Report on Lab-05

DATABASE MANAGEMENT SYSTEMS LAB

Submitted by

Adid-Al-Mahamud Shazid

Student Id: 210042172

Department: CSE

Programme: SWE

Course Title: CSE 4308

Submitted to

Zannatun Naim Sristy

Lecturer, Department of CSE

September 14, 2023

Introduction

In this lab class, we were given tasks based on advanced data manipulation techniques to solve using SQL command line to understand the basics of data definition and data manipulation. The given .sql file named banking.sql was executed before doing the following tasks.

Task

Execute the `banking.sql` script using command. It creates a set of tables along with values that maintain the following schema:

Here, the boldfaces denote the primary keys and the arcs denote the foreign key relationships.
In this lab, you have to write all SQL statements in an editor first and save them with .sql extension. Then execute the SQL script.
Write SQL statements for the following queries:

1. Find all customer names and their cities who have a loan but not an account.
2. Find all customer names who have an account as well as a loan.
3. Show the count of accounts that were opened in each month along with the month.
4. Find the months between the last acc_opening_date and last loan_date of customer 'Smith'.
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.
6. Find the customer name and account number of the account that has the highest balance.
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.
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.
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'.
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.
11. Create a new customer_new table using a similar structure to the customer table.
12. In the customer_new table insert only those customers who have either an account or a loan.
13. Add a new column Status in customer_new table of varchar2(15) type.
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'.
15. Count the occurrences of each status type in customer_new table.

Solution

```sql
--Task 01--

select customer.customer_name, customer.customer_city
from customer, borrower
where customer.customer_name=borrower.customer_name
minus
select customer.customer_name, customer.customer_city
from customer, depositor
where customer.customer_name=depositor.customer_name;

--Task 02--

select depositor.customer_name
from customer, depositor
where customer.customer_name=depositor.customer_name
intersect
select borrower.customer_name
from borrower, customer
where customer.customer_name=borrower.customer_name;

--Task 03--

select extract (month from acc_opening_date)as months,count(*) as count
from account
group by extract (month from acc_opening_date);

--Task 04--

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

```sql
--Task 05--

select a.branch_name, a.avg_amount from
(select branch_name, avg(amount) as avg_amount from loan group by branch_name) a,
branch where branch.branch_name=a.branch_name and
branch.branch_city not like '%HORSE%';

--Task 06--

select customer_name,account_number from depositor
where account_number in
(
    select account_number from account
    where balance=(select max(balance) from account)
);

--Task 07--

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

--Task 08--

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
--Task 09--

select branch_name || ' Elite' as branch_name from branch
where branch_name in
(
    select branch_name from account
    group by branch_name
    having sum(balance) >
    (
        select avg(sum_balance) + 500 from
        (
            select branch_name, sum(balance) as sum_balance from account
            group by branch_name
        )
    )
)
union
select branch_name || ' Moderate' as branch_name from branch
where branch_name in
(
    select branch_name from account
    group by branch_name
    having sum(balance) between
    (
        select avg(sum_balance) + 500 from
        (
            select branch_name, sum(balance) as sum_balance from account
            group by branch_name
        )
    )
    and
    (
        select avg(sum_balance) - 500 from
        (
            select branch_name, sum(balance) as sum_balance from account
            group by branch_name
        )
    )
)
union
select branch_name || ' Poor' as branch_name from branch
where branch_name in
(
    select branch_name from account
    group by branch_name
```

```
    having sum(balance) <
    (
        select avg(sum_balance) - 500 from
        (
            select branch_name, sum(balance) as sum_balance from account
            group by branch_name
        )
    )
);

--Task 10--

select branch_name, branch_city from branch
where branch_city in
(
    select customer_city from customer
    where customer_city not in
    (
        select customer_city from customer
        where customer_name in
        (
            select customer_name from depositor
        )
        or
        customer_name in
        (
            select customer_name from borrower
        )
    )
)
and branch_name in
(
    select branch_name from loan
    group by branch_name
    having count(*) > 0
)
and branch_name in
(
    select branch_name from account
    group by branch_name
    having count(*) > 0
);
```

```
--Task 11--

create table customer_new as
select * from customer
where customer_name='SPOON';

--Task 12--

insert into customer_new
select * from customer
where customer_name in
(
    select customer_name from depositor
)
or
customer_name in
(
    select customer_name from borrower
);

--Task 13--

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

Analysis and Explanation

From this task I learnt some new functionalities and using sub queries properly.

Difficulties

I faced a few difficulties during several tasks.