EXPERIMENT - 1

AIM : Create Table and Insert Data into them.

THEORY :

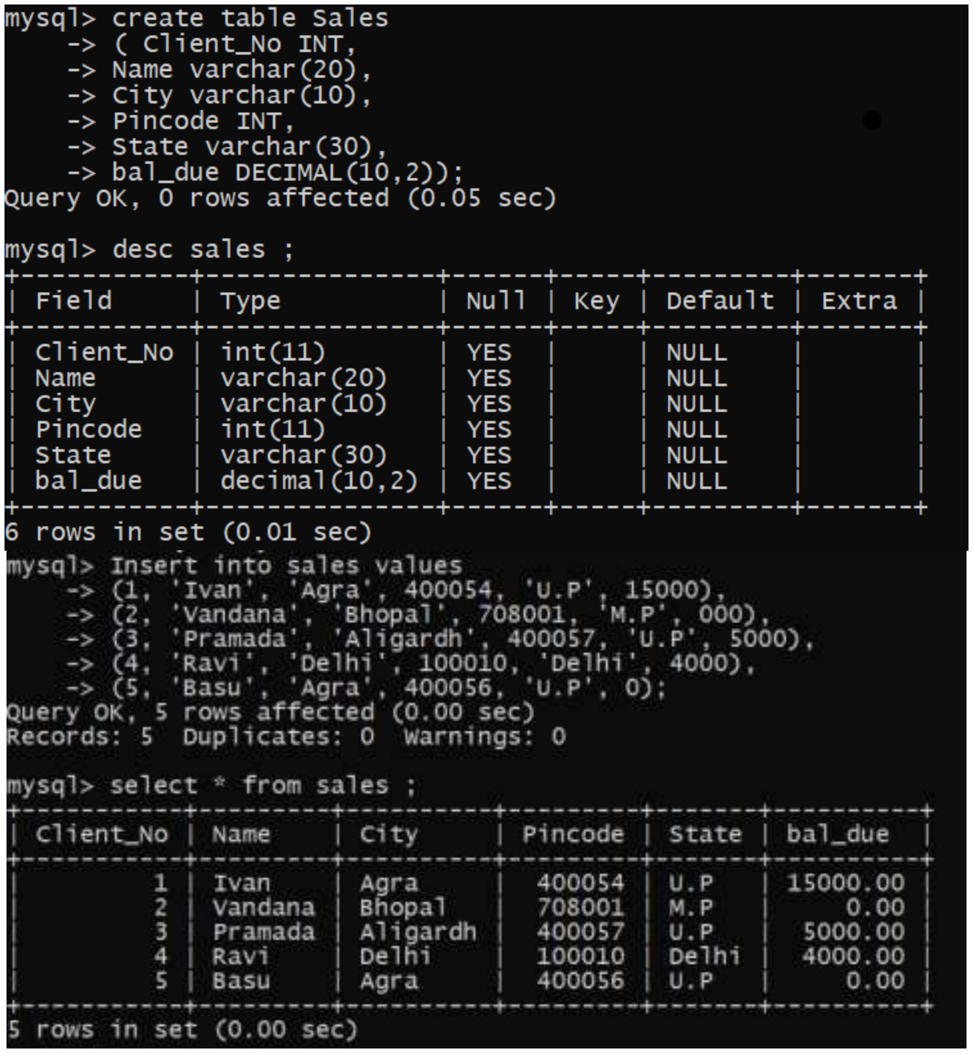
1) Statement for creating a table

CREATE TABLE tablename
(column_name datatype(size), …) ;

2) Statement for inserting data into table

INSERT INTO tablename
Values(expression, expression, …) ;

QUERY :



```
mysql> create table Sales
    -> ( Client_No INT,
    -> Name varchar(20),
    -> City varchar(10),
    -> Pincode INT,
    -> State varchar(30),
    -> bal_due DECIMAL(10,2));
Query OK, 0 rows affected (0.05 sec)

mysql> desc sales ;
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| Client_No | int(11)       | YES  |     | NULL    |       |
| Name      | varchar(20)   | YES  |     | NULL    |       |
| City      | varchar(10)   | YES  |     | NULL    |       |
| Pincode   | int(11)       | YES  |     | NULL    |       |
| State     | varchar(30)   | YES  |     | NULL    |       |
| bal_due   | decimal(10,2) | YES  |     | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
mysql> Insert into sales values
    -> (1, 'Ivan', 'Agra', 400054, 'U.P', 15000),
    -> (2, 'Vandana', 'Bhopal', 708001, 'M.P', 000),
    -> (3, 'Pramada', 'Aligardh', 400057, 'U.P', 5000),
    -> (4, 'Ravi', 'Delhi', 100010, 'Delhi', 4000),
    -> (5, 'Basu', 'Agra', 400056, 'U.P', 0);
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from sales ;
+-----------+---------+----------+---------+-------+----------+
| Client_No | Name    | City     | Pincode | State | bal_due  |
+-----------+---------+----------+---------+-------+----------+
|         1 | Ivan    | Agra     |  400054 | U.P   | 15000.00 |
|         2 | Vandana | Bhopal   |  708001 | M.P   |     0.00 |
|         3 | Pramada | Aligardh |  400057 | U.P   |  5000.00 |
|         4 | Ravi    | Delhi    |  100010 | Delhi |  4000.00 |
|         5 | Basu    | Agra     |  400056 | U.P   |     0.00 |
+-----------+---------+----------+---------+-------+----------+
5 rows in set (0.00 sec)
```

EXPERIMENT - 2

AIM : Use SELECT command for retrieving data from table.

THEORY :

1) Statement for selecting specific column from table

```
SELECT columnname, columnname
FROM tablename ;
```

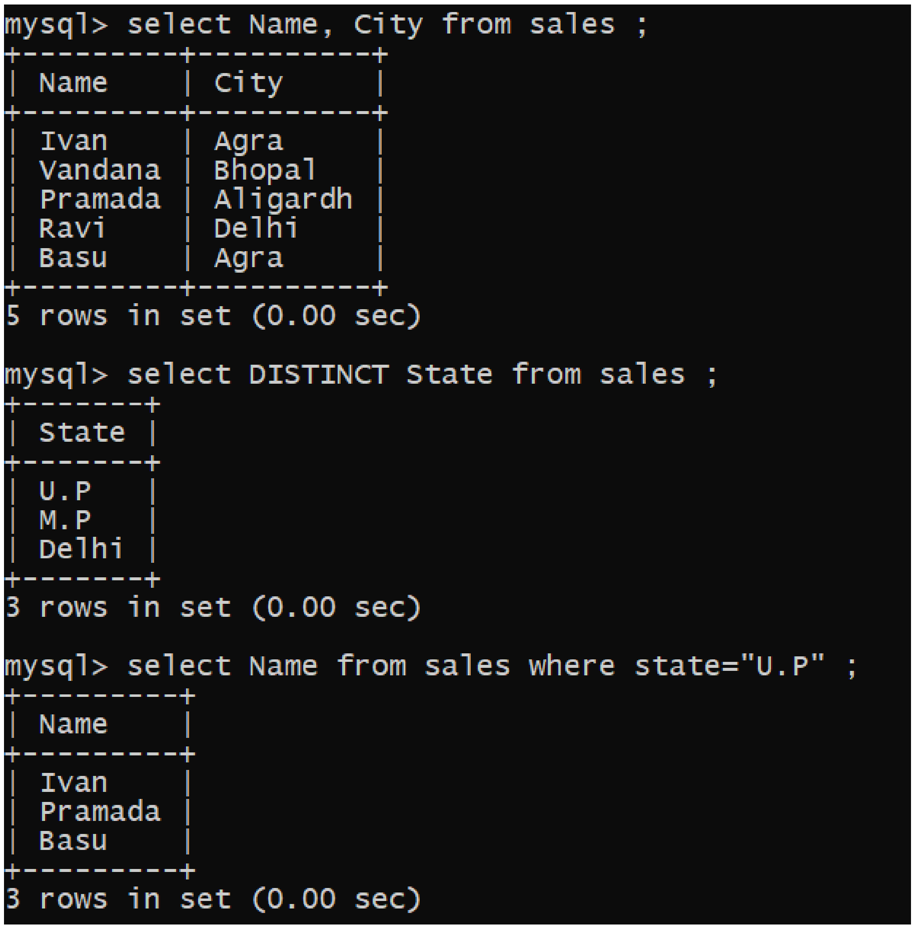
2) Statement for elimination of duplicte data from table

```
SELECT DISTINCT columnname, columnname
FROM tablename ;
```

3) Statement for selecting specific data set from table

```
SELECT columnname, columnname
FROM tablename
WHERE condition ;
```

QUERY :



```
mysql> select Name, City from sales ;
+---------+----------+
| Name    | City     |
+---------+----------+
| Ivan    | Agra     |
| Vandana | Bhopal   |
| Pramada | Aligardh |
| Ravi    | Delhi    |
| Basu    | Agra     |
+---------+----------+
5 rows in set (0.00 sec)

mysql> select DISTINCT State from sales ;
+-------+
| State |
+-------+
| U.P   |
| M.P   |
| Delhi |
+-------+
3 rows in set (0.00 sec)

mysql> select Name from sales where state="U.P" ;
+---------+
| Name    |
+---------+
| Ivan    |
| Pramada |
| Basu    |
+---------+
3 rows in set (0.00 sec)
```

EXPERIMENT - 3

AIM : a) Use DML commands UPDATE and DELETE.

THEORY :

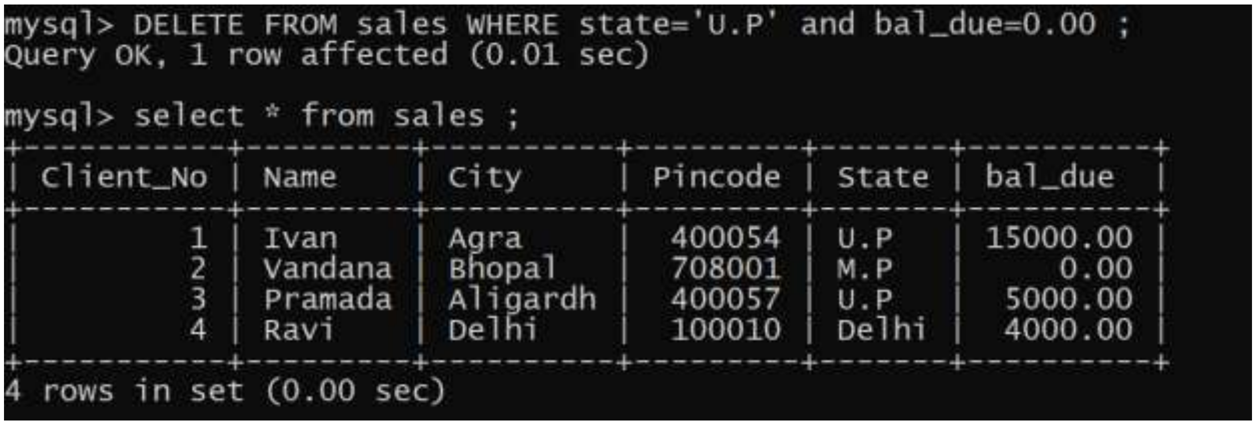
1) Statement to delete records from a table in SQL

```
DELETE FROM TableName
WHERE condition;
```

2) Statement to update records in a table in SQL

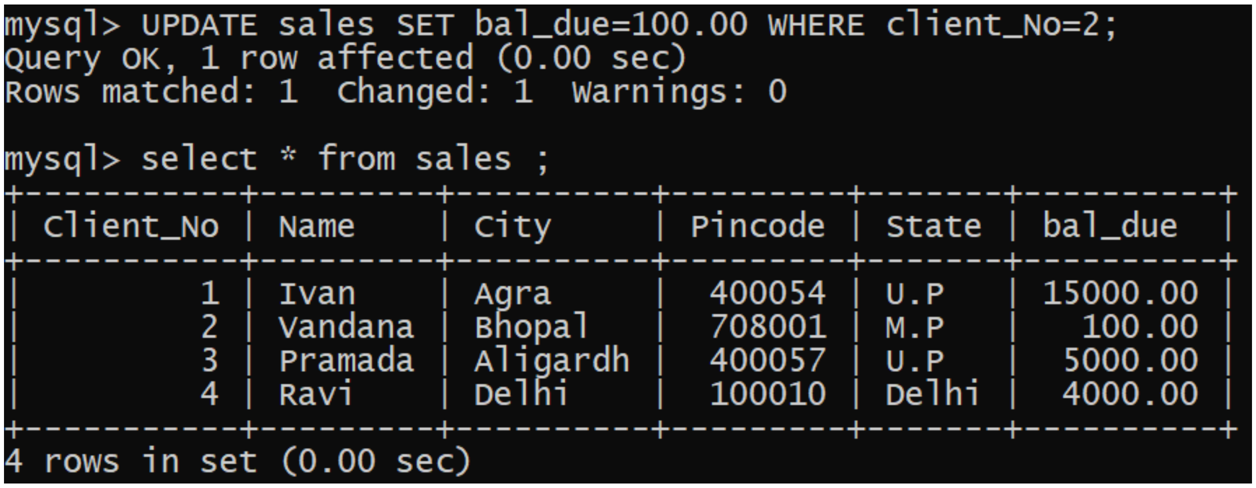
```
UPDATE TableName SET column1 = value1, column2 = value2
WHERE condition;
```

QUERY :



```
mysql> DELETE FROM sales WHERE state='U.P' and bal_due=0.00 ;
Query OK, 1 row affected (0.01 sec)

mysql> select * from sales ;
+-----------+---------+----------+---------+-------+----------+
| Client_No | Name    | City     | Pincode | State | bal_due  |
+-----------+---------+----------+---------+-------+----------+
|         1 | Ivan    | Agra     |  400054 | U.P   | 15000.00 |
|         2 | Vandana | Bhopal   |  708001 | M.P   |     0.00 |
|         3 | Pramada | Aligardh |  400057 | U.P   |  5000.00 |
|         4 | Ravi    | Delhi    |  100010 | Delhi |  4000.00 |
+-----------+---------+----------+---------+-------+----------+
4 rows in set (0.00 sec)
```



```
mysql> UPDATE sales SET bal_due=100.00 WHERE client_No=2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from sales ;
+-----------+---------+----------+---------+-------+----------+
| Client_No | Name    | City     | Pincode | State | bal_due  |
+-----------+---------+----------+---------+-------+----------+
|         1 | Ivan    | Agra     |  400054 | U.P   | 15000.00 |
|         2 | Vandana | Bhopal   |  708001 | M.P   |   100.00 |
|         3 | Pramada | Aligardh |  400057 | U.P   |  5000.00 |
|         4 | Ravi    | Delhi    |  100010 | Delhi |  4000.00 |
+-----------+---------+----------+---------+-------+----------+
4 rows in set (0.00 sec)
```

AIM : b) Use Transaction Control Statements COMMIT, ROLLBACK and SAVEPOINT .

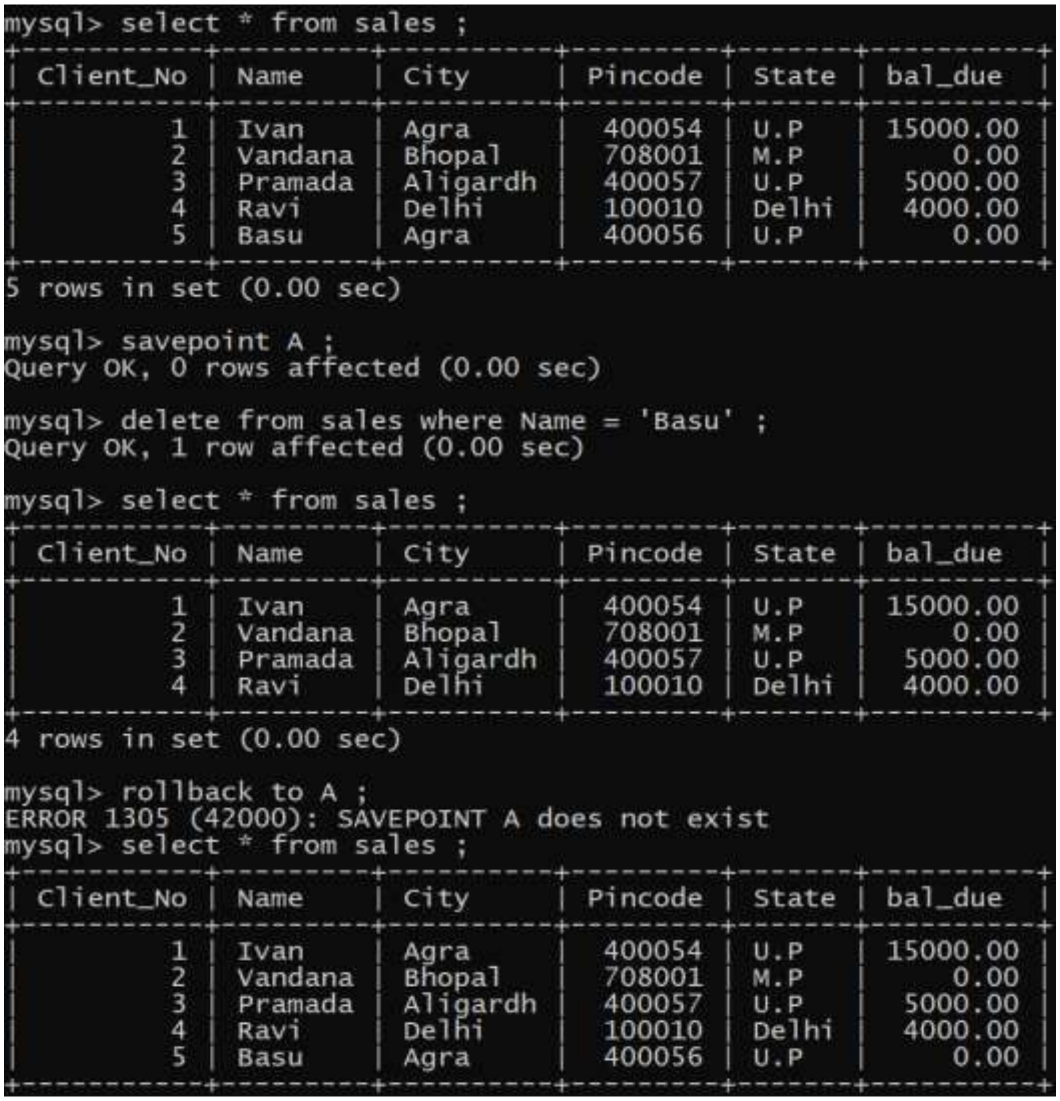
THEORY :

1) COMMIT - The COMMIT command is used to permanently save any changes made during the current transaction.
Syntax: COMMIT;

2) SAVEPOINT - A SAVEPOINT is a point within a transaction to which you can later roll back. It allows you to mark a point in a transaction so that you can later roll back to that point if needed.
Syntax: SAVEPOINT savepoint_name;

3) ROLLBACK - The ROLLBACK command is used to undo the changes made during the current transaction.
Syntax: ROLLBACK TO SAVEPOINT savepoint_name;

QUERY :



```
mysql> select * from sales ;
+-----------+---------+----------+---------+-------+----------+
| Client_No | Name    | City     | Pincode | State | bal_due  |
+-----------+---------+----------+---------+-------+----------+
|         1 | Ivan    | Agra     |  400054 | U.P   | 15000.00 |
|         2 | Vandana | Bhopal   |  708001 | M.P   |     0.00 |
|         3 | Pramada | Aligardh |  400057 | U.P   |  5000.00 |
|         4 | Ravi    | Delhi    |  100010 | Delhi |  4000.00 |
|         5 | Basu    | Agra     |  400056 | U.P   |     0.00 |
+-----------+---------+----------+---------+-------+----------+
5 rows in set (0.00 sec)

mysql> savepoint A ;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from sales where Name = 'Basu' ;
Query OK, 1 row affected (0.00 sec)

mysql> select * from sales ;
+-----------+---------+----------+---------+-------+----------+
| Client_No | Name    | City     | Pincode | State | bal_due  |
+-----------+---------+----------+---------+-------+----------+
|         1 | Ivan    | Agra     |  400054 | U.P   | 15000.00 |
|         2 | Vandana | Bhopal   |  708001 | M.P   |     0.00 |
|         3 | Pramada | Aligardh |  400057 | U.P   |  5000.00 |
|         4 | Ravi    | Delhi    |  100010 | Delhi |  4000.00 |
+-----------+---------+----------+---------+-------+----------+
4 rows in set (0.00 sec)

mysql> rollback to A ;
ERROR 1305 (42000): SAVEPOINT A does not exist
mysql> select * from sales ;
+-----------+---------+----------+---------+-------+----------+
| Client_No | Name    | City     | Pincode | State | bal_due  |
+-----------+---------+----------+---------+-------+----------+
|         1 | Ivan    | Agra     |  400054 | U.P   | 15000.00 |
|         2 | Vandana | Bhopal   |  708001 | M.P   |     0.00 |
|         3 | Pramada | Aligardh |  400057 | U.P   |  5000.00 |
|         4 | Ravi    | Delhi    |  100010 | Delhi |  4000.00 |
|         5 | Basu    | Agra     |  400056 | U.P   |     0.00 |
+-----------+---------+----------+---------+-------+----------+
```

EXPERIMENT - 4

AIM : To understand and implement Integrity Constraints using full DDL commands.

1. Create table
2. Alter table

THEORY :

Primary Key: The PRIMARY KEY constraint uniquely identifies each record in a database table. Primary keys must contain UNIQUE values, and cannot contain NULL values.
A table can have only one primary key, which may consist of single or multiple fields.
Foreign Key: field in one table that refers to primary key of another table (REFERENTIAL INTEGRITY CONSTRAINT).A FOREIGN KEY is a key used to link two tables together.
The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.
Default:

- provide default value
- will be added to all new records if no other value is specified

The DEFAULT constraint is used to provide a default value for a column.
The default value will be added to all new records IF no other value is specified.
Not Null: By default, a column can hold NULL values.
The NOT NULL constraint enforces a column to NOT accept NULL values.
This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.
Check:
-implement all business rule
- used to limit the range value
The CHECK constraint is used to limit the value range that can be placed in a column.
If you define a CHECK constraint on a single column it allows only certain values for this column. If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.
Unique: The UNIQUE constraint ensures that all values in a column are different.
Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
A PRIMARY KEY constraint automatically has a UNIQUE constraint.
Table Level Constraints: composite primary key
Column Level Constraints: one primary key

QUERY :

#Sales_Master

```
mysql> create table Sales_Master(salesman_no varchar(6) primary key
check(salesman_no like's%'),sal_name varchar(20) not null, address varchar(20) not
null,city varchar(20), pincode numeric(6),sal_amt numeric(8,2) not null check(sal_amt
>0),tgt_to_get numeric(6,2) not null check(tgt_to_get >0),ytd_sales numeric(6,2) not
null check(ytd_sales >0),remarks varchar(30));
Query OK, 0 rows affected (0.13 sec)
```

#Sales_Master

```
mysql> select * from Sales_Master;
Empty set (0.00 sec)
```

#Sales_Order

```
mysql> create table Sales_Order(S_order_no varchar(6) primary key check(S_order_no
like "O%"), S_order_date Date,Client_no varchar(6), Salesman_no varchar(6)references
Sales_Master(Salesman_no),Dely_type char(1) check(Dely_type in ('f','p')), Billed_yn
char(1), Dely_date date check(Dely_date>=S_order_date),Order_status varchar(10)
check(Order_status in('inprocess','fulfilled','backorder','canceled')));
Query OK, 0 rows affected (0.06 sec)
```

#Sales_Master

```
mysql> desc Sales_Master;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| salesman_no | varchar(6) | NO | PRI | NULL |   |
| sal_name | varchar(20) | NO |   | NULL |   |
| address | varchar(20) | NO |   | NULL |   |
| city | varchar(20) | YES |   | NULL |   |
| pincode | decimal(6,0) | YES |   | NULL |   |
| sal_amt | decimal(8,2) | NO |   | NULL |   |
| tgt_to_get | decimal(6,2) | NO |   | NULL |   |
| ytd_sales | decimal(6,2) | NO |   | NULL |   |
| remarks | varchar(30) | YES |   | NULL |   |
+-------------+-------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

#Sales_Order

```
mysql> desc Sales_Order;
+--------------+------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------------+------------+------+-----+---------+-------+
| S_order_no | varchar(6) | NO | PRI | NULL |   |
| S_order_date | date | YES |   | NULL |   |
| Client_no | varchar(6) | YES |   | NULL |   |
| Salesman_no | varchar(6) | YES |   | NULL |   |
| Dely_type | char(1) | YES |   | NULL |   |
| Billed_yn | char(1) | YES |   | NULL |   |
| Dely_date | date | YES |   | NULL |   |
| Order_status | varchar(10) | YES |   | NULL |   |
+--------------+------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

#Sales_Order_Details

```
mysql> create table Sales_Order_Details(S_order_no varchar(6) references
Sales_order(S_order_no),Product_no varchar(6) primary key, Qty_order
numeric(8),Qty_disp numeric(8), Product_rate numeric(10,2));
Query OK, 0 rows affected (0.08 sec)
```

#Sales_Order_Details

```
mysql> desc Sales_Order_Details;
+--------------+--------------+------+-----+--------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------------+--------------+------+-----+--------+-------+
| S_order_no | varchar(6) | YES | | NULL | |
| Product_no | varchar(6) | NO | PRI | NULL | |
| Qty_order | decimal(8,0) | YES | | NULL | |
| Qty_disp | decimal(8,0) | YES | | NULL | |
| Product_rate | decimal(10,2) | YES | | NULL | |
+--------------+--------------+------+-----+--------+-------+
5 rows in set (0.00 sec)
```

Query: Make the primary key to client_no in client_master

```
mysql> alter table client_master add constraint pk primary key(client_no);
```

Query: Add foreign key constraint in Sales_Order:- Client_no foreign key references client_no of client_master

```
mysql> alter table Sales_Order add constraint fk foreign key(Client_no) references
client_master(client_no);
```

#product_master

```
mysql> desc product_master;
+----------------+-------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| product_no | varchar(7) | NO | PRI | | |
| Description | varchar(15) | YES | | NULL | |
| profile_percent | decimal(2,0) | YES | | NULL | |
| unit_measure | varchar(7) | YES | | NULL | |
| Qty_on_hand | decimal(4,0) | YES | | NULL | |
| Reorder_lvl | decimal(3,0) | YES | | NULL | |
| sell_price | decimal(6,0) | YES | | NULL | |
| cost_price | decimal(6,0) | YES | | NULL | |
+----------------+-------------+------+-----+---------+-------+
```

EXPERIMENT - 5

AIM : To retrieve the data using the concept of.

1. Join
2. Set operators (Union, intersect and minus)

THEORY :

A JOIN clause is used to combine rows from two or more tables, based on related column between them.
Types of the JOINs in SQL:

1. **(INNER) JOIN:** Returns records that have matching values in both tables
2. **LEFT (OUTER) JOIN:** Return all records from the left table, and the matched records from the right table
3. **RIGHT (OUTER) JOIN:** Return all records from the right table, and the matched records from the left table
4. **FULL (OUTER) JOIN:** Return all records when there is a match in either left or right table

SQL supports few Set operations which can be performed on the table data. These are used to get meaningful results from data stored in the table, under different special conditions.

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

SYNTAX/COMMANDS USED:

```
1. SELECT columns FROM table1
LEFT [OUTER] JOIN table2
ON table1.column = table2.column;

2. SELECT columns FROM table1
RIGHT [OUTER] JOIN table2
ON table1.column = table2.column;

3. SELECT columns FROM table1
INNER JOIN table2
ON table1.column = table2.column;

4. SELECT column_name (s) FROM table1 UNION
SELECT column_name (s) FROM table2
```

```
5. SELECT column_list_1 FROM table_1
MINUS
SELECT columns_list_2 FROM table_2;

6. SELECT column_list FROM table_1
INTERSECT
(SELECT column_list FROM table_2)

7. SELECT * FROM First
UNION
SELECT * From Second

8. SELECT * FROM First
INTERSECT
SELECT * FROM Second;

9. SELECT * FROM First
MINUS
SELECT * FROM Sec

10. SELECT EXTRACT(MONTH FROM "2017-06-15");
```

QUERY :

Query: the order number,client name and day of week on which clients placed their order.

```
mysql> select s_order_no,name,extract(day from s_order_date) from sales_order
s,client_master c where s.client_no= c.client_no;
+------------+---------+-------------------------------+
| s_order_no | name | extract(day from s_order_date) |
+------------+---------+-------------------------------+
| 010008 | Ravi | 24 |
| 016865 | Pramada | 18 |
| 019001 | Ivan | 12 |
| 019002 | Vandana | 25 |
| 019003 | Ivan | 3 |
| 046866 | Basu | 20 |
```

Query: Display the month and date when the order must be delivered and the name of the salesman to whom the order was placed.

```
mysql> select sal_name,extract(month from dely_date),dely_date from sales_master
p,sales_order where p.salesman_no=o.salesman_no;
+----------+------------------------------+------------+
| sal_name | extract(month from dely_date) | dely_date |
+----------+------------------------------+------------+
| kiran | 1 | 1996-01-20 |
| kiran | 4 | 1996-04-07 |
| manish | 1 | 1996-01-27 |
| ravi | 1 | 1996-01-20 |
| ashish | 5 | 1996-05-26 |
| ashish | 5 | 1996-05-22 |
```

Query: Find out the names of products that have been sold to "Ivan".

```
mysql> select description from sales_order s,sales_order_details d,product_master
p,client_master c where s.s_order_no=d.s_order_no and d.product_no=p.product_no and
c.client_no=s.client_no and c.name='Ivan';
+--------------+
| description |
+--------------+
| 1.44floppies |
| CD Drive |
| 540 HDD |
| 1.44floppies |
| Monitors |
```

Query: For each sales order display the name of the client and the salesman.

```
mysql> select name,sal_name from client_master c,sales_master s,sales_order t where
s.salesman_no=t.salesman_no and c.client_no=t.client_no;
+---------+----------+
| name | sal_name |
+---------+----------+
| Ivan | kiran |
| Ivan | kiran |
| Vandana | manish |
| Pramada | ravi |
| Ravi | ashish |
| Basu | ashish |
```

Query :Find out the names of clients who have purchased "CD DRIVE".

```
mysql>select name from client_master c,product_master p,sales_order
s,sales_order_details d where c.client_no=s.client_no and s.s_order_no=d.s_order_no and
d.product_no=p.product_no and p.description='CD Drive';
+------+
| name |
+------+
| Ivan |
```

Query: List the product_no and s_order_no of customers who have ordered less than 5 quantity of product "1.44 floppies".

```
mysql> SELECT PRODUCT_MASTER.PRODUCT_NO,
SALES_ORDER_DETAILS.S_ORDER_NO FROM PRODUCT_MASTER,
SALES_ORDER_DETAILS WHERE PRODUCT_MASTER.PRODUCT_NO =
NAME: kushagra arora BRANCH : IT-1/A2 ROLL NO. : 1900910130065
DBMS [KCS-551]
SALES_ORDER_DETAILS.PRODUCT_NO AND DESCRIPTION = "1.44 Floppies" AND
SALES_ORDER_DETAILS.QTY_ORDER <5;
+------------+------------+
| PRODUCT_NO | S_ORDER_NO |
+------------+------------+
| P00001 | 019001 |
| P00001 | 019003 |
```


Query: Display all clients and the salesman in the city of Bombay.

```
mysql>select name ,client_no,salesman_no,sal_name from client_master c.sales_master
where c.city=T.city and c.city='Bombay';
+---------+----------+
| NAME | SAL_NAME |
+---------+----------+
| Ivan | KIRAN |
| Ivan | KIRAN |
| Vandana | MANISH |
| Basu | MANISH |
| Pramada | RAVI |
| Ravi | ASHISH |
+---------+----------+
```

EXPERIMENT - 6

AIM : To use aggregate functions in SQL.

THEORY :

By definition, an aggregate function performs a calculation on a set of values and returns a single value. Often, aggregate functions are accompanied by the GROUP BY clause of the SELECT statement.
MySQL provides many aggregate functions that include AVG, COUNT, SUM, MIN, MAX, etc. An aggregate function ignores NULL values when it performs calculation except for the COUNT function.

AVG Function:
The AVG function calculates the average value of a set of values. It ignores NULL values in the calculation.
SELECT AVG(column_name) [as name]
FROM table_name ;

COUNT Function:
The COUNT function returns the number of the rows in a table.
The COUNT function can be used as COUNT(*) and COUNT(DISTINCT expression) .
SELECT COUNT(*) AS Total
FROM table_name ;

SUM function :
The SUM function returns the sum of a set of values. The SUM function ignores NULL values. If no matching row found, the SUM function returns a NULL value.
SELECT sum(column_name) [as total]
FROM table_name;

MAX Function:
The MAX function returns the maximum value in a set of values.
SELECT MAX(price) [as highest_price]
FROM table_name;

MIN Function :
The MIN function returns the minimum value in a set of values.
SELECT MIN(Price) [as lowest_price]
FROM table_name;

QUERY :

Query: Count the total no of orders.

```
mysql> select count(s_order_no) from sales_order;
+-------------------+
| count(s_order_no) |
+-------------------+
| 6 |
+-------------------+
```

Query: Calculate the average cost price of all the products.

```
mysql> select avg(cost_price) from product_master;
+-----------------+
| avg(cost_price) |
+-----------------+
| 3427.7778 |
+-----------------+
```

Query: Calculate the minimum sale price of the products.

```
mysql> select min(sell_price) from product_master;
+-----------------+
| min(sell_price) |
+-----------------+
| 525 |
+-----------------+
```

Query: Determine the max and min cost price. Rename the title as 'max price' and 'min price' .

```
mysql> select min(cost_price) as 'min price',max(cost_price) as 'max price' from
product_master;
+-----------+-----------+
| min price | max price |
+-----------+-----------+
| 500 | 11200 |
+-----------+-----------+
```

Query: Count the no of products having price greater than or equal to 1500.

```
mysql> select count(product_no) from product_master where sell_price>1500;
+-------------------+
| count(product_no) |
+-------------------+
| 4 |
+-------------------+
```

Query: Find out the product name and their quantities to be delivered.

```
mysql> select description,qty_order from product_master pm , sales_order_details sod
where pm.product_no = sod.product_no group by(description);
+--------------+-----------+
| description | qty_order |
+--------------+-----------+
| 1.44 Drive | 1 |
| 1.44floppies | 4 |
| CD Drive | 2 |
| Keyboards | 3 |
| Monitors | 2 |
| Mouse | 1 |
+--------------+-----------+
```

Query: Find the product no and their quantities for orders placed by client_no '0001' and '0002'.

```
mysql> select product_no,qty_order from sales_order so, sales_order_details sod where
sod.s_order_no = so.s_order_no and (client_no='0001' or client_no = '0002');
+------------+-----------+
| product_no | qty_order |
+------------+-----------+
| p07885 | 2 |
| p07965 | 2 |
| p03453 | 2 |
+------------+-----------+
```

Query:Find the product no and quantities for orders placed by ' Vandana' and ' Ivan '.

```
mysql> select sod.product_no , sod.qty_order from client_master cm , sales_order so ,
sales_order_details sod where so.s_order_no = sod.s_order_no and cm.client_no =
so.client_no and cm.Name in ('Ivan','Vandana');
+------------+-----------+
| product_no | qty_order |
+------------+-----------+
| p07885 | 2 |
| p07965 | 2 |
| p03453 | 2 |
+------------+-----------+
```

Query:. Find order no , client no. , and salesman no. where more than one salesman has received a client.

```
mysql> select s_order_no , client_no , salesman_no from sales_order group
by(salesman_no) having(count(client_no)>1);
+------------+-----------+-------------+
| s_order_no | client_no | salesman_no |
+------------+-----------+-------------+
| 019002 | 0002 | 500001 |
| 019001 | 0001 | 500002 |
+------------+-----------+-------------+
```

Query: Print the description and total quantity sold for each product.

```
mysql> select description , qty_disp from product_master pm , sales_order_details sod
where pm.product_no = sod.product_no;
+--------------+----------+
| description | qty_disp |
+--------------+----------+
| 1.44floppies | 4 |
| Monitors | 2 |
| Mouse | 1 |
| Keyboards | 3 |
| CD Drive | 1 |
| 1.44 Drive | 0 |
+--------------+----------+
```

Query: Find the value of each product sold.

```
mysql> select description, sell_price from product_master group by (description);
+--------------+------------+
| description | sell_price |
+--------------+------------+
| 1.22 Drive | 1050 |
| 1.22floppies | 525 |
| 1.44 Drive | 1050 |
| 1.44floppies | 525 |
| 540 HDD | 8400 |
| CD Drive | 5250 |
| keyboards | 3150 |
| Monitors | 12000 |
| Mouse | 1050 |
+--------------+------------+
```

Query: Select product_no, qty_order for each product.

```
mysql> select product_no, qty_order from sales_order_details;
+------------+-----------+
| product_no | qty_order |
+------------+-----------+
| p00001 | 4 |
| p03453 | 2 |
| p06734 | 1 |
| p07868 | 3 |
| p07885 | 2 |
| p07965 | 2 |
| p07975 | 1 |
+------------+-----------+
```

Query: Select product_no, description and qty_order for each product.

```
mysql> select description, pm.product_no, qty_order from product_master pm,
sales_order_details sod where pm.product_no = sod.product_no;
+--------------+-----------+-----------+
| description | product_no | qty_order |
+--------------+-----------+-----------+
| 1.44floppies | P00001 | 4 |
| Monitors | P03453 | 2 |
| Mouse | P06734 | 1 |
| CD Drive | P07885 | 2 |
| keyboards | P07868 | 3 |
| 540 HDD | P07965 | 2 |
| 1.44 Drive | P07975 | 1 |
+--------------+-----------+-----------+
```

EXPERIMENT - 7

AIM : To write nested subqueries and correlated subquerries.

THEORY :

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.
A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

There are mainly two types of nested queries:
Independent Nested Queries: In independent nested queries, query execution starts from innermost query to outermost queries. The execution of inner query is independent of outer query, but the result of inner query is used in execution of outer query. Various operators like IN, NOT IN, ANY, ALL etc. are used in writing independent nested queries.
Co-related Nested Queries: In co-related nested queries, the output of inner query depends on the row which is being currently executed in outer query.

Subqueries with the INSERT Statement
Subqueries also can be used with INSERT statements. The INSERT statement uses the data returned from the subquery to insert into another table. The selected data in the subquery can be modified with any of the character, date or number functions.

Subqueries with the UPDATE Statement
The subquery can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.

Subqueries with the DELETE Statement
The subquery can be used in conjunction with the DELETE statement.

QUERY :

mysql> select * from client_master;

```
+-----------+---------+--------+------------+---------+---------+----------+
| client_no | name  | city   | state        | pincode | bal_due | phone_no |
+-----------+---------+--------+------------+---------+---------+----------+
| 2     | Vandana | Madras | Tamil Nadu | 780001 | 1000.00 |        NULL |
| 3     | Pramada | Bombay | Maharastra | 400057 | 5000.00 |        NULL |
| 4     | Basu| Bombay | Maharastra | 400056 |     0.00 |NULL |
| 5     | Ravi | Bombay | Delhi    | 100001 | 2000.00 |           NULL |
| 6     | Rukmini | Bombay | Maharastra | 400050 |        0.00 |NULL |
+-----------+---------+--------+------------+---------+---------+----------+
```

Query: Display the customer name, address, city and pincode for the clients who live in the same city as 'Basu'. Basu's details should not be displayed

```
mysql> select * from client_master where city=(select city from client_master where
name='Basu');
+-----------+---------+--------+-----------+--------+---------+----------+
| client_no | name | city   | state       | pincode | bal_due | phone_no |
+-----------+---------+--------+-----------+--------+---------+----------+
| 3      | Pramada | Bombay | Maharastra | 400057 | 5000.00 |  NULL |
| 4      | Basu| Bombay | Maharastra | 400056 |      0.00 |NULL |
| 5      | Ravi | Bombay | Delhi    | 100001 | 2000.00 |       NULL |
| 6      | Rukmini | Bombay | Maharastra | 400050 |         0.00 |NULL |
+-----------+---------+--------+-----------+--------+---------+----------+
```

Query: Display the details of all the customers who live in the same city and has the same balance due as 'Basu'. Basu's details should not be displayed

```
mysql> select name from client_master where city=(select city from client_master where
name='Basu') and bal_due=(select bal_due from client_master where name='Basu');
+---------+
| name         |
+---------+
| Basu|
| Rukmini |
+---------+
```

Query: Display the details of all the customers who live in the same city and has the same balance due as 'Basu'. Basu's details should not be displayed

```
mysql> select * from client_master where city=(Select city from client_master where
name='Basu') and bal_due=(select bal_due from client_master where name='Basu') and
name<>'Basu';
+-----------+---------+--------+-----------+--------+---------+----------+
| client_no | name | city   | state       | pincode | bal_due | phone_no |
+-----------+---------+--------+-----------+--------+---------+----------+
| 6      | Rukmini | Bombay | Maharastra | 400050 |         0.00 |NULL |
+-----------+---------+--------+-----------+--------+---------+----------+
```

Query: Display product details of those products that have profit% less than all products that have '1.44floppies' in their descriptions.

```
mysql> select * from product_master where profile_percent<(select profile_percent from
product_master where Description="1.44floppies");
Empty set (0.01 sec)
```

Query: Display the names of client who have placed orders worth Rs.10000 or more.

```
mysql> select name from client_master,Sales_Order_Details, Sales_Order where
product_rate >10000 and client_master.client_no= Sales_Order.client_no and
Sales_Order.S_order_no=Sales_Order_Details.s_order_no;
Empty set (0.00 sec)
```

Query: Display the client names who have placed orders before any orders placed by client_no '0003'.

```
mysql> select c.name from client_master c, Sales_Order s where c.client_no=s.client_no
and s_order_date<(select s_order_date from Sales_Order where client_no=0003); Empty
set (0.00 sec)

mysql> select * from Sales_Master;
+-------------+----------+------------+--------+---------+---------+-----------+----------+---------
+-------+
| salesman_no | sal_name | address    | city | pincode | sal_amt | tgt_to_get | ytd_sales |
remarks | State |
+-------------+----------+------------+--------+---------+---------+-----------+----------+---------
+-------+
| 500001      | Kiren        | A/14 Worli | Bombay | 400002 | 3000.00 |         100.00 |
        50.00 | Good         | Mah |
| 500002      | Manish   | 64, Nariman | Bombay | 400002 | 3000.00 | 100.00 |
        100.00 | Good        | Mah |
| 500003      | Ravi | P-7 Bandra | Bombay | 400032 | 3000.00 |          100.00 |
        100.00 | Good        | Mah |
| 500004      | Ashish  | A/5 Juhu         | Bombay | 400044 | 3500.00 | 200.00 |
        150.00 | Good        | Mah |
+-------------+----------+------------+--------+---------+---------+-----------+----------+---------
+-------+
4 rows in set (0.00 sec)
```

EXPERIMENT - 8

AIM : To create

i. Indexes

ii. Views

THEORY :

SQL CREATE INDEX Statement
The CREATE INDEX statement is used to create indexes in tables.
Indexes are used to retrieve data from the database very fast. The users cannot see the indexes, they are just used to speed up searches/queries.

CREATE INDEX Syntax
Creates an index on a table. Duplicate values are allowed:

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

SQL CREATE VIEW Statement
In SQL, a view is a virtual table based on the result-set of an SQL statement.
A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

CREATE VIEW Syntax

```
CREATE VIEW view_name AS SELECT column1, column2, ... FROM table_name
WHERE condition;
```

QUERY :

Query: Create an index on the table client_master, field client_no.

```
mysql> create index field on client_master(client_no);
Query OK, 0 rows affected (0.18 sec) Records: 0 Duplicates: 0 Warnings: 0

mysql> desc Sales_Order;
+--------------+------------+------+-----+---------+-------+
| Field        | Type| Null | Key | Default | Extra |
+--------------+------------+------+-----+---------+-------+
| s_order_no   | varchar(6) | NO   | PRI | NULL     |       |
| S_order_date | date       | YES |       | NULL     |       |
| client_no    | varchar(6) | YES |       | NULL     |       |
| salesman_no | varchar(6) | YES |      | NULL     |       |
| dely_type    | char(1)    | YES |       | NULL     |       |
| billed_yn    | char(1)    | YES |       | NULL     |       |
| dely_date   | date | YES |         | NULL        |        |
| order_status | varchar(10) | YES |    | NULL       |        |
+-------------+------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

Query: Create an index on the sales_order, field s_order_no.

```
mysql> create index field on Sales_Order(s_order_no); Query OK, 0 rows affected (0.17
sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc Sales_Master;
+-------------+-------------+------+-----+---------+-------+
| Field        | Type| Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| salesman_no | varchar(6)   | NO   | PRI | NULL    |        |
| sal_name    | varchar(20) | NO   |      | NULL       |        |
| address     | varchar(20) | NO   |      | NULL       |        |
| city  | varchar(20) | YES |       | NULL       |        |
| pincode     | decimal(6,0) | YES |      | NULL       |        |
| sal_amt     | decimal(8,2) | NO   |      | NULL       |        |
| tgt_to_get | decimal(6,2) | NO   |      | NULL       |        |
| ytd_sales   | decimal(6,2) | NO   |      | NULL       |        |
| remarks     | varchar(30) | YES |      | NULL       |        |
| State        | varchar(20) | YES |      | NULL       |        |
+-------------+-------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

Query: Create an unique index on the table salesman_master, field salesman_no.

```
mysql> create unique index fields on Sales_Master(salesman_no); Query OK, 0 rows
affected (0.14 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> desc Sales_Order_Details;
+--------------+--------------+------+-----+--------+-------+
| Field        | Type| Null | Key | Default | Extra |
+--------------+--------------+------+-----+--------+-------+
| s_order_no   | varchar(6)       | NO   | PRI |         |        |
| product_no   | varchar(6)       | NO   | PRI |         |        |
| qty_order   | decimal(8,0) | YES |      | NULL       |        |
| qty_disp    | decimal(8,0) | YES |      | NULL       |        |
| product_rate | decimal(10,2) | YES | | NULL       |        |
+--------------+--------------+------+-----+--------+-------+
5 rows in set (0.00 sec)
```

Query: Create an composite index on the sales_order_details table for the column s_order_no and product_no.

```
mysql> create index field on Sales_Order_Details(s_order_no, product_no); Query OK, 0
rows affected (0.16 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Query: Create view on salesman_master whose sal_amt <3500.

```
mysql> create view V as select * from Sales_Master where sal_amt<3500; Query OK, 0
rows affected (0.03 sec)
```

Query: Create a view client_view on client_master and rename the columns as name, new_city, pincode_new, state_new.

```
mysql> create view client_view(name_new,city_new,pincode_new,state_new) as select
name,city,pincode,state from client_master;
Query OK, 0 rows affected (0.03 sec)
```

Query: Select the client names from client_view who lives in city 'Bombay'.

```
mysql> select name_new from client_view where city_new="Bombay";
+----------+
| name_new |
+----------+
| Pramada |
| Basu|
| Ravi |
| Rukmini |
+----------+
4 rows in set (0.00 sec)
```

Query: Drop the view client_view.

```
mysql> drop view client_view;Query OK, 0 rows affected (0.00 sec)
```