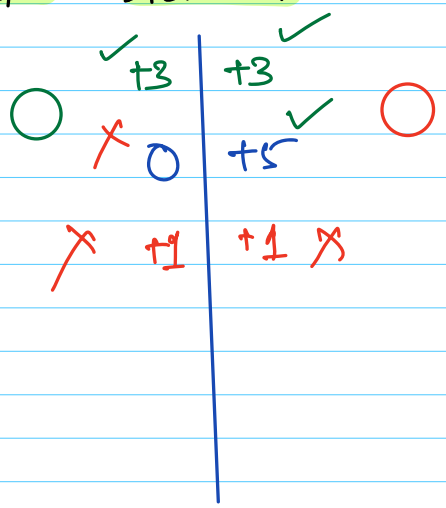
17/05/24

SQL 07 : AGGREGATE QUERIES

PRISONER'S DILEMMA



AGENDA

1. Aggregate Queries.
   - → COUNT
   - → AVG
   - → SUM
   - → MAX
   - → MIN

STDDEV
VARIANCE

2. GROUP BY
3. HAVIN
4. ORDER of EXECUTION.

AGGREGATION

- collection
- combine
- integration
- grouping.

What is avg psp of batch?
max psp
no of movies released year.

psp bid.

COUNT

Note: Aggregate functions do NOT count NULL values.

eg list1 = [1, 2, 3, 4, 5]

avg(list1) = 3

list2 = [1, 2, 3, NULL, 4]

avg(list2) = 10/5 = 2 ✗

or avg(list2) = 10/4 = 2.5 ✓

Pseudo code

```
table_name = []
count = 0

for row in table_name:
        if row[col] is NOT NULL:
                count += 1

print(count)
```

Count(1)
Count("Name")

ASTERISK (*)

Count (*)

```
table_name = []
count = 0

for row in table_name:
    count += 1

print (count)
```

Count (*)
count (1)

NULL == NULL ✗

NULL != NULL ✗

Count (NULL) = 0 ✓

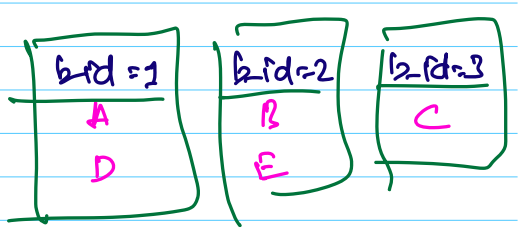
```
table_name = []
count = 0

for row in table_name:
    if (NULL == NULL):
        count += 1

print (count)
```

Table: Students

St_id	S_name	b_id	PSP
1	A	1	90
2	B	2	70
3	C	3	60
4	D	1	80
5	E	2	50



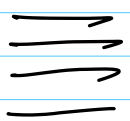
group by (b_id, b_name)

(1, A) (2, B) (3, C) (1, D) (2, E)

A = 90 B = 70 C 60 D = 80 E = 50.

HAVING

Filtering grouped values. or aggregated queries.



Individual rows.

→ WHERE

Avg PSP

1 - 87 ✗

2 - 91 ✓

3 - 92 ✓

Batches avg(PSP) > 90.

Ques Print only those batches - b_id, b_name, avg_psp whose avg psp of toppers > 90%

psp toppers > 75%

1 85 ✗

2 87 ✗

3 92 ✓

1 - 3 DAY 92

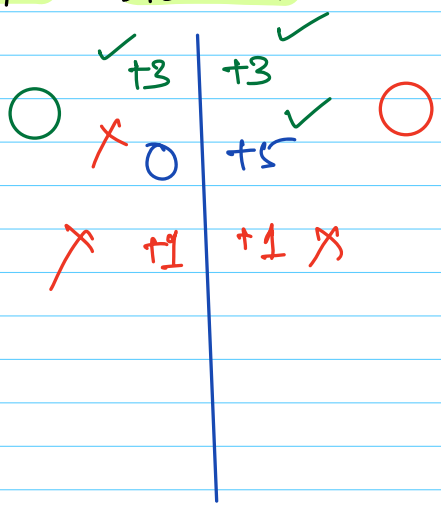
ORDER OF EXECUTION

FROM
↓
JOIN / ON (Sample space)
↓ intermediary table
FILTER → WHERE
↓
GROUP BY (on cols)
↓
AGGREGATE (count, sum, avg)
↓
HAVING (filter grouped / aggregated values)
↓
SELECT (cols. to print)
↓
DISTINCT (UNIQUE)
↓
ORDER BY (SORT)
↓
LIMIT / OFFSET (Pagination)

PRISONER'S DILEMMA

✓ Cooperate

✗ Ditch



Ditch

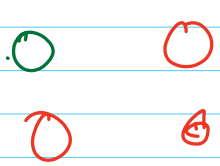
Strategy:

200 rounds. (1000)

	P1	P2
Ditch	200	200
Cooperate	600	600

Tit For Tat (10%) lenient

By default → Cooperate.



1. Be cooperative
2. Start with good
3. Dont be a pushover.

```sql
-- SQL 07 AGGREGATE QUERIES
USE SQL_030524;

-- COUNT OF STUDENTS WITH A BATCH
SELECT COUNT(B_ID)
FROM STUDENTS;

-- COUNT ALL STUDENTS
SELECT COUNT(S_ID)
FROM STUDENTS;

-- * - COUNT ALL THE ROWS IN A TABLE (INCLUDING
NULLS)
-- ASTERISK - *
SELECT COUNT(*) FROM STUDENTS;

SELECT COUNT(1) FROM STUDENTS;

SELECT COUNT('RAUSHAN') FROM STUDENTS;


-- COUNT NULL
SELECT COUNT(NULL) FROM STUDENTS;

SELECT COUNT(NOT NULL) FROM STUDENTS;

-- COUNT NULL VALUES
SELECT COUNT(*) FROM STUDENTS
WHERE B_ID IS NULL;

-- COUNT UNIQUE COL VALUES
SELECT COUNT(DISTINCT B_ID)
FROM STUDENTS;
```

```
SELECT MAX(S_NAME) FROM STUDENTS;
SELECT MIN(S_NAME) FROM STUDENTS;


-- PRINT AVG PSP OF EVERY BATCH, 1, 2, 3 SEPARATELY
-- AVG PSP OF BATCH ID =1
SELECT AVG(PSP) FROM STUDENTS
WHERE B_ID = 1;

SELECT AVG(PSP) FROM STUDENTS
WHERE B_ID IS NOT NULL;

SELECT AVG(PSP) FROM STUDENTS
WHERE B_ID = 1
UNION
SELECT AVG(PSP) FROM STUDENTS
WHERE B_ID = 2
UNION
SELECT AVG(PSP) FROM STUDENTS
WHERE B_ID = 3;
```

```
-- GROUP BY
SELECT B_ID, AVG(PSP)
FROM STUDENTS
GROUP BY B_ID;

-- NOTE:
-- Although the null value is neither equal to any other
value
-- nor not equal to any other value —
-- it is unknown whether or not it is equal to any given
value —
-- in some contexts, multiple null values are treated
together;
-- for example, the <group by clause> treats all null
values together


SELECT B_ID, S_NAME, AVG(PSP)
FROM STUDENTS
GROUP BY B_ID, S_NAME;

-- Error Code: 1055. Expression #3 of SELECT list is not in
-- GROUP BY clause and contains
-- nonaggregated column
'sql_030524.STUDENTS.S_NAME'
-- which is not functionally dependent on columns in
GROUP BY clause;
-- this is incompatible with sql_mode=only_full_group_by

-- PRINT B_ID, B_NAME, AVG(PSP) OF EVERY BATCH
-- CONSIDER ONLY TOPPERS OF THE BATCH
-- PSP OF TOPPER > 75%

SELECT B.B_ID, B.B_NAME, AVG(PSP)
FROM STUDENTS S
```

```
- PRINT B_ID, B_NAME, AVG(PSP) OF EVERY BATCH
-- CONSIDER ONLY TOPPERS OF THE BATCH
-- PSP OF TOPPER > 75%

SELECT B.B_ID, B.B_NAME, AVG(PSP)
FROM STUDENTS S
JOIN BATCHES B
ON S.B_ID = B.B_ID AND S.PSP>75
GROUP BY B.B_ID;

SELECT B.B_ID, B.B_NAME, AVG(PSP)
FROM STUDENTS S
JOIN BATCHES B
ON S.B_ID = B.B_ID
GROUP BY B.B_ID;

-- Q2 PRINT ONLY THOSE BATCHES WHOSE
-- AVG PSP OF TOPPERS > 90%
-- PSP OF TOPPERS > 75%

SELECT B.B_ID, B.B_NAME, AVG(PSP)
FROM STUDENTS S
JOIN BATCHES B
ON S.B_ID = B.B_ID AND S.PSP>75
GROUP BY B.B_ID
HAVING AVG(PSP) > 90;

-- HAVING IS FOR AGGREGATED VALUES/ GROUPED
VALUES
-- BECAUSE WHERE IS FOR INDVIDUAL ROWS

SELECT B.B_ID, B.B_NAME, AVG(PSP)
FROM STUDENTS S
JOIN BATCHES B
ON S.B_ID = B.B_ID AND S.PSP>75
```

```
-- HAVING IS FOR AGGREGATED VALUES/ GROUPED
VALUES
-- BECAUSE WHERE IS FOR INDVIDUAL ROWS

SELECT B.B_ID, B.B_NAME, AVG(PSP)
FROM STUDENTS S
JOIN BATCHES B
ON S.B_ID = B.B_ID AND S.PSP>75
GROUP BY B.B_ID
HAVING AVG(PSP) > 90;


SELECT B.B_ID, B.B_NAME, AVG(PSP)
FROM STUDENTS S
JOIN BATCHES B
ON S.B_ID = B.B_ID
WHERE S.PSP>75
GROUP BY B.B_ID
HAVING AVG(PSP) > 90;

-- Q3 FIND ALL THE ACTORS WHO APPEARED IN AT
LEAST 2 FILMS TOGETHER
-- PRINT THE ACTOR IDS AND (PAIRS)
-- THE NO. OF FILMS THEY HAVE WORKED TOGETHER
IN DESC ORDER

USE SAKILA;

SELECT A1.ACTOR_ID, A2.ACTOR_ID,
COUNT(*) AS FILMS_TOGETHER
FROM FILM_ACTOR A1
JOIN FILM_ACTOR A2
ON A1.FILM_ID = A2.FILM_ID
AND A1.ACTOR_ID < A2.ACTOR_ID
GROUP BY A1.ACTOR_ID, A2.ACTOR_ID
```