Established – 1961 Subject: OSDBMS

SEVA SADAN'S

R. K. TALREJA COLLEGE

OF

ARTS, SCIENCE & COMMERCE

ULHASNAGAR – 421 003

CERTIFICATE

This is to certify that Ms. Ayesha Vanjre of S.Y. Information Technology (SYIT) Roll No. 2542050 has satisfactorily completed the Open Source DataBase Management System Mini Project entitled MYSQL Backup & Restore Simulation during the academic year 2025 – 2026, as a part of the practical requirement. The project work is found to be satisfactory and is approved for submission.

PROF. INCHARGE **HEAD OF DEPT**

INDEX

1. INTRODUCTION

In this project, I have worked on MySQL Backup and Restore Simulation, which is a practical way to show how data can be protected and recovered when it is lost. The project begins with the design of a database called AmulDB, which contains five tables: Brand, Product, Customer, Orders, and OrderDetails. Each table represents an entity and has attributes such as brand_name, product_name, customer_name, order_date, and quantity. To organize the data properly and avoid duplication, the database design follows normalization principles, especially Third Normal Form, where Brand details are stored separately and linked to Products, Customers are linked to Orders, and OrderDetails connects Orders with Products. In terms of SQL concepts, the project uses DDL commands like CREATE DATABASE and CREATE TABLE to define the structure, DML commands like INSERT INTO to add records, and constraints such as PRIMARY KEY and FOREIGN KEY to maintain relationships and data integrity. The AUTO_INCREMENT feature is used to generate unique IDs automatically. Once the database is ready, a backup is created using the mysqldump command, which generates a .sql file containing all the SQL statements needed to recreate the database. To simulate a real-world scenario of data loss, the database is deleted using DROP DATABASE, and then a restore is performed using the mysql < amul_backup.sql command, which reloads the database from the backup file. This simulation clearly shows that even if data is deleted or lost, it can be recovered safely, proving that the system is reliable, well-designed, and secure.

2. PROBLEM DEFINITION

In today's digital world, databases store large amounts of important information for businesses and organizations. However, data can be lost due to system crashes, hardware failures, accidental deletion, or corruption. This creates a serious problem because without proper recovery methods, valuable information may be permanently lost. To solve this issue, there must be a reliable way to protect data and bring it back when needed. MySQL provides backup and restore features that allow us to create a copy of the database and reload it later. In this project, the problem is defined as how to design a database, store records, and then demonstrate the process of backup and restore through simulation. By intentionally deleting the database and then restoring it from a backup file, we can show that data recovery is possible, ensuring reliability, security, and continuity of the system.

3. OBJECTIVES OF THE PROJECT

The main objective of this project is to demonstrate how data can be protected and recovered in MySQL using backup and restore techniques. The project aims to design a well-structured database called AmulDB with multiple tables such as Brand, Product, Customer, Orders, and OrderDetails, following normalization principles to avoid duplication and maintain clean relationships. Another objective is to apply important SQL concepts, including the use of DDL commands for creating databases and tables, DML commands for inserting records, and constraints like primary keys and foreign keys to ensure data integrity. The project also focuses on simulating a real-world scenario of data loss by intentionally deleting the database and then restoring it from a backup file created using the mysqldump command. By doing this, the project highlights the importance of backup files, shows how restoration works, and proves that even if data is lost, it can be recovered safely. Overall, the objective is to provide a practical demonstration of database reliability, security, and recovery using MySQL.

4. SCOPE OF THE PROJECT

- To design a structured database called Amuldb with multiple tables such as Brand, Product, Customer, Orders, and OrderDetails.
- To apply normalization principles (especially Third Normal Form) for organizing data and avoiding duplication.
- To use SQL concepts like DDL commands (CREATE DATABASE, CREATE TABLE), DML commands (INSERT INTO), and constraints (PRIMARY KEY, FOREIGN KEY) for building and managing the database.
- To demonstrate the process of taking a backup of the database using the mysqldump command.
- To simulate a real-world scenario of data loss by deleting the database with the DROP DATABASE command.
- To restore the database from the backup file using the mysql < backup.sql command.
- To verify that all tables and records are successfully recovered after restoration.
- To highlight the importance of backup and restore techniques in ensuring data safety, reliability, and continuity.

5. REQUIREMENT SPECIFICATION

a. Hardware Requirements

• Computer or Laptop
• Minimum 4 GB RAM
• At least 10 GB free storage

b. Software Requirements

Software	**Purpose**
MYSQL Server	Database creation and management
MYSQL Workbench	Query execution interface
Window OS	Platform
SQL	Query language

6. SYSTEM DESIGN

1.Database Design

- Database Name: *AmulDB*
- Tables: Brand, Product, Customer, Orders, OrderDetails
- Attributes: brand_name, product_name, customer_name, order_date, quantity, etc.
- Relationships:
- Brand → Product (one-to-many)
- Customer → Orders (one-to-many)
- Orders → OrderDetails (one-to-many)
- Product → OrderDetails (many-to-many via OrderDetails table)

2.Normalization

- Database follows Third Normal Form (3NF).
- Data is split into multiple tables to avoid duplication.
- Foreign keys maintain relationships between tables.

3.SQL Concepts Used

- DDL: CREATE DATABASE, CREATE TABLE
- DML: INSERT INTO, SELECT
- Constraints: PRIMARY KEY, FOREIGN KEY
- AUTO_INCREMENT for unique IDs

4.Backup & Restore Process

- Backup: mysqldump -u root -p AmulDB > amul_backup.sql
- Restore: mysql -u root -p AmulDB < amul_backup.sql
- Verification: SHOW DATABASES, SHOW TABLES, SELECT queries

5.Simulation Flow

- Step 1: Create database and insert records
- Step 2: Take backup file
- Step 3: Delete database (simulate data loss)
- Step 4: Restore database from backup file
- Step 5: Verify restored data

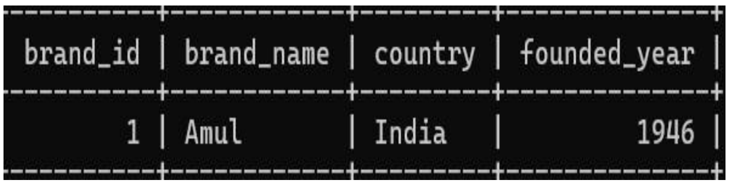
7. DATABASE DESIGN

7.1 Entity Description

- Brand: Stores details of the company brand (e.g., Amul).
- Product: Stores product details such as product name, price, and brand reference.
- Customer: Stores customer information like name, contact, and address.
- Orders: Stores order details such as order ID, customer ID, and order date.
- OrderDetails: Stores specific items in each order, linking products with orders.

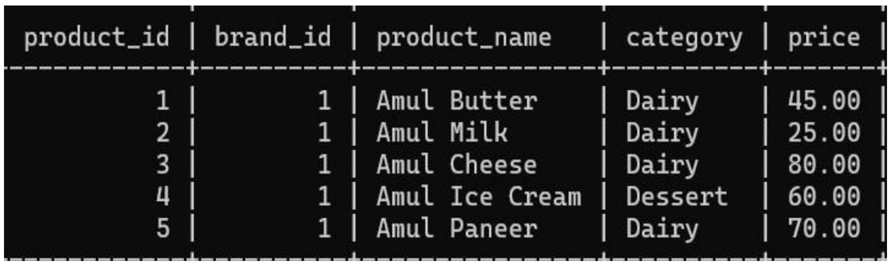
7.2 **Table Structure**

1. Brand Table



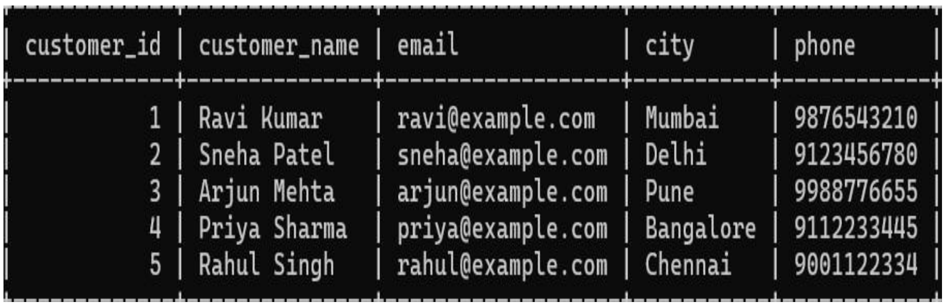
```
+----------+------------+---------+--------------+
| brand_id | brand_name | country | founded_year |
+----------+------------+---------+--------------+
|        1 | Amul       | India   |         1946 |
+----------+------------+---------+--------------+
```

2. Product Table



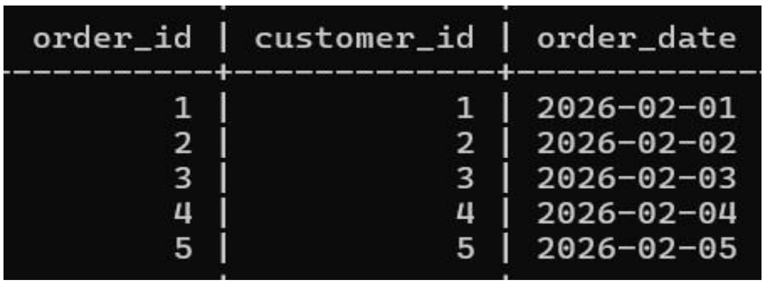
```
| product_id | brand_id | product_name   | category | price |
+------------+----------+----------------+----------+-------+
|          1 |        1 | Amul Butter    | Dairy    | 45.00 |
|          2 |        1 | Amul Milk      | Dairy    | 25.00 |
|          3 |        1 | Amul Cheese    | Dairy    | 80.00 |
|          4 |        1 | Amul Ice Cream | Dessert  | 60.00 |
|          5 |        1 | Amul Paneer    | Dairy    | 70.00 |
+------------+----------+----------------+----------+-------+
```

3. Customer Table



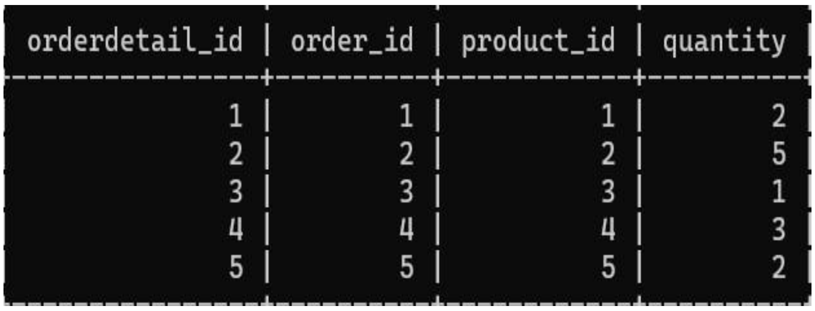
```
+-------------+---------------+-------------------+-----------+------------+
| customer_id | customer_name | email             | city      | phone      |
+-------------+---------------+-------------------+-----------+------------+
|           1 | Ravi Kumar    | ravi@example.com  | Mumbai    | 9876543210 |
|           2 | Sneha Patel   | sneha@example.com | Delhi     | 9123456780 |
|           3 | Arjun Mehta   | arjun@example.com | Pune      | 9988776655 |
|           4 | Priya Sharma  | priya@example.com | Bangalore | 9112233445 |
|           5 | Rahul Singh   | rahul@example.com | Chennai   | 9001122334 |
+-------------+---------------+-------------------+-----------+------------+
```

4. Orders Table



```
 order_id | customer_id | order_date
----------+-------------+------------
        1 |           1 | 2026-02-01
        2 |           2 | 2026-02-02
        3 |           3 | 2026-02-03
        4 |           4 | 2026-02-04
        5 |           5 | 2026-02-05
```

5. OrderDetails Table



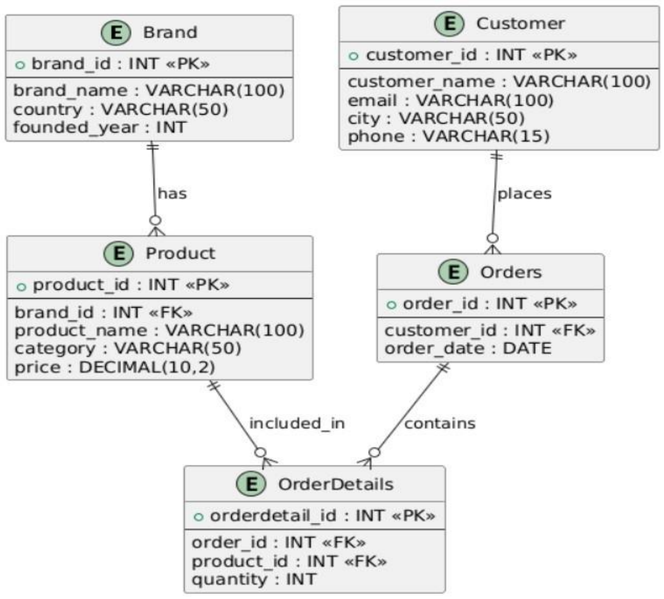
```
 orderdetail_id | order_id | product_id | quantity
----------------+----------+------------+----------
              1 |        1 |          1 |        2
              2 |        2 |          2 |        5
              3 |        3 |          3 |        1
              4 |        4 |          4 |        3
              5 |        5 |          5 |        2
```

7.3 Constraints Used

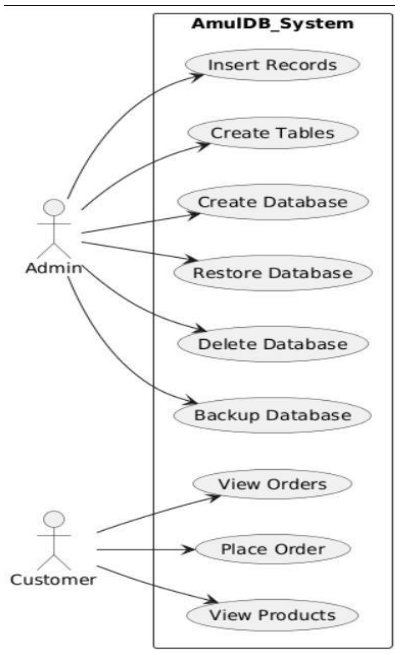
- Primary Key (PK): Ensures each record is unique (brand_id, product_id, customer_id, order_id, orderdetail_id).
- Foreign Key (FK): Maintains relationships between tables (e.g., Product linked to Brand, Orders linked to Customer).
- Auto Increment: Automatically generates unique IDs for primary keys.
- Not Null: Ensures essential fields like product_name, customer_name, and order_date cannot be empty.

8. UML DIAGRAMS

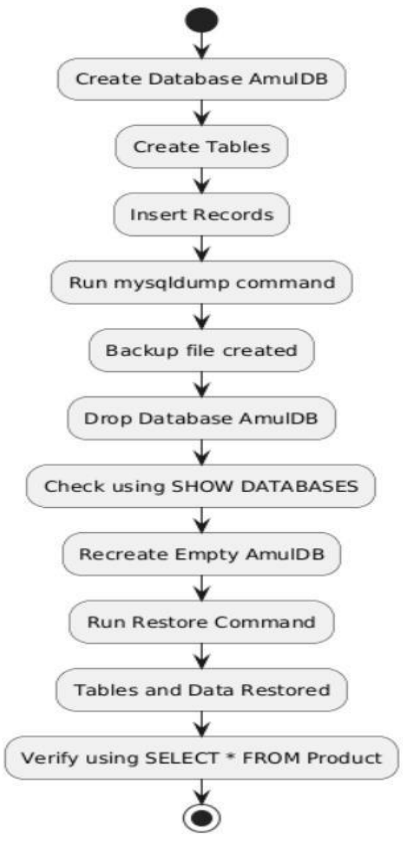
a. ER Diagram



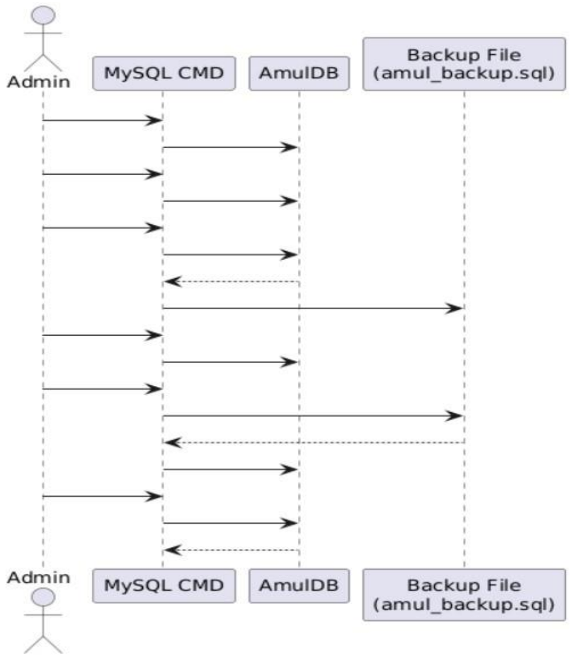
b. Use Case Activity



c. Activity Diagram



d. Sequence Diagram



9. SQL Implementation

9.1 Database Creation

```
CREATE DATABASE AmulDB;
USE AmulDB;
```

9.2 Table Creation

```
CREATE TABLE Brand (     brand_id INT
AUTO_INCREMENT PRIMARY KEY,     brand_name
VARCHAR(100) NOT NULL,
   country VARCHAR(50),
   founded_year INT
);

CREATE TABLE Product (
   product_id INT AUTO_INCREMENT PRIMARY KEY,
brand_id INT,
   product_name VARCHAR(100) NOT NULL,
   category VARCHAR(50),
price DECIMAL(10,2),
   FOREIGN KEY (brand_id) REFERENCES Brand(brand_id)
);

CREATE TABLE Customer (
   customer_id INT AUTO_INCREMENT PRIMARY KEY,
   customer_name VARCHAR(100),
email VARCHAR(100),     city
VARCHAR(50),
   phone VARCHAR(15)
);

CREATE TABLE Orders (
   order_id INT AUTO_INCREMENT PRIMARY KEY,
   customer_id INT,
order_date DATE,
   FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE OrderDetails (
   orderdetail_id INT AUTO_INCREMENT PRIMARY KEY,
order_id INT,     product_id INT,
   quantity INT,
   FOREIGN KEY (order_id) REFERENCES Orders(order_id),
   FOREIGN KEY (product_id) REFERENCES Product(product_id)
);
```

9.3 Data Insertion

```
INSERT INTO Brand (brand_name, country, founded_year) VALUES
('Amul', 'India', 1946);

INSERT INTO Product (brand_id, product_name, category, price) VALUES
(1, 'Amul Butter', 'Dairy', 45.00),
(1, 'Amul Milk', 'Dairy', 25.00),
(1, 'Amul Cheese', 'Dairy', 80.00),
(1, 'Amul Ice Cream', 'Dessert', 60.00),
(1, 'Amul Paneer', 'Dairy', 70.00);

INSERT INTO Customer (customer_name, email, city, phone) VALUES
('Ravi Kumar', 'ravi@example.com', 'Mumbai', '9876543210'),
('Sneha Patel', 'sneha@example.com', 'Delhi', '9123456780'),
('Arjun Mehta', 'arjun@example.com', 'Pune', '9988776655'),
('Priya Sharma', 'priya@example.com', 'Bangalore', '9112233445'),
('Rahul Singh', 'rahul@example.com', 'Chennai', '9001122334');

INSERT INTO Orders (customer_id, order_date) VALUES
(1, '2026-02-01'),
(2, '2026-02-02'),
(3, '2026-02-03'),
(4, '2026-02-04'),
(5, '2026-02-05');

INSERT INTO OrderDetails (order_id, product_id, quantity) VALUES
(1, 1, 2),   -- Ravi bought 2 Amul Butter
(2, 2, 5),   -- Sneha bought 5 Amul Milk
(3, 3, 1),   -- Arjun bought Amul Cheese
(4, 4, 3),   -- Priya bought 3 Amul Ice Cream
(5, 5, 2);   -- Rahul bought 2 Amul Paneer
```

9.4 Data Backup

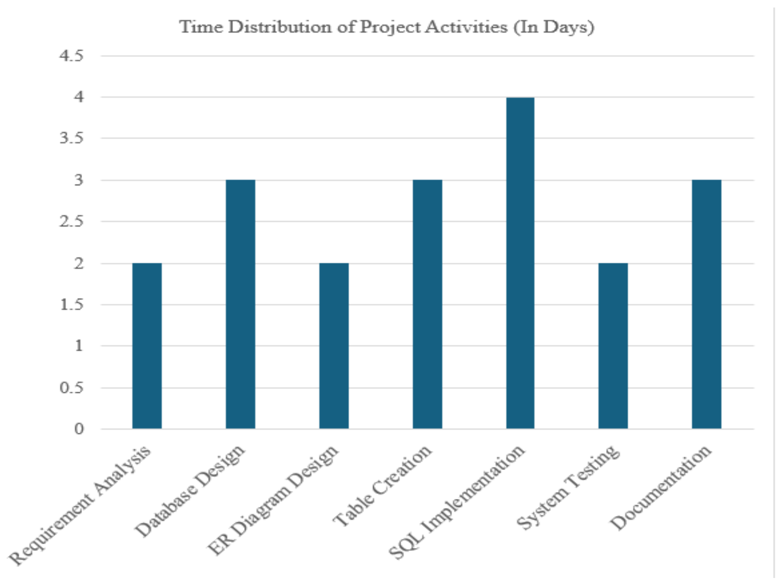
```
mysqldump -u root -p AmulDB > amul_backup.sql
```

9.5 Data Restore

```
mysql -u root -p AmulDB < amul_backup.sql
```

Time Distribution & Work Allotment

The project was divided into structured phases to ensure proper planning and execution. Each activity such as requirement analysis, database design, table creation, SQL implementation, testing, and documentation was allocated time based on its complexity and importance.
The graph below shows the number of days spent on each phase, highlighting that the SQL implementation required the most time among all activities.



10. SYSTEM TESTING AND RESULT

Testing Steps:

1. Database Creation Test ○ Created *AmulDB* database and verified tables (Brand, Product, Customer, Orders, OrderDetails).
   - Result: Tables created successfully with proper attributes and constraints.
2. Data Insertion Test ○ Inserted sample records into each table.
   - Result: Records stored correctly and relationships maintained.
3. Backup Test ○ Executed mysqldump -u root -p AmulDB > amul_backup.sql.
   - Result: Backup file generated successfully containing all SQL statements.
4. Data Loss Simulation Test
   - Executed DROP DATABASE AmulDB. ○ Result: Database deleted successfully, verified using SHOW DATABASES.
5. Restore Test ○ Executed mysql -u root -p AmulDB < amul_backup.sql. ○ Result: Database restored successfully from backup file.
6. Verification Test ○ Checked tables and records using SHOW TABLES and SELECT queries.
   - Result: All records and relationships were recovered correctly.

11.SECURITY, BACKUP AND RECOVERY

Security Measures

- Use of Primary Key and Foreign Key constraints to maintain data integrity.
- NOT NULL constraints ensure essential fields cannot be left empty.
- User authentication in MySQL (username and password) prevents unauthorized access.
- Backup files should be stored in a secure location to avoid misuse.

Backup Process

- Database backup taken using the mysqldump command.
- Backup file (.sql) contains all SQL statements to recreate the database.
- Backup ensures that a copy of the database is available in case of accidental deletion or corruption.

Recovery Process

- Simulated data loss by deleting the database using DROP DATABASE.
- Database restored using the command mysql < amul_backup.sql.
- Verification done by checking tables and records with SHOW TABLES and SELECT queries.
- Recovery ensures that all data and relationships are restored successfully.

12.FUTURE SCOPE AND CONCLUSION

Future Scope:

- **Automation of Backup:** Implement scheduled automatic backups using scripts or MySQL tools.
- **Cloud Integration:** Store backup files securely on cloud platforms (Google Drive, AWS, Azure) for remote access and disaster recovery.
- **Incremental Backup:** Instead of full backups every time, use incremental backups to save only the changes, reducing storage and time.
- **Data Security Enhancements:** Encrypt backup files to prevent unauthorized access.
- **Cross-Platform Recovery:** Demonstrate restoring the database on different operating systems or servers.
- **User Interface Development:** Create a simple GUI tool for backup and restore operations, making it easier for non-technical users.
- **Scalability Testing:** Extend the simulation to larger datasets and multiple databases to test performance.

Conclusion

The MySQL Backup and Restore Simulation project successfully demonstrates how data can be protected and recovered in case of accidental loss or deletion. By designing a structured database, applying normalization principles, and using SQL commands with constraints, the project ensures data integrity and reliability. The backup process using mysqldump and the restore process using the mysql command clearly show that even if data is deleted, it can be recovered safely. System testing confirms that all tables and records are restored correctly, proving the effectiveness of backup and recovery techniques. Overall, the project highlights the importance of database security, reliability, and continuity, and provides a practical foundation for future enhancements such as automation, cloud storage, and advanced security measures.

13.REFERENCES

1. MySQL Official Documentation, Oracle Corporation.
2. Prescribed Database Management System (DBMS) textbooks.
3. Online learning resources related to SQL and MySQL.

14. GLOSSARY

DBMS – Database Management System Software used to store, manage, and retrieve data efficiently in the form of databases.

SQL – Structured Query Language A standard language used to create, insert, update, delete, and retrieve data from a database

Normalization – Process of organizing data

Primary Key – Unique identifier A unique identifier for each record in a table that does not allow duplicate or NULL values.

Foreign Key – Relationship key A field in a table that creates a relationship with the primary key of another table to maintain data consistency.

MySQL- An open-source relational database management system used to store and manage structured data.

15. APPENDIX / SQL CODE

```
CREATE DATABASE AmulDB;
USE AmulDB;

CREATE TABLE Brand (     brand_id INT
AUTO_INCREMENT PRIMARY KEY,     brand_name
VARCHAR(100) NOT NULL,
   country VARCHAR(50),
   founded_year INT
);

CREATE TABLE Product (
   product_id INT AUTO_INCREMENT PRIMARY KEY,
brand_id INT,
   product_name VARCHAR(100) NOT NULL,
   category VARCHAR(50),
price DECIMAL(10,2),
   FOREIGN KEY (brand_id) REFERENCES Brand(brand_id)
);

CREATE TABLE Customer (
   customer_id INT AUTO_INCREMENT PRIMARY KEY,
   customer_name VARCHAR(100),
email VARCHAR(100),     city
VARCHAR(50),
   phone VARCHAR(15)
);

CREATE TABLE Orders (
   order_id INT AUTO_INCREMENT PRIMARY KEY,
   customer_id INT,
order_date DATE,
   FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

CREATE TABLE OrderDetails (
   orderdetail_id INT AUTO_INCREMENT PRIMARY KEY,
order_id INT,     product_id INT,
   quantity INT,
   FOREIGN KEY (order_id) REFERENCES Orders(order_id),
   FOREIGN KEY (product_id) REFERENCES Product(product_id)
);

    INSERT INTO Brand (brand_name, country, founded_year) VALUES
    ('Amul', 'India', 1946);
```

```
INSERT INTO Product (brand_id, product_name, category, price) VALUES
(1, 'Amul Butter', 'Dairy', 45.00),
(1, 'Amul Milk', 'Dairy', 25.00),
(1, 'Amul Cheese', 'Dairy', 80.00),
(1, 'Amul Ice Cream', 'Dessert', 60.00),
(1, 'Amul Paneer', 'Dairy', 70.00);

INSERT INTO Customer (customer_name, email, city, phone) VALUES
('Ravi Kumar', 'ravi@example.com', 'Mumbai', '9876543210'),
('Sneha Patel', 'sneha@example.com', 'Delhi', '9123456780'),
('Arjun Mehta', 'arjun@example.com', 'Pune', '9988776655'),
('Priya Sharma', 'priya@example.com', 'Bangalore', '9112233445'),
('Rahul Singh', 'rahul@example.com', 'Chennai', '9001122334');

INSERT INTO Orders (customer_id, order_date) VALUES
(1, '2026-02-01'),
(2, '2026-02-02'),
(3, '2026-02-03'),
(4, '2026-02-04'),
(5, '2026-02-05');

INSERT INTO OrderDetails (order_id, product_id, quantity) VALUES
(1, 1, 2),   -- Ravi bought 2 Amul Butter
(2, 2, 5),   -- Sneha bought 5 Amul Milk
(3, 3, 1),   -- Arjun bought Amul Cheese
(4, 4, 3),   -- Priya bought 3 Amul Ice Cream
(5, 5, 2);   -- Rahul bought 2 Amul Paneer
```

```
mysqldump -u root -p AmulDB > amul_backup.sql
```

```
mysql -u root -p AmulDB < amul_backp.sql
```