FINAL ASSIGNMENT

Information Systems and Databases
Dr. Idan Roth

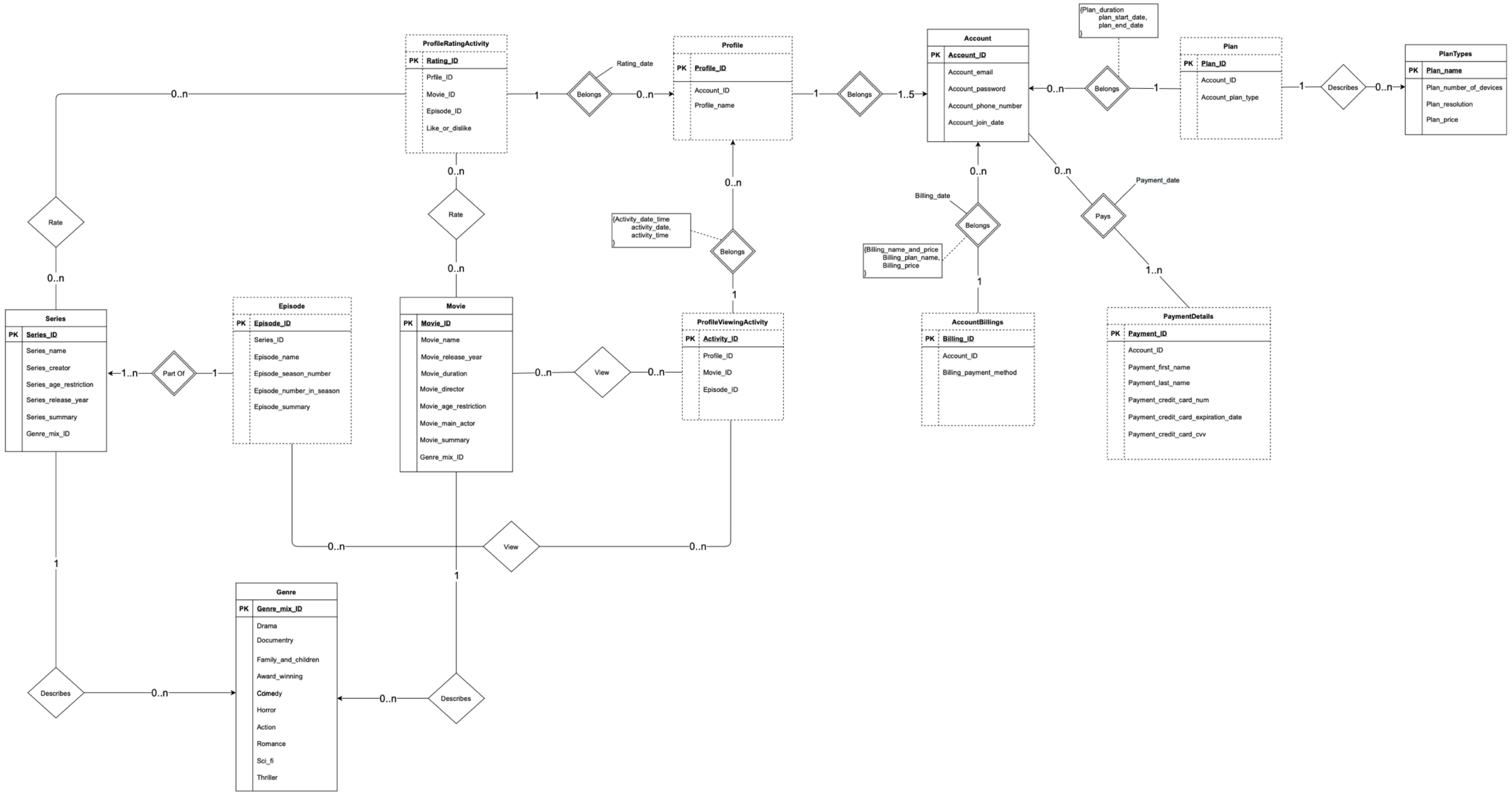
Netflix Inc.

Michael Rubinfeld – 208391367

Guy Gaash - 308577485

Q1

We chose Netflix Inc. as our case study company. Netflix Inc. is an American over-the-top content platform and production company The company's primary business is a subscription-based streaming service offering online streaming from a library of films and television series, including those produced in-house. Netflix meets the demand for online high quality content services which is available 24/7 on different devices with no commercials. In January 2021, Netflix reached 203.7 million subscribers, including 73 million in the United States and It's services available almost worldwide.



Q3

1. PlanTypes (plan_name, plan_num_of_devices, plan_resolution, plan_price)
   There are 3 different plan types:
   1. Basic
   2. Normal
   3. Premium

   Price is NIS per month.
2. Plan (plan_ID, account_ID(FK), account_plan_name(FK), plan_start_date, plan_end_date)
   In this table we put each plan every account acquires. also, every time the same account changes its plan, a tuple is put in this table in order to keep track of each account's plan history. We are interested in keeping track only on the date of the plan the account buys because we presume it cannot be changed within a month.
3. Account (account_ID, account_email, account_password, account_cellphone_num, account_join_date)

4. Profile (profile_ID, account_ID(FK), profile_name)
   All those attributes of the account and the profile are, of course, shown in the Netflix interface
5. PaymentDetailes (payment_ID, account_ID(FK), payment_first_name, payment_last_name, payment_credit_card_num, payment_credit_card_expiary_date, payment_credit_card_CVV, payment_date)

6. AccountBillings (billing_ID, account_ID(FK), billing_plan_name(FK), billing_price, billing_date, billing_payment_method)

- In order to simplify the billing process, we put the billing plan name and price as attributes here as well.
- Each tuple is a monthly billing of a specific account.

7. Movie (movie_ID, movie_name, movie_release_year, movie_duration, movie_director, movie_age_restriction, movie_main_actor, movie_summary, genre_mix_ID(FK))

8. Series (series_ID, series_name, series_creaot, series_release_year, series_age_restriction, series_summary, genre_mix_ID(FK))

- We have included mini-series as series with only season.

9. Episode (episode_ID, series_ID(FK), episode_name, season_number, episode_number_in_season, episode_summary)

10. Genre (genre_mix_ID, family_and_children, award_winning, comedy, crime, documentary, action, horror, romance, sci_fi, thriller, drama)

- Each attribute describing a genre is a BIT type attribute, marking for each movie or series it's genres.

- Each genre mix ID is a unique combination of 3 genres. We found that this is the most efficient and organized way to handle this.

IMPORTANT NOTE: in each of the following schemas we put an attribute for movie_ID and for series_ID/episode_ID. The reason is that every entry in the following schemas applies to either but not both at once so we put both attributes and, in each tuple, one of them will be NULL (we will put the integrity constraint in the SQL code).

11. ProfileViewingActivity (activity_ID, profile_ID(FK), movie_ID(FK), episode_ID(FK), activity_date, activity_time)

12. ProfileRatingActivity (rating_ID, profile_ID(FK), movie_ID(FK), episode_ID(FK), Like_or_dislike, rating_date)

Q4

```
CREATE DATABASE Netflix;
USE Netflix;


CREATE TABLE PlanTypes (
  plan_name VARCHAR(10) PRIMARY KEY,
  plan_num_of_devices INT,
  plan_resolution VARCHAR(30),
  plan_price FLOAT);

INSERT INTO PlanTypes VALUES ('Basic', 1, 'Standard Definition', 32.90);
INSERT INTO PlanTypes VALUES ('Standard', 2, 'FULL HD (1080p)', 46.90);
INSERT INTO PlanTypes VALUES ('Premium', 4, 'ULTRA HD (4K)', 60.90);


CREATE TABLE Account (
  account_ID INT PRIMARY KEY, /*start with 00*/
  account_Email VARCHAR(50),
  account_password VARCHAR(20),
  account_cellphone_num VARCHAR(15),
  account_join_date DATE);

INSERT INTO Account VALUES (00111, 'kelly.slater@gmail.com', 'ke11yslaten', '0539281922', '2020-01-02');
INSERT INTO Account VALUES (00112, 'rob.machado@walla.com', 'puravidarob', '0528391068', '2020-09-05');
INSERT INTO Account VALUES (00113, 'andy.irons@gmail.com', 'andyforthewin', '0591882714', '2020-02-01');
INSERT INTO Account VALUES (00114, 'parko@hotmail.com', 'isallgoodmate', '051662733', '2020-05-12');

INSERT INTO Account VALUES (00115, 'kelly.slater@gmail.com', 'ke11yslaten', '0539281922', '2020-07-02');
INSERT INTO Account VALUES (00116, 'rob.machado@walla.com', 'puravidarob', '0528391068', '2020-07-05');
INSERT INTO Account VALUES (00117, 'andy.irons@gmail.com', 'andyforthewin', '0591882714', '2020-07-01');
INSERT INTO Account VALUES (00118, 'parko@hotmail.com', 'isallgoodmate', '051662733', '2020-07-12');

CREATE TABLE Plan (
  Plan_ID INT PRIMARY KEY, /*start with 01*/
  account_ID INT,
  account_plan_name VARCHAR(10),
  plan_start_date DATE,
  plan_end_date DATE,
  FOREIGN KEY (account_ID) REFERENCES Account(account_ID),
  FOREIGN KEY (account_plan_name) REFERENCES PlanTypes(plan_name));
```

```
    INSERT INTO Plan VALUES (01111, 00111, 'Premium', '2019-02-01', '2019-03-
01');
    INSERT INTO Plan VALUES (01112, 00112, 'Standard', '2020-02-02', '2020-04-
02');
    INSERT INTO Plan VALUES (01113, 00113, 'Standard', '2020-10-30', '2020-04-
02');
    INSERT INTO Plan VALUES (01114, 00114, 'Basic', '2019-06-30', '2019-07-30');

    INSERT INTO Plan VALUES (01115, 00115, 'Premium', '2020-07-02', '2021-10-
01');
    INSERT INTO Plan VALUES (01116, 00116, 'Standard', '2020-07-05', '2020-01-
09');
    INSERT INTO Plan VALUES (01117, 00117, 'Standard', '2020-07-01', '2021-04-
02');
    INSERT INTO Plan VALUES (01118, 00118, 'Basic', '2019-07-12', '2020-07-30');


    CREATE TABLE Profile (
      Profile_ID INT PRIMARY KEY, /*start with 02*/
      account_ID INT,
      profile_name VARCHAR(30),
      FOREIGN KEY (account_ID) REFERENCES Account(account_ID));

    INSERT INTO Profile VALUES (02111, 00111, 'KSprofile');
    INSERT INTO Profile VALUES (02112, 00112, 'RMprofile');
    INSERT INTO Profile VALUES (02113, 00113, 'AIprofile');
    INSERT INTO Profile VALUES (02114, 00114, 'JPprofile');


    CREATE TABLE PaymentDetailes (
      payment_ID INT PRIMARY KEY, /*start with 03*/
      account_ID INT,
      payment_first_name VARCHAR(20),
      payment_last_name VARCHAR(30),
      payment_credit_card_num INT,
      payment_credit_card_expiary_date DATE,
      payment_credit_card_CVV INT,
      payment_date DATE,
      FOREIGN KEY (account_ID) REFERENCES Account(account_ID));

    INSERT INTO PaymentDetailes VALUES (03111, 00111, 'Kelly', 'Slater',
53278192, '2021-11-19', 716, '2020-11-11');
    INSERT INTO PaymentDetailes VALUES (03112, 00112, 'Rob', 'Machado',
10935277, '2021-08-12', 912, '2020-10-9');
    INSERT INTO PaymentDetailes VALUES (03113, 00113, 'Andy', 'Irons',
118266354, '2020-12-30', 788, '2021-11-4');
    INSERT INTO PaymentDetailes VALUES (03114, 00114, 'Joel', 'Parkinson',
40930029, '2021-11-15', 717, '2020-12-12');
```

```
    CREATE TABLE AccountBillings (
      billing_ID INT PRIMARY KEY, /*start with 04*/
      account_ID INT,
      billing_plan_name VARCHAR(10),
      billing_price FLOAT,
      billing_date DATE,
      billing_payment_method VARCHAR(10),
      FOREIGN KEY (account_ID) REFERENCES Account (account_ID),
      FOREIGN KEY (billing_plan_name) REFERENCES PlanTypes(plan_name));

    INSERT INTO AccountBillings VALUES (04116, 00113, 'Basic', 46.90, '2020-01-
25', 'Visa');
    INSERT INTO AccountBillings VALUES (04117, 00114, 'Basic', 60.90, '2020-01-
25', 'Visa');
    INSERT INTO AccountBillings VALUES (04118, 00111, 'Premium', 60.90, '2020-
01-22', 'Visa');
    INSERT INTO AccountBillings VALUES (04119, 00112, 'Standard', 60.90, '2020-
01-23', 'PayPal');

    INSERT INTO AccountBillings VALUES (04111, 00111, 'Premium', 60.90, '2021-
02-22', 'Visa');
    INSERT INTO AccountBillings VALUES (04112, 00112, 'Standard', 46.90, '2020-
02-23', 'PayPal');
    INSERT INTO AccountBillings VALUES (04113, 00113, 'Standard', 46.90, '2020-
02-24', 'Visa');
    INSERT INTO AccountBillings VALUES (04114, 00114, 'Basic', 32.90, '2020-02-
25', 'Visa');
    INSERT INTO AccountBillings VALUES (04115, 00114, 'Basic', 32.92, '2020-02-
25', 'Visa');


    CREATE TABLE Genre (
      genre_mix_ID INT PRIMARY KEY, /*start with 08*/
      family_and_children BIT,
      award_winning BIT,
      comedy BIT,
      crime BIT,
      documentary BIT,
      action BIT,
      horror BIT,
      romance BIT,
      sci_fi BIT,
      thriller BIT,
      drama BIT);

    INSERT INTO Genre VALUES (08111, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0);
    INSERT INTO Genre VALUES (08112, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0);
    INSERT INTO Genre VALUES (08113, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1);
    INSERT INTO Genre VALUES (08114, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0);
    CREATE TABLE Movie (
      movie_ID INT PRIMARY KEY,  /*start with 05*/
      movie_name VARCHAR(100),
      movie_release_year INT,
      movie_duration TIME,
      movie_director VARCHAR(50),
      movie_age_restriction VARCHAR(8),
      movie_main_actor VARCHAR(50),
      movie_summary VARCHAR(500),
       genre_mix_ID INT,
       FOREIGN KEY (genre_mix_ID) REFERENCES Genre(genre_mix_ID));

    INSERT INTO Movie VALUES (05111, 'Life of Brian', 1979, '01:33:00', 'Terry
Jones', 'E', 'John Cleese', 'Life of Jesus Christ with a twist', 08111);
    INSERT INTO Movie VALUES (05112, 'The Departed', 2006 , '02:31:00', 'Martin
Scorsese', 'R', 'Jack Nicholson', 'An undercover agent and a spy constantly try to
counter-attack each other in order to save themselves from being exposed in front of
the authorities. Meanwhile, both try to infiltrate an Irish gang.', 08112);
    INSERT INTO Movie VALUES (05113, 'Scott Pilgrim vs. The World', 2010 ,
'01:52:00', 'Edgar Wright', 'PG-13', 'Michael Cera', 'Scott Pilgrim meets Ramona and
instantly falls in love with her. But when he meets one of her exes at a band
competition, he realizes that he has to deal with all seven of her exes.', 08111);
    INSERT INTO Movie VALUES (05114, 'Inglorious Bastards', 2009 , '02:33:00',
'Quentin Tarantino', 'R', 'Brad Pitt', 'A few Jewish soldiers are on an undercover
mission to bring down the Nazi government and put an end to the war. Meanwhile, a
woman wants to avenge the death of her family from a German officer.',08112);


    CREATE TABLE Series (
      series_ID INT PRIMARY KEY, /*start with 06*/
      series_name VARCHAR(100),
      series_creator VARCHAR(100),
      series_release_year INT,
      series_age_restriction VARCHAR(8),
      series_summary VARCHAR(500),
       genre_mix_ID INT,
       FOREIGN KEY (genre_mix_ID) REFERENCES Genre(genre_mix_ID));


    INSERT INTO Series VALUES (06111, 'How I Met Your Mother', 'Carter Bays',
2005, 'PG-13', 'Ted Mosby recounts to his children the events that led him to meet
their mother', 08113);
    INSERT INTO Series VALUES (06112, 'Breaking Bad', 'Vince Gilligan', 2008, 'R',
'A chemistry teacher, discovers that he has cancer and decides to get into the meth-
making business to repay his medical debts', 08114);
    INSERT INTO Series VALUES (06113, 'Community', 'Dan Harmon', 2009, 'PG-
13', 'Students of diverse temperaments form a study group which leads to quirky and
memorable encounters', 08113);
```

```
    INSERT INTO Series VALUES (06114, 'Family Guy', 'Seth MacFarlane', 1999,
'R', 'Peter Griffin and his family of two teenagers, a smart dog, a devilish baby and
his wife find themselves in hilarious situations', 08114);


    CREATE TABLE Episode (
      episode_ID INT PRIMARY KEY, /*start with 07*/
      series_ID INT,
      episode_name VARCHAR(50),
      season_number INT,
      episode_number_in_season INT,
      episode_summary VARCHAR(500),
      FOREIGN KEY (series_ID) REFERENCES Series(series_ID));

    INSERT INTO Episode VALUES (07111, 06111, 'The Playbook', 5, 8, "Barney
reverts back to his bachelor ways when he and Robin break up and is soon on the
prowl looking for new women using the Playbook");
    INSERT INTO Episode VALUES (07112, 06112, 'Felina', 5, 16, "Walter makes
one last attempt to secure his family's future, while also visiting some old enemies,
during his final return to Albuquerque");
    INSERT INTO Episode VALUES (07113, 06113, 'remedial chaos theory', 3, 4,
"When Jeff introduces an element of randomness into Troy and Abed's
housewarming party, reality splits into six timelines affecting the group's
relationships.");
    INSERT INTO Episode VALUES (07114, 06114, 'Road to the Multiverse', 8, 1,
"Brian and Stewie go on another road trip through alternate universes using a special
remote control. The trip includes a post-apocalyptic world, a parallel world run by
dogs where humans are pets, and more.");


    CREATE TABLE ProfileViewingActivity (
      activity_ID INT PRIMARY KEY, /*start with 09*/
      profile_ID INT,
      movie_ID INT,
      episode_ID INT,
      activity_date DATE,
      activity_time TIME,
      CONSTRAINT CHECK ((movie_ID IS NOT NULL AND episode_ID IS NULL)
OR
      (movie_ID IS NULL OR episode_ID IS NOT NULL)),
       /*We assume that a tuple will never contain both movie_ID and episode_ID -
one of them will always be null*/
       FOREIGN KEY (profile_ID) REFERENCES Profile(profile_ID),
      FOREIGN KEY (movie_ID) REFERENCES Movie(movie_ID),
      FOREIGN KEY (episode_ID) REFERENCES Episode(episode_ID));

    INSERT INTO ProfileViewingActivity VALUES (09111, 02111, 05111, null, '2021-
01-11', '02:33:40');
    INSERT INTO ProfileViewingActivity VALUES (09112, 02112, 05112, null, '2021-
02-12', '02:35:27');
```

```
    INSERT INTO ProfileViewingActivity VALUES (09113, 02113, 05113, null, '2021-
03-11', '02:38:40');
    INSERT INTO ProfileViewingActivity VALUES (09118, 02113, 05113, null, '2021-
03-11', '05:38:40');
    INSERT INTO ProfileViewingActivity VALUES (09114, 02114, 05114, null, '2021-
01-11', '02:39:01');
    INSERT INTO ProfileViewingActivity VALUES (09119, 02111, 05114, null, '2021-
01-11', '02:39:01');
    INSERT INTO ProfileViewingActivity VALUES (09120, 02112, 05114, null, '2021-
01-11', '02:39:01');
    INSERT INTO ProfileViewingActivity VALUES (09117, 02114, 05114, null, '2021-
01-11', '03:39:01');
    INSERT INTO ProfileViewingActivity VALUES (09115, 02113, null, 07111, '2021-
03-11', '03:38:40');
    INSERT INTO ProfileViewingActivity VALUES (09116, 02114, null, 07112, '2021-
01-11', '03:39:01');

    CREATE TABLE ProfileRatingActivity (
      rating_ID INT PRIMARY KEY, /*start with 10*/
      profile_ID INT,
      movie_ID INT,
      episode_ID INT,
       Like_or_dislike BIT,
      rating_date DATE,
      CONSTRAINT CHECK ((movie_ID IS NOT NULL AND episode_ID IS NULL)
OR
      (movie_ID IS NULL OR episode_ID IS NOT NULL)),
       /*We assume that a tuple will never contain both movie_ID and episode_ID -
one of them will always be null*/
       FOREIGN KEY (profile_ID) REFERENCES Profile(profile_ID),
      FOREIGN KEY (movie_ID) REFERENCES Movie(movie_ID),
      FOREIGN KEY (episode_ID) REFERENCES Episode(episode_ID));

    INSERT INTO ProfileRatingActivity VALUES (10111, 02111, 05111, null,0, '2021-
01-11');
    INSERT INTO ProfileRatingActivity VALUES (10112, 02111, 05111, null,1, '2021-
02-12');
    INSERT INTO ProfileRatingActivity VALUES (10113, 02111, null, 07111,1, '2021-
01-16');
INSERT INTO ProfileRatingActivity VALUES (10114, 02111, null, 07111,0, '2021-
                                                                        06-11');
```

Q5

1. PaymentDetailes (payment_ID, account_ID(FK), payment_first_name, payment_last_name, payment_credit_card_num, payment_credit_card_expiary_date, payment_credit_card_CVV, payment_date)

All the attributes are atomic and unique but we have functional dependency in an attribute that is not part of the primary key (e.g. payment_credit_card_num, payment_first_name) thus we are in 2NF.

- AccountOfPaymentEntry (payment_ID, account_ID(FK))
- PaymenDetails (payment_ID, payment_first_name, payment_last_name, payment_credit_card_expiary_date, payment_credit_card_CVV, payment_date)
- PaymentCreditCard (payment_ID, payment_credit_card_num)

2. Movie (movie_ID, movie_name, movie_release_year, movie_duration, movie_director, movie_age_restriction, movie_main_actor, movie_summary, genre_mix_ID(FK))

All the attributes are atomic and unique but we have functional dependency in an attribute that is not part of the primary key (e.g. movie_name, movie_duration) thus we are in 2NF.

- MovieName (movie_ID, movie_name)
- MovieDetails (movie_ID, movie_release_year, movie_duration, movie_director, movie_age_restriction, movie_main_actor, movie_summary, genre_mix_ID(FK))
  [We assume that you won't be able to deduct from the movie summary any other details for 100%]

3. Genre (genre_mix_ID, family_and_children, award_winning, comedy, crime, documentary, action, horror, romance, sci_fi, thriller, drama)

There is no attribute in the relation that is dependent and is not a possible key there for this relation in BCNF.

Q6

KPI 1-3: https://zemoga.com/blog/post/ott-app-metrics

KPI 4: https://recurly.com/content/key-metrics-for-subscription-commerce/

KPI 5: https://www.techsmith.com/blog/video-metrics-essentials/

Q7

7.1. + 7.2.

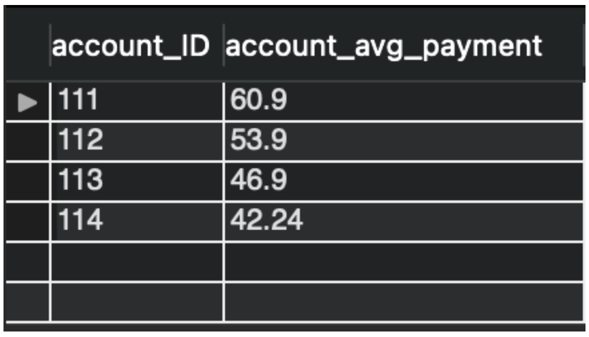
1. Average Revenue Per User (ARPU)

Average Revenue Per User (ARPU) measures the amount of money you can expect to generate from one customer. It's calculated by dividing the total revenue of your OTT business by its total number of users.

KPI 1 SQL QUERY:

```
SELECT account_ID, ROUND(AVG(billing_price), 2) AS account_avg_payment
FROM AccountBillings

GROUP BY account_ID;
```



account_ID	account_avg_payment
111	60.9
112	53.9
113	46.9
114	42.24

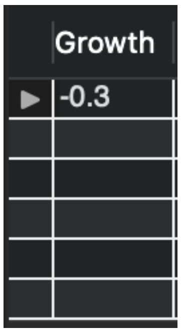
2. **Revenue Growth Rate**

This KPI helps to ensure your business continues to grow at a target rate, measured by a percentage. Ideally you would measure this monthly or on a 12 month rolling average basis.

KPI 2 SQL QUERY:

```
WITH january_revenue (jan_rev) AS
   (SELECT SUM(billing_price)
    FROM AccountBillings
    WHERE billing_date >= '2020-01-01' AND billing_date <= '2020-01-31'),
   february_revenue (feb_rev) AS
   (SELECT SUM(billing_price)
    FROM AccountBillings
    WHERE billing_date >= '2020-02-01' AND billing_date <= '2020-02-28')
SELECT ROUND(((february_revenue.feb_rev - january_revenue.jan_rev) /
january_revenue.jan_rev), 2) AS Growth

FROM february_revenue, january_revenue;
```



Growth
-0.3

3. **Engagement Rate**

Engagement measures user interaction with your website or apps. Common metrics can include how many times a user opens an app, how many videos they're watching per session, and what actions a user takes within one of your digital properties.

Your business won't be sustainable if it depends on users taking just a single action. Your users must continue to make moves over sustained sessions.

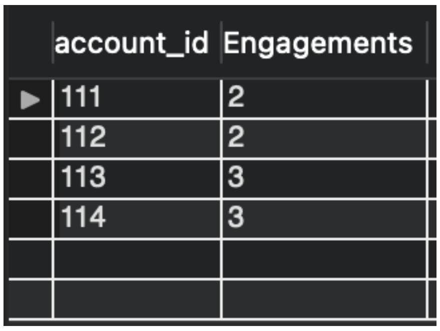
Measuring engagement and user behavior allows you to obtain a better understanding of your customer relationships. Many businesses follow the 80/20 rule: engage 20% of your customers to generate 80% of your business revenue.

https://zemoga.com/blog/post/ott-app-metrics

KPI 3 SQL QUERY:

```
WITH profile_in_account (account_id, profile_id) AS
    (SELECT account_ID, profile_ID
    FROM Profile),
   profile_activity (profile_id, activity_date) AS
   (SELECT profile_ID, activity_date
    FROM ProfileViewingActivity)
SELECT account_id, COUNT(activity_date) AS Engagements
FROM profile_in_account NATURAL JOIN profile_activity
WHERE activity_date >= '2021-01-01' AND activity_date <= '2021-03-31'

GROUP BY account_id;
```



account_id	Engagements
111	2
112	2
113	3
114	3

4. **Trail Conversion Rate**

$$\frac{(Subscription\ trails\ start\ in\ month\ Z\ that\ convert\ to\ paying\ customers)}{Subscription\ trails\ started\ in\ month\ Z}$$

Example how to check the trail conversion rate of accounts who joined on July 2020:

The trail period is 30 exactly days, thus their trail will end between 1.8.2020 and 31.8.2020. Those who have a paid plan on their name which ends after 1.9.2020 are converted.

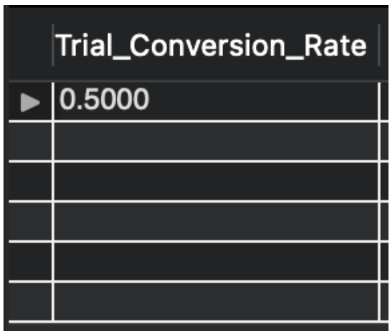
We will compute the trail conversion rate by counting accounts joined within the last 30 days and started paying after these 30 days.

https://recurly.com/content/key-metrics-for-subscription-commerce/

KPI 4 SQL QUERY:

```
WITH converted_accounts (amount) AS
  (SELECT COUNT(account_ID)
  FROM Account
  WHERE account_join_date >= '2020-07-01' AND account_join_date <= '2020-07-31' AND
  account_ID IN (SELECT account_ID FROM Plan WHERE plan_end_date >= '2020-09-01'))
SELECT amount / (SELECT COUNT(account_ID) FROM Account WHERE account_join_date >= '2020-07-01' AND
account_join_date <= '2020-07-31') AS Trial_Conversion_Rate

FROM converted_accounts;
```



Trial_Conversion_Rate
0.5000

5. Unique plays

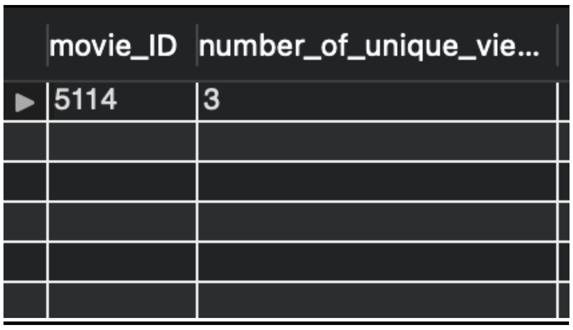
A Play counts every time the video play button is pushed, including Plays by the same person. Unique Plays filters out repeat video playbacks by the same person. Meaning, if I play a video twice, the Play amount is two, and the Unique Play amount is one.

https://www.techsmith.com/blog/video-metrics-essentials/

KPI 5 SQL QUERY:

```
WITH unique_profiles_and_movies (movie_ID, number_of_unique_views) AS
  (SELECT movie_ID, COUNT(DISTINCT profile_ID)
   FROM ProfileViewingActivity
   WHERE movie_ID IS NOT null
   GROUP BY movie_ID)
SELECT movie_ID, number_of_unique_views
FROM unique_profiles_and_movies
```

```
    WHERE number_of_unique_views >= ALL (SELECT number_of_unique_views
                                                                  FROM
unique_profiles_and_movies);
```



movie_ID	number_of_unique_vie...
5114	3

Q8

Note -

- We did some adaptations and specifications of our KPIs to make them interesting and more adaptable in RA form.
- $\mathcal{G}$ marks an aggregatory function.

1. Average Revenue Per User (ARPU) in 2020

$$\Pi_{account\ ID}\ \mathcal{G}_{\ avg(billing\ price)}(\sigma_{1.1.2020\ \le\ billing\ date\ \wedge\ \text{billing date} \le 31.12.2020}(Account\ Billings))$$

For us to build our business model we need to know our revenue stream (on average) for every new user comes to our platform.

2. Engagement Rate - Relational Algebra case - Accounts Engagement this Month

$$\Pi_{Account.accunt\ ID}((\sigma_{\text{activity date} \ge 1.1.2021\ \wedge\ \text{activity date} \le 31.3.2021}(\text{ProfileViewingActivity}) \bowtie_{profile.profile\ ID=\text{ProfileViewingActivity.profile ID}}\ Profile) \bowtie_{Account.account\ ID=profile.account\ ID}\ Account)$$

We chose engagement rate because the service is streaming and one of the best ways to know how valuable is your product is to measure how often users use it. We wanted to check the engagement rate by account per month (we can do it per any period of time) and not only by profile because we measure our revenue stream depends on the number of accounts and not the number of profiles.

3. Trial Conversion in a Specific Month (we can't calculate the percentage rate in RA)

Note - We assume we do this KPI check on September 1st .

$$\Pi_{account\ ID}\left(\sigma_{join\ date\ \ge\ 1.7.2020\ \wedge\ \text{join date} \le 31.7.2020}(Account)\right) \cap \Pi_{account\ ID}\left(\sigma_{end\ date\ \ge\ 1.9.2020}(Plan)\right)$$

Trial conversion is very important because it measures how good are we in acquiring new paying customers which will create us a revenue stream and will enable our business to keep operating.

Q9

PlanTypes		
Attribue	Data Type	SIZE
plan_name	CHAR (10)	10
plan_num_of_devices	INT	4
plan_resolution	CHAR (30)	30
plan_price	FLOAT	8
		SUM = 52
Account		
Attribue	Data Type	SIZE
account_ID	INT	4
account_Email	CHAR (50)	50
account_password	CHAR (20)	20
account_cellphone_num	CHAR (15)	15
account_join_date	DATE	3
		SUM = 89
Plan		
Attribue	Data Type	SIZE
plan_ID	INT	4
account_ID	INT	4
account_plan_name	CHAR(10)	10
plan_start_date	DATE	3
plan_end_date	DATE	3
		SUM = 24
Profile		
Attribue	Data Type	SIZE
Profile_ID	INT	4
Account_ID	INT	4
Profile_name	CHAR(30)	30
		SUM = 38
Payment Details		
Attribue	Data Type	SIZE
payment_ID	INT	4
Account_ID	INT	4
Payment_first_name	CHAR(20)	20
Payment_last_name	CHAR(30)	30
Payment_credit_card_num	INT	4
Payment_credit_card_expiary_date	DATE	3
Payment_credit_card_CVV	INT	4
Payment _date	DATE	3
		SUM = 72
Acccount Billings		
Attribue	Data Type	SIZE
Billing_ID	INT	4
Account_ID	INT	4
Billing_plan_name	CHAR(10)	10

Billing_price	FLOAT	4
Billing_date	DATE	3
Billingpayment_method	CHAR(10)	10
		SUM = 35
Movie		
Attribue	Data Type	SIZE
Movie_ID	INT	4
Movie_name	CHAR(100)	100
Movie_release_year	INT	4
Movie_duration	TIME	3
Movie_director	CHAR(50)	50
Movie_age_restriction	CHAR(8)	8
Movie_main_actor	CHAR(50)	50
Movie_summary	CHAR(500)	500
Genre_mix_ID	INT	4
		SUM = 723
Series		
Attribue	Data Type	SIZE
Series _ID	INT	4
Series _name	CHAR(100)	100
Series_release_year	INT	4
Series_creator	CHAR(100)	100
Series_age_restriction	CHAR(8)	8
Series_summary	CHAR(500)	500
Genre_mix_ID	INT	4
		SUM = 720
Episode		
Attribue	Data Type	SIZE
Episode_ID	INT	4
Series_ID	INT	4
Episode_name	CHAR(50)	50
Season_number	INT	4
Episode_number_in_season	INT	4
Episode_summary	CHAR(500)	500
		SUM = 566
Genre		
Attribue	Data Type	SIZE
Genre_mix_ID	INT	4
Family_and_children	BIT	1
Award_winning	BIT	1
Comedy	BIT	1
Crime	BIT	1
Documentry	BIT	1
Action	BIT	1
Horror	BIT	1
Romance	BIT	1
Sci_Fi	BIT	1
Thriller	BIT	1

Drama	BIT	1
		SUM = 15
Profile Viewing Activity		
Attribue	Data Type	SIZE
Activity_ID	INT	4
Profile_ID	INT	4
Movie_ID	INT - Can’t Be Both	4
Episode_ID		
Activity_date	DATE	3
Activity_time	TIME	3
		SUM = 18
Profile Rating Activity		
Attribue	Data Type	SIZE
Rating_ID	INT	4
Profile_ID	INT	4
Movie_ID	INT - Can’t Be Both	4
Episode_ID		
Like_or_dislike	BIT	1
Rating _date	DATE	3
		SUM = 16

Assumptions: Based as much as possible on real Netflix data.

- Different subscribers (accounts) = 200,000,000
- Average number of profiles per account: 3
- Number of movies = 3000
- Number of shows = 2000
- Average number of episodes per show = 30 (Thus number of episodes on Netflix = 60,000)
- Billing happens once in a month for all plans.
- Payment details are stored for a year (for siplifications)
- A person bothers to rate one tenth of everything he watches.
- There are 3 different plans: Basic, Standard, Premium.

- $Plan\ Type\ Storage\ Size:$
  $(number\ of\ plan\ types * Size\ of\ a\ type\ tuple) =$
  $3 * 52 = 156\ Bytes$

- $Accounts\ Storage\ Size:$
  $(number\ of\ users * size\ of\ a\ user\ tuple) =$
  $200{,}000{,}000 * 89 = 17{,}800{,}000{,}000\ Bytes$

- $Plan\ Storage\ Size:$
  $(changes * plan\ tuple\ size * numer\ of\ users) =$
  $1.5 * 24 * 200{,}000{,}000 = 7{,}200{,}000{,}000\ Bytes$
    - $We\ assume\ that\ half\ of\ the\ users\ will\ change\ their\ plan\ once$

- $Profiles\ Storage\ Size$:
  $(average\ number\ of\ profiles * number\ of\ users * size\ of\ a\ profile\ tuple) =$
  $3 * 200{,}000{,}000 * 38 = 22{,}800{,}000{,}000\ Bytes$

- $Payment\ Details\ Storage\ Size$:
  $(changes * number\ of\ users * size\ of\ a\ payment\ details\ tuple)$
  $1.5 * 200{,}000{,}000 * 79 = 23{,}700{,}000{,}000\ Bytes$
  - $We\ assume\ that\ half\ of\ the\ users\ will\ change\ their\ payment\ details\ once$

- $Account\ Billings\ Storage\ Size$:
  $(number\ of\ users * size\ of\ an\ account\ billing\ tuple * 12\ month\ a\ year) =$
  $200{,}000{,}000 * 35 * 12 = 8{,}400{,}000{,}000\ Bytes$

- $Movies\ Storage\ Size: (size\ of\ a\ movie\ tuple * number\ of\ movies) =$
  $723 * 3000 = 2{,}169{,}000\ Bytes$

- $Series\ Storage\ Size: (number\ of\ series * size\ of\ a\ series\ tuple) =$
  $2000 * 720 = 1{,}440{,}000\ Bytes$

- $Episodes\ Storage\ Size$:
  $(number\ of\ series * average\ number\ of\ episodes * size\ an\ episode\ tuple) =$
  $2000 * 30 * 566 = 33{,}960{,}000\ Bytes$

- $Genre\ Storage\ Size$:
  $(number\ of\ pssible\ combination * size\ of\ a\ genre\ tuple) =$
  $\binom{11}{3} * 15 = 2{,}475\ Bytes$

- $Profile\ Viewing\ Activity\ Storage\ Size$:
  $(avrage\ views\ per\ month * 12\ months * size\ of\ a\ tuple) =$
  $18{,}000{,}000 * 12 * 18 = 3{,}888{,}000{,}000\ bytes$

- $Profile\ Rating\ Activity\ Storage\ Size$:
  $(avrage\ ratings\ per\ month * 12\ months * size\ of\ a\ tuple) =$
  $1{,}800{,}000 * 12 * 16 = 345{,}600{,}000\ bytes$

$$\boldsymbol{8.417 * 10^{10}\ Bytes \sim 84.17\ GB}$$

Q10

Although “Genre” has the largest amount of attributes, we chose to take “Movie” because it’s a much more diversed relation so there will be more features to analyze.

Movie (movie_ID, movie_name, movie_release_year, movie_duration, movie_director, movie_age_restriction, movie_main_actor, movie_summary, genre_mix_ID)

We assume there are 3,000,000 tuples and it’s indexed by index sequential file.

10.1.

Because the Movie relation contains big amount of tuples, it is possible to consider making another index to make the search more efficient even though we already are in index sequential file.

10.2.

To serve our users in the best way, we would want to reduce time for searching a specific movie. Usually when a user wants to find the best movie that would fit his mood he or she would want to filter movies by genre. To make the search by genre more efficient we can add a new index by genre.

10.3.

ALTER TABLE Movie ADD INDEX(genre_mix_ID);

10.4.

If there is no indexing, in order to find a specific entry we would have perform a linear search on all the tuples which can be between 1 to 3,000,000 retrieves with time complexity of 5 to 15,000,000,000 ms.
If the file is indexed by the primary key the maximum number of retrieves from the memory is $\lceil \log_2(3{,}000{,}000) \rceil = 22$, so the maximum time will be 22*5 = 110ms.
Therefor using index sequential file is better search method.

10.5.

The indexing type we would recommend will be hashing. We chose hashing because it’s complexity time is better than b+-tree (O(1) instead of log(n)). We assume that the primary keys are generated randomly and do not represent any kind of range therefore hashing will be more effective than b+-tree indexing.

When comparing the hash indexing to sequential indexing, the difference between them is that the Hash file saves data on hash algorithm and on a hash key value, while sequential file doesn’t have any key value to save the data. Basis on this hash key feature, searching in Hash file is faster than in sequential file.

10.6.

If we want to optimize indexing in all table, some tables will not benefit from using Hash index. For example, ‘Profile Viewing Activity’ and ‘Profile Rating Activity are two relations which in both we have regular and chronological data entries therefore we would want to use b+-tree indexing because the chronological feature of the primary keys will make a range of number instead of a random set of numbers.