Ex. No:1 Data Definition Commands, Data Manipulation Commands for inserting, deleting, updating and retrieving Tables and Transaction Control statements. Database Querying – Simple queries, Nested queries, Sub queries and Joins

AIM:

To design and implement a database for manipulating & storing data items in MYSQL by using SQL commands and to implement and execute a query for manipulating & storing data items in mysql database using Structured Query Language commands.

DDL COMMANDS:

1) Create

Syntax:

Database: **Create database <database name>;**
Use <database name>;

Table:

Create table <table name> (column_name1 datatype1 constraints, column_name2 datatype2 constraint ..column_nameN datatypeN constraints);

Note: Constraints is optional

2) **ALTER**:

Syntax:

ADD:

Alter table<table name>add column_name1 datatype1 constraints;

MODIFY:

Alter table<table name>modify column_name1 datatype1;

DROP:

Alter table<table name>drop column_name;

RENAME:

Rename table <old table name>to<new table name>;

DROP:

Drop table<table name>;

DML COMMANDS:

INSERT:

Syntax 1: Insert into <table name> values (‘attributes1’, ’attributes2’……);
Syntax 2: Insert into<table name>(column names)values(list of data values);

SELECT:

Syntax 1: Select column name1,columnname2 from <table name>;
Syntax 2: select * from <tablename>;
Syntax 3: select * from <tablename> where <condition>;

UPDATE:

Syntax: Update <table name> set <column name>=’values’ where <condition>;

DELETE:

Syntax: Delete from <table name>;

TCL COMMANDS:

COMMIT:

Commit;

ROLLBACK:

Rollback to <savepoint>;

SAVEPOINT:

Savepoint <savepoint name>;

PROBLEM STATEMENT:

- A **branch** contains many **account**holders.
- A branch provides more than one **loan**.
- A loan can be availed by more than **customer**.
- A customer can get more than one loan.
- A customer can have more one account.
- An account can have more than one customer.

1. TABLE FROM THE PROBLEM STATEMENT:

1) Branch_m
2) Account_m
3) Loan_m
4) Customer_m

Database Name: it

```
mysql>create database ITAML;
mysql>use ITAML;
```

Table Name: Branch_m

```
mysql> create table branch_m(branch_name varchar(20) primary
key,branch_city varchar(20),asset int);

Query OK, 0 rows affected
mysql> desc branch_m;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| branch_name | varchar(20) | NO   | PRI | NULL    |       |
| branch_city | varchar(20) | YES  |     | NULL    |       |
| asset       | int(11)     | YES  |     | NULL    |       |
+-----------------+-----------------+--------+-------+------------+----------+
3 rows in set
```

Table name: Customer_m

```
mysql> create table customer_m(customer_id varchar(20) primary
key,customer_name varchar(20),customer_street
varchar(20),customer_city varchar(20));
Query OK, 0 rows affected

mysql> desc customer_m;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| customer_id     | varchar(20) | NO   | PRI | NULL    |       |
| customer_name   | varchar(20) | YES  |     | NULL    |       |
| customer_street | varchar(20) | YES  |     | NULL    |       |
| customer_city   | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set
```

Table name: Account_m

```
mysql> create table account_m(account_no varchar(20) primary
key,branch_name varchar(20),balance int,foreign key(branch_name)
references branch(branch_name));
Query OK, 0 rows affected

mysql> desc account_m;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| account_no  | varchar(20) | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| balance     | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set
```

Table name: Loan_m

```
mysql> create table loan_m(loan_no varchar(20) primary
key,branch_name varchar(20),amount int,foreign key(branch_name)
references branch(branch_name));

Query OK, 0 rows affected

mysql> desc loan_m;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| loan_no     | varchar(20) | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| amount      | int(11)     | YES  |     | NULL    |       |
+-----------------+-----------------+--------+-------+-----------+----------+
3 rows in set
```

2. Alter the table branch_m by increasing the field width of branch city to 25.

```
mysql> desc branch_m;
+-----------------+-----------------+--------+-------+-----------+----------+
| Field         | Type          | Null | Key | Default | Extra |
+-----------------+-----------------+--------+-------+-----------+----------+
| branch_name | varchar(20) | NO   | PRI | NULL    |       |
| branch_city | varchar(20) | YES  |     | NULL    |       |
| asset       | int(11)     | YES  |     | NULL    |       |
+-----------------+-----------------+--------+-------+-----------+----------+
3 rows in set
mysql> alter table branch_m modify branch_city varchar(25);

Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0
mysql> desc branch_m;
+-----------------+-----------------+--------+-------+-----------+----------+
| Field         | Type          | Null | Key | Default | Extra |
+-----------------+-----------------+--------+-------+-----------+----------+
| branch_name | varchar(20) | NO   | PRI | NULL    |       |
| branch_city | varchar(25) | YES  |     | NULL    |       |
| asset       | int(11)     | YES  |     | NULL    |       |
+-----------------+-----------------+--------+-------+-----------+----------+
3 rows in set
```

3. Drop the primary key from loan_m

```
mysql> desc loan_m;
+-----------------+-----------------+--------+-------+-----------+----------+
| Field         | Type          | Null | Key | Default | Extra |
+-----------------+-----------------+--------+-------+-----------+----------+
| loan_no     | varchar(20) | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| amount      | int(11)     | YES  |     | NULL    |       |
+-----------------+-----------------+--------+-------+-----------+----------+
3 rows in set
mysql> alter table loan_m drop primary key;

Query OK, 0 rows affected

Records: 0  Duplicates: 0  Warnings: 0
mysql> desc loan_m;
+-----------------+-----------------+--------+-------+-----------+----------+
| Field         | Type          | Null | Key | Default | Extra |
+-----------------+-----------------+--------+------+-----------+---------+
| loan_no     | varchar(20) | NO   |     | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| amount      | int(11)     | YES  |     | NULL    |       |
+-----------------+-----------------+--------+------+-----------+---------+
3 rows in set
```

4. Alter the primary key to loan_m

```
mysql> desc loan_m;
+-----------------+-----------------+--------+------+-----------+---------+
| Field       | Type        | Null | Key | Default | Extra |
+-----------------+-----------------+--------+------+-----------+---------+
| loan_no     | varchar(20) | NO   |     | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| amount      | int(11)     | YES  |     | NULL    |       |
+-----------------+-----------------+--------+------+-----------+---------+
3 rows in set (0.00 sec)
mysql> alter table loan_m add primary key(loan_no);
Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0
mysql> desc loan_m;
+-----------------+-----------------+--------+------+-----------+---------+
| Field       | Type        | Null | Key | Default | Extra |
+-----------------+-----------------+--------+------+-----------+---------+
| loan_no     | varchar(20) | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| amount      | int(11)     | YES  |     | NULL    |       |
+-----------------+-----------------+--------+------+-----------+---------+
3 rows in set
```

5. Add new column to loan_m

```
mysql> desc loan_m;
+-----------------+-----------------+--------+------+-----------+---------+
| Field       | Type        | Null | Key | Default | Extra |
+-----------------+-----------------+--------+------+-----------+---------+
| loan_no     | varchar(20) | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| amount      | int(11)     | YES  |     | NULL    |       |
+-----------------+-----------------+--------+------+-----------+---------+
3 rows in set
mysql> alter table loan_m add roi int;

Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0
mysql> desc loan_m;
+-----------------+-----------------+--------+------+-----------+---------+
| Field       | Type        | Null | Key | Default | Extra |
+-----------------+-----------------+--------+------+-----------+---------+
| loan_no     | varchar(20) | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| amount      | int(11)     | YES  |     | NULL    |       |
| roi         | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set
```

6. Drop the column from loan_m

```
mysql> desc loan_m;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| loan_no     | varchar(20) | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| amount      | int(11)     | YES  |     | NULL    |       |
| roi         | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set
mysql> alter table loan_m drop roi;
Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0
mysql> desc loan_m;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| loan_no     | varchar(20) | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| amount      | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set
```

7. Rename the customer_m as customer_ma

```
mysql> desc customer_m;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| customer_id     | varchar(20) | NO   | PRI | NULL    |       |
| customer_name   | varchar(20) | YES  |     | NULL    |       |
| customer_street | varchar(20) | YES  |     | NULL    |       |
| customer_city   | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set
mysql> rename table customer_m to customer_ma;
Query OK, 0 rows affected
mysql> desc customer_m;
ERROR 1146 (42S02): Table 'lab.customer' doesn't exist
mysql> desc customer_ma;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| customer_id     | varchar(20) | NO   | PRI | NULL    |       |
| customer_name   | varchar(20) | YES  |     | NULL    |       |
| customer_street | varchar(20) | YES  |     | NULL    |       |
| customer_city   | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set
```

8)a) Drop customer_ma

```
mysql> desc customer_ma;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| customer_id     | varchar(20) | NO   | PRI | NULL    |       |
| customer_name   | varchar(20) | YES  |     | NULL    |       |
| customer_street | varchar(20) | YES  |     | NULL    |       |
| customer_city   | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set

mysql> drop table customer_ma;
Query OK, 0 rows affected
mysql> desc customer_ma;
ERROR 1146 (42S02): Table 'lab.customer1' doesn't exist
```

8)b) Rename the column loanamount to amount from loan table.

```
mysql> create table loan_m(loan_no int primary key,branch_name
varchar(20),loanamount int);
Query OK, 0 rows affected (0.20 sec)
mysql> desc loan_m;
+-------------+-------------+------+-----+---------+-------
+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| loan_no     | int(11)     | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  |     | NULL    |       |
| loanamount  | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)

mysql> alter table loan_m change column loanamount amount int;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc loan_m;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+------------------+------------------+---------+-------+------------+----------+
| loan_no      | int(11)      | NO   | PRI | NULL     |         |
| branch_name | varchar(20) | YES  |     | NULL     |         |
| amount       | int(11)      | YES  |     | NULL     |         |
+------------------+------------------+---------+-------+------------+----------+
3 rows in set (0.00 sec)
```

9. INSERTRECORDS IN ALL THE FOUR CREATED TABLES:
Insert the values given below.(Branch Table)

BRANCH_NAME	BRANCH_CITY	ASSETS
Perryridge	Rye	50000
Downtown	Stamford	100000
Brighton	Paloalto	25000
Redwood	Harrison	150000
Mianus	Pitsfield	450000
Roundhill	Princeton	150000

```
mysql> desc branch_m;
+------------------+------------------+---------+-------+------------+----------+
| Field       | Type         | Null | Key | Default | Extra |
+------------------+------------------+---------+-------+------------+----------+
| branch_name | varchar(20) | NO   | PRI | NULL    |       |
| branch_city | varchar(25) | YES  |     | NULL    |       |
| asset       | int(11)     | YES  |     | NULL    |       |
+------------------+------------------+---------+-------+------------+----------+
3 rows in set
mysql> insert into branch_m values('perryridge','rye',50000);
Query OK, 1 row affected
```

Insert the values given below. (Loan Table)

LOAN	BRANCH_NAME	AMOUNT
1_11	Roundhill	900
1_14	Downtown	1500
1_15	Perryridge	1500
1_16	Perryridge	1300
1_17	Downtown	1000
1_23	Redwood	2000
1_93	Mianus	500
l_102	Mianus	Null

```
mysql> desc loan_m;
+------------------+------------------+---------+-------+------------+----------+
| Field       | Type         | Null | Key | Default | Extra |
+------------------+------------------+---------+-------+------------+----------+
| loan_no     | varchar(20) | NO   | PRI | NULL    |       |
| branch_name | varchar(20) | YES  | MUL | NULL    |       |
| amount      | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set
mysql> insert into loan_m values('l_23','redwood',2000);
Query OK, 1 row affected
```

Insert the values given below. (Customer Table)

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_STREET	CUSTOMER_CITY
c_01	Smith	north	rye
c_02	Turner	putnam	stamford
c_03	Johnson	alma	paloalto
c_04	Curry	north	rye
c_05	Jones	main	harrisdon
c_06	Adoms	spring	pittsfield
c_07	Lindsay	park	pittsfeild
c_08	Hayes	main	harrison
c_09	Williams	nassu	princeton

```
mysql> desc customer_m;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| customer_id     | varchar(20) | NO   | PRI | NULL    |       |
| customer_name   | varchar(20) | YES  |     | NULL    |       |
| customer_street | varchar(20) | YES  |     | NULL    |       |
| customer_city   | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set
mysql>insert into customer_m
values('c_01','smith','north','rye');
Query OK, 1 row affected
```

Insert the values given below. (Account Table)

ACCOUNT_NO	BRANCH_NAME	BALANCE
019_28_3746	Perryridge	1500
182_73_6091	Downtown	1800
192_83_7465	Brighton	500
321_12_3123	Redwood	2300
336_96_9999	Mianus	500
963_96_3963	Roundhill	500
376_66_9999	Mianus	900
963_96_3964	Mianus	1300

```
mysql>desc account_m;
+-------------+-------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| account_no   | varchar(20)  | NO   | PRI | NULL    |       |
| branch_name  | varchar(20)  | YES  | MUL | NULL    |       |
| balance      | int(11)      | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
3 rows in set
mysql>insert into account_mvalues(019_28_3746,'perryridge',1500);
Query ok,1 row affected.
```

10. Find the names of all branches in loan relation.

```
mysql> select branch_name from loan_m;
+--------------+
| branch_name |
+--------------+
| downtown     |
| downtown     |
| mianus       |
| perryridge   |
| perryridge   |
| redwood      |
| roundhill    |
+--------------+
7 rows in set
```

11. Find the names of all branches in loan relation eliminate duplicate.

```
mysql> select distinct branch_name from loan_m;
+--------------+
| branch_name |
+--------------+
| downtown     |
| mianus       |
| perryridge   |
| redwood      |
| roundhill    |
+--------------+
5 rows in set
```

12.Updatethe customer city stamford to rye in customerrelation.

```
mysql> update customer_m set customer_city='rye' where
customer_city='stamford';

Query OK, 2 rows affected
Rows matched: 2  Changed: 2  Warnings: 0
```

NESTED and SUB QURIES:

SELECT column_name(s) FROM table_name WHERE condition OPERATOR (SELECT column_name(s) FROM table_name);

SIMPLE QUERIES

1. Display the loan relation with attributes amount multiplied by 100.

```
mysql> select amount*100 from loan_m;
+------------------+
| amount*100 |
+------------------+
|      90000 |
|     150000 |
|     150000 |
|     130000 |
|     100000 |
|     200000 |
|      50000 |
+------------------+
7 rows in set
```

2. Findall numbers for loan made at perryridge branch with loan amount greater than 1400.

```
mysql> select loan_no from loan_m where branch_name='perryridge'
and amount>1400;
+-------------+
| loan_no |
+-------------+
| l_15    |
+-------------+
1 row in set
```

3. Findall loan numbers for loan's with loan amount between 900 and 1500.

```
mysql> select loan_no from loan_m where amount between 900 and
1500;
+-------------+
| loan_no |
+-------------+
| l_11    |
| l_14    |
| l_15    |
| l_16    |
| l_17    |
+-------------+
5 rows in set
```

4. Findthe names of customer of customer whose street names includes the character r in the third position.

```
mysql> select customer_name from customer_m where customer_street
like'__r%';
+---------------------+
| customer_name |
+---------------------+
| smith               |
| curry               |
| adoms               |
+---------------------+
3 rows in set (0.00 sec)
```

5. Find the names of customer_m whose name starts with w ends with s.

```
mysql> select customer_name from customer_m where customer_name
like'w%s';

+---------------------+
| customer_name |
+---------------------+
| williams      |
+---------------------+
1 row in set
```

6. Displaythe entire loan relation in descending order ofamount.

```
mysql> select * from loan_m order by amount desc;
+-------------+------------------+-----------+
| loan_no | branch_name | amount |
+-------------+------------------+-----------+
| l_23    | redwood     |   2000 |
| l_14    | downtown    |   1500 |
| l_15    | perryridge  |   1500 |
| l_16    | perryridge  |   1300 |
| l_17    | downtown    |   1000 |
| l_11    | roundhill   |    900 |
| l_24    | mianus      |    500 |
+-------------+------------------+-----------+
7 rows in set
```

7. Findtotal number of customer.

```
mysql> select count(customer_id) from customer_m;
+----------------------------+
| count(customer_id) |
+----------------------------+
|                  6 |
+----------------------------+
1 row in set
```

8. Findall the loan number that appears in the loan relationwith NULL values for amount.

```
mysql> select loan_no from loan_m where amount is null;
+-------------+
| loan_no |
+-------------+
| l_102   |
+-------------+
1 row in set
```

9.
```
mysql> select * from customer_m where customer_id='c_02';

+-----------------+--------------------+----------------------+---------------------+
| customer_id | customer_name | customer_street | customer_city |
+-----------------+--------------------+----------------------+---------------------+
| c_02        | turner        | putnam          | rye           |
+-----------------+--------------------+----------------------+---------------------+
1 rows in set (0.00 sec)
```

NESTED and SUB QUERIES:

1. Find all the name of all branches that have assets greater than atleast one bank located in Stanford. (Nested queries)

```
mysql>Select branch_name from branch_ma where assets >ANY
(select assets from branch_ma where branch_city='stanford');

 BRANCH_NAME
-----------------------------------------
Redwood
Mianus
Roundhill
```

2. Display the entire customer name in alphabetical order that have loan in Perryridge.(Nested queries).

```
mysql>Select customer_name from customer_ma where customer_id=ANY
( select customer_id from borrow_ma where borrow_ma.loan_no=ANY(
select loan_no from loan_ma where branch_name='Perryridge'))
order by  customer_ma.customer_name asc;

CUSTOMER_NAME
------------------------------
Johnson
```

3. Find all customer having both account and loan at same branch.(Nested queries)

```
mysql>Select depositor_ma.customer_id from depositor_ma where
customer_id IN (select borrow_ma.customer_id from borrow_ma);
CUSTOMER_ID
------------------------------
c_03
c_05
```

4. Find all customers who have loan at bank but who don't have account at same branch.(Nested queries).

```
mysql>Select borrow_ma.customer_id from borrow_ma where customer_id
NOT IN (select depositor_ma.customer_id from depositor_ma);
```

```
CUSTOMER_ID
-----------------------------------
c_01
c_01
```

5. Find the name of all branches that have assets greater than those atleast one branch located in Harrison.(Nested queries).

```
mysql>Select branch_name from branch_ma where assets>any(select
assets from branch_ma where branch_city='Harrison');

BRANCH_NAME
-----------------------------------------
Mianus
```

JOIN COMMANDS

- ❖ INNER JOIN command returns the matching rows from the tables that are being joined.

  SELECT column_name(s) FROM table_name1 INNER JOIN table_name2 ON table_name1.column_name=table_name2.column_name

- ❖ LEFT OUTER JOIN command returns matching rows from the tables being joined and also non-matching row from the left table in the result and places null values in the attributes that come from the right side table.

  SELECT column_name(s) FROM table_name1 LEFT JOIN table_name2 ON table_name1.column_name=table_name2.column_name

- ❖ RIGHT OUTER JOIN command returns matching rows from the tables being joined and also non-matching row from the right table in the result and places null values in the attributes that come from the left side table.

  SELECT column_name(s) FROM table_name1 RIGHT JOIN table_name2 ON table_name1.column_name=table_name2.column_name

Table: Employee

mysql> create table employee (Employee_Name varchar(10),Employee_no int primary key, Dept_no int,Dept_name varchar(10));

Query OK, 0 rows affected (0.01 sec)

mysql> desc employee;

```
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Employee_Name | varchar(10) | YES  |     | NULL    |       |
| Employee_no   | int(11)     | NO   | PRI | NULL    |       |
| Dept_no       | int(11)     | YES  |     | NULL    |       |
| Dept_name     | varchar(10) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

4 rows in set (0.00 sec)

==

Table: Employee1

```
mysql> create table employee1 ( Employee_Name varchar(10),Employee_no int primary key,Dept_no
int,dept_name varchar(10));
Query OK, 0 rows affected (0.00 sec)
mysql> desc employee1;

+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Employee_Name | varchar(10) | YES  |     | NULL    |       |
| Employee_no   | int(11)     | NO   | PRI | NULL    |       |
| Dept_no       | int(11)     | YES  |     | NULL    |       |
| dept_name     | varchar(10) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

==

INSERTING RECORDS IN ALL THE CREATED TABLES:

```
mysql> insert into employee values('Ajitha',234,45,'CSE');
Query OK, 1 row affected (0.00 sec)
mysql> insert into employee values('Vijaya',109,85,'EEE');
Query OK, 1 row affected (0.00 sec)
mysql> insert into employee values('Kala',132,95,'MECH');
Query OK, 1 row affected (0.00 sec)
mysql> select * from employee;
+---------------+-------------+---------+-----------+
| Employee_Name | Employee_no | Dept_no | Dept_name |
+---------------+-------------+---------+-----------+
| Ajitha        |         234 |      45 | CSE       |
| Vijaya        |         109 |      85 | EEE       |
| Kala          |         132 |      95 | MECH      |
+---------------+-------------+---------+-----------+
mysql> insert into employee1 values('Ajitha',234,45,'CSE');
Query OK, 1 row affected (0.00 sec)
mysql> insert into employee1 values('Vishnu',476,55,'IT');
Query OK, 1 row affected (0.00 sec)
mysql> insert into employee1 values('Vikram',985,75,'ECE');
Query OK, 1 row affected (0.00 sec)
mysql> select * from employee1;
+---------------+-------------+---------+-----------+
| Employee_Name | Employee_no | Dept_no | dept_name |
+---------------+-------------+---------+-----------+
| Ajitha        |         234 |       45 | CSE       |
| Vishnu        |         476 |       55 | IT        |
| Vikram        |         985 |       75 | ECE       |
+---------------+-------------+----------+-----------+
```
3 rows in set (0.00 sec)

JOIN COMMANDS

```
mysql> alter table employee1 add foreign key (employee_no) references employee1 ( employee_no);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```
==

INNER JOIN

```
mysql> select e.employee_name,d.dept_no from employee e,employee1 d where
e.employee_no=d.employee_no;
+---------------+---------+
| employee_name | dept_no |
+---------------+---------+
| Ajitha        |      45 |
+---------------+---------+
1 row in set (0.00 sec)
```

LEFT OUTER JOIN

```
mysql> select e.dept_name,d.dept_no from employee e left join employee1 d on e.employee_no =
d.employee_no;
+-----------+---------+
| dept_name | dept_no |
+-----------+---------+
| CSE       |      45 |
| EEE       |    NULL |
| MECH      |    NULL |
+-----------+---------+
3 rows in set (0.01 sec)
```

RIGHT OUTER JOIN

```
mysql> select e.dept_name,d.dept_no from employee e right join employee1 d  on e.employee_no =
d.employee_no;
+-----------+---------+
| dept_name | dept_no |
+-----------+---------+
| CSE       |      45 |
| NULL      |      55 |
| NULL      |      75 |
+-----------+---------+
3 rows in set (0.00 sec)
```

RESULT:

The design and implementation of a database for manipulating & storing data items in MYSQL by using manipulating database using Structured Query Language commands was created successfully.

Ex. No: 2 **Queries using Aggregate functions (COUNT, SUM, AVG, MAX and MIN), GROUP BY, HAVING and Creation and dropping of Views, Synonyms, Sequences**

AIM:

To implement and execute queries using Aggregate functions and to create and drop view, sequence,indexes,savepoint in MYSQL using SQL commands.

Aggregate Functions:

1. COUNT FUNCTION

COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.

COUNT function uses the COUNT(*) that returns the count of all the rows in a specified table. COUNT(*) considers duplicate and Null.

Syntax

COUNT(*) or COUNT([ALL|DISTINCT] expression)

Sample table:

PRODUCT_MAST

PRODUCT	COMPANY	QTY	RATE	COST
Item1	Com1	2	10	20
Item2	Com2	3	25	75
Item3	Com1	2	30	60
Item4	Com3	5	10	50
Item5	Com2	2	20	40
Item6	Cpm1	3	25	75
Item7	Com1	5	30	150
Item8	Com1	3	10	30
Item9	Com2	2	25	50
Item10	Com3	4	30	120

Example: COUNT()

SELECT COUNT(*) FROM PRODUCT_MAST;

Output: 10

Example: COUNT with WHERE

SELECT COUNT(*) FROM PRODUCT_MAST; WHERE RATE>=20;

Output: 7

Example: COUNT() with DISTINCT

SELECT COUNT(DISTINCT COMPANY) FROM PRODUCT_MAST;

Output: 3

Example: COUNT() with GROUP BY

SELECT COMPANY, COUNT(*) FROM PRODUCT_MAST GROUP BY COMPANY;

Output:

```
Com1    5
Com2    3
Com3    2
```

Example: COUNT() with HAVING

SELECT COMPANY, COUNT(*) FROM PRODUCT_MAST GROUP BY COMPANY HAVING COUNT(*)>2;

Output:

```
Com1    5
Com2    3
```

2. SUM Function

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

Syntax

SUM()

or

SUM([ALL|DISTINCT] expression)

Example: SUM()

SELECT SUM(COST) FROM PRODUCT_MAST;

Output:

```
670
```

Example: SUM() with WHERE

SELECT SUM(COST) FROM PRODUCT_MAST WHERE QTY>3;

Output:

```
320
```

Example: SUM() with GROUP BY

SELECT SUM(COST) FROM PRODUCT_MAST WHERE QTY>3 GROUP BY COMPANY;

Output:

```
Com1    150
Com2    170
```

Example: SUM() with HAVING

SELECT COMPANY, SUM(COST) FROM PRODUCT_MAST GROUP BY COMPANY HAVING SUM(COST)>=170;

Output:

```
Com1    335
Com3    170
```

3. AVG function

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

Syntax
AVG() or AVG([ALL|DISTINCT] expression)
Example:
SELECT AVG(COST) FROM PRODUCT_MAST;
Output:
67.00

4. MAX Function
MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

Syntax
MAX() or MAX([ALL|DISTINCT] expression)
Example: SELECT MAX(RATE) FROM PRODUCT_MAST;
Output: 30

5. MIN Function
MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.
Syntax
MIN() or MIN([ALL|DISTINCT] expression)
Example: SELECT MIN(RATE) FROM PRODUCT_MAST;
Output: 10
VIEWS:
Create view <viewname> as any select query; (Ex: select * from <tablename> where <condition>)

SEQUENCE:
Create table <tablename> (column1 datatype1primary key auto_increment,column2 datatype 2……columnN datatypeN)

SYNONYM:
Create synonym<synonym name> for <table name>;
VIEWS:
Create a view using aggregate functions to calculate the age of the Person

```
mysql> create table person_m(name varchar(20) primary
key,dob date,person_city varchar(20));
Query OK, 0 rows affected
mysql> desc person_m;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| name        | varchar(20) | NO   | PRI | NULL    |       |
| dob         | date        | YES  |     | NULL    |       |
| person_city | varchar(20) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set
```

```
mysql>   insert   into   person_m   values('abdulkalam','1931-08-
18','rameshwaram');
Query OK, 1 row affected
mysql> select * from person_m;
+----------------+------------+-------------+
| name           | dob        | person_city |
+----------------+------------+-------------+
| abdulkalam     | 1931-10-15 | rameshwaram |
| maheshboobathy | 1974-07-02 | chennai     |
| sachin         | 1973-05-04 | mumbai      |
| viswanathanand | 1970-03-06 | chennai     |
+----------------+------------+-------------+
4 rows in set
mysql>      create      view      personage_m      as      select
name,datediff(sysdate(),dob)/365.25 as age from person_m;
Query OK, 0 rows affected
mysql> select * from personage_m;
+----------------+---------+
| name           | age     |
+----------------+---------+
| abdulkalam     | 83.9726 |
| maheshboobathy | 41.1006 |
| sachin         | 42.2615 |
| viswanathanand | 45.4237 |
+----------------+---------+
4 rows in set

mysql> insert into person_m values('jorden','1970-07-08','usa');
Query OK, 1 row affected (0.06 sec)
mysql> select * from personage_m;
+----------------+---------+
| name           | age     |
+----------------+---------+
| abdulkalam     | 83.9726 |
| jorden         | 45.0842 |
| maheshboobathy | 41.1006 |
| sachin         | 42.2615 |
| viswanathanand | 45.4237 |
+----------------+---------+
5 rows in set (0.00 sec)
```

SEQUENCE:

Create a sequence and design the department table in the given attribute.

```
mysql> create table department_m(department_id int primary key
auto_increment,department_name varchar(20));
Query OK, 0 rows affected (0.14 sec)
mysql> desc department_m;
```

```
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| department_id   | int(11)     | NO   | PRI | NULL    | auto_increment |
| department_name | varchar(20) | YES  |     | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
mysql> insert into department_m(department_name)values('it');
Query OK, 1 row affected (0.08 sec)
mysql> insert into department_m(department_name)values('aml');
Query OK, 1 row affected (0.08 sec)
mysql>
insert into department_m(department_name)values('mechanical');
Query OK, 1 row affected (0.06 sec)
mysql> insert into department_m(department_name)values('aero');
Query OK, 1 row affected (0.07 sec)
mysql> select * from department_m;
+---------------+-----------------+
| department_id | department_name |
+---------------+-----------------+
|             1 | it              |
|             2 | aml             |
|             3 | mechanical      |
|             4 | aero            |
+---------------+-----------------+
4 rows in set (0.00 sec)
```

SYNONYM(UsingOracle)

CREATING A SYNONYM FOR A TABLE

CREATE TABLE product_m (product_nameVARCHAR2(25) PRIMARY KEY, product_price NUMBER(4,2), quantity_on_hand NUMBER(5,0), last_stock_date DATE);
Table created.

AFTER INSERTING THE RECORDS TO PRODUCT TABLE

```
SQL> SELECT * FROM product_m;
PRODUCT_NAME          PRODUCT_PRICE QUANTITY_ON_HAND LAST_STOC
--------------------  ---------------------------------  ----------
Product 1                  99             1              15-JAN-03
Product 2                  75             1000           15-JAN-02
Product 3                  50             100            15-JAN-03
Product 4                  25             10000          14-JAN-03
Product 5                 9.95            1234           15-JAN-04
Product 6                  45             1              31-DEC-08
6 rows selected.
```

SQL> **SELECT** * **FROM** prod_m;

```
SELECT * FROM prod_m
             *
ERROR at line 1:
ORA-00942: table or view does not exist
SQL> CREATE SYNONYM prod_m FOR product_m;
Synonym created.

SQL> SELECT * FROM prod_m;
PRODUCT_NAME              PRODUCT_PRICE QUANTITY_ON_HAND LAST_STOC
-------------------------------------------------------- ---------
Product 1                           99              1    15-JAN-03
Product 2                           75           1000    15-JAN-02
Product 3                           50            100    15-JAN-03
Product 4                           25          10000    14-JAN-03
Product 5                         9.95           1234    15-JAN-04
Product 6                           45              1    31-DEC-08
SQL> drop SYNONYM prod_m;
Synonym dropped.


SQL> drop table product_m;
Table dropped.
```

RESULT:

The SQL commands for view, sequence, indexes, savepoint and to execute queries using Aggregate functions were created successfully.

EX.NO.3 CONCEPTUAL DESIGNING USING ER DIAGRAMS

AIM: To design a database using ER modeling and identify the entities, attributes, keys and relationships between entities, cardinalities, generalization, specialization etc.

Problem Statement: ER Diagram

- A College is conducting a sports meet.
- Teams from recognized colleges are allowed.
- A team should have the players of same college.
- A player can play for more than one team.
- Events occurs in various grounds in the college.
- Winning teams receive awards.
- A captain is a player of a team.
- A player is a student of a college.
- Many teams can play a game.
- A game takes place in a ground.
- A college can have many teams.
- Only first two teams are awarded.

IDENTIFICATION OF ENTITY:

- ✓ COLLEGE
- ✓ PLAYERS
- ✓ TEAMS
- ✓ GAMES
- ✓ GROUND
- ✓ AWARDS

DESCRIPTION ABOUT ENTITY:

ENTITYNAME	TYPE	NOTATION
COLLEGE	Strong	▬
PLAYERS	Strong	▬
TEAMS	Strong	▬
GAMES	Strong	▬
GROUND	Strong	▬
AWARDS	Weak	▭

ATTRIBUTES:

- ✓ COLLEGE – CID, Name, Address line1, Address line2
- ✓ PLAYERS – FN, LN, POS, DOB, GENDER, PID
- ✓ TEAM - TID, Name, NOP, Rank, Team
- ✓ GAMES - GID, Name
- ✓ GROUND - ID, Name, Area
- ✓ AWARDS – Name, Position, Prize, Team

DESCRIPTION ABOUT ATTRIBUTES:

COLLEGE

ATTRIBUTE NAME	TYPE	NOTATION
CID	Single	⬬
Name	Single	⬬
Address line 1	Composite	⬬
Address line2	Composite	⬬

PLAYERS

ATTRIBUTE NAME	TYPE	NOTATION
PID	Single	⬬
FN	Single	⬬
LN	Single	⬬
Position	Single	⬬
Age	Derived	⬬
Gender	single	⬬

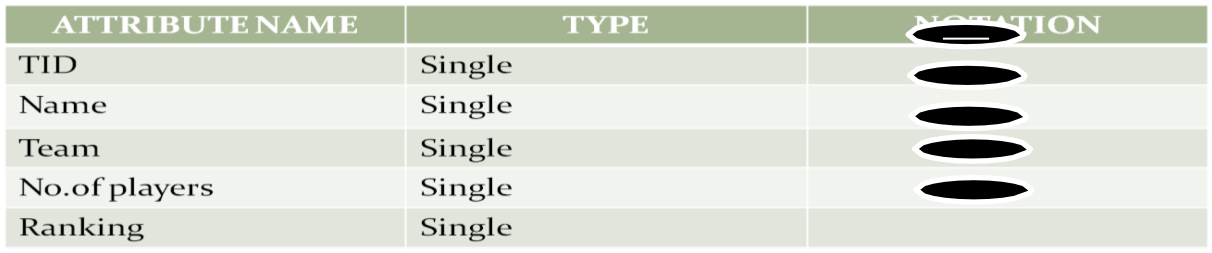
AWARDS

ATTRIBUTE NAME	TYPE	NOTATION
Name	Single	⬬
Position	Single	⬬
Prize	Single	⬬
Team	Single	⬬

GROUND

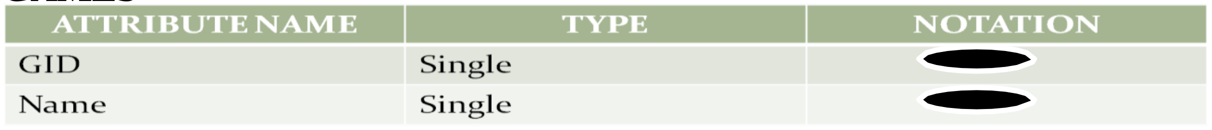
ATTRIBUTE NAME	TYPE	NOTATION
ID	Single	⬬
Name	Single	⬬
Area	single	⬬

TEAMS



ATTRIBUTE NAME	TYPE	NOTATION
TID	Single	⬬
Name	Single	⬬
Team	Single	⬬
No.of players	Single	⬬
Ranking	Single	

GAMES



ATTRIBUTE NAME	TYPE	NOTATION
GID	Single	⬬
Name	Single	⬬

RELATIONSHIP:

BINARY:

- Plays
- Has
- Student of
- Receives
- Takes place

ATTRIBUTES IN THE RELATIONSHIP:

- ✓ Student of – CID, PID
- ✓ Has – TID, CID
- ✓ Plays – TID, GID
- ✓ Takes place – GID, ID
- ✓ Receives – TID, Position

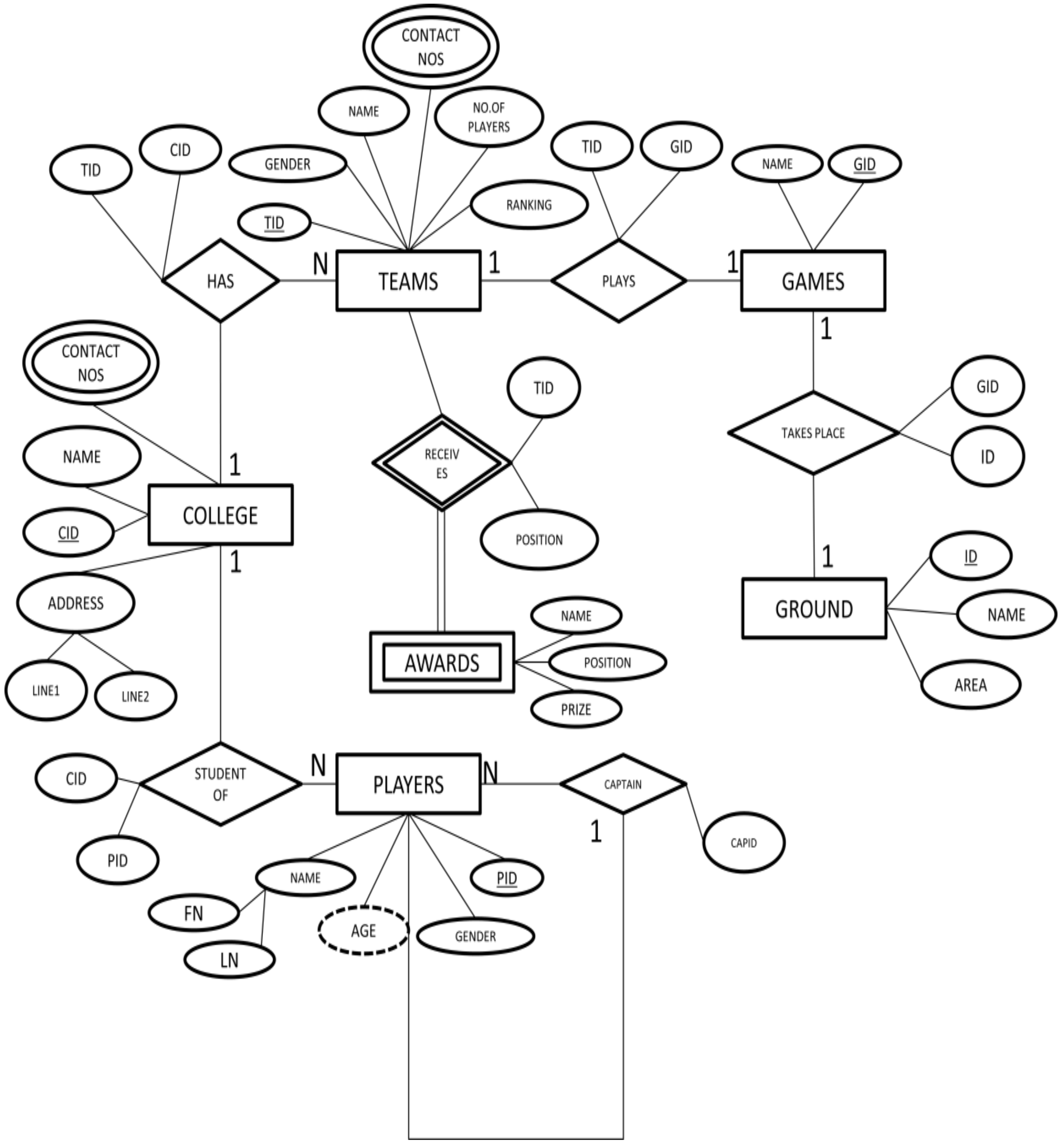
CARDINALITY AND RELATIONSHIP:

- ➢ ONE TO ONE:Plays, Takes place
- ➢ MANY TO ONE:Student of
- ➢ ONE TO MANY:has

CARDINALITY ABOUT RELATIONSHIP:

- ✓ PLAYERS STUDENT OF COLLEGE.
- ✓ MANY TEAMS PLAY A GAME.
- ✓ A GAME TAKES PLACE IN A GROUND.
- ✓ A COLLEGE HAS MANY TEAMS.

ER DIAGRAM:



RESULT:

A database was designed using ER modeling and identified the entities, attributes, keys and relationships between entities, cardinalities, generalization, specialization etc.

EX.NO.4 SIMPLE EMBEDDED SQL PROGRAM

AIM:

To write a Simple Embedded SQL Program.

EXEC SQL BEGIN DECLARE SECTION;

int STD_ID;

char STD_NAME [15];

char ADDRESS[20];

EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT * FROM STUDENT WHERE STUDENT_ID =:STD_ID;

The following code is a simple embedded SQL program, written in C. The program illustrates many, but not all, of the embedded SQL techniques. The program prompts the user for an order number, retrieves the customer number, salesperson, and status of the order, and displays the retrieved information on the screen.

```
int main() {
EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
int OrderID; /* Employee ID(fromuser) */

int CustID; /* Retrievedcustomer ID */
char SalesPerson[10] /* Retrievedsalespersonname */
char Status[6] /* Retrievedorderstatus */
EXEC SQL END DECLARE SECTION;
/* Set up error processing */
EXEC SQL WHENEVER SQLERROR GOTO query_error;
EXEC SQL WHENEVER NOT FOUND GOTO
bad_number;
/* Prompt the user for order number */
printf ("Enter order number: ");
scanf_s("%d", &OrderID);
/* Execute the SQL query */
EXEC SQL SELECT CustID, SalesPerson, Status FROM Orders
WHERE OrderID = :OrderID  INTO :CustID, :SalesPerson, :Status;
/* Display the results */
printf ("Customer number: %d\n", CustID);
printf ("Salesperson: %s\n", SalesPerson);
printf ("Status: %s\n", Status);
exit();
query_error:
printf ("SQL error: %ld\n", sqlca->sqlcode); exit(); bad_number:
printf ("Invalid order number.\n");
exit();
}
```

OUTPUT:

```
Enter order number: 123
Customer number: 121
Salesperson: Ramesh
Status: Success
```

RESULT:

] A Simple Embedded SQL Program was written and executed successfully.

Ex.No.5 DATABASE DESIGN USING NORMALIZATION AND IMPLEMENTATION

AIM:

To design a database using normalization and Implementation for any application.

Problem Statement: Normalization

Create a college database that contains studentid, studentname, studentcity, date of birth, course id, course name, duration of the course, marks and grade and their relationships. The requirements are listed below:

- A college can offer one or more courses.
- A student can enroll in one or more courses.
- Courses can be taken by one or more students.
- A student can have student_id, student_name, date _of _birth and student_city.
- A student belongs to one city.
- A city can have one or more students.
- A course can have course_id, course_name and duration.
- When a student finishes the course, a grade and marks are awarded.
- Grades are calculated based on the marks

Below 45 – U, 45-50 – E, 50-60– D, 60-70 – C, 70-80 – B, 80-90 – A, 90-100 –S

FIRST NORMAL FORM

A relation is said to be in first normal form if and only if

STUDENT ID	STUDENT NAME	STUDENT CITY	DOB	COURSE ID	COURSE NAME	DURATION	MARKS	GRADE

*All the attributes in the relation must be atomic in nature.

*No multivalued and composite attributes in the table

In a given table there is no multivalued and composite attributes, so it is satisfying normal form1

SECOND NORMAL FORM

A relation is said to be in second normal form if and only if

*It is in the first normal form and

***No partial dependencies** exist between non-key attributes and key attributes.

From Requirements: (studentid, courseid is Composite Primarykey)

studentid,courseid	$\rightarrow$	studentname	Partial Functional dependencies.
studentid,courseid	$\rightarrow$	studentcity	
studentid,courseid	$\rightarrow$	dob	
studentid,**courseid**	$\rightarrow$	coursename	
studentid,**courseid**	$\rightarrow$	duration	
studentid,courseid	$\rightarrow$	marks	Full Functional dependencies
studentid,coursed	$\rightarrow$	grade	

After removing partial functional dependencies from above table
STUDENT

STUDENTID	STUDENTNAME	STUDENTCITY	DOB

COURSE

COURSEID	COURSENAME	DURATION

RESULT

STUDENTID	COURSEID	MARKS	GRADE

THIRD NORMAL FORM

A relation is said to be in the third normal form if and only if

*it is in Second Normal Form
***No transitive dependency** exists between non-key attributes and key attribute

studentid,courseid	$\rightarrow$	marks	
marks	$\rightarrow$	grade	Transitive dependency
studentid,courseid	$\rightarrow$	grade	

After removing transitive dependency from above table
STUDENT

STUDENTID	STUDENTNAME	STUDENTCITY	DOB

COURSE

COURSEID	COURSENAME	DURATION

MARKS

MARKID	RANGE1	RANGE2

RESULT

STUDENTID	COURSEID	MARKID

RESULT:

The database was designed using normalization and implemented for college database.

Ex.No.6 USAGE OF TRANSACTION CONTROL LANGUAGE COMMANDS LIKE COMMIT,ROLLBACK AND SAVE POINT.

AIM:

To design a database to understand the usage of Transaction control language commands like commit, rollback and save point.

Transactional Control Commands

Transactional control commands are only used with the DML Commands such as - INSERT, UPDATE and DELETE only. They cannot be used while creating tables or dropping them because these operations are automatically committed in the database.

- **The COMMIT Command**

The COMMIT command is the transactional command used to save changes invoked by a transaction to the database. The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command.
The syntax for the COMMIT command is as follows.
COMMIT;

- **The ROLLBACK Command**

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database. This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.
The syntax for a ROLLBACK command is as follows −
ROLLBACK;

- **The SAVEPOINT Command**

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.

The syntax for a SAVEPOINT command is as shown below.
SAVEPOINT SAVEPOINT_NAME;

This command serves only in the creation of a SAVEPOINT among all the transactional statements. The ROLLBACK command is used to undo a group of transactions.
The syntax for rolling back to a SAVEPOINT is as shown below.
ROLLBACK TO SAVEPOINT_NAME;

```
mysql> create table students(regno int,name varchar(20))ENGINE=InnoDB;
Query OK, 0 rows affected (0.00 sec)
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)
mysql> insert into students values(11,'alma');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> savepoint s1;
Query OK, 0 rows affected (0.00 sec)
mysql> insert into students values(12,'akash');
Query OK, 1 row affected (0.00 sec)
mysql> savepoint s2;
Query OK, 0 rows affected (0.00 sec)
mysql> select * from students;
+-------+-------+
| regno | name  |
+-------+-------+
|    11 | alma  |
|    12 | akash |
+-------+-------+
2 rows in set (0.00 sec)
mysql> insert into students values(13,'delfi');
Query OK, 1 row affected (0.00 sec)
mysql> select * from students;
+-------+-------+
| regno | name  |
+-------+-------+
|    11 | alma  |
|    12 | akash |
|    13 | delfi |
+-------+-------+
3 rows in set (0.00 sec)
mysql> rollback to s1;
Query OK, 0 rows affected (0.00 sec)
mysql> select * from students;
+------+------+
| regno | name |
+-------+------+
|    11 | alma |
+-------+------+
Query OK, 0 rows affected (0.00 sec)
mysql> select * from students;
Empty set (0.00 sec)
mysql> insert into students values(17,'shanthini');
Query OK, 1 row affected (0.00 sec)
mysql> commit;
```

RESULT:

A database was designed to understand the usage of Transaction control language commands like commit rollback and save point.

Ex.No.7 TRIGGERS

AIM:
To implement and execute triggers and functions in Mysql using Procedural Language concepts.

TRIGGERS:

1) Trigger is a special type of procedure that the oracle executes when an insert, modify or delete operation is performed against a given table**.**
2) It is a stored sub program associated with a table.
3) It is used to keep an audit trial of a table, to prevent invalid transaction, enforce complex security authorization, to generate data automatically.

SYNTAX FOR TRIGGER

```
CREATE TRIGGER <TRIGGER NAME>
{BEFORE/AFTER} {INSERT/UPDATE/DELETE} ON <TABLENAME>
REFRENCECING {OLD AS OLD /NEW AS NEW}
[FOR EACH STATEMENT /FOR EACH ROW [WHEN <CONDITION>]] DECLARE
Variable declaration Constant
declaration
BEGIN
            PL/SQL Sub program body.
END;
```

<u>CREATE A TRIGGER TO DISPLAY THE RESULT OF A STUDENT IN THE STUDENT TABLE</u>

mysql> create table stud_tr(name varchar(10),regno int primary key,rollno varchar(10),total_cgpa int);

```
Query OK, 0 rows affected (0.01 sec)
mysql> select * from stud_tr;
+---------+-------+--------+------------+
| name    | regno | rollno | total_cgpa |
+---------+-------+--------+------------+
| Ashwini |   101 | 13CS11 |          8 |
| Asha    |   102 | 13CS12 |          7 |
+---------+-------+--------+------------+
2 rows in set (0.00 sec)
```

==

mysql> create table studt_pub(id int primary key auto_increment,rollno int not null,name varchar(15) not null,examres datetime default null,result varchar(10) default null);

```
Query OK, 0 rows affected (0.00 sec)
mysql> select * from studt_pub;
Empty set (0.00 sec)
```

```
mysql> desc studt_pub;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| id      | int(11)     | NO   | PRI | NULL    | auto_increment |
| rollno  | int(11)     | NO   |     | NULL    |                |
| name    | varchar(15) | NO   |     | NULL    |                |
| examres | datetime    | YES  |     | NULL    |                |
| result  | varchar(10) | YES  |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
=========================================================================
```

TRIGGER

```
mysql> delimiter $$
mysql> CREATE TRIGGER bef_student_updates
    ->     BEFORE UPDATE ON stud_tr
    ->     FOR EACH ROW BEGIN
    ->     INSERT INTO studt_pub
    ->     SET result='published',
    ->     rollno=OLD.rollno,
    ->     name=OLD.name,
    ->     examres=NOW();
    ->     END$$
Query OK, 0 rows affected (0.00 sec)
```

mysql> update stud_tr set name='Anusha' where rollno='13CS11';

```
Query OK, 1 row affected, 1 warning (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 1

mysql> select * from studt_pub;
+----+--------+---------+---------------------+-----------+
| id | rollno | name    | examres             | result    |
+----+--------+---------+---------------------+-----------+
|  1 |     13 | Ashwini | 2015-04-21 16:16:29 | published |
+----+--------+---------+---------------------+-----------+
1 row in set (0.00 sec)
```

RESULT:

The triggers and functions were created in MySQL Database using Procedural Language concepts.

Ex.No.8 IMPLEMENTATION OF B TREE AND B+ TREE

AIM:

To implement B Tree and B+ Tree using Procedural Language concepts.

ALGORITHM:

1. In a B-Tree, the new element must be added to the leaf node only. The insertion operation can be performed as follows.
2. Initially we must check if the tree is empty or not.
3. If the tree is found to be empty, then a new node with new key value is created inserted as the root node.
4. If the tree is not empty, then using the BST logic the new node is inserted to it's suitable location.
5. If there is some empty location at the leaf then, keeping in mind the increasing order of the key value, the node is inserted at the suitable position.
6. If the leaf node is filled completely, then split the node by moving the middle element upwards to the parent node.
7. If the node to be split is the root node, then the middle element that is moved becomes the new root node.

PROGRAM:

```
class BTree
{
   private int T;

   public class Node
   {
      int n;
      int key[] = new int[2 * T - 1];
      Node child[] = new Node[2 * T];
      boolean leaf = true;

      public int Find (int k)
      {
         for (int i = 0; i < this.n; i++)
                  {
            if (this.key[i] == k)
                     {
               return i;
            }
         }
         return -1;
      };
   }

   public BTree(int t)
   {
      T = t;
      root = new Node ();
      root.n = 0;
```

```
          root.leaf = true;
        }

        private Node root;

        // split
        private void split (Node x, int pos, Node y)
        {
          Node z = new Node ();
          z.leaf = y.leaf;
    z.n = T - 1;

    for (int j = 0; j < T - 1; j++)
    {
      z.key[j] = y.key[j + T];
    }
    if (!y.leaf)
    {
      for (int j = 0; j < T; j++)
        {
        z.child[j] = y.child[j + T];
      }
    }
    y.n = T - 1;

    for (int j = x.n; j >= pos + 1; j--)
    {
      x.child[j + 1] = x.child[j];
    }
    x.child[pos + 1] = z;

    for (int j = x.n - 1; j >= pos; j--)
    {
      x.key[j + 1] = x.key[j];
    }
    x.key[pos] = y.key[T - 1];
    x.n = x.n + 1;
  }

   // insert key
  public void insert (final int key)
  {
    Node r = root;

    if (r.n == 2 * T - 1)
    {
      Node s = new Node ();
      root = s;
      s.leaf = false;
      s.n = 0;
```

```
            s.child[0] = r;
            split (s, 0, r);
            _insert (s, key);
        }
        else
        {
            _insert (r, key);
        }
    }
    // insert node
    final private void _insert (Node x, int k)
    {

        if (x.leaf)
        {
            int i = 0;

               for (i = x.n - 1; i >= 0 && k < x.key[i]; i--)
               {
                x.key[i + 1] = x.key[i];
            }
            x.key[i + 1] = k;
            x.n = x.n + 1;
        }
        else
        {
            int i = 0;

            for (i = x.n - 1; i >= 0 && k < x.key[i]; i--)
               {
            };
            i++;

            Node tmp = x.child[i];
            if (tmp.n == 2 * T - 1)
               {
                split (x, i, tmp);
                if (k > x.key[i])
                   {
                    i++;
                 }
            }
            _insert (x.child[i], k);
        }
    }

    public void display ()
    {
        display (root);
    }
    // Display the tree
    private void display (Node x)
    {
        assert (x == null);
        for (int i = 0; i < x.n; i++)
        {
            System.out.print (x.key[i] + " ");
        }

        if (!x.leaf)
        {
            for (int i = 0; i < x.n + 1; i++)
              {
               display (x.child[i]);
            }
        }
    }
 }
 public class Main{
    public static void main (String[]args)
    {
        BTree b = new BTree(1);
        b.insert(5);
        b.insert (6);
        b.insert (7);
        b.insert (8);
        b.insert (12);
        b.insert (13);
        b.insert (14);
        b.display ();
    }
 }
```

OUTPUT :

```
5 6 7 8 12 13 14
```

ALGORITHM :(B+ TREE)

Step 1: Insert the new node as a leaf node
Step 2: If the leaf doesn't have required space, split the node and copy the middle node to **the** next index node.
Step 3: If the index node doesn't have required space, split the node and copy the middle element to the next index page.

PROGRAM:

```
import java.util.*;
class BplusTree {
   int[] d;
   BplusTree[] child_ptr;
   boolean l;
   int n;
}
```

```
public class Main {
 static BplusTree r = null, np = null, x = null;
 static BplusTree init() { // to create nodes
   int i;
   np = new BplusTree();
   np.d = new int[6]; // order 6
   np.child_ptr = new BplusTree[7];
   np.l = true;
   np.n = 0;
   for (i = 0; i < 7; i++) {
     np.child_ptr[i] = null;
   }
   return np;
 }
 static void traverse(BplusTree p) { // traverse tree
   int i;
   for (i = 0; i < p.n; i++) {
     if (p.l == false) {
       traverse(p.child_ptr[i]);
     }
     System.out.print(" " + p.d[i]);
   }
   if (p.l == false) {
     traverse(p.child_ptr[i]);
   }
   System.out.println();
 }
 static void sort(int[] p, int n) { // sort the tree
   int i, j, t;
   for (i = 0; i < n; i++) {
     for (j = i; j <= n; j++) {
       if (p[i] > p[j]) {
         t = p[i];
         p[i] = p[j];
         p[j] = t;
       }
     }
   }
 }
 static int split_child(BplusTree x, int i) {
   int j, mid;
   BplusTree np1, np3, y;
   np3 = init();
   np3.l = true;
   if (i == -1) {
     mid = x.d[2];
     x.d[2] = 0;
     x.n--;
     np1 = init();
     np1.l = false;
```

```
        x.l = true;
        for (j = 3; j < 6; j++) {
          np3.d[j - 3] = x.d[j];
          np3.child_ptr[j - 3] = x.child_ptr[j];
          np3.n++;
          x.d[j] = 0;
          x.n--;
        }
        for (j = 0; j < 6; j++) {
          x.child_ptr[j] = null;
        }
        np1.d[0] = mid;
        np1.child_ptr[np1.n] = x;
        np1.child_ptr[np1.n + 1] = np3;
        np1.n++;
        r = np1;
      } else {
        y = x.child_ptr[i];
        mid = y.d[2];
        y.d[2] = 0;
        y.n--;
        for (j = 3; j < 6; j++) {
          np3.d[j - 3] = y.d[j];
          np3.n++;
          y.d[j] = 0;
          y.n--;
        }
        x.child_ptr[i + 1] = y;
        x.child_ptr[i + 1] = np3;
      }
      return mid;
    }
    static void insert(int a) {
      int i, t;
      x = r;
      if (x == null) {
        r = init();
        x = r;
      } else {
        if (x.l == true && x.n == 6) {
          t = split_child(x, -1);
          x = r;
          for (i = 0; i < x.n; i++) {
            if (a > x.d[i] && a < x.d[i + 1]) {
              i++;
              break;
            } else if (a < x.d[0]) {
              break;
            } else {
              continue;
            }
          }
          x = x.child_ptr[i];
        } else {
          while (x.l == false) {
            for (i = 0; i < x.n; i++) {
              if (a > x.d[i] && a < x.d[i + 1]) {
                i++;
                break;
              } else if (a < x.d[0]) {
                break;
              } else {
                continue;
              }
            }
            if (x.child_ptr[i].n == 6) {
              t = split_child(x, i);
              x.d[x.n] = t;
              x.n++;
              continue;
            } else {
              x = x.child_ptr[i];
            }
          }
        }
      }
      x.d[x.n] = a;
      sort(x.d, x.n);
      x.n++;
    }
    public static void main(String[] args) {
      int i, n, t;
      insert(10);
      insert(20);
      insert(30);
      insert(40);
      insert(50);
         System.out.print("Insertion Done");
      System.out.println("\nB+ tree:");
      traverse(r);
    }
  }
```

OUTPUT:

```
Insertion Done
B+ tree:
 10 20 30 40 50
```

RESULT:

The B Tree and B+ Tree were implemented using java and executed successfully.

Ex.No.9 IMPLEMENTATION OF HASH TABLE

AIM
To write a C program for implementation of hashing – open addressing collision technique.

ALGORITHM
1. Start the program.
2. Enter the anyone number.
3. Find the key value by 10.
4. Check the remainder and table values are same.
5. If values i s same replace it.
6. Display the hash table.
7. Stop the program.

PROGRAM

```
#include<stdio.h>
#define MAX 10
void main()
{
        int a[MAX],num,key,i;
        int ans;
        int create(int);
        void linear_prob(int[],int,int);
        void display(int[]);
        printf("\nCollision handling by linearprobing");
        for(i=0;i<MAX;i++)
        a[i]=-1;
        do
        {
                printf("\nEnter the no.\t");
                scanf("%d",&num);
                key=create(num);
                linear_prob(a,key,num);
                printf("\n\nDo u wish to Continue?...(1/0)...\t");
                scanf("%d",&ans);
        }while(ans==1);
        display(a);
}
int create(int num)
{
        int key;
        key=num%10;
        return key;
}
void linear_prob(int a[MAX],int key,int num)
{
        int flag,count=0,i;
        void display(int a[]);
        flag=0;
if(a[key]==-1) a[key]=num;
        else
        {
                i=0;
                while(i<MAX)
                {
                        if(a[i]!=-1)
                        count++;
                        i++;
                }
                if(count==MAX)
                {
                        printf("\n Hash table is full");
                        display(a);
                        exit(1);
                }
                for(i=key+1;i<MAX;i++)
                if(a[i]==-1)
                {
                        a[i]=num;
                        flag=1;
                        break;
                }
                for(i=0;i<key && flag==0;i++)
                if(a[i]==-1)
                {
                        a[i]=num;
                        flag=1;
                        break;
                }
        }
}

void display(int a[MAX])
{
        int i;
        printf("\n The Hash table is .....\n");
        for(i=0;i<MAX;i++)
        printf("\n %d......%d",i,a[i]);
}
```

OUTPUT:

```
Collision handling by linear probing
Enter the no. 10
Do you wish to Continue?...(1/0)... 1
Enter the no. 11
Do you wish to Continue?...(1/0)... 1
Enter the no. 12
Do you wish to Continue?...(1/0)... 1
```

```
Enter the no. 13
Do you wish to Continue?...(1/0)... 1
Enter the no. 14
Do you wish to Continue?...(1/0)... 1
Enter the no. 15
Do you wish to Continue?...(1/0)... 1
Enter the no. 16
Do you wish to Continue?...(1/0)... 1
Enter the no. 17
Do you wish to Continue?...(1/0)... 1
Enter the no. 18
Do you wish to Continue?...(1/0)... 1
Enter the no. 28

Do you wish to Continue?...(1/0)... 1
Enter the no. 29
Hash table is full
The Hash table is .....
0......10
1......11
2......12
3......13
4......14
5......15
6......16
7......17
8......18
9......28
```

RESULT

Thus the program for hashing– open addressing collision technique has been implemented successfully.

Ex.No.10 DATABASE CONNECTIVITY WITH FRONT END TOOLS

AIM:

To design and implement a database application for college system using Netbeans and mysql.

RULES TO CONVERT ER DIAGRAM INTO TABLE:

- Each entity types become table
- Each single valued attributes become a column
- Derived attributes are ignored
- Composite attributes are represented by components
- Multi valued attributes are represented by separate table
- The key attribute of the entity type becomes primary key of the table

TABLE

Fields	Type	Description
Name	Text	Name of the student
Reg	Number	Registration Number
M1	Number	Mark1
M2	Number	Mark2
Total	Number	Total of mark1 and mark2

STEPS TO CONNECT JAVA WITH MYSQL

1. Create a database in MySQL.
2. Grant all privileges to user by providing password.
3. **Import the packages:** Requires that you include the packages containing the JDBC classes needed for database programming. Most often, using import java.sql.* will suffice.
4. **Register the JDBC driver:** Requires that you initialize a driver so you can open a communication channel with the database.
5. **Open a connection:** Requires using the DriverManager.getConnection() method to create a Connection object, which represents a physical connection with the database.
6. **Execute a query:** Requires using an object of type Statement for building and submitting an SQL statement to the database.
7. **Extract data from result set:** Requires that you use the appropriateResultSet.getXXX() method to retrieve the data from the result set.
8. **Clean up the environment:** Requires explicitly closing all database resources versus relying on the JVM's garbage collection.

STEPS TO IMPORT MYSQL CONNECTOR INTO YOUR PROJECT

1. Click project properties
2. Click on libraries and then Add Library
3. Click on import ,then you will see many jars available
4. Select the Mysql JDBC driver and import it.

Table Creation:

```
mysql> create database project;
Query OK, 1 row affected (0.02 sec)
```

```
mysql> use project;
Database changed
mysql> Grant all on project.* to root@'localhost' IDENTIFIED BY 'cse';
mysql> create table student(name varchar(20),reg int,m1 int,m2 int,total int);
Query OK, 0 rows affected (0.01 sec)
mysql> insert into student values('xxx',1001,67,20,87);
Query OK, 1 row affected (0.01 sec)
mysql> commit;
Query OK, 0 rows affected (0.01 sec)
mysql> select * from student;
+------+------+------+------+-------+
| name | reg  | m1   | m2   | total |
+------+------+------+------+-------+
| xxx  | 1001 |   67 |   20 |    87 |
| yyy  | 1003 |   50 |   45 |    95 |
+------+------+------+------+-------+
2 rows in set (0.01 sec)
```

CODE TO CONNECT STUDENT DATABASE

```
package College;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import static College.Student.DB_URL;

public class student extends javax.swing.JFrame
{
      static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost:3306/project";
     // Database credentials
    static final String USER = "root";
    static final String PASS = "cse";
    private Connection conn;
      public student()
     {
                    initComponents();
    }
      private void initComponents() {
      jLabel1 = new javax.swing.JLabel();
      jTextField1 = new javax.swing.JTextField();
      jLabel2 = new javax.swing.JLabel();
      jTextField2 = new javax.swing.JTextField();
      jLabel3 = new javax.swing.JLabel();
      jLabel4 = new javax.swing.JLabel();
      jLabel5 = new javax.swing.JLabel();
      jTextField3 = new javax.swing.JTextField();
```

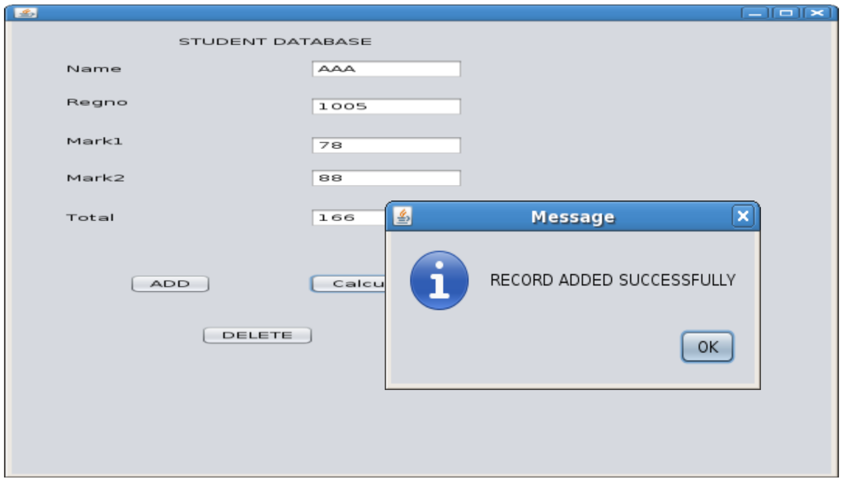
```
jTextField4 = new javax.swing.JTextField();
jTextField5 = new javax.swing.JTextField();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jLabel6 = new javax.swing.JLabel();
jButton3 = new javax.swing.JButton();
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
jLabel1.setText("Name");
jLabel2.setText("Regno");
jLabel3.setText("Mark1");
jLabel4.setText("Mark2");
jLabel5.setText("Total");
jTextField5.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jTextField5ActionPerformed(evt);
   }
});
jButton1.setText("ADD");
jButton1.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jButton1ActionPerformed(evt);
   }
});
jButton2.setText("Calculate");
jButton2.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jButton2ActionPerformed(evt);
   }
});
jLabel6.setText("STUDENT DATABASE");
jButton3.setText("DELETE");
jButton3.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jButton3ActionPerformed(evt);
   }
}); javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
   layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
   .addGroup(layout.createSequentialGroup()
      .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(layout.createSequentialGroup()
          .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
             .addGroup(layout.createSequentialGroup()
                .addGap(35, 35, 35)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                  .addComponent(jLabel1)
```

```
                        .addComponent(jLabel2)
                        .addComponent(jLabel3)
                        .addComponent(jLabel4)
                        .addComponent(jLabel5)))
                    .addGroup(layout.createSequentialGroup()
                      .addGap(75, 75, 75)
                      .addComponent(jButton1)))
                  .addGap(62, 62, 62)
addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                      .addComponent(jTextField1) .addComponent(jTextField2,
javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                      .addComponent(jTextField3) .addComponent(jTextField4)
                      .addComponent(jTextField5)) .addComponent(jButton2,
javax.swing.GroupLayout.PREFERRED_SIZE, 79, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGroup(layout.createSequentialGroup()
                  .addGap(107, 107, 107)
                  .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 193,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(layout.createSequentialGroup()
                  .addGap(121, 121, 121)
                  .addComponent(jButton3)))
              .addContainerGap(227, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
          layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
          .addGroup(layout.createSequentialGroup()
            .addGap(22, 22, 22)
            .addComponent(jLabel6)
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
              .addComponent(jLabel1)
              .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(27, 27, 27)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
              .addComponent(jLabel2)
              .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(29, 29, 29)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
              .addComponent(jLabel3)
              .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(20, 20, 20)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
              .addComponent(jLabel4)
```

```
                .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(30, 30, 30)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel5))              .addGap(68, 68, 68)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jButton1) .addComponent(jButton2))
            .addGap(47, 47, 47)             .addComponent(jButton3)
            .addContainerGap(186, Short.MAX_VALUE))
        );
        pack();
    }// </editor-fold>
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
         Connection conn = null;
          Statement stmt = null;
   try{
      //STEP 2: Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");
      //STEP 3: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL,USER,PASS);
      //STEP 4: Execute a query
      System.out.println("Creating statement...");
      stmt = conn.createStatement();
      String sql;
      String myname ="user";
      String name=jTextField1.getText();
      int reg=Integer.parseInt(jTextField2.getText());
      int m1=Integer.parseInt(jTextField3.getText());
      int m2=Integer.parseInt(jTextField4.getText());
      int total=Integer.parseInt(jTextField5.getText());
      stmt.executeUpdate("insert into student values ('" + name + "', '" + reg + "', '"+ m1 + "', '"+ m2+ "', '"+ total +
"')");
      JOptionPane.showMessageDialog(null, "RECORD ADDED SUCCESSFULLY");
      stmt.close();
      conn.close();
   }catch(SQLException se){
      se.printStackTrace();
   }catch(Exception e){
      e.printStackTrace();
   }finally{
        try{
        if(stmt!=null)
           stmt.close();
    }catch(SQLException se2){
    }
    try{
      if(conn!=null)
        conn.close();
    }catch(SQLException se){
      se.printStackTrace();
    }
  }
  System.out.println("Goodbye!");
   }

   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
  {
      // TODO add your handling code here:
       int m1=Integer.parseInt(jTextField3.getText());
      int m2=Integer.parseInt(jTextField4.getText());
      int total=m1+m2;
      jTextField5.setText(""+total);
   }

   private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
   {
      // TODO add your handling code here:
      Connection conn = null;
      Statement stmt = null;
      try{
    //STEP 2: Register JDBC driver
    Class.forName("com.mysql.jdbc.Driver");

    //STEP 3: Open a connection
    System.out.println("Connecting to database...");
    conn = DriverManager.getConnection(DB_URL,USER,PASS);

    //STEP 4: Execute a query
    System.out.println("Creating statement...");
    stmt = conn.createStatement();
    String sql;
        stmt.executeUpdate("delete from student where reg='"+jTextField2.getText()+"'");
       JOptionPane.showMessageDialog(null, "RECORD DELETED SUCCESSFULLY");
    stmt.close();
    conn.close();
       }catch(SQLException se){
    se.printStackTrace();
  }catch(Exception e){
    e.printStackTrace();
  }finally{
        try{
          if(stmt!=null)
             stmt.close();
       }catch(SQLException se2){
       }
       try{
          if(conn!=null)
             conn.close();
       }catch(SQLException se){
          se.printStackTrace();
       }
    }
    System.out.println("Goodbye!");
     }
     public static void main(String args[]) {
                try {
             for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
             }
         } catch (ClassNotFoundException ex)
{           java.util.logging.Logger.getLogger(student.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
         } catch (InstantiationException ex)
{           java.util.logging.Logger.getLogger(student.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
         } catch (IllegalAccessException ex)
{       java.util.logging.Logger.getLogger(student.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
         } catch (javax.swing.UnsupportedLookAndFeelException ex)
{           java.util.logging.Logger.getLogger(student.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
         }
         //</editor-fold>
         /* Create and display the form */
         java.awt.EventQueue.invokeLater(new Runnable() {
             public void run() {
                 new student().setVisible(true);
             }       });
     }
     // Variables declaration - do not modify
     private javax.swing.JButton jButton1;
     private javax.swing.JButton jButton2;
     private javax.swing.JButton jButton3;
     private javax.swing.JLabel jLabel1;
     private javax.swing.JLabel jLabel2;
     private javax.swing.JLabel jLabel3;
     private javax.swing.JLabel jLabel4;
     private javax.swing.JLabel jLabel5;
     private javax.swing.JLabel jLabel6;
     private javax.swing.JTextField jTextField1;
     private javax.swing.JTextField jTextField2;
     private javax.swing.JTextField jTextField3;
     private javax.swing.JTextField jTextField4;
     private javax.swing.JTextField jTextField5;
     // End of variables declaration
 }
```

OUTPUT:

INSERTION OF NEW RECORD



STUDENT DATABASE

Name: AAA

Regno: 1005

Mark1: 78

Mark2: 88

Total: 166

Message

RECORD ADDED SUCCESSFULLY

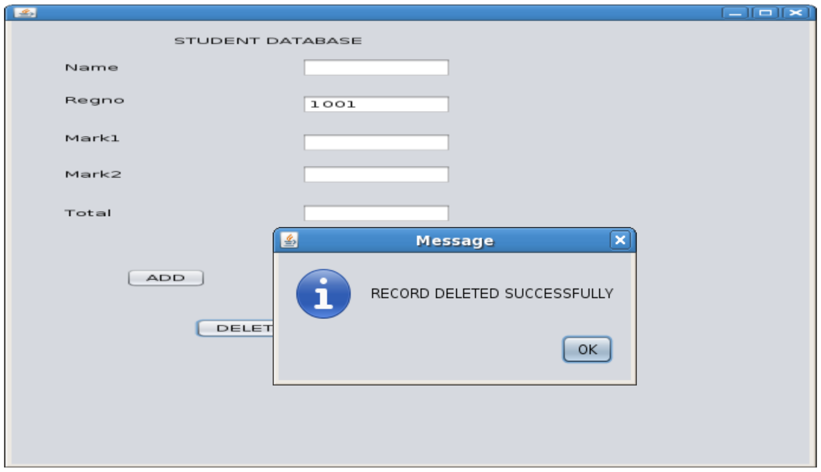
Before Update:

```
mysql> select * from student;
+------+------+------+------+-------+
| name | reg  | m1   | m2   | total |
+------+------+------+------+-------+
| xxx  | 1001 |   67 |   20 |    87
  yyy  | 1003 |   50 |   45 |    95 |
+------+------+------+------+-------+
2 row in set (0.00 sec)
```

After Update:

```
mysql> select * from student;
+------+------+------+------+-------+
| name | reg  | m1   | m2   | total |
+------+------+------+------+-------+
| xxx  | 1001 |   67 |   20 |    87 |
| yyy  | 1003 |   50 |   45 |    95 |
| AAA  | 1005 |   78 |   88 |   166 |
+------+------+------+------+-------+
3 rows in set (0.00 sec)
```

DELETION OF STUDENT RECORD



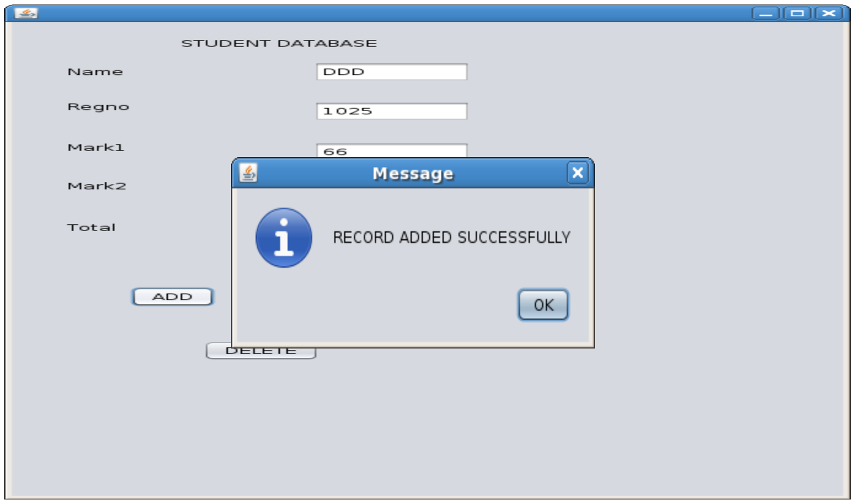
1001

RECORD DELETED SUCCESSFULLY

After Deletion

```
mysql> select * from student;
+------+------+------+------+-------+
| name | reg  | m1   | m2   | total |
+------+------+------+------+-------+
| yyy  | 1003 |   50 |   45 |    95 |
+------+------+------+------+-------+
1 row in set (0.00 sec)
```

INSERTION OF NEW RECORD



DDD

1025

66

RECORD ADDED SUCCESSFULLY

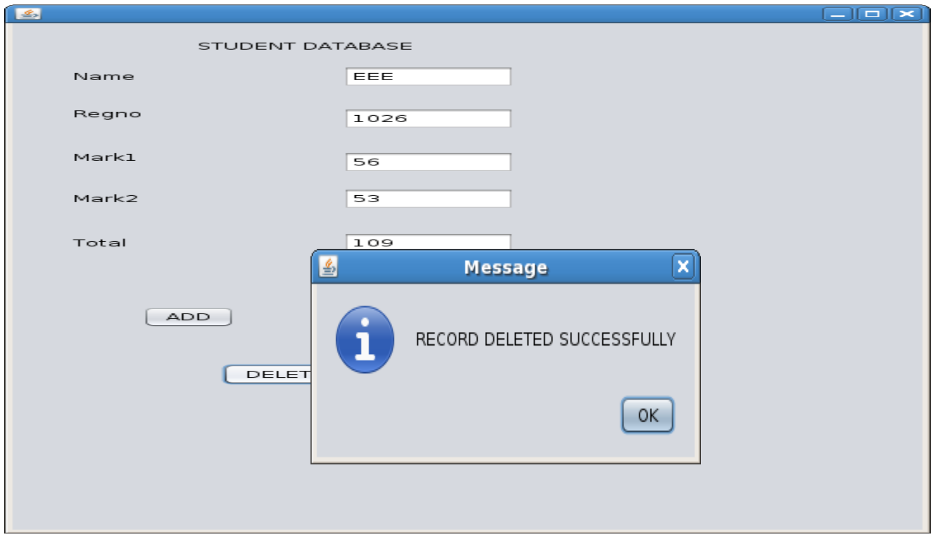
Before Update:

```
mysql> select * from student;
+------+------+------+------+-------+
| name | reg  | m1   | m2   | total |
+------+------+------+------+-------+
| yyy  | 1003 |   50 |   45 |    95 |
| EEE  | 1026 |   56 |   53 |   109 |
+------+------+------+------+-------+
```

After Update:
mysql> select * from student;

```
+------+------+------+------+-------+
| name | reg  | m1   | m2   | total |
+------+------+------+------+-------+
| DDD  | 1025 |   66 |   55 |   121 |
| yyy  | 1003 |   50 |   45 |    95 |
| EEE  | 1026 |   56 |   53 |   109 |
+------+------+------+------+-------+
```

DELETION OF STUDENT RECORD



EEE
1026
56
53
109
RECORD DELETED SUCCESSFULLY

After Deletion
mysql> select * from student;

```
+------+------+------+------+-------+
| name | reg  | m1   | m2   | total |
+------+------+------+------+-------+
| DDD  | 1025 |   66 |   55 |   121 |
| yyy  | 1003 |   50 |   45 |    95 |
+------+------+------+------+-------+
```
2 rows in set (0.00 sec)

RESULT
The student database was implemented using the java as front end and MySQL as back end.

Exp.No:11 **CASE STUDY USING REAL LIFE DATABASE APPLICATIONS**

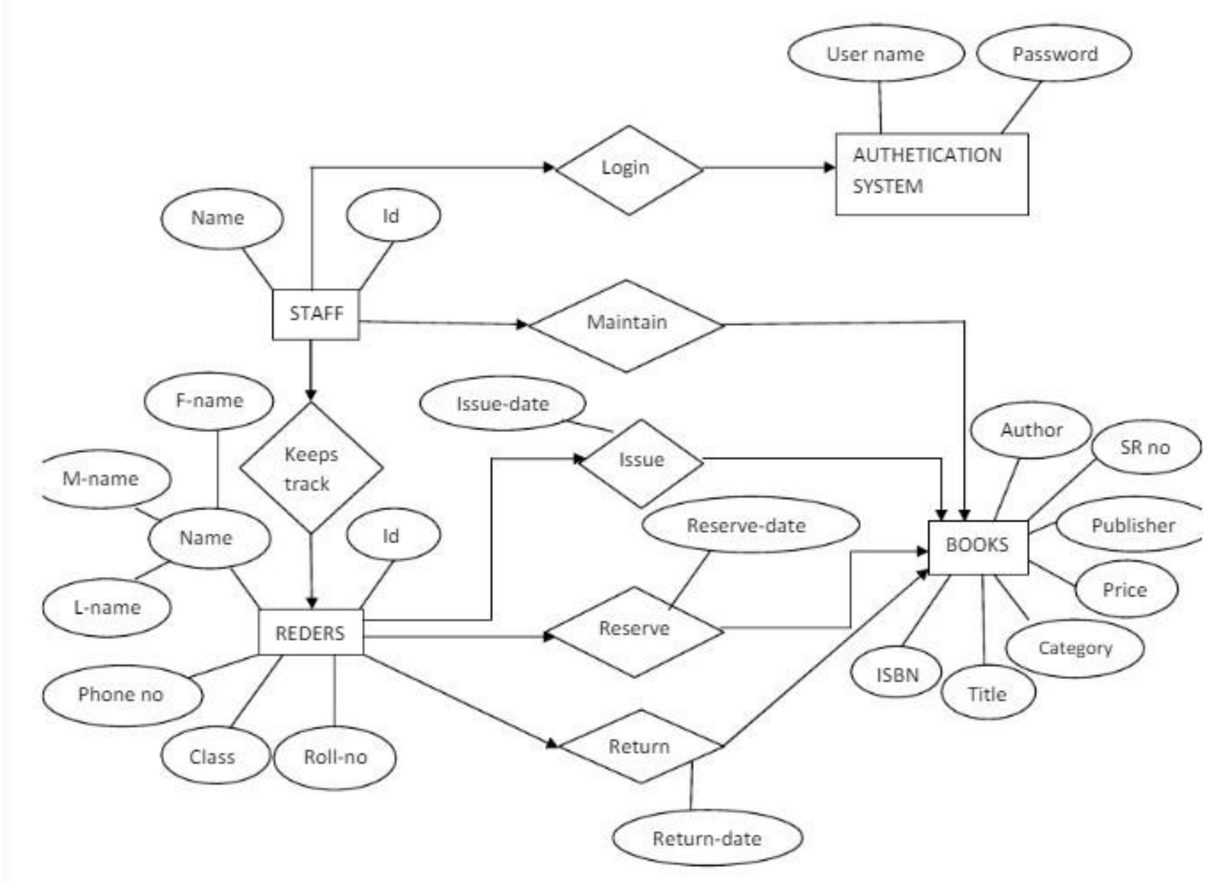
AIM:

To design and implement a database application for library management system using Netbeans and mysql.

PROBLEM STATEMENT:

The case study of library management system gives the complete information about the library. We can enter the record of new book and retrieve the details of books available in the library. We can issue the books to the students and maintain their records and also check how many books are issued and stock available in the library. We can also search the books available in the library.

ER DIAGRAM:



SAMPLE CODE:

```
import java.sql.Connection;
import java.sql.DriverManager;

public class DB {
            public static Connection getConnection(){
                        Connection con=null;
                        try{
                                    Class.forName("com.mysql.jdbc.Driver");
                                    con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test","","");
                        }
catch(Exception e)
{System.out.println(e);}
return con;
            }

}
```

LIBRARYIAN FORM

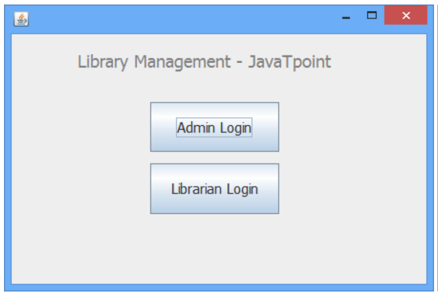
```
import java.awt.BorderLayout;
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import java.awt.Font;
import java.awt.Color;
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.LayoutStyle.ComponentPlacement;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class LibrarianForm extends JFrame {
	static LibrarianForm frame;
	private JPanel contentPane;
	private JTextField textField;
	private JTextField textField_1;
	private JTextField textField_2;
	private JTextField textField_3;
	private JTextField textField_4;
	private JPasswordField passwordField;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					frame = new LibrarianForm();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
```

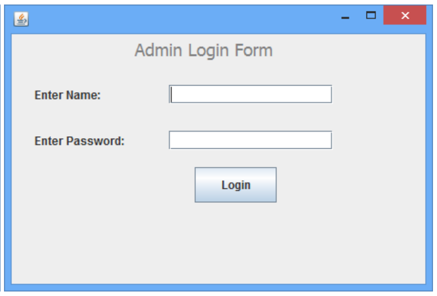
RETURN BOOK

```
import java.awt.BorderLayout;
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.GroupLayout;
import javax.swing.GroupLayout.Alignment;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import java.awt.Font;
import java.awt.Color;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.LayoutStyle.ComponentPlacement;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class ReturnBook extends JFrame {
	static ReturnBook frame;
	private JPanel contentPane;
	private JTextField textField;
	private JTextField textField_1;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					frame = new ReturnBook();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public ReturnBook() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 516, 413);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		JLabel lblReturnBook = new JLabel("Return Book");
		lblReturnBook.setForeground(Color.GRAY);
		lblReturnBook.setFont(new Font("Tahoma", Font.PLAIN, 18)); JLabel lblBookCallno =
		new JLabel("Book Callno:");
		JLabel lblStudentId = new JLabel("Student Id:");
		textField = new JTextField();
		textField.setColumns(10);
		textField_1 = new JTextField();
		textField_1.setColumns(10);
		JButton btnReturnBook = new JButton("Return Book");
		btnReturnBook.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				String bookcallno=textField.getText();
				int studentid=Integer.parseInt(textField_1.getText());
				int i=ReturnBookDao.delete(bookcallno, studentid);
				if(i>0){
					JOptionPane.showMessageDialog(ReturnBook.this,"Book
returned successfully!");
					LibrarianSuccess.main(new String[]{});
					frame.dispose();

				}else{
					JOptionPane.showMessageDialog(ReturnBook.this,"Sorry,
unable to return book!");
				}
			}
		});

		JLabel lblNewLabel = new JLabel("Note: Check the book properly!");
		lblNewLabel.setForeground(Color.RED);
		lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 13));

		JButton btnBack = new JButton("Back");
		btnBack.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				LibrarianSuccess.main(new String[]{});
				frame.dispose();
			}
		});
```

OUTPUT:

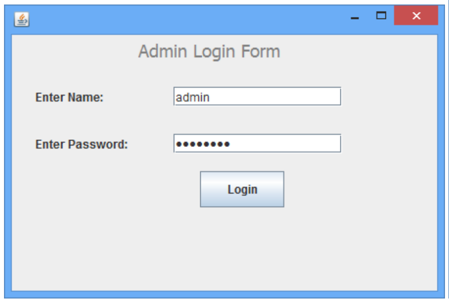
LIBRARY MANAGEMENT SYSTEM SCREEN SHOT




Admin Login Form

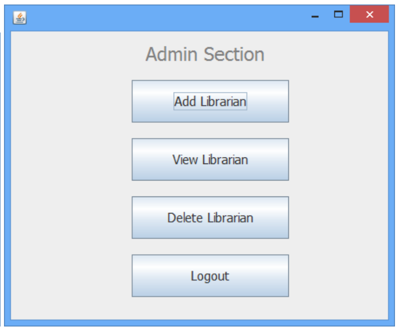
Enter Name:

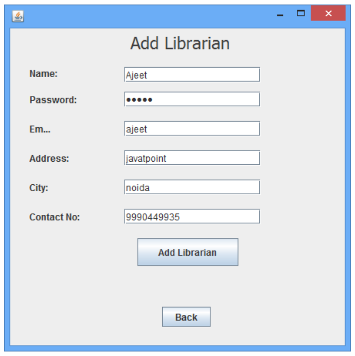
Enter Password:


Admin Login Form

Enter Name: admin

Enter Password: ••••••••




Add Librarian

Name: Ajeet

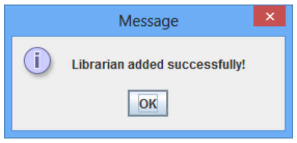
Password: •••••

Em... ajeet

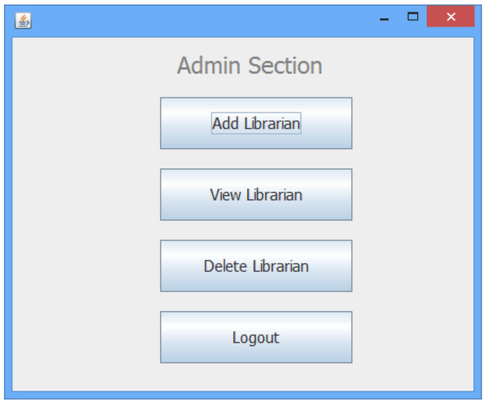
Address: javatpoint

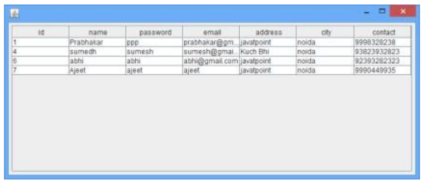
City: noida

Contact No: 9990449935

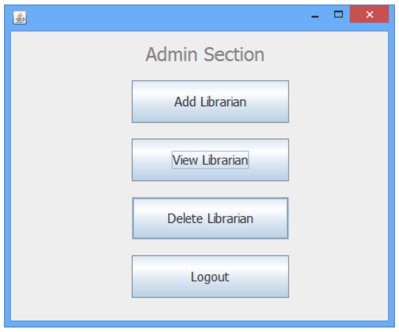

Message

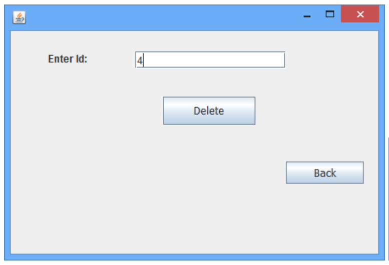
Librarian added successfully!

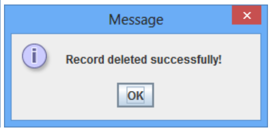




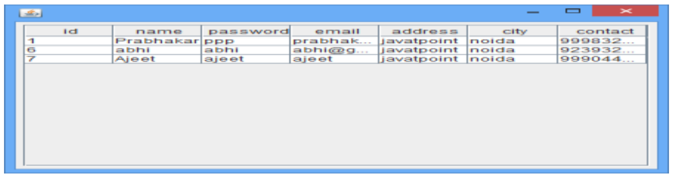
id	name	password	email	address	city	contact
1	Prabhakar	ppp	prabhakar@gm...	javatpoint	noida	9998328238
4	sumedh	sumesh	sumesh@gmai...	Kuch Bhi	noida	93823932823
6	abhi	abhi	abhi@gmail.com	javatpoint	noida	92393282323
7	Ajeet	ajeet	ajeet	javatpoint	noida	9990449935



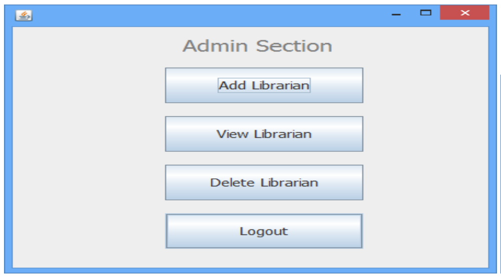

Enter Id: 4

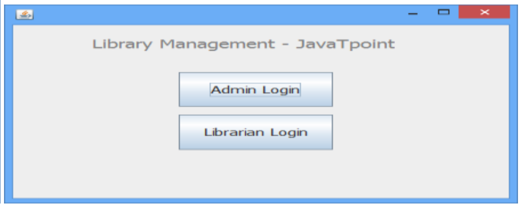


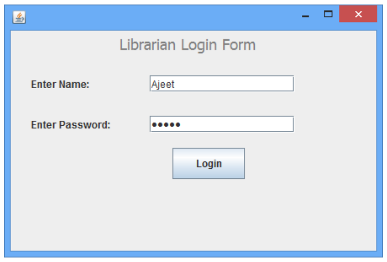
Record deleted successfully!



id	name	password	email	address	city	contact
1	Prabhakar	ppp	prabhak...	javatpoint	noida	999832...
6	abhi	abhi	abhi@g...	javatpoint	noida	923932...
7	Ajeet	ajeet	ajeet	javatpoint	noida	999044...

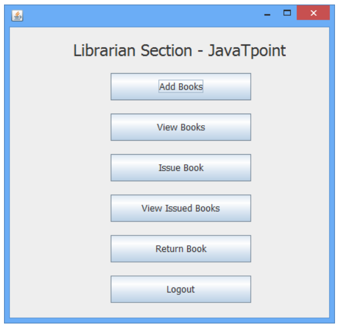


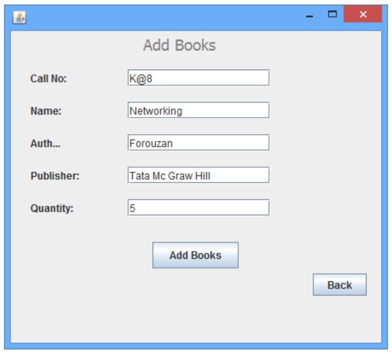




Enter Name: Ajeet

Enter Password: •••••





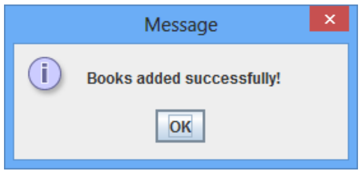
Call No: K@8

Name: Networking

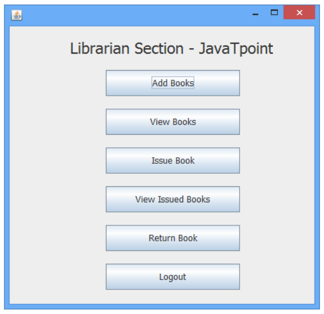
Auth...: Forouzan

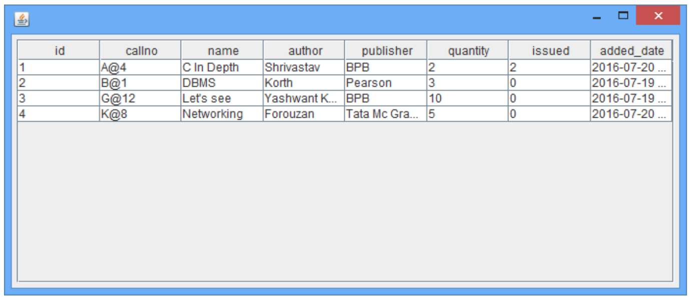
Publisher: Tata Mc Graw Hill

Quantity: 5

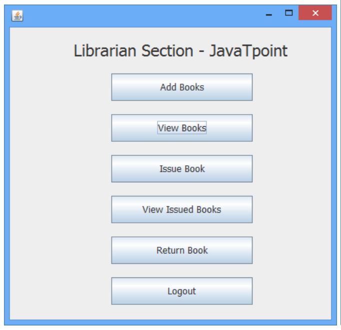


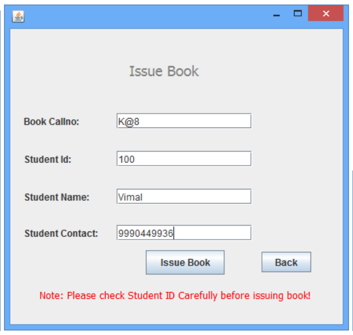
Books added successfully!





id	callno	name	author	publisher	quantity	issued	added_date
1	A@4	C In Depth	Shrivastav	BPB	2	2	2016-07-20 ...
2	B@1	DBMS	Korth	Pearson	3	0	2016-07-19 ...
3	G@12	Let's see	Yashwant K...	BPB	10	0	2016-07-19 ...
4	K@8	Networking	Forouzan	Tata Mc Gra...	5	0	2016-07-20 ...

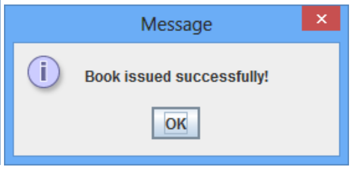




Issue Book

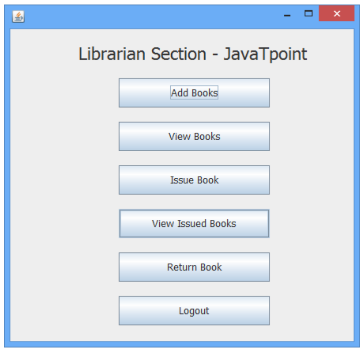
Book Callno:	K@8
Student Id:	100
Student Name:	Vimal
Student Contact:	9990449936

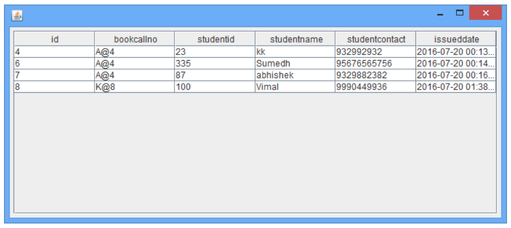
Note: Please check Student ID Carefully before issuing book!



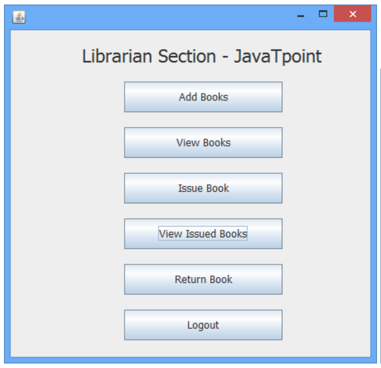
Message

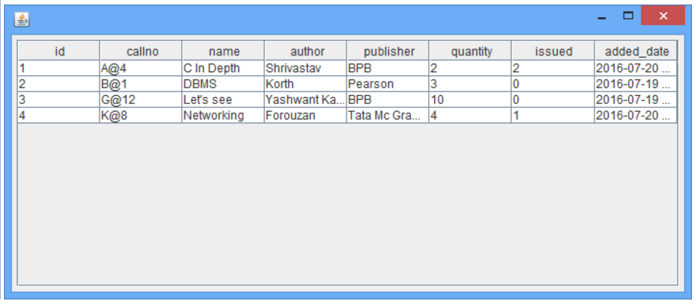
Book issued successfully!



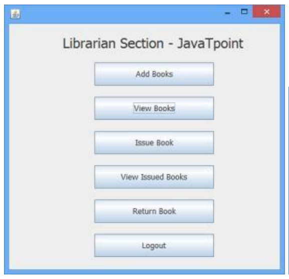


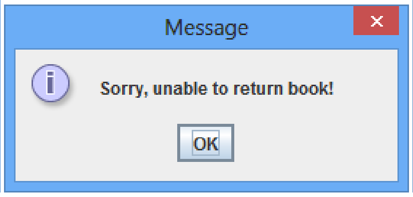
id	bookcallno	studentid	studentname	studentcontact	issueddate
4	A@4	23	kk	932992932	2016-07-20 00:13...
6	A@4	335	Sumedh	95676565756	2016-07-20 00:14...
7	A@4	87	abhishek	9329882382	2016-07-20 00:16...
8	K@8	100	Vimal	9990449936	2016-07-20 01:38...



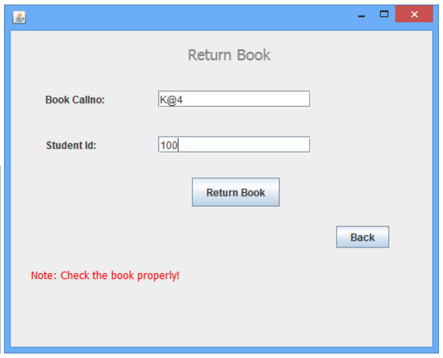


id	callno	name	author	publisher	quantity	issued	added_date
1	A@4	C In Depth	Shrivastav	BPB	2	2	2016-07-20 ...
2	B@1	DBMS	Korth	Pearson	3	0	2016-07-19 ...
3	G@12	Let's see	Yashwant Ka...	BPB	10	0	2016-07-19 ...
4	K@8	Networking	Forouzan	Tata Mc Gra...	4	1	2016-07-20 ...





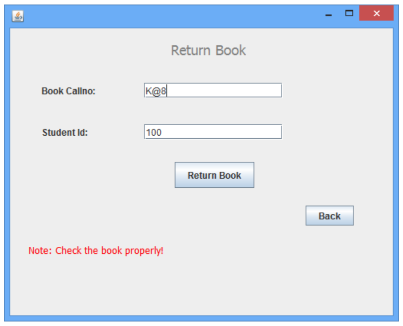
Sorry, unable to return book!



Book Callno: K@4

Student Id: 100

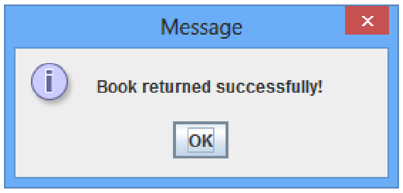
Note: Check the book properly!



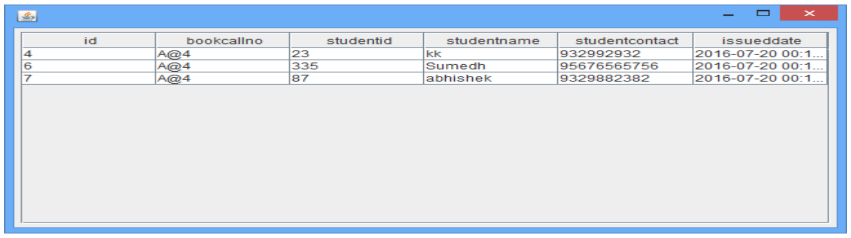
Book Callno: K@8

Student Id: 100

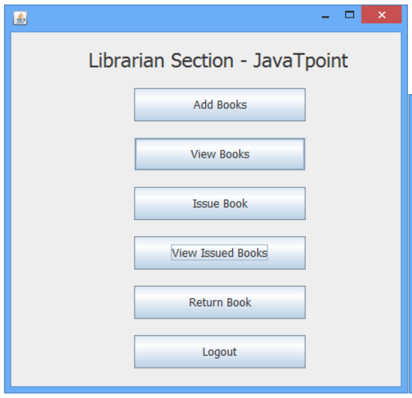
Note: Check the book properly!

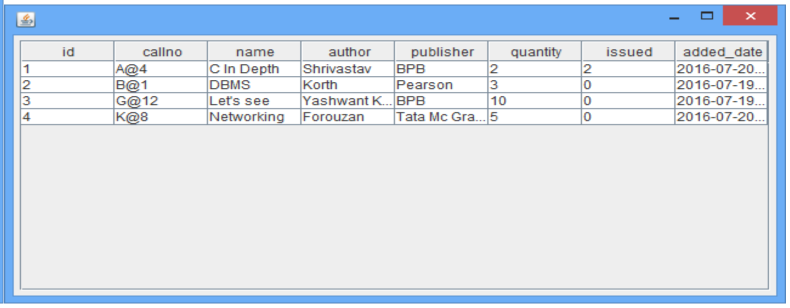


Book returned successfully!



id	bookcallno	studentid	studentname	studentcontact	issueddate
4	A@4	23	kk	932992932	2016-07-20 00:1...
6	A@4	335	Sumedh	95676565756	2016-07-20 00:1...
7	A@4	87	abhishek	9329882382	2016-07-20 00:1...





id	callno	name	author	publisher	quantity	issued	added_date
1	A@4	C In Depth	Shrivastav	BPB	2	2	2016-07-20...
2	B@1	DBMS	Korth	Pearson	3	0	2016-07-19...
3	G@12	Let's see	Yashwant K...	BPB	10	0	2016-07-19...
4	K@8	Networking	Forouzan	Tata Mc Gra...	5	0	2016-07-20...

RESULT:

A database application for library management system using Netbeans and Mysql was designed and implemented successfully.