CIS 552 Database Design Fall 2023

(Professor - Yukui Luo)

Eco Friendly lifestyle tracker

Presented By: (Group – 15)

Kirti Shreyaa

Prerak Panwar

Shail Patel

Table of Contents

https://github.com/prerakpanwar/DatabaseProject

1. Executive Summary:

The Eco Friend Lifestyle Tracker System presents itself as a game-changing platform that could seamlessly integrate into real-world applications, serving as a powerful tool for building user awareness and fostering waste reduction practices. Through its gamified approach and rewarding system, the platform becomes an engaging experience for users, motivating them to actively participate in eco-friendly activities and earn points. This innovative concept not only incentivizes individual efforts but also introduces a social aspect where users can collectively contribute to a greener community. As users accumulate points, they unlock opportunities to attend nearby events, creating a direct correlation between their eco-friendly actions and real-world, community-based activities. This symbiotic relationship between the digital platform and real-world events ensures that the system transcends virtual boundaries, effectively influencing user behavior and building a community dedicated to sustainable living.

2. Introduction:

The Eco Friendly Lifestyle Tracker is an indispensable tool for the everyday user, revolutionizing personal environmental consciousness and contributing to broader sustainability goals. This innovative platform serves as a catalyst for spreading awareness by engaging users in a gamified experience centered around waste reduction. Through a goal-oriented approach, users set daily targets, log their activities, and earn points, creating a tangible connection between individual actions and positive environmental impact.

The system plays a pivotal role in sustainable development by fostering a sense of community and shared responsibility. Users progress through achievement levels, earning badges that signify their commitment to eco-friendly practices. This communal approach not only encourages friendly competition but also cultivates a collective mindset geared towards building a more sustainable future.

Utilizing the power of SQL technology, the Eco Friendly Lifestyle Tracker excels in waste management analysis. The system captures and processes user-generated data, providing valuable insights into individual and collective waste reduction efforts. By leveraging SQL's capabilities, we can perform manipulations on the data, enabling dynamic visualizations on the user dashboard. These visualizations not only offer a

comprehensive overview of individual achievements but also contribute to informed decision-making at both the individual and community levels.

Moreover, the platform's integration of SQL technology facilitates efficient decision-making. Users, motivated by the prospect of earning points and achieving higher levels, make daily choices that align with waste reduction practices. The system, through SQL-driven analytics, enables users to track their progress and make informed decisions regarding their daily activities, thus reinforcing positive habits.

In essence, the Eco Friendly Lifestyle Tracker transcends the conventional boundaries of waste reduction initiatives. It empowers users to actively participate in sustainable practices, contributes to broader environmental awareness, and harnesses the analytical capabilities of SQL to drive meaningful insights. As we navigate towards a greener future, this platform stands as a testament to the transformative potential of technology in shaping individual behaviors and fostering a global commitment to sustainability.

3. Problem Statement: (Developing the Eco Friendly Lifestyle Tracker System)

In today's world, the escalating environmental concerns demand innovative solutions to address the ever-increasing challenges of waste management and sustainable living. Recognizing the urgent need for a comprehensive and engaging approach to waste reduction, the lack of a centralized system that seamlessly integrates user participation, awareness building, and data-driven decision-making becomes apparent.

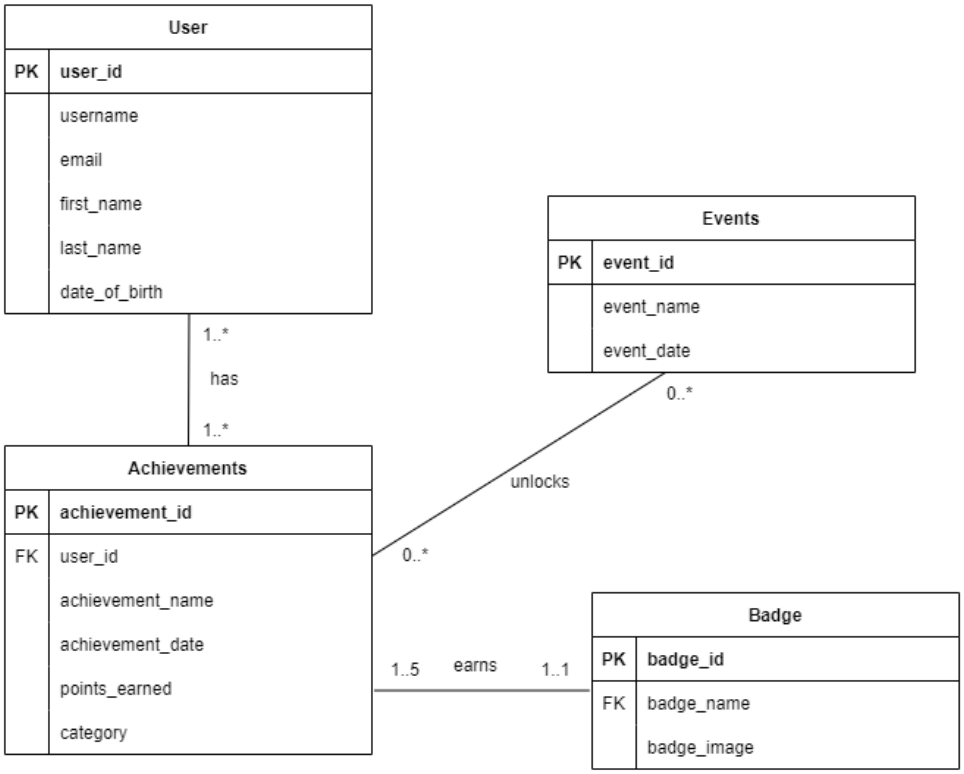
Current initiatives often fall short in actively involving individuals on a daily basis, resulting in limited impact and lack of sustained engagement. Moreover, there is a dearth of cohesive platforms that amalgamate technology, gamification, and waste management analytics to create a dynamic ecosystem fostering sustainable development.

The absence of a user-centric system that combines gamification, community engagement, and data analytics poses a significant challenge to realizing a collective commitment to waste reduction. Without an effective and centralized solution, individuals may lack motivation, awareness, and real-time insights into their environmental impact, hindering progress toward a greener and more sustainable future.

Hence, the problem at hand is to develop an innovative Eco Friendly Lifestyle Tracker System that addresses these gaps. This system must seamlessly integrate into users' daily lives, incentivize eco-friendly behaviors through gamification, build community awareness, and harness the power of data analytics to inform decision-making for sustainable living. Through this problem-solving initiative, we aim to bridge the existing gaps in waste reduction efforts and empower individuals to actively contribute to a more environmentally conscious society.

4. Database design

4.1 UML Diagram:



Entities:

1. User Table:

Fields:
user_id (INT, Primary Key)
username (VARCHAR(255))
email (VARCHAR(255))
first_name (VARCHAR(255))
last_name (VARCHAR(255))
date_of_birth (DATE)

2. Achievements Table:
Fields:
achievement_id (INT, Primary Key)
user_id (INT, Foreign Key to User Table)
achievement_name (VARCHAR(255))
achievement_date (DATE)
points_earned (INT)
category (VARCHAR(50))

3. Events Table
Fields:
event_id (INT, Primary Key)
event_name (VARCHAR(255))
event_date (DATE)

4. Badge Table
Fields:
badge_id (INT, Primary key)
badge_name (VARCHAR(255), Foreign Key to Achievement Table)
badge_image (VARCHAR(255))

Relationships:

User Table:
Relationships:
1-The user_id in the User Table is the Primary Key, uniquely identifying each user.
2-This user_id serves as a Foreign Key in the Achievement Table, linking achievements to specific users. This establishes a one-to-many relationship, as one user can have multiple achievements, but each achievement belongs to only one user.

Achievement Table:
Relationships:

1-The user_id in the Achievement Table is a Foreign Key, referencing the user_id in the User Table. This connection establishes a one-to-many relationship, signifying that one user can have multiple achievements recorded in this table.
2-The badge_name in the Badge Table is a Foreign Key that references the achievement_name in the Achievement Table. This creates a one-to-one relationship, indicating that each achievement can be associated with, at most, one badge.

Event Table:

Relationships:

1-While not explicitly defined in the provided schema, events could be associated with users based on their participation or attendance. The Event Table could potentially have a Foreign Key referencing the user_id in the Use
2-Table, establishing a relationship based on user involvement.

Badge Table:

Relationships:

1-The badge_name in the Badge Table is a Foreign Key that references the achievement_name in the Achievement Table.
2-This establishes a one-to-one relationship, indicating that each achievement can be associated with, at most, one badge.

In summary, the primary relationships are as follows:

-One-to-many relationship between User and Achievement tables, linked by the user_id.
-One-to-one relationship between Achievement and Badge tables, linked by the badge_name.
-The Event Table is not explicitly linked in the provided schema, but it could potentially have relationships with the User Table based on user participation.

Schema for the Tables are given below:

For schema we use CREATE TABLE statements for our 4 tables and ensured that primary key should not be null and mapped relations between the tables according to the dataset.

User Table Schema:

```sql
CREATE TABLE User (
    user_id INT NOT NULL PRIMARY KEY,
    username VARCHAR(255),
    email VARCHAR(255),
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    date_of_birth DATE
);
```

Achievements Table Schema:

```sql
CREATE TABLE achievements (
    achievement_id INT NOT NULL PRIMARY KEY,
    user_id INT,
    achievement_name VARCHAR(255),
    achievement_date DATE,
    points_earned INT,
    category VARCHAR(50),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);
```

Events Table Schema:

```sql
CREATE TABLE events (
    event_id INT NOT NULL PRIMARY KEY,
    event_name VARCHAR(255),
    event_date DATE
);
```

Badge Table Schema:

```sql
CREATE TABLE Badge (
    badge_id INT NOT NULL PRIMARY KEY,
    badge_name VARCHAR(255),
    badge_image VARCHAR(255),
    FOREIGN KEY (badge_name) REFERENCES achievements (acheievement_name)
);
```

5. Data collection and SQL Query Development:

We use our own created data using python libraries like :

from faker import Faker
from datetime import datetime, timedelta
import random

We will also upload the code used in our GitHub Repository.

Sample data Screenshots of the tables are as follows taking only first 5 records:

1. **User Table**

```
672
673 •  select * from user LIMIT 5;
674
675
676
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

user_id	username	email	first_name	last_name	date_of_birth
1	emmaadams	michelle03@example.org	David	Charles	1958-11-05
2	erica97	pjackson@example.com	Ashley	Gonzalez	1960-12-08
3	derek38	jay45@example.org	Olivia	Wilcox	1999-11-24
4	alexis07	nancybailey@example.org	Ashlee	Smith	1958-08-29
5	fieldsderek	ashleywilson@example.com	Mark	Salinas	1956-01-20
NULL	NULL	NULL	NULL	NULL	NULL

2. **Achievement Table:**

```
672
673 •  select * from achievements LIMIT 5;
674
675
676
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

achievement_id	user_id	achievement_name	achievement_date	points_earned	category
1	1	Apprentice	2023-05-29	119	Community Service Leader
2	2	Skywalker	2023-08-27	340	Recycling Manager
3	3	Apprentice	2023-04-12	119	Community Service Leader
4	4	Novice	2023-05-02	39	Recycling Manager
5	5	Novice	2021-03-24	57	Waste Management Captain
NULL	NULL	NULL	NULL	NULL	NULL

3. **Events Table:**

```
673 •  select * from events LIMIT 5;
674
675
676
```

Result Grid | Filter Rows: | Edit:

event_id	event_name	event_date
1	Waste Reduction Campaign	2024-05-18
2	Recycle and Vibe	2024-03-12
3	Beers and Peers	2024-02-18
NULL	NULL	NULL

4. **Badge Table:**

```
672
673 •  select * from badge LIMIT 5;
674
675
676
```

Result Grid | Filter Rows: | Edit:

Badge_id	Badge_Name	Badge_image
1	Novice	yellow
2	Apprentice	blue
3	Explorer	green
4	Skywalker	orange
5	Expert	purple
NULL	NULL	NULL

<u>We also used some preprocessed techniques to ensure the quality and consistency of your dataset:</u>

1)Data Cleaning like a)Removing Duplicates, b)Handling NULL Values:

1-Remove duplicate records from the User table.

DELETE u1 FROM User u1

JOIN User u2 ON u1.User_ID < u2.User_ID AND u1.Username = u2.Username;

2-Set default values for NULLs in the User table.

UPDATE User SET Date_of_Birth = '1900-01-01' WHERE Date_of_Birth IS NULL;

These are just to show that these inconsistencies can be handled via SQL codes mentioned above.

2)Data Transformation like a)Converting Date Format:

Change the date format in the Event table.

UPDATE Event SET Event_Date = DATE_FORMAT (Event_Date, '%Y-%m-%d');

This step is necessary so that the data is imported efficiently without errors in date format.
We can also do the below 2 transformations but our dataset does not require it.

b)Calculate Age from Date of Birth:

Add an "Age" column to the User table.

ALTER TABLE User ADD COLUMN Age INT;

UPDATE User SET Age = YEAR(NOW()) - YEAR(Date_of_Birth);

c)Concatenate Columns:

Combine the first and last names into a single "Full_Name" column.

ALTER TABLE User ADD COLUMN Full_Name VARCHAR(255);

UPDATE User SET Full_Name = CONCAT(First_Name, ' ', Last_Name);

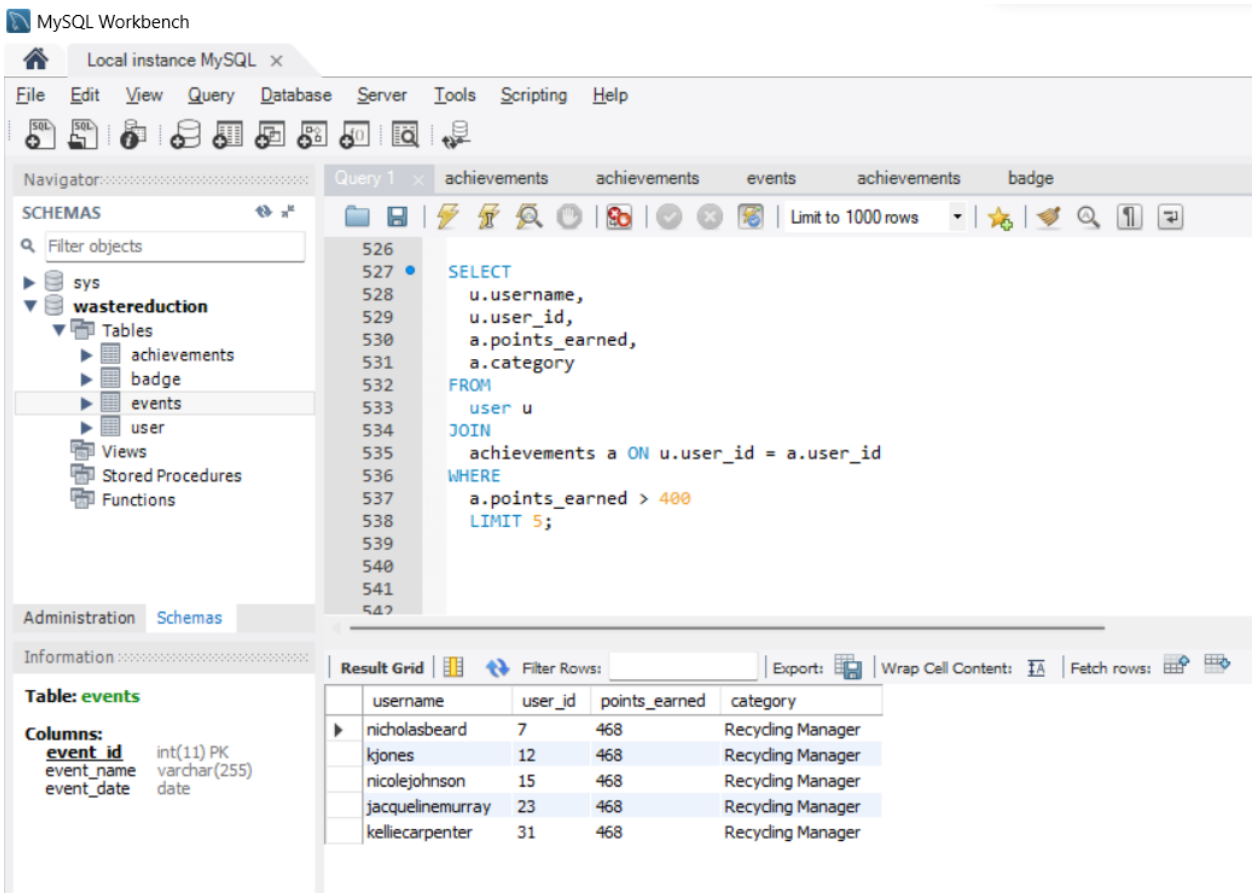
5.2-SQL Query Development:

Q1. Users with points above 200 from any category are eligible to attend the Waste Reduction Campaign event Users with points above 300 from any category are eligible to attend Recycle and Vibe event Users with points above 400 from any category are eligible to attend Beers and Peers event Based on this data, which users are eligible to attend Beers and Peer?

SQL Query:

SELECT

```
  u.username,
  u.user_id,
  a.points_earned,
  a.category
FROM
  user u
JOIN
  achievements a ON u.user_id = a.user_id
WHERE
  a.points_earned > 400
  LIMIT 5;
```

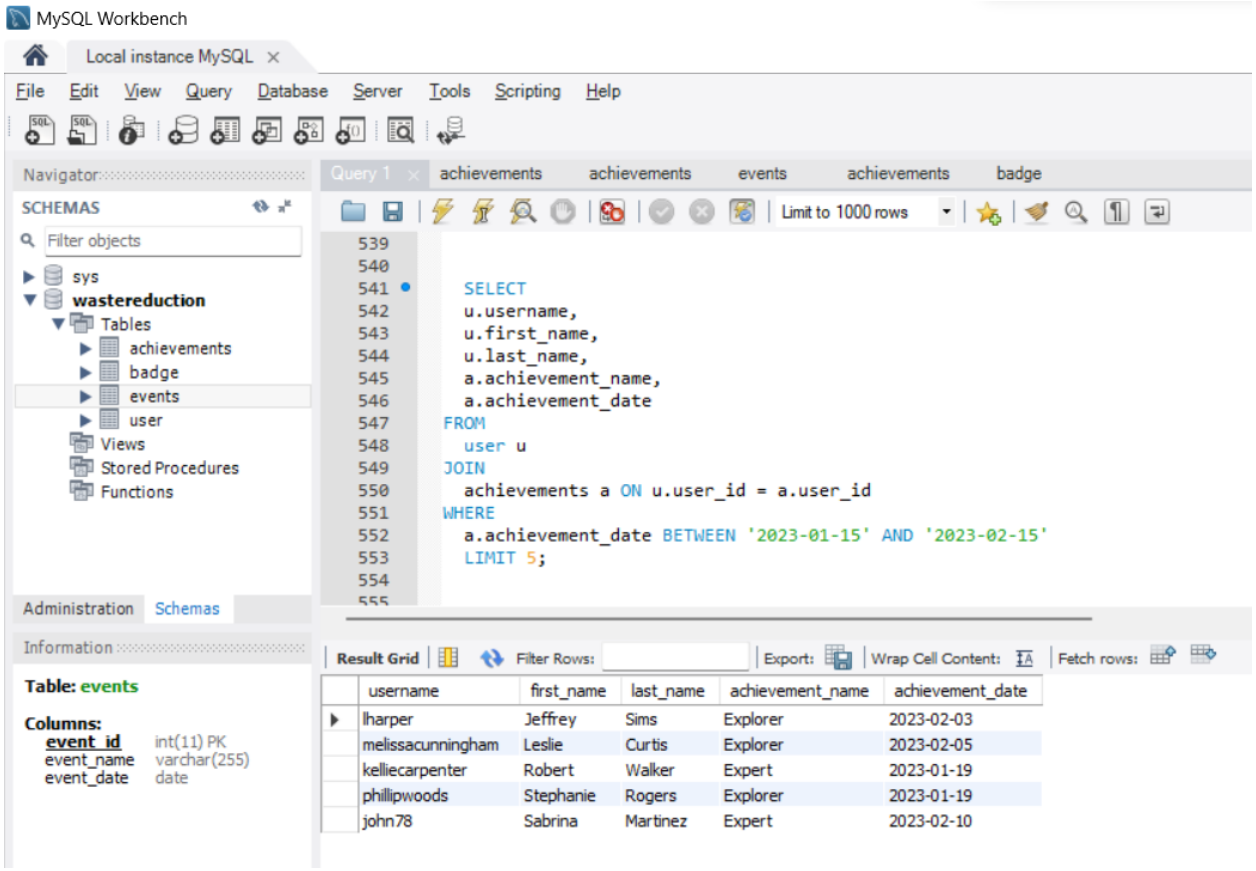


Q2. To identify users who achieved significant milestones such as during a specific date range.

SQL Query:

```
SELECT
```

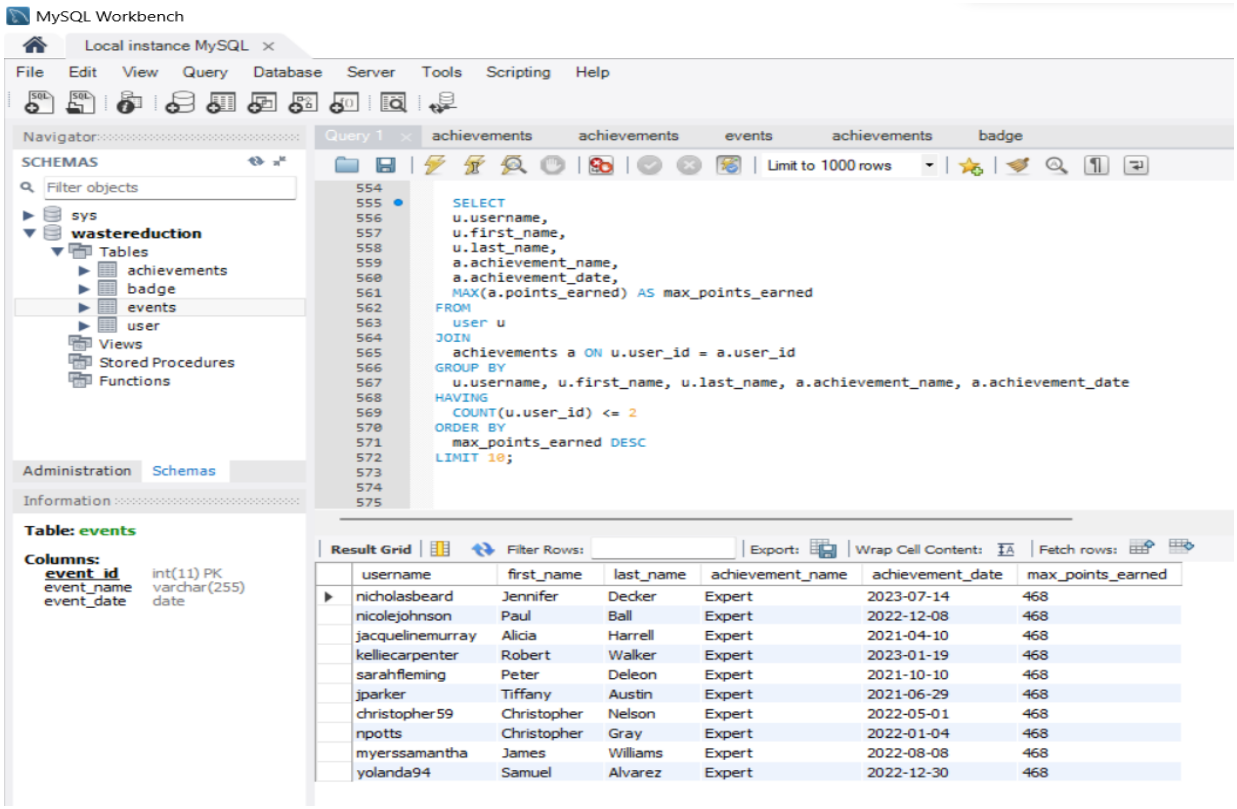
```
  u.username,
  u.first_name,
  u.last_name,
  a.achievement_name,
  a.achievement_date
FROM
  user u
JOIN
  achievements a ON u.user_id = a.user_id
WHERE
  a.achievement_date BETWEEN '2023-01-15' AND '2023-02-15'
  LIMIT 5;
```



Q3. The maximum points gained by a user is beyond 400 and the achievement is "expert". Based on this information, we need to check if there are any users who have earned points beyond 400 and gained "expert" level. Hence we need a query for a rare occurance of achievements.

SQL Query:

```
SELECT
  u.username,
  u.first_name,
  u.last_name,
  a.achievement_name,
  a.achievement_date,
  MAX(a.points_earned) AS max_points_earned
FROM
  user u
JOIN
  achievements a ON u.user_id = a.user_id
GROUP BY
  u.username, u.first_name, u.last_name, a.achievement_name, a.achievement_date
HAVING
  COUNT(u.user_id) <= 2
ORDER BY
  max_points_earned DESC
LIMIT 10;
```



6. Performance Tuning:

6.1 Indexing

The created indexes enhance query performance by facilitating faster data retrieval and optimized search, filtering, and sorting operations on the specified columns (user_id, username, achievement_date, category, and event_date) in their respective tables (user, achievements, and events).

CREATE INDEX idx_user_id ON user(user_id);

CREATE INDEX idx_username ON user(username);

CREATE INDEX idx_user_id ON achievements(user_id);

CREATE INDEX idx_achievement_date ON achievements(achievement_date);

CREATE INDEX idx_category ON achievements(category);

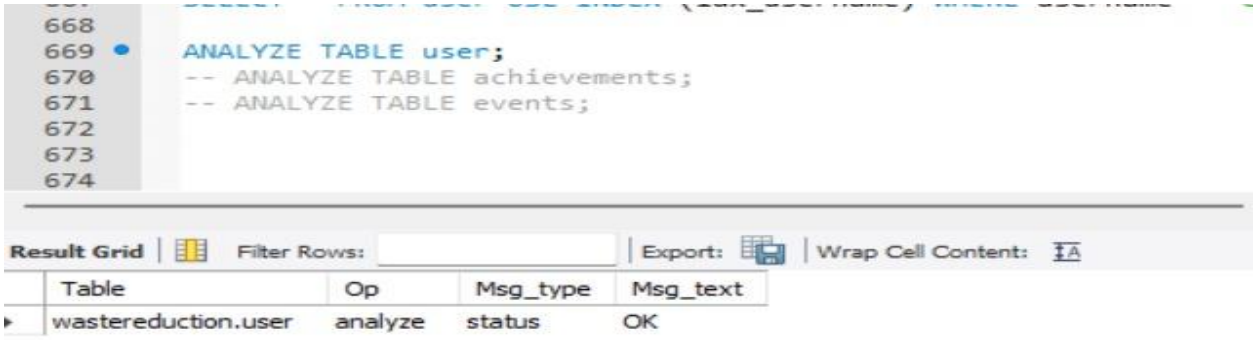
CREATE INDEX idx_event_date ON events(event_date);

The EXPLAIN statement provides insights into the query execution plan, and in this specific query, it shows how MySQL plans to retrieve all columns from the User table where the Username is 'example', revealing details about index usage and access methods.

```
672
673 •  EXPLAIN SELECT * FROM user WHERE username = 'erica97';
674
675
676
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	user	NULL	const	username	username	1023	const	1	100.00	NULL



```
668
669 •  ANALYZE TABLE user;
670    -- ANALYZE TABLE achievements;
671    -- ANALYZE TABLE events;
672
673
674
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Table	Op	Msg_type	Msg_text
wastereduction.user	analyze	status	OK

Here the ANALYZE TABLE statements update and maintain statistics, enabling the MySQL query optimizer to make informed decisions for efficient query execution on the respective tables (User, Achievement, Event).

7. Questions and Answers:

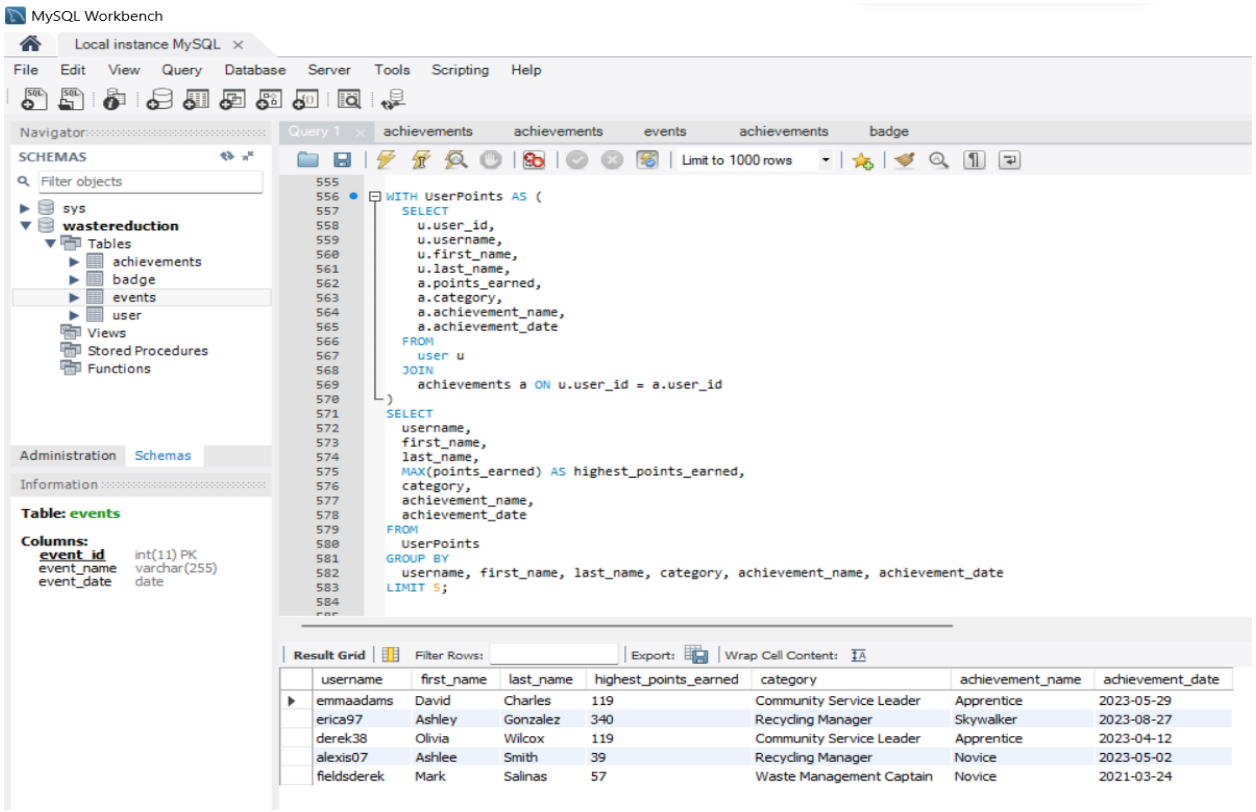
7.1. This query combines the "Top Achiever" and "User Contributions" questions by finding the user with the highest total points earned across all categories.

TRC1:

{ <username, first_name, last_name, highest_points_earned, category, achievement_name, achievement_date> |
  ∃ u, a (
   u.user_id = a.user_id ∧
   UserPoints(u, a) ∧
   (∀ a' (
    UserPoints(u, a') → a'.points_earned ≤ a.points_earned
   )) ∧
   ∃10 c, an, ad (
    c, an, ad IN a.category, a.achievement_name, a.achievement_date ∧
    (username, first_name, last_name, MAX(points_earned), c, an, ad) IN UserPoints(u, a)
)
)
}

Explanation:

UserPoints(u, a) represents the relation obtained from the join of user and achievements.
∀ denotes universal quantification, and ∃ denotes existential quantification.
The condition (username, first_name, last_name, MAX(points_earned), c, an, ad) IN UserPoints(u, a) specifies that the selected tuple is in the UserPoints relation.



7.2. This query combines the "Category Leaders" and "User Contributions" questions by finding users with the highest points in each category of achievements.

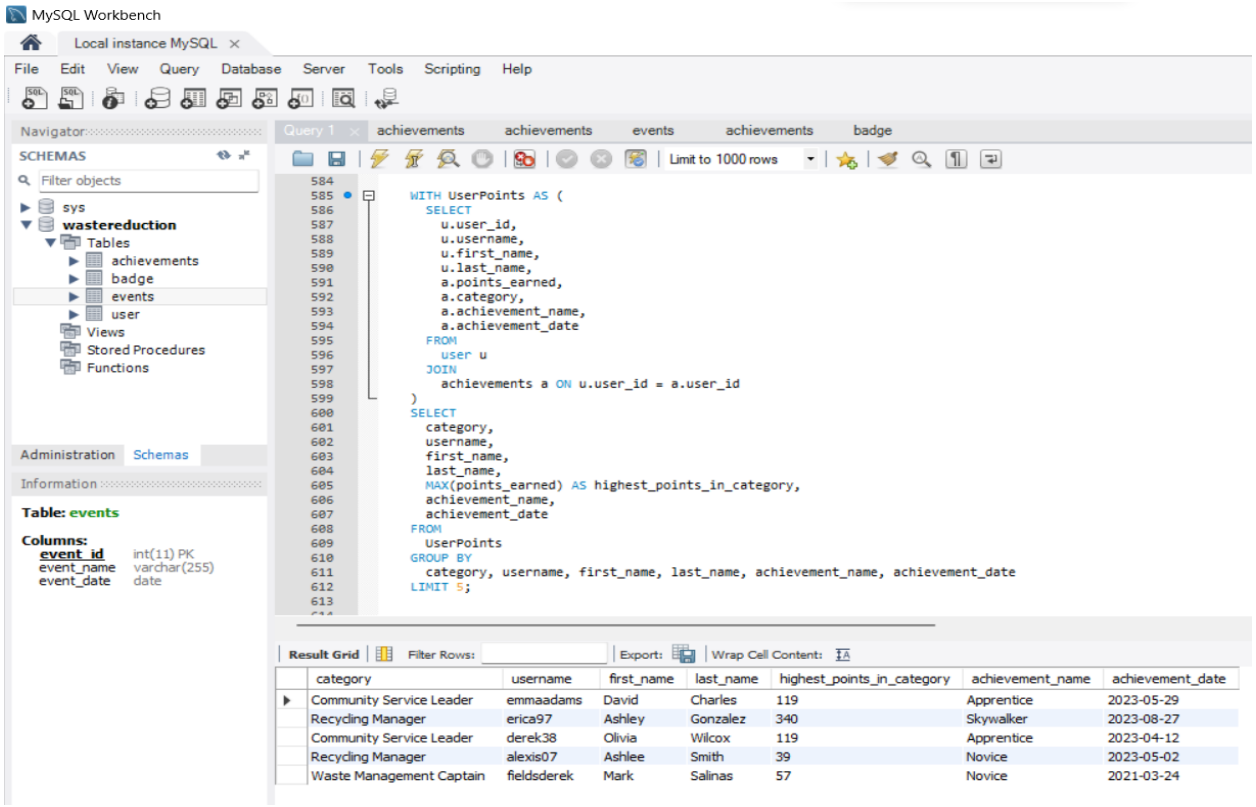
TRC2:

{ <category, username, first_name, last_name, highest_points_in_category, achievement_name, achievement_date> |
∃ u, a (
u.user_id = a.user_id ∧
UserPoints(u, a) ∧
(∀ a' (

UserPoints(u, a') → (a'.category = a.category AND a'.points_earned ≤ a.points_earned)
)) ∧
∃10 an, ad (
an, ad IN a.achievement_name, a.achievement_date ∧
(category, username, first_name, last_name, MAX(points_earned), an, ad) IN UserPoints(u, a)
)
)
}

Explanation:

UserPoints(u, a) represents the relation obtained from the join of user and achievements.
∀ denotes universal quantification, and ∃ denotes existential quantification.
The condition (category, username, first_name, last_name, MAX(points_earned), an, ad) IN UserPoints(u, a) specifies that the selected tuple is in the UserPoints relation and contains the maximum points earned in the category.



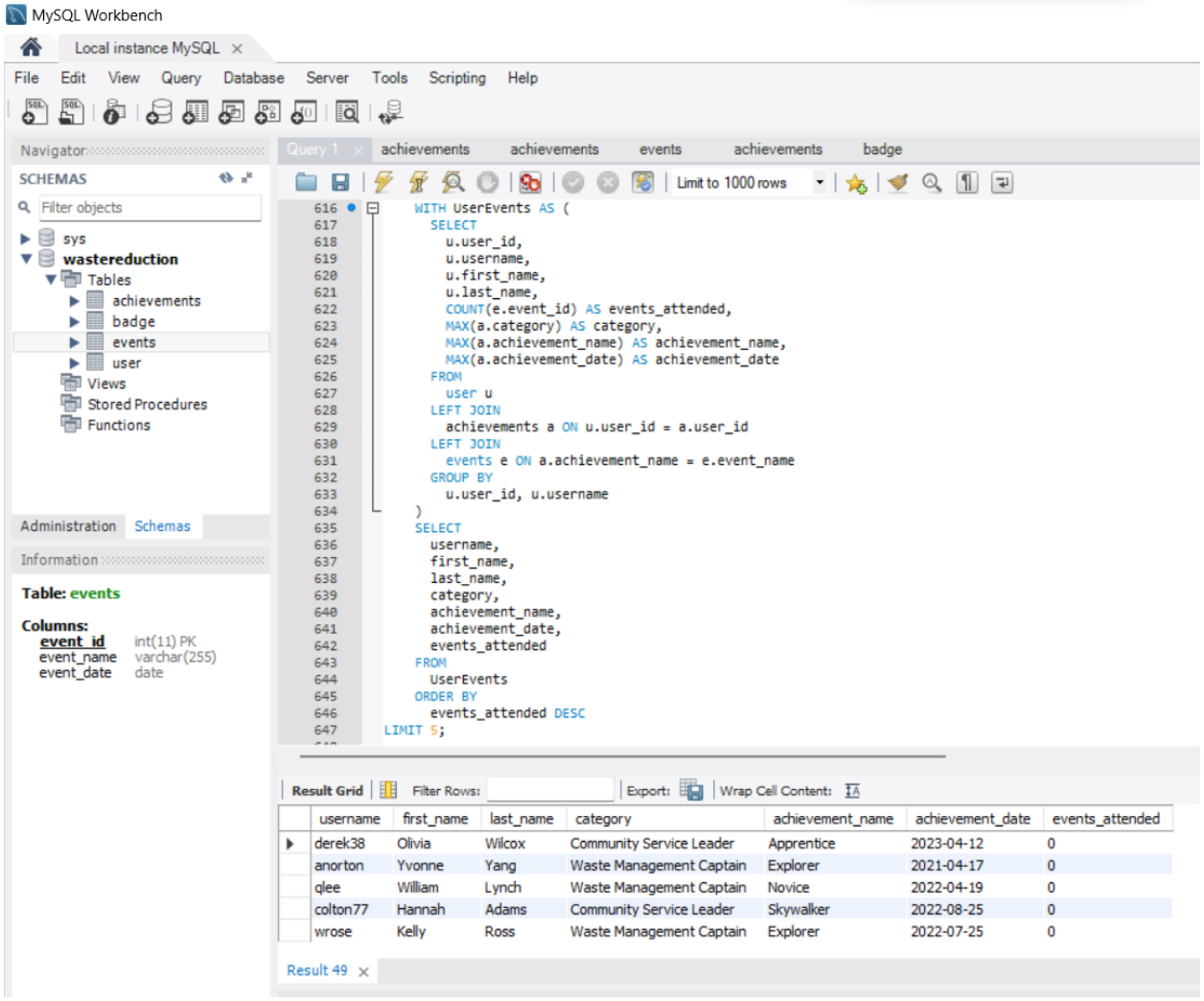
7.3. This query combines the "Event Participation" and "Event Attendance Trends" questions by finding users who have participated in the most events and showing a trend in user participation over time.

TRC3:

{ <username, first_name, last_name, category, achievement_name, achievement_date, events_attended> |
  ∃ u, a, e (
    u.user_id = a.user_id ∧
    u.user_id = e.user_id ∧
    UserEvents(u, a, e) ∧
    (∀ e' (
      u.user_id = e'.user_id ∧ e'.event_id ≠ e.event_id → e'.event_name ≠ a.achievement_name
    )) ∧
    (username, first_name, last_name, MAX(a.category), MAX(a.achievement_name), MAX(a.achievement_date), COUNT(e.event_id)) IN UserEvents(u, a, e)
)
}

Explanation:

UserEvents(u, a, e) represents the relation obtained from the left join of user, achievements, and events.
∀ denotes universal quantification, and ∃ denotes existential quantification.
The condition (username, first_name, last_name, MAX(a.category), MAX(a.achievement_name), MAX(a.achievement_date), COUNT(e.event_id)) IN UserEvents(u, a, e) specifies that the selected tuple is in the UserEvents relation.



8. Limitations:

Dependency on User Input:

The effectiveness of the system relies heavily on users consistently inputting their daily activities. If users fail to regularly participate, the system's ability to track achievements and influence behavior diminishes.

Event Eligibility Criteria:
The condition of requiring users to achieve 500 points to be eligible for community cleanup events may exclude certain users or discourage participation from those with lower point totals.

Lack of Behavioral Insights:
While the system captures user data, it may not provide in-depth behavioral insights into why certain activities are chosen or avoided. Understanding user motivations and barriers is crucial for sustained behavior change.

Data Quality:
Inaccurate User Inputs: The system's accuracy heavily depends on users providing truthful and accurate information. Inconsistent or erroneous user inputs could lead to unreliable data for analysis.

Performance:
Slow Response Times: As the user base grows, there may be performance challenges, resulting in slow response times. This could affect the user experience and discourage active participation.

Security:
Data Privacy Concerns: Collecting and storing user data poses privacy risks. Inadequate security measures could lead to unauthorized access, potentially compromising sensitive user information.

Integration:

Limited External Integrations: The system may face challenges in integrating seamlessly with external systems, such as waste management databases or community event platforms, limiting its ability to contribute to broader initiatives.

Real-time Updates:
Delayed Data Updates: Real-time updates may be hindered by delays in processing user inputs or achievements. Users may not see immediate reflections of their actions, impacting the dynamic and gamified nature of the system.

User Engagement:
Lack of User Engagement: If users do not find the system engaging or rewarding, there may be a lack of consistent engagement. Keeping users motivated over the long term poses a challenge.