A Platform for Creators to Shine Audiences to Explore

Happily Made By:

Khaled Almansour - 221110083
Osama Azab - 221110390
Mouad Ghouti - 221110516

CS340 Phase 1

Description of the Team

- Khaled Almansour - 221110083
- Osama Azab - 221110390
- Mouad Ghouti - 221110516

Description of the purpose and scope of the application

- Purpose: A platform to allow individual creatives to publish their original video content to public users without publishing cost that they cannot afford.
- Scope of the application:
    - The platform supports only video content.
    - The platform is catered to individual creatives.
    - The platform is a web application only.
    - The platform will allow users to create accounts.
    - The platform supports audience engagement. khjd

Data Requirements

- User information: email, password, and name ..etc
- Channel information: name, creation date, subscribers, views, ..etc
- Video information: title, genre, channel, duration, file_url, ..etc
- Interactions information: likes, comments, ..etc

Functional Requirements

- The application must allow a user to create an account.
- The application must allow a user to make a channel.
- The application must allow a user to publish a video on his channel.
- The application shall allow a user to view a home page, channel page, and a video page.
- The application shall allow a user to interact with a video: liking, disliking, commenting, and sharing.
- The application shall allow a user to subscribe to a channel.

- The application shall allow a user to filter videos by genre, channel, and date.
- The application shall allow a user to delete videos from his channel.
- The application shall allow a user to edit or remove a comment that he made on a video.

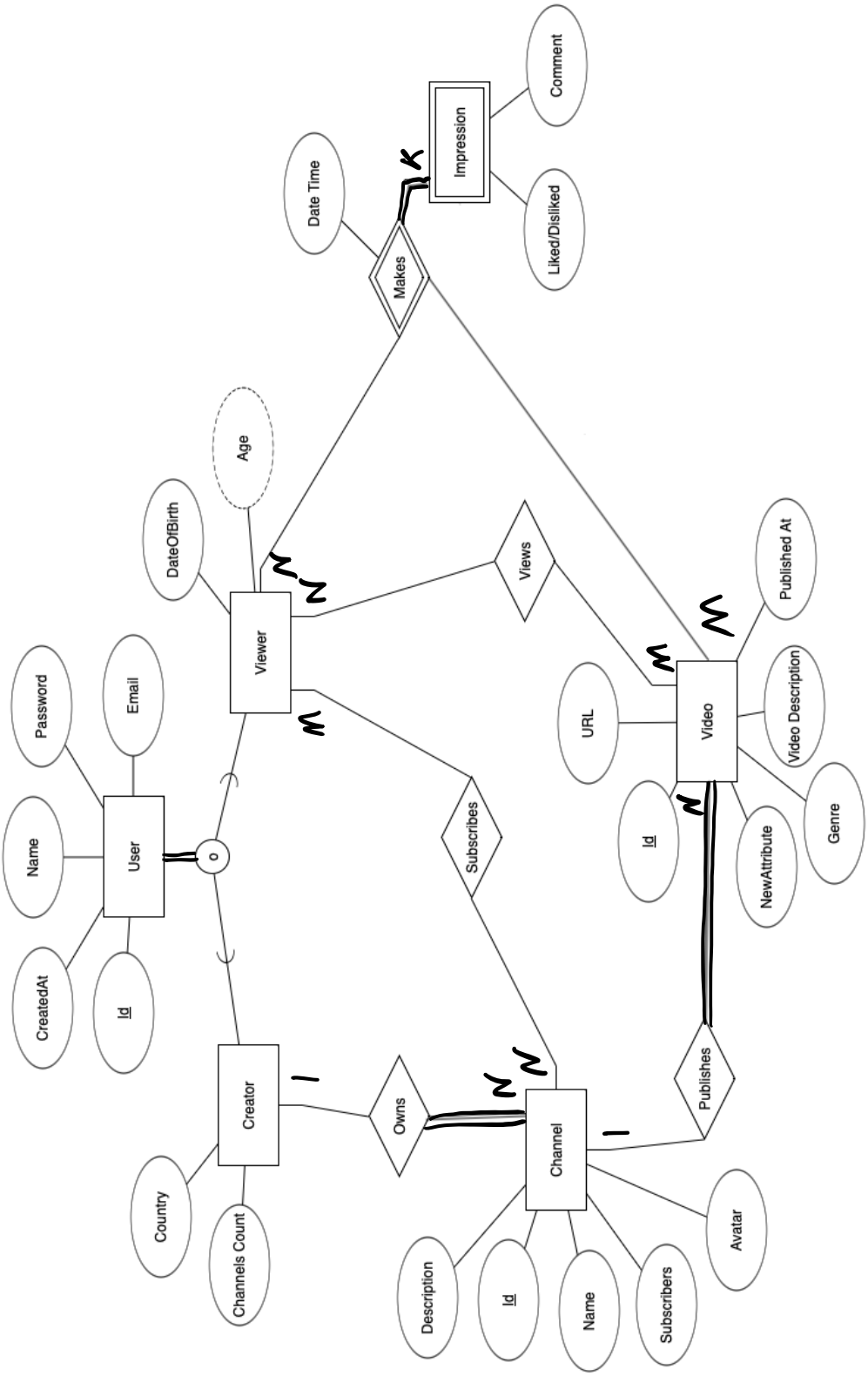
Non-Functional Requirements

- Usability: user friendly interface.
- Scalability: to accommodate increasing load.
- Compatibility: to support popular browsers and platforms.
- Performance: to have a short loading/waiting time.

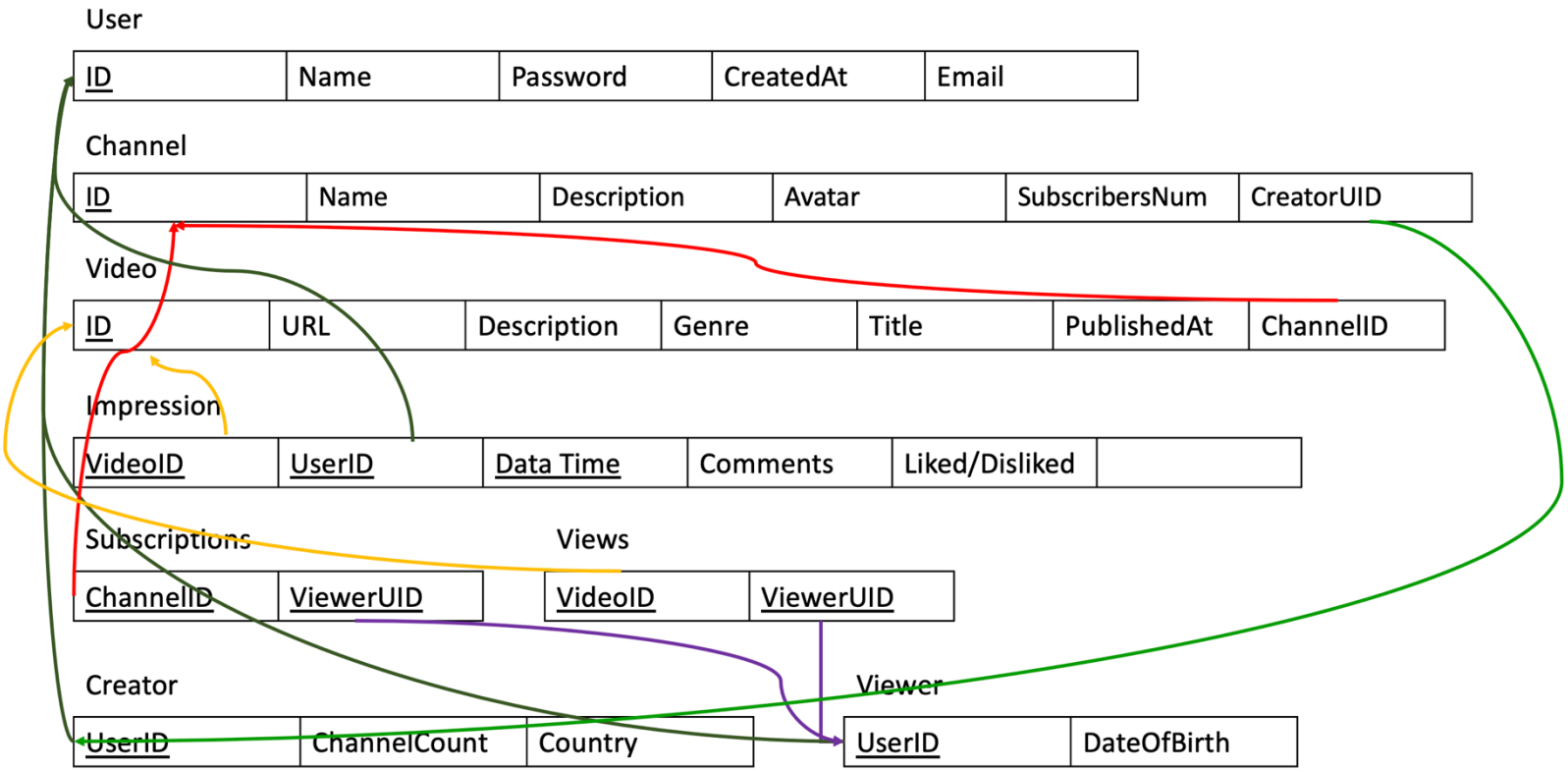
Contribution of the team members to phase 1 and strategy used to demonstrate teamwork:

- We had a meeting with all team members and after discussion and brainstorming we wrote phase 1 document together.
- We used AI technologies to help us with the brainstorming.

CS340 Phase 2:ER Diagram



S340 Phase 3: Relational Model



S340 Phase 4: SQL-Creating Database

```
-- Users Tables
CREATE TABLE users(
    id SERIAL NOT NULL,
    name VARCHAR(50) NOT NULL,
    password VARCHAR(250) NOT NULL,
    created_at date NOT NULL,
    email VARCHAR(250) NOT NULL,
    PRIMARY KEY(id)
);

CREATE UNIQUE INDEX user_email_index ON "users" USING btree ("email");

CREATE TABLE creators(
    user_id integer NOT NULL,
    channel_count integer NOT NULL DEFAULT 0,
    country VARCHAR(4) NOT NULL DEFAULT 'KSA',
    PRIMARY KEY(user_id),
    CONSTRAINT creators_user_id_fkey FOREIGN key(user_id) REFERENCES
users(id)
);

CREATE TABLE viewers(
    user_id integer NOT NULL,
    date_of_birth date NOT NULL,
    PRIMARY KEY(user_id),
    CONSTRAINT viewers_user_id_fkey FOREIGN key(user_id) REFERENCES users(id)
);

-- Channel Tables
CREATE TABLE channels(
    id SERIAL NOT NULL,
    name VARCHAR(50) NOT NULL,
    description VARCHAR(500),
    avatar_url VARCHAR(250),
    subscribers_count integer NOT NULL DEFAULT 0,
    created_at date NOT NULL,
    creator_uid integer NOT NULL,
    PRIMARY KEY(id),
    CONSTRAINT channels_creator_uid_fkey FOREIGN key(creator_uid) REFERENCES
creators(user_id)
);

CREATE UNIQUE INDEX channels_name_key ON "channels" USING btree ("name");

CREATE TABLE subscriptions(
```

```
    channel_id integer NOT NULL,
    viewer_uid integer NOT NULL,
    PRIMARY KEY(channel_id,viewer_uid),
    CONSTRAINT subscriptions_channel_id_fkey FOREIGN key(channel_id)
REFERENCES channels(id),
    CONSTRAINT subscriptions_viewer_uid_fkey FOREIGN key(viewer_uid)
REFERENCES viewers(user_id)
);

-- Video Tables
CREATE TABLE videos(
    id SERIAL NOT NULL,
    url VARCHAR(250) NOT NULL,
    title VARCHAR(250) NOT NULL,
    description VARCHAR(500),
    genre VARCHAR(15),
    published_at date NOT NULL,
    channel_id integer NOT NULL,
    PRIMARY KEY(id),
    CONSTRAINT videos_channel_id_fkey FOREIGN key(channel_id) REFERENCES
channels(id)
);

CREATE UNIQUE INDEX videos_url_key ON "videos" USING btree ("url");

CREATE TABLE views(
    video_id integer NOT NULL,
    viewer_uid integer NOT NULL,
    PRIMARY KEY(video_id,viewer_uid),
    CONSTRAINT views_video_id_fkey FOREIGN key(video_id) REFERENCES
videos(id),
    CONSTRAINT views_viewer_uid_fkey FOREIGN key(viewer_uid) REFERENCES
viewers(user_id)
);

CREATE TABLE impressions(
    video_id integer NOT NULL,
    viewer_uid integer NOT NULL,
    date_time date NOT NULL,
    liked boolean,
    disliked boolean,
    comment VARCHAR(500),
    PRIMARY KEY(video_id,viewer_uid,date_time),
    CONSTRAINT impressions_video_id_fkey FOREIGN key(video_id) REFERENCES
videos(id),
    CONSTRAINT impressions_viewer_uid_fkey FOREIGN key(viewer_uid) REFERENCES
viewers(user_id)
);
```

SAMPLE INSERTS

```
insert into
    users(
        id,
        name,
        password,
        created_at,
        email
    )
values
(
        1,
        'Davida Van Hove',
        'vV9/dw.kCVZqBvVm',
        '2022-09-02',
        'dvan0@go.com'
    );

insert into
    users(
        id,
        name,
        password,
        created_at,
        email
    )
values
(
        2,
        'Walden Seldon',
        'mC2%=qm{\'7T9',
        '2023-08-17',
        'wseldon1@yolasite.com'
    );

insert into
    users(
        id,
        name,
        password,
        created_at,
        email
    )
values
(
        3,
        'Roana Hukin',
        'bK2~Hlrqq4cmT{lw',
        '2023-10-06',
        'rhukin2@mysql.com'
    );

insert into
    users(
        id,
```

```
        name,
        password,
        created_at,
        email
    )
values
(
        4,
        'Odilia Pinnell',
        'aU0{\'K,S84uCV\\',
        '2023-03-07',
        'opinnell3@alibaba.com'
    );

insert into
    users(
        id,
        name,
        password,
        created_at,
        email
    )
values
(
        5,
        'Briano Lansdowne',
        'vD1)0\'Is%',
        '2022-10-30',
        'blansdowne4@blogger.com'
    );
```

- Full Scripts are attached to Trello

S340 Phase 5: SQL Queries

```
var dt = ExecuteQuery(
    "SELECT count(1) " +
    "FROM " +
    "impressions i, " +
    "videos v, " +
    "channels c " +
    "WHERE " +
    "i.video_id = v.id " +
    "AND v.channel_id = c.id " +
    "AND c.id = @channelId " +
    "AND i.liked = true " +
    "AND NOT EXISTS ( " +
    "SELECT 1 " +
    "FROM subscriptions " +
    "WHERE " +
    "channel_id = c.id " +
    "AND viewer_uid = i.viewer_uid)", parameter);
```

```
var UploadVideo = ExecuteCommand("INSERT INTO videos (url, title, description, genre, published_at, channel_id, thumbnail_url, duration_sec) " +
"VALUES (@url, @title, @description, @genre, now(), @channelId, @thumbnail_url, @durationSec)", parameter);
```

```
var dt = ExecuteQuery(
    "SELECT v.id, v.title, v.description, v.genre, v.duration_sec, v.published_at, v.url, v.thumbnail_url, " +
    "channel_id, c.name channel_name, c.avatar_url, (select count(1) from views where video_id = v.id) views_count"+
    " FROM videos v join channels c on v.channel_id = c.id where v.channel_id=@channelId  ", parameters);
```

```
var dt = ExecuteQuery(
    "select id, name, description, avatar_url, subscribers_count, created_at, creator_uid" +
    " from channels" +
    " where id = @channelId", new List<NpgsqlParameter>
{
    parameter
});
```

```
var dt = ExecuteQuery(
    "SELECT count(1) " +
    "FROM " +
    "impressions i, " +
    "videos v, " +
    "channels c " +
    "WHERE " +
    "i.video_id = v.id " +
    "AND v.channel_id = c.id " +
    "AND c.id = @channelId " +
    "AND i.disliked = true " +
    "AND NOT EXISTS ( " +
    "SELECT 1 " +
    "FROM subscriptions " +
    "WHERE " +
    "channel_id = c.id " +
    "AND viewer_uid = i.viewer_uid)", parameter);
```

```
var dt = ExecuteQuery(
    "SELECT v.id, v.title, v.description, v.genre, v.duration_sec, v.published_at, v.url, v.thumbnail_url, " +
    "channel_id, c.name channel_name, c.avatar_url, (select count(1) from views where video_id = v.id) views_count"+
    " FROM videos v join channels c on v.channel_id = c.id WHERE (v.genre ILIKE @Text) OR (c.name ILIKE @Text) OR (v.description ILIKE @Text)", parameters);
```

```
var dt = ExecuteQuery(
    "SELECT count(1) " +
    "FROM " +
    "impressions i, " +
    "videos v, " +
    "channels c " +
    "WHERE " +
    "i.video_id = v.id " +
    "AND v.channel_id = c.id " +
    "AND c.id = @channelId " +
    "AND i.liked = true " +
    "AND EXISTS ( " +
    "SELECT 1 " +
    "FROM subscriptions " +
    "WHERE " +
    "channel_id = c.id " +
    "AND viewer_uid = i.viewer_uid)", parameter);
```

```
var dt = ExecuteQuery(
    "SELECT count(1) " +
    "FROM " +
    "impressions i, " +
    "videos v, " +
    "channels c " +
    "WHERE " +
    "i.video_id = v.id " +
    "AND v.channel_id = c.id " +
    "AND c.id = @channelId " +
    "AND i.disliked = true " +
    "AND EXISTS ( " +
    "SELECT 1 " +
    "FROM subscriptions " +
    "WHERE " +
    "channel_id = c.id " +
    "AND viewer_uid = i.viewer_uid)", parameter);
```

```
var dt = ExecuteQuery(
    "SELECT v.id, v.title, v.description, v.genre, v.duration_sec, v.published_at, v.url, v.thumbnail_url, " +
    "channel_id, c.name channel_name, c.avatar_url, (select count(1) from views where video_id = v.id) views_count"+
    " FROM videos v join channels c on v.channel_id = c.id where v.genre ILIKE @genre ", parameters);
```

```
var dt = ExecuteQuery(
    "SELECT DISTINCT genre FROM videos", new List<NpgsqlParameter>());
```

```
var dt = ExecuteQuery(
    "SELECT v.id, v.title, v.description, v.genre, v.duration_sec, v.published_at, v.url, v.thumbnail_url, channel_id, "+
    "c.name channel_name, c.avatar_url, (select count(1) from views where video_id = v.id) views_count "+
    "FROM videos v join channels c on v.channel_id = c.id JOIN "+
    "(SELECT video_id, COUNT(*) as total_views FROM views GROUP BY video_id ORDER BY total_views DESC LIMIT 1) AS most_viewed "+
    "ON v.id = most_viewed.video_id", parameter
);
```

```
var dt = ExecuteQuery(
    "SELECT v.id, v.title, v.description, v.genre, v.duration_sec, v.published_at, v.url, v.thumbnail_url, channel_id, c.name channel_name, c.avatar_url, (select count(1) from views where video_id = v.id)
    +" FROM videos v join channels c on v.channel_id = c.id WHERE v.channel_id=@channel_id AND v.id<>@id  ORDER BY published_at DESC", parameters);
```

```
var dt = ExecuteQuery(
    "SELECT v.id, v.title, v.description, v.genre, v.duration_sec, v.published_at, v.url, v.thumbnail_url, " +
    "channel_id, c.name channel_name, c.avatar_url, (select count(1) from views where video_id = v.id) views_count"+
    " FROM videos v join channels c on v.channel_id = c.id where v.channel_id=@channelId  ", parameters);
```

```
var dt = ExecuteQuery(
    "SELECT i.comment FROM impressions i JOIN videos v ON i.video_id = v.id WHERE i.video_id = @id ", parameters);
```

Like a video:

```
INSERT INTO impressions (video_id, viewer_uid, date_time, liked)
        VALUES (@videoId, @userId, now(), true);
```

Dislike a video:

```
DELETE FROM impressions
          WHERE video_id = @videoId AND viewer_uid = @userId and liked = true;
INSERT INTO impressions (video_id, viewer_uid, date_time, disliked)
        VALUES (@videoId, @userId, now(), true);
```

Post a comment:

```
INSERT INTO impressions (video_id, viewer_uid, date_time, comment)
        VALUES (@videoId, @userId, now(), @comment);
```

Update a comment:

```
UPDATE impressions SET comment = @comment, date_time = now()
 WHERE video_id = @videoId AND viewer_uid = @userId AND id = @commentId;
```

Delete a comment:

```
DELETE FROM impressions
          WHERE video_id = @videoId AND viewer_uid = @userId and id = !commentId;
```

Login:

```
SELECT 1 FROM users
            WHERE email = @email AND password = @password
```

Signup:

```
INSERT INTO users (name, email, password, created_at)
        VALUES (@name, @email, @password, CURRENT_DATE);
INSERT INTO viewers (user_id, date_of_birth)
        VALUES (@id, @dob);
INSERT INTO creators (user_id, country)
        VALUES (@id, @country);
```

Get User Profile:

```
SELECT u.*, c.user_id, c.country, v.date_of_birth
 FROM users u left outer join creators c on u.id = c.user_id
                    left outer join viewers v on u.id = v.user_id
WHERE u.id = @id;
```

```
Execute
1  SELECT
2      u.*,
3      c.user_id,
4      c.country,
5      v.date_of_birth
6  FROM users u
7      left outer join creators c on u.id = c.user_id
8      left outer join viewers v on u.id = v.user_id
9  WHERE u.id = 13;  95ms
```

Result

Search results | Cost: 95ms | 1 | Total 1

	id	name	password	created_at	email	user_id	country
1	13	Michaeline Ruddom	b03ddf3ca2e714a6548e7	2022-10-18	mruddomc@sciencedaily.(	13	SA

Get User Channels:

```
 SELECT *
  FROM channels
WHERE creator_uid = @userId;
```

```
Execute
1  SELECT * FROM channels WHERE creator_uid = 13;
```

channels

Search results | Cost: 104ms | 1 | Total 1

	id integer	* name varchar(50)	description varchar(500)	avatar_url varchar(250)	* subscribers_count integer	* created_at date
1	11	Michaeline Channel's	ac orci phasellus egestas	https://example.com/avata	572316	2013-10-07

Most Liked Videos:

```
SELECT *
  FROM videos v
 WHERE (SELECT count(1) FROM impressions WHERE video_id = v.id and liked = true) >= ALL (
    SELECT count(1) FROM impressions WHERE liked = true GROUP BY video_id
 );
```

```
Execute
1  SELECT *
2    FROM videos v
3   WHERE (SELECT count(1) FROM impressions WHERE video_id = v.id and liked = true) >= ALL (
4      SELECT count(1) FROM impressions WHERE liked = true GROUP BY video_id
5   )
```

Result

Search results Cost: 136ms 1 Total 27

	id	url	title	description	genre	published_at	channe
1	20	https:\\youtube.com\v\8IZ	Corabel Channel's	ac orci phasellus egestas	others	2021-10-22	8
2	22	https:\\youtube.com\v\10I	Janot Channel's	ac orci phasellus egestas	others	2010-03-05	10
3	24	https:\\youtube.com\v\12I	Emmit Channel's	ac orci phasellus egestas	others	2007-11-20	12

Most Disliked Videos:

```
SELECT *
 FROM videos v
WHERE (SELECT count(1) FROM impressions WHERE video_id = v.id and disliked = true) >= ALL (
  SELECT count(1) FROM impressions WHERE disliked = true GROUP BY video_id
);
```

```
Execute
1  SELECT *
2    FROM videos v
3   WHERE (SELECT count(1) FROM impressions WHERE video_id = v.id and disliked = true) >= ALL (
4      SELECT count(1) FROM impressions WHERE disliked = true GROUP BY video_id
5   ) 124ms
```

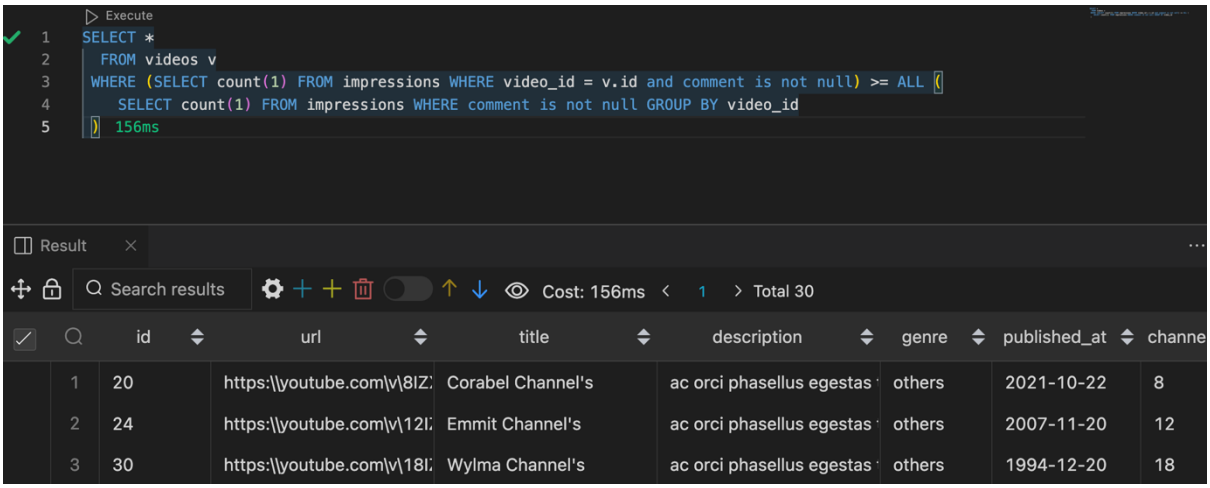
Result

Search results Cost: 124ms 1 Total 18

	id	url	title	description	genre	published_at	channe
1	127	https:\\youtube.com\v\7IZ	New Test	ac orci phasellus egestas	otherTest	2023-12-06	13
2	21	https:\\youtube.com\v\9IZ	Emeline Channel's	ac orci phasellus egestas	others	2020-08-04	9
3	27	https:\\youtube.com\v\15I	Giordano Channel's	ac orci phasellus egestas	others	2021-09-30	15

Most Commented Videos:

```
SELECT *
 FROM videos v
WHERE (SELECT count(1) FROM impressions WHERE video_id = v.id and comment is not null)
>= ALL (
  SELECT count(1) FROM impressions WHERE comment is not null GROUP BY video_id
);
```


Execute
SELECT *
FROM videos v
WHERE (SELECT count(1) FROM impressions WHERE video_id = v.id and comment is not null) >= ALL (
SELECT count(1) FROM impressions WHERE comment is not null GROUP BY video_id
) 156ms
Result
Search results
Cost: 156ms
1
Total 30
id
url
title
description
genre
published_at
channe
1 20 https:\\youtube.com\v\8IZ Corabel Channel's ac orci phasellus egestas others 2021-10-22 8
2 24 https:\\youtube.com\v\12I Emmit Channel's ac orci phasellus egestas others 2007-11-20 12
3 30 https:\\youtube.com\v\18I Wylma Channel's ac orci phasellus egestas others 1994-12-20 18