Choose Your Platform:
All of us decided to download MySQL to implement our database. Since some of our members had familiarity with MySQL workbench, it would be easier to run our code on that platform.

Create your Database:
We downloaded MySQL and input our DDL scripts from our Schema Documentation. This was how we set up our database.

```
CREATE TABLE Author (
	AuthorID INT PRIMARY KEY AUTO_INCREMENT,
	FirstName VARCHAR(255) NOT NULL,
	LastName VARCHAR(255) NOT NULL
);

CREATE TABLE Item (
	ItemID INT PRIMARY KEY AUTO_INCREMENT,
	Title VARCHAR(255) NOT NULL,
	AvailabilityStatus BOOLEAN NOT NULL,
	PublicationDate DATE,
	AuthorID INT,
	Type VARCHAR(10) NOT NULL CHECK (Type IN ('Physical', 'Digital')),
	Genre VARCHAR(255),
	FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID)
);

CREATE TABLE Book (
	ItemID INT Primary Key,
	ISBN VARCHAR(255) NOT NULL,
	FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);

CREATE TABLE Magazine (
	ItemID INT Primary Key,
	IssueNumber INT NOT NULL,
	FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);

CREATE TABLE DigitalMedia (
	ItemID INT Primary Key,
	MediaType VARCHAR(20) NOT NULL CHECK (MediaType IN ('Audiobook', 'Movie',
'Podcast', 'Pictures')),
	FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);
```

```
CREATE TABLE Creates (
   AuthorID INT,
   ItemID INT,
   PRIMARY KEY (AuthorID, ItemID),
   FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID),
   FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);

CREATE TABLE Client(
        ClientID INT PRIMARY KEY AUTO_INCREMENT,
        FirstName VARCHAR(255) NOT NULL,
        LastName VARCHAR(255) NOT NULL,
        PhoneNumber CHAR (10) NOT NULL,
        AccountStatus VARCHAR(20) NOT NULL CHECK (AccountStatus IN ('Active',
'Suspended', 'Pending Deletion')),
        MembershipType VARCHAR(10) NOT NULL CHECK (MembershipType IN ('Regular',
'Student', 'Senior'))
);

CREATE TABLE Loans(
        LoanID INT PRIMARY KEY AUTO_INCREMENT,
        StaffID INT,
        ItemID INT,
        LoanDate DATE NOT NULL,
        DueDate DATE NOT NULL,
        ReturnDate DATE,
        LateFee DECIMAL(10,2),
        FOREIGN KEY (StaffID) REFERENCES Staff(StaffID),
        FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);

CREATE TABLE Reserves(
        ReservationID INT PRIMARY KEY AUTO_INCREMENT,
        ClientID INT,
        ItemID INT,
        ReservationDate DATE NOT NULL,
        ReservationDueDate DATE NOT NULL,
        FOREIGN KEY (ClientID) REFERENCES Client(ClientID),
        FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);

CREATE TABLE Manages(
        ClientID INT PRIMARY KEY,
        StaffID INT,
FOREIGN KEY (StaffID) REFERENCES Staff(StaffID),
	FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);

CREATE TABLE Staff (
	StaffID INT PRIMARY KEY AUTO_INCREMENT,
	FirstName VARCHAR(100) NOT NULL,
	LastName VARCHAR(100) NOT NULL,
	PhoneNumber CHAR(10) NOT NULL,
	JobTitle VARCHAR(10) NOT NULL CHECK (JobTitle IN ('Employee', 'Manager',
'Owner'))
);
```

Data Population:
By using the language learning model, ChatGPT, and asking it to generate data in the right format tailored for our database schema, we were able to quickly generate contrived but realistic data for our database. However, for some items that require unique values, ChatGPT's memory is not very good and oftentimes will generate duplicates, so we made sure to take some time to validate each generation of data.

Functionality Testing:
We tested our database by using verification queries, which are just queries that implement the use of join, select, from, and other keywords to test the values returned by the database. We created queries for returning the information about books, magazines, and digital media. We then continued by making queries for returning the name of who reserved what item title.

Test and Validate:
To begin testing, we started with some SQL queries for relations such as Item joined with Book and Author(s) Creates Item(s).
Next, we tested insertion and deletion. We made sure our insertions worked as intended, with cases like inputting something with a null value that shouldn't be null wouldn't work, and making a deletion in a table that had dependencies did not work unless we deleted those first.
Finally, we made more detailed queries such as finding fees for specific names.

Documentation:
How you set up your database:
We set up our database by downloading MySQL and running our DDL script on it.

Problems faced:
DDL syntax for SQL -
We typed up our DDL script in a google document. This seemed fine until we realized that the single quotation marks that google documents uses defaulted to curly quotation marks, which is different from the straight quotation marks that the DDL syntax interprets for strings. This gave us a column reference error because the strings we wanted to use as domain restrictions were

considered actual attributes from tables that did not exist. We went and replaced all the curly quotations with straight quotations and this issue was fixed.
A similar syntax issue arose when we were inserting book titles, because some of them had apostrophes, except this time, ChatGPT generated them as straight single quotation marks, so we had to put double quotes around the titles to maintain the apostrophes.
We also mistakenly used some keywords like ‘role’ for attributes in our tables. This gave us errors until we replaced the attribute name for a different one.
Database Validation -
We found an oversight where Loans does not contain any reference to the Client that the Item is being loaned to. This made it difficult to pinpoint things such as “Who needs to pay a fee?” and “Who was this book loaned to?”, so we added a ClientID attribute, a foreign key reference to the ClientID in the Client relation, knowing that this was a loose fix that made the relation somewhat large. With more time and better insight, we would have been able to adjust our relations in a way that better normalized it for our purposes.

Where you got your data and how you imported it:
We got our data by generating, verifying, and fixing items we got from a language learning model, ChatGPT. Thanks to how we asked it to format our information, we were able to easily import it in either a CSV port because MySQL supports CSV import, or a standard bulk INSERT.

How you tested your database and the results:
We tested our database by going through SQL queries for common relations, insertion and deletion, and some more detailed queries regarding specific names and/or fees. Our intended results were confirmed to work through the terminal output of MySQL. The results shown were that a database was successfully created with the following tables: Author, Item, Book, Magazine, Digital Media, Client, Staff, Loans, Manages, Reserves, Creates. All the attributes functioned as expected as shown by the queries written in the verification tab.

Organize your code:
We have separated our inputs and queries and verifications into various tabs in a google word document during our time working as a group for Part 5. These tabs have been separated into individual documents in the github.

Provide Access:
Since we did not use the EECS Cycle Servers, we are scheduling for a demo using a local MySQL server hosted on a laptop.