29/05/25

Page No.
Date

SQL (Structured Query language) :—

It is a domain Specific language used to manage & manipulate RDBMS or relational database.

key functions :

(i) Data Query : [redacted], (DQL)
—> Select.

(ii) DDL :— Data Definition language
—> Create, Drop, alter.

(iii) DML :— Data manipulation language.
—> insert,
—> Delete,
—> update.

(iv) DCL :— Data control language
—> Grant } permission to user
—> Revoke

(v) TCL :— Transaction control language.
—> Start transaction
—> commit
—> Rolleback.

Page No.
Date

* SQL Data types :→

Numeric :→ INT, FLOAT, DECIMAL

character/String :→ char(n), varchar(n).

Date/Time :→ Date, Time, DateTime

Boolean :→ True, False.

* Command's

Database {
- Create Database db_name ;
- Drope Database db_name ;
- Create Database if not exist's dbname ;
- Drope Database if exist's dbname.

Use dbname ;

Table {
- Creat Table tb_name (id int primary key,
  name varchar(50),
  Age int not null) ;
- Delete Table tbname ;

→

Insert

Insert Into tb_name value(1, 'Neeraj', 32)

~~Insert column1, column2 From tb name;~~

Insert into tb_name (column1, column2)
values (value1, value2), (v3, v4);

Show

show databases;
show tables;

select * table_name;
select * from tb_name;

Q create a Database for your company

Step(i) create table inside this DB to store employee info.
emp (id, name, salary)

Step(ii) Add the following info:

1. 'Adam', 2500
2. 'Neeraj', 2300
3. 'Caleay', 4000

Step(iii) Select & view all your table data.

→ CREATE DATABASE ~~DB Ba~~ XYZ ;
USE XYZ ;

→ CREATE TABLE EMP(

id INT primary key ,
Name varchar(50) ,
Salary FLOAT);

→ INSERT INTO EMP (id, name, salary)
values (1, 'Adam', 2500),
(2, 'Neeraj', 2300),
(3, 'Carey', 4000) ;

→ select * From Emp ;

Page No.
Date

(*) Key's :— key's are attributes used to uniquely identify records in a table & establish relationship's b/w tables.

(A) Primary key :— A single attribute or combination of attributes, that uniquely identifies each row in a table.

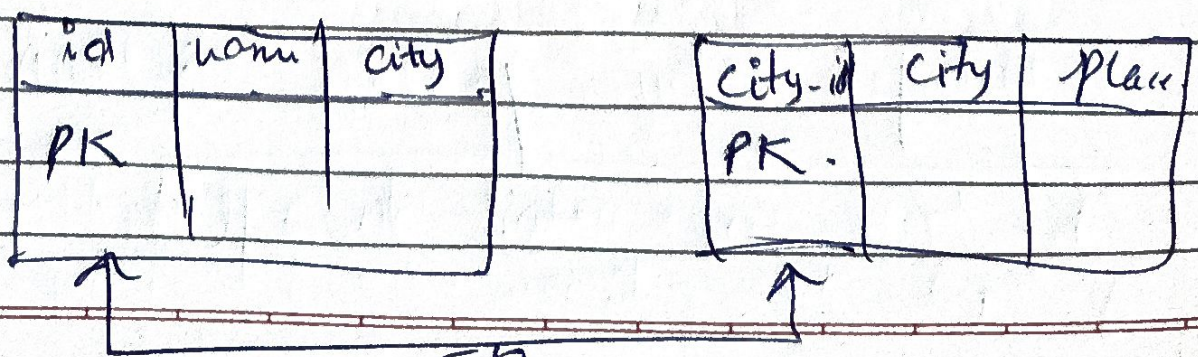
→ It should be unique, No duplicate's.
→ Cannot be null
→ only one primary key column is present in a table.

ex. Customer ID is a PK in a customer table.

(B) Foreign key :— It is a column or set of column in a table, that refers to the primary key

→ It links records b/w tables.
→ It can multiple
→ It can have Duplicate & NULL.

ex. Student



Page No.
Date

Constraints :-

- ~~Primary key~~ → uniquely identifies ~~each record~~
- → Not Null → No Null should be there.
- → Unique → No duplicate
- → ~~Reference~~ check → validation
- → Default → set default value.

* Primary key :- →

```
id int primary key
```

```
create Table temp1 (
    id int not Null,
    primary key (id)
    );
```

Primary key

```
column1 int not Null ;
column2 int unique ;
Col1 int primary key ;
```

ex.

```
create Table temp1 (
    id int unique );
```

~~Insert INTO temp1 value (101);~~

Page No.
Date

Ex create table temp (

id int,
name varchar(50)
primary key (id, name));

Here combination of id, name is a primary key while id or name can be duplicate.

(X) Foreign key constraints :→

link two Relation's & prevent action's that destroy link's b/w tables.

create table temp (
cust-id int,
Foreign key (cust_id) reference
customer (id));

Default: salary int default 25000.

If no value inserted it will insert 25k by default).

Insert Into temp(id) values(101);
Insert Into temp(id, salary) value (102, 10K);

select * from temp ;

Check :- It limits the values allowed in a column.

Create Table city (
id int primary key,
city varchar(50),
age int
constraint age_check CHECK
(age >= 18 AND city = 'Delhi'));

Not compulsory

create Table city (
age INT check (age >= 18));

Insert Into temp(id) values(101);
Insert Into temp(id, salary) value (102, 10K);
select * from temp;

Check :- It limits the value's can allowed in a column.

Create Table city (
id int primary key,
city varchar(50),
age int
constraint age_check CHECK
(age >= 18 AND city = 'Delhi'));

Not compulsory

create Table city (
age INT check (age >= 18));

Page No.
Date

Basic commands :—

(i) select :—

—> select col1, col2 From Table_name;
(print specific columns).

—> Select * From Table_name;
(print entire table).

—> select Distinct city From Table_name
(print's only distinct city).

(ii) Where clause (conditions) :—

Select column's From Table_name
Where condition;

ex:

—> Select * From Student
Where marks ≥ 80;

—> Select * From Student
Where city = 'Mumbai';

—> operation's

Arithmatic (+, −, *, /, //)

Comparison (=, >, <).

Page No.
Date

logical operations (AND, OR, NOT, IN, BETWEEN)

→ Select * From Student
Where mark > 80 AND city = 'Mumbai';

→ Select * From Student
Where marks > 80 OR city = 'Mumbai';

→ select * From Student
Where marks Between 80 AND 90;

→ Where marks IN (80, 90)

→ Where city NOT IN ('Delhi', 'Mumbai');

* limit (only limited items are show)

eg.

```
select * From Student
Where age >15
Limit 3;
```

* Order By

select * From Student
ORDER BY city ASC;

ASC - Ascending
DESC - Descend

⊕ Aggregate function's :—

It perform calculation on a set of value's & return's a single value.

- → count()
- → MAX()
- → MIN()
- → SUM()
- → AVG().

{ select column. From Table
  Apply agg. fu. }

or select count(name) From Student

→ select Max(marks) From Student;

⊕ Group By clause :—

Group's row's that have the same value into summary row's.

It collect's data from multiple records and group's the result by one or more column.

we also used agg. function to group data.

→

```
Select city, count(roll_no)
From Student
Group by city.
```

```
select city, name, count(name)
From Student
Group By city, name;
```

```
select city, Avg(marks) From Student
Group By city;
```

Q) Write a Query to find avg marks in each city in ascending order.

⇒

```
select city, Avg(marks)
From Table_name
Group By city
order By city ASC;
```

Page No.
Date

Having clause :-

Similar to WHERE clause but also includes agg. fun.

ex

```
select city, count(name)
From student
Group By city
Having max(marks) >90;
```

* General Order ——

→ Select column
→ From Table_name.
→ where condition
→ Group By column
→ Having condition.
→ Order By colum ASC/DESC;

update :—

```
update Table_name
set col1 = value1, col2 = val2
where col1 = 'A';
```

It will update 'A' — by value 1 in the Table.

Delete :—

```
Delete From Student
where age > 24;
```

Page No.
Date

⊛ Foreign key

Department

id	course
101	Science
102	Maths
103	Physics

student.

id	name	dept-id
101	A	101
102	B	101
103	C	102

A, B → Science.
C → Maths.

Here dept-id → id (dep.)

F.K.

```
create Table dept (
    id int primary key,
    name varchar(50) );

create Table teacher (
    id Int primary key,
    name varchar(50),
    dept-id int,
    Foreign key (dept_id) references
        dept (id) );
```

Page No.
Date

(*) **ALTER** :- To make changes in column.

(A) **ADD column** ——

```
ALTER TABLE table_name
ADD column column_name datatype
constraint;
```

(B) **Delete column** ——

```
ALTER TABLE tb_name
Delete column column_name ;
```

(C) **Rename** :-

```
ALTER TABLE tb_name
Rename To New_Table;
```

(D) **Modify**

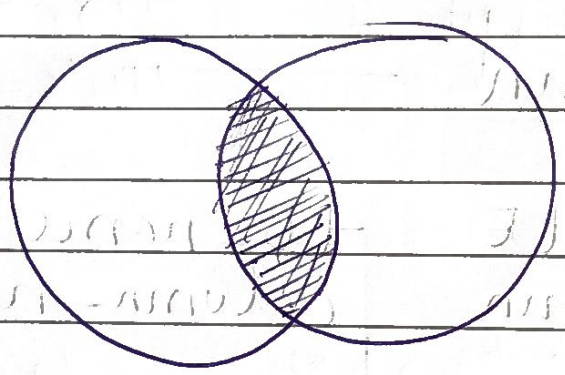
```
ALTER TABLE tbname
Modify col_name new_datatype new_const;
```

JOIN's :-

Used to combine rows of two or more table based on a related column b/w them.

(A) INNER JOIN :-

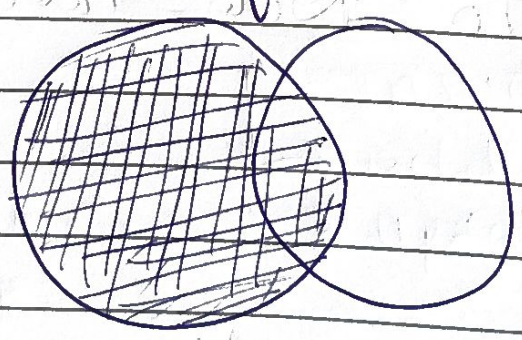
Return's only mateing rows b/w two tables.



```
select * From T-A
INNER JOIN T-B
ON T-A.id = T-B.id;
```

(B) Left JOIN's :-

Return's all rows from left Table & mateing row from right table.



```
select * From T-A
left JOIN T-B
ON T1.id = T2.id;
```

(C) Right JOIN's :-

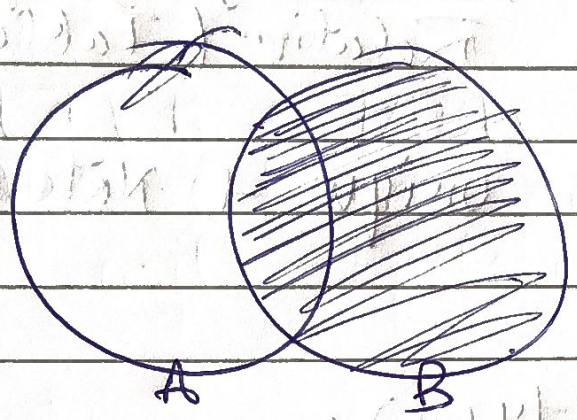
Return's all value from right table & mateing row's from left table.

INNER

Outer.

Left | Right | Full.

Non Matching row are get Null



(D) Full JOIN :—

Returns all row's from both tables.

Syntax :—

```
select * From TA.
FULL JOIN TB.
TA.id = TB.id;
```

```
select * From table A
INNER JOIN table B
ON student.id = course.id ;
```

(E) Cross Join :—

All possible combination of rows.

```
select * From Table 1.
CROSS JOIN table 2 ;
```