SQL Script Explanation

Table of Contents

---

1. Introduction

This document provides a detailed explanation of the SQL script, which demonstrates key database concepts including database creation, table creation, constraints (primary and foreign keys), cascading actions, and views.

---

2. Script Breakdown

2.1 CREATE DATABASE

The script begins by creating a database named `SchoolDB`. This is the foundation for all subsequent operations.

```
CREATE DATABASE SchoolDB;
USE SchoolDB;
```

- **Purpose**: Defines a container to store tables and data.

2.2 CREATE TABLE

Several tables are created to represent entities:

- `Students`: Contains student details.
- `Courses`: Contains course details.
- `Professors`: Contains professor details.
- `CourseAssignments`: Links professors to courses.
- `Enrollments`: Links students to courses.

Example:

```
CREATE TABLE Students (
    StudentID INT,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    DateOfBirth DATE,
    PRIMARY KEY (StudentID)
);
```

- **Purpose**: Defines the structure of each table and its columns.

2.3 Primary Key Constraints

Primary keys ensure that each row in a table is uniquely identifiable.

- Example from `Students` table:

```
PRIMARY KEY (StudentID)
```

- Example of a composite primary key in `CourseAssignments`:

```
PRIMARY KEY (CourseID, ProfessorID)
```

2.4 Foreign Key Constraints

Foreign keys establish relationships between tables.

- Example from `CourseAssignments`:

```
FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
   ON DELETE CASCADE
   ON UPDATE CASCADE;
```

- **Purpose**: Links `CourseAssignments` to `Courses` and defines actions on delete and update.

2.5 ON DELETE and ON UPDATE Actions

These actions define the behavior when related data is deleted or updated.

Examples:

- **ON DELETE CASCADE**: Deleting a course will delete related assignments.
- **ON DELETE SET NULL**: Deleting a student sets related foreign key fields to `NULL`.

```
FOREIGN KEY (StudentID) REFERENCES Students(StudentID)
   ON DELETE SET NULL
   ON UPDATE CASCADE;
```

2.6 INSERT Statements

Data is inserted into the tables to illustrate relationships.

Example:

```
INSERT INTO Students (StudentID, FirstName, LastName, DateOfBirth)
VALUES (1, 'Alice', 'Smith', '2000-05-15');
```

- **Purpose**: Populates the database with sample data for testing.

2.7 CREATE VIEW

Two views are created to simplify data queries:

- `StudentCourseView`: Shows students and their enrolled courses.
- `CourseProfessorView`: Shows courses and their assigned professors.

Example:

```
CREATE VIEW StudentCourseView AS
SELECT
   s.StudentID,
   s.FirstName AS StudentFirstName,
   s.LastName AS StudentLastName,
   c.CourseName,
   e.EnrollmentDate,
   e.Grade
FROM
   Students s
JOIN
   Enrollments e ON s.StudentID = e.StudentID
JOIN
   Courses c ON e.CourseID = c.CourseID;
```

- **Purpose**: Provides an abstracted way to query relationships between tables.

---

3. Concepts Covered

1. **Database Creation**: `CREATE DATABASE`.
2. **Table Creation**: `CREATE TABLE`.
3. **Primary and Foreign Keys**: Ensuring data integrity.
4. **ON DELETE and ON UPDATE**: Managing referential actions.
5. **Data Insertion**: Using `INSERT INTO`.
6. **Views**: Creating simplified query interfaces with `CREATE VIEW`.

---

4. Additional Notes

- **Indexing**: Primary and foreign keys implicitly create indexes, improving query performance.
- **Extensibility**: The script can be expanded with constraints like `UNIQUE`, `NOT NULL`, or `CHECK`.
- **Testing**: Run the script in a SQL environment like MySQL, PostgreSQL, or SQL Server to verify functionality.
