Department of Computer Science and Engineering

Course Code:CSE3121	
Course Name: Database Systems	

Lab 04
Introduction to Bank DB and SQL Joins

I. **Topic Overview:**
The students will solve problems to familiarize themselves with a complete database system and SQL Join queries. They will learn the concept of storing data in multiple tables, primary- foreign key constraints and the 4 different SQL Join operations.

II. **Lesson Fit:**
Students should have an understanding of the following:
1. Creating tables and inserting data
2. Foreign keys and primary keys
3. Relationships

III. **Learning Outcome:**
After this lecture, the students will be able to:
a. Understand the design of a basic database System (Bank database)
b. Understand and apply primary-foreign key constraints(referential integrity constraint) on tables
c. Apply SQL Join operations and retrieve aggregated information from multiple tables

IV. **Acceptance and Evaluation**
Students will show their progress as they complete each problem. They are expected to complete all tasks in class. They will be marked according to their class performance. Only in special circumstances, students will be allowed to complete the rest of the task at home.

V. **Activity Detail**
a. **Hour: 1**
Discussion:
Explain referential integrity constraints and the 4 types of Joins with examples.
Problem Task:
i. Task 1 and 2

b. **Hour: 2**

Discussion:
Check students have created all tables and inserted all data correctly according to Tasks 1, 2. Ensure students understand the constraints in the tables (primary key, foreign key, not null, check etc.)

Problem Task:
i. Task 3 to 6

c. **Hour: 3**
Discussion:
Check students can apply the Join queries to complete tasks 3 -7

Problem Task:
i. Task 7

VI. **Home tasks**
a. Unfinished tasks [only with permission in special circumstances]

Lab 4 Activity List

Suggestions for this Lab:

- Use a **Text editor** such as Note Pad 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 1

Create the "Bank" database and then create all necessary tables below

```
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.

```
insert into customer values
('C-101','Jones', 'Main', 'Harrison'), ('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);
```

Task-3 1a) SELECT customer.customer_id,customer.customer_name , customer.customer_city, depositor.account_number FROM customer INNER JOIN depositor
on customer.customer_id = depositor.customer_id;

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]
2. *Explain the differences you observed between the four joins in Task 3(1). If there was no difference in the result between two or more join operations then explain why.*

INNER JOIN and RIGHT JOIN yields the same result. LEFT JOIN gave different result.

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);

SELECT customer.customer_name, customer.customer_city, account.account_number, account.balance, account.branch_name
FROM ((customer INNER JOIN depositor on customer.customer_id = depositor.customer_id)
INNER JOIN account on account.account_number = depositor.account_number);

```
SELECT customer.customer_name, customer.customer_city,ACCOUNT.account_number,
   ACCOUNT.balance,ACCOUNT.branch_name
FROM customer,depositor,ACCOUNT
WHERE   (customer.customer_id = depositor.customer_id) AND
   (ACCOUNT.account_number = depositor.account_number);
```

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.   same result

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 which 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