

Name of the database

**Name of Student 1 Thomas Lawrence**

**Student number N00221602**

**Name of Student 2 Henry Donnelly**

**Student number N00220105**

Advanced Database

CA 1 – Database Design

Year 2 2021-22

DL836 BSc (Hons) in Creative Computing

Table of Contents

[1 Activity 1 - Business Outline 1](#_Toc83040624)

[1.1 Description of the Business 1](#_Toc83040625)

[1.2 Business Reporting Requirements 1](#_Toc83040626)

[2 Activity 2 – Data-Set 2](#_Toc83040627)

[2.1 The Data-Set 2](#_Toc83040628)

[2.2 Relationship Rules for the Business 2](#_Toc83040629)

[3 Activity 3 – Entity Relationship Diagram 3](#_Toc83040630)

[3.1 ERD 3](#_Toc83040631)

[3.2 Data Dictionary 3](#_Toc83040632)

[3.3 Entity Constraints 3](#_Toc83040633)

[4 Activity 4 – SQL create tables 4](#_Toc83040634)

[4.1 SQL to create the Tables 4](#_Toc83040635)

[4.2 Database Structure 4](#_Toc83040636)

[5 Activity 5 – Sample data 5](#_Toc83040637)

[6 Activity 6 – SQL queries 6](#_Toc83040638)

[7 Reflection 7](#_Toc83040639)

[7.1 Your views on the project 7](#_Toc83040640)

[7.2 Assessment of your learning. 7](#_Toc83040641)

[7.3 Technical skills 7](#_Toc83040642)

# Activity 1 - Business Outline

## Description of the Business

The name of our business is Omniplex cinemas. We are a popular Irish cinema chain with over 30 locations across Ireland, 20 of those locations consist within the Republic of Ireland while our other locations are situated in Northern Ireland. Our service provides our customers the high-quality experience that modern day cinema has to offer. This will include a vast selection of food and drink as well as several screenings of different movies throughout the day which will give our customers multiple options on what they would like to watch.

## Business Reporting Requirements

Due to the popularity of our brand, we need a consistent and reliable method to store our data. This method should be in the form of a database. The first piece of data that will consist in our database will include our multiple screens and the multiple screenings that we will be showing each day. Throughout the day, each screen will be showing a variety of different movies, meaning that One particular screen can play Many particular movies. Many movies can also be shown of multiple screens.

If a customer wants to watch one of these screenings, they must make a booking. Customers can make a booking for multiple different movies, meaning that One particular booking can be used for Many particular screenings. These movies must provide info such as the movie’s title, the main cast of the movie, the director’s credits and the running time of the movie.

Before a customer can make this booking, they must first create an Omniplex account on our website. This process will ask the customer to input their first name, last name, email, address and their phone number. Each booking a customer makes will need to show the cost of the booking as well as the general info of the screening, such as the date and time information and the seat numbers the customer selected.

# Activity 2 – Data-Set

## The Data-Set

CUSTOMER – (first\_name, last\_name, email, address, phone\_number)

BOOKING – (price, seat\_number, date, time)

MOVIE – (title, main\_cast, director, running\_time)

SCREEN – (screen\_number, screen type)

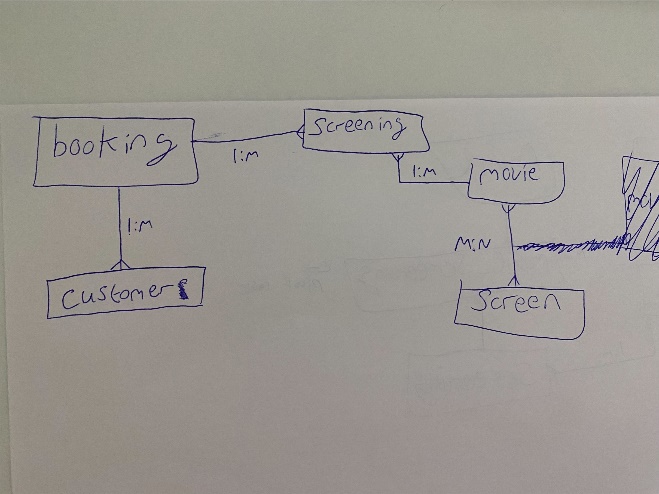
SCREENING – (movie\_id, screen\_id, booking\_id)

## Relationship Rules for the Business

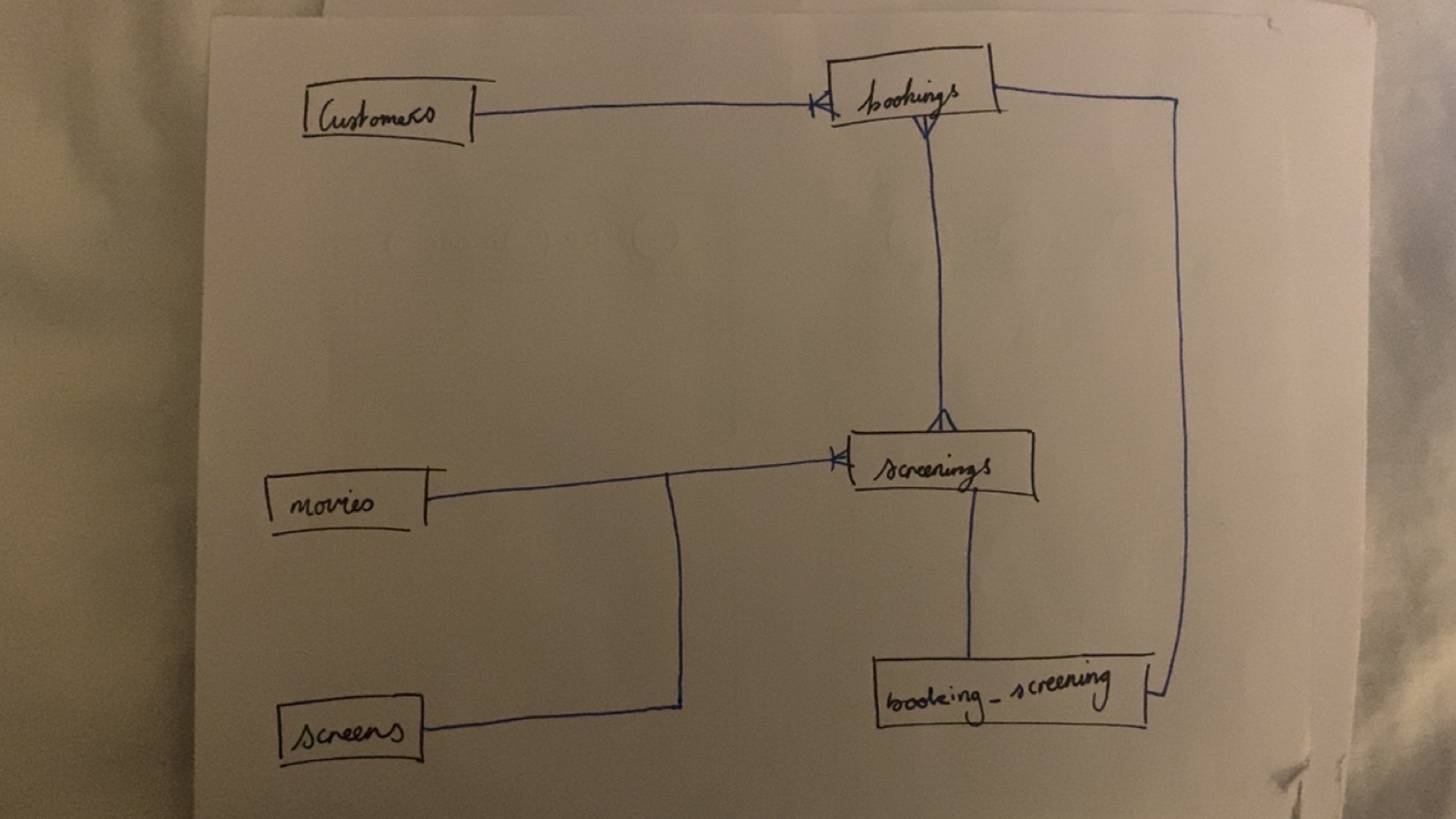
* A CUSTOMER can make one BOOKING
* A BOOKING can have many CUSTOMERs
* A BOOKING can be for many SCREENINGs
* A SCREENING needs one BOOKING to enter
* A SCREENING only shows one MOVIE
* A MOVIE can have many SCREENINGs
* A MOVIE can play on many SCREENs
* A SCREEN plays many MOVIEs throughout the day

# Activity 3 – Entity Relationship Diagram

## ERD



***Figure 1.1 - The first ERD sketched out for our cinema.***

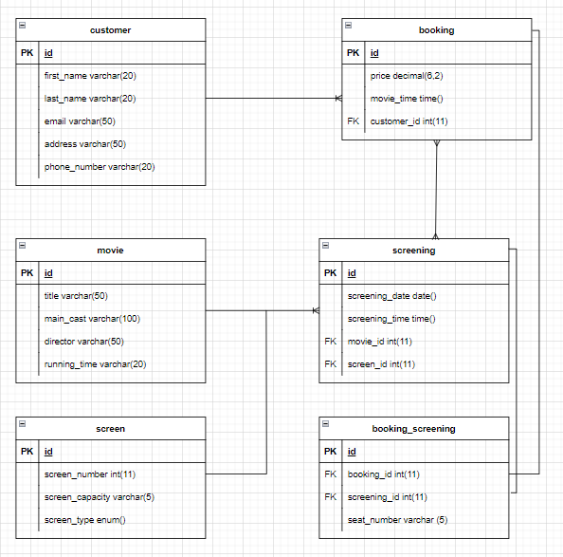
,

|  |  |
| --- | --- |
| |  | | --- | | ***Figure 1.2 - The final ERD for our cinema.*** | |

An ERD (Entity Relational Diagram) is a simple visual representation of a database. It does not contain much information and is very minimalistic, only showing us the tables and relationships.

The tables in an ERD are represented as boxes and are labelled with the appropriate table name. We used “crows feet” annotation for this ERD. This annotation will be seen later in our schema diagram.

## Database schema



|  |  |
| --- | --- |
| |  | | --- | | ***Figure 2 - An ERD schema created using Draw.io.*** | |

A database schema is a more detailed diagram of a database. Unlike an ERD, a schema shows the tables as well as each of the tables attributes and constraints. The constraints used in this database schema are PKs (Primary Keys) and FKs (Foreign Keys).

Primary keys are used to identify a table, which makes each of them unique. They are usually inserted into a table’s ‘id’. Foreign keys are used to connect different tables together. This allows different tables to link their data together.

## Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Attribute*** | ***Data types*** | ***Field length*** | ***Relations*** | ***Description*** |
| id | INT | 11 | ALL | The ‘id’ of each table |
| first\_name | VARCHAR | 20 | customers | The first name of a customer |
| last\_name | VARCHAR | 20 | customers | The last name of a customer |
| email | VARCHAR | 50 | customers | The customer’s email |
| address | VARCHAR | 50 | customers | The customer’s address |
| phone\_number | VARCHAR | 20 | customers | The customer’s phone number |
| price | DECIMAL | 6,2 | bookings | Total price for a booking |
| movie\_time | TIME | NULL | bookings | The time of a movie |
| customer\_id | INT | 11 | bookings | Links booking table to customer |
| title | VARCHAR | 50 | movies | The title of a movie |
| main\_cast | VARCHAR | 100 | movies | The main cast played in a movie |
| director | VARCHAR | 50 | movies | The director of a movie |
| running\_time | VARCHAR | 20 | movies | The running time of a movie |
| screening\_date | DATE | NULL | screenings | The date of a screening |
| screening\_time | TIME | NULL | screenings | The time of a screening |
| movie\_id | INT | 11 | screenings | Links screening table to movie |
| screen\_id | INT | 11 | screenings | Links screening table to screen |
| screen\_number | INT | 11 | screens | The screen number for a movie |
| screen\_capacity | VARCHAR | 5 | screens | The seat capactity of a screen |
| screen\_type | ENUM | NULL | screens | The type of screen being used |
| booking\_id | INT | 11 | booking\_screening | Links booking\_screening to booking |
| screening\_id | INT | 11 | booking\_screening | Links booking\_screening to booking |
| seat\_number | VARCHAR | 5 | booking\_screening | The customer’s seat number |

## Entity Constraints

|  |  |  |
| --- | --- | --- |
| ***Attribute*** | ***Table*** | ***Constraints*** |
| id | All tables | PRIMARY KEY |
| customer\_id | bookings | FOREIGN KEY |
| movie\_id | screenings | FOREIGN KEY |
| screen\_id | screenings | FOREIGN KEY |
| booking\_id | booking\_screening | FOREIGN KEY |
| screening\_id | booking\_screening | FOREIGN KEY |

# Activity 4 – SQL create tables

## SQL to create the Tables

CREATE DATABASE cinema;

use cinema;

CREATE TABLE customers(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

first\_name varchar(20),

last\_name varchar(20),

email varchar(20),

address varchar(20),

phone\_number varchar(20),

PRIMARY KEY(id));

CREATE TABLE bookings(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

price decimal(6,2),

movie\_time time,

customer\_id INT UNSIGNED,

PRIMARY KEY(id),

CONSTRAINT fk\_booking\_customer FOREIGN KEY (customer\_id) REFERENCES customers(id)

);

CREATE TABLE movies(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

title varchar(50),

main\_cast varchar(100),

director varchar(50),

running\_time varchar(20),

PRIMARY KEY(id));

CREATE TABLE screens(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

screen\_number int(11),

screen\_capacity varchar(5),

screen\_type enum('Regular','IMAX','3D'),

PRIMARY KEY(id));

CREATE TABLE screenings(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

movie\_id INT UNSIGNED,

screen\_id INT UNSIGNED,

screening\_date date,

screening\_time time,

PRIMARY KEY(id),

CONSTRAINT fk\_screening\_movie FOREIGN KEY (movie\_id) REFERENCES movies(id),

CONSTRAINT fk\_screening\_screen FOREIGN KEY (screen\_id) REFERENCES screens(id)

);

CREATE TABLE booking\_screening(

Id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

booking\_id INT UNSIGNED,

screening\_id INT UNSIGNED,

seat\_number varchar(5),

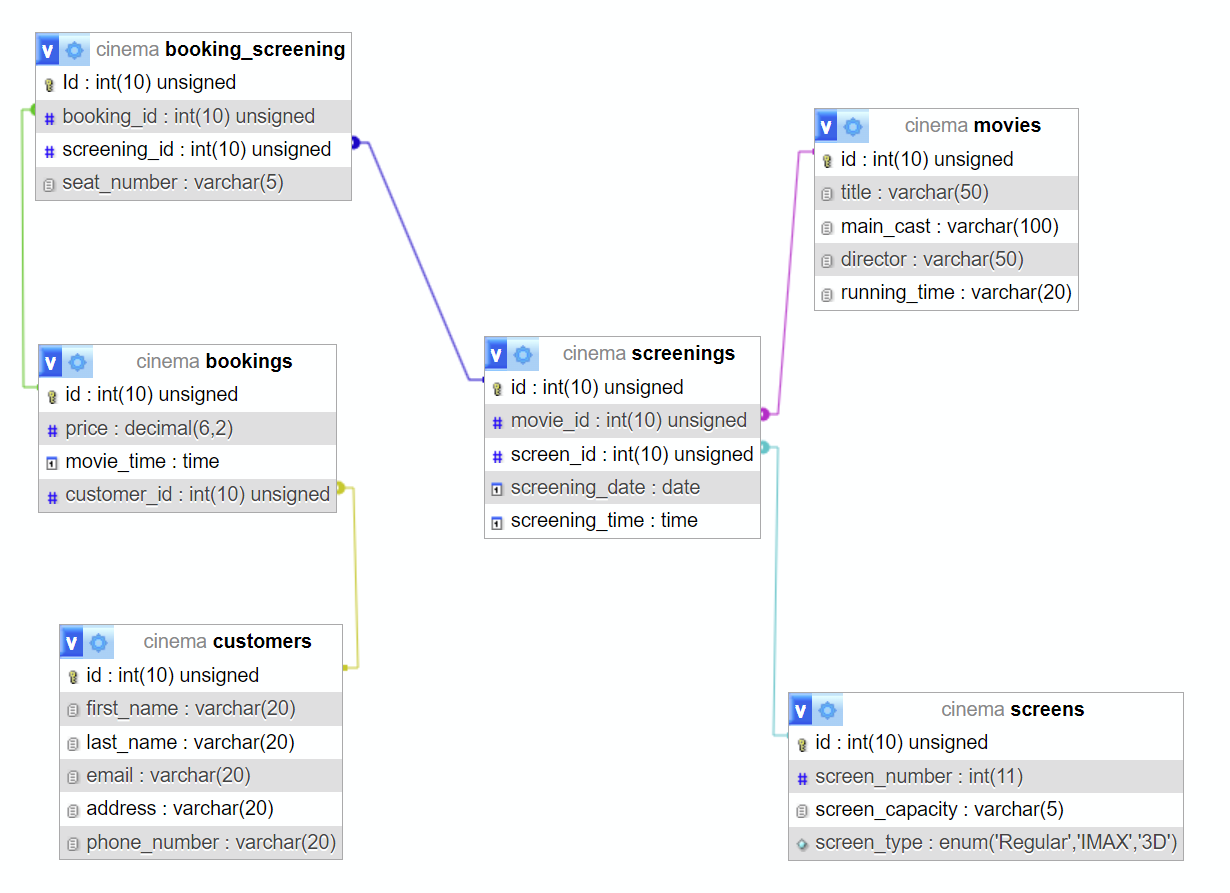
PRIMARY KEY(id),

CONSTRAINT fk\_booking\_screening\_booking FOREIGN KEY (booking\_id) REFERENCES bookings(id),

CONSTRAINT fk\_booking\_screening\_screening FOREIGN KEY (screening\_id) REFERENCES screenings(id));

## Database Structure

Select the Designer view from phpMyAdmin to display the structure of the database in a graphical way.



# Activity 5 – Sample data

Populate the tables with sample data

USE cinema;

/\*insert data\*/

INSERT INTO customers (first\_name, last\_name, email, address, phone\_number)

VALUES

('Paul', 'Jones', 'paulj@gmail.com', 'Dublin', 0871234567),

('John', 'Smith', 'johns@gmail.com', 'Cork', 0832220098),

('Sam', 'Green', 'samg@gmail.com', 'Dublin', 0833217765),

('Sarah', 'Young', 'sarahy@gmail.com', 'Laois', 0859442121),

('Milly', 'Reilly', 'millyr@gmail.com', 'Laois', 0867001035);

INSERT INTO bookings (price, movie\_time, customer\_id)

VALUES

(12.99, '18:15', '2'),

(17.99, '15:35', '4,1'),

(8.99, '14:05', '1,5'),

(23.99, '19:45', '3'),

(19.99, '13:25', '5,2,3');

INSERT INTO movies (title, main\_cast, director, running\_time)

VALUES

('Wolf of Wall Street', 'Leonardo DiCaprio, Jonah Hill, Margot Robbie', 'Martin Scorsese', '180 min'),

('Nightcrawler', 'Jake Gyllenhaal, Rene Russo, Riz Ahmed', 'Dan Gilroy', '117 min'),

('The Matrix', 'Keanu Reeves, Laurence Fishbourne, Carrie-Anne Moss', 'Lana Wachowski, Lili Wachowski', '136 min'),

('Brokeback Mountain', 'Heath Ledger, Jake Gyllenhaal, Linda Cardellini', 'Ang Lee', '134 min'),

('The Dark Knight', 'Christian Bale, Heath Ledger, Gary Oldman', 'Christopher Nolan', '152 min');

INSERT INTO screens (screen\_number, screen\_capacity, screen\_type)

VALUES

(1, 60, 'Regular'),

(2, 60, 'Regular'),

(3, 70, 'IMAX'),

(4, 50, '3D'),

(5, 70, 'IMAX');

INSERT INTO screenings (screening\_date, screening\_time, movie\_id, screen\_id)

VALUES

('2023-11-02', '18:00', 1, 1),

('2023-11-02', '15:20', 2, 2),

('2023-11-03', '13:50', 3, 4),

('2023-11-04', '19:30', 4, 3),

('2023-11-05', '13:10', 5, 5);

INSERT INTO booking\_screening (booking\_id, screening\_id, seat\_number)

VALUES

(3, 5, 'G14'),

(2, 1, 'A10, A11'),

(4, 4, 'D7'),

(1, 3, 'B4, B5, B6'),

(5, 2, 'E8, E9');

# Activity 6 – SQL queries

Write the SQL queries to meet your business reporting requirements

/\*A minimum of 6 queries which display:

o Particular rows based on a condition

o Particular rows based on more than 1 condition

o Rows in a particular sequence \*/

/\*All customers with 'john' as their first name\*/

SELECT \* FROM customers WHERE first\_name = 'john';

/\*All customer names that start with p\*/

SELECT \* FROM customers WHERE first\_name LIKE 'p%';

/\*Shows movies that start with an n and and with an r\*/

SELECT \* FROM movies WHERE title LIKE 'N%R';

/\*Sort movies in alphabetical order\*/

SELECT \* FROM movies ORDER BY title ASC;

/\*Shows books that cost over €10 and less that €20\*/

SELECT \* FROM bookings WHERE price >= '10' AND price <= '20';

/\*shows all customers that have a name beginning with s and are from Dublin \*/

SELECT \* FROM customers WHERE first\_name LIKE 'S%' AND address= 'Dublin';

/\*A minimum of:

o 5 aggregate function queries, 2 of which have a group by clause

o 5 Joins

o 2 subqueries \*/

/\*All customers with 'john' as their first name\*/

SELECT \* FROM customers WHERE first\_name = 'john';

/\*All customer names that start with p\*/

SELECT \* FROM customers WHERE first\_name LIKE 'p%';

/\*Shows movies that start with an n and and with an r\*/

SELECT \* FROM movies WHERE title LIKE 'N%R';

/\*Sort movies in alphabetical order\*/

SELECT \* FROM movies ORDER BY title ASC;

/\*Shows books that cost over €10 and less that €20\*/

SELECT \* FROM bookings WHERE price >= '10' AND price <= '20';

/\*Shows all customers that have a name beginning with s and are from Dublin \*/

SELECT \* FROM customers WHERE first\_name LIKE 'S%' AND address= 'Dublin';

/\*Shows total sum of all bookings\*/

SELECT SUM(price) FROM bookings;

/\*Shows all bookings that are over €15\*/

SELECT COUNT(price) FROM bookings WHERE price>15;

/\*Shows average running time of the movies\*/

SELECT AVG(running\_time) FROM movies;

/\*Shows all customer addresses and how many are from each address\*/

SELECT COUNT(id), address FROM customers GROUP BY address;

/\*Shows all customer addresses and how many are from each address, in a descending order\*/

SELECT COUNT(id), address FROM customers GROUP BY address ORDER BY COUNT(id) DESC;

/\*Select customers and booking price they are paying\*/

SELECT customers.first\_name, customers.last\_name, bookings.price

FROM customers

JOIN bookings

ON bookings.customer\_id = customers.id;

/\*Select all bookings with customer information\*/

SELECT bookings.id, bookings.price, bookings.movie\_time, customers.first\_name, customers.last\_name

FROM bookings

INNER JOIN customers

ON bookings.customer\_id = customers.id;

/\*select everthing about the customers that made a booking\*/

SELECT customers.id, customers.first\_name, customers.last\_name, customers.email

FROM customers

INNER JOIN bookings

ON customers.id = bookings.customer\_id;

/\*Retrieves information about movies, their screenings, and the bookings. \*/

SELECT movies.title, bookings.price, bookings.movie\_time

FROM movies

INNER JOIN screenings ON movies.id = screenings.movie\_id

INNER JOIN booking\_screening ON screenings.id = booking\_screening.screening\_id

INNER JOIN bookings ON booking\_screening.booking\_id = bookings.id;

/\*Select all customers and their bookings (including customers without bookings)\*/

SELECT customers.id, customers.first\_name, customers.last\_name, customers.email, bookings.id AS booking\_id, bookings.price, bookings.movie\_time

FROM customers

RIGHT JOIN bookings ON customers.id = bookings.customer\_id;

/\*Subquery in the WHERE Clause - Find Customers who made Bookings\*/

SELECT id, first\_name, last\_name FROM customers WHERE id IN (SELECT DISTINCT customer\_id FROM bookings);

/\*Subquery in the SELECT Clause - Count Screenings for Each Movie\*/

SELECT movies.title, (SELECT COUNT(\*) FROM screenings WHERE screenings.movie\_id = movies.id) AS screening\_count

FROM movies;

/\*A minimum of:

o 3 queries using the UPDATE statement. Each UPDATE query should update rows in different tables

o 3 queries that DELETE data from the database. Each DELETE query should delete rows from different tables

o 2 ALTER queries, 1 to ADD a column, 1 to DELETE a column \*/

/\*update customers with new email address and phone no. who's id is one\*/

UPDATE customers

SET email = 'new.email@example.com', phone\_number = '555-5678'

WHERE id = 1;

/\*update movies screening at this date and change price to 1.1x\*/

UPDATE screenings

SET screening\_date = '2023-11-20'

WHERE movie\_id = 1 AND screen\_id = 1;

UPDATE bookings SET price = price \* 1.1;

/\*adds column\*/

ALTER TABLE customers

ADD COLUMN loyalty\_points INT;

/\*delete screen\_type from screens\*/

ALTER TABLE screens

DROP COLUMN screen\_type;

/\*delete screenings with this date\*/

DELETE FROM screenings WHERE screening\_date = '2023-11-15';

/\*Adds DOB column\*/

ALTER TABLE customers ADD COLUMN date\_of\_birth DATE;

/\*deletes main cast column\*/

ALTER TABLE movies DROP COLUMN main\_cast;

# Reflection

## Your views on the project

Describe how you feel the project went from your perspective.

I think it went well, I gained a deeper insight into writing queries and worked well with Thomas. During week 1 it was rough because our DB schema was poorly designed (as you can see in the first one). We also had a poor idea of the database dictionary because our table titles were singular (I.e. customers were customer) and we had a pivot table not linked to any other table. On the other hand, once we got to work, we had it done in no time.

## Assessment of your learning.

Assess your learning. Describe what general skills and competencies you have developed in this Continuous Assessment.

I learned how to work with a teammate on a project and how to manage my work time well, I also showed good work skills when we had to redo the entire database because our database schema was wrong

## Technical skills

Describe what you have learnt from the project, from a technical skills viewpoint.

I learned how to write more queries, such as more in depth with joins and subqueries. Learning about alter queries was also interesting such as altering tables without having to manually do so. I feel I got a good grasp with the basics of SQL and look forward to no SQL databases.