Bank Management System

Presented By -

63 - Pooja Sawant
70 - Chandraprabha Tawade
73 - Sakshi Tripathi

TABLE OF CONTENTS

Problem Statement

The Bank Management System (BMS) streamlines banking operations by automating tasks like account management, transactions (deposit, withdrawal, transfer), and loan applications. Customers can easily access their accounts, while administrators manage accounts, approve loans, and generate reports. The BMS improves accuracy, speed, and security, offering a user-friendly platform for efficient financial management. It addresses issues of error-prone traditional banking, ensuring real-time transaction processing and secure data storage, enhancing overall efficiency.

Introduction

- What is a Bank Management System?

A Bank Management System (BMS) is a comprehensive solution that automates and manages banking operations such as customer account handling, loan approvals, and transaction management through a relational database. The system is designed to improve banking efficiency by eliminating human errors, streamlining operations, and ensuring faster service delivery.

- Importance of DBMS in BMS:

A DBMS provides an organized, secure, and scalable environment to manage large sets of banking data. It ensures the accuracy and integrity of customer data, enables real-time transaction updates, and helps banks meet the increasing demand for speed and efficiency.

With DBMS, relational data models can represent complex relationships like customer-account, customer-loan, and account-transaction in a way that enhances service reliability and security.

OBJECTIVE

Accuracy: Ensure precise management of customer accounts, transactions, and loan details.
Usability: Provide an easy-to-use system for both employees and customers to manage banking operations.
Efficiency: Automate the process of handling banking operations to save time and reduce manual errors.
Effectiveness: Enhance the overall performance and customer service of the bank. Speed: Improve the speed of data retrieval and transaction processing.
User-friendliness: Design a system that is intuitive and easy for non-technical users to operate. example in depth with real life example

• RELATIONAL DATABASE DESIGN

1. Customer_Info :- customer_id, name, phone_number, email, address, date_of_ birth, account_no, card_id, online_banking_id
2. Borrower :- Borrower_Id, Loan_type, Loan_amount,Loan_id, Customer_id
3. ATM_Transaction :- ATM_Transactio_id,ATM_Location, Transaction_Type, Amount, Date_of_Transsaction,Card_id
4. Insurance :- Insurance_id, Insurance_Type,Insurance_amount,Customer_id
5. Online_Banking :-Online_Banking_id, usernmae, password, Customer_id
6. Loans:- Loan_id, Loan_amount,Interest_rate, Loan_Duration, Branch_id
7. Loan_Payments :-Payment_ID, Payment_Amount, Payment_Date, Loan_ID, Customer_ID
8. Branch :- Branch_ID, Branch_Name, Branch_Location
9. Cards:- Card_ID, Card_Type, Expiry_Date, CVV, Account_No
10. Depositor :- Depositor_ID, Deposit_Type, Deposit_Amount, Date_of_Deposit, Customer_ID
11. Account:- Account_no, Account_type, Balance, Branch_id, Depositor_ID, Borrower_ID
12. Transaction_Info :- Transaction_ID, Transaction_type, Amount, Date_of_Transaction, Customer_ID, Account_No
13. Employee :- Employee_ID, First_Name, Last_Name, Position,Department, Branch_ID

REQUIREMENT ANALYSIS

REQUIREMENT ANALYSIS

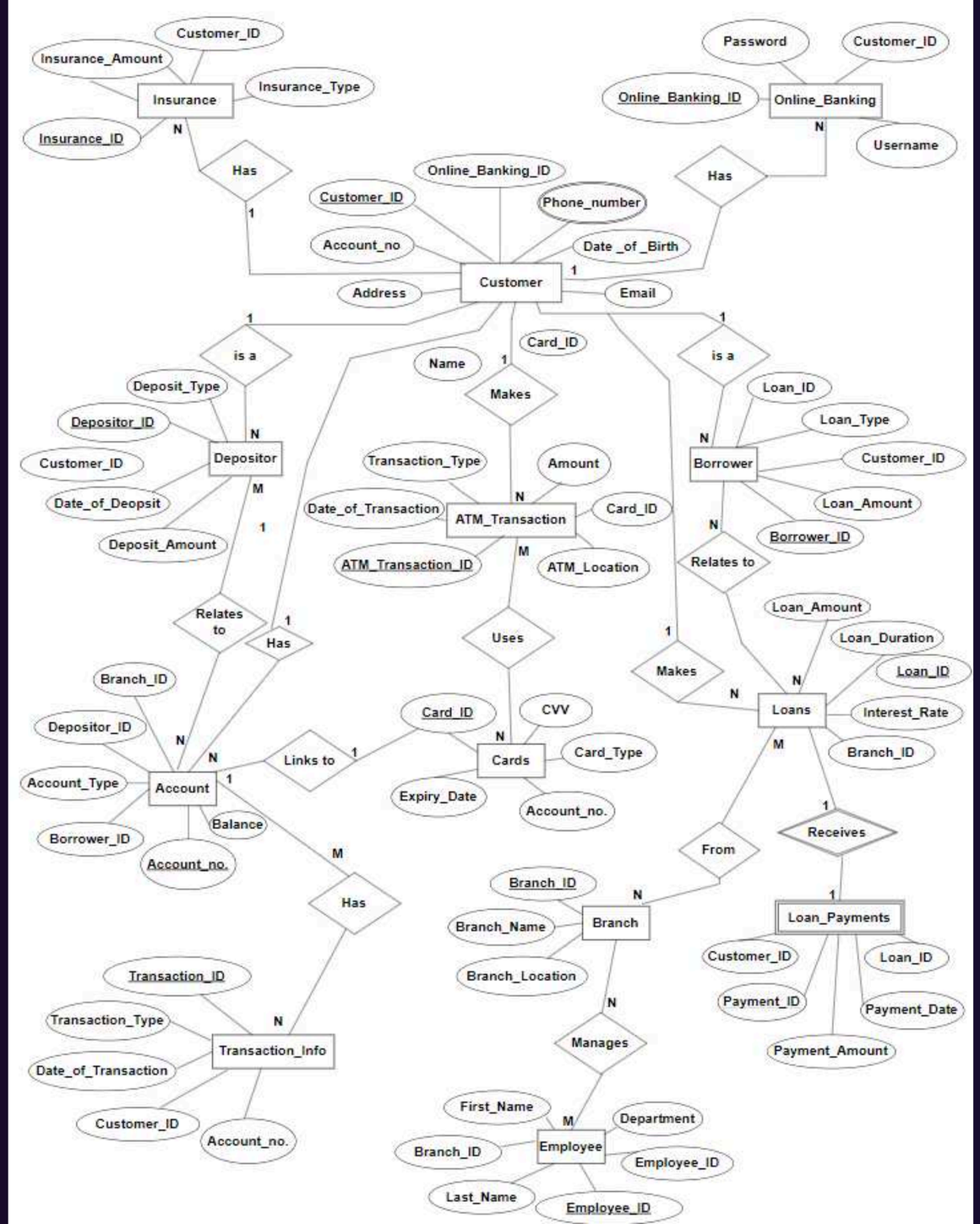
For Customers:

- The user will be able to create a bank account.
- The user should be able to check account balance, view transaction history, and account details.
- The user will be able to deposit money into their account.
- The user should be able to withdraw money from their account (if balance permits).
- The user can transfer money to other accounts within the bank or to external accounts.
- Users must provide required personal information like Name, Address, Identity proof for account creation.
- Users should be able to authenticate using a secure password or OTP system.
- After creating an account, the user may request to close or freeze the account.

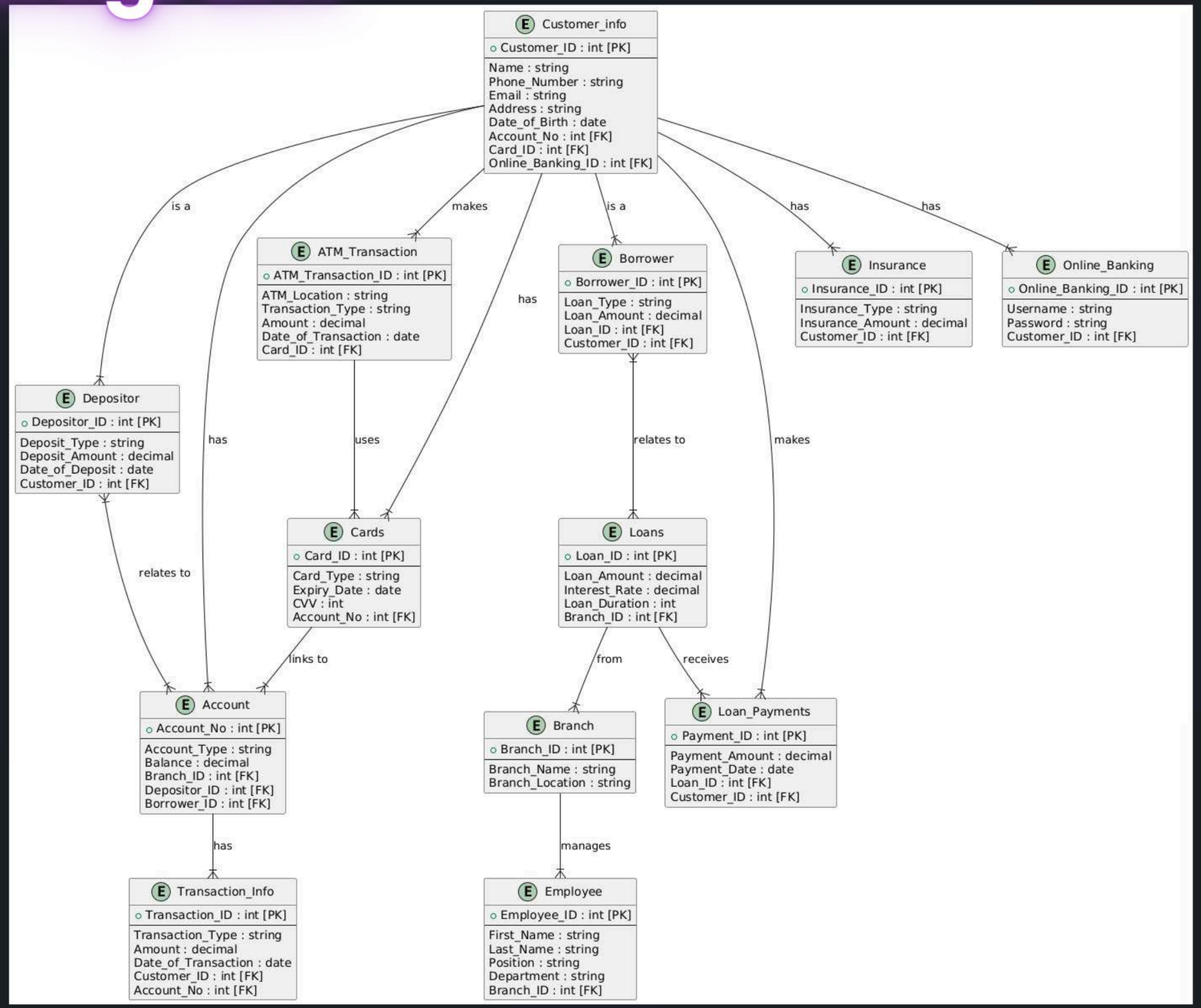
For Administrators:

- Should be able to add new customer accounts to the system.
- Should be able to modify customer details.
- Should be able to view all transactions and accounts.
- Should be able to remove or suspend accounts based on issues like fraud or inactivity.
- Should be able to set interest rates for savings and loan

ER DIAGRAM


Customer_ID
Insurance_Amount
Insurance
Insurance_Type
Insurance_ID
N
Has
1
Password
Customer_ID
Online_Banking_ID
Online_Banking
N
Username
Has
Online_Banking_ID
Customer_ID
Phone_number
Account_no
Date _of _Birth
1
Address
Customer
Email
1
1
is a
Card_ID
Name
1
Makes
1
is a
Deposit_Type
Loan_ID
Depositor_ID
Loan_Type
N
N
Depositor
Transaction_Type
Amount
Borrower
Customer_ID
Customer_ID
M
N
Date_of_Transaction
ATM_Transaction
Card_ID
Loan_Amount
Date_of_Deopsit
1
N
Borrower_ID
Deposit_Amount
M
ATM_Transaction_ID
ATM_Location
Relates to
Loan_Amount
Relates to
1
Has
Uses
1
Loan_Duration
Makes
Loan_ID
Branch_ID
N
N
Loans
Card_ID
CVV
Interest_Rate
Depositor_ID
N
N
Links to
1
Cards
Card_Type
M
Branch_ID
Account_Type
Account
1
Expiry_Date
Account_no.
Balance
Receives
Borrower_ID
Account_no.
M
From
1
Has
Branch_ID
N
Loan_Payments
Branch_Name
Branch
Customer_ID
Loan_ID
Transaction_ID
Branch_Location
Payment_ID
Payment_Date
N
Transaction_Type
N
Manages
Payment_Amount
Transaction_Info
Date_of_Transaction
First_Name
Department
M
Customer_ID
Account_no.
Branch_ID
Employee
Employee_ID
Last_Name
Employee_ID

Schema diagram



SQL Commands and Queries

A total of 52 SQL queries have been successfully performed on the database

Basic Commands(SELECT, INSERT, CREATE, DROP):

1.. Create table depositor and insert values:

```
bms=# create table Depositor(Depositor_ID serial primary key, Deposit_Type varchar(50) not null, Deposit_Amount numeric(10,2) not null, Date_of_Deposit date not null, Customer_ID
int , foreign key (Customer_ID) references Customer_info(customer_id));
CREATE TABLE
bms=# insert into Depositor(Deposit_Type, Deposit_Amount , Date_of_Deposit, Customer_ID) values('Fixed','50000','2023-06-30','1');
INSERT 0 1
bms=# select * from Depositor;
 depositor_id | deposit_type | deposit_amount | date_of_deposit | customer_id
--------------+--------------+----------------+-----------------+-------------
            1 | Fixed        |       50000.00 | 2023-06-30      |           1
(1 row)


bms=# insert into Depositor(Deposit_Type, Deposit_Amount , Date_of_Deposit, Customer_ID) values('Recurring','20000','2023-08-10','2');
INSERT 0 1
bms=# insert into Depositor(Deposit_Type, Deposit_Amount , Date_of_Deposit, Customer_ID) values('Fixed','150000','2023-12-01','3');
INSERT 0 1
bms=# insert into Depositor(Deposit_Type, Deposit_Amount , Date_of_Deposit, Customer_ID) values('Savings','10000','2023-04-15','4');
INSERT 0 1
bms=# insert into Depositor(Deposit_Type, Deposit_Amount , Date_of_Deposit, Customer_ID) values('Fixed','75000','2023-02-20','5');
INSERT 0 1
bms=# select * from Depositor;
 depositor_id | deposit_type | deposit_amount | date_of_deposit | customer_id
--------------+--------------+----------------+-----------------+-------------
            1 | Fixed        |       50000.00 | 2023-06-30      |           1
            2 | Recurring    |       20000.00 | 2023-08-10      |           2
            3 | Fixed        |      150000.00 | 2023-12-01      |           3
            4 | Savings      |       10000.00 | 2023-04-15      |           4
            5 | Fixed        |       75000.00 | 2023-02-20      |           5
(5 rows)
```

2. Create table employees and insert values:

```
bms=# CREATE TABLE Employee ( Employee_ID INT PRIMARY KEY, First_Name VARCHAR(50), Last_Name VARCHAR(50) , Position VARCHAR(50), Department VARCHAR(50), Branch_ID INT, FOREIGN KEY
 (Branch_ID) REFERENCES Branch(Branch_ID));
CREATE TABLE
bms=# INSERT INTO Employee (Employee_ID, First_Name, Last_Name, Position,  Department ,Branch_ID) VALUES (1, 'Suresh', 'Kumar', 'Manager', 'Loans', 1), (2, 'Ramesh', 'Singh', 'Cle
rk', 'Deposits', 2), (3, 'Anita', 'Gupta', 'Officer', 'Customer', 3), (4, 'Sunita', 'Reddy', 'Manager' , 'Accounts', 4), (5, 'Rajesh', 'Sharma', 'Clerk', 'Cards', 5);
INSERT 0 5
bms=# select * from Employee;
 employee_id | first_name | last_name | position | department | branch_id
-------------+------------+-----------+----------+------------+-----------
           1 | Suresh     | Kumar     | Manager  | Loans      |         1
           2 | Ramesh     | Singh     | Clerk    | Deposits   |         2
           3 | Anita      | Gupta     | Officer  | Customer   |         3
           4 | Sunita     | Reddy     | Manager  | Accounts   |         4
           5 | Rajesh     | Sharma    | Clerk    | Cards      |         5
(5 rows)
```

3. Retrieve the first three records from the account table:

```
bms=# select * from Account
bms-# limit 3;
 account_no | account_type |  balance  | branch_id | depositor_id | borrower_id
------------+--------------+-----------+-----------+--------------+-------------
       1001 | Savings      |  50000.50 |         1 |            1 |           1
       1002 | Current      | 200000.75 |         2 |            2 |           2
       1003 | Fixed        | 150000.25 |         3 |            3 |           3
(3 rows)
```

4. Insert new Branch record into the Branch table:

```
bms=# select * from Branch;
 branch_id | branch_name | branch_location
-----------+-------------+-----------------
         1 | HDFC        | Delhi
         2 | ICICI       | Bangalore
         3 | ICICI       | Kolkata
         4 | HDFC        | Mumbai
         5 | SBI         | Delhi
         6 | SBI         | Pune
(6 rows)
```

5. Delete a table from the database:

```
bms=# drop table customer_info;
DROP TABLE
bms=# select * from customer_info;
ERROR:  relation "customer_info" does not exist
LINE 1: select * from customer_info;
                      ^
```

Row and Column Queries:

6. Retrieve the loan amount above 20000:

```
bms=# select * from Loans_Payment where payment_amount>20000.00;
 payment_id | payment_amount | payment_date | loan_id | customer_id
------------+----------------+--------------+---------+-------------
          1 |       50000.00 | 2024-09-10   |       1 |           1
          2 |       20000.25 | 2024-09-15   |       2 |           2
          4 |       25000.00 | 2024-09-25   |       4 |           4
(3 rows)
```

7.Retrieve the employees whose position is manager

```
bms=# select * from Employee where position = 'Manager';
 employee_id | first_name | last_name | position | department | branch_id
-------------+------------+-----------+----------+------------+-----------
           1 | Suresh     | Kumar     | Manager  | Loans      |         1
           4 | Sunita     | Reddy     | Manager  | Accounts   |         4
(2 rows)
```

8.Retrieve the last three records from the borrower table:.

```
bms=# select * from Borrower order by Borrower_ID desc limit 3;
 borrower_id |    loan_type     | loan_amount | loan_id | customer_id
-------------+------------------+-------------+---------+-------------
           5 | Gold Loan        |   400000.25 |       5 |           5
           4 | Education Loan   |   150000.50 |       4 |           4
           3 | Car Loan         |   200000.75 |       3 |           3
(3 rows)
```

9. Find depositor ID with amount depositing greater than 50000 from the depositor table:

```
bms=# select Depositor_ID, Deposit_Amount from Depositor where Deposit_Amount
 > 50000;
 depositor_id | deposit_amount
--------------+----------------
            3 |      150000.00
            5 |       75000.00
(2 rows)
```

10. Select specific column from ATM transaction table(select ATM TransactionID ,transaction type from):

```
bms=# select atm_transaction_id,transaction_type from ATM_Transaction;
 atm_transaction_id | transaction_type
--------------------+------------------
                  1 | Withdrawal
                  2 | Withdrawal
                  3 | Deposit
                  4 | Deposit
                  5 | Withdrawal
(5 rows)
```

Modifying Commands – (UPDATE, ALTER, RENAME, DELETE):

11. Update the address of the customer with account number 1001:

```
bms=# select * from Customer_Info;
 customer_id |     name      | phone_number |         email          |  address  | date_of_birth | account_no | card_id | online_banking_id
-------------+---------------+--------------+------------------------+-----------+---------------+------------+---------+-------------------
           1 | Rahul Sharma  | 9876543210   | rahul.sharma@gamil.com | Delhi     | 1990-05-23    |       1001 |     201 |               301
           2 | Priya Gupta   | 9123456789   | priya.gupta@yahoo.com  | Mumbai    | 1992-07-11    |       1002 |     202 |               302
           3 | Amit Verma    | 9988776655   | amit.verma@outlook.com | Kolkata   | 1988-03-15    |       1003 |     203 |               303
           4 | Riya Sen      | 9898989898   | riya.sen@gmail.com     | Bangalore | 1995-01-29    |       1004 |     204 |               304
           5 | Manish Singh  | 9765432109   | manish.singh@gmail.com | Pune      | 1991-08-22    |       1005 |     205 |               305
(5 rows)


bms=# update Customer_Info
bms-# set address = 'Pune'
bms-# where Account_No = 1001;
UPDATE 1
bms=# select * from Customer_Info;
 customer_id |     name      | phone_number |         email          |  address  | date_of_birth | account_no | card_id | online_banking_id
-------------+---------------+--------------+------------------------+-----------+---------------+------------+---------+-------------------
           2 | Priya Gupta   | 9123456789   | priya.gupta@yahoo.com  | Mumbai    | 1992-07-11    |       1002 |     202 |               302
           3 | Amit Verma    | 9988776655   | amit.verma@outlook.com | Kolkata   | 1988-03-15    |       1003 |     203 |               303
           4 | Riya Sen      | 9898989898   | riya.sen@gmail.com     | Bangalore | 1995-01-29    |       1004 |     204 |               304
           5 | Manish Singh  | 9765432109   | manish.singh@gmail.com | Pune      | 1991-08-22    |       1005 |     205 |               305
           1 | Rahul Sharma  | 9876543210   | rahul.sharma@gamil.com | Pune      | 1990-05-23    |       1001 |     201 |               301
(5 rows)
```

12. Alter The account table to add a new column named “date of opening”:

```
bms=# select * from Account;
 account_no | account_type | current_balance | branch_id | depositor_id | borrower_id
------------+--------------+-----------------+-----------+--------------+-------------
       1001 | Savings      |        50000.50 |         1 |            1 |           1
       1002 | Current      |       200000.75 |         2 |            2 |           2
       1003 | Fixed        |       150000.25 |         3 |            3 |           3
       1004 | Recurring    |        10000.00 |         4 |            4 |           4
       1005 | Savings      |       750000.50 |         5 |            5 |           5
(5 rows)


bms=# alter table Account ADD COLUMN date_of_opening date;
ALTER TABLE
bms=# select * from Account;
 account_no | account_type | current_balance | branch_id | depositor_id | borrower_id | date_of_opening
------------+--------------+-----------------+-----------+--------------+-------------+-----------------
       1001 | Savings      |        50000.50 |         1 |            1 |           1 |
       1002 | Current      |       200000.75 |         2 |            2 |           2 |
       1003 | Fixed        |       150000.25 |         3 |            3 |           3 |
       1004 | Recurring    |        10000.00 |         4 |            4 |           4 |
       1005 | Savings      |       750000.50 |         5 |            5 |           5 |
(5 rows)
```

13. Rename the balance column to current_balance in the account table:

```
bms=# select * from Account;
 account_no | account_type |  balance  | branch_id | depositor_id | borrower_id
------------+--------------+-----------+-----------+--------------+-------------
       1001 | Savings      |  50000.50 |         1 |            1 |           1
       1002 | Current      | 200000.75 |         2 |            2 |           2
       1003 | Fixed        | 150000.25 |         3 |            3 |           3
       1004 | Recurring    |  10000.00 |         4 |            4 |           4
       1005 | Savings      | 750000.50 |         5 |            5 |           5
(5 rows)


bms=# alter table Account rename column balance to current_balance;
ALTER TABLE
bms=# select * from Account;
 account_no | account_type | current_balance | branch_id | depositor_id | borrower_id
------------+--------------+-----------------+-----------+--------------+-------------
       1001 | Savings      |        50000.50 |         1 |            1 |           1
       1002 | Current      |       200000.75 |         2 |            2 |           2
       1003 | Fixed        |       150000.25 |         3 |            3 |           3
       1004 | Recurring    |        10000.00 |         4 |            4 |           4
       1005 | Savings      |       750000.50 |         5 |            5 |           5
(5 rows)
```

4. Delete a transaction based on transaction id:

```
bms=# select * from Transaction_Info;
 transaction_id | account_no | transaction_type |  amount  | date_of_transaction | customer_id
----------------+------------+------------------+----------+---------------------+-------------
              1 |       1001 | Credit           | 10000.00 | 2024-09-10          |           1
              2 |       1002 | Debit            |  5000.00 | 2024-09-15          |           2
              3 |       1003 | Credit           | 15000.00 | 2024-09-18          |           3
              4 |       1004 | Debit            |  8000.00 | 2024-09-20          |           4
              5 |       1005 | Credit           | 20000.00 | 2024-09-25          |           5
(5 rows)


bms=# delete from Transaction_Info where transaction_id='2';
DELETE 1
bms=# select * from Transaction_Info;
 transaction_id | account_no | transaction_type |  amount  | date_of_transaction | customer_id
----------------+------------+------------------+----------+---------------------+-------------
              1 |       1001 | Credit           | 10000.00 | 2024-09-10          |           1
              3 |       1003 | Credit           | 15000.00 | 2024-09-18          |           3
              4 |       1004 | Debit            |  8000.00 | 2024-09-20          |           4
              5 |       1005 | Credit           | 20000.00 | 2024-09-25          |           5
(4 rows)
```

Compound Conditions – (AND, OR, BETWEEN, LIKE, NOT LIKE,NOT BETWEEN):

15.Find the loan amount between 10,00,000 and 51,00,000:

```
bms=# select * from Loans where Loan_Amount >= 1000000 AND Loan_Amount
<= 5100000;
 loan_id | loan_amount | interest_rate | loan_duration | branch_id
---------+-------------+---------------+---------------+-----------
       1 |  5000000.50 |          7.50 |            15 |         1
       2 |  1000000.25 |          9.00 |             5 |         2
       4 |  2000000.50 |          8.00 |            10 |         4
(3 rows)
```

16. Find loans with an interest rate greater than 7% and a loan duration longer than 5 years

```
bms=# select * from Loans;
 loan_id | loan_amount | interest_rate | loan_duration | branch_id
---------+-------------+---------------+---------------+-----------
       1 |  5000000.50 |          7.50 |            15 |         1
       2 |  1000000.25 |          9.00 |             5 |         2
       3 |   500000.75 |         12.00 |             3 |         3
       4 |  2000000.50 |          8.00 |            10 |         4
       5 |   300000.00 |         13.00 |             2 |         5
(5 rows)


bms=# select * from Loans where interest_rate >= 7.50 AND loan_duration >= 5;
 loan_id | loan_amount | interest_rate | loan_duration | branch_id
---------+-------------+---------------+---------------+-----------
       1 |  5000000.50 |          7.50 |            15 |         1
       2 |  1000000.25 |          9.00 |             5 |         2
       4 |  2000000.50 |          8.00 |            10 |         4
(3 rows)
```

17. Retrieve ATM transaction where the transaction type is credit OR the amount is greater than 15000:

```
bms=# select * from ATM_Transaction where transaction_type = 'Withdrawal' OR amount > 15000.00;
 atm_transaction_id | atm_location | transaction_type |  amount  | date_of_transaction | card_id
--------------------+--------------+------------------+----------+---------------------+---------
                  1 | Delhi        | Withdrawal       | 10000.00 | 2024-09-15          |     201
                  2 | Mumbai       | Withdrawal       |  5000.00 | 2024-09-10          |     202
                  4 | Pune         | Deposit          | 20000.00 | 2024-09-25          |     204
                  5 | Bangalore    | Withdrawal       | 12000.00 | 2024-09-27          |     205
(4 rows)
```

18. Find customer whose Name start with ‘A’ and are from Kolkata:

```
bms=# select * from Customer_Info where Name LIKE ('A%') AND Address = 'Kolkata';
 customer_id |    name    | phone_number |        email          | address | date_of_birth | account_no | card_id | online_banking_id
-------------+------------+--------------+-----------------------+---------+---------------+------------+---------+-------------------
           3 | Amit Verma | 9988776655   | amit.verma@outlook.com | Kolkata | 1988-03-15    |       1003 |     203 |               303
(1 row)
```

19.Find employee whose position is not manager:

```
bms=# select * from Employee where position not in ('Manager');
 employee_id | first_name | last_name | position | department | branch_id
-------------+------------+-----------+----------+------------+-----------
           2 | Ramesh     | Singh     | Clerk    | Deposits   |         2
           3 | Anita      | Gupta     | Officer  | Customer   |         3
           5 | Rajesh     | Sharma    | Clerk    | Cards      |         5
(3 rows)
```

20. Find a payment amount not between 10000 and 20000 from loan payment table:

```
oms=# select * from Loans_Payment;
 payment_id | payment_amount | payment_date | loan_id | customer_id
------------+----------------+--------------+---------+-------------
          1 |       50000.00 | 2024-09-10   |       1 |           1
          2 |       20000.25 | 2024-09-15   |       2 |           2
          3 |       15000.75 | 2024-09-20   |       3 |           3
          4 |       25000.00 | 2024-09-25   |       4 |           4
          5 |        1000.50 | 2024-09-30   |       5 |           5
(5 rows)


oms=# select payment_id,payment_amount from Loans_Payment where payment_amount not between 10000.00 and 20000.00;
 payment_id | payment_amount
------------+----------------
          1 |       50000.00
          2 |       20000.25
          4 |       25000.00
          5 |        1000.50
(4 rows)
```

Aggregate Functions – (SUM, AVG, COUNT, MIN, MAX):

21. Find the average insurance amount of all Customer_ID:

```
bms=# select * from Insurance;
 insurance_id | insurance_type | insurance_amount | customer_id
--------------+----------------+------------------+-------------
            1 | Health         |        500000.25 |           1
            2 | Life           |       1000000.00 |           2
            3 | Vehical        |        200000.75 |           3
            4 | Home           |       1500000.50 |           4
            5 | Health         |        300000.25 |           5
(5 rows)


bms=# select AVG(Insurance_Amount) AS Average_Insurance FROM Insurance;
  average_insurance
---------------------
 700000.35000000000
(1 row)
```

22.Count the number of Branch IDs from each Branch :

```
bms=# select Branch_Name, COUNT(*)
bms-# as Branch_Count
bms-# from Branch
bms-# group by Branch_Name
bms-# order by Branch_Count
bms-# desc
bms-# ;
```

```
 branch_name | branch_count
-------------+--------------
 HDFC        |            2
 SBI         |            2
 ICICI       |            2
(3 rows)
```

23. Find the minimum account balance in the account table:

```
bms=# select MIN(Balance) AS Min_Balance from Account;
 min_balance
--------------
     10000.00
(1 row)
```

24.Find the maximum loan amount from the loans table

```
bms=# select MAX(Loan_Amount) AS MAX_Loan_Amount from Loans;
 max_loan_amount
-----------------
       5000000.50
(1 row)
```

25.Calculate total deposit amount from deposit table:

```
bms=# select * from Depositor;
 depositor_id | deposit_type | deposit_amount | date_of_deposit | customer_id
--------------+--------------+----------------+-----------------+-------------
            1 | Fixed        |          50000 | 2023-06-30      |           1
            2 | Recurring    |          20000 | 2023-08-10      |           2
            3 | Fixed        |         150000 | 2023-12-01      |           3
            4 | Savings      |          10000 | 2023-04-15      |           4
            5 | Fixed        |          75000 | 2023-02-20      |           5
(5 rows)


bms=# select sum(deposit_amount) as total_deposit from depositor;
 total_deposit
---------------
        305000
(1 row)
```

ORDER BY & GROUP BY :

26.Sort loan type from loan table in ascending order:

```
bankmanage=# SELECT loan_id,interest_rate, loan_amount, loan_duration FROM l
oans ORDER BY loan_duration ASC;
 loan_id | interest_rate | loan_amount | loan_duration
---------+---------------+-------------+---------------
       5 |         13.00 |   300000.00 |             2
       3 |         12.00 |   500000.75 |             3
       2 |          9.00 |  1000000.25 |             5
       4 |          8.00 | 20000000.50 |            10
       1 |          7.50 |  5000000.50 |            15
(5 rows)


bankmanage=# ALTER TABLE loans ADD loan_type VARCHAR(50);
ALTER TABLE
```

```
bankmanage=# INSERT INTO loans (loan_id, loan_type, loan_amount, interest_rate, loan_duration) VALUES (1, 'Home Loan', 5
000000.50, 7.50, 15), (2, 'Car Loan', 1000000.25, 9.00, 5), (3, 'Personal Loan', 500000.75, 12.00, 3), (4, 'Education Lo
an', 2000000.50, 8.00, 10), (5, 'Business Loan', 300000.00, 13.00, 3);
INSERT 0 5
bankmanage=# select * from loans;
 loan_id |   loan_type    | loan_amount | interest_rate | loan_duration
---------+----------------+-------------+---------------+---------------
       1 | Home Loan      |  5000000.50 |          7.50 |            15
       2 | Car Loan       |  1000000.25 |          9.00 |             5
       3 | Personal Loan  |   500000.75 |         12.00 |             3
       4 | Education Loan |  2000000.50 |          8.00 |            10
       5 | Business Loan  |   300000.00 |         13.00 |             3
(5 rows)
```

27.Order Loan by ascending order for interest rate:

```
ankmanage=#  SELECT loan_id, loan_type, loan_amount, interest_rate FROM loa
s ORDER BY interest_rate ASC;
loan_id |   loan_type    | loan_amount | interest_rate
--------+----------------+-------------+---------------
      1 | Home Loan      |  5000000.50 |          7.50
      4 | Education Loan |  2000000.50 |          8.00
      2 | Car Loan       |  1000000.25 |          9.00
      3 | Personal Loan  |   500000.75 |         12.00
      5 | Business Loan  |   300000.00 |         13.00
5 rows)
```

28.Group by borrowers by loan types and count how many borrowers are there in each type:

```
bankmanage=# SELECT l.loan_type, COUNT(b.borrower_id) AS total_borrowers FROM borrower b JOIN loans l ON b.loan_id = l.loan_id GROUP BY l.loan_type;
   loan_type    | total_borrowers
----------------+-----------------
 Personal Loan  |               1
 Car Loan       |               1
 Business Loan  |               1
 Home Loan      |               1
 Education Loan |               1
(5 rows)
```

JOINS -(NATURAL JOIN, INNER JOIN, OUTER JOIN, FULL JOIN, RIGHT JOIN, LEFT JOIN):

29. NATURAL JOINS -

Retrieve all the transaction details, including information about the cards and their respective transactions:

```
bms=# SELECT *
bms-# FROM ATM_Transactions NATURAL JOIN Cards;
 card_id | atm_transaction_id | atm_location | transaction_type |  amount  | date_of_transaction | card_type | expiry_date | cvv | account_no
---------+--------------------+--------------+------------------+----------+---------------------+-----------+-------------+-----+------------
     201 |                  1 | Delhi        | Withdrawal       | 10000.00 | 2024-09-15          | Debit     | 2026-05-31  | 123 |       1001
     202 |                  2 | Mumbai       | Withdrawal       |  5000.00 | 2024-09-10          | Credit    | 2025-12-31  | 456 |       1002
     203 |                  3 | Kolkata      | Deposit          | 15000.00 | 2024-09-22          | Debit     | 2024-11-30  | 789 |       1003
     204 |                  4 | Pune         | Deposit          | 20000.00 | 2024-09-25          | Debit     | 2027-03-31  | 321 |       1004
     205 |                  5 | Bangalore    | Withdrawal       | 12000.00 | 2024-09-27          | Credit    | 2026-10-31  | 654 |       1005
(5 rows)
```

30. INNER JOINS-

Get all employee details along with their branch details :

```
bms=# select e.employee_id,e.first_name,b.branch_name,b.branch_location
bms-# from employee e
bms-# inner join branch b on e.branch_id=b.branch_id;
 employee_id | first_name | branch_name | branch_location
-------------+------------+-------------+-----------------
           1 | Suresh     | HDFC        | Delhi
           2 | Ramesh     | ICICI       | Bangalore
           3 | Anita      | ICICI       | Kolkata
           4 | Sunita     | HDFC        | Mumbai
           5 | Rajesh     | SBI         | Delhi
(5 rows)
```

31. LEFT JOIN:

Get all customers and their account details:

```
bms=# select e.employee_id,e.first_name,b.branch_name
bms-# from employee e
bms-# left join branch b on e.branch_id=b.branch_id;
 employee_id | first_name | branch_name
-------------+------------+-------------
           1 | Suresh     | HDFC
           2 | Ramesh     | ICICI
           3 | Anita      | ICICI
           4 | Sunita     | HDFC
           5 | Rajesh     | SBI
(5 rows)
```

32.RIGHT JOIN:

Get all accounts and their transaction information

```
bms=# select a.account_type,t.transaction_type,t.amount
bms-# from account a
bms-# right join transaction_info t on a.account_no=t.account_no;
 account_type | transaction_type |  amount
--------------+------------------+----------
 Savings      | Credit           | 10000.00
 Fixed        | Credit           | 15000.00
 Recurring    | Debit            |  8000.00
 Savings      | Credit           | 20000.00
(4 rows)
```

33. FULL OUTER JOIN

Get all customers and their associated cards, including unmatched customers and cards

```
bankmanage=# SELECT c.customer_id, c.name, cd.card_id, cd.card_number FROM customer_info c FULL OUTER JOIN cards cd ON c.customer_id = cd.customer_id;
 customer_id |     name      | card_id |   card_number
-------------+---------------+---------+------------------
           1 | Rahul Sharma  |     201 | 1234567812345678
           2 | Priya Gupta   |     202 | 2345678923456789
           3 | Amit Verma    |     203 | 3456789034567890
           4 | Riya Sen      |     204 | 4567890145678901
           5 | Manish Singh  |     205 | 5678901256789012
(5 rows)
```

Altering Keys (PRIMARY / FOREIGN) & Constraints:

34. Add Primary Key Constraint to the Cards Table:

```
bankmanage=# ALTER TABLE cards ADD CONSTRAINT PK_Cards PRIMARY KEY (card_id);
ALTER TABLE
bankmanage=# \d
                    List of relations
 Schema |             Name              |   Type   |  Owner
--------+-------------------------------+----------+----------
 public | atm_transactions              | table    | postgres
 public | cards                         | table    | postgres
 public | customer_info                 | table    | postgres
 public | depositor                     | table    | postgres
 public | depositor_depositor_id_seq    | sequence | postgres
(5 rows)


bankmanage=# \d cards
                         Table "public.cards"
   Column    |          Type          | Collation | Nullable | Default
-------------+------------------------+-----------+----------+---------
 card_id     | integer                |           | not null |
 card_type   | character varying(10)  |           |          |
 expiry_date | date                   |           |          |
 cvv         | integer                |           |          |
 account_no  | integer                |           |          |
Indexes:
    "pk_cards" PRIMARY KEY, btree (card_id)
```

35. Remove the foreign key constraint from loans table:

```
bankmanage=# SELECT conname FROM pg_constraint WHERE conrelid = 'loans'::regclass AND contype = 'f';
      conname
-------------------
 fk_loans_customer
(1 row)


bankmanage=# ALTER TABLE loans DROP CONSTRAINT constraint_name;
ERROR:  constraint "constraint_name" of relation "loans" does not exist
bankmanage=# ALTER TABLE loans DROP CONSTRAINT fk_loans_customer;
ALTER TABLE
bankmanage=#
```

36. Add a unique constraint To the “email” column In the customer table:

```
bankmanage=# ALTER TABLE loans DROP CONSTRAINT fk_loans_customer;
ALTER TABLE
bankmanage=# ALTER TABLE customer_info ADD CONSTRAINT unique_email UNIQUE(email);
ALTER TABLE
```

```
bankmanage=# \d customer_info
                    Table "public.customer_info"
      Column       |          Type          | Collation | Nullable | Default
-------------------+------------------------+-----------+----------+---------
 customer_id       | integer                |           | not null |
 name              | character varying(100) |           |          |
 phone_number      | character varying(15)  |           |          |
 email             | character varying(100) |           |          |
 address           | character varying(100) |           |          |
 date_of_birth     | date                   |           |          |
 account_no        | integer                |           |          |
 card_id           | integer                |           |          |
 online_banking_id | integer                |           |          |
Indexes:
    "customer_info_pkey" PRIMARY KEY, btree (customer_id)
    "unique_email" UNIQUE CONSTRAINT, btree (email)
Referenced by:
```

ALIASES :

37. Use aliases to retrieve the first_name and last_name of employees:

```
bms=# select * from Employee;
 employee_id | first_name | last_name | postion | department | branch_id
-------------+------------+-----------+---------+------------+-----------
           1 | Suresh     | Kumar     | Manager | Loans      |         1
           2 | Ramesh     | Singh     | Clerk   | Deposits   |         2
           3 | Anita      | Gupta     | Officer | Customer   |         3
           4 | Sunita     | Reddy     | Manager | Accounts   |         4
           5 | Rajesh     | Sharma    | Clerk   | Cards      |         5
(5 rows)


bms=# select first_name as fname,last_name as lname from Employee;
 fname  | lname
--------+--------
 Suresh | Kumar
 Ramesh | Singh
 Anita  | Gupta
 Sunita | Reddy
 Rajesh | Sharma
(5 rows)
```

38. Aliases for branch location as branch place:

```
bms=# select branch_location AS "branch_place" from Branch;
 branch_place
--------------
 Delhi
 Bangalore
 Kolkata
 Mumbai
 Delhi
(5 rows)
```

VIEWS :

39. Create a View Combining Loans and Branch Information:

```
bms=# create view loans_details as
bms-# select l.loan_amount, l.interest_rate, l.loan_duration,b.branch_name, b.branch_location
bms-# from loans l
bms-# join branch b on l.branch_id = b.branch_id;
CREATE VIEW
```

40. Display a create view:

```
bms=# select * from loans_details;
 loan_amount | interest_rate | loan_duration | branch_name | branch_location
-------------+---------------+---------------+-------------+-----------------
  5000000.50 |          7.50 |            15 | HDFC        | Delhi
  1000000.25 |          9.00 |             5 | ICICI       | Bangalore
   500000.75 |         12.00 |             3 | ICICI       | Kolkata
  2000000.50 |          8.00 |            10 | HDFC        | Mumbai
   300000.00 |         13.00 |             2 | SBI         | Delhi
(5 rows)
```

41. Create a View for Minimum, Maximum, and Average Loan Amount:(loans table)

```
bankmanage=# CREATE VIEW Loan_Stats_View AS SELECT MIN(loan_amount) AS "Mini
mum Loan Amount", MAX(loan_amount) AS "Maximum Loan Amount", AVG(loan_amount
) AS "Average Loan Amount" FROM loans;
CREATE VIEW
bankmanage=# SELECT * FROM Loan_Stats_View;
 Minimum Loan Amount | Maximum Loan Amount | Average Loan Amount
---------------------+---------------------+----------------------
           300000.00 |         20000000.50 | 5360000.400000000000
(1 row)
```

42. Drop a view loans_details :

```
bms=# drop view loans_details;
DROP VIEW
bms=# select * from loans_details;
ERROR:  relation "loans_details" does not exist
LINE 1: select * from loans_details;
                      ^
```

DATA CONTROL LANGUAGE (Grant / Revoke) :

43 . Grant SELECT Permission on the “Accounts” Table to new user:

```
bms=# \z account;
                                Access privileges
 Schema |  Name   | Type  | Access privileges | Column privileges | Policies
--------+---------+-------+-------------------+-------------------+----------
 public | account | table |                   |                   |
(1 row)

bms=# GRANT SELECT ON TABLE account TO empp1;
GRANT
bms=# \z account
                                   Access privileges
 Schema |  Name   | Type  |     Access privileges      | Column privileges | Policies
--------+---------+-------+----------------------------+-------------------+----------
 public | account | table | postgres=arwdDxtm/postgres+|                   |
        |         |       | empp1=r/postgres           |                   |
(1 row)
```

44 . Revoke INSERT Permission on the Transaction Information Table :

```
bms=# \z transaction_info;
                                   Access privileges
 Schema |       Name       | Type  |     Access privileges      | Column privileges | Policies
--------+------------------+-------+----------------------------+-------------------+----------
 public | transaction_info | table | postgres=arwdDxtm/postgres+|                   |
        |                  |       | empp2=a/postgres           |                   |
(1 row)


bms=# REVOKE INSERT ON TABLE Transaction_Info FROM empp2;
REVOKE
bms=# \z transaction_info;
                                   Access privileges
 Schema |       Name       | Type  |     Access privileges      | Column privileges | Policies
--------+------------------+-------+----------------------------+-------------------+----------
 public | transaction_info | table | postgres=arwdDxtm/postgres |                   |
(1 row)
```

TRANSACTION CONTROL LANGUAGE –
(Commit , Rollback , Savepoint, Set Transaction) :

45. Write a query that starts new transaction to update particular customer's Insurance amount :

```
bms=# select * from Insurance;
 insurance_id | insurance_type | insurance_amount | customer_id
--------------+----------------+------------------+-------------
            1 | Health         |        500000.25 |           1
            2 | Life           |       1000000.00 |           2
            3 | Vehical        |        200000.75 |           3
            4 | Home           |       1500000.50 |           4
            5 | Health         |        300000.25 |           5
(5 rows)


bms=# BEGIN TRANSACTION;
BEGIN
bms=*# UPDATE Insurance
bms-*# SET Insurance_Amount = 700000.00
bms-*# WHERE Insurance_ID = 5;
UPDATE 1
bms=*# select * from Insurance;
 insurance_id | insurance_type | insurance_amount | customer_id
--------------+----------------+------------------+-------------
            1 | Health         |        500000.25 |           1
            2 | Life           |       1000000.00 |           2
            3 | Vehical        |        200000.75 |           3
            4 | Home           |       1500000.50 |           4
            5 | Health         |        700000.00 |           5
(5 rows)
```

46 . Demonstrate a successfully COMMIT to finalise a transaction’s changes :

```
bms=# BEGIN;
BEGIN
bms=*# insert into customer_info values (6 , 'Rohan Birla' , '9516243875', 'rohan.birla@gmail.com', 'Mumbai', '1993-02-21', 1006, 206, 306)
INSERT 0 1
bms=*# COMMIT;
COMMIT
bms=# select * from Customer_Info;
 customer_id |     name      | phone_number |         email          |  address  | date_of_birth | account_no | card_id | online_banking_id
-------------+---------------+--------------+------------------------+-----------+---------------+------------+---------+-------------------
           2 | Priya Gupta   | 9123456789   | priya.gupta@yahoo.com  | Mumbai    | 1992-07-11    |       1002 |     202 |               302
           3 | Amit Verma    | 9988776655   | amit.verma@outlook.com | Kolkata   | 1988-03-15    |       1003 |     203 |               303
           4 | Riya Sen      | 9898989898   | riya.sen@gmail.com     | Bangalore | 1995-01-29    |       1004 |     204 |               304
           5 | Manish Singh  | 9765432109   | manish.singh@gmail.com | Pune      | 1991-08-22    |       1005 |     205 |               305
           1 | Rahul Sharma  | 9876543210   | rahul.sharma@gamil.com | Pune      | 1990-05-23    |       1001 |     201 |               301
           6 | Rohan Birla   | 9516243875   | rohan.birla@gmail.com  | Mumbai    | 1993-02-21    |       1006 |     206 |               306
(6 rows)
```

47 . Rollback a transaction that encountered an error during execution

```
bms=# begin;
BEGIN
bms=*# insert into branch values (1 , 'PNB', 'Gujarat');
ERROR:  duplicate key value violates unique constraint "branch_pkey"
DETAIL:  Key (branch_id)=(1) already exists.
bms=!# COMMIT;
ROLLBACK
```

48 . ROLLBACK a deleted Transaction :

```
bms=# BEGIN TRANSACTION;
BEGIN
bms=*# DELETE FROM Employee where employee_id = 5;
DELETE 1
bms=*# select * from Employee;
 employee_id | first_name | last_name | position | department | branch_id
-------------+------------+-----------+----------+------------+-----------
           1 | Suresh     | Kumar     | Manager  | Loans      |         1
           2 | Ramesh     | Singh     | Clerk    | Deposits   |         2
           3 | Anita      | Gupta     | Officer  | Customer   |         3
           4 | Sunita     | Reddy     | Manager  | Accounts   |         4
(4 rows)


bms=*# ROLLBACK;
ROLLBACK
bms=# select * from Employee;
 employee_id | first_name | last_name | position | department | branch_id
-------------+------------+-----------+----------+------------+-----------
           1 | Suresh     | Kumar     | Manager  | Loans      |         1
           2 | Ramesh     | Singh     | Clerk    | Deposits   |         2
           3 | Anita      | Gupta     | Officer  | Customer   |         3
           4 | Sunita     | Reddy     | Manager  | Accounts   |         4
           5 | Rajesh     | Sharma    | Clerk    | Cards      |         5
(5 rows)
```

49 . Use a SAVEPOINT to createa point of rollback within a transaction :

```
bms=#  begin;
BEGIN
bms=*# savepoint sp;
SAVEPOINT
bms=*# delete from cards where card_id = 205;
DELETE 1
bms=*# delete from cards where card_id = 204;
DELETE 1
bms=*# delete from cards where card_id = 203;
DELETE 1
bms=*# ROLLBACK TO SP;
ROLLBACK
bms=*# select * from cards;
 card_id | card_type | expiry_date | cvv | account_no
---------+-----------+-------------+-----+------------
     201 | Debit     | 2026-05-31  | 123 |       1001
     202 | Credit    | 2025-12-31  | 456 |       1002
     203 | Debit     | 2024-11-30  | 789 |       1003
     204 | Debit     | 2027-03-31  | 321 |       1004
     205 | Credit    | 2026-10-31  | 654 |       1005
(5 rows)
```

INDEXING :

50 . Create an index on the account_no column in the Account table :

```
bms=# create index idx_account_no on account(account_no);
CREATE INDEX
bms=# explain analyze select * from account where account_no = 5;
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on account  (cost=0.00..1.06 rows=1 width=150) (actual time=0.037..0.037 rows=0 loops=1)
   Filter: (account_no = 5)
   Rows Removed by Filter: 5
 Planning Time: 6.173 ms
 Execution Time: 1.476 ms
(5 rows)
```

51 . Create a Composite Index on the loan_amount and loan_duration columns of the loans table :

```
bms=# create index idx_loans on loans(loan_amount, loan_duration);
CREATE INDEX
```

52 . Drop the index idx_account_no :

```
DROP INDEX
bms=# explain analyze select * from account where account_no = 5;
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on account  (cost=0.00..1.06 rows=1 width=150) (actual time=0.016..0.016 rows=0 loops=1)
   Filter: (account_no = 5)
   Rows Removed by Filter: 5
 Planning Time: 0.325 ms
 Execution Time: 0.036 ms
(5 rows)
```

Conclusion

A Bank Management System (BMS) built using a Database Management System (DBMS) is designed to significantly enhance the operations of a bank by addressing the key challenges faced by traditional banking systems. Here's a detailed explanation of how such a system improves data security, reduces manual effort, ensures faster and more accurate banking operations, and provides an enhanced customer experience.Increased Efficiency: Automated operations reduce manual labour and errors.Improved Accuracy: Real-time updates and consistent data eliminate discrepancies.Better Security: Enhanced encryption and access controls protect sensitive information.Faster Transactions: The system processes tasks like fund transfers and loan approvals instantly.Enhanced Customer Satisfaction: Faster, more reliable service across all channels increases customer trust and loyalty.

Thank You