# Data Mining
# Project



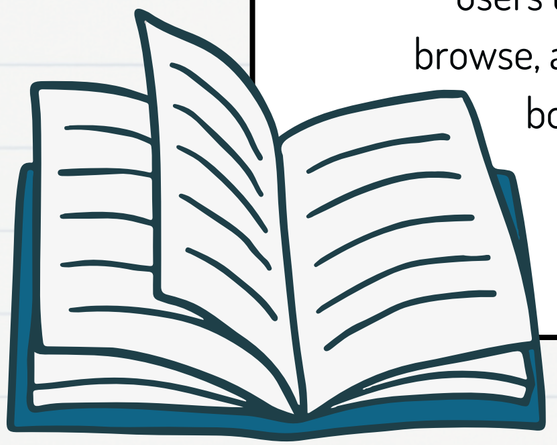

# Purpose

to create a comprehensive library system that allows users to search, browse, and manage books.

provide an efficient and user-friendly interface for accessing and maintaining library resources.

it is a practical application that can be used by libraries, schools, or individuals to manage their book collections.

a valuable tool for researchers, students, or anyone interested in literature.



| Book | |
|---|---|
| int | ID |
| string | Title |
| string | ISBN |
| string | ISSN |
| int | DateofPublishing |
| string | Genre |
| string | Series |
| string | Blurb |
| int | AuthorID |
| int | PublisherID |

| Author | |
|---|---|
| int | ID |
| string | Name |
| string | awardsandaccolades |
| int | dateofbirth |

| Checkout | |
|---|---|
| int | ID |
| int | BookID |
| int | MemberID |
| date | Dateofissue |
| date | Duedate |
| int | Extension |
| date | Actual_return_date |

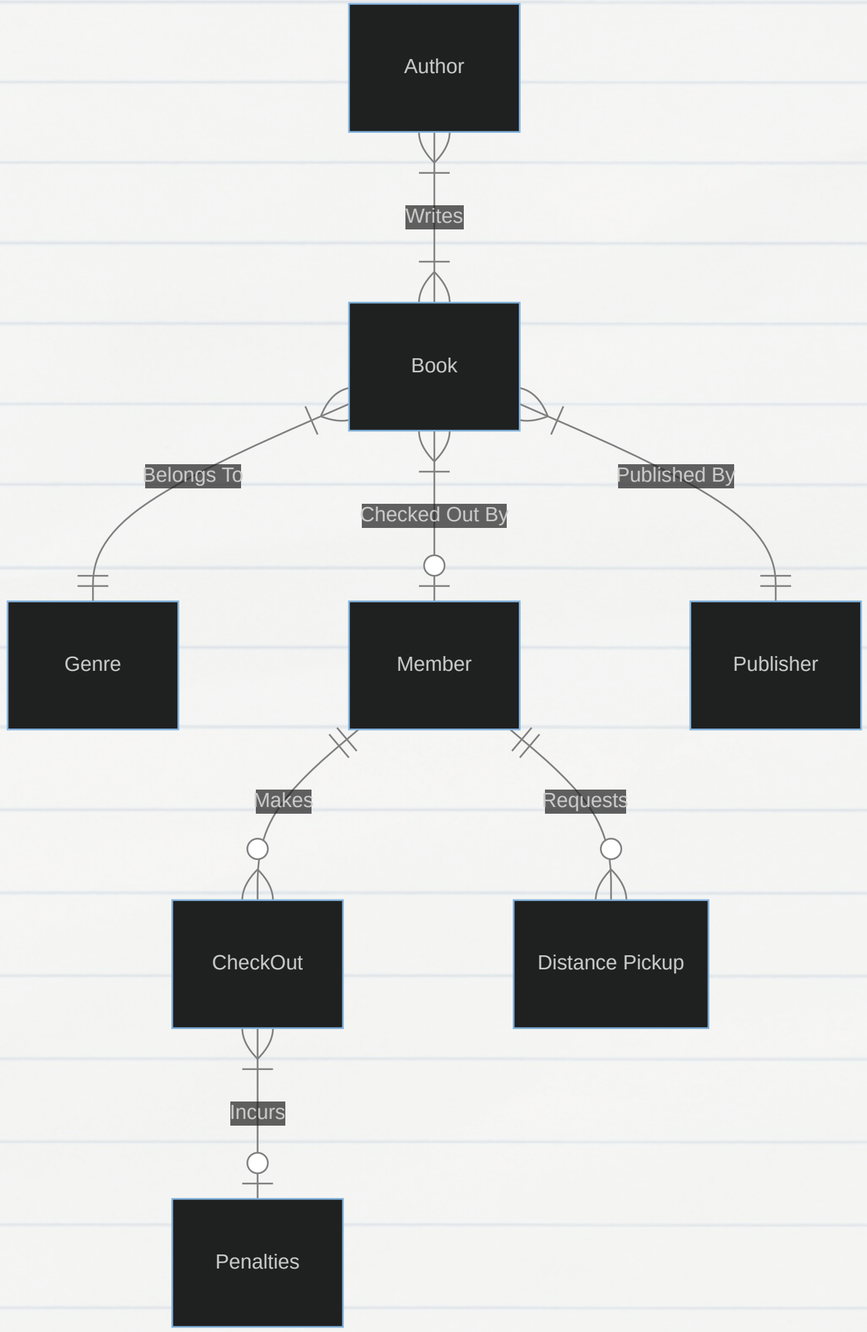
| DistancePickup | |
|---|---|
| int | ID |
| int | BookID |
| int | mEmberID |
| date | PickupDate |
| boolean | PickedUp |

| Genre | |
|---|---|
| int | ID |
| string | Name |
| string | Blurb |

| Member | |
|---|---|
| int | ID |
| string | Name |
| string | Phonenumber |
| string | Address |
| string | Email |
| boolean | Employee |
| string | Status |

| LateFeesPenalties | |
|---|---|
| int | ID |
| int | CheckoutID |
| numeric | PenaltyAmount |
| boolean | PaymentComplete |

# E R Diagram


Author
Writes
Book
Belongs To
Checked Out By
Published By
Genre
Member
Publisher
Makes
Requests
CheckOut
Distance Pickup
Incurs
Penalties

**AUTHOR – BOOK**

Many-to-Many

**BOOK – GENRE**

Many-to-One

**BOOK – PUBLISHER**

Many-to-One

**BOOK – MEMBER**

Many-to-One

**MEMBER – CHECKOUT**

One-to-Many

**CHECKOUT– PENALTIES**

One-to-One

**MEMBER – DISTANCEP ICKUP**

One-to-Many

# Insertion

```sql
INSERT INTO "genre" ("Name", "Blurb")
VALUES
    ('educational textbook', 'Books designed to educate and teach'),
    ('fantasy', 'Books containing elements of imagination and magic'),
    ('philosophy', 'Books exploring fundamental questions about existence and reality'),
    ('autobiography', 'Books recounting the life story of the author'),
    ('sci-fi', 'Books featuring futuristic or speculative science-based themes');

INSERT INTO "Author" ("Name", "awardsandaccolades", "dateofbirth")
VALUES
    ('J.K. Rowling', 'Pulitzer', 1965),
    ('Stephen Hawking', 'Nobel Prize', 1942),
    ('George Orwell', 'Man Booker', 1903),
    ('J.R.R. Tolkien', 'Neustadt', 1892),
    ('Haruki Murakami', 'Hugo', 1949),
    ('Ernest Hemingway', 'Printz', 1899),
    ('Jane Austen', 'John Newbery', 1775);

INSERT INTO "Member" ("Name", "Phonenumber", "Address", "Email", "Employee", "Status")
VALUES
    ('John Doe', '+1234567890', '123 Main St, Anytown, USA', 'john@example.com', 1, 'Valid'),
    ('Alice Smith', '+1987654321', '456 Elm St, Othertown, USA', 'alice@example.com', 0, 'Valid'),
    ('Bob Johnson', '+1122334455', '789 Oak St, Another Town, USA', 'bob@example.com', 1, 'On-Hold'),
    ('Emily Brown', '+1555666777', '321 Pine St, Somewhere, USA', 'emily@example.com', 0, 'Renewal Pending');


INSERT INTO "Book" ("Title", "ISBN", "ISSN", "DateofPublishing", "Genre", "Series", "Blurb", "AuthorID", "PublisherID")
VALUES
    ('Harry Potter and the Philosopher''s Stone','ISSN 0004-6361' , 'ISBN 978-0-306-40615-7', 1997, 'fantasy', 'Harry Potter', 'The first book in the Harry Potter series', 1,
    ('1984','ISSN 0004-0021', 'ISBN 128-0-306-40315-4', 1949, 'sci-fi', NULL, 'A dystopian novel by George Orwell', 3, 2),
    ('The Hobbit', 'ISSN 0004-6361' , 'ISBN 478-3-306-40635-5', 1937, 'fantasy', NULL, 'A fantasy novel by J.R.R. Tolkien', 4, 3),
    ('Sputnik Sweetheart', 'ISSN 0203-6123 ', 'ISBN 978-0-306-23445-7', 1999, 'sci-fi', NULL, 'A novel by Haruki Murakami', 5, 4),
    ('To Kill a Mockingbird', 'ISSN 1204-2345 ', 'ISBN 324-4-121-12345-5', 1960, 'philosophy', NULL, 'A novel by Harper Lee', NULL, 5);
```

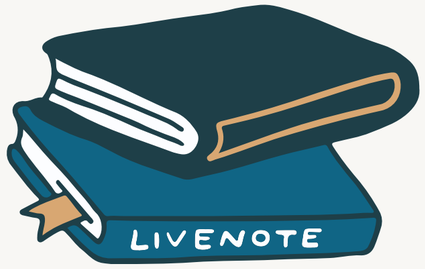
```
DELETE FROM "checkout"
WHERE "BookID" = (SELECT "ID" FROM "Book" WHERE "Title" = 'The Hobbit');
```

Before

| BookID | MemberID | Dateofissue | Duedate | Extension |
|---|---|---|---|---|
| 1 | 1 | 2024-03-10 | 2024-03-31 | 0 |
| 2 | 2 | 2024-03-05 | 2024-03-26 | 0 |
| 3 | 3 | 2024-03-01 | 2024-03-22 | 0 |
| 4 | 4 | 2024-02-25 | 2024-03-17 | 1 |

After

| BookID | MemberID | Dateofissue | Duedate | Extension |
|---|---|---|---|---|
| 1 | 1 | 2024-03-10 | 2024-03-31 | 0 |
| 2 | 2 | 2024-03-05 | 2024-03-26 | 0 |
| 4 | 4 | 2024-02-25 | 2024-03-17 | 1 |




```
UPDATE Member
SET Address = '101 STREET'
WHERE Name = 'John Doe';
```

Before

| ID | Name | Phonenumber | Address | Email | Employee | Status |
|---|---|---|---|---|---|---|
| 1 | John Doe | +1234567890 | 123 Main St, Anytown, USA | john@example.com | 1 | Valid |

After

| ID | Name | Phonenumber | Address | Email | Employee | Status |
|---|---|---|---|---|---|---|
| 1 | John Doe | +1234567890 | 101 STREET | john@example.com | 1 | Valid |

```
ALTER TABLE Member ADD COLUMN Rating INTEGER DEFAULT 0;
```

Before

| ID | Name | Phonenumber | Address | Email | Employee | Status |
|---|---|---|---|---|---|---|
| 17 | John Doe | +1234567890 | 101 STREET | john@example.com | 1 | Valid |
| 18 | Alice Smith | +1987654321 | 456 Elm St, Othertown, USA | alice@example.com | 0 | Valid |

After

| ID | Name | Phonenumber | Address | Email | Employee | Status | Rating |
|---|---|---|---|---|---|---|---|
| 17 | John Doe | +1234567890 | 101 STREET | john@example.com | 1 | Valid | 0 |
| 18 | Alice Smith | +1987654321 | 456 Elm St, Othertown, USA | alice@example.com | 0 | Valid | 0 |

# Indexing

```sql
CREATE INDEX idx_member_name ON Member (Name);
CREATE INDEX idx_book_title ON Book (Title);
```

```sql
--covering index
CREATE INDEX idx_book_covering ON Book (DateofPublishing, Genre);
```

```sql
--partial index
CREATE INDEX idx_book_partial_genre ON Book (Genre) WHERE Genre IN ('sci-fi', 'fantasy') AND DateofPublishing > 2000;
```

# Views

**Book Recommendations**
personalize recommendations for members by suggesting books in genres they've previously borrowed.

**FrequentReaders**
finds top 5 members with the most borrowed books.

**PopularGenres**
finds top 3 trending genres

**BookOfTheDay**
randomly selecting a book to display each day

# Triggers

```
CREATE TRIGGER before_checkout_update
BEFORE UPDATE ON "checkout"
FOR EACH ROW
BEGIN
    UPDATE "checkout"
    SET "Extension" = CASE
        WHEN (SELECT "Employee" FROM "Member" WHERE "ID" = NEW."MemberID") = 1 THEN LEAST(NEW."Extension" + 1, 5)
        ELSE LEAST(NEW."Extension" + 1, 4)
        END;
END;

CREATE TRIGGER update_member_status
AFTER INSERT ON "distancepickup"
FOR EACH ROW
BEGIN
    UPDATE "Member"
    SET "Status" = 'On-Hold'
    WHERE "ID" = NEW."mEmberID" AND (SELECT COUNT(*) FROM "distancepickup" WHERE "mEmberID" = NEW."mEmberID" AND "PickedUp" = 0) = 3;
END;

CREATE TRIGGER CalculateLateFees
AFTER UPDATE ON "checkout"
FOR EACH ROW
WHEN NEW.`Actual return date` > OLD.`Duedate`
BEGIN
    INSERT INTO `latefees/penalties` (`CheckoutID`, `PenaltyAmount`, `PaymentComplete`)
    VALUES (NEW.`ID`, (julianday(NEW.`Actual return date`) - julianday(OLD.`Duedate`)) * 50, FALSE);
END;
```

# Optimizations

1

- Indexed columns for faster retrieval rate
- Parameterized queries were used as opposed to using directly user inputted values into SQL String. Reduces probability of SQL Injection Attacks.

2

- .Views for commonly requested queries. promoting reusability of code and improving maintainability.
- Triggers allow for enforcement of business policies within the schema, as opposed to running a non native application.
- Used julianday to calculate the difference between the actual return date of a book and the day it was meant to be returned.

3

- Appropriate usage of Aggregate Functions and Joins increasing ability to effectively analyze data, and reducing the need for separate multiple queries.
- Use of random function to used in the view of book of the day

# Limitations

- The trigger, before_checkout_update, that updates the Extension Column, can possibly be more effectively implemented in a script or application code. For the sake of displaying proficiency in SQL and DBMS Tools, we implemented a trigger instead.
- The BookRecommendations View could take into account borrowing history and ratings of previously borrowed books to give a comprehensive recommendation
- Instead of storing genre names directly in the Book table, we could have used a genreID corresponding to each genre
- Assuming a more complex system and library, we'd have to include support for hierarchical genres. The library system we are modelling doesn't require this degree of complexity.
- We have not added a time based reset system for the extensions provided to members.

# Thank You