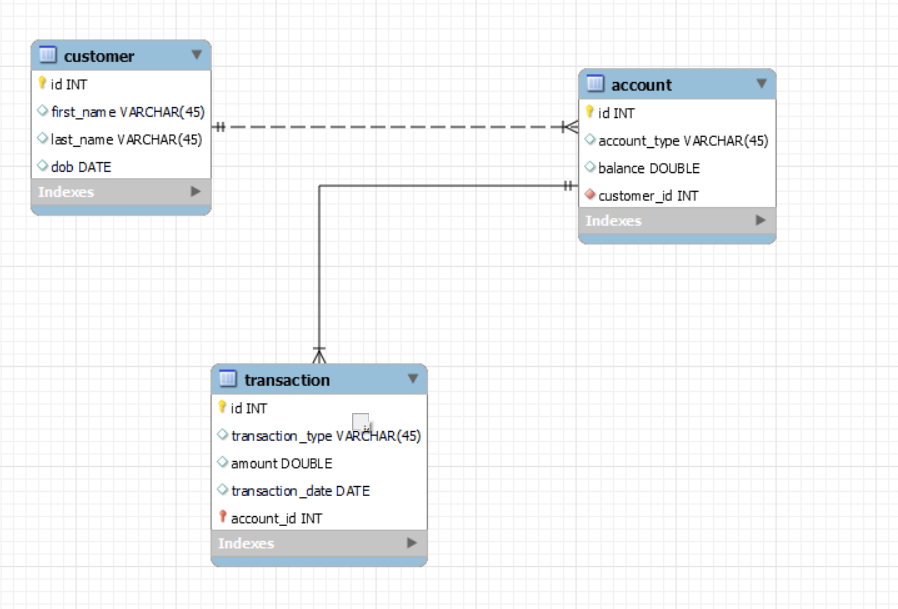
ASSIGNMENT 3 – BANKING ASSIGNMENT



-- 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`),

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;

insert into customer(first\_name,last\_name,dob) values

('harry','potter','2002-03-21'),

('ronald','weasley','2001-02-10'),

('hermione','granger','2002-11-15');

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);

--------------------------------------------------

Database changed

| Tables\_in\_bank\_hex\_feb\_24 |

| account |

| customer |

| transaction |

---------------------------------------------------

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);

Describe customer;

+------------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+------------+-------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| first\_name | varchar(45) | YES | | NULL | |

| last\_name | varchar(45) | YES | | NULL | |

| dob | date | YES | | NULL | |

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);

describe account;

+--------------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+--------------+-------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| account\_type | varchar(45) | YES | | NULL | |

| balance | double | YES | | NULL | |

| customer\_id | int(11) | NO | MUL | NULL | |

+--------------+-------------+------+-----+---------+----------------+

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);

describe transaction;

+------------------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+------------------+-------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| transaction\_type | varchar(45) | YES | | NULL | |

| amount | double | YES | | NULL | |

| transaction\_date | date | YES | | NULL | |

| account\_id | int(11) | NO | PRI | NULL | |

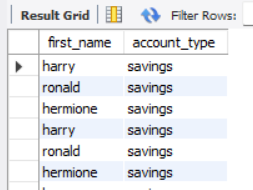
+------------------+-------------+------+-----+---------+----------------+

**TASK 2**

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

select c.first\_name,a.account\_type

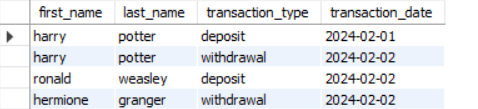
from customer c, account a;



-- 2. write a sql query to list all transaction corresponding customer.

select c.first\_name,c.last\_name,t.transaction\_type,t.transaction\_date

from customer c join account a on c.id=a.customer\_id join transaction t on a.id=t.account\_id;



-- 3. write a sql query to increase the balance of a specific account by a certain amount.

update account

set balance = balance + 100

where id=1;

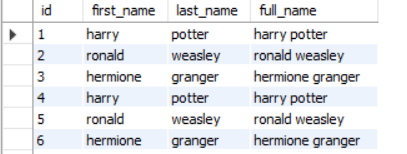
--- code is executed but no output shown

-- 4. write a sql query to combine first and last names of customers as a full\_name.

select id, first\_name,last\_name,

concat(first\_name, ' ', last\_name) as full\_name

from bank\_hex\_feb\_24.customer;



-- 5. write a sql query to remove accounts with a balance of zero where the account type is savings.

delete from bank\_hex\_feb\_24.account

where balance = 0 and account\_type = 'savings';

-- 6. write a sql query to find customers living in a specific city.

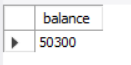
select id, first\_name, last\_name

from bank\_hex\_feb\_24.customer

where city = 'yourcityname';

-- 7. write a sql query to get the account balance for a specific account.

select balance from bank\_hex\_feb\_24.account where id = 1;



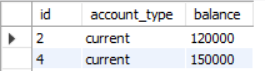
-- 8. write a sql query to list all current accounts with a balance greater than $1,000.

select id, account\_type, balance

from bank\_hex\_feb\_24.account

where account\_type = 'current'

and balance > 1000;



-- 9. write a sql query to retrieve all transactions for a specific account.

select id, transaction\_type, amount, transaction\_date

from bank\_hex\_feb\_24.transaction

where account\_id = 123;

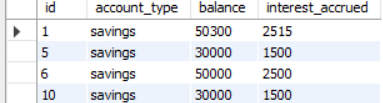
-- 10. write a sql query to calculate the interest accrued on savings accounts based on a given interest rate.

select id, account\_type, balance,

balance \* 0.05 as interest\_accrued

from bank\_hex\_feb\_24.account

where account\_type = 'savings';

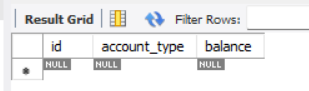


-- 11. write a sql query to identify accounts where the balance is less than a specified overdraft limit.

select id, account\_type, balance

from bank\_hex\_feb\_24.account

where balance < -500;



-- 12. write a sql query to find customers not living in a specific city.

select id, first\_name, last\_name

from bank\_hex\_feb\_24.customer

where city != 'yourcityname'

or city is null;

**task 3**

--- 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;

o/p:

+-------------+--------------+

| customer\_id | avg(balance) |

+-------------+--------------+

| 1 | 100000 |

| 2 | 120000 |

| 3 | 65000 |

+-------------+--------------+

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. also display name of the customer

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);

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.

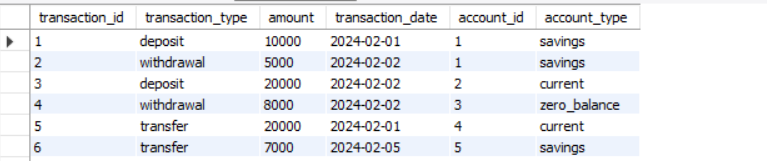
select

t.id as transaction\_id, t.transaction\_type, t.amount,

t.transaction\_date, t.account\_id, a.account\_type

from bank\_hex\_feb\_24.transaction t

join bank\_hex\_feb\_24.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.id as customer\_id , c.first\_name, c.last\_name, c.dob,

a.id as account\_id, a.account\_type, a.balance

from bank\_hex\_feb\_24.customer c

join bank\_hex\_feb\_24.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.id as transaction\_id, t.transaction\_type, t.amount, t.transaction\_date,

a.id as account\_id, a.account\_type, a.balance,

c.id as customer\_id, c.first\_name, c.last\_name, c.dob

from bank\_hex\_feb\_24.transaction t

join bank\_hex\_feb\_24.account a on t.account\_id = a.id

join bank\_hex\_feb\_24.customer c on a.customer\_id = c.id

where a.id = <your\_account\_id>;

8. write a sql query to identify customers who have more than one account.

select c.first\_name,count(c.id) as number\_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 a.customer\_id

having number\_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(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 a.id as account\_id, a.account\_type,

avg(daily\_balance) as avg\_daily\_balance

from (

select a.id, a.account\_type,

date(t.transaction\_date) as transaction\_date,

sum(case

when t.transaction\_type = 'credit' then t.amount

else -t.amount

end) as daily\_balance

from bank\_hex\_feb\_24.account a

join bank\_hex\_feb\_24.transaction t on a.id = t.account\_id

where t.transaction\_date between 'start\_date' and 'end\_date'

group by a.id, a.account\_type,

date(t.transaction\_date)

) as daily\_balances

group by

account\_id,

account\_type;

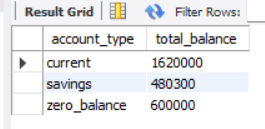
11. calculate the total balance for each account type.

select account\_type,

sum(balance) as total\_balance

from bank\_hex\_feb\_24.account

group by account\_type;



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

select a.id as account\_id,

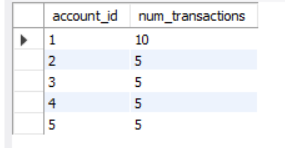
count(t.id) as num\_transactions

from bank\_hex\_feb\_24.account a

join bank\_hex\_feb\_24.transaction t on a.id = t.account\_id

group by a.id

order by num\_transactions desc;



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

select c.id as customer\_id, c.first\_name, c.last\_name, a.account\_type,

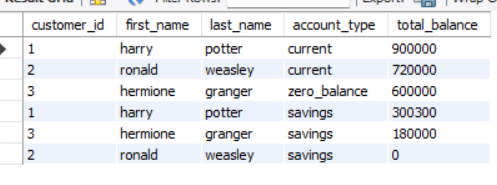
sum(a.balance) as total\_balance

from bank\_hex\_feb\_24.customer c

join bank\_hex\_feb\_24.account a on c.id = a.customer\_id

group by c.id, a.account\_type

order by total\_balance desc;



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

select t.amount, t.transaction\_date, t.account\_id,

count(\*) as num\_duplicates

from bank\_hex\_feb\_24.transaction t

group by t.amount, .transaction\_date, t.account\_id

having count(\*) > 1;