Established – 1961 Subject: OSDBMS

SEVA SADAN'S

R. K. TALREJA COLLEGE

OF

ARTS, SCIENCE & COMMERCE ULHASNAGAR – 421 003

CERTIFICATE

This is to certify that Mr./Ms. BABITA GUPTA P of S.Y. Information Technology (SYIT) Roll No. 2542010 has satisfactorily completed the Open Source DataBase Management System Mini Project entitled CUSTOMER RELATIONSHIP MANAGEMENT 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

Introduction

A database system is an organized collection of data along with the software that allows users to store, retrieve, update, and manage that data efficiently. It provides a structured way to handle large volumes of information while ensuring accuracy, consistency, and security. In modern organizations, database systems form the backbone of information management because almost all operational activities—such as sales, customer service, and marketing—depend on reliable data storage and access. Instead of maintaining scattered files or manual records, a database system centralizes data, reduces redundancy, and supports faster decision-making.

In the context of Customer Relationship Management (CRM), a database system plays a crucial role. CRM focuses on managing an organization's interactions with current and potential customers. It involves collecting and analysing customer data such as personal details, contact information, purchase history, preferences, feedback, and communication records. Without a proper database system, managing this information becomes difficult, time-consuming, and error-prone. Manual methods or isolated spreadsheets often lead to data duplication, inconsistency, and loss of important customer insights. A database system addresses these problem areas by providing a single, integrated platform where all customer-related data can be stored and accessed in an organized manner. This enables businesses to understand customer behaviour better, improve service quality, and build long-term relationships.

The main problem area addressed by a database-driven CRM system is inefficient customer data management. Organizations often deal with thousands or even millions of customers, making it impossible to track interactions accurately without automation. A database system ensures that customer information is updated in real time and is available to different departments such as sales, support, and marketing. This improves coordination within the organization and helps in delivering personalized services. For example, sales teams can view past purchases, support teams can track complaints, and marketing teams can design targeted campaigns based on stored customer data. As a result, customer satisfaction and loyalty increase, which directly impacts business growth.

MySQL is widely used as an open-source solution for building CRM database systems. One of the main reasons for using MySQL is that it is free and open source, making it cost-effective for organizations of all sizes, especially small and medium enterprises. Despite being free, MySQL is powerful, reliable, and capable of handling large amounts of data efficiently. It supports Structured Query Language (SQL), which is easy to learn and widely accepted in the industry. This makes database development and maintenance simpler for students and professionals alike.

2. Problem Definition

In many organizations, Customer Relationship Management (CRM) activities are still handled using manual or unstructured systems such as paper records, registers, or basic spreadsheet files. These methods are not designed to manage large volumes of customer data or frequent updates. As the number of customers and interactions increases, manual systems become inefficient, unreliable, and difficult to maintain. This creates serious challenges in tracking customer information accurately and delivering consistent service.

Existing Manual / Unstructured System

- Customer details are stored in paper files or separate spreadsheet documents
- Data is maintained by different departments independently
- No centralized system for customer information
- Updates are done manually, increasing the chance of errors

Problems in the Existing System

- Data redundancy: Same customer information is recorded multiple times in different files
- Data inconsistency: Conflicting or outdated customer records due to improper updates
- Lack of security: Sensitive customer data can be easily accessed, lost, or misused
- Time-consuming processes: Searching and updating records takes excessive time
- Poor data reliability: High risk of data loss, duplication, or incorrect information

Need for a Database-Driven Solution

- Centralized storage of all customer data in a single system
- Reduction of redundancy and improvement in data consistency
- Enhanced data security through controlled access and authentication
- Faster data retrieval and efficient handling of customer interactions
- Reliable support for CRM operations using a structured database system such as MySQL

3. Objectives of the Project

The main objectives of this project are focused on developing an efficient and reliable Customer Relationship Management (CRM) database system using an open-source database platform.

- To design and develop a structured database system that efficiently stores and organizes customer-related information
- To implement SQL queries for effective data insertion, retrieval, updating, and deletion
- To ensure data integrity and consistency by applying appropriate database constraints such as primary keys, foreign keys, and unique constraints
- To gain practical, hands-on experience in designing and managing databases using MySQL

4. Scope of the Project

The scope of this project defines the boundaries and applicability of the Customer Relationship Management (CRM) database system developed using MySQL. The system is designed to manage customer-related data efficiently in a structured and secure manner. It can be applied in various environments where maintaining accurate customer information and interaction history is essential.

Where the System Can Be Used

- Small and medium-sized business organizations to manage customer details, sales records, and service interactions
- Educational institutions for managing student information, inquiries, admissions, and alumni relationships
- Administrative offices that require systematic handling of client or stakeholder data
- Training labs and academic environments for learning database management concepts

Users of the System

- Administrator (Admin): Responsible for database creation, user management, security control, and maintenance
- Staff / Employees: Enter, update, and retrieve customer or student-related data for daily operations

- Students: Use the system for learning purposes, practicing SQL queries, and understanding CRM concepts
- Business Users / Managers: Access reports and customer data to support decision-making and planning

Subdomains of Application

- Educational Use: Useful for academic projects, laboratory experiments, and teaching database design and SQL concepts
- Business Use: Helps organizations track customer interactions, improve service quality, and support marketing strategies
- Administrative Use: Assists offices in organizing records, reducing paperwork, and improving data accuracy

Academic Limitations

- The project is limited to core CRM functionalities and does not include advanced features such as real-time analytics or artificial intelligence
- It is designed primarily for learning and demonstration purposes, not for large-scale commercial deployment
- Security features are implemented at a basic level suitable for academic use
- Performance testing is limited due to constrained data size and system resources

5.1 Hardware Requirements

The proposed system does not require high-end hardware and can run on commonly available devices.

- Computer / Laptop:
  Standard desktop or laptop system
- Minimum RAM:
  4 GB RAM (recommended for smooth execution)
- Storage:
  Minimum 20 GB free disk space for database files and software installation

5.2 Software Requirements

Software	Purpose
MySQL Server	Database management and storage
MySQL Workbench	Writing and executing SQL queries
Operating System (Windows / Linux)	Platform to run the DBMS
SQL	Query language for database operations

6. SYSTEM DESIGN

6.1 System Architecture (Conceptual)

The system architecture of the Customer Relationship Management (CRM) project is based on a database-driven model, where users interact directly with a relational database through structured queries. This conceptual architecture focuses only on the DBMS layer and does not involve hardware or network components.

User Interaction with the Database

In the proposed CRM system, users such as administrators or authorized staff interact with the database using a database interface tool like MySQL Workbench. The user performs operations by writing SQL queries to insert new customer records, update existing information, retrieve data, or generate reports.
Each user interaction is translated into a structured SQL command, which acts as a formal request to the database system. The interface ensures that users can communicate with the database in a controlled and organized manner, reducing errors compared to manual data handling.

Role of the MySQL Server

The MySQL Server acts as the core component of the system architecture. It is responsible for managing the entire database and ensuring that all operations are executed correctly and securely.

the MySQL Server performs the following key roles:

- Receives SQL queries from users through the interface
- Validates query syntax and semantics
- Enforces database constraints such as primary keys, foreign keys, and uniqueness
- Manages data storage, indexing, and retrieval
- Controls user authentication and access privileges
- Maintains data consistency and integrity during concurrent access

By handling these tasks, the MySQL Server ensures that customer data remains accurate, secure, and consistently available to authorized users.

Query Execution Flow

The query execution process in the CRM system follows a structured sequence of steps to ensure correctness and efficiency:

1. Query Submission:
   The user writes and submits an SQL query (such as INSERT, SELECT, UPDATE, or DELETE) through the database interface.
2. Query Reception:
   The query is received by the MySQL Server, which acts as the central processing unit for database operations.
3. Query Parsing and Validation:
   The server checks the query for correct SQL syntax and verifies table names, attributes, and constraints.
4. Authorization Check:
   The system verifies whether the user has the required permissions to execute the requested operation.
5. Execution Planning:
   The MySQL Server determines the most efficient way to execute the query using available indexes and execution plans.
6. Database Operation:
   The requested operation is performed on the database, such as inserting new data or retrieving existing records.
7. Result Generation:
   The server prepares the output in a structured format, such as a result set or confirmation message.
8. Result Display:
   The output is returned to the user interface and displayed to the user.

7. DATABASE DESIGN

Database design is a crucial stage in the development of the Customer Relationship Management (CRM) system. A well-structured database ensures efficient storage, easy retrieval, and accurate management of customer-related data. The CRM database in this project is implemented using a relational approach with MySQL.

7.1 Entity Description

Customer Table:

The Customer table stores complete information about customers who interact with the organization. It includes personal and contact details such as customer name, email address, and phone number. This table acts as the core entity of the CRM system because all other transactions and interactions are linked to customers. Accurate customer data helps organizations maintain long-term relationships and improve service quality.

Product Table:

The Product table maintains information about the products or services offered by the organization. It stores product names, pricing details, and identifiers. This table helps in tracking what products are associated with customer orders and supports sales and service analysis.

Order Table:

The Order table records transactions made by customers. It contains order-specific details such as order date, total amount, and customer reference. This table establishes a relationship between customers and the products or services they purchase, enabling tracking of customer buying behaviour.

Payment Table:

The Payment table stores payment-related information for each order. It includes details such as payment mode and payment reference. This table ensures proper tracking of financial transactions associated with customer orders and supports accounting and reporting needs.

7.2 Table Structure

Customer Table

Attribute Name	Data Type	Description
Customer_ID	INT	Unique identifier for each customer
Customer_Name	VARCHAR(50)	Name of the customer
Email	VARCHAR(50)	Email address of the customer
Phone	VARCHAR(15)	Contact number

Product Table

Attribute Name	Data Type	Description
Product_ID	INT	Unique identifier for each product
Product_Name	VARCHAR(50)	Name of the product/service
Price	DECIMAL(10,2)	Cost of the product

Order Table

Attribute Name	Data Type	Description
Order_ID	INT	Unique identifier for each order
Customer_ID	INT	Reference to Customer table
Order_Date	DATE	Date of order
Total_Amount	DECIMAL(10,2)	Total order value

Payment Table

Attribute Name	Data Type	Description
Payment_ID	INT	Unique identifier for payment
Order_ID	INT	Reference to Order table
Payment_Mode	VARCHAR(20)	Mode of payment

7.3 Constraints Used

Constraints are rules applied to database tables to maintain data integrity, accuracy, and consistency.

PRIMARY KEY

A PRIMARY KEY uniquely identifies each record in a table. It ensures that no two records have the same identifier and that the value is never NULL. In the CRM system, fields such as Customer_ID, Product_ID, Order_ID, and Payment_ID are defined as primary keys. This allows the database to quickly locate and manage records without ambiguity.

FOREIGN KEY

A FOREIGN KEY is used to establish a relationship between two tables. It ensures referential integrity by allowing only valid data references. For example, Customer_ID in the Order table is a foreign key that refers to the Customer table. This prevents orders from being created for non-existent customers and maintains logical consistency across tables.

NOT NULL

The NOT NULL constraint ensures that a column must contain a value and cannot be left empty. This is applied to essential fields such as customer name and order date. By enforcing NOT NULL, the database avoids incomplete or meaningless records that could affect system reliability.

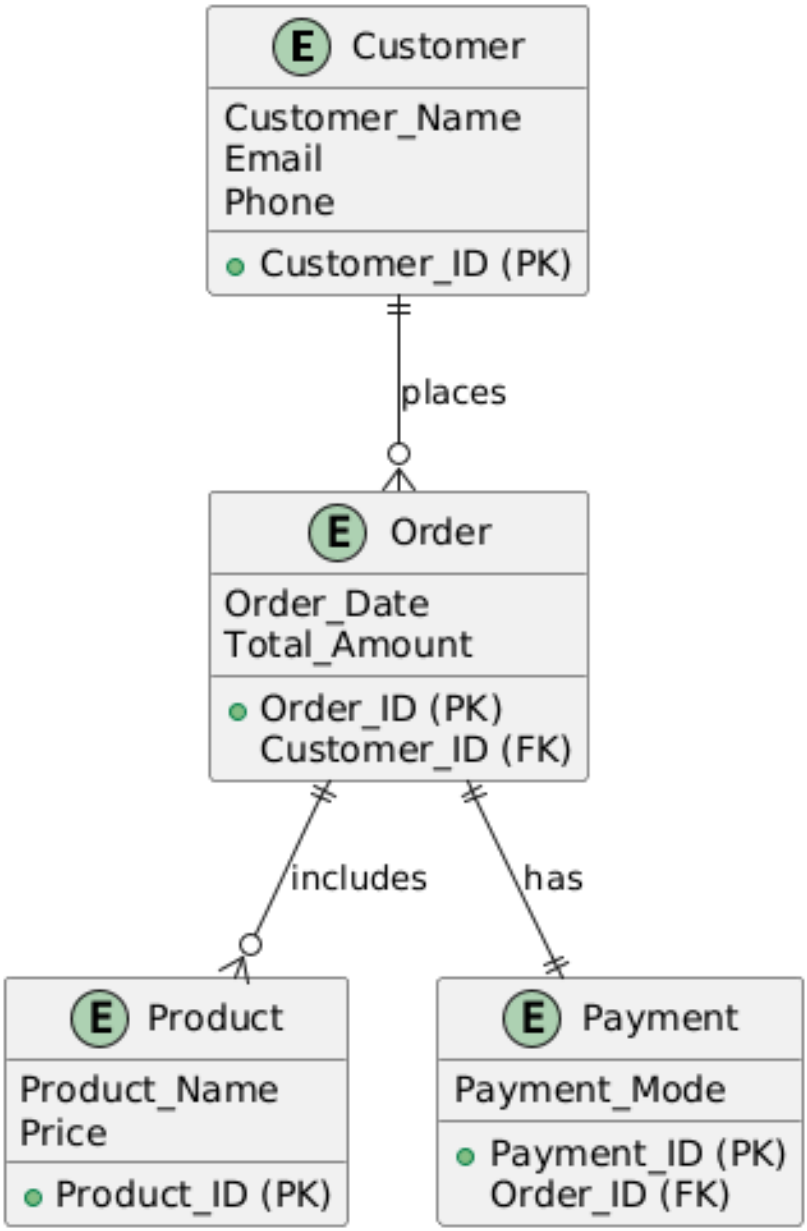
UNIQUE

The UNIQUE constraint ensures that all values in a column are different. It is commonly applied to attributes like email addresses, where duplication is not allowed. In the CRM system, the UNIQUE constraint prevents multiple customers from registering with the same email ID, ensuring accurate identification.

8. UML DIAGRAMS

UML (Unified Modelling Language) diagrams are used to visually represent the structure and behaviour of the Customer Relationship Management (CRM) database system. These diagrams help in understanding how data entities are related and how users interact with the system.

8.1 ER Diagram (Entity Relationship Diagram)



The ER Diagram represents the entities, their attributes, and the relationships among them in the CRM database.

Entities

- Customer
- Product
- Order
- Payment

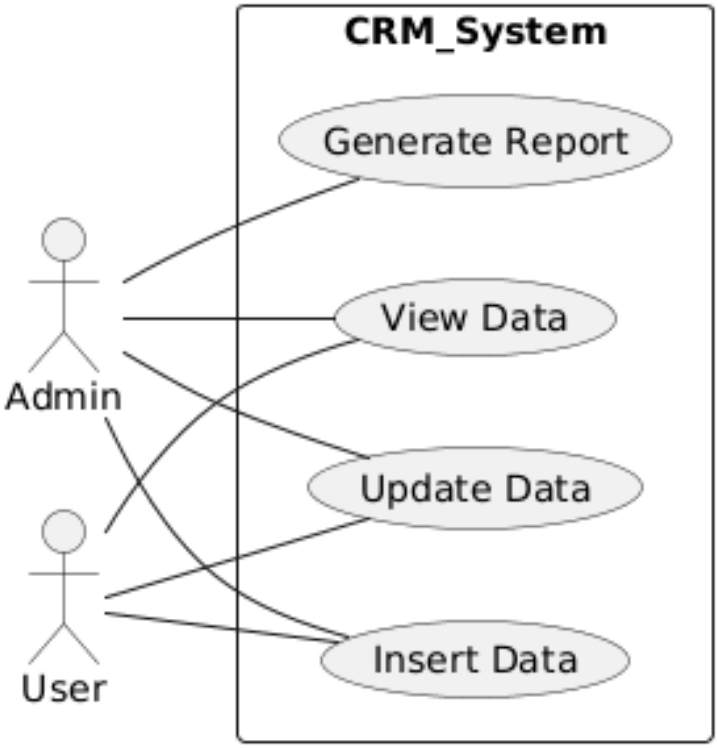
Relationships

- A Customer *places* an Order
  (One customer can place many orders — One-to-Many)
- An Order *includes* Product(s)
  (One order can include multiple products)
- An Order *has* a Payment
  (Each order is associated with one payment)

Explanation

The Customer entity is the central entity of the CRM system. Orders are dependent on customers, and payments are dependent on orders. These relationships ensure proper linkage of customer activities and transactions, maintaining referential integrity within the database.

8.2 Use Case Diagram



The Use Case Diagram shows how different users interact with the CRM database system and what operations they can perform.

Actors

- Admin
- User (Staff)

Use Cases / Operations

- Insert Data
- Update Data
- View Records
- Generate Reports

Actor–Use Case Interaction

- **Admin**
    - Insert customer, product, and order data

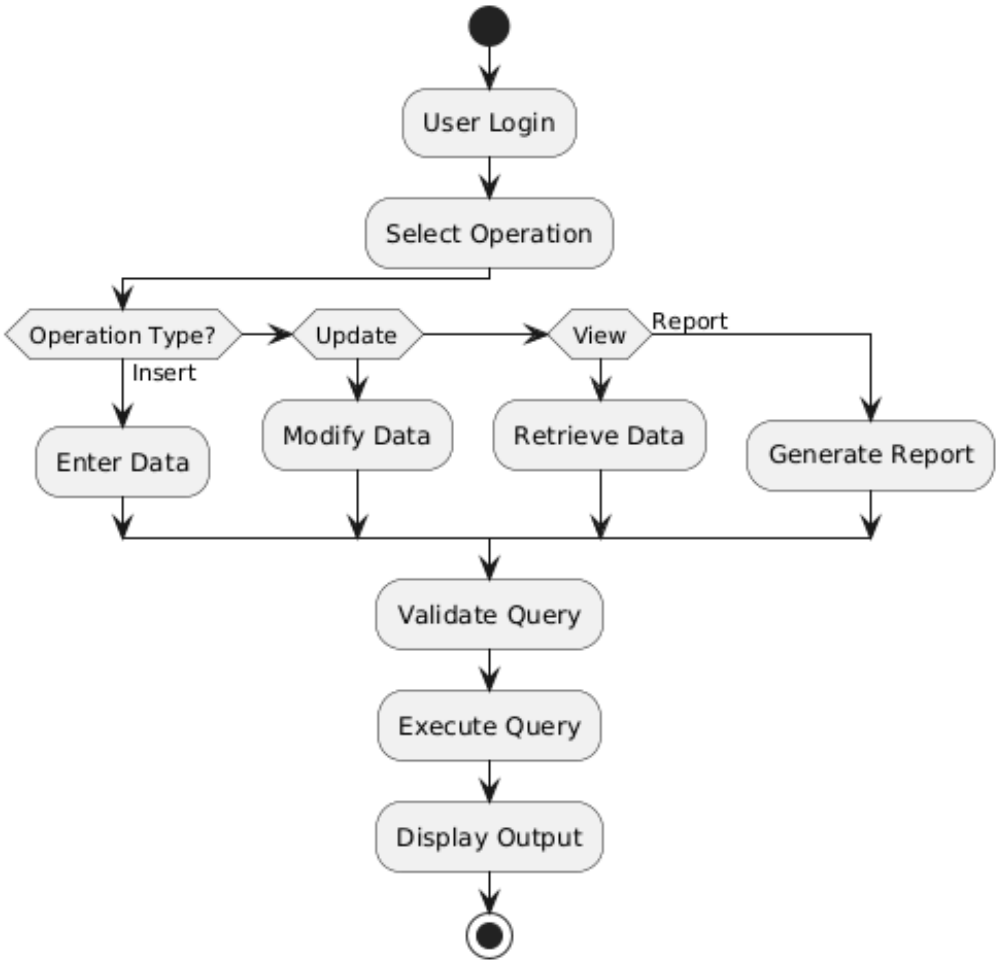
- Update and delete records
- View all data
- Generate reports

- **User (Staff)**
  - Insert new records
  - Update existing records
  - View customer and order information

Explanation

The Admin has full control over the system, while the User has limited access based on assigned privileges. This diagram highlights role-based access within the CRM system.

8.3 Activity Diagram



The **Activity Diagram** represents the **workflow of database operations** in the CRM system.

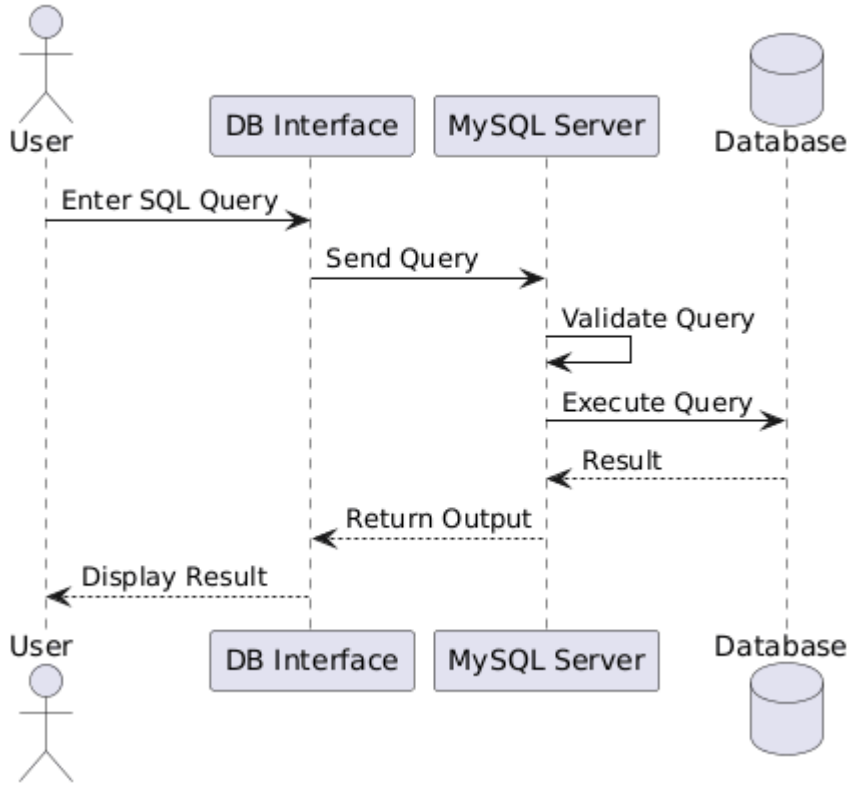
Workflow Steps

1. Start
2. User logs into the system
3. User selects an operation (Insert / Update / View / Report)
4. User enters SQL query
5. System validates query
6. Database executes operation
7. Result is generated
8. Output is displayed to the user
9. End

Explanation

This diagram shows the sequential flow of activities from user interaction to result generation. It helps in understanding how data moves through the system during database operations.

8.4 Sequence Diagram



The Sequence Diagram illustrates the query execution flow and interaction between system components over time.

Participants

- User
- Database Interface (MySQL Workbench)
- MySQL Server
- Database

Sequence Flow

1. User sends SQL query
2. Interface forwards query to MySQL Server
3. MySQL Server validates the query
4. Server accesses the database
5. Database returns result

6. MySQL Server sends output
7. Interface displays result to user

Explanation

This diagram clearly explains how a SQL query is processed step-by-step, from user input to final output, ensuring transparency in database execution.

9. SQL IMPLEMENTATION

This section explains the SQL implementation of the Customer Relationship Management (CRM) database system. SQL (Structured Query Language) is used to create the database, define tables, insert data, retrieve information, and perform advanced queries. The implementation is carried out using MySQL, which supports all standard SQL operations required for a relational database system.

9.1 Database Creation

CREATE DATABASE

The first step in SQL implementation is creating a database to store all CRM-related tables.

```
CREATE DATABASE CRM_DB;
```

After creating the database, it is selected for use:

```
USE CRM_DB;
```

This command ensures that all subsequent SQL operations are performed within the CRM database.

9.2 Table Creation

CREATE TABLE Queries

Tables are created to store structured data related to customers, products, orders, and payments.

Customer Table

```
CREATE TABLE Customer (
   Customer_ID INT PRIMARY KEY,
   Customer_Name VARCHAR(50) NOT NULL,
   Email VARCHAR(50) UNIQUE,
   Phone VARCHAR(15)
);
```

Product Table

```
CREATE TABLE Product (
   Product_ID INT PRIMARY KEY,
   Product_Name VARCHAR(50) NOT NULL,
   Price DECIMAL(10,2) NOT NULL
);
```

Order Table

```
CREATE TABLE Orders (
   Order_ID INT PRIMARY KEY,
   Order_Date DATE NOT NULL,
   Total_Amount DECIMAL(10,2),
   Customer_ID INT,
   FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
);
```

Payment Table

```
CREATE TABLE Payment (
   Payment_ID INT PRIMARY KEY,
   Payment_Mode VARCHAR(20) NOT NULL,
   Order_ID INT,
   FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID)
);
```

These tables are linked using **foreign keys**, which help maintain relationships and data consistency.

9.3 Data Insertion

INSERT Queries

INSERT queries are used to add records into database tables.

Insert into Customer

```
INSERT INTO Customer VALUES
(1, 'Amit Sharma', 'amit@gmail.com', '9876543210'),
(2, 'Neha Verma', 'neha@gmail.com', '9123456789');
```

Insert into Product

```
INSERT INTO Product VALUES
(101, 'Laptop', 55000.00),
(102, 'Printer', 12000.00);
```

Insert into Orders

```
INSERT INTO Orders VALUES
(1001, '2025-02-10', 67000.00, 1),
(1002, '2025-02-12', 12000.00, 2);
```

Insert into Payment

```
INSERT INTO Payment VALUES
(501, 'Online', 1001),
(502, 'Cash', 1002);
```

These queries populate the CRM database with sample data for testing and demonstration.

9.4 Data Retrieval

SELECT, WHERE, JOIN

SELECT Query

Used to retrieve all records from a table.

```
SELECT * FROM Customer;
```

WHERE Clause

Used to filter records based on conditions.

```
SELECT * FROM Customer
WHERE Customer_ID = 1;
```

JOIN Query

Used to retrieve data from multiple related tables.

```
SELECT Customer.Customer_Name, Orders.Order_ID, Orders.Total_Amount
FROM Customer
JOIN Orders
ON Customer.Customer_ID = Orders.Customer_ID;
```

JOIN queries are essential in CRM systems to combine customer, order, and payment information.

9.5 Advanced Queries

GROUP BY

Used to group records and apply aggregate functions.

```
SELECT Customer_ID, SUM(Total_Amount) AS Total_Spent
FROM Orders
GROUP BY Customer_ID;
```

This query calculates the total amount spent by each customer.

HAVING

Used to filter grouped results.

```
SELECT Customer_ID, SUM(Total_Amount) AS Total_Spent
FROM Orders
GROUP BY Customer_ID
HAVING SUM(Total_Amount) > 20000;
```

This query displays only those customers whose total spending exceeds a specified amount.

Subqueries

A query written inside another query.

```
SELECT Customer_Name
FROM Customer
WHERE Customer_ID IN (
   SELECT Customer_ID
   FROM Orders
   WHERE Total_Amount > 20000
);
```

This query retrieves customers who have placed high-value orders.

Views

A **view** is a virtual table created from a SELECT query.

```
CREATE VIEW Customer_Order_View AS
SELECT Customer.Customer_Name, Orders.Order_ID, Orders.Total_Amount
FROM Customer
JOIN Orders
ON Customer.Customer_ID = Orders.Customer_ID;
```

To retrieve data from the view:

```
SELECT * FROM Customer_Order_View;
```

Views simplify complex queries and improve data security by restricting direct table access.

10. SYSTEM TESTING AND RESULT

System testing is performed to ensure that the Customer Relationship Management (CRM) database system works correctly, accurately, and reliably. In this project, testing focuses on validating SQL queries, verifying data integrity, and confirming that the output produced by the database matches expected results. The testing is carried out using MySQL through SQL execution.

10.1 Query Correctness Testing

Query correctness testing ensures that all SQL queries written for the CRM system are syntactically and logically correct.

- All SQL commands such as CREATE, INSERT, SELECT, UPDATE, and DELETE were executed without syntax errors
- Queries were tested individually to verify proper execution
- JOIN queries were tested to ensure correct data retrieval from multiple tables
- Advanced queries like GROUP BY, HAVING, subqueries, and views were checked for correct logic

Result:
All queries executed successfully and produced expected results, confirming the correctness of SQL implementation.

10.2 Data Validation

Data validation testing ensures that only valid and consistent data is stored in the database.

- PRIMARY KEY constraint prevented duplicate records
- FOREIGN KEY constraint ensured referential integrity between related tables
- NOT NULL constraint blocked insertion of incomplete records
- UNIQUE constraint avoided duplicate email entries

Invalid data insertion attempts were tested deliberately, and the database correctly rejected them.

Result:
The database successfully enforced all constraints, ensuring high data accuracy and consistency.

10.3 sample output

10.3.1 Database Creation and Use Database

```sql
CREATE DATABASE crm_db;
USE crm_db;
```

10.3.2 table creation (customer, employe, order, product, order details)

10.3.3 A. customer table;

```sql
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE,
    phone VARCHAR(15) UNIQUE,
    address VARCHAR(200),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

B. Employee table;

```sql
CREATE TABLE Employee (
    employee_id INT PRIMARY KEY AUTO_INCREMENT,
    employee_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE,
    phone VARCHAR(15),
    designation VARCHAR(50),
    salary DECIMAL(10,2) CHECK (salary > 0)
);
```

C. Product table;

```sql
CREATE TABLE Product (
    product_id INT PRIMARY KEY AUTO_INCREMENT,
    product_name VARCHAR(100) NOT NULL,
    price DECIMAL(10,2) CHECK (price > 0),
    stock INT CHECK (stock >= 0)
);
```

D. Order table;

```
CREATE TABLE Orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    employee_id INT,
    order_date DATE NOT NULL,
    total_amount DECIMAL(10,2),

    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
    ON DELETE CASCADE,

    FOREIGN KEY (employee_id) REFERENCES Employee(employee_id)
    ON DELETE SET NULL
);
```

E. Order details;

```
CREATE TABLE Order_Details (
    order_detail_id INT PRIMARY KEY AUTO_INCREMENT,
    order_id INT,
    product_id INT,
    quantity INT CHECK (quantity > 0),
    subtotal DECIMAL(10,2),

    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
    ON DELETE CASCADE,

    FOREIGN KEY (product_id) REFERENCES Product(product_id)
    ON DELETE CASCADE
);
```

10.3.3 table value insertion and display;

A. insert into customer table;

```
52 •  INSERT INTO Customer (customer_name, email, phone, address) VALUES
53    ('Aarav Sharma', 'aarav1@gmail.com', '9000000001', 'Mumbai'),
54    ('Vivaan Gupta', 'vivaan2@gmail.com', '9000000002', 'Delhi'),
55    ('Aditya Verma', 'aditya3@gmail.com', '9000000003', 'Pune'),
56    ('Vihaan Mehta', 'vihaan4@gmail.com', '9000000004', 'Jaipur'),
57    ('Arjun Singh', 'arjun5@gmail.com', '9000000005', 'Lucknow'),
58    ('Sai Patel', 'sai6@gmail.com', '9000000006', 'Ahmedabad'),
59    ('Reyansh Jain', 'reyansh7@gmail.com', '9000000007', 'Surat'),
60    ('Krishna Yadav', 'krishna8@gmail.com', '9000000008', 'Kolkata'),
61    ('Ishaan Roy', 'ishaan9@gmail.com', '9000000009', 'Chennai'),
62    ('Rudra Nair', 'rudra10@gmail.com', '9000000010', 'Bangalore'),
63
64    ('Anaya Sharma', 'anaya11@gmail.com', '9000000011', 'Mumbai'),
65    ('Diya Kapoor', 'diya12@gmail.com', '9000000012', 'Delhi'),
66    ('Myra Shah', 'myra13@gmail.com', '9000000013', 'Pune'),
67    ('Aadhya Singh', 'aadhya14@gmail.com', '9000000014', 'Jaipur'),
68    ('Pari Mishra', 'pari15@gmail.com', '9000000015', 'Lucknow'),
69    ('Sara Khan', 'sara16@gmail.com', '9000000016', 'Hyderabad'),
70    ('Riya Das', 'riya17@gmail.com', '9000000017', 'Kolkata'),
```

- **Display customer table**

```
107 •  LECT * FROM customer;
```

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

customer_id	customer_name	email	phone	address	created_at
1	Aarav Sharma	aarav1@gmail.com	9000000001	Mumbai	2026-02-22 22:50:42
2	Vivaan Gupta	vivaan2@gmail.com	9000000002	Delhi	2026-02-22 22:50:42
3	Aditya Verma	aditya3@gmail.com	9000000003	Pune	2026-02-22 22:50:42
4	Vihaan Mehta	vihaan4@gmail.com	9000000004	Jaipur	2026-02-22 22:50:42
5	Arjun Singh	arjun5@gmail.com	9000000005	Lucknow	2026-02-22 22:50:42
6	Sai Patel	sai6@gmail.com	9000000006	Ahmedabad	2026-02-22 22:50:42
7	Reyansh Jain	reyansh7@gmail.com	9000000007	Surat	2026-02-22 22:50:42
8	Krishna Yadav	krishna8@gmail.com	9000000008	Kolkata	2026-02-22 22:50:42

B. insert into employee table

```
108 •   SERT INTO Employee (employee_name, email, phone, designation, salary) VALUES
109     Amit Sharma', 'amit1@company.com', '9100000001', 'Sales Manager', 60000),
110     Neha Verma', 'neha2@company.com', '9100000002', 'Sales Executive', 35000),
111     Rahul Singh', 'rahul3@company.com', '9100000003', 'Sales Executive', 32000),
112     Priya Kapoor', 'priya4@company.com', '9100000004', 'HR Manager', 55000),
113     Vikas Mehta', 'vikas5@company.com', '9100000005', 'Accountant', 40000),
114     Sneha Jain', 'sneha6@company.com', '9100000006', 'Sales Executive', 33000),
115     Arjun Gupta', 'arjun7@company.com', '9100000007', 'Team Leader', 45000),
116     Pooja Reddy', 'pooja8@company.com', '9100000008', 'Marketing Executive', 38000),
117     Kunal Shah', 'kunal9@company.com', '9100000009', 'Sales Executive', 30000),
118     Anjali Mishra', 'anjali10@company.com', '9100000010', 'Customer Support', 28000),
```

- **Display employee table;**

```
163 •   SELECT * FROM employee;
```

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

employee_id	employee_name	email	phone	designation	salary
1	Amit Sharma	amit1@company.com	9100000001	Sales Manager	60000.00
2	Neha Verma	neha2@company.com	9100000002	Sales Executive	35000.00
3	Rahul Singh	rahul3@company.com	9100000003	Sales Executive	32000.00
4	Priya Kapoor	priya4@company.com	9100000004	HR Manager	55000.00
5	Vikas Mehta	vikas5@company.com	9100000005	Accountant	40000.00
6	Sneha Jain	sneha6@company.com	9100000006	Sales Executive	33000.00
7	Arjun Gupta	arjun7@company.com	9100000007	Team Leader	45000.00
8	Pooja Reddy	pooja8@company.com	9100000008	Marketing Executive	38000.00

C insert into product table

```
164 •   INSERT INTO Product (product_name, price, stock) VALUES
165     ('Laptop Pro 14', 75000, 15),
166     ('Laptop Air 13', 65000, 20),
167     ('Gaming Laptop X', 95000, 10),
168     ('Smartphone A1', 20000, 50),
169     ('Smartphone A2', 25000, 40),
170     ('Smartphone Pro Max', 60000, 25),
171     ('Tablet S1', 30000, 18),
172     ('Tablet S2', 35000, 12),
173     ('Smartwatch Z1', 12000, 60),
174     ('Smartwatch Z2', 15000, 45),
```

- **Display product table;**

```
219 •  SELECT * FROM product;
220
221 •  USE crm_db;
222 •  SELECT COUNT(*) FROM Product
223    ALTER TABLE Orders AUTO_INCREMENT = 101;
```

Result Grid | Filter Rows: | Edit: | Expo

product_id	product_name	price	stock
1	Laptop Pro 14	75000.00	15
2	Laptop Air 13	65000.00	20
3	Gaming Laptop X	95000.00	10
4	Smartphone A1	20000.00	50
5	Smartphone A2	25000.00	40
6	Smartphone Pro Max	60000.00	25
7	Tablet S1	30000.00	18
8	Tablet S2	35000.00	12

D. insert into order table

```
226 •  INSERT INTO Orders (customer_id, employee_id, order_date, total_amount) VALUES
227    (1,1,'2026-02-01',20000),
228    (2,2,'2026-02-02',35000),
229    (3,3,'2026-02-03',18000),
230    (4,4,'2026-02-04',25000),
231    (5,5,'2026-02-05',40000),
232    (6,6,'2026-02-06',15000),
233    (7,7,'2026-02-07',30000),
234    (8,8,'2026-02-08',28000),
235    (9,9,'2026-02-09',45000),
236    (10,10,'2026-02-10',50000),
```

- **Display order table;**

```
81 •  SELECT * FROM orders;
```

esult Grid | Filter Rows: | Edit: | Export/Imp

order_id	customer_id	employee_id	order_date	total_amount
101	1	1	2026-02-01	20000.00
102	2	2	2026-02-02	35000.00
103	3	3	2026-02-03	18000.00
104	4	4	2026-02-04	25000.00
105	5	5	2026-02-05	40000.00
106	6	6	2026-02-06	15000.00
107	7	7	2026-02-07	30000.00
108	8	8	2026-02-08	28000.00

ders 5 ×

E insert into order details

```
292 •   INSERT INTO Order_Details (order_id, product_id, quantity, subtotal) VALUES
293     (101,1,1,75000),
294     (102,2,1,65000),
295     (103,3,1,95000),
296     (104,4,2,40000),
297     (105,5,1,25000),
298     (106,6,1,60000),
299     (107,7,1,30000),
300     (108,8,1,35000),
301     (109,9,2,24000),
302     (110,10,1,15000),
```

- **Display order details;**

```
347 •   SELECT * FROM Order_Details;
```

Result Grid | Filter Rows: | Edit:

order_detail_id	order_id	product_id	quantity	subtotal
1	101	1	1	75000.00
2	102	2	1	65000.00
3	103	3	1	95000.00
4	104	4	2	40000.00
5	105	5	1	25000.00
6	106	6	1	60000.00
7	107	7	1	30000.00
8	108	8	1	35000.00

Order_Details 6

D. Join colums order with customer name

```
284 •   SELECT o.order_id, c.customer_name, o.order_date, o.total_amount
285     FROM Orders o
286     JOIN Customer c ON o.customer_id = c.customer_id;
287
```

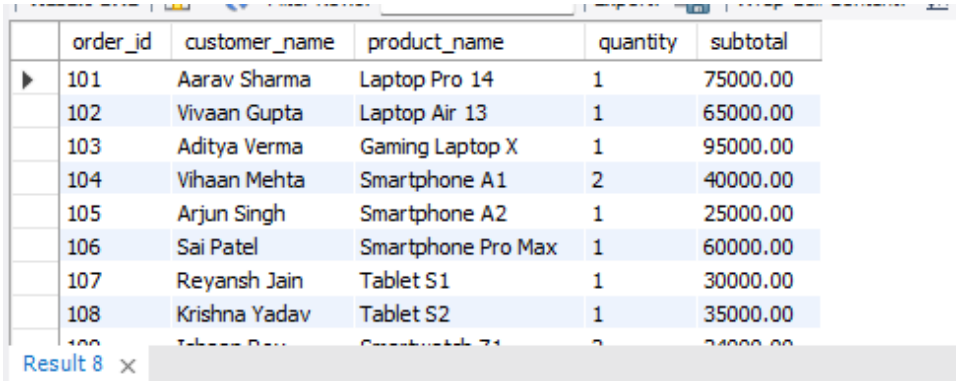
- **Display output**

order_id	customer_name	order_date	total_amount
101	Aarav Sharma	2026-02-01	20000.00
102	Vivaan Gupta	2026-02-02	35000.00
103	Aditya Verma	2026-02-03	18000.00
104	Vihaan Mehta	2026-02-04	25000.00
105	Arjun Singh	2026-02-05	40000.00
106	Sai Patel	2026-02-06	15000.00
107	Reyansh Jain	2026-02-07	30000.00
108	Krishna Yadav	2026-02-08	28000.00

E. Join columes order with customer name and products

```
288 •   SELECT o.order_id, c.customer_name, p.product_name, od.quantity, od.subtotal
289     FROM Order_Details od
290     JOIN Orders o ON od.order_id = o.order_id
291     JOIN Customer c ON o.customer_id = c.customer_id
292     JOIN Product p ON od.product_id = p.product_id;
```

- **Display output**



order_id	customer_name	product_name	quantity	subtotal
101	Aarav Sharma	Laptop Pro 14	1	75000.00
102	Vivaan Gupta	Laptop Air 13	1	65000.00
103	Aditya Verma	Gaming Laptop X	1	95000.00
104	Vihaan Mehta	Smartphone A1	2	40000.00
105	Arjun Singh	Smartphone A2	1	25000.00
106	Sai Patel	Smartphone Pro Max	1	60000.00
107	Reyansh Jain	Tablet S1	1	30000.00
108	Krishna Yadav	Tablet S2	1	35000.00

Result 8

11. SECURITY, BACKUP AND RECOVERY

Security, backup, and recovery are essential components of a reliable **Customer Relationship Management (CRM)** database system. They ensure that customer data is protected from unauthorized access, data loss, and system failures. In this project, these aspects are implemented using features provided by **MySQL**.

11.1 Security

User Privileges

User privileges define what actions a user is allowed to perform on the database. In MySQL, different privileges can be assigned to different users based on their roles.

- **Administrator (Admin):**
  Has full privileges such as CREATE, INSERT, UPDATE, DELETE, and DROP tables.
- **Normal User:**
  Has limited privileges such as SELECT, INSERT, and UPDATE only.
- **Read-Only User:**
  Can only view data using SELECT queries.

By assigning appropriate privileges, the system prevents unauthorized or accidental modification of sensitive customer data.

Access Control

Access control ensures that only **authorized users** can access the CRM database.

- User authentication is done using username and password
- Role-based access restricts users from performing unauthorized operations
- Sensitive tables can be protected by granting access only to specific users

This controlled access improves data security and maintains confidentiality of customer information.

11.2 Backup

mysqldump Explanation

mysqldump is a MySQL utility used to create a **logical backup** of a database. It exports database structure and data into a SQL file, which can be stored safely for future use.

Key features of mysqldump:

- Creates backups without stopping the database server
- Stores table structure and data in SQL format
- Can back up a single database or multiple databases

Purpose of Backup:

- Protects against accidental data deletion
- Prevents data loss due to system failure
- Enables data restoration when required

Regular backups are recommended to ensure data safety and system reliability.

11.3 Recovery

Restore Concept

Recovery refers to the process of **restoring the database** from a previously taken backup.

- In case of data loss, corruption, or system crash, the backup file is used
- The SQL file created by mysqldump is executed to recreate tables and data
- Restoration brings the database back to its last saved state

The restore process ensures business continuity and minimizes downtime by quickly recovering lost or damaged data.

12. FUTURE SCOPE AND CONCLUSION

This section discusses the **future enhancements** of the Customer Relationship Management (CRM) database system and summarizes the overall outcomes of the project. The project has been implemented using **MySQL**, which provides a strong foundation for further development.

12.1 Future Scope

Although the current CRM system fulfills basic academic and functional requirements, it can be extended further to meet advanced organizational needs.

- **Web Integration**

The database system can be integrated with a **web-based application** using technologies such as HTML, CSS, JavaScript, and server-side scripting. This would allow users to access the CRM system through a browser, enabling remote access and real-time updates.

- **Advanced Reporting**

Future enhancements may include **dynamic and customizable reports** such as sales summaries, customer activity reports, and payment analysis. Graphical reports using charts and dashboards can help management make better decisions.

- **Analytics Features**

Advanced analytics can be implemented to analyze customer behavior, purchasing patterns, and trends. Features such as customer segmentation and performance analysis can further improve business strategies and customer satisfaction.

12.2 Conclusion

This project successfully achieved the design and implementation of a **Customer Relationship Management (CRM) database system** using a relational database approach.

What Was Achieved

- Designed a structured CRM database with well-defined tables and relationships
- Implemented SQL queries for data creation, manipulation, and retrieval
- Ensured data integrity using constraints and relationships

- Tested the system for correctness, validation, and reliable output

Learning Outcomes

- Gained practical knowledge of database design and normalization
- Improved understanding of SQL queries, joins, subqueries, and views
- Learned the importance of data security, backup, and recovery
- Developed hands-on experience with real-world CRM concepts

Importance of MySQL

MySQL proved to be an efficient, reliable, and cost-effective database management system. Its open-source nature, ease of use, strong security features, and wide industry adoption make it ideal for academic projects as well as real-world applications.

13. REFERENCES

The following references were used for understanding concepts, designing the database, and implementing the CRM system using SQL and MySQL:

1. **MySQL Official Documentation**
   MySQL Documentation – Reference manuals and guides for SQL syntax, database design, security, backup, and recovery.
2. **Prescribed Textbooks**
   - **Database System Concepts** – Abraham Silberschatz, Henry F. Korth, S. Sudarshan
   - **Fundamentals of Database Systems** – Ramez Elmasri and Shamkant B. Navathe
3. **Online Learning Sources**
   - Tutorials and learning materials from educational websites on DBMS and SQL
   - Online articles and examples related to CRM database design
   - Video lectures and practice exercises for MySQL and SQL concepts

14. GLOSSARY

This glossary defines important technical terms used in the **Customer Relationship Management (CRM)** database project.

- **DBMS (Database Management System):**
  Software that allows users to create, store, manage, and retrieve data from databases in an efficient and secure manner.
- **SQL (Structured Query Language):**
  A standard programming language used to define, manipulate, and control data in a relational database system.
- **Primary Key:**
  A field or combination of fields that uniquely identifies each record in a database table. It does not allow duplicate or NULL values.
- **Foreign Key:**
  A field in one table that refers to the primary key of another table. It is used to establish relationships between tables and maintain referential integrity.
- **MySQL:**
  An open-source relational database management system that uses SQL for data storage, retrieval, and management. It is widely used for academic and real-world applications.