

Task 1

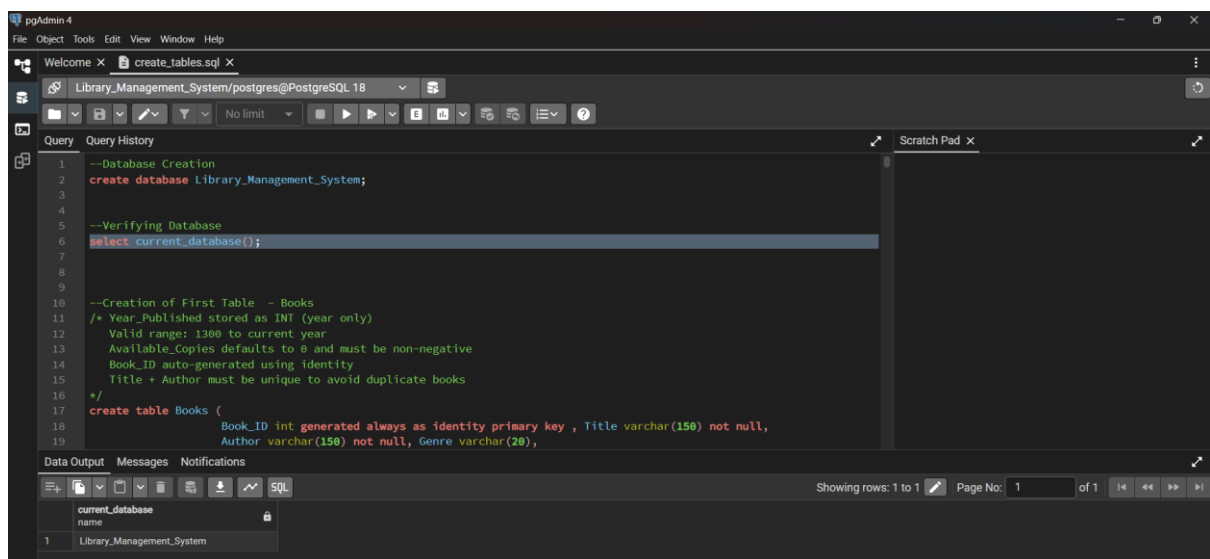
Project Title: Library Management System (using SQL)

Project Description:

Design and develop a Library Management System using SQL. The project should involve three tables: Books, Members, BorrowingRecords. The system will manage book inventories, member details, and borrowing transactions.

GITHUB: https://github.com/xrahulcrx/Library_Management_System

Database created



```
--Database Creation
create database Library_Management_System;

--Verifying Database
select current_database();

--Creation of First Table - Books
/* Year_Published stored as INT (year only)
   Valid range: 1300 to current year
   Available_Copies defaults to 0 and must be non-negative
   Book_ID auto-generated using identity
   Title + Author must be unique to avoid duplicate books
*/
create table Books (
    Book_ID int generated always as identity primary key , Title varchar(150) not null,
    Author varchar(150) not null, Genre varchar(20),
```

Data Output

current_database
Library_Management_System

Tables created



```
Library_Management_System=# \d Books
Table "public.books"
Column          | Type          | Collation | Nullable | Default
book_id         | integer       |           | not null | generated always as identity
title           | character varying(150) |           | not null |
author          | character varying(150) |           | not null |
genre           | character varying(20)  |           |          |
year_published  | integer       |           |          |
available_copies | integer       |           |          | 0
Indexes:
    "books_pkey" PRIMARY KEY, btree (book_id)
    "books_title_author_key" UNIQUE CONSTRAINT, btree (title, author)
Check constraints:
    "books_available_copies_check" CHECK (available_copies >= 0)
    "books_year_published_check" CHECK (year_published >= 1300 AND year_published::numeric <= EXTRACT(year FROM CURRENT_DATE))
Referenced by:
    TABLE "borrowingrecords" CONSTRAINT "borrowingrecords_book_id_fkey" FOREIGN KEY (book_id) REFERENCES books(book_id)

Library_Management_System=# \d Members
Table "public.members"
Column          | Type          | Collation | Nullable | Default
member_id       | integer       |           | not null | generated always as identity
name            | character varying(50) |           | not null |
email           | character varying(50) |           | not null |
phone_no        | bigint        |           |          |
address         | character varying(150) |           |          |
membership_date | date          |           |          | CURRENT_DATE
Indexes:
    "members_pkey" PRIMARY KEY, btree (member_id)
    "members_email_key" UNIQUE CONSTRAINT, btree (email)
Referenced by:
    TABLE "borrowingrecords" CONSTRAINT "borrowingrecords_member_id_fkey" FOREIGN KEY (member_id) REFERENCES members(member_id)
```

Library_Management_System=# \d BorrowingRecords

Column	Type	Collation	Nullable	Default
borrow_id	integer		not null	generated always as identity
member_id	integer		not null	
book_id	integer		not null	
borrow_date	date			CURRENT_DATE
return_date	date			

Indexes:
 "borrowingrecords_pkey" PRIMARY KEY, btree (borrow_id)

Check constraints:
 "chk_return_date" CHECK (return_date IS NULL OR return_date >= borrow_date)

Foreign-key constraints:
 "borrowingrecords_book_id_fkey" FOREIGN KEY (book_id) REFERENCES books(book_id)
 "borrowingrecords_member_id_fkey" FOREIGN KEY (member_id) REFERENCES members(member_id)

Information Retrieval:

a) Retrieve a list of books currently borrowed by a specific member.

Library_Management_System/postgres@PostgreSQL 18

Query Query History

```
-- a) Retrieve a list of books currently borrowed by a specific member.

select br.borrow_id, b.title, b.author, br.borrow_date
from BorrowingRecords br
join Books b on br.book_id = b.book_id
where br.member_id = 3 and br.return_date is null
order by borrow_date desc;
```

Data Output Messages Notifications

Showing rows: 1 to 3 Page No: 1 of 1

borrow_id	title	author	borrow_date
6	The Great Gatsby	F. Scott Fitzgerald	2025-11-05
28	1984	George Orwell	2025-07-20
7	The Lord of the Rings	J.R.R. Tolkien	2025-07-01

b) Find members who have overdue books (borrowed more than 30 days ago, not returned).

Library_Management_System/postgres@PostgreSQL 18

Query Query History

```
select m.member_id, m.name, br.borrow_date, (current_date - br.borrow_date) as No_of_days
from borrowingrecords br
join members m on br.member_id = m.member_id
where return_date is null and (current_date - br.borrow_date) > 30
order by no_of_days desc;
```

Data Output Messages Notifications

Showing rows: 1 to 15 Page No: 1 of 1

member_id	name	borrow_date	no_of_days
3	John Mathew	2025-07-01	151
3	John Mathew	2025-07-20	132
4	Priya Singh	2025-08-01	120
16	Shruti Mehta	2025-08-02	119
1	Rahul Sharma	2025-08-10	111
16	Shruti Mehta	2025-08-15	106
2	Anjali Verma	2025-09-01	89
13	Rohit Jain	2025-09-01	89
7	Aman Gupta	2025-09-05	85
10	Lakshmi Nair	2025-09-10	80
14	Aishwarya Reddy	2025-10-01	59
18	Sonali Pillai	2025-10-05	55
14	Aishwarya Reddy	2025-10-12	48
9	Saurabh Das	2025-10-18	42
8	Kavya Rao	2025-10-21	39

Total rows: 15 Query complete 00:00:00.065 CRLF Ln 226, Col 1

c) Retrieve books by genre along with the count of available copies.

The screenshot shows a PostgreSQL query editor with the following SQL query:

```
--v1 retrieves the genre and with no. of books
select genre, count(*) as No_of_Books, sum(available_copies) as Total_copies
from books
group by genre
order by Total_copies desc, genre;
```

The Data Output tab shows the following results:

genre	no_of_books	total_copies
Fantasy	3	23
Dystopian	2	12
Self-Help	1	12
Finance	1	10
Romance	2	10
Drama	1	8
Fiction	2	8
Philosophy	1	7
History	1	6
Thriller	2	6
Horror	1	5
Classic	1	4
Science	1	4
Adventure	1	2

Total rows: 14 Query complete 00:00:00.093

The screenshot shows a PostgreSQL query editor with the following SQL query:

```
--v2 one row per genre with book genre list and total available copies
select genre, count(*) as total_books_in_genre,
sum(available_copies) as Total_copies,
STRING_AGG(Title, ' | ' order by Title) as book_titles
from Books
group by genre
order by Total_copies desc, genre;
```

The Data Output tab shows the following results:

genre	total_books_in_genre	total_copies	book_titles
Fantasy	3	23	Harry Potter and the Sorcerer's Stone The Hobbit The Lord of the Rings
Dystopian	2	12	1984 The Hunger Games
Self-Help	1	12	Atomic Habits
Finance	1	10	Rich Dad Poor Dad
Romance	2	10	Pride and Prejudice The Fault in Our Stars
Drama	1	8	The Kite Runner
Fiction	2	8	The Catcher in the Rye To Kill a Mockingbird
Philosophy	1	7	The Alchemist
History	1	6	Sapiens
Thriller	2	6	The Da Vinci Code The Girl on the Train
Horror	1	5	The Shining
Classic	1	4	The Great Gatsby
Science	1	4	A Brief History of Time
Adventure	1	2	Moby Dick

Total rows: 14 Query complete 00:00:00.081

d) Find the most borrowed book(s) overall.

The screenshot shows a PostgreSQL query editor with the following SQL query:

```
--Find the most borrowed book(s) overall
/*--- Returns all books with the highest borrow count (tie supported) */

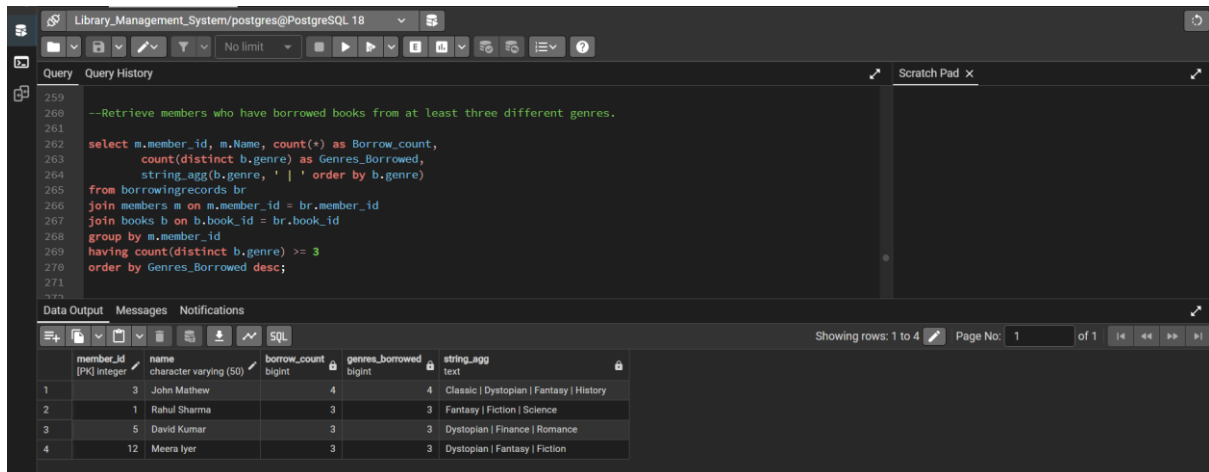
with borrow_stats as (
  select b.title, count(*) as Borrow_count
  from books b
  join borrowingrecords br on b.book_id = br.book_id
  group by b.title
)
select * from borrow_stats
where Borrow_count = (select max(Borrow_count) from borrow_stats);
```

The Data Output tab shows the following results:

title	borrow_count
The Lord of the Rings	8

Showing rows: 1 to 1 Page No: 1 of 1

e) Retrieve members who have borrowed books from at least three different genres.



```

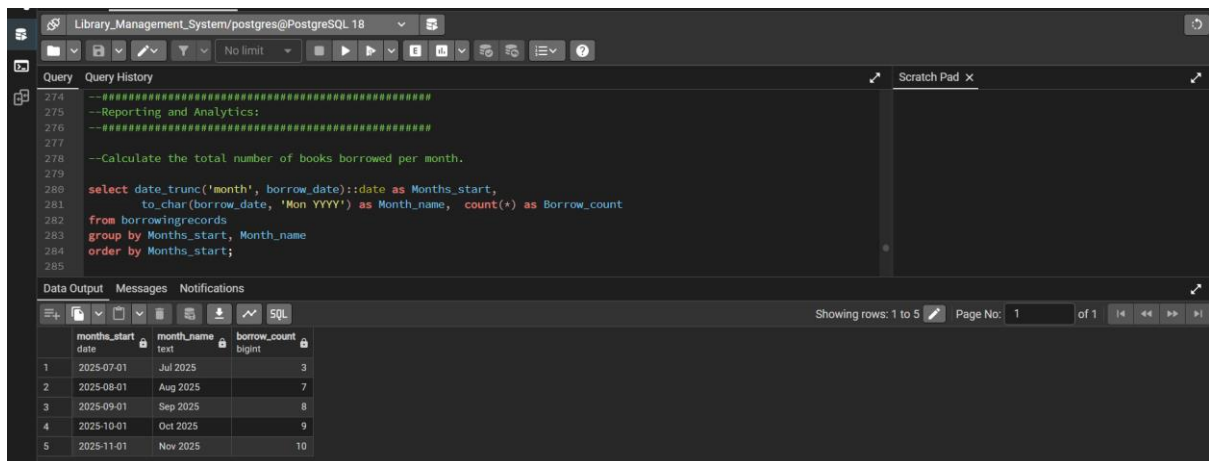
259
260
261 --Retrieve members who have borrowed books from at least three different genres.
262
263 select m.member_id, m.Name, count(*) as Borrow_count,
264        count(distinct b.genre) as Genres_Borrowed,
265        string_agg(b.genre, ' | ' order by b.genre)
266 from borrowingrecords br
267 join members m on m.member_id = br.member_id
268 join books b on b.book_id = br.book_id
269 group by m.member_id
270 having count(distinct b.genre) >= 3
271 order by Genres_Borrowed desc;
272
273

```

member_id [PK] integer	name character varying (50)	borrow_count bigint	genres_borrowed bigint	string_agg text
1	John Mathew	4	4	Classic Dystopian Fantasy History
2	Rahul Sharma	3	3	Fantasy Fiction Science
3	David Kumar	3	3	Dystopian Finance Romance
4	Meera Iyer	3	3	Dystopian Fantasy Fiction

Reporting and Analytics:

a) Calculate the total number of books borrowed per month.



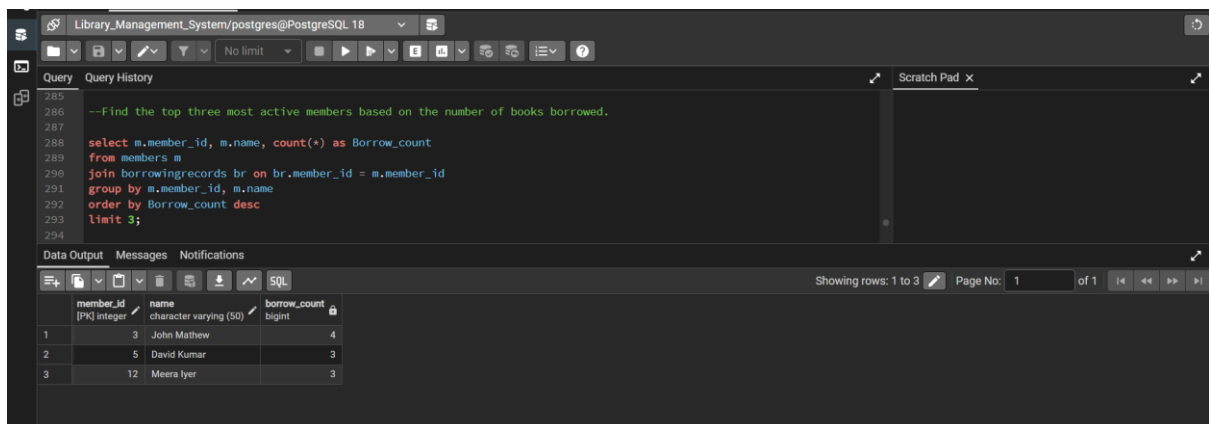
```

274
275
276 -----
277 --Reporting and Analytics:
278 -----
279
280 --Calculate the total number of books borrowed per month.
281
282 select date_trunc('month', borrow_date)::date as Months_start,
283        to_char(borrow_date, 'Mon YYYY') as Month_name, count(*) as Borrow_count
284 from borrowingrecords
285 group by Months_start, Month_name
286 order by Months_start;
287
288

```

months_start date	month_name text	borrow_count bigint
2025-07-01	Jul 2025	3
2025-08-01	Aug 2025	7
2025-09-01	Sep 2025	8
2025-10-01	Oct 2025	9
2025-11-01	Nov 2025	10

b) Find the top three most active members based on the number of books borrowed.



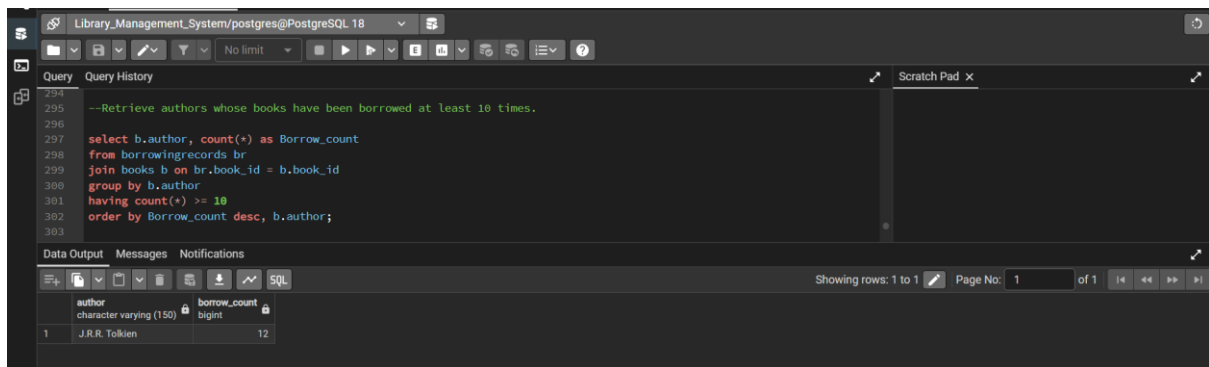
```

285
286
287 --Find the top three most active members based on the number of books borrowed.
288
289 select m.member_id, m.name, count(*) as Borrow_count
290 from members m
291 join borrowingrecords br on br.member_id = m.member_id
292 group by m.member_id, m.name
293 order by Borrow_count desc
294 limit 3;
295
296

```

member_id [PK] integer	name character varying (50)	borrow_count bigint
1	John Mathew	4
2	David Kumar	3
3	Meera Iyer	3

c) Retrieve authors whose books have been borrowed at least 10 times.



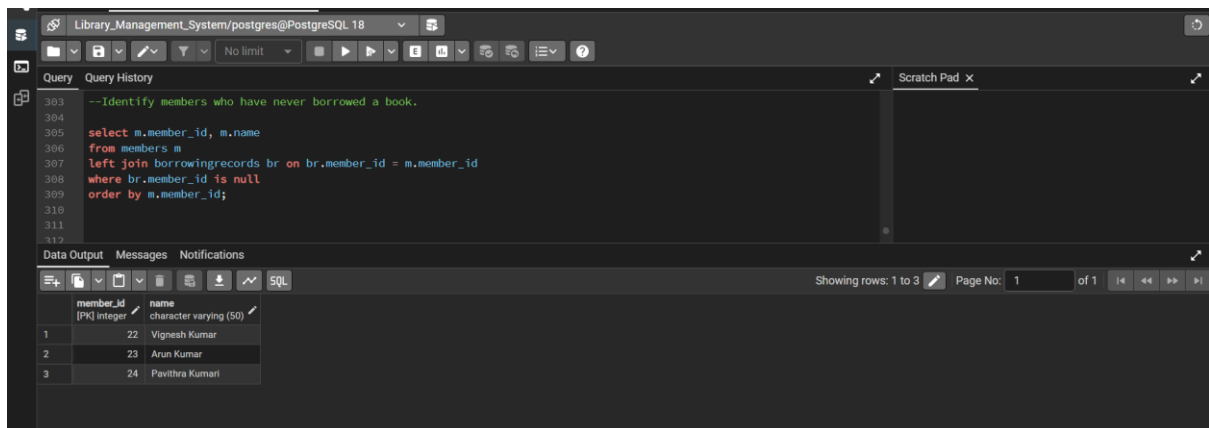
The screenshot shows a PostgreSQL query editor with the following SQL query:

```
--Retrieve authors whose books have been borrowed at least 10 times.  
select b.author, count(*) as Borrow_count  
from borrowingrecords br  
join books b on br.book_id = b.book_id  
group by b.author  
having count(*) >= 10  
order by Borrow_count desc, b.author;
```

The query is executed, and the results are displayed in a table with the following data:

author	borrow_count
J.R.R. Tolkien	12

d) Identify members who have never borrowed a book.



The screenshot shows a PostgreSQL query editor with the following SQL query:

```
--Identify members who have never borrowed a book.  
select m.member_id, m.name  
from members m  
left join borrowingrecords br on br.member_id = m.member_id  
where br.member_id is null  
order by m.member_id;
```

The query is executed, and the results are displayed in a table with the following data:

member_id	name
22	Vignesh Kumar
23	Arun Kumar
24	Pavithra Kumari