



Disaster Relief and Donation Management System

Project Report

Group 12

AS20240395- K I R Abeysekara

AS20240427- B P R Sachintha

AS20240404- A H M D Silva

AS20240577- N G Sachindu Sathsara

AS20240546- T H Sandaruwan

AS20240538- S M Wijesekara

AS20240559-E.K.S Govinda

CSC 1042 Database Management Systems

Lecturer in Charge: Dr. Surani Tissera

Department: Department of Computer Science

Submission Date: 01.02.2026

Table of Contents

1. Problem Statement	3
2. System Scope and Limitations	3
3. Conceptual Design	4
3.1 Assumptions	4
3.2 Entity-Relationship (ER) Diagram	6
4. Logical Design (Relational Schema)	7
5. Implemented Database	8
6. Populated Database Tables	8
Table: Donor	8
Table: Donation	9
Table: DonationItem	9
Table: Item	10
Table: Inventory	10
Table: Beneficiary	11
Table: Distribution	11
Table: DistributionItem	12
Table: User	12
7. Sample CRUD Operations	13
Table Creation	13
Table: Donor	13
Table: Donation	14
INSERT Operation (Create)	14
Table: Donor	14
Table: Donation	15
SELECT Operation (Read)	15
Table: Donor	15
Table: Donation	16
DELETE Operation (Delete)	16
Table: Donor	16
Table: Donation	17
8. User Interface Implementation	17
8.1 Login Interface	17
8.2 Dashboard	18
8.3 Donation Entry Form	18
8.4 Inventory View	18
9. Conclusion	19
10. Individual Contributions	20

1. Problem Statement

In the aftermath of natural disasters, the immediate management of relief efforts is often chaotic and inefficient. Manual methods of tracking donations, inventory, and distributions lead to significant data inconsistencies, loss of resources, and uneven aid distribution. There is often a lack of transparency regarding which donors have contributed, what items are currently in stock, and which beneficiaries have already received aid.

The Solution: We have designed and implemented the **Disaster Relief and Donation Management System**. This relational database system streamlines the workflow by digitalizing the tracking process. It ensures:

- **Donors** and their specific **Donations** are accurately recorded and retrievable.
- **Inventory** is automatically updated based on incoming donations and outgoing distributions.
- **Beneficiaries** are tracked to prevent duplicate distributions and ensure fair aid allocation.
- **Users** (Staff/Admins) have secure access to manage records.

2. System Scope and Limitations

This project focuses on designing and implementing a Disaster Relief and Donation Management System with primary emphasis on relational database design and SQL operations.

System Scope

The scope of the system includes:

- Designing a relational database schema using ER modeling and logical design principles.
- Defining primary keys and foreign key relationships between entities such as Donor, Donation, Inventory, Beneficiary, Distribution, and Item.
- Executing SQL DML operations (INSERT, SELECT, UPDATE, DELETE) to manage data in the database.
- Demonstrating database functionality using SQL queries executed directly in phpMyAdmin.
- Providing a basic PHP-based user interface to trigger selected SQL operations for ease of interaction and demonstration.
- Managing inventory and distribution data primarily through insert and select operations, reflecting real-world transactional behavior.

3. Conceptual Design

3.1 Assumptions

The following assumptions were made during the creation of the Entity-Relationship (ER) Diagram:

1) Each donor is uniquely identifiable

Assumption:

Each donor is uniquely identified by a system-generated DonorID.

Reason:

Donor names and contact numbers may not be unique. Therefore, a surrogate primary key is required to uniquely identify each donor.

Impact on ER Diagram:

DonorID is used as the primary key of the **Donor** entity.

2) A donor can make multiple donations

Assumption:

A single donor can make multiple donations over time, while each donation is associated with exactly one donor.

Reason:

This reflects real-world donation behavior where donors may contribute more than once.

Impact on ER Diagram:

A one-to-many relationship exists between **Donor** and **Donation**, with DonorID included as a foreign key in the **Donation** entity.

3) Inventory items are centrally managed

Assumption:

Inventory represents relief items stored and managed by the organization, and each item is uniquely identifiable.

Reason:

Centralized inventory management is required to accurately track available relief items.

Impact on ER Diagram:

The **Inventory** entity contains a unique primary key (InventoryID) and maintains item quantity information.

4) Inventory quantities change through system operations

Assumption:

Inventory quantities may change due to donations received or distributions made.

Reason:

Inventory levels must reflect real-time stock availability.

Impact on ER Diagram:

The **Inventory** entity allows controlled quantity updates through system operations.

5) Distribution records represent relief allocation events

Assumption:

Distribution records capture details of relief items distributed to beneficiaries and are mainly used for record-keeping.

Reason:

Distribution data should not be frequently modified once recorded to ensure data integrity.

Impact on ER Diagram:

The **Distribution** entity is modeled as a separate transactional entity with limited update and delete operations.

6) Each entity uses a single-attribute primary key

Assumption:

All entities use a single-column primary key.

Reason:

Single-attribute primary keys simplify database design and improve clarity.

Impact on ER Diagram:

No composite primary keys were used in the database design.

7) User authentication is outside the project scope

Assumption:

The system supports authenticated users with predefined roles (admin and staff), where administrative users have elevated privileges compared to staff users.

Reason:

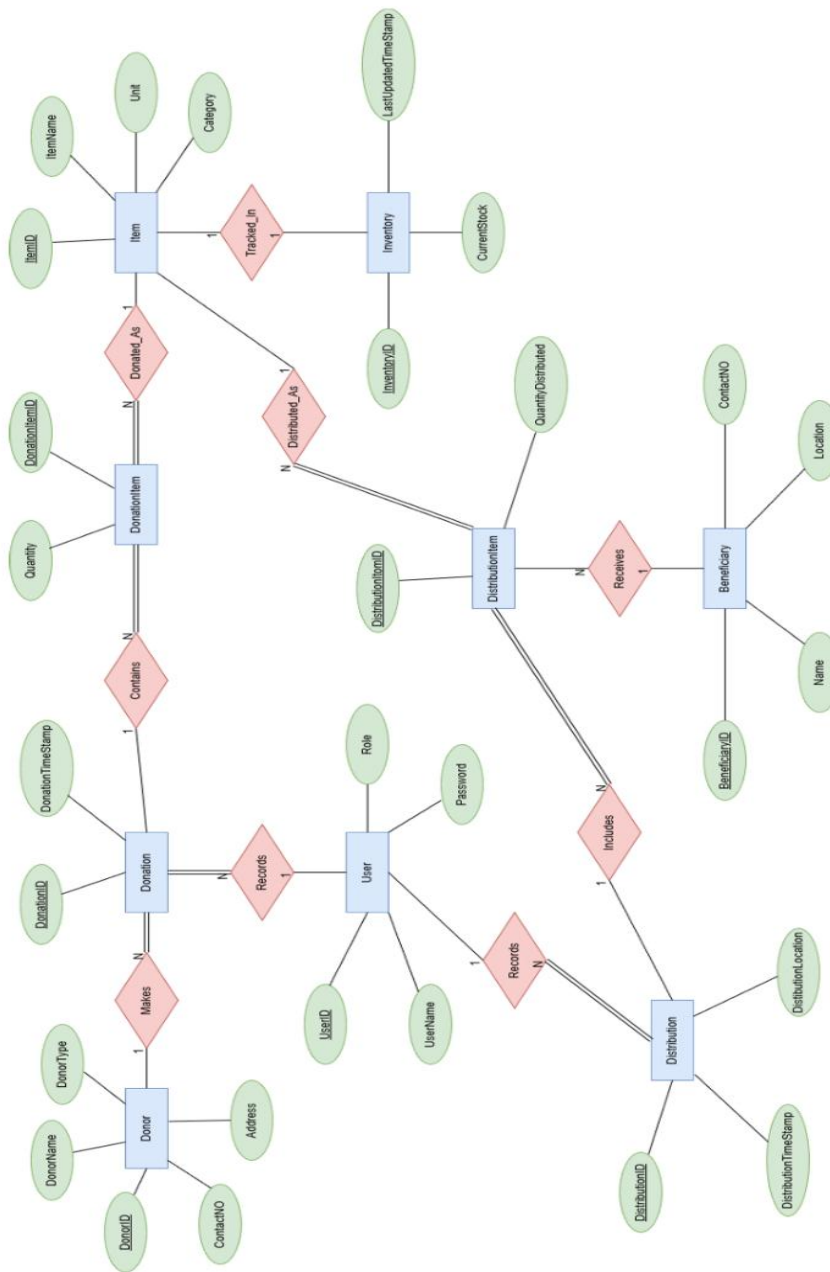
To ensure data security and integrity, role-based access control is enforced at the application level. Editing and deletion operations are restricted to administrative users, while staff users are limited to data entry and viewing functions.

Impact on ER Diagram:

The User entity includes a role attribute to represent different user types, while permission enforcement is implemented at the application level.

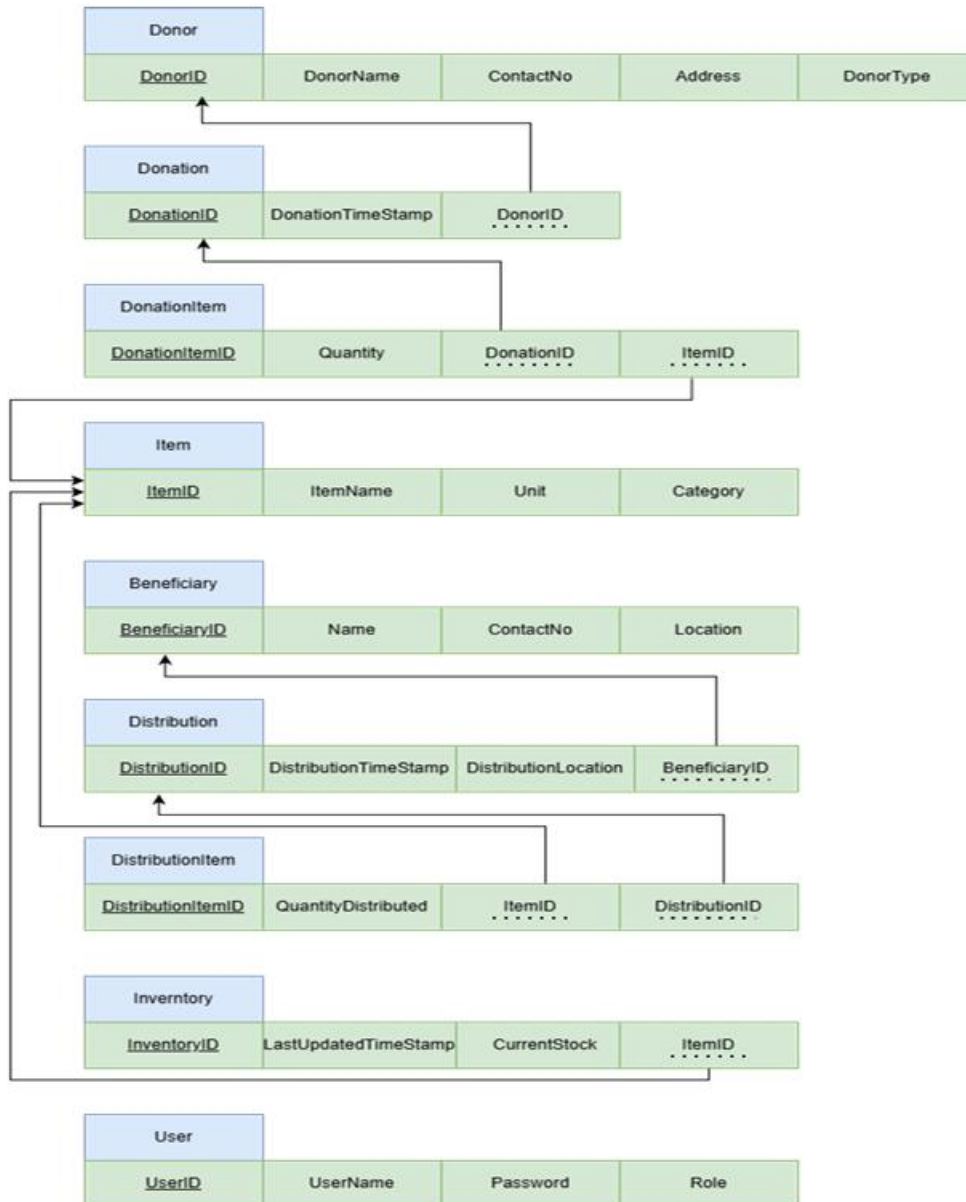
3.2 Entity-Relationship (ER) Diagram

The ER Diagram below illustrates the conceptual model of the system, defining the entities and their relationships.

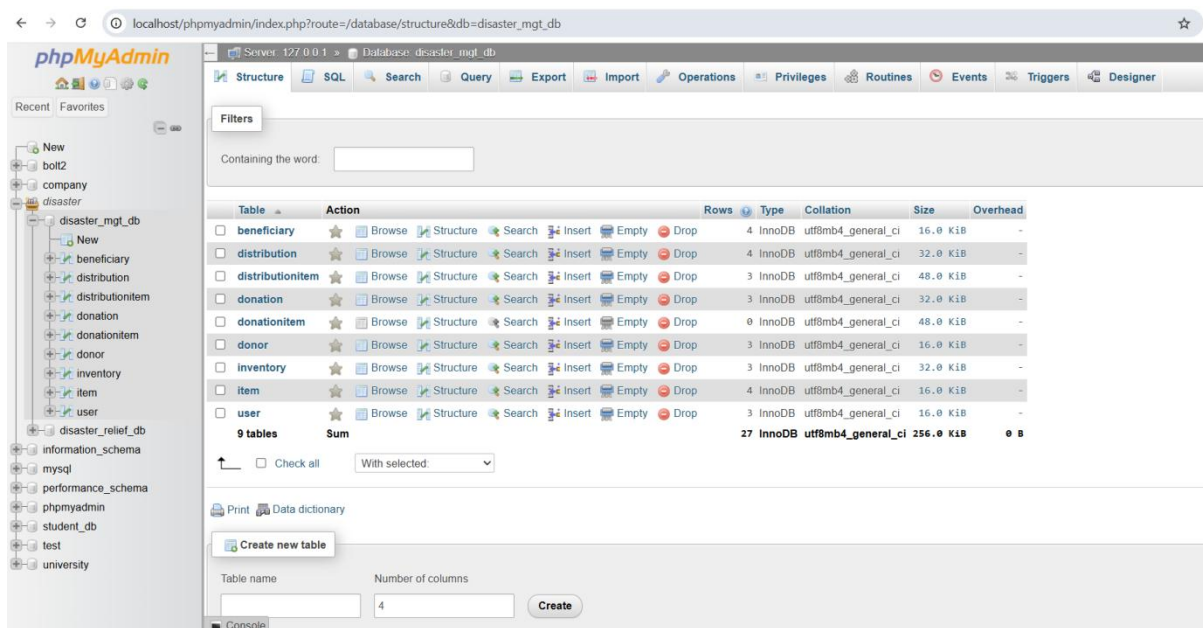


4. Logical Design (Relational Schema)

The conceptual design was converted into the following Relational Schema. Primary Keys (PK) and Foreign Keys (FK) are identified to maintain referential integrity.



5. Implemented Database



6. Populated Database Tables

The following screenshots demonstrate that the database has been successfully created and populated with data.

Table: Donor

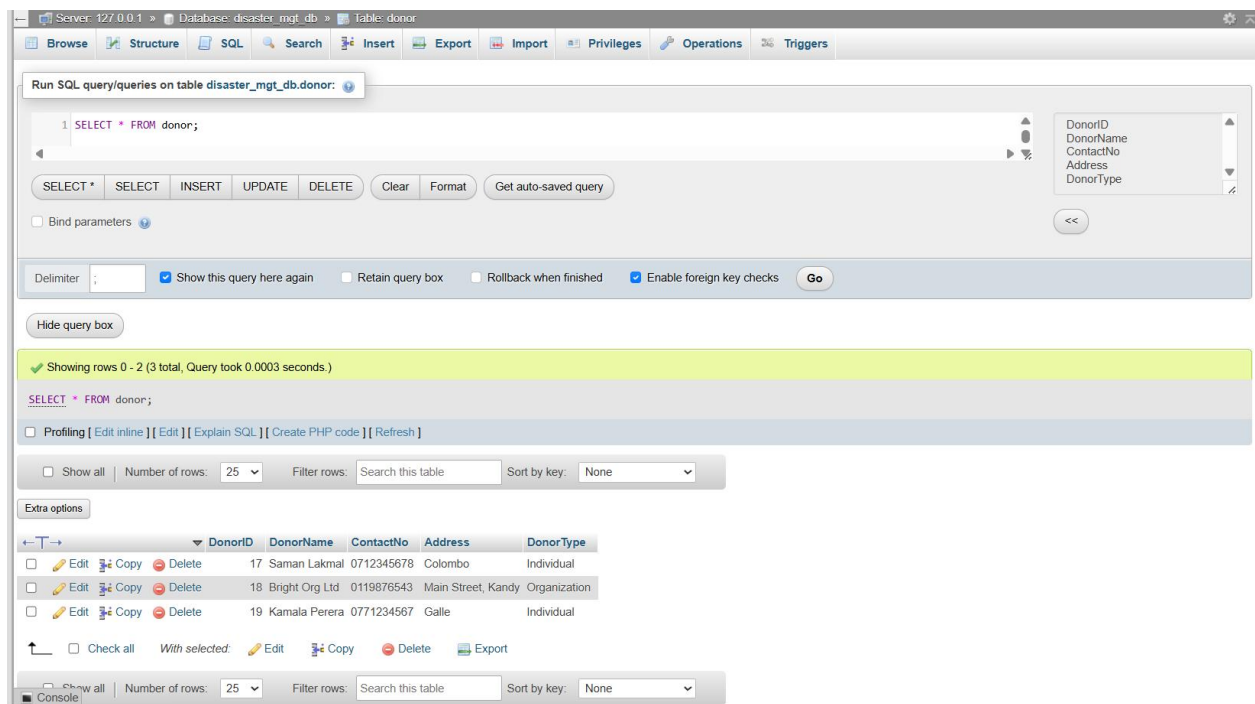


Table: Donation

The screenshot shows the phpMyAdmin interface for the 'disaster_mgt_db' database, specifically the 'donation' table. The 'Structure' tab is active, displaying the table's schema with columns: DonationID, DonationTimeStamp, and DonorID. Below the structure, the 'SQL' tab shows a query: `SELECT * FROM donation;`. The 'Data' tab is also visible, showing three rows of data. The interface includes various navigation and query execution tools.

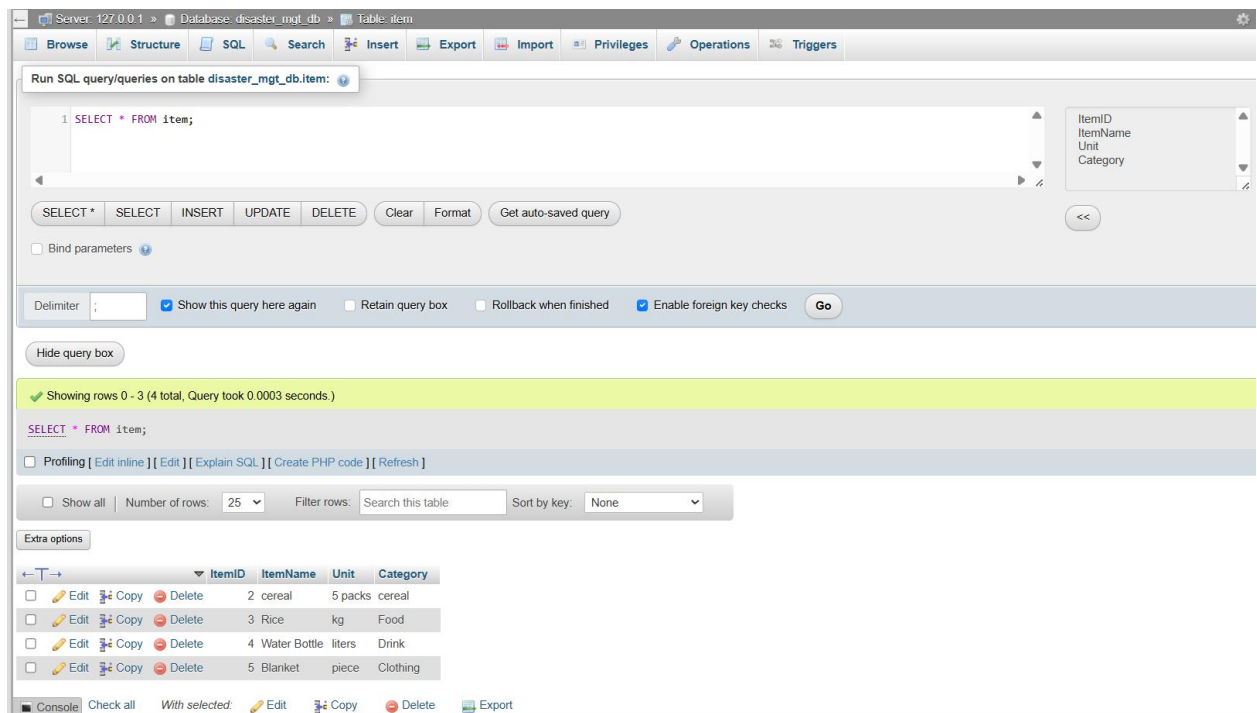
DonationID	DonationTimeStamp	DonorID
10	2026-01-27 10:00:00	17
11	2026-01-27 11:00:00	18
12	2026-01-27 12:00:00	19

Table: DonationItem

The screenshot shows the phpMyAdmin interface for the 'disaster_mgt_db' database, specifically the 'donationitem' table. The 'Structure' tab is active, displaying the table's schema with columns: DonationItemID, Quantity, DonationID, and ItemID. Below the structure, the 'SQL' tab shows a query: `SELECT * FROM donationitem;`. The 'Data' tab is also visible, showing four rows of data. The interface includes various navigation and query execution tools.

DonationItemID	Quantity	DonationID	ItemID
1	1	10	2
2	5	10	2
3	20	12	4
4	15	11	3

Table: Item



Run SQL query/queries on table disaster_mgt_db.item:

```
1 SELECT * FROM item;
```

ItemID
ItemName
Unit
Category

SELECT * SELECT INSERT UPDATE DELETE Clear Format Get auto-saved query

☐ Bind parameters

Delimiter ; ☒ Show this query here again ☐ Retain query box ☐ Rollback when finished ☒ Enable foreign key checks Go

Hide query box

Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

```
SELECT * FROM item;
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

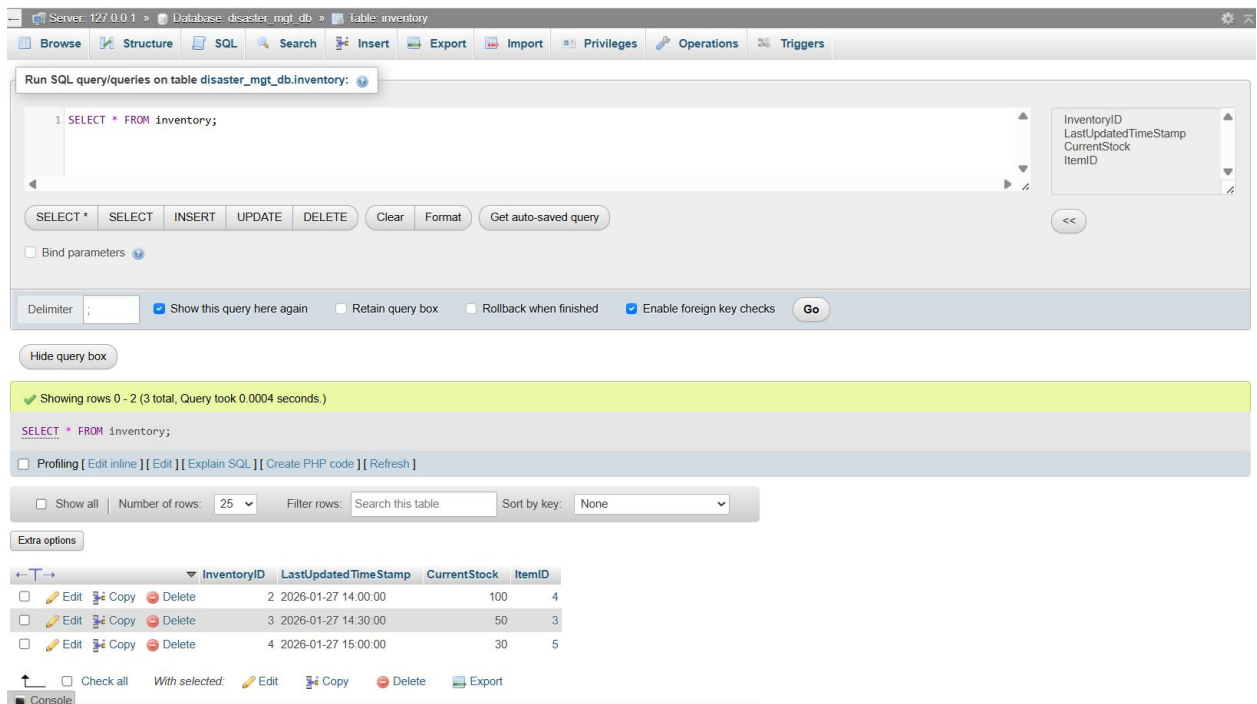
☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	ItemID	ItemName	Unit	Category
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	cereal	5 packs	cereal
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	Rice	kg	Food
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	Water Bottle	liters	Drink
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	Blanket	piece	Clothing

Console Check all With selected: Edit Copy Delete Export

Table: Inventory



Run SQL query/queries on table disaster_mgt_db.inventory:

```
1 SELECT * FROM inventory;
```

InventoryID
LastUpdatedTimeStamp
CurrentStock
ItemID

SELECT * SELECT INSERT UPDATE DELETE Clear Format Get auto-saved query

☐ Bind parameters

Delimiter ; ☒ Show this query here again ☐ Retain query box ☐ Rollback when finished ☒ Enable foreign key checks Go

Hide query box

Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

```
SELECT * FROM inventory;
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	InventoryID	LastUpdated Time Stamp	CurrentStock	ItemID
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2026-01-27 14:00:00	100	4
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	2026-01-27 14:30:00	50	3
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	2026-01-27 15:00:00	30	5

Console Check all With selected: Edit Copy Delete Export

Table: Beneficiary

Server: 127.0.0.1 » Database: disaster_mgt_db » Table: beneficiary

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Run SQL query/queries on table disaster_mgt_db.beneficiary:

```
1 SELECT * FROM beneficiary;
```

SELECT * SELECT INSERT UPDATE DELETE Clear Format Get auto-saved query

☐ Bind parameters

Delimiter: ; ☒ Show this query here again ☐ Retain query box ☐ Rollback when finished ☒ Enable foreign key checks Go

Hide query box

Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

```
SELECT * FROM beneficiary;
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	BeneficiaryID	Name	ContactNo	Location
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	Saman Silva	0712345678	Kandy Relief Camp 1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	Sithumini Gamage	0711111111	Colombo relief camp
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	A D Kelum	0722222222	Kandy relief camp 2
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	Nayani Silva	0733333333	Galle Town Hall

Table: Distribution

Server: 127.0.0.1 » Database: disaster_mgt_db » Table: distribution

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Run SQL query/queries on table disaster_mgt_db.distribution:

```
1 SELECT * FROM distribution;
```

SELECT * SELECT INSERT UPDATE DELETE Clear Format Get auto-saved query

☐ Bind parameters

Delimiter: ; ☒ Show this query here again ☐ Retain query box ☐ Rollback when finished ☒ Enable foreign key checks Go

Hide query box

Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

```
SELECT * FROM distribution;
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	DistributionID	DistributionTimeStamp	DistributionLocation	BeneficiaryID
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	2026-01-25 15:00:00	Kandy Relief Camp 1	1
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	2026-01-27 16:00:00	Community Center Bandarawela	3
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	2026-01-27 17:00:00	Galle Town Hall	4
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	2026-01-27 23:52:00	Kandy Relief Camp 1	3

Console Check all With selected: Edit Copy Delete Export

Table: DistributionItem

The screenshot shows the phpMyAdmin interface for the 'distributionitem' table in the 'disaster_mgt_db' database. The SQL query editor contains the query: `SELECT * FROM distributionitem;`. The table structure panel on the right lists the columns: DistributionItemID, QuantityDistributed, ItemID, and DistributionID. Below the query editor, the 'Show rows' section displays the first three rows of the table. The 'Extra options' section is also visible.

	DistributionItemID	QuantityDistributed	ItemID	DistributionID
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	5	3	3
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	10	4	5
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	2	3	3

Table: User

The screenshot shows the phpMyAdmin interface for the 'user' table in the 'disaster_mgt_db' database. The SQL query editor contains the query: `SELECT * FROM user;`. The table structure panel on the right lists the columns: UserID, UserName, Password, and Role. Below the query editor, the 'Show rows' section displays the first three rows of the table. A warning message states: 'Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.' The 'Extra options' section is also visible.

	UserID	UserName	Password	Role
	6	admin	admin123	Admin
	7	staff1	staff123	Staff
	8	staff2	staff456	Staff

7. Sample CRUD Operations

The following screenshots demonstrate the functional implementation of Create, Read, and Delete operations using SQL.

Database Creation

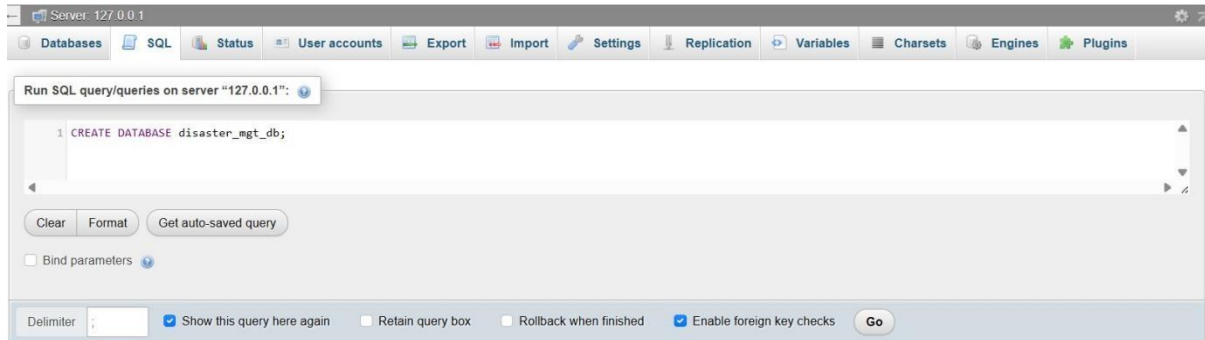


Table Creation

Table: Donor

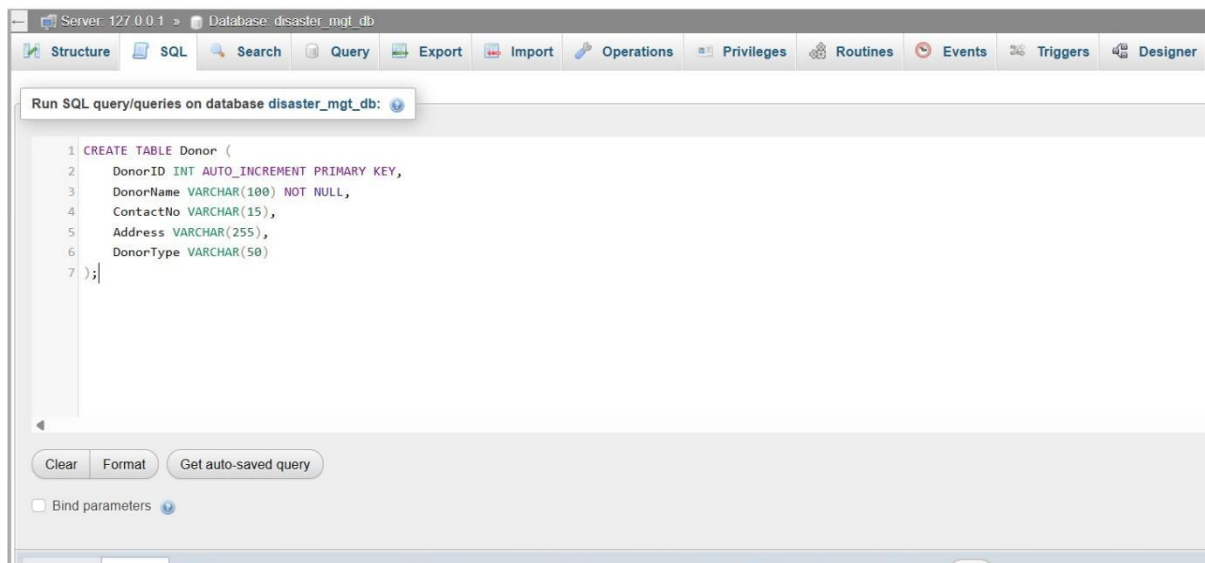
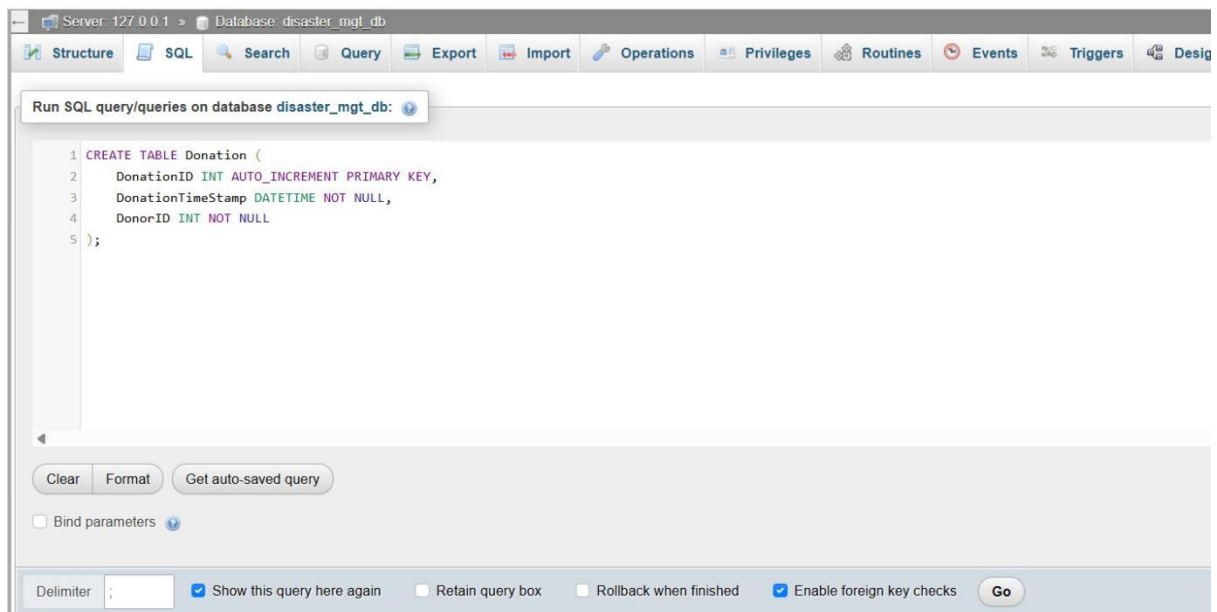


Table: Donation



INSERT Operation (Create)

SQL Query used to insert a new donation record into the system.

Table: Donor

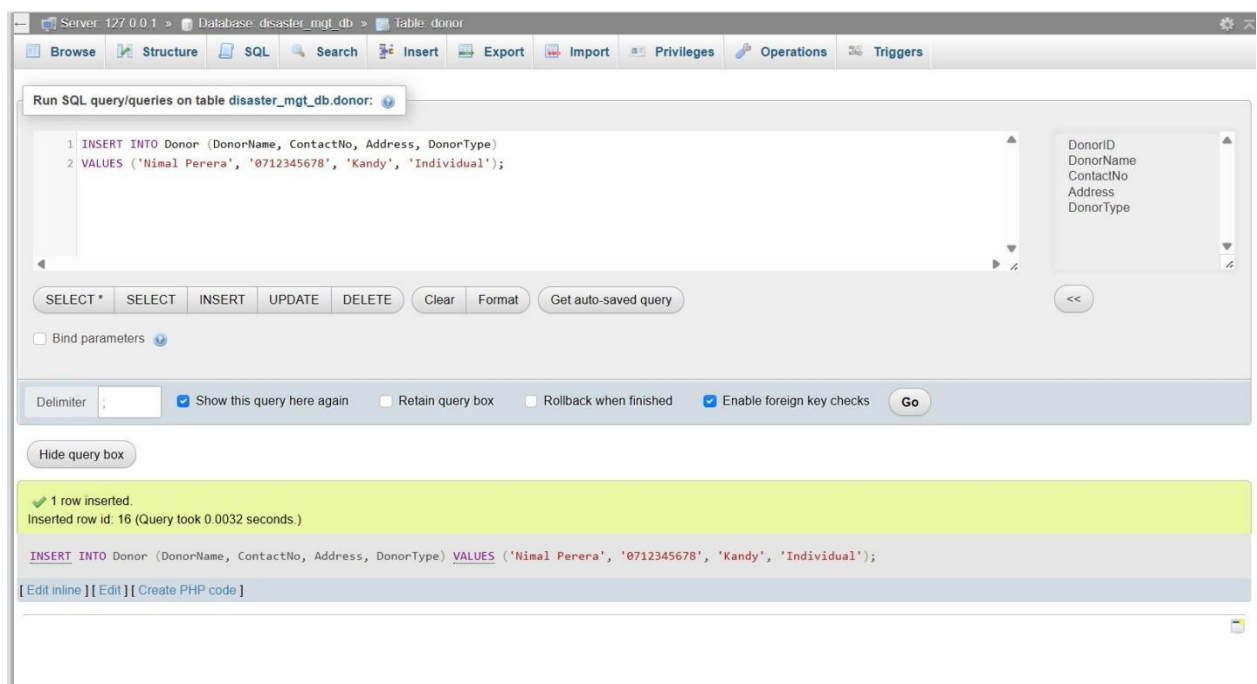
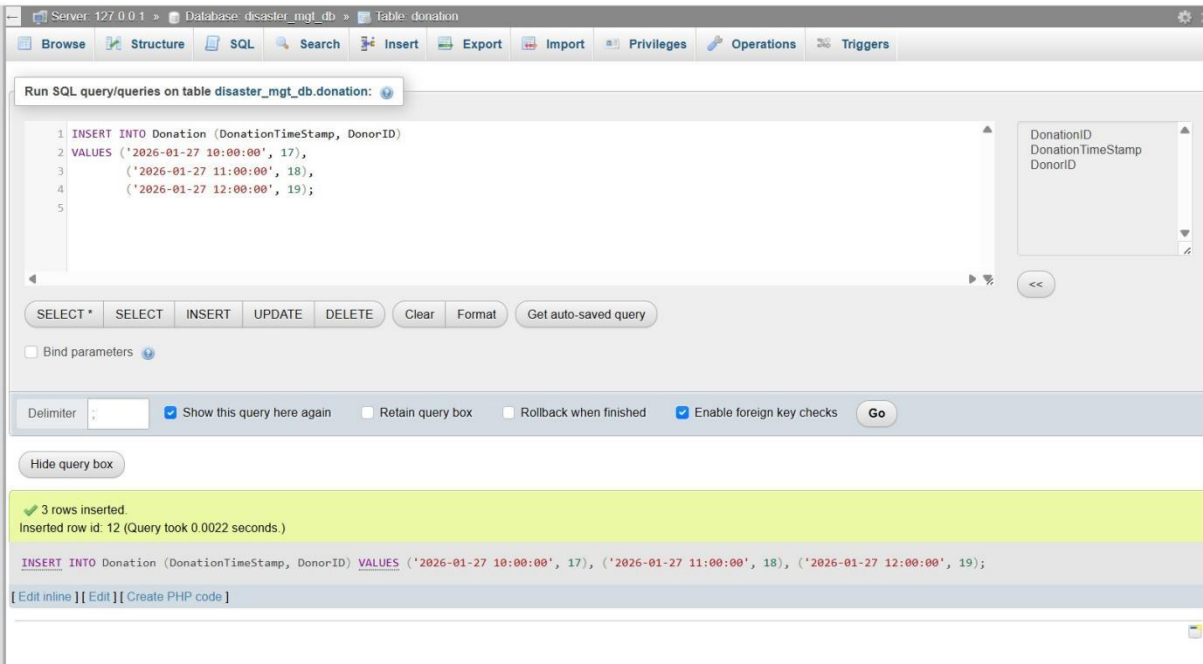


Table: Donation



SELECT Operation (Read)

A complex query used to view specific data

Table: Donor

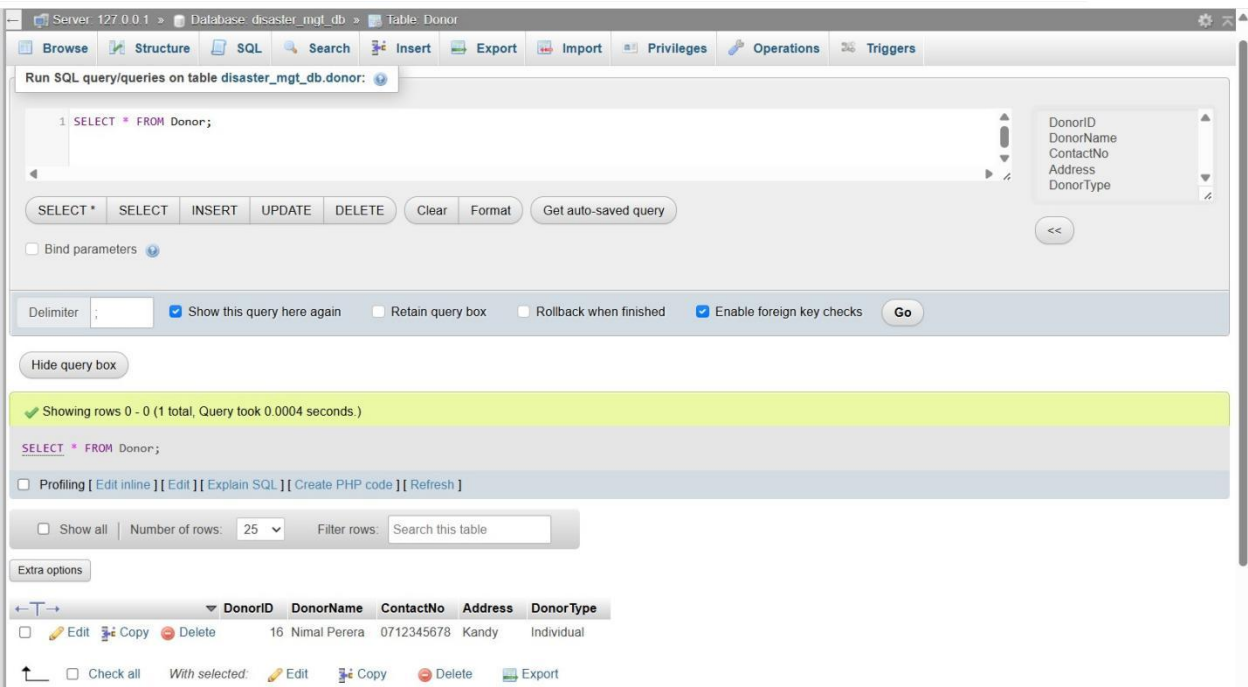


Table: Donation

The screenshot shows a database management interface for a table named 'donation' in the 'disaster_mgt_db' database. The table structure is defined with columns: DonationID, DonationTimeStamp, and DonorID. A SQL query 'SELECT * FROM donation;' has been executed, resulting in 3 rows of data. The interface includes a query editor, a results pane showing the query execution status and results, and a table view of the data.

DonationID	DonationTimeStamp	DonorID
10	2026-01-27 10:00:00	17
11	2026-01-27 11:00:00	18
12	2026-01-27 12:00:00	19

DELETE Operation (Delete)

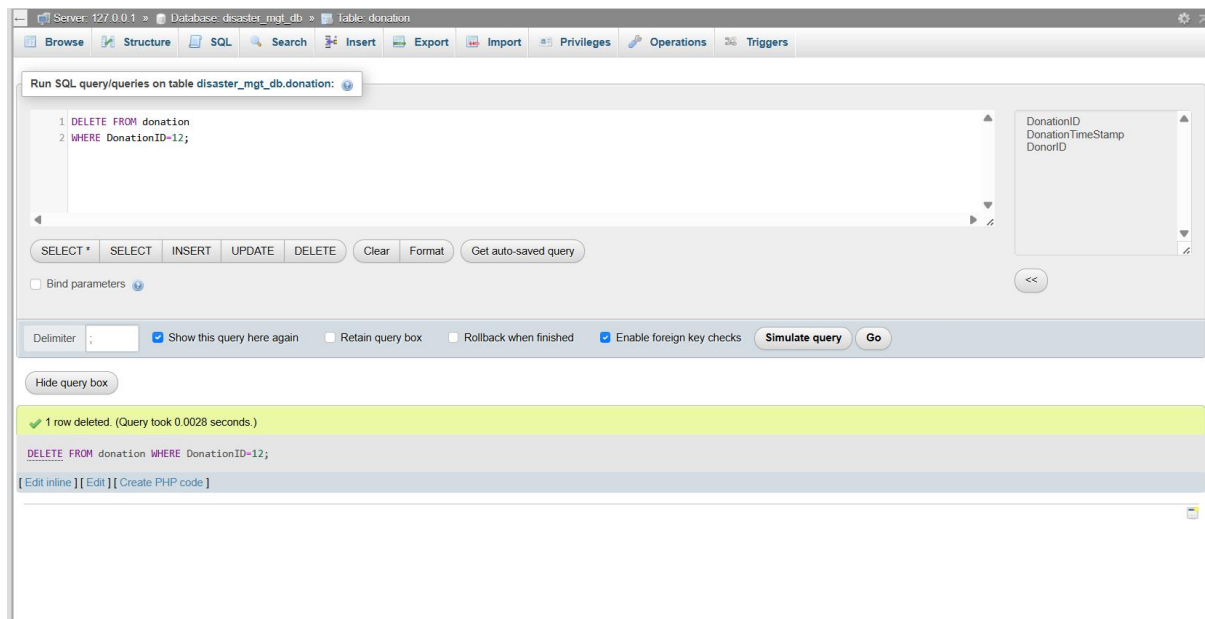
SQL Query demonstrating the removal of a record.

Table: Donor

The screenshot shows a database management interface for a table named 'donor' in the 'disaster_mgt_db' database. The table structure is defined with columns: DonorID, DonorName, ContactNo, Address, and DonorType. A SQL query 'DELETE FROM Donor WHERE DonorName = 'Nimal Perera';' has been executed, resulting in 1 row being deleted. The interface includes a query editor, a results pane showing the query execution status and results, and a table view of the data.

DonorID	DonorName	ContactNo	Address	DonorType
---------	-----------	-----------	---------	-----------

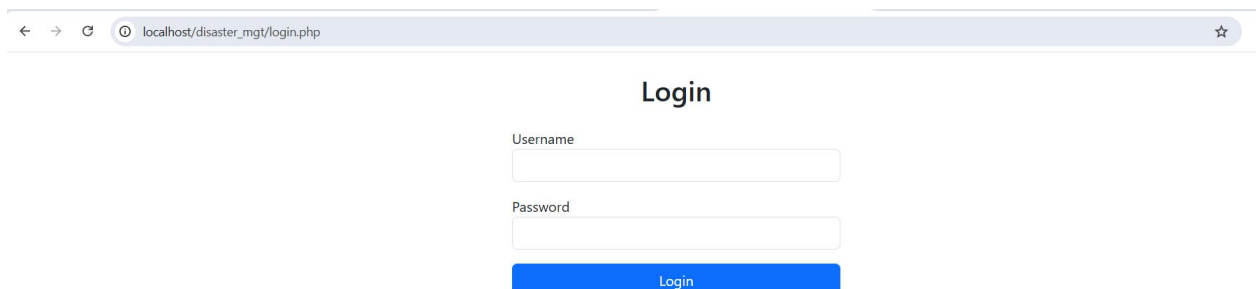
Table: Donation



8. User Interface Implementation

The following screenshots illustrate the developed frontend application, which provides a user-friendly interface for the database.

8.1 Login Interface



8.2 Dashboard

localhost/disaster_mgt/index.php

Disaster Relief and Donation Management System

Welcome, admin
Logout

Donor Management

Add Donor

View Donors

Donation Management

Add Donation

View Donations

Donation Items

Add Donation Item

View Donation Items

Inventory Management

Add Inventory

View Inventory

Distribution Management

Add Distribution

View Distributions

Distribution Items

Add Distribution Item

View Distribution Items

User Management

Add User

View Users

Beneficiary Management

Add Beneficiary

View Beneficiary

Item Management

Add Item

View Item

8.3 Donation Entry Form

localhost/disaster_mgt/add_donation.php

Add Donation

Donation Timestamp
mm/dd/yyyy --:-- --

Donor
-- Select Donor --

Add Donation Back to Dashboard

8.4 Inventory View

localhost/disaster_mgt/view_inventory.php

Inventory

Item	Stock	Last Updated
Water Bottle	100	2026-01-27 14:00:00
Rice	50	2026-01-27 14:30:00
Blanket	30	2026-01-27 15:00:00

Back to Dashboard

9. Conclusion

The **Disaster Relief and Donation Management System** has been successfully implemented, meeting the requirements of Phase 1, 2 and 3. By converting the conceptual ER model into a relational schema and implementing it within a DBMS, we have created a robust solution for tracking relief efforts. The system effectively handles data integrity through Foreign Key constraints, preventing orphaned records in critical areas like Donations and Distributions. The User Interface further enhances usability, allowing non-technical staff to perform CRUD operations without interacting directly with SQL code. This project demonstrates a complete lifecycle of database development, from problem definition to physical deployment.

10. Individual Contributions

The following table details the specific responsibilities and contributions of each group member towards the completion of the project.

Student ID	Name	Contribution	Signature
AS20240395	K I R Abeysekara	<ul style="list-style-type: none">Proposed the project topic and prepared the project proposalConverted ER diagram into relational schemaDesigned and implemented the complete database using phpMyAdminImplemented PHP-based UIProvided necessary screenshots for the report.Compiled and finalized the complete project report.	
AS20240427	B P R Sachintha	<ul style="list-style-type: none">Finalized the Project Proposal.Designed the complete Entity-Relationship (ER) Diagram.Identified entities, attributes, relationships, and cardinalities.Conducted user interface research and provided feedback.Assisted in preparing and reviewing SQL queriesPerformed System Testing and Data Validation.	
AS20240404	A H M D Silva	<ul style="list-style-type: none">Assisted in analyzing system requirements and project scopeAssisted in converting the ER diagram into relational tablesReviewed primary key and foreign key mappingsPerformed System Testing and Data Validation.Assisted in reviewing the final report.	
AS20240577	N G S Sathsara	<ul style="list-style-type: none">Reviewed screenshots and formatting consistencyAssisted in database testing	

AS20240546	T H Sandaruwan	<ul style="list-style-type: none"> Assisted in structuring the project report Assisted in database testing 	
AS20240559	E K S Govinda	<ul style="list-style-type: none"> Performed System Testing and Data Validation. Assisted in final documentation review 	
AS20240538	S M Wijesekara	<ul style="list-style-type: none"> Adding table of contents and formatting for the final report. 	

Group Members

AS20240395-K I R Abeysekara

AS20240427- B P R Sachintha

AS20240404- A H M D Silva

AS20240577- N G S Sathsara

AS20240546- T H Sandaruwan

AS20240559- E K S Govinda

AS20240538- S M Wijesekara