List of SQL commands that will guide you through the process of creating a database, making multiple tables, inserting data, and performing various operations on them. I'll provide examples for each task so you can practice:

**1. Create a Database**

Sql

CREATE DATABASE SchoolDB;

* This command creates a new database called SchoolDB.

**2. Use the Database**

sql

USE SchoolDB;

* This command selects the SchoolDB as the active database for future operations.

**3. Create Tables**

**a. Create Students Table**

sql

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

BirthDate DATE,

Email VARCHAR(100) UNIQUE

);

* This creates a Students table with columns for StudentID, FirstName, LastName, BirthDate, and Email.

**b. Create Courses Table**

sql

CREATE TABLE Courses (

CourseID INT PRIMARY KEY,

CourseName VARCHAR(100) NOT NULL,

Credits INT CHECK (Credits > 0)

);

* This creates a Courses table with columns for CourseID, CourseName, and Credits.

**c. Create Enrollments Table (Many-to-Many Relationship)**

sql

CREATE TABLE Enrollments (

EnrollmentID INT PRIMARY KEY,

StudentID INT,

CourseID INT,

EnrollmentDate DATE,

FOREIGN KEY (StudentID) REFERENCES Students(StudentID),

FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)

);

* This table connects students with courses, creating a many-to-many relationship.

**4. Insert Data into Tables**

**a. Insert Data into Students Table**

sql

INSERT INTO Students (StudentID, FirstName, LastName, BirthDate, Email)

VALUES (1, 'John', 'Doe', '2000-05-15', 'johndoe@example.com');

INSERT INTO Students (StudentID, FirstName, LastName, BirthDate, Email)

VALUES (2, 'Jane', 'Smith', '2001-08-22', 'janesmith@example.com');

**b. Insert Data into Courses Table**

sql

INSERT INTO Courses (CourseID, CourseName, Credits)

VALUES (1, 'Mathematics', 3);

INSERT INTO Courses (CourseID, CourseName, Credits)

VALUES (2, 'History', 2);

**c. Insert Data into Enrollments Table**

sql

INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, EnrollmentDate)

VALUES (1, 1, 1, '2023-09-01');

INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, EnrollmentDate)

VALUES (2, 2, 2, '2023-09-02');

**5. Update Data in Tables**

**a. Update a Student's Email**

sql

UPDATE Students

SET Email = 'john\_doe2023@example.com'

WHERE StudentID = 1;

**b. Update Course Credits**

sql

UPDATE Courses

SET Credits = 4

WHERE CourseID = 1;

**6. Delete Data from Tables**

**a. Delete a Student Record**

sql

DELETE FROM Students

WHERE StudentID = 2;

**b. Delete a Course Record**

sql

DELETE FROM Courses

WHERE CourseID = 2;

**c. Delete an Enrollment**

sql

DELETE FROM Enrollments

WHERE EnrollmentID = 1;

**7. Retrieve Data from Tables**

**a. Select All Records from Students**

sql

SELECT \* FROM Students;

**b. Select Specific Columns**

sql

SELECT FirstName, LastName, Email FROM Students;

**c. Select with a Condition**

sql

SELECT \* FROM Students

WHERE FirstName = 'John';

**d. Join Tables (Students and Enrollments)**

sql

SELECT Students.FirstName, Students.LastName, Courses.CourseName

FROM Students

JOIN Enrollments ON Students.StudentID = Enrollments.StudentID

JOIN Courses ON Enrollments.CourseID = Courses.CourseID;

* This query retrieves the FirstName and LastName of students along with the courses they are enrolled in.

**8. Alter Tables**

**a. Add a New Column to Students Table**

sql

ALTER TABLE Students

ADD PhoneNumber VARCHAR(15);

**b. Modify the Data Type of PhoneNumber**

sql

ALTER TABLE Students

MODIFY PhoneNumber VARCHAR(20);

**c. Drop a Column from Students Table**

sql

ALTER TABLE Students

DROP COLUMN PhoneNumber;

**9. Create and Use Views**

**a. Create a Simple View**

sql

CREATE VIEW StudentCourses AS

SELECT Students.FirstName, Students.LastName, Courses.CourseName

FROM Students

JOIN Enrollments ON Students.StudentID = Enrollments.StudentID

JOIN Courses ON Enrollments.CourseID = Courses.CourseID;

* This view simplifies querying students with their enrolled courses.

**b. Query the View**

sql

SELECT \* FROM StudentCourses;

**10. Using Subqueries**

**a. Select Students Enrolled in a Specific Course**

sql

SELECT FirstName, LastName

FROM Students

WHERE StudentID IN (

SELECT StudentID

FROM Enrollments

WHERE CourseID = 1

);

* This query retrieves the names of students enrolled in the course with CourseID = 1.

**b. Use Subquery in UPDATE Statement**

sql

UPDATE Students

SET Email = 'newemail@example.com'

WHERE StudentID = (SELECT StudentID FROM Enrollments WHERE CourseID = 2 LIMIT 1);

* This updates the email of the first student enrolled in CourseID = 2.

**11. Transaction Control (COMMIT, ROLLBACK, SAVEPOINT)**

**a. Begin Transaction**

sql

START TRANSACTION;

**b. COMMIT Changes**

sql

COMMIT;

* This saves all changes made during the transaction.

**c. ROLLBACK Changes**

sql

ROLLBACK;

* This undoes any changes made during the current transaction.

**d. Create Savepoint**

sql

SAVEPOINT beforeUpdate;

* This creates a savepoint, allowing you to roll back to this point later.

**e. Rollback to Savepoint**

sql

ROLLBACK TO SAVEPOINT beforeUpdate;

**12. Drop Tables**

**a. Drop Enrollments Table**

sql

DROP TABLE Enrollments;

* This deletes the Enrollments table and all its data.

**b. Drop the Entire Database**

sql

DROP DATABASE SchoolDB;

* This deletes the SchoolDB database, including all tables and data.

**Practice Suggestions:**

* **Create Complex Queries**: Combine multiple JOIN operations and use subqueries to practice more advanced querying.
* **Work with Constraints**: Add constraints like NOT NULL, CHECK, and UNIQUE to your tables.
* **Try Multiple Transactions**: Use COMMIT, ROLLBACK, and SAVEPOINT to practice managing transactions in your queries.