SQL Scripts - Uses and Applications

SQL Scripts

SQL scripts are a series of commands or a program that will be executed on an SQL server.

SQL scripts are useful for making complex database changes and can be used to create, modify, or delete database objects such as tables, views, stored procedures, and functions.

Applications of SQL Scripts

Here are some of the things that you can do with SQL scripts:

- **Create tables**
  You can use SQL scripts to create new tables in your database. This is useful when you need to add new functionality to your application or when you want to store new types of data.
- **Drop tables**
  SQL scripts often have commands to Drop tables from databases. This is especially important before Create table commands to make sure that a table with the same name doesnt exist in the database already.
- **Insert data**
  SQL scripts can also be used to insert data into your tables. This is useful when you need to populate your database with test data or when you want to import data from an external source.
- **Update data**
  You can use SQL scripts to update existing data in your tables. This is useful when you need to correct errors or update records based on changing business requirements.
- **Delete data**
  SQL scripts can also be used to delete data from your tables. This is useful when you need to remove old or obsolete records from your database.
- **Create views**
  Views are virtual tables that allow you to query data from multiple tables as if they were a single table. You can use SQL scripts to create views that simplify complex queries and make it easier to work with your data.
- **Create stored procedures**
  Stored procedures are precompiled SQL statements that can be executed on demand. You can use SQL scripts to create stored procedures that encapsulate complex business logic and make it easier to manage your database.
- **Create triggers**
  Triggers are special types of stored procedures that are automatically executed in response to certain events, such as an insert, update, or delete operation. You can use SQL scripts to create triggers that enforce business rules and maintain data integrity.

Example: Creating Tables

Let us execute a script containing the CREATE TABLE commands for all the tables in a given dataset, rather than create each table manually by typing the DDL commands in the SQL editor.

Note the following points about these scripts.

1. SQL scripts are basically a set of SQL commands compiled in a single file.
2. Each command must be terminated with a delimiter or terminator. Most often, the default delimiter is a semicolon ;.
3. It is advisable to keep the extension of the file as _.sql.
4. Upon importing this file in the phpMyAdmin interface, the commands in the file are run sequentially.

Consider the following script

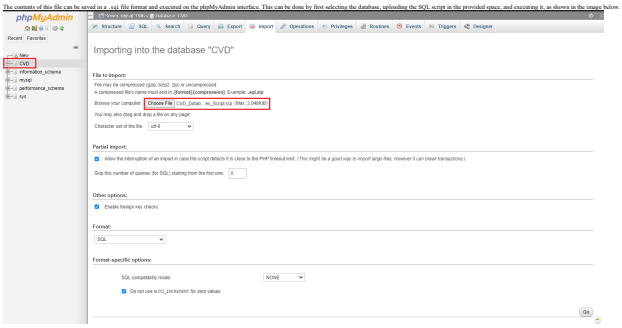
```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
39. 39
40. 40
41. 41
42. 42
43. 43
44. 44
45. 45
46. 46
47. 47
48. 48
49. 49
50. 50
51. 51
52. 52
```

```
1. DROP TABLE IF EXISTS PATIENTS;
2. DROP TABLE IF EXISTS MEDICAL_HISTORY;
3. DROP TABLE IF EXISTS MEDICAL_PROCEDURES;
4. DROP TABLE IF EXISTS MEDICAL_DEPARTMENTS;
5. DROP TABLE IF EXISTS MEDICAL_LOCATIONS;
6.
7.
8. CREATE TABLE PATIENTS (
9.   PATIENT_ID CHAR(9) NOT NULL,
10.  FIRST_NAME VARCHAR(15) NOT NULL,
11.  LAST_NAME VARCHAR(15) NOT NULL,
12.  SSN CHAR(9),
13.  BIRTH_DATE DATE,
14.  SEX CHAR(1),
15.  ADDRESS VARCHAR(30),
16.  DEPT_ID CHAR(9) NOT NULL,
17.  PRIMARY KEY (PATIENT_ID)
18. );
19.
20. CREATE TABLE MEDICAL_HISTORY (
21.   MEDICAL_HISTORY_ID CHAR(9) NOT NULL,
22.   PATIENT_ID CHAR(9) NOT NULL,
23.   DIAGNOSIS_DATE DATE,
24.   DIAGNOSIS_CODE VARCHAR(10),
25.   MEDICAL_CONDITION VARCHAR(100),
26.   DEPT_ID CHAR(9),
27.   PRIMARY KEY (MEDICAL_HISTORY_ID)
28. );
29.
30. CREATE TABLE MEDICAL_PROCEDURES (
31.   PROCEDURE_ID CHAR(9) NOT NULL,
32.   PROCEDURE_NAME VARCHAR(30),
33.   PROCEDURE_DATE DATE,
34.   PATIENT_ID CHAR(9) NOT NULL,
35.   PROVIDER CHAR(30),
36.   PRIMARY KEY (PROCEDURE_ID)
37. );
38.
39. CREATE TABLE MEDICAL_DEPARTMENTS (
40.   DEPT_ID CHAR(9) NOT NULL,
41.   DEPT_NAME VARCHAR(15),
42.   LOCATION CHAR(10),
43.   LOCATION_ID CHAR(9),
44.   PRIMARY KEY (DEPT_ID)
45. );
46.
47. CREATE TABLE MEDICAL_LOCATIONS (
48.   LOCATION_ID CHAR(9) NOT NULL,
49.   DEPT_ID CHAR(9) NOT NULL,
50.   LOCATION_NAME VARCHAR(10),
51.   PRIMARY KEY (LOCATION_ID, DEPT_ID)
52. );
```

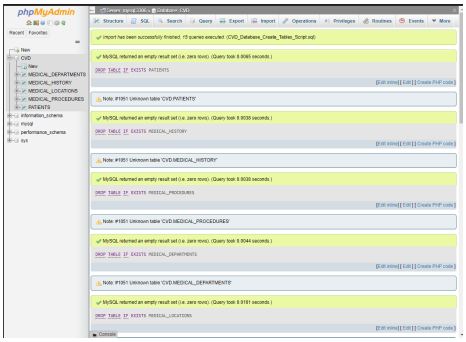
Copied!

This script incorporates commands to first drop any tables with the mentioned names in the database. After that, the script contains commands to create 5 different tables. All these commands are executed sequentially on the interface.

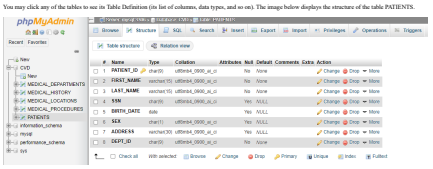
The contents of this file can be saved in a .sql file format and executed on the phpMyAdmin interface. This can be done by first selecting the database, uploading the SQL script in the provided space, and executing it, as shown in the image below.



Upon successful execution of each statement in sequence, an note appears on the interface as shown in the image below. It is also prudent to note that the tables created are now visible in the tree structure on the left under the selected database.



You may click any of the tables to see its Table Definition (its list of columns, data types, and so on). The image below displays the structure of the table PATIENTS.



Author(s)

Abhishek Gagneja