SQL

There are mainly 5 categorizes of SQL command

1) DDL → Data Definition Language → CREATE, DROP, DESCRIBE, TRUNCATE, ALTER
   ↳ Add, modify, change, drop, rename
2) DQL → Data Query Language → SELECT, SHOW, HELP
3) DML → Data Manipulation language → INSERT, UPDATE, DELET
4) DCL → Data Control Language → GRANT, REVOKE
5) TCL → Transaction Control language → COMMIT, ROLLBACK, SAVE POINT
   DB consistency

There are 3 types of Data
1) structured Data
2) Unstructured Data
3) Semistructured Data

what is Database Management system
→ (i) Used to manage Data
(ii) provides protection & security
also provide auto backup
(iii) e.g. SQL, Oracle.

4 types of Databases
1) RDBMS
2) Object oriented DBMS
3) Distributed Database
4) Hierachi DBMS
5) Network DBMS
6) Data Warehouse
7) NO SQL Database
8) Graph DBMS
9) cloud

4 types in RDBMS
1) My SQL
2) ORACLE
3) DB2
4) Postergre SQL

SQL is not Case sensitive

-- = single line comment

/* = multiline comment

* ALTER → only change the structure
it can not change the ~~structure~~ content

(DDL)

CREATION

CREATE database <database name> } Database creation

Use <database name>

create table <table name> (col1 datatype1, ---) → creation of table in the Database
Attributes

insert into <table name> values ( ) → Add data into the table

(*) Alter → (options in Alter)

1) add → To add new column (column / constraint)

alter table <table name> (add) <new column name> datatype
↳ If this not given directly DEIC of table given then also it work

2) modify → To modify datatype of column

alter table <table name> modify <col name> datatype

3) change → To change the name of column

alter table <table name> change <column name> <new col name> datatype

4) Drop → To delet (column / constraint)

alter table <table name> Drop <column name>

5) rename → To rename the table

alter table <table name> rename <new table name>

change can use to rename the column } Difference
modify can not rename the column

Add column at specific location in table

① Alter table ~~em~~ <tab_name>
add <col-name> <data type> after <after what col you want>

⑤ Alter table <table_name>
add <col_name> <Data Type> first
add <col_name> <DataType>
} Multiple columns can be added

*auto increment
set auto_increment_increment = 5; → change these value if you want to increse it
It can be run multiple time

1) change Data Type of existing table (Both can Be used)

Alter table <tablename> change column ~~salary~~ <colname> <col name> <new Data Type>
Alter table <table name> modify <col_name> <Data Type>

2) chang name of column (only change can make changes in the names of table

Alter table <table name> change column <col name> <new col name> <New Data Type>
Alter table <table name> change column <col name> to <new col name> <Data Type>

4) Drop → To delet (column / constraint)

alter table <table name> Drop <column name>

5) rename → To rename the table

alter table <table name> rename <new table name>

change can use to rename the column ?

Constraints

To apply rules & regulation we use **constraints**

1) Not null (Alter)

alter table <table name> modify column <column name> <datatype> not null;

2) Unique (Alter)

alter table <tablename> ~~modify~~ add unique (column_name);

3) Primary key (Alter) (Unique representation of the record)

alter table <table name> add constraint primary key (<column_name>);

~~We~~ we add not null and unique in the constraint place then automatically in description the table column name key will show us the name 'primary key"

not null + unique = primary

create table <table name> (<col_name> not null, unique)

→ automatically these will be shown as primary key in description table.

(i) Not NULL + unique
(ii) enforce entity integrity
(iii) only one in a table
(iv) Can be apply on string & numeric data type.

} what is primary key ?

we can create primary key seperately at the end.

primary key (<column name>)

4) ENUM (new constraint) (only single values ~~from~~ list)

limit the values chosen from a list (specified list is given then we can only insert values from the given list other values will not be insert)

<col_name> ~~datatypes~~ ENUM ('value1', 'value2', ---, 'value N');

we can't insert combination of two entity from list.

Update → To update record in table

Update <table name> set <col name> = value

where → filtering the rows

<col name> = value where condition

delet → delet the record

delet from <table name> — All things

delet from <table name> where condition — some data

constraints in SQL limitation on the table so table can't contain any type of information.

→ Types of constraints (Rules / Restrictions)

1) Not NULL → will not accept the null values
2) Unique → this will avoid duplicate values
3) default → Add default value if null is found
4) check → cheaks given range is used or not
5) Primary key → not null + unique
6) foreign key → interlink two table
   indicates primary key
7) Auto increment → increase the value
   default 1 is added.

we can use multime. same as select

set auto_increment_increment = 5; ↳ value change then can be used again

DDL { drop → it will delet all the table as well as the data inside it
Truncate → it will delet the data in table but not the table

Truncate Table <Table_name>

altering the constraints

If we have't add constraint to column then after creation we can add it

1) Not NULL →

alter table <table name> modify id int not Null → To add (column name)

alter table <table name> modify id int → To remove

2) Unique →

alter table <table name> add constraint unique (<column name>)

alter table <table name> drop index <column name>

3) check →

alter table <tablename> add constraint chk1 check(age>=20) (Name of check)

alter table <table name> drop constraint chk1

4) Primary key →

alter table <table name> add constraint primary key (<column name>)

alter table <table name> drop primary key

5) foreign key →

alter table <table name> add constraint fk1 foreign key(<column name>) reference <table name> (<column name>)

alter table <table name> drop constraint fk1;

6) ENUM

alter table <table name> add <col name> ~~Enum~~ ENUM(<values>)

* DQL

select, show, help

1) ~~select~~ →

2) show → To see the tables, data bases

show create table <table name>

meta data

3) help → we can find any syntax with it

help 'create table'

help '<syntax you want to find>'

3) select → we can select the Required data

select * from <table name> — to select all Records

select <column name> from <table name> — to select specific Records.

where → select <field> from <table name> WHERE <condition>

Decimal Data Type → Decimal (m, d) m = length of the numb
d = after "." how many number are present

3752.96 ⎫

Decimal (6,2) → To store that Number

float = char
Decimal = Varchar
} same different.

Operators

1) Arithmatic operators → +, -, *, /, %
2) comparision operators → =, !=, <, >, <=, >= <> → not equal
3) Logical operators → and, or, not
4) membership operators → in not in (cheaks whether Data is present or not
5) Range Operator → between (In between Data is shown)
   = between (lower_value) and (Upper_Value)
6) is null or not null - (value is null or not cheaked)
7) Case logic - used to get information about the data with given one more than condition

Range syntax →

Select * from <table_name> where <column_name> between (LV) and (

(Operator)
IN → **Subtitute** of **multiple or** conditions

When we want multiple values (similar) then

Syntax = Select * from <t_name> where <col_name> IN (<Para>)

Like operator → when we want to search the content In pattern manner

% used for the like operator (auto fills the next char)
three pattern present

1) %K% = K is present any where
2) %K = K present at the end
3) K% = K present at start

syntax =
select * from <table-name> where <col-name> like '%_%';

order by → used to sort the record in ascending or descending manner

default order by is set in ascending order

syntax { select * from <table_name> order by <col_name>
select * from <tab_name> order by <col_name> DESC; } for desinding order.

It can work on Both numeric & catagorical Data

IS null or not null → Used to find if the column is null or not

syntax -
select * from <table_name> where <column_name> is null;
select * from <table_name> where <column_name> is not null;

Q In whose name a comes at second position *
→ (code)=
select * from <table_name> where <col_name> Like '%a_';

* Ways to add the primary foreign key *

Primary key → <col name> datatype primary key

foreign key → <col name> datatype,
foreign key (<col name>) Refrence <table name> (<col name>)

Date Datatype → ISO
"yyyy-MM-DD"

Blob/Text → when we have to write paragraf
then use these Data Type

Limit → first 5 names or any no given name/data will be shown

Select * from <table name> limit <limit you want>

AS → temporary changes the column name (used in select)

Select <col_name> as <changed_name> from <table_name>

Distinct → diff types of data present in that column is shown

e.g. there are 3 branch (1,2,3) then it will show that

select Distinct <column_name> from <table_name>

Built in SQL functions

A) String function

Used for char & varchar

1) concat → combines two or more string into one

```
select concat ('Good', 'Morning');
output = 'Goodmorning'
```

2) Lower → converts all the characters to lowercase

```
select lower (<string>)
```

3) Upper → converts all the characters to upper case

4) Replace → searches the string & replaces that with other

```
select Replace ('Let us Learn', 'Learn', 'Earn')
```

it firstly fines the learn & then replaces it with Earn

5) Substring → extracts some character from string

```
select substr (<string>, <start>, <length>)
    substring
```

6) Length → Gives the length of string

```
select Length ('MySQL is robust');
```

other than cource. string fun^n

1) ASCII → gives Ascii value of character
2) CHAR → gives character of Ascii value
3) CHARINDEX → position of substring in string
4) CONCAT with + → same as concat but + used to add
5) CONCAT_WS → Add two or more string with separator
   concat_ws (seperator, string^1, string^2, ..., string^n)
6) DATA LENGTH → shows the number of bytes used to represent
7) DIFFERENCE →
8) FORMAT →
9) LEFT → Extract number of character from left
   * select Left (string, no-of-char)
10) LTRIM → Removes the leading spaces left (' ↑ This space will be removed SQL Totu') *
11) REPLICATE → Repeat string for specified number of times
    * Select Replicate (string, no_of_time)
12) REVERSE → reverses a string
    Select Reverse (string)
13) RIGHT → Extract number of character from Right
    * Select Right (string, no_of_char)
14) RTRIM → Removes the ending space ('SQL Totu ↑ this will be removed') *
15) STUFF → Delet a part of string at specific position & add another part in it (IMP) **
    stuff (string, start, length, new string)

B) MATH FUNCTION

Used for Numeric Values

1) ABS(X) :- Returns the absolute value of X (converts the minus into +)

Select ABS (-32)

output = 32

2) CEIL(X) :- gives the CEILING value (Point value given then convert it into full value)

Select CEIL (17.3)

output = 18

3) FLOOR(X) :- Returns largest integer value not greater than X

Select FLOOR (17.3)

Output = 17

4) MOD(N,M) :- N%M - Returns the Reminder

Select MOD (29, 9)

output = 2

5) ROUND (X) - Rounds the number to 2 Decimal Places

Select Round (-1.23)

output = -1

Round (X,D) - Rounds the number to D Decimal Places

Default D is 0

D can be Negative from that left most values can be o

Round (1.293,1)

output - 1.3

6) TRUNCATE (X,D) :- return the number with the number of digits removed, no rounding of the number will

select truncate (2565.7689, 2)

→ (2565.76)

→ +1 concept of greater than 5 will not be apply

7) EXP (x) →

select exp2(2); → display exponational value

e^2

$e = 2.71$

8) Pow(X,Y) :- X is the value and Y is How many power are you calculating from it

Power(X,Y)

pov(2,2)

output = 4

9) SQRT(x) :- Returns the Square root of x number

SQRT(4)

output = 2

other than course Math fun^n

1) ACOS → Return cosin value of Number
2) ASIN → sine
3) ATAN → tangent

4) AVG → Return the Average value *
   select AVG (<column_name>) from (<table name>)

5) count → Return the number of Records return by quary *

6) LOG → Return the LOG Number

7) MAX → Return the Maximum value *

8) MIN → Return the Minimum Value *

9) RAND → Return Random Decimal Number *

10) SUM → SUM of all values in that column. *

11) GREATEST → Return the gratest from the input

12) LEAST → Return the lowest from input

(C) Date function

1) CURDATE → Returns the current date

```
Select curdate ( );
```

2) NOW → current Date and time at which statement executed (current Pc time is shown)

```
Select now ();
```

3) SYSDATE → Returns the current Date

```
select SYSDATE ();
```

4) LAST Day → Returns the last day of the corresponding month

5) If → If expression true then given value in statement_2 is shown if false then statement_3 value shown

(Aggregate fun^n)

```
If ( condition, value_if_true, value_if_false)
```

6) Month (date) → Return only the month value

```
Select month ('2020-03-21')
```

7) year (date) → Return only the year value

```
selet year ('2020-03-21')
```

8) last_day (date) → Return the last day of the month which the input date belong

```
Selet last_day (date)
```

(9) adddate (date, value) → Return new date with the value added in the input date

value can be positive/Negative

if positive then days will be added

if negative then days will be subtracted.

(10) datediff(date1, date2) → Returns defference In the terms of days

date1 = latest date

date2 = old date

(11) timestampdiff (difference in what, date1, date2) → Returns difference but in terms of year month or days

date 1 = old date

date 2 = New date

'%.p' → AM/PM

h → 24 Hours ← time → '%.H' → Hours → 12 Hours

'%.i' → min

(12) date_format (date, format) → returns the formated Date.

(13) select date_format (sysdate(), "%.m") → gives month

select date_format (sysdate(), "%y") → gives year Y - capital y then 4 digit

y - small y then 2 digit.

select date_format (sysdate(), "%.d") → gives date

select date_format (sysdate(), '%.w') → gives the day

w → small will give only digit

W → capital - Name

(D) Aggregate functions

Comparision function

1) COALESCS ( ) → Returns first not null value from list

```
select COALESCS (NULL, NULL, 'OM', 'DARKONDE');
output → OM
```

2) ISNULL → Return the specified value if expression is NULL otherwise expression

NULL = 1
NOT NULL = 0

default 1 is when it is NULL
otherwise 0

3) GREATEST → largest argument is given

```
select Greatest (2, 4, 10, 81)
output → 81
```

4) LEAST → smallest argument is given

5) If → Explation in previous page

```
Select If(7>5, 'yes', 'NO');
→ yes
```

6) ^

Other than course Aggregate (Adv) fun^n

1) CAST ⟶ convert a value into specified datatype

```
select CAST ( expression AS datatype )
select CAST ( 25.65 as int)
  output = 25
```

2) convert ⟶ convert a value into any datatype

```
select convert ( datatype, expression, style)
select convert ( int , 25.65)
  output = 25
```

3) current user ⟶ Return the name of the current user in the SQL server db

4) ISNUMERIC ⟶ Tells wheather expression is numeric or not

Yes = 1
NO = 0

5) NULLIF ⟶ Returns NULL if two expressions are equal otherwise Return first expression.

```
select NULLIf ( expl , ex2)
```

6) SESSION USER ⟶ Return the name of current user

7) SESSION PROPERTY ⟶ Return session setting for a specified option

8) SYSTEM USER ⟶ Return the logic name for current user

9) User_Name ⟶ Return the DB username base on specific ID

* GROUP BY *

Groups Rows that have the same values into summary

these is used with Aggregate fun^n (count, MAX, MIN, SUM, AVG)

more than one can be selected

```
Select <column_name> (we can add aggrigate fun^n)
from <table_name>
GROUP BY <column_name>
```

column name must be same

* HAVING Clause * (similar are where)

it is used to apply a filter on the result. based on a specific condition

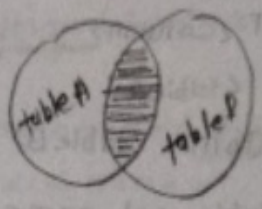
```
Select <column_name>
from <table_name>
WHERE <condition>
GROUP BY <column_name>
HAVING <condition>
```

JOIN

It is used to combine Data from two or more than two tables based on related colmn bet^n them

1) Inner Join

common values in both the table are taken only



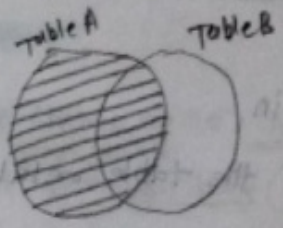

```
SELECT <column_names>
from   <table A>
INNER JOIN   <table B>
on TableA. col_name = Table B. col_name
```

e.g.
```
select *
from customer As c
Inner Join Payment As p
On c. customer_id = p. customer_ID
```

Column name must be same
As c & As P is taken for convinient

2) LEFT JOIN

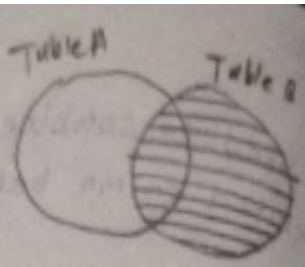
All values from left table with the common values between the all the tables




```
SELECT   <col_name>
FROM     <tableA>
LEFT JOIN <table B>
on TABLE A. col_name = TABLE B. col_name
```

TABLE Given on place of TABLE A is the left table
Al table B will B Right Table.

- 3) Right Join



All values from Right table with the common values from the tables

```
SELECT <column>
from <tableA>
Right Join <table B>
on Table.col_name = Table.col_name
```

} NAME Given at table B will Right Tabl.

4) full Join (workbench don't support these)

All Records from TableA & TableB or Tables will Be Shown

```
SELECT <column>
from <TableA>
full outer Join <TableB>
on TableA.col_name = TableB.col_name
```

5) self Join

in these the table relates to itself (Joins with itself)

a way to use full join in mysql workbench

```
select <column> from <table> left Join <table>
  on <table>.<col> = <table>.<col>
```

Union

```
select <column> from <table> Right Join <table>
  on <table>.<col> = <table>.<col>
```

MODIFY

(can also rename)

1) NOT NULL

Alter table <table name> (To add)
modify <col_name> <Data type> Not Null

Alter table <table name> (To remove)
modify <col name> <Data type>

2) Default

Alter table <table name> (To add)
modify <col name> <Data Type> Default <Def_value>

Alter table <table name> (To remove)
modify <col name> <Data Type>

ADD

(can not rename) (ADD x Drop)

1) Unique

Alter table <table name> Add constraint Unique (<col name>)

Alter table <table name> Drop constraint <col name>

2) Primary key

Alter table <table name> Add constraint primary key (<col name>)

Alter table <table name> Drop ~~constraint~~ primary key

Here name of colum doesn't required Because there is always one primary key

3) foreign key

Alter table <table name> Add constraint (dummy name) foreign key (<col name>) refrences <table name> (<col name>)

1) show create table <table name> → get name of foreign key
Alter table <table name> drop constraint foreign key <name of for.key>

2) show create table <table name>
↳ take the index name and drop that index name

Alter table <table name> drop index <index name>

(perform these two steps to drop the foreign key)

Update

set SQL_safe_update = 0; ⟶ close the safe mode first

Update <table name> set <colname> = <values> ⟶ value will be set for hole column

Update <table_name> set <col name> = <value> where <condition> ⟶ To change a specific record

Update Data from exsiting table.

DELETE

Delet from <table name> where <condition> ⟶ specific

delet	drop	truncate
possiblity of restoring	no possibility of restoring	no possibility of restoring
table structure remains	table structure deleted	table structure remains same
DML	DDL	DDL

Alter ⟶ table format update

Update ⟶ Records update.

IN

To specify multiple values in where ~~das~~ clause

select <colname> from <tablename> where <colname> IN <values>

Not IN ⟶ Rather than values given in IN

Between

Highlimit is excluded so give +1 value when you want the required high value.

Select values within given range.

select <colname> from <table name> WHERE (col condition) BETWEEN <values>
(lowlimit highlimit)

for Date Datatype ⟶ bet'n "YY-mm-DD" and "YY-mm-DD"

Distinct

IN, NOT IN (membership operator)

1) IN ⟶ Replaces multiple 'or' conditions

Select * from <table name> where <colname> IN (values)

2) Not IN ⟶ opposite of IN (excludes the values given and other are shown)

select * from <table name> where <colname> Not IN (values)

Like Operator

only works on character and date values.

2 wild cards for like

1) _ ⟶ for single character (we can give multiple)

2) % ⟶ represents 0 or more characters

select * from <table name> where <col name> like 'a%' ; → first letter a

select * from <table name> where <col name> like '%a' ; → last letter a

select * from <table name> where <col name> like '%a%' ; → any place a

select * from <table name> where <col name> like '_a' ; → only one in front of a

select * from <table name> where <col name> like 'a_' ; → only one after a

select * from <table name> where <col name> like '_a%' → a at second place.

% → modulus

IS NULL

select * from <table name> where <col name> is null

If null values are present then that values are shown

selell * from <table name> where not <col name> is Null ⟶ Rather than NULL value other values are shown

SORTING (Arranging the values)

Order By ⟶ used to specify column ordering

Select
from
where ⟶ if needed
order by (Asc DEsc)

default asc is set

select * from <~~tbl~~ Table name> order by <col name> asc / desc

limit ⟶ specifyes the number of records to get

Offset
↓
जो No दिया
आगे दिया
चुड़ाम

select
from
where
order by
offset

select * from <table name> order by <col name> DESC
limit <limit No> ⟶ Top 5, Top 3

select * from <table name> order by <col name> DESC limit <limit No>
offset ⟶ start from given no

Case Clause

used for conditioning

```
select <col names> case <col name>
when <expression> then <action>
when <expression> then <action>
else <action>
end <aliasname>
from <table name>
```

*INSERT values into tables using select

```
insert into <table name> select <col name> from <table name>
```

- `insert into <table name>` → New table which will be created
- `select <col name> from <table name>` → existing table

```
create table <table name> as select <col name> from <table name> where
                                                    <condition>
```

```
select show Help;
show create table <table name>;
```
} Information of table

```
select columns from <table name> ⟶
```

select xo

Built in functions

it is a piece of code that take input are return value when we are performing a single task multiple time so to avoid that we create function.

1) single row function = return one value for one row
works row by row

2) Multi row function = return one value for group of rows
works on group of row
Aggrigate function

* Group By *

Used to group rows from table. and it has same values as summary rows
often used with the Aggrigate function

Having

it is added because where can not used with Aggrigate function

Subquery

query within the query

Types of subquery → ① single row Subquery
② multi row subquery

① single row subquery =
single row { single column written by the inner quary.

comparision operator can be used

```
select customername, creditlimit from customers

where creditlimit >
      (select max(creditlimit)
       from customers
       where city = "nyc"
       group by city);
```

② Multi row subquery =
multi row { single column is written by the inner quary
The result can not compare by using comparision operator

Multi row operator =
A) in substitute of "="
B) any (can be combined with =, <>, <, >,)
<any = less than
>any = Greater than

c) all (can be combined with <, >, <=, >=)

```
select customername, creditlimit from customers
  where creditlimit = any
        (select max(creditlimit) from customers
          where country in ('usa', 'uk', 'france') group by
           country);
```

find all the coustomers having creaditlimit greater than avg credit limit of customers in the country usa uk france.

* Multiple column subquery *

comparing the value of the two column based on common condition

both the subquery must have the same condition so it could be combined

```
select firstname, jobtitle, officecode
  from employee
 where (jobtitle, officecode) =
       (select jobtitle, officecode
        from employees
        where first name = 'pamela');
```

Joins

1) cross join (cartesian)

every row of the first table is joined to every row of the second table.

it dosent' give any meaningful result
that's why join is used with join condition.

2) Inner Join

Join the rows of two tables but only if both the rows are specifying the join condition

There are two types

A) Equi Join condition -

most frequently used
it is based on the concept of foreign key
condition is created using the (=) operator

```
select firstname, city
from employees, offices
where employees.officode = office.officode
                           └──same──┘
```

Join condition can be specified using the where clause but where is also used to specify other condition

we make use of on or using clause to specify just join condition

To use on & using the table must be combined with the help of join keyword.

```
select firstname, e.officode, city from
employees e join offices o on
(e.officecode = o.officecode);
```

} ON

```
select firstname, city
from employees join offices
using (officecode);
```

} using

B) NON equi join condition

Here the condition is created using any other operator expect for (=) operator

```
select customername, creditlimit, grade
from customers join creditgrade
 on (creditlimit between lowval and highval);
```

3) Outer Join

Return all the rows that satisfy the join condition and also return all the rows from one of the table that does not satisfy the join condition.

There are two types

A) left outer join

B) right outer join

A) **left outer join**

```
select customersname, firstname, employeename
from customers left outer join employees
on (salesrepemployeenumber = employeenumber);
```

B) **Right outer join**

```
Select customersname, firstname, employeename
from customers Right outer join employees
on (salesrepemployeenumber = employeenumber);
```

4) **Self Join**

Join table to itself
treat one table as two different tables using alias.

```
select emp.firstname, mgr.firstname
from employee emp join employee mgr
on (emp.reportsto = mgr.employeenumber);
```

} only those who have manager

If you want all the change join to left join

Update is Used to update records from table

Update <table name> set <colname> =value where

view

view is a virtual table
is a logical table

contains rows and columns as same as real table

fields of the view are from one or more real table

change made in view will be shown in real table

Advantage → (i) Ristrict Data Ascess
(ii) Make complex query easy
(iii) Diff view of same Data

```
create view <viewname>
as < >
<sql query>
```

Types of view → 1) simple view
2) complex view

1) simple view =

not include group by, join, aggregate functions, having, set operators, subquery, single row function

perform DML (insert, Update, Delete) Operations.

2) Complex view -

It might include group by, join, aggregate, group function having, set operations, subquery, single row function

can't perform DML operations

Create or replace <view> <viewname>
as
<SQL Quary>

create view <viewname> as
Select <colnames> from <tablename> } Syntax
where <condition>