

**Lab Manual**

*for*

**Database and SQL Lab**

**3159**

**Diploma In Computer Hardware Engineering**

**3<sup>rd</sup> Semester**

*By*

**SITTTR**

**Kalamassery**

## STATE INSTITUTE OF TECHNICAL TEACHERS TRAINING AND RESEARCH

### GENERAL INSTRUCTIONS

Rough record and Fair record are needed to record the experiments conducted in the laboratory. Rough records are needed to be certified immediately on completion of the experiment. Fair records are due at the beginning of the next lab period. Fair records must be submitted as neat, legible, and complete.

#### INSTRUCTIONS TO STUDENTS FOR WRITING THE FAIR RECORD

In the fair record, the index page should be filled properly by writing the corresponding experiment number, experiment name, date on which it was done and the page number.

On the **right side** page of the record following has to be written:

1. **Title:** The title of the experiment should be written in the page in capital letters.
2. In the left top margin, experiment number and date should be written.
3. **Aim:** The purpose of the experiment should be written clearly.
4. **Apparatus/Tools/Equipments/Components used:** A list of the Apparatus/Tools /Equipments /Components used for doing the experiment should be entered.
5. **Principle:** Simple working of the circuit/experimental set up/algorithm should be written.
6. **Procedure:** steps for doing the experiment and recording the readings should be briefly described(flow chart/programs in the case of computer/processor related experiments)
7. **Results:** The results of the experiment must be summarized in writing and should be fulfilling the aim.
8. **Inference :** Inference from the results is to be mentioned.

On the **Left side** page of the record following has to be recorded:

1. **Circuit/Program:** Neatly drawn circuit diagrams/experimental set up.

2. **Design:** The design of the circuit/experimental set up for selecting the components

should be clearly shown if necessary.

3. **Observations:** i) Data should be clearly recorded using Tabular Columns.

ii) Unit of the observed data should be clearly mentioned

iii) Relevant calculations should be shown. If repetitive calculations are needed, only show a sample calculation and summarize the others in a table.

4. **Graphs :** Graphs can be used to present data in a form that shows the results obtained, as one or more of the parameters are varied. A graph has the advantage of presenting large

amounts of data in a concise visual form. Graph should be in a square format.

#### **GENERAL RULES FOR PERSONAL SAFETY**

1. Always wear tight shirt/lab coat , pants and shoes inside workshops.

2. REMOVE ALL METAL JEWELLERY since rings, wrist watches or bands, necklaces, etc. make excellent electrodes in the event of accidental contact with electric power sources.

3. DO NOT MAKE CIRCUIT CHANGES without turning off the power.

4. Make sure that equipment working on electrical power are grounded properly.

5. Avoid standing on metal surfaces or wet concrete. Keep your shoes dry.

6. Never handle electrical equipment with wet skin.

7. Hot soldering irons should be rested in its holder. Never leave a hot iron unattended.

8. Avoid use of loose clothing and hair near machines and avoid running around inside lab .

#### **TO PROTECT EQUIPMENT AND MINIMIZE MAINTENANCE:**

**DO:** 1. SET MULTIRANGE METERS to highest range before connecting to an unknown source.

2. INFORM YOUR INSTRUCTOR about faulty equipment so that it can be sent for repair.

**DO NOT:** 1. Do not MOVE EQUIPMENT around the room except under the supervision of an instructor.

# **INDEX**

<b><u>EX.NO</u></b>	<b><u>LIST OF EXPERIMENTS</u></b>	<b><u>PAGE NO</u></b>
1	Introduction to MYSQL	3
2	Data Definition Language(DDL) commands	4
3	Data Manipulation Language (DML)	8
4	Sub Queries and Joins	11
5	Views	14
6	Procedures	15
7	Cursors	16
8	Triggers	17
9	Normalization	18
10	Checking Normalization	19
11	Java –Mysql Database Connectivity	24
12	PHP-Mysql Database Connectivity	25
13	PHP-Mysql Database Connectivity	27

## **EXPERIMENT NO: 1**

### **Introduction to MYSQL**

#### **AIM:**

To study about Mysql database

#### **OBJECTIVES**

#### **THEORY**

MySQL, is one of the most popular Open Source SQL database management systems. MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company.

MySQL is becoming so popular because of many good reasons:

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table.
- MySQL is customizable.

#### **RESULT**

The mysql database is studied.

## **EXPERIMENT NO:2**

### **DATA DEFINITION LANGUAGE (DDL) COMMANDS**

#### **AIM:**

Consider the database for an organisation. Write the queries for the following

- (i) create the database
- (ii) select the current database
- (iii) Create the following tables.
  - a. **employee** (emp\_no, emp\_name, DOB, address, doj, mobile\_no, dept\_no, salary).
  - b. **department** (dept\_no, dept\_name, location).
- (iv) Include necessary constraints.
- (v) List all the tables in the current database
- (vi) Display the structure of the employee table
- (vii) Add a new column Designation to the employee table
- (viii) Drop the column location from Dept table
- (ix) Drop the tables
- (x) Delete the database

#### **OBJECTIVES**

To understand DDL commands.

#### **THEORY**

##### **Database Queries:**

Before creating any tables, MySQL requires you to create a database by executing the **CREATE DATABASE** command.

- Create a database  
CREATE DATABASE <database name>
- Delete a database  
DROP DATABASE <database name>
- Select the database  
USE <database name>
- List all databases  
SHOW databases;
- Rename a database  
ALTER DATABASE <database name> RENAME <new database name>

## **Table Queries:**

- To Create a table  
CREATE TABLE <tablename> (<fieldname>< fieldtype>(<fieldsize>) , ...)
- List all tables in the current database  
SHOW tables;
- Show table format with column names and data types  
DESCRIBE <table name>
- Modify the structure of table  
ALTER TABLE <table name> <alter specifications>  
ALTER TABLE <table name> DROP COLUMN <column name>  
ALTER TABLE <table name> ADD COLUMN <column name> datatype(<size>)
- Delete the table  
DROP TABLE <table name>

## **Constraints:**

- Primary key → A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column and this constraint create a unique index for accessing the table faster
- UNIQUE → The UNIQUE constraint in Mysql does not allow to insert a duplicate value in a column.
- NOT NULL → In Mysql NOT NULL constraint allows to specify that a column can not contain any NULL value.
- FOREIGN KEY → A FOREIGN KEY in mysql creates a link between two tables by one specific column of both table. The specified column in one table must be a PRIMARY KEY and referred by the column of another table known as FOREIGN KEY.
- CHECK → The CHECK constraint determines whether the value is valid or not from a logical expression.
- DEFAULT → While inserting data into a table, if no value is supplied to a column, then the column gets the value set as DEFAULT

## **PROCEDURE**

- (i) CREATE DATABASE command
- (ii) USE DATABASE command
- (iii) CREATE TABLE command
- (iv) PRIMARY KEY, NOT NULL etc

- (v) SHOW TABLES command
- (vi) DESCRIBE TABLE command
- (vii) ALTER TABLE command
- (viii) ALTER TABLE command
- (ix) DROP TABLE command
- (x) DROP DATABASE command

## **RESULT:**

The DDL commands have been executed successfully.

## **OUTPUT**

### **Problems**

1. Consider the database for a college and design an ER diagram. Write the query for the following.
  - (i) Create the tables:
    - Student (sid, sname, sex, dob,dno)
    - Department (dno, dname)
    - Faculty (F\_id, fname, designation, salary,dno)
    - Course (cid, cname, credits,dno)
    - Register (sid,cid,sem )
    - Teaching (f\_id,cid,sem)
    - Hostel(hid,hname,seats,)
  - (ii) Include the necessary constraints NOT NULL, DEFAULT, CHECK, and PRIMARY KEY, UNIQUE.
  - (iii) Create a database college
  - (iv) Use college as the current database



- (v) Display all the tables in college database
- (vi) Describe the structure of all tables
- (vii) Modify the student table to add a new field 'grade'

2. Consider the database for a banking enterprise. Write the queries for the below questions.

- (i) Create the following tables

Table	Attributes
customer	cid,cname,loc,sex,dob
Bank_brn	bcode,bloc,bsate
Deposit	Dacno,dtype,ddate,damt
Loan	Lacno,ltype,ldate,lamt
Accounts_in	Bcode,cid
depositor	cid,dacno
borrower	cid,lacno

- (ii) Include necessary constraints.
- (iii) Tables are created under the database 'bank'
- (iv) Display all the tables in bank database
- (v) Describe the structure of all tables
- (vi) Delete tables

## **EXPERIMENT NO 3**

### **DATA MANIPULATION LANGUAGE(DML)**

#### **AIM:**

Consider the database for an organisation. Write the queries for the following

- (i) Add 5 rows in the employee and dept tables
- (ii) Display all the records from the above tables
- (iii) Display the empno and name of all the employees from department no2
- (iv) Display empno,name,designation,dept no and salary in the descending order of salary
- (v) Display the empno and name of all employees whose salary is between 2000 and 5000
- (vi) Display all designations without duplicate values.
- (vii) Display the dept name and total salary of employees of each department.
- (viii) Change the salary of employees to 25000 whose designation is 'Typist'
- (ix) Change the mobile no of employee named 'john'
- (x) Delete all employees whose salaries are equal to Rs.7000
- (xi) Select the department that has total salary paid for its employees more than 25000

#### **OBJECTIVES**

- To understand how to insert, update and delete data from within a table.
- ➔ To learn how to retrieve data from a table using the SELECT statement.

#### **THEORY**

##### **1. INSERT**

INSERT INTO *tablename* VALUES (*value1*, *value2*, ..., *valuen*).

##### **2. UPDATE**

UPDATE <table> SET <field1> = <value1> AND <field2> = <value2> WHERE <conditions>

##### **3. DELETE**

DELETE FROM <table> WHERE <condition>

##### **4. SELECT**

- a) Retrieve from all columns  
SELECT \* FROM <table>
- b) Retrieve from selected columns  
SELECT <column 1>, <column 2> FROM <table>
- c) Retrieve unique values  
SELECT DISTINCT <column name> FROM <table>
- d) Retrieve data satisfying a given condition  
SELECT <columns> FROM <tables> WHERE <condition>

#### **PROCEDURE**

- (i) Use insert command
- (ii) Use Select command
- (iii) Use Select command with where condition
- (iv) Use Select command with order by clause
- (v) Use Select command with operators
- (vi) Use Select command with DISTINCT keyword
- (vii) Use Select command with group by clause
- (viii) Use Update command
- (ix) Use Update command
- (x) Use Delete command
- (xi) Use select command with group by and having clause

## **RESULT**

The DML commands are executed successfully.

## **OUTPUT**

## **Problems**

1. Consider the database for a college. Write the query for the following.
  - (i) Insert at least 5 tuples into each table.
  - (ii) List the details of students in the ascending order of date of birth
  - (iii) Display the details of students from computer department
  - (iv) List the faculties in the descending order of salary
  - (v) Display the total number of students in each department
  - (vi) Display the total number of faculties in each department with salary greater than 25000

2. Consider the database for a banking enterprise. Write the queries for the below questions.

- (i) Insert at least 5 tuples in each table
- (ii) Display the branch details
- (iii) List the customers of 'Mumbai' city
- (iv) List the male customers of 'Kolkata' city
- (v) List the state having more than one branch.
- (vi) List the deposit schemes provided by the bank to the customers
- (vii) Delete the entire content of any table

## **EXPERIMENT NO:4**

### **SUB QUERIES AND JOIN**

#### **AIM:**

Consider the database for the organization and Write the queries for the following

- (i) display the empno, name, and salaries for employees whose average salary is higher than the average salary of the organization
- (ii) Display the details of employees whose salary is equal to the minimum salary of organisation.
- (iii) Display all the employees whose designation is same as that of 'Arun'
- (iv) display the empno and name of employees who earn more than any Employee in dept 1.
  
- (v) Display the empno,name , departments that the departments are same in both the emp and dept
- (vi) Display the employee details by implementing left inner join
- (vii) Display employee details by implementing a right outer join

#### **OBJECTIVES**

To understand sub queries and join in Mysql.

#### **THEORY**

##### **NESTED QUERIES:**

A sub query is a query within a query. These sub queries can reside in the WHERE clause, the FROM clause, or the SELECT clause. The first query in the SQL statement is known as the outer query. The query inside the SQL statement is known as the inner query. The inner query is executed first. The output of an inner query is used as the input for the outer query. The entire SQL statement is sometimes referred to as a nested query.

##### **JOINS:**

MySQL **JOINS** are used to retrieve data from multiple tables. A MySQL JOIN is performed whenever two or more tables are joined in a SQL statement.

There are different types of MySQL joins:

- MySQL INNER JOIN (or sometimes called simple join)
- MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
- MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

## INNER JOIN (simple join)

MySQL INNER JOINS return all rows from multiple tables where the join condition is met.

### Syntax

```
SELECT columns  
From table1  
Inner join table2  
On table1.column=table2.column;
```

## LEFT OUTER JOIN

Another type of join is called a MySQL LEFT OUTER JOIN. This type of join returns all rows from the LEFT-hand table specified in the ON condition and **only** those rows from the other table where the joined fields are equal.

### Syntax

```
SELECT columns  
From table1  
left join table2  
On table1.column=table2.column;
```

## RIGHT OUTER JOIN

Another type of join is called a MySQL RIGHT OUTER JOIN. This type of join returns all rows from the RIGHT-hand table specified in the ON condition and **only** those rows from the other table where the joined fields are equal.

### Syntax

### Syntax

```
SELECT columns  
From table1  
Right join table2  
On table1.column=table2.column;
```

## **PROCEDURE**

To be written by the student

## **RESULT:**

The nested queries and joins are executed successfully.

## **Programs**

1. Consider the database for a banking enterprise. Write the queries for the below questions.
  - (i) List the deposit account number and amount in which the deposit scheme having maximum deposit is opened
  - (ii) List the account number and amount of that savings bank deposit scheme in which minimum amount is deposited.
  - (iii) List the customers having accounts in 'Chennai' branch
  - (iv) List the customers having more than one account
  - (v) List the customers having same name but different account numbers.
  - (vi) List the customer name that is having maximum deposit account in bank
  - (vii) List the customer who has borrowed highest amount of home loan
  - (viii) Display the customer details by implementing left inner join
  - (ix) Display the customer details by implementing a right outer join
  
2. Consider the database for a college. Write the queries for the below questions.
  - (i) List out the ID, Name and Date of Birth of students registered for a specific course.
  - (ii) List out the ID, Name and Date of Birth of students registered for a specific course, staying in a specific Hostel.
  - (iii) List the names of faculties who teach for a specific course.
  - (iv) Display the student details by implementing left inner join
  - (v) Display the student details by implementing a right outer join

## **EXPERIMENT NO:5**

### **VIEWS**

#### **AIM:**

Write the queries for the following

- (i) Create a view emp from employee such that it contains only emp\_no and emp\_name and department.
- (ii) Create a view dept from department with only dept\_no and location.
- (iii) Create a view that contains the details of employees who are managers only.
- (iv) drop the views.

#### **OBJECTIVES**

To understand views in Mysql

#### **THEORY**

A view is the tailored presentation of data contained in one or more table and can also be said as restricted view to the data in the tables. A view is a “virtual table” or a “stored query” which takes the output of a query and treats it as a table. The table upon which a view is created is called as base table. A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed. The tables on which a view is based are called base tables. The view is stored as a SELECT statement in the data dictionary.

##### **Advantages of a view:**

- a. Additional level of table security.
- b. Hides data complexity.
- c. Simplifies the usage by combining multiple tables into a single table

#### **Creating and dropping view:**

##### **Syntax:**

Create or replace view    view\_name AS SELECT column\_name(s) FROM table\_name  
WHERE condition;

Drop view <view name>;

#### **PROCEDURE**



- 1) Create the employee table;
- 2) Create the view
- 3) display the content of view

### **Result**

Thus the views are created successfully

### **Problems**

1. Create and drop views on student table
2. Create and drop views on customer and deposit table

## **EXPERIMENT NO:6** **PROCEDURE**

### **AIM:**

Write a procedure which increases the salary of an employee. It accepts an employee number and salary increase amount. It uses the employee number to find the current salary from the EMPLOYEE table and update the salary.

### **OBJECTIVES**

To understand procedure in Mysql

### **THEORY**

### **PROCEDURE:**

In MySQL, a procedure is a stored program that you can pass parameters into. It does not return a value like a function does.

### **Syntax**

Create procedure procedure name (parameter data type, parameter data type...)

Begin

Declaration section

Executable \_section

End;

### **Procedure name**

The name to assign to this procedure in MySQL.

### **Parameter**

When creating a procedure, there are three types of parameters that can be declared:

1. **IN** - The parameter can be referenced by the procedure. The value of the parameter cannot be overwritten by the procedure.
2. **OUT** - The parameter cannot be referenced by the procedure, but the value of the parameter can be overwritten by the procedure.
3. **IN OUT** - The parameter can be referenced by the procedure and the value of the parameter can be overwritten by the procedure.

**Declaration section**

The place in the procedure where you declare local variables.

**Executable section**

The place in the procedure where you enter the code for the procedure.

**PROCEDURE:**

- i. Write the procedure with empno and increment name as parameter
- ii. Use update command to increment salary

**OUTPUT**

**RESULT:**

The procedures are executed successfully.

**Problems**

1. Write a procedure which accept the account number of a customer and retrieve the balance.
2. Write a procedure which accepts the student number and displays the department in which he belongs to.

**EXPERIMENT NO: 7**  
**CURSOR**

**AIM**

Write a cursor to display the list of employees who are working as managers.

## **OBJECTIVES**

To implement cursor

## **THEORY**

A cursor is a SELECT statement that is defined within the *declaration* section of your stored program in MySQL.

### **1. Declare a cursor**

Declare cursor name cursor for select statement;

### **2. Open the cursor.**

Open cursor name;

### **3. Fetch cursor**

The purpose of using a cursor, in most cases, is to retrieve the rows from your cursor so that some type of operation can be performed on the data.

Fetch cursor name into variable list;

### **4. Close the cursor**

Close cursor name;

## **PROCEDURE**

To be written by the student

## **OUTPUT**

The cursor is executed successfully.

## **RESULT**

## **PROGRAMS**

1. Create a cursor to modify the salary of 'Professors' belonging to all departments by 150%.
2. Consider the college database. Retrieve all students who have registered for a specific course and store their details into another table using cursors.
3. Consider the bank database .Retrieve all customers who have loan at a particular branch using cursor.

## **EXPERIMENT NO: 8** **TRIGGER**

### **AIM**

Write a Trigger for employee table it will store the updated salary into another table SALARY while updating salary.

## **OBJECTIVES**

To understand triggers in Mysql

### **THEORY.**

A trigger is a set of actions that are run automatically when a specified change operation (INSERT, UPDATE, or DELETE statement) is performed on a specified table.

The syntax to create an AFTER UPDATE Trigger in MySQL is:

Create trigger trigger\_name

After update

On tablename for each row

Begin

Variable declarations

Trigger code

End;

### **PROCEDURE**

- (i) Create the employee table
- (ii) Insert values
- (iii) Write the after update trigger
- (iv) Update the salary in employee table
- (v) Display the SALARY table

### **RESULT:**

The trigger procedure has been executed successfully.

### **OUTPUT**

### **Problems**

1. Write an update trigger on Account table. The system should keep track of the records that are being updated.
2. Write a before delete trigger on student table.

## **Experiment No: 9**

### **CONCEPTS OF NORMALIZATION**

**AIM:** Checking Normalization of a database table (First Normal form)

**Problem Statement:**

An exercise to check whether the given database table is normalized or not. If yes find out the status of normalization and reasoning.

**Objective:**

To study the concept of various levels of normalization and understand how to convert into normalized forms.

**Requirements:** Mysql database software

**Design/Theory**

Create a database table in SQL with a few no of rows and columns. Analyze the table and determine to which normal form it belongs to according to the rules and regulations of each normal forms.

Procedure:

Consider a student table as given below

Social Security Number	FirstName	LastName	Major
123-45-6789	Jack	Jones	Library and Information Science
222-33-4444	Lynn	Lee	Library and Information Science
987-65-4321	Mary	Ruiz	Pre-Medicine
123-54-3210	Lynn	Smith	Pre-Law

We can easily verify that this table satisfies the definition of 1NF: viz., it has no duplicated rows; each cell is single-valued (i.e., there are no repeating groups or arrays); and all the entries in a given column are of the same kind. In this table we can see that the key, SSN, functionally determines the other attributes; i.e., FirstName, LastName, and Major. .

**Experiment No: 10****Checking Normalization of a database table (Third normal form).****Problem Statement:**

An exercise to check whether the given database table is normalized or not. If yes find out the status of normalization and reasoning.

**Objective:**

To study the concept of various levels of normalization and understand how to convert into normalized forms.

**Requirements:** Mysql database software

### Design/Theory

Create a database table in SQL with a few no of rows and columns. Analyze the table and determine to which normal form it belongs to according to the rules and regulations of each normal forms.

### Procedure:

Consider a book database table as given below.

Author Last Name	Author First Name	Book Title	Subject	Collection Library	or	Building
Berdahl	Robert	The Politics of the Prussian Nobility	History	PCL Stacks	General	Perry-Castañeda Library
Yudof	Mark	Child Abuse and Neglect	Legal Procedures	Law Library		Townes Hall
Harmon	Glynn	Human Memory and Knowledge	Cognitive Psychology	PCL Stacks	General	Perry-Castañeda Library
Graves	Robert	The Golden Fleece	Greek Literature	Classics Library		Waggener Hall
Miksa	Francis	Charles Cutter	Ammi Library Biography	Library and Information Science Collection		Perry-Castañeda Library
Hunter	David	Music Publishing and Collecting	Music Literature	Fine Arts Library		Fine Arts Building
Graves	Robert	English and Scottish Ballads	Folksong	PCL Stacks	General	Perry-Castañeda Library

By examining the table, we can infer that books dealing with history, cognitive psychology, and folksong are assigned to the PCL General Stacks collection; that books dealing with legal procedures are assigned to the Law Library; that books dealing with Greek literature are assigned to the Classics Library; that books dealing with library biography are

assigned to the Library and Information Science Collection (LISC); and that books dealing with music literature are assigned to the Fine Arts Library.

Moreover, we can infer that the PCL General Stacks collection and the LISC are both housed in the Perry-Castañeda Library (PCL) building; that the Classics Library is housed in Waggener Hall; and that the Law Library and Fine Arts Library are housed, respectively, in Townes Hall and the Fine Arts Building.

Thus we can see that a transitive dependency exists in the above table : any book that deals with history, cognitive psychology, or library biography will be physically housed in the PCL building (unless it is temporarily checked out to a borrower); any book dealing with legal procedures will be housed in Townes Hall; and so on. In short, if we know what subject a book deals with, we also know not only what library or collection it will be assigned to but also what building it is physically housed in.

A problem with transitive dependency is that, there is duplicated information: from three different rows we can see that the PCL General Stacks are in the PCL building. For another thing, we have possible deletion anomalies: if the Yudof book were lost and its row removed from table, we would lose the information that books on legal procedures are assigned to the Law Library and also the information the Law Library is in Townes Hall. As a third problem, we have possible insertion anomalies: if we wanted to add a chemistry book to the table, we would find that the above table nowhere contains the fact that the Chemistry Library is in Robert A. Welch Hall. As a fourth problem, we have the chance of making errors in updating: a careless data-entry clerk might add a book to the LISC but mistakenly enter Townes Hall in the building column.

To solve this problem decompose the above table into three different tables as follows

Table A

Author Last Name	Author First Name	Book Title
Berdahl	Robert	The Politics of the Prussian Nobility
Yudof	Mark	Child Abuse and Neglect
Harmon	Glynn	Human Memory and Knowledge
Graves	Robert	The Golden Fleece

Miksa	Francis	Charles Ammi Cutter
Hunter	David	Music Publishing and Collecting
Graves	Robert	English and Scottish Ballads

Table B

Book Title	Subject
The Politics of the Prussian Nobility	History
Child Abuse and Neglect	Legal Procedures
Human Memory and Knowledge	Cognitive Psychology
The Golden Fleece	Greek Literature
Charles Ammi Cutter	Library Biography
Music Publishing and Collecting	Music Literature
English and Scottish Ballads	Folksong

Table C

Subject	Collection or Library
History	PCL General Stacks
Legal Procedures	Law Library
Cognitive Psychology	PCL General Stacks
Greek Literature	Classics Library



Library Biography	Library and Information Science Collection
Music Literature	Fine Arts Library
Folksong	PCL General Stacks

Table D

Collection or Library	Building
PCL General Stacks	Perry-Castañeda Library
Law Library	Townes Hall
Classics Library	Waggener Hall
Library and Information Science Collection	Perry-Castañeda Library
Fine Arts Library	Fine Arts Building

## Experiment No: 11

### Database Connectivity

#### Java Database Connectivity

**Aim:** To understand the java database connectivity

#### Problem Statement:

A program which connects to an online book database table and select the tuple and display it.

#### Objective:

To understand the connectivity to a database table using java. In java database is accessed through java database connectivity(JDBC).

**Requirements:** Mysql database software, Java software.

#### Design/Theory:

The four main components required for implementation of JDBC are application, driver manager, data source specific drivers and corresponding data sources. The application program establishes/terminates connection with data. Driver manager will load JDBC drivers and pass JDBC function calls from the application to the driver. The data source processes commands from the driver and return the results.

1. Drivers for the database are loaded by using Class.forName.
2. The getConnection() function of DriverManager class of JDBC is used to create the connection object. The URL specifies the machine name(db.onlinebook.edu)

```
public static void Sample(String DB_id, String U_id, String Pword)
{
    String URL ="jdbc:oracle:oci8:@db.onlinebook.edu:100:onbook_db";
    Class.forName("oracle.jdbc.driver.OracleDriver");
    Connection cn=DriverManager.getConnection(URL,U_id, Pword);
```

```

Statement st = Cn.createStatement();
try
{
    st.executeUpdate("INSERT INTO AUTHOR
VALUES('A010','SMITH','JONES','TEXAS')");
}
catch(SQLException se)
{
    System.out.println("Tuple cannot be inserted."+se);
}
ResultSet rs = st.executeQuery("SELECT Aname,State from AUTHOR WHERE City =
'Seattle'");
while(rs.next())
{
    System.out.println(rs.getString(1) + " "+rs.getString(2));
}
st.close();
Cn.close();
}

```

## Experiment No:12

# DATABASE CONNECTIVITY

### AIM:

To create a PHP-Mysql Program to enter data into Mysql

### Problem Statement:

A php program that connects to a database table and insert values into the table.

### Objective:

To understand the connectivity to a database table using php and how to insert new values into that table.

### Requirements:

Mysql database software, LAMP Server and Html

### Design/Theory:

In an existing or a newly created database an employee table is formed with attributes firstname, lastname and email. A php-html program is used to insert values into the table.

Php code:

```
<?php
```

```
$host="localhost"; // Host name
$username=""; // Mysql username
$password=""; // Mysql password
$db_name="test"; // Database name
$tbl_name="test_mysql"; // Table name
```

```
// Connect to server and select database.
mysql_connect("$host", "$username", "$password")or die("cannot connect");
mysql_select_db("$db_name")or die("cannot select DB");
```

```
// Get values from form
$name=$_POST['name'];
$lastname=$_POST['lastname'];
$email=$_POST['email'];
```

```
// Insert data into mysql
$sql="INSERT INTO $tbl_name(name, lastname, email)VALUES('$name', '$lastname', '$email')";
$result=mysql_query($sql);
```

```
// if successfully insert data into database, displays message "Successful".
if($result){
    echo "Successful";
    echo "<BR>";
    echo "<a href='insert.php'>Back to main page</a>";
}
else {
    echo "ERROR";
}
?>
```

```
<?php
```

```
// close connection
```

```
mysql_close();
```

```
?>
```

```
<html>
```

```
<body>
```

```
<form action= "<?php $_php_self ?>" method = "GET">
```

```
    Empid: Empname: Empage: Dept: City
```

```
    <input type="text" name="Empid" />
```

```
    <input type="text" name="Empname" />
```

```
    <input type="text" name="Empage" />
```

```
    <input type="text" name="Dept" />
```

```
    <input type="text" name="City" />
```

```
<input type="submit" value="Show Result" />
</form>
</body></html>
```

## **EXPERIMENT NO: 13**

### **DATABASE CONNECTIVITY**

#### **AIM**

Create a php program that allows to enter the employee details into a database.

To create an application program that process a query which returns the grade result of a student after processing the marks table.

#### **OBJECTIVES**

To understand PHP mysql database connectivity.

To study the concept of connecting database using an application program (PHP) and process results after manipulating the database table

Requirements:

Mysql database software, LAMP Server and html.

#### **PROCEDURE**

Create a database in Mysql. Create a 'marks' table with fields regno, name, batch, mark1, mark2 and mark3. Write a function which finds the total marks and if the total marks is greater than 225 then the result is categorized as “Distinction”, if the total marks is less than 225 , the result is “firstclass”, if the total marks is less than 180, the result is “second class”, if the total marks is less than 150 the result is “passed”. If the marks in any one of the subject is less than 40 the result is “Failed”.

1. create a form in PHP/JAVA
2. create a database in mysql
3. provide the database connectivity
4. retrieve the details of employee through forms

php code:

```

        <?php

$user_name = "root";

$password = "";

$server = "localhost";

$database = "mysql";

$regno = $_POST['regno'];

//$regno = 1;

$db_handle = mysql_connect($server, $user_name, $password);

if (!$db_handle)

{

    die('Could not connect: ' . mysql_error());

}

//echo 'Connected successfully';

$db_found = mysql_select_db($database);

if ($db_found)

{

    //echo "Database found";

    $SQL = "SELECT * FROM marks WHERE regno = ".$regno."";

    $result_set = mysql_query($SQL);

    $record = mysql_fetch_array($result_set);

    echo "<BR>MARKLIST";

    echo "<BR>Reg No...".$record['regno'];

    echo "<BR>Name....".$record['name'];

    echo "<BR>Group....".$record['batch'];

    echo "<BR>Mark1....".$record['mark1'];

    echo "<BR>Mark2....".$record['mark2'];

    echo "<BR>Mark3....".$record['mark3'];

    echo "                                "<BR>Result...".

compute_result($record['mark1'],$record['mark2'],$record['mark3']);

```

```

    }
else
{
    echo "<BR>Database not Found";
}
mysql_close($db_handle);
function compute_result($m1, $m2, $m3)
{
    $tmarks = $m1 + $m2 + $m3;
    if (($m1<40) || ($m2<40) || ($m3<40))
        $result = "Failed";
    elseif ($tmarks < 150)
        $result = "Passed";
    elseif ($tmarks<180)
        $result = "Second Class";
    elseif ($tmarks<225)
        $result = "First Class";
    else
        $result = "Distinction";
    return $result;
}
?>

```

Html code:

```

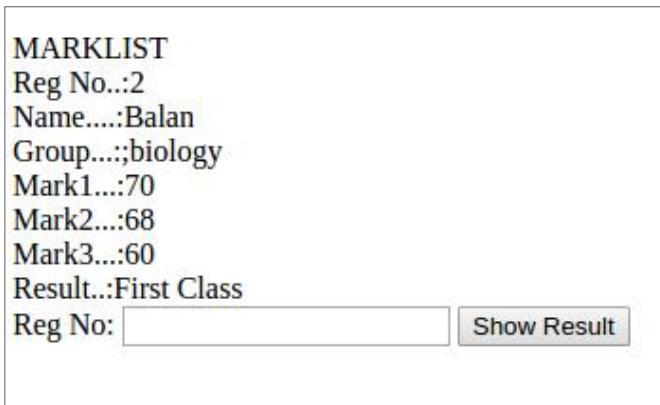
<html>
<body>

```

```
<FORM Method = "post" Action="connect1.php">  
Reg No:  
<INPUT Type = "text" name = "regno"> <BR>  
<INPUT type = "submit" value = "Show Result">  
</FORM>  
</body>  
</html>
```

## **OUTPUT**

## **RESULT**



MARKLIST  
Reg No...:2  
Name.....:Balan  
Group....;biology  
Mark1....:70  
Mark2....:68  
Mark3....:60  
Result...:First Class  
Reg No: