



# INSTAGRAM USER ANALYTICS

# Project description:

- In this project, we are going to perform some SQL queries from the given database for the given questions regarding marketing details and investor metric details

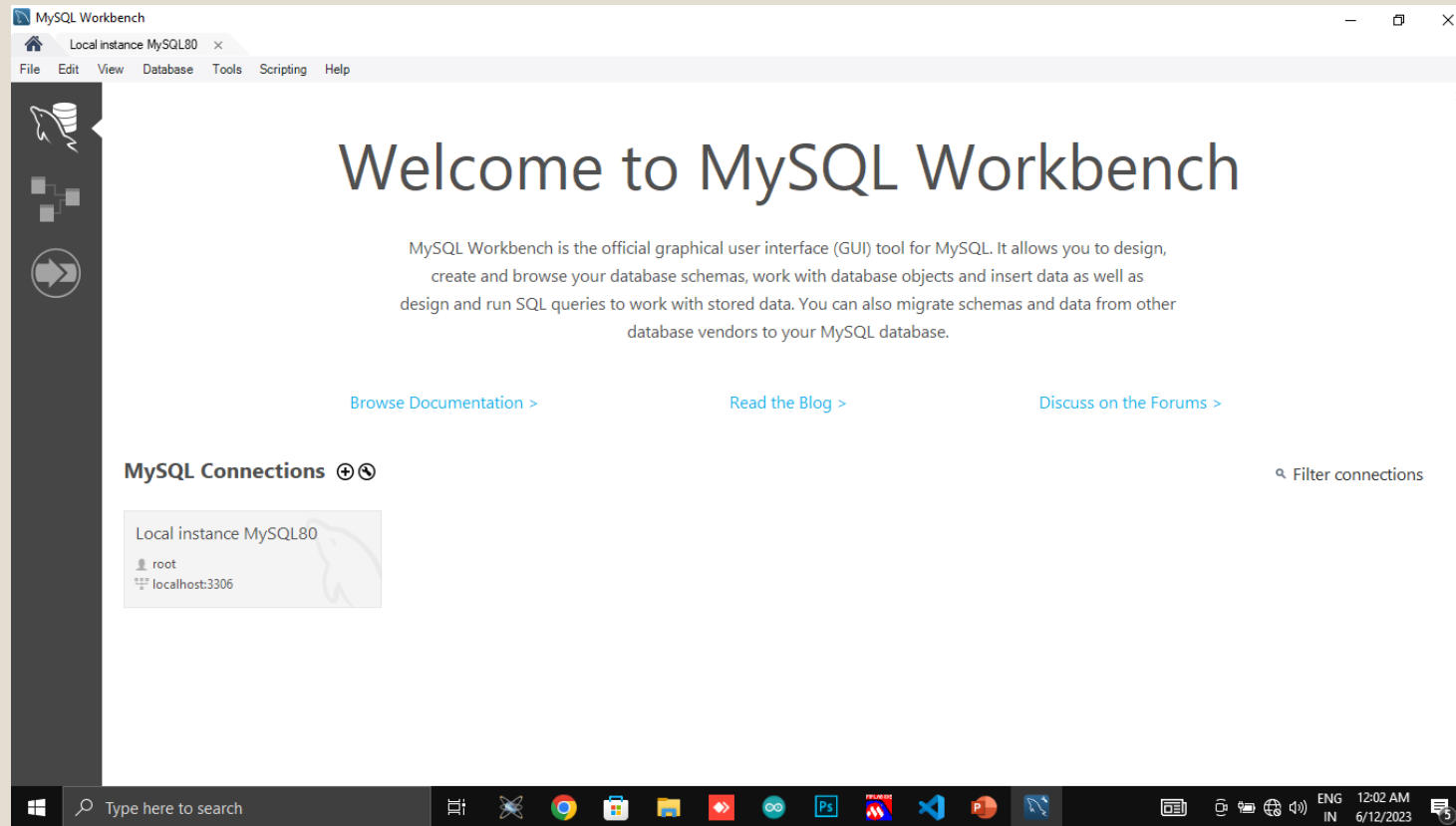
# Approach:

In this project, I am going to access data of multiple table using the concept of “joins”

Which is helpful in manipulating columns of different tables and join them together into single table and them with aliases.

# Tech stack used:

- For this project I have used “MySQL workbench” to perform the queries.



# Insights:

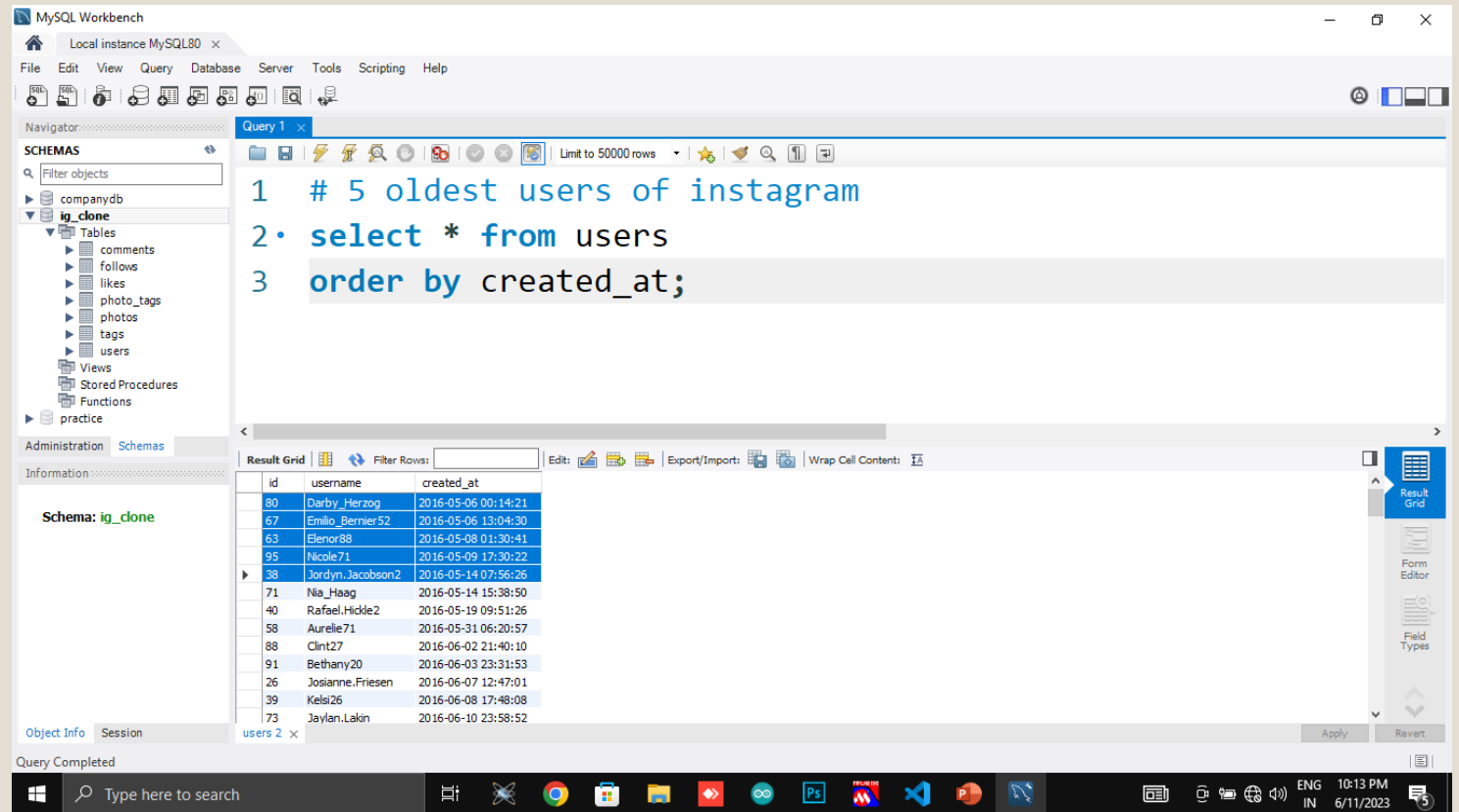
- Through this project, I gained deep understanding of the given database to perform any queries.
- I also learned how to perform subqueries and joins in-order to perform queries across the tables.

# Result:

- Through this project I have achieved the knowledge of writing some complex queries.

# Rewarding most loyal users:

To rewarding the most loyal users Based on the top five oldest user , I have used “order by” clause for Created\_at column to find oldest users.



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'companydb' expanded, showing tables like 'comments', 'follows', 'likes', 'photo\_tags', 'photos', 'tags', and 'users'. The main query editor contains the following SQL code:

```
1 # 5 oldest users of instagram
2 select * from users
3 order by created_at;
```

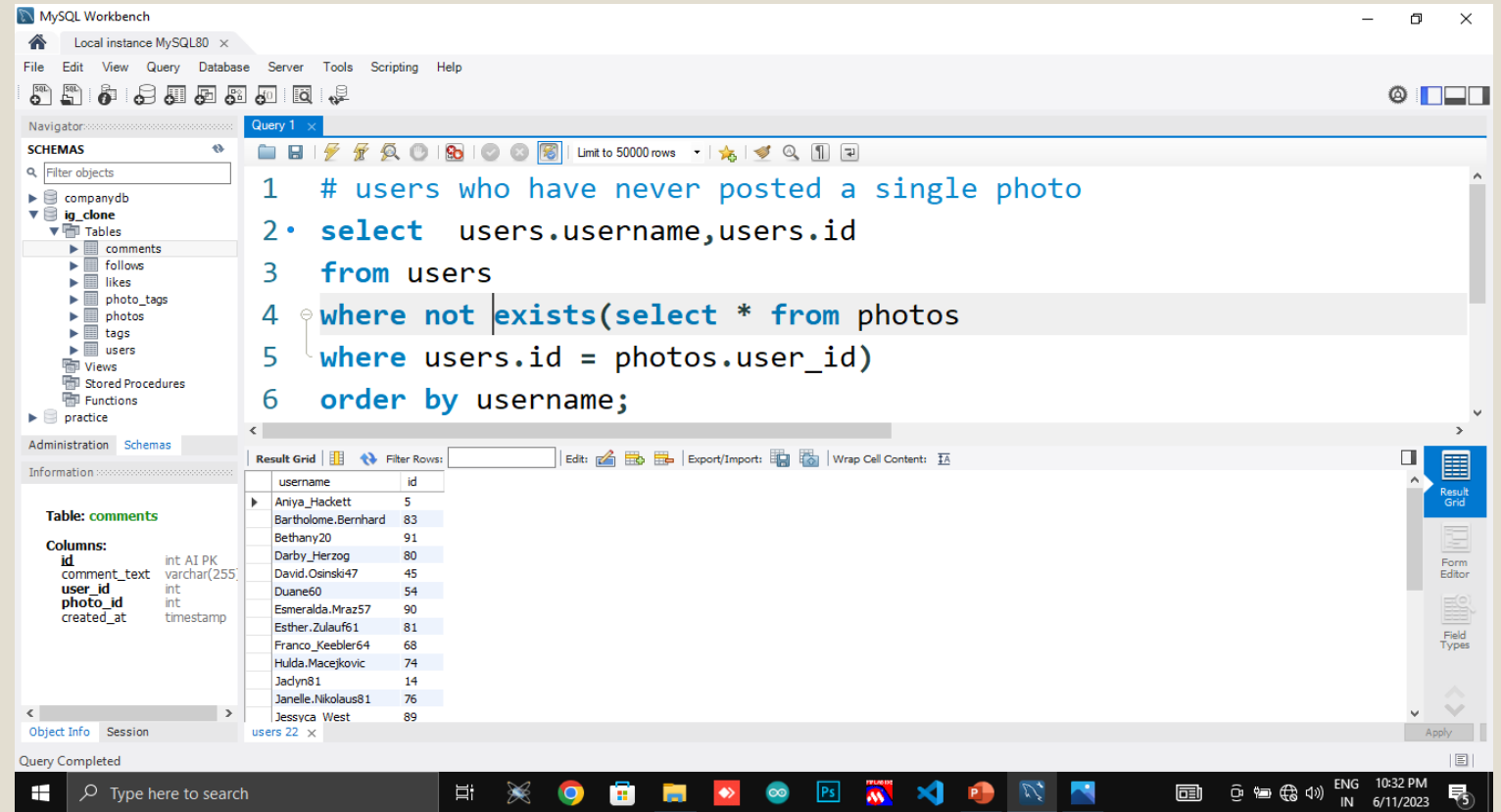
The 'Result Grid' at the bottom shows the output of the query, displaying a list of users ordered by their 'created\_at' date. The results are as follows:

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
71	Nia_Haag	2016-05-14 15:38:50
40	Rafael.Hiddle2	2016-05-19 09:51:26
58	Aurelie71	2016-05-31 06:20:57
88	Clint27	2016-06-02 21:40:10
91	Bethany20	2016-06-03 23:31:53
26	Josianne.Friesen	2016-06-07 12:47:01
39	Kelsi26	2016-06-08 17:48:08
73	Jaylan.Lakin	2016-06-10 23:58:52

The bottom status bar indicates 'Query Completed' and the system clock shows '10:13 PM 6/11/2023'.

# Remind inactive users to start posting:

In this I have selected the users Who have never posted a single Photo using a subquery in which I have compared users table And photos table.



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'companydb' expanded, showing tables like 'comments', 'follows', 'likes', 'photo\_tags', 'photos', 'tags', and 'users'. The main editor window contains a SQL query labeled 'Query 1'.

```
1 # users who have never posted a single photo
2 • select users.username,users.id
3 from users
4 where not exists(select * from photos
5 where users.id = photos.user_id)
6 order by username;
```

Below the query editor, the 'Result Grid' shows the output of the query, displaying a list of users with their usernames and IDs. The results are sorted by username.

username	id
Aniya_Hackett	5
Bartholome.Bernhard	83
Bethany20	91
Darby_Herzog	80
David.Osinski47	45
Duane60	54
Esmeralda.Mraz57	90
Esther.Zulauf61	81
Franco_Keebler64	68
Hulda.Macejkovic	74
Jadyn81	14
Janelle.Nikolaus81	76
Jessyca West	89

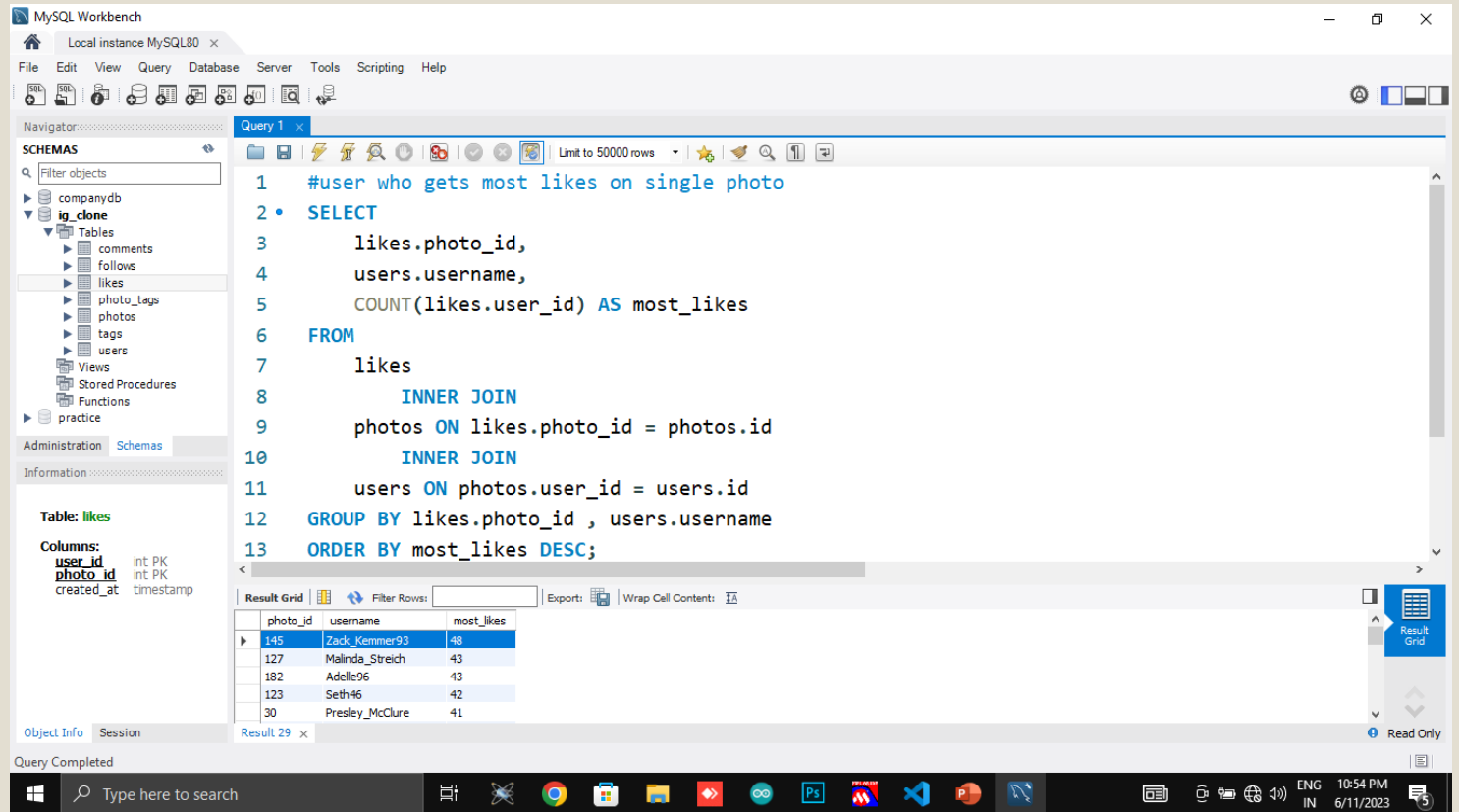
The bottom status bar indicates 'Query Completed'.



# Declaring contest winner:

In this, I have selected the contest winner based on most number of likes On single photo, by counting user\_id column In likes table.

Followed by joining three different of columns of different tables.



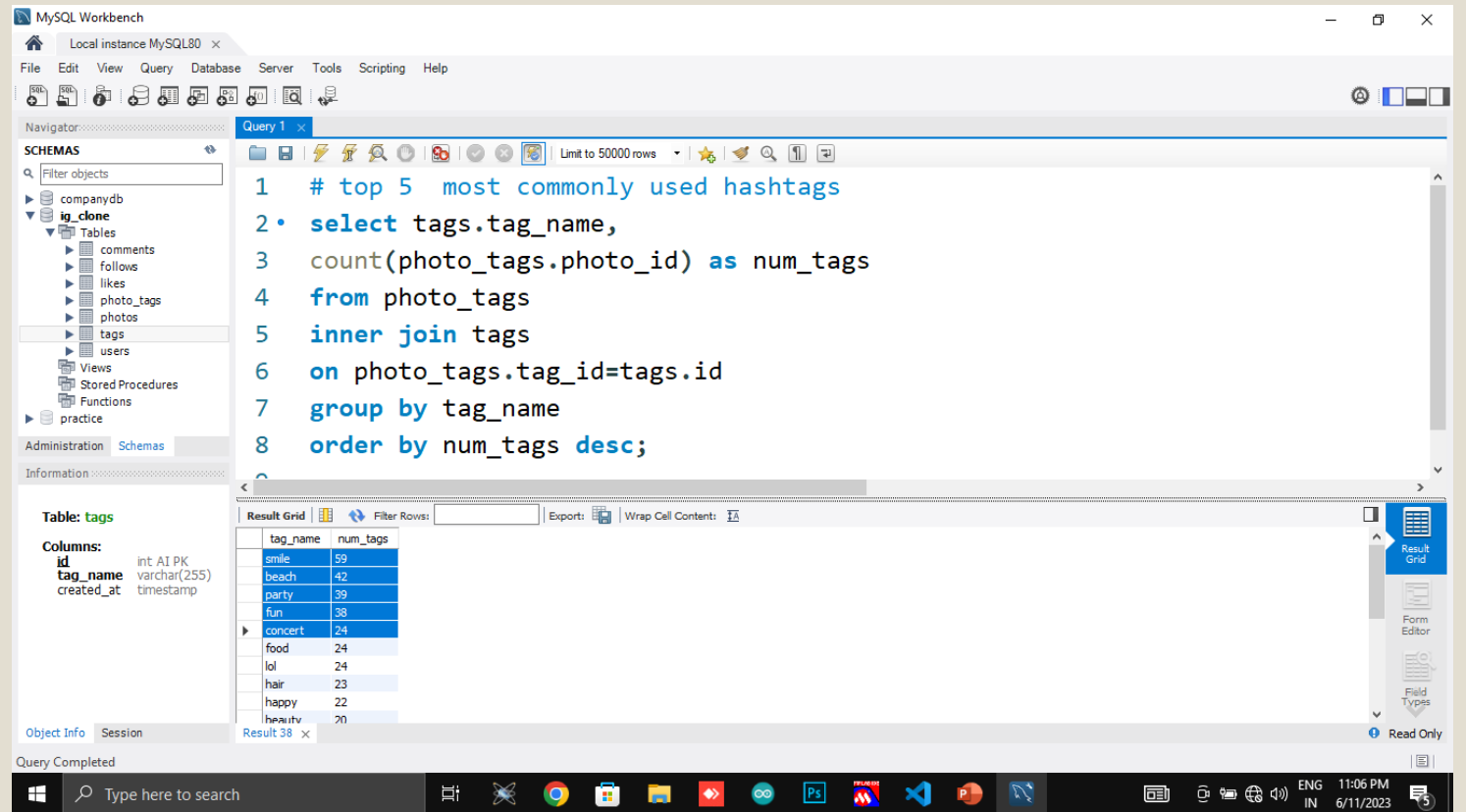
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'companydb' expanded, showing tables like 'comments', 'follows', 'likes', 'photo\_tags', 'photos', 'tags', and 'users'. The 'likes' table is selected, and its columns are listed: 'user\_id' (int PK), 'photo\_id' (int PK), and 'created\_at' (timestamp). The main editor shows a SQL query to find the user with the most likes on a single photo. The query uses an inner join between the 'likes' and 'photos' tables, groups the results by photo\_id and username, and orders them by the count of likes in descending order. The 'Result Grid' at the bottom shows the top results.

```
1 #user who gets most likes on single photo
2 • SELECT
3     likes.photo_id,
4     users.username,
5     COUNT(likes.user_id) AS most_likes
6 FROM
7     likes
8     INNER JOIN
9     photos ON likes.photo_id = photos.id
10    INNER JOIN
11    users ON photos.user_id = users.id
12 GROUP BY likes.photo_id , users.username
13 ORDER BY most_likes DESC;
```

photo_id	username	most_likes
145	Zack_Kemmer93	48
127	Malinda_Streich	43
182	Adelle96	43
123	Seth46	42
30	Presley_McClure	41

# Hashtag researching:

In this, I have found top five most commonly used hashtags by inner joining tags and photo\_tags Table and counting the photo\_id of photos\_tags.



The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a database named 'companydb' with a table 'photo\_tags' selected. The 'Table: tags' information pane shows columns: 'id' (int AI PK), 'tag\_name' (varchar(255)), and 'created\_at' (timestamp). The central query editor contains the following SQL code:

```
1 # top 5 most commonly used hashtags
2 • select tags.tag_name,
3 count(photo_tags.photo_id) as num_tags
4 from photo_tags
5 inner join tags
6 on photo_tags.tag_id=tags.id
7 group by tag_name
8 order by num_tags desc;
```

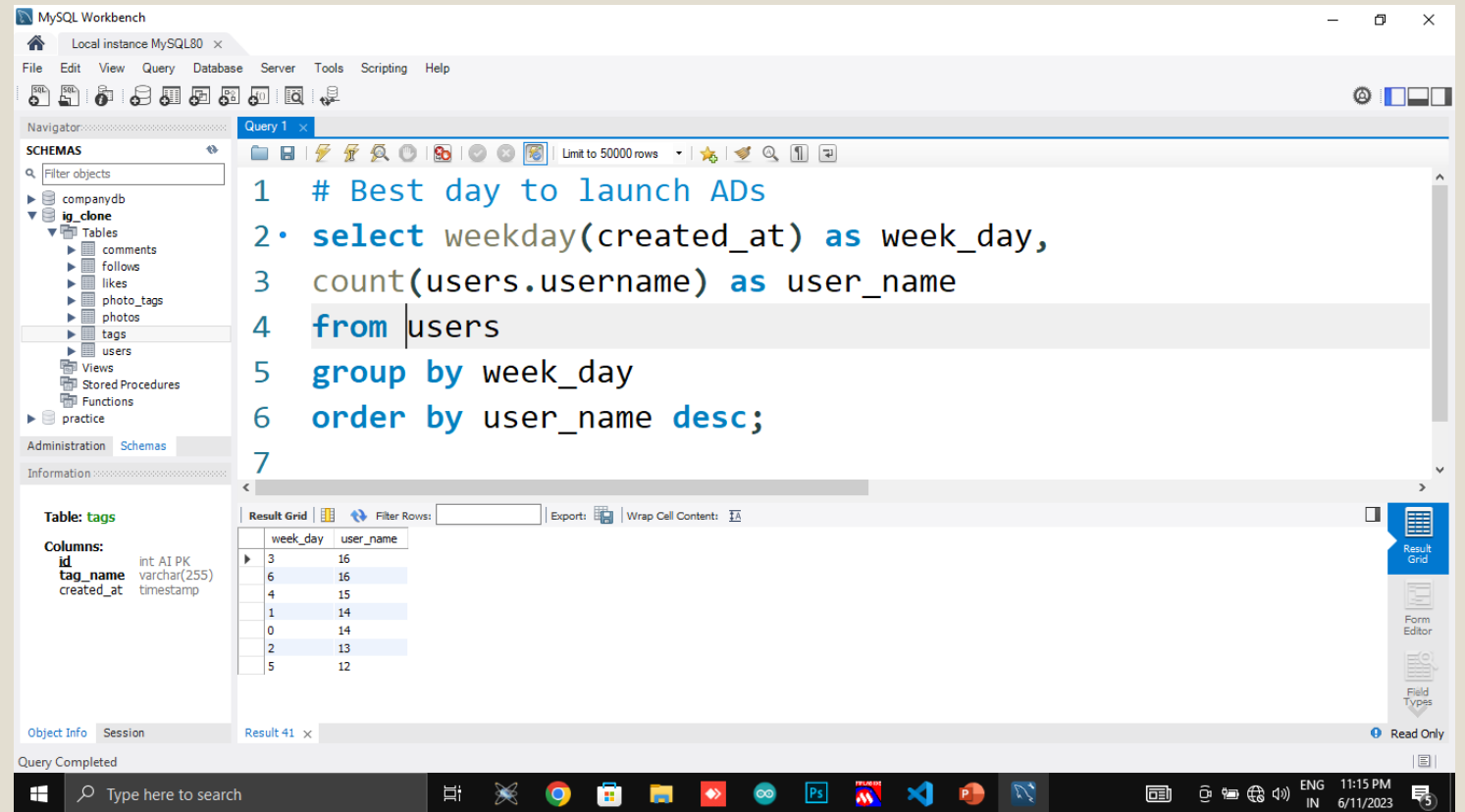
The 'Result Grid' at the bottom shows the results of the query, sorted by the number of tags in descending order:

tag_name	num_tags
smile	59
beach	42
party	39
fun	38
concert	24
food	24
lol	24
hair	23
happy	22
health	20

The Windows taskbar at the bottom shows the system clock as 11:06 PM on 6/11/2023.

# Best day to launch ADs:

In this, I have used weekday() function to find which day users most registered and counting username in users table.



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'companydb' expanded, showing tables like 'comments', 'follows', 'likes', 'photo\_tags', 'photos', 'tags', and 'users'. The 'Information' tab is active, showing details for the 'tags' table. The main query editor contains the following SQL code:

```
1 # Best day to launch ADs
2 • select weekday(created_at) as week_day,
3   count(users.username) as user_name
4 from users
5 group by week_day
6 order by user_name desc;
7
```

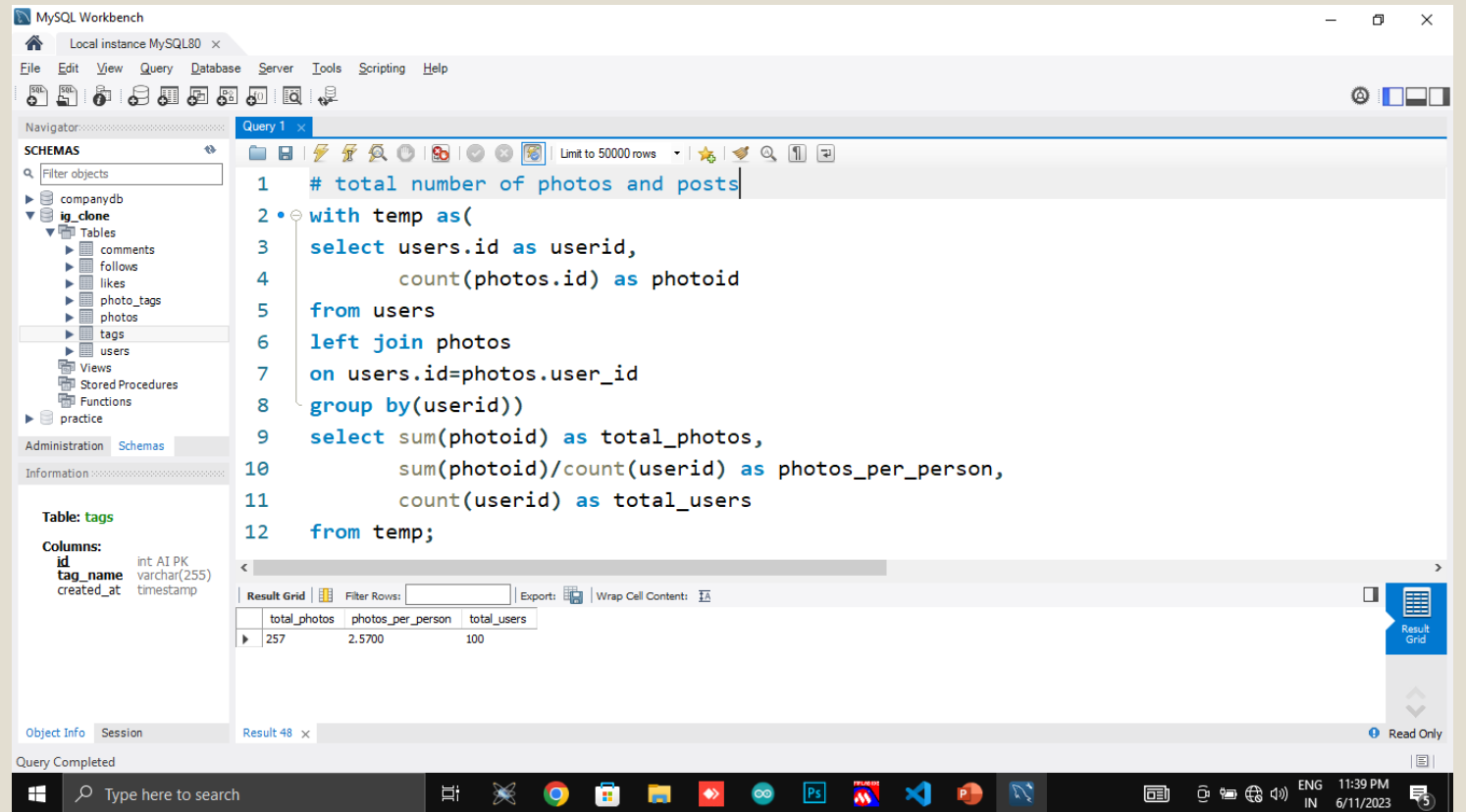
The 'Result Grid' at the bottom shows the following data:

week_day	user_name
3	16
6	16
4	15
1	14
0	14
2	13
5	12

The status bar at the bottom indicates 'Query Completed' and 'Result 41 x'. The Windows taskbar at the very bottom shows the date and time as '6/11/2023 11:15 PM'.

# Investor metrics:

In this , I have found how many times does average user posts on instagram by creating a temporary table as “temp” and performing division by Summation of photoid and number of userid.



The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' panel with a tree view of databases and tables. The 'companydb' database is selected, and the 'tags' table is highlighted. The 'Information' panel shows the table's structure: 'id' (int AI PK), 'tag\_name' (varchar(255)), and 'created\_at' (timestamp).

The main query editor shows the following SQL code:

```
1 # total number of photos and posts
2 • with temp as(
3   select users.id as userid,
4         count(photos.id) as photoid
5   from users
6   left join photos
7   on users.id=photos.user_id
8   group by(userid))
9   select sum(photoid) as total_photos,
10         sum(photoid)/count(userid) as photos_per_person,
11         count(userid) as total_users
12   from temp;
```

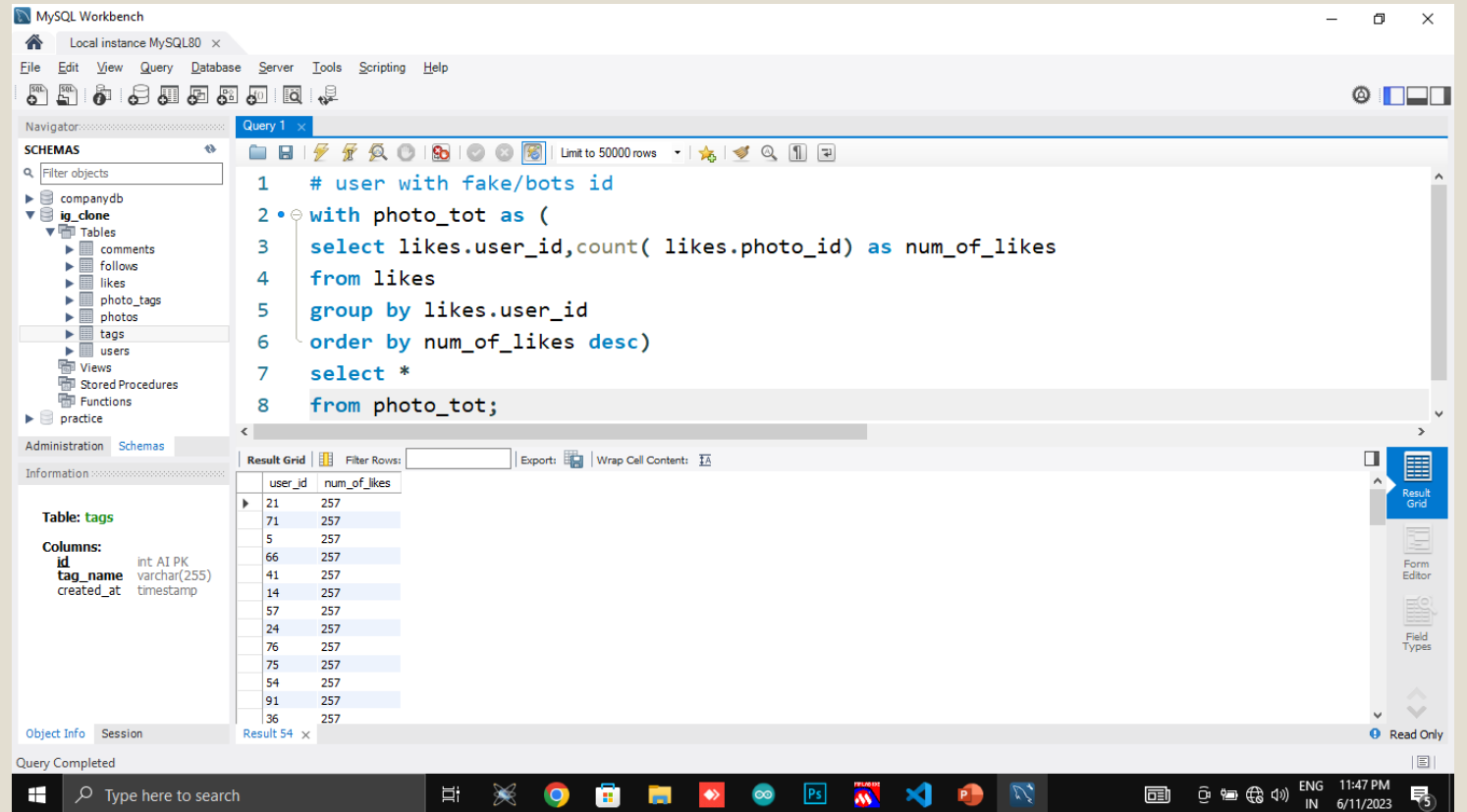
The 'Result Grid' at the bottom shows the output of the query:

total_photos	photos_per_person	total_users
257	2.5700	100

The status bar at the bottom indicates 'Query Completed' and 'Result 48 x'.

# Bots & fake Accounts:

In this, I have counted total number likes per user\_id.



The screenshot displays the MySQL Workbench interface. The 'Query 1' editor contains the following SQL code:

```
1 # user with fake/bots id
2 • with photo_tot as (
3   select likes.user_id, count( likes.photo_id) as num_of_likes
4   from likes
5   group by likes.user_id
6   order by num_of_likes desc)
7 select *
8 from photo_tot;
```

The 'Result Grid' shows the output of the query, displaying two columns: 'user\_id' and 'num\_of\_likes'. The results are sorted in descending order of 'num\_of\_likes'.

user_id	num_of_likes
21	257
71	257
5	257
66	257
41	257
14	257
57	257
24	257
76	257
75	257
54	257
91	257
36	257

The 'Object Info' panel on the left shows the schema for the 'tags' table, with columns: 'id' (int AI PK), 'tag\_name' (varchar(255)), and 'created\_at' (timestamp).

Query Completed

# Conclusion:

- In this project, I have used my knowledge based on the learnings which are provided by the trainity.

THANK YOU