**SQL Project**

**Dataset:-** [Books.csv](https://drive.google.com/file/d/11LBNkYb-zso0DjWb7ELZMczfP4f8K18o/view?usp=sharing)

* **Create a new database named 'db1'**

create schema db1;

* **Import dataset from import wizard and then select all records from the 'books' table.**

select \* from books;

* **Switch to using database 'db1'**

use db1;

* **Check for duplicate entries based on the 'uniq\_id' column**

select uniq\_id, count(\*)

from books

group by uniq\_id

having count(\*) > 1;

* **Disable safe updates to allow deleting without using a key in WHERE clause**

set sql\_safe\_updates = 0;

* **Use a Common Table Expression (CTE) to assign a row number to each row partitioned by 'uniq\_id'**
* **Delete duplicate rows, keeping only the first occurrence (rn = 1) and Rows with rn > 1 are duplicates.**

with ranked as (

select \*,

row\_number() over (partition by uniq\_id order by uniq\_id) as rn

from books

)

delete from books

where uniq\_id in (

select uniq\_id from ranked where rn > 1

);

* **Change data type of 'uniq\_id' column to varchar(255)**

alter table books

modify uniq\_id varchar(255);

* **Add a primary key constraint on 'uniq\_id' column**

alter table books

add primary key (uniq\_id);

* **Modify column data types to decimal with precision (10,2) for price-related fields**

alter table books

modify column selling\_price decimal(10,2);

alter table books

modify column list\_price decimal(10,2);

alter table books

modify column amtsave decimal(10,2);

* **Normalize data by adding computed (generated) columns based on existing data in 'breadcrumbs' and 'desc'**

alter table books

add column genre varchar(100)

generated always as (

trim(substring\_index(substring\_index(breadcrumbs, '|', 2), '|', -1))

) stored,

add column subgenre varchar(200)

generated always as (

trim(substring\_index(substring\_index(breadcrumbs, '|', 3), '|', -1))

) stored,

add column authoredby varchar(100)

generated always as (

case

when `desc` like '%"authored by"=>"%'

then trim(substring\_index(substring\_index(`desc`, '"authored by"=>"', -1), '"', 1))

else 'not provided'

end

) stored,

add column language varchar(50)

generated always as (

case

when `desc` like '%"language"=>"%'

then trim(substring\_index(substring\_index(`desc`, '"language"=>"', -1), '"', 1))

else 'not provided'

end

) stored,

add column publicationyear varchar(50)

generated always as (

case

when `desc` like '%"publication year"=>"%'

then trim(substring\_index(substring\_index(`desc`, '"publication year"=>"', -1), '"', 1))

else 'not provided'

end

) stored,

add column publishername varchar(150)

generated always as (

case

when `desc` like '%"publisher name"=>"%'

then trim(substring\_index(substring\_index(`desc`, '"publisher name"=>"', -1), '"', 1))

else 'not provided'

end

) stored;

* **Add generated column 'binding' extracted from 'specifications' column**

alter table books

add column binding varchar(100)

generated always as (

case

when specifications like '%"binding"=>"%'

then trim(substring\_index(substring\_index(specifications, '"binding"=>"', -1), '"', 1))

else 'not provided'

end

) stored;

* **Set genre to 'others' if it is not in the predefined list**

update books

set genre = 'others'

where genre not in (

'academic & professional',

'biographies & auto biographies',

'business & management',

'children & teens',

'literature & fiction',

'non fiction',

'regional books',

'religion & spirituality',

'self help'

);

* **Create a view 'books\_view' with selected useful columns from 'books' table**
* **Also includes a LEFT JOIN on itself based on 'uniq\_id' and 'model' (this join looks suspicious)**

create view books\_view as

select

b1.no as no,

b1.model as model,

b1.book\_name as bookname,

b1.uniq\_id as uniqueid,

b1.language as language,

b1.brand as brand,

b1.authoredby as author,

b1.publicationyear as publicationyear,

b1.genre as genre,

b1.subgenre as subgenre,

b1.publishername as publishername,

b1.binding as binding,

b1.weight as weight,

b1.selling\_price as sellingprice,

b1.productcode as productcode,

b1.list\_price as listprice,

b1.selling\_date as sellingdate,

b1.amtsave as amtsave

from books b1

left join books b on b.uniq\_id = b1.model;

**Questions**

**Q-1 For each genre, list the top 3 most discounted books, including the rank, and also show the previous book’s saving and next book’s saving within the same genre.**

**Input:-**

with main as (

select bookname,author,genre,amtsave,

rank() over (partition by genre order by amtsave) as ranks,

Lag(amtsave) over (partition by genre order by amtsave) as previous\_books,

Lead(amtsave) over (partition by genre order by amtsave) as next\_books

from books\_view

)

select \* from main

where ranks <=3;

**Output:-**

****

**Q-2 For each genre, we want to find the one book where the discount (amtsave) suddenly increased the most compared to the previous book (when books are sorted by their discount amounts).**

**Input:-**

with main as (

select author,genre,amtsave,

rank() over(partition by genre order by amtsave) as ranks,

Lag(amtsave) over (partition by genre order by amtsave) as previous\_books

from books\_view

),

differance as (

select \* ,(amtsave - previous\_books) as discount

from main

),

max as (

select \*,rank() over(partition by genre order by discount desc) as dis\_rank

from differance

)

select genre,author,amtsave,previous\_books,discount

from max

where dis\_rank=1;

**Output:-**



**Q-3 Which books and genre have a discount greater than 50%, and what are their corresponding list and selling prices?**

select book\_name,genre,list\_price, selling\_price, round(((list\_price - selling\_price) / list\_price) \* 100,2) AS dis\_percent

from books

where ((list\_price - selling\_price) / list\_price) \* 100 > 50;

**Output:-**



**Q-4 Please identify the top 2 most discounted books in each genre — but only include genres where we have at least 5 books listed.**

**Input:-**

with books\_genre as (

select genre

from books\_view

group by genre

having count(\*)>=5

),

rnk\_book as (

select \*,

rank() over(partition by genre order by amtsave desc) as rnk

from books\_view

where genre in (select \* from books\_genre)

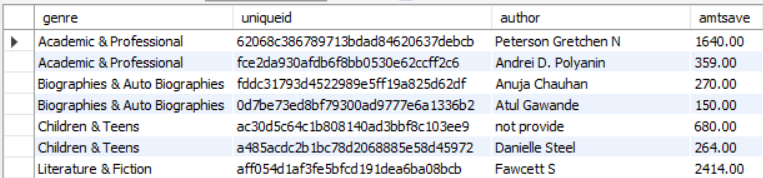
)

select genre,uniqueid,author,amtsave

from rnk\_book

where rnk <=2;

**Output:-**



**Q-5 Identify the top 2 most discounted books per genre — but only for genres where the average selling price is over ₹250 and the total number of books.**

**Input:-**

with genre\_filter as (

select genre

from books\_view

group by genre

having avg(sellingprice) > 250

),

ranked\_books as (

select \*,

rank() over(partition by genre order by amtsave desc) as rank\_in\_genre

from books\_view

where genre in (select genre from genre\_filter)

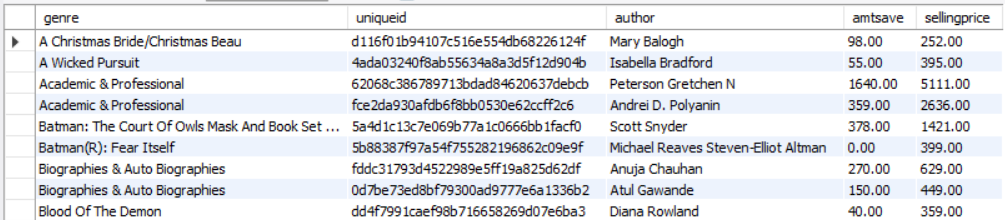
)

select genre,uniqueid,author,amtsave,sellingprice

from ranked\_books

where rank\_in\_genre <= 2;

**Output:-**



**Q-6 Given a starting book (by its uniq\_id), find all books that are related to it by model — including newer versions, reprints, or variants that are linked by model ID.**

**Input:-**

select \*

from books\_view

where model = (

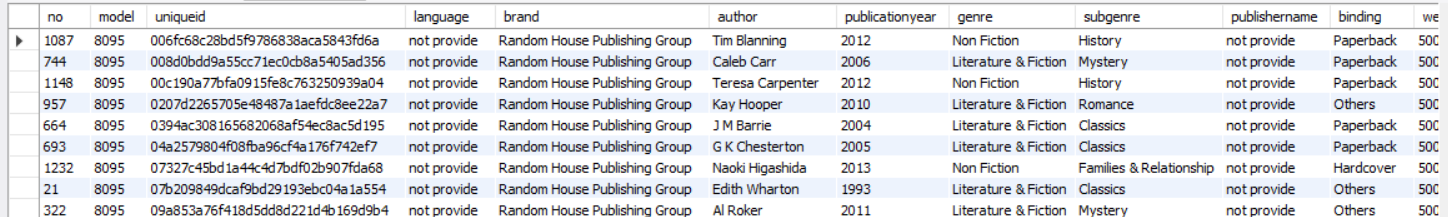
select model

from books\_view

where uniqueid = '006fc68c28bd5f9786838aca5843fd6a'

);

**Output:-**

****

**Q-7 We need to identify the full edition trail (linked by model and uniq\_id) of a book, starting from the earliest version — and highlight the most recent one with the highest selling price.**

**Input:-**

with same\_books as (

select \* from books\_view

where model = (

select model

from books\_view

where uniqueid = '006fc68c28bd5f9786838aca5843fd6a'

)

),

high\_price as (

select \*,

rank() over(order by sellingprice desc) as ranks

from same\_books

)

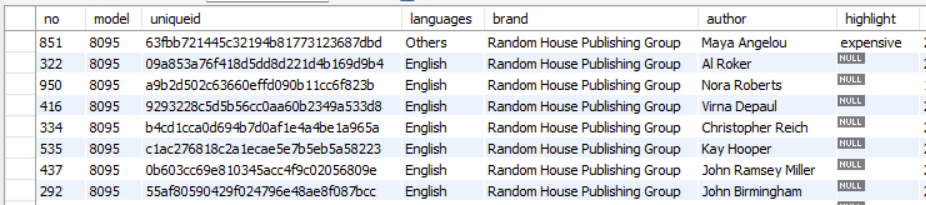
select \*,

case when ranks = 1 then 'expensive' else null end as Highlighte

from high\_price

order by sellingdate;

**Output:-**

****

**Q-8 Find the top 2 genres that have sold the most in total revenue in the past 4 months.**

**Input:-**

select genre, sum(sellingprice) as total\_revenue

from books\_view

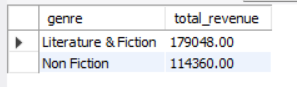
where sellingdate >= '2016-04-30'

group by genre

order by total\_revenue desc

limit 2;

**Output:-**



**Q-9 We want to find authors who sold at least 3 books in the last 3 months, with an discount (amtsave) of ₹100.**

**Input:-**

with recent\_books as (

select genre,author

from books\_view

where sellingdate >= date\_sub('2016-06-30', interval 3 month)

and amtsave >= 100

)

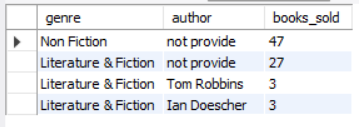
select genre,author, count(\*) as books\_sold

from recent\_books

group by genre,author

having count(\*) >= 3;

**Output:-**

****

**Q-10 Which were the last 10 days when books were sold, and how many books were sold on each of those days?**

**Input:-**

select sellingdate, count(\*) as num\_sales

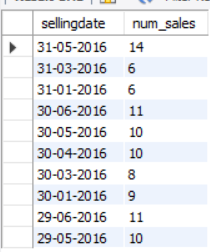
from books\_view

group by sellingdate

order by sellingdate desc

limit 10;

**Output:-**

****

**Q-11 So, for each publisher, find the single book in the last 5 months that gave customers the highest absolute savings.**

-- Include the book’s author, genre, and price details.

**Input:-**

with intervel as

(

select author,genre,publishername,sellingprice,amtsave,sellingdate

from books\_view

where sellingdate >= date\_sub('2016-10-15', interval 5 month)

),

ranking as (

select \*,

rank() over(partition by publishername order by amtsave desc) as ranks

from intervel

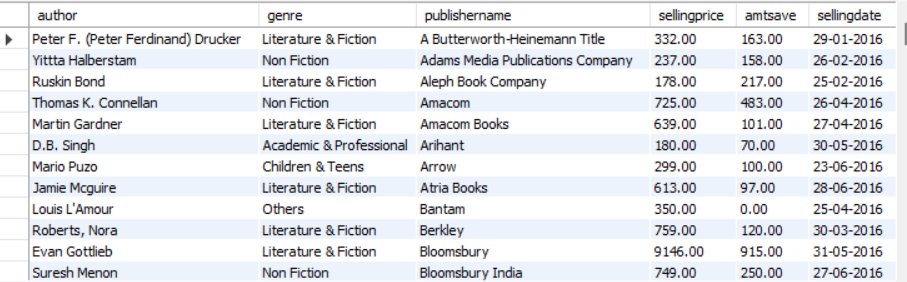
)

select author,genre,publishername,sellingprice,amtsave,sellingdate

from ranking

where ranks=1;

**Output:-**



**Q-12 When was the last time books were sold, and how many books were sold on that day?**

**Input:-**

select sellingdate, count(\*) as num\_sales

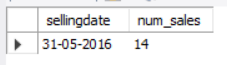
from books\_view

group by sellingdate

order by sellingdate desc

limit 1;

**Output:-**

****

**Q-13 Find all books where the selling price is higher than the list price. Show book name, list price, selling price, and publisher.**

**Input:-**

select book\_name,selling\_price,genre,publishername

from books

order by selling\_price desc

limit 10;

**Output:-**



**Q-14 For each genre, which book binding type (e.g., paperback, hardcover, spiral) was the most popular based on the number of books sold?**

**Input:-**

select \* from books\_view;

with preferances as (

select genre,binding

from books\_view

where sellingdate >= date\_sub('2016-06-30',interval 45 day)

),

ranking as (

select genre,binding,count(\*) as total\_sold,

rank() over(partition by genre order by count(\*) desc) as ranks

from preferances

group by genre,binding

)

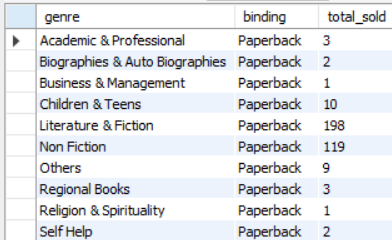
select genre,binding,total\_sold

from ranking

where ranks = 1;

select \* from books\_view;

**Output:-**

****

**Q-15 Which genres have been giving high average discounts over the last 60 days but still sold fewer than 10 books total?**

**Input:-**

with avg\_discounts as (

select genre,round(avg(amtsave),2) as aveage\_discount,count(\*) as total\_books

from books\_view

where sellingprice >= date\_sub('2016-06-30',interval 60 day)

group by genre

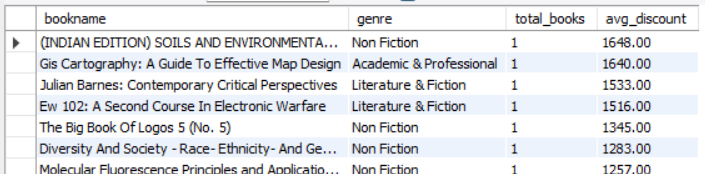
having count(\*) > 10

)

select genre,total\_books,aveage\_discount

from avg\_discounts;

**Output:-**



* **Q-16 write a store procedure to get books by brand that takes a brand name as a input parameter and return all books like unique id,author, selling price,list price,genre,subgenre,etc. for that brand.**

**Input:-**

DELIMITER //

create procedure `book\_details`(in brand\_name varchar(255))

begin

select model,bookname,brand,author,genre,sellingprice from books\_view

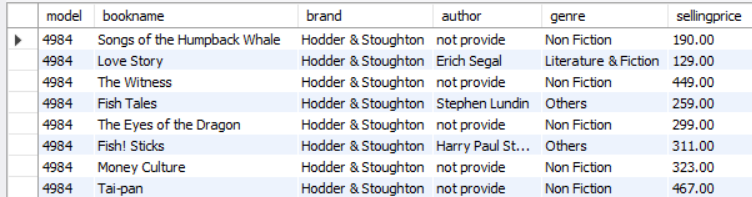
where brand = brand\_name;

end//

DELIMITER ;

call `book\_details`('Hodder & Stoughton');

**Output:-**



* **Q-17 Create a stored procedure Get Books With Missing Brand that returns all products where the brand field is NULL.**

**Input:-**

DELIMITER //

create procedure missing\_brand()

begin

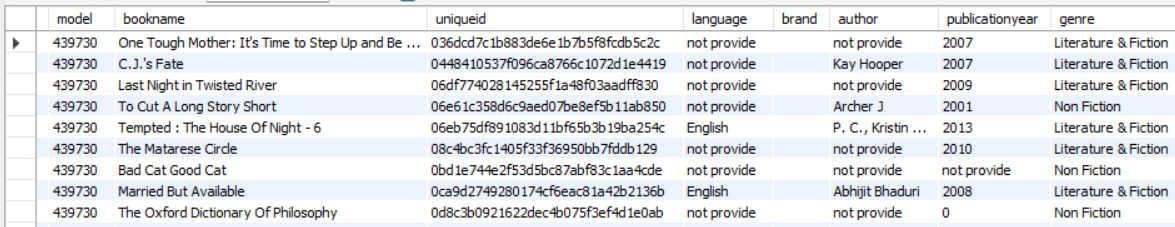
select \* from books\_view

where brand is null or brand='';

end //

DELIMITER ;

**Output:-**

****

* **Q-18 Write a stored procedure Delete Books By ProductCode that deletes a product from the dataset based on the productcode provided as an input parameter.**

**Input:-**

DELIMITER //

create procedure delete\_books(in p\_code int)

begin

delete from books

where productcode = p\_code;

end//

DELIMITER ;

call p\_code(12299410);

**Output:-**

**Before**

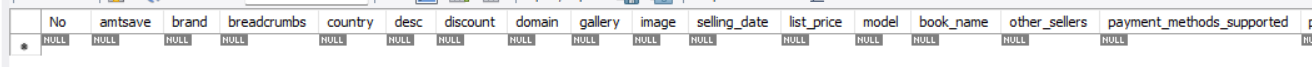
select \* from books

where productcode = 12299410;



**After**

**Call the procedure**



* **Q-19 Create a stored procedure Get Average SellingPrice By Brand that returns the average selling\_price of products for a given brand.**

**Input:-**

DELIMITER //

create procedure `avg\_selling\_price`(in b\_name varchar (200))

begin SELECT bookname,brand,round(AVG(sellingprice),2) AS average\_selling\_price

from books\_view

where b\_name = brand

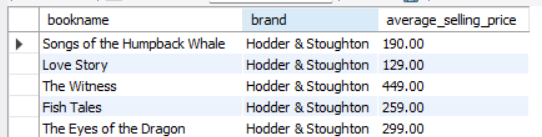
group by bookname,brand;

end //

DELIMITER ;

call `avg\_selling\_price`('Hodder & Stoughton');

**Output:-**

****

* **Q-20 Create a stored procedure GetBooksByPublicationYear that accepts a year as input and returns all books published in that year.**

**Input:-**

DELIMITER //

create procedure publish\_year(in p\_year int)

begin

select \* from books\_view

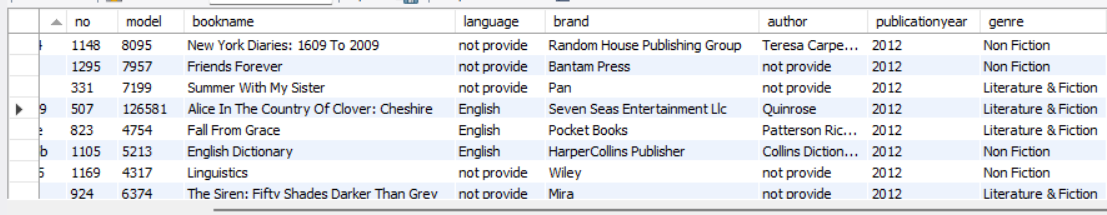
where publicationyear = p\_year;

end //

DELIMITER ;

call publish\_year(2012);

**Output:-**



* **Summary**

🡪 I cleaned up the book data and organized it to get useful insights. By analyzing the data, I found which books in each genre have the biggest discounts and where the discount suddenly changed, helping me understand pricing strategies.

🡪 I also looked at books with over 50% discount and showed their original and selling prices, while tracking different editions to see which versions are the most valuable.

🡪 Next, I identified the genres that earned the most money in the last 4 months and discovered authors who sold many discounted books recently. I also analyzed recent sales to understand overall activity and trends.

🡪 I highlighted the biggest customer savings from each publisher in the last 5 months and looked at which genres offer the best average discounts and the most popular book binding types (like paperback or hardcover).

🡪 I spotted some genres that gave big discounts but sold very few books, which could mean the discount strategy isn’t working as expected.

* **Conclusion**

🡪 I cleaned and organized the book data to understand sales, discounts, and customer choices.

🡪 The top earning genres in the last 4 months were Literature & Fiction with 179,048 and Non Fiction with 114,360.

🡪 In Literature & Fiction, authors like Ian Doescher and Tom Robbins sold at least 3 books with a discount of ₹100. Non Fiction was not considered because most books didn’t have author names.

🡪 The genres with discounts over 50% were Literature & Fiction, Non Fiction, and Children & Teens, with their list and selling prices noted.

🡪 The most recent book sold was "(INDIAN EDITION) SOILS AND ENVIRONMENTAL QUALITY 3RD EDITION 3 Rev ed Edition" on 31-05-2016, with 1 copy sold.

🡪 Customers preferred Paperback as the most popular binding in the last 45 days.

🡪 Literature & Fiction is the strongest genre, performing well in revenue, active authors, discounts, and customer choice. Some genres give big discounts but don’t sell much, which means the discount strategy might need rethinking.