

**Ex.No:2 DATABASE QUERYING – SIMPLE QUERIES, NESTED QUERIES,
SUB QUERIES AND JOINS**

AIM:

To implement and execute a query for manipulating & storing data items in MySQL database using Structured Query Language commands

SYNTAX:

NESTED and SUB QURIES:

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

JOINS:

INNER JOIN:

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

LEFT JOIN

SELECT column_name(s)FROM table_name1LEFT JOIN table_name2ON table_name1.column_name=table_name2.column_name

RIGHT JOIN

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

FULL JOIN

SELECT column_name(s)FROM table_name1FULL JOIN table_name2ON table_name1.column_name=table_name2.column_name

SIMPLE QUERIES

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

mysql> select amount*100 from loan_m;

2. Find all 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;
```

3. Find all 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;
```

4. Find the 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%';
```

5. Find the names of customer_m whose street name starts with substring 'ma'.

```
mysql> select customer_name from customer_m where customer_street
like 'ma%';
```

6. Display the entire loan relation in descending order of amount.

```
mysql> select * from loan_m order by amount desc;
```

7. Find total number of customer.

```
mysql> select count(customer_id) from customer_m;
```

8. Find all the loan number that appears in the loan relation with NULL values for amount.

```
mysql> select loan_no from loan_m where amount is null;
```

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

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');
```

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;
```

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);
```

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);
```

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');
```

JOINS:

1. For all customer who have loan from the bank find their ID's , loan number and loan amount.(Join)

```
mysql>Select      borrow_ma.customer_id,      loan_ma.loan_no,
loan_ma.amount from borrow_ma,
loan_ma where loan_ma.loan_no=borrow_ma.loan_no;
```

2. For all customers who have loan at perryridge branch find their ID's,loan ID,loan amount.(Join)

```
mysql>Select      borrow_ma.customer_id,      loan_ma.loan_no,
loan_ma.amount      from      loan_ma,borrow_ma      where
borrow_ma.loan_no=loan_ma.loan_no      and
loan_ma.branch_name='Perryridge';
```

3. Find the number of depositor at each branch.(Join)

```
mysql>Select
account_ma.branch_name,count(depositor_ma.customer_id)      from
account_ma,depositor_ma      where
depositor_ma.account_no=account_ma.account_no      group      by
account_ma.branch_name;
```

RESULT:

INFERENCE:

1. List the set operations of SQL?
2. What are aggregate functions? And list the aggregate functions supported by SQL?
3. What is the use of group by clause?
4. What is the use of sub queries?
5. What is view in SQL? How is it defined?

Ex.No:3**VIEWS, INDEXES AND SYNONYM****AIM:**

To implement and execute view,sequence,indexes,savepoint in MYSQL using SQL commands.

VIEWS:

Create view <viewname> as any select query; (Ex: select * from <tablename> where <condition>)

SEQUENCE:

Create table <tablename> (column1 datatype1 primary key auto_increment, column2 datatype 2.....columnN datatypeN)

SYNONYM:

Create synonym <synonym name> for <table name>;

VIEWS:**1) 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));
```

```
mysql> desc person_m;
```

```
mysql> insert into person_m values('abdulkalam','1931-08-18','rameshwaram');
```

```
mysql> select * from person_m;
```

```
mysql> create view personage_m as select name,datediff(sysdate(),dob)/365.25 as age from person_m;
```

```
mysql> select * from personage_m;
```

```
mysql> insert into person_m values('jorden','1970-07-08','usa');
```

```
mysql> select * from personage_m;
```

SEQUENCE:

2) 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));
```

```
mysql> desc department_m;
```

```
mysql> insert into department_m(department_name)values('it');
```

```
mysql> insert into department_m(department_name)values('cse');
```

```
mysql>  
insert into department_m(department_name)values('mechanical');
```

```
mysql> insert into department_m(department_name)values('aero');
```

```
mysql> select * from department_m;
```

SYNONYM(UsingOracle)

3) Show the effect of synonym

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;
```

```
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;
```

SQL> **SELECT * FROM** prod_m;

SQL> drop SYNONYM prod_m;

SQL> drop table product_m;

RESULT:

INFERENCE:

1. What is versioning in terms of object oriented database?
2. Specify the advantages of Data warehousing.
3. How spatial databases are more helpful than active database?
4. What is deductive database?
5. Briefly explain about applications of data warehousing.

Ex.No.4 STUDY OF PL/SQL-SIMPLE PROGRAMS**AIM:**

To implement and execute PL/SQL in Mysql using Procedural Language concepts.

PL/SQL: which is a block-structured language; this means that the PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts.

1 Declarations

This section starts with the keyword DECLARE. It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.

2 Executable Commands

This section is enclosed between the keywords BEGIN and END and it is a mandatory section. It consists of the executable PL/SQL statements of the program. It should have at least one executable line of code, which may be just a NULL command to indicate that nothing should be executed.

3 Exception Handling

This section starts with the keyword EXCEPTION. This optional section contains exception(s) that handle errors in the program.

Every PL/SQL statement ends with a semicolon (;). PL/SQL blocks can be nested within other PL/SQL blocks using BEGIN and END. Following is the basic structure of a PL/SQL block –

```
DECLARE
  <declarations section>
BEGIN
  <executable command(s)>
EXCEPTION
  <exception handling>
END;
```

Simple Programs:**(i) PL/SQL Program to add 2 numbers**

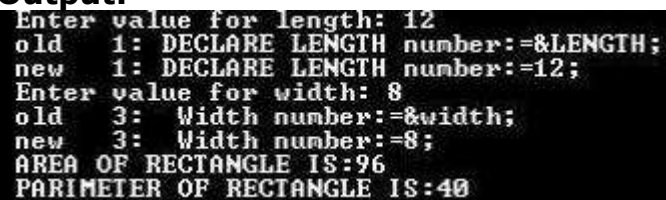
```
DECLARE
  a integer := 10;
  b integer := 20;
  c integer;
  f real;
BEGIN
  c := a + b;
  dbms_output.put_line('Value of c: ' || c);
  f := 70.0/3.0;
  dbms_output.put_line('Value of f: ' || f);
END;
/
```

When the above code is executed, it produces the following result –

(ii) PL/SQL Program to calculate area and perimeter of Rectangle.

```
DECLARE
LENGTH number:=&LENGTH;
Width number:=&width;
Area number;
Parimeter number;
BEGIN Area:=LENGTH*width;
Parimeter:= 2*(LENGTH+width);
dbms_output.put_line('AREA OF RECTANGLE IS:'||Area);
dbms_output.put_line('PARIMETER OF RECTANGLE IS:'||PARIMETER);
END;
```

/

Output:

```
Enter value for length: 12
old 1: DECLARE LENGTH number:=&LENGTH;
new 1: DECLARE LENGTH number:=12;
Enter value for width: 8
old 3: Width number:=&width;
new 3: Width number:=8;
AREA OF RECTANGLE IS:96
PARIMETER OF RECTANGLE IS:40
```

(iii) PL/SQL Program to obtain factorial of a number.

```
Declare
num number:= #
fact number:= 1;
temp number;
begin
temp := num;
while (num > 0)
loop
fact := fact * num;
num := num - 1;
end loop;
Dbms_Output.Put_line('factorial of ' || temp || ' is ' || fact);
end;
```

/

(iv) PL/SQL block to obtain Fibonacci series.

```
declare
  a number(3):=1;
  b number(3):=1;
  c number(3);
  n number(3):=&n;
begin
  Dbms_output.put_line('the fibinocci series is:');
  while a<=n
  loop
    dbms_output.put_line(a);
    c:=a+b;
    a:=b;
    b:=c;
  end loop;
end;
/
```

RESULT:

Ex.No.5 DATABASE PROGRAMMING: IMPLICIT AND EXPLICIT CURSORS**AIM:**

To implement and execute cursor in Mysql using Procedural Language concepts.

Cursor:A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.

There are two types of cursors –

- Implicit cursors
- Explicit cursors

SYNTAX:**IMPLICIT CURSOR:**

Sql%attribute name

Where attribute name can be FOUND, NOT FOUND, ISOPEN, %ROWCOUNT

EXPLICIT CURSOR:

Creating an explicit cursor

CURSOR <cursor_name> IS select_statement;

Opening the Cursor

OPEN <cursor_name>;

Fetching the Cursor

FETCH <cursor_name>INTO attributes;

Closing the Cursor

CLOSE <cursor_name>;

IMPLICIT CURSOR (in Oracle)

DECLARE

rows number(2);

BEGIN

 UPDATE student SET stu_class = stu_class + 2;

 IF sql%notfound THEN

 dbms_output.put_line('no student selected');

 ELSIF sql%found THEN

rows := sql%rowcount;

 dbms_output.put_line(rows || ' student selected ');

 END IF;

END;

4 student selected

EXPLICIT CURSOR:

```
mysql> delimiter $$
mysql> create procedure curdemo(id int)
    -> begin
    -> declare name varchar(255);
    -> declare curl cursor for select stu_name from student where
stu_id=id;
    -> open curl;
    -> fetch curl into name;
    -> select name;
    -> close curl;
    -> end $$

mysql> delimiter ;
mysql> call curdemo(2);
```

RESULT:**INFERENCE:**

1. Define Database Security.
2. Illustrate about Data Classification.
3. Define Threats and risks.
4. How Database access Control.
5. Give types of Privileges.

Ex.No.6**PROCEDURES AND FUNCTIONS****AIM:**

To implement and execute Procedures and functions in Mysql using Procedural Language concepts.

PROCEDURES: Procedure is a sub program used to perform an action. Replace-recreates the procedure if it already exists.

3 MODES:

- 1) IN – Means you must specify a value for the argument at the time execution of the procedure.
- 2) □OUT-passes back a value to its calling program.
- 3) INOUT – When calling the procedure, yu must specify the value and that procedures passes value back to the calling procedure.

SYNTAX:

Create procedure <procedure_name> (argument {in, out, in out} data type) is

Variable declaration

Begin

PI/SQL Subprogram body.

End;

FUNCTION: A function is a sub program that accepts argument and returns a unique value to the caller.

SYNTAX :

Create function <function_name> (parameter) return <data type> is

Variable declaration

Begin

PI/SQL Subprogram body.

Return statement

End;

1. Procedures:**Create a procedure to update the phone number to customer table**

```
Mysql> create table customer_m (cust_id int primary key, cust_name  
varchar(20), cust_phone int);
```

```
Mysql> desc customer_m;
```

```
Mysql> insert into customer_m (cust_id, cust_name) values (201, 'raja');
```

```
Mysql> insert into customer_m (cust_id, cust_name) values (202, 'ravi');
```

```
Mysql> select * from customer_m;
```

Procedure

```
DELIMITER //
CREATE PROCEDURE c
  (IN ph int,IN cid int)
  BEGIN
  declare name varchar(20);
  select cust_name into name from customer_m where cust_id=cid;
  if name is null then
    select 'Name is not there';
  ELSE
  update customer_m set cust_phone=ph where cust_id=cid;
  END IF;
END //
```

DELIMITER ;

Mysql> call c(66,88);

Mysql> call c(98567,201);

Mysql> select * from customer_m;

2. Functions

Create a function to check whether particular customer is having phone number or Not

```
create table customer_m (custid int,phone int);
insert into customer_m values(101,899);
```

```
mysql> create function check12(cust_id int)
->returns varchar(20)
-> begin
-> declare ph int;
->select phone into ph from customer_m where custid=cust_id;
->if ph is null then
-> return 'Mobile Number is not there';
-> ELSE
-> return 'Mobile Number is there';
-> END IF;
->end //
```

mysql> select * from customer_m;

mysql> select check12(101);

```
mysql> insert into customer_m (custid) values (102);
```

```
mysql> select * from customer_m;
```

```
mysql> select check12(102);
```

RESULT:**INFERENCE:**

1. Define mobile database with an example.
2. List the markup languages which are suitable for web databases.
3. Write two examples of multimedia databases and multimedia structure.
4. Define spatial database.
5. Differentiate distributed database and normal database.

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 trail 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>
REFERENCECING {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;
```

1) Create a trigger to calculate the total and average of a student in the student table.

```
mysql> create table student_m (regno int primary key,name
varchar(20),tamil int,english int,maths int,science int,social
int,total int,average int);
```

```
Mysql> desc student_m;
```

```
Mysql> CREATE TRIGGER avg1 BEFORE INSERT ON student_m FOR EACH ROW
-> BEGIN
->
SET
new.total=new.tamil+new.english+new.maths+new.science+new.social;
-> SET new.average=new.total/5;
-> END; //
```

```
DELIMITER ;
```

```
mysql>          insert          into          student_m
(regno,name,tamil,english,maths,science,social)values(101,'raja',90,95
,100,100,99);
```

```
Mysql> select * from student;
```

RESULT:

INFERENCE:

1. Give the measures of the quality of a disk.
2. What are the two types of ordered indices?
3. What are structured data types? What are collection types in particular?
4. State the advantages of distributed systems.
5. What is Data Warehousing?

Ex.No.8**EXCEPTION HANDLING****AIM:**

To implement and execute PL/SQL Block that handles all types of exceptions in Oracle Database using Procedural Language concepts.

SYNTAX**DECLARE**

Variable

declaration

BEGIN

Program

Execution

EXCEPTION

Exception

handling

END;**ZERO DIVIDE EXCEPTION**

SQL> BEGIN

2 DBMS_OUTPUT.PUT_LINE(1 / 0);

3 END;

4 /

BEGIN

*

ERROR at line 1:**ORA-01476: divisor is equal to zero****ORA-06512: at line 2**-----
BEGIN

2 DBMS_OUTPUT.PUT_LINE(1 / 0);

3

4 **WHEN ZERO_DIVIDE THEN**

5

DBMS_OUTPUT.PUT_LINE('Division by zero');

6 END;

7 /

Division by zero

PL/SQL procedure successfully completed.

INVALID NUMBER EXCEPTION

1 BEGIN

2 INSERT INTO employees(DEPARTMENT_ID)VALUES('101x');

3 EXCEPTION

4 WHEN INVALID_NUMBER THEN

5 DBMS_OUTPUT.PUT_LINE('Conversion of string to number failed');

6* end;

SQL> /

Conversion of string to number failed

PL/SQL procedure successfully completed.

OTHERS EXCEPTION

1 BEGIN

2 DBMS_OUTPUT.PUT_LINE(1 / 0);

3 EXCEPTION

4 WHEN OTHERS THEN

```
5  DBMS_OUTPUT.PUT_LINE('An exception occurred');
6* END;
7 /
```

An exception occurred
PL/SQL procedure successfully completed.

RESULT:**INFERENCE:**

1. What is an index?
2. What is called remapping of bad sectors?
3. What is a block and a block number?
4. What is called mirroring?
5. What is called bit and block -level striping?

Ex.No.9 Database Design using ER modeling, normalization and Implementation for any application

Aim:

To design a database using ER modeling and Normalization for student portal and sports meet Application

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:

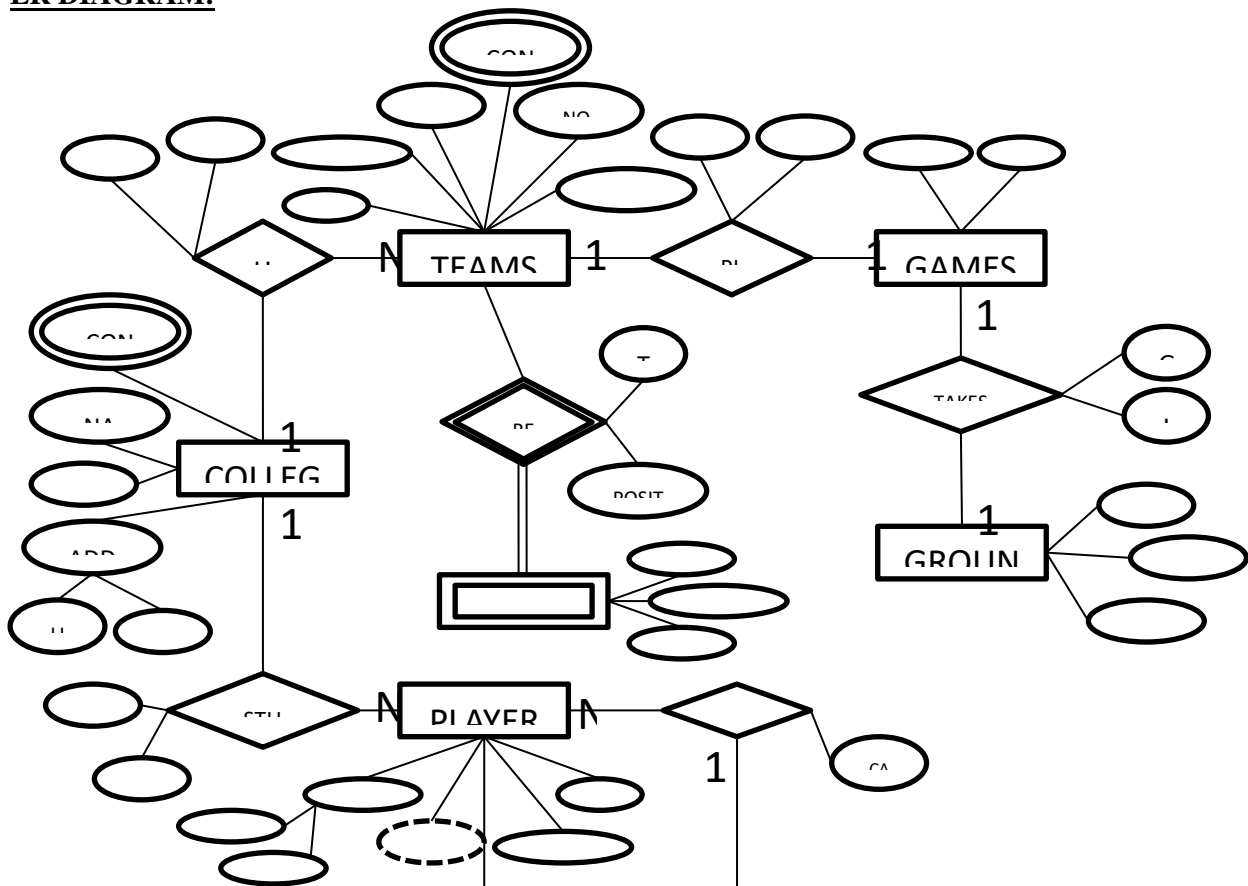
- ✓ Student of – CID, PID
- ✓ Has – TID, CID
- ✓ Plays – TID, GID
- ✓ Takes place – GID, ID
- ✓ Receives – TID, Position

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:**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

*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 form 1

SECOND NORMAL FORM

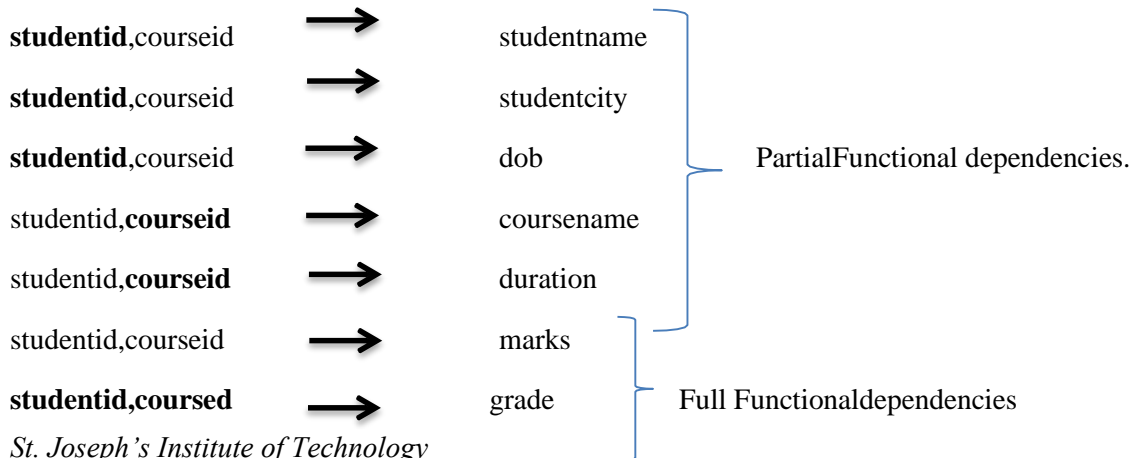
A relation is said to be in second normal form if and only if

*It is in the first normal form and

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

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

From Requirements: (studentid, courseid is Composite Primarykey)



After removing partial functional dependencies from above table

STUDENT

STUDENTID	STUDENTNAME	STUDENTCITY	DOB
-----------	-------------	-------------	-----

COURSE

COURSEID	COURSENAME	DURATION
----------	------------	----------

RESULT

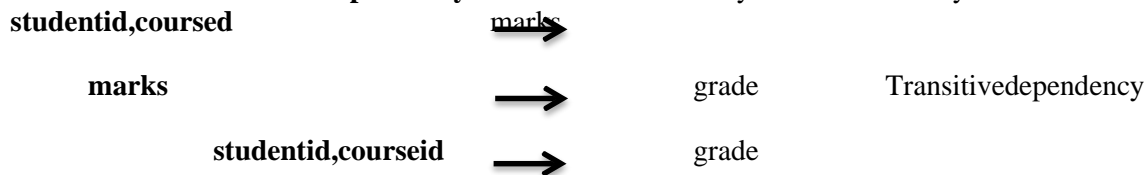
STUDENTID	COURSEID	MARKS	GRADE
-----------	----------	-------	-------

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



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:

INFERENCE:

1. Define the term ACID properties.
2. What are the three kinds of intent locks?
3. What are two pitfalls (problems) of lock-based protocols?
4. What is recovery management component?
5. When is a transaction rolled back?

Ex.No.10**Database Connectivity with Front End Tools****AIM:**

To design and implement a database application for library management system using Netbeans and mysql.(Login Module)

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.

Sample Code**Database:**

```
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;
    }
}
```

Login:

```
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 java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JPasswordField;
```

```
public class AdminLogin extends JFrame {
    static AdminLogin frame;
    private JPanel contentPane;
    private JTextField textField;
    private JPasswordField passwordField;

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

    /**
     * Create the frame.
     */
    public AdminLogin() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);

        JLabel lblAdminLoginForm = new JLabel("Admin Login Form");
        lblAdminLoginForm.setForeground(Color.GRAY);
        lblAdminLoginForm.setFont(new Font("Tahoma", Font.PLAIN, 18));

        JLabel lblEnterName = new JLabel("Enter Name:");

        JLabel lblEnterPassword = new JLabel("Enter Password:");

        textField = new JTextField();
        textField.setColumns(10);

        JButton btnLogin = new JButton("Login");
        btnLogin.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String name=textField.getText();
                String password=String.valueOf(passwordField.getPassword());
                if(name.equals("admin")&&password.equals("admin123")){
                    AdminSuccess.main(new String[]{});
                }
            }
        });
    }
}
```

```

        frame.dispose();
    }else{
        JOptionPane.showMessageDialog(AdminLogin.this, "Sorry, Username
or Password Error","Login Error!", JOptionPane.ERROR_MESSAGE);
        textField.setText("");
        passwordField.setText("");
    }
}
});

passwordField = new JPasswordField();
GroupLayout gl_contentPane = new GroupLayout(contentPane);
gl_contentPane.setHorizontalGroup(
    gl_contentPane.createParallelGroup(Alignment.TRAILING)
        .addGroup(gl_contentPane.createSequentialGroup()

.addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
            .addGroup(gl_contentPane.createSequentialGroup()
                .addGap(124)
                .addComponent(lblAdminLoginForm))
            .addGroup(gl_contentPane.createSequentialGroup()
                .addGap(19)

.addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING)
                .addComponent(lblEnterName)
                .addComponent(lblEnterPassword))
                .addGap(47)

.addGroup(gl_contentPane.createParallelGroup(Alignment.LEADING, false)
                .addComponent(passwordField)
                .addComponent(textField,
GroupLayout.DEFAULT_SIZE, 172, Short.MAX_VALUE))))
            .addContainerGap(107, Short.MAX_VALUE))
        .addGroup(gl_contentPane.createSequentialGroup()
            .addContainerGap(187, Short.MAX_VALUE)
            .addComponent(btnLogin, GroupLayout.PREFERRED_SIZE,
86, GroupLayout.PREFERRED_SIZE)
            .addGap(151))
    );
gl_contentPane.setVerticalGroup(
    gl_contentPane.createParallelGroup(Alignment.LEADING)
        .addGroup(gl_contentPane.createSequentialGroup()
            .addComponent(lblAdminLoginForm)
            .addGap(26)

.addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
            .addComponent(lblEnterName)
            .addComponent(textField,
GroupLayout.PREFERRED_SIZE, GroupLayout.DEFAULT_SIZE,
GroupLayout.PREFERRED_SIZE))
            .addGap(28)

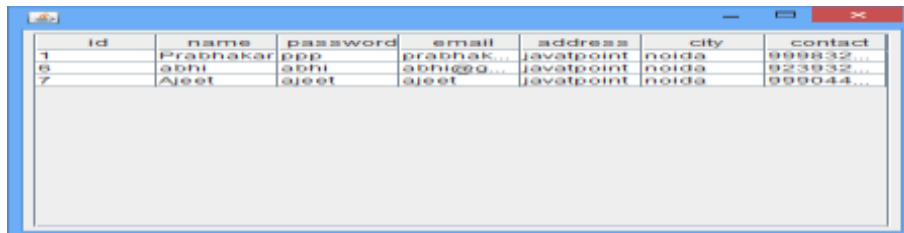
```

```

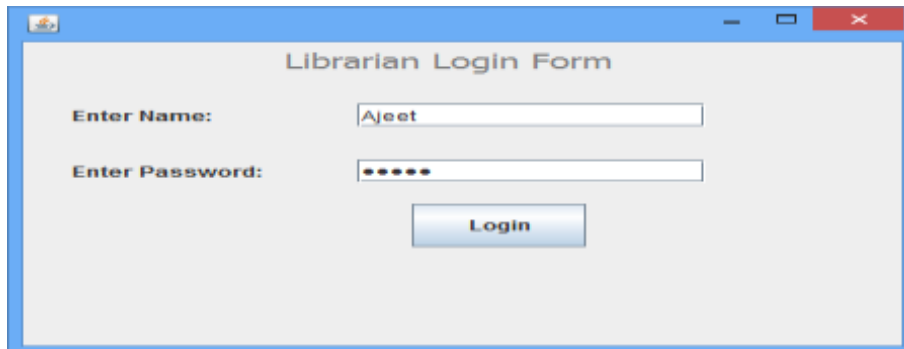
        .addGroup(gl_contentPane.createParallelGroup(Alignment.BASELINE)
            .addComponent(lblEnterPassword)
            .addComponent(passwordField,
                GroupLayout.DEFAULT_SIZE,
                GroupLayout.PREFERRED_SIZE),
            GroupLayout.PREFERRED_SIZE,
            GroupLayout.PREFERRED_SIZE))

        .addGap(18)
        .addComponent(btnLogin, GroupLayout.PREFERRED_SIZE,
            37, GroupLayout.PREFERRED_SIZE)
        .addContainerGap(80, Short.MAX_VALUE))
    );
    contentPane.setLayout(gl_contentPane);
}
}

```

Output:**Database:**


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

Login:


Librarian Login Form

Enter Name:

Enter Password:

RESULT:**INFERENCE:**

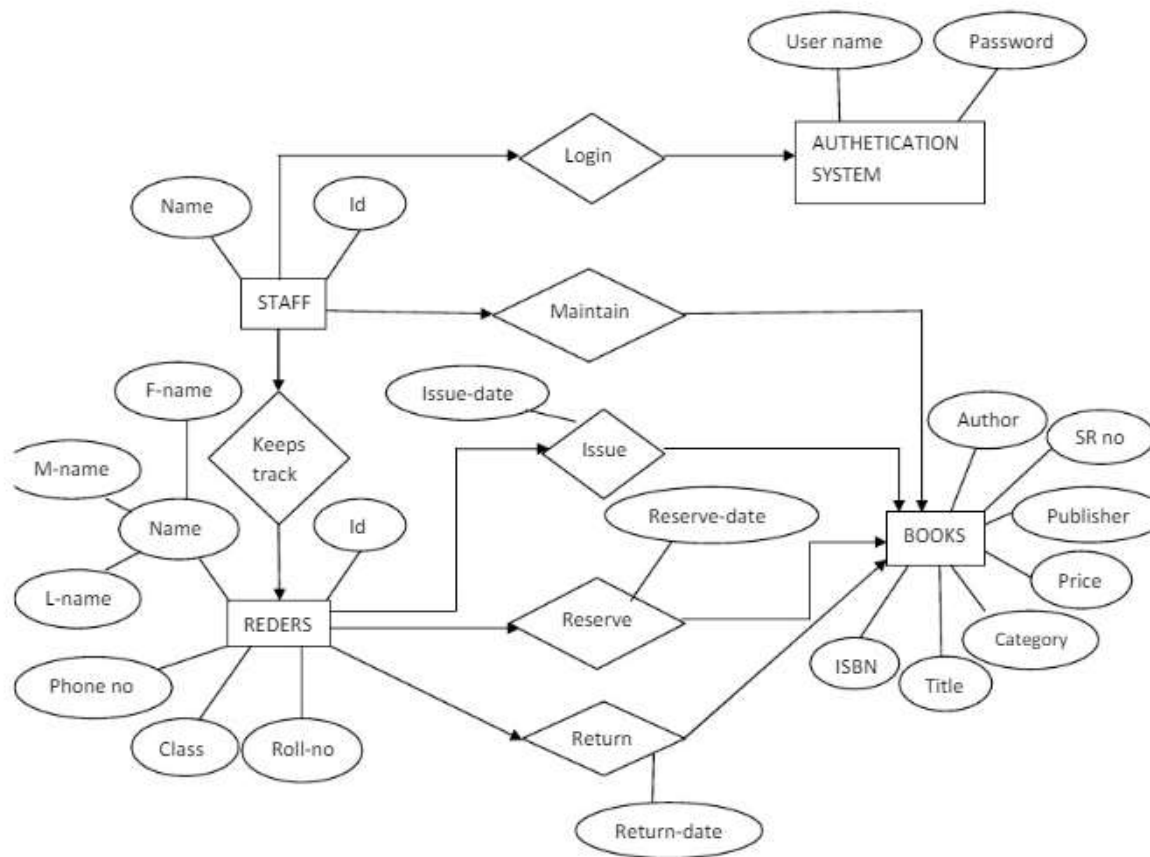
1. Define deadlock?
2. Define upgrade and downgrade?
3. What is a database graph?
4. What are the two methods for dealing deadlock problem?
5. What is a recovery scheme?

Ex.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();
```

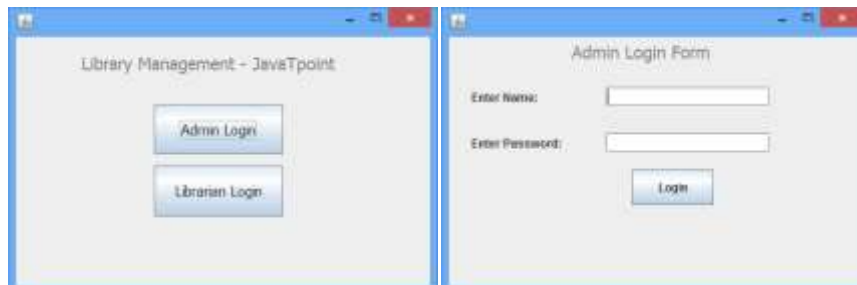
```
textField.setColumns(10);
textField_1 = new JTextField();
textField_1.setColumns(10);
JButton btnReturnBook = new JButton("Return Book");
btnReturnBook.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String bookcallno=textField.getText();
        int studentid=Integer.parseInt(textField_1.getText());
        int i=ReturnBookDao.delete(bookcallno, studentid);
        if(i>0){
            JOptionPane.showMessageDialog(ReturnBook.this,"Book
returned successfully!");

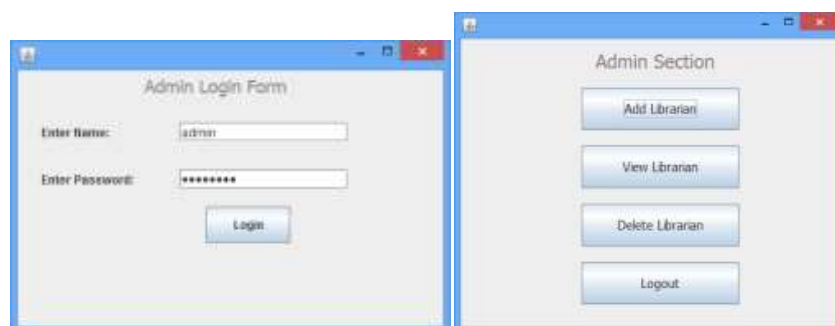
            LibrarianSuccess.main(new String[]{});
            frame.dispose();

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

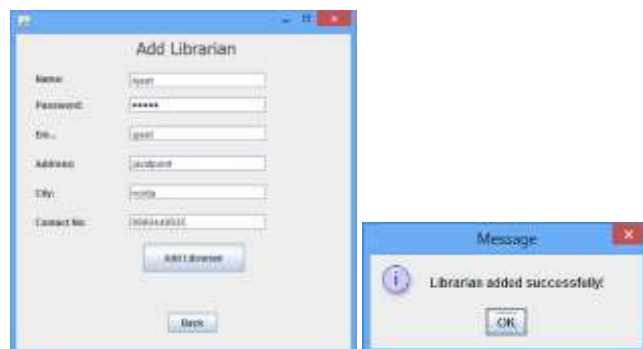
JLabel lblNewLabel = new JLabel("Note: Check the book properly!");
lblNewLabel.setForeground(Color.RED);
lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 13));

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

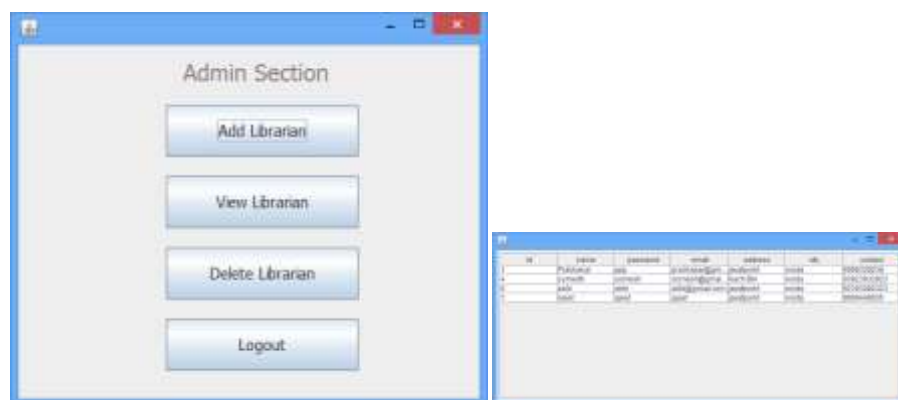
OUTPUT:**LIBRARY MANAGEMENT SYSTEM SCREEN SHOT**



The image shows two windows from a Java application. The first window, titled 'Admin Login Form', has two input fields: 'Enter Name:' with the text 'admin' and 'Enter Password:' with masked characters '*****'. Below these is a 'Login' button. The second window, titled 'Admin Section', contains four buttons stacked vertically: 'Add Librarian', 'View Librarian', 'Delete Librarian', and 'Logout'.

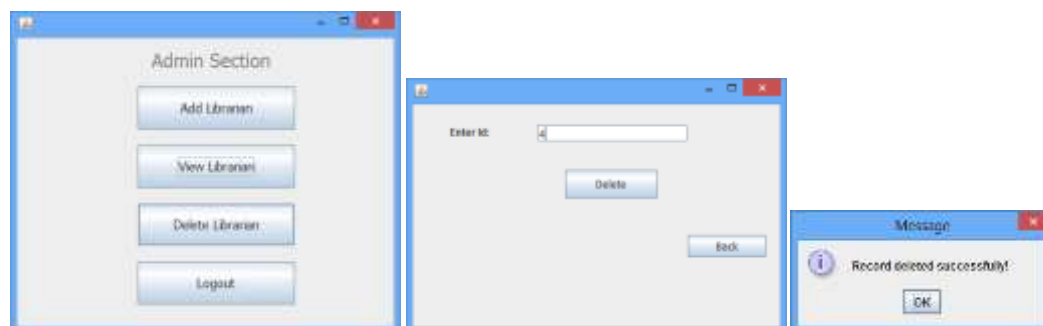


The image shows two windows. The first window, titled 'Add Librarian', has input fields for 'Name:' (ajest), 'Password:' (ajest), 'Email:' (ajest@ajest.com), 'Address:' (javatpoint), 'City:' (noida), and 'Contact No.' (9990444444). It includes an 'Add Librarian' button and a 'Back' button. The second window, titled 'Message', displays an information icon and the text 'Librarian added successfully!' with an 'OK' button.

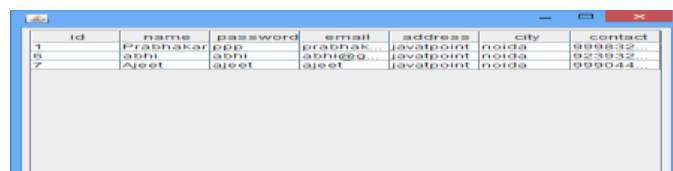


The image shows two windows. The first window, titled 'Admin Section', contains the same four buttons as before: 'Add Librarian', 'View Librarian', 'Delete Librarian', and 'Logout'. The second window displays a table with the following data:

ID	name	password	email	address	city	contact
1	Prabhakar	ppp	prabhakar@ajest.com	javatpoint	noida	999030030
2	ajest	ajest	ajest@ajest.com	javatpoint	noida	999030030
3	ajest	ajest	ajest@ajest.com	javatpoint	noida	999030030
4	ajest	ajest	ajest@ajest.com	javatpoint	noida	999030030

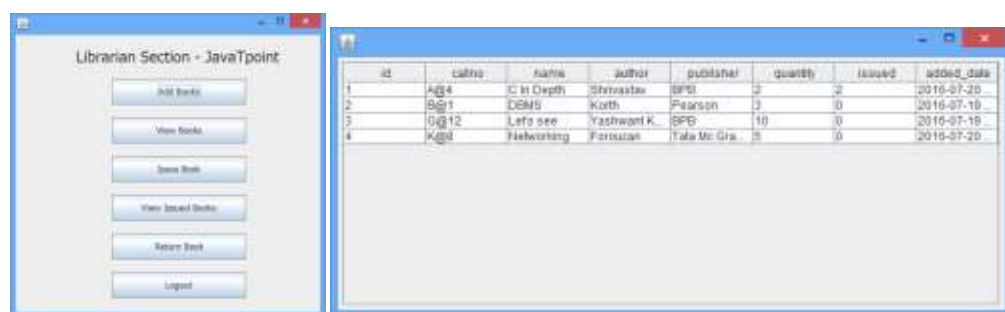
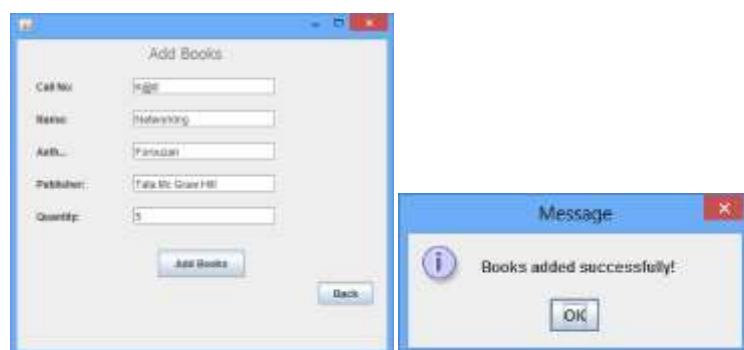
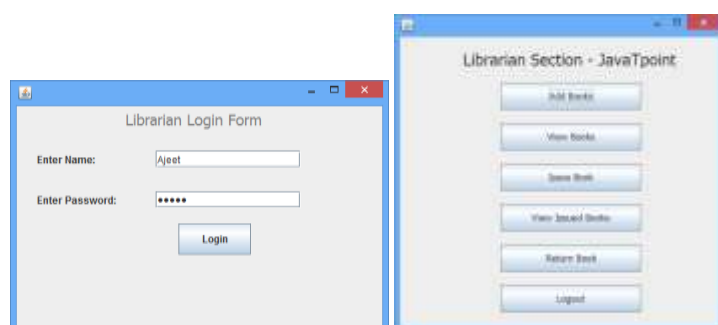
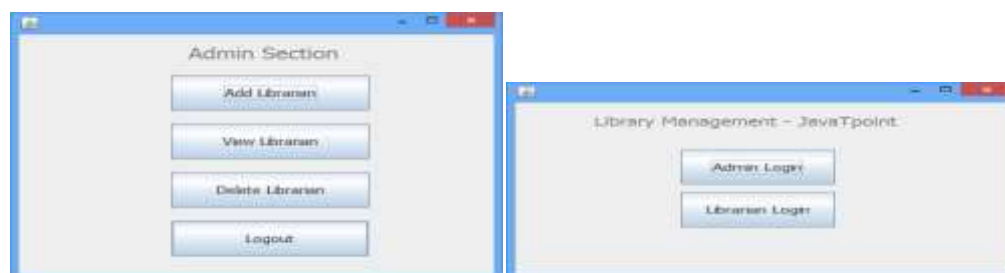


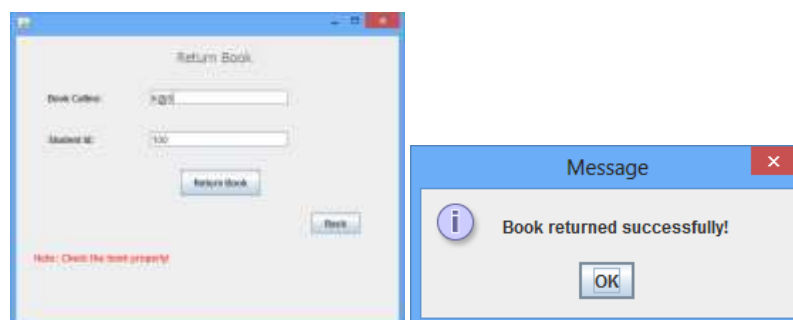
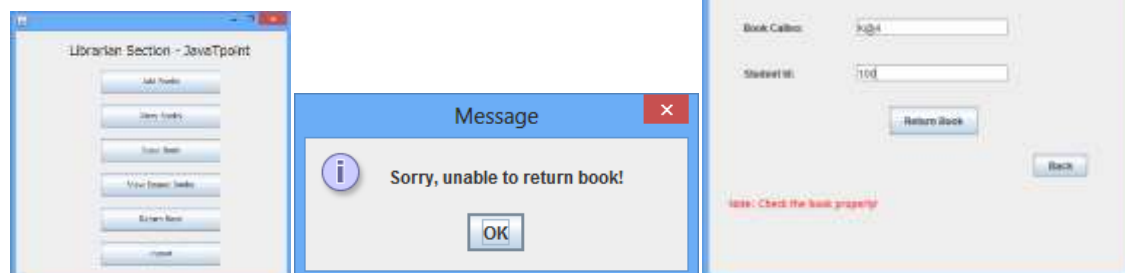
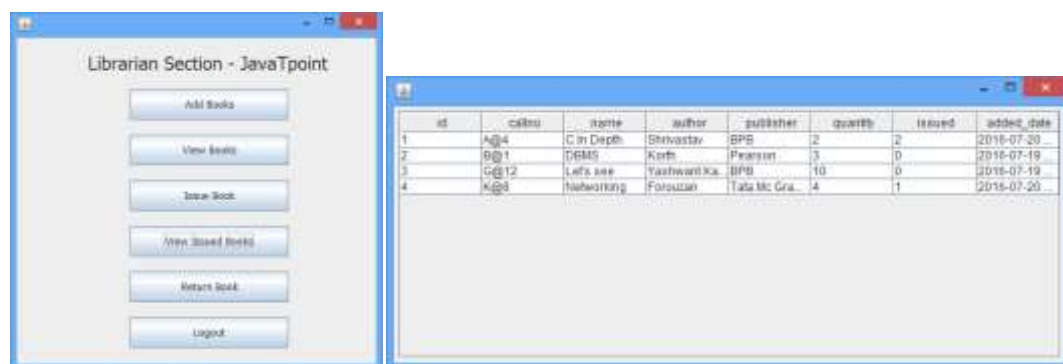
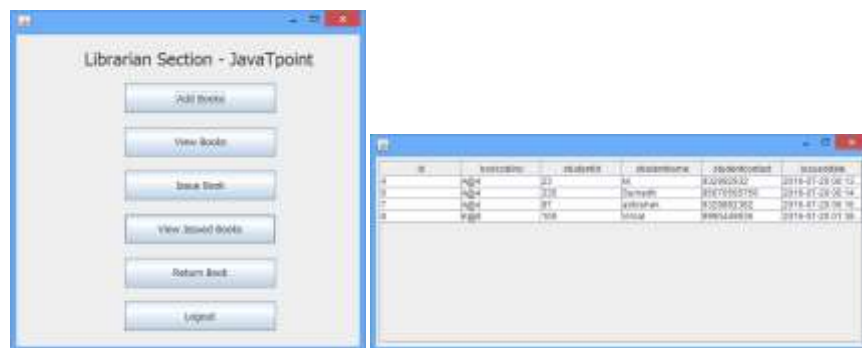
The image shows three windows. The first window, titled 'Admin Section', contains the same four buttons. The second window, titled 'Delete', has an input field 'Enter ID:' with the value '4' and buttons for 'Delete' and 'Back'. The third window, titled 'Message', displays an information icon and the text 'Record deleted successfully!' with an 'OK' button.

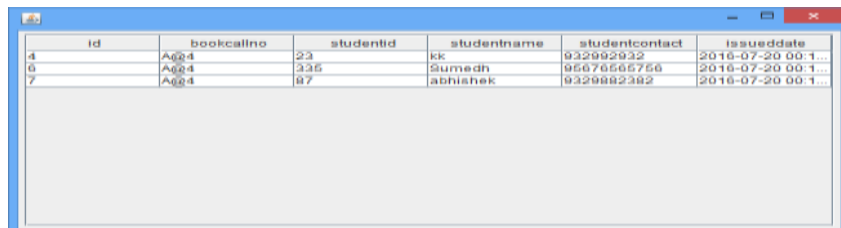


The image shows a table window with the following data:

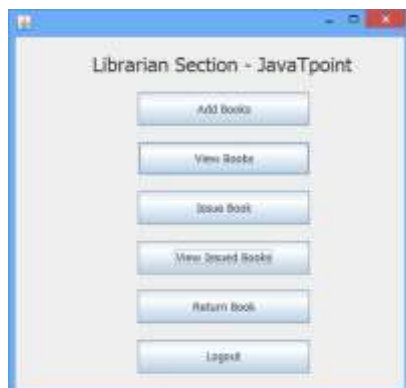
id	name	password	email	address	city	contact
1	Prabhakar	ppp	prabhakar@ajest.com	javatpoint	noida	999030030
2	ajest	ajest	ajest@ajest.com	javatpoint	noida	999030030
7	Ajest	ajest	ajest@ajest.com	javatpoint	noida	999044444







id	bookcalno	studentid	studentname	studentcontact	issueddate
4	A004	23	kk	932992932	2016-07-20 00:1...
6	A004	335	Sumedh	95676566756	2016-07-20 00:1...
7	A004	87	abhishek	9329982382	2016-07-20 00:1...



Librarian Section - JavaTpoint

Add Books

View Books

Issue Book

View Issued Books

Return Book

Logout



id	calno	name	author	publisher	quantity	issued	added_date
1	A004	C in Depth	Shrivastav	BPS	2	2	2016-07-20
2	B001	DBMS	Korth	Pearson	3	0	2016-07-19
3	C002	Let's see	Yashwant K	BPS	10	0	2016-07-19
4	K008	Networking	Forouzan	Tata Mc Gra	5	0	2016-07-20

RESULT:**INFERENCE:**

1. What is the use of with clause in SQL?
2. List the table modification commands in SQL?
3. What is transaction?
4. List the SQL domain Types?
5. What is meant by Cost Estimation?