**Case Study: Banking System**

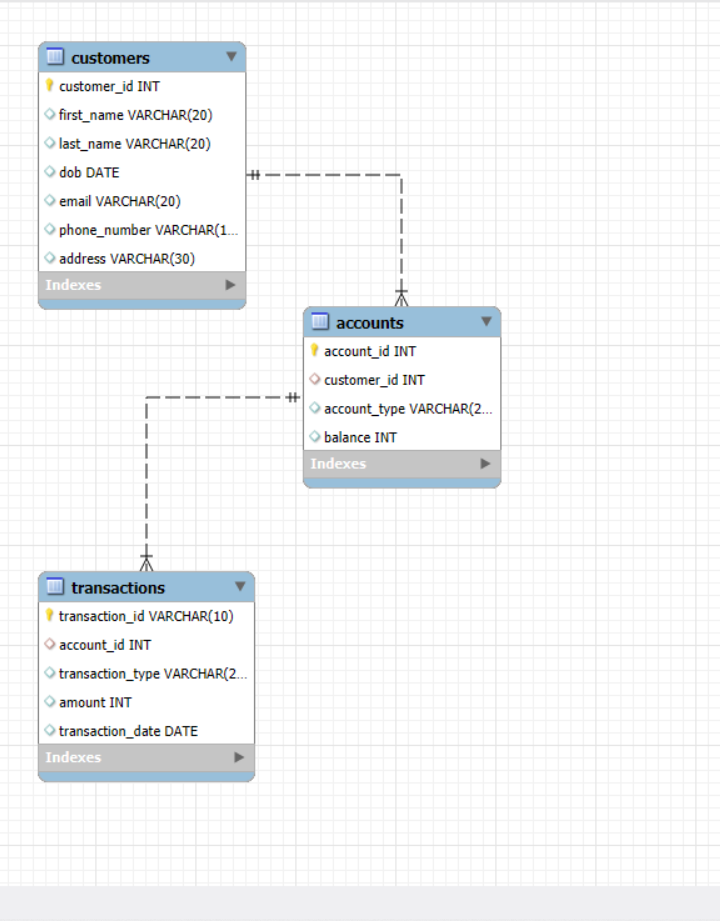
**Task 1: Database design**

1.Create the database named "HMBank"

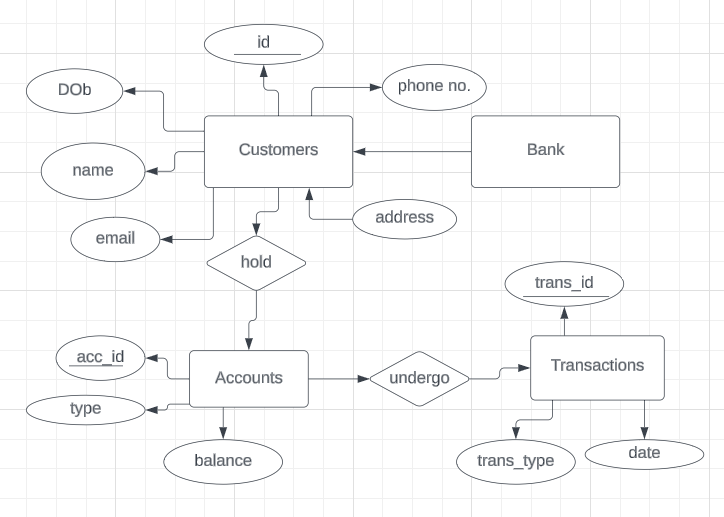
Query: create database HMBank;

2.Define the schema for the Customers, Accounts, and Transactions tables based on the

provided schema.



4.Create an ERD (Entity Relationship Diagram) for the database.



5. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

Customers: Primary key=customer\_id

Accounts: Primary key=account\_id Foreign key= customer\_id

Transactions: Primary key= transaction\_id Foreign key= account\_id

6. Write SQL scripts to create the mentioned tables with appropriate data types, constraints,

and relationships.

• Customers

• Accounts

• Transactions

**Customer:** create table customers(customer\_id integer(5),

first\_name varchar(20),

last\_name varchar(20),

dob date,

email varchar(20),

phone\_number varchar(10),

address varchar(30),

constraint customers\_customerId\_pk primary key(customer\_id));

**Accounts:** create table Accounts(account\_id integer(5),

customer\_id integer(5),

account\_type varchar(20),

balance integer(10),

constraint accounts\_accountId\_pk primary key(account\_id),

constraint customers\_customerId\_fk foreign key(customer\_id) references customers(customer\_id));

Transactions: create table transactions(transaction\_id varchar(10),

account\_id integer(5),

transaction\_type varchar(20),

amount integer(10),

transaction\_date date,

constraint transactions\_transactionId\_pk primary key(transaction\_id),

constraint transactions\_transactionId\_fk foreign key(account\_id) references accounts(account\_id));

**Task 2:** Select, Where, Between, AND, LIKE:

1. Insert at least 10 sample records into each of the following tables.

• Customers

• Accounts

• Transactions

**Customers:**

insert into customers values(101, 'Alice', 'Johnson', '1990-05-14', 'alice@gmail.com', 1234567890,'Utah 89059');

insert into customers values(102, 'Jasmine', 'Lopez', '1994-11-02', 'jasmine@gmail.com', 1234567899,'Michigan 45678');

insert into customers values(103,'Jones','Harrison','1990-02-11','jones@gmail.com', 9763987254,'Washington 98706');

insert into customers values(104, 'Bob', 'Smith', '1985-10-22', 'bob@gmail.com', 2345678901,'Tenesse 90367');

insert into customers values(105, 'Charlie', 'Brown', '1992-03-08', 'charlie@gmail.com', 3456789012,'California 83210');

insert into customers values(106, 'Diana', 'Prince', '1988-12-01', 'diana@gmail.com', 4567890123,'Washington 78560');

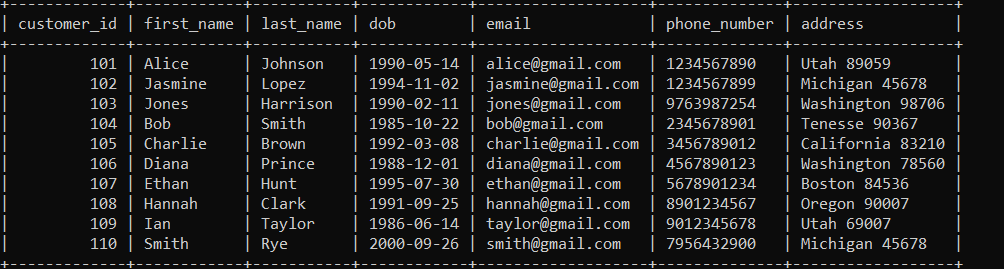
insert into customers values(107, 'Ethan', 'Hunt', '1995-07-30', 'ethan@gmail.com', 5678901234,'Boston 84536');

insert into customers values(108, 'Hannah', 'Clark', '1991-09-25', 'hannah@gmail.com', 8901234567,'Oregon 90007');

insert into customers values(109, 'Ian', 'Taylor', '1986-06-14', 'taylor@gmail.com', 9012345678,'Utah 69007');

insert into customers values(110,'Smith','Rye','2000-09-26’, 'smith@gmail.com',

7956432900,'Michigan 45678');



**Accounts:**

insert into Accounts values(1, 101, 'Savings', 15000);

insert into Accounts values(2, 102, 'Current', 25000);

insert into Accounts values(3, 101, 'zero-balance', 5000);

insert into Accounts values(4, 103, 'Savings', 3000);

insert into Accounts values(5, 104, 'zero-balance', 7000);

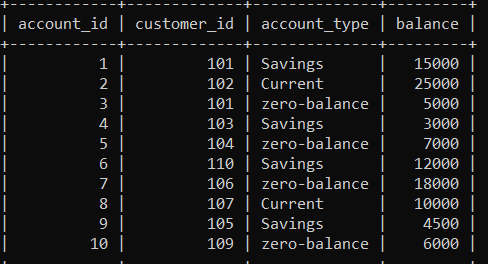
insert into Accounts values(6, 110, 'Savings', 12000);

insert into Accounts values(7, 106, 'zero-balance', 18000);

insert into Accounts values(8, 107, 'Current', 10000);

insert into Accounts values(9, 105, 'Savings', 4500);

insert into Accounts values(10, 109,'zero-balance', 6000);



**Transactions:**

insert into transactions values ('A11', 1, 'Deposit', 1000, '2024-01-15');

insert into transactions values ('B12', 2, 'Withdrawal', 5000, '2024-01-16');

insert into transactions values ('C13', 4, 'Transfer', 2000, '2024-01-17');

insert into transactions values('D14', 3, 'Withdrawal', 1500, '2024-01-18');

insert into transactions values ('E15', 6, 'Deposit', 3000, '2024-01-19');

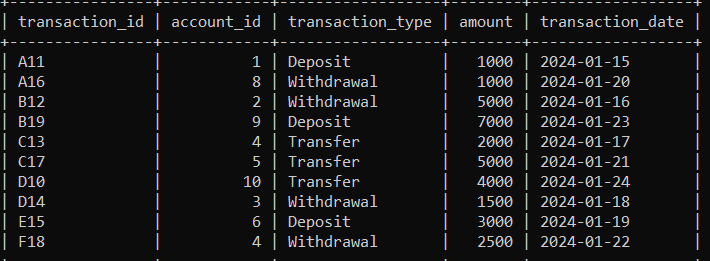
insert into transactions values ('A16', 8, 'Withdrawal', 1000, '2024-01-20');

insert into transactions values ('C17', 5, 'Transfer', 5000, '2024-01-21');

insert into transactions values ('F18', 4, 'Withdrawal', 2500, '2024-01-22');

insert into transactions values ('B19', 9, 'Deposit', 7000, '2024-01-23');

insert into transactions values('D10', 10, 'Transfer', 4000, '2024-01-24');



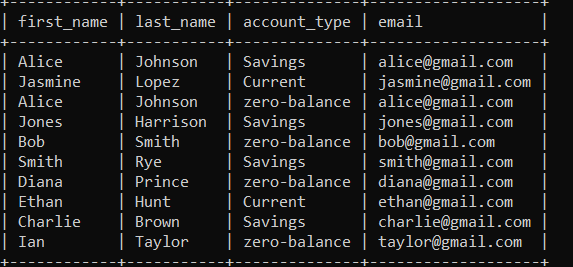
2. Write SQL queries for the following tasks:

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

Query: select c.first\_name, c.last\_name, a.account\_type, c.email

from customers c inner join accounts a

where c.customer\_id=a.customer\_id ;



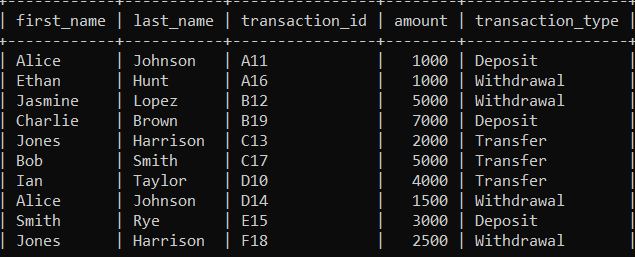
2. Write a SQL query to list all transaction corresponding customer.

Query: select c.first\_name, c.last\_name, t.transaction\_id, t.amount, t.transaction\_type

from transactions t inner join accounts a

on t.account\_id=a.account\_id inner join customers c

on a.customer\_id=c.customer\_id;

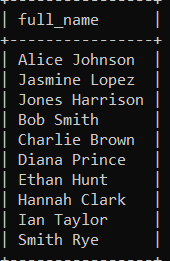


3. Write a SQL query to increase the balance of a specific account by a certain amount.

Query: update accounts set balance=balance+1000 where account\_id=1;

4. Write a SQL query to Combine first and last names of customers as a full\_name.

Query: select concat(first\_name,' ' ,last\_name) as full\_name from customers;

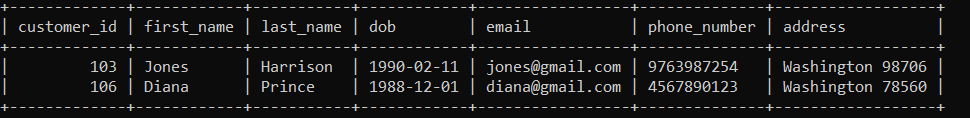


5. Write a SQL query to remove accounts with a balance of zero where the account type is savings.

Query: delete from accounts where balance=0 AND account\_type='Savings';

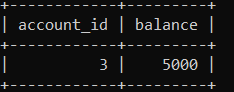
6. Write a SQL query to Find customers living in a specific city.

Query: select \* from customers where address LIKE '%Washington%';



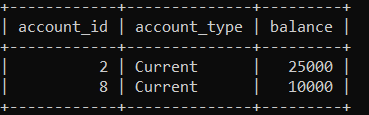
7. Write a SQL query to Get the account balance for a specific account.

Query: select account\_id, balance from accounts where account\_id=3;



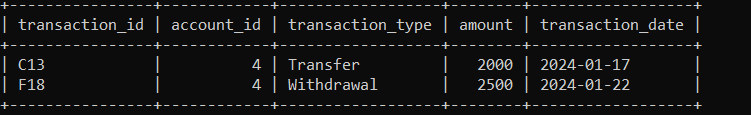
8. Write a SQL query to List all current accounts with a balance greater than $1,000.

Query: select account\_id, account\_type, balance from accounts where balance>1000 AND account\_type='current';



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

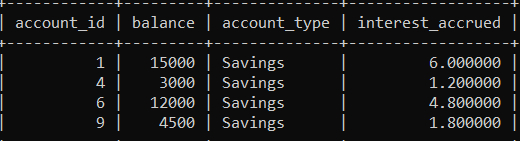
Query: select \* from transactions where account\_id=4;



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

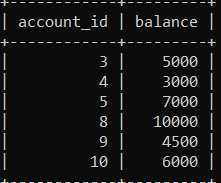
[if interest\_rate=0.04]

Query: select account\_id, balance, account\_type,(balance\*(0.04/100)) as interest\_accrued from accounts where account\_type = ‘Savings’;



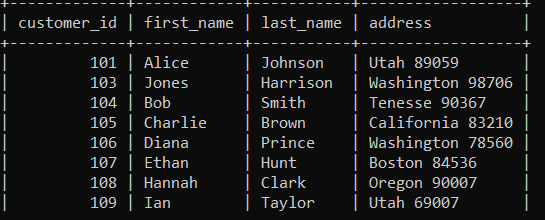
11. Write a SQL query to Identify accounts where the balance is less than a specified overdraft limit.

Query: select account\_id, balance from accounts where balance<12000; [overdraft\_limit=12000]



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

Query: select customer\_id, first\_name, last\_name, address from customers where address NOT LIKE '%Michigan%';



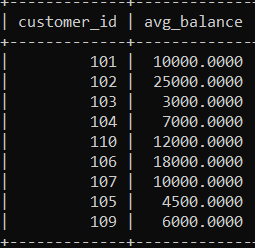
**Tasks 3: Aggregate functions, Having, Order By, Group By and Joins:**

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

Select c.customer\_id,avg(balance) as avg\_balance from customers c

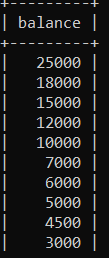
Inner join accounts a where c.customer\_id=a.customer\_id

Group by c.customer\_id;



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

Select balance from accounts order by balance desc;



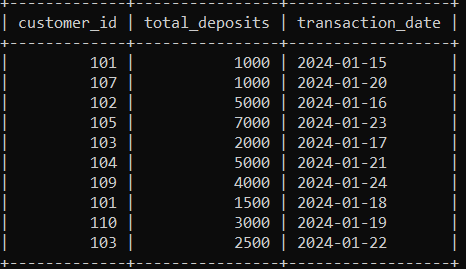
3. Write a SQL query to Calculate Total Deposits for All Customers in specific date.

Select c.customer\_id, sum(t.amount) as total\_deposits , transaction\_date

from transactions t inner join accounts a

On t.account\_id=a.account\_id inner join customers c

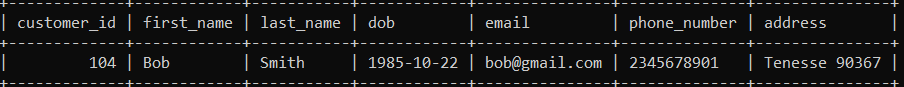
On a.customer\_id=c.customer\_id group by t.transaction\_date, c.customer\_id;



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

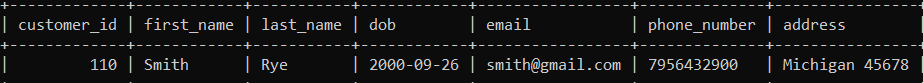
Oldest customer:

select \* from customers where dob = (select MIN(dob) from customers);



Newest customer:

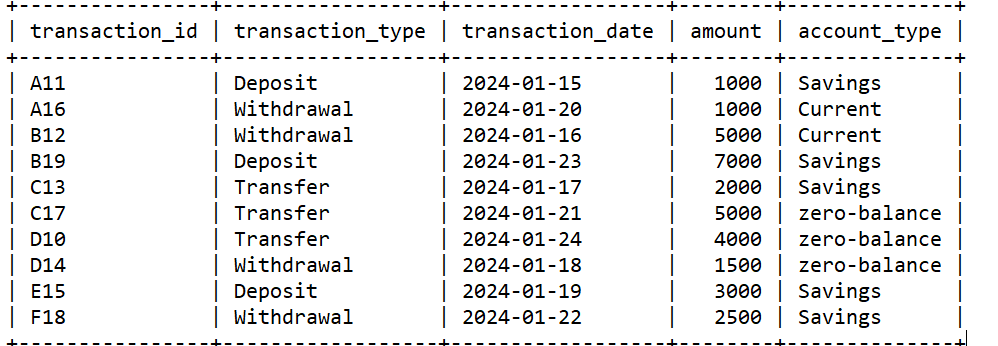
Select\* from customers where dob=(select max(dob) from customers);



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

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

From transactions as t inner join accounts a where t.account\_id=a.account\_id;

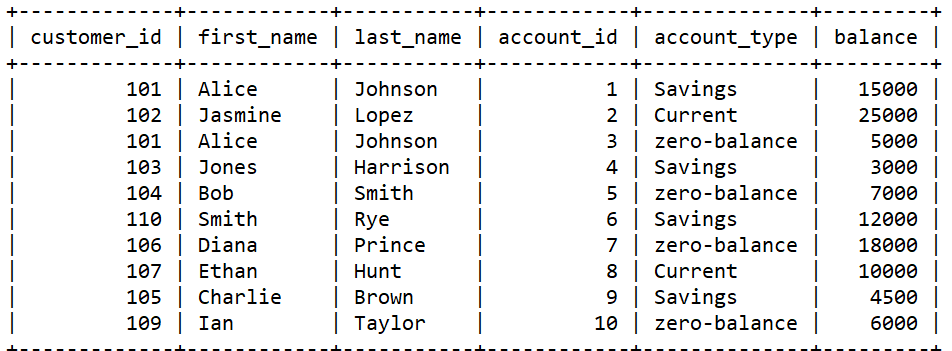


6. Write a SQL query to Get a list of customers along with their account details.

Select c.customer\_id, c.first\_name, c.last\_name, a.account\_id, a.account\_type, a.balance

from customers c inner join accounts a

where c.customer\_id=a.customer\_id;



7. Write a SQL query to Retrieve transaction details along with customer information for a

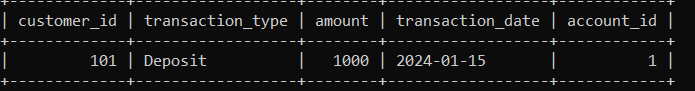
specific account.

Select c.customer\_id, t.transaction\_type, t.amount , t.transaction\_date, t.account\_id

from transactions t inner join accounts a

On t.account\_id=a.account\_id inner join customers c

On a.customer\_id=c.customer\_id where t.account\_id=1;



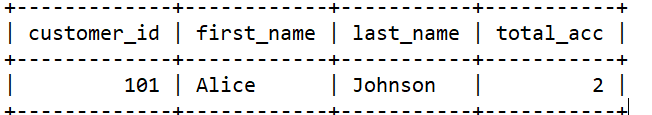
8. Write a SQL query to Identify customers who have more than one account.

Select c.customer\_id, c.first\_name, c.last\_name, count(a.account\_id) as total\_acc

From customers c inner join accounts a

where c.customer\_id=a.customer\_id

group by c.customer\_id, c.first\_name, c.last\_name having count(a.account\_id) >1;



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

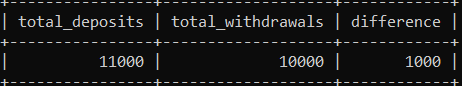
select sum(CASE WHEN transaction\_type = 'Deposit' then amount ELSE 0 END) AS total\_deposits,

sum(CASE WHEN transaction\_type = 'Withdrawal' then amount ELSE 0 END) AS total\_withdrawals,

sum(CASE WHEN transaction\_type = 'Deposit' then amount ELSE 0 END) -

sum(CASE WHEN transaction\_type = 'Withdrawal' then amount ELSE 0 END) as difference

from transactions;



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

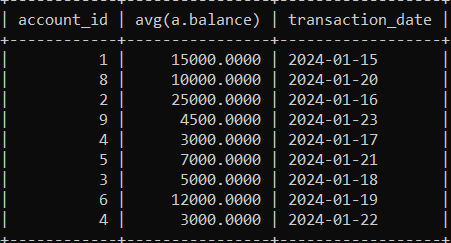
Select a.account\_id, avg(a.balance), t.transaction\_date

from accounts a inner join transactions t

on t.account\_id=a.account\_id

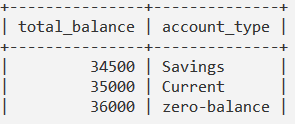
group by a.account\_id, t.transaction\_date

having t.transaction\_date between ‘2024-01-15’ and ‘2024-01-23’;



11. Calculate the total balance for each account type.

Select sum(balance) as total\_balance, account\_type from accounts group by account\_type;

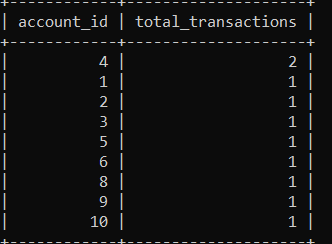


12. Identify accounts with the highest number of transactions order by descending order.

Select account\_id, count(transaction\_id) as total\_transactions

from transactions group by account\_id

Order by count(transaction\_id) desc;



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

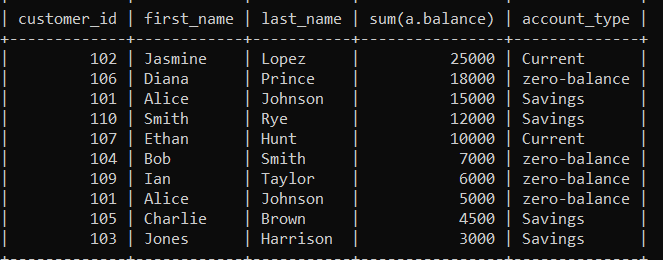
Select c.customer\_id ,sum(balance), a.account\_type

from customers c inner join accounts a

where c.customer\_id=a.customer\_id

group by c.customer\_id , a.account\_type

order by sum(balance) desc;



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

Select amount, transaction\_date, account\_id , count(\*) from transactions

Group by amount, transaction\_date, account\_id

Having count(\*)>1;



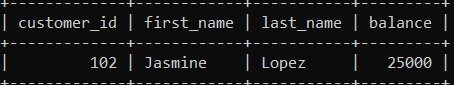
**Tasks 4:**

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

Select c.customer\_id, c.first\_name, c.last\_name, a.balance from customers c inner join

accounts a on c.customer\_id= a.customer\_id

where a.balance=(select max(balance) from accounts);

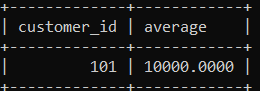


2. Calculate the average account balance for customers who have more than one account.

Select c.customer\_id, avg(a.balance) as average from customers c inner join

accounts a on c.customer\_id= a.customer\_id group by c.customer\_id

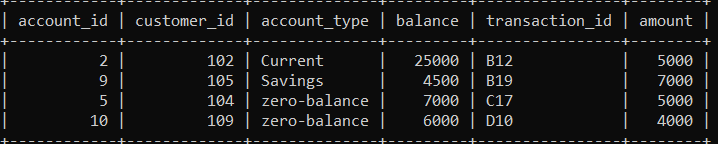
having count(\*)>1;



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

Select a.\*, t.transaction\_id, t.amount from transactions t inner join accounts a

On a.account\_id=t.account\_id where t.amount>(select avg(amount) from transactions);



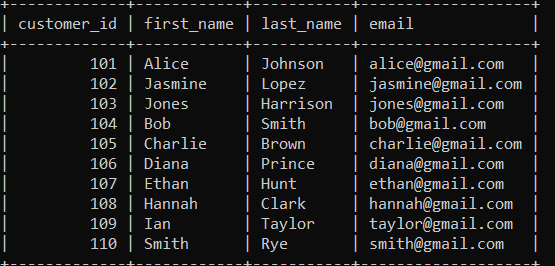
4. Identify customers who have no recorded transactions.

select c.customer\_id, c.first\_name, c.last\_name, c.email

from customers c left join

transactions t ON c.customer\_id = t.account\_id

where t.transaction\_id IS NULL;

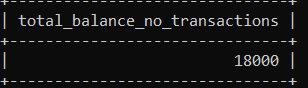


5. Calculate the total balance of accounts with no recorded transactions.

select sum(a.balance) as total\_balance\_no\_transactions

from accounts a left join transactions t on a.account\_id = t.account\_id

where t.transaction\_id IS NULL;

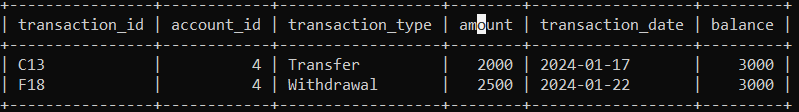


6. Retrieve transactions for accounts with the lowest balance.

Select t.\*, a.balance from transactions t inner join accounts a

On a.account\_id=t.account\_id

Where a.balance=(select min(balance) from accounts);



7. Identify customers who have accounts of multiple types.

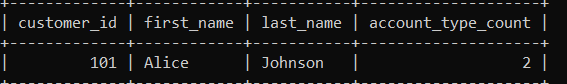
select c.customer\_id, c.first\_name, c.last\_name,

COUNT(DISTINCT a.account\_type) as account\_type\_count

from customers c join accounts a ON c.customer\_id = a.customer\_id

group by c.customer\_id, c.first\_name, c.last\_name

HAVING COUNT(DISTINCT a.account\_type) > 1;



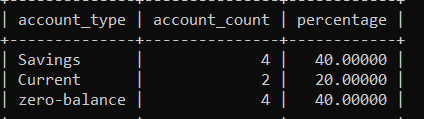
8. Calculate the percentage of each account type out of the total number of accounts.

Select account\_type,

COUNT(\*) as account\_count,

(COUNT(\*) \* 100.0 / (select COUNT(\*) from accounts)) as percentage

From accounts group by account\_type;



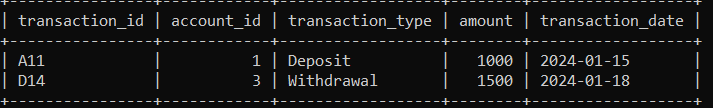
9. Retrieve all transactions for a customer with a given customer\_id.

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

from transactions t

join accounts a ON t.account\_id = a.account\_id

where a.customer\_id = 101;



10. Calculate the total balance for each account type, including a subquery within the SELECT clause

select account\_type, sum(balance) as total\_balance,

(select sum(balance) from accounts where account\_type = account\_type)

from accounts group by account\_type;

