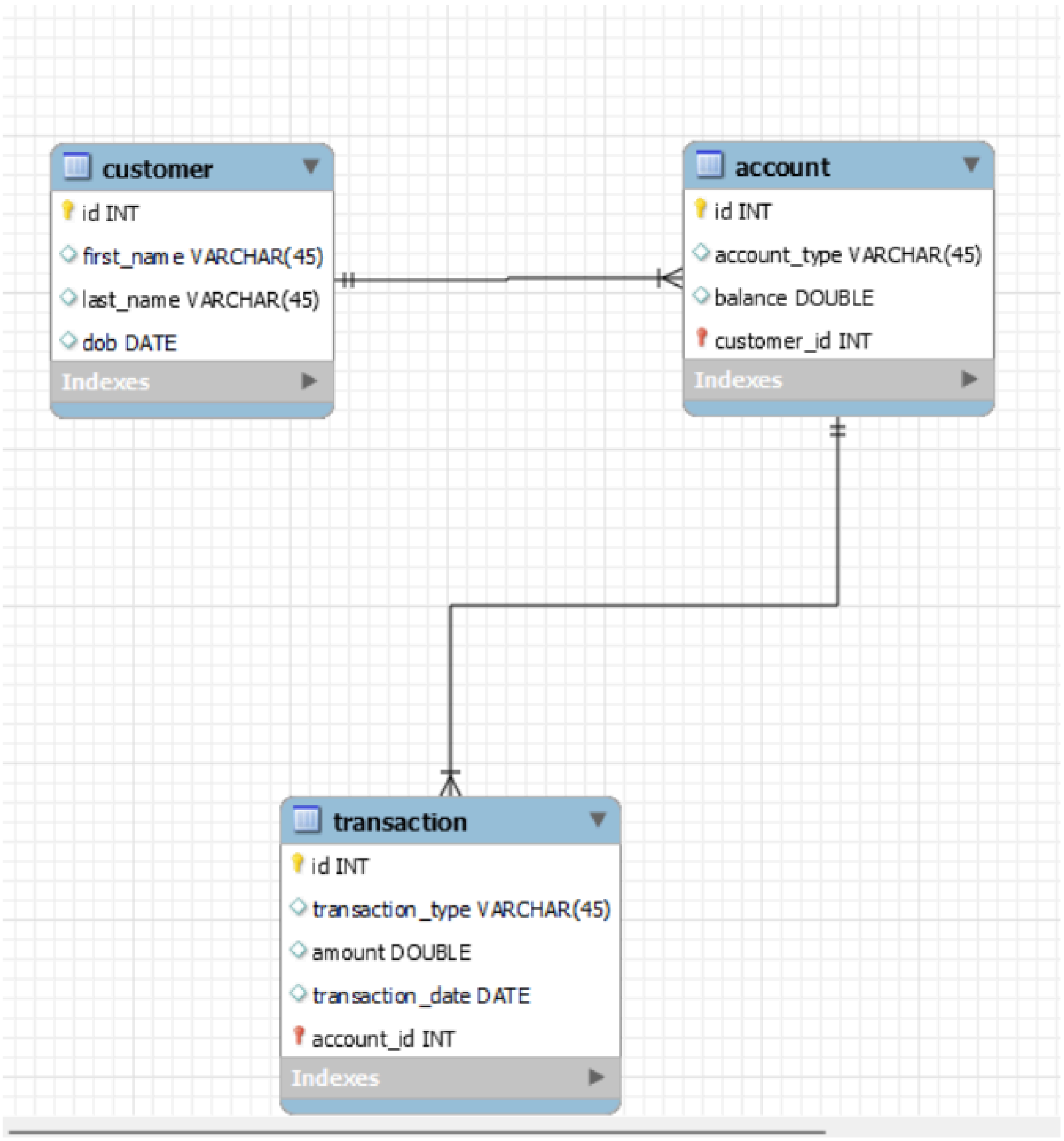
Banking System case study:

ER DIAGRAM :



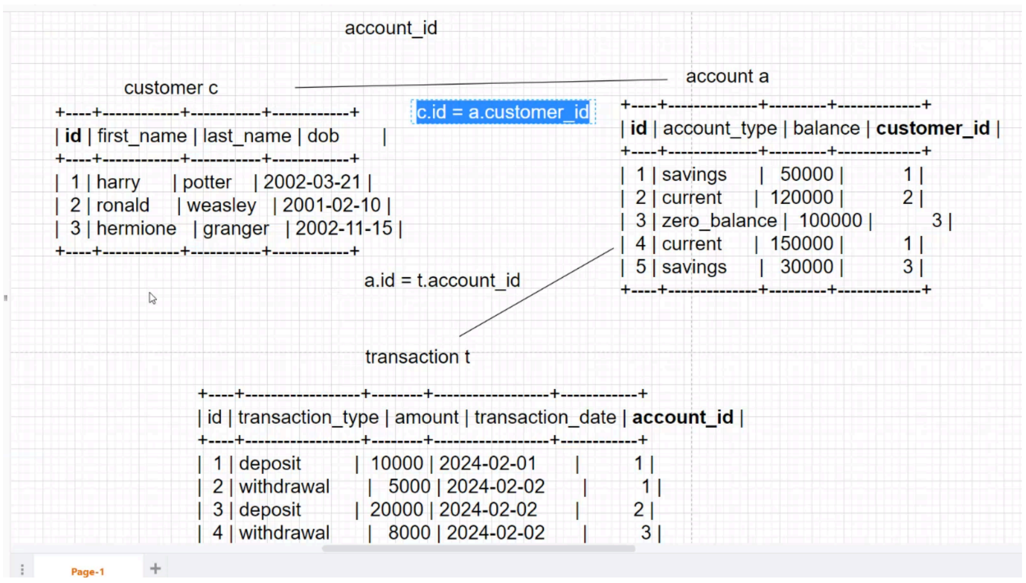
#database scripts

```
-- MySQL Workbench Forward Engineering



-- -----------------------------------------------------
-- Schema bank_hex_feb_24
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema bank_hex_feb_24
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `bank_hex_feb_24` DEFAULT CHARACTER SET utf8 ;
USE `bank_hex_feb_24` ;

-- -----------------------------------------------------
-- Table `bank_hex_feb_24`.`customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bank_hex_feb_24`.`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 `bank_hex_feb_24`.`account`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bank_hex_feb_24`.`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 `bank_hex_feb_24`.`customer` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `bank_hex_feb_24`.`transaction`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bank_hex_feb_24`.`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 `bank_hex_feb_24`.`account` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

#insertions :

```
use bank_hex_feb_24;
show tables;
describe customer;
describe account;
describe transaction;
#insertions

insert into customer (first_name, last_name, dob) values
('harry', 'potter','2002-03-01'),
('ronald', 'weasley','2001-02-10'),
('hermione', 'granger','2002-11-15');
select * from customer;

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);
select * from account;

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);

select * from transaction;
```

Reference Diagram :



```
+----+------------+-----------+------------+
| id | first_name | last_name | dob        |
+----+------------+-----------+------------+
|  1 | harry      | potter    | 2002-03-21 |
|  2 | ronald     | weasley   | 2001-02-10 |
|  3 | hermione   | granger   | 2002-11-15 |
+----+------------+-----------+------------+
```

```
+----+--------------+---------+-------------+
| id | account_type | balance | customer_id |
+----+--------------+---------+-------------+
|  1 | savings      |   50000 |           1 |
|  2 | current      |  120000 |           2 |
|  3 | zero_balance |  100000 |           3 |
|  4 | current      |  150000 |           1 |
|  5 | savings      |   30000 |           3 |
+----+--------------+---------+-------------+
```

```
+----+------------------+--------+------------------+------------+
| 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 |
```

TASK 2:

-- 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 c.first_name , t.transaction_type , t.transaction_date , t.amount
from transaction t join account a on a.id=t.account_id join customer c on
c.id=a.customer_id;

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

select id ,(100+balance) as Increased_amount from account
where account_type='savings';

-- 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 first_name , last_name from customer where city='chennai'; -- city is not found in database

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

select balance from account where id =3;

-- 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=1;
```

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

-- Interest Accrued = Principal Balance × Interest Rate

-- Interest rate = 0.05

```
select  balance * 0.05  as Interest_accured from account
where account_type='savings';
```

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

-- overdraft limit=1000

```
select id , balance from account
where balance <1000000;
```

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

```
select first_name , last_name from customer where city !='chennai'; -- city is not found in
database
```

-- Task 3

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

```
select c.first_name as customer_name,avg(balance)
 from customer c join account a on c.id = a.customer_id
 group by c.first_name;
/*
+---------------+--------------+
| customer_name | avg(balance) |
+---------------+--------------+
| harry         |       100000 |
| hermione      |        65000 |
| ronald        |       120000 |
+---------------+--------------+
*/
```

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

```
-- here i am taking top 3
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. Also display name of the customer */

```
select c.first_name,c.last_name,t.transaction_type, t.amount, t.transaction_date
from customer c join account a on c.id = a.customer_id join transaction t on
a.id=t.account_id
where t.transaction_type='deposit' AND t.transaction_date='2024-02-01';
```

/* 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);
/*
O/P:
+------------+------------+----------+
| first_name | dob        | status   |
+------------+------------+----------+
| ronald     | 2001-02-10 | oldest   |
| hermione   | 2002-11-15 | youngest |
+------------+------------+----------+
*/
```

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

```
select t.transaction_type , t.amount,t.transaction_date , t.account_id , 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.first_name , a.account_type, a.balance , a.customer_id
from account a  join customer c 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.transaction_type , t.amount,t.transaction_date , t.account_id ,c.first_name ,
c.last_name , c.dob
from transaction t join account a on t.account_id = a.id join customer c on c.id =
a.customer_id
where account_type='savings';
```

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

```
select c.first_name , count(c.id) as No_of_accounts
from customer c join account a on c.id = a.customer_id
-- -- where count(c.id) > 1 - 0 Invalid use of group function
group by c.id
having No_of_accounts >1;
/*
a.customer_id=1 (2)
        1       harry   potter   2002-03-21     1       savings         50000  1
        1       harry   potter  2002-03-21      4       current 150000          1
a.customer_id=2 (1)
        2       ronald  weasley         2001-02-10      2       current 120000          2
a.customer_id=3 (2)
        3       hermione        granger         2002-11-15      3       zero_balance  100000
        3
        3       hermione        granger         2002-11-15      5       savings         30000  3
*/
```

/*

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

***/**

```
Select max(difference) - min(difference ) as Difference_transaction
from
((select  transaction_type , sum(amount) as difference
from transaction
where transaction_type ='deposit')
UNION
(select transaction_type , sum(amount) as difference
from transaction
where transaction_type='withdrawal') ) as T ;
```

-- alternatively

```
select
(select SUM(amount)
from transaction
where transaction_type ='deposit' ) -  (select  SUM(amount)
from transaction
where transaction_type ='withdrawal') as diff;
```

/*

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

***/**

-- step 1 Write a SQL query to Calculate the average daily balance for each account

```
select a.account_type, AVG(a.balance) as Average_balance
from account a join transaction t on a.id=t.account_id
group by a.account_type;
```

-- step 2 Write a SQL query to Calculate the average daily balance for each account over a specified period.

```
select a.account_type, AVG(a.balance) as Average_balance
from account a join transaction t on a.id=t.account_id
where t.transaction_date between '2024-02-01' AND '2024-02-05'
group by a.account_type;
```

-- 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  t.account_id , count(t.id) as No_of_transaction
from transaction t join account a on t.account_id= a.id
group by t.account_id
order by No_of_transaction desc
limit 0,1;
```

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

-- step 1 fetching all the customers along with account type and getting their sum

```
select c.first_name, a.account_type , sum(a.balance) as account_balance
from customer c join account a on c.id=a.customer_id
group by a.account_type;
/*
output
first_name account_type acc_balance
ronald      current       270000
harry       savings       80000
hermione        zero_balance  100000
*/
```

-- getting highest account_balance

```
select c.first_name, a.account_type , sum(a.balance) as account_balance
from customer c join account a on c.id=a.customer_id
group by a.account_type
order by account_balance desc
limit 0 ,1;
```

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

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

select * from transaction;
```

TASK 4:

-- Tasks 4: Subquery and its type:

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

```
select id , first_name
from customer where id in ( select customer_id from account where balance =( select
max(balance) from account) );
```

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

-- find customers having more than 1 account
```
select customer_id
from account
group by customer_id
having count(customer_id)>1;
```

-- find avg account balance for all customers
```
select avg(balance) from account ;
```
-- for specific customer from above query
```
select customer_id ,avg(balance)
from account
where customer_id in (select customer_id
from account
group by customer_id
having count(customer_id)>1);
```

-- 3. Retrieve accounts with transactions whose amounts exceed the average transaction amount.
```
select account_id , id as transaction_id, transaction_type
from transaction where amount> (select avg(amount) from transaction);
```

-- 4. Identify customers who have no recorded transactions.
```
insert into customer(first_name,last_name,dob) values ('draco','malfoy','2000-05-06');
insert into account(account_type,balance,customer_id) values ('zero_balance',40000,4);
select * from customer;
select * from account;

select id, first_name
from customer where id in (select customer_id from account where id not in
(select account_id from transaction ));
```

-- troubleshooting
```
select distinct account_id from transaction; -- (1,2,3,4,5)
select customer_id from account where id NOT IN (1,2,3,4,5); -- (6)
select * from customer where id IN (4);
```

-- 5. Calculate the total balance of accounts with no recorded transactions.
```
select id ,sum(balance) as total_balance
from account where id not in (select account_id from transaction);
```

-- 6. Retrieve transactions for accounts with the lowest balance.
```
select id , transaction_type, account_id
from transaction where account_id in ( select  id from account
 where balance = (select min(balance) from account));
```
-- 7. Identify customers who have accounts of multiple types.
```
select id , first_name
from customer where id in
```

(select id from account group by account_type having count(distinct(id))>1);
-- 8. Calculate the percentage of each account type out of the total number of accounts.
Select account_type , count(*) as total_acccount,
((count(*) * 100.0) / (SELECT COUNT(*) FROM account)) AS percentage
from account group by account_type ;
-- if we want to round the values we can
Select account_type , count(*) as total_acccount,
round((count(*) * 100.0) / (SELECT COUNT(*) FROM account),2) AS percentage
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
-- In Select, we can do arithmetic operations. If sub query returns multiple rows then it cannot be written in select.

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