# Assignment-7

**Name - Md. Firoze Baba**

## 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.

## Answer:

**Entities:**

1. Patient
2. Doctor
3. Test

**Relationships:**

1. Patient visits Doctor
2. Doctor treats Patient
3. Doctor works in Department
4. Patient has Appointment
5. Patient undergoes Test

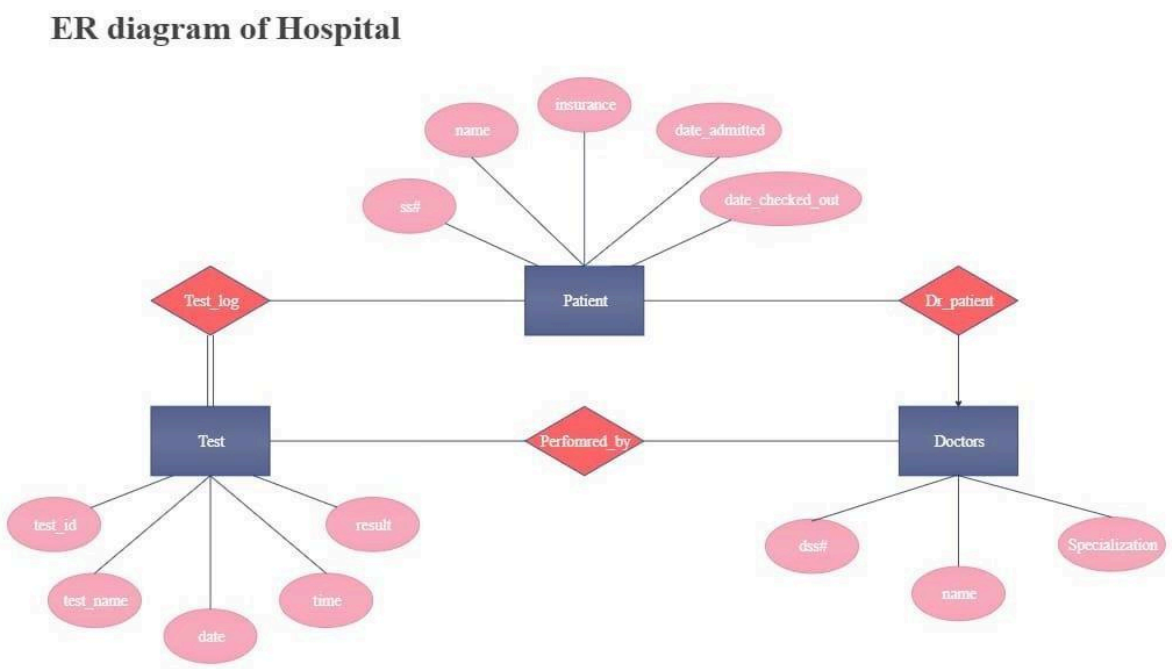
**Attributes:**

1. Patient: SS#, name, insurance, date_admitted, date_checked_out
2. Doctor: DSS, Name, Specialization

3.Test: test_ID, test_ name, Date, time, result

**Cardinality:**

1. One patient can have multiple appointments.
2. One doctor can have multiple appointments, treat multiple patients, and perform multiple tests.
3. One nurse can assist multiple doctors
4. One department can have multiple doctors.




**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.**

**Answer:**

**Constraints**:

- **NOT NULL Constraints:** Apply to fields that cannot have NULL values. For example, book titles, author names, transaction dates, etc.
- **UNIQUE Constraints**: Apply to fields that must have unique values. For example, ISBNs should be unique for each book.

- **CHECK Constraints**: Apply to enforce domain integrity. For example, ensuring that publication years are reasonable.
- **Primary Keys**: Unique identifiers for each record in a table.
- **Foreign Keys**: Establish relationships between tables. For example, the **book_id** in the Transactions table references the **book_id** in the Books table, indicating which book a transaction is associated with.

**Initial Tables:**

1. Books
    - ISBN (PK) (Unique)
    - Title (Not Null)
    - PublicationDate
    - AuthorID (FK)
2. Authors
    - AuthorID (PK)
    - Name
3. Patrons
    - PatronID (PK)
    - Name
4. Transactions
    - TransactionID (PK) (unique)
    - BookID (FK)
    - PatronID (FK)
    - DueDate

**First Normal Form (1NF):**

In the First Normal Form, each table should have a primary key, and there should be no repeating groups or arrays in any column. Our initial tables

already satisfy 1NF because they have primary keys, and each cell contains a single value.

**Second Normal Form (2NF):**

To achieve 2NF, we need to ensure that non-key attributes depend on the entire primary key. In our initial tables, the Books table has partial dependency because Title and PublicationDate depend only on ISBN, not the entire primary key. We'll split the Books table into two tables:

Books (2NF):

- ISBN (PK)
- AuthorID (FK)

BookDetails (2NF):

- ISBN (FK)
- Title
- PublicationDate

Now, each table has attributes that depend on the entire primary key.

**Third Normal Form (3NF):**

In 3NF, we eliminate transitive dependencies. The Patrons table has no transitive dependencies, but the Transactions table has a transitive dependency on Books through the BookID. To remove this dependency, we'll create a new table for transactions:

Transactions (3NF):

- TransactionID (PK)
- PatronID (FK)
- DueDate

BorrowedBooks (3NF):

- TransactionID (FK)
- BookID (FK)

Now, the Transactions table only depends on the PatronID, and the BorrowedBooks table manages the relationship between transactions and books.

Our normalized tables in 3NF are as follows:

Books (3NF):

- ISBN (PK)
- AuthorID (FK)

BookDetails (3NF):

- ISBN (FK)
- Title
- PublicationDate

Authors (3NF):

- AuthorID (PK)
- Name

Patrons (3NF):

- PatronID (PK)
- Name

Transactions (3NF):

- TransactionID (PK)
- PatronID (FK)
- DueDate

BorrowedBooks (3NF):

- TransactionID (FK)
- BookID (FK)

**Database Schema**

**Table for Authors**

```
CREATE TABLE Authors (
AuthorID INT PRIMARY KEY,
Name VARCHAR(255)
);
```

**Table for Books**

```
CREATE TABLE Books (
ISBN VARCHAR(13) PRIMARY KEY,
AuthorID INT,
FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
);
```

**Table for BookDetails**

```
CREATE TABLE BookDetails (
ISBN VARCHAR(13),
Title VARCHAR(255),
PublicationDate DATE,
FOREIGN KEY (ISBN) REFERENCES Books(ISBN)
);
```

**Table for Patrons**

```
CREATE TABLE Patrons (
PatronID INT PRIMARY KEY,
Name VARCHAR(255)
);
```

**Table for Transactions**

```
CREATE TABLE Transactions (
TransactionID INT PRIMARY KEY,
PatronID INT,
DueDate DATE,
FOREIGN KEY (PatronID) REFERENCES Patrons(PatronID)
);
```

**Table for BorrowedBooks** (to represent the many-to-many relationship between Transactions and Books)

```
CREATE TABLE BorrowedBooks (
TransactionID INT,
ISBN VARCHAR(13),
```

FOREIGN KEY (TransactionID) REFERENCES Transactions(TransactionID),
FOREIGN KEY (ISBN) REFERENCES Books(ISBN)
);

**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.**

**Answer:**

**ACID stands for:**

1. **Atomicity**: Atomicity ensures that a transaction is treated as a single unit of work, meaning that either all of its operations are completed successfully, or none of them are. If any part of the transaction fails, the entire transaction is rolled back to its original state.
2. **Consistency:** Consistency ensures that the database remains in a valid state before and after the transaction. This means that any constraints, such as foreign key relationships or unique key constraints, are maintained during and after the transaction.
3. **Isolation:** Isolation ensures that the execution of multiple transactions concurrently does not result in any interference between them. Each transaction should operate independently of other transactions, as if it were the only transaction being executed on the database.
4. **Durability:** Durability ensures that once a transaction is committed, its effects are permanently recorded in the database, even in the event of a system failure. This means that the changes made by a committed transaction persist and are not lost.

Simulating a Transaction with Locking

Scenario: Transferring $100 from account A (account_id = 1) to account B (account_id = 2)

Steps:

1. START TRANSACTION: This marks the beginning of the transaction.

2. SELECT balance FROM accounts WHERE account_id = 1 FOR UPDATE: This retrieves the balance of account A and acquires a lock on the row, preventing other transactions from modifying it.
3. UPDATE accounts SET balance = balance - 100 WHERE account_id = 1: This deducts $100 from account A's balance.
4. SELECT balance FROM accounts WHERE account_id = 2 FOR UPDATE: Similar to step 2, this retrieves the balance of account B and acquires a lock.
5. UPDATE accounts SET balance = balance + 100 WHERE account_id = 2: This adds $100 to account B's balance.
6. COMMIT: This permanently saves the changes made by the transaction.

**Demonstrating Isolation Levels**

Isolation levels control how transactions see changes made by other concurrent transactions. Here's how the above scenario might differ with different isolation levels:

1. Read Uncommitted (READ UNCOMMITTED): Transactions can see uncommitted changes from other transactions. This can lead to inconsistencies (dirty reads). Imagine Chef A removes a tomato from the fridge (read uncommitted), but Chef B finishes chopping it before Chef A commits adding it to the dish (uncommitted change).
2. Read Committed (READ COMMITTED): Transactions only see committed changes from other transactions. This avoids dirty reads but allows non-repeatable reads. Chef A might see the tomato in the fridge (read committed), but when they come back, it's gone because Chef B already committed using it.
3. Repeatable Read (REPEATABLE READ): Transactions see a consistent snapshot of the data as of the start of their read operations. This prevents non-repeatable reads but can lead to phantom reads. Chef A might see the tomato (repeatable read), but during their preparation, new tomatoes are added by the store (phantom data not visible in the initial read).
4. Serializable (SERIALIZABLE): Transactions are executed one at a time, ensuring the highest level of isolation but potentially impacting

performance. Imagine only one chef can use the kitchen at a time, preventing any conflicts but slowing down overall cooking.

```
mysql> use librarymanagement;
Database changed
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> upate accounts set balance = balance -100 where id = 1;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
t balance = balance -100 where id = 1' at line 1
mysql> select * from accounts where account_id = 1 for update;
ERROR 1146 (42S02): Table 'librarymanagement.accounts' doesn't exist
mysql> show tables
    -> ;
+-----------------------------+
| Tables_in_librarymanagement |
+-----------------------------+
| authors                     |
| books                       |
| products                    |
+-----------------------------+
3 rows in set (0.01 sec)

mysql> create table accounts ( account_id int primary key auto_increment, balance decimal(7,2) not null default 0.00 );
Query OK, 0 rows affected (0.05 sec)

mysql> select * from accounts where account_id = 1 for update;
Empty set (0.01 sec)

mysql> select * from accounts;
Empty set (0.00 sec)

mysql> desc accounts;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| account_id | int          | NO   | PRI | NULL    | auto_increment |
| balance    | decimal(7,2) | NO   |     | 0.00    |                |
+------------+--------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)

mysql> insert into accounts (1,1000.00);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
ne 1
mysql> insert into accounts values(1,1000.00);
Query OK, 1 row affected (0.01 sec)

mysql> insert into accounts values(2,1100.00);
Query OK, 1 row affected (0.01 sec)

mysql> select * from accounts;
+------------+---------+
| account_id | balance |
+------------+---------+
|          1 | 1000.00 |
|          2 | 1100.00 |
+------------+---------+
2 rows in set (0.00 sec)

mysql> select * from accounts where account_id = 1 for update;
+------------+---------+
| account_id | balance |
```

```
mysql> select * from accounts where account_id = 1 for update;
+------------+---------+
| account_id | balance |
+------------+---------+
|          1 | 1000.00 |
+------------+---------+
1 row in set (0.00 sec)

mysql> update accounts set balance = balance -100 where account_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from accounts;
+------------+---------+
| account_id | balance |
+------------+---------+
|          1 |  900.00 |
|          2 | 1100.00 |
+------------+---------+
2 rows in set (0.00 sec)

mysql> select * from accounts where account_id = 2 for update;
+------------+---------+
| account_id | balance |
+------------+---------+
|          2 | 1100.00 |
+------------+---------+
1 row in set (0.00 sec)

mysql> update accounts set balance = balance +200 where account_id = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from accounts;
+------------+---------+
| account_id | balance |
+------------+---------+
|          1 |  900.00 |
|          2 | 1300.00 |
+------------+---------+
2 rows in set (0.00 sec)

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

mysql> select * from accounts;
+------------+---------+
| account_id | balance |
+------------+---------+
|          1 |  900.00 |
|          2 | 1300.00 |
+------------+---------+
2 rows 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.**

## Answer:

## Create a database:

```
CREATE DATABASE libraryManagement;

Use librarymanagement;

Show tables;
```

**Create books table:**

```
CREATE TABLE books (
  book_id INT PRIMARY KEY AUTO_INCREMENT,
  title VARCHAR(255) NOT NULL,
  isbn VARCHAR(13) NOT NULL UNIQUE,
  publised_yr INT NOT NULL,
  author_id INT NOT NULL,
);

Desc books;
```

**Create authors table:**

```
CREATE TABLE Authors ( author_id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255) NOT NULL );

Desc authors;
```

**Alter books table:**

ALTER TABLE Books ADD COLUMN genre VARCHAR(255);

Desc books;

**Create a Department table :**

CREATE TABLE department ( Did int primary key, dName varchar(30) );

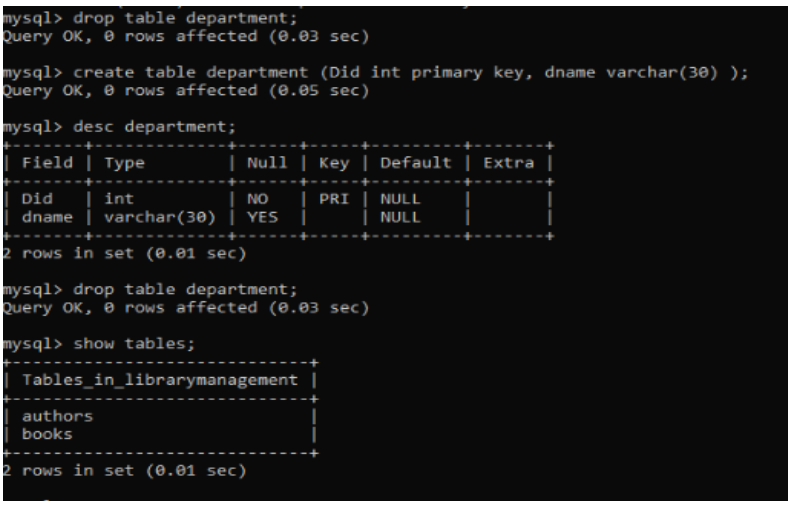
Desc department ;

Show tables;

**Drop department table:**

```
    -> author_id int not null );
ERROR 1050 (42S01): Table 'books' already exists
mysql> drop table books;
Query OK, 0 rows affected (0.06 sec)

mysql> use libraryManagement;
Database changed
mysql> create table books ( book_id int (10) primary key, title varchar(200) not null, isbn varchar(10) not null
    -> author_id int not null );
Query OK, 0 rows affected, 1 warning (0.10 sec)

mysql> desc books;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| book_id      | int          | NO   | PRI | NULL    |       |
| title        | varchar(200) | NO   |     | NULL    |       |
| isbn         | varchar(10)  | NO   | UNI | NULL    |       |
| published_yr | int          | NO   |     | NULL    |       |
| author_id    | int          | NO   |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)

mysql> create table authors ( author_id int primary key auto_increment, name varchar(200) not null );
ERROR 1050 (42S01): Table 'authors' already exists
mysql> drop table authors;
Query OK, 0 rows affected (0.03 sec)

mysql> create table authors ( author_id int primary key auto_increment, name varchar(200) not null );
Query OK, 0 rows affected (0.04 sec)

mysql> desc authors;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| author_id | int          | NO   | PRI | NULL    | auto_increment |
| name      | varchar(200) | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)

mysql> alter table books add column genre varchar(250);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc books;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| book_id      | int          | NO   | PRI | NULL    |       |
| title        | varchar(200) | NO   |     | NULL    |       |
| isbn         | varchar(10)  | NO   | UNI | NULL    |       |
| published_yr | int          | NO   |     | NULL    |       |
| author_id    | int          | NO   |     | NULL    |       |
| genre        | varchar(250) | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
6 rows in set (0.01 sec)

mysql> create table department (Did int primary key, dname varchar(30) );
ERROR 1050 (42S01): Table 'department' already exists
mysql> drop table department;
Query OK, 0 rows affected (0.03 sec)
```

Drop table department;




## 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.

## Answer:

### MySQL CREATE INDEX Statement.

The MySQL CREATE INDEX statement is a DDL (Data Definition Language) statement and it is used to create indexes on tables. It allows developers to optimize query performance by specifying which columns should be indexed. The index is a data structure that provides a quick lookup mechanism for faster data retrieval.

### Need for Indexing in MySQL:

- Indexing allows the database engine to quickly locate and retrieve the data.
- Indexed columns speed up the execution of SELECT, JOIN, and WHERE clauses.
- Unique indexes ensure the uniqueness of values in specified columns by preventing duplicate entries.
- Indexing improves the efficiency of sorting and grouping operations by providing an ordered structure for the data.

**How to Create an Index in MySQL**

To create an index we will use CREATE INDEX statement. This statement will allow you to define an index on one or more columns of a table.

**Syntax:**

CREATE [UNIQUE] INDEX index_name

ON table_name (column1, column2, …);

**Create  customers table:**

```
CREATE TABLE Customers (
  CustomerID INT PRIMARY KEY,
  FName VARCHAR(50),
  LName VARCHAR(50),
  Email VARCHAR(100)
);
```

**Insert values to customers table:**

Insert into customers values( 1,’jonh’,’gon’,’jonh@example.com), (2,’rose’,’simth’,rose@example.com), (3,’tom’,jerry’,’tom@example.com);

This query searches for customers with the lname = smith.

SELECT * FROM Customers WHERE LastName = 'Smith';

**Creating an Index:**

An index acts like a reference book for the table, allowing faster retrieval of data based on specific columns. Let's create an index named idx_lname on the lname column:

CREATE INDEX idx_lName ON Customers(LName);

**Explanation of the CREATE INDEX Statement:**

- CREATE INDEX: This keyword initiates the creation of an index.

- idx_lName: This is the user-defined name for the index. Choose a descriptive name that reflects the column(s) it applies to.
- ON Customers(LName): This specifies the table (Customers) and the column (LName) on which the index will be created.

**Re-run :**

SELECT * FROM Customers WHERE LastName = 'Smith';

Most database platforms will display information about how the query was executed, including whether it used an index. You might see something like "Using index idx_lName for table Customers". This indicates the database efficiently utilized the index to locate relevant entries.

**Drop index:**

DROP INDEX idx_lName ON Customers;

Run this query:

SELECT * FROM Customers WHERE LastName = 'Smith';

The database might need to scan through the entire LastName column to find matching entries, potentially leading to slower execution.

```
mysql> use librarymanagement;
Database changed
mysql> show tables;
+----------------------------+
| Tables_in_librarymanagement |
+----------------------------+
| accounts                   |
| authors                    |
| books                      |
| customers                  |
| products                   |
+----------------------------+
5 rows in set (0.01 sec)

mysql> desc customers;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| cust_id | int         | NO   | PRI | NULL    |       |
| Fname   | varchar(15) | NO   |     | NULL    |       |
| Lname   | varchar(15) | NO   |     | NULL    |       |
| email   | varchar(50) | NO   |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> insert into values ('Jonh', 'gon', 'jonh@example.com');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to you
gon', 'jonh@example.com')' at line 1
mysql> insert into customers values ('Jonh', 'gon', 'jonh@example.com');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> insert into customers values (1'Jonh', 'gon', 'jonh@example.com');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to you
onh@example.com')' at line 1
mysql> insert into customers values (1,'Jonh', 'gon', 'jonh@example.com');
Query OK, 1 row affected (0.01 sec)

mysql> insert into customers values (2,'rose', 'simth', 'rose@example.com');
Query OK, 1 row affected (0.01 sec)

mysql> insert into customers values (3,'Tom', 'jerry', 'tom@example.com');
Query OK, 1 row affected (0.01 sec)

mysql> select * from Customers;
+---------+-------+-------+------------------+
| cust_id | Fname | Lname | email            |
+---------+-------+-------+------------------+
|       1 | Jonh  | gon   | jonh@example.com |
|       2 | rose  | simth | rose@example.com |
|       3 | Tom   | jerry | tom@example.com  |
+---------+-------+-------+------------------+
3 rows in set (0.00 sec)

mysql> select * from customers where lname = 'simth';
+---------+-------+-------+------------------+
| cust_id | Fname | Lname | email            |
+---------+-------+-------+------------------+
|       2 | rose  | simth | rose@example.com |
+---------+-------+-------+------------------+
1 row in set (0.00 sec)
```

```
mysql> select * from customers where lname = 'simth';
+---------+-------+-------+------------------+
| cust_id | Fname | Lname | email            |
+---------+-------+-------+------------------+
|       2 | rose  | simth | rose@example.com |
+---------+-------+-------+------------------+
1 row in set (0.00 sec)

mysql> create index idx_lname on customers(lname);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from Customers;
+---------+-------+-------+------------------+
| cust_id | Fname | Lname | email            |
+---------+-------+-------+------------------+
|       1 | Jonh  | gon   | jonh@example.com |
|       2 | rose  | simth | rose@example.com |
|       3 | Tom   | jerry | tom@example.com  |
+---------+-------+-------+------------------+
3 rows in set (0.00 sec)

mysql> select * from customers where lname = 'simth';
+---------+-------+-------+------------------+
| cust_id | Fname | Lname | email            |
+---------+-------+-------+------------------+
|       2 | rose  | simth | rose@example.com |
+---------+-------+-------+------------------+
1 row in set (0.00 sec)

mysql> Drop index idx_lname on customers;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from customers where lname = 'simth';
+---------+-------+-------+------------------+
| cust_id | Fname | Lname | email            |
+---------+-------+-------+------------------+
|       2 | rose  | simth | rose@example.com |
+---------+-------+-------+------------------+
1 row in set (0.00 sec)
```

**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.**

**Answer:**

1. Create a new user named 'new_user' identified by password 'secure_password'

```
CREATE USER 'tom'@'localhost' IDENTIFIED BY 'pass';
```

2. Grant SELECT and INSERT privileges on the 'products' database and all tables within it

```
GRANT SELECT, INSERT ON products.* TO 'tom'@'localhost';
```

3. Grant additional privileges if needed (e.g., UPDATE, DELETE)

```
GRANT UPDATE, DELETE ON products.* TO 'tom'@'localhost';
```

4. Flush privileges to ensure immediate effect

```
FLUSH PRIVILEGES;
```

5. Revoke INSERT privilege from the 'new_user'

```
REVOKE INSERT ON products.* FROM 'tom'@'localhost';
```

6. Now the user can only SELECT data from 'products' database

```
INSERT INTO products (pname, price) VALUES ('pen', 100);
```

7. Drop the user 'new_user'

```
DROP USER 'Tom'@'localhost';
```

8. Verify the user is deleted by trying to grant privileges again

GRANT SELECT ON products.* TO 'tom'@'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.

## Answer:

CREATE DATABASE libraryManagement;

Use librarymanagement;

Show tables;

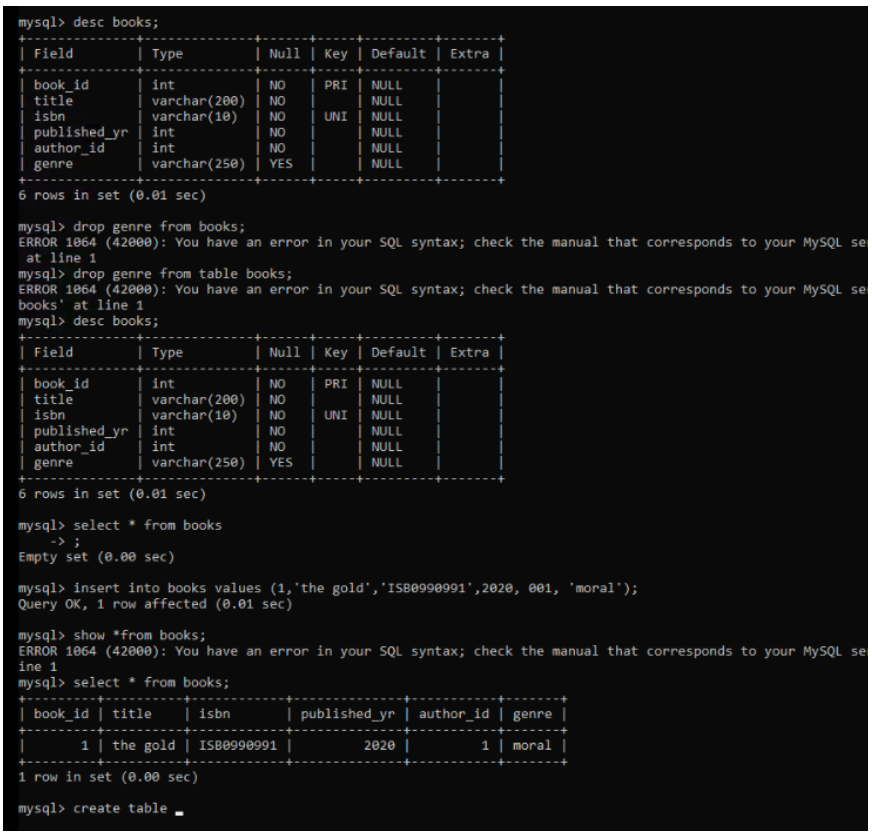
**Create books table:**

```
CREATE TABLE books (
  book_id INT PRIMARY KEY AUTO_INCREMENT,
  title VARCHAR(255) NOT NULL,
  isbn VARCHAR(13) NOT NULL UNIQUE,
  publised_yr INT NOT NULL,
  author_id INT NOT NULL,
  genre varchar(200)
);
```

Desc books;

**Insert values to books table:**

Insert into books values(1, ‘the gold’, ‘ISB0990901’, 2020, 001, ‘moral’);



```
mysql> desc books;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| book_id      | int         | NO   | PRI | NULL    |       |
| title        | varchar(200)| NO   |     | NULL    |       |
| isbn         | varchar(10) | NO   | UNI | NULL    |       |
| published_yr | int         | NO   |     | NULL    |       |
| author_id    | int         | NO   |     | NULL    |       |
| genre        | varchar(250)| YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)

mysql> drop genre from books;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL se
 at line 1
mysql> drop genre from table books;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL se
books' at line 1
mysql> desc books;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| book_id      | int         | NO   | PRI | NULL    |       |
| title        | varchar(200)| NO   |     | NULL    |       |
| isbn         | varchar(10) | NO   | UNI | NULL    |       |
| published_yr | int         | NO   |     | NULL    |       |
| author_id    | int         | NO   |     | NULL    |       |
| genre        | varchar(250)| YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)

mysql> select * from books
    -> ;
Empty set (0.00 sec)

mysql> insert into books values (1,'the gold','ISB0990991',2020, 001, 'moral');
Query OK, 1 row affected (0.01 sec)

mysql> show *from books;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL se
ine 1
mysql> select * from books;
+---------+----------+------------+--------------+-----------+-------+
| book_id | title    | isbn       | published_yr | author_id | genre |
+---------+----------+------------+--------------+-----------+-------+
|       1 | the gold | ISB0990991 |         2020 |         1 | moral |
+---------+----------+------------+--------------+-----------+-------+
1 row in set (0.00 sec)

mysql> create table
```

**Create members table:**

Create table members( name varchar (15) , email varchar(50) , ph_no int(10) primary key);

**Insert values to members table:**

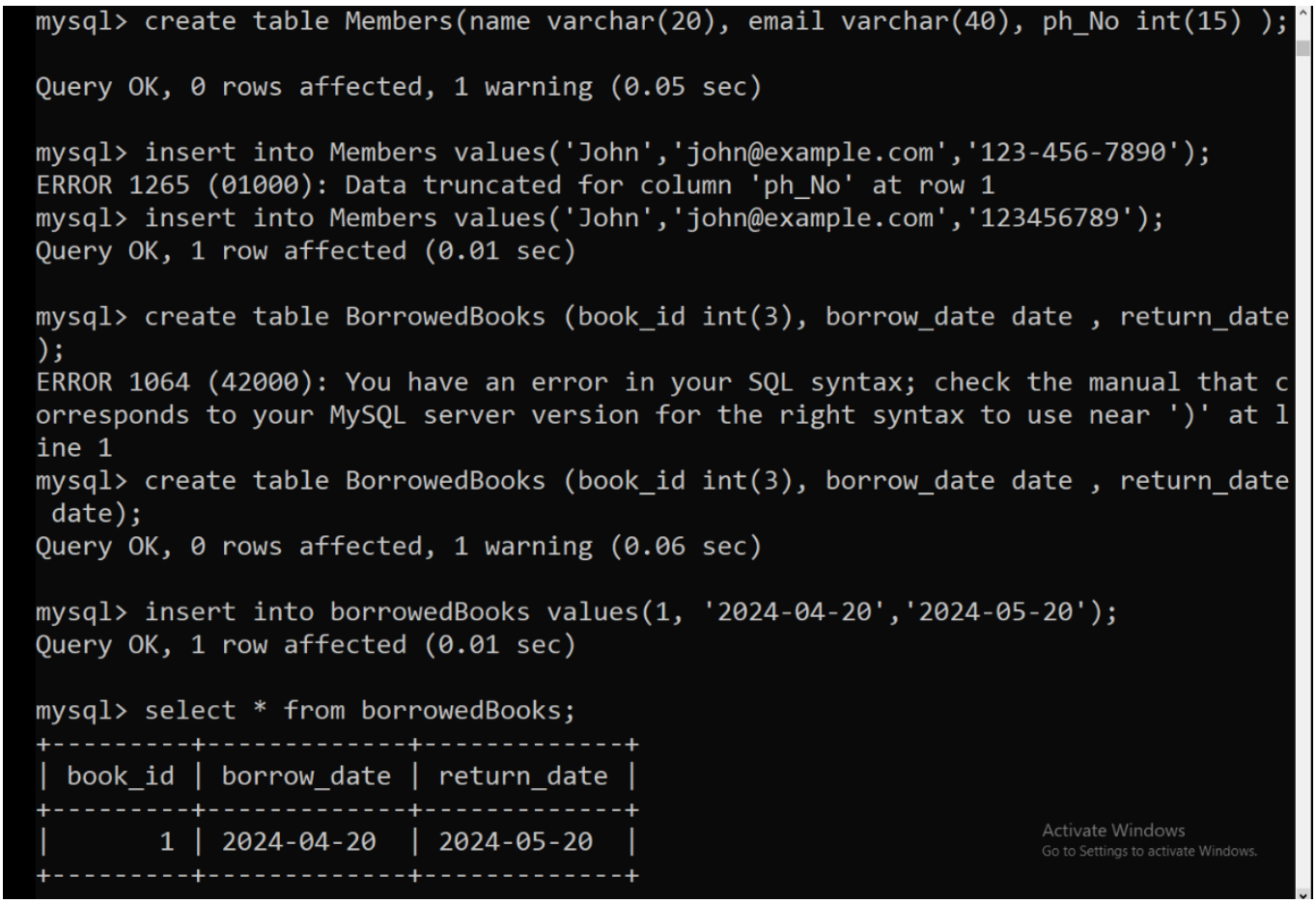
Insert into members values( ‘john’ , john@example.com’, 987245679);

**Create borrowedbooks table:**

Create table borrowedbooks(book_id int(2), borrow_date  date, return_date date);

**Insert values to borrowedbooks table:**

Insert into borrowedbooks values( 1 , ‘2024-04-12’,  current_date);




**Update:**

update books set genre = ‘fantasy’ where author_id =1;

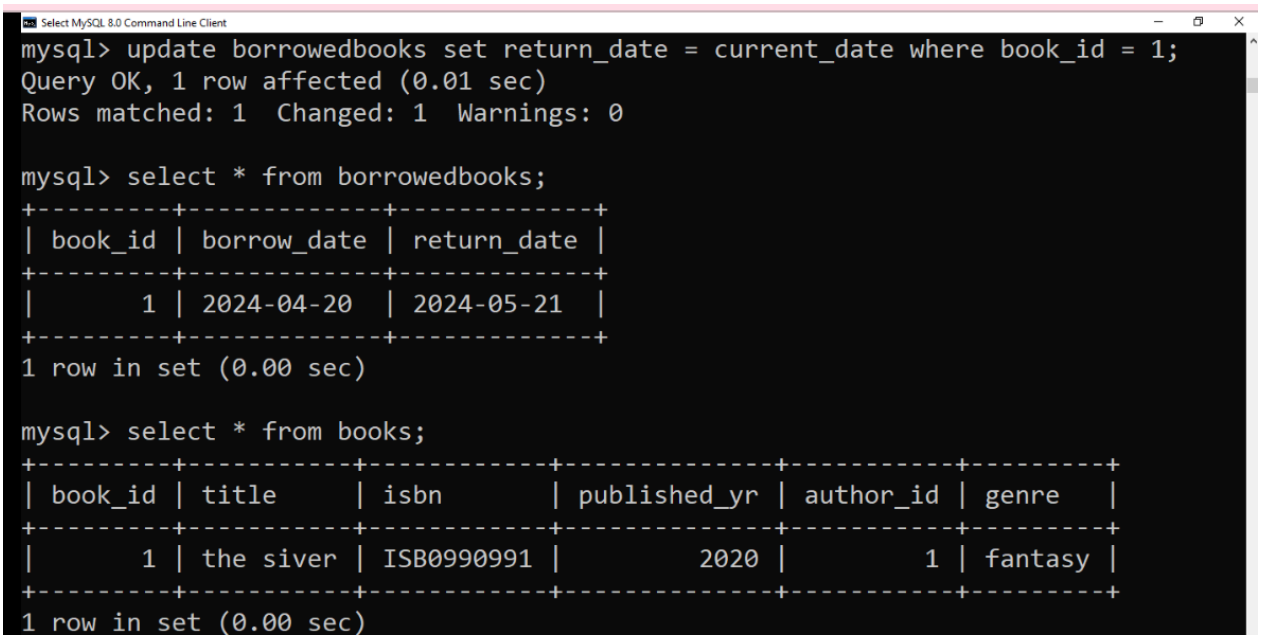
```
mysql> update books set title = 'the siver' where book_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from books;
+---------+-----------+------------+--------------+-----------+-------+
| book_id | title     | isbn       | published_yr | author_id | genre |
+---------+-----------+------------+--------------+-----------+-------+
|       1 | the siver | ISB0990991 |         2020 |         1 | moral |
+---------+-----------+------------+--------------+-----------+-------+
1 row in set (0.00 sec)

mysql> update books set genre = 'fantasy' where isbn = isb0990991;
ERROR 1054 (42S22): Unknown column 'isb0990991' in 'where clause'
mysql> update books set genre = 'fantasy' where author_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from books;
+---------+-----------+------------+--------------+-----------+---------+
| book_id | title     | isbn       | published_yr | author_id | genre   |
+---------+-----------+------------+--------------+-----------+---------+
|       1 | the siver | ISB0990991 |         2020 |         1 | fantasy |
+---------+-----------+------------+--------------+-----------+---------+
1 row in set (0.00 sec)

mysql> select * from members;
```



Update members set ph_no = ‘987865432’ where name = ‘john’

```
mysql> update members set ph_no = '987865432' where name = 'john';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from members;
+------+------------------+-----------+
| name | email            | ph_No     |
+------+------------------+-----------+
| John | john@example.com | 987865432 |
+------+------------------+-----------+
1 row in set (0.00 sec)

mysql> select * from borrowedbook;
ERROR 1146 (42S02): Table 'librarymanagement.borrowedbook' doesn't exist
mysql> select * from borrowedbooks;
+---------+-------------+-------------+
| book_id | borrow_date | return_date |
+---------+-------------+-------------+
|       1 | 2024-04-20  | 2024-05-20  |
+---------+-------------+-------------+
1 row in set (0.00 sec)

mysql> update borrowedbooks set return_date = current_date where book_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from borrowedbooks;
```

Update borrowedbooks set return_date = current_date where book_id = 1;




**Delete:**

Delete from books where book_id =1 and author_id =1;

Delete from members where name = john;

Delete from brrowedbooks where return_date = current_date;

```
mysql> delete from books where book_id = 1 and author_id = 1;
Query OK, 1 row affected (0.01 sec)

mysql> select * from books;
Empty set (0.00 sec)

mysql> select * from members;
+------+------------------+-----------+
| name | email            | ph_No     |
+------+------------------+-----------+
| John | john@example.com | 987865432 |
+------+------------------+-----------+
1 row in set (0.00 sec)

mysql> delete from members where name= john;
ERROR 1054 (42S22): Unknown column 'john' in 'where clause'
mysql> delete from members where name = 'john';
Query OK, 1 row affected (0.01 sec)

mysql> select * from members;
Empty set (0.00 sec)

mysql> select * from borrowedbooks;
+---------+-------------+-------------+
| book_id | borrow_date | return_date |
+---------+-------------+-------------+
|       1 | 2024-04-20  | 2024-05-21  |
```