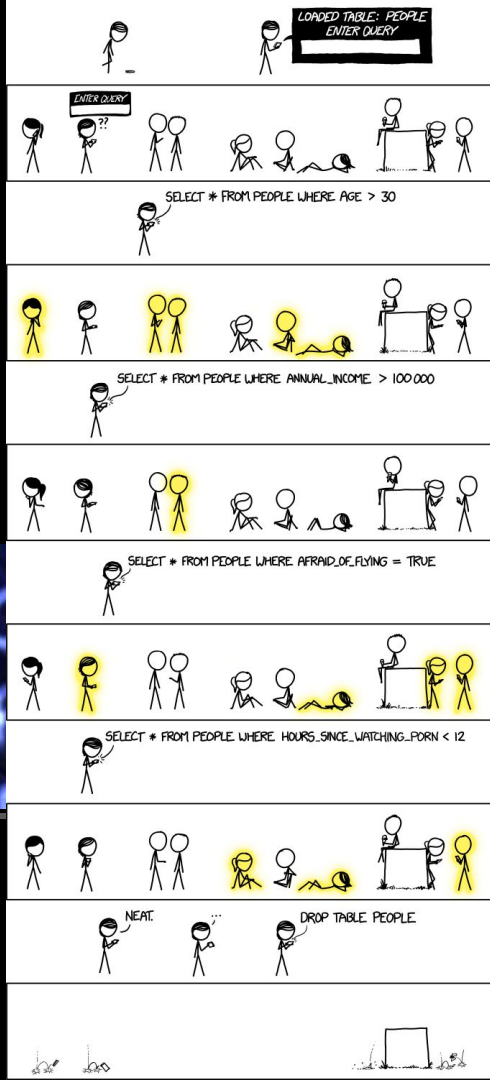
Session 7

Relational Databases (SQL II)



N/LAB: *Data at Scale*

https://xkcd.com/1409/

Remember back 9,900 minutes ago...

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

- **`CREATE TABLE`** - creates a table
- **`UPDATE`** - change row(s) in a table
- **`INSERT`** - add a row to a table
- **`DELETE - remove row(s) from a table`**
- **`SELECT`** - return the specified column(s) from a table
- **`WHERE`** - filters row(s) by condition

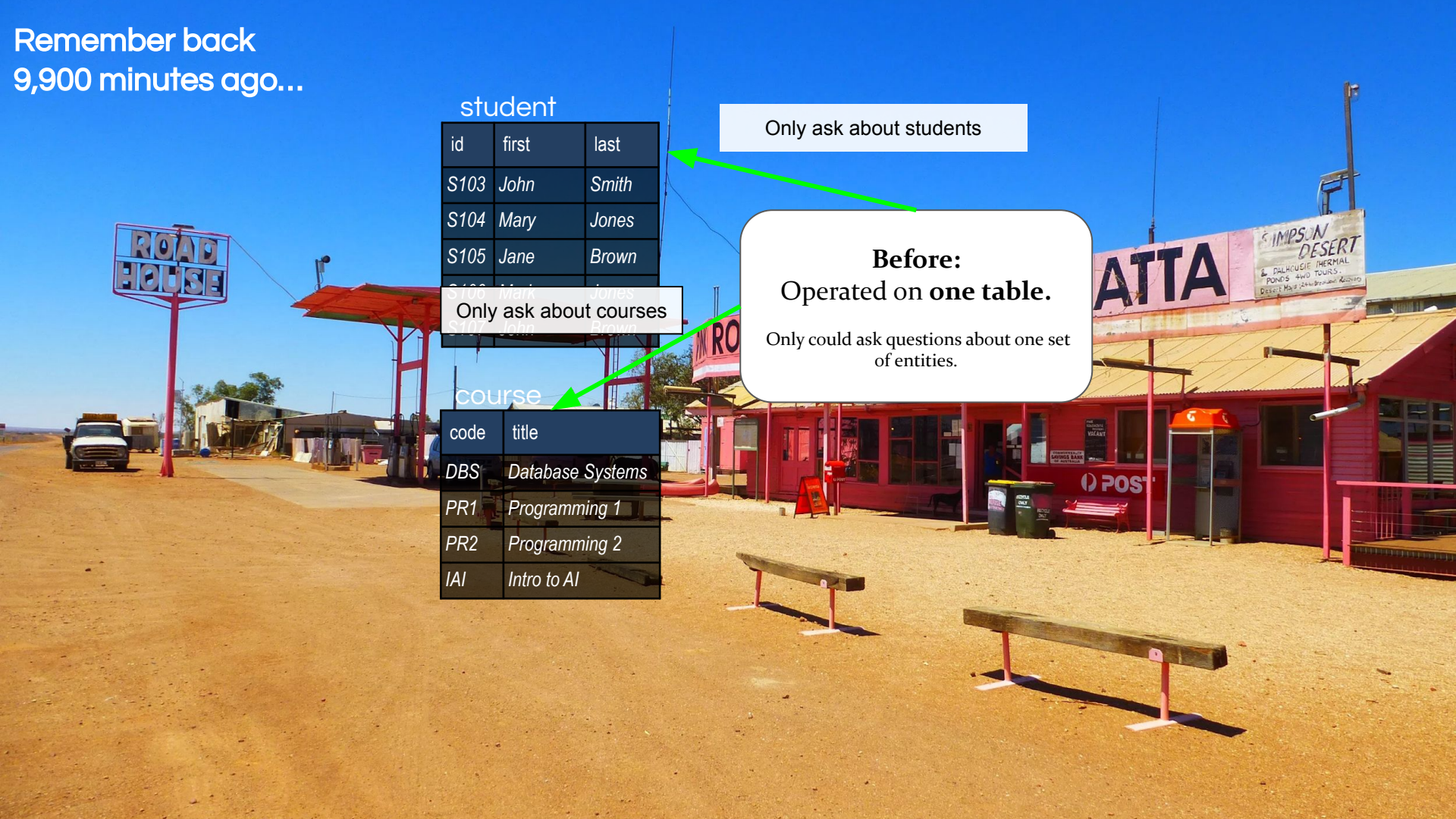
All operate on a single table.

All return a single table.

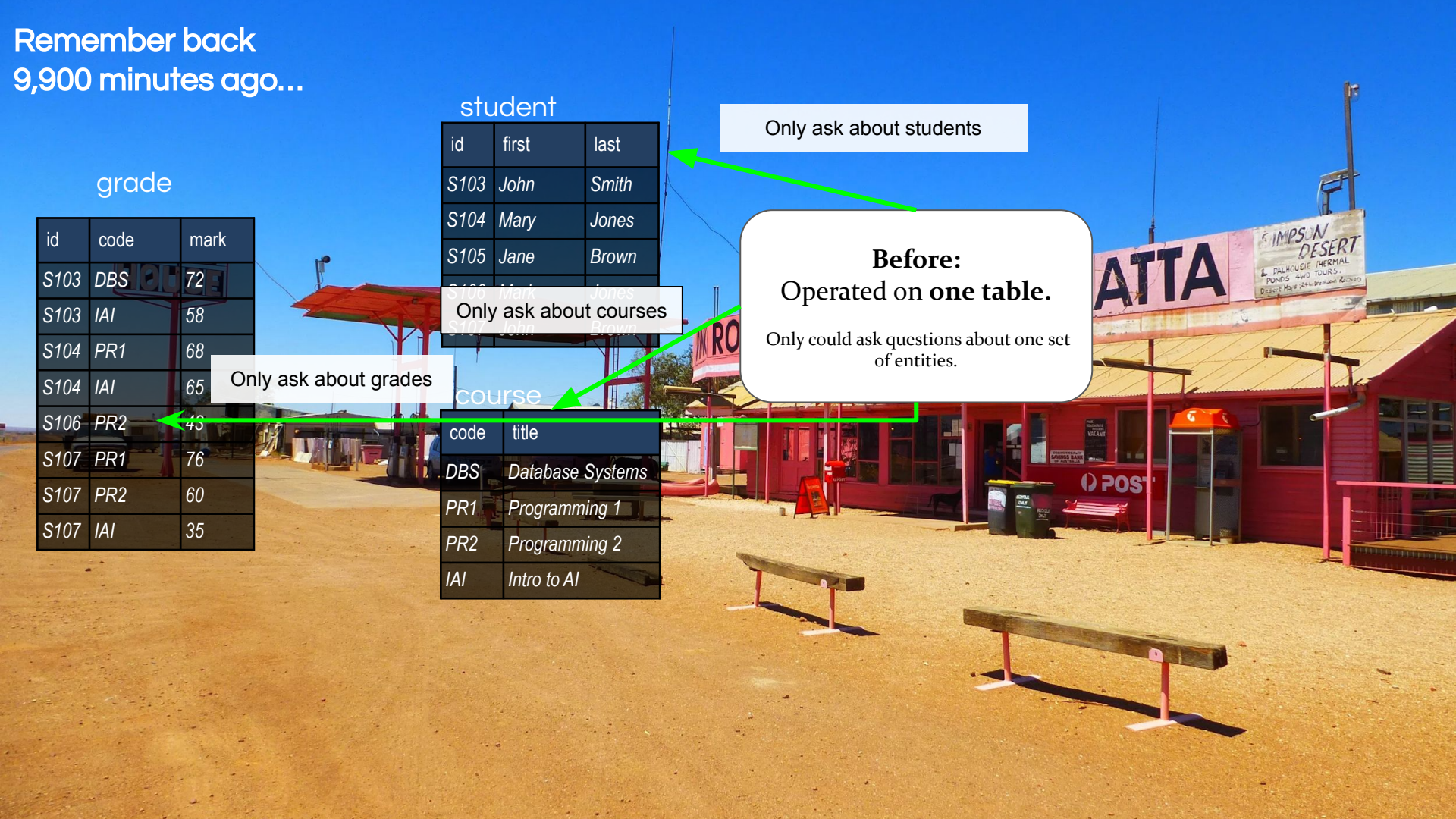
Think in tables, entities & attributes...


Remember back
9,900 minutes ago...
ROAD HOUSE
ATTA
SIMPSON DESERT
& DALHOUSIE THERMAL PONDS 4WD TOURS.
POST
Before:
Operated on one table.
Only could ask questions about one set of entities.


Remember back
9,900 minutes ago...
student
id | first | last
S103 | John | Smith
S104 | Mary | Jones
S105 | Jane | Brown
S106 | Mark | Jones
S107 | John | Brown
Only ask about students
Before:
Operated on one table.
Only could ask questions about one set of entities.
ROAD HOUSE
SIMPSON DESERT
DALHOUSIE THERMAL PONDS 4WD TOURS.
POST


Remember back
9,900 minutes ago...
student
id | first | last
S103 | John | Smith
S104 | Mary | Jones
S105 | Jane | Brown
Only ask about students
Only ask about courses
course
code | title
DBS | Database Systems
PR1 | Programming 1
PR2 | Programming 2
IAI | Intro to AI
Before:
Operated on one table.
Only could ask questions about one set of entities.
ROAD HOUSE
SIMPSON DESERT
POST

Remember back 9,900 minutes ago...



student

id	first	last
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

course

code	title
DBS	Database Systems
PR1	Programming 1
PR2	Programming 2
IAI	Intro to AI

Only ask about students

Only ask about courses

Only ask about grades

Before:
Operated on **one table.**

Only could ask questions about one set of entities.

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

student

id	first	last
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown
S106	Mark	Jones
S107	John	Brown

course

code	title
DBS	Database Systems
PR1	Programming 1
PR2	Programming 2
IAI	Intro to AI

Now: Joining tables together.

Run operators (ask questions) on new set of composite entities.

SQL: JOINING TABLES
(the traditional way)

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

student

id	first	last
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown
S106	Mark	Jones
S107	John	Brown

course

code	title
DBS	Database Systems
PR1	Programming 1
PR2	Programming 2
IAI	Intro to AI

Facts about two different entities types can be joined together to obtain facts about pairs of entity occurrences (one from each type).

We're going to learn the mechanism first, then interpretation.

Joining student occurrence: S103, John, Smith
With grade occurrence: S103, DBS, 72
Creates a new *"Graded student"* entity. (more on interpretation later)

Joining grade occurrence: S103, DBS, 72
With course occurrence: DBS, Database Systems

SQL: JOINING TABLES
(the traditional way)

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

student

id	first	last
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown
S106	Mark	Jones
S107	John	Brown

course

code	title
DBS	Database Systems
PR1	Programming 1
PR2	Programming 2
IAI	Intro to AI

Facts about two different entities types can be joined together to obtain facts about pairs of entity occurrences (one from each type).

We're going to learn the mechanism first, then interpretation.

Joining student occurrence: S103, John, Smith
With grade occurrence: S103, DBS, 72
Creates a new *"Graded student"* entity. (more on interpretation later)

Joining grade occurrence: S103, DBS, 72
With course occurrence: DBS, Database Systems
One-to-many relationship:
student 1...1 → has → 0..* grades
→ Grade table is appended / extended with its unique match in student
→ Creates a new, more detailed, grade entity.

SQL: JOINING TABLES

(the traditional way)

student

id	first	last
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown
S106	Mark	Jones
S107	John	Brown

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
			S103	IAI	58
			S104	PR1	68
			S104	IAI	65
			S106	PR2	43
			S107	PR1	76
			S107	PR2	60
			S107	IAI	35
S104	Mary	Jones	S103	DBS	72
			S103	IAI	58
			S104	PR1	68
			S104	IAI	65
			S106	PR2	43
			S107	PR1	76
			S107	PR2	60
			S107	IAI	35
S105	Jane	Brown	S103	DBS	72
			S103	IAI	58
			S104	PR1	68
			S104	IAI	65
...					

Joining tables joins **every row** in the first table **to every row** in the second.

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

```
SELECT *
FROM student, grade
```

SQL: JOINING TABLES

(the traditional way)

student

id	first	last
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown
S106	Mark	Jones
S107	John	Brown

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
			S103	IAI	58
			S104	PR1	68
			S104	IAI	65
			S106	PR2	43
			S107	PR1	76
			S107	PR2	60
			S107	IAI	35
S104	Mary	Jones	S103	DBS	72
			S103	IAI	58
			S104	PR1	68
			S104	IAI	65
			S106	PR2	43
			S107	PR1	76
			S107	PR2	60
			S107	IAI	35
S105	Jane	Brown	S103	DBS	72
			S103	IAI	58
			S104	PR1	68
			S104	IAI	65
			S106	PR2	43
			S107	PR1	76

Joining tables joins **every row** in the first table **to every row** in the second.

Database does not understand what resultant rows are **true and should be kept.**

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

```
SELECT *
FROM student, grade
```

Database does not understand what resultant rows are **true and should be kept.**

YOU must tell it.

student

id	first	last
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown
S106	Mark	Jones
S107	John	Brown

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S103	John	Smith	S104	PR1	68
S103	John	Smith	S104	IAI	65
S103	John	Smith	S106	PR2	43
S103	John	Smith	S107	PR1	76
S103	John	Smith	S107	PR2	60
S103	John	Smith	S107	IAI	35
S104	Mary	Jones	S103	DBS	72
S104	Mary	Jones	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65
S104	Mary	Jones	S106	PR2	43
S104	Mary	Jones	S107	PR1	76
S104	Mary	Jones	S107	PR2	60
S104	Mary	Jones	S107	IAI	35
S105	Jane	Brown	S103	DBS	72
S105	Jane	Brown	S103	IAI	58
S105	Jane	Brown	S104	PR1	68
S105	Jane	Brown	S104	IAI	65
S105	Jane	Brown	S106	PR2	43
S105	Jane	Brown	S107	PR1	76

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

Database does not understand what resultant rows are **true and should be kept.**

YOU must tell it.

student

id	first	last
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown
S106	Mark	Jones
S107	John	Brown

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S103	John	Smith	S104	PR1	68
S103	John	Smith	S104	IAI	65
S103	John	Smith	S106	PR2	43
S103	John	Smith	S107	PR1	76
S103	John	Smith	S107	PR2	60
S103	John	Smith	S107	IAI	35
S104	Mary	Jones	S103	DBS	72
S104	Mary	Jones	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65
S104	Mary	Jones	S106	PR2	43
S104	Mary	Jones	S107	PR1	76
S104	Mary	Jones	S107	PR2	60
S104	Mary	Jones	S107	IAI	35
S105	Jane	Brown	S103	DBS	72
S105	Jane	Brown	S103	IAI	58
S105	Jane	Brown	S104	PR1	68
S105	Jane	Brown	S104	IAI	65
S105	Jane	Brown	S106	PR2	43
S105	Jane	Brown	S107	PR1	76

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

Database does not understand what resultant rows are **true and should be kept.**

YOU must tell it.

student

id	first	last
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown
S106	Mark	Jones
S107	John	Brown

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S103	John	Smith	S104	PR1	68
S103	John	Smith	S104	IAI	65
S103	John	Smith	S106	PR2	43
S103	John	Smith	S107	PR1	76
S103	John	Smith	S107	PR2	60
S103	John	Smith	S107	IAI	35
S104	Mary	Jones	S103	DBS	72
S104	Mary	Jones	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65
S104	Mary	Jones	S106	PR2	43
S104	Mary	Jones	S107	PR1	76
S104	Mary	Jones	S107	PR2	60
S104	Mary	Jones	S107	IAI	35
S105	Jane	Brown	S103	DBS	72
S105	Jane	Brown	S103	IAI	58
S105	Jane	Brown	S104	PR1	68
S105	Jane	Brown	S104	IAI	65
S105	Jane	Brown	S106	PR2	43
S105	Jane	Brown	S107	PR1	76

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65
S106	Mark	Jones	S106	PR2	43
S107	John	Brown	S107	PR1	76
S107	John	Brown	S107	PR2	60
S107	John	Brown	S107	IAI	35

After you have a single table!
And you know how to deal with those...

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

Graded Students

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65
S106	Mark	Jones	S106	PR2	43
S107	John	Brown	S107	PR1	76
S107	John	Brown	S107	PR2	60
S107	John	Brown	S107	IAI	35

After you have a single table!
And you know how to deal with those...

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

```
SELECT first, last, code, mark
FROM student, grade
WHERE (student.id = grade.id)
```

Graded Students

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65
S106	Mark	Jones	S106	PR2	43
S107	John	Brown	S107	PR1	76
S107	John	Brown	S107	PR2	60
S107	John	Brown	S107	IAI	35

first	last	code	mark
John	Smith	DBS	72
John	Smith	IAI	58
Mary	Jones	PR1	68
Mary	Jones	IAI	65
Mark	Jones	PR2	43
John	Brown	PR1	76
John	Brown	PR2	60
John	Brown	IAI	35

After you have a single table!
And you know how to deal with those...

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

```
SELECT first, last, code, mark
FROM student, grade
WHERE (student.id = grade.id)
```

```
SELECT first, last
FROM student, grade
WHERE (student.id = grade.id)
AND code = 'PR1'
AND mark > 70
```

Graded Students

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65
S106	Mark	Jones	S106	PR2	43
S107	John	Brown	S107	PR1	76
S107	John	Brown	S107	PR2	60
S107	John	Brown	S107	IAI	35

first	last	code	mark
John	Smith	DBS	72
John	Smith	IAI	58
Mary	Jones	PR1	68
Mary	Jones	IAI	65
Mark	Jones	PR2	43
John	Brown	PR1	76
John	Brown	PR2	60
John	Brown	IAI	35

first	last
John	Brown

After we exhaustively join first two tables, let's exhaustively joint the third...

```
SELECT *
FROM student, grade, course
```

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S103	John	Smith	S104	PR1	68
S103	John	Smith	S104	IAI	65
S103	John	Smith	S106	PR2	43
S103	John	Smith	S107	PR1	76
S103	John	Smith	S107	PR2	60
S103	John	Smith	S107	IAI	35
S104	Mary	Jones	S103	DBS	72
S104	Mary	Jones	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65
S104	Mary	Jones	S106	PR2	43
S104	Mary	Jones	S107	PR1	76
S104	Mary	Jones	S107	PR2	60
S104	Mary	Jones	S107	IAI	35

After we exhaustively join first two tables exhaustively joint the third...

```
SELECT *
FROM student, grade, course
```

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S103	John	Smith	S104	PR1	68
S103	John	Smith	S104	IAI	65
S103	John	Smith	S106	PR2	43
S103	John	Smith	S107	PR1	76
S103	John	Smith	S107	PR2	60
S103	John	Smith	S107	IAI	35
S104	Mary	Jones	S103	DBS	72
S104	Mary	Jones	S103	IAI	58
S104	Mary	Jones	S104	PR1	68
S104	Mary	Jones	S104	IAI	65
S104	Mary	Jones	S106	PR2	43
S104	Mary	Jones	S107	PR1	76
S104	Mary	Jones	S107	PR2	60
S104	Mary	Jones	S107	IAI	35

id	first	last	id	code	mark	code	title
S103	John	Smith	S103	DBS	72	DBS	Database Systems
						PR1	Programming 1
						PR2	Programming 2
						IAI	Intro to AI
S103	John	Smith	S103	IAI	58	DBS	Database Systems
						PR1	Programming 1
						PR2	Programming 2
						IAI	Intro to AI
S103	John	Smith	S104	PR1	68	DBS	Database Systems
						PR1	Programming 1
						PR2	Programming 2
						IAI	Intro to AI
S103	John	Smith	S104	IAI	65	DBS	Database Systems
						PR1	Programming 1
						PR2	Programming 2
						IAI	Intro to AI
S103	John	Smith	S106	PR2	43	DBS	Database Systems
						PR1	Programming 1
						PR2	Programming 2
						IAI	Intro to AI
S103	John	Smith	S107	PR1	76	DBS	Database Systems

course

code	title
DBS	Database Systems
PR1	Programming 1
PR2	Programming 2
IAI	Intro to AI

SQL: JOINING TABLES
(the traditional way)

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

student

id	first	last
S103	John	Smith
S104	Mary	Jones
S105	Jane	Brown
S106	Mark	Jones
S107	John	Brown

course

code	title
DBS	Database Systems
PR1	Programming 1
PR2	Programming 2
IAI	Intro to AI

What about three tables?

Same deal, facts "new" entity.

Joining student: S103, John, Smith
With grade: S103, DBS, 72
With course: DBS, Database Systems
Creates a new, "**detailed graded student**" entity.

id	first	last	id	code	mark	code	title
S103	John	Smith	S103	DBS	72	DBS	Database Systems
S103	John	Smith	S103	DBS	72	PR1	Programming 1
S103	John	Smith	S103	DBS	72	PR2	Programming 2
S103	John	Smith	S103	DBS	72	IAI	Intro to AI
S103	John	Smith	S103	IAI	58	DBS	Database Systems
S103	John	Smith	S103	IAI	58	PR1	Programming 1
S103	John	Smith	S103	IAI	58	PR2	Programming 2
S103	John	Smith	S103	IAI	58	IAI	Intro to AI
S103	John	Smith	S104	PR1	68	DBS	Database Systems
S103	John	Smith	S104	PR1	68	PR1	Programming 1
S103	John	Smith	S104	PR1	68	PR2	Programming 2
S103	John	Smith	S104	PR1	68	IAI	Intro to AI
S103	John	Smith	S104	IAI	65	DBS	Database Systems
S103	John	Smith	S104	IAI	65	PR1	Programming 1
S103	John	Smith	S104	IAI	65	PR2	Programming 2
S103	John	Smith	S104	IAI	65	IAI	Intro to AI
S103	John	Smith	S106	PR2	43	DBS	Database Systems
S103	John	Smith	S106	PR2	43	PR1	Programming 1
S103	John	Smith	S106	PR2	43	PR2	Programming 2
S103	John	Smith	S106	PR2	43	IAI	Intro to AI
S103	John	Smith	S107	PR1	76	DBS	Database Systems

Again, we must filter to retain only rows that are **true**

```
SELECT *
FROM student, grade, course
WHERE student.id = grade.id
AND grade.code = course.code
```

id	first	last	id	code	mark	code	title
S103	John	Smith	S103	DBS	72	DBS	Database Systems
S103	John	Smith	S103	DBS	72	PR1	Programming 1
S103	John	Smith	S103	DBS	72	PR2	Programming 2
S103	John	Smith	S103	DBS	72	IAI	Intro to AI
S103	John	Smith	S103	IAI	58	DBS	Database Systems
S103	John	Smith	S103	IAI	58	PR1	Programming 1
S103	John	Smith	S103	IAI	58	PR2	Programming 2
S103	John	Smith	S103	IAI	58	IAI	Intro to AI
S103	John	Smith	S104	PR1	68	DBS	Database Systems
S103	John	Smith	S104	PR1	68	PR1	Programming 1
S103	John	Smith	S104	PR1	68	PR2	Programming 2
S103	John	Smith	S104	PR1	68	IAI	Intro to AI
S103	John	Smith	S104	IAI	65	DBS	Database Systems
S103	John	Smith	S104	IAI	65	PR1	Programming 1
S103	John	Smith	S104	IAI	65	PR2	Programming 2
S103	John	Smith	S104	IAI	65	IAI	Intro to AI
S103	John	Smith	S106	PR2	43	DBS	Database Systems
S103	John	Smith	S106	PR2	43	PR1	Programming 1
S103	John	Smith	S106	PR2	43	PR2	Programming 2
S103	John	Smith	S106	PR2	43	IAI	Intro to AI
S103	John	Smith	S107	PR1	76	DBS	Database Systems

Again, we must filter to retain only rows that are **true**

```
SELECT *
FROM student, grade, course
WHERE student.id = grade.id
AND grade.code = course.code
```

id	first	last	id	code	mark	code	title
S103	John	Smith	S103	DBS	72	DBS	Database Systems
S103	John	Smith	S103	DBS	72	PR1	Programming 1
S103	John	Smith	S103	DBS	72	PR2	Programming 2
S103	John	Smith	S103	DBS	72	IAI	Intro to AI
S103	John	Smith	S103	IAI	58	DBS	Database Systems
S103	John	Smith	S103	IAI	58	PR1	Programming 1
S103	John	Smith	S103	IAI	58	PR2	Programming 2
S103	John	Smith	S103	IAI	58	IAI	Intro to AI
S103	John	Smith	S104	PR1	68	DBS	Database Systems
S103	John	Smith	S104	PR1	68	PR1	Programming 1
S103	John	Smith	S104	PR1	68	PR2	Programming 2
S103	John	Smith	S104	PR1	68	IAI	Intro to AI
S103	John	Smith	S104	IAI	65	DBS	Database Systems
S103	John	Smith	S104	IAI	65	PR1	Programming 1
S103	John	Smith	S104	IAI	65	PR2	Programming 2
S103	John	Smith	S104	IAI	65	IAI	Intro to AI
S103	John	Smith	S106	PR2	43	DBS	Database Systems
S103	John	Smith	S106	PR2	43	PR1	Programming 1
S103	John	Smith	S106	PR2	43	PR2	Programming 2
S103	John	Smith	S106	PR2	43	IAI	Intro to AI
S103	John	Smith	S107	PR1	76	DBS	Database Systems

Again, we must filter to retain only rows that are **true**

```
SELECT *
FROM student, grade, course
WHERE student.id = grade.id
AND grade.code = course.code
```

id	first	last	id	code	mark	code	title
S103	John	Smith	S103	DBS	72	DBS	Database Systems
S103	John	Smith	S103	DBS	72	PR1	Programming 1
S103	John	Smith	S103	DBS	72	PR2	Programming 2
S103	John	Smith	S103	DBS	72	IAI	Intro to AI
S103	John	Smith	S103	IAI	58	DBS	Database Systems
S103	John	Smith	S103	IAI	58	PR1	Programming 1
S103	John	Smith	S103	IAI	58	PR2	Programming 2
S103	John	Smith	S103	IAI	58	IAI	Intro to AI
S103	John	Smith	S104	PR1	68	DBS	Database Systems
S103	John	Smith	S104	PR1	68	PR1	Programming 1
S103	John	Smith	S104	PR1	68	PR2	Programming 2
S103	John	Smith	S104	PR1	68	IAI	Intro to AI
S103	John	Smith	S104	IAI	65	DBS	Database Systems
S103	John	Smith	S104	IAI	65	PR1	Programming 1
S103	John	Smith	S104	IAI	65	PR2	Programming 2
S103	John	Smith	S104	IAI	65	IAI	Intro to AI
S103	John	Smith	S106	PR2	43	DBS	Database Systems
S103	John	Smith	S106	PR2	43	PR1	Programming 1
S103	John	Smith	S106	PR2	43	PR2	Programming 2
S103	John	Smith	S106	PR2	43	IAI	Intro to AI
S103	John	Smith	S107	PR1	76	DBS	Database Systems

Again, we must filter to retain only rows that are **true**

```
SELECT *
FROM student, grade, course
WHERE student.id = grade.id
AND grade.code = course.code
```

id	first	last	id	code	mark	code	title
S103	John	Smith	S103	DBS	72	DBS	Database Systems
S103	John	Smith	S103	DBS	72	PR1	Programming 1
S103	John	Smith	S103	DBS	72	PR2	Programming 2
S103	John	Smith	S103	DBS	72	IAI	Intro to AI
S103	John	Smith	S103	IAI	58	DBS	Database Systems
S103	John	Smith	S103	IAI	58	PR1	Programming 1
S103	John	Smith	S103	IAI	58	PR2	Programming 2
S103	John	Smith	S103	IAI	58	IAI	Intro to AI
S103	John	Smith	S104	PR1	68	DBS	Database Systems
S103	John	Smith	S104	PR1	68	PR1	Programming 1
S103	John	Smith	S104	PR1	68	PR2	Programming 2
S103	John	Smith	S104	PR1	68	IAI	Intro to AI
S103	John	Smith	S104	IAI	65	DBS	Database Systems
S103	John	Smith	S104	IAI	65	PR1	Programming 1
S103	John	Smith	S104	IAI	65	PR2	Programming 2
S103	John	Smith	S104	IAI	65	IAI	Intro to AI
S103	John	Smith	S106	PR2	43	DBS	Database Systems
S103	John	Smith	S106	PR2	43	PR1	Programming 1
S103	John	Smith	S106	PR2	43	PR2	Programming 2
S103	John	Smith	S106	PR2	43	IAI	Intro to AI
S103	John	Smith	S107	PR1	76	DBS	Database Systems

Again, we must filter to retain only rows that are **true**

```
SELECT *
FROM student, grade, course
WHERE student.id = grade.id
AND grade.code = course.code
```

id	first	last	id	code	mark	code	title
S103	John	Smith	S103	DBS	72	DBS	Database Systems
S103	John	Smith	S103	DBS	72	PR1	Programming 1
S103	John	Smith	S103	DBS	72	PR2	Programming 2
S103	John	Smith	S103	DBS	72	IAI	Intro to AI
S103	John	Smith	S103	IAI	58	DBS	Database Systems
S103	John	Smith	S103	IAI	58	PR1	Programming 1
S103	John	Smith	S103	IAI	58	PR2	Programming 2
S103	John	Smith	S103	IAI	58	IAI	Intro to AI
S103	John	Smith	S104	PR1	68	DBS	Database Systems
S103	John	Smith	S104	PR1	68	PR1	Programming 1
S103	John	Smith	S104	PR1	68	PR2	Programming 2
S103	John	Smith	S104	PR1	68	IAI	Intro to AI
S103	John	Smith	S104	IAI	65	DBS	Database Systems
S103	John	Smith	S104	IAI	65	PR1	Programming 1
S103	John	Smith	S104	IAI	65	PR2	Programming 2
S103	John	Smith	S104	IAI	65	IAI	Intro to AI
S103	John	Smith	S106	PR2	43	DBS	Database Systems
S103	John	Smith	S106	PR2	43	PR1	Programming 1
S103	John	Smith	S106	PR2	43	PR2	Programming 2
S103	John	Smith	S106	PR2	43	IAI	Intro to AI
S103	John	Smith	S107	PR1	76	DBS	Database Systems

Again, we must filter to retain only rows that are **true**

```
SELECT *
FROM student, grade, course
WHERE student.id = grade.id
AND grade.code = course.code
```

id	first	last	id	code	mark	code	title
S103	John	Smith	S103	DBS	72	DBS	Database Systems
S103	John	Smith	S103	IAI	58	IAI	Intro to AI
S104	Mary	Jones	S104	PR1	68	PR1	Programming 1
S104	Mary	Jones	S104	IAI	65	IAI	Intro to AI
S106	Mark	Jones	S106	PR2	43	PR2	Programming 2
S107	John	Brown	S107	PR1	76	PR1	Programming 1
S107	John	Brown	S107	PR2	60	PR2	Programming 2
S107	John	Brown	S107	IAI	35	IAI	Intro to AI

id	first	last	id	code	mark	code	title
S103	John	Smith	S103	DBS	72	DBS	Database Systems
S103	John	Smith	S103	IAI	58	IAI	Intro to AI
S104	Mary	Jones	S104	PR1	68	PR1	Programming 1
S104	Mary	Jones	S104	IAI	65	IAI	Intro to AI
S106	Mark	Jones	S106	PR2	43	PR2	Programming 2
S107	John	Brown	S107	PR1	76	PR1	Programming 1
S107	John	Brown	S107	PR2	60	PR2	Programming 2
S107	John	Brown	S107	IAI	35	IAI	Intro to AI

Finally we may manipulate the table in the standard fashion, i.e.:

```
SELECT first, last
FROM student, grade, course
WHERE student.id = grade.id
AND grade.code = course.code
AND title = 'Intro to AI'
```

id	first	last	id	code	mark	code	title
S103	John	Smith	S103	IAI	58	IAI	Intro to AI
S104	Mary	Jones	S104	IAI	65	IAI	Intro to AI
S107	John	Brown	S107	IAI	35	IAI	Intro to AI

Some observations...

grade

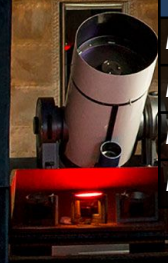
id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

students

id	first	last	age
S103	John	Smith	22
S104	Mary	Jones	27
S105	Jane	Brown	21
S106	Mark	Jones	26
S107	John	Brown	30

course

code	title
DBS	Database Systems
PR1	Programming 1
PR2	Programming 2
IAI	Intro to AI



Joining tables gives us:

New set of entities

(row: "Graded student", new set of jointly true facts)

Some observations...

grade

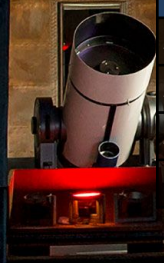
id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

students

id	first	last	age
S103	John	Smith	22
S104	Mary	Jones	27
S105	Jane	Brown	21
S106	Mark	Jones	26
S107	John	Brown	30

course

code	title
DBS	Database Systems
PR1	Programming 1
PR2	Programming 2
IAI	Intro to AI



Joining tables gives us:

New set of entities

(row: "Graded student", new set of jointly true facts)

Does the joint entity you're making make sense?
Will the each **rows facts** be **jointly true?**

Requires fields to implicitly create links we trust.

Some observations...

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

students

id	first	last	age
S103	John	Smith	22
S104	Mary	Jones	27
S105	Jane	Brown	21
S106	Mark	Jones	26
S107	John	Brown	30

course

code	title
DBS	Database Systems
PR1	Programming 1
PR2	Programming 2
IAI	Intro to AI

Joining tables gives us:

New set of entities

(row: "Graded student", new set of jointly true facts)

Does the joint entity you're making make sense?
Will the each **rows facts** be **jointly true?**

Requires fields to implicitly create links we trust.

Links considered at design-time: unambiguous
(e.g. Bob's ID in hospital & chef table
e.g. transaction_id in basket and line_items table)

Other links (e.g. linking by date)**:**
Opportunities! But think carefully...

Understanding when to link:

Documentation

Documentation describes **fields that link data** and tell us **the relationship** (how to interpret the result after linking)

Recall:

Candidate/Primary keys:
Uniquely identify a entity occurrence (row)

Foreign keys:
Record an identifier (link) to a candidate key in another table

Understanding when to link:

Documentation

Documentation describes **fields that link data** and tell us **the relationship** (how to interpret the result after linking)

How we **interpret** the resulting **entity instances** is guided by the relationship information
(documented = enforced)
(non-documented = at our own risk)

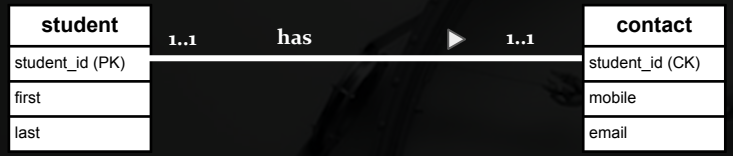
Recall:

Candidate/Primary keys:
Uniquely identify a entity occurrence (row)

Foreign keys:
Record an identifier (link) to a candidate key in another table

One-to-one

Relationship is always implicit, and is realised due to the JOIN and the constraints (enforced by design or opportunistic) on to the linking fields.



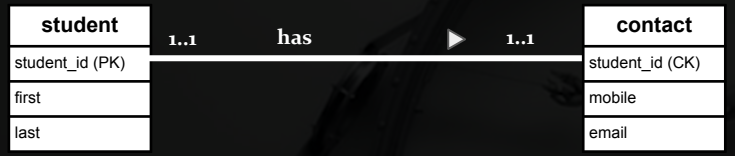
id	first	last
1	*Gav*	*Smith*
2	*Bob*	*Brown*

id	mobile	email
1	*123*	*g@s*
2	*586*	*b@b*

One-to-one

Relationship is always implicit, and is realised due to the JOIN and the constraints (enforced by design or opportunistic) on to the linking fields.

Consider the join of the tables in this example. All rows from **student** are joined with all rows from **contact.**



id	first	last
1	*Gav*	*Smith*
2	*Bob*	*Brown*

id	mobile	email
1	*123*	*g@s*
2	*586*	*b@b*

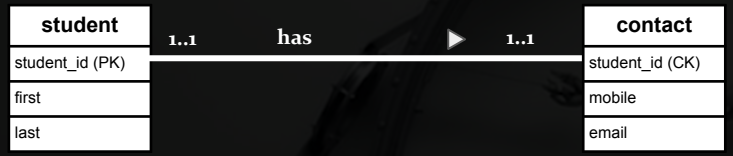
id	first	last	id	mobile	email
1	*Gav*	*Smith*	*1*	*123*	*g@s*
1	*Gav*	*Smith*	*2*	*586*	*b@b*
2	*Bob*	*Brown*	*1*	*123*	*g@s*
2	*Bob*	*Brown*	*2*	*586*	*b@b*

One-to-one

Relationship is always implicit, and is realised due to the JOIN and the constraints (enforced by design or opportunistic) on to the linking fields.

Consider the join of the tables in this example. All rows from **student** are joined with all rows from **contact.**

Due to the PK and CK constraints on the linking fields, only one row per student possible in the output table.



id	first	last
1	Gav	Smith
2	Bob	Brown

id	mobile	email
1	123	g@s
2	586	b@b

id	first	last	id	mobile	email
1	Gav	Smith	1	123	g@s
~~1~~	~~Gav~~	~~Smith~~	~~2~~	~~586~~	~~b@b~~
~~2~~	~~Bob~~	~~Brown~~	~~1~~	~~123~~	~~g@s~~
2	Bob	Brown	2	586	b@b

id	first	last	mobile	email
1	Gav	Smith	123	g@s
2	Bob	Brown	486	b@b

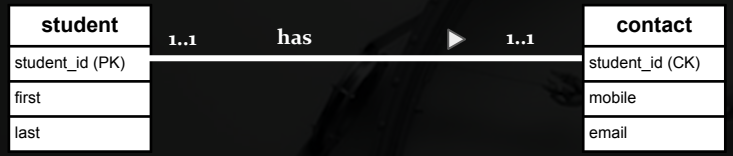
One-to-one

(enforced by cust_id being primary/candidate keys in each table)

Relationship is always implicit, and is realised due to the JOIN and the constraints (enforced by design or opportunistic) on to the linking fields.

Consider the join of the tables in this example. All rows from **student** are joined with all rows from **contact.**

Due to the PK and CK constraints on the linking fields, only one row per student possible in the output table.



id	first	last
1	*Gav*	*Smith*
2	*Bob*	*Brown*

id	mobile	email
1	*123*	*g@s*
2	*586*	*b@b*

Interpretation is simple. Equivalently:

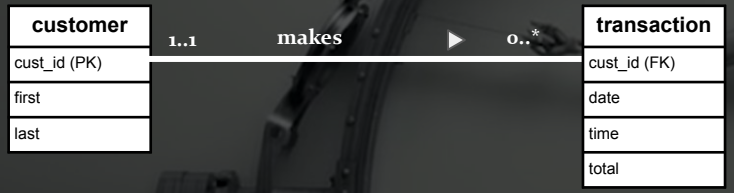
→ Each **contact** record is **extended** with **student** information.

→ Each **student** record is **extended** with **contact** information.

id	first	last	mobile	email
1	*Gav*	*Smith*	*123*	*g@s*
2	*Bob*	*Brown*	*486*	*b@b*

One-to-many

(enforced by cust_id being primary key in the "one end" of the table and a foreign key in the "many end" of the relationship)



Entity type: customer

cust_id	first	last
1	Mark	Jones
2	Victoria	Smith
3	Jane	Doe

Entity type: transaction

cust_id	date	time	total
1	2017-08-09	08:05:32	£12.00
1	2017-08-10	13:00:00	£6.50
3	2017-08-10	11:00:01	£3.00

Most Logical (maybe) Interpretation

(but not the easiest to use to "think" in SQL as it is a conceptual nested structure - not the basis of SQL!)

Each customer has a list of transactions.

Could view this as a customer "entity" with multiple transactions.

Mark

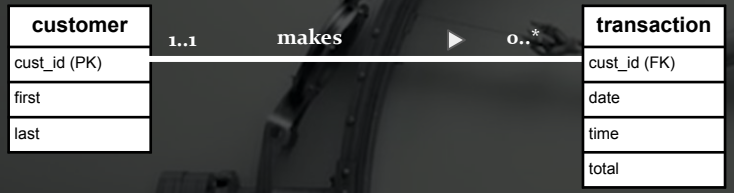
→ 2017-08-09	08:05:32	£12.00
→ 2017-08-10	13:00:00	£6.50

Jane

→ 2017-08-10	11:00:01	£3.00

One-to-many

(enforced by cust_id being primary key in the "one end" of the table and a foreign key in the "many end" of the relationship)



Entity type: customer

cust_id	first	last
1	Mark	Jones
2	Victoria	Smith
3	Jane	Doe

Entity type: transaction

cust_id	date	time	total
1	2017-08-09	08:05:32	£12.00
1	2017-08-10	13:00:00	£6.50
3	2017-08-10	11:00:01	£3.00

Most Logical (maybe) Interpretation

(but not the easiest to use to "think" in SQL as it is a conceptual nested structure - not the basis of SQL!)

Each customer has a list of transactions.

Could view this as a customer "entity" with multiple transactions.

Mark

→

2017-08-09	08:05:32	£12.00
2017-08-10	13:00:00	£6.50

Jane

→

2017-08-10	11:00:01	£3.00

Relationship is always implicit, and is realised due to the JOIN and the constraints (enforced by design or opportunistic) on to the linking fields.

Enforced by

→ **cust_id being primary key in the "one end"** of the table

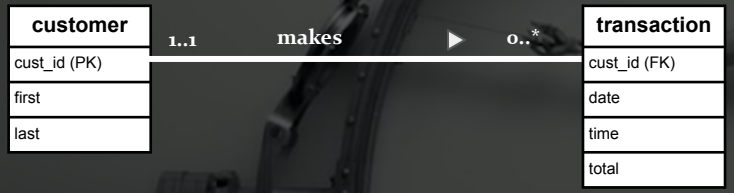
→ a **foreign key in the "many end"** of the relationship

(*records the cust_id of the transaction)*

→ When all rows from each table are compared, only true rows are those where the customer id's match.

I.e. where the transaction record has been extended with the right customer information.

One-to-many



Potentially easier if we stay "thinking in tables"

Alternate interpretation
View as a transaction entity augmented with customer information.

Entity type: customer

cust_id	first	last
1	Mark	Jones
2	Victoria	Smith
3	Jane	Doe

Entity type: transaction

cust_id	date	time	total
1	2017-08-09	08:05:32	£12.00
1	2017-08-10	13:00:00	£6.50
3	2017-08-10	11:00:01	£3.00

cust_id	first	last	date	time	total
1	Mark	Jones	2017-08-09	08:05:32	£12.00
1	Mark	Jones	2017-08-10	13:00:00	£6.50
3	Jane	Doe	2017-08-10	11:00:01	£3.00

One-to-many

Remember:

The result of joining two tables is a new table.

Always think:
What is the entity type represented?

Relationships (particularly if they are enforced) guide our interpretation.

Each customer has a list of transactions.

Could view this as a customer "entity" with multiple transactions.

Mark

2017-08-09	08:05:32	£12.00
2017-08-10	13:00:00	£6.50

Jane

2017-08-10	11:00:01	£3.00

Identical structure information. But we have a table. **Keeping things simple for SQL.**

cust_id	first	last	date	time	total
1	Mark	Jones	2017-08-09	08:05:32	£12.00
			2017-08-10	13:00:00	£6.50
3	Jane	Doe	2017-08-10	11:00:01	£3.00

Many-to-many

Entity type: basket

b_id	time	loyalty_id	total
b1	11am	7896	£3.70
b2	1pm	9934	£3.25
b3	2pm	3900	£0.07

Entity type: items

i_id	desc.	cost
i1	Doritos	£3.00
i2	Chocolate bar	£0.70
i3	Sandwich	£3.25

Many-to-many

Should not exist in a well designed relational database.

Interpretation

A basket entity (row) would need to store many items. A row must, however, only be atomic entities.

Entity type: basket

b_id	time	loyalty_id	total
b1	11am	7896	£3.70
b2	1pm	9934	£3.25
b3	2pm	3900	£0.07

Entity type: items

i_id	desc.	cost
i1	Doritos	£3.00
i2	Chocolate bar	£0.70
i3	Sandwich	£3.25

Many-to-many

Should not exist in a well designed relational database.

Interpretation

If it did, the items table is no longer "items" as we'd conceptualize it (one item is one entity occurrence).

Entity type: basket

b_id	time	loyalty_id	total
b1	11am	7896	£3.70
b2	1pm	9934	£3.25
b3	2pm	3900	£0.07

Entity type: items

i_id	desc.	cost
i1	Doritos	£3.00
i2	Chocolate bar	£0.70
i3	Sandwich	£3.25

Same information, corrected to being actual "tables". Becomes three tables and two one-to-many relationships.

AGGREGATE FUNCTIONS

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

AGGREGATE FUNCTIONS

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

OK, so joins are fun...

But let's step back...

And look at operations on a single (or post join) table.

Question: What is the maximum mark per course?

Answer in SQL:

```
SELECT code, MAX(mark)
FROM grade
GROUP BY code
```

Step 1: Group (put) the rows into buckets, one bucket per distinct module code value

AGGREGATE FUNCTIONS

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

OK, so joins are fun...

But let's step back...

And look at operations on a single (or post join) table.

Question: What is the maximum mark per course?

Answer in SQL:

```
SELECT code, MAX(mark)
FROM grade
GROUP BY code
```

Step 2: Per group (bucket) compute the aggregate function (in this case max) over all values in the bucket

Step 1: Group (put) the rows into buckets, one bucket per distinct module code value

AGGREGATE FUNCTIONS

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

OK, so joins are fun...

But let's step back...

And look at operations on a single (or post join) table.

Question: What is the maximum mark per course?

Answer in SQL:

```
SELECT code, MAX(mark)
FROM grade
GROUP BY code
```

Step 3: Return the bucket label and the aggregate values.

NOTE: Bucket labels need not be included. Fields not part of a bucket label can't be.

WHY? Ambiguous, what value of the many should have been picked per row?

Step 2: Per group (bucket) compute the aggregate function (in this case max) over all values in the bucket

Step 1: Group (put) the rows into buckets, one bucket per distinct module code value

AGGREGATE FUNCTIONS

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

What about if we want just one big bucket?

(group over everything)
(max over all rows)

Question: What is the maximum mark?

Answer in SQL:

```
SELECT      MAX(mark)
FROM grade
```

AGGREGATE FUNCTIONS

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

What about if we want just one big bucket?

(group over everything)
(max over all rows)

Question: What is the maximum mark?

Answer in SQL:

```
SELECT      MAX(mark)
FROM grade
```

Step 0: Don't specify bucket labels.

AGGREGATE FUNCTIONS

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

What about if we want just one big bucket?

(group over everything)
(max over all rows)

Question: What is the maximum mark?

Answer in SQL:

```
SELECT MAX(mark)
FROM grade
```

Step 2: The aggregate function (in this case max) is computed over all values.

Step 0: Don't specify bucket labels.

AGGREGATE FUNCTIONS

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

What about if we want just one big bucket?

(group over everything)
(max over all rows)

Question: What is the maximum mark?

Answer in SQL:

```
SELECT MAX(mark)
FROM grade
```

Step 3: Return the aggregate values

NOTE: No bucket labels, no non-aggregate terms.

WHY? Ambiguous, what value of the many should have been picked per row?

Step 2: The aggregate function (in this case max) is computed over all values.

Step 1: Don't specify bucket labels.

AGGREGATE FUNCTIONS

grade

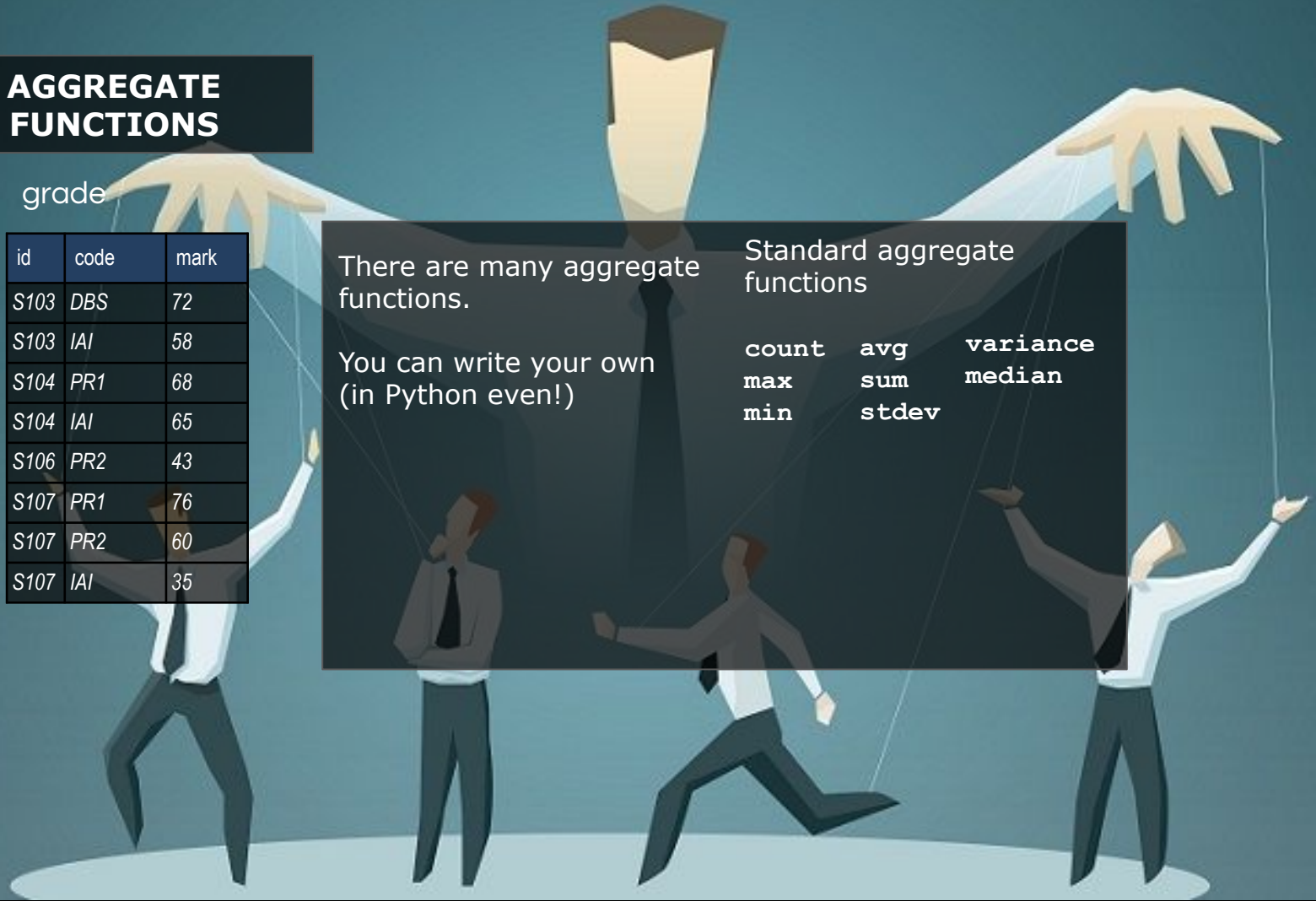
id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

There are many aggregate functions.

You can write your own (in Python even!)

Standard aggregate functions

```
count   avg     variance
max     sum     median
min     stdev
```



AGGREGATE FUNCTIONS

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

All operate under the same principal.

Work on the bucket of data given by the group by.

More than one aggregate can be run over the buckets at once.

```
SELECT code,
       MIN(mark),
       MAX(mark),
       AVG(mark),
FROM grade
GROUP BY code
```

id	code	mark
S103	DBS	72
S103	IAI	58
S104	IAI	65
S107	IAI	35
S107	PR1	76
S104	PR1	68
S107	PR2	60
S106	PR2	43

code	min	max	avg
DBS	72	72	72
IAI	35	65	52.6
PR1	68	76	72
PR2	43	60	51.5

Conditions on AGGREGATE FUNCTIONS

Aggregate function **without conditions**

id	code	mark
S103	DBS	72
S103	IAI	58
S104	IAI	65
S107	IAI	35
S107	PR1	76
S104	PR1	68
S107	PR2	60
S106	PR2	43

code	avg
DBS	72
IAI	52.6
PR1	72
PR2	51.5

```
SELECT code,
       AVG(mark)
FROM grade

GROUP BY code
```

Conditions on AGGREGATE FUNCTIONS

Where conditions operate **BEFORE** the group by.

id	code	mark
S103	DBS	72
S103	IAI	58
S104	IAI	65
S107	IAI	35
S107	PR1	76
S104	PR1	68
S107	PR2	60
S106	PR2	43

id	code	mark
S104	IAI	65
S107	IAI	35
S107	PR1	76
S104	PR1	68
S107	PR2	60
S106	PR2	43

code	avg
IAI	48.5
PR1	72
PR2	51.5

```
SELECT code,
       AVG(mark)
FROM grade
WHERE id != 'S103'
GROUP BY code
```

Conditions on AGGREGATE FUNCTIONS

New keyword **HAVING** operates **after t**he group by.

id	code	mark
S103	DBS	72
S103	IAI	58
S104	IAI	65
S107	IAI	35
S107	PR1	76
S104	PR1	68
S107	PR2	60
S106	PR2	43

id	code	mark
S104	IAI	65
S107	IAI	35
S107	PR1	76
S104	PR1	68
S107	PR2	60
S106	PR2	43

code	avg
IAI	48.5
PR1	72
PR2	51.5

code	avg
PR1	72
PR2	51.5

```
SELECT code,
       AVG(mark)
FROM grade
WHERE id != 'S103'
GROUP BY code
HAVING AVG(mark) > 50
```

We now know a lot...

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

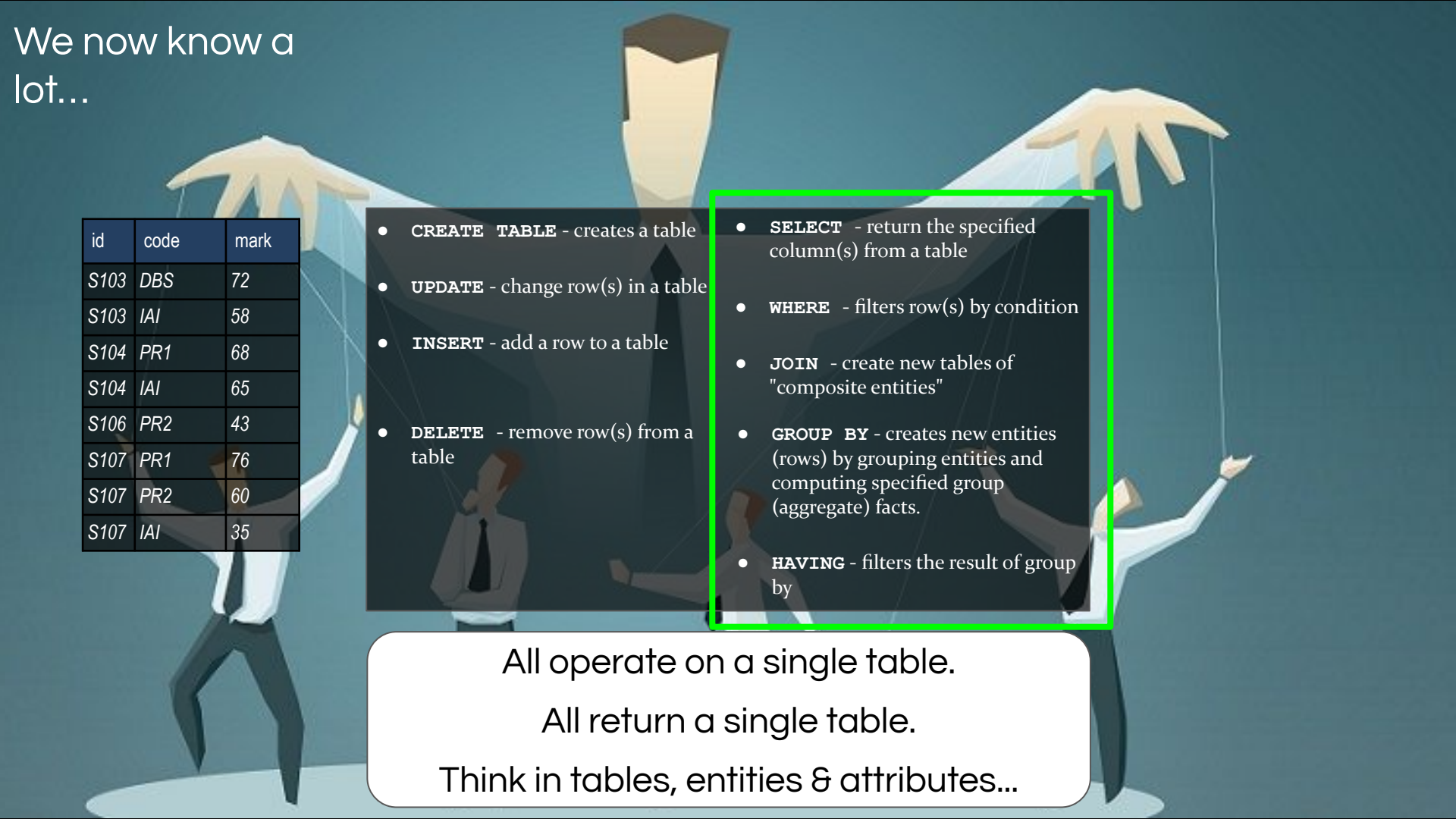
- **`CREATE TABLE`** - creates a table
- **`UPDATE`** - change row(s) in a table
- **`INSERT`** - add a row to a table
- **`DELETE`** - remove row(s) from a table
- **`SELECT`** - return the specified column(s) from a table
- **`WHERE`** - filters row(s) by condition
- **`JOIN`** - create new tables of "composite entities"
- **`GROUP BY`** - creates new entities (rows) by grouping entities and computing specified group (aggregate) facts.
- **`HAVING`** - filters the result of group by

All operate on a single table.

All return a single table.

Think in tables, entities & attributes...

We now know a lot...



id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

- **`CREATE TABLE`** - creates a table
- **`UPDATE`** - change row(s) in a table
- **`INSERT`** - add a row to a table
- **`DELETE`** - remove row(s) from a table

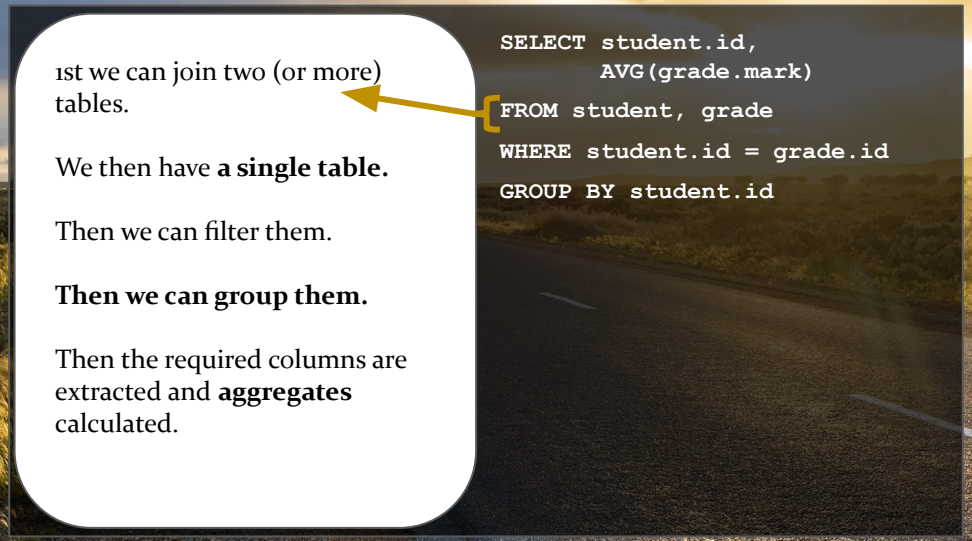
- **`SELECT`** - return the specified column(s) from a table
- **`WHERE`** - filters row(s) by condition
- **`JOIN`** - create new tables of "composite entities"
- **`GROUP BY`** - creates new entities (rows) by grouping entities and computing specified group (aggregate) facts.
- **`HAVING`** - filters the result of group by

All operate on a single table.

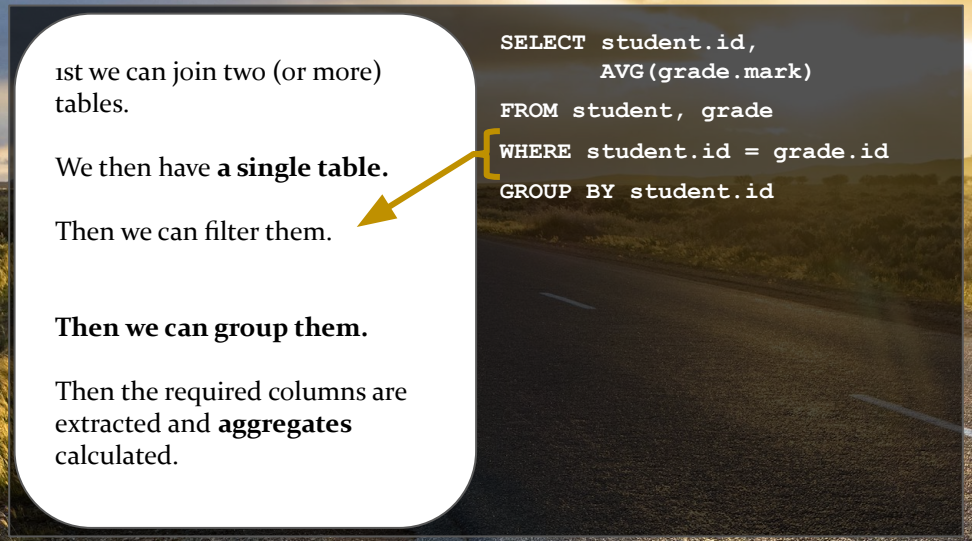
All return a single table.

Think in tables, entities & attributes...

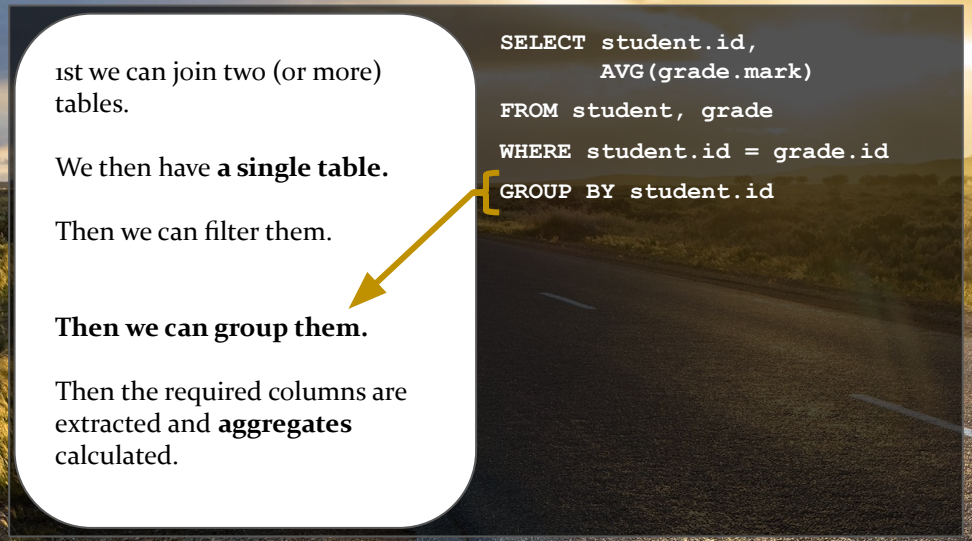
Combining this with joins...



Combining this with joins...



Combining this with joins...



Combining this with joins...

1st we can join two (or more) tables.

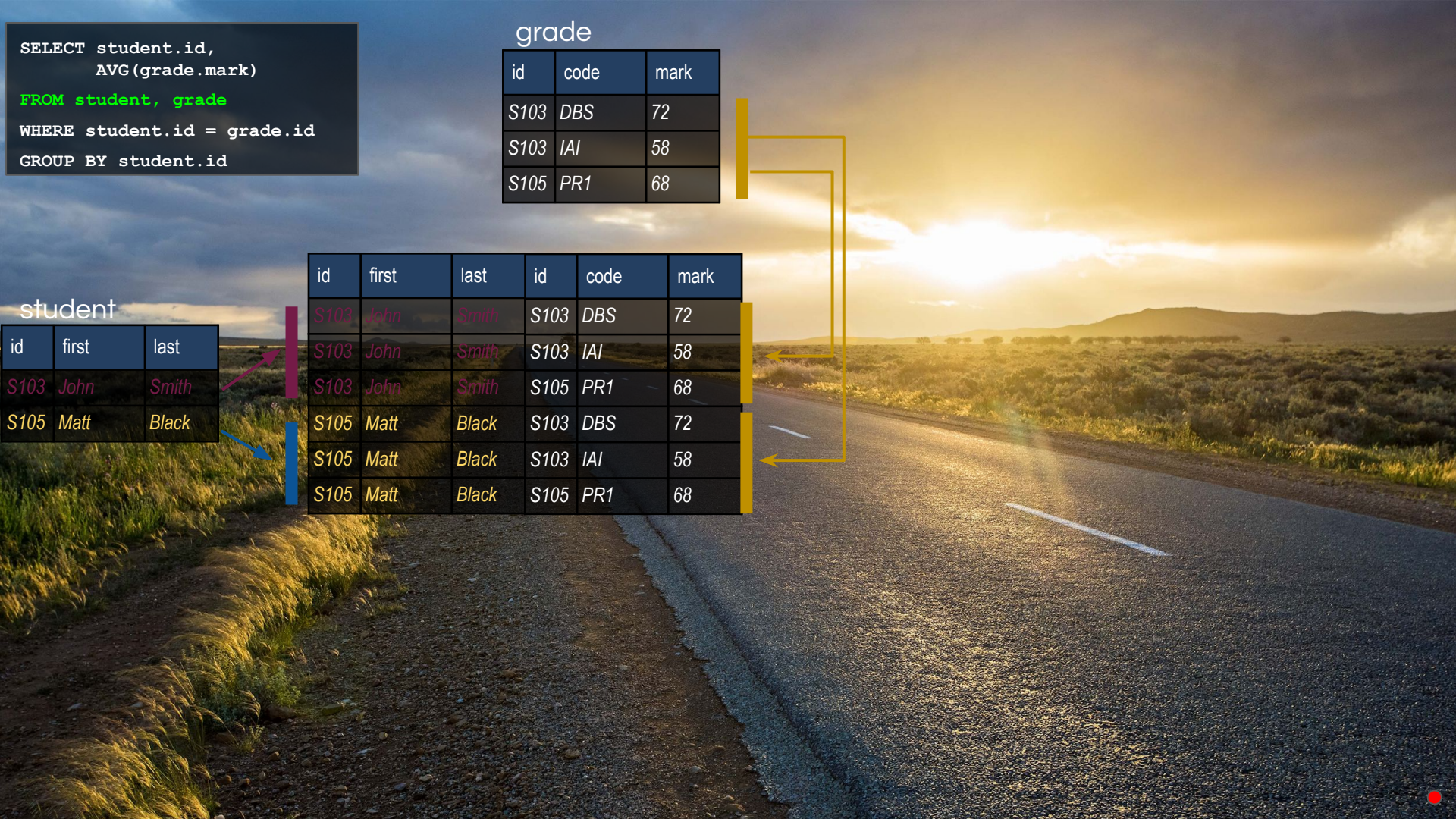
We then have **a single table.**

Then we can filter them.

Then we can group them.

Then the required columns are extracted and **aggregates** calculated.

```
SELECT student.id,
       AVG(grade.mark)
FROM student, grade
WHERE student.id = grade.id
GROUP BY student.id
```


SELECT student.id,
AVG(grade.mark)
FROM student, grade
WHERE student.id = grade.id
GROUP BY student.id
grade
id code mark
S103 DBS 72
S103 IAI 58
S105 PR1 68
student
id first last
S103 John Smith
S105 Matt Black
id first last id code mark
S103 John Smith S103 DBS 72
S103 John Smith S103 IAI 58
S103 John Smith S105 PR1 68
S105 Matt Black S103 DBS 72
S105 Matt Black S103 IAI 58
S105 Matt Black S105 PR1 68


SELECT student.id,
AVG(grade.mark)
FROM student, grade
WHERE student.id = grade.id
GROUP BY student.id
grade
id code mark
S103 DBS 72
S103 IAI 58
S105 PR1 68
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
student
id first last
S103 John Smith
S105 Matt Black
id first last id code mark
S103 John Smith S103 DBS 72
S103 John Smith S103 IAI 58
S103 John Smith S105 PR1 68
S105 Matt Black S103 DBS 72
S105 Matt Black S103 IAI 58
S105 Matt Black S105 PR1 68


SELECT student.id,
AVG(grade.mark)
FROM student, grade
WHERE student.id = grade.id
GROUP BY student.id
grade
id code mark
S103 DBS 72
S103 IAI 58
S105 PR1 68
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
id first last id code mark
S103 John Smith S103 DBS 72
S103 John Smith S103 IAI 58
S105 Matt Black S105 PR1 68
student
id first last
S103 John Smith
S105 Matt Black
id first last id code mark
S103 John Smith S103 DBS 72
S103 John Smith S103 IAI 58
S103 John Smith S105 PR1 68
S105 Matt Black S103 DBS 72
S105 Matt Black S103 IAI 58
S105 Matt Black S105 PR1 68

```
SELECT student.id,
       AVG(grade.mark)
FROM student, grade
WHERE student.id = grade.id
GROUP BY student.id
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S105	PR1	68

student

id	first	last
S103	John	Smith
S105	Matt	Black

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S103	John	Smith	S105	PR1	68
S105	Matt	Black	S103	DBS	72
S105	Matt	Black	S103	IAI	58
S105	Matt	Black	S105	PR1	68

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S105	Matt	Black	S105	PR1	68

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

id	first	last	id	code	mark
S105	Matt	Black	S105	PR1	68

```
SELECT student.id,
       AVG(grade.mark)
FROM student, grade
WHERE student.id = grade.id
GROUP BY student.id
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S105	PR1	68

student

id	first	last
S103	John	Smith
S105	Matt	Black

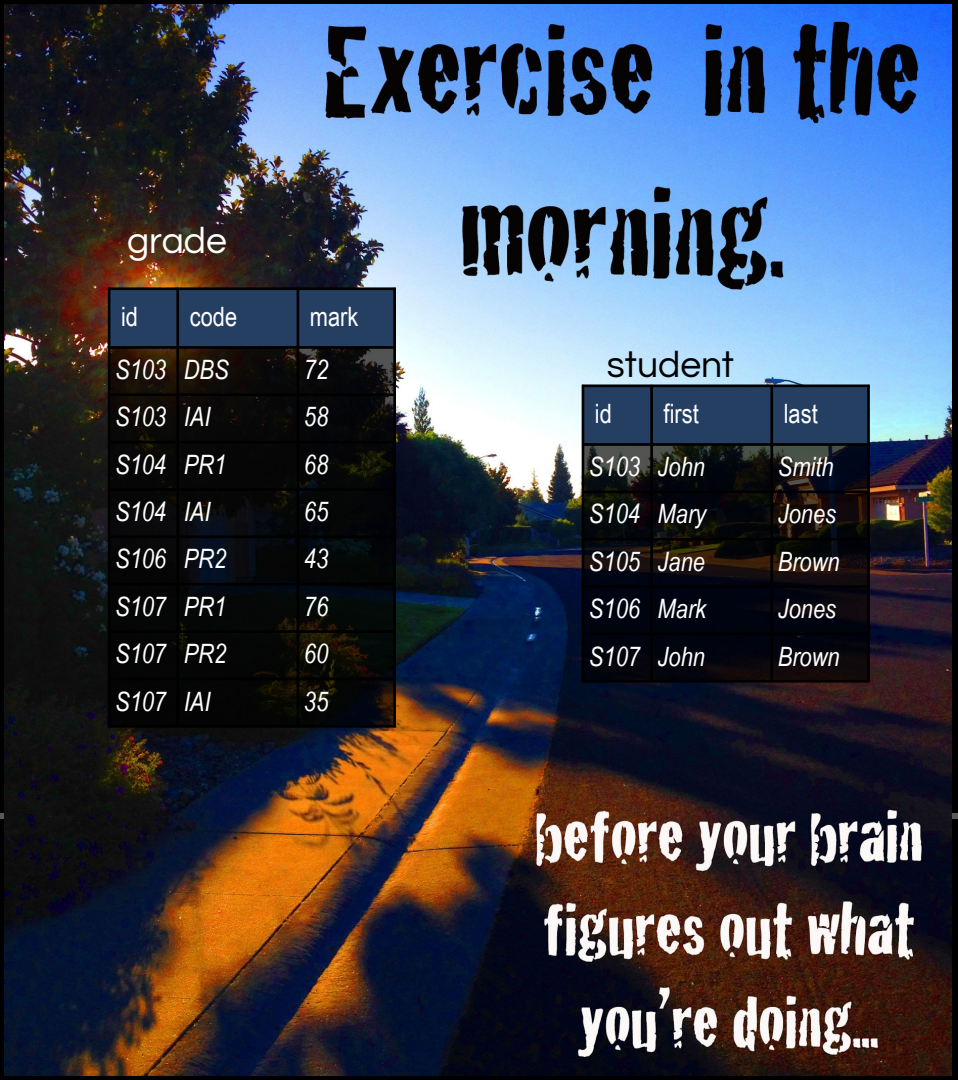
id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S103	John	Smith	S105	PR1	68
S105	Matt	Black	S103	DBS	72
S105	Matt	Black	S103	IAI	58
S105	Matt	Black	S105	PR1	68

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S105	Matt	Black	S105	PR1	68

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

id	first	last	id	code	mark
S105	Matt	Black	S105	PR1	68

id	AVG(mark)
S103	65
S105	68




```
SELECT id
FROM student
```

id
S103
S104
S105
S106
S107

```
SELECT *
FROM grade
WHERE
mark >= 60
```

id	code	mark
S103	DBS	72
S104	PR1	68
S104	IAI	65
S107	PR1	76
S107	PR2	60

```
SELECT code, mark
FROM student, grade
WHERE first = 'John' and
last = 'Smith'
```

code	mark
DBS	72
IAI	58

```
SELECT code, AVG(mark)
FROM grade
GROUP BY code
```

code	mark
DBS	72
IAI	52.667
PR1	72
PR2	53

Recap...

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68
S104	IAI	65
S106	PR2	43
S107	PR1	76
S107	PR2	60
S107	IAI	35

- JOINS via multiple tables in the FROM statement
- Filter to keep only "correct" rows

INNER JOIN

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

student

id	first	last
S103	John	Smith
S105	Jane	Brown

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S103	John	Smith	S104	PR1	68
S105	Jane	Brown	S103	DBS	72
S105	Jane	Brown	S103	IAI	58
S105	Jane	Brown	S104	PR1	68

INNER JOIN

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

student

id	first	last
S103	John	Smith
S105	Jane	Brown

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S103	John	Smith	S104	PR1	68
S105	Jane	Brown	S103	DBS	72
S105	Jane	Brown	S103	IAI	58
S105	Jane	Brown	S104	PR1	68

INNER JOIN

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

student

id	first	last
S103	John	Smith
S105	Jane	Brown

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S103	John	Smith	S104	PR1	68
S105	Jane	Brown	S103	DBS	72
S105	Jane	Brown	S103	IAI	58
S105	Jane	Brown	S104	PR1	68

What's wrong?

Lost information regarding:

(1) S105, Jane, Brown
(2) S104, PR1, 68

Might want to know that join did not match anything!

Select the students that have not recorded any grades.

student

id	first	last
S103	John	Smith
S105	Jane	Brown

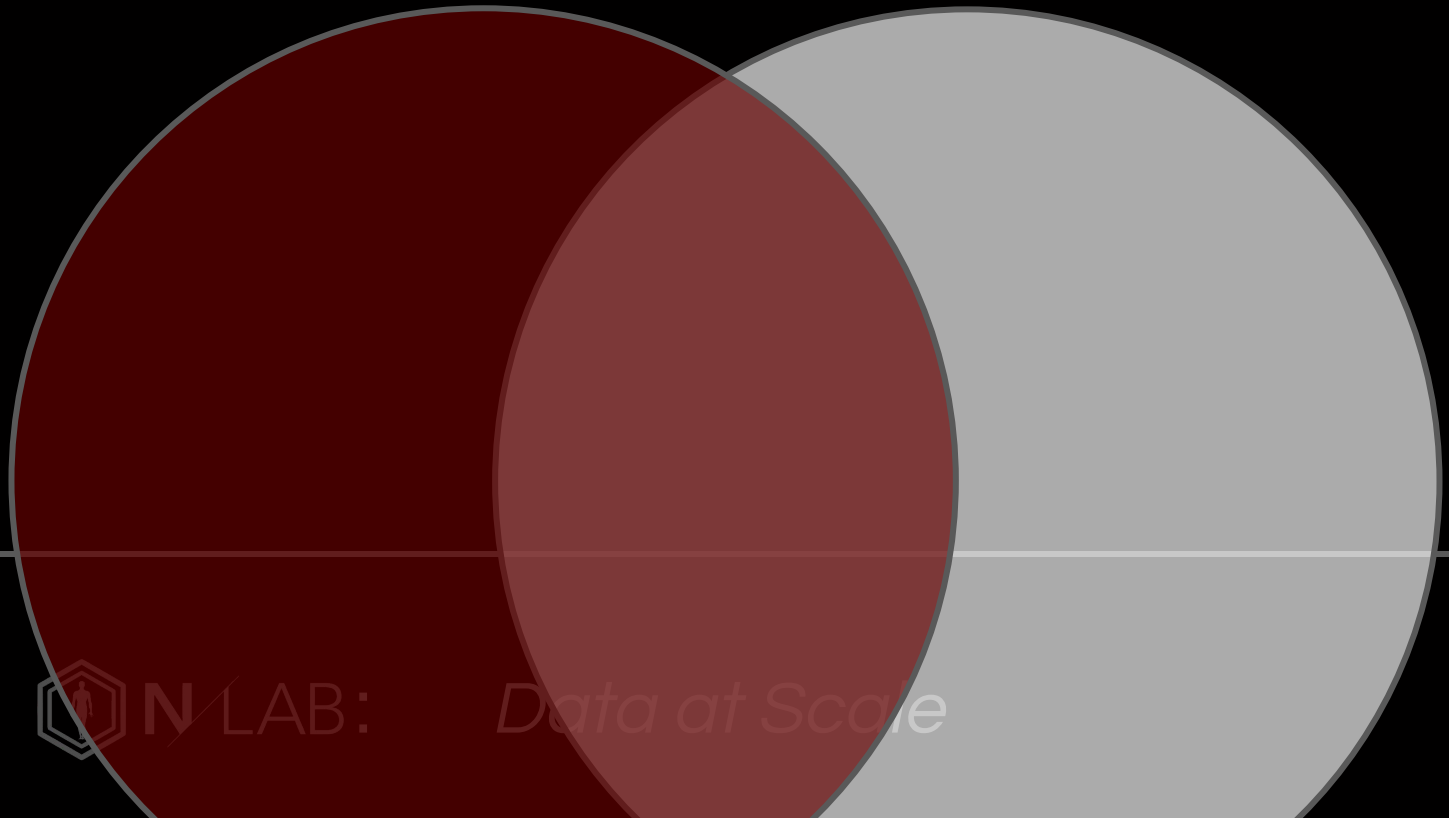
```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68

Let's consider rows in the original tables.

After the join we have...



student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68

Rows that matched and were kept (1 or more times)

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

Let's consider rows in the original tables.

After the join we have...

Dr Georgiana Nica-Avram

student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

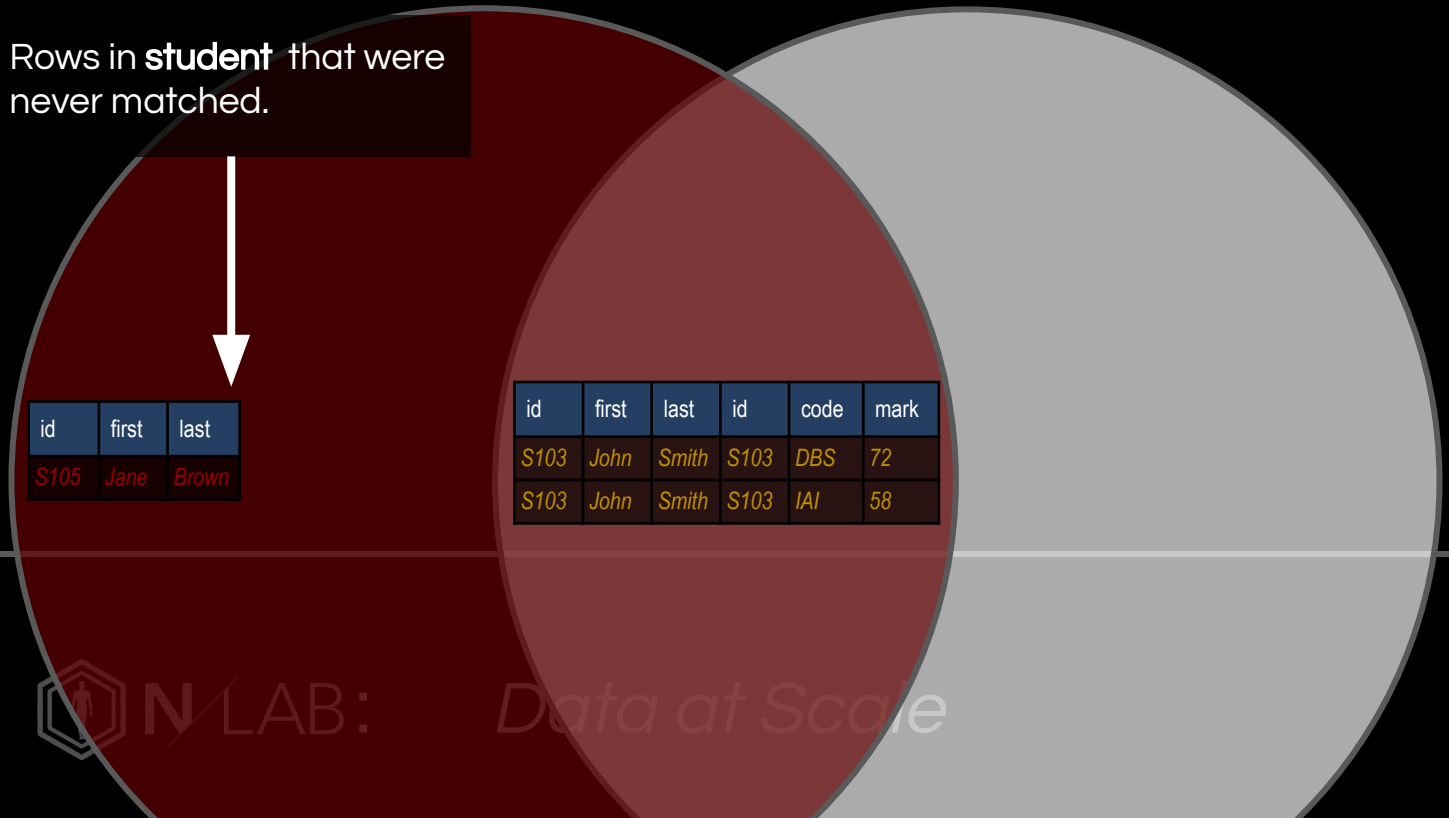
grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68

Let's consider rows in the original tables.

After the join we have...

Rows in **student** that were never matched.



id	first	last
S105	Jane	Brown

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

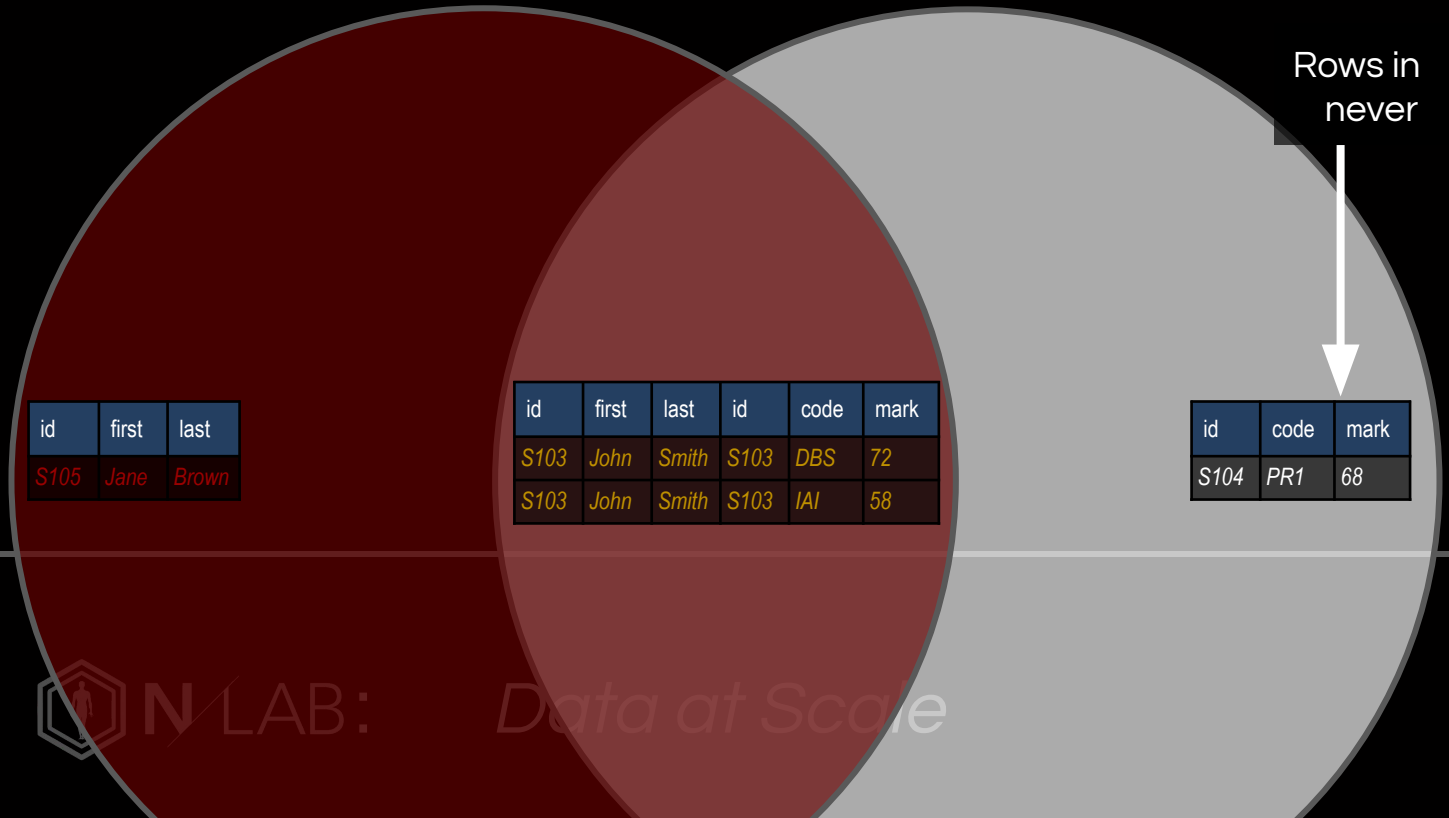
grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68

Let's consider rows in the original tables.

After the join we have...

Rows in **grade** that were never matched.



id	first	last
S105	Jane	Brown

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

id	code	mark
S104	PR1	68

Dr Georgiana Nica-Avram

student

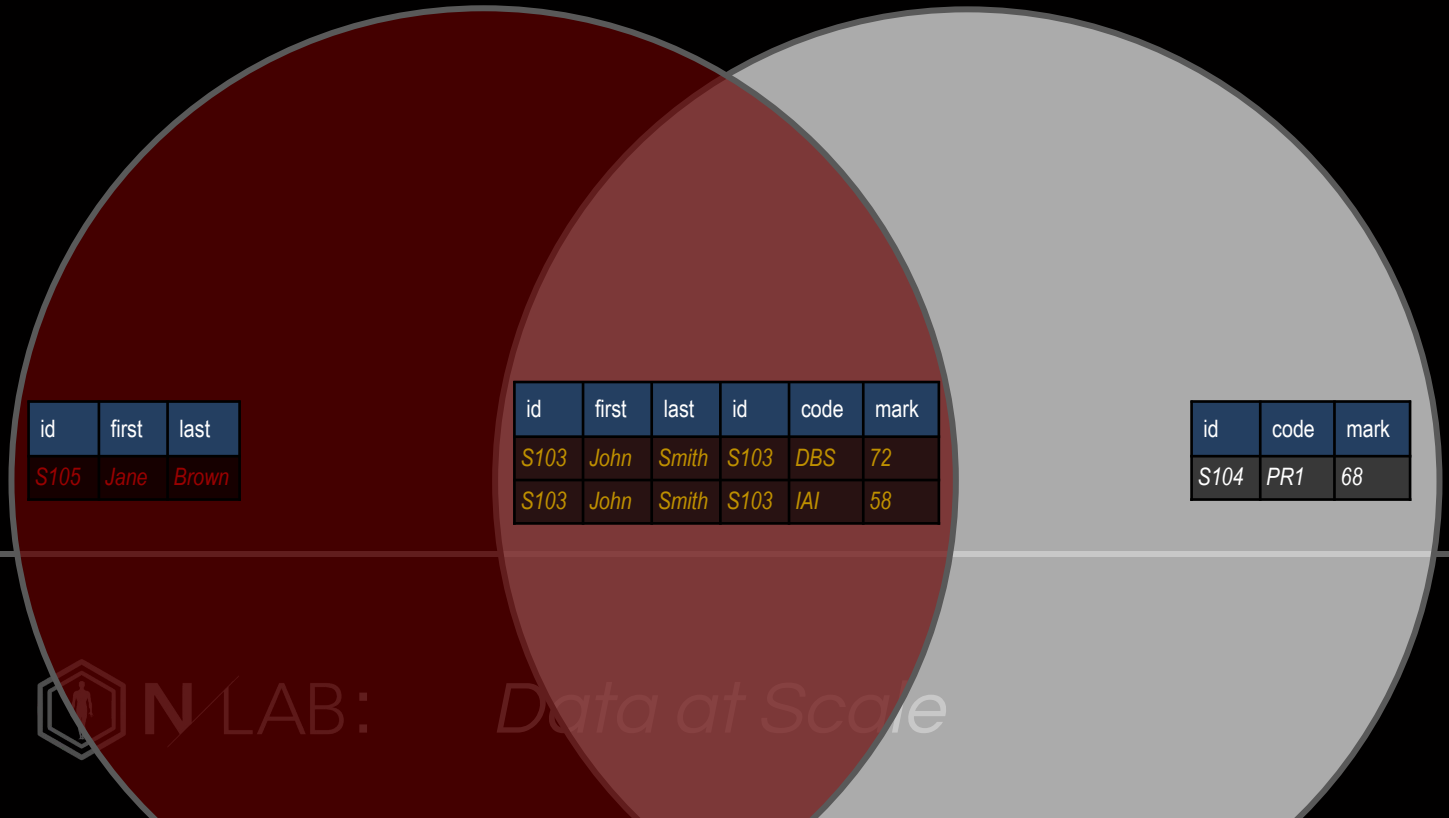
id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68

Let's explicitly encode a "did not match anything"



id	first	last
S105	Jane	Brown

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

id	code	mark
S104	PR1	68

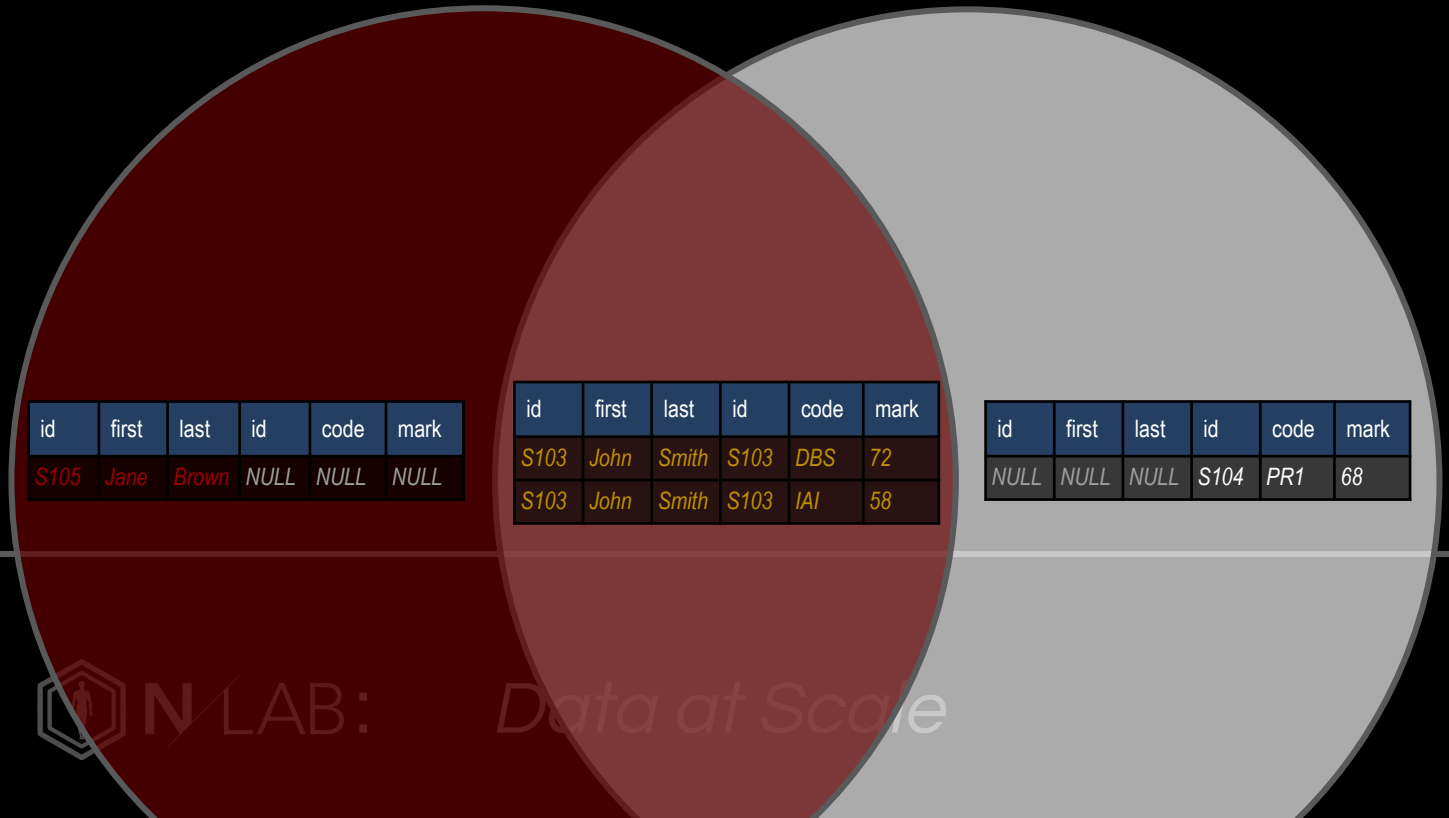
student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68



id	first	last	id	code	mark
S105	Jane	Brown	NULL	NULL	NULL

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

id	first	last	id	code	mark
NULL	NULL	NULL	S104	PR1	68

Let's explicitly encode a "did not match anything"

By matching the line to a set of NULL* values.

* Recall NULL ≈ unknown value

Dr Georgiana Nica-Avram

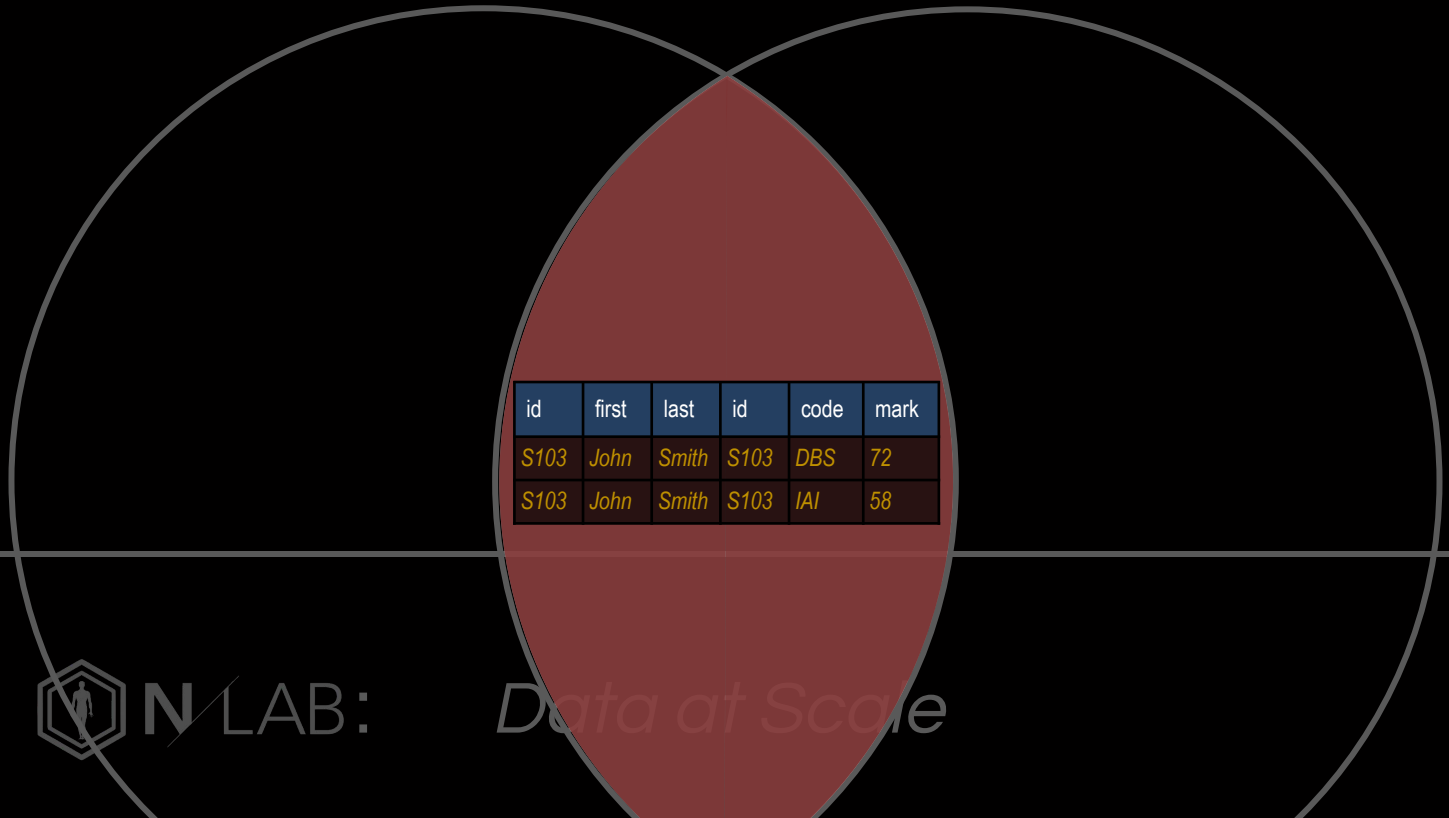
student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student, grade
WHERE (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68



id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

Back to JOINs

INNER JOIN

Only including matches is called an inner join.

Join all rows to all rows then filter on a match condition.

student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student
JOIN grade
ON (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68

New, equivalent syntax!

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

Back to JOINs
INNER JOIN

Only including matches is called an inner join.

Join all rows to all rows then filter on a match condition.

student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student
JOIN grade
ON (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68

New, equivalent syntax!

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

Back to JOINs
INNER JOIN

Only including matches is called an inner join.

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

Join all rows to all rows then filter on a match condition.

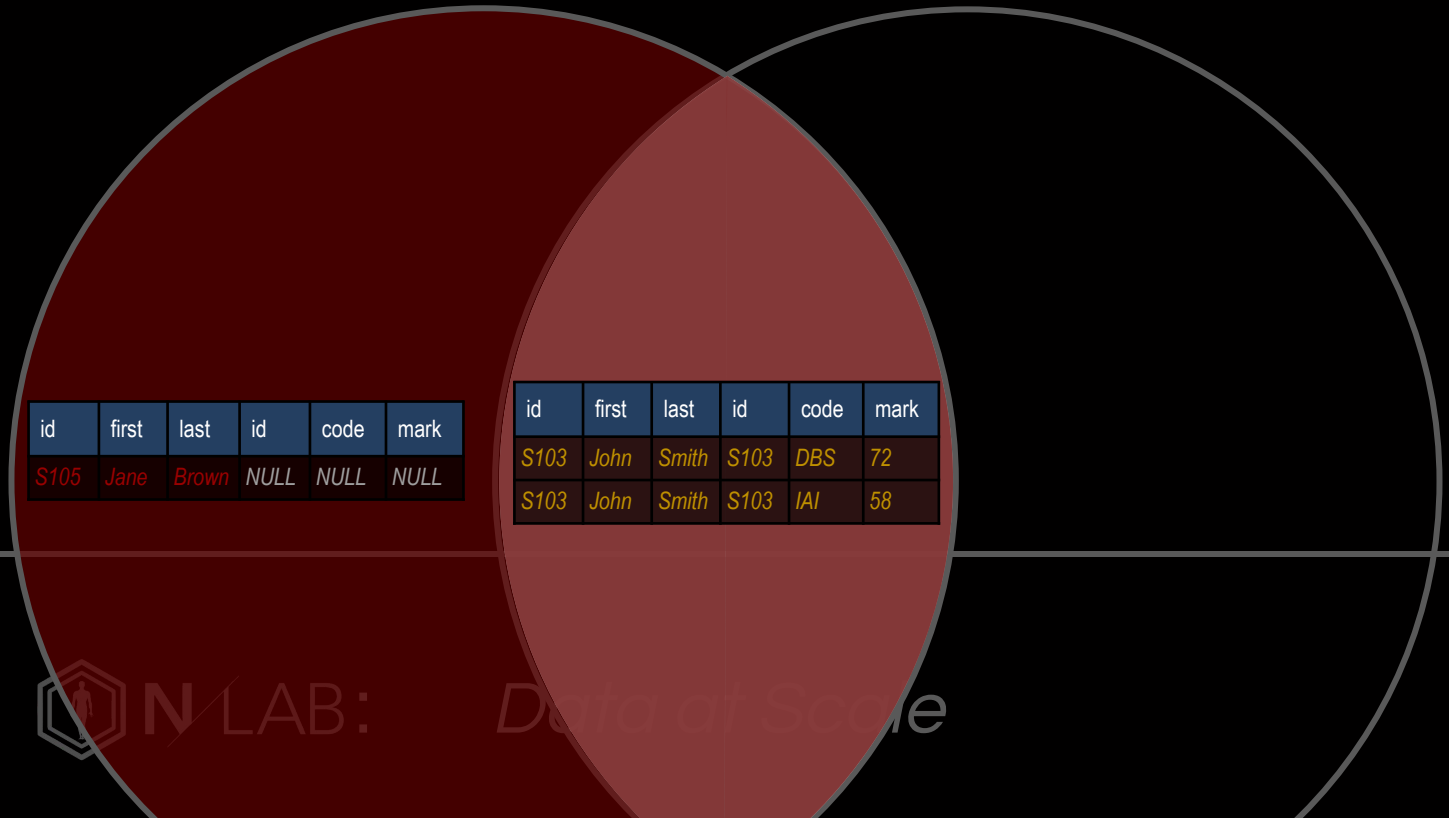
student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student
LEFT OUTER JOIN grade
ON (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68



id	first	last	id	code	mark
S105	Jane	Brown	NULL	NULL	NULL

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

Back to JOINs
LEFT OUTER JOIN

Join all rows to all rows then filter on a match condition.

THEN add any missing rows in the left table.

Dr Georgiana Nica-Avram

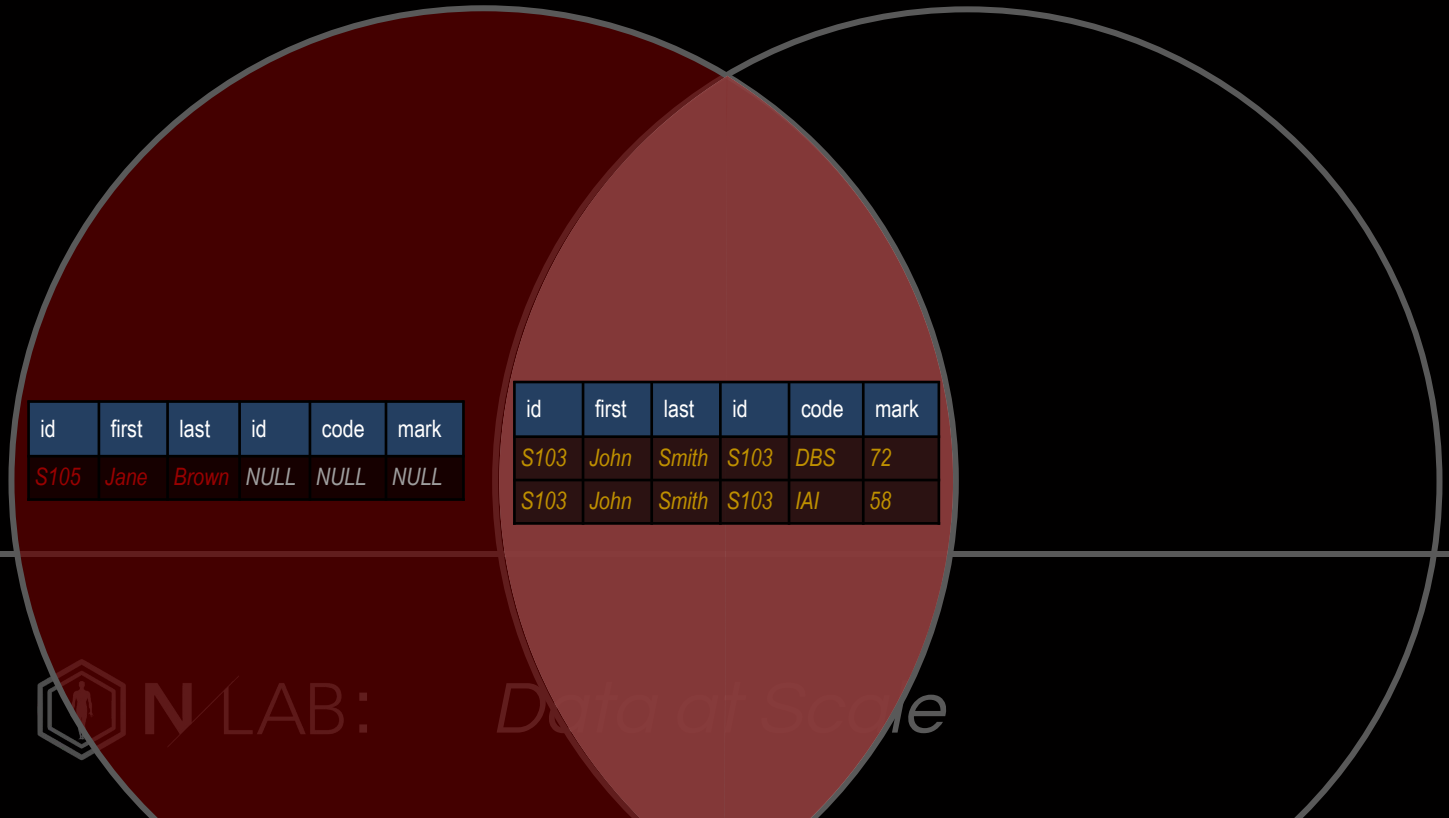
student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student
LEFT OUTER JOIN grade
ON (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68



id	first	last	id	code	mark
S105	Jane	Brown	NULL	NULL	NULL

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

Back to JOINs
LEFT OUTER JOIN

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S105	Jane	Brown	NULL	NULL	NULL

Join all rows to all rows then filter on a match condition.

THEN add any missing rows in the left table.

Dr Georgiana Nica-Avram

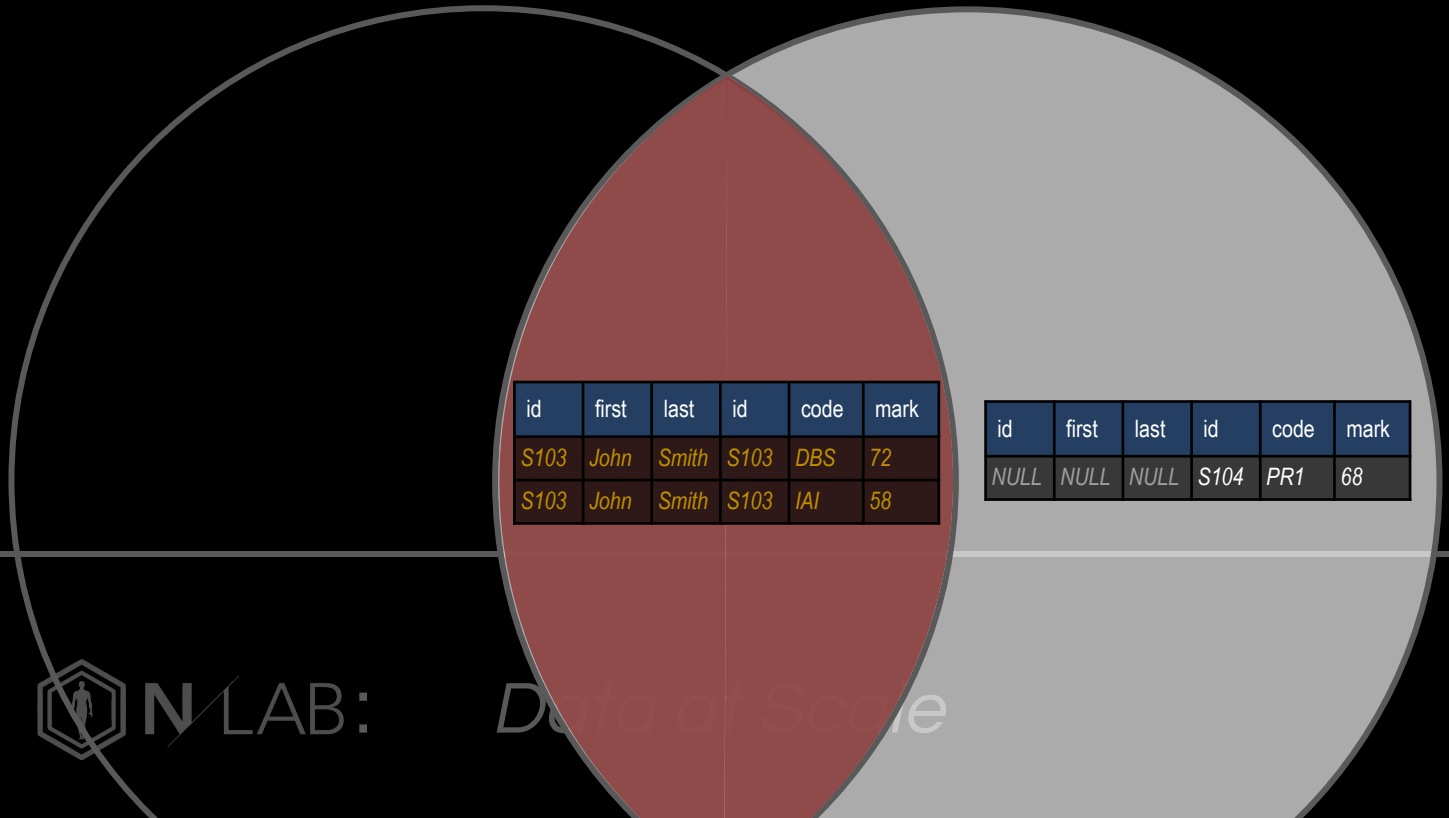
student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student
RIGHT OUTER JOIN grade
ON (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68



id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

id	first	last	id	code	mark
NULL	NULL	NULL	S104	PR1	68

Back to JOINs
RIGHT OUTER JOIN

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
NULL	NULL	NULL	S104	PR1	68

Join all rows to all rows then filter on a match condition.

THEN add any missing rows in the right table.

Dr Georgiana Nica-Avram

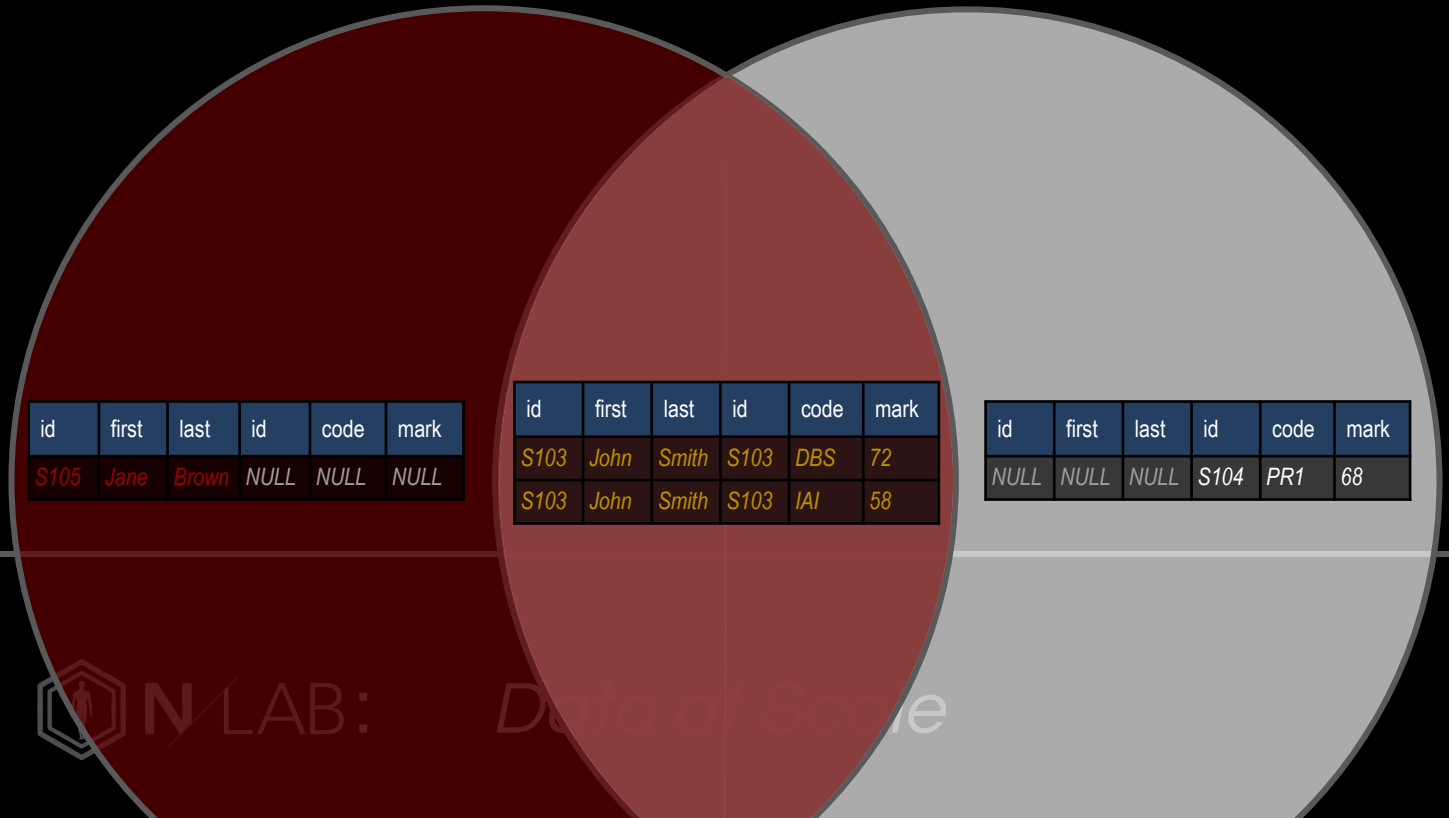
student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student
FULL OUTER JOIN grade
ON (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68



id	first	last	id	code	mark
S105	Jane	Brown	NULL	NULL	NULL

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

id	first	last	id	code	mark
NULL	NULL	NULL	S104	PR1	68

Back to JOINs
FULL OUTER JOIN

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S105	Jane	Brown	NULL	NULL	NULL
NULL	NULL	NULL	S104	PR1	68

Join all rows to all rows then filter on a match condition.

THEN add any missing rows in both the left and right tables

Dr Georgiana Nica-Avram

student

id	first	last
S103	John	Smith
S105	Jane	Brown

```
SELECT *
FROM student
FULL OUTER JOIN grade
ON (student.id = grade.id)
```

grade

id	code	mark
S103	DBS	72
S103	IAI	58
S104	PR1	68

If the field is the same in the two tables, we can use the syntax USING instead of ON.

This has the advantage that the output will only include one of the fields student.id and grade.id as we have told it is the same!

```
SELECT *
FROM student
FULL OUTER JOIN grade
USING (id)
```

id	first	last	id	code	mark
S105	Jane	Brown	NULL	NULL	NULL

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58

id	first	last	id	code	mark
NULL	NULL	NULL	S104	PR1	68

Back to JOINs
FULL OUTER JOIN

id	first	last	id	code	mark
S103	John	Smith	S103	DBS	72
S103	John	Smith	S103	IAI	58
S105	Jane	Brown	NULL	NULL	NULL
NULL	NULL	NULL	S104	PR1	68

Join all rows to all rows then filter on a match condition.

THEN add any missing rows in both the left and right tables

Dr Georgiana Nica-Avram

Dr Georgiana Nica-Avram