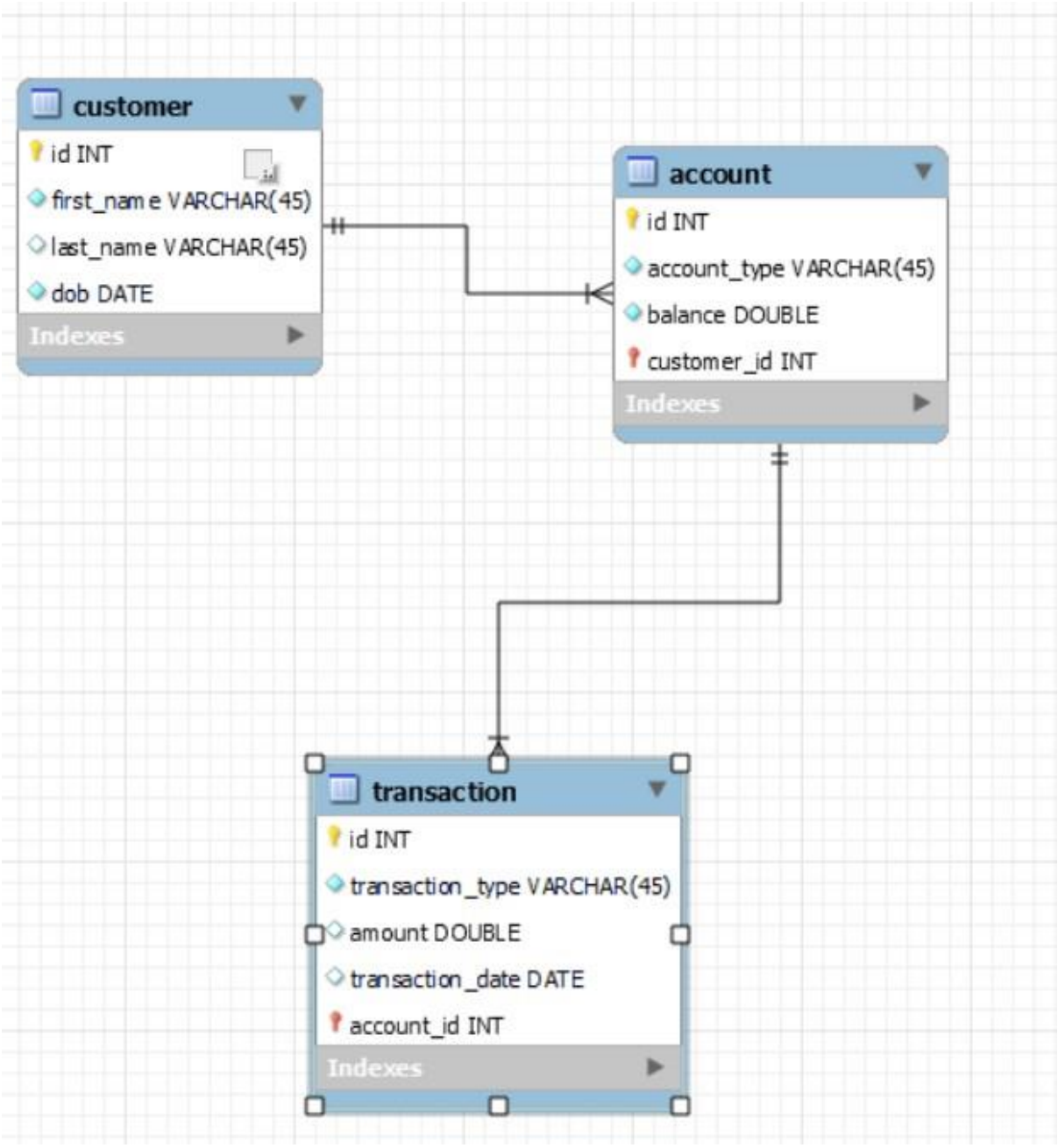
BANKING

ER DIAGRAM:



TASK – 1

```
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) NOT NULL,
  `last_name` VARCHAR(45) NULL,
  `dob` DATE NOT 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) NOT NULL,
  `balance` DOUBLE NOT 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) NOT 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;

show databases;

use banking;

show tables;

desc customer;

desc transaction;

desc account;

insert into customer(first_name,last_name,dob) values('harry','potter','2002-03-02'),

('ronald','weasley','2001-05-10'),

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

insert into customer(first_name,last_name,dob) values

('John', 'Doe', '1999-05-15'),
```

```
('Jane', 'Smith', '2000-10-20'),
('Michael', 'Johnson', '1993-03-28'),
('Emily', 'Brown', '2001-12-10'),
('David', 'Williams', '1998-07-03'),
('Sarah', 'Jones', '1999-09-18'),
('Christopher', 'Martinez', '2002-02-22');
select * from customer;
insert into account(account_type,balance,customer_id) values('savings',50025,1),
('current',40000,2),
('zero_balance',100000,3),
('current',15000,1),
('savings',30000,3);
insert into account(account_type,balance,customer_id) values ('savings',30000,3);
insert into account(account_type,balance,customer_id) values('zero_balance',29000,5),
('current',10000,4),
('current',17000,6),
('zero_balance',25000,8),
('savings',30000,10),
('savings',14000,9);
select * from account;
insert into transaction(transaction_type,amount,transaction_date,account_id) values
('deposit',10000,'2024-02-02',1),
('withdrawl',5000,'2024-02-02',1),
('deposit',20000,'2024-04-02',2),
('withdrawl',7000,'2024-05-21',3),
('transfer',20000,'2024-07-21',4),
('transfer',50000,'2024-12-21',5);
insert into transaction(transaction_type,amount,transaction_date,account_id) values
('deposit',7000,'2024-03-02',6),
```

```
('withdrawl',10000,'2024-02-22',7),
('transfer',2000,'2024-05-13',8),
('transfer',1000,'2024-04-03',9),
('withdrawl',12000,'2024-03-01',10),
('deposit',20000,'2024-02-07',12);
delete from transaction where account_id=5;
select * from transaction;
drop database banking;
```

TASK - 2

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

```
select c.first_name,a.account_type
from customer c,account a
where c.id=a.customer_id;
```

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

```
select c.first_name,t.transaction_type,t.amount,t.transaction_date
from transaction t,customer c,account a
where c.id=a.customer_id and a.id=t.account_id;
```

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

```
-- use 'UPDATE' to make changes in alreay existing table
update account set balance=balance+5000
where id=2;
select * from account;
```

#4.Write a SQL query to Combine first and last names of customers as a full_name.//use 'CONCAT' to combine strings

```
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.

```
-- set foreign_key_checks=0;
delete from account where balance=30000 and account_type='savings';
-- set foreign_key_checks=1;
select * from account;
```

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

```
select balance from account where id=2;
```

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

```
select * from account where balance>1000;
```

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

```
select t.transaction_type,amount,transaction_date
from account a,transaction t
where a.id=t.account_id and a.id=1;
```

#9. Write a SQL query to Calculate the interest accrued on savings accounts based on a

```
-- given interest rate.
select id,account_type,balance*9 as interest
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 id,account_type
from account
where balance<30000;
-- since overdraft limit is not given,assumed it as 30000
```

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

-- city is not mentioned in the schema

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;
```

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

```
select customer_id,balance
from account
order by balance desc
limit 0,3;-- since the table contains 4 rows
```

#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,t.transaction_type,t.transaction_date
from customer c,account a,transaction t
where c.id=a.customer_id and a.id=t.account_id and t.transaction_type='deposit' and t.transaction_date='2024-02-02';
```

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

```
(select * from customer
order by dob
limit 0,1)
union
(select * 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.id,t.transaction_type,t.amount,t.transaction_type,a.account_type
from account a,transaction t
where a.id=t.account_id;
```

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

```
select c.id,c.first_name,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 c.id,c.first_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 a.id=3;
```

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

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

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

```
select((select sum(amount) from transaction where transaction_type='deposit')-
```

(select sum(amount) from transaction where transaction_type='withdrawl')) as diff;

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

```
select a.id,avg(a.balance)
from account a join transaction t on a.id=t.account_id
where transaction_date between '2024-02-01' and '2024-05-30'
group by a.id;
```

#11. Calculate the total balance for each account type.

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

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

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

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

```
select c.first_name,a.account_type,a.balance
from account a join customer c on c.id=a.customer_id
where balance>50000;
```

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

```
select id,transaction_type,amount,transaction_date,count(account_id) as duplicate from transaction group by account_id;
```

TASK - 4

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

```
select customer_id,balance from account where balance = (select max(balance) from account);

select * from account;
```

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

```
-- st1:select avg(balance) from account;

-- st2: (select customer_id,count(customer_id) as acc from account group by customer_id having acc>1)

select avg(balance) from

(select customer_id,count(customer_id) as acc from account

group by customer_id having acc>1) as multi_acc join account on multi_acc.customer_id=account.customer_id;
```

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

```
select account_id,amount from transaction where amount>

(select avg(amount) from transaction);
```

#4.Identify customers who have no recorded transactions.

```
select  id,first_name from customer where id in(select distinct customer_id from account where id not in

(select  account_id from transaction));

 select * from customer;

select * from transaction;

select * from account;
```

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

```
select id,sum(balance) from account where id not in (select account_id from transaction);
```

#6.Retrieve transactions for accounts with the lowest balance.

```
select * from transaction where account_id in(select id from account where balance=(select min(balance) from account));
```

```
/*debugg

select id,min(balance) from account;

select id,balance from account where balance=(select min(balance) from account);

select * from transaction;

*/
```

#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 tot_balance from account

group by account_type;
```