CPSC 304 Project Cover Page

Milestone #: 4

Date: November 25, 2023

Group Number: 27

Name	Student Number	CS Alias (Userid)	Preferred Email Address
Clare Pan	95483459	f2l5o	clarepan0@gmail.com
Dizhe Xiang	565742	b5h9t	dizhexiang@gmail.com
Vincent Lee	84258847	i6z1i	vinlee1208@gmail.com

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 email 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

1.Project Description

a) This application is a user-focused fitness application that allows users to track their workouts, nutrition, and body measurements. The domain of this application is fitness/body management.
b) This project models the tracking of users' body measurements, daily workout and their daily nutrition.
c) The user can login, and see their own data, and there's an admin that can check users information, especially the class registration

2.How the final schema differs:

- We add NID as the primary key to Nutrition so that multiple workouts don't connect to the same nutrition data for each date. And HasNutrition, Workout and Meal tables' foreign key also changed from NDate to NID.
- We add intensity to GroupClass to fit the UPDATE rubric attributes number requirement

3.How the database looks (schema and screenshot):

User(<u>UserID</u>, userName, Gender, Age)

	USERID	USERNAME	GENDER	AGE
1	001	userA	Female	42
2	002	userB	Male	41
3	003	userC	Female	18
4	004	userD	Male	19
5	005	userE	Female	30
6	006	admin	Male	99

Trainer(<u>TID</u>, TrainerName)

	TID	TRAINERNAME
1	1	trainerA
2	2	trainerB
3	3	trainerC
4	4	trainerD
5	5	trainerE
6	6	trainerF
7	7	trainerG
8	8	trainerH
9	9	trainerI
10	10	trainerJ

Food(<u>FoodName</u>, FoodCalories)

	FOODNAME	FOODCALORIES
1	Medium Green Apple	90
2	Banana	90
3	Chicken Fried Thigh	200
4	Pizza	200
5	Black Coffee	5

Exercises(ExerciseName, Category,CaloriesBurned, Intensity)

	EXERCISENAME	CATEGORY	CALORIESBURNED	INTENSITY
1	Jogging	Aerobics	200	Low
2	SpinBiking	Cardio	250	Medium
3	Tango	Aerobics	350	High
4	Volleyball	Aerobics	250	Low
5	Swimming	Aerobics	300	Medium
6	Burpees	Cardio	150	High

Nutrition(NID, NDate, DailyConsumedCalories, DailyCaloriesGoa)

	NID	NDATE	DAILYCONSUMEDCALORIES	DAILYCALORIESGOAL
1	1	2022-01-01	1800	2000
2	2	2022-01-02	1556	1800
3	3	2022-01-03	1900	1600
4	4	2022-01-04	2500	2300
5	5	2022-01-05	2600	2800
6	6	2024-01-05	2600	2800
7	7	2024-03-10	1800	1900
8	8	2023-03-10	1800	1900

Workout(WID, **NID** NOT NULL, TotalCaloriesBurned, WorkoutDate, Total Duration)

	WID	NID	TOTALCALORIESBURNED	WORKOUTDATE	TOTALDURATION
1	001	1	200	2022-01-01	30
2	002	2	650	2022-01-02	120
3	003	3	445	2022-01-03	36
4	004	4	500	2022-01-04	65
5	005	5	110	2022-01-05	15
6	006	6	200	2024-01-05	15
7	007	7	400	2024-03-10	40
8	008	8	400	2023-03-10	40

Meal(MID, **NID** NOT NULL, Type, MealCaloriesConsumed)

	MID	NID	TYPE	MEALCALORIESCONSUMED
1	1	1	Breakfast	700
2	2	2	Snack	200
3	3	3	Lunch	900
4	4	4	Snack	200
5	5	5	Dinner	600

Class_Trainer(ClassTitle,**TID** UNIQUE NOT NULL)

	CLASSTITLE	TID
1	Cardio Class	1
2	Spin Class	2
3	DanceFit Class	3
4	Sports Class	4
5	AquaFit Class	5
6	HIIT Class	6
7	Zumba Class	7

Class_Price(ClassTitle,ClassPrice)

	CLASSTITLE	CLASSPRICE
1	Cardio Class	100
2	Spin Class	80
3	DanceFit Class	90
4	Sports Class	110
5	AquaFit Class	120
6	HIIT Class	95
7	Zumba Class	150

GroupClass(**WID**,**ClassTitle**, Intensity)

	WID	CLASSTITLE	INTENSITY
1	001	Cardio Class	Low
2	002	Spin Class	Median
3	003	DanceFit Class	Low
4	004	Sports Class	Median
5	005	AquaFit Class	High
6	006	HIIT Class	High
7	007	Zumba Class	Median
8	008	Cardio Class	Low

WorkoutIncludeExercise(**WID**, **ExerciseName**, Duration)

	WID	EXERCISENAME	DURATION
1	001	Jogging	30
2	002	SpinBiking	45
3	003	Tango	50
4	004	Volleyball	80
5	005	Swimming	60
6	006	Burpees	15
7	007	Jogging	15

MealContainFood(**MID**, **FoodName**)

	MID	FOODNAME	QUANTITY
1	1	Medium Green Apple	0
2	2	Banana	0
3	3	Chicken Fried Thigh	0
4	4	Pizza	0
5	5	Black Coffee	0

DoesWorkout(**UID**,**WID**)

	USERID	WID
1	001	001
2	001	002
3	002	002
4	002	003
5	003	004
6	003	005

HasNutrition(**UID**,**NID**)

	USERID	NID
1	001	1
2	001	2
3	002	3
4	003	4
5	003	5

HasMeal(**UID**,**MID**)

	USERID	MID
1	001	1
2	001	2
3	002	3
4	003	4
5	003	5

TrackMeasurement(**UID**, MDate, Weight, Height)

	USERID	MDATE	WEIGHT	HEIGHT
1	001	2022-01-01	52	167.6
2	001	2023-10-06	58.5	175
3	001	2023-10-12	57.6	175
4	001	2023-10-18	57	175
5	001	2023-11-08	54	175
6	001	2023-11-09	55	175
7	001	2023-11-14	54.2	175
8	001	2023-11-20	53.3	175
9	001	2023-11-21	52.7	175
10	002	2022-01-02	73.4	178.2
11	003	2022-01-03	60.5	172.3
12	004	2022-01-04	86	180.9
13	005	2022-01-05	100	200

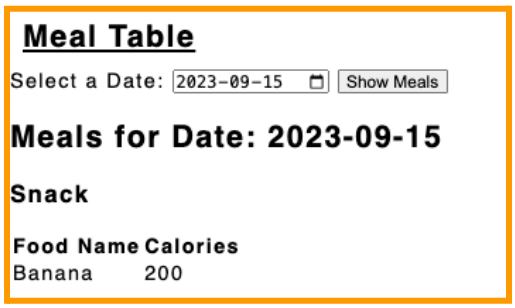
4. SQL Script

INSERT Operation:

Situation 1 - Affect more than one table:
Insert Meal table, would affect HasMeal, Nutrition tables (also able to handle the foreign key DNE)
Nutrition.php

```
Line 382:    INSERT INTO Meal (MID, NID, Type, MealCaloriesConsumed)
             VALUES (:randMID, :resultNID, :new_meal, :MealCaloriesConsumed);
Line 384:    INSERT INTO HasMeal (UserID, MID)
             VALUES (:userID, :randMID);
```

UserA is logged into their profile and is on the nutrition page
Before adding a new meal on date 2023-09-15: Select 2023-09-15 in the 'Meal Table' section and click 'Show Meals'. Meal table will show the all previously added meals on 2023-09-15



During adding a new meal on date 2023-09-15: select date, meal, food, and quantity, then click 'Add Food'

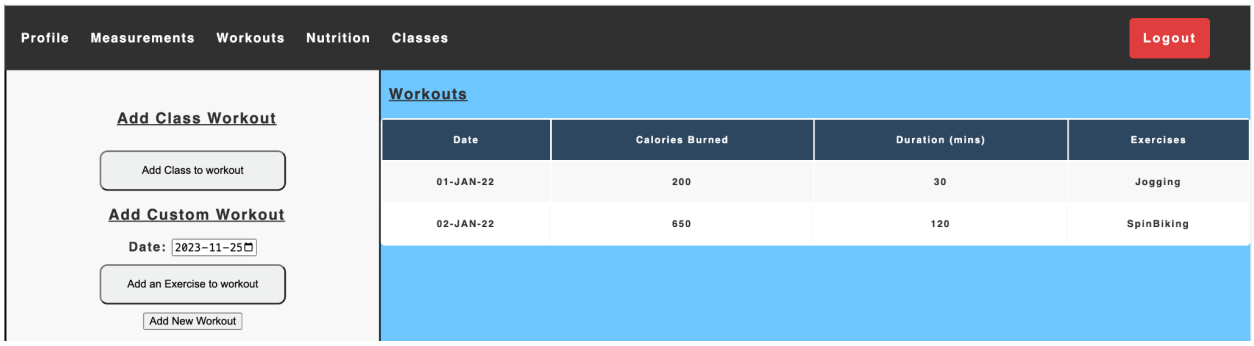
After adding a meal on date 2023-09-15: Select 2023-09-15 in the 'Meal Table' section and click 'Show Meals'. Meal table will show the meal that we just inserted into the meal table

Situation 2 - Foreign Key doesn't exist:
Insert Workout table, this would also insert into the Nutrition Table. We handle the situation that the Nutrition haven't had the NID (foreign key in Workout)
Workouts.php:
Line 132: INSERT INTO DoesWorkout (UserID, WID) VALUES (:id, :cID);

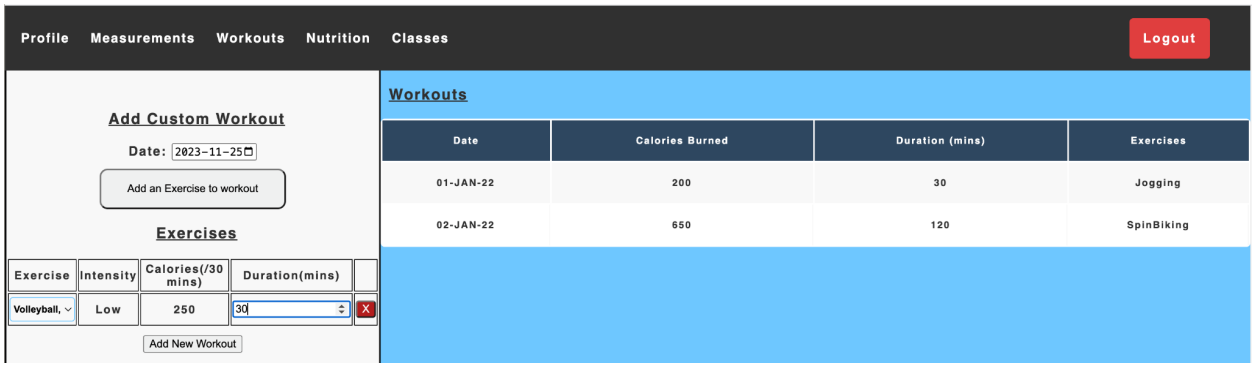
Line 176: INSERT INTO Nutrition (NID, NDate, DailyConsumedCalories, DailyCaloriesGoal)
VALUES (:nid, TO_DATE(:ndate, 'YYYY-MM-DD'), 0, 0);
Line 186: INSERT INTO HasNutrition (UserID, NID) VALUES (:id, :nid);
Line 219: INSERT INTO WorkoutIncludeExercise (WID, ExerciseName, Duration)
VALUES(:wid, :exerciseName, :exerciseDuration);
Line 229: INSERT INTO WORKOUT (WID, NID, TotalCaloriesBurned, WorkoutDate, TotalDuration)
VALUES (:wid, :nid, :calories, TO_DATE(:wdate, 'YYYY-MM-DD'), :duration);

login as UserA, select Workouts page

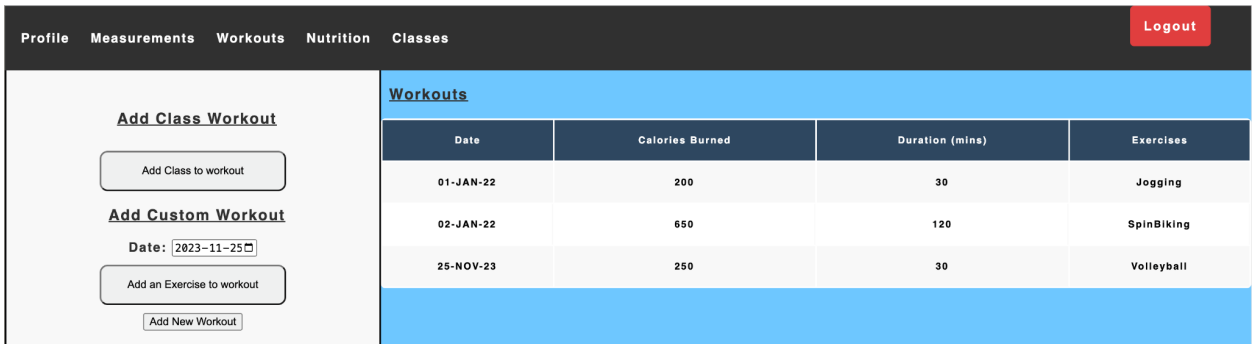
Before adding exercise to workout: click “Add an Exercise to workout” button



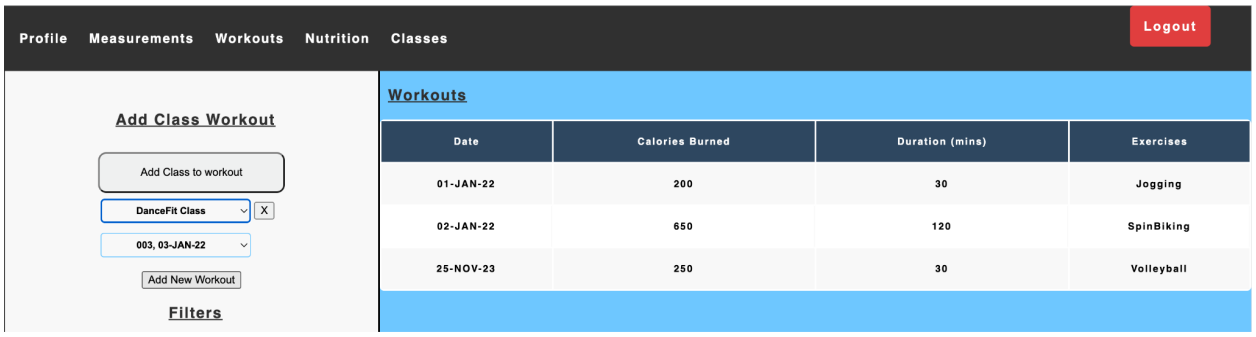
During adding exercise to workout: select date and exercise, click “Add New Workout” button



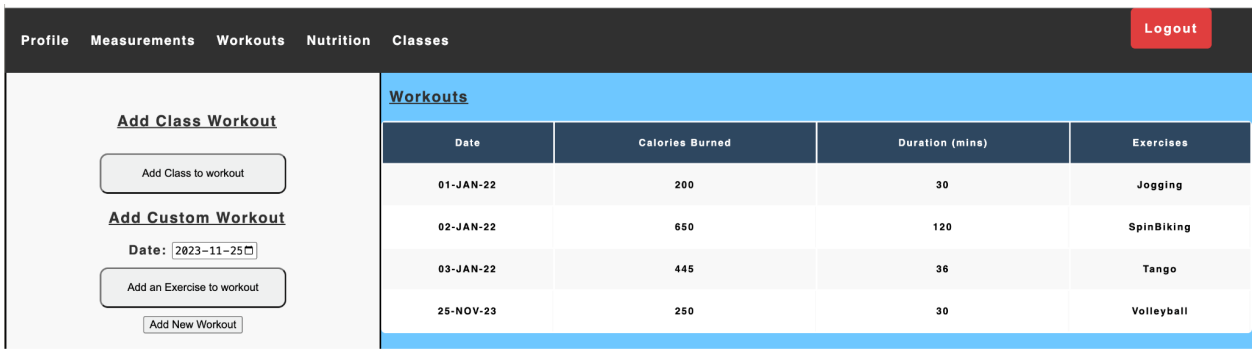
After adding exercise:



Before/During adding group class: Click “Add Class to workout” button, select available class, and click “Add New Workout” button



After add group class:



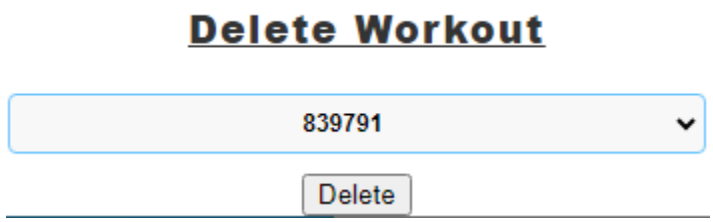
DELETE Operation: Deleting on Workout table has the cascade-on-delete situation on WorkoutIncludeExercise table

Workouts.php
Line 71:

```php
//DELETE WORKOUT
if(isset($_POST['delete_workout'])) {
    $wid = $_POST['wid-select'];

    //check if WID is a group class
    $checkWorkoutClassSQL = "SELECT * FROM GroupClass WHERE WID = :id";
    $checkWorkoutClassStmt = oci_parse($c, $checkWorkoutClassSQL);

    oci_bind_by_name($checkWorkoutClassStmt, ":id", $wid);
    oci_execute($checkWorkoutClassStmt);

    //Not a group class
    if(!oci_fetch_assoc($checkWorkoutClassStmt)) {
        //Delete workout from Workout
        $deleteWorkoutSQL = "DELETE FROM Workout WHERE WID = :id";
        $deleteWorkoutStmt = oci_parse($c, $deleteWorkoutSQL);

        oci_bind_by_name($deleteWorkoutStmt, ":id", $wid);
        if(oci_execute($deleteWorkoutStmt)) {
            $message = "Successfully deleted workout " . $wid;
        } else {
            $error = oci_error($deleteWorkoutStmt);
            $message = "Issue deleting workout " . $wid . ": " . $error;
        }
    } else {
        //Only delete workout from DoesWorkout
        $deleteWorkoutSQL = "DELETE FROM DoesWorkout WHERE WID = :id AND UserID = :userid";
        $deleteWorkoutStmt = oci_parse($c, $deleteWorkoutSQL);

        oci_bind_by_name($deleteWorkoutStmt, ":id", $wid);
        oci_bind_by_name($deleteWorkoutStmt, ":userid", $_SESSION['id']);

        if(oci_execute($deleteWorkoutStmt)) {
            $message = "Successfully deleted workout " . $wid;
        } else {
            $error = oci_error($deleteWorkoutStmt);
            $message = "Issue deleting workout " . $wid . ": " . $error;
        }
    }

    echo "<script>alert('$message');</script>";
```

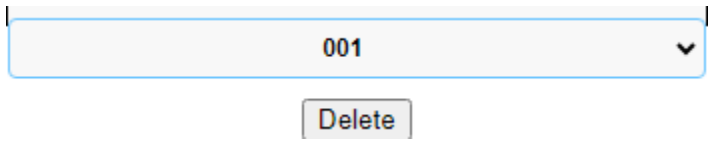
Before: User has 2 workouts that are a group class and a custom workout.

During:
Possibility 1: User deletes a custom workout



User chooses to delete WID of custom workout

Possibility 2: User deletes a workout that is linked to a group class



After for Possibility 1: Workouts no longer contains the deleted workout

Workouts

ID	Date	Calories Burned	Duration (mins)	Exercises
001	01-JUL-23	200	30	Jogging
002	15-SEP-23	650	120	SpinBiking

After for Possibility 2: Workouts no longer contains the deleted workout and the group class still contains the deleted workout '001'

Workouts

ID	Date	Calories Burned	Duration (mins)	Exercises
002	15-SEP-23	650	120	SpinBiking

Past Classes

Class ID	Class name	Class intensity	Class instructor	Price	Datetime	Duration	Capacity
001	Cardio Class	Low	trainerA	$100	01-JUL-23	30	30
002	Spin Class	Median	trainerB	$80	15-SEP-23	120	30
003	DanceFit Class	Low	trainerC	$90	20-SEP-23	36	30
004	Sports Class	Median	trainerD	$110	02-OCT-23	65	30
005	AquaFit Class	High	trainerE	$120	10-OCT-23	15	30
008	Cardio Class	Low	trainerA	$100	10-MAR-23	40	30

UPDATE Operation: User (admin) is able to Update GroupClass after selecting the tuples they want to update. The GroupClass Table contains one primary key “WID”, and two non-primary key “ClassTitle” and “Intensity”.

Classes.php
Line 121

```
$updatePriceSQL = "UPDATE Class_Price SET ClassPrice = :newPrice WHERE ClassTitle
= :className";
```

Line 229

```
$updateDateSQL = "UPDATE Workout SET WorkoutDate = TO_DATE(:newDate,
'YYYY-MM-DD') WHERE WID = :selectedID";
```

Line 246

```
$updateDurationSQL = "UPDATE Workout SET TotalDuration = :duration WHERE WID =
:selectedID";
```

Line 263

```
$updateClassTrainerSQL = "UPDATE Class_Trainer SET TID = :tID WHERE ClassTitle =
:selectedTitle";
```

Before:

Future Classes

Class ID	Class name	Class intensity	Class instructor	Price	Datetime	Duration	Capacity
006	HIIT Class	High	trainerF	$95	05-JAN-24	15	30
007	Zumba Class	Median	trainerG	$150	10-MAR-24	40	30

During (set Intensity for HIIT Class from High to Median, change the Price from $95 to $100):

Update Class

Choose a class to update: 006
Class Name: HIIT Class
Class Intensity: Median
Class Trainer: trainerF,6
Class Price: 100
Class Date: 2024-01-05
Class Duration: 15
Update Class

After:

Future Classes

Class ID	Class name	Class intensity	Class instructor	Price	Datetime	Duration	Capacity
006	HIIT Class	Median	trainerF	$100	05-JAN-24	15	30
007	Zumba Class	Median	trainerG	$150	10-MAR-24	40	30

SELECTION Query: User is able to filter through their measurements by Start Date, End Date, Weight, and Height which can be combined together using any combination of AND/OR
Measurement.php Line 267

```php
function displayMeasurements($c, $id, $filterstart = "", $filterend = "", $weightFilter = "",
$heightFilter = "", $weightOperator = "=", $heightOperator = "=",
$weightLogic = "AND", $heightLogic = "AND", $selectedAttributes = ["MDate", "Weight", "Height"],
$sortBy = "MDate", $extraDisplay = "") {
    // Initialize the conditions array
    $conditions = [];

    // Date filter conditions
    if ($filterstart !== "" && $filterend !== "") {
        $conditions[] = "(MDate >= TO_DATE(:start_date, 'YYYY-MM-DD') AND MDate <= TO_DATE(:end_date, 'YYYY-MM-DD'))";
    } elseif ($filterstart !== "") {
        $conditions[] = "MDate >= TO_DATE(:start_date, 'YYYY-MM-DD')";
    } elseif ($filterend !== "") {
        $conditions[] = "MDate <= TO_DATE(:end_date, 'YYYY-MM-DD')";
    }
```

```php
// Weight filter condition
if ($weightFilter !== "") {
    $weightCondition = "Weight $weightOperator :weightFilter";
}

// Height filter condition
if ($heightFilter !== "") {
    $heightCondition = "Height $heightOperator :heightFilter";
}
```

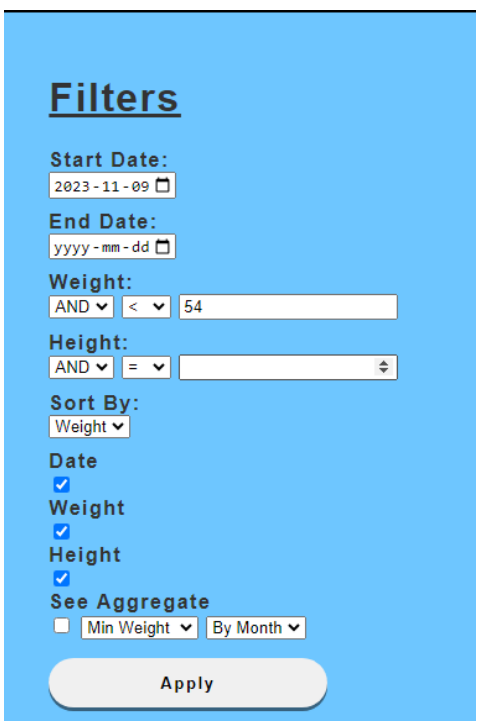
```php
// Combine date and weight conditions with weight logic
if (isset($weightCondition)) {
    if (!empty($conditions)) {
        $conditions = ["(" . implode(" AND ", $conditions) . ") $weightLogic $weightCondition"];
    } else {
        $conditions[] = $weightCondition;
    }
}

// Combine previous conditions with height condition using height logic
if (isset($heightCondition)) {
    if (!empty($conditions)) {
        $conditions = ["(" . implode(" ", $conditions) . ") $heightLogic $heightCondition"];
    } else {
        $conditions[] = $heightCondition;
    }
}
```

```php
// Construct the SQL query
$getMeasurementsSQL = "SELECT " . implode(", ", $selectedAttributes) . "
    FROM TrackMeasurement WHERE UserID = :user_id";
if (!empty($conditions)) {
    $getMeasurementsSQL .= " AND " . implode(" ", $conditions);
}
$getMeasurementsSQL .= " ORDER BY $sortBy";
```

Before Filtering:

MDate	Weight(kg)	Height(cm)
01-JAN-22	52	167.6
06-OCT-23	58.5	175
12-OCT-23	57.6	175
18-OCT-23	57	175
08-NOV-23	54	175
09-NOV-23	55	175
14-NOV-23	54.2	175
20-NOV-23	53.3	175
21-NOV-23	52.7	175

During Filter: User has selected the filter of starting after Nov 9, 2023, weight < 54, and sorted by weight



After Filtering: Only measurements that fit the filtering will appear

MDate	Weight(kg)	Height(cm)
21-NOV-23	52.7	175
20-NOV-23	53.3	175

Projection Query

Leaderboard.php

Line 14

```
$getPointsSQL = "SELECT u.UserID, u.Username,
NVL(measurement.MeasurementPoints, 0) AS
MeasurementPoints,NVL(groupclass.GroupClassPoints, 0) AS GroupClassPoints,
COALESCE(MeasurementPoints, 0) + COALESCE(GroupClassPoints, 0) AS
TotalPoints
                 FROM UserInfo u
                 LEFT JOIN (
                       SELECT
                           tm.UserID,
                           COUNT(tm.MDate) AS MeasurementPoints
                       FROM TrackMeasurement tm
                       WHERE tm.MDate >= ADD_MONTHS(TRUNC(SYSDATE,
'MONTH'), -3)
                       GROUP BY tm.UserID
                   ) measurement ON u.UserID = measurement.UserID
                   LEFT JOIN (
                       SELECT
                           u.UserID,
                           SUM(CASE WHEN gc.WID IS NOT NULL THEN 10 ELSE 0
END) AS GroupClassPoints
                       FROM UserInfo u
                       LEFT JOIN DoesWorkout dw ON u.UserID = dw.UserID
                       LEFT JOIN Workout w ON dw.WID = w.WID AND
w.WorkoutDate >= ADD_MONTHS(TRUNC(SYSDATE, 'MONTH'), -3)
                       LEFT JOIN GroupClass gc ON w.WID = gc.WID
                       GROUP BY u.UserID
                   ) groupclass ON u.UserID = groupclass.UserID
                   WHERE u.Username != 'admin'
                   ORDER BY TotalPoints DESC";
```

Join Query:

Workout table join with GroupClass, the Where clause is set together with other filters
Workouts.php Line 368

```
$getWorkoutsSQL = "SELECT Workout.WID, Workout.TotalCaloriesBurned,
Workout.WorkoutDate, Workout.TotalDuration, ";

if ($showClassTitle) {
      $getWorkoutsSQL .= "GroupClass.ClassTitle, ";
}

$getWorkoutsSQL .= "(SELECT LISTAGG(Exercises.ExerciseName, ', ') WITHIN
GROUP (ORDER BY Exercises.ExerciseName)
                    FROM WorkoutIncludeExercise
                    JOIN Exercises ON WorkoutIncludeExercise.ExerciseName
= Exercises.ExerciseName
                  WHERE WorkoutIncludeExercise.WID = Workout.WID) AS
ExercisesList
                  FROM DoesWorkout
                  JOIN Workout ON DoesWorkout.WID = Workout.WID ";

if ($showClassTitle) {
    $getWorkoutsSQL .= " LEFT JOIN GroupClass ON Workout.WID =
GroupClass.WID ";
}

$getWorkoutsSQL .= " WHERE DoesWorkout.UserID = :userID";
```

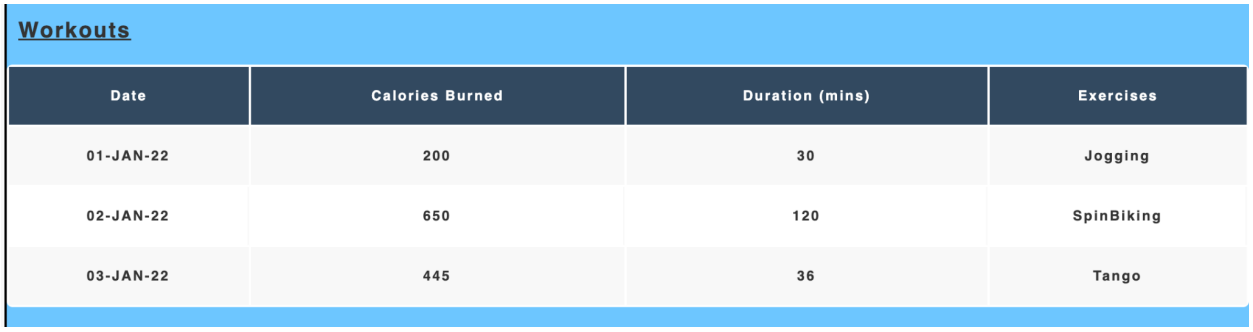
Before:

Workouts

Date	Calories Burned	Duration (mins)	Exercises	Class
01-JUL-23	200	30	Jogging	
15-SEP-23	650	120	SpinBiking	

During: (Select “Show Class Title” and click on “Apply”, the where clause can be set with other filters together since the JOIN is a LEFT JOIN)

After: the workouts table will join with GroupClass to show the class title



Workouts

Date	Calories Burned	Duration (mins)	Exercises
01-JAN-22	200	30	Jogging
02-JAN-22	650	120	SpinBiking
03-JAN-22	445	36	Tango

Aggregation with GROUP BY: Calculate and display a users min, max, and avg weight across a certain period of time

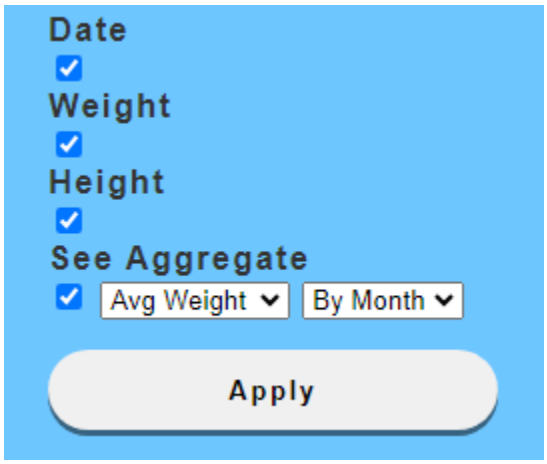
Measurements.php
Line 331

```
if ($extraDisplay !== "") {
                            // Determine the SQL function for aggregation
                            $aggFunc = ($extraDisplay == "Avg") ? "AVG" :
(($extraDisplay == "Max") ? "MAX" : "MIN");

                            // Determine the grouping field (Month/Year)
                            $groupByField = ($aggPeriod == "Year") ?
"EXTRACT(YEAR FROM MDate)" : "TO_CHAR(MDate, 'MM/YYYY')";

                            // Construct SQL for aggregation
```

```
                            $aggSQL = "SELECT $groupByField AS Period,
$aggFunc(Weight) AS AggWeight FROM TrackMeasurement WHERE UserID = :user_id ";
                            if ($filterstart !== "" && $filterend !== "") {
                                $aggSQL .= " AND MDate BETWEEN
TO_DATE(:start_date, 'YYYY-MM-DD') AND TO_DATE(:end_date, 'YYYY-MM-DD')";
                            }
                            $aggSQL .= " GROUP BY $groupByField ORDER BY
$groupByField";

                            // Prepare and execute the SQL statement for
aggregation
                            $aggStmt = oci_parse($c, $aggSQL);
                            oci_bind_by_name($aggStmt, ":user_id", $id);
                            if ($filterstart !== "" && $filterend !== "") {
                                oci_bind_by_name($aggStmt, ":start_date",
$filterstart);
                                oci_bind_by_name($aggStmt, ":end_date",
$filterend);
                            }

                            if (oci_execute($aggStmt)) {
                                echo "<table>";
                                echo "<tr><th>Period</th><th>{$extraDisplay}
Weight</th></tr>";
                                while($row = oci_fetch_assoc($aggStmt)) {
                                    echo "<tr><td>" .
htmlspecialchars($row['PERIOD']) . "</td><td>" .
htmlspecialchars($row['AGGWEIGHT']) . "</td></tr>";
                                }
                                echo "</table>";
                            } else {
                                $error = oci_error($aggStmt);
                                echo "Failed to retrieve aggregate data. Error: "
. $error['message'];
                            }

                            oci_free_statement($aggStmt);
                        }
```

Before Aggregation with Group By: Before selecting Aggregation, there extra display table is not visible

During Aggregation with Group By: User can choose between seeing the Max, Min, or Avg Weight By Month or By Year



After Aggregation with Group By:

Period	Avg Weight
01/2022	52
10/2023	57.7
11/2023	53.84

Aggregation with HAVING: User is able to filter for workouts that have an exercise count or duration <,>, or = to an inputted value

Workouts.php
Line 612

```php
function buildAdvancedDisplaySQL($attribute, $comparison, $amount, $userID, $filterstart, $filterend) {
    $advDisplaySQL = "";

    if ($attribute == "Exercise Count") {
        $advDisplaySQL = "SELECT we.ExerciseName, COUNT(*) AS ExerciseCount
                          FROM DoesWorkout
                          JOIN Workout ON DoesWorkout.WID = Workout.WID
                          LEFT JOIN WorkoutIncludeExercise we ON Workout.WID = we.WID
                          WHERE DoesWorkout.UserID = :userID";

        if ($filterstart !== "") {
            $advDisplaySQL .= " AND Workout.WorkoutDate >= TO_DATE(:start_date, 'YYYY-MM-DD')";
        }
        if ($filterend !== "") {
            $advDisplaySQL .= " AND Workout.WorkoutDate <= TO_DATE(:end_date, 'YYYY-MM-DD')";
        }

        $advDisplaySQL .= " GROUP BY we.ExerciseName
                           HAVING COUNT(*) $comparison :value";
    } elseif ($attribute == "Exercise Duration") {
        $advDisplaySQL = "SELECT we.ExerciseName, SUM(we.Duration) AS TotalDuration
                          FROM DoesWorkout
                          JOIN Workout ON DoesWorkout.WID = Workout.WID
                          LEFT JOIN WorkoutIncludeExercise we ON Workout.WID = we.WID
                          WHERE DoesWorkout.UserID = :userID";

        if ($filterstart !== "") {
            $advDisplaySQL .= " AND Workout.WorkoutDate >= TO_DATE(:start_date, 'YYYY-MM-DD')";
        }
        if ($filterend !== "") {
            $advDisplaySQL .= " AND Workout.WorkoutDate <= TO_DATE(:end_date, 'YYYY-MM-DD')";
        }

        $advDisplaySQL .= " GROUP BY we.ExerciseName
                           HAVING SUM(we.Duration) $comparison :value";
    }

    return $advDisplaySQL;
}
```

Before Aggregation with Having: The advanced filter display does not appear until the user has submitted the parameters for the query

During Aggregation with Having:

After Aggregation with Having:

Advanced Display: Exercise Duration>40

Date	Calories Burned	Duration (mins)	Exercises
15-SEP-23	650	120	SpinBiking

Nested Aggregation with GROUP BY: Admin User is able to see the average (measurements submitted or classes attended) per month for each user, in the last n months

Tables.php
Line 163:

```
if($att == 'measurement') {
    $getAverageSQL = "SELECT
                        u.UserID,
                        u.Username,
                        COALESCE(SUM(MeasurementsPerMonth), 0) AS TotalMeasurements,
                        ROUND(NVL(SUM(MeasurementsPerMonth) / :numMonths, 0), 2) AS AverageMeasurements
                    FROM UserInfo u
                    LEFT JOIN (
                        SELECT
                            tm.UserID,
                            COUNT(tm.MDate) AS MeasurementsPerMonth
                        FROM TrackMeasurement tm
                        WHERE tm.MDate >= ADD_MONTHS(TRUNC(SYSDATE, 'MONTH'), -:numMonths)
                        GROUP BY tm.UserID
                    ) measurement ON u.UserID = measurement.UserID
                    WHERE u.Username != 'admin'
                    GROUP BY u.UserID, u.Username
                    ORDER BY AverageMeasurements DESC";
} else {
    $getAverageSQL = "SELECT
                        u.UserID,
                        u.Username,
                        COALESCE(SUM(ClassesPerMonth), 0) AS TotalClasses,
                        ROUND(NVL(SUM(ClassesPerMonth) / :numMonths, 0), 2) AS AverageClasses
                    FROM UserInfo u
                    LEFT JOIN (
                        SELECT
                            u.UserID,
                            SUM(CASE WHEN gc.WID IS NOT NULL THEN 1 ELSE 0 END) AS ClassesPerMonth
                        FROM UserInfo u
                        LEFT JOIN DoesWorkout dw on u.UserID = dw.UserID
                        LEFT JOIN Workout w ON dw.WID = w.WID AND w.WorkoutDate >= ADD_MONTHS(TRUNC(SYSDATE, 'MONTH'), -:numMonths)
                        LEFT JOIN GroupClass gc ON w.WID = gc.WID
                        GROUP BY u.UserID
                    ) groupclass ON u.UserID = groupclass.UserID
                    WHERE u.Username != 'admin'
                    GROUP BY u.UserID, u.Username
                    ORDER BY AverageClasses DESC";
}
```

Before Nested Aggregation with GROUP BY: There is no table displayed prior to submitting the form

During Nested Aggregation with GROUP BY: User has selected to display the average measurements submitted by each User in the last 2 months

See Averages

See average measurements submitted per month for each User, in the last 2

View Average

After Nested Aggregation with GROUP BY:

Average Number of Measurements Submitted in Last 2 Months

User ID	User Name	Total Measurements	Average # Of Measurements
001	userA	8	4
002	userB	1	.5
003	userC	0	0
004	userD	0	0
005	userE	0	0

DIVISION Query: Admin User is able to select any number of past group classes and view the selected information of users who have attended some or all of the classes

Classes.php
Line 563

```
$displayAtts = array();

if(isset($_POST['see_user_names'])) {
    $username = "UserInfo." . $_POST['see_user_names'];
    array_push($displayAtts, $username);
}
if(isset($_POST['see_user_ages'])) {
    $userage = "UserInfo." . $_POST['see_user_ages'];
    array_push($displayAtts, $userage);
}
if(isset($_POST['see_user_gender'])) {
    $usergender = "UserInfo." . $_POST['see_user_gender'];
    array_push($displayAtts, $usergender);
}

$seeClasses = $_POST['class_name'];
$attendanceSQL = null;
if(!empty($displayAtts) && !empty($seeClasses)) {
    if($_POST['all_some_select'] == "All") {
        $attendanceSQL = "SELECT UserInfo.UserID, " . implode(", ", $displayAtts) . " FROM DoesWorkout
        JOIN UserInfo ON DoesWorkout.UserID = UserInfo.UserID
        JOIN GroupClass ON GroupClass.WID = DoesWorkout.WID
        WHERE DoesWorkout.WID IN ('" . implode("', '", $seeClasses) . "')
        GROUP BY UserInfo.UserID, " . implode(", ", $displayAtts) . "
        HAVING COUNT(DISTINCT GroupClass.ClassTitle) = " . count($seeClasses);
    } else {
        $attendanceSQL = "SELECT GroupClass.ClassTitle,
        LISTAGG(UserInfo.UserID || ', ' || " . implode(" || ', ' || ", $displayAtts) . " || '|', CHR(10))
        WITHIN GROUP (ORDER BY UserInfo.UserID) AS UserList
        FROM DoesWorkout
        JOIN UserInfo ON DoesWorkout.UserID = UserInfo.UserID
        JOIN GroupClass ON GroupClass.WID = DoesWorkout.WID
        WHERE DoesWorkout.WID IN ('" . implode("', '", $seeClasses) . "')
        GROUP BY GroupClass.ClassTitle";

    }
    $attendanceStmt = oci_parse($c, $attendanceSQL);
```

Before Division: None of the past classes have been selected and form has not been submitted

Past Classes

Class ID	Class name	Class intensity	Class instructor	Price	Datetime	Duration	Capacity	
001	Cardio Class	Low	trainerA	$100	01-JUL-23	30	30	☐
002	Spin Class	Median	trainerB	$80	15-SEP-23	120	30	☐
003	DanceFit Class	Low	trainerC	$90	20-SEP-23	36	30	☐
004	Sports Class	Median	trainerD	$110	02-OCT-23	65	30	☐
005	AquaFit Class	High	trainerE	$120	10-OCT-23	15	30	☐
008	Cardio Class	Low	trainerA	$100	10-MAR-23	40	30	☐

Class/User Division

Usernames ☑
Ages ☑
Gender ☑
See user who have attended some of the selected classes
See User Attendance

During Division: Classes have been selected as well as visible user attributes and some/all selection

Past Classes

Class ID	Class name	Class intensity	Class instructor	Price	Datetime	Duration	Capacity	
001	Cardio Class	Low	trainerA	$100	01-JUL-23	30	30	☐
002	Spin Class	Median	trainerB	$80	15-SEP-23	120	30	☑
003	DanceFit Class	Low	trainerC	$90	20-SEP-23	36	30	☑
004	Sports Class	Median	trainerD	$110	02-OCT-23	65	30	☐
005	AquaFit Class	High	trainerE	$120	10-OCT-23	15	30	☐
008	Cardio Class	Low	trainerA	$100	10-MAR-23	40	30	☐

Class/User Division

Usernames ☑
Ages ☑
Gender ☐
See user who have attended all of the selected classes
See User Attendance

After Division: The division table is displayed showing userID, username, and age for users that have attended both classes ‘002’, and ‘003’

Past Classes

Class ID	Class name	Class intensity	Class instructor	Price	Datetime	Duration	Capacity	
001	Cardio Class	Low	trainerA	$100	01-JUL-23	30	30	☐
002	Spin Class	Median	trainerB	$80	15-SEP-23	120	30	☐
003	DanceFit Class	Low	trainerC	$90	20-SEP-23	36	30	☐
004	Sports Class	Median	trainerD	$110	02-OCT-23	65	30	☐
005	AquaFit Class	High	trainerE	$120	10-OCT-23	15	30	☐
008	Cardio Class	Low	trainerA	$100	10-MAR-23	40	30	☐

Class/User Division

Usernames ☑
Ages ☑
Gender ☑
See user who have attended all of the selected classes
See User Attendance

User ID	Username	User Age
002	userB	41