

DDL - Command

1. ~~Or~~ create Command

* To create a Database

syntax :

create database database-name ;

e.g - create database College ;

* To show databases.

syntax :

show databases ;

* To create a Table

• To select a Database

syntax :

use database-name ;

e.g - use College ;

To create

syntax :

create table table-name (

- To create table.

syntax:

```
create table table-name (  
    attribute-name1 datatype(size),  
    attribute-name2 datatype(size),  
    attribute-namen datatype(size)  
);
```

e.g.-

```
create table Student (  
    Rollno int(5),  
    Name char(20),  
    Branch char(15)  
);
```

- * To show a Relation (structure)

syntax:

```
desc table-name ;
```

e.g.- desc Student;

- * To show tables

syntax:

```
show tables ;
```

2. ~~A~~ alter Command

Used :-

- To modify a structure of a table
- To add an attribute at specific position
- To delete an attribute
- To change datatype
- To change table name or rename
- To rename an attribute

* To add a attribute (at last position)

syntax:

alter table table-name add att-name datatype(size);

e.g - alter table Student add result int ;

desc Student ;

* To add attribute (at first position)

syntax :

alter table table-name add attribute-name datatype(size) first;

e.g - alter table Student add id varchar(15) first ;

desc Student ;

* To add attribute at specific position

Syntax :

```
alter table table-name add new-attribute-name datatype(size)
after attribute-name ;
```

e.g - alter table Student add address varchar(40)
after Name ;

desc Student ;

* To add multiple attribute

Syntax :

```
alter table table-name add attribute-name1 datatype(size)
add attribute-name2 datatype(size),
add attribute-name3 datatype(size),
add attribute-name4 datatype(size),
add attribute-name5 datatype(size);
```

eg- alter table Student add semester varchar(10),
add session int(5),
add city char(15);

desc Student ;

* To modify a table (datatype or size)

Syntax :

alter table table-name modify attribute-name datatype(size);

e.g- alter table Student modify address varchar(15);
desc Student ;

* To delete attribute

syntax :

alter table table-name drop column attribute-name ;

e.g- alter table Student drop column id ;
desc Student ;

* To delete multiple column

Syntax :

alter table table-name drop column attribute-name1,
drop column attribute-name2,
.....
drop column attribute-namen;

e.g- alter table student drop column Branch ,
drop column result ;

* To change a Table. (Rename a attribute)

Syntax :

alter table table-name change attribute-name
new-attribute-name datatype(size);

e.g - alter table Student change address location varchar(55);

desc Student;

* To ~~change~~ Rename a Table

Syntax :

alter table table-name rename newtable-name;

e.g - alter table Student rename Computer;

desc Computer;

3. drop Command

* To delete the entire table.

Syntax:

drop table table-name;

e.g - drop table Computer;

* To drop the database
drop database database-name;

4. Rename Command

Syntax:

rename table old-table-name to new-table-name;

e.g - rename table Computer to Student;

desc Student;

* To rename database
rename database old-database-name to new-database-name;

5. Truncate Command

(Used for delete all the data from the database)

Syntax:

truncate table table-name;

Note: The structure will remain as it is.

Integrity Constraints

Datatype

float(5,2) = 435.87

1) Number

int (10 digit)	numeric	number(5) = 435
float(precision, scale)	double	number(5,0) = 435
boolean	bit	number(5,2) = 435.87
bigint (19 digit)		number(5,1) = 435.8
smallint		number(4,1) = 435.8
		number(4,2) = Error
		number(2,0) = Error
		number(6,3) = 435.87

2) String

char
 varchar
 symbols
 blob (for image, file)

3) Date [YYYY-MM-DD]

datetime 'YYYY-MM-DD hh:mm:ss'

timestamp

time 'hh:mm:ss'

year

date 'YYYY-MM-DD'

current date & time

DML (Data Manipulation Language)

- 1) ~~to~~ insert command
- 2) update command
- 3) delete command

* 1) Insert Command

* To ^{insert} values into ~~field~~ a single row/tuple

Syntax:

insert into table-name values (value1, value2, ...
valuen);

=> insert into student values (15, 'Name', 20);

create table emp (Eid int, J-date date, J-time time, dt datetime);

~~=> insert into emp (101, '2024-12-10', '10:37:11',
'2024-12-10 10:37:11');~~

=> insert into emp values (101, '2024-12-10', '10:37:11',
'2024-12-10 10:37:11');

* To insert value into multiple tuple

Syntax:

insert into table-name values (value1, value2, ... valuen),
(value1, value2, ... valuen),
...
(value1, value2, ... valuen);

=> insert into ~~8~~ student values (1, 'A', 18),
 (2, 'B', 17),
 (3, 'C', 18);

* To show data in yield

Syntax:

select * from table-name;

=> select * from student;

Student		
Rollno	Name	age
1	A	18
2	B	20
3	C	18

* To insert values in table in random sequence in a single tuple.

Syntax:

insert into table-name (att-1, att-2, ... att-n)
 values (value1-of-att1, value2, ... valuen);

=> insert into student (Name, Rollno, age)
 values ('A', 1, 18);

* To insert values in table in random/specific sequence in a multiple tuple.

syntax:

```
insert into table-name (att1, att2, ..., attn)
values (value1-of-att1, value2, ..., valuen),
      (value1, value2, ..., valuen),
      ~~~~~
      (value1, value2, ..., valuen);
```

```
=> insert into student (age, rollno, Name)
values (18, 1, 'A'),
      (20, 2, 'B'),
      (18, 3, 'C');
```

```
=> insert into student (rollno) values (4);
```

ii) > Update Command

Used to update the data on the ~~old~~ table.

syntax

update tablename set
attribute_name = value. where condition;

=> update Student set Name = 'S' where Rollno = 1;

Student		<u>update</u> →	Student	
Rollno	Name		Rollno	Name
1	A		1	S
3	B		3	B

=> update Student set Name = 'S' ;

Student		<u>update</u> →	Student	
Rollno	Name		Rollno	Name
1	A		1	S
3	B		3	S

Syntax.

update tablename set attribute_name = attribute_name
+ 1;

⇒ update student set SNO = SNO + 1 ;

student

SNo	Rollno	Name
0	1	A
1	2	B
2	3	C
3	4	D

update

0+1

1+1

2+1

3+1

Student

SNo	Rollno	Name
1	1	A
2	2	B
3	3	C
4	4	D

iii) Delete Command

Syntax :

delete from table-name where condition;

⇒ delete from student where Rollno = 2 ;

Student

Rollno	Name
1	A
2	B
3	C

delete →

Student

Rollno	Name
1	A
3	C

Su

Syntax

delete from table-name ;

=> delete from ~~to~~ Student ;

Student

Rollno	Name
1	A
2	B

delete →

Student

Rollno	Name
--------	------

Set operator

First

Rollno	Name
1	A
2	B

Second

Rollno	Name
2	B
3	C

~~For~~ ~~union~~ select command

* select * from table-name ;

* ~~set~~

select * from second ;

Rollno	Name
2	B
3	C

* select attribute-name from table-name;

=> select Rollno from First;

Rollno
1
2

* select att1, att2, ..., Attn from table-name;

Select Name, Rollno from Second;

Rollno	Name

* select attribute-name from table-name
where condition;

select Name from First where Rollno = 2;

Name
B

i) Union

Syntax:

select * from table-1 union select * from
table-2;

=> select * from First union select * from Second;

Rollno	Name
1	A
2	B
3	C

ii) Union all

Syntax :

select * from table1 union all select * from table-2;

=> select * from First union all select * from Second;

Rollno	Name
1	A
2	B
2	B
3	C

iii) intersect

Syntax :

select * from table-1 intersect select * from table-2;

=> select * from First intersect select * from Second;

Rollno	Name
2	B

iv) minus / except

syntax :

select * from table1 except select * from table2;
or, select * from table1 minus select * from table2;
⇒ select * from "first" except select * from Second;

Rollno	Name
1	A

select * from Second except select * from First ;

Rollno	Name
3	C

Create Primary Key -

Primary key includes only one attribute.

Syntax

```
create table table-name (Att-1 datatype(size),
                        Att-2 datatype(size),
                        _____
                        Att-n datatype(size),
                        primary key (Att-name));
```

Primary key includes multiple attributes.

Syntax.

```
create table table-name (att1 datatype (size),
                        att2 datatype (size),
                        _____
                        att-n datatype (size),
                        primary key (att-2, att-3, ... att-n));
```

e.g- create table student (rollno int, name char(30),
address varchar(40),
primary key (rollno));

* to create unique key.

syntax:

```
create table table-name (att-name datatype(size)
                        unique key ;
```

* To add primary key on an existing table.

syntax:

alter table table-name add primary key (att-name);

eg- alter table student add primary key (rollno);

To create a primary key

syntax:

create table table-name (att-1 datatype(size), not null
primary key, att-2 datatype
(size), ..., att-n datatype(size));

eg- create table student (rollno int not null primary key,
name char(15), address varchar(15)
);

* To drop a primary key

syntax:

alter table table-name drop primary key att-name;

eg- alter table student drop primary key rollno;

To add foreign key.

Syntax:

```
create table table-name (att-1 datatype (size),
                        att-2 datatype (size),
                        = = = = =
                        att-n datatype (size),
                        foreign key (att-name-from-child-table)
                        references base-table-name
                        (att-name-from-base-table));
                        (P.K)
```

Base table

Student

Rollno	Name	Address

child table

(F.K)

Course

Sid	Cid	CName
(		

```
create table Student (Rollno int, Name char(15),
                      Address(20) varchar(20),
                      primary key (Rollno));
```

```
create table Course (Sid int, Cid int, CName char(15),
                    primary key (Cid),
                    foreign key (Sid)
                    references Student (Rollno));
```

* aggregate function

to Predefined function in DBMS to find specific value or information/detail from a relation.

i) max()

- ~~can be~~ gives the maximum value from the attribute
- attribute should be of int or float datatype.

ii) min()

- gives the minimum value from the attribute
- att. should contain only int or float type data.

iii) count()

- gives the total number of tuples in the table.

iv) sum()

- gives the summation of the values in the att., which contains only int or float datatype

v) avg()

- gives the average value of the entire values of the att. (contains only int or float)

Student

Rollno	Name	Sub	Marks
1	A	2	10-
2	B	3	25-
3	A	2	30-
4	C	5	10-
5	B	2	20-
6	A	3	25-
7	A	5	30-
8	A	3	10-
9	B	2	25-
10	C	4	30-

Count()Syntax

① select count(*) from table_name;

⇒ select count(*) from student;

10

② select count(att_name) from table_name;

⇒ select count(Name) from Student;
 count counts the tuple (excepting the null values)

— 10

③ select count (distinct att-name) from table-name;

⇒ select count (distinct Name) from Student;

3 — (A, B, C)

sum()

Syntax

① select sum (att-name) from table-name;

⇒ select sum (Marks) from Student;

= 215

② select sum (distinct att-name) from table-name;

⇒ select sum (distinct Marks) from Student;

= 85 (10 + 25 + 30 + 20)

average()

Syntax

① select avg (att-name) from table-name;

⇒ select avg (Marks) from Student;

= 21.5

② select avg(distinct att-name) from table-name;

⇒ select avg(distinct Marks) from Student;
= 21.25

min()

Syntax

select min(att-name) from table-name;

⇒ select min(Marks) from Student;
= 10

max()

Syntax

select max(att-name) from table-name;

= select max(Marks) from Student;
= 30

Logical Operator

1 AND

Employee

2 OR	E-id	E-name	E-city	E-country
3 IN	101	A B	ABC	MNO
4 NOT	102	C D	ABC	MNO
5 LIKE (%)	103	A E	ABC	MNO
6 SPACE (-)/underscore	104	A D	PQR	MNO
7 BETWEEN	105	F G	PQR	MNO
	106	H I	XYZ	MNO

AND

Find the records of employee ^{name & city} from the Employee table who are located in 'PQR' and belongs to 'MNO'.

Syntax:

select att-name1, att2... attn from table-name
where condition;

⇒ select E-name, E-city from Employee;

⇒ select * from Employee where E-city = 'PQR' and
E-country = 'MNO'.

OR

Syntax:

```
select att1, att2, ..., attn from table-name
      where condition1 or condition2
      ..... or condition-n ;
```

- Q. Find the details of employee from employee table who are either from 'ABC' or 'MNO' as their country.

```
select * from Employee where E-city = 'ABC'
      or E-country = 'MNO' ;
```

IN

Syntax:

```
select att1, att2, ..., attn from table-name
      where selected-att in ('value1', 'value2',
      ..., 'valuen');
```

- Q. Find the records of employee from employee table who are located in either 'PQR' or 'XYZ'.

```
select * from E-name from Employee
      where E-city in ('PQR', 'XYZ');
```

LIKE (%) (used for match characters)

Syntax:

Pattern	meaning
a%	match string that starts with 'a'
%a	match string that ends with 'a'
a%t	match string that starts with 'a' & ends with 't'
%aaa%	match string that contain 'aaa' in between of a string it.
<u>a</u> %	match string that contain 'a' at the second position

Syntax

```
select att1, att2, ..., attn from table-name
where att-name like pattern;
```

Q. Find the records of employee from employee table whose city starts with the letter 'x'.

select * from Employee where E.city like 'x%';

100	HI	xyz	MNO
-----	----	-----	-----

Q. city name second last letter is 'B'.

select * from Employee where E.city like ~~'B%'~~;
'B-%'; '%B-';

101	AB	ABC	MNO
102	CD	ABC	MNO
103	AE	ABC	MNO

NOT

Syntax:

select att1, att2, ..., attn from table_name
where not condition;

Q. Find the records of employee from employee table whose city name doesn't start with 'P'.

select ~~E.city~~ * from Employee where ~~not~~ E.city not like 'P%';

Find the records of employee from Employee table who doesn't belong to 'XYZ'.

`select * from Employee where e-city not
not like e-city = 'xyz';`

space ((-) / underscore)

Syntax:

`select att1,att2,...,attn from table_name
where att_name like '%_ _ _ _ _ %';`

- shows the value from attribute which contain — no. of characters.

• Between

Syntax:

select att1, att2, ..., attn from table-name
where att-name between
value1 and value2;
range.

Q. Find the records of employee from Employee table whose E-id is in between 102 to 104.

select * from Employee where E-id between 101 and 104 ;

AS

Syntax:

select ~~function~~ (attribut name as

select attname as new-att-name, att2-name as new-att name from table-name;

⇒ select count(*) from Employee;

count (*)
6

select count(*) as total-employee from Employee;

total-employee
6

⇒ select E-id as Emp-id , E-name as Emp-name
from Employee;

Emp-id	Emp-name
101	AB
102	CD
103	AE
104	AD
105	FG
106	HT

Clause

29/01/25

- 1) where clause
- 2) group by clause
- 3) having clause
- 4) order by clause (asc./desc)

① GROUP BY clause

- It is used for organizing similar data into groups
 - Can be used with aggregate functions also.
- Syntax :

select att1, att2, ..., attn from table-name
group by att-name order by
att-name ; optional

Emp					
E_id	E_name	E_age	E_address	E_salary	Dept-id
1	A	21	Ranchi	2000	100
2	B	23	JSR	1500	100
3	C	22	JSR	2000	100
4	E	24	Goo	6500	101
5	D	22	Goo	8500	101
6	F	25	Patna	4500	102
7	G	24	Delhi	10000	102

ORDER BY

- Used ~~for~~ to fetch the data on a specific order.
- Can be applied on numeric and as well as on alphabetic value.
- By default it fetch data in ascending order.
- It also can be used in multiple column in only one query/command.

Syntax:

```
select att1, att2, ..., attn from table-name
order by att-name asc/desc;
```

Find the details of employee in ascending order by salary

~~select E-salary from Employee order by~~

select * from Employee order by E-salary;

E-id	E-name	E-age	E-address	E-salary	Dept-id
2	B	23	JSR	1500	100
1	A	21	Ranchi	2000	100
3	C	22	JSR	2000	100
6	F	25	Patna	4500	102
5	E	24	Goa	6500	101
5	D	22	Goa	8500	101
7	G	24	Delhi	10000	102

Find the details of employee in descending order by name.

select * from emp order by E-name desc;

E-id	E-name	E-age	E-address	E-salary	Dept-id
7	G	24	Delhi	10000	102
6	F	25	Patna	4500	102
5	E	24	Goa	6500	101
5	D	22	Goa	8500	101
3	C	22	JSR	2000	100
2	B	23	JSR	1500	100
1	A	21	Ranchi	2000	100

Find the no. of employees in each department.

select Dept-id, count(*) as no-of-emp from
Emp group by Dept-id;

Dept-id	no-of-emp
100	3
101	2
102	2

Find the sum of salaries from each department.

select Dept-id, ~~sum(*)~~ as ~~sum~~ sum(e-salary)
as sum-of-salary from Emp
group by Dept-id;

Dept-id	sum-of-salary
100	5500
101	15000
102	614500

Find the avg average of salaries from each department.

select Dept-id, avg(e-salary) as average-salary
from Emp group by Dept-id;

order precession of clauses

select

from

where

group by

having

order by

Find the no. of employee in each department.
(no. of employee in as ascending)

```
select Dept-id, count(*) as no-of-emp from  
Emp group by Dept-id  
order by count(*) ;
```

Find the minimum and maximum salary of
employee from each department.

```
select Dept-id, max(E-salary) as max-salary,  
min(E-salary) as min-salary from  
Emp group by Dept-id ;
```

HAVING Clause

Syntax:

```
select att1, att2, ..., attn from table-name
      group by att-name
      having condition ;
```

- Q. Find the no. of employee in each department where no. of employee is greater than 2.

```
select Dept-id, count(*) as no-of-emp from
      Emp group by Dept-id
      having count(*) > 2 ;
```

- Q. Find the department for which the sum of salary of employees is more than 18000.

```
select Dept-id, sum(E-salary) as total-salary from
      Emp group by Dept-id having
      sum(E-salary) > 18000 ;
```

Find the no. of employees in each department location wise

```
select dep E-address, Dept-id, count(*) as
total-employee from emp
group by Dept-id, E-address;
```

Join

outer join Inner join

├ left (LEFT JOIN) (INNER JOIN)
├ Right (RIGHT JOIN)
└ full (FULL JOIN)

INNER JOIN

syntax :

```
select att_from_both_table from table-1
inner join table-2
on table-1.Att-name = table-2.Att-name;
```


Teacher		
ID	Name	Salary
1	A	5700
2	B	4000
3	C	6000
4	D	5000
5	E	5500

Course		
C-id	P-id	C-Name
101	1	PP
101	3	CHMA
201	4	CN
201	5	DBMS

Q. Retrieve the C-id, P-id, Name & salary from Teacher & Course.

select Course.C-id, Course.P-id, Teacher.Name,
Teacher.Salary from Teachers
inner join Course on ~~Course.P-id = Teacher.ID~~;
on Course.P-id = Teacher.ID;

C-id	P-id	Name	Salary
1	1	A	5700
1	3	C	6000
2	4	D	5000
2	5	E	5500

* Return only matching value,

Q. Retrieve the C-id, ID, P-id, Name, Salary from Teacher and Course relation where ID > P-id.

select course.C-id, Teacher-ID, Course.P-id, Teacher.Name,
Teacher.Salary from Course inner join
Teacher on Teacher.ID > Course.P-id;

PYQs

Q. Write a command to create table student (rollno, student-name, branch, class, DOB, city, contact-no) and ~~write~~ queries for following:

- i) insert 3 row into the table having rollno, 4, 6, 8.
- ii) save the data then insert one row into the table having rollno. 5.
- iii) undo the insertion of last inserted row
- iv) Delete one row whose rollno is 4.
- v) Update the rollno 8 to 9.

DCL

(Data Control Language)

— Gives the privilege to another user.

(1) grant command

- The main user (i.e. root) gives the permission to perform DDL/DML/DQL commands to the other user.
- The main user (i.e. root) can also grant the privilege to other user ~~also~~ to grant privilege (that he have) for an other user. (using with grant option)

e.g. → root ^{with} ~~grant~~ _{option} → user1 ~~grant~~ _{privilege} → user2

(2) Revoke command

- It takes back the ~~all~~ privilege that is/are given to the user.

* To create user

Syntax:

create user username identified by 'password';

=> create user CSE identified by 'CSE-3rd-sem';

* To show a user

Syntax:

select user, password from mysql.user;

=> ~~create user~~

select user, password from mysql.user;

show users;

```
mysql -u root -p
Enter password:
mysql> show users,
creat user CSE
identified by '3sem';
```

>quit; /exit;

```
mysql -u CSE -p
Enter password: 3sem
mysql> show databases;
>quit;
bye
```

* To grant privileges to the user.

Syntax:

grant command1, com2, ..., comn on
table-to-be-accessed to user-name;

e.g. - grant select on student to CSE;

mysql -u root -p

Enter password:

mysql> create user CSE identified by '3sem';

mysql> show users;

mysql> show databases;

mysql> use college;

mysql> show tables;

mysql> grant insert, select on student to CSE;

mysql> quit;

mysql -u CSE -p

Enter password: 3sem

mysql> use college;

mysql> select * from student;

- * To access ~~entire~~ all tables in database.
- * To grant all privileges to the user.

syntax:

grant all privileges table_name to username;

e.g. grant all privilege student to CSE;

- * To access all tables in database by the user.

syntax:

grant privilege1, privilege2, ..., privilegn on
database-name.* to username;

e.g. → grant select on college.* to CSE;

⇒ grant all privileges on college.* to CSE;

- * To grant privilege to user to access all the databases.

syntax:

grant privilege1, privilege2, ..., privilegn on
*. * to CSE;

- * To access all databases, all tables with all privileges.

grant all privileges on *. * to CSE.

with grant option

A user (except the root) can also grant privilege (that the user have only) to another user.

Syntax:

```
grant privilege1, privilege2, ..., privilen on  
database.name to user-name with  
grant option;
```

Revoke

syntax:

```
revoke privilege1, privilege2, ..., privilen on table.name  
from user-name;
```

e.g- revoke update on Student from CSE;

- revoke all privileges on Student from CSE;

Drop a user

Syntax:

drop user user-name;

Rename username

Syntax:

rename user old-user-name to new-user-name;

* To show total grants on users.

Syntax:

show grants on table table-name;

A. DDL

- i create
- ii alter
- iii rename
- iv drop
- v truncate

Alter Topic

- add
drop
modify
change
rename

B. DML

- i insert
- ii update
- iii delete

User

- creat user
drop user

C. DQL
select

Aggregate function

- min()
max()
sum()
count()
avg()

D. DCL

- i> grant
- ii> revoke

Key

- primary key
foreign key
unique key

E. TCL

- i commit
- ii rollback
- iii savepoint
- iv release savepoint

View

- creat
drop
insert data using view command
update data using view command
delete data using view command
update view name
create /replac

Set operator

union

union all

intersect

intersect all

minus / except

Clause

where

having

group by

order by

~~as~~ as

~~like~~

Logical Operator

and

or

not

like

between

in

Join

inner join

left

right