



Instagram Clone SQL Project



By - Ujjwal Mishra



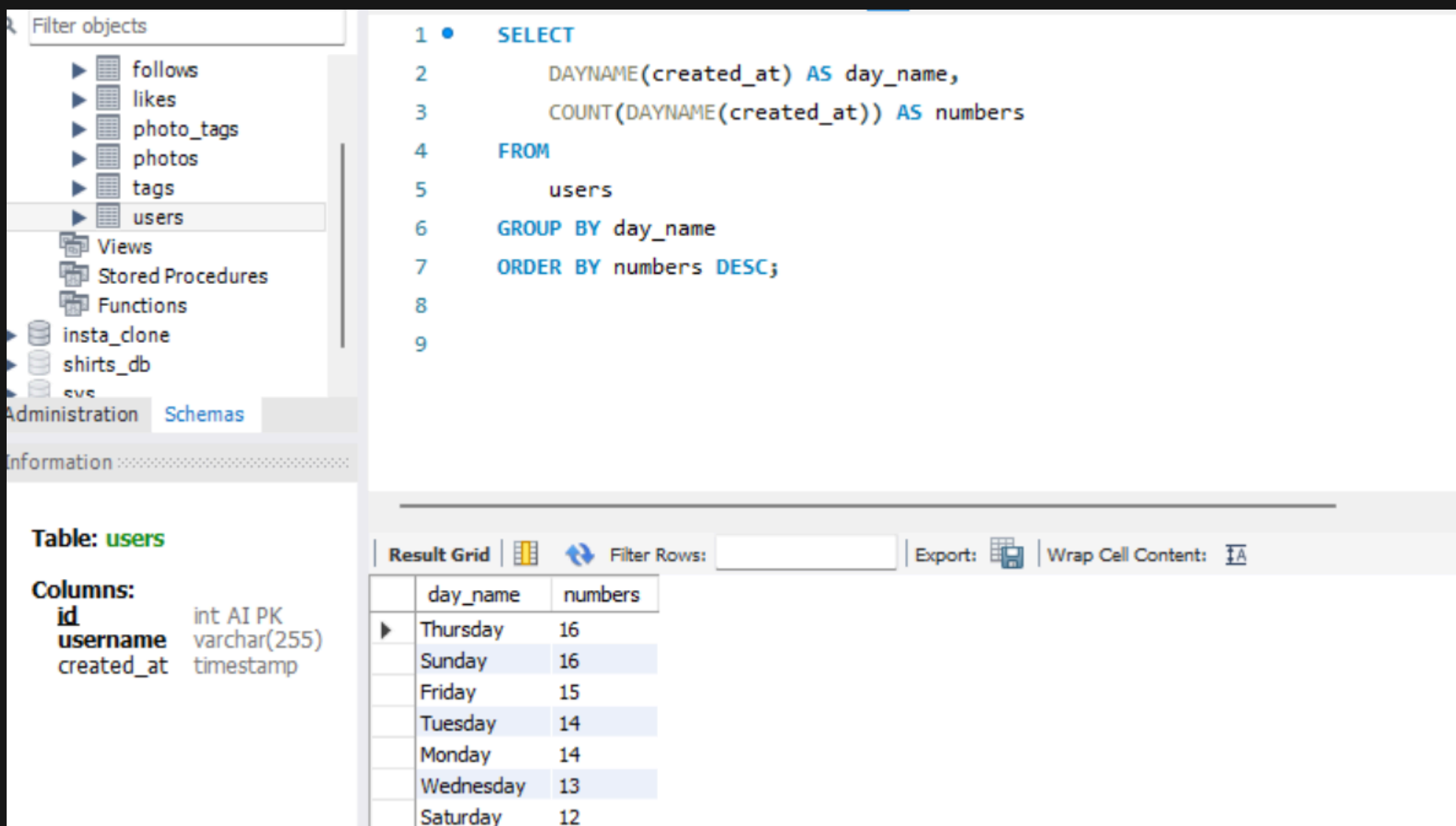
Objective :

The objective of this project is to leverage a cloned Instagram database to conduct a comprehensive SQL-based analysis, extracting actionable insights regarding user behaviour, content trends, and platform engagement.

This analysis will aim to identify key patterns and relationships within the data, providing a deeper understanding of users' behaviour on the Instagram platform.

1. We need to figure out when to schedule an ad campaign.

What day of the week do most users register on ?



The screenshot shows a database management interface. On the left, a 'Filter objects' pane lists database objects: follows, likes, photo_tags, photos, tags, users, Views, Stored Procedures, Functions, insta_clone, shirts_db, and svs. The 'users' table is selected. Below this, the 'Table: users' structure is shown with columns: id (int AI PK), username (varchar(255)), and created_at (timestamp). The main area displays a SQL query:

```
1 • SELECT
2     DAYNAME(created_at) AS day_name,
3     COUNT(DAYNAME(created_at)) AS numbers
4 FROM
5     users
6 GROUP BY day_name
7 ORDER BY numbers DESC;
8
9
```

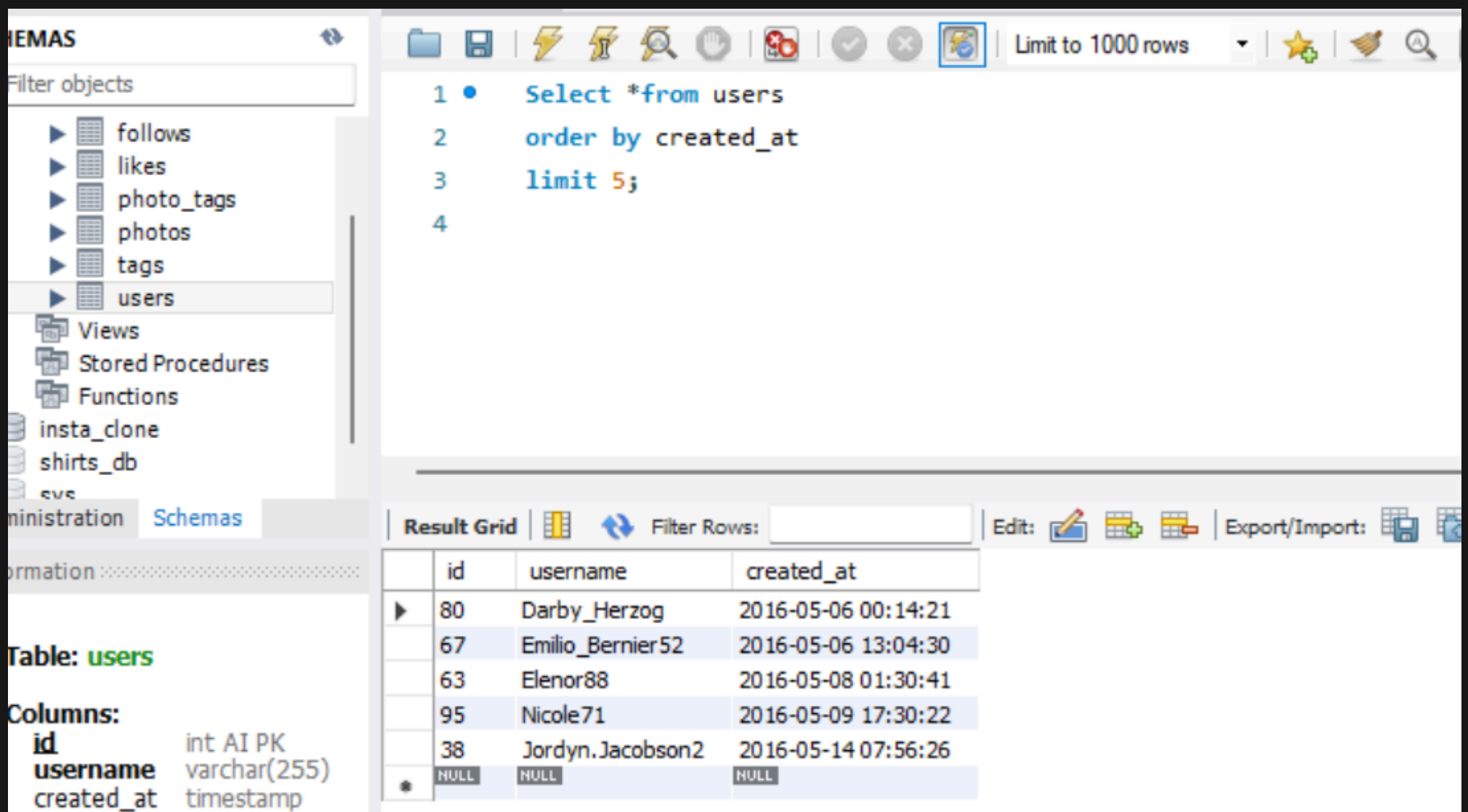
Below the query, the 'Result Grid' shows the following data:

day_name	numbers
Thursday	16
Sunday	16
Friday	15
Tuesday	14
Monday	14
Wednesday	13
Saturday	12

Select DAYNAME (created_at) AS day_name,
COUNT (DAYNAME (created_at)) AS numbers
From users
GROUP BY day_name
ORDER BY numbers DESC;

2. We want to reward our users who have been around the longest.

Find the 5 oldest users.



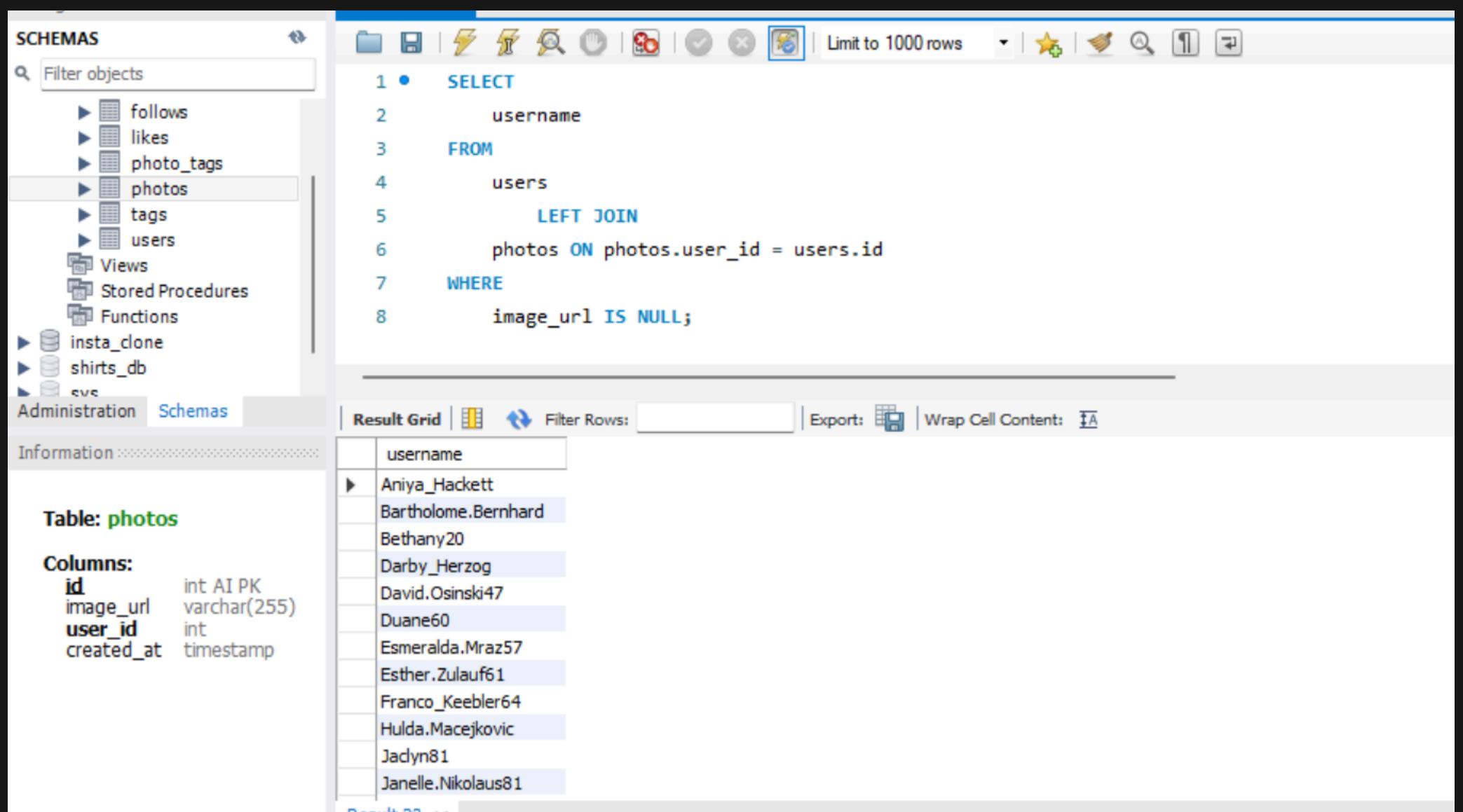
The screenshot shows a database management interface. On the left, a sidebar lists database objects including 'users'. The main area displays a SQL query: `Select *from users order by created_at limit 5;`. Below the query, a 'Result Grid' shows the top 5 oldest users based on their 'created_at' timestamp.

	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

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

3. We want to our target inactive users with an email campaign.

Find the users who have never posted a photo.



The screenshot shows a database management interface. On the left, the 'SCHEMAS' panel lists various tables including 'follows', 'likes', 'photo_tags', 'photos', 'tags', 'users', 'Views', 'Stored Procedures', and 'Functions'. The 'photos' table is selected. Below the schemas, the 'Table: photos' structure is shown with columns: 'id' (int AI PK), 'image_url' (varchar(255)), 'user_id' (int), and 'created_at' (timestamp).

The main query editor displays the following SQL query:

```
1 SELECT
2     username
3 FROM
4     users
5     LEFT JOIN
6     photos ON photos.user_id = users.id
7 WHERE
8     image_url IS NULL;
```

The 'Result Grid' shows the results of the query, listing usernames of users who have never posted a photo. The results are as follows:

username
Aniya_Hackett
Bartholome.Bernhard
Bethany20
Darby_Herzog
David.Osinski47
Duane60
Esmeralda.Mraz57
Esther.Zulauf61
Franco_Keebler64
Hulda.Macejkovic
Jadyn81
Janelle.Nikolaus81

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

4. We are running a new contest to see who can get the most the most like on a single photo.

Who won ?

The screenshot shows a database management interface. On the left, a 'Filter objects' sidebar lists various database objects, with 'photos' selected. Below this, the 'Table: photos' structure is shown with columns: id (int AI PK), image_url (varchar(255)), user_id (int), and created_at (timestamp). The main area displays a SQL query:

```
1 • SELECT
2     username, photo_id, COUNT(photo_id) AS numbers
3 FROM
4     photos
5     INNER JOIN
6     likes ON likes.photo_id = photos.id
7     INNER JOIN
8     users ON photos.user_id = users.id
9 GROUP BY photo_id
10 ORDER BY numbers DESC
11 LIMIT 2;
```

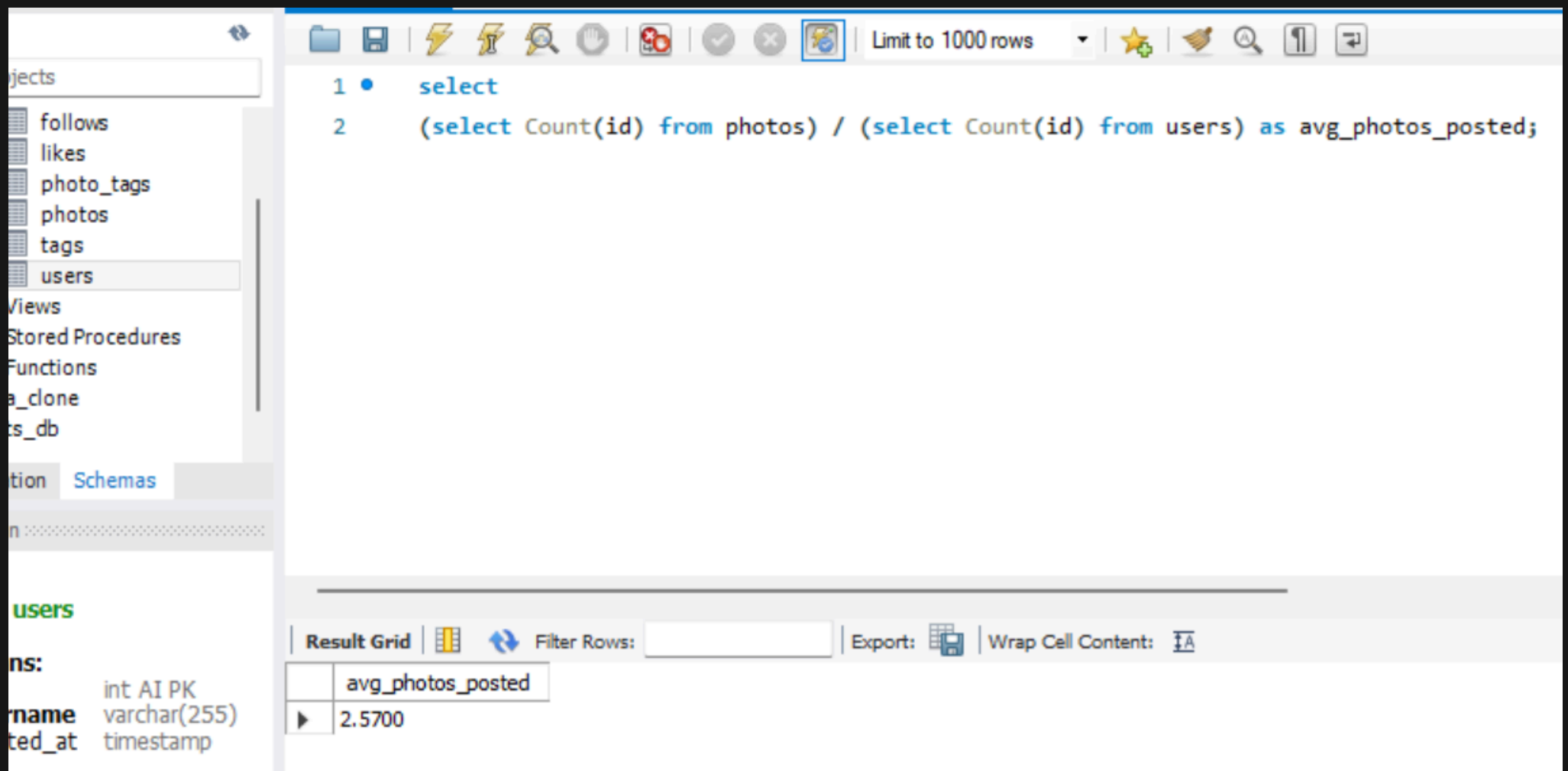
At the bottom, the 'Result Grid' shows the following data:

username	photo_id	numbers
Zack_Kemmer93	145	48
Malinda_Streich	127	43

```
SELECT username, photo_id,
COUNT(photo_id) AS numbers FROM
photos
INNER JOIN likes ON likes.photo_id =
photos.id
INNER JOIN users ON photos,user_id =
photos.id
GROUP BY photo_id
ORDER BY numbers DESC
LIMIT 2 ;
```

5. Investors want to know.....

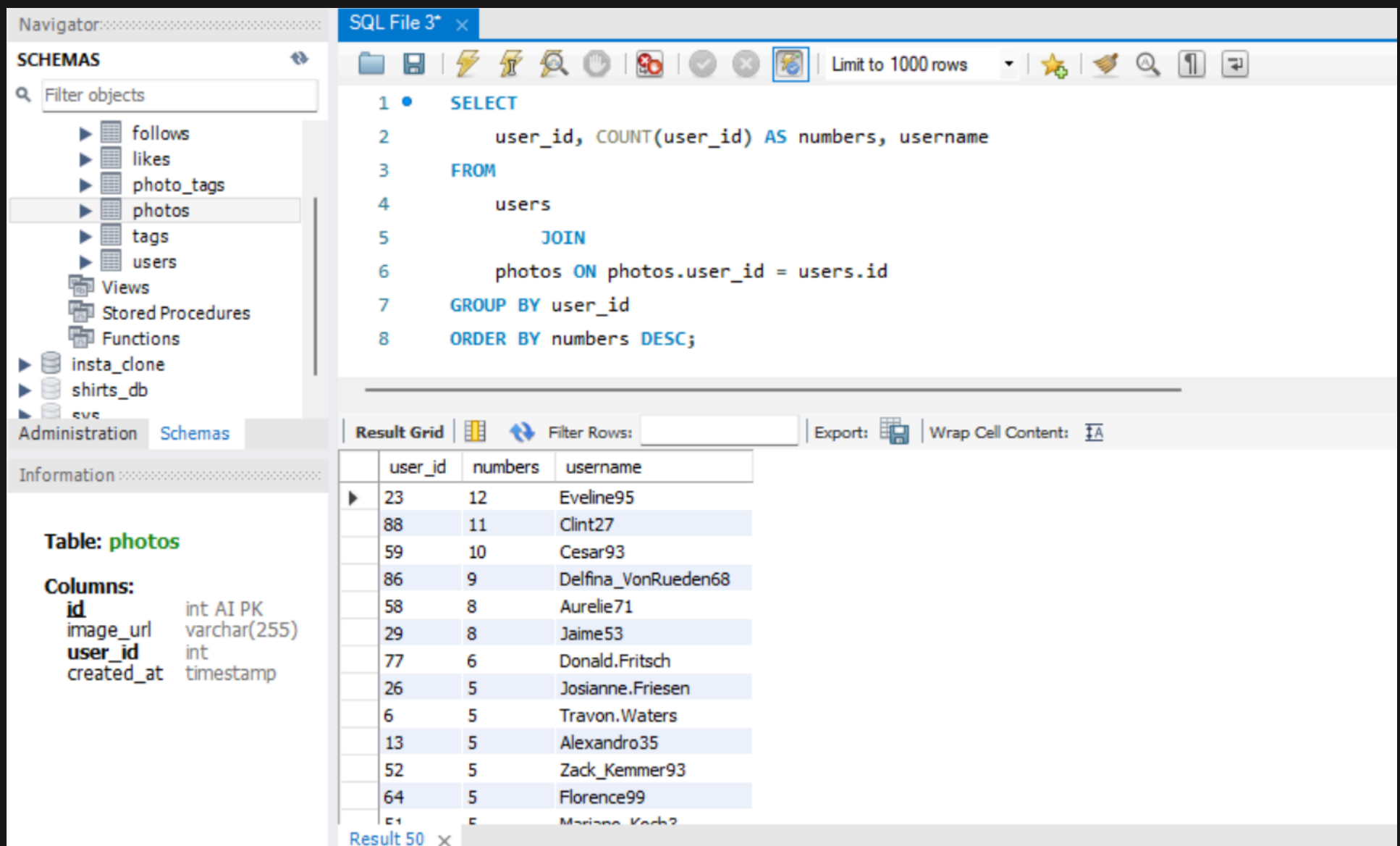
How many times does the average user posts ?



```
SELECT
(SELECT COUNT(id) FROM
photos / (SELECT COUNT (id)
FROM users)
AS avg_photos_posted ;
```

6. Investors want to know.....

**Numbers of photos
posted by each user.**



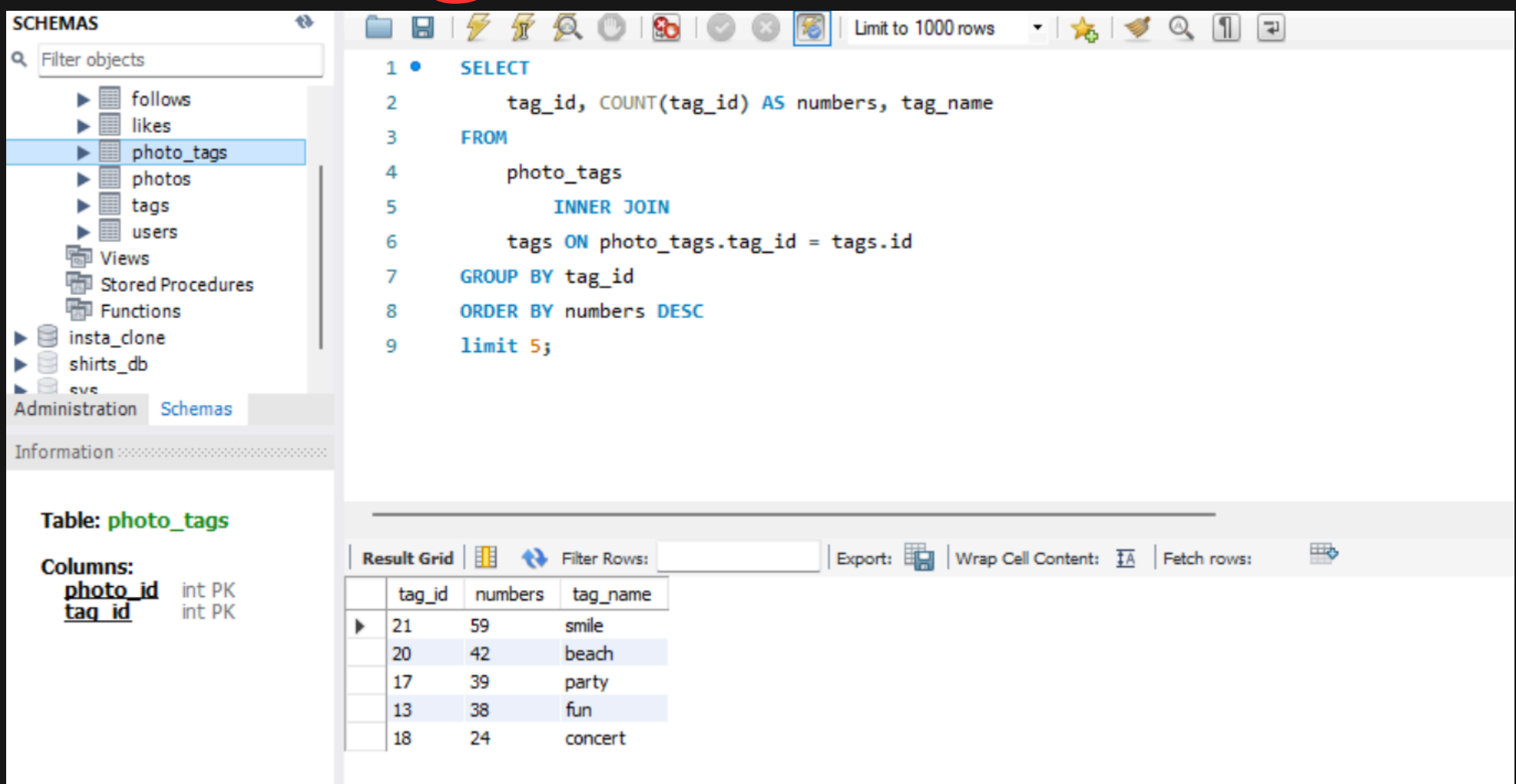
The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane lists database objects, with 'photos' selected. Below it, the 'Table: photos' structure is shown with columns: id (int AI PK), image_url (varchar(255)), user_id (int), and created_at (timestamp). The main editor displays a SQL query: `SELECT user_id, COUNT(user_id) AS numbers, username FROM users JOIN photos ON photos.user_id = users.id GROUP BY user_id ORDER BY numbers DESC;`. The 'Result Grid' at the bottom shows the query results, sorted by the number of photos in descending order.

user_id	numbers	username
23	12	Eveline95
88	11	Clint27
59	10	Cesar93
86	9	Delfina_VonRueden68
58	8	Aurelie71
29	8	Jaime53
77	6	Donald.Fritsch
26	5	Josianne.Friesen
6	5	Travon.Waters
13	5	Alexandro35
52	5	Zack_Kemmer93
64	5	Florence99
51	5	Marlene_Koch2

```
SELECT user_id, photo_id,  
COUNT (user_id)  
AS numbers, username  
FROM  
users  
JOIN photos ON photos.user_id = users.id  
GROUP BY user_id  
ORDER BY numbers DESC ;
```


7. Brands want to know most popular hashtags.

What are the top 5 most commonly used hashtags?



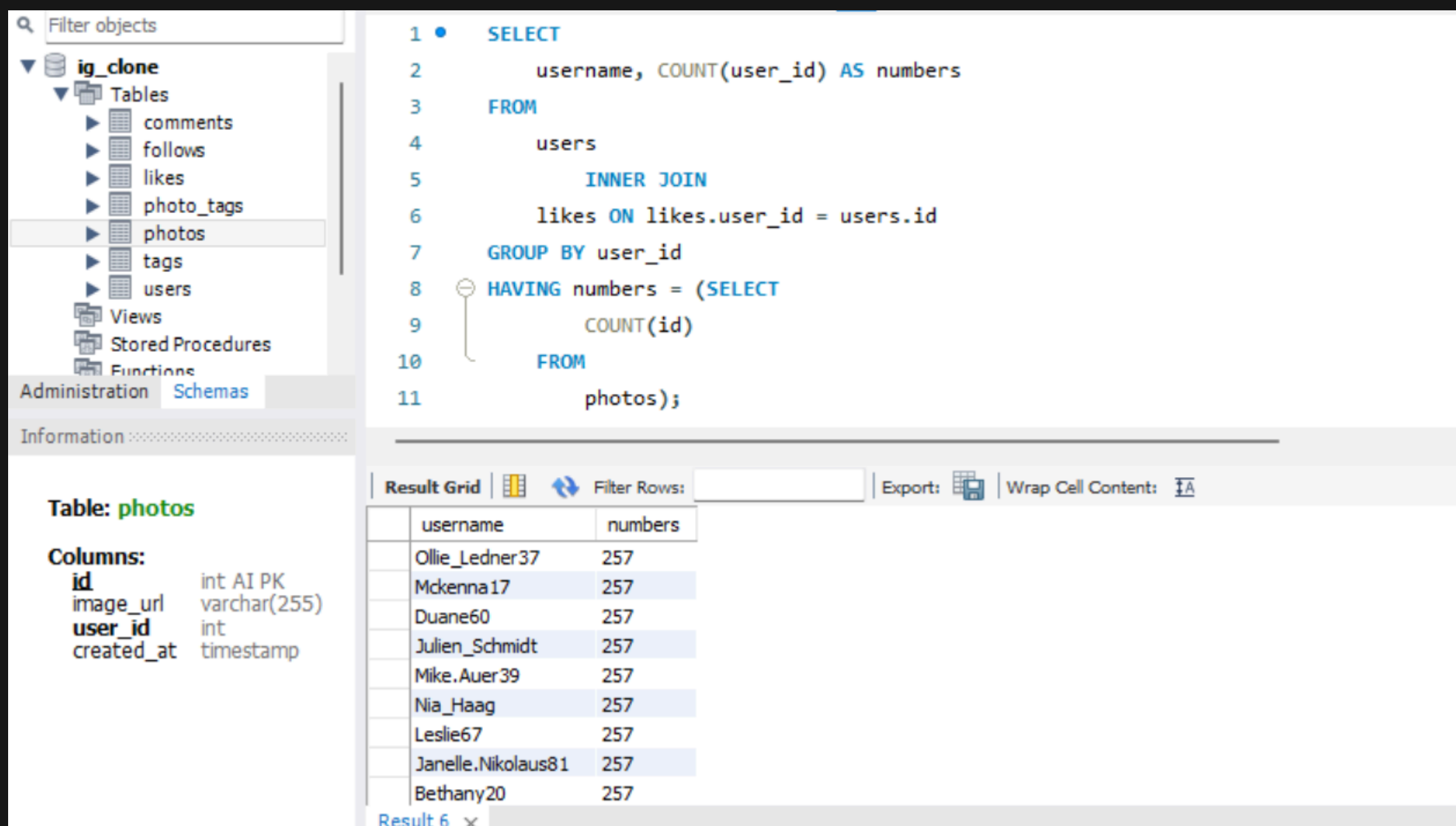
The screenshot shows a database management interface. On the left, a 'SCHEMAS' panel lists various database objects, with 'photo_tags' selected. The main area displays a SQL query: `SELECT tag_id, COUNT(tag_id) AS numbers, tag_name FROM photo_tags INNER JOIN tags ON photo_tags.tag_id = tags.id GROUP BY tag_id ORDER BY numbers DESC limit 5;`. Below the query, a 'Result Grid' shows the top 5 most common hashtags:

tag_id	numbers	tag_name
21	59	smile
20	42	beach
17	39	party
13	38	fun
18	24	concert

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

8. We have a small problem with bots on our site.

Find users who have likes every single photo on the site?



The screenshot shows a database management interface with a schema tree on the left, a SQL editor in the center, and a result grid at the bottom right. The schema tree shows a database named 'ig_clone' with tables: comments, follows, likes, photo_tags, photos, tags, and users. The SQL editor contains the following query:

```
1 • SELECT
2     username, COUNT(user_id) AS numbers
3 FROM
4     users
5     INNER JOIN
6     likes ON likes.user_id = users.id
7 GROUP BY user_id
8 HAVING numbers = (SELECT
9     COUNT(id)
10    FROM
11    photos);
```

The result grid shows the following data:

username	numbers
Ollie_Ledner37	257
Mckenna17	257
Duane60	257
Julien_Schmidt	257
Mike.Auer39	257
Nia_Haag	257
Leslie67	257
Janelle.Nikolaus81	257
Bethany20	257

```
SELECT username,
Count(user_id) AS numbers
FROM users
INNER JOIN likes
ON users.id = likes.user_id
GROUP BY user_id
HAVING numbers = (SELECT Count(id)
FROM photos);
```

Thank You For Your Attention

