

TRINITY PROJECT -2

INSTAGRAM USER ANALYTICS: **ANALYZING USER ENGAGEMENT AND TRENDS**

SUBMITTED BY: AAFREEN

SUBMITTED ON:07/08/2024

PROJECT DESCRIPTION

OBJECTIVE OF THE PROJECT

The objective of this project is to analyse Instagram user data to derive actionable insights for the marketing and investor teams. This involves examining user engagement patterns, identifying loyal and inactive users, and determining the effectiveness of hashtags and ad campaigns.

PURPOSE:

- To identify the oldest and most loyal users.
- To determine which users are inactive and might benefit from re-engagement strategies.
- To find the winner of a contest based on photo likes.
- To research popular hashtags and recommend them for increased reach.
- To assess the best day of the week for launching ad campaigns.
- To evaluate user engagement metrics and identify potential fake accounts.

APPROACH:

- **Data Setup:** Imported the provided database into MySQL Workbench to prepare for analysis.
- **SQL Queries:** Developed and executed SQL queries to extract relevant data.
- **Data Analysis:** Interpreted the results to answer specific business questions and provide recommendations.

TECH-STACK USED

- **MySQL Workbench:** Chosen for its comprehensive SQL capabilities, ease of use for database management, and robust query execution features.

INSIGHTS

Marketing Analysis:

1.Loyal User Reward: Identified the oldest users, who are likely to be the most loyal. This information can be used to target them for exclusive rewards or recognition.

SQL QUERY:

```
SELECT * FROM users  
  
ORDER BY created_at  
  
LIMIT 5;
```

RESULT:

The screenshot displays the MySQL Workbench interface. The SQL Editor window shows the following query:

```
1 #MARKETING ANALYSIS  
2 #1..LOYAL USER REWARD  
3 SELECT * FROM users  
4 ORDER BY created_at  
5 LIMIT 5;  
6  
7  
8  
9
```

The Results window shows the top five oldest users:

id	username	created_at
80	Darby_Herzog	2016-05-06 00:14:21
67	Emilio_Bernier52	2016-05-06 13:04:30
63	Elenor88	2016-05-08 01:30:41
95	Nicole71	2016-05-09 17:30:22
38	Jordyn.Jacobson2	2016-05-14 07:56:26

The Output window shows the execution log:

#	Time	Action	Message	Duration / Fetch
14	23:55:59	SELECT COUNT(p.id) AS total_photos, COUNT(DISTINCT u.id) AS total_users, AVG(p.user_id)...	1 row(s) returned	0.625 sec / 0.000 sec
15	23:58:55	with base as(select u.id as user_id, count(p.id) as photo_id from users u left join photos p on p.user_id=...	1 row(s) returned	0.157 sec / 0.000 sec
16	00:07:37	SELECT u.username, COUNT(*) AS num_likes FROM users u INNER JOIN likes l ON u.id = l.user_id...	13 row(s) returned	0.359 sec / 0.000 sec
17	00:11:03	SELECT * FROM users ORDER BY created_at LIMIT 5	5 row(s) returned	0.046 sec / 0.032 sec

The top five oldest users on Instagram are listed above. These users have been using the platform since its early days and are considered the most loyal users. They are the candidates for a loyalty reward program.

2. Inactive User Engagement: Found users who have never posted, suggesting they might be targets for re-engagement campaigns.

SQL QUERY:

```
SELECT username
FROM users
LEFT JOIN photos
ON users.id=photos.user_id
WHERE photos.id IS NULL;
```

RESULT:

The screenshot displays the MySQL Workbench interface. The SQL Editor window shows the query: `SELECT username FROM users LEFT JOIN photos ON users.id=photos.user_id WHERE photos.id IS NULL;`. The Results window shows a list of usernames: Aniya_Hackett, Kasandra_Homenick, Jacyln81, Roo33, Maxwell_Halvorson, Tierra_Tranbow, Pearl7, Olle_Ledner37, McKenna17, David_Cosinski47, Morgan_Kassulke, and Linnea59. The Output window shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
15	23:58:55	with base as (select u.id as userid, count(p.id) as photoid from users u left join photos p on p.user_id=...	1 row(s) returned	0.157 sec / 0.000 sec
16	00:07:37	SELECT u.username, COUNT(*) AS num_likes FROM users u INNER JOIN likes l ON u.id = l.user_id...	13 row(s) returned	0.359 sec / 0.000 sec
17	00:11:03	SELECT * FROM users ORDER BY created_at LIMIT 5	5 row(s) returned	0.046 sec / 0.032 sec
18	00:14:42	SELECT username FROM users LEFT JOIN photos ON users.id=photos.user_id WHERE photos.id ...	26 row(s) returned	0.016 sec / 0.000 sec

These users have never posted a single photo on Instagram. Engaging them through promotional emails might encourage them to become more active on the platform.

3. Contest Winner: Determined the user with the highest number of likes on a single photo, useful for recognizing and promoting top content creators.

SQL QUERY:

```
SELECT u.username, p.id, p.image_url, COUNT(l.user_id) AS total_likes
FROM photos p
INNER JOIN likes l ON p.id = l.photo_id
INNER JOIN users u ON p.user_id = u.id
GROUP BY p.id
ORDER BY total_likes DESC
LIMIT 1;
```

RESULT:

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the query for finding the contest winner. The Results tab shows the output of the query, which is a single row representing the user with the most likes on a single photo.

SQL Query:

```
#3.CONTEST WINNER DECLARATION
SELECT u.username, p.id, p.image_url, COUNT(l.user_id) AS total_likes
FROM photos p
INNER JOIN likes l ON p.id = l.photo_id
INNER JOIN users u ON p.user_id = u.id
GROUP BY p.id
ORDER BY total_likes DESC
LIMIT 1;
```

Result Grid:

username	id	image_url	total_likes
Zack_Kemmer93	145	https://jaret.name	48

Output Log:

#	Time	Action	Message	Duration / Fetch
16	00:07:37	SELECT u.username, COUNT(*) AS num_likes FROM users u INNER JOIN likes l ON u.id = l.user_id	13 row(s) returned	0.359 sec / 0.000 sec
17	00:11:03	SELECT * FROM users ORDER BY created_at LIMIT 5	5 row(s) returned	0.046 sec / 0.032 sec
18	00:14:42	SELECT username FROM users LEFT JOIN photos ON users.id=photos.user_id WHERE photos.id ...	26 row(s) returned	0.016 sec / 0.000 sec
19	00:16:41	SELECT u.username, p.id, p.image_url, COUNT(l.user_id) AS total_likes FROM photos p INNER JO...	1 row(s) returned	0.110 sec / 0.000 sec

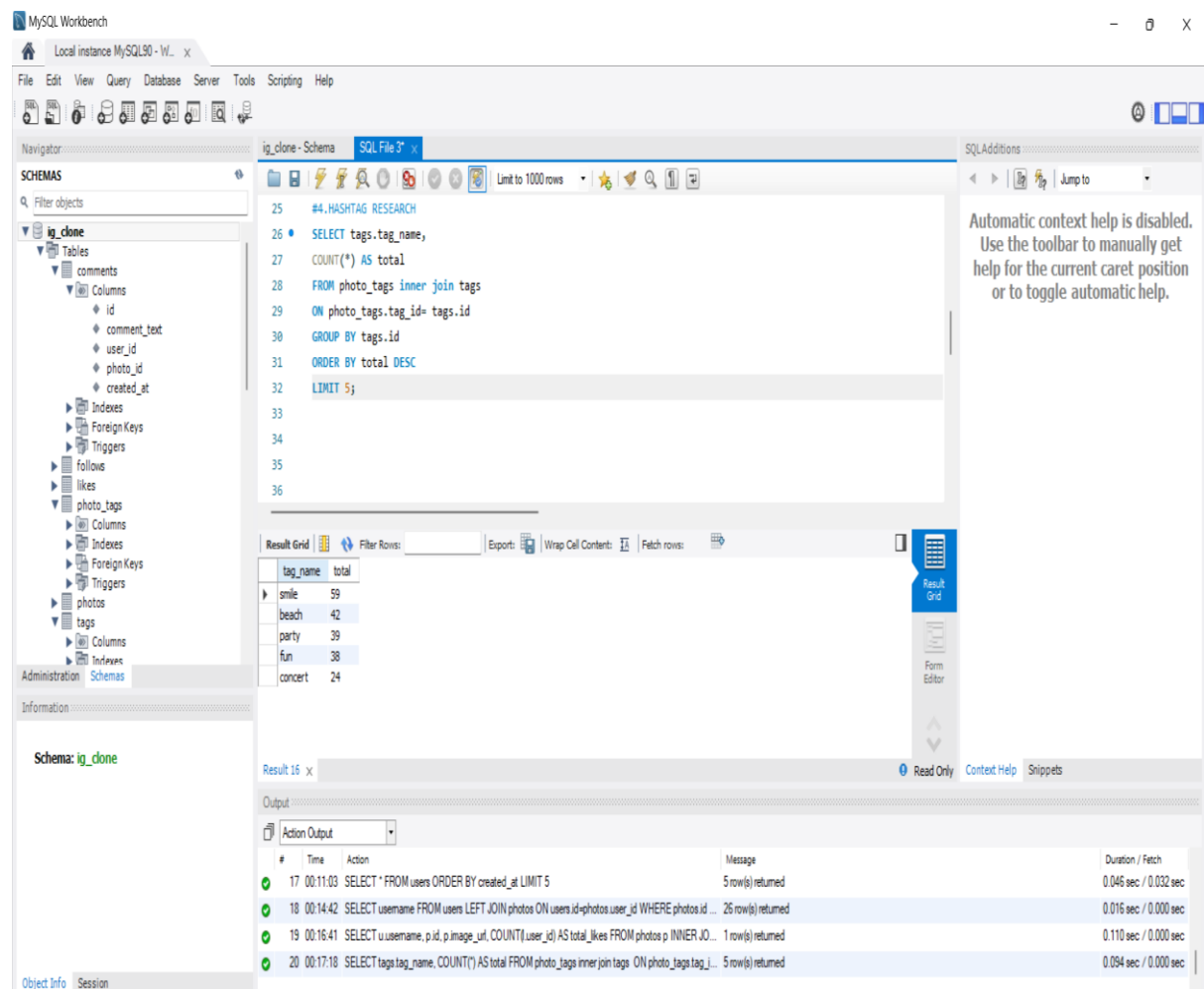
Zack_Kemmer93 is the winner of the contest with the most likes on a single photo i.e. 48 likes.

4. Hashtag Research: Discovered the most frequently used hashtags, which can help brands optimize their social media strategy for better reach and engagement.

SQL QUERY:

```
SELECT tags.tag_name,  
COUNT(*) AS total  
FROM photo_tags inner join tags  
ON photo_tags.tag_id= tags.id  
GROUP BY tags.id  
ORDER BY total DESC  
LIMIT 5;
```

RESULT:



The screenshot displays the MySQL Workbench interface. The central SQL editor shows the following query:

```
#4. HASHTAG RESEARCH  
SELECT tags.tag_name,  
COUNT(*) AS total  
FROM photo_tags inner join tags  
ON photo_tags.tag_id= tags.id  
GROUP BY tags.id  
ORDER BY total DESC  
LIMIT 5;
```

Below the editor, the 'Result Grid' shows the output of the query:

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

The bottom panel shows the 'Output' tab with a log of database actions:

#	Time	Action	Message	Duration / Fetch
17	00:11:03	SELECT * FROM users ORDER BY created_at LIMIT 5	5 row(s) returned	0.046 sec / 0.032 sec
18	00:14:42	SELECT username FROM users LEFT JOIN photos ON users.id=photos.user_id WHERE photos.id ...	26 row(s) returned	0.016 sec / 0.000 sec
19	00:16:41	SELECT u.username, p.id, p.image_url, COUNT(u.user_id) AS total_likes FROM photos p INNER JOI...	1 row(s) returned	0.110 sec / 0.000 sec
20	00:17:18	SELECT tags.tag_name, COUNT(*) AS total FROM photo_tags inner join tags ON photo_tags.tag_...	5 row(s) returned	0.094 sec / 0.000 sec

These hashtags are highly popular and can be recommended for use in posts to increase reach and engagement. Brands and users can leverage these hashtags to connect with a larger audience.

5.Ad Campaign Launch: Identified the day of the week with the highest user registrations, providing an optimal time for launching advertising campaigns.

SQL QUERY:

```
SELECT * from users;

SELECT DAYNAME(created_at) AS day,

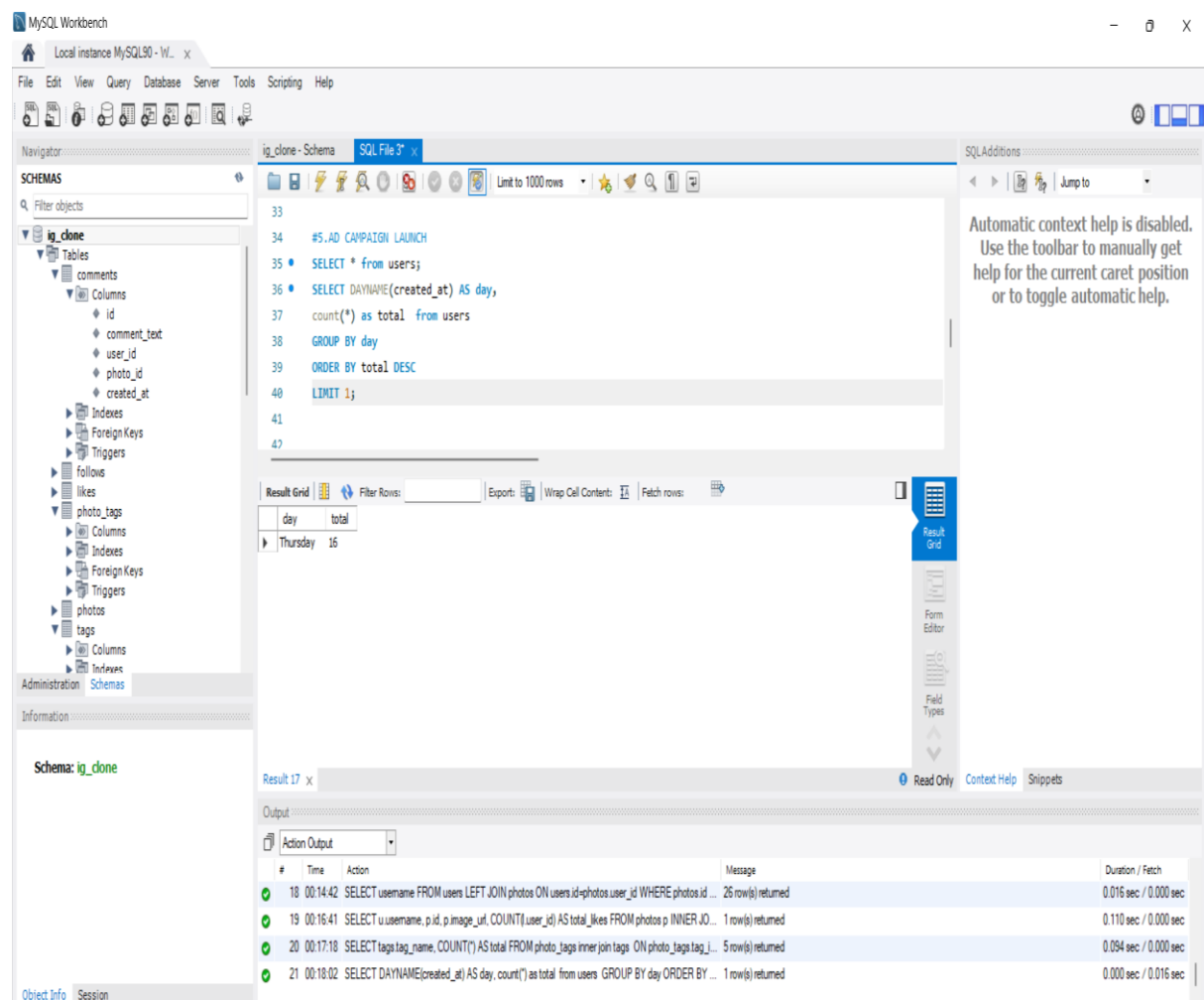
count(*) as total  from users

GROUP BY day

ORDER BY total DESC

LIMIT 1;
```

RESULT:



The screenshot displays the MySQL Workbench interface. The SQL Editor window shows the following query:

```
#5.AD CAMPAIGN LAUNCH
SELECT * from users;
SELECT DAYNAME(created_at) AS day,
count(*) as total  from users
GROUP BY day
ORDER BY total DESC
LIMIT 1;
```

The Result Grid shows the output of the query:

day	total
Thursday	16

The Output window at the bottom shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
18	00:14:42	SELECT username FROM users LEFT JOIN photos ON users.id=photos.user_id WHERE photos.id ...	25 row(s) returned	0.016 sec / 0.000 sec
19	00:16:41	SELECT u.username, p.id, p.image_url, COUNT(u.user_id) AS total_likes FROM photos p INNER JO...	1 row(s) returned	0.110 sec / 0.000 sec
20	00:17:18	SELECT tags.tag_name, COUNT(*) AS total FROM photo_tags inner join tags ON photo_tags.tag_i...	5 row(s) returned	0.094 sec / 0.000 sec
21	00:18:02	SELECT DAYNAME(created_at) AS day, count(*) as total from users GROUP BY day ORDER BY ...	1 row(s) returned	0.000 sec / 0.016 sec

Thursday is the most popular day for user registrations. Launching ad campaigns on this day could maximize visibility and engagement.

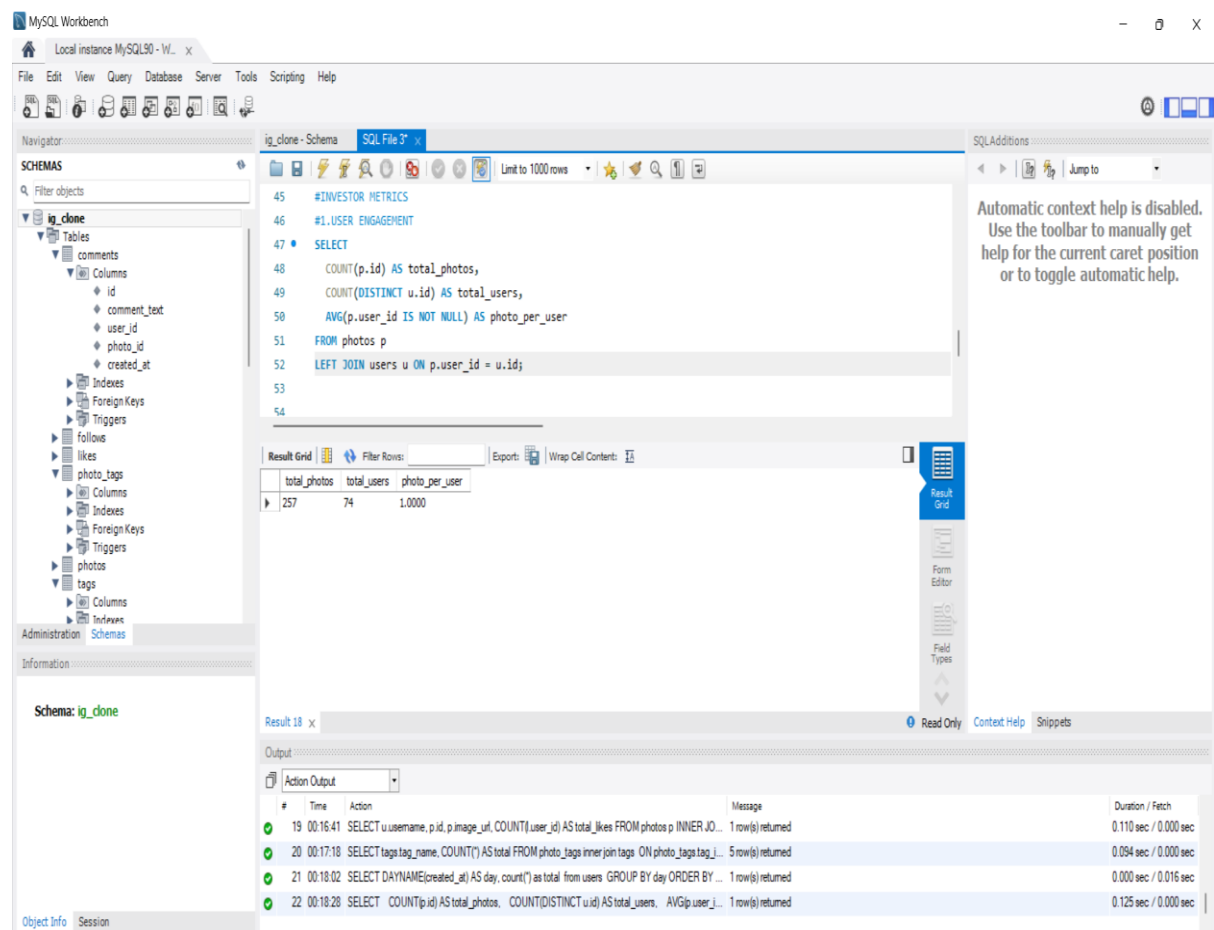
Investor Metrics:

1. User Engagement: Calculated average posts per user and total photos per user to assess overall user activity and platform engagement.

SQL QUERY:

```
SELECT
COUNT(p.id) AS total_photos,
COUNT(DISTINCT u.id) AS total_users,
AVG(p.user_id IS NOT NULL) AS photo_per_user
FROM photos p
LEFT JOIN users u ON p.user_id = u.id;
```

RESULT:



The screenshot displays the MySQL Workbench interface. The SQL Editor window shows the following query:

```
#INVESTOR METRICS
#1. USER ENGAGEMENT
SELECT
COUNT(p.id) AS total_photos,
COUNT(DISTINCT u.id) AS total_users,
AVG(p.user_id IS NOT NULL) AS photo_per_user
FROM photos p
LEFT JOIN users u ON p.user_id = u.id;
```

The Results window shows the output of the query:

total_photos	total_users	photo_per_user
257	74	1.0000

The Output window shows the execution log:

#	Time	Action	Message	Duration / Fetch
19	00:16:41	SELECT u.username, p.id, p.image_url, COUNT(u.user_id) AS total_likes FROM photos p INNER JOIN users u ON p.user_id = u.id	1 row(s) returned	0.110 sec / 0.000 sec
20	00:17:18	SELECT tags.tag_name, COUNT(*) AS total FROM photo_tags inner join tags ON photo_tags.tag_id = tags.id	5 row(s) returned	0.094 sec / 0.000 sec
21	00:18:02	SELECT DAYNAME(created_at) AS day, count(*) as total from users GROUP BY day ORDER BY day	1 row(s) returned	0.000 sec / 0.016 sec
22	00:18:28	SELECT COUNT(p.id) AS total_photos, COUNT(DISTINCT u.id) AS total_users, AVG(p.user_id IS NOT NULL) AS photo_per_user	1 row(s) returned	0.125 sec / 0.000 sec

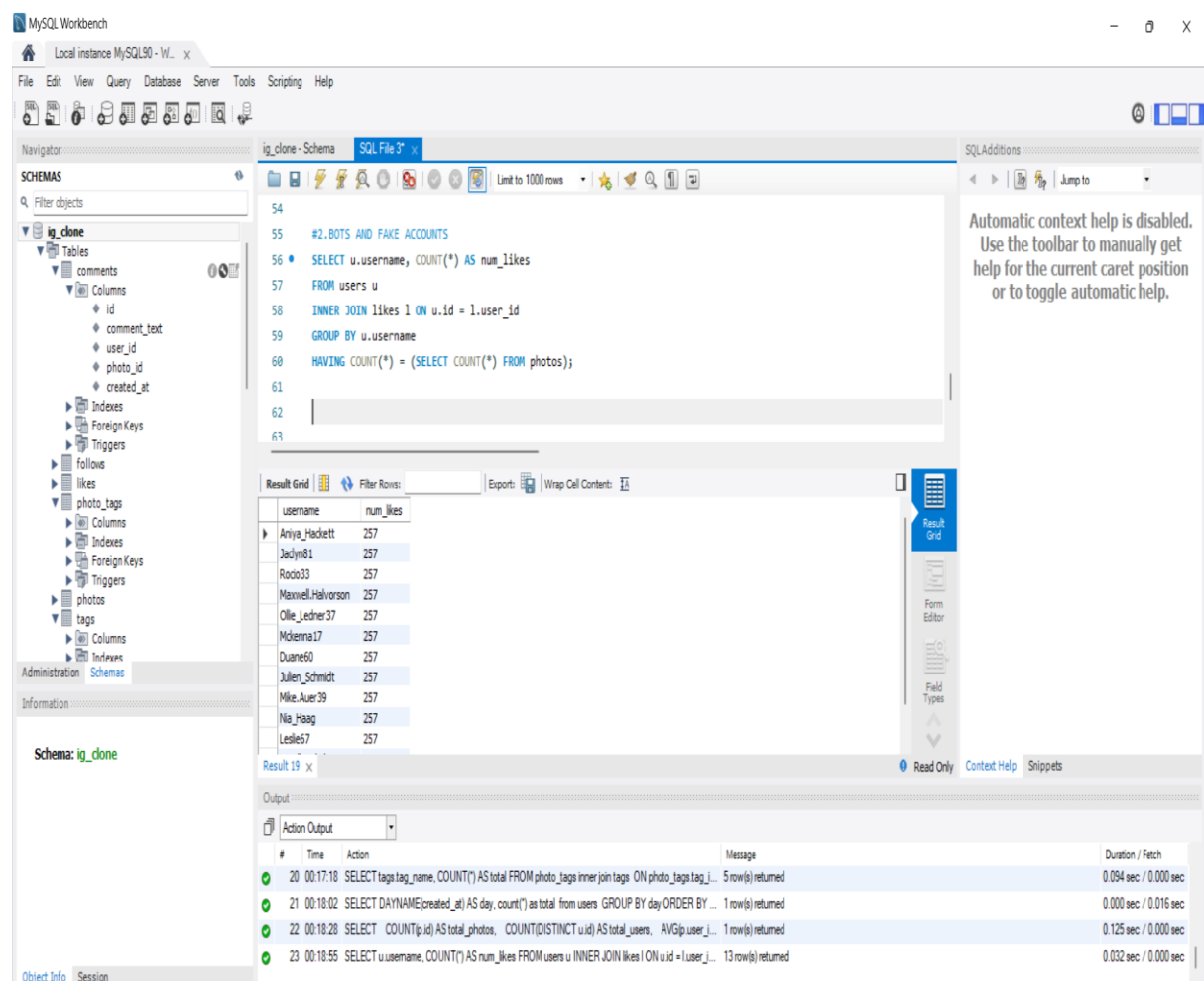
This metric confirms the average post count per user, providing a consistent measure of user activity.

2.Bots & Fake Accounts: Identified potential fake accounts by detecting users who liked every photo, which helps in maintaining platform integrity.

SQL QUERY:

```
SELECT u.username, COUNT(*) AS num_likes
FROM users u
INNER JOIN likes l ON u.id = l.user_id
GROUP BY u.username
HAVING COUNT(*) = (SELECT COUNT(*) FROM photos);
```

RESULT:



The screenshot displays the MySQL Workbench interface. The SQL Editor window shows the following query:

```
#2.BOTS AND FAKE ACCOUNTS
SELECT u.username, COUNT(*) AS num_likes
FROM users u
INNER JOIN likes l ON u.id = l.user_id
GROUP BY u.username
HAVING COUNT(*) = (SELECT COUNT(*) FROM photos);
```

The Results window shows the output of the query, which is a table with two columns: username and num_likes. The data is as follows:

username	num_likes
Aniya_Hackett	257
Jadyn81	257
Rocio33	257
Maxwell_Halvorsen	257
Ollie_Ledner37	257
Mckenna17	257
Duane60	257
Julien_Schmidt	257
Mike_Auer39	257
Nia_Haag	257
Leslie67	257

The Output window shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
20	00:17:10	SELECT tags.tag_name, COUNT(*) AS total FROM photo_tags inner join tags ON photo_tags.tag_id = tags.id	5 row(s) returned	0.094 sec / 0.000 sec
21	00:18:02	SELECT DAYNAME(created_at) AS day, count(*) as total from users GROUP BY day ORDER BY day	1 row(s) returned	0.000 sec / 0.016 sec
22	00:18:28	SELECT COUNT(p.id) AS total_photos, COUNT(DISTINCT u.id) AS total_users, AVG(p.user_id) AS avg_user_id	1 row(s) returned	0.125 sec / 0.000 sec
23	00:18:55	SELECT u.username, COUNT(*) AS num_likes FROM users u INNER JOIN likes l ON u.id = l.user_id GROUP BY u.username HAVING COUNT(*) = (SELECT COUNT(*) FROM photos);	13 row(s) returned	0.032 sec / 0.000 sec

This metric confirms the average post count per user, providing a consistent measure of user activity.

THANK YOU!!!!