rulecolor

Cinema Chain Database

Relational Database Design & Business Analytics

Course	SQL II · IE University
Project	Group 3 — Final Assignment
Year	2026
Database	MySQL — 16 Tables, 3NF Normalised

Contents

1. Business Context

This operational database models a cinema chain with multiple locations across Spain. The business operates under two key structural constraints that directly influenced the schema design:

- **Movie tickets** are sold exclusively through the website — no cash payments accepted.
- **Concession purchases** accept any payment method, including cash and walk-in customers.

Showtimes are organised into four daily periods — Morning, Afternoon, Evening, and Late Night — to support time-based analytics across both revenue streams simultaneously.

2. Database Structure Overview

The database contains **16 tables** organised into five functional sections:

Section	Tables	Purpose
Location & Facilities	`location`, `auditorium`, `auditorium_type`	Physical infrastructure of the cinema chain
Films & Scheduling	`film`, `film_genre`, `genre`, `showtime`, `showtime_group`	Movie catalogue and screening schedule
Movie Ticket Sales	`movie_ticket`, `seat`, `ticket_status`	Online ticket transactions
Concessions	`shop_ticket`, `shop_item`, `product`, `product_category`	In-store food and beverage sales
Shared Entities	`customer`, `payment_method`	Bridge entities used by both sales systems

3. Key Design Decisions

3.1 Auditorium-Based Pricing

Ticket prices are determined by **auditorium type** (Regular, IMAX, VIP, 4DX, 3D) rather than individual seats. This reflects real cinema operations — when a location invests in IMAX equipment, they build an entire auditorium, not specific seats. The `auditorium_type` table includes a `Price_Multiplier` column enabling dynamic pricing:

```
Final Price = Base Ticket Price × Auditorium Type Multiplier
```

This centralises pricing logic and simplifies maintenance across all locations.

3.2 Film–Genre Many-to-Many Relationship

Films frequently span multiple genres. The `film_genre` junction table implements this relationship properly, avoiding:

- Data redundancy from storing genres in a comma-separated field
- Query complexity when filtering by genre
- Data integrity issues from inconsistent genre naming

3.3 Separate Ticketing Systems

Movie tickets and concession purchases use independent transaction structures:

- **`movie_ticket`**: Requires customer registration, a specific seat assignment, and is linked to a showtime.
- **`shop_ticket + shop_item`**: Supports anonymous purchases (`Customer_ID` nullable), with no seat or showtime dependency — reflecting real operations where customers can buy snacks without watching a film.

3.4 Explicit `Location_ID` on `movie_ticket`

Although location can be derived through `showtime` → `auditorium` → `location`, `Location_ID` is explicitly stored on `movie_ticket` for two reasons:

1. Simplifies revenue-per-location reporting queries significantly.
2. Supports scenarios where a ticket is purchased centrally for a specific location.

3.5 Showtime Group Time Ranges

The `showtime_group` table uses `Start_Range` and `End_Range` columns. This enables direct linkage for movie tickets via `Showtime_Group_ID`, and time-based matching for shop purchases using `TIME(Purchase_DateTime)` against the ranges — answering time-of-day analytics across both revenue streams without artificial relationships.

3.6 Shared Customer and Payment Method

`customer` is required for movie tickets (web-only) but optional for shop transactions (walk-ins allowed). `payment_method` is a shared lookup table, but business rules restrict movie tickets to card-only payments.

4. Entity–Relationship Diagram

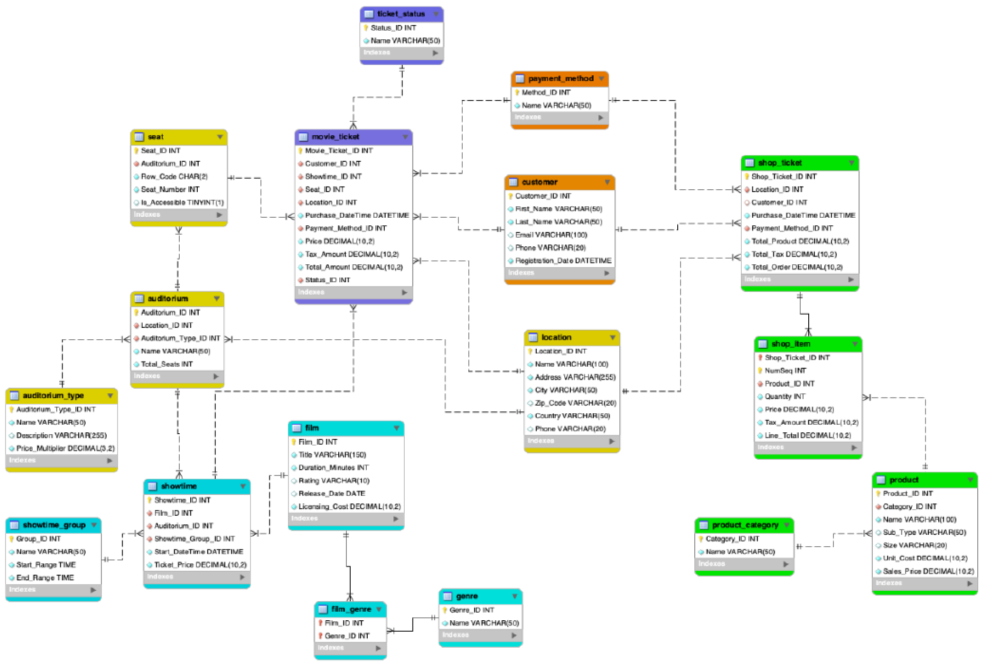

Figure 1: Entity–Relationship Diagram — Cinema Chain Database (MySQL Workbench)

ERD Colour Legend

Colour	Section
Yellow	Location-based tables
Blue	Films & Scheduling
Purple	Movie Ticket Sales
Green	Concessions (purchases)
Orange	Shared Entities (Customer, Payment Method)

5. Normalisation Compliance (3NF)

The database satisfies Third Normal Form (3NF):

Form	Compliance
1NF	All columns contain atomic values — no repeating groups or multi-valued attributes.
2NF	All non-key attributes depend on the entire primary key. Composite keys in `film_genre` and `shop_item` are fully utilised.
3NF	No transitive dependencies. Lookup tables eliminate all redundancy by storing each piece of information exactly once.

Examples from the schema:

- Genre names stored once in `genre`, referenced by ID in `film_genre`
- Auditorium type details stored once in `auditorium_type`, referenced by `auditorium`
- Product categories normalised into a separate `product_category` lookup table

6. Business Questions & SQL Queries

#	Question	Key Tables
Q1	Most profitable film title & showtime	`film`, `movie_ticket`, `showtime`, `showtime_group`
Q2	Popcorn & Coke combo purchase rate	`shop_ticket`, `shop_item`, `product`, `product_category`
Q3	Showtime group with most sales (inc. shop)	`movie_ticket`, `shop_ticket`, `showtime_group`
Q4	Monthly revenue per location (12 months)	`movie_ticket`, `shop_ticket`, `location`
Q5	Auditorium type with highest avg revenue/ticket	`movie_ticket`, `showtime`, `auditorium`, `auditorium_type`

6.1 Q1 — Most Profitable Film Title

Which is the most profitable title, and during which showtime group is it shown?

Answer: Dune: Part Two is the most profitable title (€45.17 net profit), shown during Evening showtimes. All other titles recorded losses after licensing costs.

```sql
SELECT
    f.Title,
    ROUND(SUM(mt.Total_Amount), 2)                          AS
       Total_Revenue,
    f.Licensing_Cost,
    ROUND(SUM(mt.Total_Amount) - f.Licensing_Cost, 2) AS Net_Profit,
    sg.Name                                                  AS
       Showtime_Group
FROM movie_ticket mt
JOIN showtime       s   ON mt.Showtime_ID        = s.Showtime_ID
JOIN film           f   ON s.Film_ID             = f.Film_ID
JOIN showtime_group sg  ON s.Showtime_Group_ID = sg.Group_ID
WHERE mt.Status_ID != 3    -- Exclude cancelled tickets
GROUP BY f.Film_ID, f.Title, f.Licensing_Cost, sg.Name
ORDER BY Net_Profit DESC
LIMIT 5;
```

Title	Revenue (€)	Lic. Cost (€)	Profit (€)	Showtime
Dune: Part Two	1,545.17	1,500.00	45.17	Evening
Deadpool & Wolverine	1,150.71	1,200.00	-49.29	Evening
Inside Out 2	660.66	1,000.00	-339.34	Afternoon
Gladiator II	753.83	1,100.00	-346.17	Evening
Wicked	347.27	900.00	-552.73	Evening

6.2 Q2 — Popcorn & Coke Combo Purchase Rate

What percentage of concession tickets include both popcorn and Coca-Cola?

Answer: 68.27% of customers purchase both popcorn AND Coca-Cola on the same concession ticket (71 out of 104 total orders).

```
SELECT
    s.Total_Tickets,
    s.Tickets_With_Both,
    CASE
        WHEN s.Total_Tickets = 0 THEN 0
        ELSE ROUND(s.Tickets_With_Both * 100.0 / s.Total_Tickets, 2)
    END AS Percentage
FROM (
    SELECT
        COUNT(DISTINCT st.Shop_Ticket_ID) AS Total_Tickets,
        SUM(
            CASE
                WHEN has_popcorn.Shop_Ticket_ID IS NOT NULL
                 AND has_coke.Shop_Ticket_ID    IS NOT NULL
                THEN 1 ELSE 0
            END
        ) AS Tickets_With_Both
    FROM shop_ticket st
    LEFT JOIN (                         -- Tickets containing popcorn
        SELECT DISTINCT si.Shop_Ticket_ID
        FROM shop_item si
        JOIN product p          ON si.Product_ID  = p.Product_ID
        JOIN product_category pc ON p.Category_ID = pc.Category_ID
        WHERE pc.Name = 'Popcorn'
    ) AS has_popcorn ON st.Shop_Ticket_ID =
       has_popcorn.Shop_Ticket_ID
    LEFT JOIN (                         -- Tickets containing
       Coca-Cola
        SELECT DISTINCT si.Shop_Ticket_ID
        FROM shop_item si
        JOIN product p ON si.Product_ID = p.Product_ID
        WHERE p.Name LIKE '%Coca-Cola%'
    ) AS has_coke ON st.Shop_Ticket_ID = has_coke.Shop_Ticket_ID
) AS s;
```

Total Tickets	Tickets With Both	Percentage (%)
104	71	68.27

6.3 Q3 — Peak Sales Showtime Group

At which showtime group do we record the most combined sales (movie tickets + shop)?

Answer: Evening dominates with 242 total transactions, combining both movie ticket sales (linked directly via showtime) and concession orders (matched via purchase time ranges).

```sql
SELECT
    Showtime_Group,
    SUM(Sale_Count) AS Total_Sales
FROM (
    -- Movie ticket sales per showtime group (direct FK link)
    SELECT
        sg.Name  AS Showtime_Group,
        COUNT(*) AS Sale_Count
    FROM movie_ticket mt
    JOIN showtime      s  ON mt.Showtime_ID      = s.Showtime_ID
    JOIN showtime_group sg ON s.Showtime_Group_ID = sg.Group_ID
    WHERE mt.Status_ID <> 3
    GROUP BY sg.Name

    UNION ALL

    -- Shop sales per group (match purchase time to time ranges)
    SELECT
        sg.Name  AS Showtime_Group,
        COUNT(*) AS Sale_Count
    FROM (
        SELECT Shop_Ticket_ID,
               CAST(Purchase_DateTime AS TIME) AS Purchase_Time
        FROM shop_ticket
    ) st
    JOIN showtime_group sg
        ON st.Purchase_Time BETWEEN sg.Start_Range AND sg.End_Range
    GROUP BY sg.Name
) combined
GROUP BY Showtime_Group
ORDER BY Total_Sales DESC;
```

Showtime Group	Total Sales
Evening	242
Afternoon	89
Morning	28

6.4 Q4 — Monthly Revenue per Location (12 Months)

Show monthly revenue broken down by cinema location across the full year.

Answer: CinemaChain Gran Via (Madrid) consistently leads revenue. Output was exported to CSV and visualised as a line chart in Excel.

```
SELECT
    Month_Name,
    ROUND(SUM(CASE WHEN Location_Name = 'CinemaChain Centro'
                   THEN Total_Revenue ELSE 0 END), 2) AS
                      CinemaChain_Centro,
    ROUND(SUM(CASE WHEN Location_Name = 'CinemaChain Diagonal'
                   THEN Total_Revenue ELSE 0 END), 2) AS
                      CinemaChain_Diagonal,
    ROUND(SUM(CASE WHEN Location_Name = 'CinemaChain Gran Via'
                   THEN Total_Revenue ELSE 0 END), 2) AS
                      CinemaChain_Gran_Via
FROM (
    SELECT Location_Name,
           MONTHNAME(DateValue) AS Month_Name,
           MONTH(DateValue)     AS Month_Number,
           SUM(Revenue)         AS Total_Revenue
    FROM (
        SELECT l.Name              AS Location_Name,
               mt.Purchase_DateTime AS DateValue,
               mt.Total_Amount      AS Revenue
        FROM movie_ticket mt
        JOIN location l ON mt.Location_ID = l.Location_ID
        WHERE mt.Purchase_DateTime >= '2025-01-01'
          AND mt.Status_ID <> 3
        UNION ALL
        SELECT l.Name              AS Location_Name,
               st.Purchase_DateTime AS DateValue,
               st.Total_Order       AS Revenue
        FROM shop_ticket st
        JOIN location l ON st.Location_ID = l.Location_ID
        WHERE st.Purchase_DateTime >= '2025-01-01'
    ) AS combined
    GROUP BY Location_Name, MONTHNAME(DateValue), MONTH(DateValue)
) AS summarised
GROUP BY Month_Name, Month_Number
ORDER BY Month_Number;
```

Month	Centro (€)	Diagonal (€)	Gran Via (€)
January	72.00	128.26	343.06
February	72.60	110.72	284.97
March	84.10	150.04	675.22
April	63.53	91.96	189.98
May	72.00	152.48	0.00
June	63.53	88.94	193.01
July	78.05	113.74	357.58
August	91.96	303.12	0.00
September	78.05	130.68	288.00
October	78.65	110.72	289.81
November	72.00	137.94	445.91
December	72.60	206.32	0.00

6.5 Q5 — Revenue by Auditorium Type (Custom Query)

Which auditorium type generates the highest average revenue per ticket?

Answer: VIP auditoriums lead on average revenue per ticket (€30.53), while IMAX generates the highest total revenue (€1,571.79) due to significantly higher ticket volume. For expansion decisions, IMAX offers the best balance of premium pricing and demand.

```
SELECT
    at.Name                               AS Auditorium_Type,
    COUNT(mt.Movie_Ticket_ID)             AS Tickets_Sold,
    ROUND(AVG(mt.Total_Amount), 2)        AS Avg_Revenue_Per_Ticket,
    ROUND(SUM(mt.Total_Amount), 2)        AS Total_Revenue
FROM movie_ticket mt
JOIN showtime         s   ON mt.Showtime_ID         = s.Showtime_ID
JOIN auditorium       a   ON s.Auditorium_ID        = a.Auditorium_ID
JOIN auditorium_type at ON a.Auditorium_Type_ID =
   at.Auditorium_Type_ID
WHERE mt.Status_ID <> 3
GROUP BY at.Auditorium_Type_ID, at.Name
ORDER BY Avg_Revenue_Per_Ticket DESC;
```

Auditorium Type	Tickets Sold	Avg Rev / Ticket (€)	Total Rev (€)
VIP	13	30.53	396.88
IMAX	69	22.78	1,571.79
4DX	36	19.93	717.53
3D	24	15.43	370.26
Regular	113	12.40	1,401.18

SQL files (DDL schema and DML sample data) are available in the project GitHub repository.