Islamic University of Technology (IUT)

CSE 4308: Database Management Systems Lab

Lab Report # 4

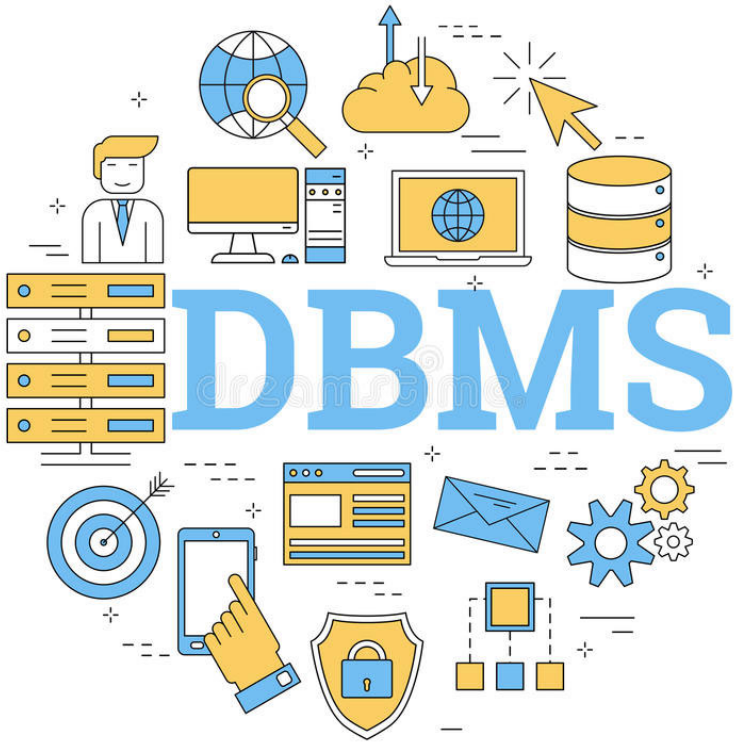
Submitted to:

Md. Bakhtiar Hasan,

Asst. Professor, CSE.

Zannatun Naim Srsity,

Lecturer, CSE.



Submitted by:

M M Nazmul Hossain
ID 200042118
CSE (SWE)

Submission Date:

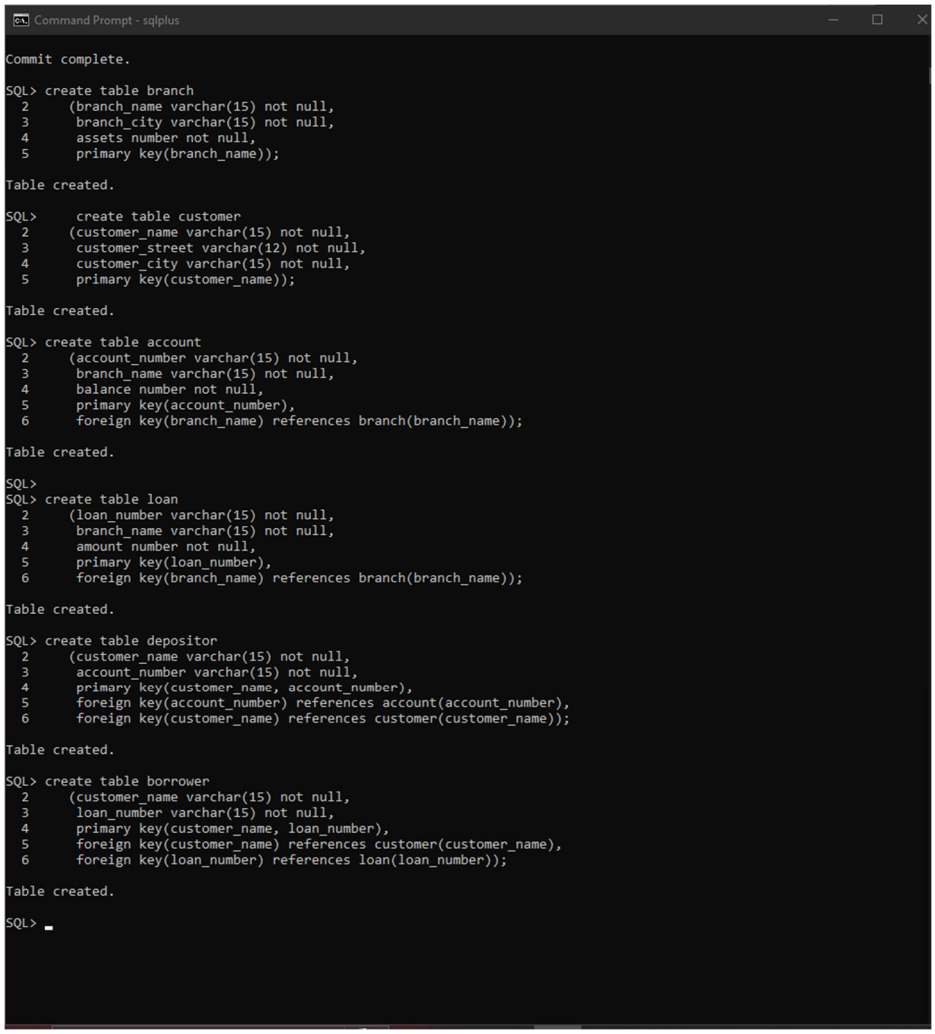
14.09.2022

Introduction

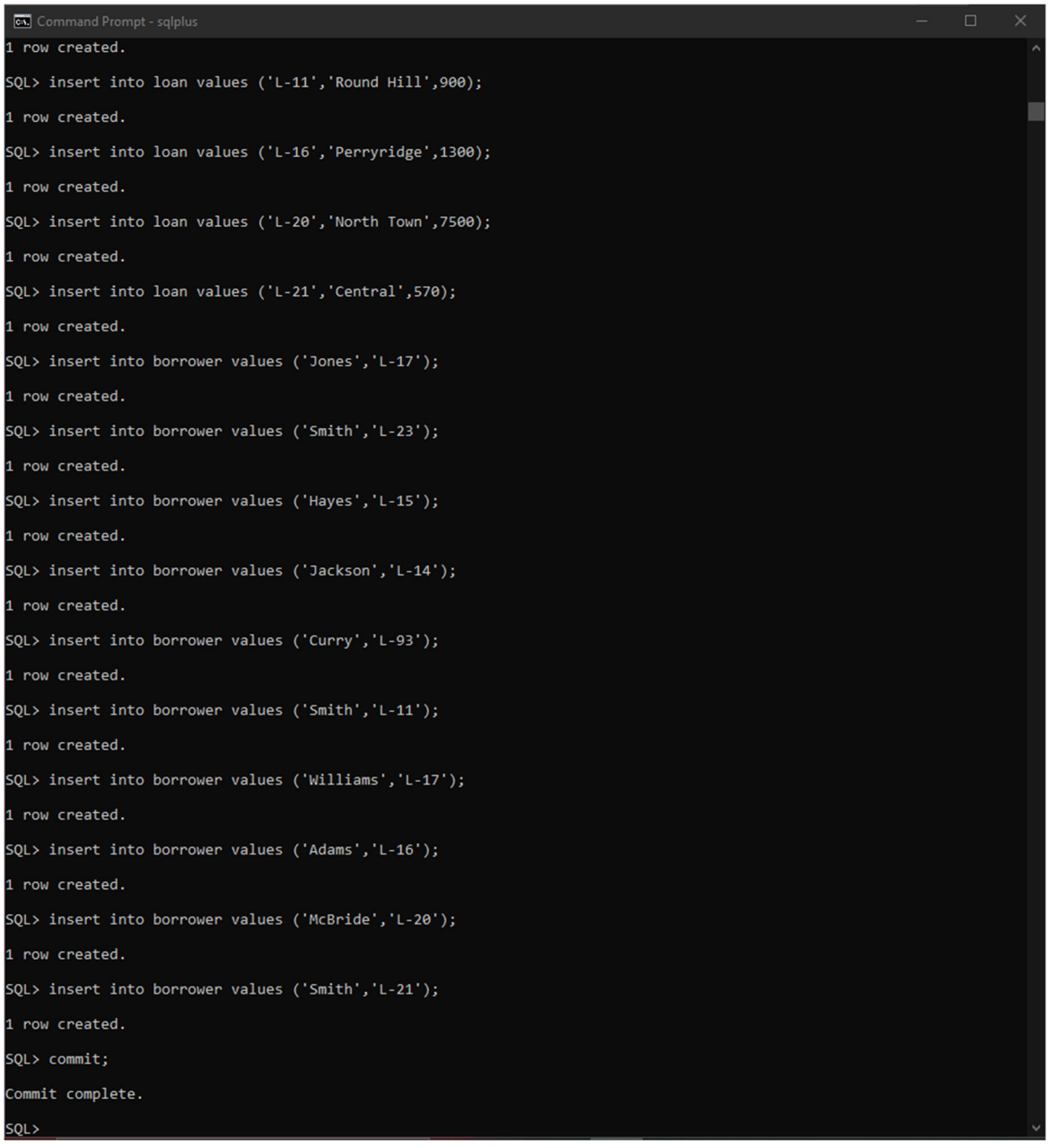
In our third database management systems lab, the goal of this lab was to realize the importance of set operations and the different types of views that allow us to use different function statements.

Method

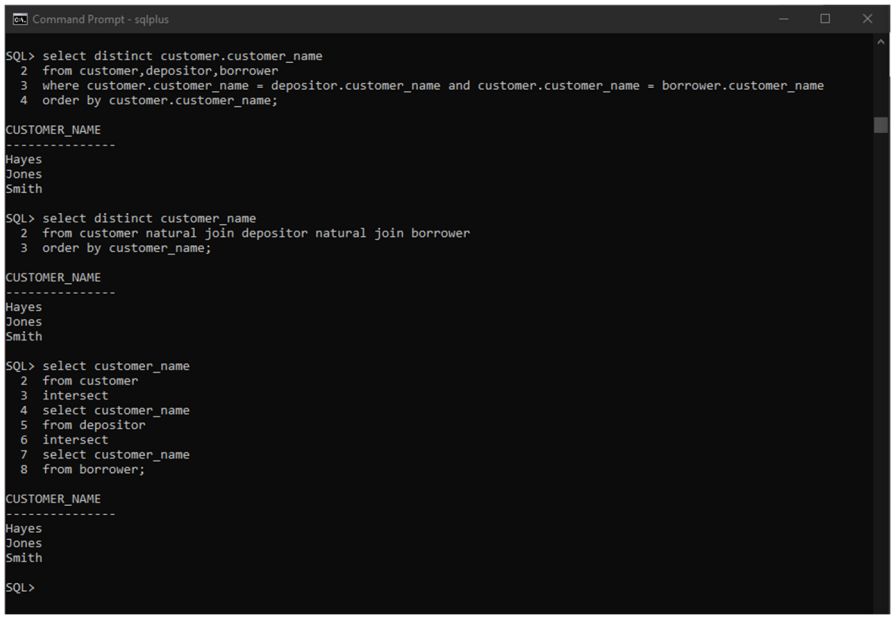
First, I fixed all the errors present in the banking.sql file, where there were only tab spaces given which the cmd line didn't register. Then I created all the tables after deleting the similar tables created in the previous lab.



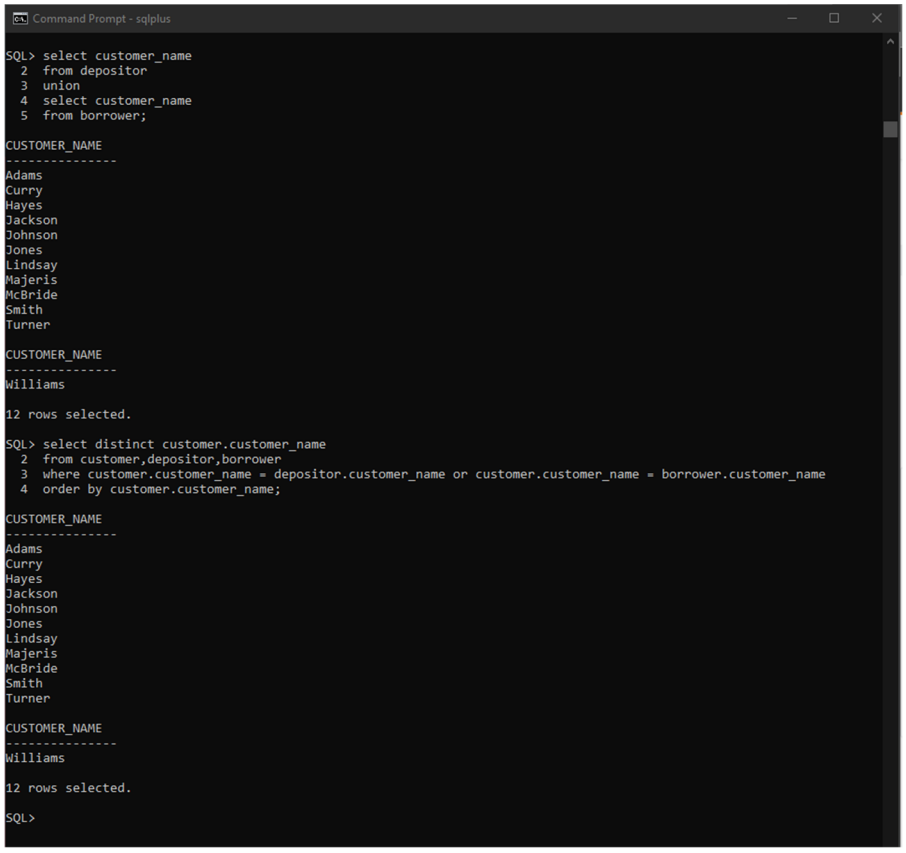
Then I inserted all the values provided in the banking.sql file and committed the data so that I can rollback whenever I needed.



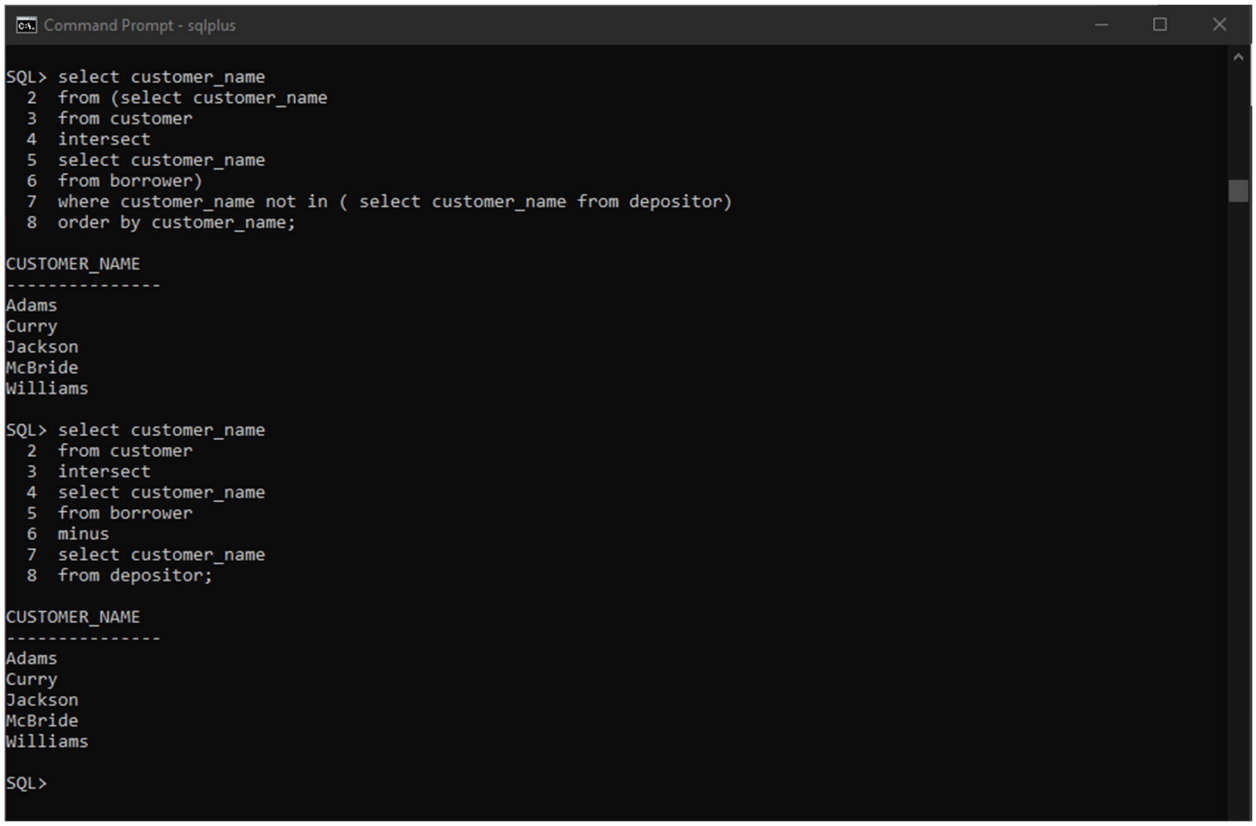
Task 1. Find all customer names who have an account as well as a loan (with and without ‘intersect’ clause).



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

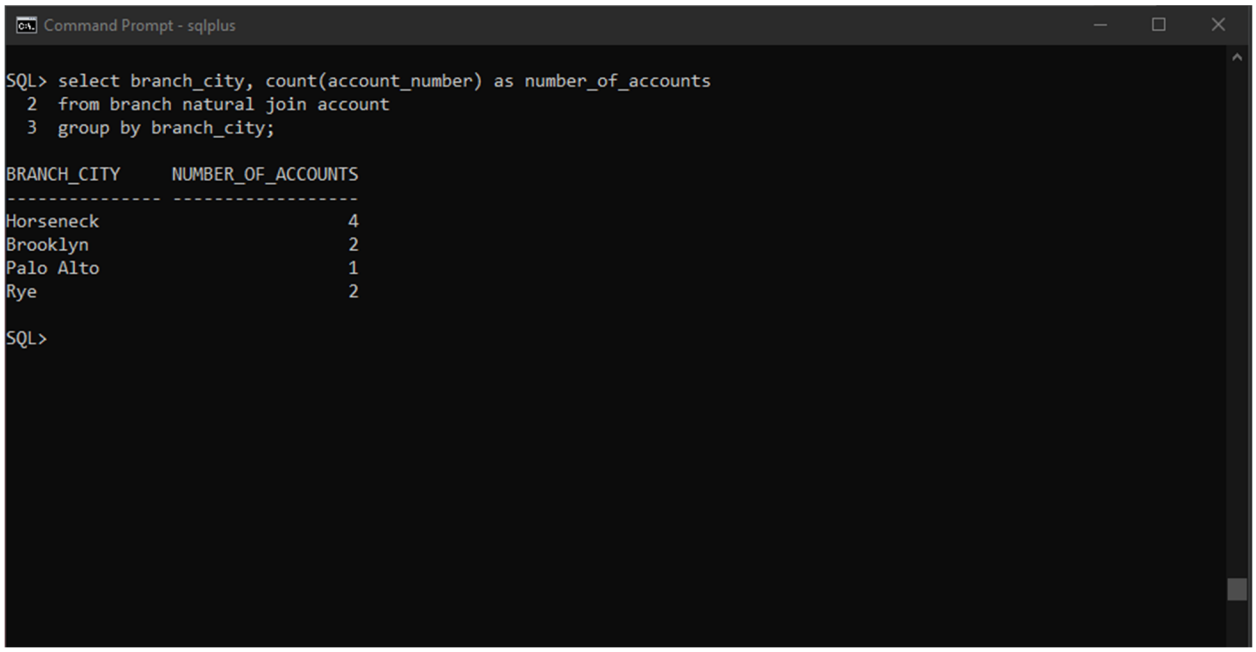


3. Find all customer names and their cities who have a loan but not an account (with and without ‘minus’ clause).

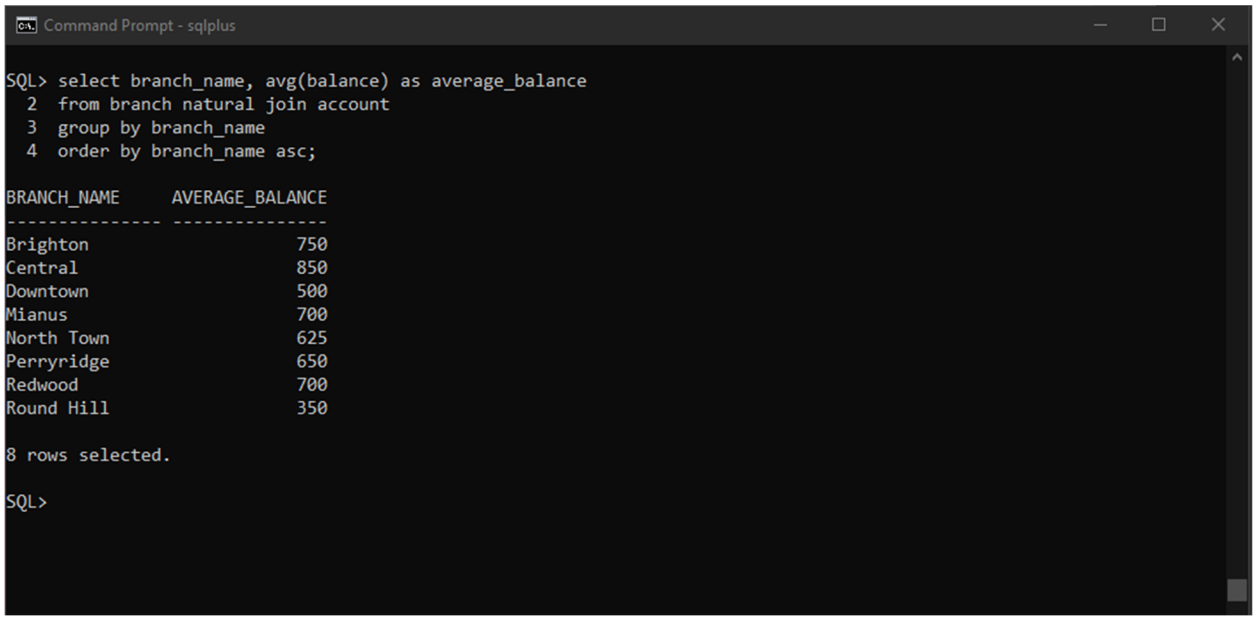


4. Find the total assets of all branches.

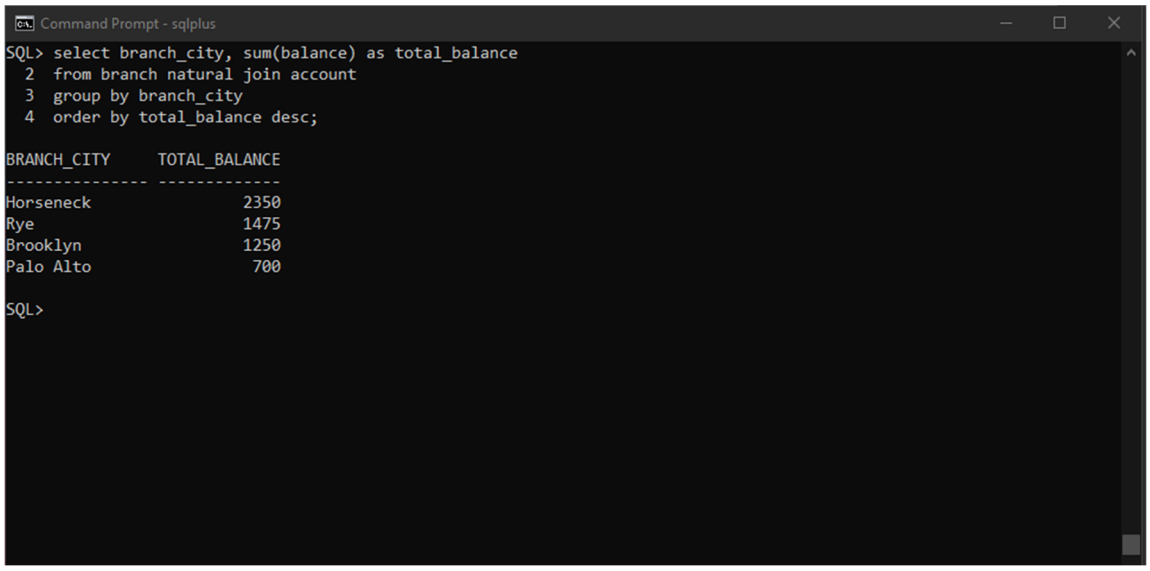
5. Find the total number of accounts at each branch city.



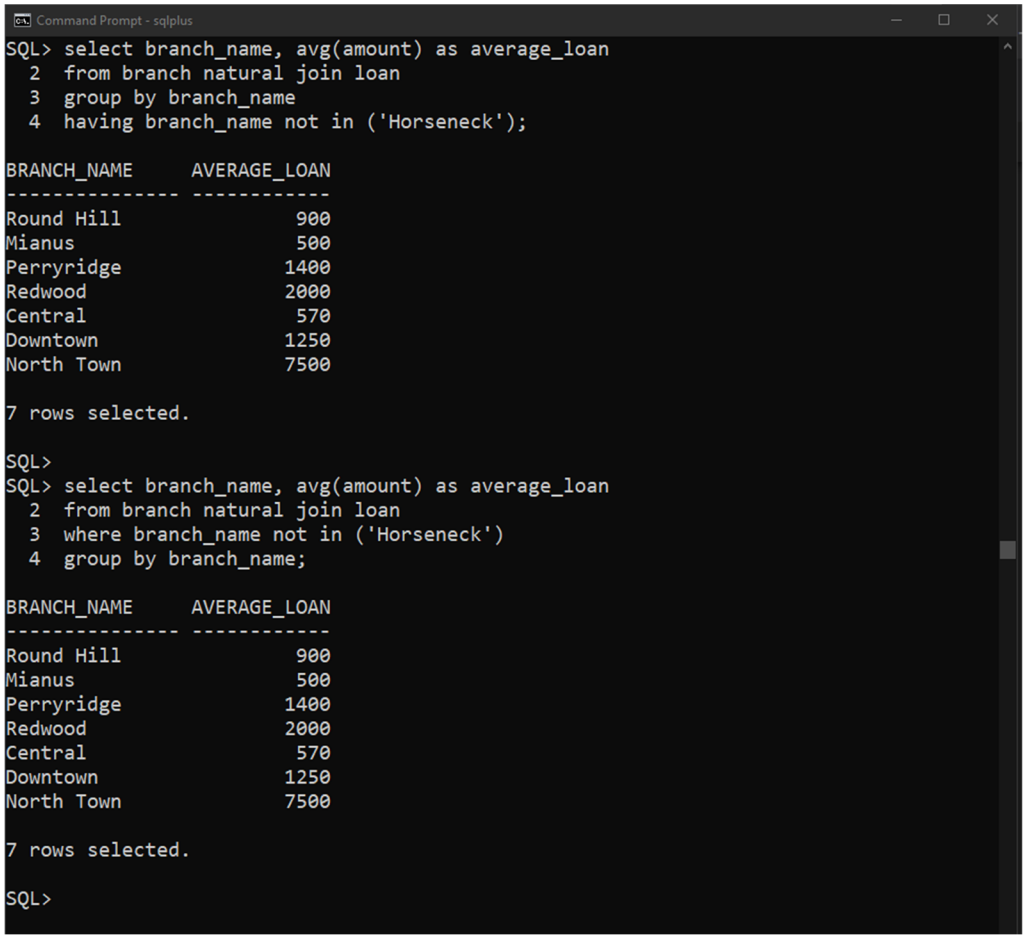
6. Find the average balance of accounts at each branch and display them in descending order of average balance.



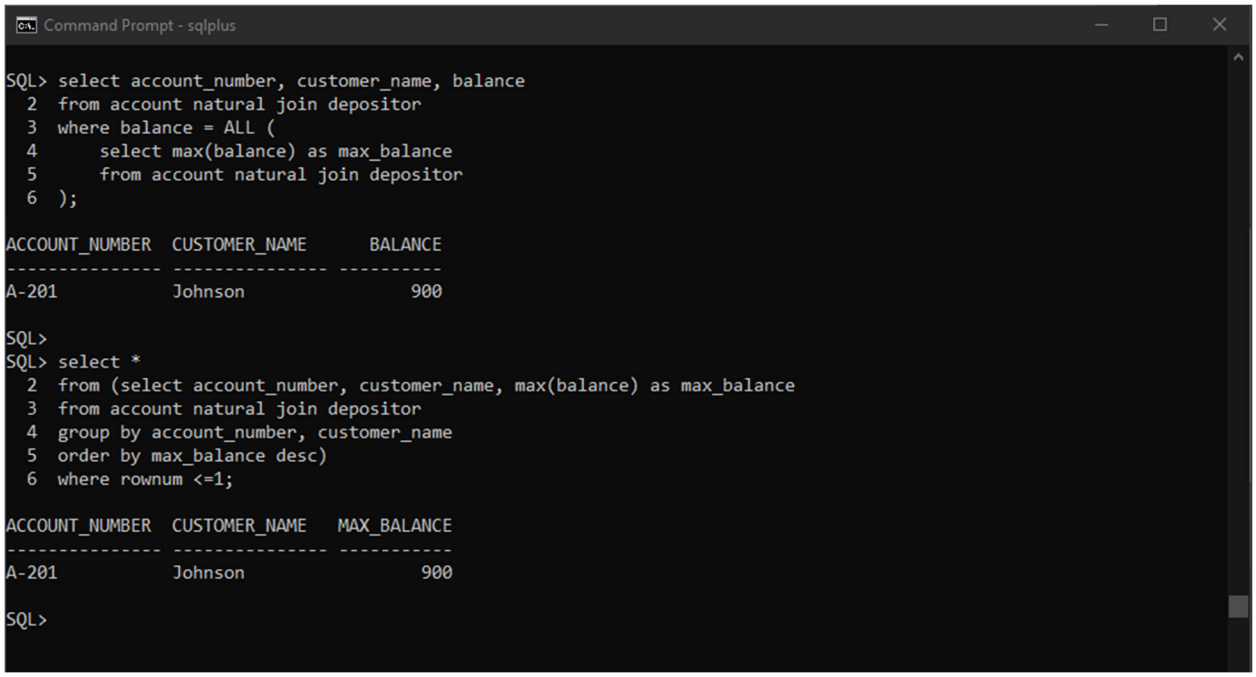
7. Find the total balance of accounts at each branch city.



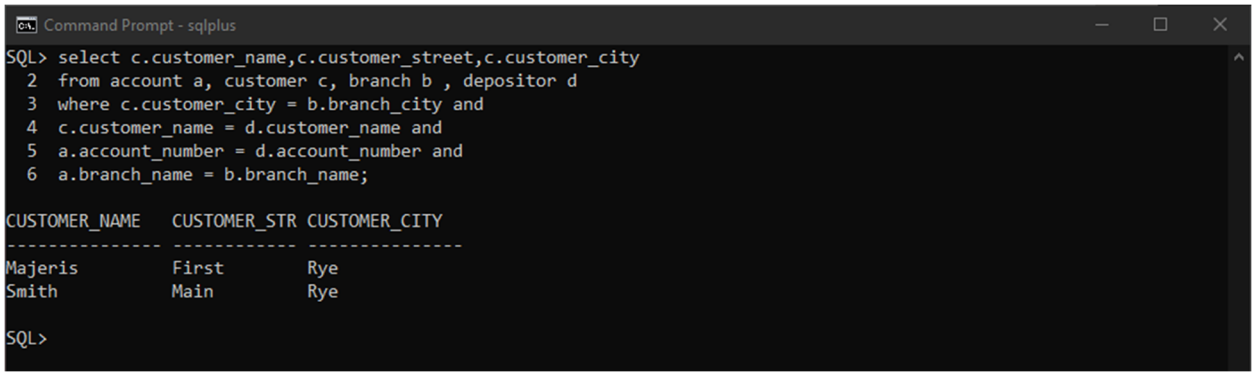
8. Find the average loan amount at each branch. Do not include any branch which is located in 'Horseneck' city (with and without 'having' clause).



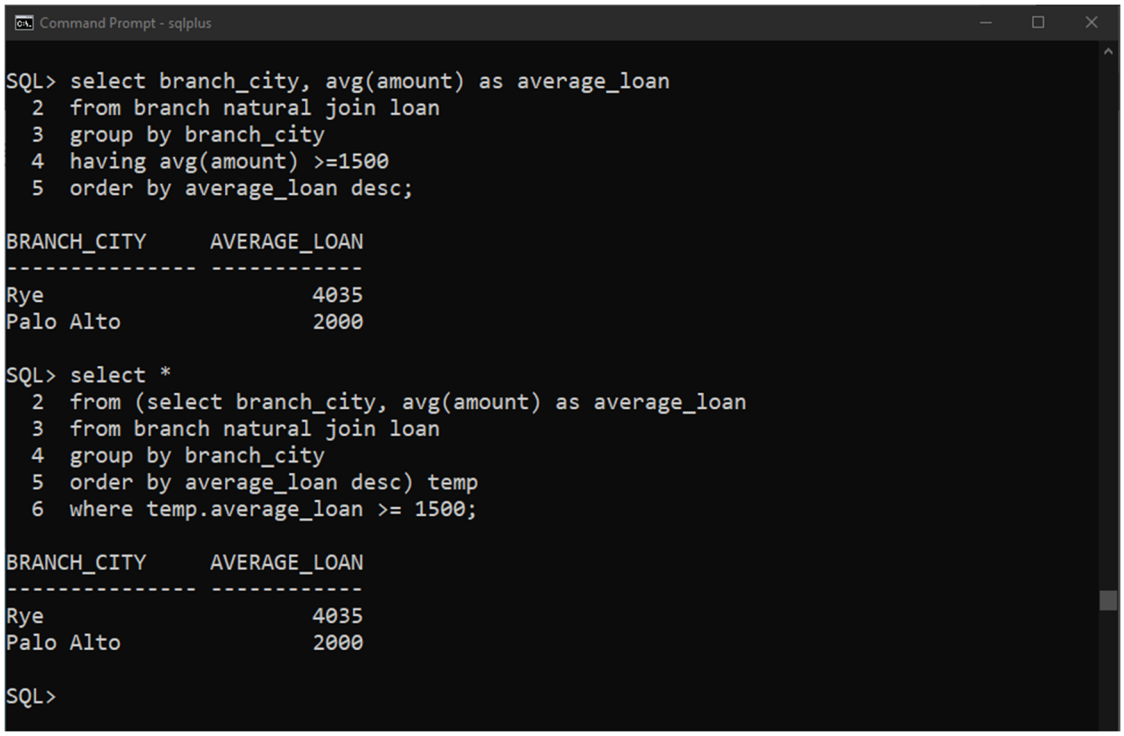
9. Find the customer name and account number of the account which has the highest balance (with and without ‘all’ clause).



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



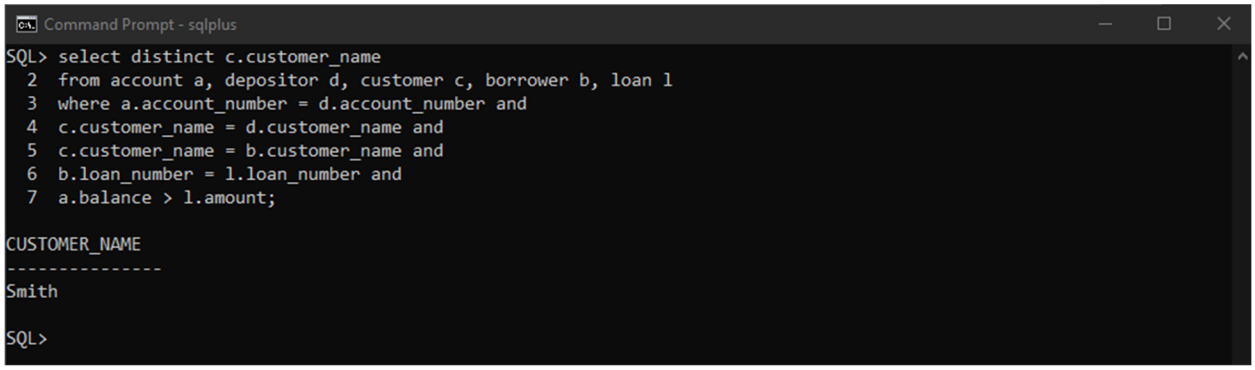
11. For each branch city, find the average amount of all the loans opened in a branch located in that branch city. Do not include any branch city in the result where the average amount of all loans opened in a branch located in that city is less than 1500. (with and without using ‘having’ clause).



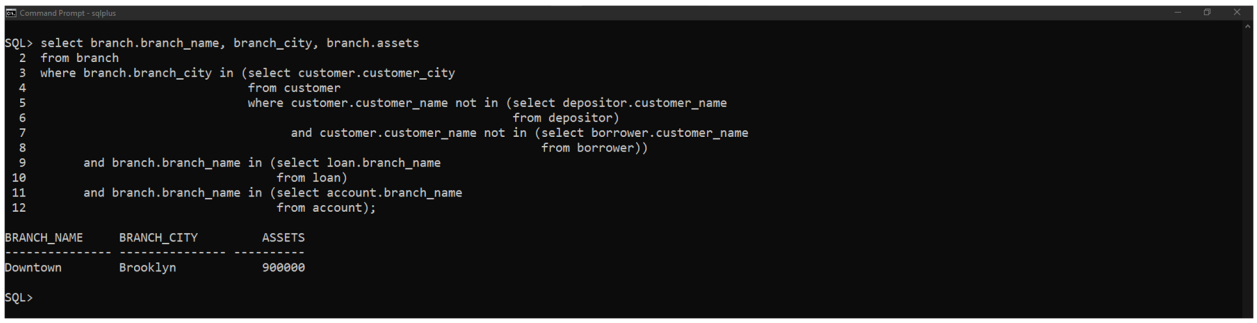
12. Find those branch names which have a total account balance greater than the average total

balance among all the branches.

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



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.



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 --->
```

```sql
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 account a, customer c, branch b , depositor d
where c.customer_city = b.branch_city and
c.customer_name = d.customer_name and
a.account_number = d.account_number and
a.branch_name = b.branch_name;

<--- 11 --->
select branch_city, branch_name, avg(amount) as average_loan
from branch natural join loan
group by branch_city,branch_name
having avg(amount) >=1500
order by average_loan desc;

select *
from (select branch_city, branch_name, avg(amount) as average_loan
from branch natural join loan
group by branch_city,branch_name
order by average_loan desc) temp
where temp.average_loan >= 1500;

<--- 12 --->
select branch_name
from (select avg(assets) as average_assets from branch) temp1,
    (select branch_name, sum(assets) as total_assets from branch group by
branch_name) temp2
where average_assets < total_assets;

<--- 13 --->
select distinct c.customer_name
from account a, depositor d, customer c, borrower b, loan l
where a.account_number = d.account_number and
c.customer_name = d.customer_name and
c.customer_name = b.customer_name and
b.loan_number = l.loan_number and
a.balance > l.amount;

select branch.branch_name, branch_city, branch.assets
from branch
where branch.branch_city in (select customer.customer_city
                             from customer
                             where customer.customer_name not in (select
depositor.customer_name
                                                                  from depositor)
                                   and customer.customer_name not in (select
borrower.customer_name
```

```
                                                                    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.