DATA LA COLONNA IN COMUNE, TIPO CUSTOMER\_id che parte da 1 a 1000, tipo nella tabella 1 manca id 3 e quindi tutte le altre relative colonne con quel dato, mentre nella tabella due c’è id 3 con relativi dati nelle colonne della tabella. E poi prendi id 578 che c’è a sx, e nelle relative colonne, ma manca a dx.

Se faccio inner join, allora né l’id 3 né l’id 578 compariranno, e quindi compariranno solo i valori della colonna customer id comuni, con le relative colonne afferenti in cui si hanno tutti i dati per riga per customer id.

Se faccio left join, io prendo l’id 3, di cui NON avrò i dati delle colonne della tabella 3, quindi verrà messo null.

---

PRIMARY KEY: se setto una colonna che deve avere primary key, significa che ogni valore deve essere unico. Tipo customer ID 1, nome e cognome in altre colonne, non si può rimettere l’ID 1 per un altro customer perché ogni ID deve essere unico.

---

SELECT author\_lname, COUNT(title) FROM books

GROUP BY author\_lname

Ti dice, raggruppando per author\_lname, raggruppa per autore, nel senso che l’autore ingloba in sé tutti i titoli che ha scritto in un’unica row. Senza il count, ti prende solo il primo titolo, mentre col count te li raggruppa e ti dà per ogni autore il numero di titoli scritti. Tipo per Gaiman, ci sono 3 libri raggruppati sotto al suo nome.

SELECT author\_fname, author\_lname, COUNT(title) FROM books

GROUP BY author\_fname, author\_lname

In questo caso, raggruppando sia per nome e per cognome, visto che il cognome può essere condiviso da più autori e si rischia per raggruppare per un solo cognome tutti i titoli e si attribuscono tipo per Gaiman anche titoli non suoi, in questo caso raggruppando per nome e cognome, li vede come unici e quindi attribuisce i titoli per quel nome e cognome unico, dandomi il numero di titoli scritti per autore.

---

SELECT title, author\_fname, author\_lname FROM books

WHERE pages = (SELECT MAX(pages) FROM books)

USANDO la subquery, cioè di base mettendo due select, SQL legge prima l’ultimo select con parentesi, quindi prima mi trova il libro con max pages, poi da questo gli dico, dammi titolo, nome e cognome dell’autore dal database books, dove le pagine sono uguali al numero massimo di pagine selezionate da books.

Se invece mettessi solo SELECT MAX(pages), title FROM books, mi dà la colonna max pages, che è 634, e poi come titolo NON quello corrispondente alla max pages, ma il primo che trova nella colonna titoli.

Un altro modo per darmi il titolo con max pages, è usare l’Order By.

SELECT title, author\_fname, author\_lname FROM books

ORDER BY pages DESC

LIMIT 1

Mi dà appunto le colonne titolo, nome e cognome da books, e ordina per le pagine dalle più alte alle più basse.

--

SELECT MIN(released\_year), author\_fname, author\_lname, title FROM books

GROUP BY author\_fname, author\_lname

ORDER BY released\_year

Per ogni nome e cognome unico di autore (ce lo dice il group by), dammi la colonna corrispondente del nome e cognome e l’anno minimo in cui hanno rilasciato un titolo, e che titolo è

--

SALTO LE LEZIONI SULLE DATE.

--

SELECT title, released\_year,

CASE

WHEN released\_year >= 2000 THEN 'Modern Lit'

ELSE '20th Century Lit'

END AS GENRE

FROM books

Date le colonne titoli e anno rilasciato da books, e categorizza le row mettendo come Modern lit i titoli e colonne afferenti superiore al 2000, altrimenti categorizzale come '20th Century Lit' nella colonna GENRE

SELECT title, stock\_quantity,

CASE

WHEN stock\_quantity BETWEEN 0 AND 50 THEN '\*'

WHEN stock\_quantity BETWEEN 51 AND 100 THEN '\*\*'

ELSE '\*\*\*'

END AS STOCK

FROM books

Crea la colonna stock, e ad ogni row mette gli asterischi a seconda delle quantità.

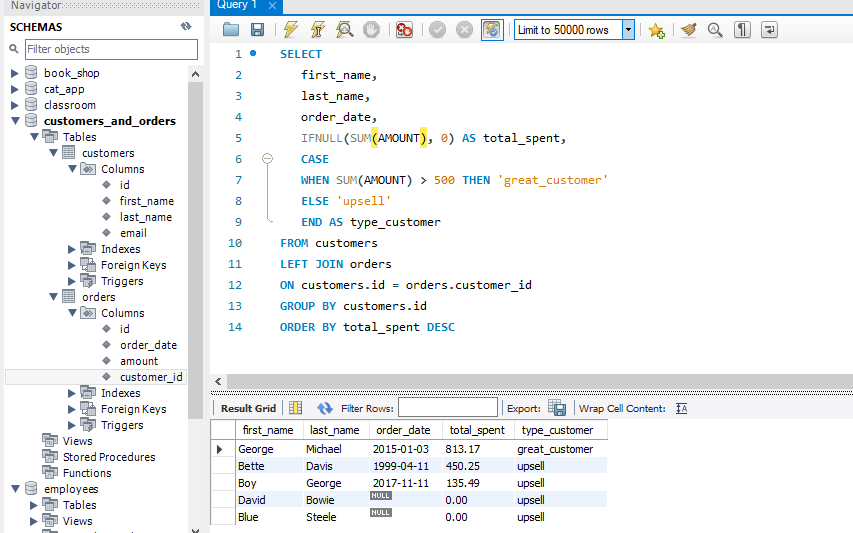
--

ALTRO ESEMPIO CASE

Immagine che contiene testo

Descrizione generata automaticamente

--



---

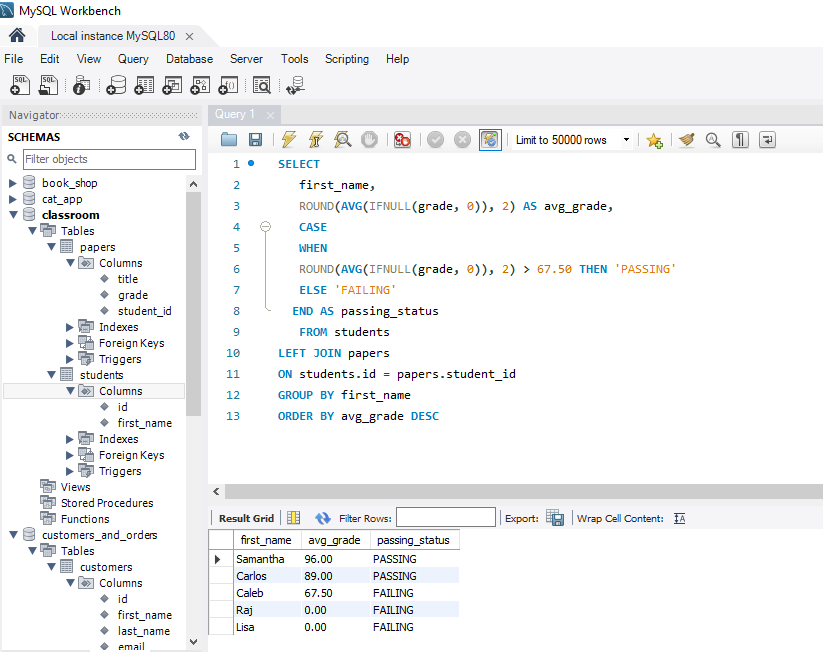


Immagine che contiene testo

Descrizione generata automaticamente

Allora abbiamo due tabelle. Faccio una left join tra la prima tabella series e la seconda tabella reviews sulla colonna che hanno in comune, ovvero series.id = reviews.series\_id.   
Essendo una left join, si prendono le colonne riferite alla prima tabella che si mettono nel select e di queste si prendono tutti i valori, tipo in questo caso si prendono tutti i titoli. La seconda colonna AVG(rating) mi fa una media riferita al primo titolo raggruppando per il titolo (che è unico).   
Dunque ho tutti i titoli, le row corrispondenti ai titoli sono quelli presenti in rating raggruppati come AVG; se non ci sono metti zero, cioè c’è un valore null. Poi il CASE, mette una terza colonna comment che dipende in questo caso in cosa succede in ogni row della seconda. Se L’AVG rating è > 7.5 metti must watch nella row, se è tra 5 e 7 metti enjoible, altrimenti metti can skip.

Più joins

Immagine che contiene testo

Descrizione generata automaticamente

--

Immagine che contiene testo

Descrizione generata automaticamente

Chi è l’utente che ha maggior like per foto?

Abbiamo due tabelle. Una likes, dove sono indicate di importanti le photo\_id in una colonna e il corrispondente user\_id. Mentre nella tabella photos, abbiamo l’id delle foto, l’image url della foto, entrambi unici, e l’user id. Quindi facendo una join su user id, ottengo, data la colonna comune user\_id, prendo tutti i valori delle row che stanno sia in photos che like, cioè quegli user\_id che stanno sia nella tabella photos sia nella tabella like, quindi ci sono gli stessi user che stanno sia in photos che like, quindi hanno sia fatto foto che avuto like, e volendo ottenere la foto che ha avuto più like, allora prendo l’image url, il photo\_id, faccio la somma di tutti gli user\_id dati per photo\_id raggruppando photo\_id, allora ottengo il conteggio di quante volte è stata collegata l’user\_id per la data (unica) photo\_id, ossia quanti like quindi ha avuto la photo\_id, con la collegata image\_url. Poi facendo un’altra JOIN, allora ottengo anche l’username di chi ha fatto quella determinata image\_url con determinata photo\_id.

---

Uso COALESCE()

Immagine che contiene testo

Descrizione generata automaticamente

Ha mezzo COALESCE una terza colonna, mettendo come row: se è presente la row della prima colonna, mette quella, altrimenti mette la seconda. Se metto invece COALESCE(dept\_no, dept\_name, ‘N/A’) allora come row vedrà se la colonna ha a le row corrispondenti, altrimenti mettere l’ultimo valore N/A

Altro esempio LEFT JOIN

Immagine che contiene testo

Descrizione generata automaticamente

Date due tabelle, ho fatto LEFT JOIN della prima tabella employees con la seconda tabella salaries, con i valori che hanno in comune nella colonna emp\_no.   
SI vede per esempio che l’emp\_no 10001 Georgi\_Facello ha avuto un salary di 30117 dal 1986-06-26 al 1987-06-25, poi ha avuto un salary pià alto dal 1986-06-26 al 1988-06-25 etc.   
Facendo una LEFT JOIN, quindi ho preso tutti i valori che stanno di emp\_no nella colonna emp\_no con annesse le row delle colonne annesse first\_name e last\_name. Ed essendoci gli emp\_no 10001 sia in employees che salaries, allora abbiamo anche le voci salary, from\_date e to\_date.

Immagine che contiene testo

Descrizione generata automaticamente

Essendo una LEFT JOIN; ho preso gli emp\_no tutte le row che stanno in emp\_no di employees, a cui sono associati nelle rispettive colonne di employees first\_name e last\_name. Evidenziando dove salary è NULL, questa salary è associata a un determinato emp\_no, per ex il 201772 che nella colonna emp\_no della colonna salary NON c’è. Dunque il comando LEFT\_JOIN mi restituisce il valore emp\_no 201772 perché c’è nella tabella employees, ma non essendoci nella tabella salaries allora SQL mi dà il valore NULL in corrispondenza.

Immagine che contiene testo

Descrizione generata automaticamente

Infatti si vede che nella tabella salaries nella colonna emp\_no NON c’è il valore 201772.