**EX.NO: 1** **DDL and DML commands**

**DDL (DATA DEFINITION LANGUAGE)**

- CREATE
- ALTER
- DROP
- TRUNCATE
- COMMENT
- RENAME

```
SQL> CREATE TABLE EMP (EMPNO NUMBER (4), ENAME VARCHAR2 (10),
       DESIGNATIN VARCHAR2 (10), SALARY NUMBER (8,2));
```

**Table created.**

```
SQL: DESC <TABLE NAME>;
SQL> DESC EMP;

Name                                    Null?           Type
-----------------------------------------------         ------------------------
EMPNO                                                   NUMBER(4)
ENAME                                                   VARCHAR2(10)
DESIGNATIN                                              VARCHAR2(10)
SALARY                                                  NUMBER(8,2)
```

```
SQL>ALTER TABLE EMP MODIFY EMPNO NUMBER (6);
```

**Table altered.**

```
SQL> DESC EMP;
Name                                    Null?  Type
---------------------------------------- --------- -------------------------------
EMPNO                                          NUMBER(6)
ENAME                                          VARCHAR2(10)
DESIGNATIN                                     VARCHAR2(10)
SALARY                                         NUMBER(8,2)
SQL>ALTER TABLE EMP ADD (DOB DATE, DOJ DATE);
Table altered.
SQL> DESC EMP;

Name                                    Null?       Type
---------------------------------------- --------- -------------------------------
EMPNO                                          NUMBER (7)
ENAME                                          VARCHAR 2(12)
DESIGNATIN                                     VARCHAR 2(10)
SALARY                                         NUMBER (8,2)
QUALIFICATION                                  VARCHAR 2(6)
DOB                                            DATE
DOJ                                            DATE
```

**REMOVE / DROP**

```
SQL> ALTER TABLE EMP DROP COLUMN DOJ;
SQL> DESC EMP;
Name Null?                                     Type
---------------------------------------- --------- ---------------
EMPNO                                          NUMBER (7)
ENAME                                          VARCHAR 2(12)
DESIGNATIN                                     VARCHAR 2(10)
```

```
SALARY                                  NUMBER (8,2)
    QUALIFICATION                       VARCHAR 2(6)
    DOB                                 DATE

SQL>ALTER TABLE EMP DROP (DOB, QUALIFICATION);
Table altered.
SQL> DESC EMP;
Name Null?                              Type
--------------------------------------------- --------- ----------------
EMPNO                                   NUMBER (7)
ENAME                                   VARCHAR 2(12)
DESIGNATIN                              VARCHAR 2(10)
SALARY                                  NUMBER (8,2)
```

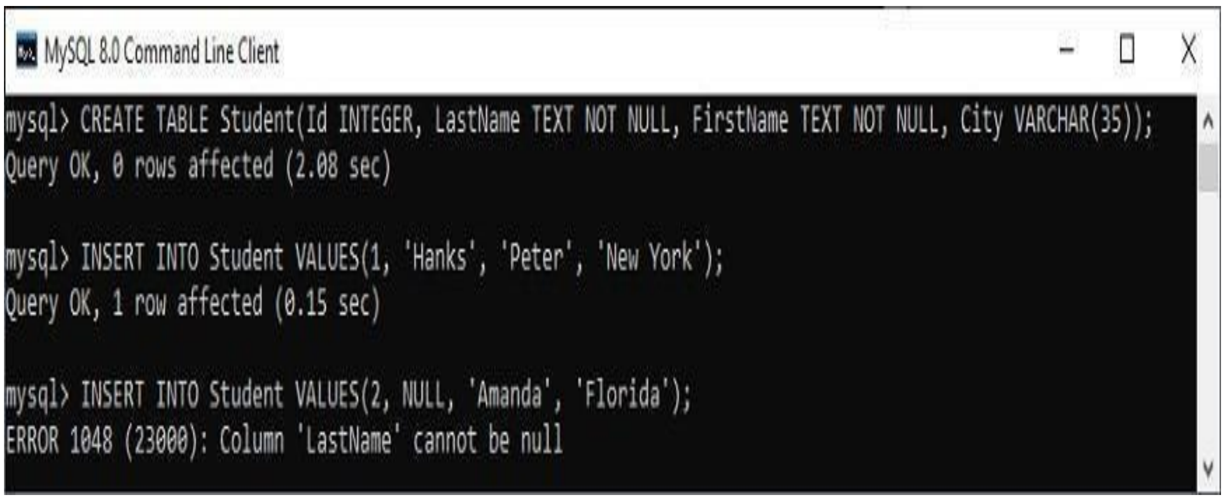
**NOT NULL Constraint**

MySQL> CREATE TABLE Student (Id INTEGER, Last Name TEXT NOT NULL, FirstName TEXT NOT NULL, City VARCHAR (35));

MySQL> INSERT INTO Student VALUES(1, 'Hanks', 'Peter', 'New York');

MySQL> INSERT INTO Student VALUES(2, NULL, 'Amanda', 'Florida');

**Output**



```
mysql> CREATE TABLE Student(Id INTEGER, LastName TEXT NOT NULL, FirstName TEXT NOT NULL, City VARCHAR(35));
Query OK, 0 rows affected (2.08 sec)

mysql> INSERT INTO Student VALUES(1, 'Hanks', 'Peter', 'New York');
Query OK, 1 row affected (0.15 sec)

mysql> INSERT INTO Student VALUES(2, NULL, 'Amanda', 'Florida');
ERROR 1048 (23000): Column 'LastName' cannot be null
```

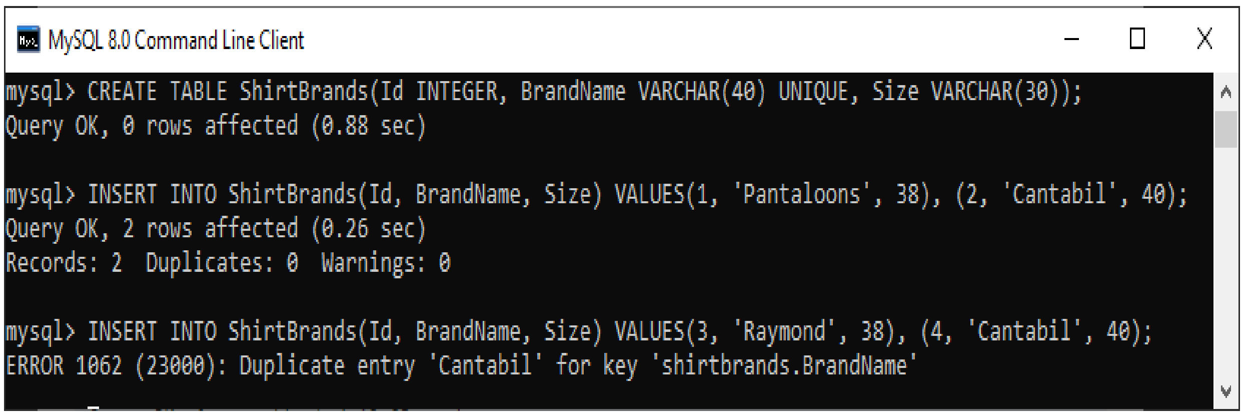
**UNIQUE Constraint**

MySQL> CREATE TABLE ShirtBrands(Id INTEGER, BrandName VARCHAR(40) UNIQUE, Size VARCHAR(30));

MySQL> INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES(1, 'Pantaloons', 38), (2, 'Cantabil', 40);

MySQL> INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES(1, 'Raymond', 38), (2, 'Cantabil', 40);

**Output**



MySQL 8.0 Command Line Client

```
mysql> CREATE TABLE ShirtBrands(Id INTEGER, BrandName VARCHAR(40) UNIQUE, Size VARCHAR(30));
Query OK, 0 rows affected (0.88 sec)

mysql> INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES(1, 'Pantaloons', 38), (2, 'Cantabil', 40);
Query OK, 2 rows affected (0.26 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO ShirtBrands(Id, BrandName, Size) VALUES(3, 'Raymond', 38), (4, 'Cantabil', 40);
ERROR 1062 (23000): Duplicate entry 'Cantabil' for key 'shirtbrands.BrandName'
```

**CHECK CONSTRAINT**

CHECK (expr)

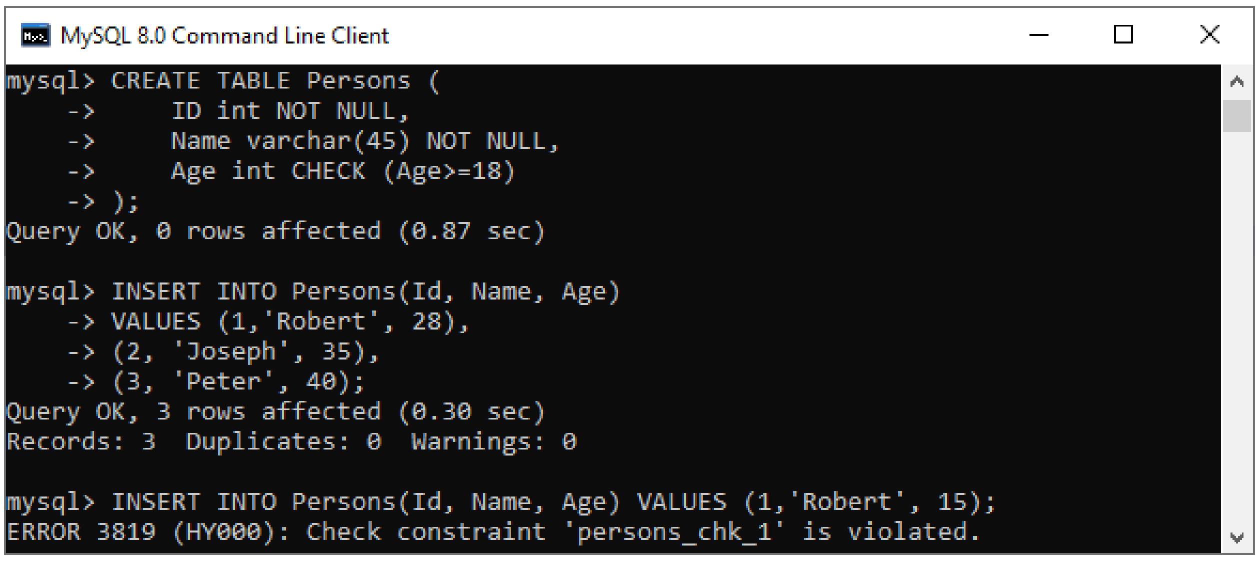
MySQL> CREATE TABLE Persons ( ID int NOT NULL,Name varchar(45) NOT NULL, Age int CHECK (Age>=18) );

MySQL> INSERT INTO Persons(Id, Name, Age) VALUES (1,'Robert', 28), (2, 'Joseph', 35), (3, 'Peter', 40);

MySQL> INSERT INTO Persons(Id, Name, Age) VALUES (1,'Robert', 15);

**Output**

In the below output, we can see that the first INSERT query executes successfully, but the second statement fails and gives an error that says: CHECK constraint is violated for key Age.



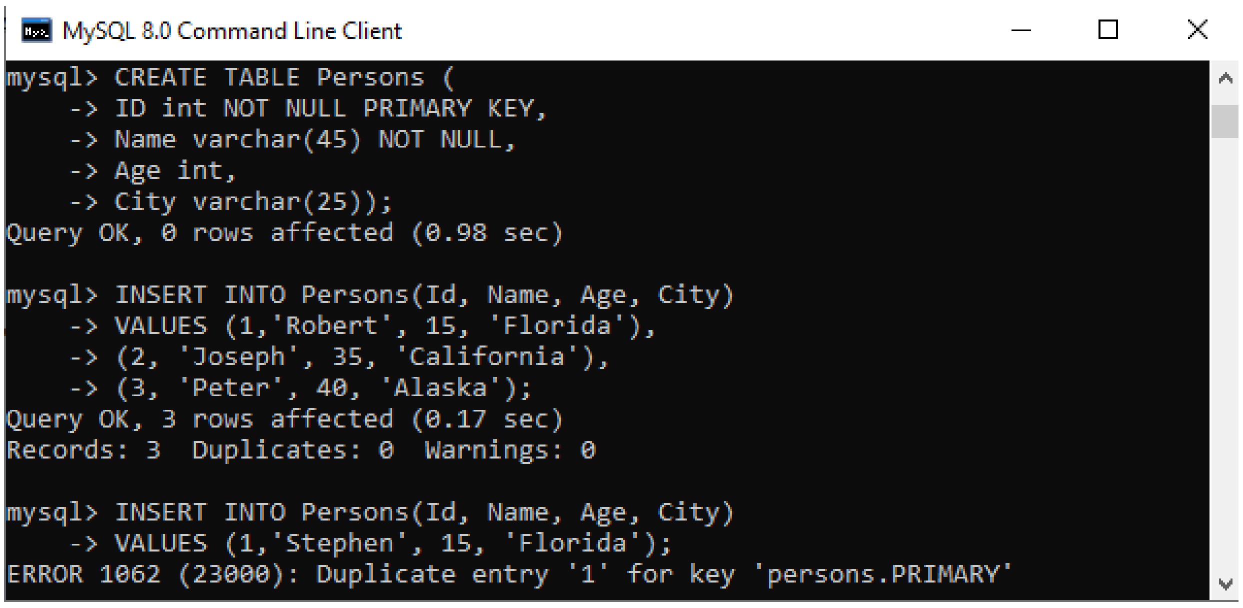
MySQL 8.0 Command Line Client

```
mysql> CREATE TABLE Persons (
    ->     ID int NOT NULL,
    ->     Name varchar(45) NOT NULL,
    ->     Age int CHECK (Age>=18)
    -> );
Query OK, 0 rows affected (0.87 sec)

mysql> INSERT INTO Persons(Id, Name, Age)
    -> VALUES (1,'Robert', 28),
    -> (2, 'Joseph', 35),
    -> (3, 'Peter', 40);
Query OK, 3 rows affected (0.30 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Persons(Id, Name, Age) VALUES (1,'Robert', 15);
ERROR 3819 (HY000): Check constraint 'persons_chk_1' is violated.
```

**PRIMARY KEY CONSTRAINT**

**CREATE TABLE** Persons ( ID **int** NOT NULL **PRIMARY KEY**, **Name varchar**(45) NO T NULL, Age **int**, City **varchar**(25));
**INSERT INTO** Persons(Id, **Name**, Age, City) **VALUES** (1,'Robert', 15, 'Florida') , (2, 'Joseph', 35, 'California'), (3, 'Peter', 40, 'Alaska');
**INSERT INTO** Persons(Id, **Name**, Age, City) **VALUES** (1,'Stephen', 15, 'Florida');

**Output**



MySQL 8.0 Command Line Client

```
mysql> CREATE TABLE Persons (
    -> ID int NOT NULL PRIMARY KEY,
    -> Name varchar(45) NOT NULL,
    -> Age int,
    -> City varchar(25));
Query OK, 0 rows affected (0.98 sec)

mysql> INSERT INTO Persons(Id, Name, Age, City)
    -> VALUES (1,'Robert', 15, 'Florida'),
    -> (2, 'Joseph', 35, 'California'),
    -> (3, 'Peter', 40, 'Alaska');
Query OK, 3 rows affected (0.17 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Persons(Id, Name, Age, City)
    -> VALUES (1,'Stephen', 15, 'Florida');
ERROR 1062 (23000): Duplicate entry '1' for key 'persons.PRIMARY'
```

## EX.NO:2 FOREIGN KEY AND REFERENTIAL INTEGRITY CONSTRAINT

DEPARTMENT

```
CREATE TABLE Department(
Id INT PRIMARY KEY,
Name NVARCHAR(50)
);
-- Insert some test data in Department Table
Insert into Department values (10, 'IT');
Insert into Department values (20, 'HR');
Insert into Department values (30, 'INFRA');
```

EMPLOYEES

```
CREATE TABLE Employees(
Id INT PRIMARY KEY,
Name VARCHAR(100) NOT NULL,
DepartmentID INT
);

-- Adding the Foreign Key Constraint
ALTER TABLE Employees ADD FOREIGN KEY (DepartmentId) REFERENCES
Department(Id);

-- Insert some test data in Employees Table
INSERT into Employees VALUES (101, 'Anurag', 10);
INSERT into Employees VALUES (102, 'Pranaya', 20);
INSERT into Employees VALUES (103, 'Hina', 30);
```

Delete from Parent Table

**DELETE FROM Department WHERE Id = 10;**

**OUTPUT**

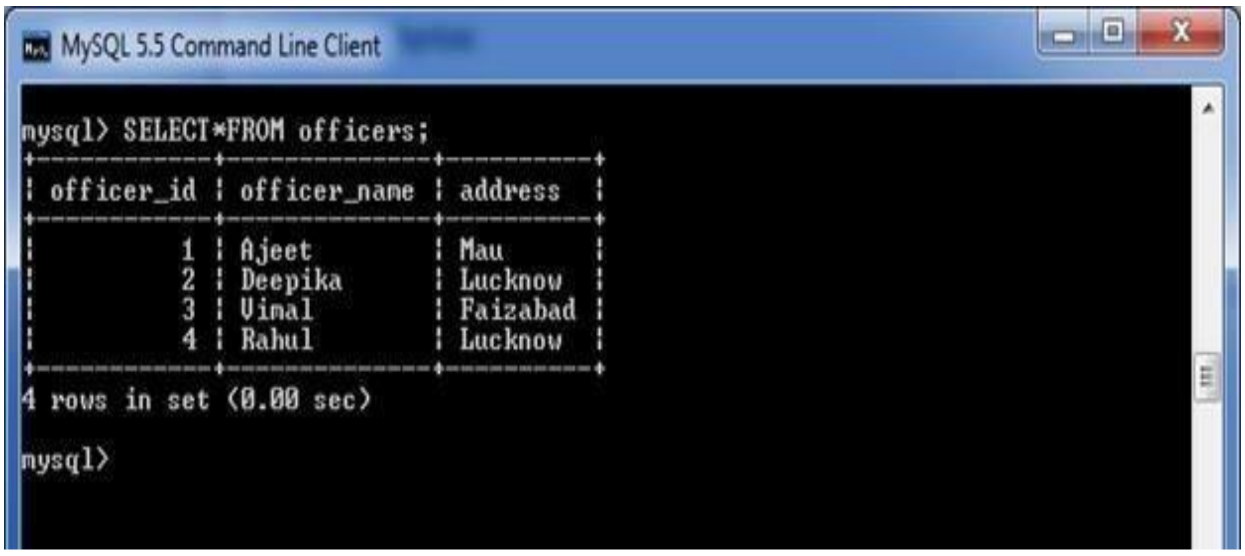
```
SQL> DELETE from Department where Id=10;

ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constrai

nt fails (`db2`.`employees`, CONSTRAINT `employees_ibfk_1` FOREIGN KEY (`Departm

entID`) REFERENCES `department` (`Id`))
```

## EX.NO: 3 QUERIES WITH WHERE CLAUSE AND AGGREATE FUNCTIONS.

**MySQL WHERE Clause**

**Syntax:**

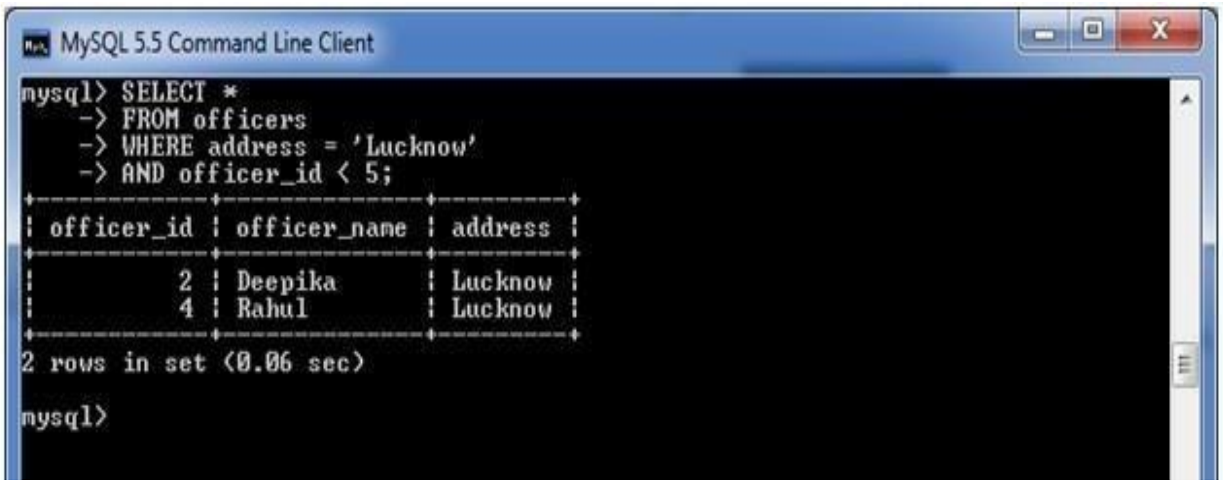
**Select * from Tablename WHERE** conditions;



```
mysql> SELECT*FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql>
```

**MySQL WHERE Clause with AND condition**

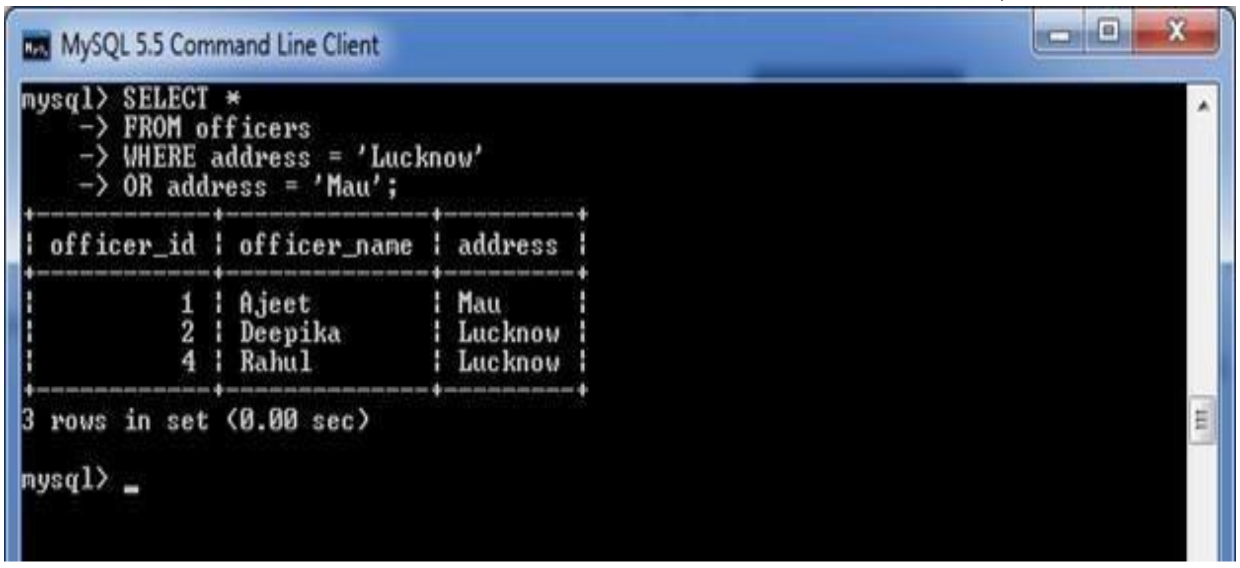
SELECT * FROM officers WHERE address = 'Lucknow' AND officer_id < 5;



```
mysql> SELECT *
    -> FROM officers
    -> WHERE address = 'Lucknow'
    -> AND officer_id < 5;
+------------+--------------+---------+
| officer_id | officer_name | address |
+------------+--------------+---------+
|          2 | Deepika      | Lucknow |
|          4 | Rahul        | Lucknow |
+------------+--------------+---------+
2 rows in set (0.06 sec)

mysql>
```

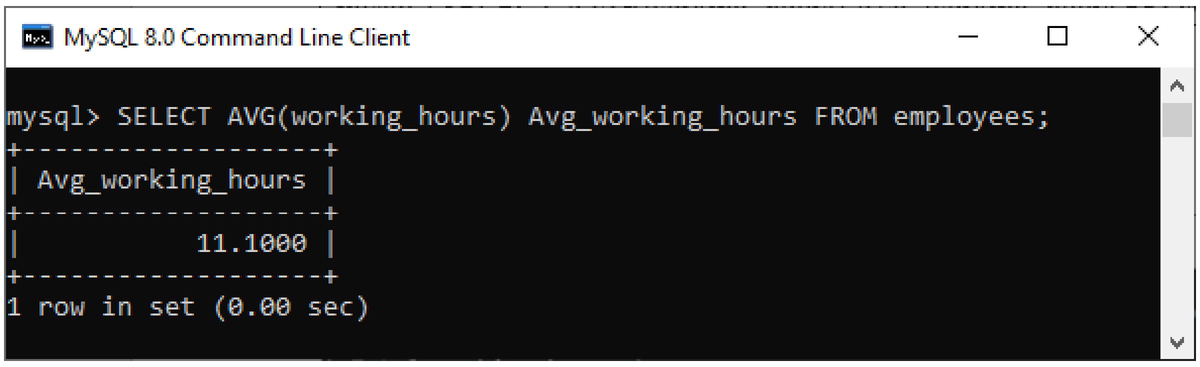
**WHERE Clause with OR condition**

SELECT * FROM officers WHERE address = 'Lucknow' OR address = 'Mau';



```
mysql> SELECT *
    -> FROM officers
    -> WHERE address = 'Lucknow'
    -> OR address = 'Mau';
+------------+--------------+---------+
| officer_id | officer_name | address |
+------------+--------------+---------+
|          1 | Ajeet        | Mau     |
|          2 | Deepika      | Lucknow |
|          4 | Rahul        | Lucknow |
+------------+--------------+---------+
3 rows in set (0.00 sec)

mysql> _
```

**MySQL WHERE Clause with combination of AND & OR conditions**



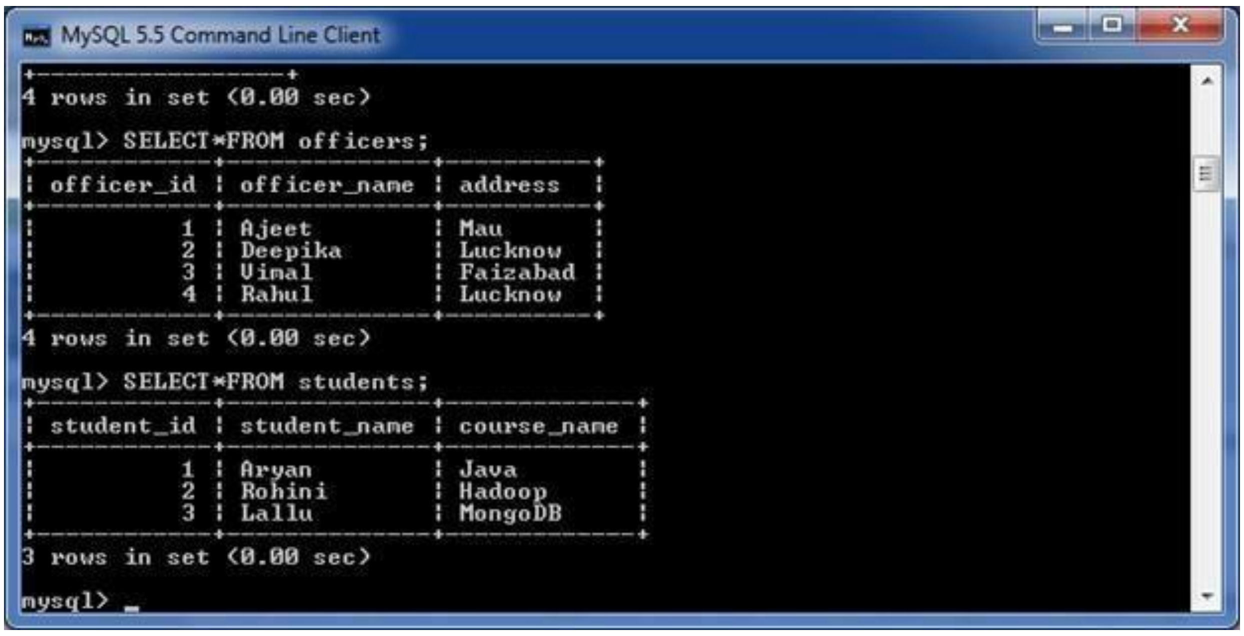
MySQL 8.0 Command Line Client

```
mysql> SELECT AVG(working_hours) Avg_working_hours FROM employees;
+-------------------+
| Avg_working_hours |
+-------------------+
|           11.1000 |
+-------------------+
1 row in set (0.00 sec)
```

EX.NO:4 SIMPLE JOIN AND SUB QUERIES

**MYSQL INNER JOIN (SIMPLE JOIN)**

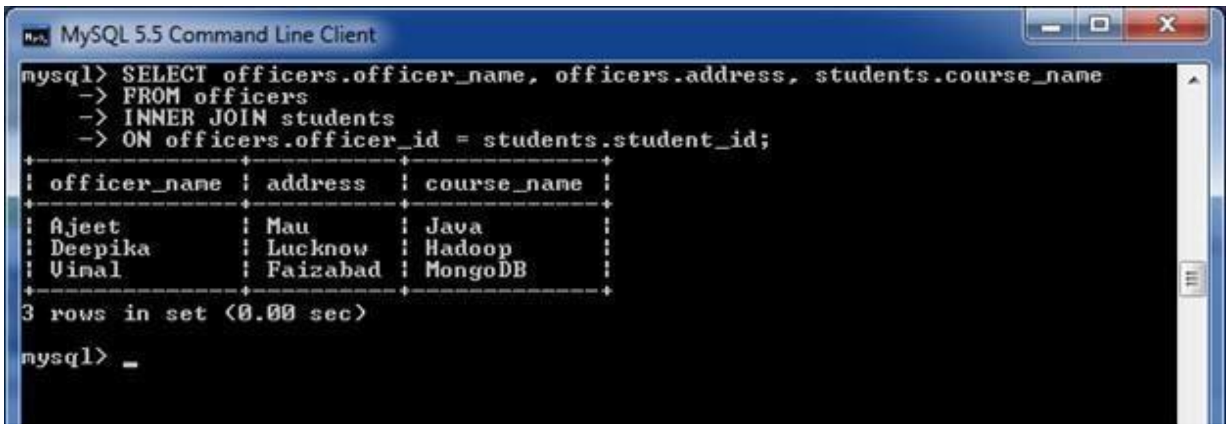
Consider two tables "officers" and "students", having the following data.



```
+-----------------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

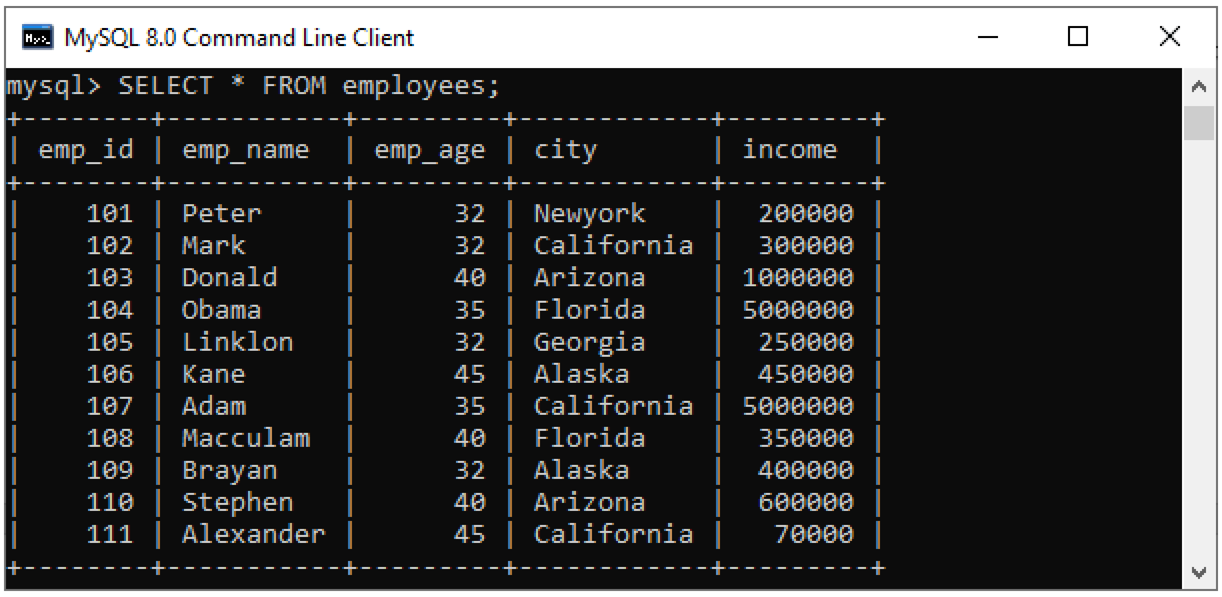
**SQL>** SELECT officers.officer_name, officers.address, students.course_name FROM officers INNER JOIN students ON officers.officer_id = students.student_id;

**Output**



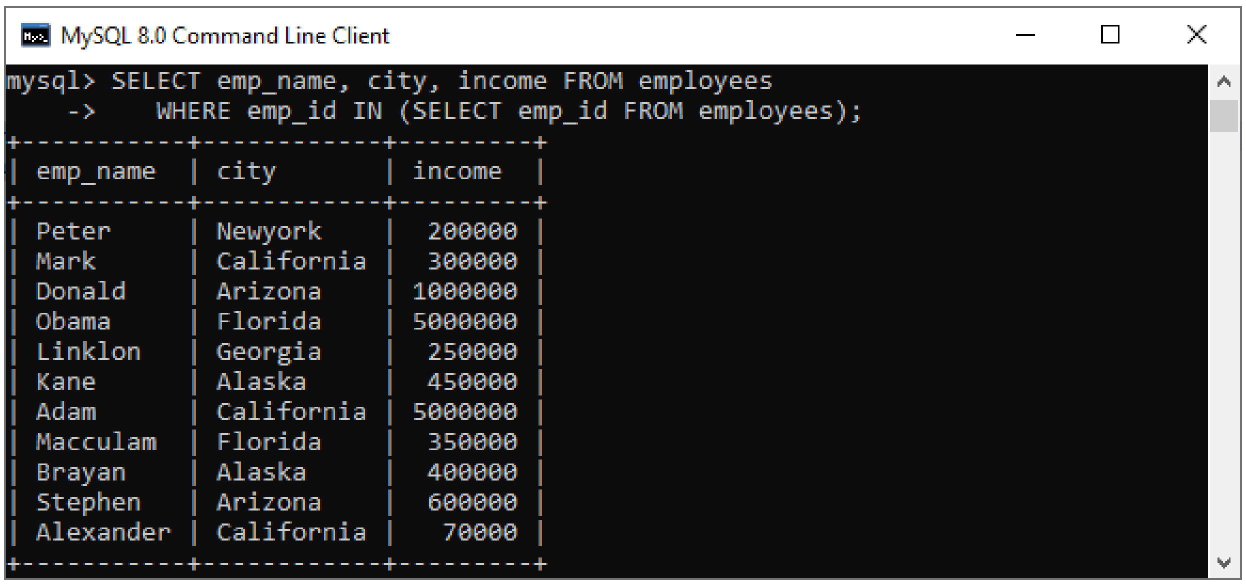
```
mysql> SELECT officers.officer_name, officers.address, students.course_name
    -> FROM officers
    -> INNER JOIN students
    -> ON officers.officer_id = students.student_id;
+--------------+----------+-------------+
| officer_name | address  | course_name |
+--------------+----------+-------------+
| Ajeet        | Mau      | Java        |
| Deepika      | Lucknow  | Hadoop      |
| Vimal        | Faizabad | MongoDB     |
+--------------+----------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

**MYSQL SUBQUERY**



```
mysql> SELECT * FROM employees;
+--------+-----------+---------+------------+---------+
| emp_id | emp_name  | emp_age | city       | income  |
+--------+-----------+---------+------------+---------+
|    101 | Peter     |      32 | Newyork    |  200000 |
|    102 | Mark      |      32 | California |  300000 |
|    103 | Donald    |      40 | Arizona    | 1000000 |
|    104 | Obama     |      35 | Florida    | 5000000 |
|    105 | Linklon   |      32 | Georgia    |  250000 |
|    106 | Kane      |      45 | Alaska     |  450000 |
|    107 | Adam      |      35 | California | 5000000 |
|    108 | Macculam  |      40 | Florida    |  350000 |
|    109 | Brayan    |      32 | Alaska     |  400000 |
|    110 | Stephen   |      40 | Arizona    |  600000 |
|    111 | Alexander |      45 | California |   70000 |
+--------+-----------+---------+------------+---------+
```

**SQL>SELECT** emp_name, city, income **FROM** employees **WHERE** emp_id IN (**SELECT** e mp_id **FROM** employees);



MySQL 8.0 Command Line Client

```
mysql> SELECT emp_name, city, income FROM employees
    ->    WHERE emp_id IN (SELECT emp_id FROM employees);
+-----------+------------+---------+
| emp_name  | city       | income  |
+-----------+------------+---------+
| Peter     | Newyork    |  200000 |
| Mark      | California |  300000 |
| Donald    | Arizona    | 1000000 |
| Obama     | Florida    | 5000000 |
| Linklon   | Georgia    |  250000 |
| Kane      | Alaska     |  450000 |
| Adam      | California | 5000000 |
| Macculam  | Florida    |  350000 |
| Brayan    | Alaska     |  400000 |
| Stephen   | Arizona    |  600000 |
| Alexander | California |   70000 |
+-----------+------------+---------+
```

**EX.NO :5 NATURAL JOIN,EQUI JOIN AND OUTER JOIN**

## Syntax:

SELECT **[column_names | *]** FROM **table_name1 NATURAL JOIN table_name2;**

/* -- Table name: customer -*/
**CREATE TABLE** customer ( id **INT** AUTO_INCREMENT **PRIMARY KEY**,
customer_name **VARCHAR**(55), account **int**, email **VARCHAR**(55) );

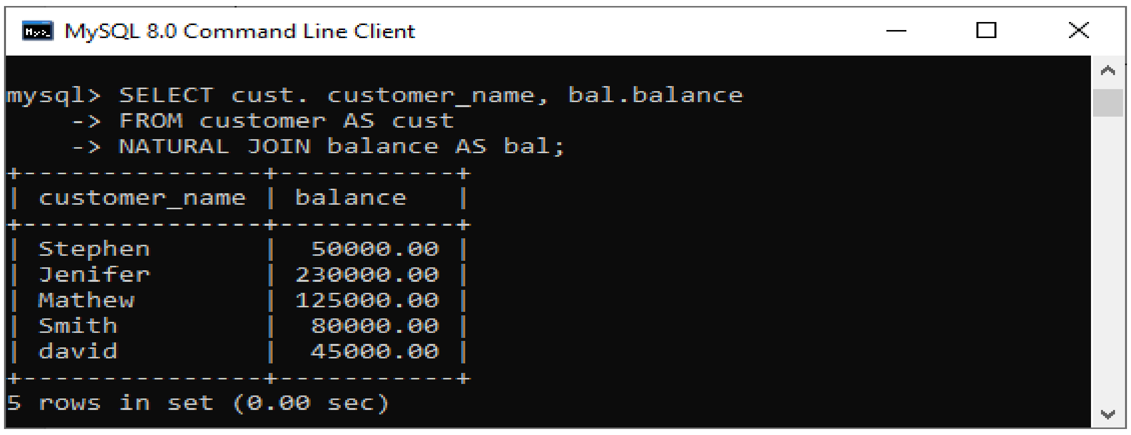
/* -- Table name: balance -*/
**CREATE TABLE** balance ( id **INT** AUTO_INCREMENT **PRIMARY KEY**,
account **int**, balance **FLOAT**(10, 2) );

/* -- Data for customer table -*/
**INSERT INTO** customer(customer_name, account, email) **VALUES**('Stephen', 1030, 'stephen @javatpoint.com'), ('Jenifer', 2035, 'jenifer@javatpoint.com'), ('Mathew', 5564, 'mathew@java tpoint.com'), ('Smith', 4534, 'smith@javatpoint.com'), ('David', 7648, 'david@javatpoint.com');

/* -- Data for balance table -*/
**INSERT INTO** balance(account, balance)
**VALUES**(1030, 50000.00), (2035, 230000.00), (5564, 125000.00), (4534, 80000.00),
(7648, 45000.00);

**NATURAL JOIN:**

MySQL> **SELECT** cust. customer_name, bal.balance **FROM** customer **AS** cust NATURAL JOIN balance **AS** bal;



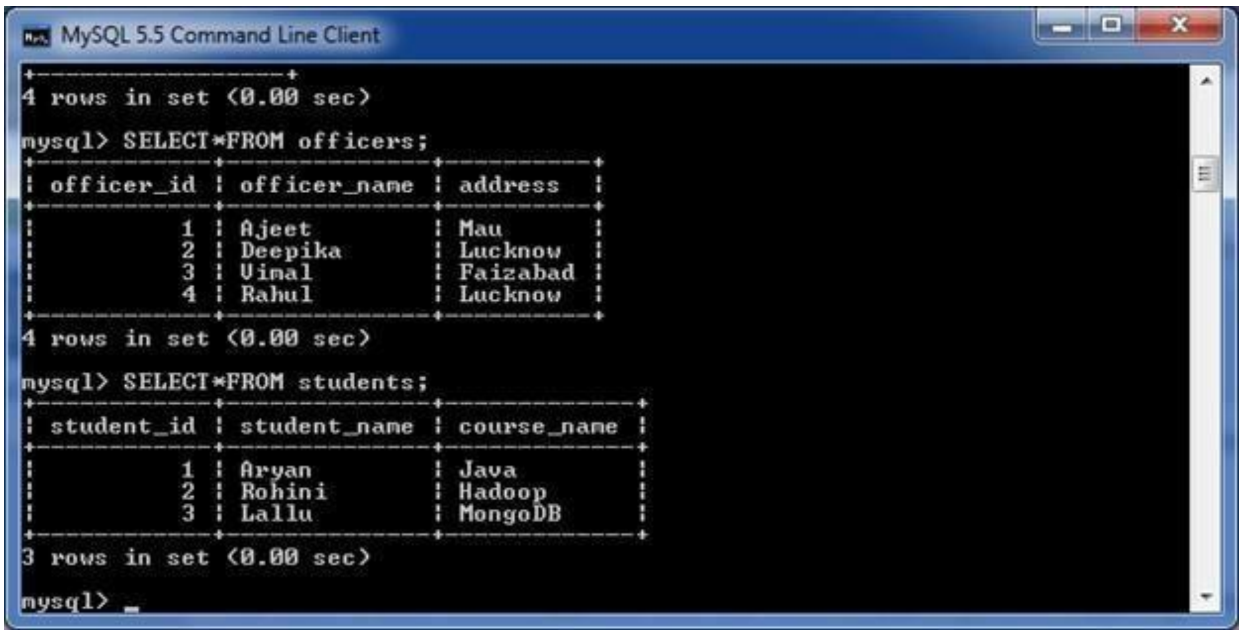
```
mysql> SELECT cust. customer_name, bal.balance
    -> FROM customer AS cust
    -> NATURAL JOIN balance AS bal;
+---------------+-----------+
| customer_name | balance   |
+---------------+-----------+
| Stephen       |  50000.00 |
| Jenifer       | 230000.00 |
| Mathew        | 125000.00 |
| Smith         |  80000.00 |
| david         |  45000.00 |
+---------------+-----------+
5 rows in set (0.00 sec)
```

**MYSQL RIGHT OUTER JOIN**

**Syntax:**

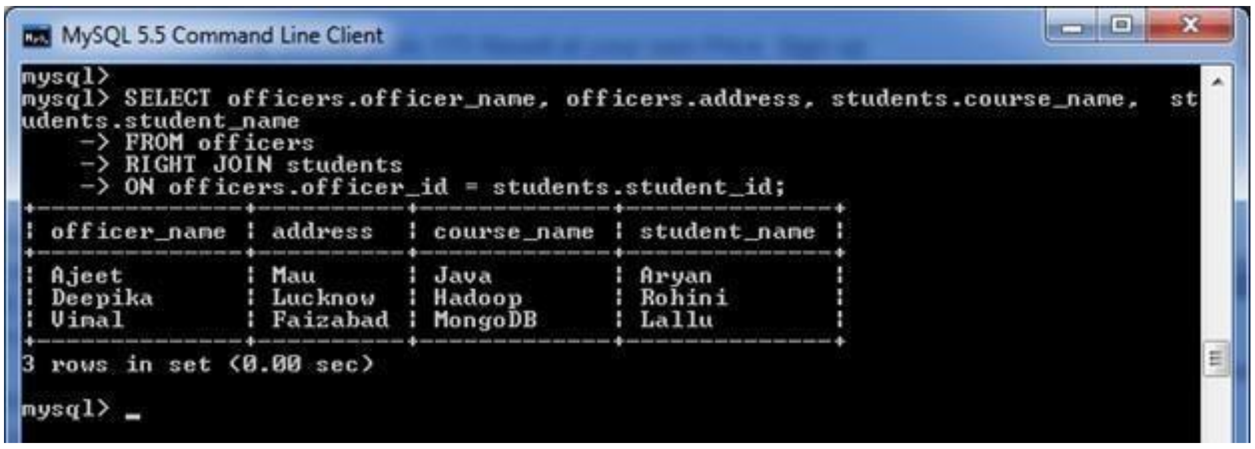
SELECT columns FROM table1 RIGHT [OUTER] JOIN table2 ON table1.column = table2.co lumn;

Consider two tables "officers" and "students", having the following data.


```
MySQL 5.5 Command Line Client
+-------------------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

**MySQL>SELECT** officers.officer_name, officers.address, students.course_name, students.student_name **FROM** officers RIGHT JOIN students **ON** officers.officer_id = students.student_id;

**Output**

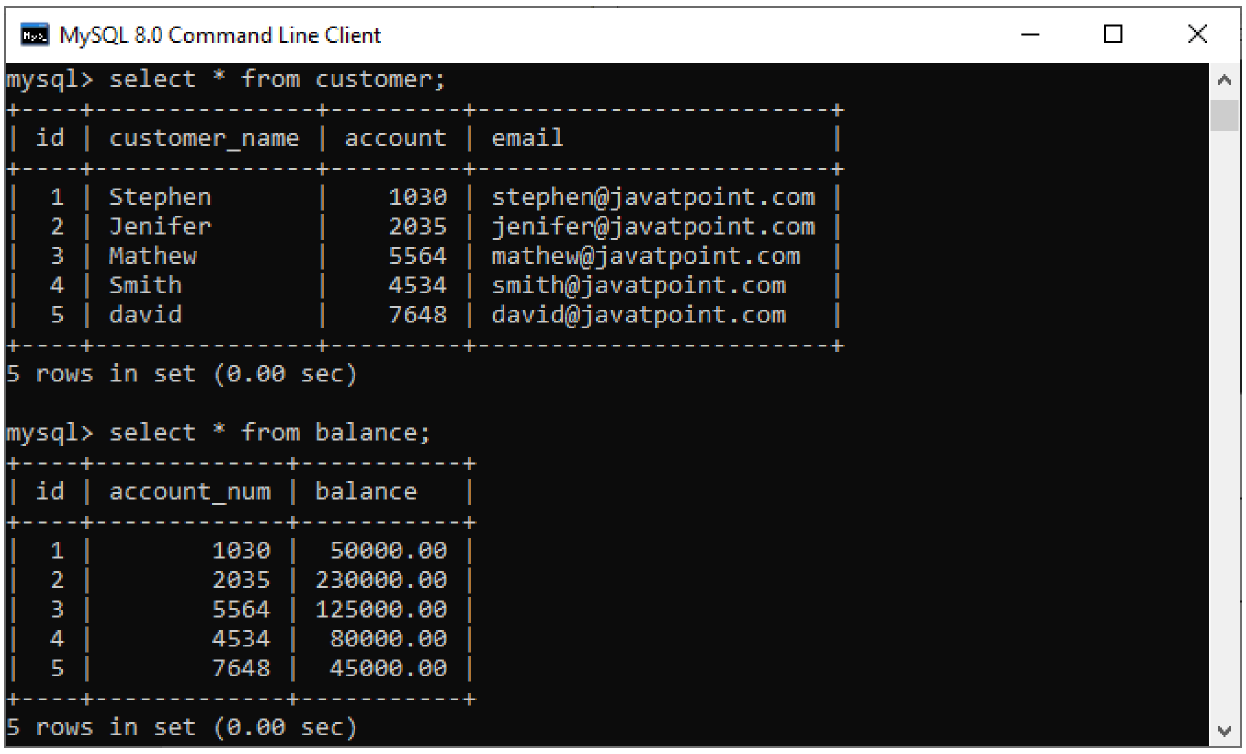

```
MySQL 5.5 Command Line Client
mysql>
mysql> SELECT officers.officer_name, officers.address, students.course_name,  st
udents.student_name
    -> FROM officers
    -> RIGHT JOIN students
    -> ON officers.officer_id = students.student_id;
+--------------+----------+-------------+--------------+
| officer_name | address  | course_name | student_name |
+--------------+----------+-------------+--------------+
| Ajeet        | Mau      | Java        | Aryan        |
| Deepika      | Lucknow  | Hadoop      | Rohini       |
| Vimal        | Faizabad | MongoDB     | Lallu        |
+--------------+----------+-------------+--------------+
3 rows in set (0.00 sec)

mysql> _
```

**EQUI JOIN**

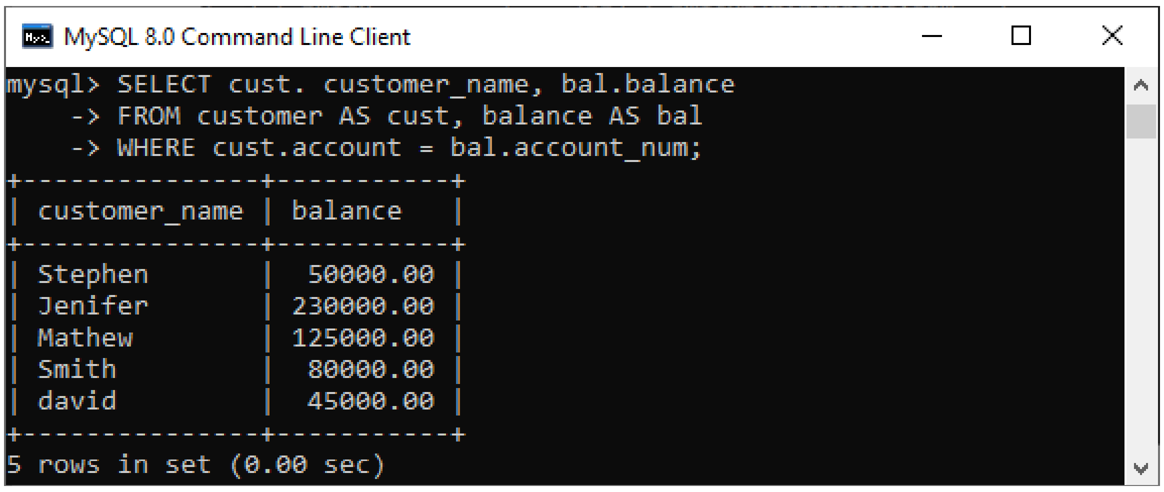
SELECT column_name (s) FROM table_name1, table_name2,...... , table_nameN

WHERE table_name1.column_name = table_name2.column_name;

Consider two tables named **customer** and **balance**


```
MySQL 8.0 Command Line Client
mysql> select * from customer;
+----+---------------+---------+------------------------+
| id | customer_name | account | email                  |
+----+---------------+---------+------------------------+
|  1 | Stephen       |    1030 | stephen@javatpoint.com |
|  2 | Jenifer       |    2035 | jenifer@javatpoint.com |
|  3 | Mathew        |    5564 | mathew@javatpoint.com  |
|  4 | Smith         |    4534 | smith@javatpoint.com   |
|  5 | david         |    7648 | david@javatpoint.com   |
+----+---------------+---------+------------------------+
5 rows in set (0.00 sec)

mysql> select * from balance;
+----+-------------+-----------+
| id | account_num | balance   |
+----+-------------+-----------+
|  1 |        1030 |  50000.00 |
|  2 |        2035 | 230000.00 |
|  3 |        5564 | 125000.00 |
|  4 |        4534 |  80000.00 |
|  5 |        7648 |  45000.00 |
+----+-------------+-----------+
5 rows in set (0.00 sec)
```

MySQL> **SELECT** cust. customer_name, bal.balance **FROM** customer **AS** cust, balance **AS** bal **WHERE** cust.account = bal.account_num;


MySQL 8.0 Command Line Client

```
mysql> SELECT cust. customer_name, bal.balance
    -> FROM customer AS cust, balance AS bal
    -> WHERE cust.account = bal.account_num;
+---------------+-----------+
| customer_name | balance   |
+---------------+-----------+
| Stephen       |  50000.00 |
| Jenifer       | 230000.00 |
| Mathew        | 125000.00 |
| Smith         |  80000.00 |
| david         |  45000.00 |
+---------------+-----------+
5 rows in set (0.00 sec)
```

**EX.NO:6 PROCEDURE AND FUNCTIONS**

**SETTING SERVEROUTPUT ON:**

```
SQL> SET SERVEROUTPUT  ON
```

**PROGRAM:**

**PROCEDURE USING POSITIONAL PARAMETERS:**

```
SQL> SET SERVEROUTPUT  ON
SQL> CREATE OR REPLACE  PROCEDURE  PROC1 AS
  2  BEGIN
  3     DBMS_OUTPUT.PUT_LINE('Hello   from procedure...');
  4  END;
  5 /
```

**Output**

```
Procedure created.

SQL> EXECUTE  PROC1
Hello  from procedure...



PL/SQL procedure successfully completed.

SQL> create table student(regno  number(4),name  varchar2)20),mark1  number(3),  mark2
        number(3),  mark3 number(3),  mark4 number(3),  mark5 number(3));

        Table created
SQL> insert  into  student  values  (101,'priya',  78, 88,77,60,89);

        1 row created.

SQL> insert  into  student  values  (102,'surya',  99,77,69,81,99);

        1 row created.
SQL> insert  into  student  values  (103,'suryapriya',  100,90,97,89,91);

        1 row created.
SQL> select  * from student;

regno    name       mark1  mark2   mark3   mark4   mark5
--------------------------------------------------------
101    priya             78     88      77         60          89
102    surya              99    77      69         81          99
103    suryapriya      100 90       97         89          91

SQL> declare
2    ave number(5,2);
3    tot number(3);
4    cursor c_mark  is  select*from  student  where  mark1>=40 and mark2>=40  and
5    mark3>=40  and  mark4>=40 and mark5>=40;
6    begin
7    dbms_output.put_line('regno  name mark1 mark2 mark3 mark4 mark4 mark5 total
8    average');
9    dbms_output.put_line('_____________________________________________');
10  for student  in  c_mark
11  loop
12  tot:=student.mark1+student.mark2+student.mark3+student.mark4+student.mark5;
13  ave:=tot/5;
```

```
14  dbms_output.put_line(student.regno||rpad(student.name,15)
15  ||rpad(student.mark1,6)||rpad(student.mark2,6)||rpad(student.mark3,6)
16  ||rpad(student.mark4,6)||rpad(student.mark5,6)||rpad(tot,8)||rpad(ave,5));
17  end loop;
18  end;
19 /
```

**OUTPUT**

```
regno    name       mark1  mark2   mark3   mark4   mark5   total  average
------------------------------------------------------------------------------
101    priya           78     88      77         60          89      393     79
102    surya           99    77      69         81          99      425      85
103    suryapriya      100 90       97         89          91      467      93

PL/SQL procedure successfully completed.
```

```
SQL> create table phonebook (phone_no number (6) primary key,username
varchar2(30),doorno varchar2(10),
street varchar2(30),place varchar2(30),pincode char(6));

Table created.

SQL> insert into phonebook values(20312,'vijay','120/5D','bharathi street','NGO
colony','629002');

1 row created.

SQL> insert into phonebook values(29467,'vasanth','39D4','RK bhavan','sarakkal
vilai','629002');

1 row created.

SQL> select * from phonebook;

PHONE_NO USERNAME   DOORNO    STREET          PLACE        PINCODE
-----------------------------------      ---------------  ------------------      ----------------------
20312            vijay               120/5D        bharathi street      NGO colony       629002

29467           vasanth              39D4          RK bhavan           sarakkal vilai       62900
```

```
SQL> create or replace function  findAddress(phone  in number)  return  varchar2  as
address varchar2(100);


    begin
    select username||','||doorno  ||','||street ||','||place||','||pincode  into address from phonebook
    where phone_no=phone;
    return address;
    exception
    when no_data_found  then return 'address not found';
    end;
    /

    Function created.

    SQL>declare
      2   address varchar2(100);
      3   begin
      4   address:=findaddress(20312);
      5   dbms_output.put_line(address);
      6   end;
      7 /
```

**OUTPUT**

```
Vijay,120/5D,bharathi  street,NGO colony,629002
```

## EX.NO:7 DCL AND TCL COMMANDS

**DCL COMMANDS**

**GRANT**

GRANT privilege_name ON object_name TO {user_name |PUBLIC |role_name}
[WITH GRANT OPTION];

**MySQL>** GRANT SELECT ON employee TO

user1;Command Successfully Completed

**REVOKE**

REVOKE privilege_name ON object_name FROM {user_name |PUBLIC |role_name}

**MySQL>** REVOKE SELECT ON employee FROM

user1;Command Successfully Completed

**TCL(TRNSACTION CONTROL LANGUAGE)**

**SQL> SAVEPOINT S1;**
Savepoint created.

```
SQL> SELECT * FROM EMP;

EMPNO      ENAME          DESIGNATIN        SALARY
-----------  ------------------------------------------------------------
101        NAGARAJAN      LECTURER          16000
102        SARAVANAN     ASST. PROF         16000
104        CHINNI          HOD, PROF         45000

SQL> INSERT INTO EMP VALUES(105,'PARTHASAR','STUDENT',100);

1 row created.

SQL> SELECT * FROM EMP;

EMPNO      ENAME          DESIGNATIN        SALARY
-----------  ------------------------------------------------------------
105        PARTHASAR       STUDENT          100
101        NAGARAJAN      LECTURER          16000
102        SARAVANAN     ASST. PROF         16000
104        CHINNI          HOD, PROF         45000
```

**ROLL BACK**

```
SQL> ROLL BACK S1;

Rollback complete.

SQL> SELECT * FROM EMP;
EMPNO      ENAME          DESIGNATIN        SALARY
--------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------
101       NAGARAJAN    LECTURER       16000
102       SARAVANAN   ASST. PROF      16000
104       CHINNI        HOD, PROF       45000
```

**COMMIT**

```
SQL> COMMIT;

Commit complete.
```

**EX.NO:8** **CREATION OF DATABASE TRIGGERS**

**SYNTAX**

create or replace trigger trigger name [before/after] {DML statements} on [table name] [for each row/statement] begin

```
-----------------------------
-----------------------------
exception
end;
```

**PROGRAM**

```
SQL>create table poo(rno number(5),name varchar2(10));
Table created.
SQL>insert into poo values (01.‟kala‟);
1 row created.
SQL>select * from poo;

RNO           NAME
------        ----------
1              kala
2              priya


SQL>create or replace trigger pool before insert on poo for each row
2 declare
3 rno poo.rno%type
4 cursor c is select rno from poo;
5 begin
6 open c;
7 loop;
8 fetch c into rno;
9 if:new.rno=rno then
10 raise_application_error(-20005,‟rno already exist‟);
11 end if;
12 exit when c%NOTFOUND
13 end loop;
14 close c;
15 end;
16 /
Trigger created.
```

**OUTPUT**

```
SQL>insert into poo values(01,‟kala‟)
Insert into poo values (01,‟kala‟)
*
ERROR at line1:
ORA-20005:rno already exist
ORA-06512:”SECONDCSEA.POOL”,line 9
ORA-04088:error during execution at trigger “SECONDCSEA.POOL”
```

**EX.NO:9 VIEWS AND INDEX**

**CREATION OF TABLE**

```
SQL>   CREATE TABLE EMPLOYEE (
         EMPLOYEE_NAMEVARCHAR2(10),
         EMPLOYEE_NONUMBER(8),
         DEPT_NAME VARCHAR2(10),
         DEPT_NO NUMBER (5),DATE_OF_JOIN  DATE);


Table created.
```

**TABLE DESCRIPTION**

```
-----------------------------------
SQL> DESC EMPLOYEE;

NAME                          NULL?           TYPE
 ----------------------------------    ---------     --------------------------
 EMPLOYEE_NAME                          VARCHAR2(10)
 EMPLOYEE_NO                            NUMBER(8)
 DEPT_NAME                              VARCHAR2(10)
 DEPT_NO                                NUMBER(5)
 DATE_OF_JOIN                           DATE
```

**CREATION OF VIEW**

```
--------------------------------
SQL> CREATE  VIEW EMPVIEW  AS SELECT
EMPLOYEE_NAME,EMPLOYEE_NO,DEPT_NAME,DEPT_NO,DATE_OF_JOIN  FROM
EMPLOYEE;

view created.
```

**DESCRIPTION OF VIEW**

```
 --------------------------------------
 SQL> DESC EMPVIEW;

 NAME                          NULL?   TYPE
 ------------------------------------------- --------- -------------------------------
 EMPLOYEE_NAME                         VARCHAR2(10)
 EMPLOYEE_NO                           NUMBER(8)
 DEPT_NAME                             VARCHAR2(10)
 DEPT_NO                               NUMBER(5)
```

**DISPLAY VIEW**

```
SQL> SELECT  *  FROM EMPVIEW;

EMPLOYEE_N  EMPLOYEE_NO  DEPT_NAME     DEPT_NO
.................................................................
RAVI           124            ECE           89
VIJAY          345            CSE           21
RAJ            98             IT            22
GIRI           100            CSE           67
```

**INSERTION INTO VIEW**

```
SQL> INSERT INTO EMPVIEW VALUES ('SRI', 120,'CSE', 67,'16-NOV-1981');

1 ROW CREATED.

SQL> SELECT  * FROM EMPVIEW;
EMPLOYEE_N  EMPLOYEE_NO  DEPT_NAME     DEPT_NO
-------------------------------------------------------------------------------------
RAVI          124                    ECE          89
VIJAY         345                    CSE          21
RAJ           98                      IT          22
GIRI          100                    CSE          67
SRI           120                    CSE          67
```

```
SQL> SELECT * FROM EMPLOYEE;

EMPLOYEE_N  EMPLOYEE_NO  DEPT_NAME     DEPT_NO DATE_OF_J
---------------------------------------------------    --------------------------------
RAVI            124                     ECE              89           15-JUN-05
VIJAY           345                     CSE              21           21-JUN-06
RAJ             98                      IT               22           30-SEP-06
GIRI            100                     CSE              67           14-NOV-81
SRI             120                     CSE              67           16-NOV-81
```

**DELETION OF VIEW**

**DELETE STATEMENT**

```
SQL> DELETE  FROM EMPVIEW  WHERE  EMPLOYEE_NAME='SRI';

SQL> SELECT  * FROM EMPVIEW;

EMPLOYEE_N  EMPLOYEE_NO  DEPT_NAME     DEPT_NO
..........................................................................
RAVI             124                    ECE            89
VIJAY            345                    CSE            21
RAJ              98                     IT             22
GIRI             100                    CSE            67
```

**UPDATE STATEMENT:**

```
SQL> UPDATE  EMPKAVIVIEW  SET EMPLOYEE_NAME='KAVI'  WHERE
EMPLOYEE_NAME='RAVI';

1 ROW UPDATED.

SQL> SELECT * FROM EMPKAVIVIEW;

EMPLOYEE_N  EMPLOYEE_NO  DEPT_NAME     DEPT_NO
------------------------------------------------------------
KAVI            124                  ECE              89
VIJAY           345                  CSE              21
RAJ             98                   IT               22
GIRI            100                  CSE              67
```

**DROP A VIEW:**

```
SQL>DROP VIEW  EMPVIEW;

  VIEW DROPED
```

**CREATE INDEX**

```
MySQL> CREATE  DATABASE
indexes;Query  OK, 1 row affected  (0.01
sec)

USE  indexes;

Database changed

MySQL>CREATE  TABLE

    employees  (employee_id  int,

    first_name  varchar(50),

    last_name  varchar(50),

    device_serial  varchar(15),      salary  int  );

Query OK, 0 rows affected  (0.00 sec)
```

```
INSERT INTO employees  VALUES

    (1, 'John', 'Smith', 'ABC123', 60000),    (2, 'Jane', 'Doe', 'DEF456', 65000),

    (3, 'Bob', 'Johnson', 'GHI789', 70000),    (4, 'Sally', 'Fields', 'JKL012', 75000),

    (5, 'Michael', 'Smith', 'MNO345', 80000),    (6, 'Emily', 'Jones', 'PQR678', 85000),

    (7, 'David', 'Williams', 'STU901', 90000),    (8, 'Sarah', 'Johnson', 'VWX234', 95000),

    (9, 'James', 'Brown', 'YZA567', 100000);

Query OK, 9 rows affected (0.010 sec)

Records: 9  Duplicates: 0 Warnings: 0

MySQL>CREATE         INDEX         salary         ON         employees(salary);

Mqsql>EXPLAIN  SELECT * FROM employees  WHERE  salary = 100000;

+___+..........+..........+..........+____+__________+........+........+_____+____+

| id | select_type  | table      | partitions  | type | possible_keys  | key     | key_len  | ref   | rows | filtered
|  1 | SIMPLE       | employees  | NULL        | ref  | salary         | salary  | 5        | const |    1 |    100.00  |+
______________+___________________+.....................+................+_______+_+_____+
..................+.......................+_____________+__________ 1  row  in  set,  1  warning
(0.00 sec)
```

**EX.NO:10** **XML DATABASE CREATION AND VALIDATION**

**CREATE TABLE**

```
CREATE TABLE person (

person_id INT NOT NULL PRIMARY KEY,

fname VARCHAR(40) NULL,

lname VARCHAR(40) NULL,

created TIMESTAMP

);
```

**XML FILE PERSON.XML**

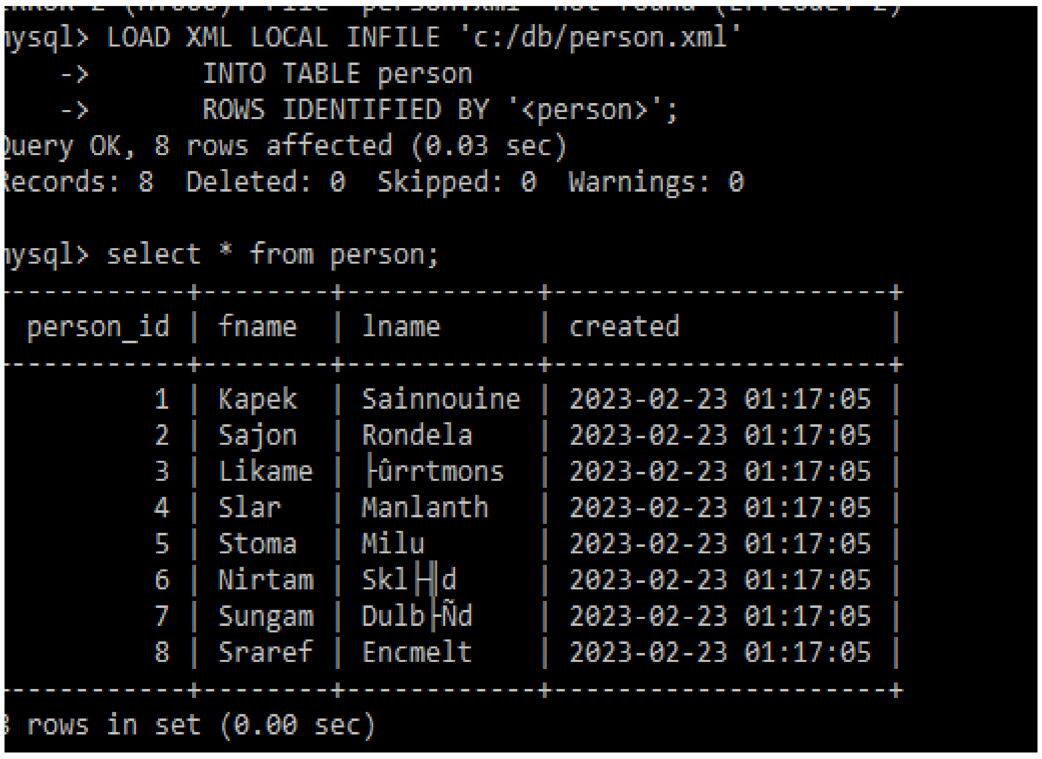
```
<list>
<personperson_id="1"fname="Kapek"lname="Sainnouine"/>
<personperson_id="2"fname="Sajon"lname="Rondela"/>
<personperson_id="3"><fname>Likame</fname><lname>Örrtmons</lname></person>
<personperson_id="4"><fname>Slar</fname><lname>Manlanth</lname></person>
<person><fieldname="person_id">5</field><fieldname="fname">Stoma</field>
<fieldname="lname">Milu</field></person>
<person><fieldname="person_id">6</field><fieldname="fname">Nirtam</field>
<fieldname="lname">Sklöd</field></person>
<personperson_id="7"><fname>Sungam</fname><lname>Dulbåd</lname></person>
<personperson_id="8"fname="Sraref"lname="Encmelt"/>
</list>
```

**INSERT VALUES USING LOADXMLDATAFILE**

```
LOAD XML LOCAL INFILE 'c:/db/person.xml'  //this is ths location of the xml data file

     INTO TABLE person

     ROWS IDENTIFIED BY '<person>';
```

**OUTPUT**

MySQL>Select * from person;

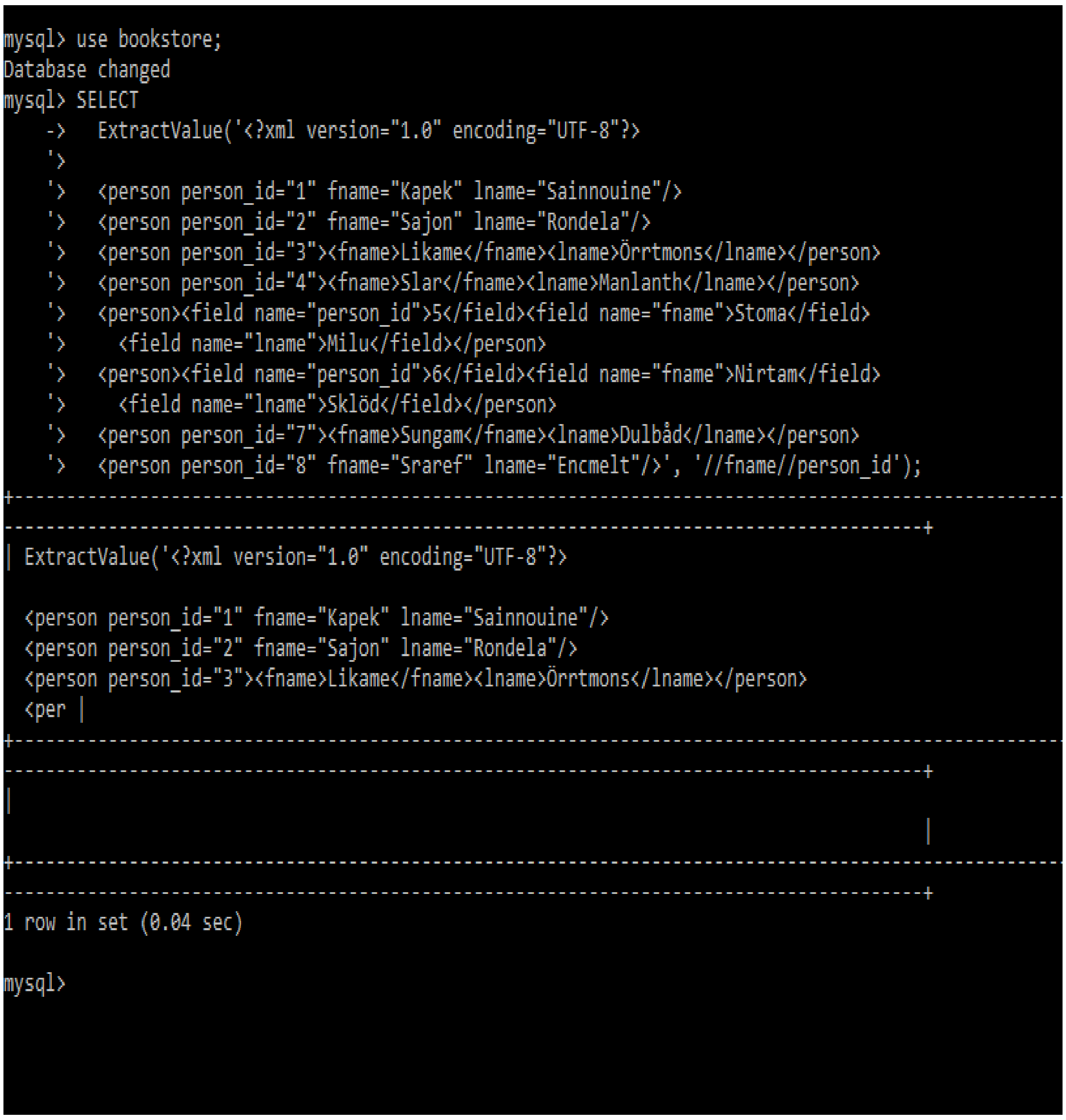


```
RROR 2 (HY000): File 'person.xml' not found (Errcode: 2)
ysql> LOAD XML LOCAL INFILE 'c:/db/person.xml'
    ->       INTO TABLE person
    ->       ROWS IDENTIFIED BY '<person>';
uery OK, 8 rows affected (0.03 sec)
ecords: 8  Deleted: 0  Skipped: 0  Warnings: 0

ysql> select * from person;
-----------+--------+------------+---------------------+
 person_id | fname  | lname      | created             |
-----------+--------+------------+---------------------+
         1 | Kapek  | Sainnouine | 2023-02-23 01:17:05 |
         2 | Sajon  | Rondela    | 2023-02-23 01:17:05 |
         3 | Likame | ├ûrrtmons  | 2023-02-23 01:17:05 |
         4 | Slar   | Manlanth   | 2023-02-23 01:17:05 |
         5 | Stoma  | Milu       | 2023-02-23 01:17:05 |
         6 | Nirtam | Skl├╢d     | 2023-02-23 01:17:05 |
         7 | Sungam | Dulb├Ñd    | 2023-02-23 01:17:05 |
         8 | Sraref | Encmelt    | 2023-02-23 01:17:05 |
-----------+--------+------------+---------------------+
 rows in set (0.00 sec)
```

**VALIDATE XML USING EXTRACTVALUE FUNCTION**

MySQL> SELECT

ExtractValue('<?xml version="1.0" encoding="UTF-8"?>



```
mysql> use bookstore;
Database changed
mysql> SELECT
    ->   ExtractValue('<?xml version="1.0" encoding="UTF-8"?>
    '>
    '>   <person person_id="1" fname="Kapek" lname="Sainnouine"/>
    '>   <person person_id="2" fname="Sajon" lname="Rondela"/>
    '>   <person person_id="3"><fname>Likame</fname><lname>Örrtmons</lname></person>
    '>   <person person_id="4"><fname>Slar</fname><lname>Manlanth</lname></person>
    '>   <person><field name="person_id">5</field><field name="fname">Stoma</field>
    '>     <field name="lname">Milu</field></person>
    '>   <person><field name="person_id">6</field><field name="fname">Nirtam</field>
    '>     <field name="lname">Sklöd</field></person>
    '>   <person person_id="7"><fname>Sungam</fname><lname>Dulbåd</lname></person>
    '>   <person person_id="8" fname="Sraref" lname="Encmelt"/>', '//fname//person_id');
+------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------+
| ExtractValue('<?xml version="1.0" encoding="UTF-8"?>

  <person person_id="1" fname="Kapek" lname="Sainnouine"/>
  <person person_id="2" fname="Sajon" lname="Rondela"/>
  <person person_id="3"><fname>Likame</fname><lname>Örrtmons</lname></person>
  <per |
+------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------+
|
                                                                                       |
+------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------+
1 row in set (0.04 sec)

mysql>
```

**EX.NO:11 CREATING DOCUMENT, COLUMNS & GRAPH USING NOSQL**

**Create database in mongodb**

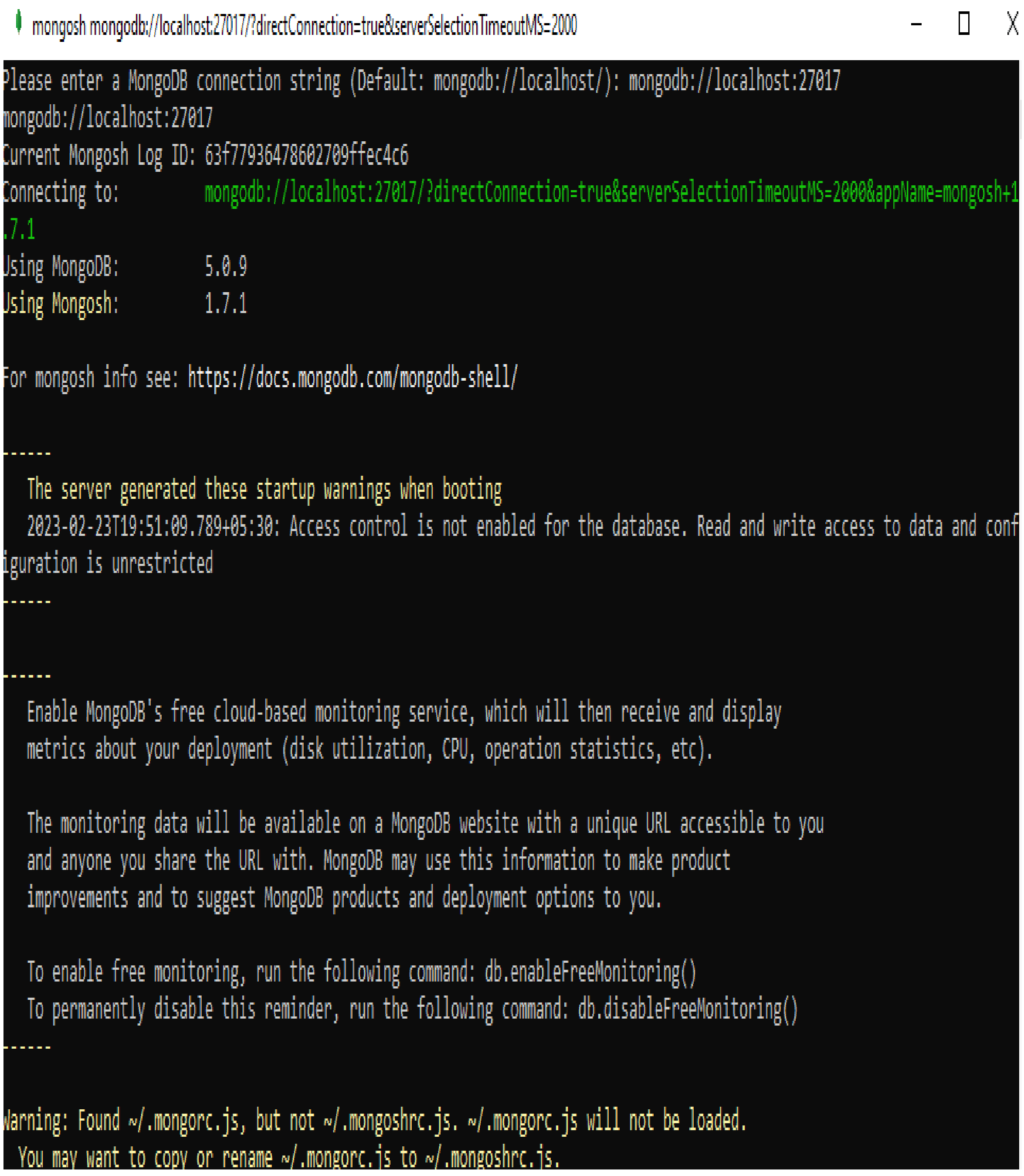
>Install Mongodb shell

>Connect with localhost

>Connection string:

mongodb://localhost:27017

**output:**

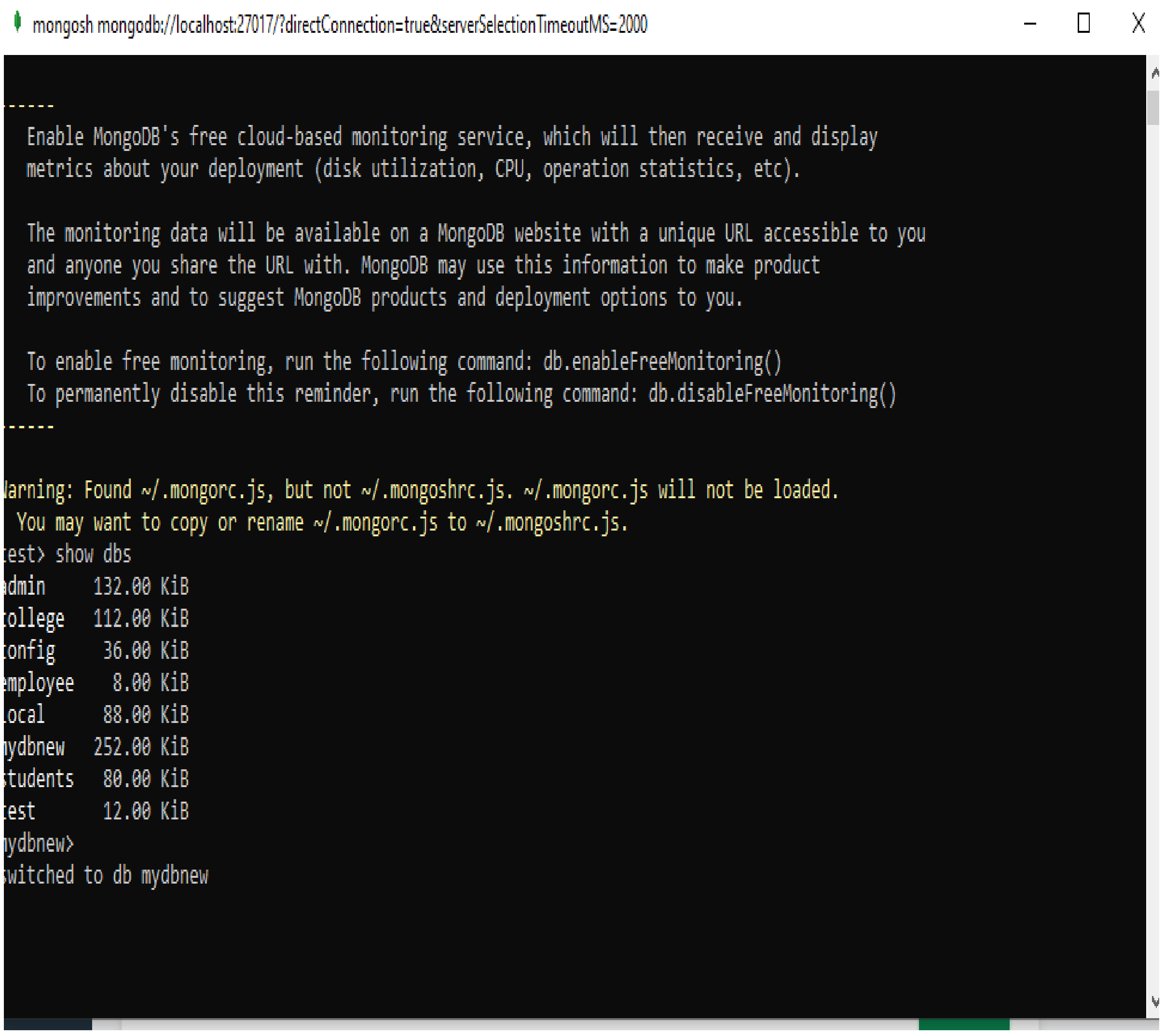


```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
Please enter a MongoDB connection string (Default: mongodb://localhost/): mongodb://localhost:27017
mongodb://localhost:27017
Current Mongosh Log ID: 63f77936478602709ffec4c6
Connecting to:          mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1
.7.1
Using MongoDB:          5.0.9
Using Mongosh:          1.7.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-02-23T19:51:09.789+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------

------
   Enable MongoDB's free cloud-based monitoring service, which will then receive and display
   metrics about your deployment (disk utilization, CPU, operation statistics, etc).

   The monitoring data will be available on a MongoDB website with a unique URL accessible to you
   and anyone you share the URL with. MongoDB may use this information to make product
   improvements and to suggest MongoDB products and deployment options to you.

   To enable free monitoring, run the following command: db.enableFreeMonitoring()
   To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
------

Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
  You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
```

**Create collection in mongodb**

use <database_name> command

**OUTPUT:**

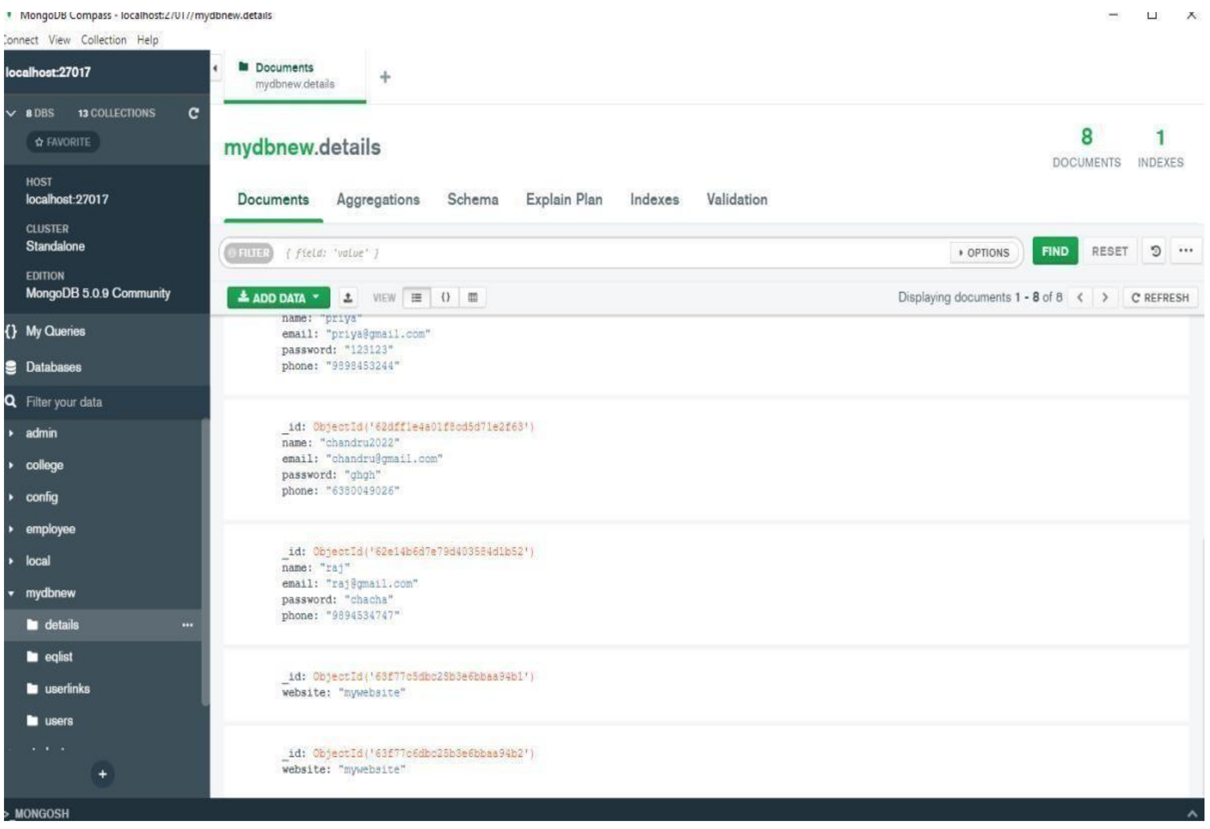


mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
-----
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
-----

arning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
 You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
est> show dbs
dmin     132.00 KiB
ollege   112.00 KiB
onfig     36.00 KiB
mployee    8.00 KiB
ocal      88.00 KiB
ydbnew   252.00 KiB
tudents   80.00 KiB
est       12.00 KiB
ydbnew>
witched to db mydbnew
```

**Create document in mongodb**

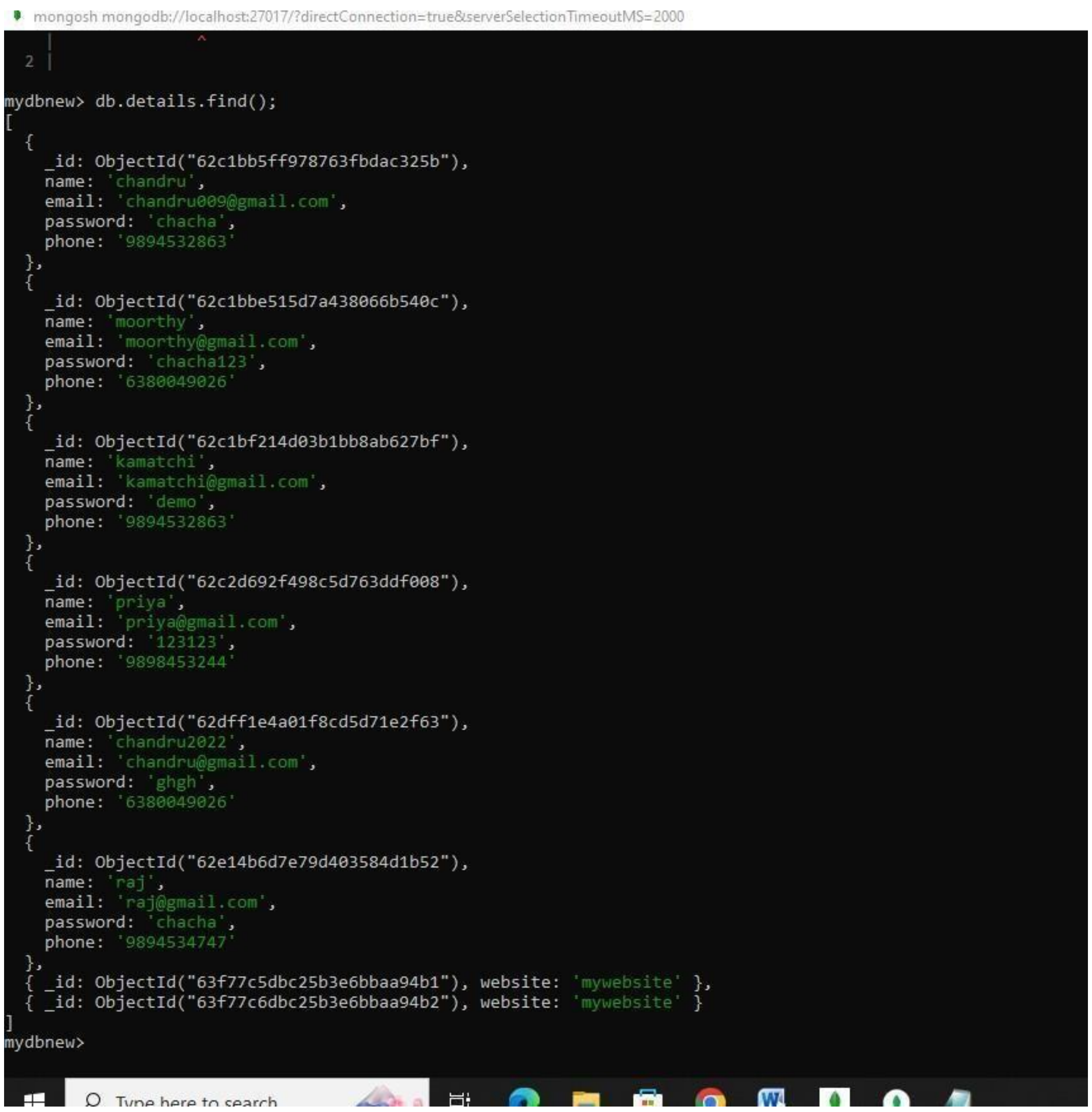
mydbnew>db.details.insertOne({"website":"mywebsite"})
Output:



```
name: "priya"
email: "priya@gmail.com"
password: "123123"
phone: "9898453244"

_id: ObjectId('62dff1e4a01f8cd5d71e2f63')
name: "chandru2022"
email: "chandru@gmail.com"
password: "ghgh"
phone: "6380049026"

_id: ObjectId('62e14b6d7e79d403584d1b52')
name: "raj"
email: "raj@gmail.com"
password: "chacha"
phone: "9894534747"

_id: ObjectId('63f77c5dbc25b3e6bbaa94b1')
website: "mywebsite"

_id: ObjectId('63f77c6dbc25b3e6bbaa94b2')
website: "mywebsite"
```

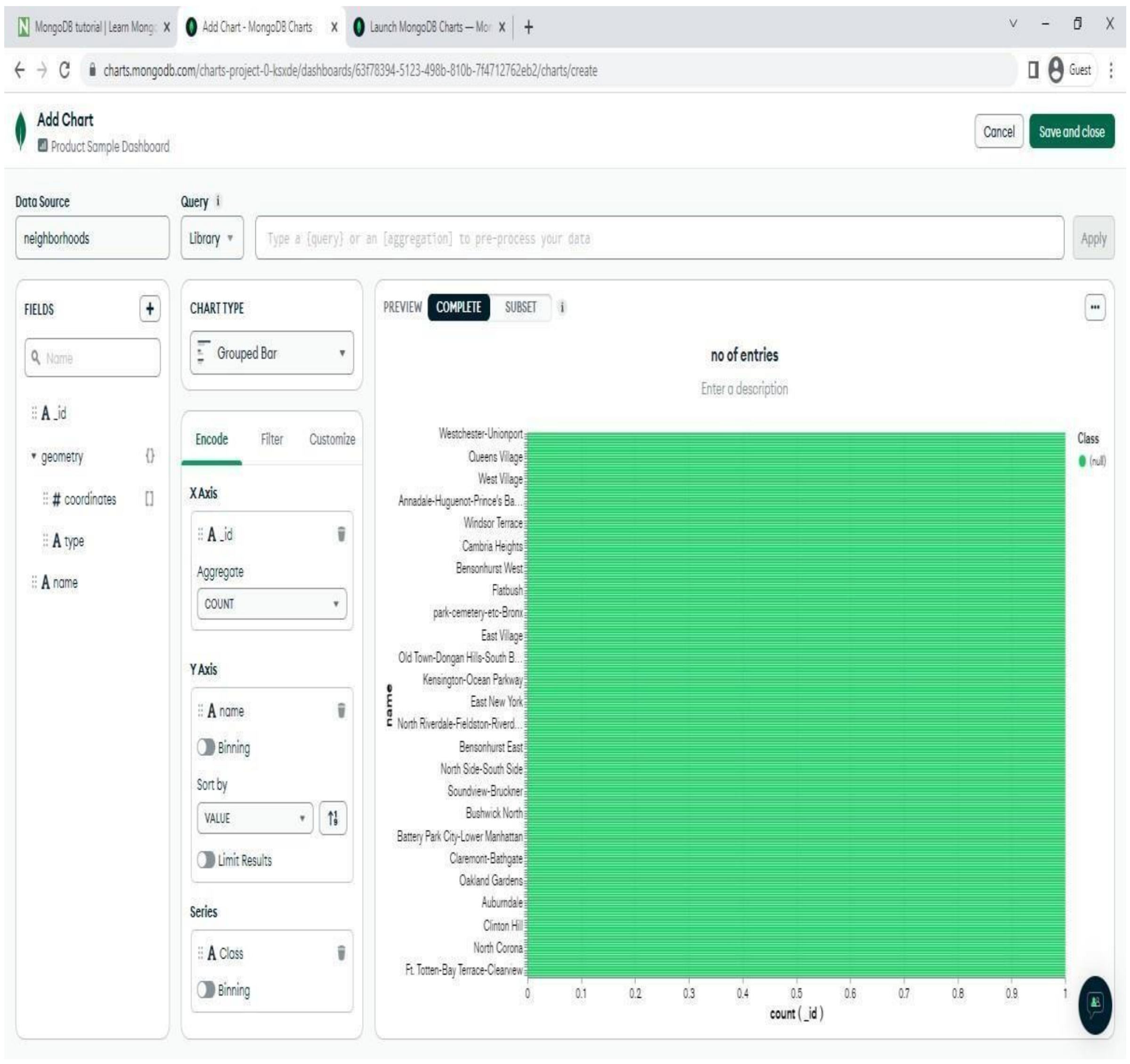
**<u>Display all documents</u>** **Db.details.find()**

**Output**



```
mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
    |               ^
  2 |

mydbnew> db.details.find();
[
  {
    _id: ObjectId("62c1bb5ff978763fbdac325b"),
    name: 'chandru',
    email: 'chandru009@gmail.com',
    password: 'chacha',
    phone: '9894532863'
  },
  {
    _id: ObjectId("62c1bbe515d7a438066b540c"),
    name: 'moorthy',
    email: 'moorthy@gmail.com',
    password: 'chacha123',
    phone: '6380049026'
  },
  {
    _id: ObjectId("62c1bf214d03b1bb8ab627bf"),
    name: 'kamatchi',
    email: 'kamatchi@gmail.com',
    password: 'demo',
    phone: '9894532863'
  },
  {
    _id: ObjectId("62c2d692f498c5d763ddf008"),
    name: 'priya',
    email: 'priya@gmail.com',
    password: '123123',
    phone: '9898453244'
  },
  {
    _id: ObjectId("62dff1e4a01f8cd5d71e2f63"),
    name: 'chandru2022',
    email: 'chandru@gmail.com',
    password: 'ghgh',
    phone: '6380049026'
  },
  {
    _id: ObjectId("62e14b6d7e79d403584d1b52"),
    name: 'raj',
    email: 'raj@gmail.com',
    password: 'chacha',
    phone: '9894534747'
  },
  { _id: ObjectId("63f77c5dbc25b3e6bbaa94b1"), website: 'mywebsite' },
  { _id: ObjectId("63f77c6dbc25b3e6bbaa94b2"), website: 'mywebsite' }
]
mydbnew>
```

**OUTPUT**



Add Chart
Product Sample Dashboard

Data Source: neighborhoods

Query: Library

CHART TYPE: Grouped Bar

X Axis: _id, Aggregate: COUNT

Y Axis: name, Sort by: VALUE

Series: Class

no of entries

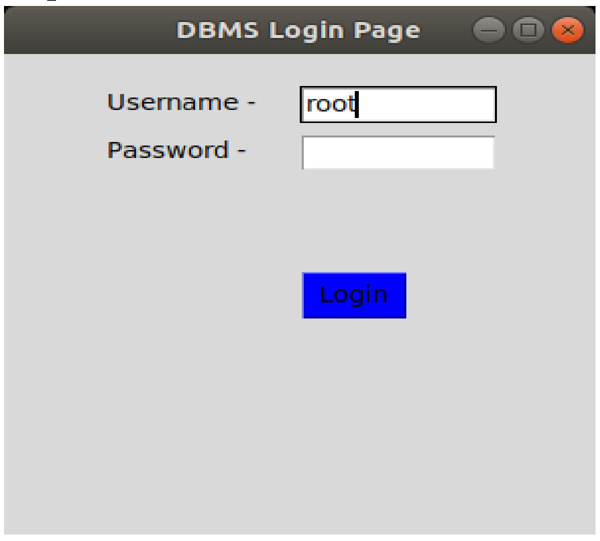
**EX.NO:12** **SIMPLE GUI APPLICATION USING DATABASE**

**PROGRAM**

```
import tkinter as tk
import
MySQL.connectorfrom
tkinter import *

 def submitact():

    user = Username.get()
    passw = password.get()

    print(f"The name entered by you is {user} {passw}")

    logintodb(user, passw)


def logintodb(user, passw):

    # If password is enetered by the
    # user
    if passw:
        db = MySQL.connector.connect(host ="localhost",
                               user = user,
                               password = passw,
                               db ="College")
        cursor = db.cursor()

    # If no password is enetered by the
    # user
    else:
        db = MySQL.connector.connect(host ="localhost",
                               user = user,
                               db ="College")
        cursor = db.cursor()

    # A Table in the database
    savequery = "select * from STUDENT"

    try:
        cursor.execute(savequery)
        myresult = cursor.fetchall()


        # Printing the result of the
        # query
        for x in myresult:
            print(x)
        print("Query Executed successfully")
    except:
        db.rollback()
        print("Error occurred")
root = tk.Tk()
root.geometry("300x300")
root.title("DBMS Login Page")

# Defining the first row
lblfrstrow = tk.Label(root, text ="Username -", )
lblfrstrow.place(x = 50, y = 20)

Username = tk.Entry(root, width = 35)
Username.place(x = 150, y = 20, width = 100)
```

```
lblsecrow = tk.Label(root, text ="Password -")
lblsecrow.place(x = 50, y = 50)

password = tk.Entry(root, width = 35)
password.place(x = 150, y = 50, width = 100)

submitbtn = tk.Button(root, text ="Login",
                      bg ='blue', command = submitact)
submitbtn.place(x = 150, y = 135, width = 55)

root.mainloop()
```

**Output:**



**EX.NO:13 CASE STUDY USING REALTIME DATABASE APPLICATIONS**

**ER diagram of Bank Management System**

ER diagram is known as Entity-Relationship diagram. It is used to analyze to structure of the Database. It shows relationships between entities and their attributes. An ER model provides a means of communication.

ER diagram of Bank has the following description :

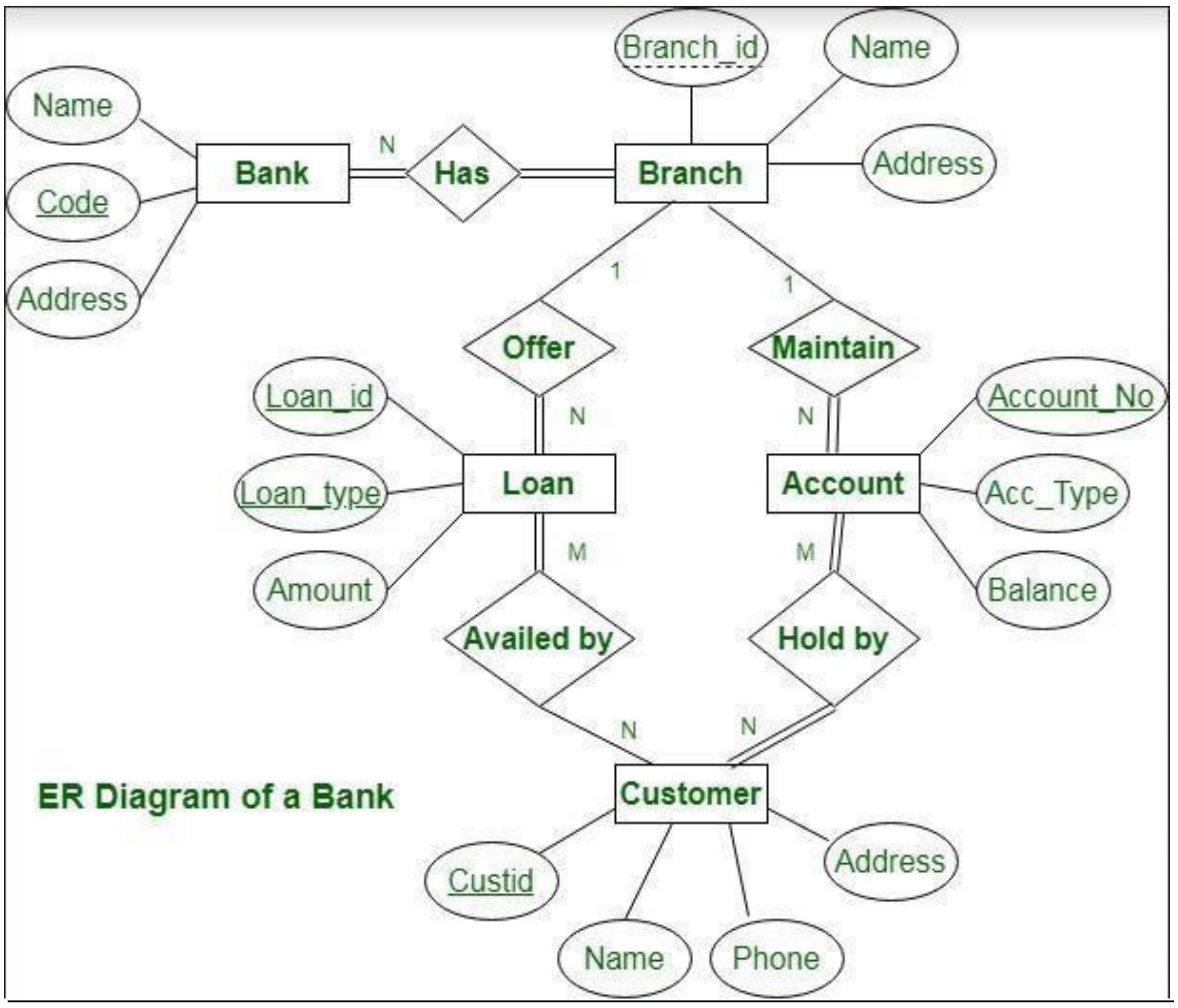
- Banks are identified by a name, code, address of main office.
- Bank have Customer
- Banks have branches.
- Branches are identified by a branch_no., branch_name, address.
- Customers are identified by name, cust-id, phone number, address.
- Customer can have one or more accounts.
- Accounts are identified by account_no., acc_type, balance.
- Customer can avail loans.
- Loans are identified by loan_id, loan_type and amount.
- Account and loans are related to bank's branch.

**Entities** and their **Attributes** are :

- **Bank Entity :** Attributes of Bank Entity are Bank Name, Code and Address.
  Code is Primary Key for Bank Entity.
- **Customer Entity :** Attributes of Customer Entity are Customer_id, Name, Phone Number and Address.
  Customer_id is Primary Key for Customer Entity.
- **Branch Entity :** Attributes of Branch Entity are Branch_id, Name and Address.
  Branch_id is Primary Key for Branch Entity.
- **Account Entity :** Attributes of Account Entity are Account_number, Account_Type and Balance.
  Account_number is Primary Key for Account Entity.
- **Loan Entity :** Attributes of Loan Entity are Loan_id, Loan_Type and Amount.
  Loan_id is Primary Key for Loan Entity.

This bank ER diagram illustrates key information about bank, including entities such as branches, customers, accounts, and loans. It allows us to understand the relationships between entities.

**ER Diagram of Bank Management System :**



Relationships are :

- **Bank has Branches => 1 : N**

  One Bank can have many Branches but one Branch can not belong to many Banks, so the relationship between Bank and Branch is one to many relationship.

- **Branch maintain Accounts => 1 : N**

  One Branch can have many Accounts but one Account can not belong to many Branches, so the relationship between Branch and Account is one to many relationship.

- **Branch offer Loans => 1 : N**

  One Branch can have many Loans but one Loan can not belong to many Branches, so the relationship between Branch and Loan is one to many relationship.

- **Account held by Customers => M : N**

  One Customer can have more than one Accounts and also One Account can be held by one or more Customers, so the relationship between Account and Customers is many to many relationship.

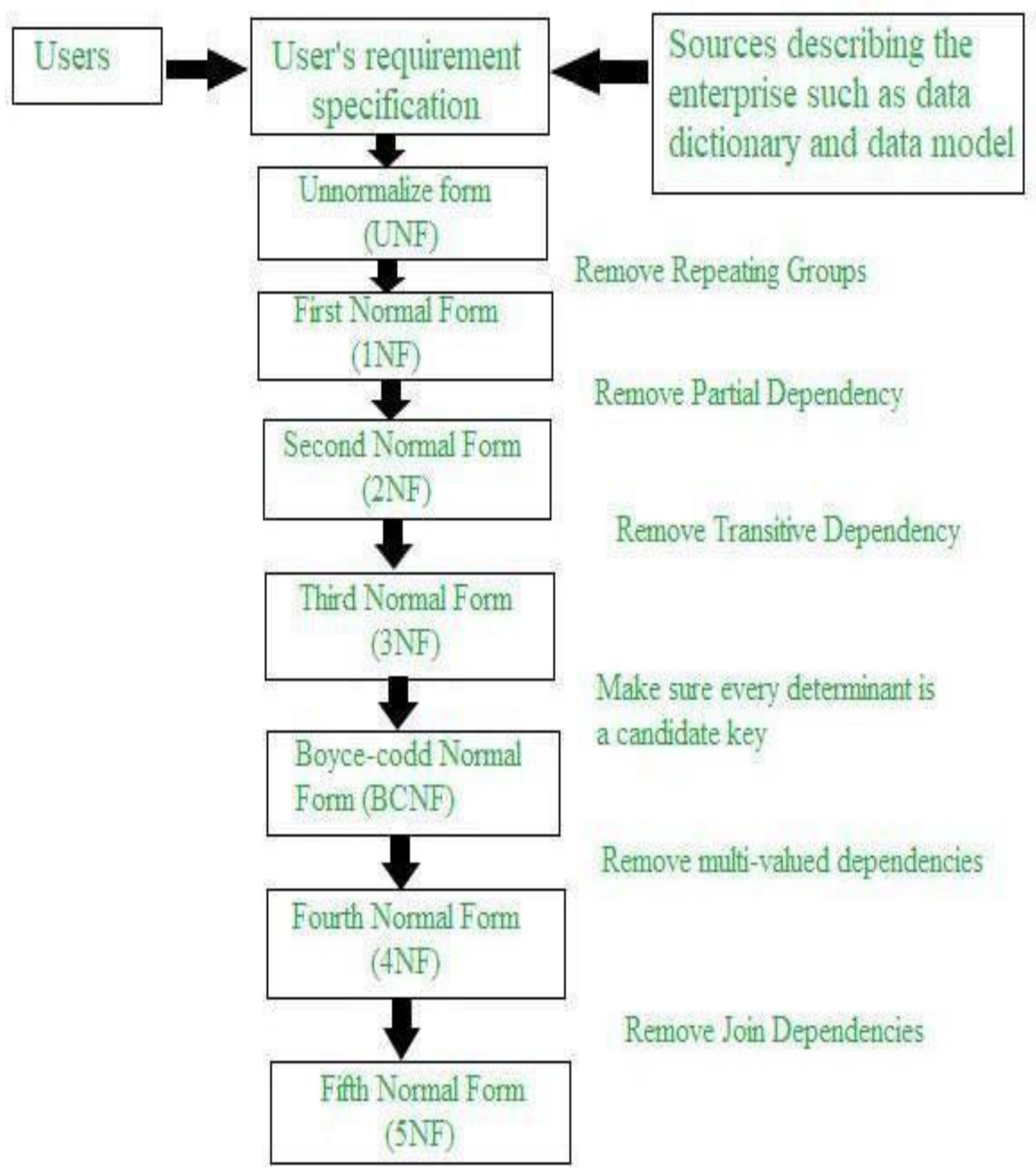
- **Loan availed by Customer => M : N**

  (Assume loan can be jointly held by many Customers).

  One Customer can have more than one Loans and also One Loan can be availed by one or more Customers, so the relationship between Loan and Customers is many to many relationship.

## NORMALIZATION PROCESS

**Database normalization** is a stepwise formal process that allows us to decompose database tables in such a way that both data dependency and update anomalies are minimized. It makes use of functional dependency that exists in the table and primary key or candidate key in analyzing the tables. Normal forms were initially proposed called First Normal Form (INF), Second Normal Form (2NF), and Third Normal Form (3NF). Subsequently, R, Boyce, and E. F. Codd introduced a stronger definition of 3NF called Boyce-Codd Normal Form. With the exception of 1NF, all these normal forms are based on functional dependency among the attributes of a table. Higher normal forms that go beyond BCNF were introduced later such as Fourth Normal Form (4NF) and Fifth Normal Form (5NF). However, these later normal forms deal with situations that are very rare.


Users
User's requirement specification
Sources describing the enterprise such as data dictionary and data model
Unnormalize form (UNF)
Remove Repeating Groups
First Normal Form (1NF)
Remove Partial Dependency
Second Normal Form (2NF)
Remove Transitive Dependency
Third Normal Form (3NF)
Make sure every determinant is a candidate key
Boyce-codd Normal Form (BCNF)
Remove multi-valued dependencies
Fourth Normal Form (4NF)
Remove Join Dependencies
Fifth Normal Form (5NF)

**TRIGGERS**

```
CREATE TRIGGER update_account AFTER INSERTON transactionsBEGIN
UPDATE accounts a SETa.balance=
   (CASE WHEN new.withdrawal=1 THEN a.balance-new.amount ELSE
a.balance+new.amountEND) WHERE a.id = new.accountID;
END;
```

pseudocode, Represents

- If the transaction is a deposit, add the money
- If the transaction is a withdrawal, check if it is discretionary
- If it is discretionary, remove from the balance and the allowance remaining
- If it is not, remove only from the balance.

**ACID properties in DBMS**

To ensure the **integrity and consistency of data** during a transaction (A transaction is a unit of program that updates various data items, read more about it here), the database system maintains **four properties**. These properties are widely known as **ACID properties.**

**Atomicity**

This property ensures that **either all the operations of a transaction reflect in database or none**. The logic here is simple, transaction is a single unit, it can't execute partially. Either it executes completely or it doesn't, there shouldn't be a partial execution.

Let's take an example of banking system to understand this:
Suppose Account A has a balance of 400$ & B has 700$. Account A is transferring 100$ to Account B.

**This is a transaction that has two operations**
a) Debiting 100$ from A's balance
b) Creating 100$ to B's balance.

Let's say first operation passed successfully while second failed, in this case A's balance would be 300$ while B would be having 700$ instead of 800$. This is unacceptable in a banking system. Either the transaction should fail without executing any of the operation or it should process both the operations. The Atomicity property ensures that.

There are **two key operations are involved** in a transaction to maintain the atomicity of the transaction.

**Abort:** If there is a failure in the transaction, abort the execution and rollback the changes made by the transaction.

**Commit:** If transaction executes successfully, commit the changes to the database.

**Consistency**

Database must be in consistent state **before and after the execution of the transaction**. This ensures that there are no errors in the database at any point of time. Application programmer is responsible for maintaining the consistency of the database.

**Example:**
A transferring 1000 dollars to B. A's initial balance is 2000 and B's initial balance is 5000.

**Before the transaction:**
Total of A+B = 2000 + 5000 = 7000$

**After the transaction:**
Total of A+B = 1000 + 6000 = 7000$

The data is consitendct before and after the execution of the transaction so this example maintains the consistency property of the database.

**Isolation**

A transaction **shouldn't interfere with the execution of another transaction**. To preserve the consistency of database, the execution of transaction should take place in isolation (that means no other transaction should run concurrently when there is a transaction already running).

For example account A is having a balance of 400$ and it is transferring 100$ to account B & Cboth. So we have two transactions here. Let's say these transactions run concurrently and both the transactions read 400$ balance, in that case the final balance of A would be 300$ instead of200$. This is wrong.

If the transaction were to run in isolation then the second transaction would have read the correct balance 300$ (before debiting 100$) once the first transaction went successful.

**Durability**

Once a transaction completes successfully, the **changes it has made into the database shouldbe permanent evenif there is a systemfailure**. The recovery-management component of database systems ensures the durability of transaction.

**STORED PROCEDURE**

```
CREATE PROCEDURE [bank].[GetTransactions]
        -- Add the parameters for the stored
        procedure here@AccountID int = 0,
        @StartDate
        datetime = 0,
        @EndDate
        datetime = 0
A
S

B
E
```

```
G
I
N
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT
        statements. SET NOCOUNT
        ON;

    -- Insert statements for
        procedure here SELECT
        * from
        bank.Transactions
    WHERE AccountID = @AccountID AND [Date] BETWEEN @StartDate AND
@EndDate END
```

Second, here's the EXEC statment:

```
EXEC
        bank.GetTran
        sactions
        @AccountID
        = 100000,
        @StartDate =
        '4/1/2007',
        @EndDate =
        '4/30/2007'
```