SAVITRIBAI PHULE PUNE UNIVERSITY

A MINI-PROJECT REPORTON

"ATM Management System"

(A project to fulfill the requirements of Database Management System Lab)

By

Dere Tanaya Balasaheb	304A051
Deshmane Nikhil Laxman	304A052
Deshmukh Saharsh Hemant	304A053
Deshmukh Suraj Mahesh	304A054
Deshmukh Sushil Diliprao	304A055

Under the guidance of

Prof. S. S. Patil

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION

SINHGAD INSTITUTE OF TECHNOLOGY & SCIENCE, PUNE

(Academic Year: 2023 – 2024)

Sinhgad Institutes

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION

SINHGAD COLLEGE OF ENGINEERING
VADGAON, PUNE

CERTIFICATE

This is to certify that final project work entitled **"ATM Management System"** was successfully carried by

1. **Dere Tanaya Balasaheb**
2. **Deshmane Nikhil Laxman**
3. **Deshmukh Saharsh Hemant**
4. **Deshmukh Suraj Mahesh**
5. **Deshmukh Sushil Diliprao**

In the partial fulfillment of the DATABASE MANAGEMENT SYSTEM LAB course during Semester-V of Third Year of ELECTRONICS AND TELECOMMUNICATION prescribed by the SAVITRIBAI PHULE PUNE UNIVERSITY,PUNE.

Date:
Place:

Guide **H.O.D**

(Dr.S.S.Patil) **(Dr.M.B.Mali)**

Principal

(Dr. S. D. Lokhande)

ACKNOWLEDGEMENT

I feel great pleasure in expressing my deepest sense of gratitude and sincere thanks to my guide **Prof.S.S.Patil** for their valuable guidance during the Project work, without which it would have been very difficult task. I have no words to express my sincere thanks for valuable guidance, extreme assistance and cooperation extended to all the **Staff Members** of Department of **Electronics And Telecommunication**.

This acknowledgement would be incomplete without expressing my special thanks to **Dr.M. B.Mali** Head of the Department (Electronics And Telecommunication) for their support during the work. I would also like to extend my heartfelt gratitude to my Principal, **Dr.S.D.Lokhande** who provided a lot of valuable support, mostly being behind the veils of college bureaucracy.

Last but not least I would like to thanks all the Teaching, Non- Teaching staff members of my Department, my parent and my colleagues those who helped me directly or indirectly for completing of this Project successfully.

Sr.No	**Name Of Students**	**Roll No.**
1.	Dere Tanaya Balasaheb	304A051
2.	Deshmane Nikhil Laxman	304A052
3.	Deshmukh Saharsh Hemant	304A053
4.	Deshmukh Suraj Mahesh	304A054
5.	Deshmukh Sushil Diliprao	304A055

ABSTRACT

The ATM Management System is a Database Management System (DBMS) that focuses on efficiently managing and maintaining Automated Teller Machines (ATMs). It provides a robust and secure platform for storing, organizing, and retrieving ATM-related data. The system's database stores information about ATMs, customers, transactions, and maintenance activities. It allows authorized personnel, such as bank administrators and technicians, to access and manipulate this data to ensure smooth ATM operations and a positive user experience.

Key features of the ATM Management System DBMS project include:
ATM Data Management, Customer Data Management, Transaction Management, Maintenance and Service Management, Reporting and Analytics.

The ATM Management System DBMS aims to enhance the efficiency, security, and reliability of ATM operations. By leveraging database management capabilities, it ensures the availability of accurate information, streamlined processes, and a seamless user experience.

TABLE OF CONTENT

CHAPTER: 1
INTRODUCTION

➢ Overview:

The ATM Database Management System project aims to develop a software solution that efficiently manages and maintains Automated Teller Machines (ATMs). It utilizes a centralized database to store and organize ATM-related data. The system facilitates secure transactions, optimized cash management, timely maintenance scheduling, and reporting and analytics. It follows a client-server architecture, with a user-friendly interface for authorized personnel. The core functionalities include ATM and customer data management, transaction processing, cash management, maintenance management, and reporting. The system utilizes MySQL for data storage and retrieval, ensuring reliability and scalability.

➢ Background Study:

The ATM Management System is developed to address the need for efficient management and maintenance of Automated Teller Machines (ATMs). ATMs have become an integral part of modern banking, providing convenient access to banking services for customers. However, managing a network of ATMs poses various challenges, including cash management, transaction processing, maintenance scheduling, and data security.

➢ Objective:

The objective of the ATM Database Management System is to develop a comprehensive software solution that efficiently manages and maintains Automated Teller Machines (ATMs) by creating a centralized database. It aims to streamline ATM operations, ensure secure transactions, optimize cash management, schedule timely maintenance, and generate insightful reports for data-driven decision making.

CHAPTER: 2
SYSTEM DESIGN

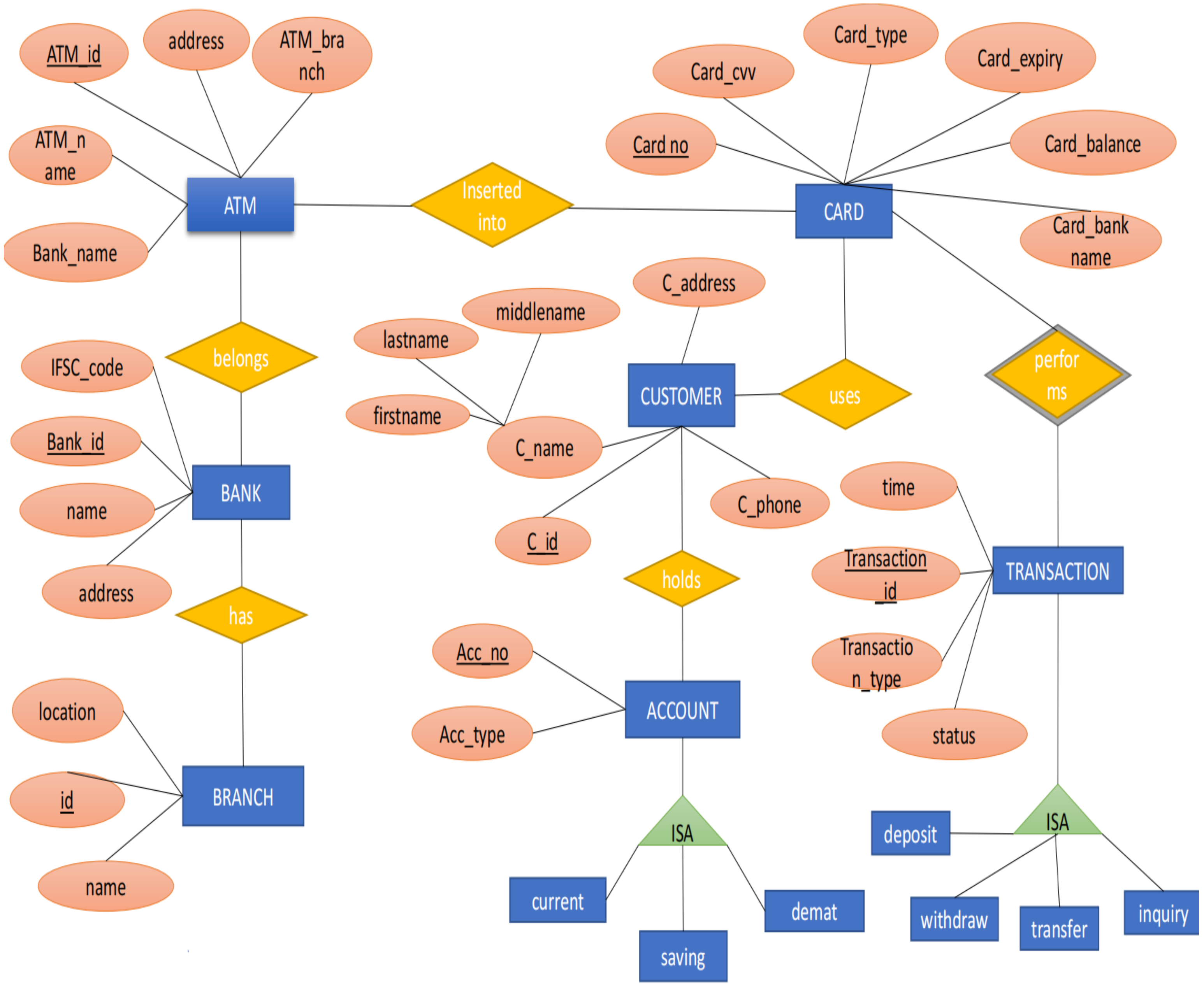


Fig A. The ER conceptual schema diagram for the ATM Management database.

➢ System Information :

1. **System Overview:**

The ATM Management System is a software solution designed to facilitate the management and operation of Automated Teller Machines (ATMs) within a banking environment. It enables banks to efficiently handle various ATM-related tasks such as customer transactions, card management, account management, and monitoring of ATM operations.

2. **Functionalities:**

User Authentication: The system provides secure user authentication mechanisms to ensure that only authorized individuals can access the system and perform transactions.
Customer Account Management: It allows banks to manage customer accounts associated with ATMs, including account creation, updating account information, and maintaining account balances.
Card Management: The system handles card-related operations such as issuing new cards, managing card details, and ensuring card validity.
Transaction Processing: It facilitates various types of transactions such as cash withdrawals, balance inquiries, fund transfers, and account deposits.
ATM Monitoring: The system monitors the status and performance of ATMs, including cash availability, machine health, and transaction logs.
Reporting and Analytics: It generates reports and provides analytics related to ATM usage, transaction trends, and system performance, helping banks make informed decisions.

3. **System Components:**

User Interface: The system includes a user interface that allows bank staff and customers to interact with the system. It provides screens for transaction processing, account management, and reporting. Database: The system utilizes a database to store and manage ATM-related data such as customer information, account details, transaction records, ATM configurations, and system logs.
Transaction Processing Engine: It includes the logic and algorithms to process various types of transactions requested by customers, ensuring accurate and secure execution.
Security Module: The system incorporates robust security measures to protect sensitive data, authenticate users, and prevent unauthorized access.
ATM Hardware Integration: The system interfaces with the physical ATM hardware components to enable transaction execution, cash dispensing, card reading, and receipt printing.

4. **Integration and Interfaces**:

Integration with Banking Systems: The ATM Management System integrates with existing banking systems, such as core banking systems, to retrieve customer account information and update transaction records.
ATM Network Communication: It communicates with the ATM network infrastructure to facilitate real-time transaction processing, authorization, and settlement.
External Services: The system may integrate with external services such as payment gateways, card networks, and fraud detection systems to ensure secure and seamless ATM operations.

5. **Security Considerations:**

Encryption: The system implements encryption protocols to secure communication channels, protect sensitive data, and prevent unauthorized interception.
Access Control: It employs access control mechanisms to restrict system access based on user roles and

permissions, ensuring that only authorized personnel can perform specific tasks.
Auditing and Logging: The system logs user activities, transaction details, and system events for auditing and monitoring purposes, aiding in detecting and investigating any security breaches.
Compliance: The system adheres to regulatory and industry standards, such as Payment Card Industry Data Security Standard (PCI DSS), to maintain the security and integrity of customer data.

➢ System Characteristics :

1) **Centralized Database:** The system utilizes a centralized database to store and manage ATM-related data, including ATM details, customer information, transaction records, and maintenance activities.

2) **Efficient Data Storage and Retrieval:** MySQL, a robust and widely-used database management system, is employed for efficient data storage, retrieval, and management within the system.

3) **User-Friendly Interface:** The system provides a user-friendly interface for authorized personnel, such as bank administrators and technicians, to interact with the database, perform operations, and access relevant information.

4) **Cash Management Optimization:** Through data analysis and forecasting, the system optimizes cash management by ensuring the availability of adequate cash supplies in ATMs, reducing cash-outs, and improving cash utilization.

5) **Security and Data Integrity:** The system implements robust security measures to protect sensitive customer information, ensuring data confidentiality and integrity. Measures such as encryption and authentication mechanisms safeguard against unauthorized access.

6) **Scalability and Future Expandability:** The system can be designed to handle a growing number of ATMs and users, accommodating larger data volumes. It can also be expanded to integrate advanced technologies such as biometric authentication, mobile integration, and blockchain for enhanced functionality and security.

CHAPTER: 3
SCHEMA DIAGRAM

➢ Steps of converting ER to Tables:

Converting an Entity-Relationship (ER) diagram to a set of tables involves the process of mapping the entities, attributes, and relationships in the diagram to their corresponding table structures in a relational database.

1. **Identify Entities**: Review the ER diagram and identify all the entities represented in it. Entities typically represent the real-world objects or concepts being modeled. Each entity will correspond to a table in the database.

2. **Define Attributes**: For each entity, identify and list down its attributes. Attributes represent the properties or characteristics of an entity. Determine the data type for each attribute (e.g., string, integer, date, etc.) based on the domain of the attribute.

3. **Determine Primary Keys**: Identify the primary key for each entity. A primary key is a unique identifier for each record in the table. It can be a single attribute or a combination of attributes that uniquely identify each entity instance.

4. **Create Tables**: Create a table for each entity identified in Step 1. The table name should correspond to the entity name, and each attribute should become a column in the table. The primary key attribute(s) should be specified as the primary key constraint for the table.

5. **Establish Relationships:** Analyze the relationships between entities in the ER diagram. Relationships can be one-to-one, one-to-many, or many-to-many. Based on the cardinality and the type of relationship, you need to establish the appropriate relationship between the tables.
   a) One-to-One Relationship
   b) One-to-Many Relationship
   c) Many-to-Many Relationship

6. **Define Foreign Keys**: For each relationship established in Step 5, define the appropriate foreign keys in the related tables. Foreign keys ensure referential integrity and maintain the relationship between entities.

7. **Set Constraints and Indexes**: Define any additional constraints, such as uniqueness constraints or data validation rules, based on the requirements of the ER diagram. Additionally, consider adding indexes to improve query performance on frequently accessed columns.

8. **Normalize the Tables** : Analyze the tables for normalization and make adjustments to ensure data integrity and eliminate redundancy. This step may involve splitting tables or creating additional tables based on normalization rule

9. **Review and Refine:** Once the initial conversion is complete, review the table Structures, relationships, and constraints to ensure they accurately represent the requirements of the ER diagram. Make any necessary refinements or adjustments.

➢ SCHEMA DIAGRAM :

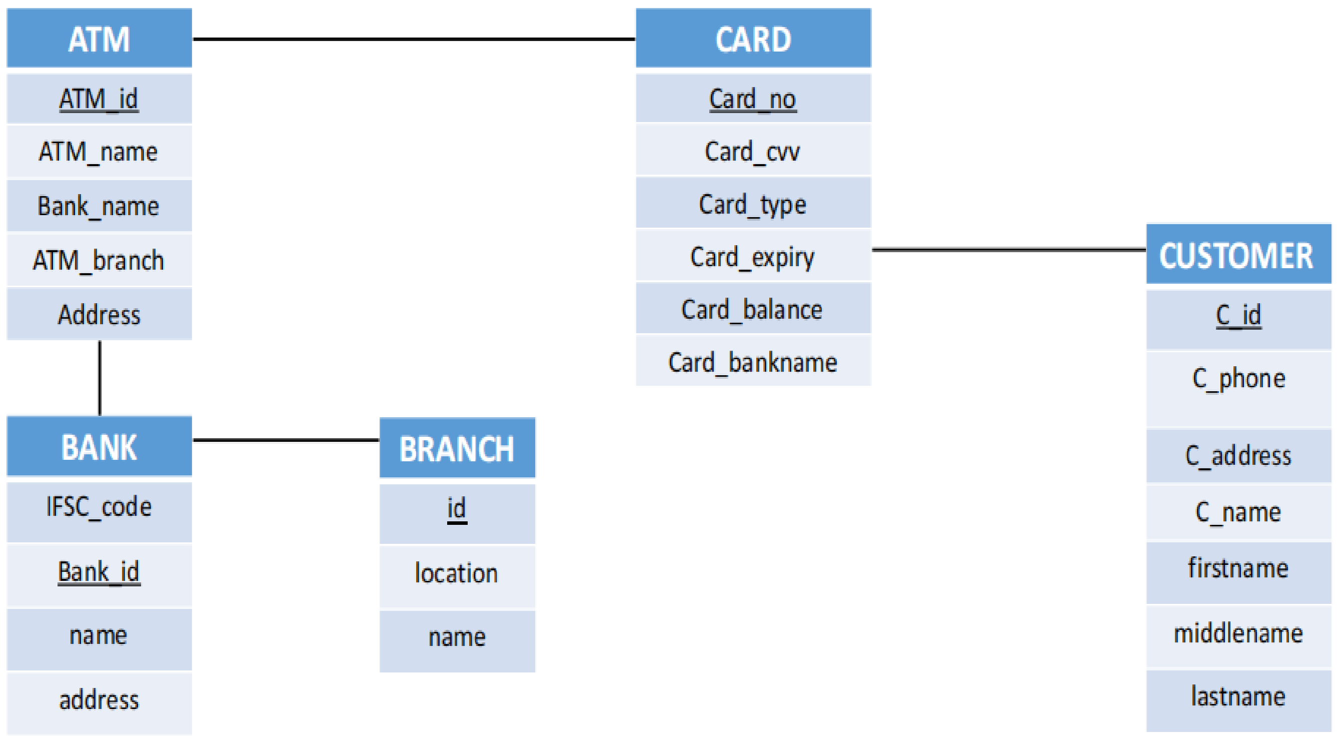


Fig B: ATM Management Schema Diagram

CHAPTER: 4
DATABASE DESIGN

4.2 CREATING DATABASE USING MYSQL :

An SQL schema is identified by a schema name, and includes an authorization identifier to indicate the user or account who owns the schema, as well as descriptors for each element in the schema. Schema elements include tables, constraints, views, domains, and other constructs (such as authorization grants) that describe the schema. A schema is created via the CREATE SCHEMA statement, which can include all the schema elements' definitions. Alternatively, the schema can be assigned a name and authorization identifier, and the elements can be defined later. The CREATE TABLE command is used to specify a new relation by giving it a name and specifying its attributes and initial constraints. The attributes are specified first, and each attribute is given a name, a data type to specify its domain of values, and any attribute constraints, such as NOT NULL. The key, entity integrity, and referential integrity constraints can be specified within the CREATE TABLE statement after the attributes are declared, or they can be added later using the ALTER TABLE command.

Sample data definition statements in SQL for the ATM management relational database schema:

```
create database ATMmanage;
Use ATMmanage;

-- Create the ATM table
CREATE TABLE ATM (
  ATM_id INT PRIMARY KEY NOT NULL,
  ATM_name VARCHAR(60),
  ATM_branch VARCHAR(60),
  address VARCHAR(60),
  Bank_name VARCHAR(100)
);
```

```
-- Insert values into the ATM table
INSERT INTO ATM (ATM_id, ATM_name, ATM_branch, address, Bank_name)
VALUES
  (1, 'ATM-001', 'Branch A', 'Sinhgad', 'State Bank Of india'),
  (2, 'ATM-002', 'Branch B', 'vanourie', 'Central bank of india'),
  (3, 'ATM-003', 'Branch C', 'Nashik', 'Maharashtra bank'),
  (4, 'ATM-004', 'Branch D', 'Pune', 'Paytm Bank');
SELECT *FROM ATM;

-- Create the CARD table
CREATE TABLE CARD (
  Card_no INT PRIMARY KEY NOT NULL,
  Card_cvv VARCHAR(30),
  Card_type VARCHAR(30),
  Card_expiry VARCHAR(30),
  Card_balance DECIMAL(10, 2),
  Card_bankname VARCHAR(30)
);

-- Insert values into the CARD table
INSERT INTO CARD (Card_no, Card_cvv, Card_type, Card_expiry, Card_balance, Card_bankname)
VALUES
  (12345678, '123', 'Visa', '12/25', 1000.00, 'State Bank Of india'),
  (98765432, '456', 'MasterCard', '06/24', 500.00, 'Central bank of india'),
  (33334444, '789', 'Visa', '09/23', 2500.00, 'Maharashtra bank'),
  (44445555, '987', 'MasterCard', '03/26', 10000.00, 'Paytm Bank');

SELECT *FROM CARD;
-- Create the CUSTOMER table
```

```
CREATE TABLE CUSTOMER (
  C_id INT PRIMARY KEY NOT NULL,
  First_name VARCHAR(30),
  Middle_name VARCHAR(30),
  Last_name VARCHAR(30),
  C_phone VARCHAR(30),
  C_address VARCHAR(30)
);

-- Insert values into the CUSTOMER table
INSERT INTO CUSTOMER (C_id, First_name, Middle_name, Last_name, C_phone, C_address)
VALUES
  (1, 'siddhesh', 'Omkar', 'rushiraj', '123-456-7890', 'Mumbai'),
  (2, 'Sarthak', 'Ritesh', 'Pranit', '987-654-3210', 'pune'),
  (3, 'Vedang', 'Sudarshan', 'Aakash', '555-123-4567', 'nashik'),
  (4, 'ayush', 'James', 'Johnson', '555-987-6543', 'Rajastan');

Select *FROM CUSTOMER;
-- Create the BANK table
CREATE TABLE BANK (
  Bank_id INT PRIMARY KEY NOT NULL,
  IFSC_code VARCHAR(30),
  Bank_name VARCHAR(30),
  Address VARCHAR(50)
);

-- Insert values into the BANK table
INSERT INTO BANK (Bank_id, IFSC_code, Bank_name, Address)
VALUES
  (1, 'XYZ123', 'Maharashtra bank', 'nashik'),
  (2, 'ABC456', 'State Bank of India', 'mumbai'),
  (3, 'XYZ789', 'Central bank of india', 'pune'),
  (4, 'ABC123', 'Paytm bank', 'rajasthan');

Select *FROM BANK;
-- Create the BRANCH table
CREATE TABLE BRANCH (
  id INT,
  FOREIGN KEY (id) REFERENCES ATM(ATM_id),
  name VARCHAR(30),
  location VARCHAR(30)
);

-- Insert values into the BRANCH table
INSERT INTO BRANCH (id, name, location)
VALUES
  (1, 'Branch A', 'Pune'),
  (2, 'Branch B', 'Mumbai'),
  (3, 'Branch C', 'Nashik'),
  (4, 'Branch D', 'Rajasthan');

  SELECT *FROM BRANCH;

-- Create the ACCOUNT table
CREATE TABLE ACCOUNT (
  Acc_no INT PRIMARY KEY NOT NULL,
  Acc_type VARCHAR(30)
);

-- Insert values into the ACCOUNT table
```

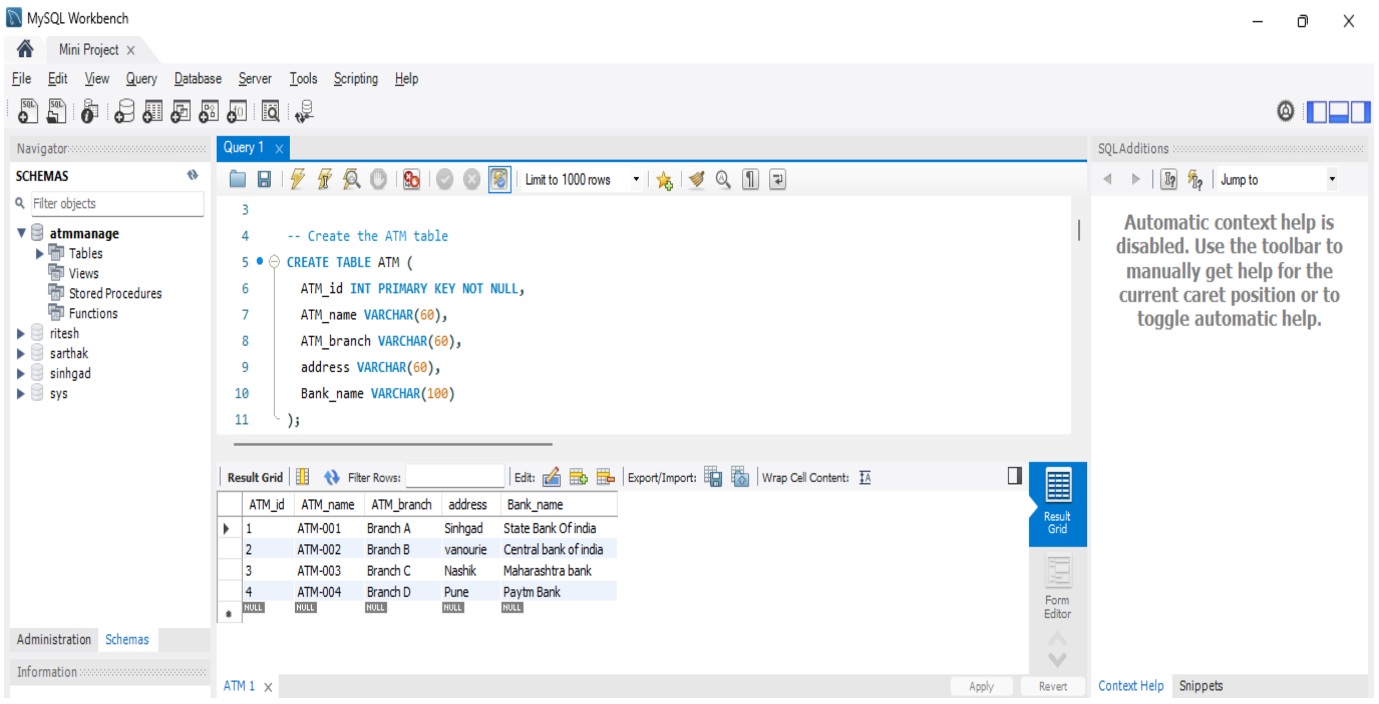
```
INSERT INTO ACCOUNT (Acc_no, Acc_type)
VALUES
  (123456, 'Savings'),
  (987654, 'Checking'),
  (555555, 'Savings'),
  (666666, 'Checking');

  SELECT *FROM ACCOUNT;

-- Create the TRANSACTION table
CREATE TABLE TRANSACTION (
  Transaction_id INT PRIMARY KEY NOT NULL,
  Transaction_type VARCHAR(30),
  Transaction_time TIMESTAMP,
  status VARCHAR(30)
);

-- Insert values into the TRANSACTION table
INSERT INTO TRANSACTION (Transaction_id, Transaction_type, Transaction_time, status)
VALUES
  (1, 'Withdrawal', NOW(), 'Success'),
  (2, 'Deposit', NOW(), 'Success'),
  (3, 'Transfer', NOW(), 'Pending'),
  (4, 'Withdrawal', NOW(), 'Success');

  SELECT *FROM TRANSACTION;
```

Chapter 4

4. DATABASE DESIGN

4.1 SQL code of database design

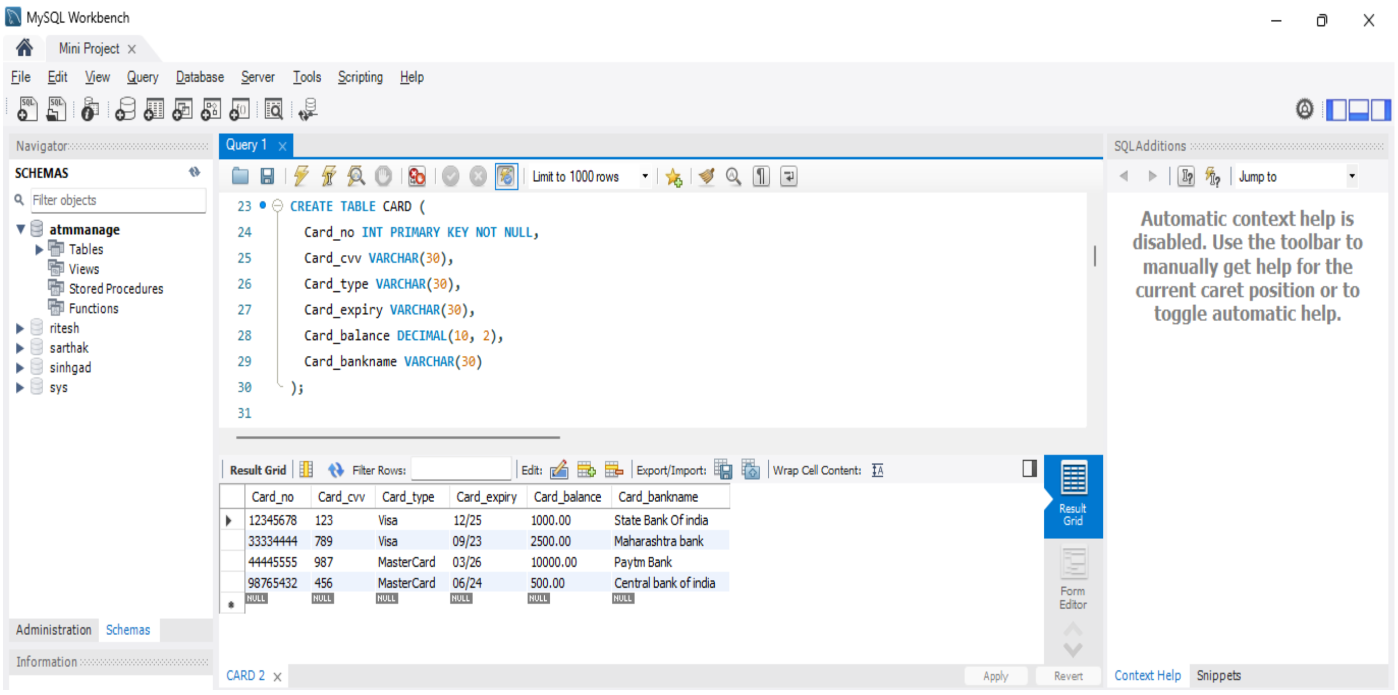
4.1.1 ATM table



```sql
-- Create the ATM table
CREATE TABLE ATM (
  ATM_id INT PRIMARY KEY NOT NULL,
  ATM_name VARCHAR(60),
  ATM_branch VARCHAR(60),
  address VARCHAR(60),
  Bank_name VARCHAR(100)
);
```

ATM_id	ATM_name	ATM_branch	address	Bank_name
1	ATM-001	Branch A	Sinhgad	State Bank Of india
2	ATM-002	Branch B	vanourie	Central bank of india
3	ATM-003	Branch C	Nashik	Maharashtra bank
4	ATM-004	Branch D	Pune	Paytm Bank
NULL	NULL	NULL	NULL	NULL

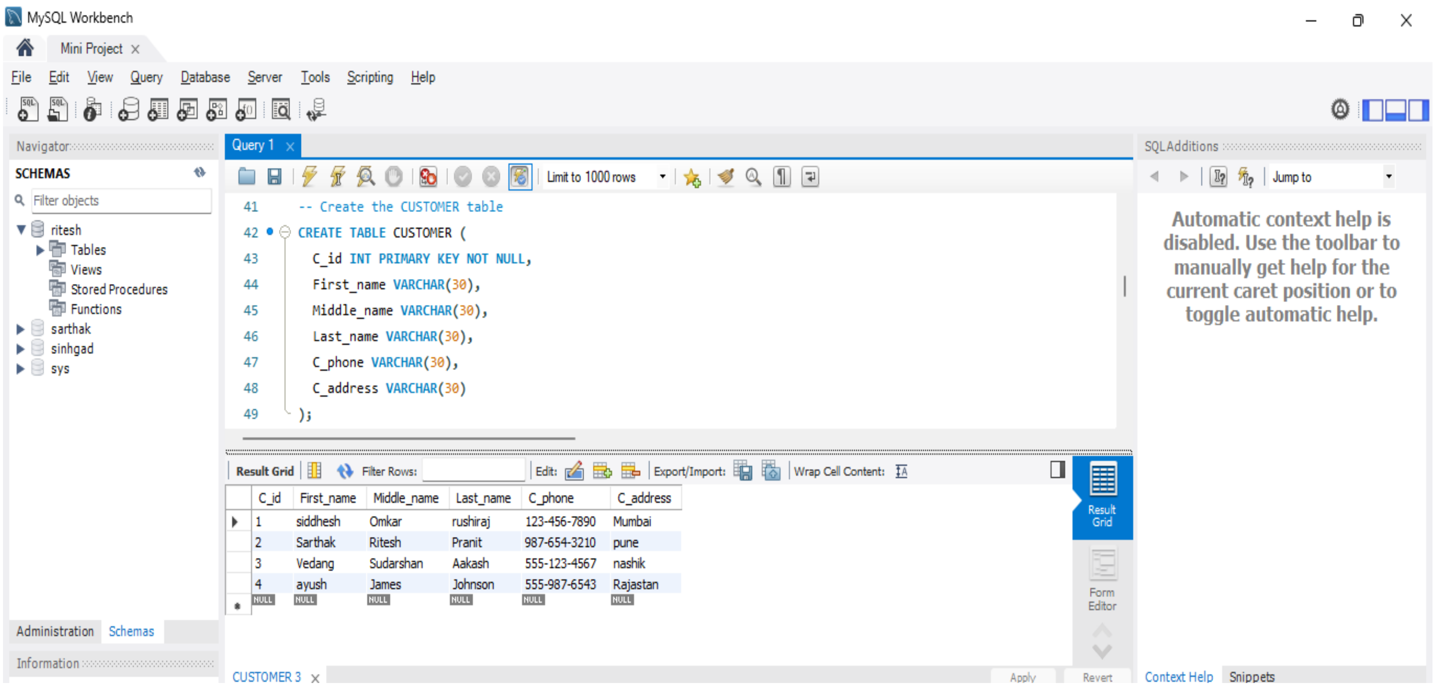
4.1.2 CARD table



```
CREATE TABLE CARD (
    Card_no INT PRIMARY KEY NOT NULL,
    Card_cvv VARCHAR(30),
    Card_type VARCHAR(30),
    Card_expiry VARCHAR(30),
    Card_balance DECIMAL(10, 2),
    Card_bankname VARCHAR(30)
);
```

Card_no	Card_cvv	Card_type	Card_expiry	Card_balance	Card_bankname
12345678	123	Visa	12/25	1000.00	State Bank Of india
33334444	789	Visa	09/23	2500.00	Maharashtra bank
44445555	987	MasterCard	03/26	10000.00	Paytm Bank
98765432	456	MasterCard	06/24	500.00	Central bank of india
NULL	NULL	NULL	NULL	NULL	NULL

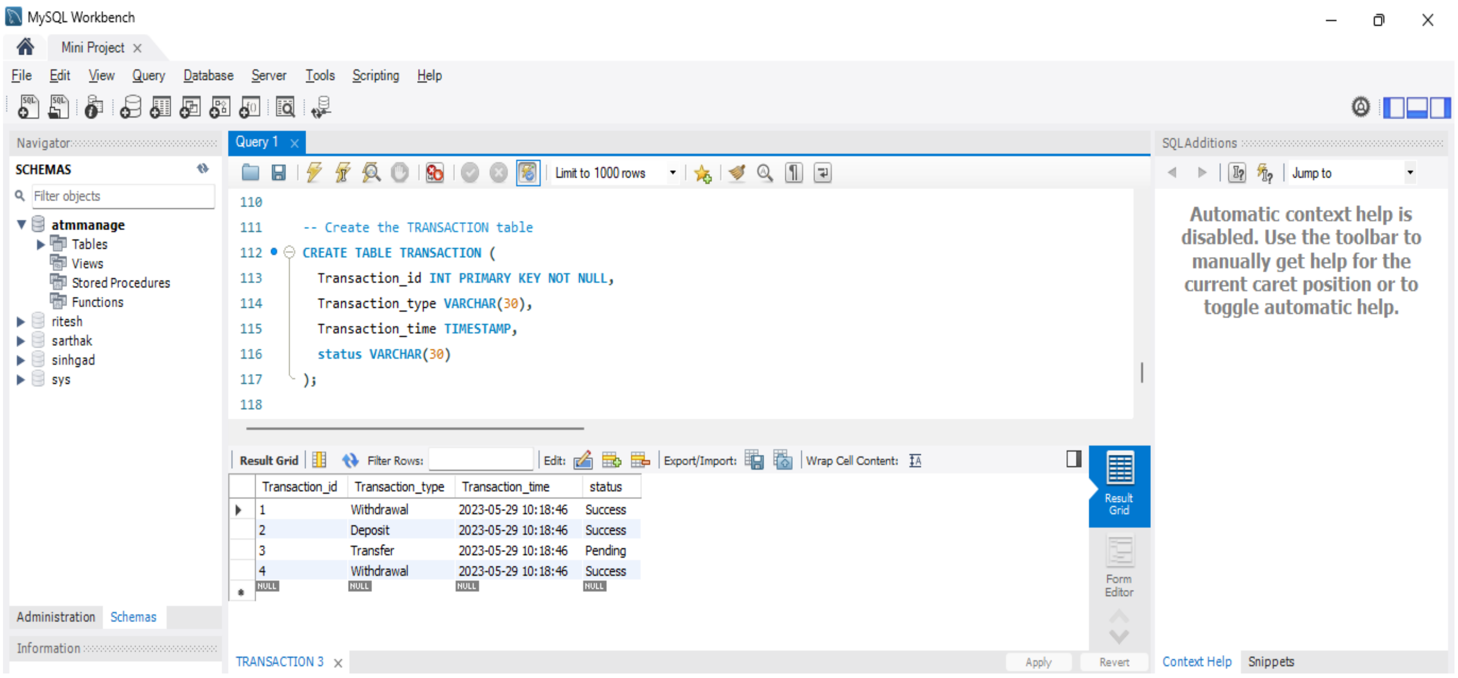
4.1.3 CUSTOMER Table



```
-- Create the CUSTOMER table
CREATE TABLE CUSTOMER (
    C_id INT PRIMARY KEY NOT NULL,
    First_name VARCHAR(30),
    Middle_name VARCHAR(30),
    Last_name VARCHAR(30),
    C_phone VARCHAR(30),
    C_address VARCHAR(30)
);
```

C_id	First_name	Middle_name	Last_name	C_phone	C_address
1	siddhesh	Omkar	rushiraj	123-456-7890	Mumbai
2	Sarthak	Ritesh	Pranit	987-654-3210	pune
3	Vedang	Sudarshan	Aakash	555-123-4567	nashik
4	ayush	James	Johnson	555-987-6543	Rajastan
NULL	NULL	NULL	NULL	NULL	NULL

4.1.4 Transaction Table

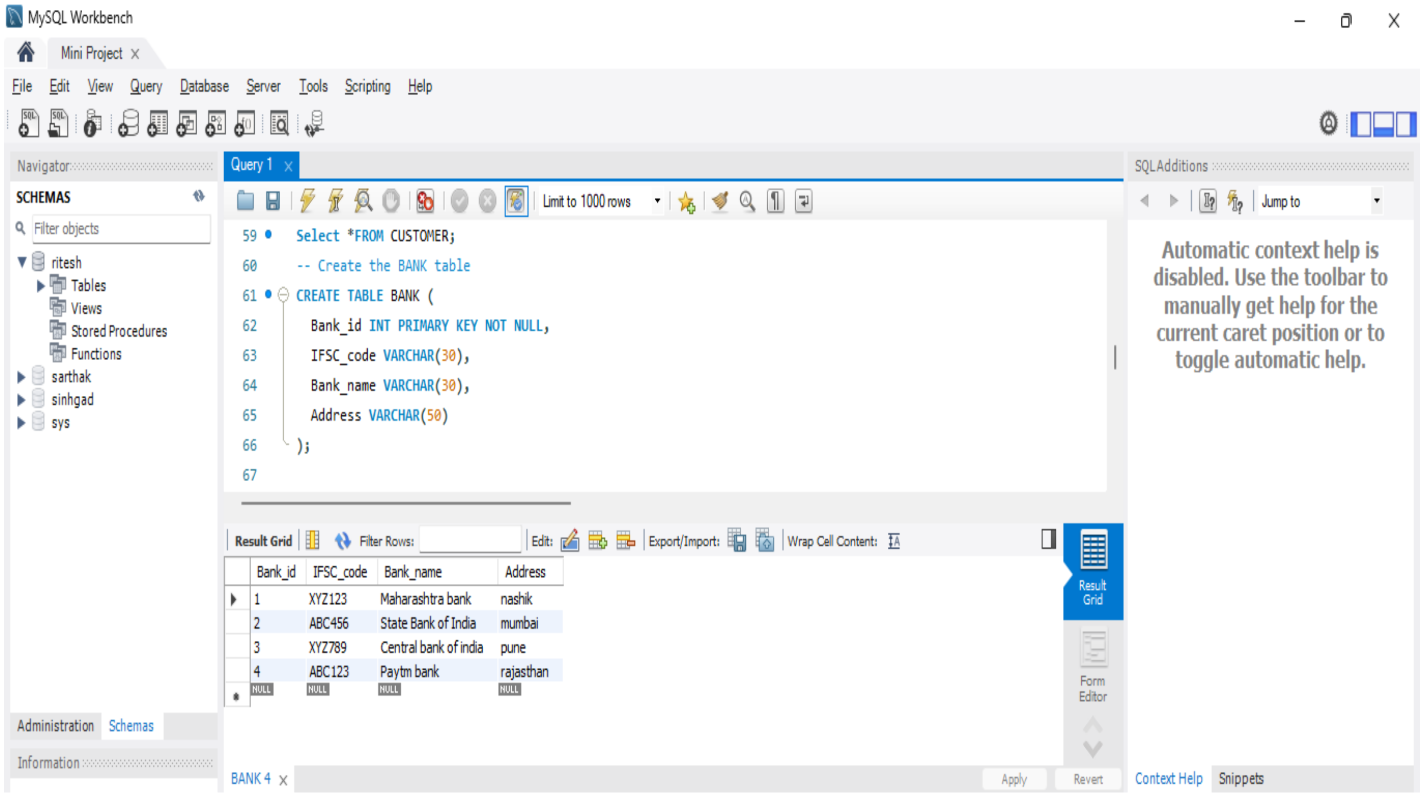


```
-- Create the TRANSACTION table
CREATE TABLE TRANSACTION (
  Transaction_id INT PRIMARY KEY NOT NULL,
  Transaction_type VARCHAR(30),
  Transaction_time TIMESTAMP,
  status VARCHAR(30)
);
```

Transaction_id	Transaction_type	Transaction_time	status
1	Withdrawal	2023-05-29 10:18:46	Success
2	Deposit	2023-05-29 10:18:46	Success
3	Transfer	2023-05-29 10:18:46	Pending
4	Withdrawal	2023-05-29 10:18:46	Success
NULL	NULL	NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

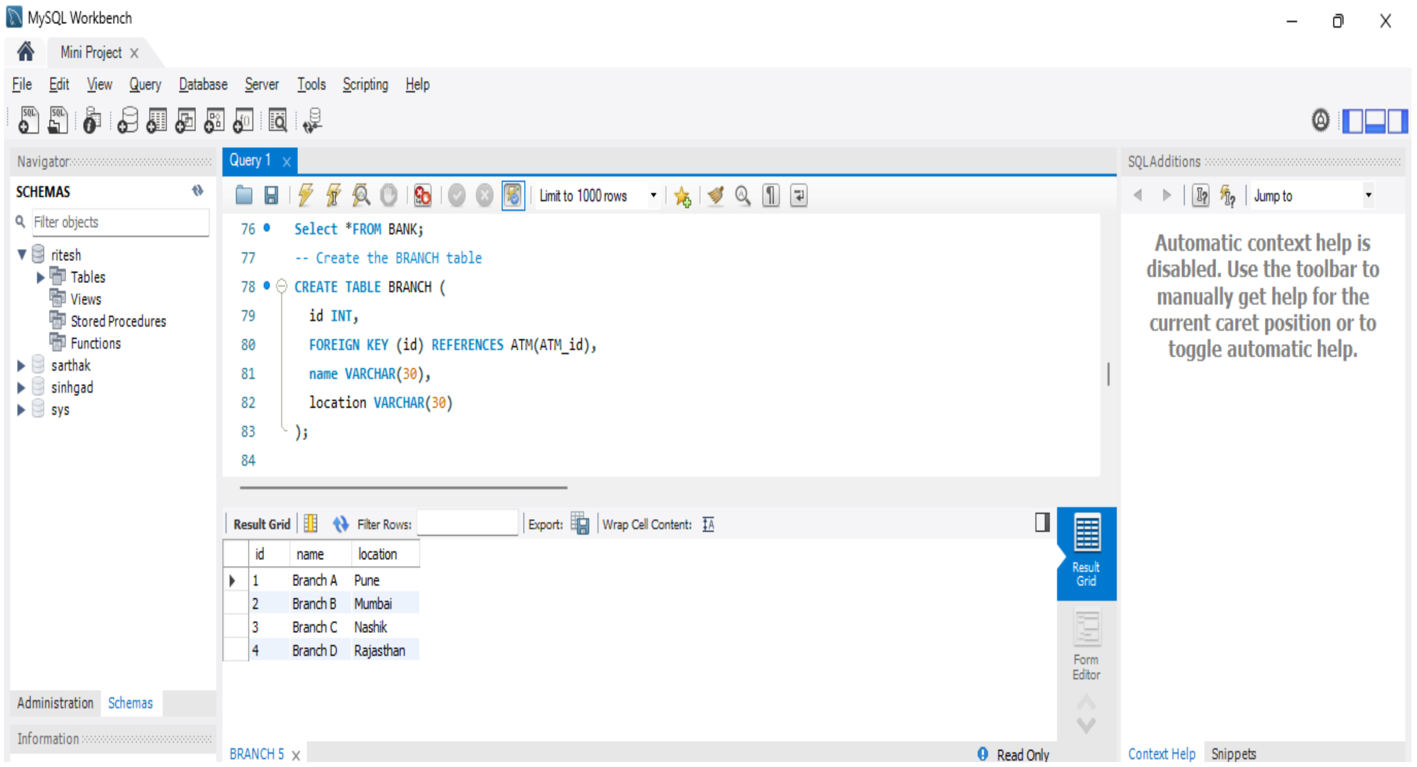
4.1.5 Bank Table



```
Select *FROM CUSTOMER;
-- Create the BANK table
CREATE TABLE BANK (
    Bank_id INT PRIMARY KEY NOT NULL,
    IFSC_code VARCHAR(30),
    Bank_name VARCHAR(30),
    Address VARCHAR(50)
);
```

Bank_id	IFSC_code	Bank_name	Address
1	XYZ123	Maharashtra bank	nashik
2	ABC456	State Bank of India	mumbai
3	XYZ789	Central bank of india	pune
4	ABC123	Paytm bank	rajasthan
NULL	NULL	NULL	NULL

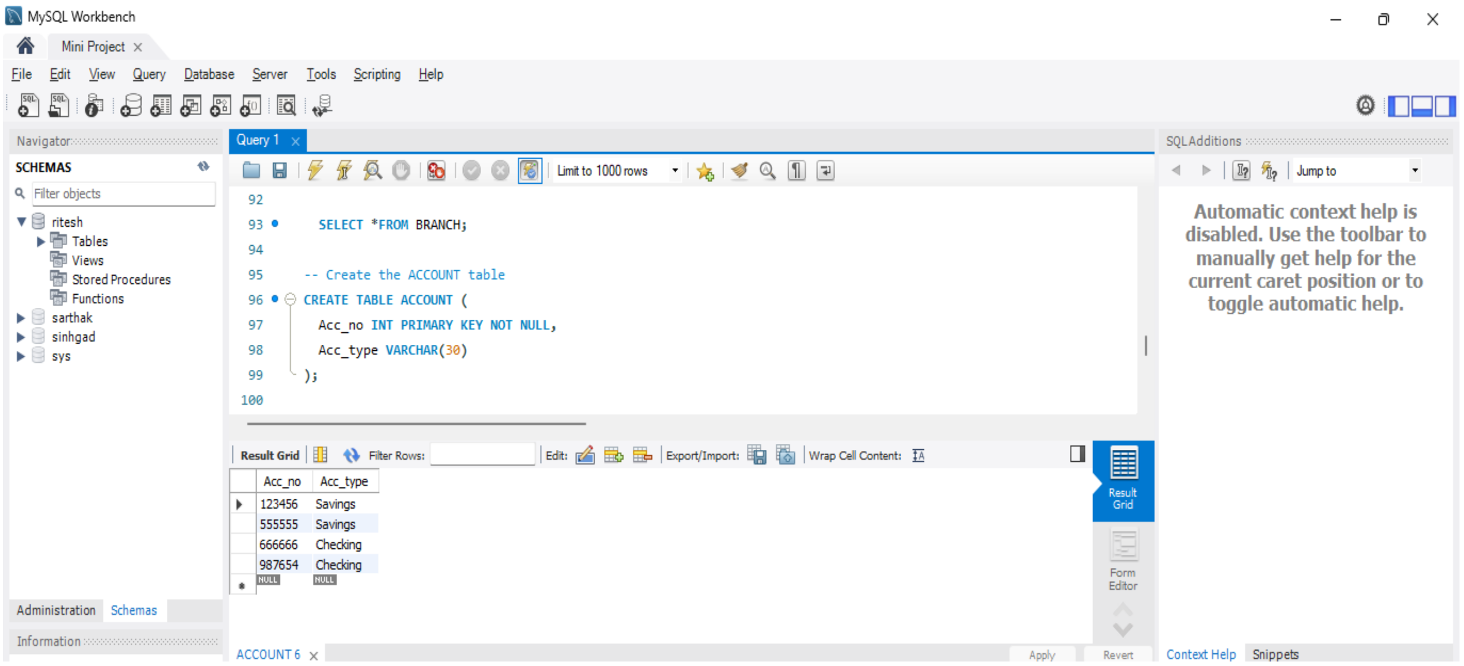
4.1.6 BRANCH Table



```
Select *FROM BANK;
-- Create the BRANCH table
CREATE TABLE BRANCH (
    id INT,
    FOREIGN KEY (id) REFERENCES ATM(ATM_id),
    name VARCHAR(30),
    location VARCHAR(30)
);
```

id	name	location
1	Branch A	Pune
2	Branch B	Mumbai
3	Branch C	Nashik
4	Branch D	Rajasthan

4.1.7 ACCOUNT Table



```
SELECT *FROM BRANCH;

-- Create the ACCOUNT table
CREATE TABLE ACCOUNT (
    Acc_no INT PRIMARY KEY NOT NULL,
    Acc_type VARCHAR(30)
);
```

Acc_no	Acc_type
123456	Savings
555555	Savings
666666	Checking
987654	Checking
NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

CHAPTER: 5
FUNCTIONAL DEPENDENCIES

Functional dependencies are used to describe the relationship between attributes in a Relational database. We can identify the following functional dependencies:

1. Functional Dependencies for the ATM table:
   - ATM_id -> ATM_name, ATM_branch, address, Bank_name

2. Functional Dependencies for the CARD table:
   - Card_no -> Card_cvv, Card_type, Card_expiry, Card_balance, Card_bankname

3. Functional Dependencies for the CUSTOMER table:
   - C_id -> First_name, Middle_name, Last_name, C_phone, C_address

4. Functional Dependencies for the BANK table:
   - Bank_id -> IFSC_code, Bank_name, Address

5. Functional Dependencies for the BRANCH table:
   - id -> name, location

6. Functional Dependencies for the ACCOUNT table:
   - Acc_no -> Acc_type

7. Functional Dependencies for the TRANSACTION table:
   - Transaction_id -> Transaction_type, Transaction_time, status

CHAPTER: 6
NORMALIZATION

The process of normalization in the Cloth Management System involves organizing the database tables and their attributes to eliminate data redundancy, improve data integrity, and simplify data management. Here is an overview of the normalization process in this system:

1. **First Normal Form (1NF):**
   Ensure each attribute contains only atomic values, avoiding multi-valued or repeating attributes.
   Separate entities into individual tables, eliminating duplicate data.
2. **Second Normal Form (2NF):**
   Identify and remove partial dependencies by ensuring each non-key attribute depends on the entire primary key.
   Split tables as necessary to achieve this dependency.
3. **Third Normal Form (3NF):**
   Eliminate transitive dependencies by ensuring that non-key attributes depend only on the primary key and not on other non-key attributes.
   Split tables further if needed to achieve this level of dependency.
4. **Higher Normal Forms:**
   Consider achieving higher normal forms, such as Boyce-Codd Normal Form (BCNF) or Fourth Normal Form (4NF), if required to eliminate more complex dependencies.

The normalization process in the Cloth Management System helps to structure the database tables efficiently, reducing data redundancy, maintaining data integrity, and facilitating effective data management. It ensures that each table serves a clear purpose, with attributes organized based on their dependencies and relationships, leading to a well-designed and optimized database structure.

To determine if the database is in 3NF or BCNF , we need to analyze the functional dependencies and the structure of the tables in the Railway Reservation System. Evaluating each table for normalization

First Normal Form (1NF):

1.ATM Table
-ATM_name
-ATM_branch
-address
-Bank_name

2. Card Table
-Card_no
-Card_cvv
-Card_type
-Card_expiry
-Card_balance
-Card_bankname

3.TRANSACTION Table
-Transaction_id
-Transaction_type
-Transaction_time
-status

4.CUSTOMER Table
- -C_id
- -First_name
- -Middle_name
- -Last_name
- -C_phone
- -C_address

5.Bank Table
- -Bank_id
- -IFSC_code
- -Bank_name
- -Address

6.Branch Table
- -id
- -name
- -location

7.ACCOUNT Table
- -Acc_no
- -Acc_type

Second Normal Form (2NF):

1.ATM Table
- -ATM_name
- -ATM_branch
- -address
- -Bank_name

2. Card Table
- -Card_no
- -Card_cvv
- -Card_type
- -Card_expiry
- -Card_balance
- -Card_bankname

3.TRANSACTION Table
- -Transaction_id
- -Transaction_type
- -Transaction_time
- -status

4.CUSTOMER Table
- -C_id
- -First_name
- -Middle_name
- -Last_name
- -C_phone
- -C_address

5.Bank Table

-Bank_id
-IFSC_code
-Bank_name
-Address

6.Branch Table
-id
-name
-location

7.ACCOUNT Table
-Acc_no
-Acc_type

Third Normal Form (3NF):

1.ATM Table
-ATM_name
-ATM_branch
-address
-Bank_name

2. Card Table
-Card_no
-Card_cvv
-Card_type
-Card_expiry
-Card_balance
-Card_bankname

3.TRANSACTION Table
-Transaction_id
-Transaction_type
-Transaction_time
-status

4.CUSTOMER Table
-C_id
-First_name
-Middle_name
-Last_name
-C_phone
-C_address

5.Bank Table
-Bank_id
-IFSC_code
-Bank_name
-Address

6.Branch Table
-id
-name
-location

7.ACCOUNT Table
-Acc_no

-Acc_type

CHAPTER: 7
CONCLUSION

7.1 Conclusion:

We have built a database for a ATM Management System using MYSQL Server. Before implementing the database, in the design phase, we have explored various features, operations of a ATM to figure out required entities, attributes and the relationship among entities to make an efficient Entity Relationship Diagram(ERD). After analysing all the requirements, we have created our ERD and then converted the ERD to relational model and normalized the tables. Using SQL Server we have created the tables for our database and inserted some sample values in the tables. Finally, we have executed sample queries on our database to check its performance to retrieve useful information accurately and speedily. We have also provided background, pseudo codes, and clear explanations of the algorithms and comparisons among them. We have converted our database tables into text files and applied our exact and approximate string matching codes on these text files to mine our database.

7.2 Future Aspects:

The Following are the future aspects of ATM Management System:

1) **Scalability:** As the number of ATMs and users increases, the ATM Database Management System can be designed to handle larger data volumes and accommodate a growing user base. Implementing techniques such as database sharding or partitioning can enhance scalability and ensure optimal performance.
2) **Advanced Analytics:** Enhancing the reporting and analytics capabilities of the system can provide deeper insights into ATM operations and customer behavior. Integrating advanced analytics tools, such as data mining algorithms or machine learning models, can uncover patterns, predict cash demand, detect anomalies, and support strategic decision making.
3) **Enhanced Security Measures:** Continuously improving the security features of the ATM Database Management System is crucial to protect sensitive customer information and prevent fraudulent activities. Implementing advanced encryption techniques, multi-factor authentication, and real-time fraud detection algorithms can enhance the system's security posture.

4) **Integration with Biometric Authentication:** Biometric authentication methods, such as fingerprint or iris scanning, can be integrated into the system to enhance security and improve user experience. By integrating with biometric devices, the system can provide secure and convenient authentication for ATM transactions.

5) **Mobile Integration:** With the rise of mobile banking, integrating the ATM Database Management System with mobile applications can offer additional convenience and functionality to users. Users can perform transactions, locate nearby ATMs, receive alerts, and access account information through their mobile devices, enhancing the overall user experience.