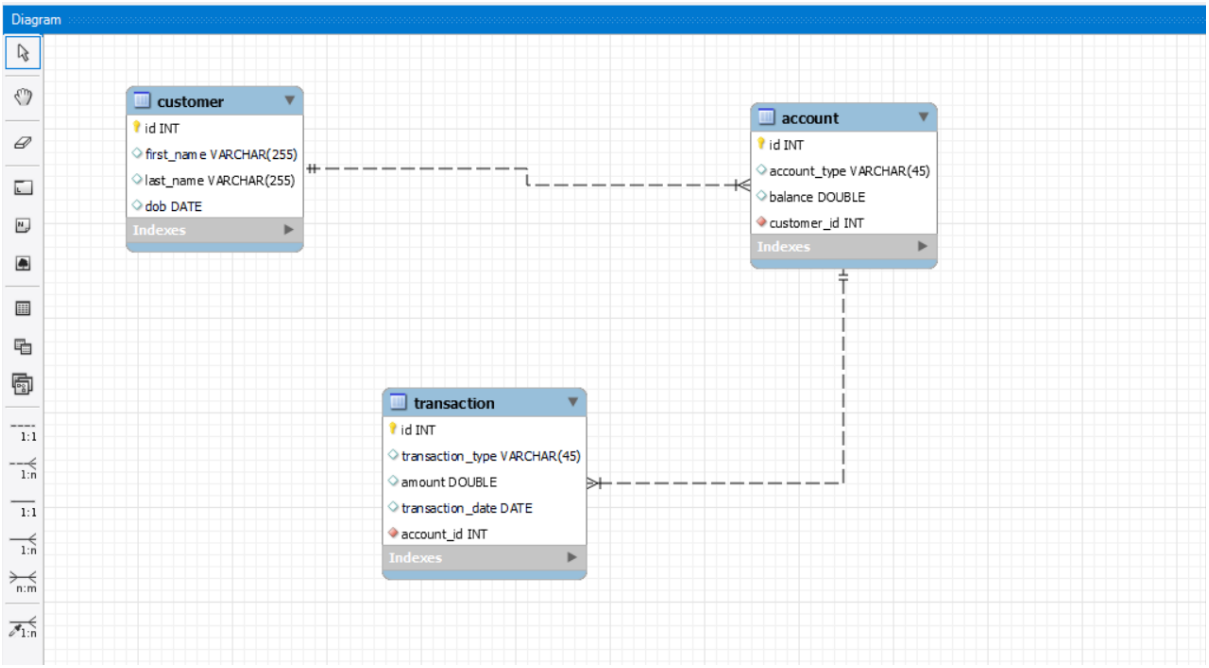
Banking Assignment

ER Diagram



TASK-1:

```
-- MySQL Workbench Forward Engineering


-- -----------------------------------------------------
-- Schema banking_db
-- -----------------------------------------------------


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


-- -----------------------------------------------------
-- Table `banking_db`.`customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `banking_db`.`customer` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(255) NULL,
```

```
  `last_name` VARCHAR(255) NULL,
  `dob` DATE NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


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


-- -----------------------------------------------------
-- Table `banking_db`.`transaction`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `banking_db`.`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`),
```

```
  INDEX `fk_transaction_account1_idx` (`account_id` ASC),
  CONSTRAINT `fk_transaction_account1`
    FOREIGN KEY (`account_id`)
    REFERENCES `banking_db`.`account` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

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,c.email
from customer c join account a on c.id=a.customer_id;
```

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

```
select * from transaction
where account_id=1;
```

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

```
update account
set balance= 90000
where customer_id=3 and 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 Customer_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 account_type='savings' and balance=0;
```

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

```
select * from customer where
city in(“ ”);
```

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

```
select customer_id, sum(balance) as Total_Balance from account
where customer_id=1
group by customer_id;
```

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

```
select id,account_type,balance 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 amount=8000;
```

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

```
select id, account_type, balance * (5 / 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 id, balance
from account
where balance <100000;
```

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

```
select * from customer where
city not in('');
```

#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;
```

#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,sum(t.amount) as Total_Deposits
from transaction t join account a on a.id=t.account_id join customer c on c.id=a.customer_id
where transaction_date='2024-02-01' and transaction_type='deposit'
group by c.id;
```

 # 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.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,c.last_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.*,t.* from
customer c join account a on c.id=a.customer_id join transaction t on a.id=t.account_id
where a.id=2;
```

#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;
```

#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;
```

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

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

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

```
select a.id as account_id, c.first_name, c.last_name,count(t.id) as Number_of_transactions
from account a join transaction t on a.id=t.account_id join customer c on c.id=a.customer_id
group by t.account_id
order by Number_of_transactions desc;
```

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

```
select sum(balance)
from customer c join account a on c.id=a.customer_id;
```

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

```
Select  amount, transaction_date, account_id, count(id) as duplicate_count
from transaction
group by amount, transaction_date, account_id
```

```
having duplicate_count> 1;
```

TASK-4: Subquery and its type:

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

```
select * from account
where balance=(select max(balance) from account);
```

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

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

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

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

#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
```

```
                where id in(select
                        account_id from transaction));
```

#7. Identify customers who have accounts of multiple types.

```
select * from customer where id in (select
                        a.customer_id from account a group by a.customer_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 account_count,
round ((count(id) * 100.0) / (select count(id) 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.

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