Grouping, Subqueries and Outer Joins

Anthony Kleerekoper

Database Systems 6G5Z1111

Reminder: Changing the Data

During a query you can change:

- column and table names (aliases)
- The data itself (using row functions)

These changes are temporary

They underlying data is not changed

Group (aggregate) Functions

Group functions operate on *sets of rows* to give one result per group

AVG (average)
COUNT
MAX (maximum)
MIN (minimum)
SUM

Example: Group Functions

Students:

snum	stu_name	points	size_hs
003	Jack Fines	110	60
009	Michelle Jones	114	50
017	Nazia Hassan	101	50
022	Shane Jordan	121	35
035	Peter Watson	117	45

```
SELECT AVG(points)
FROM Students;
```

AVG(points)
112.6

```
SELECT COUNT(points)
FROM Students;
```

COUNT(points)
5

A Word on Counting

Group functions ignore NULL values

Use `COUNT(*)` to count all rows

even those with some NULL values

Use `COUNT(DISTINCT col1)` to count unique values in a column

Including NULL Values

Group functions ignore NULL values

Use the function `IFNULL(col, val)` to provide a value for `NULLS`

```
SELECT AVG(points)
FROM Students;

Gives 117.33
```

```
SELECT AVG(IFNULL(points,0))
FROM Students;

Gives 70.4
```

Students

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

Common Mistake: Mixing groups and non-groups

If you try the following query, you get a random value in `stu_name`:

```
SELECT stu_name, AVG(points) FROM Students;
```

Could return many rows

Can only return 1 row

Some databases (e.g. Oracle) will throw an error

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

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

GROUP BY

Suppose you wanted to find the average points of students in different sized high schools

```
SELECT AVG(points)FROM Students;
```

gives 1 row – the average of all students

We want to first split the rows into groups with the same size high school

Then apply the AVG() function to each group individually

Use: `GROUP BY size_hs`

```
SELECT size_hs, AVG(points)
FROM Students
GROUP BY size_hs;
```

1. Split into groups based on `size_hs`
2. Apply "`SELECT size_hs, AVG(points)`" to each group

snum	stu_name	points	size_hs
003	Jack Fines	110	60
009	Michelle Jones	114	50
017	Nazia Hassan	101	50
022	Shane Jordan	121	35
035	Peter Watson	117	45

snum	stu_name	points	size_hs
003	Jack Fines	110	60

snum	stu_name	points	size_hs
009	Michelle Jones	114	50
017	Nazia Hassan	101	50

snum	stu_name	points	size_hs
022	Shane Jordan	121	35

snum	stu_name	points	size_hs
035	Peter Watson	117	45

```
SELECT size_hs, AVG(points)
FROM Students
GROUP BY size_hs;
```

1. Split into groups based on `size_hs`

2. Apply "`SELECT size_hs, AVG(points)`" to each group

snum	stu_name	points	size_hs
003	Jack Fines	110	60
009	Michelle Jones	114	50
017	Nazia Hassan	101	50
022	Shane Jordan	121	35
035	Peter Watson	117	45

size_hs	AVG(points)
60	110

size_hs	AVG(points)
50	107.5

size_hs	AVG(points)
35	121

size_hs	AVG(points)
45	117

```
SELECT size_hs, AVG(points)
FROM Students
GROUP BY size_hs;
```

1. Split into groups based on `size_hs`

2. Apply "`SELECT size_hs, AVG(points)`" to each group

snum	stu_name	points	size_hs
003	Jack Fines	110	60
009	Michelle Jones	114	50
017	Nazia Hassan	101	50
022	Shane Jordan	121	35
035	Peter Watson	117	45

size_hs	AVG(points)
60	110
50	107.5
35	121
45	117

Example 2: GROUP BY

```
SELECT city, MAX(enrolment)
FROM Universities
GROUP BY city;
```

city	MAX(enrolment)
Manchester	26,725
Salford	14,895
Liverpool	17,835

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

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

Common Mistake: Non-Group in Select List

```
SELECT snum, size_hs, AVG(points)
FROM Students
GROUP BY size_hs;
```

Q1. How many different values of `size_hs` are there in each group?

Q2. How many different values of `snum` are there in each group?

Common Mistake: Non-Group in Select List

Up to 2 rows per group

1 row per group

```
SELECT snum, size_hs, AVG(points)
FROM Students
GROUP BY size_hs;
```

Get a random value in `snum` column

Some databases give an error

snum	stu_name	points	size_hs
003	Jack Fines	110	60
009	Michelle Jones	114	50
017	Nazia Hassan	101	50
022	Shane Jordan	121	35
035	Peter Watson	117	45

GROUP BY: Multiple Columns that Work

```
SELECT size_hs, AVG(points),
       COUNT(stu_name)
FROM Students
GROUP BY size_hs;
```

Works fine!

How many values of `size_hs` are there per group?

How many values of `AVG(points)` are there per group?

How many values of `COUNT(stu_name)` are there per group?

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

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

GROUP BY: Multiple Columns that Work

```
SELECT size_hs, AVG(points), COUNT(stu_name)
FROM Students
GROUP BY size_hs;
```

size_hs	AVG(points)	COUNT(stu_name)
60	110	1
50	107.5	2
35	121	1
45	117	1

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

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

GROUP BY with JOIN

```
SELECT course, AVG(points) FROM Students
INNER JOIN Applications USING(snum)
GROUP BY course;
```

snum	stu_name	points	size_hs	uni_name	course	decision
003	Jack Fines	110	60	Man Met	Computing	Accept
003	Jack Fines	110	60	Man Met	Computer Science	Accept
009	Michelle Jones	114	50	Uni of Manchester	Computer Science	Reject
017	Nazia Hassan	101	35	Man Met	Computing	Reject
017	Nazia Hassan	101	35	Salford Uni	Computing	Accept
022	Shane Jordan	121	35	Man Met	Computing	Accept

$\longrightarrow$

course	AVG(points)
Computing	108.25
Computer Science	112

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

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

Group By Multiple Columns

```
SELECT uni_name, course, AVG(points) FROM Students
INNER JOIN Applications USING(snum)
GROUP BY uni_name, course;
```

snum	stu_name	points	size_hs	uni_name	course	decision
003	Jack Fines	110	60	Man Met	Computing	Accept
003	Jack Fines	110	60	Man Met	Computer Science	Accept
009	Michelle Jones	114	50	Uni of Manchester	Computer Science	Reject
017	Nazia Hassan	101	35	Man Met	Computing	Reject
017	Nazia Hassan	101	35	Salford Uni	Computing	Accept
022	Shane Jordan	121	35	Man Met	Computing	Accept

uni_name	Course	AVG(points)
Man Met	Computer Science	110.0
Man Met	Computing	110.7
Salford Uni	Computing	101.0
Uni of Manchester	Computing	114.0

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

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

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

- Use HAVING if your filter condition includes a group function

Example 1: HAVING

```
SELECT city, MAX(enrolment)
FROM Universities
GROUP BY city
HAVING MAX(enrolment) > 15000;
```

city	MAX(enrolment)
Manchester	26,725
Salford	14,895
Liverpool	17,835

Apply HAVING →

city	MAX(enrolment)
Manchester	26,725
Liverpool	17,835

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

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

Example 2: HAVING

```
SELECT city, SUM(enrolment)
FROM Universities
GROUP BY city
HAVING SUM(enrolment) < 20000;
```

city	SUM(enrolment)
Manchester	52,535
Salford	14,895
Liverpool	17,835

Apply HAVING →

city	SUM(enrolment)
Salford	14,895
Liverpool	17,835

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

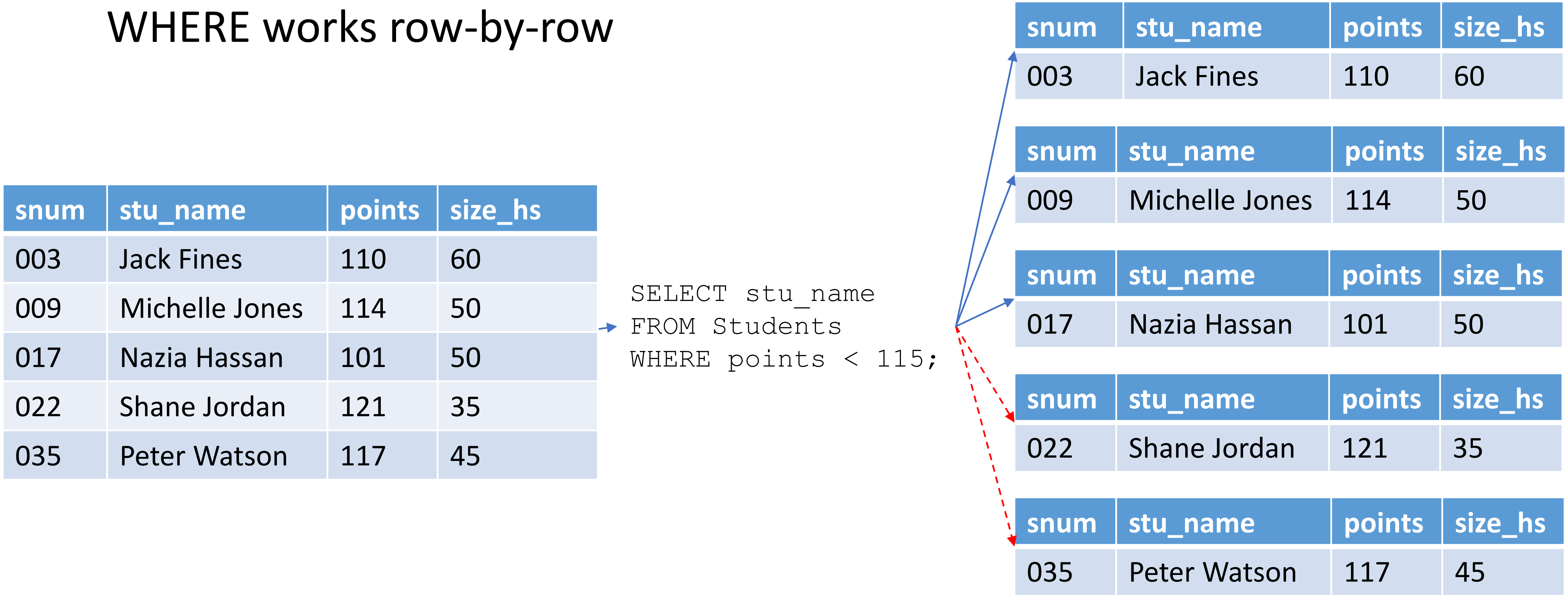
Applications (snum, uni_name, course, decision)

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

Subqueries

How WHERE works

WHERE works row-by-row



snum	stu_name	points	size_hs
003	Jack Fines	110	60
009	Michelle Jones	114	50
017	Nazia Hassan	101	50
022	Shane Jordan	121	35
035	Peter Watson	117	45

```
SELECT stu_name
FROM Students
WHERE points < 115;
```

snum	stu_name	points	size_hs
003	Jack Fines	110	60

snum	stu_name	points	size_hs
009	Michelle Jones	114	50

snum	stu_name	points	size_hs
017	Nazia Hassan	101	50

snum	stu_name	points	size_hs
022	Shane Jordan	121	35

snum	stu_name	points	size_hs
035	Peter Watson	117	45

When WHERE does not work

Can only use information available on that same row

snum	stu_name	points	size_hs
003	Jack Fines	110	60
009	Michelle Jones	114	50
017	Nazia Hassan	101	50
022	Shane Jordan	121	35
035	Peter Watson	117	45

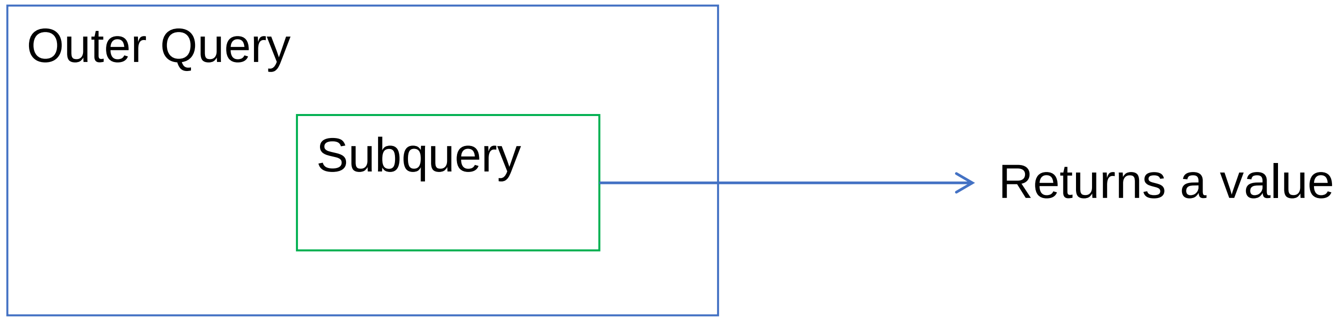
```
SELECT stu_name
FROM Students
WHERE points < AVG(points);
```

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	points	size_hs
003	Jack Fines	110	60
009	Michelle Jones	114	50
017	Nazia Hassan	101	50
022	Shane Jordan	121	35
035	Peter Watson	117	45

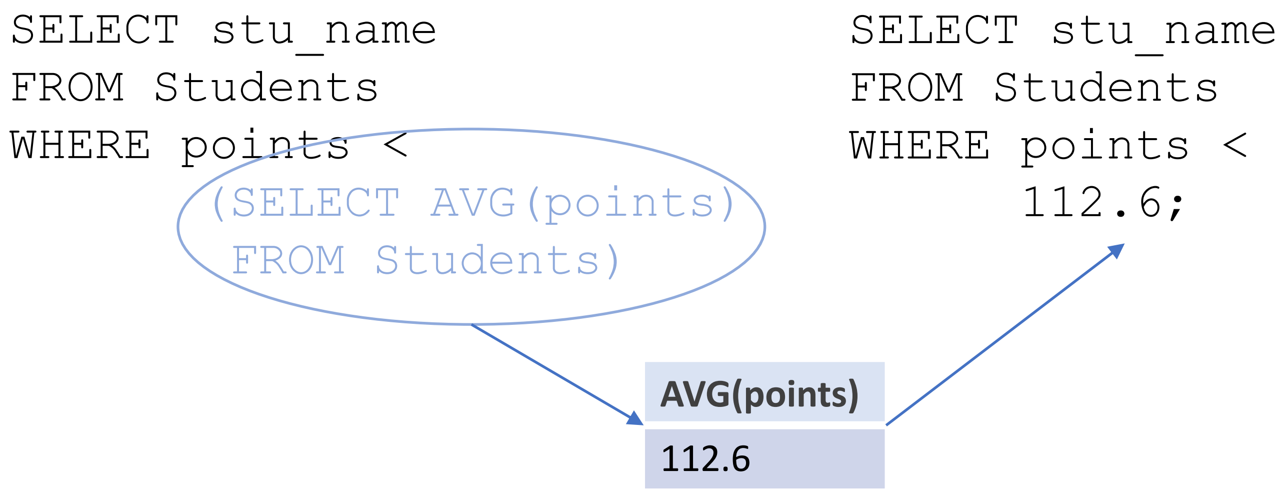
```
SELECT stu_name
FROM Students
WHERE points <
      (SELECT AVG(points)
       FROM Students);
```

stu_name
Jack Fines
Nazia Hassan

Subquery

Imagine replacing the subquery with its result

snum	stu_name	points	size_hs
003	Jack Fines	110	60
009	Michelle Jones	114	50
017	Nazia Hassan	101	50
022	Shane Jordan	121	35
035	Peter Watson	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);
```

Multi-Row Subqueries

A subquery can return many rows

treat result as a list, using IN keyword

snum	stu_name	points	size_hs
003	Jack Fines	110	60
009	Michelle Jones	114	50
017	Nazia Hassan	101	50
022	Shane Jordan	121	35
035	Peter Watson	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

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

Outer Joins

Between Inner and Cross Joins

Cross Join gives every possible combination

Inner Join gives only those combinations which match the condition

Outer Join gives the combinations which match plus some that don’t

- Left Outer Join
- Right Outer Join
- Full Outer Join

The Outer Join

Takes every row from one of the tables

If there is a row in the other table table that matches then it is used

If not, the columns from the other table are included with NULL values

Example: Outer Join

```
SELECT *
FROM Universities
LEFT OUTER JOIN Applications
  ON Universities.uni_name = Applications.uni_name;
```

uni_name	city	enrolment	app_deadline	snum	uni_name1	course	decision
Man Met	Manchester	25,810	15-SEP-18	003	Man Met	Computing	Accept
Man Met	Manchester	25,810	15-SEP-18	003	Man Met	Computer Science	Accept
Man Met	Manchester	25,810	15-SEP-18	017	Man Met	Computing	Reject
Man Met	Manchester	25,810	15-SEP-18	022	Man Met	Computing	Accept
Uni of Manchester	Manchester	26,725	20-SEP-18	009	Uni of Manchester	Computer Science	Reject
Salford Uni	Salford	14,895	18-SEP-18	017	Salford Uni	Computing	Accept
John Moores	Liverpool	17,835	22-SEP-18	(null)	(null)	(null)	(null)

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

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

Outer Join – Don’t need OUTER

```
SELECT *
FROM Universities
LEFT ~~OUTER~~ JOIN Applications
  ON Universities.uni_name = Applications.uni_name;
```

uni_name	city	enrolment	app_deadline	snum	uni_name1	course	decision
Man Met	Manchester	25,810	15-SEP-18	003	Man Met	Computing	Accept
Man Met	Manchester	25,810	15-SEP-18	003	Man Met	Computer Science	Accept
Man Met	Manchester	25,810	15-SEP-18	017	Man Met	Computing	Reject
Man Met	Manchester	25,810	15-SEP-18	022	Man Met	Computing	Accept
Uni of Manchester	Manchester	26,725	20-SEP-18	009	Uni of Manchester	Computer Science	Reject
Salford Uni	Salford	14,895	18-SEP-18	017	Salford Uni	Computing	Accept
John Moores	Liverpool	17,835	22-SEP-18	(null)	(null)	(null)	(null)

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

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

Knowing your Left from your Right

The “left” table is the one to the left/before the JOIN keyword

The “right” table is the one to the right/after the JOIN keyword

```
SELECT *
FROM TableA a
LEFT OUTER JOIN TableB b
  ON a.key = b.key
```

TableA is left, TableB is right

Outer Join – Left vs Right

```
SELECT *
FROM Universities
RIGHT OUTER JOIN Applications
  ON Universities.uni_name = Applications.uni_name;
```

uni_name	city	enrolment	app_deadline	snum	uni_name1	course	decision
Man Met	Manchester	25,810	15-SEP-18	003	Man Met	Computing	Accept
Man Met	Manchester	25,810	15-SEP-18	003	Man Met	Computer Science	Accept
Man Met	Manchester	25,810	15-SEP-18	017	Man Met	Computing	Reject
Man Met	Manchester	25,810	15-SEP-18	022	Man Met	Computing	Accept
Uni of Manchester	Manchester	26,725	20-SEP-18	009	Uni of Manchester	Computer Science	Reject
Salford Uni	Salford	14,895	18-SEP-18	017	Salford Uni	Computing	Accept

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

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

Full Outer Join

Takes every row from both tables

If a row from the other table matches then use it

If not, complete with NULL values

FULL = Left + Right

```
SELECT stu_name, course, uni_name
FROM Students
FULL OUTER JOIN Applications
  USING (snum)
FULL OUTER JOIN Universities
  USING (uni_name);
```

stu_name	course	uni_name
Jack Fines	Computer Science	Man Met
Jack Fines	Computing	Man Met
Michelle Jones	Computer Science	Uni of Manchester
Nazia Hassan	Computing	Man Met
Nazia Hassan	Computing	Salford Uni
Shane Jordan	Computing	Man Met
Peter Watson	(NULL)	(NULL)
(NULL)	(NULL)	John Moores

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	12-SEP-98	110	60
009	Michelle Jones	22-DEC-97	114	50
017	Nazia Hassan	05-APR-98	101	50
022	Shane Jordan	10-OCT-97	121	35
035	Peter Watson	29-JUN-98	117	45

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