

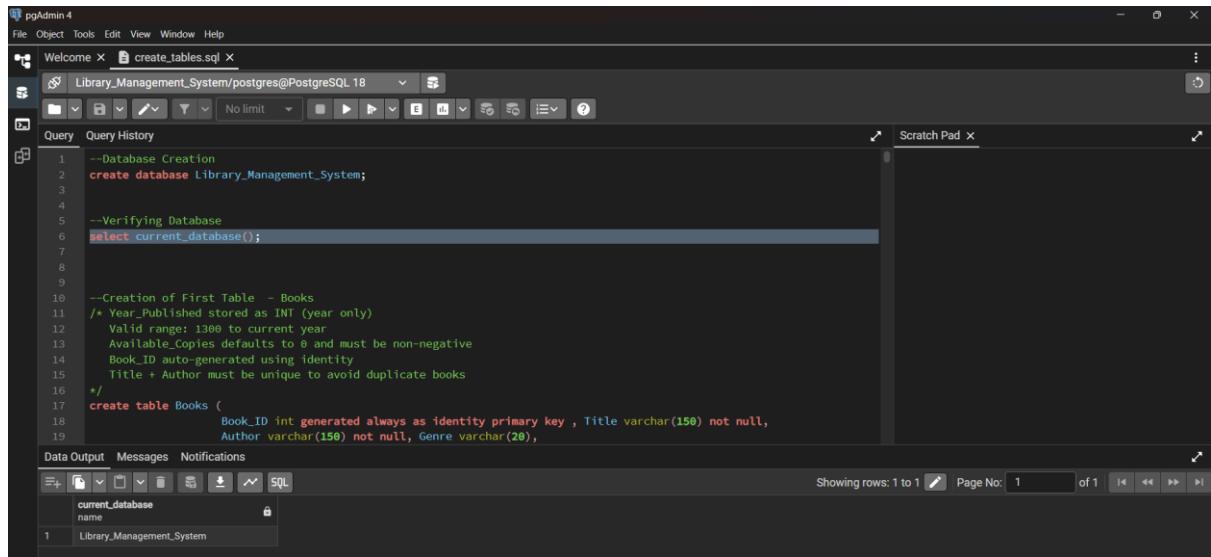
Task 1

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.

Database created



The screenshot shows the pgAdmin 4 interface. In the top-left corner, there's a file menu with options like File, Object, Tools, Edit, View, Window, Help. Below the menu, there's a toolbar with various icons. The main area has a 'Query' tab where a script named 'create_tables.sql' is running. The script contains SQL commands for creating a database and a table. The database creation part includes a comment --Database Creation and a command to create the 'Library_Management_System'. The table creation part includes comments for the first table (Books) and its columns (Book_ID, Title, Author, Genre). The table definition is as follows:

```
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),
```

Below the query tab, there's a 'Data Output' tab showing the result of the 'select current_database();' command, which returns 'Library_Management_System'. At the bottom of the window, there's a status bar with information like 'Showing rows: 1 to 1' and 'Page No: 1'.

Tables created



The screenshot shows a terminal window with PostgreSQL 18. It displays the creation of two tables: 'Books' and 'Members'. The 'Books' table has columns for book_id (integer, primary key, identity), title (character varying(150)), author (character varying(150)), genre (character varying(20)), year_published (integer), and available_copies (integer). It includes indexes for book_id, a composite index for (title, author), and check constraints for available_copies (>= 0) and year_published (>= 1300 AND year_published <= EXTRACT(year FROM CURRENT_DATE)). The 'Members' table has columns for member_id (integer, primary key, identity), name (character varying(50)), email (character varying(50)), phone_no (bigint), address (character varying(150)), and membership_date (date, default CURRENT_DATE). It includes indexes for member_id, a unique constraint for email, and a foreign key constraint referencing the 'Books' table's book_id column.

```
Welcome x Library_Management_System/postgres@PostgreSQL_18 x
page "Notes for Windows users" for details.
Type "help" for help.

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 | |
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
Table "public.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 | |
 return_date | date | | CURRENT_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.

```

-- 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;

```

borrow_id	title	author	borrow_date
1	6 The Great Gatsby	F. Scott Fitzgerald	2025-11-05
2	28 1984	George Orwell	2025-07-20
3	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).

```

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;

```

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

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

--v1 retrieves the genre and with no. of books

```
230
231
232 select genre, count(*) as No_of_Books, sum(available_copies) as Total_copies
233 from books
234 group by genre
235 order by Total_copies desc, genre;
236
```

Data Output

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

Total rows: 14 Query complete 00:00:00.093

The screenshot shows the pgAdmin 4 interface with a query editor and a results grid.

Query Editor:

```
--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;
```

Results Grid:

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

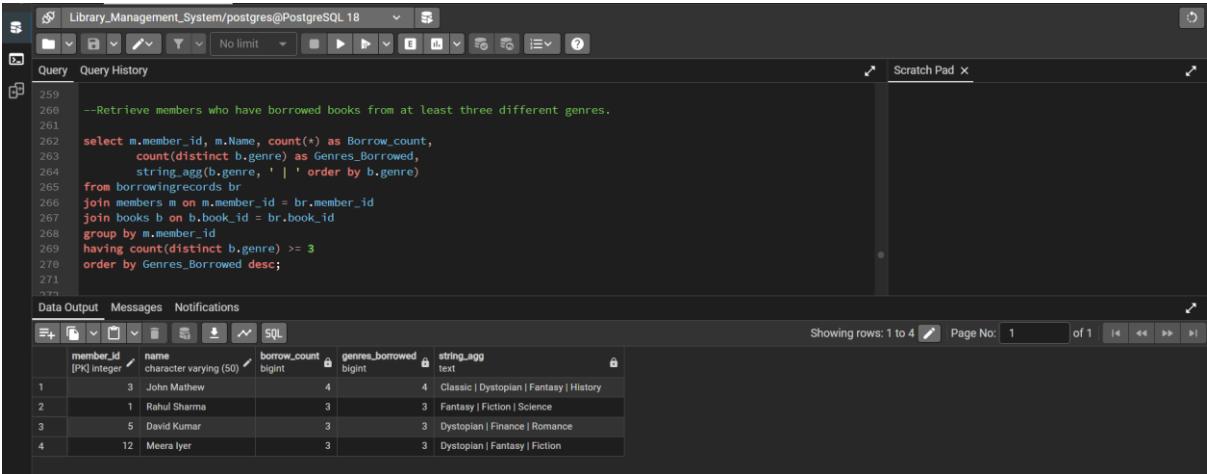
Query Editor:

```
246 --Find the most borrowed book(s) overall
247 /*-- Returns all books with the highest borrow count (tie supported) */
248
249 with borrow_stats as (
250     select b.title, count(*) as Borrow_count
251     from books b
252     join borrowingrecords br on b.book_id = br.book_id
253     group by b.title
254 )
255
256 select * from borrow_stats
257 where Borrow_count = (select max(Borrow_count) from borrow_stats);
```

Data Output:

	title	borrow_count
1	The Lord of the Rings	8

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



```

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

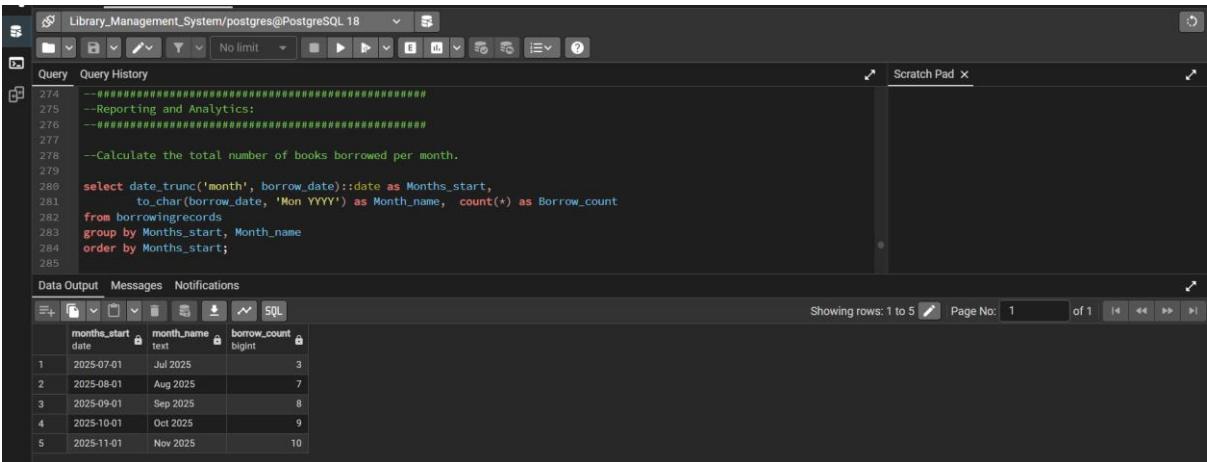
```

The screenshot shows the pgAdmin interface with the SQL tab selected. The query window contains the provided SQL code. The results window displays a table with four rows, each representing a member with their ID, name, borrow count, genres borrowed, and a concatenated string of genres separated by '|'. The columns are labeled: member_id [PK] integer, name character varying(50), borrow_count bigint, genres_borrowed bigint, and string_agg text.

member_id	name	borrow_count	genres_borrowed	string_agg
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.



```

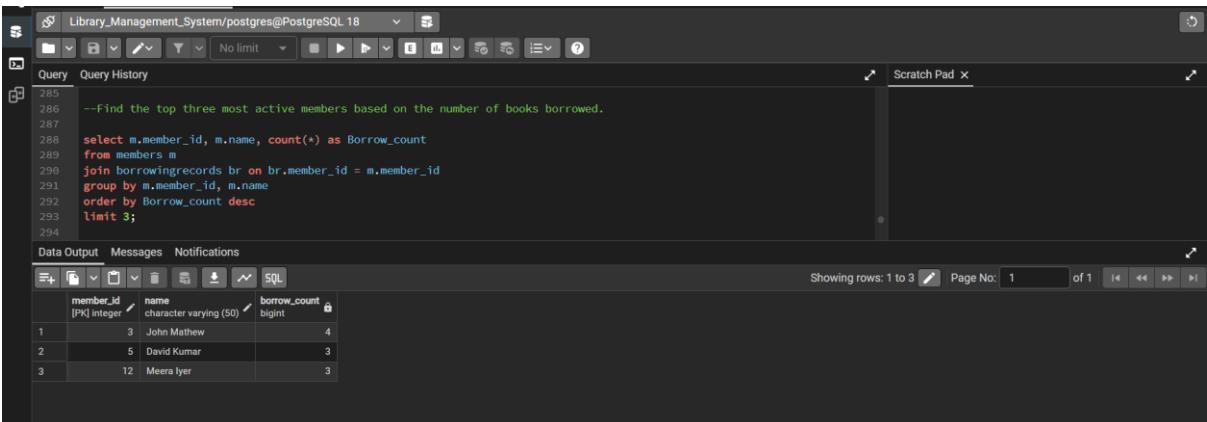
274 #####
275 --Reporting and Analytics:
276 #####
277
278 --Calculate the total number of books borrowed per month.
279
280 select date_trunc('month', borrow_date)::date as Months_start,
281        to_char(borrow_date, 'Mon YYYY') as Month_name, count(*) as Borrow_count
282   from borrowingrecords
283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

```

The screenshot shows the pgAdmin interface with the SQL tab selected. The query window contains the provided SQL code. The results window displays a table with five rows, each representing a month with its start date, name, and borrow count. The columns are labeled: months_start date, month_name text, and borrow_count bigint.

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

```

The screenshot shows the pgAdmin interface with the SQL tab selected. The query window contains the provided SQL code. The results window displays a table with three rows, each representing a member with their ID, name, and borrow count. The columns are labeled: member_id [PK] integer, name character varying(50), and borrow_count bigint.

member_id	name	borrow_count
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 the pgAdmin interface with a query editor and a results grid. The query retrieves authors whose books have been borrowed at least 10 times, ordered by count descending. The result shows one author, J.R.R. Tolkien, with a borrow count of 12.

```
294
295 --Retrieve authors whose books have been borrowed at least 10 times.
296
297 select b.author, count(*) as Borrow_count
298 from borrowingrecords br
299 join books b on br.book_id = b.book_id
300 group by b.author
301 having count(*) >= 10
302 order by Borrow_count desc, b.author;
303
```

author	borrow_count
J.R.R. Tolkien	12

d) Identify members who have never borrowed a book.

The screenshot shows the pgAdmin interface with a query editor and a results grid. The query identifies members who have never borrowed a book by performing a left join between the members and borrowingrecords tables and filtering for null member_ids. The result shows three members: Vignesh Kumar, Arun Kumar, and Pavithra Kumar.

```
303 --Identify members who have never borrowed a book.
304
305 select m.member_id, m.name
306 from members m
307 left join borrowingrecords br on br.member_id = m.member_id
308 where br.member_id is null
309 order by m.member_id;
310
311
312
313
```

member_id	name
22	Vignesh Kumar
23	Arun Kumar
24	Pavithra Kumar