BANKING MANAGEMENT SYSTEM

Team members:

1. Bharadwaj
2. Deepali
3. Umang
4. Murari
5. Biswajit
6. Shankar

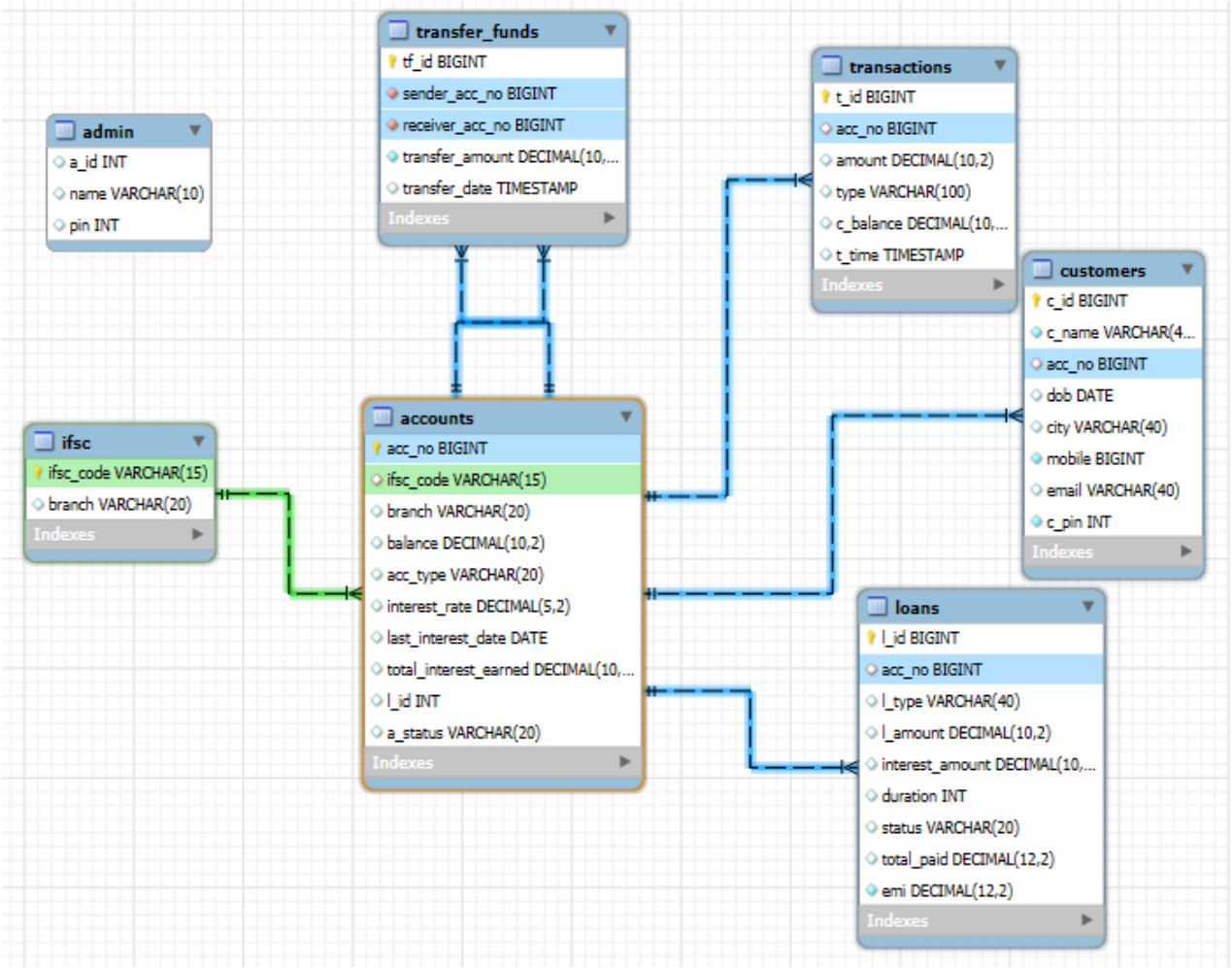
PROJECT DESCRIPTION:

The Banking Management System is a Python and MySQL-based application designed to automate and manage various banking operations efficiently. It provides two main interfaces — Admin and Customer — each with dedicated functionalities. The system allows customers to create new accounts, deposit and withdraw funds, transfer money, apply for loans, repay EMIs, and view transaction history securely. It also includes a “forgot PIN” feature for user convenience.

On the administrative side, the system enables bank staff to manage customer accounts, approve or reject loan requests, modify interest rates, generate reports, and perform financial analysis such as identifying top customers and monitoring branch-wise deposits. The database integrates multiple tables like accounts, customers, transactions, loans, and transfer_funds, ensuring data consistency and relational integrity through the use of primary and foreign keys.

Overall, this project provides a robust and interactive platform for handling core banking functions digitally, minimizing manual work, improving accuracy, and ensuring secure financial management within the organization.

ER DIAGRAM:



The Entity-Relationship (ER) Diagram of the Banking Management System illustrates the logical structure and relationships among different entities involved in managing banking operations. The main entities include Admin, Customers, Accounts, Loans, Transactions, Transfer_Funds, and IFSC. The Accounts table serves as the central entity, connecting to the Customers table through a foreign key to store individual customer account details, to the Loans table for tracking loan information, and to the Transactions table for recording deposits, withdrawals, and other financial activities. The Transfer_Funds table maintains sender and receiver details for fund transfers, ensuring transparency and accuracy in inter-account transactions.

The Admin entity manages the overall system, allowing administrative control over accounts, loans, and reports, while the IFSC table defines bank branch details linked to specific accounts. Each relationship is established using primary and foreign keys. This well-structured design supports smooth data retrieval, secure financial processing, and scalable management of customer and banking data within the system.

SQL SCRIPTS:

CREATE TABLE queries:

1. CREATE TABLE accounts (
   acc_no BIGINT PRIMARY KEY AUTO_INCREMENT,
   ifsc_code VARCHAR(15),
   branch VARCHAR(20),
   balance DECIMAL(10,2),
   acc_type VARCHAR(20),
   interest_rate decimal(5,2) Default 0.00,
   last_interest_date DATE,
   total_interest_earned DECIMAL(10,2),
   l_id BIGINT,
   a_status VARCHAR(20) DEFAULT 'active',
   FOREIGN KEY (ifsc_code) REFERENCES ifsc(ifsc_code)
) AUTO_INCREMENT = 2501000001;

2. CREATE TABLE customers (
   c_id BIGINT PRIMARY KEY AUTO_INCREMENT,
   c_name VARCHAR(40) NOT NULL,
   acc_no BIGINT,
   dob DATE,
   city VARCHAR(40),
   mobile BIGINT NOT NULL,
   email VARCHAR(40),
   c_pin INT NOT NULL,
   FOREIGN KEY (acc_no) REFERENCES accounts(acc_no)
) AUTO_INCREMENT = 2502000001;

3. CREATE TABLE transactions (
   t_id BIGINT AUTO_INCREMENT PRIMARY KEY,
   acc_no BIGINT,
   amount DECIMAL(10,2),

type VARCHAR(100),
c_balance DECIMAL(10,2),
t_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (acc_no) REFERENCES accounts(acc_no)
) AUTO_INCREMENT = 2503000001;

4. CREATE TABLE transfer_funds (
tf_id BIGINT AUTO_INCREMENT PRIMARY KEY,
sender_acc_no BIGINT NOT NULL,
receiver_acc_no BIGINT NOT NULL,
transfer_amount DECIMAL(10,2) NOT NULL,
transfer_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY(sender_acc_no) REFERENCES accounts(acc_no),
FOREIGN KEY(receiver_acc_no) REFERENCES accounts(acc_no)
) AUTO_INCREMENT = 2504000001;

5. CREATE TABLE loans (
l_id BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
acc_no BIGINT NULL,
l_type VARCHAR(40) NULL,
l_amount DECIMAL(10,2) NULL,
interest_amount DECIMAL(10,2) NULL,
duration INT NULL,
status VARCHAR(20) NULL,
total_paid DECIMAL(12,2) DEFAULT 0.00,
emi DECIMAL(12,2) NOT NULL DEFAULT 0.00,
FOREIGN KEY (acc_no) REFERENCES accounts(acc_no)
) AUTO_INCREMENT = 2505000001;

6. CREATE TABLE admin(
a_id INT,
name VARCHAR(10),

pin int
);

7. CREATE TABLE ifsc (
   ifsc_code VARCHAR(15) PRIMARY KEY,
   branch VARCHAR(20)
   );

INSERT queries:

1. INSERT INTO ifsc (ifsc_code, branch) VALUES
   ('BMSN0000847','HyderabadMain'),
   ('BMSN0001879','Charminar'),
   ('BMSN0011744','Mehdipatnam'),
   ('BMSN0032241','Kukatpally'),
   ('BMSN0040479','Banjara Hills');

2. INSERT INTO admin (a_id, name, pin) VALUES
   (101,'Deepali',1006),
   (102,'Bharadwaj',2580);

3. INSERT INTO accounts (ifsc_code, branch, balance, acc_type)
   VALUES ('BMSN0040479', 'Banjara Hills', 60599.66, 'Savings');

4. INSERT INTO customers (c_name, mobile, email, city, dob, acc_no, c_pin)
   VALUES ('Onveer Palla', 9587511295, 'onveerpalla79@example.com', 'Unnao',
   '1993-12-20', 2501000001, 5033);

5. INSERT INTO transactions (acc_no, amount, type, c_balance)
   VALUES (2501000003, 13283.53, 'deposit', 106347.78);

6. INSERT INTO transfer_funds (sender_acc_no, receiver_acc_no, transfer_amount)
   VALUES (2501000002, 2501000007, 5465.83);

7. INSERT INTO loans (acc_no, l_type, l_amount, interest_amount, duration, emi,
   total_paid, status)
   VALUES (2501000001, 'home', 800000.00, 347258.40, 10, 22291.67, 0.00,
   'pending');

SELECT queries:

1. SELECT a_status FROM accounts WHERE acc_no = 2501000001;
2. SELECT balance FROM accounts WHERE acc_no = 2501000001;
3. SELECT * FROM transactions where acc_no = 2501000001 limit 5;
4. SELECT l_id, l_amount, total_paid, emi, status FROM loans WHERE acc_no= 2501000001 AND status='ongoing';
5. SELECT c_name, mobile, c_pin, acc_no, dob FROM customers WHERE mobile = 9587511295;
6. SELECT name, pin FROM admin WHERE a_id = 101;

UPDATE queries:

1. UPDATE accounts SET l_id = 2505000001WHERE acc_no = 2501000001;
2. UPDATE accounts SET balance = 43873.03 WHERE acc_no = 2501000001;
3. UPDATE accounts SET interest_rate= 3.50 WHERE acc_type='Savings';
4. UPDATE loans SET status = 'ongoing' where l_id = 2505000001;

QUERIES USING JOIN, GROUP BY, HAVING:

Branch-wise deposit

```
SELECT
  i.branch AS Branch,
  SUM(a.balance) AS Total_Deposit
  FROM accounts a
  JOIN ifsc i ON a.ifsc_code = i.ifsc_code
  GROUP BY i.branch
  ORDER BY Total_Deposit DESC;
```

Branch	Total_Deposit
Banjara Hills	2636174.86
Kukatpally	2129643.56
Charminar	393475.06
Hyderabad Main	338616.93

Average balance by account type

```
SELECT
  acc_type AS Account_Type,
  ROUND(AVG(balance), 2) AS Average_Balance
  FROM accounts
  GROUP BY acc_type
  ORDER BY Average_Balance DESC;
```

Account_Type	Average_Balance
Savings	647473.60
Current	282567.80

Total deposits and withdrawals per month

```
SELECT
  acc_no,
  YEAR(t_time) as year,
  MONTH(t_time) as month,
  SUM(CASE WHEN type='deposit' THEN amount ELSE 0 END) as total_deposit,
  SUM(CASE WHEN type='withdraw' THEN amount ELSE 0 END) as total_withdraw
```

FROM transactions

GROUP BY acc_no, YEAR(t_time), MONTH(t_time)

ORDER BY acc_no, YEAR(t_time), MONTH(t_time);

acc_no	year	month	total_deposit	total_withdraw
2501000001	2025	10	54646.63	17309.93
2501000002	2025	10	26696.47	21280.74
2501000003	2025	10	34872.52	23543.93
2501000004	2025	10	32259.95	29969.34
2501000005	2025	10	37193.64	32504.93
2501000006	2025	10	33621.84	0.00
2501000007	2025	10	55523.59	24565.83
2501000008	2025	10	34275.98	7650.35
2501000009	2025	10	4521.56	28870.02
2501000010	2025	10	0.00	7718.41
2501000011	2025	10	5000.00	0.00

Most active customer by transaction count

SELECT

c.c_name, c.acc_no, COUNT(t.t_id) AS transaction_count

FROM customers c

JOIN transactions t ON c.acc_no = t.acc_no

GROUP BY c.acc_no, c.c_name

ORDER BY transaction_count DESC

LIMIT 1;

c_name	acc_no	transaction_count
Hiral Thaman	2501000003	17

Customer with highest balance

SELECT a.acc_no, c.c_name, a.balance, a.acc_type

FROM accounts a

JOIN customers c ON a.acc_no = c.acc_no

ORDER BY a.balance DESC

LIMIT 1;

acc_no	c_name	balance	acc_type
2501000003	Hiral Thaman	121624.79	Savings

Loan distribution by type

```
SELECT
  l_type, COUNT(*) AS total_loans, SUM(l_amount) AS total_amount
  FROM loans
  GROUP BY l_type
  ORDER BY total_amount DESC;
```

l_type	total_loans	total_amount
home	3	6600000.00
education	3	4500000.00
personal	1	300000.00

Interest collected per loan type

```
SELECT
  l_type, SUM(interest_amount) AS total_interest
  FROM loans
  GROUP BY l_type
  ORDER BY total_interest DESC;
```

l_type	total_interest
home	4208038.54
education	2845369.12
personal	16497.24