Entertainment Club Activity Registration System Project Report

1 Project launch document

1.1 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.

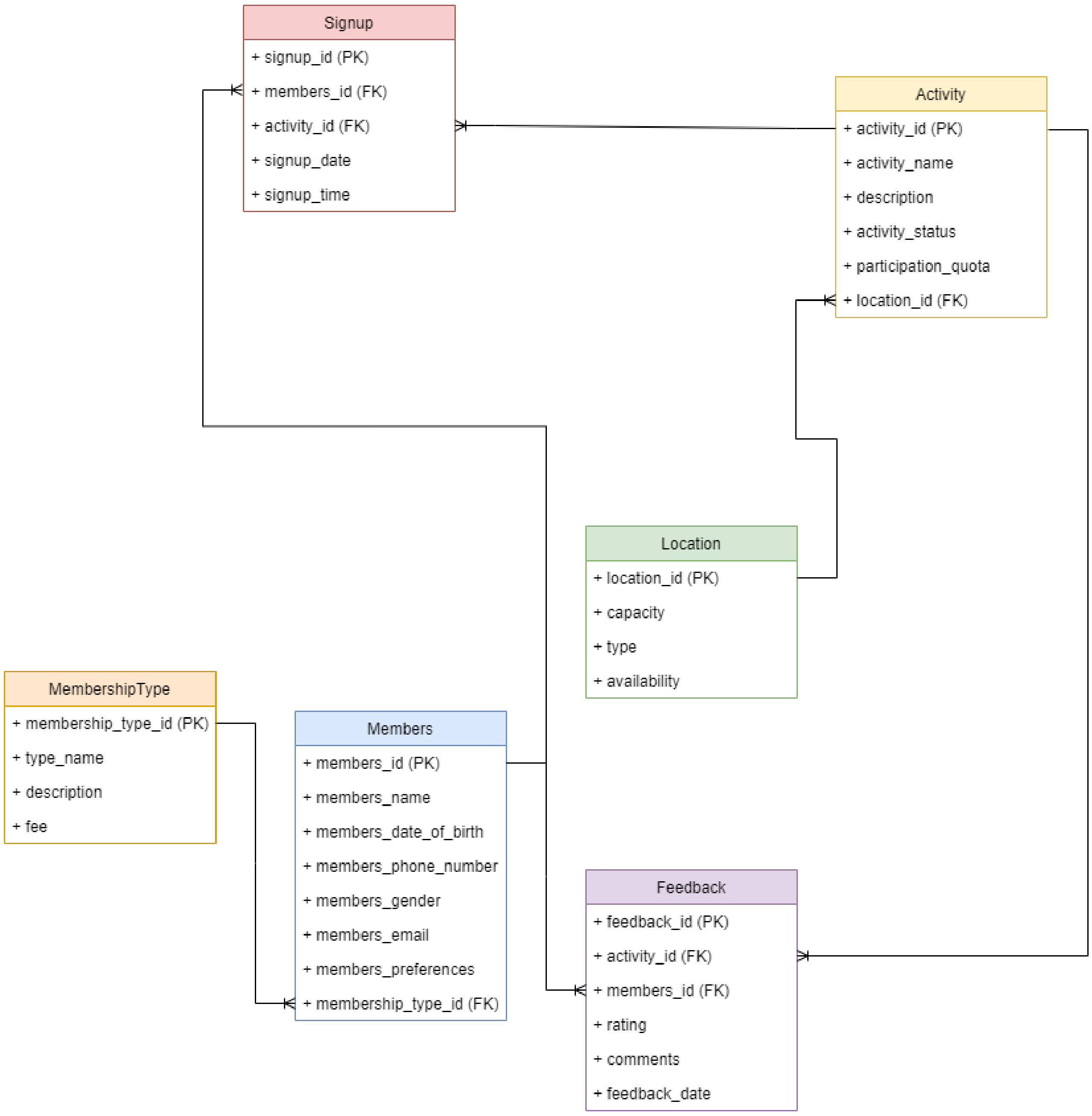
Second, the system significantly increases member engagement through personalized activity recommendations, making it easier for members to find activities that match their interests. At the same time, the system has improved the way events are organized and feedback is collected, which not only increases 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.

1.2 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

2 Design specification

2.0.1 E-R diagram：



2.0.2 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

2.1 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.

2.2 Data Dictionary

2.2.1 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

2.2.2 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

2.2.3 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

2.2.4 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

2.2.5 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

2.2.6 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

3 Configuration specification

SQL statement configuration and specification:

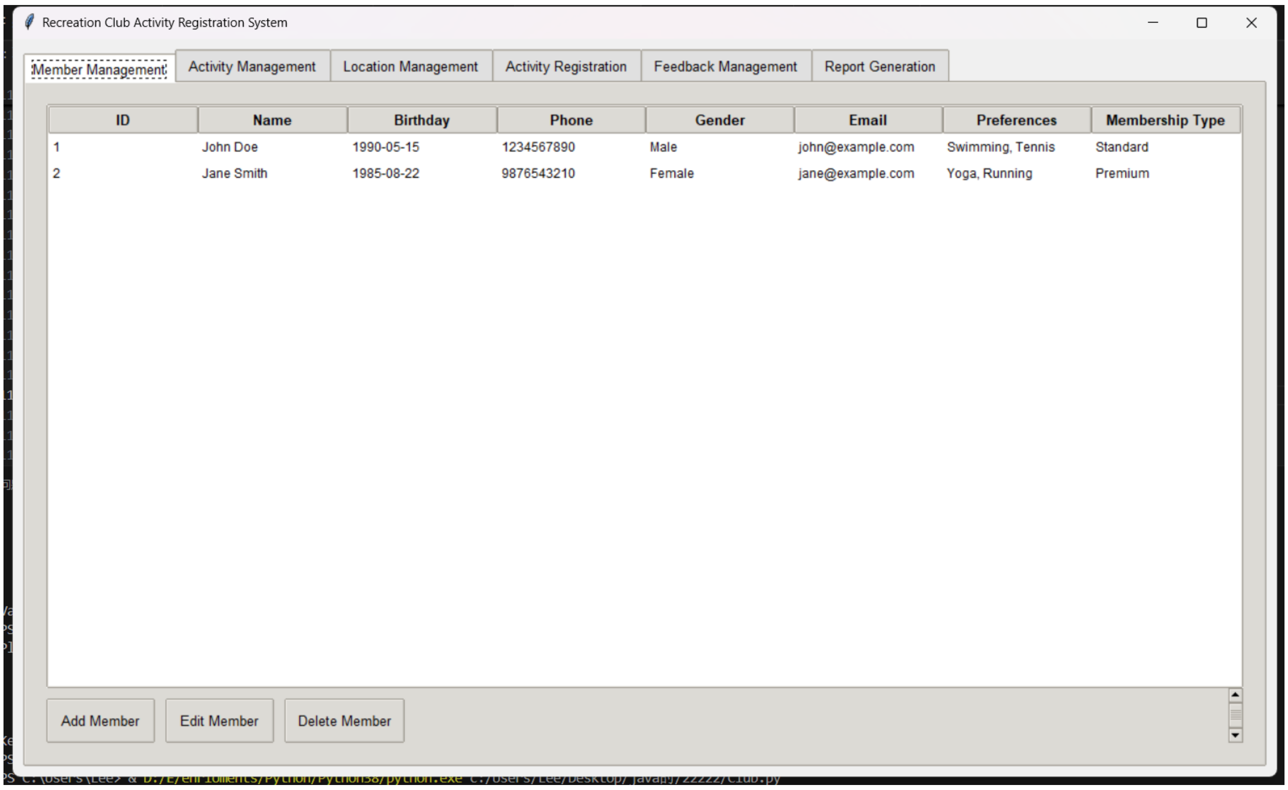
1. Basic query: List all members

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

SQl codes:

```sql
SELECT * FROM Members;
```

Project screenshot:



ID	Name	Birthday	Phone	Gender	Email	Preferences	Membership Type
1	John Doe	1990-05-15	1234567890	Male	john@example.com	Swimming, Tennis	Standard
2	Jane Smith	1985-08-22	9876543210	Female	jane@example.com	Yoga, Running	Premium

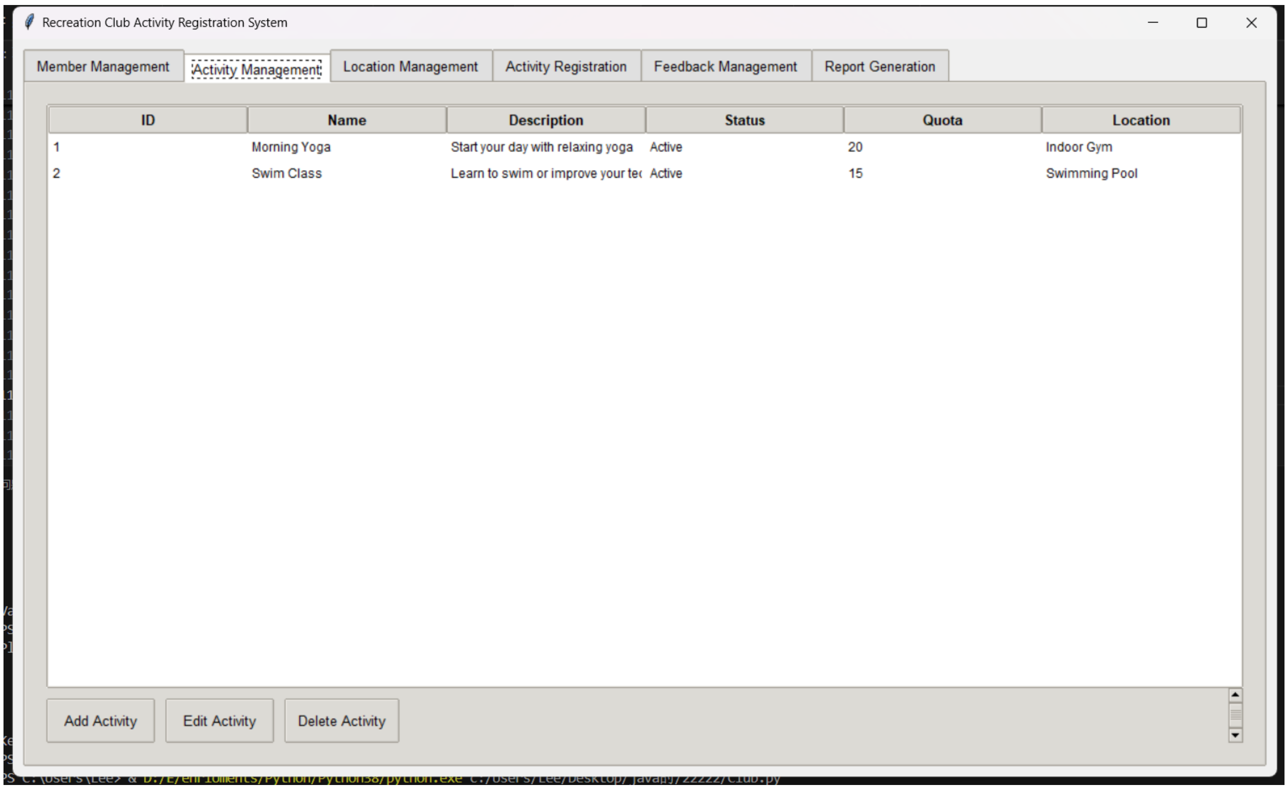
2. Join Query: Get all events and their venues

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

SQl codes:

```sql
SELECT a.activity_name, l.type AS location_type
FROM Activity a
JOIN Location l ON a.location_id = l.location_id;
```

Project screenshot:



Recreation Club Activity Registration System

ID	Name	Description	Status	Quota	Location
1	Morning Yoga	Start your day with relaxing yoga	Active	20	Indoor Gym
2	Swim Class	Learn to swim or improve your tec	Active	15	Swimming Pool

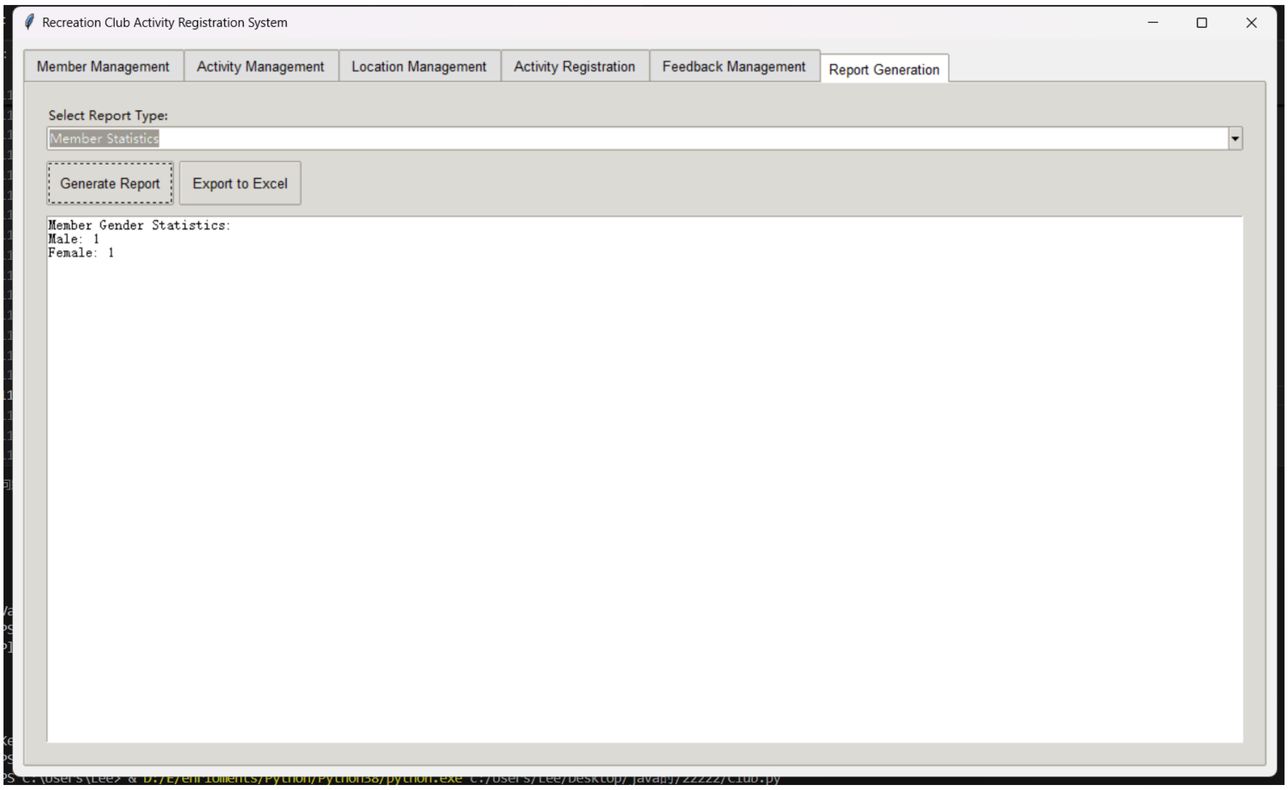
3. Aggregate query: Number of members by gender

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

SQl codes:

```sql
SELECT members_gender, COUNT(*) AS member_count
FROM Members
GROUP BY members_gender;
```

Project screenshot:



Recreation Club Activity Registration System

Select Report Type: Member Statistics

```
Member Gender Statistics:
Male: 1
Female: 1
```

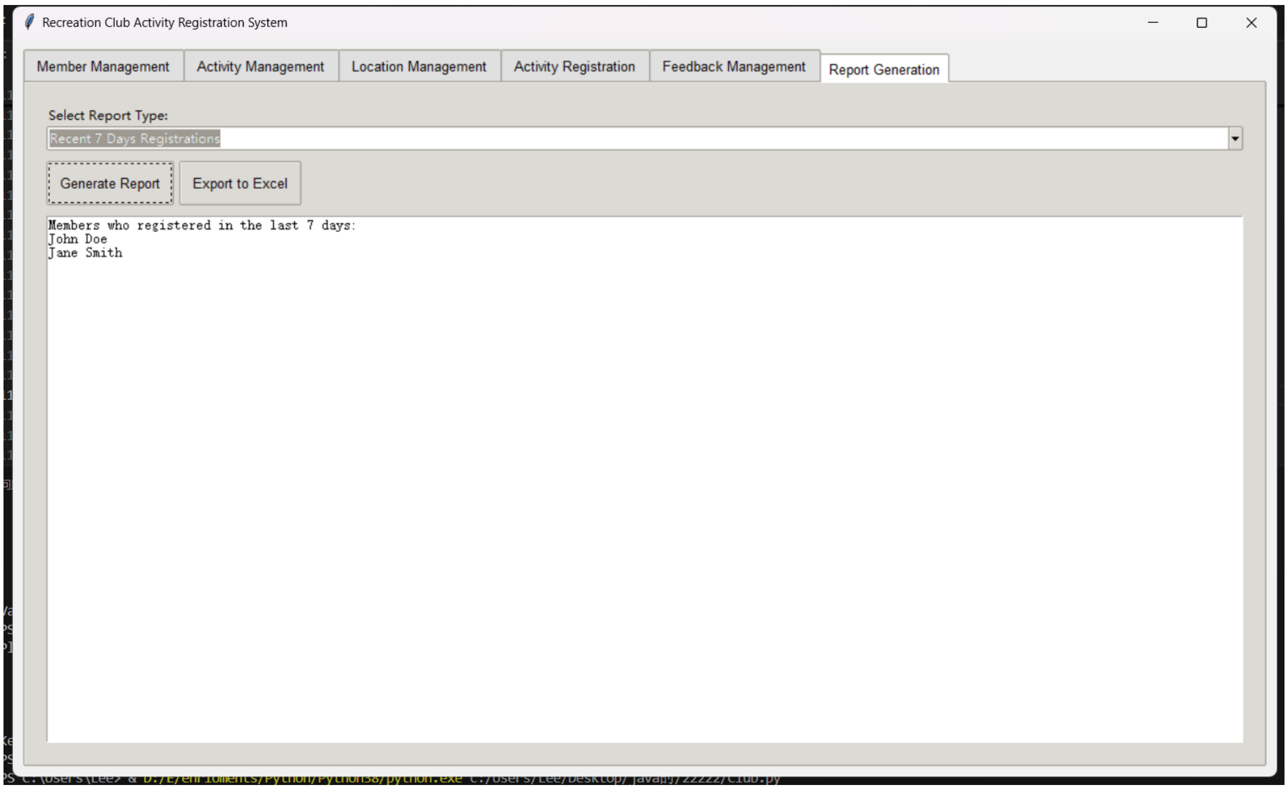
4. Subquery: Find members who have signed up for events in the last 7 days

   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:



Select Report Type: Recent 7 Days Registrations

```
Members who registered in the last 7 days:
John Doe
Jane Smith
```

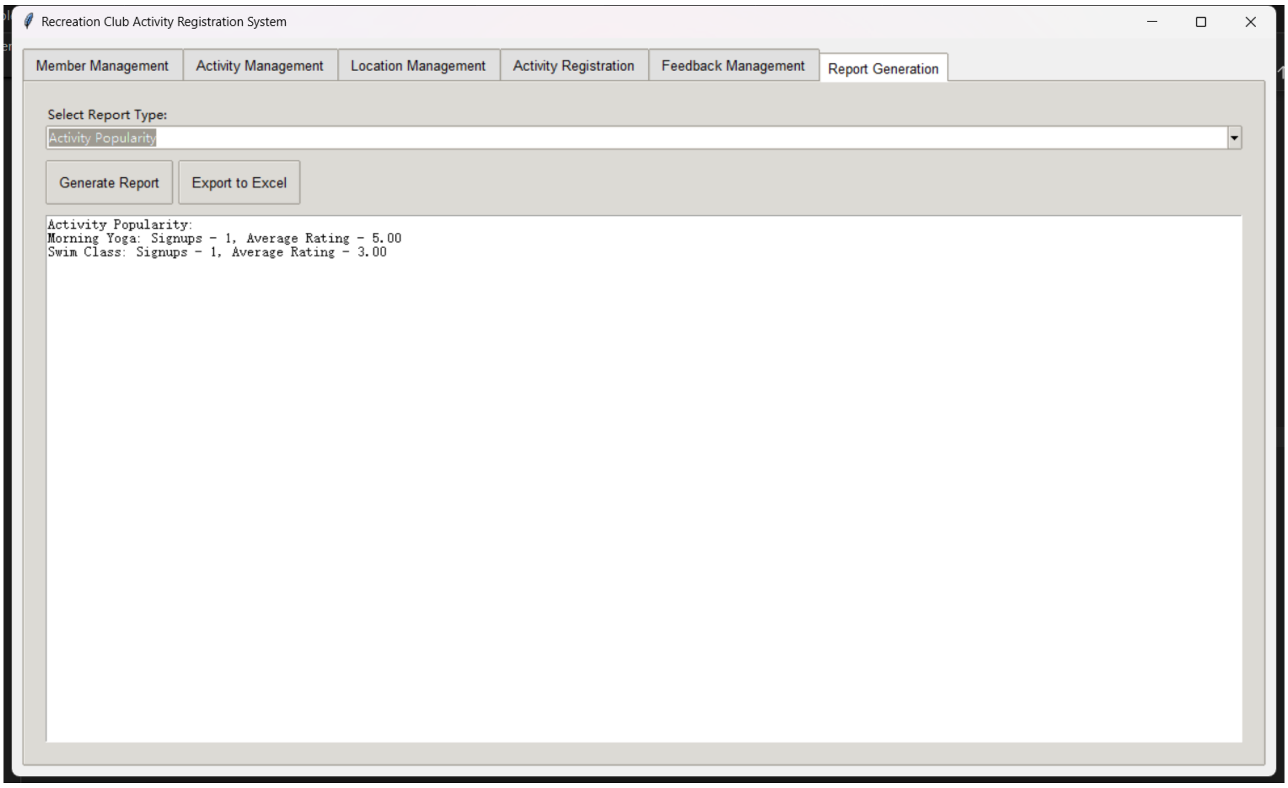
5. Multi-table Join: Get event details, including enrollment and average rating

   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:



```
Activity Popularity:
Morning Yoga: Signups - 1, Average Rating - 5.00
Swim Class: Signups - 1, Average Rating - 3.00
```

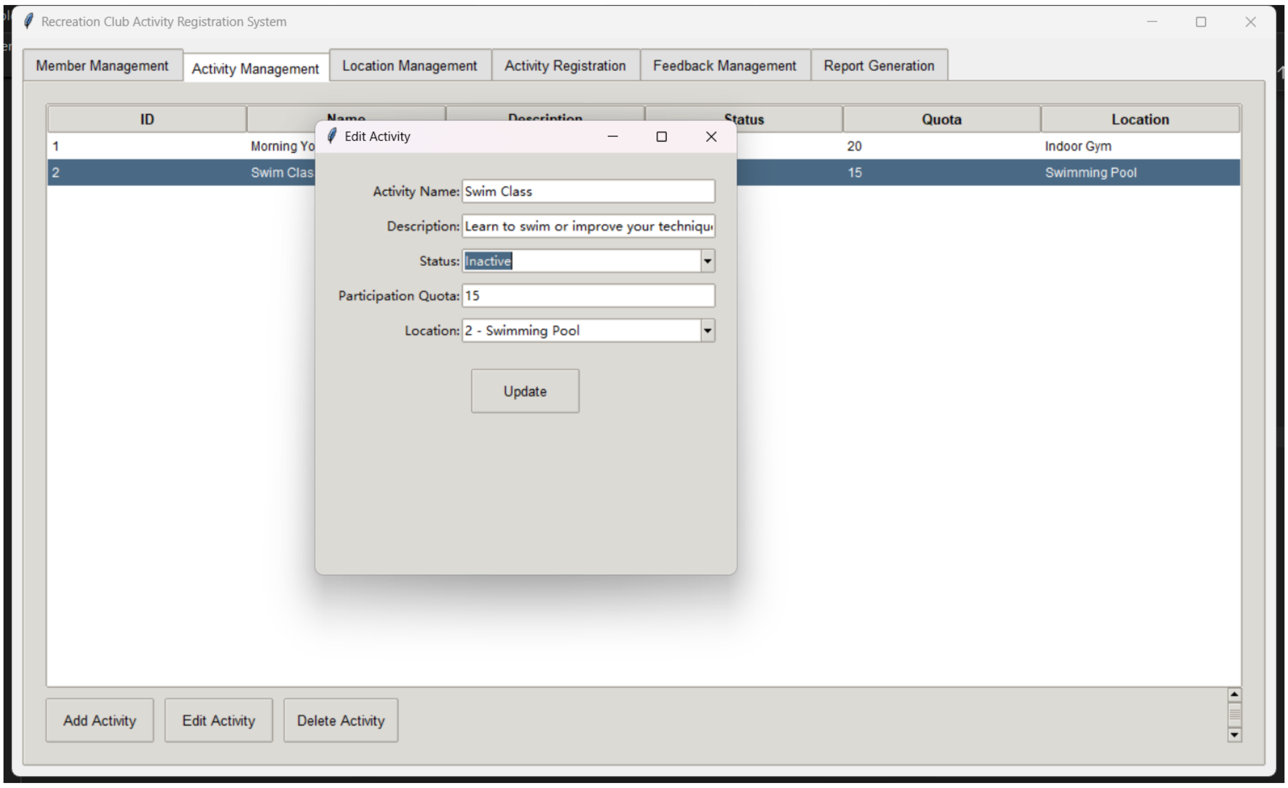
6. Update: Change unregistered event status to "Inactive"

   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:



ID	Name	Description	Status	Quota	Location
1	Morning Yo			20	Indoor Gym
2	Swim Clas			15	Swimming Pool

Activity Name: Swim Class
Description: Learn to swim or improve your techniqu
Status: Inactive
Participation Quota: 15
Location: 2 - Swimming Pool

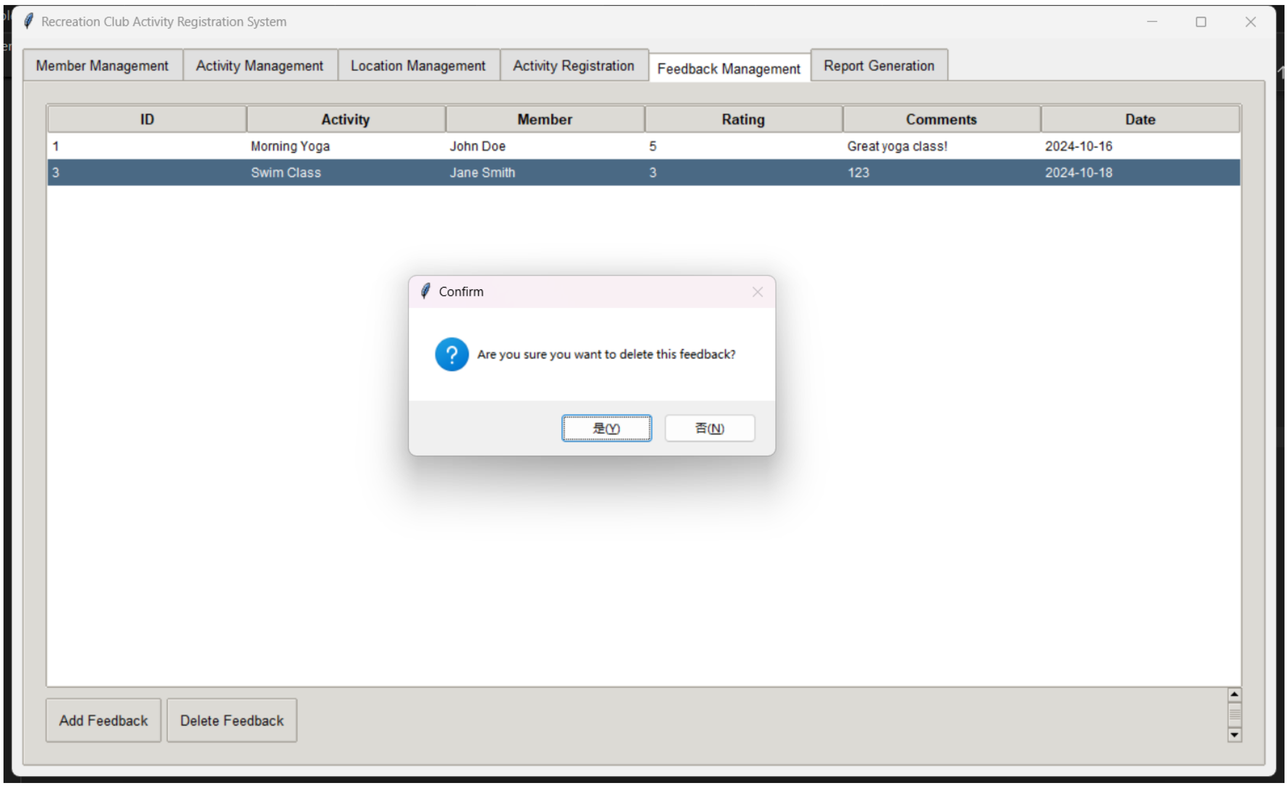
7. Delete: Delete old feedback that is more than one year old

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

SQl codes:

```sql
DELETE FROM Feedback
WHERE feedback_date < DATE_SUB(CURDATE(), INTERVAL 1 YEAR);
```

Project screenshot:



Recreation Club Activity Registration System

Member Management | Activity Management | Location Management | Activity Registration | Feedback Management | Report Generation

ID	Activity	Member	Rating	Comments	Date
1	Morning Yoga	John Doe	5	Great yoga class!	2024-10-16
3	Swim Class	Jane Smith	3	123	2024-10-18

Confirm

Are you sure you want to delete this feedback?

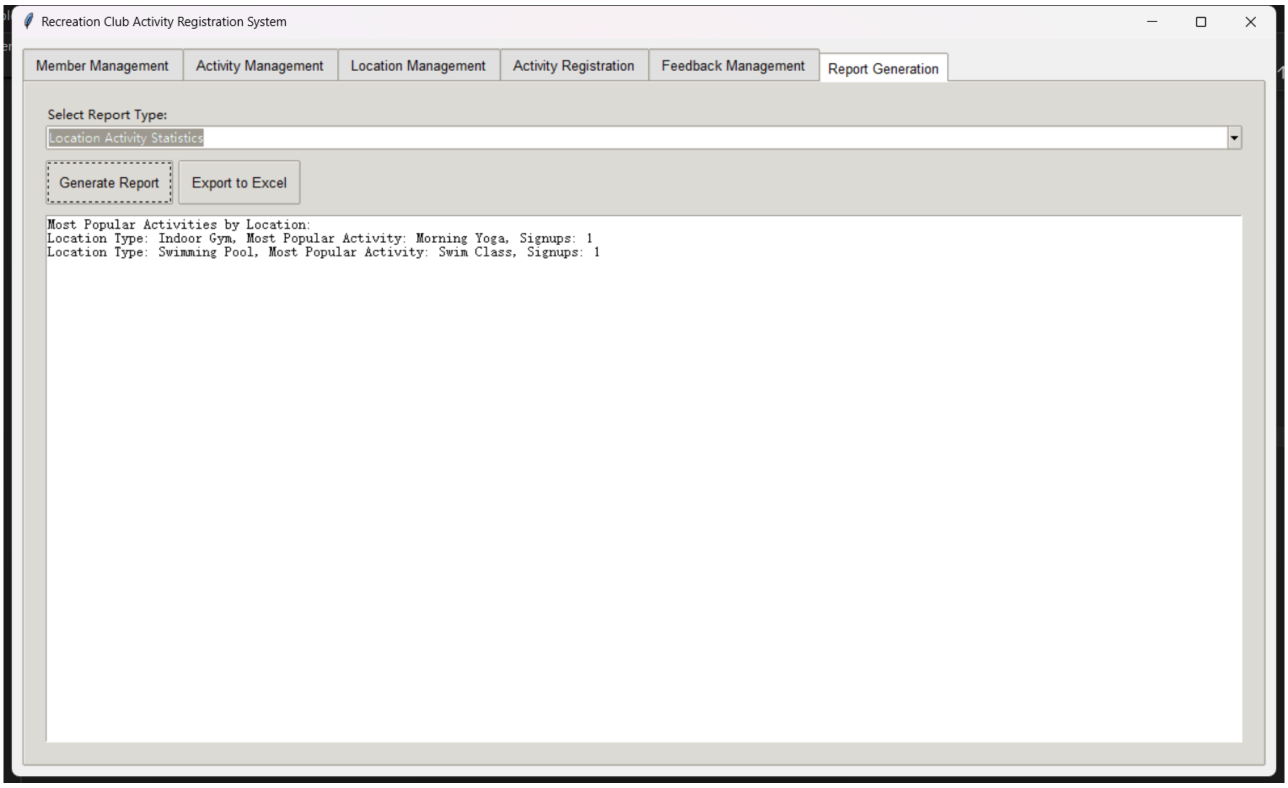
8. Complex query: Find out the most popular events at each venue (most registered)

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

SQl codes:

```sql
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:



Select Report Type: Location Activity Statistics

```
Most Popular Activities by Location:
Location Type: Indoor Gym, Most Popular Activity: Morning Yoga, Signups: 1
Location Type: Swimming Pool, Most Popular Activity: Swim Class, Signups: 1
```

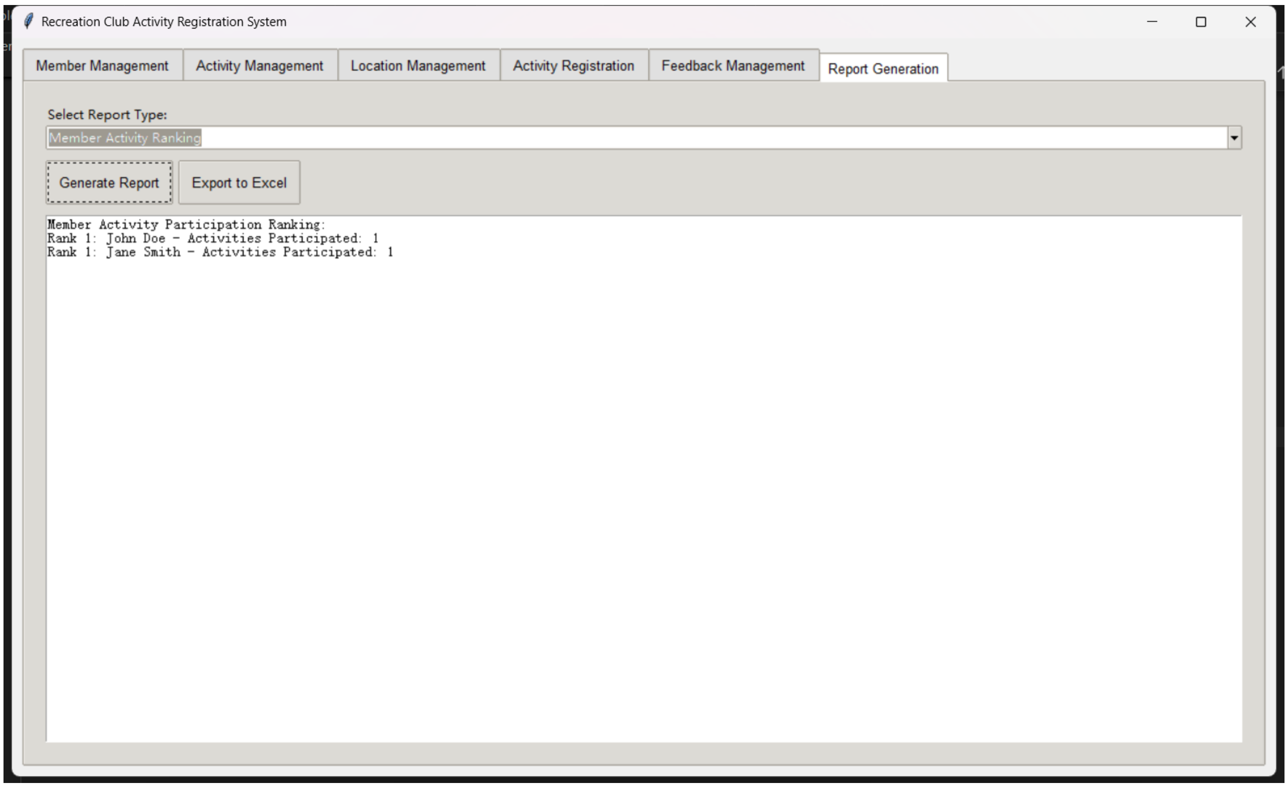
9. Analytical queries: Rank members based on the number of events they have signed up for

   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:



```
Member Activity Participation Ranking:
Rank 1: John Doe - Activities Participated: 1
Rank 1: Jane Smith - Activities Participated: 1
```

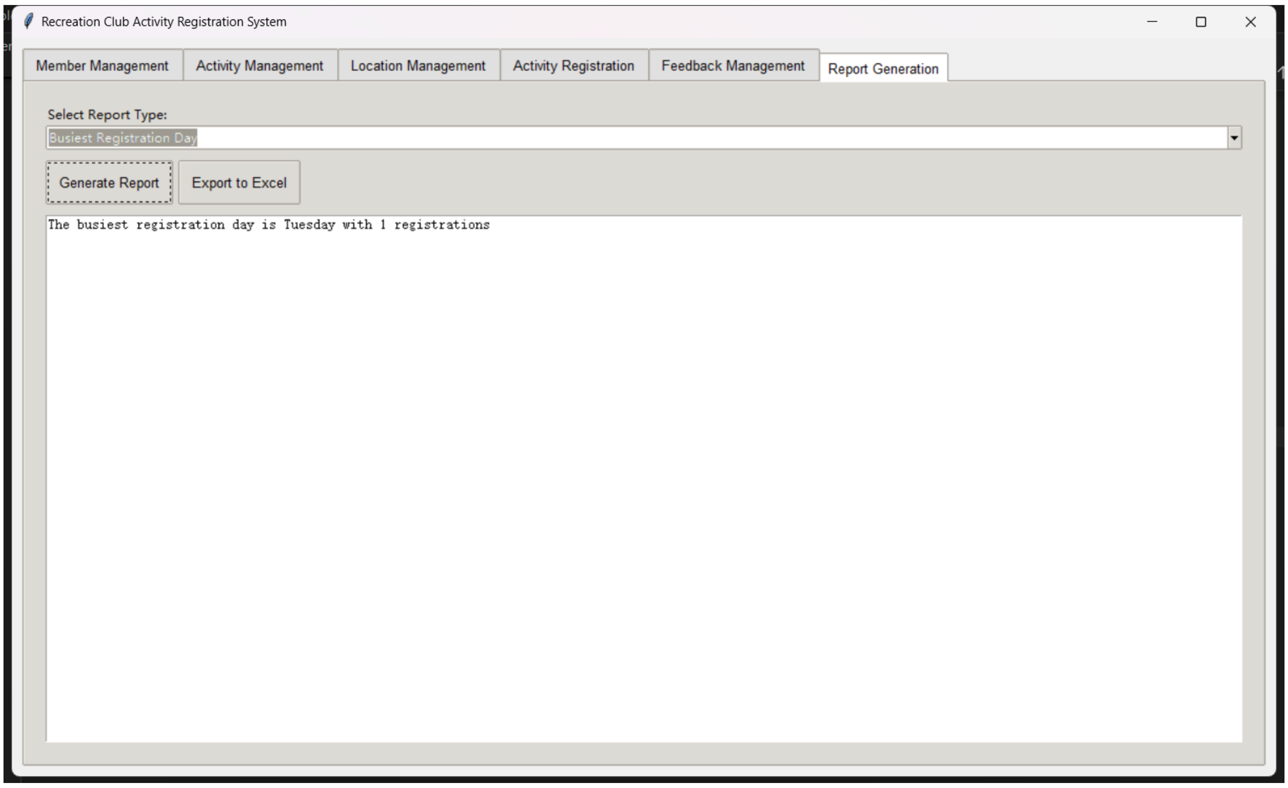
10. Time-based query: Get the busiest days of the week for enrollment

    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:



Select Report Type: Busiest Registration Day

```
The busiest registration day is Tuesday with 1 registrations
```

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

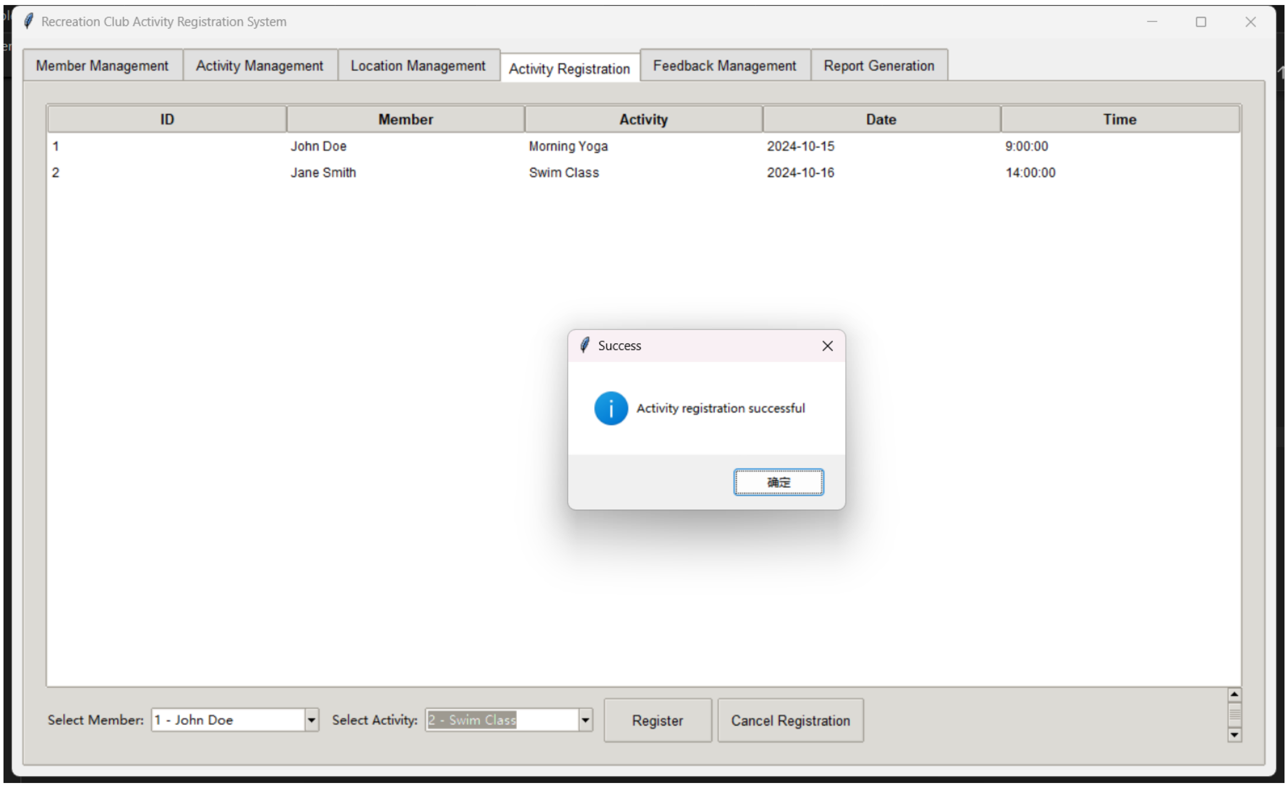
    Purpose: To ensure data consistency when registering members for events.

SQl codes:

```sql
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:



ID	Member	Activity	Date	Time
1	John Doe	Morning Yoga	2024-10-15	9:00:00
2	Jane Smith	Swim Class	2024-10-16	14:00:00

Success

Activity registration successful

Select Member: 1 - John Doe Select Activity: 2 - Swim Class

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

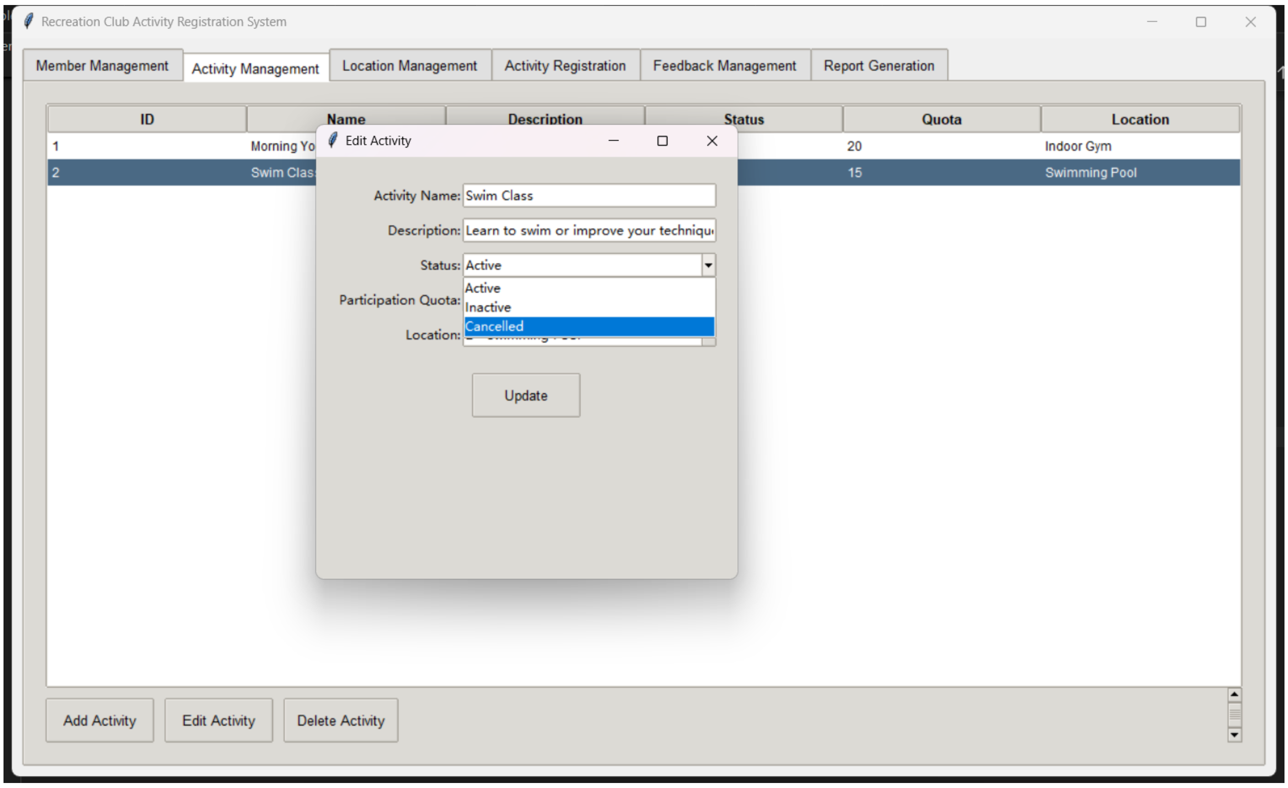
Objective: To clean the event database and free the reserved space.

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:


Recreation Club Activity Registration System

ID	Name	Description	Status	Quota	Location
1	Morning Yo			20	Indoor Gym
2	Swim Clas			15	Swimming Pool

Edit Activity

Activity Name: Swim Class

Description: Learn to swim or improve your techniqu

Status: Active

Participation Quota:

Location:

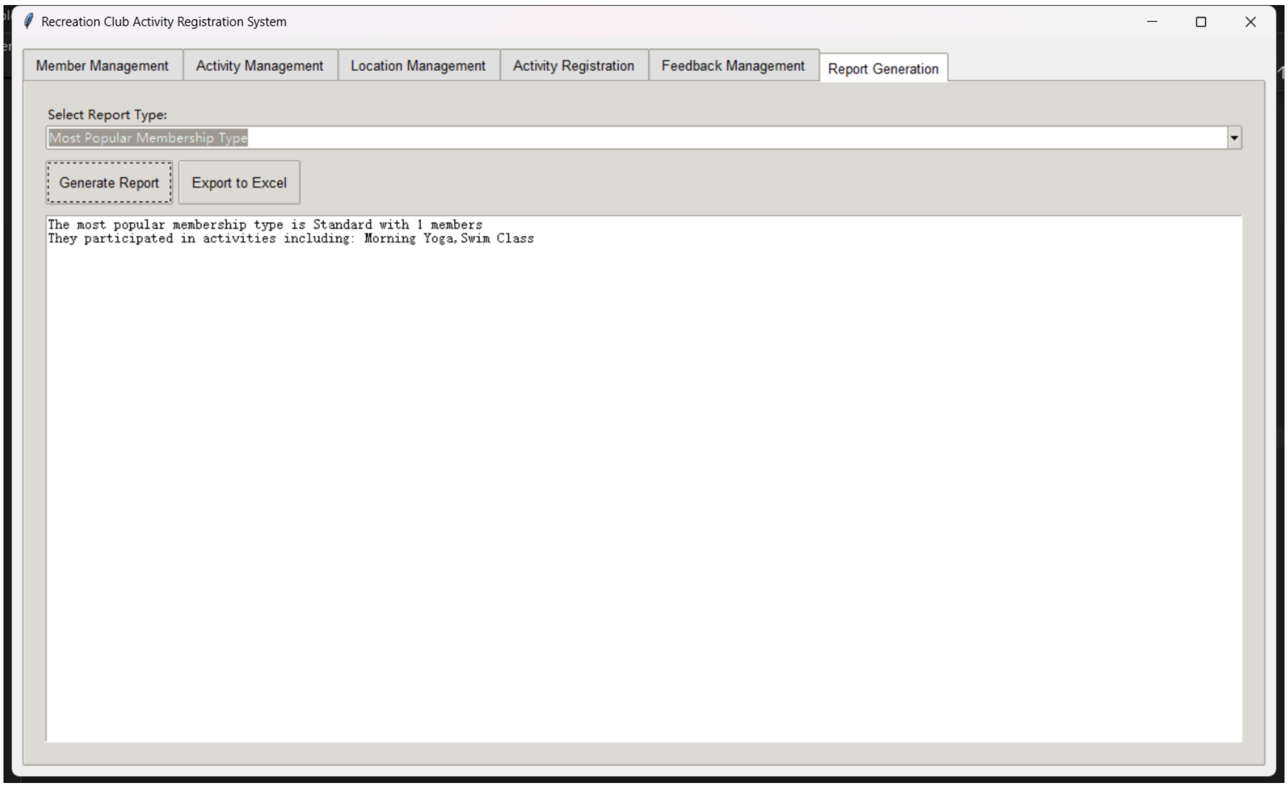
13. Complex join and aggregation: Identify the most popular types of membership and related activities

    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:



```
The most popular membership type is Standard with 1 members
They participated in activities including: Morning Yoga,Swim Class
```

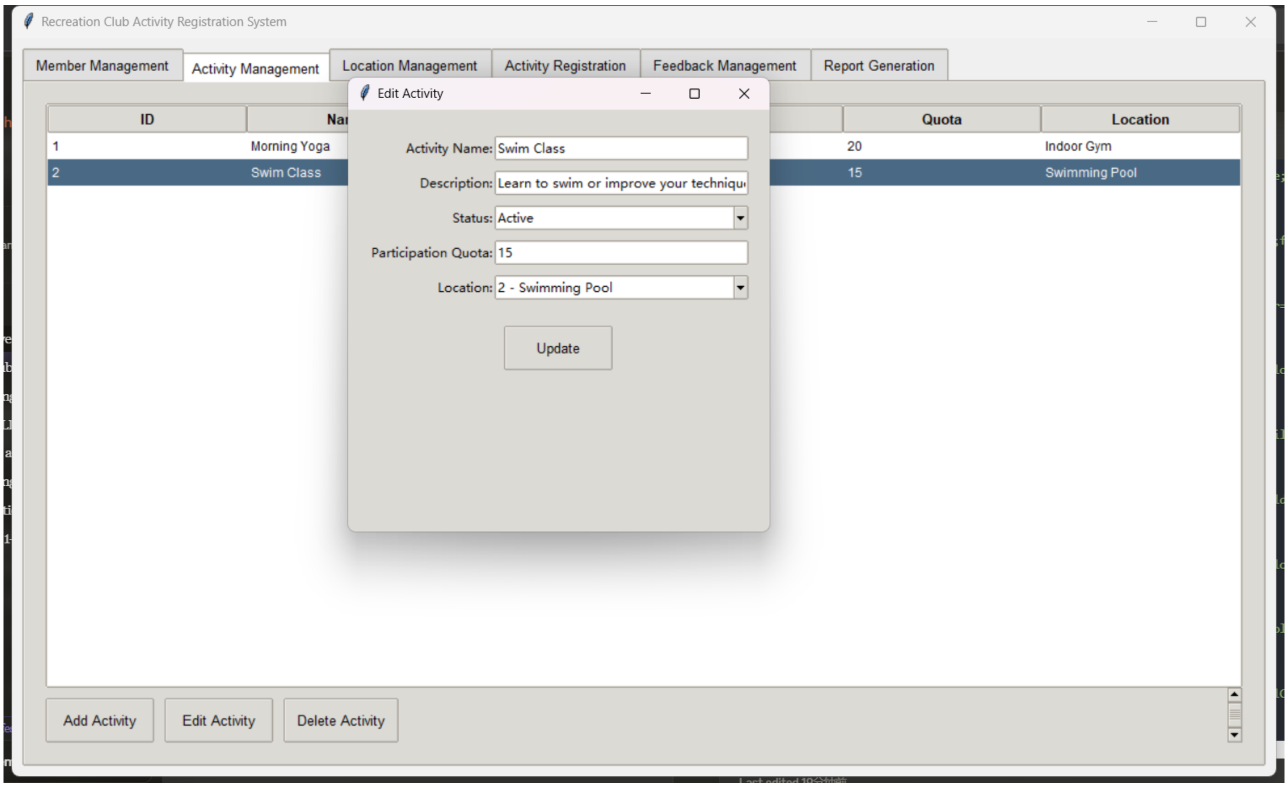
14. Updates based on aggregated data: Adjust activity status and description based on feedback

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

SQl codes:

```sql
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:


Recreation Club Activity Registration System

ID	Na	Quota	Location
1	Morning Yoga	20	Indoor Gym
2	Swim Class	15	Swimming Pool

Edit Activity

Activity Name: Swim Class
Description: Learn to swim or improve your techniqu
Status: Active
Participation Quota: 15
Location: 2 - Swimming Pool

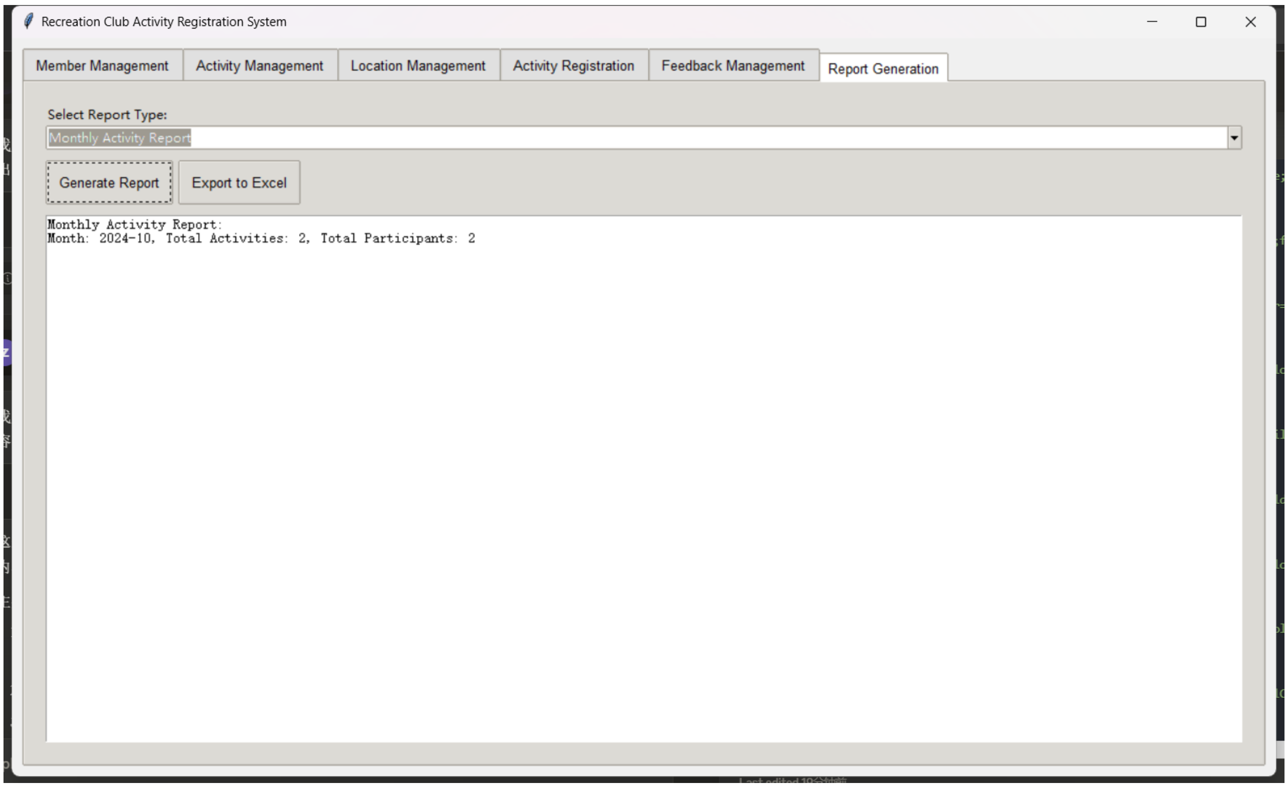
15. Comprehensive monthly report generation

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:



```
Monthly Activity Report:
Month: 2024-10, Total Activities: 2, Total Participants: 2
```

4 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