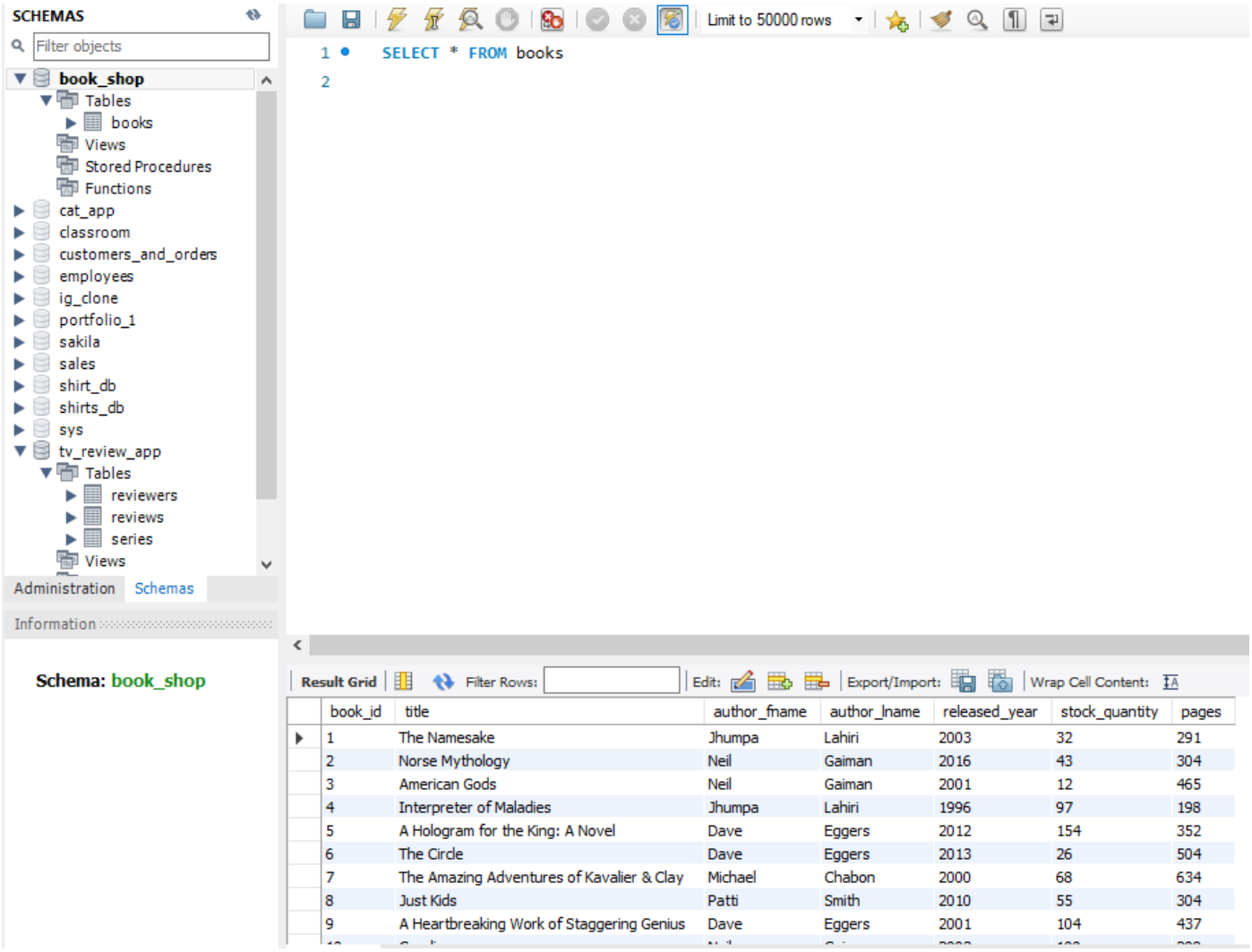
ESEMPI DI CODICE IN MYSQL

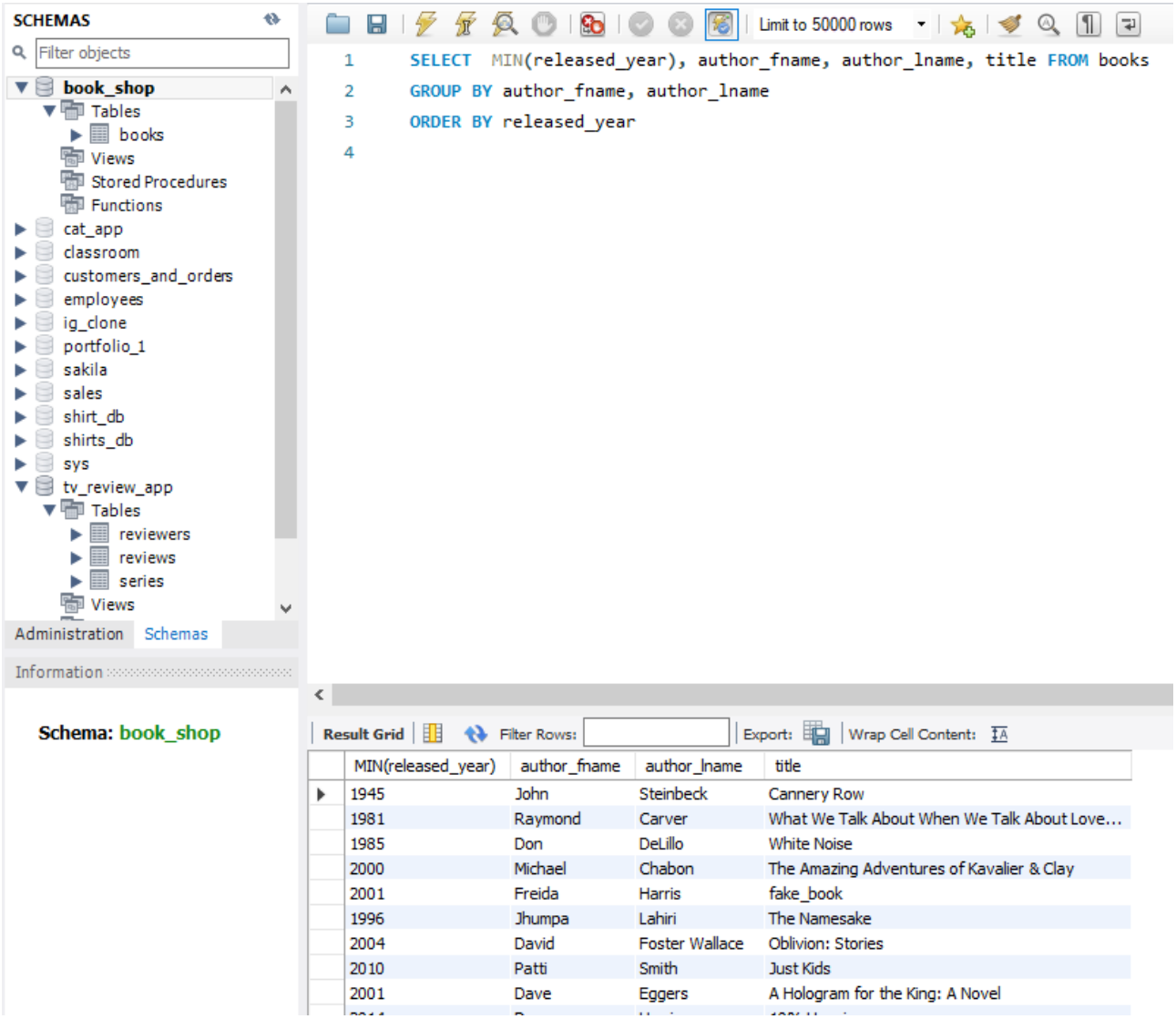
1)

Composizione della tabella books:



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

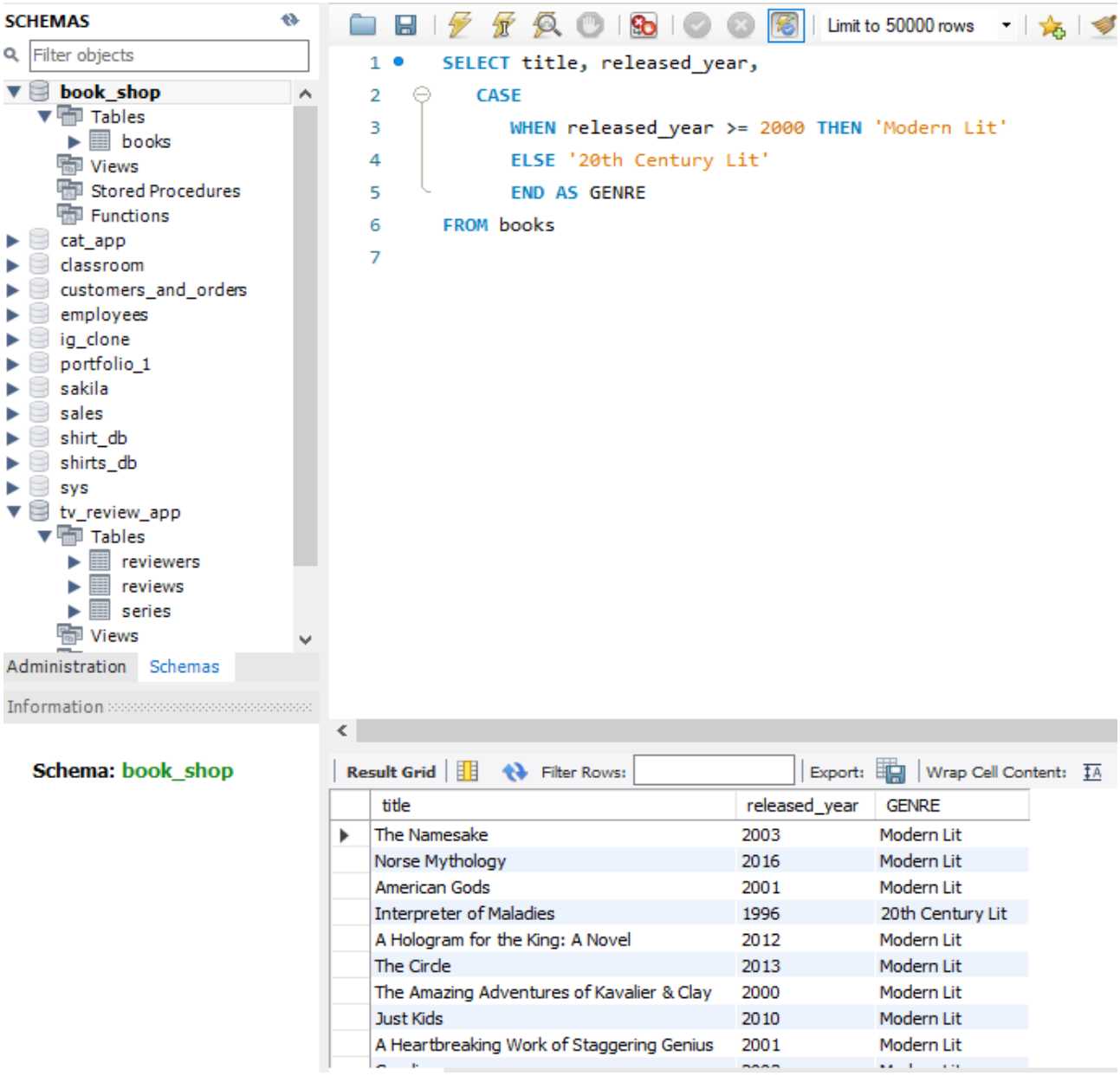
Spiegazione codice SQL:
Per ogni nome e cognome unico di autore (ce lo dice il group by), dammi la colonna corrispondente del nome e cognome e l'anno minimo in cui hanno rilasciato un titolo, e a quale titolo si riferisce.


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
Schema: book_shop
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
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)

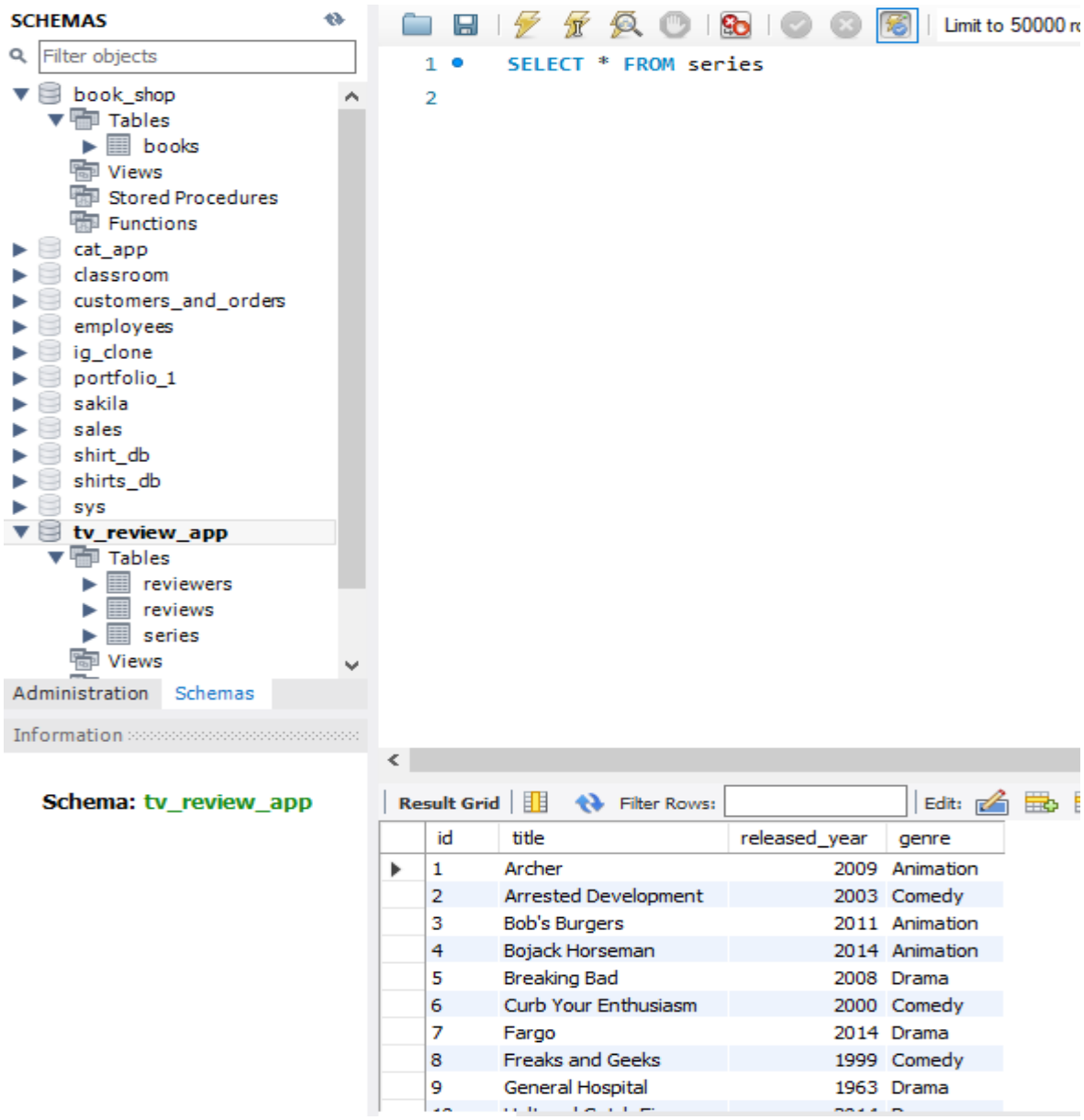
Spiegazione codice SQL:

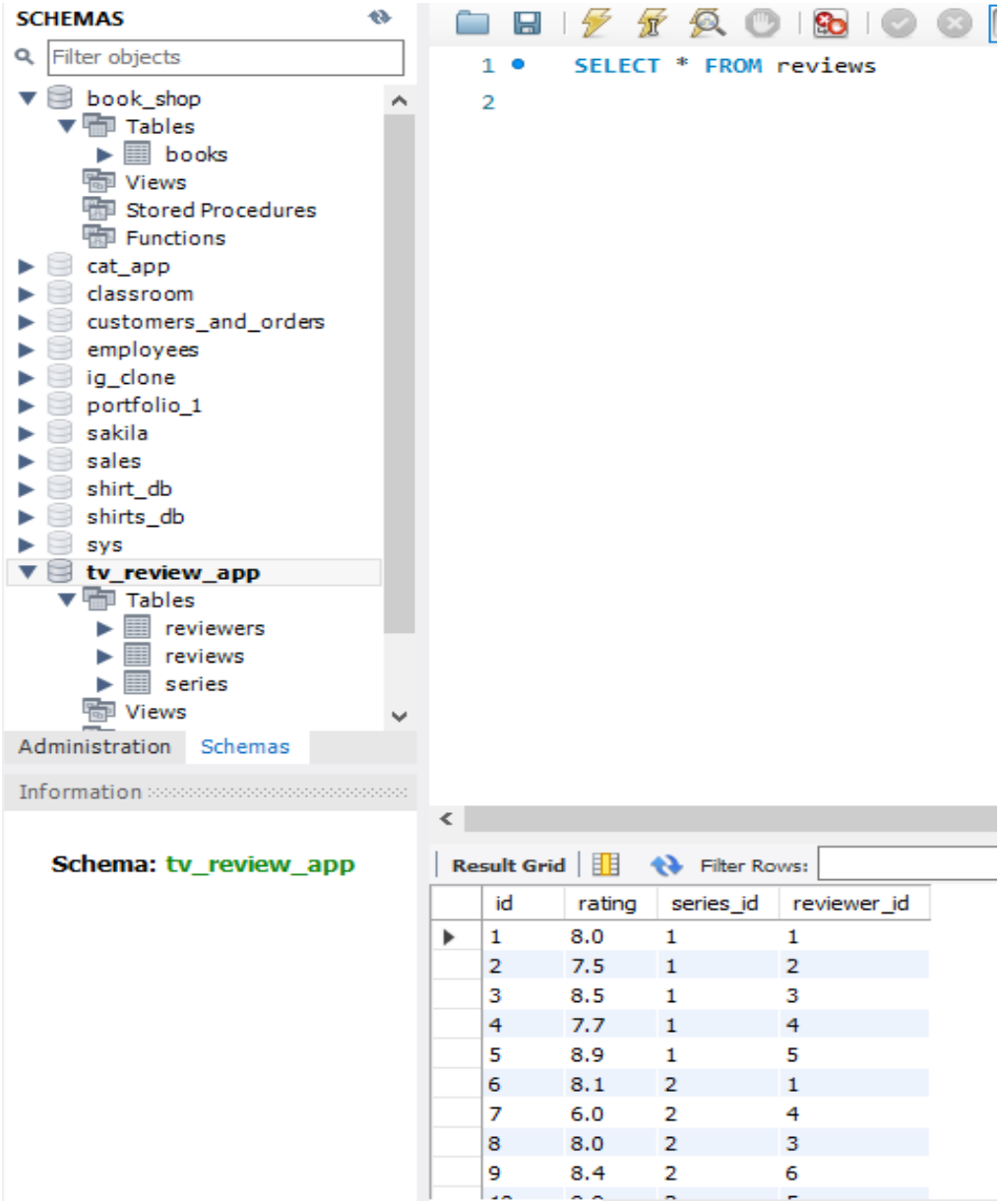
Date le colonne titoli e anno rilasciato dalla tabella books, crea una nuova colonna detta GENRE mettendo le righe come ‘Modern lit’ quei titoli che hanno l’anno in cui sono stati rilasciati maggiore o uguale 2000, altrimenti categorizza gli altri titoli come '20th Century Lit'.



3)

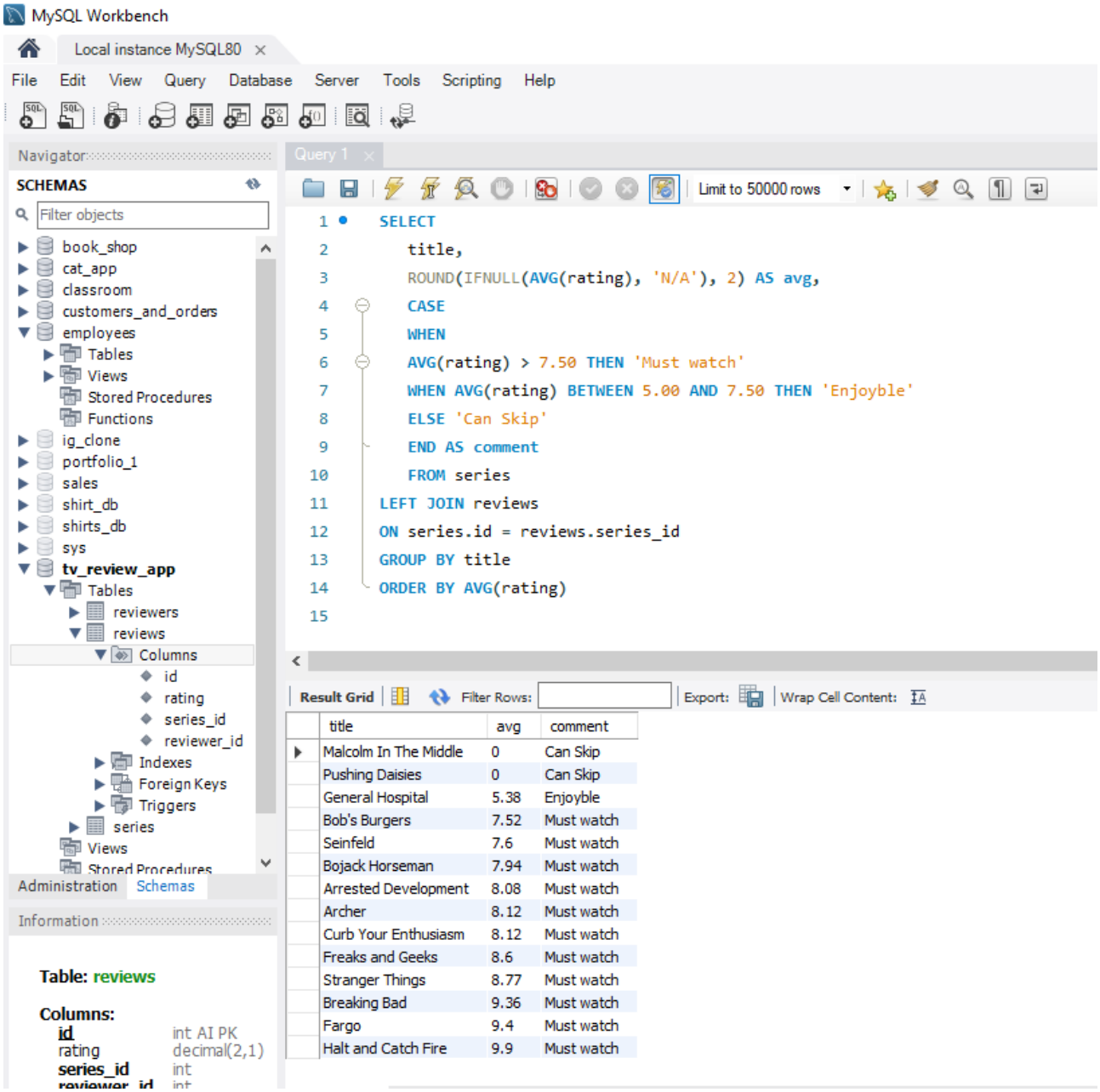
Composizione delle tabelle series e reviews nel database tv_review_app.


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 r
1 SELECT * FROM series
2
Result Grid
Filter Rows:
Edit:
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



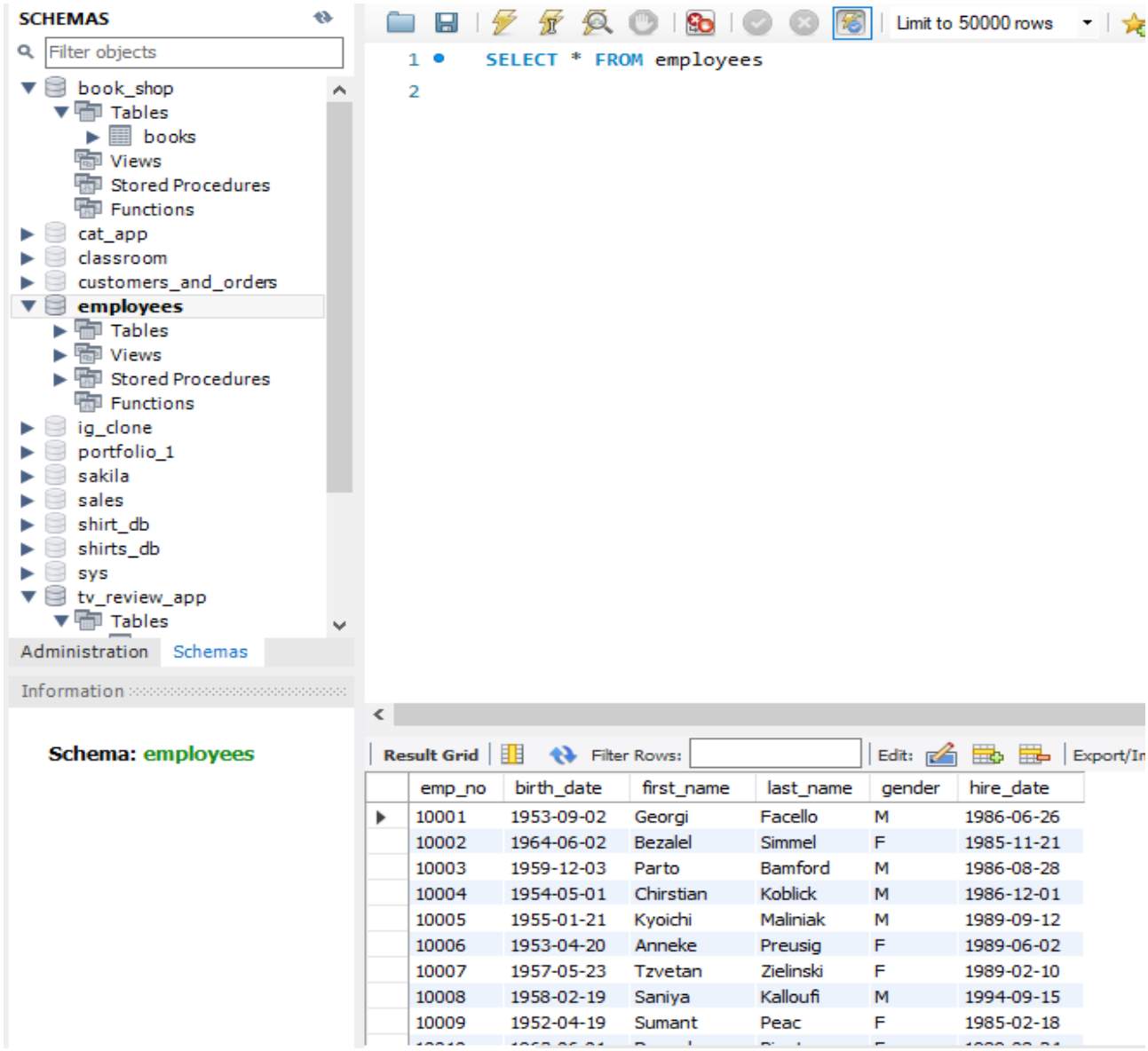
Spiegazione Codice SQL:

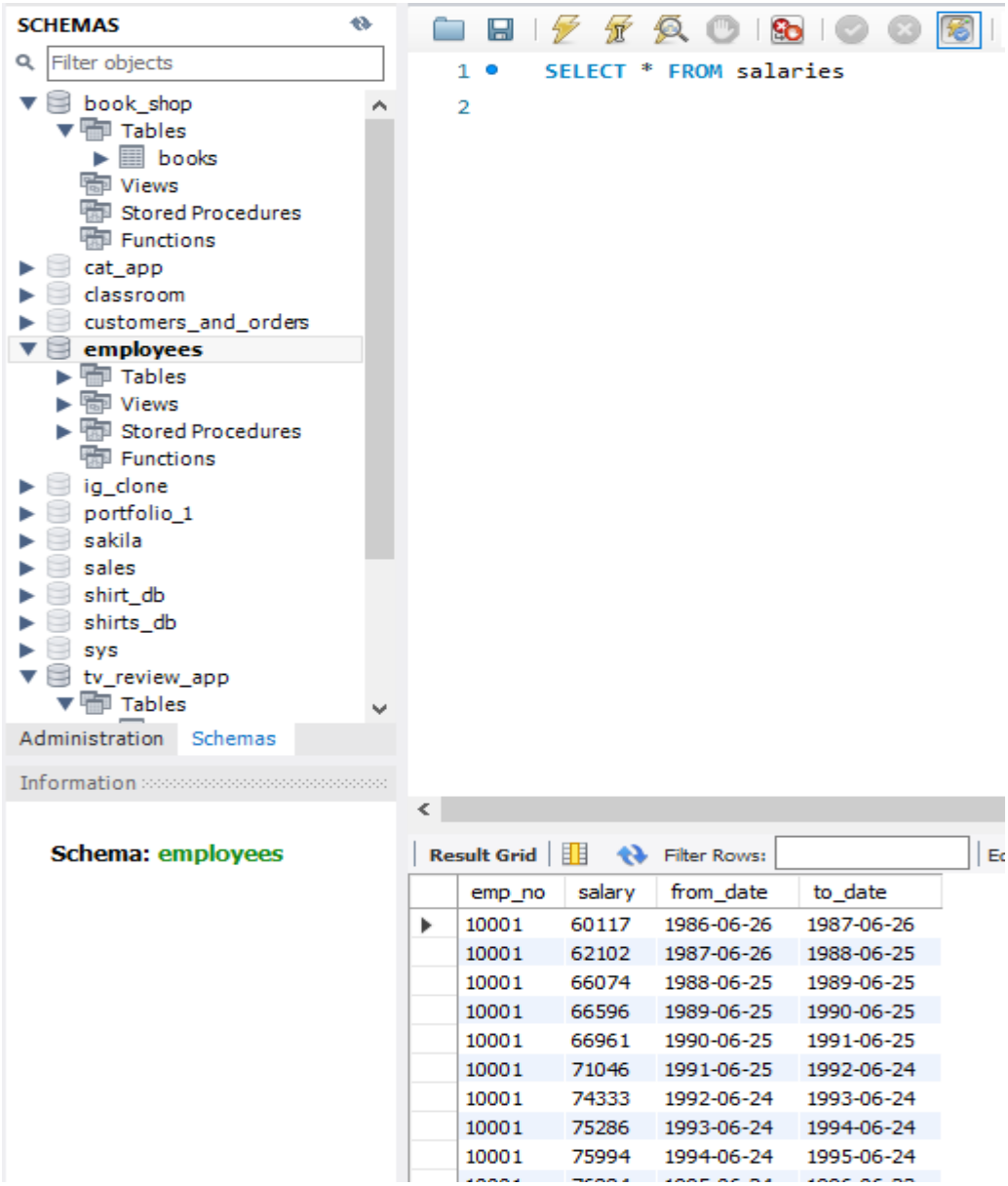
Abbiamo due tabelle. Faccio una left join tra la prima tabella series e la seconda tabella reviews sulla colonna che hanno in comune, ovvero series.id = reviews.series_id.
Essendo una left join, si prendono le colonne riferite alla prima tabella che si mettono nel select e di queste si prendono tutti 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'.


MySQL Workbench
Local instance MySQL80
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
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:
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:





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.