

# INSTAGRAM USER ANALYTICS

(SQL FUNDAMENTALS)

## PROJECT DESCRIPTION

In this project I'll do a work of data analyst. We'll analyze with user interactions and engagement with the Instagram app to provide valuable insights that can help business grow.

Also we'll track how users engage with a digital product such as a software or a mobile application or a software application.

Those things will help in many ways i.e the marketing team might use these insights to launch a new campaign, the product team might use them to decide on new features to build, and the development team might use them to improve the overall experience.

In this project I'll be using MySQL to analyze **Instagram data** and my insights will help the product manager and rest of the team to make decisions about the future features on the Instagram app.

## **APPROACH**

Firstly, I'll take approach to upload the dataset in MySQL then create and insert the values in the database by using the DDL (Data Definition Language), DML (Data Manipulation Language) by using MySQL workbench.

## **TECH – STACK USED**

The MySQL Workbench Community and MySQL and MySQL community server and MySQL Workbench release for versions 8.0 through 8.0.34.

# INSIGHTS

## A . Marketing Analysis :

The marketing team might use these insights to launch a new campaign.

### 1. Identify the five oldest users on Instagram from the provided Database.

In this question we have to analyze the five oldest users on Instagram from the database because the marketing team wants to reward the most loyal users i.e those who have been using the platform for the longest time.

So firstly, we select the **id , username, created\_at** from users table as this will help to extract the required output from the database after this we'll arrange the data in descending order by using **order by desc** approach and we'll set the **limit 5** because we need only the top 5 oldest Instagram accounts from database.

Here's our SQL output :

The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying the following SQL query:

```
1 # 1. Identify the five oldest users on Instagram from the provided database.
2
3 select id, username , created_at from users order by created_at asc limit 5 ;
```

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

#	id	username	created_at
1	80	Darby_Herring	2016-05-06 00:14:21
2	67	Emily_Bernier52	2016-05-06 13:04:30
3	63	Elenor88	2016-05-08 01:30:41
4	95	Nicole71	2016-05-09 17:30:22
5	38	Jordyn.Jacobson2	2016-05-14 07:56:26

The 'Output' tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	12:09:17	select dayofweek(created_at), count(dayofweek(created_at)) from users group by dayofweek(created_at) order by count(dayofweek(created_at)) desc limit 5	2 row(s) returned	0.000 sec / 0.000 sec
2	12:13:48	select * from users	100 row(s) returned	0.000 sec / 0.000 sec
3	12:14:58	select id, username , created_at from users order by created_at asc limit 5	5 row(s) returned	0.000 sec / 0.000 sec
4	12:15:37	select id, username , created_at from users order by created_at desc limit 5	5 row(s) returned	0.000 sec / 0.000 sec
5	12:15:43	select id, username , created_at from users order by created_at asc limit 5	5 row(s) returned	0.000 sec / 0.000 sec

## 2. Identify the users who have never posted a single photo on Instagram.

In this question, we want those users who have never posted a single photo on Instagram so the team wants to encourage inactive users to start posting by sending them promotional emails. So firstly we **select distinct ids from table Photos** which we'll use to compare id's from **users** table.

Now, we **select id, username from users** where id not in the selected ids from Photos table.

We did this because there won't be any photos posted by a person whose id's not active.

Here's our output :

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

```
select id ,username from users where id not in (select distinct user_id from Photos);
```

The Results grid displays the output of the query, showing a list of users with their IDs and usernames. The output is as follows:

id	username
5	Aniya_Hackett
7	Alexandra_Homenick
14	Jacynb81
21	Rocio33
24	Maxwell_Halvorson
25	Terra_Tranlow
34	Heath7
36	Olivia_Ledner37
41	McKenzie17
45	David_Osinski47
49	Morgan_Kassulke
53	Linnea59
54	Duane60
57	Julien_Schmidt
66	Mila_Auer39
68	Franco_Keebler64
71	Nia_Thao
74	Hilda_Macejkovic
75	Leda67
76	Janelle_Nikolaus81
80	Darby_Herzog
81	Esther_Zulauf61
83	Bertholome_Bernhard
89	Jessyca_West
90	Emeralda_Nitz57

The Output tab at the bottom shows the execution log, including the query execution time and the number of rows returned.

The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying the following SQL query:

```

1  # 2. Identify users who have never posted a single photo on Instagram.
2
3  select id ,username from users where id not in (select distinct user_id from Photos) ;

```

The 'Result Grid' shows the output of the query, listing 25 users who have never posted a photo. The columns are 'id' and 'username'.

id	username
21	Rodo33
24	Maxwell.Halvorson
25	Tierra.Trantow
34	Pearl7
36	Ollie_Leshner37
41	McKenzie17
45	David.Oanski47
49	Morgan.Kansulke
53	Linnea59
54	Duane60
57	Julien_Schmidt
66	Mike.Auer39
68	Franco_Keebler64
71	Na_Haag
74	Hilda.Macejovic
75	Leslie67
76	Janelle.Nikolaus81
80	Darby_Herzog
81	Erther.Zulauf61
83	Bartholome.Bernhard
89	Jessyca_West
90	Esmeralda.Mraz57
91	Bethany20

The 'Output' tab shows the execution details of the query, including the time taken and the number of rows returned for each step.

#	Time	Action	Message	Duration / Fetch
1	12:09:17	select dayofweek(created_at), count(dayofweek(created_at)) from users group by dayofweek(created_at) order by count(dayofweek(created_at)) desc limit 1	2 row(s) returned	0.000 sec / 0.000 sec
2	12:13:48	select * from users	100 row(s) returned	0.000 sec / 0.000 sec
3	12:14:58	select id, username, created_at from users order by created_at asc limit 5	5 row(s) returned	0.000 sec / 0.000 sec
4	12:15:37	select id, username, created_at from users order by created_at desc limit 5	5 row(s) returned	0.000 sec / 0.000 sec
5	12:15:43	select id, username, created_at from users order by created_at asc limit 5	5 row(s) returned	0.000 sec / 0.000 sec
6	12:27:17	select id ,username from users where id not in (select distinct user_id from Photos)	25 row(s) returned	0.000 sec / 0.000 sec

### 3. Determine the winner of the contest and provide their details to the team.

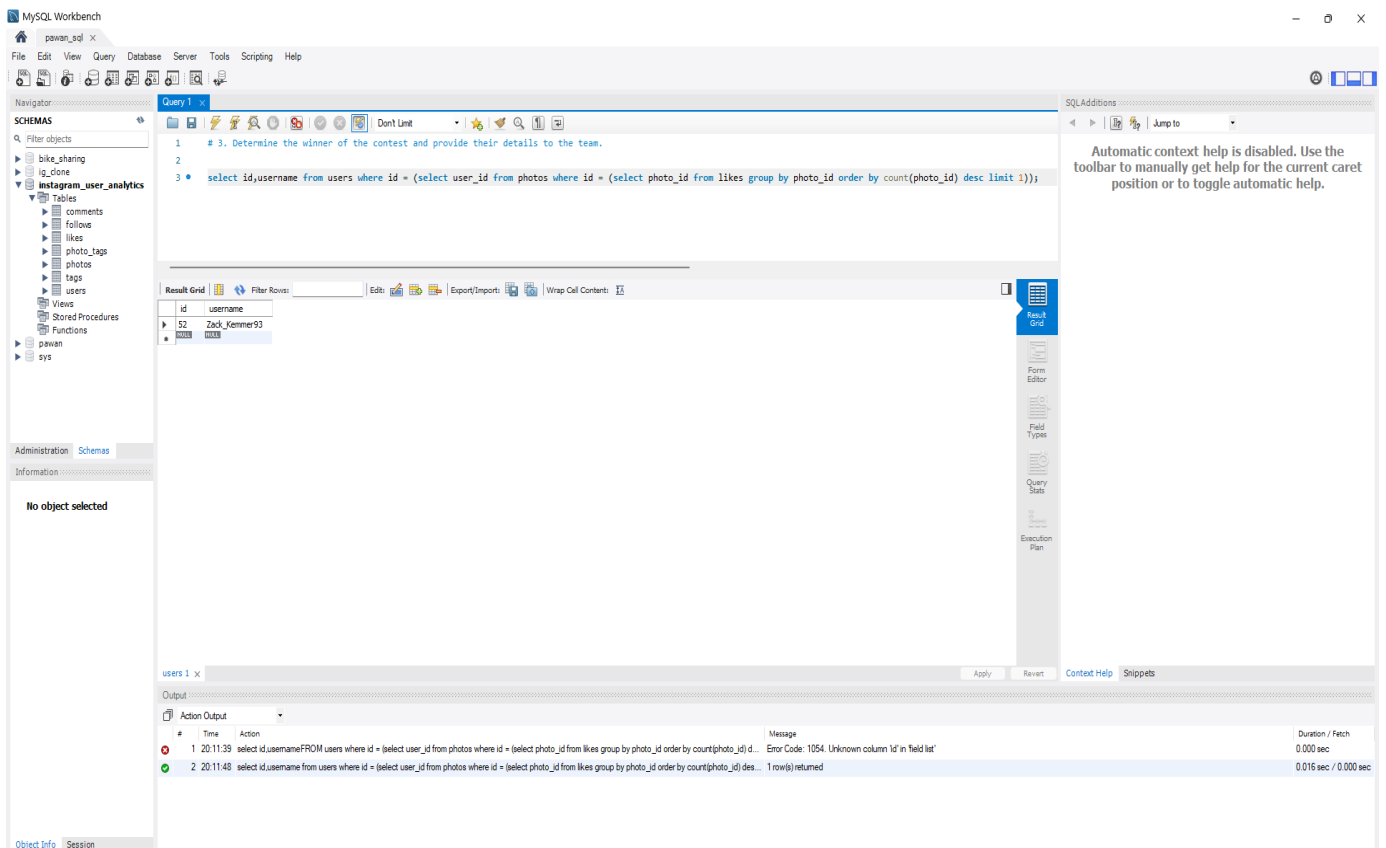
In this question, we have to analyze the winner of the contest who get the most likes in the single photo because the team has organized a contest where the user with the most likes on a single photo wins.

So firstly, we select **the photo\_id from likes** and **group by photo\_id** then **order by count of photo\_id** and arranging them in descending order by using **desc** syntax and we set the **limit to 1**.

We're taking this approach to get output of the user photo which gets most likes in a single photo.

Now we **select id, username from users** and use where function with **id** and we'll get the required output.

Here's our output :



#### 4. Identify and suggest the top five most commonly used hashtags on the platform.

In this question, we analyze the top five most commonly used hashtags because the partner brand wants to know the most popular hashtags to use in their posts to reach the most people.

For this we will be using **inner join** to join the tables **photo\_tags** and **tags** on the condition of equitable **ids** also grouping by **tag\_id** and ordering by **tag\_id**. Now, by arranging the output generated in descending order by using **desc** and limiting the record count to 5, we get the desired output.

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

```

1 #4. Identify and suggest the top five most commonly used hashtags on the platform.
2
3 select b.id , b.tag_name , count(a.tag_id) from photo_tags as a inner join Tags as b
4 on a.tag_id = b.id group by a.tag_id order by count(a.tag_id) desc limit 5
5

```

The Results grid shows the output of the query:

id	tag_name	count(a.tag_id)
21	smile	59
20	beach	42
17	party	39
13	fun	38
18	concert	24

The Output tab shows the execution log:

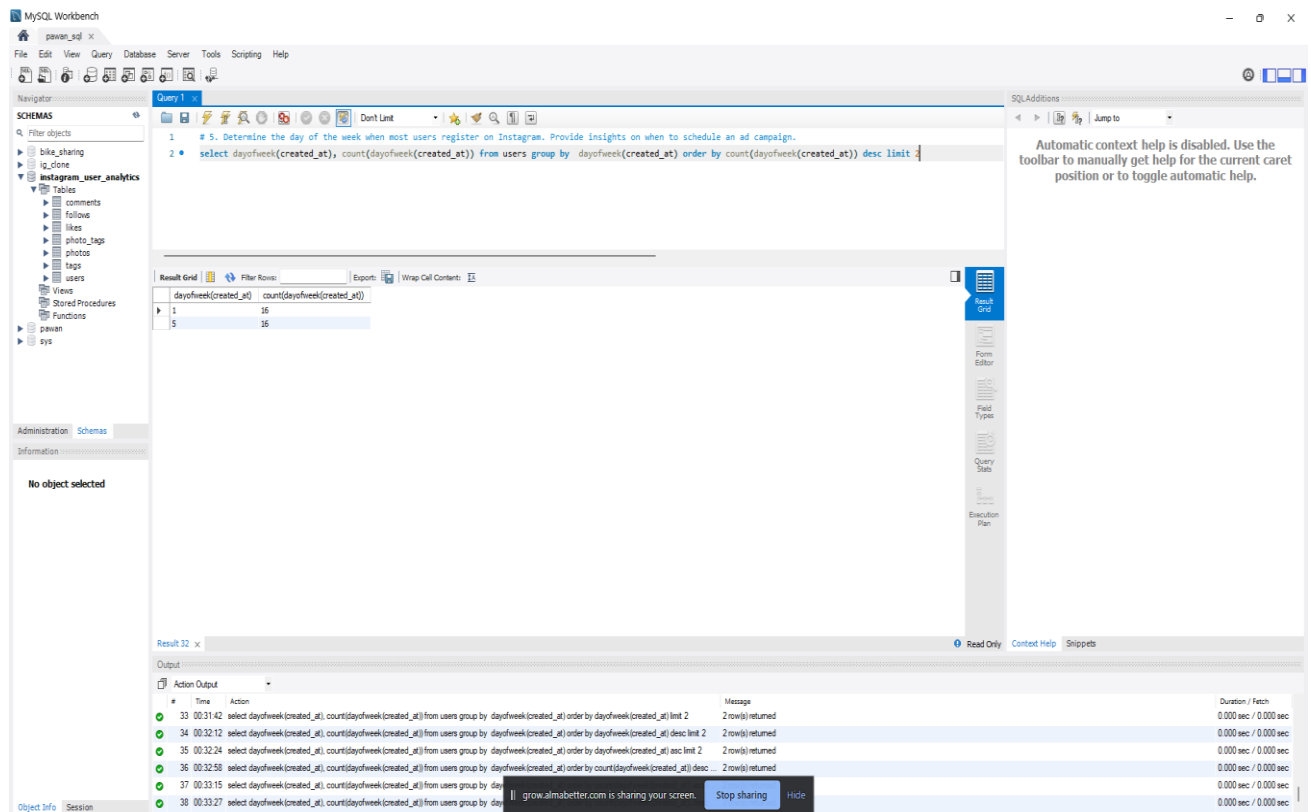
#	Time	Action	Message	Duration / Fetch
14	23:58:59	select * from photo_tags	501 row(s) returned	0.000 sec / 0.000 sec
15	00:00:59	select tag_id , count(tag_id) from photo_tags group by tag_id	21 row(s) returned	0.016 sec / 0.000 sec
16	00:01:43	select tag_id , count(tag_id) from photo_tags group by tag_id order by count(tag_id) desc	21 row(s) returned	0.000 sec / 0.000 sec
17	00:06:01	select b.id , b.tag_name , count(a.tag_id) from photo_tags as a inner join Tags as b on a.tag_id = b.id group by a.tag_id order by count(a.tag_id) desc	21 row(s) returned	0.000 sec / 0.000 sec
18	00:06:28	select b.id , b.tag_name , count(a.tag_id) from photo_tags as a inner join Tags as b on	21 row(s) returned	0.000 sec / 0.000 sec

## 5. Determine the day of the week when the most users register on Instagram. Provide insights on when to schedule on ad campaign.

In this question, we analyze the day when the most users register on Instagram because the team want to know the best day of the week to launch ads.

Firstly, we **select** the **dayofweek of created\_at** and **count** this insights from **users** and after using this we'll apply **group by** function with **dayofweek of created\_at** and after this we arrange in descending order by using **desc** and set **limit 2**.

Here's our output :



## B. Investor Metrics :

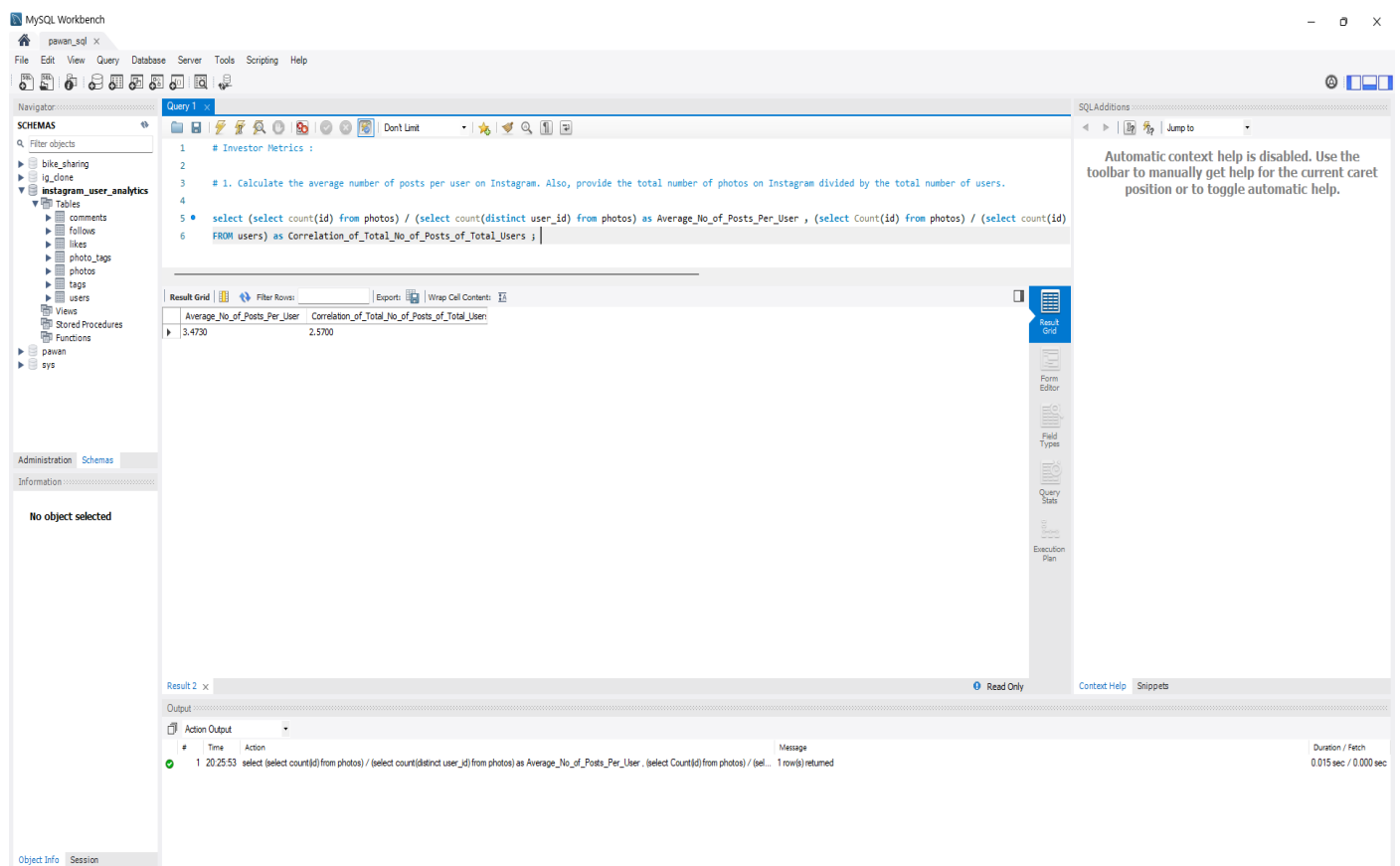
1. Calculate the average number of posts per user on Instagram. Also, provide the total number of photos on Instagram divided by the total numbers of users.

In this question we have to analyze the average number of posts per users on Instagram and the correlation between the total numbers of photos and the total numbers of users because Investor wants to know if users are still active and posting on Instagram or if they are making fewer posts.

Firstly, **select** the count of **id** from **photos** and divide by the count of **user\_id** from **photos** to get **average number of posts of users** and after this we select the count of **id** from **photos** and divide by count of **id** from **users** so after doing this we get the **correlation between of total number of posts of total users**.



Here's our output :



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of the database structure, including tables like 'comments', 'follows', 'likes', 'photo\_tags', 'photos', 'tags', 'users', 'views', 'Stored Procedures', 'Functions', 'pawan', and 'sys'. The main query editor contains the following SQL code:

```
1 # Investor Metrics :
2
3 # 1. Calculate the average number of posts per user on Instagram. Also, provide the total number of photos on Instagram divided by the total number of users.
4
5 select (select count(id) from photos) / (select count(distinct user_id) from photos) as Average_No_of_Posts_Per_User , (select Count(id) from photos) / (select count(id)
6 FROM users) as Correlation_of_Total_No_of_Posts_of_Total_Users ;
```

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

Average_No_of_Posts_Per_User	Correlation_of_Total_No_of_Posts_of_Total_Users
3.4730	2.5700

The bottom panel shows the 'Output' tab with the following message:

```
1 20:25:53 select (select count(id) from photos) / (select count(distinct user_id) from photos) as Average_No_of_Posts_Per_User , (select Count(id) from photos) / (sel... 1 row(s) returned
```

The 'Duration / Fetch' column shows '0.015 sec / 0.000 sec'.

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

In this question, we analyze the users who have liked the every single photo which is not typically possible for a normal user because the Investors want to know if the platform is crowded with fake and dummy accounts.

Firstly, we need to figure out the **ids** for which the **count of ids** from **likes** and **Photos** table are equal and also for doing this we **group by user id**. Finally we select **id** and **username** from **users** table to get the required details.

Here's our output :

The screenshot displays the MySQL Workbench interface. The 'Query' window shows the following SQL query:

```
1 # Investor Metrics :
2
3 # 2. Identify users (potential bots) who have liked every single photo on the site, as this is not typically possible for a normal user.
4
5 select id, username from users where id in (select user_id from likes group by user_id having count(user_id) = (select count(id) from photos));
6
```

The 'Result Grid' shows the following data:

id	username
5	Aniya_Hackett
14	Jadyn81
21	Roco33
24	Maxwell_Halvorson
36	Olie_Ledner37
41	Mckenna17
54	Duane60
57	Julian_Schmidt
66	Mike_Auer39
71	Nia_Haag
75	Leslie67
76	Janelle_Miclaus81
91	Bethenny20

The 'Output' window shows the execution details for the query:

#	Time	Action	Message	Duration / Fetch
1	20:25:53	select (select count(id) from photos) / (select count(distinct user_id) from photos) as Average_No_of_Posts_Per_User, (select Count(id) from photos) / (select count(id) from photos)	1 row(s) returned	0.015 sec / 0.000 sec
2	20:32:31	select id, username from users where id in (select user_id from likes group by user_id having count(user_id) = (select count(id) from photos))	13 row(s) returned	0.015 sec / 0.000 sec

## RESULTS

- After doing this project I learnt the fundamentals and got some knowledge how the data analyst works using MySQL and how they are using database to extract the output.
- I also learnt from this database and after this project how users engage with software application or a mobile application and after doing analysis of this data, I also got knowledge how the marketing team, software team and product team works together and derive the insights for business and how the data can be used to increase and optimize the application and engagement with users and audience.

