MAYANK KUMAR GANDHARV

TRAINITY PROJECT 2

Instagram User Analytics

Project Description:

The project INSTAGRAM USER ANALYTICS is a sql based project. The sql tasks given in the description is basically used to do market analysis and investor engagement for the platform.

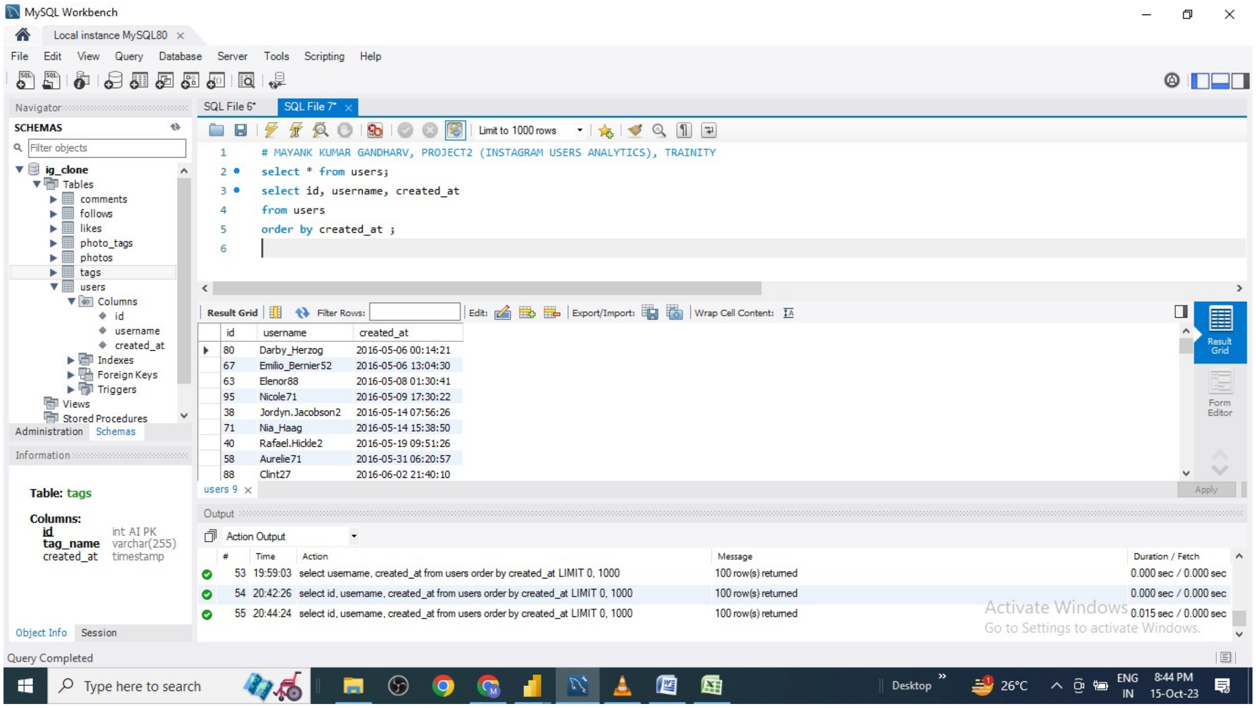
In order to do all the tasks MySQL workbench is used to sort data and derive insights from the database given. My approach towards this project is to derive as much information (insights) as I can for the betterment of the business and the dedicated teams involved in the business (eg. Campaign team, Marketing team, etc) and to do so I will use the concepts that I learned from the MySQL videos offered by Trainity. Detailed approach is given with each SQL tasks performed.

Approach:

A) **Marketing Analysis:**

1. **Loyal User Reward:**
   My Task: Identify the five oldest users on Instagram from the provided database
   Solution: In order to identify 5 oldest users I sorted the "Users" Table given in the database in Ascending Order.

id	username
80	**Darby_Herzog**
67	**Emilio_Bernier52**
63	**Elenor88**
95	**Nicole71**
38	**Jordyn.Jacobson2**



```
# MAYANK KUMAR GANDHARV, PROJECT2 (INSTAGRAM USERS ANALYTICS), TRAINITY
select * from users;
select id, username, created_at
from users
order by created_at ;
```

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.Hickle2	2016-05-19 09:51:26
58	Aurelie71	2016-05-31 06:20:57
88	Clint27	2016-06-02 21:40:10

Table: tags

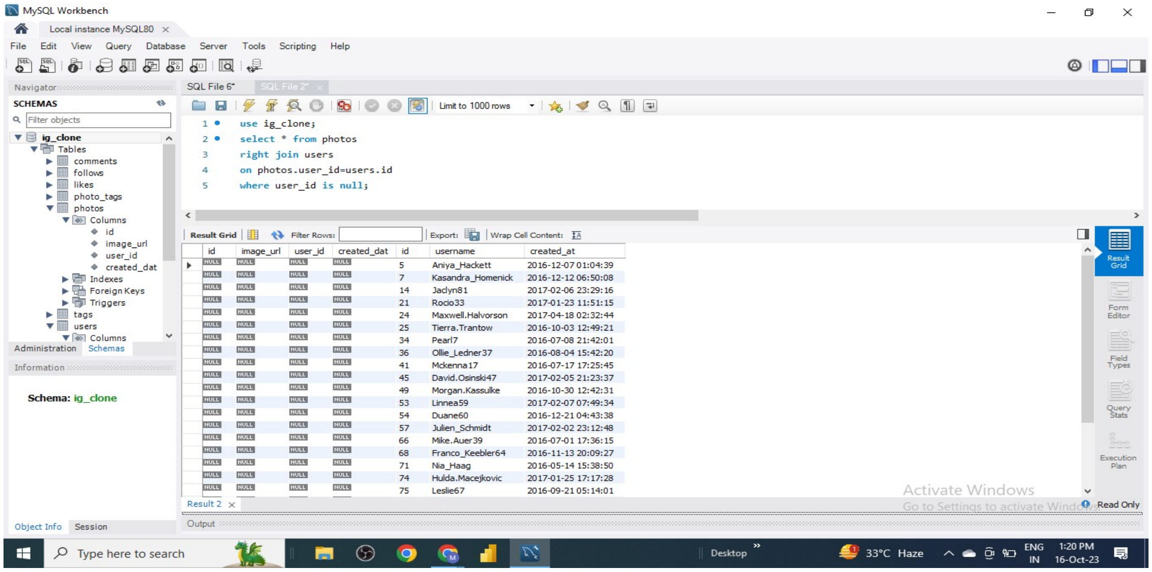
Columns:
id int AI PK
tag_name varchar(255)
created_at timestamp

```
#   Time      Action                                                                   Message              Duration / Fetch
53  19:59:03  select username, created_at from users order by created_at LIMIT 0, 1000      100 row(s) returned  0.000 sec / 0.000 sec
54  20:42:26  select id, username, created_at from users order by created_at LIMIT 0, 1000  100 row(s) returned  0.000 sec / 0.000 sec
55  20:44:24  select id, username, created_at from users order by created_at LIMIT 0, 1000  100 row(s) returned  0.015 sec / 0.000 sec
```

Query Completed

2. Inactive User Engagement:

For this SQL Task I first joined the two table i.e USERS and PHOTOS because I want data from total users that haven’t uploaded a single photo (inactive) and compare it with users that have posted photos by right join and got information of inactive users by the value NULL in front of their image_url that means these are the inactive users and lastly I put a “where” condition for sorting NULL values (got information for every inactive users).



```
use ig_clone;
select * from photos
right join users
on photos.user_id=users.id
where user_id is null;
```

id	image_url	user_id	created_dat	id	username	created_at
NULL	NULL	NULL	NULL	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

Schema: ig_clone

id	username
5	Aniya_Hackett
7	Kasandra_Homenick
14	Jaclyn81
21	Rocio33
24	Maxwell.Halvorson
25	Tierra.Trantow
34	Pearl7
36	Ollie_Ledner37
41	Mckenna17
45	David.Osinski47
49	Morgan.Kassulke
53	Linnea59
54	Duane60
57	Julien_Schmidt
66	Mike.Auer39
68	Franco_Keebler64
71	Nia_Haag
74	Hulda.Macejkovic
75	Leslie67
76	Janelle.Nikolaus81
80	Darby_Herzog
81	Esther.Zulauf61
83	Bartholome.Bernhard
89	Jessyca_West
90	Esmeralda.Mraz57
91	Bethany20

3. **Contest Winner Declaration:**
   For this SQL task first I saw the content from photos and likes table. Then to understand the query properly I observed that we have to find the most liked photo. In order to find the most liked photo I used group by and aggregations. I aggregated using “COUNT” for user_id from likes table and then grouped by photo_id. Like that I got each count for user_id for each photo_id (total photo_id = 257). The top 3 result is shown below with a tie for 2nd place between photo_id 127 & 182

photo_id	likes	user id	username
145	48	52	Zack_Kemmer93
127	43	46	Malinda_Streich
182	43	65	Adelle96
123	42	44	Seth46

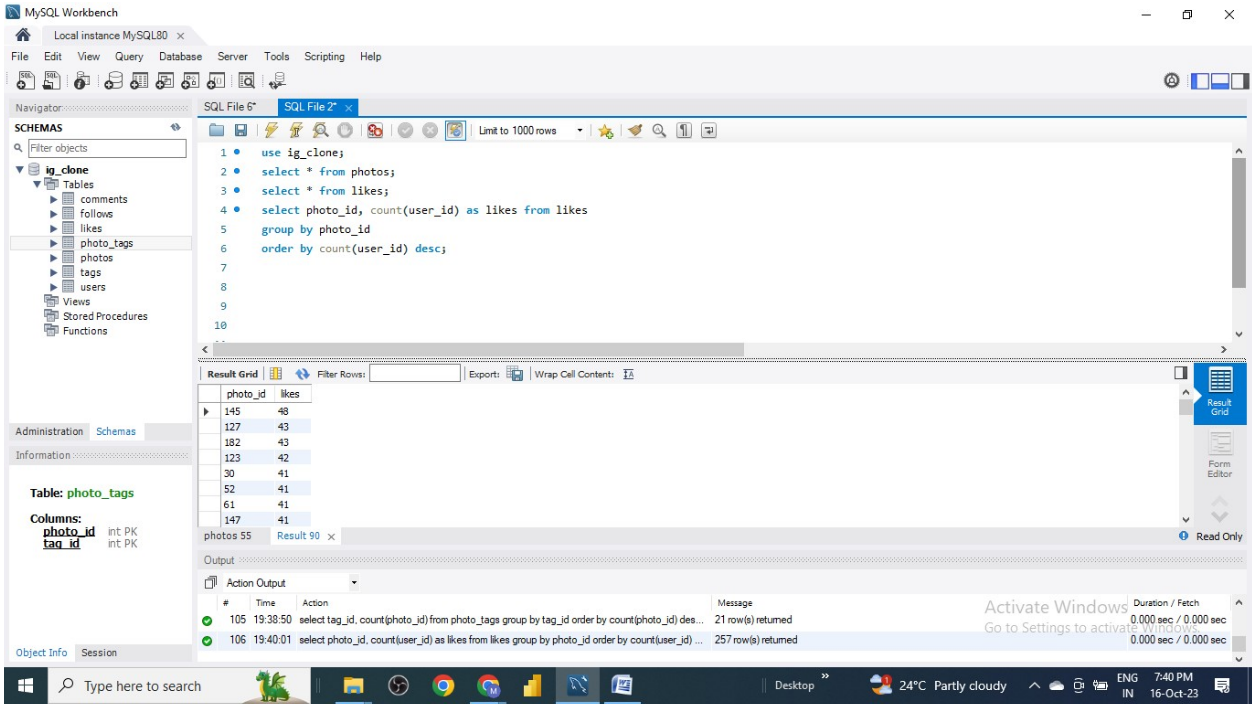


Table: photo_tags

Columns:
photo_id int PK
tag_id int PK

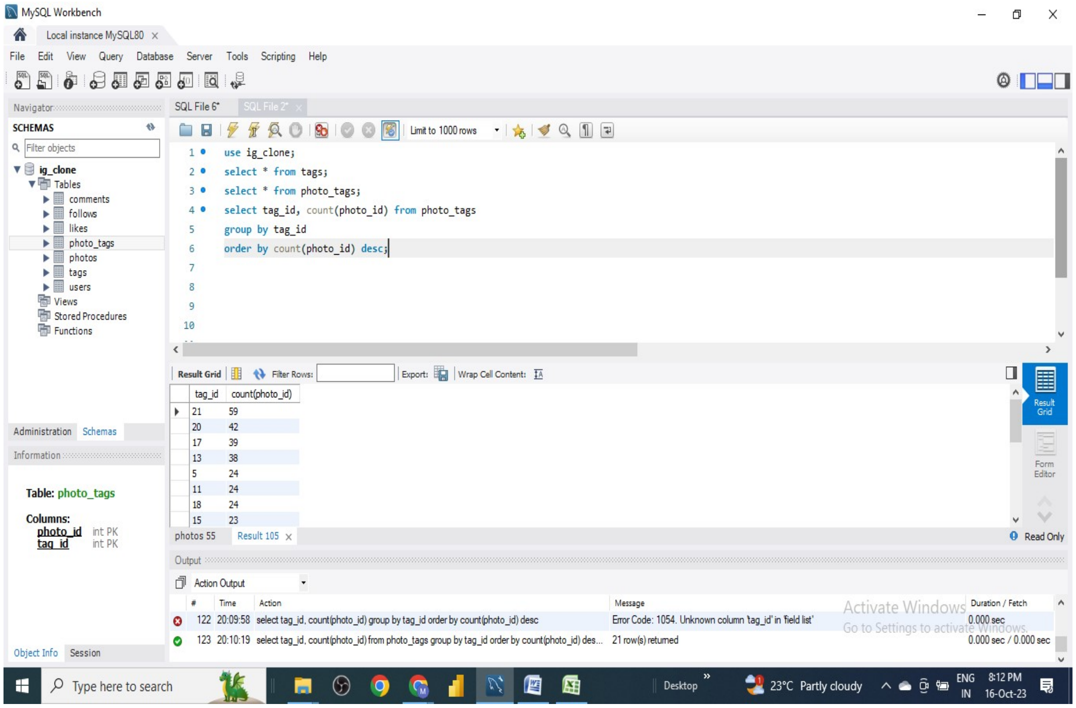
```
use ig_clone;
select * from photos;
select * from likes;
select photo_id, count(user_id) as likes from likes
group by photo_id
order by count(user_id) desc;
```

photo_id	likes
145	48
127	43
182	43
123	42
30	41
52	41
61	41
147	41

```
#    Time      Action                                                                                 Message             Duration / Fetch
105  19:38:50  select tag_id, count(photo_id) from photo_tags group by tag_id order by count(photo_id) des...  21 row(s) returned   0.000 sec / 0.000 sec
106  19:40:01  select photo_id, count(user_id) as likes from likes group by photo_id order by count(user_id) ...  257 row(s) returned  0.000 sec / 0.000 sec
```

4. **Hashtag Research:**
   For identifying top 5 most hashtags used, first I grouped tag_id corresponding to the count of photo_id. And then after executing it got the most tags used and then ordered by desc order. Lastly I filtered the tags table as per the requirement for the tag id.

tag_id	count(photo_id)	tag_name
21	59	smile
20	42	beach
17	39	party
13	38	fun
5	24	food



```
use ig_clone;
select * from tags;
select * from photo_tags;
select tag_id, count(photo_id) from photo_tags
group by tag_id
order by count(photo_id) desc;
```

tag_id	count(photo_id)
21	59
20	42
17	39
13	38
5	24
11	24
18	24
15	23

Table: photo_tags

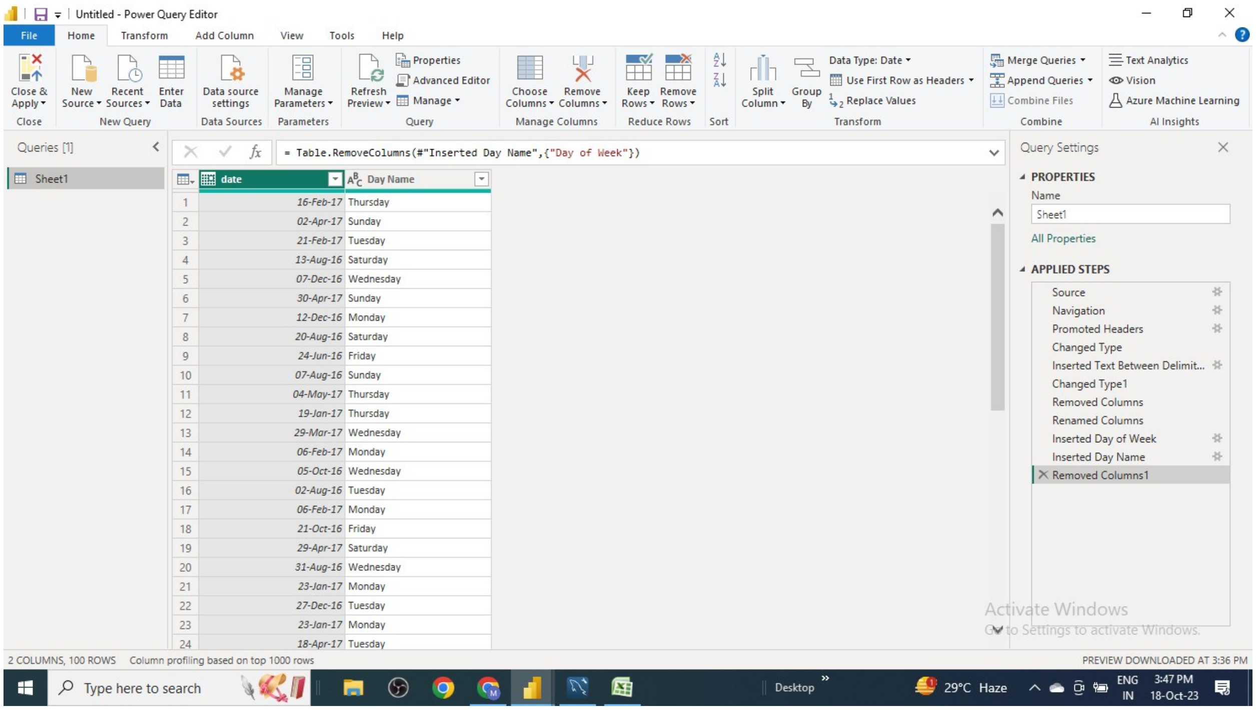
Columns:
photo_id int PK
tag_id int PK

#	Time	Action	Message	Duration / Fetch
122	20:09:58	select tag_id, count(photo_id) group by tag_id order by count(photo_id) desc	Error Code: 1054. Unknown column 'tag_id' in 'field list'	0.000 sec
123	20:10:19	select tag_id, count(photo_id) from photo_tags group by tag_id order by count(photo_id) des...	21 row(s) returned	0.000 sec / 0.000 sec

5. **Ad Campaign Launch:**

For this Question I was stuck and was not able to understand how to execute this query in My SQL. Instead I used PowerBI for this particular query, as I need to extract the day of the week from the given database, I was not able to write any SQL syntax. My approach towards this solution is purely based on Power BI. Steps that I followed are:

1. Took information of the database table “users” and copied the ‘Created_at’ row then pasted in Excel sheet.
2. After that I loaded the data from the excel sheet using ETL in power Query.
3. Sorted the data using add column by example and added a column where there were dated included.
4. After wards I was successfully able to extract the day of the week from the date column and,
5. Lastly I gave a count of each day in PowerBI.



```
= Table.RemoveColumns(#"Inserted Day Name",{"Day of Week"})
```

	date	Day Name
1	16-Feb-17	Thursday
2	02-Apr-17	Sunday
3	21-Feb-17	Tuesday
4	13-Aug-16	Saturday
5	07-Dec-16	Wednesday
6	30-Apr-17	Sunday
7	12-Dec-16	Monday
8	20-Aug-16	Saturday
9	24-Jun-16	Friday
10	07-Aug-16	Sunday
11	04-May-17	Thursday
12	19-Jan-17	Thursday
13	29-Mar-17	Wednesday
14	06-Feb-17	Monday
15	05-Oct-16	Wednesday
16	02-Aug-16	Tuesday
17	06-Feb-17	Monday
18	21-Oct-16	Friday
19	29-Apr-17	Saturday
20	31-Aug-16	Wednesday
21	23-Jan-17	Monday
22	27-Dec-16	Tuesday
23	23-Jan-17	Monday
24	18-Apr-17	Tuesday

PROPERTIES

Name: Sheet1

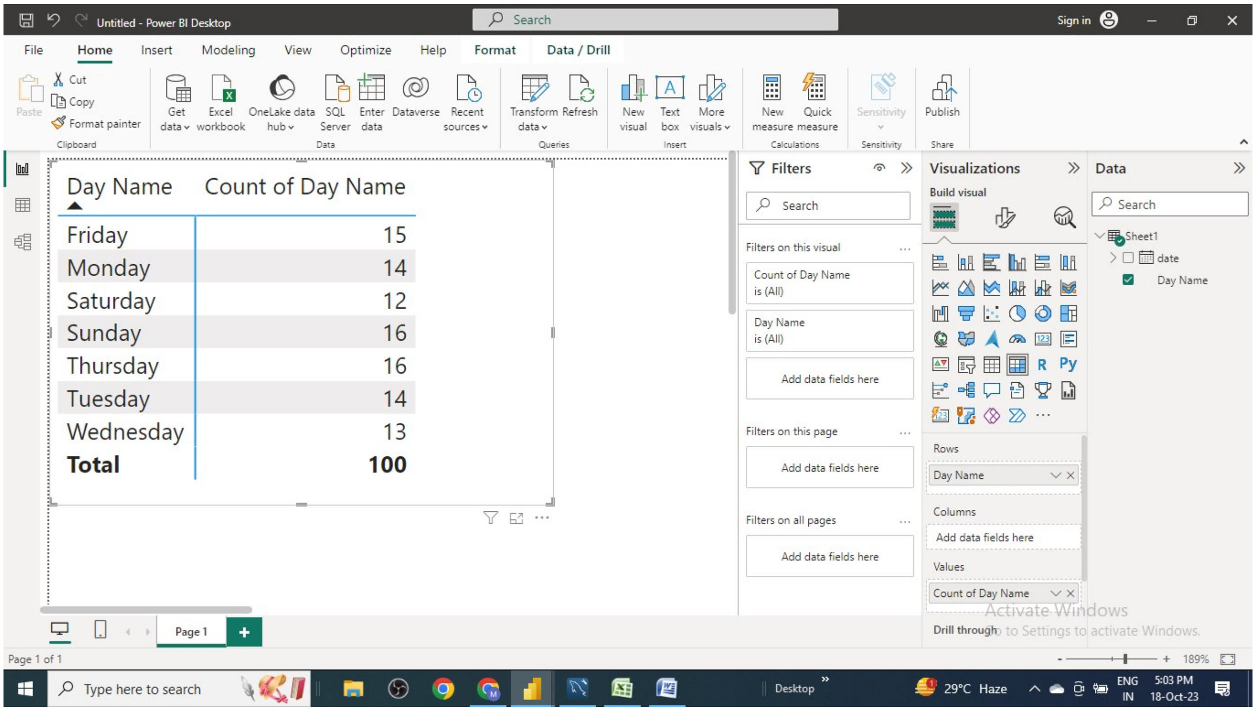
All Properties

APPLIED STEPS

- Source
- Navigation
- Promoted Headers
- Changed Type
- Inserted Text Between Delimit...
- Changed Type1
- Removed Columns
- Renamed Columns
- Inserted Day of Week
- Inserted Day Name
- Removed Columns1

2 COLUMNS, 100 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 3:36 PM



Day Name	Count of Day Name
Friday	15
Monday	14
Saturday	12
Sunday	16
Thursday	16
Tuesday	14
Wednesday	13
Total	**100**

Filters on this visual: Count of Day Name is (All); Day Name is (All)

Data: Sheet1 — date; ☑ Day Name

Rows: Day Name

Columns: Add data fields here

Values: Count of Day Name

Day Name	Count of Day Name
Sunday	16
Thursday	16
Friday	15
Monday	14
Tuesday	14
Wednesday	13
Saturday	12

Most users register on SUNDAY and THURSDAY. According to me an ad campaign should be held on Thursday and Sunday.

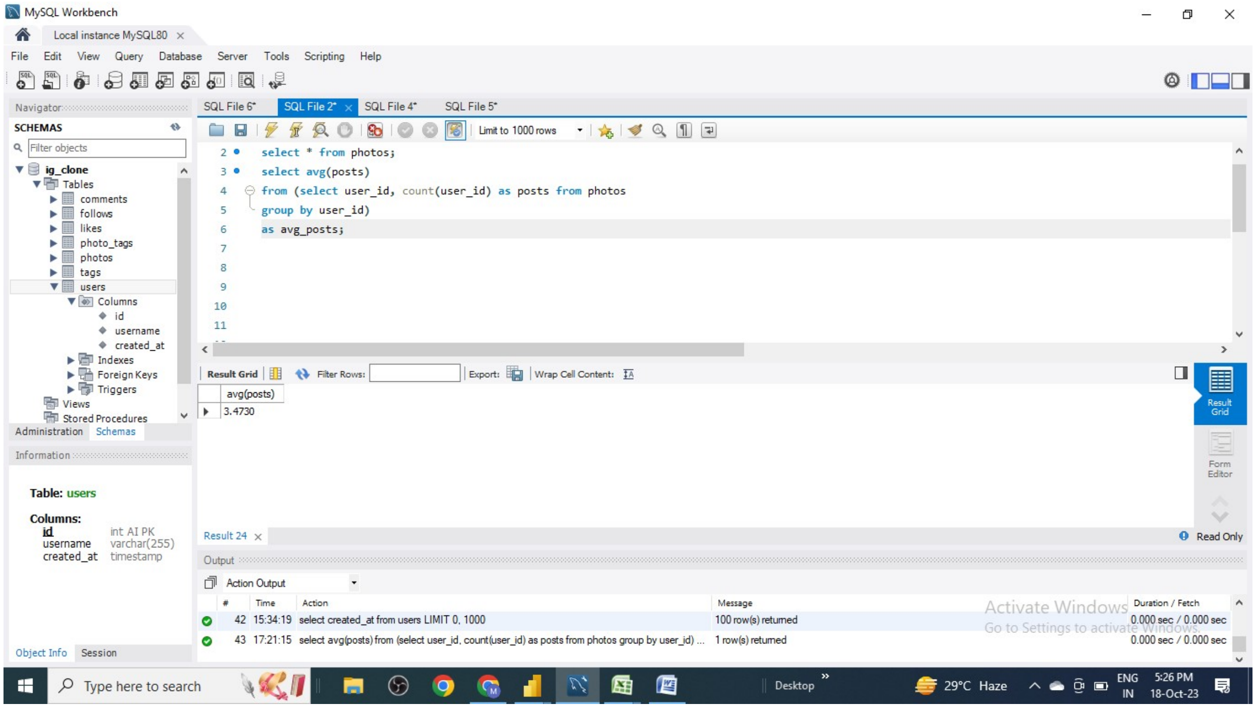
B) Investor Metrics:

1. User Engagement:

To calculate average number of posts per user on instagram I used Subquery for it.

Steps:

1. Select user_id and count of user_id from photos.
2. Count of user_id will give me the number of posts posted by every user.
3. Group it according to user_id (for every user_id it will group by the count of user_id following.)
4. Then I created an outer table so that I could calculate the average of the previous grouping.
5. At last I was able to calculate the average of the count of user_id i.e **3.4730.**



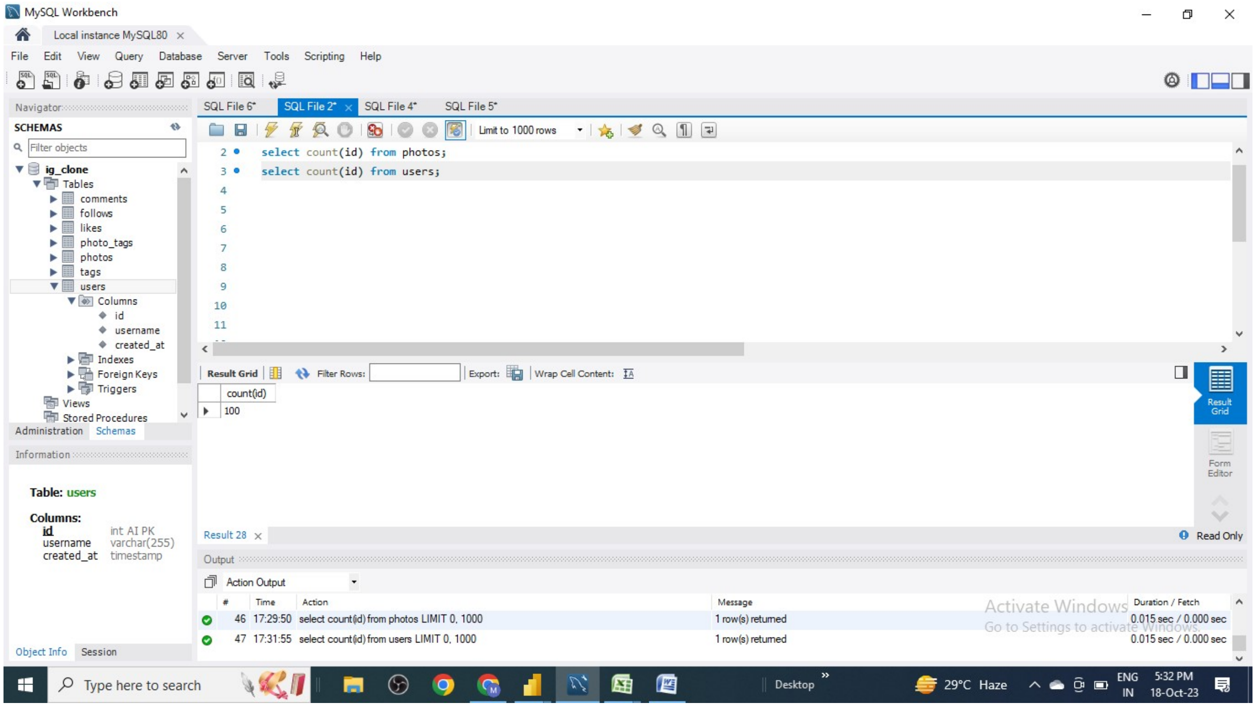
```
select * from photos;
select avg(posts)
from (select user_id, count(user_id) as posts from photos
group by user_id)
as avg_posts;
```

avg(posts)
3.4730

```
#   Time      Action                                                                          Message              Duration / Fetch
42  15:34:19  select created_at from users LIMIT 0, 1000                                      100 row(s) returned  0.000 sec / 0.000 sec
43  17:21:15  select avg(posts) from (select user_id, count(user_id) as posts from photos group by user_id) ...  1 row(s) returned  0.000 sec / 0.000 sec
```

Now, to calculate the total number of photos on Instagram divided by the total number of users,

1. First I gave command of “select count(id) from photos”
2. Then “select count(id) from users”
3. Got answer 257 & 100 respectively.
4. The final is 2.57 (257/100)



```
select count(id) from photos;
select count(id) from users;
```

```
count(id)
100
```

Table: users

Columns:
id int AI PK
username varchar(255)
created_at timestamp

```
#   Time      Action                                       Message            Duration / Fetch
46  17:29:50  select count(id) from photos LIMIT 0, 1000   1 row(s) returned  0.015 sec / 0.000 sec
47  17:31:55  select count(id) from users LIMIT 0, 1000    1 row(s) returned  0.015 sec / 0.000 sec
```

2. Bots and Fake Accounts:

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

   Solution: for this task I used subqueries and then inner joined

   a. First I selected user_id , count of photo_id from likes, then grouped according to user_id in descending order. For every user_id there is a different user name
   b. Then I subqueried the upper query and made a temporary table named fake_users.
   c. Lastly I inner joined id from users and user_id from fake_users so that I could even get the names of the users along with the id.
   d. After filtering the data from 257 I got all the bots.

username	user_id	likes
Aniya_Hackett	5	257
Jaclyn81	14	257
Rocio33	21	257
Maxwell.Halvorson	24	257
Ollie_Ledner37	36	257

Mckenna17	41	257
Duane60	54	257
Julien_Schmidt	57	257
Mike.Auer39	66	257
Nia_Haag	71	257
Leslie67	75	257
Janelle.Nikolaus81	76	257
Bethany20	91	257

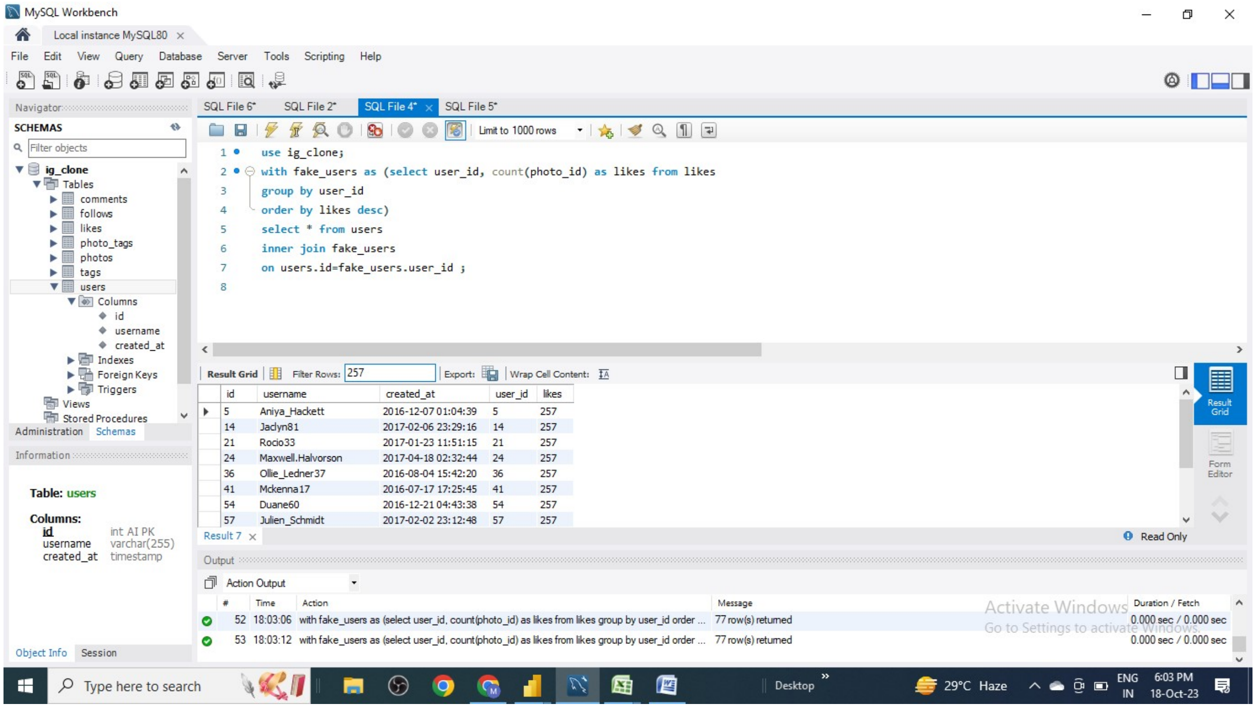


Table: users

Columns:
id int AI PK
username varchar(255)
created_at timestamp

```
use ig_clone;
with fake_users as (select user_id, count(photo_id) as likes from likes
group by user_id
order by likes desc)
select * from users
inner join fake_users
on users.id=fake_users.user_id ;
```

Filter Rows: 257

id	username	created_at	user_id	likes
5	Aniya_Hackett	2016-12-07 01:04:39	5	257
14	Jaclyn81	2017-02-06 23:29:16	14	257
21	Rocio33	2017-01-23 11:51:15	21	257
24	Maxwell.Halvorson	2017-04-18 02:32:44	24	257
36	Ollie_Ledner37	2016-08-04 15:42:20	36	257
41	Mckenna17	2016-07-17 17:25:45	41	257
54	Duane60	2016-12-21 04:43:38	54	257
57	Julien_Schmidt	2017-02-02 23:12:48	57	257

```
#   Time      Action                                                                                  Message            Duration / Fetch
52  18:03:06  with fake_users as (select user_id, count(photo_id) as likes from likes group by user_id order ...  77 row(s) returned  0.000 sec / 0.000 sec
53  18:03:12  with fake_users as (select user_id, count(photo_id) as likes from likes group by user_id order ...  77 row(s) returned  0.000 sec / 0.000 sec
```

TECH STACK USED: I used MySQL Workbench 8.0.34 for this project.

INSIGHTS: the given database for instagram users and their activities helped to find out different aspects for the business such as oldest users, most active, least active users, fake users, most users created on selected day of the week i.e **Thursdays and Sundays**. All these insights were helpful for different departments such as marketing, ad campaign etc.

RESULT: The achievements that I accomplished are:

1. Hands on Experience on sql
2. Practical use of grouping, ordering and joining tables.

3. Subqueries application
4. Deriving insights

Thoughts on insights:

Instagram is a great platform for many users to show there social life. Different ad campaigns and competitions can be made for top active users in order to get more traffic on the site. More traffic will encourage nore investors and hence the business from investers or sponsors can be done on a wide scale, Using user engagement data.

MY THOUGHTS ON THE PROJECT:

This project helped me to understand SQL better. Not only theoretically but practically as well. Now I can play along with database and get required insights using different queries syntax. Overall the project was very exciting to do.