**Assignment-Banking System**

Name : RISHATHA SHAHIN B

Superset ID : 5294964

Email : rishatha2004@gmail.com

Batch : C# Batch 4

Git: https://github.com/rishatha/BankingSystem.git

**Database Tables:**

**--Task 1**

create database HMBank;

use HMBank;

**-- 1. create tables**

create table customers (

customer\_id int primary key,

first\_name varchar(50),

last\_name varchar(50),

dob date,

email varchar(50),

phone\_number varchar(20),

address varchar(200));

create table accounts (

account\_id int primary key,

customer\_id int,

account\_type varchar(20),

balance decimal(15,2),

foreign key (customer\_id) references customers(customer\_id));

create table transactions (

transaction\_id int primary key,

account\_id int,

transaction\_type varchar(20),

amount decimal(15,2),

transaction\_date date,

foreign key (account\_id) references accounts(account\_id));

**--inserting values**

insert into customers values

(1,'arun','kumar','1990-01-15','arun@gmail.com','9000000001','chennai'),

(2,'divya','shree','1985-05-20','divya@gmail.com','9000000002','madurai'),

(3,'rahul','raj','1992-08-10','rahul@gmail.com','9000000003','coimbatore'),

(4,'meena','shah','1988-12-05','meena@gmail.com','9000000004','salem'),

(5,'vijay','patel','1995-03-30','vijay@gmail.com','9000000005','trichy'),

(6,'kavya','sandeep','1997-07-25','kavya@gmailcom','9000000006','chennai'),

(7,'anil','patel','1980-11-11','anil@gmail.com','9000000007','coimbatore'),

(8,'priya','das','1982-02-14','priya@gmail.com','9000000008','madurai'),

(9,'raj','varma','2000-09-09','raj@gmail.com','9000000009','salem'),

(10,'nisha','verma','1998-10-17','nisha@gmail.com','9000000010','trichy');

insert into accounts values

(101,1,'savings',1500.00),

(102,1,'current',500.00),

(103,2,'savings',2000.00),

(104,3,'zero\_balance',0.00),

(105,4,'savings',1200.00),

(106,5,'current',3000.00),

(107,6,'savings',800.00),

(108,7,'zero\_balance',0.00),

(109,8,'current',2500.00),

(110,9,'savings',1800.00);

insert into transactions values

(1001,101,'deposit',500.00,'2025-06-01'),

(1002,101,'withdrawal',200.00,'2025-06-05'),

(1003,102,'deposit',300.00,'2025-06-03'),

(1004,103,'deposit',1000.00,'2025-06-02'),

(1005,104,'deposit',100.00,'2025-06-04'),

(1006,105,'withdrawal',400.00,'2025-06-06'),

(1007,106,'deposit',1500.00,'2025-06-01'),

(1008,107,'deposit',800.00,'2025-06-07'),

(1009,108,'withdrawal',50.00,'2025-06-05'),

(1010,109,'deposit',2000.00,'2025-06-08'),

(1011,110,'deposit',1800.00,'2025-06-09'),

(1012,103,'withdrawal',500.00,'2025-06-10'),

(1013,105,'deposit',400.00,'2025-06-11'),

(1014,106,'withdrawal',600.00,'2025-06-12');

**--Queries**

**--Task 2**

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

select first\_name, account\_type, email

from customers c

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

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

select c.customer\_id, c.first\_name, t.\*

from transactions t

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

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.**

update accounts

set balance = balance + 500

where account\_id = 101;

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

select 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.**

delete from accounts

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

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

select \* from customers

where address like '%madurai%';

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

select balance from accounts

where account\_id=103;

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

select \* from accounts

where account\_type='current' and balance>1000;

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

select \* from transactions

where account\_id=105;

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

select account\_id, balance\*0.05 as interest

from accounts where account\_type='savings';

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

select \* from accounts

where balance < -500;

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

select \* from customers

where address not like '%salem%';

--Task 3

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

select avg(balance) as avg\_balance

from accounts;

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

select top 10 \*

from accounts

order by balance desc;

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

select sum(amount) as total\_deposits

from transactions

where transaction\_type='deposit' and transaction\_date='2025-06-01';

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

select min(dob) as oldest, max(dob) as newest

from customers;

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

select t.\*, a.account\_type

from transactions t

join accounts a on t.account\_id=a.account\_id;

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

select c.\*, a.\*

from customers c

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

**--7.Write a SQL query to Retrieve transaction details along with customer information for a specific account.**

select t.\*, c.first\_name, c.last\_name

from transactions t

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

join customers c on a.customer\_id=c.customer\_id where a.account\_id=101;

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

select customer\_id

from accounts

group by customer\_id

having count(\*)>1;

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

select t.account\_id, sum(case when transaction\_type='deposit'

then amount else -amount end) as net

from transactions t group by t.account\_id;

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

select account\_type, avg(balance) as avg\_balance

from accounts

group by account\_type;

**--11.Calculate the total balance for each account type**

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

from accounts

group by account\_type;

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

select account\_id, count(\*) as txn\_count

from transactions

group by account\_id

order by txn\_count desc;

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

select c.first\_name, a.account\_type, sum(a.balance) as total\_bal

from customers c

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

group by c.first\_name, a.account\_type;

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

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

from transactions t

group by t.transaction\_type, t.amount, t.transaction\_date

having count(\*)>1;

**--Task 4**

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

select \* from accounts

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

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

select avg(balance)

from accounts

where customer\_id in (select customer\_id

from accounts

group by customer\_id

having count(\*)>1);

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

select \* from transactions

where amount > (select avg(amount) from transactions);

**--4. Identify customers who have no recorded transactions.**

select \* from customers

where customer\_id not in (select distinct customer\_id

from accounts a

join transactions t on a.account\_id=t.account\_id);

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

select sum(balance) from accounts

where account\_id not in (select distinct account\_id from transactions);

**--6.Retrieve transactions for accounts with the lowest balance.**

select \* from transactions

where account\_id in (select account\_id from accounts

where balance = (select min(balance) from accounts));

**--7.Identify customers who have accounts of multiple types.**

select customer\_id from accounts

group by customer\_id

having count(distinct account\_type)>1;

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

select account\_type, cast(count(\*)\*100.0/

(select count(\*) from accounts) as decimal(5,2)) as pct

from accounts group by account\_type;

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

select \* from transactions

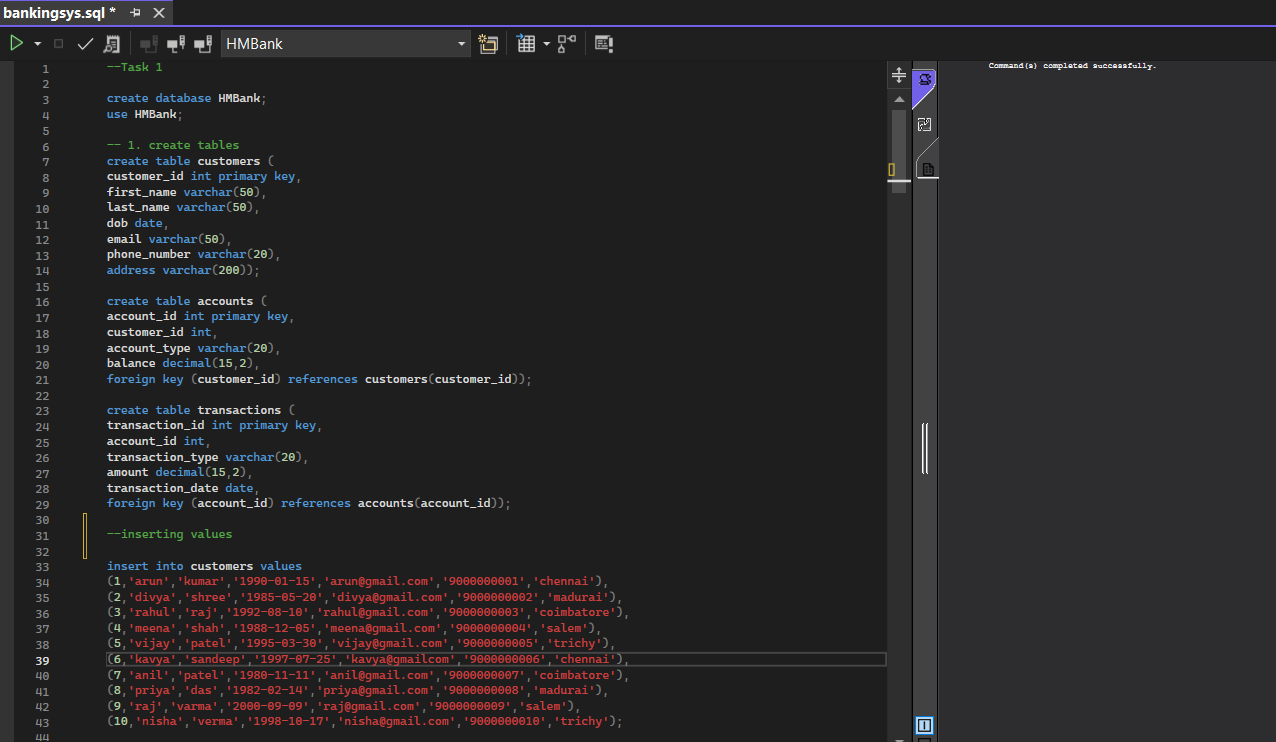
where account\_id=101;

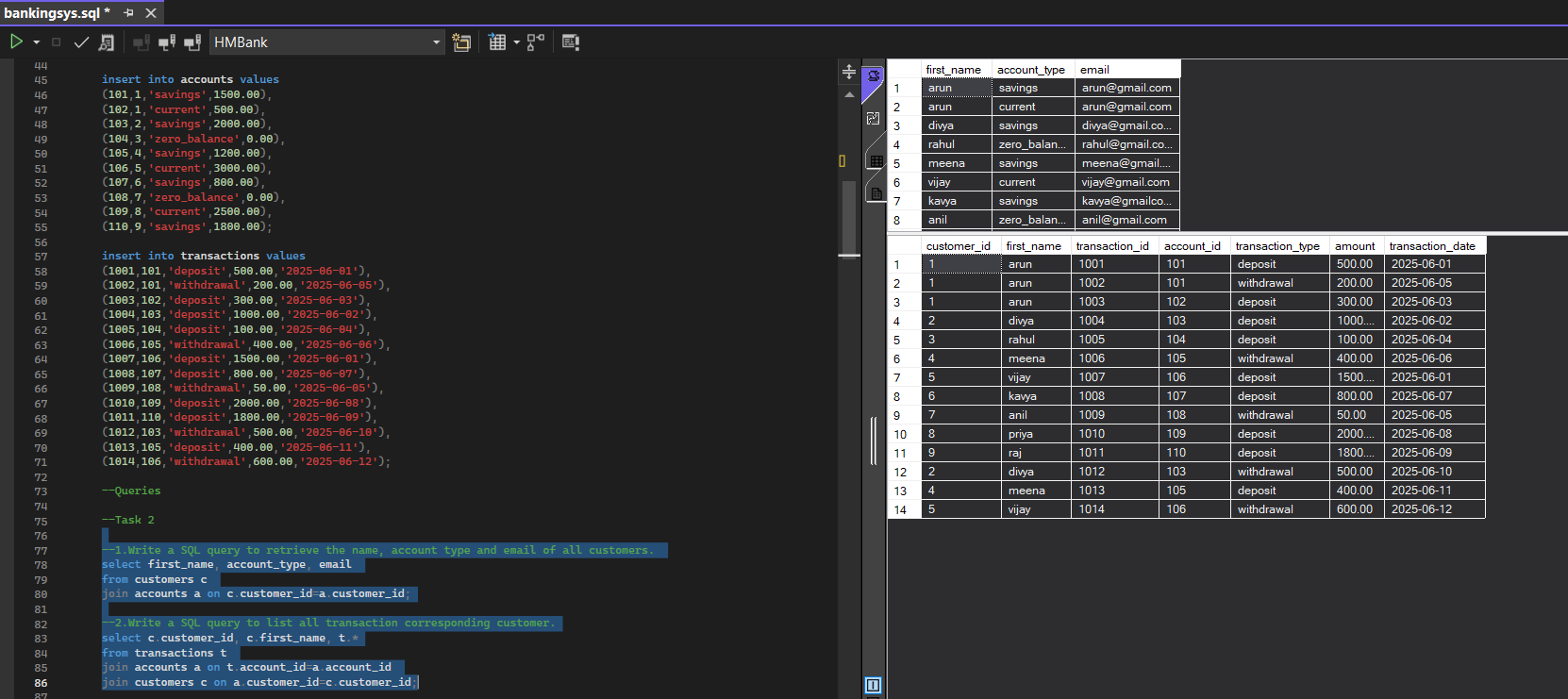
**--10. Calculate the total balance for each account type, including a subquery within the SELECT clause.**

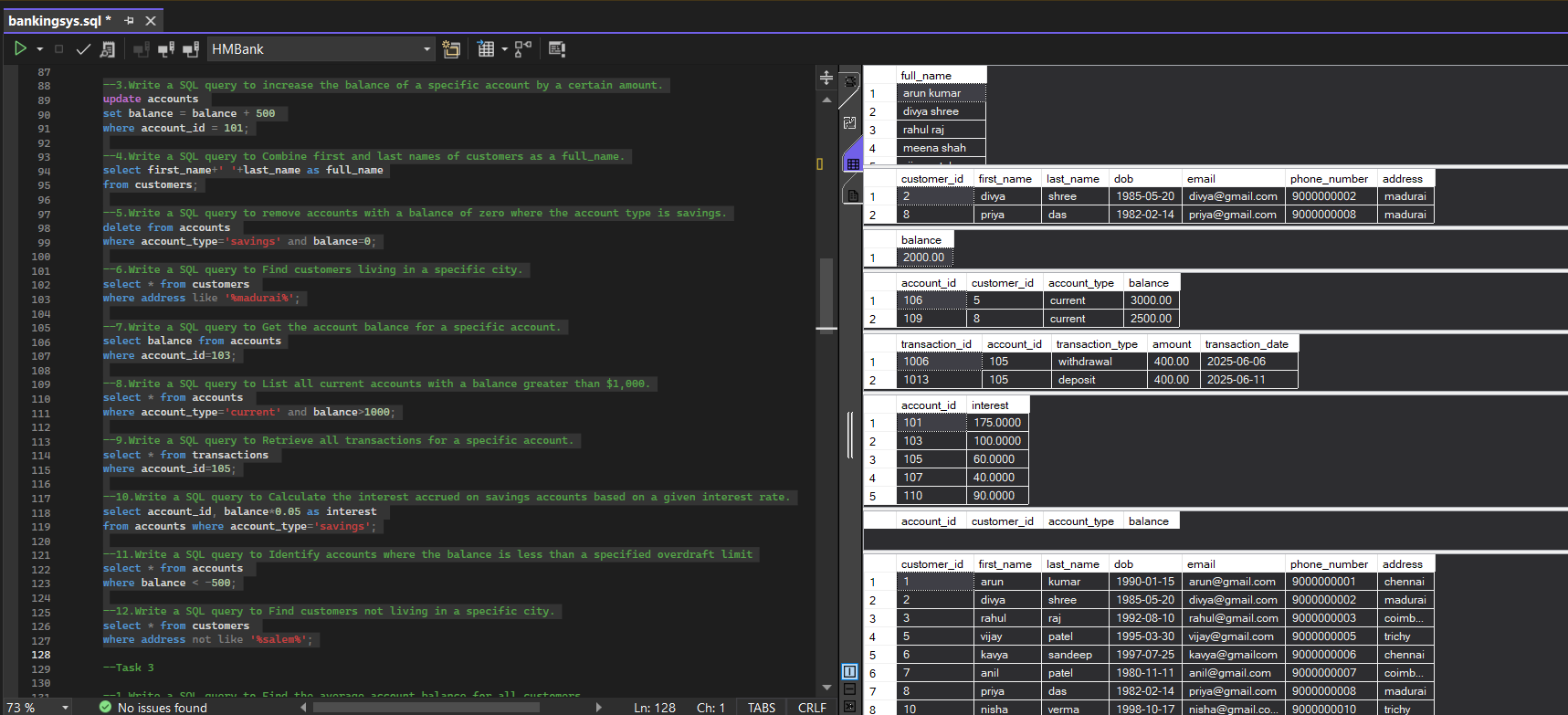
select account\_type, sum(balance) as total\_bal

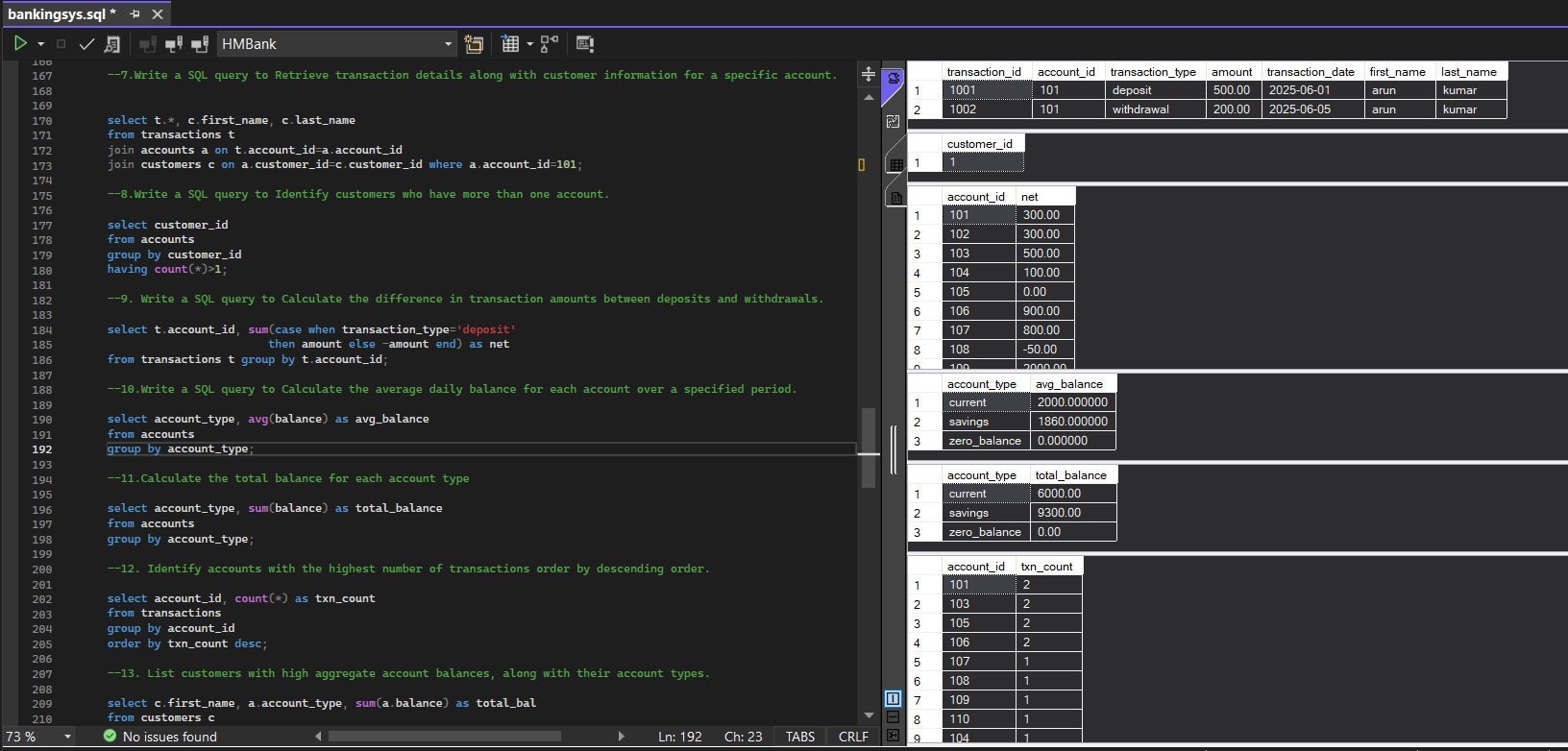
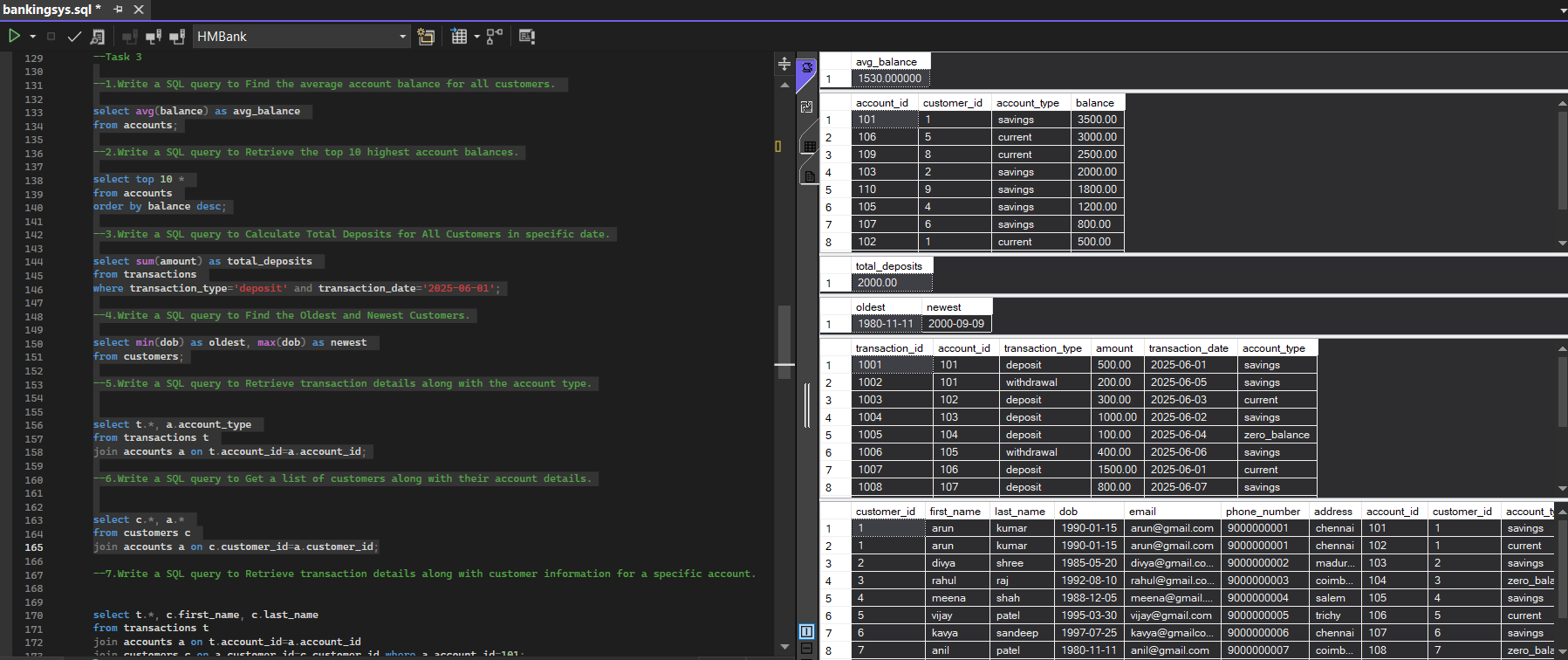
from accounts

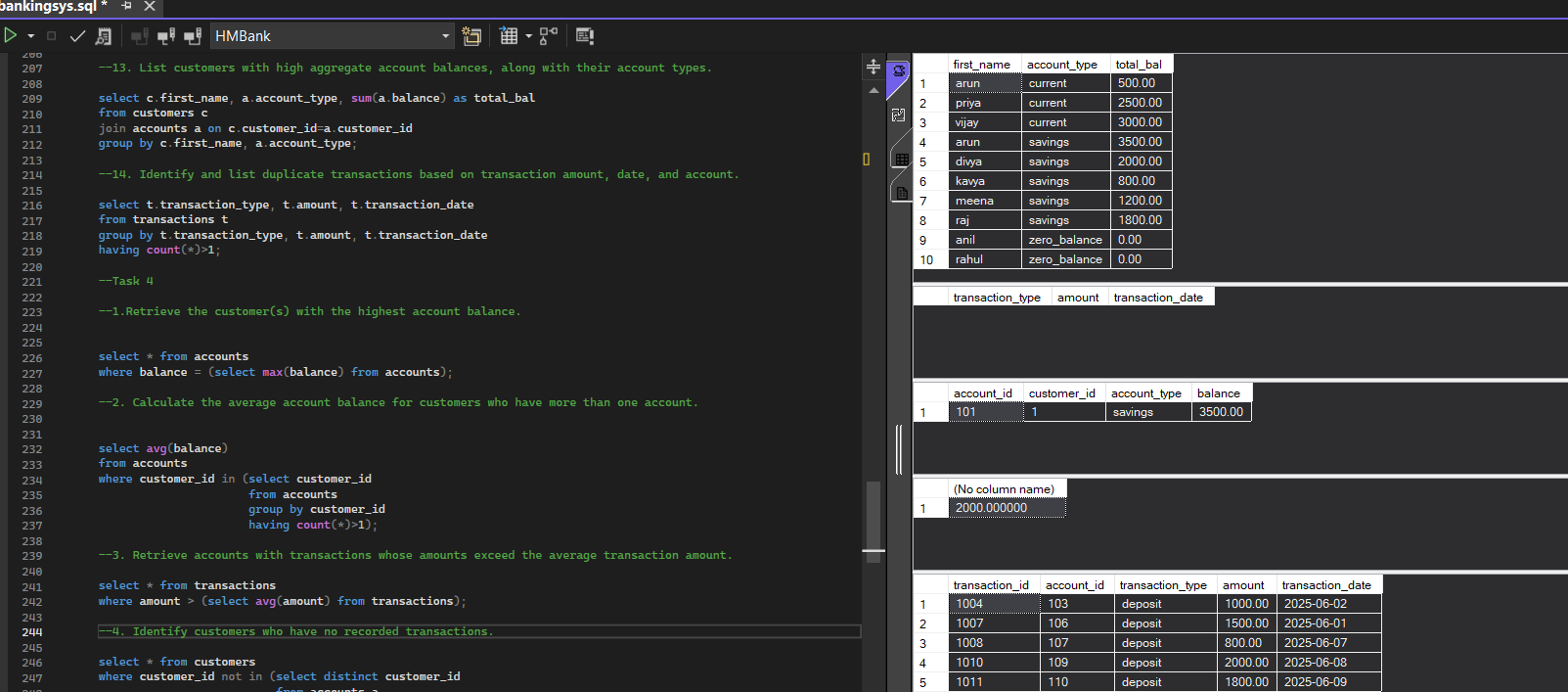
group by account\_type;

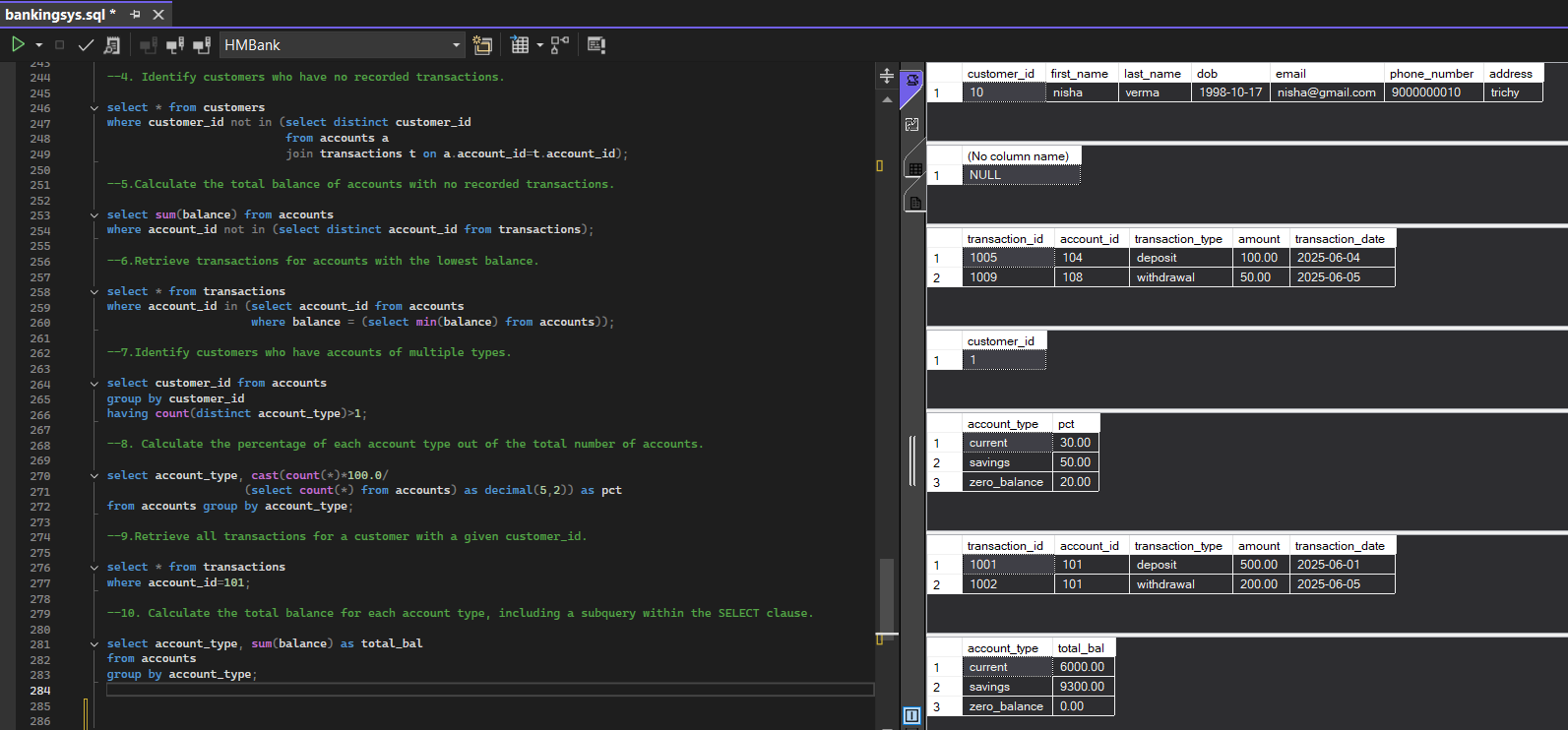












**Banking System**

**Control Structure**

Task 1: Conditional Statements

Task 2: Nested Conditional Statements

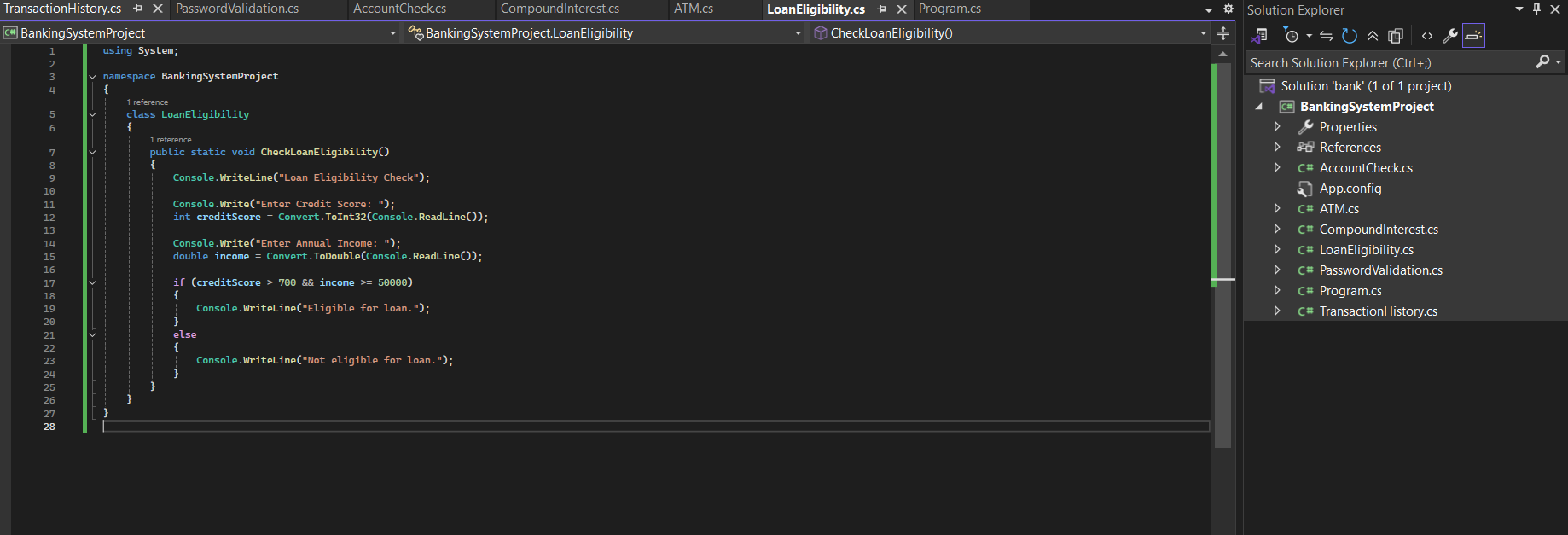
Task 3: Loop Structures

Task 4: Looping, Array and Data Validation

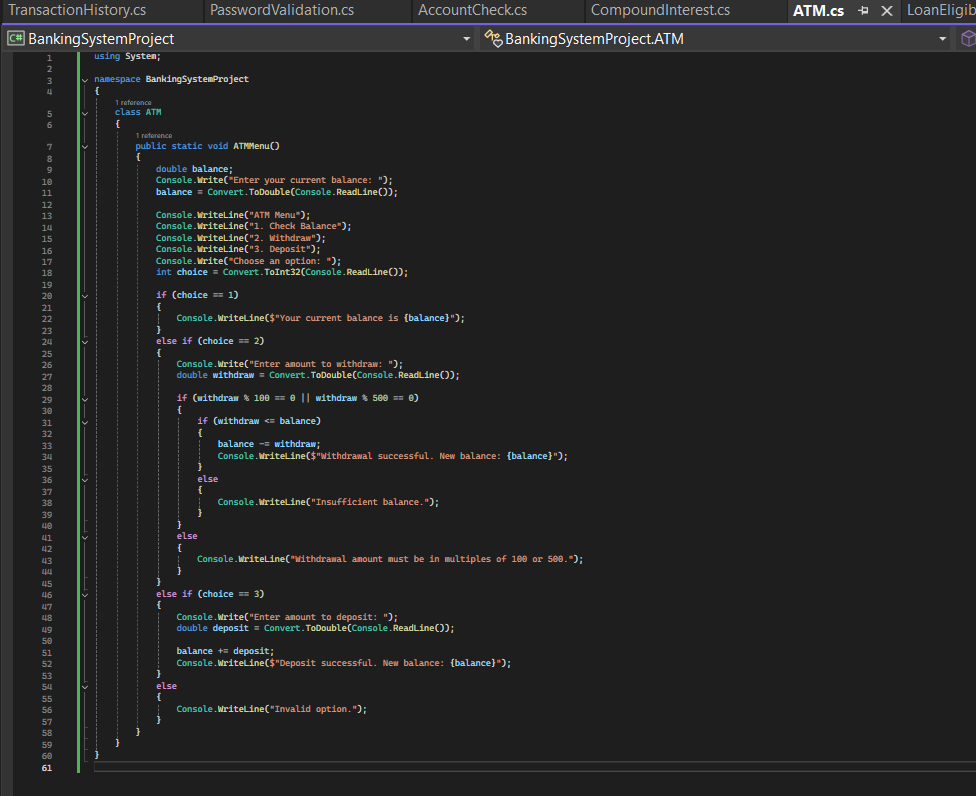
Task 5: Password Validation

Task 6: Password Validation

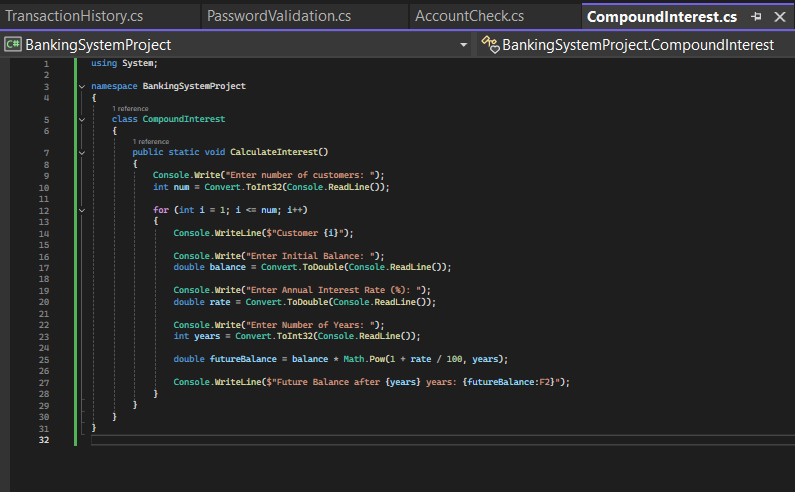
Task 1:



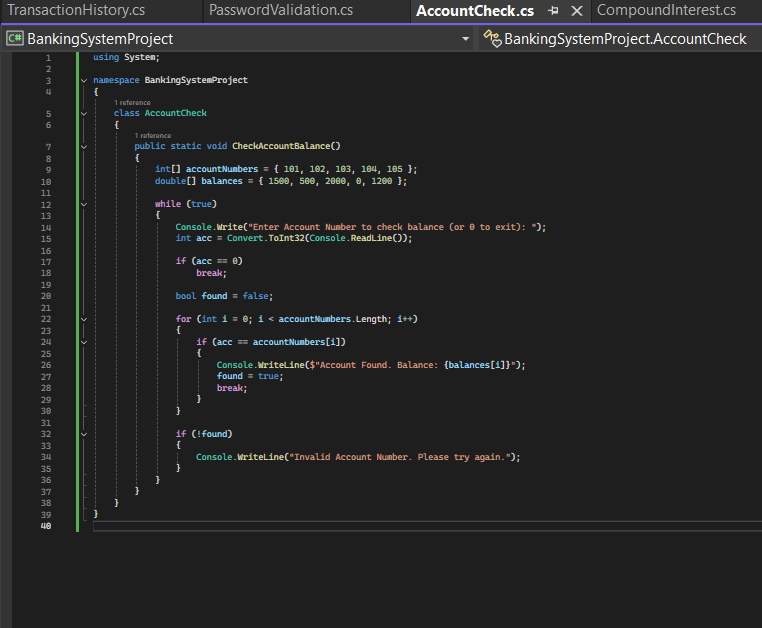
Task 2:



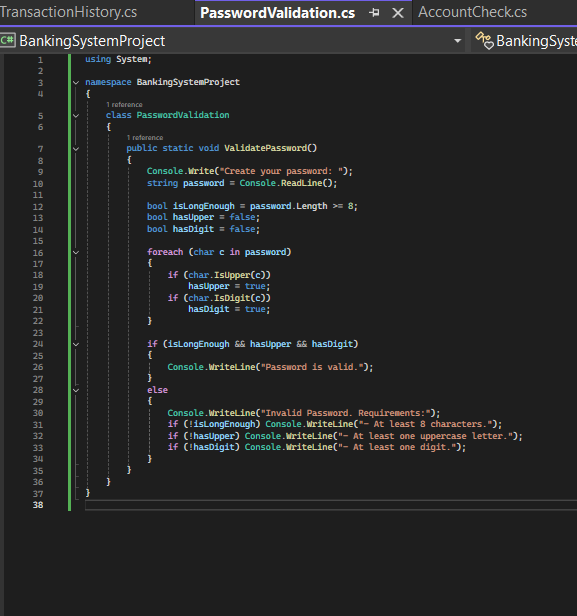
Task 3:



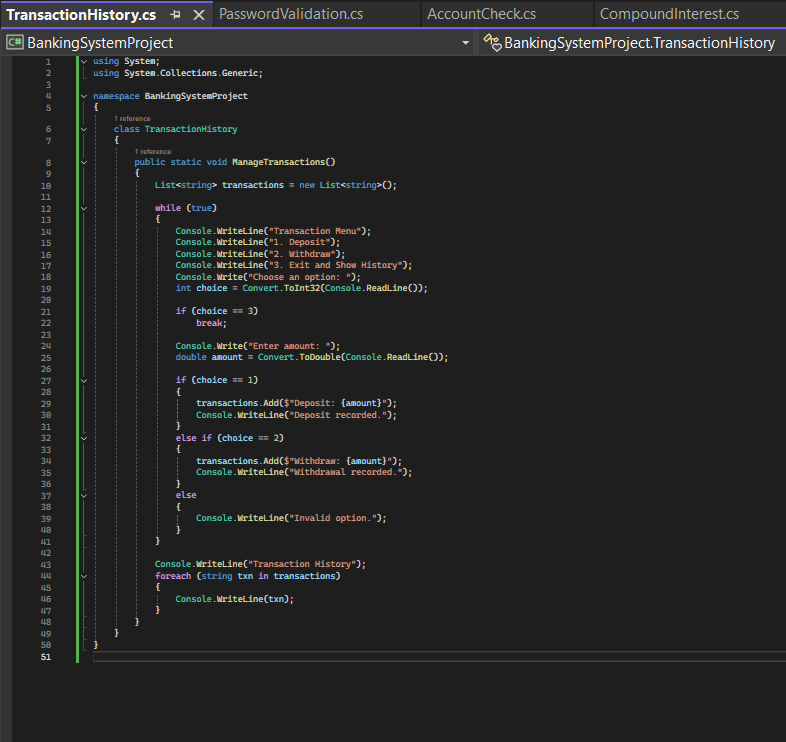
Task 4:



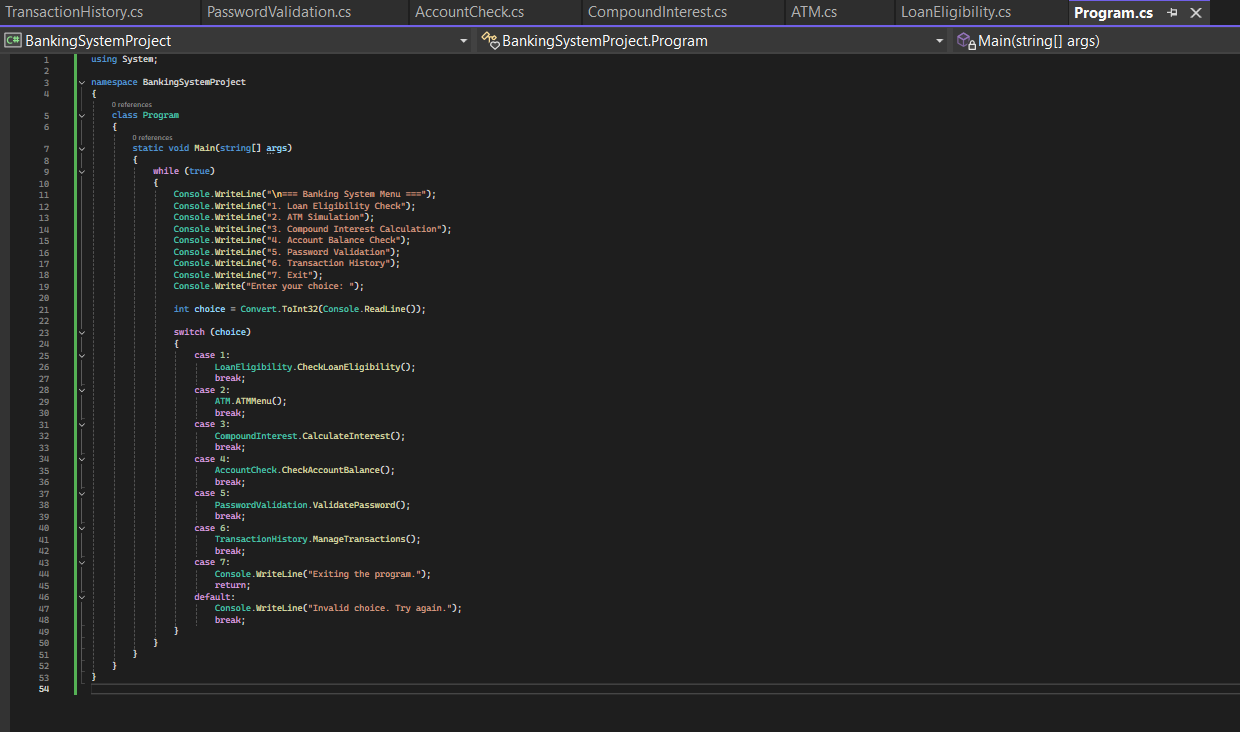
Task 5:



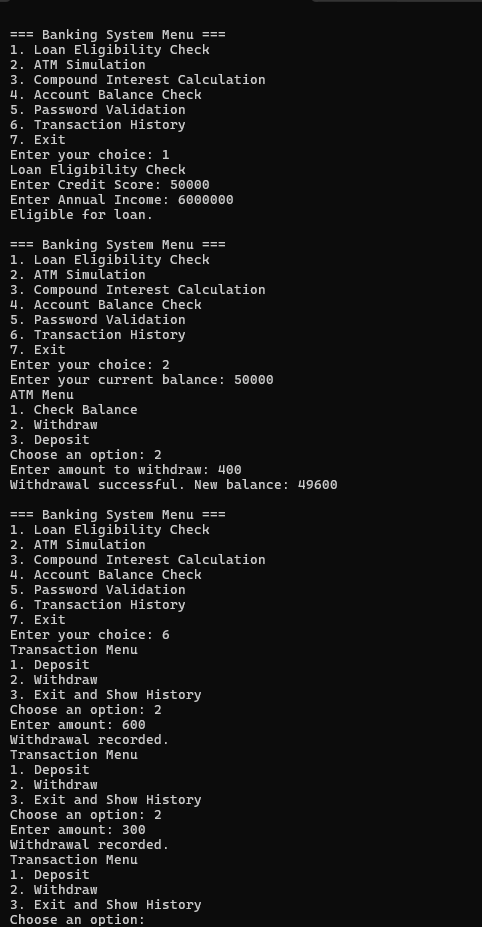
Task 6:



Main Module:



Output:



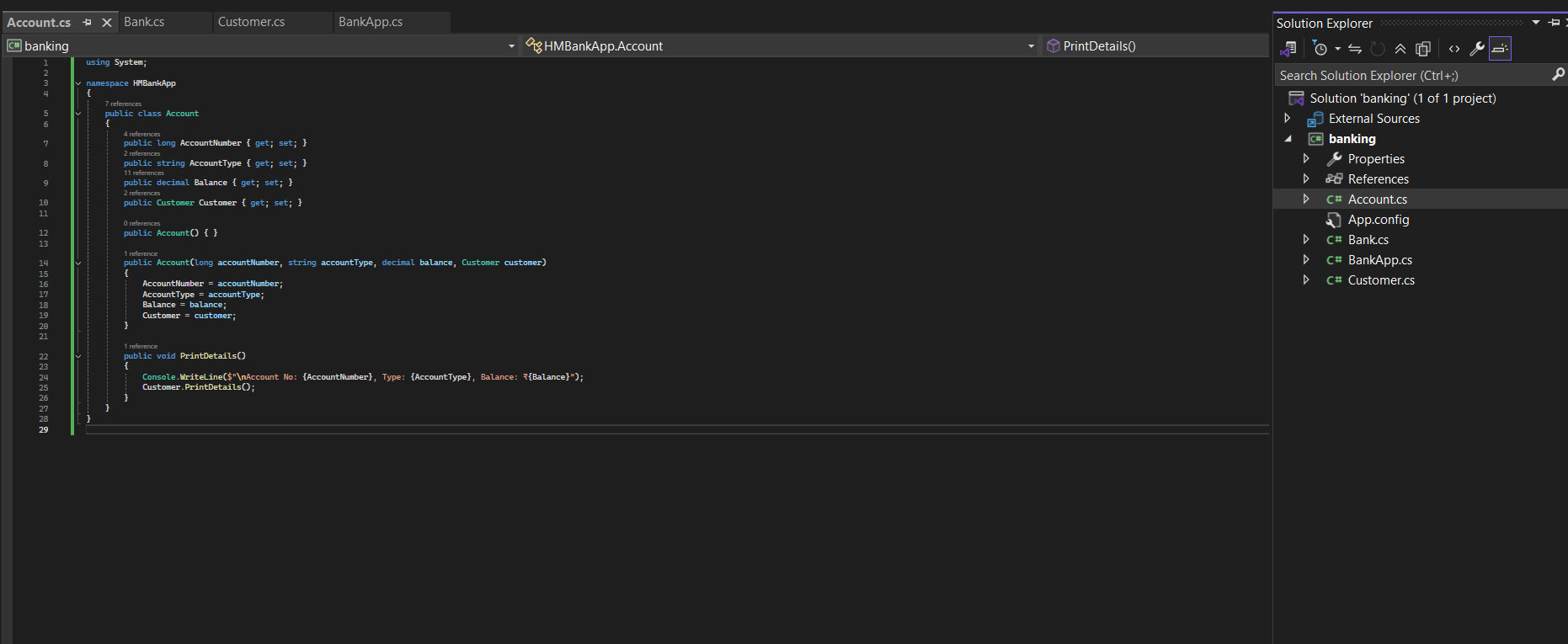
**OOPS, Collections and Exception Handling**

Task 7: Class & Object

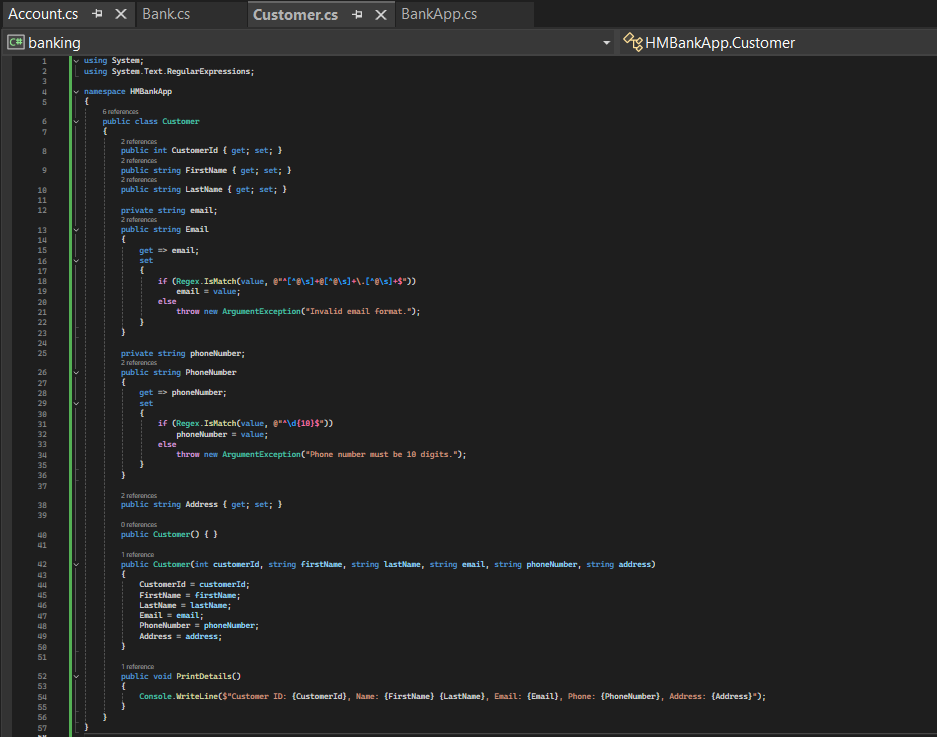
Task 8: Inheritance and polymorphism

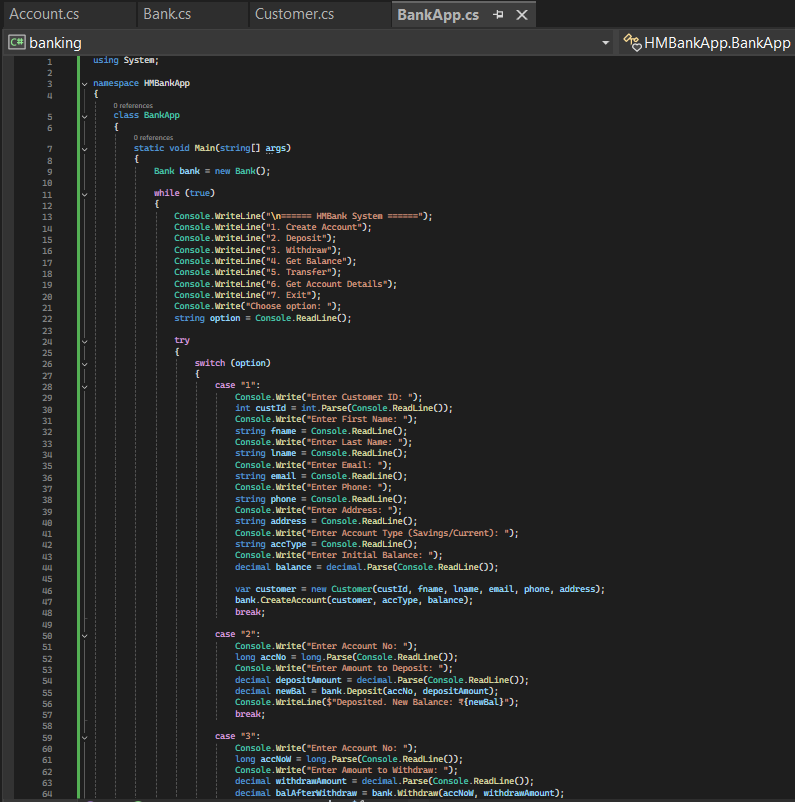
Task 9: Abstraction

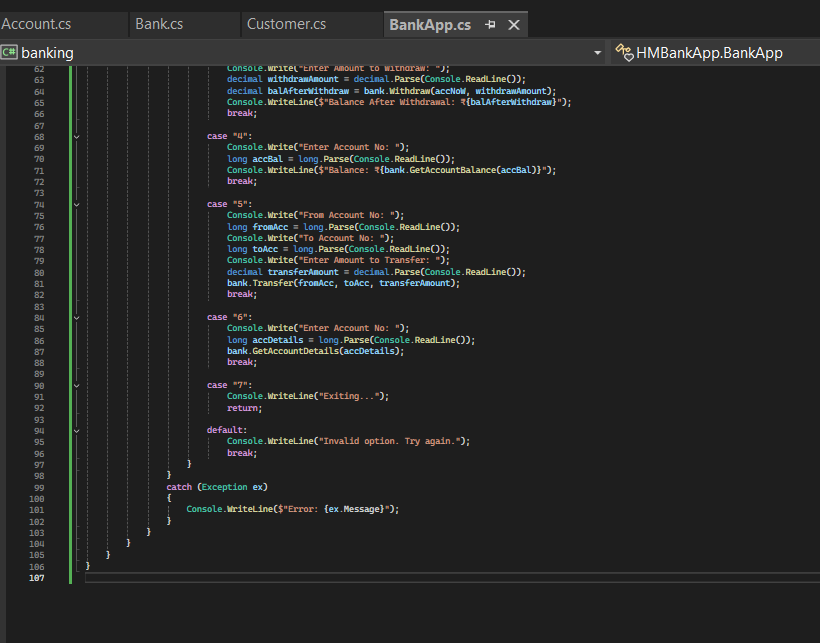
Task 10: Has A Relation / Association



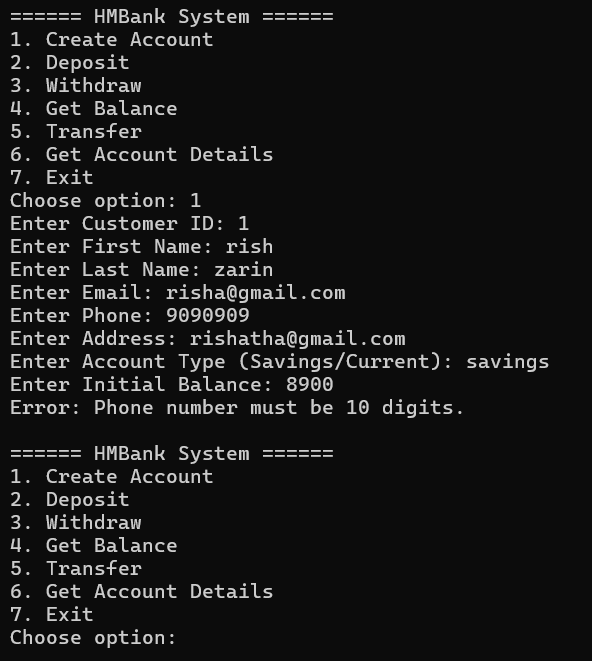








Output:



**OOPS, Collections and Exception Handling**

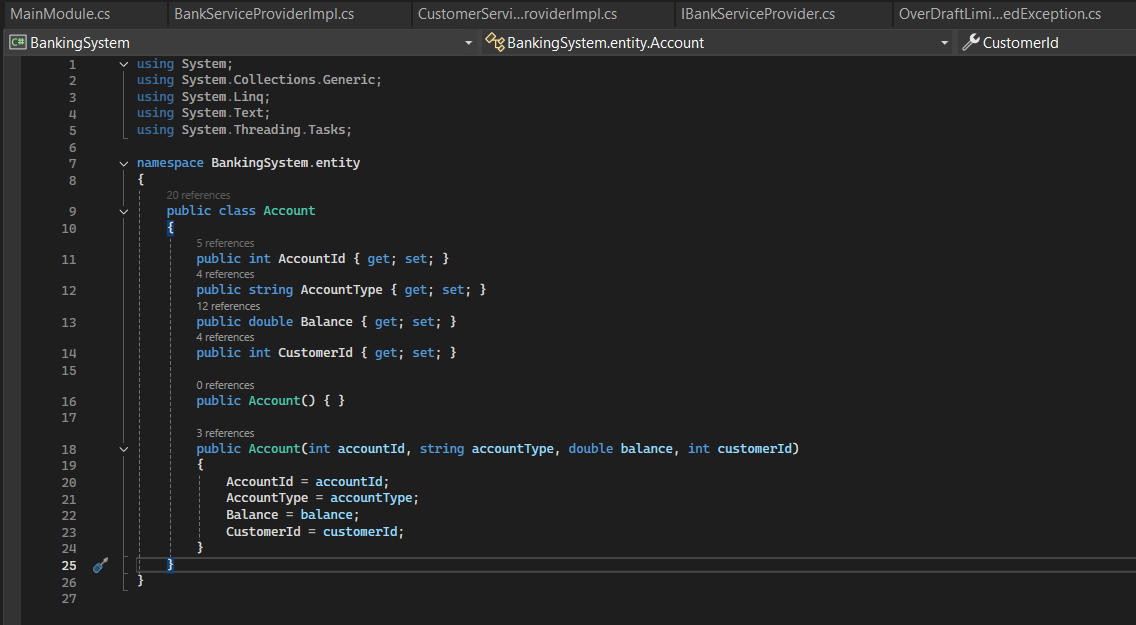
Task 11: Interface/abstract class, and Single Inheritance, static variable

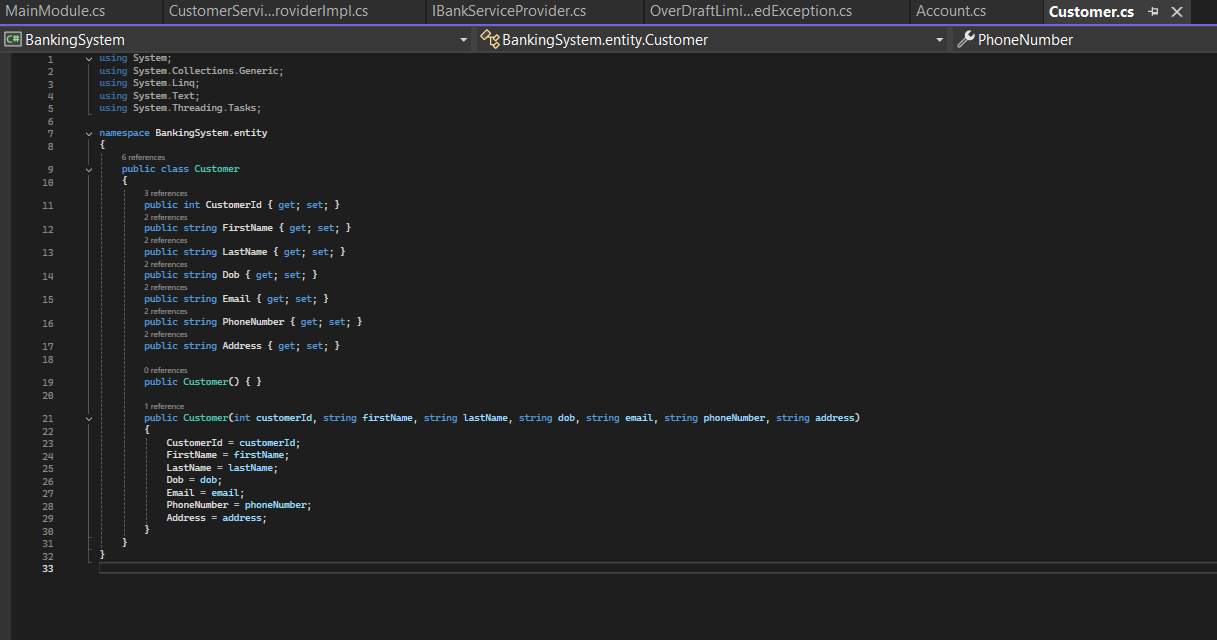
Task 12: Exception Handling

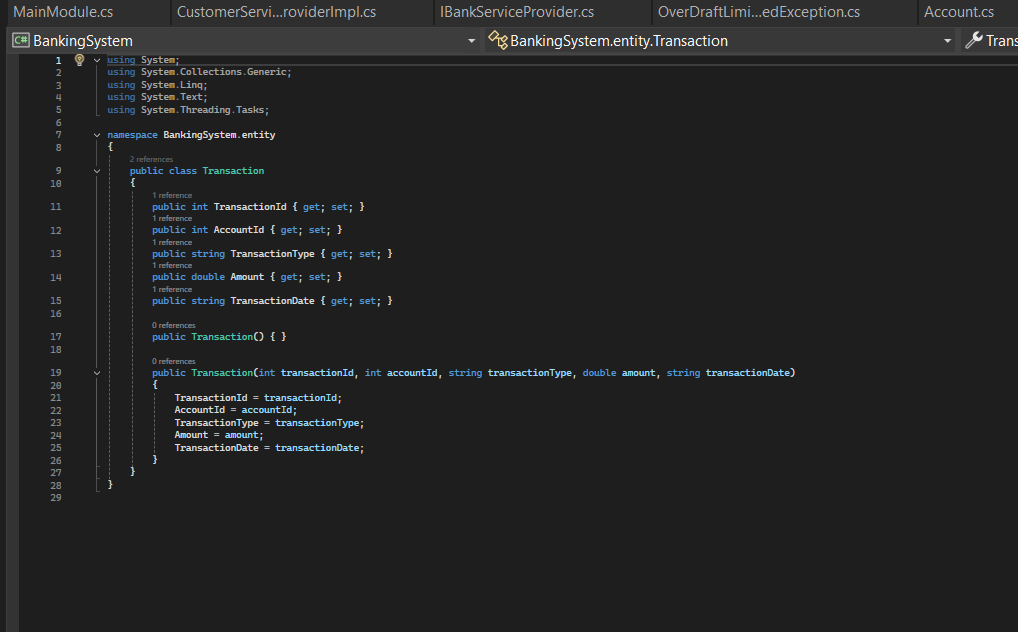
Task 13: Collection

Task 14: Database Connectivity.

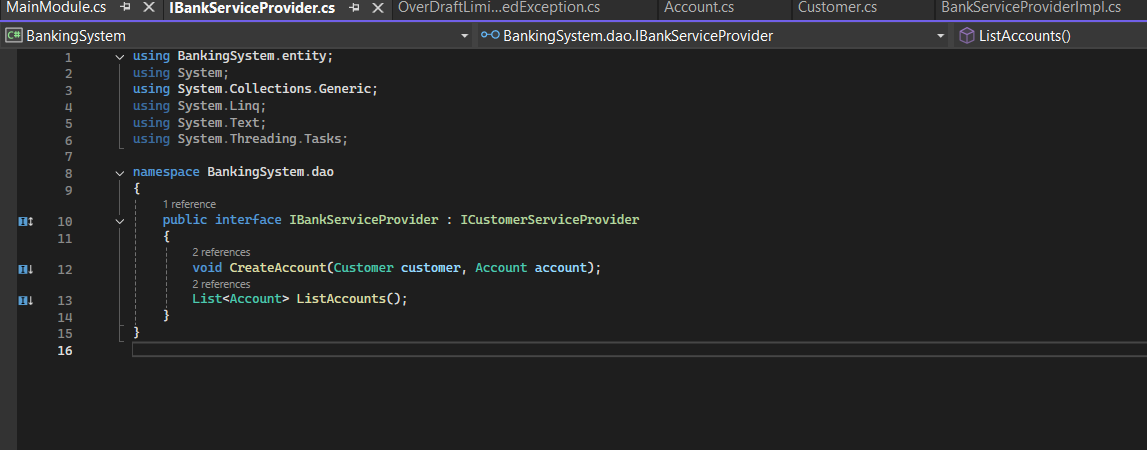
Entity

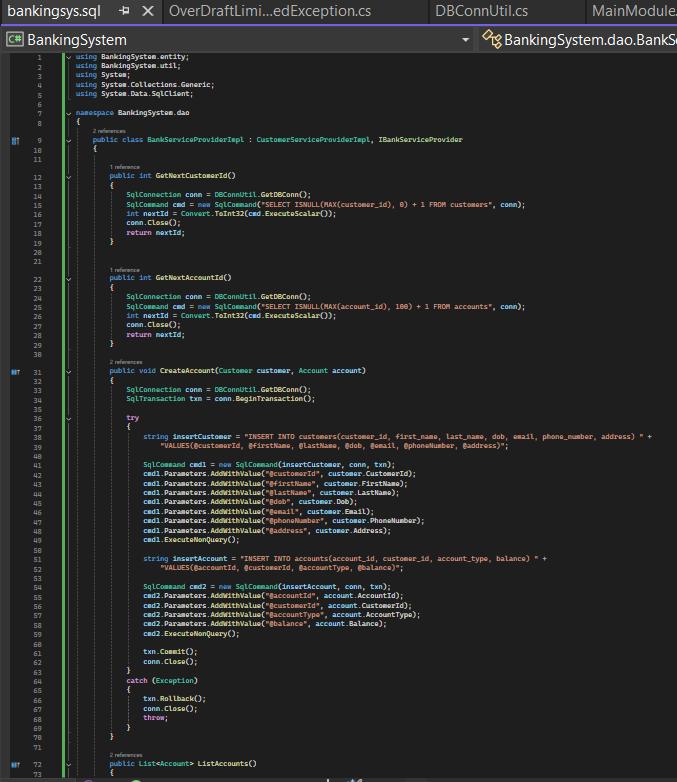


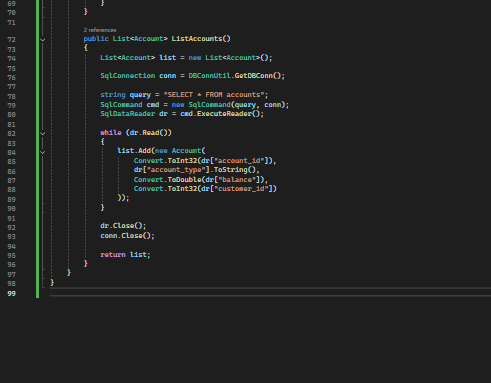


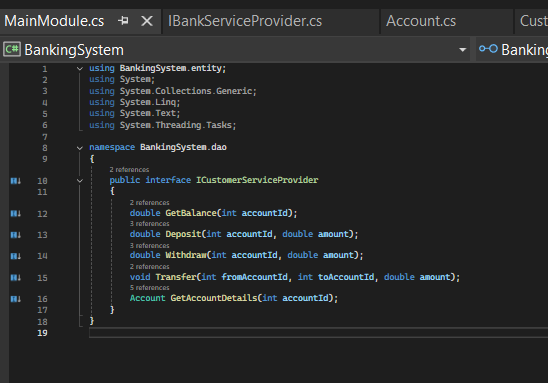


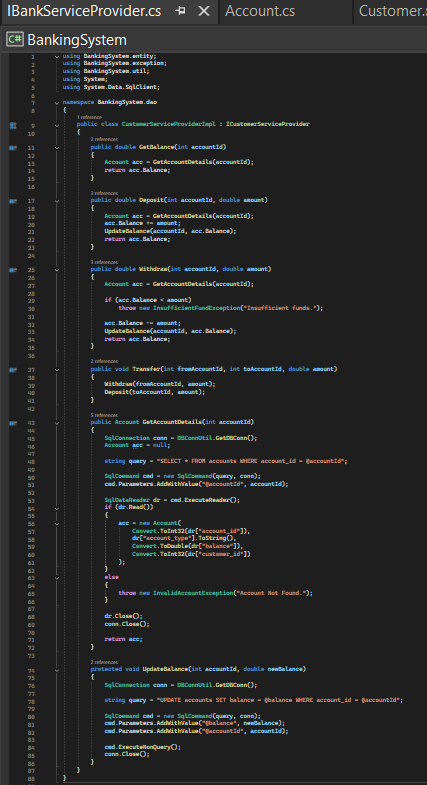
Dao:



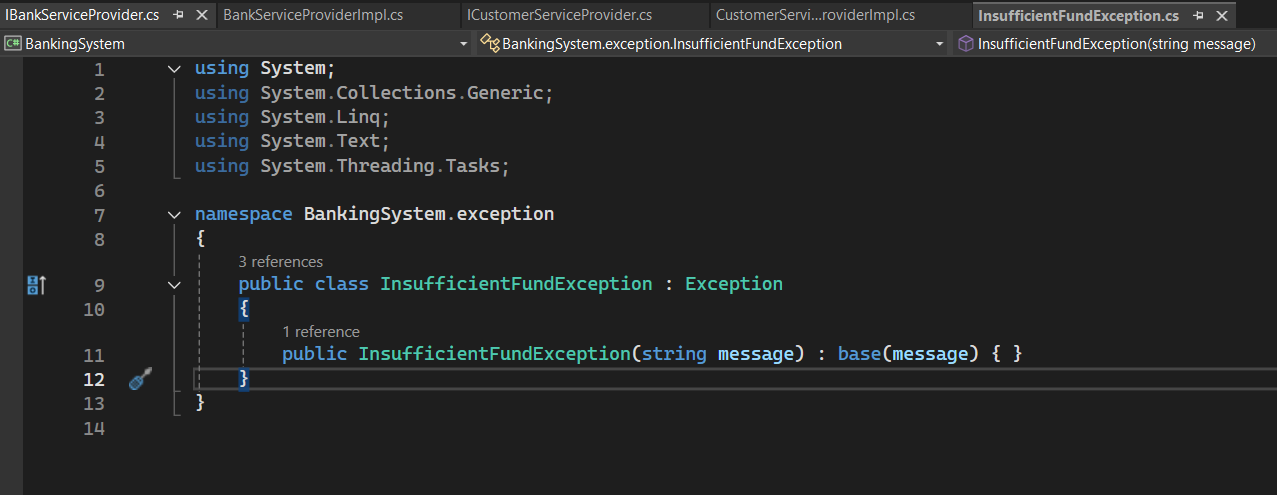


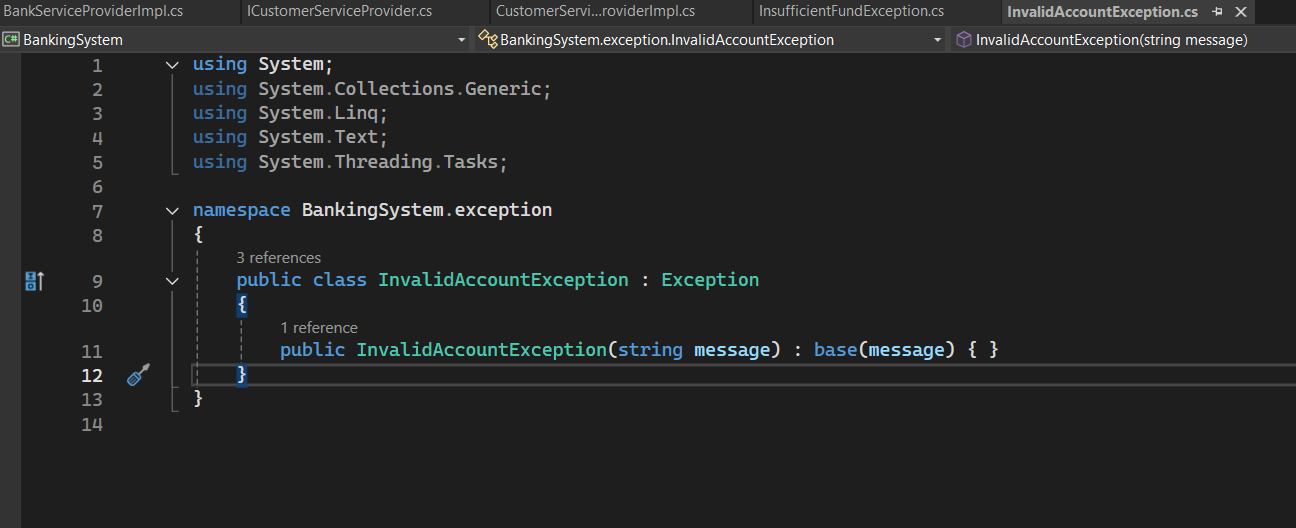


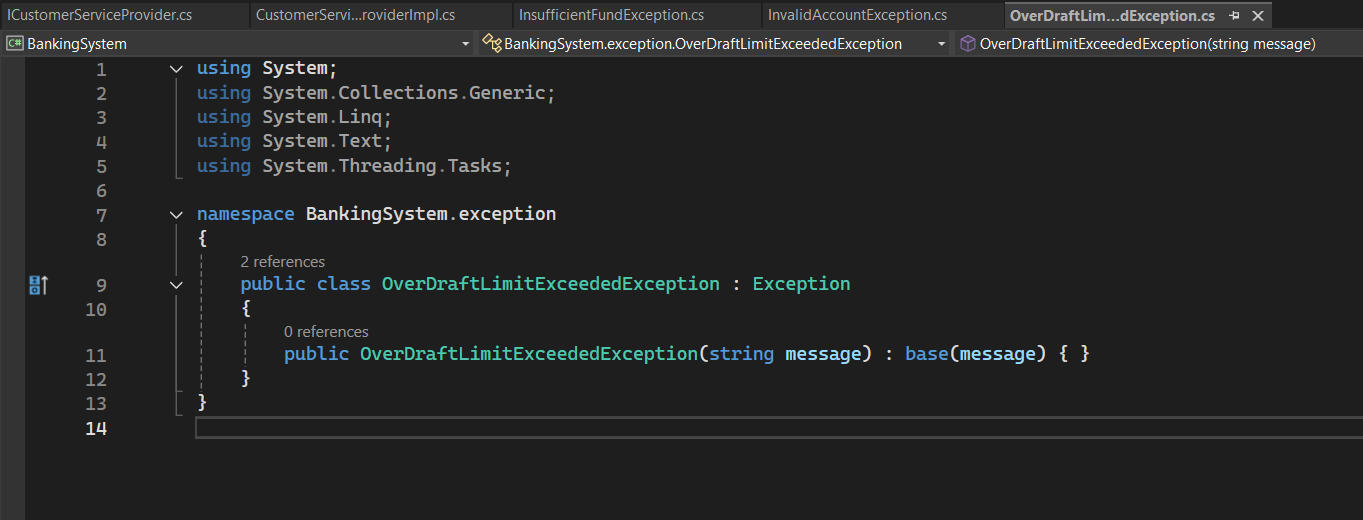




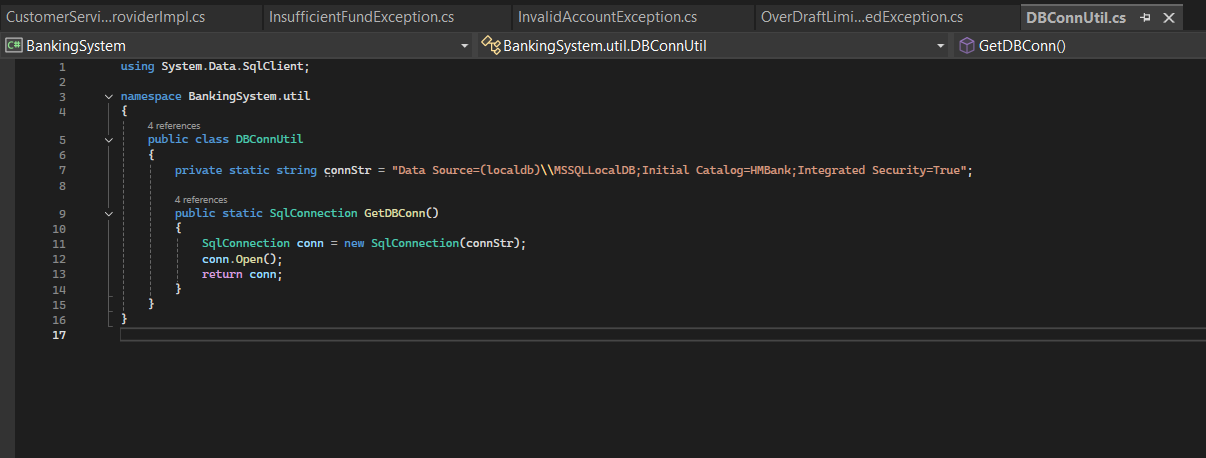
Exception:



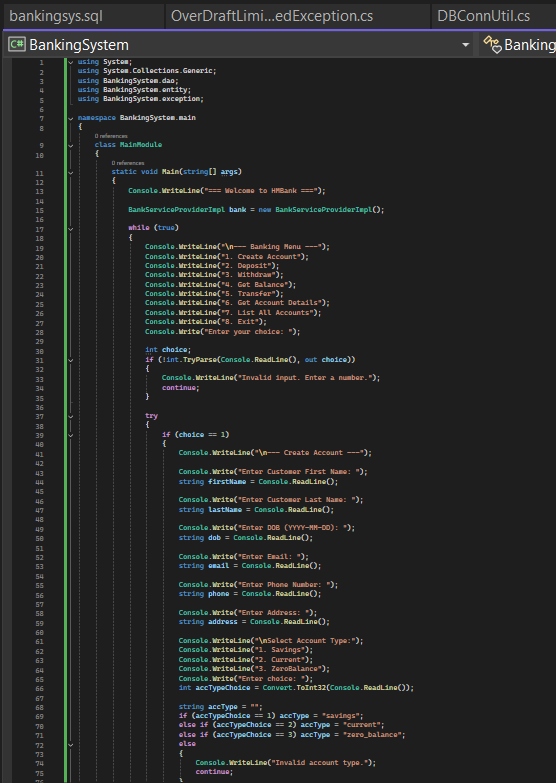




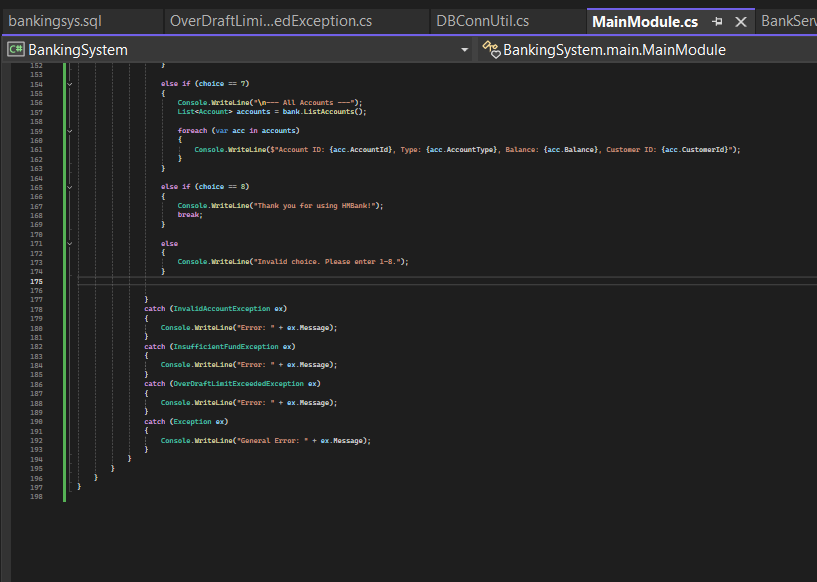
Util:



Main:







Output:

