Bases de Données
Cinémas genevois

Rebeka Mali, Valon Halili, Ayman Chidda,
Loris Thomas, Louis Gérard

Printemps 2024

Contents

1 Description générale

Le projet est une base de données des cinémas genevois. Elle contient :

- les cinémas, leurs emplacements
- les salles des cinémas, leurs équipements spéciaux (type IMAX)
- les séances qui ont lieux dans ces salles, avec leurs horaires
- les films qui y sont diffusés, leurs genres, réalisateurs...

La base de données alimente une application dédiée aux cinéphiles genevois qui souhaitent effectuer des recherches liés à ces données. Il leur est possible de trier les films par genre, les cinémas par emplacement, les salles par équipement spécifique pour trouver la séance qui leur convient le mieux.
La base de données que nous avons créée utilise la technologie MySQL. Pour donner une meilleure idée d'une utilisation possible de cette base, nous rendons également une maquette d'application Web qui utilise le framework Streamlit. Elle est disponible à cette adresse et son code source est également inclus dans le rendu du projet (voir README.md pour la build). La base de données est déployée sur le service gratuit Clever Cloud pour pouvoir être utilisée par l'application Streamlit.

Les données de la base sont un échantillon représentatif des sorties de films de ces derniers mois. Dans le cas d'un déploiement pratique de cette application, il faudrait qu'elle contienne idéalement autant de films que possible mais ceux qui sont présents actuellement suffisent amplement à présenter les fonctionnalités de la base.

2 Description des relations

Notre base de données comporte 6 relations en total. Nous allons les décrire et justifier une par une:

2.1 Lieu

Cette relation comporte 6 attributs avec la clef "lieu_id". Elle fournit à l'utilisateur les informations relatives à l'accès au cinéma : accéssibilité pour personnes à mobilité réduite, adresse, horaires ...

2.2 Salle

Cette relation dispose de 9 attributs avec la clef "salle_id". Elle est liée à la relation 'lieu' grâce à la clef étrangère 'cinema_id'. Cette clef permet de situer chaque salle dans un cinéma précis. Cette relation a plusieurs usages : elle offre des informations sur la capacité de la salle, la technologie utilisée (IMAX, ScreenX) mais aussi sur le prix des billets.

2.3 Séance

Cette relation comporte 6 attributs avec comme clef "seance_id". Il s'agit d'une relation qui fournit des informations indispensables concernant la projection d'un film : elle indique le film et la salle (à travers les clefs étrangères "salle_id" et "film_id") et d'ailleurs elle affiche la date et l'heure de la séance, ainsi que la langue dans laquelle le film est projeté.

2.4 Film

Cette relation contient 9 attributs avec la clef "film_id". Elle fournit à l'utilisateur les films diffusés. A part le titre du film, les informations notoires sont la durée, la date de sortie, le pays de production, l'âge minimal et le réalisateur ainsi qu'un ou deux acteurs. Toutes ces informations sont présentes pour permettre à l'utilisateur de filtrer selon des critères précis.

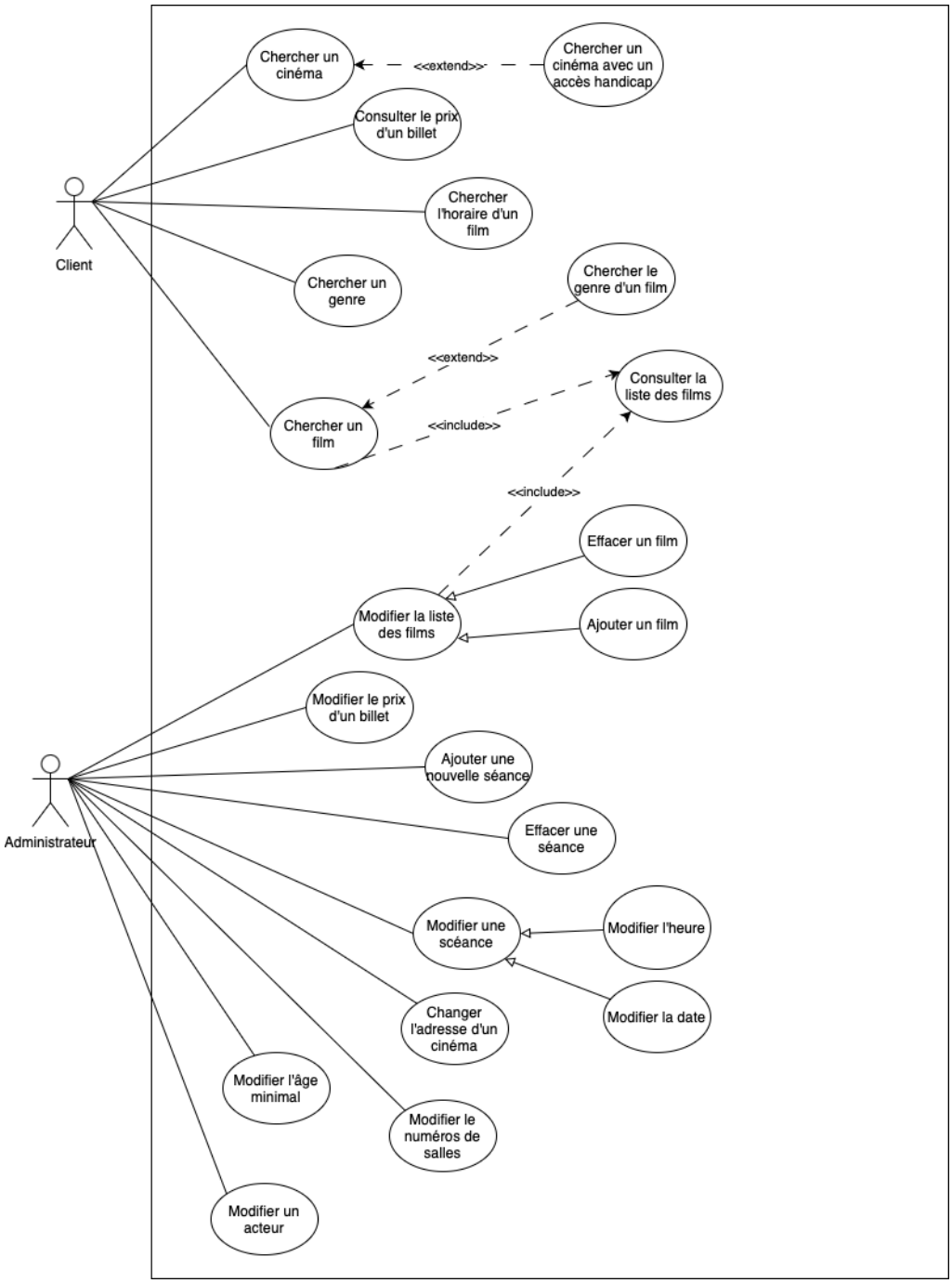
2.5 Genre

La relation genre est simple et ne comporte que deux attributs donc une est la clef "genre_id". Cette relation existe car certains films peuvent avoir plusieurs genres et d'autres un seul ou aucun. Pour éviter d'avoir un nombre élevé d'attributs dans Film (Genre1 à Genre5) dont la plupart seraient NULL, nous avons choisi de créer une relation entière avec des "id" de genre et des strings associés (pour la validation de données et éviter d'avoir 13 genres "comédie" avec des orthographes différentes).

2.6 Film_genre

La relation film_genre unit les relations Film et Genre pour attribuer un nombre indéterminé lors de la création de la BDD de genres à chaque film.

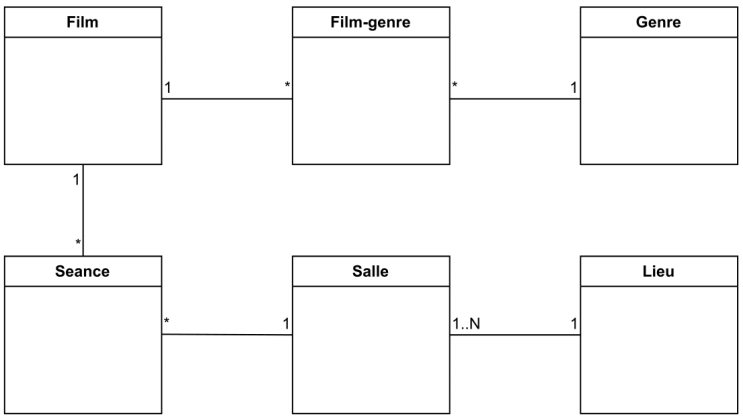
3 Diagramme des cas d'utilisations



Dans les cas d'utilisation, nous avons choisi deux acteurs principaux : le client et l'administrateur. Pour ce qui concerne le client, nous avons envisagé comme cas d'utilisation : consulter le prix d'un billet, chercher l'horaire d'un film, chercher un genre particulier (par exemple s'il veut lancer une recherche de tous les films d'horreur qui sont disponibles). D'autres cas d'utilisation sont la recherche d'un cinéma et la recherche d'un film : dans le premier cas, le client pourrait choisir de chercher un cinéma qui a un accès handicap (le fait d'être optionnel est marqué par la relation «extend») ; dans le deuxième cas, lorsqu'il cherche un film, le client a l'option de chercher aussi le genre d'un film s'il désire. Néanmoins, afin de chercher un film, il doit d'abord consulter la liste des films.

Pour ce qui concerne l'administrateur du système, il y a plusieurs cas d'utilisation : modifier le prix d'un billet, ajouter une nouvelle séance, effacer une séance, changer l'adresse d'un cinéma (si par exemple le cinéma est déplacé à une nouvelle adresse), modifier le numéro de salles disponibles (ceci peut être dû à plusieurs raisons comme l'ajout d'une nouvelle salle, son entretien ou dans le cas d'une panne), modifier l'âge minimal pour un film (par exemple suite à des plaintes de la part des clients), et modifier un acteur (cela pourrait arriver dans le cas où un acteur décide de changer son nom d'artiste, s'il/elle se marie, etc.). Certains cas d'utilisation méritent un peu plus de précisions. Par exemple, modifier la liste des films engendre un choix de la part de l'administrateur : il peut choisir entre effacer un film ou en ajouter. Cependant, il ne peut modifier la liste sans avoir consulté la liste des films, d'où la relation «include». Lorsqu'on veut modifier une séance, le programme doit nous demander si on veut modifier l'heure ou la date. Ceci est très pratique : nous pourrions imaginer que lorsqu'un nouveau film sort, nous aimerions le projeter chaque mardi et jeudi pendant 4 mois. Cependant, nous pourrions avoir oublié qu'un mardi est un jour férié et donc soit nous avons des horaires réduits soit le cinéma est fermé. Cette option nous permet de changer facilement l'horaire et/ou la date.

4 Diagramme des classes



Les détails de chaque classe sont inclus dans la section suivante.

5 Liste des attributs

Les clefs de chaque classe sont en bleu. On utilise le domaine "bool" qui n'existe pas vraiment sur certaines technologies SQL, qui sera alors remplacé par un tinyint(1).

Lieu

Attribut	Domaine	Synopsis
lieu_id	big int	Identifiant du cinéma
nom	varchar(50)	Nom du cinéma
adresse	varchar(50)	Adresse du cinéma
salle_nb	int	Nombre de salles du cinéma
NPA	int	Le code postal du cinéma [1ex]
acces_handicap	bool	Le cinéma a-t-il un accès mobilité réduite ?

Salle

Attribut	Domaine	Synopsis
salle_id	int	Identifiant de la salle
salle_nom	varchar(50)	Nom de la salle
cinema_id	int	Identifiant du cinéma
IMAX	bool	La salle est-elle équipée IMAX ?
screenX	bool	La salle est-ele équipée ScreenX ?
capacité	int	Capacité de la salle en personnes
prix_plein	int	Prix tarif régulier
prix_reduit	int	Prix tarif étudiant, senior...
prix_enfant	int	Prix tarif enfant

Séance

Attribut	Domaine	Synopsis
seance_id	int	Identifiant de la séance
cinema_id	int	Identifiant du cinéma
salle_id	int	Identifiant de la salle
film_id	int	Identifiant du film
date_seance	date	Date de la séance
heure_seance	heure	Heure de la séance
vo	bool	La séance est-elle en Version Originale ?

Film

Attribut	Domaine	Synopsis
film_id	int	Identifiant du film
titre	varchar(50)	Titre du film
duree	time	Durée du film
date_sortie	date	Date de sortie du film
pays	varchar(50)	Pays du film
acteur1	varchar(50)	Acteur principal du film
acteur2	varchar(50)	Acteur secondaire du film
réalisteur	varchar(50)	Réalisateur du film
age_minmimal	int	Age minimal pour voir le film

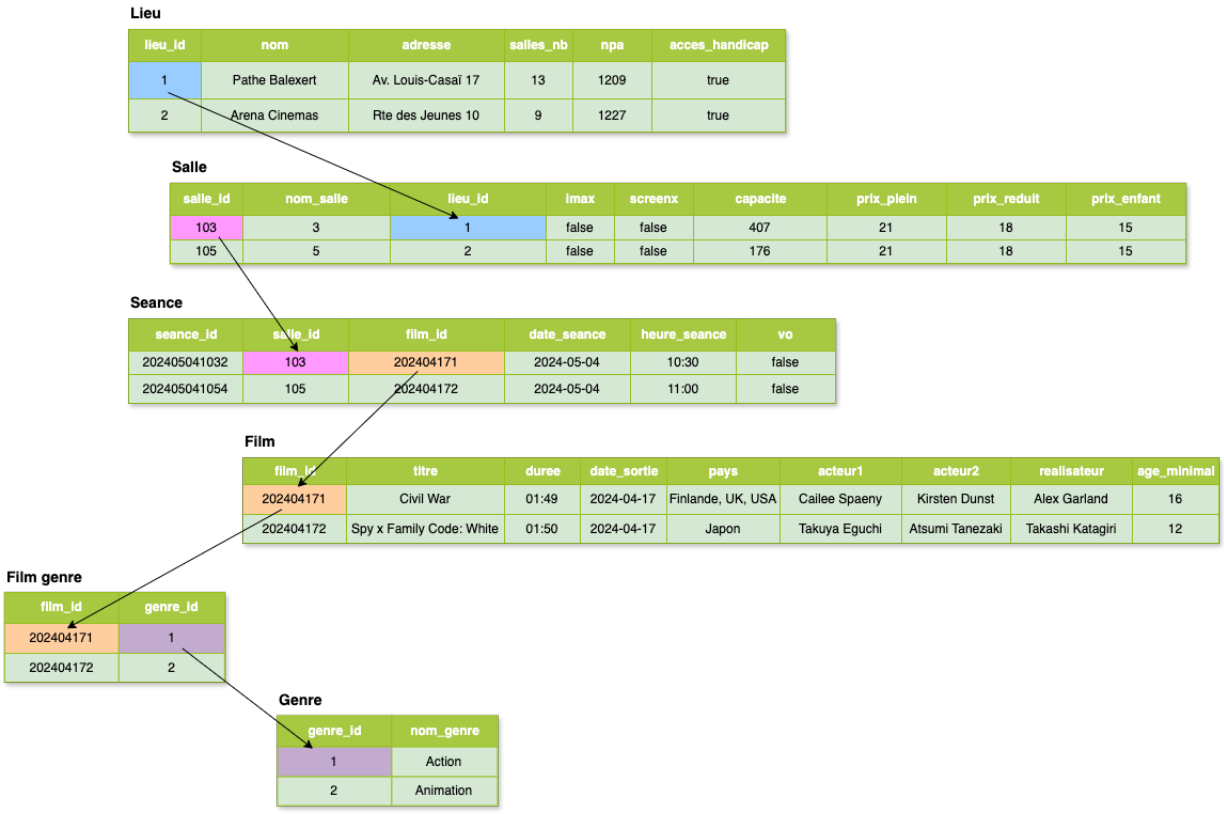
Genre

Attribut	Domaine	Synopsis
genre_id	int	Identifiant du genre
nom_genre	varchar(50)	Nom textuel du genre (utile pour streamlit)

Film_Genre

Attribut	Domaine	Synopsis
film_id	int	Identifiant du film
genre_id	int	Identifiant du genre

6 Schémas des relations



lieu_id

- lieu_id dans la table Lieu est appelé clé primaire.
- lieu_id dans la table Salle est appelé clé étrangère.

salle_id

- salle_id dans la table Salle est appelé clé primaire.
- salle_id dans la table Seance est appelé clé étrangère.

film_id

- **film_id** dans la table Film est appelé clé primaire.
- **film_id** dans la table Film_Genre est appelé clé primaire.
- **film_id** dans la table Seance est appelé clé étrangère.

genre_id

- **genre_id** dans la table Genre est appelé clé primaire.
- **genre_id** dans la table Film_Genre est appelé clé étrangère.

7 Justifications 3FN

Afin de démontrer que les relations sont en 3FN, nous avons besoin de démontrer qu'elles sont d'abord en 1FN, 2FN, et ensuite en 3FN.

7.1 Lieu

Lieu(lieu_id // nom, adresse, salles_nb, npa, acces_handicap)

- 1FN : La relation est en première forme normale car elle possède une clef(lieu_id) et ses attributs sont atomiques ('adresse' dans ce cas est un 'string'). Nous avons choisi d'avoir comme cleflieu_id car le nom d'un cinéma pourrait changer au cours des années, mais le cinéma garderait la même adresse, les numéros de salles, etc. Donc c'est plus simple d'assigner un id qui reste inchangé. D'ailleurs, cela facilite aussi les requêtes SQL.
- 2FN : Vu que la relation est en 1FN et que nous n'avons qu'un seul attribut clef, la relation est aussi en deuxième forme normale.
- 3FN : La relation est en 2FN et tous les attributs dépendent uniquement de la clef. (Le NPA ne dépend pas de l'adresse puisqu'il existe des doublons d'adresses justement différenciés par le NPA en Suisse).

7.2 Salle

Salle(salle_id // nom_salle, lieu_id, imax, screenx, capacite, prix_plein, prix_reduit, prix_enfant)

- 1FN : La relation est en première forme normale car elle possède une clef(salle_id) et ses attributs sont atomiques. Nous avons choisi d'avoir comme clefsalle_id car deux cinémas pourraient avoir le même nom pour une salle : par exemple, nous pourrions avoir la salle "A" au Nord-Sud et l'autre à Les Scala. Avec un id nous sommes sûrs d'avoir la salle d'un cinéma précis.
- 2FN : Vu que la relation est en 1FN et que nous n'avons qu'un seul attribut clef, la relation est aussi en deuxième forme normale.

- 3FN : la relation est en 2FN et tous les attributs dépendent de la clef. En effet, toutes les technologies (imax, screenx) dépendent de salle_id et non pas du lieu_id car un même cinéma peut avoir diverses salles équipées différemment. Et, comme mentionné avant, les attributs ne peuvent pas dépendre de "nom _salle", car si nous avons deux mêmes noms de salle dans deux cinémas différents, on risque d'avoir des valeurs d'attributs qui ne correspondent pas à la réalité.

7.3 Séance

Seance(seance_id// salle_id, film_id, date_seance, heure_seance, vo)

- 1FN : La relation est en première forme normale car elle possède une clef(seance_id) et ses attributs sont atomiques. Nous aurions pu utiliser comme clef(salle_id film_id date_seance heure_seance//) mais, si nous voulions utiliser la clefde la relation comme clefétrangère dans une autre relation, cela nous obligerait de répéter toutes les valeurs des attributs au lieu d'une simple clefcomme seance_id. Le choix d'une clefsimplifie aussi les requêtes SQL.

- 2FN : Vu que la relation est en 1FN et que nous n'avons qu'un seul attribut clef, la relation est aussi en deuxième forme normale.

- 3FN : la relation est en 2FN et tous les attributs dépendent de la clef. Un même film_id peut être transmis dans différentes salles à différentes dates et en différentes langues. Ce ne sera que le seance_id qui pourra indiquer la juste combinaison.

7.4 Film

Film(film_id // titre, duree, date_sortie, pays, acteur1, acteur2, realisateur, age_minimal)

- 1FN : La relation est en première forme normale car elle possède une clef(film_id) et ses attributs sont atomiques (réalisateur dans ce cas est un 'string'). Nous aurions pu utiliser comme clef(titre date_sortie//) mais, si nous voulions utiliser la clefde la relation Film comme clefétrangère dans une autre relation, cela nous obligerait de répéter les valeurs de ces attributs dans une autre relation au lieu d'une clefsimple comme film_id. Ce choix nous facilite les requêtes SQL et évite aussi des fautes des frappe qui pourraient arriver quand on saisit le nom du film.

- 2FN : Vu que la relation est en 1FN et que nous n'avons qu'un seul attribut clef, la relation est aussi en deuxième forme normale.

- 3FN : La relation est en 2FN et tous les attributs dépendent de la clef. En effet, l'attribut 'date_sortie' ne peut dépendre de l'attribut 'titre' car nous pourrions avoir deux films avec le même titre sortis à deux dates différentes. Cela pourrait arriver si le cinéma décide de projeter l'ancien film avec le nouveau qui vient de sortir avec le même titre : par exemple, le film 'Rebecca' sorti en 1940 et en 2020.

7.5 Genre

Genre(genre_id//nom_genre)

- 1FN : La relation est en première forme normale car elle possède une clef(genre_id) et ses attributs sont atomiques ('nom_genre' est un string).

- 2FN : Vu que la relation est en 1FN et que nous n'avons qu'un seul attribut clef, la relation est aussi en deuxième forme normale.

- 3FN : La relation est en 2FN et le seul attribut de la relation dépend de la clef.

7.6 Film_genre

Film_genre(FilmID GenreID //)

- 1FN : La relation est en première forme normale car elle possède une clef('Film ID GenreID'); nous n'avons pas dans ce cas d'autres attributs.
- 2FN : Vu que la relation est en 1FN et que nous n'avons qu'un seul attribut clef, la relation est aussi en deuxième forme normale.
- 3FN : La relation est en 2FN et le seul attribut de la relation dépend de la clef.

8 Requêtes SQL

Voici des exemples de requêtes SQL plus ou moins générales qui donnent une idée de ce dont la base de données est capable. Ces requêtes peuvent être effectuées automatiquement sur le site de l'application à cette adresse.

1. Les cinémas qui diffusent les films italiens pour lesquels l'âge minimal est de 12 ans

```
SELECT DISTINCT l.nom
FROM lieu l
JOIN salle s ON s.lieu_id = l.lieu_id
JOIN seance s2 ON s2.salle_id = s.salle_id
JOIN film f ON f.film_id = s2.film_id
WHERE f.pays LIKE '%Italie%' AND f.age_minimal = 12
```

2. Les réalisateurs qui ont joué dans leur propre film, ainsi que le nom du dit film

```
SELECT realisateur, titre
FROM film f
WHERE FIND_IN_SET(f.acteur1, f.realisateur) > 0
OR FIND_IN_SET(f.acteur2, f.realisateur) > 0
```

On utilise FIND IN SET car les films ont parfois plusieurs réalisateurs, c'est donc une liste qui se trouve dans la colonne réalisateur.

3. Les films et leurs genres avec une date de sortie en janvier 2024, qui sont encore diffusés entre le 13 et 19 mai

```
SELECT f.titre, GROUP_CONCAT(DISTINCT g.nom_genre SEPARATOR ', ') AS genres
FROM seance s
JOIN film f ON f.film_id = s.film_id
LEFT JOIN film_genre fg ON fg.film_id = f.film_id
LEFT JOIN genre g ON g.genre_id = fg.genre_id
WHERE f.date_sortie BETWEEN '2024-01-01' AND '2024-01-31'
AND s.date_seance BETWEEN '2024-05-13' AND '2024-05-19'
GROUP BY f.titre
```

4. Les films d'animation sortis en l'an 2000

```
SELECT f.titre
FROM film f
JOIN film_genre fg ON fg.film_id = f.film_id
JOIN genre g ON g.genre_id = fg.genre_id
WHERE g.nom_genre = 'Dessin animé' AND f.date_sortie BETWEEN
'2000-01-01' AND '2000-12-31'
```

5. Le nombre moyen de séances d'un film sorti le 8 mai 2024 entre sa sortie et le 14 mai, par cinéma

```
WITH seance_nb AS(
SELECT f.titre, s2.lieu_id, COUNT(*) AS Nb
FROM seance s
JOIN film f ON f.film_id = s.film_id
JOIN salle s2 ON s2.salle_id = s.salle_id
WHERE f.date_sortie = '2024-05-08' AND s.date_seance BETWEEN
'2024-05-08' AND '2024-05-14'
GROUP BY s.film_id, s2.lieu_id
)
SELECT l.nom, AVG(Nb) AS nombre_moyen_de_seances_pour_un_nouveau_film
FROM seance_nb
JOIN lieu l ON l.lieu_id = seance_nb.lieu_id
GROUP BY seance_nb.lieu_id
```

6. Tous les films suisses sortis au 21ème siècle

```
SELECT titre
FROM film f
WHERE date_sortie >= 2001-01-01 AND pays LIKE '%Suisse%'
```

7. Les acteurs jouant dans plusieurs films. Trier par nombre de films joués.

```
SELECT acteur, COUNT(*) as nb_film
FROM (
SELECT f.acteur1 AS acteur
FROM film f
UNION ALL
SELECT f.acteur2 AS acteur
FROM film f
) AS acteurs
WHERE acteur IS NOT NULL
GROUP BY acteur
HAVING nb_film > 1;
```

8. Les séances (Titre, heure, lieu, salle et prix) du 17 mai à partir de 17h en VO avec Drame comme l'un des genres du film

```
SELECT f.titre, s.heure_seance, l.nom, s2.nom_salle, s2.prix_plein,
s2.prix_reduit, s2.prix_enfant
FROM seance s
JOIN film f ON f.film_id = s.film_id
JOIN film_genre fg ON fg.film_id = f.film_id
JOIN genre g ON g.genre_id = fg.genre_id
JOIN salle s2 ON s2.salle_id = s.salle_id
JOIN lieu l ON l.lieu_id = s2.lieu_id
WHERE g.nom_genre ='Drame' AND s.date_seance = '2024-05-17'
AND s.heure_seance >= '17:00' AND s.vo = true
```

9. Le nombre moyen de séances par jour et par cinéma pour la semaine du 13 au 19 mai

```
WITH nb_jour AS (
SELECT l.nom, s.date_seance, COUNT(*) AS nb
FROM seance s
JOIN salle s2 ON s2.salle_id = s.salle_id
JOIN lieu l ON l.lieu_id = s2.lieu_id
WHERE s.date_seance BETWEEN '2024-05-13' AND '2024-05-19'
GROUP BY l.nom, s.date_seance
)
SELECT nom, AVG(nb) AS moyenne_par_jour
FROM nb_jour
GROUP BY nb_jour.nom
```

10. Les 5 salles ayant la plus grande capacité

```
SELECT l.nom, s.nom_salle, s.capacite
FROM salle s
JOIN lieu l ON l.lieu_id = s.lieu_id
ORDER BY s.capacite DESC
LIMIT 5;
```

11. Le nombre de films sortis en 2024 par genre

```
SELECT g.nom_genre, COUNT(*) AS nb
FROM film_genre fg
JOIN genre g ON g.genre_id = fg.genre_id
JOIN film f ON f.film_id = fg.film_id
WHERE f.date_sortie BETWEEN '2024-01-01' AND '2024-12-31'
GROUP BY g.nom_genre
ORDER BY nb ASC
```

9 Conclusions

9.1 Commentaires de fin

La base de données (et la maquette d'application associée) proposent un modèle évolutif simple mais efficace qui peut accompagner les évolutions de l'offre cinématographique genevoise Elle est relativement simple à maintenir grace à des règles d'intégrité assez strictes mais suffisamment souple pour permettre à l'application associée d'obtenir facilement les données qui comptent vraiment pour les utilisateurs (les séances et le tri de films par genre).

9.2 Limites

Il aurait été pertinent de créer des relations Acteur, Realisateur, Pays afin d'effectuer le même processus de validation des données que sur Film_genre et s'assurer qu'il n'existe aucun doublon d'un acteur, par exemple. Nous avons cependant jugé que trier les films par genre était le critère le plus déterminant et pour garder la base de données à une taille adéquate pour le projet, nous avons décidé de n'opérer cette validation que pour les genres. Pour une application complète qui veut offrir une recherche par réalisateur, acteur ou pays en s'assurant une correcte validation des données même pour un très grand nombre de films, il aurait fallu créer ces relations (et par conséquent une table FilmActeur pour inclure un nombre indéterminé d'acteurs par film)