A.Y. 2020-2021
Class: SE-ITA/B, Semester: III

Subject: **Structured Query Lab**

Experiment – 4: Perform data manipulations operations on populated database using DML queries and Aggregate functions

1. Aim:
To Perform data manipulations operations on populated database using SQL DML and Aggregate Functions for your chosen system

2. Objective:
· To Apply SQL to store and retrieve data efficiently using DML queries
· To apply SQL aggregate functions

3. Outcome: L302.3: Formulate and write SQL queries for efficient information retrieval

4. Prerequisite: Understanding of various DML queries, Aggregate functions notations and terminologies along with sample syntax.

5. Requirements: PC, Oracle 11g/SQL Server 2008 R2, Microsoft Word, Internet, MySQL workbench

6. Pre-Experiment Exercise:
Brief Theory:(To be hand written/typed)
Data Definition Language (DML)
Explain what is DML?
A data manipulation language (DML) is a computer programming language used for adding (inserting), deleting, and modifying (updating) data in a database . A DML is often a sublanguage
of a broader database language such as SQL , with the DML comprising some of the operators in
the language.Read-only selecting of data is sometimes distinguished as being part of a separate data query language (DQL), but it is closely related and sometimes also considered a component
of a DML; some operators may perform both selecting (reading) and writing.
A popular data manipulation language is that of Structured Query Language (SQL), which is used to retrieve and manipulate data in a relational database . Other forms of DML are those used
by IMS /DLI, CODASYL databases, such as IDMS and others.

What are aggregate functions?
In database management , an aggregate function or aggregation function is a function where the values of multiple rows are grouped together to form a single summary value .
Common aggregate functions include:
- Average (i.e., arithmetic mean)
- Count
- Maximum
- Median
- Minimum
- Mode

● Range
● Sum
Others include:
● Nanmean (mean ignoring NaN values, also known as "nil" or "null")
● Stddev
Formally, an aggregate function takes as input a set , a multiset (bag), or a list from some input domain I and outputs an element of an output domain O . The input and output domains may be the same, such as for SUM , or may be different, such as for COUNT .
Aggregate functions occur commonly in numerous programming languages , in spreadsheets , and in relational algebra .

7. Laboratory Exercise

A. Procedure:

i) A) Open SQL server 2008 using below login credentials:
Username: sa, Password: Lab301a (On college systems)

B) Open MySQL workbench
Enter appropriate username and password
ii) Create a new query
iii) Construct your own database
iv) Construct tables for any two to three entities from your chosen case study v) Use DML queries to manipulate the tables

· Use Insert

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

· Update

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

· Delete

```
DELETE FROM table_name WHERE condition;
```

Aggregate Functions

MIN()

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

MAX()

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

COUNT()

```
SELECT COUNT(column_name)
FROM table_name
```

```
WHERE condition;
```

AVG()

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

SUM()

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

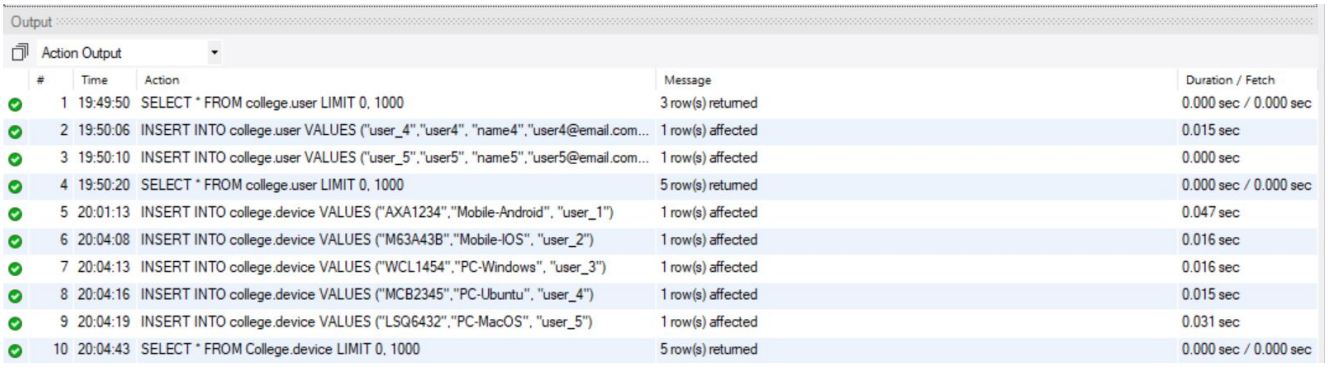
vi)Write/Print output for each query

B. **Result/Observation/Program code:** Attach all queries executed code with proper output

```
INSERT INTO college.user VALUES ("user_1","user",
"name","user@email.com","user1234",9123456780,true,20,'2000-01-01');
INSERT INTO college.user VALUES ("user_2","user2",
"name2","user2@email.com","user1234",9475381305,true,19,'2001-04-11');
INSERT INTO college.user VALUES ("user_3","user3",
"name3","user3@email.com","user1234",9475381306,true,21,'2000-04-11');
INSERT INTO college.user VALUES ("user_4","user4",
"name4","user4@email.com","user1234",9043642028,true,17,'2003-04-11');
INSERT INTO college.user VALUES ("user_5","user5",
"name5","user5@email.com","user1234",9015004314,true,16,'2004-04-11');
INSERT INTO college.user VALUES ("user_6","user6",
"name6","user6@email.com","user1234",9034006814,true,15,'2005-09-11');
INSERT INTO college.user VALUES ("user_7","user7",
"name7","user7@email.com","user1234",9035709124,true,20,'2000-09-11');

UPDATE College.user
SET phone_no = 8137403130, age = 18, DOB = '2002-04-11'
WHERE user_name = "user_3";

UPDATE College.user
SET DOB = '2004-10-11'
WHERE user_name = "user_5";

UPDATE College.user
SET DOB = '2003-06-11'
WHERE user_name = "user_4";

UPDATE College.user
```

```
SET DOB = '2001-03-11'
WHERE user_name = "user_2";

INSERT INTO college.device VALUES ("AXA1234","Mobile-Android", "user_1");
INSERT INTO college.device VALUES ("M63A43B","Mobile-IOS", "user_2");
INSERT INTO college.device VALUES ("WCL1454","PC-Windows", "user_3");
INSERT INTO college.device VALUES ("MCB2345","PC-Ubuntu", "user_4");
INSERT INTO college.device VALUES ("LSQ6432","PC-MacOS", "user_5");
INSERT INTO college.device VALUES ("BAS1245","PC-Windows", "user_6");
INSERT INTO college.device VALUES ("XMP3464","Mobile-iPadOS", "user_7");

INSERT INTO college.device VALUES ("BAS1245","PC-Windows", "user_1");
INSERT INTO college.device VALUES ("XMP3464","Mobile-iPadOS", "user_2");

SELECT * FROM College.device;

DELETE FROM College.device WHERE user_id = "user_6";
DELETE FROM College.device WHERE user_id = "user_7";

DELETE FROM College.user WHERE user_name = "user_6";
DELETE FROM College.user WHERE user_name = "user_7";

SELECT MIN(age) AS min_age
FROM College.user;

SELECT MAX(age) AS max_age
FROM College.user;

SELECT COUNT(user_name) AS user_count
FROM College.user;

SELECT SUM(age) AS sum_age
FROM College.user;

SELECT AVG(age) AS avg_age
FROM College.user;
```

Insert Output

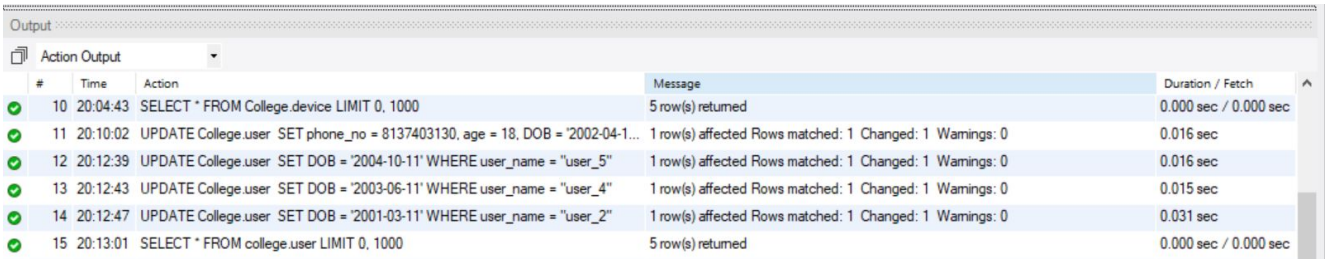


Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	19:49:50	SELECT * FROM college.user LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
2	19:50:06	INSERT INTO college.user VALUES ("user_4","user4", "name4","user4@email.com...	1 row(s) affected	0.015 sec
3	19:50:10	INSERT INTO college.user VALUES ("user_5","user5", "name5","user5@email.com...	1 row(s) affected	0.000 sec
4	19:50:20	SELECT * FROM college.user LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
5	20:01:13	INSERT INTO college.device VALUES ("AXA1234","Mobile-Android", "user_1")	1 row(s) affected	0.047 sec
6	20:04:08	INSERT INTO college.device VALUES ("M63A43B","Mobile-IOS", "user_2")	1 row(s) affected	0.016 sec
7	20:04:13	INSERT INTO college.device VALUES ("WCL1454","PC-Windows", "user_3")	1 row(s) affected	0.016 sec
8	20:04:16	INSERT INTO college.device VALUES ("MCB2345","PC-Ubuntu", "user_4")	1 row(s) affected	0.015 sec
9	20:04:19	INSERT INTO college.device VALUES ("LSQ6432","PC-MacOS", "user_5")	1 row(s) affected	0.031 sec
10	20:04:43	SELECT * FROM College.device LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Update Output

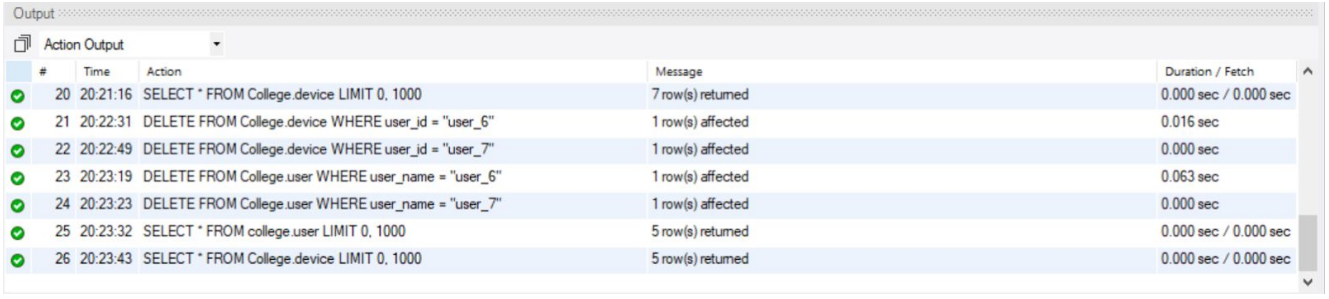


Output

Action Output

#	Time	Action	Message	Duration / Fetch
10	20:04:43	SELECT * FROM College.device LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
11	20:10:02	UPDATE College.user SET phone_no = 8137403130, age = 18, DOB = '2002-04-1...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
12	20:12:39	UPDATE College.user SET DOB = '2004-10-11' WHERE user_name = "user_5"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
13	20:12:43	UPDATE College.user SET DOB = '2003-06-11' WHERE user_name = "user_4"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
14	20:12:47	UPDATE College.user SET DOB = '2001-03-11' WHERE user_name = "user_2"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.031 sec
15	20:13:01	SELECT * FROM college.user LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Delete Output

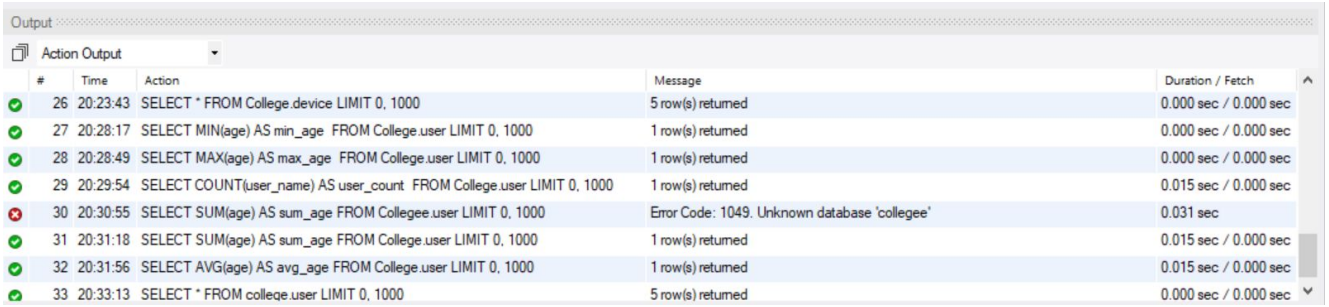


Output

Action Output

#	Time	Action	Message	Duration / Fetch
20	20:21:16	SELECT * FROM College.device LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
21	20:22:31	DELETE FROM College.device WHERE user_id = "user_6"	1 row(s) affected	0.016 sec
22	20:22:49	DELETE FROM College.device WHERE user_id = "user_7"	1 row(s) affected	0.000 sec
23	20:23:19	DELETE FROM College.user WHERE user_name = "user_6"	1 row(s) affected	0.063 sec
24	20:23:23	DELETE FROM College.user WHERE user_name = "user_7"	1 row(s) affected	0.000 sec
25	20:23:32	SELECT * FROM college.user LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
26	20:23:43	SELECT * FROM College.device LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Aggregate Functions



Output

Action Output

#	Time	Action	Message	Duration / Fetch
26	20:23:43	SELECT * FROM College.device LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
27	20:28:17	SELECT MIN(age) AS min_age FROM College.user LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
28	20:28:49	SELECT MAX(age) AS max_age FROM College.user LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
29	20:29:54	SELECT COUNT(user_name) AS user_count FROM College.user LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
30	20:30:55	SELECT SUM(age) AS sum_age FROM Collegee.user LIMIT 0, 1000	Error Code: 1049. Unknown database 'collegee'	0.031 sec
31	20:31:18	SELECT SUM(age) AS sum_age FROM College.user LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
32	20:31:56	SELECT AVG(age) AS avg_age FROM College.user LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
33	20:33:13	SELECT * FROM college.user LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

8. Post Experimental Exercise

A. Questions:

1. Advantages of using Aggregate Functions

An aggregate function allows you to perform a calculation on a set of values to return a single scalar value. We often use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

Sometimes we need aggregate functions for data analysis. These functions can be used for verifying individual column values or for comparing source data with another system to confirm aggregated values.

These functions are usually used with other SQL features to get the desired resultset.

B. Conclusion:

In this experiment we have written SQL scripts to implement various DML queries to populate the database, update the database and delete values from the database. Also we have implemented various Aggregate functions in this experiment.

A data-manipulation language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:

- Retrieval of information stored in the database.
- Insertion of new information into the database.
- Deletion of information from the database.
- Modification of information stored in the database.

Aggregate functions are functions that take a collection (a set or multiset) of values as input and return a single value. SQL offers five standard built-in aggregate functions:
Average: avg, Minimum: min, Maximum: max, Total: sum, Count: count

One of the most utilized ways to communicate with relational database management systems (RDBMS) is through Structured Query Languages (SQL). DML helps the user to access and manipulate data in the database.

C. References:

[1] Elmasri and Navathe, “Fundamentals of Database Systems”, 5th Edition, PEARSON Education.

[2] Korth, Silberchatz, Sudarshan, “Database System Concepts”, 6th Edition, McGraw – Hill

[3] https://www.w3schools.com/sql/sql_default.asp