

Department of Computer Science and Engineering

Course Code: CSE 370	Credits: 3.0
Course Name: Database Systems	Semester: Summer 24

Lab 03 : Introduction to Bank DB, SQL Joins and Constraints

Activity List

Suggestions for this Lab:

- Use a **Text editor** such as Notepad to type and save your program.
- **Copy** and **Paste** your program from the Text editor to the command line. If the program works, save the program. Otherwise, fix the error and save it.
- Save your text file regularly.

Task Create the “Bank” database and then create all necessary tables below

```
CREATE DATABASE Bank;
```

```
USE Bank;
```

```
create table customer (  
customer_id varchar(10) not null,  
customer_name varchar(20) not null,  
customer_street varchar(30),  
customer_city varchar(30),  
primary key (customer_id));
```

```
create table branch (  
branch_name varchar(15),  
branch_city varchar(30),  
assets int,  
primary key (branch_name),  
check (assets >= 0));
```

```
create table account (  
branch_name varchar(15),  
account_number varchar(10) not null,  
balance int,  
primary key (account_number),  
check (balance >= 0));
```

```
create table loan (  
loan_number varchar(10) not null,
```

```
branch_name varchar(15),  
amount int,  
primary key (loan_number));
```

```
create table depositor (  
customer_id varchar(10) not null,  
account_number varchar(10) not null,  
primary key (customer_id,account_number),  
foreign key (customer_id) references customer(customer_id),  
foreign key (account_number) references account(account_number));
```

```
create table borrower (  
customer_id varchar(10) not null,  
loan_number varchar(10) not null,  
primary key (customer_id, loan_number),  
foreign key (customer_id) references customer(customer_id),  
foreign key (loan_number) references loan(loan_number));
```

Task 2

Once all your tables have been created, you should insert the data below. The insertion code has been provided for you. After insertion, check that data has been correctly inserted in all tables using the “Select” query.

```
('C-201','Smith', 'North', 'Rye'),  
( 'C-211','Hayes', 'Main', 'Harrison'),  
( 'C-212','Curry', 'North', 'Rye'),  
( 'C-215','Lindsay', 'Park', 'Pittsfield'),  
( 'C-220','Turner', 'Putnam', 'Stamford'),  
( 'C-222','Williams', 'Nassau', 'Princeton'),  
( 'C-225','Adams', 'Spring', 'Pittsfield'),  
( 'C-226','Johnson', 'Alma', 'Palo Alto'),  
( 'C-233','Glenn', 'Sand Hill', 'Woodside'),  
( 'C-234','Brooks', 'Senator', 'Brooklyn'),  
( 'C-255','Green', 'Walnut', 'Stamford');
```

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

```
insert into account values  
( 'Downtown', 'A-101',500),
```

```
('Mianus','A-215',700) ,
('Perryridge','A-102',400),
('Round Hill','A-305',350),
('Brighton','A-201',900),
('Redwood','A-222',700),
('Brighton','A-217',750);
```

insert into loan values

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

insert into depositor values

```
('C-226', 'A-101'),
('C-201', 'A-215'),
('C-211', 'A-102'),
('C-220', 'A-305'),
('C-226', 'A-201'),
('C-101', 'A-217'),
('C-215', 'A-222');
```

insert into borrower values

```
('C-101', 'L-17'),
('C-201', 'L-23'),
('C-211', 'L-15'),
('C-226', 'L-14'),
('C-212', 'L-93'),
('C-201', 'L-11'),
('C-222', 'L-17'),
('C-225', 'L-16');
```

Task 3

The command below is a general format for joining two tables. You can replace “Inner Join” with any of the three other Join operations.

Select * from Table1 inner join Table2 on Table1.attribute=Table2.attribute;

1. Retrieve all customer’s id, name, city and account number using
 - a. Inner Join
 - b. Left Join
 - c. Right Join

d. Full Join [Not supported by MySQL]

10

Task 4

You can join more than two tables using the following format:

```
Select * from ((Table1 inner join Table2 on Table1.attribute=Table2.attribute)
inner join Table 3 on Table3.attribute = Table1/2.attribute);
```

Retrieve the following information from your database using “join”: Customer name, city, account number, balance and branch name.

Task 5

Inner join can also be accomplished without using the “join” keyword in the following way:

```
Select * from T1, T2, T3,.....Tn where T1.attr=T2.attr [ .....other conditions] ;
```

Apply the above format on Task 4 and compare your results.

Task 6

Solve all tasks below. Some tasks require multiple tables for which you may use joins as shown in “Task 3 and 4” or without using the join keyword as shown in “Task 5”. After joining your required tables, according to your need you may use any other clauses(or keywords) learnt in previous labs, such as, where, group by, having, order by. Some tasks may not need multiple tables at all.

1. Find names and cities of customers who have a loan at Perryridge branch
2. Find the accounts with balances between 700 and 900.
3. Find the names of customers on streets with names ending in "Hill".
4. Find the names of branches whose assets are greater than the assets of some branch in Brooklyn.
5. Find the set of names of branches whose assets are greater than the assets of all branches in Horseneck.
6. Find the set of names of customers at Brighton branch, in alphabetical order.
7. Show the loan data, ordered by decreasing amounts, then increasing loan numbers.
8. Find the names of branches having at least one account, with average balances greater than or equal 700.
9. Find the names and account number of customers who have the 3 highest balances in their accounts. [Hint: https://www.w3schools.com/sql/sql_top.asp]

Task 7

Solve the following tasks:

1. Find the names of customers with accounts at a branch where Johnson has an account.
2. Find the names of customers with an account but not a loan at Mianus branch.
3. Find the names of each branch and the number of customers having at least one account at that branch.
4. Find the average balance of all customers in 'Palo Alto' having at least 2 accounts
5. Find the name and account number of the customer who has the 3rd highest balance in their account.