Group 25 (Konami Code) - Project Step 5 Draft

Team members: Jesseline Velazquez, Anthony L Clary
Project Title: IncrediFilms, a Cinema Experience
Website: http://flip1.engr.oregonstate.edu:40593/

Project Step 5 Draft Post

- **What technology are you using (e.g. NodeJS/Flask Python)?**
    - Frontend is powered by the React Framework. We are utilizing the Grid.js library to provide featureful UI tables, with search and sort capabilities. Some CSS styling is provided by the lightweight Pure.css modular CSS set. API calls are made from the frontend by the Axios HTTP client.
    - Backend is powered by the Express web framework for Node.js. Calls to the MySQL DB are facilitated by the mysql connector/driver for Node.js.
- **What works?**
    - CRUD operations are implemented or work in progress as broken down in the table on the next page.
    - The customer's entity also has a functioning same-page modal form implemented. That is, shared form for edit and new customer creation.
    - Please see http://flip1.engr.oregonstate.edu:40593/customer for a demo of the current site.
- **What doesn't work?**
    - Filter on the Showtimes page remains to be implemented.
    - Create and Update operations remain to be fully implemented for the Showtime, Ticket, and MovieGenre tables. A major blocker for implementing Create and Update for these tables (with foreign keys) was to ensure that the entry forms for these tables were being fed live data from the server for the FK fields. The first iteration of this has now been implemented on the /ShowtimeNew page—-those drop downs are populated from API calls.
- **Where/why you are blocked.**
    - No major blockers. Above items need to be addressed.

- Entity CRUD Operation Status:

ENTITY	C - required	R - required	U	D	FILTER	Other adjustments
CUSTOMER	X	X	X	X		Date validation
MOVIE	X	X	X	X		Right now have the mpa ratings copy/pasted into forms and new… maybe separate into its own file if time permits
THEATER	X	X	X	X		
GENRE	X	X	X	X		Doesn't give an error for duplicate records. Just doesn't add and links back to genre page. Should it?
SHOWTIME	Need to update form to pull dynamic data	X	Need to update form to pull dynamic data	X	Between these two dates	
TICKET	Need to update form to pull dynamic data	X	Need to update form to pull dynamic data	X		
MOVIEGENRE	Need to update form to pull dynamic data	X	Need to update form to pull dynamic data	X		

Step 4 Feedback from Peer Reviewers

Reviewer 1

Do the implemented CRUD steps function as the team expects (e.g. if the team stated that a CRUD step worked but you found an error, please tell them)?

Create:

The CREATE route seems to work fine when using the add new customer form. The only problem I encountered was that the form does not re-route to the table page once a customer is created. This led to me mistakenly pressing submit multiple times creates the same customer over and over. You could fix this by either routing to '/customer' on submit, or clearing the form inputs on submit (if you wanted you could give an alert that says "Customer added, would you like to add another one?" and then do either of the prior options (Edit: just seeing now that you addressed this in your ed post.)

Read:

The SELECT query for the table view works as expected. Also, the search function narrows down the results properly. Also, when pressing the update button, the form data is SELECTed properly and fills the text boxes, on the first press (but not the second press…see below).

Update:

The update functionality works and updates the table. There is a minor bug that could probably be fixed. If I press update, it properly fills the form, but if you press another update button without pressing cancel, it does not change the data in the form to the newly pressed entity. One example is if I had "Jenna" and "Jacob" and I pressed update for Jenna, but then I pressed update for Jacob. The form is now set up to update Jacob, but Jenna's information is still in the form. If I submit without changing anything, Jacob's entity will become a copy of Jenna and Jacobs data will be lost. This could probably be amended with an additional read/SELECT when pressing a new update button.

Delete:

Delete button works properly and I found no issues.

Would a user easily be able to use the UI to complete the step? If not or you have suggestions for how the UI can be improved, please elaborate.

Your UI is very slick and inspiring, it's making me realize the possibilities that my group can strive for. I have one minor suggestion though. When clicking Update button, auto-scroll the page down to the form for a better user experience. As it stands, If the table is larger than the page height, a new user would not know that they need to scroll down to find the update form at the bottom.

What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed helpful feedback will receive higher credit.

Not many other suggestions on my end. Your code is very neat and is helping me think about react better and in a more organized way. Great job!

Reviewer 2

Do the implemented CRUD steps function as the team expects (e.g. if the team stated that a CRUD step worked but you found an error, please tell them)?

I thought the edit/update was not working at first because of the behavior Zachary describes. Also, the form being under the table means if you click the edit item icon, but don't know where to look for the form, you could miss that it opened up.

For the add issue not navigating away from the page: I think you have the right idea in your code comments. We have a few conditionals based on the server response (if response.status === 200 {}), and then at the end we utilize useNavigate from react-router-dom to navigate back to the entity's homepage. The statuses are set on the back end.

Would a user easily be able to use the UI to complete the step? If not or you have suggestions for how the UI can be improved, please elaborate.

The margins on the side of the Customers page could maybe be increased so that the table does not take up the entire page width wise?. Totally a stylistic thing, it's just that opening dev tools or decreasing the window size ends up making me (on chrome) have to scroll left and right to see all of the columns of the table. The add new customer button is a little camouflaged where it is currently on the page and it might be more user friendly if it was more obvious about being clickable. The way it is now, it looks just like the "Customers" header above the table which is not clickable.

What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed helpful feedback will receive higher credit. If the team is not blocked or having difficulty encouraging and supportive comments would be a better response than NO feedback.

First thing first, we used this site to help get started on our drop down selector (it's not working though, so…) It utilizes React-Select

https://medium.com/how-to-react/react-select-dropdown-tutorial-using-react-select-51664ab8b6f3

I haven't gone through this tutorial yet, but it was recommended too:

https://dev.to/antdp425/populate-dropdown-options-in-react-1nk0

I notice you have a component for confirming a customer delete, one thing I found helpful is the JS window.confirm which pops up an alert on click to make sure someone wants to do the thing. It allows the user to still cancel out of the action. We placed it before our delete function is called.

I think your idea about merging your add and update forms could be useful. In theory, you could do one customers form component that displays the edit data if called on click of the edit icon, maybe?

Reviewer 3

Do the implemented CRUD steps function as the team expects (e.g. if the team stated that a CRUD step worked but you found an error, please tell them)?

As other reviewers mentioned, the edit feature was confusing. Not sure how easy to implement it with react but the user can be taken to the bottom of the page where the editable fields can be seen easilySubmit feature can be improved with giving the user feedback.

Would a user easily be able to use the UI to complete the step? If not or you have suggestions for how the UI can be improved, please elaborate.

Nothing major here, but the call to action phrases can look more of a button such as 'add new customer'. Another suggestion would be to add some hover over effect to edit or delete buttons to indicate the user is engaging with them. Back to the first question UI is for sure clear for a user to complete the tasks.

What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed helpful feedback will receive higher credit. If the team is not

blocked or having difficulty encouraging and supportive comments would be a better response than NO feedback.

If I am not mistaken from CS 290, I used something like this to handle form submissions and giving feedback and taking user back to a different state if a component.This link can be helpful https://bobbyhadz.com/blog/react-redirect-after-form-submit

Good job putting it together with react!

Reviewer 4

Do the implemented CRUD steps function as the team expects (e.g. if the team stated that a CRUD step worked but you found an error, please tell them)?

C: I tried adding a customer and the form did not reroute me back to the main page where customers are displayed. The customer did appear to be added though.

R: Looks like it works well.

U: Update works well.

D: Delete works well.

Would a user easily be able to use the UI to complete the step? If not or you have suggestions for how the UI can be improved, please elaborate.

It seems pretty easy to navigate to edit, delete, and add a customer. I really like how the customer list displays real-time when we type in the search bar.

What suggestions do you have for the team in any areas where they are blocked or having difficulty? Detailed helpful feedback will receive higher credit. If the team is not blocked or having difficulty encouraging and supportive comments would be a better response than NO feedback.

I saw that the team did not identify any blockers. I think some functionality that could be implemented but not required is limiting the date of birth for a customer to the current date. I was able to add a customer born in 2026.

Step 4 Summary Feedback

1. Implement more data validation to set restrictions on the values that users can enter into our records.
2. Navigate back to the entity's home page after adding a new record for a better user experience.

Actions based on Feedback from Step 4

1. We rebuilt our add record functionality to display on the main page of each entity, instead of a new page.
2. We updated the onClick for editing records to route to the specific id.
3. *Implementing validation on dates so no customer's date of birth can be after today's date.*
4. *Adding an auto-scroll or a pop-up for the update.*

Upgrades Since Step 4 Draft

1. CRUD activities fully implemented for Movie entity
    a. CREATE - via form on http://localhost:3000/MovieNew and HTTP POST method to http://localhost:3001/movies endpoint
    b. READ - via HTTP GET method to http://localhost:3001/movies endpoint
    c. UPDATE - via edit record button on UI table, reuse of movie form, and HTTP PUT method to http://localhost:3001/movies/{movie_id} endpoint
    d. DELETE - via HTTP DELETE method to http://localhost:3001/movies/{movie_id} endpoint
2. CRUD activities fully implemented for Theater entity
    a. CREATE - via form on http://localhost:3000/TheaterNew and HTTP POST method to http://localhost:3001/theaters endpoint
    b. READ - via HTTP GET method to http://localhost:3001/theaters endpoint
    c. UPDATE - via edit record button on UI table, reuse of theater form, and HTTP PUT method to http://localhost:3001/theaters/{theater_id} endpoint
    d. DELETE - via HTTP DELETE method to http://localhost:3001/theaters/{theater_id} endpoint
3. CRUD activities fully implemented for Genre entity

a. CREATE - via form on http://localhost:3000/GenreNew and HTTP POST method to http://localhost:3001/genres endpoint

b. READ - via HTTP GET method to http://localhost:3001/genres endpoint

c. UPDATE - via edit record button on UI table, reuse of genre form, and HTTP PUT method to http://localhost:3001/genres/{genre_id} endpoint

d. DELETE - via HTTP DELETE method to http://localhost:3001/genres/{genre_id} endpoint

4. CRUD - Read and Delete activities fully implemented for Showtime entity; Create, Update functionality pending

   a. CREATE - via form on http://localhost:3000/ShowtimeNew and HTTP POST method to http://localhost:3001/showtimes endpoint

   b. READ - via HTTP GET method to http://localhost:3001/showtimes endpoint

   c. UPDATE - via edit record button on UI table, reuse of showtime form, and HTTP PUT method to http://localhost:3001/showtimes/{showtime_id} endpoint

   d. DELETE - via HTTP DELETE method to http://localhost:3001/showtimes/{showtime_id} endpoint

5. CRUD - Read and Delete activities fully implemented for Ticket entity; Create, Update functionality pending

   a. CREATE - via form on http://localhost:3000/TicketNew and HTTP POST method to http://localhost:3001/tickets endpoint

   b. READ - via HTTP GET method to http://localhost:3001/tickets endpoint

   c. UPDATE - via edit record button on UI table, reuse of ticket form, and HTTP PUT method to http://localhost:3001/tickets/{ticket_id} endpoint

   d. DELETE - via HTTP DELETE method to http://localhost:3001/tickets/{ticket_id} endpoint

6. CRUD - Read and Delete activities fully implemented for MovieGenre entity; Create, Update functionality pending

   a. CREATE - via form on http://localhost:3000/MovieGenreNew and HTTP POST method to http://localhost:3001/moviegenres endpoint

   b. READ - via HTTP GET method to http://localhost:3001/moviegenres endpoint

   c. UPDATE - via edit record button on UI table, reuse of movie genre form, and HTTP PUT method to http://localhost:3001/moviegenres/{moviegenre_id}

endpoint

   d. DELETE - via HTTP DELETE method to http://localhost:3001/moviegenres/{moviegenre_id} endpoint

7. The DML.sql file has been updated to contain queries that match the tables and needs of the current UI implementation.

Project Step 4 Draft Post

1. **What technology are you using (e.g. NodeJS/Flask Python)?**
    - Frontend is powered by the React Framework. We are utilizing the Grid.js library to provide featureful UI tables, with search and sort capabilities. Some CSS styling is provided by the lightweight Pure.css modular CSS set. API calls are made from the frontend by the Axios HTTP client.
    - Backend is powered by the Express web framework for Node.js. Calls to the MySQL DB are facilitated by the mysql connector/driver for Node.js.
2. **What works?**
    - CRUD operations for the Customer table entity are currently implemented. Please see http://flip1.engr.oregonstate.edu:40593/customer for a demo.
3. **What doesn't work?**
    - Form submission for creation and updating of records currently lacks sufficient UI feedback based on success or failure of API calls. We are considering using a small react-compatible toast notification library to implement success or failure messages based on the results of form submission.
    - We will need to implement a scalable drop-down selector for forms that will create or update our more complex entities (i.e. those with foreign keys). That is, drop-down selectors that perform calls to our API to populate selectable values from the database (e.g. all movies currently available, to schedule a showtime). The drop-down selector needs to be able to handle large amounts of entries. Therefore, we are considering something like React Select to facilitate this.
    - CRUD operations for entities other than Customer
    - React rendering is choppier than it should be. Investigate the issue.
    - Potentially merge new record creation into the same form as the "update" form. The same-page update form was built for this bi-modal task.
4. **Where/why you are blocked.**
    - No major blockers. Above items need to be addressed.

Upgrades/Changes Since Step 3 Final

- CRUD activities fully implemented for Customer entity
    - CREATE - via form on http://localhost:3000/CustomerNew and HTTP POST method to http://localhost:3001/customers endpoint
    - READ - via HTTP GET method to http://localhost:3001/customers endpoint
    - UPDATE - via edit record button on UI table, reuse of customer form, and HTTP PUT method to http://localhost:3001/customers/{customer_id} endpoint
    - DELETE - via HTTP DELETE method to http://localhost:3001/customers/{customer_id} endpoint
- Website frontend updated to utilize grid.js table library to display entity tables
    - CSS styling changes to accompany
- DDL alterations on Ticket and Showtime tables to set ON DELETE SET NULL conditions for foreign key dependencies. Additionally, corresponding FKs updated to all NULL values. Change was made in order to prefer data preservation in the event that entities associated with respective FKs are deleted. Outline / project document has been updated.

Step 3 Feedback from Peer Reviewers

Reviewer 1

- **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.**
    - All tables in the schema have a page on the site, and the UI seems to utilize a SELECT from all of these tables.
- **Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?**
    - Yes, there are a few SELECTS that search or filter by some attribute. There is a SELECT that searches by a customer's last name and then contacts that data.
- **Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.**
    - .Yes, there is a way to add data into every table from the UI. There are also INSERT statements for every table in the SQL file.
- **Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).**
    - Yes, adding data into the intersection table/UI MovieGenres adds the corresponding FK attributes to the table.
- **Is there at least one DELETE and does at least one DELETE 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 most of the tables in the UI have options to delete the record from the table. There is also corresponding DELETE statements in the SQL file.
- **Is there at least one UPDATE 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?**
    - The same thing with UPDATE too. Every record has a button for editing the values, and there are corresponding UPDATE statements for each table as well.

- **Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.**
    - Yes, in the Ticket table, the customer ID is a foreign key, but it is possible to be NULLable. It is possible to create a ticket and not have a customer attached to it.
- **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.**
    - I think the whole thing is great. My only suggestion, as an end user, is that adding data could happen right on the same page as viewing the data. It would be a better user experience if I didn't have to change pages to add data. But overall great job!

Reviewer 2

- ***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.**

    Yes, every single table in the database schema has its own page on the application, which displays the table data in a grid layout. This includes the intersection table which shows a M:N relationship between movies and genres.

- ***Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?***

    Yes. In the SELECT statements of the DML.sql file, I see a statement to retrieve all genres associated with a given movie. This searches/filters the genre table based on a movie ID, as a movie can have multiple genres associated with it.

- ***Does the UI implement an INSERT for every table in the schema?*** **In other words, there should be UI input fields that correspond to each table and attribute in that table.**

    Yes, above every table in the HTML UI there's a button to add a new record. These all redirect to new pages to add the record.

- ***Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?*** **In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).**

A genre and a movie can be added without the corresponding FK, according to the DML.sql file. However, inserting a value into the Movie_Genre table allows a movie/genre to be selected from a drop-down menu. I believe this is because associating a movie and a genre is optional, and is also the way I constructed the relationship in my project draft.

- ***Is there at least one DELETE and does at least one DELETE 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.**

  All of the tables in the UI have a delete column. There are delete statements in the DML.sql for all tables except the intersection table.

- ***Is there at least one UPDATE 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 updates for all tables except the intersection table. There is a comment in the SQL file saying that the Movie_Genre table is not permitted to be updated directly.

- ***Is at least one relationship NULLable?*** **In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.**

  Yes, in the DDL.sql, line 176, of the ticket table the customer_id which references the primary key of the customer table, does not have a 'NOT NULL' constraint on it, meaning that it can be left null. This means that a ticket cannot have a customer associated with it.

- ***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.***

  The UI looks really great. I like how the sidebar is on the left hand side. The only suggestion I would have is to add a paragraph about each table on the page to describe it and its relationship to other tables.

Reviewer 3

- ***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.**

For the most part yes this looks correct. I noticed that your schema for table movie_genre has a foreign key, movie_id. However, in your HTML/web app you've got movie name. It looks like you did the same thing in the showtime table. I don't know if this is intentional or not but my understanding was that the HTML should display the table from an "administrator perspective". I would think displaying the table exactly as it is represented in the database is the goal.

I would say here check if it's appropriate to represent the foreign keys differently. I think Instructor Curry may have had some examples in his videos where he talked about these foreign keys.

- ***Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?***

  I believe the SQL's for the drop down would be the three at the top under the simple lists section in the DML.sql file.

  I see the drop-downs but they sometimes appear to be a mix of id's and names. For example, when you click edit on the ShowTime table and you edit the movie name, there is a mix of ID's and movie names.

  I don't think your three SQL's are selecting the id's from the tables.

  Some of the drop downs are missing. For example, when you go to add a ShowTime, there's no drop down for theater. theater_id is a foreign key in ShowTime. I believe that in this case there should be a drop down.

- ***Does the UI implement an INSERT for every table in the schema?*** **In other words, there should be UI input fields that correspond to each table and attribute in that table.**

  The only issue here I noticed is that the MovieGenres page doesn't have a submit button.

- ***Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?*** **In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).**

  I honestly can't really tell if an entry would be inserted into the intersection table. It doesn't appear to me as though inserting a new movie or a new genre would automatically add an entry into the intersection table MovieGenres. My guess is really that the DBMS would determine that. I would say test the DBMS to see if that is the case.

- ***Is there at least one DELETE and does at least one DELETE 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.**

    It looks like you have a cascade statement in the DDL.SQL for the Movie_Genre table that would allow this to happen.

- ***Is there at least one UPDATE 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 it looks like you have updates available for several relations.

- ***Is at least one relationship NULLable?*** **In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.**

    I believe this is a yes. It looks like for your Ticket table you can set the foreign key customer_id to NULL.

- ***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.***

    One suggestion is to take a look at some of these dropdowns. I see a mix of ID's and names so it's a bit confusing to me. I already mentioned this in a previous answer.

    I'm actually going to try to make some of thees drop downs a combination of the foreign key and another row. For example, you could have "1 - Movie Name 1", "2 - Movie Name2", etc. That way if you have two movies with the same name, the drop down would be different. I That's what I'm going to be updating my dropdowns to.

    Also when you click on edit for Customers the HTML looks pretty weird. It doesn't seem to match the behavior that the other tables have.

Reviewer 4

- **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, every table in the schema has a separate page that displays the data.

- **Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?**

  There are a few selects in the DML.sql file that utilizes search filters. These don't appear to be implemented in the website forms yet.

- **Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.**

  Every entity in the schema has a form in the UI for adding new entries.

- **Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line_total).**

  Yes. the inserts for MovieGenres add the corresponding FK attributes and the relationship between Movies and Genres is M:N

- **Is there at least one DELETE and does at least one DELETE 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.**

  There is no delete for the intersection table of MovieGenres

- **Is there at least one UPDATE 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?**

  Every entity has a form for updating. It looks like the update for Ticket in the DML.sql file has not been finished yet. Also, there is not a update query for MovieGenres.

- **Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order.**

Thus it should be feasible to edit an Order and change the value of Employee to be empty.

customer_id attribute in Ticket table may be NULL.

- **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.**

  My only suggestion is to maybe implement filters for your entities. The pages could fill up really quickly.

Step 3 Summary Feedback

1. Data entry / add record functionality built into the same page as tables are displayed, instead of navigating away to a dedicated form page.
2. Add a paragraph or description for each table.
3. MovieGenres table needs to be updated to reflect that table (i.e. fully implemented).
4. Consider if some dropdowns should convey more information than single attributes in some places, to avoid confusion among similar entries.
5. Some mismatch between DML.sql queries and what is / will be ultimately implemented on the website.
6. At least one website table should make use of a dynamic variable within a SELECT statement (i.e. filtering query).
7. No delete for MovieGenres table.
8. Implement filters for each entities.

Actions based on Feedback from Step 3

1. Implementing data entry / add new record functionality on the same page as tables noted as a possible future feature. We will consider if this can ultimately be implemented. It will require either the form to be displayed above or below the table or in a virtual pop-up entry form. At this time, entry forms will remain on dedicated pages.
2. We have added a short description for each entity on their respective pages.
3. The MovieGenres table has been updated to reflect its layout as an intersection table and has been loaded with all of its associated sample data.
4. This is a valid point, we had made some attempt to avoid dropdown entries that could be confused/duplicated, but we can do better:
    a. **Genre dropdown** - The DDL definition for the Genre table has been updated so that the genre_name attribute is also UNIQUE. The Overview section below has also been updated.
    b. Movie dropdown - The DML query that will be used to create a list for a drop-down selector has been updated. The drop down list will now display ‘movie_name (movie_id).’ This approach was chosen, as there may be circumstances where a movie_name is the same for multiple records (e.g. extended runtime version, differently rated version, etc).
    c. Theater dropdown - The DDL definition for the Theater table has been updated so that the theater_name attribute is also UNIQUE. The Overview section below has also been updated.
    d. Customer dropdown - The DML query for this list was already built to produce unique entries. Format: ‘last, first (customer_id)’
5. The DML.sql file has been updated to contain queries that explicitly match the tables and needs of the current UI implementation.
6. The Showtimes page has been updated with a planned timespan selector. That is, in our final implementation, the user will be able to fetch a table of showtimes between two

given dates. The given dates will be inserted into a query via our API and the SQL will retrieve results based on use of a WHERE clause with the BETWEEN statement. See the DML.sql file for the query.

7. The Movie_genre table UI now implements deletion buttons for records.
8. We have reimplemented all tables via the Grid.js library. This design decision gives our record tables client-side filtering, column sorting, and pagination functionality. The table components also provide easy and clean connectors to eventually request and consume record data from our backend API.

Upgrades Since Step 3 Draft

- Major grid.js integration for UI tables.
- DML.sql has been significantly cleaned.
- Minor DDL.sql changes (see above feedback and response).

Step 2 Feedback from Peer Reviewers

Reviewer 1

- **Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?**

  Yes, the model follows the database outline and the ERD

- **Is there consistency in a) naming between overview, outline, ER and schema entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?**

  a) Yes, there is consistency amongst the naming conventions between overview, outline, & ER.

  b) Although entities are not plural, I'm sure there was a reason for it since it made it through step 1 review. As for the attributes, they are singular.

  c) Yes, names are capitalized where appropriate.

- **Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)?**

  Schema is easy to read. In fact, the whole document is so well organized that it made everything easy to read!!!

- **Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)?**

  Yes

- **Does the sample data suggest any non-normalized issues, e.g. partial dependencies or transitive dependencies?**

  No

- **Is the SQL file syntactically correct? This can be easily verified by using PhPMyAdmin and your CS 340 database (do not forget to take backup of your own database before you do this!)**

  Yes. Also contains appropriate comments and even a subquery.

- **In the SQL, are the data types appropriate considering the description of the attribute in the database outline?**

Yes

- **In the SQL, are the primary and foreign keys correctly defined when compared to the Schema? Are appropriate CASCADE operations declared?**

  The SQL file and Schema are a match. Cascades are included prior to table creation at the DROP TABLE line.

- **In the SQL, are relationship tables present when compared to the ERD/Schema?**

  Yes

- **In the SQL, is all example data shown in the PDF INSERTED?**

  Yes

- **Is the SQL well structured and commented (e.g. hand authored) or not (e.g. exported from MySQL)?**

  File is well structured (hand authored) with comments and sectioned based on Entity. For ex: Entity movie would have its table created and data inserted below. On the other hand, I had all my tables created first then its values inserted after.

  Overall, was aesthetically pleasing to look at and very organized (esp the first page).

Reviewer 2

Very nice project concept, and excellent formatting for both the PDF and SQL files. I've bolded the important feedback below. Being from Chicago, I especially liked the examples you picked for theater locations!

- Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?
  - **Yes.** The schema exactly matches the database outline and ERD.
- Is there consistency in a) naming between overview, outline, ER and schema entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
  - **Mostly:**
  - a) The naming between the overview, outline, ERD, and schema is consistent for both entities and attributes.
  - b) **Entities are singular instead of plural as expected.** Attributes are singular as expected.

    - c) Capitalization is consistent. All attributes are entirely lowercase, while all words within entity names are capitalized. All words are separated by underscores.
- Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)?
    - **Yes.** The tables are appropriately spaced and no relationship lines are crossed.
- Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)?
    - **Yes.** Both the "Showtime" and "Movie_Genre" intersection tables contain the appropriate foreign keys and enable an M:N relationship (between theaters and movies, and movies and genres, respectively).
- Does the sample data suggest any non-normalized issues, e.g. partial dependencies or transitive dependencies?
    - **Somewhat.** Ticket.payment_method is redundant data that could be prone to inconsistent entry (e.g. "Credit" vs. "Credit card" vs. "Credit Card"). It could also cause several types of anomalies (e.g. no indication that cash is accepted until someone pays cash, deleting all cash Ticket entries obscures the fact that cash is accepted). **I'd recommend creating a separate category table for payment types.** Movie.mpa_rating is similar but probably less problematic because those values come from an authoritative third party and are less likely to change over time. I did not notice any partial or transitive dependencies.
- Is the SQL file syntactically correct? This can be easily verified by using PhPMyAdmin and your CS 340 database (do not forget to take backup of your own database before you do this!
    - **Yes.** The SQL file imports to phpMyAdmin without any errors.
- In the SQL, are the data types appropriate considering the description of the attribute in the database outline?
    - **Yes.** It was an especially good choice to use DATETIME instead of TIMESTAMP for Showtime.showtime_date_time, since the focus in on calendar/clock times instead of the exact Unix time when a showtime occurred.
- In the SQL, are the primary and foreign keys correctly defined when compared to the Schema? Are appropriate CASCADE operations declared?
    - **Mostly.** All primary and foreign keys are correctly defined. All of the DROP TABLE statements use ON DELETE CASCADE. However, I think the intention is that the CREATE TABLE statements should also have ON DELETE behavior defined (what happens if a genre is deleted after table creation?), so **I'd recommend adding that in as well**.
- In the SQL, are relationship tables present when compared to the ERD/Schema?
    - **Yes.** The Showtime and Movie_Genre intersection tables are both present, and contain the attributes and keys described in the ERD and schema.

- In the SQL, is all example data shown in the PDF INSERTED?
    - **Yes**, all example data from the PDF is inserted in the SQL file and imports with the expected values.
- Is the SQL well structured and commented (e.g. hand authored) or not (e.g. exported from MySQL)?
    - **Yes.** The SQL file is well-commented, contains citations where data came from outside sources, and is structured so as to be very readable.

Reviewer 3

Group 25 Project Step 2 Review - Review by James Adelhelm

Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?

- Yes, there is consistency in matching.

Is there consistency in a) naming between overview, outline, ER and schema entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?

- a) Yes, there is consistency between the overview, outline, ER, and schema.
- b) Entities are singular instead of plural. Attributes are singular. RECOMMENDATION: I'd recommend updating to meet requirements.
- c) Capitalization is consistent. Attributes are lowercase and words in the entity names are capitalized. Underscores are used to separate words.

Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)?

- Yes, the schema is easy to read as the tables are appropriately spaced and lines are not crossed.

Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)?

- Yes, the intersection tables are properly formatted. The "Showtime" and "Movie_Genre" intersection tables have the appropriate FKs and there exists a M:N relationship.

Does the sample data suggest any non-normalized issues, e.g. partial dependencies or transitive dependencies?

- No, it does not suggest non-normalized issues. RECOMMENDATION: You could potentially create a category table for types of payment in order to avoid inconsistent entries and redundant data input.

Is the SQL file syntactically correct? This can be easily verified by using PhPMyAdmin and your CS 340 database (do not forget to take backup of your own database before you do this!)

- Yes, the SQL file works.

In the SQL, are the data types appropriate considering the description of the attribute in the database outline?

- Yes, the data types are appropriate. Like Joshua said, good idea to use DATETIME for Showtime.showtime_date_time.

In the SQL, are the primary and foreign keys correctly defined when compared to the Schema? Are appropriate CASCADE operations declared?

- All the PK's and FK's are defined correctly. The appropriate CASCADE operations are declared for the DROP TABLE statements.
- Recommendation: consider adding ON DELETE behavior defined for the CREATE TABLE statements.

In the SQL, are relationship tables present when compared to the ERD/Schema?

- Yes, the Movie_Genre and Showtime intersection tables are both present. They contain the attributes and keys described in both the Schema and ERD.

In the SQL, is all example data shown in the PDF INSERTED?

- Yes, the data is inserted with the expected values.

Is the SQL well structured and commented (e.g. hand authored) or not (e.g. exported from MySQL)?

- Yes, the SQL is hand authored and well commented.

Reviewer 4

Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?

Yes, the ERD and DB Outline match the schema.

Is there consistency in a) naming between overview, outline, ER and schema entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?

a) Consistency between the overview, outline, ER, and schema is present.

b) Entities are singular instead of plural. Attributes are singular.

c) Consistent capitalization. Attributes are lowercase and words in the entity names are capitalized. Underscores are used where appropriate.

Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)?

Yes

Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)?

Yes, intersection table are properly formatted. The "Showtime" and "Movie_Genre" intersection tables have the FKs and a M:N relationship.

Does the sample data suggest any non-normalized issues, e.g. partial dependencies or transitive dependencies?

No, it does not.

Is the SQL file syntactically correct? This can be easily verified by using PhPMyAdmin and your CS 340 database (do not forget to take backup of your own database before you do this!)

Yes, the SQL file works.

In the SQL, are the data types appropriate considering the description of the attribute in the database outline?

Data types are appropriate.

In the SQL, are the primary and foreign keys correctly defined when compared to the Schema? Are appropriate CASCADE operations declared?

All the PK's and FK's are defined correctly. The appropriate CASCADE operations are declared for the DROP TABLE statements.

In the SQL, are relationship tables present when compared to the ERD/Schema?

Yes.

In the SQL, is all example data shown in the PDF INSERTED?

Yes

Is the SQL well structured and commented (e.g. hand authored) or not (e.g. exported from MySQL)?

Yes

Step 2 Summary of Feedback

Schema Feedback

- Multiple peer reviewers recommended updating entity names to plural.

DDL Feedback

- Create a category table for types of payment in order to avoid inconsistent entries and redundant data input (e.g. "Credit" vs. "Credit card" vs. "Credit Card"). This could also cause several types of anomalies (e.g. no indication that cash is accepted until someone pays cash, deleting all cash Ticket entries obscures the fact that cash is accepted).
- Similarly, Movie.mpa_rating may cause issues but less so because those values come from an authoritative third party and are less likely to change over time.
- All of the DROP TABLE statements use ON DELETE CASCADE. However, I think the intention is that the CREATE TABLE statements should also have ON DELETE behavior defined (what happens if a genre is deleted after table creation?), so I'd recommend adding that in as well.
- A peer reviewed Recommendation: consider adding ON DELETE behavior defined for the CREATE TABLE statements.

Fixes based on Feedback from Step 2

- No change to the tense of entity names as we are following the book's format.
- No change to the payment methods in the Ticket table as suggested by a reviewer. Instead this will be handled by the front end as a drop down menu in the "Create New Ticket" form.
- Similarly, no change to Movie.mpa_rating as this will be handled in the UI.
- The DROP TABLE statements are intended to drop the given table if it exists BEFORE creating that table again. The CREATE OR REPLACE TABLE statement does not currently play nice with MySQLWorkbench.
- The CASCADE command has been removed from the DROP TABLE statements.
- ON DELETE CASCADE constraint added to FOREIGN KEY constraint definitions under the Movie_Genre table. The justification is that if the parents are deleted (i.e. Movie or Genre record), we no longer need its association recorded in the Movie_Genre table.
- No delete condition constraints added to the Showtime or Tickets tables. Justification is that these tables should act as indelible records when possible even if a movie, showtime, or customer has since been deleted from other tables. NOTE: It will still be

possible to manually delete or edit records in these tables via our UPDATE and DELETE Queries.

Upgrades since the Step 2 Draft

- Foreign Key constraints assigned aliases for easy manipulation, if needed.
- Added CHECK constraints for both Movie.mpa_rating and Ticket.payment_method as we will likely utilize hard-coded lists for input selection of these values on the frontend (note: no more than 5 possible responses).
- Removed the NOT NULL constraint on the customer_id foreign key attribute in the Ticket table.

Step 1 Feedback from Peer Reviewers

Reviewer 1

I "lol'd" at the "no $12 popcorn here...

Great idea to use for the db.

- Does the overview describe what problem is to be solved by a website with DB back end?
    - Yes. This overview describes the various entities the group will have to keep track of in the database.
- Does the overview list specific facts?
    - Yes, there are specific facts related to the problem statement and what they are trying to solve.
- Are at least four entities described, and does each one represent a single idea to be stored as a list?
    - Yes, this group lists 5.
- Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?
    - Yes, very clearly.
- Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database?
    - Yes to all.
- Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
    - One thing that I might mention here, is you might want to lower case the _ID portion of the column names to save you some typing time and consistency. I know some settings in MYSQL can be case-sensitive so just a thought.

Reviewer 2

- Does the overview describe what problem is to be solved by a website with DB back end?
    - The over view gives a broad view of the organization and it is clear that a DB back end would be helpful in managing such an organizations.
- Does the overview list specific facts?
    - The overview has some specific facts like the cost of popcorn, and the organizations dedication to paying employees a living wage, but it does not list the number of patrons, employees, movie theaters or volume of goods sold. These might be helpful numbers when designing and implementing a data base that fits the needs of IncrediFilms
- Are at least four entities described and does each one represent a single idea to be stored a s a list?
    - There are a total of 6 Entities each with their own single idea to be stored in a link. At first I thought showtimes could be an attribute of movies, but upon further inspection, the current Entity makes a lot of sense.
- Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?

    - Yes, the outline is very thorough in describing the datatypes, relationships and constraints of the entities and attributes
    - The relationship between showtime, auditorium and movie threw me for a bit of a loop at first, but I think the way you have it formatted makes sense. Is it that each showtime will have a FK of the movie and auditorium so that they can be differentiated from the same movie being shown at the same time in a different auditorium? just an opinion, but "showings" might be a better title for that entity
- Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database?
    - Yes, they have correctly formatted 1:M relationships and They have a M:M relationship. I'm a little confused with the ERD as it doesn't list all of the same attributes and FKs as the outline. For instance, the outline of the ticket entity mentions an auditorium FK, but the ERD diagram doesn't show that relationship. There is a footnote explaining some sort of inheritance, but that isn't clear in the diagram. Maybe the tickets should have a M:1 relationship with the auditorium? Either that, or the outline of tickets should only be linked to the showing which includes the auditorium, and the ticket should make no direct reference to auditorium.
- Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
    - a) it looks like the diagram and the outline are consistent in naming
    - b) Entities are all plural, and attributes are singular
    - c) I noticed that under showtimes in the ERD, the movie_id FK is capitalized and that under sales, the customer_id FK is also capitalized. Otherwise good

Nice work! Your team seems to have taken on a complex schema with this project and done a good job of it!. wishing you all the best!

Reviewer 3

The overview is detailed and gives a great understanding of what the website and back end will look like. There are specific details in the overview missing, as the cost of the popcorn seems to be a joke to hook the reader in rather than fact. I think there should be more specificity and hard numbers to get a clearer understanding and picture of the scale of the project.

There are more than four entities, with six entities representing individual ideas to be stored as lists. All of the entities make logical sense.

The outline is well-thought out and describes all the datatypes, relationships, and constraints. I really appreciated the bolding in the outline, as it clearly showed which entities were the focus.

The ERD is put together well and describes the DB similarly to the outline. All of the 1:M and single M:M relationship seem to be in order as well.

Thankfully the diagram and outline are consistent almost all the way throughout, though there were a couple inconsistencies, mainly with how ID is capitalized in the outline but not in the diagram. I think if your team is going to work with snake case for the attributes, then keep ID lowercase (e.g. movie_id).

Great job!

Reviewer 4

- Does the overview describe what problem is to be solved by a website with DB back end?
    - Yes! You did a great job of describing the problem and showing why a website with a DB back end is necessary.
- Does the overview list specific facts?
    - It does! I would say it maybe needs another number somewhere but I thought it was very detailed.
- Are at least four entities described and does each one represent a single idea to be stored a s a list?
    - Yes, the plan describes 6 different entities described as lists.
- Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?
    - The outline of entity details describe the purpose of each and list datatypes and constraints really well. The outline also describes the relationships between entities, each of them having multiple relationships working together.
- Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database?
    - As far as I can see the 1:M relationships are correctly formulated. There is at least 1 M:N relationship, I counted two. The ERD presents a very logical view of the database.
- Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?
    - For the most part, everything looks pretty consistent besides some capitalization issues, specifically between the ID portion on the outline and the diagram.

Great job!!

Reviewer 5

- Does the overview describe what problem is to be solved by a website with DB back end?

The overview describes that IncrediFilms is a movie theater company that seeks to combine a nostalgic atmosphere with modern technology and logistics.

- Does the overview list specific facts?

Yes, the overview lists specific facts such as,

- IncrediFilms is a movie theater company that seeks to combine a nostalgic atmosphere with modern technology and logistics
- The company places emphasis on keeping costs low for consumers and providing fair compensation for employees
- All employees receive a living wage, at minimum.
- IncrediFilms requires a web-enabled front-end and a well-structured and robust database to manage organizational data.
- The website aims to facilitate employee and patron needs and to orchestrate scheduling, stock, and operational needs across all locations and manage organizational data efficiently, organized, and intuitive across locations.
- Are at least four entities described and does each one represent a single idea to be stored as a list?

Yes, at least four entities are described: Movie, Auditorium, Customer, and Showtime. Each entity represents a single idea to be stored as a list, such as movie information, auditorium information, customer information, and showtime information. Each entity has a set of attributes that define the properties of that entity, and relationships are also defined between the entities to show how they are related to each other.

- Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?

Yes, the outline of entity details describes the purpose of each entity, lists attribute datatypes and constraints, and describes relationships between entities. Each entity is described as a single idea to be stored as a list, and includes attributes such as primary keys and foreign keys to indicate relationships between different entities.

- Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database?

The outline of entity details describes the purpose of each, lists attribute datatypes and constraints, and describes relationships between entities. The relationships are correctly formulated as 1:M and there is at least one M:M relationship. The ERD presents a logical view of the database.

- Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?

The entities use the Pascal case. All attributes use the snake case.

Step 1 Summary of Feedback

Overview Feedback

- The overview “broadly” describes what problem is to be solved.
- There is a lack of metrics and specificity provided so the reviewer is unable to get a clear picture about the scale of our project. A suggestion from the reviewer is to add hard numbers to the outline.

Database Outline Feedback

- Reviewers agree that the outline describes the purpose, listed attribute data types and constraints, and describes relationships between entities. There is at least 1 M:N relationship. There is some confusion about the relationship between Movie, Showtime, and Auditorium (now Theater). A reviewer suggests a rename of “Showtime” to “Showings” for clarity.
- There is confusion about the relationship between the Ticket and the Auditorium (now Theater).
- There are some differences in the naming conventions, where some entities are plural but the attributes are singular.
- Some attributes are inconsistent with the capitalization. Reviewers suggest to lowercase the “_ID” portion of the attributes to save some typing time and consistency due to MySQL being case-sensitive.

ERD Feedback

- There are some inconsistencies in the entities’ that are present outline that no longer exist in the ERD and vice versa.
- The inheritance footnote does not clearly read in the ERD.
- Some attributes are inconsistent with the capitalization. Reviewers suggest to lowercase the “_ID” portion of the attributes to save some typing time and consistency due to MySQL being case-sensitive.

Fixes based on Feedback from Step 1

Overview Actions

- Metrics about IncrediFilms' movie-goer attendance and number of locations were added, showing an increase in both the number of theaters IncrediFilms operates and the growth in attendance in the same five-year period. This addition narrows the scope of our project and provides a clearer image to the reviewer as to why IncrediFilms needs to implement a database backend to manage their organizational data.

Database Outline Actions

- Separate orders and tickets/seats entities were collapsed into directly associating tickets with customers.
- The relationship between Movie and Showtime is 1:M, where one Movie can have multiple Showtime(s) at the theater. No changes were made to this relationship.
- The relationship between Auditorium (now Theater) and Showtime is updated to 1:M, where a theater can contain multiple showtimes, but each showtime can appear in at most one theater.
- No update was made to the name of the entity 'Showtime.' We felt the original is consistent with the naming convention of our competitors' apps.
- Updates to the overview to match the intended relationships shown in the ERD were made: namely, removing inactive attributes from earlier workings of the draft.
- Attribute names in the database outlines have been adjusted to all lower-case. This implementation was made to save time and for consistency going forward.
- Entity/table names updated to singular, capitalized names, following examples provided in the course textbook.
- Earlier footnote describing how the movie and/or auditorium associated with a ticket will be found has been removed. This will be handled by SQL queries and is beyond the scope of the current assignment.

ERD Actions

- Dropped an unnecessary entity (Sale).
- Added a new 'Genre' entity.
- Attribute names in the ERD tables have been adjusted to all lower-case. This implementation was made to save time and for consistency going forward.
- Updates to the ERD to match the intended relationships outlined were made: namely, adding FKs to match the relationships mentioned or removing attributes no longer included in the outline.

Upgrades to the Step 1 Draft Version

- Finalized Ticket/Seats entity name to ‘Ticket’.
- Updated the relationship between Auditorium (now Theater) and Showtime to 1:M, where one Theater can have zero or more Showtime, but only only Showtime per Theater.
- Added an entity, Genre, which will now hold a M:N relationship with Movie.
- Condensed Ticket/Seat and Order entities to one entity, Ticket.
- Added price and payment_method attribute to Ticket to account for loss of Sale entity.
- Updated the relationship between Customer and Ticket to 1:M instead of the previous 1:M Customer had with Sale.
- Renamed Auditorium entity to Theater and renamed attribute “auditorium_name” to “theater_name”.
- Added attribute “movie_year” to Movie entity.
- Removed not NULL constraint on the email attribute in the Customer table.

Upgrades since the Step 1 Final Version

- Overview description update to better describe IncrediFilm’s objects for customer and movie details data keeping.
- Reorganized the outline layout to coincide with the DDL.
- Updated price data type from decimal(3,2) to decimal(5,2) to account for four- and five- digit prices.
- Removed UNIQUE constraint from the email attribute in the Customer table**.**
- Added NOT NULL constraint to payment_method attribute in the Ticket table.
- Added NOT NULL constraint to theater_name attribute in the Theater table.
- Added NOT NULL constraint to foreign keys in the Showtime table.
- Added NOT NULL constraint to foreign keys in the Ticket table.
- Removed width (e.g. int(4)) from multiple int attributes. MySQL is depreciating the need for small int widths.
- Edited the ERD diagram to also include Movie_Genre intersection table, matching Schema diagram.
- Added language in the Database outline entities to clarify the foreign key relationships.

Project Proposal

Overview

IncrediFilms™ is a burgeoning player in the world of movie-goer experience. IncrediFilms theaters seek to celebrate a fusion between the nostalgic atmosphere of the red-velvet curtain theater and the web native systems that today's audience expects. Each IncrediFilms theater remains community-owned and operated, while utilizing a modern logistics backend—orchestrating scheduling, stock, and operational needs across their 12 locations. IncrediFilms prides itself in keeping costs to consumers low—no $12 popcorn, here—and employee compensation fair. The organization is proud to state that all employees receive a living wage, at minimum.

In order to accomplish each of their defining qualities, the organization relies on a modern tech backend that keeps operations efficient, organized, and intuitive across all locations. The number of IncrediFilms cinema screens statewide increased 20% between 2019 and 2023 to 12, with overall attendance growing more than 40% in the same time frame to a record 4 million unique movie-goers and counting (as reported in their earnings report.) With plans to open another two locations in the next two years, they can no longer rely on their fleet of Etch-A-Sketches alone to handle record keeping. To this end, IncrediFilms requires a web-enabled frontend that facilitates employee and patron needs. Underlying this frontend and other interface systems, a well-structured and robust database is crucial for managing organizational data.

Phase I of this database implementation is to streamline Movie scheduling amongst IncrediFilms' flagship Theaters and track financial performance of each Movie and Movie Genre via Tickets sold to Customers to decide Showtimes for the following week. This system must be capable of recording sales for 5,000 Showtimes in its 12 Theaters annually. Additionally, in acting as a record of IncrediFilm's screening history, the database should track basic movie details such as year, runtime, and its genre categorizations. These details will help to inform the online information system and ticket-buying experience for customers. Finally, IncrediFilms seeks to manage its loyal customer fanbase. The system will also need to keep track of IncrediFilms customer records, supplying contact information and linking customers with their ticket purchases. It is hoped that the data gleaned from these records will inform IncrediFilms decisions and strategy in the future, catering the experience to the organization's most prolific customers.

Database Outline in Words

1. **Entity 1: Customer** - records and stores information about each customer and/or potential customer
    - customer_id int NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY
    - first_name varchar(30) NOT NULL
    - last_name varchar(30) NOT NULL
    - dob date NOT NULL
    - email varchar(254)

    Relationship(s):
        - **1:M** relationship between **Customer** and **Ticket**; a Customer can be associated with zero or more Tickets, whereas a Ticket will be associated with one and only one Customer.

2. **Entity 2: Genre** - records and stores information about possible genres associated with movies.
    - genre_id int NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY
    - genre_name varchar(45) NOT NULL UNIQUE

    Relationship(s):
        - **M:N** relationship between **Genre** and **Movie**. Each Genre can have zero or more Movies associated with it; each Movie will have one or more Genres associated with it.
            - An intersection table, **Movie_Genre**, will facilitate this **M:N** relationship between Movie and Genre.

3. **Entity 3: Movie -** records and stores information of each movie IncrediFilms shows in their theaters.
    - movie_id - int NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY
    - movie_name varchar(100) NOT NULL
    - runtime_min int NOT NULL
    - mpa_rating enum NOT NULL
    - movie_year year NOT NULL

    Relationship(s):
        - **1:M** relationship between **Movie** and **Showtime**; each Movie can have zero or more Showtimes associated with it, each Showtime will be associated with one and only one movie.
        - **M:N** relationship between **Movie** and **Genre**. Each Movie will have one or more Genres associated with it; each Genre can have zero to many associated Movies.
            - An intersection table, **Movie_Genre**, will facilitate this **M:N** relationship between Movie and Genre.

4. **Entity 4: Showtime** - records and stores information about each showtime.
    - showtime_id int NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY
    - show_date_time datetime NOT NULL
    - movie_id int, FK
    - theater_id int, FK

    Relationship(s):
        - **1:M** relationship between **Showtime** and **Ticket**; a Showtime will be associated with zero or more Tickets, while a Ticket will be associated with one and only one Showtime.
        - **1:M** relationship between **Movie** and **Showtime**; a given Showtime will be associated with one and only one Movie; a given Movie can have zero or more Showtimes. movie_id is the foreign key in the Showtime table that facilitates the relationship between the two entities.
        - **1:M** relationship between **Showtime** and **Theater**; a given Showtime will take place in one and only one Theater; a Theater can have zero to many Showtimes. theater_id is the foreign key in the Showtime table that facilitates the relationship between the two entities.

5. **Entity 5: Theater** - records and stores information about each IncrediFilms theater location.
    - theater_id int NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY
    - theater_name varchar(50) NOT NULL UNIQUE
    - no_of_seats int NOT NULL

    Relationship(s):
        - **1:M** relationship between **Theater** and **Showtime**; a Theater can have zero to many Showtime instances. A given showtime will have one and only one associated Theater.

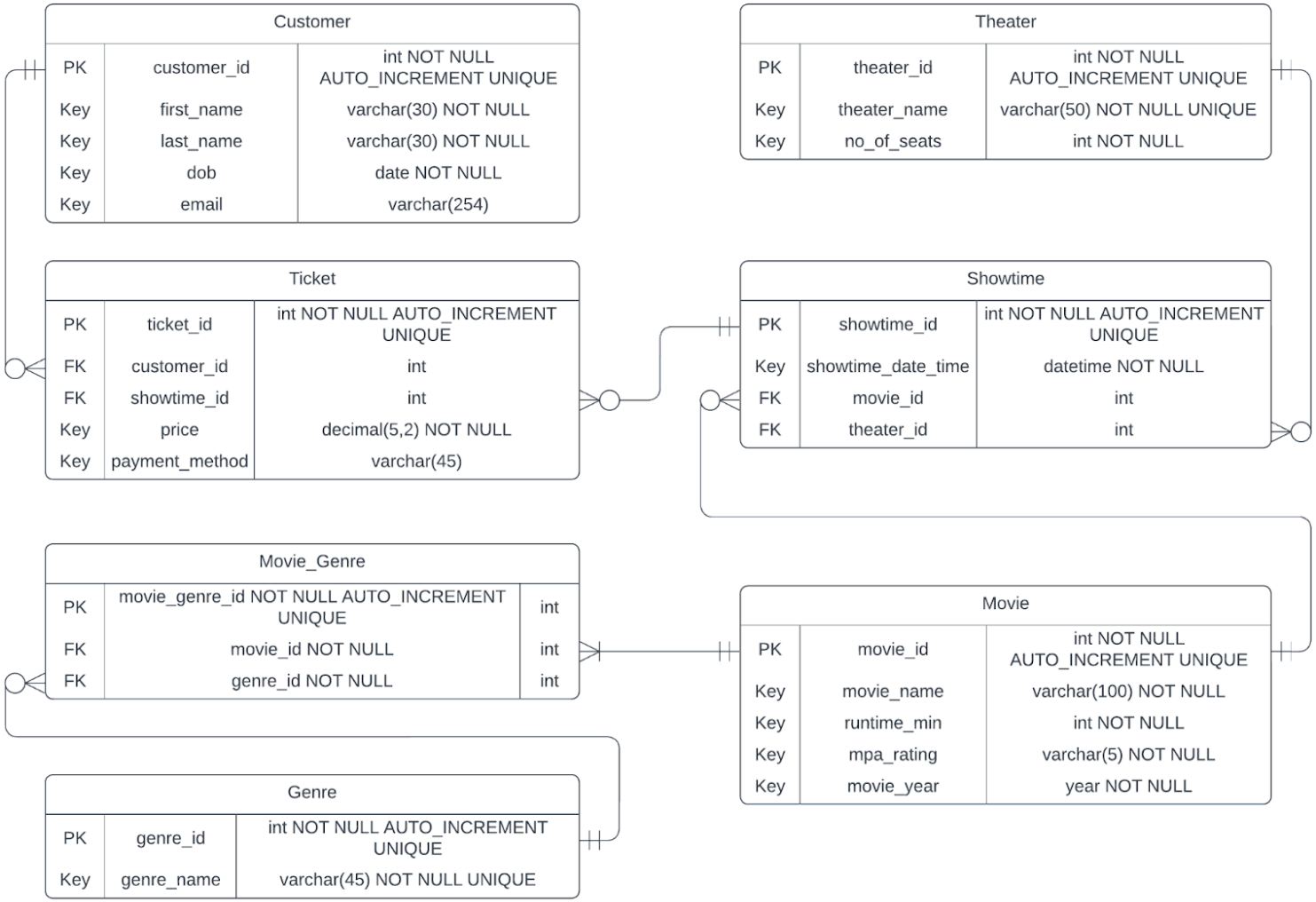
6. **Entity 6: Ticket** - records and stores the details of each movie ticket.
    - ticket_id int NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY
    - customer_id int, FK
    - showtime_id int, FK
    - price decimal(5,2) NOT NULL
    - payment_method varchar(45)

    Relationship(s):
        - **1:M** relationship between **Customer** and **Ticket**; a given Customer can have zero or more Tickets, while a given Ticket can have zero or one customer. customer_id is the foreign key in the Ticket table that facilitates the relationship between the two entities.
        - **1:M** relationship between **Showtime** and **Ticket**; a given Ticket will have one and exactly one Showtime (on creation), while a given Showtime can be associated with zero to many Tickets. showtime_id is the foreign key in the Ticket table that facilitates the relationship between the two entities.

7. **Intersection Table: Movie_Genre** - Facilitates the M:N relationship between Movie and Genre entities.
    - movie_genre_id int NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY
    - movie_id int NOT NULL, FK
    - genre_id int NOT NULL, FK

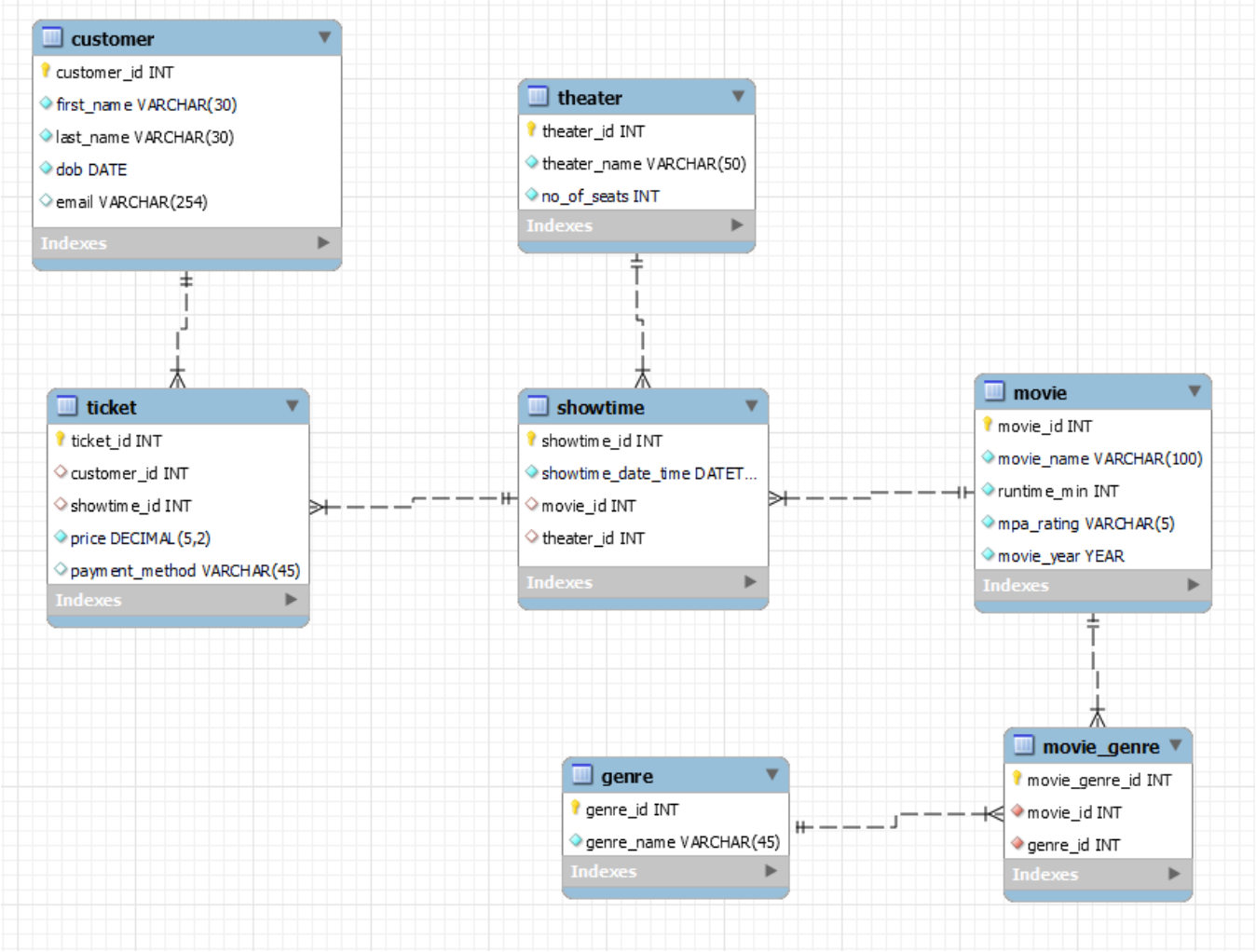
      Relationship(s):
        - Facilitates **M:N** relationship between **Movie** and **Genre**. Each Movie will have one or more Genres associated with it; each Genre can have zero to many associated Movies.
        - **1:M** relationship between Genre and Movie_Genre implemented with genre_id as the foreign key in Movie_Genre.
        - **1:M** relationship between Movie and Movie_Genre implemented with movie_id as the foreign key in Movie_Genre.

Entity-Relationship Diagram (ERD)



Project Implementation

Schema Diagram



Example Data

Customer Entity's Sample Data[1]

customer_id	first_name	last_name	dob	email
1	AJ	Styles	1977-06-02	AJ.Styles@bmail.com
2	Stephanie	Helmsley	1976-09-24	Stephanie.Helmsely@bmail.com
3	Alexa	Bliss	1991-08-09	Alexa.Bliss@bmail.com
4	Booker	T	1965-03-01	*NULL*
5	Jenna	Andrade	1989-11-03	Jenna.Andrade@bmail.com
6	Andre	Giant	1946-05-19	*NULL*
7	Michaela	Hargrove	2001-05-30	Michaela.Hargrove@bmail.com
8	Em	Patterson	2017-12-02	*NULL*

Movie Entity's Sample Data

movie_id	movie_name	runtime_min	mpa_rating	movie_year
1	Dr. Strangelove or: How I Learned to Stop Worrying...	95	PG	1964
2	Interstellar	169	PG-13	2014
3	Amélie	122	R	2001
4	The Shining	146	R	1980
5	Everything Everywhere All at Once	139	R	2022
6	Encanto	102	PG	2021
7	Bee Movie	91	PG	2007

[1] Data for Customer Table is a blend of invented data and adapted data from mdabbert (2020). See Works Cited section for more details.

Genre Entity's Sample Data

genre_id	genre_name
1	Documentary
2	Kids
3	Family
4	Comedy
5	Independent
6	International
7	Drama
8	Musical
9	Thriller
10	Horror
11	Sci-Fi
12	Romance
13	Animated
14	Sports
15	Action
16	Cult Classic
17	Adventure
18	LGBTQ+
19	Crime
20	Mystery
21	Fantasy
22	Historical

Theater Entity's Sample Data

theater_id	theater_name	no_of_seats
1	IncrediFilms Rogers Park	300
2	IncrediFilms Wicker Park	500
3	IncrediFilms Uptown	300
4	IncrediFilms Lincoln Square	250
5	IncrediFilms North Center	250
6	IncrediFilms Lake View	250

Ticket Entity's Sample Data

ticket_id	customer_id	showtime_id	price	payment_method
1	2	5	9.00	CREDIT
2	2	6	5.00	CREDIT
3	8	1	9.00	CASH
4	*NULL*	1	9.00	DEBIT
5	3	2	9.00	CREDIT
6	4	4	9.00	*NULL*

Showtime Entity's Sample Data

showtime_id	showtime_date_time	movie_id	theater_id
1	2023-02-10 16:00:00	1	1
2	2023-02-10 15:00:00	2	4
3	2023-02-14 17:00:00	2	4
4	2023-02-14 18:00:00	7	2
5	2023-02-14 18:00:00	7	1
6	2023-02-16 12:00:00	3	5
7	2023-02-16 15:30:00	4	1

Movie_Genre's Sample Data (Intersection Table)

movie_genre_id	movie_id	genre_id
1	1	4
2	1	15
3	2	11
4	2	15
5	2	17
6	2	9
7	3	12
8	3	4
9	4	10
10	4	20
11	5	11
12	5	4
13	5	20
14	6	2
15	6	3
16	6	8
17	6	17
18	7	3
19	7	4
20	7	15
21	7	17
22	7	16

IncrediFilms Database Manager UI

http://flip1.engr.oregonstate.edu:40593/

Works Cited

mdabbert. “Professional Wrestling Champions (WWE/WWF).” 2020, https://www.kaggle.com/datasets/mdabbert/professional-wrestling-champions-wwewwf. Retrieved on 2/5/23 via the web.