Task1:

1. **Describe reasons of why you database considered relational database**

   Database relational because The Primary key in our database id in author table

   And a foreign Key in book_id in book_author table

   The relational database model is a well-established and powerful way to store and organize data. It is a good choice for applications that need to store complex data and relationships between data.

   - **The books table stores information about books, such as their title and category.**
   - **The authors table stores information about authors, such as their name.**
   - **The book_author table stores the relationship between books and authors.**

2. **Identify your database constraints: key, domain, referential integrity.**

   1. **database Primary key in:**

      - **id in author table**
      - **id in book table**
      - **id in category table**

   2. **database domain:**

      - **books.id, authors.id, category.id: These columns are all varchar(8), which contain a unique set of numbers to identify books and authors and category**
      - **books.title, authors.name, category.name: These columns are all strings, there is no specified limit on the length of the text, they contain names and titles of authors and books and categorys**
      - **Copies: These columns are all integer, which contain copy of each books**

   3. **database foreign Key in:**

      - **book_id in book_author table**
      - **author_id in book_author table**
      - **book_id in book_ category table**
      - **category_id in book_ category table**

3. **Explain Relational Algebra Operations may be used for your relational model**

Relational algebra operations form the basis for manipulating and querying data in a relational database model. Here are some key operations:

1. Selection (σ): - Selects rows from a table based on a given condition.

2. Projection (π): - Extracts specific columns (attributes) from a table, creating a subset of the original data.

3. Union (∪): - Combines rows from two tables with the same structure, eliminating duplicates.

4. Intersection (∩): - Retrieves common rows between two tables with the same structure.

5. Difference (-): - Obtains rows from one table that do not exist in another table with the same structure.

6. Join (⋈): - Combines rows from two tables based on a related column (foreign key).

4. **Solve problems like SELECT, PROJECT, UNION, INTERSECTION, and MINUS should be applied between tuples.**

   1. **Selection (σ):**
      Find books with more than 2 copies:
      σ copies > 2 (Book)

      List books written by Friedrich Nietzsche:
      σ first_name = "Friedrich" AND last_name = "Nietzsche" (Author) ⋈ Book_Author ⋈ Book

   2. **Projection (π):**
      Show only book titles and authors:
      π title, first_name, last_name (Book ⋈ Book_Author ⋈ Author)

   3. **Union (∪):**
      Combine books in Poetry and Novel categories:
      (σ category_id = 1 (Book_Category) ⋈ Book) ∪ (σ category_id = 2 (Book_Category) ⋈ Book)

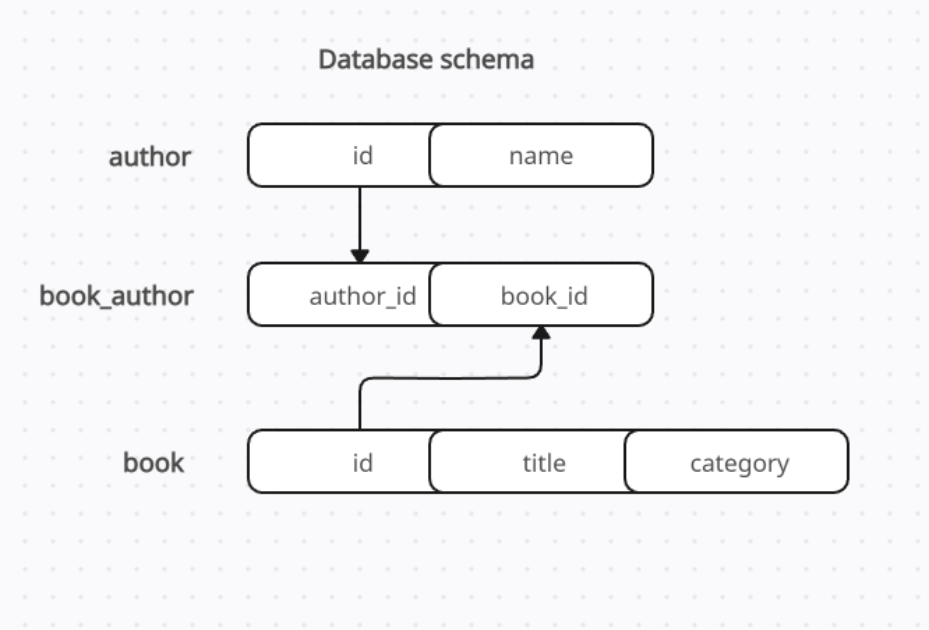
4. **Intersection (∩):**
   Find books written by both Nietzsche and Kernighan:
   (σ author_id = 232 (Book_Author) ⋈ Book) ∩ (σ author_id = 124 (Book_Author) ⋈ Book)

5. **Minus (-):**
   Identify books in Poetry but not in Philosophical Fiction: (σ category_id = 1 (Book_Category) ⋈ Book) - (σ category_id = 3 (Book_Category) ⋈ Book)

5. **To ensure efficiency in your Books and Authors database with categories, focus on normalization, indexing, and optimized queries. Invest in robust hardware, implement caching, and ensure effective concurrency control. Regularly back up data, prioritize security measures, and conduct routine maintenance. Design your database for scalability to accommodate growth without compromising performance.**

Task2:

1. **Prepare an abstract design for your relational schema considering system requirement**

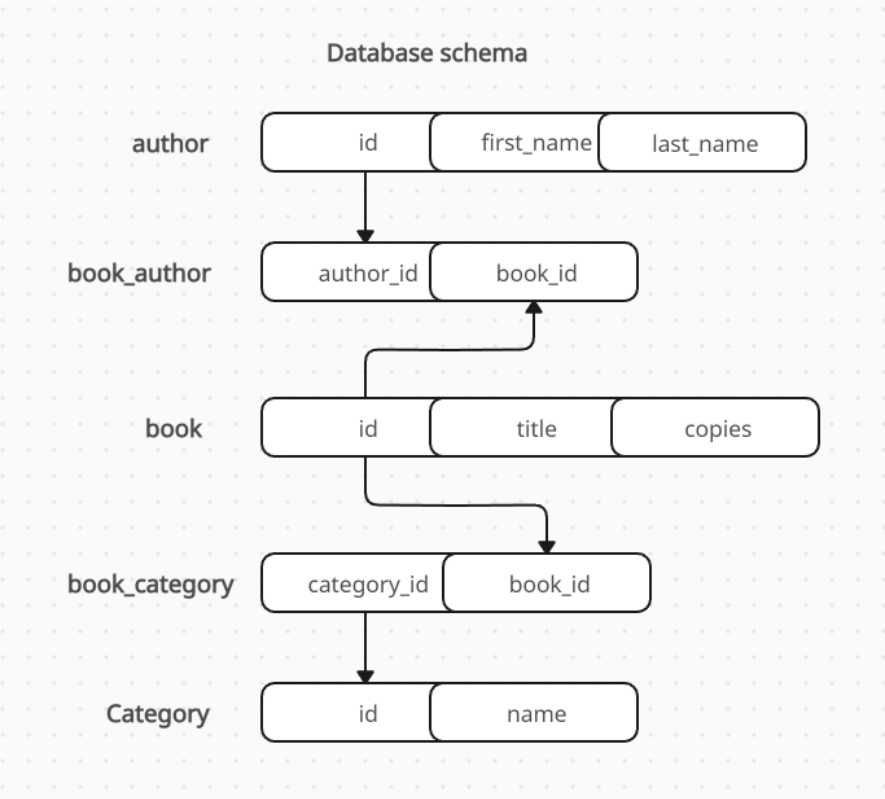


Book		
id	title	category
5648	Thus Spoke Zarathustra	Poetry, Novel, Philosophical fiction
6665	Calculus	mathematics
5648	Thus Spoke Zarathustra	Poetry, Novel, Philosophical fiction

Author	
id	name
232	Friedrich Nietzsche
124	Brian Kernighan

Book_Author	
author_id	book_id
232	5648
124	6665
232	5648

2. Match any of any of normalization first level or second level applying on your schema.



Book	
id	**title**
5648	Thus Spoke Zarathustra
6665	Calculus

Author		
id	**first_name**	**last_name**
232	Friedrich	Nietzsche
124	Brian	Kernighan

Category	
id	**name**
1	Poetry
2	Novel
3	Philosophical fiction
4	mathematics

Book_Author	
author_id	**book_id**
232	5648
124	6665

Book_Category	
category_id	**book_id**
1	5648
2	5648
3	5648
4	6665

3. Classify your relationship types such as 1:1 and 1:m and recursive relationship

1: M (One-to-Many) Relationships:

Book: Book Author (1: M): Each book can have multiple authors

Book: Book_Category (1: M): Each book can belong to multiple categories

M: M (Many-to-Many) Relationship:

Author: Book (M: M): This relationship is resolved through the intermediate table "Book_Author" allowing many authors to write many books, and vice versa.

Author: Category (M: M): This relationship is resolved through the intermediate table "Book_ Category" allowing many categories to have many books, and vice versa.

4. Contrast your database ERD model after normalization