

# Instagram User Analytics

**Project Description:** This project aims to extract useful insights from raw data using various database management tools and even visualize them to increase the platform's efficiency.

**Project Approach:** The project was executed using SQL, where queries were utilized to create a database from the provided raw data. Sorting and data extracting queries were then implemented to obtain the required data.

**Tech Stack Used:** The tech stack used included MySQL Workbench.

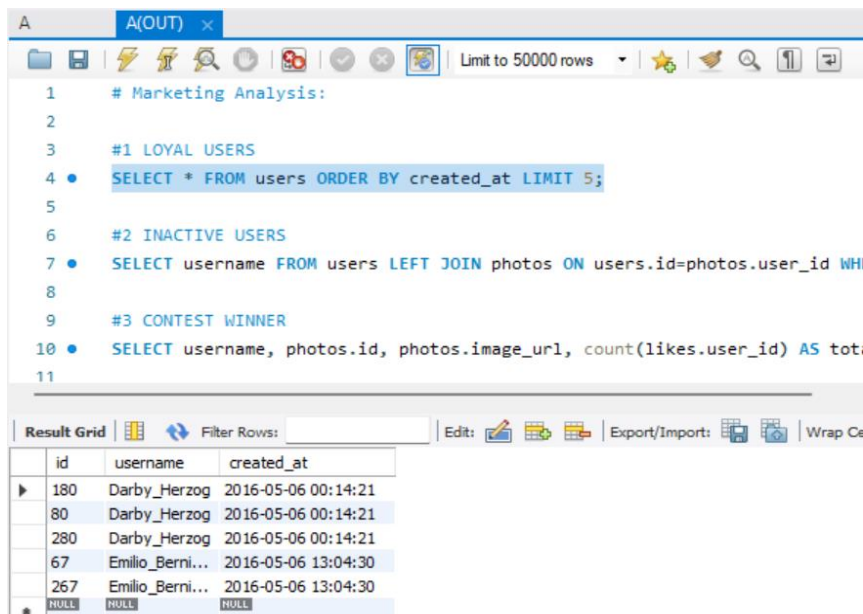
## Project Insight

### (A) Marketing Analysis:

#### **1. Identify the five oldest users on Instagram from the provided database.**

CODE:

**SELECT \* FROM users ORDER BY created\_at LIMIT 5;**



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
1 # Marketing Analysis:
2
3 #1 LOYAL USERS
4 • SELECT * FROM users ORDER BY created_at LIMIT 5;
5
6 #2 INACTIVE USERS
7 • SELECT username FROM users LEFT JOIN photos ON users.id=photos.user_id WHERE
8
9 #3 CONTEST WINNER
10 • SELECT username, photos.id, photos.image_url, count(likes.user_id) AS tot
11
```

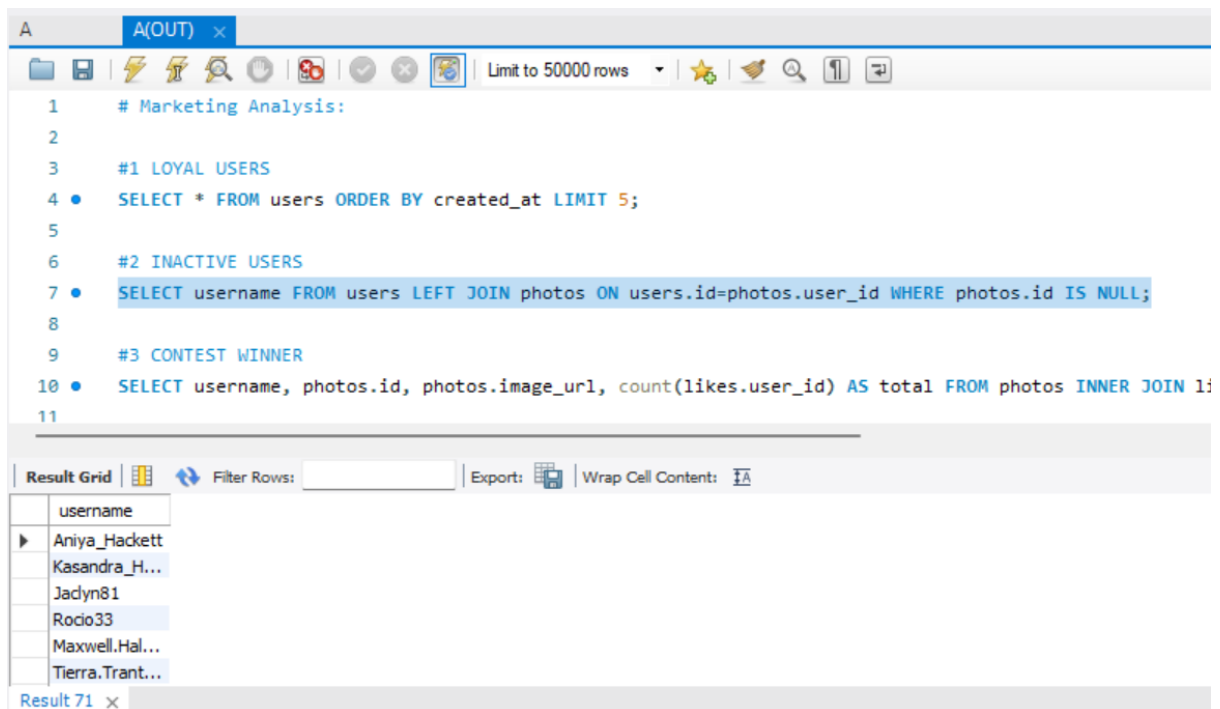
The Results tab shows the output of the query:

	id	username	created_at
▶	180	Darby_Herzog	2016-05-06 00:14:21
	80	Darby_Herzog	2016-05-06 00:14:21
	280	Darby_Herzog	2016-05-06 00:14:21
	67	Emilio_Berni...	2016-05-06 13:04:30
	267	Emilio_Berni...	2016-05-06 13:04:30
*	NULL	NULL	NULL

## 2. Identify users who have never posted a single photo on Instagram.

CODE:

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



## 3. Determine the winner of the contest and provide their details to the team.

CODE:

```
SELECT username, photos.id, photos.image_url, count(likes.user_id) AS total  
FROM photos INNER JOIN likes ON likes.photo_id=photos.id INNER JOIN  
users ON photos.user_id = users.id  
GROUP BY photos.id ORDER BY total DESC LIMIT 1;
```

A A(OUT) x

Limit to 50000 rows

```

9 #3 CONTEST WINNER
10 • SELECT username, photos.id, photos.image_url, count(likes.user_id) AS total
11 FROM photos INNER JOIN likes ON likes.photo_id=photos.id INNER JOIN users ON photos.user_id = users.id
12 GROUP BY photos.id ORDER BY total DESC LIMIT 1;
13
14 #4 MOST HASHTAG
15 • SELECT tags.tag_name, COUNT(*) AS total FROM photo_tags JOIN tags ON photo_tags.tag_id = tags.id GROUP BY t;
16
17 #5 MOST INSTAGRAM USER REGISTER
18 • SELECT DAYNAME(created_at) AS day, count(*) AS total FROM users GROUP BY day ORDER BY total DESC LIMIT 1;
19

```

Result Grid Filter Rows: Exports: Wrap Cell Content: Fetch rows:

	username	id	image_url	total
▶	Zack_Kemm...	145	https://jarret.name	48

Result 72 x

#### 4. Identify and suggest the top five most commonly used hashtags on the platform.

##### CODE:

```

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

```

|

|

|

|

V

A A(OUT)\* x

Limit to 50000 rows

```

13
14 #4 MOST HASHTAG
15 • SELECT tags.tag_name, COUNT(*) AS total
16 FROM photo_tags JOIN tags ON photo_tags.tag_id = tags.id
17 GROUP BY tags.id ORDER BY total DESC LIMIT 5;
18
19 #5 MOST INSTAGRAM USER REGISTER
20 • SELECT DAYNAME(created_at) AS day, count(*) AS total FROM users GF
21
22 # Investor Metrics:
23

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch r

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

Result 73 x

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

**CODE:**

```

SELECT DAYNAME(created_at) AS day, count(*) AS total
FROM users GROUP BY day ORDER BY total DESC LIMIT 1;

```

|

|

|

V

Limit to 50000 rows

```
FROM photo_tags JOIN tags ON photo_tags.tag_id = tags.id
GROUP BY tags.id ORDER BY total DESC LIMIT 5;
```

18

## #5 MOST INSTAGRAM USER REGISTER

```
SELECT DAYNAME(created_at) AS day, count(*) AS total
FROM users GROUP BY day ORDER BY total DESC LIMIT 1;
```

22

### # Investor Metrics:

24

## #1 USERARRANGEMENT

```
select *from photos.users:
```



Filter Rows:

Export:



Wrap Cell Content:

A	Fetch rows:
---	-------------



	day	total
►	Thursday	48

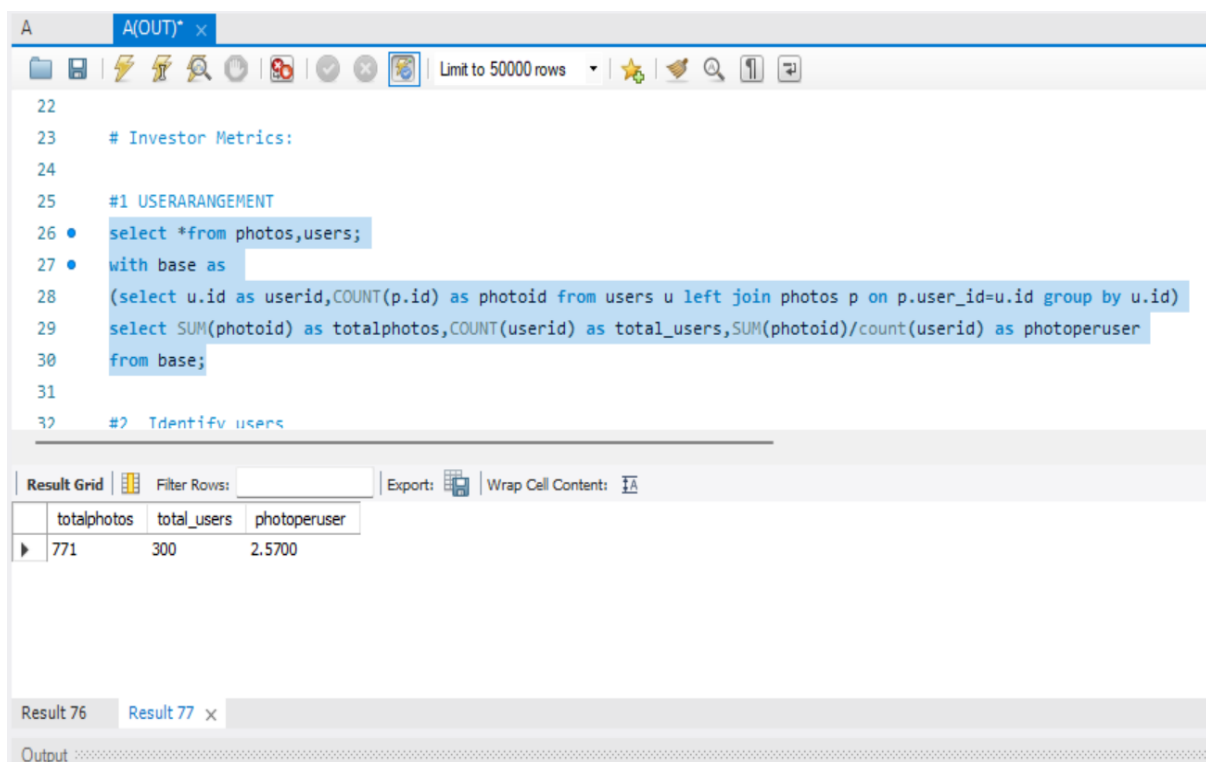
### Output

## B) Investor Metrics:

1. 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.

### CODE:

```
select *from photos,users;
with base as
(select u.id as userid,COUNT(p.id) as photoid from users u left
join photos p on p.user_id=u.id group by u.id)
select SUM(photoid) as totalphotos,COUNT(userid) as
total_users,SUM(photoid)/count(userid) as photoperuser
from base;
```



The screenshot displays a SQL query editor with the following code:

```
22
23 # Investor Metrics:
24
25 #1 USERARRANGEMENT
26 • select *from photos,users;
27 • with base as
28   (select u.id as userid,COUNT(p.id) as photoid from users u left join photos p on p.user_id=u.id group by u.id)
29   select SUM(photoid) as totalphotos,COUNT(userid) as total_users,SUM(photoid)/count(userid) as photoperuser
30   from base;
31
32 #2 Identifv users
```

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

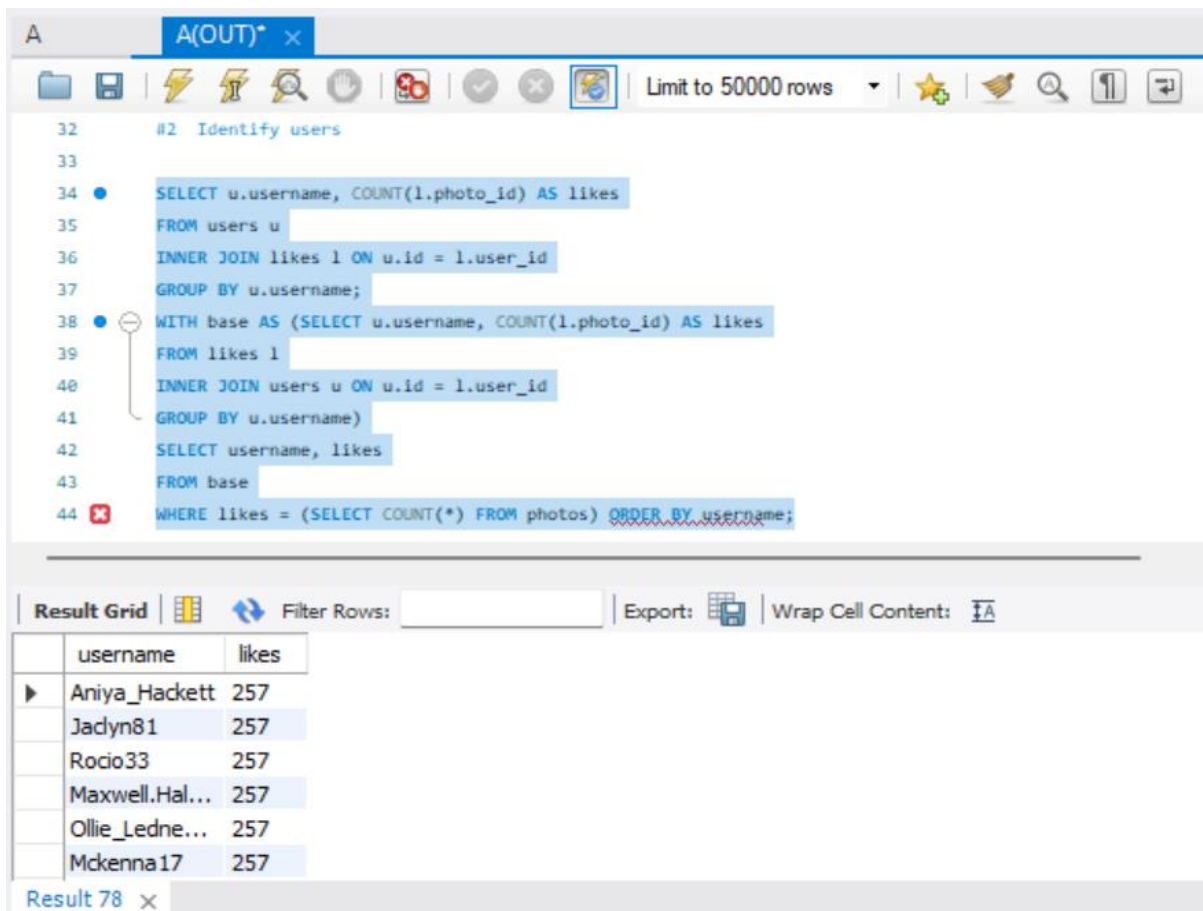
	totalphotos	total_users	photoperuser
▶	771	300	2.5700

The interface includes a toolbar with icons for file operations, a 'Limit to 50000 rows' dropdown, and an 'Export' button. The bottom of the window shows 'Result 76' and 'Result 77' tabs, and an 'Output' section.

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

**CODE:**

```
SELECT u.username, COUNT(l.photo_id) AS likes FROM users u  
INNER JOIN likes l ON u.id = l.user_id GROUP BY u.username;  
WITH base AS (SELECT u.username, COUNT(l.photo_id) AS likes  
FROM likes l INNER JOIN users u ON u.id = l.user_id  
GROUP BY u.username) SELECT username, likes  
FROM base WHERE likes = (SELECT COUNT(*) FROM photos)  
ORDER BY username;
```



The screenshot shows a SQL query editor with a query window titled "A(OUT)\*" and a "Result Grid" window. The query is as follows:

```
#2 Identify users  
SELECT u.username, COUNT(l.photo_id) AS likes  
FROM users u  
INNER JOIN likes l ON u.id = l.user_id  
GROUP BY u.username;  
WITH base AS (SELECT u.username, COUNT(l.photo_id) AS likes  
FROM likes l  
INNER JOIN users u ON u.id = l.user_id  
GROUP BY u.username)  
SELECT username, likes  
FROM base  
WHERE likes = (SELECT COUNT(*) FROM photos) ORDER BY username;
```

The "Result Grid" window displays the following data:

username	likes
Aniya_Hackett	257
Jadyn81	257
Rocio33	257
Maxwell.Hal...	257
Ollie_Ledne...	257
Mckenna17	257

**THANK FOR GIVING ME THE OPPORTUNITY TO DO THE PROJECT .**