Learn SQL by Example

from Basic to Advanced

Chananel Perel

2023

Learn SQL by Example (Chananel Perel) 2024-04-10 16:17:23.404279

SQL READ 3 CONT.

Learn SQL by Example (Chananel Perel (2023

VALUES

SQL: VALUES - Example #1 - Q:

```
SELECT * FROM (VALUES(1,2),(3,4),(5+6,8-7))
```

SQL: VALUES - Example #1 - A:

```
SELECT * FROM (VALUES(1,2),(3,4),(5+6,8-7))
```

column1	column2
1	2
3	4
11	1

Learn SQL by Example (Chananel Perel 2023)

WITH

SQL: WITH - Example #2 - Q:

```
WITH tbl1(col1,col2) AS (VALUES(1,2),(3,4),(5+6,8-7))
SELECT *
FROM tbl1
```

SQL: WITH - Example #2 - A:

```
WITH tbl1(col1,col2) AS (VALUES(1,2),(3,4),(5+6,8-7))
SELECT *
FROM tbl1
```

col1	col2
1	2
3	4
11	1

SQL: WITH - Example #3 - Q:

```
WITH tbl1(col1,col2) AS (VALUES(1,2),(3,4),(5+6,8-7)),
     tbl3(col1,col2) AS (SELECT col1 * 2, col2 + 3
                         FROM tbl1)
SELECT *
FROM tbl3
```

SQL: WITH - Example #3 - A:

```
WITH tbl1(col1,col2) AS (VALUES(1,2),(3,4),(5+6,8-7)),
     tbl3(col1,col2) AS (SELECT col1 * 2, col2 + 3
                         FROM tbl1)
SELECT *
FROM tbl3
```

col1	col2
2	5
6	7
22	4

SQL: WITH - Example #4 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	None
3	Cohen	David	22	economy	2019-02-12	19	None
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	music	2019-12-10	15	60

```
WITH  only_david AS (SELECT *
                     FROM table3
                     WHERE first_n = 'David')
SELECT COUNT(*),
       SUM(cost)
FROM only_david
```

SQL: WITH - Example #4 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	None
3	Cohen	David	22	economy	2019-02-12	19	None
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	music	2019-12-10	15	60

```
WITH  only_david AS (SELECT *
                     FROM table3
                     WHERE first_n =
'David')
SELECT COUNT(*),
       SUM(cost)
FROM only_david
```

COUNT(*)	SUM(cost)
3	54

Learn SQL by Example (Chananel Perel 2023)

GROUP BY

SQL: GROUP BY - Example #5 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course,
       COUNT(*),
       AVG(score)
FROM table1
GROUP BY course
ORDER BY 1
```

SQL: GROUP BY - Example #5 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course,
       COUNT(*),
       AVG(score)
FROM table1
GROUP BY course
ORDER BY 1
```

course	COUNT(*)	AVG(score)
economy	1	77.0
history	3	77.66666666666667
math	4	89.75
music	1	80.0

SQL: GROUP BY - Example #6 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course,
       MAX(cost) max_cost,
       MIN(age) min_age,
       ROUND(AVG(score),2) avg_score,
       GROUP_CONCAT(DISTINCT first_n||' '||
            SUBSTR(last_n,1,1)||'.') all_names
FROM table1
GROUP BY course
ORDER BY 1
```

SQL: GROUP BY - Example #6 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
    SELECT course,
           MAX(cost) max_cost,
           MIN(age) min_age,
           ROUND(AVG(score),2) avg_score,
           GROUP_CONCAT(DISTINCT first_n||'
'||
                 SUBSTR(last_n,1,1)||'.')
all_names
    FROM table1
    GROUP BY course
    ORDER BY 1
```

course	max_cost	min_age	avg_score	all_names
economy	19	22	77.0	David C.
history	18	22	77.67	David C.,Chaim P.,David G.
math	22	22	89.75	David C.,Moshe L.,Moshe C.,Chaim P.
music	20	23	80.0	Moshe C.

This looks more like "real world" query we will run..

SQL: GROUP BY - Example #7 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT first_n, last_n,
       AVG(score),
       SUM(cost)
FROM table1
GROUP BY first_n, last_n
ORDER BY 1,2
```

SQL: GROUP BY - Example #7 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT first_n, last_n,
       AVG(score),
       SUM(cost)
FROM table1
GROUP BY first_n, last_n
ORDER BY 1,2
```

first_n	last_n	AVG(score)	SUM(cost)
Chaim	Peled	65.0	31
David	Cohen	88.0	54
David	Gamil	85.0	17
Moshe	Cohen	85.0	41
Moshe	Levi	100.0	22

We can GROUP BY more than one column

SQL: GROUP BY - Example #8 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course
FROM table1
GROUP BY course
ORDER BY 1
```

SQL: GROUP BY - Example #8 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course
FROM table1
GROUP BY course
ORDER BY 1
```

course
economy
history
math
music

We can also use GROUP BY just to show unique values, without running any aggregate functior

Learn SQL by Example (Chananel Perel 2023)

HAVING

SQL: HAVING - Example #9 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course,
       COUNT(*),
       AVG(score)
FROM table1
GROUP BY course
HAVING COUNT(*) > 1
```

SQL: HAVING - Example #9 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course,
       COUNT(*),
       AVG(score)
FROM table1
GROUP BY course
HAVING COUNT(*) > 1
```

course	COUNT(*)	AVG(score)
history	3	77.66666666666667
math	4	89.75

SQL: HAVING - Example #10 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
WITH  count_course AS (
       SELECT course, COUNT(*) cc, AVG(score)
       FROM table1
       GROUP BY course
)
SELECT *
FROM count_course
WHERE cc > 1
```

SQL: HAVING - Example #10 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
WITH  count_course AS (
       SELECT course, COUNT(*) cc,
AVG(score)
       FROM table1
       GROUP BY course
)
SELECT *
FROM count_course
WHERE cc > 1
```

course	cc	AVG(score)
history	3	77.66666666666667
math	4	89.75

We can do same thing without HAVING.. but less elegant..

Learn SQL by Example (Chananel Perel 2023)

FILTER

SQL: FILTER - Example #11 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT 'age gt 22' kind,
       COUNT(*)    val
FROM table1
WHERE age > 22
UNION ALL
SELECT 'score lt 80' kind,
       COUNT(*)      val
FROM table1
WHERE score < 80
```

SQL: FILTER - Example #11 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT 'age gt 22' kind,
       COUNT(*)     val
FROM table1
WHERE age > 22
UNION ALL
SELECT 'score lt 80' kind,
       COUNT(*)      val
FROM table1
WHERE score < 80
```

kind	val
age gt 22	6
score lt 80	3

SQL: FILTER - Example #12 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT *
FROM  ( SELECT COUNT(*) high_age
        FROM table1
        WHERE age > 22 ),
      ( SELECT COUNT(*) low_score
        FROM table1
        WHERE score < 80 )
```

SQL: FILTER - Example #12 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT *
FROM  ( SELECT COUNT(*) high_age
        FROM table1
        WHERE age > 22 ),
      ( SELECT COUNT(*) low_score
        FROM table1
        WHERE score < 80 )
```

high_age	low_score
6	3

Cross Join

SQL: FILTER - Example #13 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT *
FROM  ( SELECT COUNT(*) high_age
        FROM table1
        WHERE age > 22 )
JOIN  ( SELECT COUNT(*) low_score
        FROM table1
        WHERE score < 80 )
ON True -- or you can write 1 | 1=1 | etc..
```

SQL: FILTER - Example #13 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT *
FROM  ( SELECT COUNT(*) high_age
        FROM table1
        WHERE age > 22 )
JOIN  ( SELECT COUNT(*) low_score
        FROM table1
        WHERE score < 80 )
ON True -- or you can write 1 | 1=1 |
etc..
```

high_age	low_score
6	3

Using Join and always True condition

SQL: FILTER - Example #14 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT COUNT(*) FILTER (WHERE age > 22) high_age,
       COUNT(*) FILTER (WHERE score < 80) low_score
FROM table1
```

SQL: FILTER - Example #14 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
    SELECT COUNT(*) FILTER (WHERE age > 22)
high_age,
           COUNT(*) FILTER (WHERE score <
80) low_score
    FROM table1
```

high_age	low_score
6	3

Filter can do things much easier..

SQL: FILTER - Example #15 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course,
       COUNT(*) FILTER (WHERE age > 22) high_age,
       COUNT(*) FILTER (WHERE score < 80) low_score
FROM table1
GROUP BY course
```

SQL: FILTER - Example #15 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course,
       COUNT(*) FILTER (WHERE age > 22) high_age,
       COUNT(*) FILTER (WHERE score < 80) low_score
FROM table1
GROUP BY course
```

course	high_age	low_score
economy	0	1
history	2	1
math	3	1
music	1	0

We can also do this within groups..

SQL: FILTER - Example #16 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course,
       COUNT(CASE WHEN age > 22 THEN 1 ELSE NULL END) high_age,
       COUNT(CASE WHEN score < 80 THEN 1 END) low_score
FROM table1
GROUP BY course
```

SQL: FILTER - Example #16 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
    SELECT course,
           COUNT(CASE WHEN age > 22 THEN 1
ELSE NULL END) high_age,
           COUNT(CASE WHEN score < 80 THEN 1
END) low_score
    FROM table1
    GROUP BY course
```

course	high_age	low_score
economy	0	1
history	2	1
math	3	1
music	1	0

In some version FILTER is not supported, so we can use CASE WHEN

SQL: FILTER - Example #17 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT
   1.0 * SUM(score) / (SELECT SUM(score) FROM table1)
FROM table1
WHERE course = 'history'
```

SQL: FILTER - Example #17 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT
   1.0 * SUM(score) / (SELECT SUM(score)
FROM table1)
FROM table1
WHERE course = 'history'
```

1.0 * SUM(score) / (SELECT SUM(score) FROM table1)
0.31108144192256343

Show percentage from all, we actually run here 2 queries

SQL: FILTER - Example #18 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT
  1.0 * SUM(score) FILTER (WHERE course = 'history') /
SUM(score)
FROM table1
```

SQL: FILTER - Example #18 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT
  1.0 * SUM(score) FILTER (WHERE course
= 'history') / SUM(score)
FROM table1
```

1.0 * SUM(score) FILTER (WHERE course = 'history') / SUM(score)
0.31108144192256343

And now we do the same thing, with one query..

Learn SQL by Example (Chananel Perel) 2024-04-10 16:17:24.619996

SQL Q&A

Learn SQL by Example (Chananel Perel 2023)

ALL WHO DID HISTORY COURSE 1

SQL: All Who Did History Course 1 - Example #19 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT last_n,first_n
FROM table1
WHERE course = 'history'
ORDER BY 1,2
```

SQL: All Who Did History Course 1 - Example #19 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT last_n,first_n
FROM table1
WHERE course = 'history'
ORDER BY 1,2
```

last_n	first_n
Cohen	David
Gamil	David
Peled	Chaim

Easy..

Learn SQL by Example (Chananel Perel 2023)

ALL WHO DID NOT DO HISTORY COURSE

SQL: All Who Did NOT DO History Course - Example #20 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT last_n,first_n
FROM table1
WHERE course != 'history'
ORDER BY 1,2
```

SQL: All Who Did NOT DO History Course - Example #20 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT last_n,first_n
FROM table1
WHERE course != 'history'
ORDER BY 1,2
```

last_n	first_n
Cohen	David
Cohen	Moshe
Levi	Moshe
Peled	Chaim

This is NOT good..!

SQL: All Who Did NOT DO History Course - Example #21 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT last_n,first_n
FROM table1
GROUP BY last_n,first_n
HAVING COUNT(CASE WHEN course = 'history' THEN 1 END) = 0
ORDER BY 1,2
```

SQL: All Who Did NOT DO History Course - Example #21 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT last_n,first_n
FROM table1
GROUP BY last_n,first_n
HAVING COUNT(CASE WHEN course = 'history' THEN 1 END) = 0
ORDER BY 1,2
```

last_n	first_n
Cohen	Moshe
Levi	Moshe

OK

Learn SQL by Example (Chananel Perel 2023)

ALL WHO DID HISTORY COURSE 2

SQL: All Who Did History Course 2 - Example #22 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT last_n,first_n
FROM table1
GROUP BY last_n,first_n
HAVING COUNT(CASE WHEN course = 'history' THEN 1 END) > 0
ORDER BY 1,2
```

SQL: All Who Did History Course 2 - Example #22 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
    SELECT last_n,first_n
    FROM table1
    GROUP BY last_n,first_n
    HAVING COUNT(CASE WHEN course =
'history' THEN 1 END) > 0
    ORDER BY 1,2
```

last_n	first_n
Cohen	David
Gamil	David
Peled	Chaim

So now also here can use this

Learn SQL by Example (Chananel Perel 2023)

HOW MANY DID NOT DO HISTORY COURSE

SQL: How many did NOT DO History Course - Example #23 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT COUNT(*)
FROM table1
GROUP BY last_n,first_n
HAVING COUNT(CASE WHEN course = 'history' THEN 1 END) = 0
```

SQL: How many did NOT DO History Course - Example #23 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT COUNT(*)
FROM table1
GROUP BY last_n,first_n
HAVING COUNT(CASE WHEN course =
'history' THEN 1 END) = 0
```

COUNT(*)
2
1

This is NOT good..

SQL: How many did NOT DO History Course - Example #24 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT GROUP_CONCAT(id)
FROM table1
GROUP BY last_n,first_n
HAVING COUNT(CASE WHEN course = 'history' THEN 1 END) = 0
```

SQL: How many did NOT DO History Course - Example #24 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT GROUP_CONCAT(id)
FROM table1
GROUP BY last_n,first_n
HAVING COUNT(CASE WHEN course =
'history' THEN 1 END) = 0
```

GROUP_CONCAT(id)
5,6
4

debug technique

SQL: How many did NOT DO History Course - Example #25 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
WITH  no_history AS (
   SELECT last_n,first_n
   FROM table1
   GROUP BY last_n,first_n
   HAVING COUNT(CASE WHEN course = 'history' THEN 1 END) = 0
)
SELECT count(*)
FROM no_history
```

SQL: How many did NOT DO History Course - Example #25 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
WITH  no_history AS (
   SELECT last_n,first_n
   FROM table1
   GROUP BY last_n,first_n
   HAVING COUNT(CASE WHEN course =
'history' THEN 1 END) = 0
)
SELECT count(*)
FROM no_history
```

count(*)
2

OK

Learn SQL by Example (Chananel Perel 2023)

WHAT COURSES PELED CHAIM DID

SQL: What Courses Peled Chaim Did - Example #26 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT course
FROM table1
WHERE last_n = 'Peled' AND first_n = 'Chaim'
ORDER BY 1
```

SQL: What Courses Peled Chaim Did - Example #26 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT course
FROM table1
WHERE last_n = 'Peled' AND first_n =
'Chaim'
ORDER BY 1
```

course
history
math

Easy..

Learn SQL by Example (Chananel Perel 2023)

WHAT COURSES PELED CHAIM DID NOT DO

SQL: What Courses Peled Chaim Did NOT Do - Example #27 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT course
FROM table1
WHERE last_n != 'Peled' AND first_n != 'Chaim'
ORDER BY 1
```

SQL: What Courses Peled Chaim Did NOT Do - Example #27 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT course
FROM table1
WHERE last_n != 'Peled' AND first_n !=
'Chaim'
ORDER BY 1
```

course
economy
history
math
music

This is NOT good..!

SQL: What Courses Peled Chaim Did NOT Do - Example #28 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT course
FROM table1
WHERE course NOT IN (
       SELECT DISTINCT course
       FROM table1
       WHERE last_n = 'Peled' AND first_n = 'Chaim' )
ORDER BY 1
```

SQL: What Courses Peled Chaim Did NOT Do - Example #28 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT course
FROM table1
WHERE course NOT IN (
       SELECT DISTINCT course
       FROM table1
       WHERE last_n = 'Peled' AND
first_n = 'Chaim' )
ORDER BY 1
```

course
economy
music

OK

SQL: What Courses Peled Chaim Did NOT Do - Example #29 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course
FROM table1
GROUP BY course
HAVING COUNT(*) FILTER (WHERE last_n = 'Peled'
                        AND first_n = 'Chaim') = 0
ORDER BY 1
```

SQL: What Courses Peled Chaim Did NOT Do - Example #29 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT course
FROM table1
GROUP BY course
HAVING COUNT(*) FILTER (WHERE last_n =
'Peled'
                        AND first_n =
'Chaim') = 0
ORDER BY 1
```

course
economy
music

And of course we could use HAVING

Learn SQL by Example (Chananel Perel 2023)

WHO DID MATH _OR_ HISTORY COURSE

SQL: Who did Math _OR_ History Course - Example #30 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT last_n,first_n
FROM table1
WHERE course = 'math' or course = 'history'
ORDER BY 1
```

SQL: Who did Math _OR_ History Course - Example #30 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT last_n,first_n
FROM table1
WHERE course = 'math' or course =
'history'
ORDER BY 1
```

last_n	first_n
Cohen	David
Cohen	Moshe
Gamil	David
Levi	Moshe
Peled	Chaim

Easy..

Learn SQL by Example (Chananel Perel 2023)

WHO DID MATH _AND_ HISTORY COURSE

SQL: Who did Math _AND_ History Course - Example #31 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT last_n,first_n
FROM table1
WHERE course = 'math' and course = 'history'
ORDER BY 1
```

SQL: Who did Math _AND_ History Course - Example #31 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT DISTINCT last_n,first_n
FROM table1
WHERE course = 'math' and course =
'history'
ORDER BY 1
```

We get nothing (always).. This is NOT good..!

SQL: Who did Math _AND_ History Course - Example #32 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT last_n,first_n
FROM table1
GROUP BY last_n,first_n
HAVING COUNT(*) FILTER (WHERE course = 'math') > 0
       AND
       COUNT(*) FILTER (WHERE course = 'history') > 0
ORDER BY 1
```

SQL: Who did Math _AND_ History Course - Example #32 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT last_n,first_n
FROM table1
GROUP BY last_n,first_n
HAVING COUNT(*) FILTER (WHERE course =
'math') > 0
       AND
       COUNT(*) FILTER (WHERE course =
'history') > 0
ORDER BY 1
```

last_n	first_n
Cohen	David
Peled	Chaim

This is OK. We get all who did both courses..

SQL: Who did Math _AND_ History Course - Example #33 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT last_n,first_n
FROM table1
GROUP BY last_n,first_n
HAVING COUNT(*) FILTER (WHERE course = 'math') > 0
       OR
       COUNT(*) FILTER (WHERE course = 'history') > 0
ORDER BY 1
```

SQL: Who did Math _AND_ History Course - Example #33 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT last_n,first_n
FROM table1
GROUP BY last_n,first_n
HAVING COUNT(*) FILTER (WHERE course =
'math') > 0
       OR
       COUNT(*) FILTER (WHERE course =
'history') > 0
ORDER BY 1
```

last_n	first_n
Cohen	David
Cohen	Moshe
Gamil	David
Levi	Moshe
Peled	Chaim

And now we can put OR and get all that did at least one of the courses..

Learn SQL by Example (Chananel Perel 2023)

AVERAGE WITH NULL

SQL: Average with NULL - Example #34 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	None
3	Cohen	David	22	economy	2019-02-12	19	None
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	music	2019-12-10	15	60

```
SELECT COUNT(score), AVG(score)
FROM table3
```

SQL: Average with NULL - Example #34 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	None
3	Cohen	David	22	economy	2019-02-12	19	None
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	music	2019-12-10	15	60

```
SELECT COUNT(score), AVG(score)
FROM table3
```

COUNT(score)	AVG(score)
5	85.8

Regular AVG (will not take NULLS into account)

SQL: Average with NULL - Example #35 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	None
3	Cohen	David	22	economy	2019-02-12	19	None
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	music	2019-12-10	15	60

```
SELECT COUNT(COALESCE(score,0)), AVG(COALESCE(score,0))
FROM table3
```

SQL: Average with NULL - Example #35 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	None
3	Cohen	David	22	economy	2019-02-12	19	None
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	music	2019-12-10	15	60

```
SELECT COUNT(COALESCE(score,0)),
AVG(COALESCE(score,0))
FROM table3
```

COUNT(COALESCE(score,0))	AVG(COALESCE(score,0))
7	61.285714285714285

We can decide that NULL are equal 0

SQL: Average with NULL - Example #36 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	None
3	Cohen	David	22	economy	2019-02-12	19	None
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	music	2019-12-10	15	60

```
SELECT COUNT(COALESCE(score,55)), AVG(COALESCE(score,55))
FROM table3
```

SQL: Average with NULL - Example #36 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	None
3	Cohen	David	22	economy	2019-02-12	19	None
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	music	2019-12-10	15	60

```
SELECT COUNT(COALESCE(score,55)),
AVG(COALESCE(score,55))
FROM table3
```

COUNT(COALESCE(score,55))	AVG(COALESCE(score,55))
7	77.0

and here we consider as 55 (count is same, average is different)

Learn SQL by Example (Chananel Perel 2023)

ROLLING SUM (WITHOUT WINDOW FUNCTIONS)

SQL: Rolling Sum (without Window Functions) - Example #37 - Q:

day	amount
1	4
2	3
3	-2
4	1
5	7

```
SELECT *
FROM sales s1
JOIN sales s2
  ON s1.day >= s2.day
ORDER BY 1,3
```

SQL: Rolling Sum (without Window Functions) - Example #37 - A:

day	amount
1	4
2	3
3	-2
4	1
5	7

```
SELECT *
FROM sales s1
JOIN sales s2
  ON s1.day >= s2.day
ORDER BY 1,3
```

day	amount	day	amount
1	4	1	4
2	3	1	4
2	3	2	3
3	-2	1	4
3	-2	2	3
3	-2	3	-2
4	1	1	4
4	1	2	3
4	1	3	-2
4	1	4	1
5	7	1	4
5	7	2	3
5	7	3	-2
5	7	4	1
5	7	5	7

o JOIN also with < or > and not just =

SQL: Rolling Sum (without Window Functions) - Example #38 - Q:

day	amount
1	4
2	3
3	-2
4	1
5	7

```
SELECT s1.day, SUM(s2.amount)
FROM sales s1
JOIN sales s2
  ON s1.day >= s2.day
GROUP BY s1.day
ORDER BY 1
```

SQL: Rolling Sum (without Window Functions) - Example #38 - A:

day	amount
1	4
2	3
3	-2
4	1
5	7

```
SELECT s1.day, SUM(s2.amount)
FROM sales s1
JOIN sales s2
  ON s1.day >= s2.day
GROUP BY s1.day
ORDER BY 1
```

day	SUM(s2.amount)
1	4
2	7
3	5
4	6
5	13

and here we see the requested rolling sum

SQL: Rolling Sum (without Window Functions) - Example #39 - Q:

day	amount
1	4
2	3
3	-2
4	1
5	7

```
SELECT *,
       SUM(amount) OVER (ORDER BY day)
FROM sales s1
ORDER BY 1
```

SQL: Rolling Sum (without Window Functions) - Example #39 - A:

day	amount
1	4
2	3
3	-2
4	1
5	7

```
SELECT *,
       SUM(amount) OVER (ORDER BY day)
FROM sales s1
ORDER BY 1
```

day	amount	SUM(amount) OVER (ORDER BY day)
1	4	4
2	3	7
3	-2	5
4	1	6
5	7	13

Just for comparison we show the Window Function (advanced topic)

Learn SQL by Example (Chananel Perel 2023)

NAME AND ID OF MAX SCORE

SQL: Name and Id of MAX score - Example #40 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n, MAX(score)
FROM table1
```

SQL: Name and Id of MAX score - Example #40 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n, MAX(score)
FROM table1
```

id	last_n	first_n	MAX(score)
4	Levi	Moshe	100

Works by accident.. In other SQL servers this returns error

SQL: Name and Id of MAX score - Example #41 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n, AVG(score)
FROM table1
```

SQL: Name and Id of MAX score - Example #41 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n, AVG(score)
FROM table1
```

id	last_n	first_n	AVG(score)
1	Cohen	David	83.22222222222223

If you use AVG, it shows randomly some values

SQL: Name and Id of MAX score - Example #42 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n, MAX(score), MIN(cost)
FROM table1
```

SQL: Name and Id of MAX score - Example #42 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n, MAX(score),
MIN(cost)
FROM table1
```

id	last_n	first_n	MAX(score)	MIN(cost)
7	Peled	Chaim	100	15

and now it does not work as only one id can be returned.
To conclde: if doing aggregation/GROUP BY, you can only SELECT the grouped by columns or a

SQL: Name and Id of MAX score - Example #43 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n
FROM table1
WHERE score = (SELECT MAX(score) FROM table1)
```

SQL: Name and Id of MAX score - Example #43 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n
FROM table1
WHERE score = (SELECT MAX(score) FROM table1)
```

id	last_n	first_n
4	Levi	Moshe

This is OK. Some other SQL servers may have functions for this, like: max_by etc..

SQL: Name and Id of MAX score - Example #44 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n
FROM table1
WHERE age = (SELECT MIN(age) FROM table1)
```

SQL: Name and Id of MAX score - Example #44 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n
FROM table1
WHERE age = (SELECT MIN(age) FROM
table1)
```

id	last_n	first_n
1	Cohen	David
2	Cohen	David
3	Cohen	David

We can see that this also returns more than 1 line, if we have..

Learn SQL by Example (Chananel Perel 2023)

NAME AND ID OF 2 HIGHEST SCORE

SQL: Name and Id of 2 HIGHEST score - Example #45 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n, score
FROM table1
ORDER BY score DESC
LIMIT 2
```

SQL: Name and Id of 2 HIGHEST score - Example #45 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n, score
FROM table1
ORDER BY score DESC
LIMIT 2
```

id	last_n	first_n	score
4	Levi	Moshe	100
1	Cohen	David	99

doing like this is even easier, but good only for MIN, MAX, and N highest/lowest..
(not good for average e.g.)

Learn SQL by Example (Chananel Perel 2023)

NAME AND ID OF MAX SCORE, ANOTHER OPTION

SQL: Name and Id of MAX score, another option - Example #46 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n, score
FROM table1
ORDER BY score DESC
LIMIT 1
```

SQL: Name and Id of MAX score, another option - Example #46 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT id,last_n,first_n, score
FROM table1
ORDER BY score DESC
LIMIT 1
```

id	last_n	first_n	score
4	Levi	Moshe	100

another easy option.. (not good if you need GROUP BY, and then you may want MAX for each g

Learn SQL by Example (Chananel Perel 2023)

GET MIN MAX IN 2 COLS

SQL: Get Min Max in 2 cols - Example #47 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT MIN(score) min_score,
       MAX(score) max_score
FROM table1
```

SQL: Get Min Max in 2 cols - Example #47 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT MIN(score) min_score,
       MAX(score) max_score
FROM table1
```

min_score	max_score
60	100

Easy..

Learn SQL by Example (Chananel Perel 2023)

GET MIN MAX IN 2 ROWS

SQL: Get Min Max in 2 rows - Example #48 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT 'min_score' key, MIN(score) val FROM table1
UNION
SELECT 'max_score' key, MAX(score) val FROM table1
```

SQL: Get Min Max in 2 rows - Example #48 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
    SELECT 'min_score' key, MIN(score) val
FROM table1
    UNION
    SELECT 'max_score' key, MAX(score) val
FROM table1
```

key	val
max_score	100
min_score	60

Will we have any difference if we use UNION ALL ??

Learn SQL by Example (Chananel Perel 2023)

COUNT DUPLICATE LINES 1

SQL: COUNT Duplicate Lines 1 - Example #49 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT
  (SELECT count(*) FROM table1) -
  (SELECT count(*) FROM (SELECT DISTINCT last_n, first_n
FROM table1)) count_dup
```

SQL: COUNT Duplicate Lines 1 - Example #49 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT
  (SELECT count(*) FROM table1) -
  (SELECT count(*) FROM (SELECT DISTINCT
last_n, first_n FROM table1)) count_dup
```

count_dup
4

count all minus (-) count distinct

Learn SQL by Example (Chananel Perel 2023)

COUNT DUPLICATE LINES 2

SQL: COUNT Duplicate Lines 2 - Example #50 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT last_n, first_n
FROM table1
GROUP BY last_n, first_n
HAVING COUNT(*) > 1
```

SQL: COUNT Duplicate Lines 2 - Example #50 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
SELECT last_n, first_n
FROM table1
GROUP BY last_n, first_n
HAVING COUNT(*) > 1
```

last_n	first_n
Cohen	David
Cohen	Moshe
Peled	Chaim

Here we can see the duplicate names we have

SQL: COUNT Duplicate Lines 2 - Example #51 - Q:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
WITH  dup_names AS (
      SELECT last_n, first_n
      FROM table1
      GROUP BY last_n, first_n
      HAVING COUNT(*) > 1
  )
  SELECT COUNT(*)
  FROM dup_names
```

SQL: COUNT Duplicate Lines 2 - Example #51 - A:

id	last_n	first_n	age	course	to_date	cost	score
1	Cohen	David	22	math	2019-11-01	17	99
2	Cohen	David	22	history	2019-09-01	18	88
3	Cohen	David	22	economy	2019-02-12	19	77
4	Levi	Moshe	24	math	2019-11-01	22	100
5	Cohen	Moshe	23	music	2020-01-01	20	80
6	Cohen	Moshe	23	math	2020-01-20	21	90
7	Peled	Chaim	27	history	2019-12-10	15	60
8	Peled	Chaim	27	math	2019-10-20	16	70
9	Gamil	David	25	history	2019-08-30	17	85

```
WITH  dup_names AS (
      SELECT last_n, first_n
      FROM table1
      GROUP BY last_n, first_n
      HAVING COUNT(*) > 1
  )
  SELECT COUNT(*)
  FROM dup_names
```

COUNT(*)
3

.. and now we can count them

Learn SQL by Example (Chananel Perel 2023)

REMOVE DUPLICATE LINES

SQL: REMOVE Duplicate Lines - Example #52 - Q:

item	id
aa	11
aa	11
bb	22
ee	55

```
SELECT DISTINCT *
FROM t_1
```

SQL: REMOVE Duplicate Lines - Example #52 - A:

item	id
aa	11
aa	11
bb	22
ee	55

```
SELECT DISTINCT *
FROM t_1
```

item	id
aa	11
bb	22
ee	55

The easiest way, use DISTINCT..

SQL: REMOVE Duplicate Lines - Example #53 - Q:

item	id
aa	11
aa	11
bb	22
ee	55

```
SELECT * FROM t_1
INTERSECT
SELECT * FROM t_1
```

SQL: REMOVE Duplicate Lines - Example #53 - A:

item	id
aa	11
aa	11
bb	22
ee	55

```
SELECT * FROM t_1
INTERSECT
SELECT * FROM t_1
```

item	id
aa	11
bb	22
ee	55

INTERSECT runs on sets(=each memer is unique) so removes duplicates
If you use this - write comment!

SQL: REMOVE Duplicate Lines - Example #54 - Q:

item	id
aa	11
aa	11
bb	22
ee	55

```
SELECT * FROM t_1
UNION
SELECT * FROM t_1
```

SQL: REMOVE Duplicate Lines - Example #54 - A:

item	id
aa	11
aa	11
bb	22
ee	55

```
SELECT * FROM t_1
UNION
SELECT * FROM t_1
```

item	id
aa	11
bb	22
ee	55

So, We can technically even use UNION..

SQL: REMOVE Duplicate Lines - Example #55 - Q:

item	id	val
aa	11	17
aa	11	19
bb	22	20
ee	55	21

```
SELECT DISTINCT *
FROM t_1
```

SQL: REMOVE Duplicate Lines - Example #55 - A:

item	id	val
aa	*11*	*17*
aa	*11*	*19*
bb	*22*	*20*
ee	*55*	*21*

```
SELECT DISTINCT *
FROM t_1
```

item	id	val
aa	11	17
aa	11	19
bb	22	20
ee	55	21

If we have column, that we do not want as part of the unique

SQL: REMOVE Duplicate Lines - Example #56 - Q:

item	id	val
aa	11	17
aa	11	19
bb	22	20
ee	55	21

```
SELECT DISTINCT item,id
FROM t_1
```

SQL: REMOVE Duplicate Lines - Example #56 - A:

item	id	val
aa	*11*	*17*
aa	*11*	*19*
bb	*22*	*20*
ee	*55*	*21*

```
SELECT DISTINCT item,id
FROM t_1
```

item	id
aa	11
bb	22
ee	55

But now we do not see the val

SQL: REMOVE Duplicate Lines - Example #57 - Q:

item	id	val
aa	11	17
aa	11	19
bb	22	20
ee	55	21

```
SELECT item,id,
       MAX(val) val
FROM t_1
GROUP BY item, id
```

SQL: REMOVE Duplicate Lines - Example #57 - A:

item	id	val
aa	11	17
aa	11	19
bb	22	20
ee	55	21

```
SELECT item,id,
       MAX(val) val
FROM t_1
GROUP BY item, id
```

item	id	val
aa	11	19
bb	22	20
ee	55	21

So we should use GROUP BY, and aggregate functions..

Learn SQL by Example (Chananel Perel 2023)

SIMULATE FULL (OUTER) JOIN

SQL: Simulate FULL (OUTER) JOIN - Example #58 - Q:

id	name	mngr_id
1001	Dana	1001
1002	David	1001
1003	Eli	1001
1004	Sara	1003
1005	Ben	1003

emp_id	proj_id	hours
1001	10	50
1001	11	40
1001	12	60
1002	10	110
1002	13	140
1003	11	120
1004	14	190
1009	14	190

```
SELECT e.*, w.*
FROM employees e
LEFT JOIN works w
  ON e.id = w.emp_id
UNION
SELECT e.*, w.*
FROM works w
LEFT JOIN employees e
  ON e.id = w.emp_id
ORDER BY id
```

SQL: Simulate FULL (OUTER) JOIN - Example #58 - A:

id	name	mngr_id
1001	Dana	1001
1002	David	1001
1003	Eli	1001
1004	Sara	1003
1005	Ben	1003

emp_id	proj_id	hours
1001	10	50
1001	11	40
1001	12	60
1002	10	110
1002	13	140
1003	11	120
1004	14	190
1009	14	190

```
SELECT e.*, w.*
FROM employees e
LEFT JOIN works w
  ON e.id = w.emp_id
UNION
SELECT e.*, w.*
FROM works w
LEFT JOIN employees e
  ON e.id = w.emp_id
ORDER BY id
```

id	name	mngr_id	emp_id	proj_id	hours
None	None	None	1009	14	190
1001	Dana	1001	1001	10	50
1001	Dana	1001	1001	11	40
1001	Dana	1001	1001	12	60
1002	David	1001	1002	10	110
1002	David	1001	1002	13	140
1003	Eli	1001	1003	11	120
1004	Sara	1003	1004	14	190
1005	Ben	1003	None	None	None

Can I use UNION ALL??

Learn SQL by Example (Chananel Perel 2023)

LEFT JOIN - ON VS WHERE

SQL: LEFT JOIN - ON vs WHERE - Example #59 - Q:

id	name	mngr_id
1001	Dana	1001
1002	David	1001
1003	Eli	1001
1004	Sara	1003
1005	Ben	1003

emp_id	proj_id	hours
1001	10	50
1001	11	40
1001	12	60
1002	10	110
1002	13	140
1003	11	120
1004	14	190
1009	14	190

```
SELECT e.*, w.*
FROM employees e
LEFT JOIN works w
  ON e.id = w.emp_id
     AND proj_id > 10
```

SQL: LEFT JOIN - ON vs WHERE - Example #59 - A:

id	name	mngr_id
1001	Dana	1001
1002	David	1001
1003	Eli	1001
1004	Sara	1003
1005	Ben	1003

emp_id	proj_id	hours
1001	10	50
1001	11	40
1001	12	60
1002	10	110
1002	13	140
1003	11	120
1004	14	190
1009	14	190

```
SELECT e.*, w.*
FROM employees e
LEFT JOIN works w
  ON e.id = w.emp_id
     AND proj_id > 10
```

id	name	mngr_id	emp_id	proj_id	hours
1001	Dana	1001	1001	11	40
1001	Dana	1001	1001	12	60
1002	David	1001	1002	13	140
1003	Eli	1001	1003	11	120
1004	Sara	1003	1004	14	190
1005	Ben	1003	None	None	None

ON runs as part of the JOIN

SQL: LEFT JOIN - ON vs WHERE - Example #60 - Q:

id	name	mngr_id
1001	Dana	1001
1002	David	1001
1003	Eli	1001
1004	Sara	1003
1005	Ben	1003

emp_id	proj_id	hours
1001	10	50
1001	11	40
1001	12	60
1002	10	110
1002	13	140
1003	11	120
1004	14	190
1009	14	190

```
SELECT e.*, w.*
FROM employees e
LEFT JOIN works w
  ON e.id = w.emp_id
WHERE proj_id > 10
```

SQL: LEFT JOIN - ON vs WHERE - Example #60 - A:

id	name	mngr_id
1001	Dana	1001
1002	David	1001
1003	Eli	1001
1004	Sara	1003
1005	Ben	1003

emp_id	proj_id	hours
1001	10	50
1001	11	40
1001	12	60
1002	10	110
1002	13	140
1003	11	120
1004	14	190
1009	14	190

```
SELECT e.*, w.*
FROM employees e
LEFT JOIN works w
  ON e.id = w.emp_id
WHERE proj_id > 10
```

id	name	mngr_id	emp_id	proj_id	hours
1001	Dana	1001	1001	11	40
1001	Dana	1001	1001	12	60
1002	David	1001	1002	13	140
1003	Eli	1001	1003	11	120
1004	Sara	1003	1004	14	190

WHERE runs after the JOIN