Instagram User Analytics

Description

Overview:

The project aims at analysing various interactions that users prefer to have with the Instagram App and how they get engaged with the app.

Objectives:

Developing valuable insights for usage by various teams within the business and assist the product manager and team for making informed decisions about the future direction of the app.

Deliverables:

Providing conclusions about how the users interact and engage with the app, thereby, giving assisting the various teams to form strategies to increase the user-base for the app and provide basis for other future developments that could potentially take place.

Approach

All necessary discussions on how these questions were approached are discussed in detail in each question.

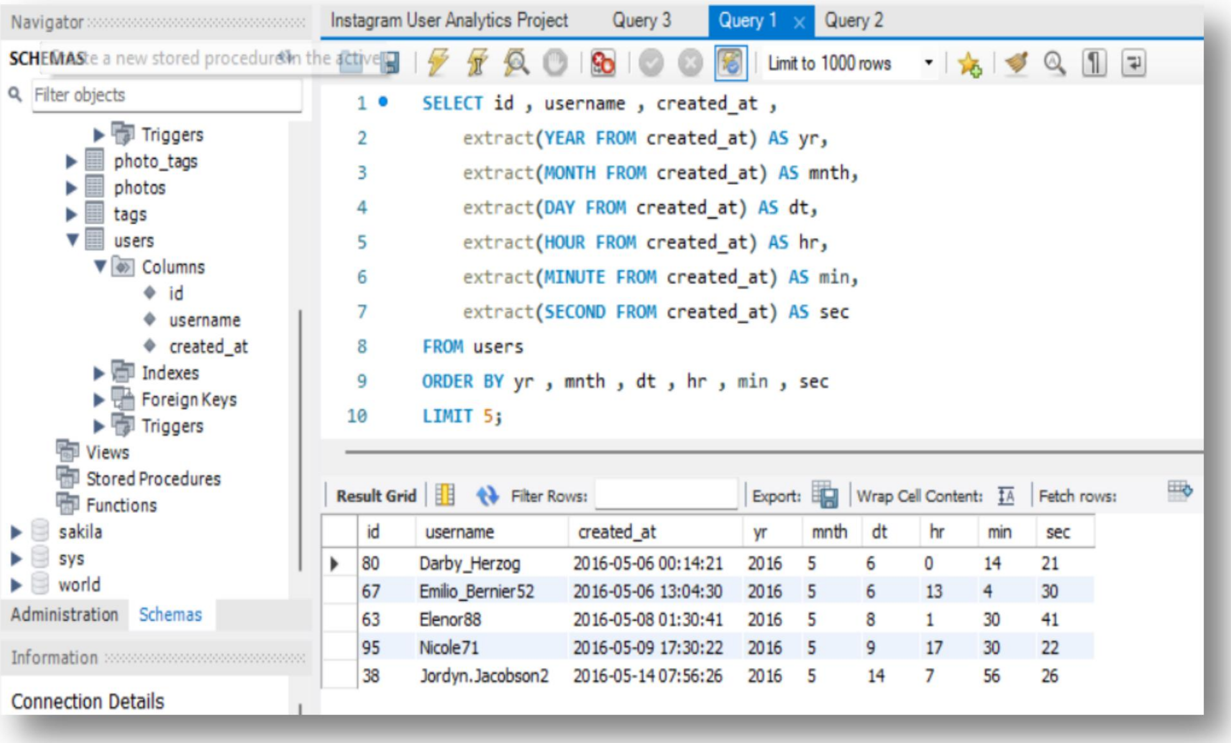
Tech-Stack

MySQL Workbench is used for performing all the queries on the dataset in order to answer the questions.

Marketing Analysis

1. **Loyal User Reward:** The marketing team wants to reward the most loyal users, i.e., those who have been using the platform for the longest time.
   The following query was used initially:

```
SELECT id , username , created_at ,
        extract(YEAR FROM created_at) AS yr,
        extract(MONTH FROM created_at) AS mnth,
         extract(DAY FROM created_at) AS dt,
        extract(HOUR FROM created_at) AS hr,
        extract(MINUTE FROM created_at) AS min,
        extract(SECOND FROM created_at) AS sec
FROM users
ORDER BY yr , mnth , dt , hr , min , sec
LIMIT 5;
```

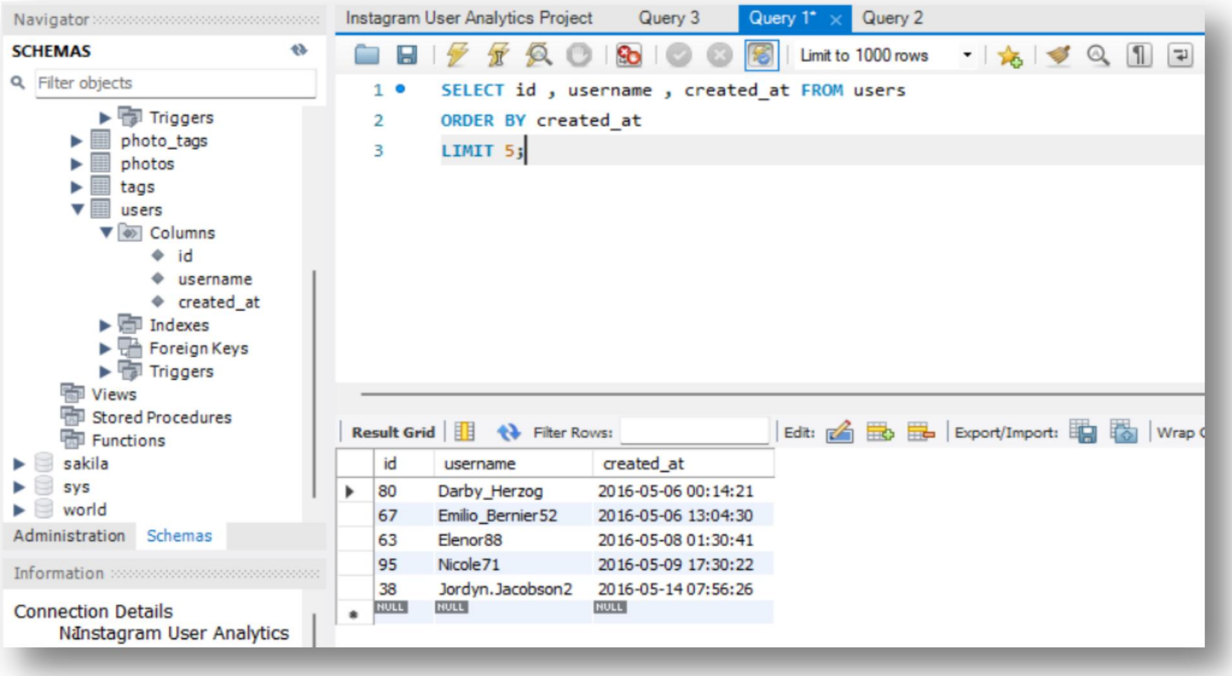


```
SELECT id , username , created_at ,
    extract(YEAR FROM created_at) AS yr,
    extract(MONTH FROM created_at) AS mnth,
    extract(DAY FROM created_at) AS dt,
    extract(HOUR FROM created_at) AS hr,
    extract(MINUTE FROM created_at) AS min,
    extract(SECOND FROM created_at) AS sec
FROM users
ORDER BY yr , mnth , dt , hr , min , sec
LIMIT 5;
```

id	username	created_at	yr	mnth	dt	hr	min	sec
80	Darby_Herzog	2016-05-06 00:14:21	2016	5	6	0	14	21
67	Emilio_Bernier52	2016-05-06 13:04:30	2016	5	6	13	4	30
63	Elenor88	2016-05-08 01:30:41	2016	5	8	1	30	41
95	Nicole71	2016-05-09 17:30:22	2016	5	9	17	30	22
38	Jordyn.Jacobson2	2016-05-14 07:56:26	2016	5	14	7	56	26

Upon further deliberation, it was noticed that the same task could be done by:

```
SELECT id , username , created_at FROM users

ORDER BY created_at

LIMIT 5;
```



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

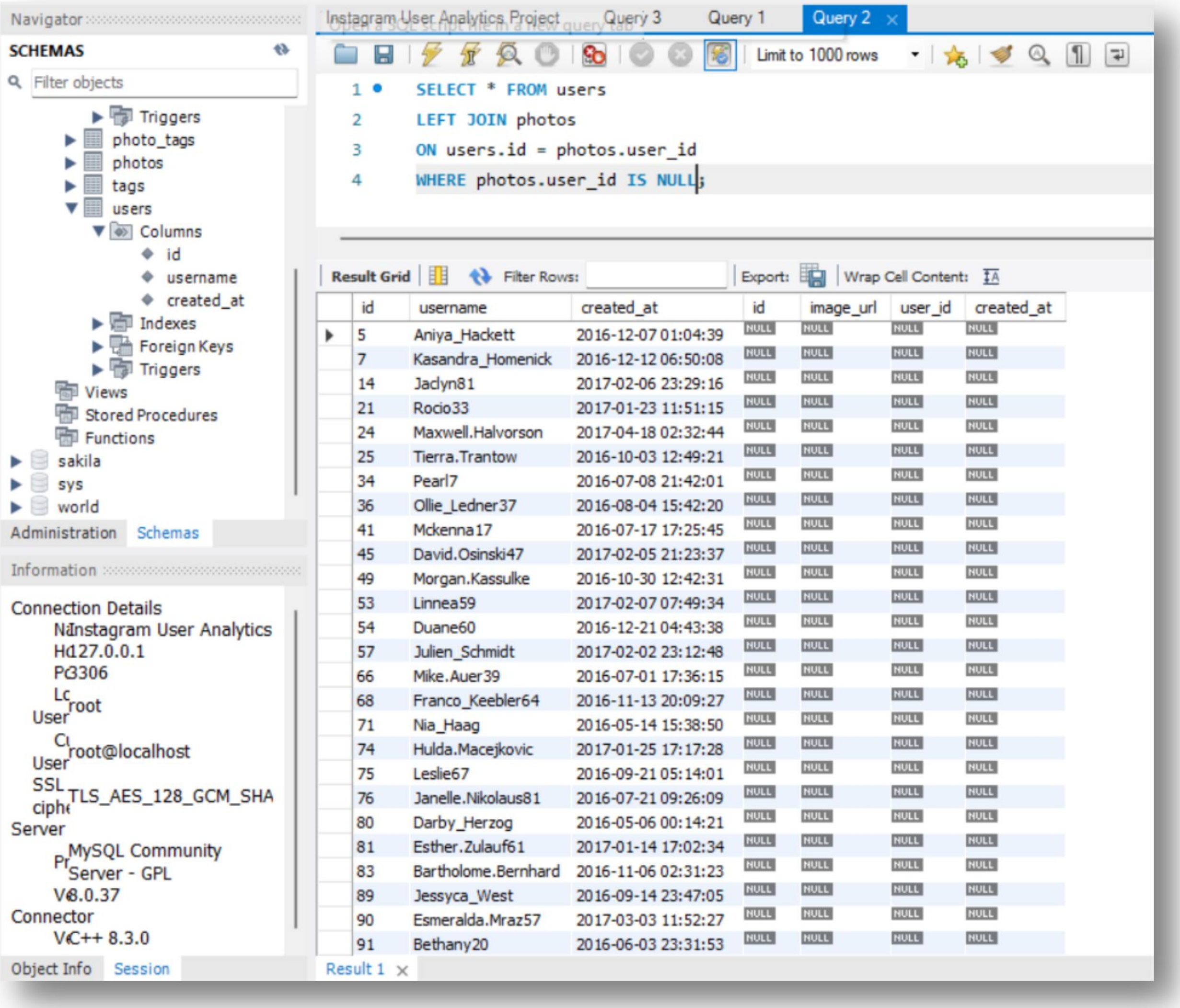
id	username	created_at
80	Darby_Herzog	2016-05-06 00:14:21
67	Emilio_Bernier52	2016-05-06 13:04:30
63	Elenor88	2016-05-08 01:30:41
95	Nicole71	2016-05-09 17:30:22
38	Jordyn.Jacobson2	2016-05-14 07:56:26
NULL	NULL	NULL

Initial thought was that it can be done by arranging the dates as well as the time in ASCENDING order, top 5 of which will be the answer. The answer is indeed correct, however the individual columns that were created could not be eliminated. Using ORDER BY function directly on created_at column eliminated this problem.

The LIMIT function limits the number of rows displayed and is a convenience.

2. **Inactive User Engagement:** The team wants to encourage inactive users to start posting by sending them promotional emails.
The following query was used:

```
SELECT * FROM users

LEFT JOIN photos

ON users.id = photos.user_id

WHERE photos.user_id IS NULL;
```



Connection Details
Na Instagram User Analytics
Ho 127.0.0.1
Po 3306
Lo User root
Cu User root@localhost
SSL ciphe TLS_AES_128_GCM_SHA
Server
Pr MySQL Community Server - GPL
Ve 8.0.37
Connector
Ve C++ 8.3.0

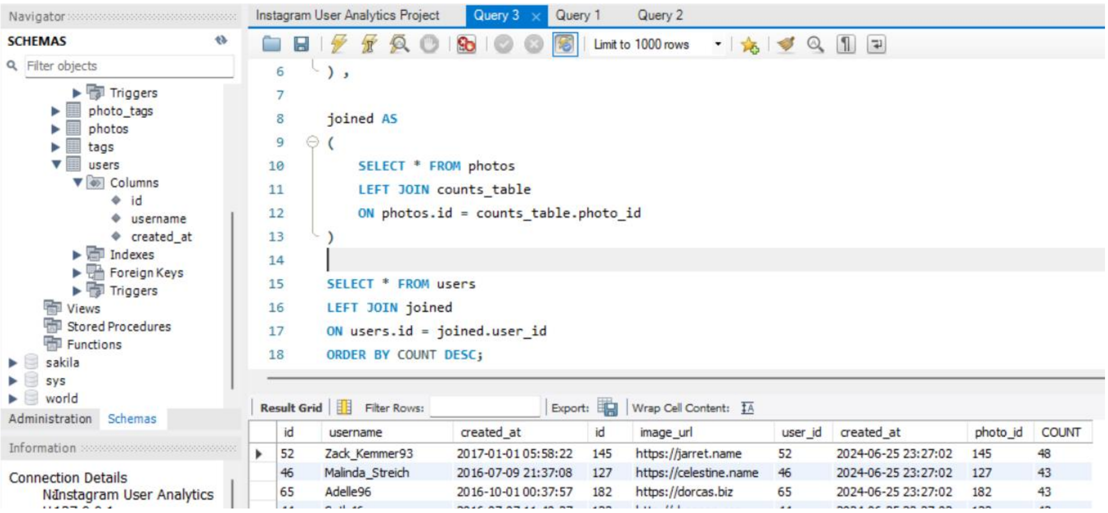
```
SELECT * FROM users
LEFT JOIN photos
ON users.id = photos.user_id
WHERE photos.user_id IS NULL;
```

id	username	created_at	id	image_url	user_id	created_at
5	Aniya_Hackett	2016-12-07 01:04:39	NULL	NULL	NULL	NULL
7	Kasandra_Homenick	2016-12-12 06:50:08	NULL	NULL	NULL	NULL
14	Jaclyn81	2017-02-06 23:29:16	NULL	NULL	NULL	NULL
21	Rocio33	2017-01-23 11:51:15	NULL	NULL	NULL	NULL
24	Maxwell.Halvorson	2017-04-18 02:32:44	NULL	NULL	NULL	NULL
25	Tierra.Trantow	2016-10-03 12:49:21	NULL	NULL	NULL	NULL
34	Pearl7	2016-07-08 21:42:01	NULL	NULL	NULL	NULL
36	Ollie_Ledner37	2016-08-04 15:42:20	NULL	NULL	NULL	NULL
41	Mckenna17	2016-07-17 17:25:45	NULL	NULL	NULL	NULL
45	David.Osinski47	2017-02-05 21:23:37	NULL	NULL	NULL	NULL
49	Morgan.Kassulke	2016-10-30 12:42:31	NULL	NULL	NULL	NULL
53	Linnea59	2017-02-07 07:49:34	NULL	NULL	NULL	NULL
54	Duane60	2016-12-21 04:43:38	NULL	NULL	NULL	NULL
57	Julien_Schmidt	2017-02-02 23:12:48	NULL	NULL	NULL	NULL
66	Mike.Auer39	2016-07-01 17:36:15	NULL	NULL	NULL	NULL
68	Franco_Keebler64	2016-11-13 20:09:27	NULL	NULL	NULL	NULL
71	Nia_Haag	2016-05-14 15:38:50	NULL	NULL	NULL	NULL
74	Hulda.Macejkovic	2017-01-25 17:17:28	NULL	NULL	NULL	NULL
75	Leslie67	2016-09-21 05:14:01	NULL	NULL	NULL	NULL
76	Janelle.Nikolaus81	2016-07-21 09:26:09	NULL	NULL	NULL	NULL
80	Darby_Herzog	2016-05-06 00:14:21	NULL	NULL	NULL	NULL
81	Esther.Zulauf61	2017-01-14 17:02:34	NULL	NULL	NULL	NULL
83	Bartholome.Bernhard	2016-11-06 02:31:23	NULL	NULL	NULL	NULL
89	Jessyca_West	2016-09-14 23:47:05	NULL	NULL	NULL	NULL
90	Esmeralda.Mraz57	2017-03-03 11:52:27	NULL	NULL	NULL	NULL
91	Bethany20	2016-06-03 23:31:53	NULL	NULL	NULL	NULL

The initial thought was the use INNER JOIN, but as soon as the query was run, it was evident that there was no way that the username could be displayed. The next step to use either the LEFT JOIN or RIGHT JOIN was obvious. Above image displays the usernames of all the users who have never posted an image on their accounts.

3. **Contest Winner Declaration:** The team has organized a contest where the user with the most likes on a single photo wins.
The following query was used:

```
WITH counts_table AS
(
        SELECT photo_id , count(*) AS COUNT FROM likes
        GROUP BY photo_id
        ORDER BY COUNT DESC
) ,

joined AS
(
        SELECT * FROM photos
        LEFT JOIN counts_table
        ON photos.id = counts_table.photo_id
)

SELECT * FROM users
LEFT JOIN joined
ON users.id = joined.user_id
ORDER BY COUNT DESC;
```



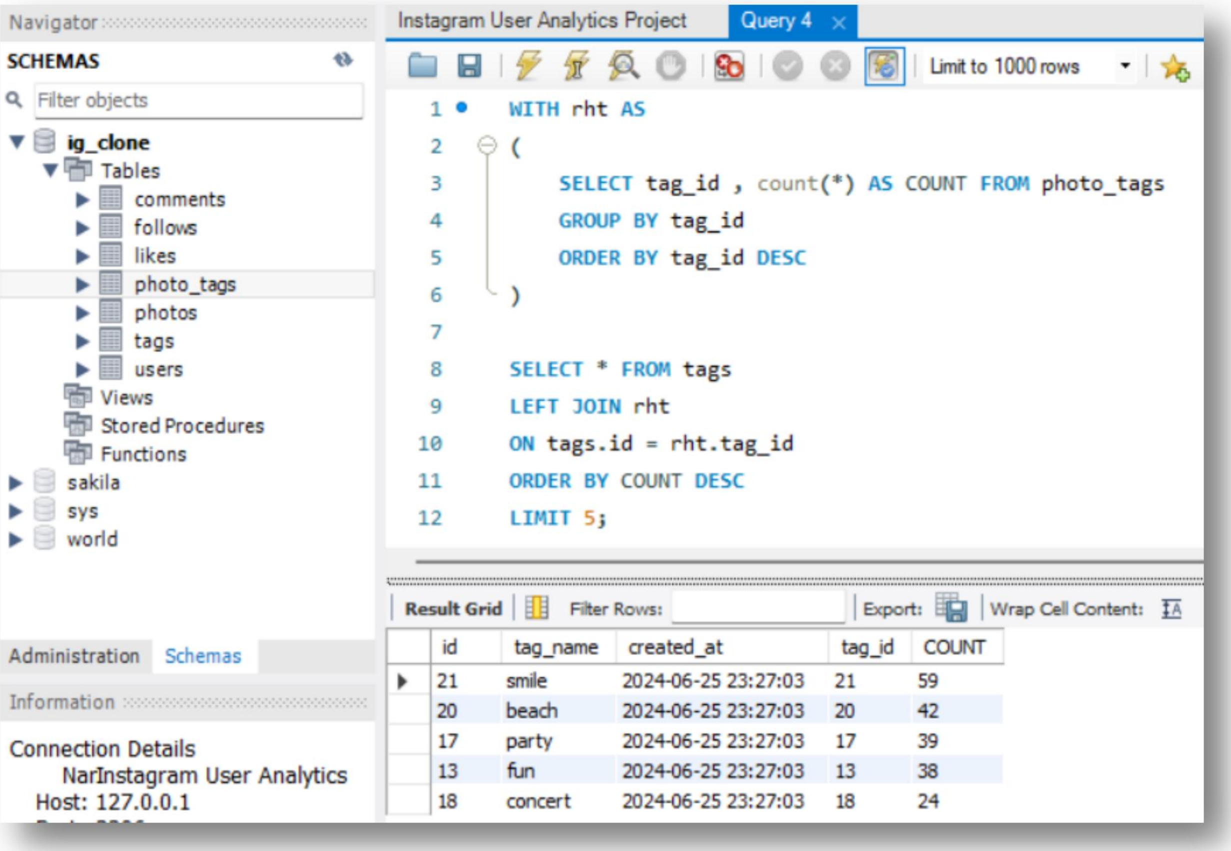
```
) ,

joined AS
(
    SELECT * FROM photos
    LEFT JOIN counts_table
    ON photos.id = counts_table.photo_id
)

SELECT * FROM users
LEFT JOIN joined
ON users.id = joined.user_id
ORDER BY COUNT DESC;
```

id	username	created_at	id	image_url	user_id	created_at	photo_id	COUNT
52	Zack_Kemmer93	2017-01-01 05:58:22	145	https://jarret.name	52	2024-06-25 23:27:02	145	48
46	Malinda_Streich	2016-07-09 21:37:08	127	https://celestine.name	46	2024-06-25 23:27:02	127	43
65	Adelle96	2016-10-01 00:37:57	182	https://dorcas.biz	65	2024-06-25 23:27:02	182	43

The winner of this contest is **Zack_Kemmer93** with id 145 and on a single photo, has maximum likes count of **48**.

4. **Hashtag Research:** A partner brand wants to know the most popular hashtags to use in their posts to reach the most people.
The following query was used:

```
WITH rht AS

(

        SELECT tag_id , count(*) AS COUNT FROM photo_tags

        GROUP BY tag_id

        ORDER BY tag_id DESC

)


SELECT * FROM tags

LEFT JOIN rht

ON tags.id = rht.tag_id

ORDER BY COUNT DESC

LIMIT 5;
```



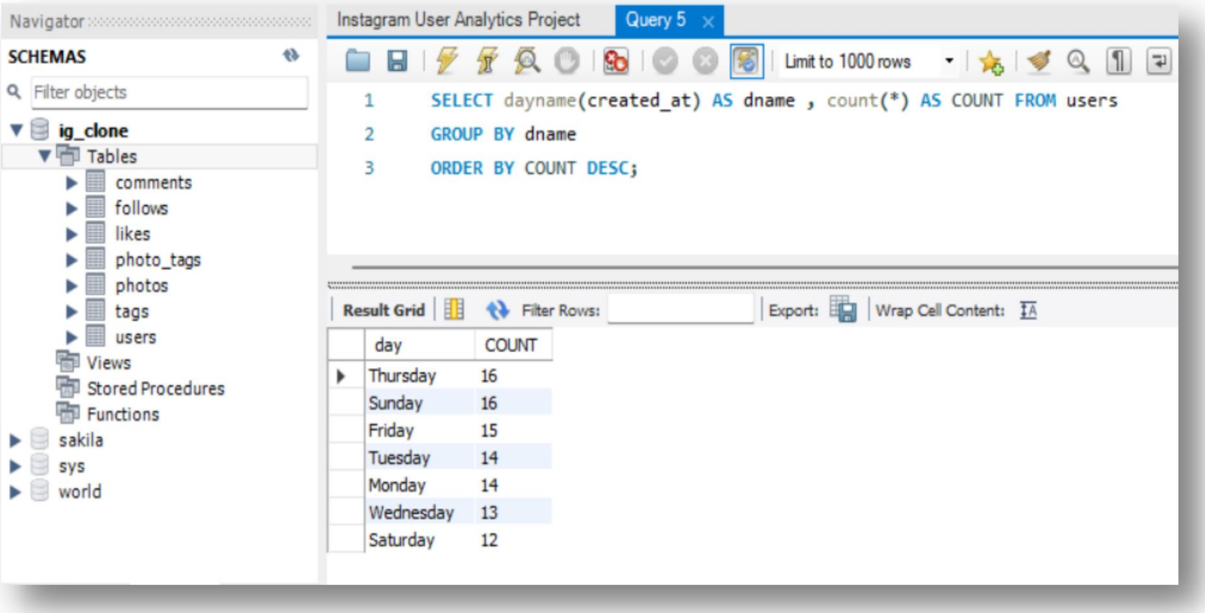
```
WITH rht AS
(
    SELECT tag_id , count(*) AS COUNT FROM photo_tags
    GROUP BY tag_id
    ORDER BY tag_id DESC
)

SELECT * FROM tags
LEFT JOIN rht
ON tags.id = rht.tag_id
ORDER BY COUNT DESC
LIMIT 5;
```

id	tag_name	created_at	tag_id	COUNT
21	smile	2024-06-25 23:27:03	21	59
20	beach	2024-06-25 23:27:03	20	42
17	party	2024-06-25 23:27:03	17	39
13	fun	2024-06-25 23:27:03	13	38
18	concert	2024-06-25 23:27:03	18	24

The top 5 entries are 1. smile 2. beach 3. party 4. fun 5. concert

5. **Ad Campaign Launch:** The team wants to know the best day of the week to launch ads.
   The following query was used:

```
SELECT dayname(created_at) AS dname , count(*) AS COUNT FROM users

GROUP BY dname

ORDER BY COUNT DESC;
```



```
SELECT dayname(created_at) AS dname , count(*) AS COUNT FROM users
GROUP BY dname
ORDER BY COUNT DESC;
```

day	COUNT
Thursday	16
Sunday	16
Friday	15
Tuesday	14
Monday	14
Wednesday	13
Saturday	12

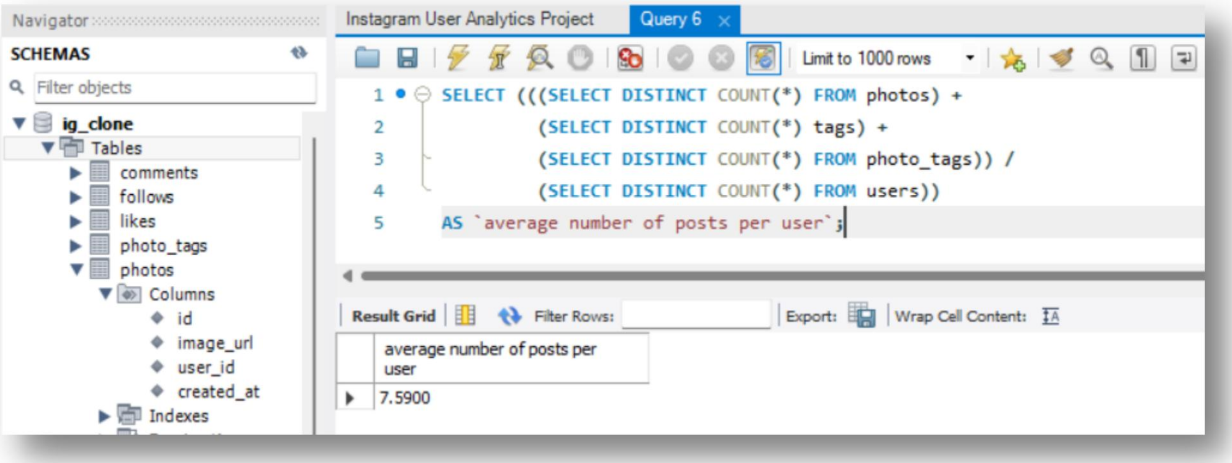
I got to learn about the DAYNAME function for directly converting a date to its day. **Thursday** and **Sunday** can be the best days for launching the campaign.

Investor Metrics

1. **User Engagement:** Investors want to know if users are still active and posting on Instagram or if they are making fewer posts.
The following queries were used:

```
SELECT (((SELECT DISTINCT COUNT(*) FROM photos) +
            (SELECT DISTINCT COUNT(*) tags) +
            (SELECT DISTINCT COUNT(*) FROM photo_tags)) /
            (SELECT DISTINCT COUNT(*) FROM users))
AS `average number of posts per user`;
```

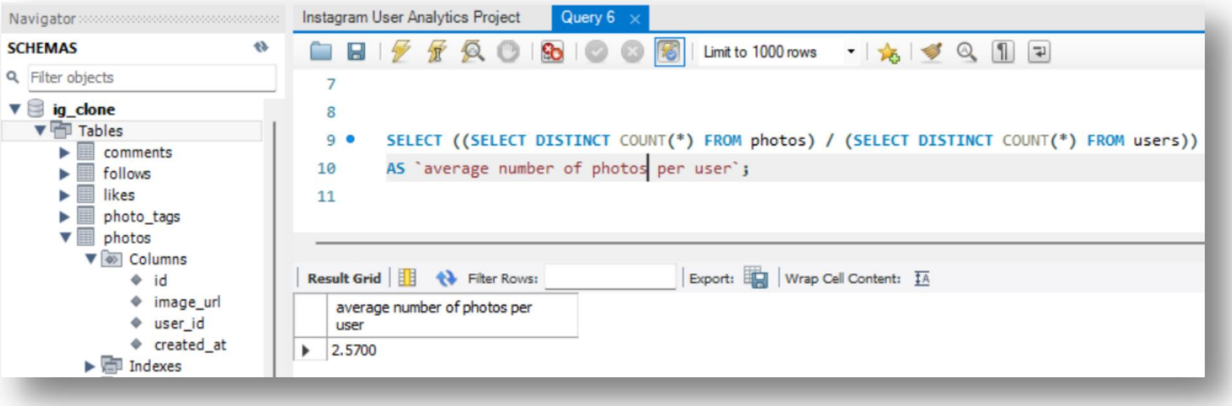
NOTE: Here, when calculating the average_number_of_posts_per_user, I have considered **photos**, **tags** and **photos** that have been tagged by any user as part of it as far as I understand their meanings. **These parameters can be subjective to the person asking the question as to what parameters they consider as part of this calculation.**



```
SELECT (((SELECT DISTINCT COUNT(*) FROM photos) +
            (SELECT DISTINCT COUNT(*) tags) +
            (SELECT DISTINCT COUNT(*) FROM photo_tags)) /
            (SELECT DISTINCT COUNT(*) FROM users))
AS `average number of posts per user`;
```

```
average number of posts per
user
7.5900
```

```
SELECT ((SELECT DISTINCT COUNT(*) FROM photos) / (SELECT DISTINCT COUNT(*) FROM users))
AS `average number of posts per user`;
```



```
SELECT ((SELECT DISTINCT COUNT(*) FROM photos) / (SELECT DISTINCT COUNT(*) FROM users))
AS `average number of photos per user`;
```

average number of photos per user
2.5700

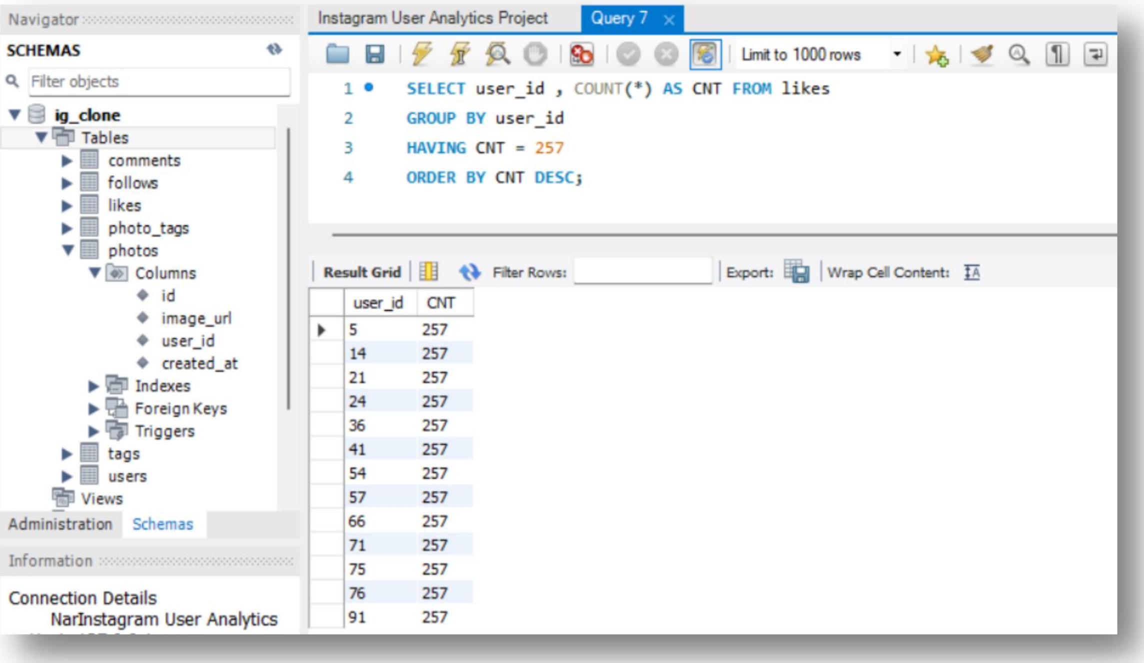
2. **Bots & Fake Accounts:** Investors want to know if the platform is crowded with fake and dummy accounts.
The following query was used:

SELECT user_id , COUNT(*) AS CNT FROM likes

GROUP BY user_id

HAVING CNT = 257

ORDER BY CNT DESC;



```
SELECT user_id , COUNT(*) AS CNT FROM likes
GROUP BY user_id
HAVING CNT = 257
ORDER BY CNT DESC;
```

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

The initial table obtained was good, however it suggested the need to filter all values with value 257. However, as soon as I executed the WHERE clause, the mistake was evident due to the order of execution, and then the usage of HAVING clause was evident.

Insights

Few insights that I feel to have developed are:

- Although the average number of posts per user is defined here, it does not represent the true state of things as they are, since there can be multiple users who have actually no posts on their account. On the other hand, there can be some users that actually have a huge number of posts on their account, which can significantly vary the MEAN. So, MEAN should NEVER be considered as any sort of standard basis for companies to determine how well users are engaged with the app.
- Having fake account and bots can be considered as a waste of money of the investors since these add no actual value to the app. However, their detection should be based on more parameters, such as online activity time, actual time spent viewing individual posts before they like them etc.
- Holding contests or competitions can have a very positive impact on the user-base of the app and can in turn encourage investors to add more capital to the value of the app.
- Targeting certain topics and channels based on the hashtags is indeed very smart since it will be viewed by a huge traffic of users. This feature can also be used to improve the recommendations sections of the app or the window below the search bar of the app where various posts and reels are hosted.
- The Ad campaign is indeed very good solution to increase the user interaction base of the app. However special care must be taken that the ads should not be repetitive nor should they often intervene the normal usage of the app by the user. Since Instagram is a free app on the Internet, people usually don’t expect ads to be repetitive and distracting. If this is not taken care of, the app may well loose its user-base by a certain amount, given the already good competition in the market by other such apps such as Telegram, Discord, Threads etc. Same care should be taken when sending promotional mails to users have not done any activity on the app, since unsubscribing from such promotional mails are just 2-3 clicks away, which will result in the probable complete loss of these users.

Result

The above insights derived from my personal analysis of the entire project did develop a better understanding of how things work on a bigger level, albeit there are various elements to all the insights provided above.

The project led me to a good and easy Hands-On on MySQL and also led me to discover a number of things in MySQL on my own.