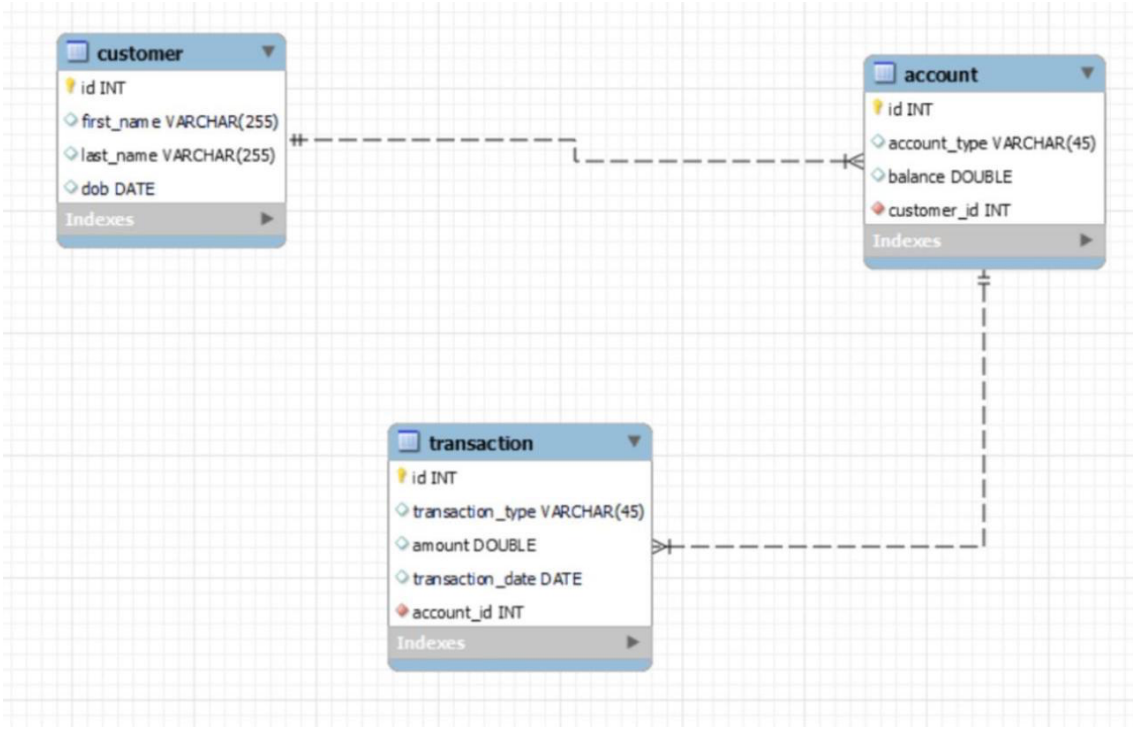
Project- Student Information System

ER Diagram:



Code:

```
create database learnings3;
use learnings3;

CREATE TABLE Customers (
   customer_id INT PRIMARY KEY AUTO_INCREMENT,
   first_name VARCHAR(50),
   last_name VARCHAR(50),
   DOB DATE,
   email VARCHAR(100),
   phone_number VARCHAR(15),
   address VARCHAR(255)
);

CREATE TABLE Accounts (
   account_id INT PRIMARY KEY AUTO_INCREMENT,
   customer_id INT,
```

```
    account_type VARCHAR(20),
    balance DECIMAL(10, 2),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);

CREATE TABLE Transactions (
    transaction_id INT PRIMARY KEY AUTO_INCREMENT,
    account_id INT,
    transaction_type VARCHAR(20),
    amount DECIMAL(10, 2),
    transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);


INSERT INTO Customers (first_name, last_name, DOB, email, phone_number, address)
VALUES
    ('Venkatesh', 'Reddy', '1980-01-15', 'venkatesh.reddy@example.com', '1234567890', '123 Main St'),
    ('Lakshmi', 'Kumar', '1992-05-20', 'lakshmi.kumar@example.com', '0987654321', '456 Elm St'),
    ('Rama', 'Raju', '1975-11-03', 'rama.raju@example.com', '5551234567', '789 Oak St'),
    ('Suresh', 'Gupta', '1988-09-10', 'suresh.gupta@example.com', '1112223333', '987 Maple St'),
    ('Padma', 'Singh', '1996-03-28', 'padma.singh@example.com', '4445556666', '654 Birch St'),
    ('Raju', 'Prasad', '1983-07-18', 'raju.prasad@example.com', '7778889999', '123 Cedar St'),
    ('Anitha', 'Sharma', '1990-12-02', 'anitha.sharma@example.com', '2223334444', '1010 Walnut St'),
    ('Prasad', 'Patel', '1987-04-05', 'prasad.patel@example.com', '9998887777', '888 Oak St'),
    ('Vijaya', 'Naidu', '1995-06-17', 'vijaya.naidu@example.com', '6667778888', '555 Pine St'),
    ('Kumar', 'Rao', '1978-08-22', 'kumar.rao@example.com', '3334445555', '222 Elm St');


INSERT INTO Accounts (customer_id, account_type, balance)
VALUES
    (1, 'savings', 1000.00),
    (2, 'current', 500.00),
    (3, 'savings', 2000.00),
    (4, 'current', 1500.00),
    (5, 'savings', 3000.00),
    (6, 'current', 200.00),
    (7, 'savings', 500.00),
    (8, 'current', 750.00),
    (9, 'savings', 1500.00),
    (10, 'current', 10000.00);


INSERT INTO Transactions (account_id, transaction_type, amount, transaction_date)
VALUES
    (1, 'deposit', 500.00, NOW()),
    (2, 'deposit', 1000.00, NOW()),
    (3, 'deposit', 200.00, NOW()),
    (4, 'withdrawal', 300.00, NOW()),
    (5, 'deposit', 800.00, NOW()),
    (6, 'withdrawal', 50.00, NOW()),
    (7, 'deposit', 200.00, NOW()),
    (8, 'withdrawal', 100.00, NOW()),
    (9, 'deposit', 400.00, NOW()),
```

```
   (10, 'withdrawal', 1000.00, NOW());


-- Basic commands

SELECT c.first_name, c.last_name, a.account_type, c.email
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id;

SELECT c.first_name, c.last_name, t.transaction_id, t.transaction_type, t.amount, t.transaction_date
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
JOIN Transactions t ON a.account_id = t.account_id;


UPDATE Accounts
SET balance = balance + 4000
WHERE account_id = 4;

SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM Customers;

SELECT *
FROM Customers
WHERE address LIKE '%Elm%';

SELECT balance
FROM Accounts
WHERE account_id = 6;

SELECT *
FROM Accounts
WHERE account_type = 'current' AND balance > 1000.00;

SELECT *
FROM Transactions
WHERE account_id = 2;

SELECT *
FROM Accounts
WHERE balance < 1000;

-- Aggregate Functions, Joins



-- Output: Average balance of all accounts.
SELECT AVG(balance) AS average_balance
FROM Accounts;

-- Output: Top 10 accounts with the highest balance.
SELECT *
FROM Accounts
```

```
ORDER BY balance DESC
LIMIT 10;

-- Output: Total deposits made on the current date.
SELECT SUM(amount) AS total_deposits
FROM Transactions
WHERE transaction_type = 'deposit'
AND DATE(transaction_date) = '2024-03-08';

-- Output: Oldest customer (by date of birth).
SELECT first_name, last_name, DOB
FROM Customers
ORDER BY DOB ASC LIMIT 1; -- Oldest

-- Output: Newest customer (by date of birth).
SELECT first_name, last_name, DOB
FROM Customers
ORDER BY DOB DESC LIMIT 1; -- Newest

-- Output: Transactions with associated account types.
SELECT t.*, a.account_type
FROM Transactions t
JOIN Accounts a ON t.account_id = a.account_id;

-- Output: Customers and their accounts.
SELECT c.*, a.*
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id;

-- Output: Customer details and associated transaction details for a specific account.
SELECT c.*, t.*
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
JOIN Transactions t ON a.account_id = t.account_id
WHERE a.account_id = 3;

-- Output: Customers who have more than one account.
SELECT c.*
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
GROUP BY c.customer_id
HAVING COUNT(*) > 1;

-- Output: Net amount (deposit - withdrawal) for each account.
SELECT account_id, SUM(CASE WHEN transaction_type = 'deposit' THEN amount ELSE -amount END) AS
net_amount
FROM Transactions
GROUP BY account_id;

-- Output: Average daily balance for each account.
SELECT account_id, AVG(balance) AS average_daily_balance
FROM Accounts
GROUP BY account_id;
```

```sql
-- Output: Total balance for each account type.
SELECT account_type, SUM(balance) AS total_balance
FROM Accounts
GROUP BY account_type;

-- Output: Transaction count for each account.
SELECT account_id, COUNT(*) AS transaction_count
FROM Transactions
GROUP BY account_id
ORDER BY transaction_count DESC;

-- Output: Transactions with duplicate amount, date, and account_id.
SELECT *
FROM Transactions
WHERE (amount, transaction_date, account_id) IN (
   SELECT amount, transaction_date, account_id
   FROM Transactions
   GROUP BY amount, transaction_date, account_id
   HAVING COUNT(*) > 1
);


-- SubQueries


-- Output: Customer details for the customer(s) with the maximum account balance.
SELECT c.*
FROM Customers c
JOIN Accounts a ON c.customer_id = a.customer_id
WHERE a.balance = (
   SELECT MAX(balance) FROM Accounts
);

-- Output: Average balance for customers who have more than one account.
SELECT AVG(balance) AS average_balance
FROM Accounts
WHERE customer_id IN (
   SELECT customer_id
   FROM Accounts
   GROUP BY customer_id
   HAVING COUNT(*) > 1
);

-- Output: Transactions with amount greater than the average transaction amount.
SELECT account_id, transaction_type, amount
FROM Transactions
WHERE amount > (
   SELECT AVG(amount)
   FROM Transactions
);

-- Output: Customers who have not made any transactions.
SELECT *
FROM Customers
```

```
WHERE customer_id NOT IN (
   SELECT DISTINCT customer_id
   FROM Transactions
);

-- Output: Total balance of accounts without associated transactions.
SELECT SUM(balance) AS total_balance
FROM Accounts
WHERE account_id NOT IN (
   SELECT DISTINCT account_id
   FROM Transactions
);

-- Output: Transactions associated with the account having the minimum balance.
SELECT *
FROM Transactions
WHERE account_id IN (
   SELECT account_id
   FROM Accounts
   WHERE balance = (
      SELECT MIN(balance)
      FROM Accounts
   )
);

-- Output: Customer IDs who have multiple account types.
SELECT customer_id
FROM Accounts
GROUP BY customer_id
HAVING COUNT(DISTINCT account_type) > 1;

-- Output: Percentage of each account type relative to the total number of accounts.
SELECT account_type,
     COUNT(*) * 100.0 / (SELECT COUNT(*) FROM Accounts) AS percentage
FROM Accounts
GROUP BY account_type;

-- Output: Transactions associated with the accounts of a specific customer.
SELECT *
FROM Transactions
WHERE account_id IN (
   SELECT account_id
   FROM Accounts
   WHERE customer_id = 4
);

-- Output: Total balance for each account type.
SELECT account_type,
     (SELECT SUM(balance) FROM Accounts WHERE account_type = a.account_type) AS total_balance
FROM Accounts a
GROUP BY account_type;
```