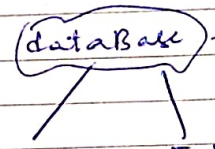
Introduction to SQL

SQL - structured Query language

* Main use of work with dataBase → This is Relation data Base collection of Tables



DBMS Tools
MYSQL
SQL server
Oracle
etc

* create data Base.
* Create Table
* Alter data Base
* Alter Table
* Reading data
* Updating data
* Delete database
* Delete tables
* Delete data

→ Using SQL commands.

SQL Commands

DATA DEFINITION LANGUAGE (DDL)

* Create
* Alter
* DROP
* Truncate

DDL → which update/change the table structure

DATA MANPULATION LANGUAGE (DML)

* select
* Update
* Insert
* Delete

DML → Used to changed the data inside the table.

DATA CONTROL LANGUAGE (DCL)

* Grant
* Revoke

DCL → Grant → permission to users
DCL → Revoke → Permission Remove From the User

TRANSACTION CONTROL LANGUAGE (TCL)

* commit
* Roll back

Which Undo the Recent Transaction.

Which will confirm the transaction

Query is simple called command.

Datatype's in SQL :

Type of Value → Data type

- Data
  - Numeric
    - Whole ——— Integers
    - Real Numbers ——— Float
    - Complex Numbers.
  - Strings ——— Char.
  - images
  - etc

Entity / Row / Tuple

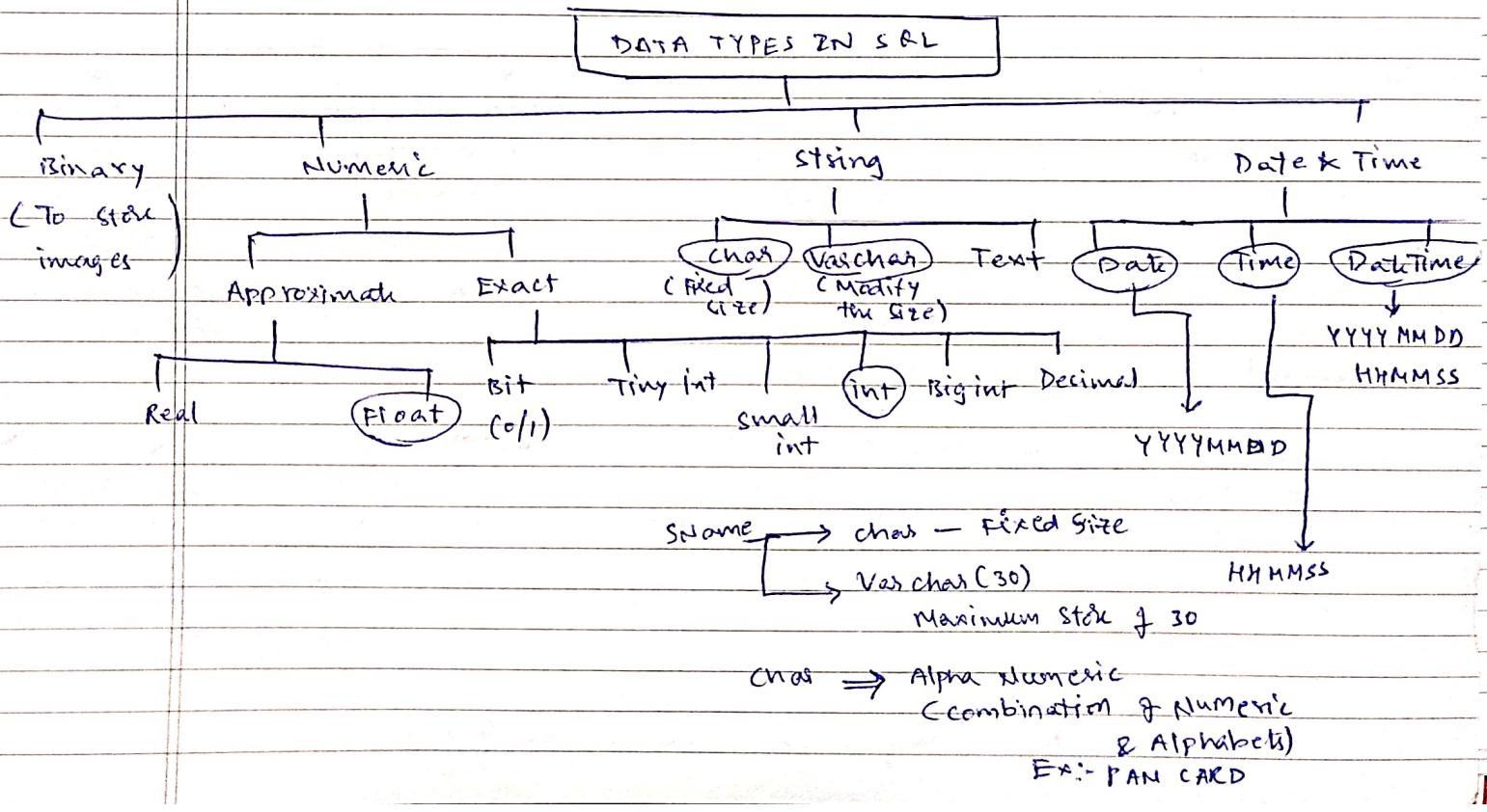
column / Attributes.

↳ hold some (values.)

↓

hold the same type —— Ex :-

	Student
Integer	* Sid – 101, 102
String – Group of char	* Sname – ABC
Float	* per – 10.2



* Float
* int
* char
* Varchar.
* Date
* Time
* Date Time

These types are supported all DBMS Tools.

↓

Ex:- Oracle
MYSQL
SQL server.
etc...

* DDL COMMANDS

*

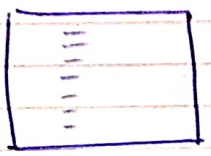
Create

Creating Table :-

```
Create table table_name (
        Attribute datatype,
        Attribute datatype,
        .
        .
        .
        );
```

Displaying Tables in Database

show tables;



Oracle :-
select * From table;

Ex:-

```
Create database Youtube;
Show databases;
```

⟶ o/p

Youtube

→ Use youtube;

Show tables; ——→ o/p :- Empty set.

(Check we not create table)

→ create table student (

std_id int,

Std-name varchar(30),

Std_branch varchar(20),

Std_ perc float);

→ Show tables; ——→ o/p

Tables in youtube
student

→ Select * from student;

o/p Empty set (Because we are not Add Anything)

→ desc student; (scheme & data base)

Field	Type	Null	Key	Default	Extra
Std_id	int	Yes		NULL	
Std-name	varchar(30)	"		"	
Std- branch	varchar(20)	"		"	
std_ perc	Float	"		"	

INSERT (DML)

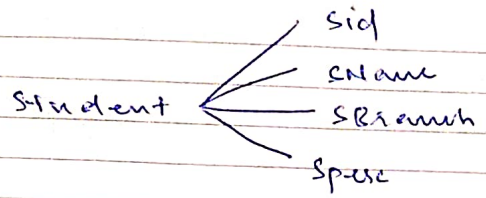
↳ (Manipulation)

* Used to insert data into tables;

Syntax :-

INSERT INTO TABLE_NAME (Attribute1 ~~value~~, Attribute2 ~~value~~ ....);

VALUES (Attribute_value1, Attribute value2 .....);



Sid	Sname	Sbranch	Spesc

*** Datatype string then enclosed with ' '

Insert Single Row

→ insert into student (sid, sname)
values (101, 'Aaaa');

Sid	Sname	Sbranch	Spesc
101	Aaa	NULL	NULL

→ insert into student values
(101, 'Asavind', 'cs', 96.0);

Sid	Sname	Sbranch	Spesc
101	Asavind	cs	96.0

→ insert into student values
(101, 'Asavind', 'cs');

o/p → Error
Because we have 4 columns we insert only 3 items.

Insert Multiple Rows

Syntax:-

INSERT INTO TABLE_NAME
values (Attribute-Value, Attribute2_value, ...)
(Attribute_value, Attribute2_value ....)
.
.
);

Ex: insert into student
values (103, 'Ram', 'CSE', 95)
(104, 'Siva', 'ECE' 95);

Sid	Sname	SBranch	Spesc
103	Ram	CSE	95
104	Siva	ECE	95

CONSTRAINTS

↓
Rules.

→ Primary key
→ NOT NULL
→ check
→ Default
→ Unique
→ Foreign key

• Primary Key

NOT NULL + No Duplicates

(Use not left the Blank default is NULL)

Ex:-
```
create table student (
    Sid integer,
    Sname varchar(30),
    Branch Varchar(20),
    Psc float, NOT NULL,
    Primary key (Sid));
```

* NOT NULL → Simply says Not kept Blank

* check → Before taking the data check condition done if true. then only add the data

* Default → we not give any value, then automatic take default value.

Ex:- create table student (
    sid integer,
    College Varchar(20) Default 'BEC');

* Unique → which is similar to primary key But Accept the NULL values, Do not Accept the Duplicates)

Ex:- contact ~~Email~~ Number integer Unique;

Foreign key

student (sid (PK), sname, cid (FK));

course (cid (PK), cname);

One table of PK is the FK of another table.

Ex:-

```
Create table course (
        Cid integer,
        Cname Varchar(20),
        Primary key (cid));

Create table student (
        Sid integer,
        Sname varchar(20),
        cid integer,
        primary key (Sid),
        Foreign key (cid) References course (cid));
```

12/01

ALTER COMMANDS

→ To change the schema / structure of database.

DDL

↓

Data Definition language

* Add column
* Delete column
* Modify datatype of existing column
* Set constraints like NOT Null, Primary key, Unique key etc...
* Remove constraints.

* Add column

- Syntax :-

```
alter table table_name
  add column_name datatype ;
```

* Drop column

Syntax :-

```
alter table table_name
  drop column column_name ;
```

* Modify Datatype

Syntax :-

```
alter table table_name
  modify column column_name datatype;
```

* ADD NOT NULL

Syntax :-

```
alter table table_name
  modify column_name datatype NOT NULL;
```

* ADD constraint Unique :-

Syntax :-

```
alter table table_name
  add constraint Unique (column_name);
```

* PRIMARY KEY CONSTRAINT

Syntax

```
alter table table_name
  add constraint primary key (column_name);
```

* ADD CONSTRAINT CHECK

alter table table_name
add constraint check (condition);

* DROP CONSTRAINT PRIMARY KEY

after table table_name
drop ~~constraint constraint name~~;
primary key;

* DROP CONSTRAINT UNIQUE

alter table table_name
drop index ~~constraint~~ column_name;

* DROP → is used to delete complete table from database.

Truncate → is used to do will delete data from the table.

* Drop → DDL - It deletes complete table from the database.
  → Syntax: drop table table_name;

* Truncate → DDL - It deletes complete data from table.
  Syntax :-
  truncate table table_name;

SELECT

↳ DML (Data Manipulation language)
Retrive Data

Syntax

projection

```
Select * From Table_name
where condition
Ordered by column-name
```

simple
compound

* → Display all the columns.

Particular Colums display

Syntax:-

```
Select col1, col2 From table_name
```

= , <> , > , < , >= , <= → Relational operators

Arranging in ascending / descending order

Compound → OR, AND, NOT

like → search for Pattern.

IN → search in multiple values

Between → search in a Range

ORDER BY → Arrange in Ascending / Descending

* Displays All elements

Syntax:-

```
Select * from table_name;
```

* Display Require items

```
Select col1, col2 ... from table_name;
```

* Without Repition Values (Display) → If use this (No repetition)

```
Select Distinct column1, column2, .....
  From table_name;
```

*

```
Select col1 , col2, ...
From table_name
where condition ;
```

operator	Description
=	Equal
>	Greater than
<	lesser Than
>=	Greater than or equal
<=	less than or equal
Between	Between a certain Range
like	Search for a pattern
In	To specify multiple possible values for a column.
<>	Not equal, some SQL versions use !=

Condition < Simple → Relational
Compound → logical

Using AND

```
Select Columns, Column2 ...
From Table_name
where condition1 AND condition2 AND condition3 ... ;
```

Using OR

```
select column1, Column2, ...
From table_name
where condition1 OR condition2 OR condition3 ...;
```

Using NOT

```
Select column1, Colum2, ....
From table_name
where NOT condition1;
```

* ORDER BY < Ascending → default
  Descending

Syn:- `select * from table_name ORDER BY column;`

→ Desc

Syntax:-

```
select * from table_name ORDER BY column DESC;
```

```
SELECT column11, column2 ....
From table_name
ORDER BY column1, column2, .... AESC | DESC;
```

* LIMIT

Syntax:-

```
Select column_name(s)
From table_name
where condition
LIMIT number;
```

If use this, we ~~cannot~~ can get number of values

* LIKE

```
SELECT column_name(s)
From table_name
Where column_name LIKE pattern;
```

search pattern

* IN

```
SELECT column_name(s)
from table_name
where column_name IN (value1, value2, ....);
```

print the

* BETWEEN

```
SELECT column_name(s)
FROM table_name
where column_name BETWEEN value1 AND value2;
```

UPDATE & DELETE

Update → Used to modify the existing data

→ DML command.

Syntax:

```
Update table_name
set column1 = value1, column2 = value2 ....
                 ↓                ↓
            (new value)      (new values)
where condition;
```

DELETE

→ Used to delete specific rows from table

→ DML

syntax:-

① Delete from Table_name
where condition;

} Deletes specific rows/Records

② Delete from table_name ⟶ Delete All Records/Rows

Update practice

```
create database youtube;

use youtube;

create table student (
    sid int,
    sname varchar(30),
    perc float );

insert into student values (
    1 , 'Aravind' , 99.0 );
```

~~Update Aravind~~

* `Update student set perc=95 where sname = 'Aravind';`

Update Multiple values

```
Alter table student
add Grade Varchar(20);

insert into student values (
    2 , 'Ravi' , 96, NULL );

select * from student;
```

sid	sname	perc	Grade
1	Aravind	95	NULL
2	Ravi	96	NULL

* Update student set Grade = 'First' where perc > 95 ;

* Update student set Grade = 'second' where perc < 95 ;

Select * from student ;

o/p →

Sid	Sname	perc	Grade
1	Aravind	95	second
2	ravi	96	first

* * * Update ~~more~~ values at a time ~~but~~ when the condition is true.

* * DELETE * *

→ Delete from student where Grade = "First" ;

Select * from student ;

o/p →

Sid	Sname	perc	Grade
1	Aravind	95	second

Delete

Syntax 1 :- delete from table_name where condition;

Syntax 2 :- delete from table_name;

Ex :-

Syntax 1

delete from student where Grade = "First";

select * from student;

o/p →

Sid	Sname	Perc	Grade
1	aravind	95	Second

Syntax 2 :-

delete from student;

select * from student;

o/p → Empty set.

Aggregate Functions

* count ( ) — which will give the count of values of an attribute.

  Syn : select count(attribute_name) from table_name;

* Sum ( ) — Numerical values → select sum(attribute_name) from tablename;

  Sum of all values of an Attribute

Attribute = column

* Avg() - Numerical values.
  - Average of all values.
  - Syntax → Select Avg(attributename) from table_name;
* Min() - Minimum value of an Attribute.
  - syntax → Select min(attributename) from table_name;
* Max() - which will give maximum value of an attribute.
  - Syntax → select max(attributename) from table_name;

** Practice

use youtube;

show tables; → o/p

Tables in Youtube
student

select * from student;

o/p →

Sid	Sname	psec	gender	Branch
101	hari	99	M	CS
102	ramya	90	F	EC
103	SUN	95	M	CS
104	SON	78	F	CS
105	SARA	88	M	CIVIL
106	YAMI	98	F	EEE

Group By & Having

These clauses will be used in SELECT command

↓

used to Retrieve data.

Group By used

* Group the rows based on columns.
* used Aggregate functions

Having

* Aggregate function.
* Conditions on result of Group By clause.

Practice

Group By clause

*

Syntax:

```
Select * from table_name group by column1, column2 ...;
```

```
use youtube;
Select * from student1;
```

o/p

sid	sname	perc	gender	Branch
101	hari	89	M	CS
102	ramya	90	F	EC
103	SNV	95	M	CS
104	SON	78	F	CS
105	seva	88	M	CIVIL
106	Yami	98	F	EEE

Ex:- * Select count (name) from Student;

o/p →

count (sname)
6

* Select count (Branch) from student;

o/p →

Count (Branch)
6

* Select ~~dist~~ count (distinct branch) from student;

o/p →

Count (distinct branch)
4

* Select count (*) from student;

o/p →

count (*)
6

* Select sum (perc) from student;

o/p →

Sum(perc)
548

* Select avg (perc) from student;

o/p →

Avg(perc)
91.33

* Select min (perc) from student;

o/p →

min(perc)
78

* Select max(perc) from student;

o/p →

max(perc)
99

* Select branch, count(sid) from student group by branch;

o/p →

branch	count(sid)
CS	3
EC	1
civil	1
EEE	1

* Select gender, count(sid) from student group by gender;

o/p →

gender	count(sid)
M	3
F	3

* Select gender, Avg(perc) from student group by gender;

o/p →

gender	Avg(perc)
M	94
F	88.66667

* Select gender, Max(perc) from student group by gender;

o/p →

gender	Max(perc)
M	99
F	98

* Select gender, min(perc) from student group by gender;

o/p

gender	Min(perc)
M	88
F	78

* select branch, max(perc) from student group by branch;

o/p →

Branch	max(perc)
CS	99
EC	90
Civil	88
EEE	98

* Select branch, min(perc) from student group by branch;

o/p →

Branch	Min (perc)
CS	78
EC	90
Civil	88
EEE	98

HAVING CLAUSE

Syntax:-

select * from table_name group by column1, column2, ...
.... having condition;

Ex:- Branches having only one student

select branch, count(branch) from student group by branch having count(sid) = 1;

o/p →

branch	count (branch)
EE	1
CIVIL	2
EEE	1

Branches having more than one student:

→ select branch, count(branch) from student group by branch having count(sid) > 1;

o/p →

branch	count(branch)
cs	3

→ Branches in which student having ~~maximum~~ minimum percentage more than (greater than) 70.

select branch, ~~count~~ min(perc) from student group by branch having min(perc) > 70;

o/p →

branch	min(perc)
cs	78
EC	90
CIVIL	88
EEE	98

Branches in which student having maximum percentage greater than 90.

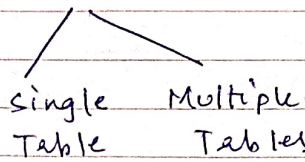
select branch, max(perc) from student group by branch having max(perc) > 90;

o/p →

branch	max(perc)
cs	99
EC	96

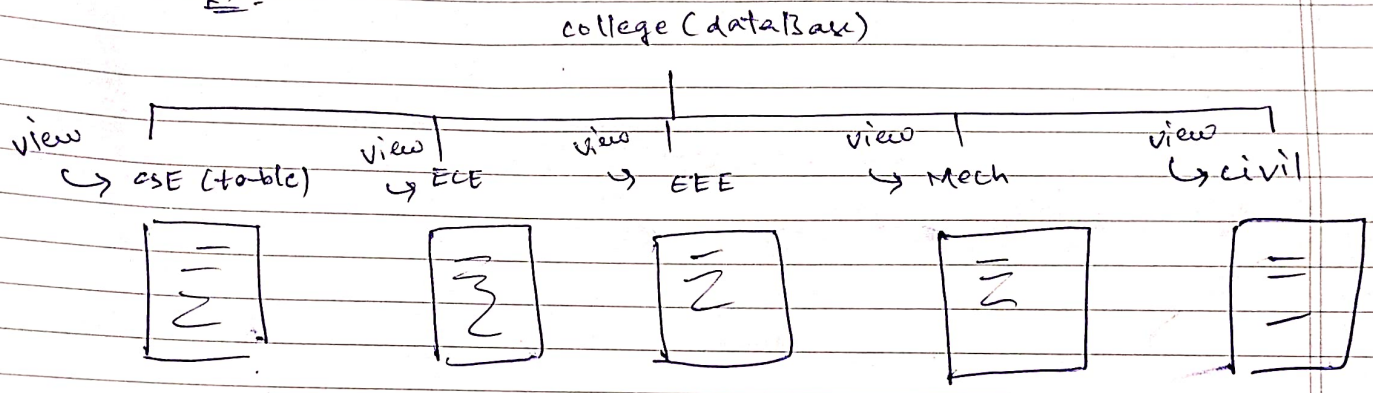
VIEWS

* Virtual Table constructed from existing Table with required Attributes.



* changes made in Table reflects on view
* changes made in view reflects on Table.

Ex:-



Syntax:-

→ Create view view_name AS
Select column_name1, column_name2 ... From Table1
where condition;

→ create view view_name AS
Select column_name1, column_name2 ... From Table1, Table2...
where Table1.column_name = Table2.column_name and
condition;

View delete

Syntax:- Drop ~~table~~ View table_name;

Ex:-

use youtube;

select * from student;

o/p →

sid	Sname	psec	gender	Branch
101	Hari	99	F	CS
102	Ramya	89	F	CS
103	Surya	90	M	EC

create view cs_students as
select * from student where branch = 'cs';

→ Select * from cs_students;

o/p →

sid	Sname	psec	gender	Branch
101	Hari	99	F	CS
102	Ramya	89	F	CS

* show tables

o/p → student
cs_students

* update ~~cs~~ students
set psec = '95' where sid = 103;

select * from student;

o/p →

Sid	Sname	Perc	gender	Branch
101	Gowri	99	F	CS
102	Ramya	895	F	CS
103	Surya	90	M	EL

select * from cs_student;

o/p →

sid	Sname	Perc	gender	Branch
101	Gowri	99	F	CS
102	Ramya	95	F	CS

Drop view cs-students;

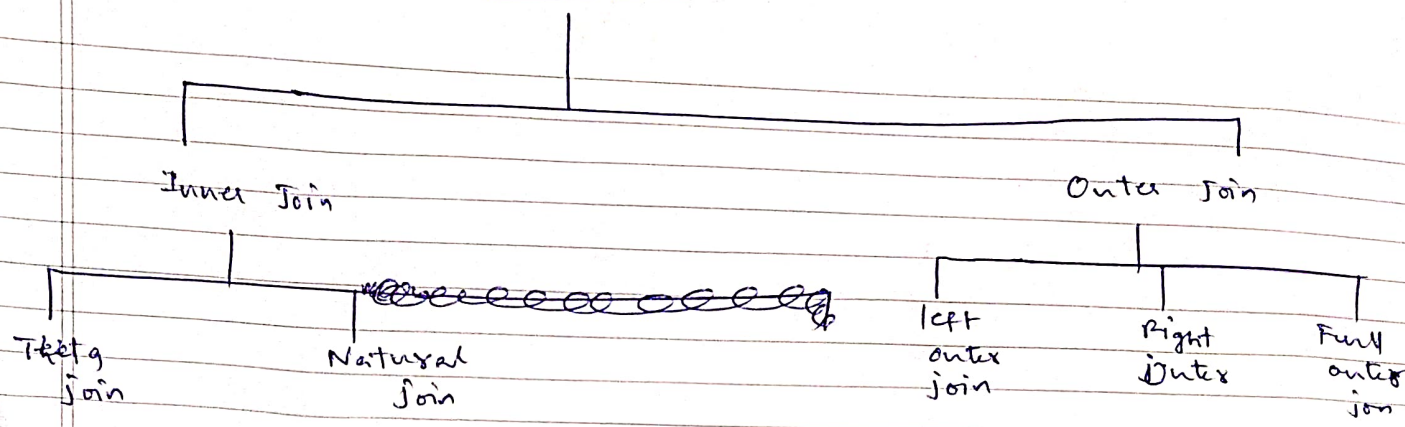
Show tables;

o/p →

Table
student

DBMS →

JOINS



Cartisian product :- (X)

↑ cartisian product

Rows / Records / Tuples.
columns / Attributes

Q table

D	E
1	c
2	d

P table

A	B	C
1	a	b
2	c	d
3	e	f

P X Q ⇒

A	B	C	D	E
1	a	b	1	c
1	a	b	2	d
2	c	d	1	c
2	c	d	2	d
3	e	f	1	c
3	e	f	2	d

~~Natural~~ join :-

condition (cartesian product) ⇒ join

Natural join :-

P →

A	B	C
1	a	b
2	c	d

Q →

A	D
1	d
3	e

P natural join Q.

→ Its ntg about common Attributes

A	B	C	D
1	a	b	d

* left outer join:-

P →

A	B
1	a
2	c

Q →

C	D
1	b
3	d

Table1 operation Table2

left → Must All columns are appear in the o/p

P left Outer join Q A=C

A	B	C	D
1	a	1	b
2	c	Null	Null

* Right Outer join:-

P right outer join Q A=C

A	B	C	D
1	a	1	b
Null	Null	3	d

Table1 operation Table2

Right

Must all columns are appear in the o/p.

* Full join (A=C)

A	B	C	D
1	a	1	b
2	c	Null	Null
Null	Null	3	d

* Inner join

P inner join Q. (A=C)

A	B	C	D
1	a	1	b

MySQL

select * From Table_name1 Join name Table_name2

ON Table name.columnname = Table name.columnname ;