



Islamic University of Technology (IUT)

## CSE 4308: Database Management Systems Lab

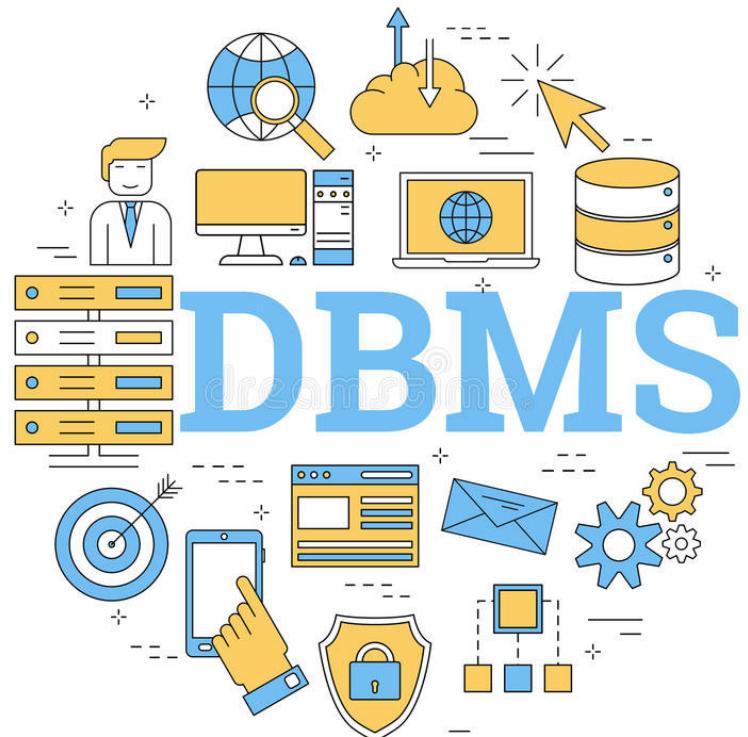
### Lab Report # 4

#### Submitted to:

Md. Bakhtiar Hasan,  
Asst. Professor, CSE.  
Zannatun Naim Srsity,  
Lecturer, CSE.

#### Submitted by:

M M Nazmul Hossain  
ID 200042118  
CSE (SWE)



#### Submission Date:

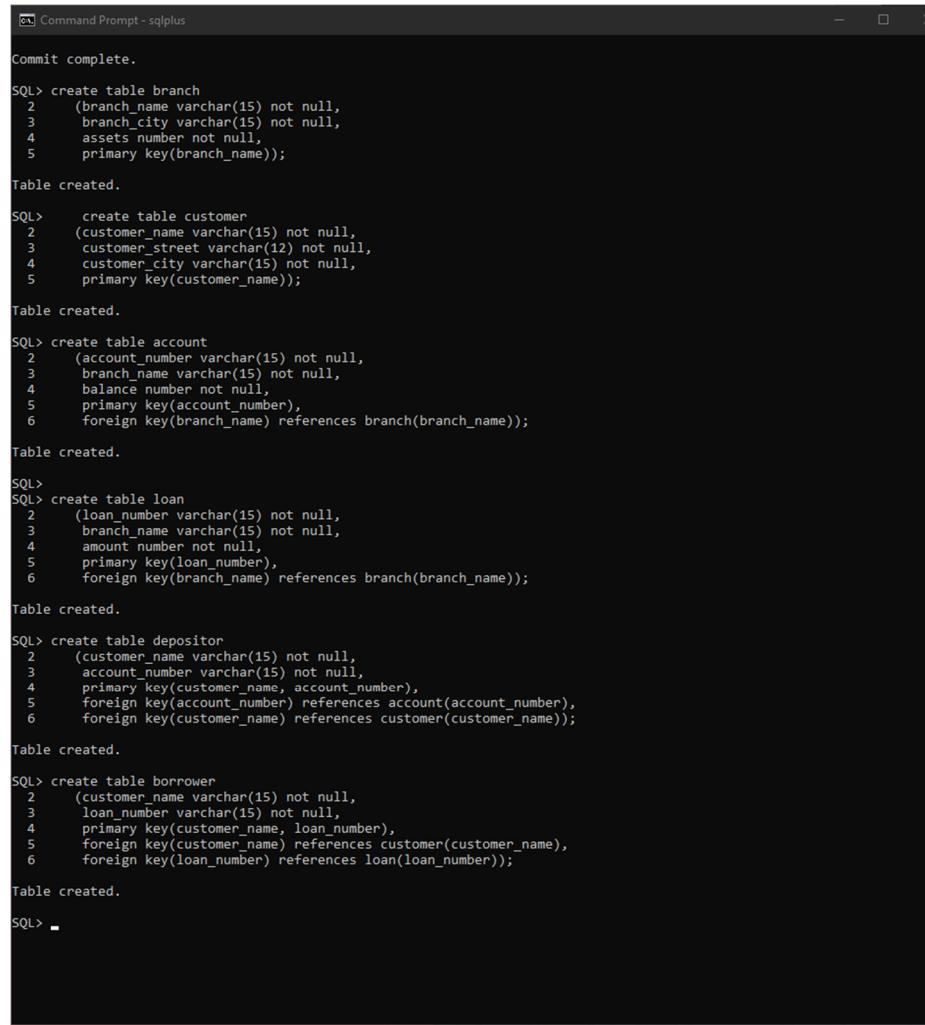
14.09.2022

# Introduction

In our third database management systems lab, the goal of this lab was to realize the importance of set operations and the different types of views that allow us to use different function statements.

## Method

First, I fixed all the errors present in the banking.sql file, where there were only tab spaces given which the cmd line didn't register. Then I created all the tables after deleting the similar tables created in the previous lab.



```
Command Prompt - sqlplus
Commit complete.

SQL> create table branch
  2      (branch_name varchar(15) not null,
  3       branch_city varchar(15) not null,
  4       assets number not null,
  5       primary key(branch_name));
Table created.

SQL> create table customer
  2      (customer_name varchar(15) not null,
  3       customer_street varchar(12) not null,
  4       customer_city varchar(15) not null,
  5       primary key(customer_name));
Table created.

SQL> create table account
  2      (account_number varchar(15) not null,
  3       branch_name varchar(15) not null,
  4       balance number not null,
  5       primary key(account_number),
  6       foreign key(branch_name) references branch(branch_name));
Table created.

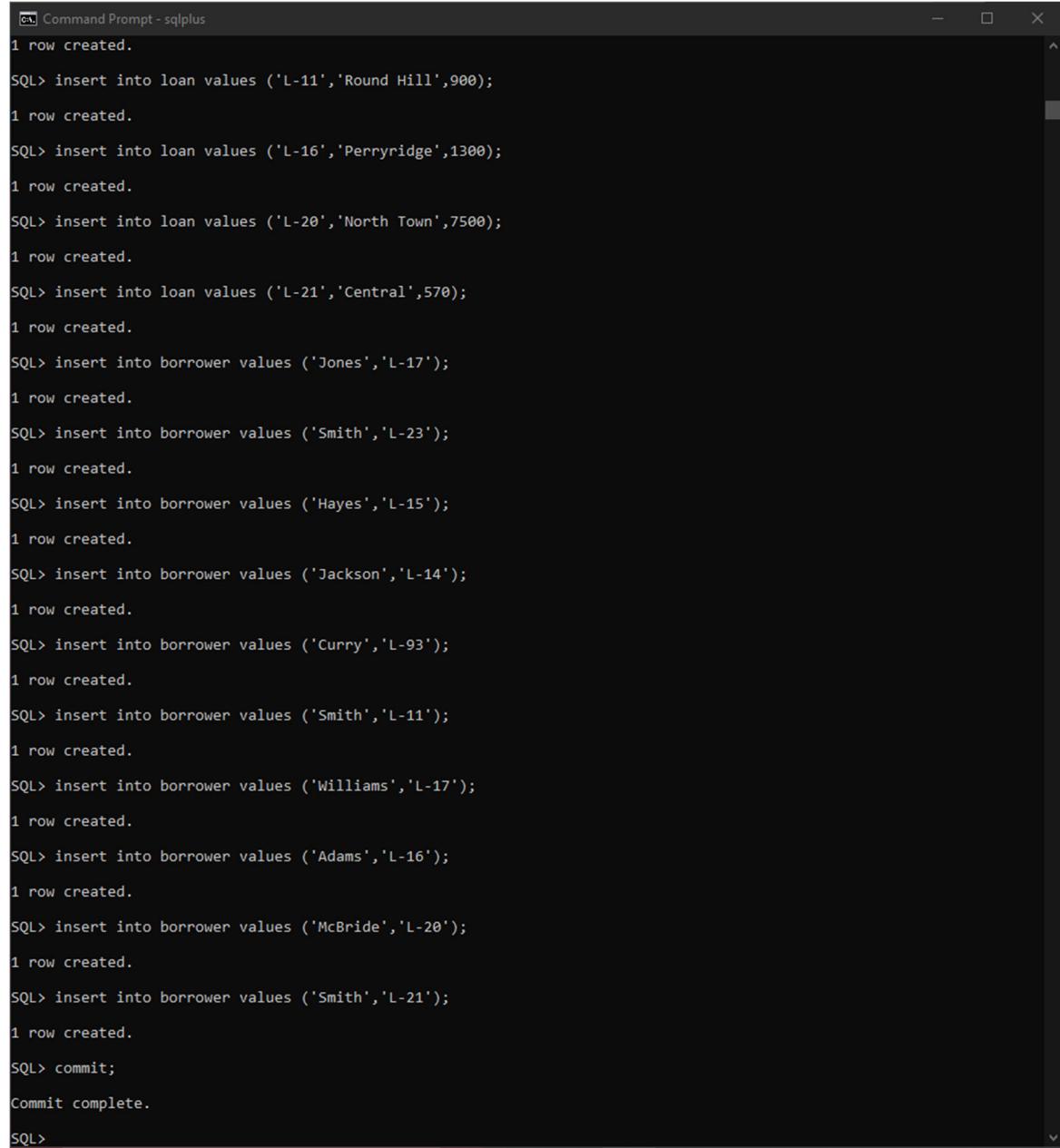
SQL>
SQL> create table loan
  2      (loan_number varchar(15) not null,
  3       branch_name varchar(15) not null,
  4       amount number not null,
  5       primary key(loan_number),
  6       foreign key(branch_name) references branch(branch_name));
Table created.

SQL> create table depositor
  2      (customer_name varchar(15) not null,
  3       account_number varchar(15) not null,
  4       primary key(customer_name, account_number),
  5       foreign key(account_number) references account(account_number),
  6       foreign key(customer_name) references customer(customer_name));
Table created.

SQL> create table borrower
  2      (customer_name varchar(15) not null,
  3       loan_number varchar(15) not null,
  4       primary key(customer_name, loan_number),
  5       foreign key(customer_name) references customer(customer_name),
  6       foreign key(loan_number) references loan(loan_number));
Table created.

SQL> -
```

Then I inserted all the values provided in the banking.sql file and committed the data so that I can rollback whenever I needed.



```
1 row created.

SQL> insert into loan values ('L-11','Round Hill',900);

1 row created.

SQL> insert into loan values ('L-16','Perryridge',1300);

1 row created.

SQL> insert into loan values ('L-20','North Town',7500);

1 row created.

SQL> insert into loan values ('L-21','Central',570);

1 row created.

SQL> insert into borrower values ('Jones','L-17');

1 row created.

SQL> insert into borrower values ('Smith','L-23');

1 row created.

SQL> insert into borrower values ('Hayes','L-15');

1 row created.

SQL> insert into borrower values ('Jackson','L-14');

1 row created.

SQL> insert into borrower values ('Curry','L-93');

1 row created.

SQL> insert into borrower values ('Smith','L-11');

1 row created.

SQL> insert into borrower values ('Williams','L-17');

1 row created.

SQL> insert into borrower values ('Adams','L-16');

1 row created.

SQL> insert into borrower values ('McBride','L-20');

1 row created.

SQL> insert into borrower values ('Smith','L-21');

1 row created.

SQL> commit;

Commit complete.

SQL>
```

Task 1. Find all customer names who have an account as well as a loan (with and without ‘intersect’ clause).

```
Command Prompt - sqlplus

SQL> select distinct customer.customer_name
  2  from customer,depositor,borrower
  3  where customer.customer_name = depositor.customer_name and customer.customer_name = borrower.customer_name
  4  order by customer.customer_name;

CUSTOMER_NAME
-----
Hayes
Jones
Smith

SQL> select distinct customer_name
  2  from customer natural join depositor natural join borrower
  3  order by customer_name;

CUSTOMER_NAME
-----
Hayes
Jones
Smith

SQL> select customer_name
  2  from customer
  3  intersect
  4  select customer_name
  5  from depositor
  6  intersect
  7  select customer_name
  8  from borrower;

CUSTOMER_NAME
-----
Hayes
Jones
Smith

SQL>
```

2. Find all customer-related information who have an account or a loan (with and without ‘union’ clause).

```
Command Prompt - sqlplus

SQL> select customer_name
  2  from depositor
  3  union
  4  select customer_name
  5  from borrower;

CUSTOMER_NAME
-----
Adams
Curry
Hayes
Jackson
Johnson
Jones
Lindsay
Majeris
McBride
Smith
Turner

CUSTOMER_NAME
-----
Williams

12 rows selected.

SQL> select distinct customer.customer_name
  2  from customer,depositor,borrower
  3  where customer.customer_name = depositor.customer_name or customer.customer_name = borrower.customer_name
  4  order by customer.customer_name;

CUSTOMER_NAME
-----
Adams
Curry
Hayes
Jackson
Johnson
Jones
Lindsay
Majeris
McBride
Smith
Turner

CUSTOMER_NAME
-----
Williams

12 rows selected.

SQL>
```

3. Find all customer names and their cities who have a loan but not an account (with and without ‘minus’ clause).

```
Command Prompt - sqlplus

SQL> select customer_name
  2  from (select customer_name
  3  from customer
  4  intersect
  5  select customer_name
  6  from borrower)
  7 where customer_name not in ( select customer_name from depositor)
  8 order by customer_name;

CUSTOMER_NAME
-----
Adams
Curry
Jackson
McBride
Williams

SQL> select customer_name
  2  from customer
  3  intersect
  4  select customer_name
  5  from borrower
  6  minus
  7  select customer_name
  8  from depositor;

CUSTOMER_NAME
-----
Adams
Curry
Jackson
McBride
Williams

SQL>
```

4. Find the total assets of all branches.

```
Select Command Prompt - sqlplus

SQL> select sum(assets) as Total_Asset
  2  from branch;

TOTAL_ASSET
-----
24600480

SQL>
```

5. Find the total number of accounts at each branch city.

```
Command Prompt - sqlplus

SQL> select branch_city, count(account_number) as number_of_accounts
  2  from branch natural join account
  3  group by branch_city;

BRANCH_CITY      NUMBER_OF_ACCOUNTS
-----
Horseneck                  4
Brooklyn                   2
Palo Alto                  1
Rye                        2

SQL>
```

6. Find the average balance of accounts at each branch and display them in descending order of average balance.

```
Command Prompt - sqlplus

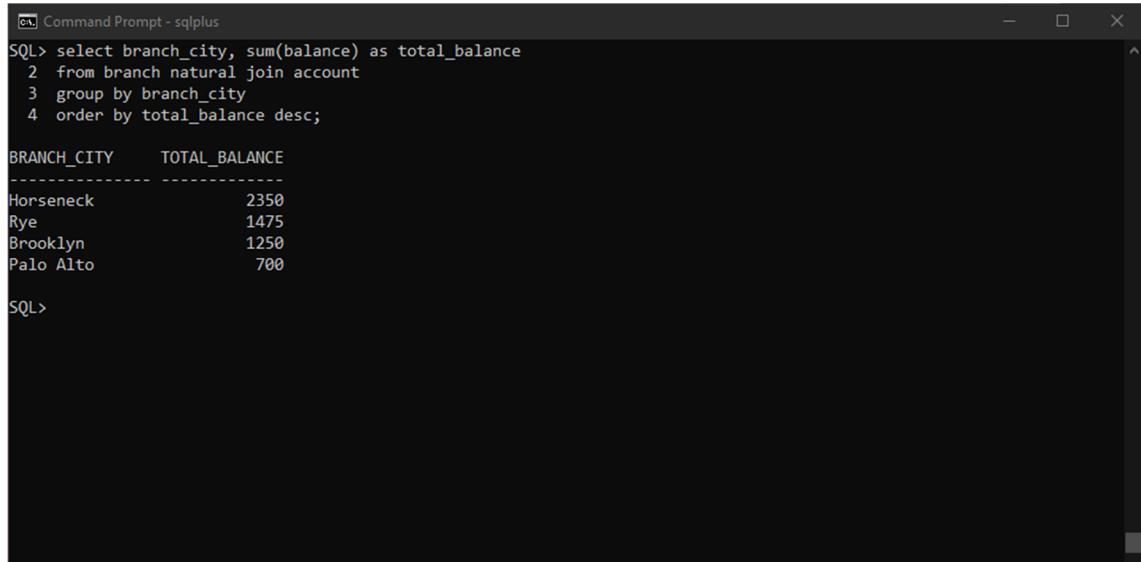
SQL> select branch_name, avg(balance) as average_balance
  2  from branch natural join account
  3  group by branch_name
  4  order by branch_name asc;

BRANCH_NAME      AVERAGE_BALANCE
-----
Brighton                 750
Central                  850
Downtown                500
Mianus                   700
North Town               625
Perryridge               650
Redwood                  700
Round Hill               350

8 rows selected.

SQL>
```

7. Find the total balance of accounts at each branch city.

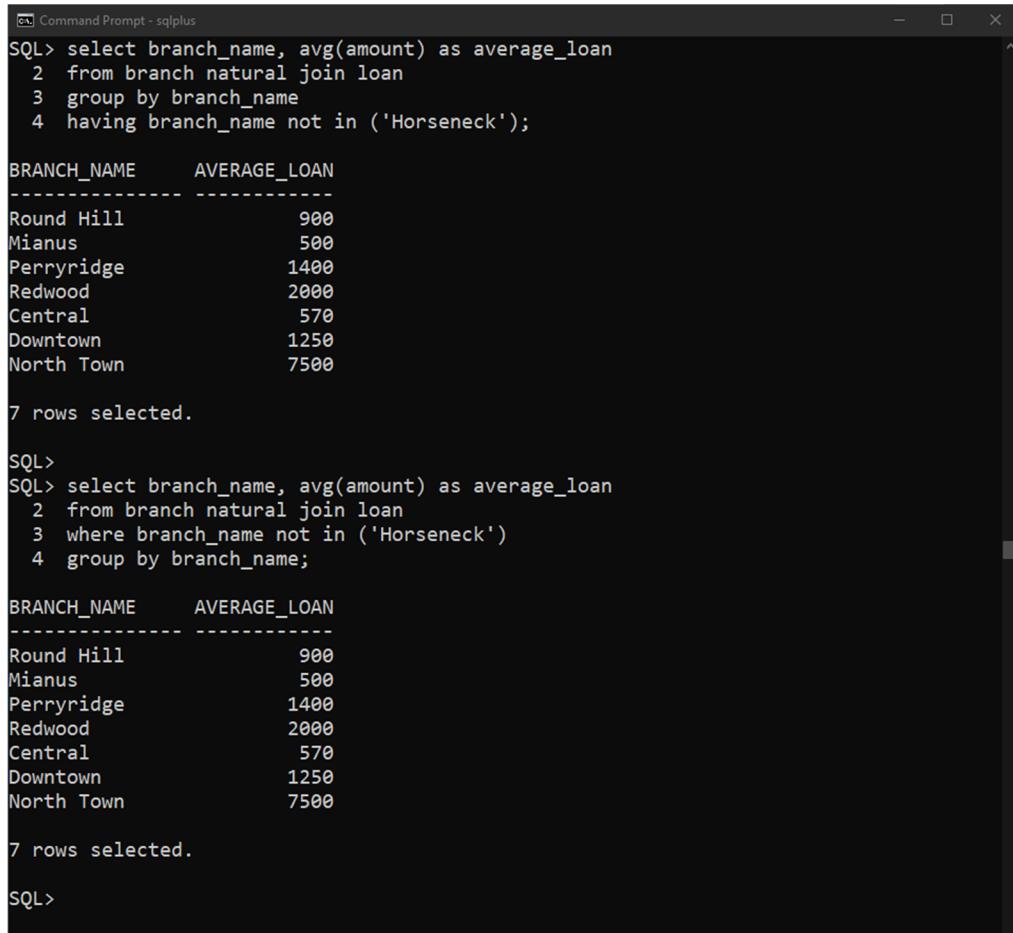


```
SQL> select branch_city, sum(balance) as total_balance
  2  from branch natural join account
  3  group by branch_city
  4  order by total_balance desc;

BRANCH_CITY      TOTAL_BALANCE
-----  -----
Horseneck          2350
Rye                 1475
Brooklyn            1250
Palo Alto             700

SQL>
```

8. Find the average loan amount at each branch. Do not include any branch which is located in 'Horseneck' city (with and without 'having' clause).



```
SQL> select branch_name, avg(amount) as average_loan
  2  from branch natural join loan
  3  group by branch_name
  4  having branch_name not in ('Horseneck');

BRANCH_NAME      AVERAGE_LOAN
-----  -----
Round Hill           900
Mianus                500
Perryridge            1400
Redwood               2000
Central                  570
Downtown                1250
North Town              7500

7 rows selected.

SQL>
SQL> select branch_name, avg(amount) as average_loan
  2  from branch natural join loan
  3  where branch_name not in ('Horseneck')
  4  group by branch_name;

BRANCH_NAME      AVERAGE_LOAN
-----  -----
Round Hill           900
Mianus                500
Perryridge            1400
Redwood               2000
Central                  570
Downtown                1250
North Town              7500

7 rows selected.

SQL>
```

9. Find the customer name and account number of the account which has the highest balance (with and without ‘all’ clause).

```
Command Prompt - sqlplus

SQL> select account_number, customer_name, balance
  2  from account natural join depositor
  3  where balance = ALL (
  4      select max(balance) as max_balance
  5      from account natural join depositor
  6  );

ACCOUNT_NUMBER CUSTOMER_NAME      BALANCE
-----          -----
A-201           Johnson          900

SQL>
SQL> select *
  2  from (select account_number, customer_name, max(balance) as max_balance
  3  from account natural join depositor
  4  group by account_number, customer_name
  5  order by max_balance desc)
  6  where rownum <=1;

ACCOUNT_NUMBER CUSTOMER_NAME      MAX_BALANCE
-----          -----
A-201           Johnson          900

SQL>
```

10. Find all customer-related information who have an account in a branch, located in the same city as they live.

```
Command Prompt - sqlplus

SQL> select c.customer_name,c.customer_street,c.customer_city
  2  from account a, customer c, branch b , depositor d
  3  where c.customer_city = b.branch_city and
  4  c.customer_name = d.customer_name and
  5  a.account_number = d.account_number and
  6  a.branch_name = b.branch_name;

CUSTOMER_NAME      CUSTOMER_STR CUSTOMER_CITY
-----          -----
Majeris           First        Rye
Smith            Main         Rye

SQL>
```

11. For each branch city, find the average amount of all the loans opened in a branch located in that branch city. Do not include any branch city in the result where the average amount of all loans opened in a branch located in that city is less than 1500. (with and without using ‘having’ clause).

```
Command Prompt - sqlplus

SQL> select branch_city, avg(amount) as average_loan
  2  from branch natural join loan
  3  group by branch_city
  4  having avg(amount) >=1500
  5  order by average_loan desc;

BRANCH_CITY      AVERAGE_LOAN
-----
Rye                  4035
Palo Alto            2000

SQL> select *
  2  from (select branch_city, avg(amount) as average_loan
  3  from branch natural join loan
  4  group by branch_city
  5  order by average_loan desc) temp
  6  where temp.average_loan >= 1500;

BRANCH_CITY      AVERAGE_LOAN
-----
Rye                  4035
Palo Alto            2000

SQL>
```

12. Find those branch names which have a total account balance greater than the average total balance among all the branches.

```
Command Prompt - sqlplus

SQL> select branch_name
  2  from (select avg(assets) as average_assets from branch) temp1,
  3        (select branch_name, sum(assets) as total_assets from branch group by branch_name) temp2
  4  where average_assets < total_assets;

BRANCH_NAME
-----
Brighton
North Town
Round Hill

SQL>
```

13. Find the name of the customer who has at least one loan that can be paid off by his/her total balance.

```
Command Prompt - sqlplus
SQL> select distinct c.customer_name
  2  from account a, depositor d, customer c, borrower b, loan l
  3  where a.account_number = d.account_number and
  4  c.customer_name = d.customer_name and
  5  c.customer_name = b.customer_name and
  6  b.loan_number = l.loan_number and
  7  a.balance > l.amount;

CUSTOMER_NAME
-----
Smith

SQL>
```

14. Find the branch information for cities where at least one customer lives who do not have any account or any loans. The branch must have given some loans and has accounts opened by other customers.

```
Command Prompt - sqlplus
SQL> select branch.branch_name, branch_city, branch.assets
  2  from branch
  3  where branch.branch_city in (select customer.customer_city
  4                                from customer
  5                                where customer.customer_name not in (select depositor.customer_name
  6                                         from depositor)
  7                                and customer.customer_name not in (select borrower.customer_name
  8                                         from borrower))
  9      and branch.branch_name in (select loan.branch_name
 10                                 from loan)
 11      and branch.branch_name in (select account.branch_name
 12                                 from account);

BRANCH_NAME     BRANCH_CITY        ASSETS
Downtown       Brooklyn           900000
SQL>
```

## The Code

```
<--- 1 --->
select distinct customer.customer_name
from customer,depositor,borrower
where customer.customer_name = depositor.customer_name and customer.customer_name
= borrower.customer_name
order by customer.customer_name;

select distinct customer_name
from customer natural join depositor natural join borrower
order by customer_name;

select customer_name
```

```
from customer
intersect
select customer_name
from depositor
intersect
select customer_name
from borrower;

<--- 2 --->
select customer_name
from depositor
union
select customer_name
from borrower;

select distinct customer.customer_name
from customer,depositor,borrower
where customer.customer_name = depositor.customer_name or customer.customer_name
= borrower.customer_name
order by customer.customer_name;

<--- 3 --->
select customer_name
from customer
intersect
select customer_name
from borrower
minus
select customer_name
from depositor;

select customer_name
from (select customer_name
from customer
intersect
select customer_name
from borrower)
where customer_name not in ( select customer_name from depositor)
order by customer_name;

<--- 4 --->
select sum(assets) as Total_Asset
from branch;

<--- 5 --->
```

```
select branch_city, count(account_number) as number_of_accounts
from branch natural join account
group by branch_city;

<--- 6 --->
select branch_name, avg(balance) as average_balance
from branch natural join account
group by branch_name
order by branch_name asc;

<--- 7 --->
select branch_city, sum(balance) as total_balance
from branch natural join account
group by branch_city
order by total_balance desc;

<--- 8 --->
select branch_name, avg(amount) as average_loan
from branch natural join loan
group by branch_name
having branch_name not in ('Horseneck');

select branch_name, avg(amount) as avg_loan
from branch natural join loan
where branch_name not in ('Horseneck')
group by branch_name;

<--- 9 --->
select account_number, customer_name, balance
from account natural join depositor
where balance = ALL (
    select max(balance) as max_balance
    from account natural join depositor
);

select *
from (select account_number, customer_name, max(balance) as max_balance
from account natural join depositor
group by account_number, customer_name
order by max_balance desc)
where rownum <=1;

<--- 10 --->
select c.customer_name,c.customer_street,c.customer_city
```



```
        from
borrower))
    and branch.branch_name in (select loan.branch_name
                                from loan)
    and branch.branch_name in (select account.branch_name
                                from account);
```

## Problems

The volume of tasks was much too large for us to complete in a single sitting. With that being the case, I was hung up on one part. I spent too much time on it. For that, I was unable to complete too many tasks in time. I hope this detailed lab report will make up for that.

## Conclusion

With that, another lab came to an end. This lab wasn't a pleasant experience. However, the things I learned here were crucial and I'm excited to learn more.