EECS 447 Project

Project Visions/Plan

Version 1.1

Revision History

Date	**Version**	**Description**	**Author**
3/29/2025	1.1		Sophia Jacob, Kusuma Murthy, Anna Lin, Nimra Syed, Ella Nguyen, Nikka Vuong

Table of Contents

1 Introduction

1.1 Project Overview

As a Software as a Service (SaaS) company, ASKNrEceive (ASKNE) hopes to help streamline database creation and management. The purpose of this project is to provide local libraries with a Library Management System (LMS) that ensures efficiency for both types of end-users: library staff and library customers. This abstracted database will facilitate better documentation of the books, magazines, and other available items for ease of management and organization.

Through this project, ASKNE will develop a fully functional and scalable database software for libraries to handle, maintain, and analyze the popularity of their collections through detailed reports. Using a streamlined approach, the goal is to design a robust relational database with a well-defined conceptual schema and physical implementation.

**** Note: Refer to *01 - Project Plan/Vision* for any additional information on the project overview and definition of the project.**

1.2 Scope

ASKNE will use MySQL and other database management tools to deliver an extensive database backed in software to provide better insights and organization to libraries regarding books, magazines, and other content they house. This project entails creating a full-scale LMS, powered by a relational database. ASKNE aims to create a reporting style that helps the library determine popular books, members' favorite genres, and more quantifiable statistics to help maintain a growing library. This project will be responsible for managing the library's database operations such as adding new books as parts of entries, cleaning the database by removing and editing the borrow and return status of loanable items, and finally creating comprehensive reports for the Library admins and staff. Other features include tracking books and members based on a variety of attributes like author names, item IDs, publication dates, and more. Users can view the database for available books, while library staff can edit and maintain the database. As for the technical aspect of this project, ASKNE will be utilizing Structured Query Language (SQL) as the main tool used for searching the database and will create various tables that will be structured in a Relational database format.

**** Note: Refer to *01 - Project Plan/Vision* for any additional information on the scope and definition of the project.**

1.3 Glossary

**** Note: Our team has a dedicated Document for Abbreviations and Glossary. Refer to *07 - Glossary*.**

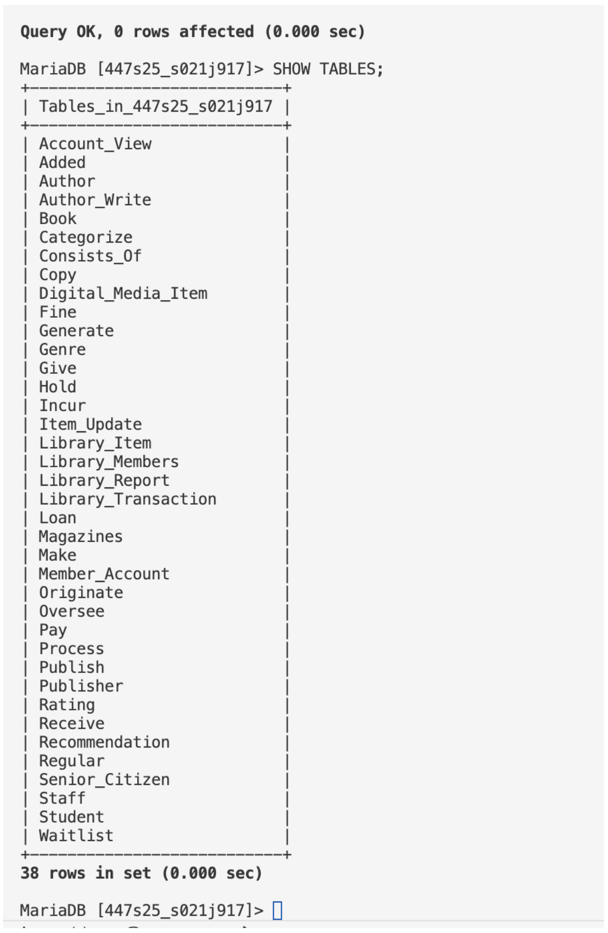
For the purpose of this document, it will also be put in this subsection.

- ASKNE - ASKNrEceive
- LMS - Library Management System
- SaaS - Software as a Service
- SQL - Structured Query Language
- TAs - Teaching Assistants
- CLI - Command Line Interface

2 Platform

ASKNE chose MariaDB because of its open-source nature, great performance, and scalability, which make it reliable for handling large amounts of data for libraries. It's also more accessible, since each team member in the group has an account for MariaDB on the cycle servers that has been set up by the TA's in this class. To manage version history and adaptability to teamwork, our team will be committing our scripts to GitHub so that everyone has access to the database that ASKNE created. A limitation is that there is no GUI to MariaDB on the cycle servers, so all commands and SQL statements will need to be run via the Command Line Interface (CLI).

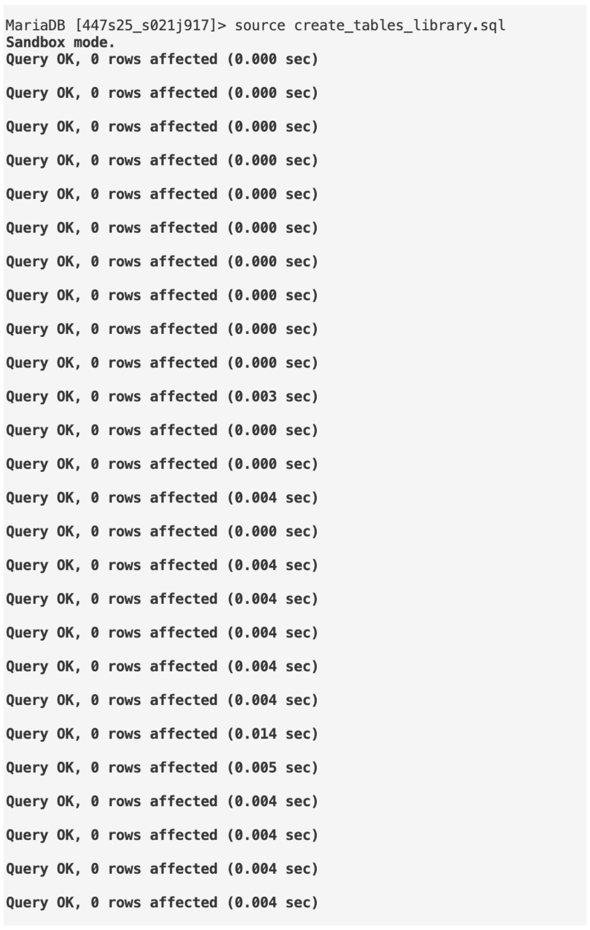
3 Database Creation



```
Query OK, 0 rows affected (0.000 sec)

MariaDB [447s25_s021j917]> SHOW TABLES;
+---------------------------+
| Tables_in_447s25_s021j917 |
+---------------------------+
| Account_View              |
| Added                     |
| Author                    |
| Author_Write              |
| Book                      |
| Categorize                |
| Consists_Of               |
| Copy                      |
| Digital_Media_Item        |
| Fine                      |
| Generate                  |
| Genre                     |
| Give                      |
| Hold                      |
| Incur                     |
| Item_Update               |
| Library_Item              |
| Library_Members           |
| Library_Report            |
| Library_Transaction       |
| Loan                      |
| Magazines                 |
| Make                      |
| Member_Account            |
| Originate                 |
| Oversee                   |
| Pay                       |
| Process                   |
| Publish                   |
| Publisher                 |
| Rating                    |
| Receive                   |
| Recommendation            |
| Regular                   |
| Senior_Citizen            |
| Staff                     |
| Student                   |
| Waitlist                  |
+---------------------------+
38 rows in set (0.000 sec)

MariaDB [447s25_s021j917]>
```



```
MariaDB [447s25_s021j917]> source create_tables_library.sql
Sandbox mode.
Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.003 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.004 sec)

Query OK, 0 rows affected (0.000 sec)

Query OK, 0 rows affected (0.004 sec)

Query OK, 0 rows affected (0.004 sec)

Query OK, 0 rows affected (0.004 sec)

Query OK, 0 rows affected (0.004 sec)

Query OK, 0 rows affected (0.004 sec)

Query OK, 0 rows affected (0.014 sec)

Query OK, 0 rows affected (0.005 sec)

Query OK, 0 rows affected (0.004 sec)

Query OK, 0 rows affected (0.004 sec)

Query OK, 0 rows affected (0.004 sec)

Query OK, 0 rows affected (0.004 sec)
```

4 Physical Schema

Refer to ***create_tables_library.sql*** in GitHub for the complete CREATE SQL Statements for all tables (or see below).

```
/*!999999\- enable the sandbox mode */
-- MariaDB dump 10.19  Distrib 10.6.18-MariaDB, for debian-linux-gnu (x86_64)
--
-- Host: mysql.eecs.ku.edu    Database: 447s25_s021j917
-- ------------------------------------------------------
-- Server version   10.6.18-MariaDB-0ubuntu0.22.04.1
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */
;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */
;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */
;
/*!40101 SET NAMES utf8mb4 */
;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */
;
```

```
/*!40103 SET TIME_ZONE='+00:00' */
;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */
;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */
;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */
;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */
;
--
-- Table structure for table `Copy`
--

DROP TABLE IF EXISTS Copy;
/*!40101 SET @saved_cs_client     = @@character_set_client */
;
/*!40101 SET character_set_client = utf8 */
;
CREATE TABLE Copy (
   -- Create the Copy table with its attributes.
   copy_id int(11) NOT NULL,
   status varchar(100) DEFAULT NULL,
   -- Status can initially be NULL.
   PRIMARY KEY (copy_id)
);
/*!40101 SET character_set_client = @saved_cs_client */
;
CREATE TABLE Author (
   -- Create Author with its attributes.
   author_id INT,
   name VARCHAR(30) NOT NULL,
   -- Name cannot be NULL.
   PRIMARY KEY (author_id)
);
-- CREATE TABLE Copy (
-- copy_id INT,
-- status VARCHAR(30) CHECK (status IN (‘in_stock’, ‘checked_out’)),
-- PRIMARY KEY (copy_id)
-- );
CREATE TABLE Genre (
   -- Create Genre table with its attributes.
   genre_id INT,
   genre_name VARCHAR(30) NOT NULL,
   -- The genre's name can't be NULL.
   PRIMARY KEY (genre_id)
);
CREATE TABLE Publisher (
   -- Create Publihser table with its attributes.
   publisher_id INT,
   name VARCHAR(30) NOT NULL,
   -- Name can't be NULL.
   PRIMARY KEY (publisher_id)
);
CREATE TABLE Library_Transaction (
   -- Create Library_Transaction and its attributes.
   transaction_id INT,
   checked_out_date DATE NOT NULL,
   -- The checked out date cannot be NULL.
   due_date DATE NOT NULL DEFAULT ADDDATE(checked_out_date, 14),
   -- Make the due date 14 days after the checked out date.
   return_date DATE,
   -- Can be NULL
   PRIMARY KEY (transaction_id),
   CHECK (
       -- Check that the return date is either NULL or after the checked out date.
       return_date >= checked_out_date
       OR return_date = NULL
   ),
   CHECK (due_date > checked_out_date) -- Check that the due date is after the
checked out date.
);
CREATE TABLE Fine (
   -- Create Fine and its attributes.
   fine_id INT,
   fine_amount DECIMAL(5, 2),
   fine_date DATE,
   fine_status VARCHAR(30),
   PRIMARY KEY (fine_id),
   CHECK(fine_status IN ("paid", "unpaid")) -- Check if the status is either paid
or unpaid.
);
CREATE TABLE Library_Item (
   -- Create Library_Item and its attributes.
   item_id INT,
   checked_out_status BOOLEAN NOT NULL DEFAULT 0,
   -- The default is 0 - not checked out.
   -- NOT USED
   current_inventory INT,
   overdue_status BOOLEAN NOT NULL DEFAULT 0,
   -- The default is 0 - not overdue.
   -- NOT USED
   PRIMARY KEY (item_id)
);
CREATE TABLE Book (
   -- Create Book and its attributes.
   item_id INT,
   author VARCHAR(30) NOT NULL,
   -- Author name can't be NULL.
   availability_status BOOLEAN NOT NULL DEFAULT 1,
   -- Default is that its available.
   -- NOT NEEDED?
   book_rating DECIMAL(3, 2),
   genre VARCHAR(30) NOT NULL,
   ISBN CHAR(16) NOT NULL,
   title VARCHAR(100) NOT NULL,
   publication_year YEAR NOT NULL,
   PRIMARY KEY (item_id),
   FOREIGN KEY (item_id) REFERENCES Library_Item (item_id) ON DELETE CASCADE ON
UPDATE CASCADE,
   -- Tied to foreign key, make sure to update/delete.
   UNIQUE (ISBN) -- ISBN must be unique.
);
CREATE TABLE Digital_Media_Item (
   -- Create Digital_Media_Item and its attributes.
   item_id INT,
   author VARCHAR(30) NOT NULL,
   availability_status BOOLEAN NOT NULL DEFAULT 1,
   -- Available by default.
   genre VARCHAR(30) NOT NULL,
   ISBN CHAR(16) NOT NULL,
   title VARCHAR(40) NOT NULL,
   publication_year YEAR NOT NULL,
   PRIMARY KEY(item_id),
   FOREIGN KEY (item_id) REFERENCES Library_Item (item_id) ON DELETE CASCADE ON
UPDATE CASCADE,
   -- Tied to foreign key, make sure to update/delete.
   UNIQUE (ISBN)
);
CREATE TABLE Magazine (
   -- Create Magazine and its attributes.
   item_id INT,
   author VARCHAR(30) NOT NULL,
   availability_status VARCHAR(30) NOT NULL DEFAULT 1,
   -- Available by default.
   genre VARCHAR(30) NOT NULL,
   ISSN CHAR(9) NOT NULL,
   issue_number INT NOT NULL,
   title VARCHAR(30) NOT NULL,
   publication_date DATE NOT NULL,
   PRIMARY KEY (item_id),
   FOREIGN KEY (item_id) REFERENCES Library_Item (item_id) ON DELETE CASCADE ON
UPDATE CASCADE,
   -- Tied to foreign key, make sure to update/delete.
   UNIQUE (ISSN) -- ISSN is unique.
);
CREATE TABLE Library_Report (
   -- Create Library_Report and its attributes.
   report_id INT,
   number_of_checkouts INT,
   report_date DATE NOT NULL,
   -- Report date can't be NULL.
   total_amount_owed_per_day DECIMAL(5, 2),
   total_amount_paid_per_day DECIMAL(5, 2),
   PRIMARY KEY (report_id)
);
CREATE TABLE Member_Account (
   -- Create Member_Account and its attributes.
   account_id INT,
   incurred_fees DECIMAL(5, 2),
   overdue_balance DECIMAL(5, 2),
   total_amount_paid DECIMAL(5, 2),
   PRIMARY KEY (account_id)
);
CREATE TABLE Staff (
   -- Create Staff and its attributes.
   staff_id INT,
   role VARCHAR(30) NOT NULL,
   -- They must have a role.
   name VARCHAR(50) NOT NULL,
   -- They must have a name.
   PRIMARY KEY (staff_id)
);
CREATE TABLE Library_Member (
   -- Create Library_Member and its attributes.
   member_id INT,
   book_limit INT,
   fee_type DECIMAL(3, 2),
   type_id INT,
   name VARCHAR(30),
   contact_information VARCHAR(30),
   account_status VARCHAR(30),
   PRIMARY KEY (member_id),
   CHECK (type_id IN (1, 2, 3)),
   -- Check that the type is either 1 (Regular), 2 (Student), or 3 (Senior
Citizen).
   CHECK (
       -- If Regular type, fee is 0.30. If Student type, fee is 0.20. If Senior
Citizen, fee is 0.15.
       fee_type = CASE
           WHEN type_id = 1 THEN 0.30
           WHEN type_id = 2 THEN 0.20
           WHEN type_id = 3 THEN 0.15
       END
   ),
   CHECK (
       -- If Regular type, borrowing limit is 5. If Student type, borrowing limit
is 10. If Senior Citizen, borrowing limit is 5.
       book_limit = CASE
           WHEN type_id = 1 THEN 5
           WHEN type_id = 2 THEN 10
           WHEN type_id = 3 THEN 5
       END
   )
);
CREATE TABLE Waitlist (
   -- Create Waitlist and its attributes.
   waitlist_id INT,
   request_date DATE NOT NULL,
   waitlist_status VARCHAR(30) NOT NULL,
   fulfilled_date DATE,
   PRIMARY KEY (waitlist_id),
   CHECK (
       waitlist_status IN ('on_hold', 'available_for_checkout') -- Status either on
hold or available for checkout.
   )
);
CREATE TABLE Regular (
   -- Create Regular and its attributes.
   member_id INT,
   available_limit INT NOT NULL DEFAULT 5,
   PRIMARY KEY (member_id),
   FOREIGN KEY (member_id) references Library_Member (member_id) ON DELETE CASCADE
ON UPDATE CASCADE -- Foreign key, so update/delete with parent table.
);
CREATE TABLE Student (
   -- Create Student and its attributes.
   member_id INT,
   available_limit INT NOT NULL DEFAULT 10,
   -- Default is 10.
   PRIMARY KEY (member_id),
   FOREIGN KEY (member_id) REFERENCES Library_Member (member_id) ON DELETE CASCADE
ON UPDATE CASCADE -- Foreign key, update/delete with parent table.
);
CREATE TABLE Senior_Citizen (
   -- Create Senior Citizen and its attributes.
   member_id INT,
   available_limit INT NOT NULL DEFAULT 5,
   -- Default is 5.
   PRIMARY KEY (member_id),
   FOREIGN KEY (member_id) REFERENCES Library_Member (member_id) ON DELETE CASCADE
ON UPDATE CASCADE -- Foreign key, update/delete with parent table.
);
CREATE TABLE Recommendation (
   -- Create Recommendation and its attributes.
   recommendation_id INT,
   PRIMARY KEY (recommendation_id)
);
CREATE TABLE Added (
   -- Create Added and its attributes.
   copy_id INT,
   waitlist_id INT,
   PRIMARY KEY (copy_id),
   FOREIGN KEY (copy_id) REFERENCES Copy (copy_id) ON DELETE CASCADE ON UPDATE
CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (waitlist_id) REFERENCES Waitlist (waitlist_id) ON DELETE CASCADE ON
UPDATE CASCADE
);
CREATE TABLE Hold (
   -- Create Hold and its attributes.
   member_id INT,
   copy_id INT,
   PRIMARY KEY (member_id, copy_id),
   FOREIGN KEY (member_id) REFERENCES Library_Member (member_id) ON DELETE CASCADE
ON UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (copy_id) REFERENCES Copy (copy_id) ON DELETE CASCADE ON UPDATE
CASCADE
);
CREATE TABLE Pay (
   -- Create Pay and its attributes.
   fine_id INT,
   member_id INT,
   amount_paid DECIMAL(5, 2) NOT NULL,
   -- If they paid, the amount can't be NULL.
   paid_date DATE NOT NULL,
   PRIMARY KEY (fine_id),
   FOREIGN KEY (fine_id) REFERENCES Fine (fine_id) ON DELETE CASCADE ON UPDATE
CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (member_id) REFERENCES Library_Member (member_id) ON DELETE CASCADE
ON UPDATE CASCADE
);
CREATE TABLE Incur (
   -- Create Incur and its attributes.
   transaction_id INT,
   fine_id INT,
   PRIMARY KEY (transaction_id, fine_id),
   FOREIGN KEY (transaction_id) REFERENCES Library_Transaction (transaction_id) ON
DELETE CASCADE ON UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (fine_id) REFERENCES Fine (fine_id) ON DELETE CASCADE ON UPDATE
CASCADE
);
CREATE TABLE Process (
   -- Create Process and its attributes.
   transaction_id INT,
   staff_id INT,
   PRIMARY KEY (transaction_id),
   FOREIGN KEY (transaction_id) REFERENCES Library_Transaction (transaction_id) ON
DELETE CASCADE ON UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (staff_id) REFERENCES Staff (staff_id) ON DELETE CASCADE ON UPDATE
CASCADE
);
CREATE TABLE Loan (
   -- Create Loan and its attributes.
   copy_id INT,
   transaction_id INT,
   PRIMARY KEY (copy_id, transaction_id),
   FOREIGN KEY (copy_id) REFERENCES Copy (copy_id) ON DELETE CASCADE ON UPDATE
CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (transaction_id) REFERENCES Library_Transaction (transaction_id) ON
DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE Make (
   -- Create Make and its attributes.
   transaction_id INT,
   member_id INT,
   PRIMARY KEY (transaction_id),
   FOREIGN KEY (transaction_id) REFERENCES Library_Transaction (transaction_id) ON
DELETE CASCADE ON UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (member_id) REFERENCES Library_Member (member_id) ON DELETE CASCADE
ON UPDATE CASCADE
);
CREATE TABLE Receive (
   -- Create Receiev and its attributes.
   recommendation_id INT,
   account_id INT,
   PRIMARY KEY (recommendation_id, account_id),
   FOREIGN KEY (recommendation_id) references Recommendation (recommendation_id) ON
DELETE CASCADE ON UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (account_id) REFERENCES Member_Account (account_id) ON DELETE
CASCADE ON UPDATE CASCADE
);
CREATE TABLE Consists_Of (
   -- Create Consists_Of and its attributes.
   recommendation_id INT,
   item_id INT,
   PRIMARY KEY (recommendation_id, item_id),
   FOREIGN KEY (recommendation_id) REFERENCES Recommendation (recommendation_id) ON
DELETE CASCADE ON UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (item_id) REFERENCES Library_Item (item_id) ON DELETE CASCADE ON
UPDATE CASCADE
);
CREATE TABLE Account_View (
   -- Create Account_View and its attributes.
   member_id INT,
   account_id INT,
   timestamp DATE NOT NULL,
   PRIMARY KEY (member_id),
   FOREIGN KEY (member_id) REFERENCES Library_Member (member_id) ON DELETE CASCADE
ON UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (account_id) REFERENCES Member_Account (account_id) ON DELETE
CASCADE ON UPDATE CASCADE
);
CREATE TABLE Generate (
   -- Create Generate and its attributes.
   report_id INT,
   staff_id INT,
   PRIMARY KEY (report_id),
   FOREIGN KEY (report_id) REFERENCES Library_Report (report_id) ON DELETE CASCADE
ON UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (staff_id) REFERENCES Staff (staff_id) ON DELETE CASCADE ON UPDATE
CASCADE
);
CREATE TABLE Rating (
   -- Create Rating and its attributes.
   member_id INT,
   item_id INT,
   stars_given INT NOT NULL,
   PRIMARY KEY (member_id, item_id),
   FOREIGN KEY (member_id) REFERENCES Library_Member (member_id) ON DELETE CASCADE
ON UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (item_id) REFERENCES Library_Item (item_id) ON DELETE CASCADE ON
UPDATE CASCADE,
   CHECK (
       stars_given >= 0
       AND stars_given <= 5
   )
);
CREATE TABLE Oversee (
   -- Create Oversee and its attributes.
   member_id INT,
   staff_id INT,
   PRIMARY KEY (member_id),
   FOREIGN KEY (staff_id) REFERENCES Staff (staff_id) ON DELETE CASCADE ON UPDATE
CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (member_id) REFERENCES Library_Member (member_id) ON DELETE CASCADE
ON UPDATE CASCADE
);
CREATE TABLE Item_Update (
   -- Create Item_Update and its attributes.
   copy_id INT,
   staff_id INT,
   PRIMARY KEY (copy_id),
   FOREIGN KEY (copy_id) REFERENCES Copy (copy_id) ON DELETE CASCADE ON UPDATE
CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (staff_id) REFERENCES Staff (staff_id) ON DELETE CASCADE ON UPDATE
CASCADE
);
CREATE TABLE Originate (
   -- Create Originate and its attributes.
   copy_id INT,
   item_id INT,
   PRIMARY KEY (copy_id),
   FOREIGN KEY (copy_id) REFERENCES Copy (copy_id) ON DELETE CASCADE ON UPDATE
CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (item_id) REFERENCES Library_Item (item_id) ON DELETE CASCADE ON
UPDATE CASCADE
);
CREATE TABLE Publish (
   -- Create Publish and its attributes.
   item_id INT,
   publisher_id INT,
   PRIMARY KEY (item_id),
   FOREIGN KEY (item_id) REFERENCES Library_Item (item_id) ON DELETE CASCADE ON
UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (publisher_id) REFERENCES Publisher (publisher_id) ON DELETE CASCADE
ON UPDATE CASCADE
);
CREATE TABLE Author_Write (
   -- Create Author_Write and its attributes.
   item_id INT,
   author_id INT,
   PRIMARY KEY (item_id),
   FOREIGN KEY (item_id) REFERENCES Library_Item (item_id) ON DELETE CASCADE ON
UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (author_id) REFERENCES Author (author_id) ON DELETE CASCADE ON
UPDATE CASCADE
);
CREATE TABLE Categorize (
   -- Create Categorize and its attributes.
   item_id INT,
   genre_id INT,
   PRIMARY KEY (item_id),
   FOREIGN KEY (item_id) REFERENCES Library_Item (item_id) ON DELETE CASCADE ON
UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (genre_id) REFERENCES Genre (genre_id) ON DELETE CASCADE ON UPDATE
CASCADE
);
CREATE TABLE Give (
   -- Create Give and its attributes.
   recommendation_id INT,
   member_id INT,
   PRIMARY KEY (recommendation_id),
   FOREIGN KEY (recommendation_id) REFERENCES Recommendation (recommendation_id) ON
DELETE CASCADE ON UPDATE CASCADE,
   -- Foreign key, update/delete with parent table.
   FOREIGN KEY (member_id) REFERENCES Library_Member (member_id) ON DELETE CASCADE
ON UPDATE CASCADE
);
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */
;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */
;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */
;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */
;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */
;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */
;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */
```

```
;
-- Dump completed on 2025-03-29 17:16:21
```

5 Data Population

Refer to ***populate_library.sql*** in GitHub for the complete INSERT SQL Statements for all tables.

6 Table Contents

Refer to ***table_output.csv*** in GitHub for the complete output of all printed tables, through the SELECT * FROM Table_Name command, and ***table_output.sql*** for the SQL query commands that gave this output.

7 Functionality Testing

Refer to ***test_queries.sql*** and ***query_output.csv*** in GitHub for the complete SQL query statements and outputs for all functionality testing.

8 Appendices

Refer to *01 - Project Plan/Vision* for any additional information on the scope and definition of the project.

Refer to *07 - Glossary* for any additional information regarding abbreviations and terms.

9 GitHub Repository Management

All members of the ASKNrEceive team will regularly manage, update, and commit to the GitHub Repository. The repository will be publicly available for viewing and accessing here: Library Database Management Project.

**** Note: All our Project Meeting Logs will be housed in the GitHub Repository on the Wiki Page. Please reference it as needed. Our Team Profile is also on the Wiki Page.**