PROJECT REPORT

Topic-College Club Website

Submitted By-

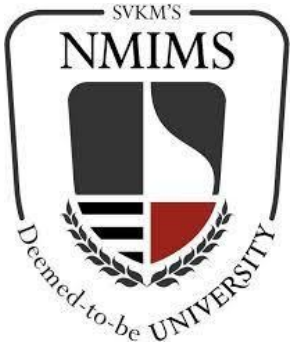
Bhumi Neema (D022)

Ashlesha Agrawal (D018)

Soumya Gangrade (D085)

Submitted To-

Dr. Shruti Sharma



Course:DBMS

AcademicYear:2024-25

TableofContents

I. Storyline

College Club Website

College clubs play a crucial role in fostering student engagement, collaboration, and skill development. Our College Club Website serves as a centralized platform for managing club activities, memberships, events, and communications. It streamlines club administration while enhancing student participation through an interactive and user-friendly interface. The system ensures that students can explore, join, and participate in various clubs while club administrators can efficiently manage events, memberships, and announcements.

1. Platform Overview

Users on our platform are classified into three primary roles:

- Students – Users who can explore clubs, join them, participate in events, and receive updates.
- Club Administrators – Members with administrative privileges to manage club events, memberships, and communications.
- Super Admin – Oversees all clubs, manages permissions, and ensures smooth platform operations.

The system enables club creation, event management, membership tracking, announcement broadcasting, and interactive discussions, fostering a dynamic and engaging student community.

2. Key Features & Functionalities

User Authentication & Interface

- Sign-in Page – Secure login for students, club admin, and super admin.
- Sign-up Page – New users can register and explore club activities.
- Profile Management – Users can create and update profiles with relevant details.

Club Directory & Membership

- Club Listing – A searchable directory of all registered clubs with descriptions.
- Join Requests & Approvals – Students can request to join clubs, and admin can approve/reject requests.
- Member Roles – Club admin can assign specific roles (e.g., President, Treasurer, Member).

Event Management

- Event Creation – Clubs can post upcoming events with details like date, location, and agenda.
- RSVP & Attendance Tracking – Students can register for events, and admins can track attendance.
- Event Reminders – Automated notifications for registered participants.

Communication & Announcements

- Discussion Forums – Interactive spaces for club members to engage in topic-based discussions.
- Announcement Board – Club admin can broadcast updates to members.

● Email & Push Notifications – Important updates on events, new members, and announcements.

Additional Services & Enhancements
Resource Sharing & Media Gallery
● Clubs can upload documents, presentations, and multimedia content for members.
● Photo and video galleries showcasing past events.

Polls & Surveys
● Clubs can conduct polls to gather member feedback or plan events.
● Results are displayed in real-time to encourage engagement.

Alumni Engagement
● Clubs can maintain an alumni section to stay connected with former members.
● Alumni can mentor current members or participate in club events.

Volunteer & Leadership Opportunities
● Students can apply for leadership positions within clubs.
● Admins can manage applications and assign responsibilities.

3. Database Structure & Efficiency
The database is designed for efficiency and scalability:
● User Management – Role-based access control for students, club admins, and super admins.
● Club & Membership Tracking – Handles club creation, memberships, and roles.
● Event Management – Stores event details, RSVPs, and attendance records.
● Communication System – Manages announcements, discussions, and notifications.

4. Real-World Impact & Use Case

Our platform enhances student engagement, leadership development, and community building by offering an organized and accessible hub for college club activities. It ensures smooth communication, simplifies event coordination, and strengthens alumni connections.

By leveraging Database Management Systems (DBMS) concepts, the platform optimizes data storage, retrieval, and security. The project demonstrates relational modeling, data normalization, and role-based access control while addressing key challenges like scalability, user experience, and secure data management.

I. Components of Database Design

Entities (Tables)

Entities represent real-world objects stored in the database. Each entity has attributes (columns).

Entity	Description
User	Represents students, club admins, and super admins.
Club	Represents different clubs in the system.
Club_Membership	Tracks users who have joined clubs.
Event	Represents events organized by clubs.
Event_Membership	Stores event attendance details.
Announcement	Stores important club updates.
Discussion	Represents discussions created within clubs.
Comment	Stores comments under discussions.
Resource	Stores documents, images, or videos shared in a club.
Poll	Represents club polls.
Volunteer_Opportunity	Stores available volunteer roles.

Attributes (Columns in Each Table)

Each entity has attributes that store relevant data.

User (User_ID - Primary Key)

Attribute	Data Type	Description
User_ID	INT (PK)	Unique identifier for a user
Name	VARCHAR	User's full name
Email	VARCHAR	User’s email address
Password	VARCHAR	Encrypted password
Role	ENUM('Student', 'Club Admin', 'Super Admin')	Defines user role
Username	VARCHAR	User’s username
Created_at	TIMESTAMP	User’s current time stamp

Club (Club_ID - Primary Key)

Attribute	Data Type	Description
Club_ID	INT (PK)	Unique club identifier
Name	VARCHAR	Club name
Description	TEXT	Club details
Created_Date	DATE	When the club was founded
Club_Admin_ID	INT (FK)	Admin user managing the club

Club Membership (Club_Membership_ID - Primary Key)

Attribute	Data Type	Description
Membership_ID	INT (PK)	Unique membership identifier
User_ID	INT (FK)	User who joined the club
Club_name	VARCHAR	Club being joined
Role	ENUM('President', 'Treasurer', 'Member')	User's role in the club
Join Date	DATE	User’s joining date

Event (Event_ID - Primary Key)

Attribute	Data Type	Description
Event_ID	INT (PK)	Unique event identifier
Name	VARCHAR	Event name
Description	TEXT	Event details
Date	DATE	Event date

Attribute	Data Type	Description
Created at	DATE	Event’s Date
Club_ID	INT (FK)	Club organizing the event

Even_Registration (Event Registration_ID - Primary Key)

Attribute	Data Type	Description
_ID	INT (PK)	Unique identifier
User_ID	INT (FK)	User attending the event
Event_ID	INT (FK)	Event being attended
Event name	VARCHAR	Event’s name
Event Date	DATE	Event’s date
Category	VARCHAR	Event’s category
Full Nmae	VARCHAR(FK)	User’s full name
Email	VARCHAR	User’s email
Phone number	INT	User’s number
Year	VARCHAR	User’s year
Registration Date	TIMESTAMP	Event’s Timestamp

Announcement (Announcement_ID - Primary Key)

Attribute	Data Type	Description
Announcement_ID	INT (PK)	Unique announcement identifier
Title	VARCHAR	Announcement title
Message	TEXT	Announcement message
Club name	VARCHAR	Club’s announcement
Importance	VARCHAR	Description
Created at	TIMESTAMP	Current time stamp

Discussion (Discussion_ID - Primary Key)

Attribute	Data Type	Description
Discussion_ID	INT (PK)	Unique discussion identifier
Club_ID	INT (FK)	Club where the discussion is posted
User_ID	INT (FK)	User who created the discussion
Topic	TEXT	Discussion topic
Date	DATE	Date discussion was created

Comment (Comment_ID - Primary Key)

Attribute	Data Type	Description
Comment_ID	INT (PK)	Unique comment identifier
Discussion_ID	INT (FK)	Discussion where the comment is posted
User_ID	INT (FK)	User who made the comment
Content	TEXT	Comment content
Date	DATE	Date comment was posted

Resource (Resource_ID - Primary Key)

Attribute	Data Type	Description
Resource_ID	INT (PK)	Unique resource identifier
Club_ID	INT (FK)	Club where resource is shared
Uploaded_By	INT (FK)	User who uploaded the resource
Type	ENUM('Document', 'Image', 'Video')	Type of resource
URL	VARCHAR	Resource link
Upload_Date	DATE	Date of upload

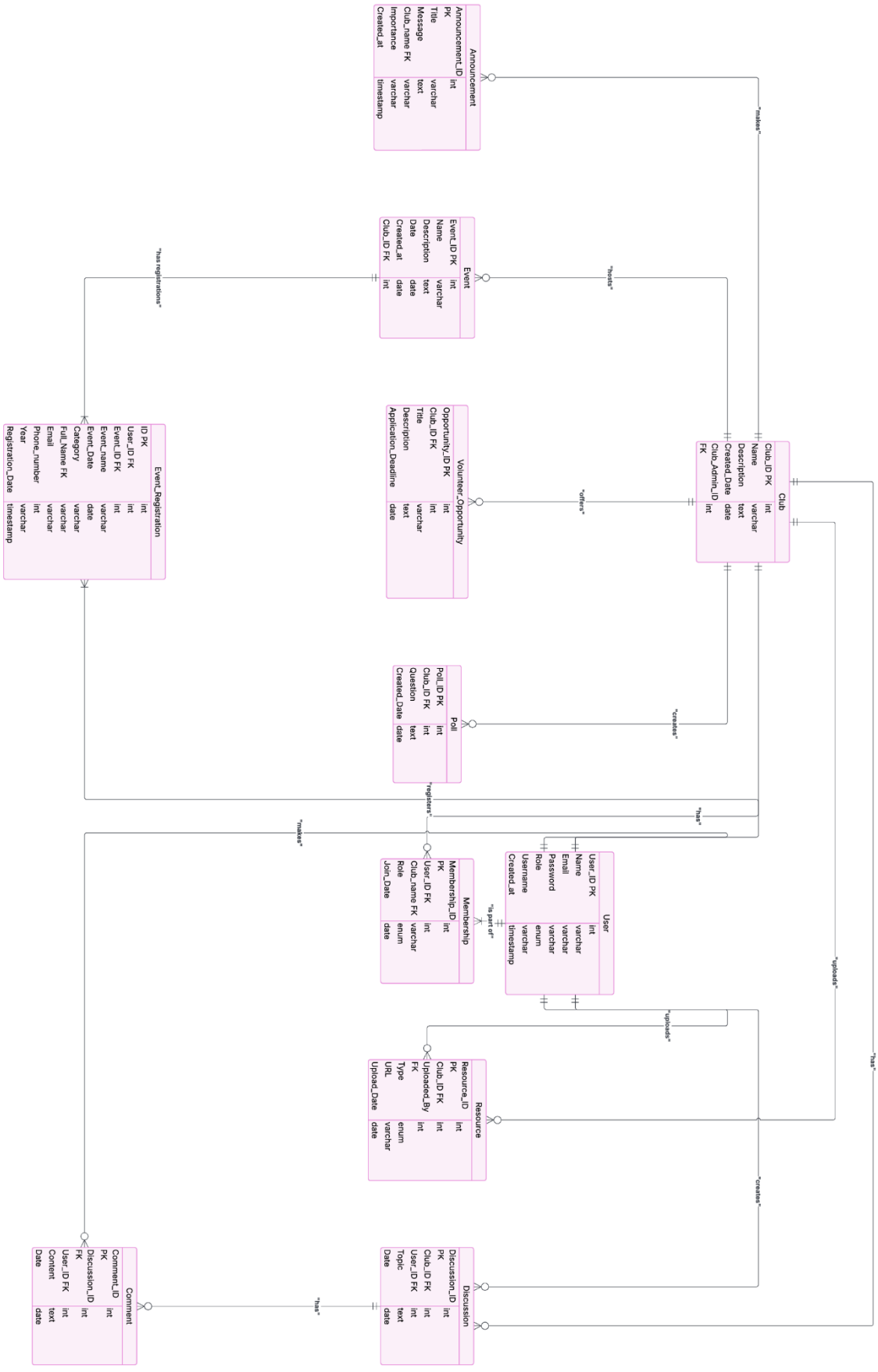
Poll (Poll_ID - Primary Key)

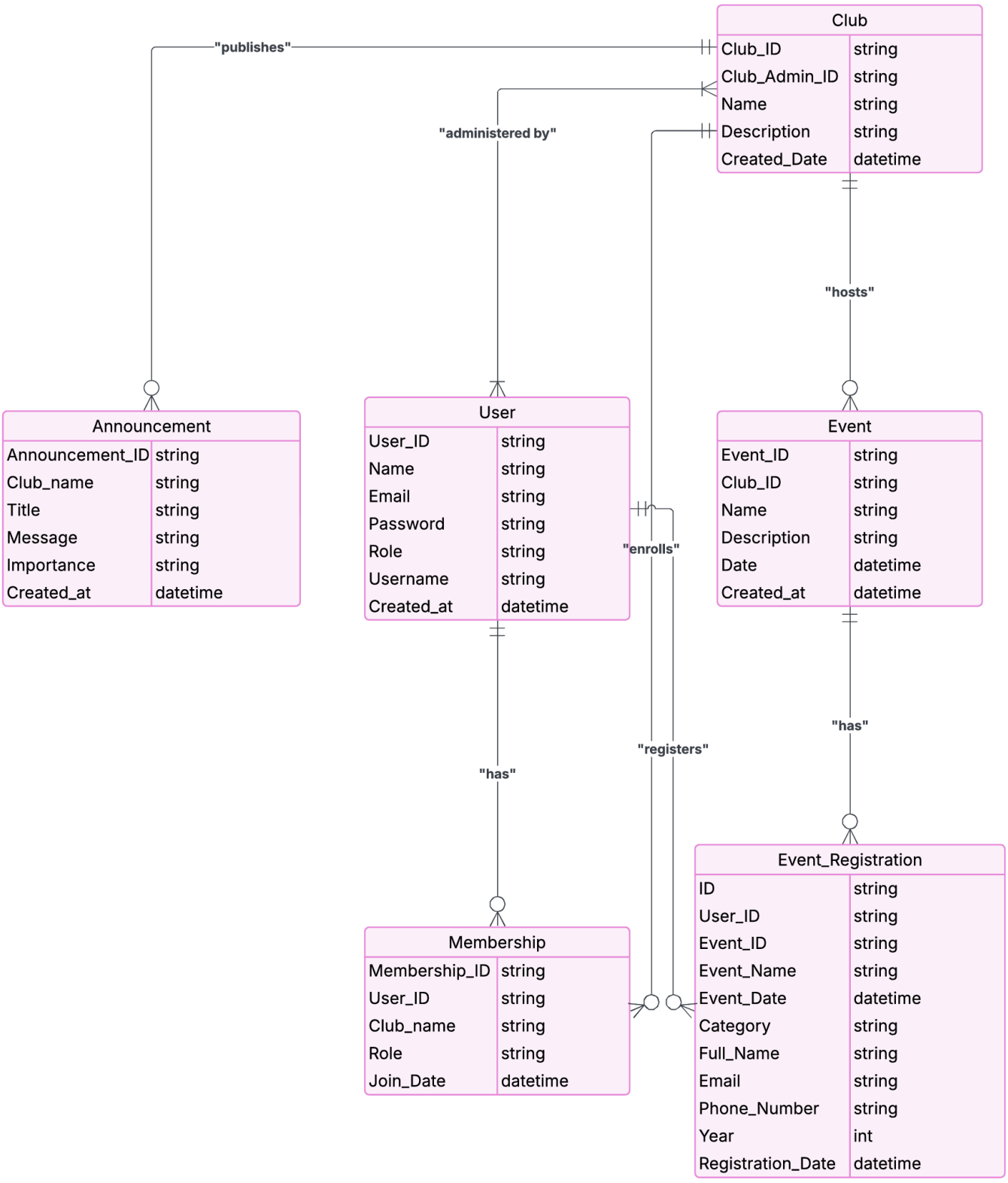
Attribute	Data Type	Description
Poll_ID	INT (PK)	Unique poll identifier
Club_ID	INT (FK)	Club where poll is created
Question	TEXT	Poll question
Created_Date	DATE	Date poll was created

Volunteer_Opportunity (Opportunity_ID - Primary Key)

Attribute	Data Type	Description
Opportunity_ID	INT (PK)	Unique opportunity identifier
Club_ID	INT (FK)	Club posting the opportunity
Title	VARCHAR	Title of the volunteer opportunity
Description	TEXT	Details of the opportunity
Application_Deadline	DATE	Last date to apply

Entity Relationship Diagram




Club
Club_ID string
Club_Admin_ID string
Name string
Description string
Created_Date datetime
"publishes"
"administered by"
"hosts"
Announcement
Announcement_ID string
Club_name string
Title string
Message string
Importance string
Created_at datetime
User
User_ID string
Name string
Email string
Password string
Role string
Username string
Created_at datetime
"enrolls"
Event
Event_ID string
Club_ID string
Name string
Description string
Date datetime
Created_at datetime
"has"
"registers"
"has"
Membership
Membership_ID string
User_ID string
Club_name string
Role string
Join_Date datetime
Event_Registration
ID string
User_ID string
Event_ID string
Event_Name string
Event_Date datetime
Category string
Full_Name string
Email string
Phone_Number string
Year int
Registration_Date datetime

II. Relational Model

Table Name	Primary Key (PK)	Foreign Keys (FK) & Relationships	Attributes
User	User_ID	None	Name, Email, Password, Role, Username, Created_at
Club	Club_ID	Club_Admin_ID → User(User_ID)	Name, Description, Created_Date
Membership	Membership_ID	User_ID → User(User_ID), Club_name → Club(Name)	Role, Join_Date
Event	Event_ID	Club_ID → Club(Club_ID)	Name, Description, Date, Created_at
Event_Registration	ID	User_ID → User(User_ID), Event_ID → Event(Event_ID)	Event_Name, Event_Date, Category, Full_Name, Email, Phone_Number, Year, Registration_Date
Announcement	Announcement_ID	Club_name → Club(Name)	Title, Message, Importance, Created_at
Discussion	Discussion_ID	Club_ID → Club(Club_ID), User_ID → User(User_ID)	Topic, Date
Comment	Comment_ID	Discussion_ID → Discussion(Discussion_ID), User_ID → User(User_ID)	Content, Date
Resource	Resource_ID	Club_ID → Club(Club_ID), Uploaded_By → User(User_ID)	Type, URL, Upload_Date
Poll	Poll_ID	Club_ID → Club(Club_ID)	Question, Created_Date
Volunteer_Opportunity	Opportunity_ID	Club_ID → Club(Club_ID)	Title, Description, Application_Deadline

Explanation of Relationships:

1. **User-Club (Many-to-Many via Membership)**

- A user can be part of multiple clubs and each club can have many users.
- This is managed by the **Membership** table.

2. **Club-Event (One-to-Many)**

- A club can host multiple events.
- **Event.Club_ID** references **Club.Club_ID**.

3. **User-Event (Many-to-Many via Event_Registration)**

- Users can register for many events, and each event can have many users.
- This is handled by **Event_Registration** using **User_ID** and **Event_ID**.

4. **Club-Announcement (One-to-Many)**

- A club can make multiple announcements.
- **Announcement.Club_name** references **Club.Name**.

5. **Club-Discussion (One-to-Many)**

- A club can have multiple discussions.
- **Discussion.Club_ID** references **Club.Club_ID**.

6. **User-Discussion (One-to-Many)**

- A user can start multiple discussions.
- **Discussion.User_ID** references **User.User_ID**.

7. **Discussion-Comment (One-to-Many)**

- A discussion can have many comments.
- **Comment.Discussion_ID** references **Discussion.Discussion_ID**.

8. **User-Comment (One-to-Many)**

- A user can post many comments.
- **Comment.User_ID** references **User.User_ID**.

9. **Club-Resource (One-to-Many)**

- A club can upload multiple resources.
- **Resource.Club_ID** references **Club.Club_ID**.

10. **User-Resource (One-to-Many)**

- A user can upload many resources.
- **Resource.Uploaded_By** references **User.User_ID**.

11. **Club-Poll (One-to-Many)**

- A club can create many polls.
- **Poll.Club_ID** references **Club.Club_ID**.

12. **Poll-Poll_Option (One-to-Many)**

- A poll can have multiple options.
- **Poll_Option.Poll_ID** references **Poll.Poll_ID**.

13. **Poll_Response (Many-to-Many)**

- A user can respond to many polls and each poll can be answered by many users.
- Managed by **Poll_Response** using **Poll_ID**, **User_ID**, and **Option_ID**.

14. **Club-Volunteer_Opportunity (One-to-Many)**

- A club can offer multiple volunteering opportunities.
- **Volunteer_Opportunity.Club_ID** references **Club.Club_ID**.

15. **User-Application (Many-to-Many via Application)**

- Users can apply to many opportunities.
- **Application** links **User_ID** and **Opportunity_ID**.

Normalisation:

First Normal Form (1NF)

Definition:

A table is in 1NF if:

- It only contains **atomic (indivisible)** values.
- Each cell holds a single value (no repeating groups or arrays).
- Each record is unique.

Analysis of Your Tables:

Most of your tables **are already in 1NF** since:

- Each column contains atomic values (e.g., no multiple phone numbers or emails in a single field).
- There are no repeating groups or multivalued attributes.

So: **All your tables are in 1NF**. No transformation is needed here.

Second Normal Form (2NF)

Definition:

A table is in 2NF if:

- It is already in 1NF.
- **All non-key attributes are fully functionally dependent** on the **entire primary key**, not just part of it (important for tables with **composite keys**).

Tables That Violate 2NF:

2NF Schema (after fixing partial dependencies)

1. User

User(User_ID PK, Name, Email, Password, Role, Username, Created_at)

---

2. Club

Club(Club_ID PK, Name, Description, Created_Date, Club_Admin_ID FK)

---

3. Club_Membership

Membership_ID PK, User_ID FK, Club_ID FK, Role, Join_Date
-- Removed Club_Name (was redundant)

---

4. Event

Event_ID PK, Name, Description, Date, Created_at, Club_ID FK

---

5. Event_Registration

Split into two:

a. Event_Registration

(Event_ID, User_ID) PK, Category, Registration_Date

b. User_Details (cached or extended User table)
User_ID PK, Full_Name, Email, Phone_Number, Year

---

6. Announcement
Announcement_ID PK, Title, Message, Club_ID FK, Importance, Created_at
-- Removed Club_Name (redundant via FK)

✅ Now there are **no partial dependencies**, and all non-key attributes depend fully on the key.

Third Normal Form (3NF)

Definition:
A table is in 3NF if:

- It is in 2NF.
- There is **no transitive dependency**, i.e., non-key attributes don't depend on other non-key attributes.

3NF Schema (removing transitive dependencies)

In User, if we were storing role-based details (like permissions, access level), and Role determined those, we'd have:
User(User_ID, Name, Role, Access_Level) -- Access_Level is transitively dependent on Role
Fix by moving role details to a new table:

User_Role(Role_Name PK, Description, Access_Level)
User(User_ID PK, Name, Email, Password, Username, Role FK, Created_at)

Boyce-Codd Normal Form (BCNF)

Definition:
A table is in BCNF if:

- It is in 3NF.
- For **every functional dependency (X → Y)**, **X is a super key**.

In **Event_Registration:**
Full_Name → Email, Phone_Number, Year

Then Full_Name becomes a determinant, which is **not a key** — violating BCNF.

Fix:
We've already done this in 2NF/3NF by removing those fields and relying on User_ID.

SQL Queries

```
-- -------------------------
-- TABLE CREATION QUERIES
-- -------------------------
```

1.
```
CREATE TABLE User (
  User_ID INT PRIMARY KEY,
  Name VARCHAR(100),
  Email VARCHAR(100) UNIQUE,
  Password VARCHAR(100),
  Role ENUM('Student', 'Club Admin', 'Super Admin'),
  Username VARCHAR(50) UNIQUE,
  Created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

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

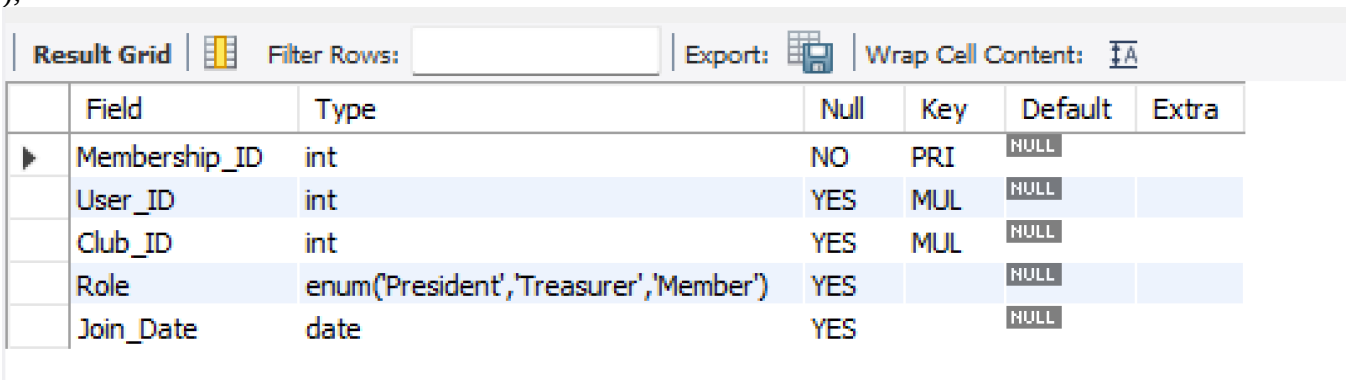
Field	Type	Null	Key	Default	Extra
User_ID	int	NO	PRI	NULL	
Name	varchar(100)	YES		NULL	
Email	varchar(100)	YES	UNI	NULL	
Password	varchar(100)	YES		NULL	
Role	enum('Student','Club Admin','Super Admin')	YES		NULL	
Username	varchar(50)	YES	UNI	NULL	
Created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

2.
```
CREATE TABLE Club (
  Club_ID INT PRIMARY KEY,
  Name VARCHAR(100),
  Description TEXT,
  Created_Date DATE,
  Club_Admin_ID INT,
  FOREIGN KEY (Club_Admin_ID) REFERENCES User(User_ID)
);
```

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

Field	Type	Null	Key	Default	Extra
Club_ID	int	NO	PRI	NULL	
Name	varchar(100)	YES		NULL	
Description	text	YES		NULL	
Created_Date	date	YES		NULL	
Club_Admin_ID	int	YES	MUL	NULL	

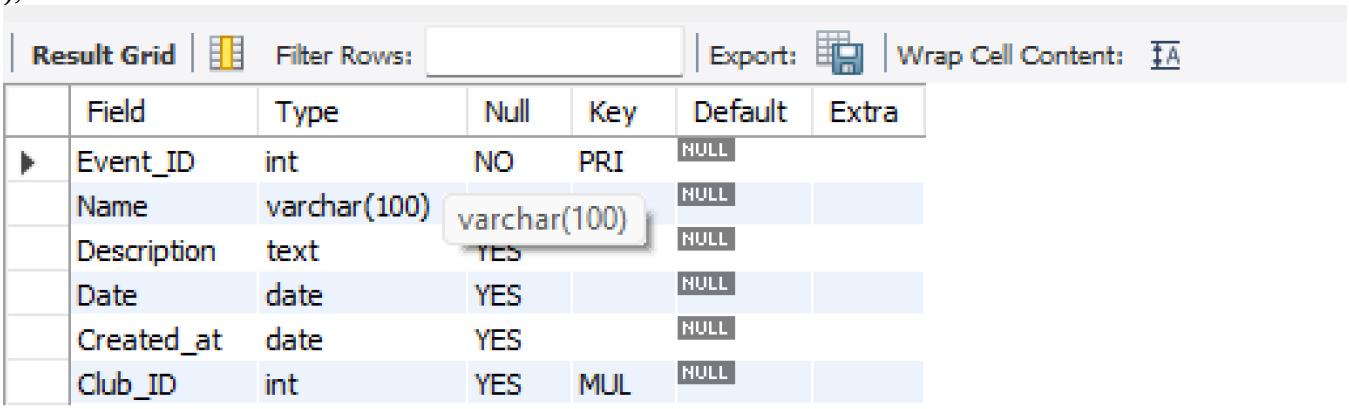
3. CREATE TABLE Club_Membership (
   Membership_ID INT PRIMARY KEY,
   User_ID INT,
   Club_ID INT,
   Role ENUM('President', 'Treasurer', 'Member'),
   Join_Date DATE,
   FOREIGN KEY (User_ID) REFERENCES User(User_ID),
   FOREIGN KEY (Club_ID) REFERENCES Club(Club_ID)
  );



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

Field	Type	Null	Key	Default	Extra
Membership_ID	int	NO	PRI	NULL	
User_ID	int	YES	MUL	NULL	
Club_ID	int	YES	MUL	NULL	
Role	enum('President','Treasurer','Member')	YES		NULL	
Join_Date	date	YES		NULL	

4. CREATE TABLE Event (
   Event_ID INT PRIMARY KEY,
   Name VARCHAR(100),
   Description TEXT,
   Date DATE,
   Created_at DATE,
   Club_ID INT,
   FOREIGN KEY (Club_ID) REFERENCES Club(Club_ID)
  );

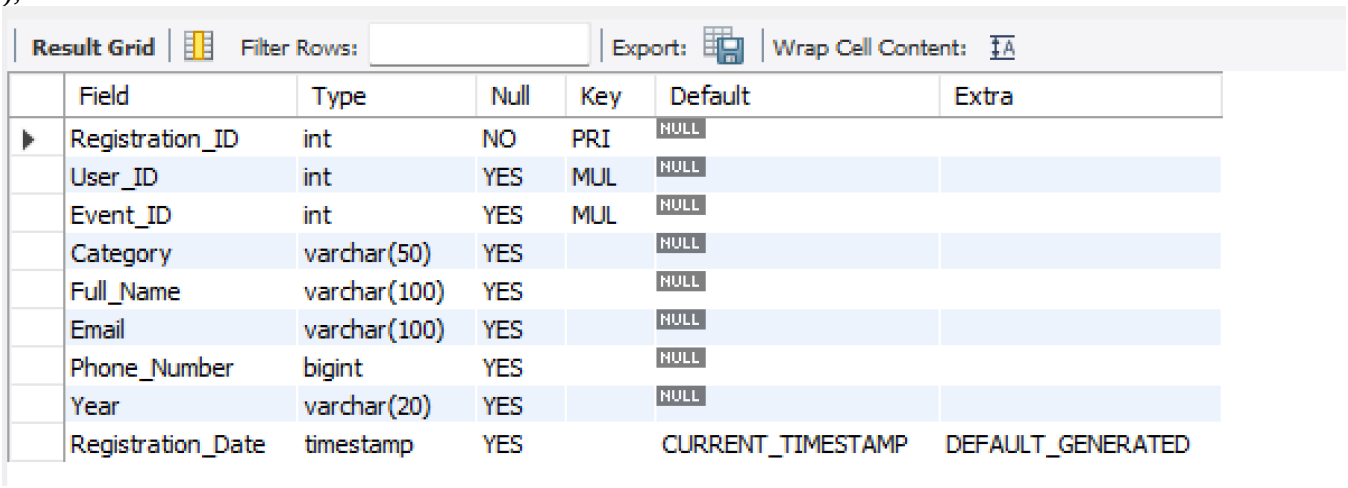


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Event_ID	int	NO	PRI	NULL	
Name	varchar(100)			NULL	
Description	text	YES		NULL	
Date	date	YES		NULL	
Created_at	date	YES		NULL	
Club_ID	int	YES	MUL	NULL	

5. CREATE TABLE Event_Registration (
   Registration_ID INT PRIMARY KEY,
   User_ID INT,
   Event_ID INT,
   Category VARCHAR(50),

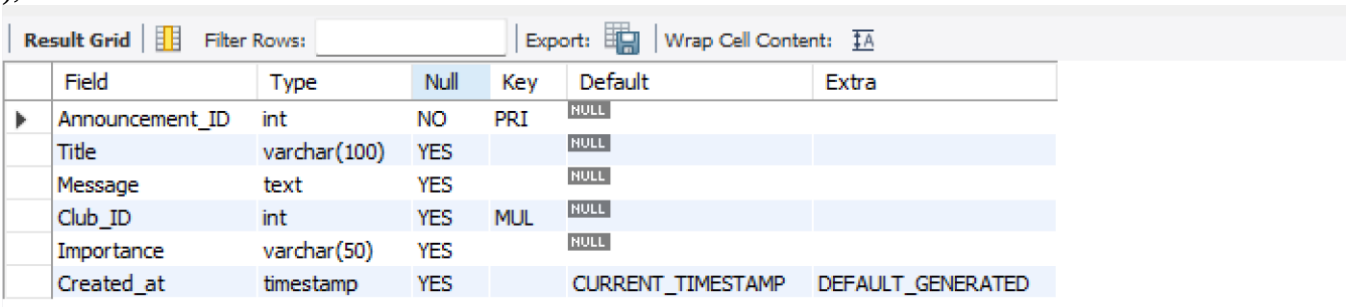
Full_Name VARCHAR(100),
Email VARCHAR(100),
Phone_Number BIGINT,
Year VARCHAR(20),
Registration_Date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (User_ID) REFERENCES User(User_ID),
FOREIGN KEY (Event_ID) REFERENCES Event(Event_ID)
);



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

Field	Type	Null	Key	Default	Extra
Registration_ID	int	NO	PRI	NULL	
User_ID	int	YES	MUL	NULL	
Event_ID	int	YES	MUL	NULL	
Category	varchar(50)	YES		NULL	
Full_Name	varchar(100)	YES		NULL	
Email	varchar(100)	YES		NULL	
Phone_Number	bigint	YES		NULL	
Year	varchar(20)	YES		NULL	
Registration_Date	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

6. CREATE TABLE Announcement (
Announcement_ID INT PRIMARY KEY,
Title VARCHAR(100),
Message TEXT,
Club_ID INT,
Importance VARCHAR(50),
Created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (Club_ID) REFERENCES Club(Club_ID)
);

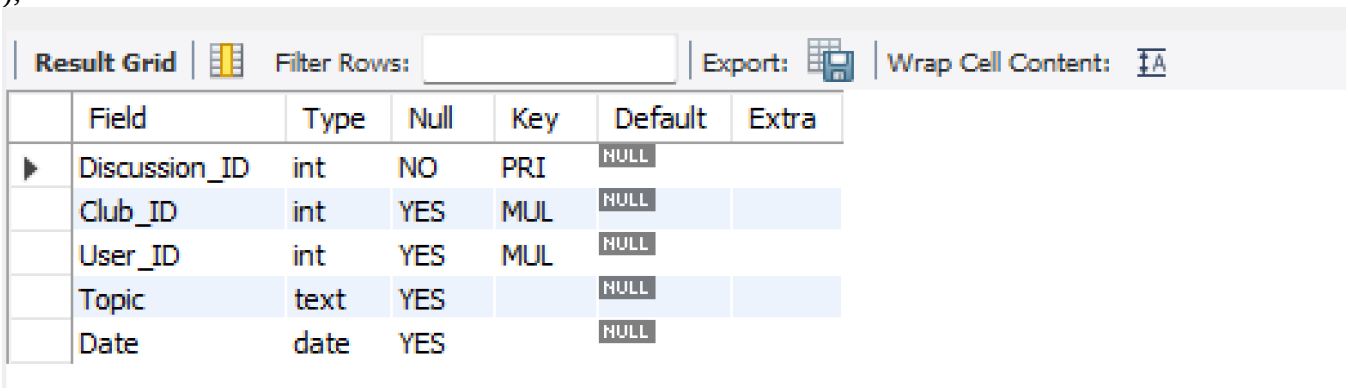


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Announcement_ID	int	NO	PRI	NULL	
Title	varchar(100)	YES		NULL	
Message	text	YES		NULL	
Club_ID	int	YES	MUL	NULL	
Importance	varchar(50)	YES		NULL	
Created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

7. CREATE TABLE Discussion (
Discussion_ID INT PRIMARY KEY,
Club_ID INT,
User_ID INT,
Topic TEXT,
Date DATE,
FOREIGN KEY (Club_ID) REFERENCES Club(Club_ID),

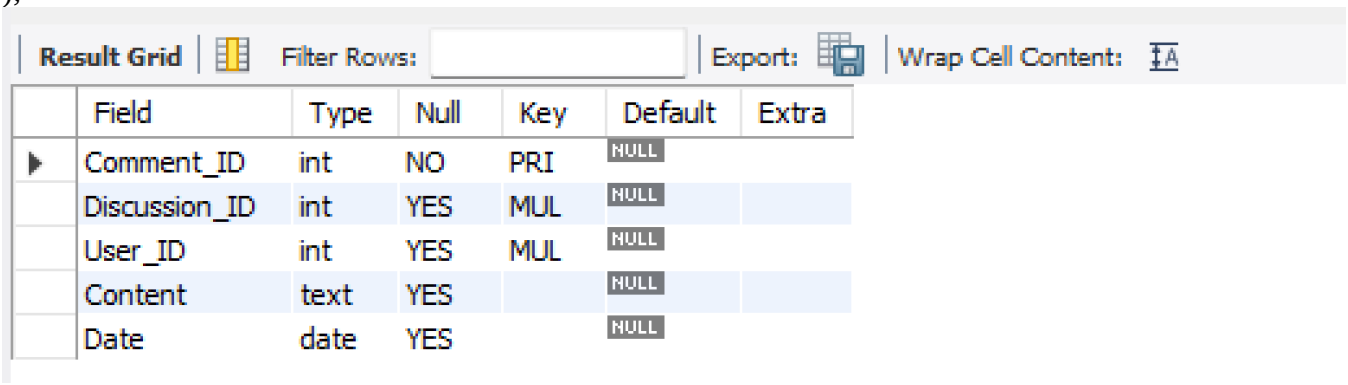
FOREIGN KEY (User_ID) REFERENCES User(User_ID)
);



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

Field	Type	Null	Key	Default	Extra
Discussion_ID	int	NO	PRI	NULL	
Club_ID	int	YES	MUL	NULL	
User_ID	int	YES	MUL	NULL	
Topic	text	YES		NULL	
Date	date	YES		NULL	

8. CREATE TABLE Comment (
Comment_ID INT PRIMARY KEY,
Discussion_ID INT,
User_ID INT,
Content TEXT,
Date DATE,
FOREIGN KEY (Discussion_ID) REFERENCES Discussion(Discussion_ID),
FOREIGN KEY (User_ID) REFERENCES User(User_ID)
);



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

Field	Type	Null	Key	Default	Extra
Comment_ID	int	NO	PRI	NULL	
Discussion_ID	int	YES	MUL	NULL	
User_ID	int	YES	MUL	NULL	
Content	text	YES		NULL	
Date	date	YES		NULL	

9. CREATE TABLE Resource (
Resource_ID INT PRIMARY KEY,
Club_ID INT,
Uploaded_By INT,
Type ENUM('Document', 'Image', 'Video'),
URL VARCHAR(255),
Upload_Date DATE,
FOREIGN KEY (Club_ID) REFERENCES Club(Club_ID),
FOREIGN KEY (Uploaded_By) REFERENCES User(User_ID)
);

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

Field	Type	Null	Key	Default	Extra
Resource_ID	int	NO	PRI	NULL	
Club_ID	int	YES	MUL	NULL	
Uploaded_By	int	YES	MUL	NULL	
Type	enum('Document','Image','Video')	YES		NULL	
URL	varchar(255)	YES		NULL	
Upload_Date	date	YES		NULL	

10. CREATE TABLE Poll (
    Poll_ID INT PRIMARY KEY,
    Club_ID INT,
    Question TEXT,
    Created_Date DATE,
    FOREIGN KEY (Club_ID) REFERENCES Club(Club_ID)
    );

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

Field	Type	Null	Key	Default	Extra
Poll_ID	int	NO	PRI	NULL	
Club_ID	int	YES	MUL	NULL	
Question	text	YES		NULL	
Created_Date	date	YES		NULL	

11. CREATE TABLE Volunteer_Opportunity (
    Opportunity_ID INT PRIMARY KEY,
    Club_ID INT,
    Title VARCHAR(100),
    Description TEXT,
    Application_Deadline DATE,
    FOREIGN KEY (Club_ID) REFERENCES Club(Club_ID)
    );

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

Field	Type	Null	Key	Default	Extra
Opportunity_ID	int	NO	PRI	NULL	
Club_ID	int	YES	MUL	NULL	
Title	varchar(100)	YES		NULL	
Description	text	YES		NULL	
Application_Deadline	date	YES		NULL	

12.

```
211 •   SELECT Role, COUNT(*) AS TotalUsers
212     FROM User
213     GROUP BY Role;
214
```

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

Role	TotalUsers
Student	3
Club Admin	1
Super Admin	1

13.

```
215 •   SELECT Role, DATE(Created_at) AS CreatedDate, COUNT(*) AS UsersPerDay
216     FROM User
217     GROUP BY Role, DATE(Created_at);
218
```

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

Role	CreatedDate	UsersPerDay
Student	2025-04-16	3
Club Admin	2025-04-16	1
Super Admin	2025-04-16	1

14.

```
219 •   SELECT *
220     FROM User
221     WHERE User_ID IN (
222         SELECT Club_Admin_ID
223         FROM Club
224     );
225
```

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

User_ID	Name	Email	Password	Role	Username	Created_at
2	Rahul Mehta	rahul@example.com	pass123	Club Admin	rahul_m	2025-04-16 13:24:18
3	Sneha Patel	sneha@example.com	pass123	Student	sneha_p	2025-04-16 13:24:18
4	Dev Jain	dev@example.com	pass123	Super Admin	dev_j	2025-04-16 13:24:18
NULL	NULL	NULL	NULL	NULL	NULL	NULL

15. CREATE VIEW View_User_Roles AS
SELECT User_ID, Name, Role
FROM User;

16. SELECT *FROM User ORDER BY Created_at DESC;

17. SELECT u.Name, c.Name AS ClubName FROM User u INNER JOIN Club c ON u.User_ID = c.Club_Admin_ID;

18. SELECT u.Name, c.Name AS ClubName FROM User u LEFT JOIN Club c ON u.User_ID = c.Club_Admin_ID;

19. SELECT u.Name, c.Name AS ClubName FROM User u RIGHT JOIN Club c ON u.User_ID = c.Club_Admin_ID;

20. SELECT u.Name, c.Name AS ClubName FROM User u LEFT JOIN Club c ON u.User_ID = c.Club_Admin_ID UNION SELECT u.Name, c.Name AS ClubName FROM User u RIGHT JOIN Club c ON u.User_ID = c.Club_Admin_ID;

21. SELECT u.Name, c.Name AS ClubName FROM User u CROSS JOIN Club c;

22. SELECT YEAR(Created_Date) AS Year, COUNT(*) AS TotalClubs FROM Club GROUP BY YEAR(Created_Date);

23. SELECT Club_Admin_ID, COUNT(*) AS ClubCount FROM Club GROUP BY Club_Admin_ID;

24. SELECT * FROM Club WHERE YEAR(Created_Date) > (SELECT AVG(YEAR(Created_Date)) FROM Club);

25. CREATE VIEW View_Club_Admins AS SELECT c.Club_ID, c.Name AS ClubName, u.Name AS AdminName FROM Club c JOIN User u ON c.Club_Admin_ID = u.User_ID;

26. -- Order clubs by creation date, newest first: SELECT *FROM Club ORDER BY Created_Date DESC;

27. -- Show club names with their admin names: SELECT c.Name AS ClubName, u.Name AS AdminName FROM Club c INNER JOIN User u ON c.Club_Admin_ID = u.User_ID;

28. -- Show all clubs and their admins, even if the admin record is missing: SELECT c.Name AS lubName, u.Name AS AdminName FROM Club c LEFT JOIN User u ON c.Club_Admin_ID = u.User_ID;

29. -- Show all users and the clubs they manage (some users may not manage any clubs): SELECT u.Name AS AdminName, c.Name AS ClubName FROM Club cRIGHT JOIN User u ON c.Club_Admin_ID = u.User_ID;

30. -- All clubs and all users, with matching where possible: SELECT c.Name AS ClubName, u.Name AS AdminName FROM Club c LEFT JOIN User u ON c.Club_Admin_ID = u.User_ID UNION SELECT c.Name AS ClubName, u.Name AS AdminName FROM Club c RIGHT JOIN User u ON c.Club_Admin_ID = u.User_ID;

31. -- Every club with every user (caution: large output): SELECT c.Name AS ClubName, u.Name AS UserName FROM Club c CROSS JOIN User u;

32. -- Count members by role: SELECT Role, COUNT(*) AS TotalMembers FROM Club_Membership GROUP BY Role;

33. --Count members by role: SELECT Club_name, COUNT(*) AS MemberCount FROM Club_Membership GROUP BY Club_name;

34. -- Count members by role:
```
SELECT Role, COUNT(*) AS TotalMembers
FROM Club_Membership
GROUP BY Role;
```

35. -- Count members by club:
```
SELECT Club_name, COUNT(*) AS MemberCount
FROM Club_Membership
GROUP BY Club_name;
```

36. -- Count members by club and their role:
```
SELECT Club_name, Role, COUNT(*) AS Total
FROM Club_Membership
GROUP BY Club_name, Role;
```

37. -- Get users who are President in any club:
```
SELECT User_ID
FROM Club_Membership
WHERE Role = 'President';
```

38. -- Get club(s) with the maximum number of members:
```
SELECT Club_name
FROM Club_Membership
GROUP BY Club_name
HAVING COUNT(*) = (
  SELECT MAX(member_count)
  FROM (
    SELECT COUNT(*) AS member_count
    FROM Club_Membership
    GROUP BY Club_name
  ) AS sub
);
```

39. -- Create view showing member names with club and role:
```
CREATE VIEW View_ClubMembers AS
SELECT cm.Membership_ID, u.Name AS MemberName, cm.Club_name, cm.Role
FROM Club_Membership cm
JOIN User u ON cm.User_ID = u.User_ID;
```

40. -- List all members ordered by Join Date:
```
SELECT *
FROM Club_Membership
ORDER BY Join_Date DESC;
```

41. -- Members sorted by club and role:
```
SELECT *
FROM Club_Membership
ORDER BY Club_name ASC, Role ASC;
```

42. -- INNER JOIN: Member name, club name, and role:
```
SELECT u.Name AS MemberName, cm.Club_name, cm.Role
FROM Club_Membership cm
INNER JOIN User u ON cm.User_ID = u.User_ID;
```

43. -- LEFT JOIN: All memberships with user info:
```
SELECT cm.*, u.Name AS UserName
FROM Club_Membership cm
LEFT JOIN User u ON cm.User_ID = u.User_ID;
```

44. -- RIGHT JOIN: All users with the clubs they are members of:
```
SELECT u.Name AS UserName, cm.Club_name
FROM Club_Membership cm
RIGHT JOIN User u ON cm.User_ID = u.User_ID;
```

45. -- FULL OUTER JOIN using UNION:
```
SELECT cm.Club_name, u.Name AS UserName
FROM Club_Membership cm
LEFT JOIN User u ON cm.User_ID = u.User_ID
UNION
SELECT cm.Club_name, u.Name AS UserName
FROM Club_Membership cm
RIGHT JOIN User u ON cm.User_ID = u.User_ID;
```

46. -- CROSS JOIN: Every membership with every user:
```
SELECT u.Name AS UserName, cm.Club_name
FROM User u
CROSS JOIN Club_Membership cm;
```

Learning from the Project

Working on the NMIMS College Clubs project has been a highly enriching experience that strengthened both my theoretical and practical understanding of database management and full-stack development. From the initial planning stage of identifying entities to constructing the ER diagram and relational schema, I learned how crucial normalization is in designing efficient and scalable databases. Understanding and applying concepts like 1NF, 2NF, 3NF, and BCNF helped minimize redundancy and improve data integrity.

Implementing SQL queries using aggregate functions, joins, subqueries, and views gave me hands-on experience in extracting meaningful insights from structured data. I also learned how to create relationships between multiple tables using primary and foreign keys, ensuring referential integrity across the database.

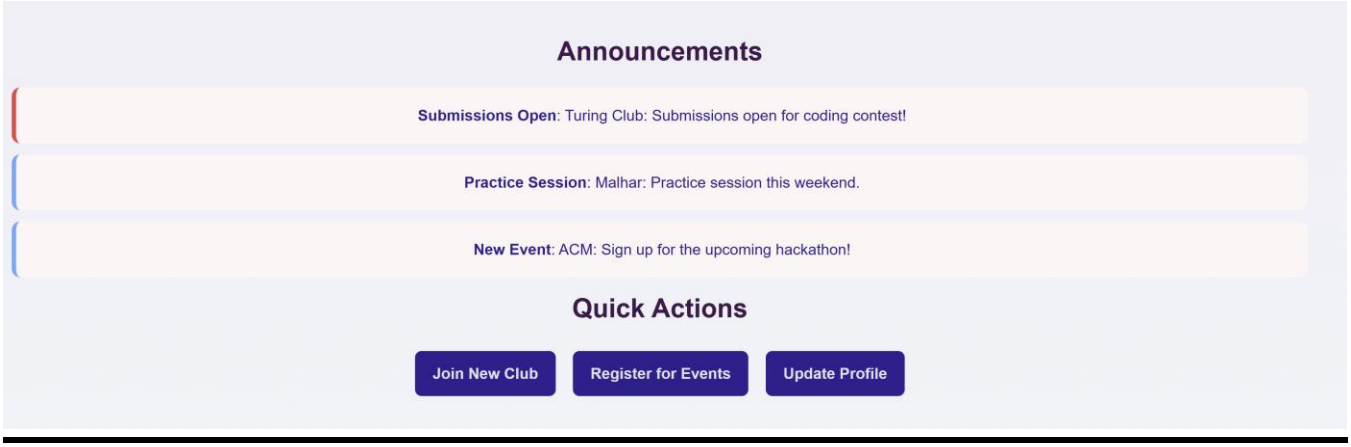
Additionally, I developed a stronger grasp of real-world applications like user-role management, event registration systems, and content organization through announcements and discussions. This project also enhanced my skills in breaking down complex requirements into manageable data structures and gave me a deeper appreciation for clean, maintainable code and database design.

Overall, the project not only refined my technical skills but also boosted my confidence in building database-driven web applications and understanding how backend logic supports frontend functionality. It was a crucial step forward in my journey as a developer.

Project Demonstration

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

id	fullname	email	username	password	created_at
1	Ashlesha Agrawal	ashlesha.522969@gmail.com	Ashle12	$2b$10$/MLR/Mrji9nDgAzsO8Gg.eSQeDPdLpuZ...	2025-04-15 19:27:56
2	bhumi Neema	bhumi30@gmail.com	Bhumi30	$2b$10$n58MfuzxGr.6ddJJ0zWNA.jH4FwGaihr...	2025-04-15 19:37:03
3	Soumya Gangrade	soumya123@gmail.com	Soumya08	$2b$10$ZZ1EdNyvtDI6/cPVmVNTz.kXoLXTusfS...	2025-04-16 00:16:06
4	Rohit Sinha	Rohit21@gmail.com	Rohit09	$2b$10$5G2ftNuzpoMA7ffsqF.ZleVRqknQYj7vT...	2025-04-16 10:46:09
5	Mahi Jaiswal	mahi12@gmail.com	mahi12	$2b$10$.GnRWOp/b1KCNputFO7EVeZvs/GLZJ1...	2025-04-16 10:55:05
6	Garv Gupta	Garv1975@gmail.com	Garv21	$2b$10$UVHW6HdCDpT8zP6W6pnvkeHKLvw/A...	2025-04-16 11:13:04
NULL	NULL	NULL	NULL	NULL	NULL

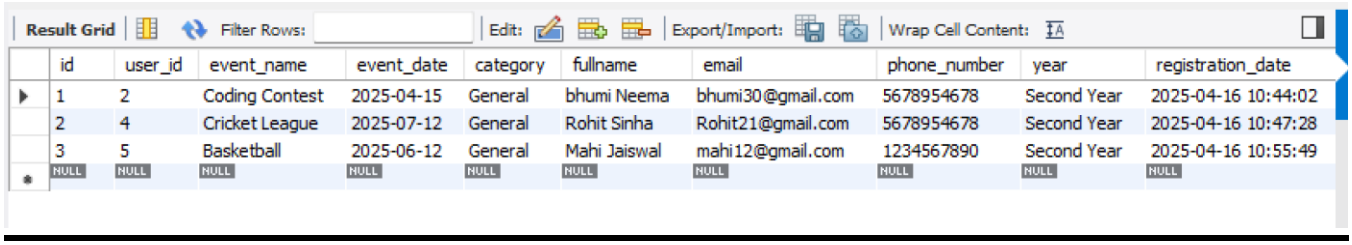


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	title	message	club_name	importance	created_at
1	Submissions Open	Turing Club: Submissions open for coding contest!	Turing	important	2025-04-16 04:14:03
2	Practice Session	Malhar: Practice session this weekend.	Malhar	normal	2025-04-16 04:14:03
3	New Event	ACM: Sign up for the upcoming hackathon!	ACM	normal	2025-04-16 04:14:03
NULL	NULL	NULL	NULL	NULL	NULL

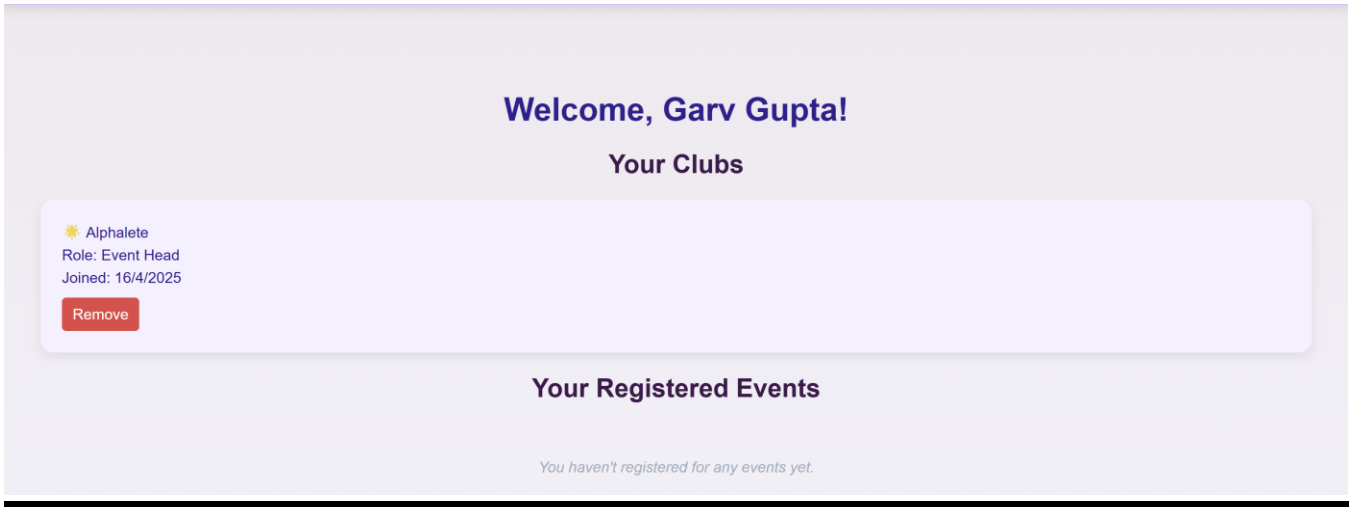
Events

- Invictus - 2025-03-20
- Victus - 2025-04-12
- Volleykick - 2025-05-12
- Basketball - 2025-06-12
- Cricket League - 2025-07-12
- Coding Contest - 2025-04-15
- Hackathon - 2025-05-20



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

id	user_id	event_name	event_date	category	fullname	email	phone_number	year	registration_date
1	2	Coding Contest	2025-04-15	General	bhumi Neema	bhumi30@gmail.com	5678954678	Second Year	2025-04-16 10:44:02
2	4	Cricket League	2025-07-12	General	Rohit Sinha	Rohit21@gmail.com	5678954678	Second Year	2025-04-16 10:47:28
3	5	Basketball	2025-06-12	General	Mahi Jaiswal	mahi12@gmail.com	1234567890	Second Year	2025-04-16 10:55:49
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



Self learning beyond classroom:

Beyond the classroom, this project encouraged us to explore and learn independently. We delved into advanced topics like user-role management, complex database relationships, and backend structuring—areas that were not covered in depth during regular lectures. We explored online resources, tutorials, and documentation to understand how real-world systems are designed and implemented. We also learned how to use tools like MySQL Workbench, GitHub, and web design frameworks, which are essential for modern development but often go beyond classroom teachings. This hands-on experience helped us connect theoretical knowledge with practical application, boosting our technical confidence and problem-solving abilities.

Learning from the project:

Through this project, we gained a strong understanding of how to design and implement a database system for a real-world application—in this case, a college club website. We learned how to identify key entities, define their attributes, and establish relationships among them to form an effective Entity-Relationship (ER) model. We also understood how to translate this conceptual design into a relational schema, applying concepts like primary keys, foreign keys, and normalization to maintain data integrity and minimize redundancy. Working collaboratively on this project helped us grasp the practical differences between ER diagrams and relational schemas, and gave us hands-on experience in organizing and structuring data for web-based platforms. Overall, it enhanced our database design skills and prepared us for more complex systems in future development tasks.

Challenges faced:

While working on this group project, we encountered several challenges. One of the major difficulties was identifying and clearly defining the relationships between different entities, especially when dealing with many-to-many relationships like membership, event participation, and poll responses. Ensuring that the relational schema remained normalized and free of redundancy while still supporting all the necessary functionalities also proved to be complex. Integrating features such as discussions, announcements, and alumni mentorship within a unified structure required a lot of brainstorming and revisions. Additionally, coordinating among team members, managing task distribution, and ensuring consistency across our design work were some team-related hurdles we had to overcome. Despite these challenges, they ultimately helped us improve our collaboration, problem-solving, and database design skills.

Conclusion:

In conclusion, this project provided us with valuable hands-on experience in designing a comprehensive and normalized database system for a real-world application. Through collaborative efforts, we successfully identified key entities, established their relationships, and created a robust relational schema suitable for a college club management platform. The process not only strengthened our understanding of database concepts like ER modeling, relational schema, and normalization but also enhanced our teamwork, problem-solving, and communication skills. This project has prepared us well for future system design and implementation tasks in both academic and professional environments.