CSG1207/CSI5135 Systems and Database Design

Assignment 2:	Database Design & Implementation (Movie Theatre)
Assignment Marks:	Marked out of 60, (30% of unit) Task 1 is marked out of 20, (10% of unit) Task 2 is marked out of 40, (20% of unit)
Due Dates:	Inform tutor if working in a pair: 17 September 2018, 9:00AM Task 1 (Database Design): 24 September 2018, 9:00AM Task 2 (Implementation): 29 October 2018, 9:00AM

General Assignment Information

Before a database can be implemented and used, it must be designed in a way that ensures it is appropriate for the task at hand. Tools such as Entity-Relationship Diagrams and Data Dictionaries assist in designing and communicating the structure of a database. Once a design has been finalised, the database can be implemented in a DBMS. The Data Definition Language commands of SQL are used to create the database and its tables, attributes and constraints, after which the Data Manipulation Language commands can be used to manipulate data within the database. This assignment takes you through the design and implementation process, using Microsoft SQL Server.

You may work in **pairs** (maximum of **2** people) to complete this assignment, or choose to work **alone**. If you wish to work in a pair, you **must** inform your tutor (by email) of the names and student numbers of both members at *least one week before the due date of Task 1*. If you choose to work alone, be aware that there are *no extra marks* to compensate for the heavier workload. If working in a pair, *work through the whole assignment together*, rather than dividing it and completing sections individually. This will help to ensure that both people learn the content and contribute evenly.

The assignment consists of two tasks. The first task, **Database Design**, requires a word processed document in PDF format detailing the design of your database. The second task, **Implementation**, is a collection of SQL scripts which create and populate the database designed in the first task, and then query the data it contains. A small amount of marks are dedicated to **presentation, notation and formatting**.

While Task 1 is worth relatively few marks it is the *basis of Task 2*, and therefore it is very important that it is done well, otherwise further work will be required before Task 2 can be attempted.

Task 1 – Database Design (20 marks)

Your first task is to design a database for the scenario detailed on the following pages. Your final database design should include approximately 10 entities.

***Note:** The scenario for this assignment is the same as the one from Task 3/4 of Assignment 1. Be sure to incorporate any feedback received in Assignment 1.*

State any **assumptions** (2 marks) you have made regarding your database design at the beginning of the database design document. Do not make any assumptions that significantly change the structure of the scenario, as this may make Task 2 of the assignment more difficult or inapplicable. Only make assumptions that influence your database design. If you are unsure about an assumption you wish to make, ask your tutor.

Once you feel you have identified the entities, attributes and relationships of the scenario in sufficient depth, you are required to create a **logical ER diagram** (4 marks) and a corresponding **physical ER diagram** (4 marks) to depict your database. Adhere to the distinctions between logical and physical ER diagrams covered in Lecture 3. It is recommended that you draw your diagrams on paper first, in order to find a layout that is clear and can be created in an electronic format.

Lastly, create a **data dictionary** (8 marks), with an entry for each entity in your database. The entries should list the name of the entity (table), a description of its purpose, a list of attributes (columns), important information about the attributes (e.g. data type, null/not null, identity, default values…), and details of any constraints applied to attributes. List the entries in your data dictionary in an appropriate *table creation order* that can be used to create the database. Include any additional information, if any, that may be needed to implement the database. Remember, a data dictionary should contain *all the information needed* to implement a database. Use the data dictionary in Lecture 4 and the data dictionary of the "company" database (Module 5) as examples.

Some marks are also awarded for **presentation and notation** (2 marks).

Your complete database design should consist of a list of assumptions, logical and physical ER diagrams and a data dictionary. This should be submitted as a ***single PDF document***. Make sure that your assignment includes the unit code, assignment number/name, year and semester and your name and student number on the first page.

Please ensure that your completed database design is in PDF format, and open the PDF file before submitting it to ensure that your diagrams appear as intended.

Scenario Details

You are required to design and create a database for a movie theatre. You have the following information about the way the movie theatre operates:

- Details of the **movies** that the theatre has must be recorded. This includes a movie ID number, the name of the movie, its duration in minutes and a description of the movie.

- The **classification** of each movie must also be recorded. To do this, the database should contain a table containing the details of classifications as shown below:

Classification	Name	Minimum Age
G	General	Not Applicable
PG	Parental Guidance	Not Applicable
M	Mature	Not Applicable
MA	Mature Audiences	15
R	Restricted	18

(The Classification column can be used as the primary key of the table)

  - The movie table will need a foreign key to identify the classification of each movie.

- The **genres** of each movie must also be recorded. The only details you need to store about genres are an ID number and name. Each movie can have multiple genres (but must have at least one), and there can be many movies of the same genre.

- The database must record details of the **cinemas** within the movie theatre – i.e., the rooms in which movies are viewed. The details that need to be stored about the cinemas are an ID number, the name of the cinema (e.g. "Cinema 3") and the seating capacity.

- To allow for better organisation and advertising of sessions, the theatre wants to keep track of their **cinema types**. Add an entity to store the list of cinema types shown below, and make sure that the cinema entity includes a foreign key to identify the type of each cinema.

Cinema Type ID	Cinema Type Name
1	2D
2	3D
3	Gold Class 2D
4	Gold Class 3D

- The database must record details of movie **sessions** – i.e., the screening of a movie in a cinema which customers can attend. The details of a session must include a session ID number, the date/time of the session (as one attribute), the cost of a ticket for the session, and foreign keys identifying the movie being played and the cinema that it is in.
    - Each movie that the theatre owns must have at least one session associated with it.

- The movie theatre only allows tickets to be purchased by **customers** who have registered online. The database must record the email address, password, first name, last name and date of birth of customers. The email address must be unique and can be the primary key.

- The database must record the details of **tickets** that customers have purchased. A customer can buy many tickets, and there can be many tickets purchased for a session. No additional details need to be recorded about tickets – seating is not reserved.

- Customers can write **reviews** of movies, and these reviews must be stored in the database. The text of the review, a rating between 0 and 5 and the date/time must be recorded for each review. Use the current date/time as the default value for the date/time column.
    - A customer can review many different movies, and a movie can be reviewed by many different customers.
    - A single customer should only be able to review a single movie once.

General Information and Guidelines

The information above describes the entities, attributes and relationships required in the database. Some minor details, such as the cardinality of some relationships, have been omitted. It is up to you to make (*and state*) any assumptions you need in order to complete the database design. If you are uncertain about any part of the scenario described above, seek clarification from your tutor.

Be sure to specify the most appropriate data type (and length, where applicable) for each attribute in your data dictionary. Note that when you are storing a date/time, it should be stored as a single column – do *not* split the date and time into two columns unless there is a very good and necessary reason to do so. Where sample data is provided, make sure it fits into the columns lengths you use. In tables with an ID number as the primary key, remember to specify that the column is auto-incrementing. Some tables will benefit from having a compound primary key.

Read the scenario details *several times* to ensure that your database design incorporates all the elements described. *If you desire feedback on your work in progress, send it to your tutor*.

CSI5135 Additional Requirements

If you are in CSI5135, the following additional requirements apply. If you are in CSG1207, you do not need to do implement these requirements (but you are encouraged to do so if you want).

Ensure that your database design (and implementation in Task 2) incorporates the following:

- Customer email addresses must contain a "@" symbol.
  - You are welcome to implement stricter or more realistic email formatting if desired.

- For security reasons, customer passwords will be encrypted using "bcrypt" before storing them in the database. You must do a small amount of research to determine an appropriate data type and length for the password column so that it is able to contain a "bcrypt hash".
  - It is up to you whether you actually put bcrypt hashes into the column when writing your sample data in Task 2 – it will not be used in any of the queries.

- The movie duration, cinema seating capacity and cost of a session columns should all have minimum values of 0 (i.e. do not allow negative numbers to be stored in these columns).

- The name of each genre, classification, cinema, cinema type must be unique within their respective tables.

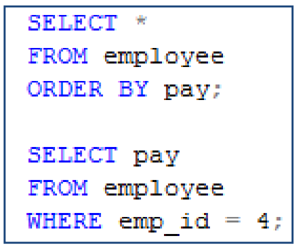
Some of these requirements will be implemented using CHECK constraints when creating the database. Specify these CHECK constraints in your data dictionary in a way that clearly describes what is being checked, or using the actual SQL code required to create the constraints.

Task 2 – Implementation (40 marks)

Once your database has been designed, it is time to implement it in a DBMS, populate the database, and then manipulate the data via queries. The deliverables of this task are **three files containing SQL statements**. We will be using Microsoft SQL Server 2014 or above – your SQL scripts *must* run in the same environment used in the unit/labs.

Create your scripts as three ".sql" files, with the filenames listed in the following headings. **Templates for the script files are provided with this assignment brief** – please use them. Format your code for **readability**, and use **comments** for headings and to provide further detail or information about your code if needed (2 marks).

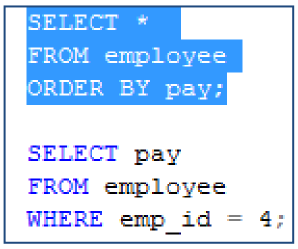
As each of the script files will contain numerous SQL statements, it is very useful to be aware of a particular feature of SQL Server Management Studio (SSMS): If you have selected some text in a query window, *only the selected text will be executed when you press the Execute button*.



```
SELECT *
FROM employee
ORDER BY pay;

SELECT pay
FROM employee
WHERE emp_id = 4;
```

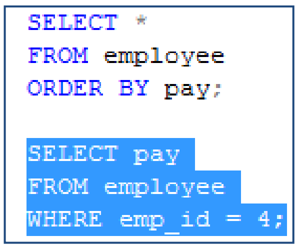
Execute both queries



```
SELECT *
FROM employee
ORDER BY pay;

SELECT pay
FROM employee
WHERE emp_id = 4;
```

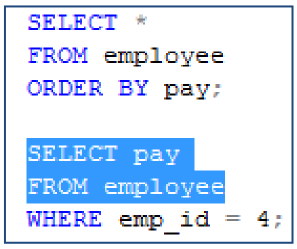
Execute query 1 only



```
SELECT *
FROM employee
ORDER BY pay;

SELECT pay
FROM employee
WHERE emp_id = 4;
```

Execute query 2 only



```
SELECT *
FROM employee
ORDER BY pay;

SELECT pay
FROM employee
WHERE emp_id = 4;
```

Execute part of query 2

This makes it much easier to test a single statement, or even part of a statement, within a script file.

You are required to create all of the scripts detailed below. Please take note of the file names and **use the template files provided with this assignment brief**.

Filename: create.sql

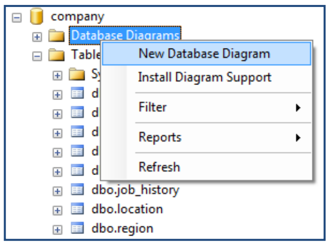
This file is a creation script, similar to the "company.sql" file (Module 5) used to create the company database.

Database Creation & Population Script (8 marks)

Produce a script to create the database you designed in Task 1 (incorporating any changes you have made since then). Be sure to give your columns the same data types, properties and constraints specified in your data dictionary, and be sure to name tables and columns consistently. Include any suitable default values and any CHECK or UNIQUE constraints that you feel are appropriate.

Make sure this script can be run *multiple times* without resulting in any errors (hint: drop the database if it exists before trying to create it). *You can use/adapt the code at the start of the creation scripts of the sample databases available in the unit materials to implement this.* You will need to follow an appropriate creation order when creating your tables – you cannot create a table with a foreign key constraint that refers to a table which does not yet exist.

Once you have created your database, it is recommended that you use SSMS to create an ER diagram and use this to verify that your implementation matches your design. This can be done by right clicking on the "Database Diagrams" folder of the database in the Object Explorer in SSMS.



Following the SQL statements to create your database and its tables, you must include statements to *populate the database with sufficient test data*. You are only required to populate the database with enough data to make sure that *all views and queries return meaningful results*. You can start working on your views and queries and write INSERT statements as needed for testing as you go.

For example, imagine you are working on a query which joins data from two tables and only displays the rows that meet certain criteria, then orders the results by a certain column. To test this, you will need to insert some data in both of the tables, making sure that some of the rows meet the criteria and others don't, and making sure that the column used for ordering contains a range of different values. Once you have inserted this data, you can test that your query works.

The final create.sql should be able to create your database and populate it with enough data to make all views and queries return meaningful results.

Make sure referential integrity is observed – you cannot add data to a column with a foreign key constraint if you do not yet have data in the column it references. Remember that when using an auto-incrementing integer, *you cannot specify a value for that column when inserting a row of data*. Simply pretend the column does not exist when inserting data – do not try to specify a value for it.

Note: *The data you add is simply for testing purposes, and therefore does not need to be particularly realistic, cohesive or consistent.*

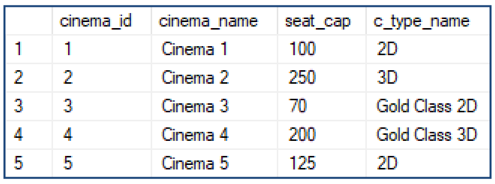
Avoid spending unnecessary amounts of time writing sample data. To assist with this, I have included data for a number of tables in the create_TEMPLATE.sql file.

Filename: views.sql

Views allow you to refer to the result of a SELECT statement as if it was a table, making query writing easier.

Cinema View (1 mark)

Create a *view* that selects the cinema ID number, cinema name, seating capacity and the name of the cinema type for all cinemas. This will involve joining the cinema and cinema type tables.



	cinema_id	cinema_name	seat_cap	c_type_name
1	1	Cinema 1	100	2D
2	2	Cinema 2	250	3D
3	3	Cinema 3	70	Gold Class 2D
4	4	Cinema 4	200	Gold Class 3D
5	5	Cinema 5	125	2D

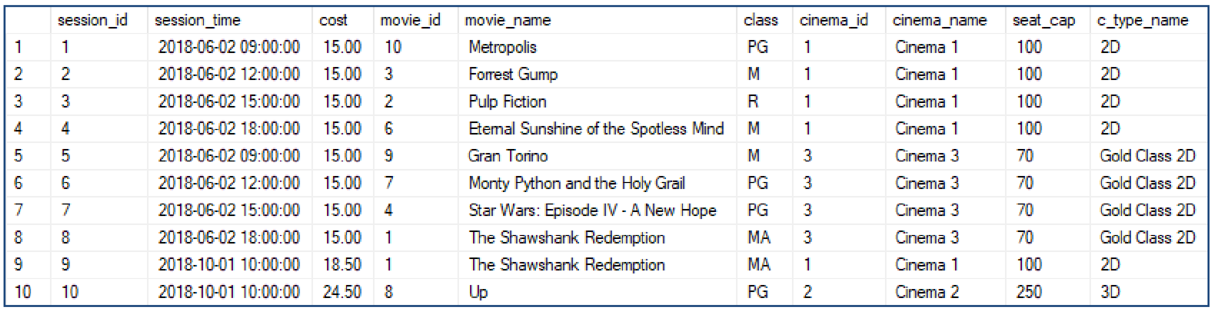
(example output of the Cinema View – your data may vary, example illustrates structure only)

Session View (2 marks)

Create a *view* that selects the following details of all movie sessions:

- The session ID number, session date/time and cost of the session.
- The movie ID number, movie name and classification of the movie (e.g. "PG") being shown.
- The cinema ID number, cinema name, seating capacity and cinema type name of the cinema that the session is in.

This statement requires multiple JOINs. Using the Cinema View in this view is recommended.



	session_id	session_time	cost	movie_id	movie_name	class	cinema_id	cinema_name	seat_cap	c_type_name
1	1	2018-06-02 09:00:00	15.00	10	Metropolis	PG	1	Cinema 1	100	2D
2	2	2018-06-02 12:00:00	15.00	3	Forrest Gump	M	1	Cinema 1	100	2D
3	3	2018-06-02 15:00:00	15.00	2	Pulp Fiction	R	1	Cinema 1	100	2D
4	4	2018-06-02 18:00:00	15.00	6	Eternal Sunshine of the Spotless Mind	M	1	Cinema 1	100	2D
5	5	2018-06-02 09:00:00	15.00	9	Gran Torino	M	3	Cinema 3	70	Gold Class 2D
6	6	2018-06-02 12:00:00	15.00	7	Monty Python and the Holy Grail	PG	3	Cinema 3	70	Gold Class 2D
7	7	2018-06-02 15:00:00	15.00	4	Star Wars: Episode IV - A New Hope	PG	3	Cinema 3	70	Gold Class 2D
8	8	2018-06-02 18:00:00	15.00	1	The Shawshank Redemption	MA	3	Cinema 3	70	Gold Class 2D
9	9	2018-10-01 10:00:00	18.50	1	The Shawshank Redemption	MA	1	Cinema 1	100	2D
10	10	2018-10-01 10:00:00	24.50	8	Up	PG	2	Cinema 2	250	3D

(example output of the Session View – your data may vary, example illustrates structure only)

You are encouraged to *use the views to simplify the queries* which follow - You can use a view in a SELECT statement in exactly the same way as you can use a table, often avoiding the need to write the same joins and calculations over and over.

When writing a view, it is easiest to write the SELECT statement first, and only add the CREATE VIEW statement to the beginning once you have confirmed that the SELECT statement is working correctly. If you wish to create additional views to use in the queries which follow, include them in this file.

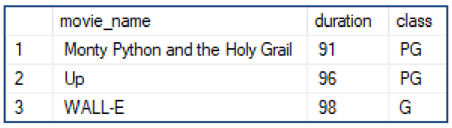
Joins are covered in Module 9, and views are covered in Module 10.

Filename: queries.sql

Write SELECT statements to complete the following queries. If you do not understand or are not sure about exactly what a query requires, contact your lecturer or tutor.

Query 1 – Child Friendly Movies (2 marks)

Write a query that selects the movie name, duration and classification of all movies that have a duration of less than 100 minutes and a classification of “G” or “PG”. Order the results by duration.

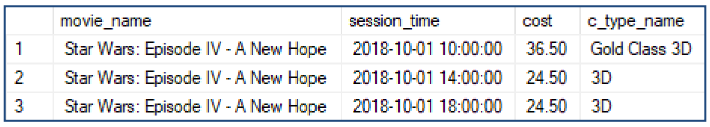


	movie_name	duration	class
1	Monty Python and the Holy Grail	91	PG
2	Up	96	PG
3	WALL-E	98	G

(expected query results, if provided data used)

Query 2 – Movie Search (2 marks)

Write a query that selects the movie name, session date/time, cinema type name and cost of all upcoming sessions (i.e. session date/time is later than the current date/time) showing movies that have “star wars” anywhere in the movie name. Order the results by session date/time.
Using the Session View in this query is recommended.

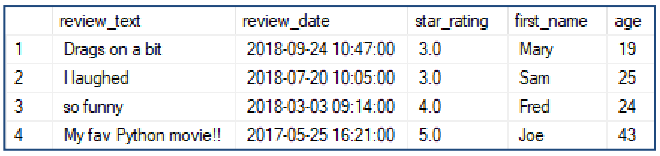


	movie_name	session_time	cost	c_type_name
1	Star Wars: Episode IV - A New Hope	2018-10-01 10:00:00	36.50	Gold Class 3D
2	Star Wars: Episode IV - A New Hope	2018-10-01 14:00:00	24.50	3D
3	Star Wars: Episode IV - A New Hope	2018-10-01 18:00:00	24.50	3D

(example of query results – your data may vary, example illustrates structure only)

Query 3 – Review Details (2 marks)

Write a query that selects the details of all reviews for the movie with a movie ID number of 5. The results should include the text of the review, the date/time the review was posted, the rating given, and the first name and age (calculated from the date of birth) of the customer who posted the review. Order the results by the review date, in descending order.



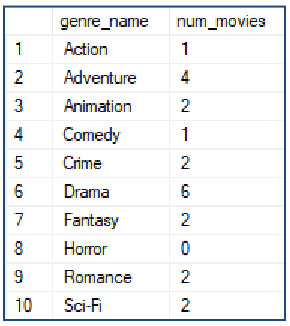
	review_text	review_date	star_rating	first_name	age
1	Drags on a bit	2018-09-24 10:47:00	3.0	Mary	19
2	I laughed	2018-07-20 10:05:00	3.0	Sam	25
3	so funny	2018-03-03 09:14:00	4.0	Fred	24
4	My fav Python movie!!	2017-05-25 16:21:00	5.0	Joe	43

(example of query results – your data may vary, example illustrates structure only)

Query 4 – Genre Count (3 marks)

Write a query that selects the name of all genres in the genre table, and the number of movies of each genre. Be sure to show all genres, even if there are no movies of that genre in the database.

Hint: *This will involve using an OUTER JOIN, GROUP BY and COUNT.*

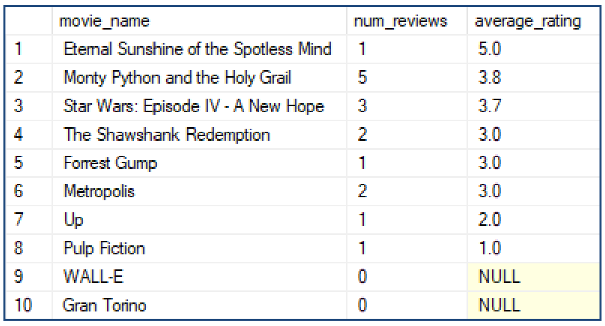


	genre_name	num_movies
1	Action	1
2	Adventure	4
3	Animation	2
4	Comedy	1
5	Crime	2
6	Drama	6
7	Fantasy	2
8	Horror	0
9	Romance	2
10	Sci-Fi	2

(expected query results, if provided data used)

Query 5 – Movie Review Stats (3 marks)

Write a query that selects the names of all movies in the movie table, how many reviews have been posted per movie, and the average rating of the reviews per movie. Be sure to include all movies, even if they have not been reviewed. Round the average rating to one decimal place and order the results by the average rating, in descending order.



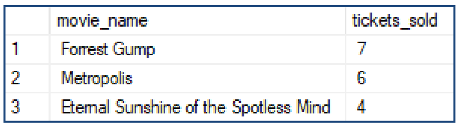
	movie_name	num_reviews	average_rating
1	Eternal Sunshine of the Spotless Mind	1	5.0
2	Monty Python and the Holy Grail	5	3.8
3	Star Wars: Episode IV - A New Hope	3	3.7
4	The Shawshank Redemption	2	3.0
5	Forrest Gump	1	3.0
6	Metropolis	2	3.0
7	Up	1	2.0
8	Pulp Fiction	1	1.0
9	WALL-E	0	NULL
10	Gran Torino	0	NULL

(example of query results – your data may vary, example illustrates structure only)

Query 6 – Top Selling Movies (3 marks)

Write a query that selects the name and total number of tickets sold of the three most popular movies (determined by total ticket sales). Using the Session View in this query is recommended.

Hint: *Join the session view/table to the ticket table and group the results by movie.*



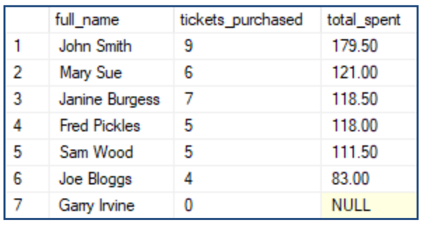
	movie_name	tickets_sold
1	Forrest Gump	7
2	Metropolis	6
3	Eternal Sunshine of the Spotless Mind	4

(example of query results – your data may vary, example illustrates structure only)

Query 7 – Customer Ticket Stats (4 marks)

Write a query that that selects the full names (by concatenating their first name and last name) of all customers in the customer table, how many tickets they have each purchased, and the total cost of these tickets. Be sure to include all customers, even if they have never purchased a ticket. Order the results by total ticket cost, in descending order.

Hint: *This will involve using OUTER JOINs, GROUP BY, COUNT and SUM.*



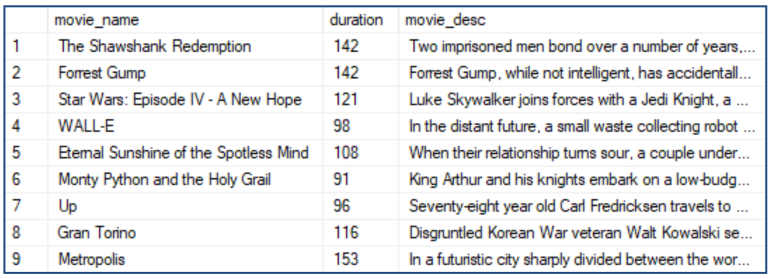
	full_name	tickets_purchased	total_spent
1	John Smith	9	179.50
2	Mary Sue	6	121.00
3	Janine Burgess	7	118.50
4	Fred Pickles	5	118.00
5	Sam Wood	5	111.50
6	Joe Bloggs	4	83.00
7	Garry Irvine	0	NULL

(example of query results – your data may vary, example illustrates structure only)

Query 8 – Age Appropriate Movies (4 marks)

Write a query that selects the name, duration and description of all movies that a certain customer (chosen by you) can legally watch, based on the customer's date of birth and the minimum age required by the movie's classification. Select a customer whose date of birth makes them 15-17 years old for this query, so that the results include all movies except those classified "R".

Hint: *Use a subquery to select the age of your chosen customer.*

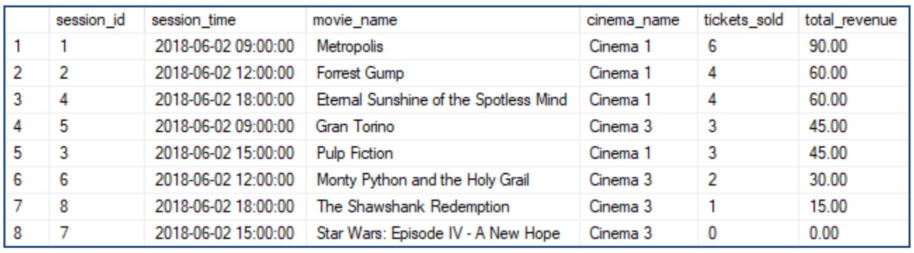


	movie_name	duration	movie_desc
1	The Shawshank Redemption	142	Two imprisoned men bond over a number of years,...
2	Forrest Gump	142	Forrest Gump, while not intelligent, has accidentall...
3	Star Wars: Episode IV - A New Hope	121	Luke Skywalker joins forces with a Jedi Knight, a ...
4	WALL-E	98	In the distant future, a small waste collecting robot ...
5	Eternal Sunshine of the Spotless Mind	108	When their relationship turns sour, a couple under...
6	Monty Python and the Holy Grail	91	King Arthur and his knights embark on a low-budg...
7	Up	96	Seventy-eight year old Carl Fredricksen travels to ...
8	Gran Torino	116	Disgruntled Korean War veteran Walt Kowalski se...
9	Metropolis	153	In a futuristic city sharply divided between the wor...

(expected query results, if provided data used)

Query 9 – Session Revenue (4 marks)

Write a query that selects the session ID, session date/time, movie name, cinema name, tickets sold and total revenue of all sessions that occurred in the past. Total revenue is the session cost multiplied by the number of tickets sold. Ensure that sessions that had no tickets sold appear in the results (with 0 tickets sold and 0 revenue). Order the results by total revenue, in descending order.



	session_id	session_time	movie_name	cinema_name	tickets_sold	total_revenue
1	1	2018-06-02 09:00:00	Metropolis	Cinema 1	6	90.00
2	2	2018-06-02 12:00:00	Forrest Gump	Cinema 1	4	60.00
3	4	2018-06-02 18:00:00	Eternal Sunshine of the Spotless Mind	Cinema 1	4	60.00
4	5	2018-06-02 09:00:00	Gran Torino	Cinema 3	3	45.00
5	3	2018-06-02 15:00:00	Pulp Fiction	Cinema 1	3	45.00
6	6	2018-06-02 12:00:00	Monty Python and the Holy Grail	Cinema 3	2	30.00
7	8	2018-06-02 18:00:00	The Shawshank Redemption	Cinema 3	1	15.00
8	7	2018-06-02 15:00:00	Star Wars: Episode IV - A New Hope	Cinema 3	0	0.00

(example of query results – your data may vary, example illustrates structure only)

Presentation, Notation and Formatting (2 marks per part)

A small amount of marks are awarded for presentation, notation and formatting. This includes:

- Presentation and appearance of word processed PDF document for Task 1
- Appropriateness and consistency of notation used for diagrams/data dictionary in Task 1
- Appropriate commenting and formatting of scripts in Task 2

Submission of Deliverables

Submit your database design (Task 1, a PDF document) by the Task 1 due date above. Submit your three script files (Task 2) by the Task 2 due date above. Submit the files to the appropriate locations in the Assessments area of Blackboard.

If you are working in pairs, **both** people should submit the same assignment. Be sure to include the **name and student number of both members** in the assignment content and submission comments. Remember, if you wish to work in a pair, you **must** inform your tutor of the names and student numbers of both members at *least one week before the due date of Task 1*.

Submissions via email are NOT permitted, unless you are specifically instructed to do so. Make sure that your assignment includes the unit code, assignment number/name, year and semester and your name and student number on the first page.

Referencing, Plagiarism and Collusion

The **entirety of your assignment must be your own work** (unless otherwise referenced) and produced for the current instance of the unit. Any use of unreferenced content you did not create constitutes plagiarism, and is deemed an act of academic misconduct. All assignments will be submitted to plagiarism checking software which includes previous copies of the assignment, and the work submitted by all other students in the unit.

Remember that this is an **individual** assignment*. Never give anyone any part of your assignment – even after the due date or after results have been released. Do not work together with other students on individual assignments – helping someone by explaining a concept or directing them to the relevant resources is fine, but doing the assignment for them or alongside them, or showing them your code is not appropriate.* An unacceptable level of cooperation between students on an assignment is collusion, and is deemed an act of academic misconduct. If you are uncertain about plagiarism, collusion or referencing, simply contact your tutor, lecturer or unit coordinator and ask.

You may be asked to **explain and demonstrate your understanding** of the work you have submitted. Always remember that the purpose of an assignment is for you to **demonstrate your understanding** of the unit content and **your ability to apply it** to the task presented to you.

Assignment 2 Task 1 Marking Key

Marks are allocated as follows for this assignment.

Criteria	Marks
Assumptions All/Any assumptions that influence the database design clearly stated.	2
Logical ER Diagram Diagram accurately depicts the scenario and includes all elements specified in the brief.	4
Physical ER Diagram Accurately depicts the scenario and is a correct translation of the logical ER diagram.	4
Data Dictionary Includes all entities and details of attributes as specified in brief and correct creation order used.	8
Presentation and Notation Assignment is well presented, uses consistent and appropriate notation.	2

Total:	**20** **(10% of unit)**

Assignment 2 Task 2 Marking Key

Marks are allocated as follows for this assignment.

Criteria All scripts are judged on correctness, appropriateness and readability of SQL code.	Marks
Database Creation & Population Script	8
Cinema View	1
Session View	2
Query 1	2
Query 2	2
Query 3	2
Query 4	3
Query 5	3
Query 6	3
Query 7	4
Query 8	4
Query 9	4
Formatting – Scripts are well formatted and use appropriate commenting.	2

Total:	**40** **(20% of unit)**