

# Instagram user analytics

## Project Description :

The "**Instagram User Analytics**" aims to analyze user behavior, engagement patterns, and content preferences on the Instagram platform. The purpose is to derive insights that can help businesses, marketers, and influencers understand their audience better and optimize their content strategy accordingly.

## Tech-Stack :

### MySQL Workbench: Version 8.0.26

MySQL Workbench was selected as the IDE for database development and administration. It provides a user-friendly interface for designing, modelling , and managing MySQL databases, along with tools for writing and executing SQL queries, making it an ideal choice for this project.

## Approach:

**Understanding Requirements:** Initially, I thoroughly reviewed the project requirements and objectives to gain a clear understanding of the goals and expectations.

**Data Collection:** Utilizing SQL queries, I extracted relevant data from the Instagram database, including user profiles, posts, likes, comments, and engagement metrics. This involved connecting to the MySQL database using MySQL Workbench and executing SELECT queries to retrieve the required data.

**User Segmentation:** Utilize SQL queries to segment users based on demographics, interests, and engagement levels, enabling the identification of distinct audience groups.

**Content Analysis:** Analyze content characteristics such as post types, captions, hashtags, and visual elements using SQL queries to understand what resonates most with the audience.

**Engagement Analysis:** Utilize SQL queries to examine factors influencing user engagement, such as posting frequency, timing, and interaction with followers, to optimize engagement strategies.

**Sentiment Analysis:** Perform sentiment analysis on comments using SQL queries to gauge audience sentiment and preferences.

## SQL Tasks :

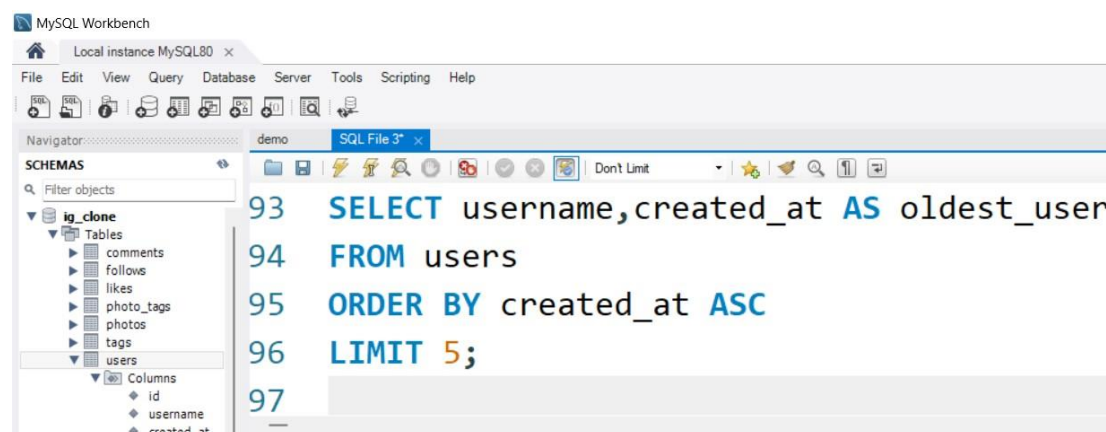
### A) Marketing Analysis:

#### 1. Loyal User Reward:

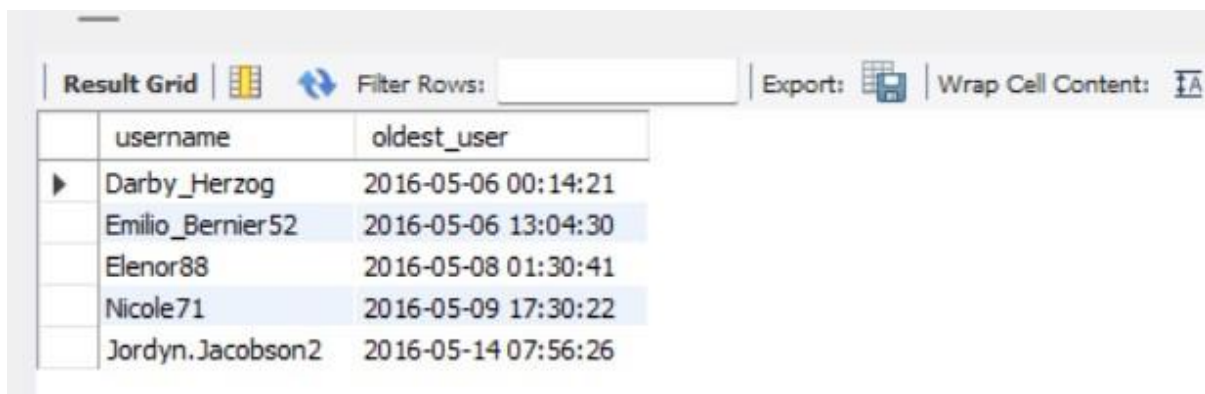
The marketing team wants to reward the most loyal users, i.e., those who have been using the platform for the longest time.

**Task:** Identify the five oldest users on Instagram from the provided database.

#### SQL Query:



## Output:



	username	oldest_user
▶	Darby_Herzog	2016-05-06 00:14:21
	Emilio_Bernier52	2016-05-06 13:04:30
	Elenor88	2016-05-08 01:30:41
	Nicole71	2016-05-09 17:30:22
	Jordyn.Jacobson2	2016-05-14 07:56:26

## Explanation:

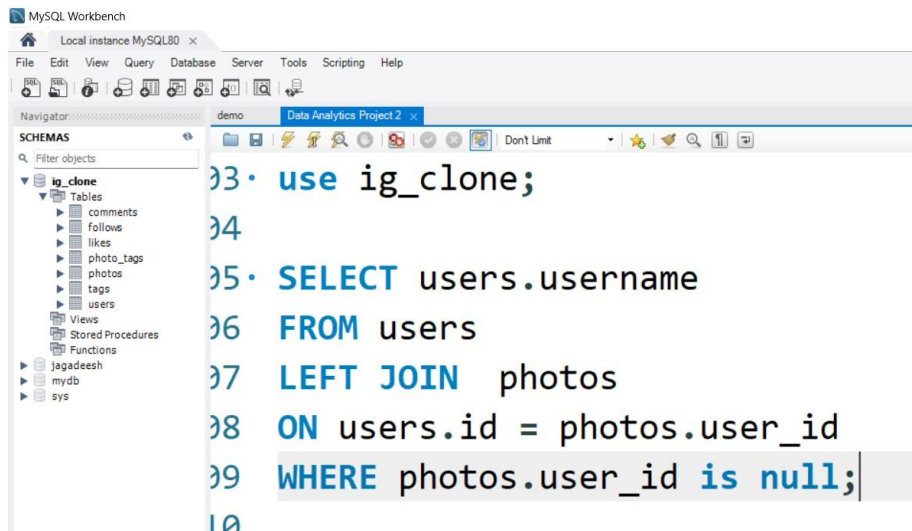
1. **SELECT:** This term indicates the columns selected for retrieval from the database. In this instance, we're choosing the username and **created\_at** columns from the **users table**.
2. **ALIASES:** This Term **AS** is used to change the column\_name as **Oldest\_user** to easily identify to the viewer
3. **FROM:** Denotes the source table from which data is being retrieved. Here, we're extracting data from the **users table**.
4. **ORDER BY:** This clause arranges the results based on a specified column. In this query, we're arranging the records based on the **created\_at** column.
5. **ASC:** It represents "**ascending**," indicating that the sorting order is from the smallest to the largest value. Consequently, the earliest users (those who registered first) will be displayed first in the sorted result set.
6. **LIMIT:** This clause restricts the number of rows returned by the query. In this case, we're constraining the result set to only **5 rows**, providing the details of the five oldest users.

## 2. Inactive User Engagement:

The team wants to encourage inactive users to start posting by sending them promotional emails.

**Task:** Identify users who have never posted a single photo on Instagram.

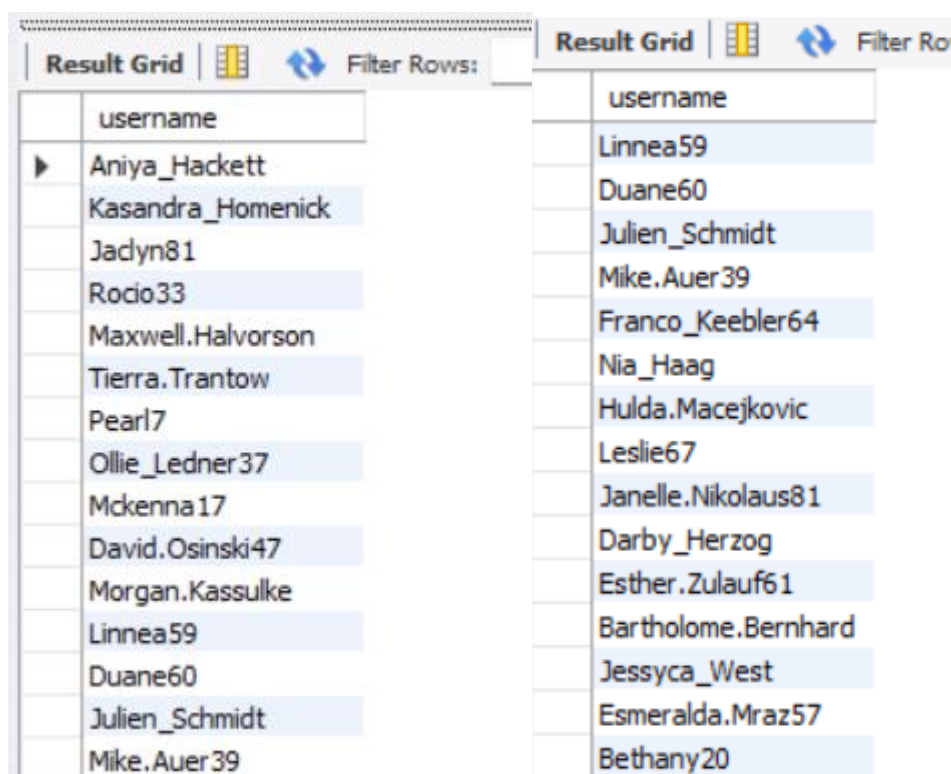
## SQL Query:



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view with 'ig\_clone' expanded, showing tables like 'comments', 'follows', 'likes', 'photo\_tags', 'photos', 'tags', and 'users'. The main query editor contains the following SQL code:

```
33. use ig_clone;
34
35. SELECT users.username
36 FROM users
37 LEFT JOIN photos
38 ON users.id = photos.user_id
39 WHERE photos.user_id is null;
40
```

## Output:



The screenshot shows the 'Result Grid' pane in MySQL Workbench. It displays the output of the SQL query, which is a list of usernames. The grid has two columns: 'username' and an empty column. The data is as follows:

username	
Aniya_Hackett	
Kassandra_Homenick	
Jadyn81	
Rocio33	
Maxwell.Halvorson	
Tierra.Trantow	
Pearl7	
Ollie_Ledner37	
Mckenna17	
David.Osinski47	
Morgan.Kassulke	
Linnea59	
Duane60	
Julien_Schmidt	
Mike.Auer39	

## Explanation:

1. **SELECT username:** This part of the query specifies that we want to retrieve the username column from the **users table**. We're interested in knowing the usernames of users who have never posted a single photo.

2. **FROM users:** Here, we specify that we're selecting data from the **users table**.
3. **LEFT JOIN photos ON users.id = photos.user\_id:** This line performs a left join operation between the users table and the photos table. A left join returns all rows from the left table (**users**), along with matching rows from the right table (**photos**). The join condition **ON users.id = photos.user\_id** ensures that rows are matched based on the id column in the **users table** and the **user\_id** column in the **photos table**.
4. **WHERE photos.user\_id IS NULL:** This part of the query filters the results to include only those rows where the **user\_id** column in the **photos table** is **NULL**. This condition ensures that we're selecting users who do not have any corresponding records in the photos table, indicating that they have never posted any photos.

### 3. Contest Winner Declaration:

The team has organized a contest where the user with the most likes on a single photo wins.

**Task:** Determine the winner of the contest and provide their details to the team.

#### SQL Query:

```
120
121 • SELECT u.id,u.username,u.created_at, COUNT(l.photo_id) AS total_likes
122 FROM users u
123 JOIN photos p ON u.id = p.user_id
124 LEFT JOIN likes l ON p.id = l.photo_id
125 GROUP BY u.id
126 ORDER BY total_likes DESC
127 LIMIT 1;
```

#### Output:

Result Grid					Filter Rows:	Export:	Wrap Cell Content:
	id	username	created_at	total_likes			
▶	23	Eveline95	2017-01-23 23:14:19	420			

### Explanation:

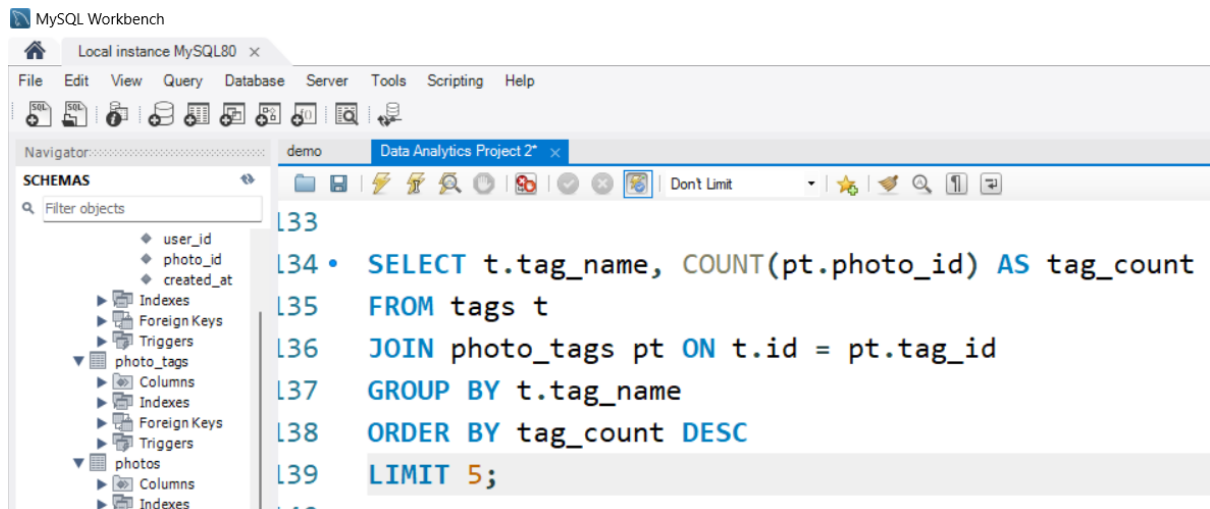
1. **SELECT** : u.username, p.image\_url, **COUNT**(l.user\_id) AS total\_likes: This part of the query selects three pieces of information. u.username the username of the user who uploaded the photo.p. image\_url The URL of the photo.
2. **COUNT**(l.user\_id) **AS** total\_likes: This counts the number of likes for each photo and gives it an alias **total\_likes**.
3. **FROM** users u **JOIN** photos p **ON** u.id = p.user\_id **JOIN** likes l **ON** p.id = l.photo\_id: This part specifies the tables being used and how they are joined together:
4. users u and photos p tables are joined on the condition that the id of the user (u.id) matches the user\_id of the photo (p.user\_id), linking each photo to its respective user.
5. Then, the photos p and likes l tables are joined based on the condition that the id of the photo (p.id) matches the photo\_id in the likes table (l.photo\_id), linking each like to its respective photo.
6. **GROUP BY** u.username, p.image\_url: This part groups the results by the username of the user and the URL of the photo. It ensures that each unique combination of user and photo is considered separately for the count of likes.
7. **ORDER BY** total\_likes **DESC**: This part orders the grouped results by the total number of likes in descending order. This means that the photo with the highest number of likes will appear first in the result set.
8. **LIMIT 1**: Finally, this limits the result to only one row, effectively giving us the user and photo with the highest number of likes, which is the winner of the contest.

### 4.Hashtag Research:

A partner brand wants to know the most popular hashtags to use in their posts to reach the most people.

**Task:** Identify and suggest the top five most commonly used hashtags on the platform.

## SQL Query:



## Output:

The screenshot shows the 'Result Grid' pane in MySQL Workbench, displaying the results of the SQL query. The results are shown in a table with two columns: 'tag\_name' and 'tag\_count'. The data is sorted in descending order of 'tag\_count'.

tag_name	tag_count
smile	59
beach	42
party	39
fun	38
concert	24

## Explanation:

1. **SELECT t.tag\_name, COUNT(pt.photo\_id) AS tag\_count:** This part of the query selects the tag names from the tags table and counts the number of occurrences of each tag by joining the tags table with the **photo\_tags** table and counting the number of **photo\_id** entries for each tag.
2. **FROM tags t JOIN photo\_tags pt ON t.id = pt.tag\_id:** This line specifies the tables being used and how they are joined together. We're joining the **tags table** (t) with the **photo\_tags table** (pt) based on the **id** of the **tag** in the tags table matching the **tag\_id** in the **photo\_tags table**.
3. **GROUP BY t.tag\_name:** This part groups the results by the tag names. It ensures that each unique tag is considered separately for the count.

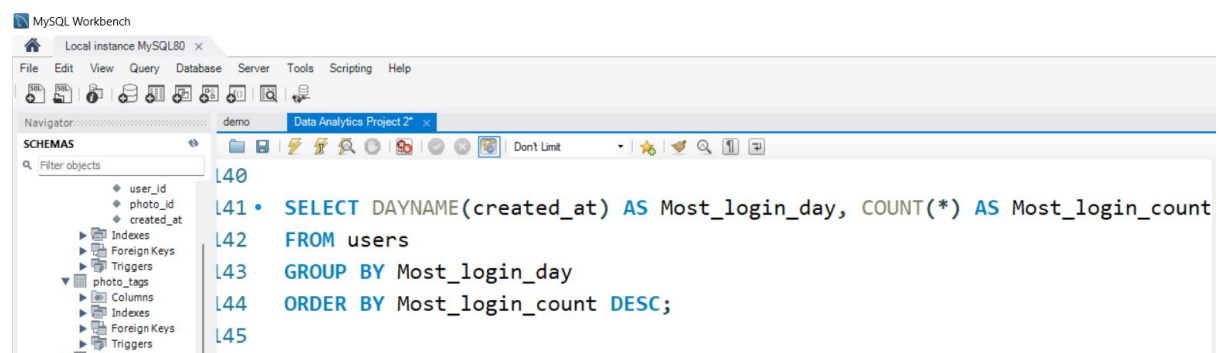
4. **ORDER BY tag\_count DESC:** This line orders the grouped results by the **tag count** in **descending order**. This means that the tag with the highest count will appear first in the result set.
5. **LIMIT 5:** Finally, this limits the result to only the **top five rows**, effectively giving us the **top five** most commonly used **hashtags** on the platform.

## 5. Ad Campaign Launch:

The team wants to know the best day of the week to launch ads.

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

### SQL Query:

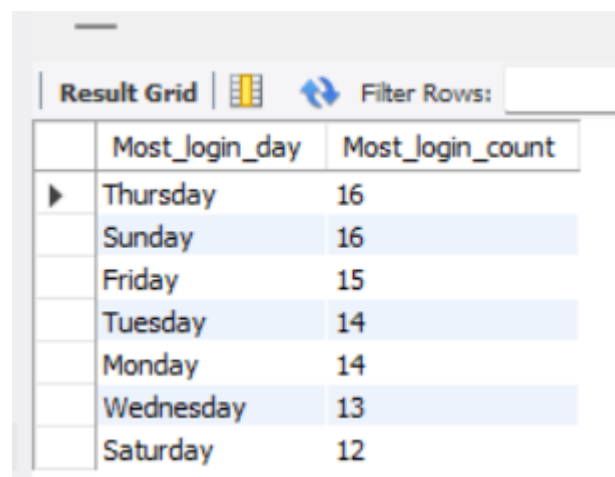


```

L40
L41 • SELECT DAYNAME(created_at) AS Most_login_day, COUNT(*) AS Most_login_count
L42 FROM users
L43 GROUP BY Most_login_day
L44 ORDER BY Most_login_count DESC;
L45

```

### Output:



	Most_login_day	Most_login_count
▶	Thursday	16
	Sunday	16
	Friday	15
	Tuesday	14
	Monday	14
	Wednesday	13
	Saturday	12



## Explanation:

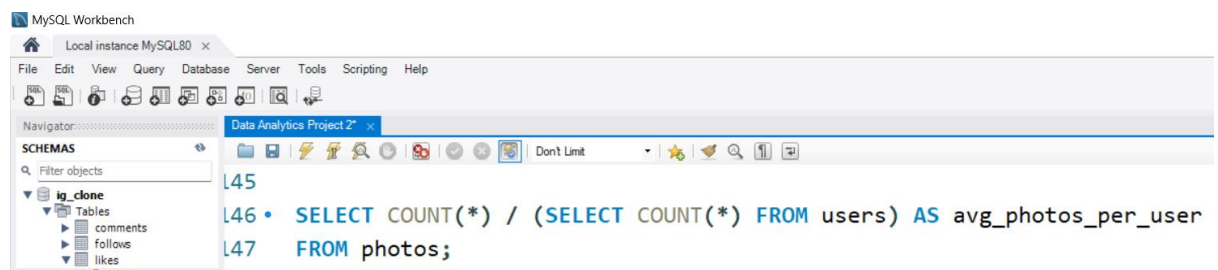
1. **SELECT DAYNAME(created\_at) AS Most\_login\_day, COUNT(\*) AS Most\_login\_count:** This part of the query selects the day of the week from the created\_at timestamp of user registrations and counts the number of registrations for each day of the week.
2. **FROM users:** Specifies the users table from which the data will be retrieved.
3. **GROUP BY Most\_login\_day:** This part groups the results by the day of the week (Most\_login\_day,). It ensures that each unique day of the week is considered separately for the count.
4. **ORDER BY Most\_login\_count DESC:** Orders the grouped results by the registration count in descending order. This means that the day with the highest number of **user Login** will appear first in the result set.

## B) Investor Metrics:

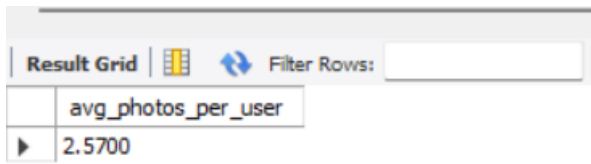
1. **User Engagement:** Investors want to know if users are still active and posting on Instagram or if they are making fewer posts.

**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.

## SQL Query:



## Output:



avg_photos_per_user
2.5700

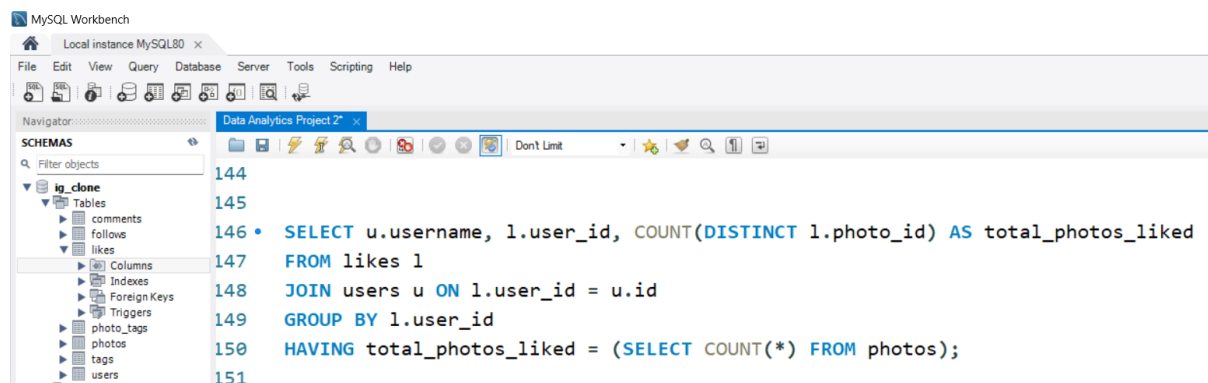
## Explanation:

1. **SELECT COUNT() / (SELECT COUNT() FROM users) AS avg\_photos\_per\_user:** This part of the query calculates the average number of photos per user. It divides the total count of photos by the total count of users. The result is aliased as **avg\_photos\_per\_user**.
2. **FROM photos:** Specifies the photos table from which the **total count of photos** will be retrieved.

**2.Bots & Fake Accounts:** Investors want to know if the platform is crowded with fake and dummy accounts.



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

## SQL Query:



```
144
145
146 • SELECT u.username, l.user_id, COUNT(DISTINCT l.photo_id) AS total_photos_liked
147 FROM likes l
148 JOIN users u ON l.user_id = u.id
149 GROUP BY l.user_id
150 HAVING total_photos_liked = (SELECT COUNT(*) FROM photos);
151
```

## Output:

Result Grid     Filter Rows: <input type="text"/>			
	username	user_id	total_photos_liked
▶	Aniya_Hackett	5	257
	Jacyln81	14	257
	Rocio33	21	257
	Maxwell.Halvorson	24	257
	Ollie_Ledner37	36	257
	Mckenna17	41	257
	Duane60	54	257
	Julien_Schmidt	57	257
	Mike.Auer39	66	257
	Nia_Haag	71	257
	Leslie67	75	257
	Janelle.Nikolaus81	76	257
	Bethany20	91	257

## Explanation:

1. **SELECT u.username, l.user\_id, COUNT(DISTINCT l.photo\_id) AS total\_photos\_liked:** This part of the query selects the username from the users table, along with the **user\_id** from the likes table. It also counts the number of distinct **photo\_id** values for each user, representing the total number of photos liked by each user.
2. **FROM likes l:** Specifies the likes table from which the data will be retrieved.
3. **JOIN users u ON l.user\_id = u.id:** Joins the likes table (l) with the users table (u) based on the **user\_id** column. This allows us to retrieve the username associated with each user ID.
4. **GROUP BY l.user\_id:** Groups the results by the **user\_id**. It ensures that each user's likes are considered separately.
5. **HAVING total\_photos\_liked = (SELECT COUNT(\*) FROM photos):** This condition filters the results to include only users whose total number of liked photos (**total\_photos\_liked**) is equal to the total number of photos on the site. The subquery (**SELECT COUNT(\*) FROM photos**) calculates the total count of photos in the photos table.

## Result:

1. **Data Analysis Proficiency:** Gained proficiency in SQL query writing and data analysis techniques, allowing for the extraction of valuable insights from the database.
2. **Insight Generation:** Identified potential bot behavior by analyzing user interactions such as likes, helping to maintain the integrity and authenticity of the platform.
3. **Database Understanding:** Deepened understanding of database schema design and relational database concepts through practical application and query formulation.
4. **Strategic Decision Making:** Provided actionable insights for strategic decision-making processes, such as scheduling ad campaigns based on user registration patterns and identifying potential bots.
5. **Enhanced Problem-Solving Skills:** Developed problem-solving skills by addressing specific challenges related to data analysis and user behavior identification.

## The impact of the analysis and insights derived from the project is significant:

1. **Improved Platform Integrity:** Identification of potential bots helps in maintaining the authenticity and credibility of the platform, fostering a positive user experience and trust among users.
2. **Informed Marketing Strategies:** Insights into user registration patterns enable more effective scheduling of ad campaigns, maximizing reach and engagement with potential users.
3. **Enhanced User Engagement:** By understanding user behavior and preferences, the platform can tailor its features and offerings to better meet user needs, ultimately driving increased user engagement and retention.

Overall, the analysis and insights derived from the project provide valuable information for decision-making processes, contributing to the overall success and growth of the platform.