

ESEMPI DI CODICE IN MYSQL

1)

Composizione della tabella books:

The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'book_shop' database selected, with its tables listed: books, cat_app, classroom, customers_and_orders, employees, ig_clone, portfolio_1, sakila, sales, shirt_db, shirts_db, sys, and tv_review_app. The 'books' table is highlighted. The main editor shows the SQL query: `SELECT * FROM books`. Below the editor, the 'Result Grid' displays the data from the 'books' table. The table has 9 columns: book_id, title, author_fname, author_lname, released_year, stock_quantity, and pages. The data is as follows:

book_id	title	author_fname	author_lname	released_year	stock_quantity	pages
1	The Namesake	Jhumpa	Lahiri	2003	32	291
2	Norse Mythology	Neil	Gaiman	2016	43	304
3	American Gods	Neil	Gaiman	2001	12	465
4	Interpreter of Maladies	Jhumpa	Lahiri	1996	97	198
5	A Hologram for the King: A Novel	Dave	Eggers	2012	154	352
6	The Circle	Dave	Eggers	2013	26	504
7	The Amazing Adventures of Kavalier & Clay	Michael	Chabon	2000	68	634
8	Just Kids	Patti	Smith	2010	55	304
9	A Heartbreaking Work of Staggering Genius	Dave	Eggers	2001	104	437

Spiegazione codice SQL:

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 a quale titolo si riferisce.

SCHEMAS

Filter objects

book_shop

Tables

books

Views

Stored Procedures

Functions

cat_app

classroom

customers_and_orders

employees

ig_clone

portfolio_1

sakila

sales

shirt_db

shirts_db

sys

tv_review_app

Tables

reviewers

reviews

series

Views

Administration Schemas

Information

Limit to 50000 rows

1 SELECT MIN(released_year), author_fname, author_lname, title FROM books

2 GROUP BY author_fname, author_lname

3 ORDER BY released_year

4

Schema: book_shop

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	MIN(released_year)	author_fname	author_lname	title
▶	1945	John	Steinbeck	Cannery Row
	1981	Raymond	Carver	What We Talk About When We Talk About Love...
	1985	Don	DeLillo	White Noise
	2000	Michael	Chabon	The Amazing Adventures of Kavalier & Clay
	2001	Freida	Harris	fake_book
	1996	Jhumpa	Lahiri	The Namesake
	2004	David	Foster Wallace	Oblivion: Stories
	2010	Patti	Smith	Just Kids
	2001	Dave	Eggers	A Hologram for the King: A Novel

2)

Spiegazione codice SQL:

Date le colonne titoli e anno rilasciato dalla tabella books, crea una nuova colonna detta GENRE mettendo le righe come 'Modern lit' quei titoli che hanno l'anno in cui sono stati rilasciati maggiore o uguale 2000, altrimenti categorizza gli altri titoli come '20th Century Lit'.

The screenshot shows a database management interface. On the left, a 'SCHEMAS' pane displays a tree view of databases, with 'book_shop' expanded to show its tables: 'books', 'reviewers', 'reviews', and 'series'. The main area displays a SQL query:

```
1 • SELECT title, released_year,  
2 CASE  
3 WHEN released_year >= 2000 THEN 'Modern Lit'  
4 ELSE '20th Century Lit'  
5 END AS GENRE  
6 FROM books  
7
```

Below the query, a 'Result Grid' shows the output of the query. The grid has columns for 'title', 'released_year', and 'GENRE'. The results are as follows:

title	released_year	GENRE
The Namesake	2003	Modern Lit
Norse Mythology	2016	Modern Lit
American Gods	2001	Modern Lit
Interpreter of Maladies	1996	20th Century Lit
A Hologram for the King: A Novel	2012	Modern Lit
The Circle	2013	Modern Lit
The Amazing Adventures of Kavalier & Clay	2000	Modern Lit
Just Kids	2010	Modern Lit
A Heartbreaking Work of Staggering Genius	2001	Modern Lit

3)

Composizione delle tabelle series e reviews nel database tv_review_app.

SCHEMAS

Filter objects

- book_shop
 - Tables
 - books
 - Views
 - Stored Procedures
 - Functions
- cat_app
- classroom
- customers_and_orders
- employees
- ig_clone
- portfolio_1
- sakila
- sales
- shirt_db
- shirts_db
- sys
- tv_review_app**
 - Tables
 - reviewers
 - reviews
 - series
 - Views

Administration Schemas

Information

Schema: tv_review_app

Limit to 50000 rows

```

1 • SELECT * FROM series
2

```

Result Grid

	id	title	released_year	genre
▶	1	Archer	2009	Animation
	2	Arrested Development	2003	Comedy
	3	Bob's Burgers	2011	Animation
	4	Bojack Horseman	2014	Animation
	5	Breaking Bad	2008	Drama
	6	Curb Your Enthusiasm	2000	Comedy
	7	Fargo	2014	Drama
	8	Freaks and Geeks	1999	Comedy
	9	General Hospital	1963	Drama
	10	Golden Girls	2011	Drama

The screenshot shows a database management interface. On the left, the 'SCHEMAS' pane displays a tree view of databases. The 'tv_review_app' database is selected, showing its tables: 'reviewers', 'reviews', and 'series'. Below the tree, the 'Administration' tab is active, showing the 'Schema: tv_review_app'.

On the right, the SQL editor shows the following query:

```
1 • SELECT * FROM reviews
2
```

Below the SQL editor, the 'Result Grid' tab is active, displaying the results of the query. The results are shown in a table with the following columns: 'id', 'rating', 'series_id', and 'reviewer_id'.

	id	rating	series_id	reviewer_id
▶	1	8.0	1	1
	2	7.5	1	2
	3	8.5	1	3
	4	7.7	1	4
	5	8.9	1	5
	6	8.1	2	1
	7	6.0	2	4
	8	8.0	2	3
	9	8.4	2	6
	10	8.0	2	5

Spiegazione Codice SQL:

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, 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'.

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

Filter objects

- book_shop
- cat_app
- classroom
- customers_and_orders
- employees
 - Tables
 - Views
 - Stored Procedures
 - Functions
- ig_clone
- portfolio_1
- sales
- shirt_db
- shirts_db
- sys
- tv_review_app
 - Tables
 - reviewers
 - reviews
 - Columns
 - id
 - rating
 - series_id
 - reviewer_id
 - Indexes
 - Foreign Keys
 - Triggers
 - series
 - Views
 - Stored Procedures

Administration Schemas Information

Table: reviews

Columns:

- id int AI PK
- rating decimal(2,1)
- series_id int
- reviewer_id int

Query 1 x

Limit to 50000 rows

```
1 • SELECT
2     title,
3     ROUND(IFNULL(AVG(rating), 'N/A'), 2) AS avg,
4     CASE
5     WHEN
6     AVG(rating) > 7.50 THEN 'Must watch'
7     WHEN AVG(rating) BETWEEN 5.00 AND 7.50 THEN 'Enjoyble'
8     ELSE 'Can Skip'
9     END AS comment
10    FROM series
11   LEFT JOIN reviews
12   ON series.id = reviews.series_id
13   GROUP BY title
14   ORDER BY AVG(rating)
15
```

Result Grid

Filter Rows:

Export: | Wrap Cell Content: IA

	title	avg	comment
▶	Malcolm In The Middle	0	Can Skip
	Pushing Daisies	0	Can Skip
	General Hospital	5.38	Enjoyble
	Bob's Burgers	7.52	Must watch
	Seinfeld	7.6	Must watch
	Bojack Horseman	7.94	Must watch
	Arrested Development	8.08	Must watch
	Archer	8.12	Must watch
	Curb Your Enthusiasm	8.12	Must watch
	Freaks and Geeks	8.6	Must watch
	Stranger Things	8.77	Must watch
	Breaking Bad	9.36	Must watch
	Fargo	9.4	Must watch
	Halt and Catch Fire	9.9	Must watch

4)

Composizione delle tabelle employees e salaries nel Database employees:

The screenshot displays a database management interface. On the left, the 'SCHEMAS' pane shows a tree view of databases. The 'employees' database is selected, and its 'Tables' folder is expanded. The main query editor shows a SQL query: `SELECT * FROM employees`. Below the query editor, the 'Result Grid' displays the data from the 'employees' table. The grid has columns: emp_no, birth_date, first_name, last_name, gender, and hire_date. The data is as follows:

	emp_no	birth_date	first_name	last_name	gender	hire_date
▶	10001	1953-09-02	Georgi	Facello	M	1986-06-26
	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
	10003	1959-12-03	Parto	Bamford	M	1986-08-28
	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
	10006	1953-04-20	Anneke	Preusig	F	1989-06-02
	10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
	10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
	10009	1952-04-19	Sumant	Peac	F	1985-02-18
	10010	1952-06-01	Daniel	Phillips	F	1986-02-01

The screenshot displays a database management interface. On the left, the 'SCHEMAS' pane shows a tree view of databases, with 'employees' selected under the 'Tables' category. The main query editor shows the following SQL statement:

```
1 • SELECT * FROM salaries
2
```

Below the query editor, the 'Result Grid' shows the results of the query. The grid has five columns: 'emp_no', 'salary', 'from_date', and 'to_date'. The results are as follows:

emp_no	salary	from_date	to_date
10001	60117	1986-06-26	1987-06-26
10001	62102	1987-06-26	1988-06-25
10001	66074	1988-06-25	1989-06-25
10001	66596	1989-06-25	1990-06-25
10001	66961	1990-06-25	1991-06-25
10001	71046	1991-06-25	1992-06-24
10001	74333	1992-06-24	1993-06-24
10001	75286	1993-06-24	1994-06-24
10001	75994	1994-06-24	1995-06-24

Spiegazione Codice:

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.