Database Project Report

Domain of interest: Online bookstore

Team Member:

Worawit Penglam 6281453

Sumet Saelow 6280154

Suppachoke Areechitsakul 6281553

Tanapon Techathammanun 6281332

What we are representing in this report

Store Procedures

- Insert
    - Customer
    - Book
    - Book_tag
    - Book_author
    - Book_genre
    - Tag
    - Discount
    - Genre
    - Author
    - Edition
    - Receipt

- Show
    - Customer
    - Book
    - Tag
    - Discount
    - Genre
    - Author
    - Edition
    - Receipt

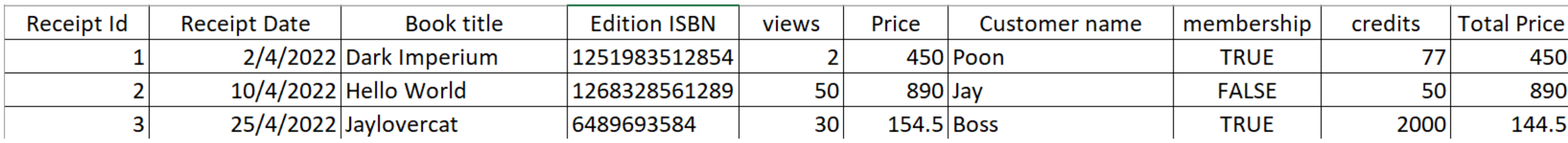
- Delete
  - Customer
  - Book
  - Tag
  - Discount
  - Genre
  - Author
  - Edition
  - Receipt

- Business Operation
  - Buy a book
  - Show author
  - Show receipt by date
  - Show receipt by customer name
  - Search book by
    - Rating
    - Author
    - Promotion
    - Tag
    - ISBN
    - Price range

- Published date
- Book title

Initial Design

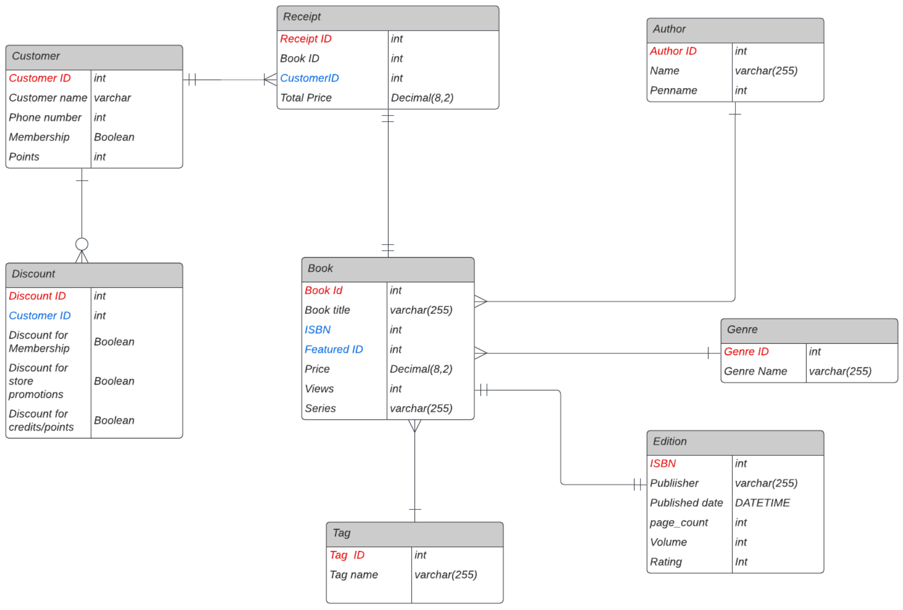
Our team decided to research the structure of the online bookstore and exchange our ideas. First we draft our online bookstore table through excel. The initial design of our database we decided to use a receipt as a root. We start from what the receipt contains, and expand it more further. After we brainstorm, our initial design of our database contains 3 major pieces of information. First is the receipt which is our main, if we consider what the receipt will provide us. For this reason, the next information of our initial design is the customer information. Moreover, the most important thing for our online bookstore is book information.



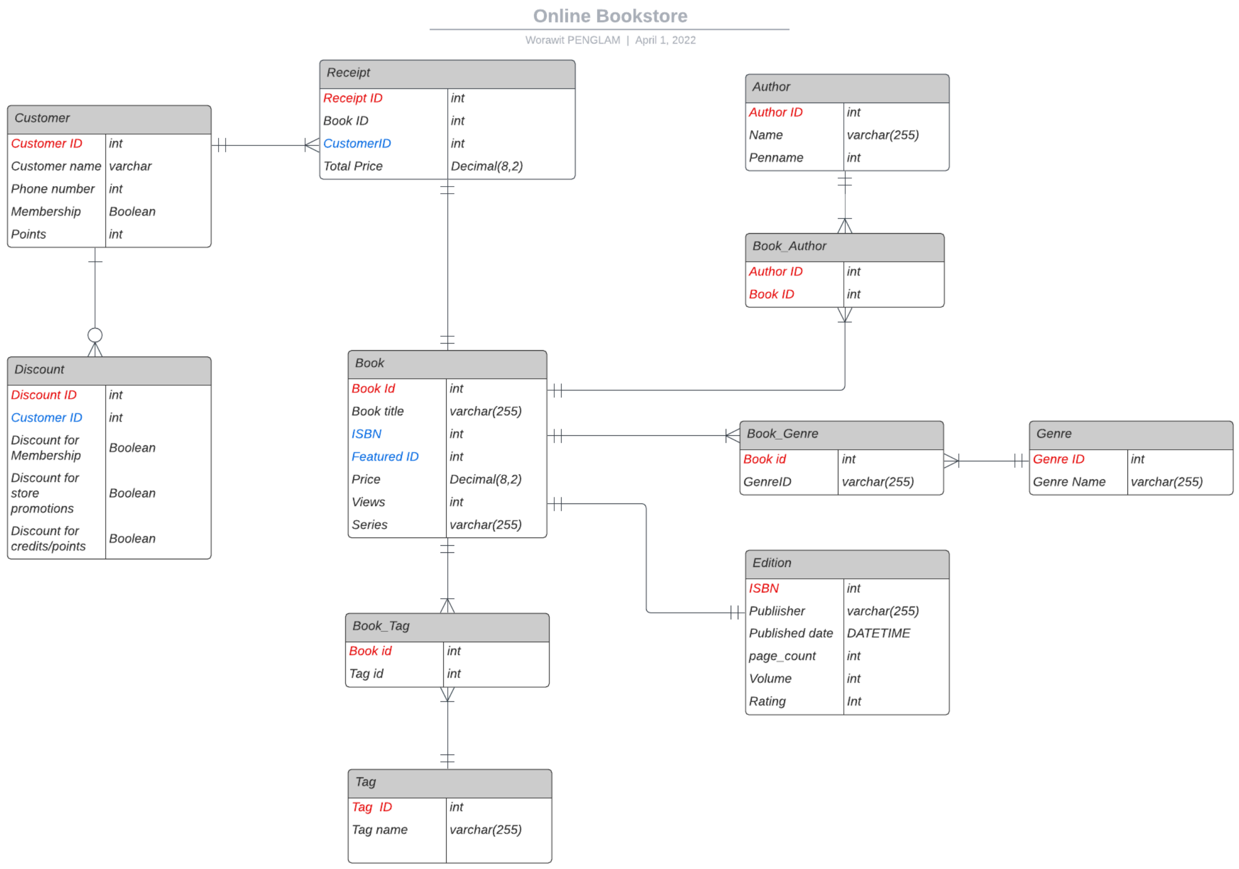
Receipt Id	Receipt Date	Book title	Edition ISBN	views	Price	Customer name	membership	credits	Total Price
1	2/4/2022	Dark Imperium	1251983512854	2	450	Poon	TRUE	77	450
2	10/4/2022	Hello World	1268328561289	50	890	Jay	FALSE	50	890
3	25/4/2022	Jaylovercat	6489693584	30	154.5	Boss	TRUE	2000	144.5

Improve Design

After we looked at our draft of the excel we decided to improve our draft in order to make it more detailed and more realistic. So we are using Lucidchart to draft our improved database. For this one at the part of the customer table. In the initial design we have only customer id, customer name, and membership. We decided to add a discount table to link with the membership of the customer table. The customer ID will be represented as a primary key. At the discount table we conclude to use the discount ID as a primary key and use customer ID as a forienge key. Next we will move to the book table. We improve this table by expanding our author into an Author table, genre into Genre table, Tag into Tag table, and ISBN into Edition table. For the Author table we store the author id as a primary key, author name and the pen name. Genre table we have a genre id as a primary key and a genre name. Tag tables have tag ID as a primary key and a tag name. Lastly for this, improve the design of the edition table. We decided to use ISBN instead of edition ID because each edition of the book will have its own ISBN. For this reason 1 book has 1 and only edition. Moreover, the edition table we adding publisher, publisher date, page count, volume, and rating.



Final Design (3NF)

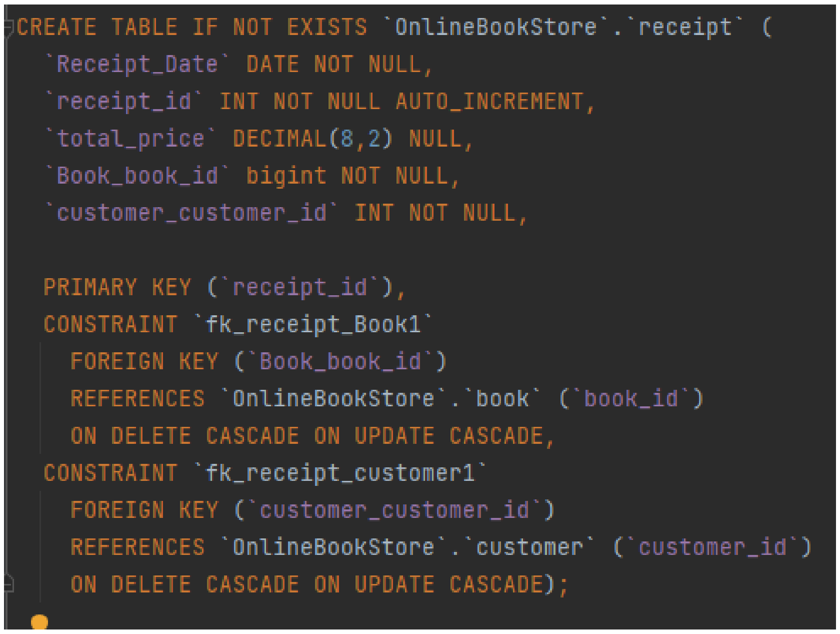


In the Final Design of our database draft. In the early design we connect the Book table with authors, tags, and genres by one to many relations. However, in reality it has to be many to many relations if looked from the both side's perspective. In the beginning, authors can compose as many books as they want and at the same time one book can have more than one author that composed it. Academic articles published by scholars can be one such example. Secondly, one book can have as many tags as long as the tag's description matches the book and at the same time a tag with the same name and description can exist in another book. As for this example, let's think of a book that its author was awarded "the best book of the year" and at the same time it was from the 90s era which means tags such as “90s” are matched to it, now that book has two tags, One is “the best book award of the year’ and second tag is “90s”. Lastly, more than one genre can exist in one book and in the same case as tag, a genre with the same name can exist in another book. One of the good examples for genre cases would be nowaday light novels, sometimes it is an isekai and at the same time, also speaking of isekai, more than one hundred stories with this type of genre have been published each year in Japan.

Tables Explain

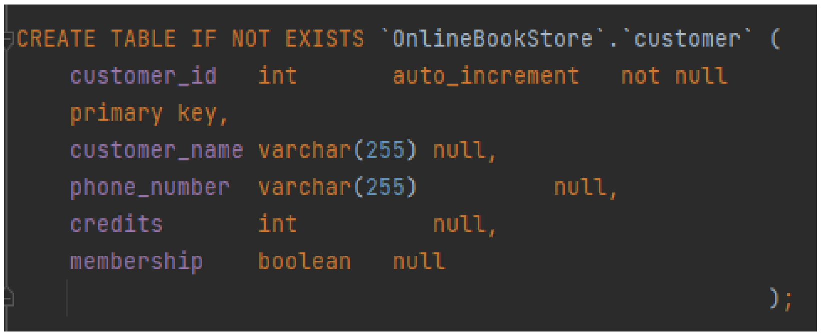
In this part, we will explain in more detail about each table.

- **Receipt**
  - This table contains the information about the purchase of the customer.
  - This table will illustrate us
    - customer information
    - book information
    - date of purchase
    - price of the product



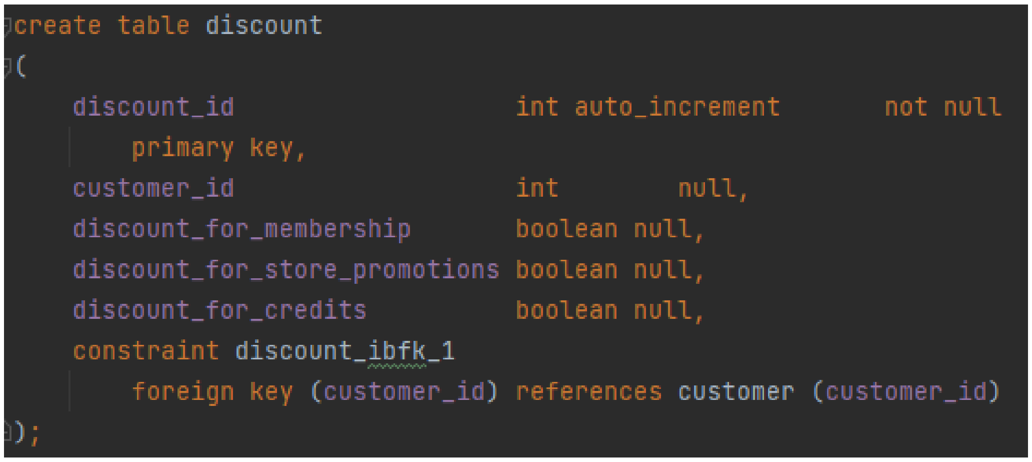
```
CREATE TABLE IF NOT EXISTS `OnlineBookStore`.`receipt` (
  `Receipt_Date` DATE NOT NULL,
  `receipt_id` INT NOT NULL AUTO_INCREMENT,
  `total_price` DECIMAL(8,2) NULL,
  `Book_book_id` bigint NOT NULL,
  `customer_customer_id` INT NOT NULL,

  PRIMARY KEY (`receipt_id`),
  CONSTRAINT `fk_receipt_Book1`
    FOREIGN KEY (`Book_book_id`)
    REFERENCES `OnlineBookStore`.`book` (`book_id`)
    ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_receipt_customer1`
    FOREIGN KEY (`customer_customer_id`)
    REFERENCES `OnlineBookStore`.`customer` (`customer_id`)
    ON DELETE CASCADE ON UPDATE CASCADE);
```

- **Customer**
  - This table contains information of the customer, credits, and membership.
  - This table illustrates
    - customer name
    - Phone number
    - Credits (points)
    - Membership status



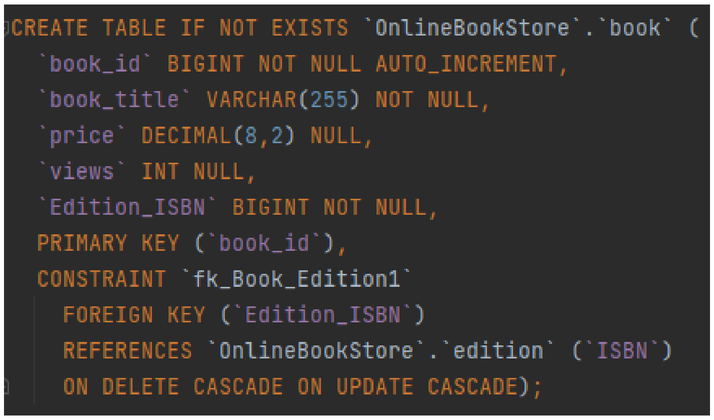
```
CREATE TABLE IF NOT EXISTS `OnlineBookStore`.`customer` (
    customer_id   int       auto_increment   not null
    primary key,
    customer_name varchar(255) null,
    phone_number  varchar(255)          null,
    credits       int          null,
    membership    boolean   null
                                                     );
```

- **Discount**
  - This table links the information with the customer table.
  - It illustrates
    - The status of discount of the membership which is linked with the membership status in the customer table.
    - Discount for store type
    - Discount for credits



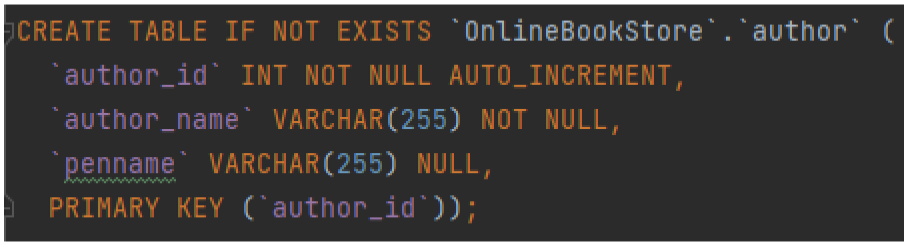
```
create table discount
(
    discount_id                     int auto_increment      not null
        primary key,
    customer_id                     int        null,
    discount_for_membership         boolean null,
    discount_for_store_promotions boolean null,
    discount_for_credits            boolean null,
    constraint discount_ibfk_1
        foreign key (customer_id) references customer (customer_id)
);
```

- **Book**
    - This table contains the basic information of the book.
    - It illustrates
        - Book title
        - Price
        - Views
        - Edition_ISBN



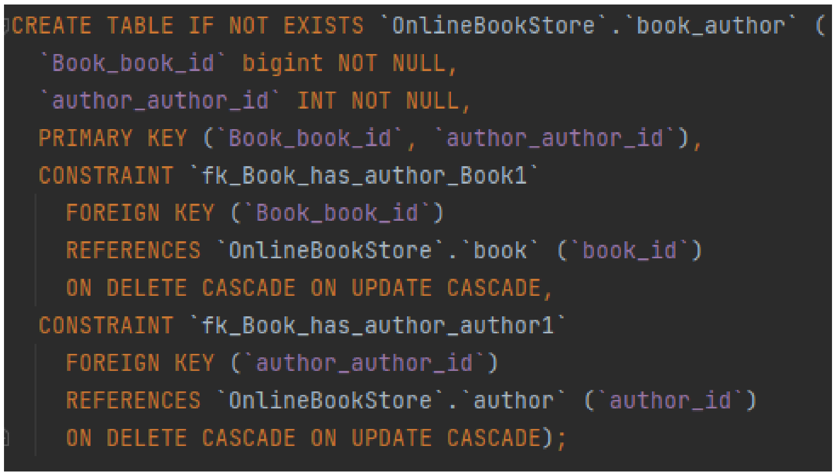
```sql
CREATE TABLE IF NOT EXISTS `OnlineBookStore`.`book` (
  `book_id` BIGINT NOT NULL AUTO_INCREMENT,
  `book_title` VARCHAR(255) NOT NULL,
  `price` DECIMAL(8,2) NULL,
  `views` INT NULL,
  `Edition_ISBN` BIGINT NOT NULL,
  PRIMARY KEY (`book_id`),
  CONSTRAINT `fk_Book_Edition1`
    FOREIGN KEY (`Edition_ISBN`)
    REFERENCES `OnlineBookStore`.`edition` (`ISBN`)
    ON DELETE CASCADE ON UPDATE CASCADE);
```

- **Author**
    - This table is designed to collect the author name and the pen name.



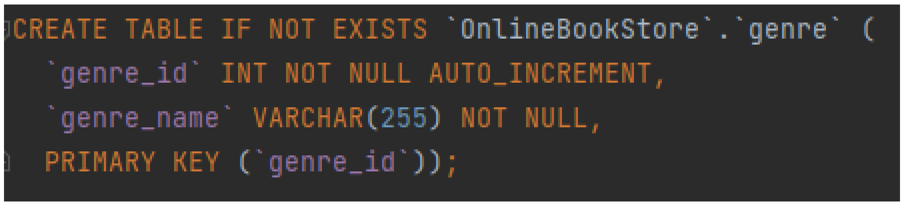
```sql
CREATE TABLE IF NOT EXISTS `OnlineBookStore`.`author` (
  `author_id` INT NOT NULL AUTO_INCREMENT,
  `author_name` VARCHAR(255) NOT NULL,
  `penname` VARCHAR(255) NULL,
  PRIMARY KEY (`author_id`));
```

- **Book_author**
    - This table is an associative entity which links book relation and author relation.
    - This table is designed to support many to many relationships. For the reason that one author can have many books and one book can have many authors.



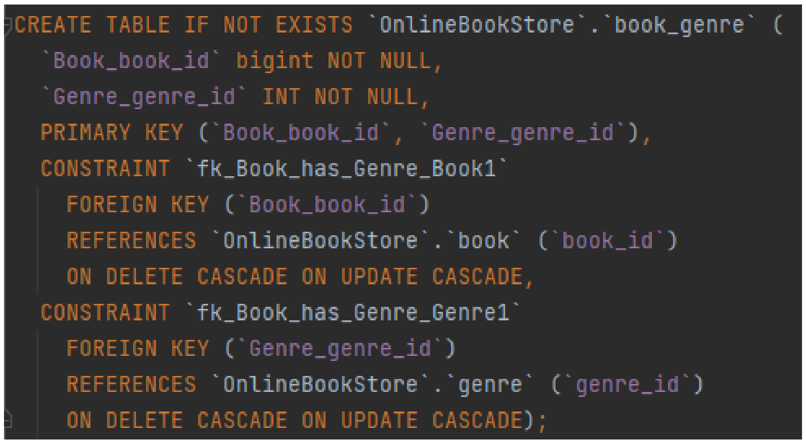
```sql
CREATE TABLE IF NOT EXISTS `OnlineBookStore`.`book_author` (
  `Book_book_id` bigint NOT NULL,
  `author_author_id` INT NOT NULL,
  PRIMARY KEY (`Book_book_id`, `author_author_id`),
  CONSTRAINT `fk_Book_has_author_Book1`
    FOREIGN KEY (`Book_book_id`)
    REFERENCES `OnlineBookStore`.`book` (`book_id`)
    ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_Book_has_author_author1`
    FOREIGN KEY (`author_author_id`)
    REFERENCES `OnlineBookStore`.`author` (`author_id`)
    ON DELETE CASCADE ON UPDATE CASCADE);
```

- **Genre**
  - This table is created to store the genre.



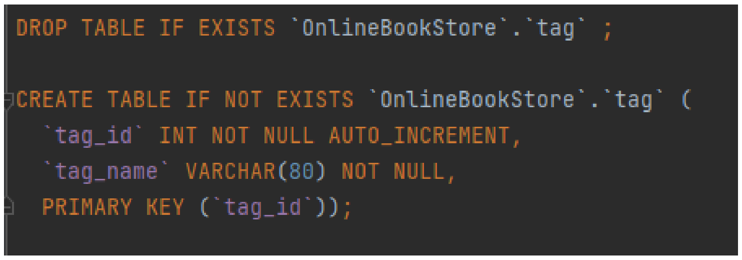
```
CREATE TABLE IF NOT EXISTS `OnlineBookStore`.`genre` (
  `genre_id` INT NOT NULL AUTO_INCREMENT,
  `genre_name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`genre_id`));
```

- **Book_genre**
  - This is another associative entity that connects book and genre together by store Book id and Genre id.
  - This table is designed to support many to many relationships. Because one book can have many genres and one genre can stored in one book



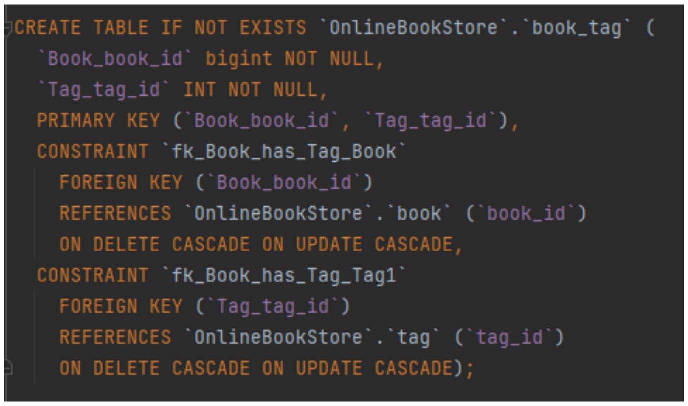
```
CREATE TABLE IF NOT EXISTS `OnlineBookStore`.`book_genre` (
  `Book_book_id` bigint NOT NULL,
  `Genre_genre_id` INT NOT NULL,
  PRIMARY KEY (`Book_book_id`, `Genre_genre_id`),
  CONSTRAINT `fk_Book_has_Genre_Book1`
    FOREIGN KEY (`Book_book_id`)
    REFERENCES `OnlineBookStore`.`book` (`book_id`)
    ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_Book_has_Genre_Genre1`
    FOREIGN KEY (`Genre_genre_id`)
    REFERENCES `OnlineBookStore`.`genre` (`genre_id`)
    ON DELETE CASCADE ON UPDATE CASCADE);
```

- **Tag**
  - This relation stores a list of tags in which a book can have.



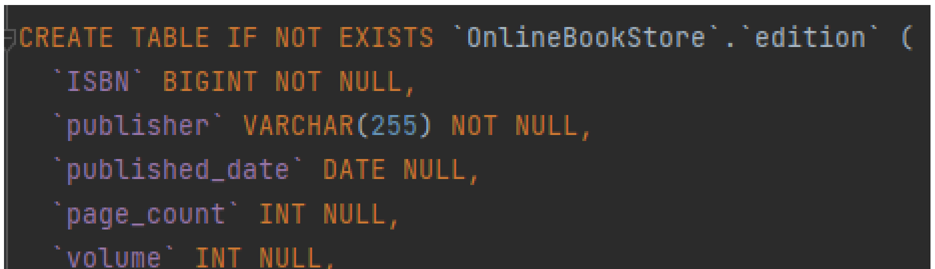
```
DROP TABLE IF EXISTS `OnlineBookStore`.`tag` ;

CREATE TABLE IF NOT EXISTS `OnlineBookStore`.`tag` (
  `tag_id` INT NOT NULL AUTO_INCREMENT,
  `tag_name` VARCHAR(80) NOT NULL,
  PRIMARY KEY (`tag_id`));
```

- **Book_tag**
  - Associative entity that connects book and tag together by store Book id and Tag id.
  - This table is designed to support many to many relationships. In order to support the action that one book can have many tags and one tag can be used in many books.



```sql
CREATE TABLE IF NOT EXISTS `OnlineBookStore`.`book_tag` (
  `Book_book_id` bigint NOT NULL,
  `Tag_tag_id` INT NOT NULL,
  PRIMARY KEY (`Book_book_id`, `Tag_tag_id`),
  CONSTRAINT `fk_Book_has_Tag_Book`
    FOREIGN KEY (`Book_book_id`)
    REFERENCES `OnlineBookStore`.`book` (`book_id`)
    ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_Book_has_Tag_Tag1`
    FOREIGN KEY (`Tag_tag_id`)
    REFERENCES `OnlineBookStore`.`tag` (`tag_id`)
    ON DELETE CASCADE ON UPDATE CASCADE);
```

- **Edition**
  - The edition table is designed to store additional details of the book. These details are not required for other relation i.e. receipt hence, it is stored outside of the book relation in order to keep schema within the book relation minimal.
  - It has one to one and only one relationship not one to many because even though it may sound like one book can have many editions, however each edition has their own ISBN, published date, page counts, etc. Therefore, one book is only linked with one edition.
  - It collects
    - ISBN



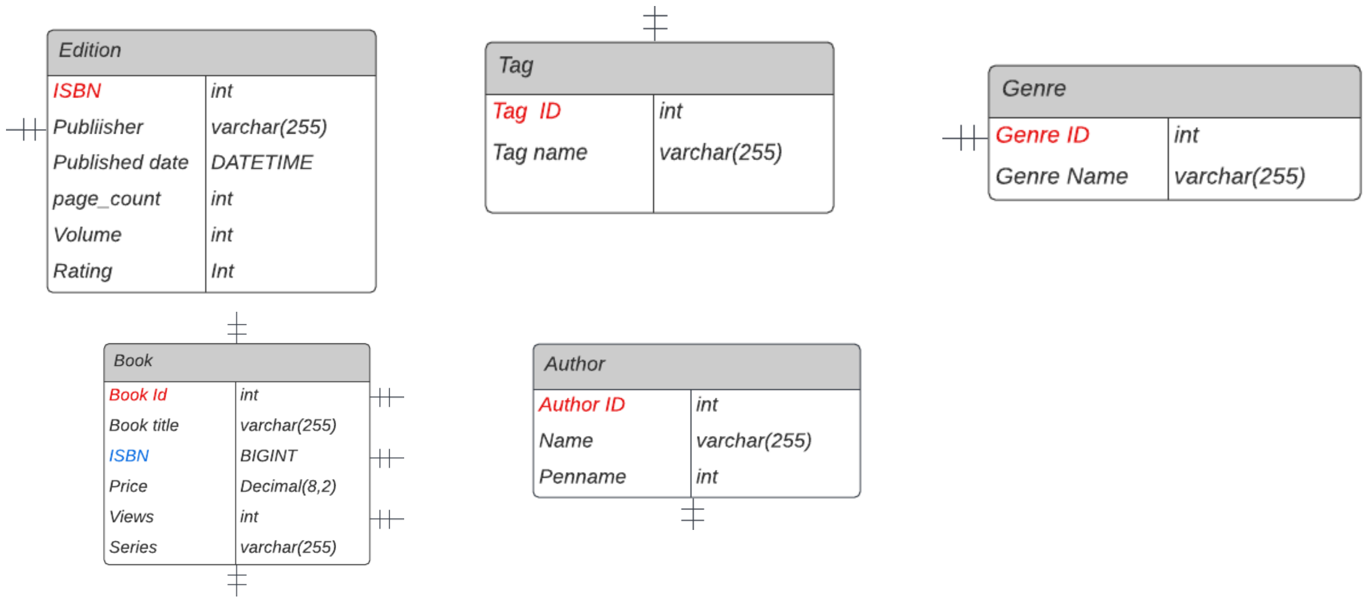
```sql
CREATE TABLE IF NOT EXISTS `OnlineBookStore`.`edition` (
  `ISBN` BIGINT NOT NULL,
  `publisher` VARCHAR(255) NOT NULL,
  `published_date` DATE NULL,
  `page_count` INT NULL,
  `volume` INT NULL,
```

- Publisher
- Publisher_date
- page count
- Volumn
- Rating

Store procedures

For this online book store we came up with 36 procedures. Which came up with 4 types of procedures which are insert, show, delete, and update.

Almost all tables can **insert** and **delete**. Except for the book_author, book_tag, and book_genre these three tables are made up from the forienge key of the other table. In order to **update** or add more details of the book table. You have to interact with these three tables. Which means if the book has multiple authors, tags, and genres you have to insert the first one in the insert book. After that you have to insert the additional information to the book_author, book_tag, and book_genre. And for **show** procedures. Simply typing script SELECT * FROM table it will not show the information that you need because it will show only the initial information of what you need. Moreover, the tables are made up of the foreign key. For example if you would like to see the information of the book “SELECT * FROM Book” will show you just the book id, book title, ISBN, price, and views.



But we want to look at the complete information of the book. However, genre, tag and the author are stored in another table. For this reason we have to use the script command JOIN to get the additional data from the other table. In order to access correct book data.

Business Operations

For the business operation of our project. We created many procedures to support our customers such as search, recommendation, membership system, and discount system. Moreover, customers can check their membership, and point status. Everytime customer purchases a book he or she will gain a credit. This credit or points are used to discount the book price. Customers can decide to use the point, membership or promotion from the book tag to discount the price. For the recommendation system of our project. We create a system such as sorting the price, views, rating, date, and discount. Moreover, we also created a search system which can search for the price range of the book, ISBN, tags, authors, publisher, and book name. Customers can see their purchase history by using customer username. In addition, customers can cancel the order. When the order is canceled if a customer does not use the credits to discount their product. The point which they achieved from the purchasing product will be deducted from their credit and if the customer spends their credits during purchasing they will gain the credits back.