DBMS Assignment 4 - Lab 5 Report

Namisa Najah Raisa
Student ID: 210042112

September 13, 2023

1 Introduction

This is a report about the tasks given in the 5th lab.

2 Tasks

2.1 Task Overview

The lab consisted of writing SQL statements for 15 queries:

2.2 Statements

2.2.1 Writing the queries

```
--1. Find all customer names and their cities who have a loan
    but not an account.

select c.customer_name,c.customer_city
from customer c,borrower b,depositor d
where c.customer_name=b.customer_name
and c.customer_name=d.customer_name
and c.customer_name not in(
select customer_name from depositor);
```

```
SQL> --1
SQL> select c.customer_name,c.customer_city
  2  from customer c,borrower b,depositor d
  3  where c.customer_name=b.customer_name
  4  and c.customer_name=d.customer_name
  5  and c.customer_name not in(
  6  select customer_name from depositor);

no rows selected
```

```sql
--2.Find all customer names who have an account as well as a
   loan.

select c.customer_name
from customer c
where c.customer_name in(
select distinct d.customer_name
from depositor d
intersect
select distinct b.customer_name
from borrower b
);
```

```
SQL>
SQL> --2
SQL> select c.customer_name
  2  from customer c
  3  where c.customer_name in(
  4  select distinct d.customer_name
  5  from depositor d
  6  intersect
  7  select distinct b.customer_name
  8  from borrower b
  9  );

CUSTOMER_NAME
---------------
HAYES
JONES
SMITH
```

```sql
--3.Show the count of accounts that were opened in each month
    along with the month.

select to_char(acc_opening_date,'yyyy-mm-dd')as opening_month
   ,count(*) as acc_count
from account
group by to_char(acc_opening_date,'yyyy-mm-dd');
```

```
SQL>
SQL> --3
SQL> select to_char(acc_opening_date,'yyyy-mm-dd')as opening_month,count(*) as acc_count
  2  from account
  3  group by to_char(acc_opening_date,'yyyy-mm-dd');

OPENING_MO  ACC_COUNT
---------- ----------
2019-03-04          1
2009-09-29          1
2011-07-19          1
2010-03-18          1
2016-05-18          1
2013-08-24          1
2012-04-20          1
2009-01-09          1
2018-11-11          1

9 rows selected.
```

```sql
--4.Find the months between the last acc_opening_date and
   last loan_date of customer ‘Smith‘.

select months_between(
(select max(acc_opening_date)
from account
where account_number in(
select account_number
from depositor
where customer_name = 'Smith')),
(select max(loan_date)
from loan
where loan_number in(
select loan_number
from borrower
where customer_name = 'Smith'
)))
as month
from dual;
```

```
SQL>
SQL> --4
SQL> select months_between(
  2  (select max(acc_opening_date)
  3  from account
  4  where account_number in(
  5  select account_number
  6  from depositor
  7  where customer_name = 'Smith')),
  8  (select max(loan_date)
  9  from loan
 10  where loan_number in(
 11  select loan_number
 12  from borrower
 13  where customer_name = 'Smith'
 14  )))
 15  as month
 16  from dual;

     MONTH
----------
```

```sql
--5.Find the average loan amount at each branch. Do not
   include any branch which is located in a that has the
   substring, ‘Horse‘ in its name.

select branch_name,avg(amount) as_avg_loan
from loan
where branch_name not in(
select branch_name
from branch
where branch_city like '%Horse%')
group by branch_name;
```

```
SQL> --5
SQL> select branch_name,avg(amount) as_avg_loan
  2  from loan
  3  where branch_name not in(
  4  select branch_name
  5  from branch
  6  where branch_city like '%Horse%')
  7  group by branch_name;

BRANCH_NAME     AS_AVG_LOAN
--------------- -----------
DOWNTOWN               1250
CENTRAL                 570
PERRYRIDGE             1400
REDWOOD                2000
NORTH TOWN             7500
ROUND HILL              900
MIANUS                  500

7 rows selected.
```

```sql
--6.Find the customer name and account number of the account
   that has the highest balance.

select d.customer_name, a.account_number
from depositor d,account a
where d.account_number=a.account_number
and balance = (select max(balance) from account);
```

```
SQL>
SQL> --6
SQL> select d.customer_name, a.account_number
  2  from depositor d,account a
  3  where d.account_number=a.account_number
  4  and balance = (select max(balance) from account);

CUSTOMER_NAME   ACCOUNT_NUMBER
--------------- ---------------
JOHNSON         A-201
```

```sql
--7.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.

select branch_city, avg(amount) as average_loan_amount
from loan, branch
where loan.branch_name = branch.branch_name
group by branch_city
having avg(amount)  >= 1500;
```

```
SQL>
SQL> --7
SQL> select branch_city, avg(amount) as average_loan_amount
  2  from loan, branch
  3  where loan.branch_name = branch.branch_name
  4  group by branch_city
  5  having avg(amount)  >= 1500;

BRANCH_CITY     AVERAGE_LOAN_AMOUNT
--------------- -------------------
RYE                            4035
PALO ALTO                      2000
```

```sql
--8.Show all the name of the customer with the suffix  '
   Eligible' who has at least one loan that can be paid off
   by his/her total balance.

select customer_name || ' Eligible'
as customer_name from depositor
where account_number in
(select account_number from account
where balance >=
(select sum(amount) from loan
where loan.branch_name = account.branch_name
and loan.loan_number in
(select loan_number from borrower
where borrower.customer_name = depositor.customer_name)
));
```

```
SQL>
SQL> --8
SQL> select customer_name || ' Eligible'
  2  as customer_name from depositor
  3  where account_number in
  4  (select account_number from account
  5  where balance >=
  6  (select sum(amount) from loan
  7  where loan.branch_name = account.branch_name
  8  and loan.loan_number in
  9  (select loan_number from borrower
 10  where borrower.customer_name = depositor.customer_name)
 11  )
 12  );

CUSTOMER_NAME
------------------------
HAYES Eligible
```

```
--9.Show all the branch names with suffixes ‘Elite‘ that have
    a total account balance greater than the (average total
   balance + 500), ‘Moderate‘ that have a total account
   balance in between(average total balance + 500) to (
   average total balance - 500), else ‘Poor‘.

select branch_name ||
case
when total_balance > (avg_total_balance + 500) then 'Elite'
when total_balance between (avg_total_balance + 500) and (
   avg_total_balance - 500) then 'Moderate'
else 'Poor'
end as branch_status
from(
select b.branch_name,(
select sum(a.balance)
from account a
where a.branch_name = b.branch_name
)as total_balance,(
select avg(a.balance)
from account a
where a.branch_name = b.branch_name
)as avg_total_balance
from branch b
) A;
```

```
SQL>
SQL> --9
SQL> select branch_name ||
  2  case
  3  when total_balance > (avg_total_balance + 500) then 'Elite'
  4  when total_balance between (avg_total_balance + 500) and (avg_total_balance - 500) then 'Moderate'
  5  else 'Poor'
  6  end as branch_status
  7  from(
  8  select b.branch_name,(
  9  select sum(a.balance)
 10  from account a
 11  where a.branch_name = b.branch_name
 12  )as total_balance,(
 13  select avg(a.balance)
 14  from account a
 15  where a.branch_name = b.branch_name
 16  )as avg_total_balance
 17  from branch b
 18  ) A;

BRANCH_STATUS
----------------------
DOWNTOWNPoor
REDWOODPoor
PERRYRIDGEElite
MIANUSPoor
ROUND HILLPoor
POWNALPoor
NORTH TOWNPoor
BRIGHTONPoor
CENTRALPoor

9 rows selected.
```

```sql
--10.Find the branch information for cities where at least
   one customer lives who does not have any account or any
   loans. The branch must have given some loans and has
   accounts opened by other customers.

elect distinct b.branch_name, b.branch_city
from branch b
where b.branch_city in (
select distinct c.customer_city
from customer c
where c.customer_name not in(
select distinct d.customer_name
from depositor d
)and c.customer_name not in(
select distinct bor.customer_name
from borrower bor
)
) and b.branch_name in(
select distinct a.branch_name
from account a
)and b.branch_name in(
select distinct l.branch_name
from loan l
);
```

```
SQL>
SQL> --10
SQL>
SQL> select distinct b.branch_name, b.branch_city
  2  from branch b
  3  where b.branch_city in (
  4  select distinct c.customer_city
  5  from customer c
  6  where c.customer_name not in(
  7  select distinct d.customer_name
  8  from depositor d
  9  )and c.customer_name not in(
 10  select distinct bor.customer_name
 11  from borrower bor
 12  )
 13  ) and b.branch_name in(
 14  select distinct a.branch_name
 15  from account a
 16  )and b.branch_name in(
 17  select distinct l.branch_name
 18  from loan l
 19  );

BRANCH_NAME     BRANCH_CITY
--------------- ---------------
DOWNTOWN        BROOKLYN
```

```sql
--11.Create a new customer_new table using a similar
   structure to the customer table.

create table customer_new as
select *
from customer
where 1=2;
```

```
SQL>
SQL> --11
SQL> create table customer_new as
  2  select *
  3  from customer
  4  where 1=2;

Table created.
```

```sql
--12. In the customer_new table insert only those customers
    who have either an account or a loan.

insert into customer_new (customer_name, customer_street,
    customer_city)
select c.customer_name, c.customer_street, c.customer_city
from customer c
where c.customer_name in(
select distinct d.customer_name
from depositor d
union
select distinct b.customer_name
from borrower b
);
```

```
SQL>
SQL> --12
SQL> insert into customer_new (customer_name, customer_street, customer_city)
  2  select c.customer_name, c.customer_street, c.customer_city
  3  from customer c
  4  where c.customer_name in(
  5  select distinct d.customer_name
  6  from depositor d
  7  union
  8  select distinct b.customer_name
  9  from borrower b
 10  );

11 rows created.
```

```sql
--13. Add a new column Status in customer_new table of
    varchar2(15) type.

```

```sql
alter table customer_new
add status  varchar2(15);
```

```
SQL>
SQL> --13
SQL> alter table customer_new
  2  add status  varchar2(15);

Table altered.
```

```sql
--14.For each customer if his/her total balance is greater
   than the total loan then set the status ‘In savings‘, if
   the vise versa then ‘In loan‘, lastly if both of the
   amounts are the same then ‘In Breakeven‘.

update customer_new c
set c.status=(
case
when
(select sum(a.balance)
from account a
where a.branch_name in
(select distinct branch_name
from depositor d
where d.customer_name = c.customer_name)
)>(
select sum(l.amount)
from loan l
where l.branch_name in
(select distinct branch_name
from borrower b
where b.customer_name = c.customer_name)) then 'In savings'
when
(select sum(a.balance)
from account a
where a.branch_name in
(select distinct branch_name
from depositor d
where d.customer_name = c.customer_name)
)<(
```

```
28 select sum(l.amount)
29 from loan l
30 where l.branch_name in
31 (select distinct branch_name
32 from borrower b
33 where b.customer_name = c.customer_name))
34 then 'In loan'
35 else 'In Breakeven'
36 end
37 );
```

```
SQL> --14
SQL>
SQL> update customer_new c
  2  set c.status=(
  3  case
  4  when
  5  (select sum(a.balance)
  6  from account a
  7  where a.branch_name in
  8  (select distinct branch_name
  9  from depositor d
 10  where d.customer_name = c.customer_name)
 11  )>(
 12  select sum(l.amount)
 13  from loan l
 14  where l.branch_name in
 15  (select distinct branch_name
 16  from borrower b
 17  where b.customer_name = c.customer_name)) then 'In savings'
 18  when
 19  (select sum(a.balance)
 20  from account a
 21  where a.branch_name in
 22  (select distinct branch_name
 23  from depositor d
 24  where d.customer_name = c.customer_name)
 25  )<(
 26  select sum(l.amount)
 27  from loan l
 28  where l.branch_name in
 29  (select distinct branch_name
 30  from borrower b
 31  where b.customer_name = c.customer_name))
 32  then 'In loan'
 33  else 'In Breakeven'
 34  end
 35  );

11 rows updated.
```

```sql
--15.Count the occurrences of each status type in
   customer_new table.

select status,count(*) as count
from customer_new
group by status;
```

```
SQL>
SQL> --15
SQL>
SQL> select status,count(*) as count
  2  from customer_new
  3  group by status;

STATUS               COUNT
-------------- ----------
In Breakeven              8
In savings                3
```

3 Challenges

The challenge in this lab was to fully understand the query asked to be executed.While writing the statements it was getting very confusing to write the correct statement.Even if they weren't giving any errors there was no way to know if the statements were actually correct or not since there were too many records to look through manually to verify.