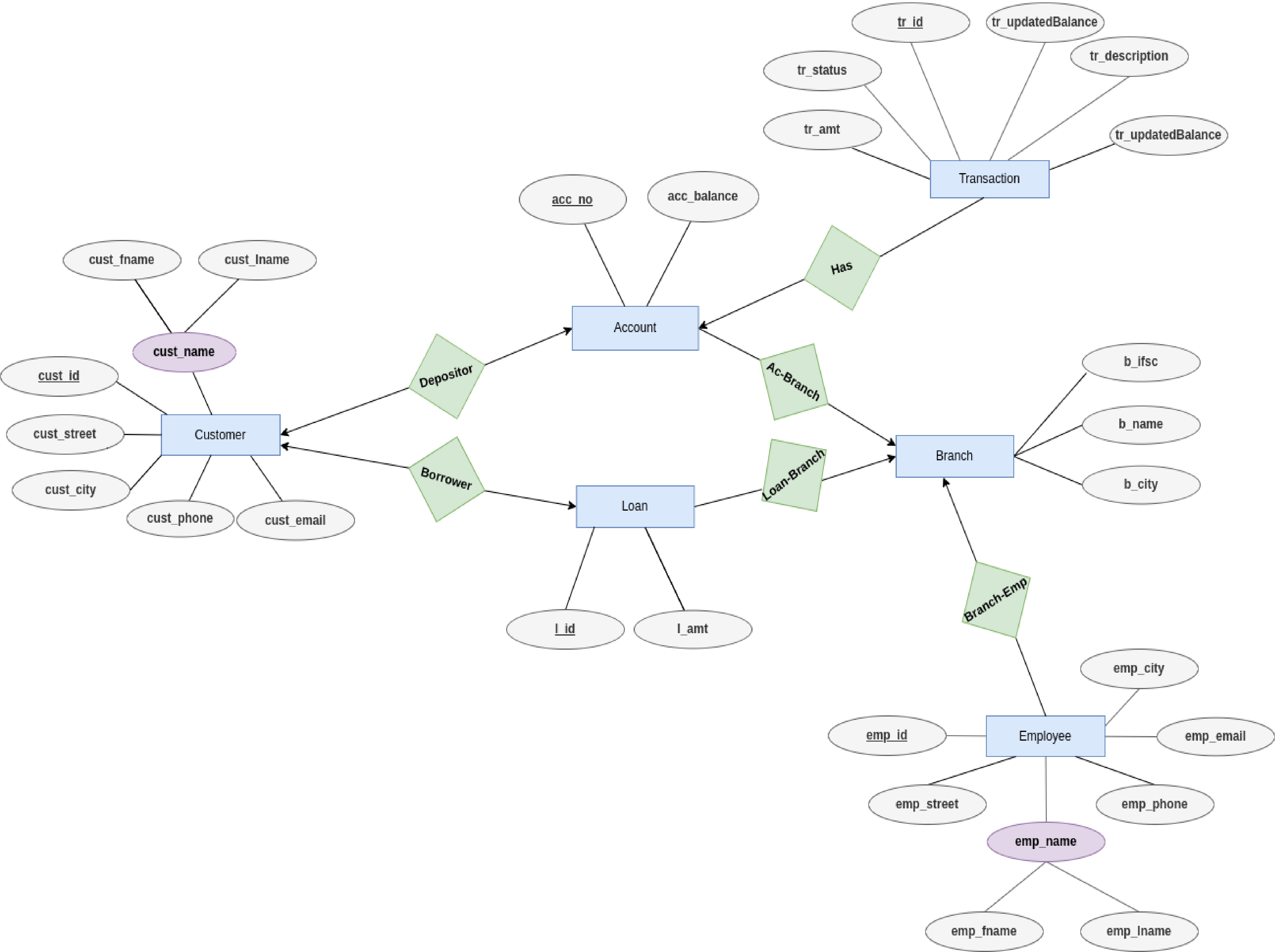
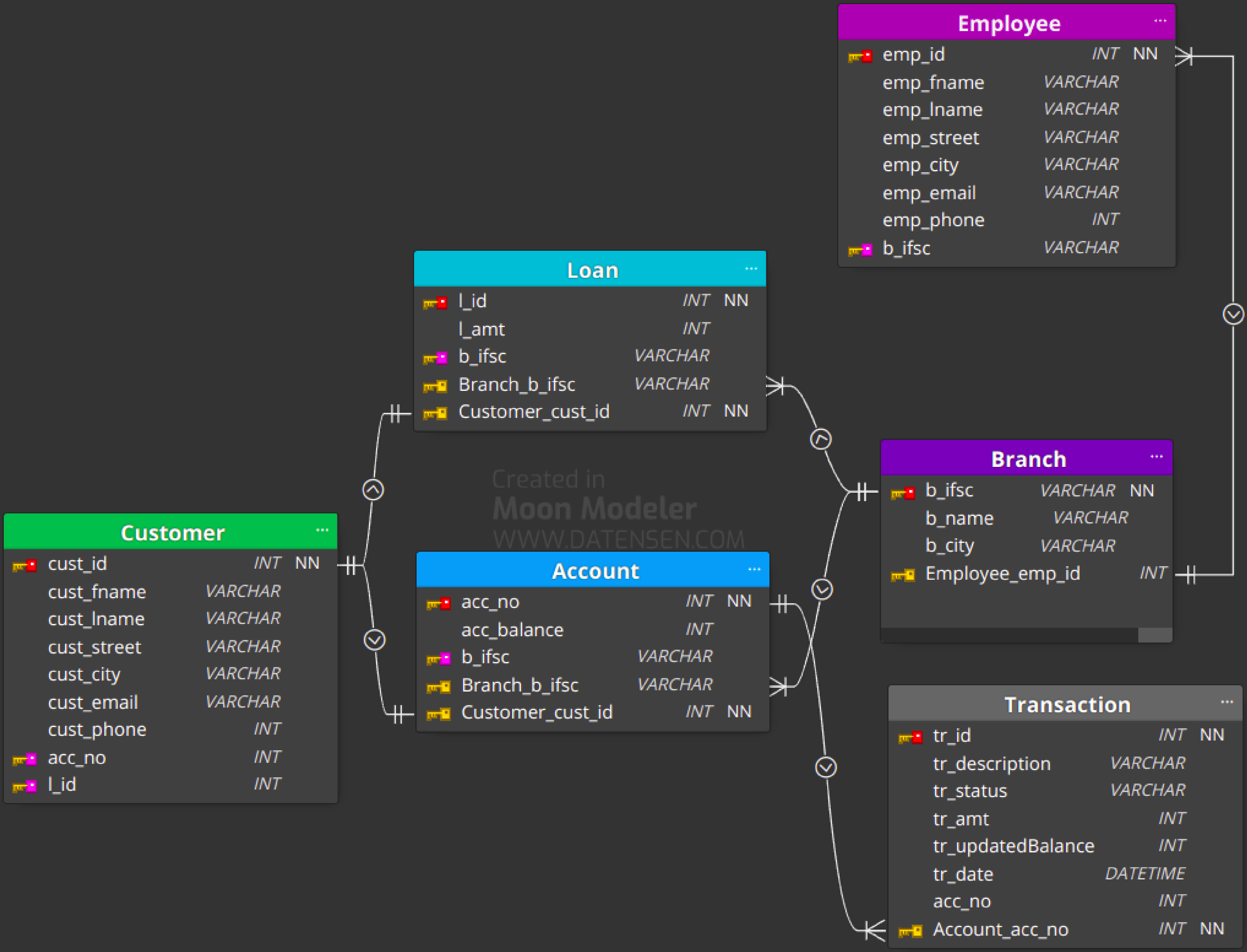
Banking Database MySQL


tr_id
tr_updatedBalance
tr_status
tr_description
tr_amt
tr_updatedBalance
Transaction
acc_no
acc_balance
Has
cust_fname
cust_lname
Account
cust_name
Depositor
Ac-Branch
cust_id
b_ifsc
cust_street
Customer
b_name
Branch
Borrower
Loan-Branch
cust_city
b_city
cust_phone
cust_email
Loan
Branch-Emp
l_id
l_amt
emp_city
emp_id
Employee
emp_email
emp_street
emp_phone
emp_name
emp_fname
emp_lname



Employee

emp_id	INT	NN
emp_fname	VARCHAR	
emp_lname	VARCHAR	
emp_street	VARCHAR	
emp_city	VARCHAR	
emp_email	VARCHAR	
emp_phone	INT	
b_ifsc	VARCHAR	

Loan

l_id	INT	NN
l_amt	INT	
b_ifsc	VARCHAR	
Branch_b_ifsc	VARCHAR	
Customer_cust_id	INT	NN

Branch

b_ifsc	VARCHAR	NN
b_name	VARCHAR	
b_city	VARCHAR	
Employee_emp_id	INT	

Customer

cust_id	INT	NN
cust_fname	VARCHAR	
cust_lname	VARCHAR	
cust_street	VARCHAR	
cust_city	VARCHAR	
cust_email	VARCHAR	
cust_phone	INT	
acc_no	INT	
l_id	INT	

Account

acc_no	INT	NN
acc_balance	INT	
b_ifsc	VARCHAR	
Branch_b_ifsc	VARCHAR	
Customer_cust_id	INT	NN

Transaction

tr_id	INT	NN
tr_description	VARCHAR	
tr_status	VARCHAR	
tr_amt	INT	
tr_updatedBalance	INT	
tr_date	DATETIME	
acc_no	INT	
Account_acc_no	INT	NN

SQL Queries for creation of schema :-

```
CREATE TABLE Branch (
	b_ifsc varchar(10) NOT NULL,
	b_name varchar(30),
	b_city varchar(30),
	PRIMARY KEY (b_ifsc)
);


CREATE TABLE Account (
	acc_no int NOT NULL,
	acc_balance int,
	b_ifsc varchar(30),
	PRIMARY KEY (acc_no),
	FOREIGN KEY (b_ifsc) REFERENCES Branch(b_ifsc)
);


CREATE TABLE Loan (
	l_id int NOT NULL,
	l_amt int,
	b_ifsc varchar(30),
	PRIMARY KEY (l_id),
	FOREIGN KEY (b_ifsc) REFERENCES Branch(b_ifsc)
);
```

```
CREATE TABLE Customer (
    cust_id int NOT NULL,
    cust_lName varchar(30),
    cust_fName varchar(30),
    cust_street varchar(50),
    cust_city varchar(30),
    cust_phone int,
    cust_email varchar(30),
    acc_no int,
    l_id int,
    PRIMARY KEY (cust_id),
    FOREIGN KEY (acc_no) REFERENCES Account(acc_no),
    FOREIGN KEY (l_id) REFERENCES Loan(l_i)
);

CREATE TABLE Employee (
    emp_id int NOT NULL,
    emp_lName varchar(30),
    emp_fName varchar(30),
    emp_street varchar(30),
    emp_city  varchar(50),
    emp_phone int,
    emp_email varchar(30),
    b_ifsc varchar(30),
    PRIMARY KEY (emp_id),
    FOREIGN KEY (b_ifsc) REFERENCES Branch(b_ifsc)
);
```

```
CREATE TABLE Transaction (
	acc_no int,
	tr_id int NOT NULL AUTO_INCREMENT,
	tr_description varchar(30),
	tr_amt int,
	tr_updatedBalance int,
	tr_date DateTime,
	PRIMARY KEY (tr_id),
	FOREIGN KEY (acc_no) REFERENCES Account(acc_no)
);
```

QUERIES/PROCEDURES FOR :-

Q1.Generate customers complete profile information, mentioning the current balance and loan info if any.

```
SELECT cust_id, cust_fname, cust_street, cust_city, cust_phone, cust_email, Account.acc_no,
acc_balance, Account.b_ifsc, Loan.l_id, l_amt, Loan.b_ifsc FROM Customer
LEFT JOIN Account ON Customer.acc_no = Account.acc_no
LEFT JOIN Loan ON Customer.l_id = Loan.l_id;
```

Q2.Fund transfer from one account to another account.

```
CREATE PROCEDURE transfer (IN Acc_No1 INT ,IN Acc_No2 INT, In amt INT)
BEGIN
    DECLARE a, b INT;
    DECLARE res VARCHAR(30);
    START TRANSACTION;
    SELECT acc_balance into a from Account WHERE acc_no = Acc_No1;
    SELECT acc_balance into b from Account WHERE acc_no = Acc_No2;

IF a >= amt THEN
    SET a = a - amt;
    SET b = b + amt;
    UPDATE Account SET acc_balance = a WHERE acc_no = Acc_No1;
    UPDATE Account SET acc_balance = b WHERE acc_no = Acc_No2;
    SET res = CONCAT("Transferred successfully ",amt);
    SELECT res;

    INSERT INTO Transaction (acc_no,tr_description,tr_amt,tr_updatedBalance,tr_Date)
    Values(Acc_No1, CONCAT("Amount Transferred to ",Acc_No2), amt, a, NOW());

    INSERT INTO Transaction (acc_no,tr_description,tr_amt,tr_updatedBalance,tr_Date)
    Values(Acc_No2,CONCAT("Amount Received From ",Acc_No1), amt, b, NOW());

ELSE
    SET res = "Insufficient Balance";
    SELECT res;
END IF;
     COMMIT;
END;
```

Q3. Retrieve the last month statements of any account number.

```
SELECT * FROM Transaction WHERE acc_no = <Input account no from user> && tr_Date >=
CURRENT_TIMESTAMP - INTERVAL 1 MONTH ORDER BY tr_date DESC;
```

Note :- We can take input account no from user in java run this query using PreparedStatement

Q4. Withdraw money from account.

```
CREATE PROCEDURE withdrawAmount (IN Acc_No INT, In amt INT)
BEGIN

    DECLARE bal INT;
    DECLARE output VARCHAR(30);
    START TRANSACTION;
    SELECT acc_balance from Account WHERE Account.acc_no = Acc_No INTO bal;

IF bal >= amt THEN
    SET bal = bal - amt;
    UPDATE Account SET Account.acc_balance = bal WHERE Account.acc_no = Acc_No;
    INSERT INTO Transaction (acc_no,tr_description,tr_amt,tr_updatedBalance,tr_Date)
    Values(Acc_No,CONCAT("Withdraw ",amt), amt, bal, NOW());
    SET output = CONCAT("Withdraw successfully ",amt);
    SELECT output;

ELSE
    SET output = "Insufficient Balance";
    SELECT output;

END IF;
    COMMIT;
END;
```

Q5. Deposit amount to any account no.

```
CREATE PROCEDURE deposit (IN Acc_No INT, In amt INT)

BEGIN
    DECLARE bal INT;
    DECLARE output VARCHAR(30);
    START TRANSACTION;
    SELECT acc_balance from Account WHERE Account.acc_no = Acc_No INTO bal;
    SET bal = bal + amt;
    UPDATE Account SET Account.acc_balance = bal WHERE Account.acc_no = Acc_No;
    INSERT INTO Transaction (acc_no, tr_description, tr_amt, tr_updatedBalance, tr_Date)
    Values(Acc_No,,CONCAT("Deposit ",amt), amt, bal, NOW());
    SET output = CONCAT("Deposit Successfully ",amt);
    SELECT output;
    COMMIT;
END;
```