Name: Prashant Suresh Shirgave
Roll No: 3 **Batch:** T1
Class: CSE(AIML)

Experiment No. 7

Title: Join operations and set operations.

Objective: To study database joins and set operations.

Theory:

1. **Joins:**

Join types	Join condition
Inner join	Natural
Left outer join	On <predicate>
Right outer join	Using (A1, …. , An)
Full outer join	

- JOINS allow us to combine data from more than one table into a single result set.
- JOINS have better performance compared to sub queries
- INNER JOINS only return rows that meet the given criteria.
- OUTER JOINS can also return rows where no matches have been found. The unmatched rows are returned with the NULL keyword.
- The frequently used clause in JOIN operations is "ON". "USING" clause requires that matching columns be of the same name.
- JOINS can also be used in other clauses such as GROUP BY, WHERE, SUB QUERIES, AGGREGATE FUNCTIONS etc.

Inner Join:

```
SELECT columns
FROM tableA
INNER JOIN tableB
ON tableA.column = tableB.column;
```

Natural join

SELECT columns

FROM tableA

NATURAL JOIN tableB

2. Set operations: union, intersect, minus

Syntax:

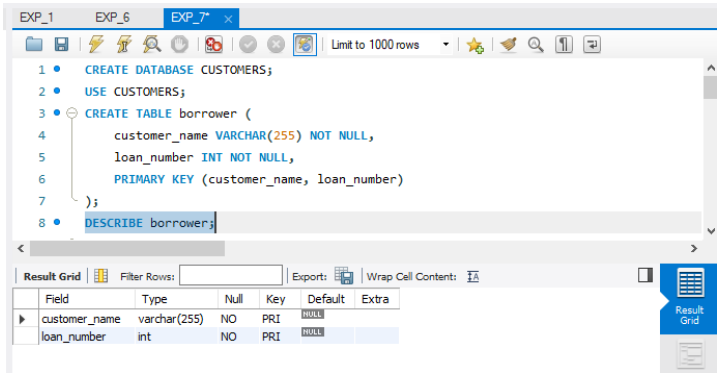
select query1

set operator

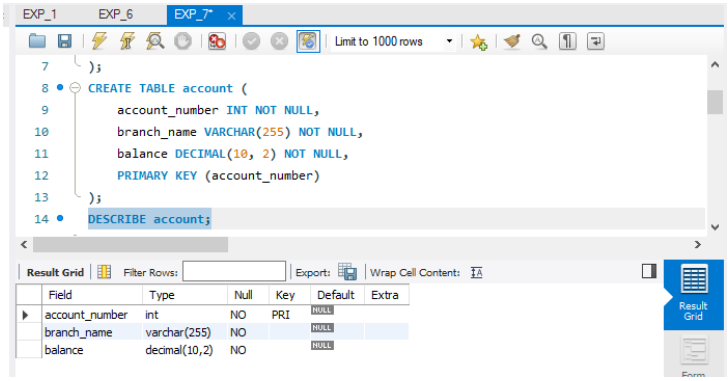
select query2;

- **Consider the following schema.**

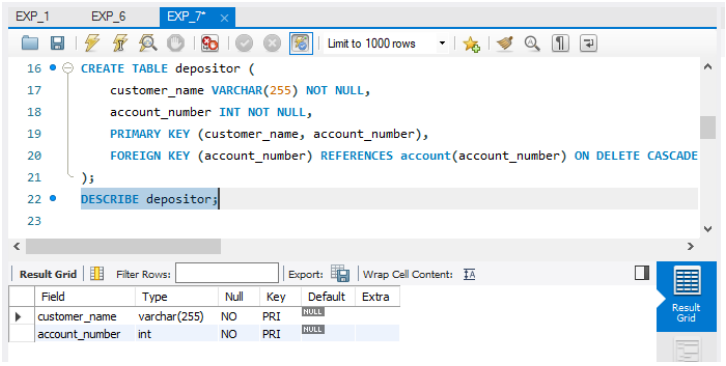
borrower (customer-name, loan-number)



account (<u>account-number</u>, branch-name, balance)



depositor (customer-name, account-number)

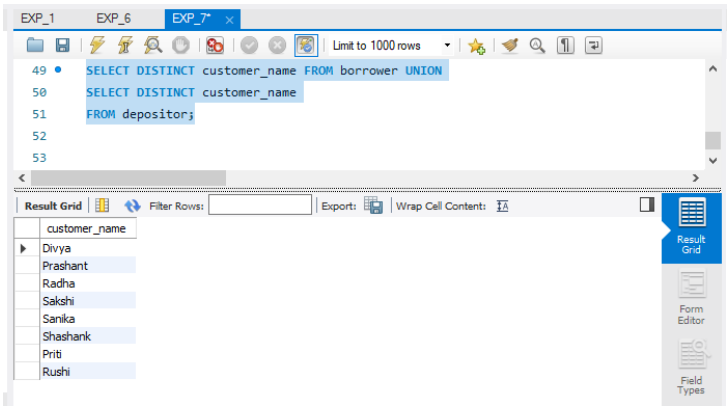


- **Execute the following queries:**

1. **Find all customers having either loan, an account or both at the bank.**

   SELECT DISTINCT customer_name FROM borrower UNION

   SELECT DISTINCT customer_name
   FROM depositor;

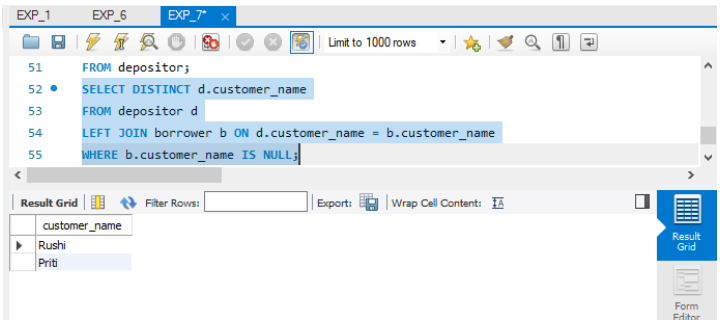


2. **Find all customers having an account but not loan at the bank.**

   SELECT DISTINCT d.customer_name FROM depositor d

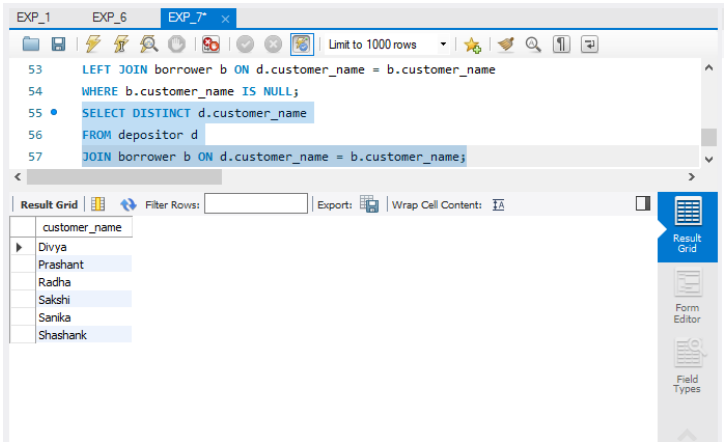
   LEFT JOIN borrower b ON d.customer_name = b.customer_name

   WHERE b.customer_name IS NULL;



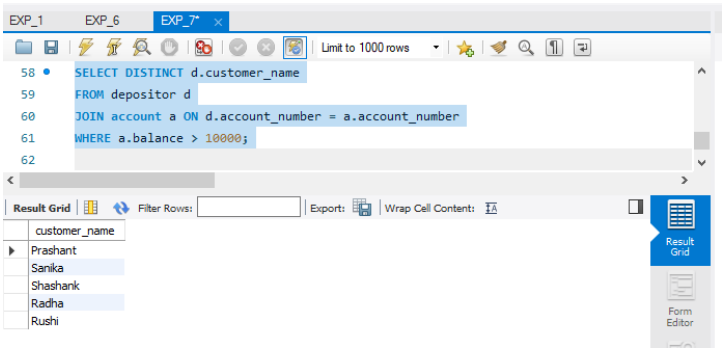
3. **Find all customers having both an account and loan at the bank.**

```
SELECT DISTINCT d.customer_name
FROM depositor d
JOIN borrower b ON d.customer_name = b.customer_name;
```



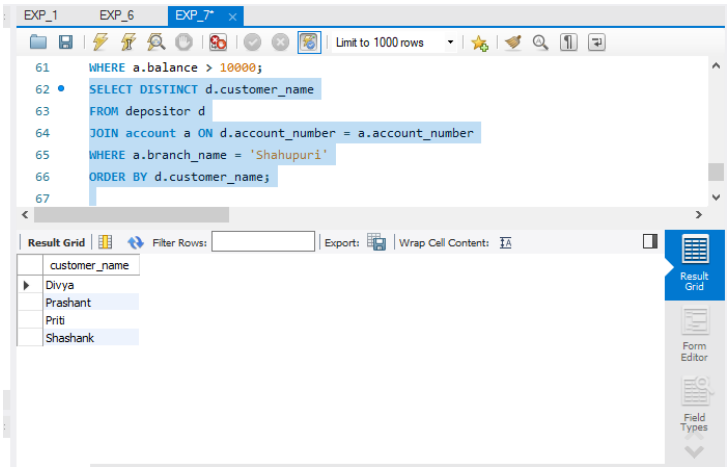
4. **Find the customers who have balance more than 10000.**

```
SELECT DISTINCT d.customer_name FROM depositor d
JOIN account a ON d.account_number = a.account_number
WHERE a.balance > 10000;
```



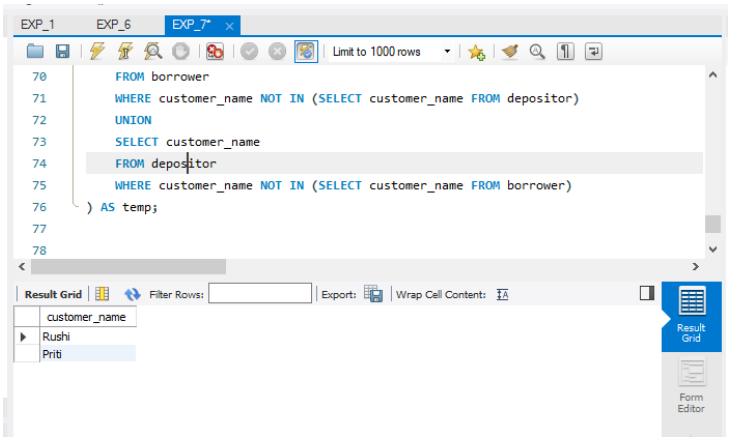
5. **List in alphabetic order, customers who have account at ‘Shahupuri’ branch.**

```
SELECT DISTINCT d.customer_name
FROM depositor d
JOIN account a ON d.account_number = a.account_number
WHERE a.branch_name = 'Shahupuri'
ORDER BY d.customer_name;
```



6. Find all customers who have either an account or loan (but not both) at the bank.

```
 SELECT customer_name
FROM (
    SELECT customer_name
    FROM borrower
    WHERE customer_name NOT IN (SELECT customer_name FROM depositor)
    UNION
    SELECT customer_name
    FROM depositor
    WHERE customer_name NOT IN (SELECT customer_name FROM borrower)
) AS temp;
```



Outcome: Students are able to perform joins and set operations on various relational tables.