Team Members: Jesus Rodriguez, Julio Jimenez
Project Title: CinePictures
Group Number: Team 52

PROJECT URL:
Frontend URL: http://classwork.engr.oregonstate.edu:32827/

a) Fixes based on Feedback from Step 1:
These were the fixes and changes we implemented based on feedback from students and TAs:

1. More numerical statistics were added to highlight the need for a database. Details such as a loss of 5% (around $60,000 annually), the number of monthly ticket sales (10,000 tickets sold per month), and the management of 20 staff members were added to the overview.
2. Removed the redundant relationship between the Movies entity and the Tickets entity.
    a. However, to make up for a lost relationship, we added two new entities (Employees and EmployeeRoles entities) to maintain the minimum 4 relationship requirement.
3. Updated the ERD diagram to ensure Crow's foot notation appears on both sides of the lines.
    a. Updated the ERD diagram to ensure no crossing lines
4. Enforced consistent naming conventions (camelCase) in the schema and corrected attribute formatting across all entities.
5. Explicitly labeled the **M:N** in the Database outline - Tickets entity.
6. Removed the seatNumber attribute from Tickets to simplify the design.
    a. Instead added the totalCapacity attribute to Screenings to manage seating availability without complex seat tracking.

We opted not to enforce ticket seat uniqueness at the database level to maintain simplicity, delegating this to application logic instead.We also chose not to split Directors into a separate table to avoid unnecessary complexity given the project scope.

Fixes based on Feedback from Step 2:
- As of the May 4, 2025, there was no feedback.

Feedback by the peer reviewers and TA from Step 3 - DRAFT:

Tamithe Beyer
20 hours ago

- ***Does the UI utilize a SELECT for every table in the schema?*** **In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.** Yes, the UI utilized a SELECT for every table in the schema.
- ***Does the UI implement an INSERT form for at least one table in the schema?*** **In other words, there should be UI input fields that correspond to at least one table.** Yes, there is a CREATE (or INSERT) form at the top of each of the table UIs.
- ***Does the UI have at least one DELETE for any one entity?*** **In other words, is there a form/link/button that will allow the deletion of a row in at least one table?** Yes, the UI has at DELETE buttons for all entities.
- ***Does the UI have at least one DELETE that will remove things from a M:M relationship?*** **In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.** Yes, the UI has at least one DELETE that will remove things from a M:M relationship.
- ***Is there at least one UPDATE form in the UI for any one entity?*** **In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?** Yes, there are UPDATE forms in the UI for all entities.
- ***Is there at least one UPDATE form in the UI to modify an M:M relationship?*** **In other words, does the UPDATE allow the user to select a different foreign key value to update the intersection table with?** Yes, there is at least one UPDATE form in the UI to modify an M:M relationship.
- ***Do you have any other suggestions for the team to help with their HTML UI? For example, using AS aliases to replace obscure column names such as fname with First Name.*** Yes, rename headers such as roleName with "Role" or firstName with "First Name" using aliases to make your HTML UI more readable. You could also use join queries to replace columns such as movieID and employeeID with actual names instead of just showing IDs which would make the tables easier to understand.
- **As a reviewer, clearly describe to what extent your feedback to the team was original (e.g. "all my work") or non-original (e.g. used AI tools per Code Citation Tips).** My feedback to the team was all original work.

Comment •••

Comments:

Great Work Group 52!

If Tickets is your intersection table (M:M relationship) between Customers and Screenings then you will need to have an Update statement in your DML file.
Per project guide: "It should be possible to UPDATE an M:M relationship"

Roshan Patel, May 11 at 10:28pm

Adding the additional feedback from STEP 3 - DRAFT:

D

Drake Vickers 6d

Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

NO, although the sql is built to have a SELECT for all tables within the schema.

Does the UI implement an INSERT form for at least one table in the schema? In other words, there should be UI input fields that correspond to at least one table.

no, although the sql is built for a CREATE form that corresponds to at least one table.

Does the UI have at least one DELETE for any one entity? In other words, is there a form/link/button that will allow the deletion of a row in at least one table?

no, although the sql does have DELETE buttons for all entities.

Does the UI have at least one DELETE that will remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

no, although the SQL has at least one DELETE that corresponds to a M:M relationship.

Is there at least one UPDATE form in the UI for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

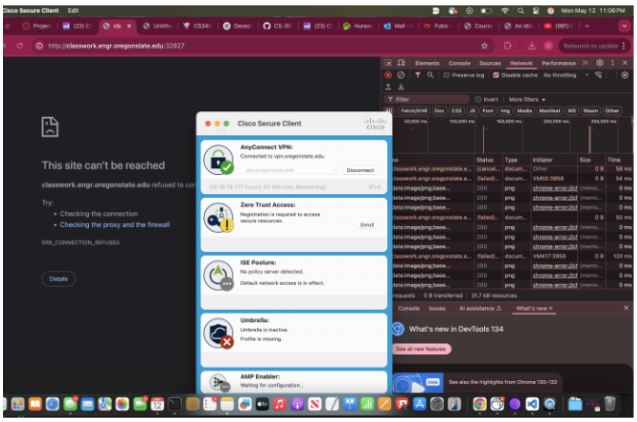
no, although there is at least one UPDATE forms in the SQL for every entity.

Is there at least one UPDATE form in the UI to modify an M:M relationship? In other words, does the UPDATE allow the user to select a different foreign key value to update the intersection table with?

No although there is at least one UPDATE form in the SQL to modify an M:M relationship.

Do you have any other suggestions for the team to help with their HTML UI? For example, using AS aliases to replace obscure column names such as fname with First Name.

While connected and trying to access the website on the engr vpn I am unfortunately not able to load the page. Not sure if the script wasn't ran properly But i am currently unable to view the actual UI. On the positive note the SQL file looks good and well adept to handle some html and js so the UI is probably fine. One note I do have is that grouping related input fields under fieldsets or using dropdowns for foreign key selections (e.g., customer names instead of customer IDs) would improve usability.



edback is original

Amarie Drollinger
6 days ago

Does the UI utilize a SELECT for every table in the schema?

Yes, the UI utilized a SELECT for every table in the schema. I am not sure if it's just me, but I can't get your link to load your webpage. I'm connected to the engr server- but it still isn't working, just so you know.

Does the UI implement an INSERT form for at least one table in the schema?

Yes, there is INSERT forms for many tables in the schema.

Does the UI have at least one DELETE for any one entity? In other words, is there a form/link/button that will allow the deletion of a row in at least one table?

Yes, all contain a DELETE option.

Does the UI have at least one DELETE that will remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

I see a DELETE for the intersection table but it doesn't look like it handles the above criteria. I don't see a CASCADE or any sort of handling to control the DELETE.

Is there at least one UPDATE form in the UI for any one entity?

Yes, there are many UPDATE forms in the UI.

Is there at least one UPDATE form in the UI to modify an M:M relationship? In other words, does the UPDATE allow the user to select a different foreign key value to update the intersection table with?

From what I can see there is no edit/update in the intersection table.

Do you have any other suggestions for the team to help with their HTML UI?

Besides what I have already included in the review, I have nothing else to add. Great job, from what I can see!

As a reviewer, clearly describe to what extent your feedback to the team was original (e.g. "all my work") or non-original (e.g. used AI tools per Code Citation Tips).

All my own work.

Comment •••

Add comment

Trevor Foote
6 days ago

- ***Does the UI utilize a SELECT for every table in the schema?*** **In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.**

Yes, the UI implements SELECT queries for each table. Data from each of these tables is also presented separately instead of using a single comprehensive joined query.

- ***Does the UI implement an INSERT form for at least one table in the schema?*** **In other words, there should be UI input fields that correspond to at least one table.**

Yes, the UI has structured INSERT forms for multiple entities. Each form corresponds clearly to the attributes defined in the respective database tables.

- ***Does the UI have at least one DELETE for any one entity?*** **In other words, is there a form/link/button that will allow the deletion of a row in at least one table?**

Yes, DELETE functionality is well-implemented in the UI for entities like Movies / Customers. These deletion options allow individual records to be removed from their tables.

- ***Does the UI have at least one DELETE that will remove things from a M:M relationship?*** **In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.**

Yes, when a Screening or Customer is deleted from the UI, related Tickets records in the intersection table (Tickets) are also removed using ON DELETE CASCADE in the schema. This also doesn't delete related Movies or Employees records.

- ***Is there at least one UPDATE form in the UI for any one entity?*** **In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?**

Yes, the UI provides working UPDATE forms for several entities like Movies / Employees.

- ***Is there at least one UPDATE form in the UI to modify an M:M relationship?*** **In other words, does the UPDATE allow the user to select a different foreign key value to update the intersection table with?**

The UI doesn't have a way to update foreign key selections in the Tickets intersection table.

- ***Do you have any other suggestions for the team to help with their HTML UI? For example, using AS aliases to replace obscure column names such as fname with First Name.***

No, it was all good.

- **As a reviewer, clearly describe to what extent your feedback to the team was original (e.g. "all my work") or non-original (e.g. used AI tools per Code Citation Tips).**

Citation for use of AI Tools: 05/13/2025. Used ChatGPT 4.5 for summarizing things and brainstorming ideas. I wrote the feedback. Source URL: chatgpt .com

Comment •••

Fixes based on Feedback from Step 3 - DRAFT:
These were the fixes and changes we implemented based on feedback from students and TAs:

1. From the **TA**: We added the missing UPDATE SQL query for the Tickets entity
2. From Peer **Tamithe**: Made the table header easier to read:
    a. Instead of "firstName", it now displays as "First Name"
3. From a Peer **Tamithe**: Displayed more meaningful FKs in some entities:
    a. In entities such as Employees and Screenings, some foreign key fields were reformatted to display as "ID – Name" instead of showing only the ID value.
        i. Employees entity: EmployeeRoles now display as "1 - Manager"
        ii. Screenings entity: Movies now displays as "1 - The Avengers: Doomsday"

We decided to not to implement the "ID – Name" format for all FKs since we felt that it clouded the tables with unnecessary data that could be quickly easily retrieved from the original entity. For example, an employee ID is usually enough to identify an employee in the backend without including full names.

UPDATED FIXES from remaining reviews from Step 3 - Draft:
Drake Vickers: He had issues accessing our website since our servers were down during his review, as we were implementing the changes to the database. Our servers are now working properly.
Amarie Drollinger: his review did not request any new changes beyond those already identified in our previous feedback.
Trevor Foote: Trevor confirmed that our SELECT, INSERT, DELETE, and UPDATE work properly. The only suggestion was to add an UPDATE form for modifying M:M relationships, which we implemented now.

Fixes based on Feedback from Step 3 - FINAL DRAFT:

Great Work Group 52!

Reasons for Deductions:

- Per rubric, needed to include verbatim feedback. Your step 3 draft has 3 reviews but there is only 1 verbatim review on the pdf.
- DDL file still shows signs of being an SQL dump (ex. ENGINE=InnoDB;). DDL file should be completely hand-authored at this stage in the project.

Other Comments:
Per project guide: "It's not acceptable to require the user to enter IDs (e.g. FK, PK, SKU number, etc.) into an HTML textbox. Instead, your website needs to use a dynamically populated drop-down list with sensible labels OR offer the ability to search using text for foreign key selection (instead of entering in the ID)"

For example, in your create form for the Screenings page, the user should not be able to manually enter the foreign keys: employeeID and movieID. These should be a drop down or search feature. This applies to all foreign key/primary key selections in your project in all your forms (create, update)

Roshan Patel, May 18 at 9:07pm

Fixes based on Feedback from Step 3 - FINAL DRAFT:
These were the fixes and changes we implemented based on feedback from the TA:

1. **Include all 3 verbatim peer reviews:** We went back a second time and included every feedback we saw.
2. **Clean up your DDL.SQL file:** We cleaned up the DDL.sql file, removing dump like leftovers from when we forward engineered the tables.
3. **Fixed Foreign Key Inputs in Forms:** We fixed the foreign keys inputs in the create forms and editing forms. Now, they are displayed in a dropdown feature in both forms.

b) Project Outline and Database Outline - Updated Version:
Overview:

A local movie theater called CinePictures is looking to update its day-to-day operations. They plan to replace their old manual record-keeping ways for movie

scheduling, screen management, and customer information with a more reliable and robust database. As of now, the theater operates 5 screens, selling an average of 10,000 tickets a month while currently employing 20 staff members with various roles. With the old system, not only was it difficult to keep track of ticket sales and scheduling movie screenings, but it also became difficult to track which employee was in charge of which screening, disrupting the workflow. Even though CinePictures generates approximately $100,000 a month, its old system caused many angry customers from operational errors like inefficient daily movie scheduling and inaccurate ticket information, which caused a loss of 5% of the annual revenue (around $60,000).

A movie theater database system will not only prevent these errors but will also improve its operational workflow. CinePictures will now be able to schedule movies efficiently on each screen by properly managing start and end times throughout the day. They can maintain accurate ticket sale records that include customer information, seat numbers, screening details, pricing, and purchase dates. The database will also allow them to track employee assignments for every screening, better allowing them to manage staffing needs. In addition, the movie theater database system will also allow them to analyze their customer preferences and viewing history through customer records. With this database, they can accurately track performance metrics to determine which movies generated the most ticket sales and which showtimes had the highest attendance. The data from the database will then be able to help the owners make the proper business decisions to run the business successfully, in order to maximize profits and improve efficiency.

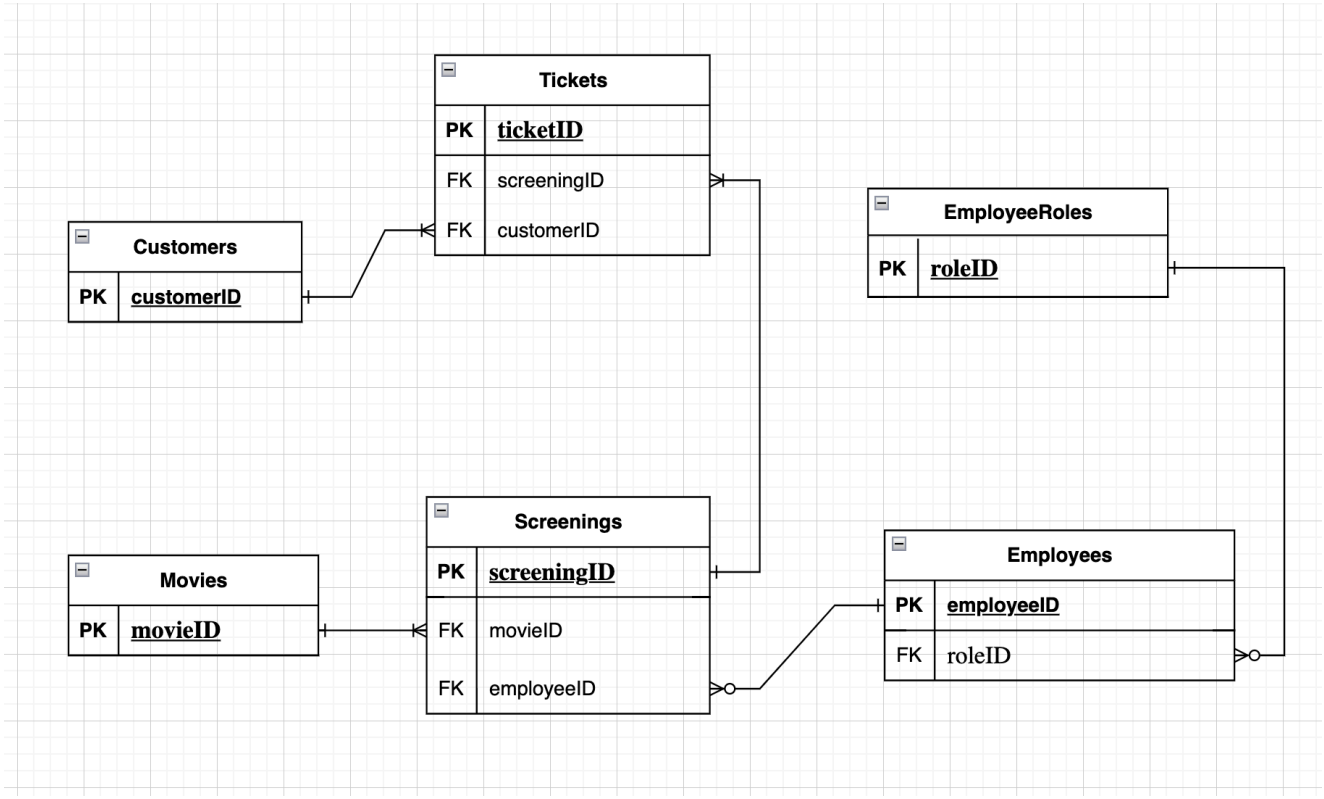
Database Outline, in Words:

- **Movies:** keeps track of the information relating to the movies showed in the theater (title, rating, genre etc…)
    - movieID: int, auto_increment, not NULL, unique, PK
    - title: varchar(255), not NULL
    - director:varchar(100), NULL
    - releaseYear: int, NULL
    - genre: varchar(100), NULL
    - runtime: int, NULL
    - rating: varchar(10), NULL - (like PG, G, R)
    - **Relationship:** 1:M relationship with Screening, implemented with movieID as FK

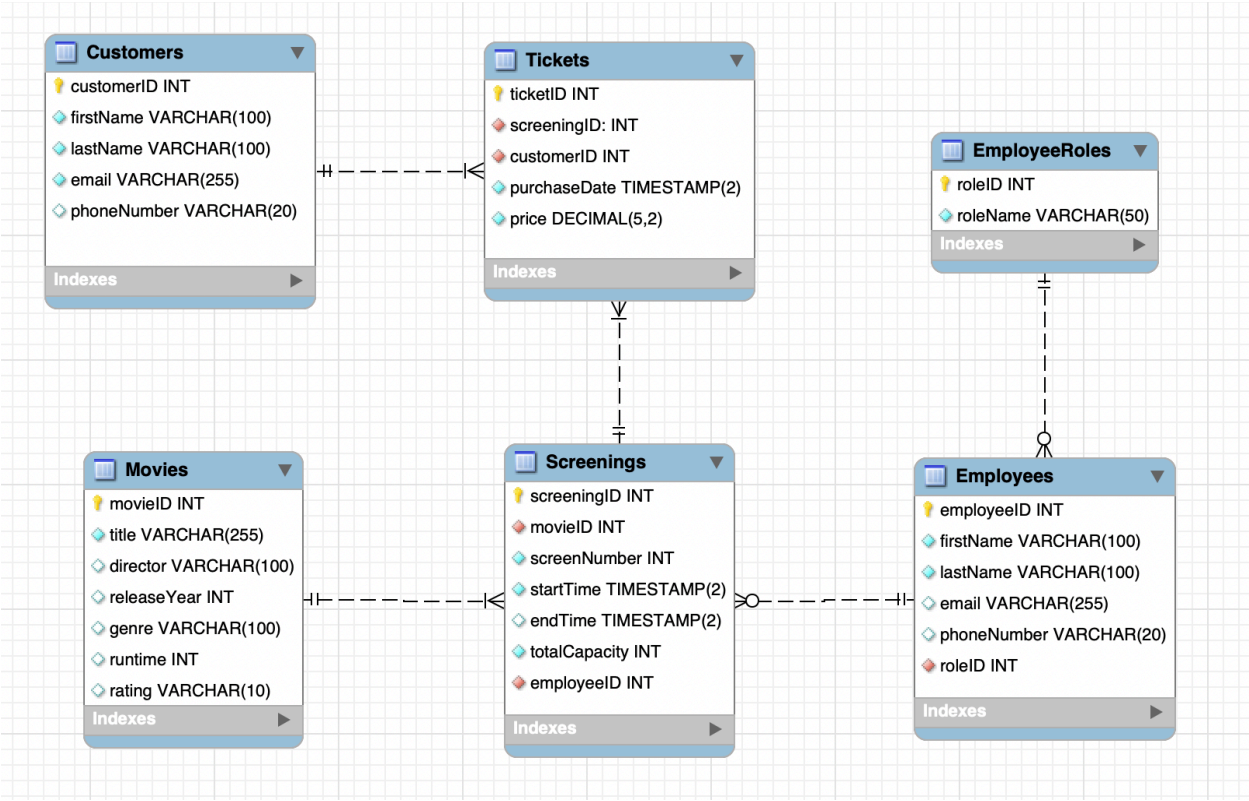
- **Screenings:** keeps track of all the scheduled showings of a movie, including start time, end time, and screen number
    - screeningID: int, auto_increment, not NULL, unique, PK
    - movieID: int, not NULL, FK referencing Movies(movieID)
    - screenNumber: int, not NULL
    - startTime: timestamp, not NULL
    - endTime: timestamp, NULL
    - totalCapacity: int, not NULL
    - employeeID: int, not NULL, FK referencing Employees(employeeID)
    - **Relationship:**
        - M:1 with Movies
        - M:1 with Employees
        - 1:M with Tickets, implemented with screeningID as FK in Tickets
- **Tickets:** keeps track of every purchase made by customers for specific screenings/movies, including price and seat number.
    - ticketID: int, auto_increment, not NULL, unique, PK
    - screeningID: int, not NULL, FK referencing Screenings(screeningID)
    - customerID: int, not NULL, FK referencing Customers(customerID)
    - purchaseDate: timestamp, not NULL
    - price: decimal(5,2), not NULL
    - **Relationship:** Join table for Customers and Screenings.
        - M:1 with Customer and Screening
        - Customers and Screenings have a **Many-to-Many (M:N)** relationship, implemented through the Tickets table.
- **Customers:** keeps track of all the personal information of customers who have purchased tickets
    - customerID: int, auto_increment, not NULL, unique, PK
    - firstName: varchar(100), not NULL
    - lastName: varchar(100), not NULL
    - email: varchar(255), not NULL, unique
    - phoneNumber: varchar(20), NULL
    - **Relationship:** 1:M with Tickets, implemented with customerID as FK in Tickets.
- **Employees:** keeps track of employee information like first name, role, and phone number.
    - employeeID: int, auto_increment, not NULL, unique, PK
    - firstName: varchar(100), not NULL
    - lastName: varchar(100), not NULL

    - email: varchar(255), NULL, unique
    - phoneNumber: varchar(20), NULL
    - roleID: int, not NULL, FK referencing EmployeeRoles(roleID)
    - **Relationship:**
        - 1:M with Screenings, implemented with employeeID as FK in Screenings. 1 employee can manage multiple screenings
        - M:1 with EmployeeRoles, implemented with roleID as FK in EmployeeRoles. Multiple employees can work under the same role.
- **EmployeeRoles**: keeps track of employee role types through a Category table
    - roleID: int, auto_increment, not NULL, unique, PK
    - roleName: varchar(50), not NULL, unique
    - **Relationship**: 1:M with Employees, implemented with roleID as FK in Employees. One role can be possessed by multiple employees

c) Entity-Relationship Diagram:



d) Schema:



e) Example Data:

Customers Dummy Data

customerID	firstName	lastName	email	phoneNumber
1	David	Johnson	david.johnson@gmail.com	555-5001-0001
2	Gabriella	Hernandez	gabriella.h@gmail.com	555-5002-0002
3	Peter	Parker	peter.parker@gmail.com	555-5003-0003
4	Juan	Santos	juan.santos@gmail.com	555-5004-0004
5	Adam	Sandler	adam.s@gmail.com	555-5005-0005

Movies Dummy Data

movieID	title	director	releaseYear	genre	runtime	rating
1	The Avengers: Doomsday	A. Director	2025	Action	120	PG-13
2	Dune Messiah	D. Villeneuve	2025	Mystery	110	PG-13
3	Sinners	R. Coogler	2025	Thriller	135	R
4	The Odyssey	C. Nolan	2025	Drama	195	PG-13
5	The Batman	M. Reeves	2022	Thriller	205	PG-13

EmployeeRoles Dummy Data

roleID	roleName
1	Manager
4	Projectionist
3	Ticket Agent
2	Usher

Employees Dummy Data

employeeID	firstName	lastName	email	phoneNumber	roleID
201	Eveyln	Garcia	eveyln.garcia@cinepic.com	555-0101-0001	1
202	Isabella	Johnson	isabella.johnson@cinepic.com	555-0102-0002	2
203	Charlie	Goldberg	charlie.goldberg@cinepic.com	555-0103-0003	3
204	Danny	Prince	danny.p@cinepic.com	555-0104-0004	2
205	Mateo	Huminsky	mateo.huminsky@cinepic.com	555-0105-0004	4

Screenings Dummy Data

screeningID	movieID	screenNumber	startTime	endTime	totalCapacity	employeeID
11001	1	1	2025-04-29 14:00:00.00	2025-04-29 16:00:00.00	150	202
11002	5	5	2025-04-29 17:00:00.00	2025-04-29 19:25:00.00	50	205
11003	2	2	2025-04-29 15:30:00.00	2025-04-29 17:30:00.00	75	202
11004	3	3	2025-04-29 19:00:00.00	2025-04-29 21:15:00.00	75	204
11005	4	2	2025-04-29 18:00:00.00	2025-04-29 21:15:00.00	75	205

Tickets Dummy Data

ticketID	screeningID	customerID	purchaseDate	price
1	11001	4	2025-04-28 10:00:00.00	15.00
2	11001	4	2025-04-28 10:00:00.00	15.00
3	11003	1	2025-04-28 11:30:00.00	12.50
4	11004	2	2025-04-28 14:00:00.00	12.50
5	11001	3	2025-04-28 16:00:00.00	15.00
6	11005	3	2025-04-28 17:00:00.00	12.50
7	11002	5	2025-04-28 15:00:00.00	10.00