ONLINE MOVIE DATABASE SYSTEM

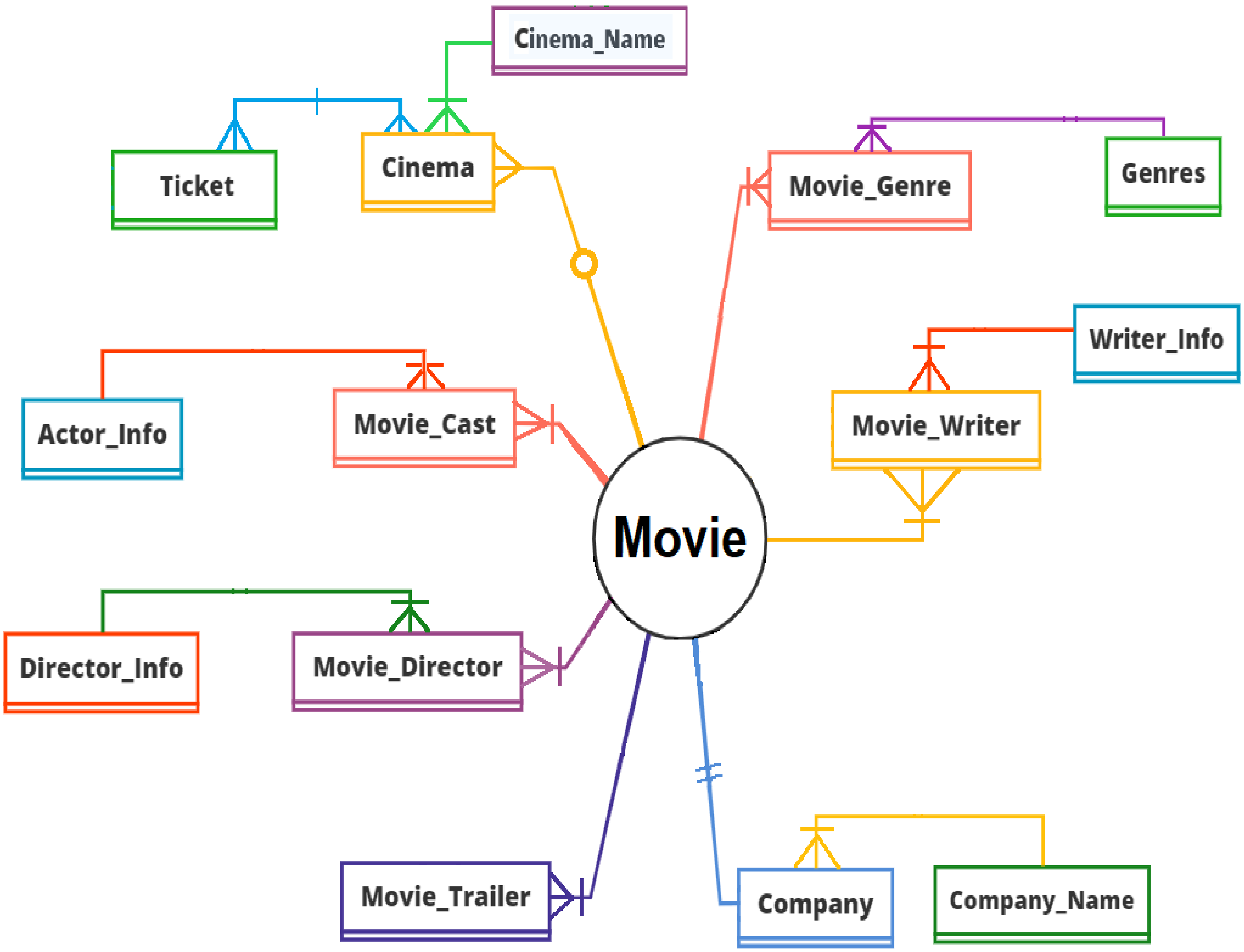
MAHER

COMPUTER SCIENCE

FEATURES OF AN ONLINE MOVIE DATABSE SYSTEM :

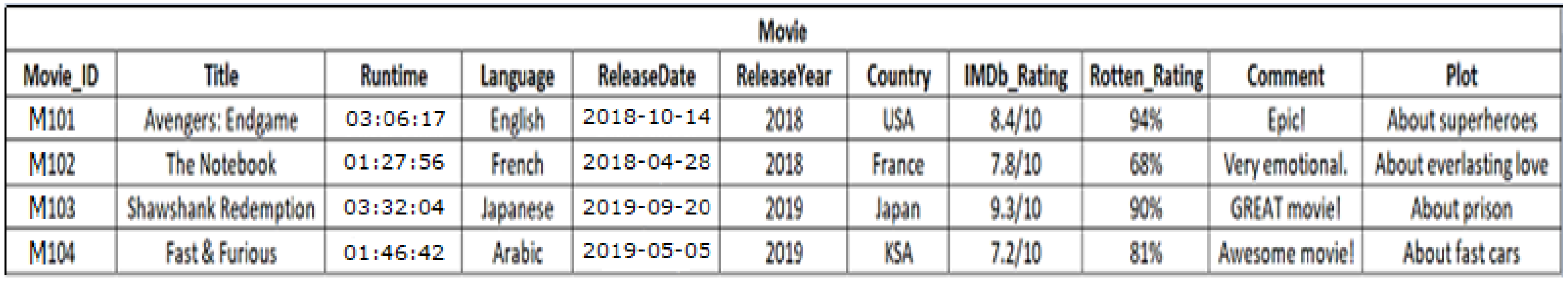
1. Trailers of the movie.
2. Information about the actors in the movie.
3. Information about the directors of the movie.
4. Information about the writers of the movie.
5. Information about the production company of the movie.
6. Showing which cinemas are currently showing the movie.
7. Information about the ticket prices to watch the movie in cinemas.

ER DIAGRAM


Cinema_Name
Ticket
Cinema
Movie_Genre
Genres
Writer_Info
Actor_Info
Movie_Cast
Movie_Writer
Movie
Director_Info
Movie_Director
Movie_Trailer
Company
Company_Name

Sample Tables

Movie table consists of all the unique information of a particular movie.



Movie										
Movie_ID	Title	Runtime	Language	ReleaseDate	ReleaseYear	Country	IMDb_Rating	Rotten_Rating	Comment	Plot
M101	Avengers: Endgame	03:06:17	English	2018-10-14	2018	USA	8.4/10	94%	Epic!	About superheroes
M102	The Notebook	01:27:56	French	2018-04-28	2018	France	7.8/10	68%	Very emotional.	About everlasting love
M103	Shawshank Redemption	03:32:04	Japanese	2019-09-20	2019	Japan	9.3/10	90%	GREAT movie!	About prison
M104	Fast & Furious	01:46:42	Arabic	2019-05-05	2019	KSA	7.2/10	81%	Awesome movie!	About fast cars

Movie_Director	
Movie_ID	Dir_ID
M101	D02
M101	D01
M102	D03
M103	D04
M104	D04
M104	D05

One movie can have many directors. Therefore Movie and Movie_Director have a one to many relationship.

Director_Info		
Dir_ID	Dir_Fname	Dir_Lname
D01	Anthony	Russo
D02	Joe	Russo
D03	Nick	Cassavetes
D04	Frank	Darabont
D05	Justin	Lin

Director_Info table consists of the information about each director, segregated by their unique IDs. One director can direct many movies, so Director_Info and Movie_Director have a one to many relationship.

Movie_Cast		
Movie_ID	Act_ID	Role
M101	A01	Iron Man
M102	A02	Red
M103	A03	Allie Hamilton
M104	A04	Dominic Toretto
M101	A04	Groot

One movie can have many actors. Therefore Movie and Movie_Cast have a one to many relationship.

Actor_Info				
Act_ID	**Act_Fname**	**Act_Lname**	**Act_Gender**	**Act_DOB**
A01	Robert	Downey Jr.	Male	1970-05-26
A02	Morgan	Freeman	Male	1961-08-03
A03	Rachel	McAdams	Female	1988-03-05
A04	Vin	Diesel	Male	1975-04-14

Actor_Info table consists of the information about each actor, segregated by their unique IDs. One actor can act in many movies, so Actor_Info and Movie_Actor have a one to many relationship.

Movie_Trailer			
Trailer_ID	**Trailer_Name**	**Movie_ID**	**Trailer_Date**
T101	Avengers: Endgame Trailer	M101	2018-06-14
T102	The Notebook Trailer	M102	2018-01-28
T103	Shawshank Redemption Trailer	M103	2019-05-20
T104	Fast & Furious Trailer	M104	2019-01-15
T105	Avengers: Endgame Trailer - 2	M101	2018-07-17
T106	Fast & Furious Trailer - 2	M104	2019-02-13

Unique trailers are segregated by their unique trailer IDs. One movie can have many trailers but one trailer cannot be for many movies. Therefore Movie and Movie_Trailer have a one to many relationship.

Cinema				
Movie_ID	**Cinema_ID**	**Show_Date**	**Show_Time**	**Ticket_Type**
M101	IMAX	2018-10-14	19:00:00	Premium
M101	IMAX	2018-10-14	14:00:00	Regular
M102	CMRK	2018-04-28	17:00:00	Premium
M103	AMC1	2019-09-20	10:00:00	Regular
M104	AMC1	2019-05-05	17:00:00	Premium
M101	CMRK	2018-10-14	12:00:00	Regular

One movie can be played in many cinemas. Therefore Movies and Cinema have a one to many relationship.

Cinema_Name	
Cinema_ID	**Cine_Name**
IMAX	IMAX
AMC1	AMC
CMRK	Cinemark

Cinema_Name consists of all the cinema names, segregated by their unique cinema IDs. One cinema can play many movies, so Cinema_Name and Cinema have a one to many relationship.

Ticket	
Ticket_Type	**Ticket_Price**
Premium	$12.99
Regular	$7.99

Ticket price depends on the type of ticket. One movie can have many types of tickets [Premium and Regular] and one type of ticket [Premium/Regular] can be for many movies. So Ticket and Cinema have a many to many relationship.

Company	
Movie_ID	**Company_ID**
M101	C103
M102	C103
M103	C102
M104	C101

One movie can be produced by only one production company. So Movie and Company have a one to one relation.

Company_Name	
Company_ID	**Comp_Name**
C101	Universal Pictures
C102	Castle Rock Entertainment
C103	Walt Disney Studios

Company_Name consists of all the movie production company names, segregated by their unique company IDs. One production company can produce many movies, so Company_Name and Company have a one to many relationship.

Movie_Writer	
Movie_ID	**Writer_ID**
M101	W01
M102	W03
M103	W02
M104	W04
M104	W02

One movie can have many writers. Therefore Movie and Movie_Writer have a one to many relationship.

Writer_Info		
Writer_ID	**Writer_Fname**	**Writer_Lname**
W01	Stan	Lee
W02	Stephen	King
W03	Nicholas	Sparks
W04	Gary	Thompson

Writer_Info table consists of the information about each writer, segregated by their unique IDs. One Writer can write many movies, so Writer_Info and Movie_Writer have a one to many relationship.

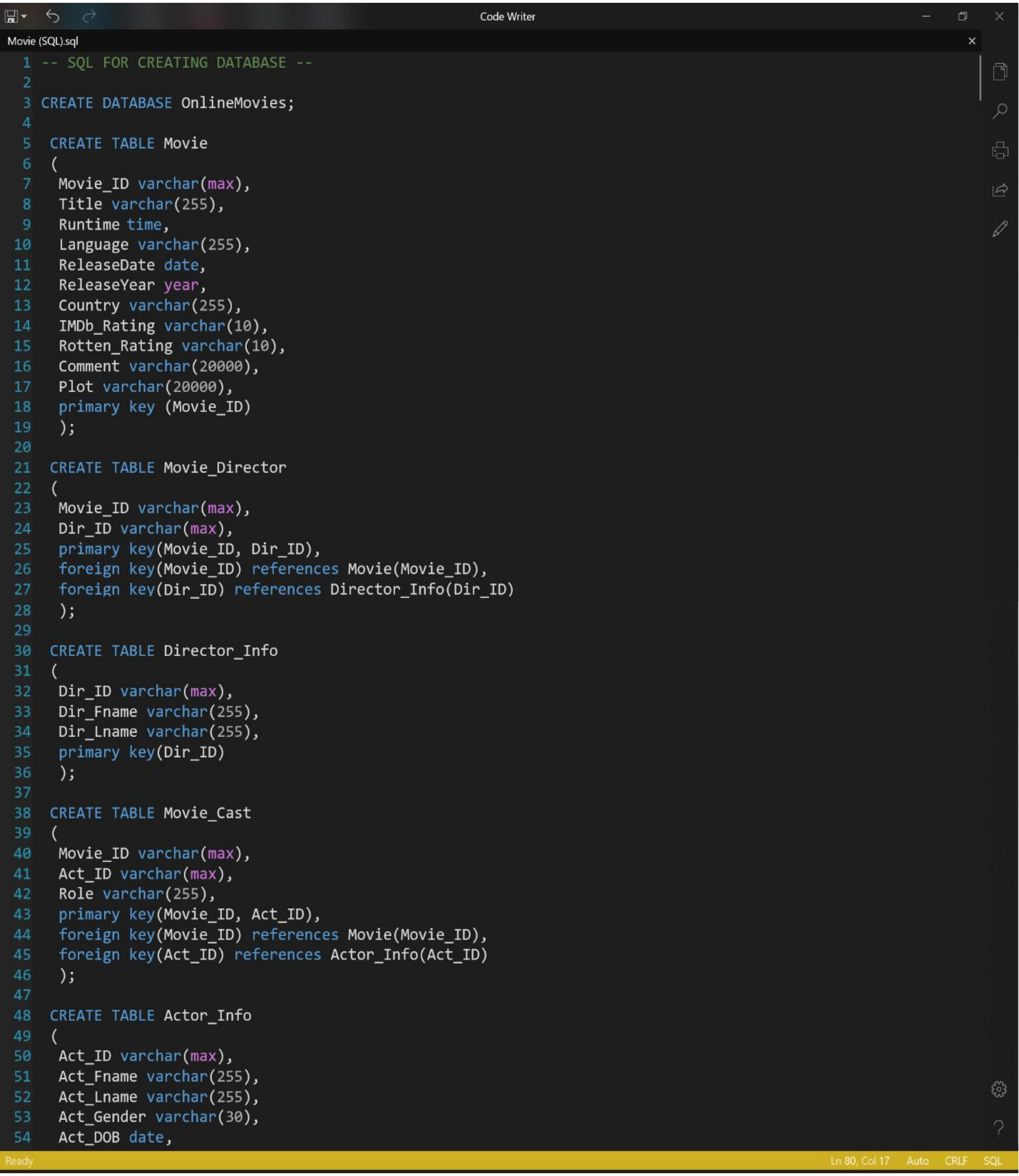
Movie_Genre	
Movie_ID	**Genre_ID**
M101	G01
M101	G02
M102	G03
M102	G04
M103	G04
M104	G01
M104	G05

One movie can have many genres. Therefore, Movie and Movie_Genre has a one to many relationship.

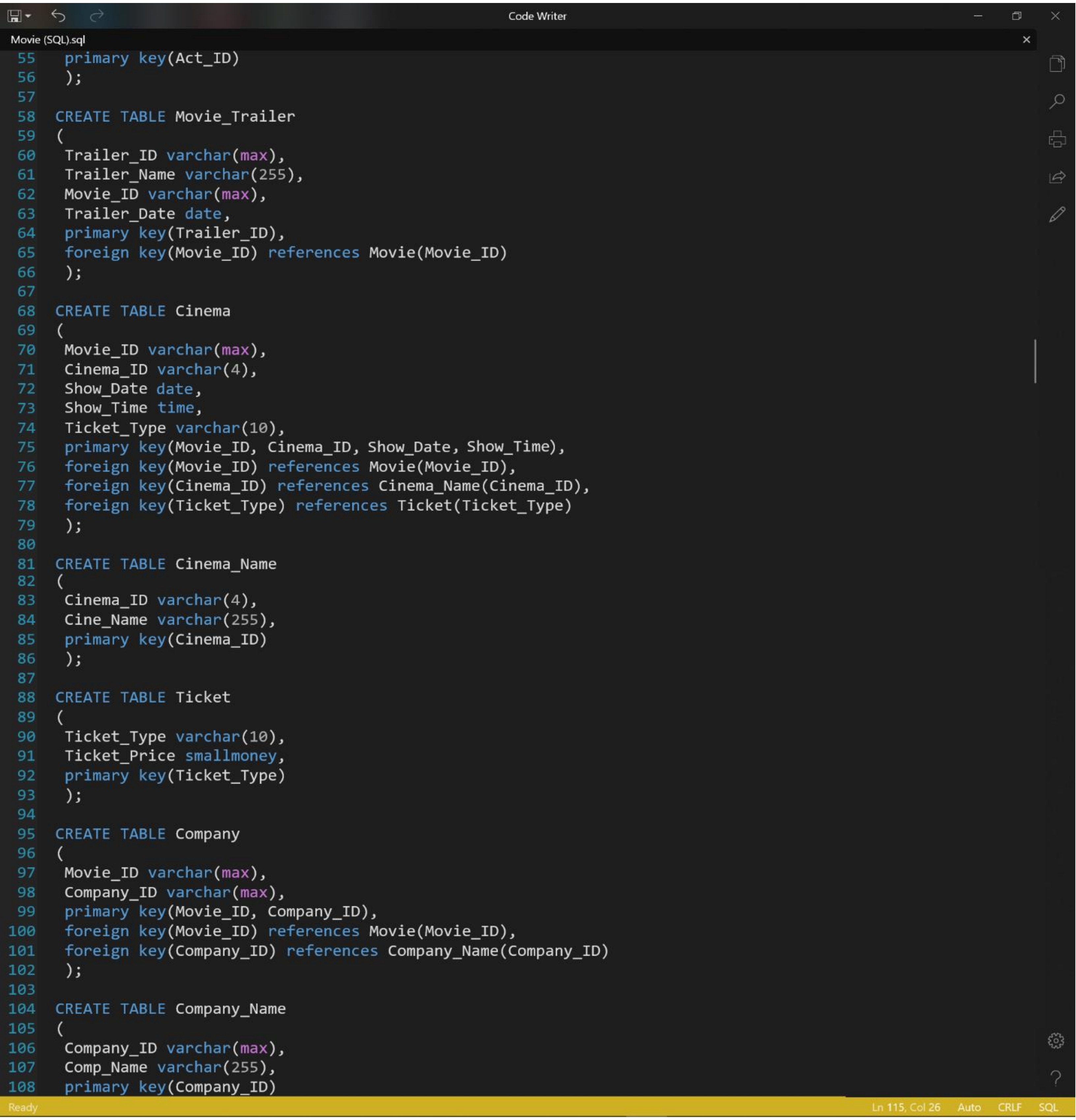
Genres	
Genre_ID	**Genre**
G01	Action
G02	Sci-fi
G03	Romance
G04	Drama
G05	Thriller

Different genres are segregated by their unique Genre IDs. There can be many movies of the same genre, so Genres and Movie_Genre have a one to many relationship.

SQL CODES TO CREATE TABLES AND THEIR REALTIONSHIPS



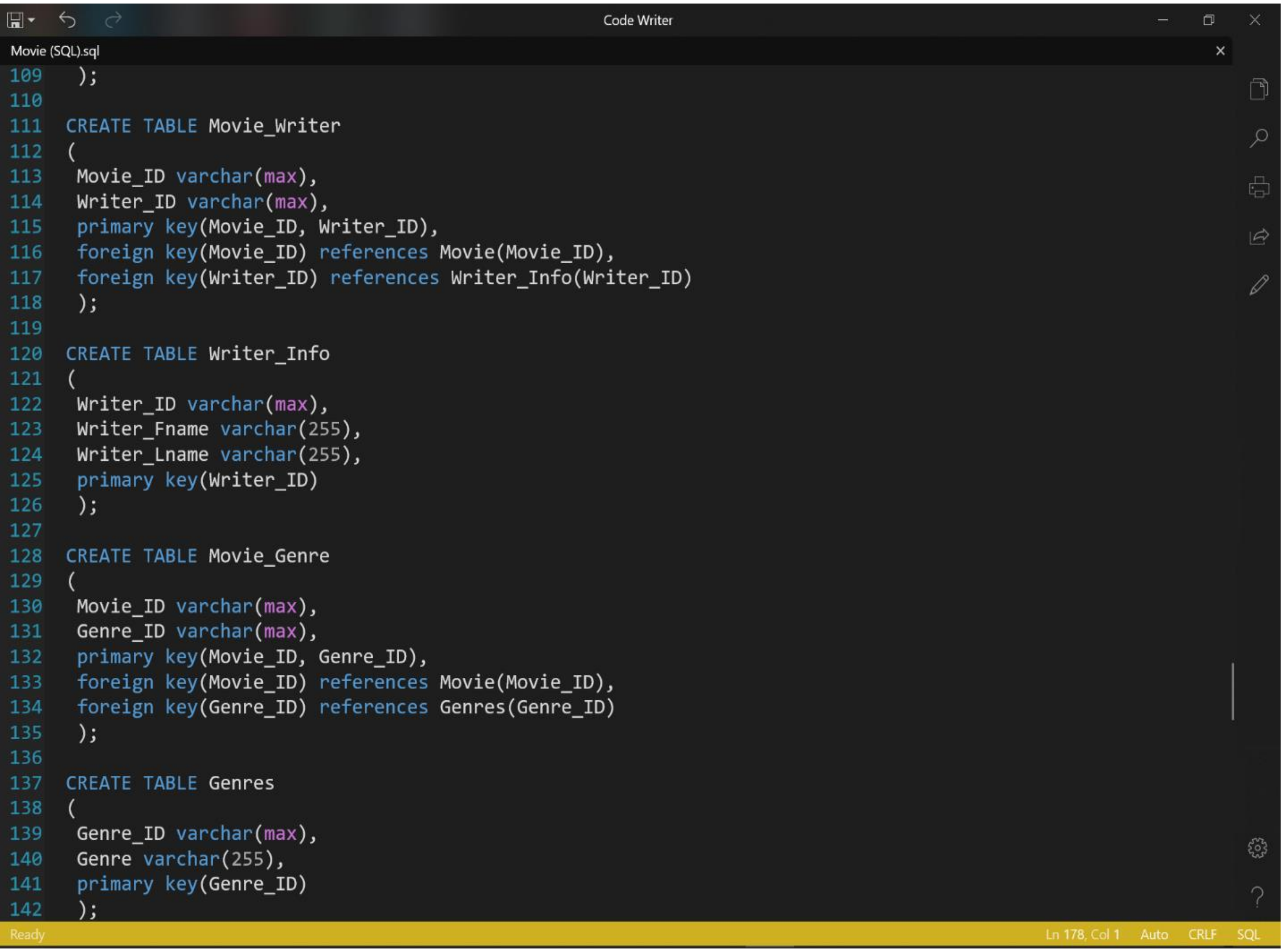
```sql
-- SQL FOR CREATING DATABASE --

CREATE DATABASE OnlineMovies;

 CREATE TABLE Movie
 (
  Movie_ID varchar(max),
  Title varchar(255),
  Runtime time,
  Language varchar(255),
  ReleaseDate date,
  ReleaseYear year,
  Country varchar(255),
  IMDb_Rating varchar(10),
  Rotten_Rating varchar(10),
  Comment varchar(20000),
  Plot varchar(20000),
  primary key (Movie_ID)
  );

 CREATE TABLE Movie_Director
 (
  Movie_ID varchar(max),
  Dir_ID varchar(max),
  primary key(Movie_ID, Dir_ID),
  foreign key(Movie_ID) references Movie(Movie_ID),
  foreign key(Dir_ID) references Director_Info(Dir_ID)
  );

 CREATE TABLE Director_Info
 (
  Dir_ID varchar(max),
  Dir_Fname varchar(255),
  Dir_Lname varchar(255),
  primary key(Dir_ID)
  );

 CREATE TABLE Movie_Cast
 (
  Movie_ID varchar(max),
  Act_ID varchar(max),
  Role varchar(255),
  primary key(Movie_ID, Act_ID),
  foreign key(Movie_ID) references Movie(Movie_ID),
  foreign key(Act_ID) references Actor_Info(Act_ID)
  );

 CREATE TABLE Actor_Info
 (
  Act_ID varchar(max),
  Act_Fname varchar(255),
  Act_Lname varchar(255),
  Act_Gender varchar(30),
  Act_DOB date,
```


Code Writer

Movie (SQL).sql

```sql
 primary key(Act_ID)
 );

CREATE TABLE Movie_Trailer
(
 Trailer_ID varchar(max),
 Trailer_Name varchar(255),
 Movie_ID varchar(max),
 Trailer_Date date,
 primary key(Trailer_ID),
 foreign key(Movie_ID) references Movie(Movie_ID)
 );

CREATE TABLE Cinema
(
 Movie_ID varchar(max),
 Cinema_ID varchar(4),
 Show_Date date,
 Show_Time time,
 Ticket_Type varchar(10),
 primary key(Movie_ID, Cinema_ID, Show_Date, Show_Time),
 foreign key(Movie_ID) references Movie(Movie_ID),
 foreign key(Cinema_ID) references Cinema_Name(Cinema_ID),
 foreign key(Ticket_Type) references Ticket(Ticket_Type)
 );

CREATE TABLE Cinema_Name
(
 Cinema_ID varchar(4),
 Cine_Name varchar(255),
 primary key(Cinema_ID)
 );

CREATE TABLE Ticket
(
 Ticket_Type varchar(10),
 Ticket_Price smallmoney,
 primary key(Ticket_Type)
 );

CREATE TABLE Company
(
 Movie_ID varchar(max),
 Company_ID varchar(max),
 primary key(Movie_ID, Company_ID),
 foreign key(Movie_ID) references Movie(Movie_ID),
 foreign key(Company_ID) references Company_Name(Company_ID)
 );

CREATE TABLE Company_Name
(
 Company_ID varchar(max),
 Comp_Name varchar(255),
 primary key(Company_ID)
```

Ready   Ln 115, Col 26   Auto   CRLF   SQL



```sql
 );

CREATE TABLE Movie_Writer
(
 Movie_ID varchar(max),
 Writer_ID varchar(max),
 primary key(Movie_ID, Writer_ID),
 foreign key(Movie_ID) references Movie(Movie_ID),
 foreign key(Writer_ID) references Writer_Info(Writer_ID)
 );

CREATE TABLE Writer_Info
(
 Writer_ID varchar(max),
 Writer_Fname varchar(255),
 Writer_Lname varchar(255),
 primary key(Writer_ID)
 );

CREATE TABLE Movie_Genre
(
 Movie_ID varchar(max),
 Genre_ID varchar(max),
 primary key(Movie_ID, Genre_ID),
 foreign key(Movie_ID) references Movie(Movie_ID),
 foreign key(Genre_ID) references Genres(Genre_ID)
 );

CREATE TABLE Genres
(
 Genre_ID varchar(max),
 Genre varchar(255),
 primary key(Genre_ID)
 );
```