LIMKOKWING UNIVERSITY OF CREATIVE TECHNOLOGY

CAMBODIA

Faculty of Information & Communication Technology

Software Engineering with Multimedia

Semester 3

Database System

Lecturer: Leng Rathana

Group Members:

Name : Meng Chanlisa ID : 601005833	Name : Soy Rithy ID : 601005829	Name : Phin Samaivironath ID : 601006196

Table of Contents

I. Introduction

1. Introduction to business entity

The business entities of an Automated Teller Machine (ATM) system refer to organizations or companies that. Operate and manage a network of ATMs. An ATM is a critical component of the banking and financial industry, providing convenience and self - service banking solutions to customers. As the term suggests , it is an automated banking platform that does not require any self-service bank tellers, they're maintained by banks and provide facilities for users. In addition, an ATM allows users to withdraw money, check balance or even transfer funds, also they're accessible anywhere and anytime. ATMs are typically located at banks, supermarkets, gas stations, educational centers such as universities, hospitals, airports and any other parts of the country across the nation. Besides, ATMs also have become an important and crucial role for obtaining a competitive edge not only by retaining customers but also by increasing overall profitability. With the advent of these machines, we no longer need to carry around a lot of cash, in the view of the fact that all banks now offer ATMs and PIN codes. On top of that, as long as PIN codes are provided by ATMs, they also significantly reduce the risk of theft as well.

As a sequence, we have to state that ATMs play an undeniable role in our country's economy. You can't spend a day out in a country without ATMs.Several banks provide ATM services by installing ATMs in different parts of the country. They really play a very important role in our daily life and in our society.

2. Overview of the proposed database system

A proposed database system for an ATM system would be designed to efficiently manage and store the data related to ATM operations, transactions, customer information, and network management. The database system would play a crucial role in ensuring the smooth functioning of the ATM network and providing reliable and secure services to customers. Here's an overview of the key components and functionalities that could be included in the proposed database system:

+ ATM Information:
  . Store information about each individual ATM, including its unique identifier, location, status, and operational details.
  . Maintain ATM configuration data such as supported transaction types, withdrawal limits, language preferences, etc.
  Track software versions and updates applied to each ATM.

+ Customer Information:
  . Store customer details, including personal information, account numbers, contact information, and authentication credentials.
  . Maintain customer transaction history, including deposits, withdrawals, transfers, and balance inquiries.
  . Implement appropriate security measures to protect sensitive customer data.

+ Transaction Management:
  . Track all ATM transactions, including the date, time, ATM identifier, transaction type, and transaction status.
  . Store details of each transaction, such as the amount withdrawal, deposit details, transfer recipients, and account balances.
  . Implement transaction reconciliation mechanisms to ensure data consistency and accuracy.

+ Network Management:
  . Maintain information about the entire ATM network, including the participating banks, financial institutions, and their respective connectivity details.
  . Track network connectivity status, response times, and error logs for troubleshooting purposes.
  . Implement monitoring and alerting mechanisms to proactively identify and address network issues.

+ Security and Access Control:
  . Implement robust security measures to protect the database from unauthorized access, tampering, or data breaches.
  . Enforce appropriate access controls based on user roles and privileges.
  . Log and audit all database activities to maintain an audit trail and detect any potential security breaches.

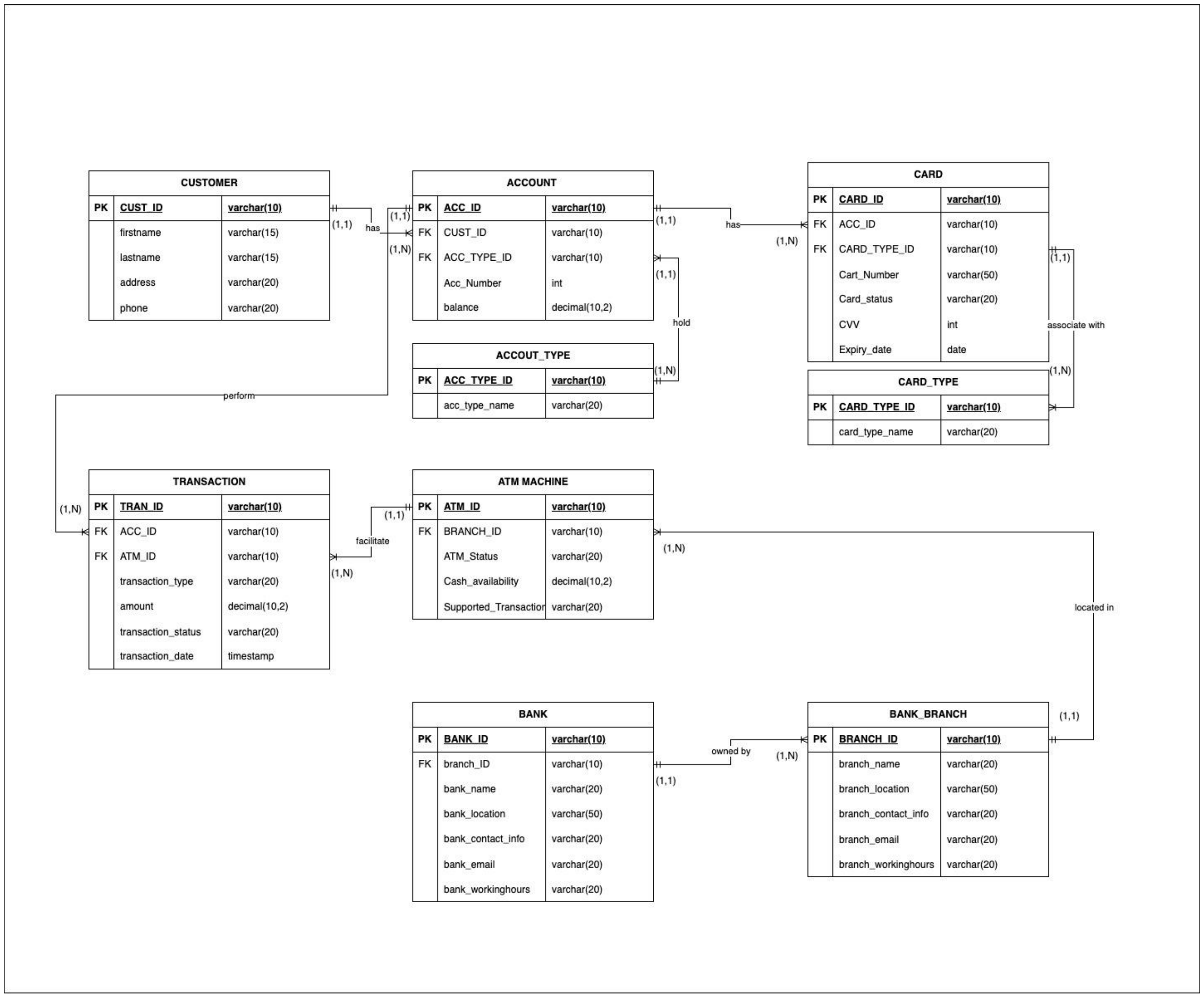
+ Reporting and Analytics:
  . Generate reports and analytics on various aspects, such as ATM performance, transaction trends, network uptime, and customer behavior.
  . Provide tools for data analysis and visualization to gain insights and support decision-making processes.

+ Backup and Recovery:
  . Implement regular backups of the database to ensure data integrity and availability.

. Define a disaster recovery plan to restore the database in case of system failures or data loss.

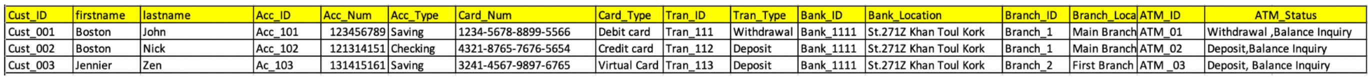
II. Database Design

1. An Entity Relationship Diagram (ERD)



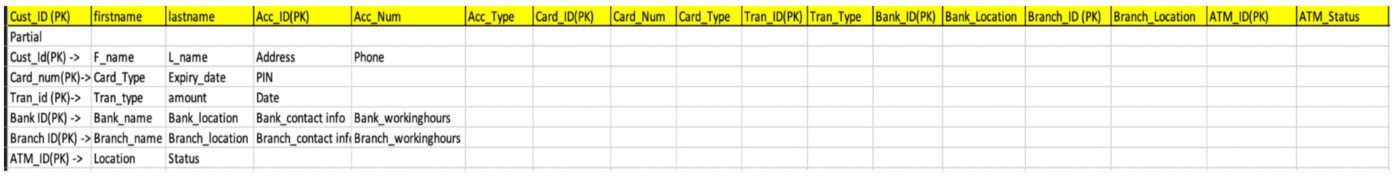
2. Normalization Table

   2.1 Denormalization Table



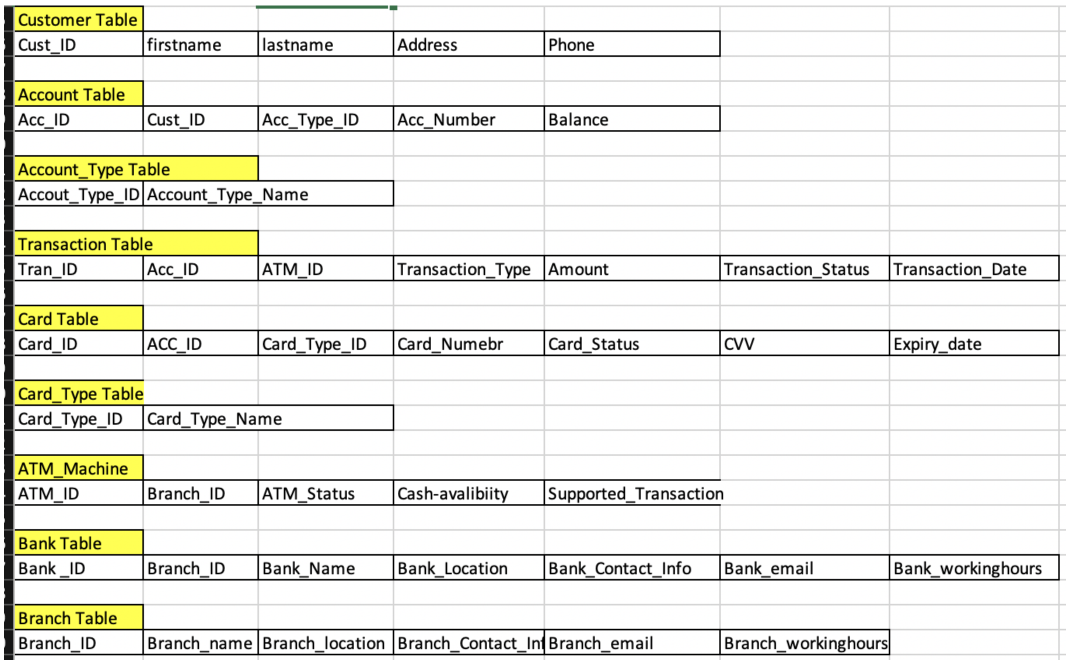
Cust_ID	firstname	lastname	Acc_ID	Acc_Num	Acc_Type	Card_Num	Card_Type	Tran_ID	Tran_Type	Bank_ID	Bank_Location	Branch_ID	Branch_Loca	ATM_ID	ATM_Status
Cust_001	Boston	John	Acc_101	123456789	Saving	1234-5678-8899-5566	Debit card	Tran_111	Withdrawal	Bank_1111	St.271Z Khan Toul Kork	Branch_1	Main Branch	ATM_01	Withdrawal ,Balance Inquiry
Cust_002	Boston	Nick	Acc_102	121314151	Checking	4321-8765-7676-5654	Credit card	Tran_112	Deposit	Bank_1111	St.271Z Khan Toul Kork	Branch_1	Main Branch	ATM_02	Deposit,Balance Inquiry
Cust_003	Jennier	Zen	Ac_103	131415161	Saving	3241-4567-9897-6765	Virtual Card	Tran_113	Deposit	Bank_1111	St.271Z Khan Toul Kork	Branch_2	First Branch	ATM _03	Deposit, Balance Inquiry

2.2 First Normalization Table (1NF)



Cust_ID (PK)	firstname	lastname	Acc_ID(PK)	Acc_Num	Acc_Type	Card_ID(PK)	Card_Num	Card_Type	Tran_ID(PK)	Tran_Type	Bank_ID(PK)	Bank_Location	Branch_ID (PK)	Branch_Location	ATM_ID(PK)	ATM_Status
Partial																
Cust_Id(PK) ->	F_name	L_name	Address	Phone												
Card_num(PK)->	Card_Type	Expiry_date	PIN													
Tran_id (PK)->	Tran_type	amount	Date													
Bank ID(PK) ->	Bank_name	Bank_location	Bank_contact info	Bank_workinghours												
Branch ID(PK) ->	Branch_name	Branch_location	Branch_contact inf	Branch_workinghours												
ATM_ID(PK) ->	Location	Status														

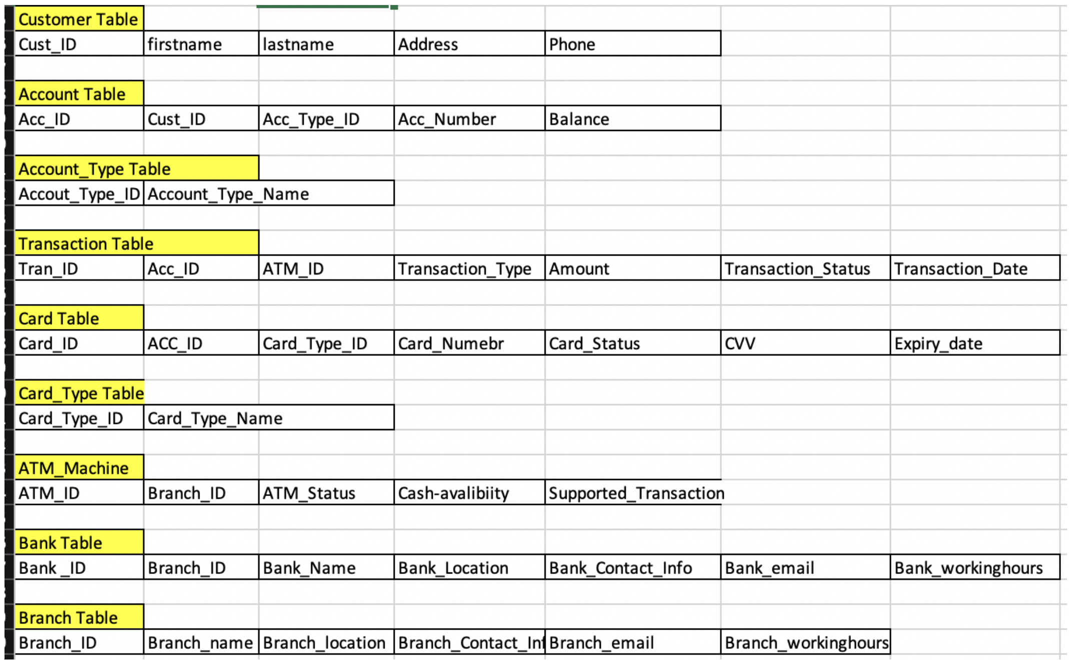
2.3 Second Normalization Table (2NF)



Customer Table						
Cust_ID	firstname	lastname	Address	Phone		
Account Table						
Acc_ID	Cust_ID	Acc_Type_ID	Acc_Number	Balance		
Account_Type Table						
Accout_Type_ID	Account_Type_Name					
Transaction Table						
Tran_ID	Acc_ID	ATM_ID	Transaction_Type	Amount	Transaction_Status	Transaction_Date
Card Table						
Card_ID	ACC_ID	Card_Type_ID	Card_Numebr	Card_Status	CVV	Expiry_date
Card_Type Table						
Card_Type_ID	Card_Type_Name					
ATM_Machine						
ATM_ID	Branch_ID	ATM_Status	Cash-avalibiity	Supported_Transaction		
Bank Table						
Bank _ID	Branch_ID	Bank_Name	Bank_Location	Bank_Contact_Info	Bank_email	Bank_workinghours
Branch Table						
Branch_ID	Branch_name	Branch_location	Branch_Contact_Inf	Branch_email	Branch_workinghours	

2.4 Third Normalization Table (3NF)

There is no transitive dependency so the table in third normalization remains the same as second normalization.



Customer Table

Cust_ID	firstname	lastname	Address	Phone

Account Table

Acc_ID	Cust_ID	Acc_Type_ID	Acc_Number	Balance

Account_Type Table

Accout_Type_ID	Account_Type_Name

Transaction Table

Tran_ID	Acc_ID	ATM_ID	Transaction_Type	Amount	Transaction_Status	Transaction_Date

Card Table

Card_ID	ACC_ID	Card_Type_ID	Card_Numebr	Card_Status	CVV	Expiry_date

Card_Type Table

Card_Type_ID	Card_Type_Name

ATM_Machine

ATM_ID	Branch_ID	ATM_Status	Cash-avalibiity	Supported_Transaction

Bank Table

Bank _ID	Branch_ID	Bank_Name	Bank_Location	Bank_Contact_Info	Bank_email	Bank_workinghours

Branch Table

Branch_ID	Branch_name	Branch_location	Branch_Contact_Inf	Branch_email	Branch_workinghours

3. Data Dictionary

Table Name	Attribute Name	Data Type	Range	PR or FK	Required
Customer	Cust_ID	varchar(10)	0-10	PK	Yes
	firstname	varchar(15)	0-15		No
	lastname	varchar(20)	0-20		No
	address	varchar(20)	0-20		No
	phone	cachar(20)	0-20		No
Account	Account_ID	varchar(10)	0-10	PK	Yes
	Cust_ID	varchar(10)	0-10	FK	No
	Account_Type _ID	varchar(10)	0-10	FK	No
	Acc_Number	varchar(10)	0-10		No
	Balance	decimal(10,2)	0-10		No
Accout_Type	Account_Type_ID	varchar(10)	0-10	PK	Yes
	Account_Type_Name	varchar(20)	0-10		No
Card	Card_ID	varchar(10)	0-10	PK	Yes
	Acc_ID	varchar(10)	0-10	FK	No
	Card_Type_ID	varchar(10)	0-10	FK	No
	Card_Num	varchar(50)	0-50		No
	Card_Status	varchar(20)	0-10		No
	CVV	int	0-N		No
	Expiry_date	date	0-N		No
Card_Type	Card_Type_ID	varchar(10)	0-N	PK	Yes
	Card_Type_Name	varchar(20)	0-20		No
Transaction	Transaction_ID	varchar(10)	0-10	PK	Yes
	Account_Num	varchar(10)	0-10	FK	No
	ATM_ID	varchar(20)	0-20	FK	No
	Transaction_Type	varchar(20)	0-20		No

	Amount	decimal(10,2)	0-N		No
	Transaction_Status	varchar(20)	0-20		No
	Transaction_Date	timestamp	0-N		No
ATM Machine	ATM_ID	varchar(10)	0-10	PK	Yes
	Branch_ID	varchar(10)	0-10	FK	No
	Status	varchar(20)	0-20		No
	Cash_availibity	decimal(10,2)			No
	Supported_Transaction	varchar(10)	0-10		No
Bank	Bank_ID	varchar(10)	0-10	PK	Yes
	Branch_ID	varchar(10)	0-10	FK	No
	Bank_Name	varchar(20)	0-10		No
	Bank_Location	varchar(50)	0-50		No
	Bank_Contact_Info	varchar(20)	0-20		No
	Bank_Email	varchar(20)	0-20		No
	Bank_Workinghours	varchar(20)	0-N		No
Bank_Branch	Branch_ID	varchar(10)	0-10	FK	Yes
	Branch_Name	varchar(50)	0-50		No
	Branch_Location	varchar(20)	0-20		No
	Branch_Contact_Info	varchar(20)	0-20		No
	Branch_Email	varchar(20)	0-20		No
	Branch_Workinghours	varchar(20)	0-20		No

III. Create and Insert to Database

+ Create

. Create the database

```
create database ATM_SYSTEM;
```

. Use the database

```
use ATM_SYSTEM;
```

. Create Customer table

```
CREATE TABLE Customer (
    Cust_ID  VARCHAR(10)PRIMARY KEY,
    firstname VARCHAR(15),
    lastname VARCHAR(15),
    address VARCHAR(20),
    phone VARCHAR(20)
);
```

. Create Account table

```
CREATE TABLE Account (
        Acc_ID VARCHAR(10) PRIMARY KEY,
    Account_Num INT,
        Cust_ID  VARCHAR(10),
    Account_Type_ID VARCHAR(10),
    Balance DECIMAL(10, 2),
    FOREIGN KEY (Cust_ID) REFERENCES Customer(Cust_ID),
    FOREIGN KEY (Account_Type_ID) REFERENCES
Account_Type(Account_Type_ID)
);
```

. Create Account_Type table

```
CREATE TABLE Account_Type (
    Account_Type_ID VARCHAR(10) PRIMARY KEY,
    Account_Type_Name VARCHAR(20)
);
```

. Create Card table

```
CREATE TABLE Card (
        Card_ID INT PRIMARY KEY,
    Acc_ID VARCHAR(10),
    Card_Type_ID VARCHAR(10),
    Card_Number INT,
    Card_Status VARCHAR(50),
    CVV INT,
    Expiry_date DATE,
    FOREIGN KEY (Acc_ID) REFERENCES Account(Acc_ID),
    FOREIGN KEY (Card_Type_ID) REFERENCES Card_Type(Card_Type_ID)
);
        ALTER TABLE Card MODIFY COLUMN Card_ID Varchar(10);
```

. Create Card_Type table

```
CREATE TABLE Card_Type (
   Card_Type_ID VARCHAR(10) PRIMARY KEY,
   Card_Type_Name VARCHAR(20)
);
```

. Create Transaction table

```
CREATE TABLE Transaction (
   Transaction_ID INT PRIMARY KEY,
   Acc_ID VARCHAR(10),
   ATM_ID VARCHAR(10),
   Transaction_Type VARCHAR(20),
   Amount DECIMAL(10, 2),
   Transaction_Status VARCHAR(20),
   Transaction_Date DATETIME,
   FOREIGN KEY (Acc_ID) REFERENCES Account(Acc_ID),
   FOREIGN KEY (ATM_ID) REFERENCES ATM_Machine (ATM_ID)
);
ALTER TABLE Transaction MODIFY COLUMN Transaction_Date timestamp;
ALTER TABLE Transaction MODIFY COLUMN Transaction_ID Varchar(10);
```

. Create Bank table

```
CREATE TABLE Bank (
   Bank_ID VARCHAR (10) PRIMARY KEY,
   Bank_Name VARCHAR(20),
   Bank_Location VARCHAR(50),
   Bank_Contact_Info VARCHAR(20),
   Bank_Email VARCHAR(20),
   Bank_Workinghours time
);
ALTER TABLE Bank MODIFY COLUMN Bank_Workinghours VARCHAR(20);
```

. Create Bank_Branch table

```
CREATE TABLE Bank_Branch (
   Branch_ID VARCHAR (10) PRIMARY KEY,
   Bank_ID VARCHAR (10),
   Branch_Name VARCHAR(20),
   Branch_Location VARCHAR(50),
   Branch_Contact_Info VARCHAR(20),
            Branch_Email VARCHAR(20),
            Branch_Workinghours time,
            FOREIGN KEY (Bank_ID) REFERENCES Bank(Bank_ID)
        );
ALTER TABLE Bank_Branch MODIFY COLUMN Branch_Workinghours VARCHAR(20);
```

. <u>Create ATM Machine table</u>

```
        CREATE TABLE ATM_Machine (
            ATM_ID VARCHAR(10) PRIMARY KEY,
            Branch_ID VARCHAR (10),
            ATM_Status VARCHAR(20),
            Cash_availibity DECIMAL(10, 2),
            Supported_Transaction VARCHAR(20),
            FOREIGN KEY (Branch_ID) REFERENCES Bank_Branch(Branch_ID)
        );
ALTER TABLE ATM_Machine MODIFY COLUMN Supported_Transaction Varchar(200);
```

+ **Insert**

. <u>Insert data into Customer table</u>

```
        INSERT INTO Customer (Cust_ID, firstname, lastname, address, phone) VALUES
        ("Cust_001", "Adam","John","Phnom Penh", "090-645-789"),
        ("Cust_002","Boston","Nick","Kandal", "092-645-779"),
        ("Cust_003","Jennier","Zen","Phnom Penh", "095-645-783"),
        ("Cust_004","Sunny","Sea","Kampong Chhang", "010-745-235"),
        ("Cust_005","Mike","Andrew","Phnom Pehn", "010-234-232"),
        ("Cust_006","Sok","San","Bathambong", "012-264-432"),
        ("Cust_007","Milli","Mee","Phnom Penh", "016-224-879"),
        ("Cust_008","Theary","Kun","Kandal", "092-786-345"),
        ("Cust_009","Nirdey","Sila","Kandal", "078-784-888"),
        ("Cust_010","Thany","Ly","Phnom Penh", "098-567-777");
```

. <u>Insert data into Account_Type table</u>

```
        INSERT INTO Account_Type (Account_Type_ID, Account_Type_Name) VALUES
        ("Sav_1", "Savings"),
        ("Che_2", "Checking");
```

. <u>Insert data into Account table</u>

```
        INSERT INTO Account (Acc_ID, Account_Num, Cust_ID, Account_Type_ID,
        Balance) VALUES
        ("Acc_101", 123456789, "Cust_001", "Sav_1", 10000.00),
        ("Acc_102", 121314151, "Cust_002", "Che_2", 30000.00),
        ("Acc_103", 131415161, "Cust_003", "Sav_1", 40000.00),
        ("Acc_104", 131516171, "Cust_004", "Sav_1", 300.00),
        ("Acc_105", 131415161, "Cust_005", "Sav_1", 55000.00),
        ("Acc_106", 141516171, "Cust_006", "Che_2", 3400.00),
        ("Acc_107", 141617189, "Cust_007", "Che_2", 6700.00),
        ("Acc_108", 141718191, "Cust_008", "Sav_1", 56000.00),
        ("Acc_109", 151617189, "Cust_009", "Sav_1", 100000.00),
        ("Acc_110", 151414121, "Cust_010", "Che_2", 56000.00);
```

. <u>Insert data into Card_Type table</u>

```
        INSERT INTO Card_Type (Card_Type_ID, Card_Type_Name) VALUES
        ("DC_1", 'Debit Card'),
        ("CC_2", 'Credit Card'),
        ("VC_3", 'Virtual Card');
```

. <u>Insert data into Card table</u>

```
INSERT INTO Card (Card_ID, Acc_ID, Card_Type_ID, Card_Number, Card_Status, CVV,
Expiry_date) VALUES
("Card_1001","Acc_101", "DC_1", 1234-5678-8899-5566, "Active",123, '2025-12-31'),
("Card_1002","Acc_102", "CC_2", 4321-8765-7676-5654, "Active",456, '2030-12-31'),
("Card_1003","Acc_103", "VC_3", 3241-4567-9897-6765, "Freeze",778, '2025-01-01'),
("Card_1004","Acc_104", "DC_1", 5554-7876-9998-6644, "Active",334, '2029-01-01'),
("Card_1005","Acc_105", "CC_2", 4532-7652-4566-9998, "Active",221, '2027-05-30'),
("Card_1006","Acc_106", "VC_3", 3332-4532-3321-5678, "Freeze",443, '2027-03-31'),
("Card_1007","Acc_107", "VC_3", 3241-4567-9897-6765, "Freeze",778, '2025-01-01'),
("Card_1008","Acc_108", "DC_1", 0998-6655-4356-3323, "Active",334, '2029-01-01'),
("Card_1009","Acc_109", "DC_1", 3240-1020-9087-6789, "Active",314, '2023-05-30'),
("Card_1010","Acc_110", "CC_2", 0976-9807-3340-2198, "Active",433, '2024-03-31');
```

. <u>Insert data into Transaction table</u>

```
        INSERT INTO Transaction (Transaction_ID, Acc_ID, ATM_ID, Transaction_Type,
Amount, Transaction_Status, Transaction_Date) VALUES
        ("Tran_111", "Acc_101", "ATM_01", "Withdrawal", 1000.00, 'Completed',
'2023-10-29 15:30:00'),
        ("Tran_112", "Acc_102", "ATM_02", "Deposit", 5000.00, 'Completed', '2023-08-29
08:30:00'),
        ("Tran_113", "Acc_103", "ATM_03", "Deposit", 200.00, 'Pending', Null),
```

("Tran_114", "Acc_104", "ATM_04", "Withdrawal", 780.00, 'Completed', '2023-06-30 09:45:10'),
("Tran_115", "Acc_105", "ATM_02", "Deposit", 4000.00, 'Completed', '2023-06-30 09:50:10'),
("Tran_116", "Acc_106", "ATM_05", "Withdrawal", 30.00, 'Completed', '2023-10-31 15:50:00'),
("Tran_117", "Acc_107", "ATM_03", "Deposit", 4000.00, 'Pending', Null),
("Tran_118", "Acc_108", "ATM_05", "Deposit", 1000.00, 'Completed', '2023-03-31 17:30:00'),
("Tran_119", "Acc_109", "ATM_01", "Withdrawal", 120.00, 'Completed', '2022-02-28 14:30:00'),
("Tran_120", "Acc_110", "ATM_04", "Withdrawal", 900.00, 'Completed', '2022-04-26 20:57:00');

. Insert data into Bank table

INSERT INTO Bank (Bank_ID, Bank_Name, Bank_Location, Bank_Contact_Info, Bank_Email, Bank_Workinghours) VALUES
("Bank_1111", "ABC Bank", "St.271Z Khan Toul Kork", '012-999-888', "abcbank@gmail.com", '09:00:00 - 05:00:00');

. Insert data into Bank_Branch table

INSERT INTO Bank_Branch (Branch_ID, Bank_ID, Branch_Name, Branch_Location, Branch_Contact_Info, Branch_Email, Branch_Workinghours) VALUES
("Branch_1","Bank_1111", "Main Branch", "St.271Z Khan Toul Kork", '012-999-888', "abcbank1@gmail.com", '09:00:00 - 05:00:00'),
("Branch_2","Bank_1111", "Second Branch", "St.360 Khan Sen Sok", '017-999-888', "abcbank2@gmail.com", '09:00:00 - 05:00:00'),
("Branch_3","Bank_1111", "Third Branch", "St.157 Khan Mean Chey", '089-999-888', "abcbank3@gmail.com", '09:00:00 - 05:00:00');

. Insert data into ATM_Machine table

INSERT INTO ATM_Machine (ATM_ID, Branch_ID, ATM_Status, Cash_availibity, Supported_Transaction) VALUES
("ATM_01", "Branch_1", "In Service", 100000.00, "Withdrawal ,Balance Inquiry"),
("ATM_02", "Branch_1", "In Service", 100000.00, "Deposit,Balance Inquiry"),
("ATM_03", "Branch_2", "Under Maintenance", 100000.00, "Deposit, Balance Inquiry"),
("ATM_04", "Branch_2", "In Service", 200000.00, "Withdrawal, Balance Inquiry"),

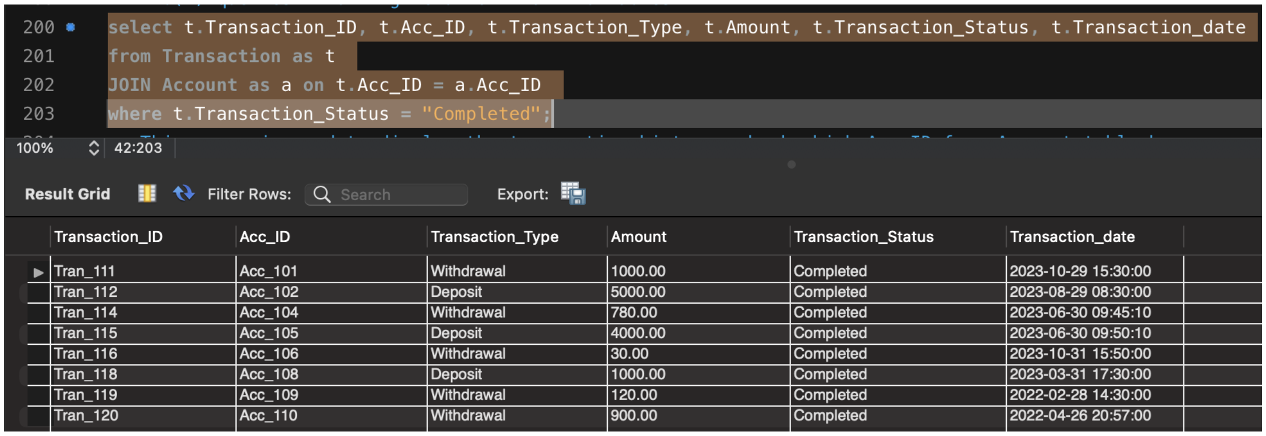
("ATM_05", "Branch_3", "In Service", 250000.00, " Deposit, Withdrawal, Balance Inquiry");

IV. Testing and Evaluation

+ TWO(2) queries involving relation from two tables

.This query is used to display the transaction history, check which Acc_ID from the Account table has successfully completed the transaction.

```
select t.Transaction_ID, t.Acc_ID, t.Transaction_Type, t.Amount, t.Transaction_Status, t.Transaction_date
from Transaction as t
JOIN Account as a on t.Acc_ID = a.Acc_ID
where t.Transaction_Status = "Completed";
```

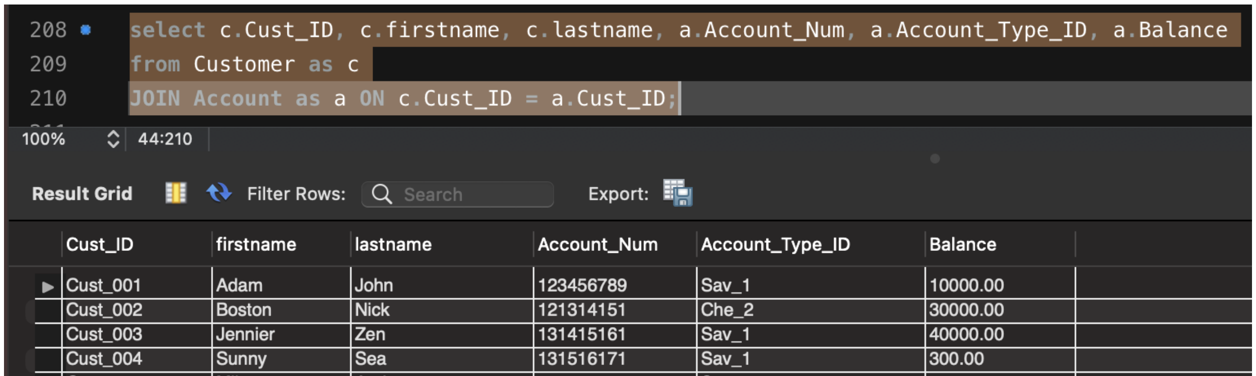


```
select t.Transaction_ID, t.Acc_ID, t.Transaction_Type, t.Amount, t.Transaction_Status, t.Transaction_date
from Transaction as t
JOIN Account as a on t.Acc_ID = a.Acc_ID
where t.Transaction_Status = "Completed";
```

Transaction_ID	Acc_ID	Transaction_Type	Amount	Transaction_Status	Transaction_date
Tran_111	Acc_101	Withdrawal	1000.00	Completed	2023-10-29 15:30:00
Tran_112	Acc_102	Deposit	5000.00	Completed	2023-08-29 08:30:00
Tran_114	Acc_104	Withdrawal	780.00	Completed	2023-06-30 09:45:10
Tran_115	Acc_105	Deposit	4000.00	Completed	2023-06-30 09:50:10
Tran_116	Acc_106	Withdrawal	30.00	Completed	2023-10-31 15:50:00
Tran_118	Acc_108	Deposit	1000.00	Completed	2023-03-31 17:30:00
Tran_119	Acc_109	Withdrawal	120.00	Completed	2022-02-28 14:30:00
Tran_120	Acc_110	Withdrawal	900.00	Completed	2022-04-26 20:57:00

. This query is used to display the Customer information associated with their account information such as account_num, account_type_id and balance.

```
SELECT c.Cust_ID, c.firstname, c.lastname, a.Account_Num, a.Account_Type_ID, a.Balance
FROM Customer as c
JOIN Account as a ON c.Cust_ID = a.Cust_ID;
```



```
select c.Cust_ID, c.firstname, c.lastname, a.Account_Num, a.Account_Type_ID, a.Balance
from Customer as c
JOIN Account as a ON c.Cust_ID = a.Cust_ID;
```

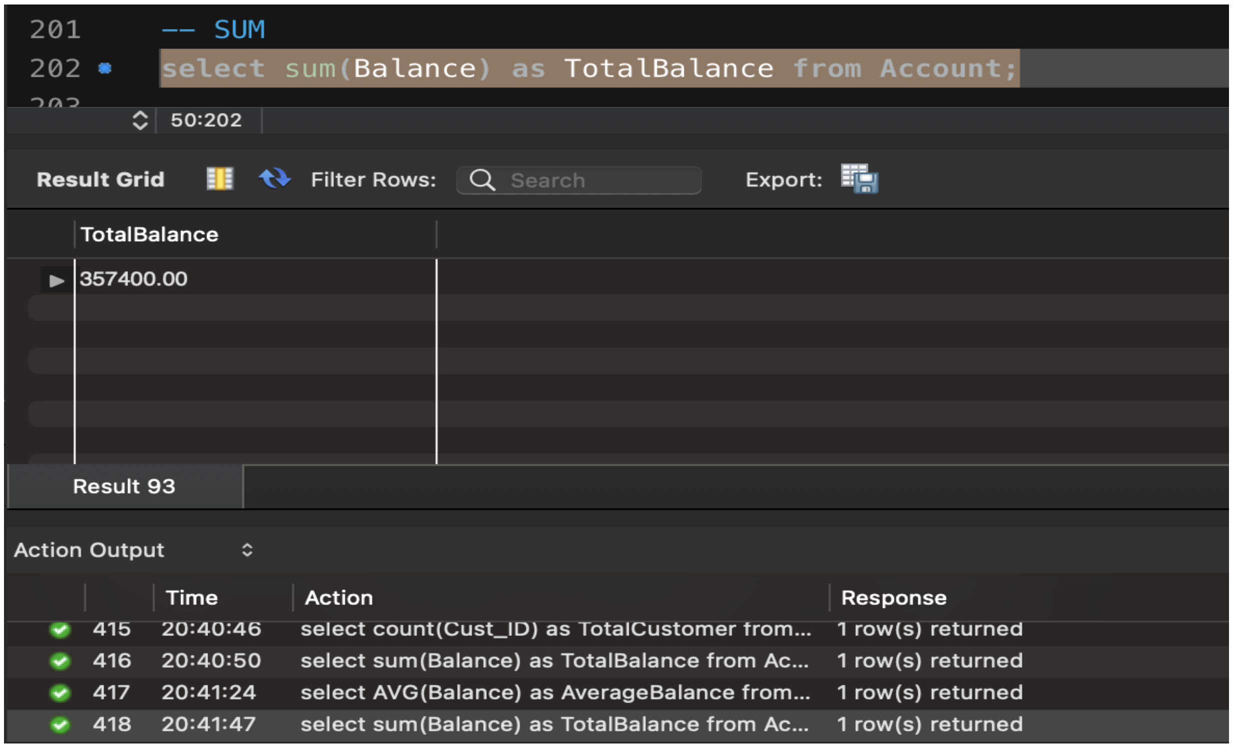
Cust_ID	firstname	lastname	Account_Num	Account_Type_ID	Balance
Cust_001	Adam	John	123456789	Sav_1	10000.00
Cust_002	Boston	Nick	121314151	Che_2	30000.00
Cust_003	Jennier	Zen	131415161	Sav_1	40000.00
Cust_004	Sunny	Sea	131516171	Sav_1	300.00

+ **Queries involving aggregate functions such as SUM, COUNT, AVG, MAX, MIN.**

_ **Sum**

. This query means calculating the sum of balance in the account table.

```
select sum(Balance) as TotalBalance from Account;
```



```
-- SUM
select sum(Balance) as TotalBalance from Account;
```

50:202

Result Grid | Filter Rows: Search | Export:

TotalBalance
357400.00

Result 93

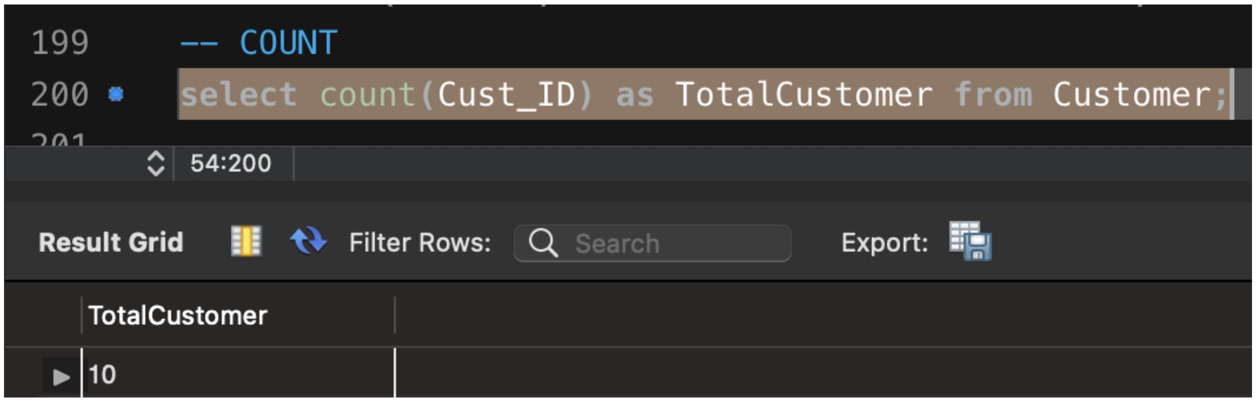
Action Output

	Time	Action	Response
415	20:40:46	select count(Cust_ID) as TotalCustomer from...	1 row(s) returned
416	20:40:50	select sum(Balance) as TotalBalance from Ac...	1 row(s) returned
417	20:41:24	select AVG(Balance) as AverageBalance from...	1 row(s) returned
418	20:41:47	select sum(Balance) as TotalBalance from Ac...	1 row(s) returned

_ **Count**

This query is displaying the number of customers that the customer table has.

```
select count(Cust_ID) as TotalCustomer from Customer;
```



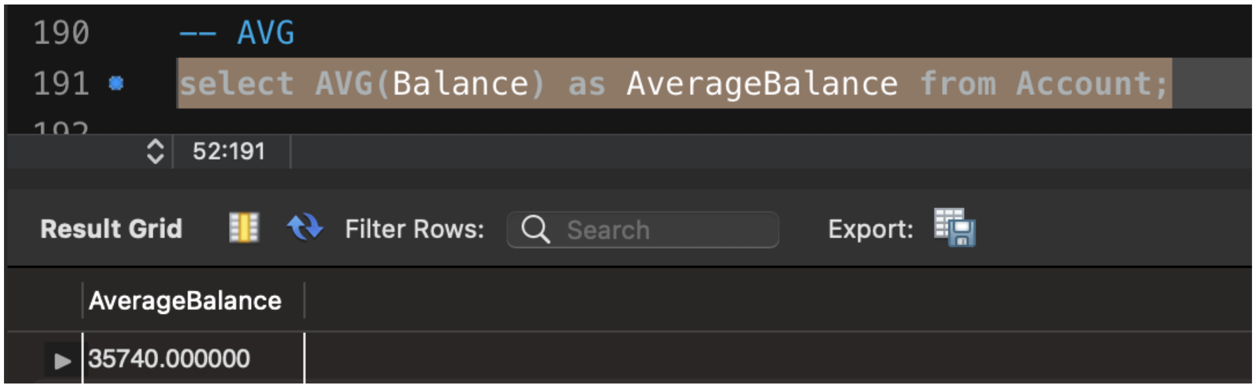
```
-- COUNT
select count(Cust_ID) as TotalCustomer from Customer;
```

```
54:200
Result Grid  Filter Rows: Search  Export:
TotalCustomer
10
```

_ **AVG**

This query is displaying the average balance in the account table.

```
select AVG(Balance) as AverageBalance from Account;
```



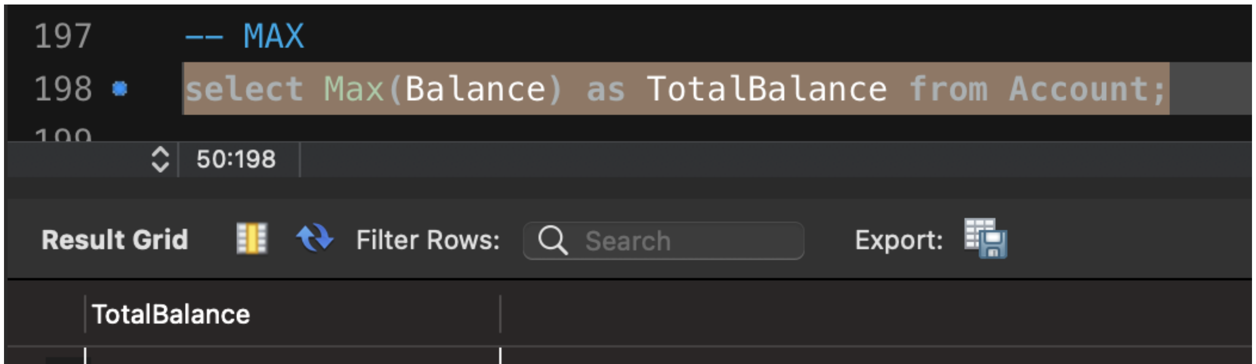
```
-- AVG
select AVG(Balance) as AverageBalance from Account;
```

```
52:191
Result Grid  Filter Rows: Search  Export:
AverageBalance
35740.000000
```

_ **MAX**

This query means the highest balance in the account table.

```
select Max(Balance) as TotalBalance from Account;
```



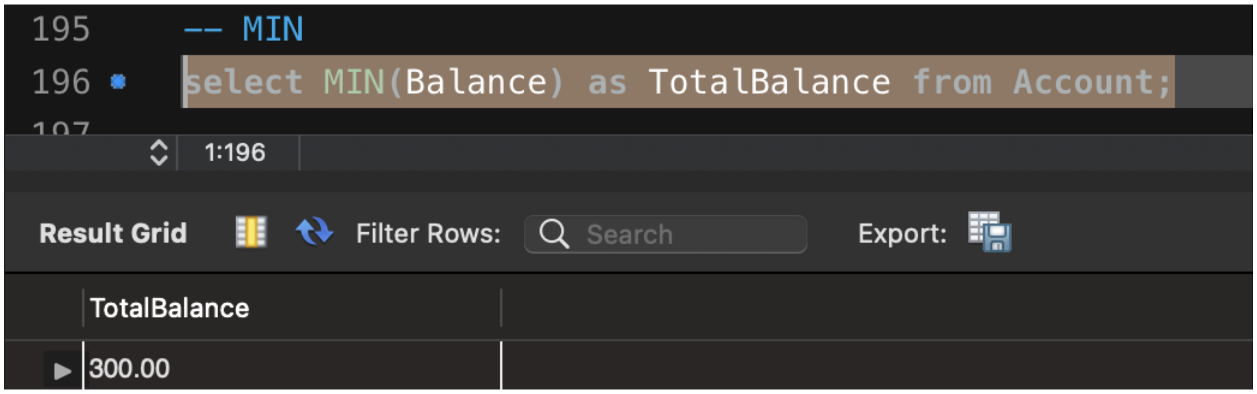
```
-- MAX
select Max(Balance) as TotalBalance from Account;
```

```
50:198
Result Grid  Filter Rows: Search  Export:
TotalBalance
```

_ **MIN**

This query is showing the lowest balance in the account table.

```
select MIN(Balance) as TotalBalance from Account;
```



```
-- MIN
select MIN(Balance) as TotalBalance from Account;
```

1:196

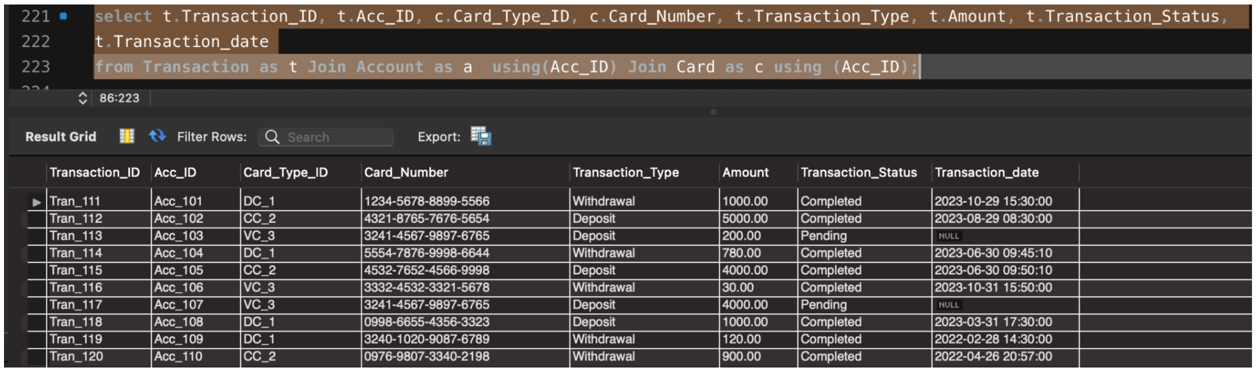
Result Grid Filter Rows: Search Export:

```
TotalBalance
300.00
```

+ **TWO (2) queries involving complicated selects and JOIN from three or more tables.**

. This query displays the Transaction history.

```
        select t.Transaction_ID, t.Acc_ID, c.Card_Type_ID, c.Card_Number, t.Transaction_Type, t.Amount,
t.Transaction_Status, t.Transaction_date
        from Transaction as t Join Account as a using(Acc_ID) Join Card as c using (Acc_ID);
```



```
select t.Transaction_ID, t.Acc_ID, c.Card_Type_ID, c.Card_Number, t.Transaction_Type, t.Amount, t.Transaction_Status,
t.Transaction_date
from Transaction as t Join Account as a  using(Acc_ID) Join Card as c using (Acc_ID);
```

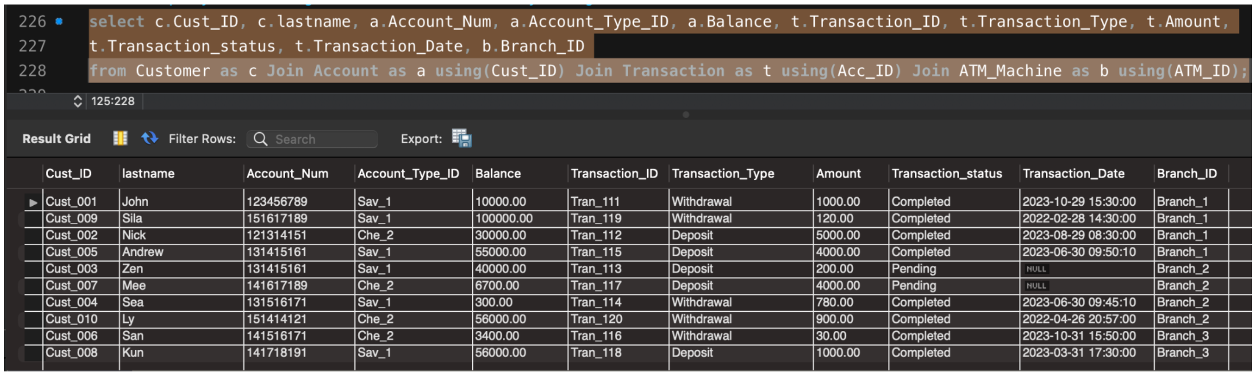
86:223

Result Grid Filter Rows: Search Export:

Transaction_ID	Acc_ID	Card_Type_ID	Card_Number	Transaction_Type	Amount	Transaction_Status	Transaction_date
Tran_111	Acc_101	DC_1	1234-5678-8899-5566	Withdrawal	1000.00	Completed	2023-10-29 15:30:00
Tran_112	Acc_102	CC_2	4321-8765-7676-5654	Deposit	5000.00	Completed	2023-08-29 08:30:00
Tran_113	Acc_103	VC_3	3241-4567-9897-6765	Deposit	200.00	Pending	NULL
Tran_114	Acc_104	DC_1	5554-7876-9998-6644	Withdrawal	780.00	Completed	2023-06-30 09:45:10
Tran_115	Acc_105	CC_2	4532-7652-4566-9998	Deposit	4000.00	Completed	2023-06-30 09:50:10
Tran_116	Acc_106	VC_3	3332-4532-3321-5678	Withdrawal	30.00	Completed	2023-10-31 15:50:00
Tran_117	Acc_107	VC_3	3241-4567-9897-6765	Deposit	4000.00	Pending	NULL
Tran_118	Acc_108	DC_1	0998-6655-4356-3323	Deposit	1000.00	Completed	2023-03-31 17:30:00
Tran_119	Acc_109	DC_1	3240-1020-9087-6789	Withdrawal	120.00	Completed	2022-02-28 14:30:00
Tran_120	Acc_110	CC_2	0976-9807-3340-2198	Withdrawal	900.00	Completed	2022-04-26 20:57:00

. This query is showing the Transaction history along with the customer, account and card information also at which bank branch.

```
select c.Cust_ID, c.lastname, a.Account_Num, a.Account_Type_ID, a.Balance, t.Transaction_ID, t.Transaction_Type,
t.Amount, t.Transaction_status, t.Transaction_Date, b.Branch_ID
from Customer as c Join Account as a using(Cust_ID) Join Transaction as t using(Acc_ID) Join ATM_Machine as b
using(ATM_ID);
```



```
select c.Cust_ID, c.lastname, a.Account_Num, a.Account_Type_ID, a.Balance, t.Transaction_ID, t.Transaction_Type, t.Amount,
t.Transaction_status, t.Transaction_Date, b.Branch_ID
from Customer as c Join Account as a using(Cust_ID) Join Transaction as t using(Acc_ID) Join ATM_Machine as b using(ATM_ID);
```

125:228

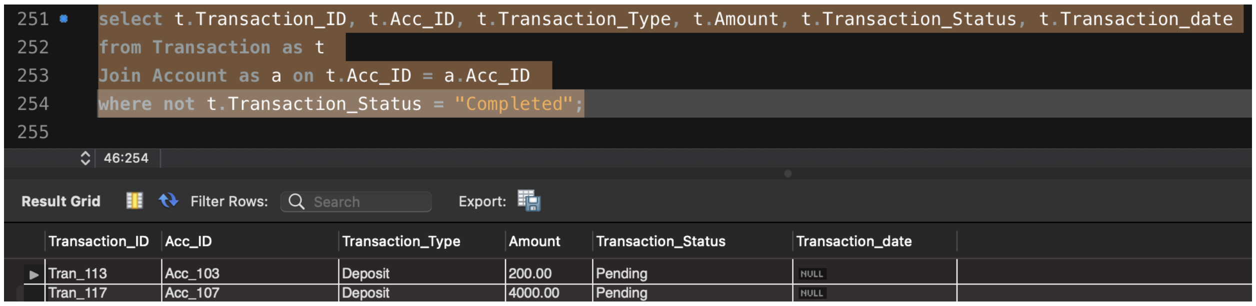
Result Grid Filter Rows: Search Export:

Cust_ID	lastname	Account_Num	Account_Type_ID	Balance	Transaction_ID	Transaction_Type	Amount	Transaction_status	Transaction_Date	Branch_ID
Cust_001	John	123456789	Sav_1	10000.00	Tran_111	Withdrawal	1000.00	Completed	2023-10-29 15:30:00	Branch_1
Cust_009	Sila	151617189	Sav_1	100000.00	Tran_119	Withdrawal	120.00	Completed	2022-02-28 14:30:00	Branch_1
Cust_002	Nick	121314151	Che_2	30000.00	Tran_112	Deposit	5000.00	Completed	2023-08-29 08:30:00	Branch_1
Cust_005	Andrew	131415161	Sav_1	55000.00	Tran_115	Deposit	4000.00	Completed	2023-06-30 09:50:10	Branch_1
Cust_003	Zen	131415161	Sav_1	40000.00	Tran_113	Deposit	200.00	Pending	NULL	Branch_2
Cust_007	Mee	141617189	Che_2	6700.00	Tran_117	Deposit	4000.00	Pending	NULL	Branch_2
Cust_004	Sea	131516171	Sav_1	300.00	Tran_114	Withdrawal	780.00	Completed	2023-06-30 09:45:10	Branch_2
Cust_010	Ly	151414121	Che_2	56000.00	Tran_120	Withdrawal	900.00	Completed	2022-04-26 20:57:00	Branch_2
Cust_006	San	141516171	Che_2	3400.00	Tran_116	Withdrawal	30.00	Completed	2023-10-31 15:50:00	Branch_3
Cust_008	Kun	141718191	Sav_1	56000.00	Tran_118	Deposit	1000.00	Completed	2023-03-31 17:30:00	Branch_3

+ **<u>ONE(1) query involving joins that have a NOT keyword in the relations</u>**

. This query is showing the transaction history which refers to the transaction _status of the account that is pending.

```
select t.Transaction_ID, t.Acc_ID, t.Transaction_Type, t.Amount, t.Transaction_Status, t.Transaction_date
from Transaction as t
Join Account as a on t.Acc_ID = a.Acc_ID
where not t.Transaction_Status = "Completed
```



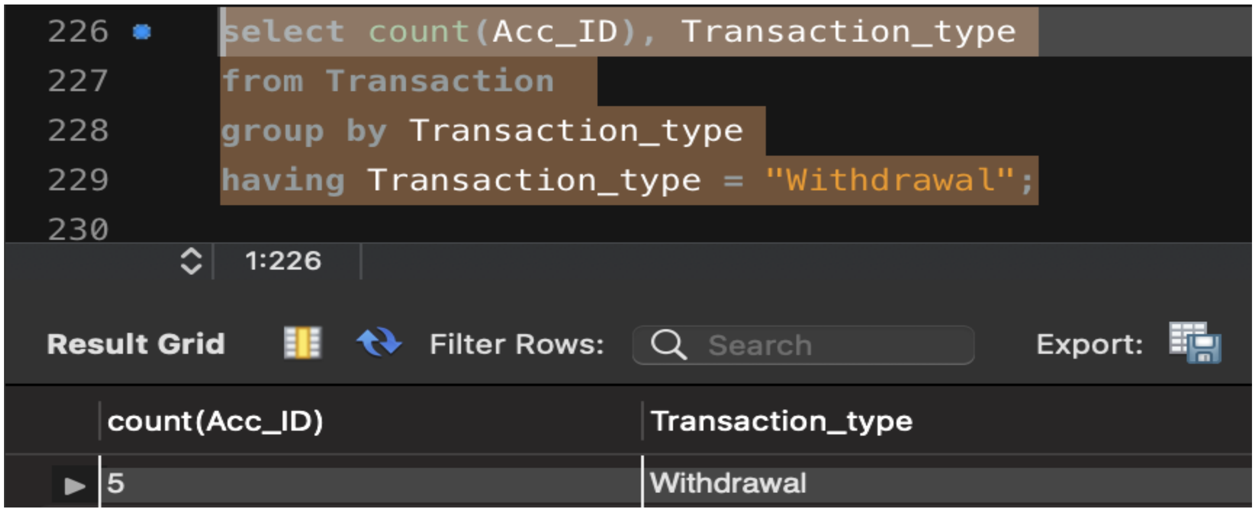
```
select t.Transaction_ID, t.Acc_ID, t.Transaction_Type, t.Amount, t.Transaction_Status, t.Transaction_date
from Transaction as t
Join Account as a on t.Acc_ID = a.Acc_ID
where not t.Transaction_Status = "Completed";

46:254
```

Transaction_ID	Acc_ID	Transaction_Type	Amount	Transaction_Status	Transaction_date
Tran_113	Acc_103	Deposit	200.00	Pending	NULL
Tran_117	Acc_107	Deposit	4000.00	Pending	NULL

+ **<u>TWO (2) queries involving GROUP BY and HAVING functions.</u>**

. This query shows how many accounts perform withdrawal transactions.

```
select count(Acc_ID), Transaction_type
from Transaction
group by Transaction_type
having Transaction_type = "Withdrawal";
```



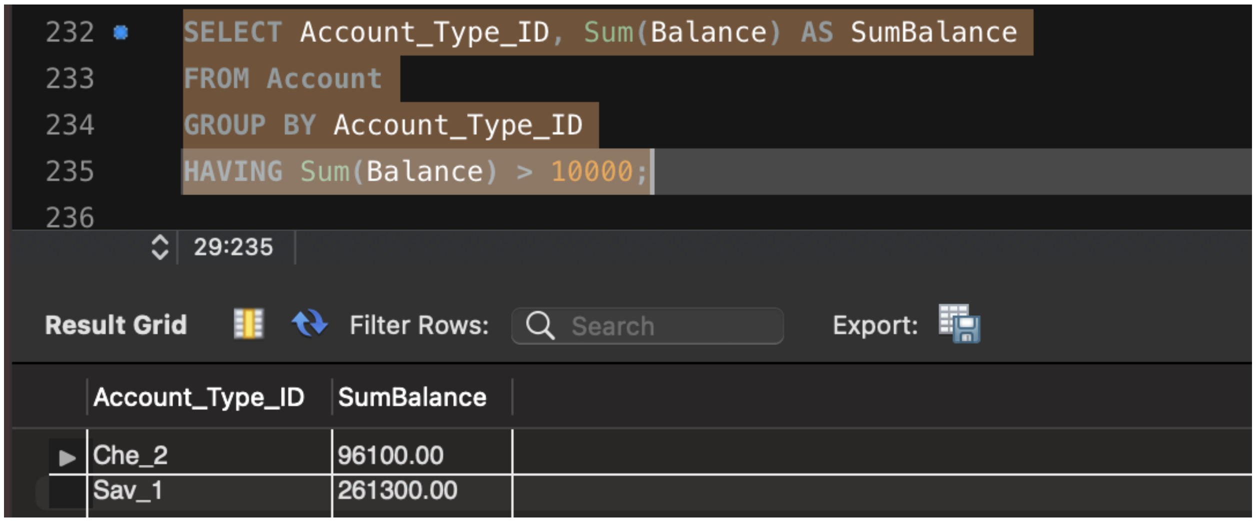
```
select count(Acc_ID), Transaction_type
from Transaction
group by Transaction_type
having Transaction_type = "Withdrawal";

1:226
```

count(Acc_ID)	Transaction_type
5	Withdrawal

. This query is calculating the sum of balance for each account_type_id.

```
SELECT Account_Type_ID, Sum(Balance) AS SumBalance
```

```
FROM Account
GROUP BY Account_Type_ID
HAVING Sum(Balance) > 10000;
```



```
SELECT Account_Type_ID, Sum(Balance) AS SumBalance
FROM Account
GROUP BY Account_Type_ID
HAVING Sum(Balance) > 10000;
```

29:235

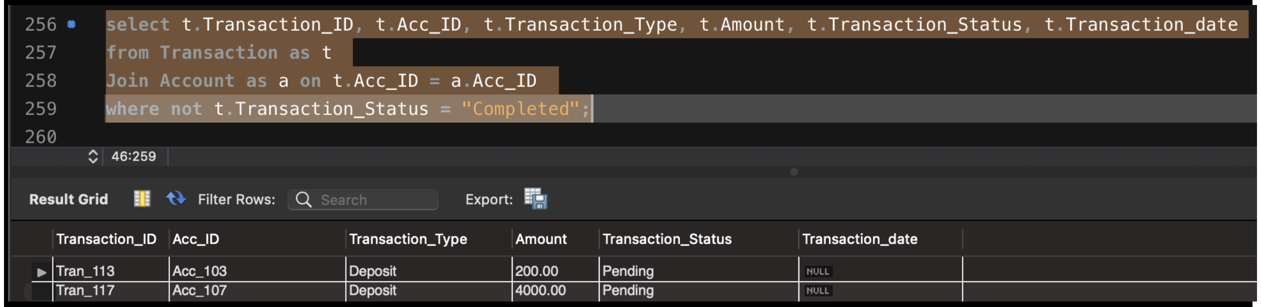
Result Grid | Filter Rows: Search | Export:

Account_Type_ID	SumBalance
Che_2	96100.00
Sav_1	261300.00

+ **<u>ONE(1) query involving joins that have a NOT keyword in the relations</u>**

. This query is showing the transaction history which refers to the transaction _status of the account that is pending.

```
select t.Transaction_ID, t.Acc_ID, t.Transaction_Type, t.Amount, t.Transaction_Status, t.Transaction_date
from Transaction as t
Join Account as a on t.Acc_ID = a.Acc_ID
where not t.Transaction_Status = "Completed
```



```
select t.Transaction_ID, t.Acc_ID, t.Transaction_Type, t.Amount, t.Transaction_Status, t.Transaction_date
from Transaction as t
Join Account as a on t.Acc_ID = a.Acc_ID
where not t.Transaction_Status = "Completed";
```

46:259

Result Grid | Filter Rows: Search | Export:

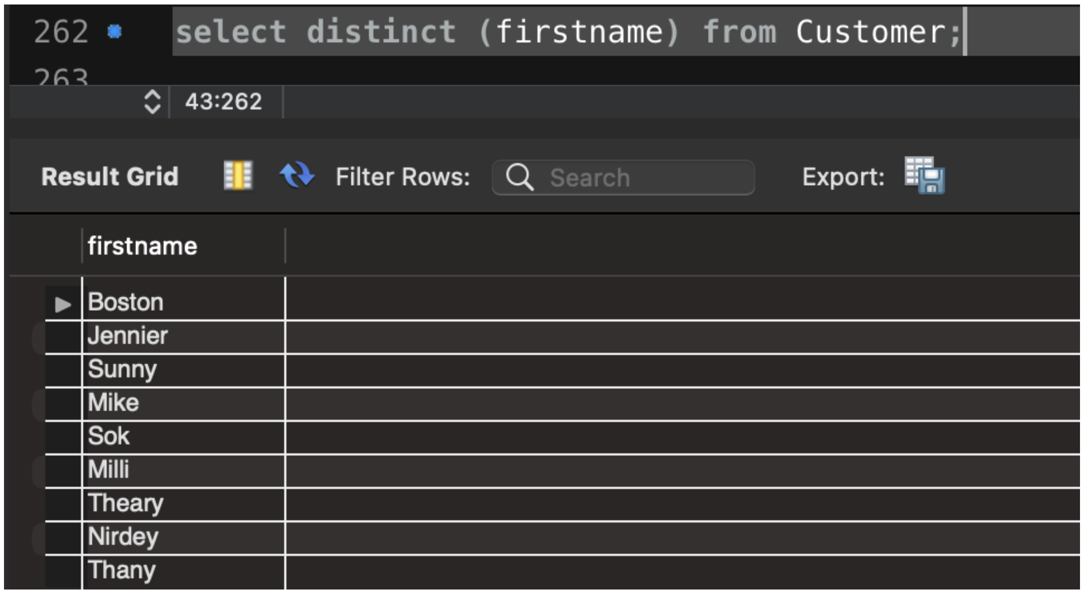
Transaction_ID	Acc_ID	Transaction_Type	Amount	Transaction_Status	Transaction_date
Tran_113	Acc_103	Deposit	200.00	Pending	NULL
Tran_117	Acc_107	Deposit	4000.00	Pending	NULL

```
";
```

+ **TWO(2) queries that require the use of the DISTINCT and ALL keywords**

. This query displays the unique value of the first name from the customer table.

```
select distinct (firstname) from Customer;
```



```
select distinct (firstname) from Customer;
```

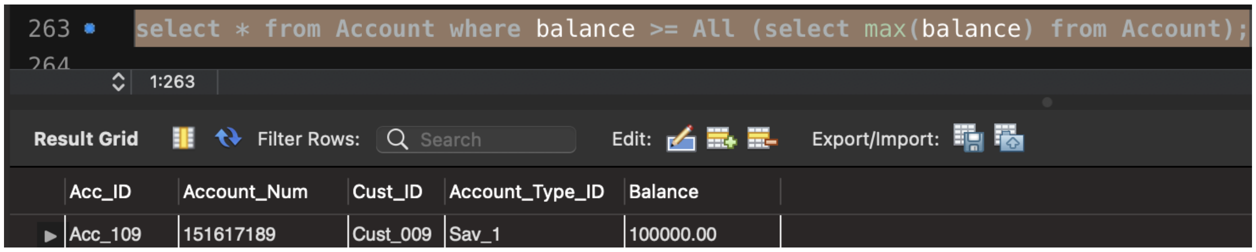
43:262

Result Grid | Filter Rows: Search | Export:

firstname
Boston
Jennier
Sunny
Mike
Sok
Milli
Theary
Nirdey
Thany

. This query is showing the account who has the highest balance among the others.

```
select * from Account where balance >= All (select max(balance) from Account);
```



```
select * from Account where balance >= All (select max(balance) from Account);
```

1:263

Result Grid | Filter Rows: Search | Edit: | Export/Import:

Acc_ID	Account_Num	Cust_ID	Account_Type_ID	Balance
Acc_109	151617189	Cust_009	Sav_1	100000.00

V. Conclusion and Future Enhancement

In conclusion, ATM(Automated Teller Machine) have become an integral part of the banking and financial industry, offering convenience and seft_service banking solutions to customers. These machines are owned and operated by organizations or companies that manage a network of the ATMs, including banks and dependent ATM service providers. A variety of services are offered by ATMs, such as bill payment, fund transfers, cash withdrawals, and balance inquiries. To provide simple access for customers, they are carefully positioned in a variety of venues, including supermarkets, education centers, convenience stores, and travel centers.

+ Some future enhancement to the system include the following:

  . Enhance the ATM system by integrating it with a mobile banking application, which users can be able to start transaction performance via their smartphone.
  . ATM systems could allow users to use cardless for transactions by just scanning the QR code with the ATM.
  . Constantly improve safety measures to safeguard user data and the database.
  . Reader and eye tracking system for user authentication for security purposes as well.

By adding these enhancement, the system can be able to evolve to meet user's expectation, offering more features and improve security.