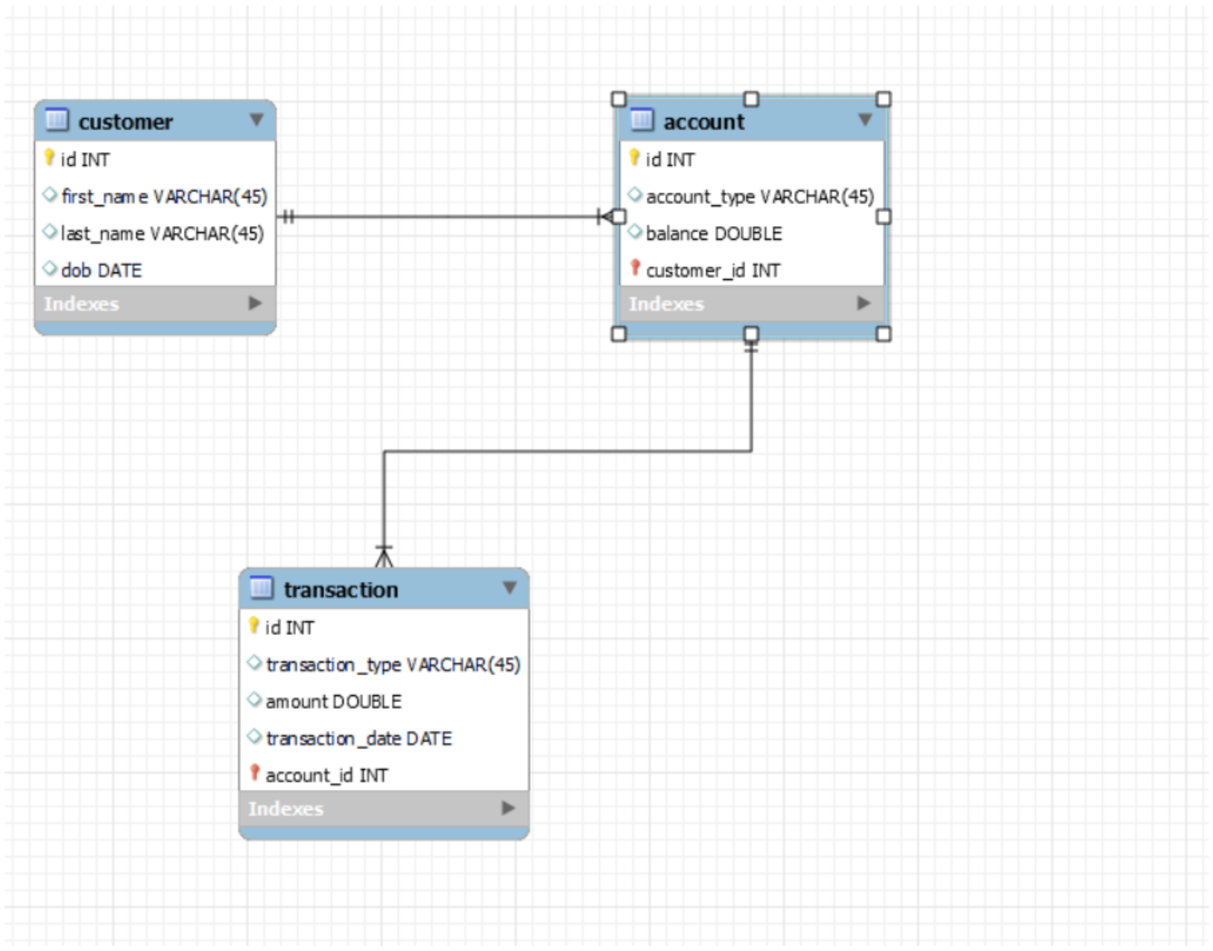
# ASSIGNMENT NO : 2

## Banking System

**ER DIAGRAM:**



**Task:1. Database Design:**

```
-- MySQL Workbench Forward Engineering


-- -----------------------------------------------------
-- Schema banking
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema banking
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `banking` DEFAULT CHARACTER SET utf8 ;
USE `banking` ;

-- -----------------------------------------------------
-- Table `banking`.`customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `banking`.`customer` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NULL,
  `last_name` VARCHAR(45) NULL,
  `dob` DATE NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `banking`.`account`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `banking`.`account` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `account_type` VARCHAR(45) NULL,
  `balance` DOUBLE NULL,
  `customer_id` INT NOT NULL,
  PRIMARY KEY (`id`, `customer_id`),
  INDEX `fk_account_customer_idx` (`customer_id` ASC) ,
  CONSTRAINT `fk_account_customer`
    FOREIGN KEY (`customer_id`)
    REFERENCES `banking`.`customer` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `banking`.`transaction`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `banking`.`transaction` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `transaction_type` VARCHAR(45) NULL,
  `amount` DOUBLE NULL,
  `transaction_date` DATE NULL,
  `account_id` INT NOT NULL,
  PRIMARY KEY (`id`, `account_id`),
  INDEX `fk_transaction_account1_idx` (`account_id` ASC) ,
  CONSTRAINT `fk_transaction_account1`
    FOREIGN KEY (`account_id`)
    REFERENCES `banking`.`account` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

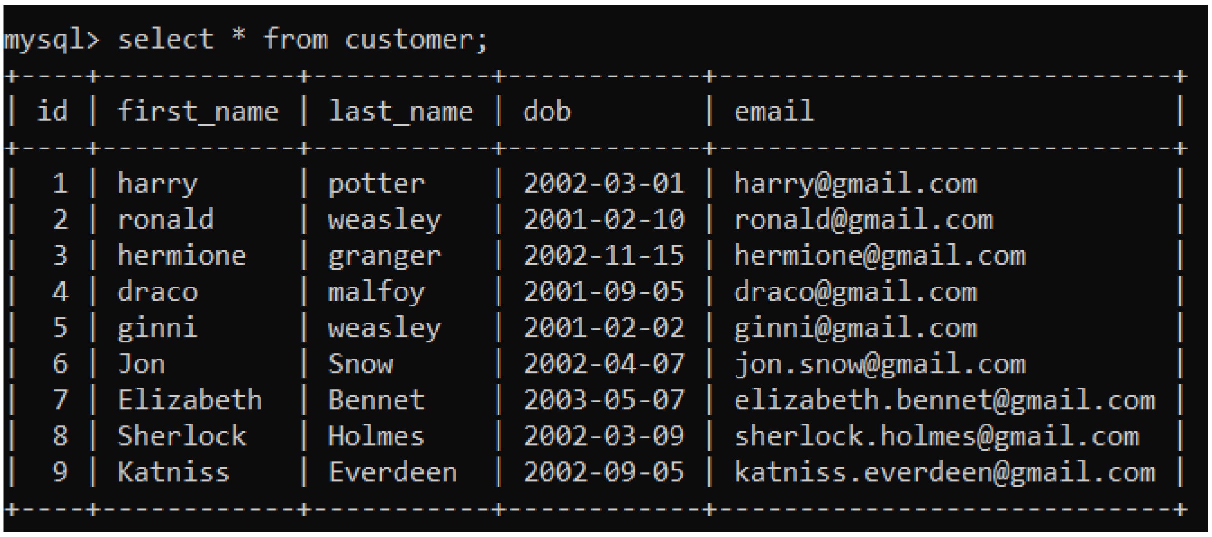
## Tasks 2: Select, Where, Between, AND, LIKE:

1. Insert at least 10 sample records into each of the following tables.

- Customers
- Accounts
- Transactions

-- customer insertion

```
insert into customer(first_name,last_name,dob) values
('harry','potter','2002-03-21'),
('ronald','weasley','2001-02-10'),
('hermione','granger','2002-11-15'),
('draco','malfoy','2001-09-05','draco@gmail.com'),
('ginni','weasley','2001-02-02','ginni@gmail.com'),
('Jon', 'Snow','2002-04-07','jon.snow@gmail.com'),
('Elizabeth' ,'Bennet','2003-05-07', 'elizabeth.bennet@gmail.com'),
('Sherlock','Holmes','2002-03-09','sherlock.holmes@gmail.com'),
('Katniss' ,'Everdeen','2002-09-05','katniss.everdeen@gmail.com');
```



```
mysql> select * from customer;
+----+------------+-----------+------------+----------------------------+
| id | first_name | last_name | dob        | email                      |
+----+------------+-----------+------------+----------------------------+
|  1 | harry      | potter    | 2002-03-01 | harry@gmail.com            |
|  2 | ronald     | weasley   | 2001-02-10 | ronald@gmail.com           |
|  3 | hermione   | granger   | 2002-11-15 | hermione@gmail.com         |
|  4 | draco      | malfoy    | 2001-09-05 | draco@gmail.com            |
|  5 | ginni      | weasley   | 2001-02-02 | ginni@gmail.com            |
|  6 | Jon        | Snow      | 2002-04-07 | jon.snow@gmail.com         |
|  7 | Elizabeth  | Bennet    | 2003-05-07 | elizabeth.bennet@gmail.com |
|  8 | Sherlock   | Holmes    | 2002-03-09 | sherlock.holmes@gmail.com  |
|  9 | Katniss    | Everdeen  | 2002-09-05 | katniss.everdeen@gmail.com |
+----+------------+-----------+------------+----------------------------+
```

-- account insertion

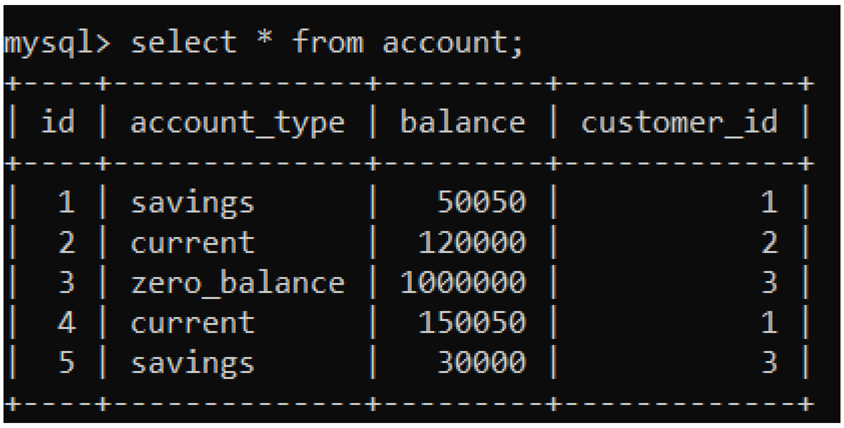
```
insert into account(account_type,balance,customer_id) values
('savings',50000,1) ,
```

('current',120000,2) ,

('zero_balance',100000,3),

('current',150000,1) ,

('savings',30000,3);



```
mysql> select * from account;
+----+--------------+---------+-------------+
| id | account_type | balance | customer_id |
+----+--------------+---------+-------------+
|  1 | savings      |   50050 |           1 |
|  2 | current      |  120000 |           2 |
|  3 | zero_balance | 1000000 |           3 |
|  4 | current      |  150050 |           1 |
|  5 | savings      |   30000 |           3 |
+----+--------------+---------+-------------+
```

-- transaction insertion

insert into transaction(transaction_type,amount,transaction_date,account_id)

values

('deposit', 10000, '2024-02-01',1),

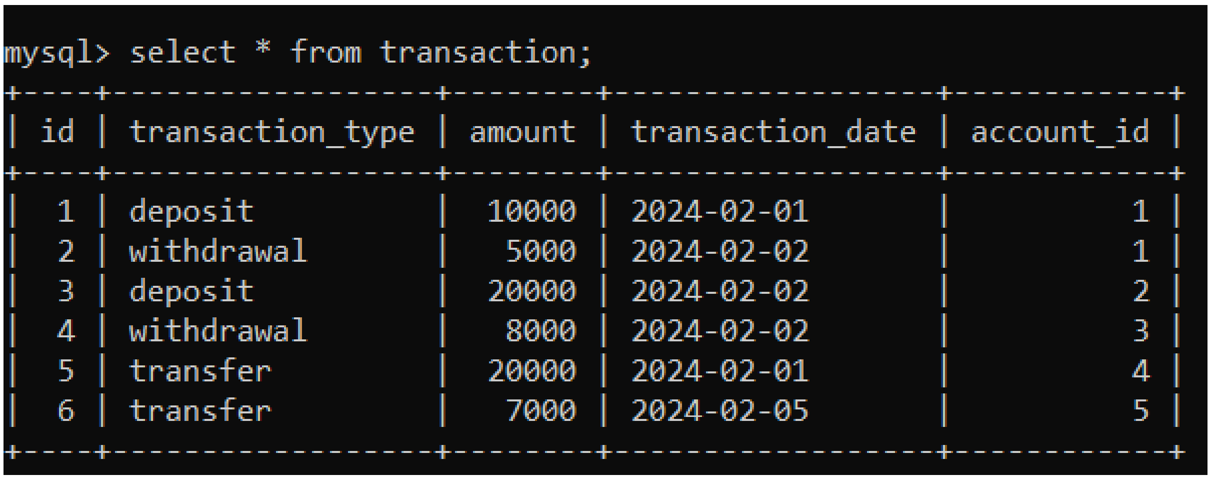
('withdrawal', 5000, '2024-02-02',1),

('deposit', 20000, '2024-02-02',2),

('withdrawal', 8000, '2024-02-02',3),

('transfer', 20000, '2024-02-01',4),

('transfer', 7000, '2024-02-05',5);



```
mysql> select * from transaction;
+----+------------------+--------+------------------+------------+
| id | transaction_type | amount | transaction_date | account_id |
+----+------------------+--------+------------------+------------+
|  1 | deposit          |  10000 | 2024-02-01       |          1 |
|  2 | withdrawal       |   5000 | 2024-02-02       |          1 |
|  3 | deposit          |  20000 | 2024-02-02       |          2 |
|  4 | withdrawal       |   8000 | 2024-02-02       |          3 |
|  5 | transfer         |  20000 | 2024-02-01       |          4 |
|  6 | transfer         |   7000 | 2024-02-05       |          5 |
+----+------------------+--------+------------------+------------+
```

2. Write SQL queries for the following tasks:

1. Write a SQL query to retrieve the name, account type and email of all customers.

```
select c.first_name, c.last_name, a.account_type from customer c
join account a ON c.id = a.customer_id;
```

2. Write a SQL query to list all transaction corresponding customer.

```
select t.id AS transaction_id, t.transaction_type, t.amount, t.transaction_date,
c.id AS customer_id, c.first_name, c.last_name   from transaction t
join account a ON t.account_id = a.id join customer c ON a.customer_id = c.id;
```

3. Write a SQL query to increase the balance of a specific account by a certain amount.

```
update account set balance = balance + 500 where id = 1;
```

4. Write a SQL query to Combine first and last names of customers as a full_name.

```
select CONCAT(first_name, ' ', last_name) as full_name from customer;
```

5. Write a SQL query to remove accounts with a balance of zero where the account type is savings.

```
Delete from account where balance = 0 and account_type = 'savings';
```

6. Write a SQL query to Find customers living in a specific city.

```
select c.id, c.first_name, c.last_name from customer c
join address a on c.id = a.customer_id where a.city = 'chennai';
```

7. Write a SQL query to Get the account balance for a specific account.

```
select balance from account where id = 1;
```

8. Write a SQL query to List all current accounts with a balance greater than $1,000.

```
select * from account where account_type = 'current' and balance > 1000;
```

9. Write a SQL query to Retrieve all transactions for a specific account.

```
select* from transaction where account_id = 2;
```

10. Write a SQL query to Calculate the interest accrued on savings accounts based on a given interest rate.

```
select id, balance * (interest_rate / 100) as interest_accrued from account where
account_type = 'savings';
```

11. Write a SQL query to Identify accounts where the balance is less than a specified overdraft limit.

```
select * from account where balance < specified_overdraft_limit;
```

12. Write a SQL query to Find customers not living in a specific city.

```
select * from customer where city != 'chennai';
```

## Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:

1. Write a SQL query to Find the average account balance for all customers.

```
select customer_id, AVG(balance) from account group by customer_id;
```

2. Write a SQL query to Retrieve the top 10 highest account balances.

```
select balance  from account order by balance DESC limit 0,3;
```

3. Write a SQL query to Calculate Total Deposits for All Customers in specific date.

```
select c.first_name,c.last_name,t.transaction_type, t.amount, t.transaction_date
from transaction t JOIN account a ON a.id = t.account_id JOIN customer c ON c.id =
a.customer_id where t.transaction_date = '2024-02-02' AND
t.transaction_type='withdrawal';
```

4. Write a SQL query to Find the Oldest and Newest Customers.

```
        (select first_name,dob,'oldest' as status from customer order by dob limit 0,1) UNION
(select first_name,dob,'youngest' as status from customer order by dob DESC limit 0,1);
```

5. Write a SQL query to Retrieve transaction details along with the account type.

```
select t.*, a.account_type from transaction t join account a on t.account_id = a.id;
```

6. Write a SQL query to Get a list of customers along with their account details.

```
select c.*, a.account_type, a.balance from customer c join account a on c.id = a.customer_id;
```

7. Write a SQL query to Retrieve transaction details along with customer information for a specific account.

```
select t.*, c.first_name, c.last_name from transaction t join account a on t.account_id = a.id
join customer c on a.customer_id = c.id where a.id=1;
```

8. Write a SQL query to Identify customers who have more than one account.

```
select c.id, c.first_name, c.last_name, count(a.id) as number_of_accounts from customer c
join account a on c.id = a.customer_id group by c.id having count(a.id) > 1;
```

9. Write a SQL query to Calculate the difference in transaction amounts between deposits and withdrawals.

```
select MAX(amount) - MIN(amount) as difference from ((select transaction_type
,SUM(amount) as amount, 'deposit' as op from transaction where transaction_type
='deposit' )  union (select transaction_type , SUM(amount) as amount, 'withdrawal' as op
from transaction where transaction_type ='withdrawal')) AS T;
```

10. Write a SQL query to Calculate the average daily balance for each account over a specified period.

```
select account_id, avg(daily_balance) as average_daily_balance from balance_history
where transaction_date between start_date and end_date group by account_id;
```

11. Calculate the total balance for each account type.

```
select account_type, sum(balance) as total_balance from account group by account_type;
```

12. Identify accounts with the highest number of transactions order by descending order.

```
select account_id, count(id) as number_of_transactions from transaction
group by account_id order by number_of_transactions desc;
```

13. List customers with high aggregate account balances, along with their account types.

```
select c.id, c.first_name, c.last_name, a.account_type, sum(a.balance) as aggregate_balance
from customer c join account a on c.id = a.customer_id group by c.id, c.first_name,
c.last_name, a.account_typehaving sum(a.balance) > 10000  order by aggregate_balance
desc;
```

14. Identify and list duplicate transactions based on transaction amount, date, and account.

```
select transaction_type, amount, transaction_date, account_id, count(*) as duplicates
from transaction group by transaction_type, amount, transaction_date, account_id
having count(*) > 1 order by duplicates desc, account_id;
```

## Tasks 4: Subquery and its type:

1. Retrieve the customer(s) with the highest account balance.

```
select avg(balance) from account where customer_id IN (select customer_id
from account group by customer_id having count(id) > 1);
```

2. Calculate the average account balance for customers who have more than one account.

```
select customer_id, avg(balance) as average_balance from account group by customer_id
having count(id) > 1;
```

3. Retrieve accounts with transactions whose amounts exceed the average transaction amount.

```
select a.* from account a join transaction t on a.id = t.account_id where t.amount > (
select avg(amount) from transaction) group by a.id;
```

4. Identify customers who have no recorded transactions.

```
select id,first_name from customer where id IN (select customer_id from account where id
NOT IN (select account_id from transaction));
```

5. Calculate the total balance of accounts with no recorded transactions.

```
select sum(a.balance) as total_balance_of_inactive_accounts from account a
left join transaction t on a.id = t.account_id where t.id is null;
```

6. Retrieve transactions for accounts with the lowest balance.

```
select t.* from transaction t join account a on t.account_id = a.id
where a.balance = (select min(balance) from account);
```

7. Identify customers who have accounts of multiple types.

```
select c.id, c.first_name, c.last_name, count(distinct a.account_type) as types_of_accounts
from customer c join account a on c.id = a.customer_id group by c.id
having count(distinct a.account_type) > 1;
```

8. Calculate the percentage of each account type out of the total number of accounts.

```
select account_type, count(id) as number_of_accounts, (count(id) / (select count(id) from
account) * 100) as percentage_of_total from account group by account_type;
```

9. Retrieve all transactions for a customer with a given customer_id.

```
select * from transaction where account_id IN (select id from account where
customer_id=1);
```

10. Calculate the total balance for each account type, including a subquery within the SELECT clause

```
select account_type, SUM(balance) as total_balance from account group by account_type;
```