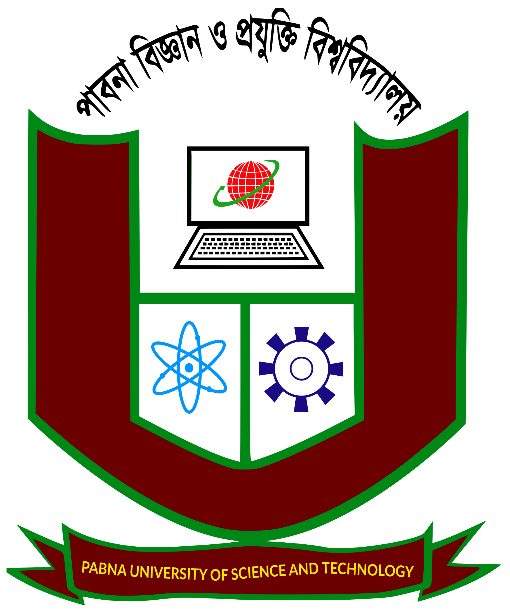
PABNA UNIVERSITY OF SCIENCE AND TECHNOLOGY



Faculty of Engineering and Technology

Department of Information and Communication Engineering

**Lab Report**

Course Title: Database Management Systems Sessional

Course Code: ICE-3106

|  |  |
| --- | --- |
| **Submitted by**  Name: **Gulam Mustofa**  Roll No: **200615**  Session:2019-2020  3rd year 1st semester  Department of Information and Communication Engineering,  Pabna University of Science and Technology | **Submitted to**  Sohag Sarker  Associate Professor  Department of Information and Communication Engineering,  Pabna University of Science and Technology |

………………………

Signature

Submission Date: 05-11-2023

**Index**

|  |  |  |
| --- | --- | --- |
| **SL No** | **Experiment Name** | **Page No** |
| 01 | Study and Implement the following DML Commands of SQL with suitable examples   * Insert * Delete * Update | 02 |
| 02 | Study and Implement the following DDL Commands of SQL with suitable examples   * Create * Alter * Drop | 05 |
| 03 | Study and Implement the following DML Commands  • Select Clause • From Clause • Where Clause | 08 |
| 04 | Study and Implement the following DML Commands  • Group By and Having Clause • Order By Clause • Create View, Indexing and Procedure Clause | 11 |
| 05 | Study and Implement the following SQL Commands of Join Operations with examples • Cartesian Product • Nutural Join • Left Outer Join • Right Outer Join • Full Outer Join | 16 |
| 06 | Study and Implement the following Aggregate Function with example • Count Function • Max Function • Min Function • Avg Function | 22 |
| 07 | Study and Implement the Triggering System on Database Table using SQL commands with sxamples. | 26 |
| 08 | Study and Implement the SQL Commands to connect MySQL Database with Java or PHP. | 29 |

**Experiment No: 01**

**Experiment Name:** Study and Implementation of DML Commands of SQL with Suitable (Insert , Delete , Update)

**Objectives:**

1. To insert elements in a database
2. To delete elements in a database
3. To update element in a database

**Theory:**

Structured collection of data that is organized in a way that allows for efficient storage, retrieval, and manipulation of information. SQL (Structured Query Language) is a programming language used for managing and manipulating relational databases. In this experiment, focusing on Data Manipulation Language (DML) commands of SQL, which are used to interact with the data stored in the database. The three main DML commands, i.e., insertion, deletion, and updating of data in a database

The SQL statement INSERT INTO is used to insert new rows of data into a table in the database. Almost all the RDBMS provide this SQL query to add the records in database tables.

The syntax of INSERT INTO statement is

INSERT INTO TABLE\_NAME (column1, column2...columnN) VALUES (value1, value2...valueN);

The SQL statement insert new column the SQL query is

ALTER TABLE *table\_name* ADD *column\_name datatype*;

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

ALTER TABLE *table\_name* DROP COLUMN *column\_name*;

Delete a record from table

DELETE FROM table\_name WHERE condition;

To rename a column in a table, use the following syntax:

ALTER TABLE *table\_name* RENAME COLUMN *old\_name* to *new\_name*;

To update records in a table using SQL, you can use the UPDATE statement. Here's the basic syntax:

UPDATE table\_name SET column1 = value1, column2 = value2, ... WHERE condition;

Example

UPDATE customers SET email = 'newemail@example.com'

WHERE customer\_id = 5;

**Code:**

create database university

use university

create table department(

dept\_name varchar(20),

building varchar (15),

budget numeric(8,2),

primary key(dept\_name)

);

insert into department values('ICE','Engineering',87000)

insert into department values('CSE','Engineering',90000)

insert into department values('EEE','Science',95000)

insert into department values('EECE','Science',80000)

insert into department values('BANGLA','BANGLA',68000)

insert into department values('ENGLISH','ENGLISH',55000)

insert into department values('IE’,'ENGINEERING',55000)

select \* from department

--deleting

delete from department where dept\_name ='CSE'

select \* from department

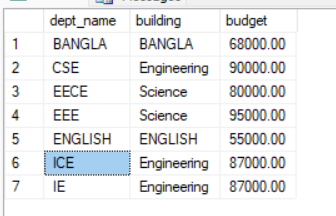
--update

update department set budget = budget + budget\*1.05 where budget <90000

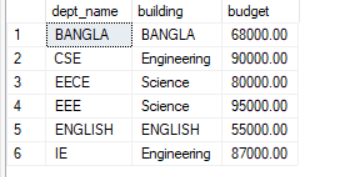
select \* from department

**Output:**

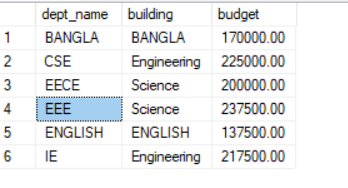
After inserting the values the table is



After deleting value the table is



After updating the table is



**Experiment No: 02**

**Experiment Name:** Study and Implementation of DDL Commands of SQL with Suitable. Example (Create , Alter , Drop)

**Objectives:**

1. To study and implement how to create a table in a database
2. To study and implement how to alter a table in database
3. To study and implement how to drop a record or attribute

**Theory:** In a database the for implementing the DDL commands of SQL with suitable are given below.

(i) CREATE:

The CREATE command in SQL is used to create objects in a database. The primary object that is created using CREATE is a table, but it can also be used to create other objects like indexes, views, and databases (depending on the DBMS).

Examples

For creating a table

CREATE TABLE table\_name (

column1 datatype1 constraints,

column2 datatype2 constraints,

...

);

For creating an index

CREATE INDEX index\_name ON table\_name (column\_name);

1. ALTER:

The ALTER command is used to modify the structure of an existing database object. It can be used to add, modify, or drop columns, constraints, indexes, etc.

Examples:

(a)Adding a Column:

ALTER TABLE table\_name ADD column\_name datatype;

(b) Modifying a Column:

ALTER TABLE table\_name MODIFY column\_name new\_datatype;

This allows you to change the datatype of an existing column.

(c) Dropping a Column:

ALTER TABLE table\_name DROP COLUMN column\_name;

(iii) DROP:

The DROP command is used to remove objects from the database. Be cautious when using this command, as it permanently deletes data.

Examples:

Dropping a Table:

DROP TABLE table\_name;

This deletes an entire table from the database.

Dropping an Index:

DROP INDEX index\_name;

**Code:**

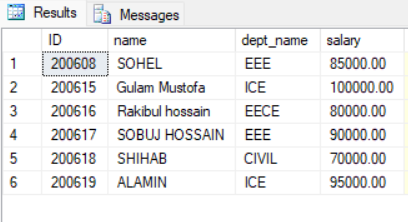
create database university  
use university  
create table instructor(  
ID varchar(20),  
name varchar(15) not null,  
dept\_name varchar(15),  
salary numeric(8,2),  
primary key(ID));  
insert into instructor(ID,name,dept\_name,salary) values('200610','alamin','ICE','86000')  
insert into instructor(ID,name,dept\_name,salary) values('200611','Nirob','CSE','80000')  
insert into instructor(ID,name,dept\_name,salary) values('200601','Naima Islam','EEE','70000')  
insert into instructor(ID,name,dept\_name,salary) values('200622','Sajeeb kumur','EECE','90000')  
insert into instructor(ID,name,dept\_name,salary) values('200605','Uamme kulsum','CE','95000')  
insert into instructor(ID,name,dept\_name,salary) values('200600','Gopal bhar','Arct','68000')  
select \* from instructor  
alter table instructor add course\_id varchar(20)

select \* from instructor  
drop table instructor

select \* from instructor

**Output:**

Create a table

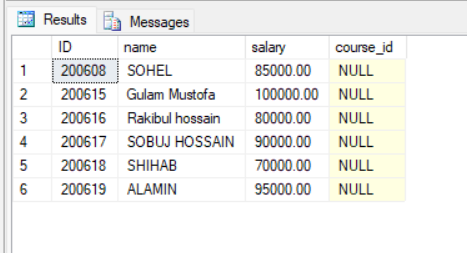
****

Alter a table

A screenshot of a computer screen

Description automatically generated

Drop a column from table



Drop table : No table found

**Experiment No: 03**

**Experiment Name:** Study and Implementation of DML Commands of ( Select Clause , From Clause, Where Clause )

**Objectives:**

1. To study and implement how select clause table in a database
2. To study and implement how from clause table in a database
3. To study and implement how where clause table in a database

**Theory:**

DML (Data Manipulation Language) commands: SELECT, FROM, and WHERE clauses in SQL.

1. SELECT Clause:

The SELECT statement is used to retrieve data from a database. It is one of the most fundamental and frequently used commands in SQL.

SELECT column1, column2, ...FROM table\_name;

Example:

SELECT first\_name, last\_name FROM customers;

This query retrieves the first\_name and last\_name columns from the customers table.

1. FROM Clause:

The FROM clause specifies the source table or tables from which to retrieve data.

Syntax:

SELECT column1, column2, ...FROM table1, table2, ...;

(iii) WHERE Clause:

The WHERE clause is used to filter records based on a specified condition.

Syntax:

SELECT column1, column2, ...FROM table\_name WHERE condition;

Example:

SELECT \*FROM products WHERE category = 'Electronics' AND price > 500;

This query retrieves all columns from the products table where the category is 'Electronics' and the price is greater than 500.

**Code:**

create database university

use university

create table insertvalue(

dept\_name varchar(15),

bulding varchar(15),

budget numeric(8,2)

primary key(dept\_name)

);

insert into insertvalue values('ICE','Engineering',87000)

insert into insertvalue values('CSE','Engineering',90000)

insert into insertvalue values('EEE','JHON',95000)

insert into insertvalue values('EECE','Watson',80000)

insert into insertvalue values('BANGLA','BANGLA',68000)

insert into insertvalue values('ENGLISH','ENGLISH',55000)

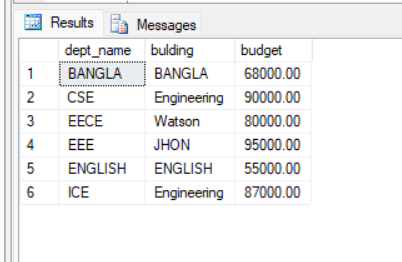
select \* from insertvalue

select dept\_name from insertvalue

select dept\_name from insertvalue where dept\_name = 'EECE'

**Output:**

Select clause

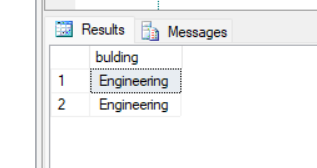


From clause

A screenshot of a computer

Description automatically generated

Where clause



**Experiment No:04**

**Experiment Name:** Study and Implementation of DML Commands of  
• Group By & Having Clause  
• Order By Clause  
• Create View, Indexing & Procedure Clause

**Objectives:**

1. To understand and implement the data definition language for using the Group by & Having Clause
2. To understand and implement the data definition language for using the Order By clause
3. To understand and implement the data definition language for using the Create View, Indexing & Procedure Clause

**Theory:**

DML (Data Manipulation Language) commands in SQL.

Group By & Having Clause:

1. Group By Clause:

The GROUP BY clause is used to group rows with identical data into summary rows. It is often used with aggregate functions like COUNT, SUM, AVG, etc.

Syntax:

SELECT column1, aggregate\_function(column2)

FROM table\_name

GROUP BY column1;

column1: The column by which you want to group the data.

aggregate\_function(column2): An aggregate function applied to column2.

2. Having Clause:

The HAVING clause works like a WHERE clause but is used specifically with aggregate functions. It filters the results after they have been grouped.

Syntax:

SELECT column1, aggregate\_function(column2)

FROM table\_name

GROUP BY column1

HAVING condition;

condition: The condition that must be met for a group to be included in the result set.

3.Order By Clause:

The ORDER BY clause is used to sort the result set based on one or more columns.

SELECT column1, column2, ...

FROM table\_name

ORDER BY column1 [ASC | DESC], column2 [ASC | DESC], ...;

column1, column2, ...: The columns by which you want to sort.

ASC: Ascending order (default).

DESC: Descending order.

Create View, Indexing & Procedure:

(i). Create View:

A view is a virtual table that is based on the result of a SELECT query. It does not store the data itself, but it provides a way to represent complex queries in a simplified form.

Syntax:

CREATE VIEW view\_name AS

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

(ii). Indexing:

Indexes are data structures that improve the speed of data retrieval operations on a table at the cost of additional storage and decreased performance on data modification operations (like INSERT, UPDATE, DELETE).

Syntax to Create an Index:

CREATE INDEX index\_name

ON table\_name (column1, column2, ...);

Syntax to Drop an Index:

DROP INDEX index\_name;

(iii). Procedure:

A stored procedure is a set of SQL statements that can be stored in a database and executed by calling the procedure. It helps in modularizing and reusing code.

Syntax to Create a Procedure:

CREATE PROCEDURE procedure\_name

AS

BEGIN

-- SQL Statements

END;

Syntax to Execute a Procedure:

EXEC procedure\_name;

These DML commands provide advanced capabilities for querying and managing data in a database. Remember to replace column\_name, table\_name, and other placeholders with actual names from your database.

**Code:**

----create alter and drop

create database uinversity

use university

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','Srinivasan','Comp.Sci',65000);

insert into instructor values ('12121','Wu','Finance',90000);

insert into instructor values ('15151','Mozart','Music',40000);

insert into instructor values ('22222','Einstein','Physics',95000);

insert into instructor values ('32343','EI Said','History',60000);

insert into instructor values ('33456','Gold','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

select \* from instructor

---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 asc,name desc;

---view

create view faculty as

select ID,name,dept\_name from instructor

select \* from instructor

----index

create index dept\_inx on instructor(dept\_name)

---procedure

create procedure instruct\_proc

AS

BEGIN

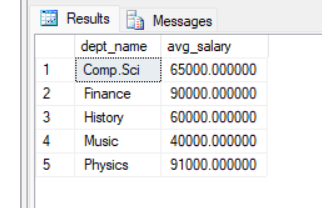
select name as authors\_name from instructor where ID = '15151'

END

exec instruct\_proc

**Output:**

Group By Clause

**A screenshot of a computer

Description automatically generated**

Having by cluse:

A screenshot of a computer

Description automatically generated

Order By (Ascending order)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedCreate view as faculty:

**Experiment No: 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:**

(i)To understand and implement the Cartesian product, Natural join in a database

(ii) To understand and implement the Left Outer Join ,Right Outer Join in a database

(iii) To understand and implement the Full Outer Join in a database

**Theory:**

1. Cartesian Product:

The Cartesian Product combines every row from the first table with every row from the second table. It results in a table with m x n rows (where m is the number of rows in the first table and n is the number of rows in the second table).

Syntax:

SELECT \* FROM table1 CROSS JOIN table2;

Example

Consider two tables A and B:

|  |  |
| --- | --- |
| Table A: | Table B: |
| ID Name  1 Alice  2 Bob | Course Grade  Math A  Science B |

The Cartesian Product (A x B) will be:

ID Name Course Grade

1 Alice Math A

1 Alice Science B

2 Bob Math A

2 Bob Science B

2. Natural Join:

A Natural Join combines two tables based on columns with the same name and data type. It eliminates duplicate columns, keeping only one instance of each column.

Syntax:

SELECT \* FROM table1 NATURAL JOIN table2;

Example:

Consider two tables A and B:

|  |  |
| --- | --- |
| Table A: | Table B |
| ID Name Course  1 Alice Math  2 Bob Science | ID Grade  1 A  2 B |

The Natural Join (A NATURAL JOIN B) will be:

ID Name Course Grade

1 Alice Math A

2 Bob Science B

3. Left Outer Join:

A Left Outer Join returns all the records from the left table (first table) and the matched records from the right table (second table). The result will contain NULL values for the columns from the right table where there is no match.

Syntax:

SELECT \* FROM table1

LEFT JOIN table2 ON table1.column = table2.column;

4. Right Outer Join:

A Right Outer Join is similar to a Left Outer Join, but it returns all the records from the right table and the matched records from the left table. The result will contain NULL values for the columns from the left table where there is no match

Syntax:

SELECT \* FROM table1 RIGHT JOIN table2 ON table1.column = table2.column;

Example:

Consider two tables A and B:

|  |  |
| --- | --- |
| Table A: | Table B: |
| ID Name  1 Alice  2 Bob  3 Charlie | ID Grade  1 A  2 B |

The Left Outer Join (A LEFT JOIN B ON A.ID = B.ID) will be:

ID Name Grade

1 Alice A

2 Bob B

3 Charlie NULL

The Right Outer Join (A RIGHT JOIN B ON A.ID = B.ID) will be:

ID Name Grade

1 Alice A

2 Bob B

NULL NULL C

5. Full Outer Join:

A Full Outer Join returns all records when there is a match in either left or right table. It returns NULL values for unmatched columns on either side.

Syntax:

SELECT \* FROM table1 FULL JOIN table2 ON table1.column = table2.column;

Example

Consider two tables A and B:

|  |  |
| --- | --- |
| Table A: | Table B: |
| ID Name  1 Alice  2 Bob | ID Grade  1 A  3 B |

The Full Outer Join (A FULL JOIN B ON A.ID = B.ID) will be:

ID Name Grade

1 Alice A

2 Bob NULL

NULL NULL B

**Code:**

use university

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')

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(ID,name,dept\_name,salary) values('3245','summuu','CSE','1001')

insert into instruct(ID,name,dept\_name,salary) values('3865','raiyan','BANGLA','1002')

insert into instruct(ID,name,dept\_name,salary) values('4755','RIYA','ENGLISH','1003')

insert into instruct(ID,name,dept\_name,salary) values('6789','MAHI','PHYSICS','10004')

select \* from depart

select \* from instruct

---cartesian product

select 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

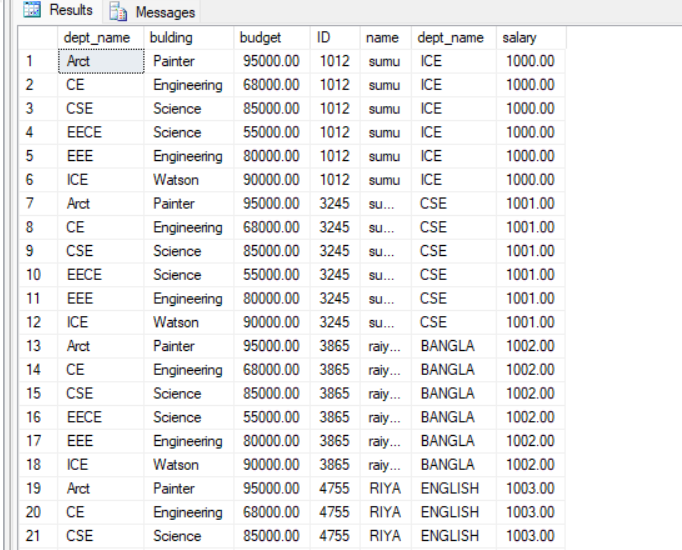
select \* from instruct right outer join depart on depart.dept\_name=instruct.dept\_name;

---full outer join

select \* from instruct full outer join depart on depart.dept\_name=instruct.dept\_name;

Output:

---cartesian product



A screenshot of a computer

Description automatically generated

----join product

A screenshot of a computer

Description automatically generated

---left outer join

A screenshot of a computer

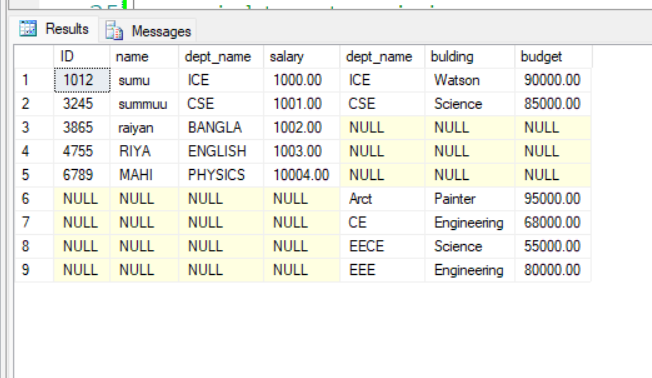
Description automatically generated

---right outer join

A screenshot of a computer

Description automatically generated

---full outer join



**Experiment No:06**

**Experiment Name:** Study and Implementation of Aggregate Function with Example( Count Function, Max Function ,Min Function Avg Function)

**Objectives:**

1. To understand the different issues in the design and implementation of a database system
2. To apply and implement the Aggregate Function

**Theory:** SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value. It is also used to summarize the data. The five type of aggregation function is Count Function, Max Function ,Min Function Avg Function ,Sum

Function.

1. COUNT Function:

The COUNT function is used to count the number of rows that meet a specified condition.

Syntax:

SELECT COUNT(column\_name) FROM table\_name WHERE condition;

2. MAX Function:

The MAX function returns the highest value in a column.

Syntax:

SELECT MAX(column\_name) FROM table\_name WHERE condition;

Example:

Consider a table products:

ID Product\_Name Price

1 Laptop 1200

2 Smartphone 800

3 Tablet 500

SELECT MAX(Price) FROM products;

This will return:

MAX(Price)

1200

3. MIN Function:

The MIN function returns the lowest value in a column.

Syntax:

SELECT MIN(column\_name) FROM table\_name WHERE condition;

Example:

Using the same products table:

SELECT MIN(Price) FROM products;

This will return:

MIN(Price)

500

4. AVG Function:

The AVG function calculates the average value of a column.

Syntax:

SELECT AVG(column\_name) FROM table\_name WHERE condition;

Example:

Using the same products table:

SELECT AVG(Price) FROM products;

This will return:

markdown

AVG(Price)

833.33

These aggregate functions are invaluable when you need to perform calculations on sets of data, such as finding totals, averages, maximum and minimum values, etc. They allow you to summarize and analyze your data effectively

**Code:**

create database university

use university

create table instructorSalary(

ID varchar(20),

dept\_name varchar(20),

salary numeric(8,2),

primary key(ID)

);

insert into instructorSalary values('1212','ICE','60000')

insert into instructorSalary values('1215','CE','77000')

insert into instructorSalary values('1219','CSE','85000')

insert into instructorSalary values('1214','EEE','65000')

select \* from instructorSalary

select count(ID) as count\_ID from instructorSalary

select max(salary) as max\_salary from instructorSalary

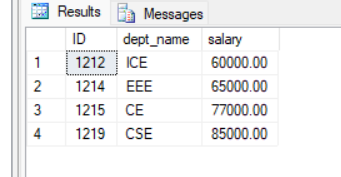
select min(salary) as min\_salary from instructorSalary

select avg(salary) as avg\_salary from instructorSalary

select SUM(salary) as total\_salary from instructorSalary

**Output:**

--Table

****

**--count**

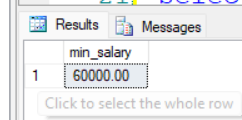
**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated--max**

**--min**

****

**--Avg**

**A screenshot of a computer screen

Description automatically generated**

**--sum**

**A screenshot of a computer

Description automatically generated**

**Experiment No: 07**

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

**Objectives:**

1. To understand and implement the triggering system on database table using sql
2. To applying triggering on database.
3. To understand how to access the data within the trigger

**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

A triggering system in a database allows you to define actions that should be automatically executed when certain events occur on a table, such as inserting, updating, or deleting records. These actions are defined using triggers, which are blocks of SQL code associated with a specific event on a table.

Syntax:

*create trigger [trigger\_name]*

*[before | after]*

*{insert | update | delete}*

*on [table\_name]*

*[for each row]*

*[trigger\_body]*

Insert Trigger: When data is inserted into the original table, a trigger can automatically add the same data to a backup table. This ensures that a copy of the data is kept for future reference or recovery.

Delete Trigger: When data is deleted from the original table, a trigger can add the deleted data to a backup table. This is valuable for maintaining an audit trail or keeping a history of changes.

**Code:**

use University

create table instructor

( ID int, name nvarchar(50), dept\_name nvarchar(50),salary int )

select \* from instructor

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)

insert into instructor values(98345,'Kim','EEE',80000)

insert into instructor values(98346,'AL AMIN','ICE',80000)

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 )

select \* from backup\_ins

--create another table for deleted value keeping

create table backup\_del

( ID int, name nvarchar(50), dept\_name nvarchar(50),salary int )

select \* from backup\_del

--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(ID, name, dept\_name, salary)

SELECT ID, name, dept\_name, salary

FROM inserted;

END;

--deleted tigger

create TRIGGER del\_trigger

ON instructor

AFTER DELETE

AS

BEGIN

INSERT INTO backup\_del (ID, name, dept\_name, salary)

SELECT ID, name, dept\_name, salary

FROM deleted;

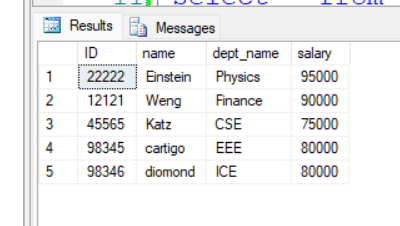
END;

DELETE FROM instructor WHERE ID = 32343;

select \* from backup\_del

Output:

The orginal table instructor is



After inserting one element the inserted table

A screenshot of a computer

Description automatically generated

After deleted tuple from instructor table the backup table is   
A screenshot of a computer

Description automatically generated

**Experiment No:08**

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

**Objectives:**

1.To study and implement the php and html form for inserting information to the database in local server xampp

2.To create a database in xampp Mysql and connect with php

**Theory:** PHP is a server-side scripting language commonly used for web development. It is particularly well-suited for database interactions. MySQL is a popular open-source relational database management system. Connecting PHP with MySQL allows web applications to dynamically interact with and manipulate data stored in a MySQL database

The objective is to learn and apply SQL commands for connecting a MySQL database with PHP. This involves creating an HTML form to input data, establishing a connection to a local XAMPP server with MySQL, and executing PHP scripts to insert information into the database. Additionally, the goal is to create a database within XAMPP's MySQL environment and establish a connection using PHP, facilitating the seamless interaction between web-based forms and the underlying database. This exercise aims to provide hands-on experience in setting up a functional database-driven web application locally.

**Code:**

<?php

$base = mysqli\_connect('localhost', 'root', '', 'insert');

if(isset($\_POST['submit'])){

$name = $\_POST['name'];

$email = $\_POST['email'];

$password = $\_POST['password'];

$sql = "INSERT INTO insertform(name, email, password) VALUES ('$name', '$email', '$password')";

if(mysqli\_query($base, $sql)){

echo "Inserted successfully";

}

else{

echo "Insertion failed: " . mysqli\_error($base); // Added error message for debugging

}

}

mysqli\_close($base); // Close the connection after use

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>indert form</title>

<style>

body{

background-color: antiquewhite;

font-family: Arial, Helvetica, sans-serif;

}

h1{

text-align: center;

}

label {

font-weight: bold;

margin-bottom: 5px;

}

input {

width: 100%;

padding: 8px;

margin-bottom: 10px;

border-radius: 8px;

border-color: green;

}

input[type="submit"] {

background-color: blueviolet;

color: white;

cursor: pointer;

padding: 5px 5px;

margin: 0 auto;

display: block;

}

</style>

</head>

<body>

<h1>Personal Details</h1>

<form action="insert.php" method="POST">

<label for="name">First Name : </label>

<input type="text" id="name" name="name" placeholder="Enter your name"><br>

<label for="email">Email : </label>

<input type="email" id="email" name="email" placeholder="Enter valid email "><br>

<label for="passward">Passward : </label>

<input type="password" id="password" name="password" placeholder="Enter 6 digit password"><br>

<input type="submit" name="submit" value="submit">

</form>

</body>

</html>

**Output:**

**php form**

A screenshot of a personal details form

Description automatically generated

Mysql database

A screenshot of a computer

Description automatically generated