Functions, Ordering and Joining

Anthony Kleerekoper

Databases 6G4Z0016

vevox

Audience Engagement

Join at vevox.app

Or search Vevox in the app store

ID: 158-418-551

Example Database

RECAP

Universities

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

Students

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	117	45

Applications

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

Basic SELECT Statement

SELECT keyword indicates we are reading data from a table

```
SELECT [one or more columns]
FROM [a table];
```

Returns the data in every row as a list

Example: Listing multiple columns

RECAP

List all university names and application deadlines with:

```
SELECT uni_name, app_deadline
FROM Universities;
```

uni_name	app_deadline
Man Met	2022-09-15
Uni of Manchester	2022-09-20
Salford Uni	2022-09-18
John Moores	2022-09-22

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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 Rows

RECAP

When specifying columns can use names

Rows don’t have names – they have data

Can only specify rows based on their data

Need to provide a condition to match the data in the rows

Specifying rows is called *filtering*

SELECT and WHERE

RECAP

The SELECT clause specifies which columns you want

The WHERE clause specifies which rows you want

uni_name	city	enrolment	app_deadline
Man Met	Manchester	25,810	2022-[illegible]-15
Uni of Manchester	Manchester	26,725	2022-09-[illegible]
Salford Uni	Salford	14,895	2022-09-1[illegible]
John Moores	Liverpool	17,835	2022-09-22

SELECT

WHERE

```
SELECT enrolment
FROM Universities
WHERE enrolment < 15000;
```

Example: Top Students

RECAP

```
SELECT snum, stu_name, points
FROM Students
WHERE points > 115;
```

snum	stu_name	points
022	Shane Jordan	121
035	Peter Watson	117

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Clauses and Conditions

RECAP

select list

SELECT clause

FROM clause

WHERE clause

```
SELECT enrolment
FROM Universities
WHERE enrolment < 15000;
```

filter condition

Convention:
keywords in UPPERCASE
column names in lowercase
table names in Sentence Case

Join: **vevox.app** ID: **158-418-551** POLL OPEN

How many rows and columns will be returned by this query?

```
SELECT snum, uni_name, decision
FROM Applications
WHERE decision = 'Accept';
```

1. 3 rows and 3 columns
2. 4 rows and 4 columns
3. 3 rows and 4 columns
4. ✓ 4 rows and 3 columns

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

30

Combining Criteria with AND and OR

You can have more than one criteria in the WHERE clause

```
SELECT col1, col2 AS alias
FROM tabl1
WHERE cond1 AND cond2 OR cond3 etc
```

Be careful with ANDs and ORs to group criteria correctly

- use brackets to make your intention clear

Example: Students accepted at Man Met

```
SELECT snum, uni_name, decision
FROM Applications
WHERE uni_name = 'Man Met' AND decision = 'Accept';
```

snum	uni_name	decision
003	Man Met	Accept
003	Man Met	Accept
022	Man Met	Accept

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Brackets are Important

Consider the following query – what is the result?

```
SELECT stu_name, points, size_hs
FROM Students
WHERE points > 115
      OR points > 110
      AND size_hs >= 50;
```

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Brackets are Important

Result depends on the brackets

```
SELECT stu_name, points, size_hs
FROM Students
WHERE (points > 115
       OR points > 110)
       AND size_hs >= 50;
```

stu_name	points	size_hs
Michelle Jones	114	50

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Brackets are Important

Result depends on the brackets

```
SELECT stu_name, points, size_hs
FROM Students
WHERE points > 115
      OR (points > 110
      AND size_hs >= 50);
```

stu_name	points	size_hs
Michelle Jones	114	50
Shane Jordan	121	35
Peter Watson	117	45

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

NULL Values

NULL values are a special placeholder

- they indicate that there is no value there

A NULL is not the same as a 0 or an empty string

A NULL is not a real value therefore cannot be compared

- Any comparison to a NULL returns False
- e.g. `WHERE x != 5` will not show NULL values

Example: NULL Values

Suppose we had some missing information about some students

```
SELECT *
FROM Students
WHERE points = size_hs
      OR points != 121;
```

Would return nothing

NULL does not even equal NULL

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

IS NULL and IS NOT NULL

Special comparison operators for dealing with NULL values

```
SELECT *
FROM Students
WHERE points IS NULL;
```

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	NULL	60
017	Nazia Hassan	2001-05-05	NULL	NULL

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

Renaming Columns

We can rename columns as we select them

new name is called an ***alias***

Use **AS** keyword to rename columns:

name	School Size
003	60
009	50
017	50
022	35
035	45

```
SELECT stu_name AS name, size_hs AS "School Size"
FROM Students;
```

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Notes on Aliases

```
SELECT stu_name AS name, size_hs AS “School Size”
FROM Students;
```

You don’t have to use quotes around a single word

You do if there is a space

You don’t have to use the AS keyword

It’s best practice to always use it

Changing the Data

During a query you can change:

- Column names (aliases)
- The data itself (using functions)

The underlying data is not changed

Only the data as we see is changed

The results of a SELECT statement are a copy of the underlying data

Functions in SQL

You should be familiar with functions in programming

SQL has pre-defined functions used on single rows

- i.e. they effect each row individually

Three different types in this topic

- CHARACTER functions
- NUMBER functions
- DATE functions

Others exist that do not fall in any category

Single-Row Functions

Syntax is like any other function:

```
function_name(parameter)
```

In the case of SQL, the parameters are column names or constants

```
SELECT function_name(col_name)
FROM table_name;
```

Too many functions to discuss all

You can look up others/details if and when you need them

LENGTH function

`LENGTH(stu_name)`

returns the number of characters in the column, row by row

```
SELECT stu_name, LENGTH(stu_name)
FROM Students;
```

Technically returns number of bytes in string

If using a multi-byte character set use

`CHAR_LENGTH` or `CHARACTER_LENGTH`

stu_name	LENGTH(stu_name)
Jack Fines	10
Michelle Jones	14
Nazia Hassan	12
Shane Jordan	12
Peter Watson	12

REPLACE function

```
REPLACE(column, string, replacement)
```

will replace every occurrence of “string” in “column”, with “replacement”

```
SELECT stu_name, REPLACE(stu_name, 'e', '*')
FROM Students;
```

stu_name	REPLACE(stu_name, ‘e’, ‘*’)
Jack Fines	Jack Fin*s
Michelle Jones	Mich*ll* Jon*s
Nazia Hassan	Nazia Hassan
Shane Jordan	Shan* Jordan
Peter Watson	P*t*r Watson

Concatenation

Concatenation is when we stick two or more strings together into one

Can use CONCAT()

```
SELECT CONCAT(stu_name, ' has ', points, ' points’) AS points
FROM Students;
```

points
Jack Fines has 110 points
Michelle Jones has 114 points
Nazia Hassan has 101 points
Shane Jordan has 121 points
Peter Watson has 117 points

Row Functions and Aliases

Column names in the output are exactly as they appear in the select list

Usually this is not very helpful

Very common to use aliases with functions

```
SELECT stu_name, LENGTH(stu_name)
FROM Students;
```

stu_name	LENGTH(stu_name)
Jack Fines	10
Michelle Jones	14
Nazia Hassan	12
Shane Jordan	12
Peter Watson	12

```
SELECT stu_name AS Name,
LENGTH(stu_name) AS Length
FROM Students;
```

Name	Length
Jack Fines	10
Michelle Jones	14
Nazia Hassan	12
Shane Jordan	12
Peter Watson	12

Nested Functions

Can apply a function to the result of another

General syntax:

```
function_2(function_1(col_name))
```

```
SELECT stu_name, LENGTH(stu_name),
    LENGTH(REPLACE(stu_name, 'e', ''))
FROM Students;
```

stu_name	LENGTH(stu_name)	LENGTH(REPLACE(stu_name, 'e', ''))
Jack Fines	10	9
Michelle Jones	14	11
Nazia Hassan	12	12
Shane Jordan	12	11
Peter Watson	12	10

Mathematical Functions

Numerical attributes can be used in equations

```
SELECT enrolment * 1.1 AS "enrolment plus 10%"
FROM Universities;
```

enrolment plus 10%
28391
29397.5
16384.5
19618.5

```
SELECT points + 5 AS "new points"
FROM Students;
```

new points
115
119
106
126
122

Single-Row Date Functions

```
EXTRACT([DAY,MONTH,YEAR] FROM date)
```

Pick out the day or month or year from a date

```
SELECT app_deadline,
       EXTRACT(DAY FROM app_deadline) AS DAY,
       EXTRACT(MONTH FROM app_deadline) AS MONTH,
       EXTRACT(YEAR FROM app_deadline) AS YEAR
       FROM Universities;
```

app_deadline	DAY	MONTH	YEAR
2022-09-22	22	9	2022
2022-09-15	15	9	2022
2022-09-18	18	9	2022
2022-09-20	20	9	2022

There are also individual functions:

```
YEAR(), MONTH(), DAY()
```

Difference between Dates

```
TIMESTAMPDIFF(UNIT, datetime1, datetime2)
```

Find the result of `datetime2 - datetime1`

Unit can be `YEAR, MONTH, DAY, HOUR, MINUTE, SECOND`

```
SELECT app_deadline,
  TIMESTAMPDIFF(DAY,'2022-01-01',app_deadline) AS Day,
  TIMESTAMPDIFF(MONTH,'2022-01-01',app_deadline) AS Month,
  TIMESTAMPDIFF(HOUR,'2022-01-01',app_deadline) AS Hour
FROM Universities;
```

app_deadline	Day	Month	Hour
2022-09-22	264	8	6336
2022-09-15	257	8	6168
2022-09-18	260	8	6240
2022-09-20	262	8	6288

Converting NULL Values

NULL Values can interfere with some functions

Example: `concat()` with a NULL value returns NULL

Can convert NULL values into something else with `IFNULL(col, val)`

If `col` is not null it returns `col` otherwise it returns `val`

```
SELECT snum, points, IFNULL(points, 0)AS P
FROM Students
```

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

snum	points	P
003	NULL	0
009	114	114
017	NULL	0
022	121	121
035	117	117

Lots of Functions

There are loads of built-in row functions

MariaDB’s website lists over 450!!

Too many to discuss all here

Ones we expect you to know are listed on Moodle

Use the MariaDB documentation to learn them

Start at https://mariadb.com/kb/en/built-in-functions/

Join: **vevox.app** ID: **158-418-551** POLL OPEN

Which filter condition or conditions find the applications that are not for Man Met

Vote for up to 2 choices

1. WHERE uni_name NOT = 'Man Met'

0%

2. WHERE uni_name != 'Man Met'

0%

3. WHERE NOT uni_name = 'Man Met'

0%

4. WHERE uni_name = 'Man Met'

0%

(% = Percentage of Voters)

30

Ordering the Results

Choosing the order of the rows

Database Tables Have No Order

Never assume your results will come in any given order

Just because it was in one order once does not mean it will always be in that order

If you want to ensure an order you must tell the database system

use the **ORDER BY** clause

Always the last clause in any SELECT query

Order can be ascending using the **ASC** keyword

or descending using the **DESC** keyword

Example: Listing universities by size

```
SELECT uni_name, city, enrolment
FROM Universities
ORDER BY enrolment DESC;
```

uni_name	city	enrolment
Uni of Manchester	Manchester	26,725
Man Met	Manchester	25,810
John Moores	Liverpool	17,835
Salford Uni	Salford	14,895

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Can sort by multiple columns

```
SELECT uni_name, city, enrolment
FROM Universities
ORDER BY city ASC, enrolment DESC;
```

uni_name	city	enrolment
John Moores	Liverpool	17,835
Uni of Manchester	Manchester	26,725
Man Met	Manchester	25,810
Salford Uni	Salford	14,895

Will first sort alphabetically by city
then by enrolment from largest to smallest

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Can Sort Using Row Functions

Can use a row function in the ORDER BY clause

```
SELECT snum, stu_name, points, size_hs
FROM Students
ORDER BY length(stu_name) ASC;
```

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

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Joining Tables

A **join** is an operation that combines data from two tables

there are different types of joins, we will discuss CROSS and INNER

Joins create temporary tables with the columns from the original tables

e.g. join Students and Applications:

Question is how to fill in the row values

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

The Cross Join

Simplest of all joins

Fill in the rows with every possible combination of date

If we have *x* rows in the first table and *y* in the second

we will have *x***y* rows in the output

Example: Cross Join

```
SELECT uni_name,
       city,
       stu_name,
       points
FROM Universities
CROSS JOIN Students;
```

uni_name	city	stu_name	points
John Moores	Liverpool	Jack Fines	110
Man Met	Manchester	Jack Fines	110
Salford Uni	Salford	Jack Fines	110
Uni of Manchester	Manchester	Jack Fines	110
John Moores	Liverpool	Michelle Jones	114
Man Met	Manchester	Michelle Jones	114
Salford Uni	Salford	Michelle Jones	114
Uni of Manchester	Manchester	Michelle Jones	114
John Moores	Liverpool	Jack Fines	110
Man Met	Manchester	Jack Fines	110
Lots more Rows			
John Moores	Liverpool	Peter Watson	117
Man Met	Manchester	Peter Watson	117
Salford Uni	Salford	Peter Watson	117
Uni of Manchester	Manchester	Peter Watson	117
John Moores	Liverpool	Peter Watson	117

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Cross Join Syntax

Can specify it explicitly:

```
SELECT uni_name, city, stu_name, points
FROM Universities
CROSS JOIN Students;
```

Or implicitly:

```
SELECT uni_name, city, stu_name, points
FROM Universities, Students;
```

Always better to be explicit

The Inner Join

Cross join is blind – every possible combination

Usually you don’t want every possible combination

Usually you want pairs of rows that “match” in some way

Inner Join finds pairs of rows that match based on a ***join condition***

Example: Inner Join

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

uni_name	city	enrolment	app_deadline	snum	uni_name	course	decision
Man Met	Manchester	25810	2022-09-15	3	Man Met	Computer Science	Accept
Man Met	Manchester	25810	2022-09-15	3	Man Met	Computing	Accept
Uni of Manchester	Manchester	26725	2022-09-20	9	Uni of Manchester	Computing	Reject
Man Met	Manchester	25810	2022-09-15	17	Man Met	Computing	Reject
Salford Uni	Salford	14895	2022-09-18	17	Salford Uni	Computing	Accept
Man Met	Manchester	25810	2022-09-15	22	Man Met	Computing	Accept

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Join Conditions

Join conditions can be any valid comparison

Most common is **=**

When the join condition is equality the join is also called an **equijoin**

Can also be **>**, **<**, **!=**, **BETWEEN**, **LIKE, IN**

But very, very rare

The database system will compare every pair of rows to see if it meets the join condition or not

Inner Join – Don't need "INNER"

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

uni_name	city	enrolment	app_deadline	snum	uni_name	course	decision
Man Met	Manchester	25810	2022-09-15	3	Man Met	Computer Science	Accept
Man Met	Manchester	25810	2022-09-15	3	Man Met	Computing	Accept
Uni of Manchester	Manchester	26725	2022-09-20	9	Uni of Manchester	Computing	Reject
Man Met	Manchester	25810	2022-09-15	17	Man Met	Computing	Reject
Salford Uni	Salford	14895	2022-09-18	17	Salford Uni	Computing	Accept
Man Met	Manchester	25810	2022-09-15	22	Man Met	Computing	Accept

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Inner Join – Implicit Syntax

```
SELECT *
FROM Universities, Applications
WHERE Universities.uni_name = Applications.uni_name;
```

uni_name	city	enrolment	app_deadline	snum	uni_name	course	decision
Man Met	Manchester	25810	2022-09-15	3	Man Met	Computer Science	Accept
Man Met	Manchester	25810	2022-09-15	3	Man Met	Computing	Accept
Uni of Manchester	Manchester	26725	2022-09-20	9	Uni of Manchester	Computing	Reject
Man Met	Manchester	25810	2022-09-15	17	Man Met	Computing	Reject
Salford Uni	Salford	14895	2022-09-18	17	Salford Uni	Computing	Accept
Man Met	Manchester	25810	2022-09-15	22	Man Met	Computing	Accept

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Stringing Joins Together

```
SELECT uni_name, snum, stu_name, course
FROM Universities
INNER JOIN Applications
 ON Universities.uni_name = Applications.uni_name
INNER JOIN Students
 ON Applications.snum = Students.snum;
```

uni_name	snum	stu_name	course
Man Met	3	Jack Fines	Computer Science
Man Met	3	Jack Fines	Computing
Man Met	17	Nazia Hassan	Computing
Man Met	22	Shane Jordan	Computing
Salford Uni	17	Nazia Hassan	Computing
Uni of Manchester	9	Michelle Jones	Computing

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Join: **vevox.app** ID: **158-418-551** POLL OPEN

True or False: A Cross Join will always result in at least as many rows as an Inner Join?

1. True

2. False

Common Column Names: A problem

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

Dot notation to remove ambiguity

The following will throw an error

```
SELECT *
FROM Universities
INNER JOIN Applications
  ON uni_name = uni_name;
```

Inner Join – Repeated Columns

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

uni_name	city	enrolment	app_deadline	snum	uni_name	course	decision
Man Met	Manchester	25810	2022-09-15	3	Man Met	Computer Science	Accept
Man Met	Manchester	25810	2022-09-15	3	Man Met	Computing	Accept
Uni of Manchester	Manchester	26725	2022-09-20	9	Uni of Manchester	Computing	Reject
Man Met	Manchester	25810	2022-09-15	17	Man Met	Computing	Reject
Salford Uni	Salford	14895	2022-09-18	17	Salford Uni	Computing	Accept
Man Met	Manchester	25810	2022-09-15	22	Man Met	Computing	Accept

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Straightforward Solution

Explicitly list the columns you want!

I've been using "`SELECT *`" so far for convenience and space

```
SELECT applications.uni_name, city, enrolment, snum,
       course, decision
FROM Universities
INNER JOIN Applications
  ON universities.uni_name = applications.uni_name;
```

The USING keyword

A shorthand for an equijoin and merges the duplicated column

```
SELECT *
FROM Universities
INNER JOIN Applications
  USING(uni_name);
```

Merges the two columns so uni_name only appears once in the output

Only works if both columns have exactly the same name

USING – Removes Duplicate Columns

```
SELECT *
FROM Universities
INNER JOIN Applications USING(uni_name);
```

uni_name	city	enrolment	app_deadline	snum	course	decision
Man Met	Manchester	25810	2022-09-15	3	Computer Science	Accept
Man Met	Manchester	25810	2022-09-15	3	Computing	Accept
Uni of Manchester	Manchester	26725	2022-09-20	9	Computing	Reject
Man Met	Manchester	25810	2022-09-15	17	Computing	Reject
Salford Uni	Salford	14895	2022-09-18	17	Computing	Accept
Man Met	Manchester	25810	2022-09-15	22	Computing	Accept

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

The Natural Join

Another shorthand for an equijoin is the Natural Join

```
SELECT *
FROM Universities
NATURAL JOIN Applications;
```

Will join on the common column (i.e. uni_name)

and will show common column only once

Natural Join – Removes Duplicate Columns

```
SELECT *
FROM Universities
NATURAL JOIN Applications;
```

uni_name	city	enrolment	app_deadline	snum	course	decision
Man Met	Manchester	25810	2022-09-15	3	Computer Science	Accept
Man Met	Manchester	25810	2022-09-15	3	Computing	Accept
Uni of Manchester	Manchester	26725	2022-09-20	9	Computing	Reject
Man Met	Manchester	25810	2022-09-15	17	Computing	Reject
Salford Uni	Salford	14895	2022-09-18	17	Computing	Accept
Man Met	Manchester	25810	2022-09-15	22	Computing	Accept

Universities (uni_name, city, enrolment, app_deadline)

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

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

snum	stu_name	dob	points	size_hs
003	Jack Fines	2001-09-12	110	60
009	Michelle Jones	2000-12-22	114	50
017	Nazia Hassan	2001-05-05	101	50
022	Shane Jordan	2002-10-10	121	35
035	Peter Watson	2001-06-29	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

Danger: The Natural Join

Joins on all columns with same name and datatypes

Consider what happens if both tables also have a “Comments” column

Best practice is to never use the Natural Join in production code

(it’s fine for ad hoc, personal queries)

Not all database systems support it (though MariaDB and Oracle do)

Join: **vevox.app** ID: **158-418-551**

Enter Text and Press Send

Please give your feedback on the lecture

Data Captured