Assignment On SQL:

Assignment 1:

_Analyze a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram reflects proper normalization up to the third normal form

Let business scenario:

A small online bookstore wants to manage its data efficiently. The store sells books and also has a membership program for customers. Here's a detailed description of their requirements

1. **Books**: Each book has a unique id, title, author, genre, and price. A book can belong to multiple genres.
2. **Customers**: Each customer has a unique customer ID, name, email, phone number, and address. Customers can purchase multiple books.
3. **Orders**: Each order has a unique order ID, order date, and is associated with one customer. An order can contain multiple books.
4. **Memberships**: Each membership has a unique membership ID, type (e.g., Silver, Gold, Platinum), start date, and end date. A customer can have only one membership at a time, but memberships can be renewed.
5. **Authors**: Each author has a unique author ID, name, and biography. An author can write multiple books.

Entities and Attributes:

1. **Book**:

- BookID (Primary Key)
- Title
- Price

2. **Genre**:
    - GenreID (Primary Key)
    - GenreName
3. **Customer**:
    - CustomerID (Primary Key)
    - Name
    - Email
    - Phone
    - Address
4. **Order**:
    - OrderID (Primary Key)
    - OrderDate
    - CustomerID (Foreign Key)
5. **OrderDetails** (to handle many-to-many relationship between Order and Book):
    - OrderID (Primary Key, Foreign Key)
    - Bookid(Primary Key, Foreign Key)
    - Quantity
6. **Membership**:
    - MembershipID (Primary Key)
    - Type
    - StartDate

- EndDate
- CustomerID (Foreign Key)

7. **Author**:
   - AuthorID (Primary Key)
   - Name
   - Biography

ER Diagram:

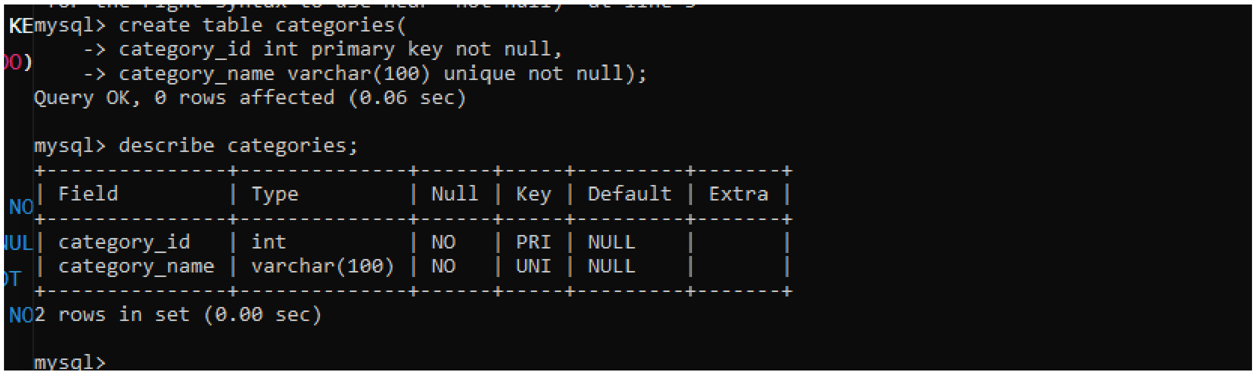
Let's now create the ER diagram.

1. **Entities** are represented by rectangles.
2. **Attributes** are represented by ovals connected to their entity.
3. **Relationships** are represented by diamonds.
4. **Primary Keys** are underlined.

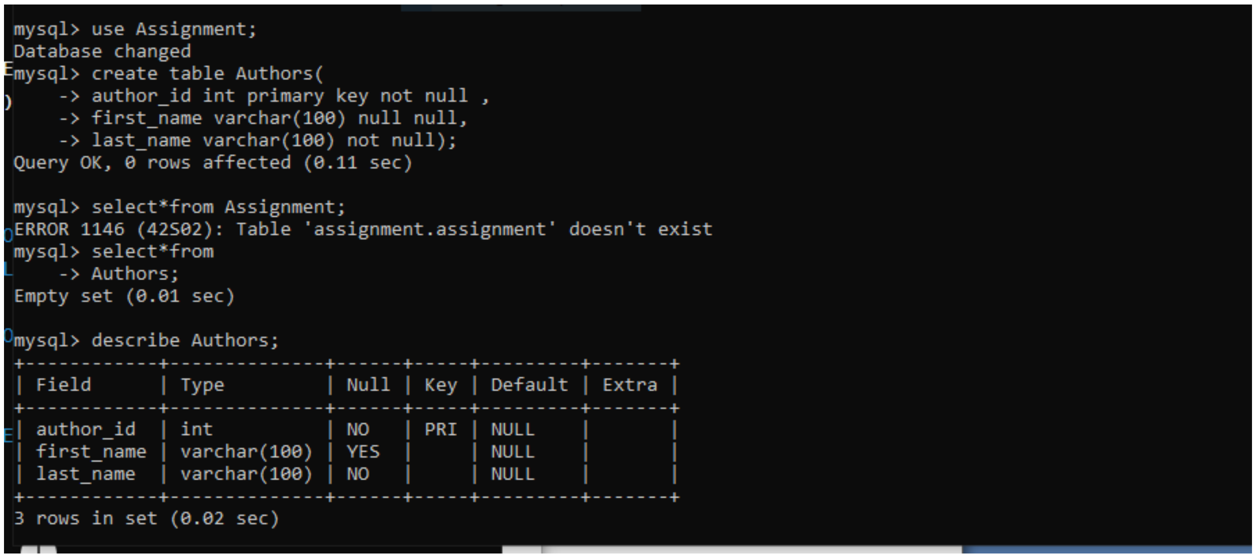
Assignment 2:

Design a database schema for a library system, including tables, fields, and constraints like NOT NULL, UNIQUE, and CHECK. Include primary and foreign keys to establish relationships between tables.

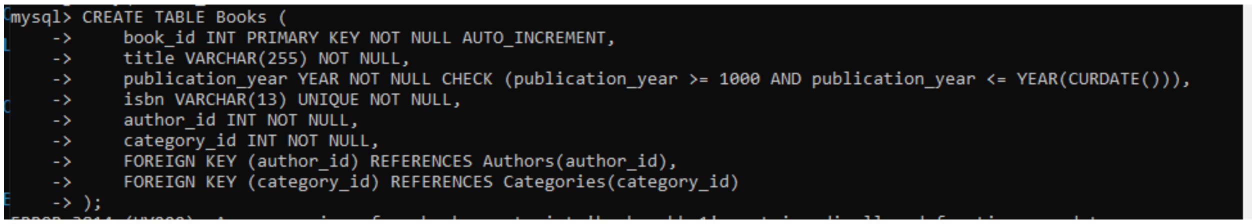
ere's a detailed database schema design for a library system. This schema includes tables for books, authors, patrons (library users), loans, and categories. Each table includes appropriate fields and constraints such as NOT NULL, UNIQUE, and CHECK, along with primary and foreign keys to establish relationships.



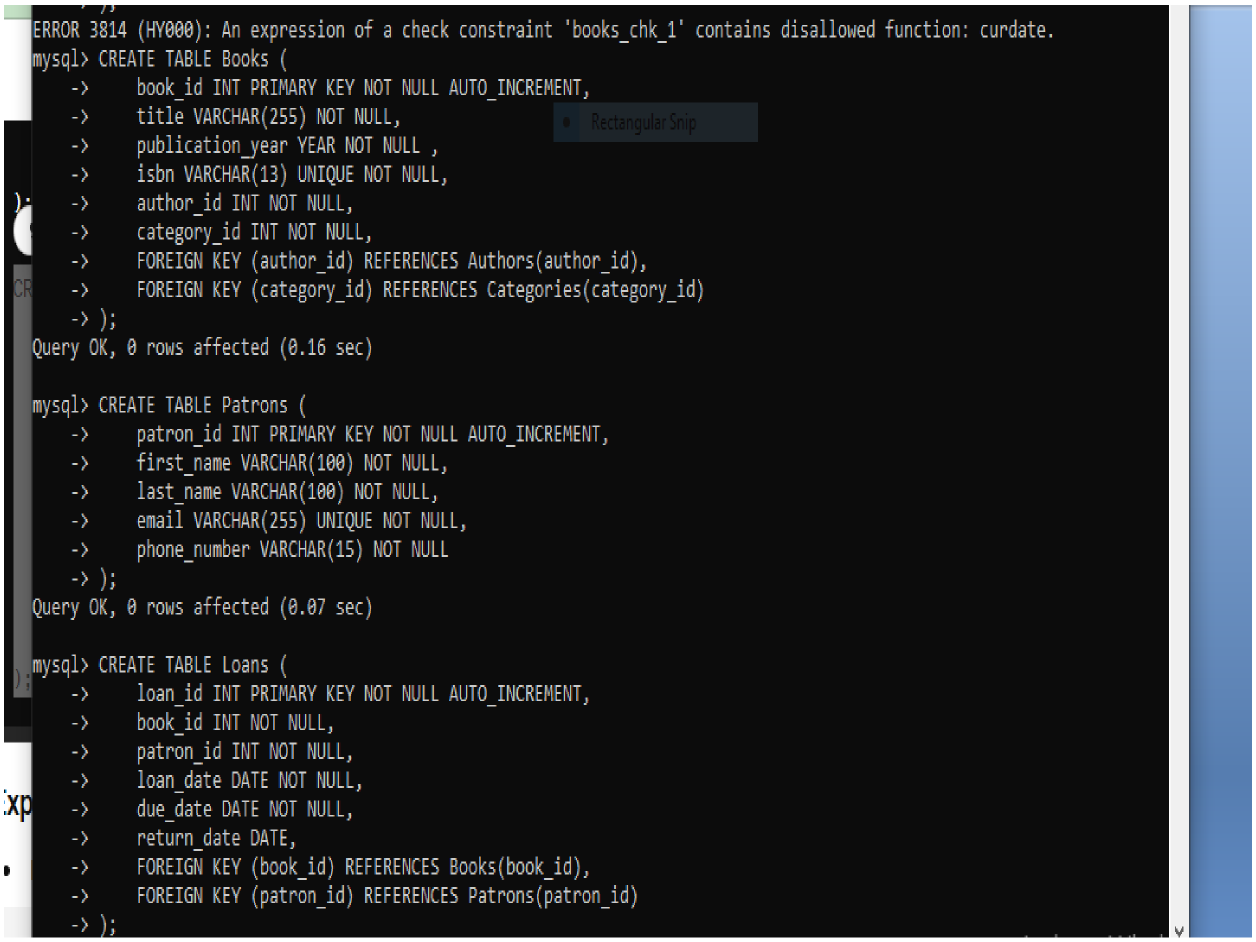
```
mysql> create table categories(
    -> category_id int primary key not null,
    -> category_name varchar(100) unique not null);
Query OK, 0 rows affected (0.06 sec)

mysql> describe categories;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| category_id   | int          | NO   | PRI | NULL    |       |
| category_name | varchar(100) | NO   | UNI | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```



```
mysql> use Assignment;
Database changed
mysql> create table Authors(
    -> author_id int primary key not null ,
    -> first_name varchar(100) null null,
    -> last_name varchar(100) not null);
Query OK, 0 rows affected (0.11 sec)

mysql> select*from Assignment;
ERROR 1146 (42S02): Table 'assignment.assignment' doesn't exist
mysql> select*from
    -> Authors;
Empty set (0.01 sec)

mysql> describe Authors;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| author_id  | int          | NO   | PRI | NULL    |       |
| first_name | varchar(100) | YES  |     | NULL    |       |
| last_name  | varchar(100) | NO   |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```



```
mysql> CREATE TABLE Books (
    ->     book_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    ->     title VARCHAR(255) NOT NULL,
    ->     publication_year YEAR NOT NULL CHECK (publication_year >= 1000 AND publication_year <= YEAR(CURDATE())),
    ->     isbn VARCHAR(13) UNIQUE NOT NULL,
    ->     author_id INT NOT NULL,
    ->     category_id INT NOT NULL,
    ->     FOREIGN KEY (author_id) REFERENCES Authors(author_id),
    ->     FOREIGN KEY (category_id) REFERENCES Categories(category_id)
    -> );
```



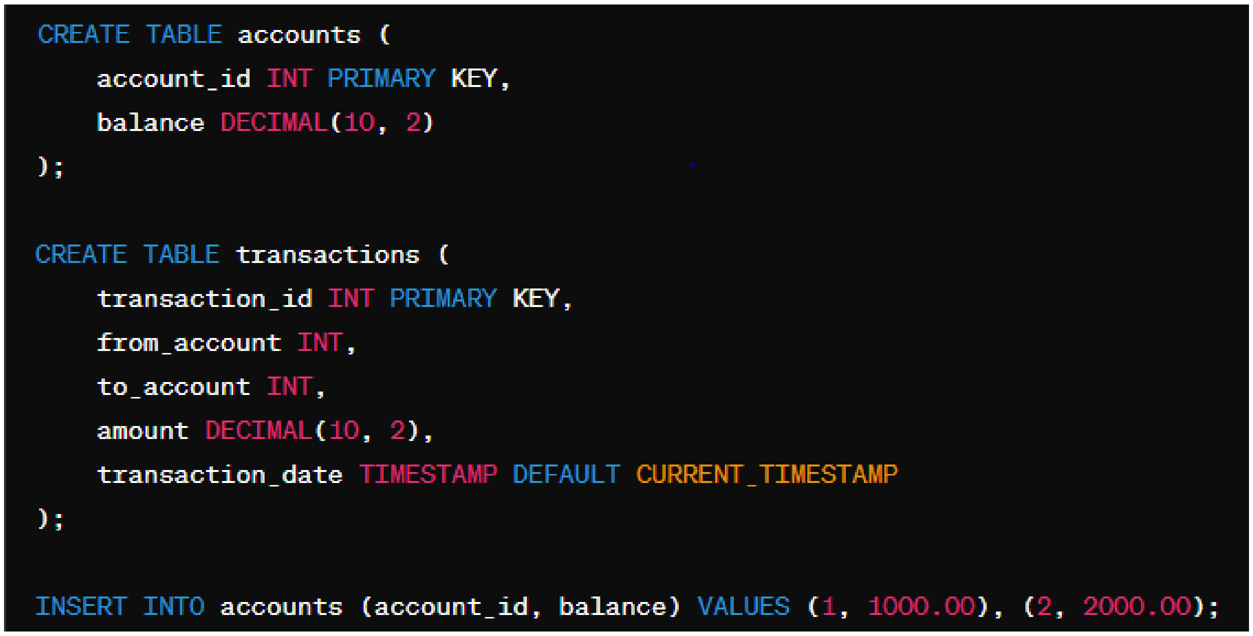
```
ERROR 3814 (HY000): An expression of a check constraint 'books_chk_1' contains disallowed function: curdate.
mysql> CREATE TABLE Books (
    ->     book_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    ->     title VARCHAR(255) NOT NULL,
    ->     publication_year YEAR NOT NULL ,
    ->     isbn VARCHAR(13) UNIQUE NOT NULL,
    ->     author_id INT NOT NULL,
    ->     category_id INT NOT NULL,
    ->     FOREIGN KEY (author_id) REFERENCES Authors(author_id),
    ->     FOREIGN KEY (category_id) REFERENCES Categories(category_id)
    -> );
Query OK, 0 rows affected (0.16 sec)

mysql> CREATE TABLE Patrons (
    ->     patron_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    ->     first_name VARCHAR(100) NOT NULL,
    ->     last_name VARCHAR(100) NOT NULL,
    ->     email VARCHAR(255) UNIQUE NOT NULL,
    ->     phone_number VARCHAR(15) NOT NULL
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql> CREATE TABLE Loans (
    ->     loan_id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    ->     book_id INT NOT NULL,
    ->     patron_id INT NOT NULL,
    ->     loan_date DATE NOT NULL,
    ->     due_date DATE NOT NULL,
    ->     return_date DATE,
    ->     FOREIGN KEY (book_id) REFERENCES Books(book_id),
    ->     FOREIGN KEY (patron_id) REFERENCES Patrons(patron_id)
    -> );
```

Assignment 3:

Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.

ACID stands for Atomicity, Consistency, Isolation, and Durability. These properties ensure reliable processing of database transactions.

1. **Atomicity**: Ensures that all operations within a transaction are completed successfully. If any part of the transaction fails, the entire transaction is rolled back, leaving the database in its previous state.
2. **Consistency**: Ensures that a transaction brings the database from one valid state to another, maintaining database invariants. The database remains in a consistent state before and after the transaction.
3. **Isolation**: Ensures that transactions are executed in isolation from each other. Changes made by one transaction are not visible to other transactions until they are committed, preventing concurrent transactions from interfering with each other.
4. **Durability:** Ensures that once a transaction has been committed, it remains committed even in the event of a system failure. The changes made by the transaction are permanently stored in the database.



```
CREATE TABLE accounts (
    account_id INT PRIMARY KEY,
    balance DECIMAL(10, 2)
);

CREATE TABLE transactions (
    transaction_id INT PRIMARY KEY,
    from_account INT,
    to_account INT,
    amount DECIMAL(10, 2),
    transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO accounts (account_id, balance) VALUES (1, 1000.00), (2, 2000.00);
```

```
-- Start the transaction
BEGIN TRANSACTION;

-- Lock the rows for the two accounts
SELECT balance FROM accounts WHERE account_id = 1 FOR UPDATE;
SELECT balance FROM accounts WHERE account_id = 2 FOR UPDATE;

-- Perform the transfer
UPDATE accounts SET balance = balance - 100.00 WHERE account_id = 1;
UPDATE accounts SET balance = balance + 100.00 WHERE account_id = 2;

-- Insert transaction record
```

INSERT INTO transactions (transaction_id, from_account, to_account, amount)

VALUES (1, 1, 2, 100.00);

Write SQL statements to CREATE a new database and tables that reflect the library schema you designed earlier. Use ALTER statements to modify the table structures and DROP statements to remove a redundant table

ALTER TABLE Books ADD COLUMN Genre VARCHAR(50);

Modifying a coluam in the ‘member table’

ALTER TABLE Books ADD COLUMN Genre VARCHAR(50);

ALTER TABLE Members MODIFY COLUMN Email VARCHAR(255) NOT NULL;

Dropping a redundant table

Let's assume we initially created a **Publishers** table, but later we decided it's redundant because we included the publisher information directly in the **Books** table.

DROP TABLE Publishers;

This set of SQL statements covers the creation, modification, and removal of database tables to reflect a basic library management system.

Assignment 2:

Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDEX statement to remove the index and analyze the impact on query execution.

Suppose we have employee table :

Syntax to create a index on table:

```
Create INDEX index_no ON employee (emp_id);
```

Analyzing query performance with the index;

```
Select *from employee where department=”Finance”;
```

statement provides detailed information about how the query is executed, including the use of indexes.

Dropping the Index

To observe the impact of removing the index, we can drop the index and re-run the query.

```
Drop INDEX  index_no ON employees;
```

Analyzing Query Performance without the Index

Run the same query again to compare the performance.

Select* from employee where department=”Finance”;

Creating and Dropping an Index in SQL

Indexes in SQL are used to improve the performance of data retrieval. They allow the database engine to find rows more quickly and efficiently. Here's a demonstration of creating an index on a table, analyzing its impact on query performance, and then removing the index to observe the change in performance.

Setup: Creating a Sample Table

- **Faster Query Execution**: The query that filters by the `department` column will be faster because the database can use the index to quickly locate the relevant rows.
- **Lower Cost**: The cost reported by will be lower because the index allows for more efficient data access.

Without Index

- **Slower Query Execution**: The query will likely be slower because the database has to perform a full table scan to find the rows that match the filter condition.
- **Higher Cost**: The cost reported by `EXPLAIN ANALYZE`will be higher due to the increased amount of data that needs to be processed.

Assignment :

Prepare a series of sql statement to insert new recourds in to the liabray table, update existing records with new information, and delite records bases on specific criteria, Include bulk indert operations to load data from an external sources.

Create a Table:

```
CREATE TABLE library (
   book_id INT PRIMARY KEY,
   title VARCHAR(255),
   author VARCHAR(255),
   published_year INT,
   genre VARCHAR(100)
);
```

Insert a records:

```
INSERT INTO library (book_id, title, author, published_year, genre) VALUES
(1, 'To Kill a Mockingbird', 'Harper Lee', 1960, 'Fiction'),
(2, '1984', 'George Orwell', 1949, 'Dystopian'),
(3, 'Moby Dick', 'Herman Melville', 1851, 'Adventure'),
(4, 'The Great Gatsby', 'F. Scott Fitzgerald', 1925, 'Tragedy')
```

Updating exiting records:

```
UPDATE library
SET author = 'George Orwell', published_year = 1948
WHERE title = '1984';

UPDATE library
SET genre = 'Classic Literature'
```

```
WHERE genre = 'Fiction';
```

Load data from external resources:

LOAD DATA INFILE statement in MySQL to load data from an external source like a CSV file.

```
LOAD DATA INFILE '/path/to/your/books.csv'
INTO TABLE library
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(book_id, title, author, published_year, genre);
```

Summary

The series of SQL statements provided includes:

1. **Inserting new records into the library table.**
2. **Updating existing records with new information.**
3. **Deleting records based on specific criteria.**
4. **Bulk inserting data from an external CSV file using COPY for PostgreSQL and LOAD DATA INFILE for MySQL.**