Experiment Number : 01

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

1.Select

2.Insert

3.Delete

4.Update

Objectives:

1. To understand and use data manipulation language to write query for database.

2. To study how to delete, insert and update data in database.

Theory:

SQL (**Structured Query Language**) is a standard programming language used for managing and manipulating relational databases. Data Manipulation Language (DML) commands in SQL deals with manipulation of data records stored within the database tables. DML (Data Manipulation Language) commands of SQL are used for adding, modifying, and deleting data in a database. we will study and implement the three DML commands of SQL: Select,Insert, Delete, and Update, along with suitable examples.

Select Comand:

Select command or statement in SQL is used to fetch data records from the database table and present it in the form of a result set. The basic syntax for writing a Select query in SQL is as follows :

Sql

```
SELECT column_name1, column_name2, …
FROM table_name
WHERE condition_ expression;
```

Insert Command:

The Insert command in SQL is used to insert new data into a table. The syntax for the Insert command is as follows:

sql

INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...);

Let's consider the following example to understand the Insert command in SQL:

Suppose we have a table named "students" with columns "id", "name", and "age". We want to add a new student to this table with id=101, name="John", and age=22. The SQL query for this would be:

sql

INSERT INTO students (id, name, age) VALUES (101, 'John', 22);

After executing this query, a new row would be added to the "students" table with the specified values.

Delete Command:

The Delete command in SQL is used to delete existing data from a table. The syntax for the Delete command is as follows:

sql

DELETE FROM table_name WHERE condition;

Let's consider the following example to understand the Delete command in SQL:

Suppose we have a table named "students" with columns "id", "name", and "age". We want to delete the row with id=101 from this table. The SQL query for this would be:

sql

DELETE FROM students WHERE id=101;

After executing this query, the row with id=101 would be deleted from the "students" table.

Update Command:

The Update command in SQL is used to modify existing data in a table. The syntax for the Update command is as follows:

sql

UPDATE table_name SET column1=value1, column2=value2, ... WHERE condition;

Let's consider the following example to understand the Update command in SQL:

Suppose we have a table named "students" with columns "id", "name", and "age". We want to update the name of the student with id=101 to "Jack". The SQL query for this would be:

sql

UPDATE students SET name='Jack' WHERE id=101;

After executing this query, the name of the student with id=101 would be updated to "Jack" in the "students" table.

Source Code:

```
-- Create a database
CREATE DATABASE StudentsDB;
-- Use the database
USE StudentsDB;
-- Create a table
CREATE TABLE Students (
    Id INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Age INT NOT NULL
);
-- Insert data into the table
INSERT INTO Students (Id, Name, Age)
VALUES (101, 'John', 22),
       (102, 'Jane', 23),
       (103, 'Alex', 21);
select *from Students;
-- Update data in the table
UPDATE Students SET Name='Jack' WHERE Id=101;

select *from Students;

-- Delete data from the table
DELETE FROM Students WHERE Id=103;

select *from Students;
drop table Students;
-- Select all data from the table
select *from Students;
```

Output:

Create and insert in Database:

	Id	Name	Age
1	101	John	22
2	102	Jane	23
3	103	Alex	21

Update Database:

	Id	Name	Age
1	101	Jack	22
2	102	Jane	23
3	103	Alex	21

Delete data from the table:

	Id	Name	Age
1	101	Jack	22
2	102	Jane	23

Experiment Number : 02

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

1. **Create**
2. **Alter**
3. **Drop**

Objectives:

1.To understand and use data definition language to write query for database

2.To study how to create alter and drop table in database.

Theory:

DDL is an abbreviation of **Data Definition Language**.The DDL Commands in Structured Query Language are used to create and modify the schema or structure of the database and its objects. The syntax of DDL commands is predefined for describing the data.

CREATE Command:

CREATE command is used for creating new databases, tables, views, indexes, and other database objects. The syntax of the CREATE command is as follows:

CREATE object_type object_name (

column1 data_type,

column2 data_type,

...);

Example:

Suppose we want to create a table named "students" in a database with columns "id," "name," "age," and "gender." The SQL query for creating the table would be:

CREATE TABLE students (

id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

age INT,

gender CHAR(1)

);

This query creates a new table named "students" with columns "id," "name," "age," and "gender." The "id" column is declared as the primary key, and the "name" column is declared as not null.

ALTER Command:

ALTER command is used for modifying the structure of existing databases, tables, views, indexes, and other database objects. The syntax of the ALTER command is as follows:

ALTER object_type object_name

ADD column_name data_type;

Example:

Suppose we want to add a new column "email" to the "students" table. The SQL query for adding the column would be:

ALTER TABLE students

ADD email VARCHAR(50);

This query adds a new column "email" with data type VARCHAR(50) to the "students" table.

DROP Command:

DROP command is used for deleting existing databases, tables, views, indexes, and other database objects. The syntax of the DROP command is as follows:

DROP object_type object_name;

Example:

Suppose we want to delete the "students" table. The SQL query for deleting the table would be:

DROP TABLE students;

This query deletes the "students" table from the database.

Source Code:

```
CREATE DATABASE mydb;

-- Use the "mydb" database

USE mydb;

-- CREATE Command: Creating a new table named "students" with columns "id," "name," "age," and "gender."

CREATE TABLE students (
id INT PRIMARY KEY,
name VARCHAR(50) NOT NULL,
age INT,
gender CHAR(1)
);

select  *from students;
-- ALTER Command: Adding a new column "email" to the "students" table.

ALTER TABLE students
ADD email VARCHAR(50);
select  *from students;
-- DROP Command: Deleting the "students" table.

DROP TABLE students;
select  *from students;
-- DROP Command: Deleting the "mydb" database.

DROP DATABASE mydb;
```

Output:

Create Table in Database:

id	name	age	gender

Alter in table column:

id	name	age	gender	email

Drop Table: The table will be empty.

Experiment Number : 03

Experiment Name : Study and Implementation of DML Commands of

1. **Select Clause**
2. **From Clause**
3. **Where Clause**

Objectives:

1. To understand and use of the sql queries.

2. To understand and how to implement select, from and where cluse in database.

Theory:

SELECT Clause:

SELECT clause is used to retrieve data from one or more tables in a database. The syntax of the SELECT clause is as follows:

SELECT column1, column2, ...

FROM table1;

Example:

Suppose we have a table named "students" with columns "id," "name," "age," and "gender." The SQL query for selecting all the data from the "students" table would be:

SELECT *FROM students;

This query retrieves all the data from the "students" table.

FROM Clause:

FROM clause is used to specify the table from which we want to retrieve data. The syntax of the FROM clause is as follows:

SELECT column1, column2, ...

FROM table1;

Example:

Suppose we want to retrieve data from the "students" table. The SQL query for selecting all the data from the "students" table would be:

SELECT *FROM students;

This query retrieves all the data from the "students" table.

WHERE Clause:

WHERE clause is used to filter data based on specific conditions. The syntax of the WHERE clause is as follows:

SELECT column1, column2, FROM table1

WHERE condition;

Example:

Suppose we want to retrieve data only for female students from the "students" table. The SQL query for selecting data based on a specific condition would be:

SELECT *FROM students

WHERE gender = 'F';

This query retrieves data only for female students from the "students" table.

Source Code:

```
CREATE TABLE students (
id INT PRIMARY KEY,
name VARCHAR(50) NOT NULL,
age INT,
gender CHAR(1)
);
INSERT INTO students (id, name, age, gender)
VALUES (1, 'John', 20, 'M'),
(2, 'Jane', 21, 'F'),
(3, 'Mike', 19, 'M'),
(4, 'Lisa', 22, 'F'),
(5, 'David', 23, 'M');
SELECT *FROM students;
SELECT id, name, age, gender
FROM students;
SELECT *FROM students
WHERE gender = 'F';
DROP TABLE students;
```

Output:

Select Clause:

	id	name	age	gender
1	1	John	20	M
2	2	Jane	21	F
3	3	Mike	19	M
4	4	Lisa	22	F
5	5	David	23	M

From Clause:

	id	name	age	gender
1	1	John	20	M
2	2	Jane	21	F
3	3	Mike	19	M
4	4	Lisa	22	F
5	5	David	23	M

WHERE Clause:

	id	name	age	gender
1	2	Jane	21	F
2	4	Lisa	22	F

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:

1.To study and implementation of DML commands of

- Group By & Having Clause
- Order By Clause
- Create View, Indexing & Procedure Clause

Theory:

Group By & Having Clause:

The GROUP BY clause is used to group rows that have the same values in a specified column or set of columns. The HAVING clause is used to filter the groups based on a specified condition.Consider the following table "students" with the columns "name", "subject", and "marks":

name subject marks

Alice Math 85

Bob Science 70

Alice Science 90

Bob Math 75

Bob English 80

If we want to find the average marks for each student, we can use the following SQL query:

sql

SELECT name, AVG(marks)

FROM students

GROUP BY name;

The result would be:

name AVG(marks)

Alice 87.5

Bob 75

Order By Clause:

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

Continuing with the same "students" table, if we want to sort the result set in descending order based on the average marks, we can use the following SQL query:

sql

SELECT name, AVG(marks)

FROM students

GROUP BY name

HAVING AVG(marks) > 80

ORDER BY AVG(marks) DESC;

The result would be:

name AVG(marks)

Alice 87.5

Create View, Indexing & Procedure Clause:

The CREATE VIEW statement is used to create a virtual table based on the result set of a SELECT statement. This virtual table can be used like any other table in subsequent queries.

The CREATE INDEX statement is used to create an index on one or more columns of a table. Indexes can improve the performance of queries by allowing the database to find data more quickly.

The CREATE PROCEDURE statement is used to create a stored procedure, which is a set of SQL statements that can be executed as a single unit.

Consider the following table "employees" with the columns "id", "name", and "salary":

id	name	salary
1	Alice	50000
2	Bob	60000
3	Carol	70000
4	Dave	55000

If we want to create a view that shows the name and salary of employees who earn more than 55000, we can use the following SQL query:

sql

```
CREATE VIEW high_earners AS
SELECT name, salary FROM employees
WHERE salary > 55000;
```

We can then use this view in subsequent queries:

sql

```
SELECT *FROM high_earners
ORDER BY salary DESC;
```

Source Code:

```sql
-- Create a new database
CREATE DATABASE my_database;

-- Use the new database
USE my_database;

-- Create a new table
CREATE TABLE my_table (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  age INT,
  salary DECIMAL(10,2)
);

-- Insert some sample data into the table
INSERT INTO my_table (id, name, age, salary)
VALUES
  (1, 'John Doe', 30, 50000),
  (2, 'Jane Smith', 25, 60000),
  (3, 'Bob Johnson', 40, 75000),
  (4, 'Mary Jones', 35, 80000),
  (5, 'Mike Brown', 28, 45000);

-- Group By & Having Clause
SELECT name, AVG(salary) AS average_salary
FROM my_table
GROUP BY name
HAVING AVG(salary) > 55000;
```

```sql
-- Order By Clause
SELECT name, age, salary
FROM my_table
ORDER BY salary DESC;

-- Create View Clause
CREATE VIEW my_view AS
SELECT name, age, salary
FROM my_table
WHERE age >= 30;

-- Indexing
CREATE INDEX my_index ON my_table (name);

-- Procedure Clause
CREATE PROCEDURE my_procedure
AS
BEGIN
  SELECT name, age, salary
  FROM my_table
  WHERE age >= 30;
END;
```

Output:

Group By & Having Clause

	name	average_salary
1	Bob Johnson	75000.000000
2	Jane Smith	60000.000000
3	Mary Jones	80000.000000

Order By Clause:

	name	age	salary
1	Mary Jones	35	80000.00
2	Bob Johnson	40	75000.00
3	Jane Smith	25	60000.00
4	John Doe	30	50000.00
5	Mike Brown	28	45000.00

Create View Clause:

	name	age	salary
1	John Doe	30	50000.00
2	Bob Johnson	40	75000.00
3	Mary Jones	35	80000.00

Procedure:

	name	age	salary
1	John Doe	30	50000.00
2	Bob Johnson	40	75000.00
3	Mary Jones	35	80000.00

Experiment Number : 05

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

1. **Cartesian Product**
2. **Nutural Join**
3. **Left Outer Join**
4. **Right Outer Join**
5. **Full Outer Join**

Objectives:

1.To study and implementation of SQL commands of all operation

Theory:

Cartesian Product:

The Cartesian product, also known as a cross join, combines every row of one table with every row of another table. The syntax is as follows:

Natural Join:

The natural join is a type of join that matches rows from two tables based on their common column names. It returns only the rows that have matching values in both tables. The syntax is as follows:

Syntax: SELECT * FROM table1 NATURAL JOIN table2;

Left Outer Join:

The left outer join returns all the rows from the left table and the matched rows from the right table. If there are no matching rows in the right table, NULL values are returned. The syntax is as follows:

Syntax: SELECT * FROM table1 LEFT OUTER JOIN table2 ON table1.column = table2.column;

Right Outer Join:

The right outer join is similar to the left outer join but returns all the rows from the right table and the matched rows from the left table. If there are no matching rows in the left table, NULL values are returned. The syntax is as follows:

Syntax: SELECT * FROM table1 RIGHT OUTER JOIN table2 ON table1.column = table2.column;

Full Outer Join:

The full outer join returns all the rows from both tables and matches the rows where possible. If there are no matching rows in one or both tables, NULL values are returned. The syntax is as follows:

Syntax: SELECT * FROM table1 FULL OUTER JOIN table2 ON table1.column = table2.column;

Source Code:

```
CREATE DATABASE mydatabase;
USE mydatabase;
--create table--
CREATE TABLE table1 (
  id INT PRIMARY KEY,
  name VARCHAR(50)
);

CREATE TABLE table2 (
  id INT PRIMARY KEY,
  age INT
);
--insert value in table--
INSERT INTO table1 VALUES (1, 'Alice'), (2, 'Bob'), (3, 'Charlie');
INSERT INTO table2 VALUES (1, 25), (2, 30), (4, 40);
--cartesian product--
SELECT * FROM table1 CROSS JOIN table2;
--Natural Join--
SELECT * FROM table1 NATURAL JOIN table2;
--Left Outer Join--
SELECT * FROM table1 LEFT OUTER JOIN table2 ON table1.id = table2.id;
--Right Outer Join--
SELECT * FROM table1 RIGHT OUTER JOIN table2 ON table1.id = table2.id;
--Full Outer Join--
SELECT * FROM table1 FULL OUTER JOIN table2 ON table1.id = table2.id;
```

Output:

Cartesian Product:

	id	name	id	age
1	1	Alice	1	25
2	2	Bob	1	25
3	3	Ch…	1	25
4	1	Alice	2	30
5	2	Bob	2	30
6	3	Ch…	2	30
7	1	Alice	4	40
8	2	Bob	4	40
9	3	Ch…	4	40

Natural Join:

Left Outer Join:

	id	name	id	age
1	1	Alice	1	25
2	2	Bob	2	30
3	3	Charlie	NULL	NULL

Right Outer Join:

	id	name	id	age
1	1	Alice	1	25
2	2	Bob	2	30
3	NULL	NULL	4	40

Full Outer Join:

	id	name	id	age
1	1	Alice	1	25
2	2	Bob	2	30
3	3	Charlie	NULL	NULL
4	NULL	NULL	4	40

Experiment Number : 06

Experiment Name : Study and Implementation of Aggregate Function with Example

Count Function

Max Function

Min Function

Avg Function

Objectives:

1.To study and implementation of Aggregate Function.

Theory:

Count Function:

The count function returns the number of rows that match a specified condition or all rows in a table. The syntax is as follows:

Syntax: SELECT COUNT(*) FROM table_name;

Syntax: SELECT COUNT(column_name) FROM table_name WHERE condition;

Max Function:

The max function returns the maximum value in a column. The syntax is as follows:

Syntax: SELECT MAX(column_name) FROM table_name WHERE condition;

Min Function:

The min function returns the minimum value in a column. The syntax is as follows:

Syntax: SELECT MIN(column_name) FROM table_name WHERE condition;

Avg Function:

The avg function returns the average value of a column. The syntax is as follows:

Syntax: SELECT AVG(column_name) FROM table_name WHERE condition;

Source Code:

```
--Creating Database:
CREATE DATABASE mydatabase6;
USE mydatabase6;
--Creating Tables and insert
CREATE TABLE employees (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  age INT,
  salary FLOAT
);
INSERT INTO employees VALUES (1, 'Alice', 25, 5000), (2, 'Bob', 30, 8000), (3, 'Charlie', 35, 6500);
--Count Function:
SELECT COUNT(*) FROM employees;
SELECT COUNT(age) FROM employees WHERE salary > 6000;
--Max Function:
SELECT MAX(salary) FROM employees;
SELECT MAX(age) FROM employees WHERE salary > 6000;
--Min Function:
SELECT MIN(salary) FROM employees;
SELECT MIN(age) FROM employees WHERE salary > 6000;
--Avg Function:
SELECT AVG(salary) FROM employees;
SELECT AVG(age) FROM employees WHERE salary > 6000;
```

Output:

Count Function:

	(No column name)
1	3

	(No column name)
1	2

Max Function:

	(No column name)
1	8000

	(No column name)
1	35

Min Function:

	(No column name)
1	5000

	(No column name)
1	30

Avg Function:

	(No column name)
1	6500

	(No column name)
1	32

Experiment Number : 07

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

Objectives:

1.To study and implementation of Triggering System on Database Table.

Theory:

A trigger is a special type of stored procedure that is automatically executed in response to certain database events, such as a change to a table or a data manipulation statement like INSERT, UPDATE, or DELETE. In this section, I will provide an example of how to implement a trigger on a database table using SQL commands.

Source Code:

```
--Creating Database:
CREATE DATABASE mydatabase7;
USE mydatabase7;
--Creating Table:
CREATE TABLE students (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  age INT,
  grade FLOAT
);
INSERT INTO students VALUES (1, 'Alice', 20, 3.5), (2, 'Bob', 22, 3.2), (3, 'Charlie', 19, 3.8);
--Creating Trigger:
CREATE TRIGGER update_grade
AFTER UPDATE ON students
FOR EACH ROW
BEGIN
  IF NEW.age > OLD.age THEN
    UPDATE students SET grade = grade + 0.1 WHERE id = NEW.id;
  ELSEIF NEW.age < OLD.age THEN
    UPDATE students SET grade = grade - 0.1 WHERE id = NEW.id;
  END IF;
END;

--Here is an example of how to use this trigger:
-- This will add 0.1 to Alice's grade
UPDATE students SET age = 21 WHERE id = 1;

-- This will subtract 0.1 from Bob's grade
UPDATE students SET age = 21 WHERE id = 2;
```

Output:

Experiment Number : 08

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

Objectives:

1.To study and implementation of Connect MySQL Database with Java or PHP.

Theory:

Create a Database: Once you have installed the MySQL driver, you need to create a database. You can use the MySQL command line or GUI tools like phpMyAdmin to create a database.

Create a Table: Once you have created the database, you need to create a table to store your data. You can use the MySQL command line or GUI tools to create a table.

Connect to MySQL Database: To connect to the MySQL database from PHP, you need to use the following code:

Source Code:

```php
<?php
$connect=mysqli_connect("localhost","root","","test");
//Insert start
if(isset($_POST["insert"]))
{
    $id =$_POST["id"];
    $name=$_POST["name"];
    $phone=$_POST["ph_number"];
    $insert="insert into test(ID,Name,Ph_Num) values('$id','$name', '$phone')";
    $result=mysqli_query($connect,$insert);
    if($result==1)
    {
        echo"Successfully insert a record!";
    }
}
//Insert end
if(isset($_POST["select"])){

$query="SELECT * FROM test"; //ORDER BY id ASC";";
$result=mysqli_query($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>Phone Number</th>
</tr>";
 if(mysqli_num_rows($result) > 0)
{
while($row = mysqli_fetch_array($result))
{
echo "<tr>";
```

```php
echo "<td style='color:black'>" . $row['ID'] ."</td>";
echo "<td style='color:black'>" . $row['Name'] . "</td>";
echo "<td style='color:black'>" . $row['Ph_Num'] . "</td>";
echo "</tr>";
}
echo "</table>";
}
} else
{
    echo "No record found!";
}
}
?>
```

Output:

All Registered Students List

ID	Name	Phone Number
121	test	01777417448
190610	Md. Sha Alam	01777417448

Student's Registration Form

ID

Name

Phone Number

Insert Show