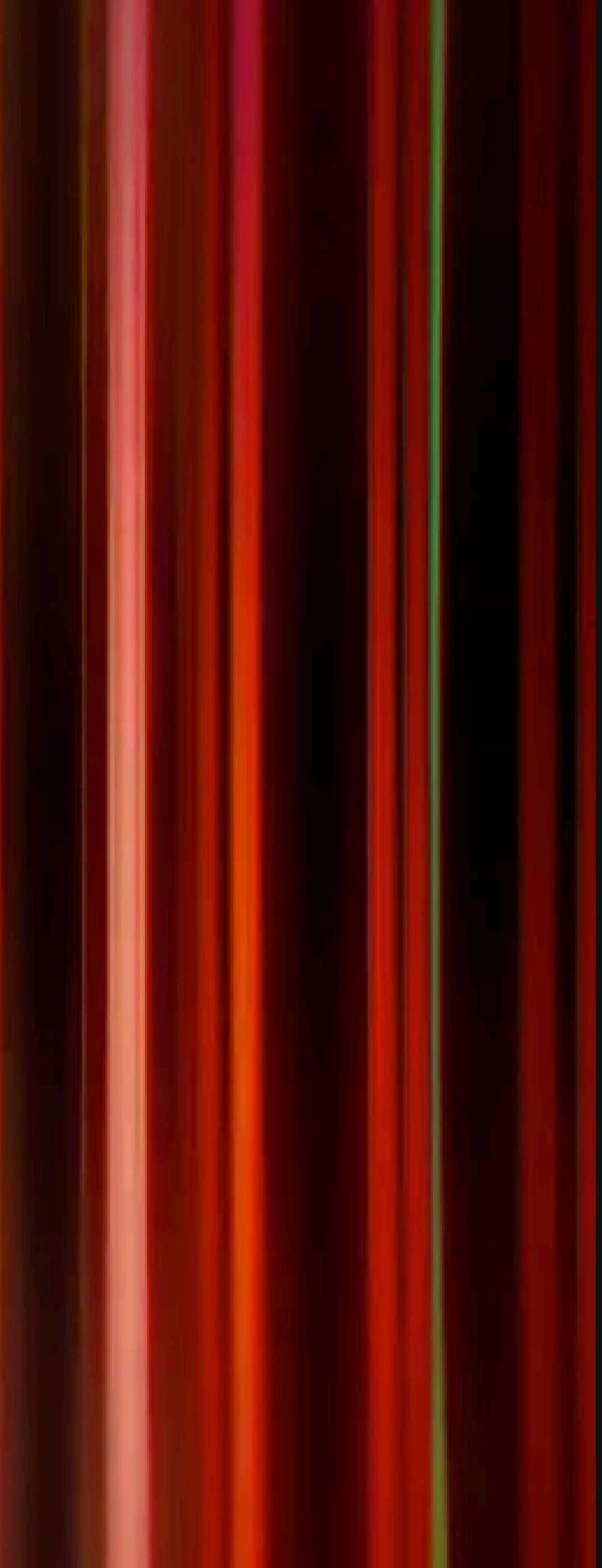


Youssef Agour - 2024

A Logical Design Implementation of Netflix’s Database

My attempt to pick apart and build from the ground up a simplified version of Netflix’s Database using SQL.

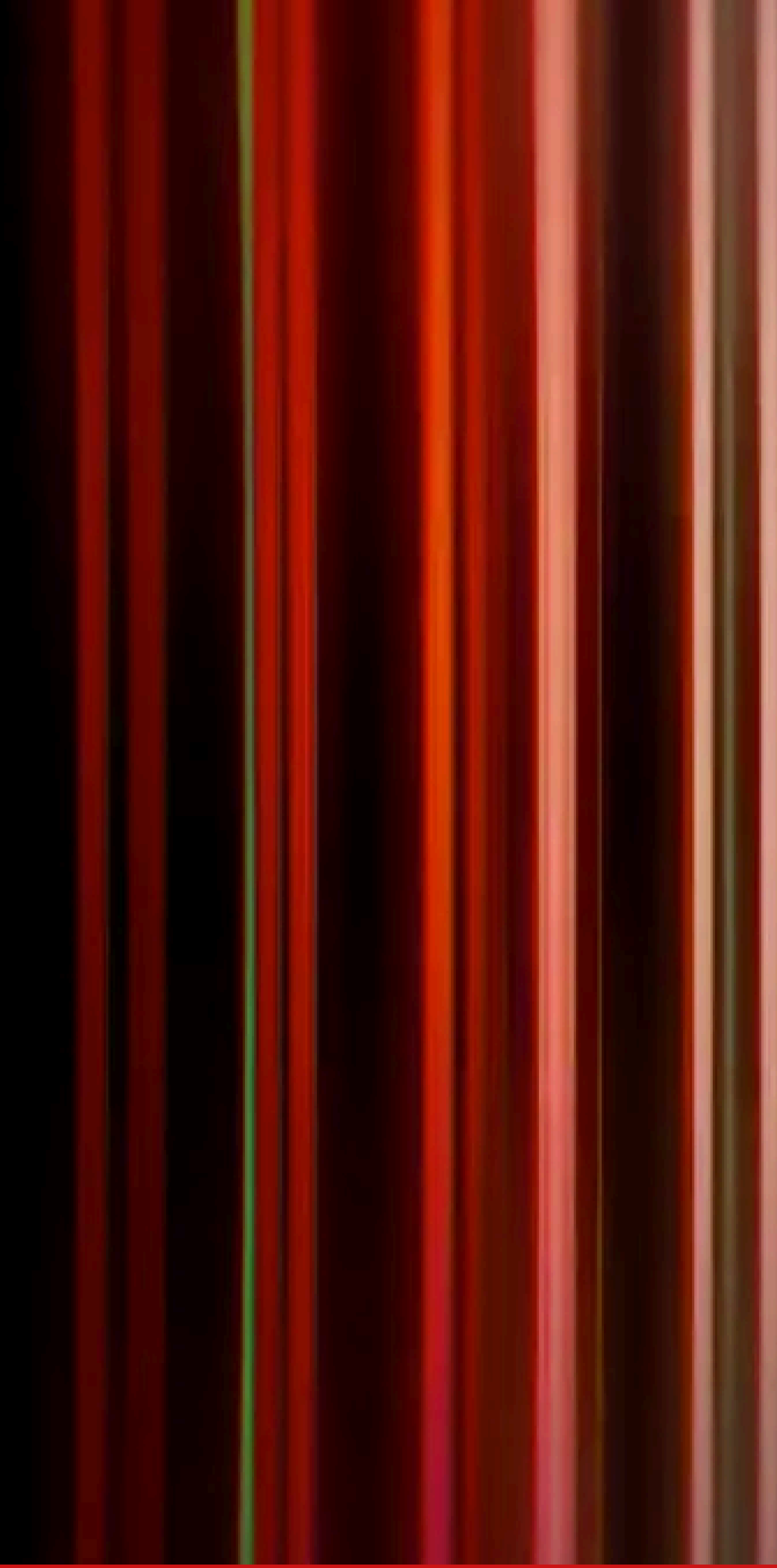
ABOUT THE BUSINESS

Netflix is an online movie streaming platform for watching movies, series and original content. This is done through an internet subscription on smart televisions (TVs), smartphones, tablets, streaming media players, and game consoles. They are a multimedia subscription based company with headquarters in the United States.

PRELIMINARY ASSUMPTIONS

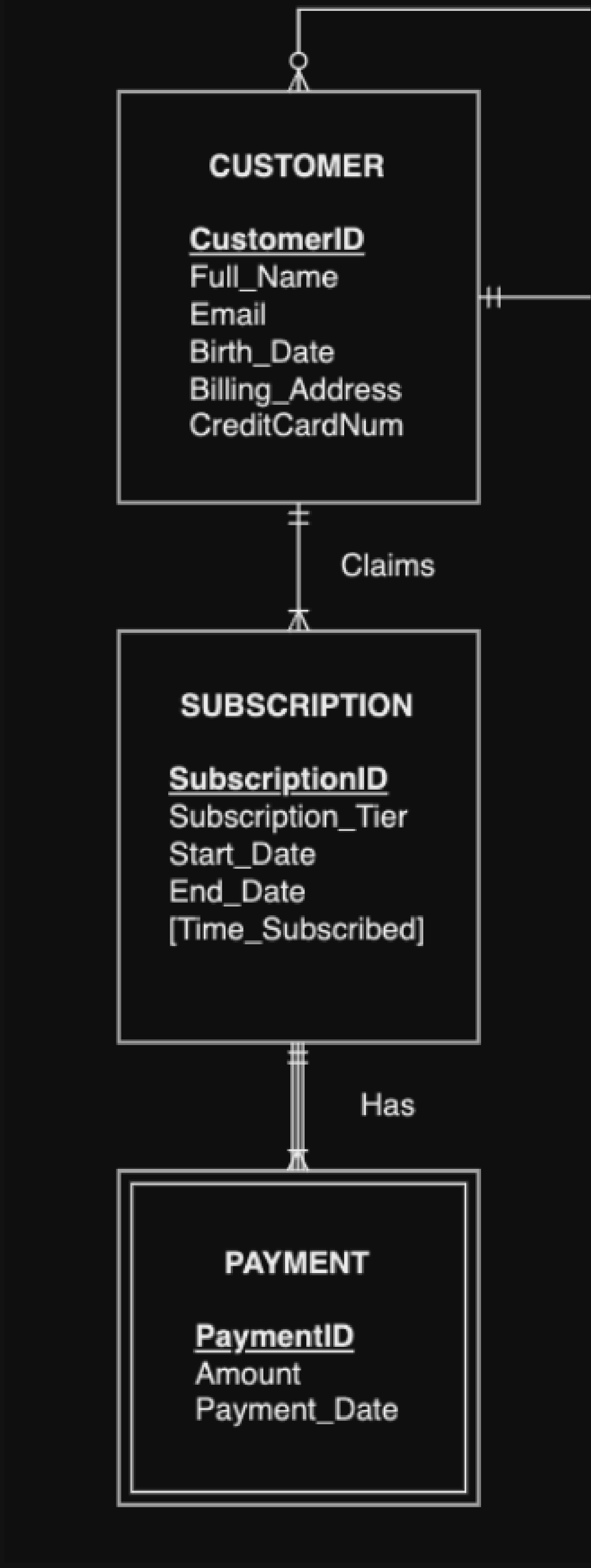
For the purposes of execution, I have chosen to make the following assumptions in the design of our database after examining Netflix's application architecture:

- Database will be solely for the use of Netflix USA
- Payments can only be made via credit card alone
- Data is only kept on movies and shows, excluding game downloads
- Payments processed for a subscription once paid cannot be refunded



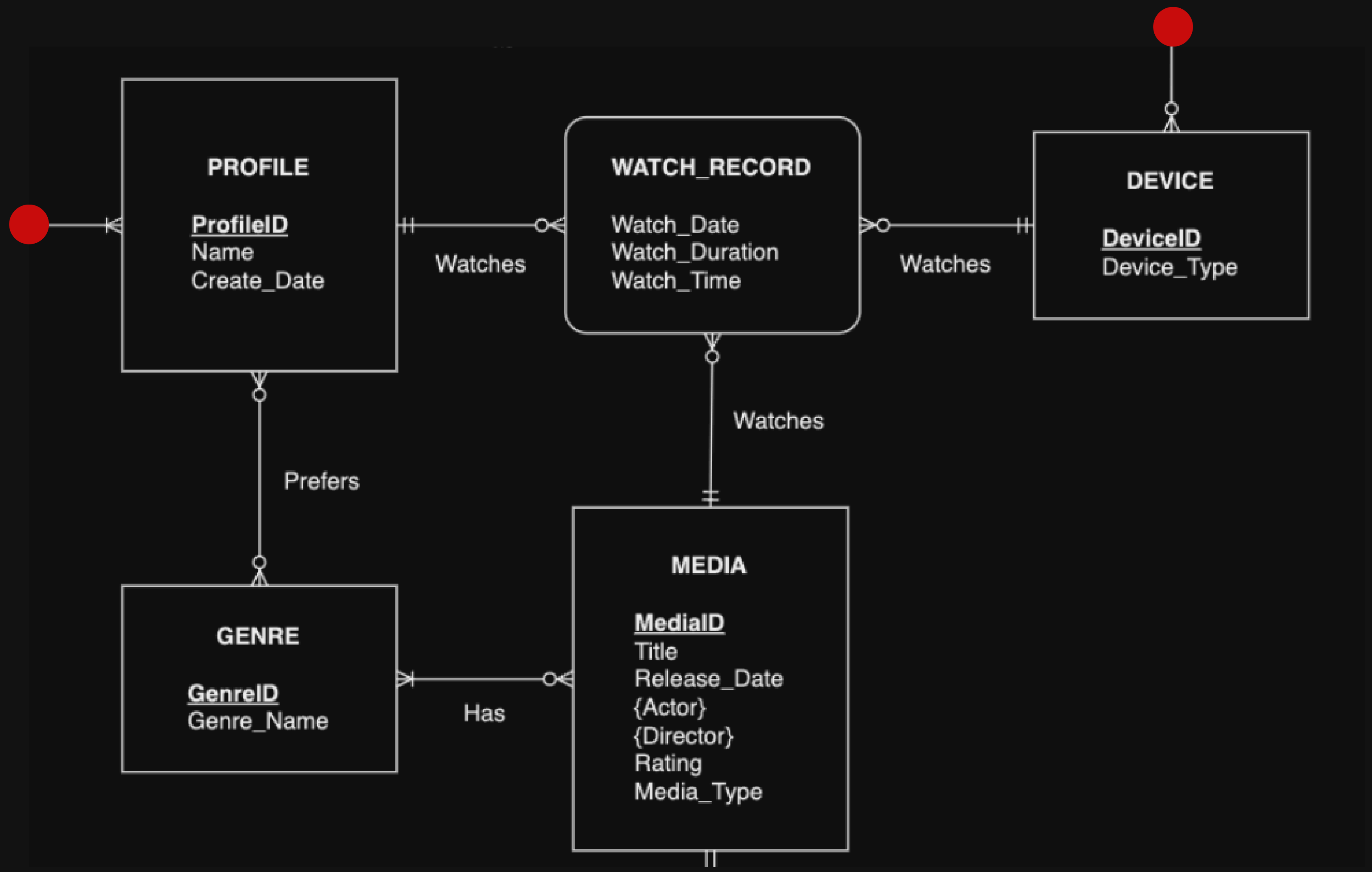
REQUIREMENTS - ABOUT CUSTOMERS

- CUSTOMER is an entity with attributes CustomerID, Full_Name, Email, Birth_Date, Billing_Address, and CreditCardNo.
- Netflix needs to track both current and historical customers, each customer can only have one subscription at a time, but customers throughout their time with Netflix can downgrade or upgrade their plan.
- SUBSCRIPTION can be identified by SubscriptionID and has attributes Subscription_Tier, Start_Date, End_Date (If Applicable), and Time_Subscibed is derived from the Start_Date Attribute.
- A single subscription can belong to one and only one customer.
- Regardless of their plan, customers create profiles. One customer must have at least one profile, but a profile links to one and only one customer.
- Payments Identified by PaymentID and SubscriptionID have the attributes Amount and Payment_Date. One or more payments can be made to one and only one subscription.



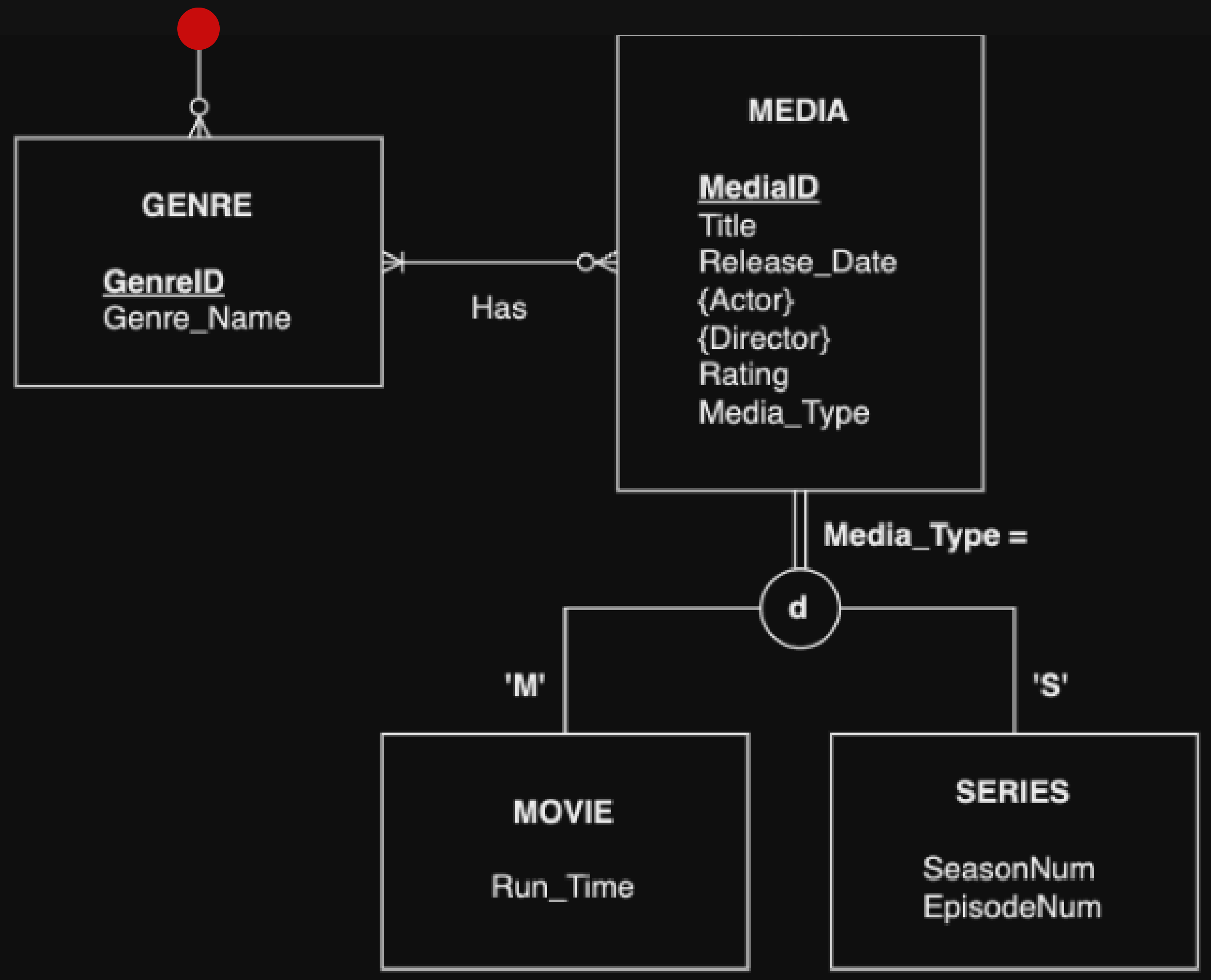
REQUIREMENTS - ABOUT WATCH RECORDS

- PROFILE is an entity with attributes ProfileID, Name and Create_Date.
- DEVICE is an entity with attributes DeviceID and Device_Type.
- PROFILES can watch zero/one/many content on different devices which is then recorded in the WATCH_RECORD, whilst a DEVICE can be used to watch from multiple profiles
- Netflix wishes to track to record WATCH DATE, DURATION and WATCH TIME, the length of time a particular profile watches and the time of the day they watch on a given device.

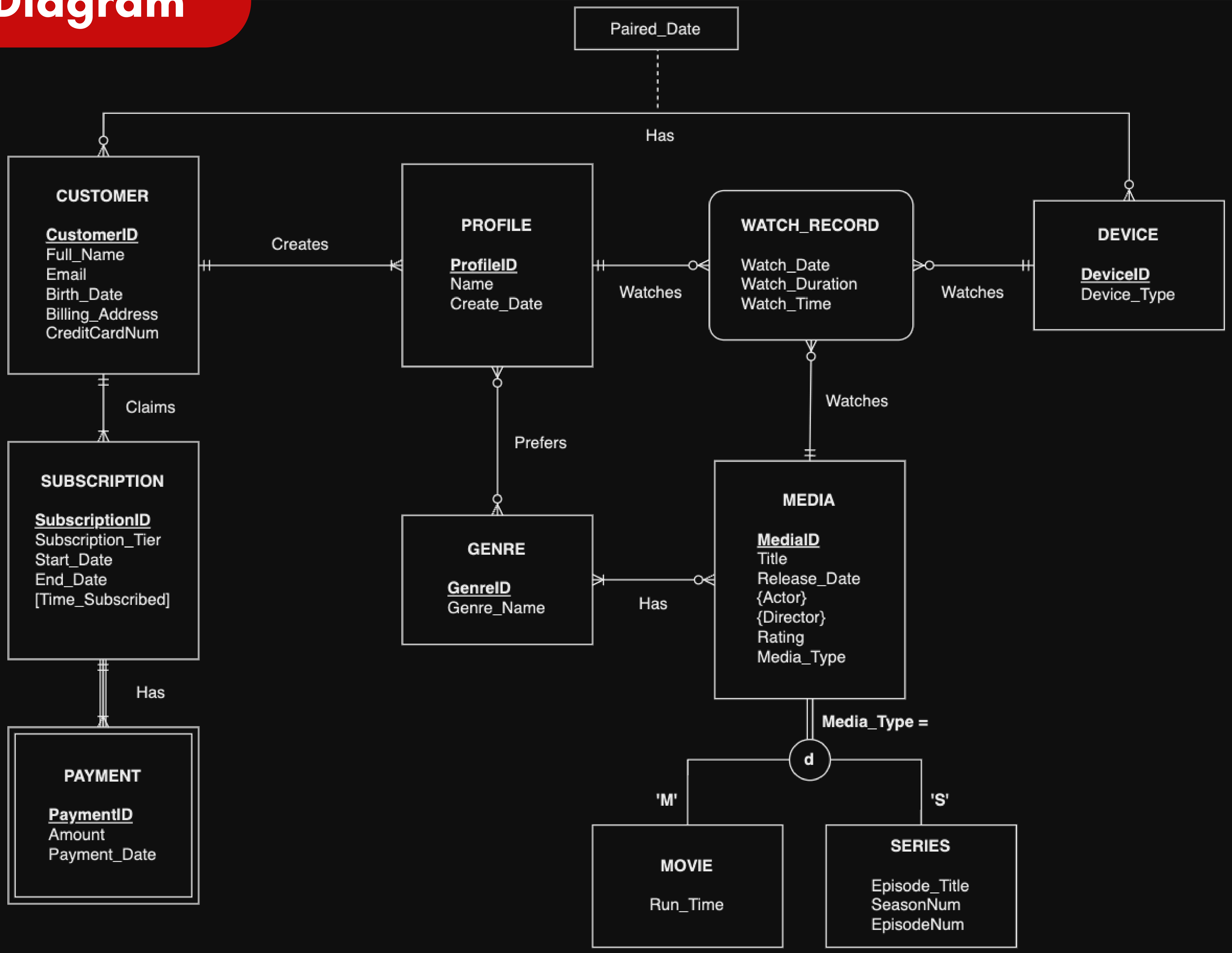


REQUIREMENTS - ABOUT MEDIA

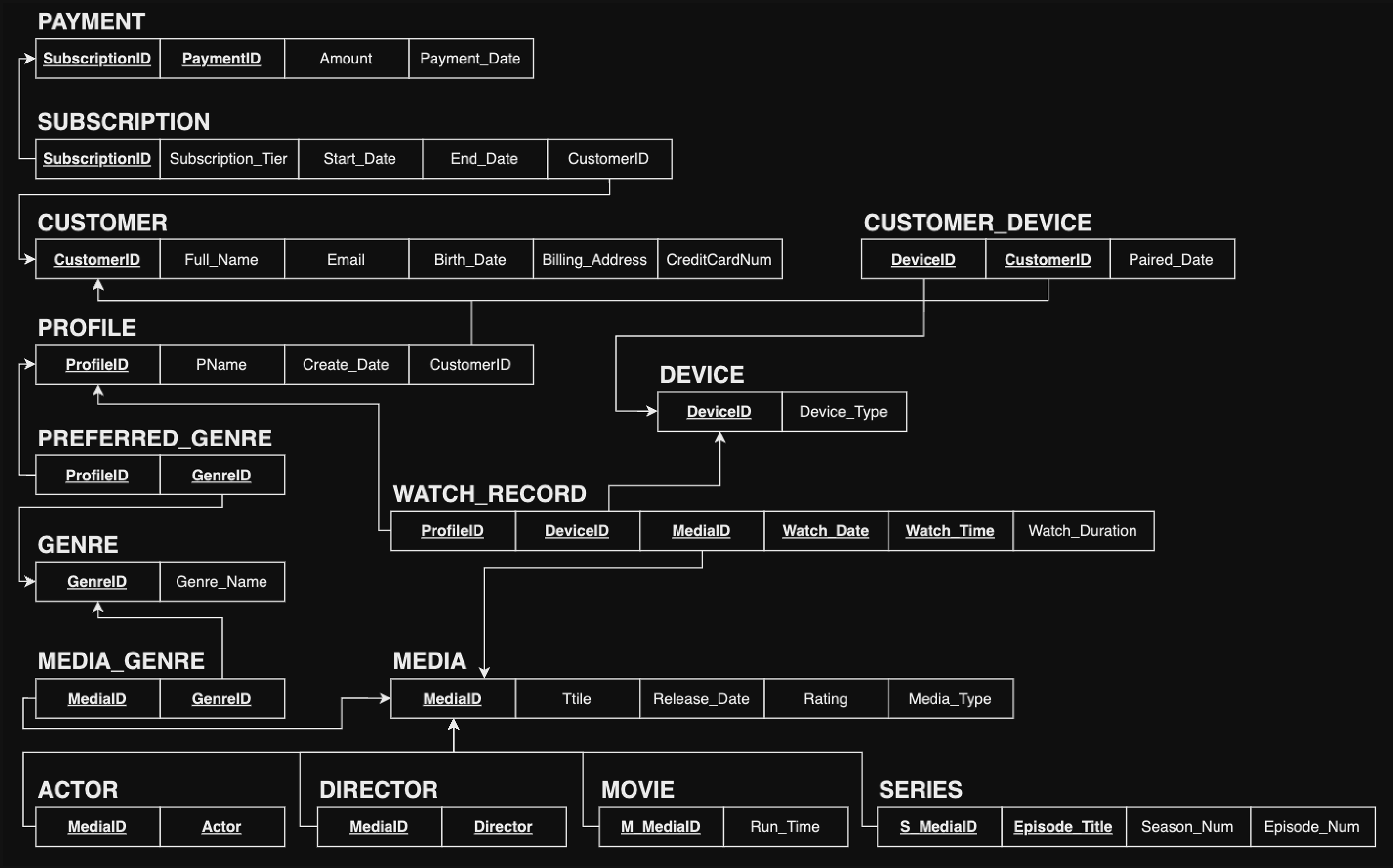
- MEDIA is an entity defined by MediaID with attributes Title, Release_Date, Actor(s), Director(s) and Rating.
- There are only two types of media that a profile can watch at a single time, MOVIE or SERIES.
- MOVIE is a subtype of MEDIA with unique attribute Run_Time.
- SERIES is a subtype of MEDIA with unique attributes, SeasonNum and EpisodeNum.
- Any media belongs to at least one genre, whilst a genre can belong to any number of media.
- GENRE is an entity defined by GenreID with attribute Genre_Name.
- Netflix also wishes to track popular content genres:
- A profile is also able to favourite up to 6 genres if they wish to do so and a genre can be favourited by any number of profiles.



Theorhetical ER Diagram



LOGICAL DESIGN ADAPTATION



PAYMENT

SubscriptionID	**PaymentID**	Amount	Payment_Date

SUBSCRIPTION

SubscriptionID	Subscription_Tier	Start_Date	End_Date	CustomerID

CUSTOMER

CustomerID	Full_Name	Email	Birth_Date	Billing_Address	CreditCardNum

CUSTOMER_DEVICE

DeviceID	**CustomerID**	Paired_Date

PROFILE

ProfileID	PName	Create_Date	CustomerID

DEVICE

DeviceID	Device_Type

PREFERRED_GENRE

ProfileID	**GenreID**

WATCH_RECORD

ProfileID	**DeviceID**	**MediaID**	**Watch_Date**	**Watch_Time**	Watch_Duration

GENRE

GenreID	Genre_Name

MEDIA_GENRE

MediaID	**GenreID**

MEDIA

MediaID	Ttile	Release_Date	Rating	Media_Type

ACTOR

MediaID	**Actor**

DIRECTOR

MediaID	**Director**

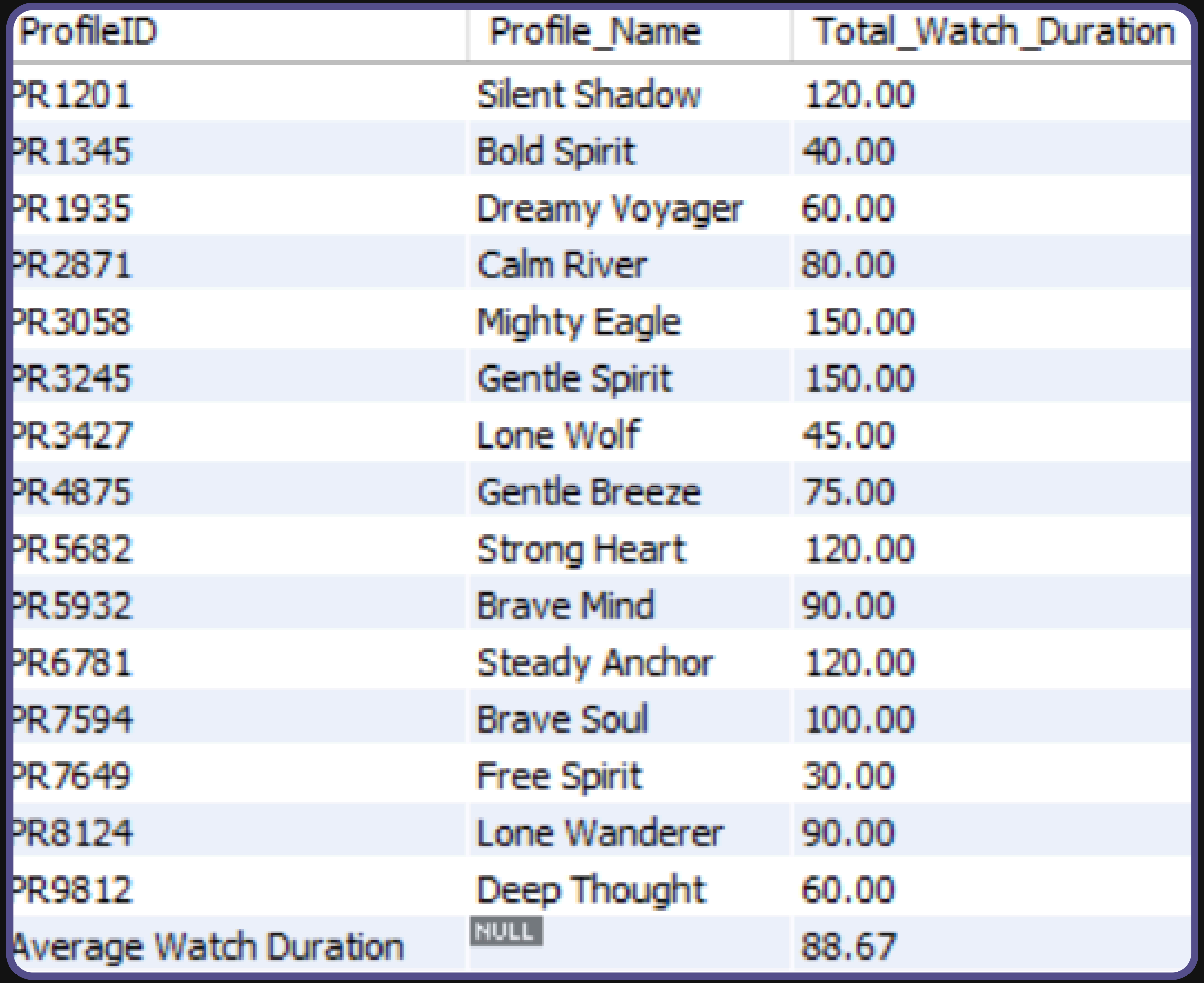
MOVIE

M_MediaID	Run_Time

SERIES

S_MediaID	**Episode_Title**	Season_Num	Episode_Num

```
SELECT P.ProfileID, P.PName AS Profile_Name,
SUM(W.Watch_Duration) AS Total_Watch_Duration
FROM Profile P
INNER JOIN Watch_Record W ON P.ProfileID = W.ProfileID
GROUP BY P.ProfileID, P.PName
UNION ALL
SELECT 'Average Watch Duration' AS Metric, NULL AS Year,
ROUND(AVG(w.Watch_Duration), 2) AS Value
FROM PROFILE p
INNER JOIN WATCH_RECORD w ON p.ProfileID = w.ProfileID;
```



ProfileID	Profile_Name	Total_Watch_Duration
PR1201	Silent Shadow	120.00
PR1345	Bold Spirit	40.00
PR1935	Dreamy Voyager	60.00
PR2871	Calm River	80.00
PR3058	Mighty Eagle	150.00
PR3245	Gentle Spirit	150.00
PR3427	Lone Wolf	45.00
PR4875	Gentle Breeze	75.00
PR5682	Strong Heart	120.00
PR5932	Brave Mind	90.00
PR6781	Steady Anchor	120.00
PR7594	Brave Soul	100.00
PR7649	Free Spirit	30.00
PR8124	Lone Wanderer	90.00
PR9812	Deep Thought	60.00
Average Watch Duration	NULL	88.67

This query provides the total watch duration for each profile to understand how long each profile has spent watching content.

This can be used to assess how (much) each profile watches over time, with a comparison to the average watch time. which could give valuable insights into what personal recommendations to provide each profile if it were compared to the previous query for example.

Profile Watch Duration

Consumer Analytics

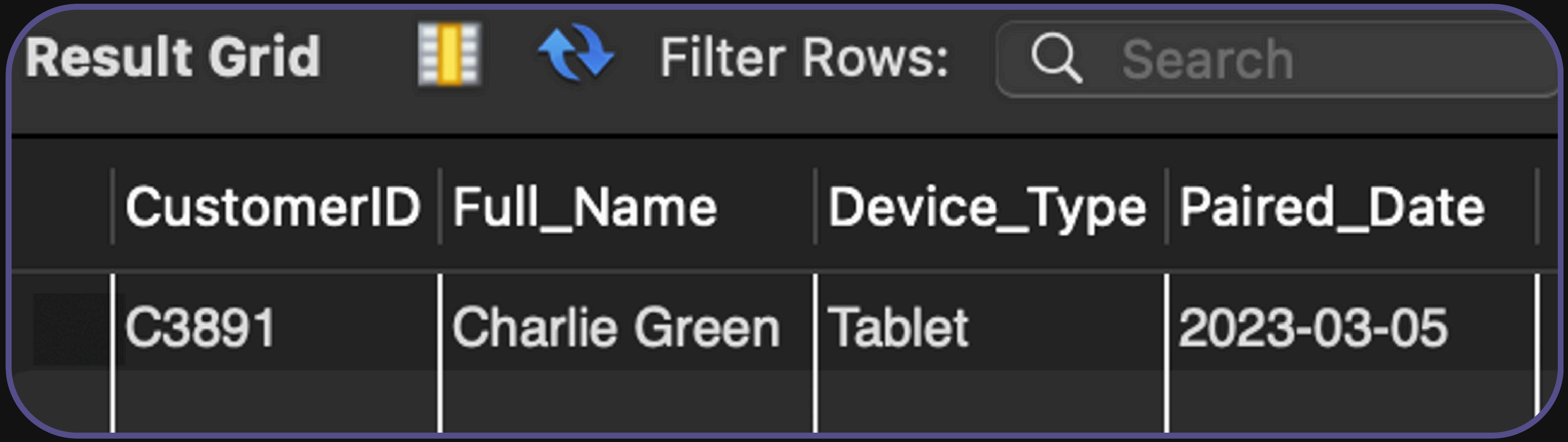
Potential Use Case 1

Customer Device

Customer Support

When customers contact support for issues related to device compatibility, playback errors, or connectivity, support representatives can use this query to identify the customer's devices. Knowing the device types allows support to provide targeted troubleshooting steps or even suggest upgrading the device for a better experience.

```
SELECT C.CustomerID, C.Full_Name, D.Device_Type,
CD.Paired_Date
FROM Customer C
INNER JOIN Customer_Device CD ON C.CustomerID =
CD.CustomerID
INNER JOIN Device D ON CD.DeviceID = D.DeviceID
WHERE C.CustomerID = 'C3891';
```



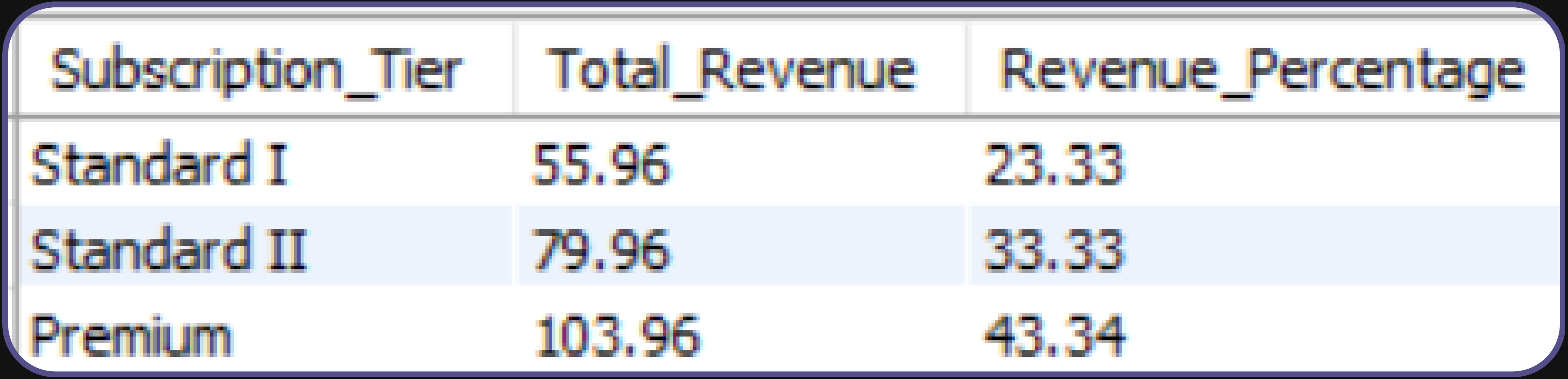
CustomerID	Full_Name	Device_Type	Paired_Date
C3891	Charlie Green	Tablet	2023-03-05

Potential Use Case 2

Revenue per Subscription

Finance

```
SELECT  S.Subscription_Tier, SUM(P.Amount) AS
Total_Revenue,
ROUND(SUM(P.Amount) / (SELECT SUM(Amount) FROM
Payment) * 100, 2) AS Revenue_Percentage
FROM Payment P
INNER JOIN Subscription S
ON P.SubscriptionID = S.SubscriptionID
GROUP BY  S.Subscription_Tier;
```



Subscription_Tier	Total_Revenue	Revenue_Percentage
Standard I	55.96	23.33
Standard II	79.96	33.33
Premium	103.96	43.34

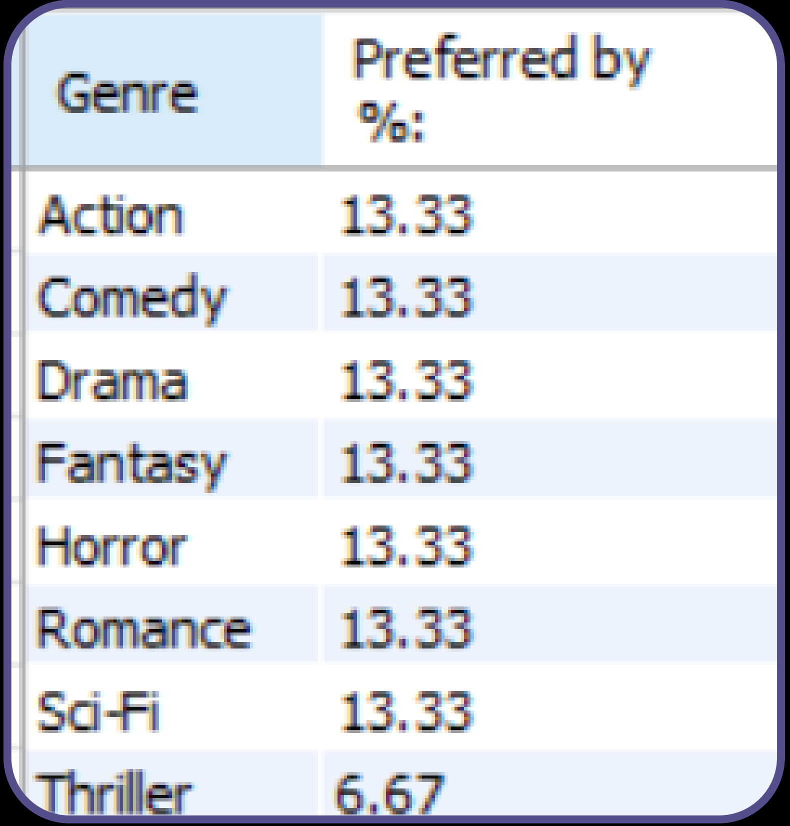
This query is useful for a quick, high-level view of revenue distribution across subscription tiers. The finance team can use this to understand which subscription tiers are bringing in the most revenue and identify any potential opportunities to adjust tier pricing or promote certain tiers.

Potential Use Case 3

Preferred Genres by Percentage Share

```
SELECT genre_name as Genre, ROUND(COUNT(*) / (SELECT COUNT(profileid) FROM
profile) * 100, 2) as 'Preferred by %:'
FROM preferred_genre pg inner join genre g on g.genreid = pg.genreid
GROUP BY Genre_name;
```

Netflix can benefit from identifying the most popular genres by using this data to personalize recommendations, guide content acquisition and production, and improve marketing efforts. By focusing on trending genres, Netflix can engage viewers more effectively, reduce churn, and ensure a more tailored user experience. Additionally, understanding genre popularity helps optimize content strategies, allocate budgets efficiently, and stay ahead of competitors, ultimately increasing user satisfaction and retention while driving growth.



Genre	Preferred by %:
Action	13.33
Comedy	13.33
Drama	13.33
Fantasy	13.33
Horror	13.33
Romance	13.33
Sci-Fi	13.33
Thriller	6.67

Potential Use Case 4