1. Good Evening
2. We will begin at 9:07 PM
3. Aggregates & Builtin fns.

AGENDA

1. Aggregates
2. GROUP BY
3. HAVING / where
4. Builtin fns.

Aggregates → Max, Min, Sum, Avg, Count

| stid | name | b_id | psp | coins |
|---|---|---|---|---|
| 1 | A | 1 | 80 | 30 |
| 2 | B | 1 | 90 | 20 |
| 3 | C | 2 | 40 | 10 |
| 4 | D | 2 | 100 | 40 |

Students

| bid | psp |
|---|---|
| 1 | 85 |
| 2 | 70 |

→ Max PSP for all students

→ Give me coins for all students.

SUM (1, 2, 5, 10) = 18

MIN (1, 2, 5, 10) = 1

Max (1, 2, 5, 10) = 10

AVG (1, 2, 5, 10) = 4.5
COUNT (1, 2, 5, 10) = 4

★ Aggregates work on non-null values

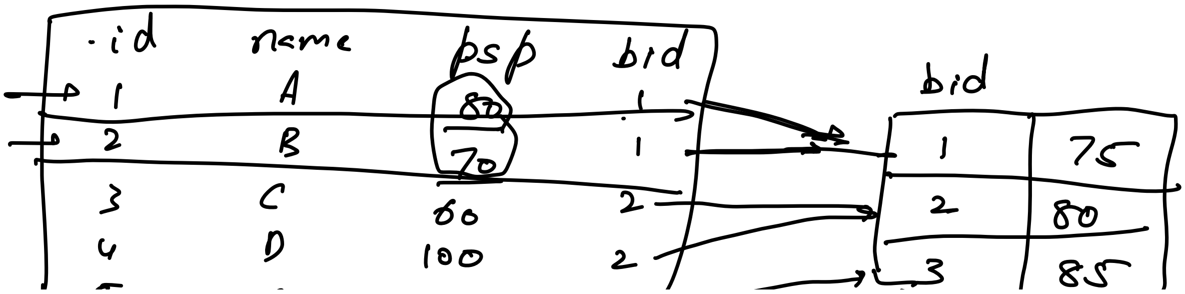
COUNT ( 1, 2, NULL, 5, 10) = 4
AVG ( 1, 2, NULL, 5, 10) = 4.5

## 2. GROUP BY

- ★ → Find average psp of all students
- ★ → Find average psp for each batch.

```
SELECT AVG(psp)
FROM students;
```

```
SELECT batch_id, AVG(psp)
FROM students
GROUP BY batch_id
```



| id | name | psp | bid |
|---|---|---|---|
| 1 | A | 80 | 1 |
| 2 | B | 70 | 1 |
| 3 | C | 60 | 2 |
| 4 | D | 100 | 2 |

| bid | |
|---|---|
| 1 | 75 |
| 2 | 80 |
| 3 | 85 |

5 E 85 3

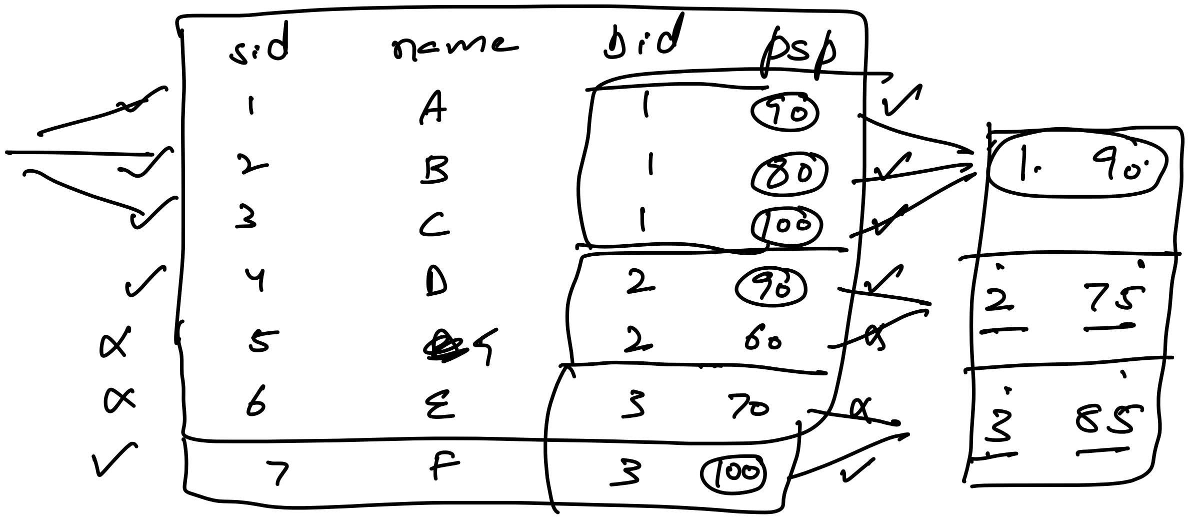
★ The select clause can have grouped by column as it is. Any other column can come in select clause only as an aggregate fn.

★ WBQ : How many orders per status.

HAVING [How is it different from where]

[1. AVG PSP of students whose psp ≥ 80

2. Avg PSP of batches with avg ≥ 80



|  | sid | name | bid | psp |  |
|---|---|---|---|---|---|
| ✓ | 1 | A | 1 | 90 | ✓ |
| ✓ | 2 | B | 1 | 80 | ✓ |
| ✓ | 3 | C | 1 | 100 | ✓ |
| ✓ | 4 | D | 2 | 90 | ✓ |
| ✗ | 5 | G | 2 | 60 | ✗ |
| ✗ | 6 | E | 3 | 70 | ✗ |
| ✓ | 7 | F | 3 | 100 | ✓ |

|  |  |
|---|---|
| 1. | 90 |
| 2 | 75 |
| 3 | 85 |

Q1

```
SELECT AVG(psp)
FROM students
```

WHERE psp ≥ 80

☆ WHERE applies to individual rows & not groups

```
SELECT  b_id, AVG (PSP)
FROM  students
GROUP BY  b_id
HAVING  AVG (psp) ≥ 80
```

| WHERE | HAVING |
|---|---|
| 1. Before group by | 1. After group by |
| 2. Filters the rows | 2. Filter groups |
| 3. Can't use aggregates | 3. Can use aggregates |

WHERE, GROUP BY, HAVING

1. where
2. group by
3. HAVING
4. order by

5. TOP | Limit

6. SELECT . . .

✩ [Find avg psp of every batch that has more than 2 students]

```
Select bid, AVG (PSP)
FROM students
GROUP BY bid
HAVING COUNT(*) ≥ 2
```

Break [10:23 – 10:30]

Builtin fns

- Number
- String
- Datetime

Number

1. Round ( n, digits after decimal)

Round (2.31$\underline{56}$, 2) → 2.32

Round ( 2.3149, 2) → 2.31

2. Truncate (n, digits after decimal)

Truncate (2.3156, 2) → 2.31

3. CEILING (3.1) → 4

CEILING (3) → 3

4. FLOOR ( 3.1) → 3

FLOOR (3) → 3

$|-5| = 5$

$|5| = 5$

5. ABS (-2.5) = ABS(2.5) = 2.5

6. RAND( ) → 0 to 1

* Generate a random b/w 0 to 10

[FLOOR (RAND () * 10)

[0 → 0.099...] → 0

[0.1 → 0.199...] → 1

[0.2 → 0.299..] → 2

:

[0.9 → 0.9999] → 9

[1] → 10 ✓

$\boxed{\text{ROUND ( Rand(), 1) x 10}}$

0.96

→ Strings

- ✓ LENGTH (s)
- Upper ("delhi") → DELHI

  Upper ( "DelHi") → DELHI
- LOWER ("DELHI") → delhi
- LTRIM, RTRIM, TRIM

LTRIM ("...delhi...") = "delhi..."

RTRIM ("...delhi...") = "...delhi"

TRIM ("... delhi...") = "delhi"

→ LEFT, RIGHT, Substring

LEFT ("delhi", 2) = "de"

RIGHT ("delhi", 2) = "hi"

SUBSTRING (s, start-pos, length)

SUBSTRING ("helloworld", 5, 3) = owo

1 2 3 4 5 6 7 8 9 10

→ LOCATE (tofind, s)

LOCATE ("el", "delhi") = 2

1 2 3 4 1

LOCATE ("war", "delhi") = 0

→ DATE functions

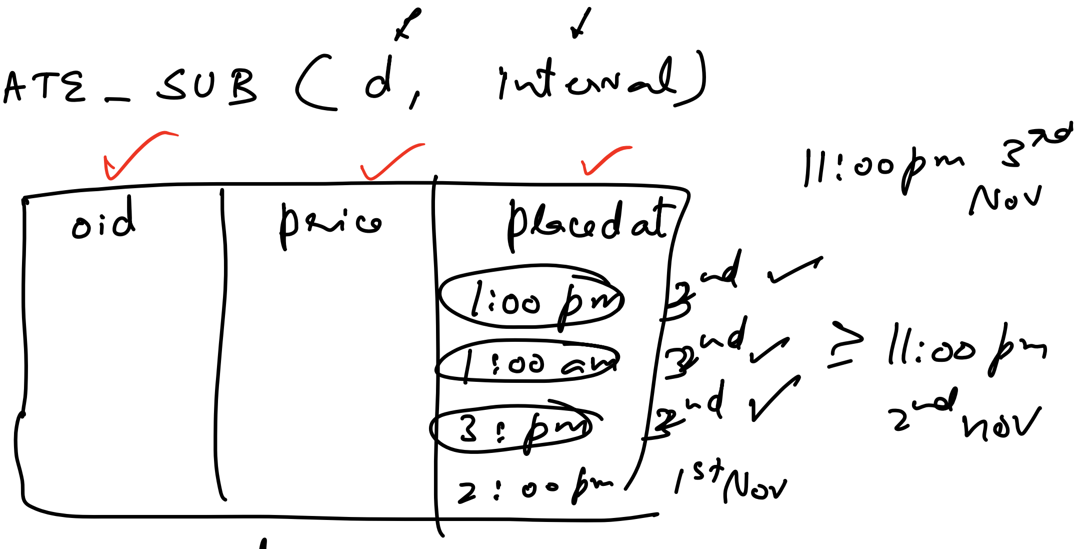
1. NOW()
2. CURDATE() ✓
3. CURTIME()
4. YEAR ("1991-12-31") = 1991 date
5. MONTH ("1991-12-31") = 12
6. DAYNAME (d

7. DATE_ADD (d, interval)

DATE_ADD ( NOW() , 10 MINUTE INTERVAL )

SECONDS<br>
HOUR<br>
DAYS<br>
MONTHS<br>
YEAR

8. DATE_SUB ( d, interval)



| oid | price | placedat |
|---|---|---|
| | | 1:00 pm 3rd ✓ |
| | | 1:00 am 3rd ✓ |
| | | 3 : pm 2nd ✓ |
| | | 2 : 00 pm 1st Nov |

11:00pm 3rd Nov

≥ 11:00 pm 2nd nov

Orders

☆ Give orders placed during ~~last~~ previous 24 hours.

```
SELECT *
  FROM orders
   WHERE placedat ≥ DATE_SUB(now(),
                        24 hour interval)
```

→ DATE_DIFF(d1, d2) → Number of days btw dates.