Addis Ababa Science and Technology University

Database Design Final Project: Eder Management System Project

Done By: Sayzana Kibru

ID: ETS1249/16

Section: C

Submitted TO: Mr. Befekadu

Submission Date: May 19, 2025 G.C.

Contents

Problem Statement

In many Ethiopian communities, Eder associations serve as vital traditional support systems that provide financial, emotional, and logistical assistance during significant life events such as funerals, illnesses, and other emergencies. However, the current management of Eder operations is largely manual, relying on paper records or word-of-mouth communication, which often leads to:

- Data loss or inaccuracies
- Difficulty tracking contributions and payouts
- Lack of transparency and accountability
- Inefficient communication among members

These challenges hinder the effectiveness and trustworthiness of Eder organizations, especially as they grow in size.

Therefore, this project proposes the development of a Digital Eder Management System — a centralized platform that will:

- Digitally track members, contributions, events, and payouts
- Improve data accuracy and accessibility
- Enhance reporting, accountability, and decision-making
- Preserve and strengthen the role of Eder in a modern, digital environment

Chapter One: Introduction

The Eder Management System is a database-driven application designed to manage membership, financial contributions, payouts, and events for an Ethiopian Eder, a community-based cooperative that supports members during bereavement. The system aims to streamline record-keeping, ensure accurate financial tracking, and generate reports for transparency.

The objectives are to track member details and contributions, log payouts and associated events, enforce data integrity, and provide accessible reports. The system benefits Eder communities by automating manual processes, reducing errors, and preserving cultural practices.

Chapter Two: Database Design

Conceptual and Logical Database Design

The Eder Management System manages membership, contributions, payouts, and events. The conceptual design identifies entities, attributes, and relationships, while the logical design prepares these for MySQL implementation.

Requirements Expected

The system must:

- Track member details and contribution history.
- Record financial contributions and payouts.
- Log events and payouts.
- Generate contribution and payout reports.
- Ensure data integrity.

Entities and Attributes

Key entities include:

1. **Members**
   - ✓ member_id (Primary Key, Integer)
   - ✓ first_name (Varchar, 50)
   - ✓ last_name (Varchar, 50)
   - ✓ contact (Varchar, 10)
   - ✓ join_date (Date)
   - ✓ status (Boolean)

2. **Contributions**
   - ✓ contributer_id (Primary Key, Integer)
   - ✓ member_id (Foreign Key to Members.member_id)
   - ✓ amount (Decimal, 10,2)
   - ✓ date (Date)
   - ✓ payment_method (Varchar, 20)

3. **Payouts**

- ✓ payout_id (Primary Key, Integer)
- ✓ member_id (Foreign Key to Members.member_id)
- ✓ event_id (Foreign Key to Events.event_id)
- ✓ amount (Decimal, 10,2)
- ✓ date (Date)
- ✓ reason (Varchar, 255)

4. **Events**

- ✓ event_id (Primary Key, Integer)
- ✓ member_id (Foreign Key to Members.member_id)
- ✓ date (Date)
- ✓ type (Varchar, 50)
- ✓ description (Varchar, 250)

Relationships between Entities

- Members to Contributions: One member, many contributions.
- Members to Payouts: One member, many payouts.
- Events to Payouts: One event, many payouts.
- Members to Events: One member, many events.

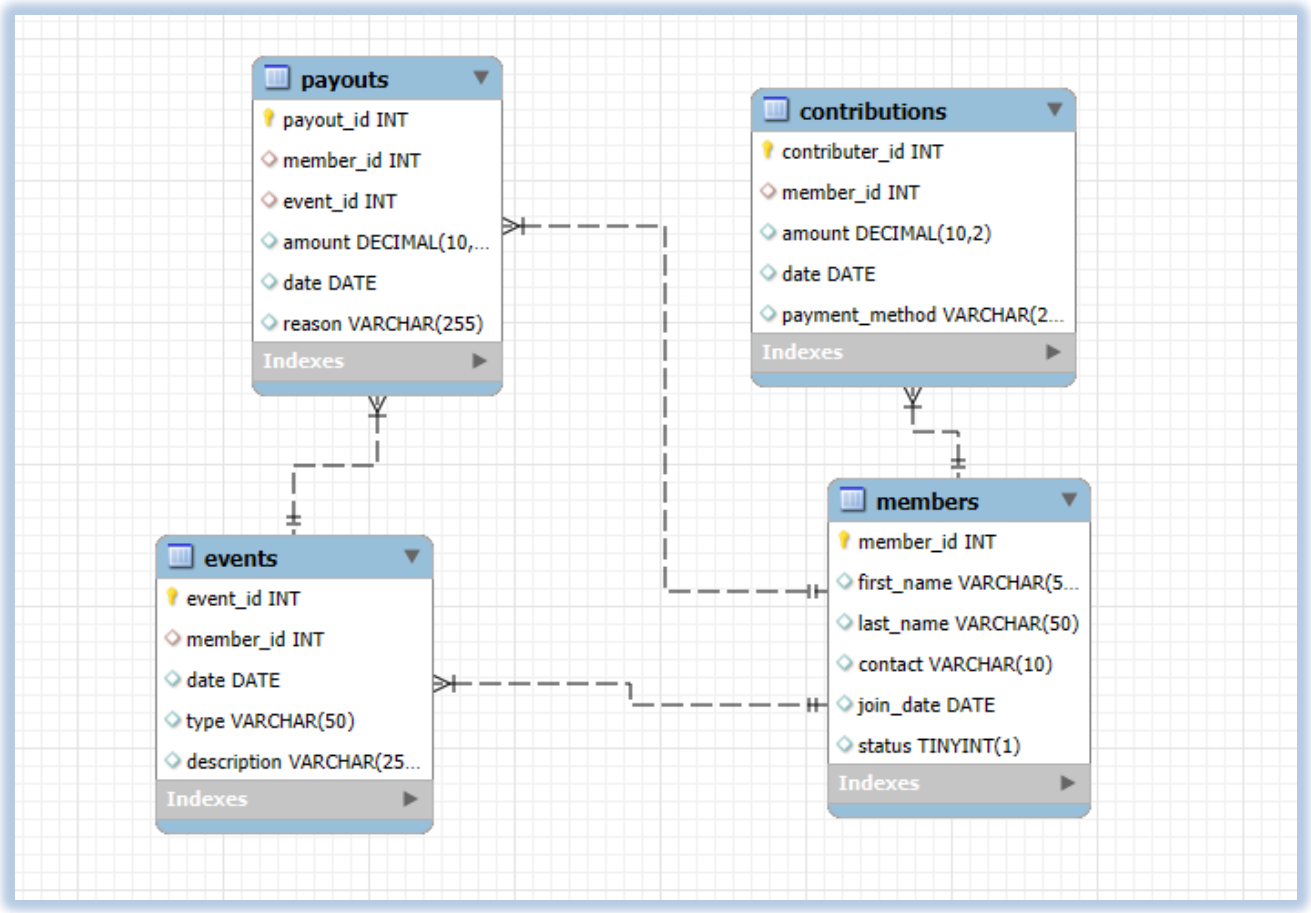
Normalization Analysis

1NF: Atomic values, no repeating groups.

2NF: No partial dependencies; single-column primary keys.

3NF: No transitive dependencies; member details stored in Members.

Entity Relationship Diagram



Physical Database Design

The physical design implements the schema in MySQL, using MySQL Workbench.

SQL Table Definitions

Members:

- member_id: INT, PRIMARY KEY, AUTO_INCREMENT
- first_name: VARCHAR(50), NOT NULL
- last_name: VARCHAR(50), NOT NULL
- contact: VARCHAR(10)
- join_date: DATE, NOT NULL
- status: BOOLEAN

Contributions:

- contrib_id: INT, PRIMARY KEY, AUTO_INCREMENT
- member_id: INT, FOREIGN KEY to Members(member_id)
- amount: DECIMAL(10,2)
- date: DATE, NOT NULL
- payment_method: VARCHAR(20)

Payouts:

- payout_id: INT, PRIMARY KEY, AUTO_INCREMENT
- member_id: INT, FOREIGN KEY to Members(member_id)
- event_id: INT, FOREIGN KEY to Events(event_id)
- amount: DECIMAL(10,2)
- date: DATE
- reason: VARCHAR(255)

Events:

- event_id: INT, PRIMARY KEY, AUTO_INCREMENT
- member_id: INT, FOREIGN KEY to Members(member_id)
- date: DATE, NOT NULL
- type: VARCHAR(50)
- description: VARCHAR(250)

Constraints & Integrity Rules

- Primary Keys: Ensure uniqueness.
- Foreign Keys: Enforce relationships.
- NOT NULL: Ensure non-empty fields

Design of Inputs, Outputs, and Reports

Inputs (Form Designs)

A. Add Member:

Fields: First Name, Last Name, Contact, Join Date, Status

Validation: Names required (non-empty), valid contact (10 digits and numeric).

B. Record Contribution:

Fields: Member, Amount, Date, Payment Method (Cash, Bank, Mobile Transfer, etc)

Validation: None

C. Request Payout:

Fields: Member, Event, Amount, Date, Reason

Validation: None

D. Log Event:

Fields: Member, Date, Type, Description

Validation: None

Outputs

- **Member List:** member_id, name, status.
- **Contribution History:** amount, date.
- **Filtered Views**: Overdue contributions.

Reports (As Views)

1. **Total Member Contributions**
   - **Purpose:** Replaces the Contribution Report to show each member's cumulative contributions.
   - **Fields:** member_id, first_name, last_name, total_contributed.

2. **Member Payout Summary**
   - Purpose: Replaces the Payout Report to track total benefits disbursed per member.
   - Fields: member_id, first_name, last_name, total_payout, number_of_payouts.

3. **Recent Events**
   - Purpose: Displays the 3 most recent events (e.g., funerals, emergencies).
   - Fields: event_id, first_name, last_name, date, type, description.

4. **Payouts by Event Type**

- Purpose: Analyzes disbursements by category (e.g., "Funeral)
- Fields: event_type, total_payouts, total_amount_paid.

Chapter Three: Implementation

The system is implemented using MySQL 8.0.42 for the database. The database is hosted locally, with tables created as defined.

Table Definition

Creating Members Table:

```
CREATE TABLE Members (
  member_id INT PRIMARY KEY AUTO_INCREMENT,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  contact VARCHAR(10),
  join_date DATE,
  status BOOLEAN
);
```

Creating Contributions Table:

```
CREATE TABLE Contributions (
  contributer_id INT PRIMARY KEY AUTO_INCREMENT,
  member_id INT,
  amount DECIMAL(10,2),
  date DATE,
  payment_method VARCHAR(20),
  FOREIGN KEY (member_id) REFERENCES Members(member_id)
);
```

Creating Events Table:

```
CREATE TABLE Events (
  event_id INT PRIMARY KEY AUTO_INCREMENT,
  member_id INT,
  date DATE,
  type VARCHAR(50),
  description VARCHAR(250),
  FOREIGN KEY (member_id) REFERENCES Members(member_id)
);
```

Creating Payouts Table:

```
CREATE TABLE Payouts (
  payout_id INT PRIMARY KEY AUTO_INCREMENT,
  member_id INT,
  event_id INT,
  amount DECIMAL(10,2),
  date DATE,
  reason VARCHAR(255),
  FOREIGN KEY (member_id) REFERENCES Members(member_id),
  FOREIGN KEY (event_id) REFERENCES Events(event_id)
);
```

Sample Data Insertion

Includes 5 members, 8 contributions, 3 events and 3 payouts for testing.

Insert 5 Members:

```
INSERT INTO Members (first_name, last_name, contact, join_date, status)
VALUES
    ('Abebe', 'Kebede', '0912345678', '2025-01-01', 1),
    ('Meron', 'Tadesse', '0998765432', '2025-02-01', 1),
    ('Selam', 'Alemu', '0923456789', '2025-03-01', 0),
    ('Yohannes', 'Getachew', '0934567890', '2025-04-01', 1),
    ('Lidya', 'Hailu', '0945678901', '2025-05-01', 1);
```

Insert 8 Contributions:

```
INSERT INTO Contributions (member_id, amount, date, payment_method)
VALUES
    (1, 50.00, '2025-05-10', 'Cash'),
    (1, 75.00, '2025-05-15', 'Mobile Money'),
    (2, 100.00, '2025-05-12', 'Bank'),
    (2, 60.00, '2025-05-18', 'Cash'),
    (3, 80.00, '2025-05-11', 'Mobile Money'),
    (4, 120.00, '2025-05-13', 'Bank'),
    (4, 55.00, '2025-05-16', 'Cash'),
    (5, 90.00, '2025-05-17', 'Mobile Money');
```

Insert 3 Events:

```
INSERT INTO Events (member_id, date, type, description)
VALUES
    (1, '2025-05-14', 'Funeral', 'Funeral for family member'),
    (2, '2025-05-16', 'Memorial', 'Community memorial event'),
    (4, '2025-05-18', 'Funeral', 'Funeral for relative');
```

Insert 3 Payouts:

```
INSERT INTO Payouts (member_id, event_id, amount, date, reason)
VALUES
    (1, 1, 500.00, '2025-05-14', 'Funeral expenses'),
    (2, 2, 300.00, '2025-05-16', 'Memorial support'),
    (4, 3, 450.00, '2025-05-18', 'Funeral costs');
```

Test Data Retrieval (Basic SELECT Queries)

List All Members:

```
mysql> SELECT member_id, first_name, last_name, contact, join_date, status
    -> FROM Members;
+-----------+------------+-----------+------------+------------+--------+
| member_id | first_name | last_name | contact    | join_date  | status |
+-----------+------------+-----------+------------+------------+--------+
|         1 | Abebe      | Kebede    | 0912345678 | 2025-01-01 |      1 |
|         2 | Meron      | Tadesse   | 0998765432 | 2025-02-01 |      1 |
|         3 | Selam      | Alemu     | 0923456789 | 2025-03-01 |      0 |
|         4 | Yohannes   | Getachew  | 0934567890 | 2025-04-01 |      1 |
|         5 | Lidya      | Hailu     | 0945678901 | 2025-05-01 |      1 |
+-----------+------------+-----------+------------+------------+--------+
5 rows in set (0.00 sec)
```

List All Contributions:

```
mysql> SELECT contributer_id, member_id, amount, date, payment_method
    -> FROM Contributions;
+----------------+-----------+--------+------------+----------------+
| contributer_id | member_id | amount | date       | payment_method |
+----------------+-----------+--------+------------+----------------+
|              1 |         1 |  50.00 | 2025-05-10 | Cash           |
|              2 |         1 |  75.00 | 2025-05-15 | Mobile Money   |
|              3 |         2 | 100.00 | 2025-05-12 | Bank           |
|              4 |         2 |  60.00 | 2025-05-18 | Cash           |
|              5 |         3 |  80.00 | 2025-05-11 | Mobile Money   |
|              6 |         4 | 120.00 | 2025-05-13 | Bank           |
|              7 |         4 |  55.00 | 2025-05-16 | Cash           |
|              8 |         5 |  90.00 | 2025-05-17 | Mobile Money   |
+----------------+-----------+--------+------------+----------------+
8 rows in set (0.00 sec)
```

List Events with Member Details:

```
mysql> SELECT e.event_id, e.type, e.date, e.description, m.first_name, m.last_name
    -> FROM Events e
    -> JOIN Members m ON e.member_id = m.member_id;
+----------+----------+------------+--------------------------+------------+-----------+
| event_id | type     | date       | description              | first_name | last_name |
+----------+----------+------------+--------------------------+------------+-----------+
|        1 | Funeral  | 2025-05-14 | Funeral for family member | Abebe      | Kebede    |
|        2 | Memorial | 2025-05-16 | Community memorial event  | Meron      | Tadesse   |
|        3 | Funeral  | 2025-05-18 | Funeral for relative      | Yohannes   | Getachew  |
+----------+----------+------------+--------------------------+------------+-----------+
3 rows in set (0.00 sec)
```

List Payouts with Event and Member Info:

```
mysql> SELECT p.payout_id, p.amount, p.date, p.reason, m.first_name, m.last_name, e.type
    -> FROM Payouts p
    -> JOIN Members m ON p.member_id = m.member_id
    -> JOIN Events e ON p.event_id = e.event_id;
+-----------+--------+------------+------------------+------------+-----------+----------+
| payout_id | amount | date       | reason           | first_name | last_name | type     |
+-----------+--------+------------+------------------+------------+-----------+----------+
|         1 | 500.00 | 2025-05-14 | Funeral expenses | Abebe      | Kebede    | Funeral  |
|         2 | 300.00 | 2025-05-16 | Memorial support | Meron      | Tadesse   | Memorial |
|         3 | 450.00 | 2025-05-18 | Funeral costs    | Yohannes   | Getachew  | Funeral  |
+-----------+--------+------------+------------------+------------+-----------+----------+
3 rows in set (0.00 sec)
```

Test Data Integrity (Constraints)

Ensure the database enforces constraints (PRIMARY KEY, FOREIGN KEY, UNIQUE).

Test UNIQUE Constraint on contact: Attempts to insert a duplicate contact (matches Abebe's).

```
mysql> INSERT INTO Members (first_name, last_name, contact, join_date, status)
    -> VALUES ('Test', 'User', '0912345678', '2025-06-01', 1);
ERROR 1062 (23000): Duplicate entry '0912345678' for key 'members.contact'
```

Test FOREIGN KEY Constraint (Invalid member_id): Tries to insert a contribution with a non-existent member_id.

```
mysql> INSERT INTO Payouts (member_id, event_id, amount, date, reason)
    -> VALUES (1, 999, 200.00, '2025-05-20', 'Test payout');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`project_ider`.`payouts`, CONSTRAI
NT `payouts_ibfk_2` FOREIGN KEY (`event_id`) REFERENCES `events` (`event_id`))
mysql>
```

Views To Generate Reports

Total Contributions

- Displays the total amount contributed by each member.

```sql
CREATE VIEW TotalContributions AS
SELECT m.member_id, m.first_name, m.last_name, SUM(c.amount) AS total_contributed
FROM Members m
JOIN Contributions c ON m.member_id = c.member_id
GROUP BY m.member_id, m.first_name, m.last_name;
```

Result of View Query:

```
mysql> SELECT * FROM TotalContributions;
+-----------+------------+-----------+--------------------+
| member_id | first_name | last_name | total_contributed |
+-----------+------------+-----------+--------------------+
|         1 | Abebe      | Kebede    |             125.00 |
|         2 | Meron      | Tadesse   |             160.00 |
|         3 | Selam      | Alemu     |              80.00 |
|         4 | Yohannes   | Getachew  |             175.00 |
|         5 | Lidya      | Hailu     |              90.00 |
+-----------+------------+-----------+--------------------+
5 rows in set (0.00 sec)
```

Payout Summary

- Shows the total payouts received and the number of payouts per member.

```sql
CREATE VIEW PayoutSummary AS
SELECT  m.member_id, m.first_name, m.last_name, SUM(p.amount) AS total_payout, COUNT(p.payout_id) AS number_of_payouts
FROM Members m
JOIN Payouts p ON m.member_id = p.member_id
GROUP BY m.member_id, m.first_name, m.last_name;
```

Result of View Query:

```
mysql> SELECT * FROM PayoutSummary;
+-----------+------------+-----------+--------------+--------------------+
| member_id | first_name | last_name | total_payout | number_of_payouts |
+-----------+------------+-----------+--------------+--------------------+
|         1 | Abebe      | Kebede    |       500.00 |                  1 |
|         2 | Meron      | Tadesse   |       300.00 |                  1 |
|         4 | Yohannes   | Getachew  |       450.00 |                  1 |
+-----------+------------+-----------+--------------+--------------------+
3 rows in set (0.00 sec)
```

Recent Events

- Lists the most recent 3 events recorded in the system, including event type and member involved.

```
CREATE VIEW RecentEvents AS
SELECT e.event_id, m.first_name, m.last_name, e.date, e.type, e.description
FROM Events e
JOIN Members m ON e.member_id = m.member_id
ORDER BY e.date DESC
LIMIT 3;
```

Result of View Query:

```
mysql> SELECT * FROM RecentEvents;
+----------+------------+-----------+------------+----------+---------------------------+
| event_id | first_name | last_name | date       | type     | description               |
+----------+------------+-----------+------------+----------+---------------------------+
|        3 | Yohannes   | Getachew  | 2025-05-18 | Funeral  | Funeral for relative      |
|        2 | Meron      | Tadesse   | 2025-05-16 | Memorial | Community memorial event  |
|        1 | Abebe      | Kebede    | 2025-05-14 | Funeral  | Funeral for family member |
+----------+------------+-----------+------------+----------+---------------------------+
3 rows in set (0.00 sec)
```

Payout by Event Type

- Summarizes the total payouts grouped by each event type.

```
CREATE VIEW PayoutByEventType AS
SELECT  e.type AS event_type, COUNT(p.payout_id) AS total_payouts, SUM(p.amount) AS total_amount_paid
FROM Payouts p
JOIN Events e ON p.event_id = e.event_id
GROUP BY e.type;
```

Result of View Query:

```
mysql> SELECT * FROM PayoutByEventType;
+------------+---------------+--------------------+
| event_type | total_payouts | total_amount_paid  |
+------------+---------------+--------------------+
| Funeral    |             2 |             950.00 |
| Memorial   |             1 |             300.00 |
+------------+---------------+--------------------+
2 rows in set (0.00 sec)
```

Future Enhancements and Improvements

Key improvements that could be implemented in future versions of the system:

1. User Interface & Accessibility Enhancements

- Web-based portal/ Mobile application
- Receive notifications
- Receive payment reminders

2. Advanced Financial Features

- Mobile money integration (M-Pesa, Telebirr, etc.)
- Bank API connections for direct debit options
- Automated late payment reminders
- Grace period tracking
- Progressive penalty calculations

3. Enhanced Reporting & Analytics

- Member engagement metrics
- Export to PDF/Excel
- Annual statement generator for tax purposes

4. Event & Benefit Management Improvements

- Document upload system for: Death certificates
- Event calendar integration showing upcoming meetings
- Contribution deadlines

Reference:

[1] A. Dereje, "Traditional Mutual Aid Societies in Ethiopia: The Role of Idir," Journal of Ethiopian Studies, vol. 45, pp. 112–130, 2012.

[2] Ethiopian Federal Cooperative Agency, "Guidelines for Community-Based Insurance Systems," Addis Ababa, 2018.

[3] MySQL Workbench 8.0 User Guide, Oracle Corp., 2023. [Online]. Available: https://dev.mysql.com/doc/workbench/en/

[4] IEEE Standards Association, "IEEE 830-1998: Recommended Practice for Software Requirements Specifications," 1998.

Annex

Survey and interview questions

Questions tailored to uncover pain points, workflows, and desired outcomes for creating an effective digital database solution.

- How do you currently track member contributions, and what difficulties arise in maintaining accurate records?
- What challenges do you face in managing contributions and benefits?
- What reporting would help your Eder management committee?
- What specific data points do you collect for each contribution (e.g., member name, date, amount, purpose), and are there any gaps in the data you wish you could capture?
- What methods (e.g., paper records, spreadsheets, or software) do you currently use to record member contributions, and how often are these records updated?
- What are the most common errors you encounter when recording contributions (e.g., human error, incomplete data, or system limitations)?
- What types of reports does the elder management committee currently rely on to make decisions about contributions and benefits?

Sample SQL statements or relevant code

```sql
-- 1. Insert sample data into Members table
INSERT INTO Members (first_name, last_name, contact, join_date, status)
VALUES
    ('John', 'Doe', '1234567890', '2025-01-01', TRUE),
    ('Jane', 'Smith', '0987654321', '2025-02-15', TRUE),
    ('Alice', 'Johnson', '1122334455', '2025-03-10', TRUE);
```

```sql
-- 2. Insert a sample contribution for a member
INSERT INTO Contributions (member_id, amount, date, payment_method)
VALUES
    (1, 100.50, '2025-04-01', 'Cash'),
    (2, 250.00, '2025-04-05', 'Bank Transfer'),
    (1, 75.25, '2025-05-01', 'Mobile Payment');


 -- 3. Retrieve total contributions per member with names
 SELECT
     m.member_id,
     m.first_name,
     m.last_name,
     count(*) AS total_contributions
 FROM Members m
 LEFT JOIN Contributions c ON m.member_id = c.member_id
 GROUP BY m.member_id, m.first_name, m.last_name
 ORDER BY total_contributions DESC;


-- 4. List all events with member details
SELECT
    e.event_id,
    e.date,
    e.type,
    e.description,
    m.first_name,
    m.last_name
FROM Events e
JOIN Members m ON e.member_id = m.member_id
WHERE e.date >= '2025-01-01'
ORDER BY e.date;


-- 5. Update member status to inactive
UPDATE Members
SET status = FALSE
WHERE member_id = 3;
```

```sql
-- 6. Find payouts associated with a specific event type
SELECT
    p.payout_id,
    p.amount,
    p.date,
    p.reason,
    m.first_name,
    m.last_name,
    e.type,
    e.description
FROM Payouts p
JOIN Members m ON p.member_id = m.member_id
JOIN Events e ON p.event_id = e.event_id
WHERE e.type = 'Charity Drive'
ORDER BY p.date;
```

Conclusion

The Eder Management System project successfully addresses the challenges faced by traditional Ethiopian Eder associations by transitioning from manual, paper-based processes to a streamlined, digital solution. By leveraging a well-designed relational database, the system ensures accurate tracking of member contributions, payouts, and events while enhancing transparency, accountability, and efficiency.

Future enhancements, such as mobile integration, advanced financial tools, and expanded analytics, could further modernize Eder operations. This project not only preserves the cultural significance of Eder associations but also equips them with the tools needed to thrive in a digital era. By bridging tradition and technology, the Eder Management System lays a foundation for sustainable community support systems in Ethiopia.

Final Words: *This project has been an invaluable learning experience, deepening my understanding of database design, real-world problem-solving, and the intersection of technology and tradition. Through the Eder Management System, I gained hands-on experience with MySQL, normalization and constraint enforcement, while also appreciating the importance of aligning technical solutions with cultural practices.*

Moving forward, I aim to refine my skills in automation, user interface design, and scalable architectures—lessons I'll carry into future projects. Above all, this work reinforced how technology can empower communities, a principle I hope to apply in even greater ways.

This project wasn't just about building a database—it was about growing as a developer and problem-solver.