COS 221

PRACTICAL ASSIGNMENT 5

GROUP 27 MEMBERS:

DAVID KALU
NAAZNEEN KHAN
JOHNA KRAUSE
TAYLOR SERGEL
MPHO TSOTETSI
PIETER VENTER
CORNE DE LANGE

Task 1: Research

What is the entertainment industry?

The entertainment industry is defined as "those involved in providing entertainment through radio, television, films and theatre". For this task we will focus solely on movie and television series.

Over the past 20 years, there has been a severe change in the way in which viewers watch movies and series, i.e:

- Cinema viewings are declining.
- DVD sales are dropping.
- TV, Cable TV/satellite is stable.
- Internet and social networks are proceeding to grow.
- And most popularly is streaming on mobile devices.

	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Cinema	858	885	945	941	1 113	1 044	1 077	1 113	1 153	1 193
% YOY		3.1%	6.8%	-0.4%	18.3%	-6.2%	3.2%	3.3%	3.6%	3.5%
Internet	12 679	17 443	25 154	32 533	39 402	45 004	50 729	56 581	62 546	68 471
% YOY		37.6%	44.2%	29.3%	21.1%	14.2%	12.7%	11.5%	10.5%	9.5%
TV and video	16 020	17 080	18 217	19 195	21 007	22 858	24 536	25 951	27 022	27 703
% YOY		6.6%	6.7%	5.4%	9.4%	8.8%	7.3%	5.8%	4.1%	2.5%

Table 1: Consumption Spending and of Media Entertainment between 2011-2020 (R Million)

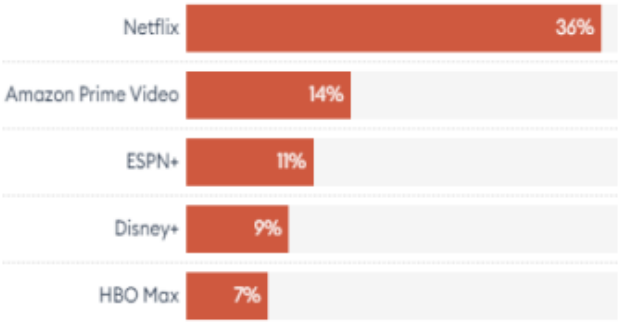


Figure 1: Best streaming services based on user experience.

The Rise of Streaming Services (Consumer Preference)

Streaming services such as Netflix, Hulu, Disney+, etc have become a huge part of today's film industry when it comes to watching movies and series. Changing the conventional way in which films are viewed.

In the past one could only watch movies at the cinemas but in today's day and age movies and series can be viewed at consumers convenience whether that be in the comfort of their living rooms or through the use of their mobile devices. The traditional distribution of film has been uprooted by the rise of streaming platforms, which has led to new release strategies. Films are now released simultaneously in theatres and on streaming platforms. This trend is gaining traction, with consumers preferring the flexibility these options provide.

The growing demand for these online streaming platforms is undeniable, with Netflix boasting 200 million subscribers and Amazon Prime hot on its heels with 100 million.

Popular Genres According to Box Office Sales

Viewers are often drawn to and enjoy specific genres according to personal preference, hence why genre categorisation is so important in the entertainment industry. Understanding your audience's interests and preferences helps filmmakers to stay ahead of competitors by producing content that resonates with viewers and drives engagement.

List of the main genres:

- Adventure - the explorations of a protagonist
- Action - contains events usually have violence/action
- Comedy - emphasizes humour
- Drama - portray serious real-life stories
- Thriller - mix of horror & crime genre
- Horror - designed to frighten, shock and disgust
- Romantic Comedy - subgenre of romance & comedy

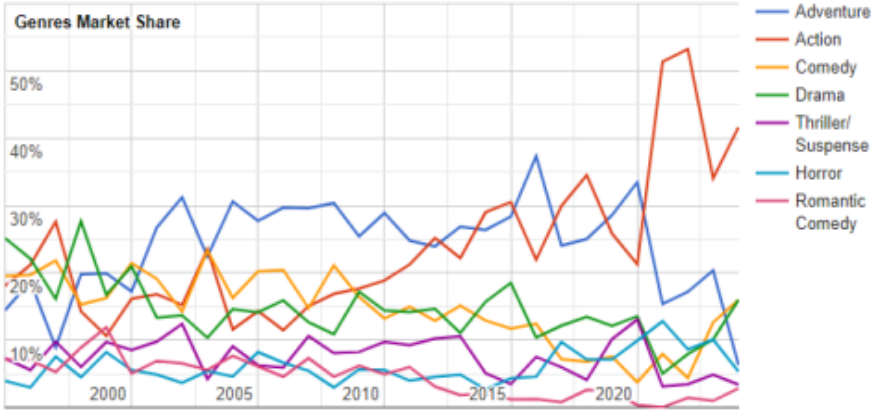


Figure 1: Line graph showing the popularity of genres based off Box Office Sales

Bibliography

AIContentfy team, 2023. *AIContentfy.* [Online]
Available at: https://aicontentfy.com/en/blog/importance-of-understanding-target-audience-for-content-distribution#:~:text=Understanding%20your%20audience's%20interests%20and%20preferences%20also%20helps%20you%20stay,with%20them%20and%20drives%20engagement.
[Accessed 6 November 2023].

Aldredge, J., 2023. *The Beat.* [Online]
Available at: https://www.premiumbeat.com/blog/guide-to-basic-film-genres/
[Accessed 12 September 2023].

Ana Durrani, 2024. *Top Streaming Statistics In 2024.* [Online]
Available at: https://www.forbes.com/home-improvement/internet/streaming-stats/
[Accessed 2 February 2024].

Cook, S., 2024. *Comparitech.* [Online]
Available at: https://www.comparitech.com/blog/vpn-privacy/netflix-statistics-facts-figures/
[Accessed 9 January 2024].

Creative Media Education, 2023. *Creative Media Education.* [Online]
Available at: https://www.sae.edu/gbr/insights/top-trends-in-modern-film-production-navigating-the-future-of-filmmaking/
[Accessed 2023].

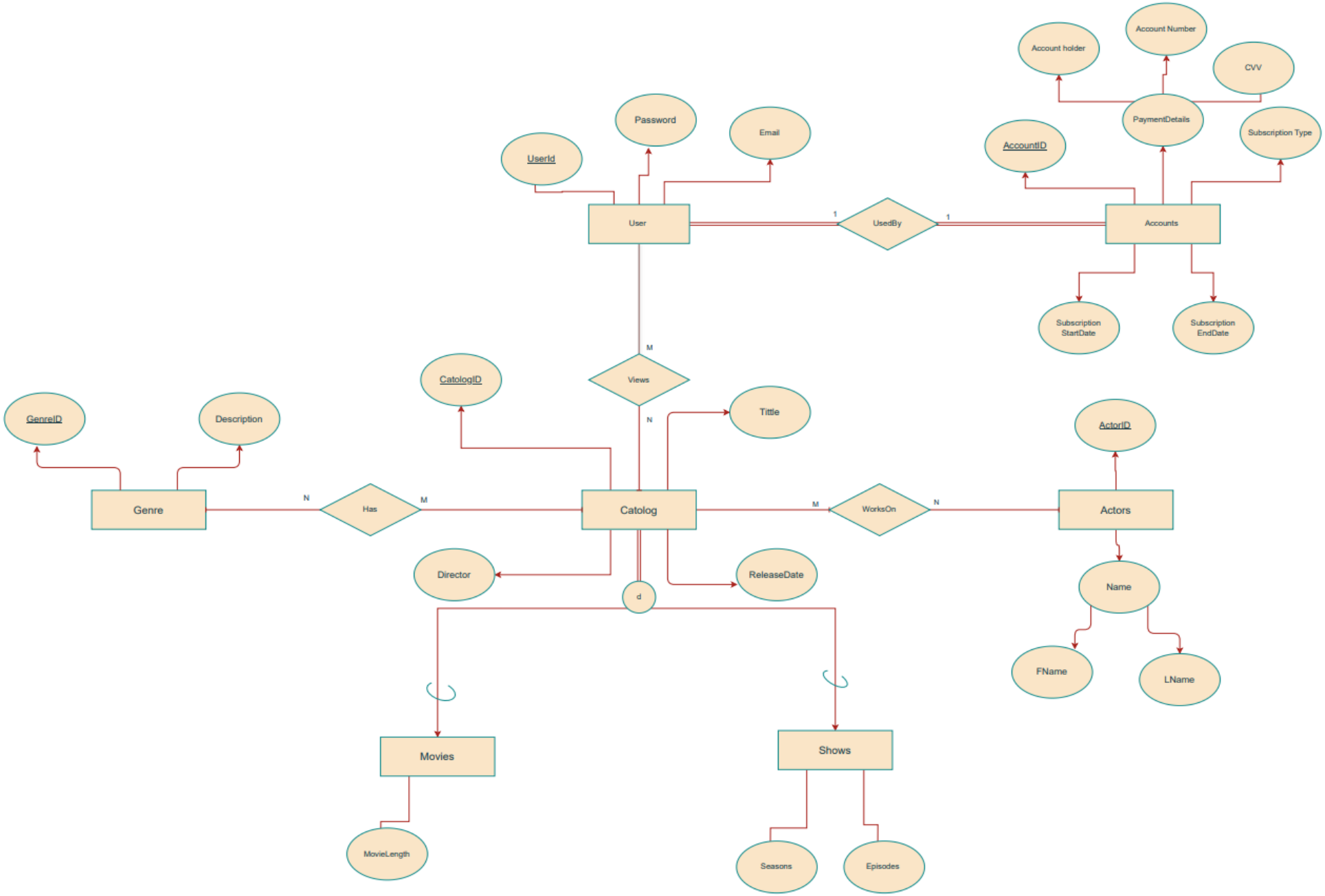
Jones, P., 2017. *RESEARCH INTO AUDIENCE TRENDS OF,* s.l.: KWAZULU-NATAL FILM COMMISSION.

Kumar, R., 2017. *ScienceDirect.* [Online]
Available at: https://www.sciencedirect.com/topics/social-sciences/entertainment-industry
[Accessed 2017].

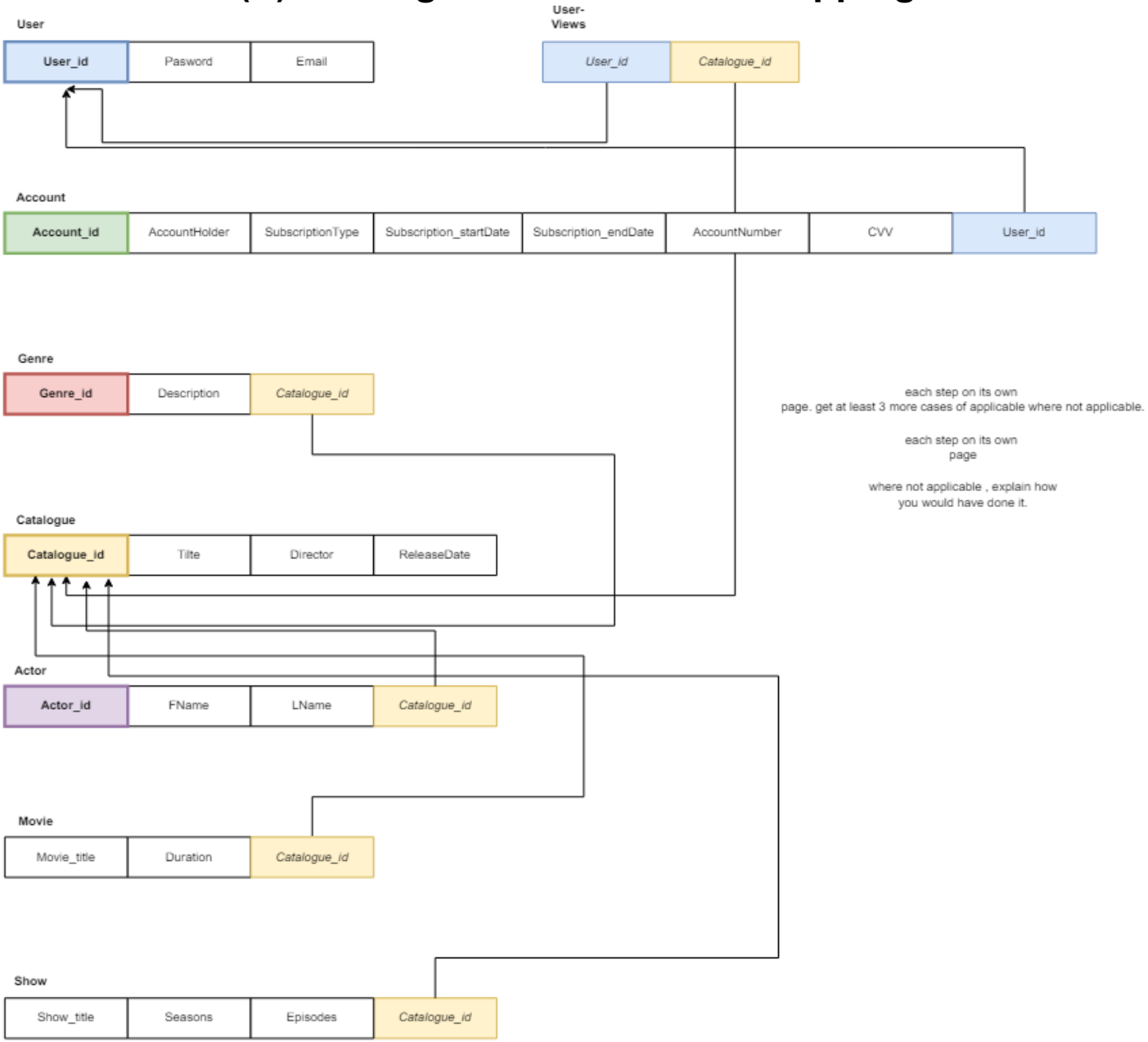
Nash Information Services, L., 2024. *Market Share for Each Genre 1995-2024.* [Online]
Available at: https://www.the-numbers.com/market/genres

Vocabulary.com Dictionary, 2024. *Entertainment industry.* [Online]
Available at: https://www.vocabulary.com/dictionary/entertainment industry
[Accessed 14 May 2024].

Task 2: (E)ER-Diagram



Task 3: (E)ER-Diagram to Relational Mapping



each step on its own
page. get at least 3 more cases of applicable where not applicable.

each step on its own
page

where not applicable , explain how
you would have done it.

MAPPING STEPS:

STEP 1

Mapping of strong entity types

STEP 2

~~Mapping of weak entity types.~~ *(not applicable in this case)*

STEP 3

Mapping of 1:1 relationships

- User & Account, by means of **User_id**

STEP 4

~~Mapping of 1:N relationships~~ *(not applicable in this case)*

STEP 5

Mapping of M:N relationships

- Now we can keep track of which user watches what catalogue. This data can be used to create a "favourites" feature to the website. where movies can be recommended to a user based on their catalogue view history

STEP 6

~~Mapping of multivalued attributes~~ *(not applicable in this case)*

STEP 7

~~Mapping of n-ary relationships~~ *(not applicable in this case)*

STEP 8

Mapping of specialization & generalization

- a catalogue must either be a movie or a show, cannot be both

STEP 8

~~Mapping of Union relationships~~ *(not applicable in this case)*

bold outline (primary keys)

faint with italics (foreign keys)

Task 4: Relational Schema

Genre(GenreID, Description, CatalogID)
Catalog(CatalogID, Title, Director, ReleaseDate)
Movies(Duration, CatalogID)
Shows(Seasons, Episodes, CatalogID)
User(UserID, Password, Email)
Actors(ActorID, Name(FName, LName), CatalogID)
Accounts(AccountID, PaymentDetails(AccountHolder, AccountNumber, CVV), SubscriptionType, SubscriptionStartDate, SubscriptionEndDate, UserID)
User-Views(UserID, CatalogID)

Primary keys:

- Genre Table: GenreID
- Catalog Table: CatalogID
- User Table: UserID
- Actors Table: ActorID
- Accounts Table: AccountID

Secondary keys:

- User Table: Email can be used as a secondary key

Foreign keys:

- Genre Table: CatalogID
- Movies Table: CatalogID
- Shows Table: CatalogID
- Actors Table: CatalogID
- Accounts Table: UserID
- User-Views Table: UserID, CatalogID

Constraints:
Primary keys cannot be NULL

Genre

Attribute	Datatype
GenreID (PK)	INTEGER
Description	VARCHAR(100)
CatalogID (FK)	INTEGER

Catalog

Attribute	Datatype
CatalogID (PK)	INTEGER
Title	VARCHAR(255)
Director	VARCHAR(100)
ReleaseDate	YEAR

Movies

Attribute	Datatype
Duration	TIME
CatalogID (FK)	INTEGER

Shows

Attribute	Datatype
Seasons	INTEGER
Episodes	INTEGER
CatalogID (FK)	INTEGER

User

Attribute	Datatype
UserID (PK)	INTEGER
Password	VARCHAR(100)
Email	VARCHAR(255)

Actors

Attribute	Datatype

ActorID (PK)	INTEGER
FName	VARCHAR(150)
LName	VARCHAR(150)
CatalogID (FK)	INTEGER

Accounts

Attribute	Datatype
AccountID (PK)	INTEGER
AccountHolder	VARCHAR(150)
AccountNumber	VARCHAR(255)
CVV	VARCHAR(4)
SubscriptionType	VARCHAR(50)
SubscriptionStartDate	DATE
SubscriptionEndDate	DATE
UserID (FK)	INTEGER

User-Views

Attribute	Datatype
UserID (FK)	INTEGER
CatalogID (FK)	INTEGER

Relational Database Schema:

Genre Table

GenreID	Description	CatalogID
1	Action	1
2	Comedy	2
3	Drama	3

Catalog Table

CatalogID	Title	Director	ReleaseDate
1	The Matrix	Lana Wachowski	1999-03-31
2	Inception	Christopher Nolan	2010-07-16
3	The Shawshank Redemption	Frank Darabont	1994-09-23

Movies Table

Duration	CatalogID
02:16:00	1
02:28:00	2
02:22:00	3

Shows Table

Seasons	Episodes	CatalogID
5	122	5
2	10	10
1	25	23

User Table

UserID	Password	Email
1	Password123	User1@example.com
2	Abc123	User2@gmail.com
3	Pass456	User3@yahoo.com

Actors Table

ActorID	FName	LName	CatalogID
1	Keanu	Reeves	1
2	Leonardo	DiCaprio	2
3	Tim	Robbins	3

Accounts Table

AccountID	AccountHolder	AccountNumber	CVV	SubscriptionType	SubscriptionStartDate	SubscriptionEndDate	UserID
1	John Doe	123456789	123	Premium	2024-01-01	2024-12-31	1
2	Jane Doe	987654321	456	Basic	2024-02-01	2024-12-31	2
3	Mary Doe	111122223	789	Premium	2024-03-01	2024-12-31	3

User-Views Table

UserID	CatalogID
1	4
2	5
3	6

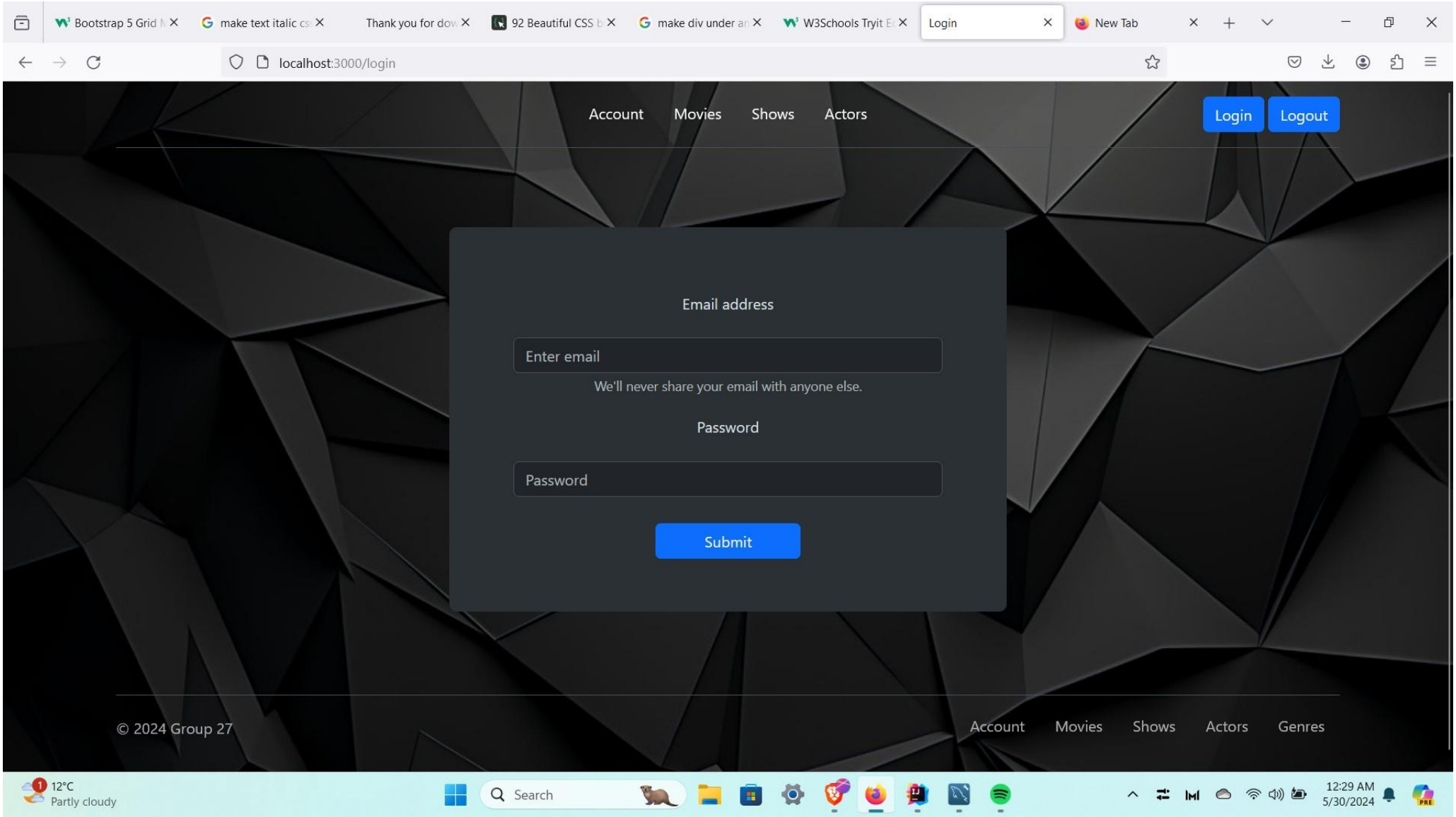
SQL statements:

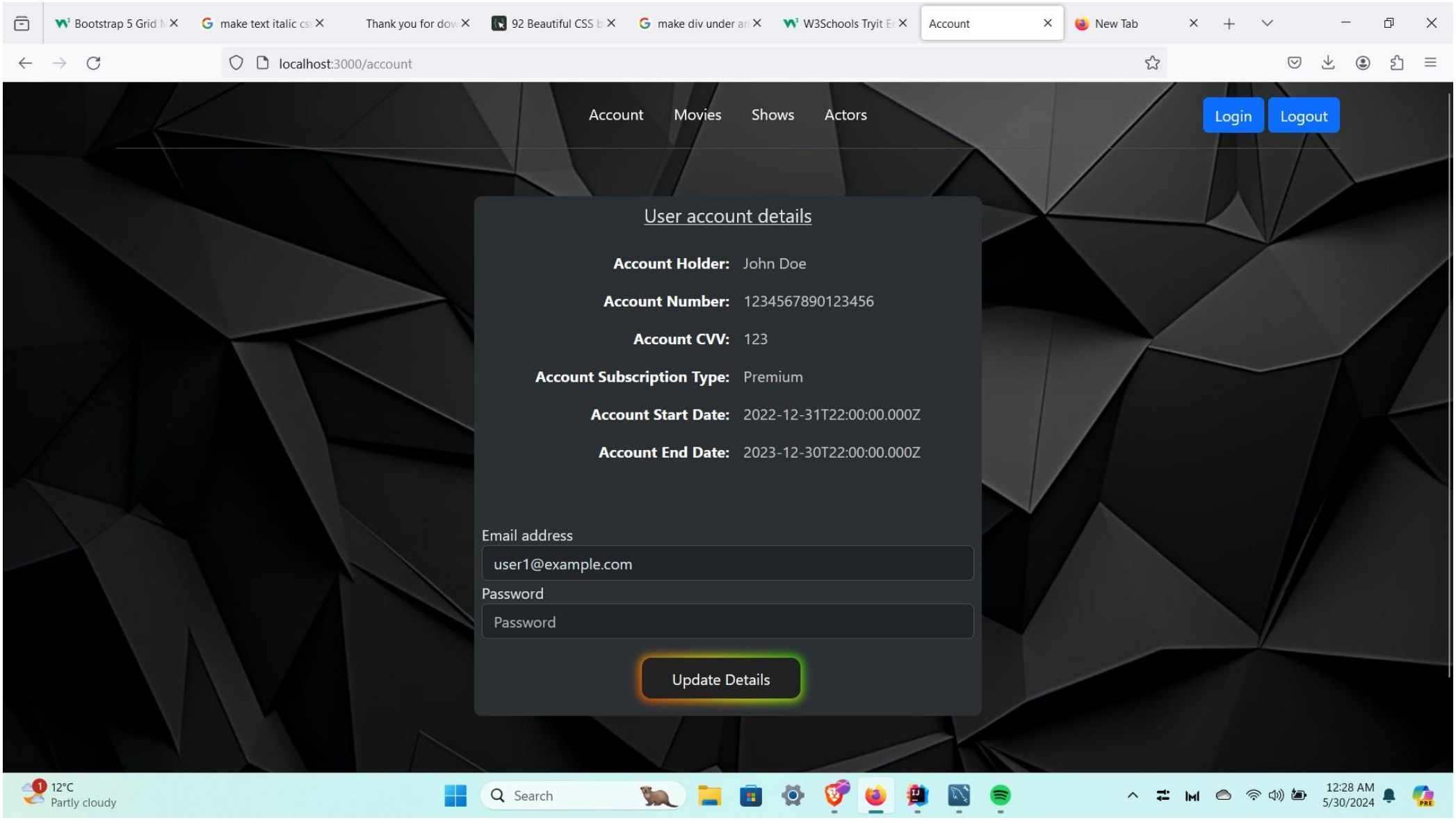
```
CREATE TABLE Genre (
   GenreID INTEGER AUTO_INCREMENT PRIMARY KEY,
   Description VARCHAR(100),
   CatalogID INTEGER,
   FOREIGN KEY (CatalogID) REFERENCES Catalog(CatalogID)
);

CREATE TABLE Catalog (
   CatalogID INTEGER PRIMARY KEY,
   Title VARCHAR(255),
   Director VARCHAR(100),
   ReleaseDate YEAR
);

CREATE TABLE Movies (
   Duration TIME,
   CatalogID INTEGER,
```

```
    FOREIGN KEY (CatalogID) REFERENCES Catalog(CatalogID)
);

CREATE TABLE Shows (
    Seasons INTEGER,
    Episodes INTEGER,
    CatalogID INTEGER,
    FOREIGN KEY (CatalogID) REFERENCES Catalog(CatalogID)
);

CREATE TABLE User (
    UserID INTEGER AUTO_INCREMENT PRIMARY KEY,
    Password VARCHAR(100),
    Email VARCHAR(255)
);

CREATE TABLE Actors (
    ActorID INTEGER AUTO_INCREMENT PRIMARY KEY,
    FName VARCHAR(150),
    LName VARCHAR(150),
    CatalogID INTEGER,
    FOREIGN KEY (CatalogID) REFERENCES Catalog(CatalogID)
);

CREATE TABLE Accounts (
    AccountID INTEGER AUTO_INCREMENT PRIMARY KEY,
    AccountHolder VARCHAR(150),
    AccountNumber VARCHAR(255),
    CVV VARCHAR(4),
    SubscriptionType VARCHAR(50),
    SubscriptionStartDate DATE,
    SubscriptionEndDate DATE,
```

```
    UserID INTEGER,
    FOREIGN KEY (UserID) REFERENCES User(UserID)
);

CREATE TABLE User_Views (
    UserID INTEGER,
    CatalogID INTEGER,
    FOREIGN KEY (UserID) REFERENCES User(UserID),
    FOREIGN KEY (CatalogID) REFERENCES Catalog(CatalogID)
);
```

Task 5: Web-based Application




Account
Movies
Shows
Actors
Login
Logout
User account details
Account Holder: John Doe
Account Number: 1234567890123456
Account CVV: 123
Account Subscription Type: Premium
Account Start Date: 2022-12-31T22:00:00.000Z
Account End Date: 2023-12-30T22:00:00.000Z
Email address
user1@example.com
Password
Password
Update Details

Bootstrap 5 Grid | make text italic cs | Thank you for dow | 92 Beautiful CSS b | make div under an | W3Schools Tryit E | Movies | New Tab

localhost:3000/movies

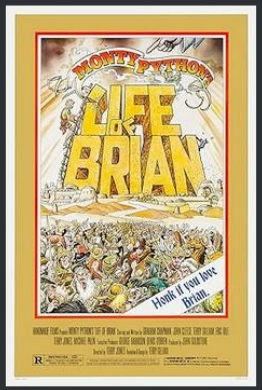
Account Movies Shows Actors

Login Logout

Filters:

Title: Director: Genre: Release Date (Year): Apply Filters

Life of Brian



Director: *Terry Jones*

Release Date: *1979*

Duration: *01:34:00*

Genre: *comedy*

Heroes

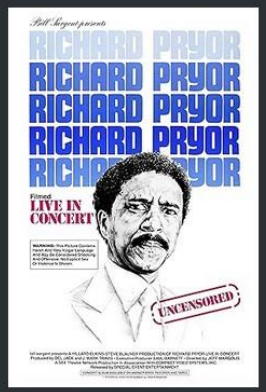
Director: *Jeremy Kagan*

Release Date: *1977*

Duration: *01:53:00*

Genre: *drama, comedy, romance*

Richard Pryor: Live in Concert



Director: *Jeff Margolis*

Release Date: *1979*

Duration: *01:18:00*

Genre: *comedy, documentation*

Bandie

Director: *Alo Sarkar*

Release Date: *1978*

Duration: *02:21:00*

Genre: *drama, romance, action*

Prince

Director: *Lekh Tandon*

Release Date: *1969*

Duration: *02:32:00*

Genre: *romance*

FTA

Director: *Francine Parker*

Release Date: *1972*

Duration: *01:37:00*

Genre: *comedy, documentation, music*

12°C Partly cloudy | Search | 12:29 AM 5/30/2024

Bootstrap 5 Grid | make text italic cs | Thank you for dow | 92 Beautiful CSS b | make div under an | W3Schools Tryit E | Shows | New Tab

localhost:3000/shows

Account Movies Shows Actors

Login Logout

Filters:

Title: Director: Seasons: Release Date (Year): Genre:

Apply Filters

Neon Genesis Evangelion

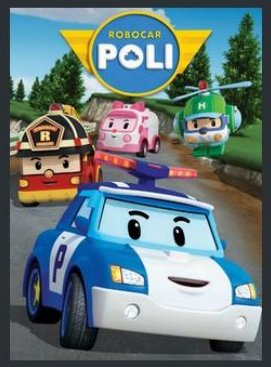
Director: Hideaki Anno

Release Date: 1995

Seasons 1

8

Robocar Poli



Director: Jun-Young Um

Release Date: 2011

Seasons 4

32

Larva

Director: Maeng Joo-gong

Release Date: 2011

Seasons 5

40

Rainbow Ruby

Director: Shin Tae sik

Release Date: 2016

Seasons 5

40

Lego Bionicle: The Journey to One

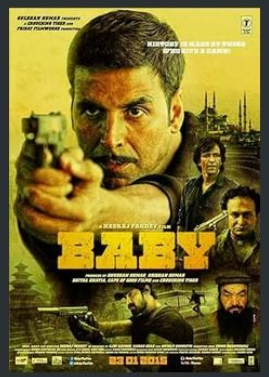
Director: Nicolas-Denis Robitaille

Release Date: 2016

Seasons 2

16

Baby



Director: channel w

Release Date: 2018

Seasons 3

Episodes: 24

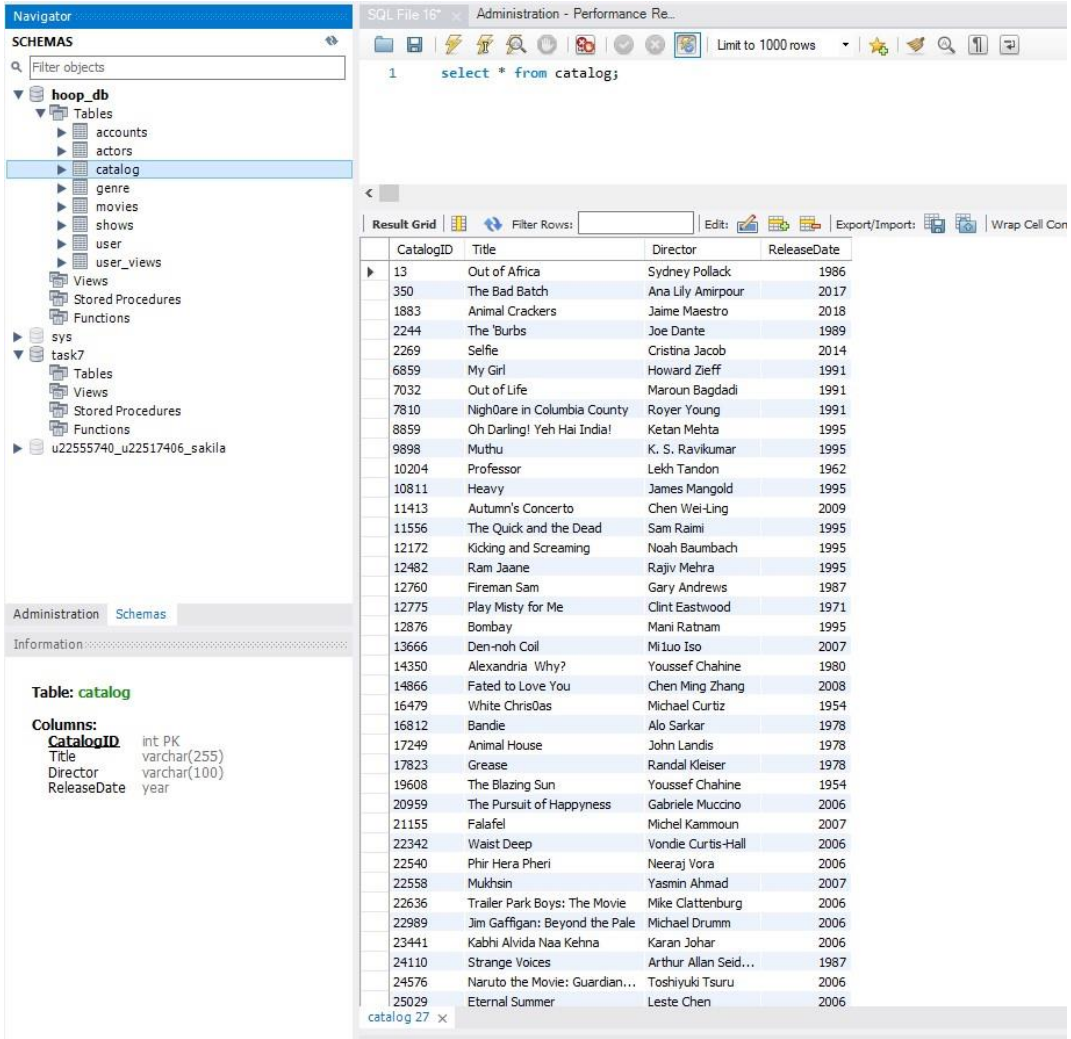
12°C Partly cloudy

Search

12:29 AM 5/30/2024

Task 6: Data

For the following task to populate the database used the data from https://www.kaggle.com/datasets/dgoenrique/netflix-movies-and-tv-shows for all the Catalog, Movies, Shows, Genres, and Actors tables. Then created a program to convert the data in tables to an Insert sql which was then used to insert all the values into the DataBase. For the User and Accounts tables used chatgpt to generate 50 mock data values. For the user_views table randomly chose userID's to reference to Catalog's CatalogID."



Screenshot to show an example of our Database that has been populated with data

Task 7: Analyse and Optimise

The below query is used to retrieve information about the different movies:

```
EXPLAIN SELECT distinct c.CatalogID, c.Title, c.Director, c.ReleaseDate, c.PosterUrl, m.Duration, g.Description as Genre
FROM movies AS m, catalog AS c, genre AS g
WHERE m.CatalogID = c.CatalogID AND c.CatalogID = g.CatalogID AND c.PosterUrl IS NOT NULL;
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	g	NULL	ALL	CatalogID	NULL	NULL	NULL	3621	100.00	Using where; Using temporary
1	SIMPLE	c	NULL	eq_ref	PRIMARY,CatalogID	PRIMARY	4	prac_5.g.CatalogID	1	90.00	Using where
1	SIMPLE	m	NULL	ref	CatalogID	CatalogID	5	prac_5.g.CatalogID	1	100.00	NULL

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.04700000

Timing (as measured by the server):
Execution time: 0:00:0.03245150
Table lock wait time: 0:00:0.00000400

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 669
Rows examined: 7911

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 1

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event Id: 1571
Thread Id: 141

According to the above query statistics analysis Execution time measured at client side is 0.047 seconds and the Execution time measured by the server is 0.03245150 seconds. One temporary table is created, meaning that the intermediate results are stored temporarily, which can slow down the performance. The temporary table is created due to the 'DISTINCT' keyword, which eliminates duplicate rows, causing additional overhead. The 'DISTINCT' keyword was used due to our initial dataset containing duplicate values; however, all our data must be unique therefore after deleting all duplicate values in our dataset we can see that optimization can be improved by removing the 'DISTINCT' keyword from the query.

```sql
EXPLAIN SELECT c.CatalogID, c.Title, c.Director, c.ReleaseDate, c.PosterUrl, m.Duration, g.Description as Genre
FROM movies AS m, catalog AS c, genre AS g
WHERE m.CatalogID = c.CatalogID AND c.CatalogID = g.CatalogID AND c.PosterUrl IS NOT NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	g	NULL	ALL	CatalogID	NULL	NULL	NULL	3621	100.00	Using where
1	SIMPLE	c	NULL	eq_ref	PRIMARY,CatalogID	PRIMARY	4	prac_5.g.CatalogID	1	90.00	Using where
1	SIMPLE	m	NULL	ref	CatalogID	CatalogID	5	prac_5.g.CatalogID	1	100.00	NULL

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.01500000

Timing (as measured by the server):
Execution time: 0:00:0.02784640
Table lock wait time: 0:00:0.00000400

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 669
Rows examined: 7911

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event Id: 1510
Thread Id: 141

*side note: it is confirmed that there are no duplicate values contained in the dataset as the number of rows sent to the client remained the same, hence no need for the 'DISTINCT' keyword.

After **optimising** our query **by removing** the '**DISTINCT'** keyword, we can observe the following performance gains achieved:

Execution time has decreased as the overhead (sorting and comparing rows) associated with removing duplicate values has been eliminated.

- Execution time measured at client side reduced from 0.047 seconds to 0.015 seconds.
- Execution time measured by the server reduced from 0.03245150 seconds to 0.02784640 seconds.

No temporary table was created as there is no need for a temporary table to store intermediate results associated with finding duplicate values.

- Leading to faster execution and better performance.

Group members' contributions:

DAVID KALU – Task 3 & CSS styling for web page
NAAZNEEN KHAN – Task 4
JOHNA KRAUSE – Task 5
TAYLOR SERGEL – Task 1 & Task 7
MPHO TSOTETSI – Task 2 & Populating web page with images
PIETER VENTER – Task 6
CORNE DE LANGE – Task 5