DATABASE FUNDAMENTALS

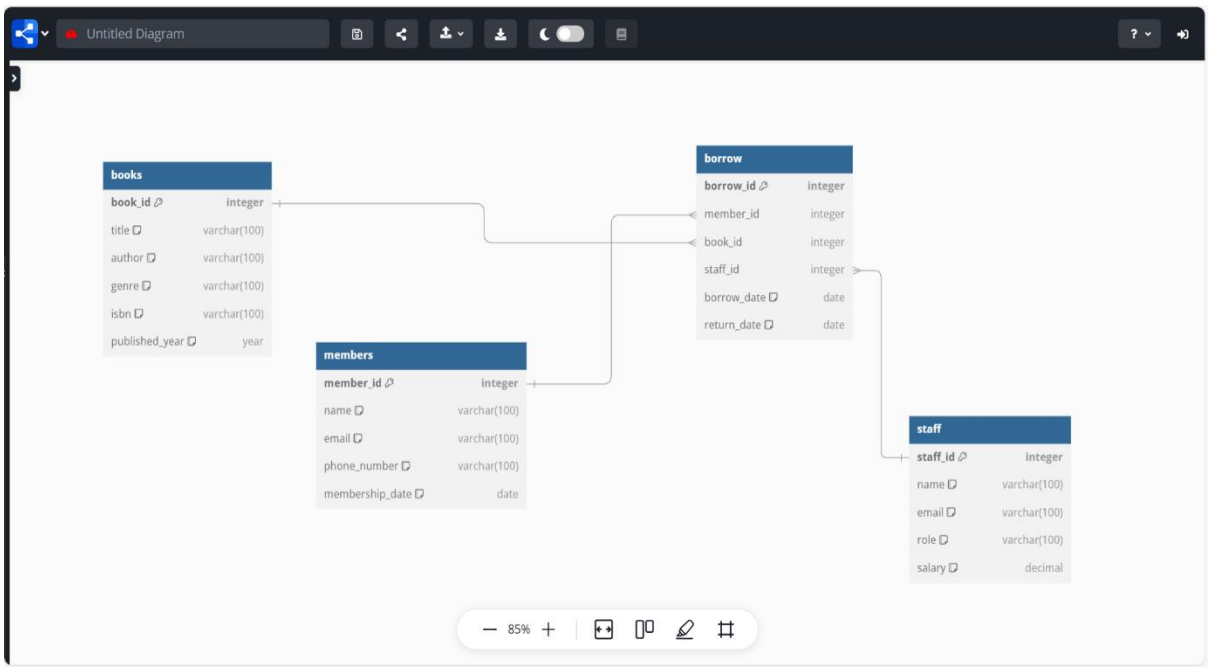
Database Fundamentals

- **Definition**: A database is an organized collection of data that can be easily accessed, managed, and updated. Databases are essential for storing large volumes of data in an efficient and structured manner.
- **Types of Databases**:
  - Relational (RDBMS)
  - NoSQL (e.g., MongoDB)
  - Hierarchical
  - Object-oriented
- **Database Management System (DBMS)**: Software that manages databases. Examples include MySQL, PostgreSQL, Oracle DB, and MongoDB.

Database Concepts and Architecture

- **Three-Tier Architecture**:
  - **Physical Layer**: Storage of data on disks.
  - **Logical Layer**: Defines the structure of data (schemas, tables, relationships).
  - **View Layer**: How users interact with data (via queries or UI).
- **Schemas**:
  - **Physical Schema**: Defines physical storage.
  - **Logical Schema**: Defines the structure (tables, views).
  - **View Schema**: Subset of data accessible to users.
- **Client-Server Architecture**: Databases operate in a client-server model where a server hosts the database, and clients access it.

Practical Task: Design a simple schema for a "Library Management System"

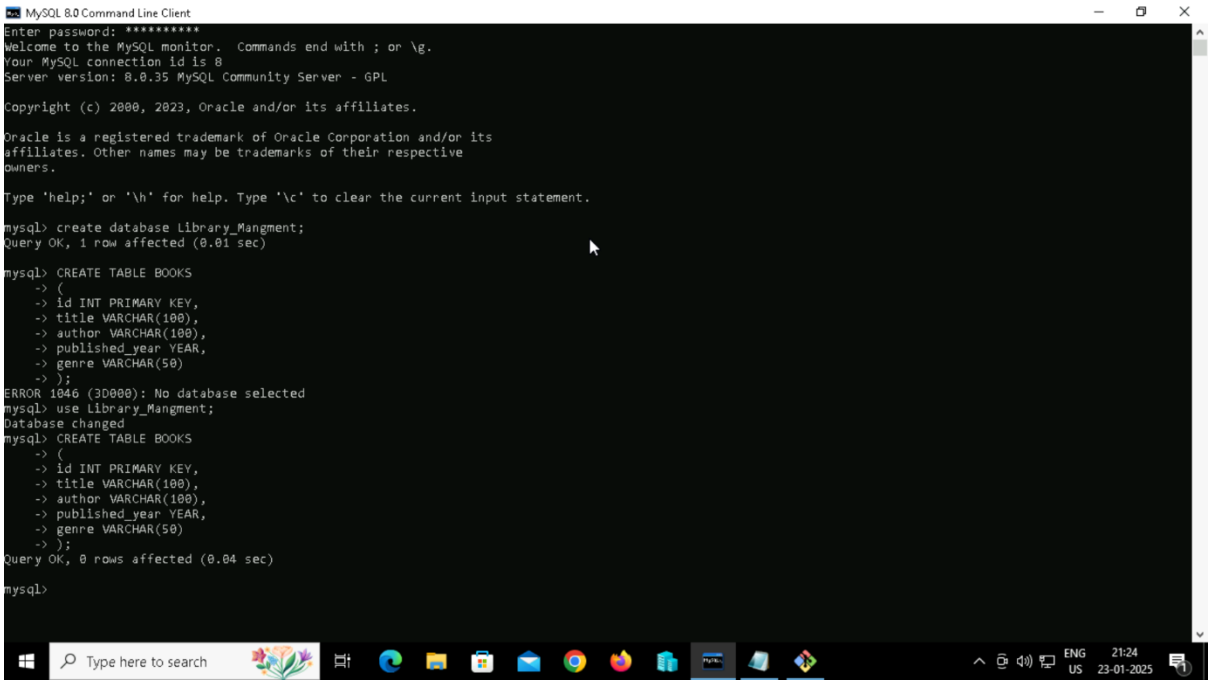


Relational Database Management Systems (RDBMS)

- **Definition**: A type of database that organizes data into tables (rows and columns) and uses relationships to connect them.
- **Key Features**:
  - Structured Data
  - Data Integrity
  - Use of SQL for operations
- **Examples**: MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Practical Task:

Create a table called books with columns: id, title, author, published_year, and genre.

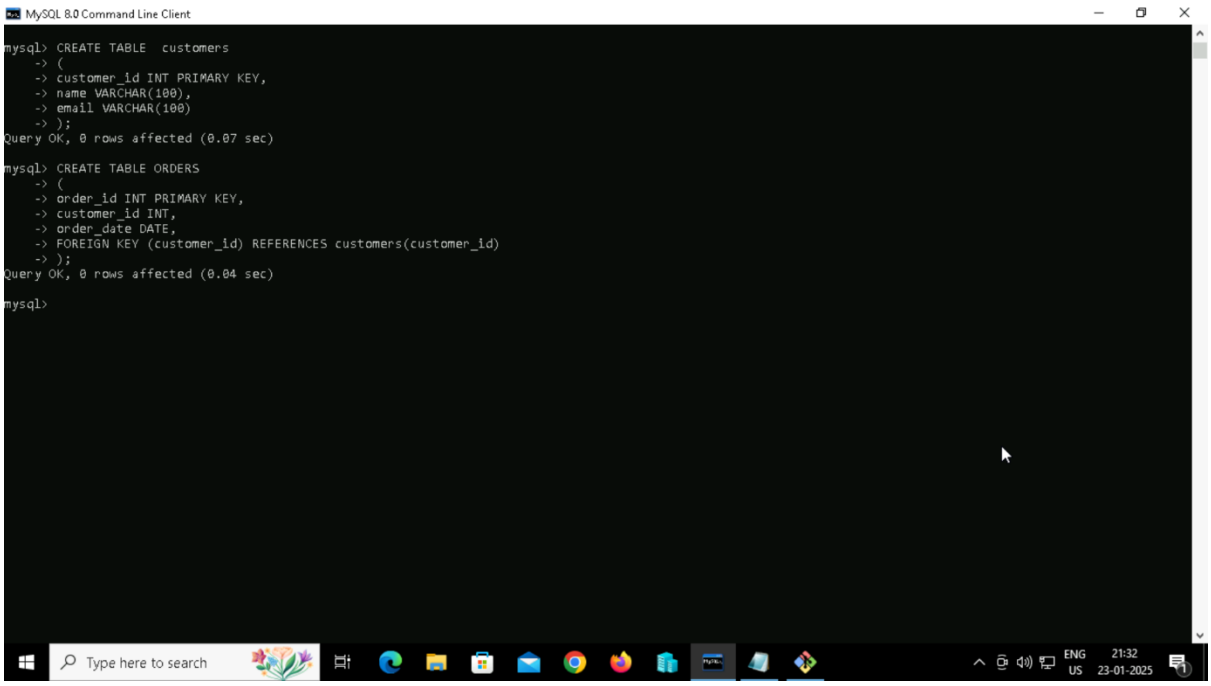


Tables and Relationships

- **Table**: A collection of rows (records) and columns (fields). Each table represents an entity.
- **Relationships**:
  - **One-to-One**: Each row in Table A maps to one row in Table B.
  - **One-to-Many**: One row in Table A maps to multiple rows in Table B.
  - **Many-to-Many**: Multiple rows in Table A map to multiple rows in Table B via a junction table.

Practical Task:

Create two tables: customers and orders. Define a one-to-many relationship between them using foreign keys.

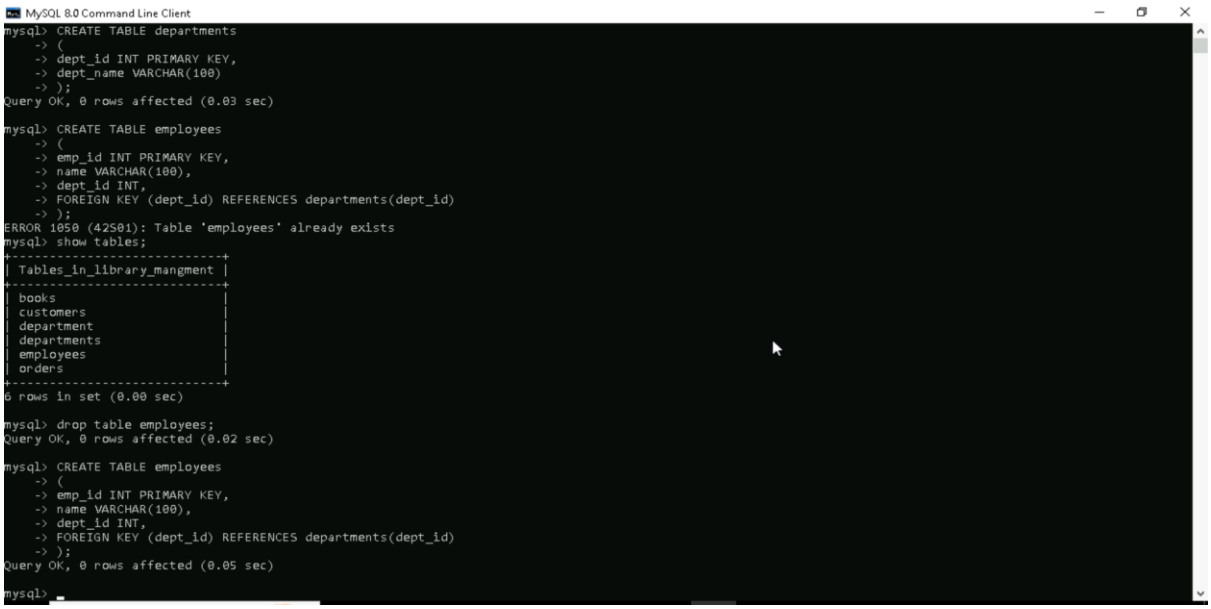


Primary Keys and Foreign Keys

- **Primary Key**:
  - Uniquely identifies each row in a table.
  - Cannot contain NULL values.
  - Example: id in a users table.
- **Foreign Key**:
  - Establishes a relationship between two tables.
  - Refers to the primary key in another table.
  - Example: user_id in an orders table references id in the users table.

Practical Task:

Create a table departments with a primary key, and another table employees with a foreign key referencing departments.



SQL Basics

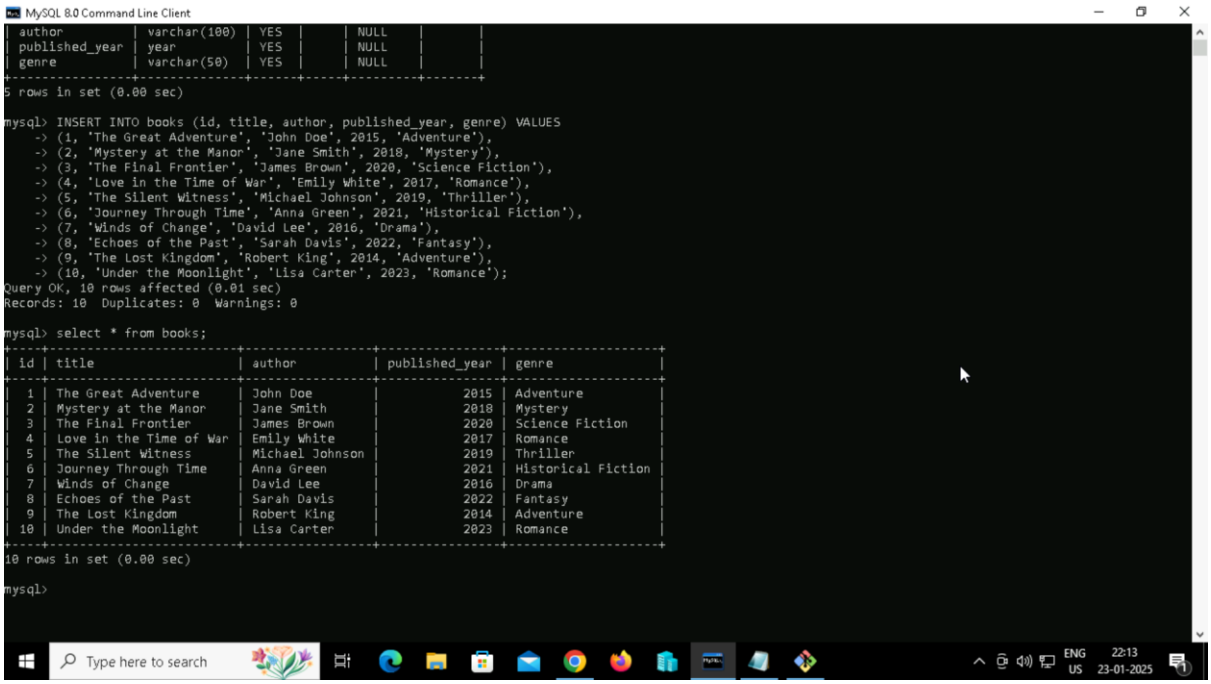
- **SELECT**: Retrieve data from a table.
- **INSERT**: Add new data to a table.
- **UPDATE**: Modify existing data.
- **DELETE**: Remove data.

Practical Task:

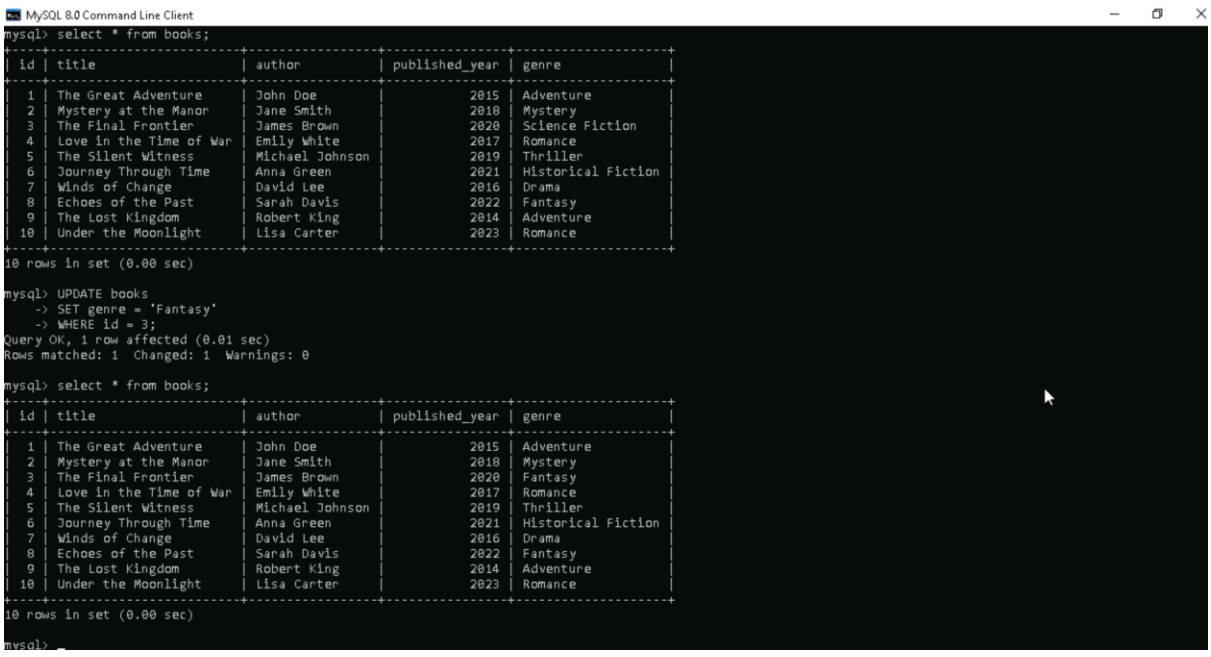
- Perform CRUD operations on the books table

  Inserting Data into books table

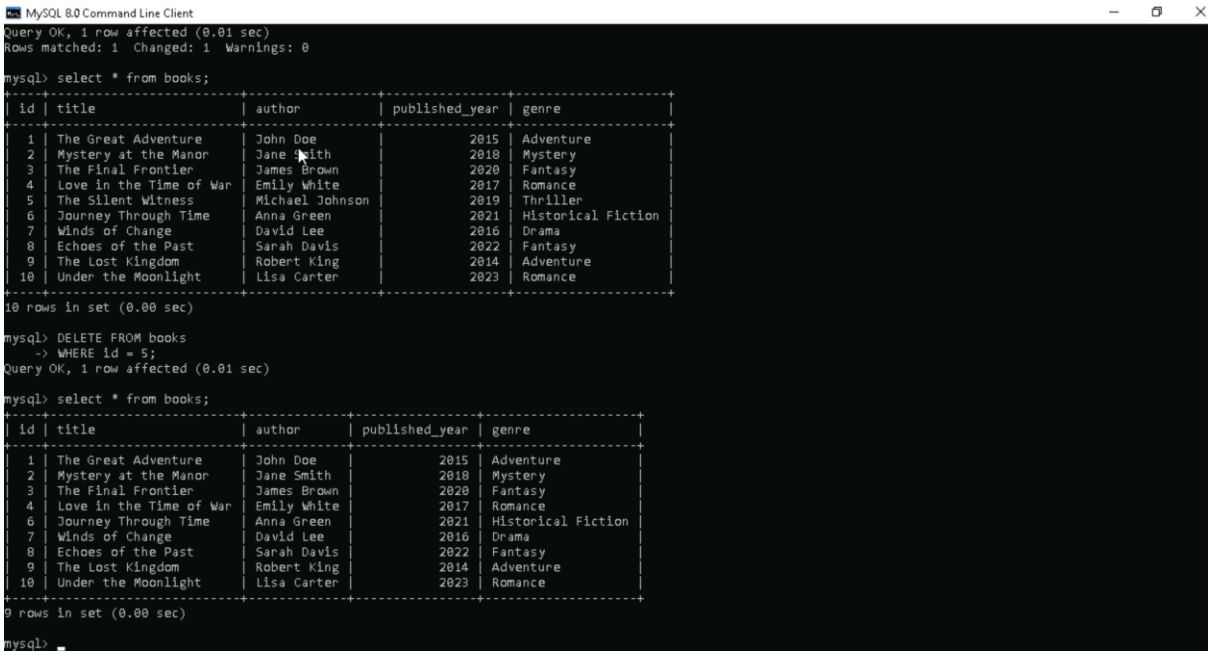
SELECT Retrieve data from a table.



UPDATE: Modify existing data.



DELETE: Remove data.



Normalization

- **Definition**: Process of organizing data to reduce redundancy and improve efficiency.
- **Forms of Normalization**:
  - **1NF (First Normal Form)**: Ensure atomic values and unique rows.
  - **2NF (Second Normal Form)**: Eliminate partial dependencies.
  - **3NF (Third Normal Form)**: Remove transitive dependencies.
  - **BCNF (Boyce-Codd Normal Form)**: Handle more complex dependencies.

Transactions and ACID Properties

- **Transaction**: A sequence of database operations treated as a single logical unit.
- **ACID Properties**:
  - **Atomicity**: Transactions are all-or-nothing.
  - **Consistency**: Transactions bring the database from one valid state to another.
  - **Isolation**: Transactions do not interfere with each other.
  - **Durability**: Once committed, data remains saved even in case of a failure.

Practical Task:

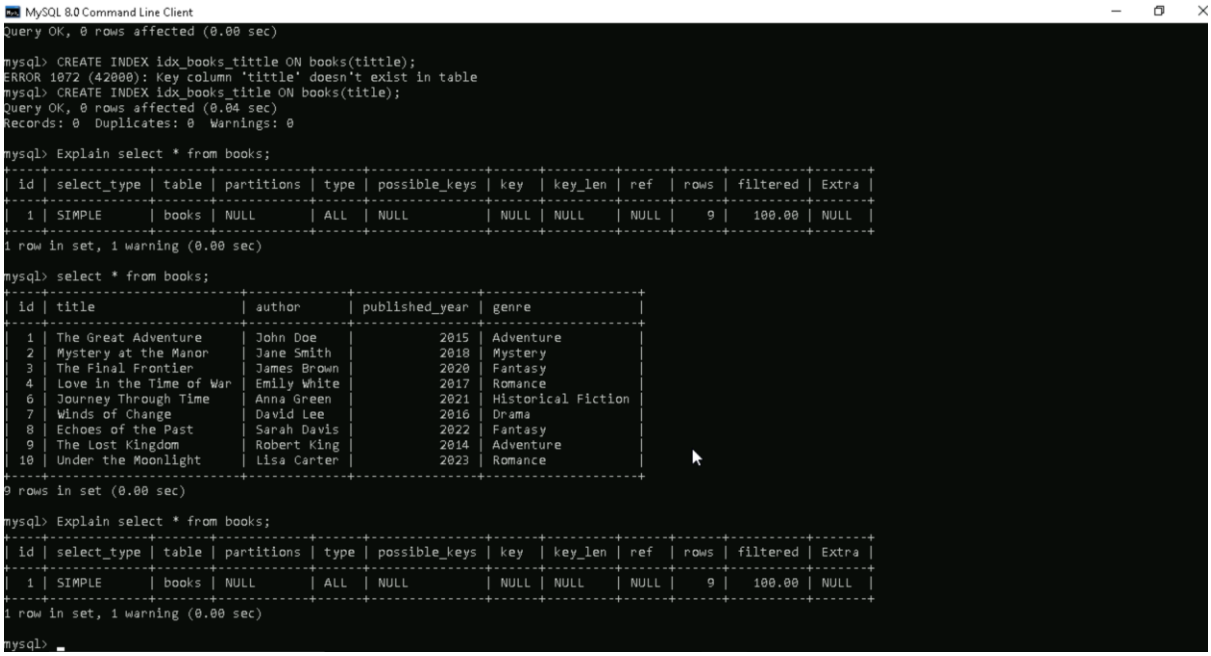
- Perform a transaction with multiple SQL operations and test rollback.

Indexes

- **Definition**: A database optimization technique to speed up data retrieval.
- **Types**:
  - **Primary Index**: Automatically created for the primary key.
  - **Secondary Index**: Manually created on other columns for faster lookups.
- **Trade-offs**:
  - Improves SELECT queries.
  - Slows down INSERT/UPDATE/DELETE due to index maintenance.

Practical Task:

- Create an index on the title column of the books table.

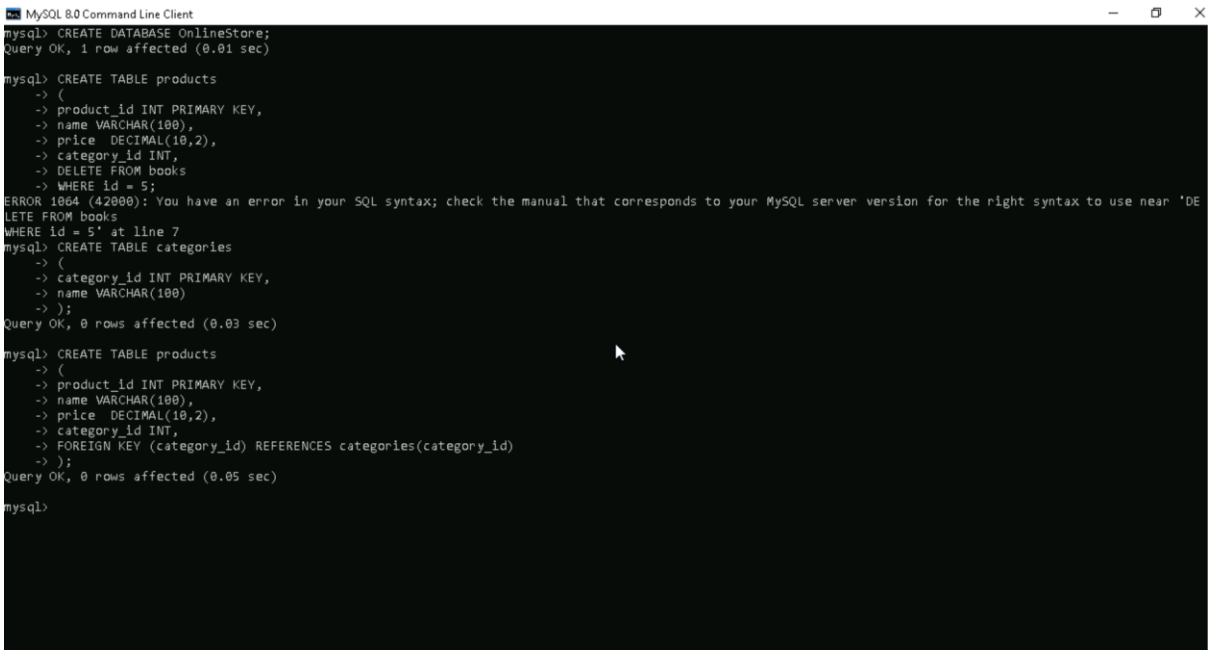


Database Design

- **Steps**:
  - **Requirement Analysis**: Understand data needs.
  - **Conceptual Design**: Create Entity-Relationship (ER) diagrams.
  - **Logical Design**: Define tables, relationships, and keys.
  - **Physical Design**: Optimize storage and indexing.
- **Best Practices**:
  - Use normalization.
  - Avoid redundancy.
  - Optimize queries for performance.

Practical Task:

- Design a database for an "Online Store" using:
  - Entities: Products, Categories, Orders, Customers.
  - Relationships: Define primary keys and foreign keys.



Backup and Recovery

- **Backup**:
  - Regularly save copies of database data.
  - Types: Full, Incremental, Differential.
- **Recovery**:
  - Process of restoring data after a failure.
  - **Point-in-Time Recovery**: Restore database to a specific moment.
- **Tools**:
  - MySQL Backup Tools, pg_dump (PostgreSQL), RMAN (Oracle).

Practical Task:

- Perform a full backup of your database:

```
mysqldump -u root -p my_database > my_database_backup.sql
```

- Simulate a data loss scenario by dropping a table, then restore it:

```
DROP TABLE books;

-- Restore using the backup
SOURCE my_database_backup.sql;
```