Task Description

Project:

Database Setup and Configuration

Description:

- Create a simple database using a popular database management system (e.g., MySQL, PostgreSQL).
- Perform basic configuration settings, including setting up a user, defining tables, and establishing basic relationships.

Skills Emphasized:

- Database creation, user management, basic schema design.

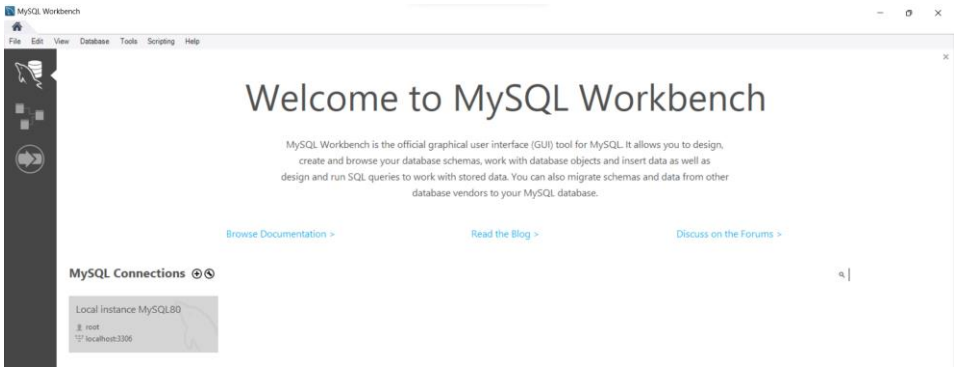
Task 1

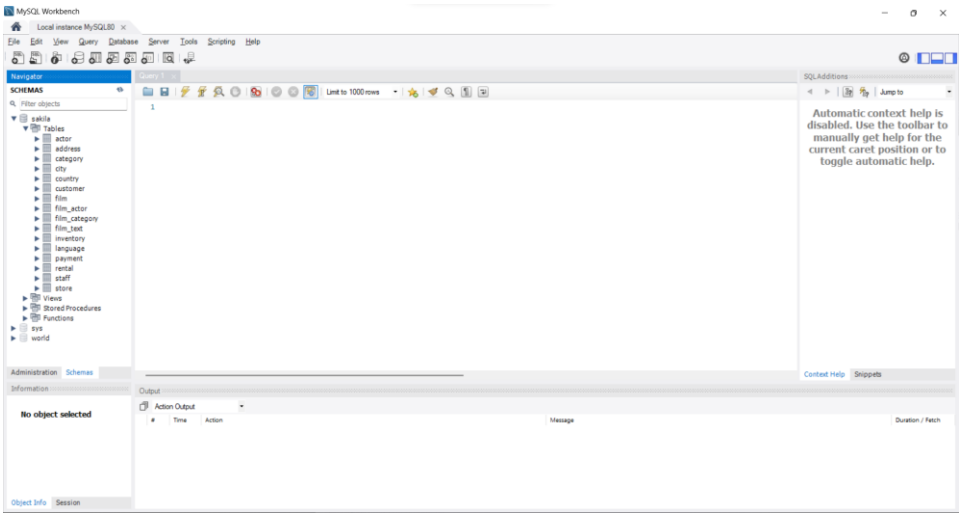
Create a simple database using a popular database management system (e.g., MySQL, PostgreSQL).

Step 1: MySQL setup

In order to create a database, we need a database management system installed on our system, which in my case is MySQL. Then, we open MySQL Workbench which is installed after we install MySQL in our system.

(Note: MySQL Workbench is a GUI tool for MySQL which allows to design, create and browse database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data.)

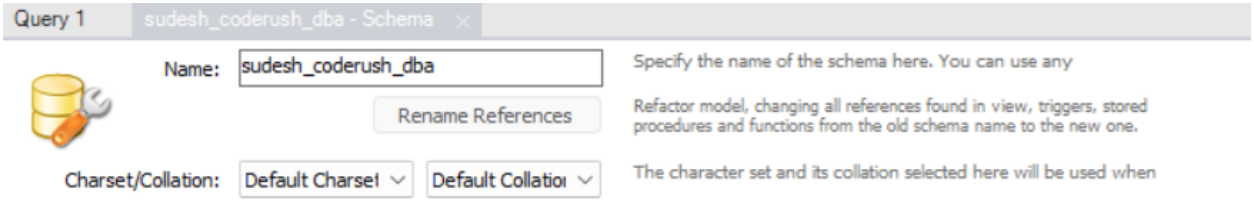




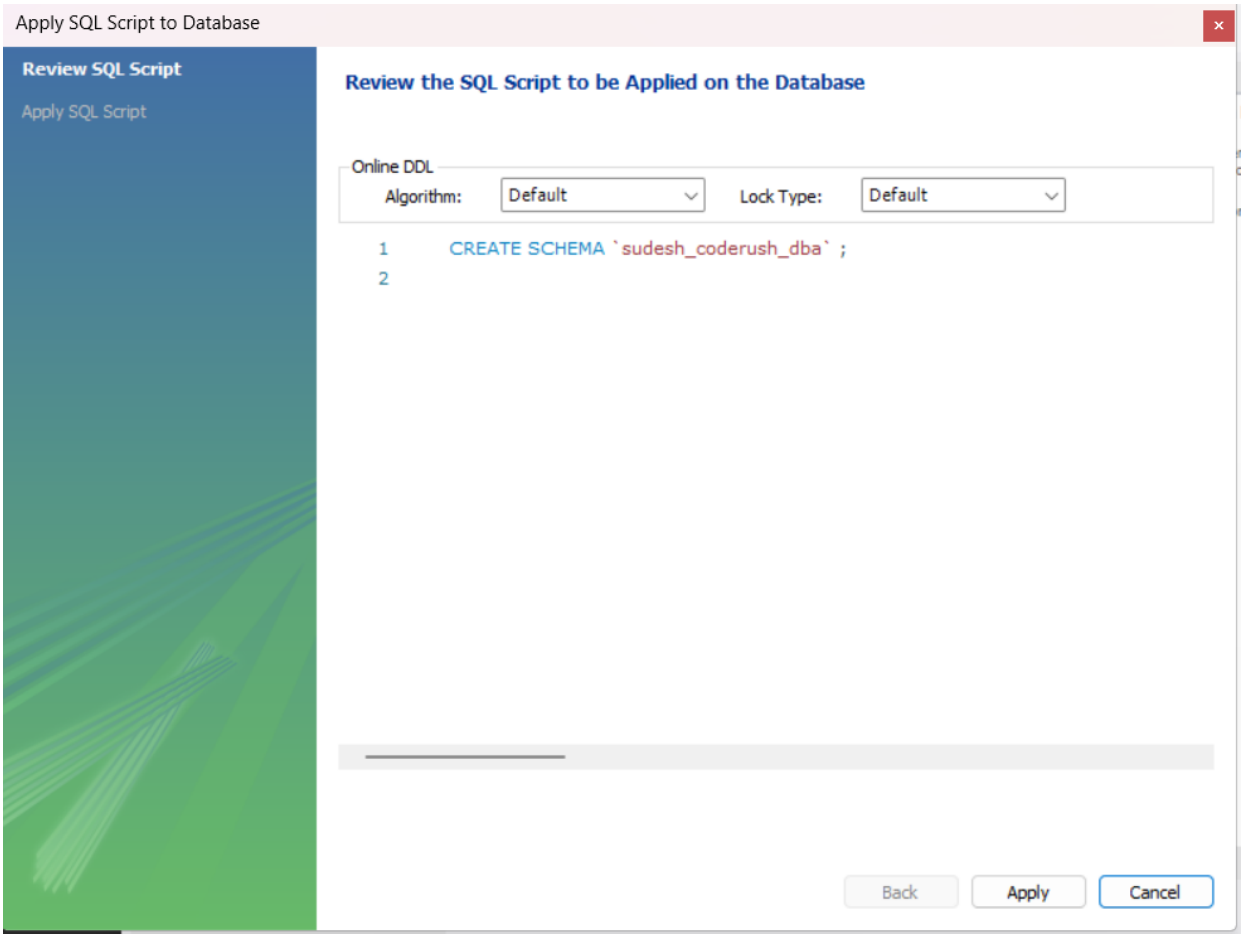
Step 2: Database Creation

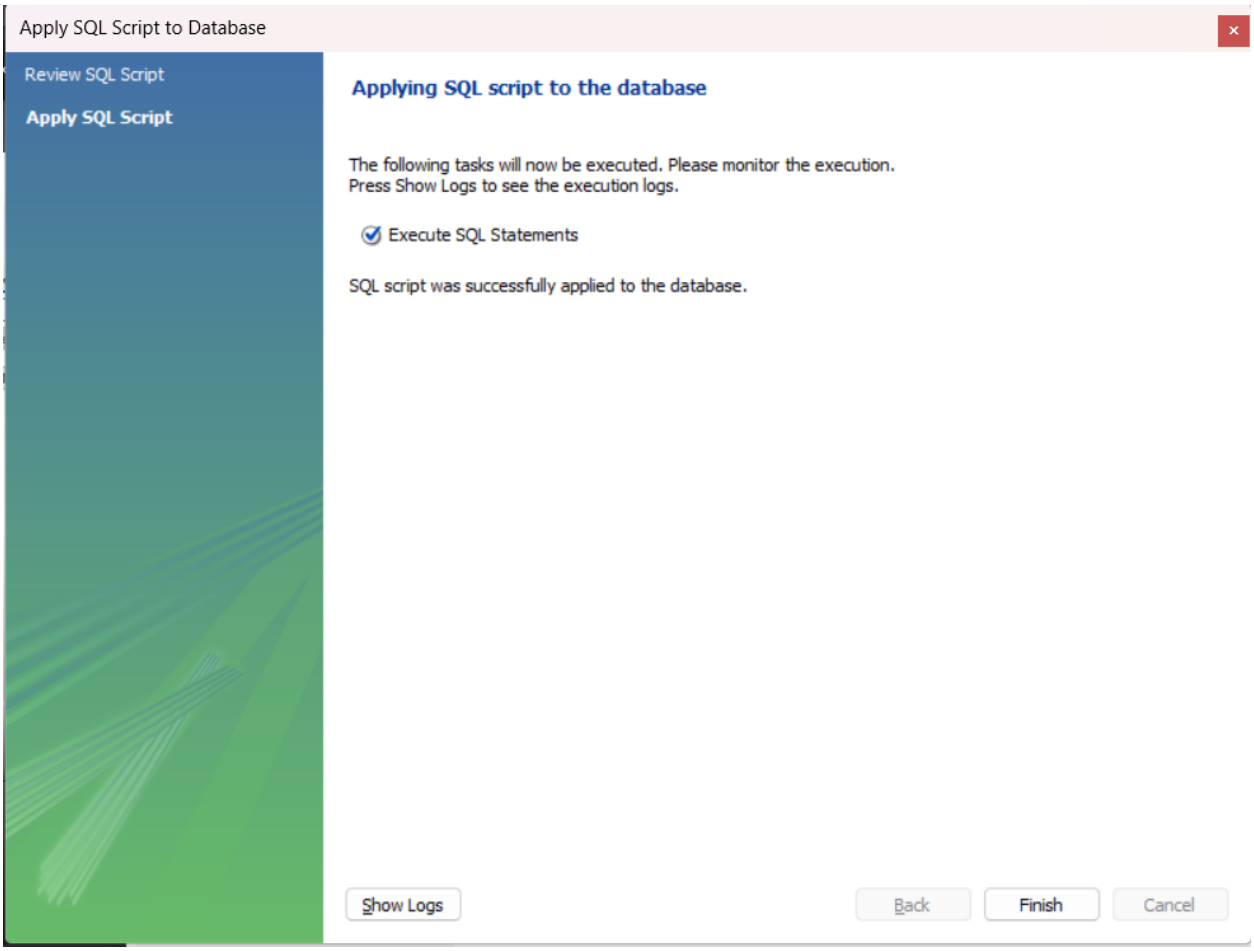
As seen on the screenshot above, we can see some sample databases that are present in default. In order to create our own database, we click on  icon on the menu bar, which says “New Schema”. It creates a new schema on the connected server.

Then, we name the schema (database) as per required.

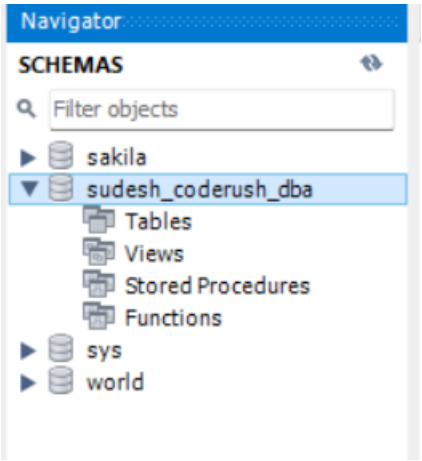


Then a pop-up will be displayed which asks to apply SQL Script to Database.





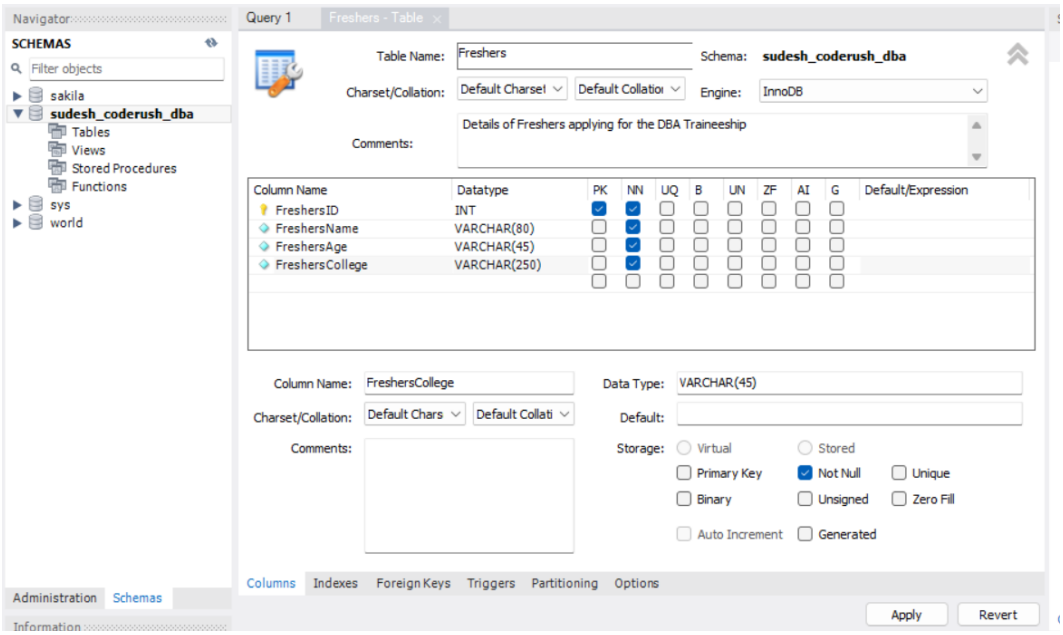
After applying the Script, we get our new database present under the Schemas Section.



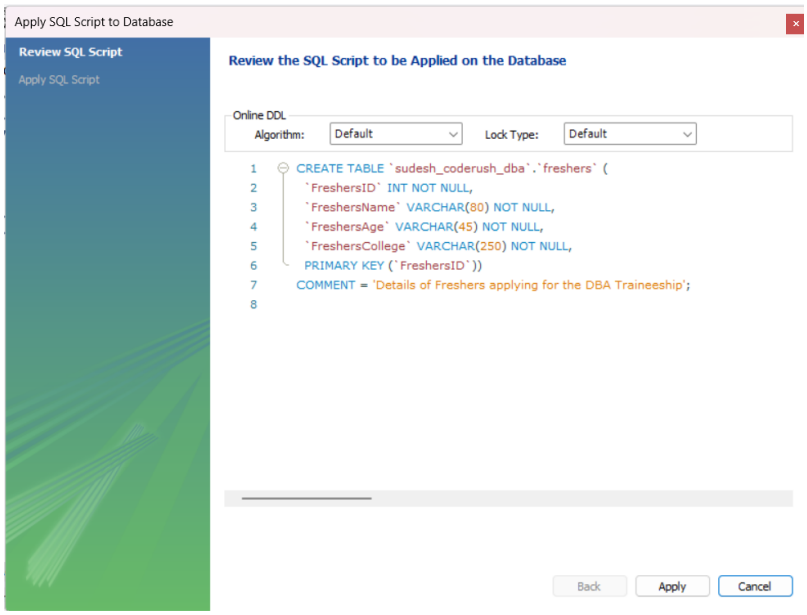
Step 3: Table Creation

Now, we while the new schema we just created is active, we insert table by clicking on  icon on the menu bar.

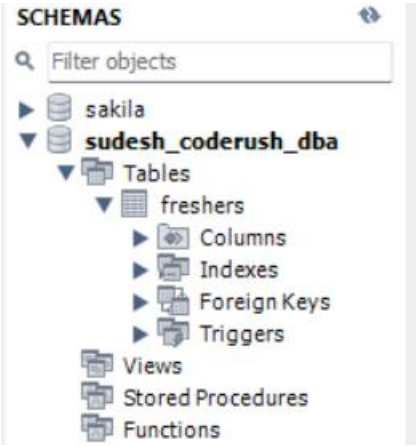
A tab will be displayed in which we need to fill in the details such as the name of the table, the attributes that the table contains along with the data types of those attributes and constraint keys which specifies the types of data, as shown in the screenshot below.



After applying the changes, a pop up will be displayed to confirm the changes as:

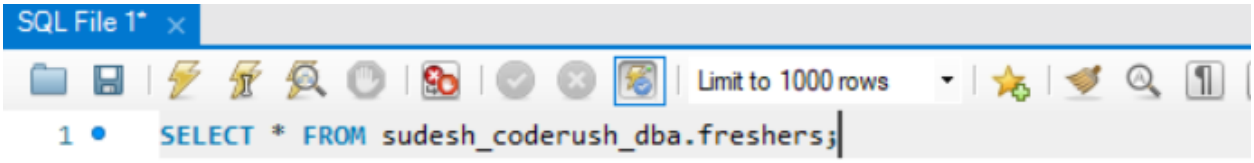


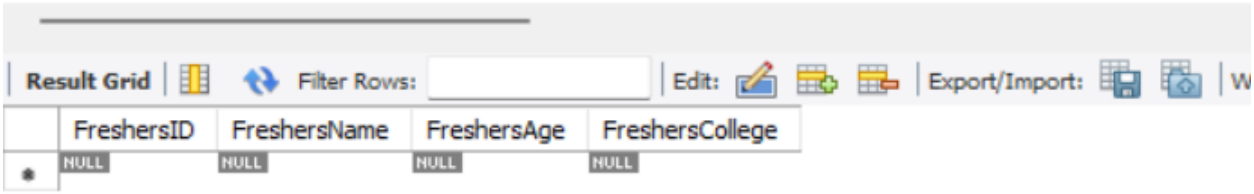
After applying the changes, we get a new set of table named ‘freshers’ in our database.



The table can be accessed with the following query as:

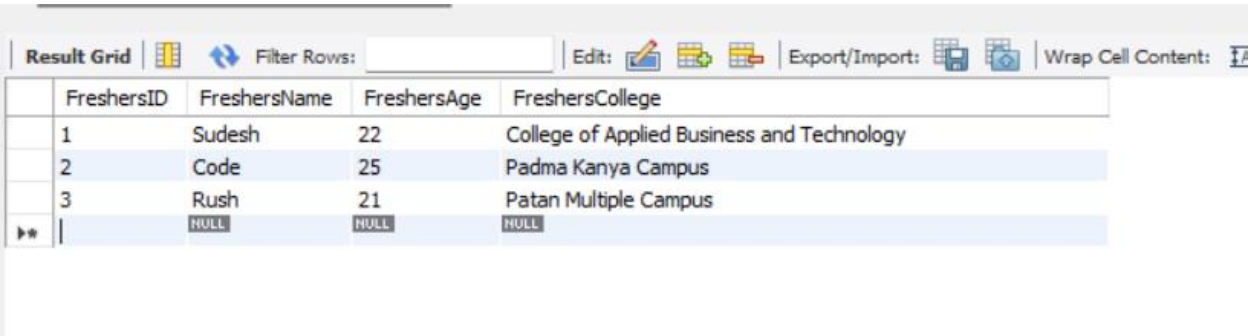
SELECT * FROM sudesh_coderush_dba.freshers;





Step 4: Data Insertion

Now, we manually insert data into the columns of the table as:

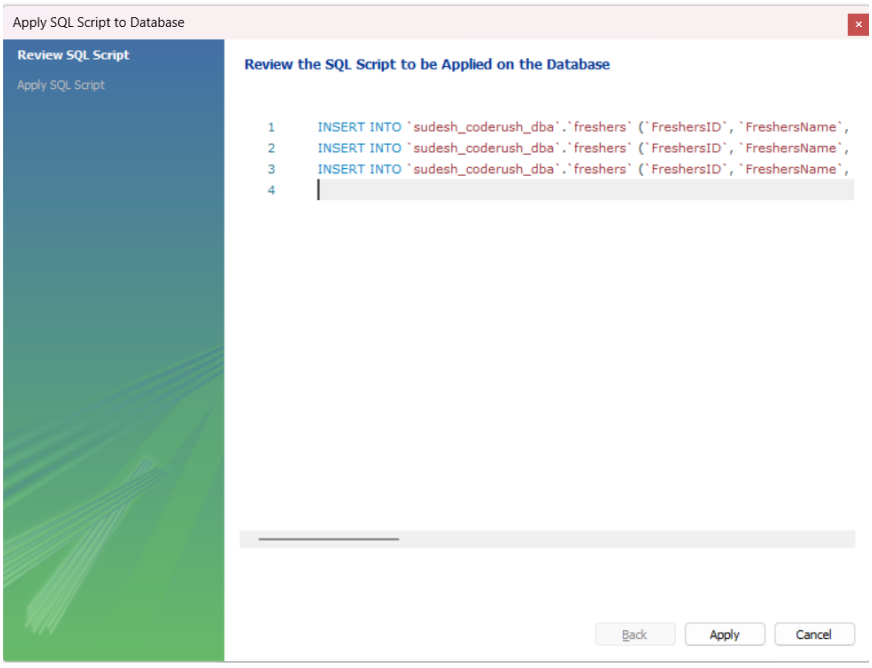


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

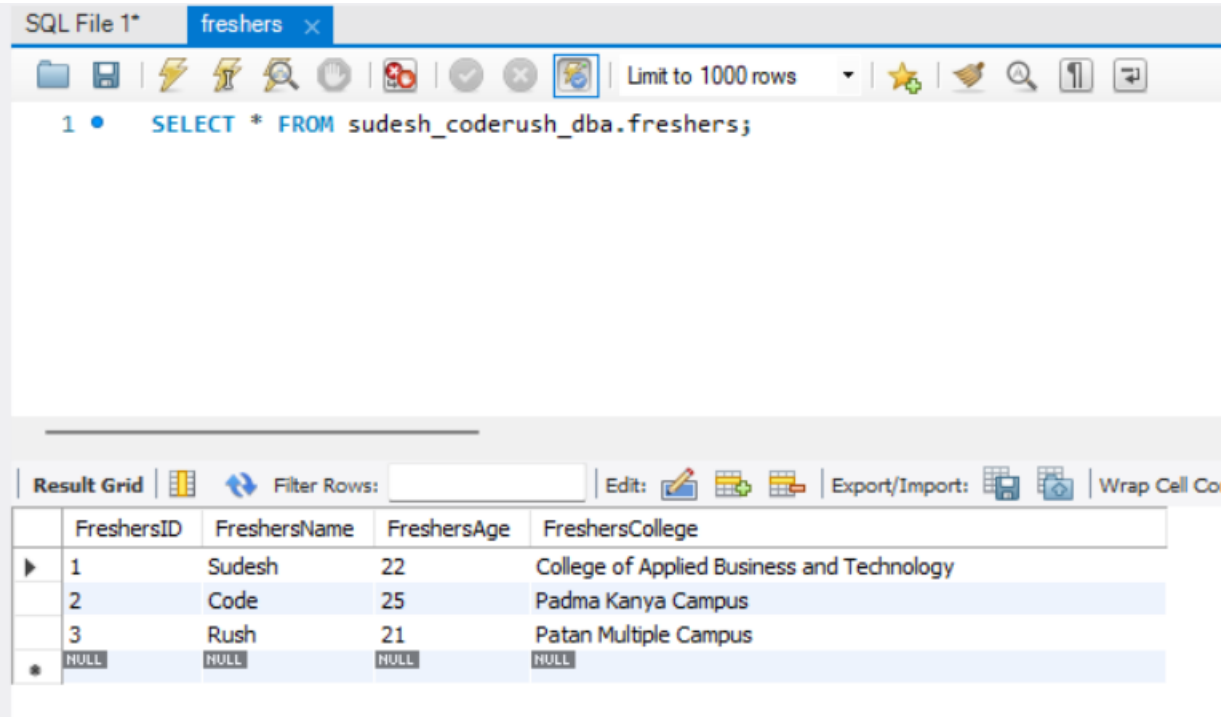
FreshersID	FreshersName	FreshersAge	FreshersCollege
1	Sudesh	22	College of Applied Business and Technology
2	Code	25	Padma Kanya Campus
3	Rush	21	Patan Multiple Campus
NULL	NULL	NULL	NULL

This will ask for confirmation after applying the changes as:

INSERT INTO `sudesh_coderush_dba`.`freshers` (`FreshersID`, `FreshersName`, `FreshersAge`, `FreshersCollege`) VALUES ('1', 'Sudesh', '22', 'College of Applied Business and Technology');
INSERT INTO `sudesh_coderush_dba`.`freshers` (`FreshersID`, `FreshersName`, `FreshersAge`, `FreshersCollege`) VALUES ('2', 'Code', '25', 'Padma Kanya Campus');
INSERT INTO `sudesh_coderush_dba`.`freshers` (`FreshersID`, `FreshersName`, `FreshersAge`, `FreshersCollege`) VALUES ('3', 'Rush', '21', 'Patan Multiple Campus');



This creates a table with the given values by the user. After we access the table, this is what we will be shown.



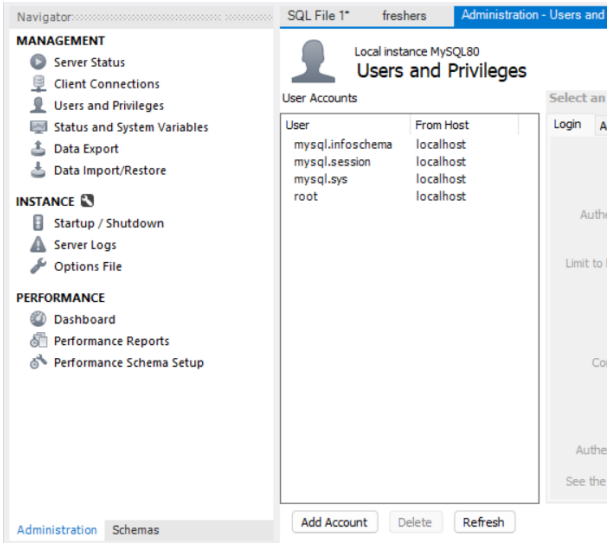
In the same way, we can create more tables in the database.

Task 2

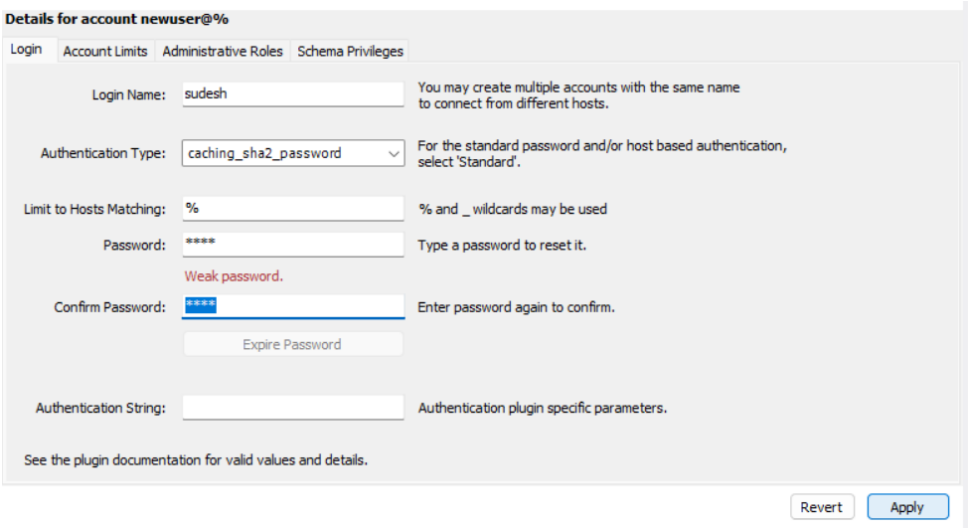
Perform basic configuration settings, including setting up a user, defining tables, and establishing basic relationships.

User Creation:

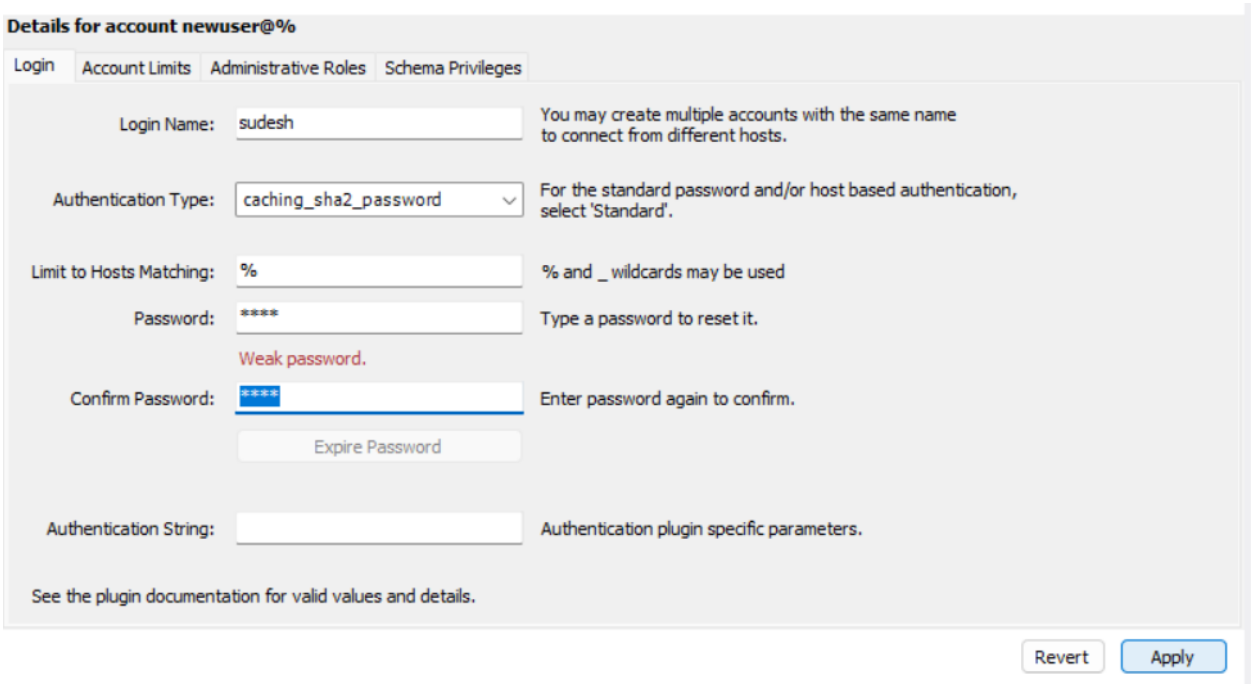
Under the ‘Administration’ tab, we can see an option called “Users and Privileges” under “Management”. After clicking on it, the users are displayed as:



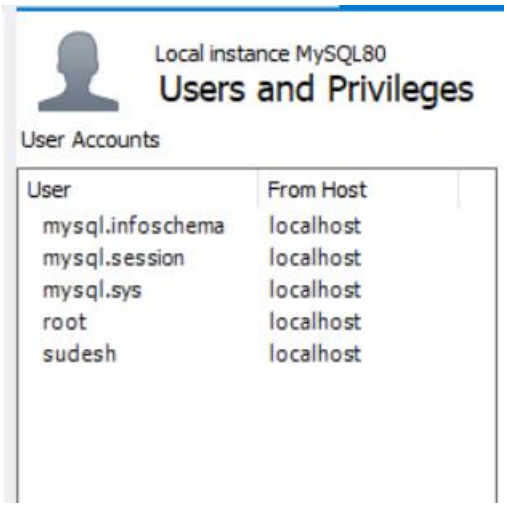
Now, in order to create a new user, we click on “Add Account” option under the Users and Privileges tab, which displays:



Here, we enter the name for our new user, along with the authentication type and a password for the new user and click on apply to save the new user.



Now, this would show the new user in the users list.

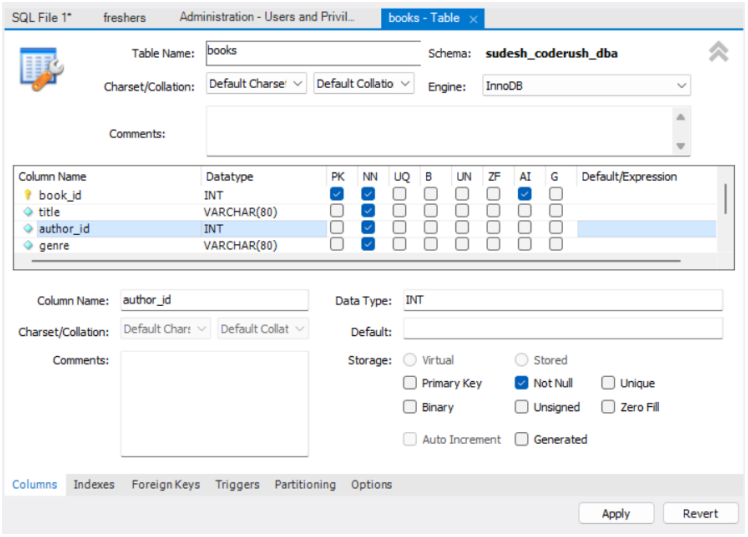


Defining tables:

We create a tables in the database following the steps mentioned in Task-1.

Let us create two tables named ‘books’ and ‘authors’ which makes use of the following SQL Commands:

Table: ‘books’:

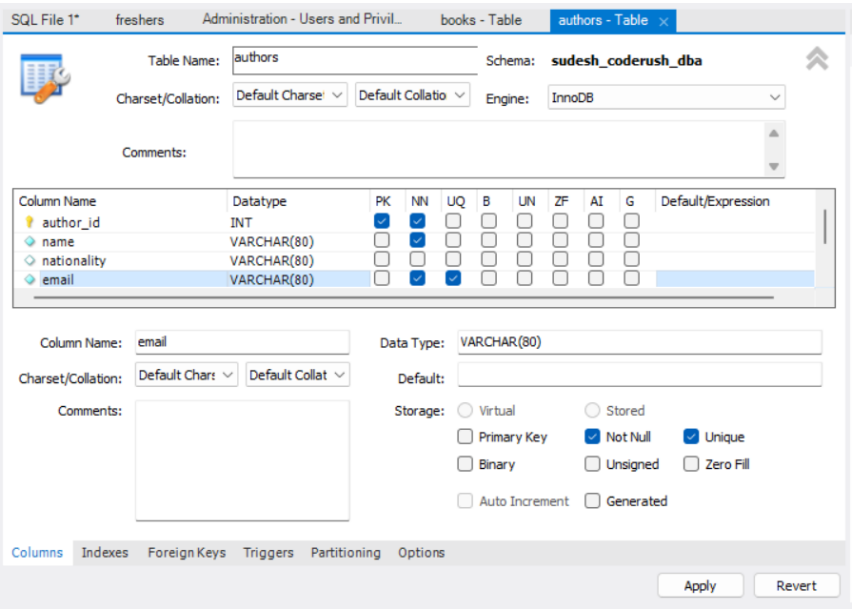



```
CREATE TABLE `sudesh_coderush_dba`.`books` (
  `book_id` INT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(80) NOT NULL,
  `author_id` INT NOT NULL,
  `genre` VARCHAR(80) NOT NULL,
  PRIMARY KEY (`book_id`));
```

Where,

- **book_id:** Unique identifier for each book, set as the primary key with AUTO_INCREMENT.
- **title:** The title of the book, a mandatory field (NOT NULL).
- **author_id:** An integer field to store the ID of the author who wrote the book. It is also a mandatory field (NOT NULL).
- **genre:** The genre of the book, a mandatory field (NOT NULL).

Table: 'authors':



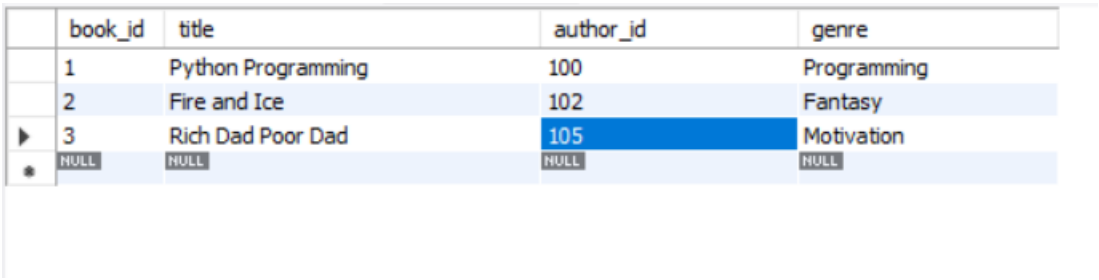

```
CREATE TABLE `sudesh_coderush_dba`.`authors` (
`author_id` INT NOT NULL,
`name` VARCHAR(80) NOT NULL,
`nationality` VARCHAR(80) NULL,
`email` VARCHAR(80) NOT NULL,
PRIMARY KEY (`author_id`),
UNIQUE INDEX `email_UNIQUE` (`email` ASC) VISIBLE);
```

Where,

- **author_id:** Unique identifier for each author, set as the primary key with AUTO_INCREMENT.
- **name:** The name of the author, a mandatory field (NOT NULL).
- **nationality:** The nationality of the author.(NOT NULL)
- **email:** The email address of the author, which must be unique. (NOT NULL)

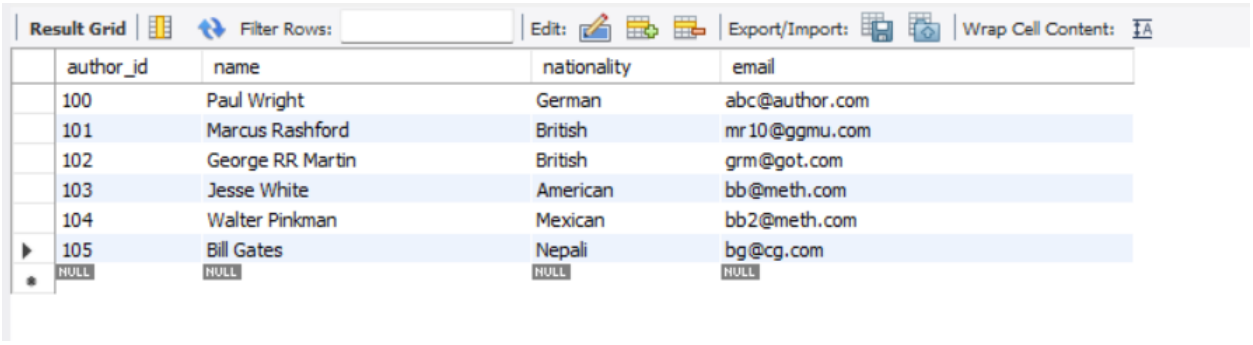
These tables provides a basic structure for storing information about books and authors in the database. After the creation of the tables, we insert values in the columns of the tables as:

Table: books:



book_id	title	author_id	genre
1	Python Programming	100	Programming
2	Fire and Ice	102	Fantasy
3	Rich Dad Poor Dad	105	Motivation
NULL	NULL	NULL	NULL

Table: authors:



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

author_id	name	nationality	email
100	Paul Wright	German	abc@author.com
101	Marcus Rashford	British	mr10@ggmu.com
102	George RR Martin	British	grm@got.com
103	Jesse White	American	bb@meth.com
104	Walter Pinkman	Mexican	bb2@meth.com
105	Bill Gates	Nepali	bg@cg.com
NULL	NULL	NULL	NULL

To view the table, we use the following SQL commands:

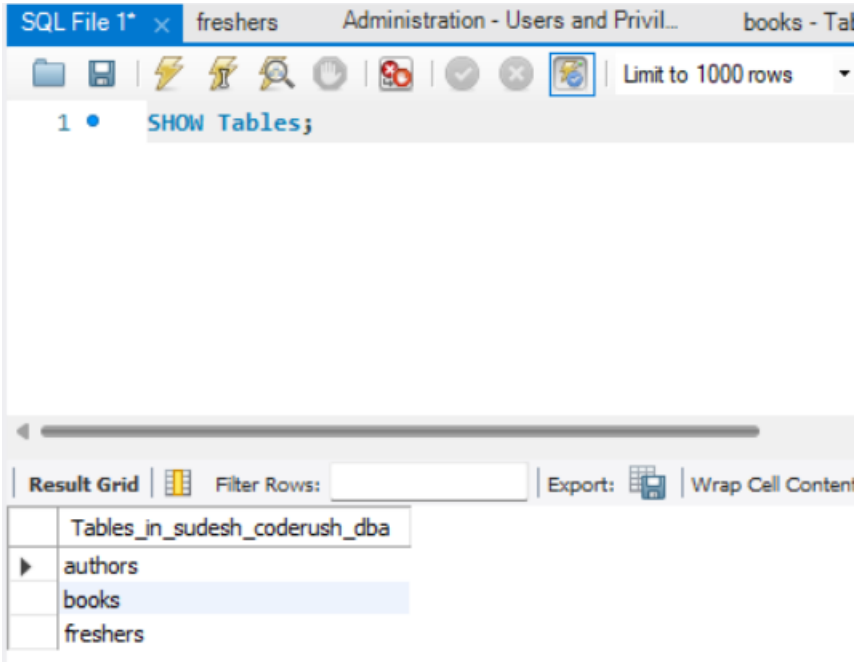
For 'books': SELECT * FROM sudesh_coderush_dba.books;

For 'authors': SELECT * FROM sudesh_coderush_dba.authors;

Establishing relationships between the tables: (GUI Method using MySQL)

In order to establish relationships between the tables, first of all we take a look at the tables present in our database using the 'SHOW Tables;' command.

In our database, the result will be like this:

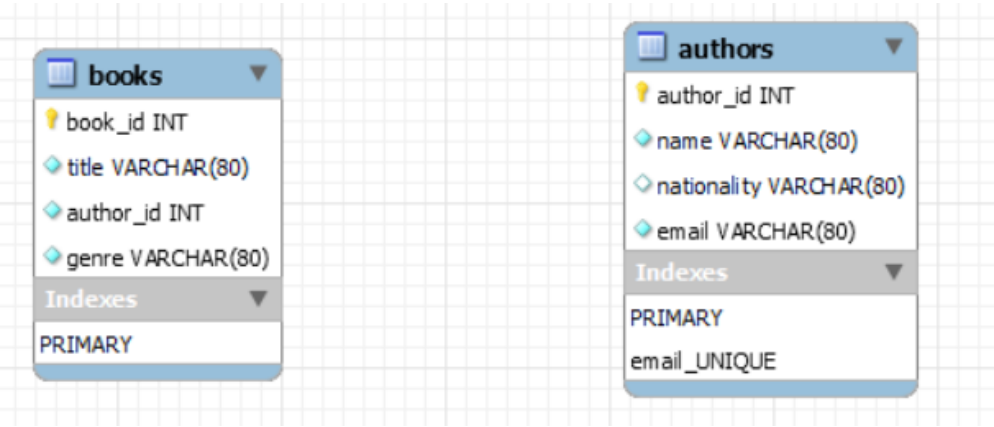


Here, we can see that we have three tables that we have previously created.

However, we will be establishing relationships between two tables "authors" and "books".

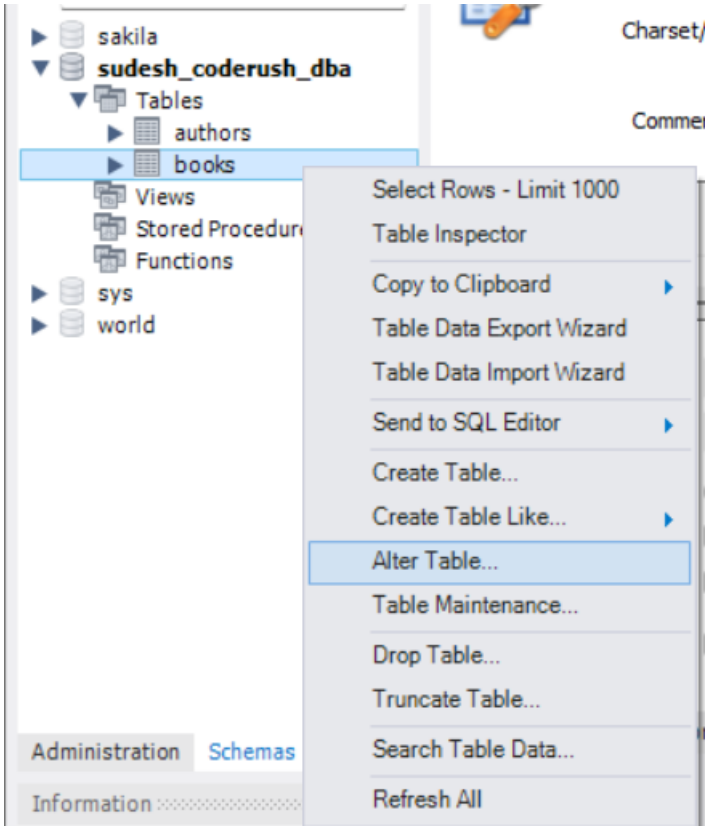
So, let us begin by deleting the entire "freshers" table using the command "DROP TABLE freshers". This deletes the table 'freshers' from the database.

Now, the remaining tables in our database and the relationship between them is shown by an ER Diagram as below:

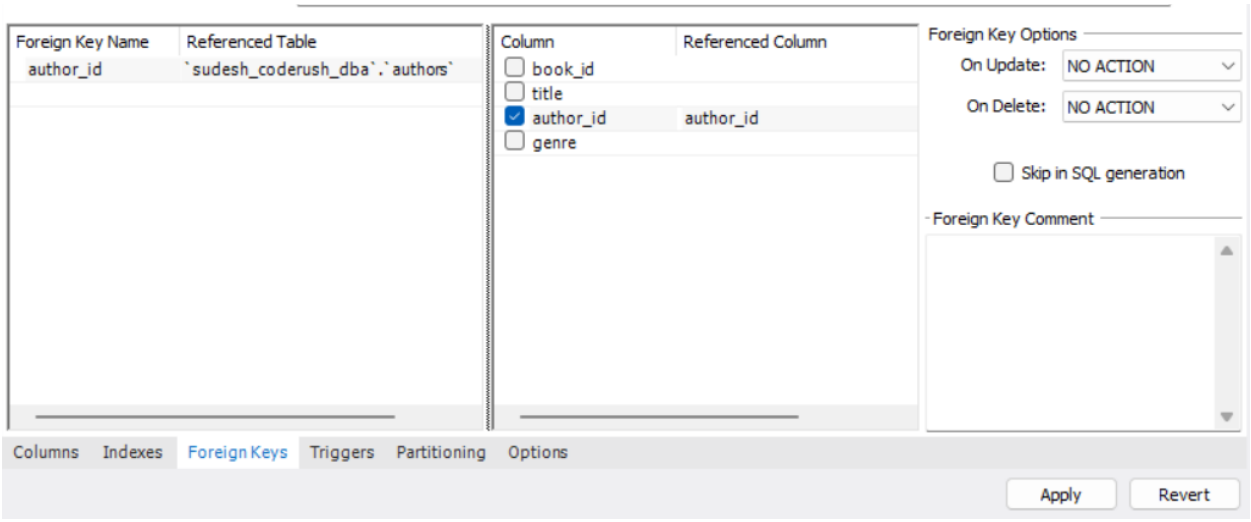


Now, in order to establish relation between these two tables, we need to alter the constraints of the columns of the table.

At first, we right-click on the table ‘books’ to and select “Alter Table” option:



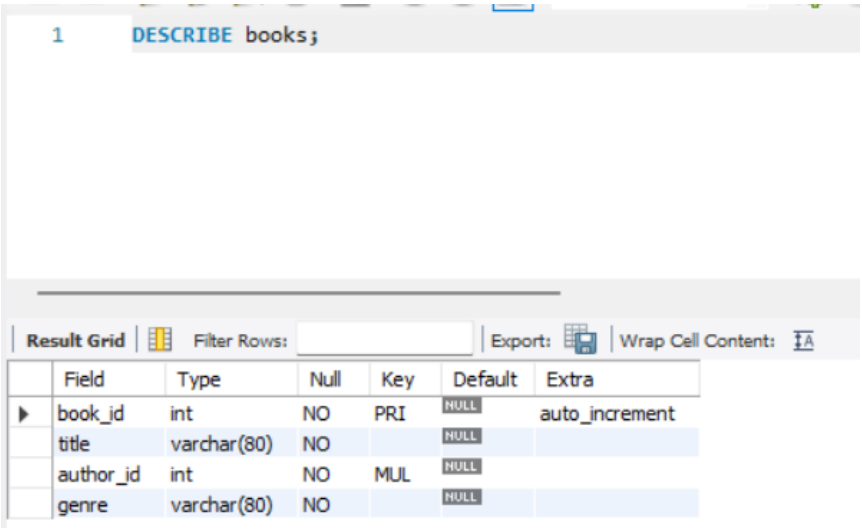
Then, under the ‘Foreign Keys’ tab, we specify the details of the foreign key from the table books that is referenced to the column of authors table.



The SQL command for this process is:

```
ALTER TABLE `sudesh_coderush_dba`.`books`
ADD INDEX `author_id_idx` (`author_id` ASC) VISIBLE;
;
ALTER TABLE `sudesh_coderush_dba`.`books`
ADD CONSTRAINT `author_id`
 FOREIGN KEY (`author_id`)
 REFERENCES `sudesh_coderush_dba`.`authors` (`author_id`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION;
```

After this, we can see that the column ‘author_id’ in our table is now a foreign key.



Field	Type	Null	Key	Default	Extra
book_id	int	NO	PRI	NULL	auto_increment
title	varchar(80)	NO		NULL	
author_id	int	NO	MUL	NULL	
genre	varchar(80)	NO		NULL	

It links the table to another table in a one to one relationship as shown in the ER Diagram below: