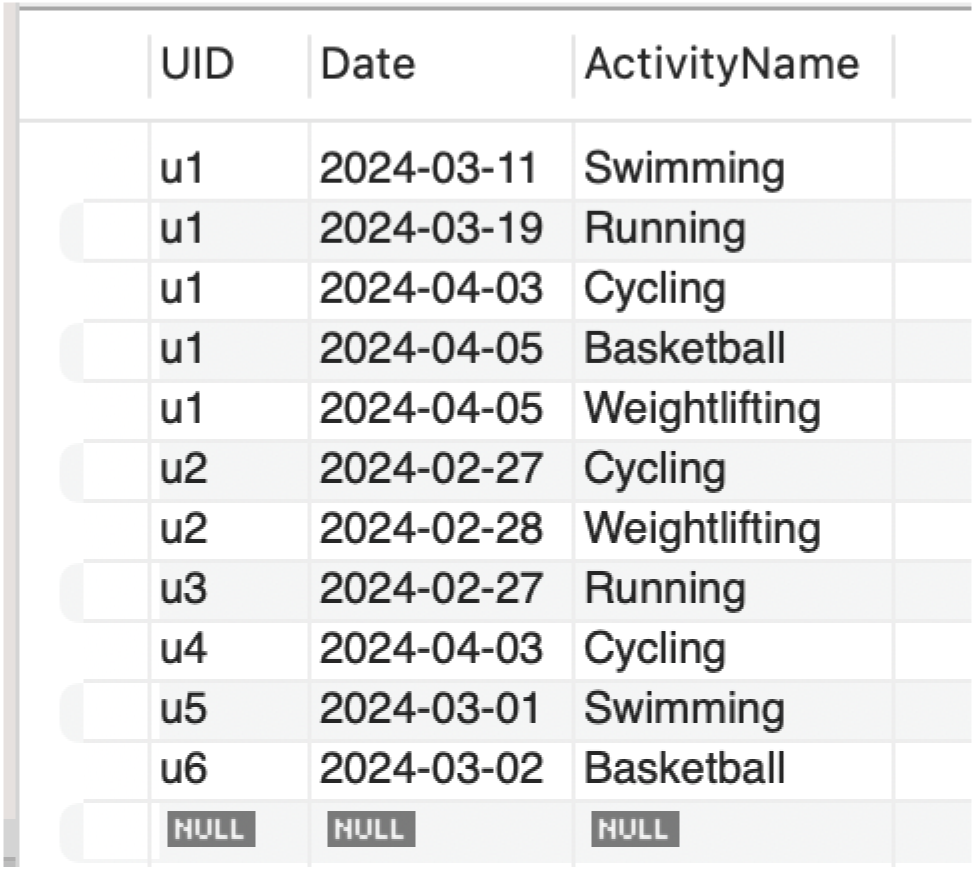
UID	FoodName
u1	Broccoli
u2	Chicken
u3	Apple
u5	Rice
u6	Tofu
NULL	NULL

TABLE Favourite_2



FoodName	TypeFood
Apple	Fruit
Broccoli	Vegetable
Chicken	Meat
Rice	Vegetable
Tofu	Vegetable
NULL	NULL

TABLE User_Do_Exercise

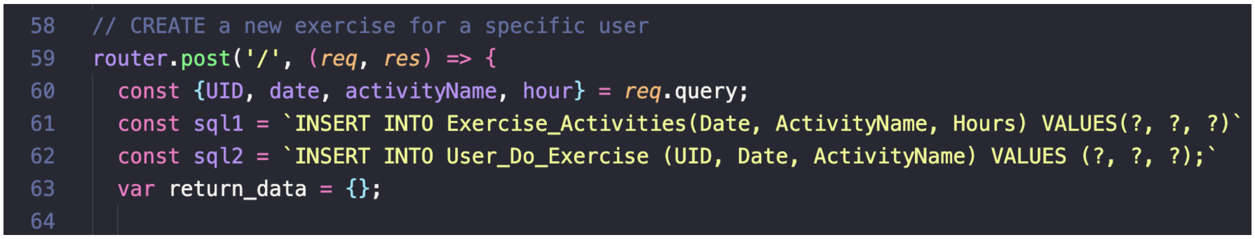


UID	Date	ActivityName
u1	2024-03-11	Swimming
u1	2024-03-19	Running
u1	2024-04-03	Cycling
u1	2024-04-05	Basketball
u1	2024-04-05	Weightlifting
u2	2024-02-27	Cycling
u2	2024-02-28	Weightlifting
u3	2024-02-27	Running
u4	2024-04-03	Cycling
u5	2024-03-01	Swimming
u6	2024-03-02	Basketball
NULL	NULL	NULL

d. Queries and Screenshots of operations:

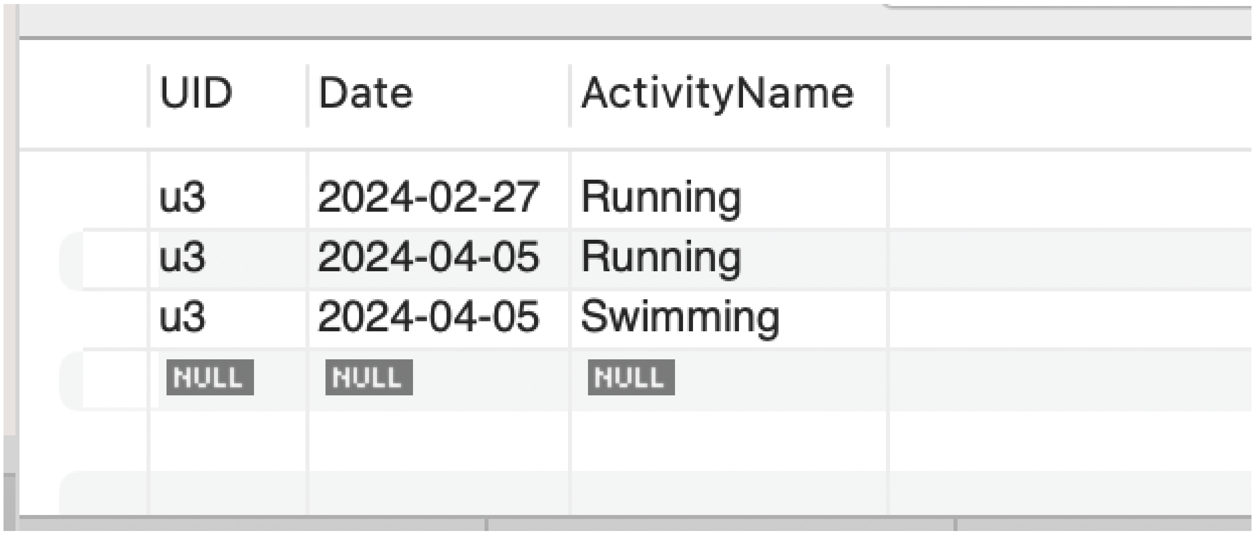
a) INSERT

Can be found at "/backend/routes/Exercises.js" line 61



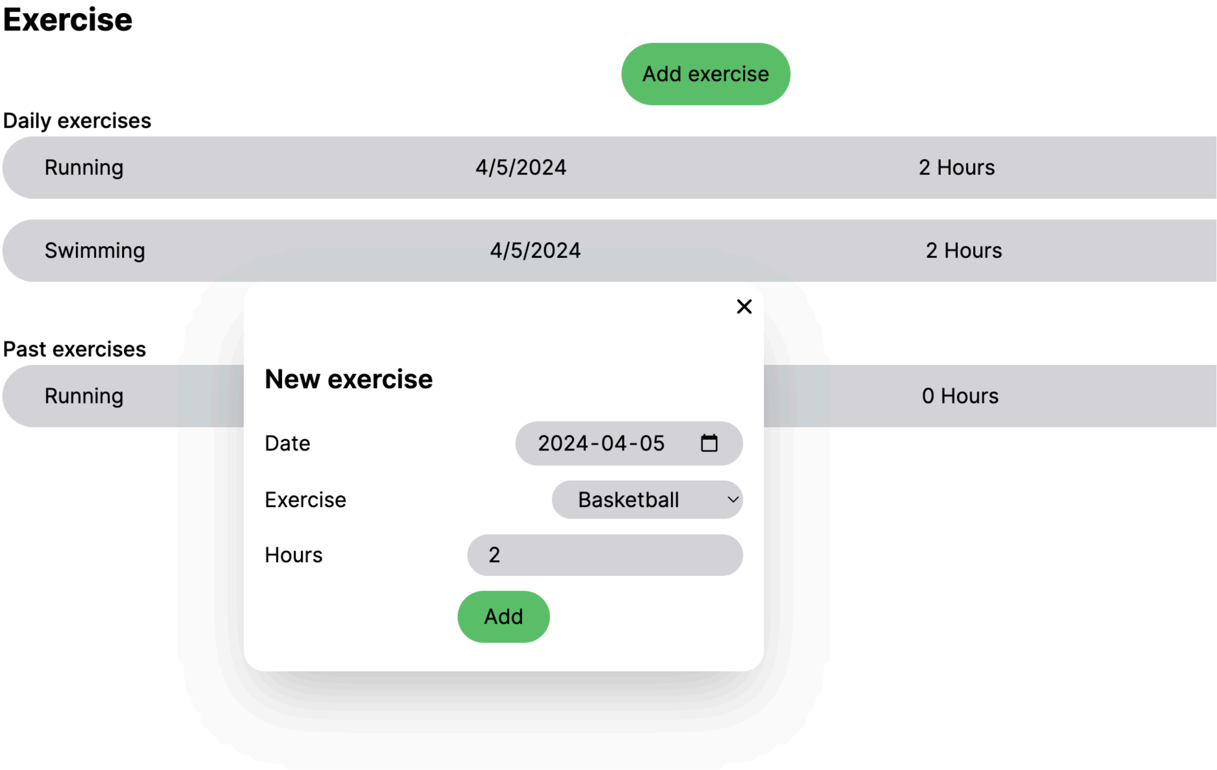
```
58  // CREATE a new exercise for a specific user
59  router.post('/', (req, res) => {
60    const {UID, date, activityName, hour} = req.query;
61    const sql1 = `INSERT INTO Exercise_Activities(Date, ActivityName, Hours) VALUES(?, ?, ?)`
62    const sql2 = `INSERT INTO User_Do_Exercise (UID, Date, ActivityName) VALUES (?, ?, ?);`
63    var return_data = {};
64
```

*Before



UID	Date	ActivityName
u3	2024-02-27	Running
u3	2024-04-05	Running
u3	2024-04-05	Swimming
NULL	NULL	NULL

*During



Exercise

Daily exercises

Running	4/5/2024	2 Hours
Swimming	4/5/2024	2 Hours

Past exercises

Running		0 Hours

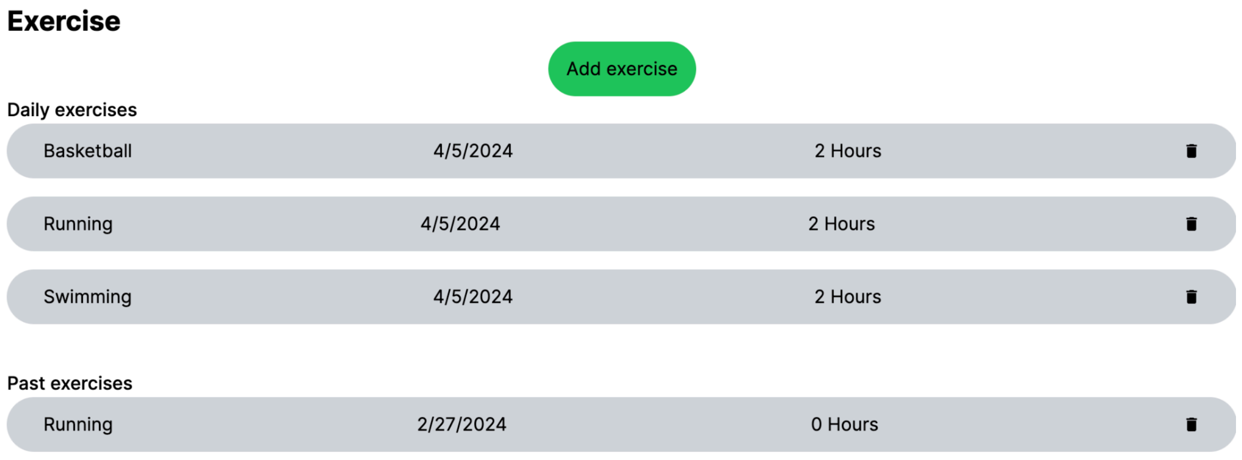
New exercise

Date 2024-04-05

Exercise Basketball

Hours 2

*After



Exercise

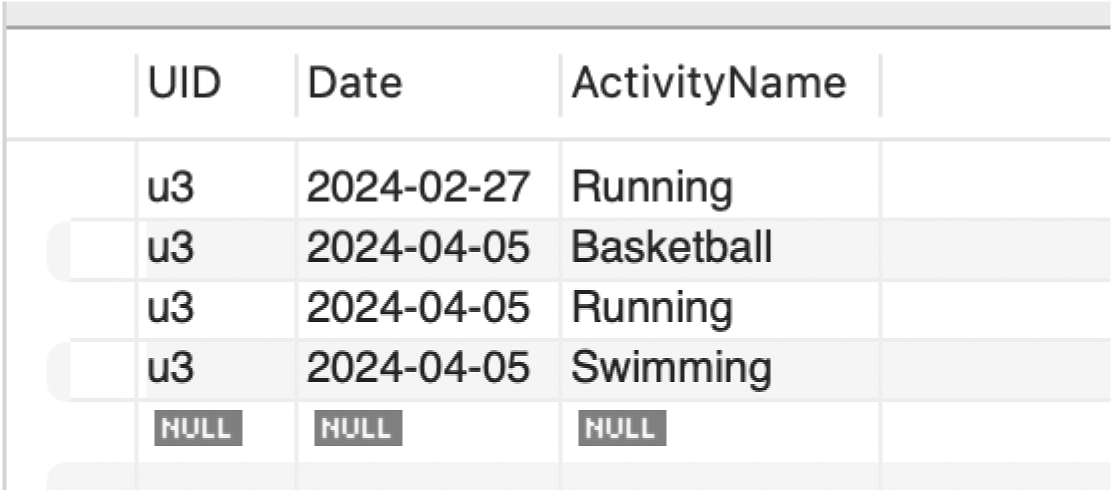
Add exercise

Daily exercises

Basketball	4/5/2024	2 Hours	🗑
Running	4/5/2024	2 Hours	🗑
Swimming	4/5/2024	2 Hours	🗑

Past exercises

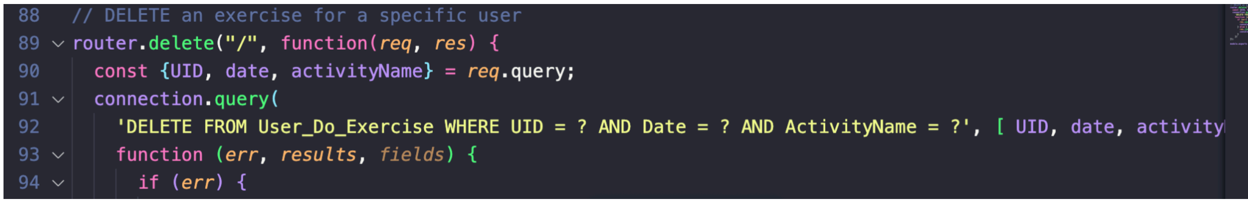
Running	2/27/2024	0 Hours	🗑



UID	Date	ActivityName
u3	2024-02-27	Running
u3	2024-04-05	Basketball
u3	2024-04-05	Running
u3	2024-04-05	Swimming
NULL	NULL	NULL

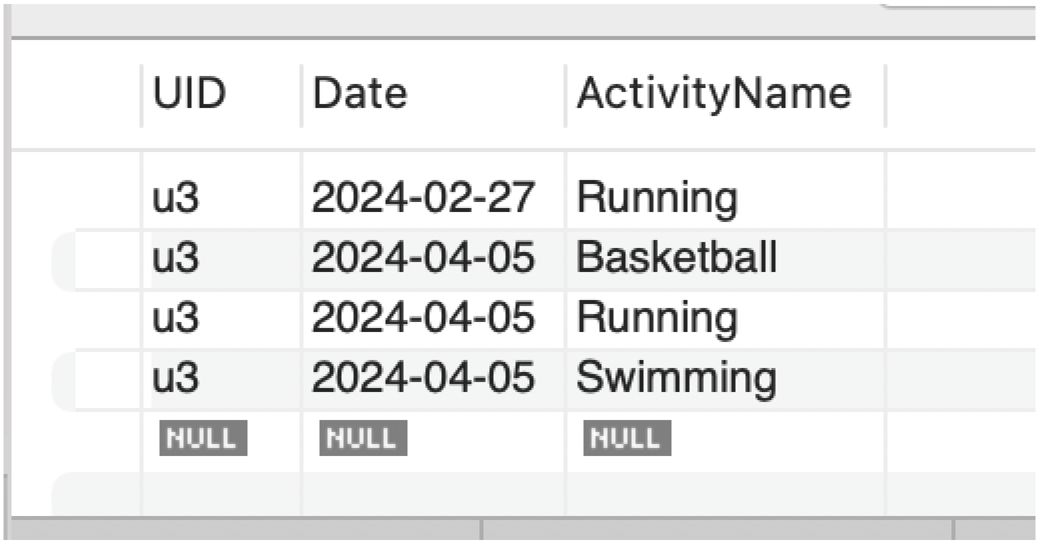
b) DELETE

Can be found at “/backend/routes/Exercises.js” line 92



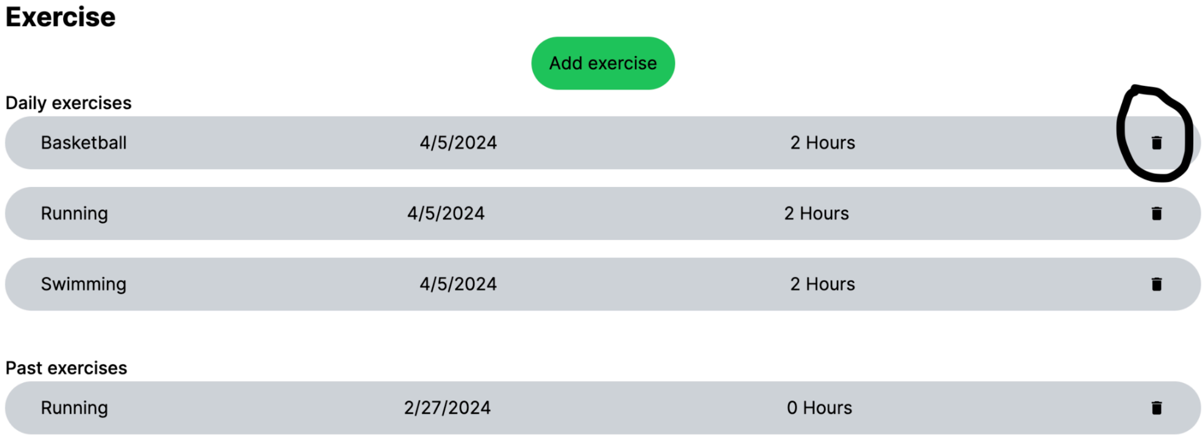
```
88   // DELETE an exercise for a specific user
89   router.delete("/", function(req, res) {
90     const {UID, date, activityName} = req.query;
91     connection.query(
92       'DELETE FROM User_Do_Exercise WHERE UID = ? AND Date = ? AND ActivityName = ?', [ UID, date, activity
93       function (err, results, fields) {
94         if (err) {
```

* Before



UID	Date	ActivityName
u3	2024-02-27	Running
u3	2024-04-05	Basketball
u3	2024-04-05	Running
u3	2024-04-05	Swimming
NULL	NULL	NULL

* During



Exercise

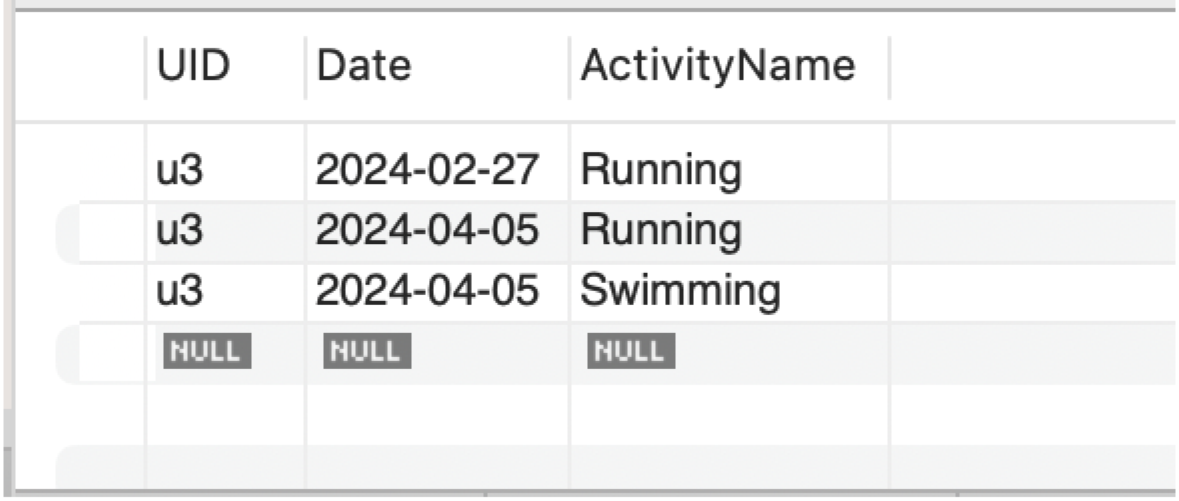
Daily exercises

Basketball	4/5/2024	2 Hours
Running	4/5/2024	2 Hours
Swimming	4/5/2024	2 Hours

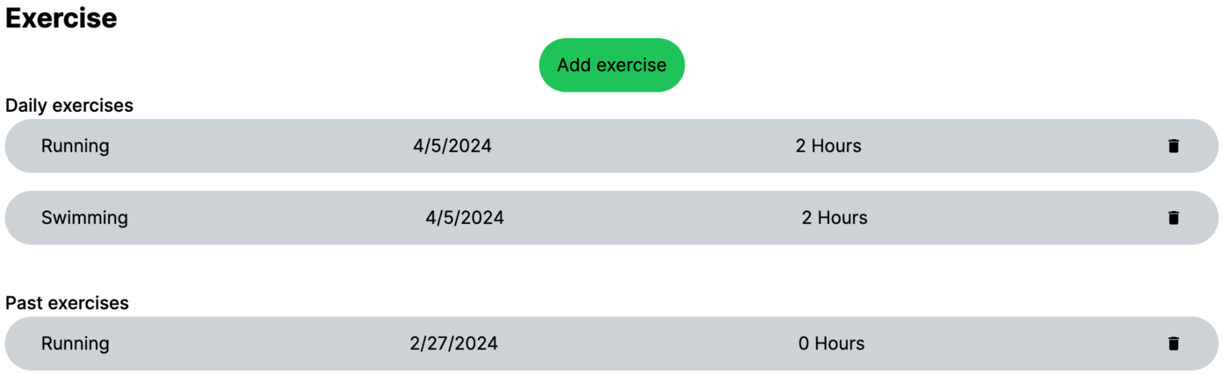
Past exercises

Running	2/27/2024	0 Hours

* After



UID	Date	ActivityName
u3	2024-02-27	Running
u3	2024-04-05	Running
u3	2024-04-05	Swimming
NULL	NULL	NULL



Exercise

Daily exercises

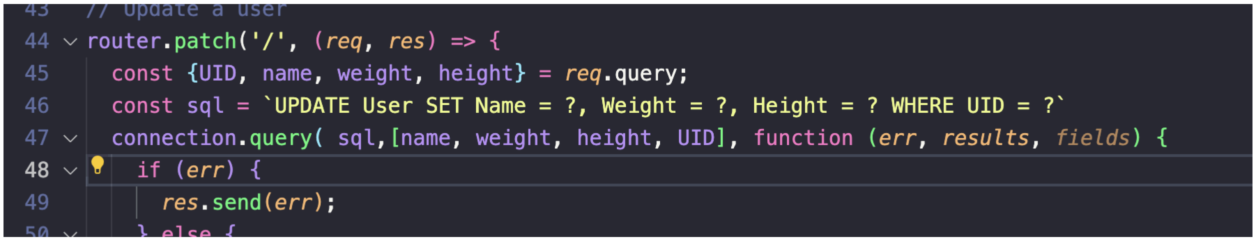
Running	4/5/2024	2 Hours	🗑
Swimming	4/5/2024	2 Hours	🗑

Past exercises

Running	2/27/2024	0 Hours	🗑

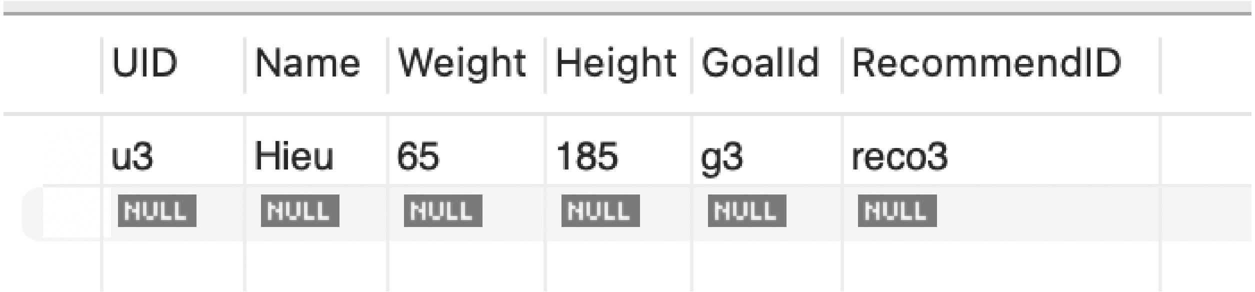
c) UPDATE

Can be found at “/backend/routes/User.js” line 46



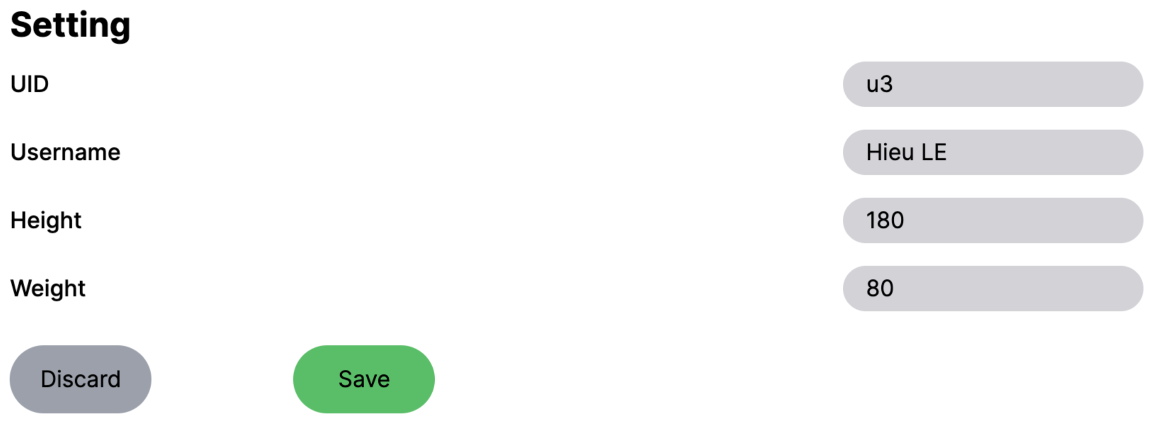
```
43  // Update a user
44  router.patch('/', (req, res) => {
45    const {UID, name, weight, height} = req.query;
46    const sql = `UPDATE User SET Name = ?, Weight = ?, Height = ? WHERE UID = ?`
47    connection.query( sql,[name, weight, height, UID], function (err, results, fields) {
48      if (err) {
49        res.send(err);
50      } else {
```

*Before



UID	Name	Weight	Height	GoalId	RecommendID
u3	Hieu	65	185	g3	reco3
NULL	NULL	NULL	NULL	NULL	NULL

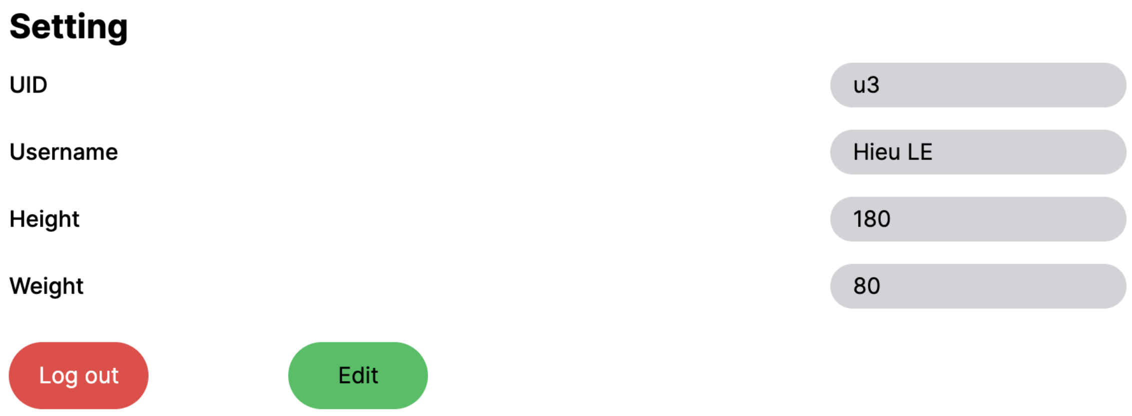
*During



Setting

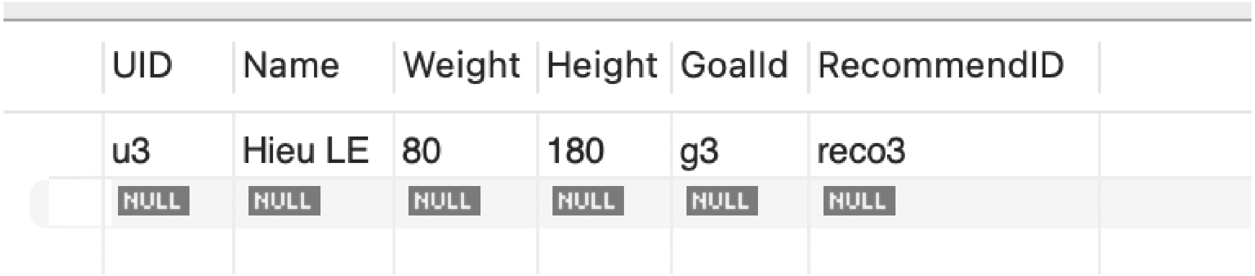
UID	u3
Username	Hieu LE
Height	180
Weight	80

*After



Setting

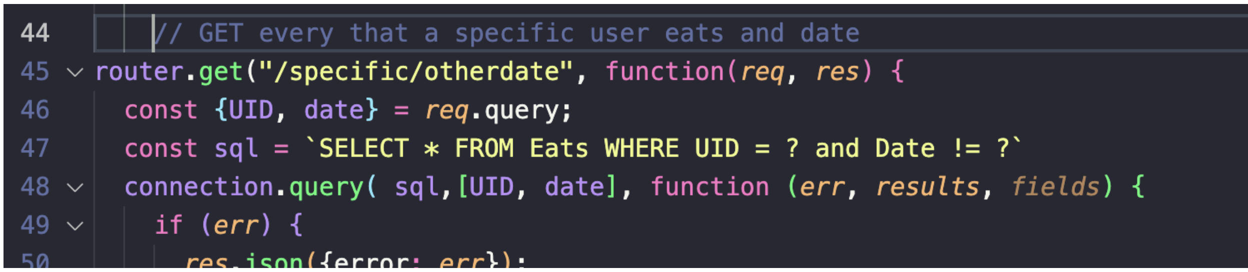
UID	u3
Username	Hieu LE
Height	180
Weight	80



UID	Name	Weight	Height	GoalId	RecommendID
u3	Hieu LE	80	180	g3	reco3
NULL	NULL	NULL	NULL	NULL	NULL

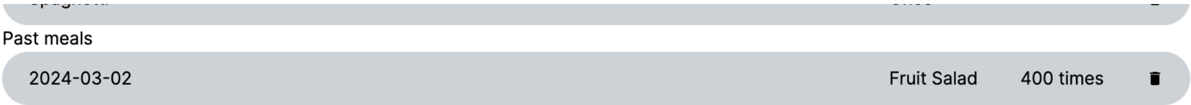
d) SELECTION

Can be found at “backend/routes/Eat.js” line 47



```
  // GET every that a specific user eats and date
router.get("/specific/otherdate", function(req, res) {
  const {UID, date} = req.query;
  const sql = `SELECT * FROM Eats WHERE UID = ? and Date != ?`
  connection.query( sql,[UID, date], function (err, results, fields) {
    if (err) {
      res.json({error: err});
```

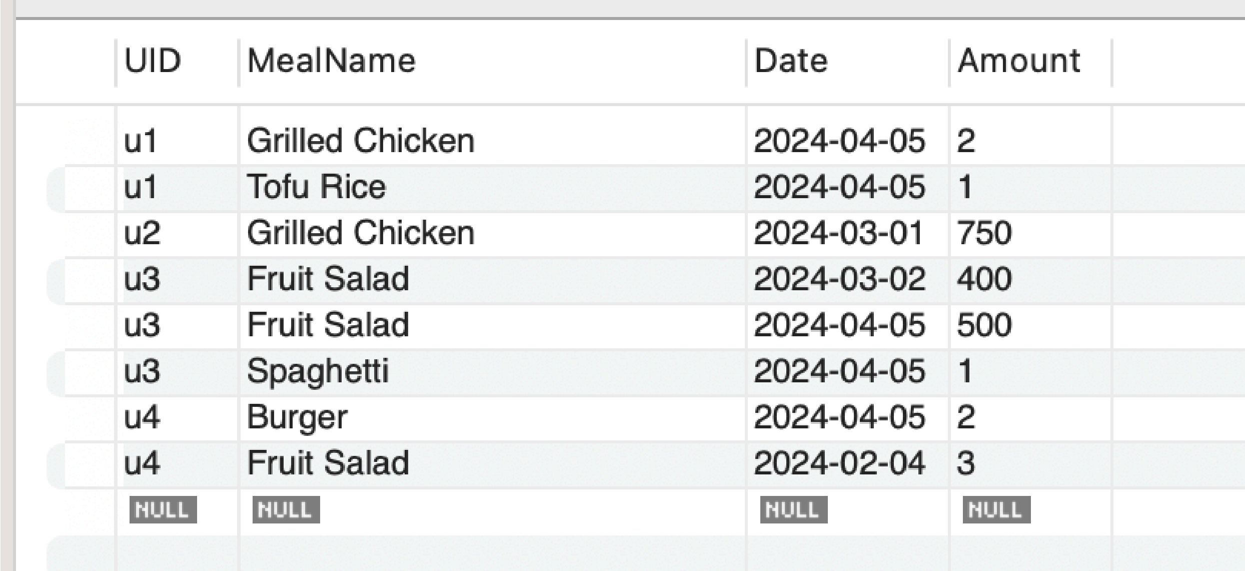
*Note: We don't have the option to let the user select what will be printed out, we automatically display everything the user Eats



Past meals

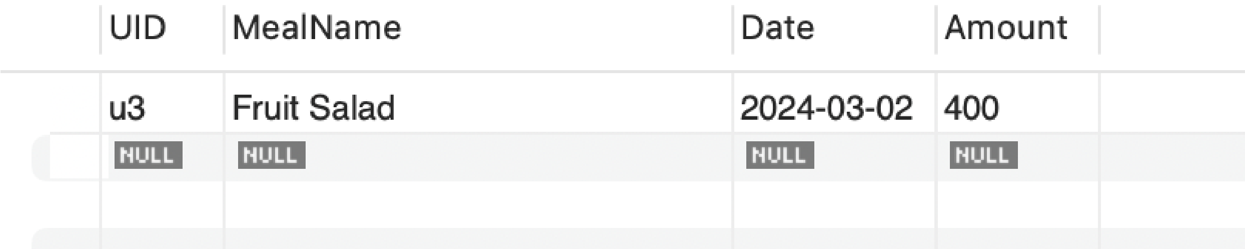
```
2024-03-02                                   Fruit Salad    400 times
```

*Before



UID	MealName	Date	Amount
u1	Grilled Chicken	2024-04-05	2
u1	Tofu Rice	2024-04-05	1
u2	Grilled Chicken	2024-03-01	750
u3	Fruit Salad	2024-03-02	400
u3	Fruit Salad	2024-04-05	500
u3	Spaghetti	2024-04-05	1
u4	Burger	2024-04-05	2
u4	Fruit Salad	2024-02-04	3
NULL	NULL	NULL	NULL

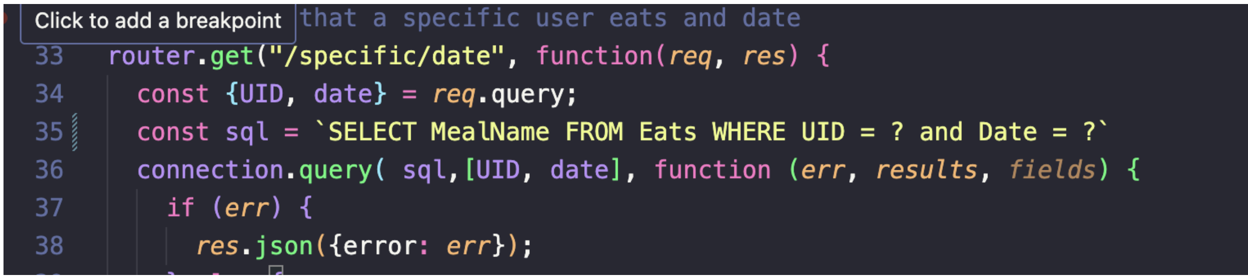
*After



UID	MealName	Date	Amount
u3	Fruit Salad	2024-03-02	400
NULL	NULL	NULL	NULL

e) PROJECTION

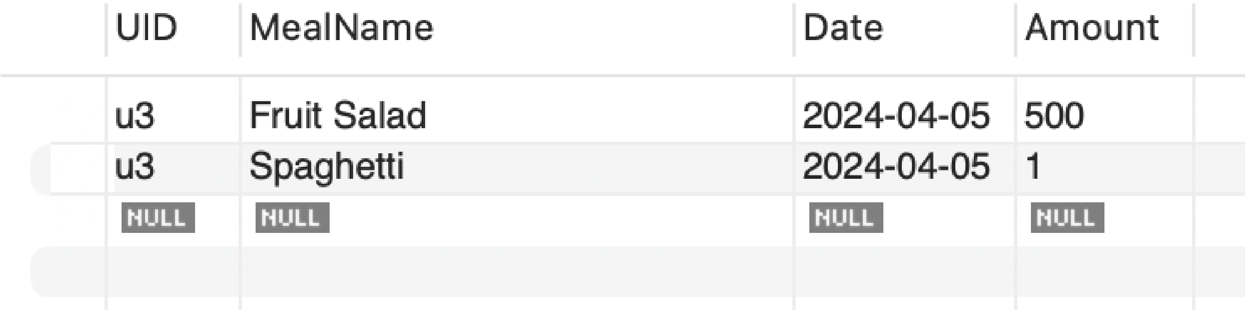
Can be found at “/backend/routes/Eat.js” line 35



```
                          that a specific user eats and date
33  router.get("/specific/date", function(req, res) {
34    const {UID, date} = req.query;
35    const sql = `SELECT MealName FROM Eats WHERE UID = ? and Date = ?`
36    connection.query( sql,[UID, date], function (err, results, fields) {
37      if (err) {
38        res.json({error: err});
```

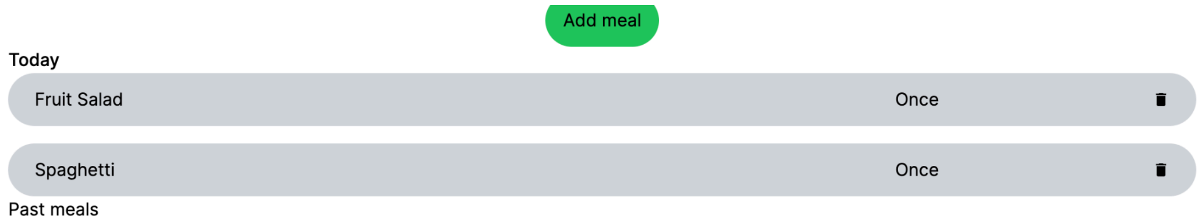
*Note: We don’t have the option to let the user select what will be printed out, we automatically display everything the user Eats

*Before



UID	MealName	Date	Amount
u3	Fruit Salad	2024-04-05	500
u3	Spaghetti	2024-04-05	1
NULL	NULL	NULL	NULL

*During

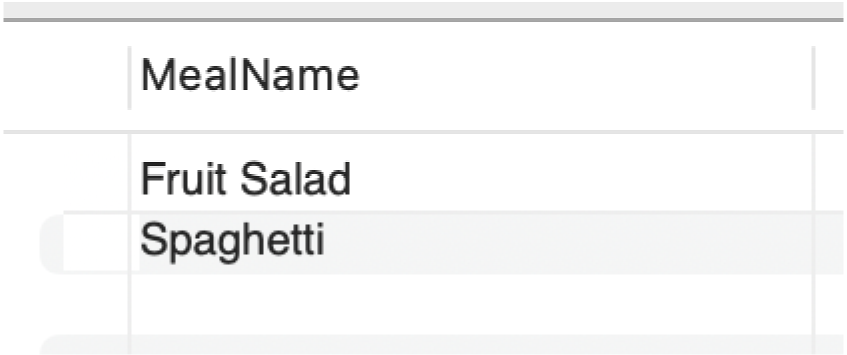


Today

Fruit Salad	Once
Spaghetti	Once

Past meals

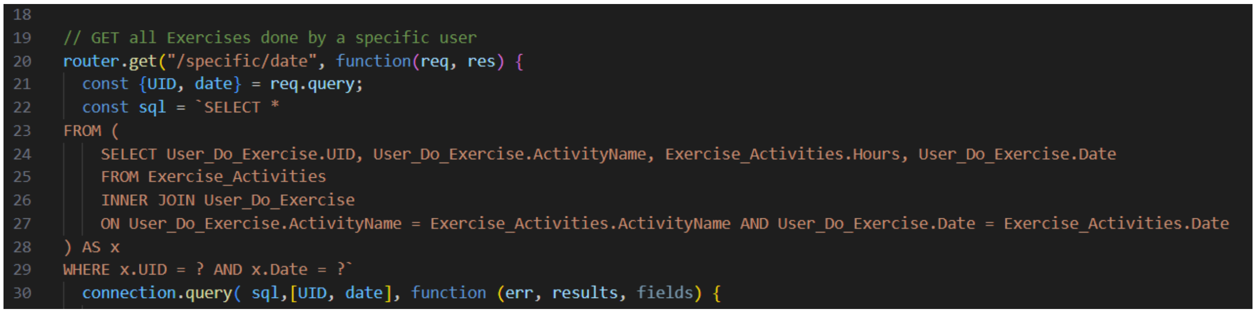
*After



MealName
Fruit Salad
Spaghetti

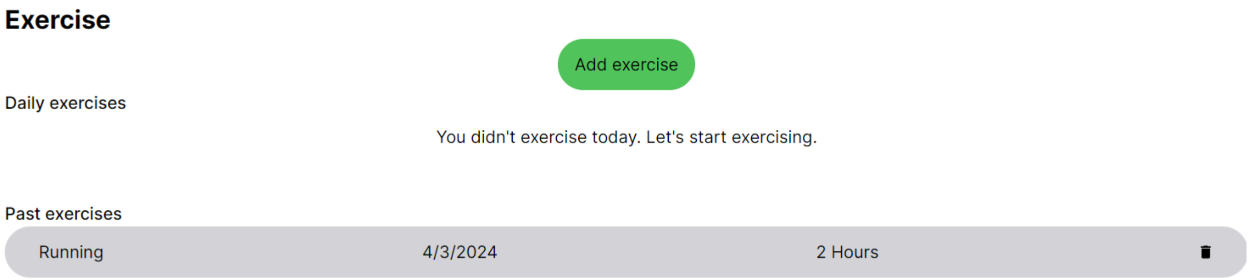
f) JOIN

*Can be found at “/backend/routes/Exercise.js” line 22



```
18
19  // GET all Exercises done by a specific user
20  router.get("/specific/date", function(req, res) {
21    const {UID, date} = req.query;
22    const sql = `SELECT *
23  FROM (
24      SELECT User_Do_Exercise.UID, User_Do_Exercise.ActivityName, Exercise_Activities.Hours, User_Do_Exercise.Date
25      FROM Exercise_Activities
26      INNER JOIN User_Do_Exercise
27      ON User_Do_Exercise.ActivityName = Exercise_Activities.ActivityName AND User_Do_Exercise.Date = Exercise_Activities.Date
28  ) AS x
29  WHERE x.UID = ? AND x.Date = ?`
30    connection.query( sql,[UID, date], function (err, results, fields) {
```

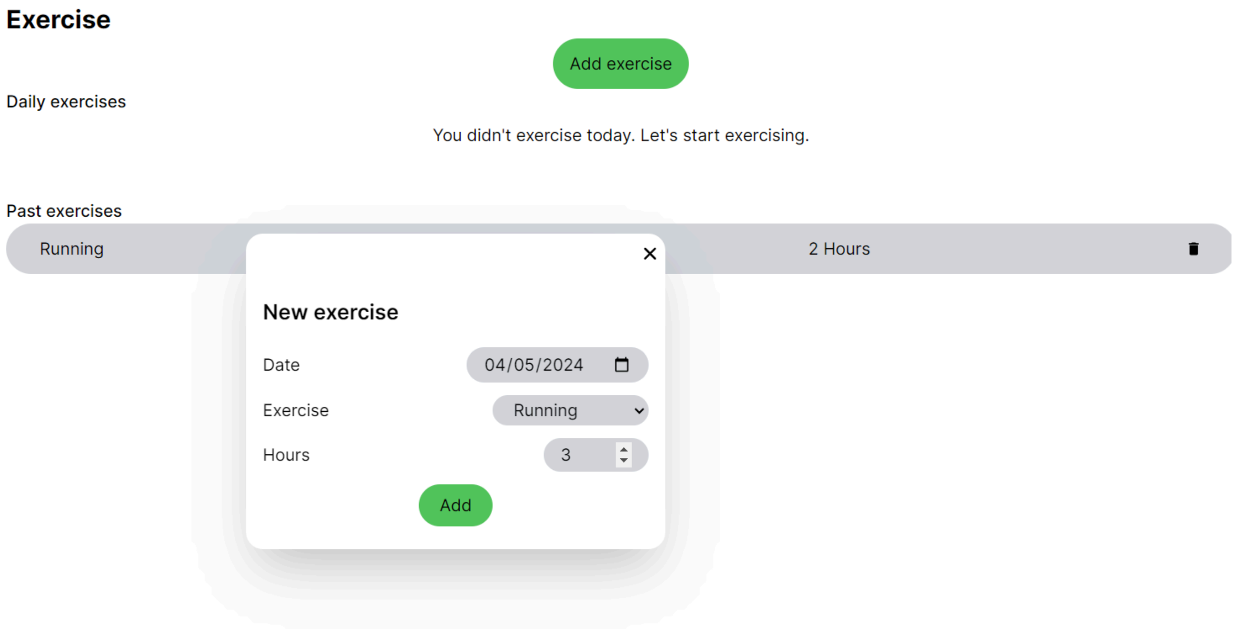
*Note: Again, we don’t have the option for a user to choose what to show, all the Exercises done by the user on a particular day will be displayed automatically.



You didn't exercise today. Let's start exercising.

Running	4/3/2024	2 Hours

*Before



You didn't exercise today. Let's start exercising.

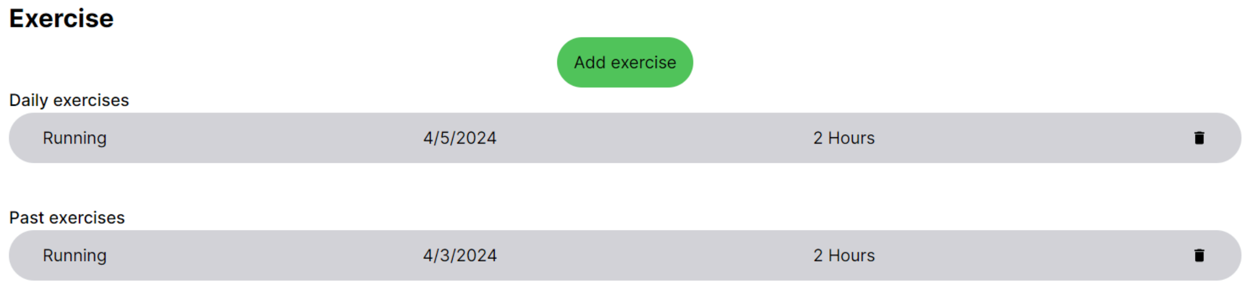
Running	2 Hours

Date 04/05/2024

Exercise Running

Hours 3

*During



Exercise

Daily exercises

Running	4/5/2024	2 Hours

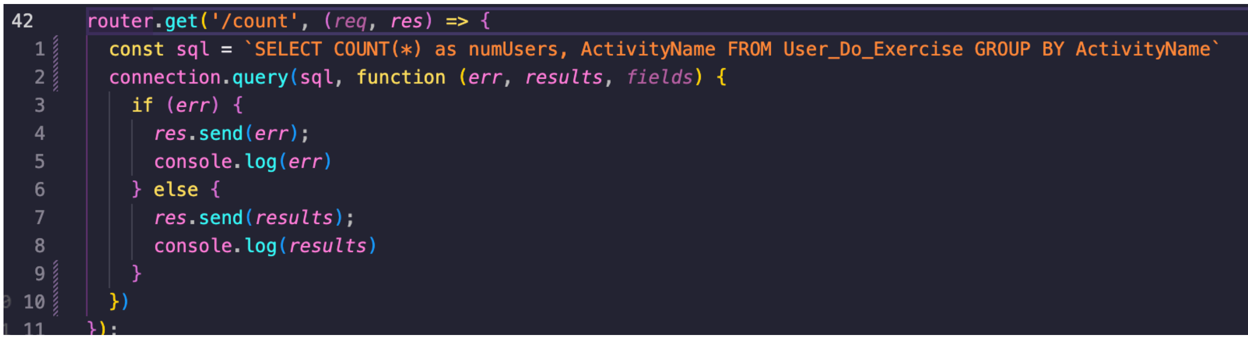
Past exercises

Running	4/3/2024	2 Hours

*After

g) AGGREGATION WITH GROUP BY

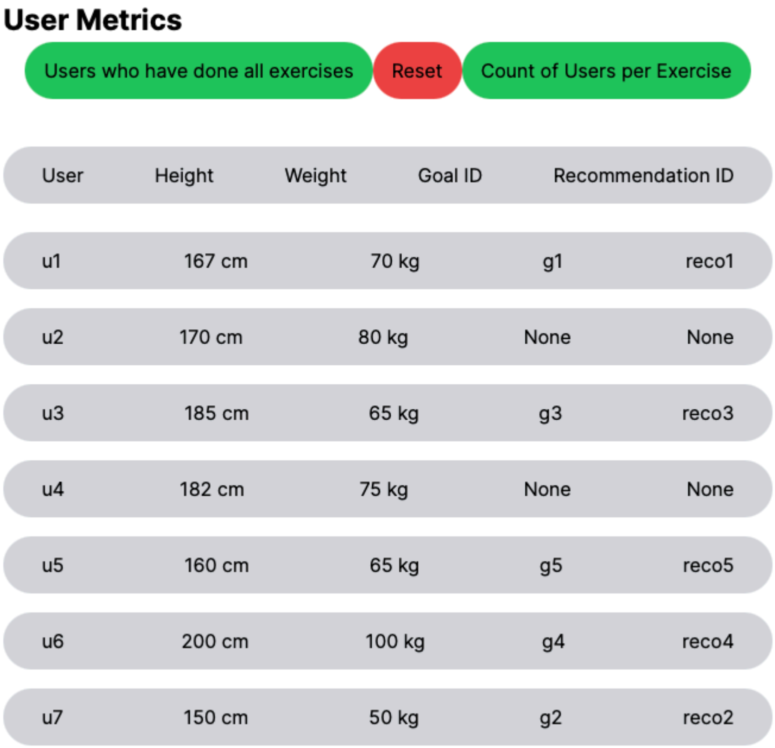
*Can be found at “/backend/routes/Util.js” line 41



```
router.get('/count', (req, res) => {
  const sql = `SELECT COUNT(*) as numUsers, ActivityName FROM User_Do_Exercise GROUP BY ActivityName`
  connection.query(sql, function (err, results, fields) {
    if (err) {
      res.send(err);
      console.log(err)
    } else {
      res.send(results);
      console.log(results)
    }
  })
});
```

*Note: Again, we don’t have the option for user to choose what to display, Record for daily nutrient intake will be automatically displayed’

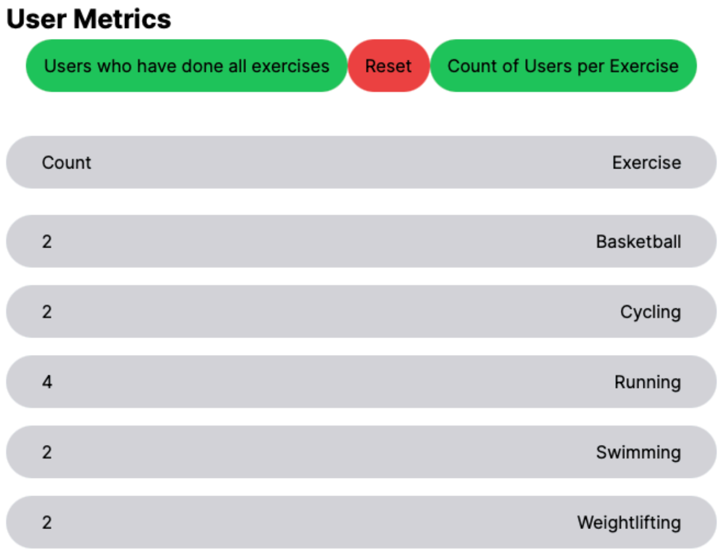
*Before:


Users who have done all exercises
Reset
Count of Users per Exercise

User Metrics

User	Height	Weight	Goal ID	Recommendation ID
u1	167 cm	70 kg	g1	reco1
u2	170 cm	80 kg	None	None
u3	185 cm	65 kg	g3	reco3
u4	182 cm	75 kg	None	None
u5	160 cm	65 kg	g5	reco5
u6	200 cm	100 kg	g4	reco4
u7	150 cm	50 kg	g2	reco2

*After:


Users who have done all exercises
Reset
Count of Users per Exercise

User Metrics

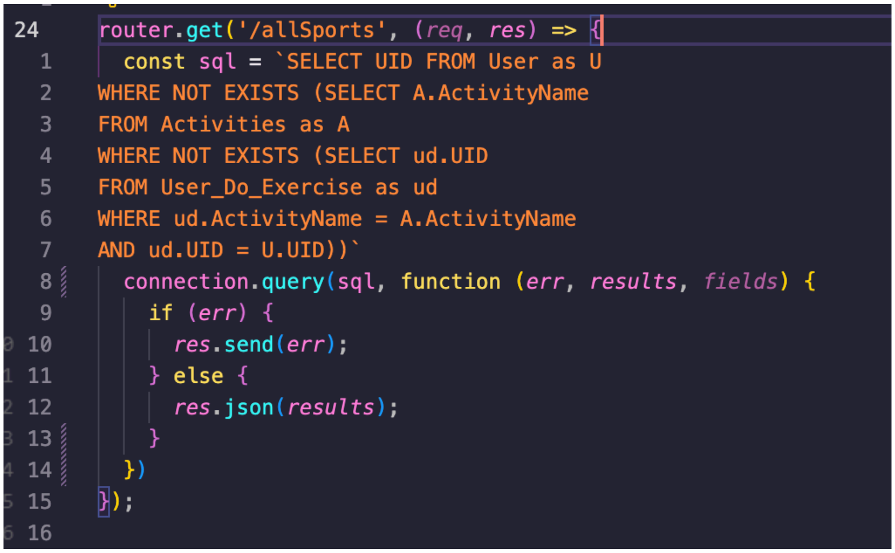
Count	Exercise
2	Basketball
2	Cycling
4	Running
2	Swimming
2	Weightlifting

h) AGGREGATION WITH HAVING

i) NESTED AGGREGATION WITH GROUP BY

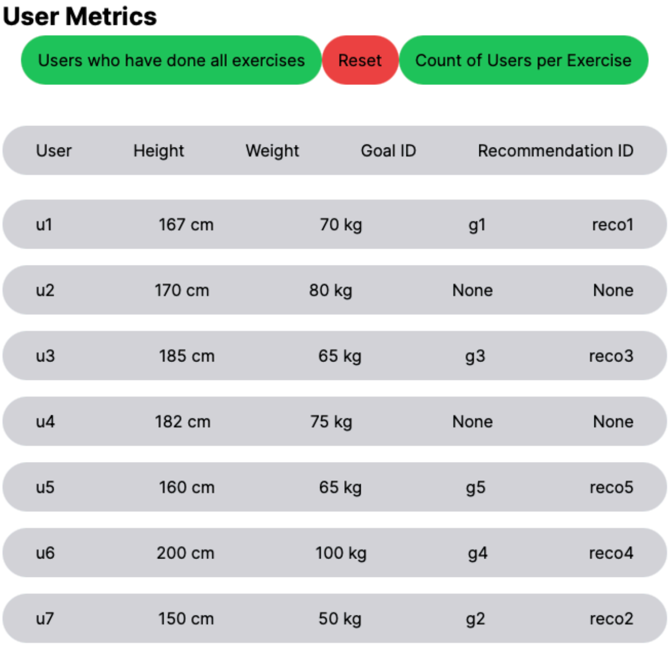
j) DIVISION

*Can be found at “/backend/routes/Util.js” line 41



```
router.get('/allSports', (req, res) => {
  const sql = `SELECT UID FROM User as U
WHERE NOT EXISTS (SELECT A.ActivityName
FROM Activities as A
WHERE NOT EXISTS (SELECT ud.UID
FROM User_Do_Exercise as ud
WHERE ud.ActivityName = A.ActivityName
AND ud.UID = U.UID))`
  connection.query(sql, function (err, results, fields) {
    if (err) {
      res.send(err);
    } else {
      res.json(results);
    }
  })
});
```

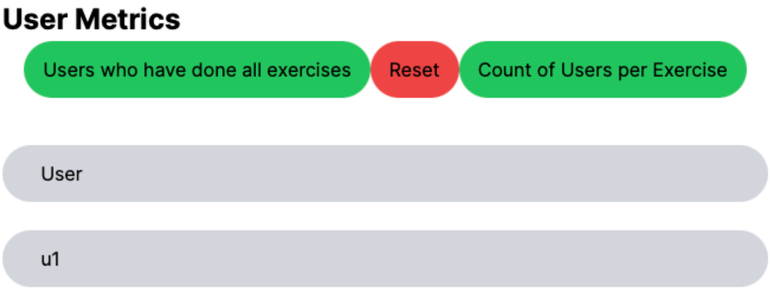
*Before:


Users who have done all exercises
Reset
Count of Users per Exercise

User Metrics

User	Height	Weight	Goal ID	Recommendation ID
u1	167 cm	70 kg	g1	reco1
u2	170 cm	80 kg	None	None
u3	185 cm	65 kg	g3	reco3
u4	182 cm	75 kg	None	None
u5	160 cm	65 kg	g5	reco5
u6	200 cm	100 kg	g4	reco4
u7	150 cm	50 kg	g2	reco2

*After:



User Metrics

Users who have done all exercises
Reset
Count of Users per Exercise

User

u1