

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

Experiment No. 8

Title: Views, Constraints and Subqueries.

Objective: To study the implementation of view and different operations on it. Also to study Constraints and Subqueries.

Theory:

View: Syntax: create view viewname

As

< query expression >

e.g. create view marks60 as select rollno, marks from student where marks >60;

Constraints: primary key, foreign key, not null, check, unique clause, on delete cascade.

Creating table using all constraints:

Syntax: create table tablename

(A1D1, , AnDn, <integrity_constraint1>,,,,,,,,,,,,, <integrity_constraint n>);

Where A1,....., An are attributes, D1,..... , Dn are datatypes.

Sub-queries:

a. Single row subquery:

Syntax: select columnlist from tablename where columnname operator

(select columnlist from tablename [where condition]) ;

b. Multiple row subquery:

Operators: In, All, any/some

Examples of constraints:

//Primary Key

create table department

(dept_name varchar(20) primary key, building varchar(20), budget numeric(15,2));

insert into department values ('cse', 'new' , 200000);

insert into department values ('etx', 'new' , 250000)

//Foreign key

```
create table instructor
(ID char(5), name varchar(20) not null,
 dept_name varchar(20),
 salary numeric(8,2),
 primary key (ID),
 foreign key (dept_name) references department(dept_name) on delete cascade on update
 cascade)
```

```
insert into instructor values ('10211', 'Ram', 'cse', 66000);
insert into instructor values ('10212', 'Shourya', 'etx', 60000);
```

```
select * from department;
select * from instructor;
```

//To see effect of on delete cascade and on update cascade

```
update department set dept_name='etc' where dept_name='etx';
select * from department;

select * from instructor;
delete from department where dept_name='etc';
```

```
select * from department;
select * from instructor;
```

```
create table student (
    ID varchar(5), name varchar(20) not null, dept_name varchar(20),
    tot_cred numeric(3,0) DEFAULT 5.5, primary key (ID),
    foreign key (dept_name) references department (dept_name));
```

//Setting Default value to column

```
mysql> ALTER TABLE table_name
    ALTER column_name SET DEFAULT default_value;
```

```
mysql> ALTER TABLE table_name
      ALTER column_name DROP DEFAULT;
```

```
mysql> create table s1 (rn int, class char(4) default 'TY', marks float(4,2));
Query OK, 0 rows affected, 1 warning (1.26 sec)
```

```
mysql> insert into s1(rn, marks) values (1,78);
Query OK, 1 row affected (0.11 sec)
```

```
mysql> select * from s1;
+-----+-----+-----+
| rn | class | marks |
+-----+-----+-----+
|  1 | TY   | 78.00 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

//Check Constraint

```
create table account1 (accno int, brnm varchar(10), balance numeric(6,1),
check(balance>20000));
```

```
insert into account1 values (11,"shahupuri", 6000);
ERROR 3819 (HY000): Check constraint 'account1_chk_1' is violated.
```

● Queries to Solve:

Consider the following schema.

```
loan (loan_number, branch_name, amount)
borrower (customer_name, loan_number)
account (account_number, branch_name, balance )
depositor (customer_name, account_number)
```

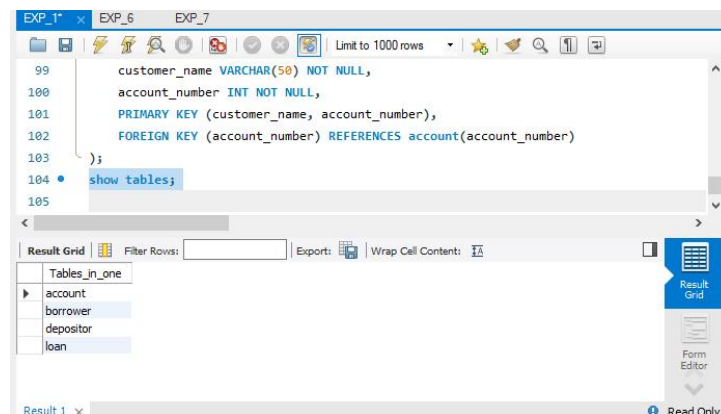
```
CREATE TABLE loan (
loan_number INT PRIMARY KEY,
branch_name VARCHAR(50) NOT NULL,
```

```
amount DECIMAL(10, 2) CHECK (amount >= 0)
);
```

```
CREATE TABLE borrower (
    customer_name VARCHAR(50) NOT NULL,
    loan_number INT NOT NULL,
    PRIMARY KEY (customer_name, loan_number),
    FOREIGN KEY (loan_number) REFERENCES
loan(loan_number)
);
```

```
CREATE TABLE account (
    account_number INT PRIMARY KEY,
    branch_name VARCHAR(50) NOT NULL,
    balance DECIMAL(10, 2) CHECK (balance >= 0)
);
```

```
CREATE TABLE depositor (
    customer_name VARCHAR(50) NOT NULL,
    account_number INT NOT NULL,
    PRIMARY KEY (customer_name, account_number),
    FOREIGN KEY (account_number) REFERENCES
account(account_number)
);
```



- **Execute the following queries:**

1. Create a view 'all_customers' consisting of branch names and the names of customers who have either an account or a loan or both.

```
CREATE VIEW all_customers AS

SELECT DISTINCT branch_name, customer_name

FROM borrower

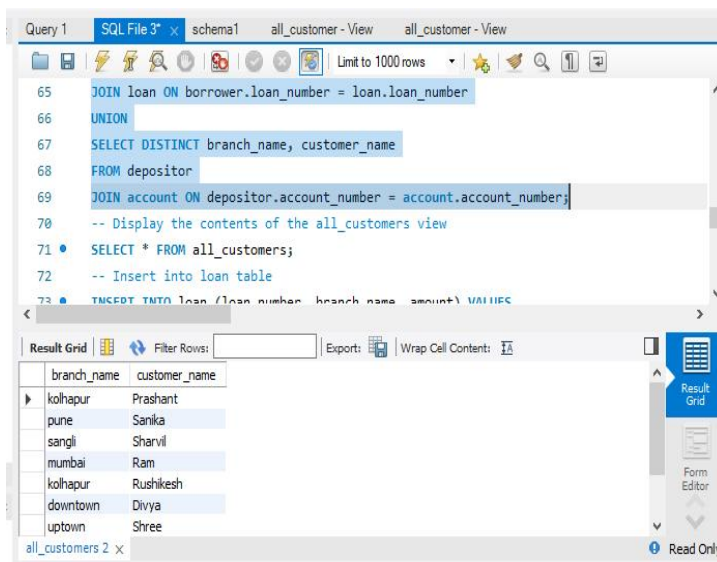
JOIN loan ON borrower.loan_number = loan.loan_number

UNION

SELECT DISTINCT branch_name, customer_name

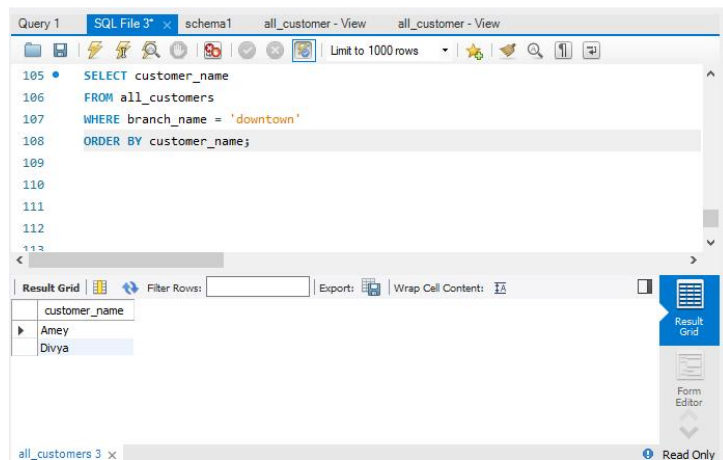
FROM depositor

JOIN account ON depositor.account_number = account.account_number;
```



2. Using view 'all_customers', list in alphabetic order, the customers of 'downtown' branch.

```
SELECT customer_name
FROM all_customers
WHERE branch_name = 'downtown'
ORDER BY customer_name;
```



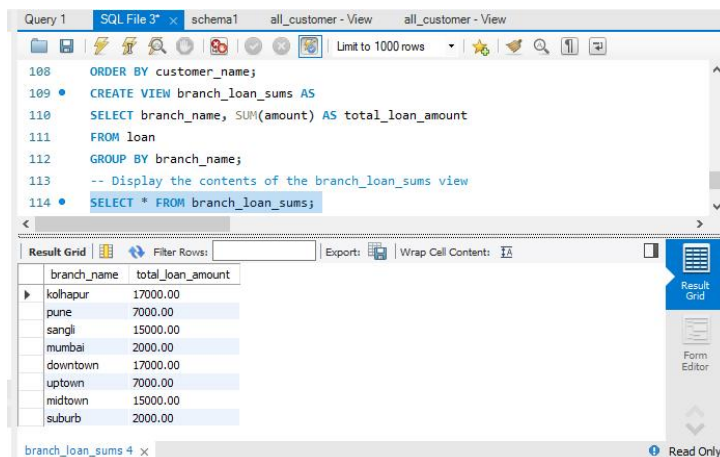
3. Create a view consisting of sum of the amounts of all the loans for each branch.

CREATE VIEW branch_loan_sums AS

SELECT branch_name, SUM(amount) AS total_loan_amount

FROM loan

GROUP BY branch_name;



● Consider the following schema.

branch (branch-name, branch-city, assets)

borrower (customer-name, loan-number)

account (account-number, branch-name, balance)

depositor (customer-name, account-number)

- **Execute the following queries:**

1. **Create above tables using all constraints- primary key, foreign key, not null, check.**

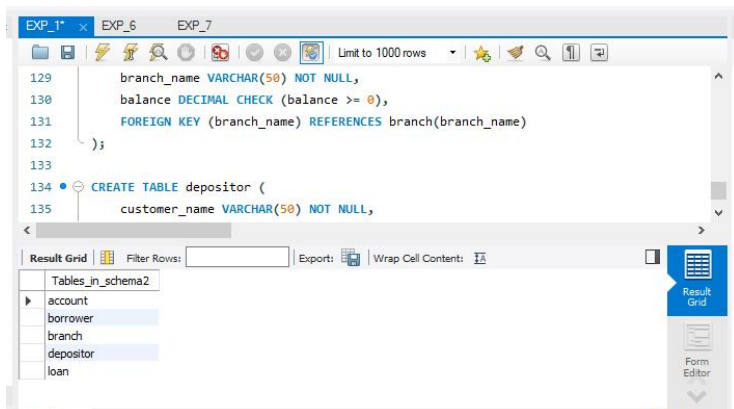
```
CREATE TABLE branch (  
    branch_name VARCHAR(50) PRIMARY KEY,  
    branch_city VARCHAR(50) NOT NULL,  
    assets DECIMAL CHECK (assets >= 0)  
);
```

```
CREATE TABLE loan (  
    loan_number INT PRIMARY KEY,  
    branch_name VARCHAR(50) NOT NULL,  
    amount DECIMAL CHECK (amount >= 0),  
    FOREIGN KEY (branch_name) REFERENCES branch(branch_name)  
);
```

```
CREATE TABLE borrower (  
    customer_name VARCHAR(50) NOT NULL,  
    loan_number INT NOT NULL,  
    PRIMARY KEY (customer_name, loan_number),  
    FOREIGN KEY (loan_number) REFERENCES loan(loan_number)  
);
```

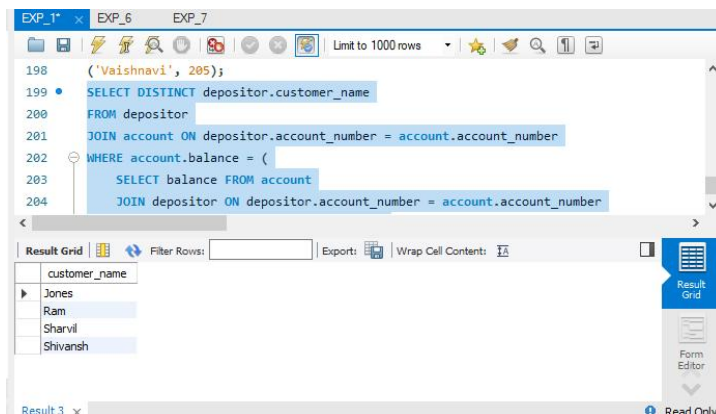
```
CREATE TABLE account (  
    account_number INT PRIMARY KEY,  
    branch_name VARCHAR(50) NOT NULL,  
    balance DECIMAL CHECK (balance >= 0),  
    FOREIGN KEY (branch_name) REFERENCES branch(branch_name)  
);
```

```
CREATE TABLE depositor (  
    customer_name VARCHAR(50) NOT NULL,  
    account_number INT NOT NULL,  
    PRIMARY KEY (customer_name, account_number),  
    FOREIGN KEY (account_number) REFERENCES account(account_number)  
);
```



2. Find the customers having same balance as 'Ram'

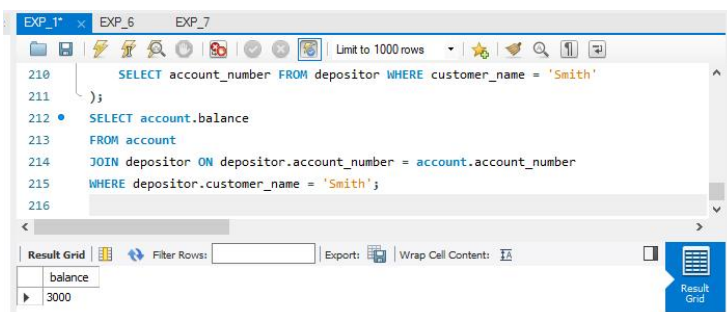
```
SELECT DISTINCT depositor.customer_name
FROM depositor
JOIN account ON depositor.account_number = account.account_number
WHERE account.balance = (
    SELECT balance FROM account
    JOIN depositor ON depositor.account_number = account.account_number
    WHERE depositor.customer_name = 'Ram'
);
```



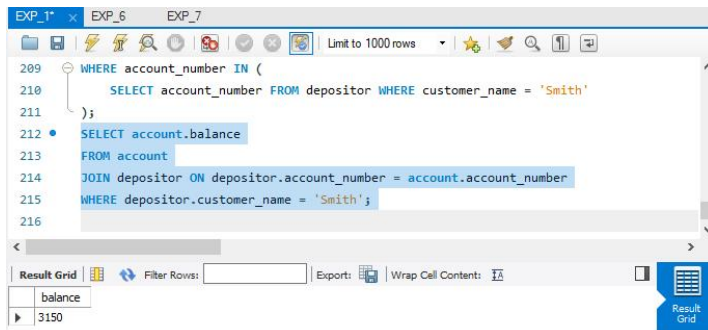
3. Update balance of 'Smith' by 5%.

```
UPDATE account
SET balance = balance * 1.05
WHERE account_number IN (
    SELECT account_number FROM depositor WHERE customer_name = 'Smith'
);
```

Before:



After:

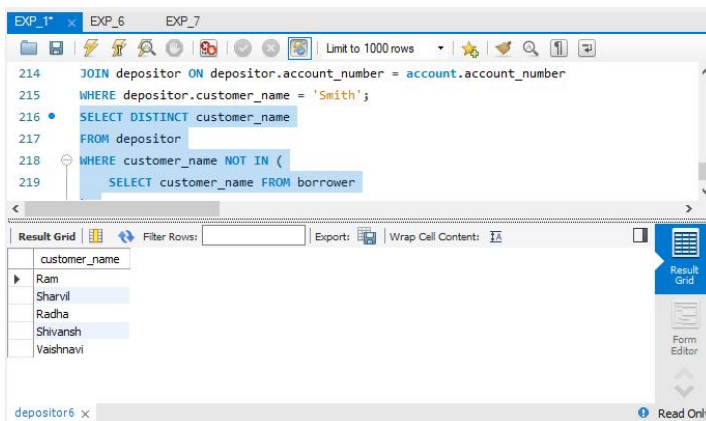


4. By using subquery, find the names of customers who have account but not loan.

```

SELECT DISTINCT customer_name
FROM depositor
WHERE customer_name NOT IN (
    SELECT customer_name FROM borrower
);

```

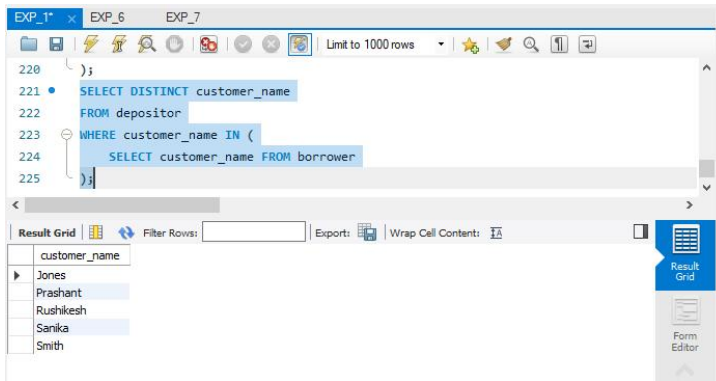


5. By using subquery, find the names of customers who have both account and loan.

```

SELECT DISTINCT customer_name
FROM depositor
WHERE customer_name IN (
    SELECT customer_name FROM borrower
);

```

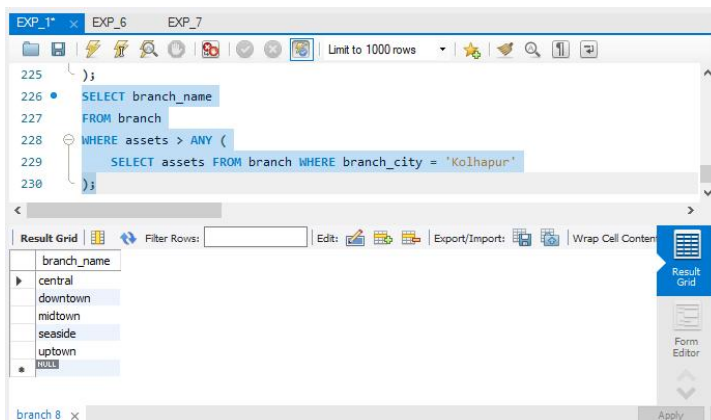


6. Find the names of all branches that have assets greater than those of at least one branch located in 'Kolhapur'.

```

SELECT branch_name
FROM branch
WHERE assets > ANY (
    SELECT assets FROM branch WHERE branch_city = 'Kolhapur'
);

```



7. Update those accounts whose balance is greater than average balance by 5%.

```

UPDATE account
SET balance = balance * 1.05
WHERE balance > (
    SELECT avg_balance FROM (SELECT AVG(balance) AS avg_balance FROM account) AS temp
);

```

EXP_1* EXP_6 EXP_7

Limit to 1000 rows

```

231 -- Use a subquery to calculate the average balance separately
232 • UPDATE account
233 SET balance = balance * 1.05
234 WHERE balance > (
235     SELECT avg_balance FROM (SELECT AVG(balance) AS avg_balance FROM account) AS temp
236 );

```

Result Grid

account_number	branch_name	balance
100	downtown	8400
101	uptown	10500
102	midtown	5000
103	suburb	3150
104	seaside	15750
201	central	5000
202	central	5000
203	central	7000
204	central	5000
205	central	9450

account 9 x Apply Revert

8. Find the customers who have a loan at bank & whose names are neither 'Smith' nor 'Jones'.

```

SELECT DISTINCT customer_name
FROM borrower
WHERE customer_name NOT IN ('Smith','Jones');

```

EXP_1* EXP_6 EXP_7

Limit to 1000 rows

```

235 SELECT avg_balance FROM (SELECT AVG(balance) AS avg_balance FROM account) AS temp
236 );
237 • SELECT DISTINCT customer_name
238 FROM borrower
239 WHERE customer_name NOT IN ('Smith', 'Jones');
240

```

Result Grid

customer_name
Prashant
Sanika
Rushikesh

borrower 10 Read Only

9. Delete all account tuples at every branch located in 'Kolhapur' city.

```

DELETE FROM depositor
WHERE account_number IN (
    SELECT account_number FROM account
    WHERE branch_name IN (
        SELECT branch_name FROM branch WHERE branch_city = 'Kolhapur'
    )
);
DELETE FROM account
WHERE branch_name IN (
    SELECT branch_name FROM branch WHERE branch_city = 'Kolhapur'
);

```

Before Deleting:

The screenshot shows a database management tool interface. The top pane displays a SQL query with line numbers 239 to 244. The query is as follows:

```
239 WHERE customer_name NOT IN ('Smith', 'Jones');  
240 • SELECT * FROM account  
241 WHERE branch_name IN (  
242     SELECT branch_name FROM branch WHERE branch_city = 'Kolhapur'  
243 );  
244
```

The bottom pane shows the 'Result Grid' with columns 'account_number', 'branch_name', and 'balance'. It contains three rows of data:

account_number	branch_name	balance
102	midtown	5000
103	suburb	3150
NULL	NULL	NULL

The interface also includes a toolbar with various icons and a sidebar on the right with options like 'Result Grid', 'Form Editor', and 'Field Types'.

After Deleting:

The screenshot shows the same database management tool interface after a deletion. The top pane displays a SQL query with line numbers 277 to 282. The query is as follows:

```
277     SELECT branch_name FROM branch WHERE branch_city = 'Kolhapur'  
278 );  
279 • SELECT * FROM account  
280 WHERE branch_name IN (  
281     SELECT branch_name FROM branch WHERE branch_city = 'Kolhapur'  
282 );
```

The bottom pane shows the 'Result Grid' with columns 'account_number', 'branch_name', and 'balance'. It now contains only one row of data:

account_number	branch_name	balance
NULL	NULL	NULL

The interface elements are consistent with the previous screenshot, showing the same toolbar and sidebar.

Outcome: Students are able to create views. Also Students are able to create the tables using primary key, foreign key, not null, check constraint.

