



Lanka Nippon Biztech Institute

Assignment Name: **Project report submission**

Course Name: **Advance Database**

Name: **Achinthya Ashinsani**

Index Number: **UGC0122016**

Batch: **SE01**

Query 01:

```
SELECT user_name, body_weight FROM Users
WHERE weight_cat_id = (SELECT weight_cat_id FROM WeightCategory WHERE category_name = 'Normal');
```

□ This SQL query fetches the names and body weights of users classified within the "Normal" weight category. It does this by first locating the unique identifier (weight_cat_id) for the "Normal" category from the WeightCategory table, then using that identifier to filter records in the Users table. As a result, only users whose weight falls into the "Normal" range are included in the query's output, providing a focused view of users in this specific category.

Output:

	user_name character varying (100) 🔒	body_weight numeric (5,2) 🔒
1	Harsha Perera	59.80
2	Sithara Gunasinghe	70.00
3	Mason Clark	65.90
4	Kosala Kumarasinghe	55.40
5	Kusal Wickramasinghe	68.30
6	Pavithra Bandara	58.80
7	Sujeewa Perera	65.00
8	Vihanga Fernando	66.20
9	Chamika Wickramasinghe	64.20
10	Ella Patel	59.20
11	Sanjani Perera	55.70
12	Ravi Wickramasinghe	58.30
13	Tushara Seneviratne	62.10
14	Sunil Perera	65.70
15	Sanjay Amarasinghe	61.40
16	Hasini Silva	60.10
17	Anushka Rajapaksha	59.80
18	Vijitha Gunaratne	63.40
19	Sanjeewa Abeywardena	56.90
20	Harini Abeywardena	70.00
21	Buddhika Rajapaksha	68.40
22	Kosala Rajanaksha	62.40

1. Personalized Exercise Recommendations:

- Based on a user's weight category, personalized exercise recommendations can be made. For example, if a user is categorized as "Obese Class 3," exercises like "HIIT" or "Weightlifting" can be suggested for higher intensity workouts, while those in the "Normal" category could be recommended exercises like "Yoga" or "Walking."

- A decision can be made on what types of exercises would be most beneficial for an individual based on their weight category, targeting specific fitness goals (e.g., weight loss, muscle gain, endurance).

Query 02:

```
SELECT food_name, calories FROM Foods
WHERE weight_cat_id = (SELECT weight_cat_id FROM WeightCategory WHERE category_name = 'Obese Class 1');
```

- This SQL query retrieves the names and calorie counts of foods recommended for individuals in the "Obese Class 1" weight category. It accomplishes this by first identifying the unique identifier (weight_cat_id) for "Obese Class 1" from the WeightCategory table, then using that identifier to filter relevant records in the Foods table. As a result, the query returns only those foods that are suitable for individuals in the specified weight category, offering targeted dietary options based on calorie content.

Output:

	food_name character varying (100)	calories integer
1	Grilled Turkey Sandwich	350
2	Pasta Primavera	450
3	Tofu Scramble with Spinach	280
4	Lentil Soup	300
5	Crispy Tofu Bites	350
6	Fried Chicken with Collard Greens	750
7	Lamb Chop with Couscous	750
8	Beef and Vegetable Kabobs	550
9	Grilled Steak with Spinach	650

2. Dietary Recommendations:

- Based on a user's weight category, specific food items can be suggested that align with their caloric needs and help them maintain or achieve their desired body weight.

- For instance, if a user is categorized as "Obese Class 2," a lower-calorie food option like "Vegetable Soup" can be recommended, while someone in the "Underweight" category may be recommended higher-calorie options such as "Beef Stew" or "Chicken and Rice."

Query 03:

```
SELECT exercise_name, duration FROM Exercise
WHERE weight_cat_id = (SELECT weight_cat_id FROM WeightCategory WHERE category_name = 'Overweight');
```

- This SQL query retrieves the names and recommended durations of exercises tailored for individuals in the "Overweight" weight category. It works by first finding the unique identifier (weight_cat_id) for "Overweight" in the WeightCategory table, then using that identifier to filter the exercises in the Exercise table. The query result provides a list of exercises along with their durations specifically suited for individuals in the "Overweight" category, supporting targeted fitness recommendations.

Output:

	exercise_name character varying (100) 🔒	duration integer 🔒
1	Jogging	30
2	Lunges	25
3	Circuit Training	45
4	Bicep Curls	35
5	Powerlifting	45
6	Leg Raises	10
7	Medicine Ball Slams	35
8	Hamstring Curls	25
9	Inverted Rows	25

3. Tracking Caloric Intake:



- By querying the total calories consumed based on a user's food choices, decisions can be made on whether their current diet is within a healthy range or needs adjustments. If the total calories exceed daily recommendations, suggestions to reduce calorie intake could be offered.

Query 04:

```
SELECT wc.category_name, AVG(u.body_weight) AS average_weight
FROM Users u
JOIN WeightCategory wc ON u.weight_cat_id = wc.weight_cat_id
GROUP BY wc.category_name;
```

- This SQL query calculates the average body weight of users within each weight category. By joining the Users table with the WeightCategory table based on the weight_cat_id, it groups the data by each weight category (category_name). For each category, it then computes the average body weight and returns the category name alongside the calculated average. This query helps provide insights into typical body weights within each weight category, allowing for data-driven analysis of weight distribution across different classifications.

Output:

	category_name character varying (50) 	average_weight numeric 
1	Normal	62.3592592592592593
2	Overweight	83.2153846153846154
3	Obese Class 3	115.5200000000000000
4	Obese Class 1	94.9625000000000000
5	Moderately Underweight	46.6428571428571429
6	Slightly Overweight	75.2700000000000000
7	Severely Underweight	33.1000000000000000
8	Morbidly Obese	131.7750000000000000
9	Obese Class 2	106.2666666666666667
10	Underweight	52.6000000000000000

4. Exercise and Caloric Burn Management:

- Based on the exercise choices (e.g., "Running," "Swimming"), decisions can be made regarding how much time needs to be spent on a specific exercise to burn a certain number of calories. This can help users balance their caloric intake with physical activity to achieve weight goals.

Query 05:

```
SELECT food_name, calories FROM Foods
WHERE calories > 500;
```

- This SQL query retrieves the names and calorie counts of foods that have more than 500 calories. By filtering the Foods table where the calories column is greater than 500, it returns only those items that are relatively high in calories. This query can be useful for identifying high-calorie foods, which might be relevant for dietary planning or nutritional analysis.

Output:

	food_name character varying (100)	calories integer
1	Roast Pork with Mashed Potatoes	600
2	Lamb Curry	700
3	Beef Stir Fry	550
4	Turkey Burger with Sweet Potato Fries	600
5	Steak and Potatoes	750
6	Spaghetti Bolognese	650
7	Beef Burritos	600
8	Chicken Parmesan	700
9	Fish and Chips	650
10	Pork Schnitzel with Sauerkraut	600
11	Steak Salad	700
12	Chicken Stir Fry with Rice	600
13	Lasagna	750
14	Chicken Tenders with Fries	550
15	Lamb Kebab	600
16	Fried Chicken with Collard Greens	750
17	Pasta Alfredo	800
18	Pork Belly with Rice	700
19	Meatball Sub	600
20	Baked Chicken Wings	550
21	Lamb Chop with Couscous	750
22	Pulled Chicken Nachos	650

5. Health and Fitness Goals Planning:

- By analyzing a user's age, body weight, and weight category, decisions can be made about the most suitable combination of exercises and food choices to help them reach their goals, whether it's losing weight, gaining muscle, or improving general fitness.

Query 06:

```

SELECT user_name, body_weight FROM Users
WHERE weight_cat_id IN (
    SELECT weight_cat_id FROM WeightCategory
    WHERE category_name IN ('Underweight', 'Moderately Underweight')
);

```

- This SQL query retrieves the names and body weights of users who fall into either the "Underweight" or "Moderately Underweight" categories. It does this by first selecting the weight_cat_id values for both "Underweight" and "Moderately Underweight" from the WeightCategory table. The main query then uses these identifiers to filter users in the Users table whose weight_cat_id matches one of these categories. This query provides a focused view of users with body weights in the lower range, allowing for targeted analysis or recommendations for these specific groups.

Output:

	user_name character varying (100) 🔒	body_weight numeric (5,2) 🔒
1	Maya Fernando	40.80
2	Nadeesha Senaratne	48.20
3	Dinesh Fernando	49.50
4	Naveen Silva	45.70
5	Anjali Senanayake	50.00
6	Tharindu Wijesinghe	47.50
7	Chandana Perera	44.80
8	Dinesh Gunaratne	53.10
9	Amaya Fernando	54.20
10	Ruwan Perera	55.00
11	Vishal Senaratne	52.60
12	Ava Johnson	50.80
13	Ravi Gunaratne	51.60
14	Nirmala Wijesinghe	51.90
15	Tharindu Senanayake	53.50
16	Arjun Rathnayake	52.90
17	Heshan Silva	50.90
18	Ravi Wickremesinghe	52.30
19	Kasun Wickramasinghe	52.40

6. Targeting Specific Weight Categories:

- If the analysis of data reveals patterns in which certain weight categories (e.g., "Overweight" or "Morbidly Obese") are more prevalent, tailored programs or campaigns can be created to target those groups specifically, promoting weight management strategies.

Query 07:

```
SELECT exercise_name FROM Exercise
WHERE weight_cat_id = (SELECT weight_cat_id FROM WeightCategory WHERE category_name = 'Obese Class 2');
```

- This SQL query retrieves the names of exercises recommended for individuals in the "Obese Class 2" weight category. It achieves this by first locating the unique identifier (weight_cat_id) for "Obese Class 2" in the WeightCategory table, then using that identifier to filter the Exercise table. As a result, the query returns only those exercises that are specifically tailored for individuals in the "Obese Class 2" category, supporting targeted exercise recommendations for this group.

Output:

	exercise_name character varying (100) 🔒
1	HIIT
2	Zumba
3	Step Aerobics
4	Leg Press
5	Sprints on Treadmill
6	Hip Thrusts
7	Bicycle Crunches
8	Pull-up Bar Hold
9	Sumo Squats

7. User Progress Evaluation:

- By comparing initial weight categories with subsequent weight changes, decisions can be made about whether a user's diet or exercise routine needs to be adjusted. For example, if a user in the "Normal" category is gaining weight over time, it could indicate the need for dietary adjustments or more physical activity.

Query 08:

```
SELECT wc.category_name, COUNT(u.user_id) AS total_users
FROM Users u
JOIN WeightCategory wc ON u.weight_cat_id = wc.weight_cat_id
GROUP BY wc.category_name;
```

- The SQL query retrieves the total count of users within each weight category in the 'ActiveLife' database. By joining the Users table with the WeightCategory table based on the weight_cat_id field, it groups users according to their respective weight categories. For each category, it counts the number of users and presents the category name (category_name) along with the total number of users (total_users) per category. This provides a summarized view of user distribution across various weight classifications.

Output:

	category_name character varying (50) 🔒	total_users bigint 🔒
1	Normal	27
2	Overweight	13
3	Obese Class 3	10
4	Obese Class 1	8
5	Moderately Underweight	7
6	Slightly Overweight	10
7	Severely Underweight	3
8	Morbidly Obese	4
9	Obese Class 2	6
10	Underweight	12

8. Exercise Duration Optimization:

- Query results that provide the duration of different exercises can guide decisions on the optimal length of time for various activities. For example, "Swimming" and "Pilates" might be longer-duration exercises compared to "Jump Rope" or "Push-ups," which are more time-efficient.

Query 09:

```
SELECT user_name, age, wc.category_name
FROM Users u
JOIN WeightCategory wc ON u.weight_cat_id = wc.weight_cat_id
WHERE u.age > 40;
```

- This SQL query retrieves information about users over the age of 40 from the 'ActiveLife' database. It joins the Users table with the WeightCategory table using the weight_cat_id field to include each user's weight category. The query returns a list of users over 40, displaying each user's name (user_name), age, and associated weight category (category_name). This provides insight into the age and weight classification of older users in the system.

Output:

	user_name character varying (100)	age integer	category_name character varying (50)
1	Rajith Perera	41	Obese Class 1
2	Anura Jayawardena	42	Obese Class 1
3	Nalini De Silva	43	Obese Class 2
4	Thilini Fernando	44	Obese Class 2
5	Mahesh Senanayake	45	Obese Class 2
6	Priyantha Kumarasinghe	46	Obese Class 2
7	Rohan Perera	47	Obese Class 2
8	Sithum Senarath	48	Obese Class 2
9	Udaya Gunaratne	49	Obese Class 3
10	Nirmala Jayawardena	50	Obese Class 3
11	Lalith Abeysekara	51	Obese Class 3
12	Malini Gunasinghe	52	Obese Class 3
13	Chandima Perera	53	Obese Class 3
14	Chamara Dissanayake	54	Obese Class 3
15	Tharaka Bandara	55	Obese Class 3
16	Sujeewa Rathnayake	56	Obese Class 3
17	Ranjith Jayasinghe	57	Obese Class 3
18	Dharshana Perera	58	Obese Class 3
19	Suranga Wickramasinghe	59	Morbidly Obese
20	Aruna Wijesinghe	60	Morbidly Obese

9. Meal and Exercise Pairing:

- Decisions can be made about pairing foods with exercises for better overall health results. For example, a heavy meal like "Lamb Curry" can be paired with high-energy exercises like "HIIT," while lighter meals like "Grilled Chicken Salad" might be best combined with exercises like "Walking" or "Yoga."

Query 10:

```
SELECT food_name, calories FROM Foods
WHERE weight_cat_id IN (
    SELECT weight_cat_id FROM WeightCategory
    WHERE min_weight BETWEEN 80 AND 100
);
```

- This SQL query retrieves a list of foods along with their calorie content from the Foods table for users in specific weight categories. It filters the results based on weight categories where the minimum weight (min_weight) falls between 80 and 100. By using a subquery to select weight_cat_ids from the WeightCategory table that meet this weight criterion, the main query returns food_name and calories for foods recommended for these weight ranges. This helps identify suitable foods for individuals within this weight category.

Output:

	food_name character varying (100)	calories integer
1	Beef Stew	500
2	Grilled Turkey Sandwich	350
3	Baked Salmon with Quinoa	500
4	Pasta Primavera	450
5	Turkey Burger with Sweet Potato Fries	600
6	Tofu Scramble with Spinach	280
7	Vegetable Curry	400
8	Lentil Soup	300
9	Steak Salad	700
10	Crispy Tofu Bites	350
11	Vegetable Lo Mein	400
12	Fried Chicken with Collard Greens	750
13	Baked Chicken Wings	550
14	Lamb Chop with Couscous	750
15	Grilled Chicken with Roasted Potatoes	600
16	Beef and Vegetable Kabobs	550
17	Vegetable and Bean Burrito	500
18	Grilled Steak with Spinach	650

10. Caloric Burn and Exercise Intensity Correlation:

- From queries that calculate caloric burn during different exercises, decisions can be made about the intensity required for users to reach their daily caloric burn targets. For example, if someone needs to burn 500 calories, they could choose between a high-intensity exercise like "HIIT" or a longer duration exercise like "Swimming."

Query 11:

```
SELECT user_name FROM Users
WHERE body_weight > 120;
```

- This SQL query retrieves the names of users whose body weight exceeds 120 units (likely in kilograms or pounds, depending on the system) from the Users table. By filtering on the body_weight column, it identifies users whose weight is above this threshold and returns only their names (user_name). This query helps quickly locate users with body weight greater than 120 in the database.

Output:

	user_name character varying (100) 
1	Suranga Wickramasinghe
2	Aruna Wijesinghe
3	Indrani Aberathna
4	Amith Jayathilaka

11. User Demographics and Weight Category Trends:

- By analyzing user demographics (age, weight), you can identify patterns or trends in specific weight categories, helping in decision-making for group-based health interventions or personalized marketing strategies for food or exercise products.

```
SELECT food_name, calories FROM Foods
WHERE weight_cat_id = (SELECT weight_cat_id FROM WeightCategory WHERE category_name = 'Normal')
ORDER BY calories DESC
LIMIT 5;
```

Query 12:

- This SQL query retrieves the top five highest-calorie foods recommended for users in the "Normal" weight category. It begins by identifying the weight_cat_id for the "Normal" category in the WeightCategory table, then uses this weight_cat_id to filter foods from the Foods table. The results, which include food_name and calories, are ordered in descending order by calorie count (calories DESC) and limited to the top five entries. This query is useful for finding the most calorie-dense foods suitable for individuals in the "Normal" weight category.

Output:

	food_name character varying (100)	calories integer
1	Pork Schnitzel with Sauerkraut	600
2	Lamb Kebab	600
3	Meatball Sub	600
4	Chicken Fajitas	500
5	Grilled Salmon with Vegetables	450

12. Optimal Weight Category Distribution:

- By examining how users are distributed across different weight categories, organizations can decide where to focus their efforts in terms of designing health campaigns or product offerings. For example, if most users fall under "Normal" or "Underweight," the focus might be on maintaining or slightly improving these groups' health.

```
SELECT exercise_name, duration FROM Exercise
WHERE weight_cat_id = (SELECT weight_cat_id FROM WeightCategory WHERE category_name = 'Slightly Overweight')
ORDER BY duration DESC
LIMIT 3;
```

- This SQL query retrieves the top three exercises with the longest durations recommended for users in the "Slightly Overweight" weight category. It first finds the weight_cat_id corresponding to the "Slightly Overweight" category in the WeightCategory table, then filters the Exercise table based on this weight_cat_id. The query selects exercise_name and duration, orders the results by duration in descending order (ORDER BY duration DESC), and limits the output to the top three exercises with the longest durations. This helps identify exercises that are typically suggested for individuals in the "Slightly Overweight" category, focusing on those that involve more time.

Query 13:

Output:

	exercise_name character varying (100)	duration integer
1	Yoga	60
2	Tai Chi	60
3	Triceps Dips	30

13. Exercise Frequency Analysis:

- Analyzing exercise data from users could help decide which exercises are most popular or effective. For example, if "Running" and "Cycling" are the most performed exercises, decisions could be made to increase availability or accessibility for these activities, such as providing more outdoor tracks or cycling classes.

```
SELECT AVG(calories) AS average_calories FROM Foods  
WHERE weight_cat_id = (SELECT weight_cat_id FROM WeightCategory WHERE category_name = 'Severely Underweight');
```

- This SQL query calculates the average calorie content of foods recommended for users in the "Severely Underweight" weight category. It first identifies the weight_cat_id for the "Severely Underweight" category in the WeightCategory table, then filters the Foods table based on this weight_cat_id. The AVG(calories) function computes the average number of calories for the selected foods. The result is returned as average_calories, providing insight into the typical calorie content of foods recommended for individuals in this weight category.

Output:

	average_calories numeric
1	355.55555555555556

Query 14:

14. Nutrition and Fitness Alignment:

- By analyzing users' food intake and exercise duration, a better understanding of nutrition and fitness alignment can be formed. Decisions can be made on whether users are consuming enough calories to match their physical activity levels or if adjustments need to be made to achieve a balance that suits their goals.

```
SELECT exercise_name, duration FROM Exercise
WHERE duration > 30;
```

- This SQL query retrieves the names and durations of exercises that last longer than 30 minutes. It filters the Exercise table based on the duration column, selecting only those exercises where the duration exceeds 30 minutes. The results include the exercise_name and duration, providing a list of exercises that are relatively longer in duration. This query is useful for identifying exercises that require a more significant time commitment.

Output:

	exercise_name character varying (100)	duration integer
1	Cycling	40
2	Swimming	45
3	Yoga	60
4	Weightlifting	45
5	Pilates	60
6	Zumba	45
7	Dance Aerobics	40
8	Rowing	40
9	Tai Chi	60
10	Circuit Training	45
11	Step Aerobics	35
12	Deadlifts	40
13	Bicep Curls	35
14	Chest Press	45
15	Leg Press	50
16	Powerlifting	45
17	Interval Walking	45
18	Barbell Rows	40
19	Resistance Training	35
20	Front Squats	40
21	Medicine Ball Slams	35
22	Dumbbell Rows	40

Query 15:

15. Fitness Program Recommendations Based on User Weight:

- Based on weight categories and exercise history, decisions can be made to recommend tailored fitness programs that are both challenging and realistic. For example, users in the "Morbidly Obese" category may be recommended a more gradual program with lower-intensity exercises and lighter food options, whereas users in "Normal" or "Underweight" categories may be encouraged to take on higher-intensity workouts.

Query 16:

```
SELECT * FROM Users
WHERE gender = 'Male';
```

- This SQL query retrieves all columns (*) from the Users table where the value in the gender column is 'Male'. It returns the complete information for each male user in the table.

Output:

	user_id [PK] integer	user_name character varying (100)	age integer	body_weight numeric (5,2)	weight_cat_id integer	gender character varying (10)	checking timestamp without time zone
247	432	Ruwani Seneviratne	31	93.00	6	Male	2024-06-12 13:43:00
248	453	Kasun Rajapaksha	32	96.50	6	Male	2024-06-13 14:30:00
249	455	Chandana Gunasekara	30	98.30	6	Male	2024-06-15 16:00:00
250	457	Tharindu Fernando	32	100.10	6	Male	2024-06-17 17:30:00
251	458	Suresh Wickramasinghe	36	101.00	7	Male	2024-06-18 08:30:00
252	463	Ruwan Gunasekara	30	106.40	7	Male	2024-06-23 12:15:00
253	465	Kasun Seneviratne	31	108.60	7	Male	2024-06-25 13:45:00
254	467	Chathura Fernando	33	110.80	7	Male	2024-06-27 15:15:00
255	469	Tharindu Abeysekera	34	113.00	8	Male	2024-06-29 16:45:00
256	472	Suresh Abeysekera	30	116.30	8	Male	2024-07-02 09:15:00
257	473	Chandana Rajapaksha	31	117.40	8	Male	2024-07-03 10:00:00
258	474	Kasun Wijesinghe	33	118.50	8	Male	2024-07-04 10:45:00
259	478	Harshana Rajapaksha	30	122.90	8	Male	2024-07-08 13:45:00
260	480	Kasun Perera	33	125.10	8	Male	2024-07-10 15:15:00
261	481	Ruwan Rajapaksha	30	126.20	9	Male	2024-07-11 16:00:00
262	482	Tharindu Seneviratne	31	127.30	9	Male	2024-07-12 16:45:00
263	484	Suresh Fernando	32	129.50	9	Male	2024-07-14 08:30:00
264	489	Ruwan Gunasekara	30	135.00	9	Male	2024-07-19 12:15:00
265	490	Chathura Rajapaksha	33	136.10	9	Male	2024-07-20 13:00:00
266	492	Kasun Perera	35	138.30	10	Male	2024-07-22 14:30:00
267	497	Tharindu Fernando	32	143.80	10	Male	2024-07-27 08:30:00
268	500	Kasun Rajapaksha	31	147.10	10	Male	2024-07-30 10:45:00

16. Male Users Detailed Report

- This query would generate a report containing all the details of male users from the Users table. The report would include every column in the table for each male user, such as their name, email, registration date, contact details, or any other relevant attributes stored in the Users table. The data would be filtered specifically for male users, providing a comprehensive view of their information. give me a name for this report

Query 17:

```
SELECT checking FROM Users
WHERE gender = 'Female';
```

- The second query, `SELECT checking FROM Users WHERE gender = 'Female';`, selects only the checking column from the Users table for female users, where the value in the gender column is 'Female'. It returns the data in the checking column for each female user in the table.

Output:

	checking timestamp without time zone 	
216	2024-07-01 08:30:00	
217	2024-07-05 11:30:00	
218	2024-07-06 12:15:00	
219	2024-07-07 13:00:00	
220	2024-07-09 14:30:00	
221	2024-07-13 17:30:00	
222	2024-07-15 09:15:00	
223	2024-07-16 10:00:00	
224	2024-07-17 10:45:00	
225	2024-07-18 11:30:00	
226	2024-07-21 13:45:00	
227	2024-07-23 15:15:00	
228	2024-07-24 16:00:00	
229	2024-07-25 16:45:00	
230	2024-07-26 17:30:00	
231	2024-07-28 09:15:00	
232	2024-07-29 10:00:00	
233	2024-07-31 11:00:00	

17. Female Users - Checking Status Report

- This query would generate a report that only includes the checking column for female users from the Users table. The report would list the values in the checking column (which could be a specific attribute like status, a flag, or a test result) for all female users, without showing other columns or personal details. The report focuses solely on the data stored in the checking field for users whose gender is marked as 'Female'.