K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch: A2 **Roll No.: 1911027**

Experiment / assignment / tutorial No. 4

Grade: AA / AB / BB / BC / CC / CD /DD

Title: DML – select, insert, update and delete

1.Group by, having clause, aggregate functions, Set Operations
2.Nested queries : AND,OR,NOT, IN, NOT IN, Exists, Not Exists, Between, Like, Alias, ANY,ALL,DISTINCT
3. Update
4. Delete

Objective: To perform various DML Operations and executing nested queries with various clauses.

Expected Outcome of Experiment:
CO 3: Use SQL for Relational database creation, maintenance and query processing

Books/ Journals/ Websites referred:

1. Dr. P.S. Deshpande, SQL and PL/SQL for Oracle 10g.Black book, Dreamtech Press
2. www.db-book.com
3. Korth, Slberchatz, Sudarshan : "Database Systems Concept", 5th Edition , McGraw Hill
4. Elmasri and Navathe,"Fundamentals of database Systems", 4th Edition PEARSON Education

Resources used: Postgres

Theory:

Select: The SQL **SELECT** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

Syntax

The basic syntax of the SELECT statement is as follows –

SELECT column1, column2, columnN FROM table_name;

Here, column1, column2... are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax.

SELECT * FROM table_name;

The following code is an example, which would fetch the ID, Name and Salary fields of the customers available in CUSTOMERS table.

SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS;

Insert: The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax

There are two basic syntaxes of the INSERT INTO statement which are shown below.

INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)

VALUES (value1, value2, value3,...valueN);

Example

The following statements would create record in the CUSTOMERS table.

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

Update: The SQL **UPDATE** Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

Syntax:

The basic syntax of the UPDATE query with a WHERE clause is as follows −

UPDATE table_name

SET column1 = value1, column2 = value2...., columnN = valueN

WHERE [condition];

You can combine N number of conditions using the AND or the OR operators.

The following query will update the ADDRESS for a customer whose ID number is 6 in the table.

```
SQL> UPDATE CUSTOMERS
SET ADDRESS = 'Pune'
WHERE ID = 6;
```

Delete: The SQL DELETE Query is used to delete the existing records from a table.

You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

Syntax

The basic syntax of the DELETE query with the WHERE clause is as follows −

```
DELETE FROM table_name
WHERE [condition];
```

The following code has a query, which will DELETE a customer, whose ID is 6.

```
SQL> DELETE FROM CUSTOMERS
WHERE ID = 6;
```

Clauses and Operators

1. **Group by clause:** These are circumstances where we would like to apply the aggregate functions to a single set of tuples but also to a group of sets of tuples we would like to specify this wish in SQL using the group by clause. The attributes or attributes given by the group by clause are used to form groups. Tuples with the same value on all attributes in the group by clause placed in one group.

Example:.

```
Select<attribute_name,avg(<attribute_name>)as
<new_attribute_name>l From <table_name>
Group by <attribute_name>
```

Example: select designation, sum(salary) as total_salary from employee group by Designation;

2. Having clause: A having clause is like a where clause but only applies only to groups as a whole whereas the where clause applies to the individual rows. A query can contain both where clause and a having clause. In that case

a. The where clause is applied first to the individual rows in the tables or table structures objects in the diagram pane. Only the rows that meet the conditions in the where clause are grouped.

b. The having clause is then applied to the rows in the result set that are produced by grouping. Only the groups that meet the having conditions appear in the query output.

Example:

```
select dept_no from EMPLOYEE group_by dept_no
having avg (salary) >=all (select avg (salary)
from EMPLOYEE group by dept_no);
```

3. Aggregate functions: Aggregate functions such as SUM, AVG, count, count (*), MAX and MIN generate summary values in query result sets. An aggregate functions (with the exception of count (*) processes all the selected values in a single column to produce a single result value

Example: select dept_no,count (*)
from EMPLOYEE group by dept_no;

Example: select max (salary)as maximum from EMPLOYEE;

Example: select sum (salary) as total_salary from EMPLOYEE;

Example: Select min (salary) as minsal from EMPLOYEE;

4. Exists and Not Exists: Subqueries introduced with exists and not queries can be used for two set theory operations: Intersection and Difference. The intersection of two sets contains all elements that belong to both of the original sets. The difference contains elements that belong to only first of the two sets.

Example:

```
Select *from DEPARTMENT
where exists(select * from PROJECT
            where DEPARTMENT.dept_no = PROJECT.dept_no) ;
```

5. IN and Not In: SQL allows testing tuples for membership in a relation. The "in" connective tests for set membership where the set is a collection of values produced by select clause. The "not in" connective tests for the absence of set membership. The in and not in connectives can also be used on enumerated sets.

Example:

1. Select fname, mname, lname from employee where designation In ("ceo","manager","hod","assistant")

2. Select fullname from department where relationship not in("brother");

6. Between: The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive. Begin and end values are included.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Example:

```
SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;
```

7. LIKE: The LIKE **operator** is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

Syntax: SELECT *column1, column2, ...*
FROM *table_name*
WHERE *columnN* LIKE *pattern*

Examples:

1. selects all customers with a CustomerName starting with "a":

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';
```

2. selects all customers with a CustomerName that have "r" in the second position:

```
SELECT * FROM Customers
WHERE CustomerName LIKE '_r%';
```

8. Alias: The use of table aliases is to rename a table in a specific SQL statement. The renaming is a temporary change and the actual table name does not change in the database. The column aliases are used to rename a table's columns for the purpose of a particular SQL query.

The basic syntax of a **table** alias is as follows.

```
SELECT column1, column2....
FROM table_name AS alias_name
WHERE [condition];
```

The basic syntax of a **column** alias is as follows.

```
SELECT column_name AS alias_name
FROM table_name
WHERE [condition];
```

Example:

```
SELECT C.ID, C.NAME, C.AGE, O.AMOUNT
FROM CUSTOMERS AS C, ORDERS AS O
WHERE  C.ID = O.CUSTOMER_ID;
```

9. Distinct: The SELECT DISTINCT statement is used to return only distinct (different) values.

Syntax: SELECT DISTINCT *column1*, *column2*, ...
FROM *table_name*;

Example: SELECT DISTINCT Country FROM Customers;

10. Set Operations: 4 different types of SET operations, along with example:

1. UNION
2. UNION ALL
3. INTERSECT
4. MINUS

UNION Operation

UNION is used to combine the results of two or more SELECT statements. However it will eliminate duplicate rows from its resultset. In case of union, number of columns and datatype must be same in both the tables, on which UNION operation is being applied.

Query: SELECT * FROM First

UNION

SELECT * FROM Second;

UNION ALL

This operation is similar to Union. But it also shows the duplicate rows.

Query: SELECT * FROM First

UNION ALL

SELECT * FROM Second;

INTERSECT

Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements. In case of **Intersect** the number of columns and datatype must be same.

Query: SELECT * FROM First

INTERSECT

SELECT * FROM Second;

MINUS

The Minus operation combines results of two SELECT statements and return only those in the final result, which belongs to the first set of the result.

Query: SELECT * FROM First

MINUS

SELECT * FROM Second;

11. ANY and ALL: The ANY and ALL operators are used with a WHERE or HAVING clause. The ANY operator returns true if any of the subquery values meet the condition. The ALL operator returns true if all of the subquery values meet the condition.

ANY

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name operator* ANY
(SELECT *column_name* FROM *table_name* WHERE *condition*);

Example: The following SQL statement returns TRUE and lists the productnames if it finds ANY records in the OrderDetails table that quantity = 10:

SELECT ProductName
FROM Products
WHERE ProductID
= ANY (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);

ALL

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name operator* ALL
(SELECT *column_name* FROM *table_name* WHERE *condition*);

Example: The following SQL statement returns TRUE and lists the productnames if ALL the records in the OrderDetails table has quantity = 10:

SELECT ProductName
FROM Products
WHERE ProductID
= ALL (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);

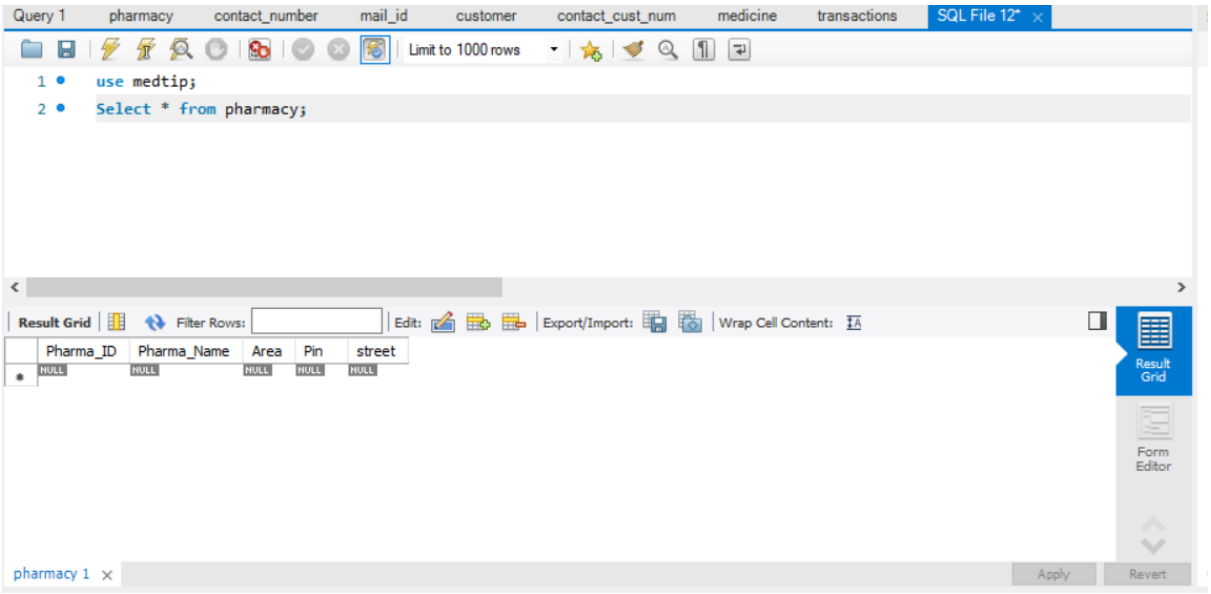
Implementation details

- Simple question based on your application, queries and screen shots for each type:

Q) select all tuples from pharmacy table.

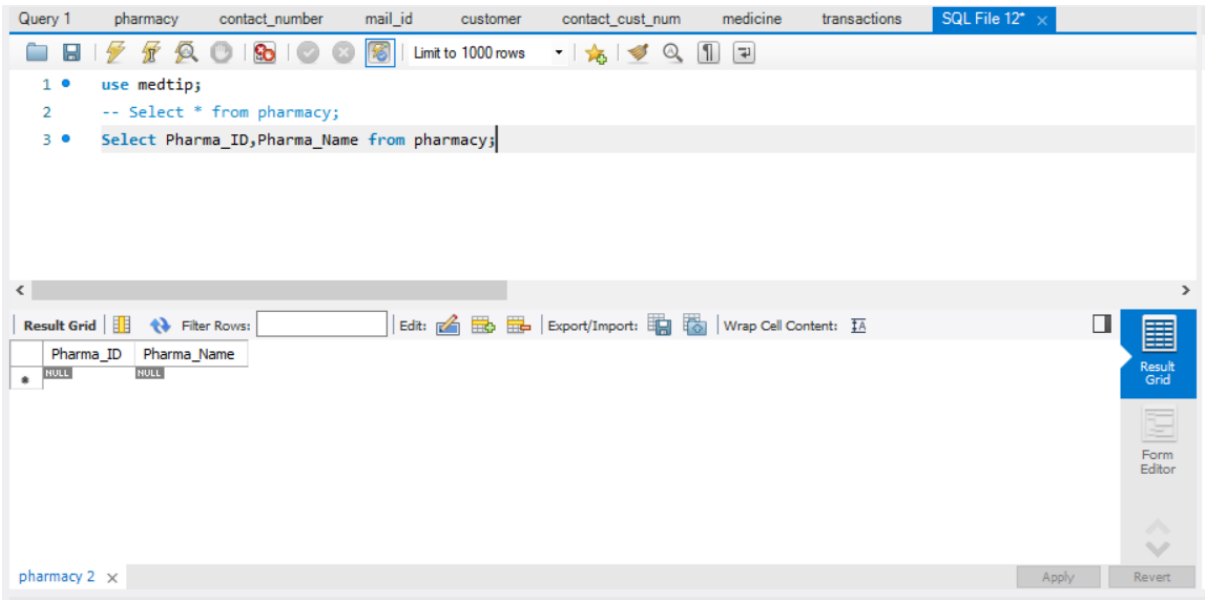
```
use medtip;

Select * from pharmacy;
```



Q) select Pharma_ID and Pharma_Name from pharmacy table.

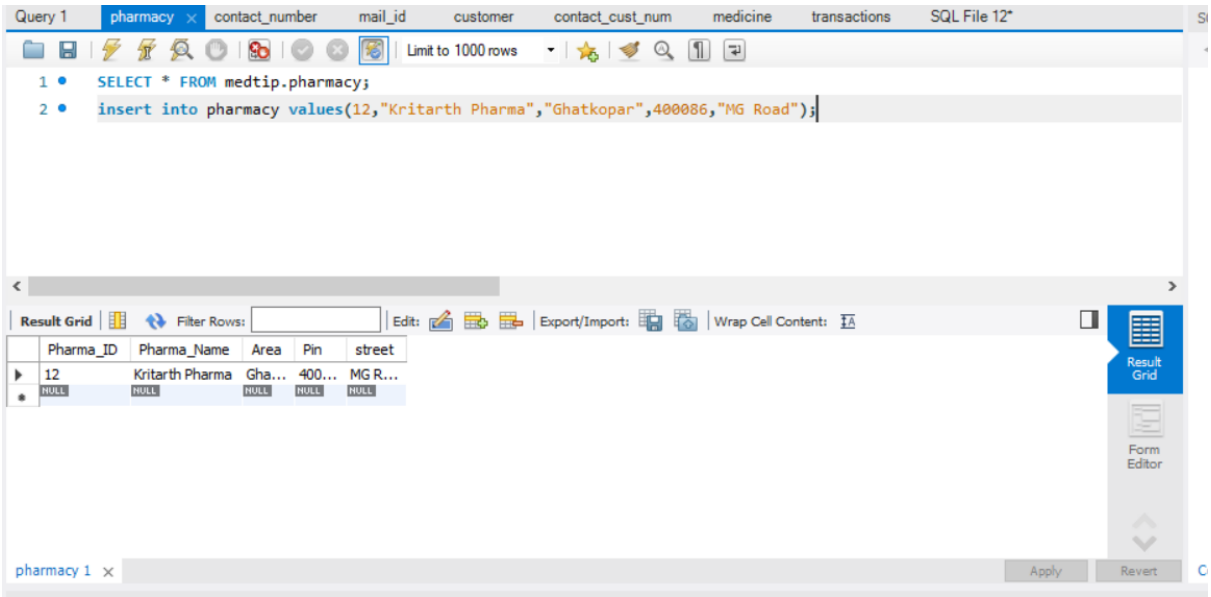
```
Select Pharma_ID,Pharma_Name from pharmacy;
```



Q) insert values into pharmacy table.

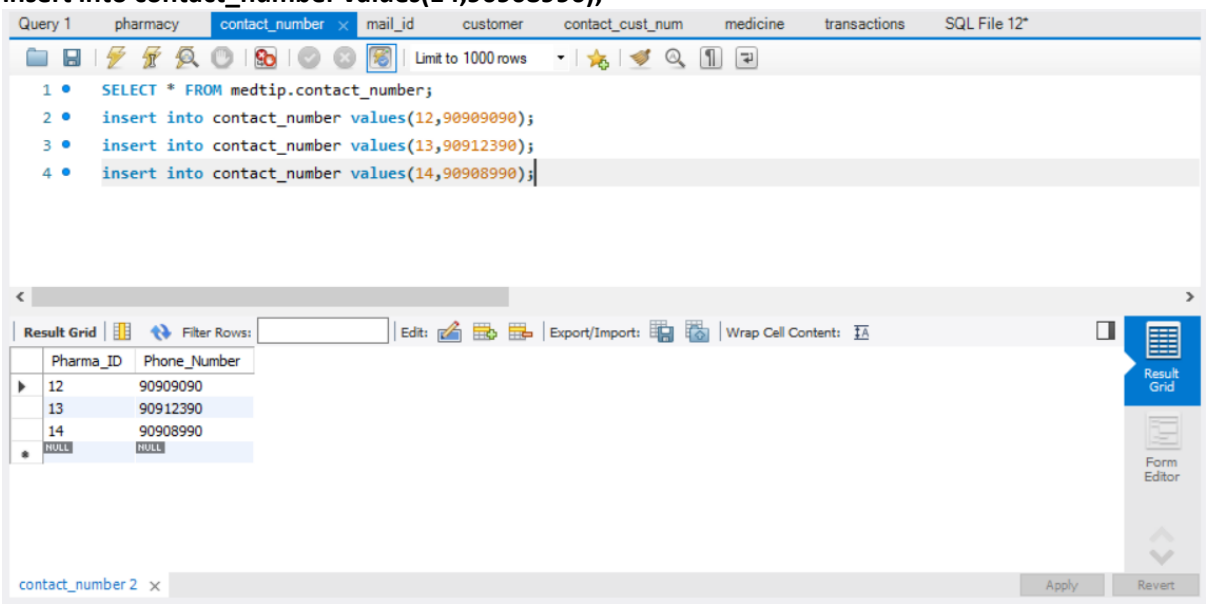
Insert statement : -

insert into pharmacy values(12,"Kritarth Pharma","Ghatkopar",400086,"MG Road");



Q) insert values into customer table

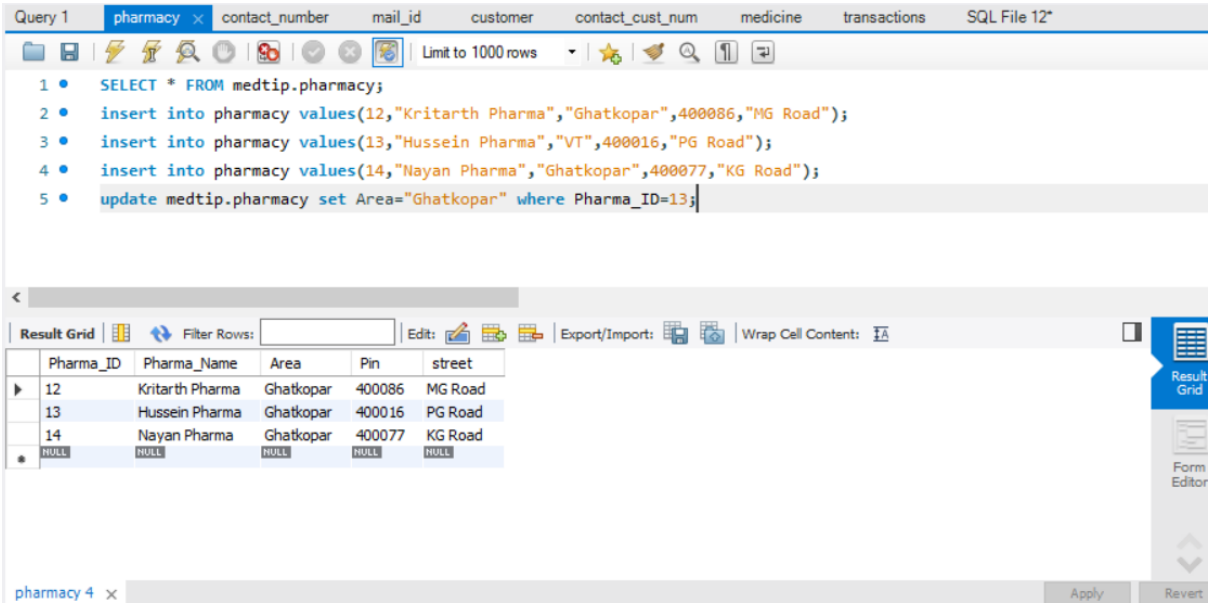
insert into contact_number values(12,90909090);
insert into contact_number values(13,90912390);
insert into contact_number values(14,90908990);



Q) update the pharmacy area to Ghatkopar where pharmacy id is equal to 13.

Update statement : -

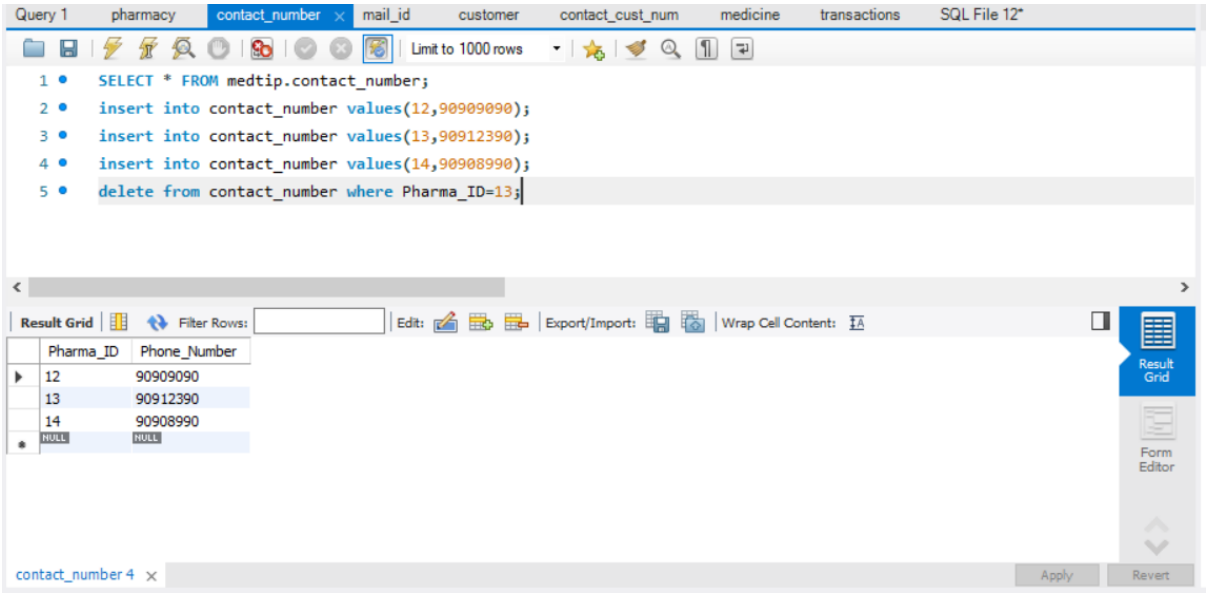
update medtip.pharmacy set Area="Ghatkopar" where Pharma_ID=13;



Q) Delete the contact number of pharmacy where pharma id is equal to 13.

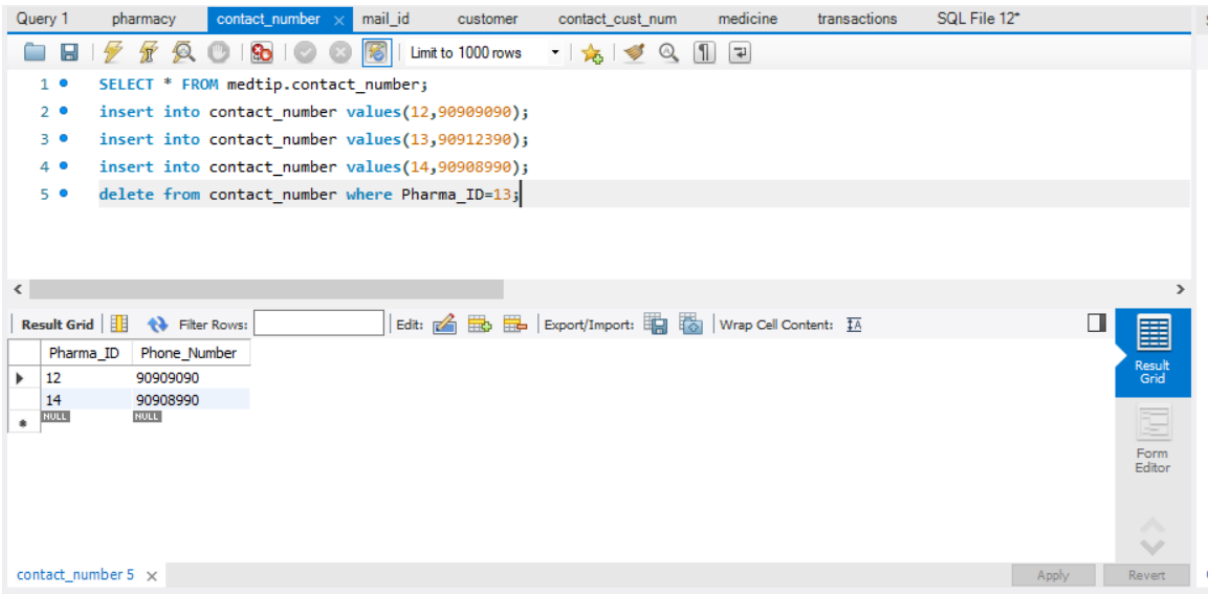
Delete Statement : -

Before Delete : -



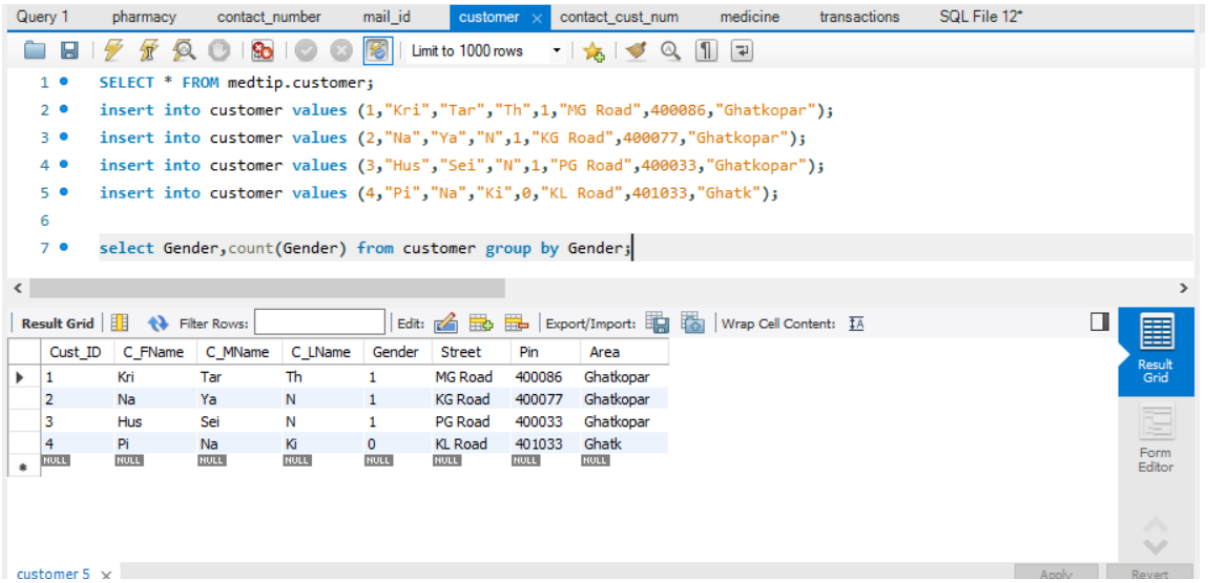
After delete : -

delete from contact_number where Pharma_ID=13;

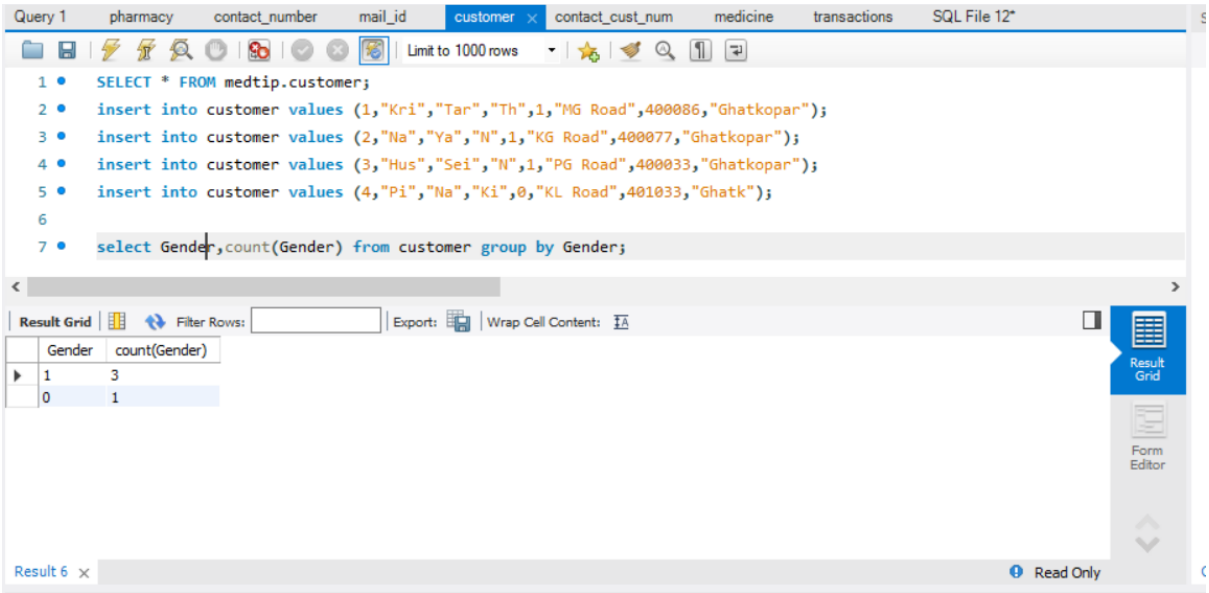


Group by clause : -

Q) print gender and count of individual gender from customer table.

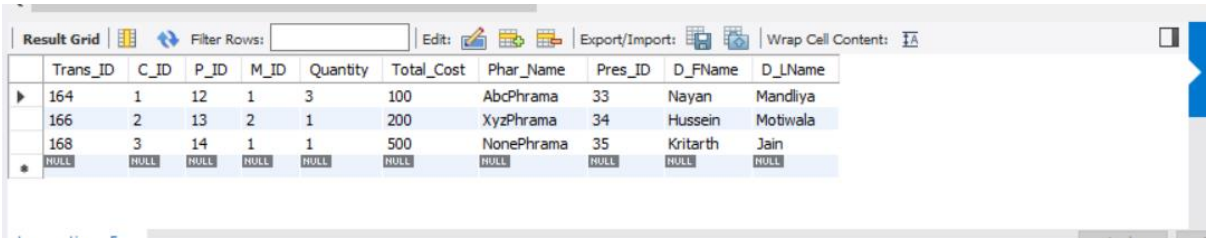


select Gender,count(Gender) from customer group by Gender;



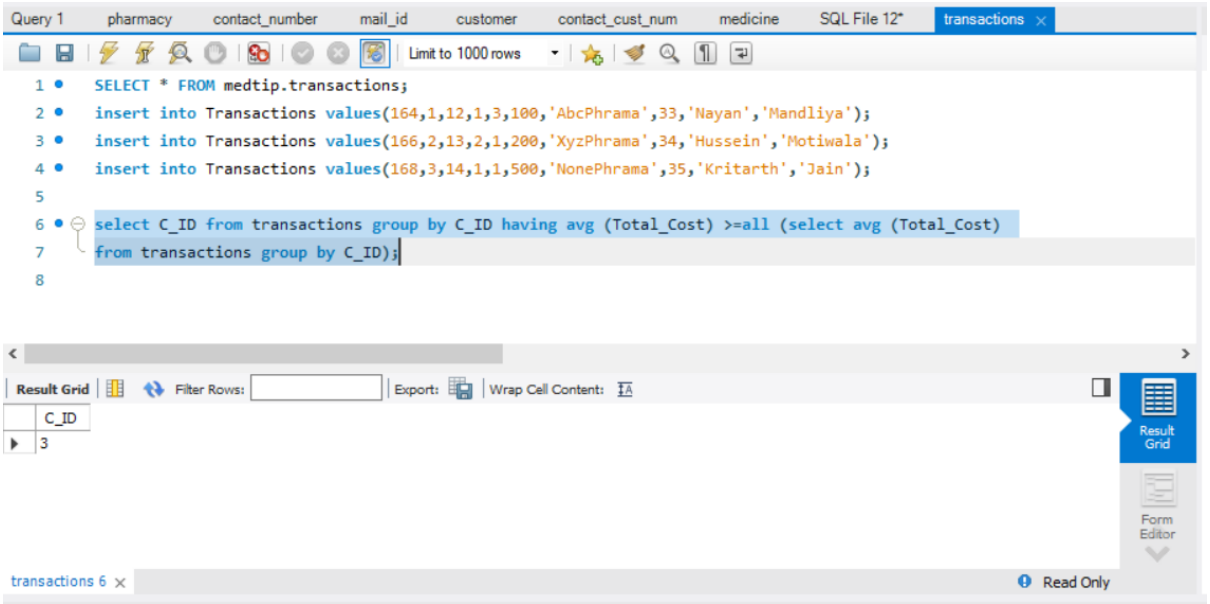
Having clause : -

Q) select all the customer id's having total cost of transaction greater than average cost .



Trans_ID	C_ID	P_ID	M_ID	Quantity	Total_Cost	Phar_Name	Pres_ID	D_FName	D_LName
164	1	12	1	3	100	AbcPhrama	33	Nayan	Mandliya
166	2	13	2	1	200	XyzPhrama	34	Hussein	Motiwala
168	3	14	1	1	500	NonePhrama	35	Kritarth	Jain
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

select C_ID from transactions group by C_ID having avg (Total_Cost) >=all (select avg (Total_Cost) from transactions group by C_ID);

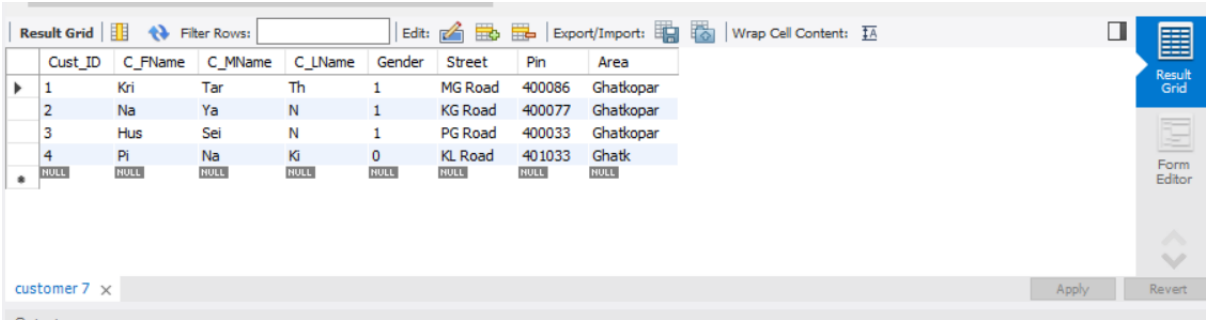


```
1  SELECT * FROM medtip.transactions;
2  insert into Transactions values(164,1,12,1,3,100,'AbcPhrama',33,'Nayan','Mandliya');
3  insert into Transactions values(166,2,13,2,1,200,'XyzPhrama',34,'Hussein','Motiwala');
4  insert into Transactions values(168,3,14,1,1,500,'NonePhrama',35,'Kritarth','Jain');
5
6  select C_ID from transactions group by C_ID having avg (Total_Cost) >=all (select avg (Total_Cost)
7  from transactions group by C_ID);
8
```

C_ID
3

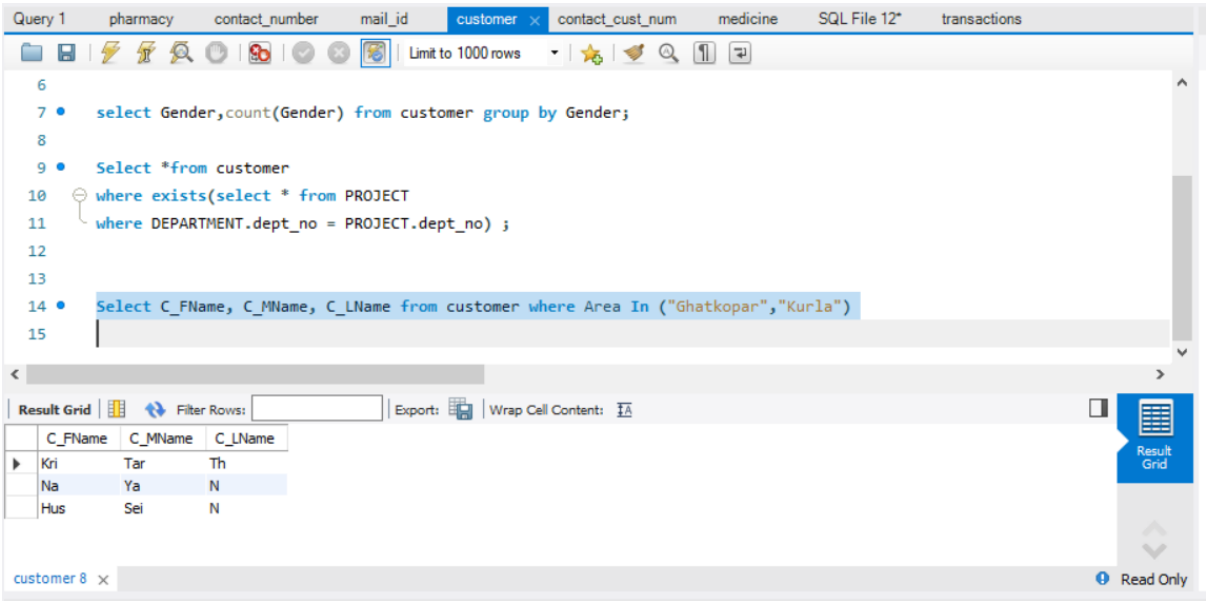
In clause : -

Q) select customers first, middle and last name from customers table where the customers area is either Ghatkopar or kurla.



Cust_ID	C_FName	C_MName	C_LName	Gender	Street	Pin	Area
1	Kri	Tar	Th	1	MG Road	400086	Ghatkopar
2	Na	Ya	N	1	KG Road	400077	Ghatkopar
3	Hus	Sei	N	1	PG Road	400033	Ghatkopar
4	Pi	Na	Ki	0	KL Road	401033	Ghatk
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Select C_FName, C_MName, C_LName from customer where Area In ("Ghatkopar","Kurla")

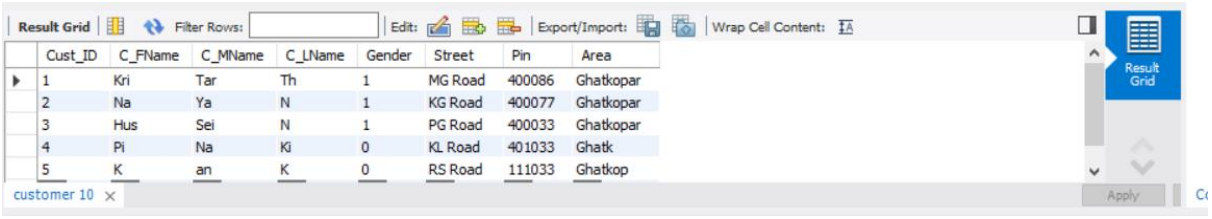


```
6
7  select Gender,count(Gender) from customer group by Gender;
8
9  Select *from customer
10 where exists(select * from PROJECT
11 where DEPARTMENT.dept_no = PROJECT.dept_no) ;
12
13
14 Select C_FName, C_MName, C_LName from customer where Area In ("Ghatkopar","Kurla")
15
```

C_FName	C_MName	C_LName
Kri	Tar	Th
Na	Ya	N
Hus	Sei	N

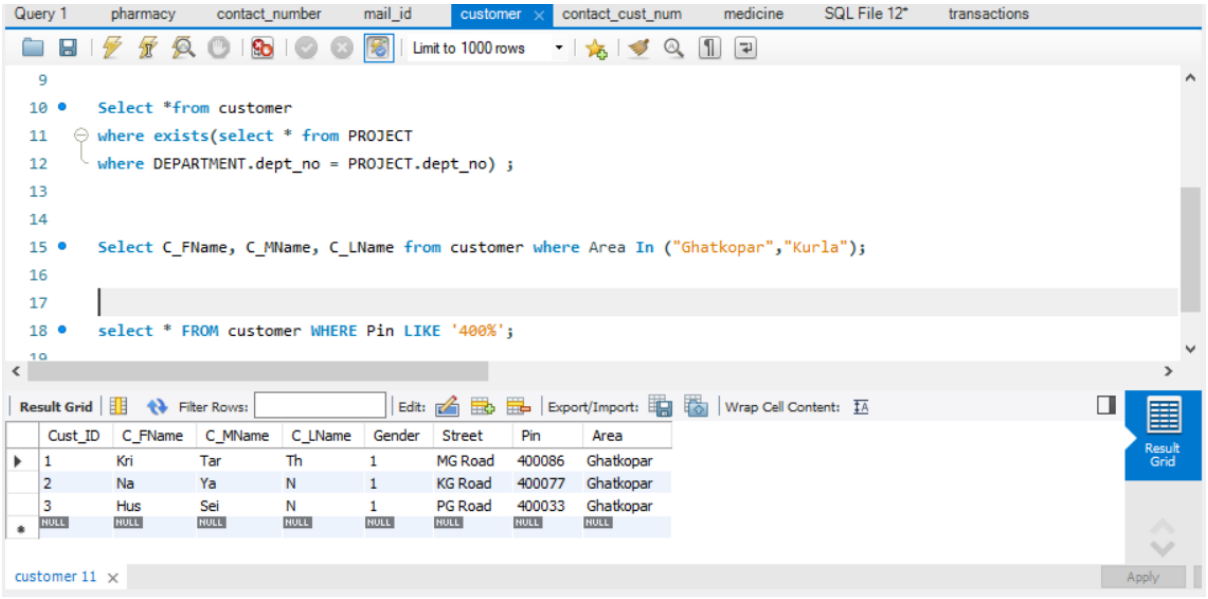
Like clause : -

Q) select all the customers having pin number staring from 400.



Cust_ID	C_FName	C_MName	C_LName	Gender	Street	Pin	Area
1	Kri	Tar	Th	1	MG Road	400086	Ghatkopar
2	Na	Ya	N	1	KG Road	400077	Ghatkopar
3	Hus	Sei	N	1	PG Road	400033	Ghatkopar
4	Pi	Na	Ki	0	KL Road	401033	Ghatk
5	K	an	K	0	RS Road	111033	Ghatkop

select * FROM customer WHERE Pin LIKE '400%';

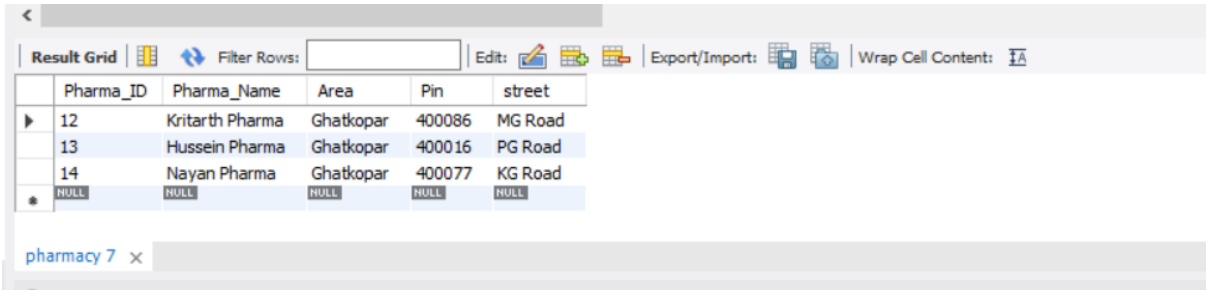


```
9
10  Select *from customer
11  where exists(select * from PROJECT
12  where DEPARTMENT.dept_no = PROJECT.dept_no) ;
13
14
15  Select C_FName, C_MName, C_LName from customer where Area In ("Ghatkopar","Kurla");
16
17
18  select * FROM customer WHERE Pin LIKE '400%';
```

Cust_ID	C_FName	C_MName	C_LName	Gender	Street	Pin	Area
1	Kri	Tar	Th	1	MG Road	400086	Ghatkopar
2	Na	Ya	N	1	KG Road	400077	Ghatkopar
3	Hus	Sei	N	1	PG Road	400033	Ghatkopar

Alias : -

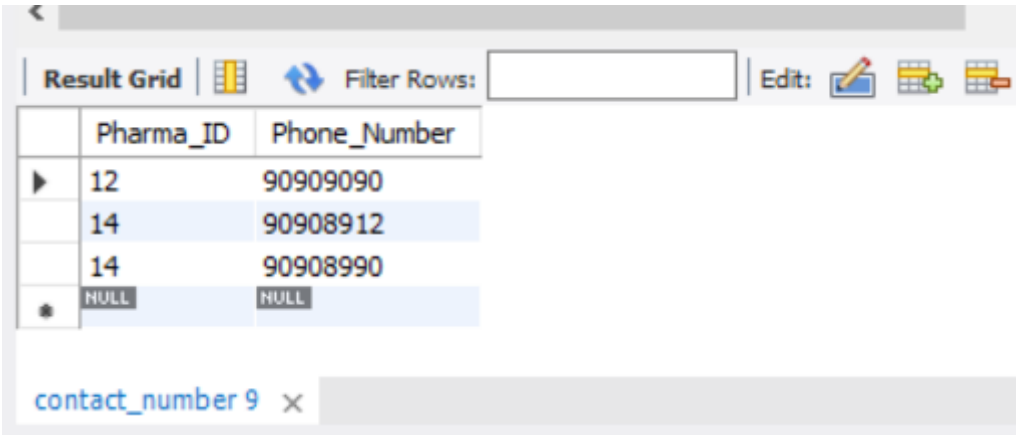
Q) select pharmacy id and pharmacy name from pharmacy table where pharmacy and contact number use alias where pharmacy id = contact number id.



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Pharma_ID	Pharma_Name	Area	Pin	street
12	Kritarth Pharma	Ghatkopar	400086	MG Road
13	Hussein Pharma	Ghatkopar	400016	PG Road
14	Nayan Pharma	Ghatkopar	400077	KG Road
NULL	NULL	NULL	NULL	NULL

pharmacy 7

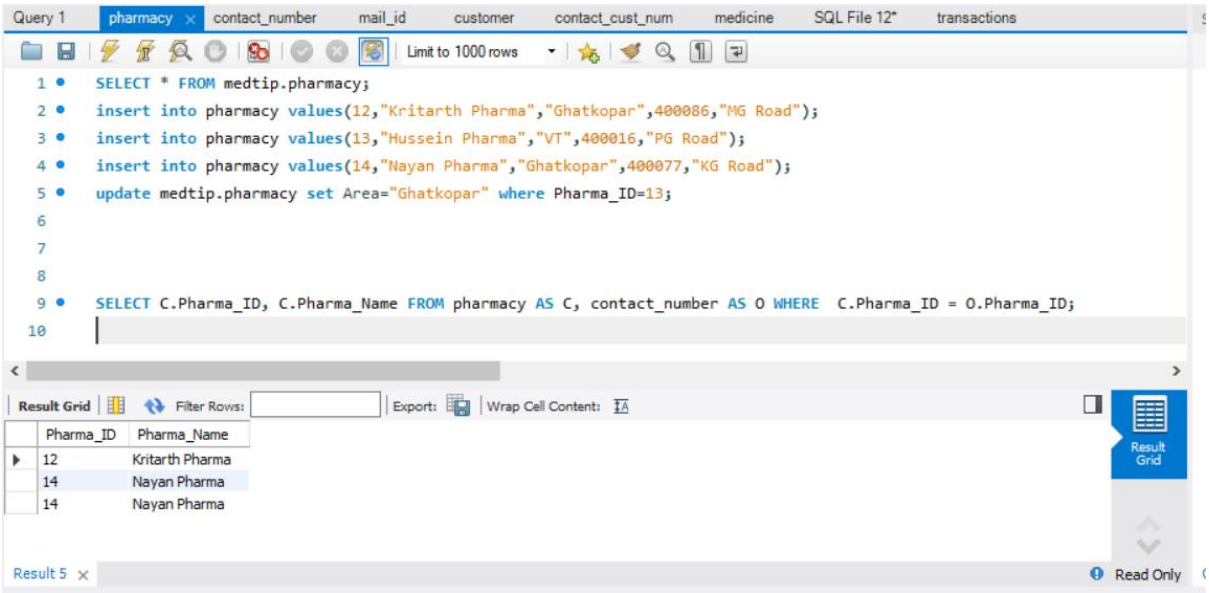


Result Grid | Filter Rows: | Edit:

Pharma_ID	Phone_Number
12	90909090
14	90908912
14	90908990
NULL	NULL

contact_number 9

SELECT C.Pharma_ID, C.Pharma_Name FROM pharmacy AS C, contact_number AS O WHERE C.Pharma_ID = O.Pharma_ID;



Query 1 | pharmacy | contact_number | mail_id | customer | contact_cust_num | medicine | SQL File 12* | transactions

Limit to 1000 rows

```
1  SELECT * FROM medtip.pharmacy;
2  insert into pharmacy values(12,"Kritarth Pharma","Ghatkopar",400086,"MG Road");
3  insert into pharmacy values(13,"Hussein Pharma","VT",400016,"PG Road");
4  insert into pharmacy values(14,"Nayan Pharma","Ghatkopar",400077,"KG Road");
5  update medtip.pharmacy set Area="Ghatkopar" where Pharma_ID=13;
6
7
8
9  SELECT C.Pharma_ID, C.Pharma_Name FROM pharmacy AS C, contact_number AS O WHERE  C.Pharma_ID = O.Pharma_ID;
10
```

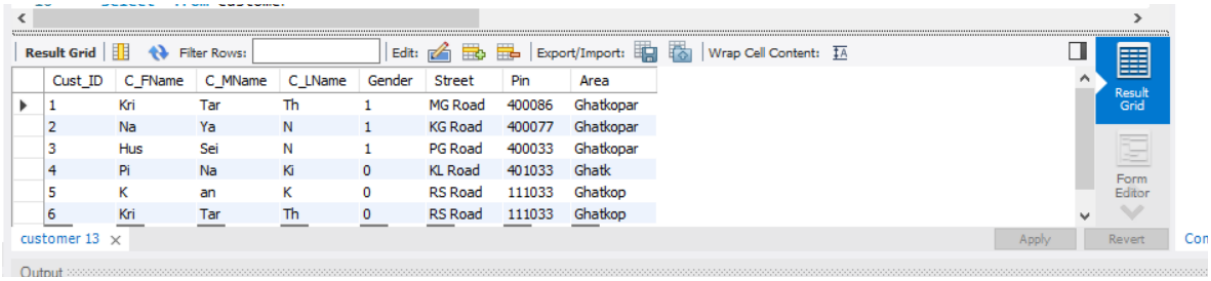
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Pharma_ID	Pharma_Name
12	Kritarth Pharma
14	Nayan Pharma
14	Nayan Pharma

Result Grid | Result 5 | Read Only

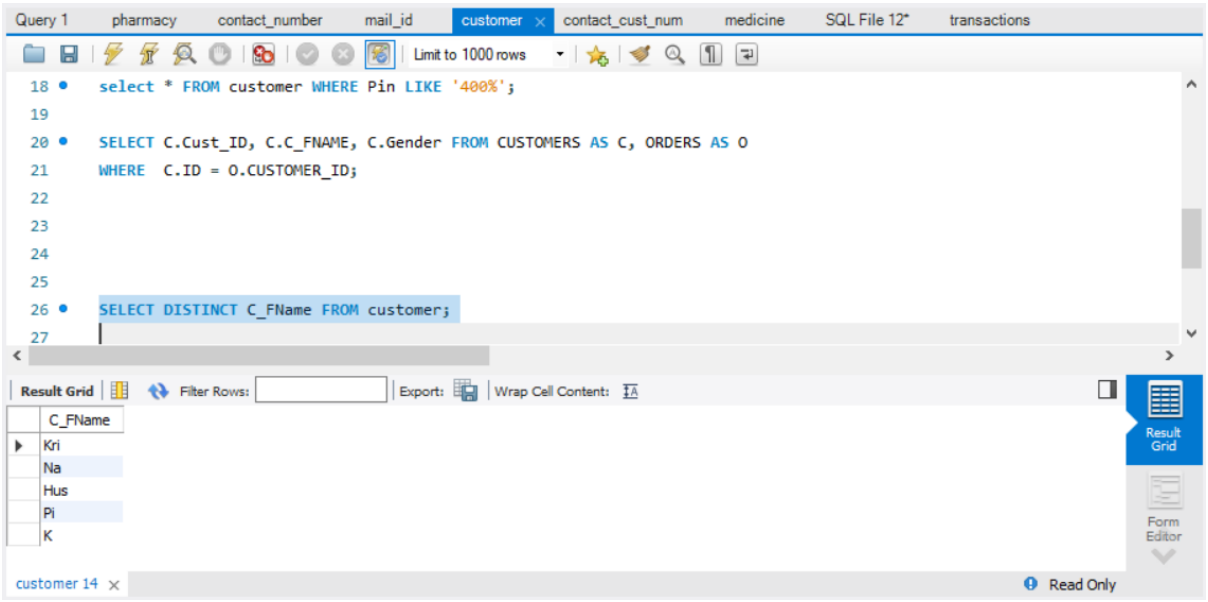
Distinct : -

Q) select all distinct customer first names from customer table.



Cust_ID	C_FName	C_MName	C_LName	Gender	Street	Pin	Area
1	Kri	Tar	Th	1	MG Road	400086	Ghatkopar
2	Na	Ya	N	1	KG Road	400077	Ghatkopar
3	Hus	Sei	N	1	PG Road	400033	Ghatkopar
4	Pi	Na	Ki	0	KL Road	401033	Ghatk
5	K	an	K	0	RS Road	111033	Ghatkop
6	Kri	Tar	Th	0	RS Road	111033	Ghatkop

SELECT DISTINCT C_FName FROM customer;



```
18  select * FROM customer WHERE Pin LIKE '400%';
19
20  SELECT C.Cust_ID, C.C_FNAME, C.Gender FROM CUSTOMERS AS C, ORDERS AS O
21  WHERE  C.ID = O.CUSTOMER_ID;
22
23
24
25
26  SELECT DISTINCT C_FName FROM customer;
27
```

C_FName
Kri
Na
Hus
Pi
K

Conclusion: By performing this experiment we successfully understood various data manipulation language operations like insert, delete, update etc. Also executed some nested queries. Defined some questions for our project and solved that questions using DML operations.

Post Lab Questions

1. **In SQL, which of the following is not a data Manipulation Language Commands?**

   a) Delete
   b) Truncate
   c) Update
   d) Create
   ANS) d) Create

2. **Write SQL query for following statements:**

   a. retrieve all student who his grade has not been awarded
   ANS) select * from students where grade = "not awarded"

   b. Find the names of all instructors in the Computer Science department
   ANS) select name from instructor where d_ID in (select d_id from dep where dep_name = " Computer Science ");

   c. Find the names of all student whose name starts with 'S'.
   ANS) select name from student where name like 'S%'

   d. find the names of instructors with salary amounts between $90,000 and $100,000.

   ANS) select name from instructor where salary between 90000 AND 100000

e. Find all student sorted by their department name , if there are two student have the same department name , then sort them bytotal credit in ascending order, then by their “student name” alias.

ANS) select * from student order by d_name,credits,s_name;

```
19  create table student (
20  s_name varchar(30),
21  d_name varchar(30),
22  credits int
23  );
24  truncate student;
25  insert into student values("hussein", "CSE", 22);
26  insert into student values("Nayan", "BSE", 500);
27  insert into student values("ansh", "CSE", 12);
28  insert into student values("Kritharth", "CSE", 12);
29
30  select * from student order by d_name, credits, s_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

s_name	d_name	credits
Nayan	BSE	500
ansh	CSE	12
Kritharth	CSE	12
hussein	CSE	22