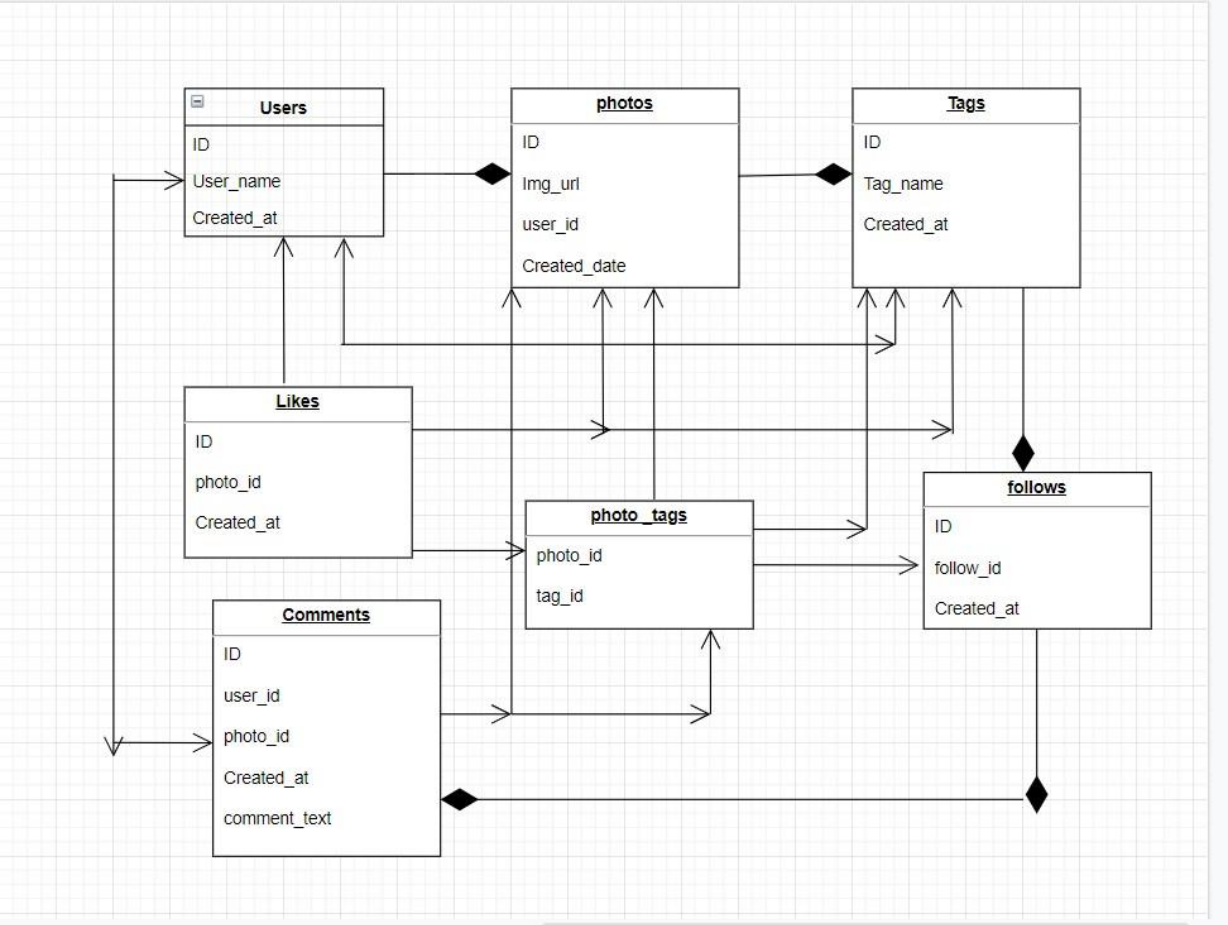
INSTAGRAM DATBASE ANALYSIS

UML Diagram For all tables



This UML diagram was made because SQL join syntax is simple to grasp and write. It helps to understand the table and connections between all of the columns that are given in the dataset.

I am using SQL Workbench 8.0.30 for this project

Host:	**LAPTOP-SD0EDQIM**
Socket:	**MySQL**
Port:	**3306**
Version:	**8.0.30 (MySQL Community Server - GPL)**
Compiled For:	**Win64 (x86_64)**

Q1.The Five Oldest Users on Instagram

```
5    /* Identify the five oldest users on Instagram from the provided database.*/
6    select username,created_at from users
7    order by created_at
8    limit 5
```

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

username	created_at
Darby_Herzog	2016-05-06 00:14:21
Darby_Herzog	2016-05-06 00:14:21
Emilio_Bernier52	2016-05-06 13:04:30
Emilio_Bernier52	2016-05-06 13:04:30
Elenor88	2016-05-08 01:30:41

In order to answer the question about the five oldest Instagram users, we must first determine which tables will work best for our search.

The users table, which has the two columns username and created date, is sufficient to answer the question when used with the mentioned syntax; however, the limit operator is required to retrieve only five users' data.

Q2.Identify users who have never posted a single photo on Instagram

```
10   /*Identify users who have never posted a single photo on Instagram*
11   SELECT u.id AS user_id, u.username, u.created_at
12   FROM users u
13   LEFT JOIN photos p ON u.id = p.user_id
14   WHERE p.id IS NULL;
15
```

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

user_id	username	created_at
5	Aniya_Hackett	2016-12-07 01:04:39
7	Kasandra_Homenick	2016-12-12 06:50:08
14	Jaclyn81	2017-02-06 23:29:16
21	Rocio33	2017-01-23 11:51:15
24	Maxwell.Halvorson	2017-04-18 02:32:44
25	Tierra.Trantow	2016-10-03 12:49:21
34	Pearl7	2016-07-08 21:42:01
36	Ollie_Ledner37	2016-08-04 15:42:20

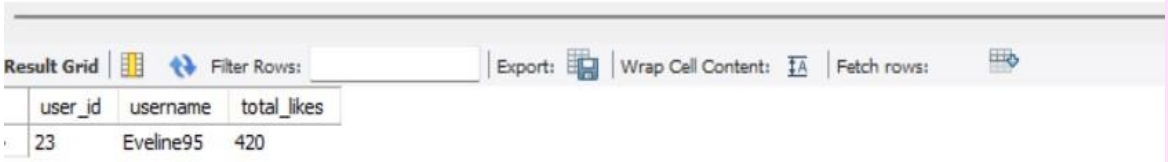
As you can notice this question is asked that who have never posted a single photo on Instagram so basically here we need two table one is users and another one is photo because we need the user name and another is photo posted date

The users table (u) and the photo table (p) sent joints using an LEFT JOIN on the id and user-id columns, respectively.

Users without photo entries in the photos database are excluded by the WHERE clause p.id IS NULL. The p.id IS NULL condition basically indicates that there isn't a match for a certain user in the photographs table, suggesting they haven't submitted any images.

Q3.Determine the winner of the contest and provide their details to the team

```
16    /* Determine the winner of the contest and provide their details to the team */
17
18 •  select * from photos,likes,users;
19
20 •  SELECT u.id AS user_id, u.username, count(l.user_id) AS total_likes
21    FROM users u
22    JOIN photos p ON u.id = p.user_id
23    LEFT JOIN likes l ON p.id = l.photo_id
24    GROUP BY u.id, u.username
25    ORDER BY total_likes DESC
26    LIMIT 1;
```



user_id	username	total_likes
23	Eveline95	420

From the users table (u), we choose the columns user_id, user_name, and COUNT(l.id) AS total_likes.

To link people with their images, we do an INNER JOIN operation between the users table (u) and the photos table (p) on the id and user_id columns, respectively.

The likes table (l) and the photographs table (p) are linked together using an LEFT JOIN on the id and photo_id columns, respectively.

This enables us to determine how many people have liked each image.The user's ID and username are used to group the results when using GROUP BY u.id, u.user_name.Using the total number of likes that each user has gotten, ORDER BY total_likes DESC orders the results in decreasing order.

We only receive the person with the highest likes (top1) by using LIMIT 1 condition.

Q4.Identify and suggest the top five most commonly used hashtags on the platform

```
28      /* Identify and suggest the top five most commonly used hashtags on the platform.
29 •    select * from photo_tags,tags
30      select t.tag_name , count(p.photo_id) as hash_tag from photo_tags p
31      inner join  tags t on t.id = p.tag_id
32      group by t.tag_name
33      order by hash_tag  desc;
34
```

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

tag_name	hash_tag
smile	59
beach	42
party	39
fun	38
concert	24
food	24
lol	24
hair	23

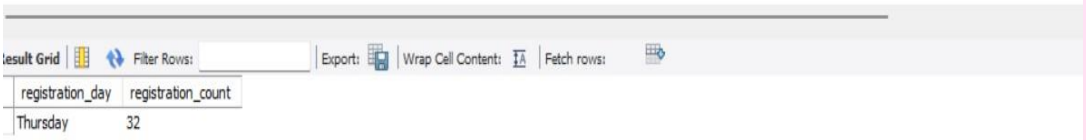
We decide to use the columns id and COUNT(photo_id) as hash_tags when forming the photo_tags(p).Utilising t.id and p.tag_id,

conduct an INNER JOIN operation between the photo_tags and tags, and then build GROUP BY functions using tag_name and

DESC is used because we want the top hash tag.

Q5.Determine the day of the week when most users register on Instagram. Provide insights on when to schedule an ad campaign

```
/* Determine the day of the week when most users register on Instagram. Provide insights on when to schedule an ad campaign.*/
select * from users
SELECT
    DAYNAME(created_at) AS registration_day,
    COUNT(*) AS registration_count
FROM
   users
GROUP BY
    registration_day
ORDER BY
    registration_count DESC
LIMIT 1;
```



registration_day	registration_count
Thursday	32

The day of the week is extracted from the "created date" column using the DAYNAME() method. Based on the date value, this method returns the name of the day (Monday, Tuesday, etc.).

The number of registrants is counted using the COUNT(*) function for each day of the week.

To group the results by the day of the week, we utilise the GROUP BY registration_day clause.

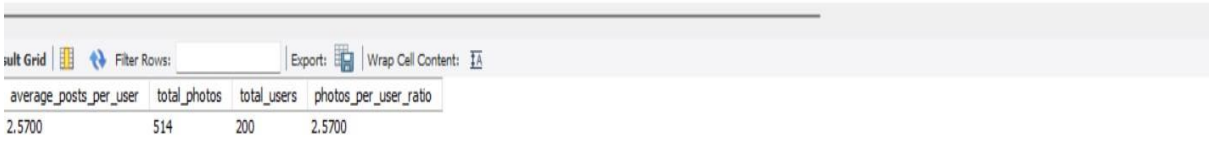
The registration_count ORDER BY The day with the most registrations will display first in the results since DESC orders the results descending depending on the number of registrations.

And used limit(1) function for single value.

Q6.Calculate the average number of posts per user on Instagram. Also, provide the total number of photos on Instagram divided by the total number of users

```
/*Calculate the average number of posts per user on Instagram. Also, provide the total number of photos on Instagram divided by the total number of users*/

SELECT
    COUNT(p.id) / COUNT(DISTINCT u.id) AS avg_posts_per_user,
    COUNT(p.id) AS total_photos,
    COUNT(DISTINCT u.id) AS total_users,
    COUNT(p.id) / COUNT(DISTINCT u.id) AS photos_per_user
FROM
    users u
LEFT JOIN
    photos p ON u.id = p.user_id;
```



average_posts_per_user	total_photos	total_users	photos_per_user_ratio
2.5700	514	200	2.5700

As average_posts_per_user, we use the result of the calculation COUNT(p.id) / COUNT(DISTINCT u.id). The COUNT(p.id) and COUNT(DISTINCT u.id) functions count the total number of users and photographs, respectively. We can get the average number of postings per user by dividing these two totals.

To determine the total number of Instagram photographs, we utilise COUNT(p.id).

To get the total number of unique users on Instagram, we utilise COUNT(DISTINCT u.id).

To obtain the ratio of photographs to users, which is the same as the average number of images per user, we repeat the COUNT(p.id) / COUNT(DISTINCT u.id) computation.

Q7.Identify users (potential bots) who have liked every single photo on the site, as this is not typically possible for a normal user.

```
/* Identify users (potential bots) who have liked every single photo on the site, as this is not typically possible for a normal user*/
select * from users,likes
SELECT
    user_id, username
FROM
    users u
INNER JOIN
    (SELECT COUNT(DISTINCT photo_id) AS total_photos FROM likes) l
LEFT JOIN
    (SELECT user_id, COUNT(DISTINCT photo_id) AS liked_photos
     FROM likes
     GROUP BY user_id) ul
ON u.id = ul.user_id
WHERE
    ul.liked_photos = l.total_photos;
```

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

user_id	username
5	Aniya_Hackett
14	Jaclyn81
21	Rocio33
24	Maxwell.Halvorson
36	Ollie_Ledner37
41	Mckenna17

Using a subquery in the INNER JOIN clause, we first determine the total number of different photographs that are accessible on the website. The subquery SELECT COUNT(DISTINCT photo_id) AS total_photos FROM likes counts all unique images in the "likes" table.

We then use a different subquery to determine how many photographs each user has liked. The subquery (SELECT user_id, COUNT(DISTINCT photo_id) AS liked_photos FROM likes GROUP BY user_id) tracks how many unique photos each user has liked.

The results of the subquery on the user_id column are then combined with the "users" table (u) using an LEFT JOIN.

Users who have liked every photo are excluded by the WHERE clause ul.liked_photos = l.total_photos since they have the same amount of liked photos as everyone else. meaning they have the same number of liked photos as the total number of photos available on the site.