

Group 6: Bibek Manandhar, Sachin Karki

Dr. Omar Darwish

Database Management, CSCI 3423

21st November, 2020

Final Project Implementation - Nutrition Tracking Database

1. Project Goals

As mentioned, and discussed previously health is a major aspect of our life. Health is what helps us to determine the entirety of our well-beingness and depicts the state of human anatomy in terms of physical, mental and emotional aspects. In order to ensure sound health, one needs to monitor eating habits and physical activities to make sure they are on good track. An easy way to manage and monitor those habits and activities is by using a proper record keeping system. The purpose and goal of this project is to become that proper record keeping system. Nutrition Tracking database is capable of recording information that can help one to understand the pattern of their health progress. Not only it will keep the record of their diet and exercise, it will aware people about the state of their health.

2. Database Description

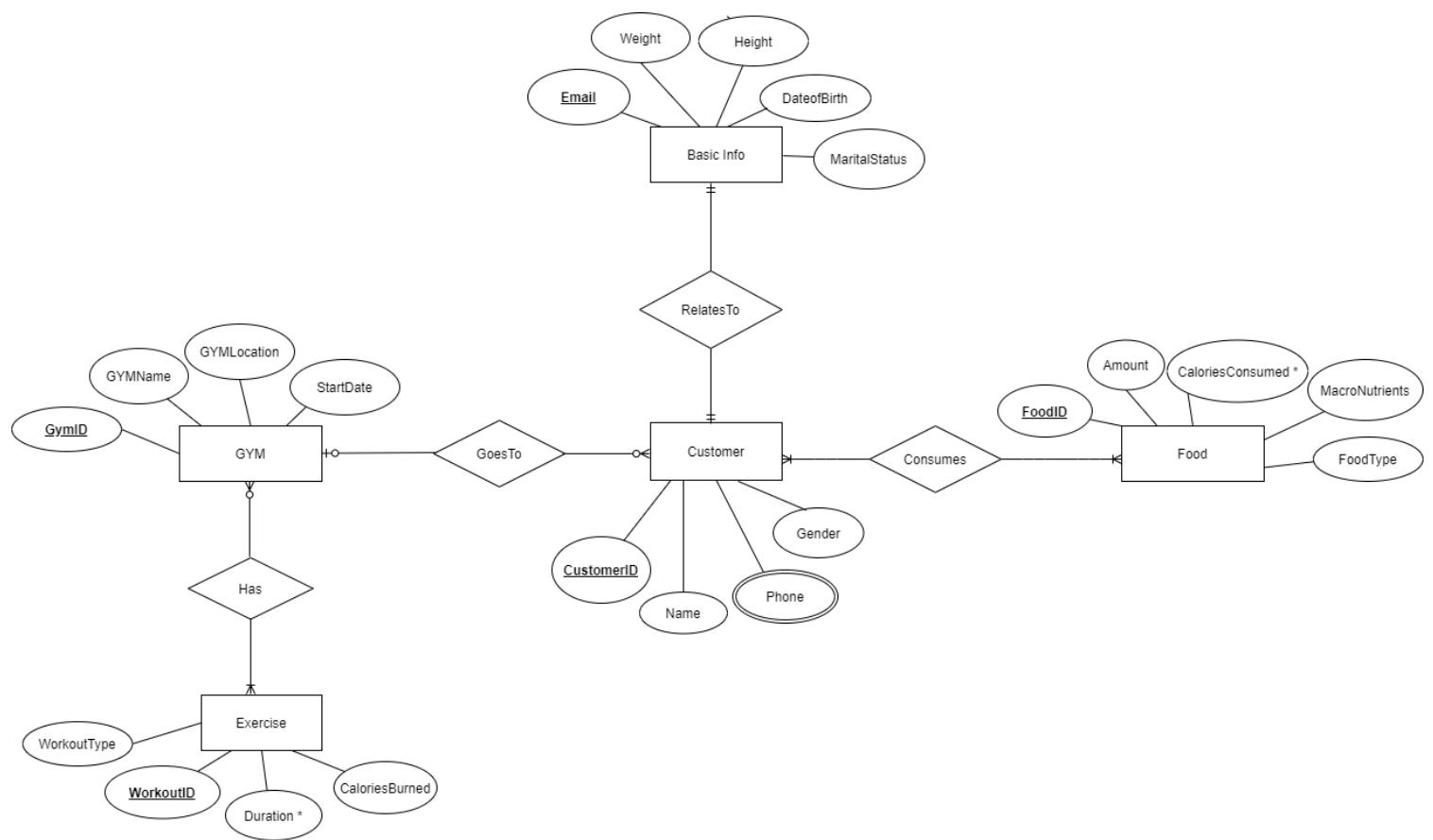
Nutrition Tracking Database does exactly what it sounds like. The database consists of several information and class to make it possible to record the data. For an instance, the database relies on three sets of account to track a data. These three sets are Customer Information, Food and Exercise. However, these three sets can be expanded into multiple sets of additional accounts to make it more relevant and accurate. To begin with, customer information will mostly store

general information of a customer like their name, gender, date of birth, height and weight. In addition to this, customer ID, marital status and contact information adds up to make that part more solid. Food is another entity that specializes in recording what type of food they ate, what was the amount of the food they ate, how much kilo calories did they gain. Lastly, exercise is another aspectual entity that takes advantage in recording what sort of activities did they perform whether it is cardio, stretch or lift. For how long did they work out and how much kilo calories did they burn. Typically, 120 kilo calories are burned by 30 minutes of walking and somewhere between 150-170 kilo calories is burned by an intense workout session.

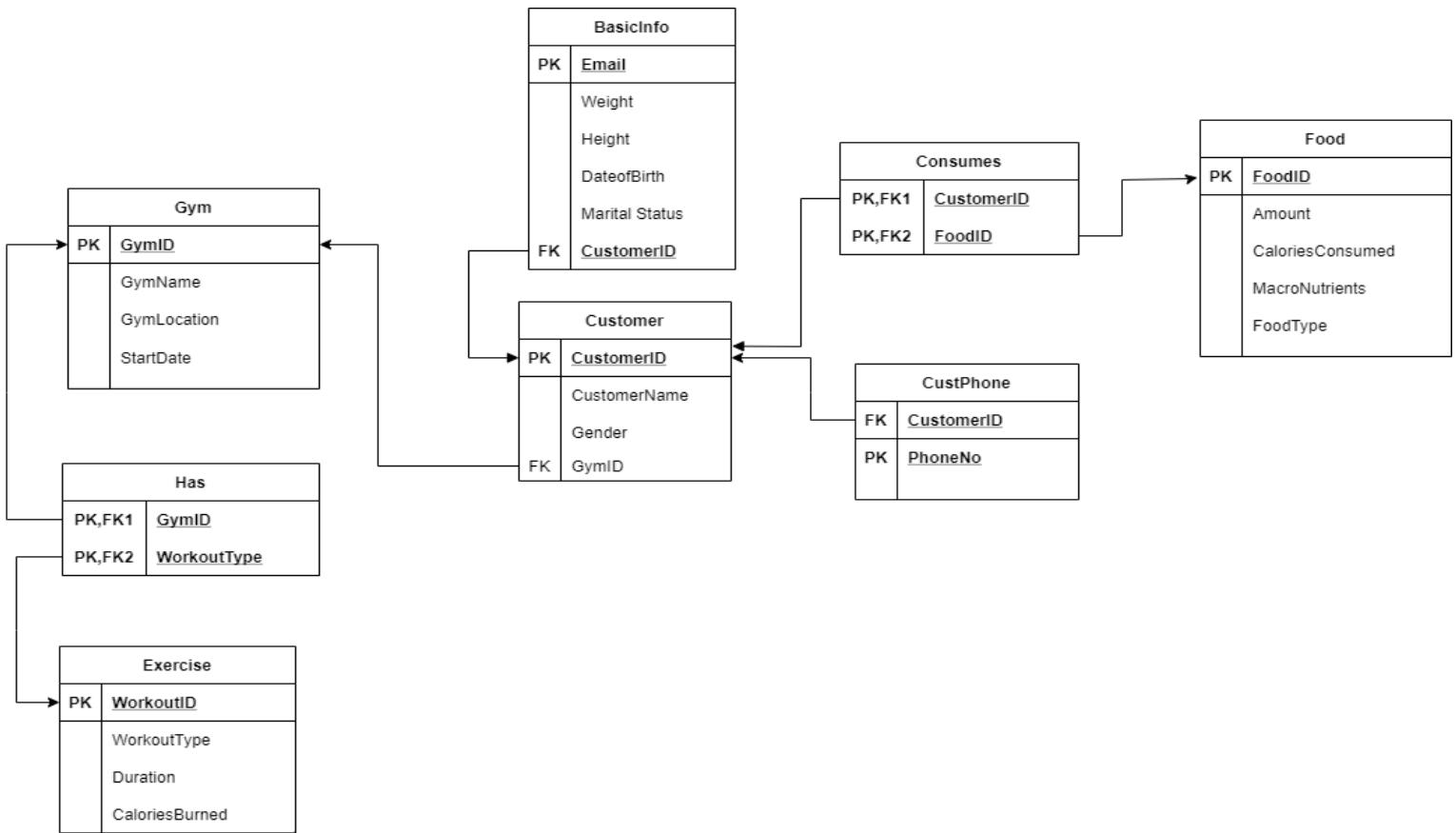
This information will pave a way for us to know exactly what kind of workout will help us burn how much of calories. Users will benefit in lot of ways with an effective and consistent usage of the database. With an accurate and precise set of information they feed, users will be able to keep track of overall calorie consumption. Every food has a different calorie count, this would allow them to be concise about which food to take and which to avoid. Users are also able to realize the right set of workouts for them. With regular update on calorie gain and calorie loss, user will have the insights on calorie ration. To sum up, users will benefit the privilege of being able to calculate BMI (Body Mass Index) easily based on these records and calculation.

3. Data Model & Design

- a. ER Diagrams:* For ER diagrams, all the resulting and relating sketches were done in draw.io website. After creative thinking and drafting a rough version of the database, a conceptual model was envisioned which was later on improvised and improved with the table and columns that were created early in the process. The ER was further developed into a meaningful relational schema that allowed us to shape and code the entirety of the database we created later on. Both ER and Relational Schema are pictured below.

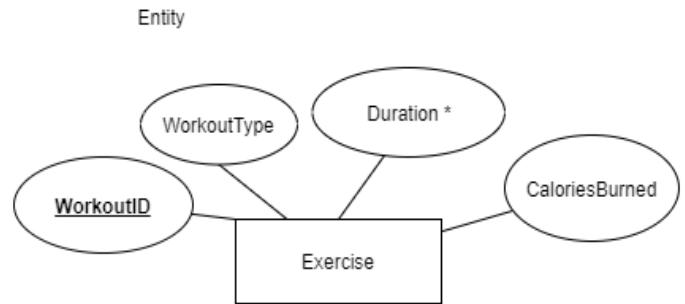
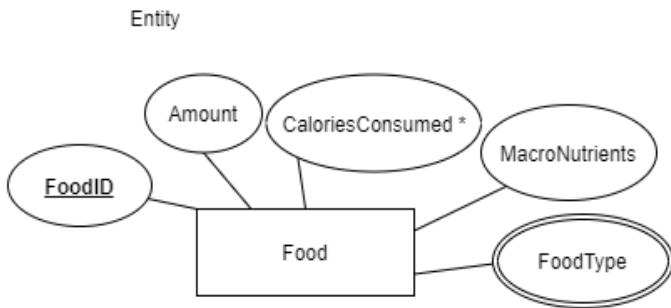


ER Diagram of the Nutrition Tracking Database



Relational Schema of Nutrition Tracking Database

b. Business Rules: A couple of business rules were added in the database to keep it more relevant and accurate. One of the tables that was created in the development phase demanded an adjustment for different gender. In the food entity, it was realized that male and female have different requirement for daily calorie intake. Male typically needs a 2500 kcal per day while female needs 2000 kcal per day. Therefore, a business rule was added in calorie consumption. Similarly, duration of workout also needed an adjustment. To qualify an activity as a ‘workout’ a minimum duration of that activity performed is needed. That being realized a minimum of 45 minutes of workout is a must. That way it was possible for us to add another business rule in duration of exercise entity. These two business rules are pictured below.



* Calorie Consumption Must be a Minimum of 2000 for Women and 2500 for Men

* Duration of a workout must be 45 minutes minimum and 2 hours maximum

Resulting Relation

| Food | |
|------|------------------|
| PK | FoodID |
| | Amount |
| | CaloriesConsumed |
| | MacroNutrients |

Business Rule for Calorie Consumption

Resulting Relation

| Exercise | |
|----------|---------------|
| PK | WorkoutID |
| | WorkoutType |
| | Duration |
| | CalorieBurned |

Business Rule for Duration of Workout

c. **Data Dictionary:** A quick way to access the information from the database is by using a data dictionary. To overview and sum up major aspects of the database, listed below is the data dictionary of the Nutrition Tracking Database.

| TableName | ColumnDescription | DataTypes | ColumnRestrictions |
|-----------|-------------------|-----------|--------------------|
| BasicInfo | Email | VARCHAR | 50 |
| BasicInfo | Weight | VARCHAR | 10 |
| BasicInfo | Height | VARCHAR | 10 |
| BasicInfo | DateofBirth | DATE | DATE |
| BasicInfo | MaritalStatus | CHAR | 15 |
| Customer | CustomerID | CHAR | 3 |
| Customer | CustomerName | VARCHAR | 40 |
| Customer | Gender | CHAR | 10 |
| CustPhone | PhoneNumber | VARCHAR | 12 |
| CustPhone | CustomerID | CHAR | 3 |
| Food | FoodID | VARCHAR | 3 |
| Food | FoodType | CHAR | 15 |
| Food | Amount | FLOAT | 15 |
| Food | CalorieConsumed | FLOAT | 20 |
| Food | MacroNutrients | CHAR | 20 |
| Consumes | CustomerID | CHAR | 3 |
| Consumes | FoodID | VARCHAR | 3 |
| Gym | GymID | CHAR | 5 |

| | | | |
|----------|----------------|---------|------|
| Gym | GymName | VARCHAR | 30 |
| Gym | GymLocation | VARCHAR | 30 |
| Gym | StartDate | DATE | DATE |
| Gym | CustomerID | CHAR | 3 |
| Exercise | WorkoutID | VARCHAR | 3 |
| Exercise | WorkoutType | CHAR | 40 |
| Exercise | Duration | VARCHAR | 20 |
| Exercise | CaloriesBurned | FLOAT | 15 |
| Has | GymID | CHAR | 5 |
| Has | WorkoutID | VARCHAR | 3 |

4. Implementation

a. *Creating Database in MySQL Server and MySQL Workbench:* After creating entity relation diagram and sketching relational schema for the database we used MySQL Workbench to carefully code each tables and columns. The resulting code is down below.

Create schema Project;
Use Project;

SQL Code:

```
CREATE TABLE gym (
    gymid             CHAR(5)          NOT NULL,
    gymname           VARCHAR(30)      NOT NULL,
    gymlocation       VARCHAR(30)      NOT NULL,
    startdate         DATE            NOT NULL,
    PRIMARY KEY (gymid)
);
CREATE TABLE customer (
    customerid        CHAR(3)          NOT NULL,
    customername      VARCHAR(40)      NOT NULL,
    gender            CHAR(10)         NOT NULL,
    gymid CHAR(5) NOT NULL,
    PRIMARY KEY (customerid),
    FOREIGN KEY (gymid) REFERENCES gym (gymid)
);
CREATE TABLE basicinfo (
    email             VARCHAR(50)       NOT NULL,
    weight            VARCHAR(10)       NOT NULL,
    height            VARCHAR(10)       NOT NULL,
    dateofbirth       DATE            NOT NULL,
    maritalstatus     CHAR(15)         NOT NULL,
    customerid        CHAR(3)          NOT NULL,
    PRIMARY KEY (email),
    FOREIGN KEY (customerid) REFERENCES customer (customerid)
);
CREATE TABLE custphone (
    phonenumbers      VARCHAR(12)       NOT NULL,
    customerid        CHAR(3)          NOT NULL,
    FOREIGN KEY (customerid) REFERENCES customer (customerid)
);
CREATE TABLE food (
    foodid            VARCHAR(3)        NOT NULL,
    foodtype          CHAR(15)         NOT NULL,
    amount            FLOAT            NOT NULL,
    caloriesconsumed FLOAT            NOT NULL,
    macronutrients   CHAR(20)         NOT NULL,
    PRIMARY KEY (foodid)
);
CREATE TABLE consumes (
    customerid        CHAR(3)          NOT NULL,
    foodid            VARCHAR(3)        NOT NULL,
    FOREIGN KEY (customerid) REFERENCES customer (customerid),
    FOREIGN KEY (foodid) REFERENCES food (foodid)
);
CREATE TABLE exercise (
    workoutid          VARCHAR(3)       NOT NULL,
    workouttype        CHAR(40)         NOT NULL,
    duration           VARCHAR(20)       NOT NULL,
    caloriesburned    FLOAT           NOT NULL,
    PRIMARY KEY (workoutid)
);
CREATE TABLE has (
    gymid             CHAR(5)          NOT NULL,
    workoutid          VARCHAR(3)       NOT NULL,
    FOREIGN KEY (gymid) REFERENCES gym (gymid),
    FOREIGN KEY (workoutid) REFERENCES exercise (workoutid)
);
```

b. Populating the Database: After coding the required tables and columns, the next step was to feed records and information into the database. In order to populate, 20 entries of records were given into the database. The resulting code is given below.

i. Populating the customer column

```
INSERT INTO customer VALUES ('101','Ted Marquis','Male','34812');  
INSERT INTO customer VALUES ('102','Sandy Yule','Male','20318');  
INSERT INTO customer VALUES ('103','Rosendo Hefner','Male','39028');  
INSERT INTO customer VALUES ('104','Quentin Tibbits','Male','28730');  
INSERT INTO customer VALUES ('105','Jonathan Thiem','Male','27364');  
INSERT INTO customer VALUES ('106','Clayton Parekh','Male','17283');  
INSERT INTO customer VALUES ('107','Christoper Lofland','Male','26354');  
INSERT INTO customer VALUES ('108','Alejandro Kubiak','Male','12093');  
INSERT INTO customer VALUES ('109','Carlos Albino','Male','28374');  
INSERT INTO customer VALUES ('110','Cruz Burtch','Male','29837');  
INSERT INTO customer VALUES ('111','Annmarie Whetstone','Female','26743');  
INSERT INTO customer VALUES ('112','Vannesa Uresti','Female','24534');  
INSERT INTO customer VALUES ('113','Olene Greco','Female','26712');  
INSERT INTO customer VALUES ('114','Cyndy Sells','Female','19023');  
INSERT INTO customer VALUES ('115','Yajaira Oyler','Female','10293');  
INSERT INTO customer VALUES ('116','Jennette Tolman','Female','12283');  
INSERT INTO customer VALUES ('117','Antonina Magee','Female','28839');  
INSERT INTO customer VALUES ('118','Rose Zerby','Female','18297');  
INSERT INTO customer VALUES ('119','Delphia Galang','Female','21723');  
INSERT INTO customer VALUES ('120','Alona Kubota','Female','12834');
```

ii. Populating the basicinfo column

```
INSERT INTO basicinfo VALUES ('tedmarquis@twu.edu', '101 kg', '173 cm', '1975/12/7', 'Married', '101');  
INSERT INTO basicinfo VALUES ('sandyyule@twu.edu', '120 kg', '149 cm', '1974/01/24', 'Unmarried', '102');  
INSERT INTO basicinfo VALUES ('rosendohefner@twu.edu', '100 kg', '160 cm', '1992/12/17', 'Married', '103');  
INSERT INTO basicinfo VALUES ('quentintibbitis@twu.edu', '95 kg', '153 cm', '1990/12/27', 'Married', '104');  
INSERT INTO basicinfo VALUES ('jonathanthiem@twu.edu', '79 kg', '123 cm', '1975/05/13', 'Unmarried', '105');  
INSERT INTO basicinfo VALUES ('claytonparekh@twu.edu', '75 kg', '119 cm', '1971/09/27', 'Married', '106');  
INSERT INTO basicinfo VALUES ('christoperlofland@gmail.com', '72 kg', '113 cm', '1973/10/14', 'Unmarried', '107');  
INSERT INTO basicinfo VALUES ('alejandrokubiak@gmail.com', '97 kg', '134 cm', '1982/09/20', 'unmarried', '108');  
INSERT INTO basicinfo VALUES ('carlosalbina@rocketmail.com', '95 kg', '145 cm', '1988/06/27', 'Married', '109');  
INSERT INTO basicinfo VALUES ('cruzburth@yahoo.com', '65 kg', '151 cm', '1974/12/23', 'Unmarried', '110');  
INSERT INTO basicinfo VALUES ('annamariewhetsone@gmail.com', '99 kg', '155 cm', '1993/11/26', 'Married', '111');  
INSERT INTO basicinfo VALUES ('vannesuaresti@gmail.com', '75 kg', '167 cm', '1982/03/07', 'Married', '112');  
INSERT INTO basicinfo VALUES ('olenegreco@rocketmail.com', '150 kg', '179 cm', '1984/01/05', 'Unmarried', '113');  
INSERT INTO basicinfo VALUES ('cyndysells@gmail.com', '110 kg', '164 cm', '1999/12/24', 'Married', '114');  
INSERT INTO basicinfo VALUES ('yajairaoyle@rocketmail.com', '90 kg', '132 cm', '1972/05/08', 'Unmarried', '115');  
INSERT INTO basicinfo VALUES ('jennettetolman@yahoo.com', '80 kg', '136 cm', '1992/10/06', 'Married', '116');  
INSERT INTO basicinfo VALUES ('antoninamagee@yahoo.com', '71 kg', '167 cm', '1997/03/08', 'Unmarried', '117');  
INSERT INTO basicinfo VALUES ('rosezerby@gmail.com', '55 kg', '148 cm', '1981/09/26', 'Unmarried', '118');  
INSERT INTO basicinfo VALUES ('delphiagalang@twu.edu', '65 kg', '131 cm', '1980/11/19', 'Married', '119');  
INSERT INTO basicinfo VALUES ('alonakubota@gmail.com', '110 kg', '124 cm', '2000/06/05', 'Unmarried', '120');
```

iii. Populating into custphone column

```
INSERT INTO custphone VALUES ('301-559-5574','101');
INSERT INTO custphone VALUES ('786-973-5150','102');
INSERT INTO custphone VALUES ('804-720-9568','103');
INSERT INTO custphone VALUES ('804-755-2917','103');
INSERT INTO custphone VALUES ('660-256-0696','104');
INSERT INTO custphone VALUES ('305-608-3417','105');
INSERT INTO custphone VALUES ('815-867-0514','105');
INSERT INTO custphone VALUES ('704-328-0960','106');
INSERT INTO custphone VALUES ('602-819-1326','106');
INSERT INTO custphone VALUES ('773-972-6969','106');
INSERT INTO custphone VALUES ('415-785-9286','107');
INSERT INTO custphone VALUES ('480-610-5150','108');
INSERT INTO custphone VALUES ('917-470-4749','109');
INSERT INTO custphone VALUES ('323-554-0534','110');
INSERT INTO custphone VALUES ('817-703-8762','110');
INSERT INTO custphone VALUES ('406-513-2388','111');
INSERT INTO custphone VALUES ('989-501-1080','112');
INSERT INTO custphone VALUES ('806-776-6909','113');
INSERT INTO custphone VALUES ('206-666-3820','113');
INSERT INTO custphone VALUES ('734-549-0888','114');
INSERT INTO custphone VALUES ('302-788-7604','114');
INSERT INTO custphone VALUES ('941-348-9673','115');
INSERT INTO custphone VALUES ('907-430-4062','116');
INSERT INTO custphone VALUES ('704-453-1826','117');
INSERT INTO custphone VALUES ('347-669-2153','118');
INSERT INTO custphone VALUES ('435-239-8387','119');
INSERT INTO custphone VALUES ('301-387-8675','120');
```

iv. Populating into food

```
INSERT INTO food VALUES ('F1','Vegetarian',1809, 2100, 'Protein' );
INSERT INTO food VALUES ('F2','Vegan',1720,1980, 'Fiber');
INSERT INTO food VALUES ('F3','Low Carbs',1600, 1990, 'Carbs' );
INSERT INTO food VALUES ('F4','Plant Based',2000, 2400, 'Protein' );
INSERT INTO food VALUES ('F5','Zone Diet',1700, 2600, 'Fiber' );
INSERT INTO food VALUES ('F6','Ketogenic',1660, 2700, 'Carbs' );
INSERT INTO food VALUES ('F7','Meat Based',1570,3200, 'Protein' );
INSERT INTO food VALUES ('F8','Zone Diet',1293, 2300, 'Carbs' );
INSERT INTO food VALUES ('F9','Vegetarian',1578, 2900, 'Fiber' );
INSERT INTO food VALUES ('F10','Low Carbs',1890, 2500, 'Protein' );
INSERT INTO food VALUES ('F11','Plant Based',1782, 2900, 'Fiber' );
INSERT INTO food VALUES ('F12','Ketogenic',1453, 2199, 'Protein' );
INSERT INTO food VALUES ('F13','Vegan',1341, 3400, 'Carbs' );
INSERT INTO food VALUES ('F14','Plant Based',2131, 2300, 'Sugar' );
INSERT INTO food VALUES ('F15','Ketogenic',2100, 2400, 'Sugar' );
INSERT INTO food VALUES ('F16','Low Carbs',1400, 2560, 'Fiber' );
INSERT INTO food VALUES ('F17','Vegetarian',2000, 2100, 'Carbs' );
INSERT INTO food VALUES ('F18','Plant Based',1740, 2810, 'Protein' );
INSERT INTO food VALUES ('F19','Vegan',1400, 2710, 'Carbs' );
INSERT INTO food VALUES ('F20','Ketogenic',1274, 2680, 'Protein' );
```

v. Populating into consumes

```
INSERT INTO consumes VALUES ('101','F1');  
INSERT INTO consumes VALUES ('102','F2');  
INSERT INTO consumes VALUES ('103','F3');  
INSERT INTO consumes VALUES ('104','F4');  
INSERT INTO consumes VALUES ('105','F5');  
INSERT INTO consumes VALUES ('106','F6');  
INSERT INTO consumes VALUES ('107','F7');  
INSERT INTO consumes VALUES ('108','F8');  
INSERT INTO consumes VALUES ('109','F9');  
INSERT INTO consumes VALUES ('110','F10');  
INSERT INTO consumes VALUES ('111','F11');  
INSERT INTO consumes VALUES ('112','F12');  
INSERT INTO consumes VALUES ('113','F13');  
INSERT INTO consumes VALUES ('114','F14');  
INSERT INTO consumes VALUES ('115','F15');  
INSERT INTO consumes VALUES ('116','F16');  
INSERT INTO consumes VALUES ('117','F17');  
INSERT INTO consumes VALUES ('118','F18');  
INSERT INTO consumes VALUES ('119','F19');  
INSERT INTO consumes VALUES ('120','F20');
```

vi. Populating into gym

```
INSERT INTO gym VALUES ('34812','LA Fitness', 'Grand paire', '2019/12/18');  
INSERT INTO gym VALUES ('20318','Anytime Fitness', 'Irving', '2020/01/19');  
INSERT INTO gym VALUES ('39028','Fitness Connection', 'Forthworth', '2020/02/13');  
INSERT INTO gym VALUES ('28730','24 Hour Fitness', 'Dallas', '2019/09/18');  
INSERT INTO gym VALUES ('27364','Planet Fitness', 'Denton', '2020/05/25');  
INSERT INTO gym VALUES ('17283','Wellness Fitness', 'Hurst', '2020/05/26');  
INSERT INTO gym VALUES ('26354','Abbott Fitness', 'Hurst', '2020/03/01');  
INSERT INTO gym VALUES ('12093','LA Fitness', 'Grand paire', '2019/11/21');  
INSERT INTO gym VALUES ('28374','Fitness Connection', 'Dallas', '2019/09/15');  
INSERT INTO gym VALUES ('29837','LA Fitness', 'Grapevine', '2019/10/10');  
INSERT INTO gym VALUES ('26743','24 Hour Fitness', 'Eueless', '2019/12/17');  
INSERT INTO gym VALUES ('24534','Wellness Fitness', 'Bedfrod', '2020/05/18');  
INSERT INTO gym VALUES ('26712','Anytime Fitness', 'Grand paire', '2020/09/10');  
INSERT INTO gym VALUES ('19023','Fitness Connection', 'Dallas', '2020/07/20');  
INSERT INTO gym VALUES ('10293','24 Hour Fitness', 'Grand paire', '2020/08/17');  
INSERT INTO gym VALUES ('12283','Anytime Fitness', 'Grapevine', '2020/09/09');  
INSERT INTO gym VALUES ('28839','Fitness Connection', 'Dallas', '2020/03/14');  
INSERT INTO gym VALUES ('18297','Anytime Fitness', 'Irving', '2019/07/12');  
INSERT INTO gym VALUES ('21723','24 Hour Fitness', 'Forthworth', '2019/04/12');  
INSERT INTO gym VALUES ('12834','LA Fitness', 'Forthworth', '2019/09/12');
```

vii. Populating into exercise

```
INSERT INTO exercise VALUES ('W1','Cardio', '40 mins', 300 );
INSERT INTO exercise VALUES ('W2','Aerobic Exercise', '60 mins',220);
INSERT INTO exercise VALUES ('W3','Anaerobic Exercise', '45 mins', 320);
INSERT INTO exercise VALUES ('W4','Circuit', '50 mins', 320 );
INSERT INTO exercise VALUES ('W5','Compound Exercises', '1 hour 20 mins', 400);
INSERT INTO exercise VALUES ('W6','Cross-Training', '60 mins', 290);
INSERT INTO exercise VALUES ('W7','Interval Training', '1 hour 30 mins', 250);
INSERT INTO exercise VALUES ('W8',' Isometrics', '1 hour', 280);
INSERT INTO exercise VALUES ('W9','Plyometrics', '40 mins', 200);
INSERT INTO exercise VALUES ('W10','Steady-State Cardio', '35 mins', 110);
INSERT INTO exercise VALUES ('W11','Strength Training', '1 hour 30 mins', 270);
INSERT INTO exercise VALUES ('W12','Tabata', '30 mins', 120);
INSERT INTO exercise VALUES ('W13','HIIT', '50 mins', 300);
INSERT INTO exercise VALUES ('W14','Heart Rate Zones', '40 mins', 265);
INSERT INTO exercise VALUES ('W15','Functional Moves', '30 mins', 130);
INSERT INTO exercise VALUES ('W16','Stretching', '30 mins', 110);
INSERT INTO exercise VALUES ('W17','Foam Rolling', '60 mins', 275);
INSERT INTO exercise VALUES ('W18','Flexibility', '30 mins', 114);
INSERT INTO exercise VALUES ('W19','Balance Exercise', '60 mins', 250);
INSERT INTO exercise VALUES ('W20','Yoga', '45 mins',210);
```

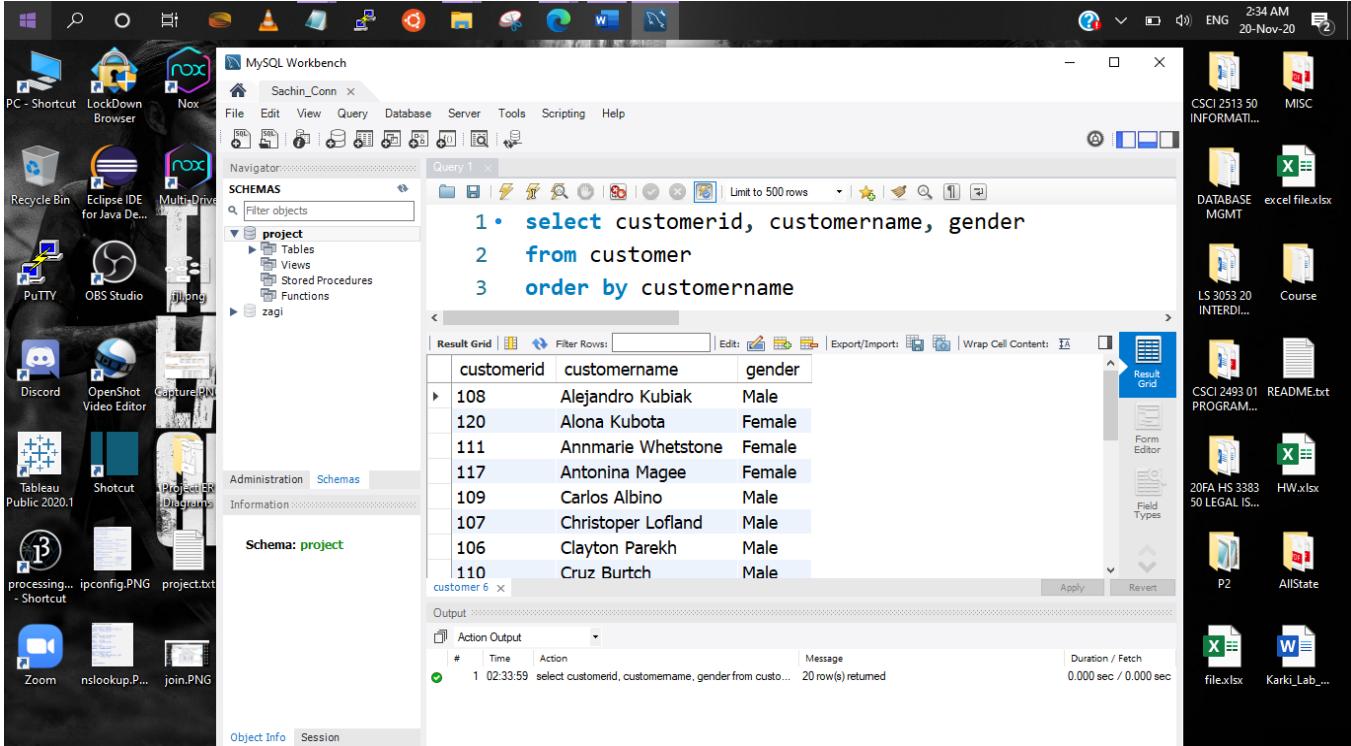
viii. Populating into has

```
INSERT INTO has VALUES ('34812','W1');
INSERT INTO has VALUES ('20318','W2');
INSERT INTO has VALUES ('39028','W3');
INSERT INTO has VALUES ('28730 ','W4');
INSERT INTO has VALUES ('27364','W5');
INSERT INTO has VALUES ('17283','W6');
INSERT INTO has VALUES ('26354','W7');
INSERT INTO has VALUES ('12093','W8');
INSERT INTO has VALUES ('28374','W9');
INSERT INTO has VALUES ('29837','W10');
INSERT INTO has VALUES ('26743','W11');
INSERT INTO has VALUES ('24534','W12');
INSERT INTO has VALUES ('26712','W13');
INSERT INTO has VALUES ('19023','W14');
INSERT INTO has VALUES ('10293','W15');
INSERT INTO has VALUES ('12283','W16');
INSERT INTO has VALUES ('28839','W17');
INSERT INTO has VALUES ('18297','W18');
INSERT INTO has VALUES ('21723','W19');
INSERT INTO has VALUES ('12834','W20');
```

c. **SQL Queries:** Pictured below are some of the queries along with their results.

2 Trivial Query - simple select with ordering

Query 1: Retrieve the entire contents of the customer and order by customername



The screenshot shows the MySQL Workbench interface with a query window titled "Query 1". The SQL code is:

```
1 • select customerid, customername, gender
2   from customer
3   order by customername
```

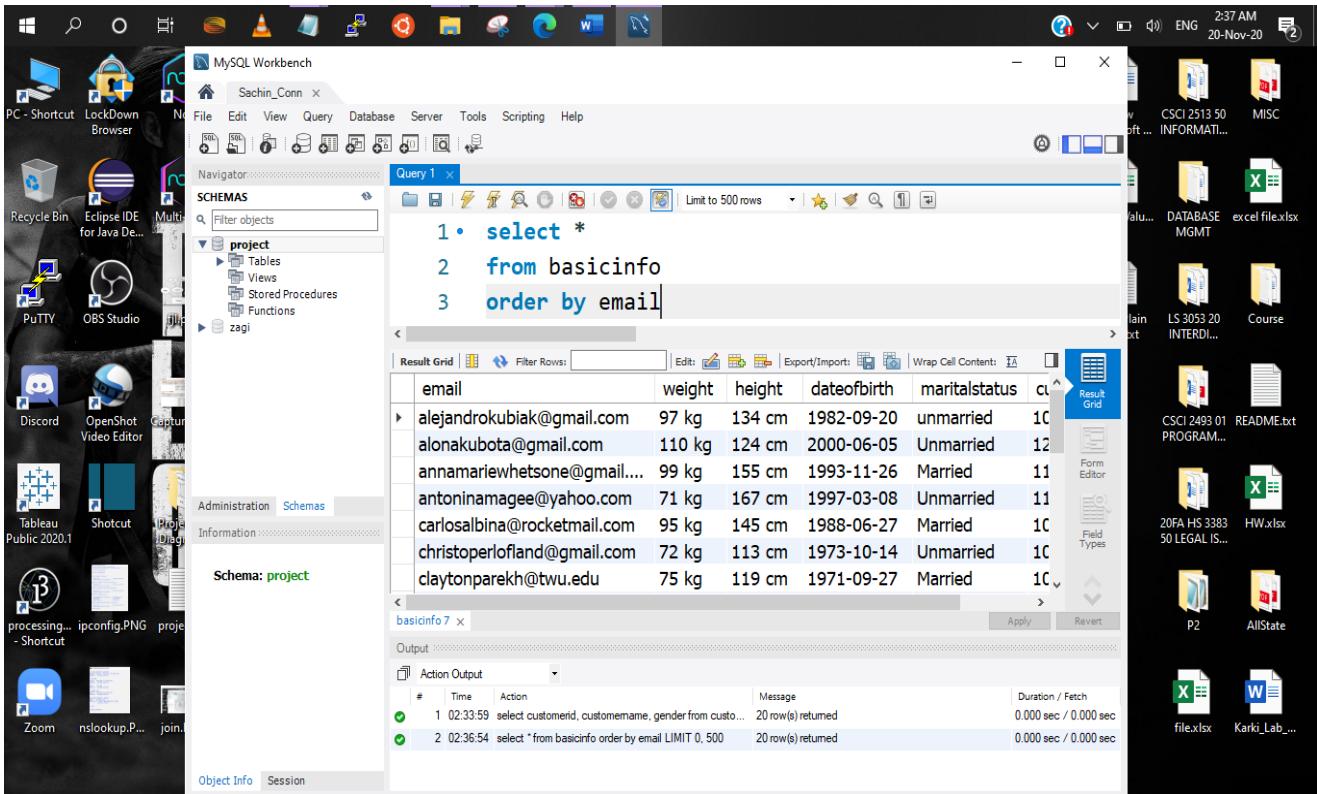
The result grid displays the following data:

| customerid | customername | gender |
|------------|---------------------|--------|
| 108 | Alejandro Kubik | Male |
| 120 | Alona Kubota | Female |
| 111 | Annamarie Whetstone | Female |
| 117 | Antonina Magee | Female |
| 109 | Carlos Albino | Male |
| 107 | Christoper Lofland | Male |
| 106 | Clayton Parekh | Male |
| 110 | Cruz Burch | Male |

The output pane shows two actions:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|---|--------------------|-----------------------|
| 1 | 02:33:59 | select customerid, customername, gender from custo... | 20 row(s) returned | 0.000 sec / 0.000 sec |
| 2 | 02:36:54 | select * from basicinfo order by email LIMIT 0, 500 | 20 row(s) returned | 0.000 sec / 0.000 sec |

Query 2. Retrieve the basic info of the customer and order by email



The screenshot shows the MySQL Workbench interface with a query window titled "Query 1". The SQL code is:

```
1 • select *
2   from basicinfo
3   order by email
```

The result grid displays the following data:

| email | weight | height | dateofbirth | maritalstatus | city |
|------------------------------|--------|--------|-------------|---------------|------|
| alejandrokubik@gmail.com | 97 kg | 134 cm | 1982-09-20 | unmarried | 10 |
| alonakubota@gmail.com | 110 kg | 124 cm | 2000-06-05 | Unmarried | 12 |
| annamariewhetstone@gmail.com | 99 kg | 155 cm | 1993-11-26 | Married | 11 |
| antoninamagee@yahoo.com | 71 kg | 167 cm | 1997-03-08 | Unmarried | 11 |
| carlosalbina@rocketmail.com | 95 kg | 145 cm | 1988-06-27 | Married | 10 |
| christoperlofland@gmail.com | 72 kg | 113 cm | 1973-10-14 | Unmarried | 10 |
| claytonparekh@twu.edu | 75 kg | 119 cm | 1971-09-27 | Married | 10 |

The output pane shows two actions:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|---|--------------------|-----------------------|
| 1 | 02:33:59 | select customerid, customername, gender from custo... | 20 row(s) returned | 0.000 sec / 0.000 sec |
| 2 | 02:36:54 | select * from basicinfo order by email LIMIT 0, 500 | 20 row(s) returned | 0.000 sec / 0.000 sec |

4 medium difficulty queries. Queries that use composite condition for selection, computations, aggregate function and grouping.

Query 3: For each user retrieve customerid, customername and amount of food while grouping the results by customerid and amount.

The screenshot shows the MySQL Workbench interface. In the central pane, a query window titled 'Query 1' contains the following SQL code:

```
1 • Select distinct c.customerid, c.customername, f.amount
2   from customer c, food f, consumes con
3   where c.customerid=con.customerid and con.foodid=f.foodid
4   group by c.customerid, f.amount;
```

The results grid displays the following data:

| customerid | customername | amount |
|------------|-----------------|--------|
| 101 | Ted Marquis | 1809 g |
| 102 | Sandy Yule | 1720 g |
| 103 | Rosendo Hefner | 1600 g |
| 104 | Quentin Tibbits | 2000 g |
| 105 | Jonathan Thiem | 1700 g |
| 106 | Clayton Donalds | 1600 g |

The status bar at the bottom indicates the session was active from 12:48 AM on 21-Nov-20.

Query 4: Retrieve number of available customers as ‘total number of customers’

The screenshot shows the MySQL Workbench interface. In the central pane, a query window titled 'Query 1' contains the following SQL code:

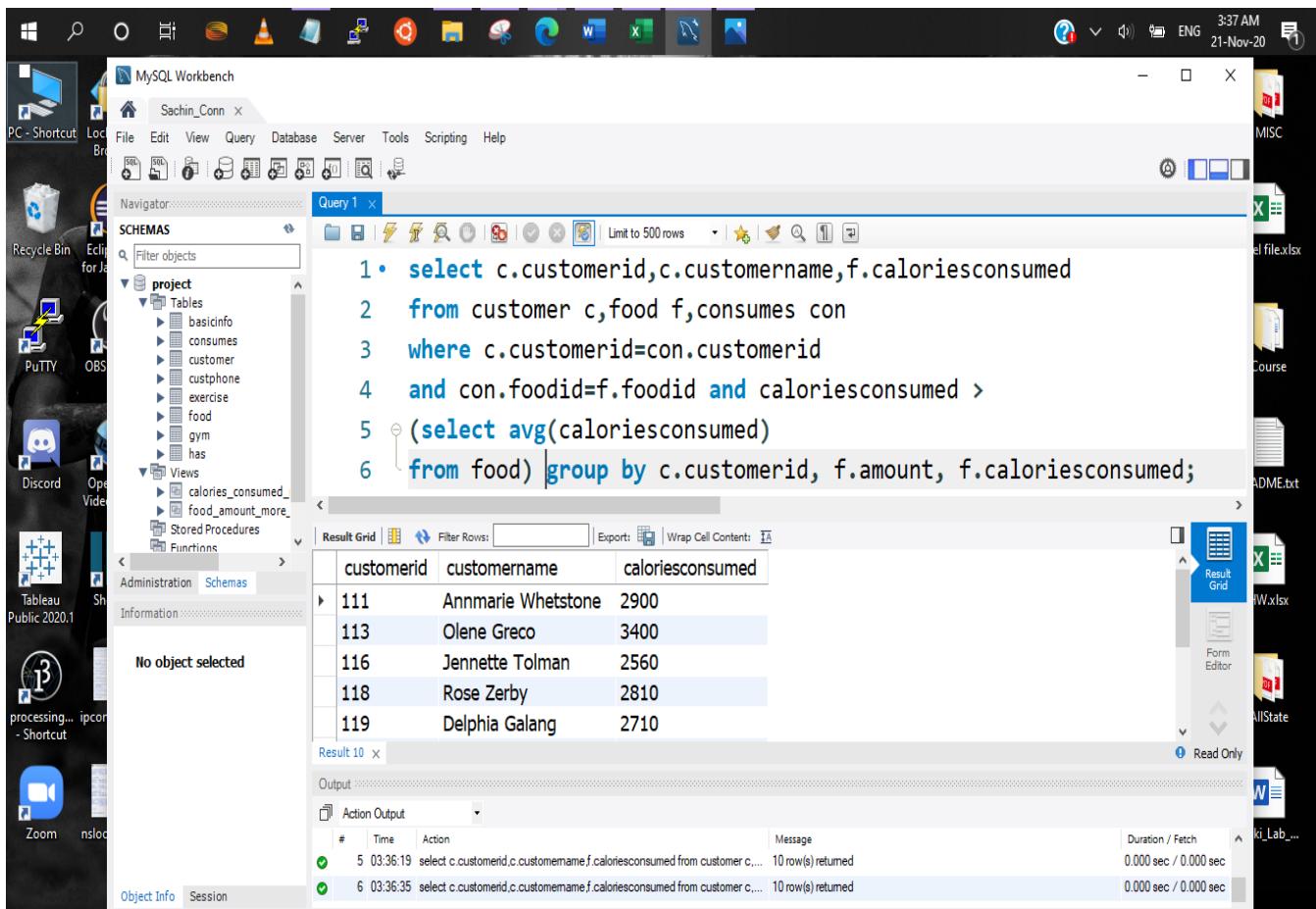
```
1 • select count(*) as 'total number of customers'
2   from customer
```

The results grid displays the following data:

| total number of customers |
|---------------------------|
| 20 |

The status bar at the bottom indicates the session was active from 3:39 AM on 21-Nov-20.

Query 5: Retrieve customerid, customername and average caloriesconsumed of each customer. Also, group the table by customerid, amount and caloriesconsumed



The screenshot shows the MySQL Workbench interface with a query window titled 'Query 1'. The code entered is:

```

1 • select c.customerid,c.customername,f.caloriesconsumed
2   from customer c,food f,consumes con
3   where c.customerid=con.customerid
4   and con.foodid=f.foodid and caloriesconsumed >
5   (select avg(caloriesconsumed)
6   from food) group by c.customerid, f.amount, f.caloriesconsumed;

```

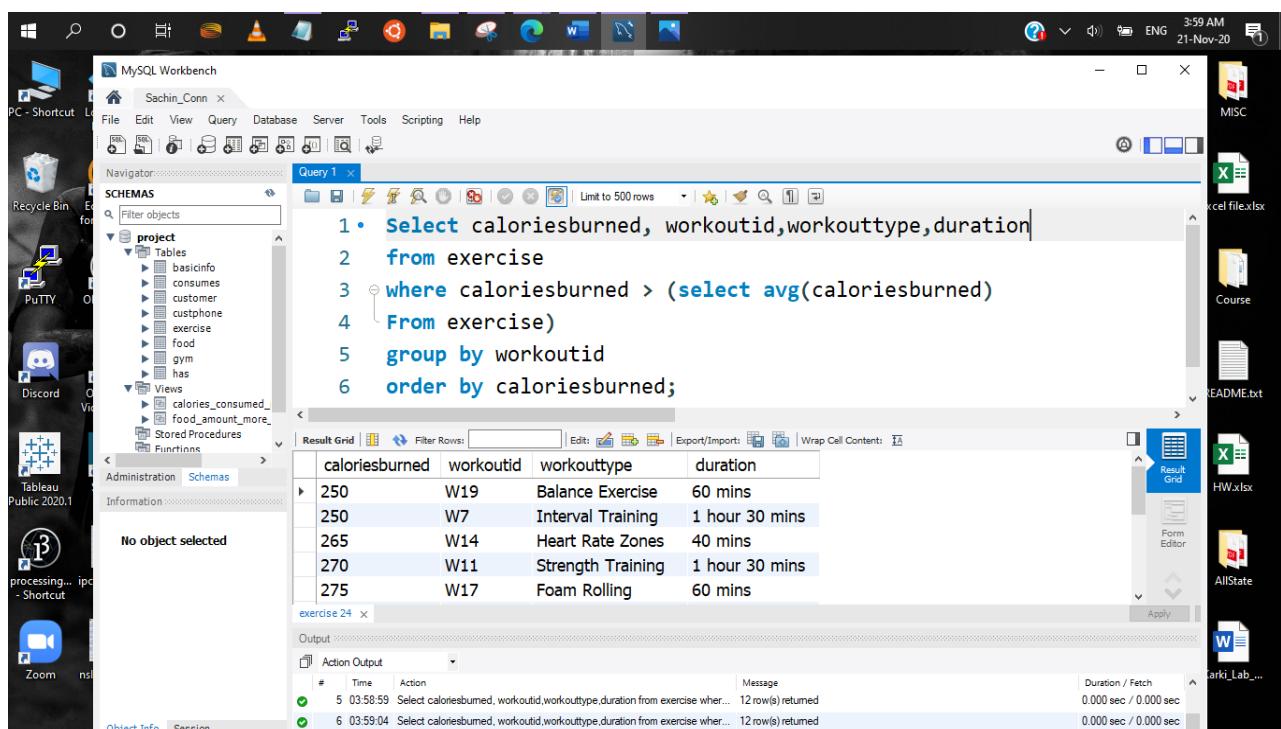
The result grid displays the following data:

| customerid | customername | caloriesconsumed |
|------------|--------------------|------------------|
| 111 | Annmarie Whetstone | 2900 |
| 113 | Olene Greco | 3400 |
| 116 | Jennette Tolman | 2560 |
| 118 | Rose Zarby | 2810 |
| 119 | Delphia Galang | 2710 |

The output pane shows two successful actions:

- Action 5: 03:36:19 select c.customerid,c.customername,f.caloriesconsumed from customer c... 10 row(s) returned Duration / Fetch: 0.000 sec / 0.000 sec
- Action 6: 03:36:35 select c.customerid,c.customername,f.caloriesconsumed from customer c... 10 row(s) returned Duration / Fetch: 0.000 sec / 0.000 sec

Query 6: Retrieve workoutid, workouttype and duration with caloriesburned on average. Also, group and sort the result by workoutid and caloriesburned respectively.



The screenshot shows the MySQL Workbench interface with a query window titled 'Query 1'. The code entered is:

```

1 • Select caloriesburned, workoutid,workouttype,duration
2   from exercise
3   where caloriesburned > (select avg(caloriesburned)
4   From exercise)
5   group by workoutid
6   order by caloriesburned;

```

The result grid displays the following data:

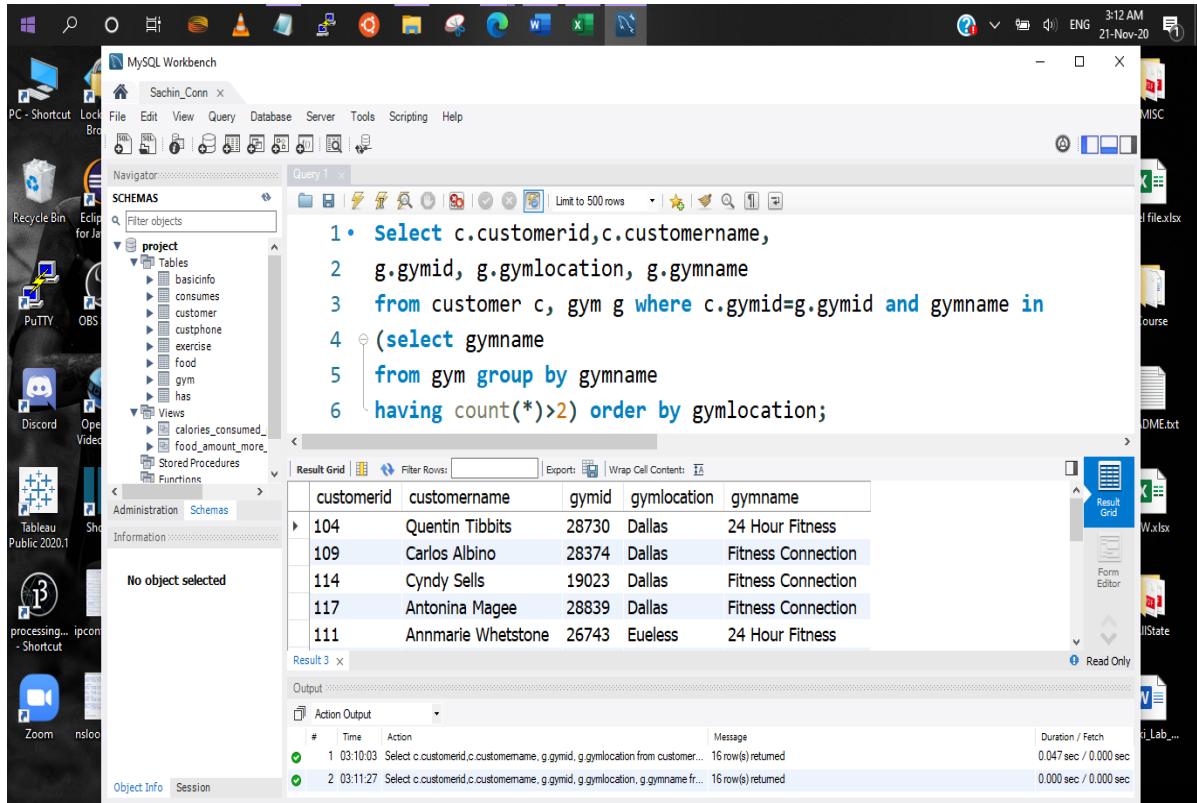
| caloriesburned | workoutid | workouttype | duration |
|----------------|-----------|-------------------|----------------|
| 250 | W19 | Balance Exercise | 60 mins |
| 250 | W7 | Interval Training | 1 hour 30 mins |
| 265 | W14 | Heart Rate Zones | 40 mins |
| 270 | W11 | Strength Training | 1 hour 30 mins |
| 275 | W17 | Foam Rolling | 60 mins |

The output pane shows two successful actions:

- Action 5: 03:58:59 Select caloriesburned, workoutid,workouttype,duration from exercise where... 12 row(s) returned Duration / Fetch: 0.000 sec / 0.000 sec
- Action 6: 03:59:04 Select caloriesburned, workoutid,workouttype,duration from exercise where... 12 row(s) returned Duration / Fetch: 0.000 sec / 0.000 sec

2 query that uses subquery.

Query 7: For each customername, retrieve customerid, gymid, gymlocation where number of gymlocation is more than 2. Also group the results by gymname



The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
1 • Select c.customerid, c.customername,
2     g.gymid, g.gymlocation, g.gymname
3     from customer c, gym g where c.gymid=g.gymid and gymname in
4     (select gymname
5      from gym group by gymname
6      having count(*)>2) order by gymlocation;
```

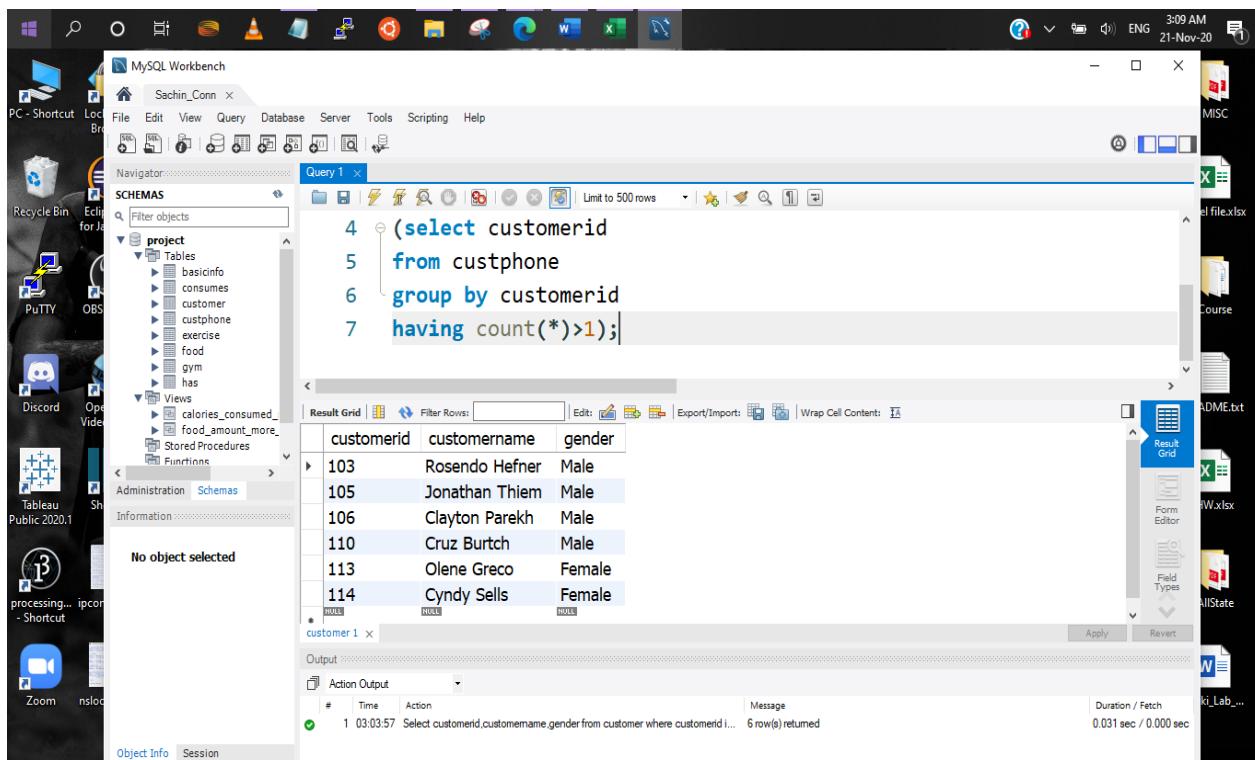
The result grid displays the following data:

| customerid | customername | gymid | gymlocation | gymname |
|------------|---------------------|-------|-------------|--------------------|
| 104 | Quentin Tibbits | 28730 | Dallas | 24 Hour Fitness |
| 109 | Carlos Albino | 28374 | Dallas | Fitness Connection |
| 114 | Cyndy Sells | 19023 | Dallas | Fitness Connection |
| 117 | Antonina Magee | 28839 | Dallas | Fitness Connection |
| 111 | Annamarie Whetstone | 26743 | Eueless | 24 Hour Fitness |

The output pane shows the execution log:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|--|---------|-----------------------|
| 1 | 03:10:03 | Select c.customerid,c.customername, g.gymid, g.gymlocation from customer... 16 row(s) returned | | 0.047 sec / 0.000 sec |
| 2 | 03:11:27 | Select c.customerid,c.customername, g.gymid, g.gymlocation, g.gymname fr... 16 row(s) returned | | 0.000 sec / 0.000 sec |

Query 8: Retrieve customerid, customername of users having more than one phone number



The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
4 • (select customerid
5     from custphone
6     group by customerid
7     having count(*)>1);
```

The result grid displays the following data:

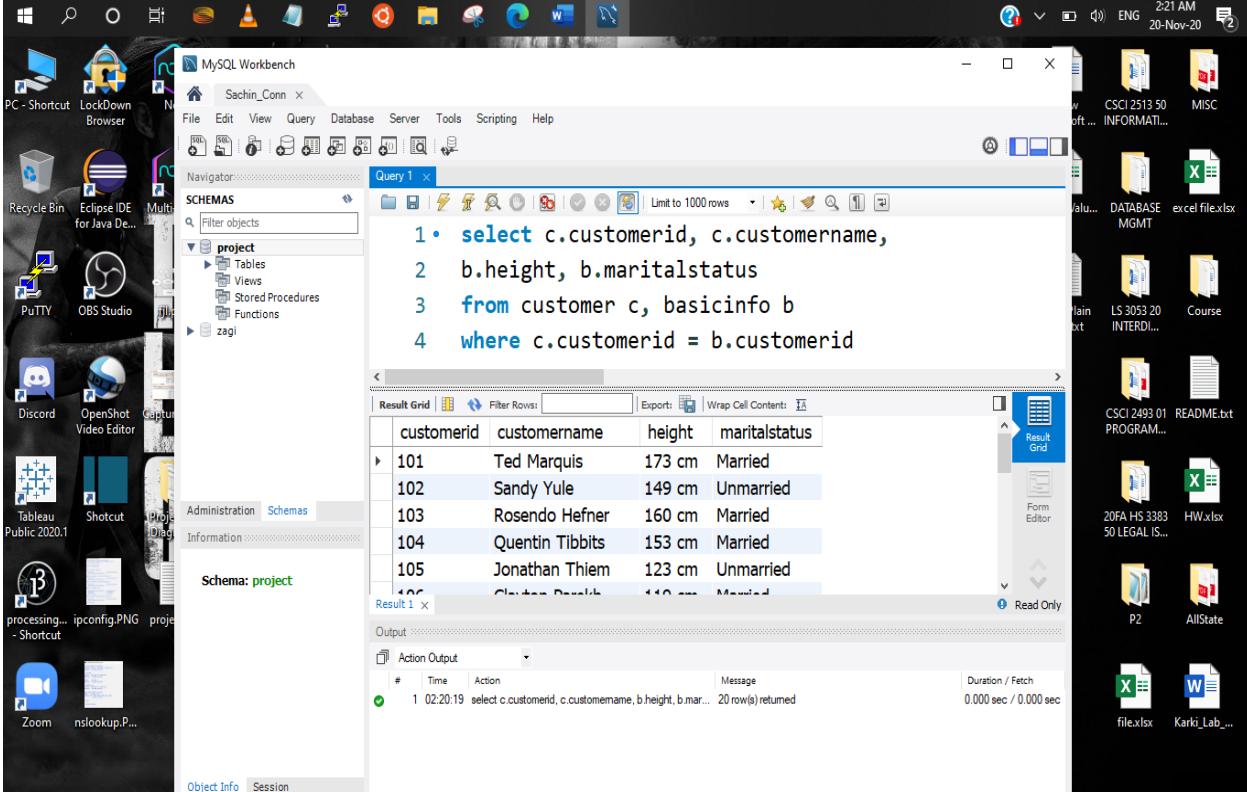
| customerid | customername | gender |
|------------|----------------|--------|
| 103 | Rosendo Hefner | Male |
| 105 | Jonathan Thiem | Male |
| 106 | Clayton Parekh | Male |
| 110 | Cruz Burch | Male |
| 113 | Olene Greco | Female |
| 114 | Cyndy Sells | Female |

The output pane shows the execution log:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|---|---------|-----------------------|
| 1 | 03:03:57 | Select customerid,customername,gender from customer where customerid i... 6 row(s) returned | | 0.031 sec / 0.000 sec |

2 queries that uses join (1 inner join, 1 left or right outer join).

Query 9: Using self-join, retrieve customer id, customer name, height and their marital status



The screenshot shows the MySQL Workbench interface with a query window titled "Query 1". The query is:

```
1 • select c.customerid, c.customername,
2      b.height, b.maritalstatus
3   from customer c, basicinfo b
4  where c.customerid = b.customerid
```

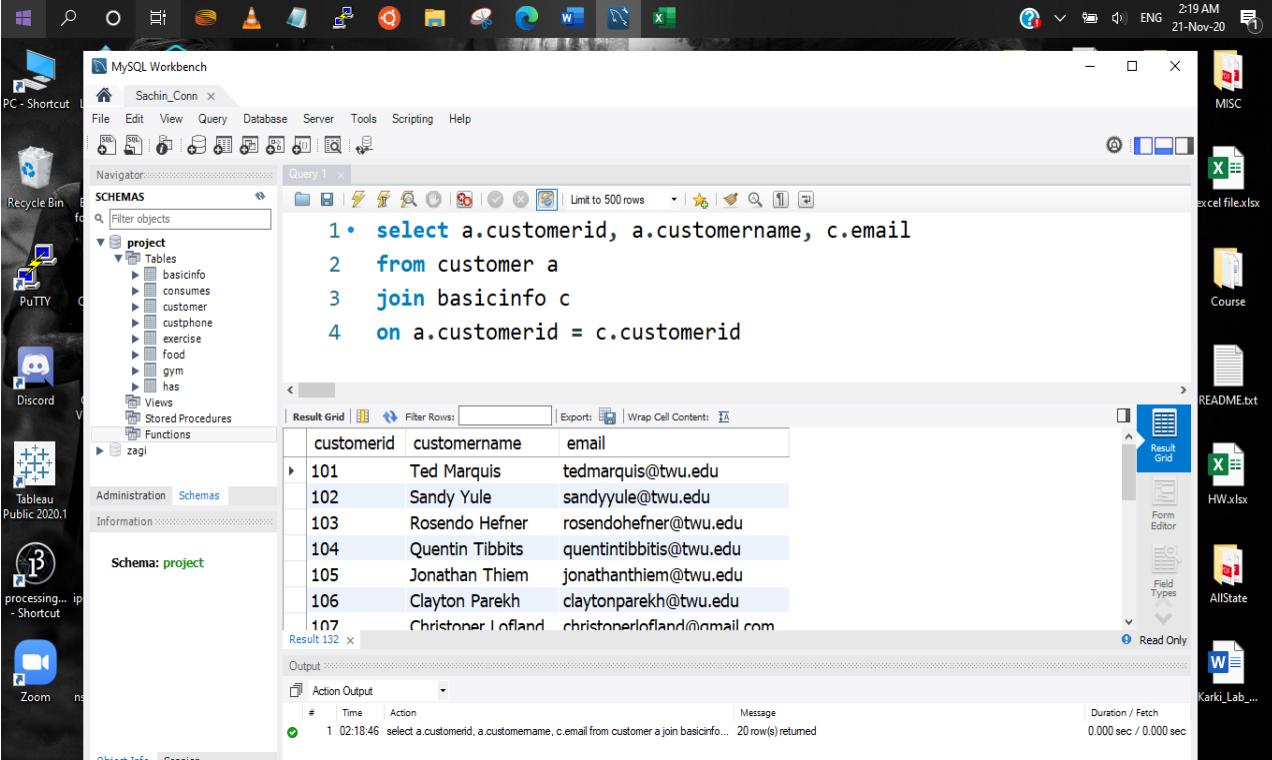
The result grid displays the following data:

| customerid | customername | height | maritalstatus |
|------------|-----------------|--------|---------------|
| 101 | Ted Marquis | 173 cm | Married |
| 102 | Sandy Yule | 149 cm | Unmarried |
| 103 | Rosendo Hefner | 160 cm | Married |
| 104 | Quentin Tibbits | 153 cm | Married |
| 105 | Jonathan Thiem | 123 cm | Unmarried |
| 106 | Clayton Parekh | 140 cm | Married |

The output pane shows the execution details:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|---|--------------------|-----------------------|
| 1 | 02:20:19 | select c.customerid, c.customername, b.height, b.mar... | 20 row(s) returned | 0.000 sec / 0.000 sec |

Query 10: Using join, for each customer retrieve customerid, customername and email



The screenshot shows the MySQL Workbench interface with a query window titled "Query 1". The query is:

```
1 • select a.customerid, a.customername, c.email
2   from customer a
3   join basicinfo c
4  on a.customerid = c.customerid
```

The result grid displays the following data:

| customerid | customername | email |
|------------|-------------------|----------------------------|
| 101 | Ted Marquis | tedmarquis@twu.edu |
| 102 | Sandy Yule | sandyyule@twu.edu |
| 103 | Rosendo Hefner | rosendohefner@twu.edu |
| 104 | Quentin Tibbits | quentintibbits@twu.edu |
| 105 | Jonathan Thiem | jonathanthiem@twu.edu |
| 106 | Clayton Parekh | claytonparekh@twu.edu |
| 107 | Christoner Ioland | christonerioland@gmail.com |

The output pane shows the execution details:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|--|--------------------|-----------------------|
| 1 | 02:18:46 | select a.customerid, a.customername, c.email from customer a join basicinfo... | 20 row(s) returned | 0.000 sec / 0.000 sec |

Query 11: Using inner join, for each customer retrieve customerid, customername and dateofbirth

The screenshot shows the MySQL Workbench interface with a query window titled 'Query 1'. The code is:

```
1 • Select c.customerid, c.customername, b.dateofbirth
2   from customer c
3   Inner Join basicinfo b
4   on c.customerid=b.customerid;
```

The result grid displays the following data:

| customerid | customername | dateofbirth |
|------------|--------------------|-------------|
| 108 | Alejandro Kubiak | 1982-09-20 |
| 120 | Alona Kubota | 2000-06-05 |
| 111 | Annmarie Whetstone | 1993-11-26 |
| 117 | Antonina Magee | 1997-03-08 |
| 109 | Carlos Albino | 1988-06-27 |
| 107 | Christoper Lofland | 1973-10-14 |
| 106 | Clayton Parekh | 1971-09-27 |

The output pane shows two log entries:

- 1 02:18:46 select c.customerid, c.customername, c.email from customer c join basicinfo ... 20 row(s) returned Duration / Fetch 0.000 sec / 0.000 sec
- 2 02:22:15 Select c.customerid, c.customername, b.dateofbirth from customer c. Inner ... 20 row(s) returned 0.046 sec / 0.000 sec

Query 12: Using right outer join, for each customer retrieve customerid, customername, weight and height

The screenshot shows the MySQL Workbench interface with a query window titled 'Query 1'. The code is:

```
1 • Select c.customerid, c.customername, b.weight, b.height
2   from customer c
3   Right Outer Join basicinfo b
4   on c.customerid=b.customerid;
```

The result grid displays the following data:

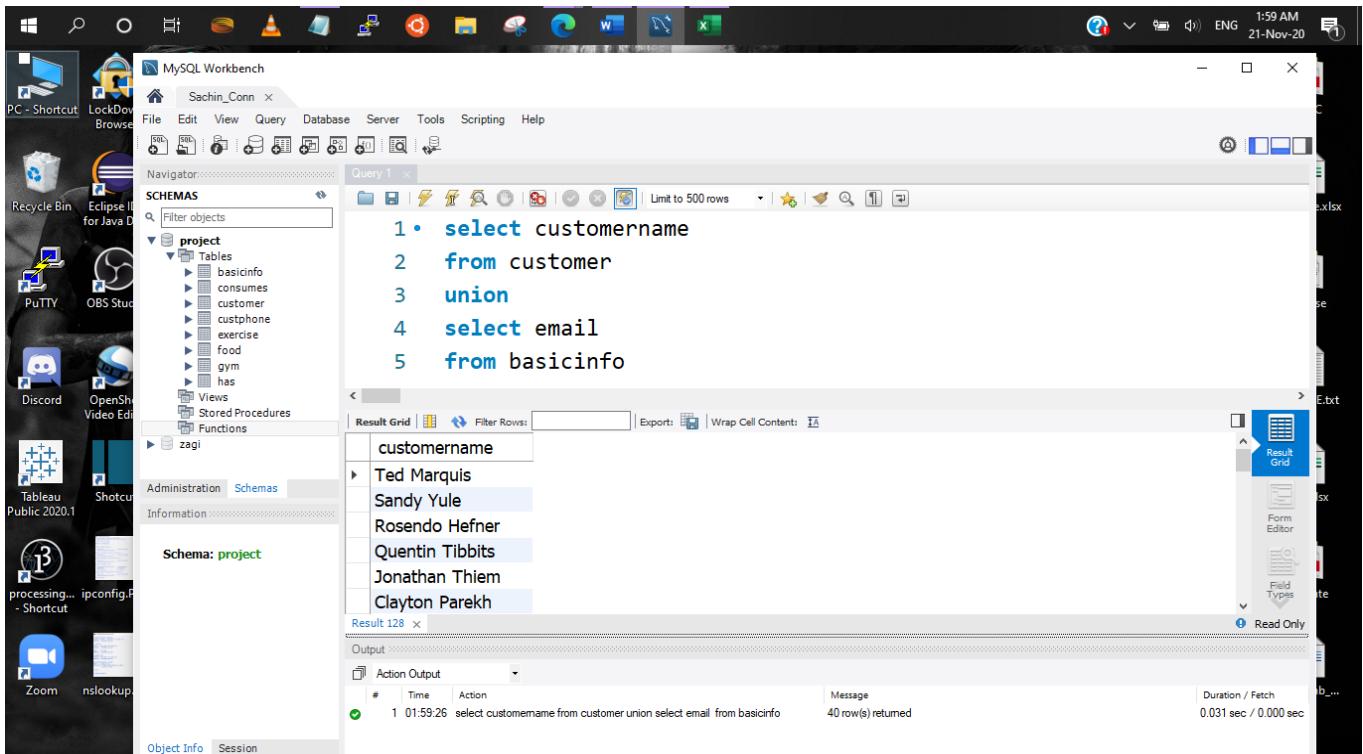
| customerid | customername | weight | height |
|------------|--------------------|--------|--------|
| 108 | Alejandro Kubiak | 97 kg | 134 cm |
| 120 | Alona Kubota | 110 kg | 124 cm |
| 111 | Annmarie Whetstone | 99 kg | 155 cm |
| 117 | Antonina Magee | 71 kg | 167 cm |
| 109 | Carlos Albino | 95 kg | 145 cm |
| 107 | Christoper Lofland | 72 kg | 113 cm |
| 106 | Clayton Parekh | 75 kg | 119 cm |

The output pane shows one log entry:

- 4 02:24:38 Select c.customerid, c.customername, b.weight from customer c Right Ou... 20 row(s) returned 0.000 sec / 0.000 sec

2 query that uses union.

Query 13: For each customer, retrieve customer name and email using union



The screenshot shows the MySQL Workbench interface with a query window titled 'Query 1'. The query is:`1 • select customername
2 from customer
3 union
4 select email
5 from basicinfo`

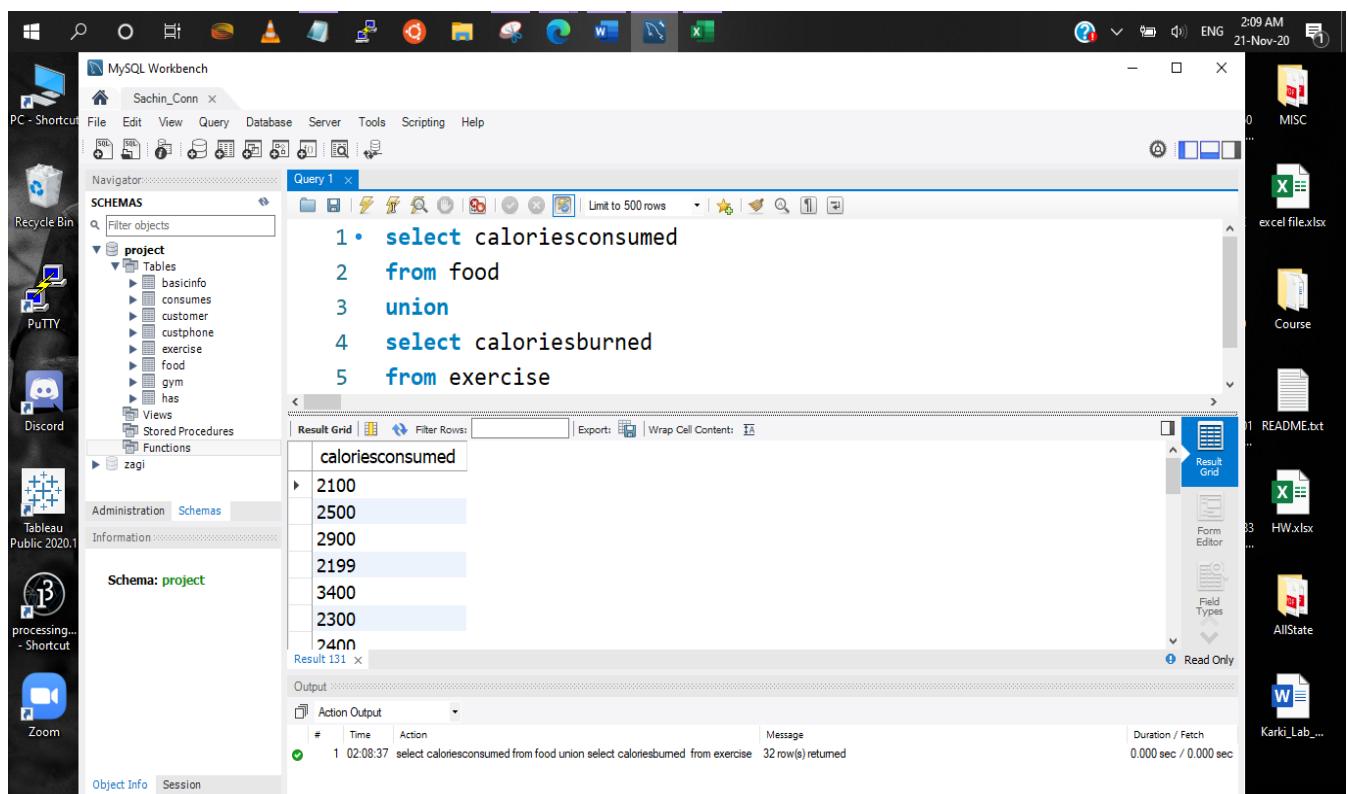
The result grid displays the following data:

| customername |
|-----------------|
| Ted Marquis |
| Sandy Yule |
| Rosendo Hefner |
| Quentin Tibbits |
| Jonathan Thiem |
| Clayton Parekh |

The output pane shows the execution details:

| Action | Time | Message | Duration / Fetch |
|--------|----------|---|-----------------------|
| 1 | 01:59:26 | select customername from customer union select email from basicinfo | 0.031 sec / 0.000 sec |

Query 14: Retrieve calories consumed and calories burned using union



The screenshot shows the MySQL Workbench interface with a query window titled 'Query 1'. The query is:`1 • select caloriesconsumed
2 from food
3 union
4 select caloriesburned
5 from exercise`

The result grid displays the following data:

| caloriesconsumed |
|------------------|
| 2100 |
| 2500 |
| 2900 |
| 2199 |
| 3400 |
| 2300 |
| 2400 |

The output pane shows the execution details:

| Action | Time | Message | Duration / Fetch |
|--------|----------|---|-----------------------|
| 1 | 02:08:37 | select caloriesconsumed from food union select caloriesburned from exercise | 0.000 sec / 0.000 sec |

2 view (query must use join – hint: you can use one of the queries from the previous question).

Query 15: using create view, retrieve customerid, customername, gender of customer whose calories consumption is more than 2500.

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' section, there is a 'project' schema containing several tables like basicinfo, consumes, customer, custphone, exercise, food, gym, and has. A 'Views' folder is also present. In the main 'Query 1' editor window, a SQL script is being typed:

```
1 • Create view Calories_consumed_more_than_2500_calories As
2 Select c.customerid,c.customername,c.gender,f.caloriesconsumed
3 from customer c,food f,consumes con
4 where c.customerid=con.customerid and con.foodid=f.foodid
5 and caloriesconsumed in (
6 select caloriesconsumed
7 from food
8 group by foodid
9 having caloriesconsumed>2500);
```

The status bar at the bottom indicates the action was completed successfully: '1 02:36:53 Create view Calories_consumed_more_than_2500_calories As Select c.cust... 0 row(s) affected Duration / Fetch 0.218 sec'.

Results:

The screenshot shows the MySQL Workbench interface again. The 'Query 1' editor now contains a simple SELECT query:

```
1 • Select *
2 from Calories_consumed_more_than_2500_calories;
```

The results are displayed in a 'Result Grid' table:

| customerid | customername | gender | caloriesconsumed |
|------------|--------------------|--------|------------------|
| 105 | Jonathan Thiem | Male | 2600 |
| 106 | Clayton Parekh | Male | 2700 |
| 107 | Christoper Lofland | Male | 3200 |
| 109 | Carlos Albino | Male | 2900 |
| 111 | Annmarie Whetstone | Female | 2900 |
| 113 | Olene Greco | Female | 3400 |
| 116 | Tennette Tolman | Female | 2560 |

The status bar at the bottom indicates the query was executed successfully: '1 02:38:09 Select * from Calories_consumed_more_than_2500_calories LIMIT 0, 500 10 row(s) returned Duration / Fetch 0.125 sec / 0.000 sec'.

Query 16: Using create view, retrieve customerid, customername, gender of a customer whose amount of food intake is more than 1500 g.

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'project' schema, a new view named 'Food_amount_more_than_1500g' is being created. The query editor contains the following SQL code:

```
1 • Create view Food_amount_more_than_1500g As
2 Select c.customerid,c.customername,c.gender,f.amount
3 from customer c,food f,consumes con
4 where c.customerid=con.customerid and con.foodid=f.foodid
5 and amount in (
6 select amount
7 from food
8 group by foodid
9 having amount>1500);
```

The output pane shows the execution log:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|---|-------------------|------------------|
| 1 | 02:50:08 | Create view Food_amount_more_than_1500g As Select c.customerid,c.cus... | 0 row(s) affected | 0.219 sec |

Results:

The screenshot shows the MySQL Workbench interface with the results of the query. The query editor contains:

```
1 • Select *
2 from Food_amount_more_than_1500g;
```

The result grid displays the following data:

| customerid | customername | gender | amount |
|------------|-------------------|--------|--------|
| 101 | Ted Marquis | Male | 1809 |
| 102 | Sandy Yule | Male | 1720 |
| 103 | Rosendo Hefner | Male | 1600 |
| 104 | Quentin Tibbits | Male | 2000 |
| 105 | Jonathan Thiem | Male | 1700 |
| 106 | Clayton Parekh | Male | 1660 |
| 107 | Christoner Ioland | Male | 1570 |

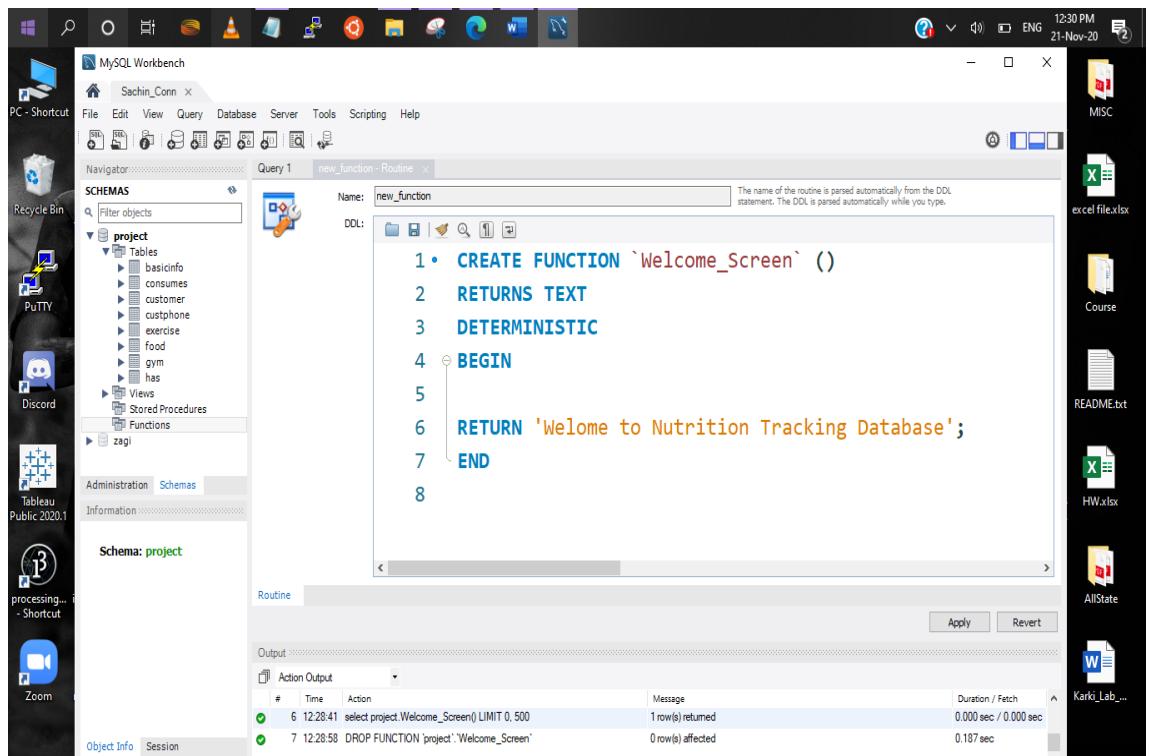
The output pane shows the execution log:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|---|--------------------|-----------------------|
| 1 | 02:50:08 | Create view Food_amount_more_than_1500g As Select c.customerid,c.cus... | 0 row(s) affected | 0.219 sec |
| 2 | 02:51:00 | Select * from Food_amount_more_than_1500g LIMIT 0, 500 | 14 row(s) returned | 0.000 sec / 0.000 sec |

2 custom stored function.

Query 17: Create a custom stored function to display name of the database system as welcome screen

Query Code:



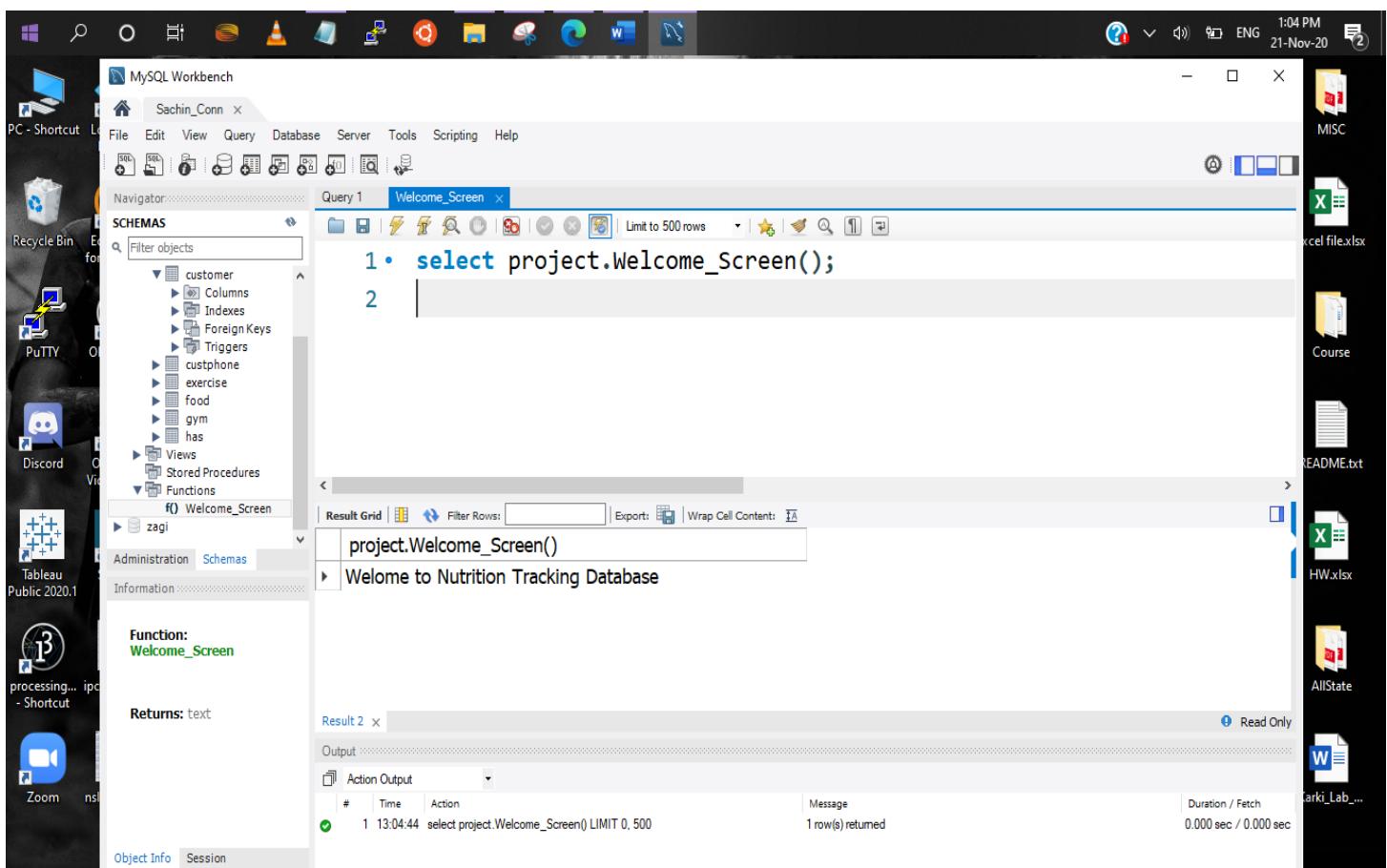
The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' section, the 'project' schema is selected. In the main pane, a routine named 'new_function' is being created with the following DDL:

```
1 • CREATE FUNCTION `Welcome_Screen` ()  
2     RETURNS TEXT  
3     DETERMINISTIC  
4     BEGIN  
5  
6         RETURN 'Welcome to Nutrition Tracking Database';  
7     END  
8
```

The 'Output' pane at the bottom shows two log entries:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|--|-------------------|-----------------------|
| 6 | 12:28:41 | select project.Welcome_Screen() LIMIT 0, 500 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 7 | 12:28:58 | DROP FUNCTION project.'Welcome_Screen' | 0 row(s) affected | 0.187 sec |

Result:



The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' section, the 'project' schema is selected. In the main pane, a query is run:

```
1 • select project.Welcome_Screen();  
2
```

The results are displayed in the 'Result Grid' tab:

| project.Welcome_Screen() |
|--|
| Welcome to Nutrition Tracking Database |

The 'Object Info' pane on the left shows details for the 'Welcome_Screen' function:

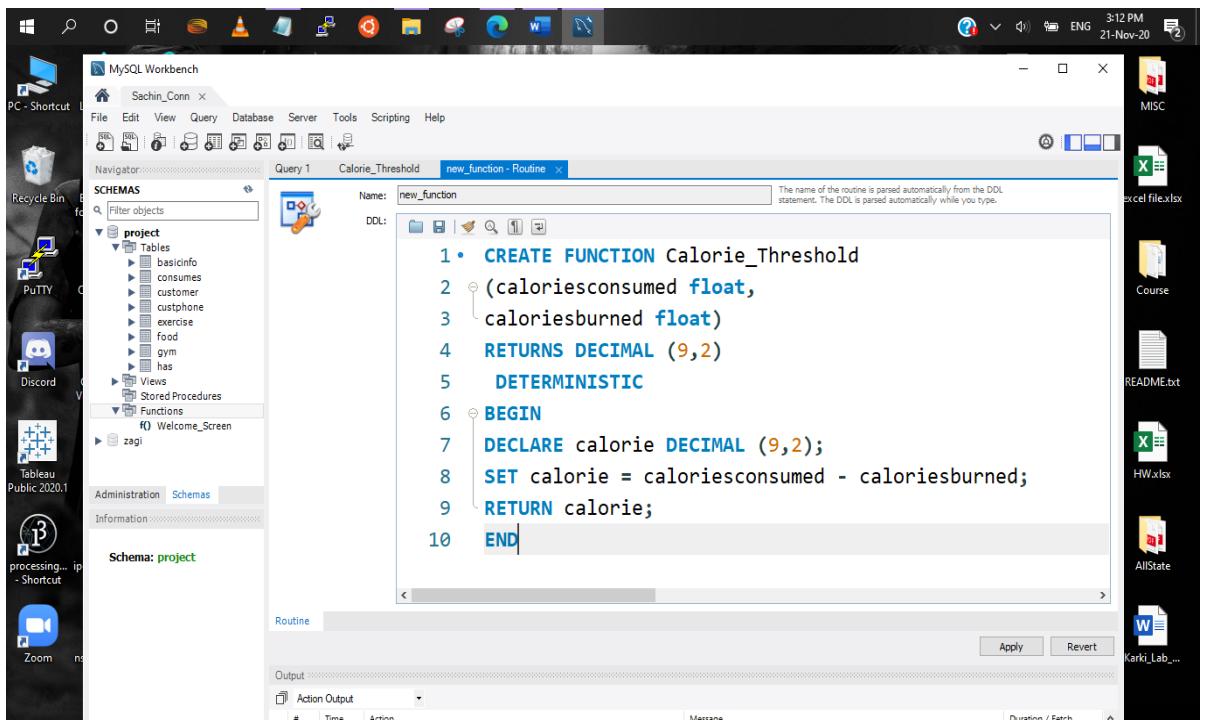
- Function:** Welcome_Screen
- Returns:** text

The 'Output' pane at the bottom shows one log entry:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|--|-------------------|-----------------------|
| 1 | 13:04:44 | select project.Welcome_Screen() LIMIT 0, 500 | 1 row(s) returned | 0.000 sec / 0.000 sec |

Query 18: Custom stored function to calculate threshold of calorie intake.

Query Code:

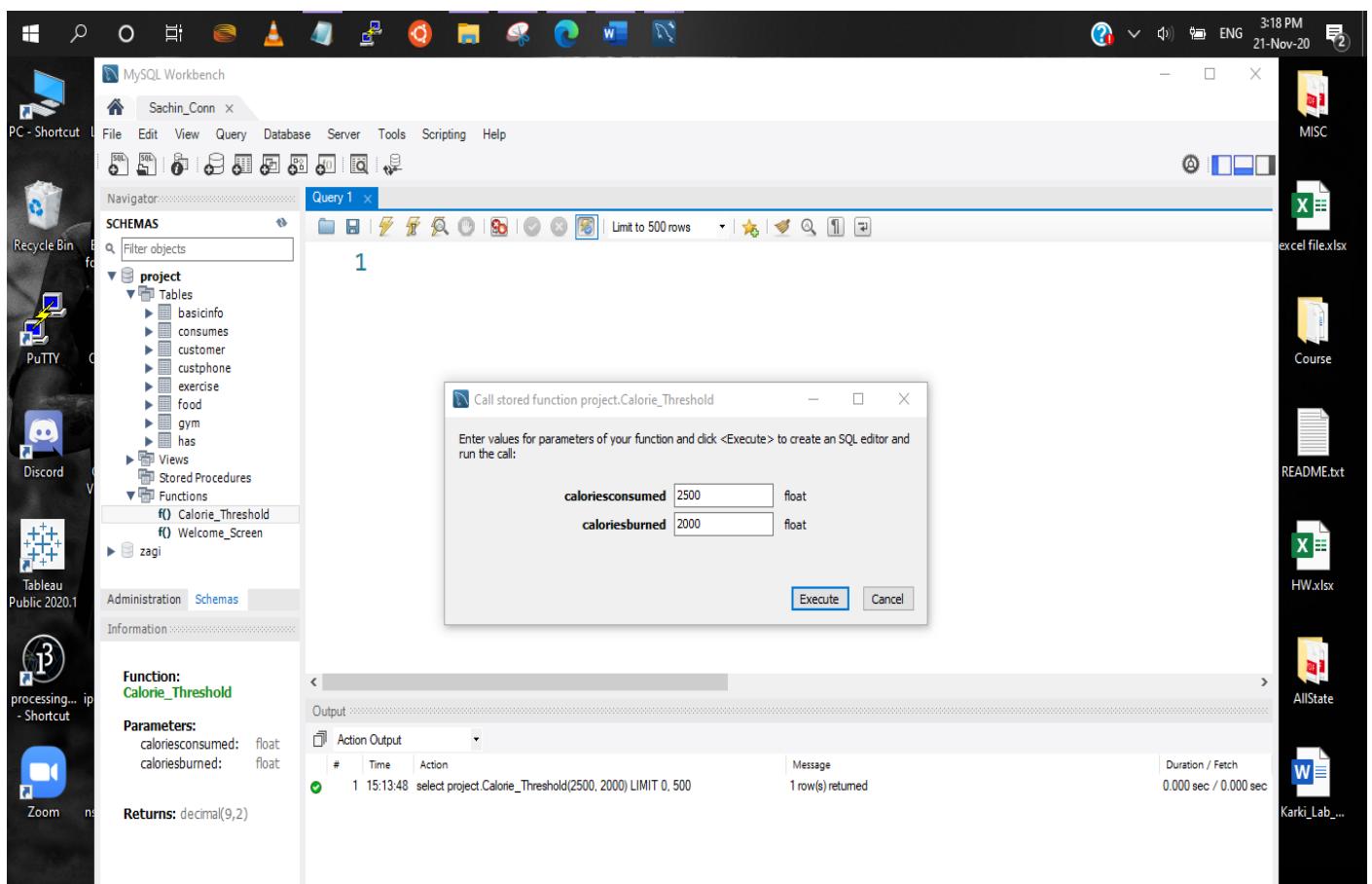


The screenshot shows the MySQL Workbench interface. In the center, the Query Editor window displays the following SQL code:

```
1 • CREATE FUNCTION Calorie_Threshold
2   (caloriesconsumed float,
3    caloriesburned float)
4   RETURNS DECIMAL (9,2)
5   DETERMINISTIC
6   BEGIN
7     DECLARE calorie DECIMAL (9,2);
8     SET calorie = caloriesconsumed - caloriesburned;
9     RETURN calorie;
10    END
```

The 'Name' field in the top right is set to 'new_function'. The 'Schema' dropdown shows 'project'. The 'Output' section at the bottom indicates 'Action Output'.

Query Prompt:



The screenshot shows the MySQL Workbench interface. The Query Editor window contains the number '1'. A call dialog box is open, prompting for values for the parameters of the function:

Call stored function project.Calorie_Threshold

Enter values for parameters of your function and click <Execute> to create an SQL editor and run the call:

| | | |
|------------------|------|-------|
| caloriesconsumed | 2500 | float |
| caloriesburned | 2000 | float |

Buttons for 'Execute' and 'Cancel' are visible at the bottom of the dialog.

On the left, the Navigator pane shows the 'project' schema with various tables and functions. The 'Function: Calorie_Threshold' section is expanded, showing 'Parameters' (caloriesconsumed: float, caloriesburned: float) and 'Returns: decimal(9,2)'.

The Output pane at the bottom shows the result of the query: '1 15:13:48 select project.Calorie_Threshold(2500, 2000) LIMIT 0, 500'.

Query Result:

The screenshot shows the MySQL Workbench interface with a connection named "Sachin_Conn". In the "Query 1" tab, the following SQL code is run:

```
1 • select project.Calorie_Threshold(2500, 2000);  
2
```

The results grid displays the output of the function call:

| |
|--|
| project.Calorie_Threshold(2500, 2000) |
| 500.00 |

In the bottom pane, the "Result 1" tab shows the execution details:

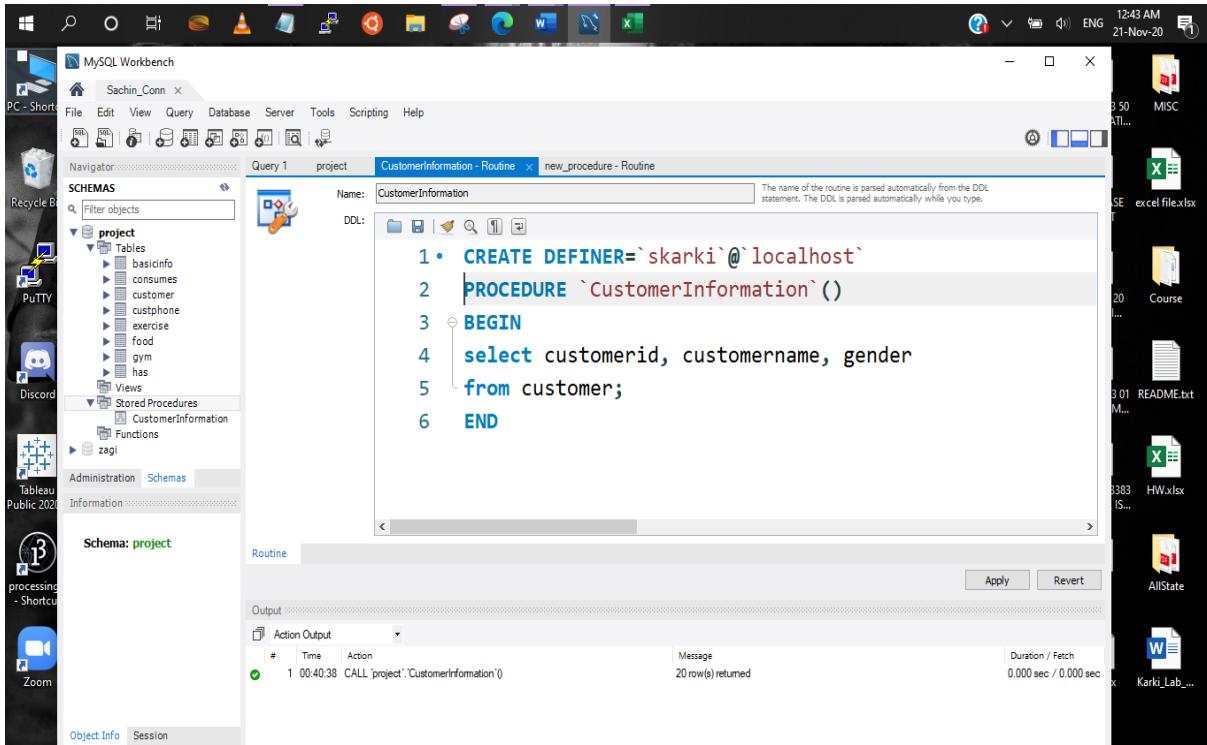
| # | Time | Action | Message | Duration / Fetch |
|---|----------|---|-------------------|-----------------------|
| 1 | 15:13:48 | select project.Calorie_Threshold(2500, 2000) LIMIT 0, 500 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 2 | 15:18:35 | select project.Calorie_Threshold(2500, 2000) LIMIT 0, 500 | 1 row(s) returned | 0.000 sec / 0.000 sec |

On the right side of the interface, there is a vertical sidebar with various file icons and names, including "MISC", "excel file.xlsx", "Course", "README.txt", "HW.xlsx", "AllState", and "Karki_Lab...".

2 custom stored procedure.

Query 19: Custom stored procedure to display entire records of customer information

Query Code:



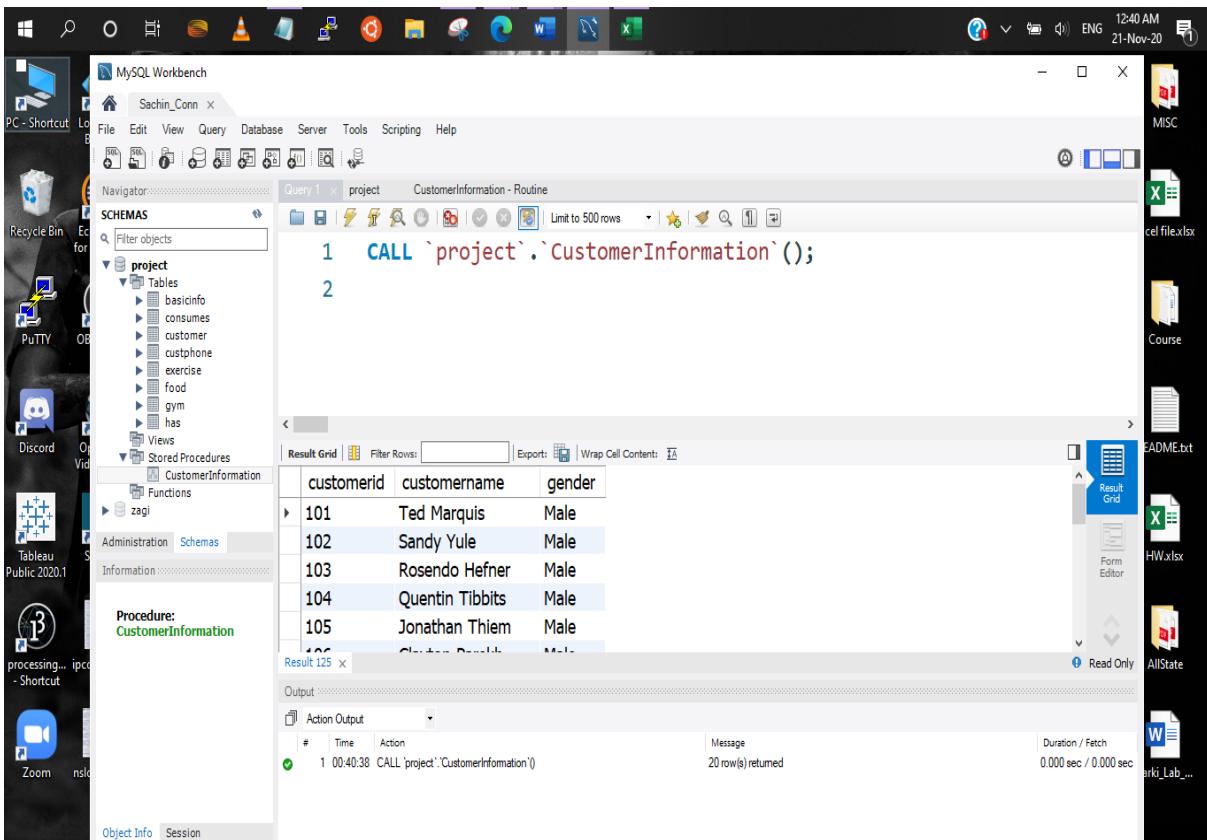
The screenshot shows the MySQL Workbench interface. In the center, the 'CustomerInformation - Routine' tab is active. The 'Name:' field contains 'CustomerInformation'. The 'DDL:' pane displays the following SQL code:

```
1 • CREATE DEFINER=`skarki`@`localhost` PROCEDURE `CustomerInformation`() BEGIN select customerid, customername, gender from customer; END
```

Below the DDL, the 'Output' pane shows the result of a call to the procedure:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|--|--------------------|-----------------------|
| 1 | 00:40:38 | CALL 'project'.'CustomerInformation'() | 20 row(s) returned | 0.000 sec / 0.000 sec |

Query Result:



The screenshot shows the MySQL Workbench interface with the 'CustomerInformation - Routine' tab active. The 'Procedure:' dropdown shows 'CustomerInformation'. The 'Result Grid' pane displays the results of the procedure execution:

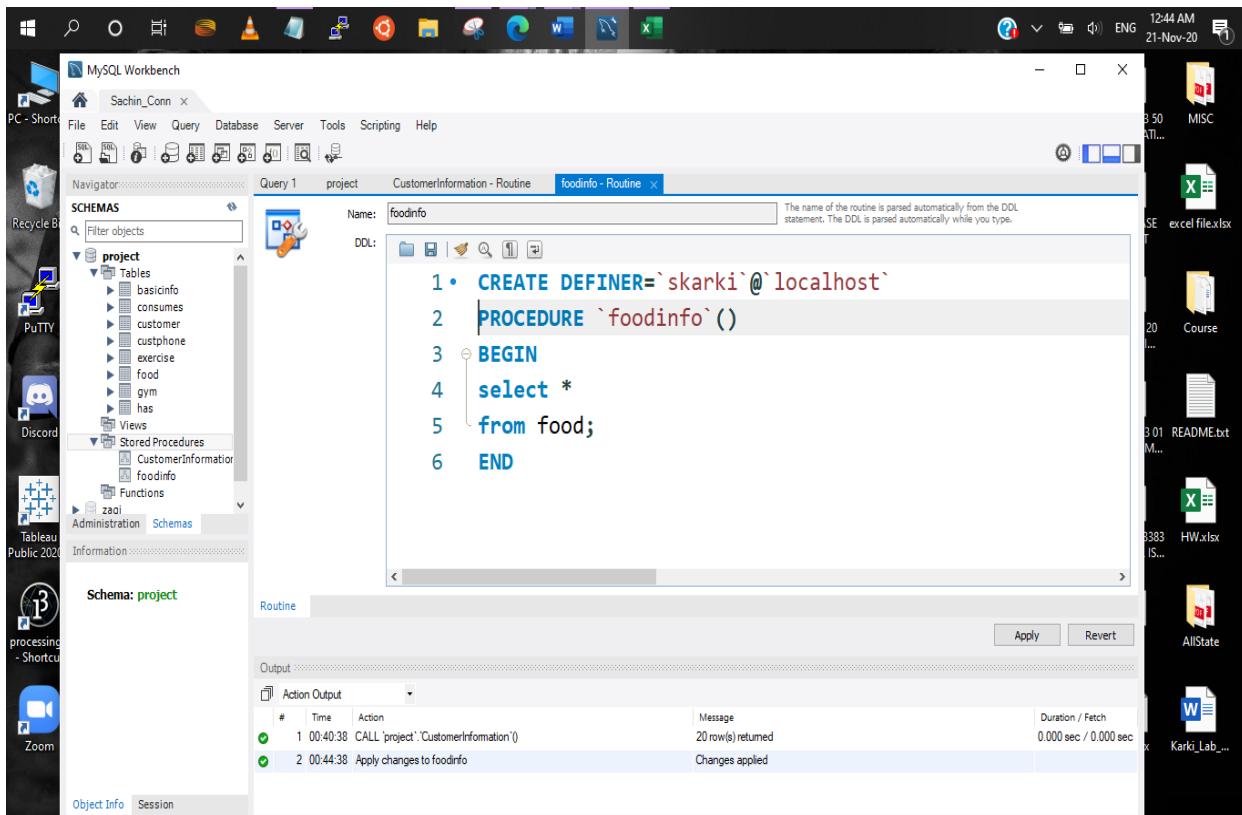
| customerid | customername | gender |
|------------|-----------------|--------|
| 101 | Ted Marquis | Male |
| 102 | Sandy Yule | Male |
| 103 | Rosendo Hefner | Male |
| 104 | Quentin Tibbits | Male |
| 105 | Jonathan Thiem | Male |

The 'Output' pane at the bottom shows the same log entry as the previous screenshot:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|--|--------------------|-----------------------|
| 1 | 00:40:38 | CALL 'project'.'CustomerInformation'() | 20 row(s) returned | 0.000 sec / 0.000 sec |

Query 20: Custom stored procedure to display entire contents of food.

Query code:



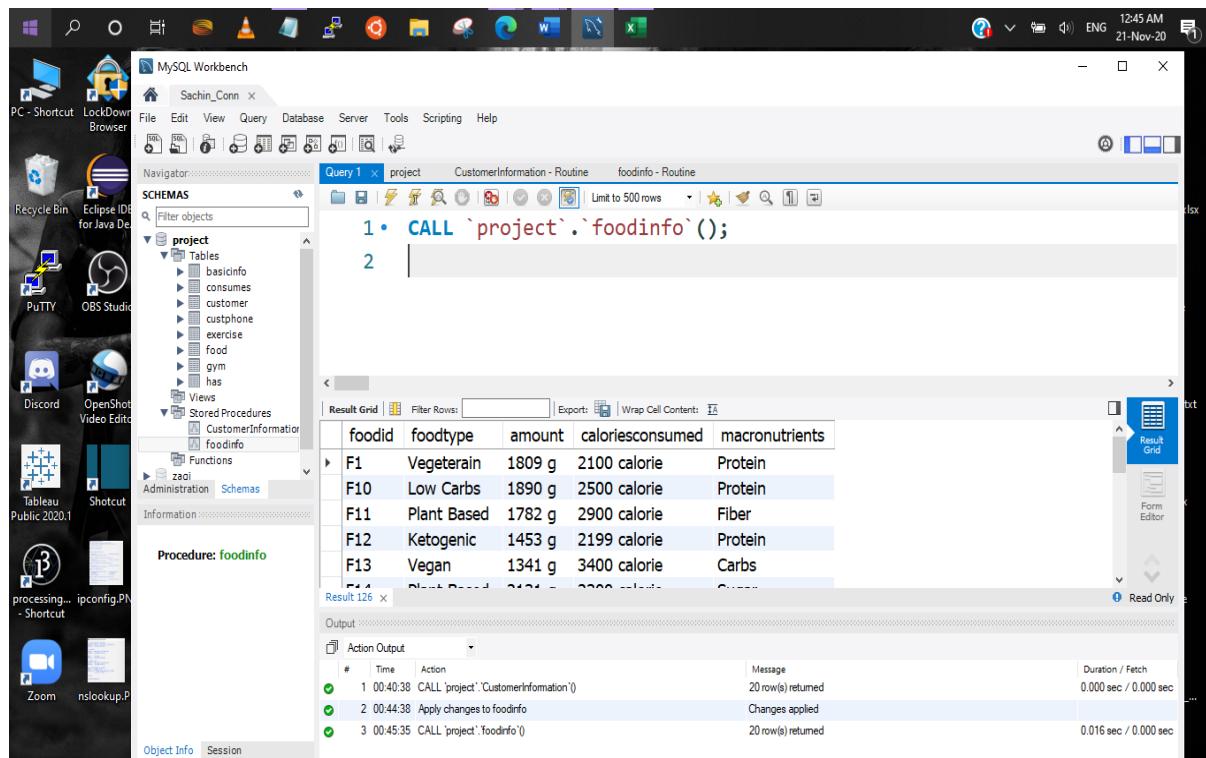
The screenshot shows the MySQL Workbench interface. In the center, the 'Routine' tab of the 'CustomerInformation - Routine' editor is active. The code area contains the following SQL:

```
1 • CREATE DEFINER='skarki'@'localhost'
2 PROCEDURE `foodinfo`()
3 BEGIN
4     select *
5     from food;
6 END
```

The 'Output' pane at the bottom shows two log entries:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|--------------------------------------|--------------------|-----------------------|
| 1 | 00:40:38 | CALL project.'CustomerInformation'() | 20 row(s) returned | 0.000 sec / 0.000 sec |
| 2 | 00:44:38 | Apply changes to foodinfo | Changes applied | |

Query Results:



The screenshot shows the MySQL Workbench interface with the 'Result Grid' tab selected. The query window contains the command:

```
1 • CALL `project`.`foodinfo`();
```

The result grid displays the following data:

| foodid | foodype | amount | caloriesconsumed | macronutrients |
|--------|-------------|--------|------------------|----------------|
| F1 | Vegetarian | 1809 g | 2100 calorie | Protein |
| F10 | Low Carbs | 1890 g | 2500 calorie | Protein |
| F11 | Plant Based | 1782 g | 2900 calorie | Fiber |
| F12 | Ketogenic | 1453 g | 2199 calorie | Protein |
| F13 | Vegan | 1341 g | 3400 calorie | Carbs |

The 'Output' pane at the bottom shows three log entries:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|--------------------------------------|--------------------|-----------------------|
| 1 | 00:40:38 | CALL project.'CustomerInformation'() | 20 row(s) returned | 0.000 sec / 0.000 sec |
| 2 | 00:44:38 | Apply changes to foodinfo | Changes applied | |
| 3 | 00:45:35 | CALL `project`.`foodinfo`() | 20 row(s) returned | 0.016 sec / 0.000 sec |

5. Summary

The proposed database has a potential to become an ideal prototype for a fully functioning database system . As it has presented 20 queries to demonstrate the reliability and accuracy of the database itself there is still room for various interpretations and iterations of same as well as additional queries. To sum up, the Nutrition Tracking Database is flexible and supportive of additional records and information.