BANK MANAGEMENT SYSTEM

A Comprehensive Database Management Project Report

Session - 2023-24

Prepared By:
Aman Kumar
UEN:- 2022UG000010

TABLE OF CONTENTS

Vidyashilp Bank Management System

Problem Statement-

Vidyashilp Group plans to open a Bank with several branches in different part of the city to provide services to people. So, developing a comprehensive Database Management System (DBMS) for a Bank Management System, aimed at efficiently managing customer accounts, transactions, employee records, and financial operations, ensuring secure and seamless banking services.

Requirements:

1. **Bank Branches:**
   - Storing information about each bank branch, including branch code, location, contact details, and any other relevant information.
2. **Customer Details:**
   - Storing and managing customer information, including unique customer ID, name, contact details, address, and other relevant personal details.
3. **Employee Details:**
   - Establish a table for employee details, capturing information such as employee ID, name, position, contact information, and possibly their assigned branch.
4. **Loan Details:**
   - Create a comprehensive table to store loan details, including loan ID, type, amount, duration, interest rate, and any other relevant information.
5. **Customer associated with Bank/Loan Manager:**
   - Establish relationships between the customer table and the employee table, indicating which employee (potentially a bank manager or loan officer) is associated with each customer.
   - Account type - Saving/Current:
   - Include a field in the customer table to denote the type of account each customer holds, whether it's a savings account or a current account.
6. **Loan Origination - Types, Duration, Payment Schedules:**
   - Extend the loan details table to include information about loan types, duration, and payment schedules. Ensure that the database can accommodate various loan products and their specific attributes.
7. **Transactions:**
   - Develop a transaction table to record all banking transactions, including deposits, withdrawals, transfers, loan repayments, etc. This table should capture transaction ID, date, type, amount, and relevant customer and account details.

Entities Required-

- ➢ Branch
- ➢ Customer
- ➢ Employee
- ➢ Loan
- ➢ Account
- ➢ Transaction
- ➢ Payment

Customer Management:

- Capture customer information (name, address, contact details).
- Maintain customer accounts.

Account Management:

- Support different types of accounts (e.g., savings, checking).
- Track account balance and transaction history.

Transaction Processing:

- Enable deposits, withdrawals, and fund transfers.
- Record transaction details (amount, type).

Loan Management:

- Manage loan applications and approvals.
- Track loan details (amount, interest rate, duration).

Employee Management:

- Maintain employee information.
- Assign roles and responsibilities.

Attributes-

1. **Branch**
   - ➢ Branch Code (Primary Key)
   - ➢ Branch Name
   - ➢ Branch Location

2. **Customer**
   - ➢ Customer ID (Primary Key)
   - ➢ Name
   - ➢ DOB
   - ➢ Age
   - ➢ Gender
   - ➢ Contact
   - ➢ Address

3. **Employee**
    - ➢ Employee_Reference
    - ➢ Name
    - ➢ DOB
    - ➢ Age
    - ➢ Gender
    - ➢ Contact
    - ➢ Address
    - ➢ Salary
    - ➢ Position

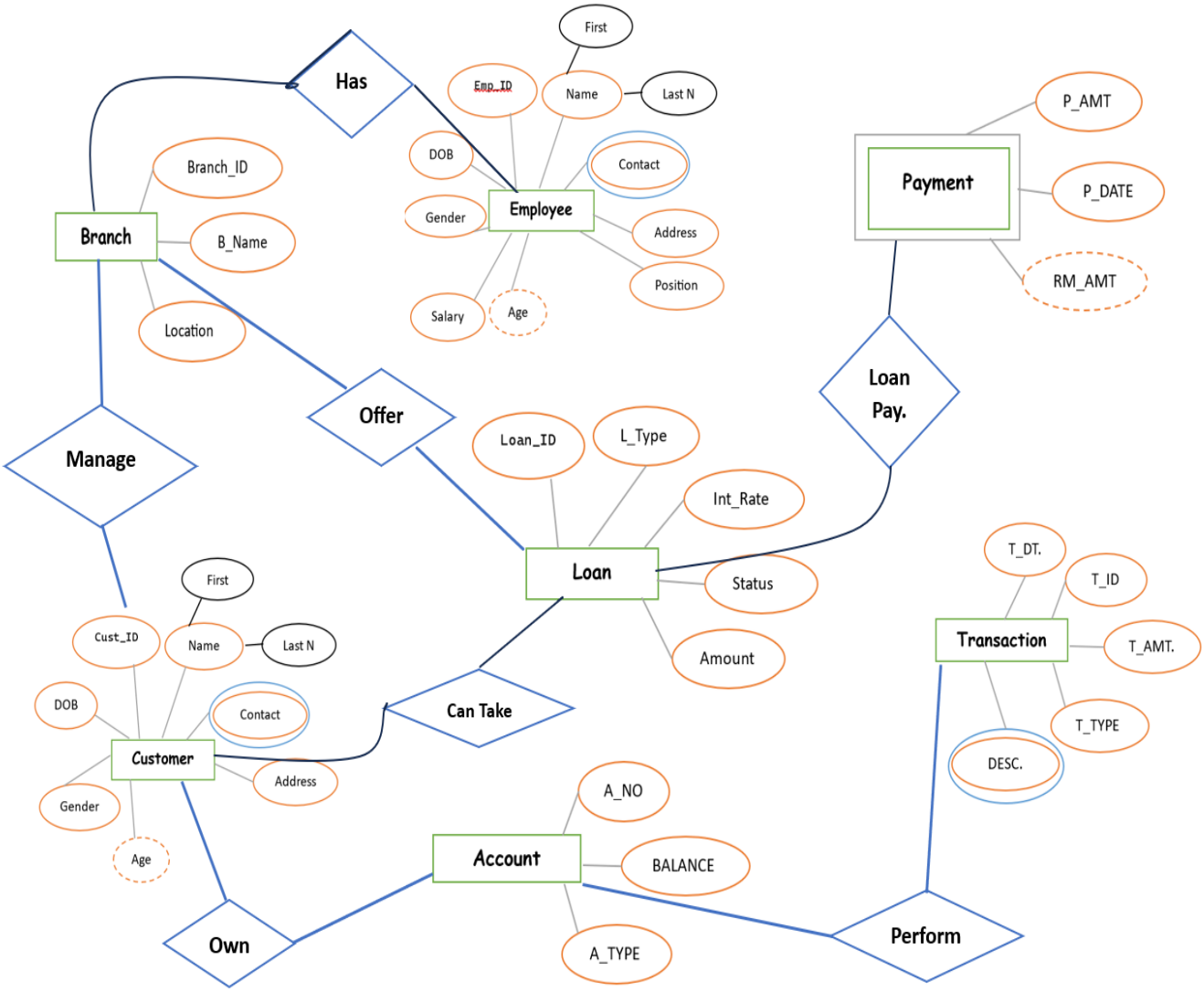
4. **Loan**
    - ➢ Loan Type
    - ➢ Status
    - ➢ Interest Rate
    - ➢ Amount
    - ➢ Duration

5. **Account**
    - ➢ Account No.
    - ➢ Balance
    - ➢ Account Type

6. **Transaction**
    - ➢ Transaction Reference No
    - ➢ Transaction Date
    - ➢ Transaction Amount
    - ➢ Transaction Type
    - ➢ Description

7. **Repayment**
    - ➢ Payment Sequence No
    - ➢ Payment Amount
    - ➢ Payment Date
    - ➢ Remaining Amount

ER Diagram


Has
First
Emp_ID
Name
Last N
DOB
Contact
Gender
Employee
Address
Position
Salary
Age
Branch_ID
Branch
B_Name
Location
Payment
P_AMT
P_DATE
RM_AMT
Loan
Pay.
Offer
Manage
Loan_ID
L_Type
Int_Rate
Loan
Status
Amount
T_DT.
T_ID
Transaction
T_AMT.
T_TYPE
DESC.
First
Cust_ID
Name
Last N
DOB
Contact
Customer
Address
Gender
Age
Can Take
A_NO
Account
BALANCE
A_TYPE
Own
Perform

Applying Normalization

Entity: Branch

Attributes:

Branch_Code	Branch_Name	Branch_Loc

Functional Dependencies:

Branch_Code -> Branch_Name, Branch_Loc

Candidate Key- Branch_Code

1. First Normal Form (1NF):

1NF is already satisfied when all attributes have atomic (indivisible) values. In this case, each attribute (Branch_Code, Branch_Name, and Branch_Location) holds atomic values.

2. Second Normal Form (2NF):

2NF is achieved when there are no partial dependencies. Since there is only one candidate key (Branch_Code), and all attributes are fully functionally dependent on it, 2NF is satisfied.

3. Third Normal Form (3NF):

3NF is met when there are no transitive dependencies. In this case, Branch_Code is the only candidate key, and both Branch_Name and Branch_Loc is directly dependent on it. Therefore, 3NF is satisfied.

4. Boyce-Codd Normal Form (BCNF):

BCNF is satisfied when every non-trivial functional dependency is a super key. In the Branch entity, Branch_Code is the only candidate key, and it already satisfies the BCNF condition as it is a super key.

Final Table-

Branch_Code	Branch_Name	Branch_Loc

Entity: Customer

Cust_ID	Name	DOB	Gender	Contact	Address	Age

Functional Dependencies:

- Cust_ID -> Name, DOB, Gender, Contact
- DOB->Age
- Name-> Address

Candidate key- Cust_ID

1. First Normal Form (1NF):

'Contact' can contain multiple values (no atomicity), hence violation of 1NF. So, restructuring of the existing table is needed.

Updated Tables:

Cust_ID	Contact

Cust_ID	Name	DOB	Age	Gender	Address

Now, 1NF is Satisfied

2. Second Normal Form (2 NF):

2NF property is already satisfied as there is no partial dependency in the tables. Because there is just one candidate key.

3. Third Normal Form (3 NF):

- Cust_ID -> Name, DOB, Gender, Contact
- DOB->Age (Transitive Dependency, as DOB derives Age and DOB is a non-key attribute)
- Name-> Address (Transitive Dependency exist as Name is a non-key attribute)

As transitive dependency exists in the tables, we need to decompose it to eliminate the redundancy.

Creating Separate tables:

Customer_Contact:

Cust_ID	Contact

Age_Info:

Cust_ID	DOB	Age

Address_Info:

Cust_ID	Name	Address

Customer_Info:

Cust_ID	Name	DOB	Gender

Now, the tables satisfy 3NF.

4. Boyce-Codd Normal Form (BCNF):

- All the above tables follow BCNF as for all the tables in here, A → B, A is a super key

5. Lossless decomposition:

- Union of all the tables results in the final table
  R- {Cust_ID, Name, Contact, Age, Address, Gender}

- Intersection of all tables result in 'Cust_ID' which is a candidate key

Hence, it follows Lossless decomposition

Entity: Employee

Emp_Ref	Name	DOB	Gender	Contact	Address	Salary	Position

Functional Dependencies:

- Emp_Ref -> Name, Gender, Contact, Address, Position
- Position -> Salary
- DOB->Age

Candidate keys- Emp_Ref + DOB

1. First Normal Form (1NF):

'Contact' can contain multiple values (no atomicity), hence violation of 1NF. So, restructuring of the existing table is needed.

Updated tables:

Emp_Ref	Contact

Emp_Ref	Name	DOB	Age	Gender	Address	Salary	Position

Now, 1NF is Satisfied

2. Second Normal Form (2 NF):

Partial dependency exists in the tables as 'Age' is just derived from 'DOB' alone. So, this needs to eliminated. Therefore, there is a need to create a separate table for DOB and Age

Emp_Ref	Contact

DOB	Age

Emp_Ref	Name	DOB	Gender	Address	Position	Salary

3. Third Normal Form (3 NF):

- Position -> Salary (Transitive dependency exists here as 'Salary' is derived from 'Position' and 'Position' is not part of the Candidate Key. Hence, this dependency needs to be removed to avoid redundancy and anomalies by decomposing the tables.

Updated Tables-

Emp_Contact:

Emp_Ref	Contact

Emp_Age_info:

DOB	Age

Emp_Info:

Emp_Ref	Name	Gender	Address

Emp_Status:

Emp_Ref	Position	Salary

Emp_DOB_Rel:

Emp_Ref	DOB

4. Boyce-Codd Normal Form (BCNF):

The tables are in BCNF form as there is no such key attributes which are derived from a non-key attribute

5. Lossless Decomposition

The union of the all the decomposed tables results in the original table-

R1- {Emp_Ref, Contact}

R2- {Emp_Ref, DOB, Age}

R3- {Emp_Ref, Name, Gender, Address}

R4- {Emp_Ref, Position, Salary}

R5- {Emp_Ref, DOB}

R1 U R2 U R3 U R4 U R5 = {Emp_Ref, DOB, Age, Name, Address, Contact, Position, Salary}
Also, the intersection of all the decomposed tables is {EMP_Ref} and it's a part of Candidate key, hence it follows lossless decomposition.

Entity– Account

Cust_ID	Acc_No	Balance	Status	Account Type	Interest Rate

Functional Dependencies:

- Cust_ID, Acc_No. -> Account Type
- Account Type -> Status, Interest Rate, Balance

Candidate Keys: (Cust_ID + Acc_No)

1. First Normal Form (1NF):

'Account Type' can contain multiple values and it violates atomicity hence it needs to be separated from the original table.

Updated Tables:

Cust_Accounts:

Cust_ID	Acc_No	Account Type

Account_Details:

Account Type	Status	Interest Rate	Balance

Now, 1NF is Satisfied

As 'Status', 'Interest Rate', 'Balance' depend on a multivalued attribute 'Cust_ID' & 'Acc No'. Hence, there is a need of redefining the functional dependency.

Revised Functional Dependencies:

Customer_Accounts:

(Cust_ID, Acc_No) -> Account_Type

Account_Details:

Account_Type -> Status, Interest Rate, Balance

2. Second Normal Form (2 NF):

There is no partial dependency in both the tables, hence 2NF is already satisfied so there is no need of creating separate tables.

3. Third Normal Form (3 NF):

There is no transitive dependency in both the tables as all the non-key attributes are derived from key attributes.

4. Boyce-Codd Normal Form (BCNF):

The tables are already in BCNF as there is no key attribute derived from non-key attribute, hence no decomposition/modification in the tables.

5. Lossless Decomposition:

Both the tables are not following lossless property as the intersection is null, hence we need to introduce tables to establish the relation between tables.

Updated Tables:

T1- {Cust_ID, Acc_No}

T2- {Cust_ID, Acc_No, Account Type}

T3- {Acc_No, Account Type, Balance, Status, Interest Rate}

Now, the union of above tables give the original table and also the intersection is ‘Acc_No’ and it is a part of the candidate keys. Hence, now the tables are following lossless property.

Entity- Loan

Acc_No	Loan_type	Amount	Duration	Status	Interest Rate

Functional Dependencies:

- Acc_No -> Duration, Amount, Status
- Loan Type -> Interest rate

Candidate Keys- (Acc_No + Loan Type)

1. First Normal Form (1NF):

Loan Type acts a multivalued attribute here hence to follow atomicity in the table lets decompose the tables.

Acc_No	Loan_type

Acc_No	Duration	Amount	Status	Interest

2. Second Normal Form (2 NF):

Partial dependency exists in the table as 'Interest' just depends on 'Loan type', not on Acc_No, hence, this partial dependency needs to be removed.

Acc_No	Loan_type

Loan type	Interest

Acc_No	Duration	Amount	Status

3. Third Normal Form (3 NF):

There is no transitive dependency in the above tables, hence the tables are in 3NF.

4. Boyce-Codd Normal Form (BCNF):

The tables are already in BCNF as there are no such key attribute which are derived form non-key attributes.

5. Lossless Decomposition:

Currently the tables are not in lossless form as the union of all the tables are original table but the intersection is null, hence there is no proper relation between all the tables. So, adding certain attributes in the tables to establish relation.

T1- {Acc_No, Loan type}

T2- {Loan type, Interest}

T3- {Acc_No, Loan type, Duration, Amount, Status}

Now, the union is the original table and the intersection is ‘Loan type’ and its a part of candidate key.

Entity- Transaction

Transaction_Ref	Tr_date	Tr_amount	Tr_Type	Tr_description

Functional Dependencies:

- Transaction_Ref -> Tr_date, Tr_Amount, Tr_Type, Tr_desc

Candidate key- Transaction_Ref

1. First Normal Form (1NF):

All the attributes in the tale re following atomicity hence the table is already in 1NF.

2. Second Normal Form (2 NF):

There is no partial dependency in the table as there is only one key (Transaction_Ref) which can derive all other attributes. Hence, the table is in 2NF**.**

3. Third Normal Form (3 NF):

- Transaction_Ref -> Tr_date, Tr_Amount, Tr_Type

The tables are already in 3NF as there is no transtive dependency in the table

Transaction

Transaction_Ref	Tr_Date	Tr_Amount	Tr_type	Tr_Desc

4. Boyce-Codd Normal Form (BCNF):

The tables are already in BCNF as there is no such key attributes which are derived from non-key attributes.

Entity- Repayment

Paymen_seqno	Payment_Amt	Payment_Date	Remaining_Amt

Functional Dependency

- Payment_Seqno -> Payment _Amt, Payment_Date, Remaining_Amt

  Candidate key- Payment_Seqno

This table is following 1NF, 2NF, 3NF & BCNF.

Now, all the tables are following normalization.

Implementation

Tables with Constraints:

Branch Table:-

```
CREATE DATABASE BANK_MANAGEMENT_SYSTEM;
USE BANK_MANAGEMENT_SYSTEM;

-- Branch Table
CREATE TABLE Branch (
  Branch_Code INT PRIMARY KEY CHECK(length(Branch_Code)=4),     -- Branch code must be 4 digit
  Branch_Name VARCHAR(50) NOT NULL,
  Branch_Loc VARCHAR(100) NOT NULL
);
```

Customer Tables: -

```
CREATE TABLE Age_Info (
  Cust_ID INT NOT NULL,
  DOB DATE NOT NULL,
  Age INT NOT NULL,
  PRIMARY KEY (Cust_ID, DOB),
  FOREIGN KEY (Cust_ID) REFERENCES Customer_Info(Cust_ID)
);
    CREATE TABLE Address_Info (
      Cust_ID INT,
      Name VARCHAR(50) NOT NULL,
      Address VARCHAR(100) NOT NULL,
      PRIMARY KEY (Name),
      INDEX fk_cust_id (Cust_ID), -- Index on the foreign key column
      FOREIGN KEY (Cust_ID) REFERENCES Customer_Info(Cust_ID),
      CONSTRAINT fk_name_cust_id FOREIGN KEY (Name, Cust_ID) REFERENCES Customer_Info(Name, Cust_ID)
);
-- Add index on the referenced columns in Customer_Info
CREATE INDEX idx_customer_info_name_cust_id ON Customer_Info (Name, Cust_ID);
```

```
-- Customer Tables
CREATE TABLE Customer_Info (
  Cust_ID INT PRIMARY KEY CHECK(length(Cust_ID)=6),        -- Cust_Id must be 6 digit
  Name VARCHAR(50) NOT NULL,
  DOB DATE NOT NULL,
  Gender CHAR(1) NOT NULL
);
-- Creating a relation between Customer table and Branch table
ALTER TABLE Customer_Info
ADD COLUMN Branch_Code INT,
ADD FOREIGN KEY (Branch_Code) REFERENCES Branch(Branch_Code);
```

Employee Tables: -

```
-- Employee Tables
CREATE TABLE Emp_Info (
  Emp_Ref INT PRIMARY KEY,
  Name VARCHAR(50) NOT NULL,
  Gender CHAR(1) NOT NULL,
  Address VARCHAR(100) NOT NULL,
  Branch_Code int,
  FOREIGN KEY (Branch_Code) REFERENCES Branch(Branch_Code)
);
CREATE TABLE Emp_Contact (
  Emp_Ref INT PRIMARY KEY,
  Contact CHAR(10) NOT NULL,
  FOREIGN KEY (Emp_Ref) REFERENCES Emp_Info(Emp_Ref)
);
CREATE TABLE Emp_Age_Info (
Emp_ref INT NOT NULL,
  DOB DATE,
  Age INT NOT NULL,
  PRIMARY KEY (Emp_Ref, DOB),
  FOREIGN KEY (Emp_Ref) REFERENCES Emp_Contact(Emp_Ref)
);
CREATE TABLE Emp_Status (
  Emp_Ref INT PRIMARY KEY,
  Position VARCHAR(50) NOT NULL,
  Salary DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (Emp_Ref) REFERENCES Emp_Info(Emp_Ref)
);
```

Account Tables: -

```
-- Account Tables
CREATE TABLE Cust_Accounts (
  Cust_ID INT,
  Acc_No INT,
  Account_Type VARCHAR(20) NOT NULL,
  PRIMARY KEY (Cust_ID, Acc_No),
  FOREIGN KEY (Cust_ID) REFERENCES Customer_Info(Cust_ID)
);
CREATE TABLE Account_Details (
      Account_Type VARCHAR(20) PRIMARY KEY,
      Status VARCHAR(20) NOT NULL,
      Interest_Rate DECIMAL(5, 2) NOT NULL,
      Balance DECIMAL(12, 2) NOT NULL
    );
```

Loan Tables :-

```
-- Loan Tables
CREATE TABLE Loan_Type (
    Loan_Type VARCHAR(20) PRIMARY KEY,
    Interest DECIMAL(5, 2) NOT NULL
    );

CREATE TABLE Loan_Details (
  Acc_No INT,
  Loan_Type VARCHAR(20),
  Duration INT NOT NULL,
  Amount DECIMAL(12, 2) NOT NULL,
  Status VARCHAR(20) NOT NULL,
  PRIMARY KEY (Acc_No, Loan_Type),
  FOREIGN KEY (Loan_Type) REFERENCES Loan_Type(Loan_Type),
  FOREIGN KEY (Acc_No) REFERENCES Cust_Accounts(Acc_No)
);
-- Adding index on referenced column in Loan_Type
CREATE INDEX idx_loan_type_loan_type ON Loan_Type (Loan_Type);

-- Adding index on referenced column in Cust_Accounts
CREATE INDEX idx_cust_accounts_acc_no ON Cust_Accounts (Acc_No);
```

Transaction Table: -

```
-- Transaction Table
CREATE TABLE Transaction (
  Transaction_Ref INT PRIMARY KEY,
  Tr_Date DATE NOT NULL,
  Tr_Amount DECIMAL(12, 2) NOT NULL,
  Tr_Type VARCHAR(20) NOT NULL,
  Tr_Desc VARCHAR(100) NOT NULL,
  Cust_ID INT NULL,
  Acc_No INT NULL,
  FOREIGN KEY (Cust_ID, Acc_No) REFERENCES Cust_Accounts(Cust_ID, Acc_No)
);
```

Loan Repayment Table :-

```
-- Loan Repayment Table
CREATE TABLE Repayment (
  Payment_Seqno INT PRIMARY KEY,
  Payment_Amt DECIMAL(12, 2) NOT NULL,
  Payment_Date DATE NOT NULL,
  Remaining_Amt DECIMAL(12, 2) NOT NULL,
  Acc_No INT NOT NULL,
  FOREIGN KEY (Acc_No) REFERENCES Loan_Details(Acc_No)
);
```

Structure of each table: -

Field	Type	Null	Key	Default	Extra
Cust_ID	int	YES	MUL	NULL	
Name	varchar(50)	NO	PRI	NULL	
Address	varchar(100)	NO		NULL	

Field	Type	Null	Key	Default	Extra
Branch_Code	int	NO	PRI	NULL	
Branch_Name	varchar(50)	NO		NULL	
Branch_Loc	varchar(100)	NO		NULL	

Field	Type	Null	Key	Default	Extra
Cust_ID	int	NO	PRI	NULL	
DOB	date	NO	PRI	NULL	
Age	int	NO		NULL	

Field	Type	Null	Key	Default	Extra
Cust_ID	int	NO	PRI	NULL	
Acc_No	int	NO	PRI	NULL	
Account_Type	varchar(20)	NO		NULL	

Field	Type	Null	Key	Default	Extra
Cust_ID	int	NO	PRI	NULL	
Name	varchar(50)	NO	MUL	NULL	
DOB	date	NO		NULL	
Gender	char(1)	NO		NULL	
Branch_Code	int	YES	MUL	NULL	

Field	Type	Null	Key	Default	Extra
Emp_Ref	int	NO	PRI	NULL	
Name	varchar(50)	NO		NULL	
Gender	char(1)	NO		NULL	
Address	varchar(100)	NO		NULL	
Branch_Code	int	YES	MUL	NULL	

Field	Type	Null	Key	Default	Extra
Transaction_Ref	int	NO	PRI	NULL	
Tr_Date	date	NO		NULL	
Tr_Amount	decimal(12,2)	NO		NULL	
Tr_Type	varchar(20)	NO		NULL	
Tr_Desc	varchar(100)	NO		NULL	
Cust_ID	int	YES	MUL	NULL	
Acc_No	int	YES		NULL	

Field	Type	Null	Key	Default	Extra
Acc_No	int	NO	PRI	NULL	
Loan_Type	varchar(20)	NO	PRI	NULL	
Duration	int	NO		NULL	
Amount	decimal(12,2)	NO		NULL	
Status	varchar(20)	NO		NULL	

Field	Type	Null	Key	Default	Extra
Payment_Seqno	int	NO	PRI	NULL	
Payment_Amt	decimal(12,2)	NO		NULL	
Payment_Date	date	NO		NULL	
Remaining_Amt	decimal(12,2)	NO		NULL	
Acc_No	int	NO	MUL	NULL	

Field	Type	Null	Key	Default	Extra
Account_Type	varchar(20)	NO	PRI	NULL	
Status	varchar(20)	NO		NULL	
Interest_Rate	decimal(5,2)	NO		NULL	
Balance	decimal(12,2)	NO		NULL	

Insert Queries: -

```sql
-- Inserting into Branch Table
INSERT INTO Branch (Branch_Code, Branch_Name, Branch_Loc) VALUES
(1001, 'Main Branch', 'Bangalore'),
(1002, 'Central Branch', 'Bangalore'),
(1003, 'North Branch', 'Bangalore'),
(1004, 'South Branch', 'Bangalore'),
(1005, 'East Branch', 'Bangalore'),
(1006, 'West Branch', 'Bangalore');
```

```sql
-- Inserting into Customer Tables
INSERT INTO Customer_Info (Cust_ID, Name, DOB, Gender, Branch_Code) VALUES
(100001, 'Ramesh Sharma', '1990-05-15', 'M', 1001),
(100002, 'Priya Patel', '1985-08-21', 'F', 1002),
(100003, 'Amit Singh', '1988-12-10', 'M', 1003),
(100004, 'Neha Gupta', '1992-02-28', 'F', 1004),
(100005, 'Rajesh Kumar', '1995-07-03', 'M', 1005),
(100006, 'Sunita Devi', '1983-09-17', 'F', 1006);
```

```sql
-- Inserting into Employee Tables
INSERT INTO Emp_Info (Emp_Ref, Name, Gender, Address, Branch_Code) VALUES
(20001, 'Vikas Mishra', 'M', '123, ABC Street, Mumbai', 1001),
(20002, 'Anita Verma', 'F', '456, XYZ Street, Delhi', 1002),
(20003, 'Sunil Patel', 'M', '789, PQR Street, Jaipur', 1003),
(20004, 'Manisha Singh', 'F', '101, LMN Street, Chennai', 1004),
(20005, 'Alok Kumar', 'M', '111, DEF Street, Kolkata', 1005),
(20006, 'Meena Devi', 'F', '222, GHI Street, Ahmedabad', 1006);

INSERT INTO Emp_Contact (Emp_Ref, Contact) VALUES
(20001, '8765432101'),
(20002, '8765432102'),
(20003, '8765432103'),
(20004, '8765432104'),
(20005, '8765432105'),
(20006, '8765432106');
```

```sql
INSERT INTO Address_Info (Cust_ID, Name, Address) VALUES
(100001, 'Ramesh Sharma', '123, Devanhalli, Bangalore'),
(100002, 'Priya Patel', '456, Marathanhalli, Bangalore'),
(100003, 'Amit Singh', '789, Majestic, Bangalore'),
(100004, 'Neha Gupta', '101, Perambur, Chennai'),
(100005, 'Rajesh Kumar', '111,Ayoydhya'),
(100006, 'Sunita Devi', '222,New Chowk, Lucknow ');
```

```sql
INSERT INTO Customer_Contact (Cust_ID, Contact) VALUES
(100001, '9876543210'),
(100002, '9876543211'),
(100003, '9876543212'),
(100004, '9876543213'),
(100005, '9876543214'),
(100006, '9876543215');
```

```sql
-- Inserting into Loan Tables
INSERT INTO Loan_Type (Loan_Type, Interest) VALUES
('Home Loan', 8.5),
('Personal Loan', 12.0),
('Vehicle Loan', 9.0);

INSERT INTO Loan_Details (Acc_No, Loan_Type, Duration, Amount, Status) VALUES
(1000003, 'Home Loan', 15, 5000000.00, 'Active'),
(1000006, 'Vehicle Loan', 5, 300000.00, 'Active');
```

```sql
-- Inserting into Account Tables
INSERT INTO Cust_Accounts (Cust_ID, Acc_No, Account_Type) VALUES
(100001, 1000001, 'Savings'),
(100002, 1000002, 'Current'),
(100003, 1000003, 'Fixed Deposit'),
(100004, 1000004, 'Savings'),
(100005, 1000005, 'Current'),
(100006, 1000006, 'Fixed Deposit');

INSERT INTO Account_Details (Account_Type, Status, Interest_Rate, Balance) VALUES
('Savings', 'Active', 4.5, 50000.00),
('Current', 'Active', 0.0, 10000.00),
('Fixed Deposit', 'Active', 7.0, 100000.00);
```

```sql
-- Inserting into Transaction Table
INSERT INTO Transaction (Transaction_Ref, Tr_Date, Tr_Amount, Tr_Type, Tr_Desc, Cust_ID, Acc_No) VALUES
(500001, '2024-04-01', 5000.00, 'Deposit', 'Deposit for savings account', 100001, 1000001),
(500002, '2024-04-02', 2500.00, 'Withdrawal', 'Withdrawal for expenses', 100002, 1000002),
(500003, '2024-04-03', 10000.00, 'Deposit', 'Deposit for fixed deposit account', 100003, 1000003),
(500004, '2024-04-04', 2000.00, 'Withdrawal', 'Withdrawal for emergency', 100004, 1000004);

INSERT INTO Transaction (Transaction_Ref, Tr_Date, Tr_Amount, Tr_Type, Tr_Desc, Cust_ID, Acc_No) VALUES
(500005, '2024-04-05', 7500.00, 'Deposit', 'Deposit for savings account', 100005, 1000005),
(500006, '2024-04-06', 3000.00, 'Withdrawal', 'Withdrawal for expenses', 100006, 1000006),
(500007, '2024-04-07', 15000.00, 'Deposit', 'Deposit for fixed deposit account', 100001, 1000001),
(500008, '2024-04-08', 5000.00, 'Withdrawal', 'Withdrawal for emergency', 100002, 1000002),
(500009, '2024-04-09', 10000.00, 'Deposit', 'Deposit for savings account', 100003, 1000003),
(500010, '2024-04-10', 2000.00, 'Withdrawal', 'Withdrawal for expenses', 100004, 1000004);
```

```sql
INSERT INTO Repayment (Payment_Seqno, Payment_Amt, Payment_Date, Remaining_Amt, Acc_No) VALUES
(600001, 2000.00, '2024-04-01', 3000.00, 1000003),
(600002, 1500.00, '2024-04-02', 1500.00, 1000006),
(600003, 5000.00, '2024-04-03', 8000.00, 1000003),
(600004, 1000.00, '2024-04-04', 4000.00, 1000006);
```

Table Values: -

Branch:

Branch_Code	Branch_Name	Branch_Loc
1001	Main Branch	Bangalore
1002	Central Branch	Bangalore
1003	North Branch	Bangalore
1004	South Branch	Bangalore
1005	East Branch	Bangalore
1006	West Branch	Bangalore
NULL	NULL	NULL

Customer Tables' Values:

Cust_ID	Name	Address
100003	Amit Singh	789, Majestic, Bangalore
100007	Amita Sharma	789, ABC Nagar, Bangalore
100009	Anjali Singh	101, PQR Road, Jaipur
100010	Karan Gupta	222, MNO Street, Chennai
100004	Neha Gupta	101, Perambur, Chennai
100002	Priya Patel	456, Marathanhalli, Bangalore
100005	Rajesh Kumar	111,Ayoydhya
100001	Ramesh Sharma	123, Devanhalli, Bangalore
100008	Sanjay Verma	456, XYZ Colony, Delhi
100011	Shreya Kumar	333, GHI Lane, Kolkata
100006	Sunita Devi	222,New Chowk, Lucknow
NULL	NULL	NULL

Cust_ID	Contact
100001	9876543210
100002	9876543211
100003	9876543212
100004	9876543213
100005	9876543214
100006	9876543215
100007	9876543216
100008	9876543217
100009	9876543218
100010	9876543219
100011	9876543220
NULL	NULL

Cust_ID	Name	DOB	Gender	Branch_Code
100001	Ramesh Sharma	1990-05-15	M	1001
100002	Priya Patel	1985-08-21	F	1002
100003	Amit Singh	1988-12-10	M	1003
100004	Neha Gupta	1992-02-28	F	1004
100005	Rajesh Kumar	1995-07-03	M	1005
100006	Sunita Devi	1983-09-17	F	1006
100007	Amita Sharma	1993-11-12	F	1001
100008	Sanjay Verma	1980-06-25	M	1002
100009	Anjali Singh	1991-03-18	F	1003
100010	Karan Gupta	1987-09-30	M	1004
100011	Shreya Kumar	1996-04-07	F	1005
NULL	NULL	NULL	NULL	NULL

Employee Tables' Values:

Emp_Ref	Name	Gender	Address	Branch_Code
20001	Vikas Mishra	M	123, ABC Street, Mumbai	1001
20002	Anita Verma	F	456, XYZ Street, Delhi	1002
20003	Sunil Patel	M	789, PQR Street, Jaipur	1003
20004	Manisha Singh	F	101, LMN Street, Chennai	1004
20005	Alok Kumar	M	111, DEF Street, Kolkata	1005
20006	Meena Devi	F	222, GHI Street, Ahmedabad	1006
NULL	NULL	NULL	NULL	NULL

Emp_ref	DOB	Age
20001	1988-03-25	36
20002	1985-09-15	39
20003	1990-12-03	31
20004	1983-05-21	38
20005	1987-07-12	34
20006	1992-11-30	29
NULL	NULL	NULL

Emp_Ref	Contact
20001	8765432101
20002	8765432102
20003	8765432103
20004	8765432104
20005	8765432105
20006	8765432106
NULL	NULL

Emp_Ref	Position	Salary
20001	Manager	75000.00
20002	Clerk	35000.00
20003	Assistant Manager	60000.00
20004	Cashier	40000.00
20005	Accountant	50000.00
20006	Assistant	30000.00
NULL	NULL	NULL

Account & Transaction Tables' Values:

Account_Type	Status	Interest_Rate	Balance
Current	Active	0.00	10000.00
Fixed Deposit	Active	7.00	100000.00
Savings	Active	4.50	50000.00
NULL	NULL	NULL	NULL

Transaction_Ref	Tr_Date	Tr_Amount	Tr_Type	Tr_Desc	Cust_ID	Acc_No
500001	2024-04-01	5000.00	Deposit	Deposit for savings account	100001	1000001
500002	2024-04-02	2500.00	Withdrawal	Withdrawal for expenses	100002	1000002
500003	2024-04-03	10000.00	Deposit	Deposit for fixed deposit account	100003	1000003
500004	2024-04-04	2000.00	Withdrawal	Withdrawal for emergency	100004	1000004
500005	2024-04-05	7500.00	Deposit	Deposit for savings account	100005	1000005
500006	2024-04-06	3000.00	Withdrawal	Withdrawal for expenses	100006	1000006
500007	2024-04-07	15000.00	Deposit	Deposit for fixed deposit account	100001	1000001
500008	2024-04-08	5000.00	Withdrawal	Withdrawal for emergency	100002	1000002
500009	2024-04-09	10000.00	Deposit	Deposit for savings account	100003	1000003
500010	2024-04-10	2000.00	Withdrawal	Withdrawal for expenses	100004	1000004
500011	2024-04-11	7000.00	Deposit	Deposit for savings account	100007	1000007
500012	2024-04-12	3500.00	Withdrawal	Withdrawal for expenses	100008	1000008
500013	2024-04-13	12000.00	Deposit	Deposit for fixed deposit account	100009	1000009
500014	2024-04-14	3000.00	Withdrawal	Withdrawal for emergency	100010	1000010
500015	2024-04-15	8000.00	Deposit	Deposit for savings account	100011	1000011
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Loan & Repayment Table Values:

Loan_Type	Interest
Home Loan	8.50
Personal Loan	12.00
Vehicle Loan	9.00
NULL	NULL

Acc_No	Loan_Type	Duration	Amount	Status
1000003	Home Loan	15	5000000.00	Active
1000006	Vehicle Loan	5	300000.00	Active
1000007	Home Loan	20	8000000.00	Active
1000010	Personal Loan	5	200000.00	Active
NULL	NULL	NULL	NULL	NULL

Payment_Seqno	Payment_Amt	Payment_Date	Remaining_Amt	Acc_No
600001	2000.00	2024-04-01	3000.00	1000003
600002	1500.00	2024-04-02	1500.00	1000006
600003	5000.00	2024-04-03	8000.00	1000003
600004	1000.00	2024-04-04	4000.00	1000006
NULL	NULL	NULL	NULL	NULL

Procedures: -

Sets of SQL statements for specific tasks, stored in the database, callable by name, and often used for data manipulation and business logic.

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetCustomerLoanRepayments`(IN customer_id INT)
BEGIN
    SELECT r.Payment_Seqno, r.Payment_Amt, r.Payment_Date, r.Remaining_Amt, ld.Loan_Type
    FROM Repayment r
    JOIN Loan_Details ld ON r.Acc_No = ld.Acc_No
    JOIN Cust_Accounts ca ON ld.Acc_No = ca.Acc_No
    WHERE ca.Cust_ID = customer_id
    ORDER BY r.Payment_Date;
END
```

```sql
CALL GetCustomerLoanRepayments(100003);
```

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

Payment_Seqno	Payment_Amt	Payment_Date	Remaining_Amt	Loan_Type
600001	2000.00	2024-04-01	3000.00	Home Loan
600003	5000.00	2024-04-03	8000.00	Home Loan

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetCustomerBalance`(IN custID INT)
    DETERMINISTIC
BEGIN
    DECLARE totalBalance DECIMAL(12, 2);

    SELECT SUM(ad.Balance)
    INTO totalBalance
    FROM Cust_Accounts ca
    JOIN Account_Details ad ON ca.Account_Type = ad.Account_Type
    WHERE ca.Cust_ID = custID;

    SELECT totalBalance;
END
```

```sql
call GetCustomerBalance(100002);
```

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

totalBalance
10000.00

Functions: -

Sets of SQL statements returning a value, commonly used for calculations and data transformations, used when code reusability is preferred.

```sql
CREATE DEFINER=`root`@`localhost` FUNCTION `Emp_Ref_Length`(Emp_Ref INT) RETURNS int
    DETERMINISTIC
BEGIN
    DECLARE len INT;
    SET len = CHAR_LENGTH(CAST(Emp_Ref AS CHAR));
    RETURN CASE WHEN len = 5 THEN 1 ELSE 0 END;
END
```

```sql
-- Call the function to check employee reference length
SELECT Emp_Ref_Length(12345) AS Is_Length_5;
```

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

Is_Length_5
1

```sql
CREATE DEFINER=`root`@`localhost` FUNCTION `CalculateInterest`(amount DECIMAL(12, 2), interest_rate DECIMAL(5, 2)) RETURNS decimal(12,2)
    DETERMINISTIC
BEGIN
    DECLARE interest DECIMAL(12, 2);
    SET interest = (amount * interest_rate) / 100;
    RETURN interest;
END
```

```sql
-- Call the function to calculate interest
SELECT CalculateInterest(100000.00, 6.5) AS Total_Interest;
```

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

Total_Interest
6500.00

```sql
CREATE DEFINER=`root`@`localhost` FUNCTION `CalculateAge`(DOB DATE) RETURNS int
    DETERMINISTIC
BEGIN
    RETURN YEAR(CURDATE()) - YEAR(DOB) - (DATE_FORMAT(CURDATE(), '%m%d') < DATE_FORMAT(DOB, '%m%d'));
END
```

```sql
-- Call the function to calculate age
SELECT CalculateAge('1990-05-15') AS Age;
```

Result Grid | Filter Rows: | Export: | Wrap

Age
33

Triggers: -

Special stored procedures executing automatically in response to specific database events, like INSERT, UPDATE, DELETE operations, commonly used for enforcing data integrity and auditing changes.

```
DELIMITER $$
CREATE TRIGGER Emp_Ref BEFORE INSERT ON Emp_Info
FOR EACH ROW
BEGIN
    DECLARE len INT;
    SET len = CHAR_LENGTH(CAST(NEW.Emp_Ref AS CHAR));
    IF len != 5 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Emp_Ref length must be 5 digits';
    END IF;
END$$
DELIMITER ;
```

```
-- Trigger for audit logging
-- for insert
DELIMITER $$
CREATE TRIGGER Emp_insert_Audit
AFTER INSERT ON Emp_Info
FOR EACH ROW
BEGIN
    INSERT INTO Emp_Info_Audit_Log (Emp_Ref, New_Name, Change_Date)
    VALUES (NEW.Emp_Ref, NEW.Name, NOW());
END$$
DELIMITER ;
```

```
-- for update
DELIMITER $$
CREATE TRIGGER Emp_Update_Audit
AFTER UPDATE ON Emp_Info
FOR EACH ROW
BEGIN
    INSERT INTO Emp_Info_Audit_Log (Emp_Ref, Old_Name, New_Name, Change_Date)
    VALUES (OLD.Emp_Ref, OLD.Name, NEW.Name, NOW());
END$$
DELIMITER ;
```

```sql
-- for delete
DELIMITER $$
CREATE TRIGGER Emp_Info_Audit_Delete
AFTER DELETE ON Emp_Info
FOR EACH ROW
BEGIN
    INSERT INTO Emp_Info_Audit_Log (Emp_Ref, Old_Name, Change_Date)
    VALUES (OLD.Emp_Ref, OLD.Name, NOW());
END$$
DELIMITER ;
```

```sql
-- Trigger for address updation
DELIMITER $$
CREATE TRIGGER Customer_Info_Up_Add
AFTER INSERT ON Customer_Info
FOR EACH ROW
BEGIN
    UPDATE Address_Info
    SET Address = NEW.Address
    WHERE Cust_ID = NEW.Cust_ID;
END$$
DELIMITER ;
```

```sql
-- Implement a trigger to automatically update the age of customers
-- whenever a new customer is added or the date of birth is updated.

DELIMITER $$
CREATE TRIGGER Update_Customer_Age
AFTER INSERT ON Customer_Info
FOR EACH ROW
BEGIN
    INSERT INTO Age_Info (Cust_ID, DOB, Age)
    VALUES (NEW.Cust_ID, NEW.DOB, TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE()));
END$$
DELIMITER ;
```

Views: -

Virtual tables generated by SELECT queries, simplifying complex queries and presenting data in a structured format.

```
-- Employee details view
CREATE VIEW Employee_Details AS
SELECT e.*, b.Branch_Name, b.Branch_Loc
FROM Emp_Info e
JOIN Branch b ON e.Branch_Code = b.Branch_Code;
```

Emp_Ref	Name	Gender	Address	Branch_Code	Branch_Name	Branch_Loc
20001	Vikas Mishra	M	123, ABC Street, Mumbai	1001	Main Branch	Bangalore
20002	Anita Verma	F	456, XYZ Street, Delhi	1002	Central Branch	Bangalore
20003	Sunil Patel	M	789, PQR Street, Jaipur	1003	North Branch	Bangalore
20004	Manisha Singh	F	101, LMN Street, Chennai	1004	South Branch	Bangalore
20005	Alok Kumar	M	111, DEF Street, Kolkata	1005	East Branch	Bangalore
20006	Meena Devi	F	222, GHI Street, Ahmedabad	1006	West Branch	Bangalore

```
-- Customer summary view
CREATE VIEW Customer_Summary AS
SELECT ci.Cust_ID, ci.Name, ci.DOB, cc.Contact, ai.Address, b.Branch_Name, b.Branch_Loc
FROM Customer_Info ci
JOIN Customer_Contact cc ON ci.Cust_ID = cc.Cust_ID
JOIN Address_Info ai ON ci.Cust_ID = ai.Cust_ID
JOIN Branch b ON ci.Branch_Code = b.Branch_Code;
```

Cust_ID	Name	DOB	Contact	Address	Branch_Name	Branch_Loc
100003	Amit Singh	1988-12-10	9876543212	789, Majestic, Bangalore	North Branch	Bangalore
100007	Amita Sharma	1993-11-12	9876543216	789, ABC Nagar, Bangalore	Main Branch	Bangalore
100009	Anjali Singh	1991-03-18	9876543218	101, PQR Road, Jaipur	North Branch	Bangalore
100010	Karan Gupta	1987-09-30	9876543219	222, MNO Street, Chennai	South Branch	Bangalore
100004	Neha Gupta	1992-02-28	9876543213	101, Perambur, Chennai	South Branch	Bangalore
100002	Priya Patel	1985-08-21	9876543211	456, Marathanhalli, Bangalore	Central Branch	Bangalore
100005	Rajesh Kumar	1995-07-03	9876543214	111,Ayoydhya	East Branch	Bangalore
100001	Ramesh Sharma	1990-05-15	9876543210	123, Devanhalli, Bangalore	Main Branch	Bangalore
100008	Sanjay Verma	1980-06-25	9876543217	456, XYZ Colony, Delhi	Central Branch	Bangalore
100011	Shreya Kumar	1996-04-07	9876543220	333, GHI Lane, Kolkata	East Branch	Bangalore
100006	Sunita Devi	1983-09-17	9876543215	222,New Chowk, Lucknow	West Branch	Bangalore

```sql
-- Loan Summary View
CREATE VIEW Loan_Summary AS
SELECT ld.Acc_No, ld.Loan_Type, lt.Interest, ld.Amount, ld.Duration, ld.Status
FROM Loan_Details ld
JOIN Loan_Type lt ON ld.Loan_Type = lt.Loan_Type;
select * from Loan_Summary;
```

Acc_No	Loan_Type	Interest	Amount	Duration	Status
1000003	Home Loan	8.50	5000000.00	15	Active
1000007	Home Loan	8.50	8000000.00	20	Active
1000010	Personal Loan	12.00	200000.00	5	Active
1000006	Vehicle Loan	9.00	300000.00	5	Active

Performing Queries: -

1.Details of customers along with their contact information and branch location:

```
SELECT ci.Name AS Customer_Name,
ci.DOB AS Date_of_Birth, cc.Contact AS Contact_Number,
b.Branch_Loc AS Branch_Location
FROM Customer_Info ci
JOIN Customer_Contact cc ON ci.Cust_ID = cc.Cust_ID
JOIN Branch b ON ci.Branch_Code = b.Branch_Code;
```

Customer_Name	Date_of_Birth	Contact_Number	Branch_Location
Ramesh Sharma	1990-05-15	9876543210	Bangalore
Amita Sharma	1993-11-12	9876543216	Bangalore
Priya Patel	1985-08-21	9876543211	Bangalore
Sanjay Verma	1980-06-25	9876543217	Bangalore
Amit Singh	1988-12-10	9876543212	Bangalore
Anjali Singh	1991-03-18	9876543218	Bangalore
Neha Gupta	1992-02-28	9876543213	Bangalore
Karan Gupta	1987-09-30	9876543219	Bangalore
Rajesh Kumar	1995-07-03	9876543214	Bangalore
Shreya Kumar	1996-04-07	9876543220	Bangalore
Sunita Devi	1983-09-17	9876543215	Bangalore

2.List of employees along with their position and salary, sorted by salary in descending order:

```
SELECT ei.Name AS Employee_Name, es.Position AS Position, es.Salary AS Salary
FROM Emp_Info ei
JOIN Emp_Status es ON ei.Emp_Ref = es.Emp_Ref
ORDER BY es.Salary DESC;
```

Employee_Name	Position	Salary
Vikas Mishra	Manager	75000.00
Sunil Patel	Assistant Manager	60000.00
Alok Kumar	Accountant	50000.00
Manisha Singh	Cashier	40000.00
Anita Verma	Clerk	35000.00
Meena Devi	Assistant	30000.00

3. Details of loans taken by customers, including the loan type and amount:

```
SELECT ci.Name AS Customer_Name, lt.Loan_Type AS Loan_Type, ld.Amount AS Loan_Amount
FROM Customer_Info ci
JOIN Cust_Accounts ca ON ci.Cust_ID = ca.Cust_ID
JOIN Loan_Details ld ON ca.Acc_No = ld.Acc_No
JOIN Loan_Type lt ON ld.Loan_Type = lt.Loan_Type;
```

Customer_Name	Loan_Type	Loan_Amount
Amit Singh	Home Loan	5000000.00
Amita Sharma	Home Loan	8000000.00
Karan Gupta	Personal Loan	200000.00
Sunita Devi	Vehicle Loan	300000.00

4. Calculating age of customers

```
-- calculating age of customers
SELECT ci.Name AS Customer_Name, ci.DOB AS Date_of_Birth,
DATEDIFF(CURRENT_DATE(), ci.DOB) / 365 AS Age
FROM Customer_Info ci;
```

Customer_Name	Date_of_Birth	Age
Ramesh Sharma	1990-05-15	33.9315
Priya Patel	1985-08-21	38.6658
Amit Singh	1988-12-10	35.3589
Neha Gupta	1992-02-28	32.1397
Rajesh Kumar	1995-07-03	28.7945
Sunita Devi	1983-09-17	40.5945
Amita Sharma	1993-11-12	30.4329
Sanjay Verma	1980-06-25	43.8247
Anjali Singh	1991-03-18	33.0904
Karan Gupta	1987-09-30	36.5562
Shreya Kumar	1996-04-07	28.0301

5. Identify the average age of customers in each branch and gender distribution within each age group to understand the demographic profile of customers

```
SELECT Branch.Branch_Name,
    AVG(Age_Info.Age) AS Avg_Customer_Age,
    SUM(CASE WHEN Customer_Info.Gender = 'M' THEN 1 ELSE 0 END) AS Male_Customers,
    SUM(CASE WHEN Customer_Info.Gender = 'F' THEN 1 ELSE 0 END) AS Female_Customers
FROM Branch JOIN
    Customer_Info ON Branch.Branch_Code = Customer_Info.Branch_Code
JOIN
    Age_Info ON Customer_Info.Cust_ID = Age_Info.Cust_ID
GROUP BY
    Branch.Branch_Name;
```

Branch_Name	Avg_Customer_Age	Male_Customers	Female_Customers
Main Branch	32.0000	1	1
Central Branch	41.0000	1	1
North Branch	34.5000	1	1
South Branch	34.0000	1	1
East Branch	28.5000	1	1
West Branch	41.0000	0	1

6. Identify the total outstanding loan amount for each loan type to assess the bank's exposure

```
SELECT Loan_Details.Loan_Type,
    SUM(Loan_Details.Amount) AS Total_Outstanding_Amount
FROM Loan_Details
GROUP BY Loan_Details.Loan_Type;
```

Loan_Type	Total_Outstanding_Amount
Home Loan	13000000.00
Personal Loan	200000.00
Vehicle Loan	300000.00

7. Identify customers with the highest transaction volumes and their transaction patterns to offer personalized banking services and incentives.

```
SELECT ci.Name,
    COUNT(t.Transaction_Ref) AS Total_Transactions,
    SUM(t.Tr_Amount) AS Total_Amount_Transacted
FROM Customer_Info ci
JOIN Transaction t ON ci.Cust_ID = t.Cust_ID
GROUP BY ci.Name
ORDER BY Total_Transactions DESC
LIMIT 10;
```

Name	Total_Transactions	Total_Amount_Transacted
Ramesh Sharma	2	20000.00
Priya Patel	2	7500.00
Amit Singh	2	20000.00
Neha Gupta	2	4000.00
Rajesh Kumar	1	7500.00
Sunita Devi	1	3000.00
Amita Sharma	1	7000.00
Sanjay Verma	1	3500.00
Anjali Singh	1	12000.00
Karan Gupta	1	3000.00

8. Retrieve the loan types with the highest interest rates.

```
SELECT lt.Loan_Type, lt.Interest
FROM Loan_Type lt
WHERE lt.Interest = (SELECT MAX(Interest) FROM Loan_Type);
```

Loan_Type	Interest
Personal Loan	12.00
NULL	NULL

9. Generate a report showing the top 5 customers with the highest account balances.

```
SELECT c.Cust_ID, c.Name, ca.Acc_No, ad.Balance
FROM Customer_Info c
JOIN Cust_Accounts ca ON c.Cust_ID = ca.Cust_ID
JOIN Account_Details ad ON ca.Account_Type = ad.Account_Type
ORDER BY ad.Balance DESC
LIMIT 5;
```

Cust_ID	Name	Acc_No	Balance
100003	Amit Singh	1000003	100000.00
100006	Sunita Devi	1000006	100000.00
100009	Anjali Singh	1000009	100000.00
100001	Ramesh Sharma	1000001	50000.00
100004	Neha Gupta	1000004	50000.00

10. Calculate the total deposit amount for each branch.

```
SELECT b.Branch_Name,
    SUM(CASE WHEN t.Tr_Type = 'Deposit' THEN t.Tr_Amount ELSE 0 END) AS Total_Deposits
FROM Branch b
LEFT JOIN Customer_Info ci ON b.Branch_Code = ci.Branch_Code
LEFT JOIN Transaction t ON ci.Cust_ID = t.Cust_ID
GROUP BY b.Branch_Name;
```

Branch_Name	Total_Deposits
Main Branch	27000.00
Central Branch	0.00
North Branch	32000.00
South Branch	0.00
East Branch	15500.00
West Branch	0.00

Summary: -

In conclusion, the development of the bank management system project has provided a comprehensive understanding of database management concepts and their practical application. Through the design and implementation of database structures such as tables, views, procedures, functions, and triggers, the project has demonstrated proficiency in data modeling, manipulation, and retrieval. The utilization of SQL queries, functions, and stored procedures has facilitated efficient data management, ensuring integrity, security, and scalability of the system. Furthermore, the project has enhanced problem-solving skills by addressing real-world scenarios such as customer demographics analysis, employee performance evaluation, and transaction trend analysis. Overall, the project has been instrumental in honing database development skills and preparing for future endeavors in database management and application development.