# LAB 05

# CSE 4308
# Database Management Systems Lab

**Hasin Mahtab Alvee**
210042174
Department of CSE
B.Sc in Software Engineering

September 14, 2023

# Contents

# Introduction

In DBMS lab 05, we were tasked to manipulate data using different queries and sub-queries. A banking.sql is given that creates all the necessary tables and inserts all the data in them. We only need to write queries to output the tables.

## 1 Task 01

For task 1 we need to find the names, and city of customers that are both Borrowers and Depositors.

```
-- Task 01
SELECT
   CUSTOMER_NAME, CUSTOMER_CITY
FROM
   CUSTOMER
WHERE
   CUSTOMER_NAME IN (
   SELECT
      CUSTOMER_NAME
   FROM
      BORROWER
   )
AND CUSTOMER_NAME NOT IN (
   SELECT
      CUSTOMER_NAME
   FROM
      DEPOSITOR
   );
```

### 1.1 Difficulties

While creating the user, some minor inconvenience were faced, such as -

- It was confusing where to use the AND operator.

## 2 Task 02

In the second task, we need to find all customer names who have an account as well as a loan.

```sql
-- Task 02
SELECT
   CUSTOMER_NAME
FROM
   CUSTOMER
WHERE
   CUSTOMER_NAME IN (
   SELECT
      CUSTOMER_NAME
   FROM
      BORROWER
   )
AND CUSTOMER_NAME IN (
   SELECT
      CUSTOMER_NAME
   FROM
      DEPOSITOR
   );
```

### 2.1 Difficulties

While creating the table structure, some minor inconvenience were faced, such as -

- I faced no difficulty doing this task.

## 3 Task 03

We need to show the count of accounts that were opened in each month along with the month.

```sql
-- Task 03
SELECT
INITCAP(TO_CHAR(TO_DATE(EXTRACT(MONTH FROM ACC_OPENING_DATE), 'MM'),
    'month')) AS MONTH,
COUNT(*)
    AS COUNT
FROM
ACCOUNT
GROUP BY
EXTRACT(MONTH FROM ACC_OPENING_DATE)
ORDER BY
COUNT DESC;
```

### 3.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- I faced no difficulties while doing this task.

## 4 Task 04

We need to find the months between the last account opening date and last loan date of customer 'Smith'.

```
-- Task 04
SELECT
ABS(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;
```

### 4.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- Using the months between method was difficult and to keep it positive.

## 5 Task 05

We need to 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.

```
-- Task 05
SELECT
BRANCH_NAME,
AVG(AMOUNT) AS AVG_LOAN_AMOUNT
FROM
LOAN
WHERE
BRANCH_NAME IN (
SELECT
BRANCH_NAME
FROM
BRANCH
WHERE
BRANCH_CITY NOT LIKE '%a%'
AND BRANCH_NAME NOT LIKE '%Horse%'
)
GROUP BY
BRANCH_NAME
ORDER BY
AVG_LOAN_AMOUNT DESC;
```

### 5.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- Using the substrings to find the result was hard.

## 6 Task 06

We need to find the customer name and account number of the account that has the highest balance.

```sql
-- 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
)
);
```

### 6.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- Finding the Account number from both table was hard.

## 7 Task 07

We need to find 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) from loan, branch

```sql
-- 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;
```

### 7.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- I faced no difficulty doing this task.

## 8 Task 08

We need to 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..

```
-- 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
)
```

```
)
);
```

### 8.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- I faced no difficulties while doing this task.

## 9 Task 09

We need to find the customers while assigning a role to them using Case.

```
-- Task 09
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
BRANCH_NAME,
SUM(BALANCE) AS TOTAL_BALANCE,
AVG(BALANCE) AS AVG_TOTAL_BALANCE
FROM
ACCOUNT
GROUP BY
BRANCH_NAME
);
```

### 9.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- Using the Case syntax was difficult.

## 10 Task 10

We need to find all the branch names and city name where customers are depositors and not borrowers.

```
-- Task 10
SELECT
BRANCH_NAME,
BRANCH_CITY
FROM
BRANCH
WHERE
BRANCH_CITY IN (
SELECT
CUSTOMER_CITY
FROM
CUSTOMER
WHERE
CUSTOMER_NAME NOT IN (
SELECT
CUSTOMER_NAME
FROM
DEPOSITOR
)
AND CUSTOMER_NAME NOT IN (
SELECT
CUSTOMER_NAME
FROM
BORROWER
)
)
AND BRANCH_NAME IN (
SELECT
BRANCH_NAME
FROM
LOAN
)
AND BRANCH_NAME IN (
SELECT
BRANCH_NAME
FROM
ACCOUNT
WHERE
ACCOUNT_NUMBER IN (
```

```
SELECT
ACCOUNT_NUMBER
FROM
DEPOSITOR
)
);
```

### 10.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- Retriving both the branch name and branch city was hard.

## 11 Task 11

We need to create a new customer table using the same structure of the customer table.

```
-- Task 11
CREATE TABLE CUSTOMER_NEW AS
SELECT
*
FROM
CUSTOMER
WHERE
1 = 0;
```

### 11.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- I faced no difficulties while doing this task.

## 12 Task 12

We need to insert data from the old customer table to the new customer table using only the values from the depositor and the borrowers.

```
-- 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
);
```

### 12.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- Figuring out the OR condition was hard.

## 13 Task 13

We need to alter the table to add a new column named Status of Varchar2.

```
-- TASK 13
ALTER TABLE CUSTOMER_NEW ADD STATUS VARCHAR2(15);
```

### 13.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- I faced no difficulties doing this.

## 14 Task 14

We need to set the status values according to cases given.

```
-- TASK 14
UPDATE CUSTOMER_NEW
SET
STATUS = (
SELECT CASE WHEN TOTAL_BALANCE > TOTAL_LOAN THEN 'IN SAVINGS' WHEN
    TOTAL_BALANCE < TOTAL_LOAN THEN 'IN LOAN' ELSE 'IN BREAKEVEN' END
    FROM ( SELECT CUSTOMER_NAME, SUM(BALANCE) AS TOTAL_BALANCE,
    SUM(AMOUNT) AS TOTAL_LOAN FROM ACCOUNT, LOAN WHERE
    ACCOUNT.BRANCH_NAME = LOAN.BRANCH_NAME AND ACCOUNT.ACCOUNT_NUMBER IN
    ( SELECT ACCOUNT_NUMBER FROM DEPOSITOR WHERE CUSTOMER_NAME =
```

```
    CUSTOMER_NEW.CUSTOMER_NAME ) AND LOAN.LOAN_NUMBER IN ( SELECT
    LOAN_NUMBER FROM BORROWER WHERE CUSTOMER_NAME =
    CUSTOMER_NEW.CUSTOMER_NAME ) GROUP BY CUSTOMER_NAME ) WHERE
    CUSTOMER_NAME = CUSTOMER_NEW.CUSTOMER_NAME
);
```

### 14.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- Setting the values according to cases was pretty hard.

## 15 Task 15

We need to Count the number of customers according to their Status.

```
-- TASK 15
SELECT
STATUS,
COUNT(*) AS COUNT
FROM
CUSTOMER_NEW
GROUP BY
STATUS;
```

### 15.1 Difficulties

While inserting the data, some minor inconvenience were faced, such as -

- Separating by status was difficult.

```
SQL> @G:\DOCS\IUT\DBMS1_CSE4307\Lab_05\Tasks.sql

CUSTOMER_NAME   CUSTOMER_CITY
--------------- ---------------
Jackson         Salt Lake
McBride         Rye
Adams           Pittsfield
Curry           Rye


CUSTOMER_NAME
---------------
Jones
Smith
Hayes


MONTH                                    COUNT
---------------------------------- ----------
March                                       2
November                                    1
April                                       1
September                                   1
August                                      1
July                                        1
January                                     1
May                                         1

8 rows selected.
```

Figure 1:

```
     MONTH
----------
3.77419355


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

6 rows selected.


CUSTOMER_NAME   ACCOUNT_NUMBER
--------------- ---------------
Johnson         A-201


BRANCH_CITY     AVG(AMOUNT)
--------------- -----------
Palo Alto              2000
Rye                    4035
```

Figure 2: