Assignment A3

Date:

TITLE	Design at least 10 SQL queries for suitable database application using SQL.
PROBLEM STATEMENT / DEFINITION	Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions and set operators
LEARNING OBJECTIVE	• To understand & implement the various DML Commands. • To understand database concepts like functions and set operators.
LEARNING OUTCOME	The students will be able to • Implement the various DML Commands with options. • Implement database concepts like functions and set operators.
S/W PACKAGES & HARDWARE APPARATUS USED	• MySQL • 64-bit Linux based open source OS • 8 GB RAM

CONCEPT RELATED THEORY:

DML is short name of Data Manipulation Language which deals with data manipulation, and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE etc, and it is used to store, modify, retrieve, delete and update data in database.

SELECT: MySQL SELECT statement is used to fetch data from a database table.

Syntax: SELECT column_name(s) FROM table_name

INSERT: MySQL Query statement “INSERT ” is used to insert new records in a table

Syntax: INSERT INTO table_name (column, column1, column2, column3, ...) VALUES (value, value1, value2, value3 ...)

UPDATE: The UPDATE statement is used to modify data in a table.

Syntax: UPDATE table_name

SET column=value, column1=value1,…

WHERE someColumn=someValue

DELETE: The DELETE FROM statement is used to delete data from a database table.

Syntax: DELETE FROM tableName

WHERE someColumn = someValue

SET-OPERATORS:-

UNION: It returns a union of two select statements. It is returning unique (distinct) values of them.

Syntax: SELECT * FROM table1

UNION

SELECT * FROM table2;

UNION ALL

Similar to UNION just that UNION ALL returns also the duplicated values.

Syntax: SELECT * FROM table1

UNION

SELECT * FROM table2;

When using UNION and UNION ALL columns in SELECT statements need to match. This would return an error:

Syntax :

SELECT column1 FROM table1

UNION

SELECT * FROM table2;

MINUS

MINUS (also known as EXCEPT) returns the difference between the first and second SELECT statement. It is the one where we need to be careful which statement will be put first, because we will get only those results that are in the first SELECT statement and not in the second.

Syntax: SELECT * FROM table1

MINUS

SELECT * FROM table2;

INTERSECT

INTERSECT is opposite from MINUS as it returns us the results that are both to be found in first and second SELECT statement.

Syntax: SELECT * FROM table1

INTERSECT

SELECT * FROM table2;

INPUT (in tables):

```
insert into customer values
    -> (1, "S", "Sharma", "ABC", "Bandra", "Mumbai", "9999999999")
```

```
    -> (2, "R", "Kulkarni", "DEF", "Kothrud", "Pune", "8888888888"),
    -> (3, "P", "Singh", "GHI", "Rajiv Chowk", "New Delhi", "7777777777"),
    -> (4, "J", "Kapoor", "JKL", "Sector-50", "Noida", "6666666666"),
    -> (5, "K", "D'Souza", "MNO", "Andheri(W)", "Mumbai", "5555555555"),
    -> (6, "P", "Newalkar", "PQR", "Warje", "Pune", "4444444444"),
    -> (7, "S", "Holmes", "STU", "221 Baker St", "London", "3333333333"),
    -> (8, "T", "Stark", "SI", "10880", "Malibu", "2222222222"),
    -> (9, "E", "Macron", "VWX", "Gare de Lyon", "Paris", "1111111111"),
    -> (10, "H", "Lesnitsky", "ZZZ", "Polis", "Moscow", "9090909090");

insert into orders values
    -> (1, 1, 1, 1, "2020-08-24"),
    -> (2, 2, 2, 2, "2020-08-23"),
    -> (3, 3, 3, 3, "2020-08-22"),
    -> (4, 4, 4, 4, "2020-08-21"),
    -> (5, 5, 5, 5, "2020-08-20"),
    -> (6, 6, 6, 6, "2020-08-19");

insert into book values
    -> (1, "Hello", 123, "9999999999", "1111111111", "2006"),
    -> (2, "World", 100, "8888888888", "2222222222", "2015"),
    -> (3, "Bye", 200, "7777777777", "3333333333", "2015"),
    -> (4, "Boom", 340, "6666666666", "4444444444", "2014"),
    -> (5, "Thanks", 500, "5555555555", "5050505050", "2004"),
    -> (6, "Done", "360", "1010101010", "2020202020", "2000");

insert into author values
    -> ("9999999999", "KO", "India"),
    -> ("8888888888", "CJ", "Australia"),
    -> ("7777777777", "JKR", "UK"),
    -> ("6666666666", "IA", "Russia");

insert into publisher values
    -> ("1111111111", "AZ", "Pune", "2020"),
    -> ("2222222222", "BY", "Delhi", "2015"),
    -> ("3333333333", "CX", "Sydney", "2016"),
    -> ("4444444444", "DW", "Kyoto", "2015"),
    -> ("5050505050", "EV", "Seoul", "2020");
```

TESTCASES (description – input/output)

1. Insert at least 10 records in customer table and insert other tables accordingly.
2. Display all customer details with city pune and mumbai and customer first name starting with 'p' or 'h'.
3. lists the number of different customer cities.
4. Give 5% increase in price of the books with publishing year 2015.
5. Delete customer details living in pune.
6. Find the names of authors living in India or Australia .
7. Find the publishers who are established in year 2015 as well as in 2016
8. Find the book having maximum price and find titles of book having price between 300 and 400.
9. Display all titles of books with price and published year in decreasing order of publishing year.
10. Display title,author_no and publisher_no of all books published in 2000,2004,2006.

1. select * from customer;

cust_no	cust_fname	cust_lname	cust_company	cust_addr	city	cust_phone
1	S	Sharma	ABC	Bandra	Mumbai	9999999999
2	R	Kulkarni	DEF	Kothrud	Pune	8888888888
3	P	Singh	GHI	Rajiv Chowk	New Delhi	7777777777
4	J	Kapoor	JKL	Sector-50	Noida	6666666666
5	K	D'Souza	MNO	Andheri(W)	Mumbai	5555555555
6	P	Newalkar	PQR	Warje	Pune	4444444444
7	S	Holmes	STU	221 Baker St	London	3333333333
8	T	Stark	SI	10880	Malibu	2222222222
9	E	Macron	VWX	Gare de Lyon	Paris	1111111111
10	H	Lesnitsky	ZZZ	Polis	Moscow	9090909090

10 rows in set (0.00 sec)

2. select * from customer where city="Pune" OR city="Mumbai" OR cust_fname like 'p%' OR cust_fname like 'h%';

cust_no	cust_fname	cust_lname	cust_company	cust_addr	city	cust_phone
1	S	Sharma	ABC	Bandra	Mumbai	9999999999
2	R	Kulkarni	DEF	Kothrud	Pune	8888888888
3	P	Singh	GHI	Rajiv Chowk	New Delhi	7777777777
5	K	D'Souza	MNO	Andheri(W)	Mumbai	5555555555
6	P	Newalkar	PQR	Warje	Pune	4444444444
10	H	Lesnitsky	ZZZ	Polis	Moscow	9090909090

6 rows in set (0.00 sec)

3. select count(distinct city) from customer;

count(distinct city)
8

1 row in set (0.00 sec)

4. mysql> select * from book;

isbn	title	unit_price	author_no	publisher_no	publisher_year
1	Hello	123	9999999999	1111111111	2006
2	World	105	8888888888	2222222222	2015
3	Bye	210	7777777777	3333333333	2015
4	Boom	340	6666666666	4444444444	2014
5	Thanks	500	5555555555	5050505050	2004
6	Done	360	1010101010	2020202020	2000

6 rows in set (0.00 sec)

update book set unit_price=1.05*unit_price where publisher_year="2015";
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2 Changed: 2 Warnings: 0

select * from book;

isbn	title	unit_price	author_no	publisher_no	publisher_year
1	Hello	123	9999999999	1111111111	2020
2	World	105	8888888888	2222222222	2015
3	Bye	210	7777777777	3333333333	2015

```
|    4 | Boom   |        340 | 6666666666 | 4444444444   |           2014 |
|    5 | Thanks |        500 | 5555555555 | 5050505050   |           2005 |
|    6 | Done   |        360 | 1010101010 | 2020202020   |           2000 |
+------+--------+------------+------------+--------------+----------------+
5 rows in set (0.00 sec)
```

5. delete from customer where city = "Pune";
Query OK, 2 rows affected (1.16 sec)

select * from customer;

```
+---------+-----------+-----------+-------------+-------------+-----------+------------+
| cust_no | cust_fname | cust_lname | cust_company | cust_addr    | city      | cust_phone |
+---------+-----------+-----------+-------------+-------------+-----------+------------+
|       1 | S         | Sharma     | ABC          | Bandra       | Mumbai    | 9999999999 |
|       3 | P         | Singh      | GHI          | Rajiv Chowk  | New Delhi | 7777777777 |
|       4 | J         | Kapoor     | JKL          | Sector-50    | Noida     | 6666666666 |
|       5 | K         | D'Souza    | MNO          | Andheri(W)   | Mumbai    | 5555555555 |
|       7 | S         | Holmes     | STU          | 221 Baker St | London    | 3333333333 |
|       8 | T         | Stark      | SI           | 10880        | Malibu    | 2222222222 |
|       9 | E         | Macron     | VWX          | Gare de Lyon | Paris     | 1111111111 |
|      10 | H         | Lesnitsky  | ZZZ          | Polis        | Moscow    | 9090909090 |
+---------+-----------+-----------+-------------+-------------+-----------+------------+
8 rows in set (0.01 sec)
```

6. select * from author;

```
+------------+-------------+-----------+
| author_no  | author_name | country   |
+------------+-------------+-----------+
| 9999999999 | KO          | India     |
| 8888888888 | CJ          | Australia |
| 7777777777 | JKR         | UK        |
| 6666666666 | IA          | Russia    |
+------------+-------------+-----------+
4 rows in set (0.00 sec)
```

select * from author where country="India" or country="Australia";

```
+------------+-------------+-----------+
| author_no  | author_name | country   |
+------------+-------------+-----------+
| 9999999999 | KO          | India     |
| 8888888888 | CJ          | Australia |
+------------+-------------+-----------+
2 rows in set (0.00 sec)
```

7. select * from publisher;

```
+--------------+----------------+----------------+------+
| publisher_no | publisher_name | publisher_addr | year |
+--------------+----------------+----------------+------+
| 1111111111   | AZ             | Pune           | 2020 |
| 2222222222   | BY             | Delhi          | 2015 |
| 3333333333   | CX             | Sydney         | 2016 |
| 4444444444   | DW             | Kyoto          | 2015 |
| 5050505050   | EV             | Seoul          | 2020 |
+--------------+----------------+----------------+------+
5 rows in set (0.00 sec)
```

select * from publisher where year="2015" or year="2016";

```
+--------------+----------------+----------------+------+
| publisher_no | publisher_name | publisher_addr | year |
+--------------+----------------+----------------+------+
```

```
| 2222222222   | BY             | Delhi          | 2015 |
| 3333333333   | CX             | Sydney         | 2016 |
| 4444444444   | DW             | Kyoto          | 2015 |
+--------------+----------------+----------------+------+
3 rows in set (0.00 sec)
```

8. select title, unit_price from book where unit_price = (select MAX(unit_price) from book);

```
+--------+------------+
| title  | unit_price |
+--------+------------+
| Thanks |        500 |
+--------+------------+
1 row in set (0.00 sec)
```

select title, unit_price from book where unit_price >= 300 and unit_price <= 400;

```
+-------+------------+
| title | unit_price |
+-------+------------+
| Boom  |        340 |
| Done  |        360 |
+-------+------------+
2 rows in set (0.00 sec)
```

9. select title, unit_price, publisher_year from book order by publisher_year DESC;

```
+--------+------------+----------------+
| title  | unit_price | publisher_year |
+--------+------------+----------------+
| World  |        105 |           2015 |
| Bye    |        210 |           2015 |
| Boom   |        340 |           2014 |
| Hello  |        123 |           2006 |
| Thanks |        500 |           2004 |
| Done   |        360 |           2000 |
+--------+------------+----------------+
6 rows in set (0.00 sec)
```

10. select title, author_no, publisher_no from book where publisher_year =
"2000" or publisher_year = "2004" or publisher_year = "2006"

```
+--------+------------+--------------+
| title  | author_no  | publisher_no |
+--------+------------+--------------+
| Hello  | 9999999999 | 1111111111   |
| Thanks | 5555555555 | 5050505050   |
| Done   | 1010101010 | 2020202020   |
+--------+------------+--------------+
3 rows in set (0.00 sec)
```