

Instagram User Analytics

Description: In this project we track user engagement and interaction with Digital Product in an attempt to derive business insight for marketing and product development teams.

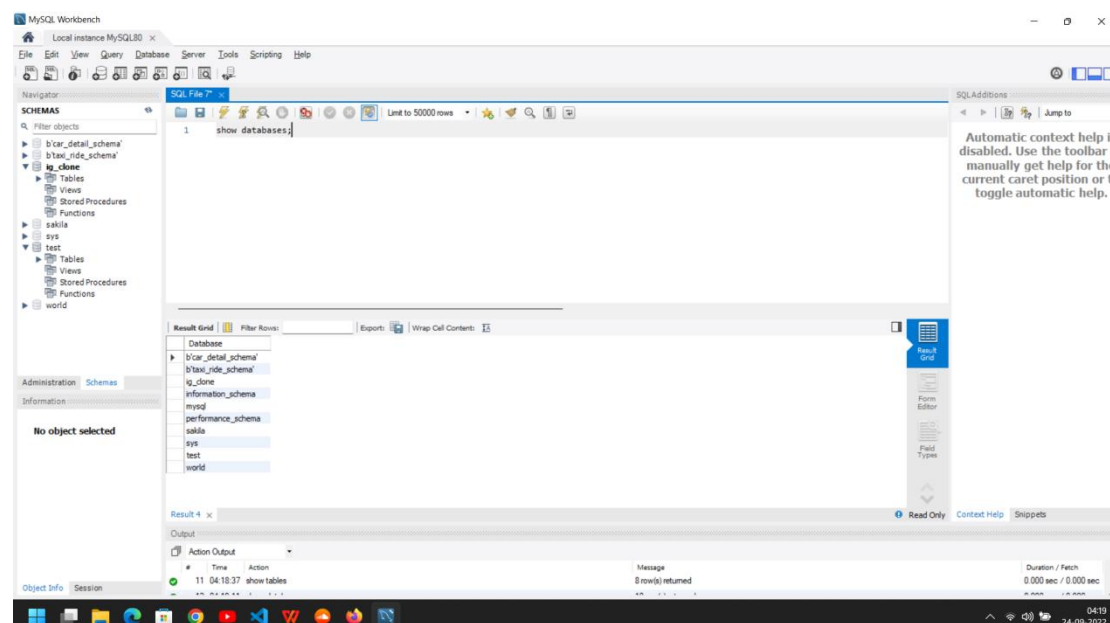
A) Marketing: The marketing team wants to launch some campaigns, these are questions they want to answer:

1. **Rewarding Most Loyal Users:** People who have been using the platform for the longest time.

Tech stack used: We have our database setup in MySQL DBMS system and we will use MySQL Workbench 8.0 CE for this project.

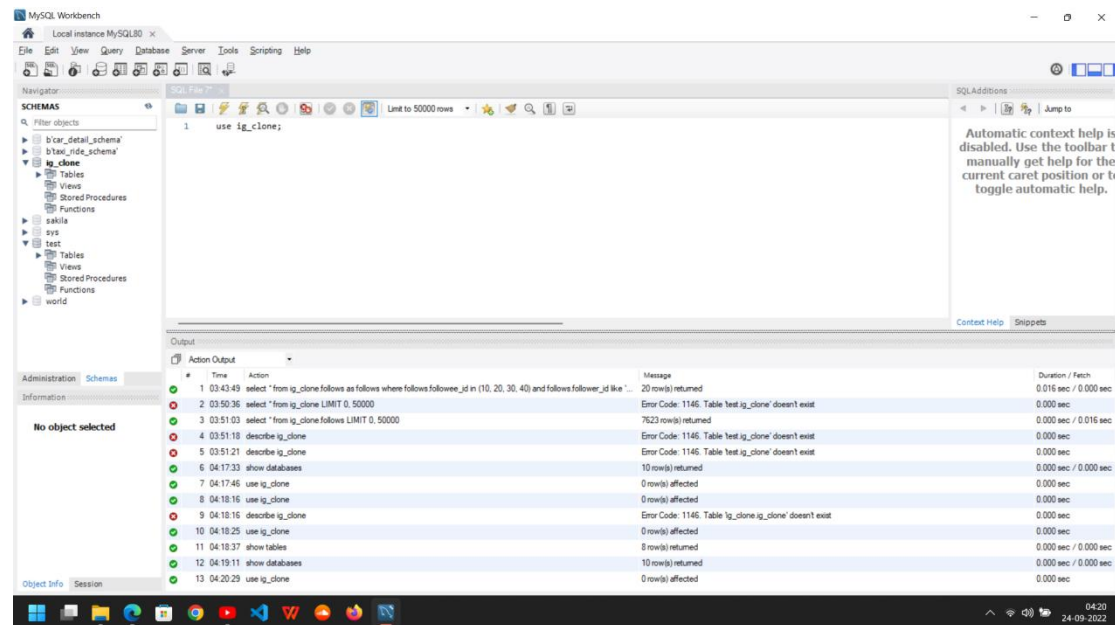
Step 1:

The first step when querying any database is to list out all the database we want to query, this step is crucial when using DBMS in a terminal, however using a console we have a view of all the databases on the left side SCHEMA pane:



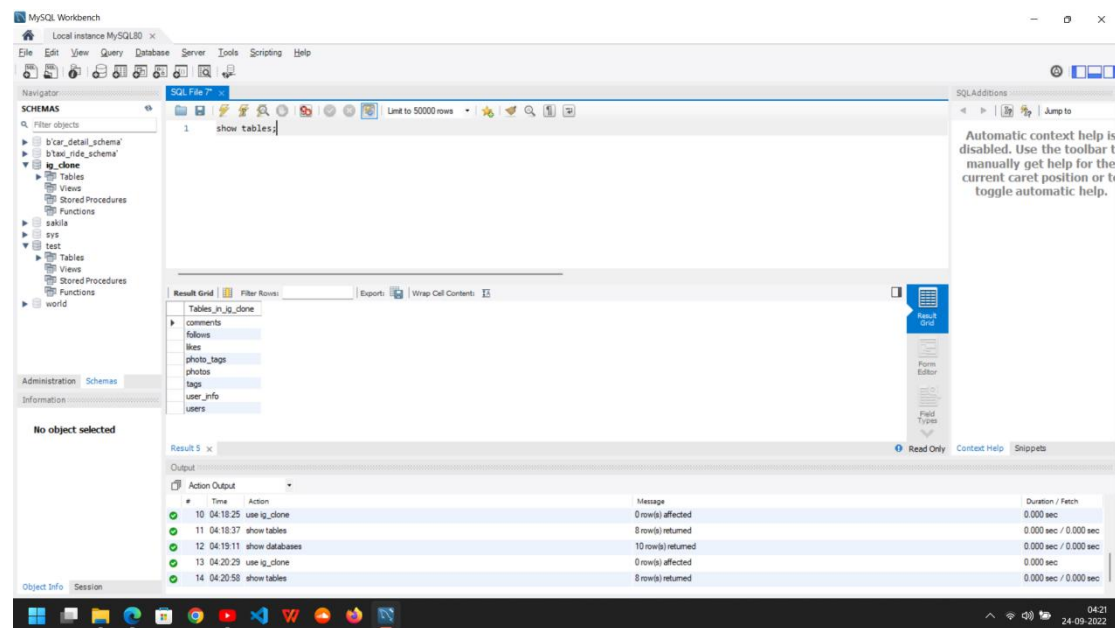
Step 2:

Use the database we want to query:



Step 3:

Now we will list out all the tables in the database:



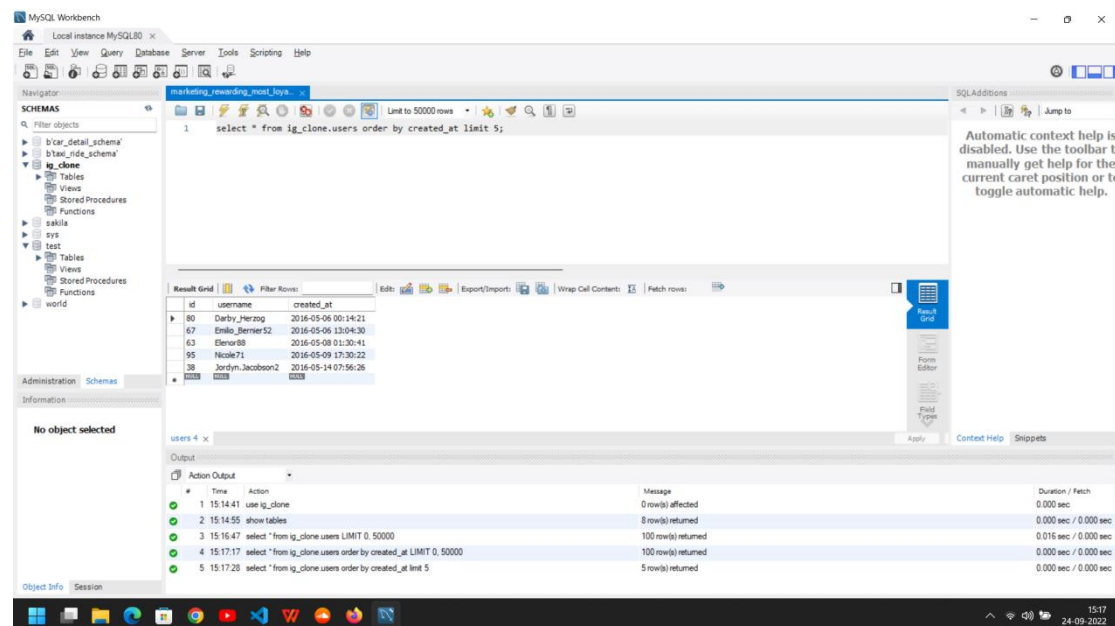
We see following tables present in the database:

- comments
- follows
- likes
- photo_tags
- photos
- tags
- users

Now we want to get the list of users who have been using the platform for longest time.

Step 4:

Now we will query the database to get the answer, we query the table users as the name is self descriptive and it contains the user information:



We use `"select * from ig_clone.users order by created_at limit 5;"` this returns the list 5 oldest users we can forward this list to marketing team so they can customize the reward for each users.

Note: ig_clone is the name of the database we are using in this project.

Every time we query a table the first is to get a overview of the whole table this can be achieved using `"describe table_name;"` or simply querying with `"*"` all columns and limiting value to 5 to get a sense of what the data looks like.

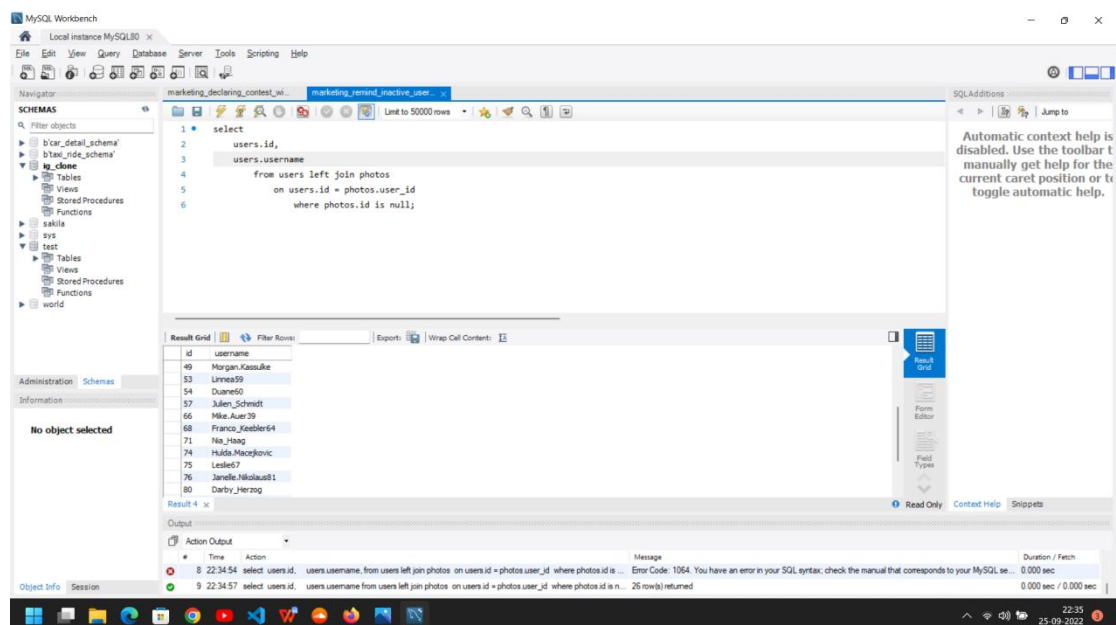
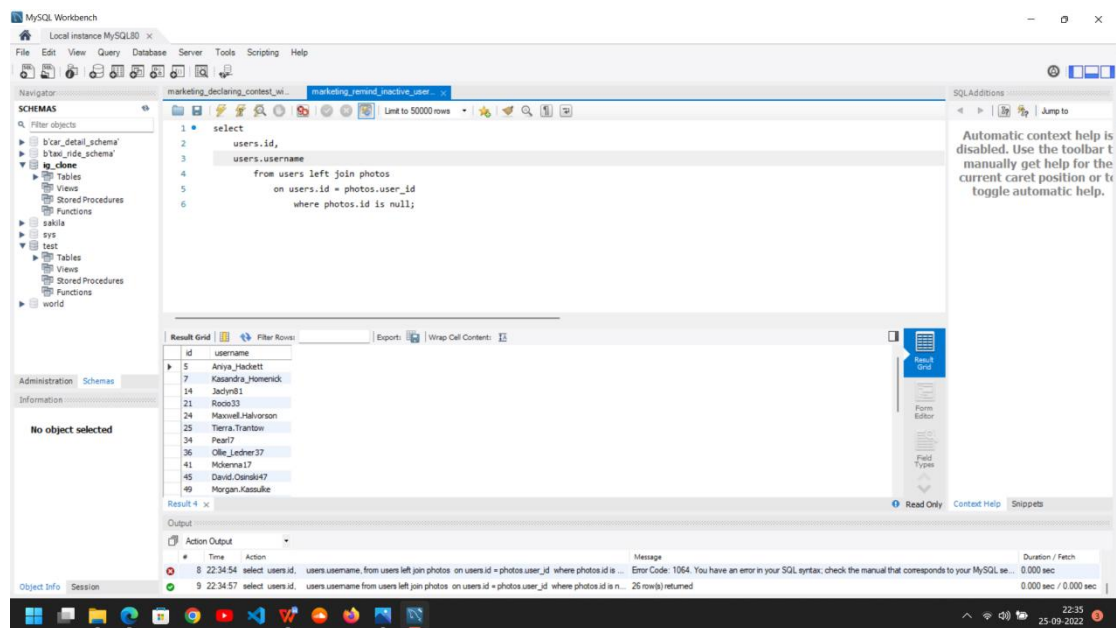
2. Remind Inactive Users to Start Posting: By sending them promotional emails to post their 1st photo.

Step 1:

To answer this question we query **users** and **photos** tables, as to answer this question we want to know who has not posted anything yet on the platform.

We use left join to join two tables **users** and **photos** as we want to match all the users in users table and we check for photo_id in **photos** table to be null.

Here is the full query “select users.id from users left join photos on users.id = photos.user_id where photos.id is null”

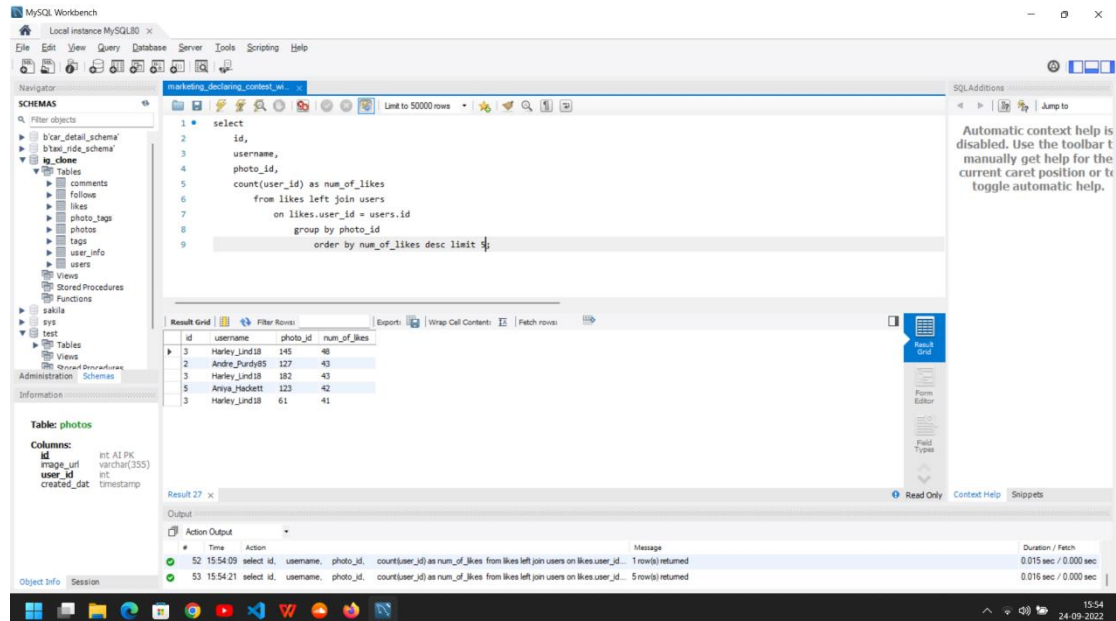


We forward our finding so that marketing team can send these people promotional emails.

3. **Declaring Contest Winner:** The team started a contest and the user who gets the most likes on a single photo will win the contest now they wish to declare the winner.

Step 1:

To answer this question we need to find out the user who got the highest likes on a photo:



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

```
1 select
2   id,
3   username,
4   photo_id,
5   count(user_id) as num_of_likes
6   from likes left join users
7     on likes.user_id = users.id
8   group by photo_id
9   order by num_of_likes desc limit 5;
```

The Results tab displays the following data:

id	username	photo_id	num_of_likes
3	Harley_Lind18	149	43
2	Andre_Purdy85	127	43
3	Harley_Lind18	182	43
5	Aniya_Hackett	123	42
3	Harley_Lind18	61	41

The bottom pane shows the Action Output with the following messages:

Time	Action	Message	Duration / Fetch
S2 15:54:09	select id, username, photo_id, count(user_id) as num_of_likes from likes left join users on likes.user_id = ...	5 row(s) returned	0.015 sec / 0.000 sec
S3 15:54:21	select id, username, photo_id, count(user_id) as num_of_likes from likes left join users on likes.user_id = ...	5 row(s) returned	0.016 sec / 0.000 sec

We query the tables **likes** and **users** to get the highest like on a photo. We are joining the two tables and using the following query for this

“select id, username, photo_id, count(user_id) as num_of_likes from likes left join users on likes.user_id = users.id group by photo_id order by number_of_likes desc limit 5”

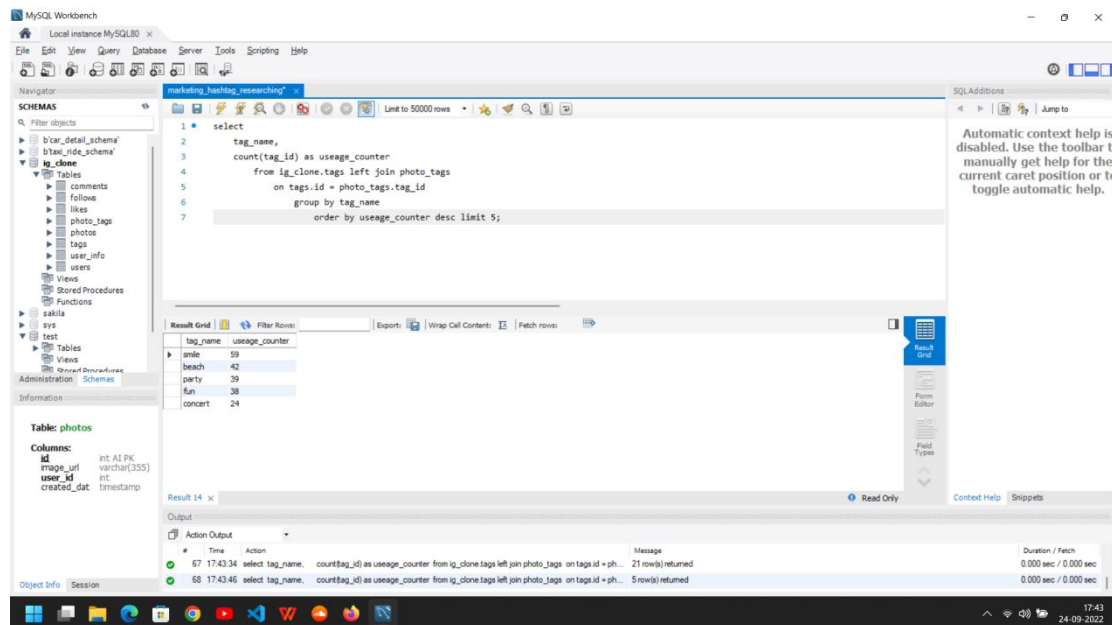
Note: We use aggregate function count and then group by to count the number of likes on each photo.

We forward this information to marketing team to declare the contest winners and reward other popular users.

4. Hashtag Researching: A partner brand wants to know, which hashtags to use in the post to reach the most people on the platform.

Step 1:

We need to find out the most used tags in the post:



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

```
1 select
2   tag_name,
3   count(tag_id) as useage_counter
4   from ig_clone.tags left join photo_tags
5     on tags.id = photo_tags.tag_id
6   group by tag_name
7   order by useage_counter desc limit 5;
```

The Results grid displays the following data:

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

The bottom panel shows the Action Output with the following messages:

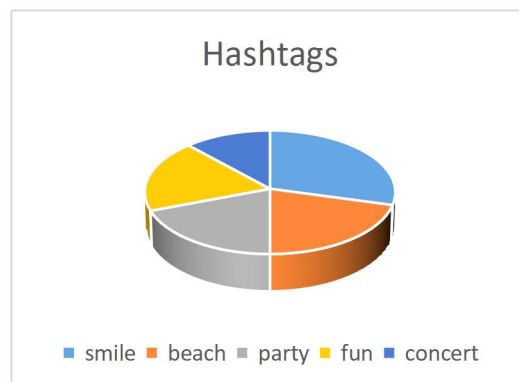
```
67 17:43:34 select tag_name, count(tag_id) as useage_counter from ig_clone.tags left join photo_tags on tags.id = ph... 21 row(s) returned
68 17:43:45 select tag_name, count(tag_id) as useage_counter from ig_clone.tags left join photo_tags on tags.id = ph... 5 row(s) returned
```

Now we use tables **tags** and **photo_tags** to answer this question.

We use this query to answer this question

“select tag_name, count(tag_id) as useage_counter from ig_clone.tags left join photo_tags on tags.id = photo_tags.tag_id group by tag_name order by useage_counter desc limit 5”

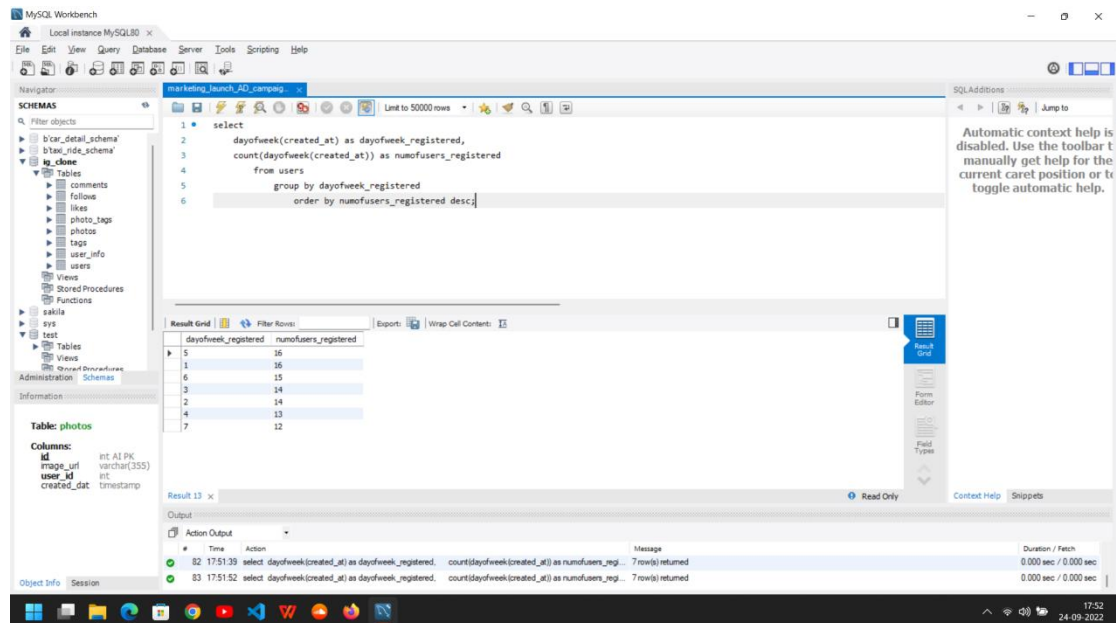
We forward our finding to partner brand to use right kind of hashtags to maximize their impact while using our platform. A chart below shows us the distribution.



5. **Launch AD Campaign:** The team wants to know, which day would be the best day to launch ADs.

Step 1:

To answer this part we want to understand the user registration pattern or the days on which more users join our platform:



We query the table **users** to answer this question.

We use the following query

“select daysofweek(created_at) as daysofweek_registered, count(dayofweek(created_at)) as numberofusers_registered from users group by dayofweek_registered order by numofusers_registered desc”

Let's break down this query:

- We use daysofweek() built-in function to find out the day of the week from a timestamp and group the list.
- We use aggregate function count to number of users and order the list.

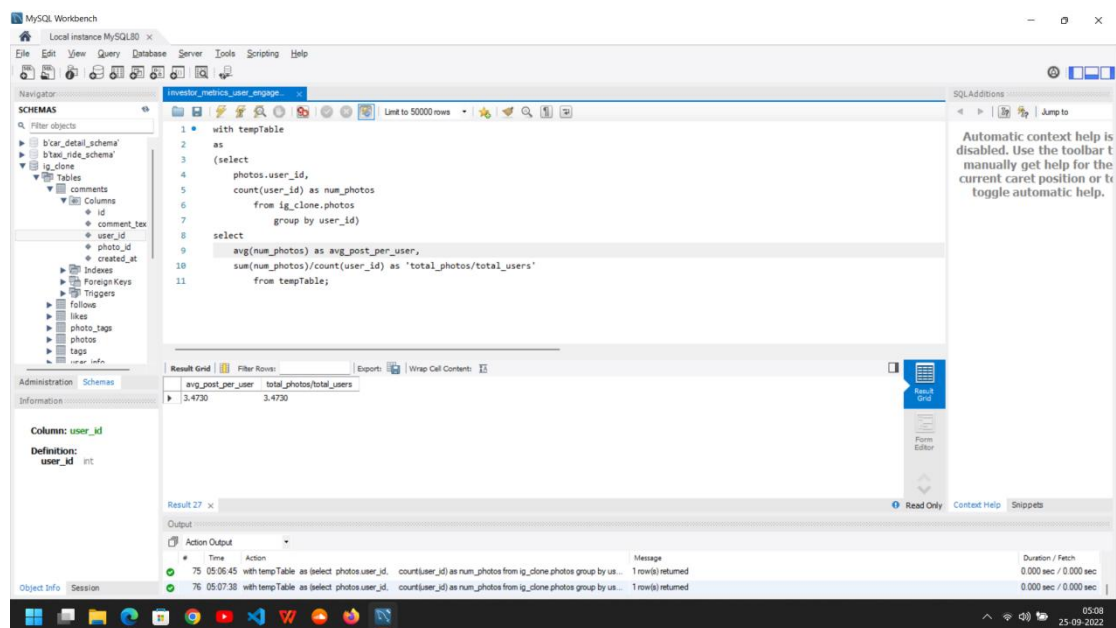
We share our finding with the marketing team which then uses the fact that most of new users register on days 5, 6 and 1 to launch the Advertisement Campaign.

B) Investor Metrics: Our investors want to know if Instagram is performing well and is not becoming redundant like Facebook, they want to assess the app on the following grounds

1. **User Engagement:** Are users still as active and post on Instagram or they are making fewer posts

Step 1:

To answer this question we need to find out post to user ratio:



We use the following query

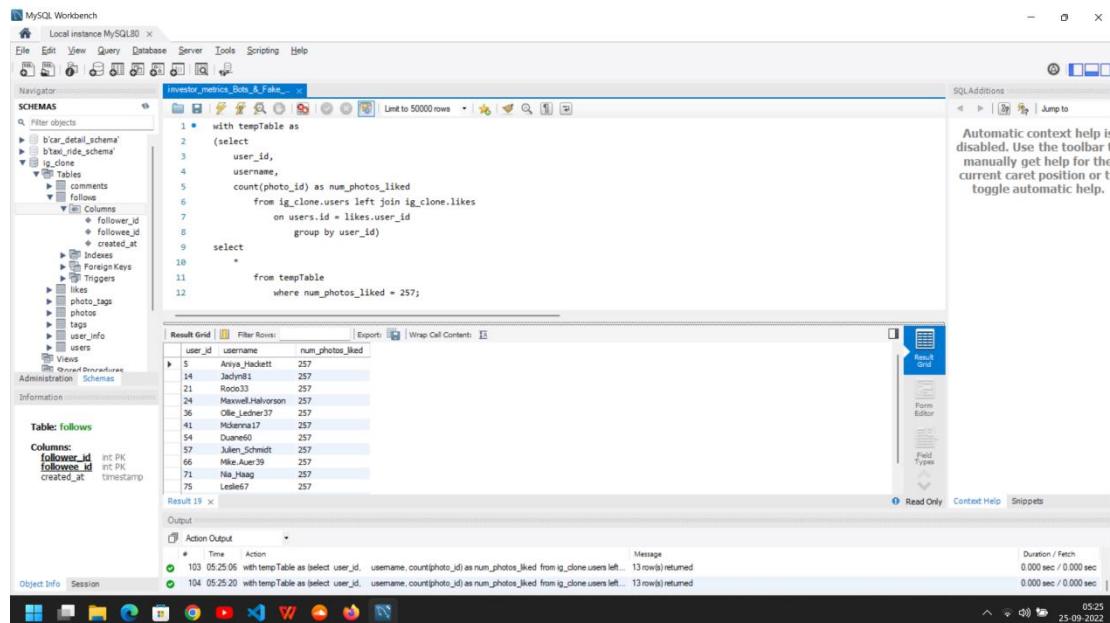
“with tempTable as (select photos.user_id, count(user_id) as num_photos from ig_clone.photos group by user_id) select avg(num_photos) as avg_post_per_user, sum(num_photos)/count(user_id) as 'total_photos/total_users' from tempTable;”

We see on average a user is still posting at least 3 posts. We share the finding with investors.

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

Step 1:

To answer this question we use tables **users** and **likes** to create a temporary table using a 'with' clause:



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

```
1 with tempTable as
2 (select
3   user_id,
4   username,
5   count(photo_id) as num_photos_liked
6   from ig_clone.users left join ig_clone.likes
7   on users.id = likes.user_id
8   group by user_id)
9 select
10  *
11  from tempTable
12  where num_photos_liked = 257;
```

The Results Grid shows the following data:

user_id	username	num_photos_liked
5	Anya_Hackett	257
14	JodyM1	257
21	Roco33	257
24	Maxwell_Halverson	257
26	Oke_Lesher37	257
41	Mikenna17	257
54	Duane60	257
57	Julien_Schmidt	257
66	Mike_Auer39	257
71	Nia_Hagg	257
75	Leslie67	257

The Action Output pane shows the execution of the query, indicating that 13 rows were returned.

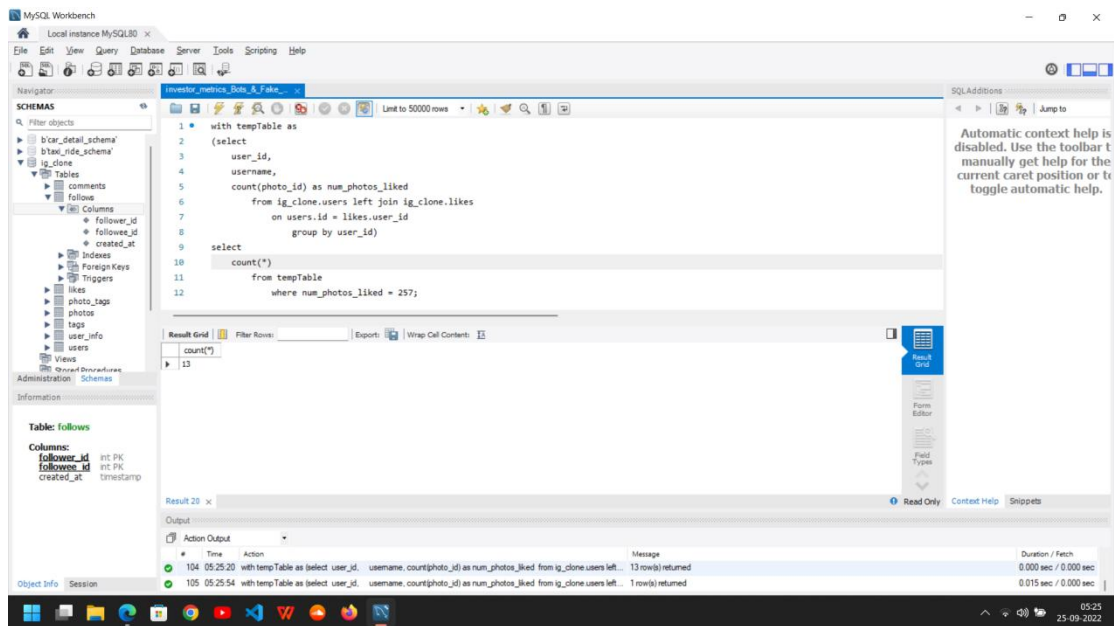
The logic here is to find users with unusual behaviour like liking every single photo on the platform.

We use the following query

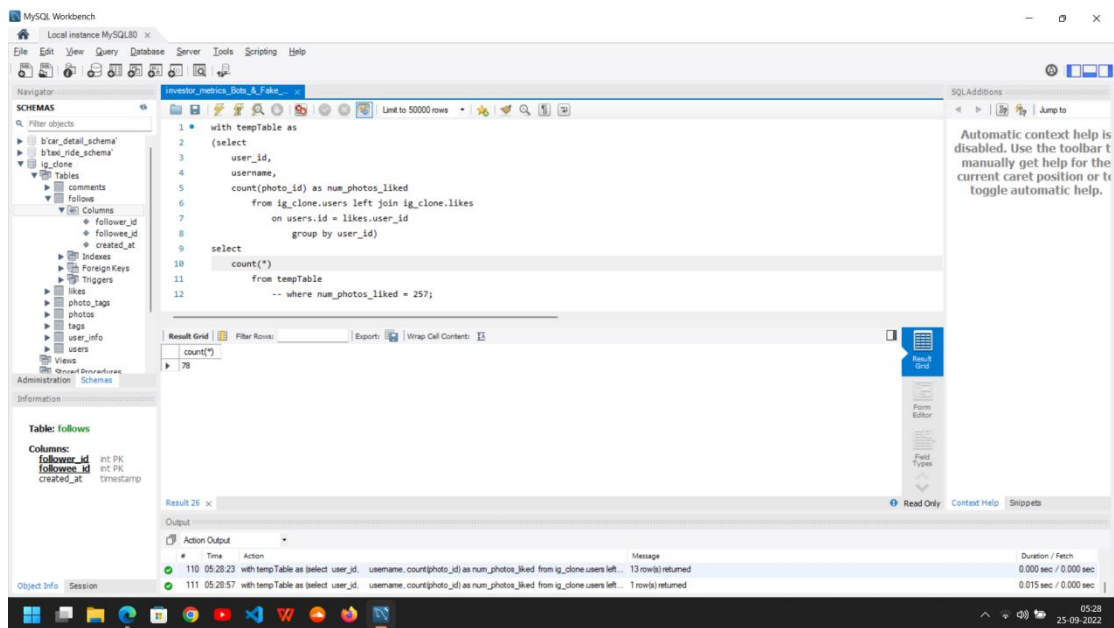
“use ig_clone;

with tempTable as (select user_id, username, count(photo_id) as num_photos_liked from ig_clone.users left join ig_clone.likes on users.id = likes.user_id group by user_id) select count(*) from tempTable where num_photos_liked = 257;”

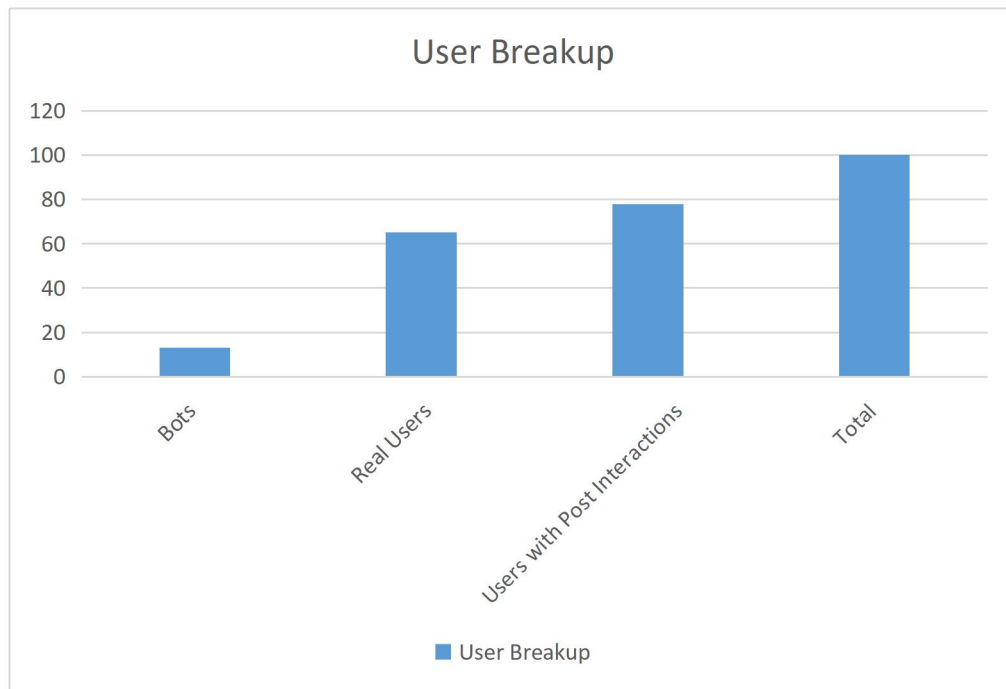
We modify the query a little to count the number of bots:



We then count total number of users total number of users who have liked posts:



We share our finding with investors, we have 13 bots out of 100 users.



This brings us to the end of this project.

We learned how to use basic SQL commands to query database.