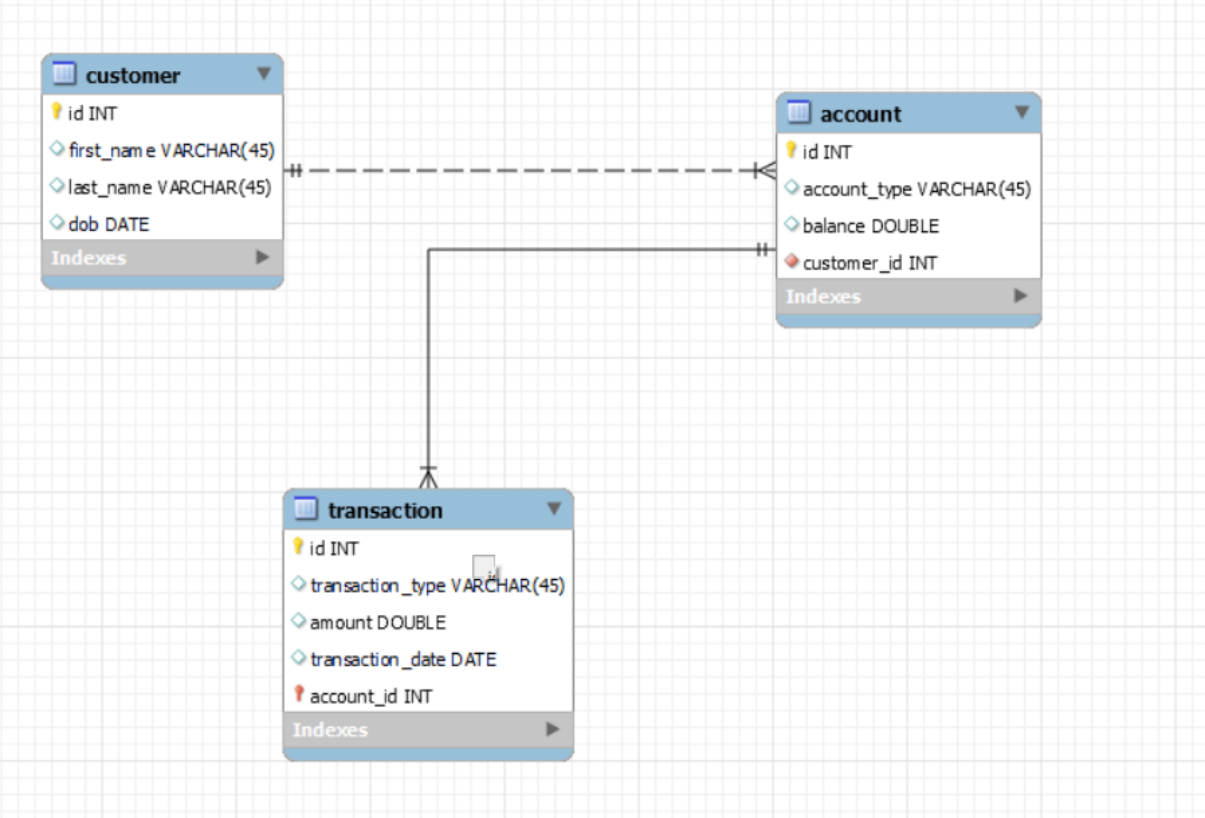
ASSIGNMENT 3 – BANKING ASSIGNMENT




```
-- 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;
```

Result Grid | Filter Rows:

first_name	account_type
harry	savings
ronald	savings
hermione	savings
harry	savings
ronald	savings
hermione	savings

-- 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;

first_name	last_name	transaction_type	transaction_date
harry	potter	deposit	2024-02-01
harry	potter	withdrawal	2024-02-02
ronald	weasley	deposit	2024-02-02
hermione	granger	withdrawal	2024-02-02

-- 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;

id	first_name	last_name	full_name
1	harry	potter	harry potter
2	ronald	weasley	ronald weasley
3	hermione	granger	hermione granger
4	harry	potter	harry potter
5	ronald	weasley	ronald weasley
6	hermione	granger	hermione granger

-- 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;
```

	balance
▶	50300

-- 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;
```

	id	account_type	balance
▶	2	current	120000
	4	current	150000

-- 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';
```

id	account_type	balance	interest_accrued
1	savings	50300	2515
5	savings	30000	1500
6	savings	50000	2500
10	savings	30000	1500

-- 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;
```

Result Grid | Filter Rows:

id	account_type	balance
NULL	NULL	NULL

-- 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;
```

transaction_id	transaction_type	amount	transaction_date	account_id	account_type
1	deposit	10000	2024-02-01	1	savings
2	withdrawal	5000	2024-02-02	1	savings
3	deposit	20000	2024-02-02	2	current
4	withdrawal	8000	2024-02-02	3	zero_balance
5	transfer	20000	2024-02-01	4	current
6	transfer	7000	2024-02-05	5	savings

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;
```

customer_id	first_name	last_name	dob	account_id	account_type	balance
1	harry	potter	2002-03-21	1	savings	50300

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;
```

Result Grid | Filter Rows:

account_type	total_balance
current	1620000
savings	480300
zero_balance	600000

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;
```

account_id	num_transactions
1	10
2	5
3	5
4	5
5	5

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;
```

customer_id	first_name	last_name	account_type	total_balance
1	harry	potter	current	900000
2	ronald	weasley	current	720000
3	hermione	granger	zero_balance	600000
1	harry	potter	savings	300300
3	hermione	granger	savings	180000
2	ronald	weasley	savings	0

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;
```