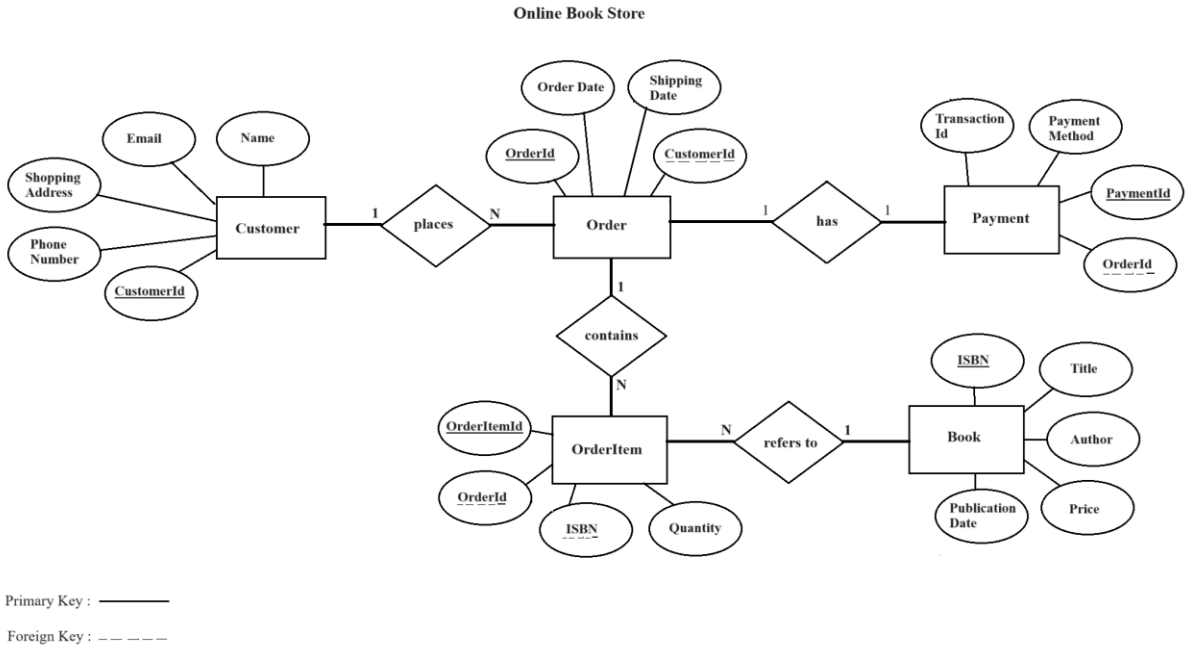
My SQL - Assignment 1

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.



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.

Tables and Fields along with Constraints :

1. Books

- **BookID** (Primary Key, INT, NOT NULL)
- **Title** (VARCHAR(100), NOT NULL)
- **Author** (VARCHAR(100), NOT NULL)
- **ISBN** (VARCHAR(13), NOT NULL, UNIQUE)

2. Members

- **MemberID** (Primary Key, INT, NOT NULL)
- **FirstName** (VARCHAR(50), NOT NULL)
- **LastName** (VARCHAR(50), NOT NULL)
- **Email** (VARCHAR(100), NOT NULL, UNIQUE)

3. **Borrowings**

- **BorrowingID** (Primary Key, INT, NOT NULL)
- **BookID** (Foreign Key referencing '**Books(BookID)'**, INT, NOT NULL)
- **MemberID** (Foreign Key referencing '**Members(MemberID)'**, INT, NOT NULL)
- **BorrowDate** (DATE, NOT NULL)
- **ReturnDate** (DATE)

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 Properties :

1. **Atomicity**: Imagine buying a set of toys as a gift. You either get all the toys in the set, or you get none at all. There's no in-between where you end up with some of the toys. Similarly, in a transaction, either all the changes happen, or none of them do.
2. **Consistency**: Think of consistency like making sure your bank balance always adds up correctly. If you deposit money, your balance should increase. If you withdraw money, it should decrease. There shouldn't be any weird surprises like money disappearing or magically appearing out of nowhere.
3. **Isolation**: Imagine you're playing a game on your phone, and your friend starts playing a different game on their phone next to you. Your game shouldn't suddenly glitch or crash just because your friend is playing their game. In a database, isolation means one transaction doesn't mess with another transaction's data.
4. **Durability**: Picture writing something down in a notebook. Once it's written, even if you close the notebook and put it away, what you wrote is still there when you open it again later. In a database, durability means once a transaction is done, its changes stick around, even if there's a power outage or a system crash.

Steps:

- We create tables for Accounts and Transactions.
- Sample data is inserted into the Accounts table.
- We start a transaction and set the desired isolation level (SERIALIZABLE or READ COMMITTED).

- Row-level locking is applied to ensure data consistency and prevent concurrent modifications.
- Money is transferred between accounts within the transaction.
- The transaction details are logged in the Transactions table.
- Finally, the transaction is committed, ensuring durability.

SQL Statements :

```
mysql> CREATE TABLE Accounts (
    ->     AccountID INT PRIMARY KEY,
    ->     Balance DECIMAL(10,2)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Transactions (
    ->     TransactionID INT PRIMARY KEY AUTO_INCREMENT,
    ->     AccountFrom INT,
    ->     AccountTo INT,
    ->     Amount DECIMAL(10,2),
    ->     TransactionDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO Accounts (AccountID, Balance) VALUES (1, 1000), (2, 2000);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
Query OK, 0 rows affected (0.00 sec)

mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @balance1 := Balance
    -> FROM Accounts
    -> WHERE AccountID = 1
    -> FOR UPDATE;
+----------------------+
| @balance1 := Balance |
+----------------------+
|              1000.00 |
+----------------------+
1 row in set, 1 warning (0.00 sec)

mysql> SELECT @balance2 := Balance
    -> FROM Accounts
    -> WHERE AccountID = 2
    -> FOR UPDATE;
+----------------------+
| @balance2 := Balance |
+----------------------+
|              2000.00 |
+----------------------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> UPDATE Accounts
    -> SET Balance = @balance1 - 100
    -> WHERE AccountID = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE Accounts
    -> SET Balance = @balance2 + 100
    -> WHERE AccountID = 2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> INSERT INTO Transactions (AccountFrom, AccountTo, Amount)
    -> VALUES (1, 2, 100);
Query OK, 1 row affected (0.01 sec)

mysql> commit;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from accounts;
+-----------+---------+
| AccountID | Balance |
+-----------+---------+
|         1 |  900.00 |
|         2 | 2100.00 |
+-----------+---------+
2 rows in set (0.00 sec)

mysql> select * from transactions;
+---------------+-------------+-----------+--------+---------------------+
| TransactionID | AccountFrom | AccountTo | Amount | TransactionDate     |
+---------------+-------------+-----------+--------+---------------------+
|             1 |           1 |         2 | 100.00 | 2024-05-21 21:13:56 |
+---------------+-------------+-----------+--------+---------------------+
1 row in set (0.00 sec)
```

Assignment 4: 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.

1. Create a New Database :

CREATE DATABASE LibrarySystem;

2. Use the new database :

USE LibrarySystem;

3. Create Tables :

A. Create Books table :

CREATE TABLE Books (

BookID INT **PRIMARY KEY**,

```
    Title VARCHAR(100) NOT NULL,
    Author VARCHAR(100) NOT NULL,
    ISBN VARCHAR(13) NOT NULL UNIQUE
);
```

B. Create Members table

```
CREATE TABLE Members (
    MemberID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE
);
```

C. Create Borrowings table

```
CREATE TABLE Borrowings (
    BorrowingID INT PRIMARY KEY,
    BookID INT NOT NULL,
    MemberID INT NOT NULL,
    BorrowDate DATE NOT NULL,
    ReturnDate DATE,
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID)
);
```

D. Create a LibraryData table

```
CREATE TABLE LibraryData (
    DataID INT PRIMARY KEY,
    Data VARCHAR(100) NOT NULL
);
```

4. Modify Table Structure :

```
ALTER TABLE Books
ADD Category VARCHAR(50) NOT NULL;
```

5. Remove Redundant Table :

```
DROP TABLE LibraryData;
```

Assignment 5: 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.

Creating an Index & Analyze Query Performance with the Index

```
mysql> CREATE INDEX idx_job ON Employee (job);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT * FROM Employee WHERE Job='Developer';
+----------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                              |
+----------------------------------------------------------------------------------------------------------------------+
| -> Index lookup on Employee using idx_job (Job='Developer')  (cost=0.7 rows=2) (actual time=0.04..0.0431 rows=2 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

How Indexes Improve Query Performance :

Indexes improve query performance by allowing the database to quickly locate and access the rows that match the query conditions. For example, an index on the Job column lets the database efficiently find rows with Job = ‘Developer’ without scanning the entire table. This results in significantly faster query execution times, especially for large tables.

Removing the Index & Analyze Query Performance without the Index

```
mysql> DROP INDEX idx_job ON Employee;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT * FROM Employee WHERE Job='Developer';
+-------------------------------------------------------------------------------------------------------------
------------------------------------------+
| EXPLAIN
                                                                  |
+-------------------------------------------------------------------------------------------------------------
------------------------------------------+
| -> Filter: (employee.Job = 'Developer')  (cost=1.05 rows=1) (actual time=0.0529..0.0587 rows=2 loops=1)
    -> Table scan on Employee  (cost=1.05 rows=8) (actual time=0.0462..0.0533 rows=8 loops=1)
 |
+-------------------------------------------------------------------------------------------------------------
------------------------------------------+
1 row in set (0.00 sec)
```

Impact of Dropping the Index :

When we drop the index, the database loses this quick lookup capability and has to revert to scanning the entire table to find matching rows. This leads to increased query execution times as the table size grows.

Assignment 6: Create a new database user with specific privileges using the CREATE USER and GRANT commands. Then, write a script to REVOKE certain privileges and DROP the user.

1. Create a new database user :

CREATE USER 'library_user'@'localhost' **IDENTIFIED BY** 'password123';

2. Grant privileges to the new user :

GRANT SELECT, INSERT, UPDATE, DELETE **ON** LibrarySystem.* **TO** 'library_user'@'localhost';

3. Revoke the DELETE privilege from the user :

REVOKE DELETE **ON** LibrarySystem.* **FROM** 'library_user'@'localhost';

4. Drop the user :

DROP USER 'library_user'@'localhost';

Assignment 7: Prepare a series of SQL statements to INSERT new records into the library tables, UPDATE existing records with new information, and DELETE records based on specific criteria. Include BULK INSERT operations to load data from an external source.

Step 1: Insert New Records :

```
mysql> -- Insert new records into Books table
mysql> INSERT INTO Books (Title, Author, ISBN, PublishedYear)
    -> VALUES
    -> ('The Great Gatsby', 'F. Scott Fitzgerald', '9780743273565', 1925),
    -> ('To Kill a Mockingbird', 'Harper Lee', '9780061120084', 1960),
    -> ('1984', 'George Orwell', '9780451524935', 1949);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> -- Insert new records into Members table
mysql> INSERT INTO Members (FirstName, LastName, Email)
    -> VALUES
    -> ('John', 'Doe', 'john.doe@example.com'),
    -> ('Jane', 'Smith', 'jane.smith@example.com'),
    -> ('Alice', 'Johnson', 'alice.johnson@example.com');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> -- Insert new records into Borrowings table
mysql> INSERT INTO Borrowings (BookID, MemberID, BorrowDate, ReturnDate)
    -> VALUES
    -> (1, 1, '2023-05-01', '2023-05-15'),
    -> (2, 2, '2023-05-03', '2023-05-17'),
    -> (3, 3, '2023-05-05', '2023-05-19');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Step 2: Update Existing Records :

```
mysql> -- Update the PublishedYear of a book
mysql> UPDATE Books
    -> SET PublishedYear = 1936
    -> WHERE ISBN = '9780743273565';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> -- Update the email address of a member
mysql> UPDATE Members
    -> SET Email = 'john.updated@example.com'
    -> WHERE FirstName = 'John' AND LastName = 'Doe';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> -- Update the ReturnDate of a borrowing record
mysql> UPDATE Borrowings
    -> SET ReturnDate = '2023-05-20'
    -> WHERE BorrowingID = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Step 3: Delete Records Based on Specific Criteria :

```
mysql> -- Delete borrowing records that reference the book with ISBN '9780451524935'
mysql> DELETE FROM Borrowings
    -> WHERE BookID = (SELECT BookID FROM Books WHERE ISBN = '9780451524935');
Query OK, 1 row affected (0.01 sec)

mysql> -- Delete the book record with ISBN '9780451524935'
mysql> DELETE FROM Books
    -> WHERE ISBN = '9780451524935';
Query OK, 1 row affected (0.01 sec)
```

Step 4: BULK INSERT Operations :

- **Insert_values.csv file :**

	A	B	C	D
1	Title	Author	ISBN	PublishedYear
2	Moby Dick	Herman Me	9.782E+12	1851
3	War and Pe	Leo Tolstoy	9.78E+12	1869
4	Pride and Pr	Jane Austen	9.782E+12	1813

- **Statement to Insert data using csv file :**

```
mysql> LOAD DATA INFILE 'E:\Ep-Java-WIpro\insert_values.csv'
    -> INTO TABLE Books
    -> FIELDS TERMINATED BY ','
    -> ENCLOSED BY ""
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 LINES
    -> (Title, Author, ISBN, PublishedYear);
```