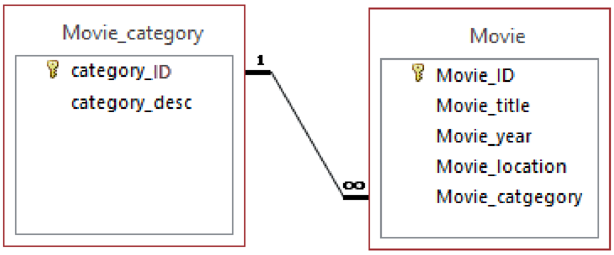
Database MDI Tutorial - Part 1

Movies database

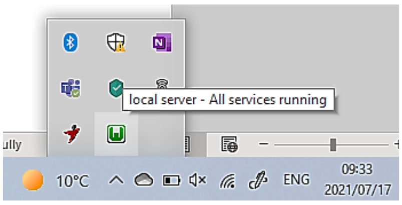
In this first part of a number of a tutorials, we will start to create a MySQL database that will contain a table with list of movies as well as a table that will contain the categories according to which the movies will be classified. The primary key of the movie categories table will then be used as a foreign key in the movies table.

An Entity Relationship Diagram shows the relationship between the tables and the various fields and tables.



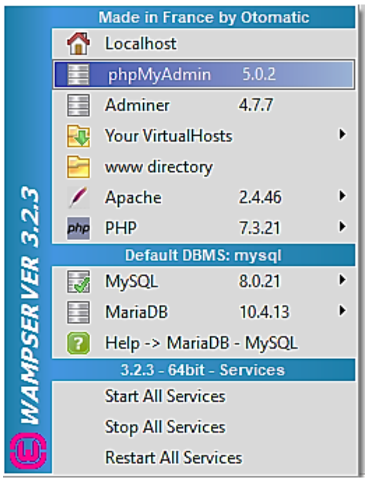
Step 1 Open the WAMPserver phpAdmin MySQL Console

- Open the WAMPserver phpAdmin MySQL Console by clicking on the icon in the windows tray

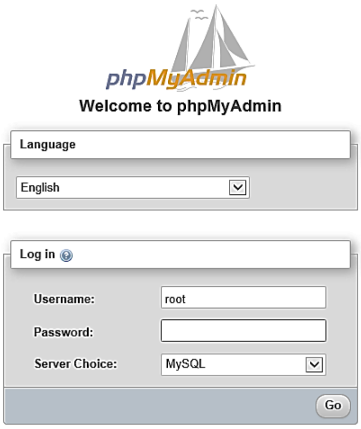


- If the icon does not appear in the tray, start the server by clicking on the icon in start menu

- Click on the phpAdmin option



- If you set a password during setup enter it and click on Go otherwise just leave it blank and click on Go



Language: English

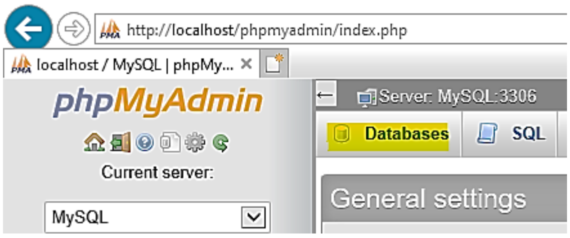
Username: root

Password:

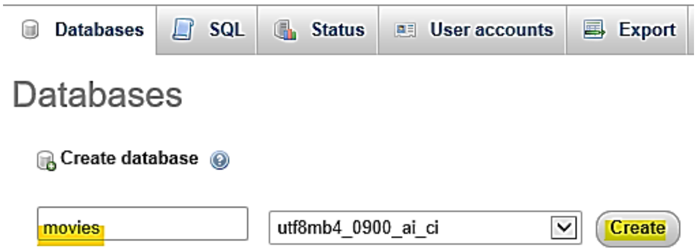
Server Choice: MySQL

Step 2 Create the database

- We want to create a database called movies so click on the Databases link at the top of the screen



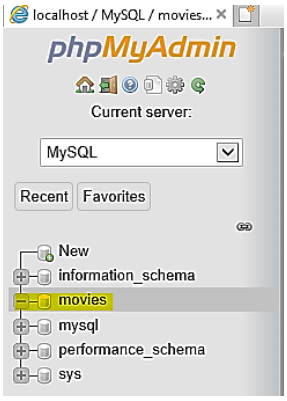
- Enter *movies* in the textbox and click on the Create button



movies

utf8mb4_0900_ai_ci

- The database will now appear in the list of databases for this server in the left pane

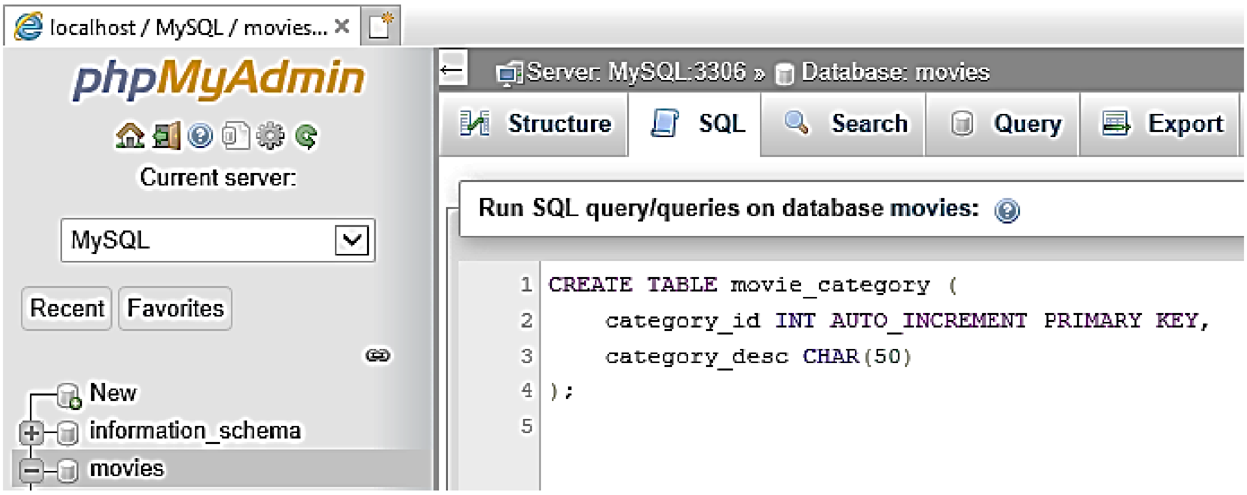


Step 3 Create a table for movie categories

- We want to create a table called `movie_category` that will contain two fields, so enter it in the textboxes as per the screenshot as below and click on the Go button:

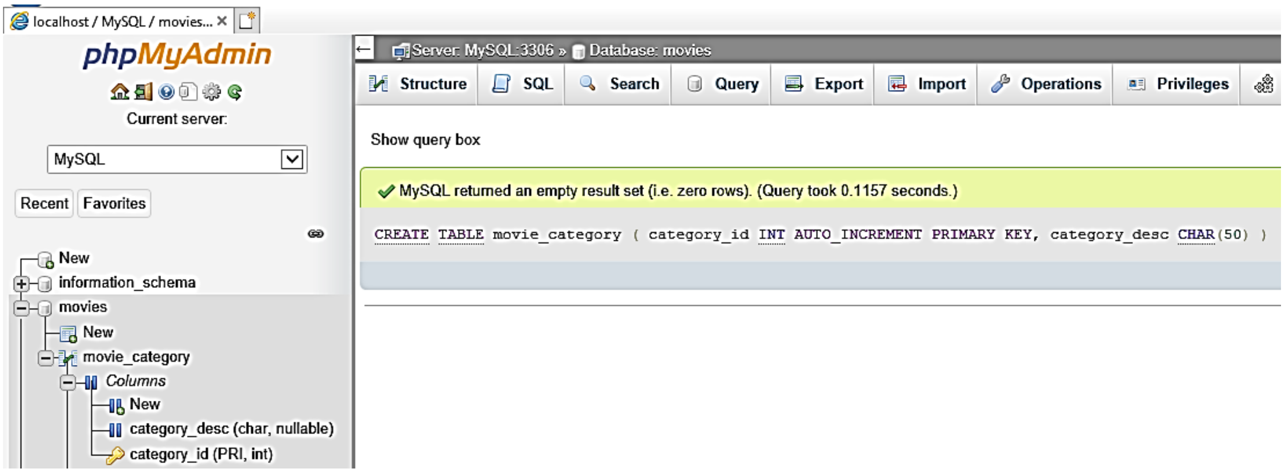
- Click on the SQL link and enter the following statement to create a table called `movie_category` that will contain the description of categories of movies

```
CREATE TABLE movie_category (
      category_id INT AUTO_INCREMENT PRIMARY KEY,
      category_desc CHAR(50)
);
```


localhost / MySQL / movies...
phpMyAdmin
Current server:
MySQL
Recent
Favorites
New
information_schema
movies
Server: MySQL:3306 » Database: movies
Structure
SQL
Search
Query
Export
Run SQL query/queries on database movies:

```
CREATE TABLE movie_category (
    category_id INT AUTO_INCREMENT PRIMARY KEY,
    category_desc CHAR(50)
);
```

- Click on the Go button to execute the query. The following screen should appear to indicate that the query executed successfully and the movie_category table and the fields should be displayed within the movie database in the left pane



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.1157 seconds.)

```
CREATE TABLE movie_category ( category_id INT AUTO_INCREMENT PRIMARY KEY, category_desc CHAR(50) )
```

- We created a primary key called `category_id` that will be an integer value and that will increase automatically each time a new record is added

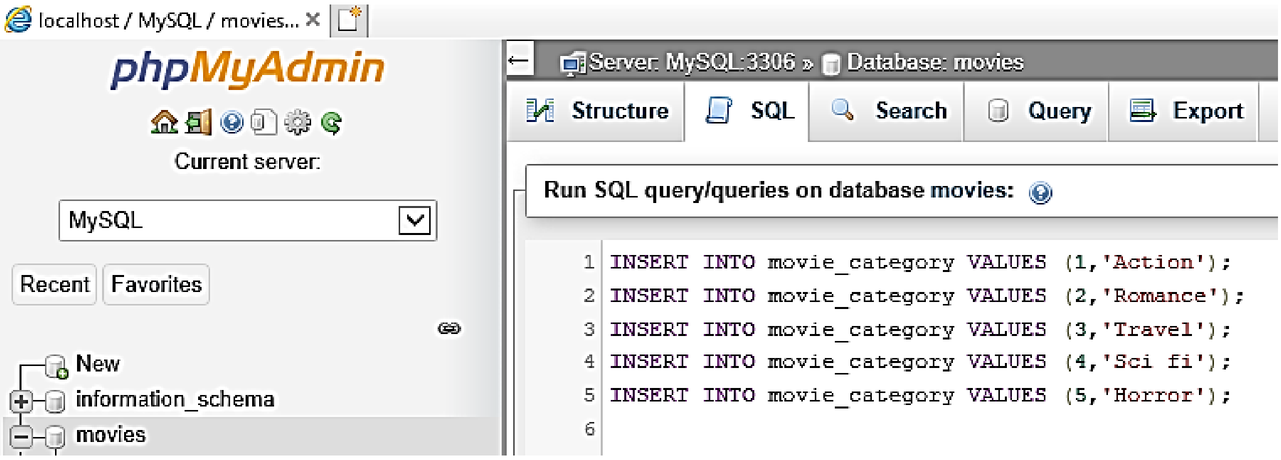
- We also added a description field called `category_desc` that can be a maximum of 50 characters

Step 4 Add records into the movie_category table

- We will add records into the table use SQL statements for simplicity, of course in a full application you would allow a user to add records into the table using a window with widgets

- Add a view records into the `movie_category` table by entering the following statements in the SQL tab and clicking on the Go button

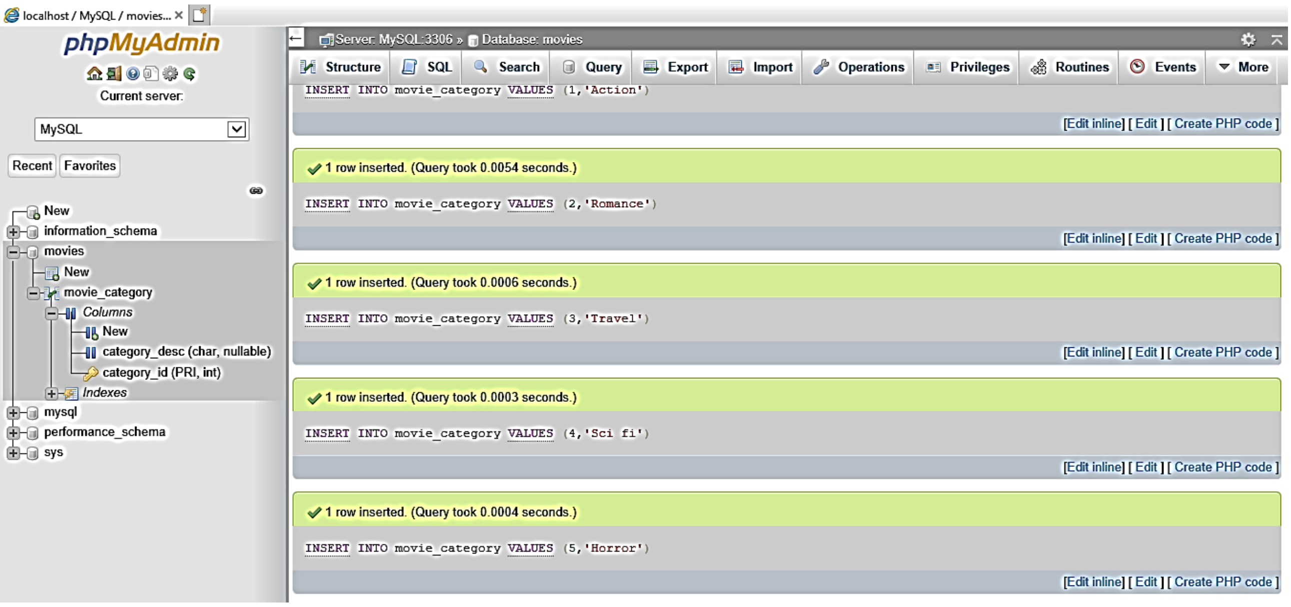
```
INSERT INTO movie_category VALUES (1,’Action’);
INSERT INTO movie_category VALUES (2,’Romance’);
INSERT INTO movie_category VALUES (3,’Travel’);
INSERT INTO movie_category VALUES (4,’Sci fi’);
INSERT INTO movie_category VALUES (5,’Horror’);
```



Run SQL query/queries on database movies:

```
INSERT INTO movie_category VALUES (1,'Action');
INSERT INTO movie_category VALUES (2,'Romance');
INSERT INTO movie_category VALUES (3,'Travel');
INSERT INTO movie_category VALUES (4,'Sci fi');
INSERT INTO movie_category VALUES (5,'Horror');
```

The following output should appear to confirm that the statements were executed



```
INSERT INTO movie_category VALUES (1,'Action')
```

✓ 1 row inserted. (Query took 0.0054 seconds.)

```
INSERT INTO movie_category VALUES (2,'Romance')
```

✓ 1 row inserted. (Query took 0.0006 seconds.)

```
INSERT INTO movie_category VALUES (3,'Travel')
```

✓ 1 row inserted. (Query took 0.0003 seconds.)

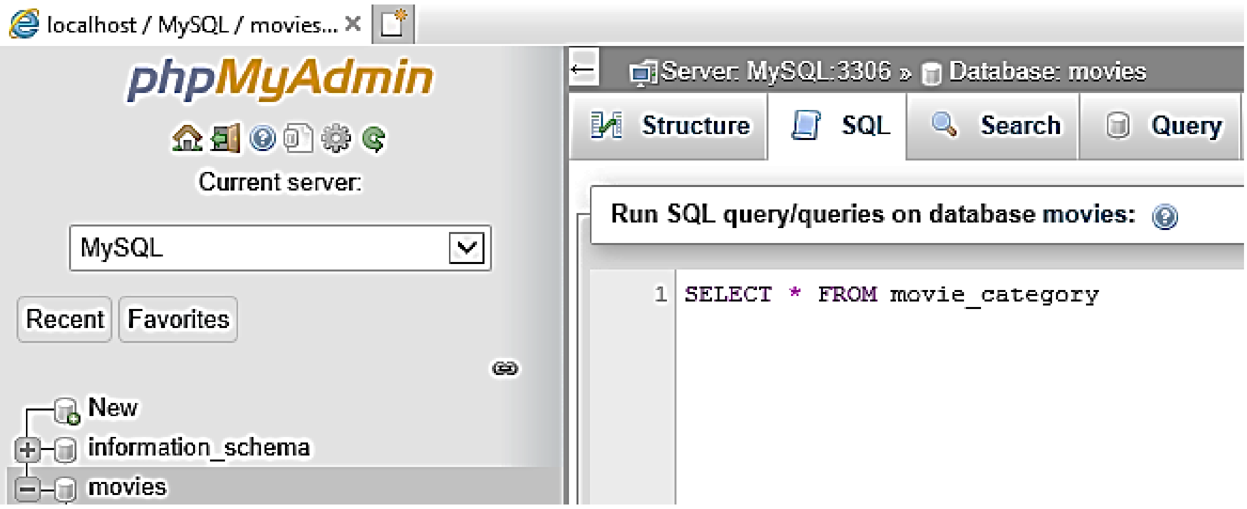
```
INSERT INTO movie_category VALUES (4,'Sci fi')
```

✓ 1 row inserted. (Query took 0.0004 seconds.)

```
INSERT INTO movie_category VALUES (5,'Horror')
```

- Use a select statement to check the contents of the `movie_category` table by adding the following statement in the SQL tab

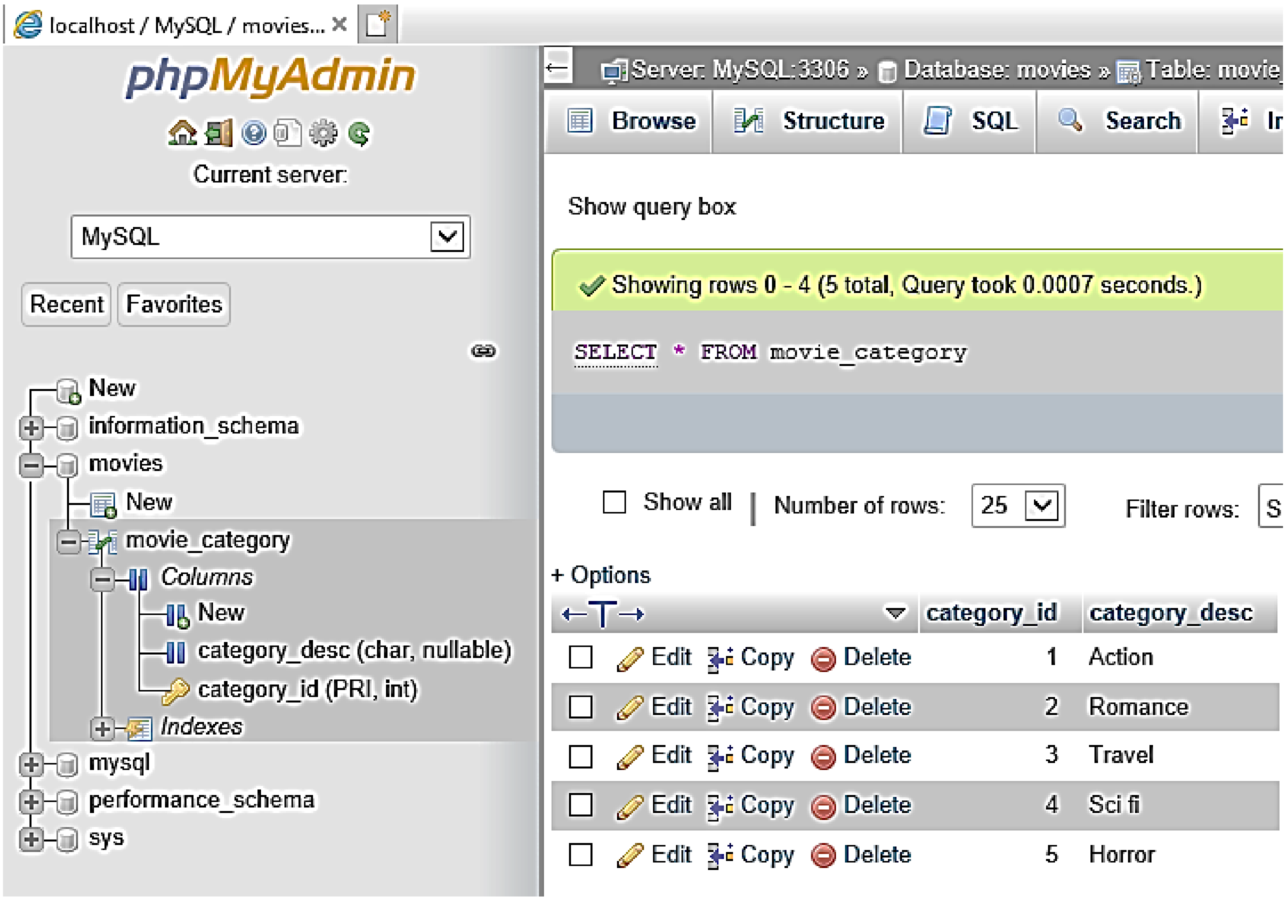
```
SELECT * FROM movie_category
```



Run SQL query/queries on database movies:

```
SELECT * FROM movie_category
```

- You should get the following output showing that all five the movie categories have been added to the table



✓ Showing rows 0 - 4 (5 total, Query took 0.0007 seconds.)

```
SELECT * FROM movie_category
```

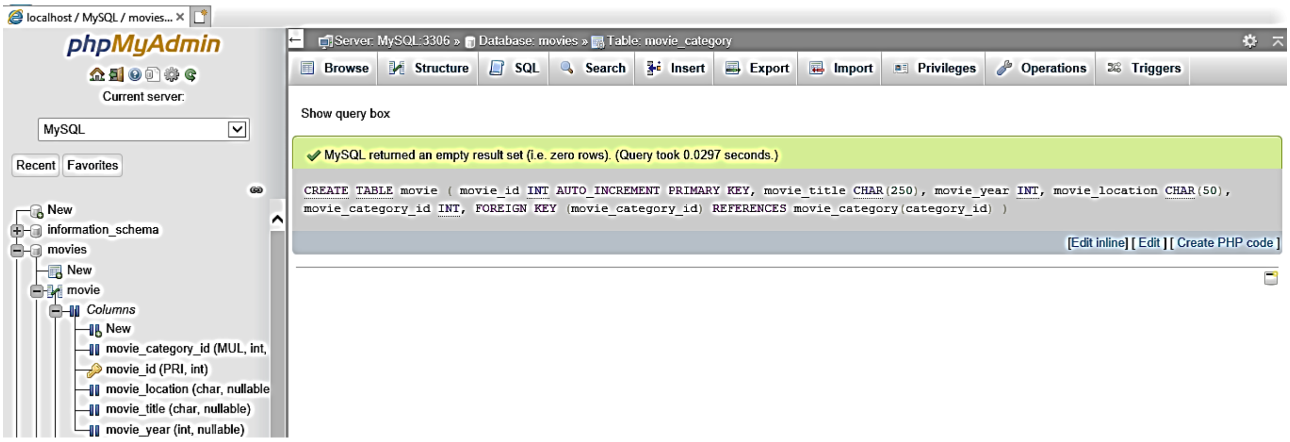
	category_id	category_desc
□ Edit Copy Delete	1	Action
□ Edit Copy Delete	2	Romance
□ Edit Copy Delete	3	Travel
□ Edit Copy Delete	4	Sci fi
□ Edit Copy Delete	5	Horror

Step 5 Create a table for movies

- Enter the following statement in the SQL to create a table called `movie` that will contain the details of movies and click the Go button

```
CREATE TABLE movie (
        movie_id INT AUTO_INCREMENT PRIMARY KEY,
        movie_title CHAR(250),
        movie_year INT,
        movie_location CHAR(50),
        movie_category_id INT,
        FOREIGN KEY (movie_category_id) REFERENCES movie_category(category_id)
);
```

- You should get this result:


localhost / MySQL / movies...
phpMyAdmin
Current server:
MySQL
Recent
Favorites
New
information_schema
movies
New
movie
Columns
New
movie_category_id (MUL, int,
movie_id (PRI, int)
movie_location (char, nullable
movie_title (char, nullable)
movie_year (int, nullable)
Server: MySQL:3306 » Database: movies » Table: movie_category
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers
Show query box
[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0297 seconds.)

```
CREATE TABLE movie ( movie_id INT AUTO_INCREMENT PRIMARY KEY, movie_title CHAR(250), movie_year INT, movie_location CHAR(50),
movie_category_id INT, FOREIGN KEY (movie_category_id) REFERENCES movie_category(category_id) )
```

- We created a primary key called `movie_id` that will be an integer value that will be increased automatically each time a new record is added

- We added a title field called `movie_title` that can be a maximum of 250 characters that will contain the title of the movie

- We added a year field called `movie_year` that will be an integer number containing the year that the movie was released

- We added a location field called `movie_location` that can be a maximum of 50 characters where the user can specify where the movie is stored.

- The last field `movie_category_id` will be an integer number containing the integer number corresponding to the movie category that we want to classify this movie as. This is a foreign key, with the `category_id` field in the `movie_category` table. We specify this with the foreign key statement