CS1003 Practical 3 Report

Overview

In this practical I was asked to:

- Model an ER diagram of a given scenario
- Create a relational schema for the modelled scenario
- Create a java program to read in said relational schema and create the database with JDBC
- Create a java program to populate the database using data stored in files
- Create a java program to query the database with hard coded queries

And all these have been achieved fully. I even included some extra queries that take use of user input.

Design

ERD & Database Design

When modelling the database scenario, first the actors and movies tables were modelled, then realising this would be a many to many relationship (as many actors can star in many different movies) the film cast table was modelled to aid with this (including the character the actor played in the movie). Afterwards the award table was modelled which contains actor awards and movie awards and are thus connected to both tables as many actors/movies can win awards. Then since a movie can be directed by multiple directors and be of several different genres the film directors and film genres tables were modelled. The ERD can be seen below in figure 1:

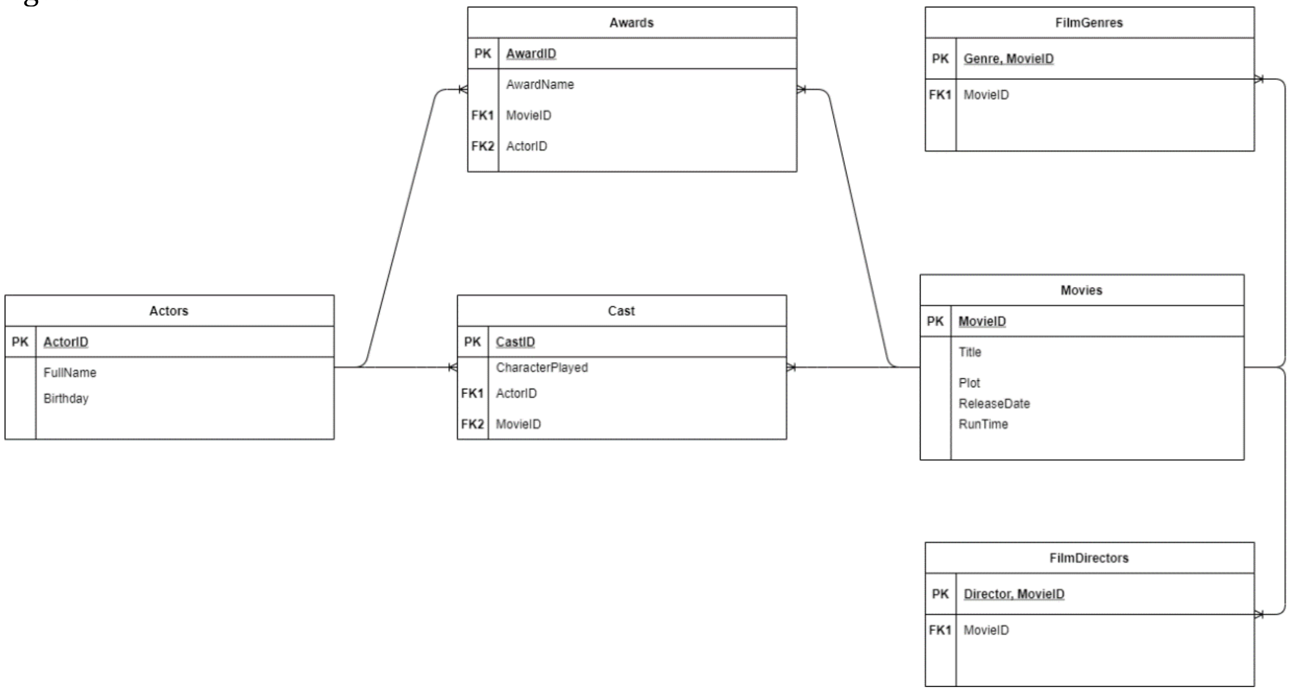


Figure 1: Entity Relationship Diagram

Here we see the attribute in said tables. They include the attribute from the specification along with primary keys as unique identifiers. The genre and directors tables instead use combined keys as opposed to a normal primary keys. Something to note is if the specification required a lot more details on the directors, other than their name and the movies they have directed, e.g.

their birthday, awards *they've* won, etc. Then another director table would be needed as it would be a many to many relationship. So, what has actually been modelled here is one half of a many to many relationships that just includes the directors name that would be in the non-existent second half.

For the java programs a portion of the code is a heavily modified version of the example JDBC code from studres[1]

InitialiseDB Design

First the util, files/file path and SQL packages are imported. Then the main method taking in a string array of the argument in the command line. First an if statement checking if there are an appropriate number of arguments, if not then display an error message demonstrating which arguments should be included and then quits the program.
Next the first 2 values of the arguments are stored in strings for the file name of the database and schema file. Then a new initialise database object is made and calls the create table method passing the two file names.

The create table method takes in 2 strings, those being the database and schema file names. First the connection is reset then the program tries to create a table. It declares a path object using the schema file. Then it checks if a file with the path of that path object exists and is readable. If not, then it throws an exception. So, if it is readable then the exception is not thrown so the program reads all lines of the schema file into a string list. Then creates an iterator for the list.

Next a string attribute is declared for the URL of the database, then using that string the connection to the database is created. Then each line of the schema file a create statement created and executed using the file iterator. Then if no exceptions were thrown and no errors occurred then print the ok message to the terminal to indicate that the tables were created correctly. If an exception occurs, then print out the error message to the terminal. And finally, if the connection was actually established then close it.

PopulateDB Design

This is quite similar to the InitialiseDB, it in string array of arguments, checks there is an appropriate number of arguments and then creates a populate database object and runs the populate tables method. The main difference is the second argument is for the file name for the data for the table being populated. The populate tables again resets the connection but also resets a string of the table being populated. Then again, the file is checked if it exists and is readable and will throw an exception if not. However, there are a set of if/else if statements that check if the file name contains the table name then sets the string that will be used in the insert query to the appropriate table name.

Then read the data file and create an iterator then establish a connection to the database. Also store the first line in the file into a string which is the fieldnames being inserted into the table then loop for each line in the file after the first line. Each loop the current file line is stored in a string then a string array is declared by splitting the current data row by the comma. Then the insert query starts to get constructed with string building by using insert into then adding the table being populate then the fieldnames line in brackets, then the values. Then it loops for each value in the data array. The current data is added to the insert query. After the loop, the last character which will be a comma is trimmed off as the for loop will leave a comma at the end of the string. Then a closing bracket is added, and the insert query is created as a statement object and executed, and the statement is closed.

Next the SQL exception is caught, and an error message is displayed. Finally, again no matter if an exception occurred or not then then if the database connection exists then close the connection.

QueryDB Design

Again, the structure of this program is similar to each of the other programs but instead this takes an extra argument for the query number being executed instead of a file name and the imports are the scanner and SQL packages. It checks for the correct number of arguments, then checks if the query number is between 1 and 10 inclusive, if not then display an error message and display what each query will do then quit the program. Then a new query database object is made and runs the query tables method with the database file name and the query number.

The query database method will again clear the connection try establishing a connection to the database, etc. This time a scanner object is instantiated that takes input from the command line. Then several if/else if statements for each query number from 1 to 10 inclusive. Queries 1 to 7 are hard coded and simply create a SQL statement and get a result set from executing each of them, then using a while loop to display the results of the queries. However, queries 8,9 and 10 use prepared statements as they display a message to the command line asking for a parameter to sue in the query. The statement is preparing and then the user input is inserted into the queries and they are run, and the results are displayed similarly to the other queries.

Then at the end of the if/else if statements the scanner object is closed and if an exception occurred it is caught and displayed to the console. Finally, again like the other programs, if the connection exists, close it.

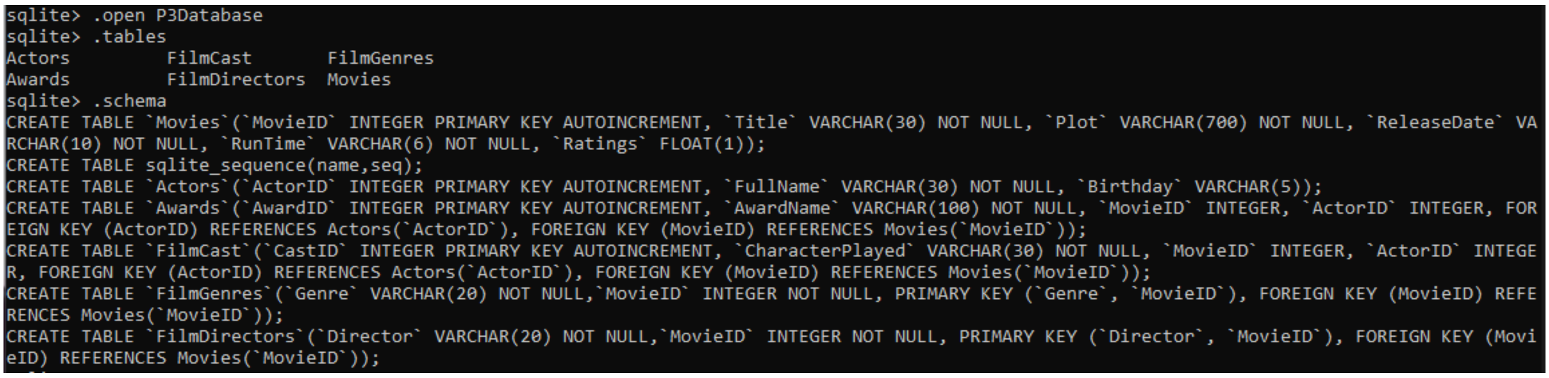
Testing

The following table contains what I tested, how I tested them, and what the results were.

What is being tested?	**Name of method being tested**	**Pre-condition**	**Expected outcome**	**Actual Outcome**	**Evidence**
Initialising the database	createTable();	No database file exists	The database and tables will be created, and the program will display ok	The expected outcome occurred	SQLite gets the schema (figure 2) and the console displays ‘ok’ (figure 3)
Try to initialise the database but the schema file is wrong	createTable();	No database file exists	An error will occur saying the file is incorrect.	The expected outcome occurred	The command line displayed the evidence(figure 4)
Try to initialise the database but the wrong number of	InitialiseDB main method	None	An error will occur, and the program will not run	The expected	The error was sent to the command line (figure 5)

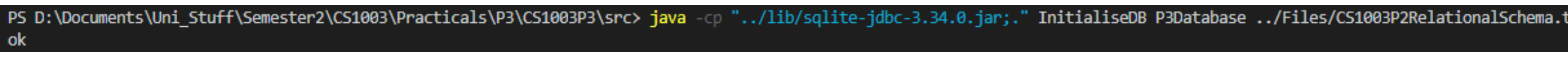
arguments are passed				outcome occurred	
Check each of the tables are populated correctly	populateTables();	The database has been initialised correctly	The data will appear in the tables correctly	The expected outcome occurred	SQLite3 displays the data correctly (figure 6/7)
Try to populate a table with a file that does not exist	populateTables();	The database has been initialised correctly	An exception will occur, and program will quit without inserting values	The expected outcome occurred	The exception was printed to the console (figure 8)
Try to populate a table with an incorrect file	populateTables();	The database has been initialised correctly	An exception will occur, and program will quit without inserting values	The expected outcome occurred	The SQL exception stopped the program and was displayed in the terminal(figure 9)
Try to populate a table with an incorrect database file	populateTables();	None	An exception will occur, and program will quit without inserting values as even though it will not matter at first as the database file if it did not exist would be created by JDBC it would not have the table schema initialised in it and thus cause an error	The expected outcome occurred	The database was created (figure 10) but an error still occurred as it had not tables in it (figure 11)
Try to populate a table but the wrong number of arguments are passed	PopulateDB main method	None	An error will occur, and the program will not run	The expected outcome occurred	The error was sent to the command line (figure 12)
Query the database with one of the hard coded queries	QueryTables();	The tables have been initialised and populated correctly	The results of the query will display in the console	The expected outcome occurred	The query displayed correctly (figure 13) The same query was also run in SQLite3 and displayed the same (figure 14)
Query the database with one of the user input queries	QueryTables();	The tables have been initialised and populated correctly	The results of the query will display in the console	The expected outcome occurred	The query displayed correctly (figure 15) The same query was also run in SQLite3 and displayed the same (figure 16)
Try to query the database with the wrong database file	QueryTables();	None	An exception will occur, and program will quit	The expected outcome occurred	The error was sent to the command line (figure 17)
Try to query the database with the wrong query number	QueryTables();	None	An exception will occur, and program will quit	The expected outcome occurred	The error was sent to the command line (figure 18)
Try to populate a table but the wrong number of arguments are passed	QueryDB main method	None	An exception will occur, and program will quit	The expected outcome occurred	The error was sent to the command line (figure 19)

Testing Evidence



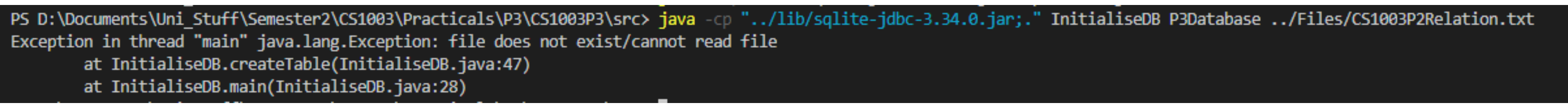
```
sqlite> .open P3Database
sqlite> .tables
Actors         FilmCast       FilmGenres
Awards         FilmDirectors  Movies
sqlite> .schema
CREATE TABLE `Movies`(`MovieID` INTEGER PRIMARY KEY AUTOINCREMENT, `Title` VARCHAR(30) NOT NULL, `Plot` VARCHAR(700) NOT NULL, `ReleaseDate` VA
RCHAR(10) NOT NULL, `RunTime` VARCHAR(6) NOT NULL, `Ratings` FLOAT(1));
CREATE TABLE sqlite_sequence(name,seq);
CREATE TABLE `Actors`(`ActorID` INTEGER PRIMARY KEY AUTOINCREMENT, `FullName` VARCHAR(30) NOT NULL, `Birthday` VARCHAR(5));
CREATE TABLE `Awards`(`AwardID` INTEGER PRIMARY KEY AUTOINCREMENT, `AwardName` VARCHAR(100) NOT NULL, `MovieID` INTEGER, `ActorID` INTEGER, FOR
EIGN KEY (ActorID) REFERENCES Actors(`ActorID`), FOREIGN KEY (MovieID) REFERENCES Movies(`MovieID`));
CREATE TABLE `FilmCast`(`CastID` INTEGER PRIMARY KEY AUTOINCREMENT, `CharacterPlayed` VARCHAR(30) NOT NULL, `MovieID` INTEGER, `ActorID` INTEGE
R, FOREIGN KEY (ActorID) REFERENCES Actors(`ActorID`), FOREIGN KEY (MovieID) REFERENCES Movies(`MovieID`));
CREATE TABLE `FilmGenres`(`Genre` VARCHAR(20) NOT NULL,`MovieID` INTEGER NOT NULL, PRIMARY KEY (`Genre`, `MovieID`), FOREIGN KEY (MovieID) REFE
RENCES Movies(`MovieID`));
CREATE TABLE `FilmDirectors`(`Director` VARCHAR(20) NOT NULL,`MovieID` INTEGER NOT NULL, PRIMARY KEY (`Director`, `MovieID`), FOREIGN KEY (Movi
eID) REFERENCES Movies(`MovieID`));
```

Figure 2: SQLite3 displaying the tables and schema for the tables



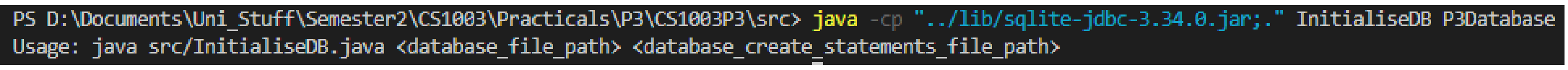
```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." InitialiseDB P3Database ../Files/CS1003P2RelationalSchema.t
ok
```

Figure 3: Ok is displayed to the console indicating the schema has been implemented properly



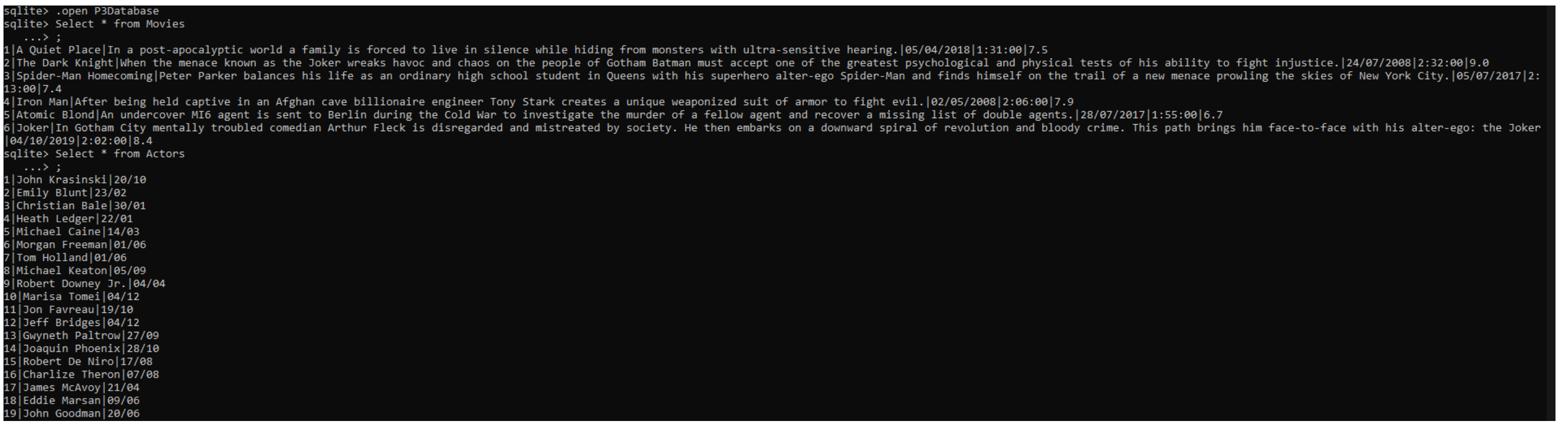
```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." InitialiseDB P3Database ../Files/CS1003P2Relation.txt
Exception in thread "main" java.lang.Exception: file does not exist/cannot read file
        at InitialiseDB.createTable(InitialiseDB.java:47)
        at InitialiseDB.main(InitialiseDB.java:28)
```

Figure 4: The wrong relational schema file is used and caused this error

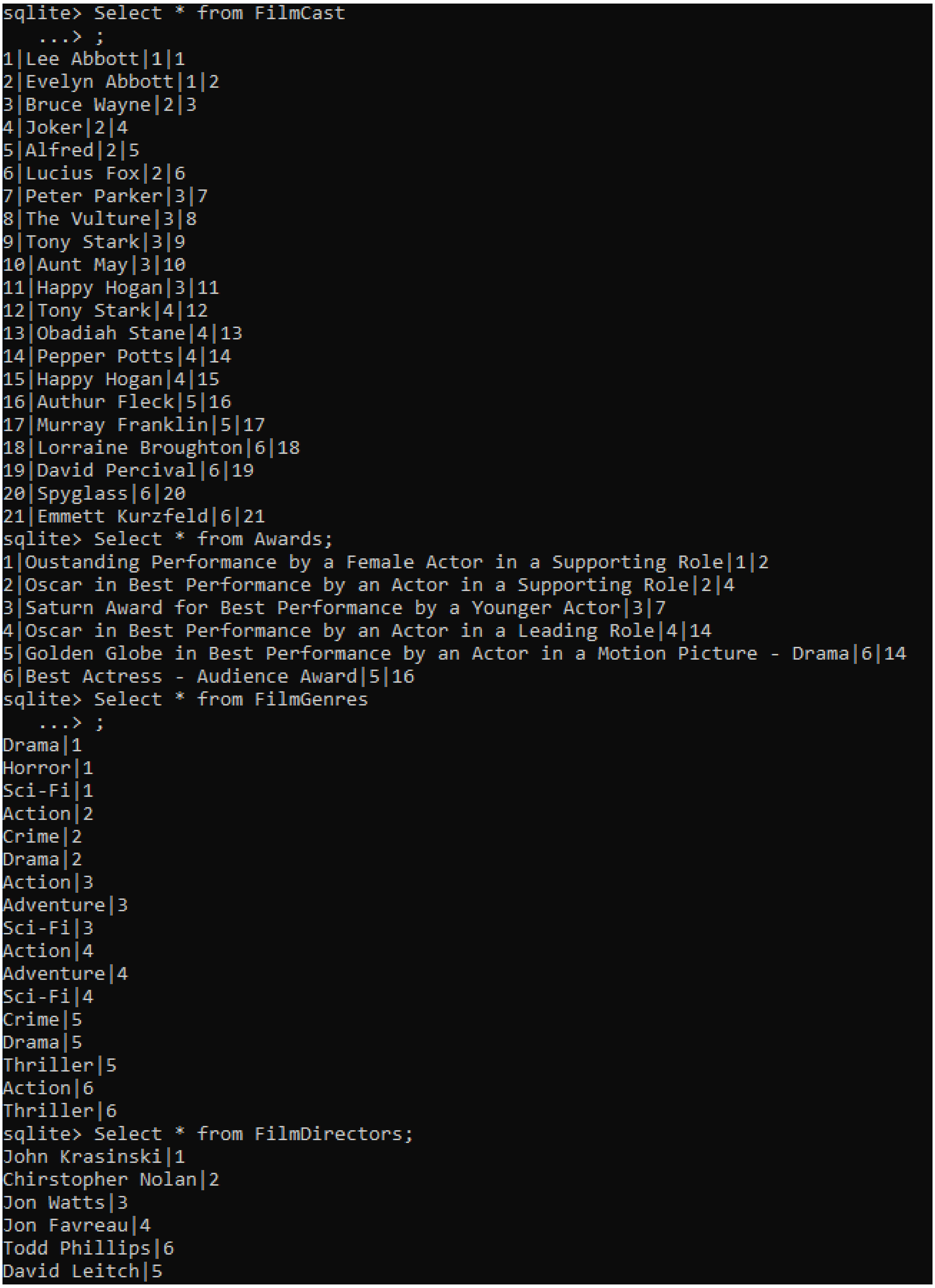


```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." InitialiseDB P3Database
Usage: java src/InitialiseDB.java <database_file_path> <database_create_statements_file_path>
```

Figure 5: The wrong number of arguments were passed and thus an error occurred.

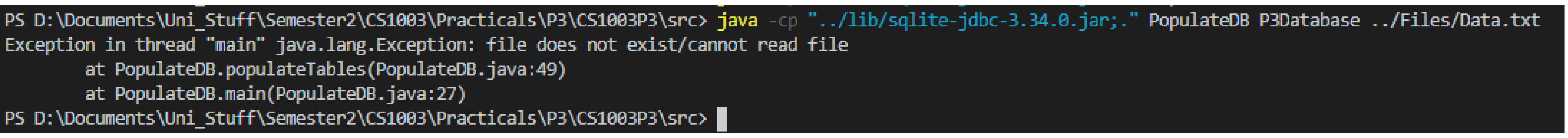


```
sqlite> .open P3Database
sqlite> Select * from Movies
   ...> ;
1|A Quiet Place|In a post-apocalyptic world a family is forced to live in silence while hiding from monsters with ultra-sensitive hearing.|05/04/2018|1:31:00|7.5
2|The Dark Knight|When the menace known as the Joker wreaks havoc and chaos on the people of Gotham Batman must accept one of the greatest psychological and physical tests of his ability to fight injustice.|24/07/2008|2:32:00|9.0
3|Spider-Man Homecoming|Peter Parker balances his life as an ordinary high school student in Queens with his superhero alter-ego Spider-Man and finds himself on the trail of a new menace prowling the skies of New York City.|05/07/2017|2:
13:00|7.4
4|Iron Man|After being held captive in an Afghan cave billionaire engineer Tony Stark creates a unique weaponized suit of armor to fight evil.|02/05/2008|2:06:00|7.9
5|Atomic Blond|An undercover MI6 agent is sent to Berlin during the Cold War to investigate the murder of a fellow agent and recover a missing list of double agents.|28/07/2017|1:55:00|6.7
6|Joker|In Gotham City mentally troubled comedian Arthur Fleck is disregarded and mistreated by society. He then embarks on a downward spiral of revolution and bloody crime. This path brings him face-to-face with his alter-ego: the Joker
|04/10/2019|2:02:00|8.4
sqlite> Select * from Actors
   ...> ;
1|John Krasinski|20/10
2|Emily Blunt|23/02
3|Christian Bale|30/01
4|Heath Ledger|22/01
5|Michael Caine|14/03
6|Morgan Freeman|01/06
7|Tom Holland|01/06
8|Michael Keaton|05/09
9|Robert Downey Jr.|04/04
10|Marisa Tomei|04/12
11|Jon Favreau|19/10
12|Jeff Bridges|04/12
13|Gwyneth Paltrow|27/09
14|Joaquin Phoenix|28/10
15|Robert De Niro|17/08
16|Charlize Theron|07/08
17|James McAvoy|21/04
18|Eddie Marsan|09/06
19|John Goodman|20/06
```



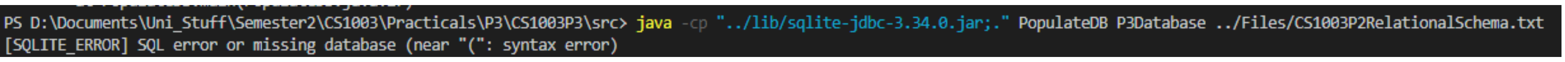
```
sqlite> Select * from FilmCast
   ...> ;
1|Lee Abbott|1|1
2|Evelyn Abbott|1|2
3|Bruce Wayne|2|3
4|Joker|2|4
5|Alfred|2|5
6|Lucius Fox|2|6
7|Peter Parker|3|7
8|The Vulture|3|8
9|Tony Stark|3|9
10|Aunt May|3|10
11|Happy Hogan|3|11
12|Tony Stark|4|12
13|Obadiah Stane|4|13
14|Pepper Potts|4|14
15|Happy Hogan|4|15
16|Authur Fleck|5|16
17|Murray Franklin|5|17
18|Lorraine Broughton|6|18
19|David Percival|6|19
20|Spyglass|6|20
21|Emmett Kurzfeld|6|21
sqlite> Select * from Awards;
1|Oustanding Performance by a Female Actor in a Supporting Role|1|2
2|Oscar in Best Performance by an Actor in a Supporting Role|2|4
3|Saturn Award for Best Performance by a Younger Actor|3|7
4|Oscar in Best Performance by an Actor in a Leading Role|4|14
5|Golden Globe in Best Performance by an Actor in a Motion Picture - Drama|6|14
6|Best Actress - Audience Award|5|16
sqlite> Select * from FilmGenres
   ...> ;
Drama|1
Horror|1
Sci-Fi|1
Action|2
Crime|2
Drama|2
Action|3
Adventure|3
Sci-Fi|3
Action|4
Adventure|4
Sci-Fi|4
Crime|5
Drama|5
Thriller|5
Action|6
Thriller|6
sqlite> Select * from FilmDirectors;
John Krasinski|1
Chirstopher Nolan|2
Jon Watts|3
Jon Favreau|4
Todd Phillips|6
David Leitch|5
```

Figure 6/7: All the data in the tables from SQLite3



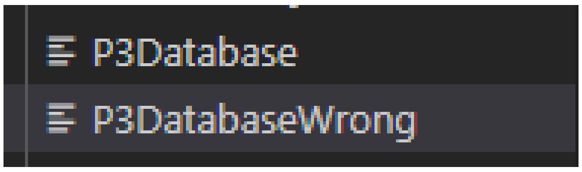
```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." PopulateDB P3Database ../Files/Data.txt
Exception in thread "main" java.lang.Exception: file does not exist/cannot read file
        at PopulateDB.populateTables(PopulateDB.java:49)
        at PopulateDB.main(PopulateDB.java:27)
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src>
```

Figure 8: Error occurred due to non-existent file being entered



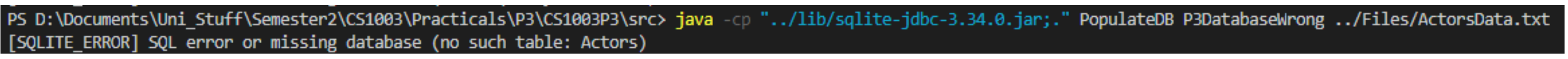
```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." PopulateDB P3Database ../Files/CS1003P2RelationalSchema.txt
[SQLITE_ERROR] SQL error or missing database (near "(": syntax error)
```

Figure 9: Error occurred due to incorrect file being entered



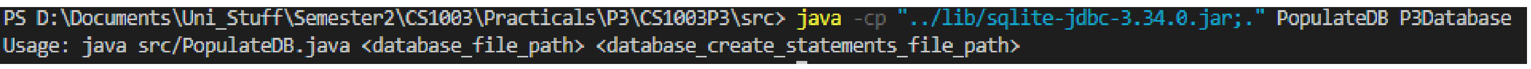
```
P3Database
P3DatabaseWrong
```

Figure 10: The wrong database file was indeed created



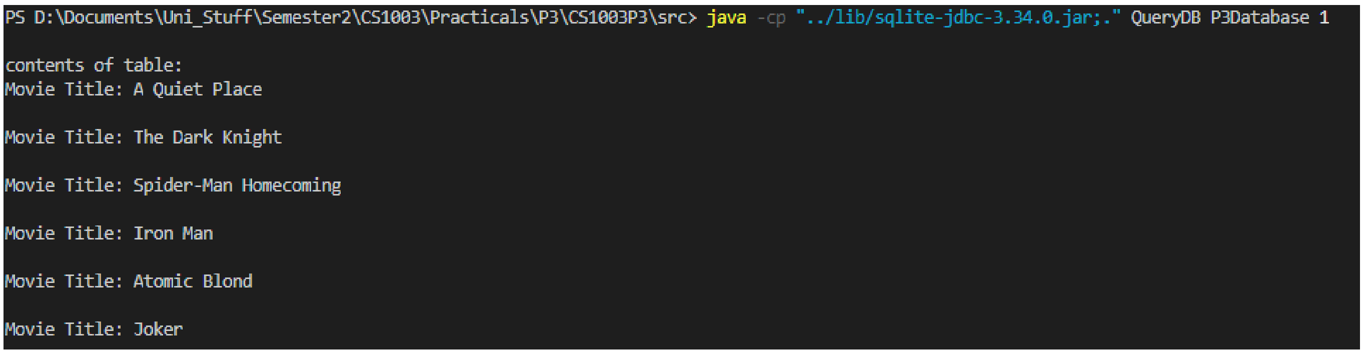
```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." PopulateDB P3DatabaseWrong ../Files/ActorsData.txt
[SQLITE_ERROR] SQL error or missing database (no such table: Actors)
```

Figure 11: Error occurred due to incorrect database name



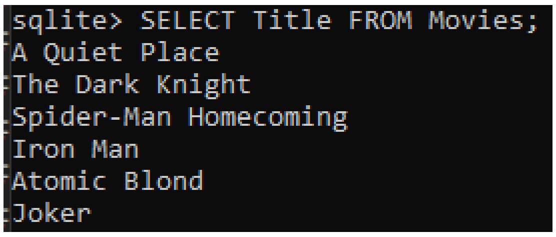
```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." PopulateDB P3Database
Usage: java src/PopulateDB.java <database_file_path> <database_create_statements_file_path>
```

Figure 12: Error occurred due to incorrect number of argument parameters



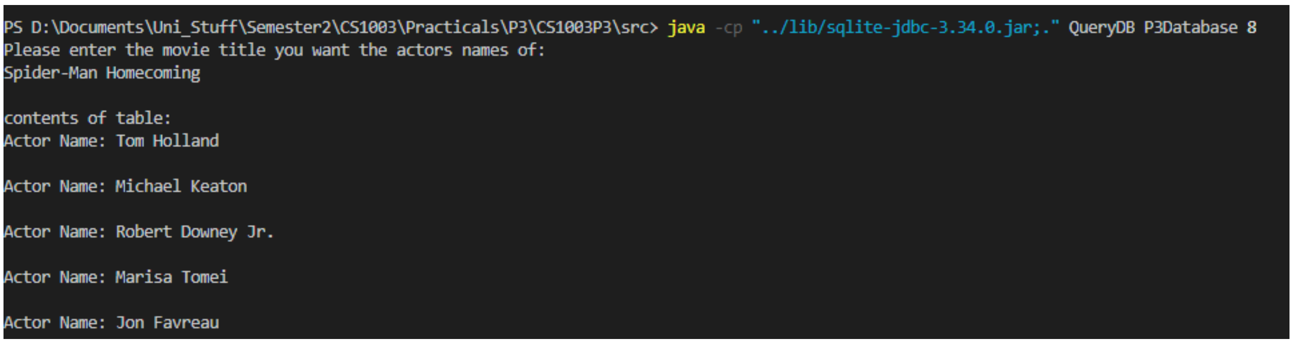
```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 1

contents of table:
Movie Title: A Quiet Place

Movie Title: The Dark Knight

Movie Title: Spider-Man Homecoming

Movie Title: Iron Man

Movie Title: Atomic Blond

Movie Title: Joker
```

Figure 13: The results of query displayed correctly



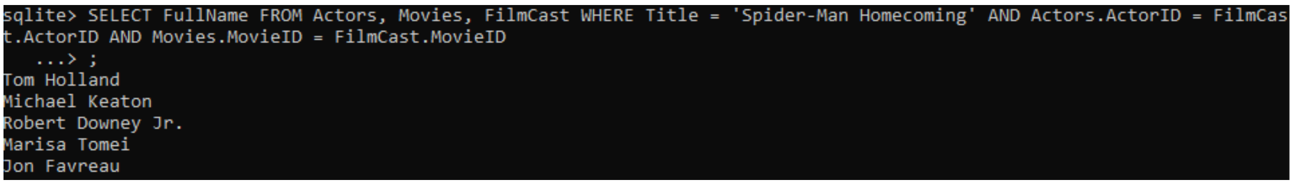
```
sqlite> SELECT Title FROM Movies;
A Quiet Place
The Dark Knight
Spider-Man Homecoming
Iron Man
Atomic Blond
Joker
```

Figure 14: The results of query displayed in SQLite3



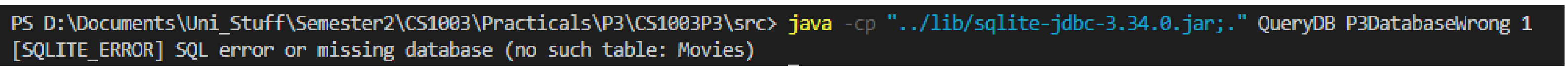
```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 8
Please enter the movie title you want the actors names of:
Spider-Man Homecoming

contents of table:
Actor Name: Tom Holland

Actor Name: Michael Keaton

Actor Name: Robert Downey Jr.

Actor Name: Marisa Tomei

Actor Name: Jon Favreau
```

Figure 15: The results of query displayed correctly



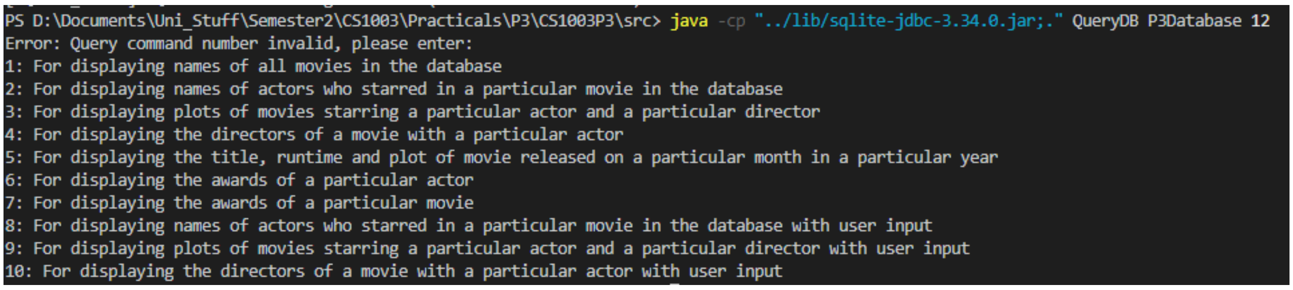
```
sqlite> SELECT FullName FROM Actors, Movies, FilmCast WHERE Title = 'Spider-Man Homecoming' AND Actors.ActorID = FilmCas
t.ActorID AND Movies.MovieID = FilmCast.MovieID
   ...> ;
Tom Holland
Michael Keaton
Robert Downey Jr.
Marisa Tomei
Jon Favreau
```

Figure 16: The results of query displayed in SQLite3



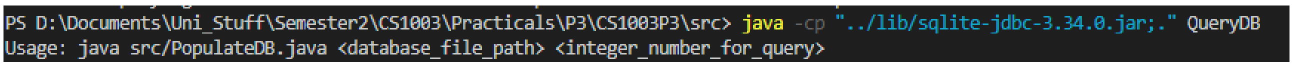
```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3DatabaseWrong 1
[SQLITE_ERROR] SQL error or missing database (no such table: Movies)
```

Figure 17: Error occurred due to entering wrong database file name



```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 12
Error: Query command number invalid, please enter:
1: For displaying names of all movies in the database
2: For displaying names of actors who starred in a particular movie in the database
3: For displaying plots of movies starring a particular actor and a particular director
4: For displaying the directors of a movie with a particular actor
5: For displaying the title, runtime and plot of movie released on a particular month in a particular year
6: For displaying the awards of a particular actor
7: For displaying the awards of a particular movie
8: For displaying names of actors who starred in a particular movie in the database with user input
9: For displaying plots of movies starring a particular actor and a particular director with user input
10: For displaying the directors of a movie with a particular actor with user input
```

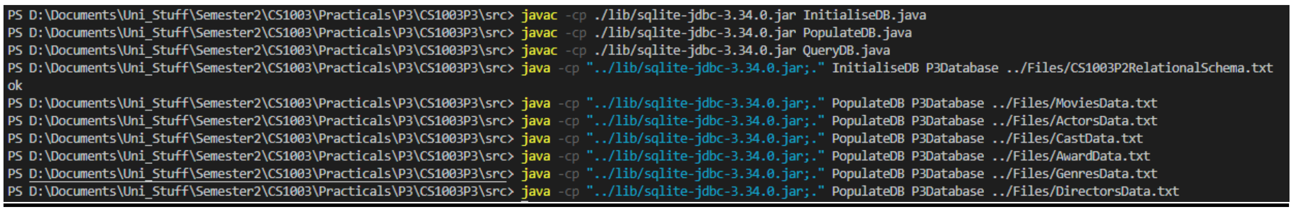
Figure 18: Error occurred due to entering wrong query number



```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB
Usage: java src/PopulateDB.java <database_file_path> <integer_number_for_query>
```

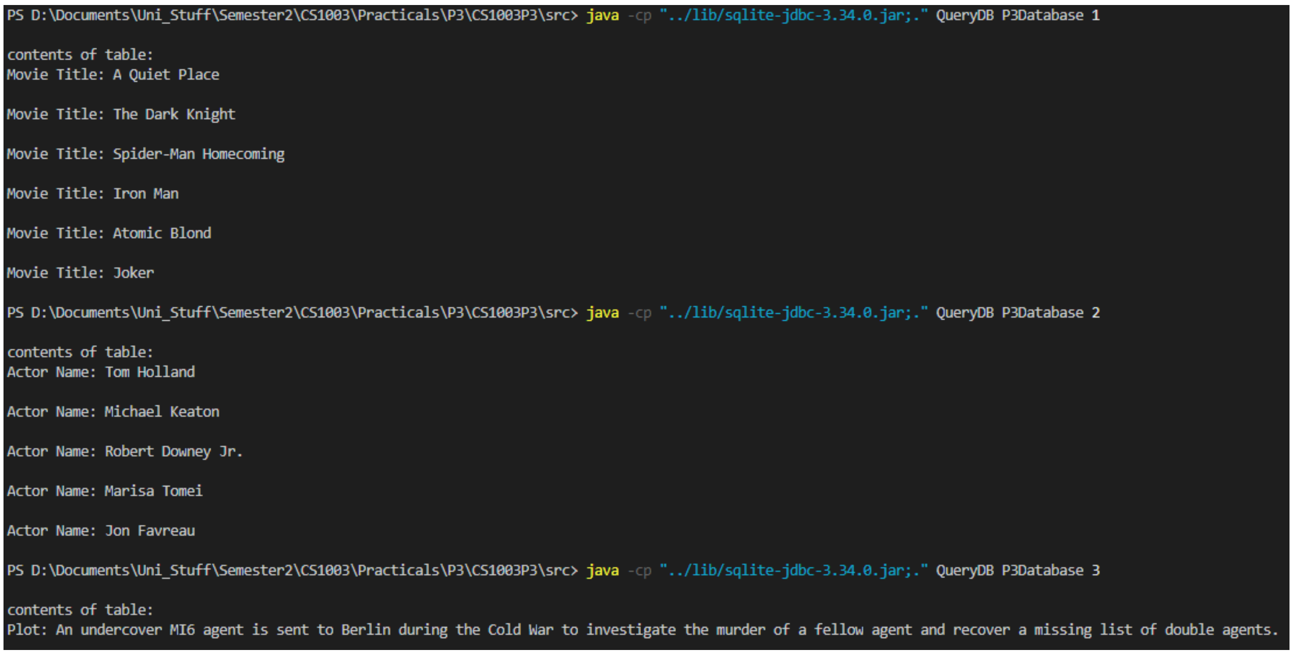
Figure 19: Error occurred due to entering wrong amount of arguments

Example Program Runs



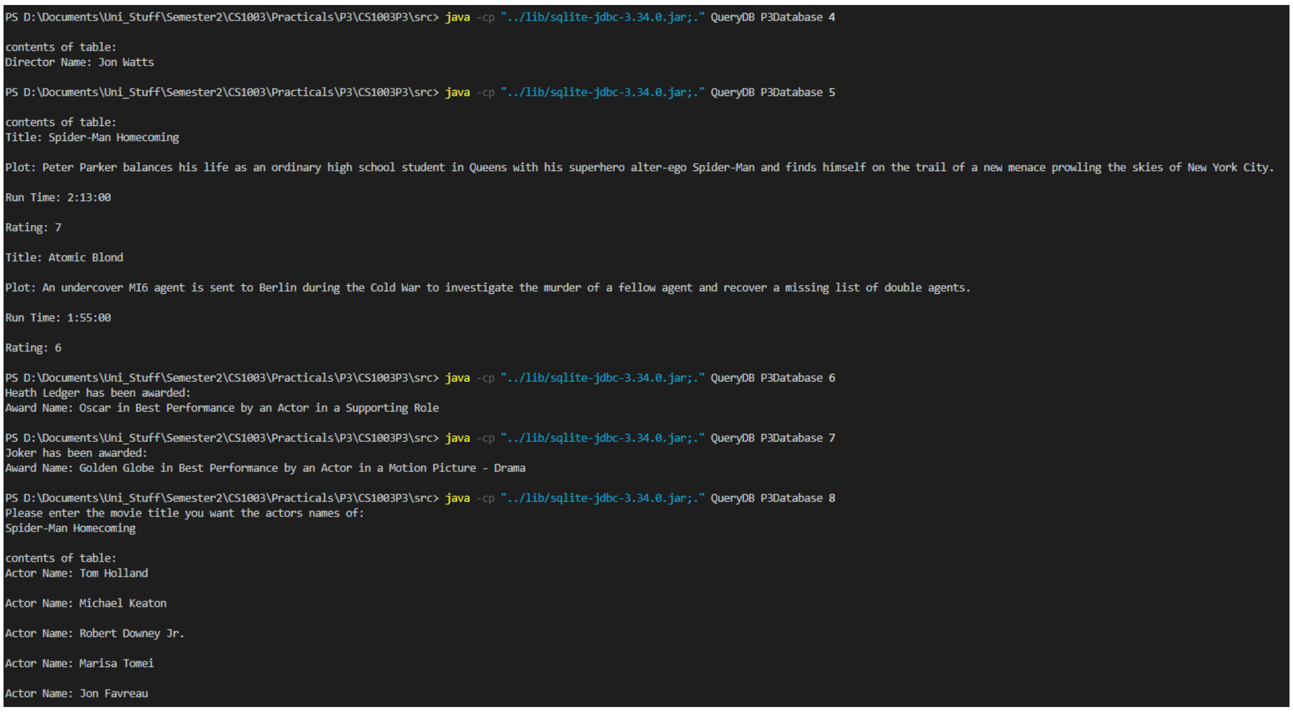
```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> javac -cp ./lib/sqlite-jdbc-3.34.0.jar InitialiseDB.java
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> javac -cp ./lib/sqlite-jdbc-3.34.0.jar PopulateDB.java
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> javac -cp ./lib/sqlite-jdbc-3.34.0.jar QueryDB.java
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." InitialiseDB P3Database ../Files/CS1003P2RelationalSchema.txt
ok
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." PopulateDB P3Database ../Files/MoviesData.txt
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." PopulateDB P3Database ../Files/ActorsData.txt
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." PopulateDB P3Database ../Files/CastData.txt
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." PopulateDB P3Database ../Files/AwardData.txt
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." PopulateDB P3Database ../Files/GenresData.txt
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." PopulateDB P3Database ../Files/DirectorsData.txt
```

Figure 20: The programs InitialiseDB and PopulateDB are run



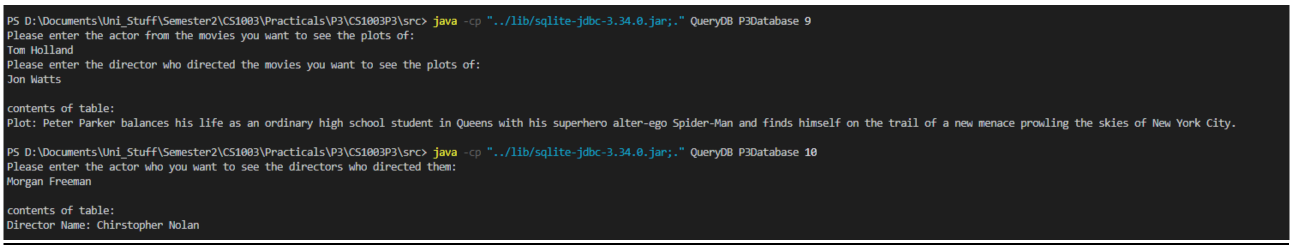
```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 1

contents of table:
Movie Title: A Quiet Place

Movie Title: The Dark Knight

Movie Title: Spider-Man Homecoming

Movie Title: Iron Man

Movie Title: Atomic Blond

Movie Title: Joker

PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 2

contents of table:
Actor Name: Tom Holland

Actor Name: Michael Keaton

Actor Name: Robert Downey Jr.

Actor Name: Marisa Tomei

Actor Name: Jon Favreau

PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 3

contents of table:
Plot: An undercover MI6 agent is sent to Berlin during the Cold War to investigate the murder of a fellow agent and recover a missing list of double agents.
```

Figure 21: The program QueryDB is run for the first 3 queries



```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 4

contents of table:
Director Name: Jon Watts

PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 5

contents of table:
Title: Spider-Man Homecoming

Plot: Peter Parker balances his life as an ordinary high school student in Queens with his superhero alter-ego Spider-Man and finds himself on the trail of a new menace prowling the skies of New York City.

Run Time: 2:13:00

Rating: 7

Title: Atomic Blond

Plot: An undercover MI6 agent is sent to Berlin during the Cold War to investigate the murder of a fellow agent and recover a missing list of double agents.

Run Time: 1:55:00

Rating: 6

PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 6
Heath Ledger has been awarded:
Award Name: Oscar in Best Performance by an Actor in a Supporting Role

PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 7
Joker has been awarded:
Award Name: Golden Globe in Best Performance by an Actor in a Motion Picture - Drama

PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 8
Please enter the movie title you want the actors names of:
Spider-Man Homecoming

contents of table:
Actor Name: Tom Holland

Actor Name: Michael Keaton

Actor Name: Robert Downey Jr.

Actor Name: Marisa Tomei

Actor Name: Jon Favreau
```

Figure 22: The program QueryDB is run for the next 5 queries



```
PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 9
Please enter the actor from the movies you want to see the plots of:
Tom Holland
Please enter the director who directed the movies you want to see the plots of:
Jon Watts

contents of table:
Plot: Peter Parker balances his life as an ordinary high school student in Queens with his superhero alter-ego Spider-Man and finds himself on the trail of a new menace prowling the skies of New York City.

PS D:\Documents\Uni_Stuff\Semester2\CS1003\Practicals\P3\CS1003P3\src> java -cp "../lib/sqlite-jdbc-3.34.0.jar;." QueryDB P3Database 10
Please enter the actor who you want to see the directors who directed them:
Morgan Freeman

contents of table:
Director Name: Chirstopher Nolan
```

Figure 23: The program QueryDB is run for the last 2 queries

Evaluation

I would say my program does exactly what it set out to do and more. It can create databases and populate them using files given. It can query them as well. It performs all queries necessary as well as the same queries but with user input instead. It is fairly robust as it makes sure it catches exceptions and prevents errors from incorrect arguments and user input.

Conclusion

Overall I found this practical very engaging yet enjoyable, especially compared to the last two, the specification was vague which made it a lot harder but for this kind of practical I enjoyed the bit more freedom. I had some problems her and there but I mainly took my time ensuring the program would stay robust so I did not have to go back and change things as that could get very confusing.

References

1. https://studres.cs.st-andrews.ac.uk/CS1003/Lectures/W06-Examples/JDBC/src/JDBCExampleSQLite1.java [Accessed 10/03/2021]