Subqueries

Anthony Kleerekoper

Databases 6G4Z0016

Emerald Lies by Marillion

Reminder: Filtering the Groups: HAVING

Do you need all the groups?

HAVING is like WHERE for GROUP BY expressions

Specify a condition that each group must meet before being included in the output

But why not use a WHERE clause?

When WHERE does not work

WHERE works row-by-row

Can only use information available on that same row

snum	stu_name	dob	points	size_hs
003	Jack Fines	2002-09-12	110	60
009	Michelle Jones	2001-12-22	114	50
017	Nazia Hassan	2002-05-05	101	50
022	Shane Jordan	2001-10-10	121	35
035	Peter Watson	2002-06-29	117	45

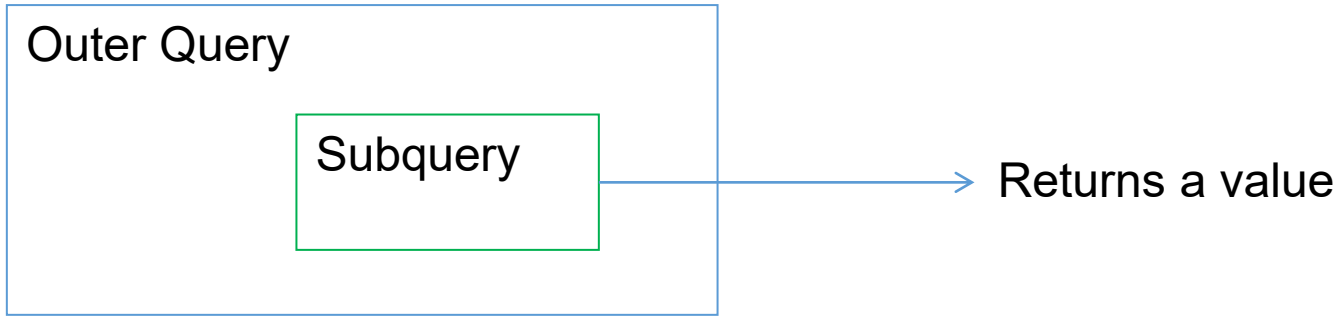
```
SELECT stu_name
FROM Students
WHERE points < AVG(points);
```

Won’t work because AVG(points) is not available row-by-row

Subqueries

We can use the result of one query inside another one

A query inside another one is called a **subquery**



The subquery is executed first and its result is used in the outer query

Subquery

Is the WHERE condition dependent on data on a different row or rows?

Use a ***subquery***

Compute the result of a query and make that result available in another query

snum	stu_name	dob	points	size_hs
003	Jack Fines	2002-09-12	110	60
009	Michelle Jones	2001-12-22	114	50
017	Nazia Hassan	2002-05-05	101	50
022	Shane Jordan	2001-10-10	121	35
035	Peter Watson	2002-06-29	117	45

```
SELECT stu_name
FROM Students
WHERE points <
      (SELECT AVG(points)
       FROM Students);
```

stu_name
Jack Fines
Nazia Hassan

Subquery

Imagine replacing the subquery with its result

snum	stu_name	dob	points	size_hs
003	Jack Fines	2002-09-12	110	60
009	Michelle Jones	2001-12-22	114	50
017	Nazia Hassan	2002-05-05	101	50
022	Shane Jordan	2001-10-10	121	35
035	Peter Watson	2002-06-29	117	45

```
SELECT stu_name
FROM Students
WHERE points <
      (SELECT AVG(points)
       FROM Students)
```

AVG(points)
112.6

```
SELECT stu_name
FROM Students
WHERE points <
      112.6;
```

Designing a Subquery

1. Design the outer query with a placeholder
2. Design the inner query
3. Replace the placeholder with the inner query inside brackets

```
SELECT stu_name
FROM Students
WHERE size_hs = "number of rows*10"
```

placeholder

```
SELECT COUNT(*)*10
FROM Students
```

subquery

```
SELECT stu_name
FROM Students
WHERE size_hs =
       (SELECT COUNT(*)*10
        FROM Students);
```

Subqueries in Other Clauses

Most subqueries appear in the WHERE clause

But can also be used in:

SELECT
FROM
JOIN
ORDER BY
HAVING

Subquery in SELECT Clause

Write a query to list all students and the difference between their date of birth and Nazia Hassan's

```
SELECT stu_name, dob,
    DATEDIFF(dob, (SELECT dob
        FROM Students
        WHERE stu_name = 'Nazia Hassan'))
    AS 'dob diff'
FROM Students;
```

stu_name	dob	dob diff
Jack Fines	2002-09-12	130
Michelle Jones	2001-12-22	-134
Nazia Hassan	2002-05-05	0
Shane Jordan	2001-10-10	-207
Peter Watson	2002-06-29	55

Universities (uni_name, city, enrolment, app_deadline)

uni_name	city	enrolment	app_deadline
John Moores	Liverpool	17,835	2022-09-22
Man Met	Manchester	25,810	2022-09-15
Salford Uni	Salford	14,895	2022-09-18
Uni of Manchester	Manchester	26,725	2022-09-20

Applications (snum, uni_name, course, decision)

snum	uni_name	course	
003	Man Met	Computing	
003	Man Met	Computer Science	
009	Uni of Manchester	Computer Science	
017	Man Met	Computing	Reject
017	Salford Uni	Computing	Accept
022	Man Met	Computing	Accept

				hs
				60
				50
				50
022	Shane Jordan	2001-10-10	121	35
035	Peter Watson	2002-06-29	117	45

Subquery in FROM Clause

```
SELECT stu_name, dob,
   DATEDIFF(dob,naz_dob) AS 'dob diff'
FROM (SELECT dob AS naz_dob FROM Students
      WHERE stu_name = 'Nazia Hassan') AS naz_tbl
CROSS JOIN Students;
```

stu_name	dob	dob diff
Jack Fines	2002-09-12	130
Michelle Jones	2001-12-22	-134
Nazia Hassan	2002-05-05	0
Shane Jordan	2001-10-10	-207
Peter Watson	2002-06-29	55

Universities (uni_name, city, enrolment, app_deadline)

uni_name	city	enrolment	app_deadline
John Moores	Liverpool	17,835	2022-09-22
Man Met	Manchester	25,810	2022-09-15
Salford Uni	Salford	14,895	2022-09-18
Uni of Manchester	Manchester	26,725	2022-09-20

Applications (snum, uni_name, course, decision)

snum	uni_name	course	
003	Man Met	Computing	Accept
003	Man Met	Computer Science	Accept
009	Uni of Manchester	Computer Science	Reject
017	Man Met	Computing	Reject
017	Salford Uni	Computing	Accept
022	Man Met	Computing	Accept

				hs
003	Jack Fines	2002-09-12	110	60
009	Michelle Jones	2001-12-22	114	50
017	Nazia Hassan	2002-05-05	101	50
022	Shane Jordan	2001-10-10	121	35
035	Peter Watson	2002-06-29	117	45

Subquery in JOIN Clause

```
SELECT stu_name, dob,
   DATEDIFF(dob,naz_dob) AS 'dob diff'
FROM Students
CROSS JOIN (SELECT dob AS naz_dob FROM Students
   WHERE stu_name = 'Nazia Hassan')AS naz_tbl;
```

stu_name	dob	dob diff
Jack Fines	2002-09-12	130
Michelle Jones	2001-12-22	-134
Nazia Hassan	2002-05-05	0
Shane Jordan	2001-10-10	-207
Peter Watson	2002-06-29	55

Universities (uni_name, city, enrolment, app_deadline)

uni_name	city	enrolment	app_deadline
John Moores	Liverpool	17,835	2022-09-22
Man Met	Manchester	25,810	2022-09-15
Salford Uni	Salford	14,895	2022-09-18
Uni of Manchester	Manchester	26,725	2022-09-20

Applications (snum, uni_name, course, decision)

snum	uni_name	course	
003	Man Met	Computing	
003	Man Met	Computer Science	
009	Uni of Manchester	Computer Science	Reject
017	Man Met	Computing	Reject
017	Salford Uni	Computing	Accept
022	Man Met	Computing	Accept

				hs
				60
				50
017	Nazia Hassan	2002-05-05	101	50
022	Shane Jordan	2001-10-10	121	35
035	Peter Watson	2002-06-29	117	45

vevox.app ID: 104-878-117

Multi-Row Subqueries

A subquery can return many rows

treat result as a list, using IN keyword

snum	stu_name	dob	points	size_hs
003	Jack Fines	2002-09-12	110	60
009	Michelle Jones	2001-12-22	114	50
017	Nazia Hassan	2002-05-05	101	50
022	Shane Jordan	2001-10-10	121	35
035	Peter Watson	2002-06-29	117	45

```
SELECT stu_name
FROM Students
WHERE snum IN
    (SELECT snum
     FROM Applications);
```

stu_name
Jack Fines
Michelle Jones
Nazia Hassan
Shane Jordan

Multi-Row Subqueries

```
SELECT stu_name
FROM Students
WHERE snum IN
   (SELECT snum
    FROM Applications);
```

snum
003
003
009
017
017
022

```
SELECT stu_name
FROM Students
WHERE snum IN
    (003,003,009,017,017,022);
```

stu_name
Jack Fines
Michelle Jones
Nazia Hassan
Shane Jordan

ANY and ALL

Two new keywords for dealing with lists: ANY and ALL

Used in conjunction with other comparators

```
SELECT snum, stu_name, points, size_hs
FROM Students
WHERE points < ANY (110, 115, 120);
```

snum	stu_name	points	size_hs
003	Jack Fines	110	60
009	Michelle Jones	114	50
017	Nazia Hassan	101	50
035	Peter Watson	117	45

ANY and ALL

No added functionality
but maybe clearer in subqueries

ANY or ALL	Equivalent
< ANY	< MAX
> ANY	> MIN
< ALL	< MIN
> ALL	> MAX
= ANY	IN
!= ALL	NOT IN

Not supported by MariaDB
Except when used with subqueries

NULL Values and Subqueries

Important: Anything compared to NULL returns NULL

If the subquery contains a NULL:

- IN and ANY will still work with the non-NULL values
- ALL will fail because have to compare to the NULL value

Also a problem with single-row subqueries

Can include the condition `WHERE x IS NOT NULL` to be safe

Subqueries and Joins

Both subqueries and joins allow us to include data from two tables

Very often we can rewrite a subquery using a join instead

```
SELECT stu_name
FROM Students
WHERE snum IN
    (SELECT snum
     FROM Applications);
```

Equivalent

```
SELECT stu_name
FROM Students
INNER JOIN Applications
  USING(snum);
```

Join may be faster

Subquery may be more readable

Correlated Subqueries

So far the subqueries have been independent of the outer query

- The subquery is valid by itself
- The subquery is run first

But you can use values from the outer query inside the subquery

- Called a ***correlated subquery***

The subquery is evaluated once for each row in the outer query

- The outer query is partially evaluated first

Example: Correlated Subquery

List all students and how many students were born after them

```
SELECT stu_name, dob,
    (SELECT COUNT(*)
     FROM Students s2
     WHERE s2.dob > s1.dob)
    AS Num_After
FROM Students s1;
```

stu_name	dob	Num_After
Jack Fines	2002-09-12	0
Michelle Jones	2001-12-22	3
Nazia Hassan	2002-05-05	2
Shane Jordan	2001-10-10	4
Peter Watson	2002-06-29	1

Universities (uni_name, city, enrolment, app_deadline)

uni_name	city	enrolment	app_deadline
John Moores	Liverpool	17,835	2022-09-22
Man Met	Manchester	25,810	2022-09-15
Salford Uni	Salford	14,895	2022-09-18
Uni of Manchester	Manchester	26,725	2022-09-20

Applications (snum, uni_name, course, decision)

snum	uni_name	course	decision
003	Man Met	Computing	Accept
003	Man Met	Computer Science	Accept
009	Uni of Manchester	Computer Science	Reject
017	Man Met	Computing	Reject
017	Salford Uni	Computing	Accept
022	Man Met	Computing	Accept

Students (snum, stu_name, dob, points, size_hs)

snum	stu_name	dob	points	size_hs
003	Jack Fines	2002-09-12	110	60
009	Michelle Jones	2001-12-22	114	50
017	Nazia Hassan	2002-05-05	101	50
022	Shane Jordan	2001-10-10	121	35
035	Peter Watson	2002-06-29	117	45

Correlated Subquery Explained I

List all students and how many students were born after them

```
SELECT stu_name, dob,
    (SELECT COUNT(*)
     FROM Students s2
     WHERE s2.dob > s1.dob)
    AS Num_After
FROM Students s1;
```

stu_name	dob
Jack Fines	2002-09-12

Num_After
0

```
SELECT COUNT(*)
FROM Students s2
WHERE s2.dob > '2002-09-12';
```

Universities (uni_name, city, enrolment, app_deadline)

uni_name	city	enrolment	app_deadline
John Moores	Liverpool	17,835	2022-09-22
Man Met	Manchester	25,810	2022-09-15
Salford Uni	Salford	14,895	2022-09-18
Uni of Manchester	Manchester	26,725	2022-09-20

Applications (snum, uni_name, course, decision)

snum	uni_name	course	decision
003	Man Met	Computing	Accept
003	Man Met	Computer Science	Accept
009	Uni of Manchester	Computer Science	Reject
017	Man Met	Computing	Reject
017	Salford Uni	Computing	Accept
022	Man Met	Computing	Accept

Students (snum, stu_name, dob, points, size_hs)

snum	stu_name	dob	points	size_hs
003	Jack Fines	2002-09-12	110	60
009	Michelle Jones	2001-12-22	114	50
017	Nazia Hassan	2002-05-05	101	50
022	Shane Jordan	2001-10-10	121	35
035	Peter Watson	2002-06-29	117	45

Correlated Subquery Explained II

List all students and how many students were born after them

```
SELECT stu_name, dob,
    (SELECT COUNT(*)
     FROM Students s2
     WHERE s2.dob > s1.dob)
    AS Num_After
FROM Students s1;
```

stu_name	dob
Michelle Jones	2001-12-22

Num_After
3

```
SELECT COUNT(*)
FROM Students s2
WHERE s2.dob > '2001-12-22';
```

Universities (uni_name, city, enrolment, app_deadline)

uni_name	city	enrolment	app_deadline
John Moores	Liverpool	17,835	2022-09-22
Man Met	Manchester	25,810	2022-09-15
Salford Uni	Salford	14,895	2022-09-18
Uni of Manchester	Manchester	26,725	2022-09-20

Applications (snum, uni_name, course, decision)

snum	uni_name	course	decision
003	Man Met	Computing	Accept
003	Man Met	Computer Science	Accept
009	Uni of Manchester	Computer Science	Reject
017	Man Met	Computing	Reject
017	Salford Uni	Computing	Accept
022	Man Met	Computing	Accept

Students (snum, stu_name, dob, points, size_hs)

snum	stu_name	dob	points	size_hs
003	Jack Fines	2002-09-12	110	60
009	Michelle Jones	2001-12-22	114	50
017	Nazia Hassan	2002-05-05	101	50
022	Shane Jordan	2001-10-10	121	35
035	Peter Watson	2002-06-29	117	45

Correlated Subquery Explained III

List all students and how many students were born after them

```
SELECT stu_name, dob,
    (SELECT COUNT(*)
     FROM Students s2
     WHERE s2.dob > s1.dob)
    AS Num_After
FROM Students s1;
```

stu_name	dob
Nazia Hassan	2002-05-05

Num_After
2

```
SELECT COUNT(*)
FROM Students s2
WHERE s2.dob > '2002-05-05';
```

Universities (uni_name, city, enrolment, app_deadline)

uni_name	city	enrolment	app_deadline
John Moores	Liverpool	17,835	2022-09-22
Man Met	Manchester	25,810	2022-09-15
Salford Uni	Salford	14,895	2022-09-18
Uni of Manchester	Manchester	26,725	2022-09-20

Applications (snum, uni_name, course, decision)

snum	uni_name	course	decision
003	Man Met	Computing	Accept
003	Man Met	Computer Science	Accept
009	Uni of Manchester	Computer Science	Reject
017	Man Met	Computing	Reject
017	Salford Uni	Computing	Accept
022	Man Met	Computing	Accept

Students (snum, stu_name, dob, points, size_hs)

snum	stu_name	dob	points	size_hs
003	Jack Fines	2002-09-12	110	60
009	Michelle Jones	2001-12-22	114	50
017	Nazia Hassan	2002-05-05	101	50
022	Shane Jordan	2001-10-10	121	35
035	Peter Watson	2002-06-29	117	45

WHERE EXISTS

Used when checking if data exists without caring what it is

Example: Which universities had applications?

Do not care how many, so long as at least 1

```
SELECT uni_name, city, enrolment, app_deadline
FROM Universities
WHERE EXISTS (SELECT * FROM Applications
              WHERE uni_name = Universities.uni_name);
```

uni_name	city	enrolment	app_deadline
Man Met	Manchester	25,810	2022-09-15
Salford Uni	Salford	14,895	2022-09-18
Uni of Manchester	Manchester	26,725	2022-09-20

Reusing a subquery

Sometimes use the same subquery more than once

for example, using it again in ORDER BY clause

Use a method called ***Common Table Expression (CTEs)***

Write the subquery once before SELECT clause, give it a name

Can refer to it by name many times

Example: Common Table Expression

```
WITH average_points AS
     (SELECT AVG(points)AS avg
     FROM Students)
SELECT stu_name
FROM Students
WHERE points < (SELECT avg FROM average_points);
```

Note: You have to `SELECT ... FROM` the named subquery

A Second Example

List every student, their points and the difference between their points and the maximum points of any student

```
WITH mx_p AS
 (SELECT MAX(points) AS max FROM Students)
SELECT stu_name, points, (SELECT * FROM mx_p) AS 'Max',
     points - (SELECT max FROM mx_p) AS 'Diff'
FROM Students
ORDER BY points - (SELECT max FROM mx_p) DESC;
```

stu_name	points	Max	Diff
Shane Jordan	121	121	0
Peter Watson	117	121	-4
Michelle Jones	114	121	-7
Jack Fines	110	121	-11
Nazia Hassan	101	121	-20