A Database Application for Fintech Holdings Incorporated

Group 5: The Primary Keys

Abel Deulofeu, Anirudh Naram Reddy, David Heller, Khushi Sarin, Shikha Shikha

ISM6205: Database Management

Dr. Arijit Sengupta

April 21, 2024

**Team Members and Roles:**

1. Abel Deulofeu as SQL Guru
2. David Heller as Research Lead
3. Anirudh Naram Reddy as Tester
4. Khushi Sarin as Team Lead
5. Shikha Shikha as Design Lead

**Project Description**

Fintech Holdings Incorporated has set a goal of developing an integrated database system that can facilitate account handling and customer data management. In this age where digital technologies dominate all industries, security and efficiency in dealing with financial data are crucial to both clients and financial institutions. The present proposal is dedicated to solving the problem of lacking a reliable solution that would allow encoding customer information in a secure way while providing bank administrators with access to user statuses and important transaction information. The system will be equipped with several modules that enable different banking functions to occur, such as account opening and administration, transaction processing, tracking account details, and client assistance. Users can securely open new accounts, make deposits and withdrawals, transfer funds, and finally investigate the transaction history using a user-friendly interface. Additionally, the system will incorporate advanced security measures to safeguard sensitive financial data, including encryption protocols and multi-factor authentication.

For banking administrators, the system will provide comprehensive insights into customer accounts, allowing them to monitor user activities, track transaction trends, and identify potential fraud or suspicious activities. Moreover, administrators can manage user accounts, update them, and generate reports for auditing and regulatory compliance purposes. Overall, Fintech Holdings Incorporated aims to streamline banking operations, enhance customer experience, and ensure the security and integrity of financial data. By leveraging modern database technologies and adhering to industry best practices, this project seeks to provide a sustainable solution for managing bank accounts in the digital era.

**Features and Requirements**

* Customer Information Management: Safely store customer information and maintain a record of customer accounts.
* Account management: Create, modify, and delete accounts for various banking products and the ability to track account balances and transaction history.
* Transaction Processing: Record and manage transactions including deposits, withdrawals, transfers, loans, and ATM transactions.
* Reporting and analysis: Generate various reports such as financial reports and audit reports.
* Customer Relationship Management: Helps Fintech manage interactions with current customers by tracking transaction frequency and preferences.
* ATM management: Manage ATM machines available on location, including cash levels.
* Employee Management: Manage employee details, jobs, roles, and work schedules.
* Mobile Banking: Allows customers to manage transactions and accounts through the ease of their phone.
* Loan Management: Handles customer loans and loan payments.
* Transfers: Tracks transfer transactions between internal customer accounts.

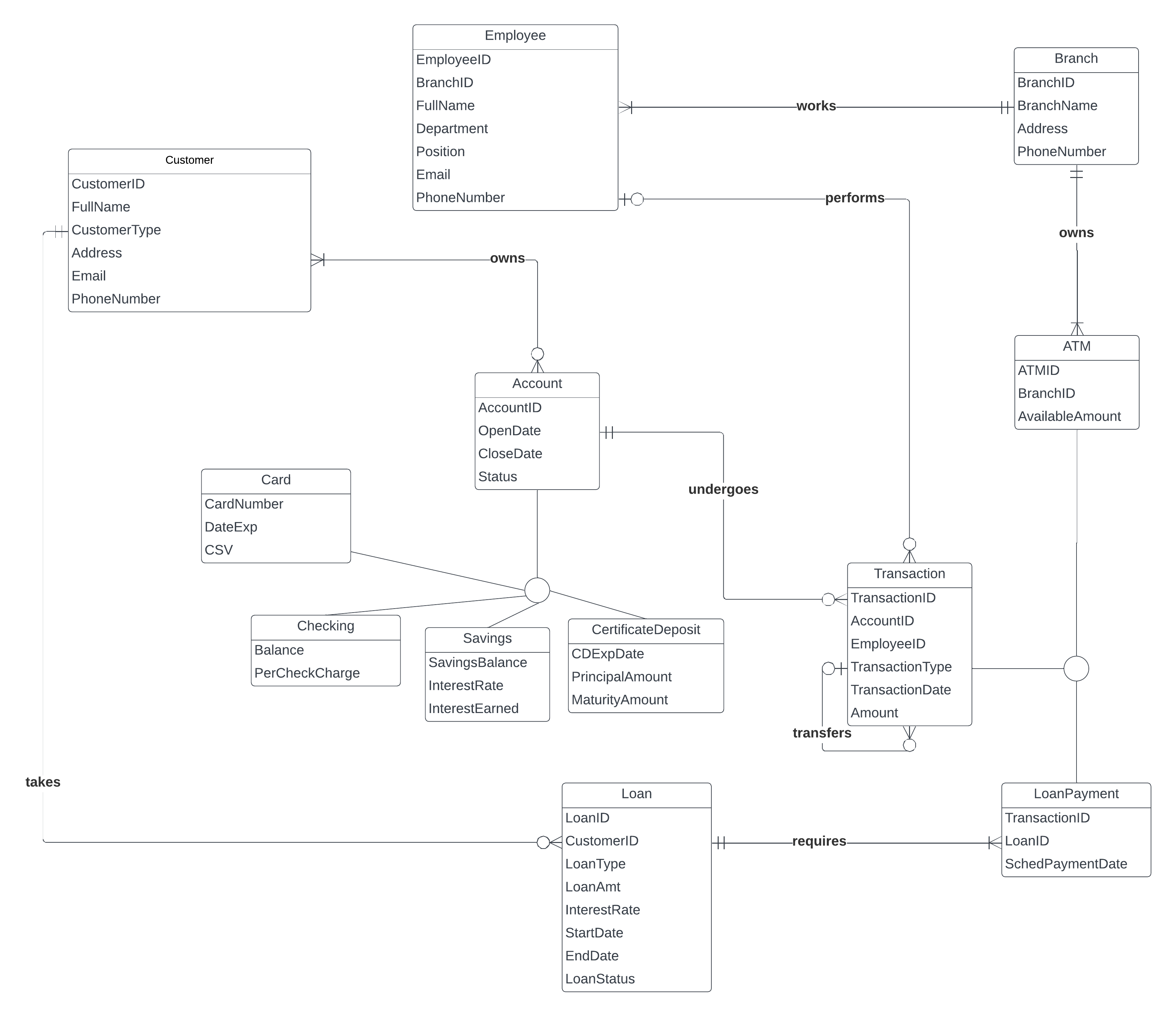
**Business rules.**

FinTech Holdings Incorporated has employees at branches across the country from which it serves customers. A customer may have multiple accounts with the bank, and multiple cards associate with said account(s). An account can also be accessed by multiple customers. Customers can choose from four account types: card, checking, savings, and certification deposit. ATMs at branches and loans are also considered transactions. Each transaction is associated with an account.

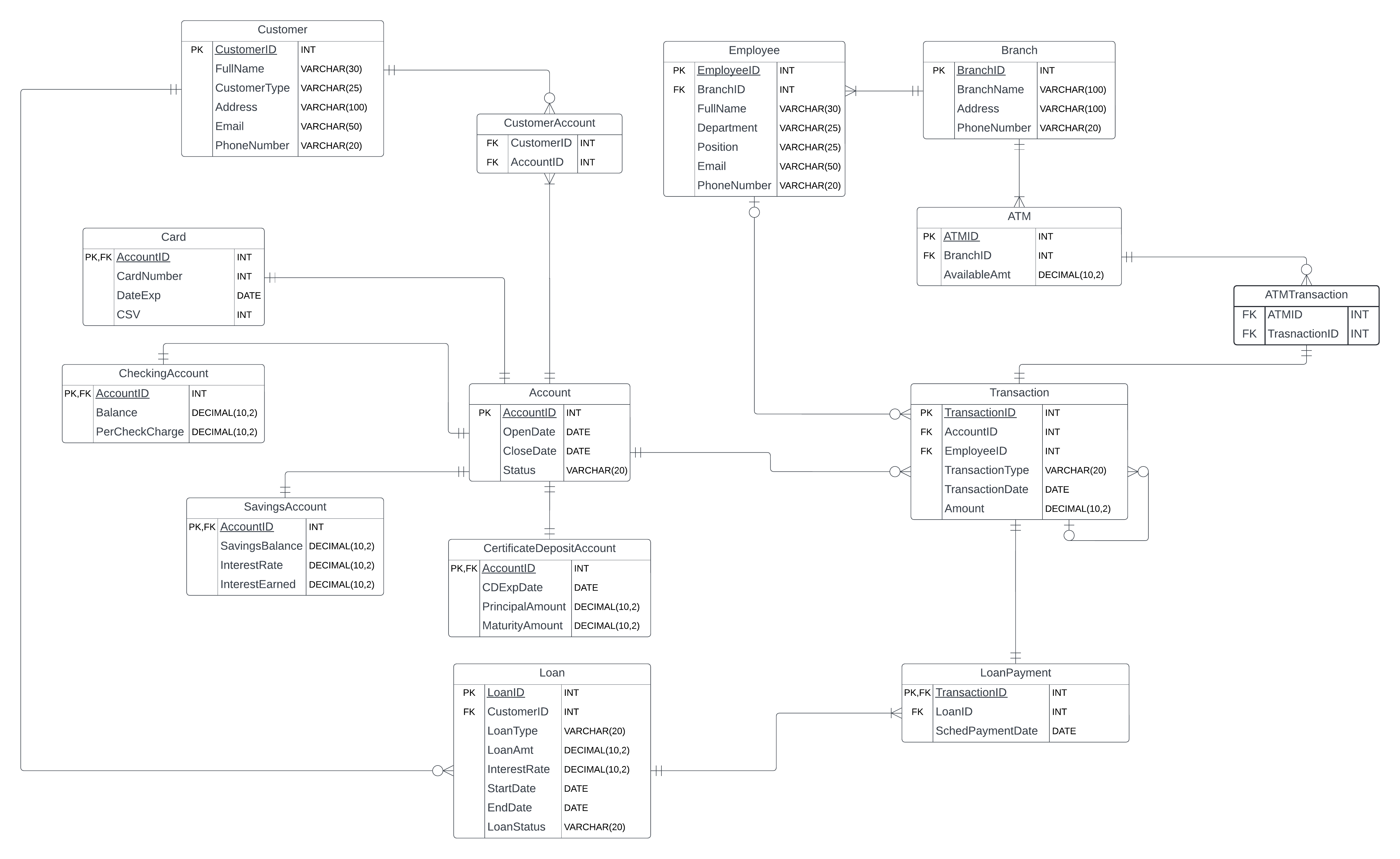
* Every branch has at least one employee, and every employee must work at only one branch.
* Every branch has at least one ATM, and an ATM must be located at only one branch.
* A customer might have more than one account, and every account must belong to at least one customer.
* An account might be a card, checking account, savings account, or certificate of deposit account. Every card must be an account. Every checking account must be an account. Every savings account must be an account. Every certificate of deposit account must be an account.
* An account might have more than one transaction, and a transaction can only be made for one account.
* A customer might have more than one loan, and a loan is granted to only one customer.
* A loan must have at least one scheduled payment, and a payment must be made to only one loan.
* A loan payment might be completed in many transactions, and a transaction might be made for one loan payment.
* An employee might perform more than one transaction, and a transaction might be completed by only one employee.
* A transaction might trigger other transactions, a transaction might be triggered by only one other transaction.
* A transaction can be through an account, ATM transaction, or loan payment.

**Draft Models**

For accessibility, the models are also available in PDF format and live links.



*Figure 1: ER Diagram*

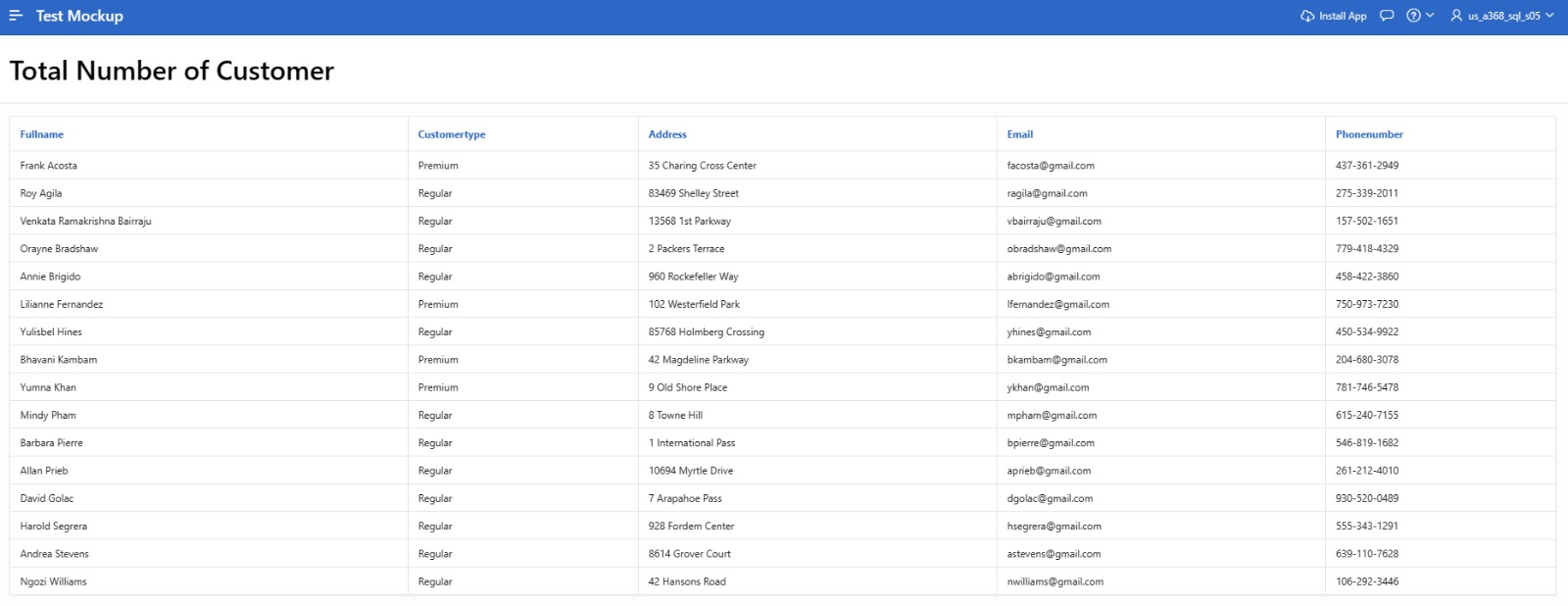


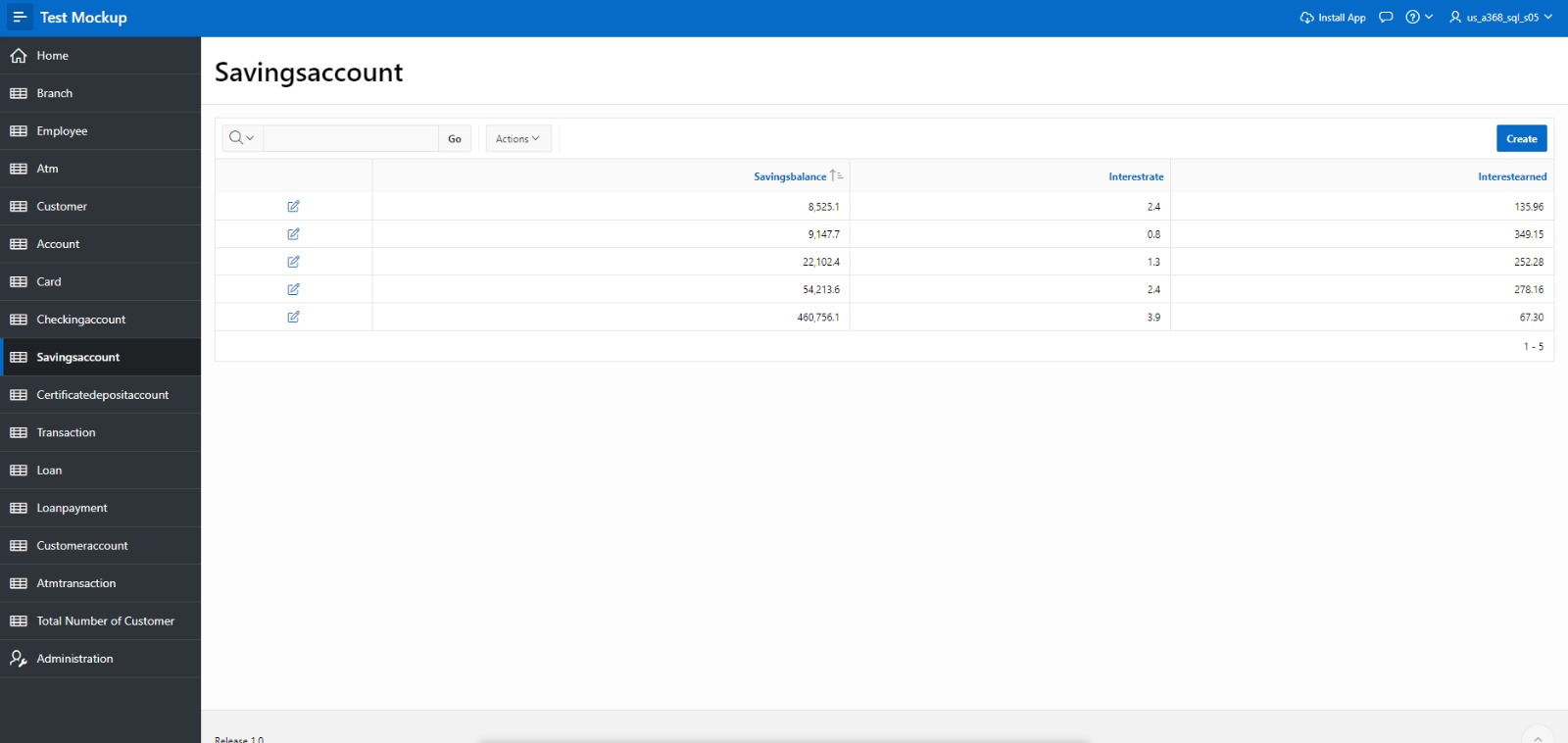
*Figure 2: Schema Diagram*

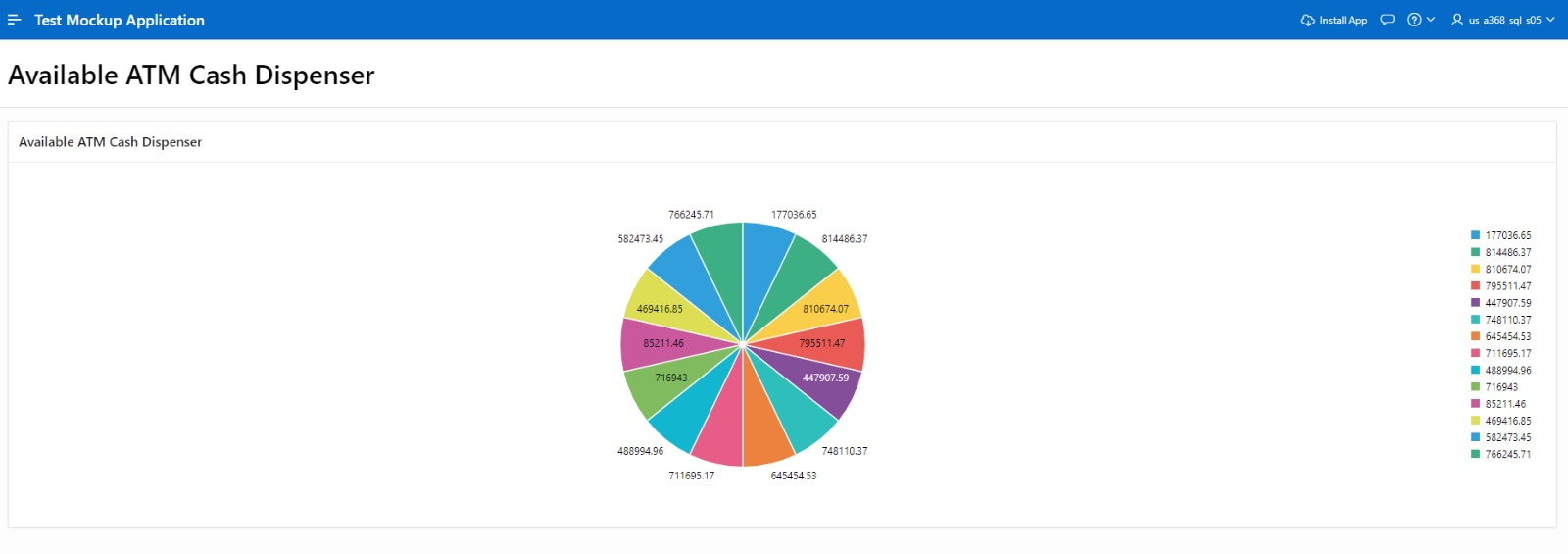
**Use Cases**

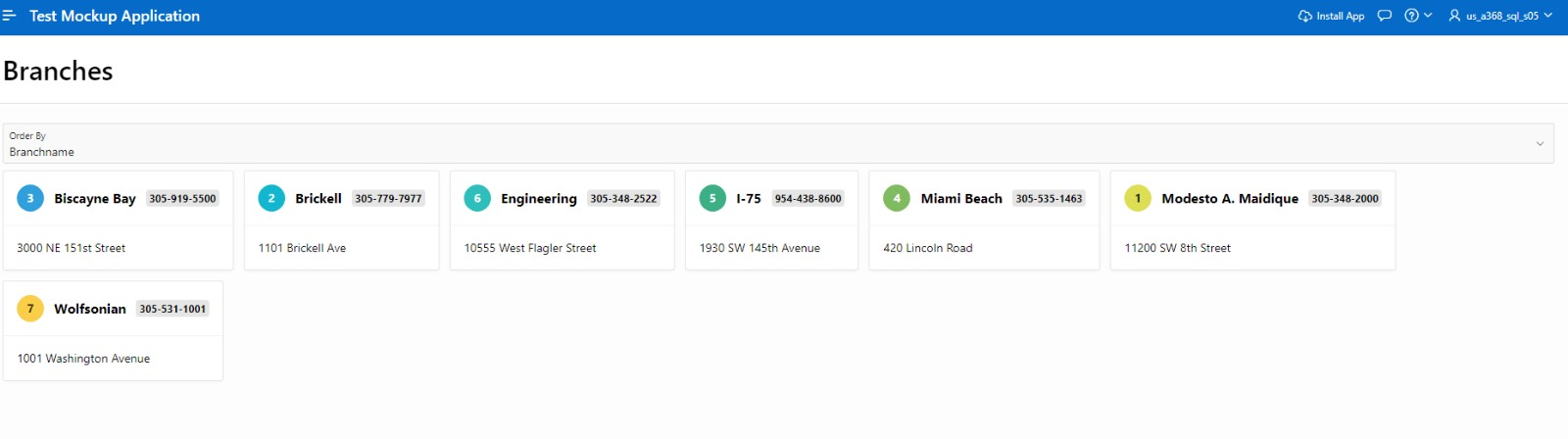
1. The owner of one of the branches believes there was a fraudulent transaction on January 30, 2024. They need the customer’s name, phone number, and account ID to contact the customer and confirm the transaction.
2. Branches 1-3 need to notify their employees that the ATMs are down temporarily. The ATM IDs and available amounts are needed, as are the employee names and phone numbers for those in branches 1-3.
3. On March 13, Ngozi Williams transferred $150 dollars from her checking account with ID 106 to her account with ID 111.
4. Employee 1009 helps Annie Brigido submit her $1,000 loan payment for her personal loan for $10,000 and it is noted as transaction 6666.

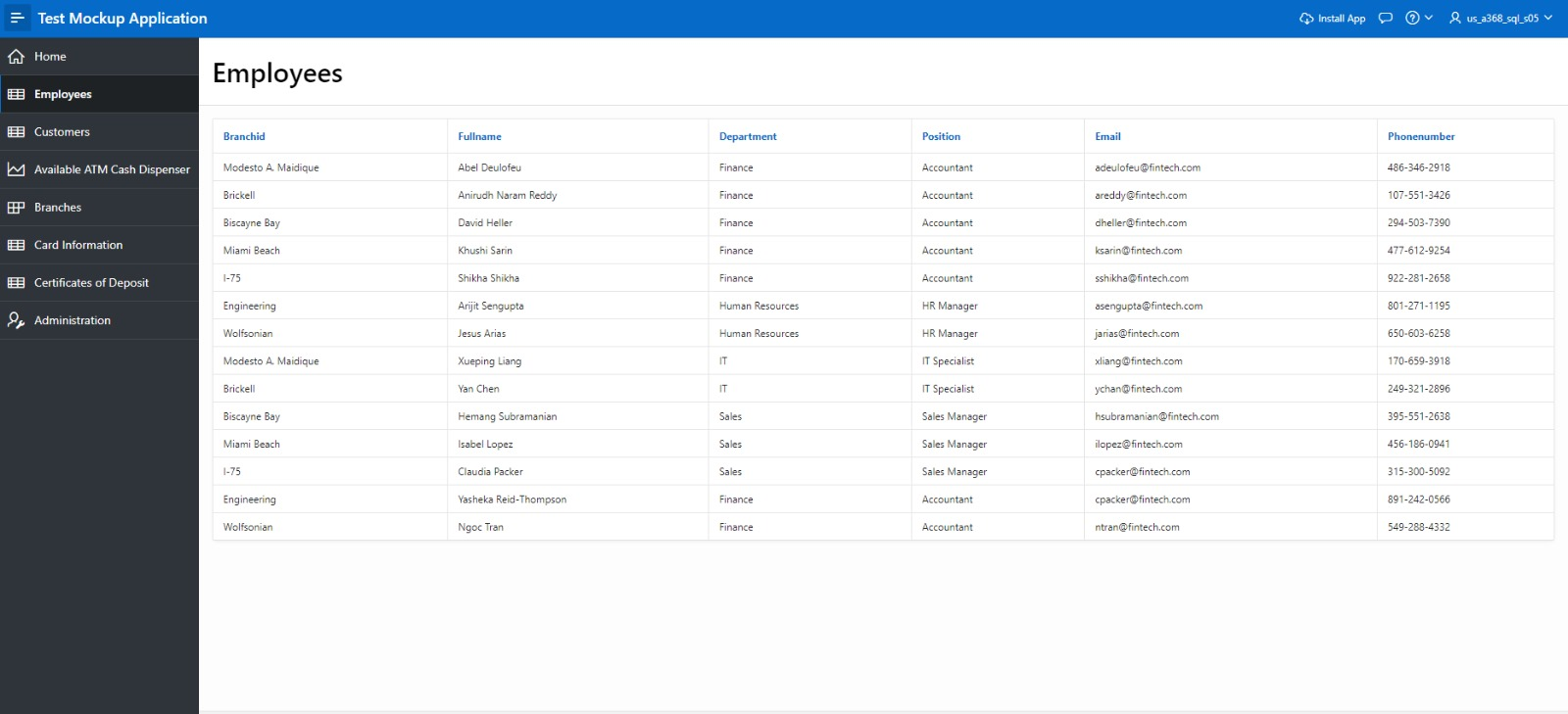
**Application Designs**

*Image 1: A mock-up of the application showing all customers and information*

*Image 2: A mock-up of the application showing all savings accounts and information*

*Image 3: A mock-up showing all available cash in all ATMs*

*Image 4: A mock-up showing all employees*



*Image 5: A mock-up showing all employees*

**Conclusion**

Team Primary Keys learned a lot regarding the relationship between data, creation of databases, and the intricacies that go into querying relevant data from said databases. It requires meticulous planning prior to execution and is everchanging even during execution. The created database will serve well as a starting point for this fictional bank’s application needs, but clearly needs much additional work to ensure proper management of vast amounts of data regarding customers, accounts, loans, and employees.

The data used for testing this database was small and the actual application would require indexing to ensure efficient operations. A change that the team would make to the database would be editing the self-referential transaction to direct to a referential entity that would allow the second transaction to reference the initial transaction. Additionally, the loanpayment entity should have been a sub entity for account. This would make it easier to join loans to transactions when looking at all account transactions.

After learning table creation, altering, and the basics of querying, the team feels that they are better prepared to understand and deal with the practical applications of data and databases. Overall, the team came away from this project with a greater understanding of the critical role that databases play in nearly every aspect of life in today’s data-driven world.

**Appendix**

**Complete Create Tables Script**

-- drop tables

DROP TABLE atmtransaction;

DROP TABLE customeraccount;

DROP TABLE loanpayment;

DROP TABLE loan;

DROP TABLE transaction;

DROP TABLE certificatedepositaccount;

DROP TABLE savingsaccount;

DROP TABLE checkingaccount;

DROP TABLE card;

DROP TABLE account;

DROP TABLE customer;

DROP TABLE atm;

DROP TABLE employee;

DROP TABLE branch;

-- branch table

CREATE TABLE branch(

branchid INT PRIMARY KEY,

branchname VARCHAR2(100),

address VARCHAR2(100),

phonenumber VARCHAR2(20));

-- employee table

CREATE TABLE employee(

employeeid INT PRIMARY KEY,

branchid INT,

fullname VARCHAR2(30),

department VARCHAR2(25),

position VARCHAR2(25),

email VARCHAR2(50),

phonenumber VARCHAR(20));

ALTER TABLE employee

ADD CONSTRAINT employee\_fk

FOREIGN KEY(branchid)

REFERENCES branch(branchid)

ON DELETE SET NULL;

-- ATM

CREATE TABLE atm(

atmid INT PRIMARY KEY,

branchid INT,

availableamt DECIMAL(10,2));

ALTER TABLE atm

ADD CONSTRAINT atm\_fk

FOREIGN KEY(branchid)

REFERENCES branch(branchid)

ON DELETE CASCADE;

-- customer table

CREATE TABLE customer(

customerid INT PRIMARY KEY,

fullname VARCHAR2(30) NOT NULL,

customertype VARCHAR2(25),

address VARCHAR2(100),

email VARCHAR2(50),

phonenumber VARCHAR2(20));

-- account table

CREATE TABLE account(

accountid INT PRIMARY KEY,

opendate DATE,

closedate DATE,

status VARCHAR2(20));

-- card table

CREATE TABLE card(

accountid INT,

cardnumber INT,

dateexp DATE,

CSV INT);

ALTER TABLE card

ADD CONSTRAINT card\_pk

PRIMARY KEY(accountid);

ALTER TABLE card

ADD CONSTRAINT card\_fk

FOREIGN KEY(accountid)

REFERENCES account(accountid)

ON DELETE CASCADE;

-- checking account table

CREATE TABLE checkingaccount(

accountid INT,

balance DECIMAL(10,2),

percheckcharge DECIMAL(10,2));

ALTER TABLE checkingaccount

ADD CONSTRAINT checkingaccount\_pk

PRIMARY KEY(accountid);

ALTER TABLE checkingaccount

ADD CONSTRAINT checkingaccount\_fk

FOREIGN KEY(accountid)

REFERENCES account(accountid)

ON DELETE CASCADE;

-- savings account table

CREATE TABLE savingsaccount(

accountid INT,

savingsbalance DECIMAL(10,2),

interestrate DECIMAL(10,2),

interestearned DECIMAL(10,2));

ALTER TABLE savingsaccount

ADD CONSTRAINT savingsaccount\_pk

PRIMARY KEY(accountid);

ALTER TABLE savingsaccount

ADD CONSTRAINT savingsaccount\_fk

FOREIGN KEY(accountid)

REFERENCES account(accountid)

ON DELETE CASCADE;

-- certificatedepositaccount table

CREATE TABLE certificatedepositaccount(

accountid INT,

cdexpdate DATE,

principalamount DECIMAL(10,2),

maturityamount DECIMAL(10,2));

ALTER TABLE certificatedepositaccount

ADD CONSTRAINT cd\_pk

PRIMARY KEY(accountid);

ALTER TABLE certificatedepositaccount

ADD CONSTRAINT cd\_fk

FOREIGN KEY(accountid)

REFERENCES account(accountid)

ON DELETE CASCADE;

-- transaction table

CREATE TABLE transaction(

transactionid INT PRIMARY KEY,

accountid INT,

employeeid INT,

transactiontype VARCHAR2(20),

transactiondate DATE,

amount DECIMAL(10,2));

ALTER TABLE transaction

ADD CONSTRAINT transaction\_fk1

FOREIGN KEY(accountid)

REFERENCES account(accountid)

ON DELETE SET NULL;

ALTER TABLE transaction

ADD CONSTRAINT transaction\_fk2

FOREIGN KEY(employeeid)

REFERENCES employee(employeeid)

ON DELETE SET NULL;

-- loan table

CREATE TABLE loan(

loanid INT PRIMARY KEY,

customerid INT,

loantype VARCHAR2(20),

loanamt DECIMAL(10,2),

interestrate DECIMAL(10,2),

startdate DATE,

enddate DATE,

loanstatus VARCHAR2(20));

ALTER TABLE loan

ADD CONSTRAINT loan\_fk

FOREIGN KEY(customerid)

REFERENCES customer(customerid)

ON DELETE SET NULL;

-- loanpayment table

CREATE TABLE loanpayment(

transactionid INT,

loanid INT,

schedpaymentdate DATE);

ALTER TABLE loanpayment

ADD CONSTRAINT loanpayment\_pk

PRIMARY KEY(transactionid);

ALTER TABLE loanpayment

ADD CONSTRAINT loanpayment\_fk1

FOREIGN KEY(transactionid)

REFERENCES transaction(transactionid)

ON DELETE CASCADE;

ALTER TABLE loanpayment

ADD CONSTRAINT loanpayment\_fk2

FOREIGN KEY(loanid)

REFERENCES loan(loanid)

ON DELETE CASCADE;

-- customeraccount table

CREATE TABLE customeraccount(

customerid INT,

accountid INT,

FOREIGN KEY(customerid) REFERENCES customer(customerid),

FOREIGN KEY(accountid) REFERENCES account(accountid));

-- atmtransaction table

CREATE TABLE atmtransaction(

atmid INT,

transactionid INT,

FOREIGN KEY(atmid) REFERENCES atm(atmid),

FOREIGN KEY(transactionid) REFERENCES transaction(transactionid));

**Complete Insert Script**

-- INSERT STATEMENTS

-- branch

INSERT INTO branch VALUES(1, 'Modesto A. Maidique', '11200 SW 8th Street', '305-348-2000');

INSERT INTO branch VALUES(2, 'Brickell', '1101 Brickell Ave', '305-779-7977');

INSERT INTO branch VALUES(3, 'Biscayne Bay', '3000 NE 151st Street', '305-919-5500');

INSERT INTO branch VALUES(4, 'Miami Beach', '420 Lincoln Road', '305-535-1463');

INSERT INTO branch VALUES(5, 'I-75', '1930 SW 145th Avenue', '954-438-8600');

INSERT INTO branch VALUES(6, 'Engineering', '10555 West Flagler Street', '305-348-2522');

INSERT INTO branch VALUES(7, 'Wolfsonian', '1001 Washington Avenue', '305-531-1001');

-- employee

INSERT INTO employee VALUES(1001, 1, 'Abel Deulofeu', 'Finance', 'Accountant', 'adeulofeu@fintech.com', '486-346-2918');

INSERT INTO employee VALUES(1002, 2, 'Anirudh Naram Reddy', 'Finance', 'Accountant', 'areddy@fintech.com', '107-551-3426');

INSERT INTO employee VALUES(1003, 3, 'David Heller', 'Finance', 'Accountant', 'dheller@fintech.com', '294-503-7390');

INSERT INTO employee VALUES(1004, 4, 'Khushi Sarin', 'Finance', 'Accountant', 'ksarin@fintech.com', '477-612-9254');

INSERT INTO employee VALUES(1005, 5, 'Shikha Shikha', 'Finance', 'Accountant', 'sshikha@fintech.com', '922-281-2658');

INSERT INTO employee VALUES(1006, 6, 'Arijit Sengupta', 'Human Resources', 'HR Manager', 'asengupta@fintech.com', '801-271-1195');

INSERT INTO employee VALUES(1007, 7, 'Jesus Arias', 'Human Resources', 'HR Manager', 'jarias@fintech.com', '650-603-6258');

INSERT INTO employee VALUES(1008, 1, 'Xueping Liang', 'IT', 'IT Specialist', 'xliang@fintech.com', '170-659-3918');

INSERT INTO employee VALUES(1009, 2, 'Yan Chen', 'IT', 'IT Specialist', 'ychan@fintech.com', '249-321-2896');

INSERT INTO employee VALUES(1010, 3, 'Hemang Subramanian', 'Sales', 'Sales Manager', 'hsubramanian@fintech.com', '395-551-2638');

INSERT INTO employee VALUES(1011, 4, 'Isabel Lopez', 'Sales', 'Sales Manager', 'ilopez@fintech.com', '456-186-0941');

INSERT INTO employee VALUES(1012, 5, 'Claudia Packer', 'Sales', 'Sales Manager', 'cpacker@fintech.com', '315-300-5092');

INSERT INTO employee VALUES(1013, 6, 'Yasheka Reid-Thompson', 'Finance', 'Accountant', 'cpacker@fintech.com', '891-242-0566');

INSERT INTO employee VALUES(1014, 7, 'Ngoc Tran', 'Finance', 'Accountant', 'ntran@fintech.com', '549-288-4332');

-- atm

INSERT INTO atm VALUES(001, 1, 85211.46);

INSERT INTO atm VALUES(002, 1, 795511.47);

INSERT INTO atm VALUES(003, 2, 469416.85);

INSERT INTO atm VALUES(004, 2, 810674.07);

INSERT INTO atm VALUES(005, 3, 582473.45);

INSERT INTO atm VALUES(006, 3, 645454.53);

INSERT INTO atm VALUES(007, 4, 748110.37);

INSERT INTO atm VALUES(008, 4, 711695.17);

INSERT INTO atm VALUES(009, 5, 447907.59);

INSERT INTO atm VALUES(010, 5, 177036.65);

INSERT INTO atm VALUES(011, 6, 814486.37);

INSERT INTO atm VALUES(012, 6, 488994.96);

INSERT INTO atm VALUES(013, 7, 766245.71);

INSERT INTO atm VALUES(014, 7, 716943.00);

-- customer table

INSERT INTO customer VALUES(1, 'Frank Acosta', 'Premium', '35 Charing Cross Center', 'facosta@gmail.com', '437-361-2949');

INSERT INTO customer VALUES(2, 'Roy Agila', 'Regular', '83469 Shelley Street', 'ragila@gmail.com', '275-339-2011');

INSERT INTO customer VALUES(3, 'Venkata Ramakrishna Bairraju', 'Regular', '13568 1st Parkway', 'vbairraju@gmail.com', '157-502-1651');

INSERT INTO customer VALUES(4, 'Orayne Bradshaw', 'Regular', '2 Packers Terrace', 'obradshaw@gmail.com', '779-418-4329');

INSERT INTO customer VALUES(5, 'Annie Brigido', 'Regular', '960 Rockefeller Way', 'abrigido@gmail.com', '458-422-3860');

INSERT INTO customer VALUES(6, 'Lilianne Fernandez', 'Premium', '102 Westerfield Park', 'lfernandez@gmail.com', '750-973-7230');

INSERT INTO customer VALUES(7, 'Yulisbel Hines', 'Regular', '85768 Holmberg Crossing', 'yhines@gmail.com', '450-534-9922');

INSERT INTO customer VALUES(8, 'Bhavani Kambam', 'Premium', '42 Magdeline Parkway', 'bkambam@gmail.com', '204-680-3078');

INSERT INTO customer VALUES(9, 'Yumna Khan', 'Premium', '9 Old Shore Place', 'ykhan@gmail.com', '781-746-5478');

INSERT INTO customer VALUES(10, 'Mindy Pham', 'Regular', '8 Towne Hill', 'mpham@gmail.com', '615-240-7155');

INSERT INTO customer VALUES(11, 'Barbara Pierre', 'Regular', '1 International Pass', 'bpierre@gmail.com', '546-819-1682');

INSERT INTO customer VALUES(12, 'Allan Prieb', 'Regular', '10694 Myrtle Drive', 'aprieb@gmail.com', '261-212-4010');

INSERT INTO customer VALUES(13, 'David Golac', 'Regular', '7 Arapahoe Pass', 'dgolac@gmail.com', '930-520-0489');

INSERT INTO customer VALUES(14, 'Harold Segrera', 'Regular', '928 Fordem Center', 'hsegrera@gmail.com', '555-343-1291');

INSERT INTO customer VALUES(15, 'Andrea Stevens', 'Regular', '8614 Grover Court', 'astevens@gmail.com', '639-110-7628');

INSERT INTO customer VALUES(16, 'Ngozi Williams', 'Regular', '42 Hansons Road', 'nwilliams@gmail.com', '106-292-3446');

-- account

INSERT INTO account VALUES(101, '04-Nov-23', NULL, 'Active');

INSERT INTO account VALUES(102, '15-Apr-21', '10-Feb-24', 'Inactive');

INSERT INTO account VALUES(103, '09-Mar-24', NULL, 'Active');

INSERT INTO account VALUES(104, '11-Dec-20', NULL, 'Active');

INSERT INTO account VALUES(105, '02-Jun-21', NULL, 'Active');

INSERT INTO account VALUES(106, '20-Aug-23', NULL, 'Active');

INSERT INTO account VALUES(107, '09-May-23', NULL, 'Active');

INSERT INTO account VALUES(108, '29-Mar-24', NULL, 'Active');

INSERT INTO account VALUES(109, '22-Jun-22', NULL, 'Active');

INSERT INTO account VALUES(110, '21-Apr-22', NULL, 'Active');

INSERT INTO account VALUES(111, '31-May-21', NULL, 'Active');

INSERT INTO account VALUES(112, '03-Nov-21', NULL, 'Active');

INSERT INTO account VALUES(113, '29-Jan-20', NULL, 'Active');

INSERT INTO account VALUES(114, '18-Sep-23', NULL, 'Active');

INSERT INTO account VALUES(115, '28-Jun-21', NULL, 'Active');

INSERT INTO account VALUES(116, '14-Nov-20', NULL, 'Active');

INSERT INTO account VALUES(117, '26-Feb-22', NULL, 'Active');

INSERT INTO account VALUES(118, '15-Jul-21', NULL, 'Active');

INSERT INTO account VALUES(119, '23-Feb-23', NULL, 'Active');

INSERT INTO account VALUES(120, '03-Apr-20', NULL, 'Active');

-- card

INSERT INTO card VALUES (101, 5048377185968208, '14-Oct-25', 706);

INSERT INTO card VALUES (102, 5108755683933492, '10-Feb-24', 519);

INSERT INTO card VALUES (103, 5108755923496771, '22-Nov-25', 936);

INSERT INTO card VALUES (104, 5048374943216079, '15-Jun-24', 413);

INSERT INTO card VALUES (105, 5108756498338828, '27-Aug-26', 428);

-- checking account

INSERT INTO checkingaccount VALUES (106, 331.93, 5.00);

INSERT INTO checkingaccount VALUES (107, 107.42, 5.00);

INSERT INTO checkingaccount VALUES (108, 7028.2, 5.00);

INSERT INTO checkingaccount VALUES (109, 413.48, 5.00);

INSERT INTO checkingaccount VALUES (110, 95.82, 5.00);

-- savings account

INSERT INTO savingsaccount VALUES (111, 22102.39, 1.31, 252.28);

INSERT INTO savingsaccount VALUES (112, 8525.10, 2.44, 135.96);

INSERT INTO savingsaccount VALUES (113, 54213.57, 2.35, 278.16);

INSERT INTO savingsaccount VALUES (114, 9147.71, 0.79, 349.15);

INSERT INTO savingsaccount VALUES (115, 460756.1, 3.9, 67.3);

-- cd account

INSERT INTO certificatedepositaccount VALUES (116, '19-Oct-25', 20000.00, 22000.00);

INSERT INTO certificatedepositaccount VALUES (117, '10-Jan-26', 25000.00, 27500.00);

INSERT INTO certificatedepositaccount VALUES (118, '05-Jun-27', 30000.00, 33000.00);

INSERT INTO certificatedepositaccount VALUES (119, '22-Sep-28', 35000.00, 38500.00);

INSERT INTO certificatedepositaccount VALUES (120, '14-May-29', 40000.00, 44000.00);

-- transaction

INSERT INTO transaction VALUES (1111, 112, 1005, 'Deposit', '30-Jan-2024', 100);

INSERT INTO transaction VALUES (2222, 106, 1011, 'Transfer From', '13-Mar-2024', -150);

INSERT INTO transaction VALUES (3333, 111, 1011, 'Transfer To', '13-Mar-2024', 150);

INSERT INTO transaction VALUES (4444, 112, NULL, 'ATM Withdrawal', '15-Sep-2023', -200);

INSERT INTO transaction VALUES (5555, 109, 1014, 'Deposit', '28-Feb-2023', 1000);

INSERT INTO transaction VALUES (6666, NULL, 1009, 'Loan Payment', '28-Jul-2023', 1000);

INSERT INTO transaction VALUES (7777, 110, 1002, 'Deposit', '20-Sep-2023', 200);

INSERT INTO transaction VALUES (8888, 117, 1008, 'Withdrawal', '08-Jun-2023', -200);

INSERT INTO transaction VALUES (9999, NULL, 1006, 'Loan Payment', '20-Aug-2023', 250);

INSERT INTO transaction VALUES (0000, 112, 1012, 'Deposit', '23-Nov-2023', 500);

-- loan

INSERT INTO loan VALUES (201, 5, 'Personal', 10000, 10.00, '01-Jan-23', '01-Jan-25', 'Active');

INSERT INTO loan VALUES (202, 11, 'Student', 100000, 5.50, '01-Jan-20', '01-Jan-30', 'Active');

-- loanpayment

INSERT INTO loanpayment VALUES (6666, 201, '28-Jul-23');

INSERT INTO loanpayment VALUES (9999, 202, '20-Aug-23');

-- atmtransaction

INSERT INTO atmtransaction VALUES (4, 4444);

-- customeraccount

INSERT INTO customeraccount VALUES (15, 101);

INSERT INTO customeraccount VALUES (1, 102);

INSERT INTO customeraccount VALUES (12, 103);

INSERT INTO customeraccount VALUES (13, 104);

INSERT INTO customeraccount VALUES (10, 105);

INSERT INTO customeraccount VALUES (16, 106);

INSERT INTO customeraccount VALUES (5, 107);

INSERT INTO customeraccount VALUES (2, 108);

INSERT INTO customeraccount VALUES (4, 109);

INSERT INTO customeraccount VALUES (3, 110);

INSERT INTO customeraccount VALUES (16, 111);

INSERT INTO customeraccount VALUES (11, 112);

INSERT INTO customeraccount VALUES (8, 113);

INSERT INTO customeraccount VALUES (14, 114);

INSERT INTO customeraccount VALUES (9, 115);

INSERT INTO customeraccount VALUES (7, 116);

INSERT INTO customeraccount VALUES (6, 117);

INSERT INTO customeraccount VALUES (13, 118);

INSERT INTO customeraccount VALUES (6, 119);

INSERT INTO customeraccount VALUES (14, 120);

**Queries**

The owner of one of the branches believes there was a fraudulent transaction on January 30, 2024. They need the customer’s name, phone number, and account ID to contact the customer and confirm the transaction.

SELECT c.fullname, c.phonenumber, a.accountid, t.transactiondate

from transaction t

INNER JOIN account a ON t.accountid = a.accountid

INNER JOIN customeraccount ca ON a.accountid = ca.accountid

INNER JOIN customer c ON ca.customerid = c.customerid

WHERE t.transactiondate = '30-Jan-2024'

Retrieve the full names and email addresses of all customers

SELECT c.fullname, c.phonenumber, a.accountid, t.transactiondate

Find all employees who work in the Finance department

SELECT \* FROM employee WHERE department = 'Finance';

Retrieve the branch names and phone numbers of all branches.

SELECT branchname, phonenumber

FROM branch;

Display the length name of an employee with their designated ID

SELECT employeeid, fullname, LENGTH(fullname) AS name\_length

FROM employee

WHERE LENGTH(fullname) = (SELECT MAX(LENGTH(fullname)) FROM employee);

Obtain the total number of employees per branch

SELECT BRANCHNAME,

(SELECT COUNT(\*) FROM EMPLOYEE WHERE BRANCH.BRANCHID = EMPLOYEE.BRANCHID) AS NUM\_EMPLOYEES

FROM BRANCH;

Obtain the available atm withdrawal amount per branch and atm id

SELECT \* FROM employee WHERE department = 'Finance';

Obtain the available ATM withdrawal amount per ATM at each branch

SELECT b.branchname, a.atmid, a.availableamt

FROM branch b

LEFT JOIN atm a ON b.branchid = a.branchid;

Obtain the total ATM withdrawal amount available at each branch

SELECT b.branchname, SUM(a.availableamt) AS total\_available\_atm\_amount

FROM branch b

LEFT JOIN atm a ON b.branchid = a.branchid

GROUP BY b.branchname;

Gather all customers along with their associated accounts

SELECT c.fullname, a.accountid

FROM customer c

RIGHT JOIN customeraccount ca ON c.customerid = ca.customerid

RIGHT JOIN account a ON ca.accountid = a.accountid;

Get the total balance of checking accounts at each branch

SELECT b.branchname, SUM(ca.balance) AS total\_balance

FROM branch b

JOIN employee e ON b.branchid = e.branchid

JOIN checkingaccount ca ON e.employeeid = ca.accountid

GROUP BY b.branchname; ';

Retrieve the full names and account IDs of customers who have a savings account with an interest rate greater than 2%

SELECT c.fullname, ca.accountid

FROM customer c

INNER JOIN customeraccount ca ON c.customerid = ca.customerid

INNER JOIN savingsaccount sa ON ca.accountid = sa.accountid

WHERE sa.interestrate > 2;

Retrieve the full names of employees along with the branch names where they work

SELECT e.fullname, b.branchname

FROM employee e

INNER JOIN branch b ON e.branchid = b.branchid;

Retrieve the count of active loans.

SELECT COUNT(\*) AS active\_loans

FROM loan

WHERE loanstatus = 'Active';

Retrieve the customer full names, account IDs, and transaction types for transactions with amounts greater than $500, ordered by transaction date

SELECT c.fullname, t.accountid, t.transactiontype

FROM customer c

INNER JOIN customeraccount ca ON c.customerid = ca.customerid

INNER JOIN transaction t ON ca.accountid = t.accountid

WHERE t.amount > 500

ORDER BY t.transactiondate;

Find the number of ATM transactions made by each customer

SELECT c.fullname, COUNT(at.atmid) AS num\_transactions

FROM customer c

JOIN customeraccount ca ON c.customerid = ca.customerid

JOIN account a ON ca.accountid = a.accountid

JOIN transaction t ON a.accountid = t.accountid

JOIN atmtransaction at ON t.transactionid = at.transactionid

GROUP BY c.fullname;

Retrieve the names and email addresses of customers who have a savings account with an interest rate greater than 2%

SELECT c.fullname, c.email

FROM customer c

INNER JOIN customeraccount ca ON c.customerid = ca.customerid

INNER JOIN savingsaccount sa ON ca.accountid = sa.accountid

WHERE sa.interestrate > 2;

View customer's data with all its associated tables

CREATE VIEW CustomerDataView AS

SELECT

c.customerid,

c.fullname AS customer\_name,

c.customertype,

c.address AS customer\_address,

c.email AS customer\_email,

c.phonenumber AS customer\_phone,

a.accountid,

a.opendate AS account\_open\_date,

a.closedate AS account\_close\_date,

a.status AS account\_status,

ca.balance AS checking\_balance,

ca.percheckcharge AS checking\_per\_check\_charge

FROM

customer c

LEFT JOIN

customeraccount cac ON c.customerid = cac.customerid

LEFT JOIN

account a ON cac.accountid = a.accountid

LEFT JOIN

checkingaccount ca ON a.accountid = ca.accountid;

SELECT \* FROM CustomerData;

Extract the first name of a customer and count the number of occurrences it appears in the customer table

SELECT

SUBSTR(fullname, 1, INSTR(fullname, ' ') - 1) AS first\_name,

COUNT(\*) AS occurrences

FROM

customer

GROUP BY

SUBSTR(fullname, 1, INSTR(fullname, ' ') - 1)

ORDER BY

occurrences DESC;

**Statement of Collaboration**

Abel Deulofeu contributed 20%. David Heller contributed 20%. Anirudh Naram Reddy contributed 20%. Khushi Sarin contributed 20%. Shikha Shikha contributed 20%. Everyone was an equal participant for this project and collaborated efficiently.