CSC-634 Assignment 2 - Bank Database

Author: Yunting Chiu
Email: yc6705a@american.edu

Create the Bank database

```
create database if not exists Bank;
use Bank;

CREATE TABLE IF NOT EXISTS account (
    account_number CHAR(5) NOT NULL PRIMARY KEY,
    branch_name VARCHAR(10),
    balance DOUBLE
);
CREATE TABLE IF NOT EXISTS branch (
    branch_name VARCHAR(10) NOT NULL PRIMARY KEY,
    branch_city VARCHAR(10),
    assets DOUBLE
);
CREATE TABLE IF NOT EXISTS customer (
    customer_name VARCHAR(20) NOT NULL PRIMARY KEY,
    customer_street VARCHAR(20),
    customer_city VARCHAR(10)
);
CREATE TABLE IF NOT EXISTS loan (
    loan_number VARCHAR(5) NOT NULL PRIMARY KEY,
    branch_name VARCHAR(10),
    amount DOUBLE
);
CREATE TABLE IF NOT EXISTS borrower (
    customer_name VARCHAR(20) NOT NULL,
    loan_number VARCHAR(5) NOT NULL,
    PRIMARY KEY (customer_name , loan_number)
);
CREATE TABLE IF NOT EXISTS depositor (
    customer_name VARCHAR(20) NOT NULL,
    account_number CHAR(5) NOT NULL,
    PRIMARY KEY (customer_name , account_number)
);
CREATE TABLE IF NOT EXISTS employee (
```

```
    employee_name VARCHAR(20) NOT NULL,
    branch_name VARCHAR(10) NOT NULL,
    salary DOUBLE,
    PRIMARY KEY (employee_name , branch_name)
);

# populate the tables
## account
insert into account values('A-101', 'Downtown', 500);
insert into account values('A-102', 'Perryridge', 400);
insert into account values('A-201', 'Brighton', 900);
insert into account values('A-215', 'Mianus', 700);
insert into account values('A-217', 'Brighton', 750);
insert into account values('A-222', 'Redwood', 700);
insert into account values('A-305', 'Round Hill', 350);

## branch
insert into branch values('Brighton', 'Brooklyn', 7100000);
insert into branch values('Downtown', 'Brooklyn', 9000000);
insert into branch values('Mianus', 'Horseneck', 400000);
insert into branch values('North Town', 'Rye', 3700000);
insert into branch values('Perryridge', 'Horseneck', 1700000);
insert into branch values('Pownal', 'Bennington', 300000);
insert into branch values('Redwood', 'Palo Alto', 2100000);
insert into branch values('Round Hill', 'Horseneck', 8000000);

## customer
insert into customer values('Adams', 'Spring', 'Pittsfield');
insert into customer values('Brooks', 'Senator', 'Brooklyn');
insert into customer values('Curry', 'North', 'Rye');
insert into customer values('Glenn', 'Sand Hill', 'Woodside');
insert into customer values('Green', 'Walnut', 'Stamford');
insert into customer values('Hayes', 'Main', 'Harrison');
insert into customer values('Johnson', 'Alma', 'Palo Alto');
insert into customer values('Jones', 'Main', 'Harrison');
insert into customer values('Lindsay', 'Park', 'Pittsfield');
insert into customer values('Smith', 'North', 'Rye');
insert into customer values('Turner', 'Putnam', 'Stamford');
insert into customer values('Williams', 'Nassau', 'Princeton');
```

```
## loan
insert into loan values('L-11', 'Round Hill', 900);
insert into loan values('L-14', 'Downtown', 1500);
insert into loan values('L-15', 'Perryridge', 1500);
insert into loan values('L-16', 'Perryridge', 1300);
insert into loan values('L-17', 'Downtown', 1000);
insert into loan values('L-23', 'Redwood', 2000);
insert into loan values('l-93', 'Mianus', 500);

## borrower
insert into borrower values('Adams', 'L-16');
insert into borrower values('Curry', 'L-93');
insert into borrower values('Hayes', 'L-15');
insert into borrower values('Jackson', 'L-14');
insert into borrower values('Jones', 'L-17');
insert into borrower values('Smith', 'L-11');
insert into borrower values('Smith', 'L-23');
insert into borrower values('Williams', 'L-17');

## depositor
insert into depositor values('Hayes', 'A-102');
insert into depositor values('Johnson', 'A-102');
insert into depositor values('Johnson', 'A-201');
insert into depositor values('Jones', 'A-217');
insert into depositor values('Lindsay', 'A-222');
insert into depositor values('Smith', 'A-215');
insert into depositor values('Turner', 'A-305');

## employee
insert into employee values('Adams', 'Perryridge', 1500);
insert into employee values('Brown', 'Perryridge', 1300);
insert into employee values('Gopal', 'Perryridge', 5300);
insert into employee values('Johnson', 'Downtown', 1500);
insert into employee values('Loreena', 'Downtown', 1300);
insert into employee values('Peterson', 'Downtown', 2500);
insert into employee values('Rao', 'Austin', 1500);
insert into employee values('Sato', 'Austin', 1600);
```

Retrieval Queries

1. Find all loan number for loans made at the Perryridge branch with loan amounts greater than $1100.

```
select loan_number from loan
where branch_name = "Perryridge" and amount > 1100;
```

```
# loan_number
L-15
L-16
```

2. Find the loan number of those loans with loan amounts between $1,000 and $1,500 (that is, >=$1,000 and <=$1,500)

```
select loan_number from loan
where amount between 1000 and 1500;
```

```
# loan_number
L-14
L-15
L-16
L-17
```

3. Find the names of all branches that have greater assets than some branch located in Brooklyn.

```
select distinct tb1.branch_name from branch tb1 join branch tb2
where tb1.assets > tb2.assets and tb2.branch_city = "Brooklyn";
```

second solution

```
SELECT branch_name FROM branch
WHERE assets > ANY(
SELECT assets
FROM branch
WHERE branch_city = 'Brooklyn'
);
```

```
# branch_name
Downtown
Round Hill
```

4. Find the customer names and their loan numbers for all customers having a loan at some branch.

```
select distinct B.customer_name, B.loan_number from borrower as B inner join loan as
L
on B.loan_number = L.loan_number;
```

```
# customer_name    loan_number
Adams          L-16
Curry          L-93
Hayes          L-15
Jackson        L-14
Jones          L-17
Smith          L-11
Smith          L-23
Williams       L-17
```

5. Find all customers who have a loan, an account, or both:

```
(select customer_name from depositor)
union
(select customer_name from borrower);
```

```
# customer_name
Hayes
Johnson
Jones
Lindsay
Smith
Turner
Adams
Curry
Jackson
Williams
```

6. Find all customers who have an account but no loan. (no minus operator provided in mysql)

```
select distinct customer_name from depositor
where customer_name not in (select customer_name from borrower);
```

```
# customer_name
Johnson
```

Lindsay
Turner

7. Find the number of depositors for each branch.

```
select branch_name, count(distinct customer_name)
from depositor, account
where depositor.account_number = account.account_number
group by branch_name;
```

```
# branch_name       count(distinct customer_name)
Brighton        2
Mianus          1
Perryridge      2
Redwood         1
Round Hill      1
```

8. Find the names of all branches where the average account balance is more than $500.

```
select branch_name, avg(balance) from account
group by branch_name
having avg(balance) > 500;
```

```
# branch_name       avg(balance)
Brighton        825
Mianus          700
Redwood         700
```

9. Find all customers who have both an account and a loan at the bank.

```
select distinct customer_name from borrower
where customer_name in (select customer_name from depositor);
```

```
# customer_name
Hayes
Jones
Smith
```

10. Find all customers who have a loan at the bank but do not have an account at the bank

```
select distinct customer_name from borrower
where customer_name not in (select customer_name from depositor);
```

customer_name
Adams
Curry
Jackson
Williams

11. Find the names of all branches that have greater assets than all branches located in Horseneck. (using both non-nested and nested select statement)

- non-nested

```
# create a temporary table
create table maxHorseNeckAssets
select max(assets) as assets from branch where branch_city = 'HorseNeck';
# compare branch to maxHorseNeckAssets
select distinct branch_name from branch , maxHorseNeckAssets
where branch.assets > maxHorseNeckAssets.assets;
```

```
# branch_name
Downtown
```

- nested

```
select branch_name from branch
where assets > all (
select assets from branch where branch_city in (
select branch_city from branch where branch_city = "Horseneck"
)
);
```

```
# branch_name
Downtown
```

12. 1 query of your choice involving aggregate functions

```
# Find the brance_name in the loan table whose total amount is greater than 1000.
select branch_name , sum(amount) from loan
group by branch_name
having sum(amount) > 1000;
```

```
# branch_name       sum(amount)
Downtown     2500
Perryridge     2800
```

Redwood 2000

13. 1 query of your choice involving group by feature.
Add up the count of each branch_name in account table
```
select branch_name, count(*) from account
group by branch_name;
```

```
# branch_name sum(amount)
Downtown 2500
Perryridge 2800
Redwood 2000
```

Insert Queries

Do 2 insert queries requiring multiple records insertion as follow:
1. Create a HighLoan table with loan amount >=1500.
```
create table HighLoan select * from loan
where amount >= 1500;
select * from HighLoan;
```

```
# loan_number branch_name amount
L-14 Downtown 1500
L-15 Perryridge 1500
L-23 Redwood 2000
```

2. Create a HighSalaryEmployee table with employee having salary more than 2000.
```
create table HighSalaryEmployee select * from employee
where salary > 2000;
select * from HighSalaryEmployee;
```

```
# employee_name branch_name salary
Gopal Perryridge 5300
Peterson Downtown 2500
```

3. 1 more query (meaningful) of your choice on any table.
Find all branch_name who have a HighLoan, a HighSalaryEmployee, or both, then save this table as HighBranch.
```
create table HighBranch
select branch_name from HighLoan
union
select branch_name from HighSalaryEmployee;
```

```
select * from HighBranch;

# branch_name
Downtown
Perryridge
Redwood
```

Update Queries

1. Increase all accounts with balances over $800 by 7%, all other accounts receive 8%.

- before the query

```
select * from account;
```

# account_number	branch_name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

- after the query

```
update account
set balance = balance * 1.07 where balance > 800;
update account
set balance = balance * 1.08 where balance <= 800;
select * from account;
```

# account_number	branch_name	balance
A-101	Downtown	540
A-102	Perryridge	432
A-201	Brighton	963
A-215	Mianus	756
A-217	Brighton	810
A-222	Redwood	756
A-305	Round Hill	378

2. Do 2 update queries, each involving 2 tables.

First: If an employee is located in a high-loan area, their salary should be increased by 20%.

- before the query

```
select * from employee;
```

```
# employee_name   branch_name         salary
Adams         Perryridge     1500
Brown Perryridge     1300
Gopal Perryridge     5300
Johnson       Downtown       1500
Loreena       Downtown       1300
Peterson      Downtown       2500
Rao     Austin 1500
Sato    Austin 1600
```

- after the query

```
update employee
set salary = salary * 1.2 where branch_name in (select branch_name from HighLoan);
select * from employee;
```

```
# employee_name   branch_name         salary
Adams         Perryridge     1800
Brown Perryridge     1560
Gopal Perryridge     6360
Johnson       Downtown       1800
Loreena       Downtown       1560
Peterson      Downtown       3000
Rao     Austin 1500
Sato    Austin 1600
```

Second: If employees are not located in a high-loan area, their salary should be decreased by 10%.

- before the query

```
select * from employee;
```

```
# employee_name   branch_name         salary
Adams         Perryridge     1800
Brown Perryridge     1560
Gopal Perryridge     6360
Johnson       Downtown       1800
```

Loreena Downtown 1560
Peterson Downtown 3000
Rao Austin 1500
Sato Austin 1600

after the query
update employee
set salary = salary * 0.9 where branch_name not in (select branch_name from HighLoan);
select * from employee;

employee_name branch_name salary
Adams Perryridge 1800
Brown Perryridge 1560
Gopal Perryridge 6360
Johnson Downtown 1800
Loreena Downtown 1560
Peterson Downtown 3000
Rao Austin 1350
Sato Austin 1440

3. 1 more update query of your choice on any table.
If the loan amount is greater than $1,000, the loan amount will be increased by 5%.

- before the query

select * from loan;

loan_number branch_name amount
L-11 Round Hill 900
L-14 Downtown 1500
L-15 Perryridge 1500
L-16 Perryridge 1300
L-17 Downtown 1000
L-23 Redwood 2000
l-93 Mianus 500

- after the query

update loan
set amount = amount + amount * 0.05 where amount > 1000;
select * from loan;

# loan_number	branch_name	amount
L-11	Round Hill	900
L-14	Downtown	1575
L-15	Perryridge	1575
L-16	Perryridge	1365
L-17	Downtown	1000
L-23	Redwood	2100
l-93	Mianus	500

Delete Queries

1. Delete the record of all accounts with balances below the average at the bank.

- before the query

```
select * from account;
```

# account_number	branch_name	balance
A-101	Downtown	540
A-102	Perryridge	432
A-201	Brighton	963
A-215	Mianus	756
A-217	Brighton	810
A-222	Redwood	756
A-305	Round Hill	378

- after the query

```
delete from account
where balance < (select * from(select avg(balance) from account)
as tmp);
select * from account;
```

# account_number	branch_name	balance
A-201	Brighton	963
A-215	Mianus	756
A-217	Brighton	810
A-222	Redwood	756

2. Do 2 delete queries, each involving 2 tables.

first: delete the employee name which branch_name is in loan table.

- before the query

```
select * from employee;
```

employee_name branch_name salary
Adams Perryridge 1800
Brown Perryridge 1560
Gopal Perryridge 6360
Johnson Downtown 1800
Loreena Downtown 1560
Peterson Downtown 3000
Rao Austin 1350
Sato Austin 1440

- after the query

```
delete from employee
where branch_name in (select distinct branch_name from loan);
select * from employee;
```

employee_name branch_name salary
Rao Austin 1350
Sato Austin 1440

second: Remove the customer name from the depositor table that is NOT in the borrower table.

- before the query

```
select * from depositor;
```

customer_name account_number
Hayes A-102
Johnson A-102
Johnson A-201
Jones A-217
Lindsay A-222
Smith A-215
Turner A-305

- after the query

```
delete from depositor
where customer_name not in (select distinct customer_name from borrower);
select * from depositor;
```

customer_name account_number
Hayes A-102

Jones A-217
Smith A-215

3. 1 more delete query of your choice from any table.
If an employee's salary is less than $1,400, their name should be removed.

- before the query

```
select * from employee;
```

```
# employee_name   branch_name         salary
Rao    Austin 1350
Sato    Austin 1440
```

- after the query

```
delete from employee
where salary < 1400;
select * from employee;
```

```
# employee_name   branch_name         salary
Sato    Austin 1440
```

Views Queries

1. A view consisting of branches and their customers.

```
create view customerView as
(select branch_name, customer_name
from depositor D, account A
where D.account_number = A.account_number)
union
(select branch_name, customer_name
from borrower B, loan L
where B.loan_number = L.loan_number);
select * from customerView;
```

```
# branch_name       customer_name
Perryridge    Hayes
Brighton      Jones
Mianus        Smith
Perryridge    Adams
Mianus        Curry
Downtown      Jackson
```

Downtown Jones
Round Hill Smith
Redwood Smith
Downtown Williams

2. Create a view of HQEmployee who work in the downtown branch.
check the employee entity first
select * from employee;

employee_name branch_name salary
Sato Austin 1440

create a view as HQEmployee
create view HQEmployee as
select employee_name, branch_name from employee
where branch_name = "Downtown";
select * from HQEmployee;

There are no values in the HQEmployee view because some observations from previous queries were deleted

employee_name, branch_name

3. Do one insert, delete, update, and select queries on HQEmployee view.

- **insert**

insert into HQEmployee (employee_name, branch_name) values ("Yunting", "Downtown");
insert into HQEmployee values ("Guzman", "Downtown");
select * from HQEmployee;

employee_name branch_name
Guzman Downtown
Yunting Downtown

- **delete**

delete from HQEmployee where employee_name = "Guzman";
select * from HQEmployee;

employee_name branch_name
Yunting Downtown

- **update**

```
update HQEmployee
set employee_name = "Yunting Chiu"
where branch_name = "Downtown";
select * from HQEmployee;
```

```
# employee_name   branch_name
Yunting Chiu Downtown
```

- **select**

```
select employee_name from HQEmployee
where branch_name is not null;
```

```
# employee_name
Yunting Chiu
```

Complex Queries: provide results

```
## 1. 1 select query involving 3 tables
# display the names of customers with assets greater than $ 1,000,000
select customer_name from depositor D inner join account A inner join branch B
on D.account_number = A.account_number and A.branch_name = B.branch_name
group by customer_name
having sum(assets) > 1000000;
```

```
# customer_name
Hayes
Jones
```

```
## 2. 1 Delete query involving 3 table
# delete the names of customers with balance greater than $ 800
```

- before the query

```
select customer.customer_name, balance from customer
inner join depositor on customer.customer_name = depositor.customer_name
inner join account on depositor.account_number = account.account_number;
```

```
# customer_name    balance
Hayes 400
Smith  756
Jones 810
```

- after the query

```
delete customer from customer
inner join depositor on customer.customer_name = depositor.customer_name
inner join account on depositor.account_number = account.account_number
where account.balance > 800;
# see the result
select customer.customer_name, balance from customer
inner join depositor on customer.customer_name = depositor.customer_name
inner join account on depositor.account_number = account.account_number;

# customer_name    balance
Hayes 400
Smith  756

## 3. 1 Update query involving 3 tables
# crate view to make the code readable
create view threeTables as
select customer.customer_name, balance from customer
inner join depositor on customer.customer_name = depositor.customer_name
inner join account on depositor.account_number = account.account_number;
select * from threeTables;


# customer_name    balance
Hayes 400
Smith  756

# increase each customer's balance by 10%
update threeTables
set balance = round(balance * 1.1, 2)
where customer_name is not null;
select * from threeTables;

# customer_name    balance
Hayes 440
Smith  831.6
```