Experiment Number: 01

Name of the Experiment: To study and implementation of DML Commands of SQL with Suitable Example:

- Insert.
- Update.
- Delete.

Objectives:

1. To understand and use data manipulation language to write query for database.
2. To implement *Insertion*, *Deletion* and *Update* operations on a database.

Theory: DML, or Data Manipulation Language, is a subset of SQL (Structured Query Language) used in relational database management systems. DML focuses on the manipulation of data stored in a database. It includes commands and statements for performing operations such as inserting new records into a table (using the INSERT statement), modifying existing records (using the UPDATE statement), and deleting records (using the DELETE statement). DML also provides querying capabilities through the SELECT statement, which allows users to retrieve and filter data from a database. In essence, DML serves as the interface for users and applications to interact with and control the data within a database.

Insert: The insert operation in a database involves adding new records (rows) into a table. It is used to populate a table with fresh data, and the newly inserted data becomes part of the database.

```
insert into <table name>
values('value1','value2','value3')
```

Update: The update operation in a database is used to modify existing records within a table. It allows us to change the values of one or more columns in specific rows, typically based on certain conditions.

```
update <table name> set <value3>=#####
where <primary key>='#####'
```

Delete: The delete operation in a database is used to remove records (rows) from a table. It helps in eliminating unwanted or outdated data, and the table is updated without the deleted records.

```
delete from <table name>
where <primary key>='#####'
```

Experimental Requirements:

1. Laptop/ Desktop Computer.
2. Microsoft SQL Server application.
3. Microsoft Server Management Studio application.

Source Code:

```
use master
create database Sayeeda_Khan

use Sayeeda_Khan

create table classroom(
      building varchar(20),
      room_number varchar(10),
      capacity numeric(5,0),
      primary key(building,room_number))

select * from classroom

insert into classroom
                  values('Engineering','301','25'),
                           ('Engineering','302','50'),
                           ('Engineering','303','45'),
                           ('Engineering','304','25')

select * from classroom

update classroom
set capacity= 40
where room_number ='304'

select * from classroom

delete from classroom
where room_number=301

select * from classroom
```

Output:

Table (1): Classroom table created in the *Sayeeda Khan* database.

	building	room_number	capacity

Table (2): Data Inserted successfully in the classroom table.

	building	room_number	capacity
1	Engineering	301	25
2	Engineering	302	50
3	Engineering	303	45
4	Engineering	304	25

Table (3): Data updated successfully in the classroom table.

	building	room_number	capacity
1	Engineering	301	25
2	Engineering	302	50
3	Engineering	303	45
4	Engineering	304	40

Table (4): Data deleted successfully from the classroom table.

	building	room_number	capacity
1	Engineering	302	50
2	Engineering	303	45
3	Engineering	304	40

Result and Discussion: Using DML command we have created a database and then have inserted values on the database then we have inserted the rows using the command then has updated some values in the database and finally we have deleted some values from our database and in the output, we can see those result. Because we have fulfilled our objectives so we can say that the experiment went successfully.

Experiment Number: 02

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

- Create.
- Alter.
- Delete.

Objectives:

1. To understand and use data definition language to write query for database.
2. To implement *create*, *alter* and *drop* operations on a database.

Theory: DDL (Data Definition Language) commands are SQL statements used to manage the structure of a database. They include commands like CREATE (to define tables, indexes, and databases), ALTER (for modifying existing structures), and DROP (for deleting tables or databases). DDL commands are essential for defining and maintaining the database schema, specifying data types, constraints, and relationships. Certainly, here are concise definitions for the SQL commands CREATE, ALTER, and DELETE:

CREATE: In SQL, the CREATE command is used to define new database objects, such as tables, indexes, or databases themselves. It specifies the structure and attributes of the object, allowing data to be stored and organized.

ALTER: The ALTER command is used to modify the structure of existing database objects, such as tables or columns. It can add, delete, or modify columns, constraints, or other properties without recreating the entire object.

DELETE: The DELETE command is employed to remove data rows from a table, effectively erasing specific records while keeping the table's structure intact. It is often used to eliminate unwanted or outdated data from a database.

Experimental Requirements:

1. Laptop/ Desktop Computer.
2. Microsoft SQL Server application.
3. Microsoft Server Management Studio application.

Source Code:

```
use master
create database Sayeeda_Khan

use Sayeeda_Khan

create table student(
      ID numeric(8,0),
      stu_name varchar(20),
      dept_name varchar(20),
      total_credit numeric(7,2),
      sesion varchar(20),
      reg_num numeric(8,0),
      primary key(ID))

insert into student
values('200101','Panda','CSE','160','2019-2020','1065319'),
            ('200210','Penguin','EEE','155','2017-2018','1056779'),
            ('200618','Master Shifu','ICE','165','2019-2020','1065383')

select * from student

alter table student drop column reg_num

select * from student

select name from sysobjects where xtype='u'

drop table classroom

select name from sysobjects where xtype='u'
```

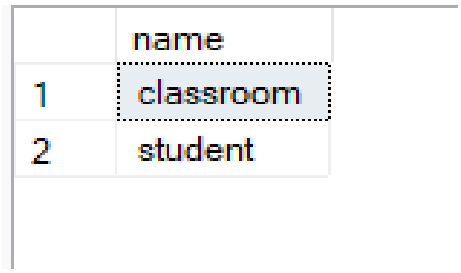
Output:

Table (1): Student table data before altering in *Sayeeda_Khan* database.

	ID	stu_name	dept_name	total_credit	sesion	reg_num
1	123456	Groot	Physics	155.00	2017-2018	1056779
2	200101	Panda	CSE	160.00	2019-2020	1065319
3	200210	Penguin	EEE	155.00	2017-2018	1056779
4	200604	Sayeeda Khan	ICE	165.00	2019-2020	1065369
5	200614	Body Jawad	ICE	165.00	2019-2020	1065379
6	200618	Master Shifu	ICE	165.00	2019-2020	1065383
7	201234	Iron Man	EECE	155.00	2017-2018	1056779
8	203456	Bat Man	Math	155.00	2017-2018	1056779
9	205678	Spider Man	Bangla	155.00	2017-2018	1056779
10	209012	Thanos	English	155.00	2017-2018	1056779

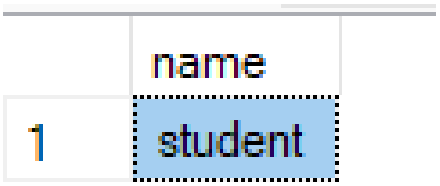
Table (2): Student table data after altering in *Sayeeda_Khan* database ('reg_num' dropped from the table)

	ID	stu_name	dept_name	total_credit	sesion
1	123456	Groot	Physics	155.00	2017-2018
2	200101	Panda	CSE	160.00	2019-2020
3	200210	Penguin	EEE	155.00	2017-2018
4	200604	Sayeeda Khan	ICE	165.00	2019-2020
5	200614	Body Jawad	ICE	165.00	2019-2020
6	200618	Master Shifu	ICE	165.00	2019-2020
7	201234	Iron Man	EECE	155.00	2017-2018
8	203456	Bat Man	Math	155.00	2017-2018
9	205678	Spider Man	Bangla	155.00	2017-2018
10	209012	Thanos	English	155.00	2017-2018



	name
1	classroom
2	student

Figure (1): Before performing drop table



	name
1	student

Figure (2): After performing drop table.

Result and Discussion: Using DDL command first we have created a database and inside the database we have created a table. Then inserted some data in the table and we can see that we can easily insert those data in the table that means the database and table created successfully. Then we altered the table, we deleted one attribute from the table and that is the alter operation, after altering the table we can see and compare the output of before and after. Finally, we dropped a table from the database and as a result we can see before there were two databases in the table and after dropping one database only one database remains. If we drop the database, then the database will be deleted we can perform that as well. So, we can say from the output that we have completed the experiment successfully.

Experiment Number: 03

Name of the Experiment: To Study and Implementation of DML Commands of

- Select Clause.
- From Clause.
- Where Clause.

Objectives:

1. To understand and use data manipulation language to write query for database.
2. To implement *select, from* and *where* clause on a database.

Theory: DML, or Data Manipulation Language, is a subset of SQL (Structured Query Language) used for managing and manipulating data within a relational database. It includes commands like INSERT (to add new data), UPDATE (to modify existing data), and DELETE (to remove data). DML also involves the SELECT statement for querying and retrieving data. These commands enable users to interact with and control the data stored in a database, making DML an essential part of database management. Certainly, let's delve into the SELECT statement, the FROM clause, and the WHERE clause in SQL:

SELECT statement: The SELECT statement is a fundamental component of SQL's Data Manipulation Language (DML). It is primarily used to retrieve data from one or more tables within a database. By specifying the desired columns or expressions, we can shape the data we want to extract.

FROM Clause: The FROM clause is an essential part of the SELECT statement. It indicates the source tables from which the data is to be retrieved. We specify the table name after the FROM keyword, and if necessary, we can join multiple tables together to create more comprehensive result sets.

WHERE Clause: The WHERE clause is often paired with the SELECT statement to filter the results. This clause allows us to define conditions that the retrieved data must meet, such as specific values, ranges, or logical expressions.

Experimental Requirements:

1. Laptop/ Desktop Computer.
2. Microsoft SQL Server application.
3. Microsoft Server Management Studio application.
4. Sayeeda_Khan Database.

Source Code:

```
use Sayeeda_Khan
select * from student
select stu_name,dept_name,session from student
select ID,stu_name,sesion from student
where dept_name='ICE'
```

Output:

Table (1): Table showing all data from Student table in *Sayeeda_Khan* database.

	ID	stu_name	dept_name	total_credit	sesion
1	123456	Groot	Physics	155.00	2017-2018
2	200101	Panda	CSE	160.00	2019-2020
3	200210	Penguin	EEE	155.00	2017-2018
4	200604	Sayeeda Khan	ICE	165.00	2019-2020
5	200614	Body Jawad	ICE	165.00	2019-2020
6	200618	Master Shifu	ICE	165.00	2019-2020
7	201234	Iron Man	EECE	155.00	2017-2018
8	203456	Bat Man	Math	155.00	2017-2018
9	205678	Spider Man	Bangla	155.00	2017-2018
10	209012	Thanos	English	155.00	2017-2018

Table (2): Table showing data using select clause from Student table.

	stu_name	dept_name	sesion
1	Groot	Physics	2017-2018
2	Panda	CSE	2019-2020
3	Penguin	EEE	2017-2018
4	Sayeeda Khan	ICE	2019-2020
5	Body Jawad	ICE	2019-2020
6	Master Shifu	ICE	2019-2020
7	Iron Man	EECE	2017-2018
8	Bat Man	Math	2017-2018
9	Spider Man	Bangla	2017-2018
10	Thanos	English	2017-2018

Table (3): Table showing data using select clause then where clause from Student table.

	ID	stu_name	sesion
1	200604	Sayeeda Khan	2019-2020
2	200614	Body Jawad	2019-2020
3	200618	Master Shifu	2019-2020

Resul and Discussion: Using select clause we first have shown all the tuples of the student table from Sayeeda_Khan database, then we have shown only student name, department name and session from student table using the from clause then finally we have shown the student's name their roll and session from department of ICE using the 'where clause' . From the output we can say that our SQL command is accurate, that's why we have a perfect output. So, we can say that the experiment went successfully.

Experiment Number: 04

Name of the Experiment: To Study and Implementation of DML Commands of

- Group by & Having Clause.
- Order by Clause.
- Create View, Indexing & Procedure Clause.

Objectives:

1. To understand and use data manipulation language to write query for database.
2. To implement group by & having, order by and create view, indexing & procedure clause on a database.

Theory: DML, or Data Manipulation Language, is a subset of SQL (Structured Query Language) used for managing and manipulating data within a relational database. It includes commands like INSERT (to add new data), UPDATE (to modify existing data), and DELETE (to remove data). DML also involves the SELECT statement for querying and retrieving data. These commands enable users to interact with and control the data stored in a database, making DML an essential part of database management. Certainly, here are definitions for each of the SQL clauses and operations you've mentioned:

Group By & Having Clause:

GROUP BY Clause: The GROUP BY clause is used to group rows from a table based on the values in one or more columns. It divides the result set into groups, where each group has the same values in the specified column(s).

HAVING Clause: The HAVING clause is used in combination with the GROUP BY clause to filter the grouped results. It specifies conditions that the groups must meet, and only the groups that satisfy these conditions are included in the result.

Order By Clause:

ORDER BY clause is used to sort the result set of a query in ascending (ASC) or descending (DESC) order based on one or more columns. It helps to arrange the output data in a specific sequence, making it easier to analyze or present.

Create View, Indexing & Procedure Clause:

Create View: A view in SQL is a virtual table created by a query. It allows us to simplify complex queries by storing them as a view and then querying the view like a regular table. It doesn't store data but provides a dynamic way to retrieve specific subsets of data.

Indexing: Indexes are database structures used to optimize data retrieval. They work like the index of a book, making data access faster by pre-sorting and storing key column values.

Procedure Clause: SQL procedures (often referred to as stored procedures) are precompiled SQL statements and commands that are saved in the database. They can be executed on demand, allowing you to encapsulate and reuse a series of SQL statements as a single unit, enhancing code organization and security.

Experimental Requirements:

1. Laptop/ Desktop Computer.
2. Microsoft SQL Server application.
3. Microsoft Server Management Studio application.

Source Code:

```
use Sayeeda_Khan
select * from instructor

select count(ID) as Teachers,dept_name
from instructor
group by dept_name

select count(ID) as Teachers,dept_name,avg(salary) as average_salary
from instructor
group by dept_name
having avg(salary)>72000


select ID,Ins_name,dept_name,salary
from instructor
where dept_name='ICE'
order by salary ASC

create view Teacher as
select Ins_name,dept_name
from instructor
where salary>80000

select * from Teacher

create index teacher on instructor(Ins_name)
/* speed up data retrieval for queries that involve the "Ins_name"
column*/

CREATE PROCEDURE get_teacher_by_dept
    @tech_dept VARCHAR(30)
AS
BEGIN
```

```
    SELECT id, Ins_name
    FROM instructor
    WHERE dept_name = @tech_dept;
END

EXEC get_teacher_by_dept @tech_dept = 'ICE'
```

OUTPUT:

Table (1): Instructor table after applying the 'group by' clause.

	Teachers	dept_name
1	2	CSE
2	2	EECE
3	3	EEE
4	12	ICE
5	3	MATH
6	1	STAT

Table (2): Instructor table after applying the 'group by and having' clause.

	Teachers	dept_name	average_salary
1	2	CSE	72500.000000
2	2	EECE	75000.000000
3	3	EEE	73333.333333
4	12	ICE	96250.000000
5	3	MATH	80000.000000

Table (3): Instructor table after applying the 'order by' clause on salary Ascending.

	ID	Ins_name	dept_name	salary
1	ICE12	Muntasir Ahmed	ICE	85000
2	ICE08	Taskin Noor Turna	ICE	90000
3	ICE09	Tarun Debnath	ICE	90000
4	ICE10	Md. Tofail Ahmed	ICE	90000
5	ICE11	A F Mohammad Zainul Abadin	ICE	100000
6	ICE01	Dr. Pallab Kanti Poddar	ICE	100000
7	ICE02	Dr. Md. Omar Faruk	ICE	100000
8	ICE03	Md. Anwar Hossain	ICE	100000
9	ICE04	Iffat Ara	ICE	100000
10	ICE05	Sohag Sarker	ICE	100000
11	ICE06	Md. Sarwar Hosain	ICE	100000
12	ICE07	Dr. Md. Imran Hossain	ICE	100000

Table (4): Instructor table after applying the 'Create View' clause.

	Ins_name	dept_name
1	Safiul Islam	EECE
2	Dr. Pallab Kanti Poddar	ICE
3	Dr. Md. Omar Faruk	ICE
4	Md. Anwar Hossain	ICE
5	Iffat Ara	ICE
6	Sohag Sarker	ICE
7	Md. Sarwar Hosain	ICE
8	Dr. Md. Imran Hossain	ICE
9	Taskin Noor Turna	ICE
10	Tarun Debnath	ICE
11	Md. Tofail Ahmed	ICE
12	A F Mohammad Zainul Abadin	ICE
13	Muntasir Ahmed	ICE
14	Dr. Md. Azizur Rahman	MATH

Table (5): Instructor table after applying the 'Procedure' clause.

	id	Ins_name
1	ICE01	Dr. Pallab Kanti Poddar
2	ICE02	Dr. Md. Omar Faruk
3	ICE03	Md. Anwar Hossain
4	ICE04	Iffat Ara
5	ICE05	Sohag Sarker
6	ICE06	Md. Sarwar Hosain
7	ICE07	Dr. Md. Imran Hossain
8	ICE08	Taskin Noor Turna
9	ICE09	Tarun Debnath
10	ICE10	Md. Tofail Ahmed
11	ICE11	A F Mohammad Zainul Abadin
12	ICE12	Muntasir Ahmed

Resul and Discussion: We can see that while using group by clause we must use the aggregate functions otherwise it wouldn't work. Having clause can work alone but with group by having clause must use the aggregate functions as well. In the order by clause, we can set them in ascending or descending order. While using create view is like creating a new table from an old one with selected attributes. The index tag doesn't show any output, it just increases the data retrieving speed for the selected attributes. Procedure clause is like a function where if we pass different values, it will produce different output as we created the cause.

Experiment Number: 05

Name of the Experiment: To 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:

1. To understand the concept of the Join Operations.
2. To implement Cartesian Product, Natural Join, Left Outer Join, Right Outer Join and Full Outer Join on a database.

Theory: A Cartesian product is like combining every item from one set with every item from another set to create a new set that includes all possible pairings. For example, if you have a set of 3 colors and a set of 2 sizes, the Cartesian product will give you all possible color-size combinations, resulting in 6 pairs. It's used in mathematics and databases, but it can generate a lot of data, so it's typically avoided in database queries to keep results manageable. Join operations in SQL are used to combine rows from two or more tables based on a related column between them. The primary purpose of joins is to retrieve and display data from multiple tables in a way that establishes relationships and connections between the tables' records. Common types of join operations include:

INNER JOIN: Returns only the rows where there is a match in both tables, based on the specified join condition. It combines data from two tables where the values in the specified columns are equal.

LEFT JOIN (or LEFT OUTER JOIN): Returns all rows from the left table and the matching rows from the right table. If there is no match in the right table, NULL values are returned.

RIGHT JOIN (or RIGHT OUTER JOIN): The opposite of a LEFT JOIN. It returns all rows from the right table and the matching rows from the left table. Rows from the left table with no match in the right table result in NULL values.

FULL JOIN (or FULL OUTER JOIN): Returns all rows when there is a match in either the left or right table. It combines data from both tables, including unmatched rows with NULL values for missing matches.

Experimental Requirements:

1. Laptop/ Desktop Computer.
2. Microsoft SQL Server application.
3. Microsoft SQL Server Management Studio application.
4.

Source Code:

```
use Sayeeda_Khan
 select * from instructor
 select * from student

 exec sp_rename 'instructor.dept_name', 'teach_dept', 'column'
 exec sp_rename 'instructor.ID','Teacher_ID','column'
 exec sp_rename 'student.Studentr_ID','Student_ID','column'
 exec sp_rename 'student.dept_name','Student_deptNAME','column'

 delete from student
 where Student_ID='200614'

 select  Teacher_ID,Ins_name,teach_dept,
 Student_ID,stu_name,stude_deptNAME,total_credit
 from instructor
 cross join student



 select instructor.Ins_name as Teacher_name,
      instructor.teach_dept as Teacher_Dept,
      student.Student_ID as Student_ID,
      student.stu_name as Student_Name,
      student.stude_deptNAME as Student_Dept
 from instructor
 inner join student
 on instructor.teach_dept=student.stude_deptNAME


 select instructor.Ins_name as Teacher_name,
      instructor.teach_dept as Teacher_Dept,
      student.Student_ID as Student_ID,
      student.stu_name as Student_Name,
      student.stude_deptNAME as Student_Dept
 from instructor
 left join student
 on instructor.teach_dept=student.stude_deptNAME

 select instructor.Ins_name as Teacher_name,
      instructor.teach_dept as Teacher_Dept,
      student.Student_ID as Student_ID,
      student.stu_name as Student_Name,
      student.stude_deptNAME as Student_Dept
 from instructor
```

```
 right join student
 on instructor.teach_dept=student.stude_deptNAME

select instructor.Ins_name as Teacher_name,
      instructor.teach_dept as Teacher_Dept,
      student.Student_ID as Student_ID,
      student.stu_name as Student_Name,
      student.stude_deptNAME as Student_Dept
 from instructor
 full outer join student
 on instructor.teach_dept=student.stude_deptNAME
```

OUTPUT:

Table (1): Table after applying the Cartesian Product operation.

77	ICE01	Dr. Pallab Kanti Poddar	ICE	200604	Sayeeda Khan	ICE	165.00
78	ICE02	Dr. Md. Omar Faruk	ICE	200604	Sayeeda Khan	ICE	165.00
79	ICE03	Md. Anwar Hossain	ICE	200604	Sayeeda Khan	ICE	165.00
80	ICE04	Iffat Ara	ICE	200604	Sayeeda Khan	ICE	165.00
81	ICE05	Sohag Sarker	ICE	200604	Sayeeda Khan	ICE	165.00
82	ICE06	Md. Sarwar Hosain	ICE	200604	Sayeeda Khan	ICE	165.00
83	ICE07	Dr. Md. Imran Hossain	ICE	200604	Sayeeda Khan	ICE	165.00
84	ICE08	Taskin Noor Turna	ICE	200604	Sayeeda Khan	ICE	165.00
85	ICE09	Tarun Debnath	ICE	200604	Sayeeda Khan	ICE	165.00
86	ICE10	Md. Tofail Ahmed	ICE	200604	Sayeeda Khan	ICE	165.00
87	ICE11	A F Mohammad Zainul Abadin	ICE	200604	Sayeeda Khan	ICE	165.00
88	ICE12	Muntasir Ahmed	ICE	200604	Sayeeda Khan	ICE	165.00
89	Math-01	Rajendra Chandra Bhowmik	MATH	200604	Sayeeda Khan	ICE	165.00
90	Math-02	Uday Sankar Basak	MATH	200604	Sayeeda Khan	ICE	165.00
91	Math-03	Dr. Md. Azizur Rahman	MATH	200604	Sayeeda Khan	ICE	165.00
92	Stat01	Mira Khatun	STAT	200604	Sayeeda Khan	ICE	165.00

Table (2): Table after applying the Inner Join operation.

	stu_name	Student_ID	stude_deptNAME	total_credit	Ins_name
6	Sayeeda Khan	200604	ICE	165.00	Dr. Pallab Kanti Poddar
7	Sayeeda Khan	200604	ICE	165.00	Dr. Md. Omar Faruk
8	Sayeeda Khan	200604	ICE	165.00	Md. Anwar Hossain
9	Sayeeda Khan	200604	ICE	165.00	Iffat Ara
10	Sayeeda Khan	200604	ICE	165.00	Sohag Sarker
11	Sayeeda Khan	200604	ICE	165.00	Md. Sarwar Hosain
12	Sayeeda Khan	200604	ICE	165.00	Dr. Md. Imran Hossain
13	Sayeeda Khan	200604	ICE	165.00	Taskin Noor Turna
14	Sayeeda Khan	200604	ICE	165.00	Tarun Debnath
15	Sayeeda Khan	200604	ICE	165.00	Md. Tofail Ahmed
16	Sayeeda Khan	200604	ICE	165.00	A F Mohammad Zainul Abadin
17	Sayeeda Khan	200604	ICE	165.00	Muntasir Ahmed

Table (3): Table after applying the left outer join operation.

	Teacher_name	Teacher_Dept	Student_ID	Student_Name	Student_Dept
1	NULL	NULL	123456	Groot	Physics
2	Nitun Kumar Podder	CSE	200101	Panda	CSE
3	Abur Rahim	CSE	200101	Panda	CSE
4	Dipongkor Kundu	EEE	200210	Penguin	EEE
5	Md. Shamim Hossain	EEE	200210	Penguin	EEE
6	Tonmoy Ghosh	EEE	200210	Penguin	EEE
7	Dr. Pallab Kanti Poddar	ICE	200604	Sayeeda Khan	ICE
8	Dr. Md. Omar Faruk	ICE	200604	Sayeeda Khan	ICE
9	Md. Anwar Hossain	ICE	200604	Sayeeda Khan	ICE
10	Iffat Ara	ICE	200604	Sayeeda Khan	ICE
11	Sohag Sarker	ICE	200604	Sayeeda Khan	ICE
12	Md. Sarwar Hosain	ICE	200604	Sayeeda Khan	ICE
13	Dr. Md. Imran Hossain	ICE	200604	Sayeeda Khan	ICE
14	Taskin Noor Turna	ICE	200604	Sayeeda Khan	ICE
15	Tarun Debnath	ICE	200604	Sayeeda Khan	ICE

Table (4): Table after applying the right outer join operation.

22	Taskin Noor Turna	ICE	200604	Sayeeda Khan	ICE
23	Taskin Noor Turna	ICE	200618	Master Shifu	ICE
24	Tarun Debnath	ICE	200604	Sayeeda Khan	ICE
25	Tarun Debnath	ICE	200618	Master Shifu	ICE
26	Md. Tofail Ahmed	ICE	200604	Sayeeda Khan	ICE
27	Md. Tofail Ahmed	ICE	200618	Master Shifu	ICE
28	A F Mohammad Zain...	ICE	200604	Sayeeda Khan	ICE
29	A F Mohammad Zain...	ICE	200618	Master Shifu	ICE
30	Muntasir Ahmed	ICE	200604	Sayeeda Khan	ICE
31	Muntasir Ahmed	ICE	200618	Master Shifu	ICE
32	Rajendra Chandra Bh...	MATH	203456	Bat Man	Math
33	Uday Sankar Basak	MATH	203456	Bat Man	Math
34	Dr. Md. Azizur Rahman	MATH	203456	Bat Man	Math
35	Mira Khatun	STAT	NULL	NULL	NULL

Table (5): Table after applying the full outer join operation.

	Teacher_name	Teacher_Dept	Student_ID	Student_Name	Student_Dept
23	Taskin Noor Turna	ICE	200618	Master Shifu	ICE
24	Tarun Debnath	ICE	200604	Sayeeda Khan	ICE
25	Tarun Debnath	ICE	200618	Master Shifu	ICE
26	Md. Tofail Ahmed	ICE	200604	Sayeeda Khan	ICE
27	Md. Tofail Ahmed	ICE	200618	Master Shifu	ICE
28	A F Mohammad Zain...	ICE	200604	Sayeeda Khan	ICE
29	A F Mohammad Zain...	ICE	200618	Master Shifu	ICE
30	Muntasir Ahmed	ICE	200604	Sayeeda Khan	ICE
31	Muntasir Ahmed	ICE	200618	Master Shifu	ICE
32	Rajendra Chandra Bh...	MATH	203456	Bat Man	Math
33	Uday Sankar Basak	MATH	203456	Bat Man	Math
34	Dr. Md. Azizur Rahman	MATH	203456	Bat Man	Math
35	Mira Khatun	STAT	NULL	NULL	NULL
36	NULL	NULL	123456	Groot	Physics
37	NULL	NULL	205678	Spider Man	Bangla
38	NULL	NULL	209012	Thanos	English

Result and Discussion: In the first Table we can notice that there is a very big table because of the cartesian product, with each tuple of the student table multiplies with the whole instructor table that's why it's such a large table. In the second Table we can see that only those tuples are available where the department of both student and instructor are same. In the 3rd Table we have performed the left outer join that why all values of left table are here if corresponding right tablet values are not present then it is indicates as null. In the final Table we can see we right outer join. It is same as the left outer join but only right tables all value are present. Here we have taken the student table as the left table and instructor table as the right table. So, from the output we can say that the experiment went successfully.

Experiment Number: 06

Name of the Experiment: To Study and Implementation of Aggregate Function with Example:

- Count Function
- Max Function
- Min Function
- Avg Function

Objectives:

1. To understand the concept of the aggregate function.
2. To understand the working process of *count*, *max, min* and *average* function.

Theory: Aggregate functions are a set of operations in SQL that perform calculations on a group of rows and return a single result. They summarize and provide insights into the data in a database table. Common aggregate functions include COUNT (to count the number of rows), SUM (to calculate the total of a numeric column), AVG (to find the average of values), MAX (to determine the maximum value), and MIN (to find the minimum value within a group of rows. These functions are essential for generating statistical and summary information from data, especially when working with large datasets. Certainly, here are concise definitions for the count, average (AVG), minimum (MIN), and maximum (MAX) aggregate functions in SQL:

COUNT: The COUNT aggregate function is used to determine the number of rows in a dataset that meets a specific condition. It is often used to count the occurrences of rows in a column, helping us understand the size of a dataset or the number of records that meet certain criteria.

AVERAGE (AVG): The AVG aggregate function calculates the arithmetic mean of values in a numeric column. It provides the average value of a set of numbers, helping to assess the typical value within a dataset.

MINIMUM (MIN): The MIN aggregate function identifies the lowest value in a dataset for a specified column. It is used to find the smallest value in a column, which can be valuable for determining the minimum or starting point of a dataset.

MAXIMUM (MAX): The MAX aggregate function locates the highest value in a dataset for a specified column. It helps to find the largest value in a column, which is essential for identifying the maximum or peak value in a dataset.

Experimental Requirements:

1. Laptop/ Desktop Computer.
2. Microsoft SQL Server application.
3. Microsoft SQL Server Management Studio application.

Source Code:

```
use Sayeeda_Khan

create table instructor(
       ID varchar(20),
       Ins_name varchar(50),
       dept_name varchar(50),
       salary numeric(9,0),
       primary key(ID))

insert into instructor
      values('ICE01','Dr. Pallab Kanti Poddar','ICE','100000'),
            ('ICE02','Dr. Md. Omar Faruk','ICE','100000'),
              ('ICE03','Md. Anwar Hossain','ICE','100000'),
              ('ICE04','Iffat Ara','ICE','100000'),
              ('ICE05','Sohag Sarker','ICE','100000'),
              ('ICE06','Md. Sarwar Hosain','ICE','100000'),
              ('ICE07','Abul Fazal Mohammad Zainul
Abadin','ICE','100000'),
              ('ICE08','Dr. Md. Imran Hossain','ICE','100000'),
              ('ICE09','Taskin Noor Turna','ICE','90000'),
              ('ICE10','Md. Tofail Ahmed','ICE','90000'),
              ('ICE11','Tarun Debnath','ICE','90000'),
              ('ICE12','Muntasir Ahmed','ICE','85000'),
              ('CSE01','Nitun Kumar Podder','CSE','70000'),
              ('EEE01','Md. Shamim Hossain','EEE','70000'),
              ('EEE02','Tonmoy Ghosh','EEE','70000'),
              ('Stat01','Mira Khatun','STAT','70000')


select * from instructor

select dept_name,count(ID) as Department_Teachers
from instructor
group by dept_name

select min(Ins_name) as TeacherMinname,dept_name
from instructor
group by  dept_name

select max(Ins_name) as TeacherMaxname,dept_name
from instructor
group by  dept_name

select avg(salary) as DepartmentAVG_salary,dept_name
```

```
from instructor
group by  dept_name
```

Output:

Table (1): Showing all data from instructor table in *Sayeeda_Khan* database.

	ID	Ins_name	dept_name	salary
1	CSE01	Nitun Kumar Podder	CSE	70000
2	EEE01	Md. Shamim Hossain	EEE	70000
3	EEE02	Tonmoy Ghosh	EEE	70000
4	ICE01	Dr. Pallab Kanti Poddar	ICE	100000
5	ICE02	Dr. Md. Omar Faruk	ICE	100000
6	ICE03	Md. Anwar Hossain	ICE	100000
7	ICE04	Iffat Ara	ICE	100000
8	ICE05	Sohag Sarker	ICE	100000
9	ICE06	Md. Sarwar Hosain	ICE	100000
10	ICE07	Abul Fazal Mohammad Zainul Abadin	ICE	100000
11	ICE08	Dr. Md. Imran Hossain	ICE	100000
12	ICE09	Taskin Noor Turna	ICE	90000
13	ICE10	Md. Tofail Ahmed	ICE	90000
14	ICE11	Tarun Debnath	ICE	90000
15	ICE12	Muntasir Ahmed	ICE	85000
16	Stat01	Mira Khatun	STAT	70000

Table (2): Table After applying the 'count' function.

	dept_name	Department_Teachers
1	CSE	1
2	EEE	2
3	ICE	12
4	STAT	1

Table (3): Table after applying the 'min' function.

	TeacherMinname	dept_name
1	Nitun Kumar Podder	CSE
2	Md. Shamim Hossain	EEE
3	Abul Fazal Mohammad Zainul Abadin	ICE
4	Mira Khatun	STAT

Table (4): Table after applying the 'max' function.

	TeacherMaxname	dept_name
1	Nitun Kumar Podder	CSE
2	Tonmoy Ghosh	EEE
3	Taskin Noor Turna	ICE
4	Mira Khatun	STAT

Table (5): Table after applying the 'avg' function.

	DepartmentAVG_salary	dept_name
1	70000.000000	CSE
2	70000.000000	EEE
3	96250.000000	ICE
4	70000.000000	STAT

Result and Discussion: First we have created an instructor table where there is an attribute called Salary which is numeric type the average () can only work on numeric data. Other functions can work with any data. The count function counts the total number of tuples it's used with group by function which groups the total number of instructors in a department. The "min" function finds smaller number while its numeric type but when its varchar in finds the name starts with minimum alphabet (a-z) it's also group by the department name on the other hand max function finds name starts with maximum alphabet group by department name. The Average function only works with numeric data, that is why it finds the average salary of each department.

Experiment Number: 07

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

Objectives: To understand the concept of Triggering System on Database Table Using SQL Commands.

Theory: A database trigger is a predefined action or piece of code that automatically executes in response to specific events or changes in a database table. These events can include data insertion, updates, or deletions. Triggers are used to enforce data integrity, perform audits, log changes, or automate tasks like sending notifications or updating related data. They are valuable tools for maintaining data accuracy, ensuring security, and streamlining database operations. Triggers act as a built-in alarm system, responding to predefined conditions in the database, reducing the need for manual intervention, and enhancing the efficiency and reliability of database management. A database trigger is like an automatic reaction that happens when certain events occur in a database table. It's a piece of code or set of actions that you can set up to run in response to specific activities, like inserting, updating, or deleting data in the table.

Imagine we have a "Tasks" table in our database, and we want to keep track of when a new task is added. We can create a trigger that automatically sends a notification or updates another table whenever a new task is inserted into the "Tasks" table.

Experimental Requirements:

1. Laptop/ Desktop Computer.
2. Microsoft SQL Server application.
3. Microsoft SQL Server Management Studio application.

Source Code:

```
use Sayeeda_Khan


create table CustomersAndSuppliers(
      customer_supplier_ID varchar(20),
      Customer_name varchar(50),
      sold_amount decimal(10,2) default 0.00,
      proc_amount decimal(10,2) default 0.00,
      primary key (customer_supplier_ID))
```

```
create table transactions(
      transaction_ID varchar(10),
      customer_supplier_ID varchar(20),
      transaction_type varchar(15),
      amount decimal(10,2),
      transaction_date date,
      foreign key(customer_supplier_ID) references
CustomersAndSuppliers(customer_supplier_ID))


CREATE TRIGGER trg_update_customers_suppliers
ON transactions -- Use square brackets to specify the table name
AFTER INSERT
AS
BEGIN
    -- Update the sold_amount or proc_amount based on the transaction type
    UPDATE cs
    SET sold_amount = CASE
                WHEN i.transaction_type = 'Sale' THEN cs.sold_amount +
i.amount
                ELSE cs.sold_amount
              END,
        proc_amount = CASE
                WHEN i.transaction_type = 'Purchase' THEN cs.proc_amount +
i.amount
                ELSE cs.proc_amount
              END
    FROM CustomersAndSuppliers cs
    JOIN INSERTED i ON cs.customer_supplier_ID = i.customer_supplier_ID;
END;



INSERT INTO transactions (transaction_ID, customer_supplier_ID,
transaction_type, amount, transaction_date)
VALUES
    ('1', 'ABC123', 'Sale', 100.00, '2023-10-01'),
    ('2', 'DEF456', 'Purchase', 50.00, '2023-10-02')
```

Experiment Number: 08

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

Objectives: To understand the concept of SQL Commands to Connect MySQL Database with Java or PHP.

Theory: To connect a MySQL database with PHP, we use PHP's MySQLi or PDO extensions. These extensions allow your PHP script to interact with the MySQL database server. We'll need to provide specific database credentials, including the server's hostname or IP address, a username, and a password. These credentials are used to authenticate your PHP script with the database.

Once the connection is established, we can send SQL queries to the database. These queries can retrieve, insert, update, or delete data. The results of these queries can be processed in our PHP script for various purposes, such as displaying data on a web page or performing calculations.

Proper error handling is essential to manage potential issues with database connections or query execution. Closing the database connection when it's no longer needed helps free up server resources. This process enables the creation of dynamic, data-driven web applications.

Experimental Requirements:

1. Laptop/ Desktop Computer.
2. Microsoft SQL Server application.
3. Microsoft SQL Server Management Studio application.

Source Code:

```
<?php
$connect=mysqli_connect("localhost","root","","student");

if(isset($_POST["insert"])){
    $id =$_POST["id"];
    $name=$_POST["name"];
    $sess=$_POST["session"];
    $phone=$_POST["ph_number"];
    $city=$_POST["city"];
```

```
    ;
    $insert="insert into Semester_Reg(ID,Name,Session,Phone_No,City)
     values('$id','$name','$sess', '$phone', '$city')";
    $result=mysqli_query($connect,$insert);
    if($result==1){
        echo"Successfully insert a record!";
    }else{
        echo"Unsucess";}
}if(isset($_POST["delete"])){
    $id =$_POST["id"];
    $name=$_POST["name"];
    $sess=$_POST["session"];
    $delete="delete from `semester_reg` where ID='$id'
    and Name='$name' and Session='$sess'";
    $result=mysqli_query($connect,$delete);
    if($result==1){
        echo"Successfully delete your record!";
    }else{
        echo"Unsucess";}
} if(isset($_POST["update"])){
    $id =$_POST["id"];
    $name=$_POST["name"];
    $sess=$_POST["session"];
    $phone=$_POST["ph_number"];
    $city=$_POST["city"];
    $insert="update semester_reg  set
Name='$name',Session='$sess',Phone_No='$phone',
     City='$city' where ID='$id'";
    $result=mysqli_query($connect,$insert);
    if($result==1){
        echo"Successfully updated your record!";
    }else{
        echo"Unsucess";}
}if(isset($_POST["select"])){
    $query="SELECT * FROM semester_reg";
    $result=mysqli_query($connect,$query);
    if($result==true){
        echo "All  Registered Students List <br>";
    echo "<table cellpadding=10 border='1'>
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Session</th>
        <th>Phone Number</th>
        <th>City</th>

    </tr>";
    if(mysqli_num_rows($result) > 0){
```

```
        while($row = mysqli_fetch_array($result)){
            echo "<tr>";
            echo "<td style='color:black'>" . $row['ID'] ."</td>";
            echo "<td style='color:black'>" . $row['Name'] . "</td>";
            echo "<td style='color:black'>" . $row['Session'] . "</td>";
            echo "<td style='color:black'>" . $row['Phone_No'] . "</td>";
            echo "<td style='color:black'>" . $row['City'] . "</td>";
            echo "</tr>";
        }echo "</table>";
    }
    } else{
        echo "No record found!";
    }
}
//end of show data

?>
<!DOCTYPE html>
<html>
<head>
    <title>Student Registration Form</title>
    <style type="text/css">
        body {
            text-align: center;
            font-family: Cambria, Cochin, Georgia, Times,
            'Times New Roman', serif, sans-serif;
            background-color: antiquewhite;
            padding: 10px;
        }h2 {
            font-size: 30px;
            margin-top: 20px;
            background-color:aquamarine;
            text-align: center;
        }table {
            margin: auto;
            font-size: 20px;
            border-collapse: collapse;
            background-color: blanchedalmond;
        }th, td {
            padding: 10px;
        }th {
            text-align: center;
            vertical-align: middle;
             }input[type="text"] {
            font-size: 20px;
            width: 100%;
            padding: 5px;
        }input[type="radio"] {
```

```
            margin-right: 15px;
        }label {
            color: crimson;
        }
    </style>
</head>
<body>
    <h2>Student's Registration Form</h2>
    <form method="post" action="">
        <table border="1">
            <tr>
                <th>ID</th>
                <td><input type="text" name="id" required></td>
            </tr>
            <tr>
                <th>Name</th>
                <td><input type="text" name="name" required></td>
            </tr>
            <tr>
                <th>Session</th>
                <td><input type="text" name="session" required></td>
            </tr>
            <tr>
                <th>Phone Number</th>
                <td><input type="text" name="ph_number"></td>
            </tr>
            <tr>
                <th>City</th>
                <td><input type="text" name="city" value=""></td>
            </tr>

            <tr>
                <td colspan="2" class="button-container">
                    <input type="submit" name="insert" value="Insert">
                    <input type="submit" name="select" value="Show">
                    <input type="submit" name="delete" value="Delete">
                    <input type="submit" name="update" value="Update">
                </td>
            </tr>
        </table>
        <br>
        <label style="color: red">N.B.</label>
        1. To delete a record, please enter your ID, Name, and
Session.<br>
        2. You can update all information except the ID Number.<br>
        3. To show all records, enter your ID, Name, and Session.
    </form>
</body>
```

```
</html>
```

SQL code

```
create database student
create table Semester_Reg(
      ID varchar (20),
      Name varchar (25),
      Phone_no char(11),
      City varchar(20),
      primary key(ID))
```

Output:

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**ID**	text	utf8mb4_general_ci		No	*None*			Change Drop More
☐	2	**Name**	text	utf8mb4_general_ci		No	*None*			Change Drop More
☐	3	**Session**	text	utf8mb4_general_ci		No	*None*			Change Drop More
☐	4	**Phone_No**	text	utf8mb4_general_ci		No	*None*			Change Drop More
☐	5	**City**	text	utf8mb4_general_ci		No	*None*			Change Drop More

Figure (1): "Student_Reg" table in "student" database in the XAMPP server.

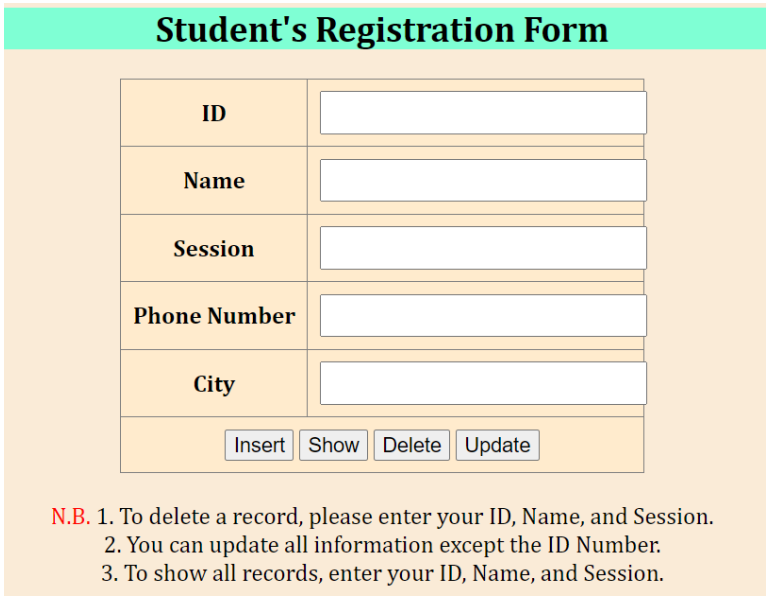


Figure (2): HTML form, from here we will collect the PHP values.

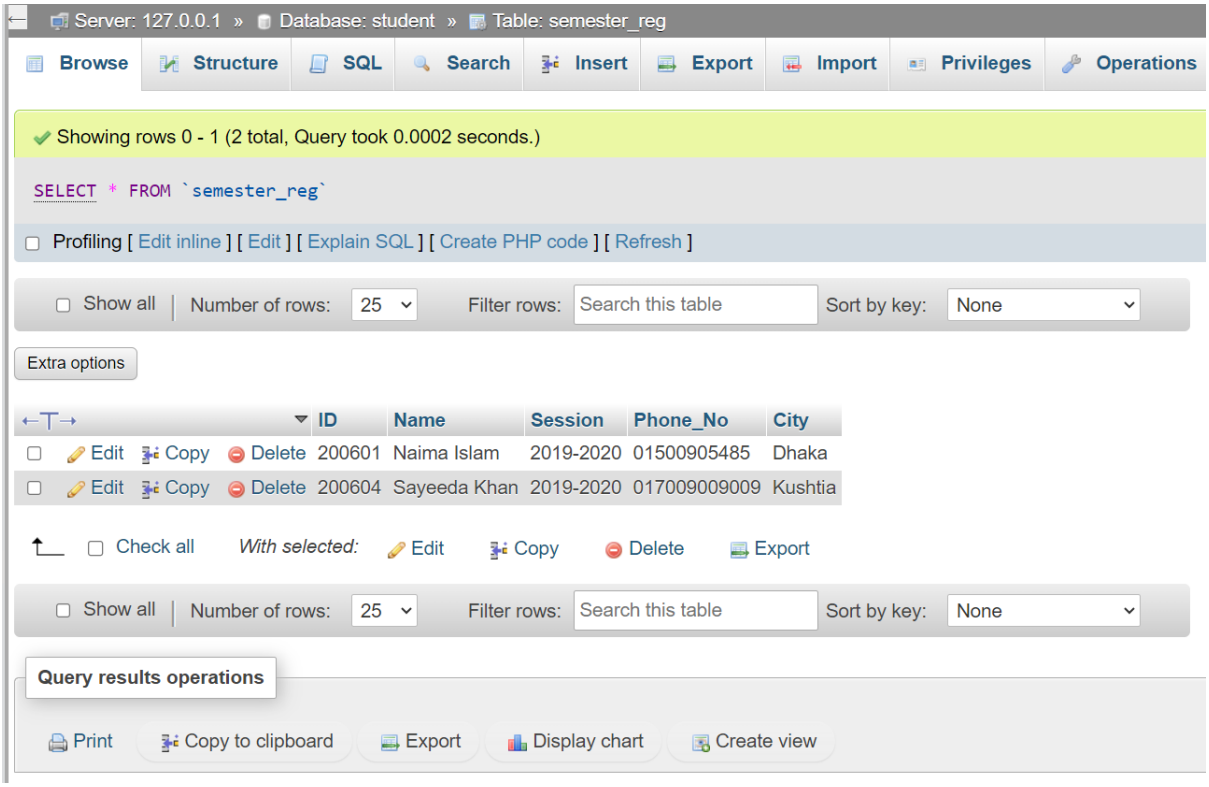


Figure (3): Inserted value "Student_Reg" table in "student" database in the XAMPP server via PHP.

Result and Discussion: From the HTML form we have collected the values and then we have sent the values in our corresponding PHP file via post method from the html form then we have passed the PHP values in our database and the database was created using the SQL and finally we can see that via html form we can submit information and the database is storing our given information via form correctly. So, from here we can say that our experiment was successful.