SQL Commands

1. Library System

a. Create the `Books` table:

```
CREATE TABLE Books (
    BookID INT PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(50),
    PublishedYear INT
);
```

b. Add a new column `ISBN`:

```
ALTER TABLE Books
ADD ISBN VARCHAR(13);
```

c. Delete the `PublishedYear` column:

```
ALTER TABLE Books
DROP COLUMN PublishedYear;
```

2. School Database

a. Retrieve names of students older than 15:

```
SELECT Name
FROM Students
WHERE Age > 15;
```

b. Insert a new record:

```
INSERT INTO Students (StudentID, Name, Age, Grade)
VALUES (104, 'Tom Brown', 15, 'C');
```

c. Update Jane Doe's grade:

```
UPDATE Students
SET Grade = 'A+'
WHERE Name = 'Jane Doe';
```

d. Delete records of students with grade 'C':

```
DELETE FROM Students
WHERE Grade = 'C';
```

3. Permissions for `User2`

a. Grant `SELECT` and `INSERT` privileges:

```
GRANT SELECT, INSERT ON Books TO User2;
```

b. Revoke the `INSERT` privilege:

```
REVOKE INSERT ON Books FROM User2;
```

4. Transactions for Shopping Cart

```
BEGIN TRANSACTION;

-- a. Start the transaction
INSERT INTO Orders (OrderID, ProductID, Quantity)
VALUES (201, 105, 2);

-- b. Deduct quantity from inventory
UPDATE Inventory
SET Quantity = Quantity - 2
WHERE ProductID = 105;

-- c. Commit or rollback
IF @@ERROR <> 0
BEGIN
   ROLLBACK TRANSACTION;
END
ELSE
BEGIN
   COMMIT TRANSACTION;
END
```

5. Online Store Operations

a. Create `Customers` table:

```
CREATE TABLE Customers (
   CustomerID INT PRIMARY KEY,
   Name VARCHAR(50),
   Email VARCHAR(50),
   Phone VARCHAR(15)
);
```

b. Insert a customer record:

```
INSERT INTO Customers (CustomerID, Name, Email, Phone)
VALUES (201, 'Emily Clark', 'emily@example.com', '1234567890');
```

c. Grant `SELECT` privilege to `AdminUser`:

```
GRANT SELECT ON Customers TO AdminUser;
```

d. Roll back the last inserted record:

```
BEGIN TRANSACTION;

DELETE FROM Customers
WHERE CustomerID = 201;

ROLLBACK TRANSACTION;
```

6. Employees and Departments

a. Create `Employees` table:

```
CREATE TABLE Employees (
   EmployeeID INT PRIMARY KEY,
   Name VARCHAR(50),
   Department VARCHAR(30),
   Salary INT
);
```

b. Rename `Department` to `Dept`:

```
ALTER TABLE Employees
RENAME COLUMN Department TO Dept;
```

c. Add `JoiningDate` column:

```
ALTER TABLE Employees
ADD JoiningDate DATE DEFAULT '2024-01-01';
```

d. Drop `Salary` column:

```
ALTER TABLE Employees
DROP COLUMN Salary;
```

e. Create `Departments` table and add foreign key:

```
CREATE TABLE Departments (
   DeptID INT PRIMARY KEY,
   DeptName VARCHAR(30) UNIQUE
);

ALTER TABLE Employees
ADD DeptID INT,
```

ADD CONSTRAINT FK_Dept FOREIGN KEY (DeptID) REFERENCES Departments(DeptID);

7. Employee Records

a. Insert record:

```
INSERT INTO Employees (EmployeeID, Name, Dept, Salary)
VALUES (1, 'Alice', 'HR', 50000);
```

b. Retrieve all employees in `HR` department:

```
SELECT * FROM Employees
WHERE Dept = 'HR';
```

c. Update salary of `IT` department employees:

```
UPDATE Employees
SET Salary = Salary * 1.10
WHERE Dept = 'IT';
```

d. Delete employees with salary < 40,000:

```
DELETE FROM Employees
WHERE Salary < 40000;
```

e. Use `MERGE` for updating or inserting:

```
MERGE INTO Employees AS Target
USING (SELECT 1 AS EmployeeID, 'Alice' AS Name, 'HR' AS Dept, 55000 AS Salary) AS
Source
ON Target.EmployeeID = Source.EmployeeID
WHEN MATCHED THEN
   UPDATE SET Salary = Source.Salary
WHEN NOT MATCHED THEN
   INSERT (EmployeeID, Name, Dept, Salary)
   VALUES (Source.EmployeeID, Source.Name, Source.Dept, Source.Salary);
```

8. Permissions Management

a. Grant privileges to `HRManager`:

```
GRANT SELECT, UPDATE ON Employees TO HRManager;
```

b. Revoke `UPDATE` privilege:

```
REVOKE UPDATE ON Employees FROM HRManager;
```

c. Create role `DataViewer` and grant privileges:

```
CREATE ROLE DataViewer;
```

GRANT SELECT ON ALL TABLES TO DataViewer;

d. Grant role to `Viewer1`:

GRANT DataViewer TO Viewer1;

e. Revoke all privileges from `InternUser`:

REVOKE ALL PRIVILEGES ON ALL TABLES FROM InternUser;

9. Transactions with Savepoints

a. Insert a record and commit:

```
BEGIN TRANSACTION;

INSERT INTO Departments (DeptID, DeptName)
VALUES (1, 'Finance');

COMMIT TRANSACTION;
```

b. Rollback on error:

```
BEGIN TRANSACTION;

INSERT INTO Departments (DeptID, DeptName)
VALUES (2, 'IT');

IF @@ERROR <> 0
   ROLLBACK TRANSACTION;
ELSE
   COMMIT TRANSACTION;
```

c. Savepoint during transaction:

```
BEGIN TRANSACTION;

SAVEPOINT InsertPoint;

INSERT INTO Employees (EmployeeID, Name, Dept, Salary)
VALUES (2, 'Bob', 'IT', 40000);

IF @@ERROR <> 0
   ROLLBACK TRANSACTION TO InsertPoint;
ELSE
   COMMIT TRANSACTION;
```

10. Projects Table

a. Create `Projects` table:

```
CREATE TABLE Projects (
   ProjectID INT PRIMARY KEY,
   ProjectName VARCHAR(50),
   EmployeeID INT,
   FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

b. Insert project record:

```
INSERT INTO Projects (ProjectID, ProjectName, EmployeeID)
VALUES (101, 'Website Upgrade', 1);
```

c. Transaction with rollback:

```
BEGIN TRANSACTION;

INSERT INTO Projects (ProjectID, ProjectName, EmployeeID)
VALUES (102, 'Mobile App Development', 2);

IF @@ERROR <> 0
   ROLLBACK TRANSACTION;
ELSE
   COMMIT TRANSACTION;
```