BANK MANAGEMENT SYSTEM
USING MySQL

---

1. OBJECTIVE, TOOLS AND TECHNOLOGY, KEY FEATURES

The objective of this project is to design a relational database system that simulates the core functionalities of a bank. It helps manage customer information, account details, branch data, loans, and transactions efficiently. The system ensures data integrity, supports real-time banking operations like deposits and withdrawals, and enables effective data retrieval using SQL queries.

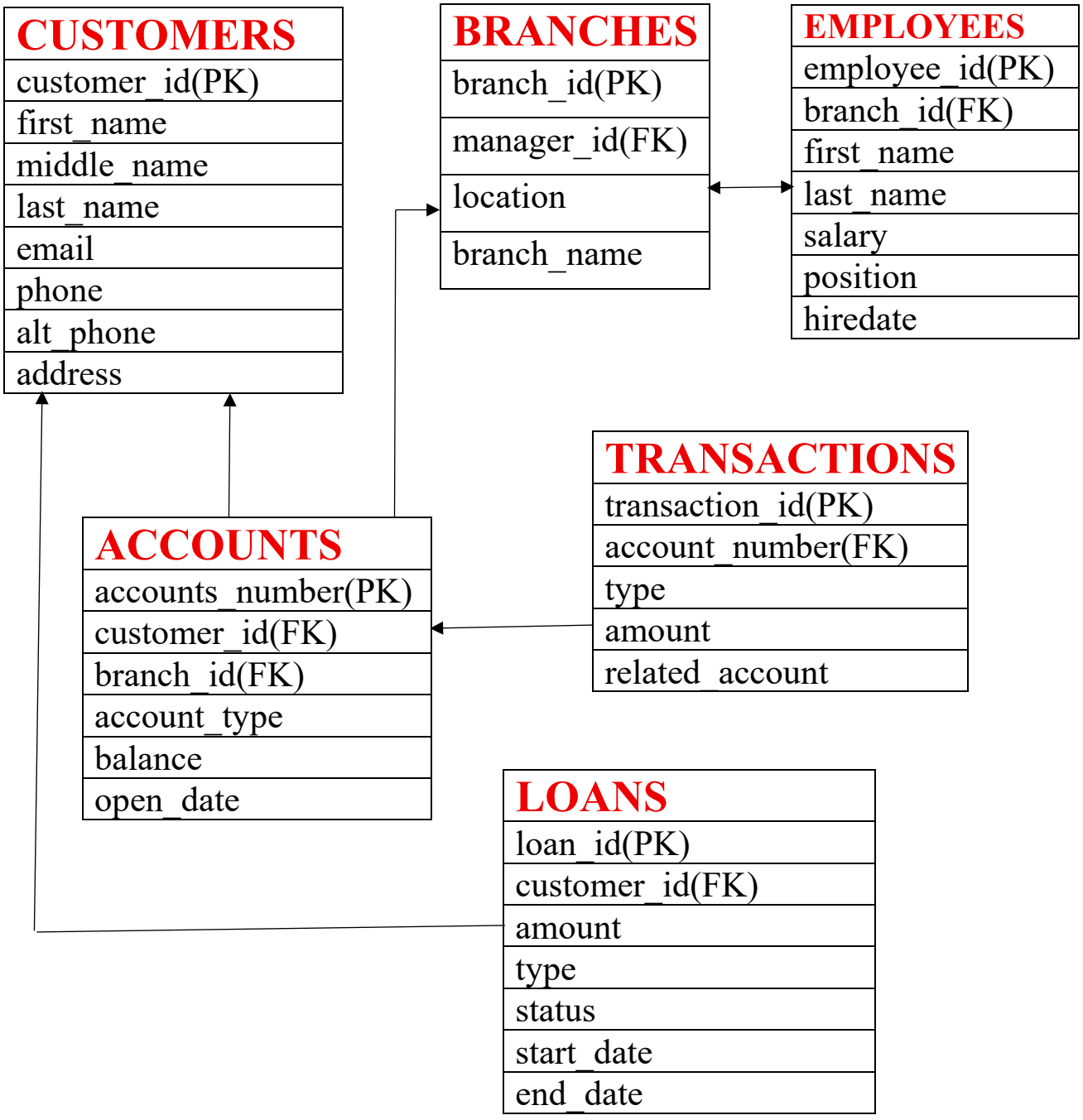
TOOLS AND TECHNOLOGIES USED:

Category	Tools / Technologies
Database	MySQL
Language	SQL (Structured Query Language)
Environment/Editor	MySQL Workbench

KEY FEATURES:

Relational Database Design: Well-structured schema using normalization principles.
Data Integrity: Maintained with NOT NULL, AUTO_INCREMENT, ENUM, and FOREIGN KEY constraints.
Join Operations: Can perform complex queries like fetching customer loan status, account balances, etc.
Handling Nulls: Accounts without branches, loans without end dates, etc., are properly handled.
Transaction Table: Demonstrates real-time banking operations (Deposit, Withdrawal, Transfer).

2. ER DIAGRAM


CUSTOMERS
customer_id(PK)
first_name
middle_name
last_name
email
phone
alt_phone
address
BRANCHES
branch_id(PK)
manager_id(FK)
location
branch_name
EMPLOYEES
employee_id(PK)
branch_id(FK)
first_name
last_name
salary
position
hiredate
TRANSACTIONS
transaction_id(PK)
account_number(FK)
type
amount
related_account
ACCOUNTS
accounts_number(PK)
customer_id(FK)
branch_id(FK)
account_type
balance
open_date
LOANS
loan_id(PK)
customer_id(FK)
amount
type
status
start_date
end_date

3. CREATING TABLES AND INSERTING VALUES

TABLES TO BE CREATED:

- ✓ Customers Table
- ✓ Branches Table
- ✓ Accounts Table
- ✓ Loans Table
- ✓ Transaction Table
- ✓ Employees Table

CUSTOMERS TABLE:

QUERY:

```
CREATE TABLE `customers` (
  `customer_id` int NOT NULL AUTO_INCREMENT,
  `first_name` varchar(50) NOT NULL,
  `middle_name` varchar(50) DEFAULT NULL,
  `last_name` varchar(50) NOT NULL,
  `email` varchar(100) DEFAULT NULL,
  `phone` varchar(15) NOT NULL,
  `alt_phone` varchar(15) DEFAULT NULL,
  `address` text,
  PRIMARY KEY (`customer_id`),
  UNIQUE KEY `email` (`email`)
)ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
INSERT INTO `customers` VALUES (1,'Rahul','Kumar','Sharma','rahul@example.com','9876543210',NULL,'Mumbai'),(2,'Priya',NULL,'Patel','priya@example.com','9876543211','9876543212','Delhi'),(3,'Amit','Ramesh','Verma','amit@example.com','9876543213',NULL,'Bangalore'),(4,'Sneha',NULL,'Singh','sneha@example.com','9876543214',NULL,'Hyderabad'),(5,'Vikram','Raj','Gupta','vikram@example.com','9876543215','9876543216','Chennai'),(6,'Anjali','Priya','Mehta','anjali@example.com','9876543217',NULL,'Kolkata'),(7,'Ravi',NULL,'Joshi','ravi@example.com','9876543218','9876
```

543219','Pune'),(8,'Neha','Sunil','Malhotra','neha@example.com','9876543220',NULL,'Ahmedabad'),(9,'Sanjay',NULL,'Reddy','sanjay@example.com','9876543221',NULL,'Jaipur'),(10,'Pooja','Anil','Desai','pooja@example.com','9876543222','9876543223','Lucknow'),(11,'Arun',NULL,'Iyer','arun@example.com','9876543224',NULL,'Chandigarh'),(12,'Kavita','Vijay','Rao','kavita@example.com','9876543225','9876543226','Bhopal'),(13,'Rajesh',NULL,'Thakur','rajesh@example.com','9876543227',NULL,'Surat'),(14,'Swati','Mohan','Chopra','swati@example.com','9876543228','9876543229','Nagpur'),(15,'Alok',NULL,'Bansal','alok@example.com','9876543230',NULL,'Indore'),(16,'Meera','Prakash','Srivastava','meera@example.com','9876543231','9876543232','Patna'),(17,'Vivek',NULL,'Dubey','vivek@example.com','9876543233',NULL,'Coimbatore'),(18,'Anita','Suresh','Shukla','anita@example.com','9876543234','9876543235','Visakhapatnam'),(19,'Rohan',NULL,'Gandhi','rohan@example.com','9876543236',NULL,'Thane'),(20,'Divya','Rajat','Sinha','divya@example.com','9876543237','9876543238','Agra');

TABLE:

To view table query:

select * from customers;

BRANCHES TABLE:

QUERY:

```
CREATE TABLE `branches` (
  `branch_id` int NOT NULL AUTO_INCREMENT,
  `branch_name` varchar(100) NOT NULL,
  `location` varchar(100) NOT NULL,
  PRIMARY KEY (`branch_id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

INSERT INTO `branches` VALUES (1,'Kotak Mumbai Main','Mumbai'),(2,'Kotak Delhi Central','Delhi'),(3,'Kotak Bangalore Tech','Bangalore'),(4,'Kotak Hyderabad City','Hyderabad'),(5,'Kotak Chennai Coast','Chennai'),(6,'Kotak Kolkata East','Kolkata'),(7,'Kotak Pune West','Pune'),(8,'Kotak Ahmedabad North','Ahmedabad'),(9,'Kotak Jaipur Heritage','Jaipur'),(10,'Kotak Lucknow Central','Lucknow'),(11,'Kotak Chandigarh Sector 17','Chandigarh'),(12,'Kotak Bhopal Lakeview','Bhopal'),(13,'Kotak

Surat Diamond','Surat'),(14,'Kotak Nagpur Orange','Nagpur'),(15,'Kotak Indore Trade','Indore'),(16,'Kotak Patna Historic','Patna'),(17,'Kotak Coimbatore Textile','Coimbatore'),(18,'Kotak Vizag Port','Visakhapatnam'),(19,'Kotak Thane Suburban','Thane'),(20,'Kotak Agra Taj','Agra');

TABLE:

To view table query:

select * from branches;

ACCOUNTS TABLE:

QUERY:

```
CREATE TABLE `accounts` (
  `account_number` VARCHAR(20) NOT NULL,
  `customer_id` INT NOT NULL,
  `account_type` ENUM('Savings', 'Current', 'FD') NOT NULL,
  `balance` DECIMAL(15,2) DEFAULT '0.00',
  `branch_id` INT DEFAULT NULL,
  `opened_date` DATE NOT NULL,
  PRIMARY KEY (`account_number`),
  KEY `customer_id` (`customer_id`),
  KEY `branch_id` (`branch_id`),
  CONSTRAINT `accounts_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`),
  CONSTRAINT `accounts_ibfk_2` FOREIGN KEY (`branch_id`) REFERENCES `branches` (`branch_id`)
) ENGINE=InnoDB
  DEFAULT CHARSET=utf8mb4
  COLLATE=utf8mb4_0900_ai_ci;
INSERT INTO `accounts` VALUES ('KOTAK123456',1,'Savings',50000.00,1,'2020-01-15'),('KOTAK123457',2,'Current',120000.00,NULL,'2021-05-20'),('KOTAK123458',3,'FD',200000.00,3,'2022-03-10'),('KOTAK123459',4,'Savings',75000.00,4,'2019-11-25'),('KOTAK123460',5,'Current',90000.00,NULL,'2020-08-14'),('KOTAK123461',6,'Savings',30000.00,6,'2021-07-30'),('KOTAK123462',7,'FD',150000.00,7,'2022-02-
```

05'),('KOTAK123463',8,'Savings',60000.00,8,'2020-09-12'),('KOTAK123464',9,'Current',180000.00,NULL,'2019-04-18'),('KOTAK123465',10,'Savings',45000.00,10,'2021-12-01'),('KOTAK123466',11,'FD',250000.00,11,'2022-06-15'),('KOTAK123467',12,'Savings',55000.00,12,'2020-03-22'),('KOTAK123468',13,'Current',80000.00,NULL,'2021-08-10'),('KOTAK123469',14,'Savings',35000.00,14,'2019-10-05'),('KOTAK123470',15,'FD',300000.00,15,'2022-04-20'),('KOTAK123471',16,'Savings',70000.00,16,'2020-11-15'),('KOTAK123472',17,'Current',95000.00,NULL,'2021-09-25'),('KOTAK123473',18,'Savings',40000.00,18,'2019-06-30'),('KOTAK123474',19,'FD',175000.00,19,'2022-07-05'),('KOTAK123475',20,'Savings',48000.00,20,'2020-12-12');

TABLE:

To view table query:

select * from accounts;

LOANS TABLE:

QUERY:

```
CREATE TABLE `loans` (
  `loan_id` int NOT NULL AUTO_INCREMENT,
  `customer_id` int NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  `type` enum('Personal','Home','Car') NOT NULL,
  `status` enum('Pending','Approved','Rejected') DEFAULT 'Pending',
  `start_date` date NOT NULL,
  `end_date` date DEFAULT NULL,
  PRIMARY KEY (`loan_id`),
  KEY `customer_id` (`customer_id`),
  CONSTRAINT `loans_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `customers` (`customer_id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
INSERT INTO `loans` VALUES (1,1,500000.00,'Home','Approved','2022-01-10','2027-01-10'),(2,2,200000.00,'Personal','Pending','2022-05-15',NULL),(3,3,300000.00,'Car','Approved','2022-03-20','2025-03-
```

20'),(4,4,1000000.00,'Home','Pending','2022-07-01',NULL),(5,5,150000.00,'Personal','Rejected','2022-02-05',NULL),(6,6,400000.00,'Home','Approved','2022-04-12','2032-04-12'),(7,7,250000.00,'Car','Pending','2022-06-20',NULL),(8,8,600000.00,'Home','Approved','2022-08-15','2030-08-15'),(9,9,100000.00,'Personal','Pending','2022-09-10',NULL),(10,10,350000.00,'Car','Approved','2022-10-05','2026-10-05'),(11,11,700000.00,'Home','Pending','2022-11-20',NULL),(12,12,180000.00,'Personal','Approved','2022-12-01','2024-12-01'),(13,13,220000.00,'Car','Rejected','2023-01-15',NULL),(14,14,800000.00,'Home','Approved','2023-02-10','2033-02-10'),(15,15,120000.00,'Personal','Pending','2023-03-05',NULL),(16,16,450000.00,'Car','Approved','2023-04-20','2028-04-20'),(17,17,950000.00,'Home','Pending','2023-05-12',NULL),(18,18,160000.00,'Personal','Approved','2023-06-01','2025-06-01'),(19,19,280000.00,'Car','Pending','2023-07-10',NULL),(20,20,550000.00,'Home','Approved','2023-08-05','2035-08-05');

TABLE:

To view table query:

```
select * from loans;
```

TRANSACTIONS TABLE:

QUERY:

```
CREATE TABLE `transactions` (
  `transaction_id` int NOT NULL AUTO_INCREMENT,
  `account_number` varchar(20) NOT NULL,
  `type` enum('Deposit','Withdrawal','Transfer') NOT NULL,
  `amount` decimal(10,2) NOT NULL,
  `transaction_date` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `related_account` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`transaction_id`),
  KEY `account_number` (`account_number`),
  CONSTRAINT `transactions_ibfk_1` FOREIGN KEY (`account_number`) REFERENCES `accounts` (`account_number`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
INSERT INTO `transactions` VALUES (1,'KOTAK123456','Deposit',10000.00,'2025-03-16 10:52:52',NULL),(2,'KOTAK123457','Withdrawal',5000.00,'2025-03-16 10:52:52',NULL),(3,'KOTAK123458','Transfer',20000.00,'2025-03-16 10:52:52','KOTAK123456'),(4,'KOTAK123459','Deposit',15000.00,'2025-03-16 10:52:52',NULL),(5,'KOTAK123460','Transfer',10000.00,'2025-03-16 10:52:52','KOTAK123457'),(6,'KOTAK123461','Withdrawal',3000.00,'2025-03-16 10:52:52',NULL),(7,'KOTAK123462','Deposit',50000.00,'2025-03-16 10:52:52',NULL),(8,'KOTAK123463','Transfer',12000.00,'2025-03-16 10:52:52','KOTAK123460'),(9,'KOTAK123464','Withdrawal',8000.00,'2025-03-16 10:52:52',NULL),(10,'KOTAK123465','Deposit',7000.00,'2025-03-16 10:52:52',NULL),(11,'KOTAK123466','Transfer',25000.00,'2025-03-16 10:52:52','KOTAK123462'),(12,'KOTAK123467','Withdrawal',4500.00,'2025-03-16 10:52:52',NULL),(13,'KOTAK123468','Deposit',9000.00,'2025-03-16 10:52:52',NULL),(14,'KOTAK123469','Transfer',6000.00,'2025-03-16 10:52:52','KOTAK123465'),(15,'KOTAK123470','Withdrawal',10000.00,'2025-03-16 10:52:52',NULL),(16,'KOTAK123471','Deposit',12000.00,'2025-03-16 10:52:52',NULL),(17,'KOTAK123472','Transfer',15000.00,'2025-03-16 10:52:52','KOTAK123470'),(18,'KOTAK123473','Withdrawal',2000.00,'2025-03-16 10:52:52',NULL),(19,'KOTAK123474','Deposit',30000.00,'2025-03-16 10:52:52',NULL),(20,'KOTAK123475','Transfer',8000.00,'2025-03-16 10:52:52','KOTAK123473');
```

TABLE:

To view table query:

select * from transactions;

EMPLOYEES TABLE:

QUERY:

```
CREATE TABLE `employees` (
  `employee_id` int NOT NULL AUTO_INCREMENT,
  `first_name` varchar(50) NOT NULL,
  `last_name` varchar(50) NOT NULL,
  `branch_id` int DEFAULT NULL,
  `position` varchar(50) DEFAULT NULL,
  `salary` decimal(10,2) DEFAULT NULL,
  `hire_date` date NOT NULL,
  PRIMARY KEY (`employee_id`),
  KEY `branch_id` (`branch_id`),
  CONSTRAINT `employees_ibfk_1` FOREIGN KEY (`branch_id`) REFERENCES `branches` (`branch_id`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
INSERT INTO `employees` VALUES (1,'Aarav','Shah',1,'Manager',75000.00,'2018-01-15'),(2,'Isha','Patil',2,'Clerk',35000.00,'2020-03-22'),(3,'Rohan','Mehta',3,NULL,42000.00,'2019-07-10'),(4,'Priya','Singh',4,'Loan Officer',50000.00,'2021-05-05'),(5,'Vijay','Kumar',5,NULL,NULL,'2022-01-30'),(6,'Anaya','Desai',6,'Clerk',32000.00,'2020-11-12'),(7,'Karan','Joshi',7,'Manager',78000.00,'2017-09-18'),(8,'Sanya','Reddy',8,'Clerk',34000.00,'2021-08-25'),(9,'Rahul','Verma',9,NULL,45000.00,'2019-04-14'),(10,'Neha','Iyer',10,'Loan Officer',52000.00,'2020-06-09'),(11,'Arjun','Malhotra',11,'Manager',76000.00,'2018-12-01'),(12,'Mira','Gupta',12,'Clerk',33000.00,'2022-02-15'),(13,'Amit','Sharma',13,NULL,NULL,'2021-03-20'),(14,'Pooja','Rao',14,'Clerk',31000.00,'2020-07-07'),(15,'Vikram','Thakur',15,'Manager',77000.00,'2016-10-10'),(16,'Anjali','Dubey',16,NULL,48000.00,'2019-11-
```

21'),(17,'Raj','Sinha',17,'Clerk',34000.00,'2021-09-05'),(18,'Sunita','Chopra',18,'Loan Officer',51000.00,'2020-04-18'),(19,'Alok','Bansal',19,'Manager',79000.00,'2015-08-30'),(20,'Kiran','Srivastava',20,'Clerk',32000.00,'2022-05-12');

TABLE:

To view table query:

select * from employees;

4. ANALYZING BANK DATABASE WITH BASIC SQL QUERIES

1. List customers with a missing middle name

QUERY:

```
SELECT customer_id, first_name, last_name
FROM Customers
WHERE middle_name IS NULL;
```

TABLE:

Customer_id	First_name	Last_name
2	Priya	Patel
4	Sneha	Singh
7	Ravi	Joshi
9	Sanjay	Reddy
11	Arun	Iyer
13	Rajesh	Thakur
15	Alok	Bansal
17	Vivek	Dubey
19	Rohan	Gandhi

2. Find accounts with a balance greater than ₹1,00,000.

QUERY:

```
SELECT account_number, balance
FROM Accounts
WHERE balance > 100000;
```

TABLE:

Account_number	balance
KOTAK123457	120000.00
KOTAK123458	200000.00
KOTAK123462	150000.00
KOTAK123464	180000.00
KOTAK123466	250000.00
KOTAK123470	300000.00
KOTAK123474	175000.00

3.Show employees with an undefined position (NULL).

QUERY:

```
SELECT employee_id, first_name, last_name
FROM Employees
WHERE position IS NULL;
```

TABLE:

Employee_id	First_name	Lat_name
3	Rohan	Mehta
5	Vijay	Kumar
9	Rahul	Verma
13	Amit	Sharma
16	Anjali	Dubey

4. Count the number of approved loans.

QUERY:

```
SELECT COUNT(*) AS approved_loans
FROM Loans
```

```
WHERE status = 'Approved';
```

TABLE:

approved_loans
10

5.List transactions of type 'Deposit'.

QUERY:

```
SELECT transaction_id, account_number, amount
FROM Transactions
WHERE type = 'Deposit';
```

TABLE:

transaction_id	account_number	amount
1	KOTAK123456	10000.00
4	KOTAK123459	15000.00
7	KOTAK123462	50000.00
10	KOTAK123465	7000.00
13	KOTAK123468	9000.00
16	KOTAK123471	12000.00
19	KOTAK123474	30000.00

6.Find branches in Mumbai.

QUERY:

```
SELECT * FROM Branches
WHERE location LIKE '%Mumbai%';
```

TABLE:

branch_id	branch_name	location
1	Kotak Mumbai Main	Mumbai

7.Calculate the average salary of employees.

QUERY:

```
SELECT AVG(salary) AS avg_salary
FROM Employees;
```

TABLE:

avg_salary
50222.222222

8.List customers with alternate phone numbers (non-NULL).

QUERY:

```
SELECT customer_id, first_name, phone, alt_phone
FROM Customers
WHERE alt_phone IS NOT NULL;
```

TABLE:

customer_id	first_name	phone	alt_phone
2	Priya	9876543211	9876543212
5	Vikram	9876543215	9876543216
7	Ravi	9876543218	9876543219
10	Pooja	9876543222	9876543223
12	Kavita	9876543225	9876543226
14	Swati	9876543228	9876543229
16	Meera	9876543231	9876543232
18	Anita	9876543234	9876543235
20	Divya	9876543237	9876543238

9. Show FD accounts opened after 2022-01-01.

QUERY:

```
SELECT account_number, opened_date
FROM Accounts
WHERE account_type = 'FD' AND opened_date > '2022-01-01';
```

TABLE:

account_number	opened_date
KOTAK123474	2022-07-05
KOTAK123470	2022-04-20
KOTAK123466	2022-06-15
KOTAK123462	2022-02-05
KOTAK123458	2022-03-10

10. Find loans with no end date (NULL).

QUERY:

```
SELECT loan_id, customer_id, amount
FROM Loans
WHERE end_date IS NULL;
```

TABLE:

loan_id	customer_id	amount
2	2	200000.00
4	4	1000000.00
5	5	150000.00
7	7	250000.00
9	9	100000.00
11	11	700000.00

13	13	220000.00
15	15	120000.00
17	17	950000.00
19	19	280000.00

11. Count transactions involving transfers.

QUERY:

```
SELECT COUNT(*) AS total_transfers
FROM Transactions
WHERE type = 'Transfer';
```

TABLE:

total_transfers
7

12. List employees hired in 2022.

QUERY:

```
SELECT employee_id, first_name, hire_date
FROM Employees
WHERE YEAR(hire_date) = 2022;
```

TABLE:

employee_id	first_name	hire_date
20	Kiran	2022-05-12
12	Mira	2022-02-15
5	Vijay	2022-01-30

13. Find the highest account balance.

QUERY:

```
SELECT MAX(balance) AS max_balance
FROM Accounts;
```

TABLE:

max_balance
300000.00

14. List customers from Delhi or Mumbai.

QUERY:

```
SELECT customer_id, first_name, address
FROM Customers
WHERE address LIKE '%Delhi%' OR address LIKE '%Mumbai%';
```

TABLE:

customer_id	first_name	address
1	Rahul	Mumbai
2	Priya	Delhi

15. Show loans with amounts between ₹2,00,000 and ₹5,00,000.

QUERY:

```
SELECT loan_id, customer_id, amount
FROM Loans
WHERE amount BETWEEN 200000 AND 500000;
```

TABLE:

loan_id	customer_id	amount
1	1	500000.00
2	2	200000.00
3	3	300000.00
6	6	400000.00
7	7	250000.00
10	10	350000.00
13	13	220000.00
16	16	450000.00
19	19	280000.00

5. ANALYZING BANK DATABASE WITH SQL JOINS

16. List customers with their account numbers and types.

QUERY:

```
SELECT c.first_name, a.account_number, a.account_type
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id;
```

TABLE:

first_name	account_number	account_type
Rahul	KOTAK123456	Savings
Priya	KOTAK123457	Current
Amit	KOTAK123458	FD
Sneha	KOTAK123459	Savings
Vikram	KOTAK123460	Current
Anjali	KOTAK123461	Savings
Ravi	KOTAK123462	FD
Neha	KOTAK123463	Savings
Sanjay	KOTAK123464	Current
Pooja	KOTAK123465	Savings
Arun	KOTAK123466	FD
Kavita	KOTAK123467	Savings
Rajesh	KOTAK123468	Current
Swati	KOTAK123469	Savings
Alok	KOTAK123470	FD
Meera	KOTAK123471	Savings
Vivek	KOTAK123472	Current
Anita	KOTAK123473	Savings

Rohan	KOTAK123474	FD
Divya	KOTAK123475	Savings

17. Show transactions with customer names.

QUERY:

```
SELECT t.transaction_id, c.first_name, t.amount
FROM Transactions t
JOIN Accounts a ON t.account_number = a.account_number
JOIN Customers c ON a.customer_id = c.customer_id;
```

TABLE:

transaction_id	first_name	amount
1	Rahul	10000.00
2	Priya	5000.00
3	Amit	20000.00
4	Sneha	15000.00
5	Vikram	10000.00
6	Anjali	3000.00
7	Ravi	50000.00
8	Neha	12000.00
9	Sanjay	8000.00
10	Pooja	7000.00
11	Arun	25000.00
12	Kavita	4500.00
13	Rajesh	9000.00
14	Swati	6000.00
15	Alok	10000.00
16	Meera	12000.00
17	Vivek	15000.00

18	Anita	2000.00
19	Rohan	30000.00
20	Divya	8000.00

18. Find employees and their branch locations.

QUERY:

```
SELECT e.first_name, b.branch_name
FROM Employees e
LEFT JOIN Branches b ON e.branch_id = b.branch_id;
```

TABLE:

first_name	branch_name
Aarav	Kotak Mumbai Main
Isha	Kotak Delhi Central
Rohan	Kotak Bangalore Tech
Priya	Kotak Hyderabad City
Vijay	Kotak Chennai Coast
Anaya	Kotak Kolkata East
Karan	Kotak Pune West
Sanya	Kotak Ahmedabad North
Rahul	Kotak Jaipur Heritage
Neha	Kotak Lucknow Central
Arjun	Kotak Chandigarh Sector 17
Mira	Kotak Bhopal Lakeview
Amit	Kotak Surat Diamond
Pooja	Kotak Nagpur Orange
Vikram	Kotak Indore Trade
Anjali	Kotak Patna Historic
Raj	Kotak Coimbatore Textile

Sunita	Kotak Vizag Port
Alok	Kotak Thane Suburban
Kiran	Kotak Agra Taj

19. List approved loans with customer details.

QUERY:

```
SELECT c.first_name, l.amount, l.type
FROM Loans l
JOIN Customers c ON l.customer_id = c.customer_id
WHERE l.status = 'Approved';
```

TABLE:

first_name	amount	type
Rahul	500000.00	Home
Amit	300000.00	Car
Anjali	400000.00	Home
Neha	600000.00	Home
Pooja	350000.00	Car
Kavita	180000.00	Personal
Swati	800000.00	Home
Meera	450000.00	Car
Anita	160000.00	Personal
Divya	550000.00	Home

20. Calculate total deposits per account.

QUERY:

```
SELECT a.account_number, SUM(t.amount) AS total_deposits
FROM Transactions t
```

```
JOIN Accounts a ON t.account_number = a.account_number
WHERE t.type = 'Deposit'
GROUP BY a.account_number;
```

TABLE:

account_number	total_deposits
KOTAK123456	10000.00
KOTAK123459	15000.00
KOTAK123462	50000.00
KOTAK123465	7000.00
KOTAK123468	9000.00
KOTAK123471	12000.00
KOTAK123474	30000.00

21. Show branches with no employees.

QUERY:

```
SELECT b.branch_name
FROM Branches b
LEFT JOIN Employees e ON b.branch_id = e.branch_id
WHERE e.employee_id IS NULL;
```

TABLE:

branch_name

22. List customers with their total account balances.

QUERY:

```
SELECT c.customer_id, c.first_name, SUM(a.balance) AS
total_balance
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
```

```
GROUP BY c.customer_id;
```

TABLE:

customer_id	first_name	total_balance
1	Rahul	50000.00
2	Priya	120000.00
3	Amit	200000.00
4	Sneha	75000.00
5	Vikram	90000.00
6	Anjali	30000.00
7	Ravi	150000.00
8	Neha	60000.00
9	Sanjay	180000.00
10	Pooja	45000.00
11	Arun	250000.00
12	Kavita	55000.00
13	Rajesh	80000.00
14	Swati	35000.00
15	Alok	300000.00
16	Meera	70000.00
17	Vivek	95000.00
18	Anita	40000.00
19	Rohan	175000.00
20	Divya	48000.00

23. Find transactions with recipient account details (for transfers).

QUERY:

```
SELECT t.transaction_id, a.account_type AS recipient_type
FROM Transactions t
LEFT JOIN Accounts a ON t.related_account = a.account_number
WHERE t.type = 'Transfer';
```

TABLE:

transaction_id	recipient_type
3	Savings
5	Current
8	Current
11	FD
14	Savings
17	FD
20	Savings

24. Show employees earning more than their branch's average salary.

QUERY:

```
SELECT e1.first_name, e1.salary
FROM Employees e1
JOIN (
    SELECT branch_id, AVG(salary) AS avg_salary
    FROM Employees
    GROUP BY branch_id
) e2 ON e1.branch_id = e2.branch_id
WHERE e1.salary > e2.avg_salary;
```

TABLE:
first_name, salary

25. List customers with active loans and their loan types.

QUERY:
```
SELECT c.first_name, l.type
FROM Customers c
JOIN Loans l ON c.customer_id = l.customer_id
WHERE l.status = 'Approved';
```

TABLE:

first_name	type
Rahul	Home
Amit	Car
Anjali	Home
Neha	Home
Pooja	Car
Kavita	Personal
Swati	Home
Meera	Car
Anita	Personal
Divya	Home

26. Find accounts with no transactions.

QUERY:
```
SELECT a.account_number
FROM Accounts a
LEFT JOIN Transactions t ON a.account_number = t.account_number
```

```
WHERE t.transaction_id IS NULL;
```

TABLE:

account_number

27. Show the latest transaction for each account.

QUERY:

```
SELECT a.account_number, MAX(t.transaction_date) AS latest_transaction
FROM Accounts a
LEFT JOIN Transactions t ON a.account_number = t.account_number
GROUP BY a.account_number;
```

TABLE:

account_number	latest_transaction
KOTAK123456	2025-03-16 10:52:52
KOTAK123457	2025-03-16 10:52:52
KOTAK123458	2025-03-16 10:52:52
KOTAK123459	2025-03-16 10:52:52
KOTAK123460	2025-03-16 10:52:52
KOTAK123461	2025-03-16 10:52:52
KOTAK123462	2025-03-16 10:52:52
KOTAK123463	2025-03-16 10:52:52
KOTAK123464	2025-03-16 10:52:52
KOTAK123465	2025-03-16 10:52:52
KOTAK123466	2025-03-16 10:52:52
KOTAK123467	2025-03-16 10:52:52
KOTAK123468	2025-03-16 10:52:52
KOTAK123469	2025-03-16 10:52:52

KOTAK123470	2025-03-16 10:52:52
KOTAK123471	2025-03-16 10:52:52
KOTAK123472	2025-03-16 10:52:52
KOTAK123473	2025-03-16 10:52:52
KOTAK123474	2025-03-16 10:52:52
KOTAK123475	2025-03-16 10:52:52

28. List branches with total employee salaries.

QUERY:

```
SELECT b.branch_name, SUM(e.salary) AS total_salary
FROM Branches b
LEFT JOIN Employees e ON b.branch_id = e.branch_id
GROUP BY b.branch_id;
```

TABLE:

branch_name	total_salary
Kotak Mumbai Main	75000.00
Kotak Delhi Central	35000.00
Kotak Bangalore Tech	42000.00
Kotak Hyderabad City	50000.00
Kotak Chennai Coast	
Kotak Kolkata East	32000.00
Kotak Pune West	78000.00
Kotak Ahmedabad North	34000.00
Kotak Jaipur Heritage	45000.00
Kotak Lucknow Central	52000.00
Kotak Chandigarh Sector 17	76000.00
Kotak Bhopal Lakeview	33000.00
Kotak Surat Diamond	

Kotak Nagpur Orange	31000.00
Kotak Indore Trade	77000.00
Kotak Patna Historic	48000.00
Kotak Coimbatore Textile	34000.00
Kotak Vizag Port	51000.00
Kotak Thane Suburban	79000.00
Kotak Agra Taj	32000.00

29. Find customers with both Savings and FD accounts.

QUERY:

```
SELECT c.customer_id, c.first_name
FROM Customers c
JOIN Accounts a1 ON c.customer_id = a1.customer_id AND a1.account_type = 'Savings'
JOIN Accounts a2 ON c.customer_id = a2.customer_id AND a2.account_type = 'FD';
```

TABLE:

customer_id, first_name

30. List employees and their managers (if manager_id exists)...

(Add a manager_id column to the Employees table first.)

UPDATE:

```
ALTER TABLE Employees ADD manager_id INT;

-- Top managers (no manager_id)
UPDATE Employees SET manager_id = NULL WHERE employee_id IN (1, 7, 11, 15, 19);
-- Employees reporting to Aarav Shah
```

```
UPDATE Employees SET manager_id = 1 WHERE employee_id IN (2, 3);
-- Employees reporting to Karan Joshi
UPDATE Employees SET manager_id = 7 WHERE employee_id IN (4, 5, 6);
-- Employees reporting to Arjun Malhotra
UPDATE Employees SET manager_id = 11 WHERE employee_id IN (8, 9, 10);
-- Employees reporting to Vikram Thakur
UPDATE Employees SET manager_id = 15 WHERE employee_id IN (12, 13, 14);
-- Employees reporting to Alok Bansal
UPDATE Employees SET manager_id = 19 WHERE employee_id IN (16, 17, 18, 20);
```

QUERY:

```
SELECT e1.first_name AS employee, e2.first_name AS manager
FROM Employees e1
LEFT JOIN Employees e2 ON e1.manager_id = e2.employee_id;
```

TABLE:

Employee	manager
Aarav	
Isha	Aarav
Rohan	Aarav
Priya	Karan
Vijay	Karan
Anaya	Karan
Karan	

Sanya	Arjun
Rahul	Arjun
Neha	Arjun
Arjun	
Mira	Vikram
Amit	Vikram
Pooja	Vikram
Vikram	
Anjali	Alok
Raj	Alok
Sunita	Alok
Alok	
Kiran	Alok

6. ANALYZING BANK DATABASE WITH WINDOW FUNCTION

31. Rank customers by balance (highest to lowest)

QUERY:

```
SELECT
  c.customer_id,
  CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
  a.balance,
  RANK() OVER (ORDER BY a.balance DESC) AS balance_rank
FROM accounts a
JOIN customers c ON a.customer_id = c.customer_id;
```

TABLE:

Customer_id	Customer_name	balance	balance_rank
15	Alok Bansal	300000.00	1
11	Arun Iyer	250000.00	2
3	Amit Verma	200000.00	3
9	Sanjay Reddy	180000.00	4
19	Rohan Gandhi	175000.00	5
7	Ravi Joshi	150000.00	6
2	Priya Patel	120000.00	7
17	Vivek Dubey	95000.00	8
5	Vikram Gupta	90000.00	9
13	Rajesh Thakur	80000.00	10
4	Sneha Singh	75000.00	11
16	Meera Srivastava	70000.00	12
8	Neha Malhotra	60000.00	13

12	Kavita Rao	55000.00	14
1	Rahul Sharma	50000.00	15
20	Divya Sinha	48000.00	16
10	Pooja Desai	45000.00	17
18	Anita Shukla	40000.00	18
14	Swati Chopra	35000.00	19
6	Anjali Mehta	30000.00	20

32. Get the running total of transaction amounts per account

QUERY:

```
SELECT
  account_number,
  transaction_id,
  amount,
  SUM(amount) OVER (PARTITION BY account_number
ORDER BY transaction_id) AS running_total
FROM transactions;
```

TABLE:

Account_number	transaction_id	amount	running_total
KOTAK123456	1	10000.00	10000.00
KOTAK123457	2	5000.00	5000.00
KOTAK123458	3	20000.00	20000.00
KOTAK123459	4	15000.00	15000.00
KOTAK123460	5	10000.00	10000.00
KOTAK123461	6	3000.00	3000.00
KOTAK123462	7	50000.00	50000.00
KOTAK123463	8	12000.00	12000.00
KOTAK123464	9	8000.00	8000.00

KOTAK123465	10	7000.00	7000.00
KOTAK123466	11	25000.00	25000.00
KOTAK123467	12	4500.00	4500.00
KOTAK123468	13	9000.00	9000.00
KOTAK123469	14	6000.00	6000.00
KOTAK123470	15	10000.00	10000.00
KOTAK123471	16	12000.00	12000.00
KOTAK123472	17	15000.00	15000.00
KOTAK123473	18	2000.00	2000.00
KOTAK123474	19	30000.00	30000.00
KOTAK123475	20	8000.00	8000.00

33. Get the average balance per branch and customer balance compared to branch average

QUERY:

```
SELECT
  b.branch_name,
  a.account_number,
  a.balance,
  ROUND(AVG(a.balance) OVER (PARTITION BY b.branch_id), 2) AS branch_avg_balance
FROM accounts a
JOIN branches b ON a.branch_id = b.branch_id;
```

TABLE:

Branch_name	Account_number	balance	Branch_avg_balance
Kotak Mumbai Main	KOTAK123456	50000.00	50000.00
Kotak Bangalore Tech	KOTAK123458	200000.00	200000.00

Kotak Hyderabad City	KOTAK123459	75000.00	75000.00
Kotak Kolkata East	KOTAK123461	30000.00	30000.00
Kotak Pune West	KOTAK123462	150000.00	150000.00
Kotak Ahmedabad North	KOTAK123463	60000.00	60000.00
Kotak Lucknow Central	KOTAK123465	45000.00	45000.00
Kotak Chandigarh Sector 17	KOTAK123466	250000.00	250000.00
Kotak Bhopal Lakeview	KOTAK123467	55000.00	55000.00
Kotak Nagpur Orange	KOTAK123469	35000.00	35000.00
Kotak Indore Trade	KOTAK123470	300000.00	300000.00
Kotak Patna Historic	KOTAK123471	70000.00	70000.00
Kotak Vizag Port	KOTAK123473	40000.00	40000.00
Kotak Thane Suburban	KOTAK123474	175000.00	175000.00
Kotak Agra Taj	KOTAK123475	48000.00	48000.00

7. ANALYZING BANK DATABASE WITH STORED PROCEDURE

34. Procedure to get all accounts of a customer by their email

QUERY:

```
DELIMITER $$
CREATE PROCEDURE GetCustomerAccountsByEmail(IN
customerEmail VARCHAR(100))
BEGIN
  SELECT
    c.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS
customer_name,
    a.account_number,
    a.account_type,
    a.balance,
    a.opened_date
  FROM customers c
  JOIN accounts a ON c.customer_id = a.customer_id
  WHERE c.email = customerEmail;
END $$
DELIMITER ;
CALL
GetCustomerAccountsByEmail('rahul@example.com');
```

TABLE:

Customer_id	Customer_name	Account_number	Account_type	balance	Opened_date
1	Rahul Sharma	KOTAK123456	Savings	50000.00	2020-01-15

35. Procedure to deposit money into an account

QUERY:

```
DELIMITER $$
CREATE PROCEDURE DepositAmount(IN acc_num VARCHAR(20), IN amt DECIMAL(10,2))
BEGIN
  UPDATE accounts
  SET balance = balance + amt
  WHERE account_number = acc_num;
  INSERT INTO transactions(account_number, type, amount, transaction_date)
  VALUES (acc_num, 'Deposit', amt, NOW());
END $$
DELIMITER ;
CALL DepositAmount('KOTAK123456', 5000.00);
```

8. ANALYZING BANK DATABASE WITH TRIGGERS

36. Trigger to automatically update balance after transaction

QUERY:

```
DELIMITER $$

CREATE TRIGGER update_balance_after_transaction
AFTER INSERT ON transactions
FOR EACH ROW
BEGIN
  IF NEW.type = 'Deposit' THEN
    UPDATE accounts
    SET balance = balance + NEW.amount
    WHERE account_number = NEW.account_number;
  ELSEIF NEW.type = 'Withdrawal' THEN
    UPDATE accounts
    SET balance = balance - NEW.amount
    WHERE account_number = NEW.account_number;
  ELSEIF NEW.type = 'Transfer' THEN
    UPDATE accounts
    SET balance = balance - NEW.amount
    WHERE account_number = NEW.account_number;
    UPDATE accounts
    SET balance = balance + NEW.amount
    WHERE account_number = NEW.related_account;
  END IF;
END $$

DELIMITER ;
```

9. CHALLENGES AND LEARNINGS

CHALLENGES:

1. Data Consistency Across Tables:
Initially, it was tough to keep data consistent between related tables. I overcame this by using foreign key constraints and enabling cascade updates/deletes.

2. Understanding Foreign Keys:
I struggled to relate multiple tables correctly. I solved this by studying normalization and practicing with ER diagrams to map one-to-many relationships.

3. Designing the Transactions Table:
It was challenging to design a flexible structure for deposits and withdrawals. I handled this by adding a transaction_type column and linking each record with account_id.

LEARNINGS:

I learned how to:

- ✓ Build a normalized relational database from scratch.
- ✓ Use SQL constraints effectively.
- ✓ Write optimized JOIN queries to fetch relevant data.
- ✓ Understand banking logic like linking accounts to branches and mapping loans to customers.