|  |
| --- |
| Advanced Database Development   Cinema Database Development |
| To: Gemma Charters  From: Andrew Laing |
| CITY OF LIVERPOOL COLLEGE 2019 |

Table of Contents

[Terms of Reference 3](#_Toc535588315)

[Designing a database solution for FECT 4](#_Toc535588316)

[Context Diagram for the Booking System 7](#_Toc535588317)

[Database Diagram 9](#_Toc535588318)

[Entity Relationship Diagrams showing Cardinality 11](#_Toc535588319)

[Table Design 13](#_Toc535588320)

[bookingsTBL 14](#_Toc535588321)

[customersTBL 15](#_Toc535588322)

[filmsTBL 16](#_Toc535588323)

[performanceTBL 17](#_Toc535588324)

[screenTBL 18](#_Toc535588325)

[Database Creation Script 19](#_Toc535588326)

[Test Data Creation Script 76](#_Toc535588327)

[Test Queries 85](#_Toc535588328)

[References 104](#_Toc535588329)

[Bibliography 104](#_Toc535588330)

[Appendix A - Unit Tests for Stored Procedures and Functions 105](#_Toc535588331)

# 

# Terms of Reference

As an analyst-programmer working for Greasby Software Solutions, the author of this report was tasked to develop a relational database system for FECT (the Foundation for Electronic and Cinematic Technology.) The pages which follow document the design of this system.

The report commences with a description of the design process. This is followed by database diagrams and the commented SQL scripts used to create and test the solution. All scripts have been tested and are ready for use.  
  
 Andrew Laing 16/01/2019

# Designing a database solution for FECT

Currently, FECT uses a simple spreadsheet to store its customer booking information. This needs to be replaced with a relational database system. This section of the report describes how the solution was designed.

Using the list of recommended tables and fields provided within the specification as a starting point, a rough plan of the database layout was drafted onto paper. This plan showed that it would be necessary to determine who the database’s end users would be. A context diagram was then drawn up. It identified five key user groups;

Customers

Publicity Staff and the Press

The Performance Scheduler

Booking Staff

Administrators

Each group, as consumers and producers of information, required custom procedures written to simplify their interactions with the database. After examining group needs closely, a candidate list of procedures and functions was created.

Customers are FECT’s largest information providers. The database will need to store their personal information plus details of bookings. Customers may also want to confirm, change or cancel bookings over the telephone or ask for performance details.

Publicity Staff and the Press often need information about performances including film details and start times. Providing this information to the press is very important as it will help to keep the organisation in the public eye.

FECT’s film performance scheduler needs to retrieve basic information about films such as length and genre to create well-balanced timetables for the film screenings. This information will be entered by Administrators as soon as suppliers deliver celluloid or digital films to the organisation.

Administrators, the performance scheduler, and the Press could also benefit from having access to booking statistics which give details of the best performing films.

Finally, booking staff will need to access customer records in order to keep them informed of changes to the published schedule, and to make and confirm new bookings.

Now that a rough idea of the database system including tables and procedures had been drafted onto paper, it was time to produce a more polished design. A semi-working prototype was created using Microsoft Access. Although this model did not include procedures it allowed primary and foreign key relationships to be tested, and helped table designs to be normalised to the Third Normal Form.

Creating a prototype of the solution using Access helped to accelerate the design process. Ideas could be tested quickly, and the model created was used to guide development of the solution.

Although datatypes had been chosen for the prototype, the final design still needed to consider how procedures would take advantage of field values and try to optimise the database’s use of storage space. Initial guidance was found within the United Kingdom Cabinet Office’s Data Standards Catalogue (United Kingdom Government, 2006).

This paper, applied throughout the U.K. Government’s data operations, provides standards for choosing the lengths for data fields such as Surname, Address and Email.

Designing other fields needed a re-examination of user requirements and the list of procedures produced earlier. For some fields allocating enough space for their contents was the most significant factor whilst for others the operations allowed upon the datatype was more important.

The INT type was chosen to store all primary keys. It allows values up to 2,147,483,647.[[1]](#footnote-1) This will ensure that there is enough room for records, although for a long-term solution the BIGINT type would be preferable.

For other numeric fields however, smaller datatypes were chosen. SMALLINT was selected for the number of tickets field in the bookings table and the capacity in the screening rooms table. It will accommodate the current largest screen size of 250 and allow for the capacity to be expanded up to 32,767 if necessary. The limit of the next smallest datatype TINYINT is only 255 which allows little room for the current maximum capacities to be expanded.

For the telephone number field, VARCHAR was selected. Modern telephone numbers, especially those of mobile phones, start with zeros. These zeros are lost if stored within a numeric datatype. For example, the number *00741632960* would be stored as *741632960* creating the potential for some interesting errors.

The DATE and TIME datatypes were chosen rather than DATETIME to reduce disc space consumption. Although some of the calculations, for example date comparison, require converting values into the DATETIME type, using the shorter DATE and TIME types will reduce storage needs by over 50%.

The MONEY datatype was used for prices rather than FLOAT. It is more accurate and thus conforms to Generally Accepted Accounting Principles[[2]](#footnote-2). Numbers stored using the FLOAT datatype can result in calculations producing floating-point errors. Where money is involved, this is unacceptable.

Finally, as SQL does not include a Boolean datatype BIT was used instead. BIT allows only values of one or zero so can be used as a replacement for the True and False Boolean values. This field will also only consume 1 byte of space which is considerably less than the 4 bytes required to store an INT or the 5 bytes needed for storing a VARCHAR string value.

After adding the Foreign Key constraints used to establish relationships between tables, three other constraints were added to provide boundary checks for several numeric fields. Although these checks can easily be implemented on the front-end application accessing the database, the values which these fields contain were considered important enough to warrant their own built-in constraints.

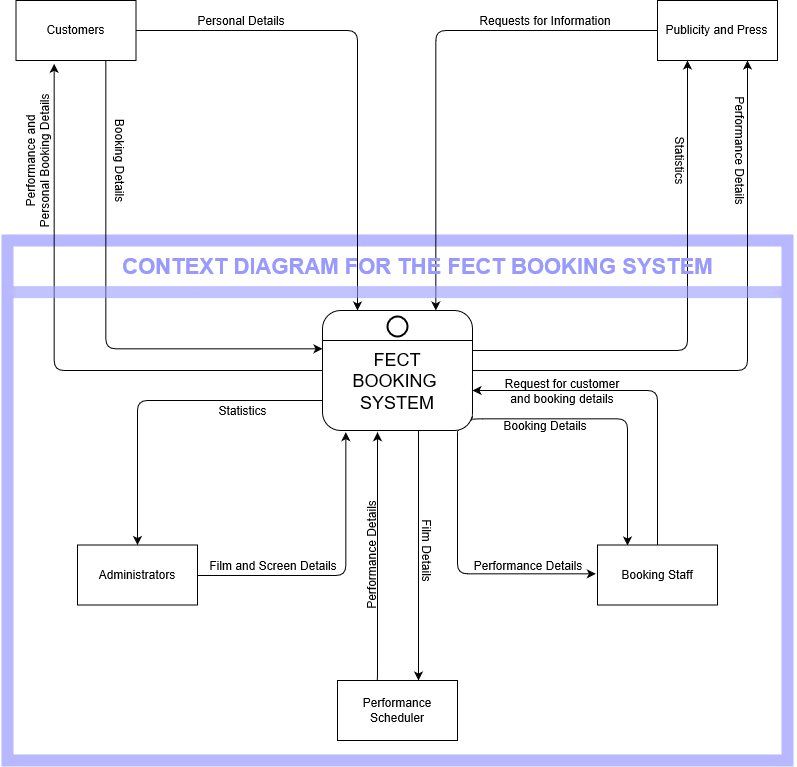
Next, stored procedures and functions were developed. Each new procedure was written along with a unit test to ensure that it worked and behaved as intended.[[3]](#footnote-3) These unit tests were run at regular intervals to confirm that no errors had found their way into the source code.

Once initial coding was completed, the database creation script was refactored[[4]](#footnote-4) and optimised. Procedure and variable names were then normalised, and meaningful comments added for the database’s future maintainers.

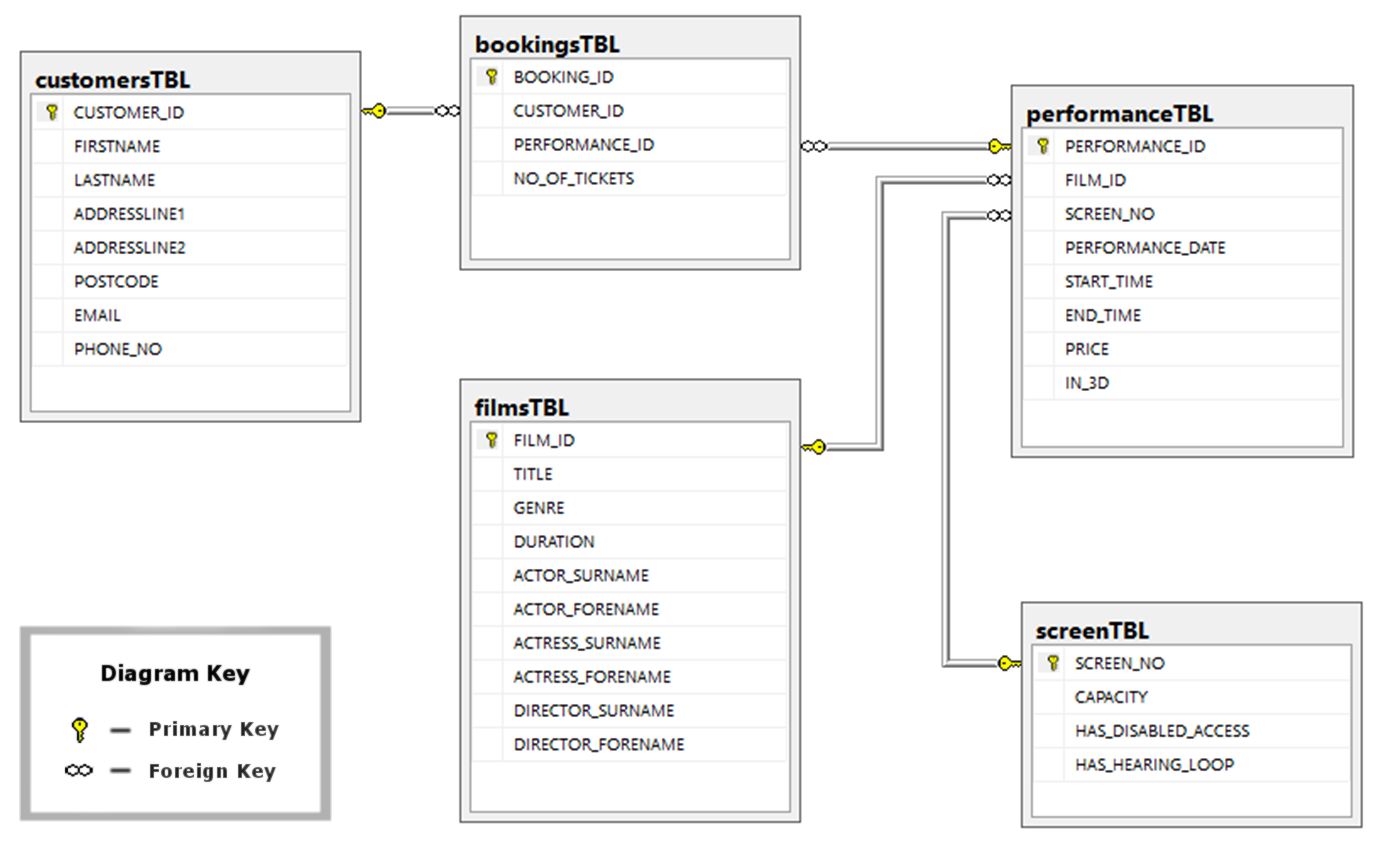
Finally, a new script was written to create the larger dataset and queries requested by the employer. This script inserts a sizeable number of film, customer, and screen records before generating random performance and booking records.

Now that development is complete, the database creation and test scripts are fit for purpose and ready to use. The following pages contain the designs used during development and the SQL source code.

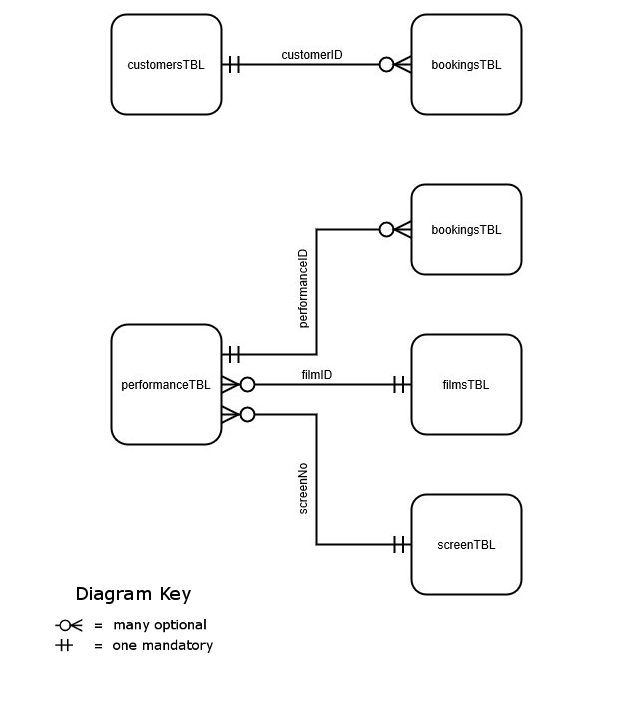
# Context Diagram for the Booking System



# Database Diagram



# Entity Relationship Diagrams showing Cardinality



# Table Design

|  |  |  |  |
| --- | --- | --- | --- |
| bookingsTBL | | | |
| **Description:** The bookingsTBL will hold details about bookings. | | | |
| **Field name** | **Data Type** | **Description** | **Notes** |
| **BOOKING\_ID** | INT | Unique ID number for a booking. | PRIMARY KEY. Starts at one increments by one. |
| **CUSTOMER\_ID** | INT | ID number for a customer. | Required field. FOREIGN KEY CONSTRAINT FK\_CustomerID REFERENCES customersTBL  ( CUSTOMER\_ID ) |
| **PERFORMANCE\_ID** | INT | ID number for a performance. | Required field. FOREIGN KEY CONSTRAINT FK\_PerformanceID REFERENCES performanceTBL  ( PERFORMANCE\_ID ) |
| **NO\_OF\_TICKETS** | SMALLINT | The number of tickets being booked for the customer. | Required field. CONSTRAINT CHK\_Tickets CHECK ( NO\_OF\_TICKETS > 0 ) |

|  |  |  |  |
| --- | --- | --- | --- |
| customersTBL | | | |
| **Description:** customersTBL will hold details about customers. | | | |
| **Field name** | **Data Type** | **Description** | **Notes** |
| **CUSTOMER\_ID** | INT | Unique ID number for a customer record. | PRIMARY KEY. Starts at one increments by one. |
| **FIRSTNAME** | VARCHAR(35) | Customer's first name. | Required field. |
| **LASTNAME** | VARCHAR(35) | Customer's surname. | Required field. |
| **ADDRESSLINE1** | VARCHAR(50) | Customer's address(line 1). |  |
| **ADDRESSLINE2** | VARCHAR(50) | Customer's address(line 2). |  |
| **POSTCODE** | VARCHAR(10) | Customer's postcode. |  |
| **EMAIL** | VARCHAR(50) | Customer's email address. |  |
| **PHONE\_NO** | VARCHAR(15) | Customer's phone number. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| filmsTBL | | | |
| **Description:** filmsTBL will hold details about films. | | | |
| **Field name** | **Data Type** | **Description** | **Notes** |
| **FILM\_ID** | INT | Unique ID number for a film. | PRIMARY KEY. Starts at one increments by one. |
| **TITLE** | VARCHAR(100) | Title of a film. | Required field. |
| **GENRE** | VARCHAR(25) | Genre of a film. |  |
| **DURATION** | TIME | Duration of a film in Hours, Minutes & Seconds. | Required field. |
| **ACTORSURNAME** | VARCHAR(35) | The lead actor's surname. |  |
| **ACTORFORENAME** | VARCHAR(35) | The lead actor's first name. |  |
| **ACTRESSSURNAME** | VARCHAR(35) | The lead actress's surname. |  |
| **ACTRESSFORENAME** | VARCHAR(35) | The lead actress's first name. |  |
| **DIRECTORSURNAME** | VARCHAR(35) | The lead director's surname. |  |
| **DIRECTORFORENAME** | VARCHAR(35) | The lead director's first name. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| performanceTBL | | | |
| **Description:** performanceTBL will hold details about scheduled performances. | | | |
| **Field name** | **Data Type** | **Description** | **Notes** |
| **PERFORMANCE\_ID** | INT | Unique ID number for a film performance | PRIMARY KEY. Starts at one increments by one. |
| **FILM\_ID** | INT | ID number for a film. | Required field. FOREIGN KEY CONSTRAINT FK\_FilmID REFERENCES filmsTBL  ( FILM\_ID ) |
| **SCREEN\_NO** | INT | The number of the screen where the performance will take place. | Required field. |
| **PERFORMANCE\_DATE** | DATE | The date that the performance is taking place on. | Stored in format YYYY-MM-DD |
| **START\_TIME** | TIME | The time that the performance is scheduled to start. | Stored in format hh:mm:ss |
| **END\_TIME** | TIME | The time that the performance is scheduled to finish. | Stored in format hh:mm:ss |
| **PRICE** | MONEY | The cost of the performance in Pounds and Pence. | CONSTRAINT CHK\_Price  CHECK ( PRICE > 0 ) |
| **IN\_3D** | BIT | Whether or not the film is in 3D. | 1 if the film is in 3D, otherwise 0. |

|  |  |  |  |
| --- | --- | --- | --- |
| screenTBL | | | |
| **Description:** screenTBL will hold details about the cinema’s screening rooms. | | | |
| **Field name** | **Data Type** | **Description** | **Notes** |
| **SCREEN\_NO** | INT | Unique ID number for a screening room. | PRIMARY KEY. Starts at one increments by one. |
| **CAPACITY** | SMALLINT | The seating capacity of the screening room. | CONSTRAINT CHK\_Capacity CHECK  ( CAPACITY > 0   AND CAPACITY < 500 ) |
| **HAS\_DISABLED\_ACCESS** | BIT | 1 if the room has access for wheelchairs, otherwise 0. |  |
| **HAS\_HEARING\_LOOP** | BIT | 1 if the room has a hearing loop for the deaf, otherwise 0. |  |

# Database Creation Script

-- ----------------------------------------------------------------------------

-- ----------------------------------------------------------------------------

-- @@@@@@@@@@@@@@@@@@@@@@@

-- @ CREATE THE DATABASE @

-- @@@@@@@@@@@@@@@@@@@@@@@

PRINT 'Create the filmPerformanceDB database.'

**GO**

-- Make sure that the database does not already exist before creating it.

**IF** **DB\_ID(**'filmPerformanceDB'**)** **IS** **NOT** **NULL**

**BEGIN**

PRINT ''

PRINT 'The filmPerformanceDB Database already exists!'

PRINT 'Exiting the installation script!'

**SET** NOEXEC **ON** -- Stop the rest of the script from executing if the database already exists.

**END**

**ELSE**

**BEGIN**

**CREATE** DATABASE filmPerformanceDB**;**

**END**

**GO**

-- Work with the newly created database. **use** filmPerformanceDB**;**

-- ----------------------------------------------------------------------------

-- ----------------------------------------------------------------------------

-- @@@@@@@@@@@@@@@@@@@@@

-- @ CREATE THE TABLES @

-- @@@@@@@@@@@@@@@@@@@@@

PRINT 'Create the tables.'

-- Create the filmsTBL TABLE.

-- Make sure that the table does not already exist before creating it.

**IF** **OBJECT\_ID(**'filmsTBL' **,**'U'**)** **IS** **NOT** **NULL**

**DROP** **TABLE** filmsTBL**;**

**GO**

**CREATE** **TABLE** filmsTBL **(**

FILM\_ID **int** **IDENTITY(**1**,**1**)** **PRIMARY** **KEY,** -- This is the primary key for the table. The value is

-- an automatically assigned number.   
 -- Starts at 1 and increments by 1.

TITLE **VARCHAR(**100**)** **NOT** **NULL,** -- NOT NULL indicates that this is a required field.

GENRE **VARCHAR(**25**),**

DURATION **TIME** **NOT** **NULL,** -- Stored in the time format hh:mm:ss

ACTOR\_SURNAME **VARCHAR(**35**),**

ACTOR\_FORENAME **VARCHAR(**35**),**

ACTRESS\_SURNAME **VARCHAR(**35**),**

ACTRESS\_FORENAME **VARCHAR(**35**),**

DIRECTOR\_SURNAME **VARCHAR(**35**),**

DIRECTOR\_FORENAME **VARCHAR(**35**)**

**);**

**GO**

-- Create the customersTBL TABLE

**IF** **OBJECT\_ID(**'customersTBL' **,**'U'**)** **IS** **NOT** **NULL**

**DROP** **TABLE** customersTBL**;**

**GO**

**CREATE** **TABLE** customersTBL **(**

CUSTOMER\_ID **int** **IDENTITY(**1**,**1**)** **PRIMARY** **KEY,**

FIRSTNAME **VARCHAR(**35**)** **NOT** **NULL,**

LASTNAME **VARCHAR(**35**)** **NOT** **NULL,**

ADDRESSLINE1 **VARCHAR(**50**),**

ADDRESSLINE2 **VARCHAR(**50**),**

POSTCODE **VARCHAR(**10**),**

EMAIL **VARCHAR(**50**),**

PHONE\_NO **VARCHAR(**15**)**

**);**

**GO**

-- Create the screenTBL TABLE

**IF** **OBJECT\_ID(**'screenTBL' **,**'U'**)** **IS** **NOT** **NULL**

**DROP** **TABLE** screenTBL**;**

**GO**

**CREATE** **TABLE** screenTBL **(**

SCREEN\_NO **int** **IDENTITY(**1**,**1**)** **PRIMARY** **KEY,**

CAPACITY **SMALLINT,**

HAS\_DISABLED\_ACCESS **BIT,** -- Use BIT for Yes/No as MSSQL has no Boolean datatype.

HAS\_HEARING\_LOOP **BIT**

**);**

**GO**

-- Create the performanceTBL TABLE

**IF** **OBJECT\_ID(**'performanceTBL' **,**'U'**)** **IS** **NOT** **NULL**

**DROP** **TABLE** performanceTBL**;**

**GO**

**CREATE** **TABLE** performanceTBL **(**

PERFORMANCE\_ID **int** **IDENTITY(**1**,**1**)** **PRIMARY** **KEY,**

FILM\_ID **INT** **NOT** **NULL,**

SCREEN\_NO **INT** **NOT** **NULL,**

PERFORMANCE\_DATE **DATE,** -- Stored in format YYYY-MM-DD

START\_TIME **TIME,**

END\_TIME **TIME,**

PRICE MONEY**,**

IN\_3D **BIT**

**);**

**GO**

-- Create the bookingsTBL TABLE

**IF** **OBJECT\_ID(**'bookingsTBL' **,**'U'**)** **IS** **NOT** **NULL**

**DROP** **TABLE** bookingsTBL**;**

**GO**

**CREATE** **TABLE** bookingsTBL **(**

BOOKING\_ID **INT** **IDENTITY(**1**,**1**)** **PRIMARY** **KEY,**

CUSTOMER\_ID **INT** **NOT** **NULL,**

PERFORMANCE\_ID **INT** **NOT** **NULL,**

NO\_OF\_TICKETS **SMALLINT** **NOT** **NULL**

**);**

**GO**

-- ----------------------------------------------------------------------------

-- ----------------------------------------------------------------------------

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- @ ADD THE FOREIGN KEY CONSTRAINTS @

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

PRINT 'Create the Foreign Key Constraints.'

**GO**

**ALTER** **TABLE** bookingsTBL

**ADD** **CONSTRAINT** FK\_CustomerID **FOREIGN** **KEY(**CUSTOMER\_ID**)**

**REFERENCES** customersTBL**(**CUSTOMER\_ID**)**

**ON** **DELETE** **CASCADE;** -- If a record in the parent table(customersTBL) is deleted the

-- corresponding records in bookingsTBL will be automatically deleted.

**GO**

-- Ensure the Constraint is enabled.

**ALTER** **TABLE** bookingsTBL **CHECK** **CONSTRAINT** FK\_CustomerID

**GO**

**ALTER** **TABLE** bookingsTBL

**ADD** **CONSTRAINT** FK\_PerformanceID **FOREIGN** **KEY(**PERFORMANCE\_ID**)**

**REFERENCES** performanceTBL**(**PERFORMANCE\_ID**)**

**ON** **DELETE** **CASCADE;**

**GO**

**ALTER** **TABLE** bookingsTBL **CHECK** **CONSTRAINT** FK\_PerformanceID**;**

**GO**

**ALTER** **TABLE** performanceTBL

**ADD** **CONSTRAINT** FK\_FilmID **FOREIGN** **KEY(**FILM\_ID**)**

**REFERENCES** filmsTBl**(**FILM\_ID**)**

**ON** **DELETE** **CASCADE;**

**GO**

**ALTER** **TABLE** performanceTBL **CHECK** **CONSTRAINT** FK\_FilmID**;**

**GO**

**ALTER** **TABLE** performanceTBL

**ADD** **CONSTRAINT** FK\_ScreenID **FOREIGN** **KEY(**SCREEN\_NO**)**

**REFERENCES** screenTBL**(**SCREEN\_NO**)**

**ON** **DELETE** **CASCADE;**

**GO**

**ALTER** **TABLE** performanceTBL **CHECK** **CONSTRAINT** FK\_ScreenID**;**

-- ----------------------------------------------------------------------------

-- ----------------------------------------------------------------------------

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- @ ADD THE OTHER CONSTRAINTS @

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- These constraints are used to verify that only a valid positive number can

-- be added to various fields.

**GO**

**ALTER** **TABLE** screenTBL

**ADD** **CONSTRAINT** CHK\_Capacity **CHECK** **(**CAPACITY **>** 0 **AND** CAPACITY **<** 500**);**

**GO**

**GO**

**ALTER** **TABLE** performanceTBL

**ADD** **CONSTRAINT** CHK\_Price **CHECK** **(**PRICE **>** 0**);**

**GO**

**GO**

**ALTER** **TABLE** bookingsTBL

**ADD** **CONSTRAINT** CHK\_Tickets **CHECK** **(**NO\_OF\_TICKETS **>** 0**);**

**GO**

-- ----------------------------------------------------------------------------

-- ----------------------------------------------------------------------------

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- @ CREATE THE INSERT PROCEDURES @

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

PRINT 'Create the INSERT procedures.'

-- ----------------------------------------------------------------------------

-- Procedure Name: insertFilm

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure inserts a new film record.

-- Parameters: @TITLE - The film's title.

-- @GENRE - The genre of the film.

-- @DURATION - The duration of the film in TIME format.

-- @ACTOR\_SURNAME - The surname of the lead actor.

-- @ACTOR\_FORENAME - The first name of the lead actor.

-- @ACTRESS\_SURNAME - The surname of the lead actress.

-- @ACTRESS\_FORENAME - The first name of the lead actress.

-- @DIRECTOR\_SURNAME - The surname of the director.

-- @DIRECTOR\_FORENAME - The first name of the director

--

-- Example usage: EXECUTE insertFilm 'La Dolce Vita', 'Drama', '03:12:22',

-- 'Mastroianni','Marcello','Bellucci','Monica','Fellini','Federico';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'insertFilm' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** insertFilm**;**

**GO**

**CREATE** **PROCEDURE** insertFilm @TITLE **VARCHAR(**100**),**

@GENRE **VARCHAR(**25**),**

@DURATION **TIME,**

@ACTOR\_SURNAME **VARCHAR(**35**),**

@ACTOR\_FORENAME **VARCHAR(**35**),**

@ACTRESS\_SURNAME **VARCHAR(**35**),**

@ACTRESS\_FORENAME **VARCHAR(**35**),**

@DIRECTOR\_SURNAME **VARCHAR(**35**),**

@DIRECTOR\_FORENAME **VARCHAR(**35**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;** -- Turn off the messages sent back by SQL Server to improve performance.

**INSERT** **INTO** filmsTBL

**(** TITLE**,**

GENRE**,**

DURATION**,**

ACTOR\_SURNAME**,**

ACTOR\_FORENAME**,**

ACTRESS\_SURNAME**,**

ACTRESS\_FORENAME**,**

DIRECTOR\_SURNAME**,**

DIRECTOR\_FORENAME **)**

**VALUES** **(** @TITLE**,**

@GENRE**,**

@DURATION**,**

@ACTOR\_SURNAME**,**

@ACTOR\_FORENAME**,**

@ACTRESS\_SURNAME**,**

@ACTRESS\_FORENAME**,**

@DIRECTOR\_SURNAME**,**

@DIRECTOR\_FORENAME **);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: insertCustomer

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure inserts a new customer record.

-- Parameters: @FIRSTNAME - The first name of the customer.

-- @LASTNAME - The surname of the customer

-- @ADDRESSLINE1 - The customer's address (Line 1)

-- @ADDRESSLINE2 - The customer's address (Line 2)

-- @POSTCODE - The customer's postcode.

-- @EMAIL - The email address of the customer.

-- @PHONE\_NO - The customer's phone number

--

-- Example usage: EXECUTE insertCustomer 'Randolph', 'Scott', '22 Pelican Grove',   
-- 'Liverpool', 'L19 0NE',

-- 'randyscott@gmail.com', '0151 666 6666';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'insertCustomer' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** insertCustomer**;**

**GO**

**CREATE** **PROCEDURE** insertCustomer @FIRSTNAME **VARCHAR(**35**),**

@LASTNAME **VARCHAR(**35**),**

@ADDRESSLINE1 **VARCHAR(**50**),**

@ADDRESSLINE2 **VARCHAR(**50**),**

@POSTCODE **VARCHAR(**10**),**

@EMAIL **VARCHAR(**50**),**

@PHONE\_NO **VARCHAR(**15**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**INSERT** **INTO** customersTBL

**(** FIRSTNAME**,**

LASTNAME**,**

ADDRESSLINE1**,**

ADDRESSLINE2**,**

POSTCODE**,**

EMAIL**,**

PHONE\_NO **)**

**VALUES** **(** @FIRSTNAME**,**

@LASTNAME**,**

@ADDRESSLINE1**,**

@ADDRESSLINE2**,**

@POSTCODE**,**

@EMAIL**,**

@PHONE\_NO **);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getEndTime

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the calculated end time of a film performance.

-- Parameters: @START\_TIME - The start of a film performance.

-- @DURATION - The duration of the performance.

--

-- Example usage: SELECT dbo.getEndTime('20:30','01:45') AS END\_TIME;

-- Example result:

-- END\_TIME

-- 22:15:00.0000000

--

-- Notes: The getEndTime function is used by the insertPerformance   
-- and updatePerformance procedures.

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getEndTime' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getEndTime**;**

**GO**

**CREATE** **FUNCTION** getEndTime **(** @START\_TIME **TIME,** @DURATION **TIME)**

**RETURNS** **TIME**

**AS**

**BEGIN**

**RETURN**

**(**

-- To add times together they must first be converted to DATETIME.

**CONVERT(** **time,** **(CONVERT(datetime,**@start\_time**)** **+** **CONVERT(datetime,**@duration**))** **)**

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: insertScreen

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure inserts a new screen record.

-- Parameters: @CAPACITY - The capacity of the screening room.

-- @HAS\_DISABLED\_ACCESS - 1 if the screening room has disabled access, otherwise 0.

-- @HAS\_HEARING\_LOOP - 1 if the screening room has a hearing loop, otherwise 0.

--

-- Example usage: EXECUTE insertScreen 50,0,1;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'insertScreen' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** insertScreen**;**

**GO**

**CREATE** **PROCEDURE** insertScreen @CAPACITY **SMALLINT,**

@HAS\_DISABLED\_ACCESS **BIT,**

@HAS\_HEARING\_LOOP **BIT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**INSERT** **INTO** screenTBL

**(** CAPACITY**,**

HAS\_DISABLED\_ACCESS**,**

HAS\_HEARING\_LOOP **)**

**VALUES** **(** @CAPACITY**,**

@HAS\_DISABLED\_ACCESS**,**

@HAS\_HEARING\_LOOP **);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: insertPerformance

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure inserts a new performance record.

-- Parameters: @FILM\_ID - The ID number of the film being screened.

-- @SCREEN\_NO - The number of the screen where the film will be projected.

-- @PERFORMANCE\_DATE - The date when the film will be screened.

-- @START\_TIME - The start time of the performance.

-- @PRICE - The cost of a single ticket for the performance.

-- @IN\_3D - 1 if the film is in 3D, otherwise 0.

--

-- Example usage: EXECUTE insertPerformance 1,1,'2018-08-21','20:00:00',12.99,0;

-- Notes: This procedure can only insert films from the filmsTBL as a performance.

-- The END\_TIME is calculated using the getEndTime function.

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'insertPerformance' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** insertPerformance**;**

**GO**

**CREATE** **PROCEDURE** insertPerformance @FILM\_ID **INT,**

@SCREEN\_NO **INT,**

@PERFORMANCE\_DATE **DATE,**

@START\_TIME **TIME,**

@PRICE MONEY**,**

@IN\_3D **BIT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

-- Calculate the END\_TIME.

**DECLARE** @duration **TIME** -- Declare variables used in the procedure.

**DECLARE** @END\_TIME **TIME**

**SET** @duration **=** **(SELECT** duration **from** filmsTBL **where** film\_ID **=** @FILM\_ID**)**

**SET** @END\_TIME **=** dbo**.**getEndTime**(**@START\_TIME**,**@duration**);**

-- INSERT the record.

**INSERT** **INTO** performanceTBL

**(** FILM\_ID**,**

SCREEN\_NO**,**

PERFORMANCE\_DATE**,**

START\_TIME**,**

END\_TIME**,**

PRICE**,**

IN\_3D **)**

**VALUES** **(** @FILM\_ID**,**

@SCREEN\_NO**,**

@PERFORMANCE\_DATE**,**

@START\_TIME**,**

@END\_TIME**,**

@PRICE**,**

@IN\_3D **);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: insertBooking

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure inserts a new booking record.

-- Parameters: @CUSTOMER\_ID - The ID number for the customer making the booking.

-- @PERFORMANCE\_ID - The ID number for the performance.

-- @NO\_OF\_TICKETS - The number of tickets being booked.

--

-- Example usage: EXECUTE insertBooking 1,2,2;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'insertBooking' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** insertBooking**;**

**GO**

**CREATE** **PROCEDURE** insertBooking @CUSTOMER\_ID **INT,**

@PERFORMANCE\_ID **INT,**

@NO\_OF\_TICKETS **SMALLINT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**INSERT** **INTO** bookingsTBL

**(** CUSTOMER\_ID**,**

PERFORMANCE\_ID**,**

NO\_OF\_TICKETS **)**

**VALUES** **(** @CUSTOMER\_ID**,**

@PERFORMANCE\_ID**,**

@NO\_OF\_TICKETS **);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- ----------------------------------------------------------------------------

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- @ CREATE THE DELETE PROCEDURES @

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

PRINT 'Create the DELETE procedures.'

-- ----------------------------------------------------------------------------

-- Procedure Name: cascadeDeleteFilm

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure cascade deletes a specified film record.

-- Parameters: @FILM\_ID - The ID number for the record being deleted.

--

-- Example usage: EXECUTE deleteFilm 1;

--

-- Notes: Only an Administrator should be given permission to use

-- this procedure. All other users should utilise the deleteFilm

-- procedure instead.

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'cascadeDeleteFilm' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** cascadeDeleteFilm**;**

**GO**

**CREATE** **PROCEDURE** cascadeDeleteFilm @FILM\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DELETE** **FROM** filmsTBL

**WHERE** FILM\_ID**=**@FILM\_ID**;**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: deleteFilm

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure deletes a specified film record where

-- no records within the performanceTBL table refer it.

-- Parameters: @FILM\_ID - The ID number for the record being deleted.

--

-- Example usage: EXECUTE deleteFilm 1;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'deleteFilm' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** deleteFilm**;**

**GO**

**CREATE** **PROCEDURE** deleteFilm @FILM\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DELETE** **FROM** filmsTBL

-- Only delete the record if it is not referred to by a Foreign Key.

**WHERE** FILM\_ID**=**@FILM\_ID

**AND** **NOT** **EXISTS** **(** **SELECT** **\*** **FROM** performanceTBL **WHERE** FILM\_ID**=**@FILM\_ID **);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: cascadeDeleteCustomer

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure cascade deletes the record for a specified customer.

-- Parameters: @CUSTOMER\_ID - The ID number for the record being deleted.

--

-- Example usage: EXECUTE deleteCustomer 1;

--

-- Notes: Only an Administrator should be given permission to use

-- this procedure. All other users should utilise the deleteCustomer

-- procedure instead.

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'cascadeDeleteCustomer' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** cascadeDeleteCustomer**;**

**GO**

**CREATE** **PROCEDURE** cascadeDeleteCustomer @CUSTOMER\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DELETE** **FROM** customersTBL **WHERE** CUSTOMER\_ID**=**@CUSTOMER\_ID**;**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: deleteCustomer

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure deletes the record for a specified customer

-- where no record references it within the bookingsTBL.

-- Parameters: @CUSTOMER\_ID - The ID number for the record being deleted.

--

-- Example usage: EXECUTE deleteCustomer 1;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'deleteCustomer' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** deleteCustomer**;**

**GO**

**CREATE** **PROCEDURE** deleteCustomer @CUSTOMER\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DELETE** **FROM** customersTBL **WHERE** CUSTOMER\_ID**=**@CUSTOMER\_ID

**AND** **NOT** **EXISTS** **(** **SELECT** **\*** **FROM** bookingsTBL **WHERE** CUSTOMER\_ID**=**@CUSTOMER\_ID **);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: cascadeDeleteScreen

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure cascade deletes a specified screen record.

-- Parameters: @SCREEN\_NO - The ID number for the record being deleted.

--

-- Example usage: EXECUTE deleteScreen 9;

--

-- Notes: Only an Administrator should be given permission to use

-- this procedure. All other users should utilise the deleteScreen

-- procedure instead.

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'cascadeDeleteScreen' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** cascadeDeleteScreen**;**

**GO**

**CREATE** **PROCEDURE** cascadeDeleteScreen @SCREEN\_NO **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DELETE** **FROM** screenTBL

**WHERE** SCREEN\_NO**=**@SCREEN\_NO**;**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: deleteScreen

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure deletes a specified screen record where

-- no records within the performanceTBL table refer it.

-- Parameters: @SCREEN\_NO - The ID number for the record being deleted.

--

-- Example usage: EXECUTE deleteScreen 9;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'deleteScreen' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** deleteScreen**;**

**GO**

**CREATE** **PROCEDURE** deleteScreen @SCREEN\_NO **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DELETE** **FROM** screenTBL

**WHERE** SCREEN\_NO**=**@SCREEN\_NO

**AND** **NOT** **EXISTS** **(** **SELECT** **\*** **FROM** performanceTBL **WHERE** SCREEN\_NO**=**@SCREEN\_NO **);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: deleteBooking

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure deletes the record for a specified booking.

-- Parameters: @BOOKING\_ID - The ID number for the record being deleted.

--

-- Example usage: EXECUTE deleteBooking 1;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'deleteBooking' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** deleteBooking**;**

**GO**

**CREATE** **PROCEDURE** deleteBooking @BOOKING\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DELETE** **FROM** bookingsTBL **WHERE** BOOKING\_ID**=**@BOOKING\_ID**;**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: cascadeDeletePerformance

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure cascade deletes the record for a specified performance

-- Parameters: @PERFORMANCE - The ID number for the record being deleted.

--

-- Example usage: EXECUTE deletePerformance 1;

--

-- Notes: Only an Administrator should be given permission to use

-- this procedure. All other users should utilise the deletePerformance

-- procedure instead.

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'cascadeDeletePerformance' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** cascadeDeletePerformance**;**

**GO**

**CREATE** **PROCEDURE** cascadeDeletePerformance @PERFORMANCE\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DELETE** **FROM** performanceTBL

**WHERE** PERFORMANCE\_ID**=**@PERFORMANCE\_ID**;**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: deletePerformance

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure deletes the record for a specified performance

-- where no record references it within bookingsTBL.

-- Parameters: @PERFORMANCE - The ID number for the record being deleted.

--

-- Example usage: EXECUTE deletePerformance 1;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'deletePerformance' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** deletePerformance**;**

**GO**

**CREATE** **PROCEDURE** deletePerformance @PERFORMANCE\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DELETE** **FROM** performanceTBL

**WHERE** PERFORMANCE\_ID**=**@PERFORMANCE\_ID

**AND** **NOT** **EXISTS** **(** **SELECT** **\*** **FROM** bookingsTBL **WHERE** PERFORMANCE\_ID**=**@PERFORMANCE\_ID **);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- ----------------------------------------------------------------------------

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- @ CREATE THE UPDATE PROCEDURES @

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

PRINT 'Create the UPDATE procedures.'

-- ----------------------------------------------------------------------------

-- Procedure Name: updateFilm

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure updates the record for a specified film.

-- Parameters: @FILM - The ID number for the record being updated.

-- @TITLE - The film's title.

-- @GENRE - The genre of the film.

-- @DURATION - The duration of the film in the TIME format.

-- @ACTOR\_SURNAME - The surname of the lead actor.

-- @ACTOR\_FORENAME - The first name of the lead actor.

-- @ACTRESS\_SURNAME - The surname of the lead actress.

-- @ACTRESS\_FORENAME - The first name of the lead actress.

-- @DIRECTOR\_SURNAME - The surname of the director.

-- @DIRECTOR\_FORENAME - The first name of the director

--

-- Example usage: EXECUTE updateFilm 2, 'La Dolce Vita', 'Drama', '03:12:22',

-- 'Mastroianni','Marcello','Vitti','Monica','Fellini','Federico';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'updateFilm' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** updateFilm**;**

**GO**

**CREATE** **PROCEDURE** updateFilm @FILM\_ID **INT,**

@TITLE **VARCHAR(**100**),**

@GENRE **VARCHAR(**25**),**

@DURATION **TIME,**

@ACTOR\_SURNAME **VARCHAR(**35**),**

@ACTOR\_FORENAME **VARCHAR(**35**),**

@ACTRESS\_SURNAME **VARCHAR(**35**),**

@ACTRESS\_FORENAME **VARCHAR(**35**),**

@DIRECTOR\_SURNAME **VARCHAR(**35**),**

@DIRECTOR\_FORENAME **VARCHAR(**35**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**UPDATE** filmsTBL

**SET** TITLE**=**@TITLE**,**

GENRE**=**@GENRE**,**

DURATION**=**@DURATION**,**

ACTOR\_SURNAME**=**@ACTOR\_SURNAME**,**

ACTOR\_FORENAME**=**@ACTOR\_FORENAME**,**

ACTRESS\_SURNAME**=**@ACTRESS\_SURNAME**,**

ACTRESS\_FORENAME**=**@ACTRESS\_FORENAME**,**

DIRECTOR\_SURNAME**=**@DIRECTOR\_SURNAME**,**

DIRECTOR\_FORENAME**=**@DIRECTOR\_FORENAME

**WHERE** FILM\_ID**=**@FILM\_ID**;**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: updateCustomer

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure updates the record for a specified customer.

-- Parameters: @CUSTOMER\_RECORD - The ID number for the record being updated.

-- @FIRSTNAME - The customer's first name.

-- @LASTNAME - The customer's surname.

-- @ADDRESSLINE1 - The customer's address (Line 1)

-- @ADDRESSLINE2 - The customer's address (Line 2)

-- @POSTCODE - The customer's postcode.

-- @EMAIL - The email address of the customer.

-- @PHONE\_NO - The customer's phone number

--

-- Example usage: EXECUTE updateCustomer 'Randolph', 'Scott', '22 Pelican Grove', 'Liverpool',

-- 'L19 0NE', 'randyscott123@gmail.com', '01516666666';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'updateCustomer' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** updateCustomer**;**

**GO**

**CREATE** **PROCEDURE** updateCustomer @CUSTOMER\_ID **INT,**

@FIRSTNAME **VARCHAR(**35**),**

@LASTNAME **VARCHAR(**35**),**

@ADDRESSLINE1 **VARCHAR(**50**),**

@ADDRESSLINE2 **VARCHAR(**50**),**

@POSTCODE **VARCHAR(**10**),**

@EMAIL **VARCHAR(**50**),**

@PHONE\_NO **VARCHAR(**15**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**UPDATE** customersTBL

**SET** FIRSTNAME**=**@FIRSTNAME**,**

LASTNAME**=**@LASTNAME**,**

ADDRESSLINE1**=**@ADDRESSLINE1**,**

ADDRESSLINE2**=**@ADDRESSLINE2**,**

POSTCODE**=**@POSTCODE**,**

EMAIL**=**@EMAIL**,**

PHONE\_NO**=**@PHONE\_NO

**WHERE** CUSTOMER\_ID**=**@CUSTOMER\_ID

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: updateScreen

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure updates the record for a specified film.

-- Parameters: @SCREEN\_NO - The ID number of a screen.

-- @CAPACITY - The capacity of the screening room.

-- @HAS\_DISABLED\_ACCESS - 1 if the screening room has disabled access, otherwise 0.

-- @HAS\_HEARING\_LOOP - 1 if the screening room has a hearing loop, otherwise 0.

--

-- Example usage: EXECUTE updateScreen 1, 200, 1, 0;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'updateScreen' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** updateScreen**;**

**GO**

**CREATE** **PROCEDURE** updateScreen @SCREEN\_NO **INT,**

@CAPACITY **SMALLINT,**

@HAS\_DISABLED\_ACCESS **BIT,**

@HAS\_HEARING\_LOOP **BIT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**UPDATE** screenTBL

**SET** CAPACITY**=**@CAPACITY**,**

HAS\_DISABLED\_ACCESS**=**@HAS\_DISABLED\_ACCESS**,**

HAS\_HEARING\_LOOP**=**@HAS\_HEARING\_LOOP

**WHERE** SCREEN\_NO**=**@SCREEN\_NO

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: updateBooking

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure updates the record for a specified booking.

-- Parameters: @BOOKING\_ID - The ID number for the record being updated.

-- @CUSTOMER\_ID - The customer's ID number

-- @PERFORMANCE\_ID - The ID number for the performance.

-- @NO\_OF\_TICKETS - The number of tickets being booked.

--

-- Example usage: EXECUTE updateBooking 2,4,2,5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'updateBooking' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** updateBooking**;**

**GO**

**CREATE** **PROCEDURE** updateBooking @BOOKING\_ID **INT,**

@CUSTOMER\_ID **INT,**

@PERFORMANCE\_ID **INT,**

@NO\_OF\_TICKETS **SMALLINT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**BEGIN** **TRANSACTION;** -- Mark the starting point of the transaction.

SAVE **TRANSACTION** updateBookingSave**;** -- Name the save point.  
 -- Run the update within a try-catch statement.   
 -- If anything goes wrong then the transaction will be rolled back.

**BEGIN** TRY  
 **UPDATE** bookingsTBL

**SET** CUSTOMER\_ID**=**@CUSTOMER\_ID**,**

PERFORMANCE\_ID**=**@PERFORMANCE\_ID**,**

NO\_OF\_TICKETS**=**@NO\_OF\_TICKETS

**WHERE** BOOKING\_ID**=**@BOOKING\_ID

**END** TRY

**BEGIN** CATCH

**IF** @@TRANCOUNT **>** 0

**BEGIN** -- Undo any changes made since the save point.

**ROLLBACK** **TRANSACTION** updateBookingSave

PRINT 'Unable to update Booking record'

**END**

**END** CATCH  
 -- If the code ran correctly commit the changes to the database.

**COMMIT** **TRANSACTION**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: updatePerformance

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure updates the record for a specified performance

-- Parameters: @PERFORMANCE\_ID - The ID number for the record being updated.

-- @FILM\_ID - The film's ID number.

-- @SCREEN\_NO - The number of the screen where the film will be projected.

-- @PERFORMANCE\_DATE - The date when the film will be projected.

-- @START\_TIME - The start time for the performance.

-- @PRICE - The cost of a single ticket for the performance.

-- @IN\_3D - 1 if the film is in 3D, otherwise 0.

--

-- Example usage: EXECUTE updatePerformance 2,11,3,'2018-08-25','20:00:00',12.99,0;

--

-- Notes: The END\_TIME is calculated using the getEndTime function.

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'updatePerformance' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** updatePerformance**;**

**GO**

**CREATE** **PROCEDURE** updatePerformance @PERFORMANCE\_ID **INT,**

@FILM\_ID **INT,**

@SCREEN\_NO **INT,**

@PERFORMANCE\_DATE **DATE,**

@START\_TIME **TIME,**

@PRICE MONEY**,**

@IN\_3D **BIT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**BEGIN** **TRANSACTION;**

SAVE **TRANSACTION** updatePerformanceSave**;**

**BEGIN** TRY

-- Calculate the END\_TIME.

**DECLARE** @duration **TIME**

**DECLARE** @END\_TIME **TIME**

-- Get the duration for the film from filmTBL.

**SET** @duration **=** **(SELECT** duration **from** filmsTBL **where** film\_ID **=** @FILM\_ID**)**

-- Calculate the end time for the film.

**SET** @END\_TIME **=** dbo**.**getEndTime**(**@START\_TIME**,**@duration**);**

-- UPDATE the record.

**UPDATE** performanceTBL

**SET** FILM\_ID**=**@FILM\_ID**,**

SCREEN\_NO**=**@SCREEN\_NO**,**

PERFORMANCE\_DATE**=**@PERFORMANCE\_DATE**,**

START\_TIME**=**@START\_TIME**,**

END\_TIME**=**@END\_TIME**,**

PRICE**=**@PRICE**,**

IN\_3D**=**@IN\_3D

**WHERE** PERFORMANCE\_ID**=**@PERFORMANCE\_ID

**END** TRY

**BEGIN** CATCH

**IF** @@TRANCOUNT **>** 0

**BEGIN**

**ROLLBACK** **TRANSACTION** updatePerformanceSave

PRINT 'Unable to update Performance record'

**END**

**END** CATCH

**COMMIT** **TRANSACTION**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- ----------------------------------------------------------------------------

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- @ CREATE THE SELECT PROCEDURES @

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

PRINT 'Create the SELECT procedures and functions.'

-- ----------------------------------------------------------------------------

-- Procedure Name: getFilmRecordByID

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the record for a specified film.

-- Parameters: @FILM\_ID - The ID number for a film.

--

-- Example usage: EXECUTE getFilmRecordByID 2;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getFilmRecordByID' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getFilmRecordByID**;**

**GO**

**CREATE** **PROCEDURE** getFilmRecordByID @FILM\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** filmsTBL

**WHERE** FILM\_ID**=**@FILM\_ID

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getFilmIDsByTitle

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Parameters: @TITLE - The start of or the whole of a film title.

--

-- Example usage: EXECUTE getFilmIDsByTitle 'La';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getFilmIDsByTitle' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getFilmIDsByTitle**;**

**GO**

**CREATE** **PROCEDURE** getFilmIDsByTitle @TITLE **VARCHAR(**100**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DECLARE** @QUERY **VARCHAR(**100**)**

-- Add the supplied title and wildcard symbol to the query.

**SET** @QUERY**=**@TITLE**+**'%'

**SELECT** FILM\_ID**,** TITLE **FROM** filmsTBL

**WHERE** TITLE **LIKE** @QUERY

-- Sort the results by title in ascending order.

**ORDER** **BY** TITLE **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getFilmRecordsByActorSurname

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns details for all film records with a specified actor surname.

-- Parameters: @ACTOR\_SURNAME - An actor's surname.

--

-- Example usage: EXECUTE getFilmRecordsByActorSurname 'Mastroianni';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getFilmRecordsByActorSurname' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getFilmRecordsByActorSurname**;**

**GO**

**CREATE** **PROCEDURE** getFilmRecordsByActorSurname @ACTOR\_SURNAME **VARCHAR(**35**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

-- Format the returned results as hh:mm (e.g., 12:45)

**SELECT** ACTOR\_SURNAME**,** ACTOR\_FORENAME**,** TITLE**,** GENRE**,**

**FORMAT(**duration**,** N'hh\:mm'**)** "DURATION"

**FROM** filmsTBL

**WHERE**

ACTOR\_SURNAME**=**@ACTOR\_SURNAME

-- Sort the results first by duration in ascending order,

-- then by title in ascending order.

**ORDER** **BY** DURATION **ASC,** TITLE **ASC;**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getFilmRecordsByActressSurname

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns details for all film records with a specified actor surname.

-- Parameters: @ACTRESS\_SURNAME - An actor's surname.

--

-- Example usage: EXECUTE getFilmRecordsByActressSurname 'Loren';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getFilmRecordsByActressSurname' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getFilmRecordsByActressSurname**;**

**GO**

**CREATE** **PROCEDURE** getFilmRecordsByActressSurname @ACTRESS\_SURNAME **VARCHAR(**35**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** ACTRESS\_SURNAME**,** ACTRESS\_FORENAME**,** TITLE**,** GENRE**,**

**FORMAT(**duration**,** N'hh\:mm'**)** "DURATION"

**FROM** filmsTBL

**WHERE**

ACTRESS\_SURNAME**=**@ACTRESS\_SURNAME

**ORDER** **BY** DURATION **ASC,** TITLE **ASC;**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getFilmRecordsByDirectorSurname

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns details for all film records with   
-- a specified director surname.

-- Parameters: @DIRECTOR\_SURNAME - A film director's surname.

--

-- Example usage: EXECUTE getFilmRecordsByDirectorSurname 'Hitchcock';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getFilmRecordsByDirectorSurname' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getFilmRecordsByDirectorSurname**;**

**GO**

**CREATE** **PROCEDURE** getFilmRecordsByDirectorSurname @DIRECTOR\_SURNAME **VARCHAR(**35**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** DIRECTOR\_SURNAME**,** DIRECTOR\_FORENAME**,** TITLE**,** GENRE**,**

**FORMAT(**duration**,** N'hh\:mm'**)** "DURATION"

**FROM** filmsTBL

**WHERE**

DIRECTOR\_SURNAME**=**@DIRECTOR\_SURNAME

**ORDER** **BY** DURATION **ASC,** TITLE **ASC;**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getFilmRecordsByDuration

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns details for all films between duration   
-- @MIN\_LENGTH and @MAX\_LENGTH.

-- Parameters: @MIN\_LENGTH - The minimum duration that a film should run for.

-- @MAX\_LENGTH - The maximum duration that a film should run for.

--

-- Example usage: EXECUTE getFilmRecordsByDuration '00:45','02:30';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getFilmRecordsByDuration' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getFilmRecordsByDuration**;**

**GO**

**CREATE** **PROCEDURE** getFilmRecordsByDuration @MIN\_LENGTH **TIME,** @MAX\_LENGTH **TIME**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** TITLE**,**GENRE**,FORMAT(**duration**,** N'hh\:mm'**)** "DURATION" **from** filmsTBL

**WHERE**

-- Time datatypes cannot be compared with each other so they must first be cast to datetime.

**CAST(**DURATION **AS** **DATETIME)** **>=** **CAST(**@MIN\_LENGTH **AS** **DATETIME)**

**AND**

**CAST(**DURATION **AS** **DATETIME)** **<=** **CAST(**@MAX\_LENGTH **AS** **DATETIME)**

**ORDER** **BY** DURATION **ASC,** TITLE **ASC;**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getFilmRecordsByGenre

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns details for all films belonging to the genre specified.

-- Parameters: @GENRE - The genre that the returned films belong to.

--

-- Example usage: EXECUTE getFilmRecordsByGenre 'Comedy';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getFilmRecordsByGenre' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getFilmRecordsByGenre**;**

**GO**

**CREATE** **PROCEDURE** getFilmRecordsByGenre @GENRE **VARCHAR(**25**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** TITLE**,**GENRE**,FORMAT(**duration**,** N'hh\:mm'**)** "DURATION" **from** filmsTBL

**WHERE**

GENRE**=**@GENRE

**ORDER** **BY** TITLE **ASC;**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getAllScreenNumbers

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns all screen numbers.

-- Parameters: None

--

-- Example usage: SELECT \* FROM dbo.getAllScreenNumbers();

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAllScreenNumbers' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getAllScreenNumbers**;**

**GO**

**CREATE** **FUNCTION** getAllScreenNumbers **(** **)**

**RETURNS** **TABLE**

**AS**

**RETURN**

**(**

-- Select unique screen numbers from the table.

**SELECT** **DISTINCT** SCREEN\_NO **from** screenTBL

**);**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getAllScreenDetails

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns all screen details.

-- Parameters: None

--

-- Example usage: EXECUTE getAllScreenDetails;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAllScreenDetails' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getAllScreenDetails**;**

**GO**

**CREATE** **PROCEDURE** getAllScreenDetails

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** screenTBL

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getScreenDetailsByMinimumCapacity

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns all screen details.

-- Parameters: @MIN\_CAPACITY - The minimum number of seats available in the screening room.

--

-- Example usage: EXECUTE getScreenDetailsByMinimumCapacity 250;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getScreenDetailsByMinimumCapacity' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getScreenDetailsByMinimumCapacity**;**

**GO**

**CREATE** **PROCEDURE** getScreenDetailsByMinimumCapacity @MIN\_CAPACITY **SMALLINT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** screenTBL

**WHERE** CAPACITY**>=**@MIN\_CAPACITY

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getScreenDetailsByScreenNumber

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the details for a specified screen.

-- Parameters: @SCREEN\_NO - The ID number of a screen.

--

-- Example usage: EXECUTE getScreenDetailsByScreenNumber 2;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getScreenDetailsByScreenNumber' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getScreenDetailsByScreenNumber**;**

**GO**

**CREATE** **PROCEDURE** getScreenDetailsByScreenNumber @SCREEN\_NO **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** screenTBL

**WHERE** SCREEN\_NO**=**@SCREEN\_NO

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getScreensWithDisabledAccess

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the details for screens with disabled access.

-- Parameters: None

--

-- Example usage: EXECUTE getScreensWithDisabledAccess;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getScreensWithDisabledAccess' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getScreensWithDisabledAccess**;**

**GO**

**CREATE** **PROCEDURE** getScreensWithDisabledAccess

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** screenTBL

**WHERE** HAS\_DISABLED\_ACCESS**=**1

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getScreensWithHearingLoop

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the details for screens with hearing loops.

-- Parameters: None

--

-- Example usage: EXECUTE getScreensWithHearingLoop;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getScreensWithHearingLoop' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getScreensWithHearingLoop**;**

**GO**

**CREATE** **PROCEDURE** getScreensWithHearingLoop

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** screenTBL

**WHERE** HAS\_HEARING\_LOOP**=**1

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getBookingRecordByID

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the record for a specified booking.

-- Parameters: @BOOKING\_ID - The ID number for a booking record.

--

-- Example usage: EXECUTE getBookingRecordByID 2;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getBookingRecordByID' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getBookingRecordByID**;**

**GO**

**CREATE** **PROCEDURE** getBookingRecordByID @BOOKING\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** bookingsTBL

**WHERE** BOOKING\_ID**=**@BOOKING\_ID

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getBookingRecordsByCustomerID

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the booking records for a specified customer ID.

-- Parameters: @CUSTOMER\_ID - Customer ID number used to find booking records.

--

-- Example usage: EXECUTE getBookingRecordsByCustomerID 7;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getBookingRecordsByCustomerID' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getBookingRecordsByCustomerID**;**

**GO**

**CREATE** **PROCEDURE** getBookingRecordsByCustomerID @CUSTOMER\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** bookingsTBL

**WHERE** CUSTOMER\_ID**=**@CUSTOMER\_ID

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getBookingRecordsByFilmID

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the booking records for a specified film ID.

-- Parameters: @FILM\_ID - Film ID number used to find booking records.

--

-- Example usage: EXECUTE getBookingRecordsByFilmID 11;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getBookingRecordsByFilmID' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getBookingRecordsByFilmID**;**

**GO**

**CREATE** **PROCEDURE** getBookingRecordsByFilmID @FILM\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** BT**.**BOOKING\_ID**,**BT**.**CUSTOMER\_ID**,**BT**.**PERFORMANCE\_ID**,**BT**.**NO\_OF\_TICKETS

**FROM** bookingsTBL "BT"

-- Perform an INNER JOIN on the performance table to get performance records

-- with the supplied filmID, so that the booking records can be filtered to

-- contain the appropriate performances.

**INNER** **JOIN** performanceTBL "PT" **ON** **(**BT**.**PERFORMANCE\_ID**=**PT**.**PERFORMANCE\_ID**)**

**WHERE**

PT**.**FILM\_ID**=**@FILM\_ID

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getBookingRecordsForDate

-- Author: Andrew Laing

-- Last updated 03/01/2019

-- Description: This procedure searches for all booking records for the date specified.

-- Parameters: @DATE - The date of the query.

--

-- Example usage: EXECUTE getBookingRecordsForDate '2018-08-23';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getBookingRecordsForDate' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getBookingRecordsForDate**;**

**GO**

**CREATE** **PROCEDURE** getBookingRecordsForDate @**DATE** **DATE**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DECLARE** @theDate **DATETIME**

**DECLARE** @dayAfter **DATETIME**

**SET** @theDate **=** **cast(**@**DATE** **as** **datetime)**

**SET** @dayAfter **=** **DATEADD(DAY,** 1**,** @theDate**)**

**SELECT** BT**.**BOOKING\_ID**,**BT**.**CUSTOMER\_ID**,**BT**.**PERFORMANCE\_ID**,**BT**.**NO\_OF\_TICKETS

**FROM** bookingsTBL "BT"

-- Perform an INNER JOIN to get performances on the date specified

-- Note: it is done using the two comparisons to get all records between the start of

-- the day and Midnight because comparisons can only be done with datetime.

**INNER** **JOIN** performanceTBL "PT" **ON** **(**BT**.**PERFORMANCE\_ID**=**PT**.**PERFORMANCE\_ID**)**

**WHERE**

PT**.**PERFORMANCE\_DATE**>=**@theDate

**AND**

PT**.**PERFORMANCE\_DATE**<**@dayAfter

**ORDER** **BY** PT**.**PERFORMANCE\_DATE **ASC,** PT**.**START\_TIME

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getBookingRecordsForPeriod

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure searches for all booking records for the dates

-- between and including @MIN\_DATE and @MAX\_DATE.

-- Parameters: @MIN\_DATE - The start date of the query period.

-- @MAX\_DATE - The end date of the query period.

--

-- Example usage: EXECUTE getBookingRecordsForPeriod '2018-08-23','2018-08-25';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getBookingRecordsForPeriod' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getBookingRecordsForPeriod**;**

**GO**

**CREATE** **PROCEDURE** getBookingRecordsForPeriod @MIN\_DATE **DATE,** @MAX\_DATE **DATE**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** BT**.**BOOKING\_ID**,**BT**.**CUSTOMER\_ID**,**BT**.**PERFORMANCE\_ID**,**BT**.**NO\_OF\_TICKETS

**FROM** bookingsTBL "BT"

**INNER** **JOIN** performanceTBL "PT" **ON** **(**BT**.**PERFORMANCE\_ID**=**PT**.**PERFORMANCE\_ID**)**

**WHERE**

PT**.**PERFORMANCE\_DATE **>=** **CAST(**@MIN\_DATE **AS** **DATETIME)**

**AND**

PT**.**PERFORMANCE\_DATE **<=** **CAST(**@MAX\_DATE **AS** **DATETIME)**

**ORDER** **BY** PT**.**PERFORMANCE\_DATE **ASC,** PT**.**START\_TIME

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getBookingRecordsForToday

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure searches for all booking records for today.

-- Parameters: None.

--

-- Example usage: EXECUTE getBookingRecordsForToday;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getBookingRecordsForToday' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getBookingRecordsForToday**;**

**GO**

**CREATE** **PROCEDURE** getBookingRecordsForToday

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DECLARE** @todaysDate **DATETIME**

**DECLARE** @tomorrowsDate **DATETIME**

-- Note: getDate supplies the day and time. By converting it to a date and then

-- back to a datetime it is set to the start of the day and will therefore

-- find all records matching today instead of only those after the current time.

**SET** @todaysDate **=** **cast(cast(getdate()** **as** **date** **)** **as** **datetime)**

-- Get the start of tomorrow by adding 1 day to today.

**SET** @tomorrowsDate **=** **DATEADD(DAY,** 1**,** @todaysDate**)**

**SELECT** BT**.**BOOKING\_ID**,**BT**.**CUSTOMER\_ID**,**BT**.**PERFORMANCE\_ID**,**BT**.**NO\_OF\_TICKETS

**FROM** bookingsTBL "BT"

**INNER** **JOIN** performanceTBL "PT" **ON** **(**BT**.**PERFORMANCE\_ID**=**PT**.**PERFORMANCE\_ID**)**

**WHERE**

PERFORMANCE\_DATE**>=**@todaysDate

**AND**

PERFORMANCE\_DATE**<**@tomorrowsDate

**ORDER** **BY** PT**.**PERFORMANCE\_DATE **ASC,** PT**.**START\_TIME

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getBookingRecordsForTomorrow

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure searches for all booking records for tomorrow.

-- Parameters: None.

--

-- Example usage: EXECUTE getBookingRecordsForTomorrow;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getBookingRecordsForTomorrow' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getBookingRecordsForTomorrow**;**

**GO**

**CREATE** **PROCEDURE** getBookingRecordsForTomorrow

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DECLARE** @today **DATETIME**

**DECLARE** @tomorrow **DATETIME**

**DECLARE** @dayAfterTomorrow **DATETIME**

-- Calculate the start of tomorrow by adding 1 day to today.

**SET** @today **=** **cast(cast(getdate()** **as** **date** **)** **as** **datetime)**

**SET** @tomorrow **=** **cast(DATEADD(DAY,** 1**,** @today**)** **as** **datetime)**

-- Calculate the day after tomorrow by adding 1 day to tomorrow.

**SET** @dayAfterTomorrow **=** **DATEADD(DAY,** 1**,** @tomorrow**)**

**SELECT** BT**.**BOOKING\_ID**,**BT**.**CUSTOMER\_ID**,**BT**.**PERFORMANCE\_ID**,**BT**.**NO\_OF\_TICKETS

**FROM** bookingsTBL "BT"

**INNER** **JOIN** performanceTBL "PT" **ON** **(**BT**.**PERFORMANCE\_ID**=**PT**.**PERFORMANCE\_ID**)**

**WHERE**

PERFORMANCE\_DATE**>=**@tomorrow

**AND**

PERFORMANCE\_DATE**<**@dayAfterTomorrow

**ORDER** **BY** PT**.**PERFORMANCE\_DATE **ASC,** PT**.**START\_TIME

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getBookingRecordsByPerformanceID

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the booking records for a specified performance ID.

-- Parameters: @PERFORMANCE\_ID - Performance ID number used to find booking records.

--

-- Example usage: EXECUTE getBookingRecordsByPerformanceID 7;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getBookingRecordsByPerformanceID' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getBookingRecordsByPerformanceID**;**

**GO**

**CREATE** **PROCEDURE** getBookingRecordsByPerformanceID @PERFORMANCE\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** bookingsTBL

**WHERE** PERFORMANCE\_ID**=**@PERFORMANCE\_ID

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBooked

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked.

-- Parameters: None.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBooked() AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBooked' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBooked**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBooked**()**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

-- If the result is null, coalesce will return 0 instead.

**SELECT** **COALESCE(SUM(**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL" **FROM** bookingsTBL

**)**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByActorName

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for a specified film.

-- Parameters: @ACTOR\_FORENAME - The first name of an actor.

-- @ACTOR\_SURNAME - The surname of an actor.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByActorName('Marcello',  
 'Mastroianni') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByActorName' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByActorName**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByActorName **(** @ACTOR\_FORENAME **VARCHAR(**35**),**

@ACTOR\_SURNAME **VARCHAR(**35**)** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**BT**.**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL"

**FROM** **(** filmsTBL "FT" **INNER** **JOIN** performanceTBL "PT" **ON** FT**.**FILM\_ID **=** PT**.**FILM\_ID **)**

**INNER** **JOIN** bookingsTBL "BT" **ON** PT**.**PERFORMANCE\_ID **=** BT**.**PERFORMANCE\_ID

**WHERE** FT**.**ACTOR\_FORENAME**=**@ACTOR\_FORENAME

**AND** FT**.**ACTOR\_SURNAME**=**@ACTOR\_SURNAME

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByActressName

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for a specified film.

-- Parameters: @ACTRESS\_FORENAME - The first name of an actress.

-- @ACTRESS\_SURNAME - The surname of an actress.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByActressName('Anouk','Aimee') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByActressName' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByActressName**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByActressName **(** @ACTRESS\_FORENAME **VARCHAR(**35**),**

@ACTRESS\_SURNAME **VARCHAR(**35**)** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**BT**.**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL"

**FROM** **(** filmsTBL "FT" **INNER** **JOIN** performanceTBL "PT" **ON** FT**.**FILM\_ID **=** PT**.**FILM\_ID **)**

**INNER** **JOIN** bookingsTBL "BT" **ON** PT**.**PERFORMANCE\_ID **=** BT**.**PERFORMANCE\_ID

**WHERE** FT**.**ACTRESS\_FORENAME**=**@ACTRESS\_FORENAME

**AND** FT**.**ACTRESS\_SURNAME**=**@ACTRESS\_SURNAME

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByCustomerID

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked by a specified customer ID.

-- Parameters: @CUSTOMER\_ID - The ID number for a customer record.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByCustomerID(4) AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByCustomerID' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByCustomerID**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByCustomerID **(** @CUSTOMER\_ID **INT** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL" **FROM** bookingsTBL

**WHERE** CUSTOMER\_ID**=**@CUSTOMER\_ID

**)**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByDirectorName

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for a specified director.

-- Parameters: @DIRECTOR\_FORENAME - The first name of a director.

-- @DIRECTOR\_SURNAME - The surname of a director.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByDirectorName('Federico','Fellini') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByDirectorName' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByDirectorName**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByDirectorName **(** @DIRECTOR\_FORENAME **VARCHAR(**35**),**

@DIRECTOR\_SURNAME **VARCHAR(**35**)** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**BT**.**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL"

**FROM** **(** filmsTBL "FT" **INNER** **JOIN** performanceTBL "PT" **ON** FT**.**FILM\_ID **=** PT**.**FILM\_ID **)**

**INNER** **JOIN** bookingsTBL "BT" **ON** PT**.**PERFORMANCE\_ID **=** BT**.**PERFORMANCE\_ID

**WHERE** FT**.**DIRECTOR\_FORENAME**=**@DIRECTOR\_FORENAME

**AND** FT**.**DIRECTOR\_SURNAME**=**@DIRECTOR\_SURNAME

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByFilmID

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for a specified film.

-- Parameters: @FILM\_ID - The ID number for a film.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByFilmID(7) AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByFilmID' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByFilmID**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByFilmID **(** @FILM\_ID **INT** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL" **FROM** bookingsTBL "BT"

**INNER** **JOIN** performanceTBL "PT" **ON** **(** PT**.**PERFORMANCE\_ID**=**BT**.**PERFORMANCE\_ID **)**

**WHERE** PT**.**FILM\_ID**=**@FILM\_ID

**)**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByFilmTitle

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for a specified film.

-- Parameters: @FILM\_TITLE - The full title of a film.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByFilmTitle('La Haine') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByFilmTitle' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByFilmTitle**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByFilmTitle **(** @FILM\_TITLE **VARCHAR(**100**)** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**BT**.**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL"

**FROM** **(** filmsTBL "FT" **INNER** **JOIN** performanceTBL "PT" **ON** FT**.**FILM\_ID **=** PT**.**FILM\_ID **)**

**INNER** **JOIN** bookingsTBL "BT" **ON** PT**.**PERFORMANCE\_ID **=** BT**.**PERFORMANCE\_ID

**WHERE** FT**.**TITLE **Like** @FILM\_TITLE

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByGenre

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for a specified genre.

-- Parameters: @GENRE - A film genre.(e.g., Comedy)

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByGenre('Comedy') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByGenre' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByGenre**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByGenre **(** @GENRE **VARCHAR(**25**)** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**BT**.**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL"

**FROM** **(** filmsTBL "FT" **INNER** **JOIN** performanceTBL "PT" **ON** FT**.**FILM\_ID **=** PT**.**FILM\_ID **)**

**INNER** **JOIN** bookingsTBL "BT" **ON** PT**.**PERFORMANCE\_ID **=** BT**.**PERFORMANCE\_ID

**WHERE** FT**.**GENRE **Like** @GENRE

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByPerformanceID

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked   
-- for a specified performance.

-- Parameters: @PERFORMANCE\_ID - The ID number for a film performance.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByPerformanceID(3) AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByPerformanceID' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByPerformanceID**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByPerformanceID **(** @PERFORMANCE\_ID **INT** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL" **FROM** bookingsTBL

**WHERE** PERFORMANCE\_ID**=**@PERFORMANCE\_ID

**)**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByScreenNo

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for a specified screen.

-- Parameters: @SCREEN\_NO - A screen number.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByScreenNo(2) AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByScreenNo' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByScreenNo**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByScreenNo **(** @SCREEN\_NO **INT** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL" **FROM** bookingsTBL "BT"

**INNER** **JOIN** performanceTBL "PT" **ON** **(** PT**.**PERFORMANCE\_ID**=**BT**.**PERFORMANCE\_ID **)**

**WHERE** PT**.**SCREEN\_NO**=**@SCREEN\_NO

**)**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedInPeriod

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for   
-- a specified period.

-- Parameters: @MIN\_DATE - The start date of the search period.

-- @MAX\_DATE - The finish date of the search period.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedInPeriod('2018-08-21', '2018-08-23') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedInPeriod' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedInPeriod**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedInPeriod **(** @MIN\_DATE **DATE,** @MAX\_DATE **DATE** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**BT**.**NO\_OF\_TICKETS**),** 0**)** **AS** "TOTAL"

**FROM** performanceTBL "PT"

**INNER** **JOIN** bookingsTBL "BT"

**ON** PT**.**PERFORMANCE\_ID **=** BT**.**PERFORMANCE\_ID

**WHERE** PT**.**PERFORMANCE\_DATE**>=**@MIN\_DATE

**AND**

PT**.**PERFORMANCE\_DATE**<=**@MAX\_DATE

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedOnDate

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked on a specified day.

-- Parameters: @PERFORMANCE\_DATE - The search date.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedOnDate('2018-08-22') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedOnDate' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedOnDate**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedOnDate **(** @PERFORMANCE\_DATE **DATE** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**BT**.**NO\_OF\_TICKETS**),** 0**)** **AS** "TOTAL"

**FROM** performanceTBL "PT"

**INNER** **JOIN** bookingsTBL "BT"

**ON** PT**.**PERFORMANCE\_ID **=** BT**.**PERFORMANCE\_ID

**WHERE** PT**.**PERFORMANCE\_DATE**=**@PERFORMANCE\_DATE

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByCustomerIDDuringPeriod

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for

-- a specified customer during a specified period.

-- Parameters: @CUSTOMER\_ID - A customer's ID number.

-- @MIN\_DATE - The start date of the search period.

-- @MAX\_DATE - The finish date of the search period.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByCustomerIDDuringPeriod(4, '2018-08-21',

-- '2018-08-23') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByCustomerIDDuringPeriod' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByCustomerIDDuringPeriod**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByCustomerIDDuringPeriod **(** @CUSTOMER\_ID **INT,**

@MIN\_DATE **DATE,** @MAX\_DATE **DATE** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**BT**.**NO\_OF\_TICKETS**),** 0**)** **AS** "TOTAL"

**FROM** performanceTBL "PT"

**INNER** **JOIN** bookingsTBL "BT"

**ON** PT**.**PERFORMANCE\_ID **=** BT**.**PERFORMANCE\_ID

**WHERE** BT**.**CUSTOMER\_ID**=**@CUSTOMER\_ID

**AND**

PT**.**PERFORMANCE\_DATE**>=**@MIN\_DATE

**AND**

PT**.**PERFORMANCE\_DATE**<=**@MAX\_DATE

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByFilmIDDuringPeriod

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for

-- a specified film during a specified period.

-- Parameters: @FILM\_ID - A film's ID number.

-- @MIN\_DATE - The start date of the search period.

-- @MAX\_DATE - The finish date of the search period.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByFilmIDDuringPeriod(7, '2018-08-21',

-- '2018-08-23') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByFilmIDDuringPeriod' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByFilmIDDuringPeriod**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByFilmIDDuringPeriod **(** @FILM\_ID **INT,**

@MIN\_DATE **DATE,** @MAX\_DATE **DATE** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**BT**.**NO\_OF\_TICKETS**),** 0**)** **AS** "TOTAL"

**FROM** performanceTBL "PT"

**INNER** **JOIN** bookingsTBL "BT"

**ON** PT**.**PERFORMANCE\_ID **=** BT**.**PERFORMANCE\_ID

**WHERE** PT**.**FILM\_ID**=**@FILM\_ID

**AND**

PT**.**PERFORMANCE\_DATE**>=**@MIN\_DATE

**AND**

PT**.**PERFORMANCE\_DATE**<=**@MAX\_DATE

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByGenreDuringPeriod

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for

-- a specified genre during a specified period.

-- Parameters: @GENRE - A film genre.(e.g., Comedy)

-- @MIN\_DATE - The start date of the search period.

-- @MAX\_DATE - The finish date of the search period.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByGenreDuringPeriod('Comedy', '2018-08-21',

-- '2018-08-23') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByGenreDuringPeriod' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByGenreDuringPeriod**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByGenreDuringPeriod **(** @GENRE **VARCHAR(**25**),**

@MIN\_DATE **DATE,** @MAX\_DATE **DATE** **)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**BT**.**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL"

**FROM** **(** filmsTBL "FT" **INNER** **JOIN** performanceTBL "PT" **ON** FT**.**FILM\_ID **=** PT**.**FILM\_ID **)**

**INNER** **JOIN** bookingsTBL "BT" **ON** PT**.**PERFORMANCE\_ID **=** BT**.**PERFORMANCE\_ID

**WHERE** FT**.**GENRE **Like** @GENRE

**AND**

PT**.**PERFORMANCE\_DATE**>=**@MIN\_DATE

**AND**

PT**.**PERFORMANCE\_DATE**<=**@MAX\_DATE

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByScreenNoDuringPeriod

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for

-- a specified screen during a specified period.

-- Parameters: @SCREEN\_NO - A screen number.

-- @MIN\_DATE - The start date of the search period.

-- @MAX\_DATE - The finish date of the search period.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByScreenNoDuringPeriod(3, '2018-08-21',

-- '2018-08-23') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByScreenNoDuringPeriod' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByScreenNoDuringPeriod**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByScreenNoDuringPeriod **(** @SCREEN\_NO **INT,**

@MIN\_DATE **DATE,** @MAX\_DATE **DATE)**

**RETURNS** **INT**

**AS**

**BEGIN**

**RETURN**

**(**

**SELECT** **COALESCE(SUM(**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL" **FROM** bookingsTBL "BT"

**INNER** **JOIN** performanceTBL "PT" **ON** **(** PT**.**PERFORMANCE\_ID**=**BT**.**PERFORMANCE\_ID **)**

**WHERE** PT**.**SCREEN\_NO**=**@SCREEN\_NO

**AND**

PT**.**PERFORMANCE\_DATE**>=**@MIN\_DATE

**AND**

PT**.**PERFORMANCE\_DATE**<=**@MAX\_DATE

**)**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedDuringPreviousWeek

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked   
-- during the previous week.

-- Parameters: None.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedDuringPreviousWeek() AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedDuringPreviousWeek' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedDuringPreviousWeek**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedDuringPreviousWeek**()**

**RETURNS** **INT**

**AS**

**BEGIN**

**DECLARE** @startDate **DATETIME;**

**DECLARE** @endDate **DATETIME;**

**SET** @endDate **=** **GETDATE();** -- set end date as today.

**SET** @startDate **=** **DATEADD(DAY,** -7**,** @EndDate**)** -- set start date as today minus 7 days.

**RETURN**

**(**

**SELECT** dbo**.**getTotalNoOfTicketsBookedInPeriod**(**@startDate**,** @endDate**)**

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByCustomerDuringPreviousWeek

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked by a customer during

-- the previous week.

-- Parameters: @CUSTOMER\_ID - A customer's ID Number.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByCustomerDuringPreviousWeek(3) AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByCustomerDuringPreviousWeek' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByCustomerDuringPreviousWeek**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByCustomerDuringPreviousWeek**(**@CUSTOMER\_ID **INT)**

**RETURNS** **INT**

**AS**

**BEGIN**

**DECLARE** @startDate **DATETIME;**

**DECLARE** @endDate **DATETIME;**

**SET** @endDate **=** **GETDATE();** -- Set end date as today.

**SET** @startDate **=** **DATEADD(DAY,** -7**,** @EndDate**)** -- Set start date as today minus 7 days.

**RETURN**

**(**

**SELECT** dbo**.**getTotalNoOfTicketsBookedByCustomerIDDuringPeriod**(**@CUSTOMER\_ID**,**@startDate**,** @endDate**)**

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByFilmIDDuringPreviousWeek

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked for a film

-- during the previous week.

-- Parameters: @FILM\_ID - A film's ID number.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByFilmIDDuringPreviousWeek(4) AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByFilmIDDuringPreviousWeek' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByFilmIDDuringPreviousWeek**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByFilmIDDuringPreviousWeek**(**@FILM\_ID **INT)**

**RETURNS** **INT**

**AS**

**BEGIN**

**DECLARE** @startDate **DATETIME;**

**DECLARE** @endDate **DATETIME;**

**SET** @endDate **=** **GETDATE();** -- Set end date as today.

**SET** @startDate **=** **DATEADD(DAY,** -7**,** @EndDate**)** -- Set start date as today minus 7 days.

**RETURN**

**(**

**SELECT** dbo**.**getTotalNoOfTicketsBookedByFilmIDDuringPeriod**(**@FILM\_ID**,**@startDate**,** @endDate**)**

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByGenreDuringPreviousWeek

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked by genre

-- during the previous week.

-- Parameters: @GENRE - A genre of films.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByGenreDuringPreviousWeek('Comedy') AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByGenreDuringPreviousWeek' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByGenreDuringPreviousWeek**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByGenreDuringPreviousWeek**(**@GENRE **VARCHAR(**25**))**

**RETURNS** **INT**

**AS**

**BEGIN**

**DECLARE** @startDate **DATETIME;**

**DECLARE** @endDate **DATETIME;**

**SET** @endDate **=** **GETDATE();** -- Set end date as today.

**SET** @startDate **=** **DATEADD(DAY,** -7**,** @EndDate**)** -- Set start date as today minus 7 days.

**RETURN**

**(**

**SELECT** dbo**.**getTotalNoOfTicketsBookedByGenreDuringPeriod**(**@GENRE**,**@startDate**,** @endDate**)**

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getTotalNoOfTicketsBookedByScreenNoDuringPreviousWeek

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns the total number of tickets booked by screen number

-- during the previous week.

-- Parameters: @SCREEN\_NO - The number of a screen.

--

-- Example usage: SELECT dbo.getTotalNoOfTicketsBookedByScreenNoDuringPreviousWeek(2) AS "TOTAL";

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTotalNoOfTicketsBookedByScreenNoDuringPreviousWeek' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getTotalNoOfTicketsBookedByScreenNoDuringPreviousWeek**;**

**GO**

**CREATE** **FUNCTION** getTotalNoOfTicketsBookedByScreenNoDuringPreviousWeek**(**@SCREEN\_NO **INT)**

**RETURNS** **INT**

**AS**

**BEGIN**

**DECLARE** @startDate **DATETIME;**

**DECLARE** @endDate **DATETIME;**

**SET** @endDate **=** **GETDATE();** -- Set end date as today.

**SET** @startDate **=** **DATEADD(DAY,** -7**,** @EndDate**)** -- Set start date as today minus 7 days.

**RETURN**

**(**

**SELECT** dbo**.**getTotalNoOfTicketsBookedByScreenNoDuringPeriod**(**@SCREEN\_NO**,**@startDate**,** @endDate**)**

**);**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getCustomerRecordByID

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the record for a specified customer

-- Parameters: @CUSTOMER\_ID - The ID number for a customer record.

--

-- Example usage: EXECUTE getCustomerRecordByID 2;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getCustomerRecordByID' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getCustomerRecordByID**;**

**GO**

**CREATE** **PROCEDURE** getCustomerRecordByID @CUSTOMER\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** customersTBL

**WHERE** CUSTOMER\_ID**=**@CUSTOMER\_ID

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getCustomerRecordsByEmail

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the customer record with the specified email.

-- Parameters: @CUSTOMER\_EMAIL - A customer's postcode address.

--

-- Example usage: EXECUTE getCustomerRecordsByEmail 'sbiggins@gmail.com';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getCustomerRecordsByEmail' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getCustomerRecordsByEmail**;**

**GO**

**CREATE** **PROCEDURE** getCustomerRecordsByEmail @CUSTOMER\_EMAIL **VARCHAR(**50**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** customersTBL

**WHERE** EMAIL**=**@CUSTOMER\_EMAIL

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getCustomerRecordsByPhoneNo

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns customer records with the specified phone number.

-- Parameters: @PHONE\_NO - A customer's phone number.

--

-- Example usage: EXECUTE getCustomerRecordsByPhoneNo '01514448383';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getCustomerRecordsByPhoneNo' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getCustomerRecordsByPhoneNo**;**

**GO**

**CREATE** **PROCEDURE** getCustomerRecordsByPhoneNo @PHONE\_NO **VARCHAR(**15**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** customersTBL

**WHERE** PHONE\_NO**=**@PHONE\_NO

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getCustomerRecordsByPostcode

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the customer record with the specified postcode.

-- Parameters: @CUSTOMER\_POSTCODE - A customer's postcode.

--

-- Example usage: EXECUTE getCustomerRecordsByPostcode 'L22 4ES';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getCustomerRecordsByPostcode' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getCustomerRecordsByPostcode**;**

**GO**

**CREATE** **PROCEDURE** getCustomerRecordsByPostcode @CUSTOMER\_POSTCODE **VARCHAR(**10**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** customersTBL

**WHERE** POSTCODE**=**@CUSTOMER\_POSTCODE

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getCustomersWithNoBookings

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects all customers who have never made a booking.

-- Parameters: None.

--

-- Example usage: EXECUTE getCustomersWithNoBookings;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getCustomersWithNoBookings' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getCustomersWithNoBookings**;**

**GO**

**CREATE** **PROCEDURE** getCustomersWithNoBookings

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** CUSTOMER\_ID**,**

FIRSTNAME**,**

LASTNAME**,**

EMAIL**,**

POSTCODE

**FROM** customersTBL "CT"

**WHERE** dbo**.**getTotalNoOfTicketsBookedByCustomerID**(**CT**.**CUSTOMER\_ID**)=**0

**ORDER** **BY** LASTNAME

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getCustomersWithNoBookingsDuringPeriod

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects all customers who did not make a booking during the specified period.

-- Parameters: @MIN\_DATE - The start date of the search period.

-- @MAX\_DATE - The finish date of the search period.

--

-- Example usage: EXECUTE getCustomersWithNoBookingsDuringPeriod '2018-08-21','2018-08-22';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getCustomersWithNoBookingsDuringPeriod' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getCustomersWithNoBookingsDuringPeriod**;**

**GO**

**CREATE** **PROCEDURE** getCustomersWithNoBookingsDuringPeriod @MIN\_DATE **DATE,** @MAX\_DATE **DATE**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** CUSTOMER\_ID**,**

FIRSTNAME**,**

LASTNAME**,**

EMAIL**,**

POSTCODE

**FROM** customersTBL "CT"

**WHERE** dbo**.**getTotalNoOfTicketsBookedByCustomerIDDuringPeriod**(**CT**.**CUSTOMER\_ID**,**@MIN\_DATE**,**@MAX\_DATE**)=**0

**ORDER** **BY** LASTNAME

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getPerformanceRecordByID

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the record for a specified performance

-- Parameters: @PERFORMANCE\_ID - The ID number for a performance record.

--

-- Example usage: EXECUTE getPerformanceRecordByID 2;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getPerformanceRecordByID' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getPerformanceRecordByID**;**

**GO**

**CREATE** **PROCEDURE** getPerformanceRecordByID @PERFORMANCE\_ID **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **\*** **FROM** performanceTBL

**WHERE** PERFORMANCE\_ID**=**@PERFORMANCE\_ID

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getPerformanceDetailsByDateAndStartTime

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure searches for performances on a specified date at a specified time.

-- Parameters: @PERFORMANCE\_DATE - The date when the performance will take place.

-- @START\_TIME - The time when the performance will start.

-- @MINS\_DIFF - How many minutes before or after the start time entered that a

-- performance should start to be included into the result set.

--

-- Example usage: EXECUTE getPerformanceDetailsByDateAndStartTime '2018-08-22','20:00:00','00:30:00';

--

-- Notes: To perform the comparisons the times were cast first as DATETIMEs to allow

-- the add and subtract operations. The method below appears to be the fastest

-- method of performing the operation.

-- see: https://dwaincsql.com/2015/08/28/the-fastest-way-to-combine-date-and-time-data-types-to-a-datetime/

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getPerformanceDetailsByDateAndStartTime' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getPerformanceDetailsByDateAndStartTime**;**

**GO**

**CREATE** **PROCEDURE** getPerformanceDetailsByDateAndStartTime @PERFORMANCE\_DATE **DATE,**

@START\_TIME **TIME,**

@MINS\_DIFF **TIME**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** FT**.**TITLE**,** FT**.**GENRE**,** PT**.**SCREEN\_NO**,**

-- Return the date in UK format DD/MM/YYYY (e.g., 16/01/2019)

**FORMAT** **(**PT**.**PERFORMANCE\_DATE**,** 'd'**,** 'en-gb' **)** "PERFORMANCE\_DATE"**,**

-- Return the time in short format hh:mm (e.g., 12:45)

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START\_TIME"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END\_TIME"**,**

PT**.**PRICE**,**PT**.**IN\_3D

**FROM** performanceTBL "PT"

**INNER** **JOIN** filmsTBL "FT" **ON** **(** PT**.**FILM\_ID**=**FT**.**FILM\_ID **)**

**WHERE** PERFORMANCE\_DATE**=**@PERFORMANCE\_DATE

**AND**

**CAST(**START\_TIME **AS** **DATETIME)** **>=** **(CAST(**@START\_TIME **AS** **DATETIME)** **-** **CAST(**@MINS\_DIFF **AS** **DATETIME))**

**AND**

**CAST(**START\_TIME **AS** **DATETIME)** **<=** **(CAST(**@START\_TIME **AS** **DATETIME)** **+** **CAST(**@MINS\_DIFF **AS** **DATETIME))**

**ORDER** **BY** START\_TIME**,** SCREEN\_NO

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getPerformanceDetailsForToday

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure searches for all of today's performances.

-- Parameters: None

--

-- Example usage: EXECUTE getPerformanceDetailsForToday;

--

-- Notes: Today's date was first cast to a DATE format to get a datetime

-- starting at 12am rather than the time at which the query was run. It could

-- have been done by converting the performance date but this way the conversion

-- only needs to be done once.

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getPerformanceDetailsForToday' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getPerformanceDetailsForToday**;**

**GO**

**CREATE** **PROCEDURE** getPerformanceDetailsForToday

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DECLARE** @todaysDate **DATETIME**

**DECLARE** @tomorrowsDate **DATETIME**

**SET** @todaysDate **=** **cast(cast(getdate()** **as** **date** **)** **as** **datetime)**

**SET** @tomorrowsDate **=** **DATEADD(DAY,** 1**,** @todaysDate**)**

**SELECT** FT**.**TITLE**,** FT**.**GENRE**,** PT**.**SCREEN\_NO**,**

**FORMAT** **(**PT**.**PERFORMANCE\_DATE**,** 'd'**,** 'en-gb' **)** "PERFORMANCE\_DATE"**,**

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START\_TIME"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END\_TIME"**,**

PT**.**PRICE**,** PT**.**IN\_3D

**FROM** performanceTBL "PT"

**INNER** **JOIN** filmsTBL "FT" **ON** **(** PT**.**FILM\_ID**=**FT**.**FILM\_ID **)**

**WHERE**

PERFORMANCE\_DATE**>=**@todaysDate

**AND**

PERFORMANCE\_DATE**<**@tomorrowsDate

**ORDER** **BY** START\_TIME**,** SCREEN\_NO **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getPerformanceDetailsForTomorrow

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure searches for all of tomorrow's performances.

-- Parameters: None

--

-- Example usage: EXECUTE getPerformanceDetailsForTomorrow;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getPerformanceDetailsForTomorrow' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getPerformanceDetailsForTomorrow**;**

**GO**

**CREATE** **PROCEDURE** getPerformanceDetailsForTomorrow

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DECLARE** @today **DATETIME**

**DECLARE** @tomorrow **DATETIME**

**DECLARE** @dayAfterTomorrow **DATETIME**

**SET** @today **=** **cast(cast(getdate()** **as** **date** **)** **as** **datetime)**

**SET** @tomorrow **=** **cast(DATEADD(DAY,** 1**,** @today**)** **as** **datetime)**

**SET** @dayAfterTomorrow **=** **DATEADD(DAY,** 1**,** @tomorrow**)**

**SELECT** FT**.**TITLE**,** FT**.**GENRE**,** PT**.**SCREEN\_NO**,**

**FORMAT** **(**PT**.**PERFORMANCE\_DATE**,** 'd'**,** 'en-gb' **)** "PERFORMANCE\_DATE"**,**

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START\_TIME"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END\_TIME"**,**

PT**.**PRICE**,** PT**.**IN\_3D

**FROM** performanceTBL "PT"

**INNER** **JOIN** filmsTBL "FT" **ON** **(** PT**.**FILM\_ID**=**FT**.**FILM\_ID **)**

**WHERE**

PERFORMANCE\_DATE**>=**@tomorrow

**AND**

PERFORMANCE\_DATE**<**@dayAfterTomorrow

**ORDER** **BY** START\_TIME**,** SCREEN\_NO **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getPerformanceDetailsForDate

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure searches for performances on a specified date.

-- Parameters: None

--

-- Example usage: EXECUTE getPerformanceDetailsForDate '2018-08-22';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getPerformanceDetailsForDate' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getPerformanceDetailsForDate**;**

**GO**

**CREATE** **PROCEDURE** getPerformanceDetailsForDate @**DATE** **DATE**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DECLARE** @dayAfter **DATETIME**

**SET** @dayAfter **=** **DATEADD(DAY,** 1**,** @**DATE)**

**SELECT** FT**.**TITLE**,** FT**.**GENRE**,** PT**.**SCREEN\_NO**,**

**FORMAT** **(**PT**.**PERFORMANCE\_DATE**,** 'd'**,** 'en-gb' **)** "PERFORMANCE\_DATE"**,**

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START\_TIME"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END\_TIME"**,**

PT**.**PRICE**,** PT**.**IN\_3D

**FROM** performanceTBL "PT"

**INNER** **JOIN** filmsTBL "FT" **ON** **(** PT**.**FILM\_ID**=**FT**.**FILM\_ID **)**

**WHERE**

PERFORMANCE\_DATE**>=**@**DATE**

**AND**

PERFORMANCE\_DATE**<**@dayAfter

**ORDER** **BY** START\_TIME**,** SCREEN\_NO **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getPerformanceDetailsByFilmTitle

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure returns the performance records for a film specified by title.

-- Parameters: @FILM\_TITLE - The title of a film.

--

-- Example usage: EXECUTE getPerformanceDetailsByFilmTitle 'Lovely Donuts';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getPerformanceDetailsByFilmTitle' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getPerformanceDetailsByFilmTitle**;**

**GO**

**CREATE** **PROCEDURE** getPerformanceDetailsByFilmTitle @FILM\_TITLE **VARCHAR(**100**)**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** FT**.**TITLE**,** FT.GENRE, PT**.**SCREEN\_NO**,**

**FORMAT** **(**PT**.**PERFORMANCE\_DATE**,** 'd'**,** 'en-gb' **)** "PERFORMANCE\_DATE"**,**

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START\_TIME"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END\_TIME"**,**

PT**.**PRICE**,**  PT**.**IN\_3D

**FROM** performanceTBL "PT"

**INNER** **JOIN** filmsTBL "FT" **ON** **(** PT**.**FILM\_ID**=**FT**.**FILM\_ID **)**

**WHERE** FT**.**TITLE**=**@FILM\_TITLE

**ORDER** **BY** PT**.**PERFORMANCE\_DATE **ASC,** PT**.**START\_TIME **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getAll3DPerformanceDetails

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure searches for all 3D performances.

-- Parameters: None

--

-- Example usage: EXECUTE getAll3DPerformanceDetails;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAll3DPerformanceDetails' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getAll3DPerformanceDetails**;**

**GO**

**CREATE** **PROCEDURE** getAll3DPerformanceDetails

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** FT**.**TITLE**,** FT**.**GENRE**,** PT**.**SCREEN\_NO**,**

**FORMAT** **(**PT**.**PERFORMANCE\_DATE**,** 'd'**,** 'en-gb' **)** "PERFORMANCE\_DATE"**,**

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START\_TIME"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END\_TIME"**,**

PT**.**PRICE**,** PT**.**IN\_3D

**FROM** performanceTBL "PT"

**INNER** **JOIN** filmsTBL "FT" **ON** **(** PT**.**FILM\_ID**=**FT**.**FILM\_ID **)**

**WHERE** PT**.**IN\_3D**=**1

**ORDER** **BY** PT**.**PERFORMANCE\_DATE **ASC,** PT**.**START\_TIME **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: get3DPerformanceDetailsByDateAndStartTime

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This procedure searches for 3D performances on a specified date   
-- at a specified time.

-- Parameters: @PERFORMANCE\_DATE - The date when the performance will take place.

-- @START\_TIME - The time when the performance will start.

-- @MINS\_DIFF - How many minutes before or after the start time

-- entered that a performance should start to be included   
-- into the result set.

--

-- Example usage: EXECUTE get3DPerformanceDetailsByDateAndStartTime '2018-08-22','20:00:00','00:30:00';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'get3DPerformanceDetailsByDateAndStartTime' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** get3DPerformanceDetailsByDateAndStartTime**;**

**GO**

**CREATE** **PROCEDURE** get3DPerformanceDetailsByDateAndStartTime @PERFORMANCE\_DATE **DATE,**

@START\_TIME **TIME,**

@MINS\_DIFF **TIME**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** FT**.**TITLE**,** FT**.**GENRE**,** PT**.**SCREEN\_NO**,**

**FORMAT** **(**PT**.**PERFORMANCE\_DATE**,** 'd'**,** 'en-gb' **)** "PERFORMANCE\_DATE"**,**

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START\_TIME"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END\_TIME"**,**

PT**.**PRICE**,** PT**.**IN\_3D

**FROM** performanceTBL "PT"

**INNER** **JOIN** filmsTBL "FT" **ON** **(** PT**.**FILM\_ID**=**FT**.**FILM\_ID **)**

**WHERE** PT**.**IN\_3D**=**1

**AND** PERFORMANCE\_DATE**=**@PERFORMANCE\_DATE

**AND**

**CAST(**START\_TIME **AS** **DATETIME)** **>=** **(CAST(**@START\_TIME **AS** **DATETIME)** **-** **CAST(**@MINS\_DIFF **AS** **DATETIME))**

**AND**

**CAST(**START\_TIME **AS** **DATETIME)** **<=** **(CAST(**@START\_TIME **AS** **DATETIME)** **+** **CAST(**@MINS\_DIFF **AS** **DATETIME))**

**ORDER** **BY** PT**.**PERFORMANCE\_DATE **ASC,** PT**.**SCREEN\_NO**,** PT**.**START\_TIME **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNFilmBookingsOfAllTime

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N films which received the highest number of bookings.

-- Parameters: @N - The number of films to return.

--

-- Example usage: EXECUTE getTopNFilmBookingsOfAllTime 5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNFilmBookingsOfAllTime' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNFilmBookingsOfAllTime**;**

**GO**

**CREATE** **PROCEDURE** getTopNFilmBookingsOfAllTime @N **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** TOP **(**@N**)** FILM\_ID**,**

TITLE**,**

GENRE**,**

**FORMAT(**DURATION**,** N'hh\:mm'**)** **AS** DURATION**,**

-- Get total number of tickets booked for the film.

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByFilmID**(**FT**.**FILM\_ID**))** **AS** "TOTAL\_BOOKINGS"

**FROM** filmsTBL "FT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** TITLE **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNBookingCustomersOfAllTime

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N customers who made the highest number of bookings.

-- Parameters: @N - The number of customers to return.

--

-- Example usage: EXECUTE getTopNBookingCustomersOfAllTime 5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNBookingCustomersOfAllTime' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNBookingCustomersOfAllTime**;**

**GO**

**CREATE** **PROCEDURE** getTopNBookingCustomersOfAllTime @N **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** TOP **(**@N**)**

CUSTOMER\_ID**,**

FIRSTNAME**,**

LASTNAME**,**

EMAIL**,**

POSTCODE**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByCustomerID**(**CT**.**CUSTOMER\_ID**))** **AS** "TOTAL\_BOOKINGS"

**FROM** customersTBL "CT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** LASTNAME **ASC,** FIRSTNAME **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNActorBookingsOfAllTime

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N actors who received the highest number of bookings.

-- Parameters: @N - The number of actors to return.

--

-- Example usage: EXECUTE getTopNActorBookingsOfAllTime 5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNActorBookingsOfAllTime' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNActorBookingsOfAllTime**;**

**GO**

**CREATE** **PROCEDURE** getTopNActorBookingsOfAllTime @N **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

-- DISTINCT TOP() Returns n Unique rows from the top of the results.

**SELECT** **DISTINCT** TOP **(**@N**)**

ACTOR\_FORENAME**,**

ACTOR\_SURNAME**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByActorName**(**FT**.**ACTOR\_FORENAME**,**   
 FT**.**ACTOR\_SURNAME**))** **AS** "TOTAL\_BOOKINGS"

**FROM** filmsTBL "FT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** ACTOR\_SURNAME **ASC,** ACTOR\_FORENAME **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNActressBookingsOfAllTime

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N actresses who received the highest number of bookings.

-- Parameters: @N - The number of actresses to return.

--

-- Example usage: EXECUTE getTopNActressBookingsOfAllTime 5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNActressBookingsOfAllTime' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNActressBookingsOfAllTime**;**

**GO**

**CREATE** **PROCEDURE** getTopNActressBookingsOfAllTime @N **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **DISTINCT** TOP **(**@N**)**

ACTRESS\_FORENAME**,**

ACTRESS\_SURNAME**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByActressName**(**FT**.**ACTRESS\_FORENAME**,**

FT**.**ACTRESS\_SURNAME**))** **AS** "TOTAL\_BOOKINGS"

**FROM** filmsTBL "FT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** ACTRESS\_SURNAME **ASC,** ACTRESS\_FORENAME **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNDirectorBookingsOfAllTime

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N directors who received the highest number of bookings.

-- Parameters: @N - The number of directors to return.

--

-- Example usage: EXECUTE getTopNDirectorBookingsOfAllTime 5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNDirectorBookingsOfAllTime' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNDirectorBookingsOfAllTime**;**

**GO**

**CREATE** **PROCEDURE** getTopNDirectorBookingsOfAllTime @N **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **DISTINCT** TOP **(**@N**)**

DIRECTOR\_FORENAME**,**

DIRECTOR\_SURNAME**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByDirectorName**(**FT**.**DIRECTOR\_FORENAME**,**   
 FT**.**DIRECTOR\_SURNAME**))** **AS** "TOTAL\_BOOKINGS"

**FROM** filmsTBL "FT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** DIRECTOR\_SURNAME **ASC,** DIRECTOR\_FORENAME **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNGenreBookingsOfAllTime

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N genres which received the highest number of bookings.

-- Parameters: @N - The number of genres to return.

--

-- Example usage: EXECUTE getTopNGenreBookingsOfAllTime 5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNGenreBookingsOfAllTime' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNGenreBookingsOfAllTime**;**

**GO**

**CREATE** **PROCEDURE** getTopNGenreBookingsOfAllTime @N **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **DISTINCT** TOP **(**@N**)**

GENRE**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByGenre**(**FT**.**GENRE**))** **AS** "TOTAL\_BOOKINGS"

**FROM** filmsTBL "FT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** GENRE **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNScreenBookingsOfAllTime

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N screens which received the highest number of bookings.

-- Parameters: @N - The number of screens to return.

--

-- Example usage: EXECUTE getTopNScreenBookingsOfAllTime 5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNScreenBookingsOfAllTime' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNScreenBookingsOfAllTime**;**

**GO**

**CREATE** **PROCEDURE** getTopNScreenBookingsOfAllTime @N **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **DISTINCT** PT**.**SCREEN\_NO**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByScreenNo**(**PT**.**SCREEN\_NO**))** **AS** "TOTAL\_BOOKINGS"

**from** performanceTBL "PT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** PT**.**SCREEN\_NO **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNFilmBookingsDuringPeriod

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N films which received the highest number

-- of bookings during the specified search period.

-- Parameters: @N - The number of films to return.

-- @MIN\_DATE - The start date of the search period.

-- @MAX\_DATE - The finish date of the search period.

--

-- Example usage: EXECUTE getTopNFilmBookingsDuringPeriod 5, '2018-08-21', '2018-08-23';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNFilmBookingsDuringPeriod' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNFilmBookingsDuringPeriod**;**

**GO**

**CREATE** **PROCEDURE** getTopNFilmBookingsDuringPeriod @N **INT,** @MIN\_DATE **DATE,** @MAX\_DATE **DATE**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** TOP **(**@N**)** FILM\_ID**,**

TITLE**,**

GENRE**,**

**FORMAT(**DURATION**,** N'hh\:mm'**)** **AS** DURATION**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByFilmIDDuringPeriod**(**FT**.**FILM\_ID**,** @MIN\_DATE**,** @MAX\_DATE**))** **AS** "TOTAL\_BOOKINGS"

**FROM** filmsTBL "FT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** TITLE **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNBookingCustomersDuringPeriod

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N customers who made the highest number

-- of bookings during the specified search period.

-- Parameters: @N - The number of customers to return.

-- @MIN\_DATE - The start date of the search period.

-- @MAX\_DATE - The finish date of the search period.

--

-- Example usage: EXECUTE getTopNBookingCustomersDuringPeriod 5, '2018-08-21', '2018-08-23';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNBookingCustomersDuringPeriod' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNBookingCustomersDuringPeriod**;**

**GO**

**CREATE** **PROCEDURE** getTopNBookingCustomersDuringPeriod @N **INT,** @MIN\_DATE **DATE,** @MAX\_DATE **DATE**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** TOP **(**@N**)**

CUSTOMER\_ID**,**

FIRSTNAME**,**

LASTNAME**,**

EMAIL**,**

POSTCODE**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByCustomerIDDuringPeriod**(**CT**.**CUSTOMER\_ID**,**

@MIN\_DATE**,**@MAX\_DATE**))** **AS** "TOTAL\_BOOKINGS"

**FROM** customersTBL "CT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** LASTNAME **ASC,** FIRSTNAME **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNGenreBookingsDuringPeriod

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N genres which received the highest number

-- of bookings during the specified search period.

-- Parameters: @N - The number of genres to return.

-- @MIN\_DATE - The start date of the search period.

-- @MAX\_DATE - The finish date of the search period.

--

-- Example usage: EXECUTE getTopNGenreBookingsDuringPeriod 5, '2018-08-21', '2018-08-23';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNGenreBookingsDuringPeriod' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNGenreBookingsDuringPeriod**;**

**GO**

**CREATE** **PROCEDURE** getTopNGenreBookingsDuringPeriod @N **INT,** @MIN\_DATE **DATE,** @MAX\_DATE **DATE**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **DISTINCT** TOP **(**@N**)**

GENRE**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByGenreDuringPeriod**(**FT**.**GENRE**,** @MIN\_DATE**,** @MAX\_DATE**))** **AS** "TOTAL\_BOOKINGS"

**FROM** filmsTBL "FT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** GENRE **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNScreenBookingsDuringPeriod

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N screens which received the highest number of bookings

-- during the specified search period.

-- Parameters: @N - The number of screens to return.

-- @MIN\_DATE - The start date of the search period.

-- @MAX\_DATE - The finish date of the search period.

--

-- Example usage: EXECUTE getTopNScreenBookingsDuringPeriod 5, '2018-08-21', '2018-08-23';

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNScreenBookingsDuringPeriod' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNScreenBookingsDuringPeriod**;**

**GO**

**CREATE** **PROCEDURE** getTopNScreenBookingsDuringPeriod @N **INT,** @MIN\_DATE **DATE,** @MAX\_DATE **DATE**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**SELECT** **DISTINCT** PT**.**SCREEN\_NO**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByScreenNoDuringPeriod**(**PT**.**SCREEN\_NO**,**

@MIN\_DATE**,** @MAX\_DATE**))** **AS** "TOTAL\_BOOKINGS"

**from** performanceTBL "PT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** PT**.**SCREEN\_NO **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNFilmBookingsDuringPreviousWeek

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N films which received the highest number

-- of bookings during the previous week.

-- Parameters: @N - The number of films to return.

--

-- Example usage: EXECUTE getTopNFilmBookingsDuringPreviousWeek 5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNFilmBookingsDuringPreviousWeek' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNFilmBookingsDuringPreviousWeek**;**

**GO**

**CREATE** **PROCEDURE** getTopNFilmBookingsDuringPreviousWeek @N **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DECLARE** @startDate **DATETIME;**

**DECLARE** @endDate **DATETIME;**

**SET** @endDate **=** **GETDATE();** -- Set end date as today.

**SET** @startDate **=** **DATEADD(DAY,** -7**,** @EndDate**)** -- Set start date as today minus 7 days.

**SELECT** TOP **(**@N**)** FILM\_ID**,**

TITLE**,**

GENRE**,**

**FORMAT(**DURATION**,** N'hh\:mm'**)** **AS** DURATION**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByFilmIDDuringPeriod**(**FT**.**FILM\_ID**,**

@startDate**,** @endDate**))** **AS** "TOTAL\_BOOKINGS"

**FROM** filmsTBL "FT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** TITLE **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNBookingCustomersDuringPreviousWeek

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N customers who made the highest number of bookings

-- during the previous week.

-- Parameters: @N - The number of customers to return.

--

-- Example usage: EXECUTE getTopNBookingCustomersDuringPreviousWeek 5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNBookingCustomersDuringPreviousWeek' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNBookingCustomersDuringPreviousWeek**;**

**GO**

**CREATE** **PROCEDURE** getTopNBookingCustomersDuringPreviousWeek @N **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DECLARE** @startDate **DATETIME;**

**DECLARE** @endDate **DATETIME;**

**SET** @endDate **=** **GETDATE();** -- Set end date as today.

**SET** @startDate **=** **DATEADD(DAY,** -7**,** @EndDate**)** -- Set start date as today minus 7 days.

**SELECT** TOP **(**@N**)**

CUSTOMER\_ID**,**

FIRSTNAME**,**

LASTNAME**,**

EMAIL**,**

POSTCODE**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByCustomerIDDuringPeriod**(**CT**.**CUSTOMER\_ID**,**

@startDate**,** @endDate**))** **AS** "TOTAL\_BOOKINGS"

**FROM** customersTBL "CT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** LASTNAME **ASC,** FIRSTNAME **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNGenreBookingsDuringPreviousWeek

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N genres which received the highest number

-- of bookings during the previous week.

-- Parameters: @N - The number of genres to return.

--

-- Example usage: EXECUTE getTopNGenreBookingsDuringPreviousWeek 5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNGenreBookingsDuringPreviousWeek' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNGenreBookingsDuringPreviousWeek**;**

**GO**

**CREATE** **PROCEDURE** getTopNGenreBookingsDuringPreviousWeek @N **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DECLARE** @startDate **DATETIME;**

**DECLARE** @endDate **DATETIME;**

**SET** @endDate **=** **GETDATE();** -- Set end date as today.

**SET** @startDate **=** **DATEADD(DAY,** -7**,** @EndDate**)** -- Set start date as today minus 7 days.

**SELECT** **DISTINCT** TOP **(**@N**)**

GENRE**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByGenreDuringPeriod**(**FT**.**GENRE**,**

@startDate**,** @endDate**))** **AS** "TOTAL\_BOOKINGS"

**FROM** filmsTBL "FT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** GENRE **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Procedure Name: getTopNScreenBookingsDuringPreviousWeek

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: Selects the top N screens which received the highest number of bookings

-- during the previous week.

-- Parameters: @N - The number of screens to return.

--

-- Example usage: EXECUTE getTopNScreenBookingsDuringPreviousWeek 5;

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getTopNScreenBookingsDuringPreviousWeek' **,**'P'**)** **IS** **NOT** **NULL**

**DROP** **PROCEDURE** getTopNScreenBookingsDuringPreviousWeek**;**

**GO**

**CREATE** **PROCEDURE** getTopNScreenBookingsDuringPreviousWeek @N **INT**

**AS**

**BEGIN**

**SET** NOCOUNT **ON;**

**DECLARE** @startDate **DATETIME;**

**DECLARE** @endDate **DATETIME;**

**SET** @endDate **=** **GETDATE();** -- Set end date as today.

**SET** @startDate **=** **DATEADD(DAY,** -7**,** @EndDate**)** -- Set start date as today minus 7 days.

**SELECT** **DISTINCT** PT**.**SCREEN\_NO**,**

**(SELECT** dbo**.**getTotalNoOfTicketsBookedByScreenNoDuringPeriod**(**PT**.**SCREEN\_NO**,**

@startDate**,** @endDate**))** **AS** "TOTAL\_BOOKINGS"

**FROM** performanceTBL "PT"

**ORDER** **BY** TOTAL\_BOOKINGS **DESC,** SCREEN\_NO **ASC**

**END;**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getAll3DPerformanceIDs

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns all 3D performance IDs stored within   
-- the performanceTBL table.

-- Parameters: None

--

-- Example usage: SELECT \* FROM dbo.getAll3DPerformanceIDs();

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAll3DPerformanceIDs' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getAll3DPerformanceIDs**;**

**GO**

**CREATE** **FUNCTION** getAll3DPerformanceIDs **(** **)**

**RETURNS** **TABLE**

**AS**

**RETURN**

**(**

**SELECT** **DISTINCT** PERFORMANCE\_ID **from** performanceTBL

**WHERE** IN\_3D**=**1

**);**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getAllBookingIDs

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns all booking IDs stored within bookingsTBL.

-- Parameters: None

--

-- Example usage: SELECT \* FROM getAllBookingIDs();

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAllBookingIDs' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getAllBookingIDs**;**

**GO**

**CREATE** **FUNCTION** getAllBookingIDs **(** **)**

**RETURNS** **TABLE**

**AS**

**RETURN**

**(**

**SELECT** **DISTINCT** BOOKING\_ID **from** bookingsTBL

**);**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getAllCustomerIDs

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns all customer IDs stored within customersTBL.

-- Parameters: None

--

-- Example usage: SELECT \* FROM dbo.getAllCustomerIDs();

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAllCustomerIDs' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getAllCustomerIDs**;**

**GO**

**CREATE** **FUNCTION** getAllCustomerIDs **(** **)**

**RETURNS** **TABLE**

**AS**

**RETURN**

**(**

**SELECT** **DISTINCT** CUSTOMER\_ID **from** customersTBL

**);**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getAllActorNames

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns actor names within filmsTBL.

-- Parameters: None

--

-- Example usage: SELECT \* FROM dbo.getAllActorNames();

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAllActorNames' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getAllActorNames**;**

**GO**

**CREATE** **FUNCTION** getAllActorNames **(** **)**

**RETURNS** **TABLE**

**AS**

**RETURN**

**(**

**SELECT** **DISTINCT** ACTOR\_FORENAME**,** ACTOR\_SURNAME **from** filmsTBL

**);**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getAllActressNames

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns actress names within filmsTBL.

-- Parameters: None

--

-- Example usage: SELECT \* FROM dbo.getAllActressNames();

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAllActressNames' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getAllActressNames**;**

**GO**

**CREATE** **FUNCTION** getAllActressNames **(** **)**

**RETURNS** **TABLE**

**AS**

**RETURN**

**(**

**SELECT** **DISTINCT** ACTRESS\_FORENAME**,** ACTRESS\_SURNAME **from** filmsTBL

**);**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getAllDirectorNames

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns directors within filmsTBL.

-- Parameters: None

--

-- Example usage: SELECT \* FROM dbo.getAllDirectorNames();

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAllDirectorNames' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getAllDirectorNames**;**

**GO**

**CREATE** **FUNCTION** getAllDirectorNames **(** **)**

**RETURNS** **TABLE**

**AS**

**RETURN**

**(**

**SELECT** **DISTINCT** DIRECTOR\_FORENAME**,** DIRECTOR\_SURNAME **from** filmsTBL

**);**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getAllFilmGenres

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns all genres used within filmsTBL.

-- Parameters: None

--

-- Example usage: SELECT \* FROM dbo.getAllFilmGenres();

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAllFilmGenres' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getAllFilmGenres**;**

**GO**

**CREATE** **FUNCTION** getAllFilmGenres **(** **)**

**RETURNS** **TABLE**

**AS**

**RETURN**

**(**

**SELECT** **DISTINCT** GENRE **from** filmsTBL

**);**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getAllFilmIDs

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns all film IDs stored within filmsTBL.

-- Parameters: None

--

-- Example usage: SELECT \* FROM dbo.getAllFilmIDs();

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAllFilmIDs' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getAllFilmIDs**;**

**GO**

**CREATE** **FUNCTION** getAllFilmIDs **(** **)**

**RETURNS** **TABLE**

**AS**

**RETURN**

**(**

**SELECT** **DISTINCT** FILM\_ID **from** filmsTBL

**);**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getAllFilmTitles

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns all film titles stored within filmsTBL.

-- Parameters: None

--

-- Example usage: SELECT \* FROM dbo.getAllFilmTitles();

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAllFilmTitles' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getAllFilmTitles**;**

**GO**

**CREATE** **FUNCTION** getAllFilmTitles **(** **)**

**RETURNS** **TABLE**

**AS**

**RETURN**

**(**

**SELECT** **DISTINCT** TITLE **from** filmsTBL

**);**

**GO**

-- ----------------------------------------------------------------------------

-- Function Name: getAllPerformanceIDs

-- Author: Andrew Laing

-- Last updated: 03/01/2019

-- Description: This function returns all performance IDs stored within performanceTBL.

-- Parameters: None

--

-- Example usage: SELECT \* FROM dbo.getAllPerformanceIDs();

-- ----------------------------------------------------------------------------

**IF** **OBJECT\_ID(**'getAllPerformanceIDs' **,**'FN'**)** **IS** **NOT** **NULL**

**DROP** **FUNCTION** getAllPerformanceIDs**;**

**GO**

**CREATE** **FUNCTION** getAllPerformanceIDs **(** **)**

**RETURNS** **TABLE**

**AS**

**RETURN**

**(**

**SELECT** **DISTINCT** PERFORMANCE\_ID **from** performanceTBL

**);**

**GO**

PRINT 'Database creation finished.'

**SET** NOEXEC **OFF** -- Reset NOEXEC if the script was stopped because the DB already exists.

# Test Data Creation Script

-- Work with the filmPerformanceDB database.  
**use** filmPerformanceDB**;**

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- @ Set up timing variables and temporary performance times table @

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- Create a table variable to hold results to hold the screening times.

PRINT 'Create the #times table.'

IF OBJECT\_ID('#times' ,'U') IS NOT NULL

DROP TABLE #times;

GO

CREATE TABLE #times

(Id INT, timeOfPerformance TIME)

GO

INSERT INTO #times ( Id, timeOfPerformance)

Values (1, '12:00'),

(2,'14:00'),

(3,'16:00'),

(4,'18:00'),

(5,'20:00'),

(6,'22:00')

-- @CURRENT\_TIME is used for timing the running of the script.

DECLARE @CURRENT\_TIME VARCHAR(8)

SET @CURRENT\_TIME = ( SELECT CONVERT(VARCHAR(8), GETDATE(), 108) 'hh:mi:ss' );

PRINT 'Record creation started at ' + @CURRENT\_TIME

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
-- @ Add some records to the screens table @  
-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

PRINT 'Adding records to screenTBL'**;**

**INSERT** **INTO** screenTBL

**(** CAPACITY**,**

HAS\_DISABLED\_ACCESS**,**

HAS\_HEARING\_LOOP **)**

**VALUES** **(** 150**,** 1**,** 1 **),**

**(** 100**,** 1**,** 1 **),**

**(** 150**,** 0**,** 0 **),**

**(** 250**,** 1**,** 0 **),**

**(** 200**,** 0**,** 0 **),**

**(** 50**,** 1**,** 1 **),**

**(** 200**,** 1**,** 0 **),**

**(** 100**,** 0**,** 1 **),**

**(** 150**,** 1**,** 0 **),**

**(** 100**,** 1**,** 1 **);**

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- @ Add some records to the films table @

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

PRINT 'Adding records to filmsTBL'**;**

**INSERT** **INTO** filmsTBL

**(** TITLE**,** GENRE**,** DURATION**,** ACTOR\_SURNAME**,**

ACTOR\_FORENAME**,** ACTRESS\_SURNAME**,** ACTRESS\_FORENAME**,**

DIRECTOR\_SURNAME**,** DIRECTOR\_FORENAME **)**

**VALUES** **(** 'La Dolce Vita'**,** 'Drama'**,** '03:12:22'**,** 'Mastroianni'**,**

'Marcello'**,** 'Aimee'**,** 'Anouk'**,** 'Fellini'**,** 'Federico' **),**

**(** 'Una Journata Particolare'**,** 'Drama'**,** '02:45:00'**,**

'Mastroianni'**,** 'Marcello'**,** 'Loren'**,** 'Sophia'**,** 'Scola'**,**

'Ettore' **),**

**(** 'La Vie est Belle'**,** 'Drama'**,** '02:32:32'**,** 'Benigni'**,**

'Roberto'**,** 'Braschi'**,** 'Nicoletta'**,** 'Benigni'**,** 'Roberto'**),**

**(** 'La Haine'**,** 'Drama'**,** '02:15:12'**,** 'Cassel'**,** 'Vincent'**,**

'Rauth'**,** 'Heloise'**,** 'Kassovitz'**,** 'Mathieu'**),**

**(** 'La Belle et La Bete'**,** 'Romance'**,** '01:54:22'**,** 'Marais'**,**

'Jean'**,** 'Day'**,** 'Josette'**,** 'Cocteau'**,** 'Jean'**),**

**(** 'Heat'**,** 'Police Drama'**,** '03:02:11'**,** 'Pacino'**,** 'Al'**,**

'Venora'**,** 'Diane'**,** 'Mann'**,** 'Michael' **),**

**(** 'From Russia with Love'**,** 'Drama'**,** '02:34:11'**,** 'Connery'**,**

'Sean'**,** 'Lenya'**,** 'Lotte'**,** 'Young'**,** 'Terence' **),**

**(** 'Educating Rita'**,** 'Comedy'**,** '02:55:00'**,** 'Caine'**,**

'Michael'**,** 'Walters'**,** 'Julie'**,** 'Gilbert'**,** 'Lewis' **),**

**(** 'His Girl Friday'**,** 'Comedy'**,** '01:34:11'**,** 'Grant'**,** 'Cary'**,**

'Russell'**,** 'Rosalind'**,** 'Hawks'**,** 'Howard' **),**

**(** 'L Eclisse'**,** 'Comedy'**,** '03:44:22'**,** 'Delon'**,** 'Alain'**,**

'Vitti'**,** 'Monica'**,** 'Antonioni'**,** 'Michaelangelo' **),**

**(** 'Lovely Donuts'**,** 'Comedy'**,** '02:12:26'**,** 'Keaton'**,** 'Bobby'**,**

'Gish'**,** 'Cheryl'**,** 'Hitchcock'**,** 'John' **),**

**(** 'adsfghdsad'**,** 'dsadadad'**,** '02:12:26'**,** 'qwqweq'**,** 'qwewqe'**,**

'qweqwew'**,** 'qeqe'**,** 'qeqwe'**,** 'qweqwe'**),**

**(** 'Ruthless People'**,**'Comedy'**,**'01:30:12'**,**'DeVito'**,**'Danny'**,**

'Slater'**,**'Helen'**,**'Abrahams'**,**'Jim' **),**

**(** 'The Count of Old Town'**,**'Comedy'**,**'01:25:00'**,**'Adolphson'**,**

'Edvin'**,**'Bergman'**,**'Ingrid'**,**'Adolphson'**,**'Edvin' **),**

**(** 'Sweet Liberty'**,**'Comedy'**,**'01:47:02'**,**'Alda'**,**'Alan'**,**

'Pfeiffer'**,**'Michelle'**,**'Alda'**,**'Alan' **),**

**(** 'Hush Hush'**,**'Mystery'**,**'02:13:19'**,**'Cotten'**,**'Joseph'**,**

'Davis'**,**'Bette'**,**'Aldrich'**,**'Robert' **),**

**(** 'Mademoiselle Striptease'**,**'Comedy'**,**'01:39:22'**,**'Gelin'**,**

'Daniel'**,**'Bardot'**,**'Brigitte'**,**'Allegret'**,**'Marc' **),**

**(** 'Alice'**,**'Comedy'**,**'01:40:00'**,**'Baldwin'**,**'Alec'**,**'Danner'**,**

'Blythe'**,**'Allen'**,**'Woody' **),**

**(** 'Broadway Danny Rose'**,**'Comedy'**,**'01:25:44'**,**'Allen'**,**'Woody'**,**

'Farrow'**,**'Mia'**,**'Allen'**,**'Woody' **),**

**(** 'Husbands & Wives'**,**'Comedy'**,**'01:48:33'**,**'Allen'**,**'Woody'**,**

'Farrow'**,**'Mia'**,**'Allen'**,**'Woody' **),**

**(** 'The Purple Rose of Cairo'**,**'Comedy'**,**'01:24:37'**,**'Aiello'**,**

'Danny'**,**'Farrow'**,**'Mia'**,**'Allen'**,**'Woody' **),**

**(** 'Hannah & Her Sisters'**,**'Comedy'**,**'01:43:44'**,**'Caine'**,**

'Michael'**,**'Farrow'**,**'Mia'**,**'Allen'**,**'Woody' **),**

**(** 'Annie Hall'**,**'Comedy'**,**'01:34:11'**,**'Allen'**,**'Woody'**,**'Keaton'**,**

'Diane'**,**'Allen'**,**'Woody' **),**

**(** 'Manhattan'**,**'Comedy'**,**'01:36:16'**,**'Allen'**,**'Woody'**,**'Keaton'**,**

'Diane'**,**'Allen'**,**'Woody' **),**

**(** 'Another Woman'**,**'Drama'**,**'01:21:52'**,**'Hackman'**,**'Gene'**,**

'Rowlands'**,**'Gena'**,**'Allen'**,**'Woody' **),**

**(** 'Matador'**,**'Comedy'**,**'02:35:25'**,**'Banderas'**,**'Antonio'**,**'Maura'**,**

'Carmen'**,**'Almodovar'**,**'Pedro' **),**

**(** 'Women on the Verge of a Nervous Breakdown'**,**'Comedy'**,**'01:28:28'**,**

'Banderas'**,**'Antonio'**,**'Maura'**,**'Carmen'**,**'Almodovar'**,**'Pedro' **),**

**(** 'Pepi Luci Bom'**,**'Comedy'**,**'01:26:55'**,**'Rotaeta'**,**'Felix'**,**'Maura'**,**

'Carmen'**,**'Almodovar'**,**'Pedro' **),**

**(** 'Tie Me Up! Tie Me Down!'**,**'Comedy'**,**'01:51:00'**,**'Banderas'**,**

'Antonio'**,**'Abril'**,**'Victoria'**,**'Almodovar'**,**'Pedro' **),**

**(** 'High Heels'**,**'Comedy'**,**'01:53:15'**,**'Bose'**,**'Miguel'**,**'Abril'**,**

'Victoria'**,**'Almodovar'**,**'Pedro' **),**

**(** 'O. C. & Stiggs'**,**'Comedy'**,**'01:49:55'**,**'Jenkins'**,**'Daniel H.'**,**

'Curtin'**,**'Jane'**,**'Altman'**,**'Robert' **),**

**(** 'Quintet'**,**'Drama'**,**'01:50:43'**,**'Newman'**,**'Paul'**,**'Andersson'**,**

'Bibi'**,**'Altman'**,**'Robert' **),**

**(** 'The Statue'**,**'Drama'**,**'01:24:58'**,**'Niven'**,**'David'**,**'Lisi'**,**

'Virna'**,**'Amateau'**,**'Rod' **),**

**(** 'Flashback'**,**'Comedy'**,**'01:48:00'**,**'Hopper'**,**'Dennis'**,**'Kane'**,**

'Carol'**,**'Amurri'**,**'Franco' **),**

**(** 'Inauguration of the Pleasure Dome'**,**'Short'**,**'00:40:12'**,**

'De Brier'**,**'Sampson'**,**'Cameron'**,**'Marjorie'**,**'Anger'**,**'Kenneth' **),**

**(** 'Black & White in Color'**,**'Comedy'**,**'01:30:22'**,**'Carmet'**,**'Jean'**,**

'Rouvel'**,**'Catherine'**,**'Annaud'**,**'Jean-Jacques' **),**

**(** 'Hoosiers'**,**'Drama'**,**'01:44:21'**,**'Hackman'**,**'Gene'**,**'Hershey'**,**

'Barbara'**,**'Anspaugh'**,**'David' **),**

**(** 'The Matchmaker'**,**'Comedy'**,**'01:41:23'**,**'Perkins'**,**'Anthony'**,**

'Booth'**,**'Shirley'**,**'Anthony'**,**'Joseph' **),**

**(** 'The Rainmaker'**,**'Drama'**,**'02:01:22'**,**'Lancaster'**,**'Burt'**,**

'Hepburn'**,**'Katharine'**,**'Anthony'**,**'Joseph' **),**

**(** 'Between Friends'**,**'Drama'**,**'01:40:21'**,**'Ramer'**,**'Henry'**,**

'Taylor'**,**'Elizabeth'**,**'Antonio'**,**'Lou' **),**

**(** 'The Passenger'**,**'Drama'**,**'01:58:44'**,**'Nicholson'**,**'Jack'**,**

'Schneider'**,**'Maria'**,**'Antonioni'**,**'Michelangelo' **),**

**(** 'The Squeeze'**,**'Action'**,**'01:46:22'**,**'Keach'**,**'Stacy'**,**

'White'**,**'Carol'**,**'Apted'**,**'Michael' **),**

**(** 'Critical Condition'**,**'Comedy'**,**'01:39:00'**,**'Pryor'**,**'Richard'**,**

'Ticotin'**,**'Rachel'**,**'Apted'**,**'Michael' **),**

**(** 'Caddyshack 2'**,**'Comedy'**,**'01:38:39'**,**'Mason'**,**'Jackie'**,**

'Merrill'**,**'Dina'**,**'Arkush'**,**'Allan' **),**

**(** 'Shampoo'**,**'Comedy'**,**'01:49:44'**,**'Beatty'**,**'Warren'**,**'Christie'**,**

'Julie'**,**'Ashby'**,**'Hal' **),**

**(** 'Being There'**,**'Comedy'**,**'02:09:47'**,**'Sellers'**,**'Peter'**,**

'MacLaine'**,**'Shirley'**,**'Ashby'**,**'Hal' **),**

**(** 'Gandhi'**,**'Drama'**,**'02:06:33'**,**'Kingsley'**,**'Ben'**,**'Bergen'**,**

'Candice'**,**'Attenborough'**,**'Richard' **),**

**(** 'A Chorus Line - The Movie'**,**'Musical'**,**'01:58:22'**,**'Douglas'**,**

'Michael'**,**'Landers'**,**'Audrey'**,**'Attenborough'**,**'Richard' **),**

**(** 'Wheel of Fortune'**,**'Drama'**,**'01:25:52'**,**'Wayne'**,**'John'**,**

'Dee'**,**'Frances'**,**'Auer'**,**'John H.' **),**

**(** 'The Women'**,**'Drama'**,**'01:26:19'**,**'Ronet'**,**'Maurice'**,**'Bardot'**,**

'Brigitte'**,**'Aurel'**,**'Jean' **),**

**(** 'Fifth Floor - The'**,**'Mystery'**,**'01:30:01'**,**'Hopkins'**,**'Bo'**,**

'DArbanville'**,**'Patti'**,**'Avedis'**,**'Howard Hikmet' **),**

**(** 'Lean on Me'**,**'Drama'**,**'01:49:21'**,**'Freeman'**,**'Morgan'**,**

'Todd'**,**'Beverly'**,**'Avildsen'**,**'John G.' **),**

**(** 'Rocky'**,**'Drama'**,**'01:30:00'**,**'Stallone'**,**'Sylvester'**,**

'Shire'**,**'Talia'**,**'Avildsen'**,**'John G.' **),**

**(** 'The Formula'**,**'Mystery'**,**'01:47:12'**,**'Scott'**,**'George C.'**,**

'Keller'**,**'Marthe'**,**'Avildsen'**,**'John G.' **),**

**(** 'Between Two Women'**,**'Drama'**,**'01:35:23'**,**'Nouri'**,**

'Michael'**,**'Fawcett'**,**'Farrah'**,**'Avnet'**,**'John' **),**

**(** 'Babettes Feast'**,**'Drama'**,**'01:42:23'**,**'LaFont'**,**

'Jean-Philippe'**,**'Audran'**,**'Stephane'**,**'Axel'**,**'Gabriel' **),**

**(** 'Nothing But Trouble'**,**'Comedy'**,**'01:34:17'**,**'Candy'**,**

'John'**,**'Moore'**,**'Demi'**,**'Aykroyd'**,**'Dan' **),**

**(** 'Ironweed'**,**'Drama'**,**'02:15:45'**,**'Nicholson'**,**'Jack'**,**

'Streep'**,**'Meryl'**,**'Babenco'**,**'Hector' **),**

**(** 'At Play in the Fields of the Lord'**,**'Drama'**,**'02:59:24'**,**

'Berenger'**,**'Tom'**,**'Hannah'**,**'Daryl'**,**'Babenco'**,**'Hector' **),**

**(** 'Kiss of the Spider Woman'**,**'Drama'**,**'02:59:25'**,**'Hurt'**,**

'William'**,**'Braga'**,**'Sonia'**,**'Babenco'**,**'Hector' **),**

**(** 'Stakeout'**,**'Comedy'**,**'01:55:12'**,**'Dreyfuss'**,**'Richard'**,**

'Stowe'**,**'Madeleine'**,**'Badham'**,**'John' **),**

**(** 'Born to Ride'**,**'Action'**,**'01:28:24'**,**'Stamos'**,**'John'**,**

'Polo'**,**'Teri'**,**'Baker'**,**'Graham' **),**

**(** 'Cool World'**,**'Drama'**,**'01:40:33'**,**'Byrne'**,**'Gabriel'**,**

'Basinger'**,**'Kim'**,**'Bakshi'**,**'Ralph' **),**

**(** 'Meridian'**,**'Science Fiction'**,**'01:43:23'**,**'Jamieson'**,**

'Malcolm'**,**'Fenn'**,**'Sherilyn'**,**'Band'**,**'Charles' **),**

**(** 'Crash & Burn'**,**'Science Fiction'**,**'01:25:43'**,**'Ganus'**,**

'Paul'**,**'Ward'**,**'Megan'**,**'Band'**,**'Charles' **),**

**(** 'Unsettled Land'**,**'Drama'**,**'01:49:52'**,**'Shea'**,**'John'**,**

'McGillis'**,**'Kelly'**,**'Barbash'**,**'Uri' **),**

**(** 'Nightbreed'**,**'Horror'**,**'01:42:33'**,**'Cronenberg'**,**'David'**,**

'Bobby'**,**'Anne'**,**'Barker'**,**'Clive' **),**

**(** 'Nights in White Satin'**,**'Drama'**,**'01:36:33'**,**'Gilman'**,**

'Kenneth'**,**'Harris'**,**'Priscilla'**,**'Barnard'**,**'Michael' **),**

**(** 'A Show of Force'**,**'Drama'**,**'01:33:35'**,**'Garcia'**,**

'Andy'**,**'Irving'**,**'Amy'**,**'Barreto'**,**'Bruno' **),**

**(** 'Business As Usual'**,**'Comedy'**,**'01:29:56'**,**'Thaw'**,**

'John'**,**'Jackson'**,**'Glenda'**,**'Barrett'**,**'Lezli-An' **),**

**(** 'The Second Coming of Suzanne'**,**'Drama'**,**'01:30:00'**,**

'Dreyfuss'**,**'Richard'**,**'Locke'**,**'Sondra'**,**'Barry'**,**'Michael' **),**

**(** 'Mr Love'**,**'Comedy'**,**'01:31:21'**,**'Jackson'**,**'Barry'**,**

'Tyzack'**,**'Margaret'**,**'Battersby'**,**'Roy' **),**

**(** 'Dick Tracy'**,**'Comedy'**,**'01:49:33'**,**'Beatty'**,**'Warren'**,**

'Headley'**,**'Glenne'**,**'Beatty'**,**'Warren' **),**

**(** 'Reds'**,**'Drama'**,**'02:59:27'**,**'Beatty'**,**'Warren'**,**'Keaton'**,**

'Diane'**,**'Beatty'**,**'Warren' **),**

**(** 'The Ape Man'**,**'Horror'**,**'01:04:00'**,**'Ford'**,**'Wallace'**,**

'Lugosi'**,**'Bela'**,**'Beaudine'**,**'William' **),**

**(** 'The Moon in the Gutter'**,**'Action'**,**'02:12:22'**,**'Depardieu'**,**

'Gerard'**,**'Kinski'**,**'Nastassia'**,**'Beineix'**,**'Jean-Jacques' **),**

**(** 'Betty Blue'**,**'Drama'**,**'02:00:00'**,**'Anglade'**,**'Jean-Hughes'**,**

'Dalle'**,**'Beatrice'**,**'Beineix'**,**'Jean-Jacques' **),**

**(** 'Surrender'**,**'Comedy'**,**'01:35:34'**,**'Caine'**,**'Michael'**,**

'Field'**,**'Sally'**,**'Belson'**,**'Jerry' **),**

**(** 'The Wild One'**,**'Drama'**,**'01:19:44'**,**'Brando'**,**'Marlon'**,**

'OMalley'**,**'Pat'**,**'Benedek'**,**'Laslo' **),**

**(** 'Racing with the Moon'**,**'Drama'**,**'01:49:44'**,**'Penn'**,**'Sean'**,**

'McGovern'**,**'Elizabeth'**,**'Benjamin'**,**'Richard' **),**

**(** 'Young Love Lemon Popsicle Seven'**,**'Comedy'**,**'01:31:15'**,**

'Katzur'**,**'Yftach'**,**'Noy'**,**'Zachi'**,**'Bennett'**,**'Walter' **),**

**(** 'Nadine'**,**'Comedy'**,**'01:23:45'**,**'Bridges'**,**'Jeff'**,**'Basinger'**,**

'Kim'**,**'Benton'**,**'Robert' **),**

**(** 'Places in the Heart'**,**'Drama'**,**'01:52:33'**,**'Harris'**,**'Ed'**,**

'Field'**,**'Sally'**,**'Benton'**,**'Robert' **),**

**(** 'Kramer vs. Kramer'**,**'Drama'**,**'01:45:23'**,**'Hoffman'**,**

'Dustin'**,**'Streep'**,**'Meryl'**,**'Benton'**,**'Robert' **),**

**(** 'Still of the Night'**,**'Mystery'**,**'02:12:55'**,**'Scheider'**,**

'Roy'**,**'Streep'**,**'Meryl'**,**'Benton'**,**'Robert' **),**

**(** 'Crimes of the Heart'**,**'Comedy'**,**'01:45:08'**,**'Shepard'**,**

'Sam'**,**'Keaton'**,**'Diane'**,**'Beresford'**,**'Bruce' **),**

**(** 'Her Alibi'**,**'Comedy'**,**'01:34:08'**,**'Selleck'**,**'Tom'**,**

'Porizkova'**,**'Paulina'**,**'Beresford'**,**'Bruce' **),**

**(** 'Tender Mercies'**,**'Drama'**,**'01:33:45'**,**'Duvall'**,**'Robert'**,**

'Harper'**,**'Tess'**,**'Beresford'**,**'Bruce' **),**

**(** 'Driving Miss Daisy'**,**'Drama'**,**'01:39:29'**,**'Freeman'**,**

'Morgan'**,**'Tandy'**,**'Jessica'**,**'Beresford'**,**'Bruce' **),**

**(** 'So Fine'**,**'Comedy'**,**'01:31:22'**,**'ONeal'**,**'Ryan'**,**'Melato'**,**

'Mariangela'**,**'Bergman'**,**'Andrew' **),**

**(** 'Honeymoon in Vegas'**,**'Comedy'**,**'01:35:26'**,**'Caan'**,**

'James'**,**'Parker'**,**'Sarah Jessica'**,**'Bergman'**,**'Andrew' **),**

**(** 'Secrets of Women'**,**'Comedy'**,**'01:47:48'**,**'Malmsten'**,**

'Birger'**,**'Björk'**,**'Anita'**,**'Bergman'**,**'Ingmar' **),**

**(** 'A Lesson in Love'**,**'Comedy'**,**'01:36:33'**,**'Björnstrand'**,**

'Gunnar'**,**'Dahlbeck'**,**'Eva'**,**'Bergman'**,**'Ingmar' **),**

**(** 'Smiles of a Summer Night'**,**'Comedy'**,**'01:48:45'**,**

'Björnstrand'**,**'Gunnar'**,**'Jacobsson'**,**'Ulla'**,**'Bergman'**,**'Ingmar' **),**

**(** 'Crisis'**,**'Drama'**,**'01:33:51'**,**'Andersson'**,**'Wiktor'**,**'Baude'**,**

'Anna-Lisa'**,**'Bergman'**,**'Ingmar' **),**

**(** 'From the Life of the Marionettes'**,**'Drama'**,**'01:44:42'**,**

'Atzorn'**,**'Robert'**,**'Buchegger'**,**'Christine'**,**'Bergman'**,**'Ingmar' **),**

**(** 'Brink of Life'**,**'Drama'**,**'01:22:02'**,**'Josephson'**,**'Erland'**,**

'Dahlbeck'**,**'Eva'**,**'Bergman'**,**'Ingmar' **),**

**(** 'Three Strange Loves'**,**'Drama'**,**'01:24:02'**,**'Malmsten'**,**

'Birger'**,**'Henning'**,**'Eva'**,**'Bergman'**,**'Ingmar' **),**

**(** 'Port of Call'**,**'Drama'**,**'01:40:46'**,**'Eklund'**,**'Bengt'**,**

'Jönsson'**,**'Nine-Christine'**,**'Bergman'**,**'Ingmar' **),**

**(** 'To Joy'**,**'Drama'**,**'01:33:55'**,**'Olin'**,**'Stig'**,**'Nilsson'**,**

'Maj-Britt'**,**'Bergman'**,**'Ingmar' **),**

**(** 'After the Rehearsal'**,**'Drama'**,**'01:12:09'**,**'Josephson'**,**

'Erland'**,**'Olin'**,**'Lena'**,**'Bergman'**,**'Ingmar' **),**

**(** 'The Devils Wanton'**,**'Drama'**,**'01:18:16'**,**'Malmsten'**,**

'Birger'**,**'Svedlund'**,**'Doris'**,**'Bergman'**,**'Ingmar' **),**

**(** 'Winter Light'**,**'Drama'**,**'01:20:22'**,**'Björnstrand'**,**

'Gunnar'**,**'Thulin'**,**'Ingrid'**,**'Bergman'**,**'Ingmar' **),**

**(** 'The Silence'**,**'Drama'**,**'01:35:36'**,**'Malmsten'**,**'Birger'**,**

'Thulin'**,**'Ingrid'**,**'Bergman'**,**'Ingmar' **),**

**(** 'The Magician'**,**'Drama'**,**'01:40:22'**,**'Sydow'**,**'Max von'**,**

'Thulin'**,**'Ingrid'**,**'Bergman'**,**'Ingmar' **),**

**(** 'Hour of the Wolf'**,**'Drama'**,**'01:28:54'**,**'Sydow'**,**

'Max von'**,**'Ullman'**,**'Liv'**,**'Bergman'**,**'Ingmar' **),**

**(** 'The Passion of Anna'**,**'Drama'**,**'01:41:11'**,**'Sydow'**,**

'Max von'**,**'Ullman'**,**'Liv'**,**'Bergman'**,**'Ingmar' **),**

**(** 'Night Is My Future'**,**'Drama'**,**'01:27:27'**,**'Malmsten'**,**

'Birger'**,**'Zetterling'**,**'Mai'**,**'Bergman'**,**'Ingmar' **),**

**(** 'The Devils Eye'**,**'Drama'**,**'01:30:09'**,**'Kulle'**,**'Jarl'**,**

'Andersson'**,**'Bibi'**,**'Bergman'**,**'Ingmar' **)**

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- @ Add some records to the customers table @

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

PRINT 'Adding records to customersTBL'**;**

**INSERT** **INTO** customersTBL

**(** FIRSTNAME**,**

LASTNAME**,**

ADDRESSLINE1**,**

ADDRESSLINE2**,**

POSTCODE**,**

EMAIL**,**

PHONE\_NO **)**

**VALUES** **(** 'Randolph'**,** 'Scott'**,** '13 Pleasant Street'**,** 'Liverpool'**,**

'L19 0NE'**,** 'randyscott@gmail.com'**,** '0151345987' **),**

**(** 'Judy'**,** 'Garland'**,** '22 Harlington Cresent'**,** 'Liverpool'**,**

'L24 2ED'**,** 'jgarland@gmail.com'**,** '01514448383' **),**

**(** 'Romeo'**,** 'Echowhisky'**,** '16b Tarwood Terrace'**,** 'Liverpool'**,**

'L1 2LF'**,** 'recho@yahoo.com'**,** '089414567854' **),**

**(** 'Sally'**,** 'Biggins'**,** '22 Plikington Place'**,** 'Liverpool'**,**

'L22 4ES'**,** 'sbiggins@gmail.com'**,** '08982241161' **),**

**(** 'James'**,** 'Garner'**,** '94 Bold Street'**,** 'Liverpool'**,**

'L14 3DD'**,** 'jgarner123@gmail.com'**,** '01517775656' **),**

**(** 'Christopher'**,** 'Scott'**,** '6 Alexandra Mews'**,** 'Liverpool'**,**

'L9 0ND'**,** 'chrissyscott@gmail.com'**,** '015109987784' **),**

**(** 'Percival'**,** 'Morrisson'**,** '21 Everly Close'**,** 'Liverpool'**,**

'L11 4FQ'**,** 'pmoggyio@gmail.com'**,** '08982215663' **),**

**(** 'Gigi'**,** 'Delacourt'**,** '96 Evergreen Terrace'**,** 'Liverpool'**,**

'L22 6TR'**,** 'gggirlie@gmail.com'**,** '01515556343' **),**

**(** 'Lloyd'**,** 'Ferdinand'**,** '124 Mainbridge Street'**,** 'Liverpool'**,**

'L2 4FD'**,** 'ferdielar@gmail.com'**,** '08942278876' **),**

**(** 'Mark'**,** 'Chapman'**,** '12 Leepworth Hill'**,** 'Liverpool'**,**

'L16 2WW'**,** 'chappio@gmail.com' **,** '01518989777' **),**

**(** 'Paul'**,** 'Henderson'**,** '65 Impressa Grove'**,** 'Liverpool'**,**

'L2 4EE'**,** 'hennyla@gmail.com'**,** '08983267769' **),**

**(** 'Ffff'**,** 'Pwwww'**,** 'uug'**,** 'ewe'**,** 'wwwwwwww'**,**

'wwwwwww'**,** '09898765455' **),**

**(** 'Leonard'**,**'Sherrod'**,**'4 Woodland Rd'**,**'Liverpool'**,**

'L26 1XF'**,**'LeonardCSherrod@dayrep.com'**,**'07084209786' **),**

**(** 'Lilian'**,**'Brodhurst'**,**'2 Partridge Road'**,**'Liverpool'**,**

'L17 2TP'**,**'lbrodhurst@yahoo.com'**,**'07865123665' **),**

**(** 'Rachel'**,**'Hayes'**,**'61 North Manor Way'**,**'Liverpool'**,**

'L25 9NN'**,**'RachelHayes@teleworm.us'**,**'07851031883' **),**

**(** 'Luca'**,**'McKenzie'**,**'48 July Rd'**,**'Liverpool'**,**'L6 4BT'**,**

'LucaMcKenzie@dayrep.com'**,**'07879393786' **),**

**(** 'Oscar'**,**'Walters'**,**'12 Rosemary Lane'**,**'Ormskirk'**,**

'L39 7JP'**,**'wallyboy@gmail.com'**,**'07737825773' **),**

**(** 'Lewis'**,**'Chapman'**,**'72 Fairclough St'**,**'Liverpool'**,**

'L1 1JX'**,**'chappiechap@yahoo.com'**,**'07821786251' **),**

**(** 'Aidan'**,**'Evans'**,**'80 Mariners Wharf'**,**'Liverpool'**,**

'L3 4DH'**,**'AidanEvans@sporty.com'**,**'07057791379' **),**

**(** 'Faith'**,**'Jennings'**,**'9 Barrymore Rd'**,**'Liverpool'**,**

'L13 3BA'**,**'faithJ123@yahoo.com'**,**'07815084870' **),**

**(** 'Amelie'**,**'Wall'**,**'55 North Barcombe Rd'**,**'Liverpool'**,**

'L16 7PU'**,**'amelialol@gmail.com'**,**'07838703302' **),**

**(** 'Taylor'**,**'Holmes'**,**'59 Landford Place'**,**'Liverpool'**,**

'L9 6BT'**,**'TaylorHolmes@jourrapide.com'**,**'07088786109' **),**

**(** 'Katie'**,**'Myers'**,**'111 Kings Dock St'**,**'Liverpool'**,**

'L3 4EU'**,**'KatieMyers@freelancer.com'**,**'07978901547' **),**

**(** 'Ella'**,**'Bryant'**,**'74 Drewell Rd'**,**'Liverpool'**,**'L18 7TP'**,**

'EllaBryant@armyspy.com'**,**'07066289644' **),**

**(** 'Beverly'**,**'Nelson'**,**'23 Halkyn Avenue'**,**'Liverpool'**,**

'L17 2AH'**,**'bnelson307@nhs.uk.co'**,**'07660630642' **),**

**(** 'Belinda'**,**'Carlisle'**,**'34 Stapleton Crescent'**,**'Liverpool'**,**

'L18 2PR'**,**'bcarlisle311@nhs.co.uk'**,**'07081162323' **),**

**(** 'Keith'**,**'Alexander'**,**'78 Seel St'**,**'Liverpool'**,**'L1 4BH'**,**

'kalexander496@nhs.uk.co'**,**'015112496' **),**

**(** 'Kathleen'**,**'Davis'**,**'82 Wood St'**,**'Liverpool'**,**'L1 4DQ'**,**

'kdavis345@nhs.uk.co'**,**'07510879572' **),**

**(** 'Linda'**,**'Stewart'**,**'4 Queenscourt Road'**,**'Liverpool'**,**

'L12 8RH'**,**'lstewart226@nhs.co.uk'**,**'07380024485' **),**

**(** 'Kelly'**,**'Russell'**,**'16 Tullimore Rd'**,**'Liverpool'**,**

'L18 4PR'**,**'krussell797@nhs.co.uk'**,**'01510771734' **),**

**(** 'Patrick'**,**'Williams'**,**'23 Robertson St'**,**'Liverpool'**,**

'L8 6UG'**,**'pwilliams823@nhs.co.uk'**,**'07494737290' **),**

**(** 'Nancy'**,**'Rodriguez'**,**'198 Aigburth Rd'**,**'Liverpool'**,**

'L17 9PE'**,**'nrodriguez791@nhs.co.uk'**,**'07706193140' **),**

**(** 'Rose'**,**'Coleman'**,**'5 Alexandra Drive'**,**'Liverpool'**,**

'L17 8TA'**,**'rcoleman295@nhs.co.uk'**,**'07624567204' **),**

**(** 'Raymond'**,**'Cox'**,**'78 Arundel Ave'**,**'Liverpool'**,**'L17 2AX'**,**

'rcox041@nhs.co.uk'**,**'01515622876' **),**

**(** 'Karen'**,**'Ross'**,**'16 East Albert Rd'**,**'Liverpool'**,**'L17 4JU'**,**

'kros753@nhs.co.uk'**,**'07488210618' **),**

**(** 'Lillian'**,**'Brown'**,**'44 Adler Way'**,**'Liverpool'**,**'L3 4FX'**,**

'lbrown183@nhs.co.uk'**,**'07911050135' **),**

**(** 'Stephen'**,**'Foster'**,**'9 Back Sir Howard St'**,**'Liverpool'**,**

'L8 7SQ'**,**'sfoster234@nhs.co.uk'**,**'07457461205' **),**

**(** 'Christine'**,**'Cooper'**,**'11A Upper Newington'**,**'Liverpool'**,**

'L1 2SR'**,**'ccooper973@nhs.co.uk'**,**'07479859913' **),**

**(** 'Sandra'**,**'Howard'**,**'3 Slater St'**,**'Liverpool'**,**'L1 4BS'**,**

'showard819@nhs.co.uk'**,**'07893685330' **),**

**(** 'Belinda'**,**'Duchamp'**,**'3 Slater St'**,**'Liverpool'**,**'L1 4BS'**,**

'bduchamp092@nhs.co.uk'**,**'01512521055' **),**

**(** 'Cynthia'**,**'Gonzales'**,**'85 Bold St'**,**'Liverpool'**,**'L1 4HF'**,**

'cgonzales901@nhs.co.uk'**,**'07548128566' **),**

**(** 'Anna'**,**'Patterson'**,**'59 Overton Circle'**,**'Liverpool'**,**

'L3 8HB'**,**'annepatterson799@nhs.co.uk'**,**'07882133338' **),**

**(** 'Kim'**,**'Sung'**,**'50 Parr St'**,**'Liverpool'**,**'L1 4JN'**,**

'ksung263@nhs.co.uk'**,**'07556554401' **),**

**(** 'Louise'**,**'Watson'**,**'1f Columbus Quay'**,**'Liverpool'**,**

'L3 4DB'**,**'lwatson808@nhs.co.uk'**,**'01512976704' **),**

**(** 'Craig'**,**'Griffin'**,**'3 Rumford St'**,**'Liverpool'**,**'L2 8SZ'**,**

'cgriffon602@nhs.co.uk'**,**'01515017986' **)**

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- @ Set up variables for random records @

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- Declare the variables.

DECLARE @number\_of\_film\_records INT;

DECLARE @number\_of\_cust\_records INT;

DECLARE @number\_of\_perf\_records INT;

DECLARE @number\_of\_book\_records INT;

-- Initialise the variables.

SET @number\_of\_film\_records = 109;

SET @number\_of\_cust\_records = 45;

SET @number\_of\_perf\_records = 1000; -- Set the number of records to create.

SET @number\_of\_book\_records = 1000;

DECLARE @START\_PERFORMANCE\_DATE DATE; -- The start date for performance records.

DECLARE @PERF\_RECORDS\_PER\_DAY INT; -- Number of perf records to create for each day.

SET @START\_PERFORMANCE\_DATE = '01-JAN-2019';

SET @PERF\_RECORDS\_PER\_DAY = 100;

DECLARE @recordCount INT;

DECLARE @maxRecords INT;

DECLARE @count INT;

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

-- @ Create Random Performance Records @

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

PRINT 'Creating random performance records'

-- Declare the variables to be used.

**DECLARE** @FILM\_ID **INT;**

**DECLARE** @SCREEN\_NO **INT;**

**DECLARE** @PERFORMANCE\_DATE **DATE;**

**DECLARE** @START\_TIME **TIME**

**DECLARE** @PRICE **DECIMAL(**5**,**2**);**

**DECLARE** @IN\_3D **BIT;**

**SET** @PERFORMANCE\_DATE **=** @START\_PERFORMANCE\_DATE**;**

**SET** @PRICE **=** 10.99**;**

**DECLARE** @maxscreens **INT** **=** 10

**DECLARE** @maxperformances **INT** **=** 6 -- Number of performances per screen per day .

**SET** @recordCount **=** 0**;** -- Initialise the recordCount variable.

**WHILE** @recordCount **<** @number\_of\_perf\_records

**BEGIN**

-- Iterate through screening times.

**SET** @**count** **=** 1

**WHILE(**@**count** **<=** @maxperformances**)** **AND** **(**@recordCount **<** @number\_of\_perf\_records**)**

**BEGIN**

**SELECT** @START\_TIME **=** timeOfPerformance

**FROM** #times **WHERE** Id **=** @**count**

**SET** @SCREEN\_NO **=** 1

-- Iterate through the screen numbers.

**WHILE(**@SCREEN\_NO **<=** @maxscreens**)** **AND** **(**@recordCount **<** @number\_of\_perf\_records**)**

**BEGIN**

-- Choose a random film ID.

**SET** @FILM\_ID **=** **(FLOOR(** **RAND()\***@number\_of\_film\_records**)**+1**)**

-- Choose a random value for 'is in 3D'.

**IF** **ABS(CHECKSUM(NewId()))** **%** 10 **>** 6

**SET** @IN\_3D **=** 0**;**

**ELSE**

**SET** @IN\_3D **=** 1**;**

-- Insert the random performance.

**EXECUTE** insertPerformance @FILM\_ID**,** @SCREEN\_NO**,** @PERFORMANCE\_DATE**,** @START\_TIME**,** @PRICE**,** @IN\_3D**;**

**SET** @recordCount **=** @recordCount **+** 1**;**

**SET** @SCREEN\_NO **=** @SCREEN\_NO **+** 1

**END**

**SET** @**count** **=** @**count** **+** 1

**END**

-- Change performance date.

**SET** @PERFORMANCE\_DATE **=** **DATEADD(DAY,** 1**,** @PERFORMANCE\_DATE**)**

**END;**

PRINT 'Dropping temporary #times table.'

**DROP** **TABLE** #times

**SET** @**CURRENT\_TIME** **=** **(** **SELECT** **CONVERT(VARCHAR(**8**),** **GETDATE(),** 108**)** 'hh:mi:ss' **);**

PRINT 'Performance record creation finished at ' **+** @**CURRENT\_TIME**

-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
-- @ Create Random BOOKING Records @  
-- @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

PRINT 'Creating random booking records'

**DECLARE** @CUSTOMER\_ID **INT;**

**DECLARE** @PERFORMANCE\_ID **INT;**

**DECLARE** @NO\_OF\_TICKETS **INT;**

**SET** @recordCount **=** 0**;** -- Set the number of records to create

**WHILE** @recordCount **<** @number\_of\_book\_records

**BEGIN**

-- Choose a random customer ID.

**SET** @CUSTOMER\_ID **=** **(FLOOR(** **RAND()\***@number\_of\_cust\_records**)**+1**)**

-- Choose a random performance ID.

**SET** @PERFORMANCE\_ID **=** **(FLOOR(** **RAND()\***@number\_of\_perf\_records**)**+1**)**

-- Choose a random number of tickets.

**SET** @NO\_OF\_TICKETS **=** **(ABS(CHECKSUM(NewId()))** **%** 20**)** **+** 1**;**

-- Insert random booking record

**EXECUTE** insertBooking @CUSTOMER\_ID**,** @PERFORMANCE\_ID**,** @NO\_OF\_TICKETS**;**

**SET** @recordCount **=** @recordCount **+** 1**;**

**END;**

**SET** @**CURRENT\_TIME** **=** **(** **SELECT** **CONVERT(VARCHAR(**8**),** **GETDATE(),** 108**)** 'hh:mi:ss' **);**

PRINT 'Booking record creation finished at ' **+** @**CURRENT\_TIME**

PRINT 'Random record creation finished.'

# Test Queries

-- ----------------------------------------------------------------------------

--

-- Test Queries

--

-- Note: Before performing these tests please create a fresh installation of

-- the database and run the Test Data Creation script.

--

-- ----------------------------------------------------------------------------

-- ----------------------------------------------------------------------------

-- Test Query 1: Get all of today's bookings.

-- ----------------------------------------------------------------------------

**DECLARE** @todaysDate **DATETIME;**

**DECLARE** @tomorrowsDate **DATETIME;**

**SET** @todaysDate **=** **cast(cast(getdate()** **as** **date** **)** **as** **datetime);**

**SET** @tomorrowsDate **=** **DATEADD(DAY,** 1**,** @todaysDate**);**

**SELECT** BT**.**BOOKING\_ID**,**

BT**.**CUSTOMER\_ID**,**

BT**.**PERFORMANCE\_ID**,**

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START\_TIME"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END\_TIME"**,**

BT**.**NO\_OF\_TICKETS

**FROM** bookingsTBL "BT"

**INNER** **JOIN** performanceTBL "PT" **ON** **(**BT**.**PERFORMANCE\_ID**=**PT**.**PERFORMANCE\_ID**)**

**WHERE** PERFORMANCE\_DATE**>=**@todaysDate

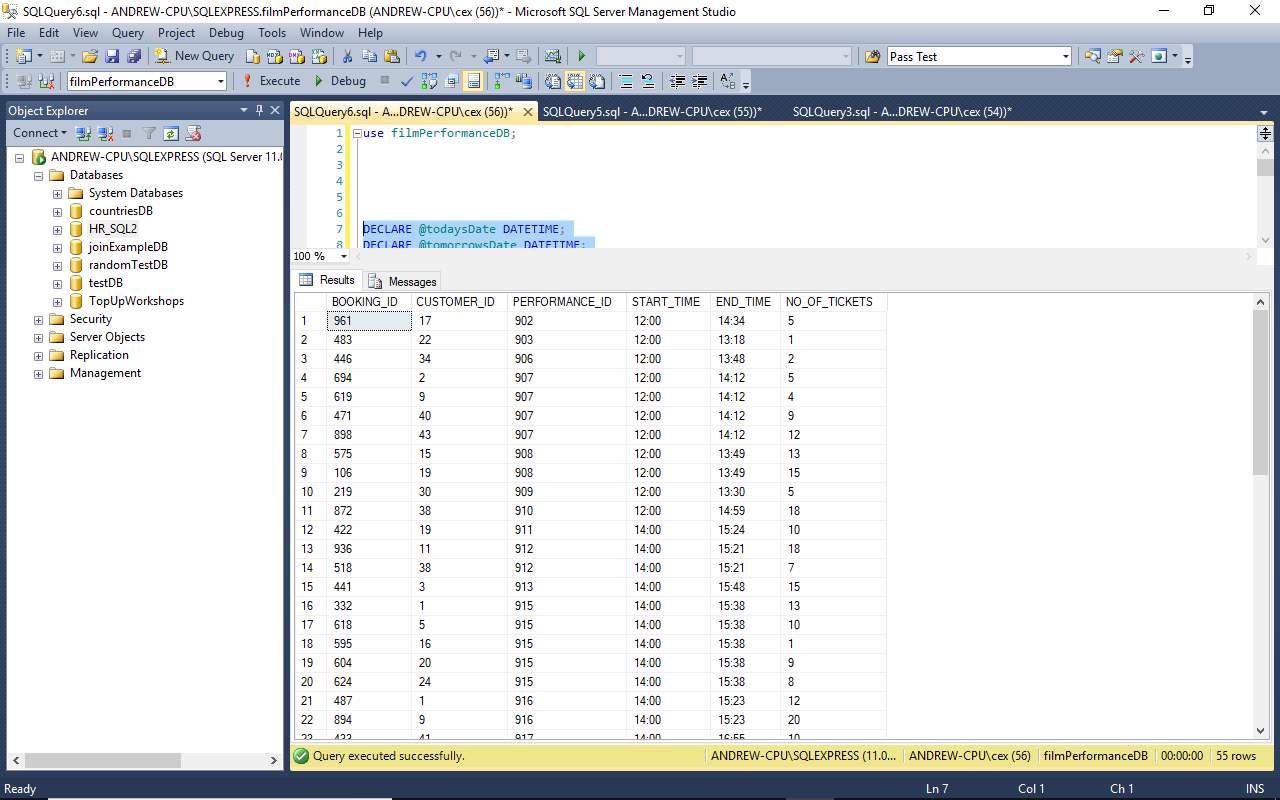
**AND** PERFORMANCE\_DATE**<**@tomorrowsDate

**ORDER** **BY** PT**.**START\_TIME **ASC,**

PT**.**PERFORMANCE\_ID **ASC,**

BT**.**CUSTOMER\_ID **ASC;**

-- ----------------------------------------------------------------------------

 *Figure 1: Screenshot of the results produced by running Test Query 1.*

-- ----------------------------------------------------------------------------

-- Test Query 2: Get total number of tickets booked

-- for films by the actor Woody Allen.

-- ----------------------------------------------------------------------------

**SELECT** **COALESCE(SUM(**BT**.**NO\_OF\_TICKETS**),**0**)** **AS** "TOTAL"

**FROM** **(** filmsTBL "FT"

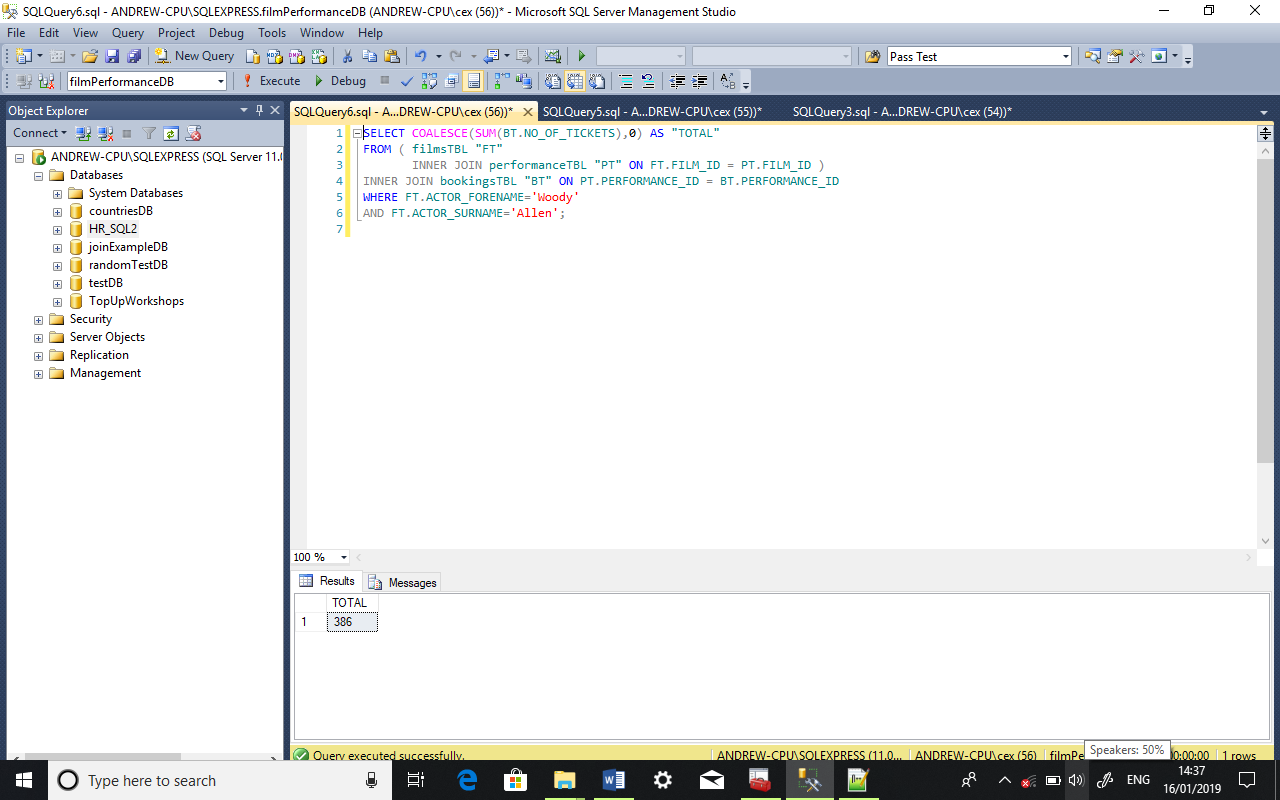
**INNER** **JOIN** performanceTBL "PT" **ON** FT**.**FILM\_ID **=** PT**.**FILM\_ID **)**

**INNER** **JOIN** bookingsTBL "BT" **ON** PT**.**PERFORMANCE\_ID **=** BT**.**PERFORMANCE\_ID

**WHERE** FT**.**ACTOR\_FORENAME**=**'Woody'

**AND** FT**.**ACTOR\_SURNAME**=**'Allen'**;**

-- ----------------------------------------------------------------------------

 *Figure 2: Screenshot of the results produced by running Test Query 2.*

-- ----------------------------------------------------------------------------

-- Test Query 3: Find the names of actors who are also directors.

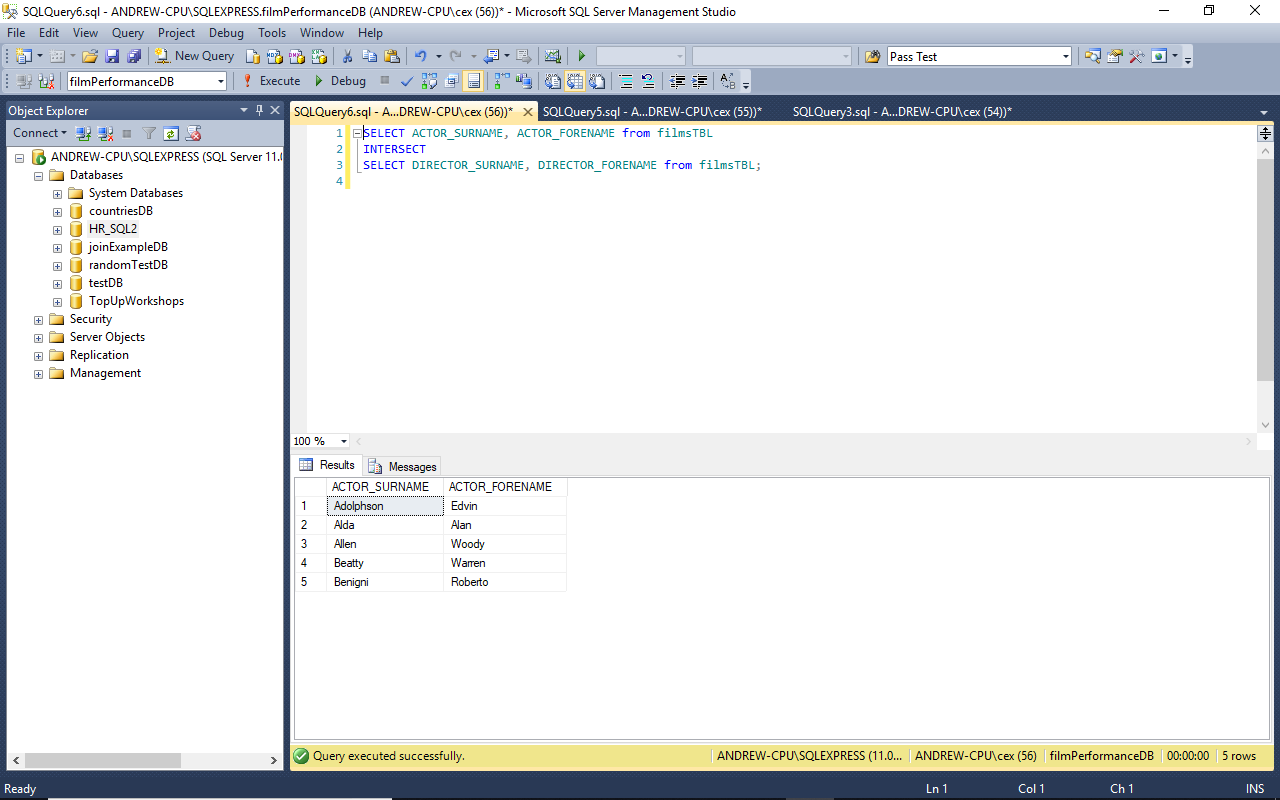
-- ----------------------------------------------------------------------------

-- INTERSECT will return results which occur in the first and second query. **SELECT** ACTOR\_SURNAME**,** ACTOR\_FORENAME **from** filmsTBL

**INTERSECT**

**SELECT** DIRECTOR\_SURNAME**,** DIRECTOR\_FORENAME **from** filmsTBL**;**

-- ----------------------------------------------------------------------------

 *Figure 3: Screenshot of the results produced by running Test Query 3.*

-- ----------------------------------------------------------------------------

-- Test Query 4: Find the names of directors who are also actors

-- but not comedy actors.

-- ----------------------------------------------------------------------------

-- EXCEPT will return results which occur in the INTERSECT query but not  
-- in the last query. **SELECT** DIRECTOR\_SURNAME**,** DIRECTOR\_FORENAME **from** filmsTBL

**INTERSECT**

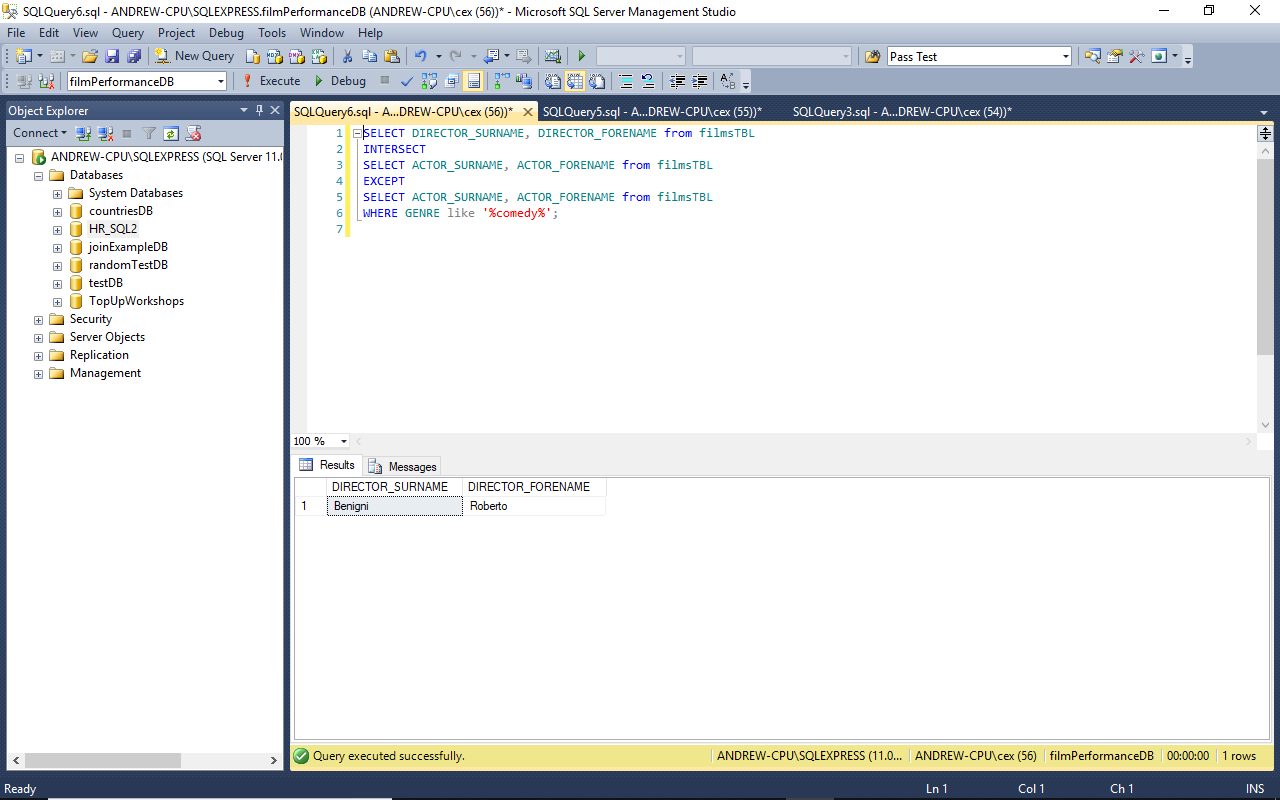
**SELECT** ACTOR\_SURNAME**,** ACTOR\_FORENAME **from** filmsTBL

**EXCEPT**

**SELECT** ACTOR\_SURNAME**,** ACTOR\_FORENAME **from** filmsTBL

**WHERE** GENRE **like** '%comedy%'**;**

-- ----------------------------------------------------------------------------

 *Figure 4: Screenshot of the results produced by running Test Query 4.*

-- ----------------------------------------------------------------------------

-- Test Query 5: Get all the first and last names from the filmsTBL

-- and customersTBL tables.

-- ----------------------------------------------------------------------------

-- UNION will combine the result sets from all the queries **SELECT** ACTOR\_SURNAME**,** ACTOR\_FORENAME **from** filmsTBL

**UNION**

**SELECT** ACTRESS\_SURNAME**,** ACTRESS\_FORENAME **from** filmsTBL

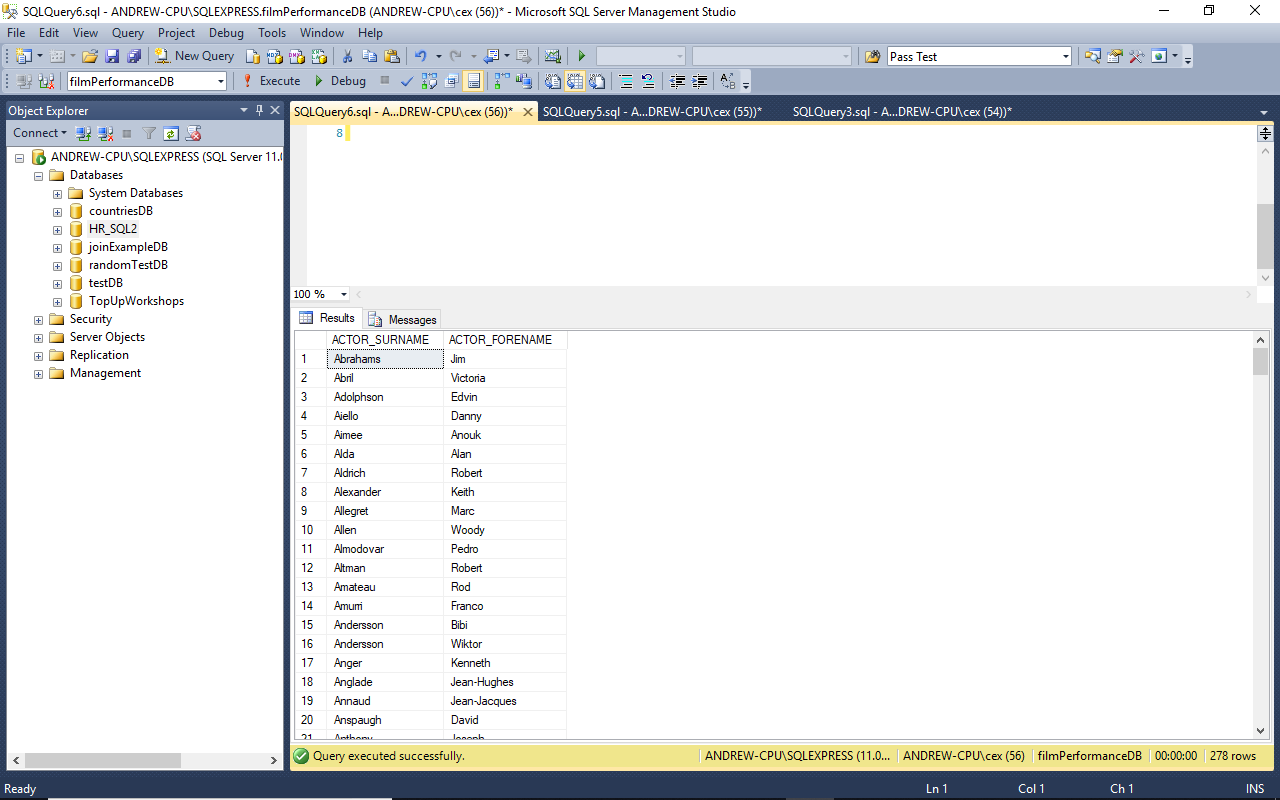
**UNION**

**SELECT** DIRECTOR\_SURNAME**,** DIRECTOR\_FORENAME **from** filmsTBL

**UNION**

**SELECT** LASTNAME**,** FIRSTNAME **from** customersTBL**;**

-- ----------------------------------------------------------------------------

 *Figure 5: Screenshot of the results produced by running Test Query 5.*

-- ----------------------------------------------------------------------------

-- Test Query 6: Get the details for all of today's performances.

-- ----------------------------------------------------------------------------

**DECLARE** @todaysDate **DATETIME;**

**DECLARE** @tomorrowsDate **DATETIME;**

**SET** @todaysDate **=** **cast(cast(getdate()** **as** **date** **)** **as** **datetime);**

**SET** @tomorrowsDate **=** **DATEADD(DAY,** 1**,** @todaysDate**);**

**SELECT** FT**.**TITLE**,**

FT**.**GENRE**,**

PT**.**SCREEN\_NO "SN"**,**

**FORMAT** **(**PT**.**PERFORMANCE\_DATE**,** 'd'**,** 'en-gb' **)** "DATE"**,**

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END"**,**

PT**.**PRICE**,**

IIF**(**PT**.**IN\_3D **=** 1**,** 'YES'**,** 'NO'**)** "3D" -- If value is 1 return Yes, otherwise No.

**FROM** performanceTBL "PT"

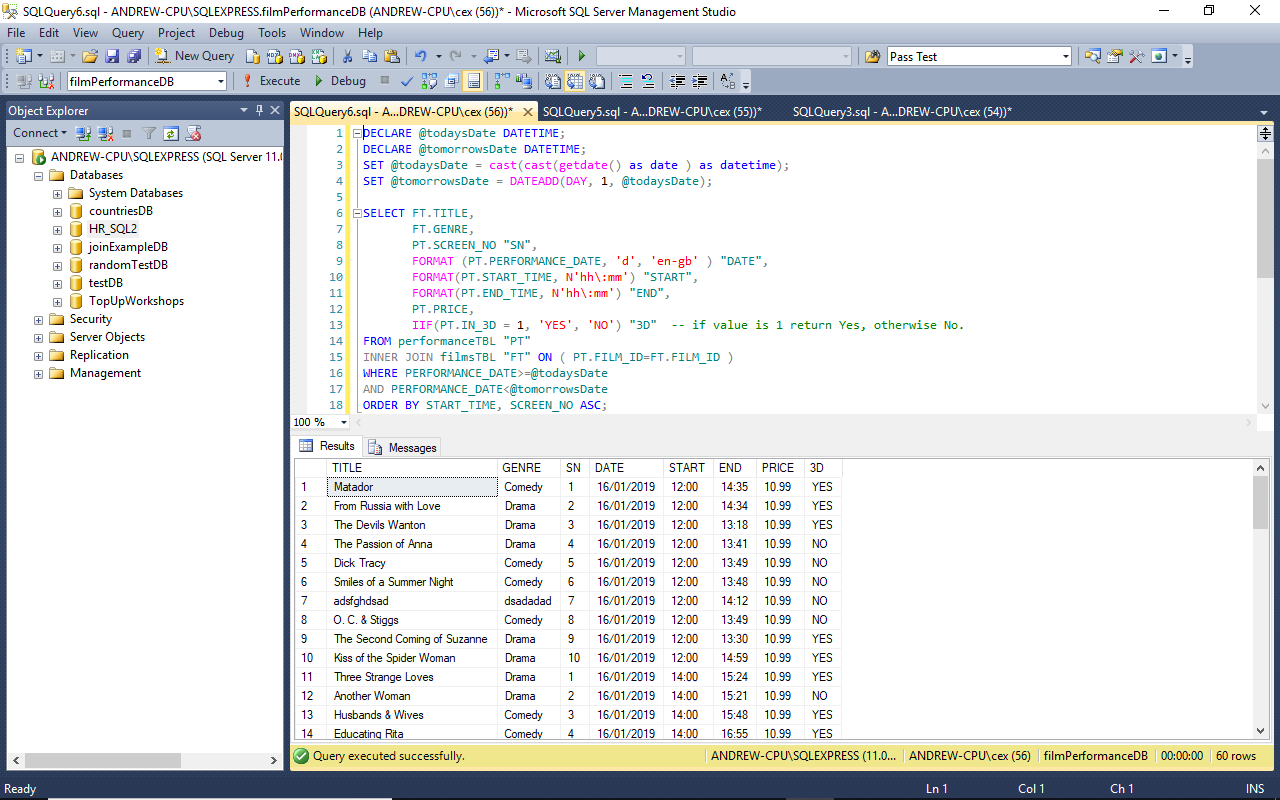
**INNER** **JOIN** filmsTBL "FT" **ON** **(** PT**.**FILM\_ID**=**FT**.**FILM\_ID **)**

**WHERE** PERFORMANCE\_DATE**>=**@todaysDate

**AND** PERFORMANCE\_DATE**<**@tomorrowsDate

**ORDER** **BY** START\_TIME**,** SCREEN\_NO **ASC;**

-- ----------------------------------------------------------------------------

 *Figure 6: Screenshot of the results produced by running Test Query 6.*

-- ----------------------------------------------------------------------------

-- Test Query 7: Get a list of all director names from filmsTBL.

-- ----------------------------------------------------------------------------

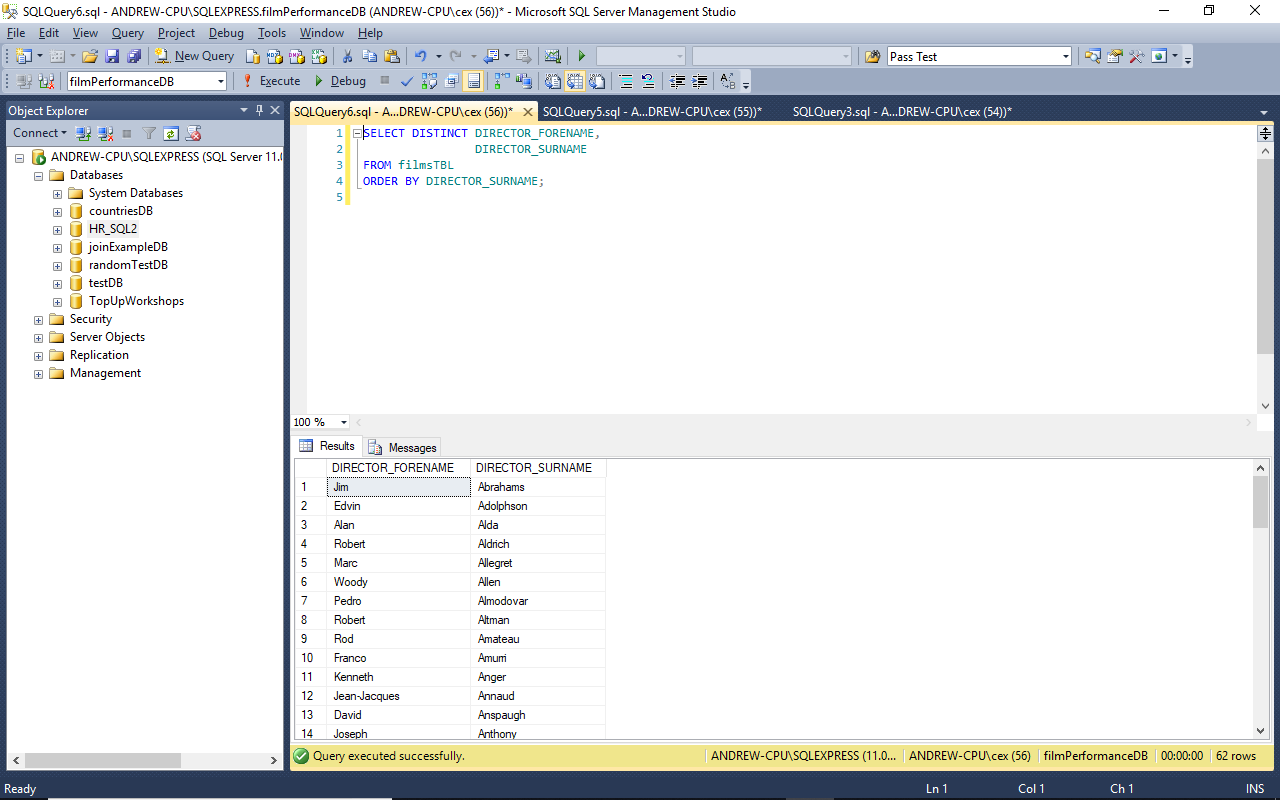
**SELECT** **DISTINCT** DIRECTOR\_FORENAME**,**

DIRECTOR\_SURNAME

**FROM** filmsTBL

**ORDER** **BY** DIRECTOR\_SURNAME**;**

-- ----------------------------------------------------------------------------

 *Figure 7: Screenshot of the results produced by running Test Query 7.*

-- ----------------------------------------------------------------------------

-- Test Query 8: Insert and update a record.

-- ----------------------------------------------------------------------------

**INSERT** **INTO** filmsTBL

**(** TITLE**,** GENRE**,** DURATION**,** ACTOR\_SURNAME**,**

ACTOR\_FORENAME**,** ACTRESS\_SURNAME**,** ACTRESS\_FORENAME**,**

DIRECTOR\_SURNAME**,** DIRECTOR\_FORENAME **)**

**VALUES** **(** 'Test8'**,** 'Test'**,** '03:00:00'**,** 'Morris'**,**

'Alpha'**,** 'Murphy'**,** 'Doris'**,** 'Fletcher'**,** 'Ricardo' **);**

**SELECT** **\***

**FROM** filmsTBL

**WHERE** TITLE **like** 'Test8'**;**

**UPDATE** filmsTBL

**SET** TITLE**=**'Test Film'**,**

GENRE**=**'Test'**,**

DURATION**=**'03:00:00'**,**

ACTOR\_SURNAME**=**'Morris'**,**

ACTOR\_FORENAME**=**'Alpha'**,**

ACTRESS\_SURNAME**=**'Murphy'**,**

ACTRESS\_FORENAME**=**'Doris'**,**

DIRECTOR\_SURNAME**=**'Fletcher'**,**

DIRECTOR\_FORENAME**=**'Ricardo'

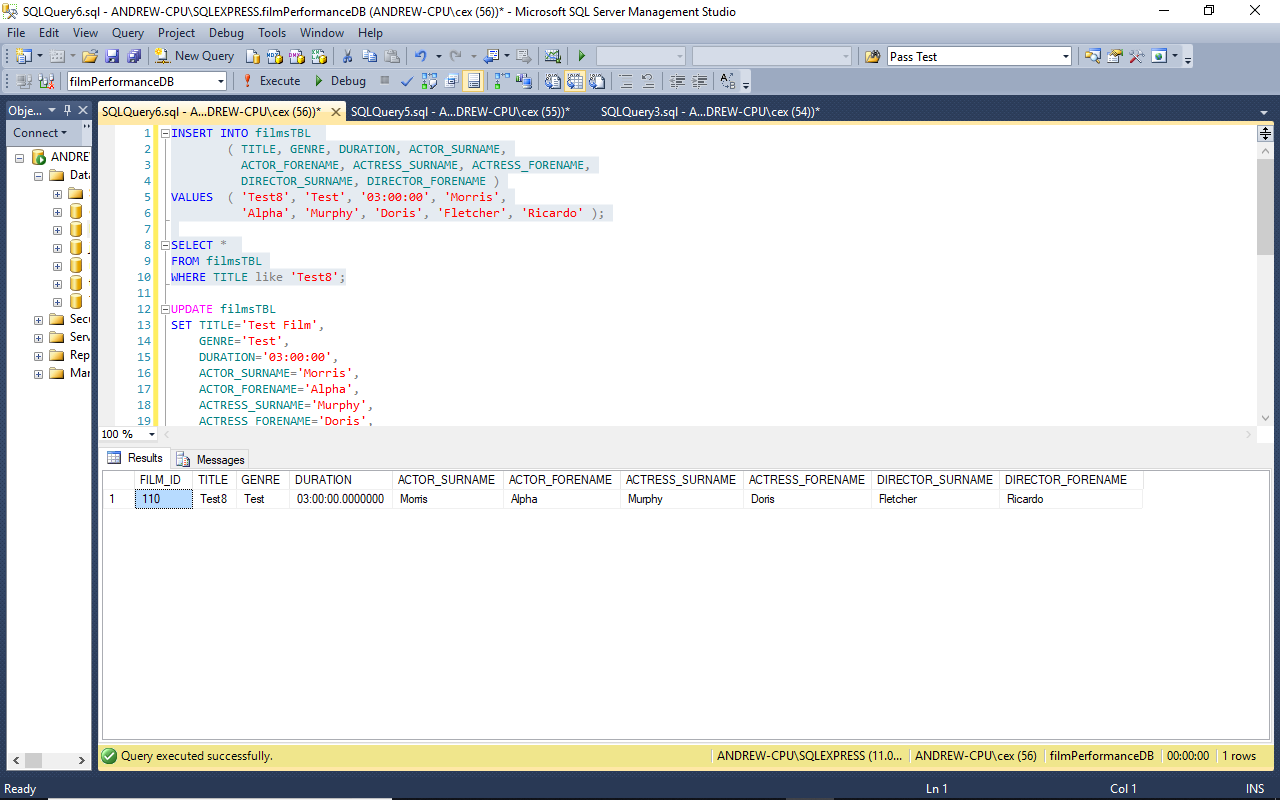
**WHERE** TITLE**=**'Test8'**;**

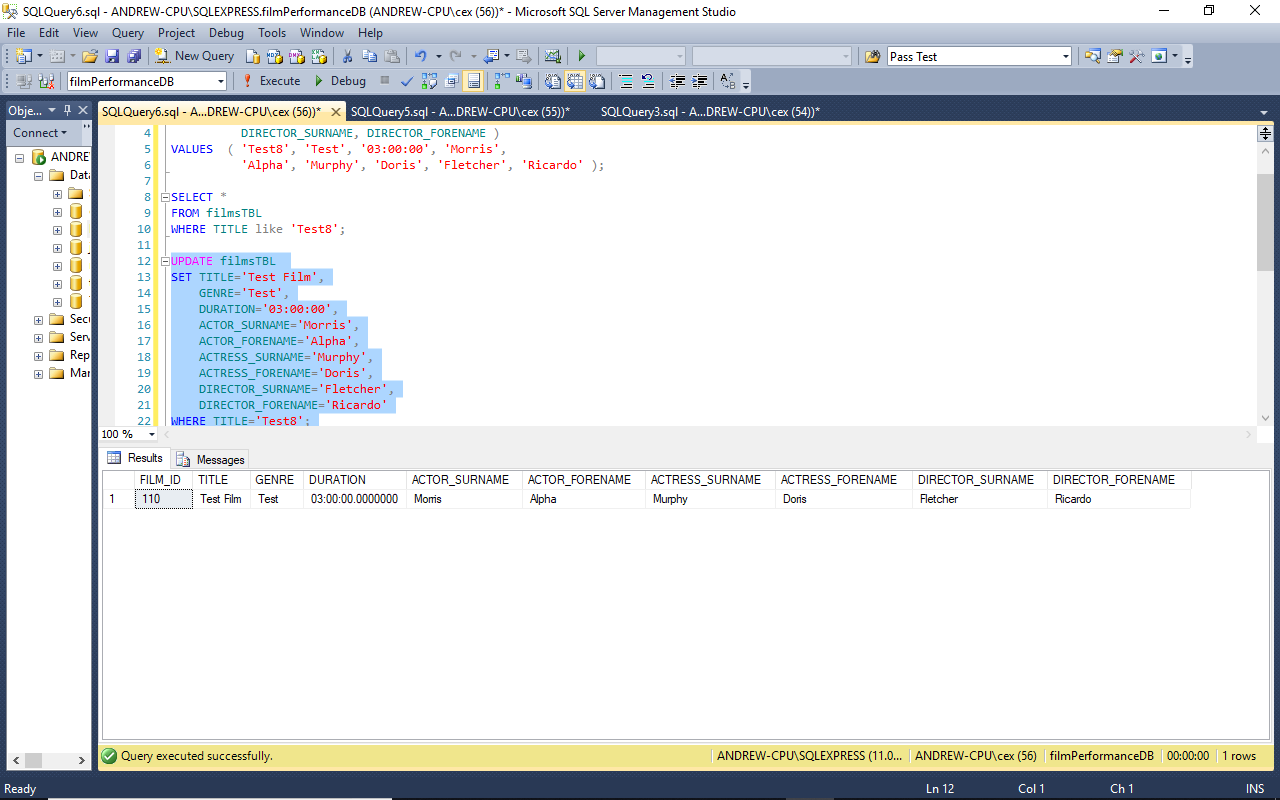
**SELECT** **\***

**FROM** filmsTBL

**WHERE** TITLE **like** 'Test Film'**;**

-- ----------------------------------------------------------------------------

 *Figure 8: Screenshot of the results produced by inserting the new record for Test Query 8.*

 *Figure 9: Screenshot of the results produced by updating the record for Test Query 8.*

-- ----------------------------------------------------------------------------

-- Test Query 9: Delete the record created during the previous test.

-- ----------------------------------------------------------------------------

**DELETE** **FROM** filmsTBL

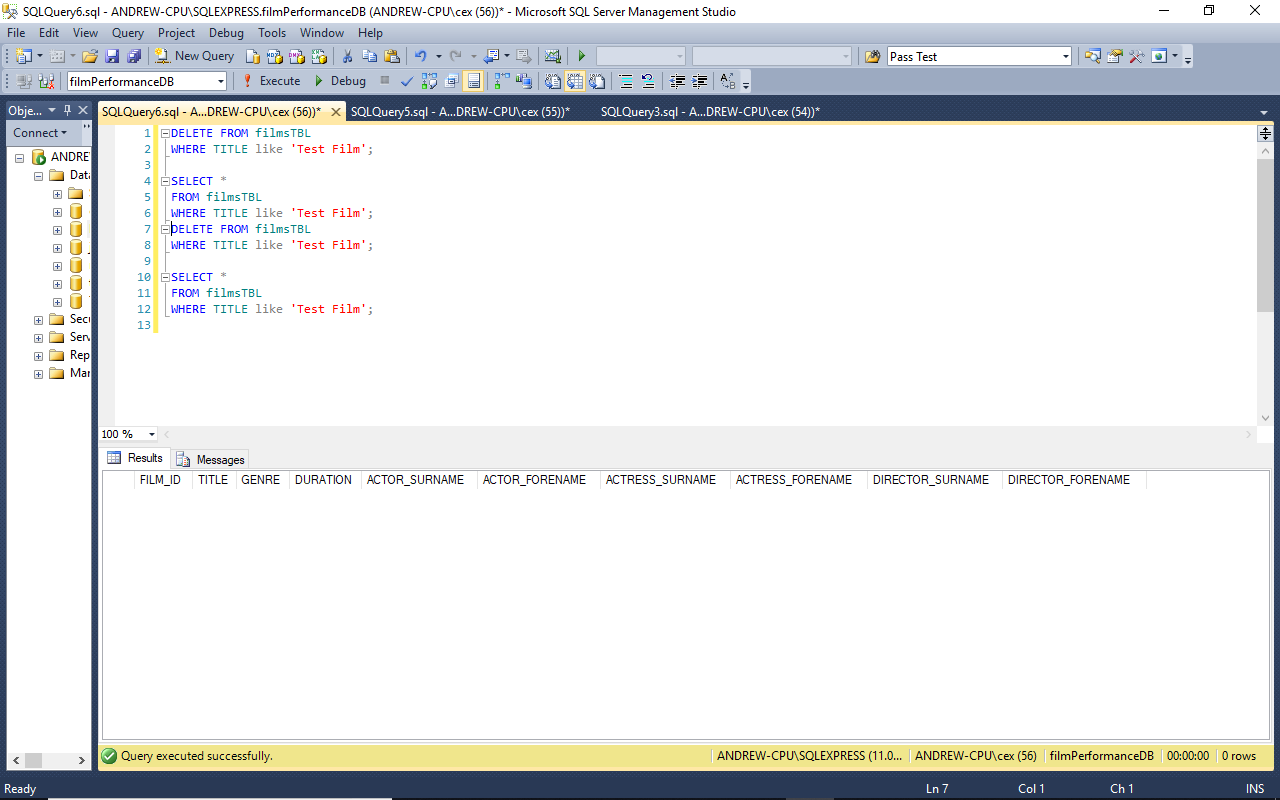
**WHERE** TITLE **like** 'Test Film'**;**

**SELECT** **\***

**FROM** filmsTBL

**WHERE** TITLE **like** 'Test Film'**;**

-- ----------------------------------------------------------------------------

 *Figure 10: Screenshot of the results produced by running Test Query 9.*

-- ----------------------------------------------------------------------------

-- Test Query 10: Get contact details for all customers

-- who have made bookings today.

-- ----------------------------------------------------------------------------

**DECLARE** @todaysDate **DATETIME;**

**DECLARE** @tomorrowsDate **DATETIME;**

**SET** @todaysDate **=** **cast(cast(getdate()** **as** **date** **)** **as** **datetime);**

**SET** @tomorrowsDate **=** **DATEADD(DAY,** 1**,** @todaysDate**);**

**SELECT** **DISTINCT** CT**.**EMAIL**,**

CT**.**PHONE\_NO**,**

CT**.**FIRSTNAME**,**

CT**.**LASTNAME

**FROM** customersTBL "CT"

**INNER** **JOIN** bookingsTBL "BT" **ON** **(**BT**.**CUSTOMER\_ID**=**CT**.**CUSTOMER\_ID**)**

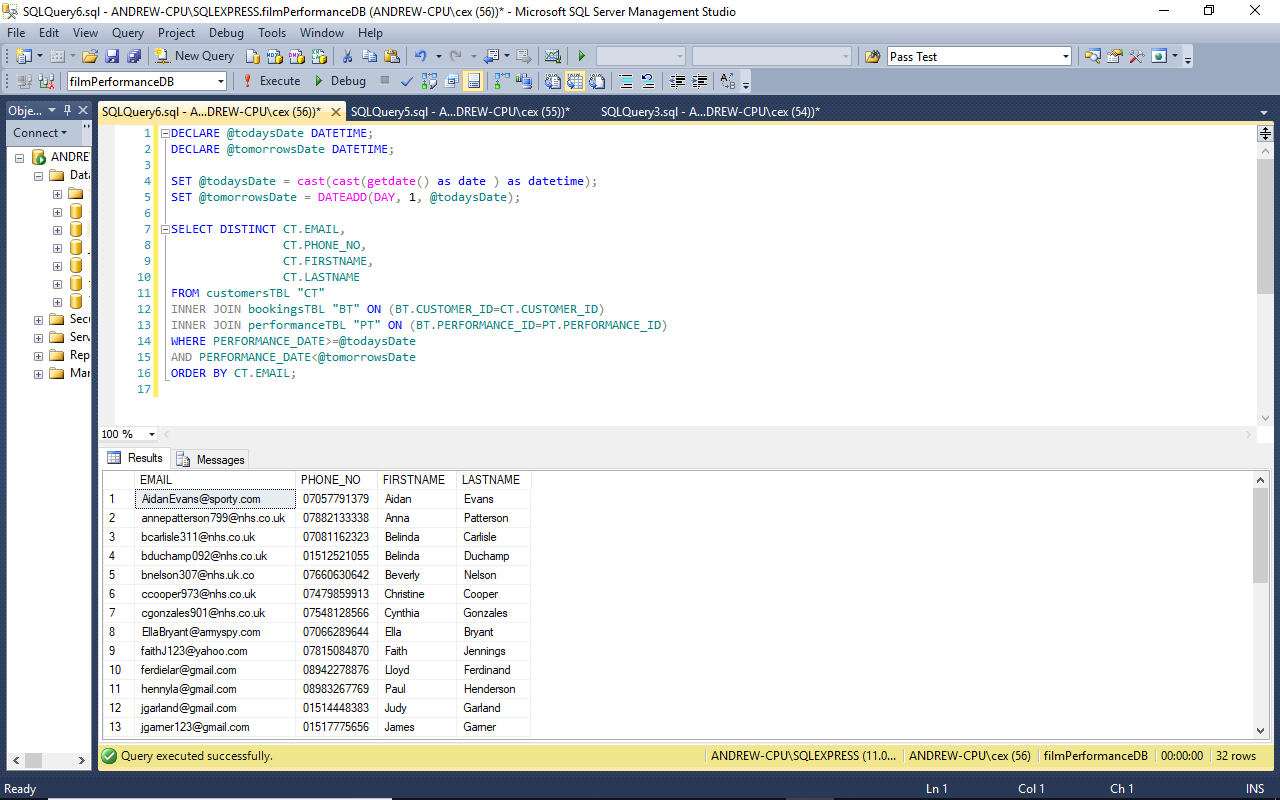
**INNER** **JOIN** performanceTBL "PT" **ON** **(**BT**.**PERFORMANCE\_ID**=**PT**.**PERFORMANCE\_ID**)**

**WHERE** PERFORMANCE\_DATE**>=**@todaysDate

**AND** PERFORMANCE\_DATE**<**@tomorrowsDate

**ORDER** **BY** CT**.**EMAIL**;**

-- ----------------------------------------------------------------------------

 *Figure 11: Screenshot of the results produced by running Test Query 10.*

-- ----------------------------------------------------------------------------

-- Test Query 11: Get details for all films with Love in the title.

-- ----------------------------------------------------------------------------

**SELECT** FILM\_ID**,**

TITLE**,**

GENRE**,**

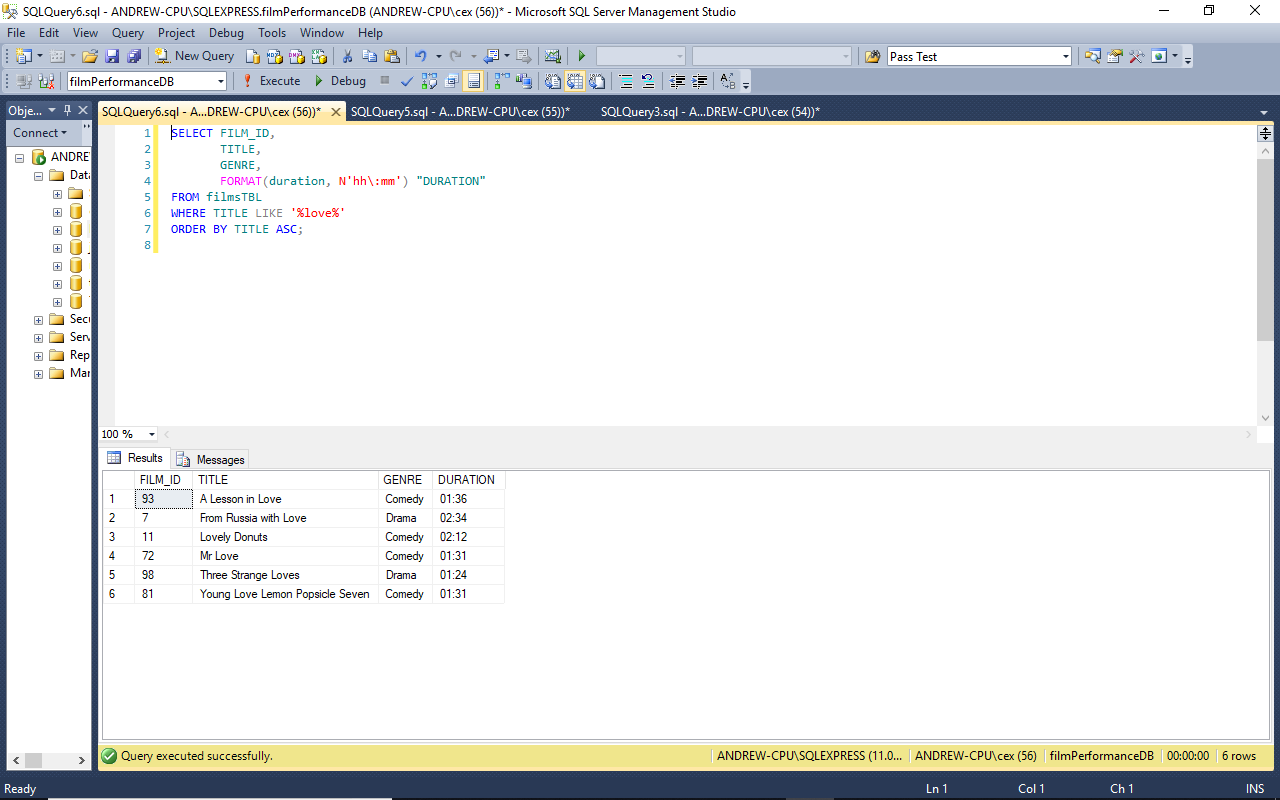
**FORMAT(**duration**,** N'hh\:mm'**)** "DURATION"

**FROM** filmsTBL

**WHERE** TITLE **LIKE** '%love%'

**ORDER** **BY** TITLE **ASC;**

-- ----------------------------------------------------------------------------

 *Figure 12: Screenshot of the results produced by running Test Query 11.*

-- ----------------------------------------------------------------------------

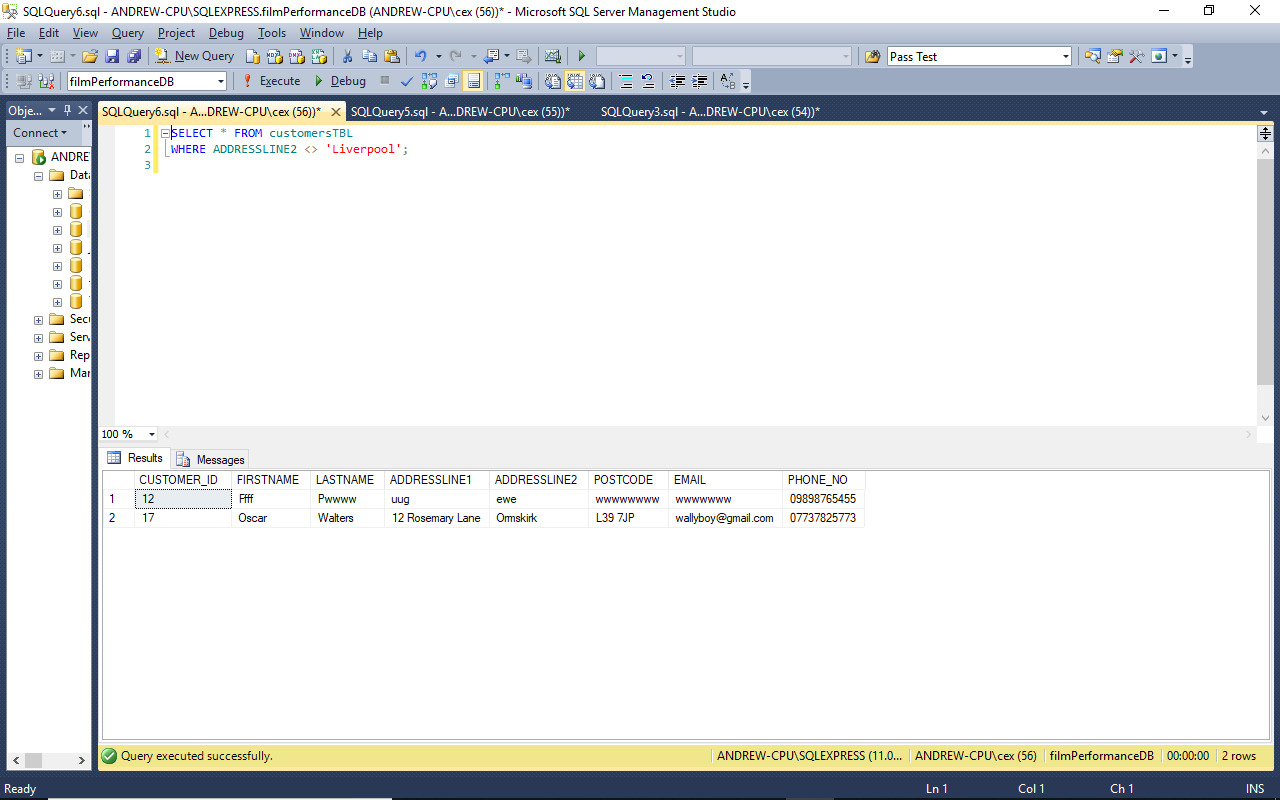
-- Test Query 12: Get customer records for all people not living in Liverpool.

-- ----------------------------------------------------------------------------

**SELECT** **\*** **FROM** customersTBL

**WHERE** ADDRESSLINE2 **<>** 'Liverpool'**;**

-- ----------------------------------------------------------------------------

 *Figure 13: Screenshot of the results produced by running Test Query 12.*

-- ----------------------------------------------------------------------------

-- Test Query 13: Get the customer name, film title and number of tickets of   
-- all performance bookings made by customers who wanted more  
-- than 19 tickets.

-- ----------------------------------------------------------------------------

**SELECT** **DISTINCT** CT**.**FIRSTNAME**,**

CT**.**LASTNAME**,**

FT**.**TITLE "Film Title"**,**

BT**.**NO\_OF\_TICKETS

**FROM** bookingsTBL "BT"

**INNER** **JOIN** performanceTBL "PT"

**ON** **(** BT**.**PERFORMANCE\_ID**=**PT**.**PERFORMANCE\_ID **)**

**INNER** **JOIN** filmsTBL "FT"

**ON** **(** FT**.**FILM\_ID**=**PT**.**FILM\_ID **)**

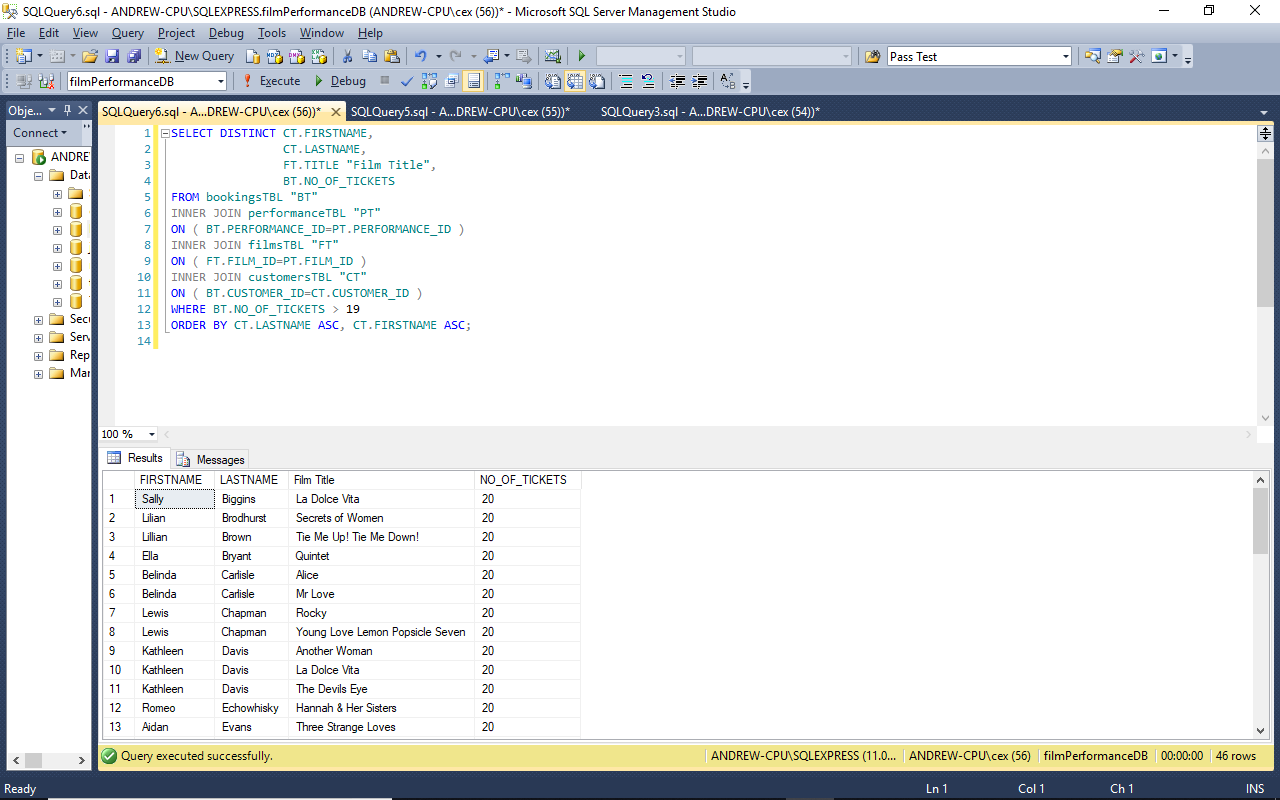
**INNER** **JOIN** customersTBL "CT"

**ON** **(** BT**.**CUSTOMER\_ID**=**CT**.**CUSTOMER\_ID **)**

**WHERE** BT**.**NO\_OF\_TICKETS **>** 19

**ORDER** **BY** CT**.**LASTNAME **ASC,** CT**.**FIRSTNAME **ASC;**

-- ----------------------------------------------------------------------------

 *Figure 14: Screenshot of the results produced by running Test Query 13.*

-- ----------------------------------------------------------------------------

-- Test Query 14: Get details for all films being screened today in

-- rooms with a hearing loop.

-- ----------------------------------------------------------------------------

**DECLARE** @todaysDate **DATETIME;**

**DECLARE** @tomorrowsDate **DATETIME;**

**SET** @todaysDate **=** **cast(cast(getdate()** **as** **date** **)** **as** **datetime);**

**SET** @tomorrowsDate **=** **DATEADD(DAY,** 1**,** @todaysDate**)** **;**

**SELECT** PT**.**SCREEN\_NO "SN"**,**

FT**.**TITLE**,**

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END"

**FROM** performanceTBL "PT"

**INNER** **JOIN** filmsTBL "FT"

**ON** **(** FT**.**FILM\_ID**=**PT**.**FILM\_ID **)**

**INNER** **JOIN** screenTBL "ST"

**ON** **(** PT**.**SCREEN\_NO**=**ST**.**SCREEN\_NO **)**

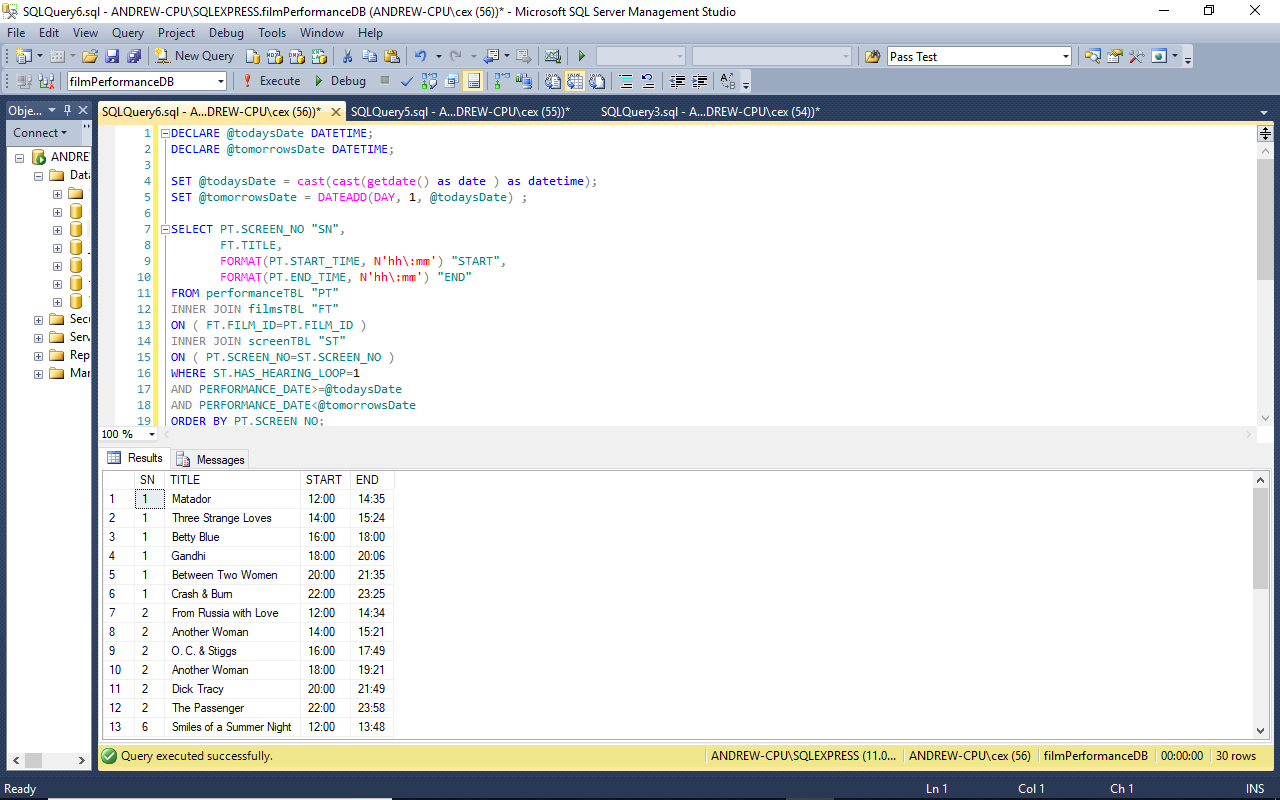
**WHERE** ST**.**HAS\_HEARING\_LOOP**=**1

**AND** PERFORMANCE\_DATE**>=**@todaysDate

**AND** PERFORMANCE\_DATE**<**@tomorrowsDate

**ORDER** **BY** PT**.**SCREEN\_NO**;**

-- ----------------------------------------------------------------------------

 *Figure 15: Screenshot of the results produced by running Test Query 14.*

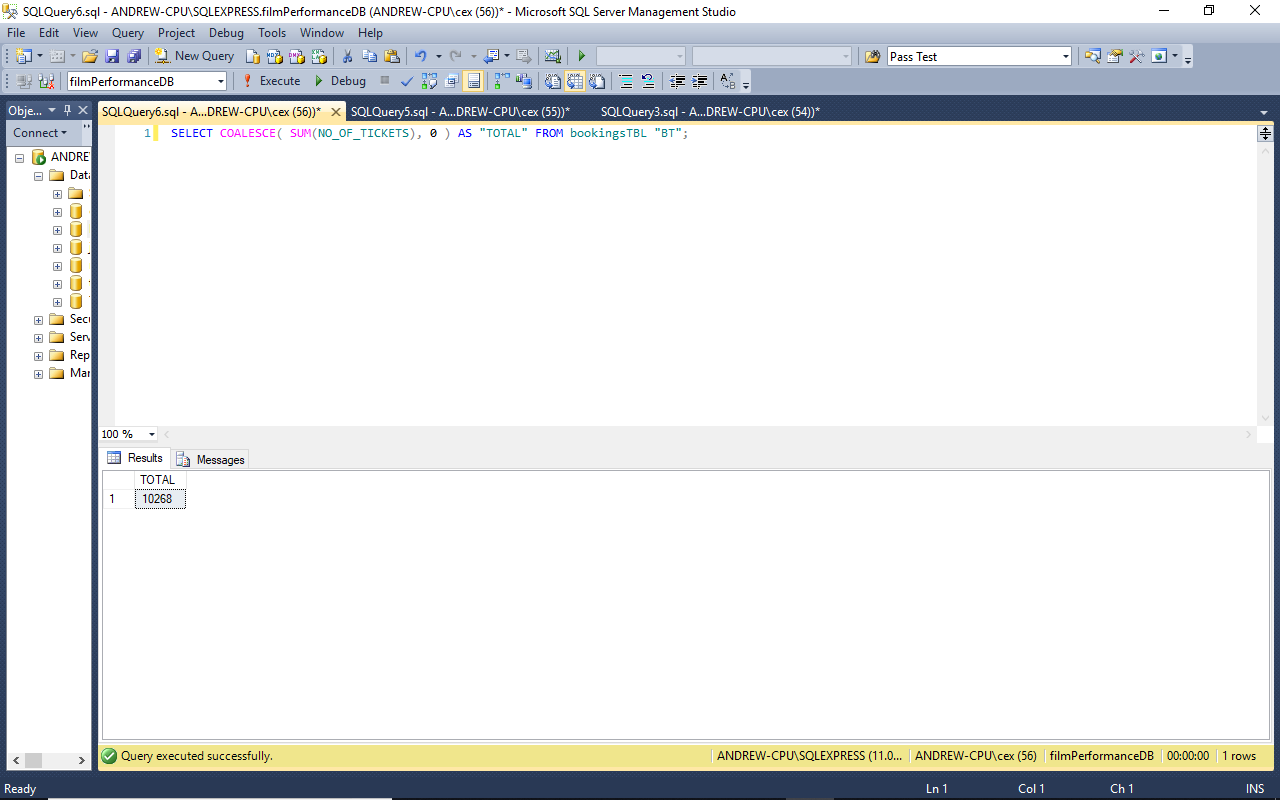
-- ----------------------------------------------------------------------------

-- Test Query 15: Get total number of tickets booked for all time.

-- ----------------------------------------------------------------------------

**SELECT** **COALESCE(** **SUM(**NO\_OF\_TICKETS**),** 0 **)** **AS** "TOTAL" **FROM** bookingsTBL "BT"**;**

-- ----------------------------------------------------------------------------

 *Figure 16: Screenshot of the results produced by running Test Query 15.*

-- ----------------------------------------------------------------------------

-- Test Query 16: Get the details for all scheduled performances of the film

-- Nightbreed.

-- ----------------------------------------------------------------------------

**SELECT** FT**.**TITLE**,**

PT**.**SCREEN\_NO**,**

**FORMAT** **(**PT**.**PERFORMANCE\_DATE**,** 'd'**,** 'en-gb' **)** "DATE"**,**

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END"**,**

PT**.**PRICE**,**

IIF**(**PT**.**IN\_3D **=** 1**,** 'YES'**,** 'NO'**)** "3D"

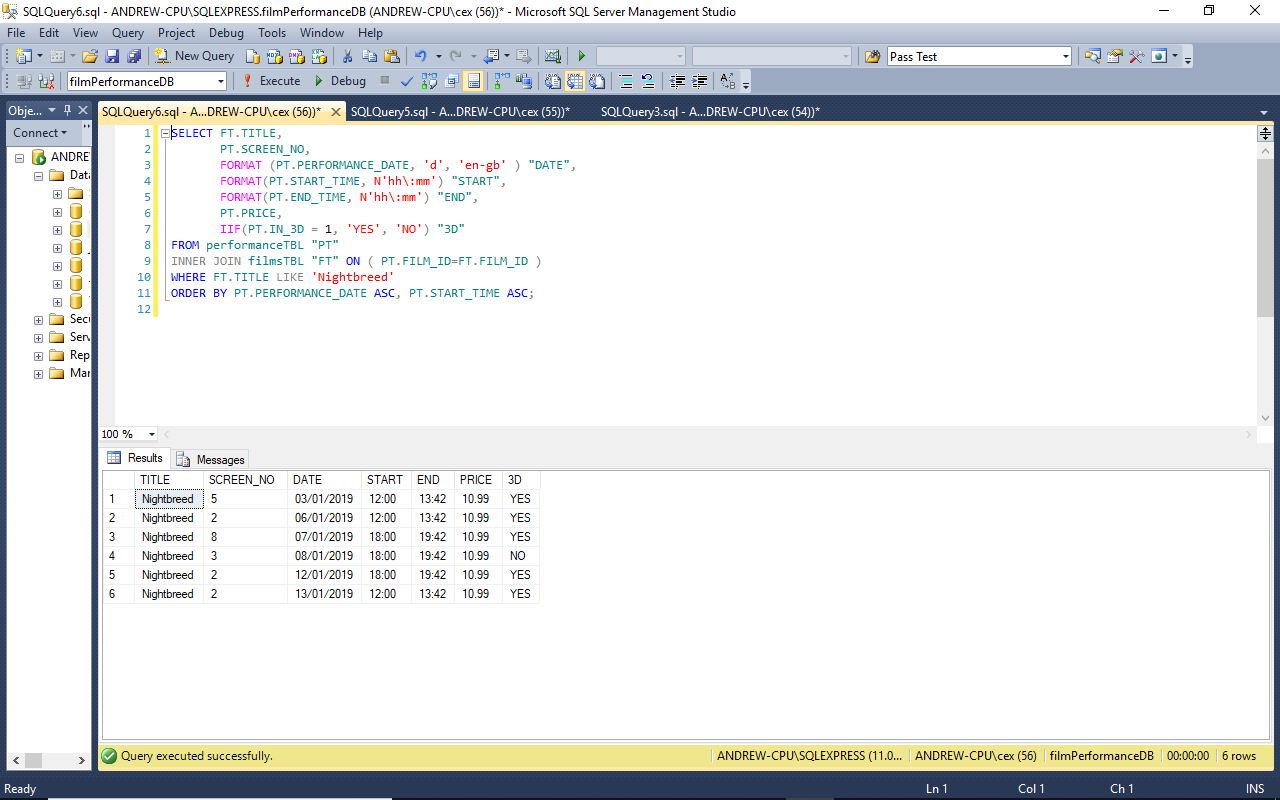
**FROM** performanceTBL "PT"

**INNER** **JOIN** filmsTBL "FT" **ON** **(** PT**.**FILM\_ID**=**FT**.**FILM\_ID **)**

**WHERE** FT**.**TITLE **LIKE** 'Nightbreed'

**ORDER** **BY** PT**.**PERFORMANCE\_DATE **ASC,** PT**.**START\_TIME **ASC;**

-- ----------------------------------------------------------------------------

 *Figure 17: Screenshot of the results produced by running Test Query 16.*

-- ----------------------------------------------------------------------------

-- Test Query 17: Get a list of comedy films less than an hour and a half long.

-- ----------------------------------------------------------------------------

**SELECT** FT**.**TITLE**,**

**FORMAT(**FT**.**DURATION**,** N'hh\:mm'**)** "DURATION"

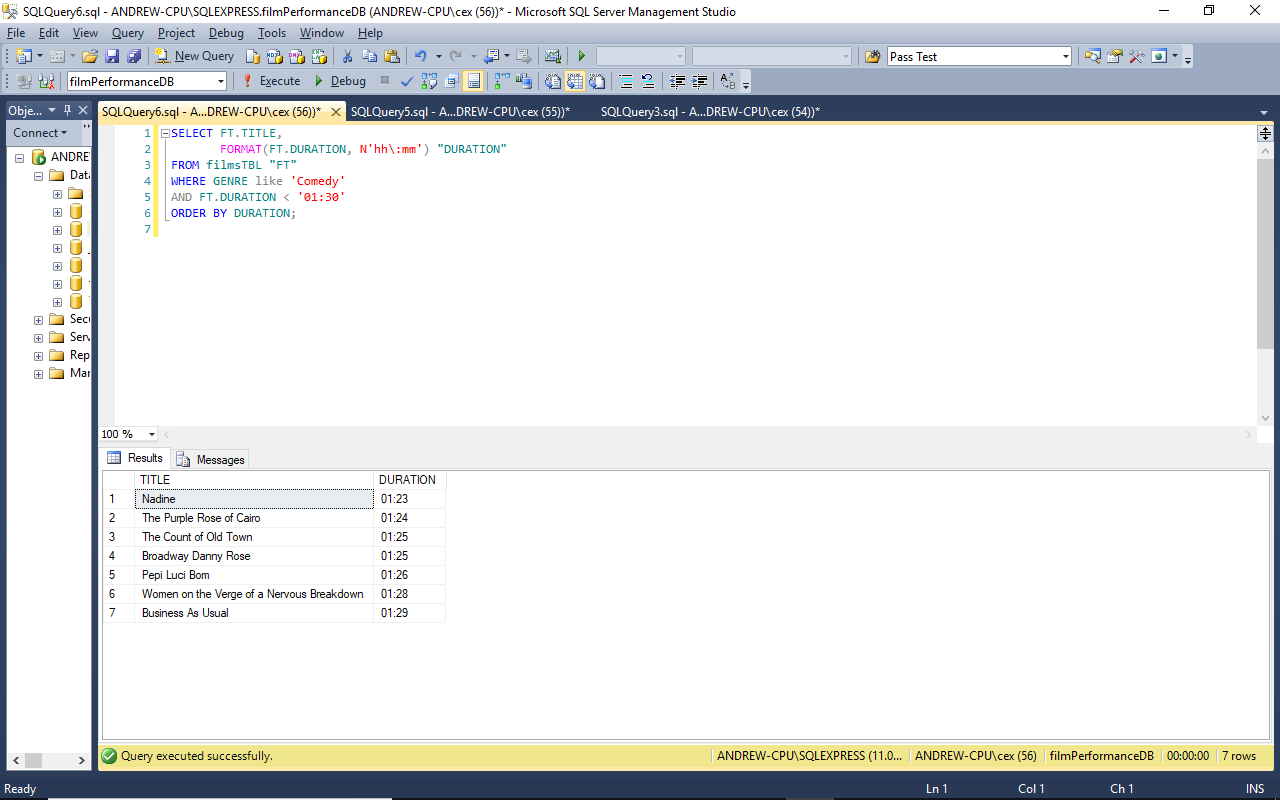
**FROM** filmsTBL "FT"

**WHERE** GENRE **like** 'Comedy'

**AND** FT**.**DURATION **<** '01:30'

**ORDER** **BY** DURATION**;**

-- ----------------------------------------------------------------------------

 *Figure 18: Screenshot of the results produced by running Test Query 17.*

-- ----------------------------------------------------------------------------

-- Test Query 18: Get details for all today’s performances on screen 1.

-- ----------------------------------------------------------------------------

**DECLARE** @todaysDate **DATETIME**

**DECLARE** @tomorrowsDate **DATETIME**

**SET** @todaysDate **=** **cast(cast(getdate()** **as** **date** **)** **as** **datetime)**

**SET** @tomorrowsDate **=** **DATEADD(DAY,** 1**,** @todaysDate**)**

**SELECT** FT**.**TITLE**,**

FT**.**GENRE**,**

PT**.**SCREEN\_NO "SN"**,**

**FORMAT** **(**PT**.**PERFORMANCE\_DATE**,** 'd'**,** 'en-gb' **)** "DATE"**,**

**FORMAT(**PT**.**START\_TIME**,** N'hh\:mm'**)** "START"**,**

**FORMAT(**PT**.**END\_TIME**,** N'hh\:mm'**)** "END"**,**

PT**.**PRICE**,**

IIF**(**PT**.**IN\_3D **=** 1**,** 'YES'**,** 'NO'**)** "3D"

**from** performanceTBL "PT"

**INNER** **JOIN** filmsTBL "FT"

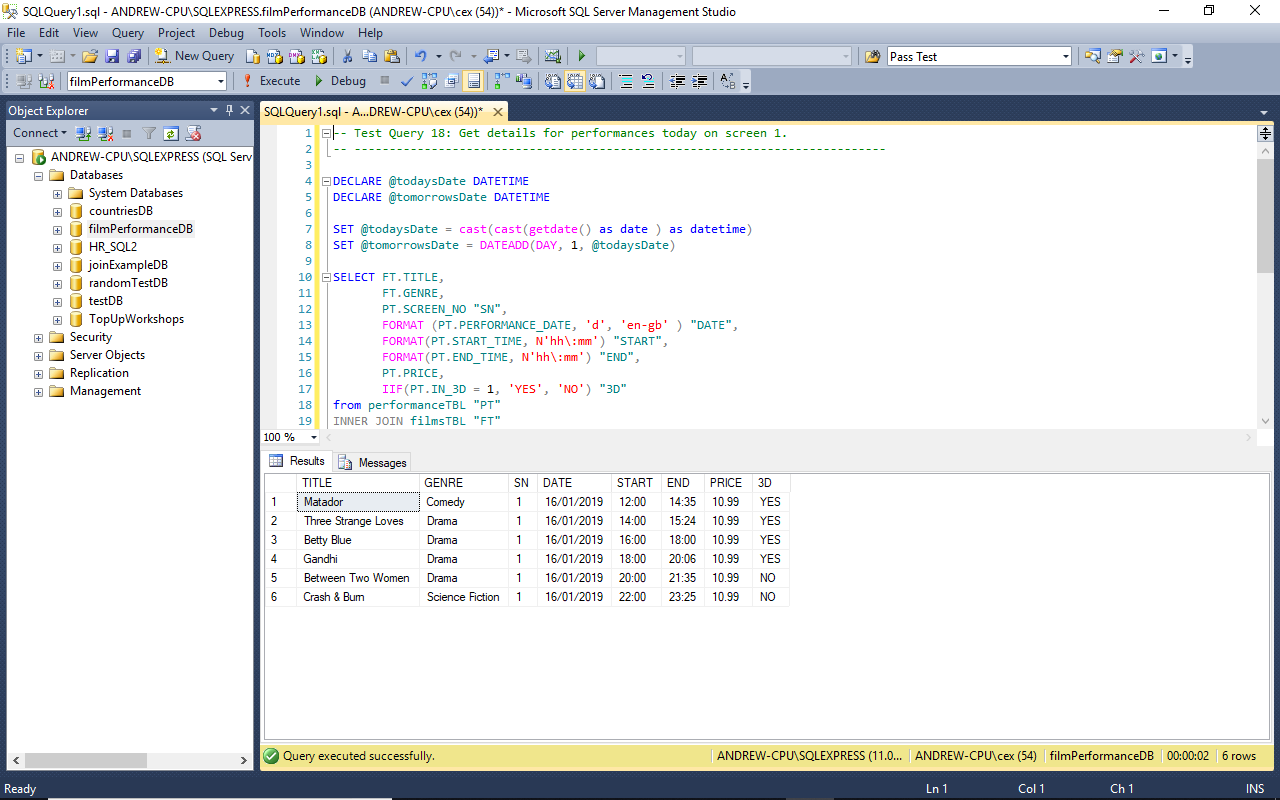
**ON** **(** PT**.**FILM\_ID**=**FT**.**FILM\_ID **)**

**WHERE** PERFORMANCE\_DATE**>=**@todaysDate

**AND** PERFORMANCE\_DATE**<**@tomorrowsDate

**AND** PT**.**SCREEN\_NO**=**1

-- ----------------------------------------------------------------------------

 *Figure 19: Screenshot of the results produced by running Test Query 18.*

# References

Fowler, M. (2018) Refactoring: Improving the Design of Existing Code. 1st ed. New York: Pearson Education.

United Kingdom Government (2006) Government Data Standards Catalogue Volume 2 – Data Types Standards, London: Cabinet Office U.K. Government.

# Bibliography

Beck, K. (2003) Test-driven Development: By Example. 1st ed. Boston: Pearson Education.

Ben-Gan, I. (2015) Microsoft SQL Server 2012 T-SQL Fundamentals. 5th ed. Redmond: Microsoft Press Books.

Churcher, C. (2013) Beginning Database Design: From Novice to Professional. 1st ed. New York: Apress.

Harrington, J. L. (2016) Relational Database Design and Implementation. 4th ed. Cambridge: Elsevier.

Haughton, D., McLaughlin, M., Mentzer, K. & Zhang, C. (2015) Movie Analytics: A Hollywood Introduction to Big Data. 1st ed. Cham: Springer.

Hernandez, M. J. (2013) Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design. 3rd ed. Boston: Addison-Wesley.

Kuznetsov, A. (2010) Defensive Database Programming with SQL Server. 1st ed. Cambridge: Red Gate Books.

Maltin, L. (2015) Leonard Maltin's 2015 Movie Guide. 1st ed. London: Penguin.

Petkovic, D. (2016) Microsoft SQL Server 2016: A Beginner's Guide. 6th ed. New York: McGraw Hill Professional.

Smith, I. H. (2008) TCM International Film Guide 2008: The Definitive Annual Review of World Cinema. 1st ed. Michigan: Wallflower Press.

# Appendix A Unit Tests for Stored Procedures and Functions

**use** filmPerformanceDB**;**

**SET** NOCOUNT **ON;** -- SET NOCOUNT HERE to avoid messages when deleting from temp tables

**DECLARE** @fails **INT;**

**SET** @fails**=**0**;**

**DECLARE** @passes **INT;**

**SET** @passes**=**0**;**

-- Used for timing the tests

**DECLARE** @**CURRENT\_TIME** **VARCHAR(**8**)**

**SET** @**CURRENT\_TIME** **=** **(** **SELECT** **CONVERT(VARCHAR(**8**),** **GETDATE(),** 108**)** 'hh:mi:ss' **);**

print 'Script started at ' **+** @**CURRENT\_TIME**

PRINT '---------------------------------------------------------------'

PRINT 'Running Unit Tests'

-- -------------------------------------------------------------------

-- Test number: 1

-- Procedure Name: insertFilm

-- -------------------------------------------------------------------

**EXECUTE** insertFilm 'La Dolce Vita'**,** 'Drama'**,** '03:12:22'**,** 'Mastroianni'**,** 'Marcello'**,**

'Aimee'**,** 'Anouk'**,** 'Fellini'**,** 'Federico'**;**

**EXECUTE** insertFilm 'Una Journata Particolare'**,** 'Drama'**,** '02:45:00'**,** 'Mastroianni'**,**

'Marcello'**,** 'Loren'**,** 'Sophia'**,** 'Scola'**,** 'Ettore'**;**

**EXECUTE** insertFilm 'La Vie est Belle'**,** 'Drama'**,** '02:32:32'**,** 'Benigni'**,** 'Roberto'**,**

'Braschi'**,** 'Nicoletta'**,** 'Benigni'**,** 'Roberto'**;**

**EXECUTE** insertFilm 'La Haine'**,** 'Drama'**,** '02:15:12'**,** 'Cassel'**,** 'Vincent'**,** 'Rauth'**,**

'Heloise'**,** 'Kassovitz'**,** 'Mathieu'**;**

**EXECUTE** insertFilm 'La Belle et La Bete'**,** 'Romance'**,** '01:54:22'**,** 'Marais'**,** 'Jean'**,**

'Day'**,** 'Josette'**,** 'Cocteau'**,** 'Jean'**;**

**EXECUTE** insertFilm 'Heat'**,** 'Police Drama'**,** '03:02:11'**,** 'Pacino'**,** 'Al'**,** 'Venora'**,**

'Diane'**,** 'Mann'**,** 'Michael'**;**

**EXECUTE** insertFilm 'From Russia with Love'**,** 'Drama'**,** '2:34:11'**,** 'Connery'**,** 'Sean'**,**

'Lenya'**,** 'Lotte'**,** 'Young'**,** 'Terence'**;**

**EXECUTE** insertFilm 'Educating Rita'**,** 'Comedy'**,** '02:55:00'**,** 'Caine'**,** 'Michael'**,**

'Walters'**,** 'Julie'**,** 'Gilbert'**,** 'Lewis'**;**

**EXECUTE** insertFilm 'His Girl Friday'**,** 'Comedy'**,** '01:34:11'**,** 'Grant'**,** 'Cary'**,**

'Russell'**,** 'Rosalind'**,** 'Hawks'**,** 'Howard'**;**

**EXECUTE** insertFilm 'L Eclisse'**,** 'Comedy'**,** '03:44:22'**,** 'Delon'**,** 'Alain'**,** 'Vitti'**,**

'Monica'**,** 'Antonioni'**,** 'Michaelangelo'**;**

**EXECUTE** insertFilm 'Lovely Donuts'**,** 'Comedy'**,** '02:12:26'**,** 'Keaton'**,** 'Bobby'**,** 'Gish'**,**

'Cheryl'**,** 'Hitchcock'**,** 'John'**;**

**EXECUTE** insertFilm 'adsfghdsad'**,** 'dsadadad'**,** '02:12:26'**,** 'qwqweq'**,** 'qwewqe'**,** 'qweqwew'**,**

'qeqe'**,** 'qeqwe'**,** 'qweqwe'**;**

**if** **((SELECT** **COUNT(**FILM\_ID**)** **FROM** filmsTBL **)=**12**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 1'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 2

-- Procedure Name: insertCustomer

-- -------------------------------------------------------------------

**EXECUTE** insertCustomer 'Randolph'**,** 'Scott'**,** '13 Pleasant Street'**,** 'Liverpool'**,**

'L19 0NE'**,** 'randyscott@gmail.com'**,** '0151345987'**;**

**EXECUTE** insertCustomer 'Judy'**,** 'Garland'**,** '22 Harlington Cresent'**,** 'Liverpool'**,**

'L24 2ED'**,** 'jgarland@gmail.com'**,** '01514448383'**;**

**EXECUTE** insertCustomer 'Romeo'**,** 'Echowhisky'**,** '16b Tarwood Terrace'**,** 'Liverpool'**,**

'L1 2LF'**,** 'recho@yahoo.com'**,** '089414567854'**;**

**EXECUTE** insertCustomer 'Sally'**,** 'Biggins'**,** '22 Plikington Place'**,** 'Liverpool'**,**

'L22 4ES'**,** 'sbiggins@gmail.com'**,** '08982241161'**;**

**EXECUTE** insertCustomer 'James'**,** 'Garner'**,** '94 Bold Street'**,** 'Liverpool'**,**

'L14 3DD'**,** 'jgarner123@gmail.com'**,** '01517775656'**;**

**EXECUTE** insertCustomer 'Christopher'**,** 'Scott'**,** '6 Alexandra Mews'**,** 'Liverpool'**,**

'L9 0ND'**,** 'chrissyscott@gmail.com'**,** '015109987784'**;**

**EXECUTE** insertCustomer 'Percival'**,** 'Morrisson'**,** '21 Everly Close'**,** 'Liverpool'**,**

'L11 4FQ'**,** 'pmoggyio@gmail.com'**,** '08982215663'**;**

**EXECUTE** insertCustomer 'Gigi'**,** 'Delacourt'**,** '96 Evergreen Terrace'**,** 'Liverpool'**,**

'L22 6TR'**,** 'gggirlie@gmail.com'**,** '01515556343'**;**

**EXECUTE** insertCustomer 'Lloyd'**,** 'Ferdinand'**,** '124 Mainbridge Street'**,** 'Liverpool'**,**

'L2 4FD'**,** 'ferdielar@gmail.com'**,** '08942278876'**;**

**EXECUTE** insertCustomer 'Mark'**,** 'Chapman'**,** '12 Leepworth Hill'**,** 'Liverpool'**,**

'L16 2WW'**,** 'chappio@gmail.com' **,** '01518989777'**;**

**EXECUTE** insertCustomer 'Paul'**,** 'Henderson'**,** '65 Impressa Grove'**,** 'Liverpool'**,**

'L2 4EE'**,** 'hennyla@gmail.com'**,** '08983267769'**;**

**EXECUTE** insertCustomer 'Ffff'**,** 'Pwwww'**,** 'uug'**,** 'ewe'**,** 'wwwwwwww'**,** 'wwwwwww'**,** 'werrw'**;**

**if** **((SELECT** **COUNT(**CUSTOMER\_ID**)** **FROM** customersTBL **)=**12**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 2'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 3

-- Procedure Name: insertScreen

-- -------------------------------------------------------------------

**EXECUTE** insertScreen 50**,**0**,**1**;**

**EXECUTE** insertScreen 150**,**1**,**1**;**

**EXECUTE** insertScreen 100**,**1**,**1**;**

**EXECUTE** insertScreen 150**,**0**,**0**;**

**EXECUTE** insertScreen 250**,**1**,**0**;**

**EXECUTE** insertScreen 200**,**0**,**0**;**

**EXECUTE** insertScreen 100**,**1**,**1**;**

**if** **((SELECT** **COUNT(**SCREEN\_NO**)** **FROM** screenTBL **)=**7**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 3'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 4

-- Procedure Name: insertPerformance

-- -------------------------------------------------------------------

**EXECUTE** insertPerformance 1**,**1**,**'2018-08-21'**,**'20:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 2**,**1**,**'2018-08-22'**,**'12:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 3**,**1**,**'2018-08-22'**,**'16:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 4**,**1**,**'2018-08-22'**,**'20:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 5**,**2**,**'2018-08-22'**,**'16:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 6**,**2**,**'2018-08-22'**,**'20:00:00'**,**12.99**,**1**;**

**EXECUTE** insertPerformance 7**,**3**,**'2018-08-22'**,**'20:30:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 8**,**1**,**'2018-08-23'**,**'16:00:00'**,**12.99**,**1**;**

**EXECUTE** insertPerformance 9**,**1**,**'2018-08-23'**,**'20:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 10**,**2**,**'2018-08-23'**,**'16:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 11**,**2**,**'2018-08-23'**,**'20:22:00'**,**12.99**,**0**;**

**if** **((SELECT** **COUNT(**PERFORMANCE\_ID**)** **FROM** performanceTBL **)=**11**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 4'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 5

-- Procedure Name: insertBooking

-- -------------------------------------------------------------------

**EXECUTE** insertBooking 1**,**2**,**2**;**

**EXECUTE** insertBooking 2**,**3**,**1**;**

**EXECUTE** insertBooking 3**,**3**,**1**;**

**EXECUTE** insertBooking 4**,**3**,**2**;**

**EXECUTE** insertBooking 5**,**4**,**2**;**

**EXECUTE** insertBooking 6**,**6**,**1**;**

**EXECUTE** insertBooking 7**,**6**,**2**;**

**EXECUTE** insertBooking 8**,**7**,**4**;**

**EXECUTE** insertBooking 9**,**2**,**2**;**

**EXECUTE** insertBooking 10**,**7**,**1**;**

**EXECUTE** insertBooking 11**,**8**,**2**;**

**if** **((SELECT** **COUNT(**BOOKING\_ID**)** **FROM** bookingsTBL **)=**11**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 5'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 6

-- Procedure Name: deleteFilm

-- -------------------------------------------------------------------

**EXECUTE** deleteFilm 12**;**

**if** **((SELECT** **COUNT(\*)**

**FROM** **(SELECT** **\*** **FROM** filmsTBL **WHERE** FILM\_ID**=**12**)** subquery**)=**0   
 -- The alias for the subquery must be used for this to work

**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 6'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 7

-- Procedure Name: deleteCustomer

-- -------------------------------------------------------------------

**EXECUTE** deleteCustomer 12**;**

**if** **((SELECT** **COUNT(\*)**

**FROM** **(SELECT** **\*** **FROM** customersTBL **WHERE** CUSTOMER\_ID**=**12**)** subquery**)=**0

**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 7'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 8

-- Procedure Name: deleteBooking

-- -------------------------------------------------------------------

**EXECUTE** deleteBooking 1**;**

**if** **((SELECT** **COUNT(\*)**

**FROM** **(SELECT** **\*** **FROM** bookingsTBL **WHERE** BOOKING\_ID**=**1**)** subquery**)=**0

**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 8'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 9

-- Procedure Name: deletePerformance

-- -------------------------------------------------------------------

**EXECUTE** deletePerformance 1**;**

**if** **((SELECT** **COUNT(\*)**

**FROM** **(SELECT** **\*** **FROM** performanceTBL **WHERE** PERFORMANCE\_ID**=**1**)** subquery**)=**0

**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 9'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 10

-- Procedure Name: updateFilm

-- -------------------------------------------------------------------

**EXECUTE** updateFilm 2**,** 'Una Journata Particolare'**,** 'Drama'**,** '03:15:00'**,** 'Mastroianni'**,**

'Marcello'**,** 'Loren'**,** 'Sophia'**,** 'Scola'**,** 'Ettore'**;**

**DECLARE** @res10 **TIME;**

**DECLARE** @res10a **TIME**

**SET** @res10 **=** '03:15:00'**;**

**SET** @res10a **=**

**(SELECT** DURATION **FROM** filmsTBL

**WHERE** FILM\_ID**=**2**);**

**if** **(**@res10**=**@res10a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 10'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 11

-- Procedure Name: updateCustomer

-- -------------------------------------------------------------------

**EXECUTE** updateCustomer 2**,** 'Judy'**,** 'Garland'**,** '22 Harlington Cresent'**,** 'Liverpool'**,**

'L24 2ED'**,** 'realjgarland@gmail.com'**,** '01514448383'**;;**

**DECLARE** @res11 **VARCHAR(**50**);**

**DECLARE** @res11a **VARCHAR(**50**);**

**SET** @res11 **=** 'realjgarland@gmail.com'**;**

**SET** @res11a **=**

**(SELECT** EMAIL **FROM** customersTBL

**WHERE** CUSTOMER\_ID**=**2**);**

**if** **(**@res11**=**@res11a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 11'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 12

-- Procedure Name: updateBooking

-- -------------------------------------------------------------------

**EXECUTE** updateBooking 2**,**4**,**3**,**1**;**

**DECLARE** @res12 **INT;**

**DECLARE** @res12a **INT;**

**SET** @res12 **=** 4**;**

**SET** @res12a **=**

**(SELECT** CUSTOMER\_ID **FROM** bookingsTBL

**WHERE** BOOKING\_ID**=**2**);**

**if** **(**@res12**=**@res12a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 12'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 13

-- Procedure Name: updatePerformance

-- -------------------------------------------------------------------

**EXECUTE** updatePerformance 2**,**2**,**1**,**'2018-08-25'**,**'12:00:00'**,**12.99**,**0**;**

**DECLARE** @res13 **DATE;**

**DECLARE** @res13a **DATE;**

**SET** @res13 **=** '2018-08-25'**;**

**SET** @res13a **=**

**(SELECT** PERFORMANCE\_DATE **FROM** performanceTBL

**WHERE** PERFORMANCE\_ID**=**2**);**

**if** **(**@res13**=**@res13a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 13'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 14

-- Procedure Name: getFilmRecordByID

-- -------------------------------------------------------------------

**DECLARE** @res14 **VARCHAR(**100**);**

**DECLARE** @res14a **VARCHAR(**100**);**

**SET** @res14 **=** 'Una Journata Particolare'**;**

-- Create a table variable to hold results.

**Declare** @TempFilm **Table** **(**

FILM\_ID **int,**

TITLE **VARCHAR(**100**)** **NOT** **NULL,**

GENRE **VARCHAR(**25**),**

DURATION **TIME** **NOT** **NULL,**

ACTOR\_SURNAME **VARCHAR(**35**),**

ACTOR\_FORENAME **VARCHAR(**35**),**

ACTRESS\_SURNAME **VARCHAR(**35**),**

ACTRESS\_FORENAME **VARCHAR(**35**),**

DIRECTOR\_SURNAME **VARCHAR(**35**),**

DIRECTOR\_FORENAME **VARCHAR(**35**)**

**)**

-- Add result set to table variable.

**Insert** @TempFilm **EXECUTE** getFilmRecordByID 2**;**

**SET** @res14a **=** **(Select** TITLE **from** @TempFilm **Where** FILM\_ID**=**2**);**

**DELETE** **FROM** @TempFilm**;** -- Clear @TempFilm table data.

**if** **(**@res14**=**@res14a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 14'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 15

-- Procedure Name: getBookingRecordByID

-- -------------------------------------------------------------------

**DECLARE** @res15 **INT;**

**DECLARE** @res15a **INT;**

**SET** @res15 **=** 4**;**

-- Create a table variable to hold results.

**Declare** @TempBookings **Table** **(**

BOOKING\_ID **INT,**

CUSTOMER\_ID **INT** **NOT** **NULL,**

PERFORMANCE\_ID **INT** **NOT** **NULL,**

NO\_OF\_TICKETS **SMALLINT** **NOT** **NULL**

**)**

-- Add result set to table variable.

**Insert** @TempBookings **EXECUTE** getBookingRecordByID 2**;**

**SET** @res15a **=** **(Select** CUSTOMER\_ID **from** @TempBookings **Where** BOOKING\_ID**=**2**);**

**DELETE** **FROM** @TempBookings**;**

**if** **(**@res15**=**@res15a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 15'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 16

-- Procedure Name: getCustomerRecordByID

-- -------------------------------------------------------------------

**DECLARE** @res16 **VARCHAR(**50**);**

**DECLARE** @res16a **VARCHAR(**50**);**

**SET** @res16 **=** 'realjgarland@gmail.com'**;**

-- Create a table variable to hold results.

**Declare** @TempCustomers **Table** **(**

CUSTOMER\_ID **int,**

FIRSTNAME **VARCHAR(**35**)** **NOT** **NULL,**

LASTNAME **VARCHAR(**35**)** **NOT** **NULL,**

ADDRESSLINE1 **VARCHAR(**50**),**

ADDRESSLINE2 **VARCHAR(**50**),**

POSTCODE **VARCHAR(**10**),**

EMAIL **VARCHAR(**50**),**

PHONE\_NO **VARCHAR(**15**)**

**)**

-- Add result set to table variable.

**Insert** @TempCustomers **EXECUTE** getCustomerRecordByID 2**;**

**SET** @res16a **=** **(Select** EMAIL **from** @TempCustomers **Where** CUSTOMER\_ID**=**2**);**

**DELETE** **FROM** @TempCustomers**;**

**if** **(**@res16**=**@res16a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 16'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 17

-- Procedure Name: getPerformanceRecordByID

-- -------------------------------------------------------------------

**DECLARE** @res17 **INT;**

**DECLARE** @res17a **INT;**

**SET** @res17 **=** 2**;**

-- Create a table variable to hold results.

**Declare** @TempPerformance **Table** **(**

PERFORMANCE\_ID **int,**

FILM\_ID **INT** **NOT** **NULL,**

SCREEN\_NO **INT** **NOT** **NULL,**

PERFORMANCE\_DATE **DATE,**

START\_TIME **TIME,**

END\_TIME **TIME,**

PRICE MONEY**,**

IN\_3D **BIT**

**)**

-- Add result set to table variable.

**Insert** @TempPerformance **EXECUTE** getPerformanceRecordByID 2**;**

**SET** @res17a **=** **(Select** FILM\_ID **from** @TempPerformance **Where** PERFORMANCE\_ID**=**2**);**

**DELETE** **FROM** @TempPerformance**;**

**if** **(**@res17**=**@res17a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 17'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 18

-- Procedure Name: getPerformanceDetailsByDateAndStartTime

-- -------------------------------------------------------------------

**DECLARE** @res18 **VARCHAR(**100**);**

**DECLARE** @res18a **VARCHAR(**100**);**

**SET** @res18 **=** 'Una Journata Particolare'**;**

-- Create a table variable to hold results.

**Declare** @TempPDDST **Table** **(**

TITLE **VARCHAR(**100**)** **NOT** **NULL,**

GENRE **VARCHAR(**25**),**

SCREEN\_NO **INT** **NOT** **NULL,**

PERFORMANCE\_DATE **VARCHAR(**20**),**

START\_TIME **TIME,**

END\_TIME **TIME,**

PRICE MONEY**,**

IN\_3D **BIT**

**)**

-- Add result set to table variable.

**Insert** @TempPDDST **EXECUTE** getPerformanceDetailsByDateAndStartTime '2018-08-25'**,**'12:00:00'**,**'00:00:00'**;**

**SET** @res18a **=** **(Select** TITLE **from** @TempPDDST **WHERE** START\_TIME**=**'12:00:00'**);**

**DELETE** **FROM** @TempPDDST**;**

**if** **(**@res18**=**@res18a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 18'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 19

-- Procedure Name: getPerformanceDetailsByDateAndStartTime

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempPDDST **EXECUTE** getPerformanceDetailsByDateAndStartTime '2018-08-22'**,**'20:00:00'**,**'00:30:00'**;**

**if** **((** **SELECT** **COUNT(**TITLE**)** **FROM** @TempPDDST**)=**3**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 19'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempPDDST**;**

-- -------------------------------------------------------------------

-- Test number: 20

-- Procedure Name: getPerformanceDetailsByDateAndStartTime

-- -------------------------------------------------------------------

-- Add result set to table variable

**Insert** @TempPDDST **EXECUTE** getPerformanceDetailsByDateAndStartTime '2018-08-23'**,**'20:00:00'**,**'00:30:00'**;**

**if** **((** **SELECT** **COUNT(**TITLE**)** **FROM** @TempPDDST**)=**2**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 20'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempPDDST**;**

-- -------------------------------------------------------------------

-- Test number: 21

-- Procedure Name: getPerformanceDetailsByDateAndStartTime

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempPDDST **EXECUTE** getPerformanceDetailsByDateAndStartTime '2018-08-23'**,**'20:00:00'**,**'00:30'**;**

**if** **((** **SELECT** **COUNT(**TITLE**)** **FROM** @TempPDDST**)=**2**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 21'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempPDDST**;**

-- -------------------------------------------------------------------

-- Test number: 22

-- Procedure Name: getPerformanceDetailsByDateAndStartTime

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempPDDST **EXECUTE** getPerformanceDetailsByDateAndStartTime '2018-08-25'**,**'21:00:00'**,**'00:30:00'**;**

**if** **((** **SELECT** **COUNT(**TITLE**)** **FROM** @TempPDDST**)=**0**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 22'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempPDDST**;**

-- -------------------------------------------------------------------

-- Test number: 23

-- Procedure Name: getPerformanceDetailsByFilmTitle

-- -------------------------------------------------------------------

-- Add result set to table variable

**Insert** @TempPDDST **EXECUTE** getPerformanceDetailsByFilmTitle 'Lovely Donuts'**;**

**if** **((** **SELECT** **COUNT(**TITLE**)** **FROM** @TempPDDST**)=**1**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 23'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempPDDST**;**

-- -------------------------------------------------------------------

-- Test number: 24

-- Procedure Name: getAll3DPerformanceDetails;

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempPDDST **EXECUTE** getAll3DPerformanceDetails**;**

**if** **((** **SELECT** **COUNT(**TITLE**)** **FROM** @TempPDDST**)=**2**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 24'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempPDDST**;**

-- -------------------------------------------------------------------

-- Test number: 25

-- Procedure Name: get3DPerformanceDetailsByDateAndStartTime

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempPDDST **EXECUTE** get3DPerformanceDetailsByDateAndStartTime '2018-08-22'**,**'20:00:00'**,**'00:30:00'**;**

**if** **((** **SELECT** **COUNT(**TITLE**)** **FROM** @TempPDDST**)=**1**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 25'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempPDDST**;**

-- -------------------------------------------------------------------

-- Test number: 26

-- Procedure Name: getFilmRecordsByDuration

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempFRD **Table** **(**

TITLE **VARCHAR(**100**)** **NOT** **NULL,**

GENRE **VARCHAR(**25**),**

DURATION **VARCHAR(**20**)**

**)**

-- Add result set to table variable.

**Insert** @TempFRD **EXECUTE** getFilmRecordsByDuration '00:45'**,**'02:30'**;**

**if** **((** **SELECT** **COUNT(**TITLE**)** **FROM** @TempFRD**)=**4**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 26'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempFRD**;**

-- -------------------------------------------------------------------

-- Test number: 27

-- Procedure Name: getFilmRecordsByGenre

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempFRD **EXECUTE** getFilmRecordsByGenre 'Comedy'**;**

**if** **((** **SELECT** **COUNT(**TITLE**)** **FROM** @TempFRD**)=**4**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 27'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempFRD**;**

-- -------------------------------------------------------------------

-- Test number: 28

-- Function Name: getAllFilmGenres

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempTitles **Table** **(**

TITLE **VARCHAR(**100**)** **NOT** **NULL**

**)**

**Insert** @TempTitles **SELECT** **\*** **FROM** dbo**.**getAllFilmGenres**();**

**if** **((** **SELECT** **COUNT(**TITLE**)** **FROM** @TempTitles**)=**4**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 28'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTitles**;**

-- -------------------------------------------------------------------

-- Test number: 29

-- Function Name: getAllFilmTitles

-- -------------------------------------------------------------------

**Insert** @TempTitles **SELECT** **\*** **FROM** dbo**.**getAllFilmTitles**();**

**if** **((** **SELECT** **COUNT(**TITLE**)** **FROM** @TempTitles**)=**11**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 29'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTitles**;**

-- -------------------------------------------------------------------

-- Test number: 30

-- Procedure Name: getCustomerRecordsByEmail

-- -------------------------------------------------------------------

**DECLARE** @res30 **INT;**

**DECLARE** @res30a **INT;**

**SET** @res30 **=** 4**;**

-- Add result set to table variable.

**Insert** @TempCustomers **EXECUTE** getCustomerRecordsByEmail 'sbiggins@gmail.com'**;**

**SET** @res30a **=** **(Select** CUSTOMER\_ID **from** @TempCustomers **Where** EMAIL**=**'sbiggins@gmail.com'**);**

**DELETE** **FROM** @TempCustomers**;**

**if** **(**@res30**=**@res30a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 30'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 31

-- Procedure Name: getCustomerRecordsByPostcode

-- -------------------------------------------------------------------

**DECLARE** @res31 **INT;**

**DECLARE** @res31a **INT;**

**SET** @res31 **=** 4**;**

-- Add result set to table variable.

**Insert** @TempCustomers **EXECUTE** getCustomerRecordsByPostcode 'L22 4ES'**;**

**SET** @res31a **=** **(Select** CUSTOMER\_ID **from** @TempCustomers **Where** POSTCODE**=**'L22 4ES'**);**

**DELETE** **FROM** @TempCustomers**;**

**if** **(**@res31**=**@res31a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 31'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 32

-- Procedure Name: getBookingRecordsByCustomerID

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempBookings **EXECUTE** getBookingRecordsByCustomerID 7**;**

**if** **((** **SELECT** **COUNT(**BOOKING\_ID**)** **FROM** @TempBookings**)=**1**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 32'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempBookings**;**

-- -------------------------------------------------------------------

-- Test number: 33

-- Procedure Name: getBookingRecordsForDate

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempBookings **EXECUTE** getBookingRecordsForDate '2018-08-23'**;**

**if** **((** **SELECT** **COUNT(**BOOKING\_ID**)** **FROM** @TempBookings**)=**1**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 33'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempBookings**;**

-- -------------------------------------------------------------------

-- Test number: 34

-- Procedure Name: getBookingRecordsByFilmID

-- -------------------------------------------------------------------

**Insert** @TempBookings **EXECUTE** getBookingRecordsByFilmID 3**;**

**if** **((** **SELECT** **COUNT(**BOOKING\_ID**)** **FROM** @TempBookings**)=**3**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 34'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempBookings**;**

-- -------------------------------------------------------------------

-- Test number: 35

-- Procedure Name: getBookingRecordsByPerformanceID

-- -------------------------------------------------------------------

**Insert** @TempBookings **EXECUTE** getBookingRecordsByPerformanceID 7**;**

**if** **((** **SELECT** **COUNT(**BOOKING\_ID**)** **FROM** @TempBookings**)=**2**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 35'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempBookings**;**

-- -------------------------------------------------------------------

-- Test number: 36

-- Procedure Name: getFilmIDsByTitle

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempFIBT **Table** **(**

FILM\_ID **int,**

TITLE **VARCHAR(**100**)** **NOT** **NULL**

**)**

**Insert** @TempFIBT **EXECUTE** getFilmIDsByTitle 'La'**;**

**if** **((** **SELECT** **COUNT(**FILM\_ID**)** **FROM** @TempFIBT**)=**4**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 36'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 37

-- Procedure Name: getTotalNoOfTicketsBookedByCustomerID

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByCustomerID**(**4**))=**3**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 37'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 38

-- Procedure Name: getTotalNoOfTicketsBookedByPerformanceID

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByPerformanceID**(**3**))=**4**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 38'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 39

-- Procedure Name: getTotalNoOfTicketsBookedByFilmID

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByFilmID**(**3**))=**4**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 39'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 40

-- Procedure Name: getTotalNoOfTicketsBookedByGenre

-- -------------------------------------------------------------------

**if** **((**dbo**.**getTotalNoOfTicketsBookedByGenre**(**'Drama'**))=**13**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 40'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 41

-- Procedure Name: getTotalNoOfTicketsBookedByFilmTitle

-- -------------------------------------------------------------------

**if** **((**dbo**.**getTotalNoOfTicketsBookedByFilmTitle**(**'Heat'**))=**3**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 41'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 42

-- Procedure Name: getTotalNoOfTicketsBookedInPeriod

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedInPeriod**(**'2018-08-21'**,** '2018-08-23'**))=**16**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 42'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 43

-- Procedure Name: getTotalNoOfTicketsBookedOnDate

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedOnDate**(**'2018-08-22'**))=**14**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 43'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 44

-- Procedure Name: getTotalNoOfTicketsBookedByScreenNo

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByScreenNo**(**2**))=**3**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 44'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 45

-- Procedure Name: getTopNFilmBookingsOfAllTime

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempTopNFBAT **Table** **(**

FILM\_ID **INT,**

TITLE **VARCHAR(**100**)** **NOT** **NULL,**

GENRE **VARCHAR(**25**),**

DURATION **VARCHAR(**20**),**

TOTAL\_BOOKINGS **INT**

**)**

-- Add result set to table variable

**Insert** @TempTopNFBAT **EXECUTE** getTopNFilmBookingsOfAllTime 5**;**

**if** **((** **SELECT** **SUM(**TOTAL\_BOOKINGS**)** **FROM** @TempTopNFBAT**)=**16**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 45'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNFBAT**;**

-- -------------------------------------------------------------------

-- Test number: 46

-- Procedure Name: getTopNBookingCustomersOfAllTime

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempTopNBCAT **Table** **(**

CUSTOMER\_ID **int,**

FIRSTNAME **VARCHAR(**35**)** **NOT** **NULL,**

LASTNAME **VARCHAR(**35**)** **NOT** **NULL,**

EMAIL **VARCHAR(**50**),**

POSTCODE **VARCHAR(**10**),**

TOTAL\_BOOKINGS **INT**

**)**

-- Add result set to table variable

**Insert** @TempTopNBCAT **EXECUTE** getTopNBookingCustomersOfAllTime 5**;**

**if** **((** **SELECT** **SUM(**TOTAL\_BOOKINGS**)** **FROM** @TempTopNBCAT**)=**13**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 46'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNBCAT**;**

-- -------------------------------------------------------------------

-- Test number: 47

-- Procedure Name: getTopNGenreBookingsOfAllTime

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempTopGBAT **Table** **(**

GENRE **VARCHAR(**25**),**

TOTAL\_BOOKINGS **INT**

**)**

-- Add result set to table variable.

**Insert** @TempTopGBAT **EXECUTE** getTopNGenreBookingsOfAllTime 5**;**

**if** **((** **SELECT** **SUM(**TOTAL\_BOOKINGS**)** **FROM** @TempTopGBAT**)=**18**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 47'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopGBAT**;**

-- -------------------------------------------------------------------

-- Test number: 48

-- Procedure Name: getTopNScreenBookingsOfAllTime

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempTopNSBAT **Table** **(**

SCREEN\_NO **INT,**

TOTAL\_BOOKINGS **INT**

**)**

-- Add result set to table variable.

**Insert** @TempTopNSBAT **EXECUTE** getTopNScreenBookingsOfAllTime 5**;**

**if** **((SELECT** TOTAL\_BOOKINGS **FROM** @TempTopNSBAT **WHERE** SCREEN\_NO**=**1**)=**10**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 48'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNSBAT**;**

-- -------------------------------------------------------------------

-- Test number: 49

-- Procedure Name: getTotalNoOfTicketsBookedByCustomerIDDuringPeriod

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByCustomerIDDuringPeriod**(**4**,** '2018-08-21'**,** '2018-08-23'**))=**3**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 49'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 50

-- Procedure Name: getTotalNoOfTicketsBookedByFilmIDDuringPeriod

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByFilmIDDuringPeriod**(**7**,** '2018-08-21'**,** '2018-08-23'**))=**5**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 50'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 51

-- Procedure Name: getTotalNoOfTicketsBookedByGenreDuringPeriod

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByGenreDuringPeriod**(**'Comedy'**,** '2018-08-21'**,** '2018-08-23'**))=**2**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 51'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 52

-- Procedure Name: getTotalNoOfTicketsBookedByScreenNoDuringPeriod

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByScreenNoDuringPeriod**(**3**,** '2018-08-21'**,** '2018-08-23'**))=**5**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 52'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 53

-- Procedure Name: getTopNFilmBookingsDuringPeriod

-- -------------------------------------------------------------------

-- Add result set to table variable

**Insert** @TempTopNFBAT **EXECUTE** getTopNFilmBookingsDuringPeriod 5**,** '2018-08-21'**,** '2018-08-23'**;**

**if** **((SELECT** TOTAL\_BOOKINGS **FROM** @TempTopNFBAT **WHERE** FILM\_ID**=**7**)=**5**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 53'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNFBAT**;**

-- -------------------------------------------------------------------

-- Test number: 54

-- Procedure Name: getTopNBookingCustomersDuringPeriod

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempTopNBCAT **EXECUTE** getTopNBookingCustomersDuringPeriod 5**,** '2018-08-21'**,** '2018-08-23'**;**

**if** **((SELECT** TOTAL\_BOOKINGS **FROM** @TempTopNBCAT **WHERE** CUSTOMER\_ID**=**8**)=**4**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 54'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNBCAT**;**

-- -------------------------------------------------------------------

-- Test number: 55

-- Procedure Name: getTopNGenreBookingsDuringPeriod

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempTopGBAT **EXECUTE** getTopNGenreBookingsDuringPeriod 5**,** '2018-08-21'**,** '2018-08-23'**;**

**if** **((SELECT** TOTAL\_BOOKINGS **FROM** @TempTopGBAT **WHERE** GENRE **LIKE** 'Drama'**)=**11**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 55'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopGBAT**;**

-- -------------------------------------------------------------------

-- Test number: 56

-- Procedure Name: getTopNScreenBookingsDuringPeriod

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempTopNSBAT **EXECUTE** getTopNScreenBookingsDuringPeriod 5**,** '2018-08-21'**,** '2018-08-23'**;**

**if** **((SELECT** TOTAL\_BOOKINGS **FROM** @TempTopNSBAT **WHERE** SCREEN\_NO**=**1**)=**8**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 56'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNSBAT**;**

-- -------------------------------------------------------------------

-- Test number: 57

-- Function Name: getAllFilmIDs

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempAllIDs **Table** **(**

ID **INT**

**)**

-- Add result set to table variable.

**Insert** @TempAllIDs **SELECT** **\*** **FROM** dbo**.**getAllFilmIDs**();**

**if** **((SELECT** **COUNT(**ID**)** **FROM** @TempAllIDs**)=**11**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 57'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempAllIDs**;**

-- -------------------------------------------------------------------

-- Test number: 58

-- Function Name: getAllBookingIDs

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempAllIDs **SELECT** **\*** **FROM** dbo**.**getAllBookingIDs**();**

**if** **((SELECT** **COUNT(**ID**)** **FROM** @TempAllIDs**)=**10**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 58'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempAllIDs**;**

-- -------------------------------------------------------------------

-- Test number: 59

-- Function Name: getAllCustomerIDs

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempAllIDs **SELECT** **\*** **FROM** dbo**.**getAllCustomerIDs**();**

**if** **((SELECT** **COUNT(**ID**)** **FROM** @TempAllIDs**)=**11**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 59'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempAllIDs**;**

-- -------------------------------------------------------------------

-- Test number: 60

-- Function Name: getAllPerformanceIDs

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempAllIDs **SELECT** **\*** **FROM** dbo**.**getAllPerformanceIDs**();**

**if** **((SELECT** **COUNT(**ID**)** **FROM** @TempAllIDs**)=**10**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 60'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempAllIDs**;**

-- -------------------------------------------------------------------

-- Test number: 61

-- Function Name: getAll3DPerformanceIDs

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempAllIDs **SELECT** **\*** **FROM** dbo**.**getAll3DPerformanceIDs**();**

**if** **((SELECT** **COUNT(**ID**)** **FROM** @TempAllIDs**)=**2**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 61'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempAllIDs**;**

-- -------------------------------------------------------------------

-- Test number: 62

-- Procedure Name: getCustomersWithNoBookings

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempCNB **Table** **(**

CUSTOMER\_ID **int,**

FIRSTNAME **VARCHAR(**35**)** **NOT** **NULL,**

LASTNAME **VARCHAR(**35**)** **NOT** **NULL,**

EMAIL **VARCHAR(**50**),**

POSTCODE **VARCHAR(**10**)**

**)**

-- Add result set to table variable.

**Insert** @TempCNB **EXECUTE** getCustomersWithNoBookings**;**

**if** **((** **SELECT** **COUNT(**CUSTOMER\_ID**)** **FROM** @TempCNB**)=**2**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 62'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempCNB**;**

-- -------------------------------------------------------------------

-- Test number: 63

-- Procedure Name: getCustomersWithNoBookingsDuringPeriod

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempCNB **EXECUTE** getCustomersWithNoBookingsDuringPeriod '2018-08-21'**,**'2018-08-22'**;**

**if** **((** **SELECT** **COUNT(**CUSTOMER\_ID**)** **FROM** @TempCNB**)=**4**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 63'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempCNB**;**

-- -------------------------------------------------------------------

-- Test number: 64

-- Procedure Name: getTotalNoOfTicketsBooked

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBooked**())=**18**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 64'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- ----------------------------------------------------

-- Insert performance and booking records for yesterday

-- ----------------------------------------------------

**DECLARE** @todaysDate **DATETIME**

**DECLARE** @yesterdaysDate **DATETIME**

**SET** @todaysDate **=** **cast(cast(getdate()** **as** **date** **)** **as** **datetime)**

-- Get the start of yesterday by adding -1 day to today.

**SET** @yesterdaysDate **=** **DATEADD(DAY,** -1**,** @todaysDate**)**

-- @FILM\_ID, @SCREEN\_NO, @PERFORMANCE\_DATE, @START\_TIME, @PRICE

**EXECUTE** insertPerformance 5**,** 2**,** @yesterdaysDate**,**'12:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 2**,** 2**,** @yesterdaysDate**,**'16:00:00'**,**12.99**,**1**;**

**EXECUTE** insertPerformance 7**,** 1**,** @yesterdaysDate**,**'20:00:00'**,**12.99**,**0**;**

-- @CUSTOMER\_ID, @PERFORMANCE\_ID, @NO\_OF\_TICKETS

**EXECUTE** insertBooking 1**,**13**,**2**;**

**EXECUTE** insertBooking 4**,**13**,**5**;**

**EXECUTE** insertBooking 1**,**12**,**1**;**

**EXECUTE** insertBooking 4**,**14**,**7**;**

**EXECUTE** insertBooking 1**,**12**,**3**;**

**EXECUTE** insertBooking 4**,**12**,**2**;**

-- -------------------------------------------------------------------

-- Test number: 65

-- Procedure Name: getTotalNoOfTicketsBookedDuringPreviousWeek

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedDuringPreviousWeek**())=**20**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 65'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 66

-- Procedure Name: getTotalNoOfTicketsBookedByCustomerDuringPreviousWeek

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByCustomerDuringPreviousWeek**(**4**))=**14**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 66'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 67

-- Procedure Name: getTotalNoOfTicketsBookedByFilmIDDuringPreviousWeek

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByFilmIDDuringPreviousWeek**(**5**))=**6**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 67'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 68

-- Procedure Name: getTotalNoOfTicketsBookedByGenreDuringPreviousWeek

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByGenreDuringPreviousWeek**(**'Romance'**))=**6**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 68'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 69

-- Procedure Name: getTotalNoOfTicketsBookedByScreenNoDuringPreviousWeek

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByScreenNoDuringPreviousWeek**(**2**))=**13**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 69'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 70

-- Procedure Name: getTopNFilmBookingsDuringPreviousWeek

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempTopNFBAT **EXECUTE** getTopNFilmBookingsDuringPreviousWeek 5**;**

**if** **((SELECT** TOTAL\_BOOKINGS **FROM** @TempTopNFBAT **WHERE** FILM\_ID**=**7**)=**7**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 70'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNFBAT**;**

-- -------------------------------------------------------------------

-- Test number: 71

-- Procedure Name: getTopNBookingCustomersDuringPreviousWeek

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempTopNBCAT **EXECUTE** getTopNBookingCustomersDuringPreviousWeek 5**;**

**if** **((SELECT** TOTAL\_BOOKINGS **FROM** @TempTopNBCAT **WHERE** CUSTOMER\_ID**=**4**)=**14**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 71'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNBCAT**;**

-- -------------------------------------------------------------------

-- Test number: 72

-- Procedure Name: getTopNGenreBookingsDuringPreviousWeek

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempTopGBAT **EXECUTE** getTopNGenreBookingsDuringPreviousWeek 5**;**

**if** **((SELECT** TOTAL\_BOOKINGS **FROM** @TempTopGBAT **WHERE** GENRE **LIKE** 'Drama'**)=**14**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 72'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopGBAT**;**

-- -------------------------------------------------------------------

-- Test number: 73

-- Procedure Name: getTopNScreenBookingsDuringPreviousWeek

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempTopNSBAT **EXECUTE** getTopNScreenBookingsDuringPreviousWeek 5**;**

**if** **((SELECT** TOTAL\_BOOKINGS **FROM** @TempTopNSBAT **WHERE** SCREEN\_NO**=**2**)=**13**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 73'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNSBAT**;**

-- -------------------------------------------------------------------

-- Test number: 74

-- Procedure Name: getPerformanceDetailsForDate

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempPDFD **Table** **(**

TITLE **VARCHAR(**100**)** **NOT** **NULL,**

GENRE **VARCHAR(**25**),**

SCREEN\_NO **INT** **NOT** **NULL,**

PERFORMANCE\_DATE **VARCHAR(**20**),**

START\_TIME **TIME,**

END\_TIME **TIME,**

PRICE MONEY**,**

IN\_3D **BIT**

**)**

-- Add result set to table variable.

**Insert** @TempPDFD **EXECUTE** getPerformanceDetailsForDate '2018-08-22'**;**

**if** **((SELECT** **COUNT(**TITLE**)** **FROM** @TempPDFD**)=**5**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 74'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempPDFD**;**

-- -------------------------------------------------------------------

-- Test number: 75

-- Procedure Name: getBookingRecordsForPeriod

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempBookings **EXECUTE** getBookingRecordsForPeriod '2018-08-23'**,**'2018-08-25'**;**

**if** **((SELECT** **COUNT(**BOOKING\_ID**)** **FROM** @TempBookings**)=**2**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 75'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempBookings**;**

-- ------------------------------------------------

-- Insert performance and booking records for today

-- ------------------------------------------------

-- @FILM\_ID, @SCREEN\_NO, @PERFORMANCE\_DATE, @START\_TIME, @PRICE

**EXECUTE** insertPerformance 3**,**1**,**@todaysDate**,**'12:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 4**,**3**,**@todaysDate**,**'16:00:00'**,**12.99**,**1**;**

**EXECUTE** insertPerformance 4**,**3**,**@todaysDate**,**'20:00:00'**,**12.99**,**0**;**

-- @CUSTOMER\_ID, @PERFORMANCE\_ID, @NO\_OF\_TICKETS

**EXECUTE** insertBooking 2**,**15**,**2**;**

**EXECUTE** insertBooking 3**,**15**,**1**;**

**EXECUTE** insertBooking 1**,**15**,**1**;**

**EXECUTE** insertBooking 2**,**16**,**3**;**

**EXECUTE** insertBooking 1**,**16**,**2**;**

**EXECUTE** insertBooking 6**,**17**,**2**;**

-- -------------------------------------------------------------------

-- Test number: 76

-- Procedure Name: getBookingRecordsForToday

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempBookings **EXECUTE** getBookingRecordsForToday**;**

**if** **((** **SELECT** **COUNT(**BOOKING\_ID**)** **FROM** @TempBookings**)=**6**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 76'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempBookings**;**

-- -------------------------------------------------------------------

-- Test number: 77

-- Procedure Name: getPerformanceDetailsForToday

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempPDFD **EXECUTE** getPerformanceDetailsForToday**;**

**if** **((SELECT** **COUNT(**TITLE**)** **FROM** @TempPDFD**)=**3**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 77'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempPDFD**;**

-- ----------------------------------------------------

-- Insert performance and booking records for yesterday

-- ----------------------------------------------------

**DECLARE** @tomorrowsDate **DATETIME**

**SET** @tomorrowsDate **=** **cast(DATEADD(DAY,** 1**,** @todaysDate**)** **as** **datetime)**

-- @FILM\_ID, @SCREEN\_NO, @PERFORMANCE\_DATE, @START\_TIME, @PRICE

**EXECUTE** insertPerformance 1**,**1**,**@tomorrowsDate**,**'20:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 2**,**1**,**@tomorrowsDate**,**'12:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 5**,**2**,**@tomorrowsDate**,**'16:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 7**,**3**,**@tomorrowsDate**,**'20:30:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 3**,**2**,**@tomorrowsDate**,**'20:00:00'**,**12.99**,**0**;**

**EXECUTE** insertPerformance 6**,**1**,**@tomorrowsDate**,**'16:00:00'**,**12.99**,**0**;**

-- @CUSTOMER\_ID, @PERFORMANCE\_ID, @NO\_OF\_TICKETS

**EXECUTE** insertBooking 1**,**18**,**3**;**

**EXECUTE** insertBooking 3**,**18**,**2**;**

**EXECUTE** insertBooking 5**,**19**,**5**;**

**EXECUTE** insertBooking 4**,**19**,**3**;**

**EXECUTE** insertBooking 1**,**21**,**2**;**

**EXECUTE** insertBooking 6**,**22**,**2**;**

-- -------------------------------------------------------------------

-- Test number: 78

-- Procedure Name: getBookingRecordsForTomorrow

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempBookings **EXECUTE** getBookingRecordsForTomorrow**;**

**if** **((** **SELECT** **COUNT(**BOOKING\_ID**)** **FROM** @TempBookings**)=**6**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 78'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempBookings**;**

-- -------------------------------------------------------------------

-- Test number: 79

-- Procedure Name: getPerformanceDetailsForTomorrow

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempPDFD **EXECUTE** getPerformanceDetailsForTomorrow**;**

**if** **((SELECT** **COUNT(**TITLE**)** **FROM** @TempPDFD**)=**6**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 79'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempPDFD**;**

-- -------------------------------------------------------------------

-- Test number: 80

-- Procedure Name: getCustomerRecordsByPhoneNo

-- -------------------------------------------------------------------

**DECLARE** @res80 **INT;**

**DECLARE** @res80a **INT;**

**SET** @res80 **=** 2**;**

-- Add result set to table variable.

**Insert** @TempCustomers **EXECUTE** getCustomerRecordsByPhoneNo '01514448383'**;**

**SET** @res80a **=** **(Select** CUSTOMER\_ID **from** @TempCustomers **Where** PHONE\_NO**=**'01514448383'**);**

**DELETE** **FROM** @TempCustomers**;**

**if** **(**@res80**=**@res80a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 80'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 81

-- Procedure Name: getFilmRecordsByDirectorSurname

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempFRBDS **Table** **(**

SURNAME **VARCHAR(**35**),**

FORENAME **VARCHAR(**35**),**

TITLE **VARCHAR(**100**)** **NOT** **NULL,**

GENRE **VARCHAR(**25**),**

DURATION **VARCHAR(**20**)**

**)**

-- Add result set to table variable.

**Insert** @TempFRBDS **EXECUTE** getFilmRecordsByDirectorSurname 'Hitchcock'**;**

**if** **((SELECT** **COUNT(**TITLE**)** **FROM** @TempFRBDS**)=**1**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 81'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempFRBDS**;**

-- -------------------------------------------------------------------

-- Test number: 82

-- Procedure Name: getFilmRecordsByActorSurname

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempFRBDS **EXECUTE** getFilmRecordsByActorSurname 'Mastroianni'**;**

**if** **((SELECT** **COUNT(**TITLE**)** **FROM** @TempFRBDS**)=**2**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 82'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempFRBDS**;**

-- -------------------------------------------------------------------

-- Test number: 83

-- Procedure Name: getFilmRecordsByActressSurname

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempFRBDS **EXECUTE** getFilmRecordsByActressSurname 'Loren'**;**

**if** **((SELECT** **COUNT(**TITLE**)** **FROM** @TempFRBDS**)=**1**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 83'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempFRBDS**;**

-- -------------------------------------------------------------------

-- Test number: 84

-- Procedure Name: getAllActorNames

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempGAAN **Table** **(**

SURNAME **VARCHAR(**35**),**

FORENAME **VARCHAR(**35**)**

**)**

-- Add result set to table variable.

**Insert** @TempGAAN **SELECT** **\*** **FROM** dbo**.**getAllActorNames**();**

**if** **((SELECT** **COUNT(**SURNAME**)** **FROM** @TempGAAN**)=**10**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 84'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempGAAN**;**

-- -------------------------------------------------------------------

-- Test number: 85

-- Procedure Name: getAllActressNames

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempGAAN **SELECT** **\*** **FROM** dbo**.**getAllActressNames**();**

**if** **((SELECT** **COUNT(**SURNAME**)** **FROM** @TempGAAN**)=**11**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 85'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempGAAN**;**

-- -------------------------------------------------------------------

-- Test number: 86

-- Procedure Name: getAllDirectorNames

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempGAAN **SELECT** **\*** **FROM** dbo**.**getAllDirectorNames**();**

**if** **((SELECT** **COUNT(**SURNAME**)** **FROM** @TempGAAN**)=**11**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 86'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempGAAN**;**

-- -------------------------------------------------------------------

-- Test number: 87

-- Procedure Name: getTotalNoOfTicketsBookedByActorName

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByActorName**(**'Marcello'**,**'Mastroianni'**))=**22**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 87'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 88

-- Procedure Name: getTotalNoOfTicketsBookedByActressName

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByActressName**(**'Anouk'**,**'Aimee'**))=**5**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 88'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 89

-- Procedure Name: getTotalNoOfTicketsBookedByDirectorName

-- -------------------------------------------------------------------

**if** **((SELECT** dbo**.**getTotalNoOfTicketsBookedByDirectorName**(**'Federico'**,**'Fellini'**))=**5**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 89'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 90

-- Procedure Name: getTopNDirectorBookingsOfAllTime

-- -------------------------------------------------------------------

-- Create a table variable to hold results.

**Declare** @TempTopNDBAT **Table** **(**

SURNAME **VARCHAR(**35**),**

FORENAME **VARCHAR(**35**),**

TOTAL\_BOOKINGS **INT**

**)**

-- Add result set to table variable.

**Insert** @TempTopNDBAT **EXECUTE** getTopNDirectorBookingsOfAllTime 5**;**

**if** **((** **SELECT** **SUM(**TOTAL\_BOOKINGS**)** **FROM** @TempTopNDBAT**)=**56**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 90'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNDBAT**;**

-- -------------------------------------------------------------------

-- Test number: 91

-- Procedure Name: getTopNActorBookingsOfAllTime

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempTopNDBAT **EXECUTE** getTopNActorBookingsOfAllTime 5**;**

**if** **((** **SELECT** **SUM(**TOTAL\_BOOKINGS**)** **FROM** @TempTopNDBAT**)=**61**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 91'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNDBAT**;**

-- -------------------------------------------------------------------

-- Test number: 92

-- Procedure Name: getTopNActressBookingsOfAllTime

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempTopNDBAT **EXECUTE** getTopNActressBookingsOfAllTime 5**;**

**if** **((** **SELECT** **SUM(**TOTAL\_BOOKINGS**)** **FROM** @TempTopNDBAT**)=**56**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 92'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempTopNDBAT**;**

-- -------------------------------------------------------------------

-- Test number: 93

-- Procedure Name: updateScreen

-- -------------------------------------------------------------------

**EXECUTE** updateScreen 1**,** 200**,** 0**,** 1**;**

**DECLARE** @res93 **SMALLINT;**

**DECLARE** @res93a **SMALLINT**

**SET** @res93 **=** 200**;**

**SET** @res93a **=** **(SELECT** CAPACITY **FROM** screenTBL **WHERE** SCREEN\_NO**=**1**);**

**if** **(**@res93**=**@res93a**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 93'**;**

**SET** @fails **=** @fails **+** 1

**END**

-- -------------------------------------------------------------------

-- Test number: 94

-- Procedure Name: deleteScreen

-- -------------------------------------------------------------------

**EXECUTE** deleteScreen 3**;**

-- Add result set to table variable.

**Insert** @TempAllIDs **SELECT** SCREEN\_NO **from** screenTBL**;**

**if** **((SELECT** **COUNT(**ID**)** **FROM** @TempAllIDs**)=**7**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 94'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempAllIDs**;**

-- -------------------------------------------------------------------

-- Test number: 95

-- Procedure Name: getAllScreenNumbers

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempAllIDs **SELECT** **\*** **FROM** dbo**.**getAllScreenNumbers**();**

**if** **((SELECT** **COUNT(**ID**)** **FROM** @TempAllIDs**)=**7**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 95'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempAllIDs**;**

-- -------------------------------------------------------------------

-- Test number: 96

-- Procedure Name: getAllScreenDetails

-- -------------------------------------------------------------------

**Declare** @TempScreen **Table** **(**

SCREEN\_NO **int,**

CAPACITY **SMALLINT,**

HAS\_DISABLED\_ACCESS **BIT,**

HAS\_HEARING\_LOOP **BIT**

**);**

-- Add result set to table variable.

**Insert** @TempScreen **EXECUTE** getAllScreenDetails**;**

**if** **((SELECT** **COUNT(**SCREEN\_NO**)** **FROM** @TempScreen**)=**7**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 96'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempScreen**;**

-- -------------------------------------------------------------------

-- Test number: 97

-- Procedure Name: getScreenDetailsByMinimumCapacity

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempScreen **EXECUTE** getScreenDetailsByMinimumCapacity 200**;**

**if** **((SELECT** **COUNT(**SCREEN\_NO**)** **FROM** @TempScreen**)=**3**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 97'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempScreen**;**

-- -------------------------------------------------------------------

-- Test number: 98

-- Procedure Name: getScreenDetailsByScreenNumber

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempScreen **EXECUTE** getScreenDetailsByScreenNumber 2**;**

**if** **((SELECT** CAPACITY **FROM** @TempScreen **WHERE** SCREEN\_NO**=**2**)=**150**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 98'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempScreen**;**

-- -------------------------------------------------------------------

-- Test number: 99

-- Procedure Name: getScreensWithDisabledAccess

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempScreen **EXECUTE** getScreensWithDisabledAccess**;**

**if** **((SELECT** **COUNT(**SCREEN\_NO**)** **FROM** @TempScreen**)=**4**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 99'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempScreen**;**

-- -------------------------------------------------------------------

-- Test number: 100

-- Procedure Name: getScreensWithHearingLoop

-- -------------------------------------------------------------------

-- Add result set to table variable.

**Insert** @TempScreen **EXECUTE** getScreensWithHearingLoop**;**

**if** **((SELECT** **COUNT(**SCREEN\_NO**)** **FROM** @TempScreen**)=**4**)**

**BEGIN**

**SET** @passes **=** @passes **+** 1

**END**

**else**

**BEGIN**

Print 'Fail test 100'**;**

**SET** @fails **=** @fails **+** 1

**END**

**DELETE** **FROM** @TempScreen**;**

-- -------------------------------------------------------------------

-- END OF UNIT TESTS

-- -------------------------------------------------------------------

-- -------------------------------------------------------------------

-- Print stats  
-- -------------------------------------------------------------------

PRINT '---------------------------------------------------------------'

PRINT 'Tests passed = ' **+** **CAST(**@passes **AS** **VARCHAR)**

PRINT 'Tests failed = ' **+** **CAST(**@fails **AS** **VARCHAR)**

PRINT '---------------------------------------------------------------'

**SET** @**CURRENT\_TIME** **=** **(** **SELECT** **CONVERT(VARCHAR(**8**),** **GETDATE(),** 108**)** 'hh:mi:ss' **);**

print 'Script finished at ' **+** @**CURRENT\_TIME**

1. Full documentation on SQL Server datatypes can be found at: <https://docs.microsoft.com/en-gb/sql/t-sql/data-types> [↑](#footnote-ref-1)
2. The body of accounting standards and other guidance published by the UK’s Financial Reporting Council. [↑](#footnote-ref-2)
3. See Appendix A. [↑](#footnote-ref-3)
4. Refactoring is "… *the process of changing a software system in such a way that it does not alter the external behaviour of the code yet improves its internal structure.*" (Fowler, 2018) [↑](#footnote-ref-4)