|  |  |  |
| --- | --- | --- |
| **SL** | **Experiment Name** | **Page** |
| **01.** | Study and Implementation of DML Commands of SQL with Suitable Example  • Insert  • Delete  • Update | **02** |
| **02.** | Study and Implementation of DDL Commands of SQL with Suitable Example  • Create  • Alter  • Drop | **04** |
| **03.** | Study and Implementation of DML Commands of  • Select Clause  • From Clause  • Where Clause | **06** |
| **04.** | Study and Implementation of DML Commands of  • Group By & Having Clause  • Order By Clause  • Create View, Indexing & Procedure Clause | **08** |
| **05.** | Study and Implementation of SQL Commands of Join Operations with Example  • Cartesian Product  • Natural Join  • Left Outer Join  • Right Outer Join  • Full Outer Join | **10** |
| **06.** | Study and Implementation of Aggregate Function with Example  • Count Function  • Max Function  • Min Function  • Avg Function | **13** |
| **07.** | Study and Implementation of Triggering System on Database Table Using SQL Commands with Example. | **15** |
| **08.** | Study and Implementation of SQL Commands to Connect MySQL Database with Java or PHP. | **17** |

**INDEX**

**Experiment Number: 01**

**Experiment Name:** Study and Implementation of DML Commands of SQL with Suitable Example

• Insert

• Delete

• Update

**Objectives:** To understand and use data manipulation language to write query for database.

**Theory:** Data Manipulation Language (DML) commands in SQL are fundamental for managing and manipulating data within a database. The primary DML commands include INSERT, DELETE, and UPDATE, each serving a specific purpose in modifying data stored in tables.

1. **INSERT Command:** The INSERT command is used to add new records (rows) into a table. It allows the user to specify the values for each column or a subset of columns when adding new data to the database.

* **Syntax:** INSERT INTO table\_name (column1, column2, ...) VALUES (value1, value2, ...);
* **Example:** INSERT INTO Students (student\_name, age, grade) VALUES ('Alice', 20, 'A');

1. **DELETE Command:** The DELETE command is employed to remove specific records (rows) from a table based on certain conditions specified in the WHERE clause. It deletes records that match the condition provided.

* **Syntax:** DELETE FROM table\_name WHERE condition;
* **Example:** DELETE FROM Students WHERE student\_id = 1;

1. **UPDATE Command:** The UPDATE command allows the user to modify existing records in a table. It changes the values of specific columns in selected rows that meet the specified conditions.

* **Syntax:** UPDATE table\_name SET column1 = value1, column2 = value2, ... WHERE condition;
* **Example:** UPDATE Students SET age = 21 WHERE student\_name = 'Alice';

The implementation of DML commands in SQL provides a powerful way to manipulate the data stored in a database. INSERT adds new records, DELETE removes specific records, and UPDATE modifies existing records. These commands are crucial for maintaining data accuracy and integrity within a database.

**Code:**

|  |
| --- |
| use master  create database university  use university  -- For checking the existing tables within the database  SELECT table\_name  FROM information\_schema.tables  WHERE table\_type = 'BASE TABLE';  create table department(  dept\_name varchar(20),  building varchar (15),  budget numeric(8,2),  primary key(dept\_name)  );  --inserting  insert into department(dept\_name, building, budget) values('BCE','Sajeeb',87000)  insert into department values('CSE','Engineering',90000)  insert into department values('EEE','JHON',95000)  insert into department values('EECE','Watson',80000)  insert into department values('BANGLA','BANGLA',68000)  insert into department values('ENGLISH','ENGLISH',55000)  select \* from department  --deleting  delete from department where dept\_name ='CSE'  select \* from department  --update  update department set budget = budget\*1.05 where budget < 90000  select \* from department |

**Output:**

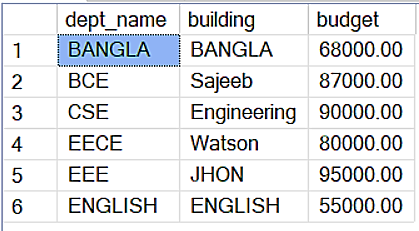
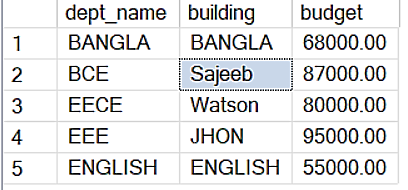
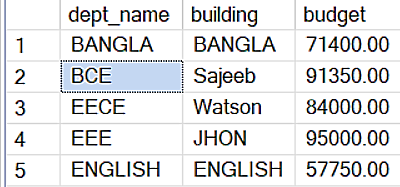
****

Figure 03: Updating salary on department table

Figure 02: Delete from department table

Figure 01: Insert into department table

**Experiment Number: 02**

**Experiment Name:** Study and Implementation of DDL Commands of SQL with Suitable Example

• Create

• Alter

• Drop

**Objectives:** To understand and use data definition language to write query for database.

**Theory:** In database management systems (DBMS), DDL is a subset of SQL (Structured Query Language) used to define, modify, and remove the structure of database objects. It primarily includes commands for creating, altering, and dropping objects within a database. The main operations included in DDL are:

1. **CREATE:** This command is used to create new database objects, such as tables, views, indexes, or stored procedures. For instance, CREATE TABLE, CREATE INDEX, CREATE VIEW, etc., are examples of creating database objects.
2. **ALTER:** The ALTER command is used to modify the structure of existing database objects. It enables changes such as adding, modifying, or dropping columns in a table, altering constraints, or changing the data types.
3. **DROP:** The DROP command is used to delete or remove existing database objects such as tables, views, or indexes. This operation is irreversible and permanently removes the object from the database.

These DDL commands are crucial for defining the structure of the database, including the organization of data, constraints, relationships, and indices. They are responsible for managing the overall architecture of the database and play a significant role in database design, maintenance, and administration.

**Code:**

|  |
| --- |
| use master  create database Sajeeb  use Sajeeb  create table instructor(  ID varchar(20),  name varchar(15) not null,  dept\_name varchar(15),  salary numeric(8,2),  primary key(ID)  );  insert into instructor(ID,name,dept\_name,salary) values('10121','Rasel','ICE','86000')  insert into instructor values('10122','MITU','CSE','80000')  insert into instructor values('10123','Naima','EEE','70000')  insert into instructor values('10124','Rani','EECE','90000')  insert into instructor values('10125','Kumari','CE','95000')  insert into instructor values('10126','Supriya','Arct','68000')  select \* from instructor  alter table instructor add course\_id varchar(20)  select \* from instructor  drop table instructor |

**Output:**

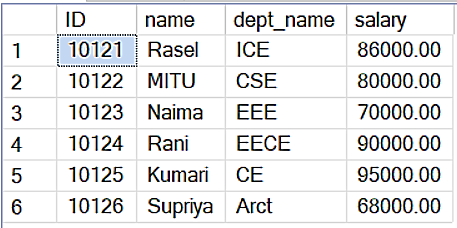


Figure-01: Creating a table

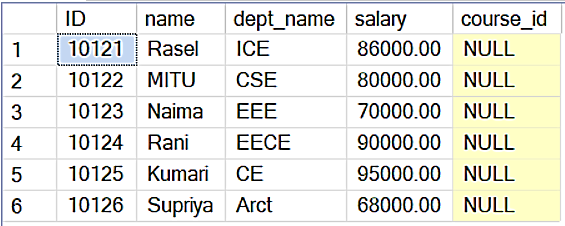


Figure-02: Alter table and add course\_id



Figure-03: After dropping the table

**Experiment Number: 03**

**Experiment Name:** Study and Implementation of DML Commands of

• Select Clause

• From Clause

• Where Clause

**Objectives**:

1. To understand and use of the SQL queries.
2. To study how to implement select, from and where clause in database.

**Theory:** Data Manipulation Language (DML) commands in SQL are used to retrieve, manipulate, and manage data stored in a database. The primary components of a DML query include the SELECT, FROM, and WHERE clauses.

1. **SELECT Clause:** The SELECT clause is used to specify the columns that will be returned in the query result. It retrieves data from the database based on the specified columns.

* **Syntax:** SELECT column1, column2, … FROM table\_name;
* **Example:** SELECT student\_name, age FROM Students;

1. **FROM Clause:** The FROM clause specifies the table or tables from which data will be retrieved. It indicates the source of the data that the SELECT statement will be querying.
   * **Syntax:** SELECT column1, column2, …

FROM table\_name;

* + **Example:** SELECT \* FROM Students;

1. **WHERE Clause:** The WHERE clause is used to filter rows based on specific conditions. It allows for the retrieval of data that meets certain criteria.

* **Syntax**: SELECT column1, column2, …

FROM table\_name

WHERE condition;

* **Example:** SELECT student\_name, age

FROM Students

WHERE grade = ‘A’;

The SELECT, FROM, and WHERE clauses are fundamental components of a SQL query. They enable the selection of specific columns, specify the source table, and filter data based on given conditions, respectively.

**Code:**

|  |
| --- |
| use master  create database SajeebRay  use SajeebRay  create table university(  dept\_name varchar(15),  building varchar(15),  budget numeric(8,2)  primary key(dept\_name)  );  insert into university values('ICE','Engineering',87000)  insert into university values('CSE','Engineering',90000)  insert into university values('EEE','JHON',95000)  insert into university values('EECE','Watson',80000)  insert into university values('BANGLA','BANGLA',68000)  insert into university values('ENGLISH','ENGLISH',55000)  select dept\_name, building from university  select dept\_name from university  select dept\_name, budget from university where dept\_name = 'EECE' |

**Output:**

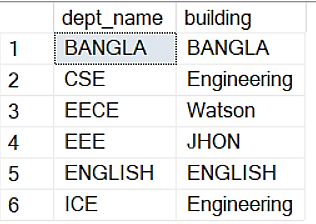
****

Figure-01: Using SELECT command, dept\_name and building is projected

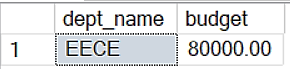


Figure-02: From and Where clause as- select dept\_name, budget from university where dept\_name = 'EECE'

**Experiment Number: 04**

**Experiment Name:** Study and Implementation of DML Commands of

• Group By & Having Clause

• Order By Clause

• Create View, Indexing & Procedure Clause

**Objectives**: To understand and implement advanced Data Manipulation Language (DML) commands and database objects in SQL, including GROUP BY, HAVING, ORDER BY, creating Views, Indexing, and Procedures.

**Theory:** In addition to basic DML commands, there are advanced SQL functionalities and database objects that enable more complex data manipulation, organization, and optimization.

1. **GROUP BY & HAVING Clause:** The GROUP BY clause is used to group rows with identical values in specified columns. It works in conjunction with aggregate functions to produce summary reports on groups of rows. The HAVING clause filters groups based on specified conditions.
2. **ORDER BY Clause:** The ORDER BY clause is used to sort the result set by one or more columns, either in ascending or descending order.
3. **Create View, Indexing, and Procedure Clause:**

* **CREATE VIEW:** Views are virtual tables created as a result of a SELECT query. They allow users to store complex queries and present them as a single table.
* **Indexing:** Indexes are used to speed up the retrieval of rows from a table by creating an index on one or more columns.
* **Procedure Clause:** Procedures are a set of SQL statements that can be stored and executed on demand, allowing for reusable and modular code execution within the database.

Advanced DML commands and database objects in SQL, such as GROUP BY, HAVING, ORDER BY, Views, Indexing, and Procedures, enhance data manipulation, optimization, and code modularity within the database.

**Code:**

|  |
| --- |
| use master  create database varsity  use varsity  create table instructor(  ID varchar(20),  name varchar(20) not null,  dept\_name varchar(20),  salary numeric(8,2),  primary key(ID)  );  insert into instructor values ('10101','Sajeeb','Comp.Sci',65000);  insert into instructor values ('12121','Mujib','Finance',90000);  insert into instructor values ('15151','Rajib','Music',40000);  insert into instructor values ('22222','Naima','Physics',95000);  insert into instructor values ('32343','Sumona','History',60000);  insert into instructor values ('33456','Shama','Physics',87000);  select \* from instructor  select dept\_name from instructor  ---group by  select name from instructor group by name  select dept\_name,avg(salary) as avg\_salary from instructor group by dept\_name  select dept\_name,count(\*) from instructor group by dept\_name  ---having clause  select dept\_name,avg(salary) as avg\_salary from instructor group by dept\_name having avg(salary)>55000;  ----order by clause  select \* from instructor order by salary desc,name asc;  ---view  create view faculty as  select ID,name,dept\_name from instructor  select \* from faculty  ----index  create index dept\_inx on instructor(dept\_name)  ---procedure  create procedure instruct\_proc  AS  BEGIN  select name as authors\_name from instructor where ID = '15151'  END  exec instruct\_proc |

**Output:**

|  |  |
| --- | --- |
| Figure-01: instructor table    Figure-02: “select dept\_name,avg(salary) as avg\_salary from instructor group by dept\_name”    Figure-03: “having avg(salary)>55000” in the previous table | Figure-04: “order by salary desc,name asc” instruction in the instructor table    Figure-05: Creating view as faculty |

**Experiment Number: 05**

**Experiment Name:** Study and Implementation of SQL Commands of Join Operations with Example

• Cartesian Product

• Natural Join

• Left Outer Join

• Right Outer Join

• Full Outer Join

**Objectives:** To understand and implement various SQL join operations, including Cartesian Product, Natural Join, Left Outer Join, Right Outer Join, and Full Outer Join.

**Theory:** In SQL, join operations are used to combine rows from multiple tables based on a related column between them. Various types of joins allow for different methods of merging data from multiple tables.

1. **Cartesian Product:** The Cartesian Product or Cross Join combines every row from one table with every row from another table, resulting in a combination of all possible pairs of rows.
2. **Natural Join:** The Natural Join matches columns with the same name from both tables. It selects the columns with the same name in both tables and creates a result set by combining the rows based on these columns.
3. **Left Outer Join:** The Left Outer Join returns all rows from the left table and the matched rows from the right table. If there are no matching rows from the right table, NULL values are used.
4. **Right Outer Join:** The Right Outer Join returns all rows from the right table and the matched rows from the left table. If there are no matching rows from the left table, NULL values are used.
5. **Full Outer Join:** The Full Outer Join returns all rows when there is a match in either the left or right table. It includes all rows from both tables, with NULL values where there is no match.

SQL join operations enable the merging of data from multiple tables based on relationships, allowing for comprehensive analysis and data retrieval.

**Code:**

|  |
| --- |
| use master  create table depart(  dept\_name varchar(20),  bulding varchar(20),  budget numeric(8,2),  primary key(dept\_name)  );  insert into depart values('ICE','Watson','90000')  insert into depart values('CSE','Science','85000')  insert into depart values('EEE','Engineering','80000')  insert into depart values('CE','Engineering','68000')  insert into depart values('EECE','Science','55000')  insert into depart values('Arct','Painter','95000')  select \* from depart  create table instruct(  ID varchar(20),  name varchar(15) not null,  dept\_name varchar(15),  salary numeric(8,2),  primary key(ID));  insert into instruct(ID,name,dept\_name,salary) values('1012','sumu','ICE','1000')  insert into instruct values('3245','summuu','CSE','1001')  insert into instruct values('3865','raiyan','BANGLA','1002')  insert into instruct values('4755','RIYA','ENGLISH','1003')  insert into instruct values('6789','MAHI','PHYSICS','10004')  select \* from instruct  select \* from depart  ---cartesian product  select ID,name,bulding,salary from instruct,depart where depart.dept\_name = instruct.dept\_name;  ----join product  select ID,name,budget from instruct join depart on depart.dept\_name = instruct.dept\_name;  ---left outer join  select \* from instruct left outer join depart on depart.dept\_name=instruct.dept\_name;  ---right outer join  select \* from instruct right outer join depart on depart.dept\_name=instruct.dept\_name;  ---full outer join  select \* from instruct full outer join depart on depart.dept\_name=instruct.dept\_name; |

**Output:**

|  |  |
| --- | --- |
| Figure-01: instruct table | Figure-02: depart table |
| Figure-03: Cartesian Product | Figure-04: Natural Join |
|  |  |
| Figure-05: instruct left outer join depart | |
| Figure-06: instruct right outer join depart | |
|  | |
| Figure-07: instruct full outer join depart | |

**Experiment Number: 06**

**Experiment Name:** Study and Implementation of Aggregate Function with Example

• Count Function

• Max Function

• Min Function

• Avg Function

**Objectives:** To learn and implement various aggregate functions in SQL, focusing on COUNT, MAX, MIN, and AVG functions.

**Theory:** Aggregate functions in SQL are used to perform calculations on a set of values and return a single value. These functions summarize and process groups of rows to produce a single result.

**COUNT Function:** The COUNT function is used to count the number of rows that match a specified condition.

* **Syntax:** SELECT COUNT(column\_name)

FROM table\_name

WHERE condition;

* **Example**: SELECT COUNT(employee\_id)

FROM Employees

WHERE department\_id = 3;

**MAX Function:** The MAX function is used to find the maximum value of a column.

* Syntax: SELECT MAX(column\_name)

FROM table\_name;

* Example: SELECT MAX(salary)

FROM Employees;

**MIN Function:** The MIN function is used to find the minimum value of a column.

* **Syntax:** SELECT MIN(column\_name)

FROM table\_name;

* **Example:** SELECT MIN(age)

FROM Students;

**AVG Function:** The AVG function is used to calculate the average value of a column.

* **Syntax:** SELECT AVG(column\_name)

FROM table\_name;

* **Example:** SELECT AVG(salary)

FROM Employees;

Aggregate functions in SQL allow for summarizing and analyzing data effectively by providing valuable insights such as counts, maximums, minimums, and averages.

**Code:**

|  |
| --- |
| use master  create database university1  create table instructor(  ID varchar(20),  dept\_name varchar(20),  salary numeric(8,2),  primary key(ID)  );  insert into instructor values('1212','ICE','60000')  insert into instructor values('1215','CE','77000')  insert into instructor values('1219','CSE','85000')  insert into instructor values('1214','EEE','65000')  select \* from instructor  select count(ID) as count\_ID from instructor  select max(salary) as max\_salary from instructor  select min(salary) as min\_salary from instructor  select avg(salary) as avg\_salary from instructor |

**Output:**

|  |
| --- |
| Figure-01: instructor table |
| Figure-02: count operation on instructor |
| Figure-03: max operation on instructor |
| Figure-04: min operation on instructor |
| Figure-05: average operation on instructor |

**Experiment Number: 07**

**Experiment Name:** Study and Implementation of Triggering System on Database Table Using SQL Commands with Example.

**Objectives**: To learn and implement triggers within a SQL database table.

**Theory:** An SQL trigger is a database object that is associated with a table and automatically executes a set of SQL statements when a specific event occurs on that table. Triggers are used to enforce business rules, maintain data integrity, and automate certain actions within a database. They can be triggered by various events, such as inserting, updating, or deleting data in a table, and they allow you to perform additional operations based on those events.There are two types of triggers such as:

* **BEFORE Trigger**: Executes before an operation (e.g., INSERT, UPDATE, DELETE) is performed on a table.
* **AFTER Trigger:** Executes after an operation is performed on a table.

Triggers in SQL provide an automated way to execute specific actions or log events in response to operations performed on a table. They are useful for maintaining data integrity, enforcing business rules, and creating audit trails within the database.

**Code:**

|  |
| --- |
| use master  create database University  use University  -- to see all the table names within the database  SELECT table\_name  FROM information\_schema.tables  WHERE table\_type = 'Base table';  drop table backup\_ins; drop table backup\_del; drop table instructor;  create table instructor (  ID int,  name nvarchar(50),  dept\_name nvarchar(50),  salary int );  insert into instructor values(22222,'Einstein','Physics',95000)  insert into instructor values(12121,'We','Finance',90000)  insert into instructor values(32343,'El Said','History',60000)  insert into instructor values(45565,'Katz','CSE',75000)  select \* from instructor  --create another table for update value keeping  create table backup\_ins (  ID int,  name nvarchar(50),  dept\_name nvarchar(50),  salary int ) ;  --create another table for deleted value keeping  create table backup\_del (  ID int,  name nvarchar(50),  dept\_name nvarchar(50),  salary int );  --creating trrigger  create trigger ins\_trigger  on instructor  after insert  as begin  print'The tigger inserted successfully'  end;  --update trigger  alter trigger ins\_trigger  on instructor  after insert  as begin  insert into backup\_ins select ID, name, dept\_name, salary from inserted  end;  insert into instructor values(45862,'Sajeeb','CSE',75000)  insert into instructor values(58642,'Kumar','EEE',80000)  insert into instructor values(89652,'Ray','ICE',80000)  select \* from instructor  select \* from backup\_ins  --deleted tigger  create trigger del\_trigger  on instructor  after delete  as begin  insert into backup\_del select ID, name, dept\_name, salary from deleted;  end;  DELETE FROM instructor WHERE ID = 32343;  select \* from instructor  select \* from backup\_del |

**Output:**

|  |  |
| --- | --- |
| Figure-01: instructor table    Figure-02: Inserting some records into instructor after initializing trigger. | Figure-03: Reference records from backup\_ins table.    Figure-04: Deleted a record from instructor after initializing trigger.    Figure-05: Backup record from backup\_del table. |

**Experiment Number: 08**

**Experiment Name:** Study and Implementation of SQL Commands to Connect MySQL Database with Java or PHP.

**Objectives:**

1. To understand and use the SQL queries on database.
2. To study the SQL commands to connect MySQL database with JAVA or PHP.

**Theory: Connection Process in Java-**Java applications connect to a MySQL database using

JDBC (Java Database Connectivity). The steps involved in connecting Java with MySQL include:

* Loading the Driver: Load the MySQL JDBC driver class using Class.forName("com.mysql.cj.jdbc.Driver").
* Establishing Connection: Create a connection to the MySQL database using Connection interface and DriverManager.getConnection().
* Executing Queries: Use Statement or PreparedStatement interfaces to execute SQL queries on the established connection.

**Connection Process in PHP-**PHP connects to a MySQL database using MySQLi (MySQL Improved) extension or PDO (PHP Data Objects). The steps involved in connecting PHP with MySQL include:

* Establishing Connection: Use mysqli\_connect() function or PDO to connect to the MySQL database.
* Executing Queries: Use mysqli\_query() or PDO prepared statements to execute SQL queries on the established connection.

**Code:**

|  |
| --- |
| <?php  $server = "localhost";  $user = "root";  $password = "";  $database = "dbms\_lab";  $connect = mysqli\_connect($server, $user, $password, $database);  if($connect){  echo "Database Connected Succefully. <hr>";  }  else {  echo "Database Not Connected. <hr>";  }  function selectDepartment($table)  {  if ($table) {  echo "Department Table";  echo "  <table border='1'>  <tr>  <th> Name </th>  <th> Building </th>  <th> Budget </th>  </tr>  ";  if (mysqli\_num\_rows($table)) {  while ($row = mysqli\_fetch\_array($table)) {  echo "  <tr>  <td> " . $row['dept\_name'] . " </td>  <td> " . $row['bulding'] . " </td>  <td> " . $row['budget'] . " </td>  </tr>  ";  }  }  echo "</table> <hr>";  } else {  echo "Table not created or invalid query <hr>";  }  }  #If the table created before or you already execute the program  mysqli\_query($connect, "drop table department");  $res = mysqli\_query($connect, "create table department(  dept\_name varchar(20),  bulding varchar(20),  budget numeric(8,2),  primary key(dept\_name)  )");  if($res){  echo "Table Created Succefully. <hr>";  }  else {  echo "Table Not Created. <hr>";  }  mysqli\_query($connect, "insert into department values('ICE','Watson','90000')");  mysqli\_query($connect, "insert into department values('CSE','Science','85000')");  mysqli\_query($connect, "insert into department values('EEE','Engineering','80000')");  mysqli\_query($connect, "insert into department values('CE','Engineering','68000')");  mysqli\_query($connect, "insert into department values('EECE','Science','55000')");  mysqli\_query($connect, "insert into department values('Arct','Painter','95000')");  $table = mysqli\_query($connect, "select \* from department");  selectDepartment($table);  mysqli\_query($connect, "delete from department where dept\_name ='CSE' ");  echo "Deleted a record where department name CSE <hr> ";  $table = mysqli\_query($connect, "select \* from department");  selectDepartment($table);  ?> |

**Output:**

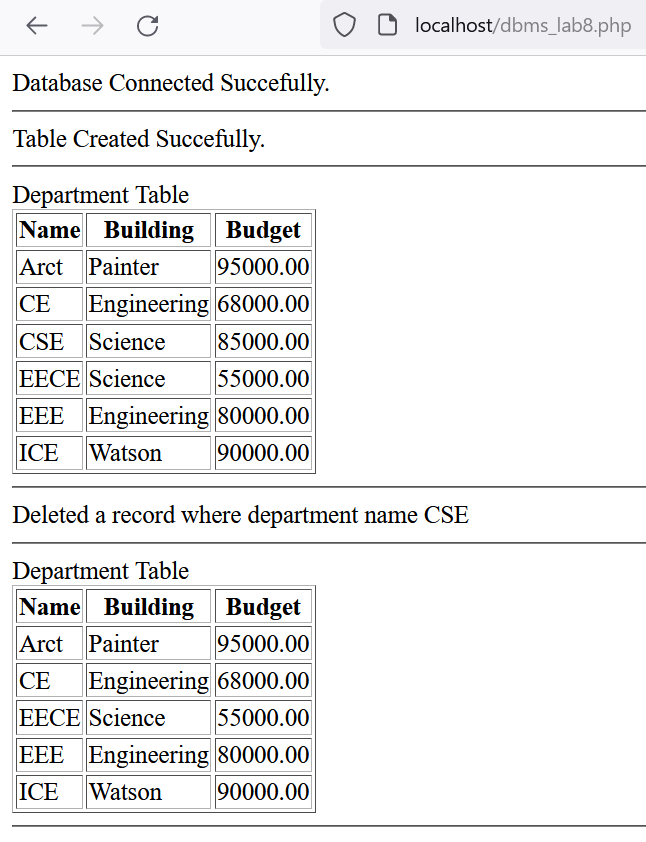
****

Figure-01: Screenshot of a browser, SQL Commands to Connect MySQL Database with PHP.