Objective: You have designed the logical blueprint for your database. Now, create a physical database based on this logical design and populate it with meaningful data. Although the deliverable (artifact) for this part of the project will be a Canvas submission, you must ensure you have a fully functional database. This database should be rigorously tested to verify that it meets all the requirements outlined in your project proposal.

Introduction [5 points].

Project Overview: This database is intended to maintain records of inventory and services for a small town's library. This includes all typical operations expected of a small town's library. The system will cover a variety of library operations such as renting different kinds of literature, managing memberships, handling transactions, and other various actions users and employees take. The project will include designing an Entity Relationship diagram, then mapping it to a real schema and implementation using MariaDB. Then creating a simple user interface that can handle the various queries needed for customers and employees of the library. Finally, wrapping up with some form of testing, possibly unit and integration tests, and documentation/instructions.

*Scope***:** Items included in the scope: loanable items (Books, DVDs, CDs, Magazines), in library services (computer access, newspaper, microfiche), customers, and librarians. It will include a borrowing/returning system with limits, fees, and reservations. Different user interfaces will be created for staff and clients, allowing book checkouts, catalog searches, and overdue fee tracking. The database will be implemented in MariaDB and will include queries for generating various reports. The system will not include a fully developed UI, AI-based recommendations, external payment integration, or multi-library support.

*Glossary***:** Update your glossary with any new terms or concepts relevant to relational database design. The introduction section is for continuity; the *italic* parts are subsections.

- **ISBN** – International Standard Book Number, a unique identifier for books.
- **Loanable Items** – Books, DVDs, CDs, and magazines that members can borrow.
- **Reservation** – The process of placing a hold on a currently checked-out item.
- **Overdue Fee** – A fine charged to a member for returning an item late.
- **MariaDB** – The relational database management system used for implementation.
- **ISSN** – International Standard Serial Number, a unique code used to identify journals, magazines, and newspapers
- **DDC** – the Dewey Decimal System, a library classification system that organizes books by subject

Choose Your Platform [5 points]: You can use the EECS Cycle servers with MariaDB, or if your team prefers, select another database product (like MySQL, PostgreSQL, or SQL Server). Be sure to explain your choice in your report. What factors influenced your decision? Familiarity with the system? Specific features? Potential limitations?

While we are hosting the database on our own infrastructure, we chose to use MariaDB as it is the same software that is used on the EECS cycle servers. While this was the primary factor in our decision, we also chose it because it is simple and easy to use like MySQL, but has a few performance improvements. It was also the first database software we found that had a docker container which made setup easy and was compatible with the database tool DBeaver which we plan to use for this project.

Create Your Database: Create a physical database schema using your DDL scripts. Ensure all tables are correctly created with appropriate constraints (primary keys, foreign keys, etc.). Organize all SQL scripts (DDL, data population, queries, reports) clearly, name them descriptively, and include comments explaining their functions.

Located in the scripts folder on the github

Print Your Physical Schema [10 points]: Print the SQL DDL statements that you used to create the physical schema.

```
CREATE DATABASE `eecs` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_uca1400_ai_ci */;
```

```
-- eecs.CDs definition

CREATE TABLE `CDs` (
  `itemID` int(11) NOT NULL,
  `title` varchar(100) DEFAULT NULL,
  `performingArtists` varchar(100) DEFAULT NULL,
  `distributor` varchar(100) DEFAULT NULL,
  `year` date DEFAULT NULL,
  `stock` int(11) NOT NULL,
  `shelved` tinyint(1) NOT NULL,
  PRIMARY KEY (`itemID`),
  CONSTRAINT `CDs_Rentable_Item_FK` FOREIGN KEY (`itemID`) REFERENCES
`rentableItem` (`itemID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```
-- eecs.DVDs definition

CREATE TABLE `DVDs` (
  `itemID` int(11) NOT NULL,
  `title` varchar(100) DEFAULT NULL,
  `director` varchar(100) DEFAULT NULL,
  `publisher` varchar(100) DEFAULT NULL,
  `year` date DEFAULT NULL,
  `stock` int(11) NOT NULL,
  `shelved` tinyint(1) NOT NULL,
  PRIMARY KEY (`itemID`),
  CONSTRAINT `DVDs_Rentable_Item_FK` FOREIGN KEY (`itemID`) REFERENCES
`rentableItem` (`itemID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```sql
-- eecs.books definition

CREATE TABLE `books` (
  `itemID` int(11) NOT NULL,
  `ISBN` bigint(20) unsigned NOT NULL,
  `deweyDecimal` varchar(100) DEFAULT NULL,
  `title` varchar(100) DEFAULT NULL,
  `author` varchar(100) DEFAULT NULL,
  `genre` varchar(100) DEFAULT NULL,
  `year` date DEFAULT NULL,
  `publisher` varchar(100) DEFAULT NULL,
  `stock` int(11) NOT NULL,
  `shelved` tinyint(1) NOT NULL,
  PRIMARY KEY (`itemID`),
  CONSTRAINT `Books_Rentable_Item_FK` FOREIGN KEY (`itemID`) REFERENCES
`rentableItem` (`itemID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```sql
-- eecs.borrows definition

CREATE TABLE `borrows` (
  `loanID` int(11) NOT NULL,
  `itemID` int(11) NOT NULL,
  `userID` int(11) NOT NULL,
  `loanDate` date NOT NULL,
  `dueDate` date DEFAULT NULL,
  `returnDate` date DEFAULT NULL,
  PRIMARY KEY (`loanID`),
  KEY `Borrows_Rentable_Item_FK` (`itemID`),
  KEY `Borrows_Member_FK` (`userID`),
  CONSTRAINT `Borrows_Member_FK` FOREIGN KEY (`userID`) REFERENCES `member`
(`userID`),
  CONSTRAINT `Borrows_Rentable_Item_FK` FOREIGN KEY (`itemID`) REFERENCES
`rentableItem` (`itemID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```sql
-- eecs.computer definition

CREATE TABLE `computer` (
  `computerID` int(11) NOT NULL,
  PRIMARY KEY (`computerID`),
  CONSTRAINT `Computer_Rentable_Device_FK` FOREIGN KEY (`computerID`)
REFERENCES `rentableDevice` (`deviceID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```sql
-- eecs.`contains` definition

CREATE TABLE `contains` (
  `ID` int(11) NOT NULL,
  `ISSN` varchar(100) NOT NULL,
  PRIMARY KEY (`ID`,`ISSN`),
  KEY `Contains_Newspaper_FK` (`ISSN`),
  CONSTRAINT `Contains_Microfiche_FK` FOREIGN KEY (`ID`) REFERENCES
`microfiche` (`deviceID`),
  CONSTRAINT `Contains_Newspaper_FK` FOREIGN KEY (`ISSN`) REFERENCES
`newspaper` (`ISSN`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```sql
-- eecs.customer definition

CREATE TABLE `customer` (
  `userID` int(11) NOT NULL,
  `fees` float DEFAULT NULL,
  `status` varchar(100) NOT NULL,
  `membershipType` varchar(100) DEFAULT NULL,
  `libraryCardNumber` bigint(20) NOT NULL,
  PRIMARY KEY (`userID`),
  CONSTRAINT `Customer_Member_FK` FOREIGN KEY (`userID`) REFERENCES
`member` (`userID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```
-- eecs.magazines definition

CREATE TABLE `magazines` (
  `itemID` int(11) NOT NULL,
  `title` varchar(100) DEFAULT NULL,
  `issueNumber` int(11) DEFAULT NULL,
  `publicationDate` date DEFAULT NULL,
  `stock` int(11) NOT NULL,
  `shelved` tinyint(1) NOT NULL,
  PRIMARY KEY (`itemID`),
  CONSTRAINT `Magazines_Rentable_Item_FK` FOREIGN KEY (`itemID`) REFERENCES
`rentableItem` (`itemID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```
-- eecs.`member` definition

CREATE TABLE `member` (
  `userID` int(11) NOT NULL,
  `name` varchar(100) NOT NULL,
  `contactInfo` varchar(100) DEFAULT NULL,
  `age` int(11) DEFAULT NULL,
  PRIMARY KEY (`userID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```
-- eecs.microfiche definition

CREATE TABLE `microfiche` (
  `deviceID` int(11) NOT NULL,
  PRIMARY KEY (`deviceID`),
  CONSTRAINT `Microfiche_Rentable_Device_FK` FOREIGN KEY (`deviceID`)
REFERENCES `rentableDevice` (`deviceID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```sql
-- eecs.newspaper definition

CREATE TABLE `newspaper` (
  `ISSN` varchar(100) NOT NULL,
  `publisher` varchar(100) NOT NULL,
  `date` date NOT NULL,
  PRIMARY KEY (`ISSN`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```sql
-- eecs.rentableDevice definition

CREATE TABLE `rentableDevice` (
  `deviceID` int(11) NOT NULL,
  `available` tinyint(1) NOT NULL DEFAULT 1,
  PRIMARY KEY (`deviceID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```sql
-- eecs.rentableItem definition

CREATE TABLE `rentableItem` (
  `itemID` int(11) NOT NULL,
  `loanable` tinyint(1) DEFAULT NULL,
  `ageRating` int(11) DEFAULT NULL,
  PRIMARY KEY (`itemID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```sql
-- eecs.staffMember definition

CREATE TABLE `staffMember` (
  `userID` int(11) NOT NULL,
  `position` varchar(100) NOT NULL,
  `staffID` int(11) NOT NULL,
  PRIMARY KEY (`userID`),
  CONSTRAINT `Staff_member_Member_FK` FOREIGN KEY (`userID`) REFERENCES
`member` (`userID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

```
-- eecs.uses definition

CREATE TABLE `uses` (
  `userID` int(11) NOT NULL,
  `deviceID` int(11) NOT NULL,
  PRIMARY KEY (`userID`,`deviceID`),
  KEY `Uses_Rentable_Device_FK` (`deviceID`),
  CONSTRAINT `Uses_Member_FK` FOREIGN KEY (`userID`) REFERENCES `member`
(`userID`),
  CONSTRAINT `Uses_Rentable_Device_FK` FOREIGN KEY (`deviceID`) REFERENCES
`rentableDevice` (`deviceID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_uca1400_ai_ci;
```

Data Population. Use tools or scripts to create realistic data matching the type of information your database will hold. Utilize public datasets where available, or manually enter data if your project is small. Use bulk data loading utilities for larger datasets. Validate your data for accuracy and consistency with your design.

Printing Table Contents [10 points]. Once the tables are populated, print the contents of each table to provide a quick overview of the table contents, allowing examination of data size and composition. You may use the following SQL command for each table:

Located in docs\database_contents.txt in the github