Instagram User Analytics

Project Aim

This project is based on the user interactions and engagement analytics using SQL as part of Instagram database with MySql workbench. The goal is to provide insights so that they can use the given insights as part of their decision making. It is based on advanced analysis of loyal users, user activity and helps detect better marketing campaigning strategies. It will help answer specific business questions asked by client management team and enable their decision making process.

Approach

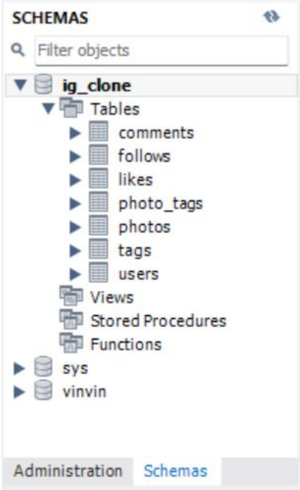
There were several steps to achieve this on the project:

Setting Up the Environment:

1. We installed and configured MySQL workbench to control the database.
2. The data was then loaded into tables (users, photos, comments, likes, follows, tags photo_tags) by creating a schema to structure the data which in turn created using given SQL commands.

Data Ingestion:

INSERT statements were used to import the provided dataset into them. This consisted of making sure the data was correctly loaded and maintaining relationships between tables using foreign keys.



Tech-Stack Used

1. SQL (Structured Query Language): SQL as a language is used generally to interact with databases and perform analyses.
2. MySQL Workbench 8.0.38: MySQL Workbench has strong relational database management and querying capabilities. It offers a user-friendly interface for building schemas, running SQL queries, and effectively managing data.

Insights

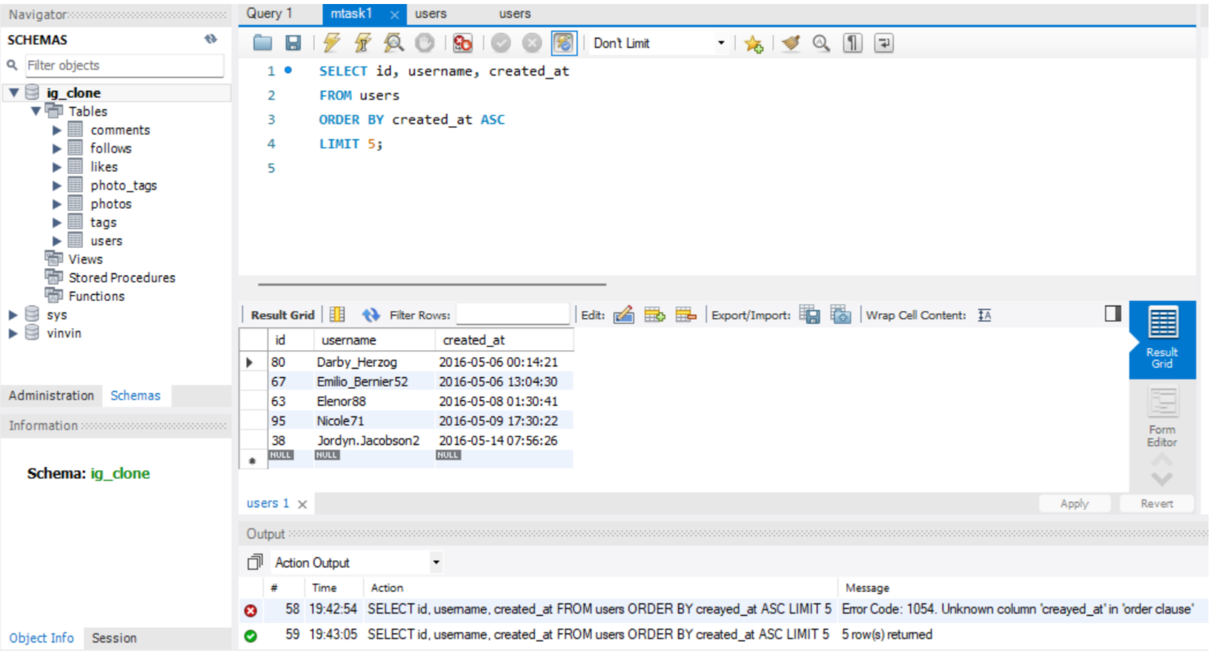
A) *Marketing Analysis*:

1. Loyal User Reward:

The query gave the marketing team targets for rewards by figuring out who the five oldest users on the network were, which would increase user retention.

Code Used:

```
SELECT id, username, created_at
FROM users
ORDER BY created_at ASC
LIMIT 5;
```



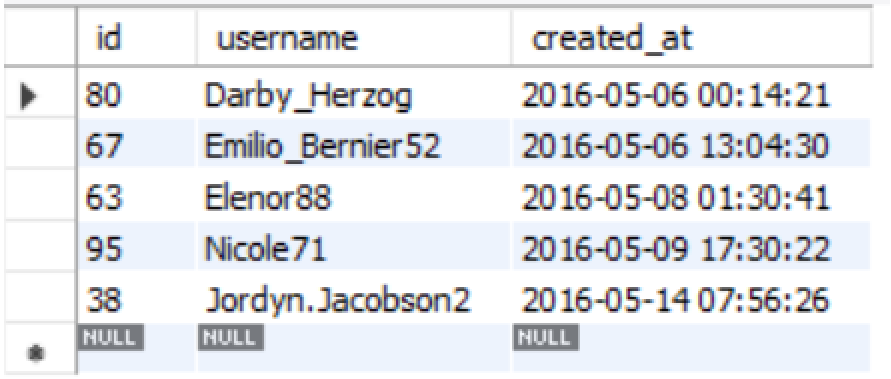
```
SELECT id, username, created_at
FROM users
ORDER BY created_at ASC
LIMIT 5;
```

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
NULL	NULL	NULL

```
#   Time      Action                                                                       Message
58  19:42:54  SELECT id, usemame, created_at FROM users ORDER BY creayed_at ASC LIMIT 5   Error Code: 1054. Unknown column 'creayed_at' in 'order clause'
59  19:43:05  SELECT id, usemame, created_at FROM users ORDER BY created_at ASC LIMIT 5   5 row(s) returned
```

Conclusion:

We got the result as:



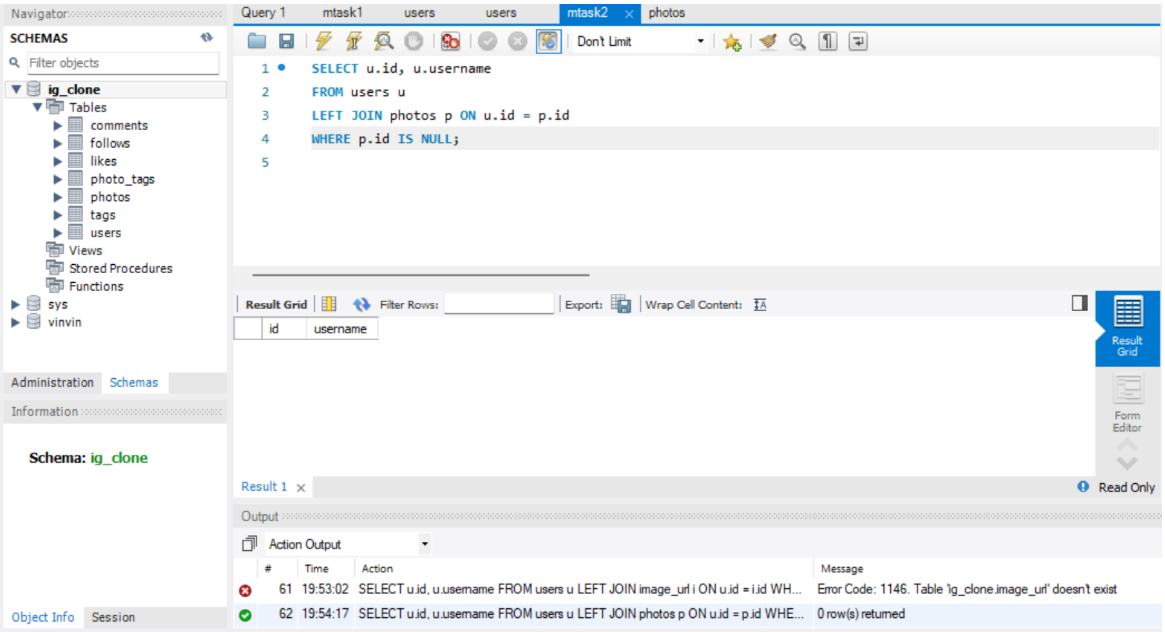
	id	username	created_at
▶	80	Darby_Herzog	2016-05-06 00:14:21
	67	Emilio_Bernier52	2016-05-06 13:04:30
	63	Elenor88	2016-05-08 01:30:41
	95	Nicole71	2016-05-09 17:30:22
	38	Jordyn.Jacobson2	2016-05-14 07:56:26
✱	NULL	NULL	NULL

2. Inactive User Engagement:

   The query discovered a subset of users who had never posted a photo, providing a chance for re-engagement via targeted promotions.

   Code Used:

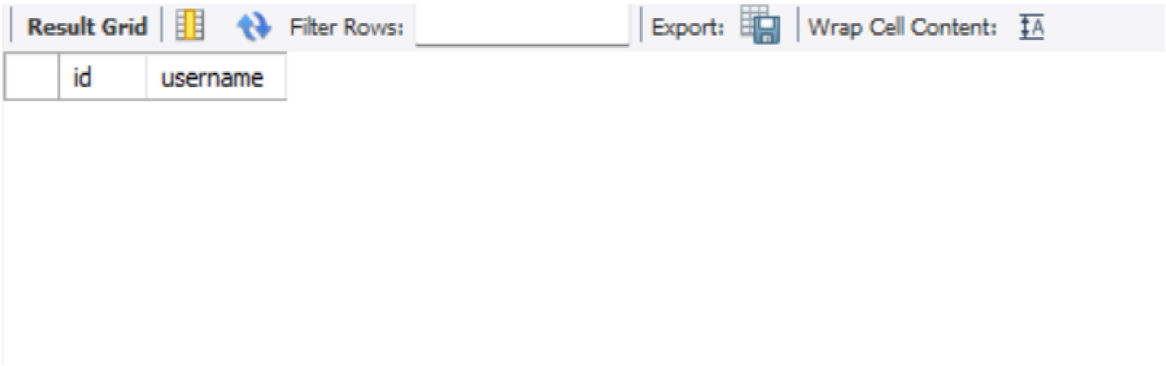
   SELECT u.id, u.username
   FROM users u
   LEFT JOIN photos p ON u.id = p.id
   WHERE p.id IS NULL;



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

```
    #   Time      Action                                                              Message
✖  61  19:53:02  SELECT u.id, u.username FROM users u LEFT JOIN image_url i ON u.id = i.id WH...   Error Code: 1146. Table 'ig_clone.image_url' doesn't exist
✔  62  19:54:17  SELECT u.id, u.username FROM users u LEFT JOIN photos p ON u.id = p.id WHE...     0 row(s) returned
```

Conclusion:
We got the result as:



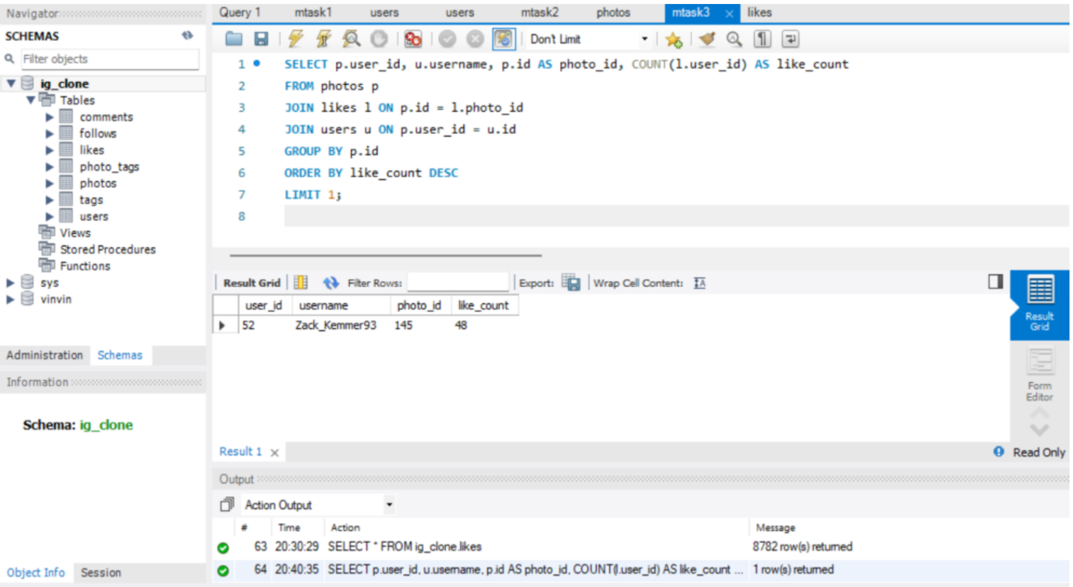
	id	username

3. <u>Contest Winner Declaration</u>:

The query identified the user who received the most likes on a single photo, allowing the contest winner to be declared and providing important input for future engagement activities.

Code Used:

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



Schema: ig_clone

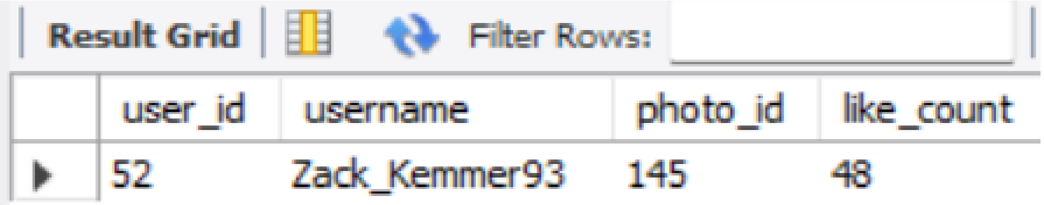
```
SELECT p.user_id, u.username, p.id AS photo_id, COUNT(l.user_id) AS like_count
FROM photos p
JOIN likes l ON p.id = l.photo_id
JOIN users u ON p.user_id = u.id
GROUP BY p.id
ORDER BY like_count DESC
LIMIT 1;
```

	user_id	username	photo_id	like_count
▶	52	Zack_Kemmer93	145	48

```
    #   Time      Action                                                                    Message
✓   63  20:30:29  SELECT * FROM ig_clone.likes                                              8782 row(s) returned
✓   64  20:40:35  SELECT p.user_id, u.username, p.id AS photo_id, COUNT(l.user_id) AS like_count ...  1 row(s) returned
```

Conclusion:

We get the result as:



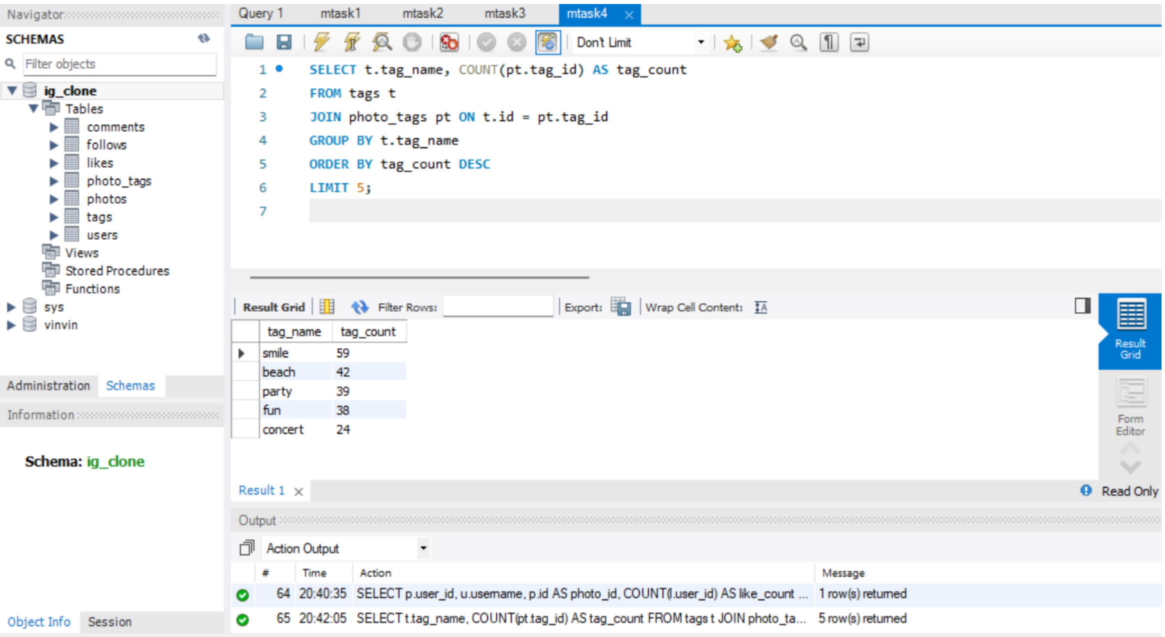
	user_id	username	photo_id	like_count
▶	52	Zack_Kemmer93	145	48

4. <u>Hashtag Research</u>:

The top five most regularly used hashtags had been identified, providing strategic recommendations for the partner brand to maximize their exposure.

Code Used:

```
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;
```



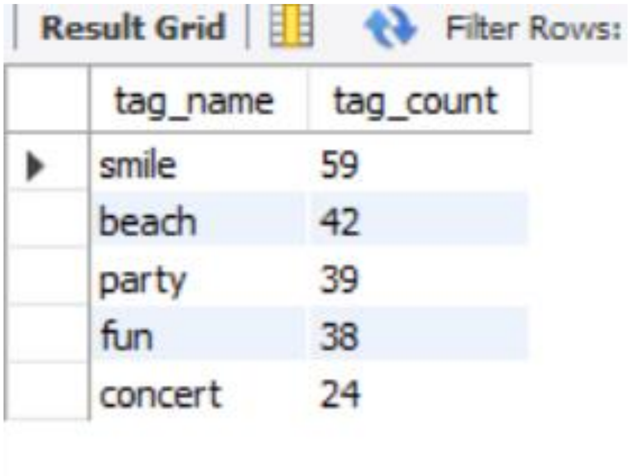
```
SELECT t.tag_name, COUNT(pt.tag_id) AS tag_count
FROM tags t
JOIN photo_tags pt ON t.id = pt.tag_id
GROUP BY t.tag_name
ORDER BY tag_count DESC
LIMIT 5;
```

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

```
   #  Time      Action                                                                  Message
✓ 64  20:40:35  SELECT p.user_id, u.usemame, p.id AS photo_id, COUNT(l.user_id) AS like_count ...  1 row(s) returned
✓ 65  20:42:05  SELECT t.tag_name, COUNT(pt.tag_id) AS tag_count FROM tags t JOIN photo_ta...      5 row(s) returned
```

Conclusion:

We get the result as:


Result Grid
Filter Rows:

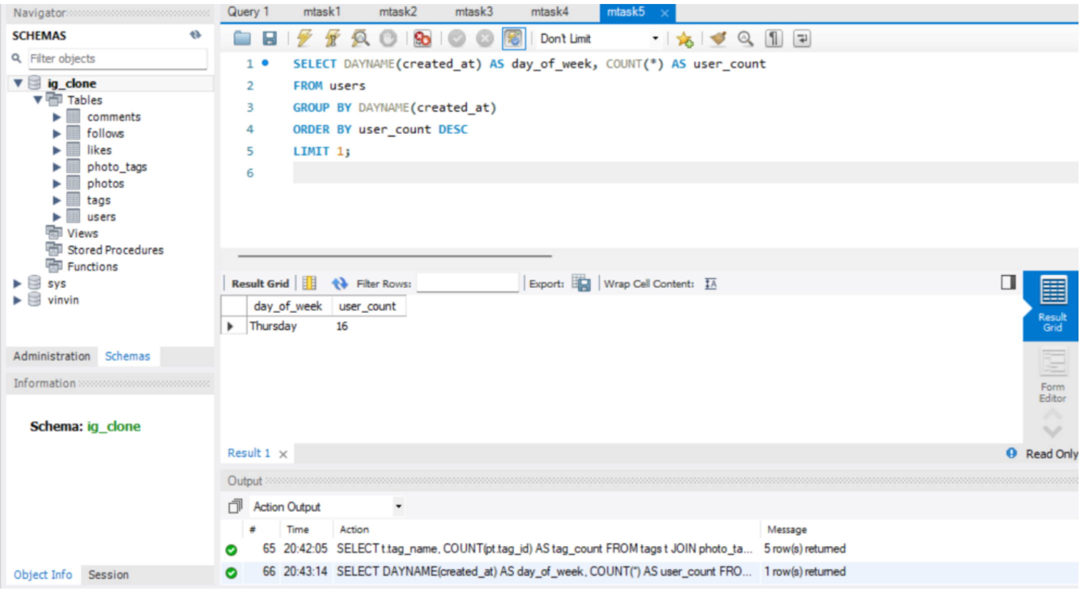
tag_name	tag_count
smile	59
beach	42
party	39
fun	38
concert	24

5. Ad Campaign launch:

By monitoring user registration statistics, we were able to predict the ideal day of the week to debut ads, providing insights for improving campaign performance.

Code Used:

```
SELECT DAYNAME(created_at) AS day_of_week, COUNT(*) AS user_count
FROM users
GROUP BY DAYNAME(created_at)
ORDER BY user_count DESC
LIMIT 1;
```



Schema: ig_clone

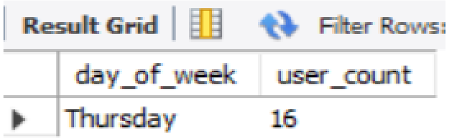
```
SELECT DAYNAME(created_at) AS day_of_week, COUNT(*) AS user_count
FROM users
GROUP BY DAYNAME(created_at)
ORDER BY user_count DESC
LIMIT 1;
```

day_of_week	user_count
Thursday	16

```
#   Time      Action                                                                  Message
65  20:42:05  SELECT t.tag_name, COUNT(pt.tag_id) AS tag_count FROM tags t JOIN photo_ta...  5 row(s) returned
66  20:43:14  SELECT DAYNAME(created_at) AS day_of_week, COUNT(*) AS user_count FRO...       1 row(s) returned
```

Conclusion:

We get the result as:



day_of_week	user_count
Thursday	16

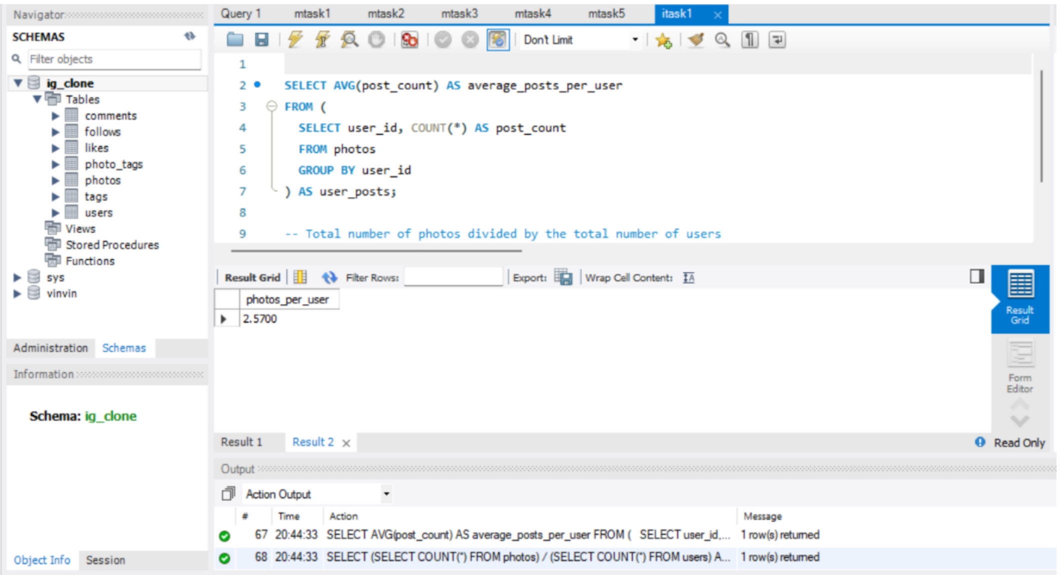
B) *Investor metrics:*

1. User Engagement:

   The query revealed the average amount of postings per user. This information is critical for assessing the general health of the platform and user satisfaction, as well as guiding future product development and marketing tactics.

   Code Used:

```
SELECT AVG(post_count) AS average_posts_per_user
FROM (
  SELECT user_id, COUNT(*) AS post_count
  FROM photos
  GROUP BY user_id
) AS user_posts;

-- Total number of photos divided by the total number of users
SELECT (SELECT COUNT(*) FROM photos) / (SELECT COUNT(*) FROM users)
AS photos_per_user;
```



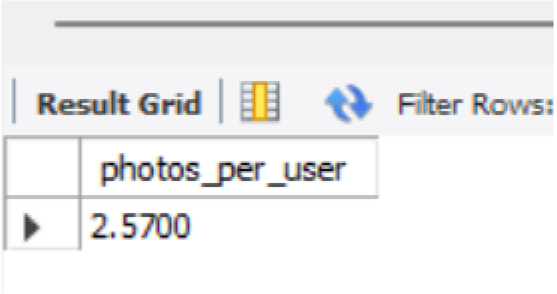
```

SELECT AVG(post_count) AS average_posts_per_user
FROM (
  SELECT user_id, COUNT(*) AS post_count
  FROM photos
  GROUP BY user_id
) AS user_posts;

-- Total number of photos divided by the total number of users
```

photos_per_user
2.5700

#	Time	Action	Message
67	20:44:33	SELECT AVG(post_count) AS average_posts_per_user FROM (SELECT user_id,...	1 row(s) returned
68	20:44:33	SELECT (SELECT COUNT(*) FROM photos) / (SELECT COUNT(*) FROM users) A...	1 row(s) returned

Conclusion:

We get the results as:



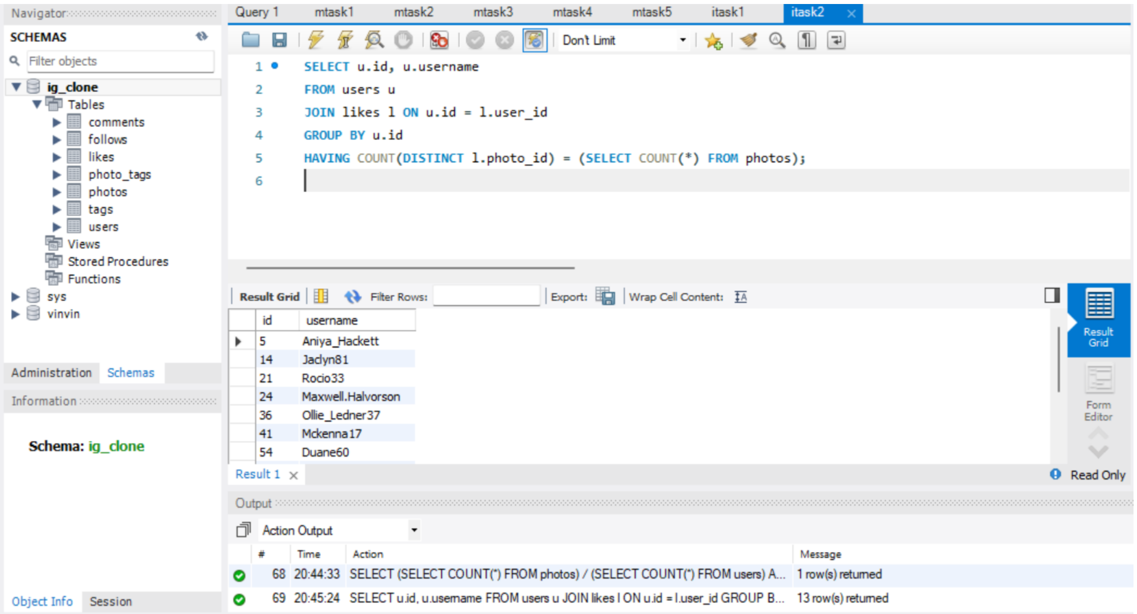
	photos_per_user
▶	2.5700

2. Bots and Fake Accounts:

The query identified suspected bot accounts, allowing the team to protect the platform's integrity through the detection of users with unusual engagement patterns.

Code Used:

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



Schema: ig_clone

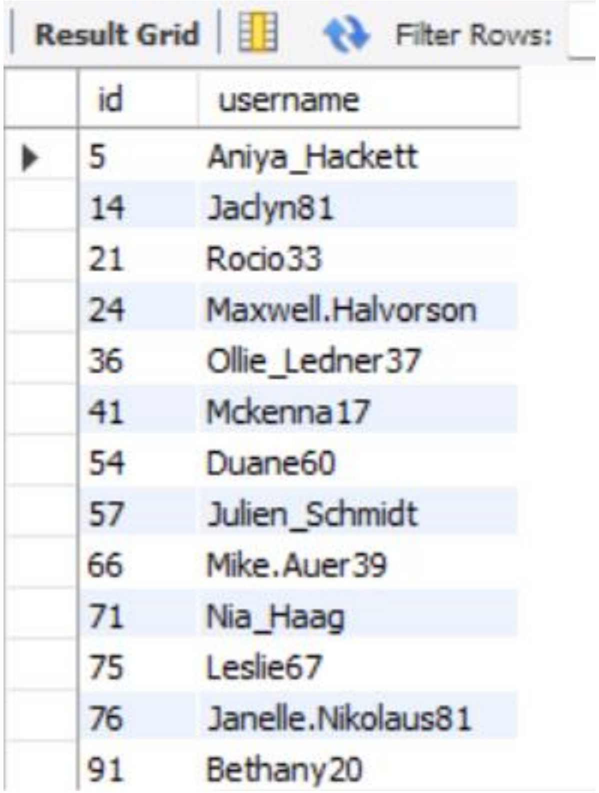
```
SELECT u.id, u.username
FROM users u
JOIN likes l ON u.id = l.user_id
GROUP BY u.id
HAVING COUNT(DISTINCT l.photo_id) = (SELECT COUNT(*) FROM photos);
```

	id	username
▶	5	Aniya_Hackett
	14	Jaclyn81
	21	Rocio33
	24	Maxwell.Halvorson
	36	Ollie_Ledner37
	41	Mckenna17
	54	Duane60

	#	Time	Action	Message
✓	68	20:44:33	SELECT (SELECT COUNT(*) FROM photos) / (SELECT COUNT(*) FROM users) A...	1 row(s) returned
✓	69	20:45:24	SELECT u.id, u.username FROM users u JOIN likes l ON u.id = l.user_id GROUP B...	13 row(s) returned

Conclusion:

We get the result as:



id	username
5	Aniya_Hackett
14	Jadyn81
21	Rocio33
24	Maxwell.Halvorson
36	Ollie_Ledner37
41	Mckenna17
54	Duane60
57	Julien_Schmidt
66	Mike.Auer39
71	Nia_Haag
75	Leslie67
76	Janelle.Nikolaus81
91	Bethany20

Result

Overall, the project was effective in delivering several actionable items that could be immediately situated and implemented by the product and marketing teams at Instagram. The analysis also helps the platform boost repeat usage by developing reward activities for those users who were most consistent. The decision to analyze inactive users would allow for finding ways to re-activate said users and thus, improve activity rates on the site. Information on some of the trending hashtags in the market and best time to post the ad campaign was helpful for strategic marketing approach towards improving the reach and engagement. Furthermore, the identification of abnormal engagement patterns proved useful in identifying bots or fake accounts that compromised the integrity of the platform. In summary, the project provided a valuable service in providing an extensive insight into user behavior relevant to the future enhancement of features, marketing strategies and subsequent user management methods.