Schema Design & Explanation :

Summary of the schema design :-

5 tables are created as mentioned below :-

Tables and their Attributes:

1 : Authors Table :-

->author_id

->name

->years_of_experience

Primary Key : author_id

2 : Books Table:-

->book_id

->title

->genre : (e.g., Fantasy, Mystery).

->price

->copies_available

->publication_year

Primary Key : book_id

3 : Customers Table:-

->c_id

->name

->email

->phone

->dob

Primary Key : c_id

4 : Book Transactions Table(Relationship between Books and customers):-

->transaction_id

->book_id: Foreign Key referencing book_id in the books table.

->c_id: Foreign Key referencing c_id in the customers table.

->issue_date

->status

->return_date

Primary Key : transaction_id

5 : BookAuthors (Relationship between Books and Authors):-

->book_id: Foreign Key referencing book_id in the books table.

->author_id: Foreign Key referencing author_id in the authors table.

Primary Key: Here Combination of book_id and author_id will act as Primary Key as it is MANY TO MANY Relationship.

Constraints used in Schema Design:

->Primary Key (PK)

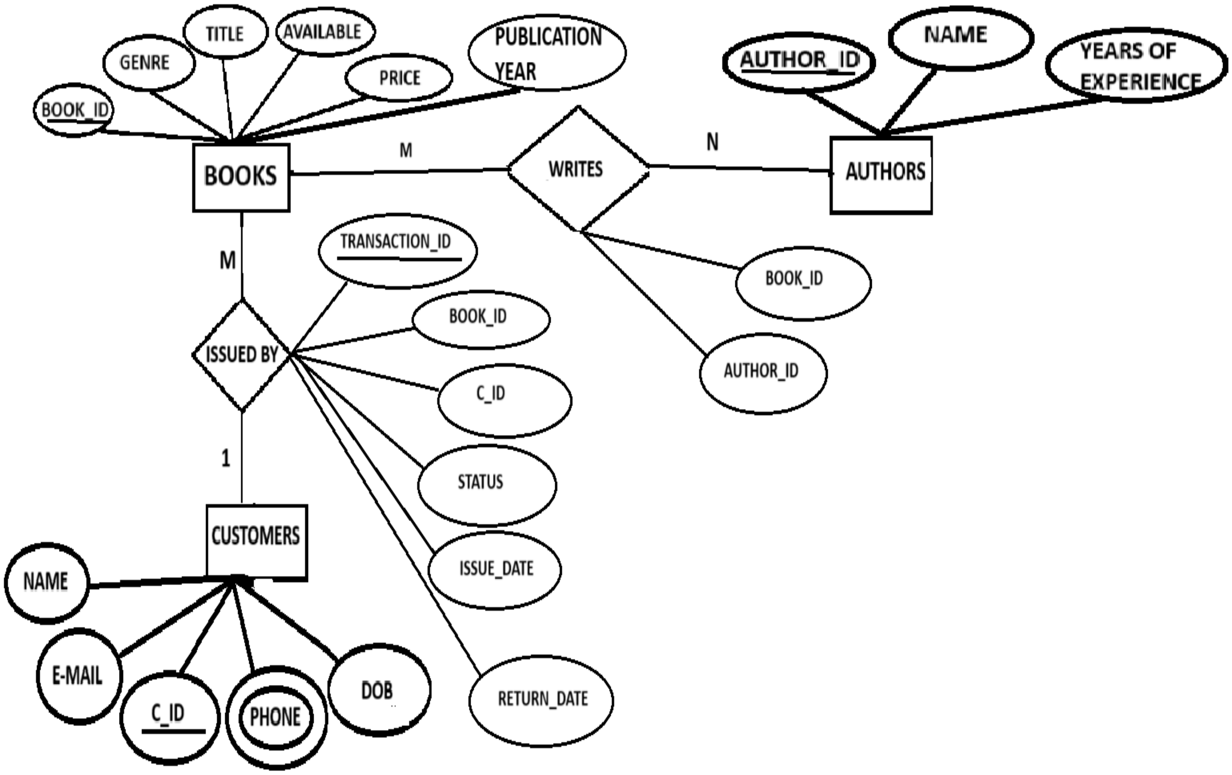
->Foreign Key (FK)

->NOT NULL

->UNIQUE

->ON DELETE CASCADE

ER DIAGRAM :-



BOOKS and AUTHORS have many to many relationship between them due to which separate table namely BOOKAUTHORS needs to be created which have combination of BOOK_ID and AUTHOR_ID AS Primary key.

In this Table BOOK_ID acts as foreign key linking to the BOOK Table and AUTHOR_ID acts as a Foreign Key Linking to the Author Table.

BOOKS and CUSTOMERS are linked through the TRANSACTIONS Table.

In this Table BOOK_ID acts as foreign key linking to the BOOK Table and C_ID acts as a Foriegn Key Linking to the Customers Table.

Relationships :-

->One-to-many: Customers and Books have a one to Many Relationship.

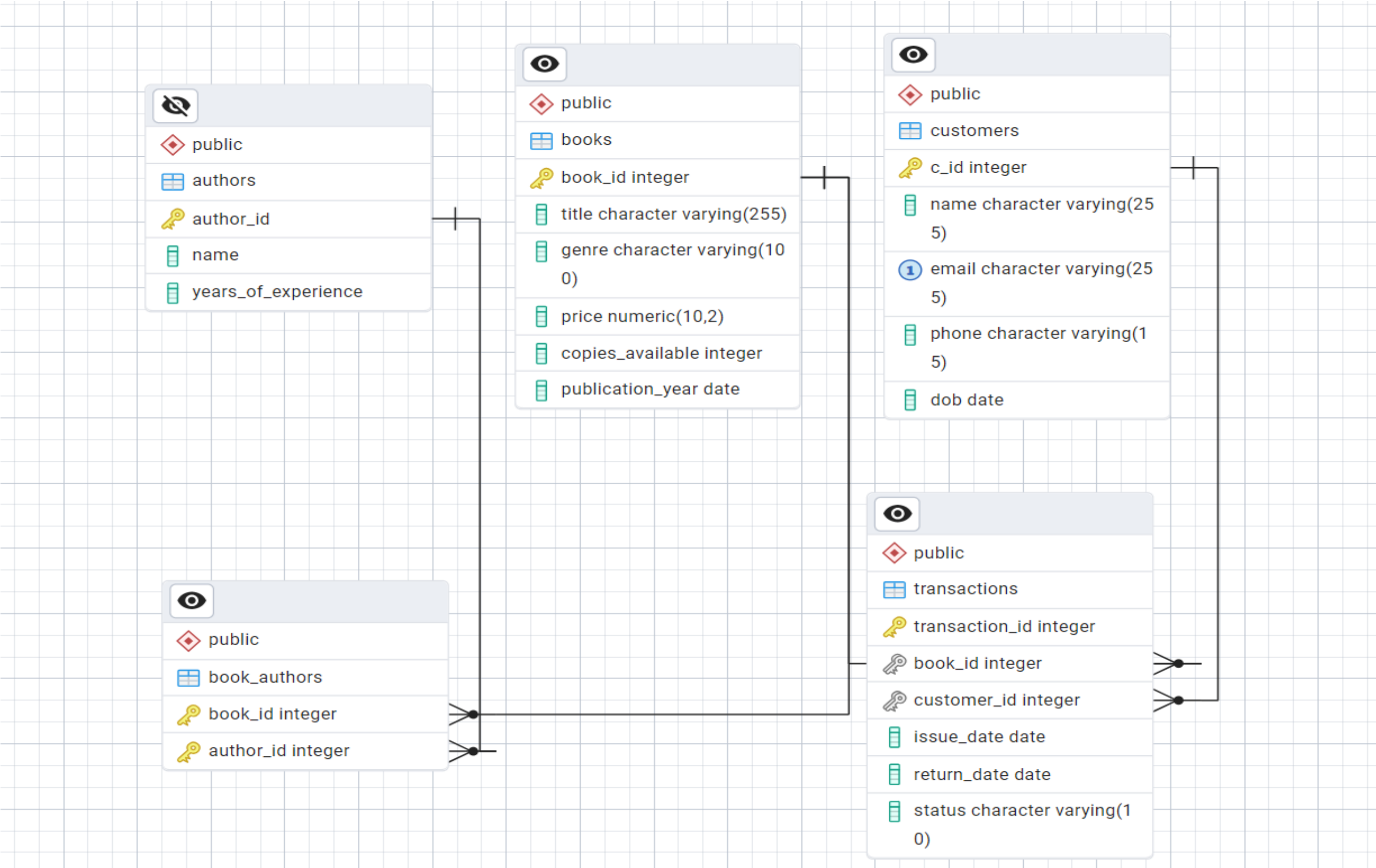
One Customer can borrow many books at a time.

->Many-to-many: Authors and Books have Many to Many Relationship Between them.

A book can be Written by multiple authors, and an author can write multiple books.

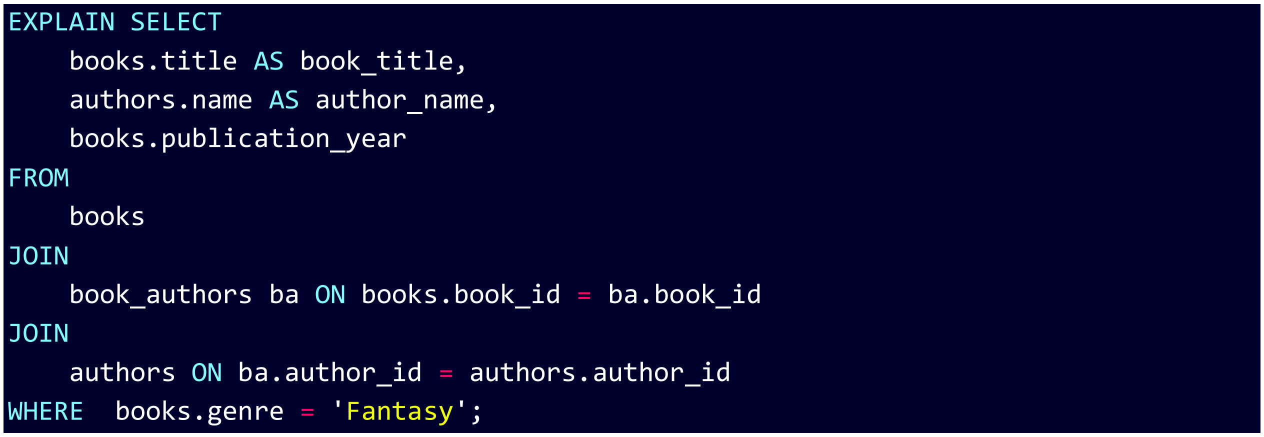
Due to this many to many Relationship BookAuthors table is created.

ERD for Database : -



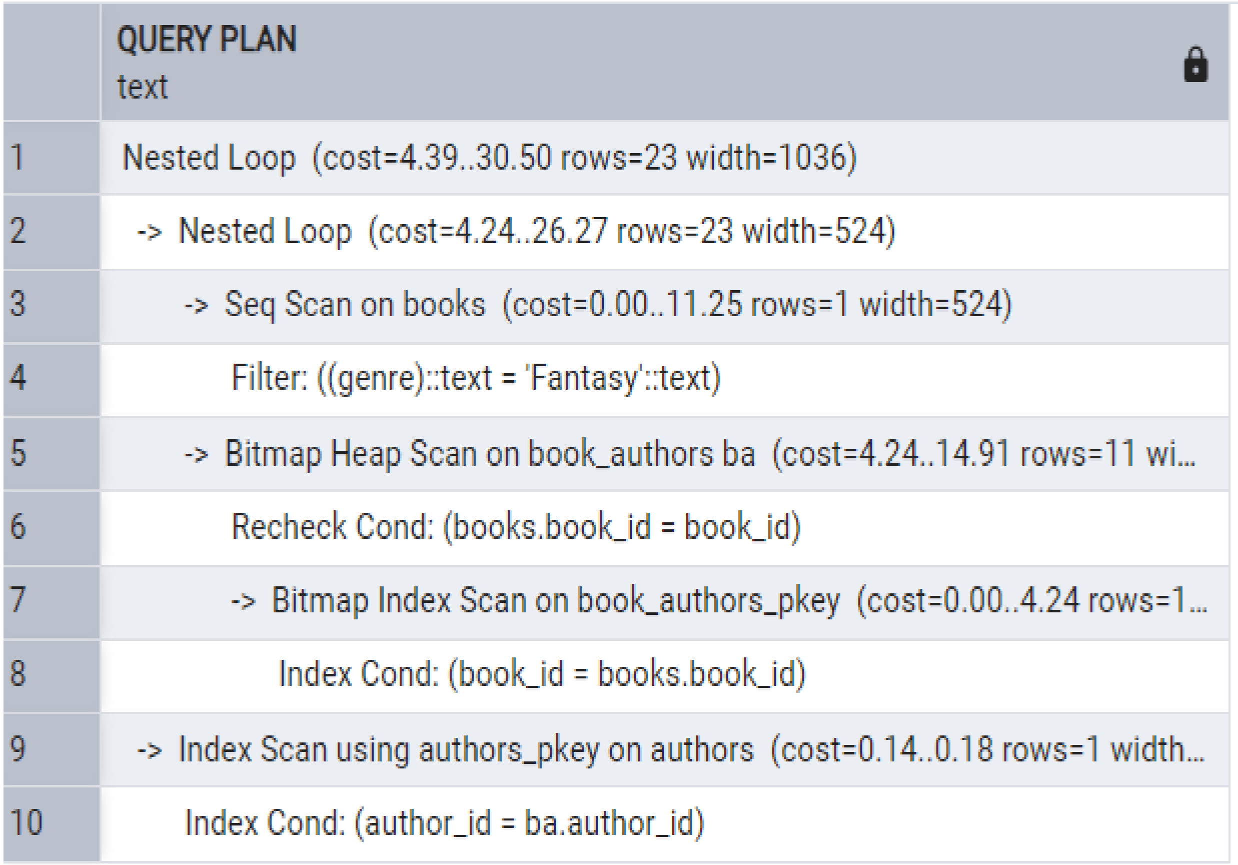
EXECUTION PLANS : -

Consider a Query :



```sql
EXPLAIN SELECT
    books.title AS book_title,
    authors.name AS author_name,
    books.publication_year
FROM
    books
JOIN
    book_authors ba ON books.book_id = ba.book_id
JOIN
    authors ON ba.author_id = authors.author_id
WHERE  books.genre = 'Fantasy';
```

EXPLAIN OUTPUT : -



	QUERY PLAN text
1	Nested Loop (cost=4.39..30.50 rows=23 width=1036)
2	-> Nested Loop (cost=4.24..26.27 rows=23 width=524)
3	-> Seq Scan on books (cost=0.00..11.25 rows=1 width=524)
4	Filter: ((genre)::text = 'Fantasy'::text)
5	-> Bitmap Heap Scan on book_authors ba (cost=4.24..14.91 rows=11 wi...
6	Recheck Cond: (books.book_id = book_id)
7	-> Bitmap Index Scan on book_authors_pkey (cost=0.00..4.24 rows=1...
8	Index Cond: (book_id = books.book_id)
9	-> Index Scan using authors_pkey on authors (cost=0.14..0.18 rows=1 width...
10	Index Cond: (author_id = ba.author_id)

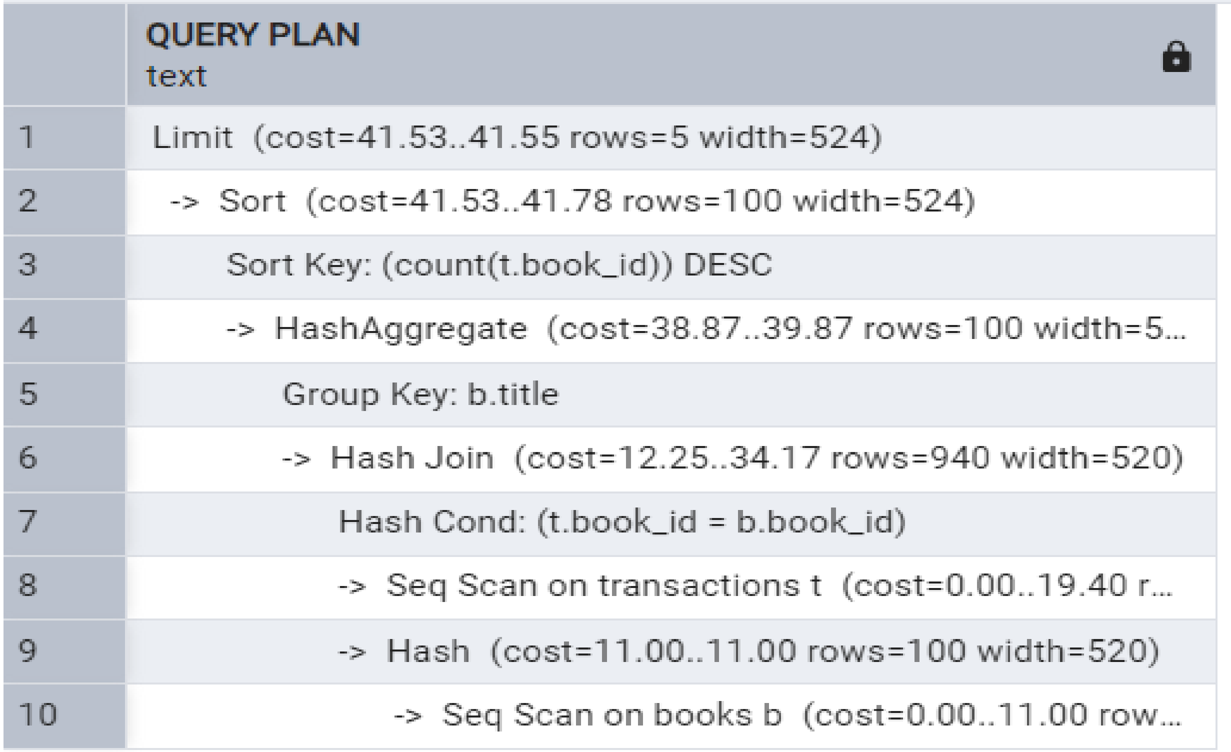
Consider Another Query :

```
EXPLAIN SELECT
    b.title AS Title,
    COUNT(t.book_id) AS issue_count   FROM

    transactions AS t
JOIN
    books b ON t.book_id = b.book_id
GROUP BY    b.title
ORDER BY
    issue_count DESC
LIMIT 5;
```

EXPLAIN OUTPUT : -

--> Scans the book_transactions table sequentially.
-->Join: Matches rows from book_transactions and books on given condition.
-->Group By: Groups rows by title.
-->Sort: Orders the results by issue_count in descending order.
-->Limit: Limits output to 5 rows.



	QUERY PLAN text
1	Limit (cost=41.53..41.55 rows=5 width=524)
2	-> Sort (cost=41.53..41.78 rows=100 width=524)
3	Sort Key: (count(t.book_id)) DESC
4	-> HashAggregate (cost=38.87..39.87 rows=100 width=5...
5	Group Key: b.title
6	-> Hash Join (cost=12.25..34.17 rows=940 width=520)
7	Hash Cond: (t.book_id = b.book_id)
8	-> Seq Scan on transactions t (cost=0.00..19.40 r...
9	-> Hash (cost=11.00..11.00 rows=100 width=520)
10	-> Seq Scan on books b (cost=0.00..11.00 row...

SUMMARY : -

DATABASE USED :

PostgreSql

PROGRAMMING LANGUAGE FOR GENERATING FAKE DATA :

Python

MODULES USED :

->Psycopg2

->Faker

The overall Schema contains proper Normalization Techniques and use of different constraints to ensure efficiency and reduce data redundancy.