**// Selecting a database form the databases**

**//USE wale saare**

USE database\_name;

**//Selecting the column from the database table**

**//SELECT wale saare**

// SELECT DISTINCT column name

SELECT \* (all database column get selected)

SELECT column1 ,column2,column3,column4

SELECT column,

column1,

column2,

column3 AS "CUSTOMER name" (for giving the column an alias name)

**//FROM wale sare**

FROM column\_NAME

**// filtering the data-----**

**//WHERE wale sare**

WHERE price>4 (> ,>=,<=,=,!=,<>)

where name=”harry”

where birth\_date>2001 AND points>2999

where birth\_date>2001 OR points>2999

//------------------------------------------------

PRECEDENCE OF THE OPERATOR

AND

OR

NOT

**//------IN OPERATOR-----------------------------------**

**// checking multiple value at same time**

WHERE state IN ('value1','value2','value3')

WHERE state NOT IN ('value1','value2','value3')

**// BETWEEN OPERATOR-----------------------------------**

**// checking the value between an interval**

WHERE points BETWEEN 1000 AND 3000

**// LIKE CLAUSE---------------------------------------**

**// checking the string in the database**

WHERE last\_name LIKE 'B%' RETURN ALL WITH STARTING WITH B capital ignored

where last\_name like '%y' return all with ending with y

where last\_name like '%b%' return all with including b in the name

where last\_name like '-----y' return the last name with having 6 character and end with y

where last\_name not like '%y'

**// REGEXP string clause**

Where last\_name REGEXP ‘string’

Where last\_name REGEXP ‘^STRING’ (**It means start with string )**

WHERE last\_name REGEXP ‘STRING$’ (**it means end with string)**

Where last\_name REGEXP ‘^field|rose$|ram’

Where last\_name REGEXP ‘e[fmq]’ (search for ef,em,eq)

Where last\_name REGEXP ‘Z[a-h]I’ (search for all za,zb,zc,zd………….zh)

**//FINDING THE MISSING VALUE**

WHERE phone is null

Where phone is not null

**// SORTING OF THE DATA**

**USING ORDER BY CLAUSE**

ORDER BY first name

Order by first name DESC

ORDER BY LASTNAME **DESC**,FIRST NAME

**// LIMITING THE RECORD**

SELECT \*

FROM customer

LIMIT 3

(OR WE CAN DO)

LIMIT 6,3 (skip 6 record and show 3)

//**join**

**From big\_table\_name o s\_tbl\_name (alias for table)**

join customers on orders.customer\_id=customers.customer\_id

**// joining across database**

**Use sql\_inventory;**

**Select products.product\_id,orders.order\_id**

**From store.orders o**

**Join products p on p.product\_id=0.orderid**

**//self join**

**use** sql\_hr;

**select** e.employee\_id,e.first\_name as employee,m.first\_name as boss

**from** employees e

**join** employees m

**on** e.reports\_to=m.employee\_id

// **Joining more than two table**

Compound columns

If primary key is not a single column then

**Select** \*

**From** order\_items oi

**Join** order\_items\_notes oin

**On** oi.order\_id=oin.order\_id

**And** oi.product\_id=oin.product\_id

Or we can write

Using(order\_id,product\_id)

**//IMPLICIT SYNTAX**

**use** store;

**SELECT** \*

**from** orders o,customers c

**where** o.customer\_id=c.customer\_id

**OUTER JOIN**

**use** store ;

**select** o.order\_date as date,o.order\_id,c.first\_name as name,s.name as shipper,os.name as status

**from** orders o

**right**  **join** customers c

**on** o.customer\_id=c.customer\_id

**left** **join** shippers s

**on** s.shipper\_id=o.shipper\_id

**left** **join** order\_statuses os

**on** o.status=**os**.order\_status\_id

**SELF OUTER JOIN**

**USE** sql\_hr;

**select** e.employee\_id,e.first\_name,m.first\_name

**from** employees e

**left** **join** employees m

**on** e.reports\_to =m.employee\_id

**//using clause**

If a table has two column having same name then we can use the **using** clause for

Doing the work easier

From order

Join customer

Using (customer\_id)

**//Natural join**

**USE** store**;**

**select \***

**from** orders **o**

**natural join** customers **c**

**//cross join**

Select \*

From customer c

Cross join orders o

Both part of the query should produce the same no. of column in order to avoid the errors

**//Union clause**

select order\_id,

order\_date,

"active" as status

from orders

where order\_date>='2019-01-01'

**union**

select order\_id,

order\_date,

"archived" as status

from orders

where order\_date<'2019-01-01'

**INSERTING IN A single row**

**INSERT INTO customers**

**VALUES**(default,'abhisumant','tiwari','2001-06-03'

,null ,'sector 8','noida','up',default)

**INSERT INTO** customers**(**first\_name,last\_name,date\_birth,address,city ,state)

**VALUES**('abhisumant','tiwari','2001-06-03'

,'sector 8','noida','up',)

**Inserting multiple row**

**Insert into TABLENAME**

**Values**  (‘value1’,value2,’vlaue3’),

(‘value1`’,value2`,value3~)

**Inserting multiple table**

insert into orders(customer\_id,order\_date,status)

values (1,'2019-09-28',1);

insert into order\_items

values (last\_insert\_id(),1,1,2.95),

(last\_insert\_id(),2,1,2.23);

select last\_insert\_id()

**//creating copy of a table**

**CREATE table** table\_name **as**

**Select \* from** table\_to\_be\_copied

**// using select statement as an query for insert statement**

**Insert into** newtablename

**Select \***

**From** orders

**Where** DATE\_COLUMN\_NAME**>**’2019-01-01’

**UPDATE SINGLE ROW**

Update table\_name

Set value=new\_val

Where some condition

**UPDATE MULTIPLE COLUMN**

UPDATE table\_name

Set col1=new val 1, col2=new val2

Where some condition

**UPDATING MULTIPLE ROW**

JUST THE SAME BUT THE WHERE CLAUSE SHOULD RETURN THE MULTIPLE ROW VALUES

**UPDATING USING THE SUBQUERY**

UPDATE TABLE

SET VAL=NEW\_VAL

WHERE CLIENT ID =

(SELECT CLIENT ID

FROM CUSTOMERS

WHERE SOME CONDITION WHICH GIVE THE **EXACT** RESULT)

UPDATE TABLE

SET VAL=NEW\_VAL

WHERE CLIENT ID  **IN**

(SELECT CLIENT ID

FROM CUSTOMERS

WHERE SOME CONDITION WHICH GIVE THE **MULTIPLE** RESULT)