Team: Triple G
Karanjit Gill (kgill15@csustan.edu)
Giancarlo Garcia Deleon(ggarciadeleon@csustan.edu)
Chelcey Guptail(cguptail@csustan.edu)
CS 4250
Project Part - 7
https://hopper.csustan.edu/~kgill/index.php

Project Report

Part 1:

I. Meet Triple G

The name of the project is Banksy. The group members consist of Giancarlo Garcia Deleon (ggarciadeleon@csustan.edu), Karanjit Gill (kgill15@csustan.edu), and Chelcey Guptail (cguptail@csustan.edu).

II. Domain of the Database being Design

[Banksy]

We chose our domain to be a banking system. In this system, we will have the banks, the members, and the bank employees. The members will have a relationship to their accounts which would branch to their checkings and/or savings with more specific attributes, such as cash amount in their account, interest rate, and credit scores. The bank entity would also branch an employment relationship with its employees.

III. The intended user group for the database application

The intended user group for the bank database application would be the management working at a bank. The managers would have access to the information of both the employees, members and non-members of the bank who simply withdraw money from the bank.

IV. What will be modeled by the system (and what will not)

The dataset would need to keep track of the members. Members come into a bank to deposit and withdraw their money, which is kept logged in the database. In addition, further entities that would need to be observed are the member's credentials, backed up logged transaction statements, employee's credentials and member's bank accounts.

V. Rules of the Group

Our group has decided that Thursday evenings work best as meeting time in addition to any times where we can meet online through Discord. (i) We have agreed to meet at least once a week to go over the responsibilities of each and discuss our progress. (ii) If a member is unable to make it to a delegated meeting they will be issued a warning in advance by digital document or electronic mail and all the group members will be notified by their uncooperativeness. (iii) Everyone will actively participate in the meetings and they will help with whatever tasks are necessary to ensure that all assignments are completed on time. (iv) If anyone in the group is having trouble completing any of the assignments or is ailed in any way, they will notify the group to ensure that a proper procedure to get them back on track is executed. (v) All members of the group must be up to date with the content that is instructed within the in-person lecture to ensure effective collaboration.

VI. What other "value-added" facilities can be implemented.

A mobile application for all members of the bank to be able to deposit checks is the tool we will create. One of the value-added facilities that we could implement is a transaction management system of the bank with non-members and members. Some banks allow non-members to cash checks that are written out to the non-member but are the bank's check.

The database will store the transaction ID of these non-members along with their names in case of fraud or any other malicious reasons. Another value-added facility that we could add is the ability for members to open an investment account with the bank which can either let the member decide how they would like to invest or let the bank invest for them using different strategies. These strategies would be for short term aggressive investing, medium-term reactive investing, or long term passive investing.

Part 2:

Explanations:

1. Banksy the database models how a bank works and looks from the inside for management.
    a. A bank can employ many employees.
    b. A bank can have many members.
    c. A bank can also have many non-members.
2. The employees model many people who are employed in a bank
    a. An employee can only be part of one bank.
    b. An employee model can become a manager.
        i. A manager can evaluate many employees.
        ii. A manager can be a different type. (ex: local, regional)
    c. An employee model can become a salesperson
        i. A salesperson can be a Sales Associate.
            1. A Sales Associate has an officeNumber.
            2. A Sales Associate can have many clients.
        ii. A salesperson can be a Teller.
            1. A Teller has a bankBoothNumber.
            2. Many Tellers can process many transactions.
        iii. A salesperson can be an Investment Advisor.
            1. An Investment Advisor has an officeNumber.
3. A member is a member of a bank
    a. A member can be a client of many Sales Associates
    b. A member can have holdings in many Bank Accounts
        i. A Bank account can have more than 1 member.
        ii. A Bank account be different types of accounts
            1. A Bank Account can be an IRA account

        2. A Bank Account can be an Investment Account
        3. A Bank Account can be a Savings Account
        4. A Bank Account can be a Checkings Account
4. Non-members can withdraw from a bank

Constraints:

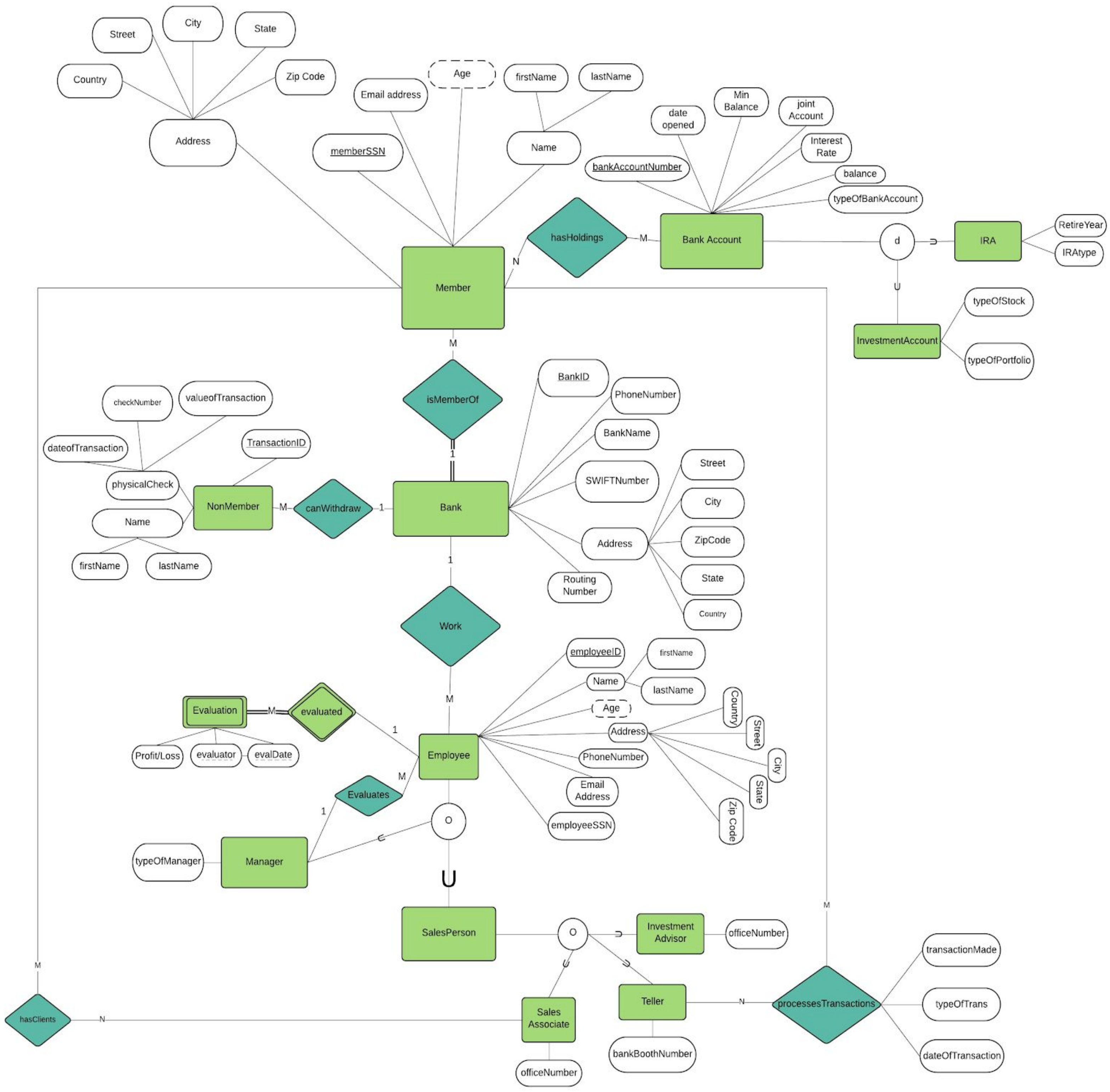
Some of the constraints that our ER diagram cannot show is if the SSN of a member or employee is NULL. This would not be allowed as that is crucial information used to identify a member and employee. Another constraint could be that a nonMembers checks valueofTransaction cannot be negative as that would mean the nonMember is depositing a check into the bank while not having an account with the bank. Another constraint could be that members can not have a country, street, city, state, and zipCode be null because the bank would not give out accounts to members without an address. A member's interest rate cannot be negative, or else the bank is stealing their money.

Group Contributions:

Chelcey: Created the first written draft and contributed to helping to expand the ER diagram.
Giancarlo: Contributed to our second draft of the ER diagram
Karanjit: Created/finished the team's ER diagram in the online program lucid chart.

ER Diagram Part 2 Transaction Management System

Karanjit Gill, Chelcey Guptail, Giancarlo Garcia Deleon | May 24, 2020



Part 3:

Bank: bankID, phoneNumber, bankName, swiftNumber, address, routingNumber
Member: memberSSN, emailAddress, age, name
Employee: employeeID, name, age, address, phoneNumber, emailAddress, employeeSSN
Bank Account: bankAccountNumber, dateOpened, minBalance, jointAccount, interestRate, balance, typeOfBankAccount
Non-Member: transactionID, physicalCheck, name

Our domain shows the inner workings of a banking system. This includes data for bank, non-members, members, employees, employee evaluations, and bank accounts. The constraints that we were unable to show are the specific stocks held by an investment account, which other members have access to a joint account, and the different IRA account types benefits.

Karanjit Gill: Worked on the ER diagram, and fixed up part 3 SQL code.
Giancarlo Garcia Deleon: Worked on revising and editing the ER diagram and co-authored Part 3 SQL code.
Chelcey Guptail: Worked on fixing up the ER diagram and Part 3 code.

```
CREATE DATABASE Banksy;
USE Banksy;

CREATE TABLE Member(
country char(30),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
age INT,
emailAddress char(50),
memberSSN char(11) NOT NULL,
firstName char(50),
lastName char(50),
PRIMARY KEY (memberSSN)
);

CREATE TABLE NonMember(
firstName char(50),
lastName char(50),
dateofTransaction date,
checkNumber INT(10),
valueOfTransaction INT,
transactionID char(50) NOT NULL,
PRIMARY KEY (transactionID)
);

CREATE TABLE Bank(
phoneNumber INT,
bankID char(50) NOT NULL,
bankName char(50),
swiftNumber char(50),
routingNumber INT(9),
country char(30),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
PRIMARY KEY (bankID)
);


CREATE TABLE Employee(
emailAddress char(50),
employeeID char(20),
employeeSSN char(11),
firstName char(50),
lastName char(50),
age INT,
phoneNumber INT,
country char(30),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
PRIMARY KEY (employeeID)
);

CREATE TABLE BankAccount(
interestRate decimal(p,s),
dateOpened date,
minBalance INT,
jointAccount BOOL,
bankAccountNumber INT(12) NOT NULL,
balance INT,
typeOfBankAccount char(20),
memberSSN char(11),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (memberSSN) references Member(memberSSN) ON DELETE CASCADE
);

CREATE TABLE Manager(
typeOfManager char(15),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) references Employee(employeeID) ON DELETE CASCADE
);

CREATE TABLE EvaluationEvaluated(
profitLoss INT,
evaluator char(50) NOT NULL,
evalDate date NOT NULL,
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID, evalDate, evaluator),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID) ON DELETE CASCADE
);

CREATE TABLE IRA(
bankAccountNumber INT(12) NOT NULL,
retireYear date,
iraType char(20),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber ON
DELETE CASCADE
);

CREATE TABLE InvestmentAccount(
bankAccountNumber INT(12) NOT NULL,
typeOfStock char(20),
typeOfPortfolio char(20),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber ON
DELETE CASCADE
);


CREATE TABLE SalesPerson(
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES Employee(employeeID) ON DELETE CASCADE
);

CREATE TABLE SalesAssociate (
officeNumber INT,
Member name char(100),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES SalesPerson(employeeID) ON DELETE CASCADE
);

CREATE TABLE Teller(
bankBoothNumber INT,
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES SalesPerson(employeeID) ON DELETE CASCADE
);

CREATE TABLE InvestmentAdvisor(
officeNumber char(10),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES SalesPerson(employeeID) ON DELETE CASCADE
);

CREATE TABLE isMemberOf(
memberSSN char(11) NOT NULL,
bankID char(50) NOT NULL,
PRIMARY KEY (memberSSN),
FOREIGN KEY (memberSSN) REFERENCES Member(memberSSN) ON DELETE CASCADE,
FOREIGN KEY (bankID) REFERENCES Bank(bankID) ON DELETE CASCADE
);

CREATE TABLE hasHoldings(
memberSSN char(11),
bankAccountNumber INT(12),
PRIMARY KEY (memberSSN, bankAccountNumber),
FOREIGN KEY (memberSSN) REFERENCES Member(memberSSN) ON DELETE CASCADE,
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber) ON
DELETE CASCADE
);
```

```
CREATE TABLE canWithdraw(
transactionID char(50) NOT NULL,
bankID char(50) NOT NULL,
PRIMARY KEY (transactionID),
FOREIGN KEY (transactionID) REFERENCES NonMember(transactionID) ON DELETE CASCADE,
FOREIGN KEY (bankID) REFERENCES Bank(bankID) ON DELETE CASCADE
);

CREATE TABLE Work(
employeeID char(20),
bankID char(50) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID),
FOREIGN KEY (bankID) REFERENCES Bank(bankID)
);

CREATE TABLE Evaluates(,
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID) ON DELETE CASCADE
);

CREATE TABLE processTransactions(
transactionMade char(30),
typeOfTrans char(30),
dateOfTrans date,
employeeID char(20) NOT NULL,
memberSSN char(11) NOT NULL,
PRIMARY KEY (employeeID, memberSSN),
FOREIGN KEY (employeeID) REFERENCES Teller(employeeID),
FOREIGN KEY (memberSSN) REFERENCES Member (memberSSN)
);

CREATE TABLE hasClients(
employeeID char(20)NOT NULL,
memberSSN char(11)NOT NULL,
PRIMARY KEY (employeeID, memberSSN),
FOREIGN KEY (employeeID) REFERENCES SalesAssociate(employeeID),
FOREIGN KEY (memberSSN) REFERENCES Member (memberSSN)
);
```

Part 4:

Group Contributions:

Chelcey: Assisted with the contribution of section 2 and 3.
Giancarlo: Assisted with sections 1 and 3, contributed in some decomposition to BCNF. Helped address incorrect relational schemas.
Karanjit: Assisted with sections 2 and 3, contributed in correcting section 1 in part 3.

Section 1: Initial Relations

```
CREATE DATABASE Banksy;
USE Banksy;

CREATE TABLE Member(
country char(30),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
age INT,
emailAddress char(50),
memberSSN char(11) NOT NULL,
firstName char(50),
lastName char(50),
PRIMARY KEY (memberSSN)
);

CREATE TABLE NonMember(
firstName char(50),
lastName char(50),
dateofTransaction date,
checkNumber INT(10),
valueOfTransaction INT,
transactionID char(50) NOT NULL,
PRIMARY KEY (transactionID)
```

```
);

CREATE TABLE Bank(
phoneNumber INT (13),
bankID char(50) NOT NULL,
bankName char(50),
swiftNumber char(50),
routingNumber INT(9),
country char(30),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
PRIMARY KEY (bankID)
);

CREATE TABLE Employee(
emailAddress char(50),
employeeID char(20),
employeeSSN char(11),
firstName char(50),
lastName char(50),
age INT,
phoneNumber INT,
country char(50),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
PRIMARY KEY (employeeID)
);

CREATE TABLE BankAccount(
interestRate decimal(p,s),
dateOpened date,
minBalance INT,
jointAccount BOOL,
bankAccountNumber INT(12) NOT NULL,
balance INT,
typeOfBankAccount char(20),
memberSSN char(11),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (memberSSN) references Member(memberSSN) ON DELETE CASCADE
);

CREATE TABLE Manager(
typeOfManager char(15),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) references Employee(employeeID) ON DELETE CASCADE
);

CREATE TABLE EvaluationEvaluated(
profitLoss INT,
evaluator char(50) NOT NULL,
evalDate date NOT NULL,
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID, evalDate, evaluator),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID) ON DELETE
CASCADE
);

CREATE TABLE IRA(
bankAccountNumber INT(12) NOT NULL,
retireYear date,
iraType char(20),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber
ON DELETE CASCADE
);

CREATE TABLE InvestmentAccount(
bankAccountNumber INT(12) NOT NULL,
typeOfStock char(20),
typeOfPortfolio char(20),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber
ON DELETE CASCADE
```

```
);

CREATE TABLE SalesPerson(
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES Employee(employeeID) ON DELETE
CASCADE
);

CREATE TABLE SalesAssociate (
officeNumber INT,
memberName char(100),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES SalesPerson(employeeID) ON DELETE
CASCADE
);

CREATE TABLE Teller(
bankBoothNumber INT,
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES SalesPerson(employeeID) ON DELETE
CASCADE
);

CREATE TABLE InvestmentAdvisor(
officeNumber char(10),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES SalesPerson(employeeID) ON DELETE
CASCADE
);

CREATE TABLE isMemberOf(
memberSSN char(11) NOT NULL,
bankID char(50) NOT NULL,
PRIMARY KEY (memberSSN),
FOREIGN KEY (memberSSN) REFERENCES Member(memberSSN) ON DELETE
CASCADE,
FOREIGN KEY (bankID) REFERENCES Bank(bankID) ON DELETE CASCADE
);

CREATE TABLE hasHoldings(
memberSSN char(11),
bankAccountNumber INT(12),
PRIMARY KEY (memberSSN, bankAccountNumber),
FOREIGN KEY (memberSSN) REFERENCES Member(memberSSN) ON DELETE
CASCADE,
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber)
ON DELETE CASCADE
);

CREATE TABLE canWithdraw(
transactionID char(50) NOT NULL,
bankID char(50) NOT NULL,
PRIMARY KEY (transactionID),
FOREIGN KEY (transactionID) REFERENCES NonMember(transactionID) ON DELETE
CASCADE,
FOREIGN KEY (bankID) REFERENCES Bank(bankID) ON DELETE CASCADE
);

CREATE TABLE Work(
employeeID char(20),
bankID char(50) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID),
FOREIGN KEY (bankID) REFERENCES Bank(bankID)
);

CREATE TABLE Evaluates(
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID) ON DELETE
CASCADE
);
```

```
CREATE TABLE processTransactions(
transactionMade char(30),
typeOfTrans char(30),
dateOfTrans date,
employeeID char(20) NOT NULL,
memberSSN char(11) NOT NULL,
PRIMARY KEY (employeeID, memberSSN),
FOREIGN KEY (employeeID) REFERENCES Teller(employeeID),
FOREIGN KEY (memberSSN) REFERENCES Member (memberSSN)
);


CREATE TABLE hasClients(
employeeID char(20)NOT NULL,
memberSSN char(11)NOT NULL,
PRIMARY KEY (employeeID, memberSSN),
FOREIGN KEY (employeeID) REFERENCES SalesAssociate(employeeID),
FOREIGN KEY (memberSSN) REFERENCES Member (memberSSN)
);
```

Section 2: Functional Dependencies

FD for Bank

BankID → PhoneNumber
BankID → BankName
BankID → SWIFTNumber
BankID → street
BankID → city
BankID → zipCode
BankID →state
BankID → country
BankID → routingNumber

FD for canWithdraw

<u>All FDs are trivial</u>

FD for NonMember

TransactionID → dateofTransaction

TransactionID →checkNumber
TransactionID → valueofTransaction
TransactionID → firstName
TransactionID → lastName

FD for isMemberOf
All FDs are trivial

FD for Member
memberSSN → emailAddress
memberSSN → age
memberSSN → firstName
memberSSN → lastName
memberSSN → street
memberSSN → city
memberSSN → zipCode
memberSSN →state
memberSSN → country

FD for hasHoldings
All FDs are trivial

FD for BankAccount
bankAccountNumber → dateOpened
bankAccountNumber → MinBalance
bankAccountNumber → joinAccount
bankAccountNumber → InterestRate
bankAccountNumber → Balance
bankAccountNumber → typeOfAcount

FD for IRA
bankAccountNumber → RetireYear
bankAccountNumber → IRAtype

FD for InvestmentAccount
bankAccountNumber → typeOfStock
bankAccountNumber → typeOfPortfolio

FD for Work

<u>All FDs are trivial</u>

FD for Employee

employeeID → firstName
employeeID → lastName
employeeID → Age
employeeID → street
employeeID → city
employeeID → zipCode
employeeID → state
employeeID → country
employeeID → PhoneNumber
employeeID → emailAddress
employeeSSN → employeeID
employeeSSN → lastName
employeeSSN → firstName
employeeSSN → Age
employeeSSN → street
employeeSSN → city
employeeSSN → zipCode
employeeSSN → state
employeeSSN → country
employeeSSN → PhoneNumber
employeeSSN → emailAddress

FD for Evaluates

<u>All FDs are trivial</u>

FD for Manager

employeeID → typeOfManager

FD for SalesPerson

<u>All FDs are trivial</u>

FD for SalesAssociate

employeedID → employeeIDaddress
employeedID → officeNumber

FD for hasClients
All FDs are trivial

FD for Teller
employeeID → bankBoothNumber

FD for processTransactions
employeeID → transcationMade
employeeID → typeOfTrans
employeeID → dateOfTransaction

FD for InvestmentAdvisor
employeeID → officeNumber

FD for Evaluation
employeeID, Evaluator, evalDate → profitLoss

FD for evaluated
All FDs are trivia

Section 3: Normalization

Bank - This table is in BCNF because BankID is the candidate key and BankID is the only thing on the left in all of the Functional Dependencies.

canWithdraw - This table is in 3NF because bankID and transactionID can define all the attributes in canWithdraw. This table is not in BCNF because there are two overlapping candidate keys, making it trivial.

NonMember - This table is in BCNF because TransactionID is the candidate key and memberSSN is the only thing on the left of the Functional Dependencies.

isMemberOf - This table is in 3NF because BankID and memberSSN can define all the attributes in isMemberOf. This table is not in BCNF because there are two overlapping candidate keys, making it trivial. We cannot convert 3NF to BCNF because there are overlapping candidate keys being declared, with no attributes.

Member - This table is in BCNF because memberSSN is the candidate key and memberSSN is the only thing on the left of the Functional Dependencies.

hasHoldings - This table is in BCNF because memberSSN and bankAccountNumber can define all the attributes in hasHoldings. This is not 3NF because there are only non-trivial dependencies.

BankAccount - This table is in BCNF because bankAccountNumber is the candidate key and bankAccountNumber is the only thing on the left of the Functional Dependencies.

IRA - This table is in BCNF because bankAccountNumber is the candidate key and bankAccountNumber is the only thing on the left of the Functional Dependencies.

InvestmentAccount - This table is in BCNF because bankAccountNumber is the candidate key and bankAccountNumber is the only thing on the left of the Functional Dependencies.

Work - This table is in BCNF because BankID and employeeID can define all the attributes in Work. This table is not in BCNF because there are two overlapping candidate keys, making it trivial.

Employee - This table is in 3NF because employeeSSN and employeeID can define all the attributes in Work. This table is not in BCNF because there are two overlapping candidate keys, making it trivial.

```
CREATE TABLE EmployeeContact(
employeeID → Age
employeeID  → street
employeeID  → city
employeeID  → zipCode
employeeID  → state
employeeID  → country
employeeID → PhoneNumber
employeeID → emailAddress
employeeID → employeeSSN
);
```

```
CREATE TABLE EmployeeName(
employeeSSN → lastName
employeeSSN →firstName
);
```

Evaluates - This table is in BCNF because employeeID is the candidate key and employeeID is the only thing on the left of the Functional Dependencies.

Manager - This table is in BCNF because employeeID is the candidate key and employeeID is the only thing on the left of the Functional Dependencies.

SalesPerson - This table is in BCNF because employeeID is the candidate key and employeeID is the only thing on the left of the Functional Dependencies.

SalesAssociate - This table is in BCNF because employeeID is the candidate key and employeeID is the only thing on the left of the Functional Dependencies

hasClients - This table is in 3NF because employeeID and memberSSN can define all the attributes in Work. This table is not in BCNF because there are two overlapping candidate keys, making it trivial. We cannot convert 3NF to BCNF because there are overlapping candidate keys being declared, with no attributes.

Teller - This table is in BCNF because employeeID is the candidate key and employeeID is the only thing on the left of the Functional Dependencies.

processTransactions - This table is in BCNF because employeeID is the candidate key and employeeID is the only thing on the left of the Functional Dependencies.

InvestmentAdvisor - This table is in BCNF because employeeID is the candidate key and employeeID is the only thing on the left of the Functional Dependencies.

Evaluation - This table is in 3NF because evaluator and evalDate can define all the attributes in Work. This table is not in BCNF because there are two overlapping candidate keys, making it trivial. We cannot convert 3NF to BCNF because there are overlapping candidate keys being declared, with no attributes.

Evaluated - This table is in 3NF because employeeID, Evaluator, and evalDate can define all the attributes in Work. This table is not in BCNF because there are two overlapping candidate keys, making it trivial. We cannot convert 3NF to BCNF because there are overlapping candidate keys being declared, with no attributes.

Part 5:

Group Contributions:
Chelcey: Created all foreign keys for all tables. Contributed to populating tables. Took screenshots for a sample of tuples.
Giancarlo: Created all tables. Contributed to populating tables. Added observations and inserts, updates, and delete examples.
Karanjit: Created and fixed relationship constraints for foreign keys, helped populate data and took screenshots for the explain command.

Section 1: CREATE TABLE STATEMENTS

```
CREATE DATABASE Banksy;
USE Banksy;

CREATE TABLE Member(
country char(30),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
age INT,
emailAddress char(50),
memberSSN char(11) NOT NULL,
firstName char(50),
lastName char(50),
PRIMARY KEY (memberSSN)
);

CREATE TABLE NonMember(
firstName char(50),
lastName char(50),
dateofTransaction date,
checkNumber INT(10),
valueOfTransaction INT, Changed to decimal(10,2) for more realistic input data
transactionID char(50) NOT NULL,
PRIMARY KEY (transactionID)
```

```
);

CREATE TABLE Bank(
phoneNumber INT (13), Changed to varchar(30) to input phoneNumber properly
bankID char(50) NOT NULL,
bankName char(50),
swiftNumber char(50),
routingNumber INT(9), Changed to varchar(30) to input routingNumber properly
country char(30),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
PRIMARY KEY (bankID)
);

CREATE TABLE Employee(
emailAddress char(50),
employeeID char(20),
employeeSSN char(11),
firstName char(50),
lastName char(50),
age INT,
phoneNumber INT, Changed to varchar(30) to input phoneNumber properly
country char(50),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
PRIMARY KEY (employeeID)
);

CREATE TABLE BankAccount(
interestRate decimal(p,s),
dateOpened date,
minBalance INT,
jointAccount BOOL,
bankAccountNumber INT(12) NOT NULL,
balance INT,
typeOfBankAccount char(20),
memberSSN char(11),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (memberSSN) references Member(memberSSN) ON DELETE CASCADE
);

CREATE TABLE Manager(
typeOfManager char(15),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) references Employee(employeeID) ON DELETE CASCADE
);

CREATE TABLE EvaluationEvaluated(
profitLoss INT, Changed to bigint so bigger numbers could be inputed
evaluator char(50) NOT NULL,
evalDate date NOT NULL,
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID, evalDate, evaluator),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID) ON DELETE
CASCADE
);

CREATE TABLE IRA(
bankAccountNumber INT(12) NOT NULL,
retireYear date,
iraType char(20),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber
ON DELETE CASCADE
);

CREATE TABLE InvestmentAccount(
bankAccountNumber INT(12) NOT NULL,
typeOfStock char(20),
typeOfPortfolio char(20),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber
ON DELETE CASCADE
```

```
);

CREATE TABLE SalesPerson(
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES Employee(employeeID) ON DELETE
CASCADE
);

CREATE TABLE SalesAssociate (
officeNumber INT,
memberName char(100),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES SalesPerson(employeeID) ON DELETE
CASCADE
);

CREATE TABLE Teller(
bankBoothNumber INT,
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES SalesPerson(employeeID) ON DELETE
CASCADE
);

CREATE TABLE InvestmentAdvisor(
officeNumber char(10),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES SalesPerson(employeeID) ON DELETE
CASCADE
);

CREATE TABLE isMemberOf(
memberSSN char(11) NOT NULL,
bankID char(50) NOT NULL,
PRIMARY KEY (memberSSN),
FOREIGN KEY (memberSSN) REFERENCES Member(memberSSN) ON DELETE
CASCADE,
FOREIGN KEY (bankID) REFERENCES Bank(bankID) ON DELETE CASCADE
);

CREATE TABLE hasHoldings(
memberSSN char(11),
bankAccountNumber INT(12),
PRIMARY KEY (memberSSN, bankAccountNumber),
FOREIGN KEY (memberSSN) REFERENCES Member(memberSSN) ON DELETE
CASCADE,
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber)
ON DELETE CASCADE
);

CREATE TABLE canWithdraw(
transactionID char(50) NOT NULL,
bankID char(50) NOT NULL,
PRIMARY KEY (transactionID),
FOREIGN KEY (transactionID) REFERENCES NonMember(transactionID) ON DELETE
CASCADE,
FOREIGN KEY (bankID) REFERENCES Bank(bankID) ON DELETE CASCADE
);

CREATE TABLE Work(
employeeID char(20),
bankID char(50) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID),
FOREIGN KEY (bankID) REFERENCES Bank(bankID)
);

CREATE TABLE Evaluates(
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID) ON DELETE
CASCADE
);
```

```
CREATE TABLE processTransactions(
transactionMade char(30),
typeOfTrans char(30),
dateOfTrans date,
employeeID char(20) NOT NULL,
memberSSN char(11) NOT NULL,
PRIMARY KEY (employeeID, memberSSN),
FOREIGN KEY (employeeID) REFERENCES Teller(employeeID),
FOREIGN KEY (memberSSN) REFERENCES Member (memberSSN)
);


CREATE TABLE hasClients(
employeeID char(20)NOT NULL,
memberSSN char(11)NOT NULL,
PRIMARY KEY (employeeID, memberSSN),
FOREIGN KEY (employeeID) REFERENCES SalesAssociate(employeeID),
FOREIGN KEY (memberSSN) REFERENCES Member (memberSSN)
);
```

SECTION 2: LIST OF SQL CREATE TABLE COMMANDS

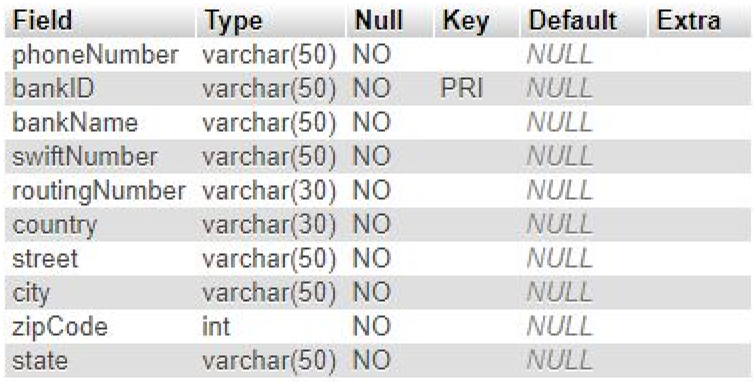
```
-- Database: `gdeleon`
CREATE TABLE `Bank` (
        `phoneNumber` varchar(50) NOT NULL,
        `bankID` varchar(50) NOT NULL,
        `bankName` varchar(50) NOT NULL,
        `swiftNumber` varchar(50) NOT NULL COMMENT 'Called a number but in a form of a
string',
        `routingNumber` varchar(30) NOT NULL,
        `country` varchar(30) CHARACTER NOT NULL,
        `street` varchar(50) CHARACTER NOT NULL,
        `city` varchar(50) NOT NULL,
        `zipCode` int NOT NULL,
        `state` varchar(50) NOT NULL,
        ADD PRIMARY KEY (`bankID`)
);

CREATE TABLE `BankAccount` (
        `interestRate` decimal(10,2) NOT NULL,
        `dateOpened` date NOT NULL,
        `minBalance` int NOT NULL,
        `jointAccount` tinyint(1) NOT NULL,
        `bankAccountNumber` varchar(30) NOT NULL,
        `balance` int NOT NULL,
        `typeOfBankAccount` varchar(20) NOT NULL,
        `memberSSN` varchar(11) NOT NULL
        ADD PRIMARY KEY (`bankAccountNumber`),
        ADD CONSTRAINT `bankaccount_ibfk_1` FOREIGN KEY (`memberSSN`)
REFERENCES `Member` (`memberSSN`) ON DELETE CASCADE
);

CREATE TABLE `canWithdraw` (
        `transactionID` varchar(50) NOT NULL,
        `bankID` varchar(50) NOT NULL,
        ADD PRIMARY KEY (`transactionID`),
	ADD CONSTRAINT `canwithdraw_ibfk_1` FOREIGN KEY (`bankID`) REFERENCES
`Bank` (`bankID`) ON DELETE CASCADE,
	ADD CONSTRAINT `canwithdraw_ibfk_2` FOREIGN KEY (`transactionID`)
REFERENCES `NonMember` (`transactionID`) ON DELETE CASCADE
);

CREATE TABLE `Employee` (
	`emailAddress` varchar(50) NOT NULL,
	`employeeID` varchar(20) NOT NULL,
	`employeeSSN` varchar(11) NOT NULL,
	`firstName` varchar(50) NOT NULL,
	`lastName` varchar(50) NOT NULL,
	`phoneNumber` varchar(30) NOT NULL,
	`age` int NOT NULL,
	`country` varchar(50) NOT NULL,
	`street` varchar(50) NOT NULL,
	`state` varchar(50) NOT NULL,
	`city` varchar(50) NOT NULL,
	`zipCode` int NOT NULL,
	ADD PRIMARY KEY (`employeeID`)
);

CREATE TABLE `Evaluates` (
	`employeeID` varchar(20) NOT NULL,
	ADD PRIMARY KEY (`employeeID`),
	ADD CONSTRAINT `evaluates_eid_1` FOREIGN KEY (`employeeID`)
REFERENCES `Employee` (`employeeID`) ON DELETE CASCADE;
);

CREATE TABLE `EvaluationEvaluated` (
	`profitLoss` bigint NOT NULL,
	`evaluator` varchar(50) NOT NULL,
	`evalDate` date NOT NULL,
	`employeeID` varchar(20) NOT NULL,
	ADD PRIMARY KEY (`employeeID`,`evalDate`,`evaluator`),
	ADD CONSTRAINT `evaluationevaluated_ibfk_1` FOREIGN KEY (`employeeID`)
REFERENCES `Employee` (`employeeID`) ON DELETE CASCADE
);
```

```
CREATE TABLE `hasClients` (
	`employeeID` varchar(20) NOT NULL,
	`memberSSN` varchar(11) NOT NULL,
	ADD PRIMARY KEY (`employeeID`,`memberSSN`),
	ADD CONSTRAINT `hasclients_ibfk_1` FOREIGN KEY (`employeeID`)
REFERENCES `Employee` (`employeeID`) ON DELETE CASCADE,
	ADD CONSTRAINT `hasclients_ibfk_2` FOREIGN KEY (`memberSSN`)
REFERENCES `Member` (`memberSSN`) ON DELETE CASCADE
);

CREATE TABLE `hasHoldings` (
	`memberSSN` varchar(11) NOT NULL,
	`bankAccountNumber` varchar(30) NOT NULL,
	ADD PRIMARY KEY (`memberSSN`) USING BTREE,
	ADD CONSTRAINT `hasholdings_ibfk_2` FOREIGN KEY (`memberSSN`)
REFERENCES `Member` (`memberSSN`) ON DELETE CASCADE ON UPDATE
RESTRICT,
	ADD CONSTRAINT `hasholdings_ibfk_3` FOREIGN KEY (`bankAccountNumber`)
REFERENCES `BankAccount` (`bankAccountNumber`) ON DELETE CASCADE;
);

CREATE TABLE `InvestmentAccount` (
	`bankAccountNumber` varchar(30) NOT NULL,
	`typeOfStock` varchar(20) NOT NULL,
	`typeOfPortfolio` varchar(20) NOT NULL,
	ADD PRIMARY KEY (`bankAccountNumber`),
	ADD CONSTRAINT `investmentaccount_ibfk_1` FOREIGN KEY
(`bankAccountNumber`) REFERENCES `BankAccount` (`bankAccountNumber`) ON DELETE
CASCADE
);

CREATE TABLE `InvestmentAdvisor` (
	`officeNumber` varchar(10) NOT NULL,
	`employeeID` varchar(20) NOT NULL,
	ADD PRIMARY KEY (`employeeID`),
	ADD CONSTRAINT `investmentadvisor_ibfk_1` FOREIGN KEY (`employeeID`)
REFERENCES `Employee` (`employeeID`) ON DELETE CASCADE
);
```

```
CREATE TABLE `IRA` (
	`bankAccountNumber` varchar(30) NOT NULL,
	`retireYear` date NOT NULL,
	`iraType` varchar(20) NOT NULL,
	ADD PRIMARY KEY (`employeeID`),
	ADD CONSTRAINT `ira_ibfk_1` FOREIGN KEY (`bankAccountNumber`)
REFERENCES `BankAccount` (`bankAccountNumber`) ON DELETE CASCADE
);

CREATE TABLE `isMemberOf` (
	`memberSSN` varchar(11) NOT NULL,
	`bankID` varchar(50) NOT NULL,
	ADD PRIMARY KEY (`memberSSN`),
	ADD CONSTRAINT `ismemberof_ibfk_1` FOREIGN KEY (`bankID`) REFERENCES
`Bank` (`bankID`) ON DELETE CASCADE ON UPDATE RESTRICT,
	ADD CONSTRAINT `ismemberof_ibfk_2` FOREIGN KEY (`memberSSN`)
REFERENCES `Member` (`memberSSN`) ON DELETE CASCADE ON UPDATE RESTRICT
);

CREATE TABLE `Manager` (
	`typeOfManager` varchar(15) NOT NULL,
	`employeeID` varchar(20) NOT NULL,
	ADD PRIMARY KEY (`employeeID`),
	ADD CONSTRAINT `manager_ibfk_1` FOREIGN KEY (`employeeID`)
REFERENCES `Employee` (`employeeID`) ON DELETE CASCADE ON UPDATE
RESTRICT
);

CREATE TABLE `Member` (
	`country` varchar(30) NOT NULL,
	`street` varchar(50) NOT NULL,
	`city` varchar(50) NOT NULL,
	`state` varchar(50) NOT NULL,
	`zipcode` int NOT NULL,
	`age` int NOT NULL,
	`emailAddress` varchar(50) NOT NULL,
	`memberSSN` varchar(11) NOT NULL,
	`firstName` varchar(50) NOT NULL,
	`lastName` varchar(50) NOT NULL,
	ADD PRIMARY KEY (`memberSSN`),
);

CREATE TABLE `NonMember` (
	`firstName` varchar(50) NOT NULL,
	`lastName` varchar(50) NOT NULL,
	`dateofTransaction` date NOT NULL,
	`checkNumber` decimal(10,0) NOT NULL,
	`valueOfTransaction` decimal(10,2) NOT NULL,
	`transactionID` varchar(50) NOT NULL,
	ADD PRIMARY KEY (`transactionID`)
);

CREATE TABLE `processTransactions` (
	`transactionMade` varchar(30) NOT NULL,
	`typeOfTrans` varchar(30) NOT NULL,
	`dateOfTrans` date NOT NULL,
	`employeeID` varchar(20) NOT NULL,
	`memberSSN` varchar(11) NOT NULL,
	ADD PRIMARY KEY (`employeeID`,`memberSSN`),
	ADD CONSTRAINT `processtransactions_ibfk_1` FOREIGN KEY (`employeeID`)
REFERENCES `Employee` (`employeeID`) ON DELETE CASCADE,
	ADD CONSTRAINT `processtransactions_ibfk_2` FOREIGN KEY (`memberSSN`)
REFERENCES `Member` (`memberSSN`) ON DELETE CASCADE ON UPDATE RESTRICT
);

CREATE TABLE `SalesAssociate` (
	`officeNumber` varchar(4) NOT NULL,
	`memberName` varchar(100) NOT NULL,
	`employeeID` varchar(20) NOT NULL,
	ADD PRIMARY KEY (`employeeID`),
	ADD CONSTRAINT `salesassociate_ibfk_1` FOREIGN KEY (`employeeID`)
REFERENCES `Employee` (`employeeID`) ON DELETE CASCADE
);

CREATE TABLE `SalesPerson` (
	`employeeID` varchar(20) NOT NULL,
	ADD PRIMARY KEY (`employeeID`),
        ADD CONSTRAINT `salesperson_ibfk_1` FOREIGN KEY (`employeeID`)
REFERENCES `Employee` (`employeeID`) ON DELETE CASCADE
);

CREATE TABLE `Teller` (
        `bankBoothNumber` varchar(3) DEFAULT NULL,
        `employeeID` varchar(20) NOT NULL,
        ADD PRIMARY KEY (`employeeID`),
        ADD CONSTRAINT `teller_ibfk_1` FOREIGN KEY (`employeeID`) REFERENCES
`Employee` (`employeeID`) ON DELETE CASCADE
);

CREATE TABLE `Work` (
        `employeeID` varchar(20) NOT NULL,
        `bankID` varchar(50) NOT NULL,
        ADD PRIMARY KEY (`employeeID`),
        ADD CONSTRAINT `work_ibfk_1` FOREIGN KEY (`bankID`) REFERENCES `Bank`
(`bankID`) ON DELETE CASCADE ON UPDATE RESTRICT,
        ADD CONSTRAINT `work_ibfk_2` FOREIGN KEY (`employeeID`) REFERENCES
`Employee` (`employeeID`) ON DELETE CASCADE ON UPDATE RESTRICT
);
```

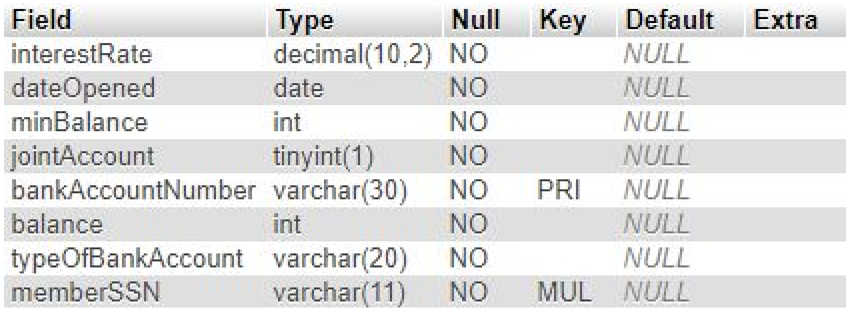
SECTION 3: SCREENSHOT OUTPUT OF THE EXPLAIN COMMAND

Bank:



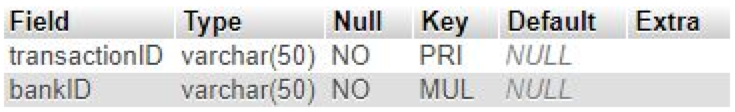
Field	Type	Null	Key	Default	Extra
phoneNumber	varchar(50)	NO		NULL	
bankID	varchar(50)	NO	PRI	NULL	
bankName	varchar(50)	NO		NULL	
swiftNumber	varchar(50)	NO		NULL	
routingNumber	varchar(30)	NO		NULL	
country	varchar(30)	NO		NULL	
street	varchar(50)	NO		NULL	
city	varchar(50)	NO		NULL	
zipCode	int	NO		NULL	
state	varchar(50)	NO		NULL	

BankAccount:



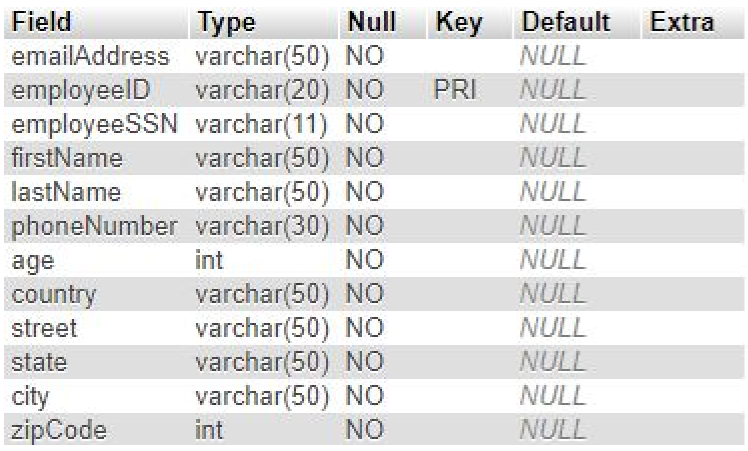
Field	Type	Null	Key	Default	Extra
interestRate	decimal(10,2)	NO		*NULL*	
dateOpened	date	NO		*NULL*	
minBalance	int	NO		*NULL*	
jointAccount	tinyint(1)	NO		*NULL*	
bankAccountNumber	varchar(30)	NO	PRI	*NULL*	
balance	int	NO		*NULL*	
typeOfBankAccount	varchar(20)	NO		*NULL*	
memberSSN	varchar(11)	NO	MUL	*NULL*	

CanWithdraw:



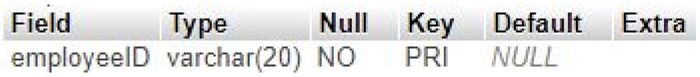
Field	Type	Null	Key	Default	Extra
transactionID	varchar(50)	NO	PRI	*NULL*	
bankID	varchar(50)	NO	MUL	*NULL*	

Employee:



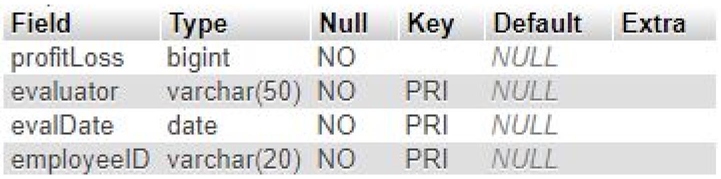
Field	Type	Null	Key	Default	Extra
emailAddress	varchar(50)	NO		*NULL*	
employeeID	varchar(20)	NO	PRI	*NULL*	
employeeSSN	varchar(11)	NO		*NULL*	
firstName	varchar(50)	NO		*NULL*	
lastName	varchar(50)	NO		*NULL*	
phoneNumber	varchar(30)	NO		*NULL*	
age	int	NO		*NULL*	
country	varchar(50)	NO		*NULL*	
street	varchar(50)	NO		*NULL*	
state	varchar(50)	NO		*NULL*	
city	varchar(50)	NO		*NULL*	
zipCode	int	NO		*NULL*	

Evaluates:



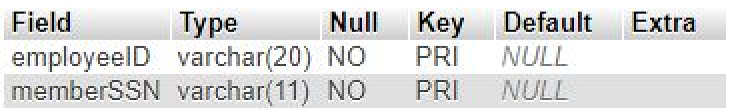
Field	Type	Null	Key	Default	Extra
employeeID	varchar(20)	NO	PRI	*NULL*	

EvaluationEvaluated:



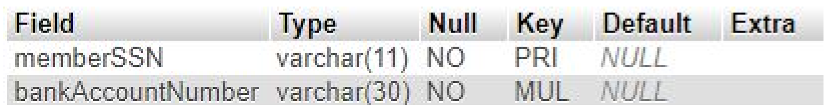
Field	Type	Null	Key	Default	Extra
profitLoss	bigint	NO		*NULL*	
evaluator	varchar(50)	NO	PRI	*NULL*	
evalDate	date	NO	PRI	*NULL*	
employeeID	varchar(20)	NO	PRI	*NULL*	

hasClients:



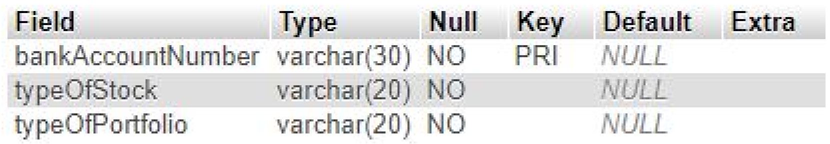
Field	Type	Null	Key	Default	Extra
employeeID	varchar(20)	NO	PRI	*NULL*	
memberSSN	varchar(11)	NO	PRI	*NULL*	

hasHoldings:



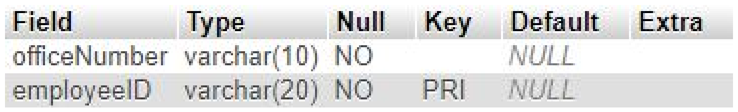
Field	Type	Null	Key	Default	Extra
memberSSN	varchar(11)	NO	PRI	*NULL*	
bankAccountNumber	varchar(30)	NO	MUL	*NULL*	

InvestmentAccount:



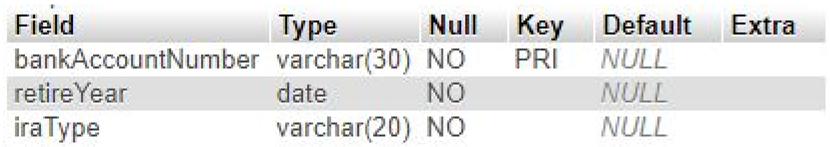
Field	Type	Null	Key	Default	Extra
bankAccountNumber	varchar(30)	NO	PRI	*NULL*	
typeOfStock	varchar(20)	NO		*NULL*	
typeOfPortfolio	varchar(20)	NO		*NULL*	

InvestmentAdvisor:



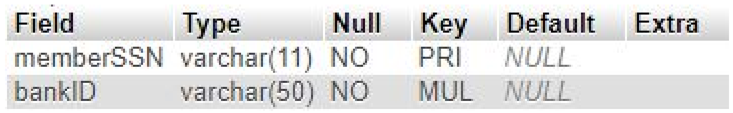
Field	Type	Null	Key	Default	Extra
officeNumber	varchar(10)	NO		*NULL*	
employeeID	varchar(20)	NO	PRI	*NULL*	

IRA:



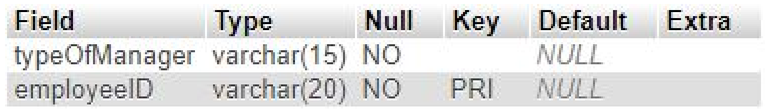
Field	Type	Null	Key	Default	Extra
bankAccountNumber	varchar(30)	NO	PRI	*NULL*	
retireYear	date	NO		*NULL*	
iraType	varchar(20)	NO		*NULL*	

isMemberOf:



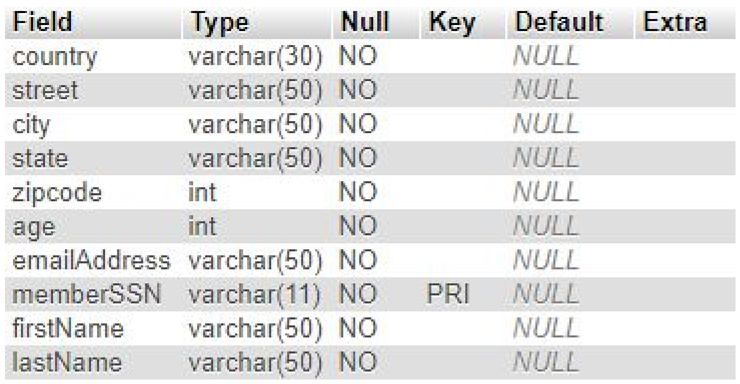
Field	Type	Null	Key	Default	Extra
memberSSN	varchar(11)	NO	PRI	*NULL*	
bankID	varchar(50)	NO	MUL	*NULL*	

Manager:



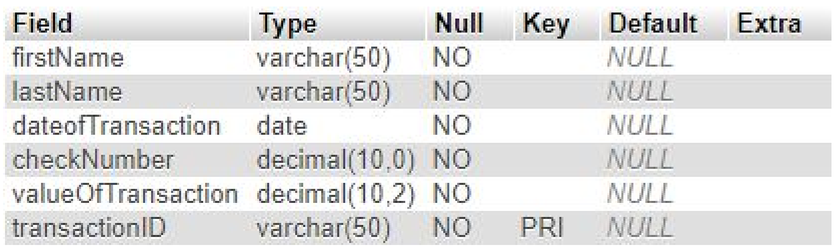
Field	Type	Null	Key	Default	Extra
typeOfManager	varchar(15)	NO		*NULL*	
employeeID	varchar(20)	NO	PRI	*NULL*	

Member:



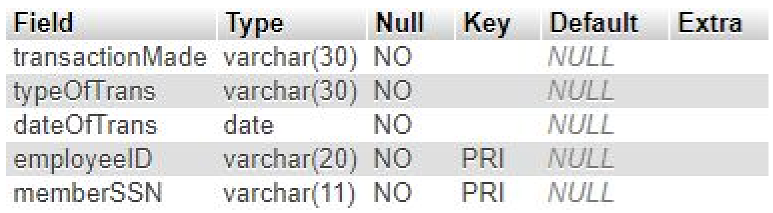
Field	Type	Null	Key	Default	Extra
country	varchar(30)	NO		*NULL*	
street	varchar(50)	NO		*NULL*	
city	varchar(50)	NO		*NULL*	
state	varchar(50)	NO		*NULL*	
zipcode	int	NO		*NULL*	
age	int	NO		*NULL*	
emailAddress	varchar(50)	NO		*NULL*	
memberSSN	varchar(11)	NO	PRI	*NULL*	
firstName	varchar(50)	NO		*NULL*	
lastName	varchar(50)	NO		*NULL*	

NonMember:



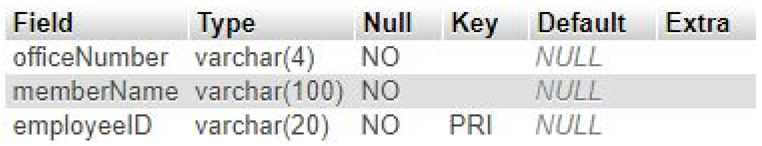
Field	Type	Null	Key	Default	Extra
firstName	varchar(50)	NO		*NULL*	
lastName	varchar(50)	NO		*NULL*	
dateofTransaction	date	NO		*NULL*	
checkNumber	decimal(10,0)	NO		*NULL*	
valueOfTransaction	decimal(10,2)	NO		*NULL*	
transactionID	varchar(50)	NO	PRI	*NULL*	

<u>processTransactions:</u>



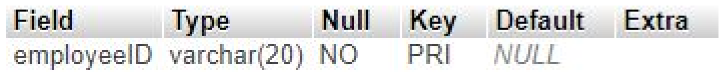
Field	Type	Null	Key	Default	Extra
transactionMade	varchar(30)	NO		*NULL*	
typeOfTrans	varchar(30)	NO		*NULL*	
dateOfTrans	date	NO		*NULL*	
employeeID	varchar(20)	NO	PRI	*NULL*	
memberSSN	varchar(11)	NO	PRI	*NULL*	

<u>SalesAssociate:</u>



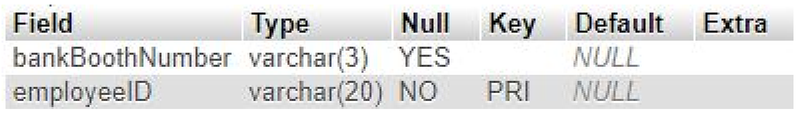
Field	Type	Null	Key	Default	Extra
officeNumber	varchar(4)	NO		*NULL*	
memberName	varchar(100)	NO		*NULL*	
employeeID	varchar(20)	NO	PRI	*NULL*	

<u>SalesPerson:</u>



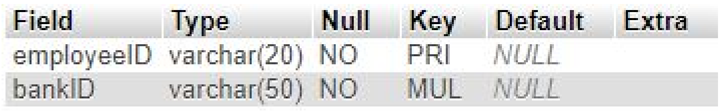
Field	Type	Null	Key	Default	Extra
employeeID	varchar(20)	NO	PRI	*NULL*	

<u>Teller:</u>



Field	Type	Null	Key	Default	Extra
bankBoothNumber	varchar(3)	YES		*NULL*	
employeeID	varchar(20)	NO	PRI	*NULL*	

<u>Work:</u>



Field	Type	Null	Key	Default	Extra
employeeID	varchar(20)	NO	PRI	*NULL*	
bankID	varchar(50)	NO	MUL	*NULL*	

SECTION 4: SAMPLES OF INSERT, DELETE, AND UPDATE

Insert new tuple into a table
INSERT INTO \`Bank\` (\`phoneNumber\`, \`bankID\`, \`bankName\`, \`swiftNumber\`, \`routingNumber\`, \`country\`, \`street\`, \`city\`, \`zipCode\`, \`state\`) VALUES ('2093384911', '123456789', 'JPMorganChase', 'SDFG87F78DG4654', '2418456484785458', 'US', '1300 Marin Ln', 'Modesto', '95358', 'California');

Delete a tuple from a table
DELETE FROM `Bank` WHERE swiftNumber = "DF56G4FD854GF5";
Update a tuple from table Bank

UPDATE `Bank` SET `phoneNumber` = '123456789', `bankName` = 'WellsFargo', `swiftNumber` = 'DF56G4FD854GF5', `street` = '717 Marin Lane' WHERE `Bank`.`bankID` = '123456789';

Observations

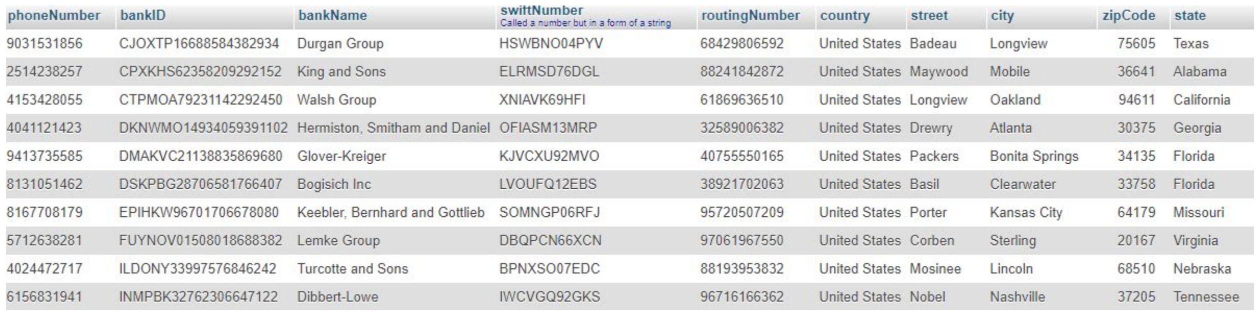
For manual insert SQL Code, the application expects all values to be placed in ‘ ‘, so that the syntax knows where the value begins. The type of value then renders the value inside of the ‘ ‘ as that type of value, not a string. We are storing phoneNumbers as a varchar because we ran into the error where INT had issues.

SECTION 5: WHERE WE GOT OUR DATA

We got our data from the realistic data generator website https://www.mockaroo.com/. We then used the website https://sqlizer.io/#/ to convert our files into MYSQL so we could insert the data easily into our database. Some of the more tricky strings, we had to use the regular expression on mockaroo to create custom inputs for our data such as the different types of IRA accounts, types of stocks, investment account types, and the different types of ID for multiple different tables.

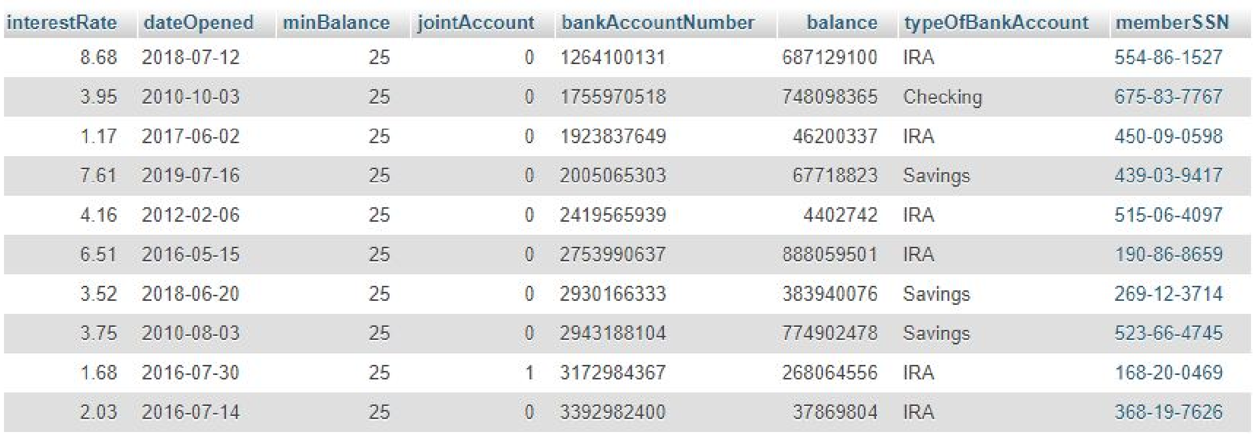
SECTION 6: SAMPLE OF TUPLES FOR EACH RELATIONS

Bank:



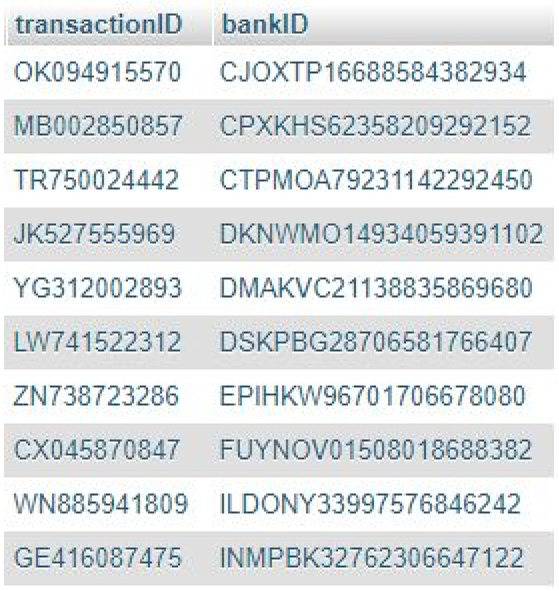
phoneNumber	bankID	bankName	swiftNumber Called a number but in a form of a string	routingNumber	country	street	city	zipCode	state
9031531856	CJOXTP16688584382934	Durgan Group	HSWBNO04PYV	68429806592	United States	Badeau	Longview	75605	Texas
2514238257	CPXKHS62358209292152	King and Sons	ELRMSD76DGL	88241842872	United States	Maywood	Mobile	36641	Alabama
4153428055	CTPMOA79231142292450	Walsh Group	XNIAVK69HFI	61869636510	United States	Longview	Oakland	94611	California
4041121423	DKNWMO14934059391102	Hermiston, Smitham and Daniel	OFIASM13MRP	32589006382	United States	Drewry	Atlanta	30375	Georgia
9413735585	DMAKVC21138835869680	Glover-Kreiger	KJVCXU92MVO	40755550165	United States	Packers	Bonita Springs	34135	Florida
8131051462	DSKPBG28706581766407	Bogisich Inc	LVOUFQ12EBS	38921702063	United States	Basil	Clearwater	33758	Florida
8167708179	EPIHKW96701706678080	Keebler, Bernhard and Gottlieb	SOMNGP06RFJ	95720507209	United States	Porter	Kansas City	64179	Missouri
5712638281	FUYNOV01508018688382	Lemke Group	DBQPCN66XCN	97061967550	United States	Corben	Sterling	20167	Virginia
4024472717	ILDONY33997576846242	Turcotte and Sons	BPNXSO07EDC	88193953832	United States	Mosinee	Lincoln	68510	Nebraska
6156831941	INMPBK32762306647122	Dibbert-Lowe	IWCVGQ92GKS	96716166362	United States	Nobel	Nashville	37205	Tennessee

BankAccount:



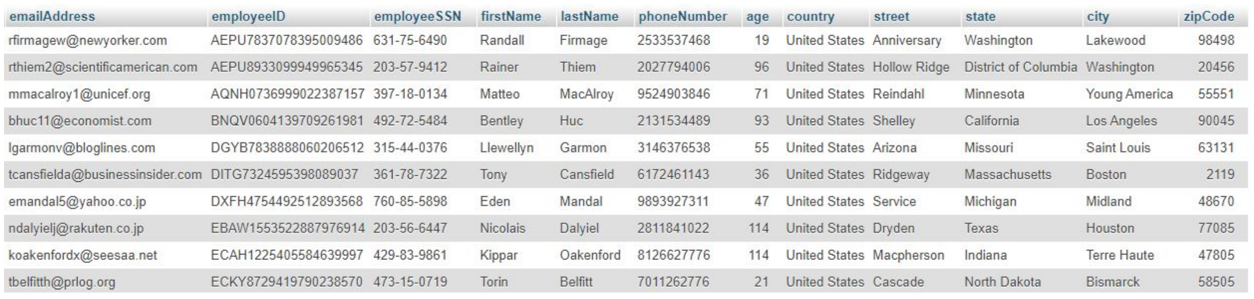
interestRate	dateOpened	minBalance	jointAccount	bankAccountNumber	balance	typeOfBankAccount	memberSSN
8.68	2018-07-12	25	0	1264100131	687129100	IRA	554-86-1527
3.95	2010-10-03	25	0	1755970518	748098365	Checking	675-83-7767
1.17	2017-06-02	25	0	1923837649	46200337	IRA	450-09-0598
7.61	2019-07-16	25	0	2005065303	67718823	Savings	439-03-9417
4.16	2012-02-06	25	0	2419565939	4402742	IRA	515-06-4097
6.51	2016-05-15	25	0	2753990637	888059501	IRA	190-86-8659
3.52	2018-06-20	25	0	2930166333	383940076	Savings	269-12-3714
3.75	2010-08-03	25	0	2943188104	774902478	Savings	523-66-4745
1.68	2016-07-30	25	1	3172984367	268064556	IRA	168-20-0469
2.03	2016-07-14	25	0	3392982400	37869804	IRA	368-19-7626

<u>CanWithdraw:</u>



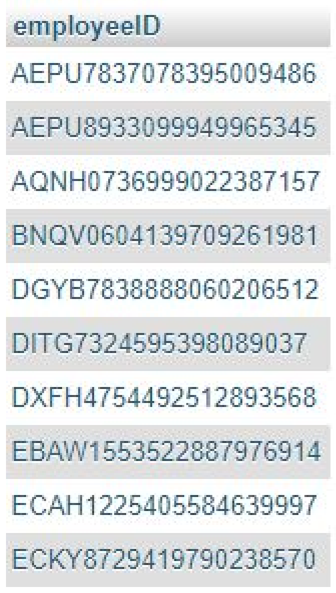
transactionID	bankID
OK094915570	CJOXTP16688584382934
MB002850857	CPXKHS62358209292152
TR750024442	CTPMOA79231142292450
JK527555969	DKNWMO14934059391102
YG312002893	DMAKVC21138835869680
LW741522312	DSKPBG28706581766407
ZN738723286	EPIHKW96701706678080
CX045870847	FUYNOV01508018688382
WN885941809	ILDONY33997576846242
GE416087475	INMPBK32762306647122

<u>Employee:</u>



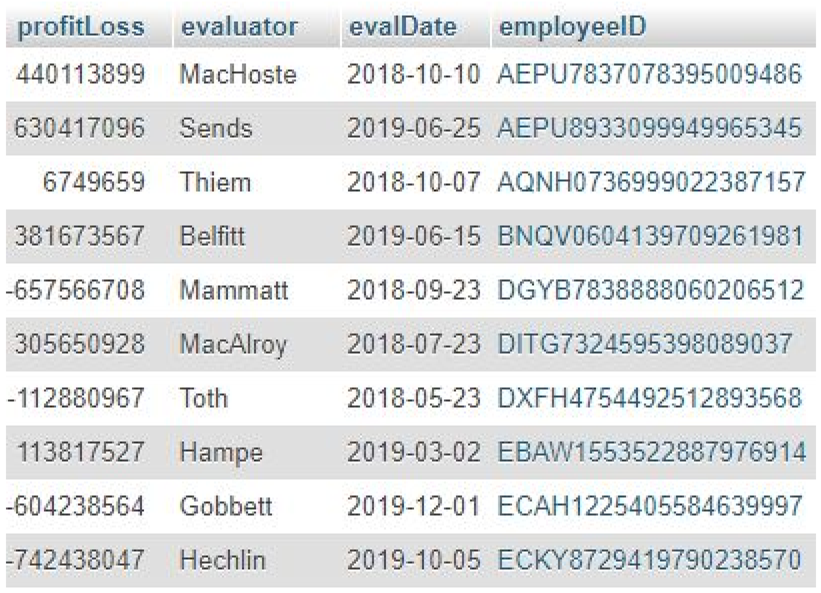
emailAddress	employeeID	employeeSSN	firstName	lastName	phoneNumber	age	country	street	state	city	zipCode
rfirmagew@newyorker.com	AEPU7837078395009486	631-75-6490	Randall	Firmage	2533537468	19	United States	Anniversary	Washington	Lakewood	98498
rthiem2@scientificamerican.com	AEPU8933099949965345	203-57-9412	Rainer	Thiem	2027794006	96	United States	Hollow Ridge	District of Columbia	Washington	20456
mmacalroy1@unicef.org	AQNH0736999022387157	397-18-0134	Matteo	MacAlroy	9524903846	71	United States	Reindahl	Minnesota	Young America	55551
bhuc11@economist.com	BNQV0604139709261981	492-72-5484	Bentley	Huc	2131534489	93	United States	Shelley	California	Los Angeles	90045
lgarmonv@bloglines.com	DGYB7838888060206512	315-44-0376	Llewellyn	Garmon	3146376538	55	United States	Arizona	Missouri	Saint Louis	63131
tcansfielda@businessinsider.com	DITG7324595398089037	361-78-7322	Tony	Cansfield	6172461143	36	United States	Ridgeway	Massachusetts	Boston	2119
emandal5@yahoo.co.jp	DXFH4754492512893568	760-85-5898	Eden	Mandal	9893927311	47	United States	Service	Michigan	Midland	48670
ndalyielj@rakuten.co.jp	EBAW1553522887976914	203-56-6447	Nicolais	Dalyiel	2811841022	114	United States	Dryden	Texas	Houston	77085
koakenfordx@seesaa.net	ECAH1225405584639997	429-83-9861	Kippar	Oakenford	8126627776	114	United States	Macpherson	Indiana	Terre Haute	47805
tbelfitth@prlog.org	ECKY8729419790238570	473-15-0719	Torin	Belfitt	7011262776	21	United States	Cascade	North Dakota	Bismarck	58505

<u>Evaluates:</u>



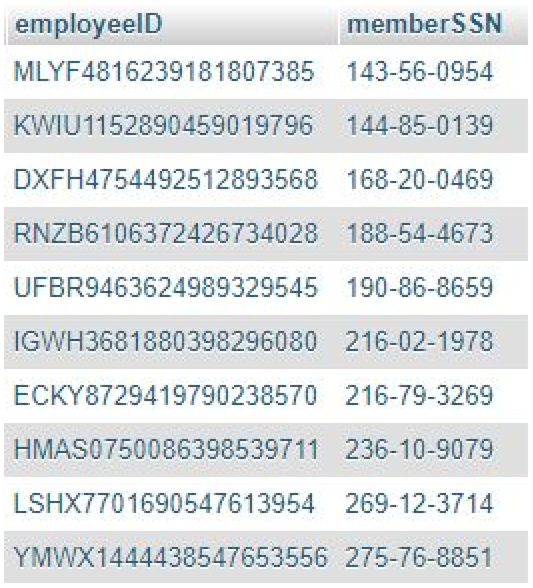
employeeID
AEPU7837078395009486
AEPU8933099949965345
AQNH0736999022387157
BNQV0604139709261981
DGYB7838888060206512
DITG7324595398089037
DXFH4754492512893568
EBAW1553522887976914
ECAH1225405584639997
ECKY8729419790238570

<u>EvaluationEvaluated:</u>



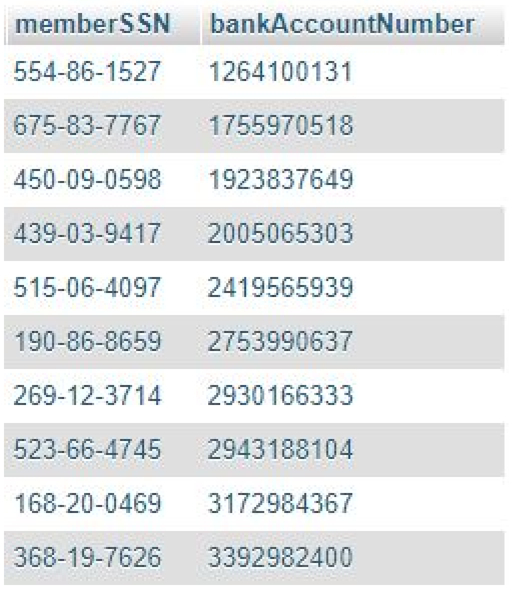
profitLoss	evaluator	evalDate	employeeID
440113899	MacHoste	2018-10-10	AEPU7837078395009486
630417096	Sends	2019-06-25	AEPU8933099949965345
6749659	Thiem	2018-10-07	AQNH0736999022387157
381673567	Belfitt	2019-06-15	BNQV0604139709261981
-657566708	Mammatt	2018-09-23	DGYB7838888060206512
305650928	MacAlroy	2018-07-23	DITG7324595398089037
-112880967	Toth	2018-05-23	DXFH4754492512893568
113817527	Hampe	2019-03-02	EBAW1553522887976914
-604238564	Gobbett	2019-12-01	ECAH1225405584639997
-742438047	Hechlin	2019-10-05	ECKY8729419790238570

<u>hasClients:</u>



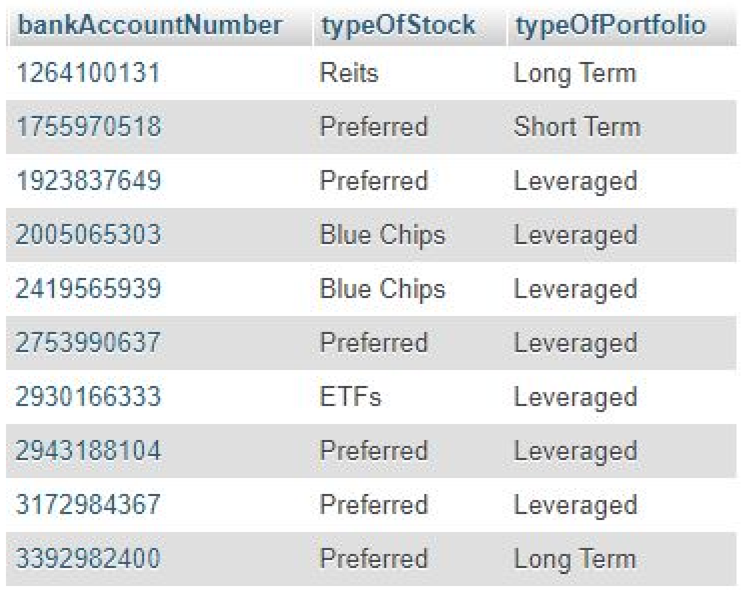
employeeID	memberSSN
MLYF4816239181807385	143-56-0954
KWIU1152890459019796	144-85-0139
DXFH4754492512893568	168-20-0469
RNZB6106372426734028	188-54-4673
UFBR9463624989329545	190-86-8659
IGWH3681880398296080	216-02-1978
ECKY8729419790238570	216-79-3269
HMAS0750086398539711	236-10-9079
LSHX7701690547613954	269-12-3714
YMWX1444438547653556	275-76-8851

hasHoldings:



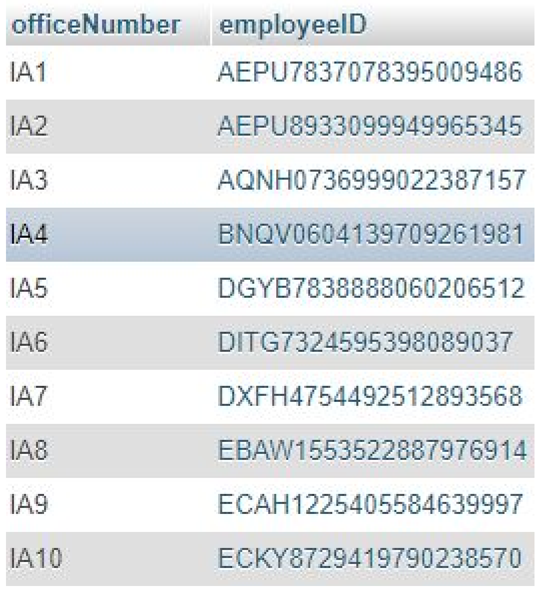
memberSSN	bankAccountNumber
554-86-1527	1264100131
675-83-7767	1755970518
450-09-0598	1923837649
439-03-9417	2005065303
515-06-4097	2419565939
190-86-8659	2753990637
269-12-3714	2930166333
523-66-4745	2943188104
168-20-0469	3172984367
368-19-7626	3392982400

InvestmentAccount:



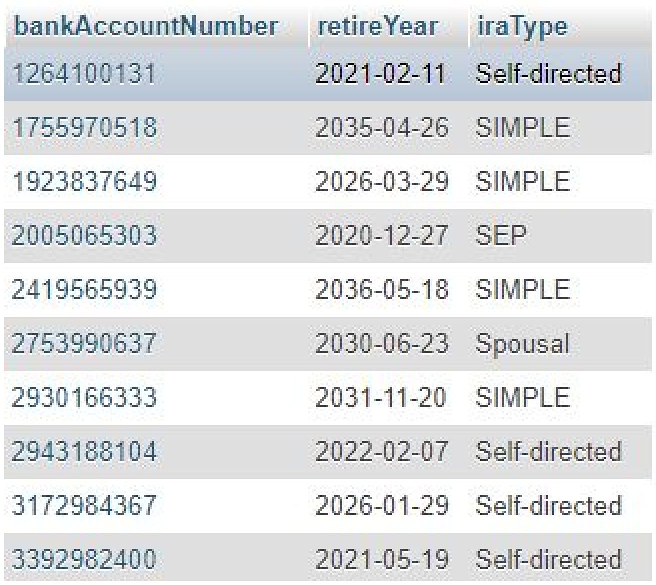
bankAccountNumber	typeOfStock	typeOfPortfolio
1264100131	Reits	Long Term
1755970518	Preferred	Short Term
1923837649	Preferred	Leveraged
2005065303	Blue Chips	Leveraged
2419565939	Blue Chips	Leveraged
2753990637	Preferred	Leveraged
2930166333	ETFs	Leveraged
2943188104	Preferred	Leveraged
3172984367	Preferred	Leveraged
3392982400	Preferred	Long Term

InvestmentAdvisor:



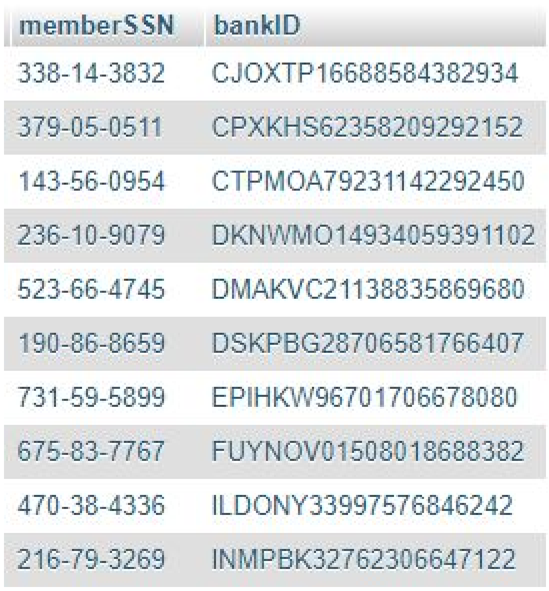
officeNumber	employeeID
IA1	AEPU7837078395009486
IA2	AEPU8933099949965345
IA3	AQNH0736999022387157
IA4	BNQV0604139709261981
IA5	DGYB7838888060206512
IA6	DITG7324595398089037
IA7	DXFH4754492512893568
IA8	EBAW1553522887976914
IA9	ECAH1225405584639997
IA10	ECKY8729419790238570

IRA:



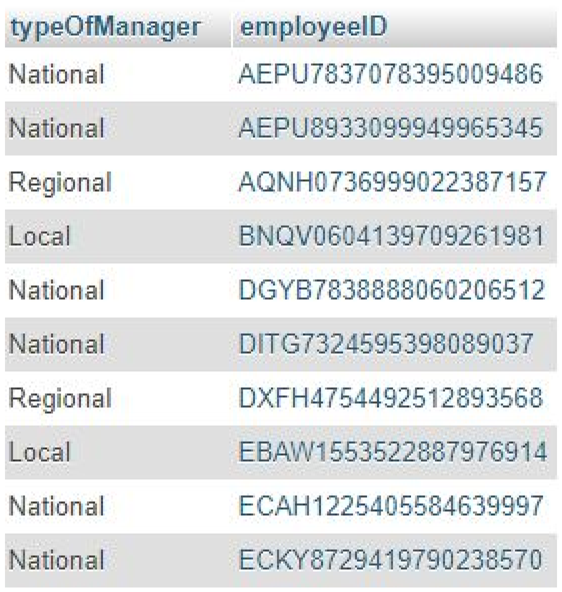
bankAccountNumber	retireYear	iraType
1264100131	2021-02-11	Self-directed
1755970518	2035-04-26	SIMPLE
1923837649	2026-03-29	SIMPLE
2005065303	2020-12-27	SEP
2419565939	2036-05-18	SIMPLE
2753990637	2030-06-23	Spousal
2930166333	2031-11-20	SIMPLE
2943188104	2022-02-07	Self-directed
3172984367	2026-01-29	Self-directed
3392982400	2021-05-19	Self-directed

isMemberOf:



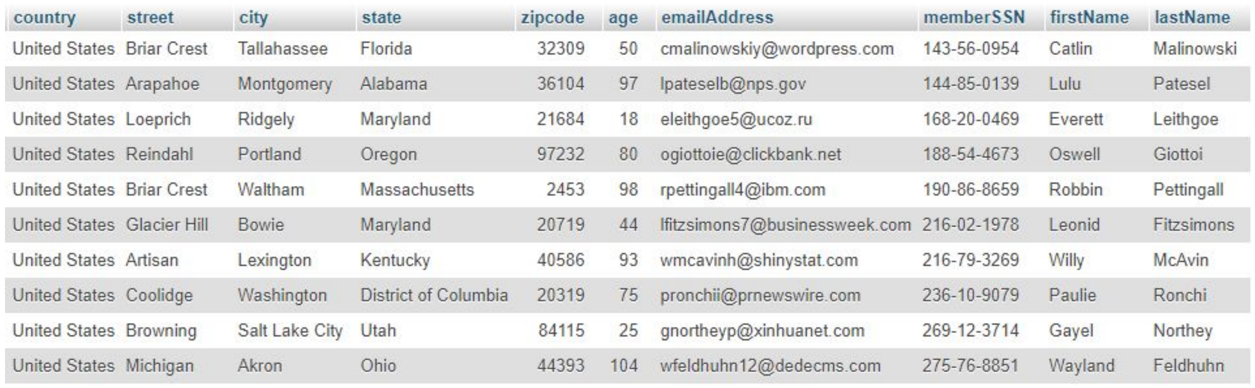
memberSSN	bankID
338-14-3832	CJOXTP16688584382934
379-05-0511	CPXKHS62358209292152
143-56-0954	CTPMOA79231142292450
236-10-9079	DKNWMO14934059391102
523-66-4745	DMAKVC21138835869680
190-86-8659	DSKPBG28706581766407
731-59-5899	EPIHKW96701706678080
675-83-7767	FUYNOV01508018688382
470-38-4336	ILDONY33997576846242
216-79-3269	INMPBK32762306647122

Manager:



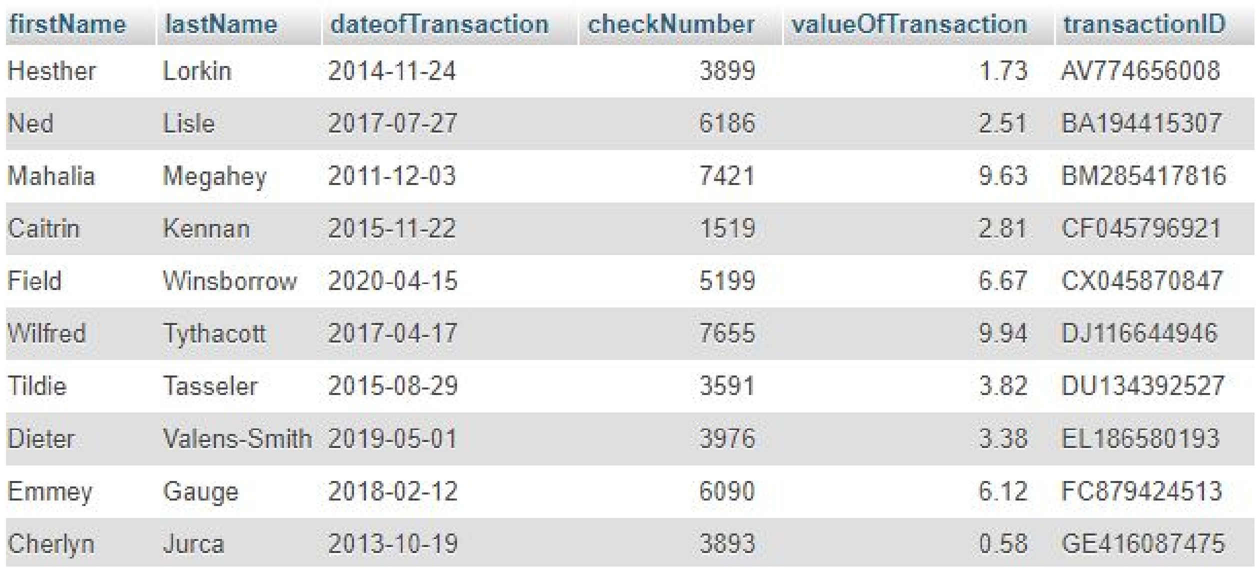
typeOfManager	employeeID
National	AEPU7837078395009486
National	AEPU8933099949965345
Regional	AQNH0736999022387157
Local	BNQV0604139709261981
National	DGYB7838888060206512
National	DITG7324595398089037
Regional	DXFH4754492512893568
Local	EBAW1553522887976914
National	ECAH1225405584639997
National	ECKY8729419790238570

<u>Member:</u>



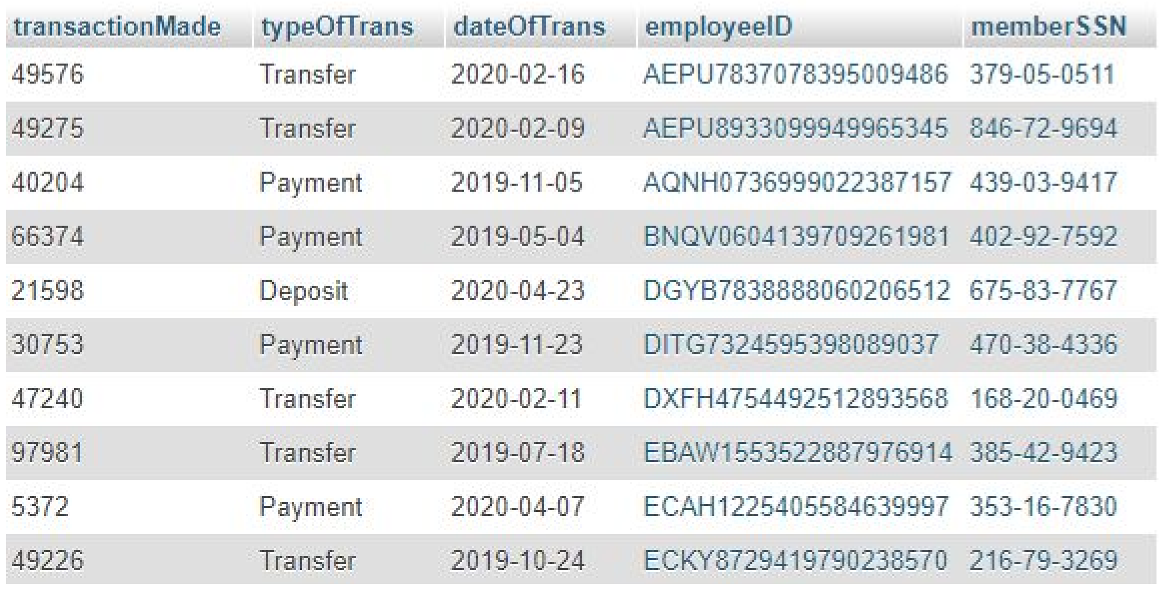
country	street	city	state	zipcode	age	emailAddress	memberSSN	firstName	lastName
United States	Briar Crest	Tallahassee	Florida	32309	50	cmalinowskiy@wordpress.com	143-56-0954	Catlin	Malinowski
United States	Arapahoe	Montgomery	Alabama	36104	97	lpateselb@nps.gov	144-85-0139	Lulu	Patesel
United States	Loeprich	Ridgely	Maryland	21684	18	eleithgoe5@ucoz.ru	168-20-0469	Everett	Leithgoe
United States	Reindahl	Portland	Oregon	97232	80	ogiottoie@clickbank.net	188-54-4673	Oswell	Giottoi
United States	Briar Crest	Waltham	Massachusetts	2453	98	rpettingall4@ibm.com	190-86-8659	Robbin	Pettingall
United States	Glacier Hill	Bowie	Maryland	20719	44	lfitzsimons7@businessweek.com	216-02-1978	Leonid	Fitzsimons
United States	Artisan	Lexington	Kentucky	40586	93	wmcavinh@shinystat.com	216-79-3269	Willy	McAvin
United States	Coolidge	Washington	District of Columbia	20319	75	pronchii@prnewswire.com	236-10-9079	Paulie	Ronchi
United States	Browning	Salt Lake City	Utah	84115	25	gnortheyp@xinhuanet.com	269-12-3714	Gayel	Northey
United States	Michigan	Akron	Ohio	44393	104	wfeldhuhn12@dedecms.com	275-76-8851	Wayland	Feldhuhn

<u>NonMember:</u>



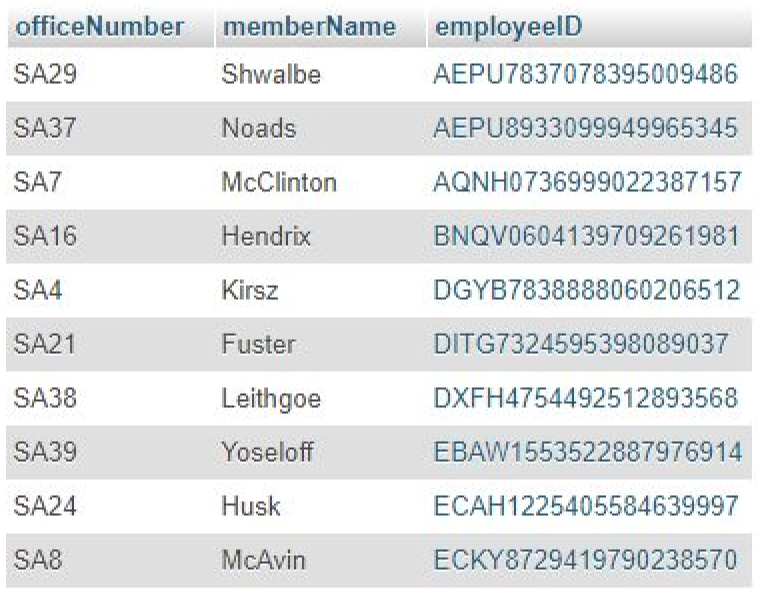
firstName	lastName	dateofTransaction	checkNumber	valueOfTransaction	transactionID
Hesther	Lorkin	2014-11-24	3899	1.73	AV774656008
Ned	Lisle	2017-07-27	6186	2.51	BA194415307
Mahalia	Megahey	2011-12-03	7421	9.63	BM285417816
Caitrin	Kennan	2015-11-22	1519	2.81	CF045796921
Field	Winsborrow	2020-04-15	5199	6.67	CX045870847
Wilfred	Tythacott	2017-04-17	7655	9.94	DJ116644946
Tildie	Tasseler	2015-08-29	3591	3.82	DU134392527
Dieter	Valens-Smith	2019-05-01	3976	3.38	EL186580193
Emmey	Gauge	2018-02-12	6090	6.12	FC879424513
Cherlyn	Jurca	2013-10-19	3893	0.58	GE416087475

processTransactions:



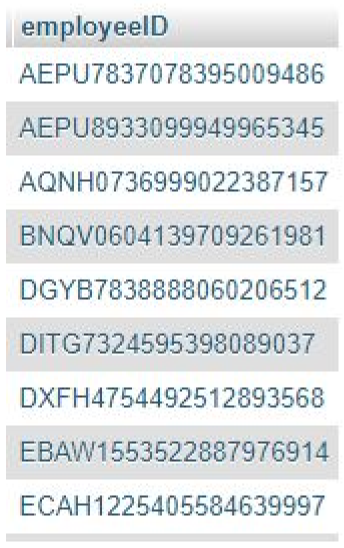
transactionMade	typeOfTrans	dateOfTrans	employeeID	memberSSN
49576	Transfer	2020-02-16	AEPU7837078395009486	379-05-0511
49275	Transfer	2020-02-09	AEPU8933099949965345	846-72-9694
40204	Payment	2019-11-05	AQNH0736999022387157	439-03-9417
66374	Payment	2019-05-04	BNQV0604139709261981	402-92-7592
21598	Deposit	2020-04-23	DGYB7838888060206512	675-83-7767
30753	Payment	2019-11-23	DITG7324595398089037	470-38-4336
47240	Transfer	2020-02-11	DXFH4754492512893568	168-20-0469
97981	Transfer	2019-07-18	EBAW1553522887976914	385-42-9423
5372	Payment	2020-04-07	ECAH1225405584639997	353-16-7830
49226	Transfer	2019-10-24	ECKY8729419790238570	216-79-3269

SalesAssociate:



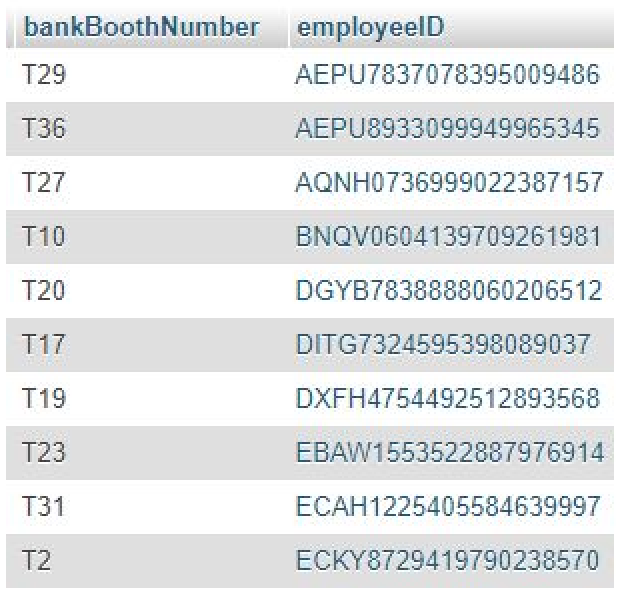
officeNumber	memberName	employeeID
SA29	Shwalbe	AEPU7837078395009486
SA37	Noads	AEPU8933099949965345
SA7	McClinton	AQNH0736999022387157
SA16	Hendrix	BNQV0604139709261981
SA4	Kirsz	DGYB7838888060206512
SA21	Fuster	DITG7324595398089037
SA38	Leithgoe	DXFH4754492512893568
SA39	Yoseloff	EBAW1553522887976914
SA24	Husk	ECAH1225405584639997
SA8	McAvin	ECKY8729419790238570

SalesPerson:



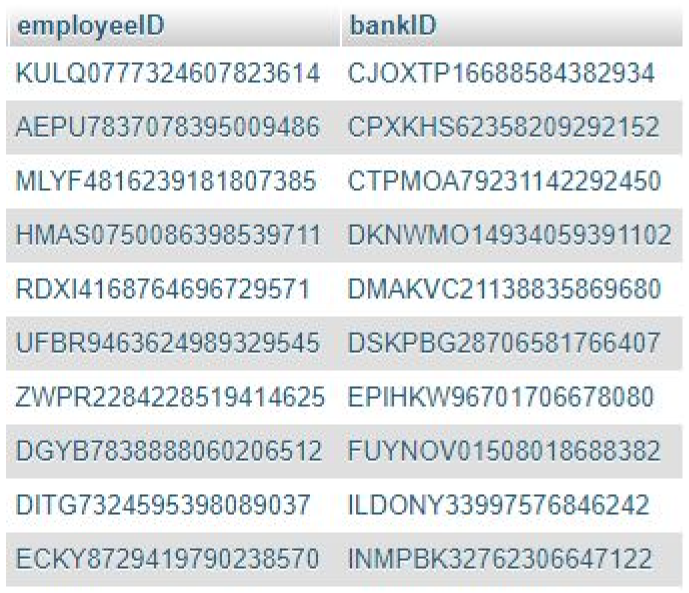
employeeID
AEPU7837078395009486
AEPU8933099949965345
AQNH0736999022387157
BNQV0604139709261981
DGYB7838888060206512
DITG7324595398089037
DXFH4754492512893568
EBAW1553522887976914
ECAH1225405584639997

Teller:



bankBoothNumber	employeeID
T29	AEPU7837078395009486
T36	AEPU8933099949965345
T27	AQNH0736999022387157
T10	BNQV0604139709261981
T20	DGYB7838888060206512
T17	DITG7324595398089037
T19	DXFH4754492512893568
T23	EBAW1553522887976914
T31	ECAH1225405584639997
T2	ECKY8729419790238570

Work:



employeeID	bankID
KULQ0777324607823614	CJOXTP16688584382934
AEPU7837078395009486	CPXKHS62358209292152
MLYF4816239181807385	CTPMOA79231142292450
HMAS0750086398539711	DKNWMO14934059391102
RDXI4168764696729571	DMAKVC21138835869680
UFBR9463624989329545	DSKPBG28706581766407
ZWPR2284228519414625	EPIHKW96701706678080
DGYB7838888060206512	FUYNOV01508018688382
DITG7324595398089037	ILDONY33997576846242
ECKY8729419790238570	INMPBK32762306647122

Part 6:

Project Part - 6

Group Contributions:

Chelcey: Contributed to figuring out what queries we were going to do and assisting in helping write the queries.
Giancarlo: Contributed to fixing queries and figuring out what should be queried.
Karanjit: Contributed to figuring out why some queries were not working properly, and came up with a few queries.

Section 1: List of defined SQL schemas

```
CREATE DATABASE Banksy;
USE Banksy;

CREATE TABLE Member(
country char(30),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
age INT,
emailAddress char(50),
memberSSN char(11) NOT NULL,
firstName char(50),
lastName char(50),
PRIMARY KEY (memberSSN)
);

CREATE TABLE NonMember(
firstName char(50),
lastName char(50),
dateofTransaction date,
checkNumber INT(10),
valueOfTransaction INT,
transactionID char(50) NOT NULL,
PRIMARY KEY (transactionID)
);

CREATE TABLE Bank(
phoneNumber INT (13),
bankID char(50) NOT NULL,
bankName char(50),
swiftNumber char(50),
routingNumber INT(9),
country char(30),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
PRIMARY KEY (bankID)
);

CREATE TABLE Employee(
emailAddress char(50),
employeeID char(20),
employeeSSN char(11),
firstName char(50),
lastName char(50),
age INT,
phoneNumber INT,
country char(50),
street char(50),
city char(50),
state char(50),
zipCode INT(5),
PRIMARY KEY (employeeID)
);

CREATE TABLE BankAccount(
interestRate decimal(p,s),
dateOpened date,
minBalance INT,
jointAccount BOOL,
bankAccountNumber INT(12) NOT NULL,
balance INT,
typeOfBankAccount char(20),
memberSSN char(11),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (memberSSN) references Member(memberSSN) ON DELETE CASCADE
);

CREATE TABLE Manager(
typeOfManager char(15),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) references Employee(employeeID) ON DELETE CASCADE
);

CREATE TABLE EvaluationEvaluated(
profitLoss INT,
evaluator char(50) NOT NULL,
evalDate date NOT NULL,
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID, evalDate, evaluator),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID) ON DELETE
CASCADE
);

CREATE TABLE IRA(
bankAccountNumber INT(12) NOT NULL,
retireYear date,
iraType char(20),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber
ON DELETE CASCADE
);

CREATE TABLE InvestmentAccount(
bankAccountNumber INT(12) NOT NULL,
typeOfStock char(20),
typeOfPortfolio char(20),
PRIMARY KEY (bankAccountNumber),
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber
ON DELETE CASCADE
);

CREATE TABLE SalesPerson(
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES Employee(employeeID) ON DELETE
CASCADE
);

CREATE TABLE SalesAssociate (
officeNumber INT,
memberName char(100),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES SalesPerson(employeeID) ON DELETE
CASCADE
);

CREATE TABLE Teller(
bankBoothNumber INT,
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES SalesPerson(employeeID) ON DELETE
CASCADE
);
```

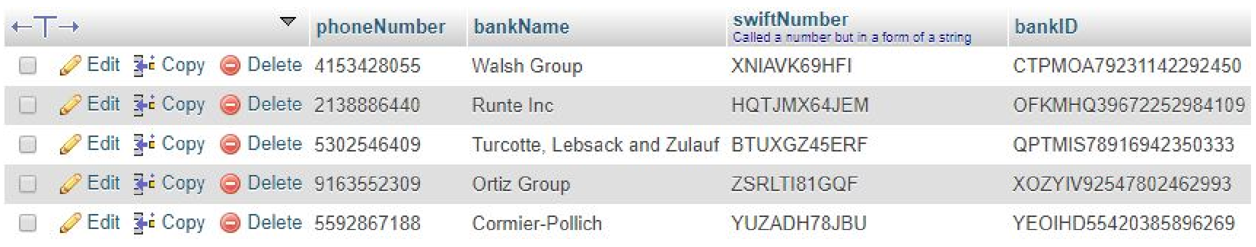
```
CREATE TABLE InvestmentAdvisor(
officeNumber char(10),
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY(employeeID) REFERENCES SalesPerson(employeeID) ON DELETE
CASCADE
);

CREATE TABLE isMemberOf(
memberSSN char(11) NOT NULL,
bankID char(50) NOT NULL,
PRIMARY KEY (memberSSN),
FOREIGN KEY (memberSSN) REFERENCES Member(memberSSN) ON DELETE
CASCADE,
FOREIGN KEY (bankID) REFERENCES Bank(bankID) ON DELETE CASCADE
);

CREATE TABLE hasHoldings(
memberSSN char(11),
bankAccountNumber INT(12),
PRIMARY KEY (memberSSN, bankAccountNumber),
FOREIGN KEY (memberSSN) REFERENCES Member(memberSSN) ON DELETE
CASCADE,
FOREIGN KEY (bankAccountNumber) REFERENCES BankAccount(bankAccountNumber)
ON DELETE CASCADE
);

CREATE TABLE canWithdraw(
transactionID char(50) NOT NULL,
bankID char(50) NOT NULL,
PRIMARY KEY (transactionID),
FOREIGN KEY (transactionID) REFERENCES NonMember(transactionID) ON DELETE
CASCADE,
FOREIGN KEY (bankID) REFERENCES Bank(bankID) ON DELETE CASCADE
);

CREATE TABLE Work(
employeeID char(20),
bankID char(50) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID),
FOREIGN KEY (bankID) REFERENCES Bank(bankID)
);

CREATE TABLE Evaluates(
employeeID char(20) NOT NULL,
PRIMARY KEY (employeeID),
FOREIGN KEY (employeeID) REFERENCES Employee(employeeID) ON DELETE
CASCADE
);

CREATE TABLE processTransactions(
transactionMade char(30),
typeOfTrans char(30),
dateOfTrans date,
employeeID char(20) NOT NULL,
memberSSN char(11) NOT NULL,
PRIMARY KEY (employeeID, memberSSN),
FOREIGN KEY (employeeID) REFERENCES Teller(employeeID),
FOREIGN KEY (memberSSN) REFERENCES Member (memberSSN)
);


CREATE TABLE hasClients(
employeeID char(20)NOT NULL,
memberSSN char(11)NOT NULL,
PRIMARY KEY (employeeID, memberSSN),
FOREIGN KEY (employeeID) REFERENCES SalesAssociate(employeeID),
FOREIGN KEY (memberSSN) REFERENCES Member (memberSSN)
);
```

Section 2: 5 different queries

None of our official 5 outputs were truncated

1. **Grabs a bank name, phone number, and swift number from banks only in California.**

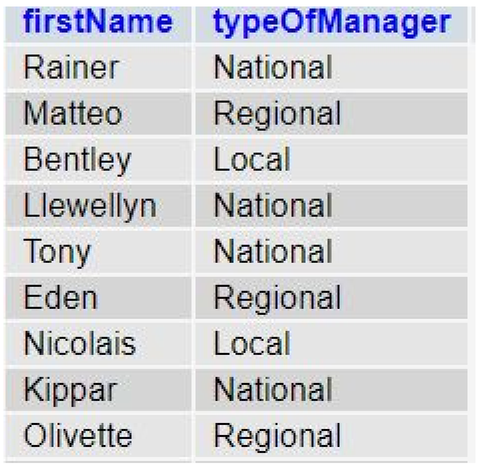
```
SELECT phoneNumber, bankName, swiftNumber, bankID FROM Bank WHERE
state='California';
```



←T→	phoneNumber	bankName	swiftNumber Called a number but in a form of a string	bankID
Edit Copy Delete	4153428055	Walsh Group	XNIAVK69HFI	CTPMOA79231142292450
Edit Copy Delete	2138886440	Runte Inc	HQTJMX64JEM	OFKMHQ39672252984109
Edit Copy Delete	5302546409	Turcotte, Lebsack and Zulauf	BTUXGZ45ERF	QPTMIS78916942350333
Edit Copy Delete	9163552309	Ortiz Group	ZSRLTI81GQF	XOZYIV92547802462993
Edit Copy Delete	5592867188	Cormier-Pollich	YUZADH78JBU	YEOIHD55420385896269

2. **Prints out the first name and the type of manager, and who is over the age of 30.**

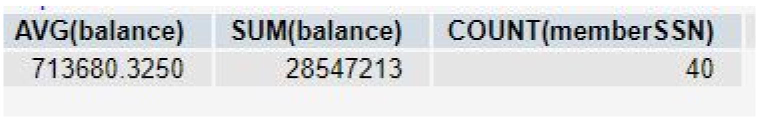
```
SELECT E.firstName, M.typeOfManager FROM Manager M, Employee E WHERE
M.employeeID=E.employeeID and E.age > 30;
```



firstName	typeOfManager
Rainer	National
Matteo	Regional
Bentley	Local
Llewellyn	National
Tony	National
Eden	Regional
Nicolais	Local
Kippar	National
Olivette	Regional

3. **Prints out the average balance of all the accounts of a member in the BankAccount table, and the sum of all of the balances in BankAccount combined.**

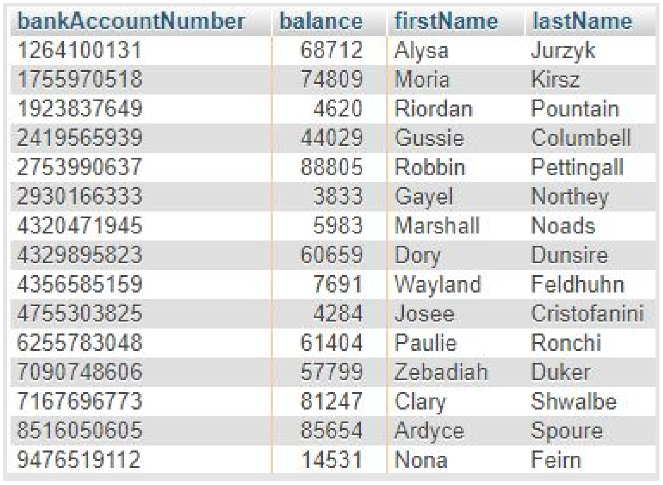
```
SELECT AVG(balance), SUM(balance), COUNT(memberSSN) FROM `BankAccount`;
```



AVG(balance)	SUM(balance)	COUNT(memberSSN)
713680.3250	28547213	40

4. **Prints out a members first and last name, their account balance, and their bank account number that has a balance less than 100,000 and is not a joint account. This was done using a JOIN.**

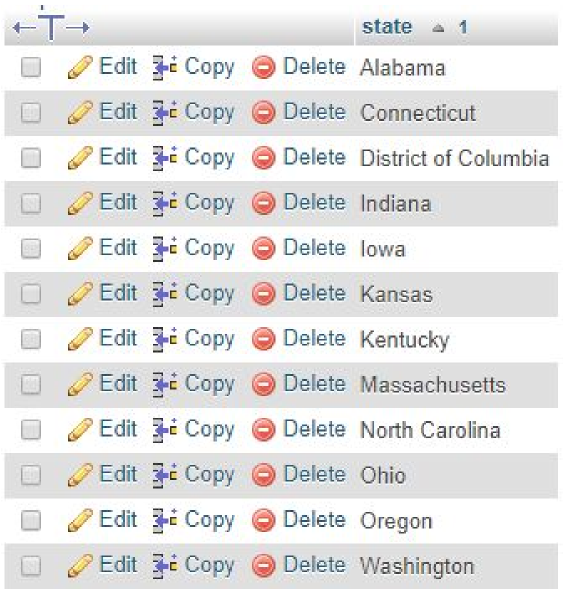
```sql
SELECT BankAccount.bankAccountNumber, BankAccount.balance,
Member.firstName, Member.lastName FROM BankAccount INNER JOIN
Member ON BankAccount.memberSSN=Member.memberSSN WHERE
(BankAccount.balance < 100000 AND BankAccount.jointAccount = 0);
```



bankAccountNumber	balance	firstName	lastName
1264100131	68712	Alysa	Jurzyk
1755970518	74809	Moria	Kirsz
1923837649	4620	Riordan	Pountain
2419565939	44029	Gussie	Columbell
2753990637	88805	Robbin	Pettingall
2930166333	3833	Gayel	Northey
4320471945	5983	Marshall	Noads
4329895823	60659	Dory	Dunsire
4356585159	7691	Wayland	Feldhuhn
4755303825	4284	Josee	Cristofanini
6255783048	61404	Paulie	Ronchi
7090748606	57799	Zebadiah	Duker
7167696773	81247	Clary	Shwalbe
8516050605	85654	Ardyce	Spoure
9476519112	14531	Nona	Feirn

5. **Prints out any states which have less than 2 users, this information can be used to inform a bank on which states they should focus their marketing campaign on.**

```sql
SELECT DISTINCT state FROM Member GROUP by state HAVING COUNT(state) < 2 ORDER BY state;
```



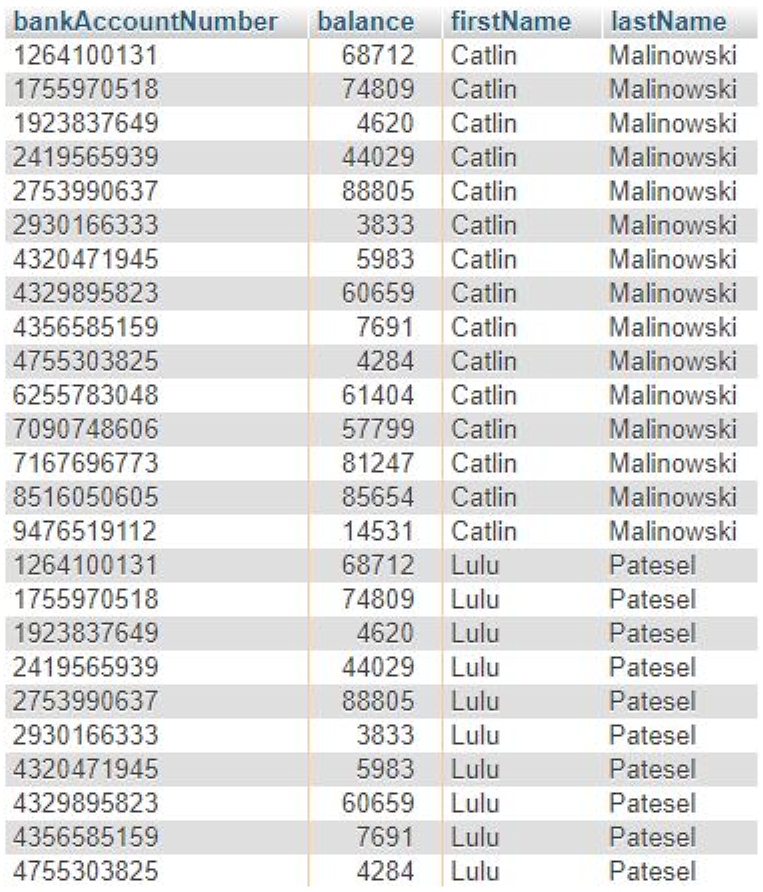
	state
Edit Copy Delete	Alabama
Edit Copy Delete	Connecticut
Edit Copy Delete	District of Columbia
Edit Copy Delete	Indiana
Edit Copy Delete	Iowa
Edit Copy Delete	Kansas
Edit Copy Delete	Kentucky
Edit Copy Delete	Massachusetts
Edit Copy Delete	North Carolina
Edit Copy Delete	Ohio
Edit Copy Delete	Oregon
Edit Copy Delete	Washington

What didn't work: our learning experience:

In this specific query from number 4 (**Prints out a members first and last name, their account balance, and their bank account number that has a balance less than 100,000 and is not a joint account.)** we weren't able to resolve the duplicating issue. This duplicated issue resulted in us getting 600 results from our query search. We tried to use SELECT DISTINCT, it worked for the first test query, however after we started adding more output statements, the results again started printing out duplicates. We remedied this by adding a join statement between the two tables which can be seen in Number 4.

```
SELECT BankAccount.bankAccountNumber, BankAccount.balance, Member.firstName,
Member.lastName FROM BankAccount, Member WHERE ( BankAccount.balance <
100000 AND BankAccount.jointAccount = 0 );
```

This is a truncated version of our output, our original output had 600 results.



bankAccountNumber	balance	firstName	lastName
1264100131	68712	Catlin	Malinowski
1755970518	74809	Catlin	Malinowski
1923837649	4620	Catlin	Malinowski
2419565939	44029	Catlin	Malinowski
2753990637	88805	Catlin	Malinowski
2930166333	3833	Catlin	Malinowski
4320471945	5983	Catlin	Malinowski
4329895823	60659	Catlin	Malinowski
4356585159	7691	Catlin	Malinowski
4755303825	4284	Catlin	Malinowski
6255783048	61404	Catlin	Malinowski
7090748606	57799	Catlin	Malinowski
7167696773	81247	Catlin	Malinowski
8516050605	85654	Catlin	Malinowski
9476519112	14531	Catlin	Malinowski
1264100131	68712	Lulu	Patesel
1755970518	74809	Lulu	Patesel
1923837649	4620	Lulu	Patesel
2419565939	44029	Lulu	Patesel
2753990637	88805	Lulu	Patesel
2930166333	3833	Lulu	Patesel
4320471945	5983	Lulu	Patesel
4329895823	60659	Lulu	Patesel
4356585159	7691	Lulu	Patesel
4755303825	4284	Lulu	Patesel

Part 7:

Group Contributions:

Chelcey: Fully implemented the initial hosting of the database and the team's connections to the database and the server. Ensured that all traffic was only incoming from the team and correctly instantiated the info.php file with Deep's assistance. Also worked on the report.

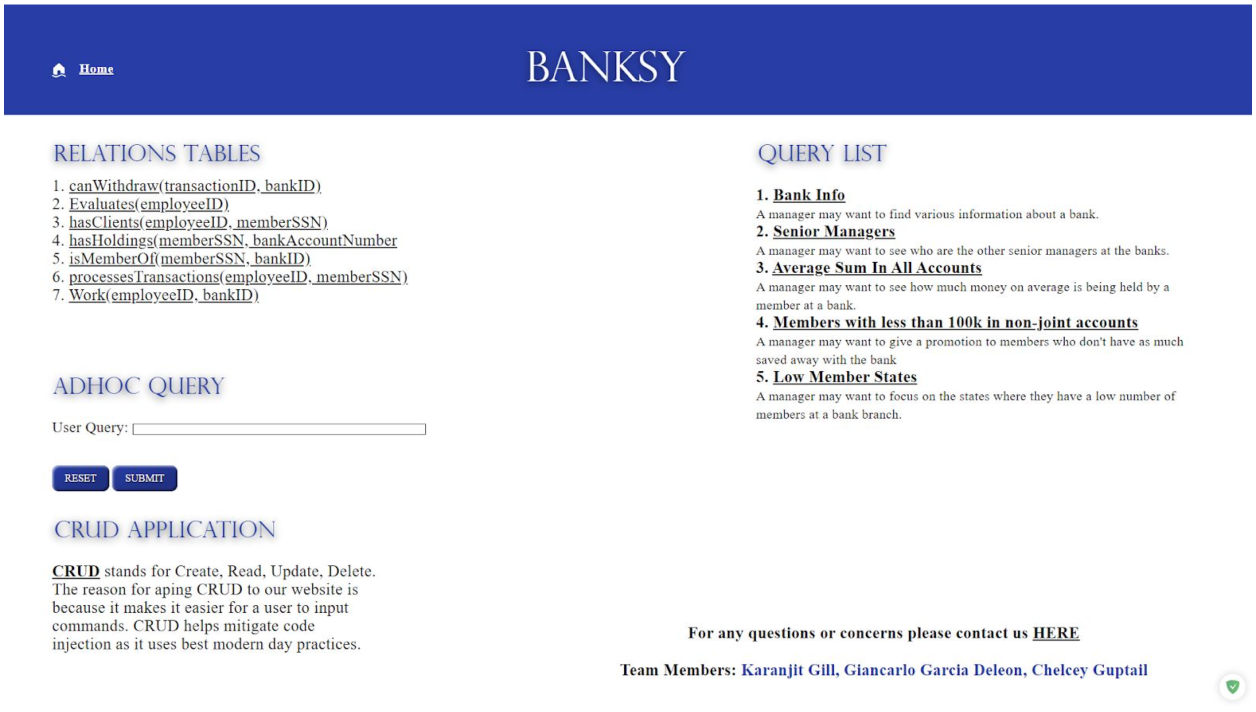
Giancarlo: Focused on the front end design and implementation of the site and sanitized the AD HOC Query. Implemented some forms of protection against SQL injection. Assisted Karanjit on the deployment of the database on the site, and the connection between the database and PHP's script for HTML's forms.

Karanjit: Heavily worked on the organization of all files that the team would use, including index.php, all relations, and queries. Fully worked on the CRUD paradigm where the user has limited control of the system. Created scripts for all relation tables to appear. Designed the query lists and the scripts.

References:

Inspiration for the CRUD Application:
https://www.tutorialrepublic.com/php-tutorial/php-mysql-crud-application.php
Inspiration for the sanitation of AD HOC Query and SQL Injection Protection:
https://www.w3schools.com/sql/sql_injection.asp

1. Explain whether or not somebody who is truly working in the domain of your application would use your web-enabled database, or if there is something missing or intrinsically complex that will deter them from using it. (Is your web-enabled database not powerful enough for the real world, or too complicated for the real world?) For example, if you are in the books domain, would a bookseller use your database? It would be ideal if you had access to a real application domain person; you could just ask him/her to visit your web page and give you feedback. Alternately, look at "similar" systems or web pages and see if they are doing something differently and if your approach is better (or worse) and why.
    - I believe that our website would be useful if someone were working in the domain of our application. Of course, everything that is provided on the website is not all that a user would need, but to find short, quick answers to the queries that we have provided would do the job.
    - The queries that we have provided are able to be used in real life. Banksy is able to give quick and simple information that one working with or at a bank may want to know at the touch of a hand.
2. Identify, in English, where I can find, in your source code, one (or more) safety checks to the code that interacts with your database, to prevent some type of SQL injection attacks. These do not need to be complicated; I just want you to illustrate your understanding of the danger of SQL injection by blocking some possible attack. (A simple if-statement, in the correct location, will be sufficient.)
    - The sanitize starts in userQuery on line 29, which is calling a function from our file phpfuncs.php in the cs4250 directory, from that file, it runs the check() function from lines 3 to line 39, from there we check if the command is one of our allowed commands or not.


Home

BANKSY

RELATIONS TABLES

1. canWithdraw(transactionID, bankID)
2. Evaluates(employeeID)
3. hasClients(employeeID, memberSSN)
4. hasHoldings(memberSSN, bankAccountNumber
5. isMemberOf(memberSSN, bankID)
6. processesTransactions(employeeID, memberSSN)
7. Work(employeeID, bankID)

ADHOC QUERY

User Query:

RESET SUBMIT

CRUD APPLICATION

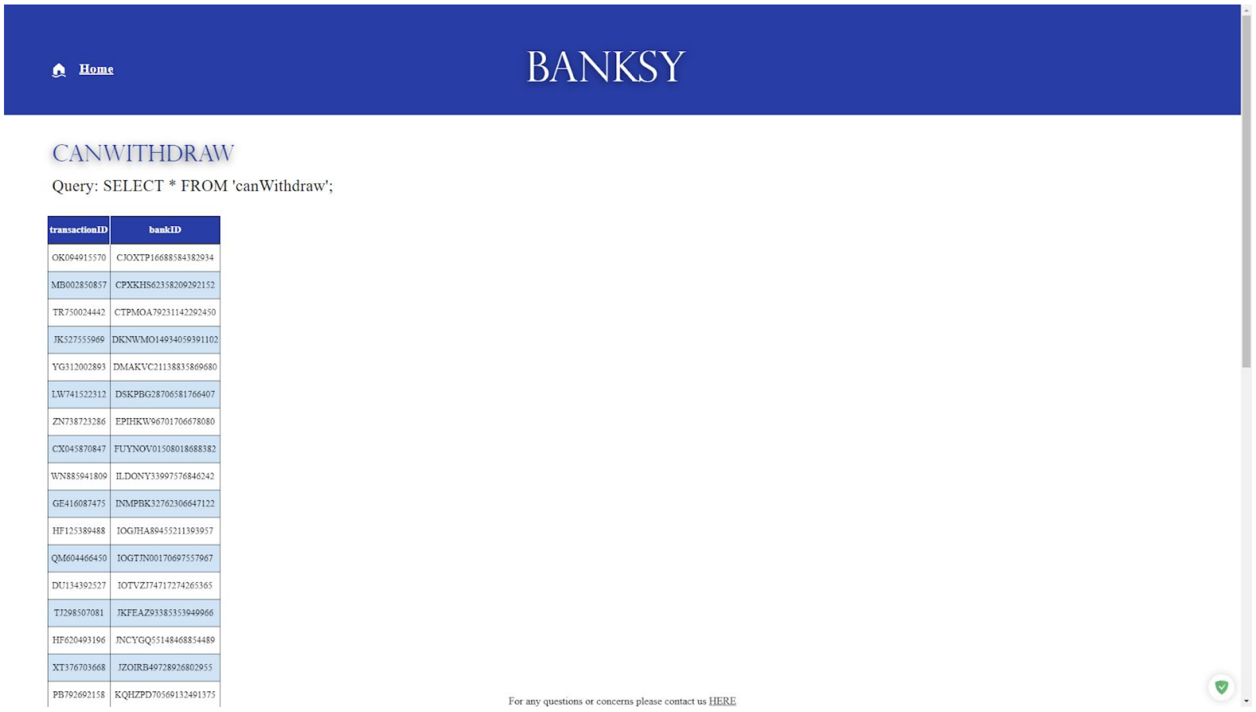
CRUD stands for Create, Read, Update, Delete. The reason for aping CRUD to our website is because it makes it easier for a user to input commands. CRUD helps mitigate code injection as it uses best modern day practices.

QUERY LIST

1. **Bank Info**
A manager may want to find various information about a bank.
2. **Senior Managers**
A manager may want to see who are the other senior managers at the banks.
3. **Average Sum In All Accounts**
A manager may want to see how much money on average is being held by a member at a bank.
4. **Members with less than 100k in non-joint accounts**
A manager may want to give a promotion to members who don't have as much saved away with the bank
5. **Low Member States**
A manager may want to focus on the states where they have a low number of members at a bank branch.

For any questions or concerns please contact us HERE

Team Members: Karanjit Gill, Giancarlo Garcia Deleon, Chelcey Guptail


Home

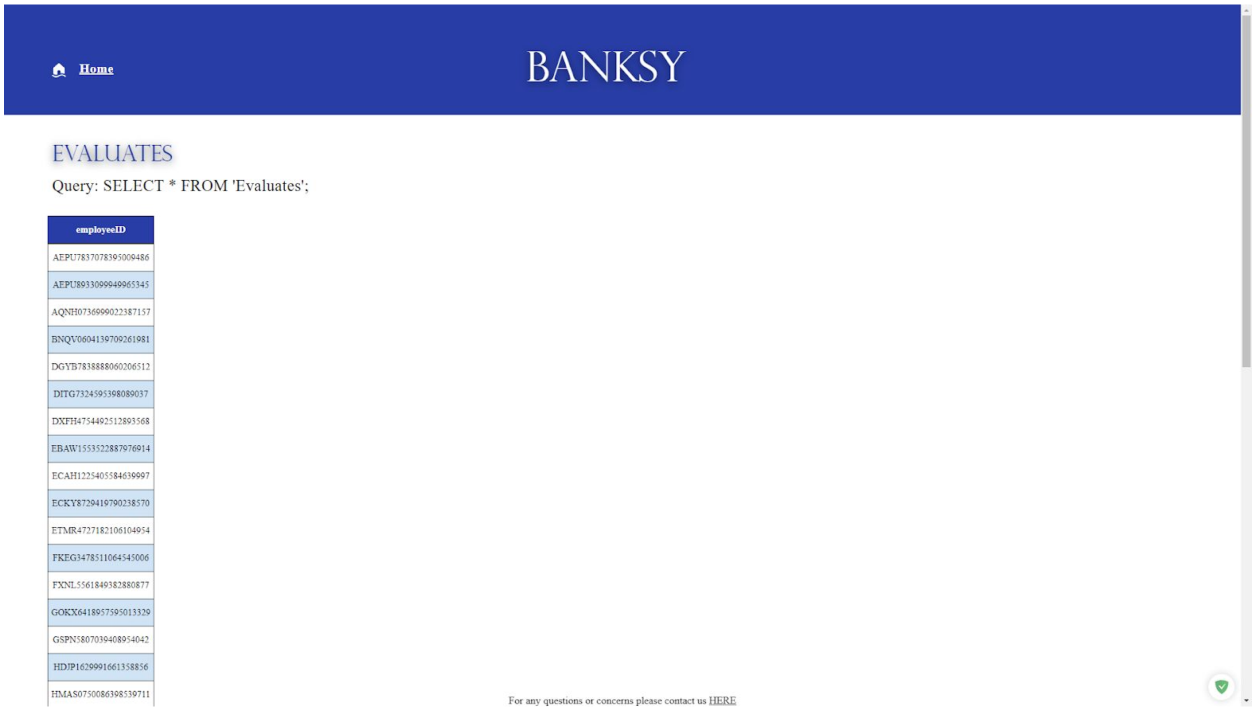
BANKSY

CANWITHDRAW

Query: SELECT * FROM 'canWithdraw';

transactionID	bankID
OK094915570	CJOXTP16688584382934
MB002850857	CPXKHS62358209292152
TR750024442	CTPMOA79231142292450
JK527555969	DKNWMO14934059391102
YG312002893	DMAKVC21138835869680
LW741522312	DSKPBG28706581766407
ZN738723286	EPIHKW96701706678080
CX045870847	FUYNOV01508018688382
WN885941809	ILDONY33997576846242
GE416087475	INMPBK32762306647122
HF125389488	IOGJHA89455211393957
QM604466450	IOGTJN00170697557967
DU134392527	IOTVZJ74717274265365
TJ298507081	JKFEAZ93385353949966
HF620493196	JNCYGQ55148468854489
XT376703668	JZOIRB49728926802955
PB792692158	KQHZPD70569132491375

For any questions or concerns please contact us HERE



Home

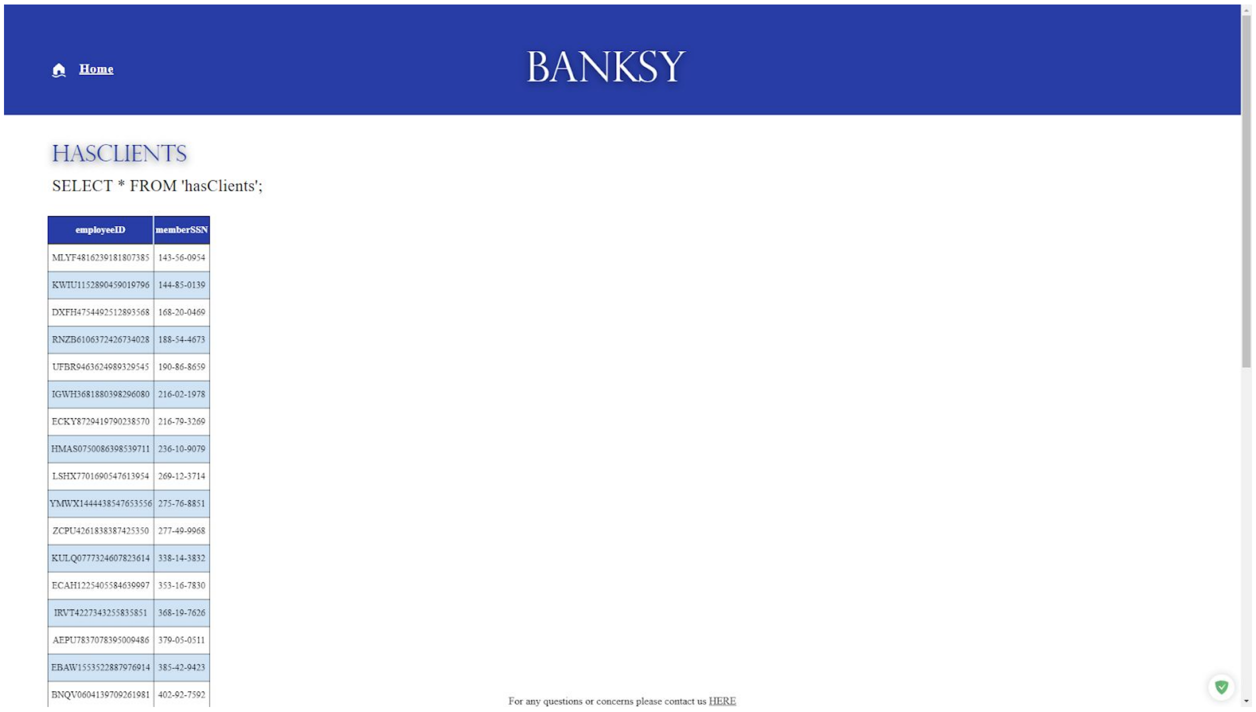
BANKSY

EVALUATES

Query: SELECT * FROM 'Evaluates';

employeeID
AEPU7837078395009486
AEPU8933099949965345
AQNH0736999022387157
BNQV0604139709261981
DGYB7838888060206512
DITG7324595398089037
DXFH4754492512893568
EBAW1553522887976914
ECAH1225405584639997
ECKY8729419790238570
ETMR4727182106104954
FKEG3478511064545006
FXNL5561849382880877
GOKX6418957595013329
GSPN5807039408954042
HDJP1629991661358856
HMAS0750086398539711

For any questions or concerns please contact us HERE



Home

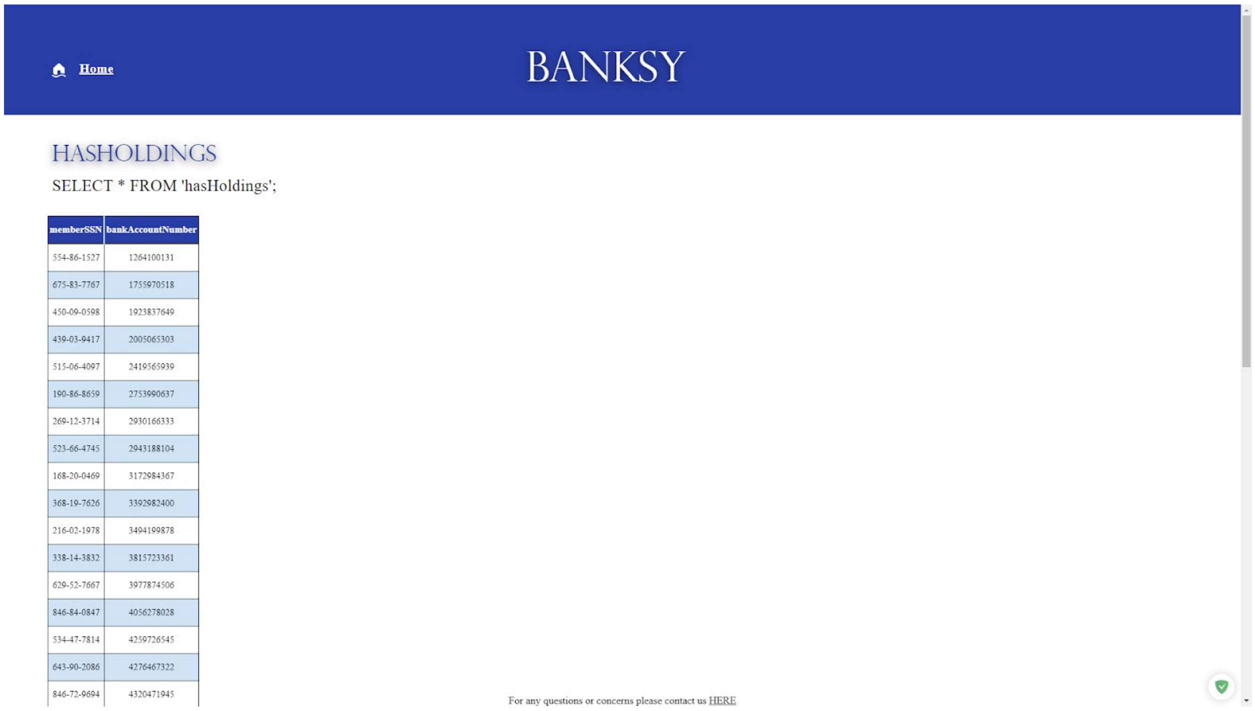
BANKSY

HASCLIENTS

SELECT * FROM 'hasClients';

employeeID	memberSSN
MLYF4816239181807385	143-56-0954
KWIU1152890459019796	144-85-0139
DXFH4754492512893568	168-20-0469
RNZB6106372426734028	188-54-4673
UFBR9463624989329545	190-86-8659
IGWH3681880398296080	216-02-1978
ECKY8729419790238570	216-79-3269
HMAS0750086398539711	236-10-9079
LSHX7701690547613954	269-12-3714
YMWX1444438547653556	275-76-8851
ZCPU4261838387425350	277-49-9968
KULQ0777324607823614	338-14-3832
ECAH1225405584639997	353-16-7830
IRVT4227343255835851	368-19-7626
AEPU7837078395009486	379-05-0511
EBAW1553522887976914	385-42-9423
BNQV0604139709261981	402-92-7592

For any questions or concerns please contact us HERE



Home

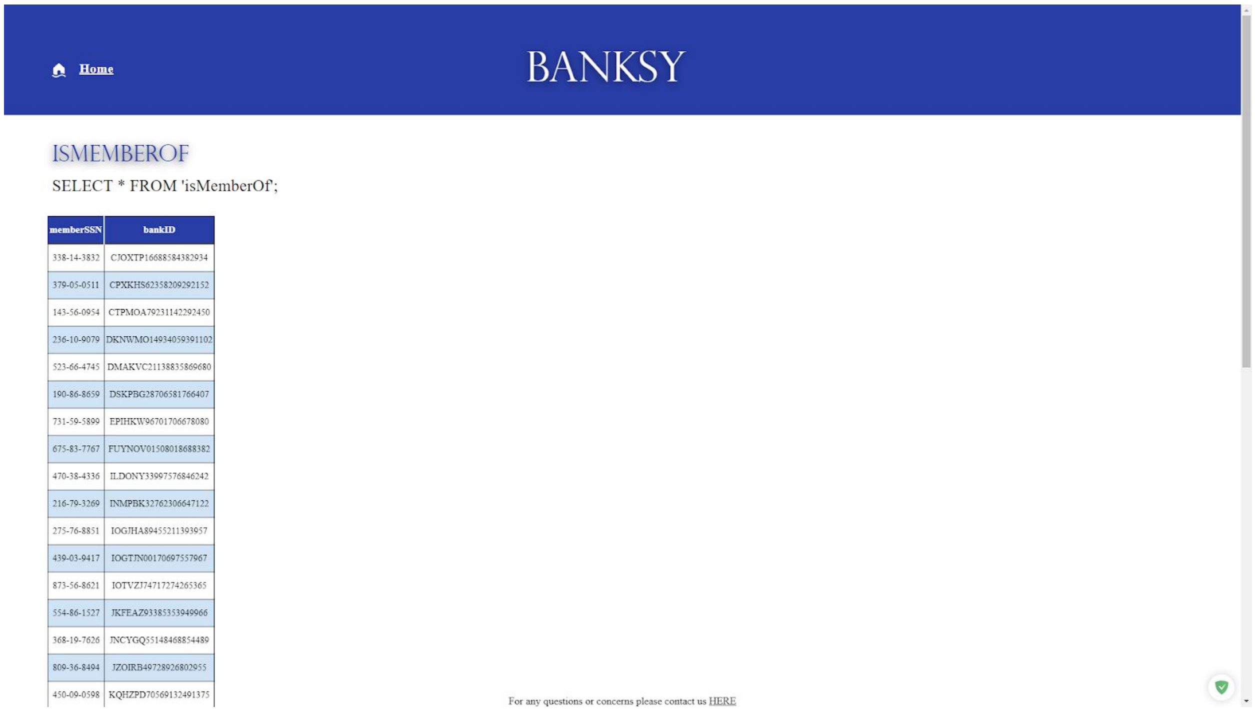
BANKSY

HASHOLDINGS

SELECT * FROM 'hasHoldings';

memberSSN	bankAccountNumber
554-86-1527	1264100131
675-83-7767	1755970518
450-09-0598	1923837649
439-03-9417	2005065303
515-06-4097	2419565939
190-86-8659	2753990637
269-12-3714	2930166333
523-66-4745	2943188104
168-20-0469	3172984367
368-19-7626	3392982400
216-02-1978	3494199878
338-14-3832	3815723361
629-52-7667	3977874506
846-84-0847	4056278028
534-47-7814	4259726545
643-90-2086	4276467322
846-72-9694	4320471945

For any questions or concerns please contact us HERE



Home

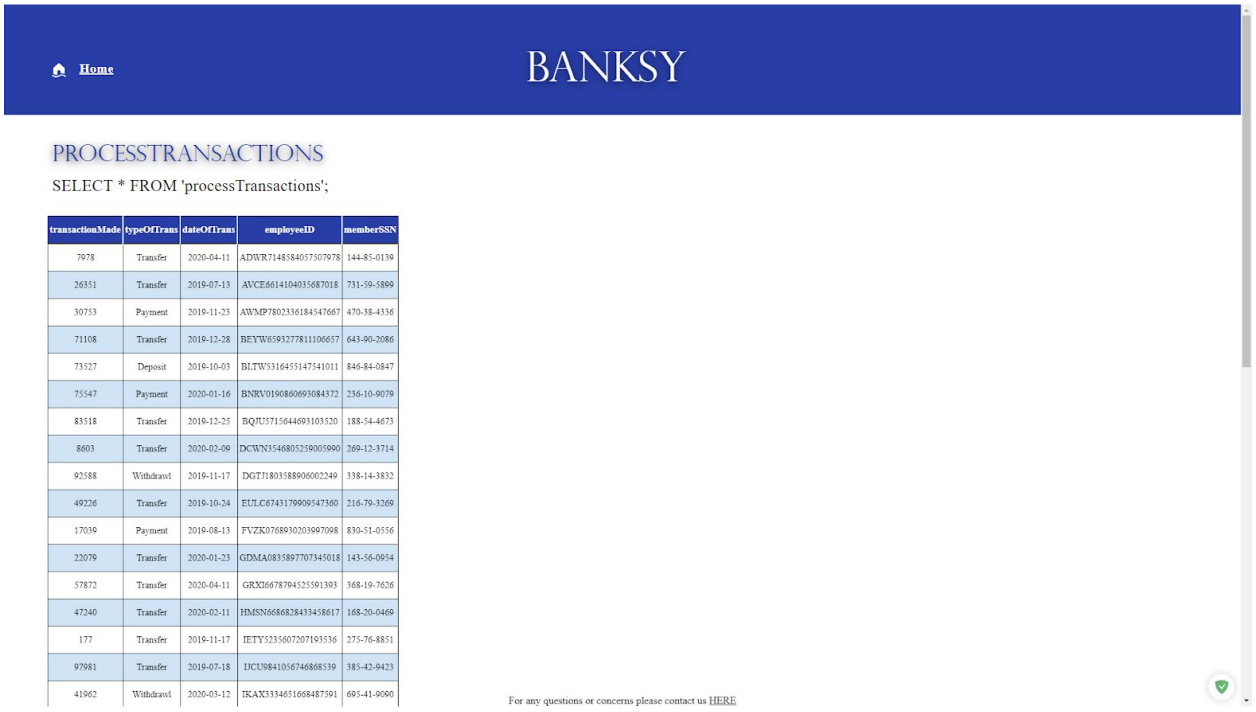
BANKSY

ISMEMBEROF

SELECT * FROM 'isMemberOf';

memberSSN	bankID
338-14-3832	CJOXTP16688584382934
379-05-0511	CPXKHS62358209292152
143-56-0954	CTPMOA79231142292450
236-10-9079	DKNWMO14934059391102
523-66-4745	DMAKVC21138835869680
190-86-8659	DSKPBG28706581766407
731-59-5899	EPIHKW96701706678080
675-83-7767	FUYNOV01508018688382
470-38-4336	ILDONY33997576846242
216-79-3269	INMPBK32762306647122
275-76-8851	IOGJHA89455211393957
439-03-9417	IOGTJN00170697557967
873-56-8621	IOTVZJ74717274265365
554-86-1527	JKFEAZ93385353949966
368-19-7626	JNCYGQ55148468854489
809-36-8494	JZOIRB49728926802955
450-09-0598	KQHZPD70569132491375

For any questions or concerns please contact us HERE


Home

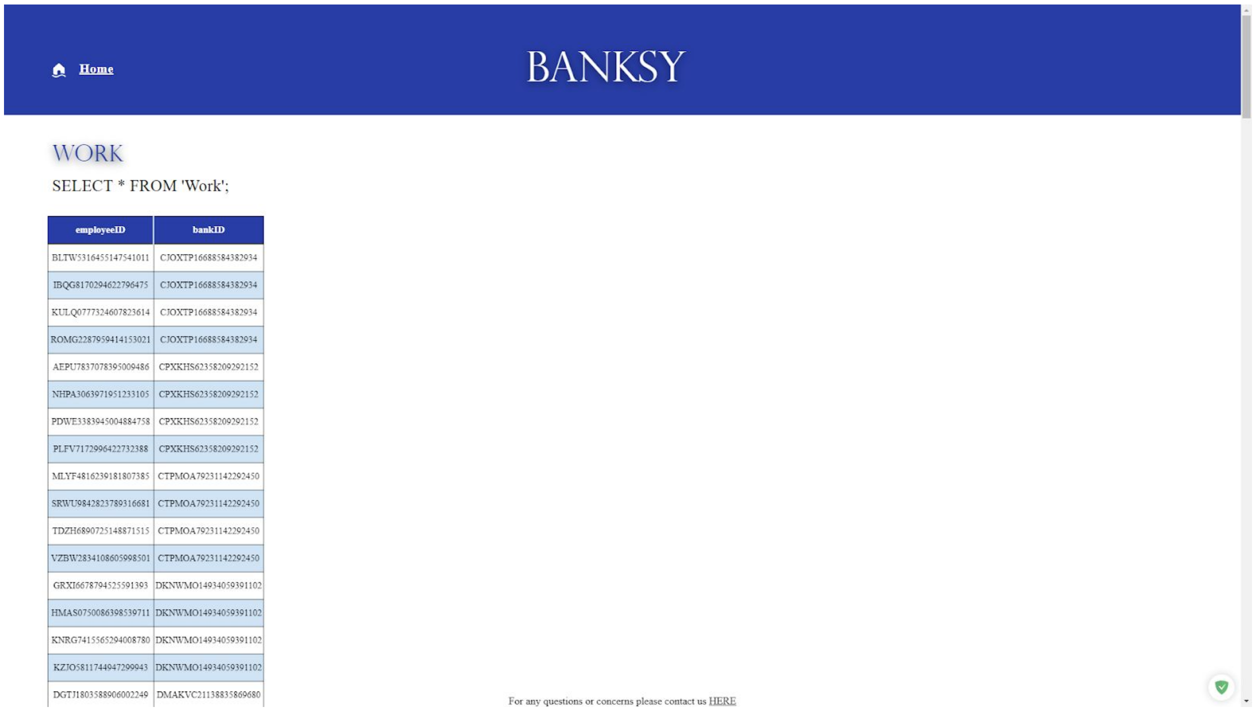
BANKSY

PROCESSTRANSACTIONS

SELECT * FROM 'processTransactions';

transactionMade	typeOfTrans	dateOfTrans	employeeID	memberSSN
7978	Transfer	2020-04-11	ADWR7148584057507978	144-85-0139
26351	Transfer	2019-07-13	AVCE6614104035687018	731-59-5899
30753	Payment	2019-11-23	AWMP7802336184547667	470-38-4336
71108	Transfer	2019-12-28	BEYW6593277811106657	643-90-2086
73527	Deposit	2019-10-03	BLTW5316455147541011	846-84-0847
75547	Payment	2020-01-16	BNRV0190860693084372	236-10-9079
83518	Transfer	2019-12-25	BQJU5715644693103520	188-54-4673
8603	Transfer	2020-02-09	DCWN3546805259005990	269-12-3714
92588	Withdrawl	2019-11-17	DGTJ1803588906002249	338-14-3832
49226	Transfer	2019-10-24	EULC6743179909547360	216-79-3269
17039	Payment	2019-08-13	FVZK0768930203997098	830-51-0556
22079	Transfer	2020-01-23	GDMA0835897707345018	143-56-0954
57872	Transfer	2020-04-11	GRXI6678794525591393	368-19-7626
47240	Transfer	2020-02-11	HMSN6686828433458617	168-20-0469
177	Transfer	2019-11-17	IETY5235607207193536	275-76-8851
97981	Transfer	2019-07-18	IJCU9841056746868539	385-42-9423
41962	Withdrawl	2020-03-12	IKAX3334651668487591	695-41-9090

For any questions or concerns please contact us HERE


Home

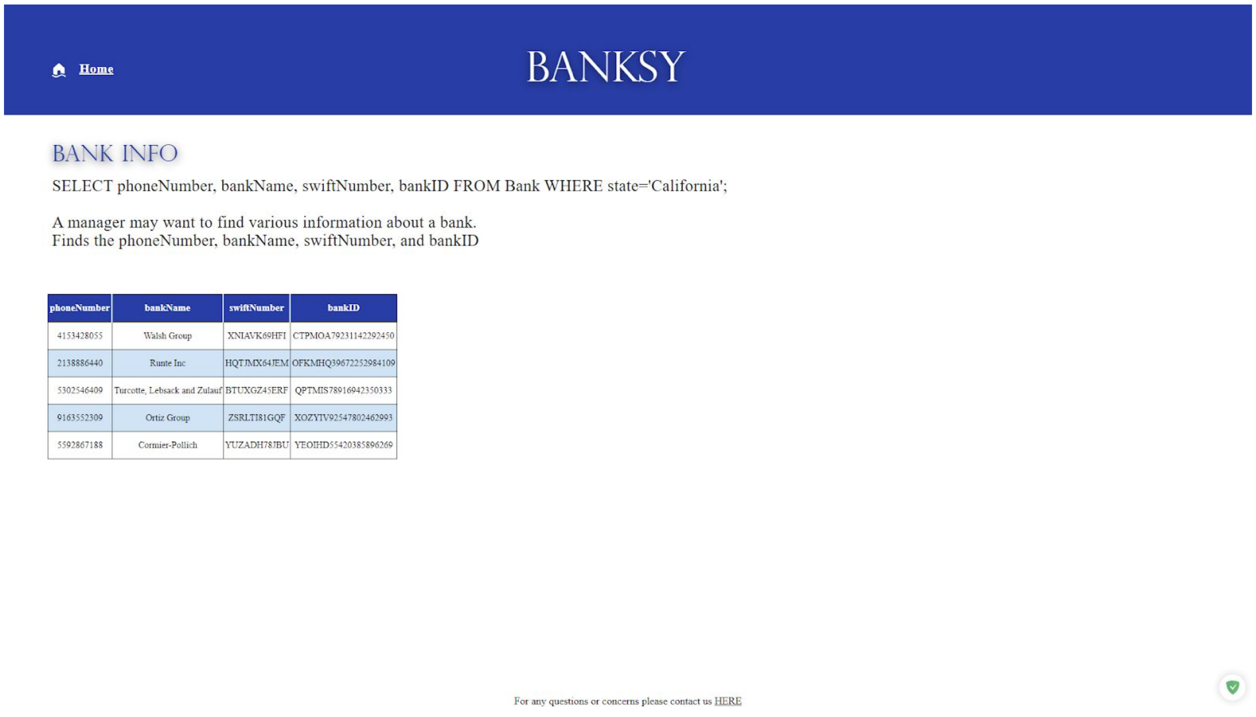
BANKSY

WORK

SELECT * FROM 'Work';

employeeID	bankID
BLTW5316455147541011	CJOXTP16688584382934
IBQG8170294622796475	CJOXTP16688584382934
KULQ0777324607823614	CJOXTP16688584382934
ROMG2287959414153021	CJOXTP16688584382934
AEPU7837078395009486	CPXKHS62358209292152
NHPA3063971951233105	CPXKHS62358209292152
PDWE3383945004884758	CPXKHS62358209292152
PLFV7172996422732388	CPXKHS62358209292152
MLYF4816239181807385	CTPMOA79231142292450
SRWU9842823789316681	CTPMOA79231142292450
TDZH6890725148871515	CTPMOA79231142292450
VZBW2834108605998501	CTPMOA79231142292450
GRXI6678794525591393	DKNWMO14934059391102
HMAS0750086398539711	DKNWMO14934059391102
KNRG7415565294008780	DKNWMO14934059391102
KZJO5811744947299943	DKNWMO14934059391102
DGTJ1803588906002249	DMAKVC21138835869680

For any questions or concerns please contact us HERE


Home

BANKSY

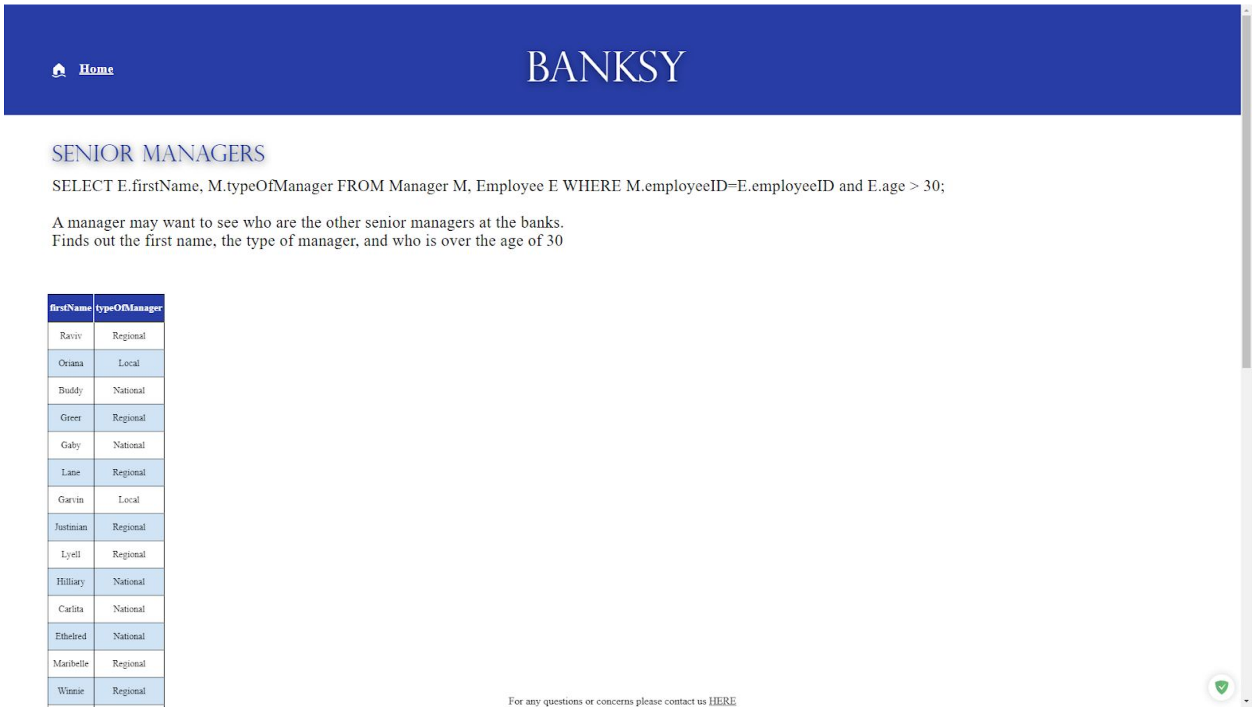
BANK INFO

SELECT phoneNumber, bankName, swiftNumber, bankID FROM Bank WHERE state='California';

A manager may want to find various information about a bank.
Finds the phoneNumber, bankName, swiftNumber, and bankID

phoneNumber	bankName	swiftNumber	bankID
4153428055	Walsh Group	XNIAVK69HFI	CTPMOA79231142292450
2138886440	Runte Inc	HQTJMX64JEM	OFKMHQ39672252984109
5302546409	Turcotte, Lebsack and Zulauf	BTUXGZ45ERF	QPTMIS78916942350333
9163552309	Ortiz Group	ZSRLTI81GQF	XOZYIV92547802462993
5592867188	Cormier-Pollich	YUZADH78JBU	YEOIHD55420385896269

For any questions or concerns please contact us HERE


Home

BANKSY

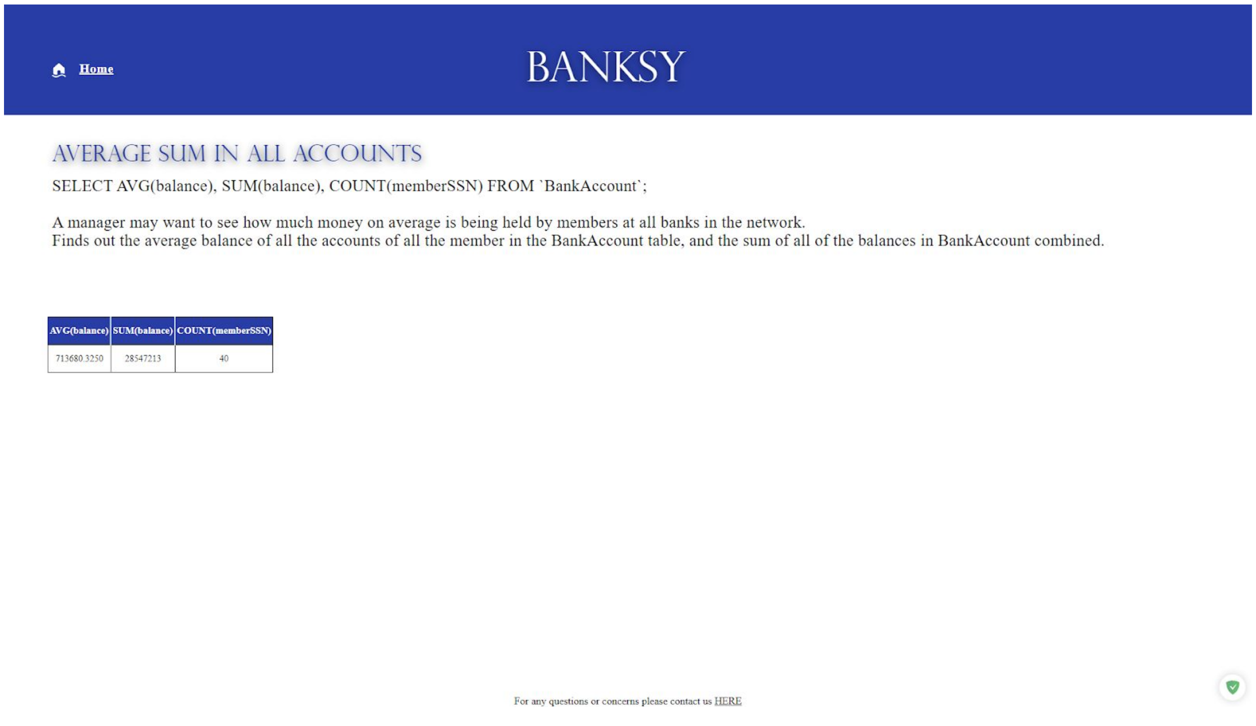
SENIOR MANAGERS

SELECT E.firstName, M.typeOfManager FROM Manager M, Employee E WHERE M.employeeID=E.employeeID and E.age > 30;

A manager may want to see who are the other senior managers at the banks.
Finds out the first name, the type of manager, and who is over the age of 30

firstName	typeOfManager
Raviv	Regional
Oriana	Local
Buddy	National
Greer	Regional
Gaby	National
Lane	Regional
Garvin	Local
Justinian	Regional
Lyell	Regional
Hilliary	National
Carlita	National
Ethelred	National
Maribelle	Regional
Winnie	Regional

For any questions or concerns please contact us HERE



Home

BANKSY

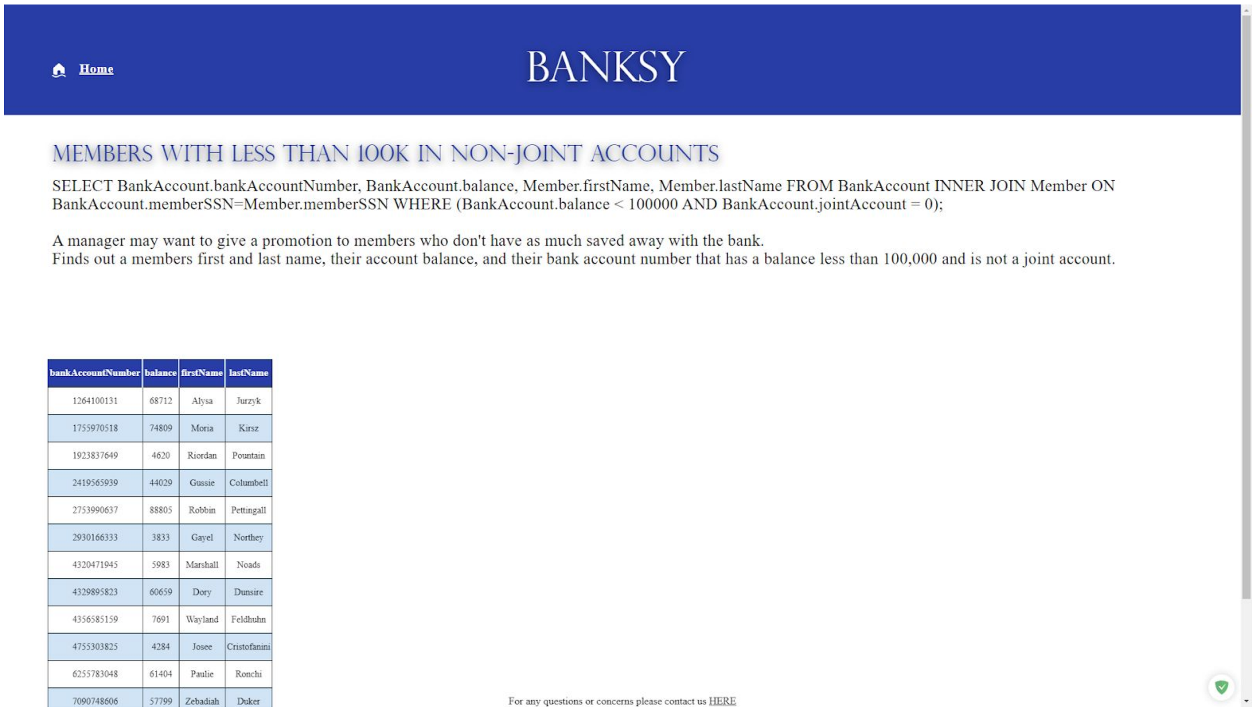
AVERAGE SUM IN ALL ACCOUNTS

SELECT AVG(balance), SUM(balance), COUNT(memberSSN) FROM `BankAccount`;

A manager may want to see how much money on average is being held by members at all banks in the network.
Finds out the average balance of all the accounts of all the member in the BankAccount table, and the sum of all of the balances in BankAccount combined.

AVG(balance)	SUM(balance)	COUNT(memberSSN)
713680.3250	28547213	40

For any questions or concerns please contact us HERE



Home

BANKSY

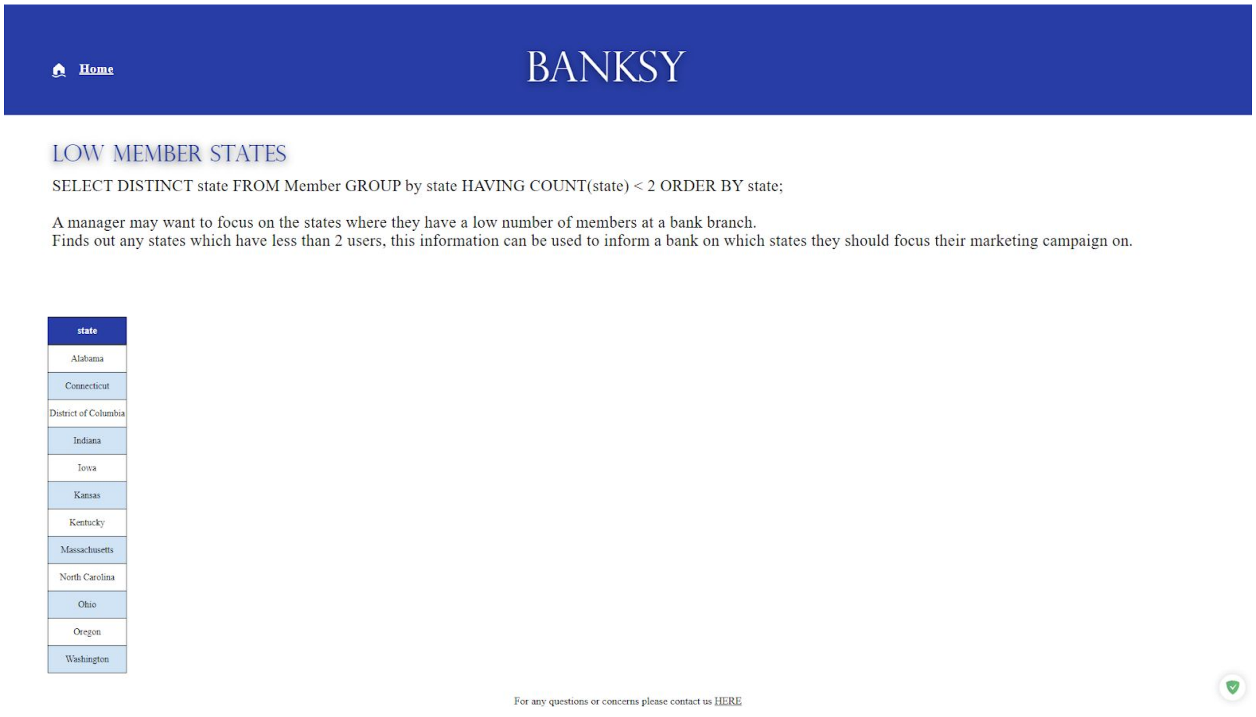
MEMBERS WITH LESS THAN 100K IN NON-JOINT ACCOUNTS

SELECT BankAccount.bankAccountNumber, BankAccount.balance, Member.firstName, Member.lastName FROM BankAccount INNER JOIN Member ON BankAccount.memberSSN=Member.memberSSN WHERE (BankAccount.balance < 100000 AND BankAccount.jointAccount = 0);

A manager may want to give a promotion to members who don't have as much saved away with the bank.
Finds out a members first and last name, their account balance, and their bank account number that has a balance less than 100,000 and is not a joint account.

bankAccountNumber	balance	firstName	lastName
1264100131	68712	Alysa	Jurzyk
1755970518	74809	Moria	Kirsz
1923837649	4620	Riordan	Pountain
2419565939	44029	Gussie	Columbell
2753990637	88805	Robbin	Pettingall
2930166333	3833	Gayel	Northey
4320471945	5983	Marshall	Noads
4329895823	60659	Dory	Dunsire
4356585159	7691	Wayland	Feldhuhn
4755303825	4284	Josee	Cristofanini
6255783048	61404	Paulie	Ronchi
7090748606	57799	Zebadiah	Duker

For any questions or concerns please contact us HERE



Home

BANKSY

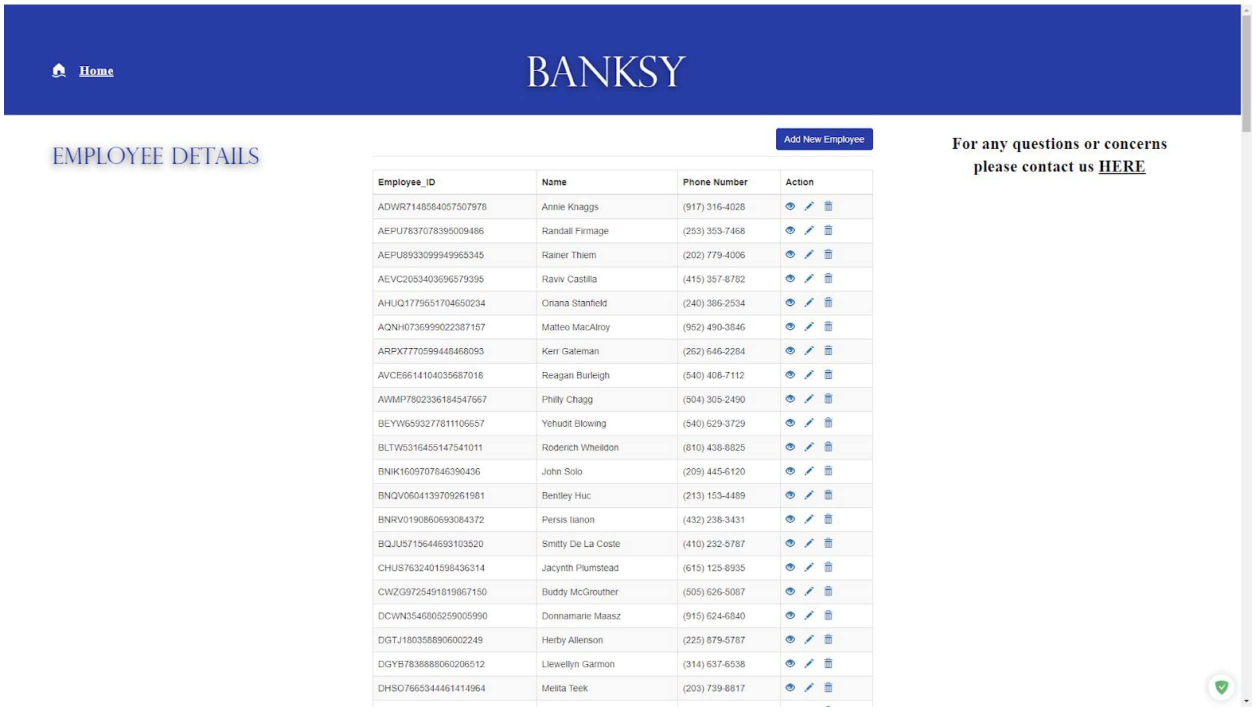
LOW MEMBER STATES

SELECT DISTINCT state FROM Member GROUP by state HAVING COUNT(state) < 2 ORDER BY state;

A manager may want to focus on the states where they have a low number of members at a bank branch.
Finds out any states which have less than 2 users, this information can be used to inform a bank on which states they should focus their marketing campaign on.

state
Alabama
Connecticut
District of Columbia
Indiana
Iowa
Kansas
Kentucky
Massachusetts
North Carolina
Ohio
Oregon
Washington

For any questions or concerns please contact us HERE



Home

BANKSY

EMPLOYEE DETAILS

Employee_ID	Name	Phone Number	Action
ADWR7148584057507978	Annie Knaggs	(917) 316-4028	
AEPU7837078395009486	Randall Firmage	(253) 353-7468	
AEPU8933099949965345	Rainer Thiem	(202) 779-4006	
AEVC2053403696579395	Raviv Castilla	(415) 357-8782	
AHUQ1779551704650234	Oriana Stanfield	(240) 386-2534	
AQNH0736999022387157	Matteo MacAlroy	(952) 490-3846	
ARPX7770599448468093	Kerr Gateman	(262) 646-2284	
AVCE6614104035687018	Reagan Burleigh	(540) 408-7112	
AWMP7802336184547667	Philly Chagg	(504) 305-2490	
BEYW6593277811106657	Yehudit Blowing	(540) 629-3729	
BLTW5316455147541011	Roderich Wheildon	(810) 438-8825	
BNIK1609707846390436	John Solo	(209) 445-6120	
BNQV0604139709261981	Bentley Huc	(213) 153-4489	
BNRV0190860693084372	Persis Iianon	(432) 238-3431	
BQJU5715644693103520	Smitty De La Coste	(410) 232-5787	
CHUS7632401598436314	Jacynth Plumstead	(615) 125-8935	
CWZG9725491819867150	Buddy McGrouther	(505) 626-5087	
DCWN3546805259005990	Donnamarie Maasz	(915) 624-6840	
DGTJ1803588906002249	Herby Allenson	(225) 879-5787	
DGYB7838888060206512	Llewellyn Garmon	(314) 637-6538	
DHSO7665344461414964	Melita Teek	(203) 739-8817	

For any questions or concerns please contact us HERE