**TRAINITY ASSIGNMENT**

**INSTAGRAM USER ANALYTICS**

**Problem statement :**

You're a data analyst at Instagram, tasked with using SQL and MySQL Workbench to analyze user data. Your goal is to provide valuable insights that can guide decision-making for the product team. By examining how users interact with the app, you'll help identify trends and patterns that can inform decisions about new features, marketing campaigns, and overall user experience improvements. Your findings will play a crucial role in shaping the future direction of Instagram.

**SQL Tasks :**

**A) Marketing Analysis:**

1. **Loyal User Reward:** The marketing team wants to reward the most loyal users, i.e., those who have been using the platform for the longest time.  
   Your Task: Identify the five oldest users on Instagram from the provided database.

**Approach:** First we look for the tables we need for the solution, .i.e. here it is users table so from part is done . Then , oldest time means we can understand that we need to look for the time , which is represented by created\_at in users table. To complete the solution we select id , username, created\_at attributes from usera table and since we are want the 5 first users who registered on instagram we order them by ascending order and give the limit to 5 records.

**Query :** SELECT id, username, created\_at

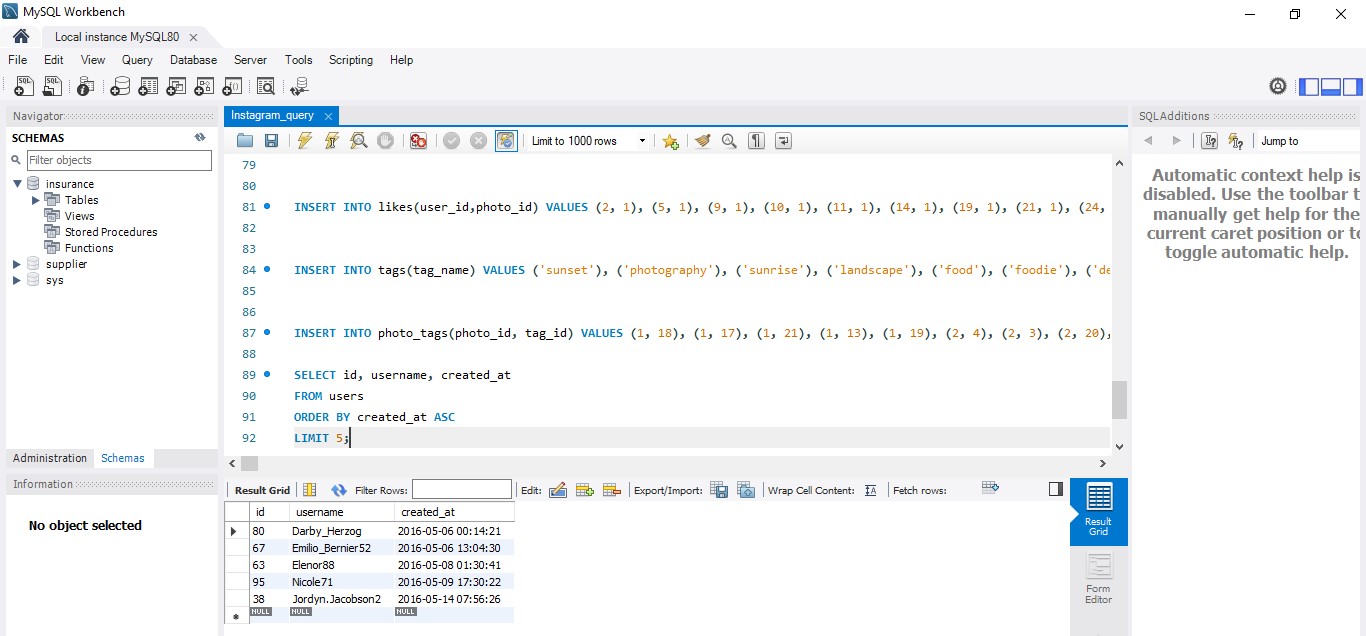
FROM users

ORDER BY created\_at ASC

LIMIT 5;

**Result :**

|  | **id** | **username** | **created\_at** |
| --- | --- | --- | --- |
|  | 38 | Jordyn.Jacobson2 | 2016-05-14 07:56:26 |
|  | 63 | Elenor88 | 2016-05-08 01:30:41 |
|  | 67 | Emilio\_Bernier52 | 2016-05-06 13:04:30 |
|  | 80 | Darby\_Herzog | 2016-05-06 00:14:21 |
|  | 95 | Nicole71 | 2016-05-09 17:30:22 |



1. **Inactive User Engagement:** The team wants to encourage inactive users to start posting by sending them promotional emails.  
   Your Task: Identify users who have never posted a single photo on Instagram.

**Approach:** Firstly tables involved – users and photo. Then we perform left join on the tables because we need the users data to be printed so we consider users as left circle and photos in right circle as in the venn diagram. We retrive the attributes id and username from users and join the tables based o the common tuple i.e user\_id and add a constraint is null on photos id as we want the users who never posted a picture.

**Query :** SELECT users.id, users.username

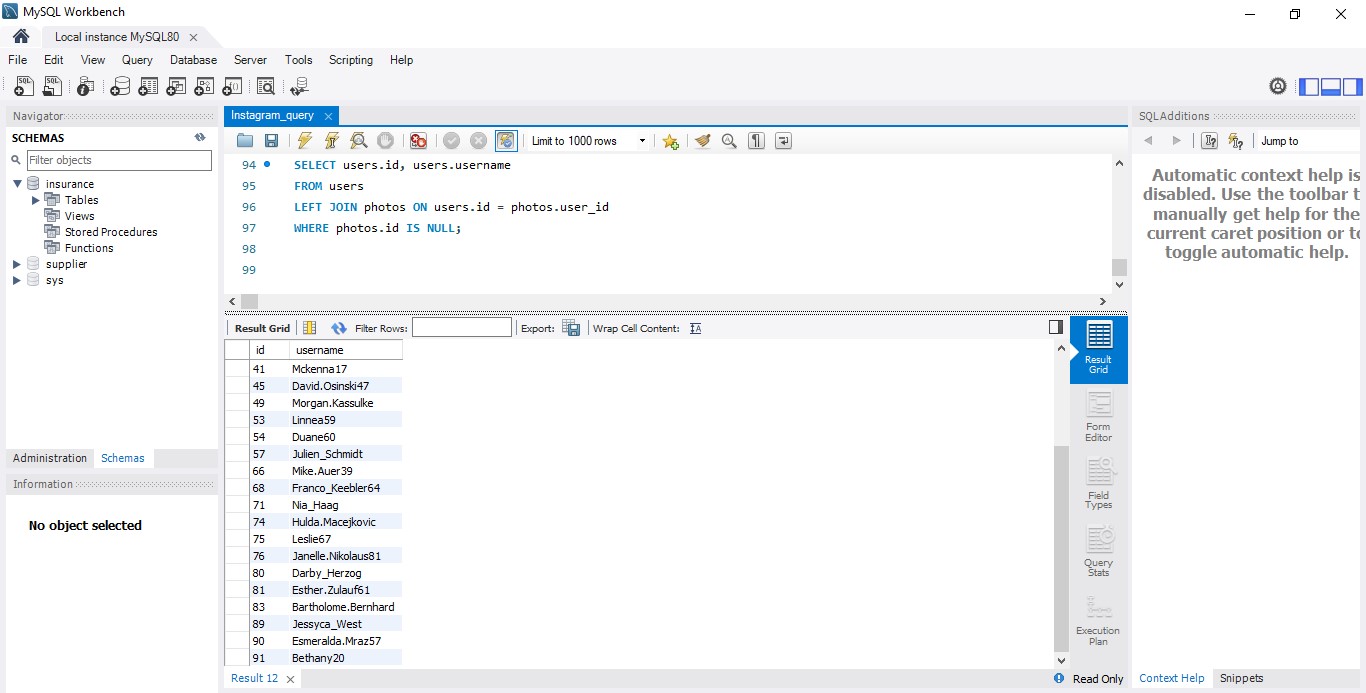
FROM users

LEFT JOIN photos ON users.id = photos.user\_id

WHERE photos.id IS NULL;

**Result :**

|  |  |
| --- | --- |
| **id** | **username** |
| 5 | Aniya\_Hackett |
| 83 | Bartholome.Bernhard |
| 91 | Bethany20 |
| 80 | Darby\_Herzog |
| 45 | David.Osinski47 |
| 54 | Duane60 |
| 90 | Esmeralda.Mraz57 |
| 81 | Esther.Zulauf61 |
| 68 | Franco\_Keebler64 |
| 74 | Hulda.Macejkovic |
| 14 | Jaclyn81 |
| 76 | Janelle.Nikolaus81 |
| 89 | Jessyca\_West |
| 57 | Julien\_Schmidt |
| 7 | Kasandra\_Homenick |
| 75 | Leslie67 |
| 53 | Linnea59 |
| 24 | Maxwell.Halvorson |
| 41 | Mckenna17 |
| 66 | Mike.Auer39 |
| 49 | Morgan.Kassulke |
| 71 | Nia\_Haag |
| 36 | Ollie\_Ledner37 |
| 34 | Pearl7 |
| 21 | Rocio33 |
| 25 | Tierra.Trantow |



1. **Contest Winner Declaration:** The team has organized a contest where the user with the most likes on a single photo wins.  
   Your Task: Determine the winner of the contest and provide their details to the team.

**Approach:** Firstly tables involved – users, likes and photo. We need to count the most likes on the single photo posted by the user and display their details , so we perform natural join on all 3 tables with commom tuple id . Then we select user\_id name , their photo id of single post which has the highest number of likes for which we use descending order and group by photo id. We also use alias to display the highest number of like as total likes.

**Query :** SELECT P.user\_id,U.username,P.id, COUNT(L.user\_id) AS total\_likes

FROM likes L

JOIN photos P ON L.photo\_id = P.id

JOIN users U ON P.user\_id = U.id

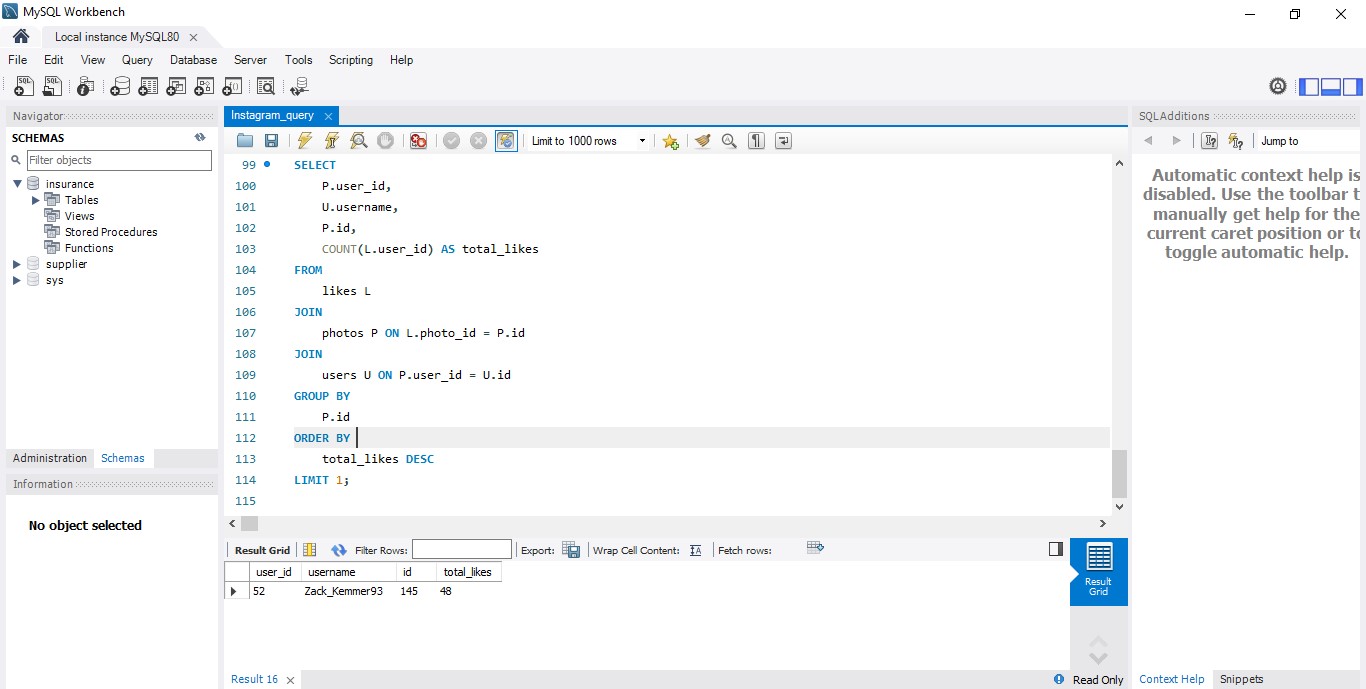
GROUP BY P.id

ORDER BY total\_likes DESC

LIMIT 1;

**Result :**

|  |  |  |  |
| --- | --- | --- | --- |
| **user\_id** | **username** | **id** | **total\_likes** |
| 52 | Zack\_Kemmer93 | 145 | 48 |



1. **Hashtag Research:** A partner brand wants to know the most popular hashtags to use in their posts to reach the most people.  
   Your Task: Identify and suggest the top five most commonly used hashtags on the platform.

**Approach:** Firstly tables involved – tags and photo tags. We select the attributes tagname from tags and count the most popular hashtags from their id and alias them as tag\_count. To do this we perform natural join of both tables based on common tuple id. Since we need highest 5 we order them by descending order and group them by tagname.

**Query :** SELECT t.tag\_name, COUNT(pt.tag\_id) AS tag\_count

FROM tags t

JOIN photo\_tags pt ON t.id = pt.tag\_id

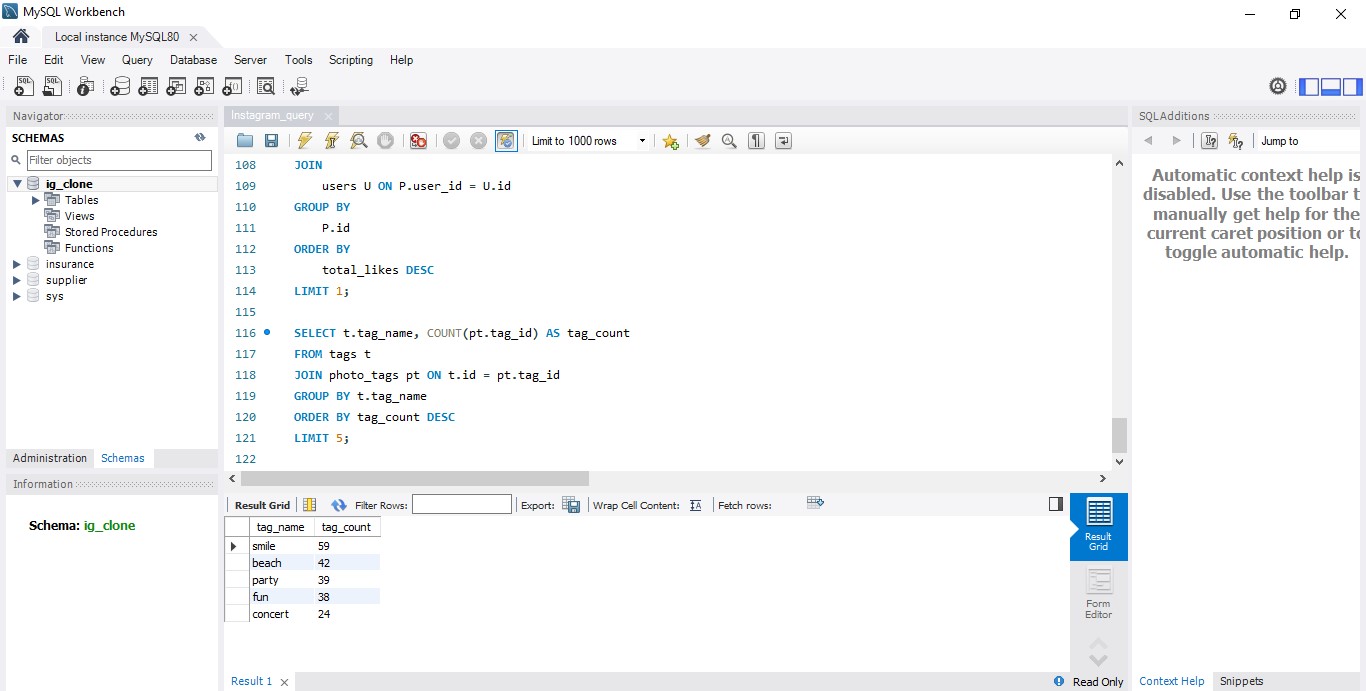
GROUP BY t.tag\_name

ORDER BY tag\_count DESC

LIMIT 5;

**Result :**

|  |  |
| --- | --- |
| **tag\_name** | **tag\_count** |
| smile | 59 |
| beach | 42 |
| party | 39 |
| fun | 38 |
| concert | 24 |



1. **Ad Campaign Launch:** The team wants to know the best day of the week to launch ads.  
   Your Task: Determine the day of the week when most users register on Instagram. Provide insights on when to schedule an ad campaign.

**Approach:** Firstly tables involved – users. We select the day of the week from the created\_at timestamp column from the users table using the dayname() function. Then we count the number of user registrations for each day of the week using count and \* and alias them as registration count and group them by results i.e the day of the week, and orders them in descending order based on registration count.

**Query :** SELECT DAYNAME(created\_at) AS registration\_day, COUNT(\*) AS registration\_count

FROM users

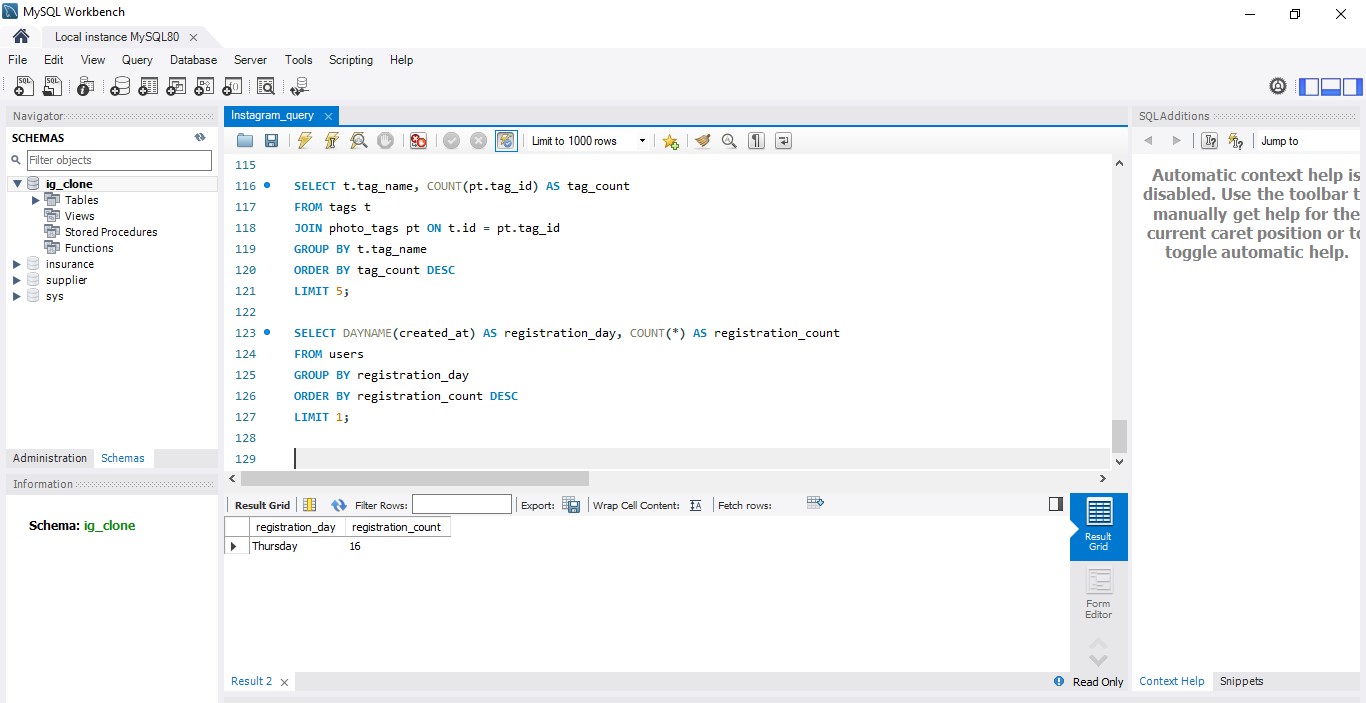
GROUP BY registration\_day

ORDER BY registration\_count DESC

LIMIT 1;

**Result :**

|  | **registration\_day** | **registration\_count** |
| --- | --- | --- |
|  | Thursday | 16 |



**B) Investor Metrics:**

1. **User Engagement:** Investors want to know if users are still active and posting on Instagram or if they are making fewer posts.  
   Your Task: 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.

**Approach:** Firstly tables involved for average - users and photos. We use subquery to calculate the avearge. We select the records from photos table and grop them by user id for this. In the outer query we take the results from inner query photos per user and use average aggregation function of those count.

Firstly tables involved for division - users and photos. We count the total photos using count \* and total users the same , give them an alias and perform simple division.

**Query :** SELECT AVG(post\_count) AS avg\_posts\_per\_user

FROM (SELECT COUNT(\*) AS post\_count

FROM photos

GROUP BY user\_id) AS user\_post\_counts;

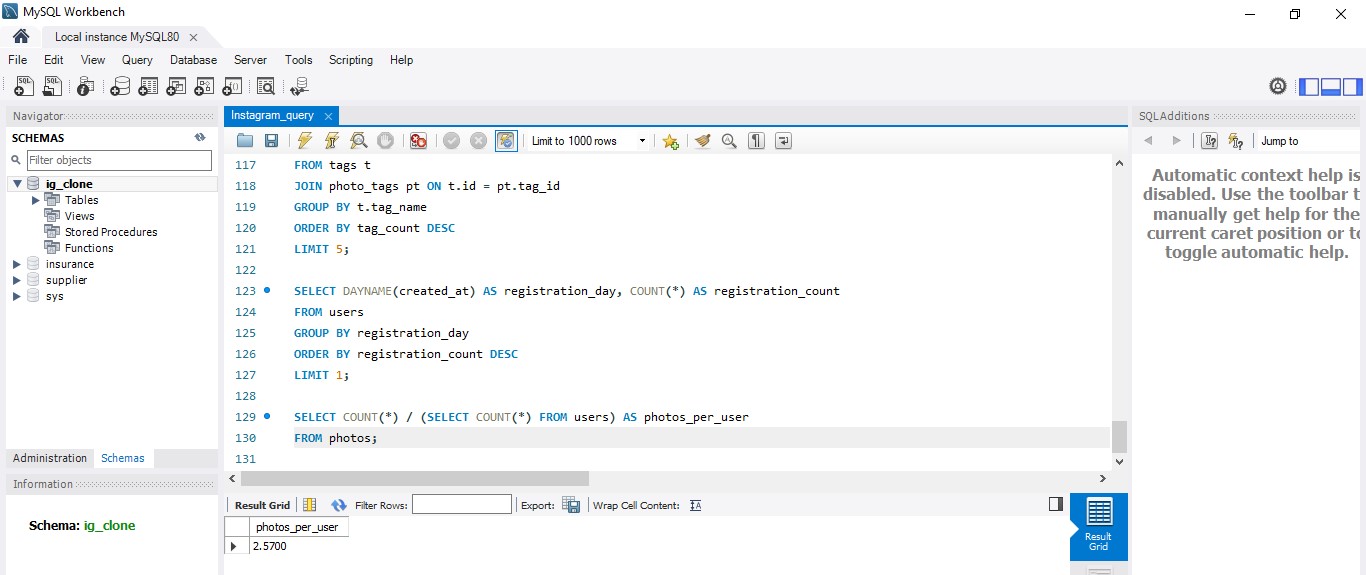
SELECT COUNT(\*) / (SELECT COUNT(\*) FROM users) AS photos\_per\_user

FROM photos;

**Result :**

|  | **avg\_posts\_per\_user** |
| --- | --- |
|  | 3.4730 |

|  | **photos\_per\_user** |
| --- | --- |
|  | 2.5700 |



1. **Bots & Fake Accounts:** Investors want to know if the platform is crowded with fake and dummy accounts.  
   Your Task: Identify users (potential bots) who have liked every single photo on the site, as this is not typically possible for a normal user.

**Approach:** Firstly tables involved- users and likes. We select the user id and count the number of distinct photoid values from likes table and group them by users. Then we add a condition by comparing the count of photos liked by each user with the total count of photos in the photos table.

**Query :** SELECT user\_id, COUNT(DISTINCT photo\_id) AS total\_photos\_liked

FROM likes

GROUP BY user\_id

HAVING total\_photos\_liked = (SELECT COUNT(\*) FROM photos);

**Result :**

|  | **user\_id** | **total\_photos\_liked** |
| --- | --- | --- |
|  | 5 | 257 |
|  | 14 | 257 |
|  | 21 | 257 |
|  | 24 | 257 |
|  | 36 | 257 |
|  | 41 | 257 |
|  | 54 | 257 |
|  | 57 | 257 |
|  | 66 | 257 |
|  | 71 | 257 |
|  | 75 | 257 |
|  | 76 | 257 |
|  | 91 | 257 |

