A database for a music streaming service

Leonardo Masci

June 20, 2023

1 Functional analysis

1.1 Requirements

From the functional requirement is possible to classify the expected features of the Database into the following items:

1. The music streaming platform has to provide contents such as:
   - albums and singles (of different production companies)
   - podcast
   - audio-books
   - radio
   - live-streaming
   - user created audio-tracks
2. Registration of customers is allowed by providing:
   - full name
   - address
   - e-mail
3. The system needs to keep track of:
   - payment methods
   - content that users listen to
   - when a content is listen
   - from which device a content was heard
4. customers (or companies) that are owners of contents, have to set the price for them; a content can be accessed by a one-time individual fee for free or subscribing a package (until the subscription expires)
5. users of the platform can rate contents (already listen) from 1 to 5
6. customers can create playlists and decide to make them public or private
7. the platform has to be able to recommend new content to customer based on:
   - genre of already listen contents
   - the current time of the day
   - the age, gender and language of each customer[1]

[1] these are not required attributes for profiling strategy

1.2 Use cases

1. Users can list the audio contents of a given type they like the most, by setting a minimum rating.
2. User can visualize new content on the platform, given some preferences like type and/or genre.
3. Administrators of the service can list the email addresses of users who don't renew their subscriptions, along with which types of subscriptions were previously owned by them.
4. The service can recommend content that users may like based on their profile.

2 ER Diagram

For the Entity-Relation Diagram design is used Chen's notation; the procedure is composed of the following steps:

2.1 Identify the entity types

The entity types identified are:

1. customer (req. 2)
2. content (req. 1)
3. subscription_plan (req. 4)
4. playlist (req. 6)

2.2 Identify the attributes of each entity type

The attributes of above entities are respectively:

1. - fullname
   - email (unique key)
   - address
   - age
   - gender (male or female)
   - language

   A customer can register to the platform giving at least his/her fullname, address and email (that has to be unique in the system) (req.s 2 + 7)

2. - name(unique key)
   - genre (pop, rock, metal, indie, rap, jazz, blues, techno)
   - price
   - owner(unique key)
   - type (album, single, podcast, audio-books, radio, live-streaming, audio-track)
   - production_company
   - upload_date
   - owner_type (private or company)

   Contents needs to have at least a (unique) name, genre among the proposed, an upload date and a corresponding owner. It can have a price (if is 0 or missing is a free content) and a production company. (req.s 2 + 4)

3. - name(unique key)
   - type (daily, weekly, monthly, yearly)

- price
- end_date

A subscription type has a (unique) name, a type based on the duration (but is possible to create features based on the production company of each content). They can have a price (if is 0 or missing is a free plan) and an expiration date (until a plan can be subscribed). (req 4)

4.
   - name(unique key)
   - type (public or private)
   - owner(unique_key)

   A playlist must have a (unique) name and an owner (that can choose to make it public or private). (req. 6)

2.3 Define the relationship types

The relations created among the entities are:

1. subscription: customer N → 1 subscription_plan (req. 4)
2. rate: customer M → N content (req. 5)
3. listen: customer M → N content (req.s 3 + 5 + 7)
4. part_of: content M → N subscription_plan (req. 4)
5. part_of: content M → N playlist (req. 6)

2.4 ER Diagram (First attempt)

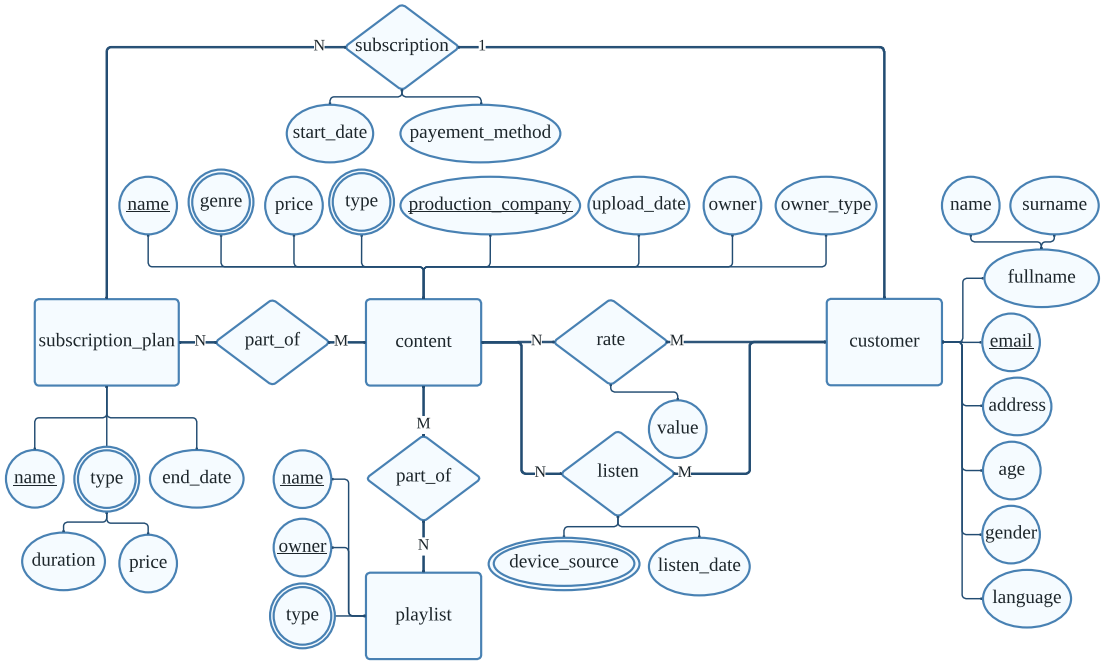


Figure 1: This Entity-Relation Diagram represent the conceptual schema of the Database; The notation used corresponds to: rectangles for Strong Entity types, rhombus for the Relationship types, ellipses for the attributes, underlined attributes are the unique identifiers of the entities, double ellipses for multivalued attributes. The cardinality of relations is reported on the edges of rhombus.

2.5 Check your design against the requirements and the use cases

Doing a review of the conceptual schema, is possible to notice that modelling *playlist* as a strong entity type could be redundant, because it is an entity that exist only when a *customer* create it and when it is composed by *content*. So it is possible to modify the design in two ways: is possible to model it as a weak entity type, or removing the owner attribute, that refers to the strong entities *customer*, and substitute it with a relation 1:N with the entity *customer*; In this specific case, is not convenient to store the information as a weak entity type, because they actually are different objects in this *microword*, so the choose is to maintain it as a strong entity type and apply the above changes. Same argument can be applied to the attribute *owner* of the entity *content*; is convenient for the optimisation of the system to model it as a new relationship *owner* with the entity *customer*, and assign to it the attribute *type*. Also the cardinality of the relation *subscription* has been changed because of the requirement n.4, so that many *customers* can submit more than one *subscription_plan*; this is done in anticipation of new subscription plans entering the system, for example, based on content production houses, that may eventually be bought together.
The attribute *production_company* has been added as participant unique key of the entity *content* to avoid ambiguity, since is possible to have different content with the same name but not with same name and from same production company.
The composed attributes of *type* and *fullname* are removed to decrease the complexity of the design, whereas they are not required in the functional analysis, and the attribute *price* is added to the entity *subscription_plan*.

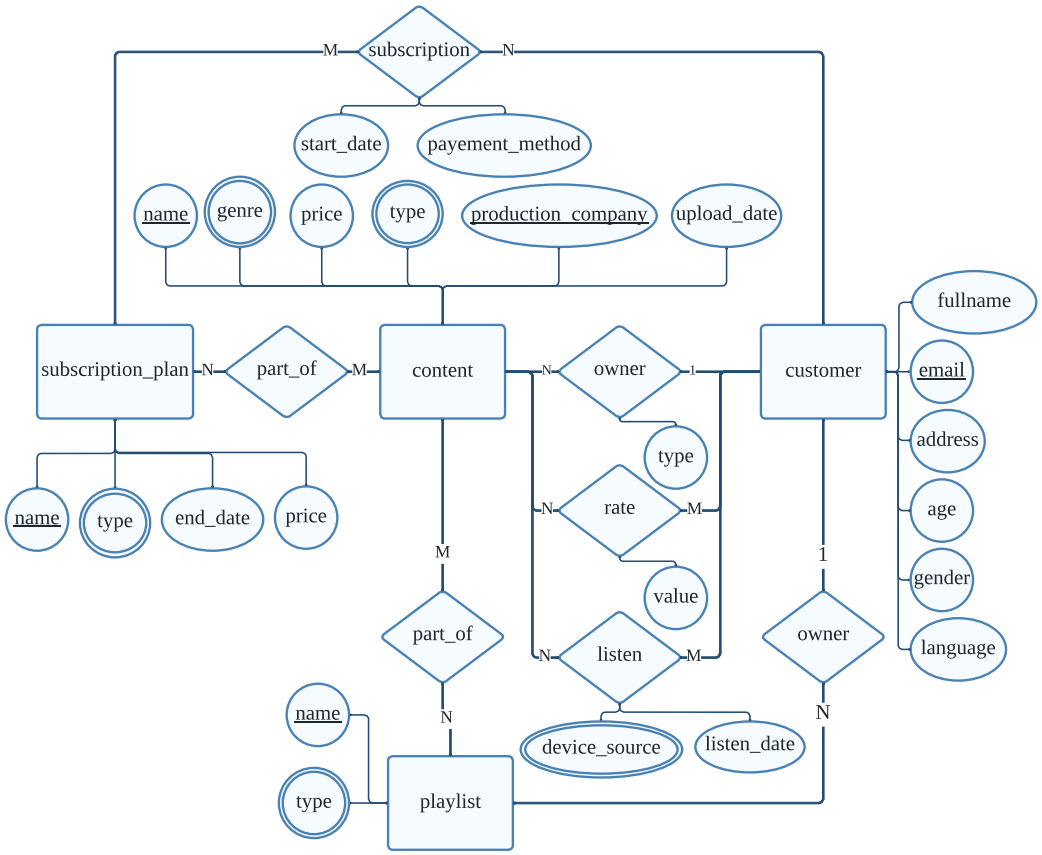


Figure 2: Refined ER Diagram

Analyzing the use-cases required we found that, following this new schema, is possible to satisfy all of them, in particular:

1. Is possible for users to list the audio contents of a given type they like the most, by setting a minimum rating, thanks to the attribute *value* of the relation *rate* and the attribute *type* of the entity *content*

2. Is possible for users to visualize new content on the platform, given some preferences like type and/or genre, thanks to the attributes *upload_date* and *type/genre* attributes in the entity *content*

3. Administrators of the service can list the email addresses of users who do not renew their subscriptions, by analysing the attributes *email* of the entity *customer* and *end_date* and *type* in *subscription_plan*, and compare it with which types of subscriptions were previously owned by them, by the attribute *type* of the entity *subscription_plan*

4. The service can recommend content that users may like based on their profile that they never listened to, by the relation *listen* and attributes *value* in *rate* relation, and all the profiling attributes of the entity *user*

3 Relational Model

Adopting the procedure presented in Chapter 9 of the textbook [2], the Conceptual Schema of the Database is mapped into a Relational Model by the following steps.

3.1 Mapping of Regular Entity Types

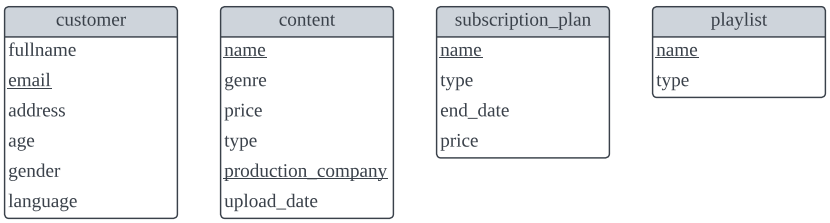
The relations created from strong entity types are:

1. customer (req. 2)

2. content (req. 1)

3. subscription_plan (req. 4)

4. playlist (req. 6)



3.2 Mapping of Weak Entity Types

No weak entity types are identified

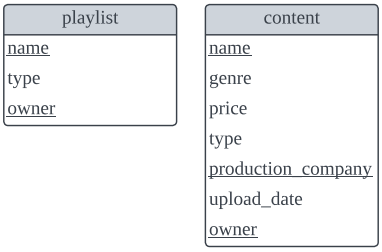
3.3 Mapping of Binary 1:1 Relation Types

No 1:1 relations are created

[2] *Fundamentals of Database Systems (7th Edition Ramez Elmasri and Shamkant B. Navathe*, Pearson)

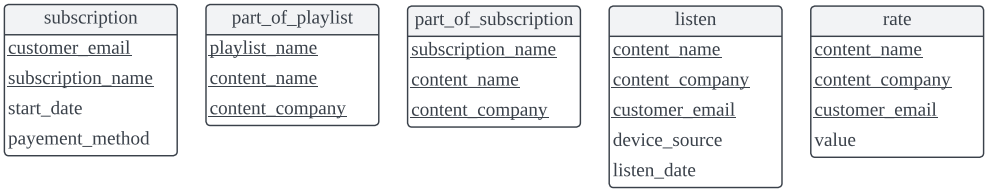
3.4 Mapping of Binary 1:N Relationship Types

owner relationship between playlist and customer is mapped by adding to *playlist* relation the attribute *owner* as a foreign key. *owner* relationship between content and customer is mapped by adding to *content* relation the attribute *owner* as a foreign key.



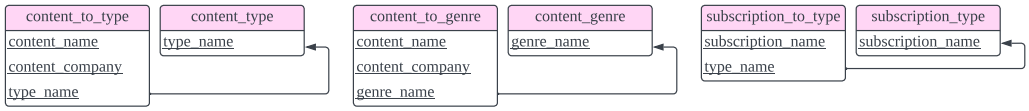
3.5 Mapping of Binary M:N Relationship Types

For regular binary M:N relationship types, are created new relations called *relationship relations*. Foreign keys attributes are included in the primary keys of the relations, that represent the participating entity types to form the primary key; Also any simple attributes of the original M:N relationship type are included as attributes of the relationship relations.



3.6 Mapping of Multivalued attributes

For each multivalued attribute of relations generated from the strong entity types, are created two new relations: one that holds the value of the attribute and the other that holds the relationship with the entity; the latter relation will include as foreign keys the primary key attributes of the entity which it refers, plus a foreign key that is the primary key attribute of the multivalued relation.



3.7 Mapping of N-ary Relationship Types.

No N-ary relationship are identified.

3.8 Relational Model

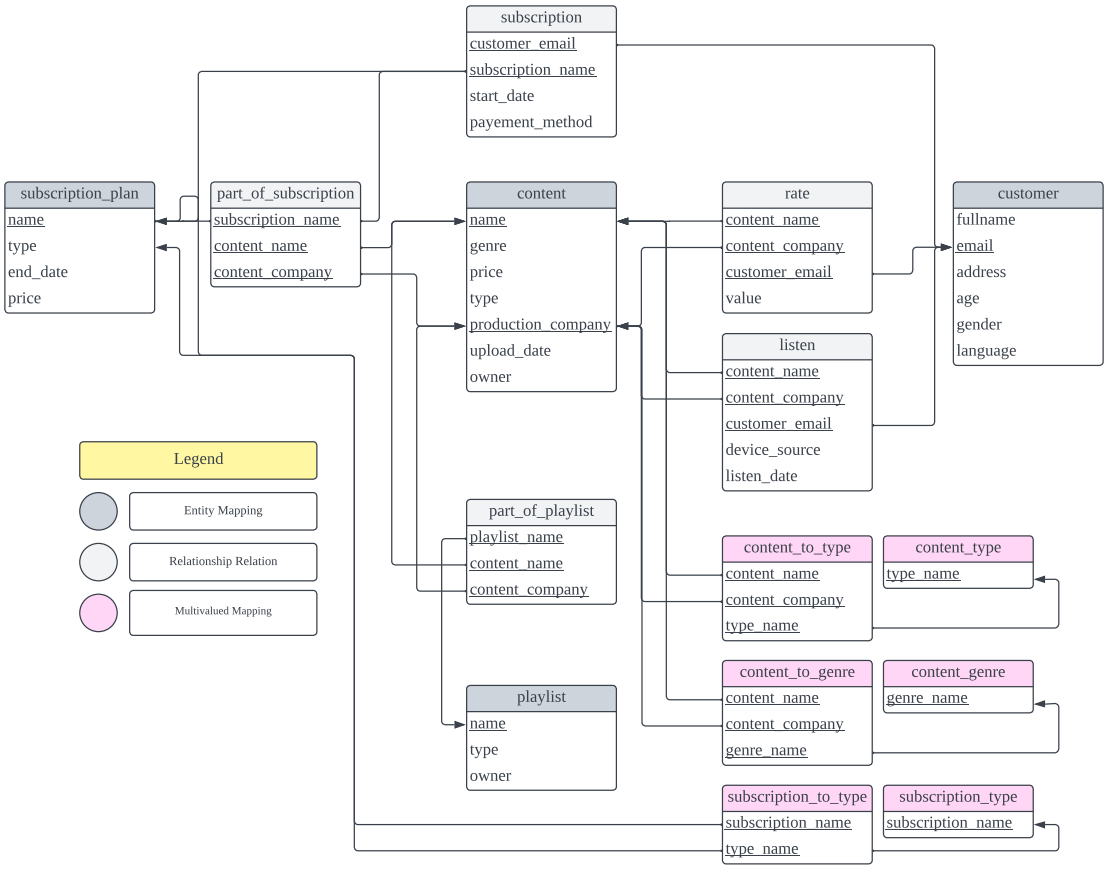


Figure 3: In the relational model are mapped the different objects of the microword created, with the links between all the attributes, taking into account the *references* that will be transcribed in the MySQL environment (arrows from dependent attributes to primary keys).

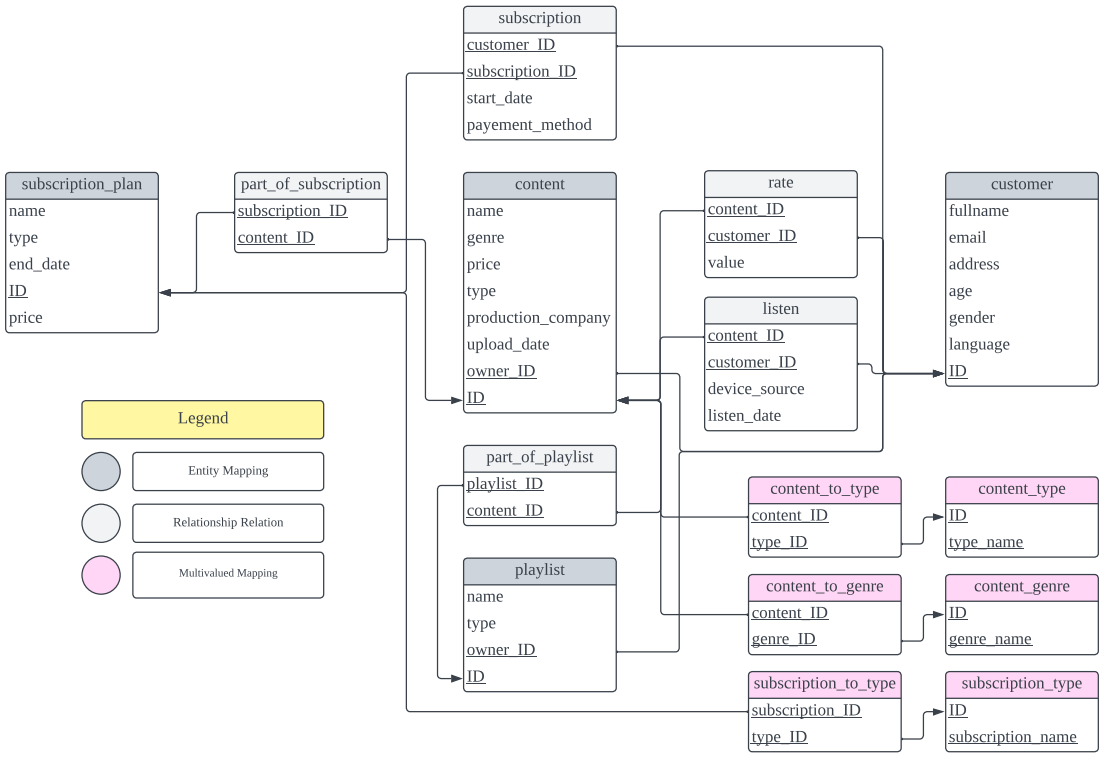


Figure 4: To built easily the relations in the MySQL environment and to reduce the complexity of the Relational Model, composed primary keys are replaced with ID attributes, that are unique for each entity and auto-incremented stacked.

4 MySQL Implementation

From the (Refined) Relational Model, is possible to build structures on *MySQL workbench*, corresponding to each relation in the diagram.
The DDL scripts reports the attributes of each relation, and the DML populates with data the Database. The strategy used is to simulate a real case, in which different customers want to subscribe plans, listen contents and create playlists.
Actually, the simulation is used to simplify the process of implementation and make it easily understandable; it is composed of the following elements:

1. Marco, Anna, Fabio, Giovanni and Lisa make a registration to the platform.
2. the following contents are uploaded into the platform:
   'Infinite Horizons' - a single, 'Oltre il confine' - a single, 'Un nuovo inizio' - a single, 'Luci nella notte' - a single, 'Sogno infinito' - a single, 'Nel cuore dell'amore' - a single, 'Sussurri del Silenzio' - a podcast, 'Tra le Onde del Tempo' - an album, 'Conversazioni Intime' - an audio-book, 'I Colori dell'Anima' - an audio-track.
3. three playlists are created from male customer and two from female, called respectively:
   'Energia Esplosiva', 'Notte Magica', 'Risveglio Sereno' and 'Retro Vibes', 'Indie Vibes'.
4. three kind of subscription plan are avaiable:
   'Melody Unlimited', 'Rhythm Plus', 'Harmony Pro'.
5. Marco, Lisa and Fabio submit to 'Melody Unlimited', Anna and Giovanni to 'Harmony Pro'.
6. Playlists are composed as follow:
   'Risveglio Sereno' is composed by: 'Infinite Horizons', 'Oltre il confine', 'Un nuovo inizio';

'Notte Magica' is composed by: 'Un nuovo inizio', 'Luci nella notte';
'Energia Esplosiva' by 'Sogno infinito', 'Nel cuore dell'amore', 'Sussurri del Silenzio';
'Retro Vibes' is composed by 'I Colori dell'Anima', 'Nel cuore dell'amore'.
'Indie Vibes' is composed by 'Infinite Horizons', 'Oltre il confine'.

7. Marco and Lisa listen to the contents 'Infinite Horizons', 'Oltre il confine', 'Un nuovo inizio' and rated them;
   Giovanni and Anna listen to 'Conversazioni Intime', 'I Colori dell'Anima' and rated them.

8. The subscription plans are composed by the following contents:
   'Oltre il confine', 'Un nuovo inizio', 'Luci nella notte', 'Sogno infinito', 'Nel cuore dell'amore' are part of the plan 'Harmony Pro';
   'Sussurri del Silenzio', 'Tra le Onde del Tempo' are part of the plan 'Melody Unlimited'.

For the points 1,2,3 and 4, queries with specified values are used to populate the tables; the remaining tables are populated from the latter, maintaining the foreign key's constraints.
To simulate the multivalued attribute that belongs to the relationship in the conceptual schema 2, is used a *check* statement on the possible values available for these attributes, since does not exist a step to map this situation in the steps of Section 3. Attributes as gender of customer or the type of playlist are modelled using *flags* (respectively "male" or "female" and "public" or "private").

customer_fullname	email	address	age	gender	language	device_source	listen_date	content_name	genre	price	type	production_company	upload_date
Anna Annina	anna.a@example.com	456 Elm St	30	F	Italian	smartphone	2023-06-20	Conversazioni Intime	rock	2	single	Production Co 2	2023-02-23
Anna Annina	anna.a@example.com	456 Elm St	30	F	Italian	radio	2023-06-20	I Colori dell'Anima	rock	2	single	Production Co 3	2023-05-01
Giovanni Giannoni	giova.gio@example.com	789 Oak St	35	M	French	smart_tv	2023-06-20	Conversazioni Intime	rock	2	single	Production Co 2	2023-02-23
Giovanni Giannoni	giova.gio@example.com	789 Oak St	35	M	French	smartphone	2023-06-20	I Colori dell'Anima	rock	2	single	Production Co 3	2023-05-01
Lisa Li	lisa.l@example.com	789 Oak St	35	F	French	smartphone	2023-06-20	Infinite Horizons	pop	10	single	Production Co 1	2023-01-01
Lisa Li	lisa.l@example.com	789 Oak St	35	F	French	radio	2023-06-20	Oltre il confine	indie	2	single	Production Co 2	2023-02-01
Lisa Li	lisa.l@example.com	789 Oak St	35	F	French	smartphone	2023-06-20	Un nuovo inizio	metal	15	album	Production Co 3	2023-03-01
Marco Marconi	marco.marc@example	123 Main St	25	M	English	smart_tv	2023-06-20	Infinite Horizons	pop	10	single	Production Co 1	2023-01-01
Marco Marconi	marco.marc@example	123 Main St	25	M	English	smartphone	2023-06-20	Oltre il confine	indie	2	single	Production Co 2	2023-02-01
Marco Marconi	marco.marc@example	123 Main St	25	M	English	smart_tv	2023-06-20	Un nuovo inizio	metal	15	album	Production Co 3	2023-03-01

Figure 5: Overview of data uploaded in the database

4.1 Results

To test the Database's Design, queries for use cases have been implemented; the results are reported below and match the expected values, in particular, for each use case listed in Section 1:

```
500     #primo use-case
501 •   select name from content where ID in (select content_ID from rate where value>=4 ) and type='single';
502
```

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

name
Oltre il confine
Conversazioni Intime
I Colori dell'Anima

Figure 6: Contents of 'single' type and with a rate greater or equal at 4 are retrieved by this query.

```
503     #secondo use-case
504 •   select name from content where upload_date >= (date_sub(current_date(), interval 3 month)) or genre='pop';
505
```

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

name
Infinite Horizons
Nel cuore dell'amore
I Colori dell'Anima

Figure 7: Content with an 'upload_date' greater than three months ago or with genre 'pop' are retrieved by this query.

```
507     # terzo use-case
508 •   SELECT * from (
509     SELECT
510         email,
511         CASE
512             WHEN sp.type = 'weekly' THEN DATE_ADD(start_date, INTERVAL 7 DAY)
513             WHEN sp.type = 'monthly' THEN DATE_ADD(start_date, INTERVAL 1 MONTH)
514             WHEN sp.type = 'yearly' THEN DATE_ADD(start_date, INTERVAL 1 YEAR)
515         END AS subscription_end_date,
516         name
517     FROM
518         customer c
519             JOIN
520         subscription s ON c.ID = s.customer_ID
521             JOIN
522         subscription_plan sp ON s.subscription_ID = sp.ID) as derived
523     WHERE
524         subscription_end_date >= current_date()
525             AND subscription_end_date <= DATE_ADD(current_date(), INTERVAL 31 DAY);
526
```

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

email	subscription_end_date	name
fabio.f33@example.com	2023-07-20	Melody Unlimited

Figure 8: Customer's email, subscription's end-date and subscription's name are obtained using *case-when* statement; the duration of the active plan is added to the start-date of the subscription to retrieve the end-date. Querying the joint table of customer, subscription and subscription_plan is obtained (as expected) that only Fabio has a subscription expiring within 31 days from 'now'.

```
526  #quarto use-case
527  select customer.ID as cust, content.ID as cont from content inner join
528  (SELECT t.genre, t.customer_ID, t.cont_count
529      FROM
530      (SELECT genre, customer_ID, cont_count,ROW_NUMBER() OVER
531      (PARTITION BY customer_ID ORDER BY cont_count DESC) AS rn
532          FROM
533          (SELECT COUNT(*) AS cont_count, genre, customer_ID
534              FROM
535              (SELECT customer_ID, content_ID, name, genre, fullname FROM customer cu
536              LEFT JOIN listen li ON cu.ID = li.customer_ID
537              JOIN content co ON content_ID = co.ID) AS a
538              GROUP BY genre, customer_ID
539              ORDER BY cont_count desc
540          ) as n
541      ) AS t WHERE t.rn = 1
542  ) as tot on tot.genre=content.genre
543  inner join customer on tot.customer_ID=customer.ID
544  where (content.ID, customer.ID) not in (select content_ID, customer_ID from listen);
545
```

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

cust	cont
5	6
1	6
4	8
2	8

Figure 9: This query simulates a recommendation system for customers of the platform, based on the most listen genre of contents; first, is counted the number of contents listen of a each genre, after is filtered just the most frequent genre and, in the end, contents of the same genre (not already listen) are recommended to each customer.