Filtrare i dati - GROUP BY

Le funzioni sono spesso usate in accoppiata con GROUP BY, per questo sono anche dette **funzioni di aggregazione**.

GROUP BY raggruppa le righe in cui è presente lo stesso valore della colonna indicata.

Quanti film sono usciti dal 1 gennaio 2017 in poi, per ogni nazionalità?

```sql
SELECT COUNT(id)
FROM `movies`
WHERE `date` > '2017-01-01'
GROUP BY `nationality`
```

nationality	COUNT(id)
Italian	4
Indian	10
French	2

Filtrare i dati - GROUP BY

utenti

id	nome	cognome	anno_nascita
1	Mario	Rossi	1988
2	Luigi	Verdi	1975
3	Silvia	Neri	1974
4	Francesco	Bianchi	1975
5	Anna	Gialli	1976

id	nome	cognome	anno_nascita
1	Mario	Rossi	1988

id	nome	cognome	anno_nascita
2	Luigi	Verdi	1975
4	Francesco	Bianchi	1975

id	nome	cognome	anno_nascita
3	Silvia	Neri	1974

id	nome	cognome	anno_nascita
5	Anna	Gialli	1976

COUNT(id)	anno_nascita
1	1988
2	1975
1	1974
1	1976

```sql
SELECT COUNT(id), anno_nascita
FROM utenti
GROUP BY anno_nascita
```

Azione nascosta del GROUP BY

LIVE CODING

Le query per le relazioni

Ora come facciamo a prendere dati relazionati tra più tabelle?

Dobbiamo fare una operazione di **JOIN tra due tabelle** indicando la colonna

JOIN è il modo con cui si definisce il collegamento tra due tabelle per mezzo di due colonne che sono relazionate in qualche modo.

JOIN Base

Esistono diversi tipi di JOIN:
INNER JOIN è considerata la versione base.

INNER JOIN tabella che vogliamo relazionare
ON quali colonne devono essere relazionate

```sql
SELECT *
FROM movies
INNER JOIN categories
ON categories.id = movies.category_id;
```

Per le colonne con nomi identici dobbiamo specificare la tabella di appartenenza con la **dot notation** **nomeTabella.colonna**

INNER JOIN

Player Id	First Name	Last Name
1001	Ross	Sharma
1002	Cary	Carlisle
1003	Stephan	Stroup
1004	Corean	Knott
1005	Guadalupe	Hernandez
1006	Carroll	Courtney
1007	Terry	Starr
1008	Rosevelt	Pruett
1009	Zelma	Roddy
1010	Elmina	Fay

Player Id	High Score
1001	72
1002	7
1003	20
1004	10
1005	87
1006	27
1007	59
1008	83
1009	53
1010	41

Player Id	First Name	Last Name	High Score
1001	Ross	Sharma	72
1002	Cary	Carlisle	7
1003	Stephan	Stroup	20
1004	Corean	Knott	10
1005	Guadalupe	Hernandez	87
1006	Carroll	Courtney	27
1007	Terry	Starr	59
1008	Rosevelt	Pruett	83
1009	Zelma	Roddy	53
1010	Elmina	Fay	41

INNER vs OUTER JOIN

INNER JOIN inserisce tra i risultati **solamente i record in cui c'è perfetta corrispondenza** dei dati tra una tabella e l'altra.

OUTER JOIN inserisce tra i risultati **anche i record che non hanno alcuna corrispondenza** in una delle due tabelle.

Prendiamo ad esempio il caso in cui un film non ha nessuna categoria associata:

Utilizzando **INNER JOIN**,
il film viene **escluso** dai risultati.

Utilizzando **OUTER JOIN**,
il film viene **incluso** tra i risultati.

OUTER JOIN

Sono di due tipi:

LEFT JOIN restituisce tutti i valori della tabella di sinistra anche nel caso in cui non ci sia una relazione con la tabella di destra.

RIGHT JOIN restituisce tutti i valori della tabella di destra anche nel caso in cui non ci sia una relazione con la tabella di sinistra.

LEFT JOIN

Per avere tutti i movies presenti nel database, anche quelli senza relazione con la tabella categories, possiamo usare una LEFT JOIN con movies come tabella di sinistra:

```sql
SELECT *
FROM `movies`
LEFT JOIN `categories`
ON `categories`.`id` = `movies`.`category_id`;

```

Restituisce tutti i film (left),
con la rispettiva categoria associata (right) se presente,
altrimenti tutte le colonne relative alla category conterranno il valore NULL.

RIGHT JOIN

Per avere tutte le categories presenti nel database, anche quelli senza relazione con la tabella movies, possiamo usare una RIGHT JOIN con categories come tabella di destra:

```sql
SELECT *
FROM `movies`
RIGHT JOIN `categories`
ON `categories`.`id` = `movies`.`category_id`;
```

Restituisce tutte le categories (right),
con i rispettivi movies associati (left) se presenti,
altrimenti le colonne relative al movie conterranno il valore NULL.

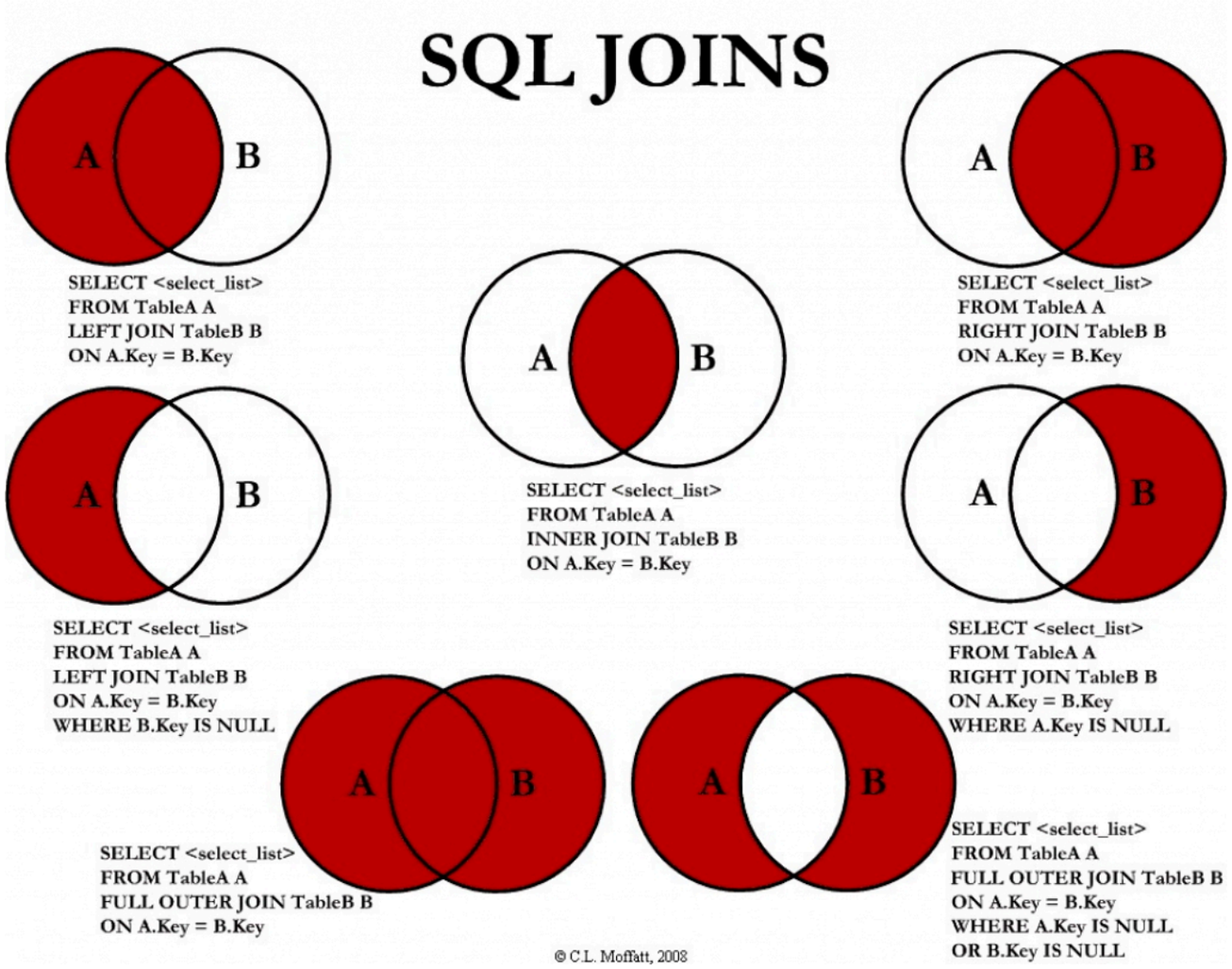
Trasformare RIGHT in LEFT JOIN

Spesso si tende ad utilizzare la left join, perché più di facile comprensione

Se, come nell'esempio precedente, volessimo avere tutte le categorie
a prescindere dal fatto che abbiano un film associato o meno,
potremmo girare la query partendo dalle categorie e utilizzare LEFT JOIN.

```sql
SELECT *
FROM `categories`
LEFT JOIN `movies`
ON `categories`.id` = `movies`.`category_id`;
```

Riepiloghiamo



LIVE CODING

ESERCITAZIONE