(SQL Database Project)

1. Introduction

A **Banking Management System** is a software solution designed to manage and automate core banking operations such as customer management, account handling, transactions, loans, and employee administration.
This project demonstrates how **SQL and relational database concepts** are used to design a structured, secure, and efficient banking database system.

2. Objectives of the Project

The main objectives of this project are:

To design a **normalized relational database** for a banking system

- To store and manage customer, account, transaction, loan, and employee data
- To implement **relationships using primary and foreign keys**
- To retrieve meaningful insights using **JOINs and aggregate queries**
- To simulate real-world banking operations using SQL

3. Technology Used

Component	Description
Database	MySQL
Language	SQL
Tool	MySQL Workbench / phpMyAdmin
Database Type	Relational Database

4. Database Design Overview

Entities Used:

1. Customers
2. Branches
3. Accounts
4. Transactions
5. Employees
6. Loans

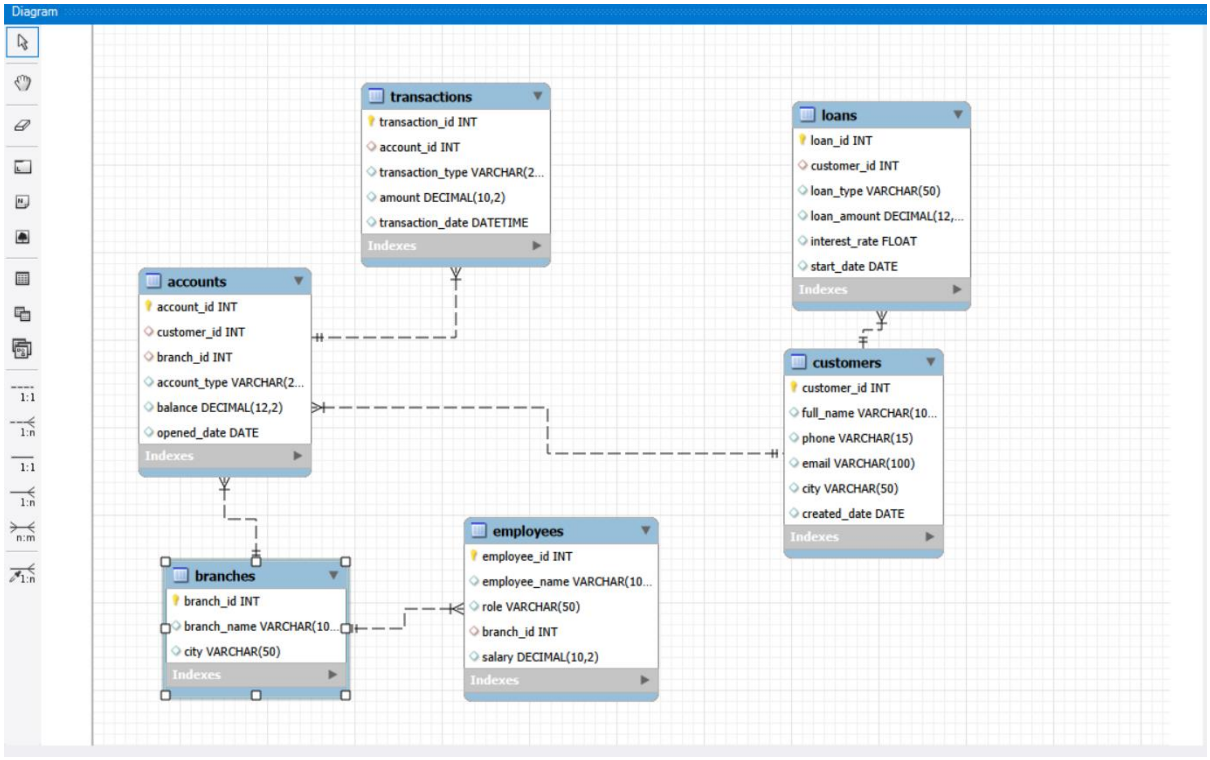
Relationships:

- One **Customer** → Many **Accounts**
- One **Account** → Many **Transactions**
- One **Branch** → Many **Accounts**
- One **Branch** → Many **Employees**
- One **Customer** → Many **Loans**

5. Database Schema (Structure)

ER Diagram

The Entity–Relationship (ER) Diagram illustrates how different entities in the banking system are connected.



Key Features:

- Customers are linked to Accounts and Loans.
- Accounts are connected to Transactions.
- Branches manage both Accounts and Employees.
- Employees are assigned to specific Branches.

👉 This diagram ensures referential integrity and models real-world banking workflows.

Database Creation

```
1
2 • CREATE DATABASE Banking_Management_System;
```

Customers Table

Stores personal and contact information of customers.

Input:

```
CREATE TABLE Customers (customer_id INT PRIMARY KEY AUTO_INCREMENT,
full_name VARCHAR(100),phone VARCHAR(15),email VARCHAR(100),
city VARCHAR(50),created_date DATE);
```

Output:

```
10 •  desc Customers;
11
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	auto_increment
full_name	varchar(100)	YES		NULL	
phone	varchar(15)	YES		NULL	
email	varchar(100)	YES		NULL	
city	varchar(50)	YES		NULL	
created_date	date	YES		NULL	

Branches Table

Stores branch details of the bank.

```
CREATE TABLE Branches (branch_id INT PRIMARY KEY AUTO_INCREMENT,
branch_name VARCHAR(100),city VARCHAR(50));
```

Accounts Table

Stores customer account details.

```
CREATE TABLE Accounts (account_id INT PRIMARY KEY AUTO_INCREMENT,customer_id INT,
branch_id INT,account_type VARCHAR(20),balance DECIMAL(12,2),opened_date DATE,
FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
FOREIGN KEY (branch_id) REFERENCES Branches(branch_id));
```

Transactions Table

Stores all deposit and withdrawal transactions.

```
CREATE TABLE Transactions (transaction_id INT PRIMARY KEY AUTO_INCREMENT,account_id INT,
transaction_type VARCHAR(20),amount DECIMAL(10,2),transaction_date DATETIME,
FOREIGN KEY (account_id) REFERENCES Accounts(account_id));
```

Employees Table

Stores employee information for each branch.

```
CREATE TABLE Employees (employee_id INT PRIMARY KEY AUTO_INCREMENT,employee_name VARCHAR(100),
role VARCHAR(50),branch_id INT,salary DECIMAL(10,2),
FOREIGN KEY (branch_id) REFERENCES Branches(branch_id));
```

Loans Table

Stores loan details issued to customers.

```
CREATE TABLE Loans (loan_id INT PRIMARY KEY AUTO_INCREMENT,customer_id INT,loan_type VARCHAR(50),
loan_amount DECIMAL(12,2),interest_rate FLOAT,start_date DATE,
FOREIGN KEY (customer_id) REFERENCES Customers(customer_id));
```

6. Data Population

- **5 Branches**
- **20 Customers**
- **20 Accounts**
- **20 Transactions**
- **20 Employees**
- **20 Loans**

✓ Data represents **realistic banking operations**
✓ Ensures **referential integrity**

6.1 Branches (5 Records)

The **Branches** table stores information about different bank branches operating in major Indian cities

Input:

```
INSERT INTO Branches (branch_name, city) VALUES
('Main Branch','Delhi'),
('City Branch','Mumbai'),
('Central Branch','Bangalore'),
('North Branch','Chandigarh'),
('South Branch','Chennai');
```

Output:

```
41 •  select * from branches;
42
```

Result Grid | Filter Rows: | Edit: | Export/Import:

branch_id	branch_name	city
1	Main Branch	Delhi
2	City Branch	Mumbai
3	Central Branch	Bangalore
4	North Branch	Chandigarh
5	South Branch	Chennai
NULL	NULL	NULL

Explanation:

- Five branches are created to represent geographical distribution.
- Each branch is uniquely identified by `branch_id`.
- These branches are later linked with **accounts and employees**.

6.2 Customers (20 Records)

The **Customers** table stores personal and contact details of bank customers.

Input:

```
INSERT INTO Customers (full_name, phone, email, city, created_date) VALUES
('Ravi Kumar','9876543210','ravi@gmail.com','Delhi','2024-01-01'),
('Anita Sharma','9123456780','anita@gmail.com','Mumbai','2024-01-02'),
('Suresh Patel','9988776655','suresh@gmail.com','Ahmedabad','2024-01-03'),
('Neha Verma','9876123450','neha@gmail.com','Jaipur','2024-01-04'),
('Amit Singh','9090909090','amit@gmail.com','Lucknow','2024-01-05'),
('Pooja Mehta','9012345678','pooja@gmail.com','Pune','2024-01-06'),
('Rahul Jain','9898989898','rahul@gmail.com','Indore','2024-01-07'),
('Kavita Nair','9765432109','kavita@gmail.com','Kochi','2024-01-08'),
('Manoj Yadav','9812345670','manoj@gmail.com','Patna','2024-01-09'),
('Sunita Roy','9944556677','sunita@gmail.com','Kolkata','2024-01-10'),
('Vikas Malhotra','8888777766','vikas@gmail.com','Chandigarh','2024-01-11'),
('Rina Das','9777665544','rina@gmail.com','Guwahati','2024-01-12'),
('Arjun Reddy','9000111222','arjun@gmail.com','Hyderabad','2024-01-13'),
('Sneha Iyer','9555666777','sneha@gmail.com','Chennai','2024-01-14'),
('Kunal Shah','9222333444','kunal@gmail.com','Surat','2024-01-15'),
('Deepak Joshi','9666554433','deepak@gmail.com','Udaipur','2024-01-16'),
('Priya Banerjee','9333444555','priya@gmail.com','Howrah','2024-01-17'),
('Nitin Kulkarni','9444555666','nitin@gmail.com','Nagpur','2024-01-18'),
('Ayesha Khan','9888999000','ayesha@gmail.com','Bhopal','2024-01-19'),
('Rohit Mishra','9777888999','rohit@gmail.com','Varanasi','2024-01-20');
```

Output:

```
65 •    select * from Customers;
```

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

customer_id	full_name	phone	email	city	created_date
1	Ravi Kumar	9876543210	ravi@gmail.com	Delhi	2024-01-01
2	Anita Sharma	9123456780	anita@gmail.com	Mumbai	2024-01-02
3	Suresh Patel	9988776655	suresh@gmail.com	Ahmedabad	2024-01-03
4	Neha Verma	9876123450	neha@gmail.com	Jaipur	2024-01-04
5	Amit Singh	9090909090	amit@gmail.com	Lucknow	2024-01-05
6	Pooja Mehta	9012345678	pooja@gmail.com	Pune	2024-01-06
7	Rahul Jain	9898989898	rahul@gmail.com	Indore	2024-01-07
8	Kavita Nair	9765432109	kavita@gmail.com	Kochi	2024-01-08
9	Manoj Yadav	9812345670	manoj@gmail.com	Patna	2024-01-09
10	Sunita Roy	9944556677	sunita@gmail.com	Kolkata	2024-01-10
11	Vikas Malhotra	8888777766	vikas@gmail.com	Chandigarh	2024-01-11
12	Rina Das	9777665544	rina@gmail.com	Guwahati	2024-01-12
13	Arjun Reddy	9000111222	arjun@gmail.com	Hyderabad	2024-01-13
14	Sneha Iyer	9555666777	sneha@gmail.com	Chennai	2024-01-14
15	Kunal Shah	9222333444	kunal@gmail.com	Surat	2024-01-15
16	Deepak Joshi	9666554433	deepak@gmail.com	Udaipur	2024-01-16
17	Priya Banerjee	9333444555	priya@gmail.com	Howrah	2024-01-17
18	Nitin Kulkarni	9444555666	nitin@gmail.com	Nagpur	2024-01-18
19	Ayesha Khan	9888999000	ayesha@gmail.com	Bhopal	2024-01-19
20	Rohit Mishra	9777888999	rohit@gmail.com	Varanasi	2024-01-20
NULL	NULL	NULL	NULL	NULL	NULL

Explanation:

- A total of **20 customers** are inserted.
- Each customer has a **unique customer_id** generated automatically.
- The data includes customers from **different cities**, ensuring diversity.
- `created_date` shows when the customer joined the bank.

6.3 Accounts (20 Records)

The **Accounts** table stores details about customer bank accounts.

Input:

```
INSERT INTO Accounts (customer_id, branch_id, account_type, balance, opened_date) VALUES
(1,1,'Savings',50000,'2024-01-10'),
(2,2,'Current',120000,'2024-01-11'),
(3,3,'Savings',45000,'2024-01-12'),
(4,4,'Savings',38000,'2024-01-13'),
(5,5,'Current',200000,'2024-01-14'),
(6,1,'Savings',60000,'2024-01-15'),
(7,2,'Current',95000,'2024-01-16'),
(8,3,'Savings',72000,'2024-01-17'),
(9,4,'Savings',15000,'2024-01-18'),
(10,5,'Current',300000,'2024-01-19'),
(11,1,'Savings',40000,'2024-01-20'),
(12,2,'Current',180000,'2024-01-21'),
(13,3,'Savings',52000,'2024-01-22'),
(14,4,'Savings',68000,'2024-01-23'),
(15,5,'Current',250000,'2024-01-24'),
(16,1,'Savings',33000,'2024-01-25'),
(17,2,'Savings',47000,'2024-01-26'),
(18,3,'Current',90000,'2024-01-27'),
(19,4,'Savings',56000,'2024-01-28'),
(20,5,'Current',175000,'2024-01-29');
```

Explanation:

- Each customer is assigned **one account**.
- Accounts are linked to both **Customers** and **Branches** using foreign keys.
- Two account types are used: **Savings** and **Current**.
- Balances vary to represent different customer profiles.

6.4 Transactions (20 Records)

The **Transactions** table stores deposit and withdrawal activities.

Input:

```
INSERT INTO Transactions (account_id, transaction_type, amount, transaction_date) VALUES
(1,'Deposit',10000,NOW()),
(2,'Withdrawal',5000,NOW()),
(3,'Deposit',8000,NOW()),
(4,'Withdrawal',3000,NOW()),
(5,'Deposit',25000,NOW()),
(6,'Deposit',12000,NOW()),
(7,'Withdrawal',7000,NOW()),
(8,'Deposit',9000,NOW()),
(9,'Withdrawal',2000,NOW()),
(10,'Deposit',50000,NOW()),
(11,'Deposit',6000,NOW()),
(12,'Withdrawal',10000,NOW()),
(13,'Deposit',7000,NOW()),
(14,'Withdrawal',4000,NOW()),
(15,'Deposit',30000,NOW()),
(16,'Deposit',5500,NOW()),
(17,'Withdrawal',3500,NOW()),
(18,'Deposit',15000,NOW()),
(19,'Withdrawal',4500,NOW()),
(20,'Deposit',22000,NOW());
```

Explanation:

- Each account has at least **one transaction**.
- Two transaction types are used: **Deposit** and **Withdrawal**.
- `NOW()` is used to store the **current date and time**, simulating real-time banking.
- Transactions help track customer activity and balances.

6.5 Employees (20 Records)

The **Employees** table stores staff details working across branches.

Input:

```sql
INSERT INTO Employees (employee_name, role, branch_id, salary) VALUES
('Suresh Verma','Manager',1,85000),
('Neha Singh','Clerk',1,35000),
('Amit Kapoor','Cashier',2,30000),
('Ritu Malhotra','Manager',2,88000),
('Vikas Rao','Clerk',3,36000),
('Anjali Mehta','Cashier',3,32000),
('Rohit Sharma','Manager',4,90000),
('Poonam Joshi','Clerk',4,34000),
('Manish Gupta','Cashier',5,31000),
('Kavita Nair','Manager',5,87000),
('Deepak Yadav','Clerk',1,33000),
('Shalini Roy','Cashier',2,30000),
('Nitin Kulkarni','Manager',3,89000),
('Priya Saxena','Clerk',4,35000),
('Arjun Patel','Cashier',5,32000),
('Sneha Iyer','Clerk',1,34000),
('Rahul Jain','Cashier',2,31000),
('Sunita Das','Clerk',3,36000),
('Vivek Mishra','Manager',4,92000),
('Ayesha Khan','Clerk',5,35500);
```

Explanation:

- A total of **20 employees** are distributed across all branches.
- Roles include **Manager, Clerk, and Cashier**.
- Each employee is assigned to one branch using `branch_id`.
- Salary varies based on job role.

6.6 Loans (20 Records)

The **Loans** table stores loan details issued to customers.

Input:

```sql
INSERT INTO Loans (customer_id, loan_type, loan_amount, interest_rate, start_date) VALUES
(1,'Home Loan',2500000,8.5,'2024-01-15'),
(2,'Car Loan',800000,9.2,'2024-01-18'),
(3,'Personal Loan',300000,12.5,'2024-01-20'),
(4,'Education Loan',600000,7.8,'2024-01-22'),
(5,'Home Loan',3200000,8.4,'2024-01-25'),
(6,'Car Loan',650000,9.0,'2024-01-27'),
(7,'Personal Loan',400000,12.0,'2024-01-29'),
(8,'Education Loan',550000,7.5,'2024-02-01'),
(9,'Home Loan',2100000,8.6,'2024-02-03'),
(10,'Car Loan',900000,9.1,'2024-02-05'),
(11,'Personal Loan',350000,12.2,'2024-02-07'),
(12,'Education Loan',700000,7.9,'2024-02-09'),
(13,'Home Loan',2800000,8.3,'2024-02-11'),
(14,'Car Loan',750000,9.4,'2024-02-13'),
(15,'Personal Loan',450000,11.8,'2024-02-15'),
(16,'Education Loan',500000,7.6,'2024-02-17'),
(17,'Home Loan',2300000,8.7,'2024-02-19'),
(18,'Car Loan',820000,9.3,'2024-02-21'),
(19,'Personal Loan',380000,12.1,'2024-02-23'),
(20,'Education Loan',650000,7.7,'2024-02-25');
```

Explanation:

- Each customer has been assigned **one loan**.
- Multiple loan types are included: **Home, Car, Personal, and Education loans**.
- Interest rates differ according to loan type.
- Loan data helps analyze bank credit exposure.

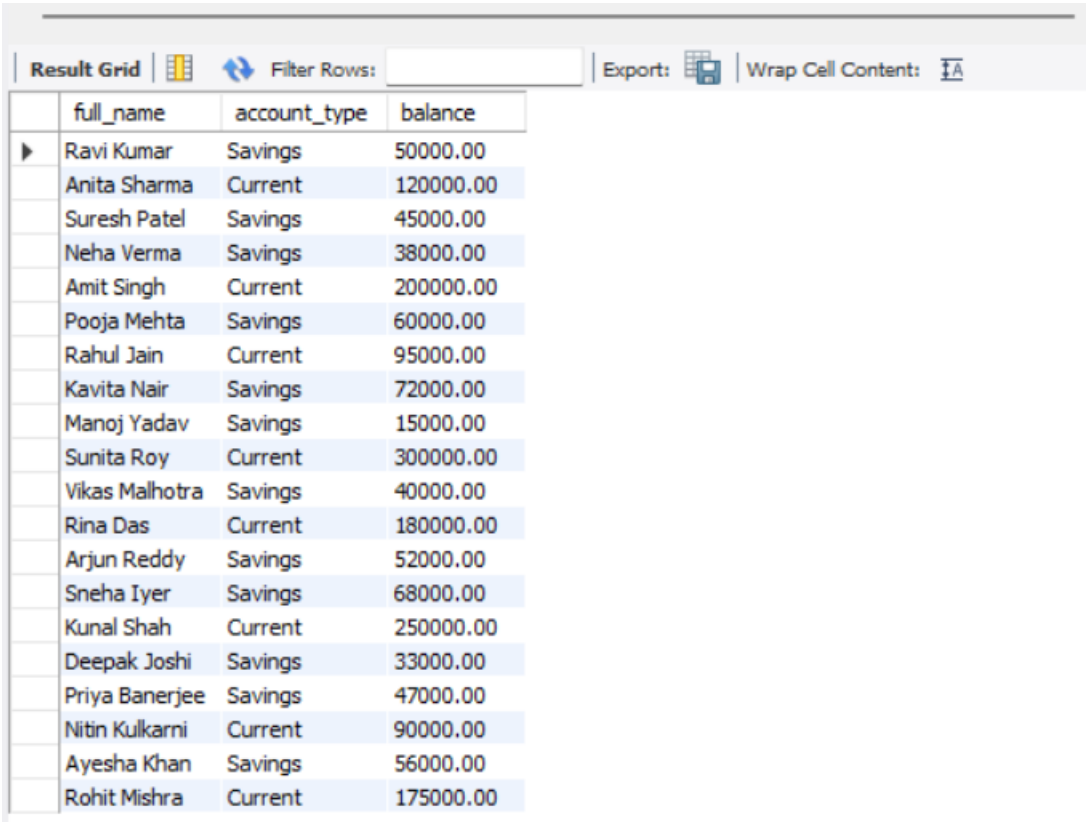
7. SQL Queries & Insights

1. Customers and Their Account Details

Input:

```
SELECT c.full_name, a.account_type, a.balance
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id;
```

Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

full_name	account_type	balance
Ravi Kumar	Savings	50000.00
Anita Sharma	Current	120000.00
Suresh Patel	Savings	45000.00
Neha Verma	Savings	38000.00
Amit Singh	Current	200000.00
Pooja Mehta	Savings	60000.00
Rahul Jain	Current	95000.00
Kavita Nair	Savings	72000.00
Manoj Yadav	Savings	15000.00
Sunita Roy	Current	300000.00
Vikas Malhotra	Savings	40000.00
Rina Das	Current	180000.00
Arjun Reddy	Savings	52000.00
Sneha Iyer	Savings	68000.00
Kunal Shah	Current	250000.00
Deepak Joshi	Savings	33000.00
Priya Banerjee	Savings	47000.00
Nitin Kulkarni	Current	90000.00
Ayesha Khan	Savings	56000.00
Rohit Mishra	Current	175000.00

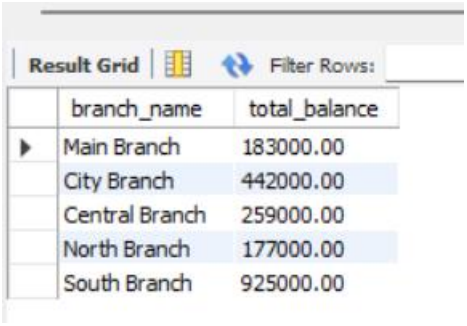
Insight: Displays which customer owns which account and their current balance.

2. Total Balance in Each Branch

Input:

```sql
SELECT b.branch_name, SUM(a.balance) AS total_balance
FROM Branches b
JOIN Accounts a ON b.branch_id = a.branch_id
GROUP BY b.branch_name;
```

Output:



branch_name	total_balance
Main Branch	183000.00
City Branch	442000.00
Central Branch	259000.00
North Branch	177000.00
South Branch	925000.00

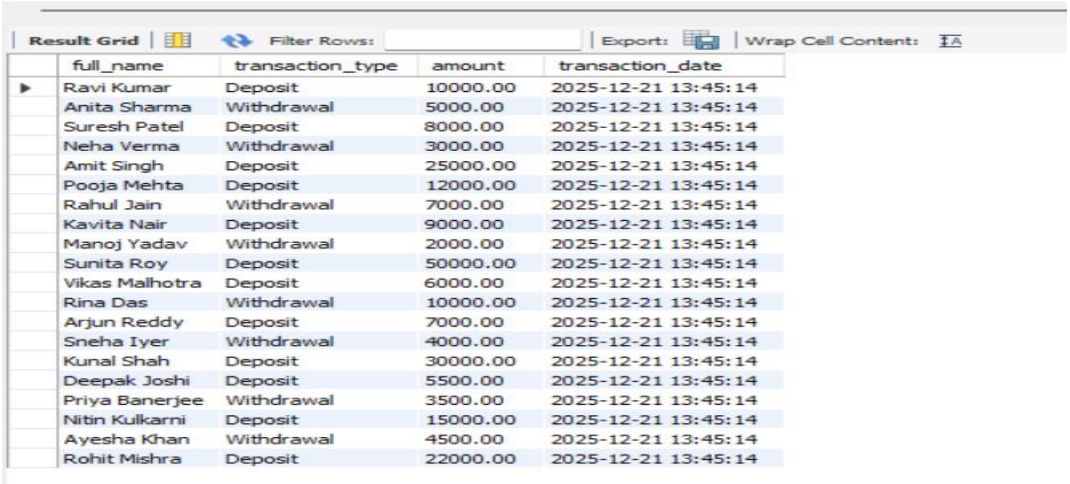
Insight: Identifies branches with the highest deposits.

3. Complete Transaction History

Input:

```sql
SELECT c.full_name, t.transaction_type, t.amount, t.transaction_date
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
JOIN Transactions t ON a.account_id = t.account_id;
```

Output:



full_name	transaction_type	amount	transaction_date
Ravi Kumar	Deposit	10000.00	2025-12-21 13:45:14
Anita Sharma	Withdrawal	5000.00	2025-12-21 13:45:14
Suresh Patel	Deposit	8000.00	2025-12-21 13:45:14
Neha Verma	Withdrawal	3000.00	2025-12-21 13:45:14
Amit Singh	Deposit	25000.00	2025-12-21 13:45:14
Pooja Mehta	Deposit	12000.00	2025-12-21 13:45:14
Rahul Jain	Withdrawal	7000.00	2025-12-21 13:45:14
Kavita Nair	Deposit	9000.00	2025-12-21 13:45:14
Manoj Yadav	Withdrawal	2000.00	2025-12-21 13:45:14
Sunita Roy	Deposit	50000.00	2025-12-21 13:45:14
Vikas Malhotra	Deposit	6000.00	2025-12-21 13:45:14
Rina Das	Withdrawal	10000.00	2025-12-21 13:45:14
Arjun Reddy	Deposit	7000.00	2025-12-21 13:45:14
Sneha Iyer	Withdrawal	4000.00	2025-12-21 13:45:14
Kunal Shah	Deposit	30000.00	2025-12-21 13:45:14
Deepak Joshi	Deposit	5500.00	2025-12-21 13:45:14
Priya Banerjee	Withdrawal	3500.00	2025-12-21 13:45:14
Nitin Kulkarni	Deposit	15000.00	2025-12-21 13:45:14
Ayesha Khan	Withdrawal	4500.00	2025-12-21 13:45:14
Rohit Mishra	Deposit	22000.00	2025-12-21 13:45:14

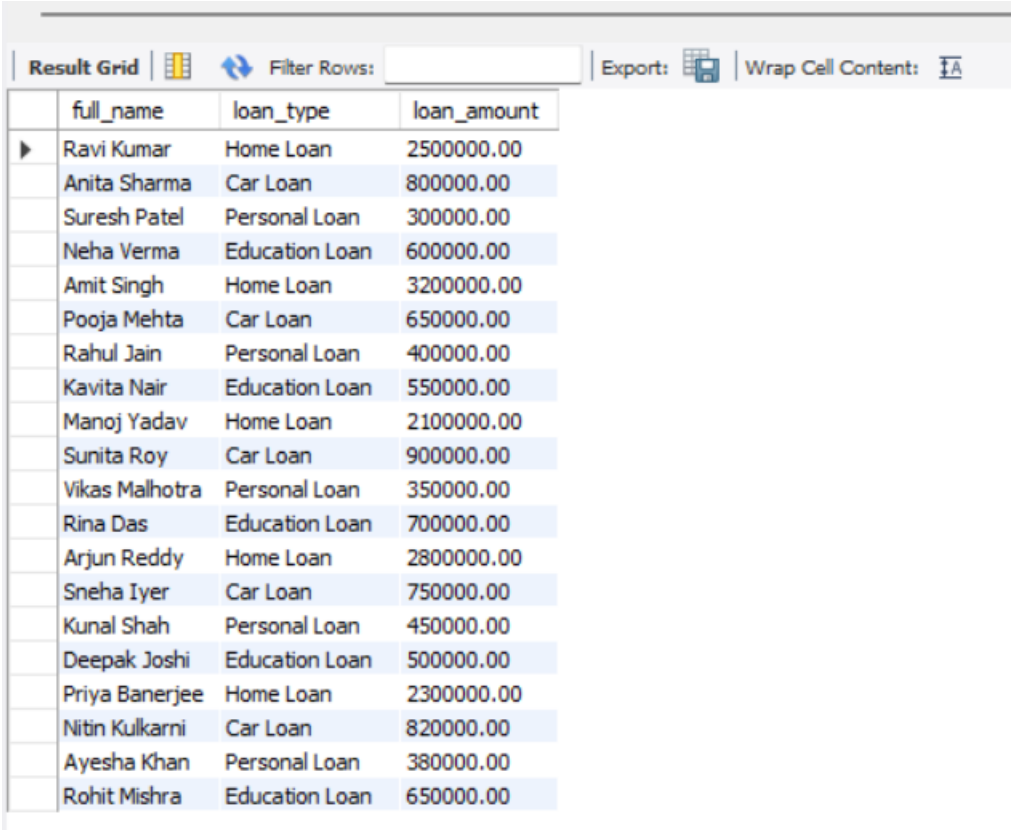
Insight: Useful for auditing and fraud monitoring.

4. Customers Who Have Taken Loans

Input:

```
SELECT c.full_name, l.loan_type, l.loan_amount
FROM Customers c
JOIN Loans l ON c.customer_id = l.customer_id;
```

Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

full_name	loan_type	loan_amount
Ravi Kumar	Home Loan	2500000.00
Anita Sharma	Car Loan	800000.00
Suresh Patel	Personal Loan	300000.00
Neha Verma	Education Loan	600000.00
Amit Singh	Home Loan	3200000.00
Pooja Mehta	Car Loan	650000.00
Rahul Jain	Personal Loan	400000.00
Kavita Nair	Education Loan	550000.00
Manoj Yadav	Home Loan	2100000.00
Sunita Roy	Car Loan	900000.00
Vikas Malhotra	Personal Loan	350000.00
Rina Das	Education Loan	700000.00
Arjun Reddy	Home Loan	2800000.00
Sneha Iyer	Car Loan	750000.00
Kunal Shah	Personal Loan	450000.00
Deepak Joshi	Education Loan	500000.00
Priya Banerjee	Home Loan	2300000.00
Nitin Kulkarni	Car Loan	820000.00
Ayesha Khan	Personal Loan	380000.00
Rohit Mishra	Education Loan	650000.00

Insight: Shows loan distribution among customers.

5. Customers Who Made Deposits

Input:

```
SELECT DISTINCT c.full_name
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
JOIN Transactions t ON a.account_id = t.account_id
WHERE t.transaction_type = 'Deposit';
```

Output:

full_name
Ravi Kumar
Suresh Patel
Amit Singh
Pooja Mehta
Kavita Nair
Sunita Roy
Vikas Malhotra
Arjun Reddy
Kunal Shah
Deepak Joshi
Nitin Kulkarni
Rohit Mishra

Insight: Identifies active customers.

6. Employees Working in Each Branch

Input:

```
SELECT b.branch_name, e.employee_name, e.role
FROM Branches b
JOIN Employees e ON b.branch_id = e.branch_id;
```

Output:

branch_name	employee_name	role
Main Branch	Suresh Verma	Manager
Main Branch	Neha Singh	Clerk
Main Branch	Deepak Yadav	Clerk
Main Branch	Sneha Iyer	Clerk
City Branch	Amit Kapoor	Cashier
City Branch	Ritu Malhotra	Manager
City Branch	Shalini Roy	Cashier
City Branch	Rahul Jain	Cashier
Central Branch	Vikas Rao	Clerk
Central Branch	Anjali Mehta	Cashier
Central Branch	Nitin Kulkarni	Manager
Central Branch	Sunita Das	Clerk
North Branch	Rohit Sharma	Manager
North Branch	Poonam Joshi	Clerk
North Branch	Priya Saxena	Clerk
North Branch	Vivek Mishra	Manager
South Branch	Manish Gupta	Cashier
South Branch	Kavita Nair	Manager
South Branch	Arjun Patel	Cashier
South Branch	Ayesha Khan	Clerk

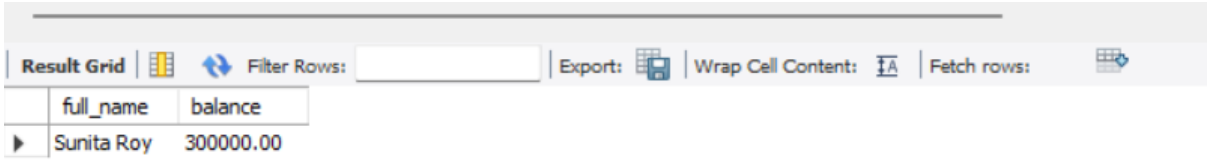
Insight: Helps in workforce planning.

7. Highest Account Balance

Input:

```
SELECT c.full_name, a.balance
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
ORDER BY a.balance DESC
LIMIT 1;
```

Output:



full_name	balance
Sunita Roy	300000.00

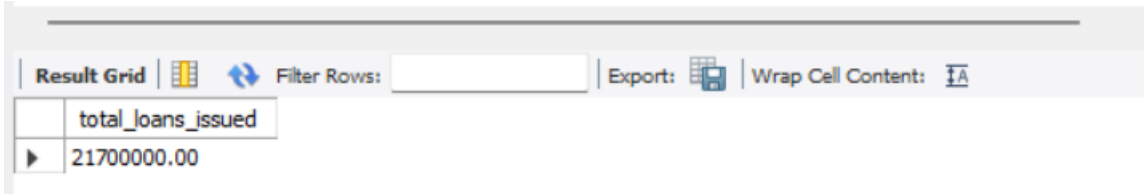
Insight: Identifies premium customers.

8. Total Loan Amount Issued

Input:

```
SELECT SUM(loan_amount) AS total_loans_issued
FROM Loans;
```

Output:



total_loans_issued
21700000.00

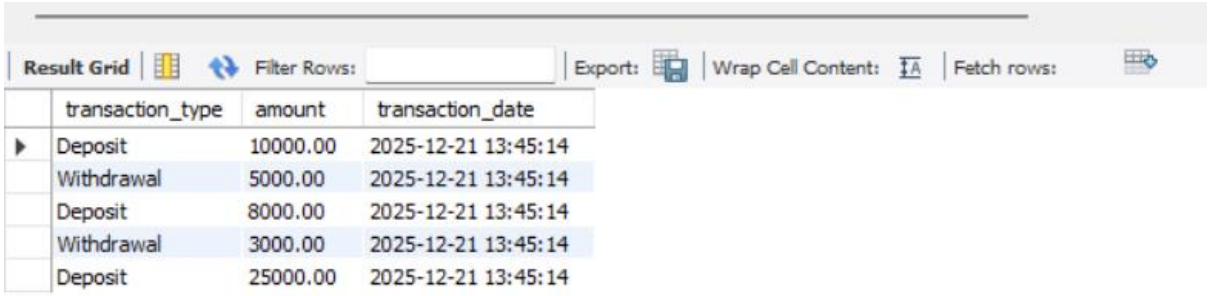
Insight: Shows total credit exposure of the bank.

9. Recent Transactions

Input:

```
SELECT transaction_type, amount, transaction_date
FROM Transactions
ORDER BY transaction_date DESC
LIMIT 5;
```

Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

transaction_type	amount	transaction_date
Deposit	10000.00	2025-12-21 13:45:14
Withdrawal	5000.00	2025-12-21 13:45:14
Deposit	8000.00	2025-12-21 13:45:14
Withdrawal	3000.00	2025-12-21 13:45:14
Deposit	25000.00	2025-12-21 13:45:14

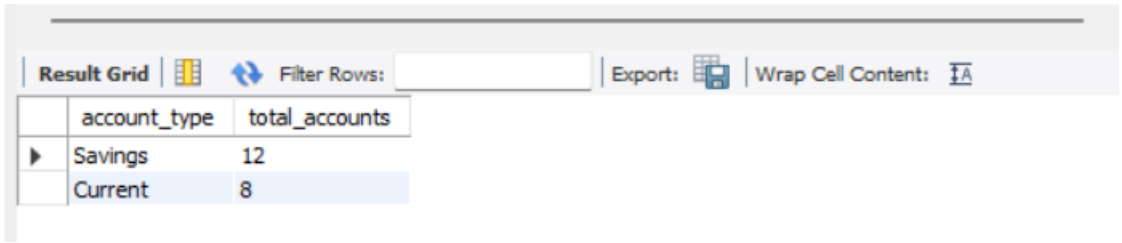
Insight: Useful for dashboards and real-time monitoring.

10. Number of Accounts by Type

Input:

```
SELECT account_type, COUNT(*) AS total_accounts
FROM Accounts
GROUP BY account_type;
```

Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

account_type	total_accounts
Savings	12
Current	8

Insight: Analyzes customer preference for Savings vs Current accounts.

8. Conclusion

This **Banking Management System** project successfully demonstrates the use of **SQL, relational database design, and business logic** to manage core banking operations. It provides a strong foundation for understanding **real-world database applications in banking systems**.