GROUP BY

Students

RNO	Name	Gender	Marks	Attendance
1	a	m	210	60
2	b	m	235	80
3	c	f	245	90
4	d	m	205	55
5	e	f	200	75
6	f	f	250	85
7	g	f	215	92
8	h	m	230	72
9	i			

1) Find the average marks of the girl & Boy student in the class.

```
SELECT gender, AVG(marks)
FROM Students
GROUP BY gender.
```

AVG( ) → Aggregate fn.

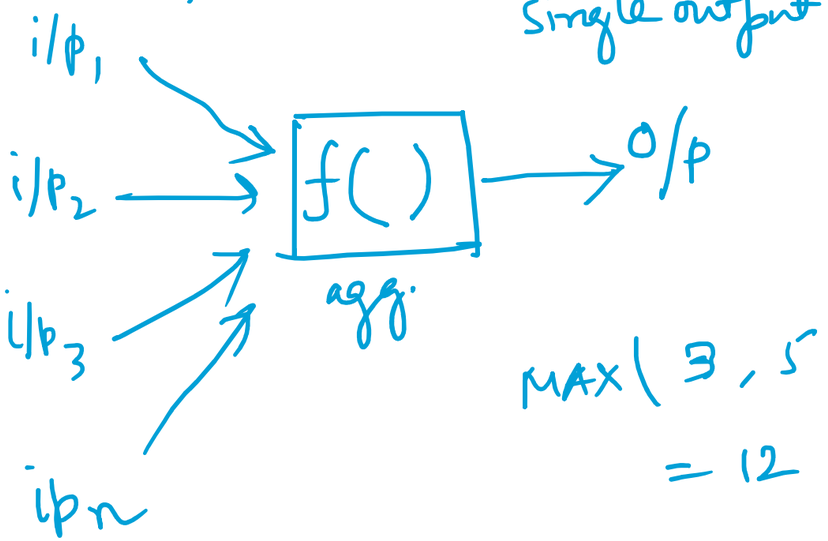
SUM( )

COUNT( )

MAX( )

MIN( )

multi-input



Gender	AVG(marks)
m	—
f	—

MAX(3, 5, 12, 10)

= 12

Employee

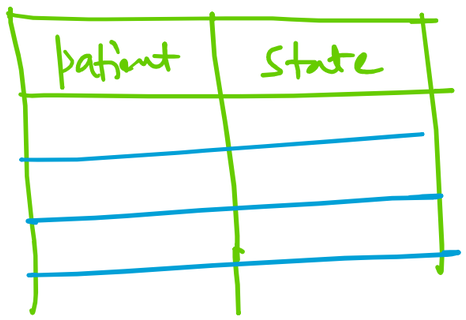
ID	name	Role	Salary
—	—	Eng.	—
—	—	Mang.	—
—	—	Sales.	—
—	—	Tax	—
—	—	Engs	—
—	—	Mi. Tax.	—

Find the AVG salary & count of employees in each role.

Role	Count	AVG(Salary)
Eng	—	—
Man	—	—
Tax	—	—
Sale.	—	—

```
SELECT role, COUNT(*), AVG(Salary)
FROM Employee
GROUP BY role.
```

SELECT ....
FROM
WHERE
GROUP BY
(HAVING)
ORDER BY
LIMIT
OFFSET

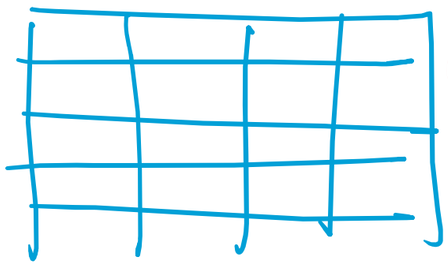


# of patients in Each state in descending order. Consider only those states where patients number is more than 30

SELECT . . . . , ...

FROM

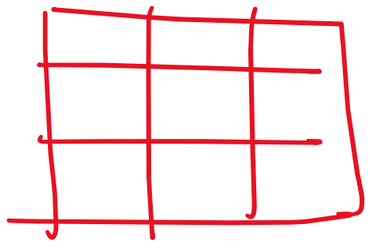
WHERE — Filter



Raw table.

GROUP BY

HAVING — to apply filter on grouped table



X

Grouped table

```
SELECT state, COUNT(patient) AS tot_count
FROM patients
GROUP BY state
HAVING tot_count > 30
ORDER BY tot_count.
```

% wildcard in string matching

players.

name	tot_score
Sachin	35000
Rahul	22000
Saurav	20000
Virat	23000
MS Dhoni	25000

LEFT(name, 1) = 'S'

✓ S(. . . .)

name = "S%"

name = "%l"

. l

name = "%ra%"

name = 'sachin' ✓

Sachin Tend

Sam

name = 'Sachin%"

%xyz

```
SELECT CONCAT ("My", " Name", " is", " Sourav")
```

"My Name is Sourav"

NULL values:- Null → absence of value.

allergies IS NULL → Give the records where allergy is NULL

allergies IS NOT NULL → " " " " " is not null

CASE-WHEN Statement

citizens

UID	name	Gender	Age	Adult
				yes
				no

age >= 18

age < 18

CASE WHEN age >= 18 THEN "yes"
ELSE "no"
END AS Adult → name of the derived column.

age	age_group
age < 18	"Not Adult"
60 >= age >= 18	"Working"
age > 60	"Senior Citizen"

```
SELECT UID, name, gender, age,
CASE WHEN age < 18 THEN "Not Adult"
     WHEN age BETWEEN 18 AND 60 THEN "Working"
     ELSE "Senior Citizen"
END AS age_group
FROM citizens.
```

Inner Join (JOIN)

Items.

ID	name	price
I1	CD	5
I2	CS	25
I3	Park	15
I4	5th	20
I5	Harry	90

primary key

Foreign key.

Orders

OID	Item	Qty
A1	I3 ✓	10
A2	I2	8
A3	I3 ✓	1
A4	I5	6

primary key

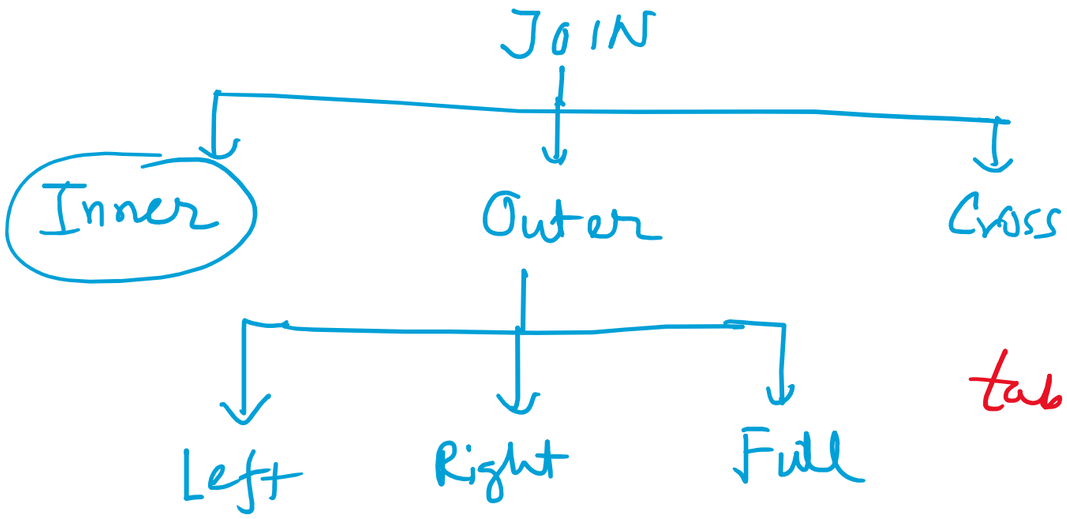
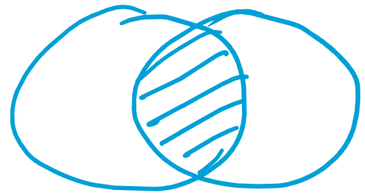


table . column

OID Item Name Total price

```
SELECT OID, Item, Name, Qty*Price AS Total_price
FROM Orders
JOIN Items
ON Orders.item = Items.id
```

Inner Join (JOIN)



Customers

ID	Name	Address	Birth_date
2			
3			
⋮			
1000			

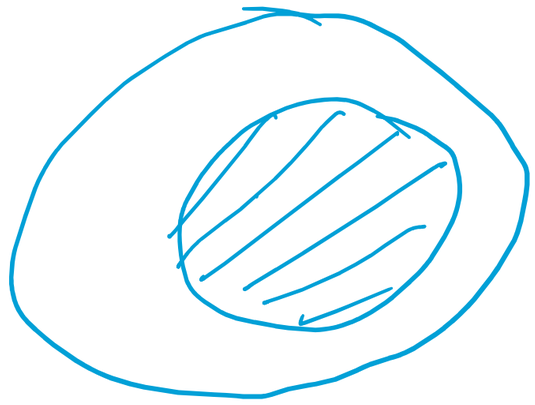
1000 UniqueID

Orders

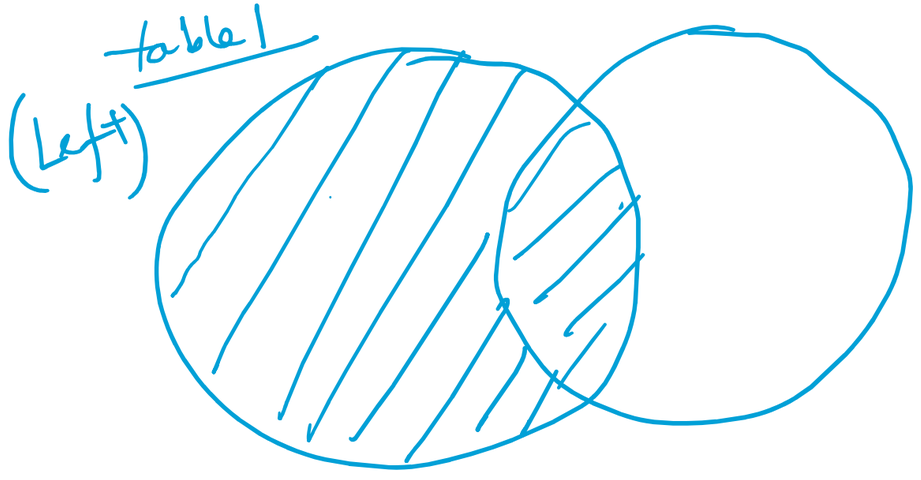
OrderID	C.ID	Price	Discount
	3		
	15		
	23		
	⋮		

UniqueID ≤ 1000

```
SELECT Orders.CID, Customer.name, , COUNT(*)
FROM Orders
JOIN Customers
 ON Orders.CID = Customers.ID
GROUP BY Orders.CID
ORDER BY Order.CID DESC
```

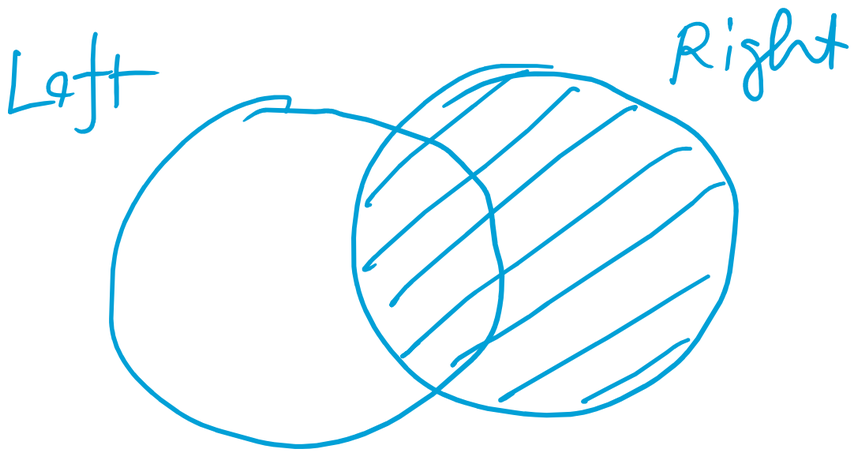


LEFT JOIN

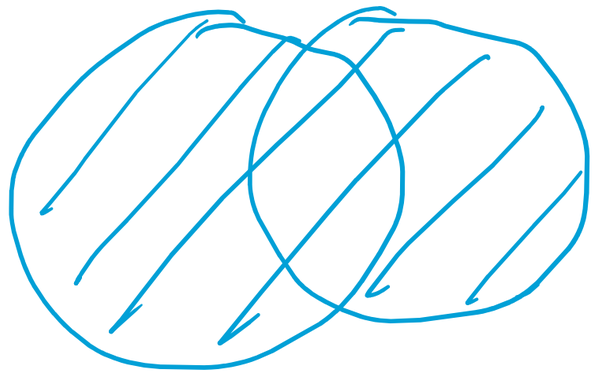


Resultant table will contain the information from table1 (left table) & the common information between table1 & 2 but will discard those which are present solely in table2 (right)

RIGHT JOIN



FULL JOIN



Examples of Joins

Employee

EC	MC	Salary	Dept
			Engg
			Sales
			Tax

2000

Dept

Dept	Head	Building
Engg		B1
Sales	1	B2
Tax	1	B4
Audit		B5
R&D		

Building

BuildID	Address	Rent
B1		
B200		

Dept	Emp
Sales	Sales
Tax	Tax
NULL	Audit
NULL	R&D

LEFT

CROSS JOIN

t1

A	B
1	α
2	β
3	δ

t2

C	D
100	5
200	6
300	7

t_1 JOIN t_2

Don't mention "ON" then it will do "CROSS JOIN"

A	B	C	D
1	α	100	5
1	α	200	6
1	α	300	7
2	β	100	5
2	β	200	6
2	β	300	7
3	δ	100	5
3	δ	200	6
3	δ	300	7