# Entertainment Club Activity Registration System Project Report

## Project launch document

### Business value of the system:

This entertainment club activity registration system has significant business value, mainly reflected in the improvement of operational efficiency and enhance the member experience two aspects. First of all, the system can efficiently manage club activities and member registration, simplify the management process of staff, and thus greatly improve the operational efficiency of the club. By leveraging the data-driven insights provided by the reporting function, management can make more informed decisions and further optimize the club's overall operations.

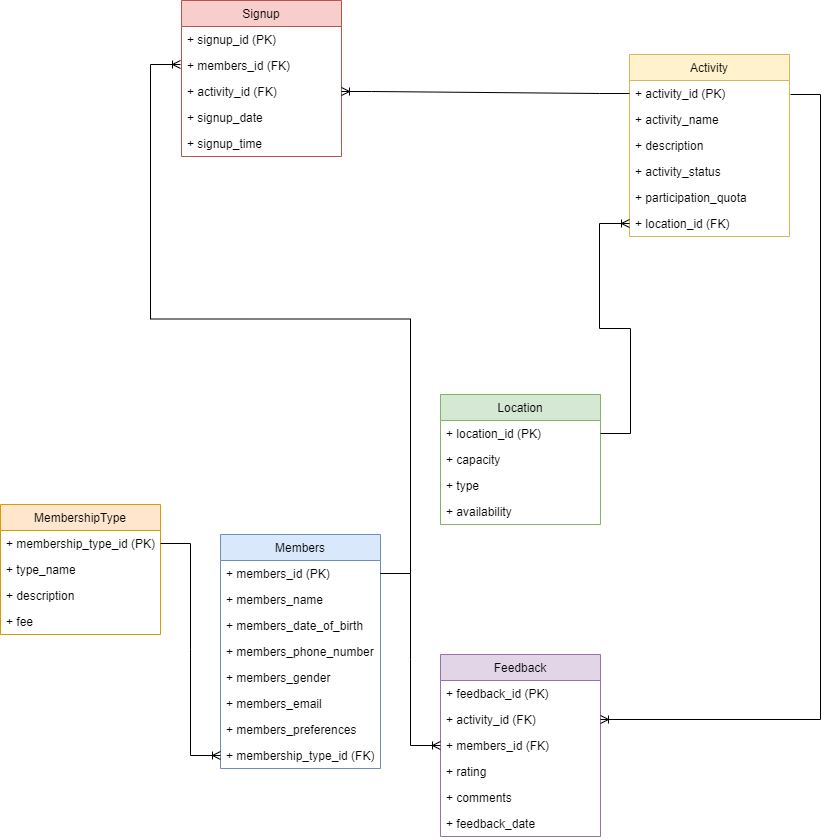
Second, the system significantly increases member engagement through personalized activity recommendations, making it easier for members to find activities that match their interests. At the same time, the system has improved the way events are organized and feedback is collected, which not only increases member satisfaction, but also provides valuable suggestions for the club to improve. Together, these features not only enhance the club experience for members, but also help increase member loyalty and the long-term growth of the club.

### System Scope:

* Membership Management (personal data, preferences, type of membership)
* Activity Management (Create, edit, and delete activities)
* Venue management (capacity, type and availability tracking)
* Event registration system
* Feedback collection and management
* Reporting and analysis functions
* Administrator user-friendly graphical interface for all system functions

## Design specification

#### E-R diagram：



#### E-R diagram interpretation：

**Entity:** Members, Activity, Location, Signup, Feedback, MembershipType.

**Relation:**

* Members participate in activities (many-to-many, indicated by Signup)
* Member Members provide Feedback on activities (many-to-many)
* The Activity is held at the Location of the venue (many to one)
* Members Specifies the member type. MembershipType (many-to-one)

**Business rules and constraints:**

1. Each member must have a unique identifier (members\_id).
2. Each activity must have a unique identifier (activity\_id)
3. Each location must have a unique identifier (location\_id).
4. The activity status can only be "Active", "Inactive" or "Cancelled"
5. Member's gender can only be "male", "female" or "other"
6. Feedback rating must be between 1 and 5
7. The number of Activity registrations cannot exceed their participation quota
8. Member activities cannot register for the same Activity more than once
9. We can only provide feedback for activities that members have signed up for
10. When deleting the location of the venue, all related activities should be deleted
11. When deleting an activity, delete all signups and feedback related to the activity
12. When deleting members, all relevant signup and feedback should be deleted

### Logical data model:

Members(members\_id, members\_name, members\_date\_of\_birth, members\_phone\_number, members\_gender, members\_email, members\_preferences, membership\_type\_id)  
MembershipType(membership\_type\_id, type\_name, description, fee)  
Activity(activity\_id, activity\_name, description, activity\_status, participation\_quota, location\_id)  
Location(location\_id, capacity, type, availability)  
Signup(signup\_id, members\_id, activity\_id, signup\_date, signup\_time)  
Feedback(feedback\_id, activity\_id, members\_id, rating, comments, feedback\_date)

**Functional dependence:**  
Members：members\_id → {members\_name, members\_date\_of\_birth, members\_phone\_number, members\_gender, members\_email, members\_preferences, membership\_type\_id}  
MembershipType：membership\_type\_id → {type\_name, description, fee}  
Activity：activity\_id → {activity\_name, description, activity\_status, participation\_quota, location\_id}  
Location：location\_id → {capacity, type, availability}  
Signup：signup\_id → {members\_id, activity\_id, signup\_date, signup\_time}  
Feedback：feedback\_id → {activity\_id, members\_id, rating, comments, feedback\_date}

All relationships are in the third normal form (3NF) because there are no passing dependencies and all properties are completely dependent on their respective primary keys.

### Data Dictionary

#### Members Table

| Column Name | Data Type | Constraint | Description |
| --- | --- | --- | --- |
| members\_id | INT | PK | Unique identifier for each member |
| members\_name | VARCHAR(100) |  | Full name of the member |
| members\_date\_of\_birth | DATE |  | Member's date of birth |
| members\_phone\_number | VARCHAR(20) |  | Member's contact number |
| members\_gender | ENUM |  | Member's gender (Male, Female, Other) |
| members\_email | VARCHAR(100) |  | Member's email address |
| members\_preferences | TEXT |  | Member's activity preferences |
| membership\_type\_id | INT | FK | References MembershipType table |

#### MembershipType Table

| Column Name | Data Type | Constraint | Description |
| --- | --- | --- | --- |
| membership\_type\_id | INT | PK | Unique identifier for each membership type |
| type\_name | VARCHAR(50) |  | Name of the membership type |
| description | TEXT |  | Description of the membership type |
| fee | DECIMAL(10, 2) |  | Cost of the membership |

#### Activity Table

| Column Name | Data Type | Constraint | Description |
| --- | --- | --- | --- |
| activity\_id | INT | PK | Unique identifier for each activity |
| activity\_name | VARCHAR(100) |  | Name of the activity |
| description | TEXT |  | Detailed description of the activity |
| activity\_status | ENUM |  | Current status of the activity (Active, Inactive, Cancelled) |
| participation\_quota | INT |  | Maximum number of participants allowed |
| location\_id | INT | FK | References Location table |

#### Location Table

| Column Name | Data Type | Constraint | Description |
| --- | --- | --- | --- |
| location\_id | INT | PK | Unique identifier for each location |
| capacity | INT |  | Maximum capacity of the location |
| type | VARCHAR(50) |  | Type or name of the location |
| availability | BOOLEAN |  | Whether the location is currently available |

#### Signup Table

| Column Name | Data Type | Constraint | Description |
| --- | --- | --- | --- |
| signup\_id | INT | PK | Unique identifier for each signup record |
| members\_id | INT | FK | References Members table |
| activity\_id | INT | FK | References Activity table |
| signup\_date | DATE |  | Date of signup |
| signup\_time | TIME |  | Time of signup |

#### Feedback Table

| Column Name | Data Type | Constraint | Description |
| --- | --- | --- | --- |
| feedback\_id | INT | PK | Unique identifier for each feedback record |
| activity\_id | INT | FK | References Activity table |
| members\_id | INT | FK | References Members table |
| rating | INT |  | Numeric rating given by the member (1-5) |
| comments | TEXT |  | Additional comments provided by the member |
| feedback\_date | DATE |  | Date when the feedback was submitted |

## Configuration specification

**SQL statement configuration and specification:**

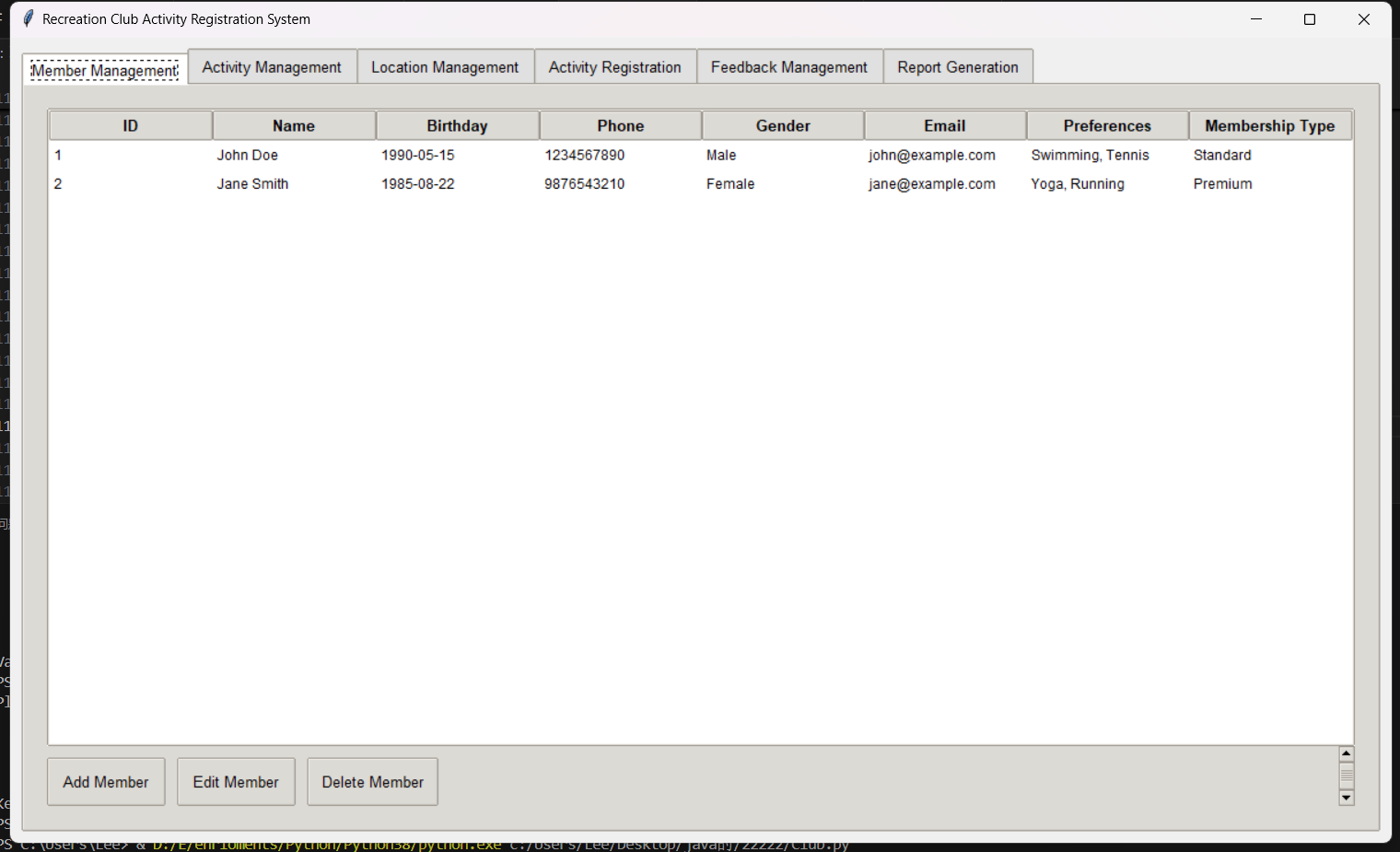
1. Basic query: List all members

* Purpose: To retrieve all club member information for administrative purposes.

SQl codes:

SELECT \* FROM Members;

Project screenshot:



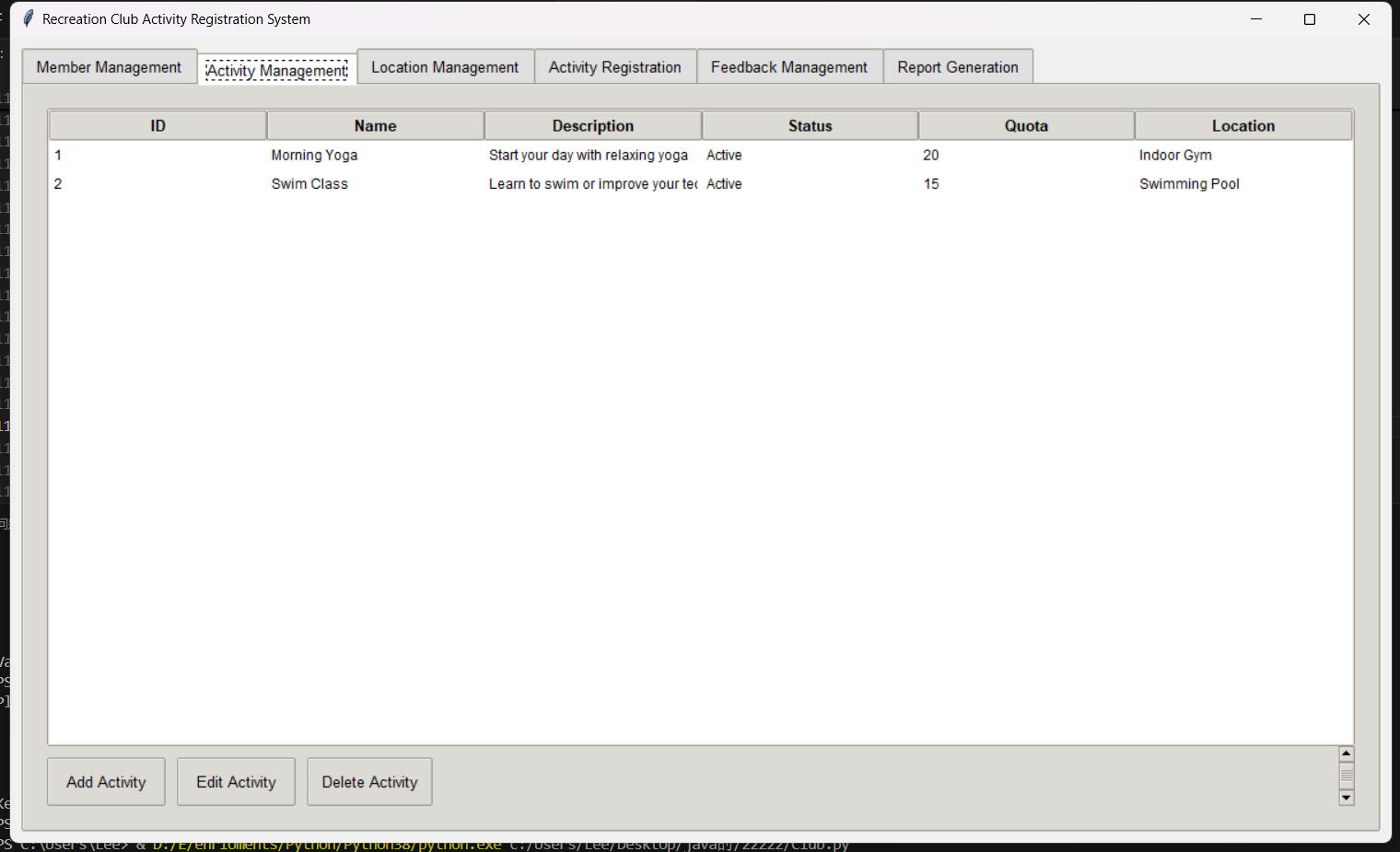
1. Join Query: Get all events and their venues

* Purpose: To display a list of events and their respective venues for better event management.

SQl codes:

SELECT a.activity\_name, l.type AS location\_type
  
FROM Activity a
  
JOIN Location l ON a.location\_id = l.location\_id;

Project screenshot:



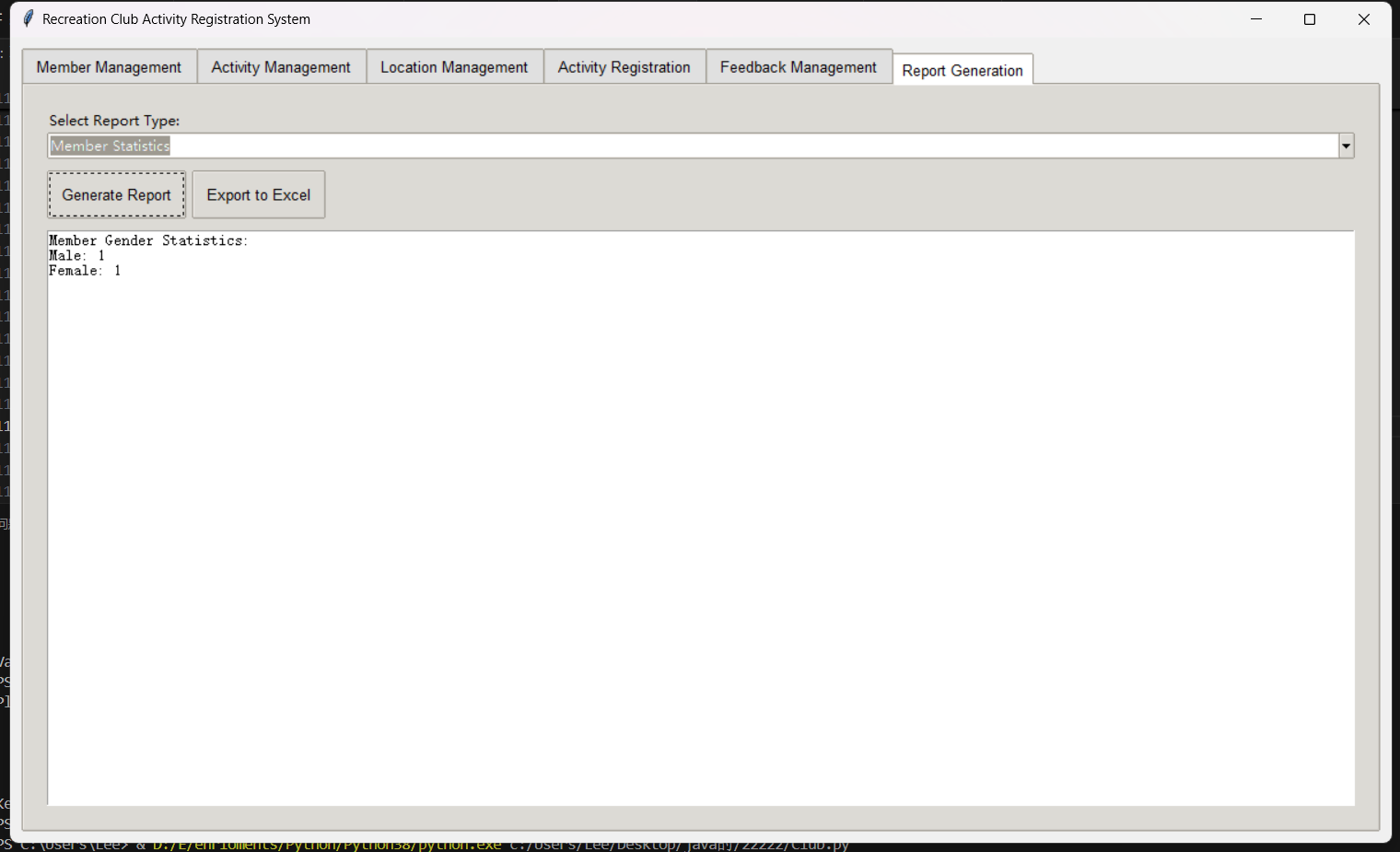
1. Aggregate query: Number of members by gender

* Objective: To analyze the gender distribution of club members and gain demographic insight.

SQl codes:

SELECT members\_gender, COUNT(\*) AS member\_count
  
FROM Members
  
GROUP BY members\_gender;

Project screenshot:



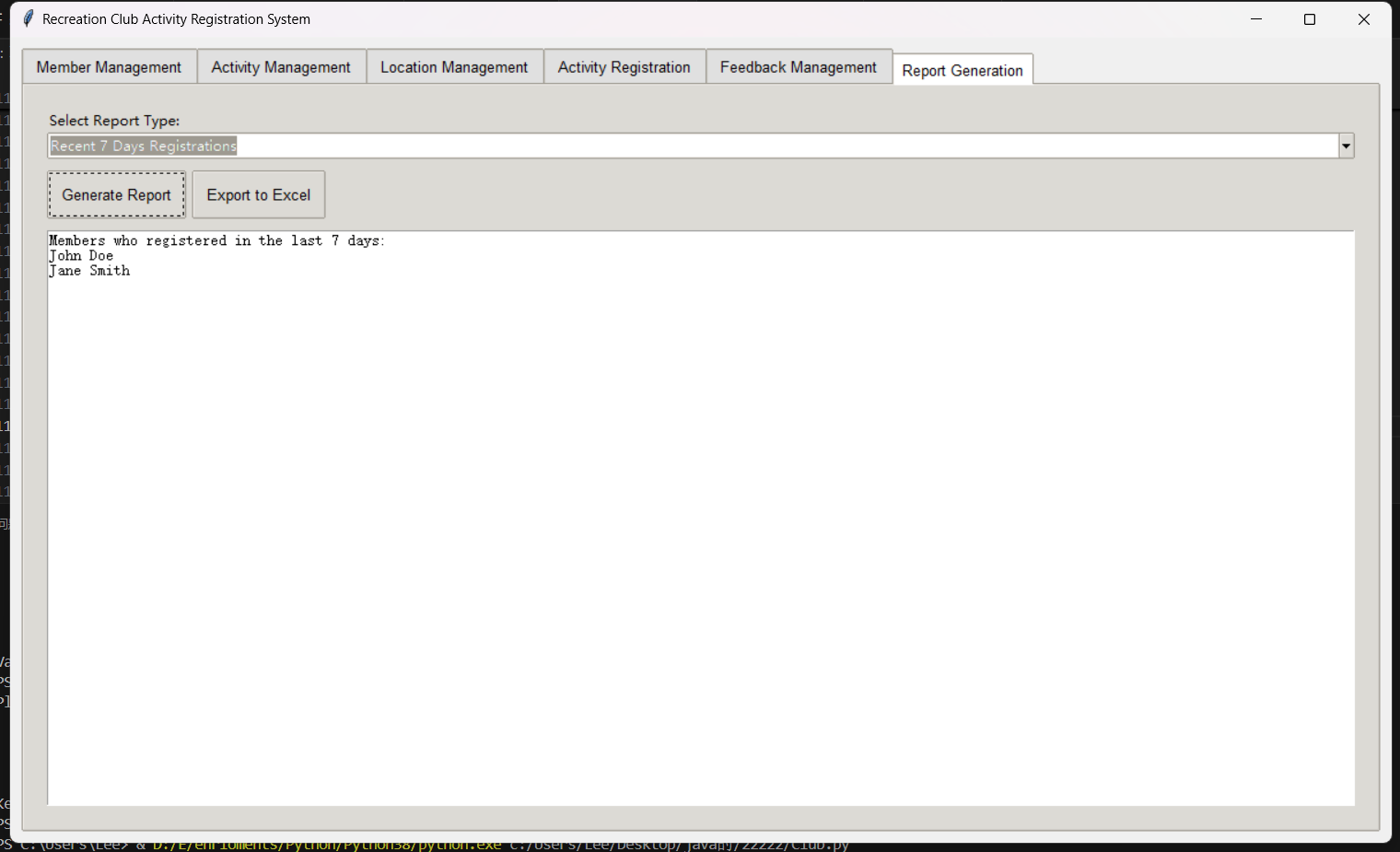
1. Subquery: Find members who have signed up for events in the last 7 days

* Purpose: To identify recently active members for interaction and follow-up.

SQl codes:

SELECT m.members\_name
  
FROM Members m
  
WHERE m.members\_id IN (
  
 SELECT s.members\_id
  
 FROM Signup s
  
 WHERE s.signup\_date >= CURDATE() - INTERVAL 7 DAY
  
);

Project screenshot:



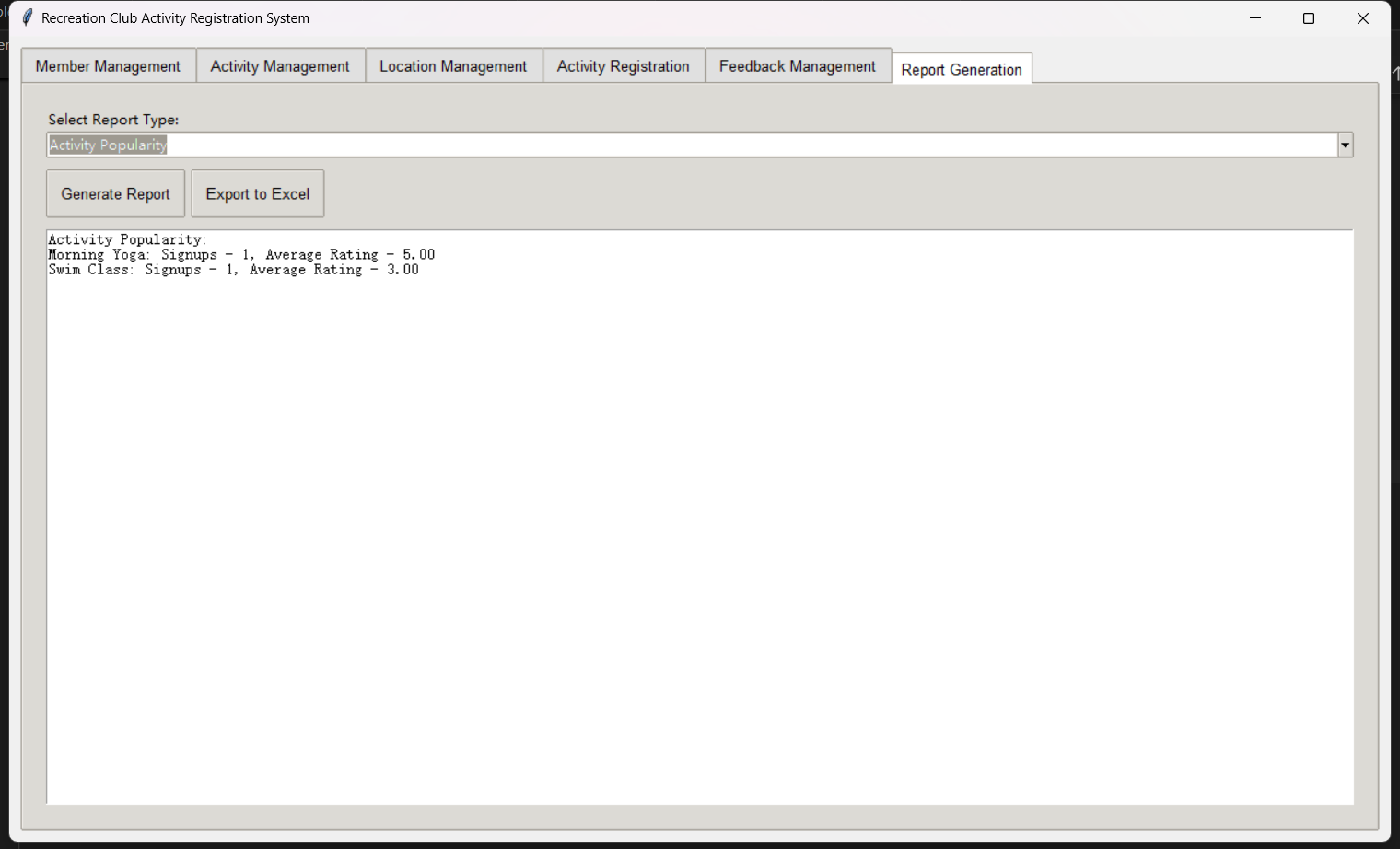
1. Multi-table Join: Get event details, including enrollment and average rating

* Objective: To assess the popularity and satisfaction of different activities.

SQl codes:

SELECT a.activity\_name,
  
 COUNT(DISTINCT s.members\_id) AS signup\_count,
  
 AVG(f.rating) AS avg\_rating
  
FROM Activity a
  
LEFT JOIN Signup s ON a.activity\_id = s.activity\_id
  
LEFT JOIN Feedback f ON a.activity\_id = f.activity\_id
  
GROUP BY a.activity\_id;

Project screenshot:



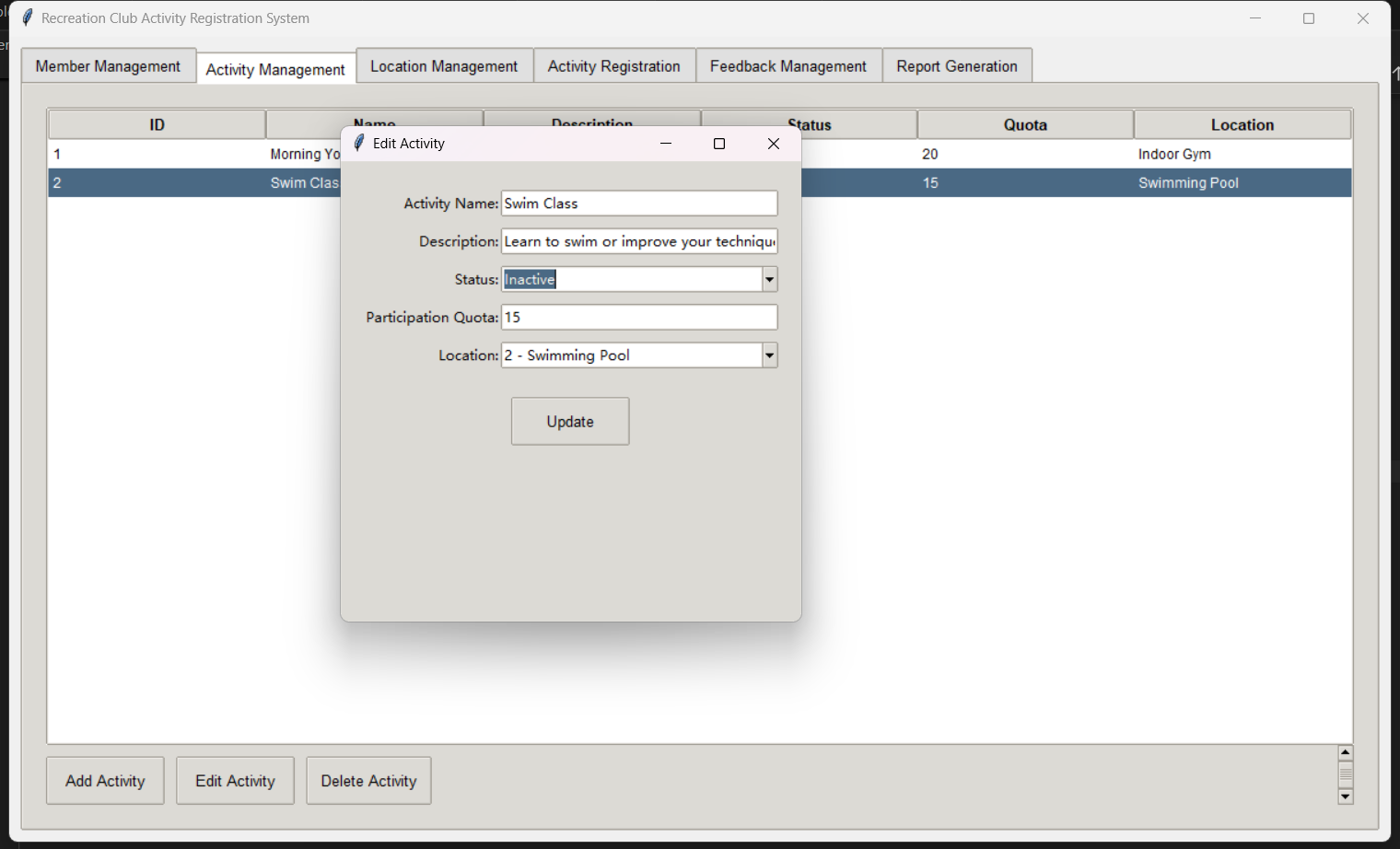
1. Update: Change unregistered event status to "Inactive"

* Purpose: Automatically manage inactive activities and keep activity lists relevant.

SQl codes:

UPDATE Activity a
  
SET a.activity\_status = 'Inactive'
  
WHERE a.activity\_id NOT IN (SELECT DISTINCT activity\_id FROM Signup);

Project screenshot:



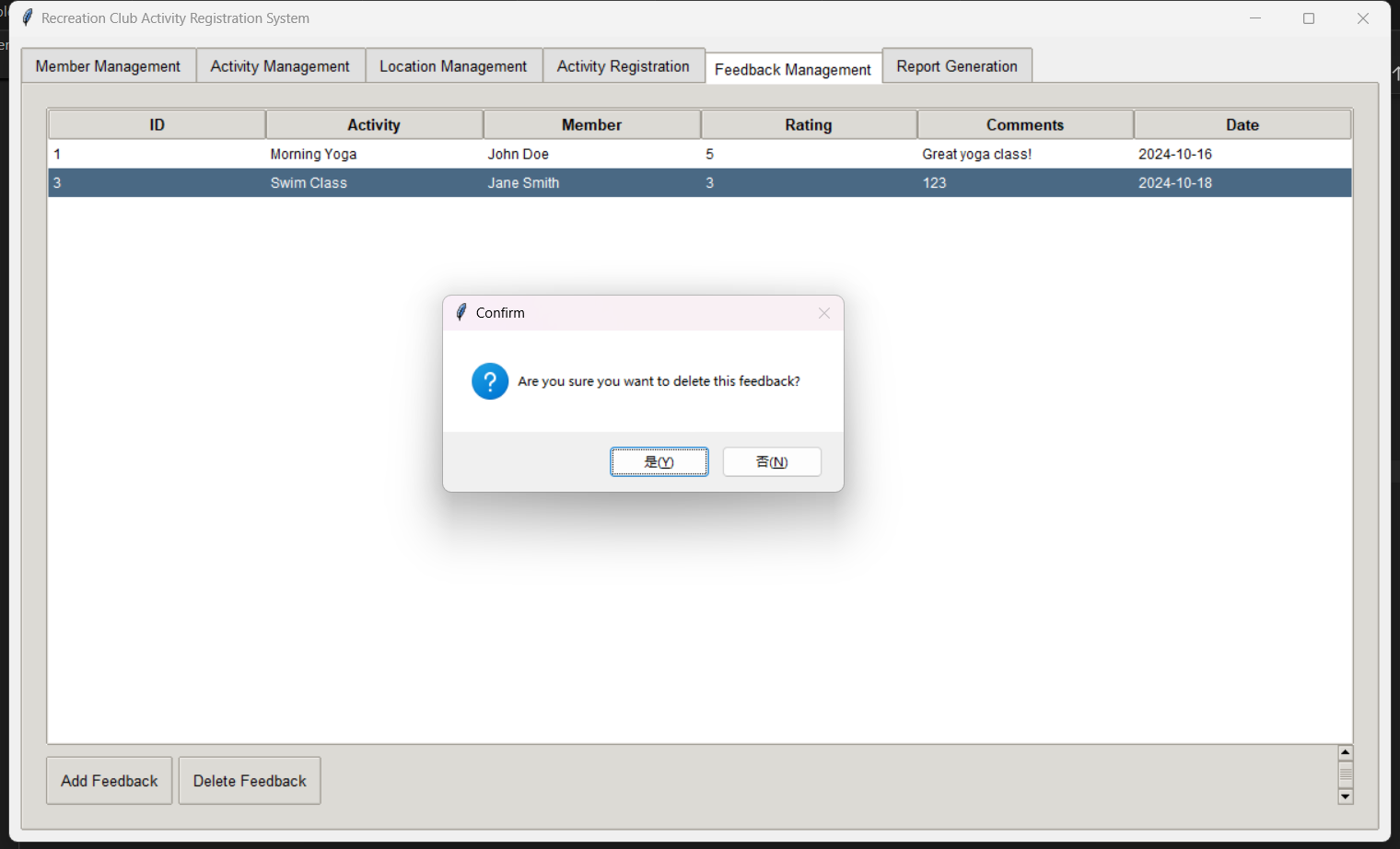
1. Delete: Delete old feedback that is more than one year old

* Objective: To maintain relevant and up-to-date feedback data in the system.

SQl codes:

DELETE FROM Feedback
  
WHERE feedback\_date < DATE\_SUB(CURDATE(), INTERVAL 1 YEAR);

Project screenshot:



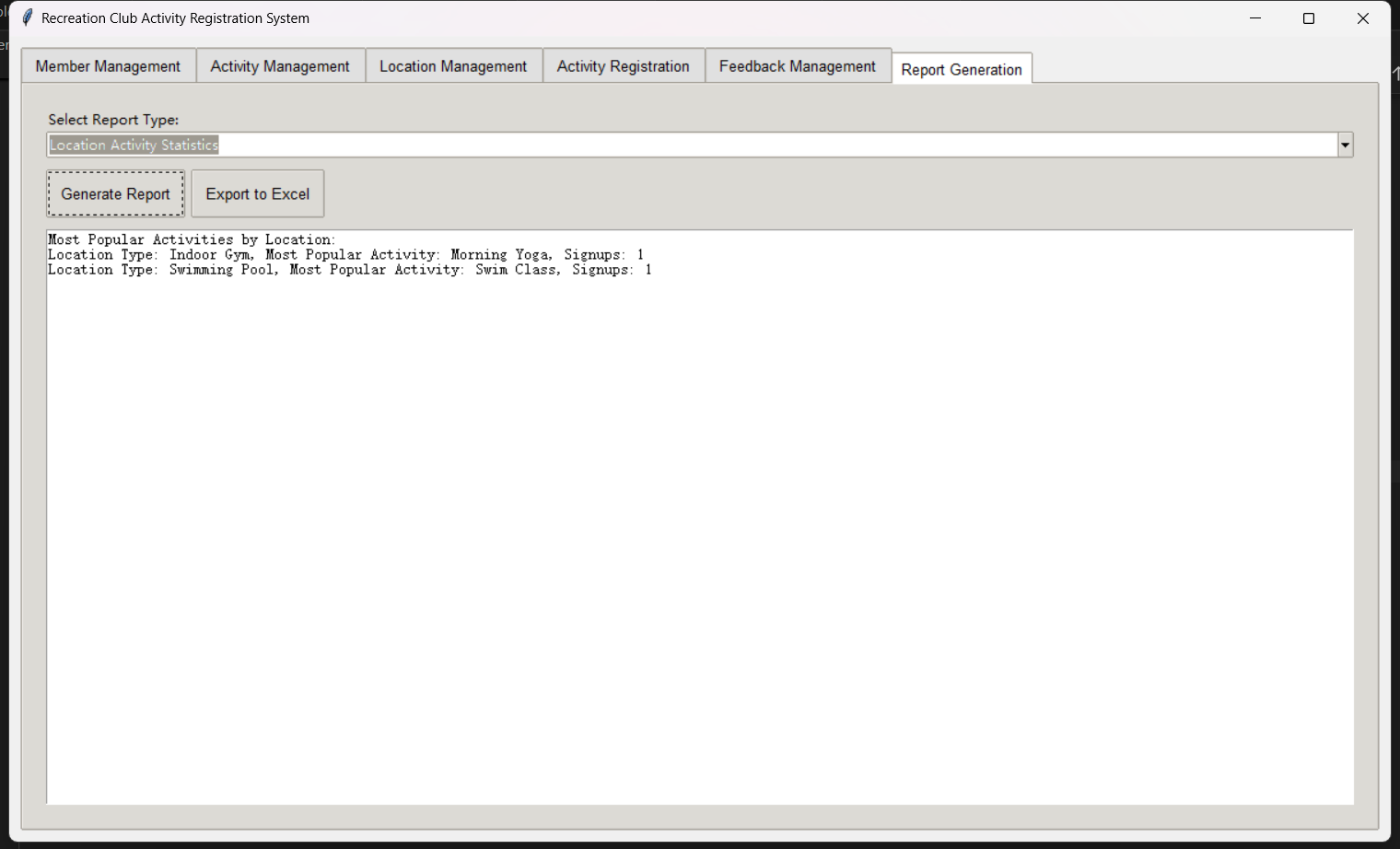
1. Complex query: Find out the most popular events at each venue (most registered)

* Purpose: To identify the most successful events at each venue and assist in event planning.

SQl codes:

SELECT l.type AS location\_type, a.activity\_name, signup\_count
  
FROM Location l
  
JOIN Activity a ON l.location\_id = a.location\_id
  
JOIN (
  
 SELECT activity\_id, COUNT(\*) AS signup\_count
  
 FROM Signup
  
 GROUP BY activity\_id
  
) s ON a.activity\_id = s.activity\_id
  
WHERE (l.location\_id, s.signup\_count) IN (
  
 SELECT a2.location\_id, MAX(s2.signup\_count)
  
 FROM Activity a2
  
 JOIN (
  
 SELECT activity\_id, COUNT(\*) AS signup\_count
  
 FROM Signup
  
 GROUP BY activity\_id
  
 ) s2 ON a2.activity\_id = s2.activity\_id
  
 GROUP BY a2.location\_id
  
);

Project screenshot:



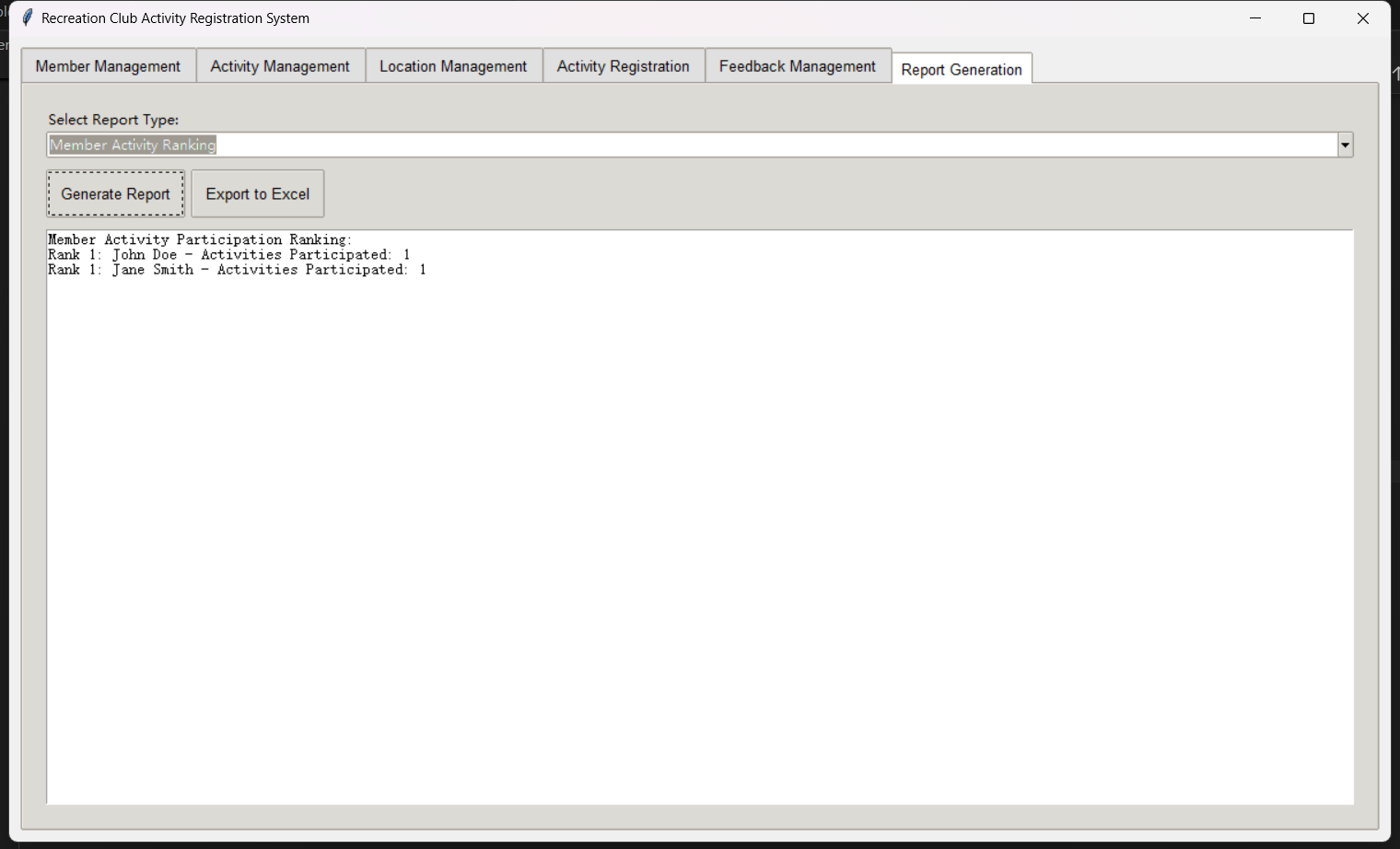
1. Analytical queries: Rank members based on the number of events they have signed up for

* Purpose: To identify and possibly reward the most active club members.

SQl codes:

SELECT m.members\_name,
  
 COUNT(s.activity\_id) AS activity\_count,
  
 RANK() OVER (ORDER BY COUNT(s.activity\_id) DESC) AS rank
  
FROM Members m
  
LEFT JOIN Signup s ON m.members\_id = s.members\_id
  
GROUP BY m.members\_id
  
ORDER BY rank;

Project screenshot:



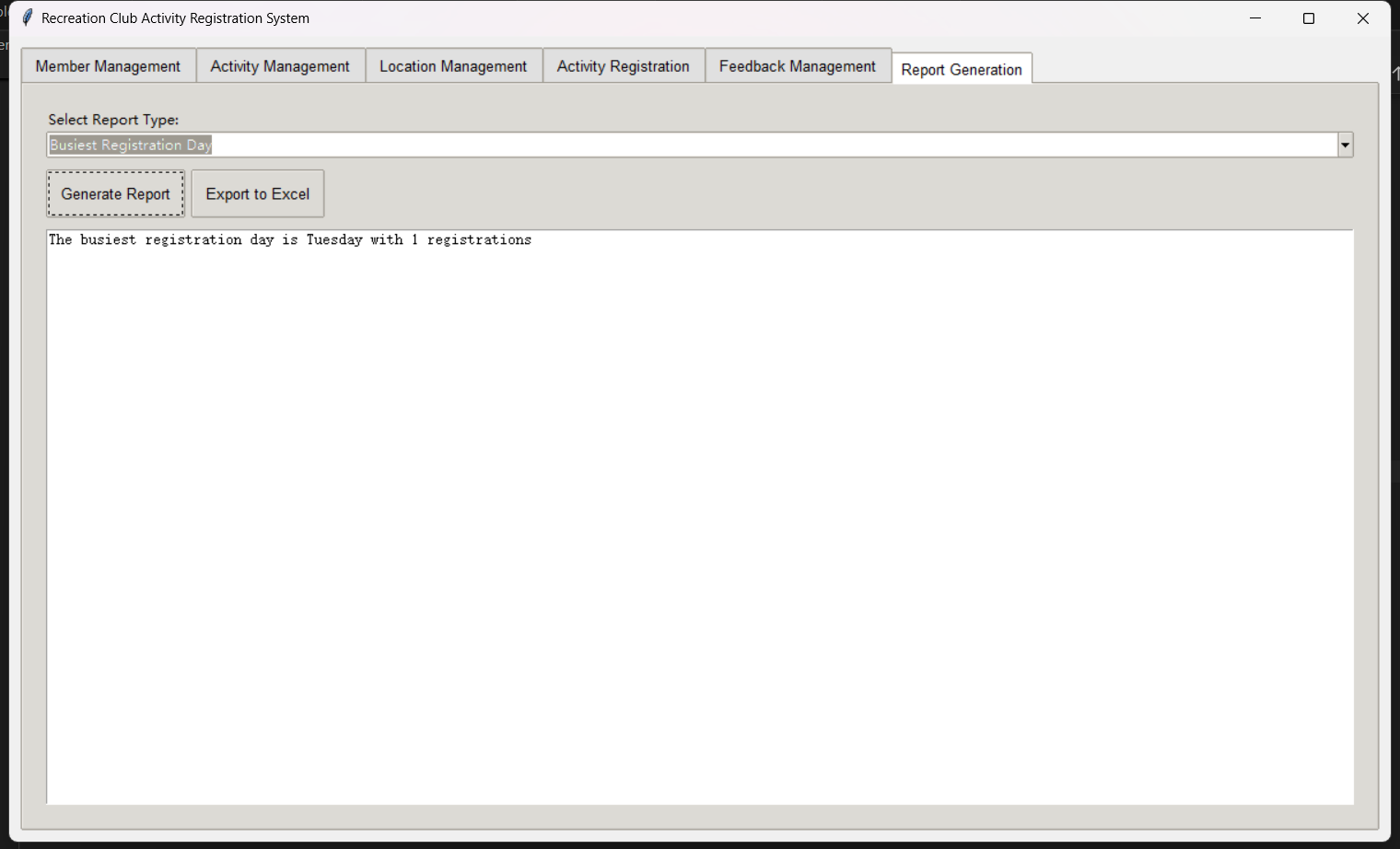
1. Time-based query: Get the busiest days of the week for enrollment

* Objective: Optimize staffing and resource allocation based on peak registration times.

SQl codes:

SELECT DAYNAME(signup\_date) AS day\_of\_week,
  
 COUNT(\*) AS signup\_count
  
FROM Signup
  
GROUP BY day\_of\_week
  
ORDER BY signup\_count DESC
  
LIMIT 1;

Project screenshot:



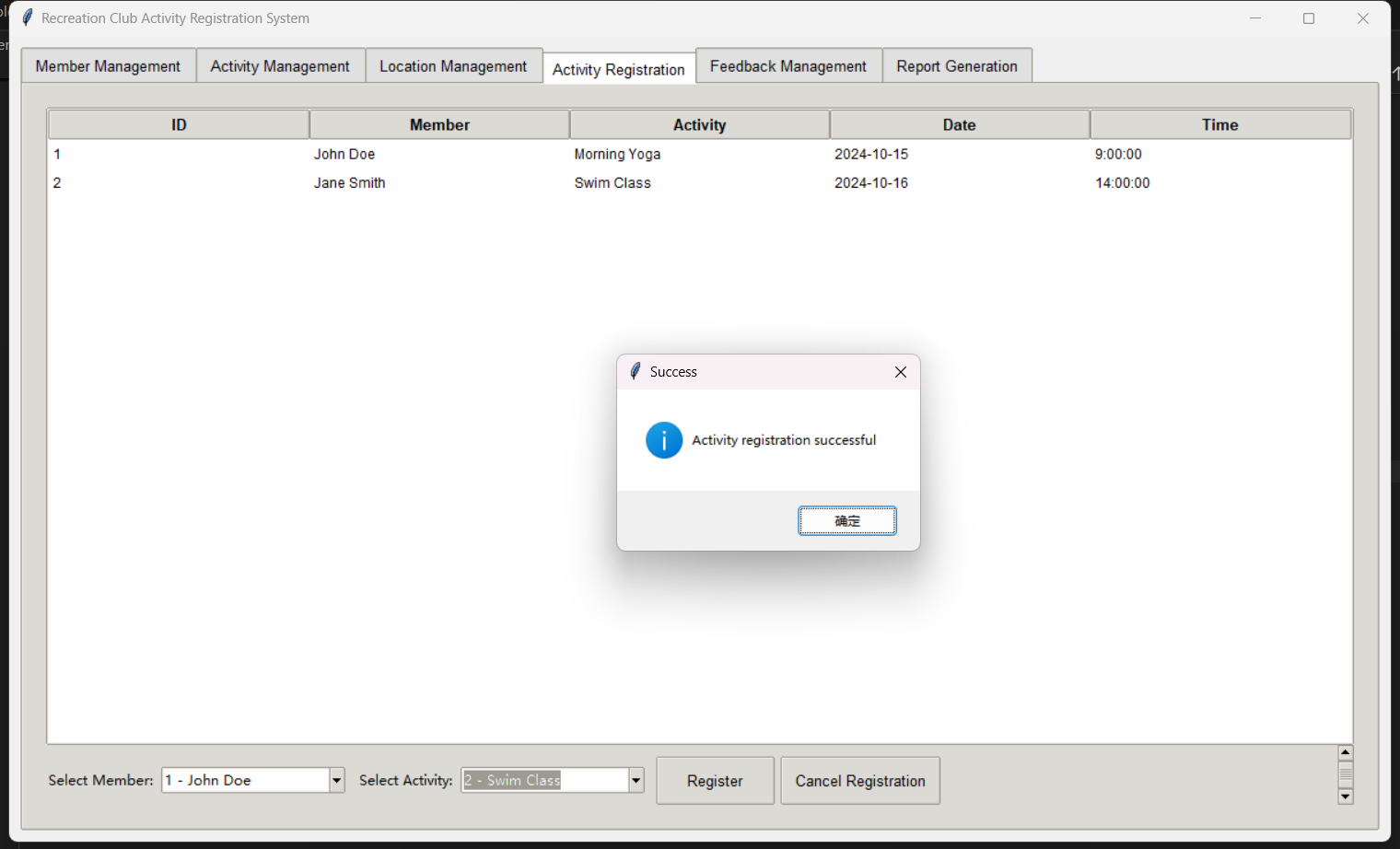
1. Transaction: Register the event for new members and update the participation quota

* Purpose: To ensure data consistency when registering members for events.

SQl codes:

INSERT INTO Signup (members\_id, activity\_id, signup\_date, signup\_time)
  
VALUES (1, 3, CURDATE(), CURTIME());
  
  
UPDATE Activity a
  
SET a.participation\_quota = a.participation\_quota - 1
  
WHERE a.activity\_id = 3
  
 AND a.participation\_quota > 0;

Project screenshot:



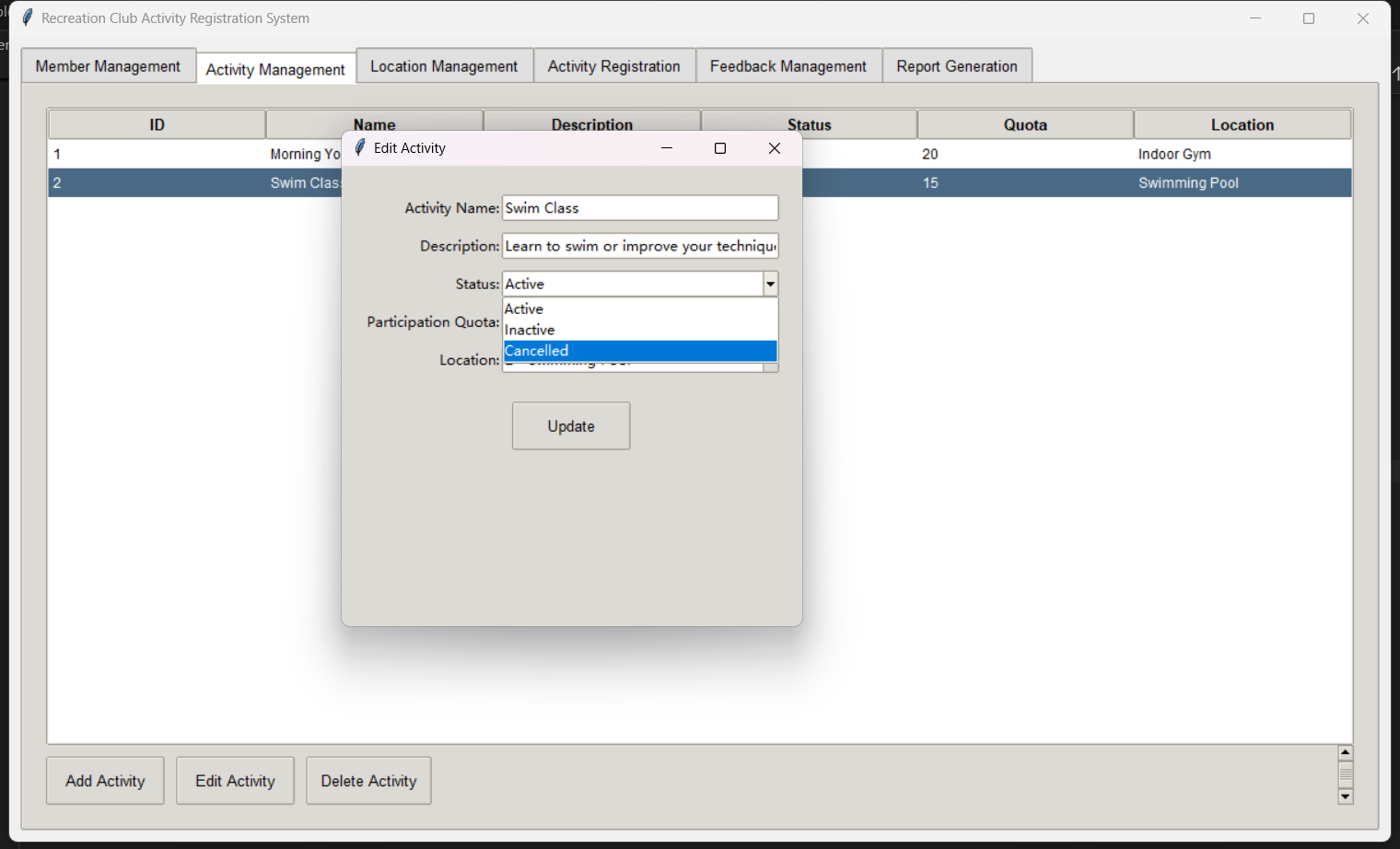
1. Cascading Delete: Cancel expired activities that have not been registered

* Objective: To clean the event database and free the reserved space.

SQl codes:

DELETE FROM Activity
  
WHERE activity\_id IN (
  
 SELECT a.activity\_id
  
 FROM (SELECT \* FROM Activity) AS a
  
 LEFT JOIN Signup s ON a.activity\_id = s.activity\_id
  
 WHERE s.signup\_id IS NULL
  
 AND a.activity\_status = 'Active'
  
 AND EXISTS (
  
 SELECT 1
  
 FROM Location l
  
 WHERE l.location\_id = a.location\_id
  
 AND l.availability = FALSE
  
 )
  
);
  
  
UPDATE Location l
  
SET l.availability = TRUE
  
WHERE l.location\_id NOT IN (
  
 SELECT DISTINCT location\_id
  
 FROM Activity
  
 WHERE activity\_status = 'Active'
  
);

Project screenshot:



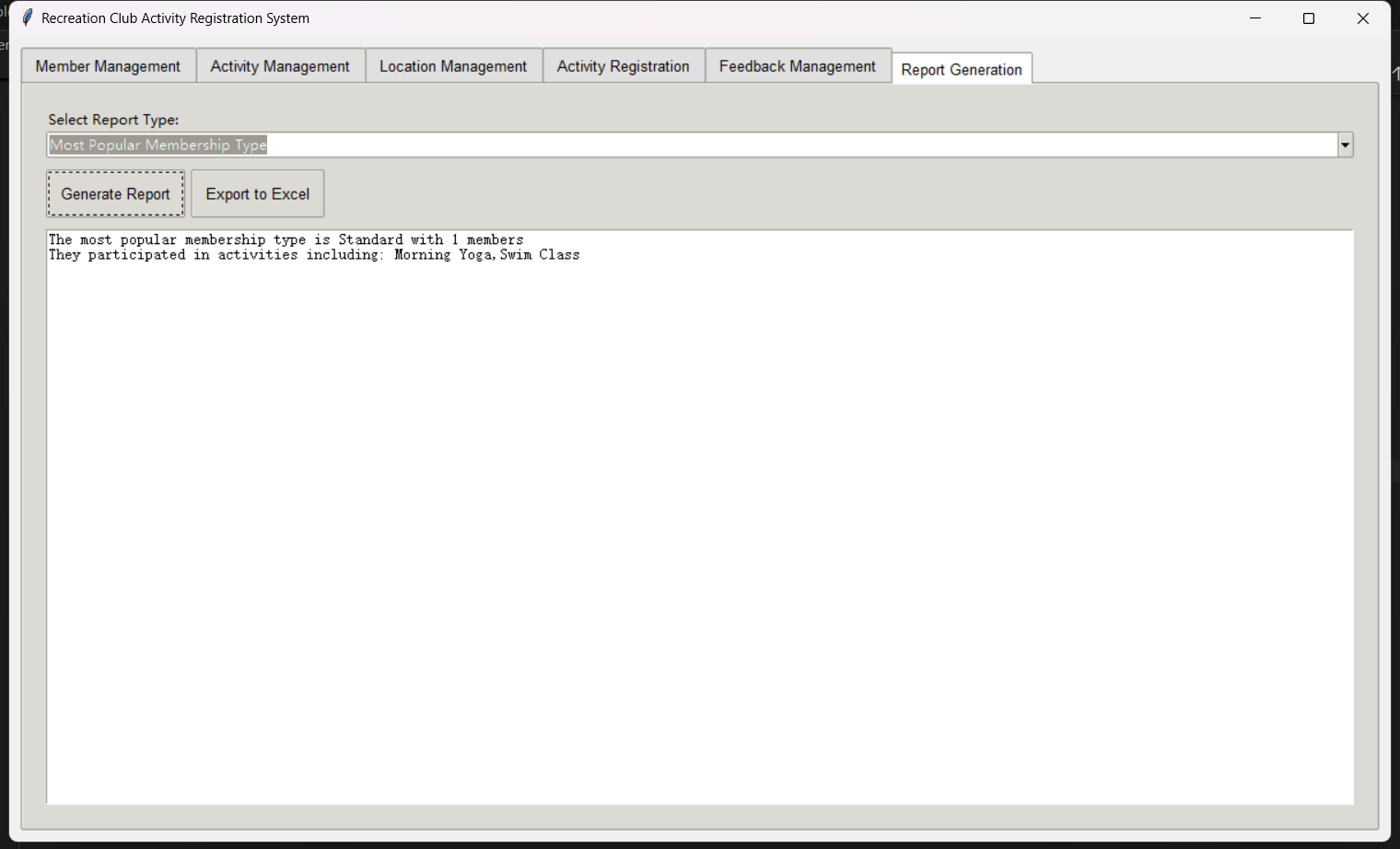
1. Complex join and aggregation: Identify the most popular types of membership and related activities

* Objective: To analyze the effects of different membership types and their effects on activity participation.

SQl codes:

SELECT mt.type\_name, COUNT(DISTINCT s.members\_id) as member\_count,
  
 GROUP\_CONCAT(DISTINCT a.activity\_name) as popular\_activities
  
FROM MembershipType mt
  
JOIN Members m ON mt.membership\_type\_id = m.membership\_type\_id
  
JOIN Signup s ON m.members\_id = s.members\_id
  
JOIN Activity a ON s.activity\_id = a.activity\_id
  
GROUP BY mt.membership\_type\_id
  
ORDER BY member\_count DESC
  
LIMIT 1;

Project screenshot:



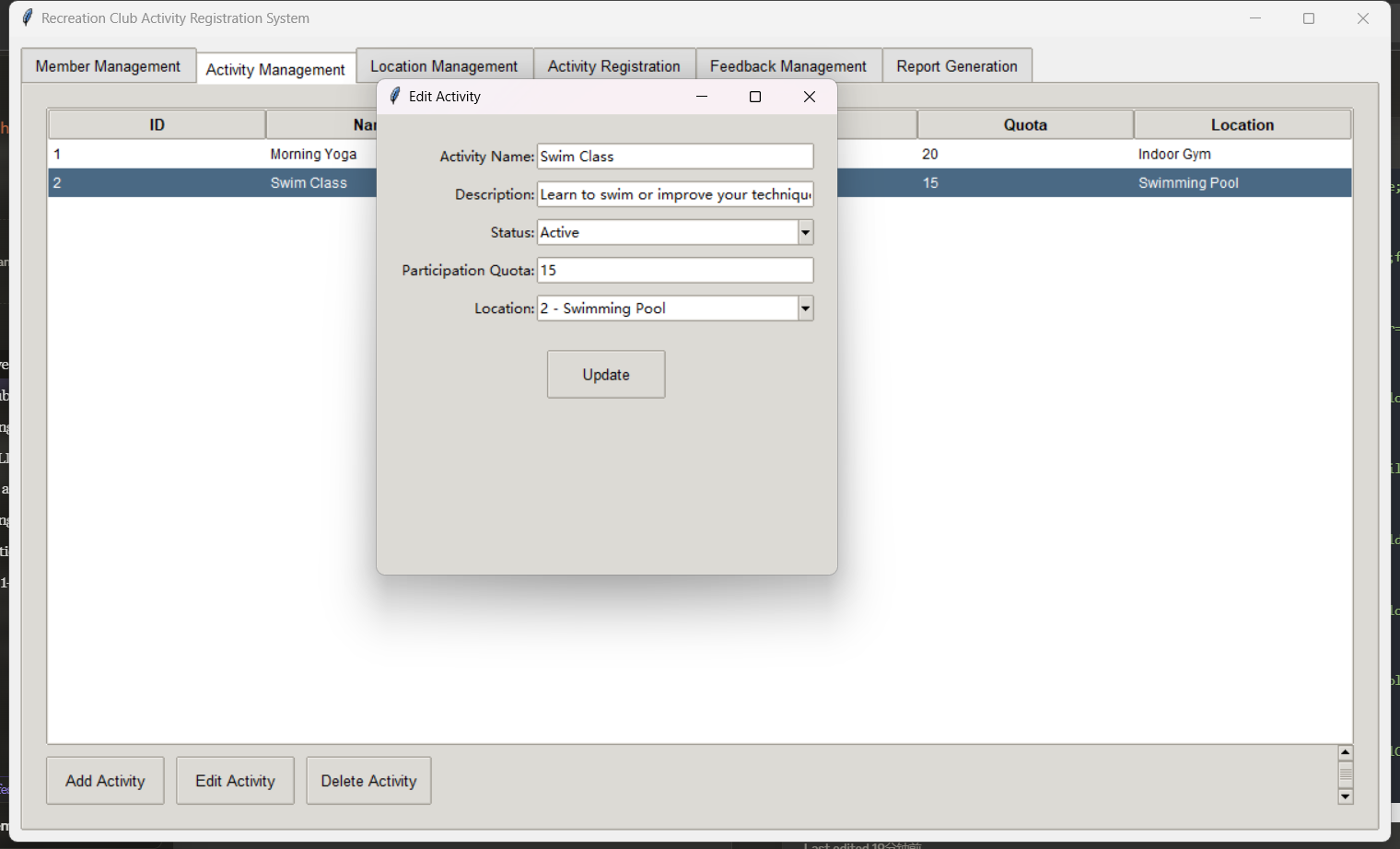
1. Updates based on aggregated data: Adjust activity status and description based on feedback

* Purpose: Automatically improve activity description and status based on member feedback.

SQl codes:

UPDATE Activity a
  
JOIN (
  
 SELECT activity\_id, AVG(rating) as avg\_rating
  
 FROM Feedback
  
 GROUP BY activity\_id
  
) f ON a.activity\_id = f.activity\_id
  
SET a.activity\_status = CASE
  
 WHEN f.avg\_rating < 2 THEN 'Inactive'
  
 WHEN f.avg\_rating >= 4 THEN 'Active'
  
 ELSE a.activity\_status
  
END,
  
a.description = CASE
  
 WHEN f.avg\_rating < 2 THEN CONCAT(a.description, ' (Currently under review based on feedback)')
  
 WHEN f.avg\_rating >= 4 THEN CONCAT(a.description, ' (Highly rated activity!)')
  
 ELSE a.description
  
END
  
WHERE f.avg\_rating IS NOT NULL;

Project screenshot:



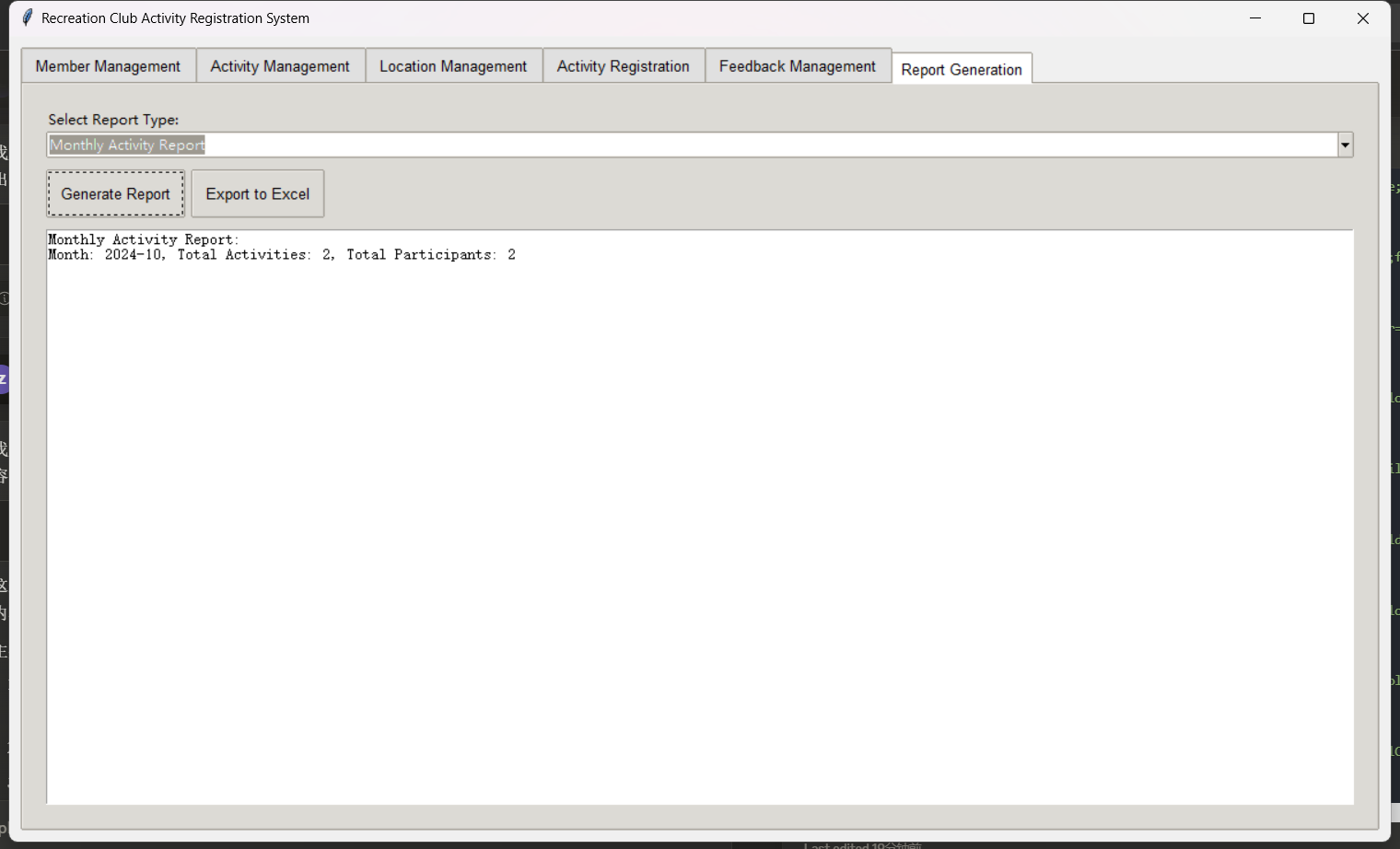
1. Comprehensive monthly report generation

* Purpose: To provide a detailed monthly overview of club activities, participation and member engagement.

SQl codes:

WITH monthly\_activity AS (
  
 SELECT
  
 DATE\_FORMAT(s.signup\_date, '%Y-%m') AS month,
  
 a.activity\_id,
  
 a.activity\_name,
  
 COUNT(DISTINCT s.members\_id) AS participant\_count
  
 FROM Activity a
  
 JOIN Signup s ON a.activity\_id = s.activity\_id
  
 GROUP BY month, a.activity\_id
  
),
  
top\_participants AS (
  
 SELECT
  
 DATE\_FORMAT(s.signup\_date, '%Y-%m') AS month,
  
 m.members\_name,
  
 COUNT(DISTINCT s.activity\_id) AS activity\_count,
  
 ROW\_NUMBER() OVER (PARTITION BY DATE\_FORMAT(s.signup\_date, '%Y-%m') ORDER BY COUNT(DISTINCT s.activity\_id) DESC) AS rank
  
 FROM Members m
  
 JOIN Signup s ON m.members\_id = s.members\_id
  
 GROUP BY month, m.members\_id
  
)
  
SELECT
  
 ma.month,
  
 COUNT(DISTINCT ma.activity\_id) AS total\_activities,
  
 SUM(ma.participant\_count) AS total\_participants,
  
 GROUP\_CONCAT(DISTINCT CONCAT(ma.activity\_name, ': ', ma.participant\_count) ORDER BY ma.participant\_count DESC SEPARATOR '; ') AS activity\_breakdown,
  
 GROUP\_CONCAT(DISTINCT CONCAT(tp.members\_name, ' (', tp.activity\_count, ' activities)') ORDER BY tp.activity\_count DESC SEPARATOR ', ') AS top\_3\_participants
  
FROM monthly\_activity ma
  
LEFT JOIN top\_participants tp ON ma.month = tp.month AND tp.rank <= 3
  
GROUP BY ma.month
  
ORDER BY ma.month DESC;

Project screenshot:



## Conclusion

The "Entertainment Club Activity Registration System" project successfully implemented a comprehensive solution for managing club events, member registration and feedback. The system provides a user-friendly interface for club administrators and streamlines various processes such as event management, membership enrollment and report generation.

**The main achievements of the project include:**

* Efficient data management through well-designed relational databases
* Use Tkinter for user-friendly graphical interface
* Comprehensive reporting and analysis capabilities
* Flexible event and venue management system
* Integrated feedback mechanism for continuous improvement

**Suggestions for further development:**

1. Implement Web-based interface to facilitate members' access
2. Integrated payment system for payment of membership fees and event fees
3. Develop mobile apps to access club activities and register anytime and anywhere
4. Implement an automatic recommendation system based on member preferences and past activities
5. Enhance reporting systems with more advanced analytics and data visualization tools