# Instagram Clone Database

# download.jpg

The Instagram Clone Database is designed using SQL and consists of multiple tables to store user information, photos, likes, comments, hashtags, and more. Below are the main tables in the schema:

* Users: Connected to every other table in the dataset
* Photos: Stores image URL and the user who has posted it
* Comments: Stores details of comment text, user, and picture
* Likes: Stores details of the user who has liked the picture
* Follows: Stores relationship between users i.e., followers & following
* tags: Stores different hashtags used
* Photo\_tags: Connects hashtags with posted pictures

Challenges and Queries: Practical Scenarios  
Here are a few scenarios I tackled, along with MySQL queries used and their significance:

1. Finding the Five Oldest Users
2. Most Popular Registration Days
3. Users Who Never Posted a Photo
4. The Most Liked Photo
5. How many times does the average user post?
6. Top 5 Most Used Tags
7. Users Who Liked Every Photo
8. User who have never commented on a photo
9. Find the percentage of our users who have either never commented on a photo or have commented on every photo
10. Retrieve a list of users along with their usernames and the rank of their account creation
11. Show the tag name and the number of photos associated with that tag. Rank the tags
12. List the usernames of users who have posted photos along with the count of photos they have posted. Rank them by the number of photos

Key Learnings:

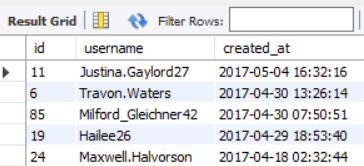
* Writing optimized SQL queries for efficient data extraction.
* Using **JOINs, subqueries, aggregate functions**, **rank()** to uncover hidden insights.
* Understanding user behaviors and engagement patterns from raw data.

Q.1 Find the 5 oldest users. We want to reward our users who have been around the longest

select \* from users

order by created\_at desc

limit 5;



Q.2 What day of the week do most users register on?

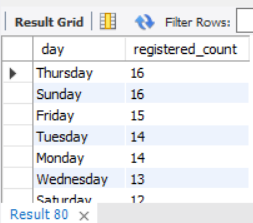
select DAYNAME(created\_at) as day,

count(\*) as registered\_count

from users

group by day

order by registered\_count desc;



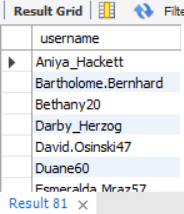
Q.3 We want to target our inactive users with an email campaign. Find the users who have never posted a photo

select username

from users u

left join photos p on u.id = p.user\_id

where p.user\_id is NULL;



Q.4 We're running a new contest to see who can get the most likes on a single photo.

select u.username, p.id, p.image\_url, count(\*) as total

from photos p

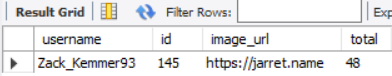
join likes l on l.photo\_id = p.id

join users u on u.id = p.user\_id

group by p.id

order by total desc

limit 1;



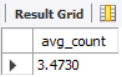
Q.5 Our Investors want to know how many times does the average user post?

select avg(total\_count) as avg\_count from

(select user\_id, count(\*) as total\_count

from photos

group by user\_id) as user\_count;



Q.6 A brand wants to know which hashtags to use in a post. What are the top 5 most commonly used hashtags?

select t.tag\_name, p.tag\_id, count(\*) as count\_of\_use

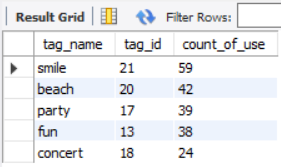
from photo\_tags p

join tags t on p.tag\_id = t.id

group by p.tag\_id

order by count\_of\_use desc

limit 5;



Q.7 We have a small problem with bots on our site.Find users who have liked every single photo on the site.

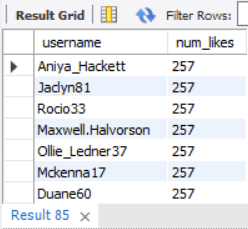
select u.username, count(\*) as num\_likes

from users u

join likes l on l.user\_id = u.id

group by l.user\_id

having num\_likes = (select count(\*) from photos);

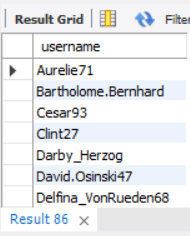


Q.8 We also have a problem with celebrities. Find users who have never commented on a photo

select username

from users u left join comments c on u.id = c.user\_id

where c.user\_id is NULL;



Q.9 Find the percentage of our users who have either never commented on a photo or have commented on every photo

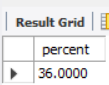
select (count(\*)/(select count(\*) from users)\*100) as percent

from (select users.id, count(distinct photo\_id) as comms

from comments right join users on comments.user\_id = users.id

group by users.id

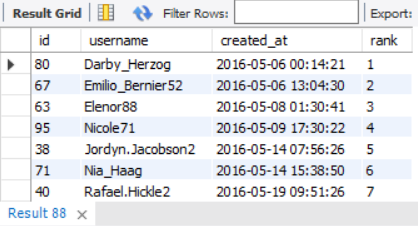
having comms = (select count(\*) from photos) or comms = 0 ) as total;



Q.10 Retrieve a list of users along with their usernames and the rank of their account creation, ordered by the creation date in ascending order.

select id, username, created\_at,

rank () over (order by created\_at) as 'rank' from users;



Q.11 For each tag, show the tag name and the number of photos associated with that tag. Rank the tags by the number of photos in descending order.

select tag\_name, num\_of\_photos, rank() over (order by num\_of\_photos desc) as 'Rank'

from

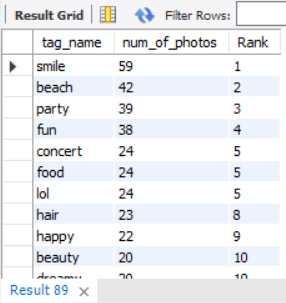
( select t.tag\_name, count(pt.photo\_id) as num\_of\_photos

from photo\_tags pt

join tags t on pt.tag\_id = t.id

group by t.tag\_name

order by num\_of\_photos desc ) as t;



Q.12 List the usernames of users who have posted photos along with the count of photos they have posted. Rank them by the number of photos in descending order.

select username, photo\_posted, rank() over (order by photo\_posted desc) as 'Rank'

from

( select u.username, count(p.id) as photo\_posted

from photos p

join users u on u.id = p.user\_id

group by username

order by photo\_posted desc) as temp;

