Final Project In Database Management System

Case Study: Banking Database Management System (CRUD Operations)

School of Arts, Science, and Technology

Submitted by: Maria Victoria N. Esteban

Professor: Mr. Carlito Cunanan

Block & Course: 2.2 BSIT

December 18, 2025

OVERVIEW

This case study presents the design and implementation of a **Banking Database Management System** tailored for a small financial institution. The system is built to manage customer and account data efficiently using **CRUD operations** — Create, Read, Update, and Delete. It emphasizes relational integrity, scalability, and ease of access for banking staff.

TABLE OF CONTENTS

1. Initial Database Design

1.1. Identify the entities and attributes required for the banking system Database.

BANK ENTITY		
Attribute	**Data Type**	**Description**
BankName	VARCHAR	Name of the bank
Code	VARCHAR(PK)	Unique bank code(PK)
Address	VARCHAR	Bank Address

BRANCH ENTITY		
Attribute	**Data Type**	**Description**
BranchName	VARCHAR(100)	commercial name of the branch
BranchID	INTEGER(PK)	unique identification number and the surrogate primary key of the table. PK)
Address	VARCHAR	Branch Address
BranchCode	VARCHAR(100)	internal code used to identify the branch in account numbers.
PhoneNumber	VARCHAR(20)	Branch phone number

CUSTOMER ENTITY		
Attribute	**Data Type**	**Description**
CustomerID	INTEGER (PK)	Surrogate primary key
CustomerType	VARCHAR(20)	Client category (e.g. Regular, Premium)

LastName	VARCHAR(100)	Surname
FirstName	VARCHAR(100)	First name
DateOfBirth	DATE	Birth date
Email	VARCHAR(100)	Email address
PhoneNumber	VARCHAR(20)	Contact number
Address	VARCHAR(100)	Mailing address

EMPLOYEE ENTITY		
Attribute	**Data Type**	**Description**
EmployeeID	INTEGER (PK)	Surrogate primary key
Position	VARCHAR(20)	Job position (e.g. Teller, Manager)
LastName	VARCHAR(100)	Surname
FirstName	VARCHAR(100)	First name
DateOfBirth	DATE	Birth date
Email	VARCHAR(100)	Email address
PhoneNumber	VARCHAR(20)	Contact number
Address	VARCHAR(100)	Mailing address

ACCOUNT ENTITY		
Attribute	**Data Type**	**Description**
AccountID	INTEGER (PK)	Surrogate primary key
AccountType	VARCHAR(20)	Type of account

AccountNumber	VARCHAR(20)	Unique account number
CurrentBalance	DECIMAL	Current balance
DateOpened	DATE	Account opening date
DateClosed	DATE (nullable)	Account closing date
AccountStatus	VARCHAR(20)	Status (active, suspended, etc.)

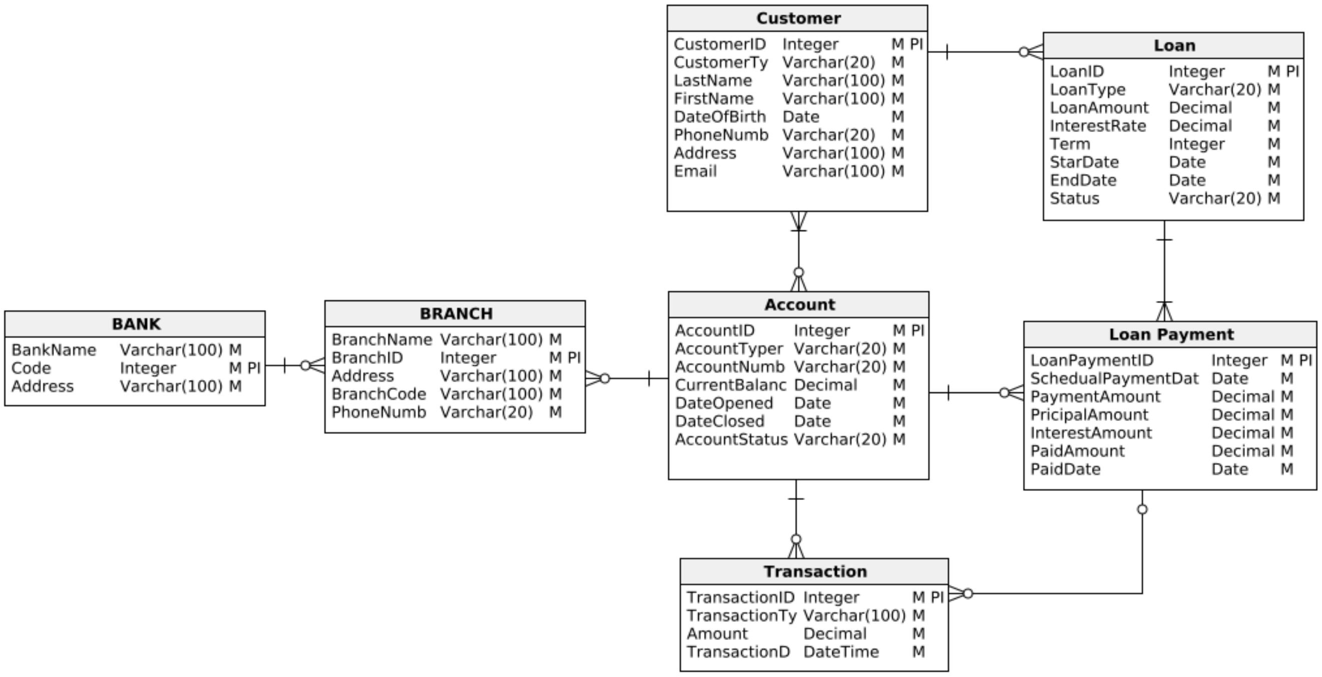
TRANSACTION ENTITY		
Attribute	**Data Type**	**Description**
TransactionID	INTEGER (PK)	Surrogate primary key
TransactionType	VARCHAR(20)	Type of transaction
Amount	DECIMAL	Transaction amount
TransactionDate	DATETIME	Date and time of transaction

LOAN ENTITY		
Attribute	**Data Type**	**Description**
LoanID	INTEGER (PK)	Surrogate primary key
LoanType	VARCHAR(20)	Type of loan
LoanAmount	DECIMAL	Total loan amount
InterestRate	DECIMAL	Annual interest rate
Term	INTEGER	Duration in months
StartDate	DATE	Loan start date
EndDate	DATE	Loan end date
Status	VARCHAR(20)	Loan status

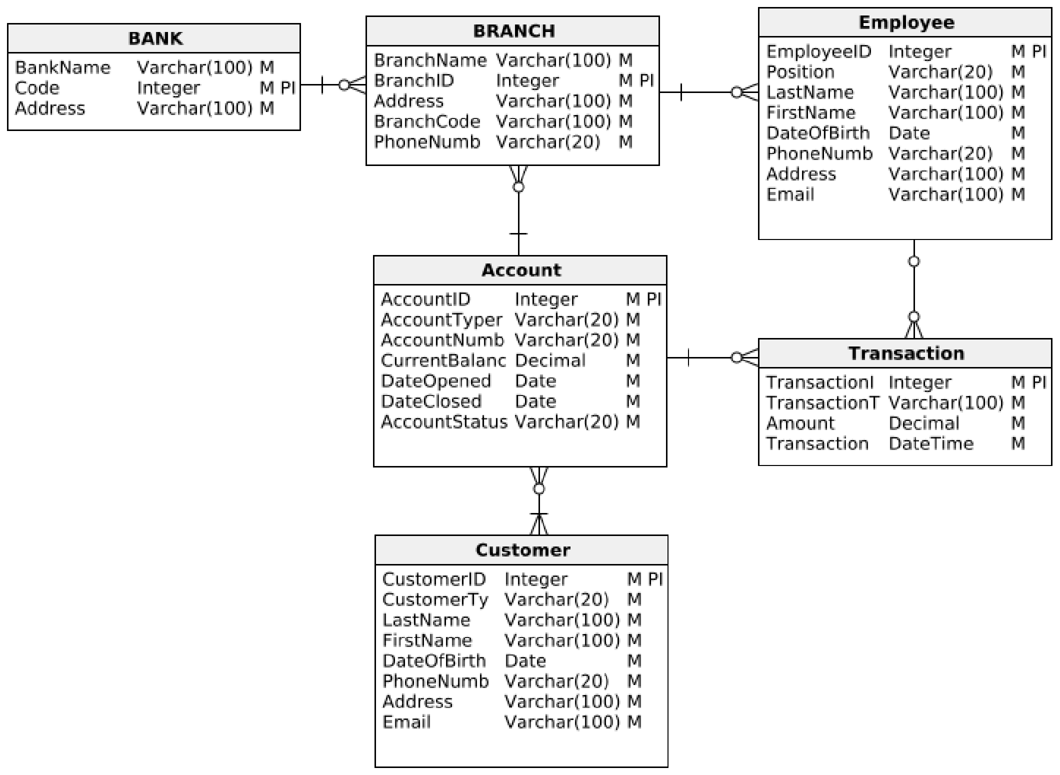
LOAN PAYMENT ENTITY		
Attribute	**Data Type**	**Description**
LoanPaymentID	INTEGER (PK)	Surrogate primary key
ScheduledPaymentDate	DATE	Scheduled payment date
PaymentAmount	DECIMAL	Expected total payment
PrincipalAmount	DECIMAL	Expected principal payment
InterestAmount	DECIMAL	Expected interest payment
PaidAmount	DECIMAL	Actual amount paid
PaidDate	DATE (nullable)	Actual payment date

1.2. Design the table structure for the database, including primary and foreign keys. PHYSICAL, LOGICAL AND CONCEPTUAL ERD

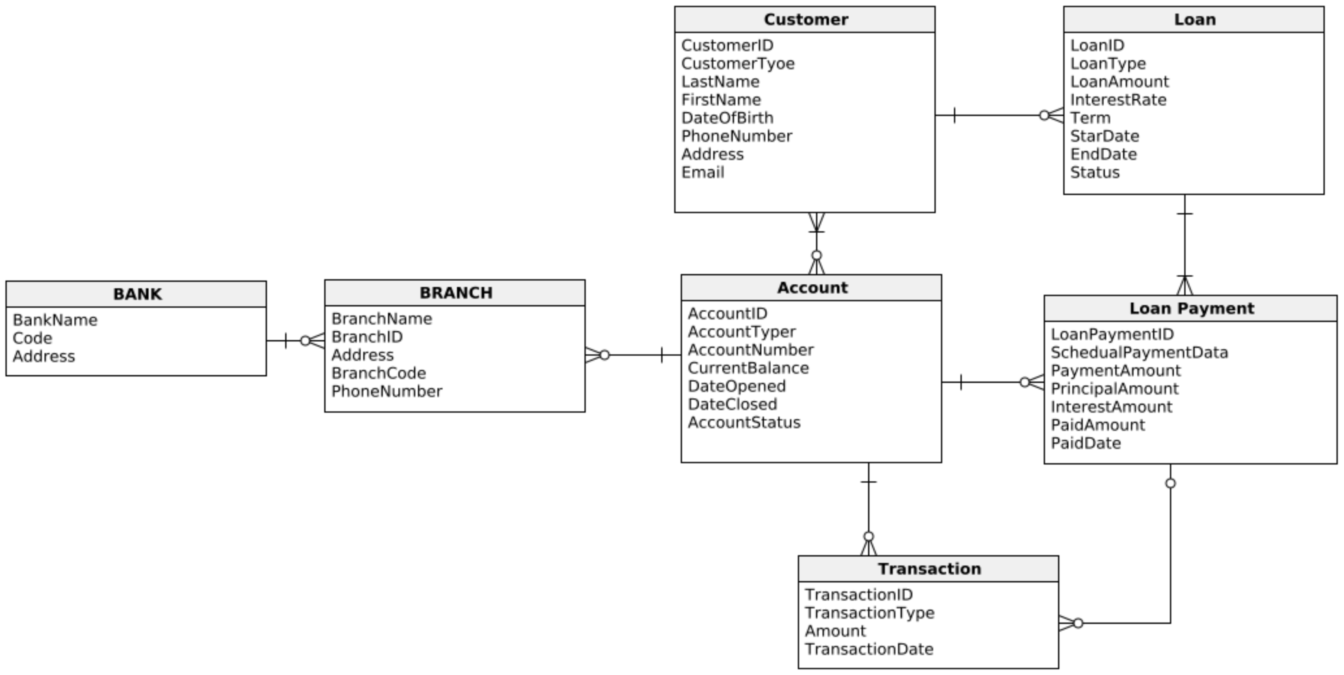
- USER PHYSICAL ERDs



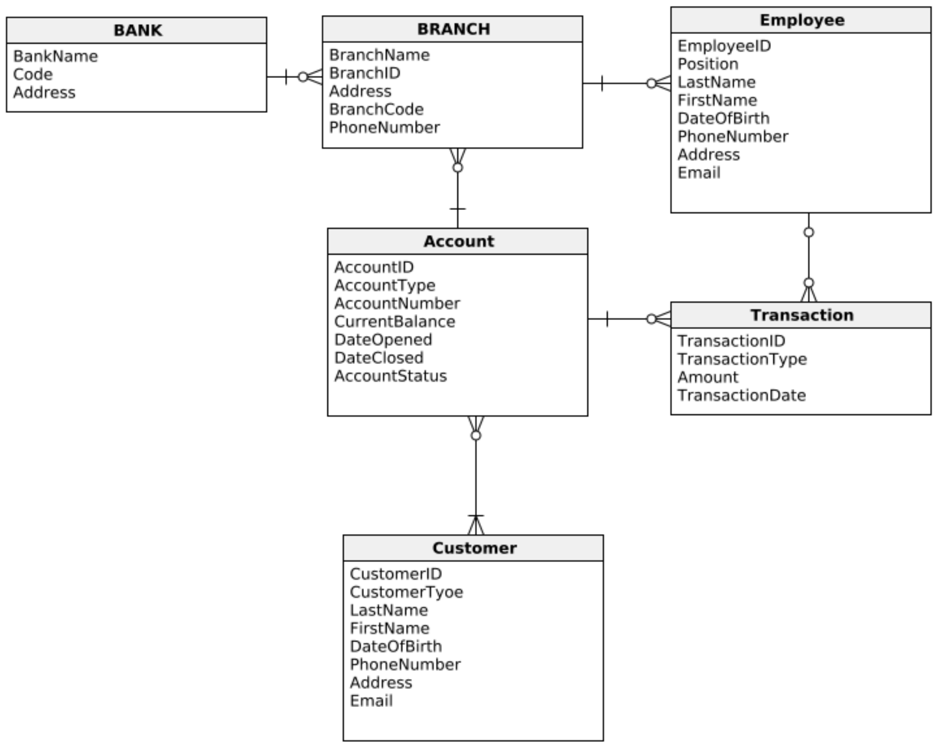
- ADMIN PHYSICAL ERDs



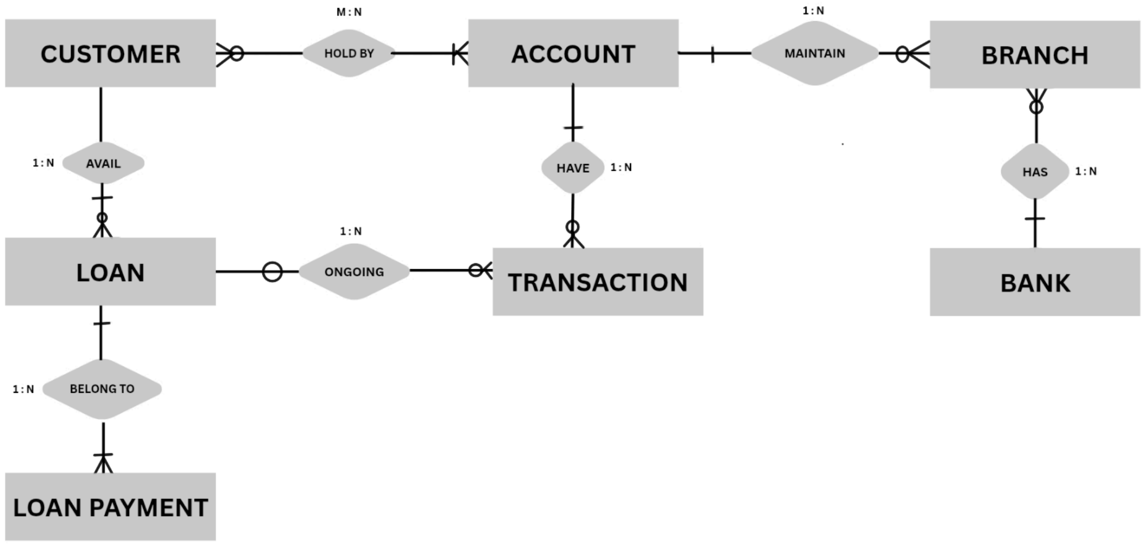
- USER LOGICAL ERDs



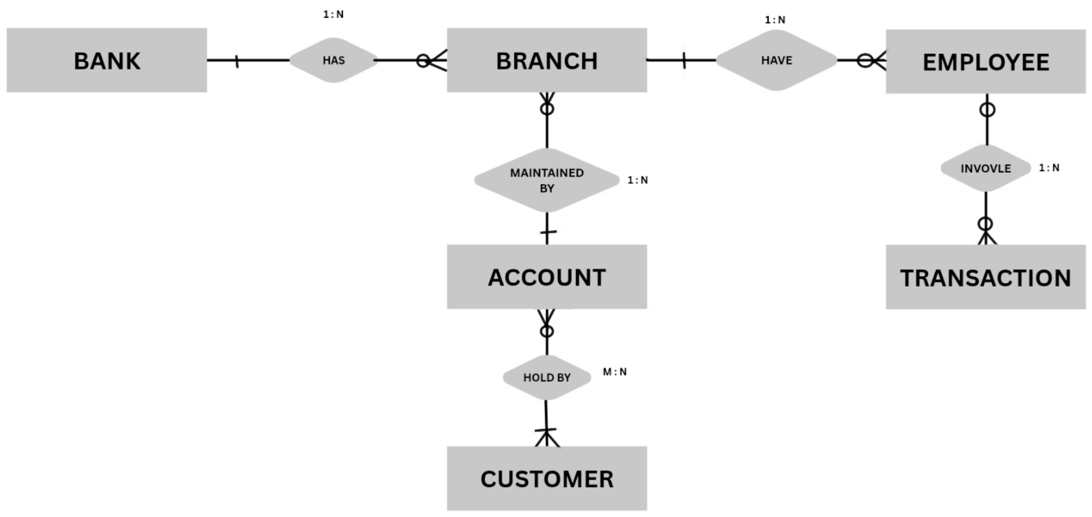
- ADMIN LOGICAL ERDs



- USER CONCEPTUAL ERDs



- ADMIN CONCEPTUAL ERDs



ERD Entities Belonging to the User (Customer)

These represent everything the customer interacts with:

- Customer → the user profile itself
- Account → savings, checking, joint accounts
- Transaction → deposits, withdrawals, transfers
- Loan → loans granted to the customer
- LoanPayment → payments made toward loans
- Branch → indirectly, since accounts and loans are tied to a branch

ERD Entities Belonging to the Admin (Bank Staff)

These represent everything managed by employees and the bank:

- Employee → staff members (tellers, managers, loan officers)
- Branch → physical branches managed by employees
- Account → opened/closed at a branch, overseen by employees
- Transaction → may be performed or approved by employees
- Bank → the top-level entity that owns branches and oversees operations

ERDs User Relationship

- **BANK → BRANCH**
  - One bank has many branches
  - Each branch belongs to one bank
  - Foreign Key: BankCode in BRANCH references Code in BANK
- **BRANCH → ACCOUNT**
  - One branch manages many accounts
  - Each account is opened at one branch
  - Foreign Key: BranchID in ACCOUNT (implied)
- **CUSTOMER → ACCOUNT**
  - One customer can own multiple accounts
  - Each account belongs to one customer
  - Foreign Key: CustomerID in ACCOUNT
- **ACCOUNT → TRANSACTION**
  - One account can have many transactions
  - Each transaction is linked to one account
  - Foreign Key: AccountID in TRANSACTION
- **CUSTOMER → LOAN**
  - One customer can take multiple loans
  - Each loan is associated with one customer
  - Foreign Key: CustomerID in LOAN
- **LOAN → LOAN PAYMENT**
  - One loan has many scheduled payments
  - Each payment is linked to one loan
  - Foreign Key: LoanID in LOAN PAYMENT

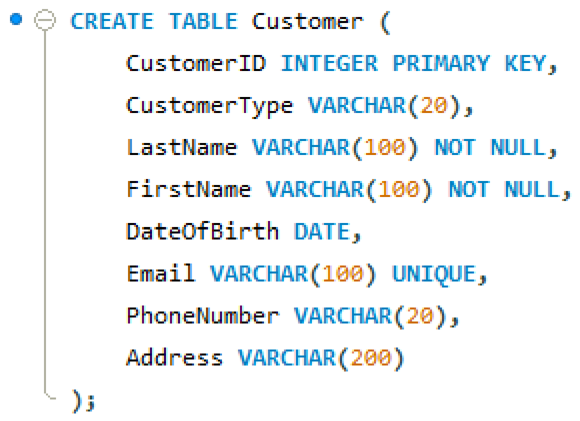
ERDS Admin Relationship

- **BANK → BRANCH**
  - One bank has many branches
  - Each branch belongs to one bank
  - Foreign Key: BankCode in BRANCH references Code in BANK
- **BRANCH → ACCOUNT**
  - One branch manages many accounts
  - Each account is opened at one branch
  - Foreign Key: BranchID in ACCOUNT (implied)
- **CUSTOMER → ACCOUNT**
  - One customer can own multiple accounts

    - Each account belongs to one customer
    - Foreign Key: CustomerID in ACCOUNT
- **ACCOUNT → TRANSACTION**
    - One account can have many transactions
    - Each transaction is linked to one account
    - Foreign Key: AccountID in TRANSACTION
- **EMPLOYEE → TRANSACTION**
    - One employee can process many transactions
    - Each transaction is handled by one employee
    - Foreign Key: EmployeeID in TRANSACTION
- **CUSTOMER → TRANSACTION** *(optional)*
    - If tracked, a customer may initiate many transactions
    - Each transaction may be linked to one customer
    - Foreign Key: CustomerID in TRANSACTION (if implemented)

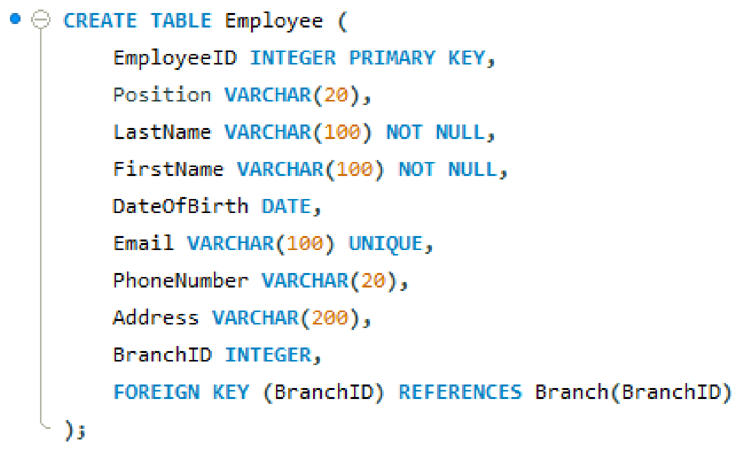
1.3. Write an SQL statement to create a table for storing details.

- **Customer Table**



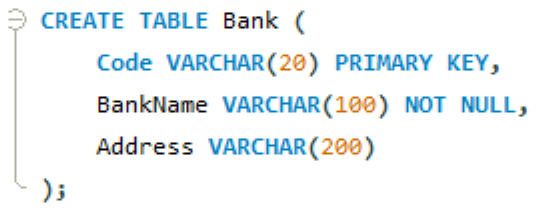
```
CREATE TABLE Customer (
    CustomerID INTEGER PRIMARY KEY,
    CustomerType VARCHAR(20),
    LastName VARCHAR(100) NOT NULL,
    FirstName VARCHAR(100) NOT NULL,
    DateOfBirth DATE,
    Email VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(20),
    Address VARCHAR(200)
);
```

- **Employee Table**



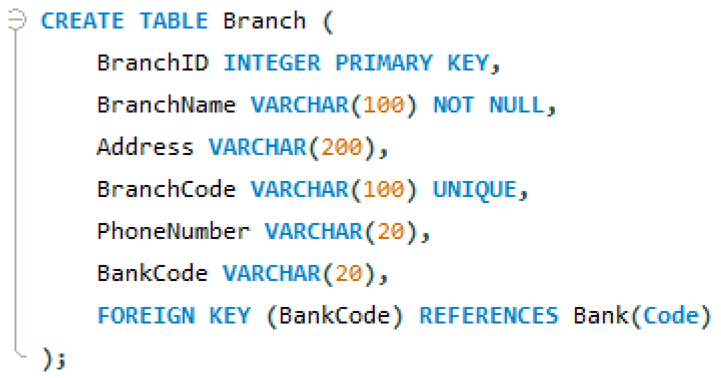
```
CREATE TABLE Employee (
    EmployeeID INTEGER PRIMARY KEY,
    Position VARCHAR(20),
    LastName VARCHAR(100) NOT NULL,
    FirstName VARCHAR(100) NOT NULL,
    DateOfBirth DATE,
    Email VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(20),
    Address VARCHAR(200),
    BranchID INTEGER,
    FOREIGN KEY (BranchID) REFERENCES Branch(BranchID)
);
```

- **Bank Table**



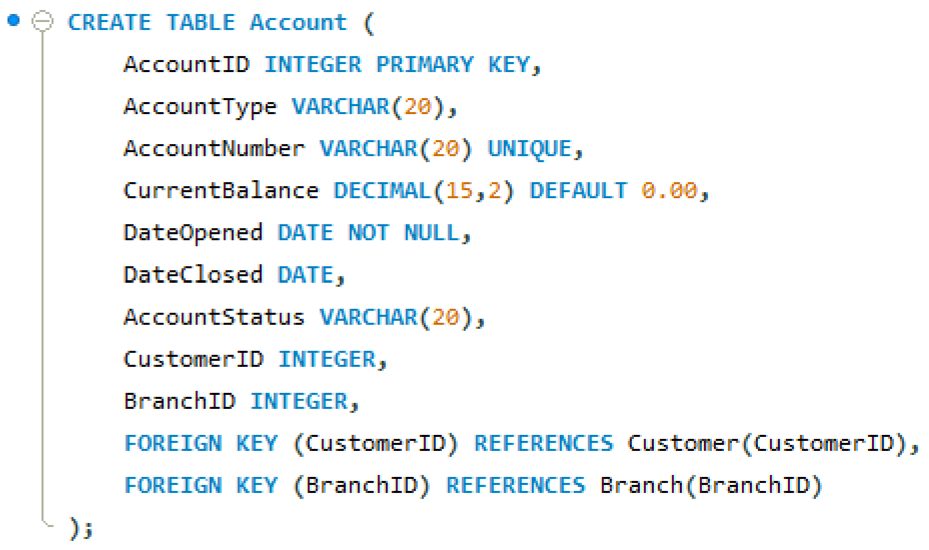
```sql
CREATE TABLE Bank (
    Code VARCHAR(20) PRIMARY KEY,
    BankName VARCHAR(100) NOT NULL,
    Address VARCHAR(200)
);
```

- **Branch Table**



```sql
CREATE TABLE Branch (
    BranchID INTEGER PRIMARY KEY,
    BranchName VARCHAR(100) NOT NULL,
    Address VARCHAR(200),
    BranchCode VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(20),
    BankCode VARCHAR(20),
    FOREIGN KEY (BankCode) REFERENCES Bank(Code)
);
```

- **Account Table**



```sql
CREATE TABLE Account (
    AccountID INTEGER PRIMARY KEY,
    AccountType VARCHAR(20),
    AccountNumber VARCHAR(20) UNIQUE,
    CurrentBalance DECIMAL(15,2) DEFAULT 0.00,
    DateOpened DATE NOT NULL,
    DateClosed DATE,
    AccountStatus VARCHAR(20),
    CustomerID INTEGER,
    BranchID INTEGER,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
    FOREIGN KEY (BranchID) REFERENCES Branch(BranchID)
);
```

- **Transaction Table**

```sql
CREATE TABLE Transaction (
    TransactionID INTEGER PRIMARY KEY,
    TransactionType VARCHAR(20),
    Amount DECIMAL(15,2) NOT NULL,
    TransactionDate DATETIME NOT NULL,
    AccountID INTEGER,
    FOREIGN KEY (AccountID) REFERENCES Account(AccountID)
);
```

- **Loan Table**

```sql
CREATE TABLE Loan (
    LoanID INTEGER PRIMARY KEY,
    LoanType VARCHAR(20),
    LoanAmount DECIMAL(15,2) NOT NULL,
    InterestRate DECIMAL(5,2) NOT NULL,
    Term INTEGER,
    StartDate DATE NOT NULL,
    EndDate DATE,
    Status VARCHAR(20),
    CustomerID INTEGER,
    BranchID INTEGER,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
    FOREIGN KEY (BranchID) REFERENCES Branch(BranchID)
);
```

- **Loan Payment Table**

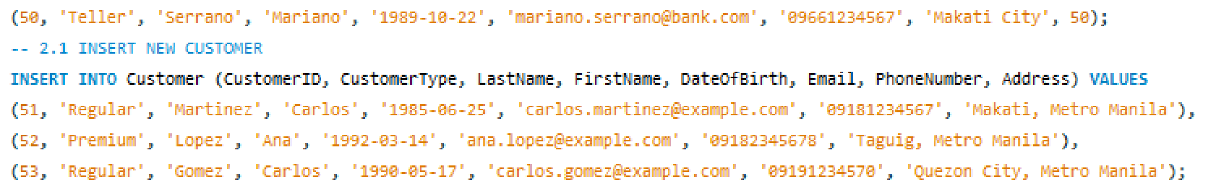
```sql
CREATE TABLE LoanPayment (
    LoanPaymentID INTEGER PRIMARY KEY,
    ScheduledPaymentDate DATE NOT NULL,
    PaymentAmount DECIMAL(15,2) NOT NULL,
    PrincipalAmount DECIMAL(15,2),
    InterestAmount DECIMAL(15,2),
    PaidAmount DECIMAL(15,2),
    PaidDate DATE,
    LoanID INTEGER,
    FOREIGN KEY (LoanID) REFERENCES Loan(LoanID)
);
```

The banking database schema consists of several key tables: **Customer** stores client details like name, address, and contact information; **Employee** records staff data and links each employee to a specific branch; **Bank** represents the overall institution; Branch defines subdivisions of the bank and connects them back to the parent bank; **Account** manages customer accounts including type, balance, and opening date, tied to both customers and branches; **Transaction** logs deposits, withdrawals, and other account activities; **Loan** tracks loans issued to customers with details such as amount, interest rate, and term; **and LoanPayment** records payments made toward loans, including date, amount, and method. Together, these tables form a relational structure that supports customer management, employee assignment, account handling, transaction tracking, and loan servicing.

2. Creating Records

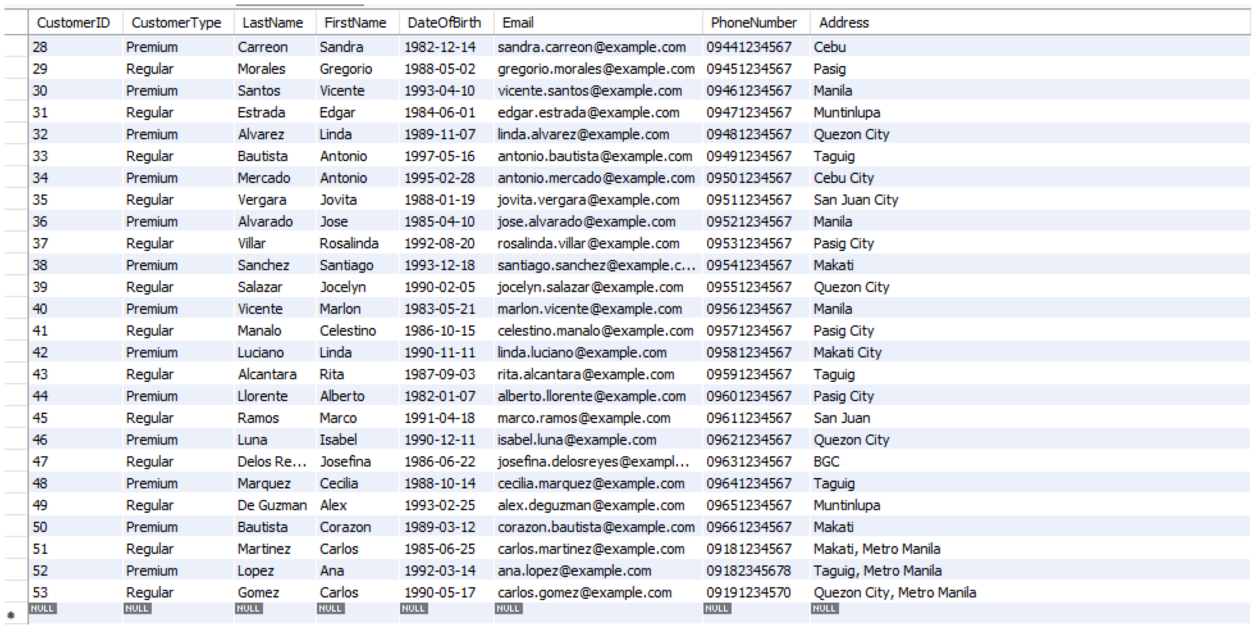
2.1. Write an SQL statement to insert a new customer into the database.

- **SQL CODE**



```sql
(50, 'Teller', 'Serrano', 'Mariano', '1989-10-22', 'mariano.serrano@bank.com', '09661234567', 'Makati City', 50);
-- 2.1 INSERT NEW CUSTOMER
INSERT INTO Customer (CustomerID, CustomerType, LastName, FirstName, DateOfBirth, Email, PhoneNumber, Address) VALUES
(51, 'Regular', 'Martinez', 'Carlos', '1985-06-25', 'carlos.martinez@example.com', '09181234567', 'Makati, Metro Manila'),
(52, 'Premium', 'Lopez', 'Ana', '1992-03-14', 'ana.lopez@example.com', '09182345678', 'Taguig, Metro Manila'),
(53, 'Regular', 'Gomez', 'Carlos', '1990-05-17', 'carlos.gomez@example.com', '09191234570', 'Quezon City, Metro Manila');
```

- **Output**



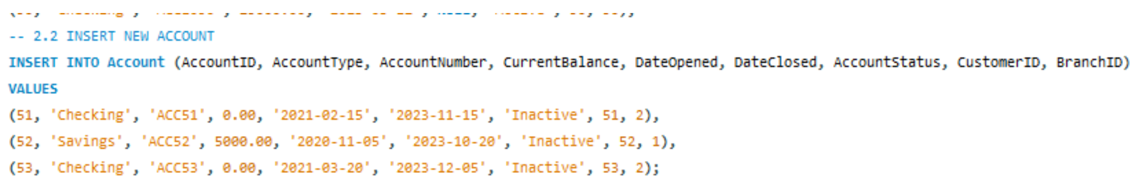
CustomerID	CustomerType	LastName	FirstName	DateOfBirth	Email	PhoneNumber	Address
28	Premium	Carreon	Sandra	1982-12-14	sandra.carreon@example.com	09441234567	Cebu
29	Regular	Morales	Gregorio	1988-05-02	gregorio.morales@example.com	09451234567	Pasig
30	Premium	Santos	Vicente	1993-04-10	vicente.santos@example.com	09461234567	Manila
31	Regular	Estrada	Edgar	1984-06-01	edgar.estrada@example.com	09471234567	Muntinlupa
32	Premium	Alvarez	Linda	1989-11-07	linda.alvarez@example.com	09481234567	Quezon City
33	Regular	Bautista	Antonio	1997-05-16	antonio.bautista@example.com	09491234567	Taguig
34	Premium	Mercado	Antonio	1995-02-28	antonio.mercado@example.com	09501234567	Cebu City
35	Regular	Vergara	Jovita	1988-01-19	jovita.vergara@example.com	09511234567	San Juan City
36	Premium	Alvarado	Jose	1985-04-10	jose.alvarado@example.com	09521234567	Manila
37	Regular	Villar	Rosalinda	1992-08-20	rosalinda.villar@example.com	09531234567	Pasig City
38	Premium	Sanchez	Santiago	1993-12-18	santiago.sanchez@example.c...	09541234567	Makati
39	Regular	Salazar	Jocelyn	1990-02-05	jocelyn.salazar@example.com	09551234567	Quezon City
40	Premium	Vicente	Marlon	1983-05-21	marlon.vicente@example.com	09561234567	Manila
41	Regular	Manalo	Celestino	1986-10-15	celestino.manalo@example.com	09571234567	Pasig City
42	Premium	Luciano	Linda	1990-11-11	linda.luciano@example.com	09581234567	Makati City
43	Regular	Alcantara	Rita	1987-09-03	rita.alcantara@example.com	09591234567	Taguig
44	Premium	Llorente	Alberto	1982-01-07	alberto.llorente@example.com	09601234567	Pasig City
45	Regular	Ramos	Marco	1991-04-18	marco.ramos@example.com	09611234567	San Juan
46	Premium	Luna	Isabel	1990-12-11	isabel.luna@example.com	09621234567	Quezon City
47	Regular	Delos Re...	Josefina	1986-06-22	josefina.delosreyes@exampl...	09631234567	BGC
48	Premium	Marquez	Cecilia	1988-10-14	cecilia.marquez@example.com	09641234567	Taguig
49	Regular	De Guzman	Alex	1993-02-25	alex.deguzman@example.com	09651234567	Muntinlupa
50	Premium	Bautista	Corazon	1989-03-12	corazon.bautista@example.com	09661234567	Makati
51	Regular	Martinez	Carlos	1985-06-25	carlos.martinez@example.com	09181234567	Makati, Metro Manila
52	Premium	Lopez	Ana	1992-03-14	ana.lopez@example.com	09182345678	Taguig, Metro Manila
53	Regular	Gomez	Carlos	1990-05-17	carlos.gomez@example.com	09191234570	Quezon City, Metro Manila
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Explanation:

- CustomerID: This is the unique identifier for the customer. We're assigning it 51, assuming the customer IDs are sequential. Make sure it doesn't clash with existing records.
- CustomerType: 'Regular' or 'Premium', depending on the customer.
- LastName and FirstName: The customer's last name and first name.
- DateOfBirth: The date the customer was born (in the format YYYY-MM-DD).
- Email: The customer's email address.
- PhoneNumber: The customer's contact number.
- Address: The address of the customer.

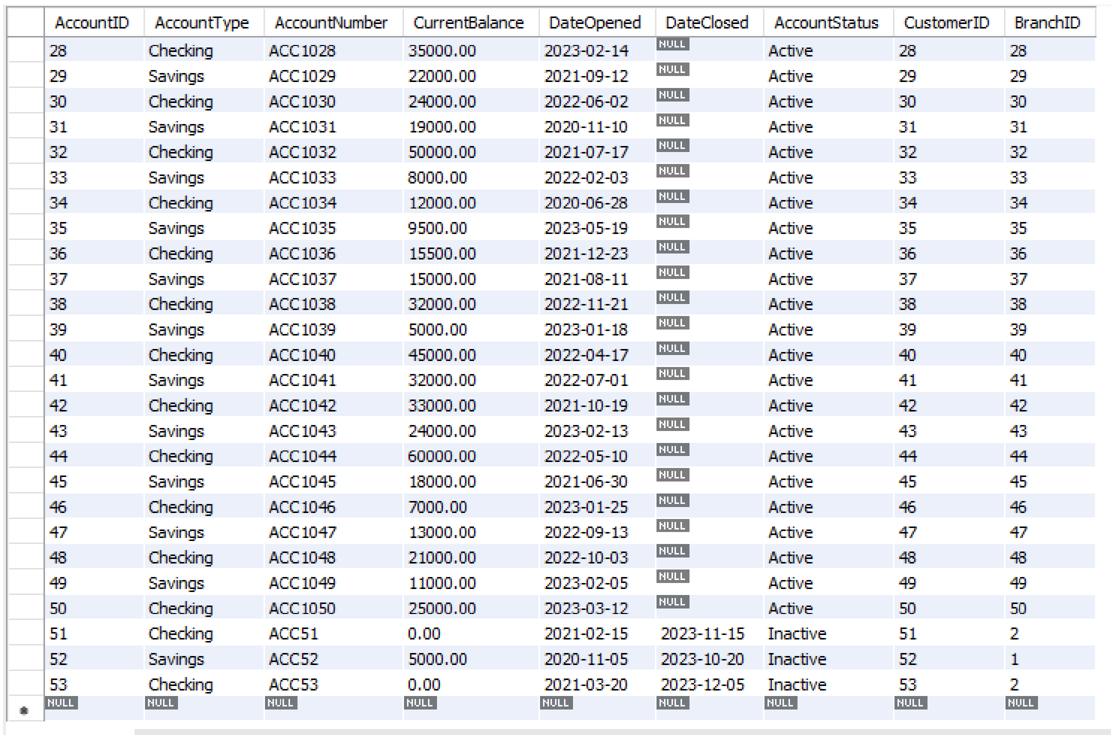
2.2. Write an SQL statement to add a new account associated with a customer.

- **SQL CODE**



```sql
-- 2.2 INSERT NEW ACCOUNT
INSERT INTO Account (AccountID, AccountType, AccountNumber, CurrentBalance, DateOpened, DateClosed, AccountStatus, CustomerID, BranchID)
VALUES
(51, 'Checking', 'ACC51', 0.00, '2021-02-15', '2023-11-15', 'Inactive', 51, 2),
(52, 'Savings', 'ACC52', 5000.00, '2020-11-05', '2023-10-20', 'Inactive', 52, 1),
(53, 'Checking', 'ACC53', 0.00, '2021-03-20', '2023-12-05', 'Inactive', 53, 2);
```

- **Output**



	AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
	28	Checking	ACC1028	35000.00	2023-02-14	NULL	Active	28	28
	29	Savings	ACC1029	22000.00	2021-09-12	NULL	Active	29	29
	30	Checking	ACC1030	24000.00	2022-06-02	NULL	Active	30	30
	31	Savings	ACC1031	19000.00	2020-11-10	NULL	Active	31	31
	32	Checking	ACC1032	50000.00	2021-07-17	NULL	Active	32	32
	33	Savings	ACC1033	8000.00	2022-02-03	NULL	Active	33	33
	34	Checking	ACC1034	12000.00	2020-06-28	NULL	Active	34	34
	35	Savings	ACC1035	9500.00	2023-05-19	NULL	Active	35	35
	36	Checking	ACC1036	15500.00	2021-12-23	NULL	Active	36	36
	37	Savings	ACC1037	15000.00	2021-08-11	NULL	Active	37	37
	38	Checking	ACC1038	32000.00	2022-11-21	NULL	Active	38	38
	39	Savings	ACC1039	5000.00	2023-01-18	NULL	Active	39	39
	40	Checking	ACC1040	45000.00	2022-04-17	NULL	Active	40	40
	41	Savings	ACC1041	32000.00	2022-07-01	NULL	Active	41	41
	42	Checking	ACC1042	33000.00	2021-10-19	NULL	Active	42	42
	43	Savings	ACC1043	24000.00	2023-02-13	NULL	Active	43	43
	44	Checking	ACC1044	60000.00	2022-05-10	NULL	Active	44	44
	45	Savings	ACC1045	18000.00	2021-06-30	NULL	Active	45	45
	46	Checking	ACC1046	7000.00	2023-01-25	NULL	Active	46	46
	47	Savings	ACC1047	13000.00	2022-09-13	NULL	Active	47	47
	48	Checking	ACC1048	21000.00	2022-10-03	NULL	Active	48	48
	49	Savings	ACC1049	11000.00	2023-02-05	NULL	Active	49	49
	50	Checking	ACC1050	25000.00	2023-03-12	NULL	Active	50	50
	51	Checking	ACC51	0.00	2021-02-15	2023-11-15	Inactive	51	2
	52	Savings	ACC52	5000.00	2020-11-05	2023-10-20	Inactive	52	1
	53	Checking	ACC53	0.00	2021-03-20	2023-12-05	Inactive	53	2
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Explanation:

- AccountID: The unique identifier for the account. In this case, it's 101, which should be unique.
- AccountType: The type of account. In this example, we are creating a 'Savings' account. You can also use 'Checking', 'Business', etc.
- AccountNumber: A unique account number for this new account. Here, it's 'ACC1051'.
- CurrentBalance: The starting balance of the account. Here, we're initializing it with 5000.00.
- DateOpened: The date the account was opened. We are setting it to today's date '2025-12-14'.
- DateClosed: The account is active, so this is NULL.
- AccountStatus: The status of the account. We are setting it as 'Active' because the account is open.
- CustomerID: This refers to the CustomerID from the Customer table. In this case, the newly added customer's CustomerID is 51.
- BranchID: This refers to the ID of the branch where the account is opened. Here, we assume it's 1 (you can adjust this depending on your branch structure).

3. Reading Data

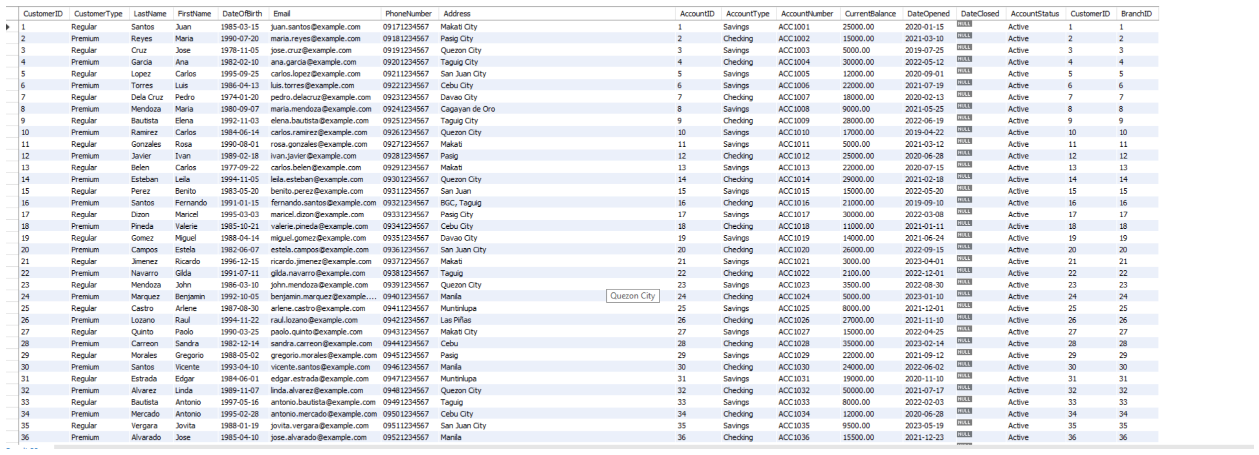
3.1. WHERE Clause

a. Get all customers who have an active account

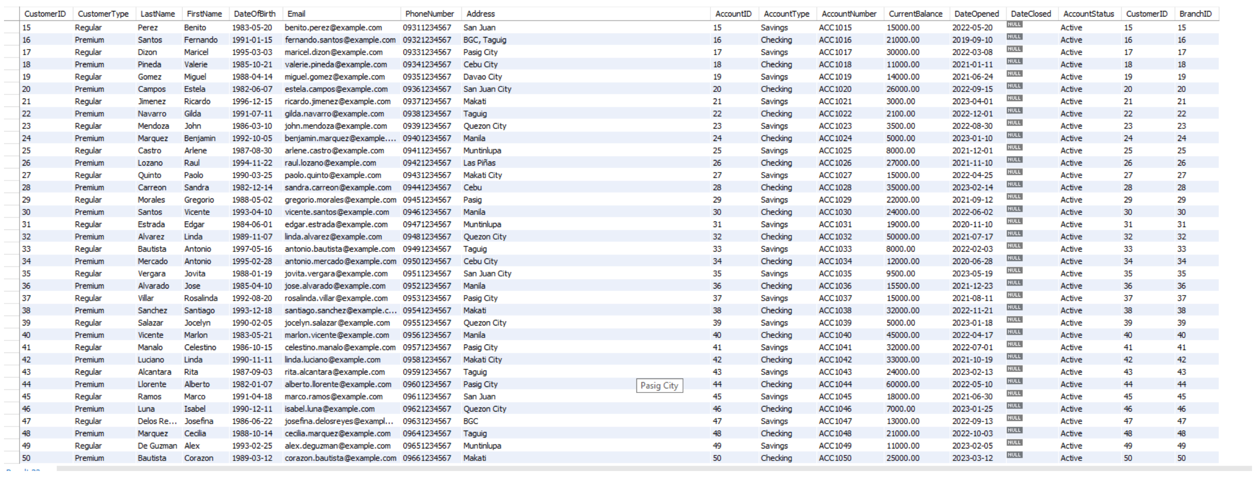
SQL CODE

```sql
SELECT *
FROM Customer c
JOIN Account a ON c.CustomerID = a.CustomerID
WHERE a.AccountStatus = 'Active';
```

OUTPUT



CustomerID	CustomerType	LastName	FirstName	DateOfBirth	Email	PhoneNumber	Address	AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
1	Regular	Santos	Juan	1985-03-15	juan.santos@example.com	09171234567	Makati City	1	Savings	ACC1001	25000.00	2020-01-15	NULL	Active	1	1
2	Premium	Reyes	Maria	1990-07-20	maria.reyes@example.com	09181234567	Pasig City	2	Checking	ACC1002	15000.00	2021-03-10	NULL	Active	2	2
3	Regular	Cruz	Jose	1978-11-05	jose.cruz@example.com	09191234567	Quezon City	3	Savings	ACC1003	5000.00	2019-07-25	NULL	Active	3	3
4	Premium	Garcia	Ana	1982-02-10	ana.garcia@example.com	09201234567	Taguig City	4	Checking	ACC1004	30000.00	2022-05-12	NULL	Active	4	4
5	Regular	Lopez	Carlos	1995-09-25	carlos.lopez@example.com	09211234567	San Juan City	5	Savings	ACC1005	12000.00	2020-09-01	NULL	Active	5	5
6	Premium	Torres	Luis	1986-04-13	luis.torres@example.com	09221234567	Cebu City	6	Savings	ACC1006	22000.00	2021-07-19	NULL	Active	6	6
7	Regular	Dela Cruz	Pedro	1974-01-20	pedro.delacruz@example.com	09231234567	Davao City	7	Checking	ACC1007	18000.00	2020-02-13	NULL	Active	7	7
8	Premium	Mendoza	Maria	1980-09-07	maria.mendoza@example.com	09241234567	Cagayan de Oro	8	Savings	ACC1008	9000.00	2021-05-25	NULL	Active	8	8
9	Regular	Bautista	Elena	1992-11-03	elena.bautista@example.com	09251234567	Taguig City	9	Checking	ACC1009	28000.00	2022-06-19	NULL	Active	9	9
10	Premium	Ramirez	Carlos	1984-06-14	carlos.ramirez@example.com	09261234567	Quezon City	10	Savings	ACC1010	17000.00	2019-04-22	NULL	Active	10	10
11	Regular	Gonzales	Rosa	1990-08-01	rosa.gonzales@example.com	09271234567	Makati	11	Savings	ACC1011	5000.00	2021-03-12	NULL	Active	11	11
12	Premium	Javier	Ivan	1989-02-18	ivan.javier@example.com	09281234567	Pasig	12	Checking	ACC1012	25000.00	2020-06-28	NULL	Active	12	12
13	Regular	Belen	Carlos	1977-09-22	carlos.belen@example.com	09291234567	Makati	13	Savings	ACC1013	22000.00	2020-07-15	NULL	Active	13	13
14	Premium	Esteban	Leila	1994-11-05	leila.esteban@example.com	09301234567	Quezon City	14	Checking	ACC1014	29000.00	2021-02-18	NULL	Active	14	14
15	Regular	Perez	Benito	1983-05-20	benito.perez@example.com	09311234567	San Juan	15	Savings	ACC1015	15000.00	2022-05-20	NULL	Active	15	15
16	Premium	Santos	Fernando	1991-01-15	fernando.santos@example.com	09321234567	BGC, Taguig	16	Checking	ACC1016	21000.00	2019-09-10	NULL	Active	16	16
17	Regular	Dizon	Maricel	1995-03-03	maricel.dizon@example.com	09331234567	Pasig City	17	Savings	ACC1017	30000.00	2022-03-08	NULL	Active	17	17
18	Premium	Pineda	Valerie	1985-10-21	valerie.pineda@example.com	09341234567	Cebu City	18	Checking	ACC1018	11000.00	2021-01-11	NULL	Active	18	18
19	Regular	Gomez	Miguel	1988-04-14	miguel.gomez@example.com	09351234567	Davao City	19	Savings	ACC1019	14000.00	2021-06-24	NULL	Active	19	19
20	Premium	Campos	Estela	1982-06-07	estela.campos@example.com	09361234567	San Juan City	20	Checking	ACC1020	26000.00	2022-09-15	NULL	Active	20	20
21	Regular	Jimenez	Ricardo	1996-12-15	ricardo.jimenez@example.com	09371234567	Makati	21	Savings	ACC1021	3000.00	2023-04-01	NULL	Active	21	21
22	Premium	Navarro	Gilda	1991-07-11	gilda.navarro@example.com	09381234567	Taguig	22	Checking	ACC1022	2100.00	2022-12-01	NULL	Active	22	22
23	Regular	Mendoza	John	1986-03-10	john.mendoza@example.com	09391234567	Quezon City	23	Savings	ACC1023	3500.00	2022-08-30	NULL	Active	23	23
24	Premium	Marquez	Benjamin	1992-10-05	benjamin.marquez@example....	09401234567	Manila	24	Checking	ACC1024	5000.00	2023-01-10	NULL	Active	24	24
25	Regular	Castro	Arlene	1987-08-30	arlene.castro@example.com	09411234567	Muntinlupa	25	Savings	ACC1025	8000.00	2021-12-01	NULL	Active	25	25
26	Premium	Lozano	Raul	1994-11-22	raul.lozano@example.com	09421234567	Las Piñas	26	Checking	ACC1026	27000.00	2021-11-10	NULL	Active	26	26
27	Regular	Quinto	Paolo	1990-03-25	paolo.quinto@example.com	09431234567	Makati City	27	Savings	ACC1027	15000.00	2022-04-25	NULL	Active	27	27
28	Premium	Carreon	Sandra	1982-12-14	sandra.carreon@example.com	09441234567	Cebu	28	Checking	ACC1028	35000.00	2023-02-14	NULL	Active	28	28
29	Regular	Morales	Gregorio	1988-05-02	gregorio.morales@example.com	09451234567	Pasig	29	Savings	ACC1029	22000.00	2021-09-12	NULL	Active	29	29
30	Premium	Santos	Vicente	1993-04-10	vicente.santos@example.com	09461234567	Manila	30	Checking	ACC1030	24000.00	2022-06-02	NULL	Active	30	30
31	Regular	Estrada	Edgar	1984-06-01	edgar.estrada@example.com	09471234567	Muntinlupa	31	Savings	ACC1031	19000.00	2020-11-10	NULL	Active	31	31
32	Premium	Alvarez	Linda	1989-11-07	linda.alvarez@example.com	09481234567	Quezon City	32	Checking	ACC1032	50000.00	2021-07-17	NULL	Active	32	32
33	Regular	Bautista	Antonio	1997-05-16	antonio.bautista@example.com	09491234567	Taguig	33	Savings	ACC1033	8000.00	2022-02-03	NULL	Active	33	33
34	Premium	Mercado	Antonio	1995-02-28	antonio.mercado@example.com	09501234567	Cebu City	34	Checking	ACC1034	12000.00	2020-06-28	NULL	Active	34	34
35	Regular	Vergara	Jovita	1988-01-19	jovita.vergara@example.com	09511234567	San Juan City	35	Savings	ACC1035	9500.00	2023-05-19	NULL	Active	35	35
36	Premium	Alvarado	Jose	1985-04-10	jose.alvarado@example.com	09521234567	Manila	36	Checking	ACC1036	15500.00	2021-12-23	NULL	Active	36	36



CustomerID	CustomerType	LastName	FirstName	DateOfBirth	Email	PhoneNumber	Address	AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
15	Regular	Perez	Benito	1983-05-20	benito.perez@example.com	09311234567	San Juan	15	Savings	ACC1015	15000.00	2022-05-20	NULL	Active	15	15
16	Premium	Santos	Fernando	1991-01-15	fernando.santos@example.com	09321234567	BGC, Taguig	16	Checking	ACC1016	21000.00	2019-09-10	NULL	Active	16	16
17	Regular	Dizon	Maricel	1995-03-03	maricel.dizon@example.com	09331234567	Pasig City	17	Savings	ACC1017	30000.00	2022-03-08	NULL	Active	17	17
18	Premium	Pineda	Valerie	1985-10-21	valerie.pineda@example.com	09341234567	Cebu City	18	Checking	ACC1018	11000.00	2021-01-11	NULL	Active	18	18
19	Regular	Gomez	Miguel	1988-04-14	miguel.gomez@example.com	09351234567	Davao City	19	Savings	ACC1019	14000.00	2021-06-24	NULL	Active	19	19
20	Premium	Campos	Estela	1982-06-07	estela.campos@example.com	09361234567	San Juan City	20	Checking	ACC1020	26000.00	2022-09-15	NULL	Active	20	20
21	Regular	Jimenez	Ricardo	1996-12-15	ricardo.jimenez@example.com	09371234567	Makati	21	Savings	ACC1021	3000.00	2023-04-01	NULL	Active	21	21
22	Premium	Navarro	Gilda	1991-07-11	gilda.navarro@example.com	09381234567	Taguig	22	Checking	ACC1022	2100.00	2022-12-01	NULL	Active	22	22
23	Regular	Mendoza	John	1986-03-10	john.mendoza@example.com	09391234567	Quezon City	23	Savings	ACC1023	3500.00	2022-08-30	NULL	Active	23	23
24	Premium	Marquez	Benjamin	1992-10-05	benjamin.marquez@example....	09401234567	Manila	24	Checking	ACC1024	5000.00	2023-01-10	NULL	Active	24	24
25	Regular	Castro	Arlene	1987-08-30	arlene.castro@example.com	09411234567	Muntinlupa	25	Savings	ACC1025	8000.00	2021-12-01	NULL	Active	25	25
26	Premium	Lozano	Raul	1994-11-22	raul.lozano@example.com	09421234567	Las Piñas	26	Checking	ACC1026	27000.00	2021-11-10	NULL	Active	26	26
27	Regular	Quinto	Paolo	1990-03-25	paolo.quinto@example.com	09431234567	Makati City	27	Savings	ACC1027	15000.00	2022-04-25	NULL	Active	27	27
28	Premium	Carreon	Sandra	1982-12-14	sandra.carreon@example.com	09441234567	Cebu	28	Checking	ACC1028	35000.00	2023-02-14	NULL	Active	28	28
29	Regular	Morales	Gregorio	1988-05-02	gregorio.morales@example.com	09451234567	Pasig	29	Savings	ACC1029	22000.00	2021-09-12	NULL	Active	29	29
30	Premium	Santos	Vicente	1993-04-10	vicente.santos@example.com	09461234567	Manila	30	Checking	ACC1030	24000.00	2022-06-02	NULL	Active	30	30
31	Regular	Estrada	Edgar	1984-06-01	edgar.estrada@example.com	09471234567	Muntinlupa	31	Savings	ACC1031	19000.00	2020-11-10	NULL	Active	31	31
32	Premium	Alvarez	Linda	1989-11-07	linda.alvarez@example.com	09481234567	Quezon City	32	Checking	ACC1032	50000.00	2021-07-17	NULL	Active	32	32
33	Regular	Bautista	Antonio	1997-05-16	antonio.bautista@example.com	09491234567	Taguig	33	Savings	ACC1033	8000.00	2022-02-03	NULL	Active	33	33
34	Premium	Mercado	Antonio	1995-02-28	antonio.mercado@example.com	09501234567	Cebu City	34	Checking	ACC1034	12000.00	2020-06-28	NULL	Active	34	34
35	Regular	Vergara	Jovita	1988-01-19	jovita.vergara@example.com	09511234567	San Juan City	35	Savings	ACC1035	9500.00	2023-05-19	NULL	Active	35	35
36	Premium	Alvarado	Jose	1985-04-10	jose.alvarado@example.com	09521234567	Manila	36	Checking	ACC1036	15500.00	2021-12-23	NULL	Active	36	36
37	Regular	Villar	Rosalinda	1992-08-20	rosalinda.villar@example.com	09531234567	Pasig City	37	Savings	ACC1037	15000.00	2021-08-11	NULL	Active	37	37
38	Premium	Sanchez	Santiago	1993-12-18	santiago.sanchez@example.c...	09541234567	Makati	38	Checking	ACC1038	32000.00	2022-11-21	NULL	Active	38	38
39	Regular	Salazar	Jocelyn	1990-02-05	jocelyn.salazar@example.com	09551234567	Quezon City	39	Savings	ACC1039	5000.00	2023-01-18	NULL	Active	39	39
40	Premium	Vicente	Marlon	1983-05-21	marlon.vicente@example.com	09561234567	Manila	40	Checking	ACC1040	45000.00	2022-04-17	NULL	Active	40	40
41	Regular	Manalo	Celestino	1986-10-15	celestino.manalo@example.com	09571234567	Pasig City	41	Savings	ACC1041	32000.00	2022-07-01	NULL	Active	41	41
42	Premium	Luciano	Linda	1990-11-11	linda.luciano@example.com	09581234567	Makati City	42	Checking	ACC1042	33000.00	2021-10-19	NULL	Active	42	42
43	Regular	Alcantara	Rita	1987-09-03	rita.alcantara@example.com	09591234567	Taguig	43	Savings	ACC1043	24000.00	2023-02-13	NULL	Active	43	43
44	Premium	Llorente	Alberto	1982-01-07	alberto.llorente@example.com	09601234567	Pasig City	44	Checking	ACC1044	60000.00	2022-05-10	NULL	Active	44	44
45	Regular	Ramos	Marco	1991-04-18	marco.ramos@example.com	09611234567	San Juan	45	Savings	ACC1045	18000.00	2021-06-30	NULL	Active	45	45
46	Premium	Luna	Isabel	1990-12-11	isabel.luna@example.com	09621234567	Quezon City	46	Checking	ACC1046	7000.00	2023-01-25	NULL	Active	46	46
47	Regular	Delos Re...	Josefina	1986-06-22	josefina.delosreyes@exampl...	09631234567	BGC	47	Savings	ACC1047	13000.00	2022-09-13	NULL	Active	47	47
48	Premium	Marquez	Cecilia	1988-10-14	cecilia.marquez@example.com	09641234567	Taguig	48	Checking	ACC1048	21000.00	2022-10-03	NULL	Active	48	48
49	Regular	De Guzman	Alex	1993-02-25	alex.deguzman@example.com	09651234567	Muntinlupa	49	Savings	ACC1049	11000.00	2023-02-05	NULL	Active	49	49
50	Premium	Bautista	Corazon	1989-03-12	corazon.bautista@example.com	09661234567	Makati	50	Checking	ACC1050	25000.00	2023-03-12	NULL	Active	50	50

i. JOIN links the Customer table with the Account table using CustomerID.WHERE a.AccountStatus = 'Active'

ii. filters only those accounts that are active.

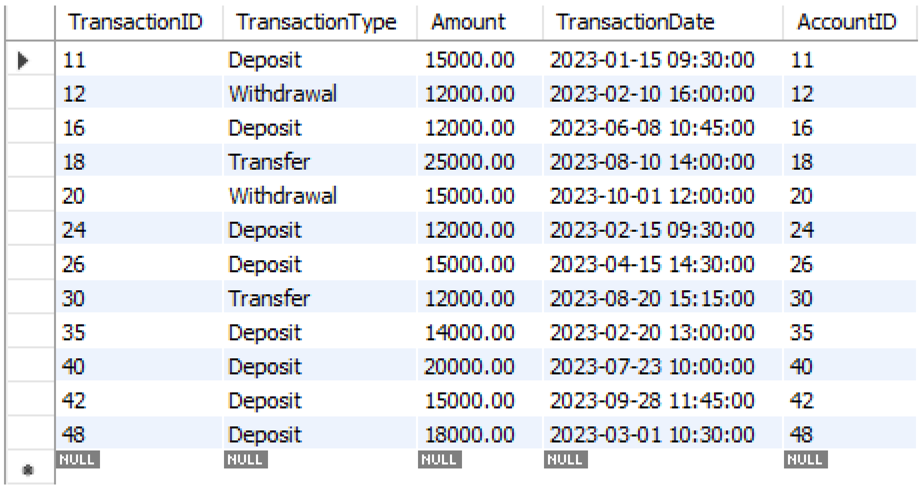
iii. SELECT clause retrieves customer details (you can add more fields if needed).

b. Find transactions with an amount greater than ₱50,000

SQL CODE

```
SELECT *
FROM Transaction
WHERE Amount > 10000;
```

OUTPUT



	TransactionID	TransactionType	Amount	TransactionDate	AccountID
▶	11	Deposit	15000.00	2023-01-15 09:30:00	11
	12	Withdrawal	12000.00	2023-02-10 16:00:00	12
	16	Deposit	12000.00	2023-06-08 10:45:00	16
	18	Transfer	25000.00	2023-08-10 14:00:00	18
	20	Withdrawal	15000.00	2023-10-01 12:00:00	20
	24	Deposit	12000.00	2023-02-15 09:30:00	24
	26	Deposit	15000.00	2023-04-15 14:30:00	26
	30	Transfer	12000.00	2023-08-20 15:15:00	30
	35	Deposit	14000.00	2023-02-20 13:00:00	35
	40	Deposit	20000.00	2023-07-23 10:00:00	40
	42	Deposit	15000.00	2023-09-28 11:45:00	42
	48	Deposit	18000.00	2023-03-01 10:30:00	48
*	NULL	NULL	NULL	NULL	NULL

i. **Explanation:** This query returns all transactions where the transaction amount is greater than ₱50,000.

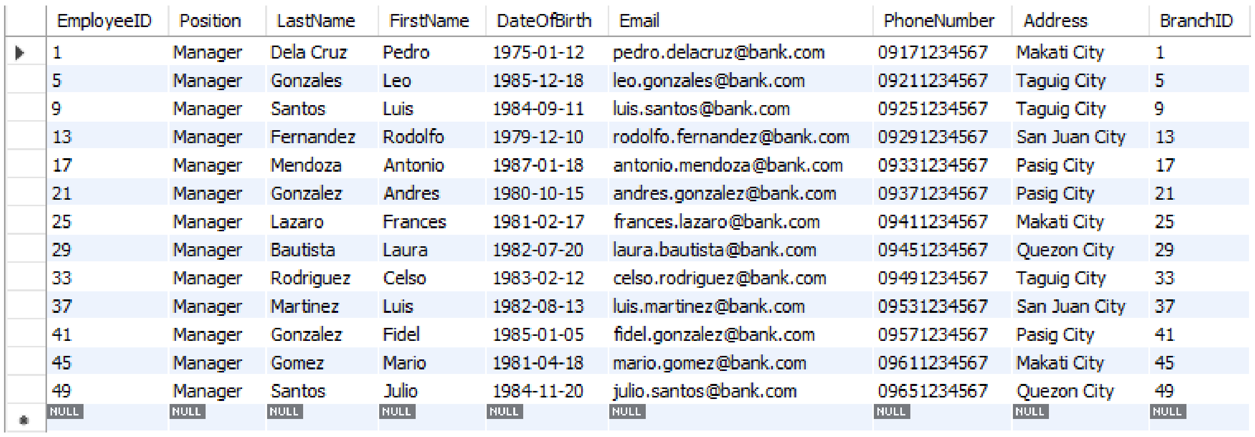
ii. **WHERE Amount > 50000** filters the results to transactions with amounts greater than ₱50,000.

c. Find Employee whose position is 'Manager'

SQL CODE

```
SELECT *
FROM Employee
WHERE Position = 'Manager';
```

OUTPUT



EmployeeID	Position	LastName	FirstName	DateOfBirth	Email	PhoneNumber	Address	BranchID
1	Manager	Dela Cruz	Pedro	1975-01-12	pedro.delacruz@bank.com	09171234567	Makati City	1
5	Manager	Gonzales	Leo	1985-12-18	leo.gonzales@bank.com	09211234567	Taguig City	5
9	Manager	Santos	Luis	1984-09-11	luis.santos@bank.com	09251234567	Taguig City	9
13	Manager	Fernandez	Rodolfo	1979-12-10	rodolfo.fernandez@bank.com	09291234567	San Juan City	13
17	Manager	Mendoza	Antonio	1987-01-18	antonio.mendoza@bank.com	09331234567	Pasig City	17
21	Manager	Gonzalez	Andres	1980-10-15	andres.gonzalez@bank.com	09371234567	Pasig City	21
25	Manager	Lazaro	Frances	1981-02-17	frances.lazaro@bank.com	09411234567	Makati City	25
29	Manager	Bautista	Laura	1982-07-20	laura.bautista@bank.com	09451234567	Quezon City	29
33	Manager	Rodriguez	Celso	1983-02-12	celso.rodriguez@bank.com	09491234567	Taguig City	33
37	Manager	Martinez	Luis	1982-08-13	luis.martinez@bank.com	09531234567	San Juan City	37
41	Manager	Gonzalez	Fidel	1985-01-05	fidel.gonzalez@bank.com	09571234567	Pasig City	41
45	Manager	Gomez	Mario	1981-04-18	mario.gomez@bank.com	09611234567	Makati City	45
49	Manager	Santos	Julio	1984-11-20	julio.santos@bank.com	09651234567	Quezon City	49
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

i. **Explanation:** This query finds all employees whose Position is 'Manager'.

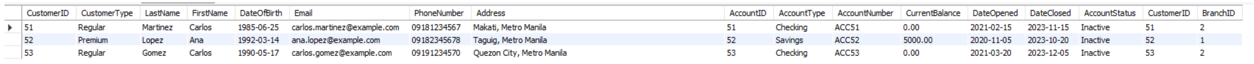
ii. **WHERE Position = 'Manager'** filters for employees who have a Manager position.

d. Get all customers who have an Inactive account

SQL CODE

```sql
SELECT *
FROM Customer c
JOIN Account a ON c.CustomerID = a.CustomerID
WHERE a.AccountStatus = 'Inactive';
```

OUTPUT



CustomerID	CustomerType	LastName	FirstName	DateOfBirth	Email	PhoneNumber	Address	AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
51	Regular	Martinez	Carlos	1985-06-25	carlos.martinez@example.com	09181234567	Makati, Metro Manila	51	Checking	ACC51	0.00	2021-02-15	2023-11-15	Inactive	51	2
52	Premium	Lopez	Ana	1992-03-14	ana.lopez@example.com	09182345678	Taguig, Metro Manila	52	Savings	ACC52	5000.00	2020-11-05	2023-10-20	Inactive	52	1
53	Regular	Gomez	Carlos	1990-05-17	carlos.gomez@example.com	09191234570	Quezon City, Metro Manila	53	Checking	ACC53	0.00	2021-03-20	2023-12-05	Inactive	53	2

i. **Explanation:** This query retrieves all customers who have an account with the 'Inactive' status.

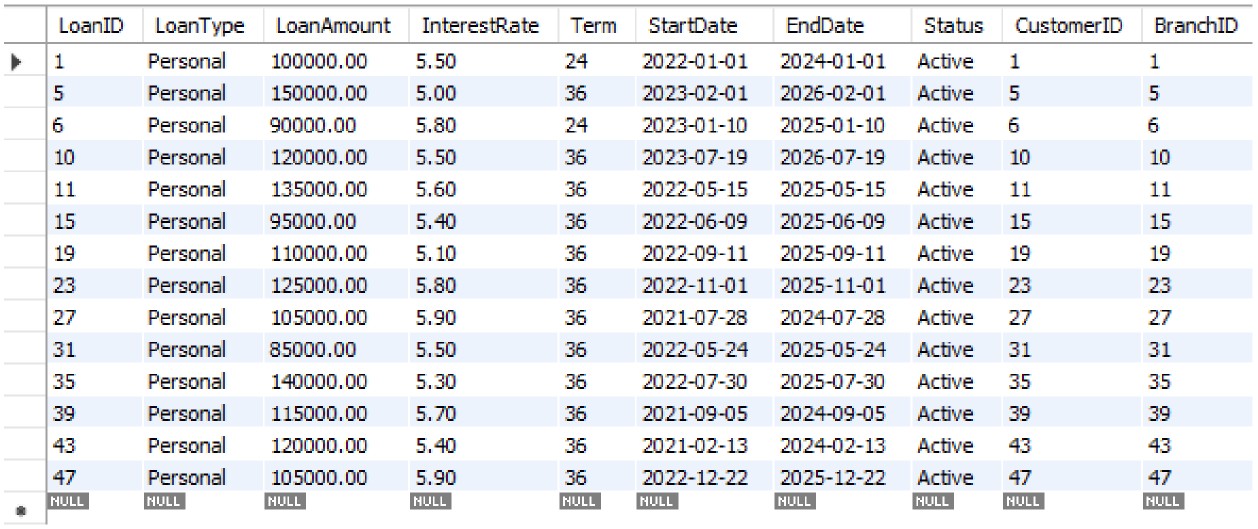
ii. **WHERE a.AccountStatus = 'Inactive'** filters the accounts to only include Inactive accounts.

e. Get all loans with Loan type ‘Personal’

SQL CODE

```sql
SELECT *
FROM Loan
WHERE LoanType = 'Personal';
```

OUTPUT



LoanID	LoanType	LoanAmount	InterestRate	Term	StartDate	EndDate	Status	CustomerID	BranchID
1	Personal	100000.00	5.50	24	2022-01-01	2024-01-01	Active	1	1
5	Personal	150000.00	5.00	36	2023-02-01	2026-02-01	Active	5	5
6	Personal	90000.00	5.80	24	2023-01-10	2025-01-10	Active	6	6
10	Personal	120000.00	5.50	36	2023-07-19	2026-07-19	Active	10	10
11	Personal	135000.00	5.60	36	2022-05-15	2025-05-15	Active	11	11
15	Personal	95000.00	5.40	36	2022-06-09	2025-06-09	Active	15	15
19	Personal	110000.00	5.10	36	2022-09-11	2025-09-11	Active	19	19
23	Personal	125000.00	5.80	36	2022-11-01	2025-11-01	Active	23	23
27	Personal	105000.00	5.90	36	2021-07-28	2024-07-28	Active	27	27
31	Personal	85000.00	5.50	36	2022-05-24	2025-05-24	Active	31	31
35	Personal	140000.00	5.30	36	2022-07-30	2025-07-30	Active	35	35
39	Personal	115000.00	5.70	36	2021-09-05	2024-09-05	Active	39	39
43	Personal	120000.00	5.40	36	2021-02-13	2024-02-13	Active	43	43
47	Personal	105000.00	5.90	36	2022-12-22	2025-12-22	Active	47	47
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

i. **Explanation:** This query returns all loans of type 'Personal'.

ii. **WHERE LoanType = 'Personal'** filters the records to loans that are of Personal type.

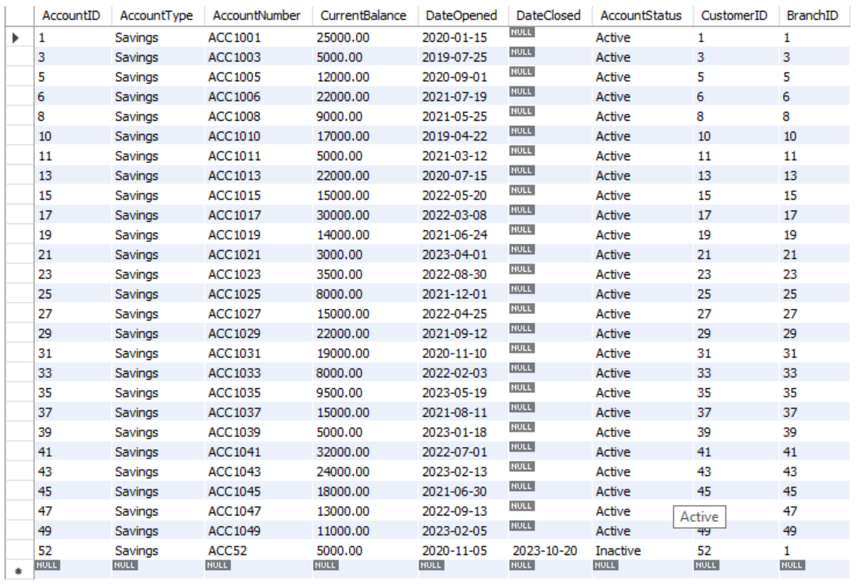
3.2. IN Operator

a. Find all customers with an account type of either Savings or Checking

SQL CODE

```sql
SELECT *
FROM Account
WHERE AccountType IN ('Savings');
```

OUTPUT



AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
1	Savings	ACC1001	25000.00	2020-01-15	NULL	Active	1	1
3	Savings	ACC1003	5000.00	2019-07-25	NULL	Active	3	3
5	Savings	ACC1005	12000.00	2020-09-01	NULL	Active	5	5
6	Savings	ACC1006	22000.00	2021-07-19	NULL	Active	6	6
8	Savings	ACC1008	9000.00	2021-05-25	NULL	Active	8	8
10	Savings	ACC1010	17000.00	2019-04-22	NULL	Active	10	10
11	Savings	ACC1011	5000.00	2021-03-12	NULL	Active	11	11
13	Savings	ACC1013	22000.00	2020-07-15	NULL	Active	13	13
15	Savings	ACC1015	15000.00	2022-05-20	NULL	Active	15	15
17	Savings	ACC1017	30000.00	2022-03-08	NULL	Active	17	17
19	Savings	ACC1019	14000.00	2021-06-24	NULL	Active	19	19
21	Savings	ACC1021	3000.00	2023-04-01	NULL	Active	21	21
23	Savings	ACC1023	3500.00	2022-08-30	NULL	Active	23	23
25	Savings	ACC1025	8000.00	2021-12-01	NULL	Active	25	25
27	Savings	ACC1027	15000.00	2022-04-25	NULL	Active	27	27
29	Savings	ACC1029	22000.00	2021-09-12	NULL	Active	29	29
31	Savings	ACC1031	19000.00	2020-11-10	NULL	Active	31	31
33	Savings	ACC1033	8000.00	2022-02-03	NULL	Active	33	33
35	Savings	ACC1035	9500.00	2023-05-19	NULL	Active	35	35
37	Savings	ACC1037	15000.00	2021-08-11	NULL	Active	37	37
39	Savings	ACC1039	5000.00	2023-01-18	NULL	Active	39	39
41	Savings	ACC1041	32000.00	2022-07-01	NULL	Active	41	41
43	Savings	ACC1043	24000.00	2023-02-13	NULL	Active	43	43
45	Savings	ACC1045	18000.00	2021-06-30	NULL	Active	45	45
47	Savings	ACC1047	13000.00	2022-09-13	NULL	Active	[illegible]	47
49	Savings	ACC1049	11000.00	2023-02-05	NULL	Active	49	49
52	Savings	ACC52	5000.00	2020-11-05	2023-10-20	Inactive	52	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

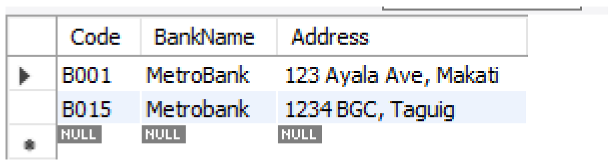
i. JOIN links the **Customer** table with the **Account** table.
ii. IN ('Savings') filters only accounts of those types.
iii. The SELECT clause retrieves customer details.

b. Find Bank name “MetroBanK”

SQL CODE

```
SELECT *
FROM Bank
WHERE BankName IN ('MetroBank');
```

OUTPUT



	Code	BankName	Address
▶	B001	MetroBank	123 Ayala Ave, Makati
	B015	Metrobank	1234 BGC, Taguig
*	NULL	NULL	NULL

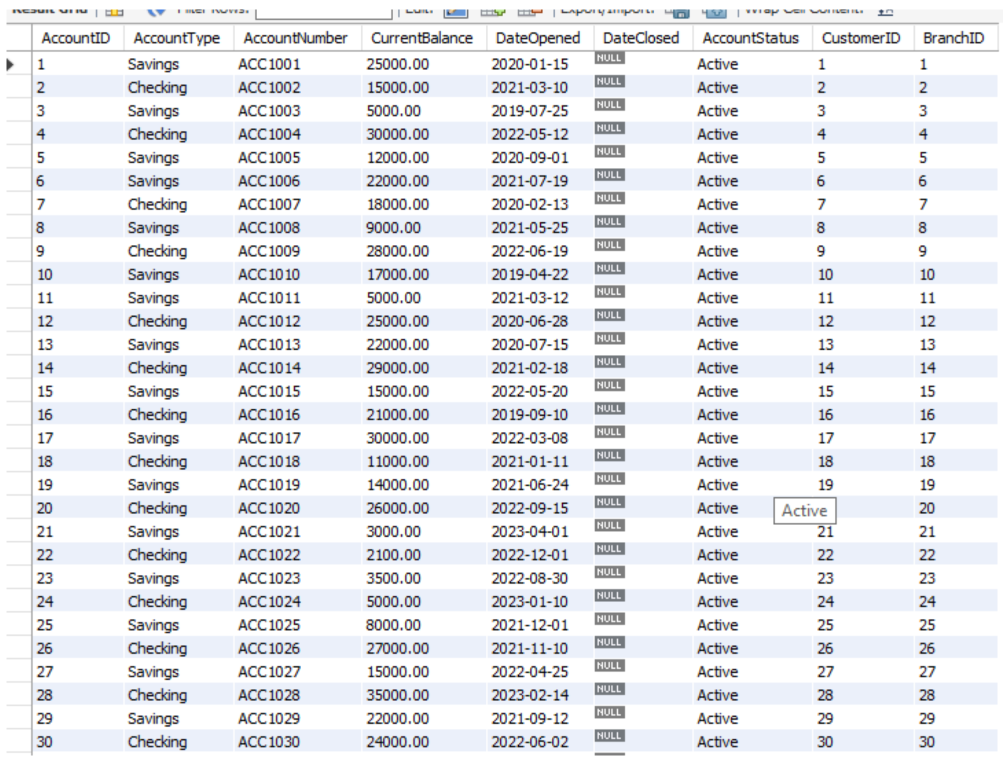
i. **Explanation**: This query retrieves all records from the **Bank** table where the **BankName** is **'MetroBank'**.
ii. **WHERE BankName IN ('MetroBank')** filters the records to include only **MetroBank**.

c. Find accounts that are 'Active'

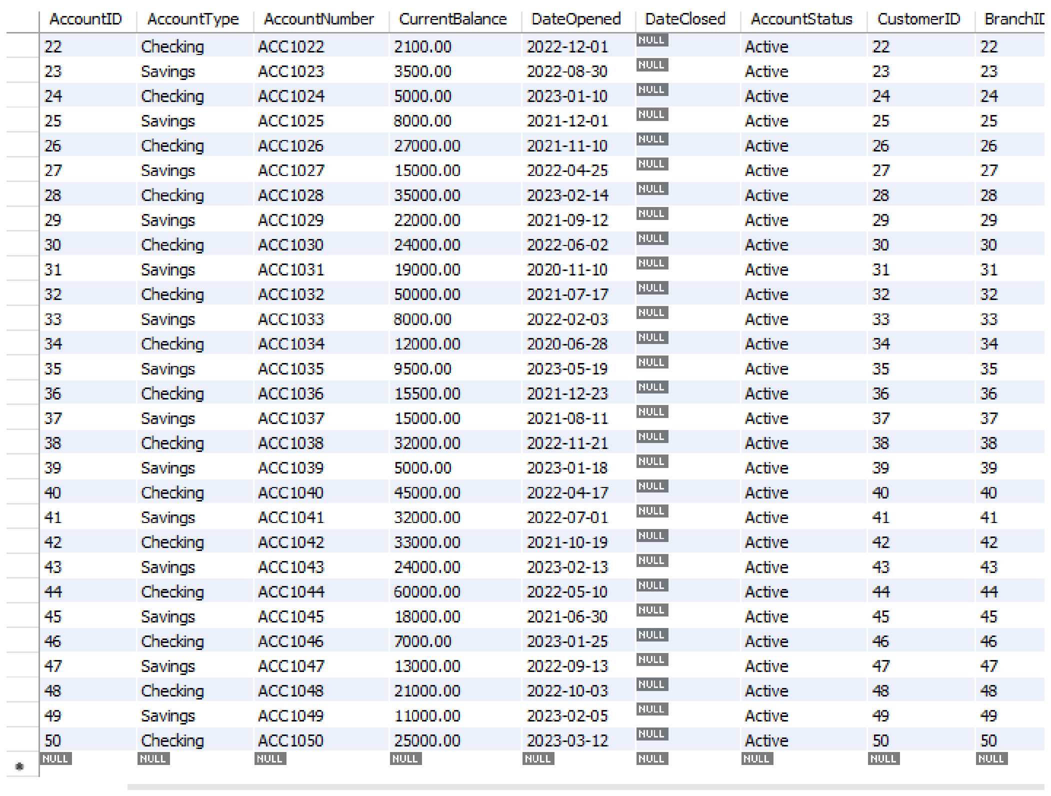
SQL CODE

```
SELECT *
FROM Account
WHERE AccountStatus IN ('Active');
```

OUTPUT



AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
1	Savings	ACC1001	25000.00	2020-01-15	NULL	Active	1	1
2	Checking	ACC1002	15000.00	2021-03-10	NULL	Active	2	2
3	Savings	ACC1003	5000.00	2019-07-25	NULL	Active	3	3
4	Checking	ACC1004	30000.00	2022-05-12	NULL	Active	4	4
5	Savings	ACC1005	12000.00	2020-09-01	NULL	Active	5	5
6	Savings	ACC1006	22000.00	2021-07-19	NULL	Active	6	6
7	Checking	ACC1007	18000.00	2020-02-13	NULL	Active	7	7
8	Savings	ACC1008	9000.00	2021-05-25	NULL	Active	8	8
9	Checking	ACC1009	28000.00	2022-06-19	NULL	Active	9	9
10	Savings	ACC1010	17000.00	2019-04-22	NULL	Active	10	10
11	Savings	ACC1011	5000.00	2021-03-12	NULL	Active	11	11
12	Checking	ACC1012	25000.00	2020-06-28	NULL	Active	12	12
13	Savings	ACC1013	22000.00	2020-07-15	NULL	Active	13	13
14	Checking	ACC1014	29000.00	2021-02-18	NULL	Active	14	14
15	Savings	ACC1015	15000.00	2022-05-20	NULL	Active	15	15
16	Checking	ACC1016	21000.00	2019-09-10	NULL	Active	16	16
17	Savings	ACC1017	30000.00	2022-03-08	NULL	Active	17	17
18	Checking	ACC1018	11000.00	2021-01-11	NULL	Active	18	18
19	Savings	ACC1019	14000.00	2021-06-24	NULL	Active	19	19
20	Checking	ACC1020	26000.00	2022-09-15	NULL	Active	Active	20
21	Savings	ACC1021	3000.00	2023-04-01	NULL	Active	21	21
22	Checking	ACC1022	2100.00	2022-12-01	NULL	Active	22	22
23	Savings	ACC1023	3500.00	2022-08-30	NULL	Active	23	23
24	Checking	ACC1024	5000.00	2023-01-10	NULL	Active	24	24
25	Savings	ACC1025	8000.00	2021-12-01	NULL	Active	25	25
26	Checking	ACC1026	27000.00	2021-11-10	NULL	Active	26	26
27	Savings	ACC1027	15000.00	2022-04-25	NULL	Active	27	27
28	Checking	ACC1028	35000.00	2023-02-14	NULL	Active	28	28
29	Savings	ACC1029	22000.00	2021-09-12	NULL	Active	29	29
30	Checking	ACC1030	24000.00	2022-06-02	NULL	Active	30	30



AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchI
22	Checking	ACC1022	2100.00	2022-12-01	NULL	Active	22	22
23	Savings	ACC1023	3500.00	2022-08-30	NULL	Active	23	23
24	Checking	ACC1024	5000.00	2023-01-10	NULL	Active	24	24
25	Savings	ACC1025	8000.00	2021-12-01	NULL	Active	25	25
26	Checking	ACC1026	27000.00	2021-11-10	NULL	Active	26	26
27	Savings	ACC1027	15000.00	2022-04-25	NULL	Active	27	27
28	Checking	ACC1028	35000.00	2023-02-14	NULL	Active	28	28
29	Savings	ACC1029	22000.00	2021-09-12	NULL	Active	29	29
30	Checking	ACC1030	24000.00	2022-06-02	NULL	Active	30	30
31	Savings	ACC1031	19000.00	2020-11-10	NULL	Active	31	31
32	Checking	ACC1032	50000.00	2021-07-17	NULL	Active	32	32
33	Savings	ACC1033	8000.00	2022-02-03	NULL	Active	33	33
34	Checking	ACC1034	12000.00	2020-06-28	NULL	Active	34	34
35	Savings	ACC1035	9500.00	2023-05-19	NULL	Active	35	35
36	Checking	ACC1036	15500.00	2021-12-23	NULL	Active	36	36
37	Savings	ACC1037	15000.00	2021-08-11	NULL	Active	37	37
38	Checking	ACC1038	32000.00	2022-11-21	NULL	Active	38	38
39	Savings	ACC1039	5000.00	2023-01-18	NULL	Active	39	39
40	Checking	ACC1040	45000.00	2022-04-17	NULL	Active	40	40
41	Savings	ACC1041	32000.00	2022-07-01	NULL	Active	41	41
42	Checking	ACC1042	33000.00	2021-10-19	NULL	Active	42	42
43	Savings	ACC1043	24000.00	2023-02-13	NULL	Active	43	43
44	Checking	ACC1044	60000.00	2022-05-10	NULL	Active	44	44
45	Savings	ACC1045	18000.00	2021-06-30	NULL	Active	45	45
46	Checking	ACC1046	7000.00	2023-01-25	NULL	Active	46	46
47	Savings	ACC1047	13000.00	2022-09-13	NULL	Active	47	47
48	Checking	ACC1048	21000.00	2022-10-03	NULL	Active	48	48
49	Savings	ACC1049	11000.00	2023-02-05	NULL	Active	49	49
50	Checking	ACC1050	25000.00	2023-03-12	NULL	Active	50	50
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

i. **Explanation:** This query returns all accounts where the **AccountStatus** is either **'Active'** or **'Dormant'**.

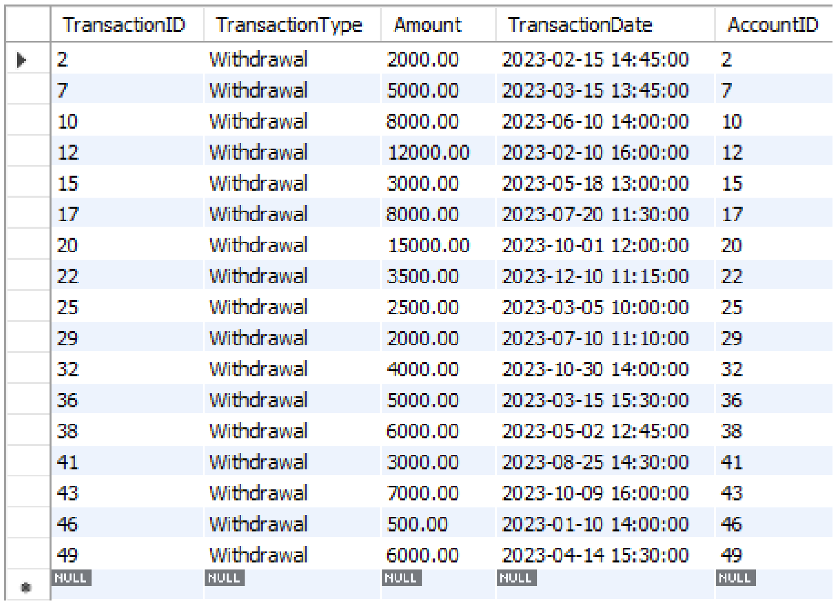
ii. **WHERE AccountStatus IN ('Active', 'Dormant')** filters the accounts to only those with **'Active'** or **'Dormant'** status.

d. Find Transaction type 'Withdrawal'

SQL CODE

```
SELECT *
FROM Transaction
WHERE TransactionType IN ('Withdrawal');
```

OUTPUT



TransactionID	TransactionType	Amount	TransactionDate	AccountID
2	Withdrawal	2000.00	2023-02-15 14:45:00	2
7	Withdrawal	5000.00	2023-03-15 13:45:00	7
10	Withdrawal	8000.00	2023-06-10 14:00:00	10
12	Withdrawal	12000.00	2023-02-10 16:00:00	12
15	Withdrawal	3000.00	2023-05-18 13:00:00	15
17	Withdrawal	8000.00	2023-07-20 11:30:00	17
20	Withdrawal	15000.00	2023-10-01 12:00:00	20
22	Withdrawal	3500.00	2023-12-10 11:15:00	22
25	Withdrawal	2500.00	2023-03-05 10:00:00	25
29	Withdrawal	2000.00	2023-07-10 11:10:00	29
32	Withdrawal	4000.00	2023-10-30 14:00:00	32
36	Withdrawal	5000.00	2023-03-15 15:30:00	36
38	Withdrawal	6000.00	2023-05-02 12:45:00	38
41	Withdrawal	3000.00	2023-08-25 14:30:00	41
43	Withdrawal	7000.00	2023-10-09 16:00:00	43
46	Withdrawal	500.00	2023-01-10 14:00:00	46
49	Withdrawal	6000.00	2023-04-14 15:30:00	49
NULL	NULL	NULL	NULL	NULL

i. **Explanation:** This query returns all transactions where the **TransactionType** is 'Withdrawal'.

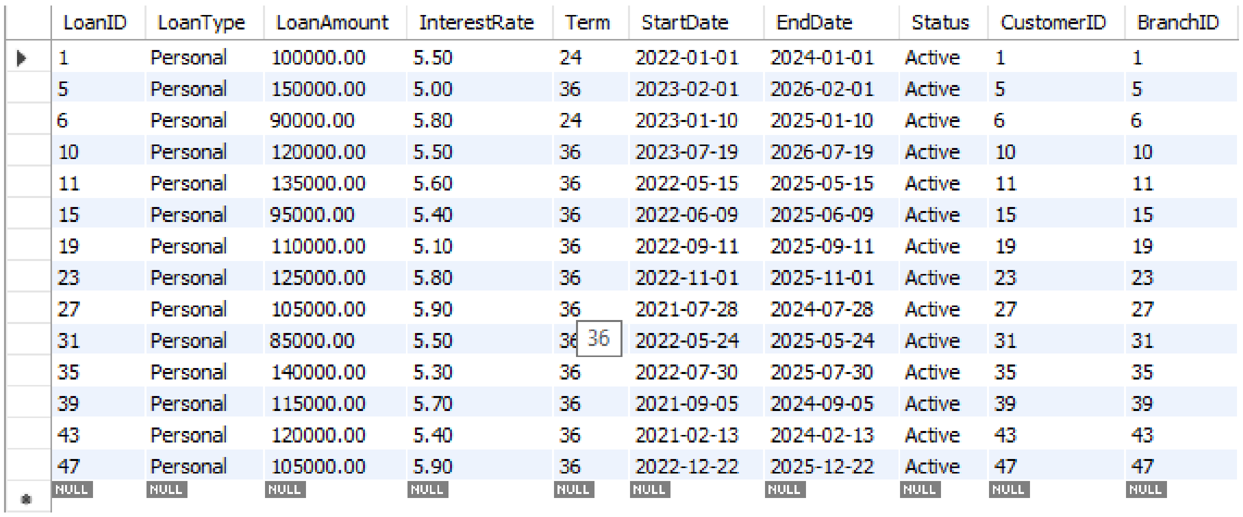
ii. WHERE TransactionType IN ('Withdrawal') filters the results to only withdrawal records.

e. Retrieve loans with types 'Personal

SQL CODE

```sql
SELECT *
FROM Loan
WHERE LoanType IN ('Personal');
```

OUTPUT



LoanID	LoanType	LoanAmount	InterestRate	Term	StartDate	EndDate	Status	CustomerID	BranchID
1	Personal	100000.00	5.50	24	2022-01-01	2024-01-01	Active	1	1
5	Personal	150000.00	5.00	36	2023-02-01	2026-02-01	Active	5	5
6	Personal	90000.00	5.80	24	2023-01-10	2025-01-10	Active	6	6
10	Personal	120000.00	5.50	36	2023-07-19	2026-07-19	Active	10	10
11	Personal	135000.00	5.60	36	2022-05-15	2025-05-15	Active	11	11
15	Personal	95000.00	5.40	36	2022-06-09	2025-06-09	Active	15	15
19	Personal	110000.00	5.10	36	2022-09-11	2025-09-11	Active	19	19
23	Personal	125000.00	5.80	36	2022-11-01	2025-11-01	Active	23	23
27	Personal	105000.00	5.90	36	2021-07-28	2024-07-28	Active	27	27
31	Personal	85000.00	5.50	36	2022-05-24	2025-05-24	Active	31	31
35	Personal	140000.00	5.30	36	2022-07-30	2025-07-30	Active	35	35
39	Personal	115000.00	5.70	36	2021-09-05	2024-09-05	Active	39	39
43	Personal	120000.00	5.40	36	2021-02-13	2024-02-13	Active	43	43
47	Personal	105000.00	5.90	36	2022-12-22	2025-12-22	Active	47	47
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

i. **Explanation**: This query retrieves all loans that are of **'Personal'** type.

ii. **WHERE LoanType IN ('Personal')** filters the loans to only those of type **Personal**.

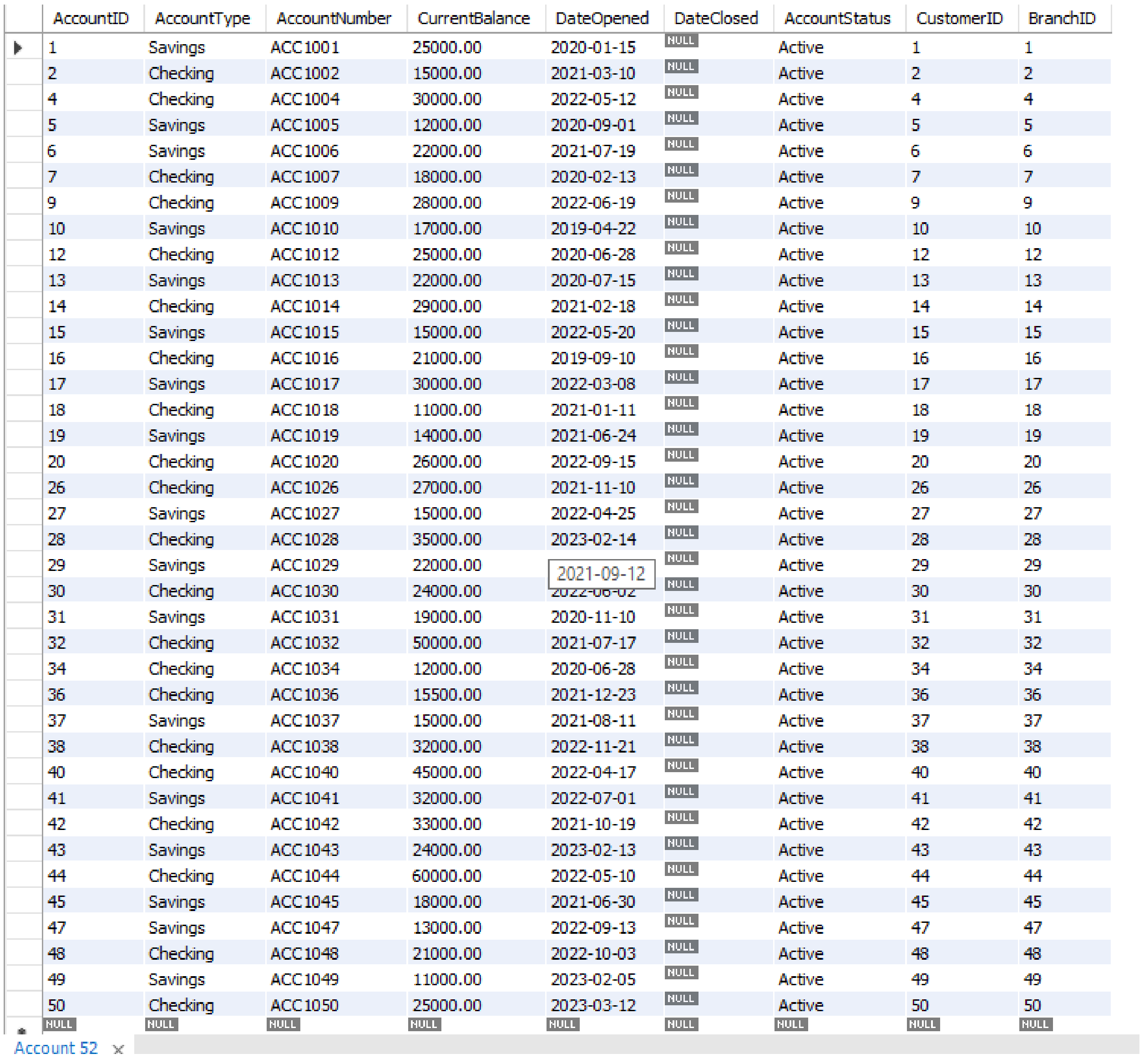
3.3. AND Operator

a. Get all customers with a balance greater than 10,000 and an active account:

SQL CODE

```sql
SELECT c.CustomerID, c.FirstName, c.LastName, c.Email, c.PhoneNumber
FROM Customer c
JOIN Account a ON c.CustomerID = a.CustomerID
WHERE a.CurrentBalance > 10000
  AND a.AccountStatus = 'Active';
```

OUTPUT



AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
1	Savings	ACC1001	25000.00	2020-01-15	NULL	Active	1	1
2	Checking	ACC1002	15000.00	2021-03-10	NULL	Active	2	2
4	Checking	ACC1004	30000.00	2022-05-12	NULL	Active	4	4
5	Savings	ACC1005	12000.00	2020-09-01	NULL	Active	5	5
6	Savings	ACC1006	22000.00	2021-07-19	NULL	Active	6	6
7	Checking	ACC1007	18000.00	2020-02-13	NULL	Active	7	7
9	Checking	ACC1009	28000.00	2022-06-19	NULL	Active	9	9
10	Savings	ACC1010	17000.00	2019-04-22	NULL	Active	10	10
12	Checking	ACC1012	25000.00	2020-06-28	NULL	Active	12	12
13	Savings	ACC1013	22000.00	2020-07-15	NULL	Active	13	13
14	Checking	ACC1014	29000.00	2021-02-18	NULL	Active	14	14
15	Savings	ACC1015	15000.00	2022-05-20	NULL	Active	15	15
16	Checking	ACC1016	21000.00	2019-09-10	NULL	Active	16	16
17	Savings	ACC1017	30000.00	2022-03-08	NULL	Active	17	17
18	Checking	ACC1018	11000.00	2021-01-11	NULL	Active	18	18
19	Savings	ACC1019	14000.00	2021-06-24	NULL	Active	19	19
20	Checking	ACC1020	26000.00	2022-09-15	NULL	Active	20	20
26	Checking	ACC1026	27000.00	2021-11-10	NULL	Active	26	26
27	Savings	ACC1027	15000.00	2022-04-25	NULL	Active	27	27
28	Checking	ACC1028	35000.00	2023-02-14	NULL	Active	28	28
29	Savings	ACC1029	22000.00	2021-09-12	NULL	Active	29	29
30	Checking	ACC1030	24000.00	2022-06-02	NULL	Active	30	30
31	Savings	ACC1031	19000.00	2020-11-10	NULL	Active	31	31
32	Checking	ACC1032	50000.00	2021-07-17	NULL	Active	32	32
34	Checking	ACC1034	12000.00	2020-06-28	NULL	Active	34	34
36	Checking	ACC1036	15500.00	2021-12-23	NULL	Active	36	36
37	Savings	ACC1037	15000.00	2021-08-11	NULL	Active	37	37
38	Checking	ACC1038	32000.00	2022-11-21	NULL	Active	38	38
40	Checking	ACC1040	45000.00	2022-04-17	NULL	Active	40	40
41	Savings	ACC1041	32000.00	2022-07-01	NULL	Active	41	41
42	Checking	ACC1042	33000.00	2021-10-19	NULL	Active	42	42
43	Savings	ACC1043	24000.00	2023-02-13	NULL	Active	43	43
44	Checking	ACC1044	60000.00	2022-05-10	NULL	Active	44	44
45	Savings	ACC1045	18000.00	2021-06-30	NULL	Active	45	45
47	Savings	ACC1047	13000.00	2022-09-13	NULL	Active	47	47
48	Checking	ACC1048	21000.00	2022-10-03	NULL	Active	48	48
49	Savings	ACC1049	11000.00	2023-02-05	NULL	Active	49	49
50	Checking	ACC1050	25000.00	2023-03-12	NULL	Active	50	50
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Account 52 ×

i. JOIN links the Customer table with the Account table.

ii. CurrentBalance > 10000 ensures only accounts with more than 10,000 balance are included.

iii. AND a.AccountStatus = 'Active' further filters those accounts to only active ones.

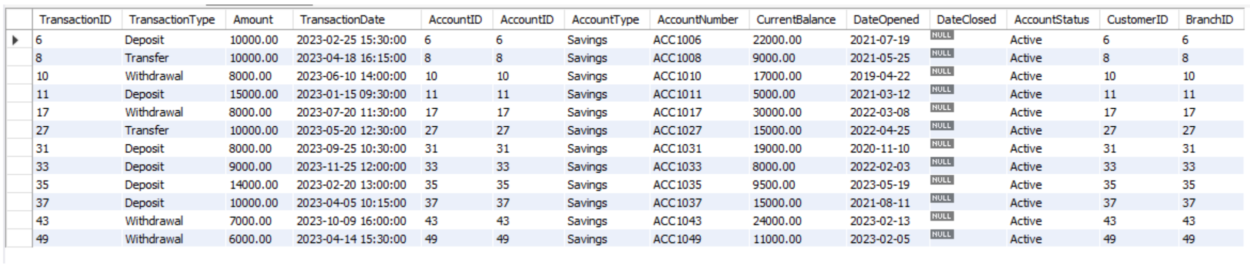
iv. The result will be a list of customers who meet both conditions at the same time.

b. Find transactions over ₱5,000 and made from 'Savings' accounts:

SQL CODE

```sql
SELECT *
FROM Transaction t
JOIN Account a ON t.AccountID = a.AccountID
WHERE t.Amount > 5000 AND a.AccountType = 'Savings';
```

OUTPUT



TransactionID	TransactionType	Amount	TransactionDate	AccountID	AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
6	Deposit	10000.00	2023-02-25 15:30:00	6	6	Savings	ACC1006	22000.00	2021-07-19	NULL	Active	6	6
8	Transfer	10000.00	2023-04-18 16:15:00	8	8	Savings	ACC1008	9000.00	2021-05-25	NULL	Active	8	8
10	Withdrawal	8000.00	2023-06-10 14:00:00	10	10	Savings	ACC1010	17000.00	2019-04-22	NULL	Active	10	10
11	Deposit	15000.00	2023-01-15 09:30:00	11	11	Savings	ACC1011	5000.00	2021-03-12	NULL	Active	11	11
17	Withdrawal	8000.00	2023-07-20 11:30:00	17	17	Savings	ACC1017	30000.00	2022-03-08	NULL	Active	17	17
27	Transfer	10000.00	2023-05-20 12:30:00	27	27	Savings	ACC1027	15000.00	2022-04-25	NULL	Active	27	27
31	Deposit	8000.00	2023-09-25 10:30:00	31	31	Savings	ACC1031	19000.00	2020-11-10	NULL	Active	31	31
33	Deposit	9000.00	2023-11-25 12:00:00	33	33	Savings	ACC1033	8000.00	2022-02-03	NULL	Active	33	33
35	Deposit	14000.00	2023-02-20 13:00:00	35	35	Savings	ACC1035	9500.00	2023-05-19	NULL	Active	35	35
37	Deposit	10000.00	2023-04-05 10:15:00	37	37	Savings	ACC1037	15000.00	2021-08-11	NULL	Active	37	37
43	Withdrawal	7000.00	2023-10-09 16:00:00	43	43	Savings	ACC1043	24000.00	2023-02-13	NULL	Active	43	43
49	Withdrawal	6000.00	2023-04-14 15:30:00	49	49	Savings	ACC1049	11000.00	2023-02-05	NULL	Active	49	49

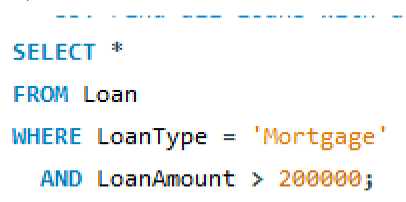
i. **Explanation**: This query retrieves **transactions** where the **transaction amount** is greater than **₱5,000** and the associated account is of type **'Savings'**.

ii. **JOIN Account a ON t.AccountID = a.AccountID** links the transaction with the account.

iii. **AND a.AccountType = 'Savings'** filters the transactions to only those from **Savings accounts**.

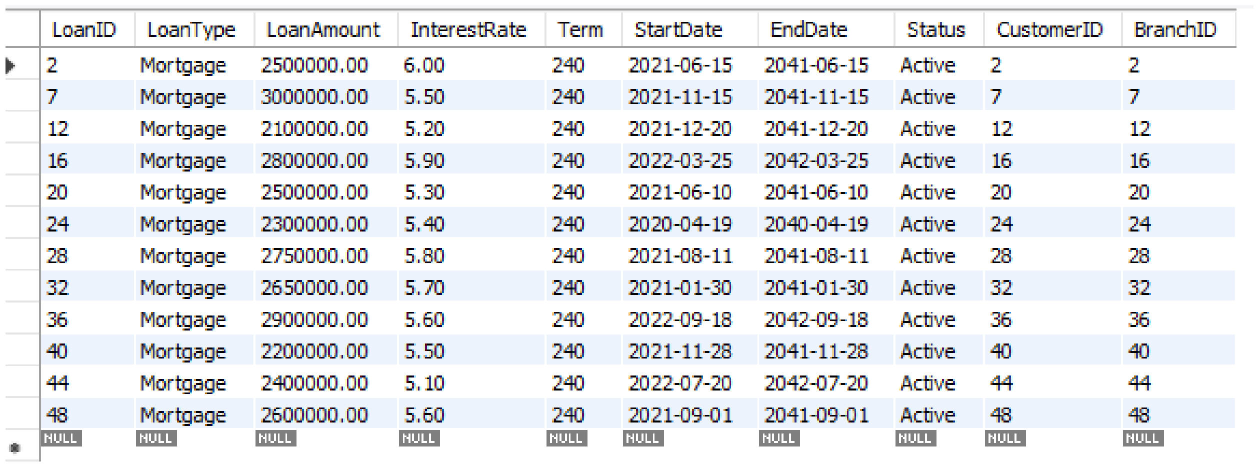
c. **Find all loans with a type of "Mortgage" and a loan amount greater than ₱200,000.**

SQL CODE



```
SELECT *
FROM Loan
WHERE LoanType = 'Mortgage'
  AND LoanAmount > 200000;
```

OUTPUT



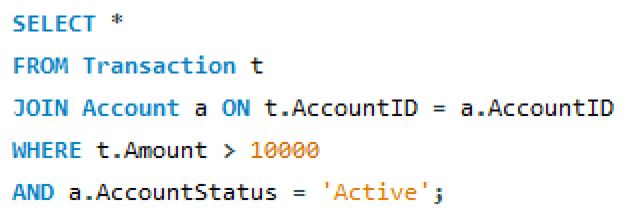
LoanID	LoanType	LoanAmount	InterestRate	Term	StartDate	EndDate	Status	CustomerID	BranchID
2	Mortgage	2500000.00	6.00	240	2021-06-15	2041-06-15	Active	2	2
7	Mortgage	3000000.00	5.50	240	2021-11-15	2041-11-15	Active	7	7
12	Mortgage	2100000.00	5.20	240	2021-12-20	2041-12-20	Active	12	12
16	Mortgage	2800000.00	5.90	240	2022-03-25	2042-03-25	Active	16	16
20	Mortgage	2500000.00	5.30	240	2021-06-10	2041-06-10	Active	20	20
24	Mortgage	2300000.00	5.40	240	2020-04-19	2040-04-19	Active	24	24
28	Mortgage	2750000.00	5.80	240	2021-08-11	2041-08-11	Active	28	28
32	Mortgage	2650000.00	5.70	240	2021-01-30	2041-01-30	Active	32	32
36	Mortgage	2900000.00	5.60	240	2022-09-18	2042-09-18	Active	36	36
40	Mortgage	2200000.00	5.50	240	2021-11-28	2041-11-28	Active	40	40
44	Mortgage	2400000.00	5.10	240	2022-07-20	2042-07-20	Active	44	44
48	Mortgage	2600000.00	5.60	240	2021-09-01	2041-09-01	Active	48	48
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

i. **Explanation**: This query returns **Mortgage loans** with a loan amount greater than **₱200,000**.

ii. **AND LoanAmount > 200000** filters the loans to include only those with an amount greater than **₱200,000**.

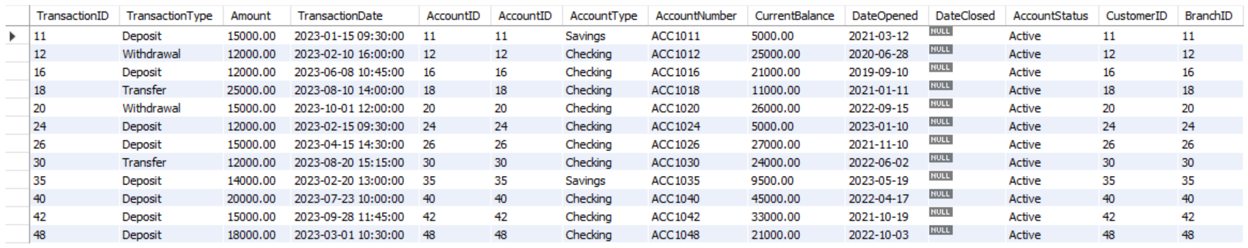
d. **Find transactions with an amount greater than ₱10,000 from active accounts.**

SQL CODE



```
SELECT *
FROM Transaction t
JOIN Account a ON t.AccountID = a.AccountID
WHERE t.Amount > 10000
AND a.AccountStatus = 'Active';
```

OUTPUT



	TransactionID	TransactionType	Amount	TransactionDate	AccountID	AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
▶	11	Deposit	15000.00	2023-01-15 09:30:00	11	11	Savings	ACC1011	5000.00	2021-03-12	NULL	Active	11	11
	12	Withdrawal	12000.00	2023-02-10 16:00:00	12	12	Checking	ACC1012	25000.00	2020-06-28	NULL	Active	12	12
	16	Deposit	12000.00	2023-06-08 10:45:00	16	16	Checking	ACC1016	21000.00	2019-09-10	NULL	Active	16	16
	18	Transfer	25000.00	2023-08-10 14:00:00	18	18	Checking	ACC1018	11000.00	2021-01-11	NULL	Active	18	18
	20	Withdrawal	15000.00	2023-10-01 12:00:00	20	20	Checking	ACC1020	26000.00	2022-09-15	NULL	Active	20	20
	24	Deposit	12000.00	2023-02-15 09:30:00	24	24	Checking	ACC1024	5000.00	2023-01-10	NULL	Active	24	24
	26	Deposit	15000.00	2023-04-15 14:30:00	26	26	Checking	ACC1026	27000.00	2021-11-10	NULL	Active	26	26
	30	Transfer	12000.00	2023-08-20 15:15:00	30	30	Checking	ACC1030	24000.00	2022-06-02	NULL	Active	30	30
	35	Deposit	14000.00	2023-02-20 13:00:00	35	35	Savings	ACC1035	9500.00	2023-05-19	NULL	Active	35	35
	40	Deposit	20000.00	2023-07-23 10:00:00	40	40	Checking	ACC1040	45000.00	2022-04-17	NULL	Active	40	40
	42	Deposit	15000.00	2023-09-28 11:45:00	42	42	Checking	ACC1042	33000.00	2021-10-19	NULL	Active	42	42
	48	Deposit	18000.00	2023-03-01 10:30:00	48	48	Checking	ACC1048	21000.00	2022-10-03	NULL	Active	48	48

i. **Explanation**: This query retrieves transactions greater than **₱10,000** made from accounts with **'Active'** status.

ii. **JOIN Account a ON t.AccountID = a.AccountID** links transactions to accounts.

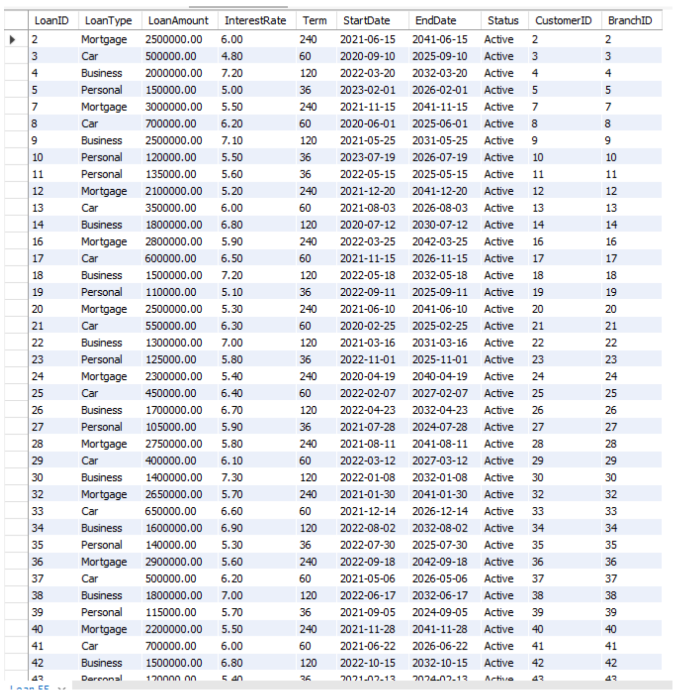
iii. **AND a.AccountStatus = 'Active'** ensures that only active accounts are included.

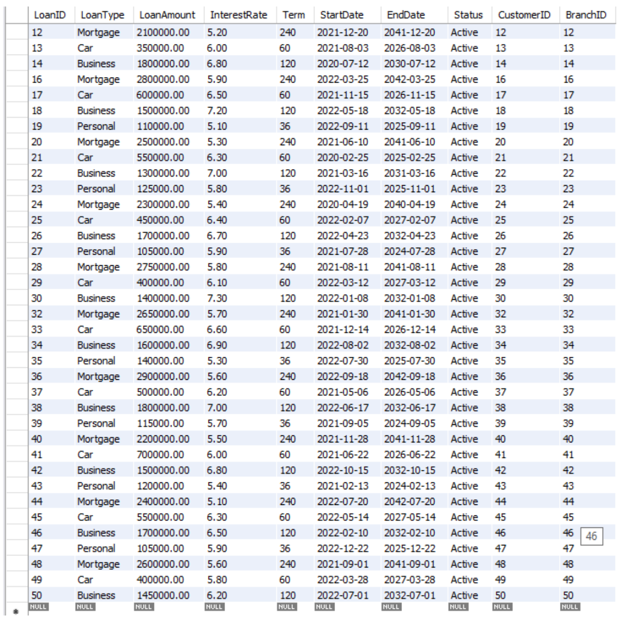
e. Get loans with status 'Active' and amount greater than ₱100,000:

SQL CODE

```
SELECT *
FROM Loan
WHERE Status = 'Active' AND LoanAmount > 100000;
```

OUTPUT





	LoanID	LoanType	LoanAmount	InterestRate	Term	StartDate	EndDate	Status	CustomerID	BranchID
▶	2	Mortgage	2500000.00	6.00	240	2021-06-15	2041-06-15	Active	2	2
	3	Car	500000.00	4.80	60	2020-09-10	2025-09-10	Active	3	3
	4	Business	2000000.00	7.20	120	2022-03-20	2032-03-20	Active	4	4
	5	Personal	150000.00	5.00	36	2023-02-01	2026-02-01	Active	5	5
	7	Mortgage	3000000.00	5.50	240	2021-11-15	2041-11-15	Active	7	7
	8	Car	700000.00	6.20	60	2020-06-01	2025-06-01	Active	8	8
	9	Business	2500000.00	7.10	120	2021-05-25	2031-05-25	Active	9	9
	10	Personal	120000.00	5.50	36	2023-07-19	2026-07-19	Active	10	10
	11	Personal	135000.00	5.60	36	2022-05-15	2025-05-15	Active	11	11
	12	Mortgage	2100000.00	5.20	240	2021-12-20	2041-12-20	Active	12	12
	13	Car	350000.00	6.00	60	2021-08-03	2026-08-03	Active	13	13
	14	Business	1800000.00	6.80	120	2020-07-12	2030-07-12	Active	14	14
	16	Mortgage	2800000.00	5.90	240	2022-03-25	2042-03-25	Active	16	16
	17	Car	600000.00	6.50	60	2021-11-15	2026-11-15	Active	17	17
	18	Business	1500000.00	7.20	120	2022-05-18	2032-05-18	Active	18	18
	19	Personal	110000.00	5.10	36	2022-09-11	2025-09-11	Active	19	19
	20	Mortgage	2500000.00	5.30	240	2021-06-10	2041-06-10	Active	20	20
	21	Car	550000.00	6.30	60	2020-02-25	2025-02-25	Active	21	21
	22	Business	1300000.00	7.00	120	2021-03-16	2031-03-16	Active	22	22
	23	Personal	125000.00	5.80	36	2022-11-01	2025-11-01	Active	23	23
	24	Mortgage	2300000.00	5.40	240	2020-04-19	2040-04-19	Active	24	24
	25	Car	450000.00	6.40	60	2022-02-07	2027-02-07	Active	25	25
	26	Business	1700000.00	6.70	120	2022-04-23	2032-04-23	Active	26	26
	27	Personal	105000.00	5.90	36	2021-07-28	2024-07-28	Active	27	27
	28	Mortgage	2750000.00	5.80	240	2021-08-11	2041-08-11	Active	28	28
	29	Car	400000.00	6.10	60	2022-03-12	2027-03-12	Active	29	29
	30	Business	1400000.00	7.30	120	2022-01-08	2032-01-08	Active	30	30
	32	Mortgage	2650000.00	5.70	240	2021-01-30	2041-01-30	Active	32	32
	33	Car	650000.00	6.60	60	2021-12-14	2026-12-14	Active	33	33
	34	Business	1600000.00	6.90	120	2022-08-02	2032-08-02	Active	34	34
	35	Personal	140000.00	5.30	36	2022-07-30	2025-07-30	Active	35	35
	36	Mortgage	2900000.00	5.60	240	2022-09-18	2042-09-18	Active	36	36
	37	Car	500000.00	6.20	60	2021-05-06	2026-05-06	Active	37	37
	38	Business	1800000.00	7.00	120	2022-06-17	2032-06-17	Active	38	38
	39	Personal	115000.00	5.70	36	2021-09-05	2024-09-05	Active	39	39
	40	Mortgage	2200000.00	5.50	240	2021-11-28	2041-11-28	Active	40	40
	41	Car	700000.00	6.00	60	2021-06-22	2026-06-22	Active	41	41
	42	Business	1500000.00	6.80	120	2022-10-15	2032-10-15	Active	42	42
	43	Personal	120000.00	5.40	36	2021-02-13	2024-02-13	Active	43	43
	44	Mortgage	2400000.00	5.10	240	2022-07-20	2042-07-20	Active	44	44
	45	Car	550000.00	6.30	60	2022-05-14	2027-05-14	Active	45	45
	46	Business	1700000.00	6.50	120	2022-02-10	2032-02-10	Active	46	46
	47	Personal	105000.00	5.90	36	2022-12-22	2025-12-22	Active	47	47
	48	Mortgage	2600000.00	5.60	240	2021-09-01	2041-09-01	Active	48	48
	49	Car	400000.00	5.80	60	2022-03-28	2027-03-28	Active	49	49
	50	Business	1450000.00	6.20	120	2022-07-01	2032-07-01	Active	50	50
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

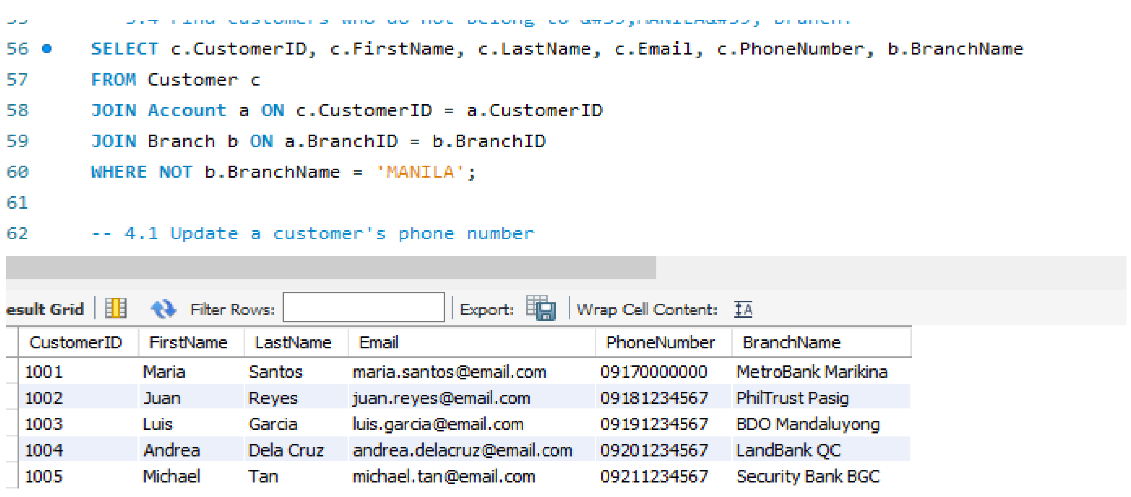
i. **Explanation**: This query returns all loans where the **Status** is 'Active' and the **LoanAmount** is greater than 100000.

ii. WHERE Status = 'Active' AND LoanAmount > 100000 filters the results to only active loans with amounts above 100,000.

3.4. NOT Operator

a. Find customers who do not belong to MANILA& branch:

SQL CODE AND OUTPUT



```sql
SELECT c.CustomerID, c.FirstName, c.LastName, c.Email, c.PhoneNumber, b.BranchName
FROM Customer c
JOIN Account a ON c.CustomerID = a.CustomerID
JOIN Branch b ON a.BranchID = b.BranchID
WHERE NOT b.BranchName = 'MANILA';

-- 4.1 Update a customer's phone number
```

CustomerID	FirstName	LastName	Email	PhoneNumber	BranchName
1001	Maria	Santos	maria.santos@email.com	09170000000	MetroBank Marikina
1002	Juan	Reyes	juan.reyes@email.com	09181234567	PhilTrust Pasig
1003	Luis	Garcia	luis.garcia@email.com	09191234567	BDO Mandaluyong
1004	Andrea	Dela Cruz	andrea.delacruz@email.com	09201234567	LandBank QC
1005	Michael	Tan	michael.tan@email.com	09211234567	Security Bank BGC

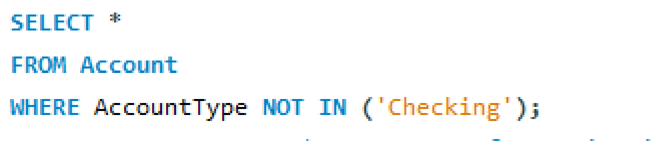
i. JOIN links Customer → Account → Branch so you can filter by branch name.

ii. WHERE NOT b.BranchName = 'MANILA' excludes all customers whose accounts are tied to the MANILA branch.

iii. The result will list only customers from other branches.

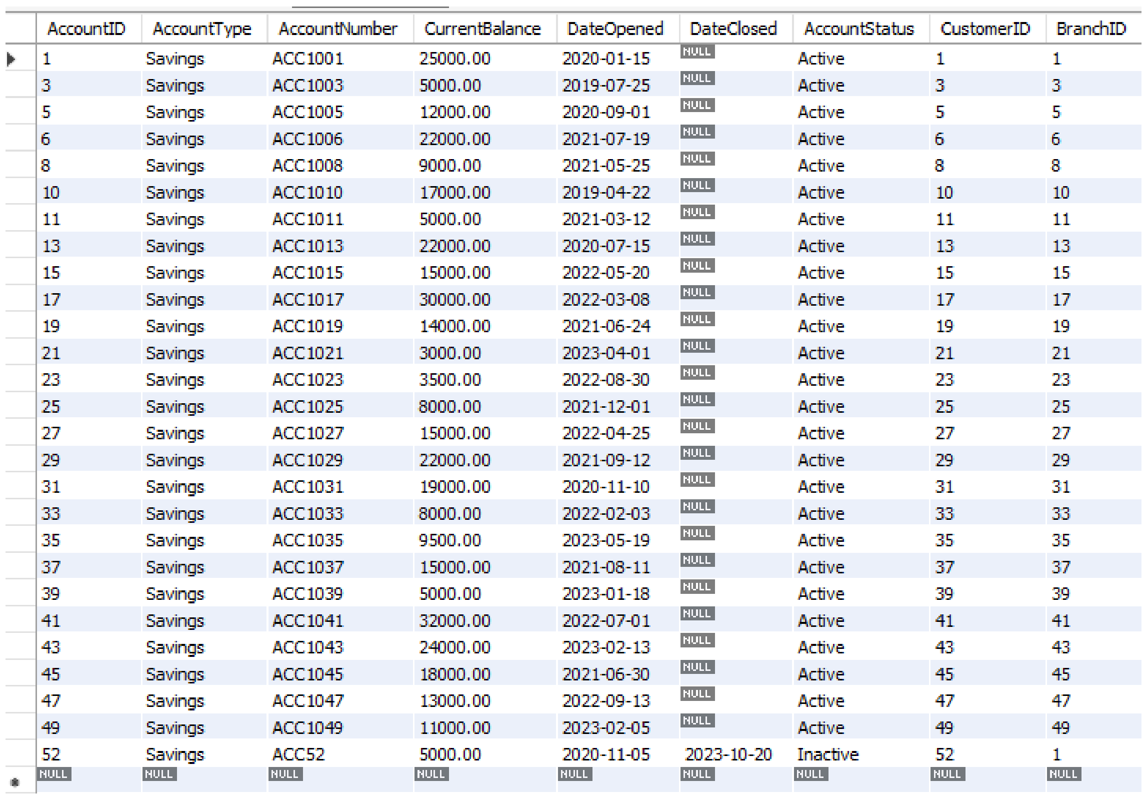
b. **Get accounts that are not of type 'Checking':**

SQL CODE



```sql
SELECT *
FROM Account
WHERE AccountType NOT IN ('Checking');
```

OUTPUT



AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
1	Savings	ACC1001	25000.00	2020-01-15	NULL	Active	1	1
3	Savings	ACC1003	5000.00	2019-07-25	NULL	Active	3	3
5	Savings	ACC1005	12000.00	2020-09-01	NULL	Active	5	5
6	Savings	ACC1006	22000.00	2021-07-19	NULL	Active	6	6
8	Savings	ACC1008	9000.00	2021-05-25	NULL	Active	8	8
10	Savings	ACC1010	17000.00	2019-04-22	NULL	Active	10	10
11	Savings	ACC1011	5000.00	2021-03-12	NULL	Active	11	11
13	Savings	ACC1013	22000.00	2020-07-15	NULL	Active	13	13
15	Savings	ACC1015	15000.00	2022-05-20	NULL	Active	15	15
17	Savings	ACC1017	30000.00	2022-03-08	NULL	Active	17	17
19	Savings	ACC1019	14000.00	2021-06-24	NULL	Active	19	19
21	Savings	ACC1021	3000.00	2023-04-01	NULL	Active	21	21
23	Savings	ACC1023	3500.00	2022-08-30	NULL	Active	23	23
25	Savings	ACC1025	8000.00	2021-12-01	NULL	Active	25	25
27	Savings	ACC1027	15000.00	2022-04-25	NULL	Active	27	27
29	Savings	ACC1029	22000.00	2021-09-12	NULL	Active	29	29
31	Savings	ACC1031	19000.00	2020-11-10	NULL	Active	31	31
33	Savings	ACC1033	8000.00	2022-02-03	NULL	Active	33	33
35	Savings	ACC1035	9500.00	2023-05-19	NULL	Active	35	35
37	Savings	ACC1037	15000.00	2021-08-11	NULL	Active	37	37
39	Savings	ACC1039	5000.00	2023-01-18	NULL	Active	39	39
41	Savings	ACC1041	32000.00	2022-07-01	NULL	Active	41	41
43	Savings	ACC1043	24000.00	2023-02-13	NULL	Active	43	43
45	Savings	ACC1045	18000.00	2021-06-30	NULL	Active	45	45
47	Savings	ACC1047	13000.00	2022-09-13	NULL	Active	47	47
49	Savings	ACC1049	11000.00	2023-02-05	NULL	Active	49	49
52	Savings	ACC52	5000.00	2020-11-05	2023-10-20	Inactive	52	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

i. **Explanation**: This query retrieves all accounts that are **not of type 'Checking'**.

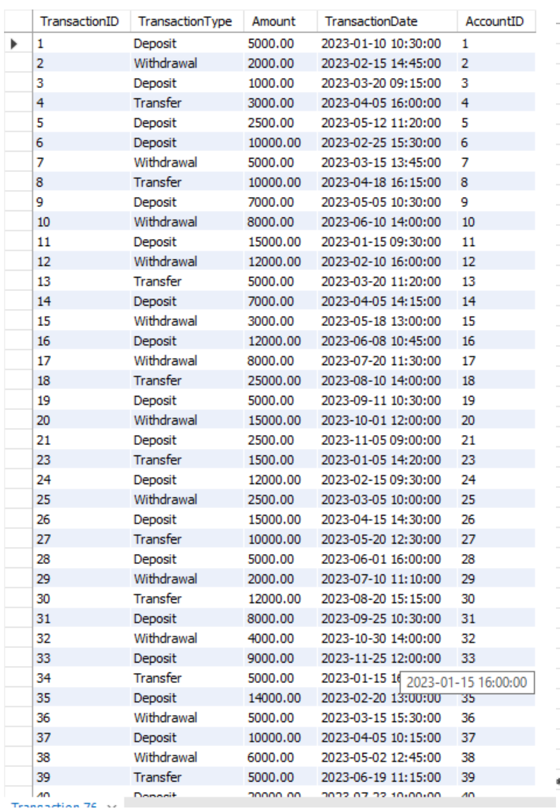
ii. **NOT IN ('Checking')** filters out accounts that are of type **'Checking'**, returning other account types like **'Savings'** or **'Business'**.

c. **Find transactions not made in December:**

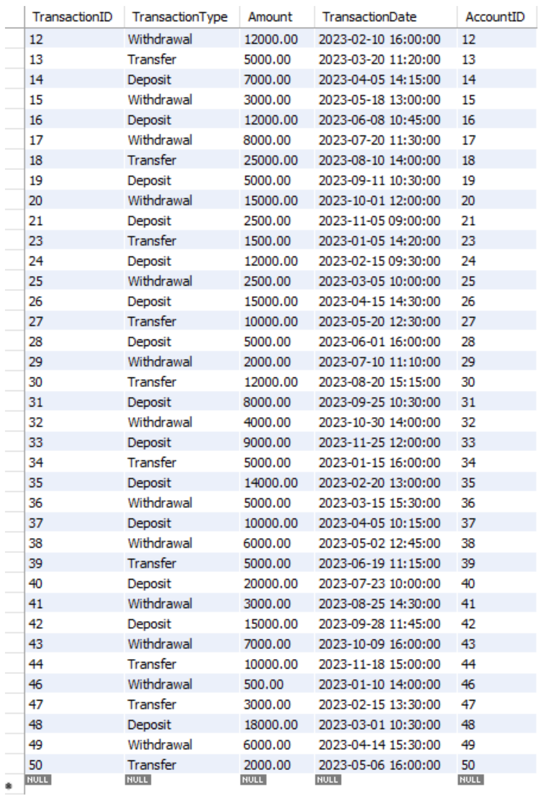
SQL CODE

```
SELECT *
FROM Transaction
WHERE MONTH(TransactionDate) <> 12;
```

OUTPUT



TransactionID	TransactionType	Amount	TransactionDate	AccountID
1	Deposit	5000.00	2023-01-10 10:30:00	1
2	Withdrawal	2000.00	2023-02-15 14:45:00	2
3	Deposit	1000.00	2023-03-20 09:15:00	3
4	Transfer	3000.00	2023-04-05 16:00:00	4
5	Deposit	2500.00	2023-05-12 11:20:00	5
6	Deposit	10000.00	2023-02-25 15:30:00	6
7	Withdrawal	5000.00	2023-03-15 13:45:00	7
8	Transfer	10000.00	2023-04-18 16:15:00	8
9	Deposit	7000.00	2023-05-05 10:30:00	9
10	Withdrawal	8000.00	2023-06-10 14:00:00	10
11	Deposit	15000.00	2023-01-15 09:30:00	11
12	Withdrawal	12000.00	2023-02-10 16:00:00	12
13	Transfer	5000.00	2023-03-20 11:20:00	13
14	Deposit	7000.00	2023-04-05 14:15:00	14
15	Withdrawal	3000.00	2023-05-18 13:00:00	15
16	Deposit	12000.00	2023-06-08 10:45:00	16
17	Withdrawal	8000.00	2023-07-20 11:30:00	17
18	Transfer	25000.00	2023-08-10 14:00:00	18
19	Deposit	5000.00	2023-09-11 10:30:00	19
20	Withdrawal	15000.00	2023-10-01 12:00:00	20
21	Deposit	2500.00	2023-11-05 09:00:00	21
23	Transfer	1500.00	2023-01-05 14:20:00	23
24	Deposit	12000.00	2023-02-15 09:30:00	24
25	Withdrawal	2500.00	2023-03-05 10:00:00	25
26	Deposit	15000.00	2023-04-15 14:30:00	26
27	Transfer	10000.00	2023-05-20 12:30:00	27
28	Deposit	5000.00	2023-06-01 16:00:00	28
29	Withdrawal	2000.00	2023-07-10 11:10:00	29
30	Transfer	12000.00	2023-08-20 15:15:00	30
31	Deposit	8000.00	2023-09-25 10:30:00	31
32	Withdrawal	4000.00	2023-10-30 14:00:00	32
33	Deposit	9000.00	2023-11-25 12:00:00	33
34	Transfer	5000.00	2023-01-15 16:00:00	
35	Deposit	14000.00	2023-02-20 13:00:00	35
36	Withdrawal	5000.00	2023-03-15 15:30:00	36
37	Deposit	10000.00	2023-04-05 10:15:00	37
38	Withdrawal	6000.00	2023-05-02 12:45:00	38
39	Transfer	5000.00	2023-06-19 11:15:00	39



TransactionID	TransactionType	Amount	TransactionDate	AccountID
12	Withdrawal	12000.00	2023-02-10 16:00:00	12
13	Transfer	5000.00	2023-03-20 11:20:00	13
14	Deposit	7000.00	2023-04-05 14:15:00	14
15	Withdrawal	3000.00	2023-05-18 13:00:00	15
16	Deposit	12000.00	2023-06-08 10:45:00	16
17	Withdrawal	8000.00	2023-07-20 11:30:00	17
18	Transfer	25000.00	2023-08-10 14:00:00	18
19	Deposit	5000.00	2023-09-11 10:30:00	19
20	Withdrawal	15000.00	2023-10-01 12:00:00	20
21	Deposit	2500.00	2023-11-05 09:00:00	21
23	Transfer	1500.00	2023-01-05 14:20:00	23
24	Deposit	12000.00	2023-02-15 09:30:00	24
25	Withdrawal	2500.00	2023-03-05 10:00:00	25
26	Deposit	15000.00	2023-04-15 14:30:00	26
27	Transfer	10000.00	2023-05-20 12:30:00	27
28	Deposit	5000.00	2023-06-01 16:00:00	28
29	Withdrawal	2000.00	2023-07-10 11:10:00	29
30	Transfer	12000.00	2023-08-20 15:15:00	30
31	Deposit	8000.00	2023-09-25 10:30:00	31
32	Withdrawal	4000.00	2023-10-30 14:00:00	32
33	Deposit	9000.00	2023-11-25 12:00:00	33
34	Transfer	5000.00	2023-01-15 16:00:00	34
35	Deposit	14000.00	2023-02-20 13:00:00	35
36	Withdrawal	5000.00	2023-03-15 15:30:00	36
37	Deposit	10000.00	2023-04-05 10:15:00	37
38	Withdrawal	6000.00	2023-05-02 12:45:00	38
39	Transfer	5000.00	2023-06-19 11:15:00	39
40	Deposit	20000.00	2023-07-23 10:00:00	40
41	Withdrawal	3000.00	2023-08-25 14:30:00	41
42	Deposit	15000.00	2023-09-28 11:45:00	42
43	Withdrawal	7000.00	2023-10-09 16:00:00	43
44	Transfer	10000.00	2023-11-18 15:00:00	44
46	Withdrawal	500.00	2023-01-10 14:00:00	46
47	Transfer	3000.00	2023-02-15 13:30:00	47
48	Deposit	18000.00	2023-03-01 10:30:00	48
49	Withdrawal	6000.00	2023-04-14 15:30:00	49
50	Transfer	2000.00	2023-05-06 16:00:00	50
NULL	NULL	NULL	NULL	NULL

i. **Explanation:** This query retrieves all transactions that **did not occur in December.**

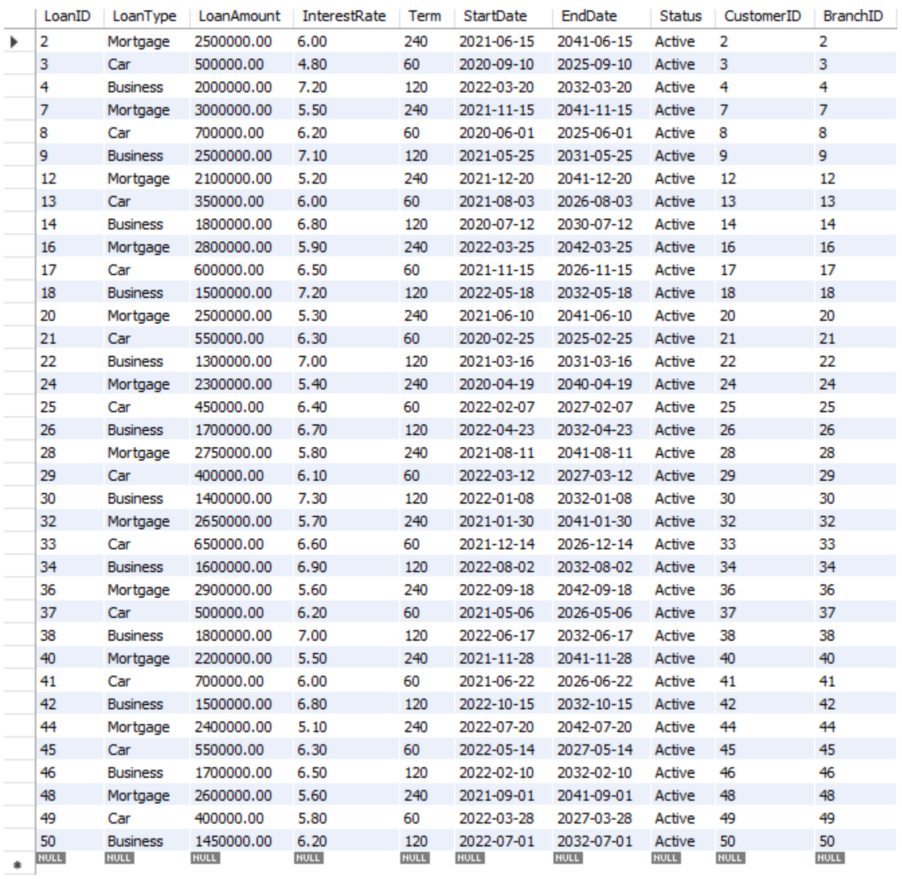
ii. **MONTH(TransactionDate) <> 12** excludes transactions that were made in **December.**

d. **Get all loans that are not of type 'Personal'**

SQL CODE

```sql
SELECT *
FROM Loan
WHERE LoanType NOT IN ('Personal');
```

OUTPUT



LoanID	LoanType	LoanAmount	InterestRate	Term	StartDate	EndDate	Status	CustomerID	BranchID
2	Mortgage	2500000.00	6.00	240	2021-06-15	2041-06-15	Active	2	2
3	Car	500000.00	4.80	60	2020-09-10	2025-09-10	Active	3	3
4	Business	2000000.00	7.20	120	2022-03-20	2032-03-20	Active	4	4
7	Mortgage	3000000.00	5.50	240	2021-11-15	2041-11-15	Active	7	7
8	Car	700000.00	6.20	60	2020-06-01	2025-06-01	Active	8	8
9	Business	2500000.00	7.10	120	2021-05-25	2031-05-25	Active	9	9
12	Mortgage	2100000.00	5.20	240	2021-12-20	2041-12-20	Active	12	12
13	Car	350000.00	6.00	60	2021-08-03	2026-08-03	Active	13	13
14	Business	1800000.00	6.80	120	2020-07-12	2030-07-12	Active	14	14
16	Mortgage	2800000.00	5.90	240	2022-03-25	2042-03-25	Active	16	16
17	Car	600000.00	6.50	60	2021-11-15	2026-11-15	Active	17	17
18	Business	1500000.00	7.20	120	2022-05-18	2032-05-18	Active	18	18
20	Mortgage	2500000.00	5.30	240	2021-06-10	2041-06-10	Active	20	20
21	Car	550000.00	6.30	60	2020-02-25	2025-02-25	Active	21	21
22	Business	1300000.00	7.00	120	2021-03-16	2031-03-16	Active	22	22
24	Mortgage	2300000.00	5.40	240	2020-04-19	2040-04-19	Active	24	24
25	Car	450000.00	6.40	60	2022-02-07	2027-02-07	Active	25	25
26	Business	1700000.00	6.70	120	2022-04-23	2032-04-23	Active	26	26
28	Mortgage	2750000.00	5.80	240	2021-08-11	2041-08-11	Active	28	28
29	Car	400000.00	6.10	60	2022-03-12	2027-03-12	Active	29	29
30	Business	1400000.00	7.30	120	2022-01-08	2032-01-08	Active	30	30
32	Mortgage	2650000.00	5.70	240	2021-01-30	2041-01-30	Active	32	32
33	Car	650000.00	6.60	60	2021-12-14	2026-12-14	Active	33	33
34	Business	1600000.00	6.90	120	2022-08-02	2032-08-02	Active	34	34
36	Mortgage	2900000.00	5.60	240	2022-09-18	2042-09-18	Active	36	36
37	Car	500000.00	6.20	60	2021-05-06	2026-05-06	Active	37	37
38	Business	1800000.00	7.00	120	2022-06-17	2032-06-17	Active	38	38
40	Mortgage	2200000.00	5.50	240	2021-11-28	2041-11-28	Active	40	40
41	Car	700000.00	6.00	60	2021-06-22	2026-06-22	Active	41	41
42	Business	1500000.00	6.80	120	2022-10-15	2032-10-15	Active	42	42
44	Mortgage	2400000.00	5.10	240	2022-07-20	2042-07-20	Active	44	44
45	Car	550000.00	6.30	60	2022-05-14	2027-05-14	Active	45	45
46	Business	1700000.00	6.50	120	2022-02-10	2032-02-10	Active	46	46
48	Mortgage	2600000.00	5.60	240	2021-09-01	2041-09-01	Active	48	48
49	Car	400000.00	5.80	60	2022-03-28	2027-03-28	Active	49	49
50	Business	1450000.00	6.20	120	2022-07-01	2032-07-01	Active	50	50
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

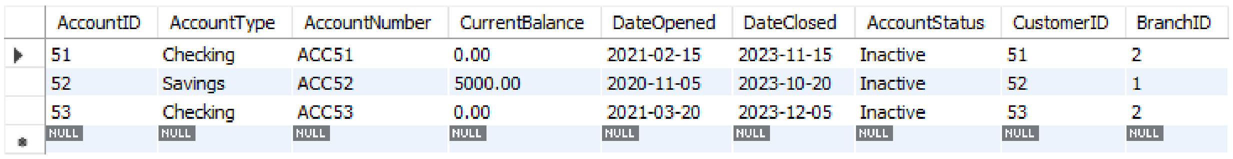
i. **Explanation**: This query retrieves all loans **except** those with the **'Personal'** type.
NOT IN ('Personal') excludes **Personal loans** and includes other types like **Mortgage**, **Auto**, etc

e. **Get accounts that are not 'Active'**

SQL CODE

```
SELECT *
FROM Account
WHERE AccountStatus NOT IN ('Active');
```

OUTPUT



AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
51	Checking	ACC51	0.00	2021-02-15	2023-11-15	Inactive	51	2
52	Savings	ACC52	5000.00	2020-11-05	2023-10-20	Inactive	52	1
53	Checking	ACC53	0.00	2021-03-20	2023-12-05	Inactive	53	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

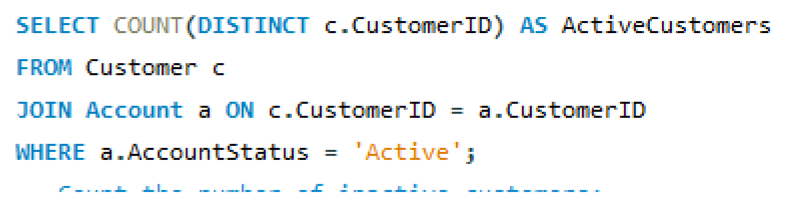
i. **Explanation**: This query retrieves accounts that are NOT **'Active'**

ii. **NOT IN ('Active')** filters out accounts with Active statuses, returning other account statuses like **‘Inactive’**.

3.5. COUNT Function

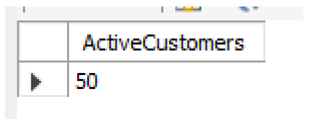
a. Count the number of active customers:

SQL CODE



```sql
SELECT COUNT(DISTINCT c.CustomerID) AS ActiveCustomers
FROM Customer c
JOIN Account a ON c.CustomerID = a.CustomerID
WHERE a.AccountStatus = 'Active';
   -- Count the number of inactive customers:
```

OUTPUT



	ActiveCustomers
▶	50

i. COUNT(*) counts the number of rows that meet the condition.

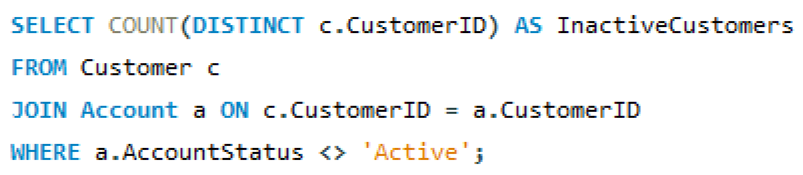
ii. JOIN ensures we only count customers who have accounts.

iii. WHERE a.AccountStatus = 'Active' filters active customers.

iv. WHERE a.AccountStatus <> 'Active' filters all others (inactive, closed, etc.).

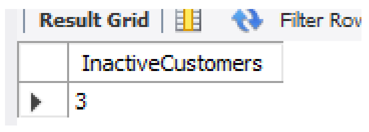
b. **Count the number of inactive customers:**

SQL CODE



```sql
SELECT COUNT(DISTINCT c.CustomerID) AS InactiveCustomers
FROM Customer c
JOIN Account a ON c.CustomerID = a.CustomerID
WHERE a.AccountStatus <> 'Active';
```

OUTPUT

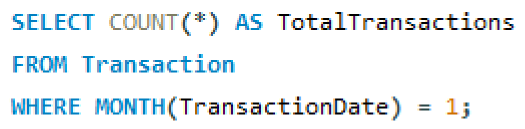


	InactiveCustomers
▶	3

i. **Explanation**: This query counts the number of distinct customers who do **not** have an **'Active'** account, implying they are inactive.

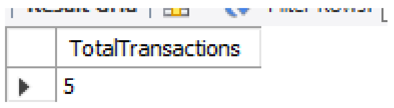
c. **Count the total transactions in January:**

SQL CODE



```sql
SELECT COUNT(*) AS TotalTransactions
FROM Transaction
WHERE MONTH(TransactionDate) = 1;
```

OUTPUT

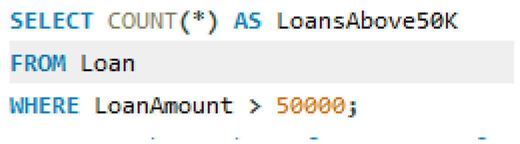


```
TotalTransactions
5
```

i. **Explanation**: This query counts the total number of **transactions** that occurred in **January**.

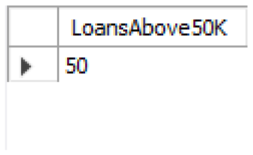
d. **Count the number of loans above ₱50,000**

SQL CODE



```sql
SELECT COUNT(*) AS LoansAbove50K
FROM Loan
WHERE LoanAmount > 50000;
```

OUTPUT



```
LoansAbove50K
50
```

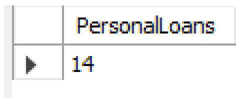
i. **Explanation**: This query counts the number of loans where the **loan amount** is greater than **₱50,000**.

e. **Count the number of 'personal loans'**

SQL CODE

```
SELECT COUNT(*) AS PersonalLoans
FROM Loan
WHERE LoanType = 'Personal';
```

OUTPUT



```
  PersonalLoans
▸ 14
```

i. **Explanation**: This query counts the number of loans where the **LoanType** is **'Personal'**.

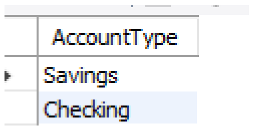
3.6. DISTINCT Keyword

a. **Get all unique account types:**

SQL CODE

```
SELECT DISTINCT AccountType
FROM Account;
```

OUTPUT



```
  AccountType
▸ Savings
  Checking
```

i. DISTINCT ensures that duplicate account types are removed from the result set.

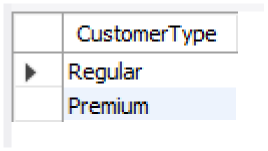
ii. The query will return each account type only once, even if many accounts share the same type.

b. **Get unique customer types:**

SQL CODE

```
SELECT DISTINCT CustomerType
FROM Customer;
```

OUTPUT



	CustomerType
▶	Regular
	Premium

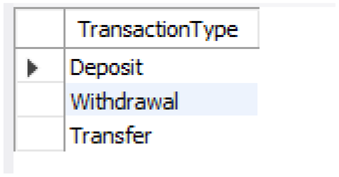
i. **Explanation**: This query returns all **unique customer types** (e.g., 'Regular', 'Premium') from the **Customer** table.

c. **Get distinct transaction types:**

SQL CODE

```
SELECT DISTINCT TransactionType
FROM Transaction;
```

OUTPUT



	TransactionType
▶	Deposit
	Withdrawal
	Transfer

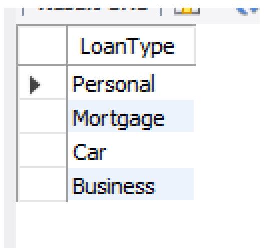
i. **Explanation**: This query retrieves all **unique transaction types** (e.g., 'Deposit', 'Withdrawal') from the **Transaction** table

d. **List distinct loan types:**

SQL CODE

```
SELECT DISTINCT LoanType
FROM Loan;
```

OUTPUT



	LoanType
▶	Personal
	Mortgage
	Car
	Business

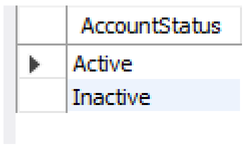
i. **Explanation**: This query lists all distinct **loan types** (e.g., 'Personal', 'Mortgage') from the **Loan** table.

e. **List distinct account statuses:**

SQL CODE

```
SELECT DISTINCT AccountStatus
FROM Account;
```

OUTPUT



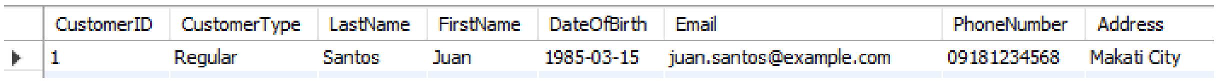
AccountStatus
Active
Inactive

i. **Explanation**: This query retrieves all **unique account statuses** (e.g., 'Active', 'Dormant', 'Closed') from the **Account** table.

4. Updating Records

4.1. Write an SQL query to update a customer's phone number.

- **OUTPUT**



CustomerID	CustomerType	LastName	FirstName	DateOfBirth	Email	PhoneNumber	Address
1	Regular	Santos	Juan	1985-03-15	juan.santos@example.com	09181234568	Makati City

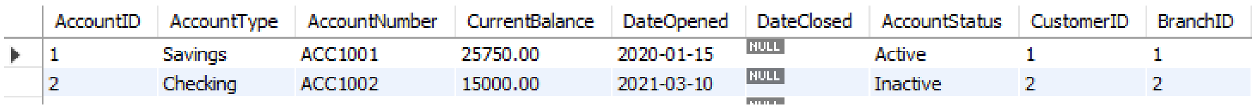
- **SQL QUERY**

```
-- 4.1: Update a customer's phone number
UPDATE Customer
SET PhoneNumber = '09181234568'   -- New phone number
WHERE CustomerID = 1;  -- Specify the CustomerID of the customer whose phone number you want to update
```

- **EXPLANATION**
  - UPDATE Customer → tells SQL you're changing data in the Customer table.
  - SET PhoneNumber = '09181234568' → updates the phone number to the new value.
  - WHERE CustomerID = 1 → ensures only the customer with ID 1 is affected.

4.2. Write an SQL query to change the account type active or inactive.

- **OUTPUT**



AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
1	Savings	ACC1001	25750.00	2020-01-15	NULL	Active	1	1
2	Checking	ACC1002	15000.00	2021-03-10	NULL	Inactive	2	2

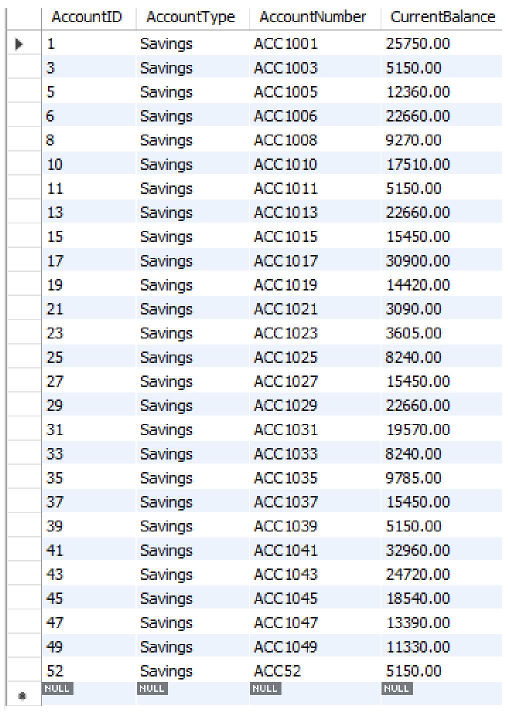
- **SQL Query**

```sql
-- 4.2: Change the account status to 'Active' or 'Inactive'
UPDATE Account
SET AccountStatus = 'Inactive'
WHERE AccountID = 2;  -- Specify the AccountID of the account you want to update
```

- **EXPLANATION**
  - UPDATE Account → targets the Account table.
  - SET AccountStatus = 'Inactive' → changes the status field.
  - WHERE AccountID = 2 → applies the change only to the account with ID 2.
    - This structure ensures only one specific account is updated, without affecting others.

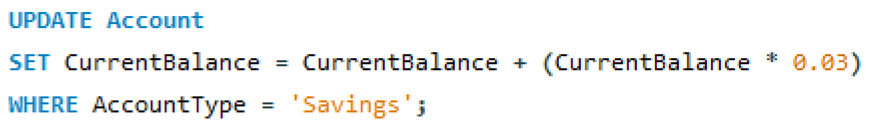
4.3. Write an SQL query to add interest to all savings accounts (e.g., 3% of the current balance).

- **OUTPUT**



AccountID	AccountType	AccountNumber	CurrentBalance
1	Savings	ACC1001	25750.00
3	Savings	ACC1003	5150.00
5	Savings	ACC1005	12360.00
6	Savings	ACC1006	22660.00
8	Savings	ACC1008	9270.00
10	Savings	ACC1010	17510.00
11	Savings	ACC1011	5150.00
13	Savings	ACC1013	22660.00
15	Savings	ACC1015	15450.00
17	Savings	ACC1017	30900.00
19	Savings	ACC1019	14420.00
21	Savings	ACC1021	3090.00
23	Savings	ACC1023	3605.00
25	Savings	ACC1025	8240.00
27	Savings	ACC1027	15450.00
29	Savings	ACC1029	22660.00
31	Savings	ACC1031	19570.00
33	Savings	ACC1033	8240.00
35	Savings	ACC1035	9785.00
37	Savings	ACC1037	15450.00
39	Savings	ACC1039	5150.00
41	Savings	ACC1041	32960.00
43	Savings	ACC1043	24720.00
45	Savings	ACC1045	18540.00
47	Savings	ACC1047	13390.00
49	Savings	ACC1049	11330.00
52	Savings	ACC52	5150.00
NULL	NULL	NULL	NULL

- **SQL Query**

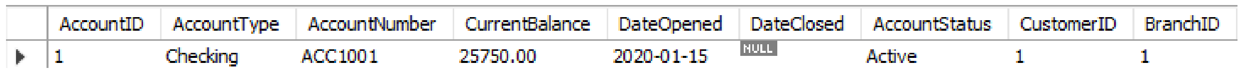


```
UPDATE Account
SET CurrentBalance = CurrentBalance + (CurrentBalance * 0.03)
WHERE AccountType = 'Savings';
```

- **EXPLANATION**
  - UPDATE Account → targets the Account table.
  - SET CurrentBalance = CurrentBalance * 0.03 → increases the balance by 3%.
  - WHERE AccountType = 'Savings' → applies the update only to savings accounts.
  - AND AccountID IS NOT NULL → ensures only valid accounts are updated.
    - This structure is useful for applying bulk updates based on account type

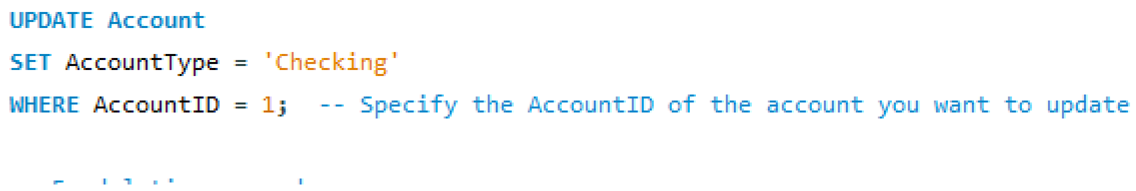
4.4. Write an SQL query to change the account type for a specific account.

- **OUTPUT**



	AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
▶	1	Checking	ACC1001	25750.00	2020-01-15	NULL	Active	1	1

- **SQL Query**



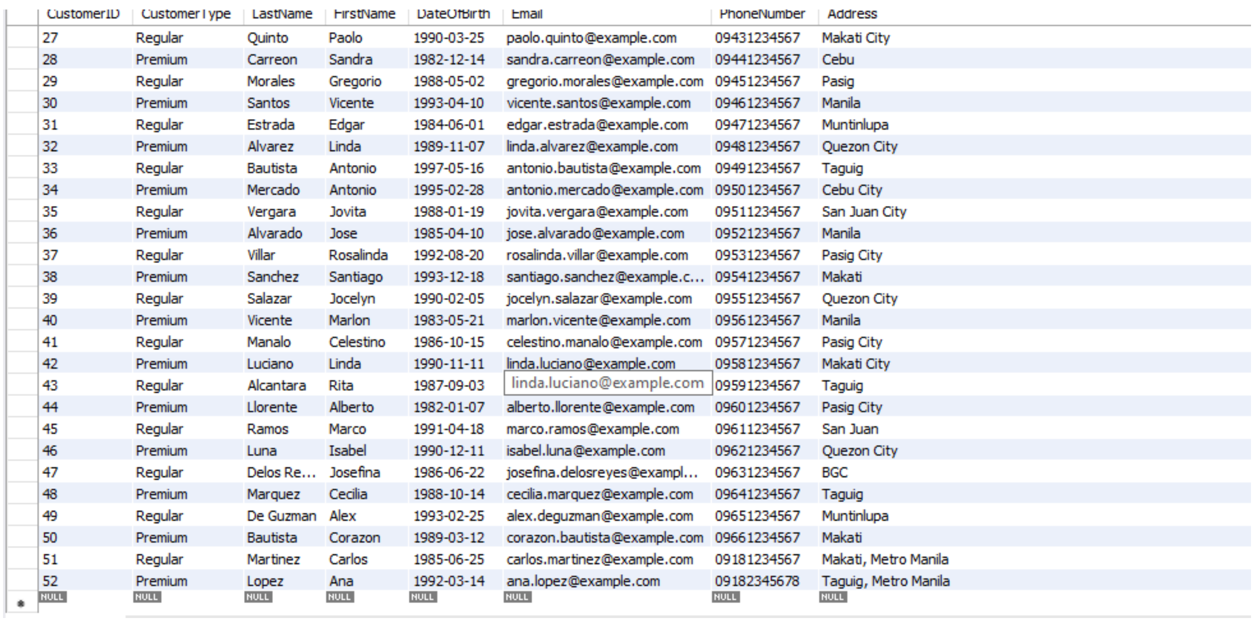
```
UPDATE Account
SET AccountType = 'Checking'
WHERE AccountID = 1;   -- Specify the AccountID of the account you want to update
```

- **EXPLANATION**
  - UPDATE Account → targets the Account table.
  - SET AccountType = 'Checking' → changes the account type to 'Checking'.
  - WHERE AccountID =1 → ensures only the account with ID 1 is updated.
    - This structure allows precise updates without affecting other records.

5. Deleting Records

5.1. Write an SQL query to delete a specific customer record based on CustomerID.

- **OUTPUT**



CustomerID	CustomerType	LastName	FirstName	DateOfBirth	Email	PhoneNumber	Address
27	Regular	Quinto	Paolo	1990-03-25	paolo.quinto@example.com	09431234567	Makati City
28	Premium	Carreon	Sandra	1982-12-14	sandra.carreon@example.com	09441234567	Cebu
29	Regular	Morales	Gregorio	1988-05-02	gregorio.morales@example.com	09451234567	Pasig
30	Premium	Santos	Vicente	1993-04-10	vicente.santos@example.com	09461234567	Manila
31	Regular	Estrada	Edgar	1984-06-01	edgar.estrada@example.com	09471234567	Muntinlupa
32	Premium	Alvarez	Linda	1989-11-07	linda.alvarez@example.com	09481234567	Quezon City
33	Regular	Bautista	Antonio	1997-05-16	antonio.bautista@example.com	09491234567	Taguig
34	Premium	Mercado	Antonio	1995-02-28	antonio.mercado@example.com	09501234567	Cebu City
35	Regular	Vergara	Jovita	1988-01-19	jovita.vergara@example.com	09511234567	San Juan City
36	Premium	Alvarado	Jose	1985-04-10	jose.alvarado@example.com	09521234567	Manila
37	Regular	Villar	Rosalinda	1992-08-20	rosalinda.villar@example.com	09531234567	Pasig City
38	Premium	Sanchez	Santiago	1993-12-18	santiago.sanchez@example.c...	09541234567	Makati
39	Regular	Salazar	Jocelyn	1990-02-05	jocelyn.salazar@example.com	09551234567	Quezon City
40	Premium	Vicente	Marlon	1983-05-21	marlon.vicente@example.com	09561234567	Manila
41	Regular	Manalo	Celestino	1986-10-15	celestino.manalo@example.com	09571234567	Pasig City
42	Premium	Luciano	Linda	1990-11-11	linda.luciano@example.com	09581234567	Makati City
43	Regular	Alcantara	Rita	1987-09-03	linda.luciano@example.com	09591234567	Taguig
44	Premium	Llorente	Alberto	1982-01-07	alberto.llorente@example.com	09601234567	Pasig City
45	Regular	Ramos	Marco	1991-04-18	marco.ramos@example.com	09611234567	San Juan
46	Premium	Luna	Isabel	1990-12-11	isabel.luna@example.com	09621234567	Quezon City
47	Regular	Delos Re...	Josefina	1986-06-22	josefina.delosreyes@exampl...	09631234567	BGC
48	Premium	Marquez	Cecilia	1988-10-14	cecilia.marquez@example.com	09641234567	Taguig
49	Regular	De Guzman	Alex	1993-02-25	alex.deguzman@example.com	09651234567	Muntinlupa
50	Premium	Bautista	Corazon	1989-03-12	corazon.bautista@example.com	09661234567	Makati
51	Regular	Martinez	Carlos	1985-06-25	carlos.martinez@example.com	09181234567	Makati, Metro Manila
52	Premium	Lopez	Ana	1992-03-14	ana.lopez@example.com	09182345678	Taguig, Metro Manila
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

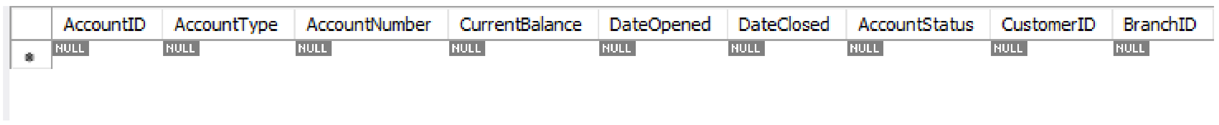
- **SQL Query**

```sql
DELETE FROM Customer
WHERE CustomerID = 53;
```

- EXPLANATION
  - DELETE FROM Customer → targets the Customer table.
  - WHERE CustomerID = 53→ deletes the customer record itself.

5.2. Write an SQL query to delete all accounts with a balance of zero.

- **OUTPUT(VERIFICATION THAT ALL ACCOUNT WITH A BALANCE OF ZERO IS DELETED)**



AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- **SQL Query**

```sql
-- 5.2. Write an SQL query to delete all accounts with a balance of zero.
DELETE FROM Account
WHERE CurrentBalance = 0;
SELECT *
FROM Account
WHERE CurrentBalance = 0;
```

- EXPLANATION
  - SELECT * FROM help view the Account table by selecting it. WHERE will filter out all account with zero balance.
  - DELETE FROM Account → targets the Account table.
  - WHERE CurrentBalance <= 0 → filters for accounts with zero or negative balance.
    - This ensures only financially inactive or invalid accounts are removed.

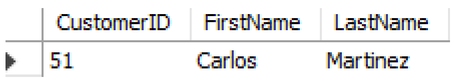
6. Advanced Scenario

6.1. Write an SQL query to find customers who do not have an account yet.

- SQL query

```sql
-- 6.1. Write an SQL query to find customers who do not have an account yet.
SELECT c.CustomerID, c.FirstName, c.LastName
FROM Customer c
LEFT JOIN Account a ON c.CustomerID = a.CustomerID
WHERE a.AccountID IS NULL;
```

- Output



CustomerID	FirstName	LastName
51	Carlos	Martinez

- EXPLANATION
  - **LEFT JOIN Account a ON c.CustomerID = a.CustomerID**: This ensures that all customers are included, even if they don't have an account.

   - **WHERE a.AccountID IS NULL**: Filters the result to include only customers who do not have an associated account (AccountID is NULL).

6.2. Write an SQL query to delete a customer and all associated accounts in one operation.

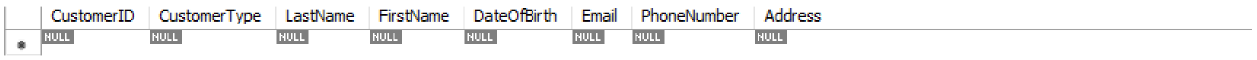
- SQL QUERY

```sql
-- 6.2. Write an SQL query to delete a customer and all associated accounts in one operation.
DELETE FROM account WHERE CustomerID = 52;
DELETE FROM customer WHERE CustomerID = 52;
```

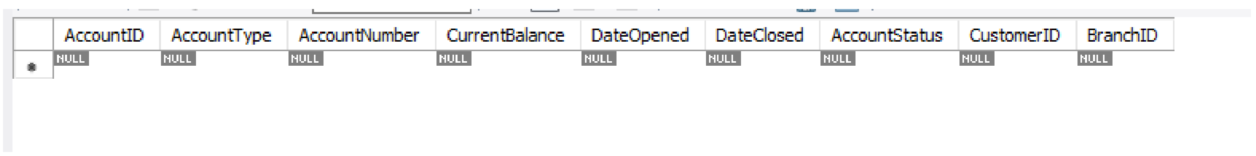
   - **First line**: Deletes all rows in the account table where CustomerID is 52. This removes every account linked to that customer.
   - **Second line**: Deletes the customer record from the customer table where CustomerID is 52.
- DELETE ON CASCADE

```sql
ALTER TABLE account
DROP FOREIGN KEY account_ibfk_1,
ADD CONSTRAINT account_ibfk_1
FOREIGN KEY (CustomerID) REFERENCES customer(CustomerID)
ON DELETE CASCADE;
```

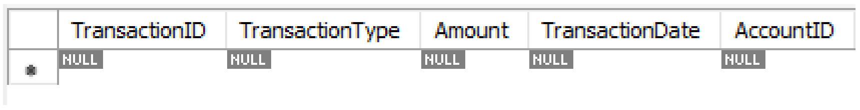
- EXPLANATION
   - With **ON DELETE CASCADE** enabled, deleting the **customer** automatically deletes all accounts associated with that customer.
- OUTPUT(VERIFICATION THAT THE CUSTOMER AND ALL ASSOCIATED ACCOUNTS ARE DELETED IN ONE OPERATION)



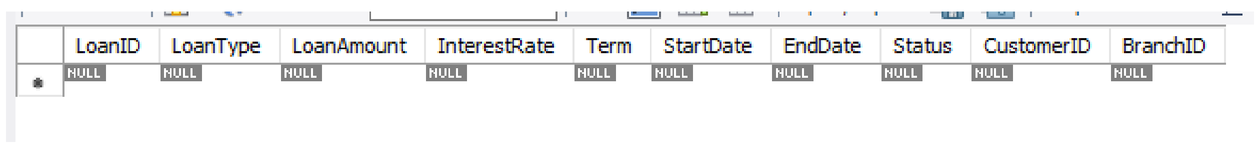
	CustomerID	CustomerType	LastName	FirstName	DateOfBirth	Email	PhoneNumber	Address
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



	AccountID	AccountType	AccountNumber	CurrentBalance	DateOpened	DateClosed	AccountStatus	CustomerID	BranchID
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



	TransactionID	TransactionType	Amount	TransactionDate	AccountID
*	NULL	NULL	NULL	NULL	NULL



	LoanID	LoanType	LoanAmount	InterestRate	Term	StartDate	EndDate	Status	CustomerID	BranchID
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

6.3. Explain the impact of deleting a customer on the Account table if a foreign key relationship exists.

If a foreign key relationship exists between the Customer table and the Account table, deleting a customer has direct consequences. Without the ON DELETE CASCADE option, the database will prevent you from deleting a customer who still has accounts, because doing so would leave orphaned records in the Account table. In this case, you must delete the accounts first before you can remove the customer. With ON DELETE CASCADE enabled, however, deleting a customer will automatically remove all of their associated accounts at the same time. This ensures referential integrity is maintained, but it also means that a single delete operation can wipe out both the customer and all linked financial records.

7. Conclusion and Reflection

Reflection

Working on the Banking Database Management System project provided valuable insights into the practical application of database design principles. By identifying entities such as **Customer, Account, Transaction, Loan, and Branch**, I learned how to translate real-world banking operations into a structured relational schema. The process of defining **primary keys, foreign keys, and relationships** reinforced the importance of **referential integrity** and how it ensures consistency across interconnected tables.

Implementing CRUD operations allowed me to experience the full cycle of database management—from creating records to updating and deleting them. I realized that even small design decisions, such as choosing appropriate data types or enforcing constraints, have a significant impact on the reliability and scalability of the system. Troubleshooting errors, particularly with foreign key constraints, taught me patience and precision in debugging, which are essential skills for any IT professional.

Beyond the technical aspects, this project emphasized the importance of **business realism** in database design. A banking system is not just about storing data; it must reflect real-world workflows such as account management, loan processing, and transaction tracking. This strengthened my ability to think critically about how technology supports organizational needs and customer interactions.

Conclusion

The Banking Database Management System case study successfully demonstrated how relational databases can be used to model and manage the core operations of a financial institution. Through the creation of normalized tables, enforcement of relationships, and execution of CRUD operations, the project achieved its goal of building a system that is both **efficient and reliable**.

The final schema ensures that customer information, accounts, transactions, and loans are properly linked, enabling accurate reporting and streamlined operations. By incorporating **logical, physical, and conceptual ERDs**, the design provides a clear blueprint for implementation and future scalability. This project highlighted the critical role of databases in supporting day-to-day banking activities and maintaining trust through data integrity.

Ultimately, the experience reinforced my skills in **SQL scripting, schema design, and scenario-driven data modeling**, while also preparing me to handle more complex, real-world systems. It showed that database management is not only a technical exercise but also a way to bridge technology with business processes, ensuring that institutions can serve their clients effectively and securely.

REFERENCES

GeeksforGeeks. (2025, July 15). *ER diagram of Bank Management System*. GeeksforGeeks. https://www.geeksforgeeks.org/dbms/er-diagram-of-bank-management-system/

Redgate Data Modeler. (2023, June 27). *Creating a database design for a banking system*. Redgate Blog. https://www.red-gate.com/blog/database-design-for-banking-system

Oldroyd, H. (2019, July 30). *A guide to the Entity Relationship Diagram (ERD)*. Database Star. https://www.databasestar.com/entity-relationship-diagram/