Class starts at 9:05 PM

Agenda :

→ Aggregates
→ Group By
→ Having
- count()
- min()
- max()
- avg()
- sum()

→ Till now whatever SQL Queries we have written, were to get few rows of the table. (Filtering out rows using ON and WHERE)

Sometimes, instead of getting few rows of the table, we have to get analysis of the entries of table.

Q1) Get the average psp of every batch of Scaler.

Batches

id	name

Students

id	name	batch_id	psp

Approaching this as a DSA problem:

Step 1: Iterate over the Join of Batches and Students.

Step 2: Use maps to store data of Students batch-wise.

Step 3: Get the Sum of psp / Sum of Students

To answer such type of Queries, we need the Aggregate Fn.

Count(), SUM(), MIN(), MAX(), AVG().

Behaviour of every Agg. F^n

Count

Students

id	name	age	batch_id
1	A	20	1
2	B	21	1
3	C	22	Null
4	D	23	2

Q) Give the count of students that have a batch?

Count → takes a lot of values and combines them into a single value equal to the number of elements in the list.

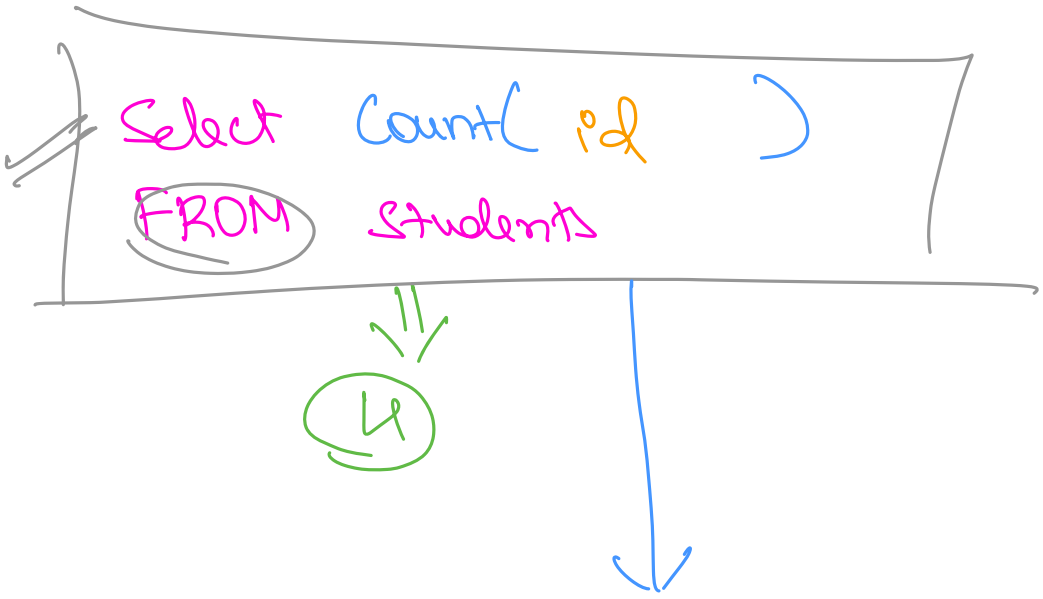
Eg Count (1, 2, 3, 5, 6) → 5

```
Select Count(batch_id)
FROM Students
```

Count (1, 1, Null, 2) ⟹ 3

All aggregate F^n ignore Null when computing their result.

what if I use id in count.




```
Select  Count(id)
FROM    Students
WHERE   batch_id IS NOT NULL
```

Students

id	name	age	batch_id
1	A	20	1
2	B	21	1
3	C	22	Null
4	D	25	2

the count of

Q) Give me those students that have a batch and their age is less than 23.

A)

```
Select  Count(batch_id)
FROM    Students
WHERE   age < 23
```

Students

id	name	batch_id
1	x	1
2	y	1
3	z	2

Batch

id	name
1	A
2	B
3	C

Q) Get count of students with batch_name 'A'

A)

```
select  Count(s.id)
FROM    Students  s
JOIN    batches   b
ON      s.batch_id = b.id
AND     b.name = 'A'
```

```
Select      Count(s.id)
FROM        Students  s
JOIN        batches   b
ON          s.batch_id = b.id
WHERE       b.name = 'A'
```

CODE

```
Select
FROM A
JOIN B
ON   cond^n 1
WHERE  cond^n 2
```

A = [{} ..... {} ----]

B = [{} ..... {} --]

ans1 = [ ]

```
for row1 in A:
    for row2 in B:                                   JOIN
        if row1 and row2 match ON cond":
            ans1.add( row1 + row2 (stitch))
```

ans2 = [ ]

```
for row3 in (ans1):          → virtual table.
    if row3 matches the WHERE cond":
        ans2.add[row3];
```

```
count_&_id = 0                  → filtered table
for row in (ans2):
    if row[&.id] is (Not Null):          ] Aggregate f"
        count_&_id += 1;
```

print (count_&_id) ;

Other aggregate f^n

→ you can print multiple aggregate f^n in the same query.

```
Select  count(S.id), sum(S.gpa), max(S.gpa)
        min(S.age)
FROM    Students.
```

Max (- - - - - -)
Min (- - - - - - -)

Both these functions can take a col^n which is comparable.

- int
- double
- Strings (lexicographic)

Average

Avg(1, 2, 3, ~~Null~~) →

→ $\frac{6}{3} = 2$ ✓

→ $\frac{6}{4} = 1.5$ ✗

Student

id	psp
1	1
2	2
3	3
4	Null

Select (Sum(psp) / count(psp)) = avg(psp) (avg(psp) = 2)

True? ✓

False?

Print

Select (Sum(psp) / count(id)) = avg(psp) (Sum(psp) = 6, count(id) = 4, avg(psp) = 2)

False ✓ $\frac{6}{4} \neq 2.$

Q) Tell me how many students are there?

students

id	name	batch_id
1	x	1
2	y	1
3	z	2

```
Select count(id)
FROM Students;
```

→ what if id could also be Null.

A)
```
Select count(*)
FROM Students;
```

```
count(*)
```

A = [{} {} .... {} ----]

B = [{} {} ---- {} ..]

ans1 = [ ]

```
for row1 in A:
    for row2 in B:                          JOIN
        if row1 and row2 match ON cond":
            ans1.add( row1 + row2 (stitch))
```

ans2 = [ ]

```
for row3 in (ans1):          → virtual table.
    if row3 matches the WHERE cond":
        ans2.add[row3]:
```

```
count_&_id = 0                          → filtered table
for (row) in (ans2):                                    ] Aggregate f"
    if 1 is (Not Null):
        count_&_id += 1;
```

```
print ( count_&_id) ;
```

* means any value

existing Query:

```
Select  count(*)
FROM    students
```

exactly same as.

```
Select  count(1)
FROM    students;
```

⇒ Output?

whatever we give inside the count(x) f^n
It will check for x is not Null.

count (s.id) → checks if s.id is not Null
count (batch_id) → checks if batch_id is not Null.

count(*)
count(1)
count('hello')
} all are same as *

Checks if 1 is Not Null for every iteration.
↳ Always true.

Select count(1)
Select count(2)
Select count('hello')
} Same as
Select count(*)

Which one is faster?

Select count(1) OR Select count(*)

Select count(1) is faster ⇒ No memory access.

Select count(*) ⇒ have to check for an actual values for all columns.

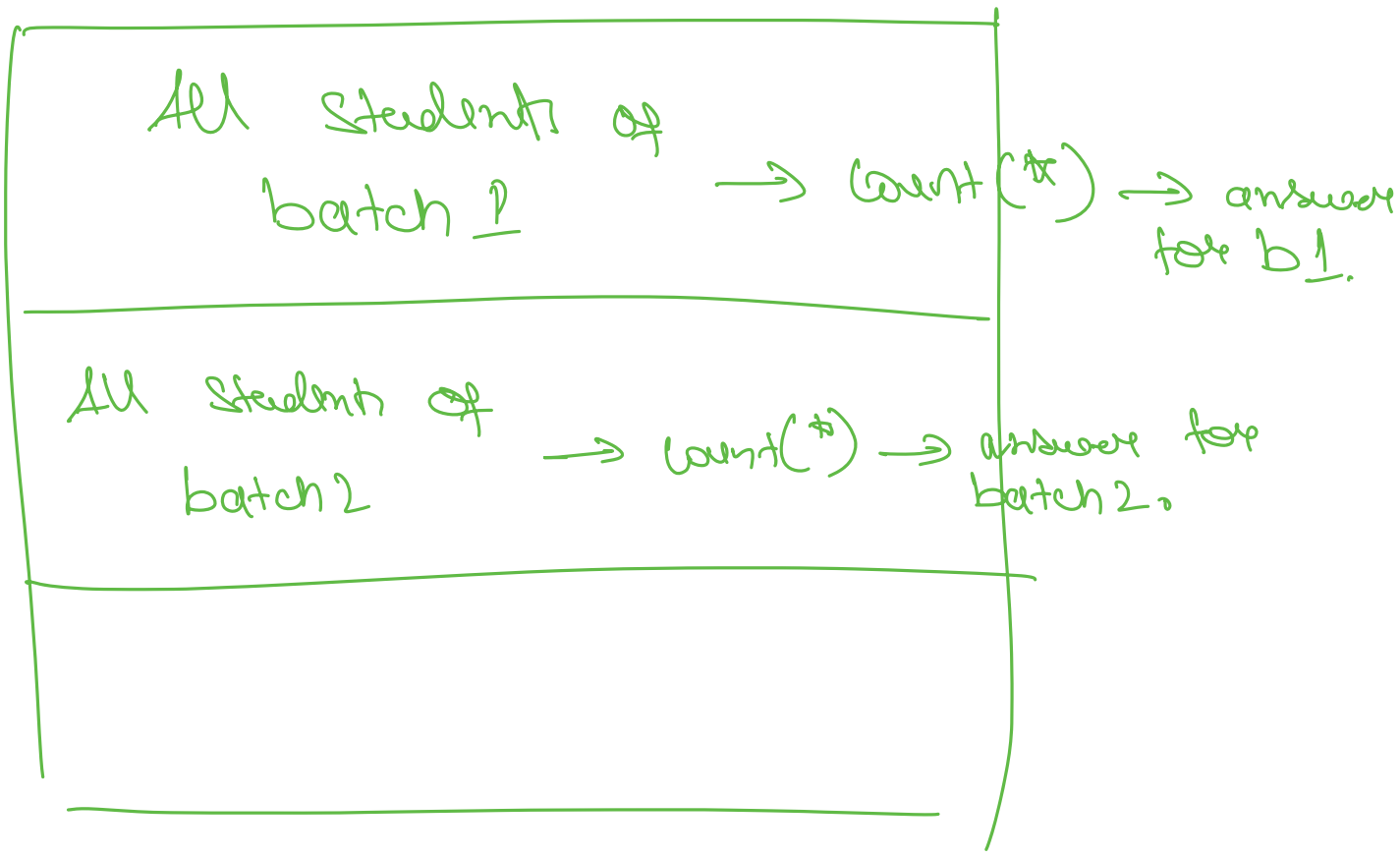
Break till 10:28 PM

Students

id	name	psp	batch_id

Q) Get the count of students for every batch.

batch_id	count of students
1	50
2	30
3	40
4	60

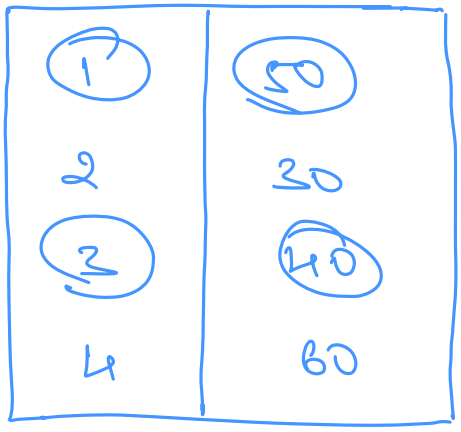


I can make use of aggregates, Not on the entire table, but on the sub-parts (groups) of the table.

Group By: Allows you to break your table into smaller groups, so as to be used by the aggregate functions.

Eg → 1) Group by batch_id.
// It will bring all the rows of same batch together.

```
Select batchid, count(id)
FROM Students
Group By batchid
```



2) Group by on 2 diff. columns.

Q) How many different groups will be there?

Group by b_id, ins_id.

b_id	ins_id	st_id
1	100	20
1	100	21
1	150	30
1	150	31
1	150	32
1	100	33

1	100	34
2	200	40
2	200	45
1	100	50

Groups

1, 100 → 5 students

1, 150 → 3

2, 200 → 2

Students

id	name	batch_id	psp
1	A	1	10
2	B	1	20
3	C	2	30
4	D	1	40

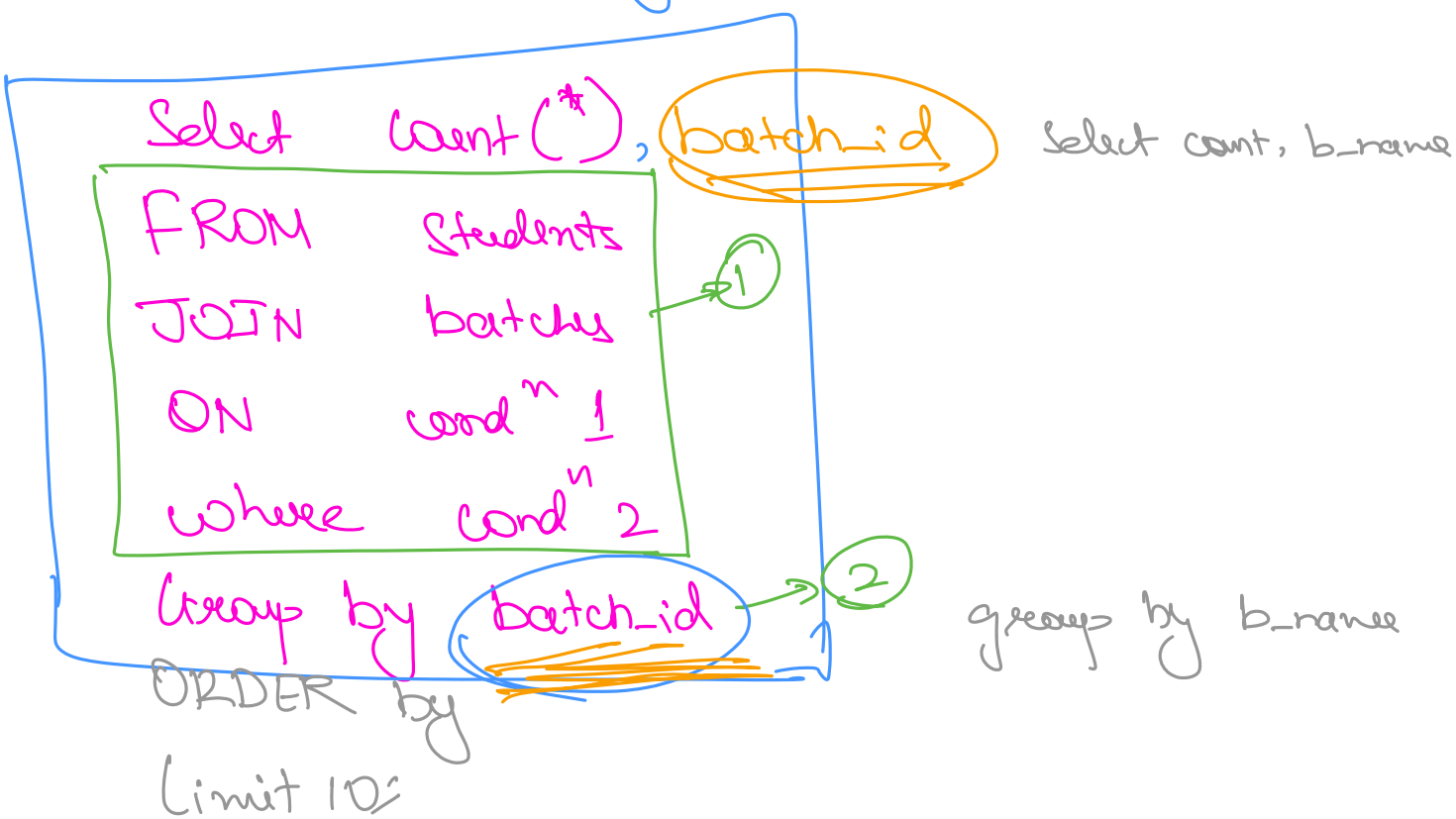
Q) For each batch, get me the no. of students.

```
Select  count(id)
from    Students
```

Group By batch_id

↓

General SQL Query.



Output :

count(*)	batch_id	
3	1	Nov22
1	2	Nov22

→ You can only use those columns that are present in Group By.

↳ Select will have those columns, that are mentioned in Group By

---

Q) Print the batch names with more than 100 students.

Students

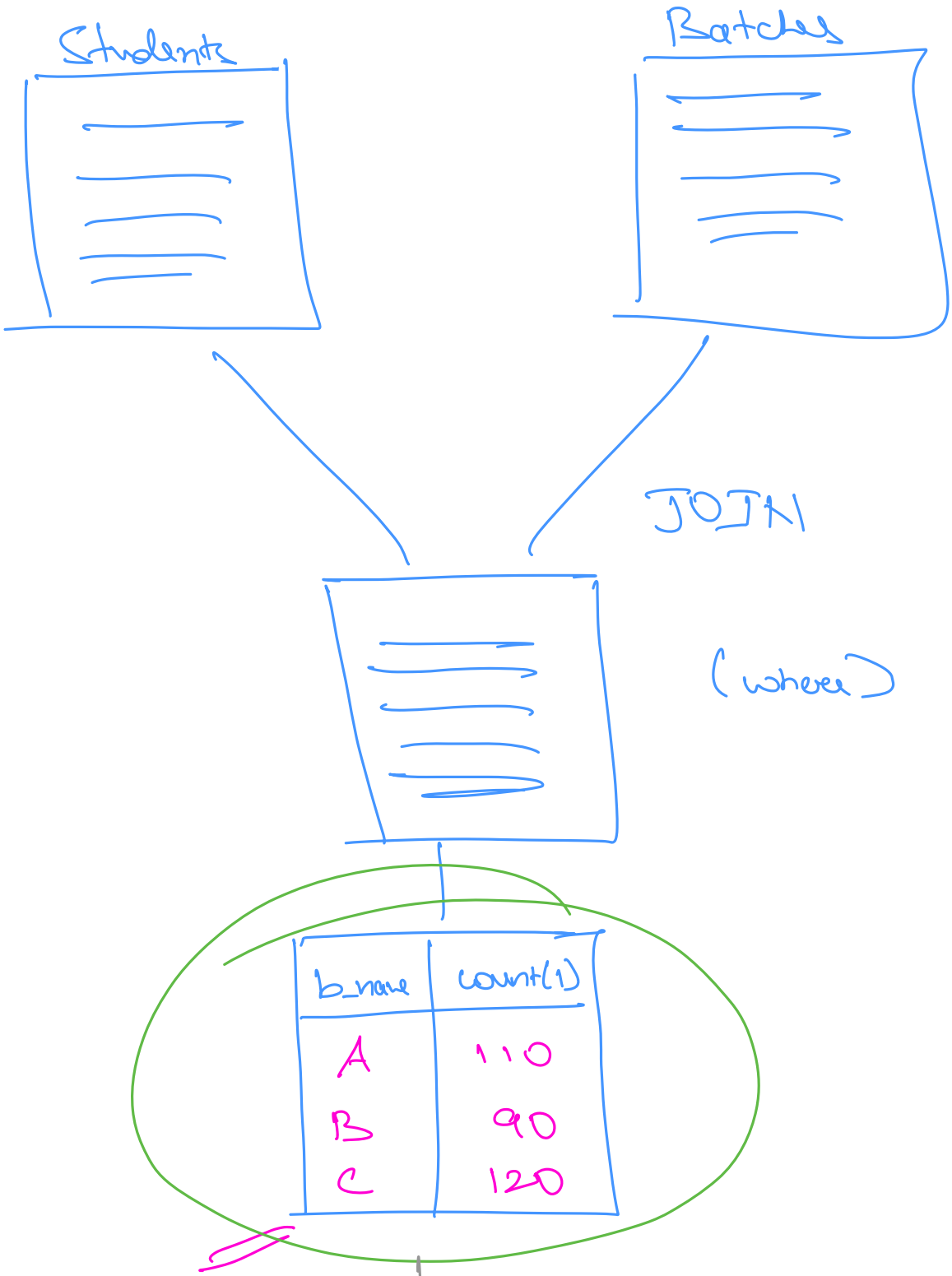
id	name	batch_id	psp

Batches

id	name

1) Step 1 : Q) Print batch_names along with the count of students.

```
SELECT  b.name, count(1)
FROM    students  s
JOIN    batches   b
ON      s.batch_id = b.id
GROUP BY b.name
```



I need to do some filtering on this table.

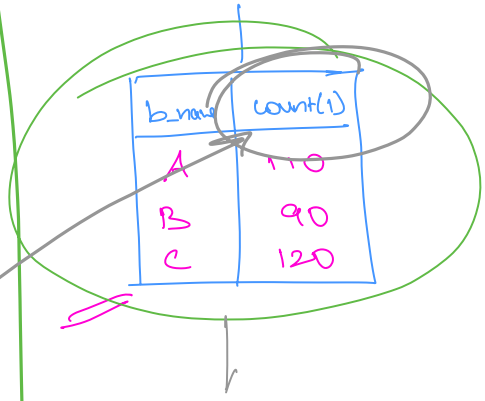
Do I need to filter on Rows? No.
I want to filter on the groups

HAVING

→ Allows you to filter on the groups.

Q) Print the batch names with more than 100 students.

```
Select  b.name, count(1)
FROM    students s
JOIN    batches b
ON      s.batch_id = b.id
GROUP BY b.name
HAVING  count(1) > 100
```



Code

A = [{} .... {} ....]
B = [{} ..... {} ..]
ans1 = [ ]

```
for row1 in A:
    for row2 in B:                    JOIN
        if row1 and row2 match ON cond^n:
            ans1.add(row1 + row2 (Stitch))
```

ans2 = [ ]

```
for row3 in (ans1):        → virtual table.
    if row3 matches the (WHERE) cond^n:
        ans2.add[row3];
```

Map<(Group By col^m), int> countMap
Map<(Group By col^m), Double> avg Map

```
for row in ans2:
```

// populate the maps

countMap.put(b.name, existing_value + 1)

avgMap.update( )

GROUP BY

generating the aggregates

for each group in countMap:

if condition in Having clause is true:

print(countMap.get(group), avgMap.get(group))

countMap	
A	110
B	90
C	120

ORDER of Operations:

FROM

↓

JOIN (ON)

↓

WHERE

↓

GROUP BY

↓

HAVING

↓

SELECT

↓

DISTINCT

↓

ORDER BY

↓

LIMIT

Q) Can I apply WHERE after Group By?

No

HAVING will be applied.