

## **Project 2 : By Ramana Bansal**

### **Instagram User Analytics**

**Project Description :** The project focuses on a clone of Instagram user dataset. The data contains information regarding various activities such as user id, names, posts, likes, etc. of a common Instagram User. We will be using this data to derive insights such as oldest users, inactive users, bots, average number of posts, common hashtags used. The data will also be used to find the most liked photo and the most popular day of registration which will aid in finding contest winners and launching ad campaigns.

**Approach:** The data was grabbed and analysed using MySql queries. MySql Workbench was used to run various queries and observe the outputs.

**Tech-Stack Used:** MySQL Workbench 8.0.33 : To Execute the Sql queries.

**Insights:** The project was helpful in getting a general idea about the practical application of data analysis. It also helped in understanding some of the data-driven processes behind various social media such as Instagram.

**Result:** The project made me feel a little more confident with MySql and its use in data cleaning and interaction. It also gave insights regarding how data can be used to make decisions and derive policies for businesses.

## A) Marketing:

1. **Rewarding Most Loyal Users:** Find the 5 oldest users of the Instagram from the database provided.

Select \*

From Users

Order By created\_at

Limit 5;

The screenshot shows a database management interface with a menu bar (File, Edit, View, Query, Database, Server, Tools, Scripting, Help) and a toolbar. The left sidebar displays a 'SCHEMAS' tree with 'ig\_clone' expanded, showing tables like comments, follows, likes, photo\_tags, photos, tags, and users. The main query editor contains the following SQL code:

```
1
2  -- OLDEST USERS
3  • Select *
4  From Users
5  Order By created_at
6  Limit 5;
```

Below the query editor, the 'Result Grid' shows the results of the query. The table has three columns: id, username, and created\_at. 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
*	NULL	NULL

The bottom status bar indicates 'Closing Administrator.'

We sort the users table at created\_at table in descending order, and set the limit to 5 to get the five oldest users. The created\_at table gives the timestamp for when the account was created.

2. **Remind Inactive Users to Start Posting:** Find the users who have never posted a single photo on Instagram.

Select users.id, username

From users

Left join photos

On users.id = photos.user\_id

Where photos.image\_url Is Null;

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

```
-- INACTIVE USERS
Select users.id, username
From users
Left join photos
On users.id = photos.user_id
Where photos.image_url Is Null;
```

The 'Result Grid' at the bottom shows the results of the query, listing 15 inactive users with their IDs and usernames:

id	username
5	Aniya_Hackett
7	Kassandra_Homenick
14	Jadlyn81
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

The status bar at the bottom indicates 'Query Completed'.

We used left join to join the tables users and photos, and extracted the usernames which had null values for image\_urls.

3. **Declaring Contest Winner:** Identify the winner of the contest and provide their details to the team.

Select

photos.user\_id,  
users.username,  
likes.photo\_id,  
count(\*) as no\_of\_likes

From likes

Left Join photos

On photos.id = likes.photo\_id

Left Join users

On users.id = photos.user\_id

Group By photo\_id

Order By no\_of\_likes Desc

Limit 1;

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the 'ig\_clone' database, including tables like 'comments', 'follows', 'likes', 'photo\_tags', 'photos', 'tags', and 'users'. The main editor window contains a SQL query that identifies the contest winner by joining the 'likes', 'photos', and 'users' tables, grouping by 'photo\_id', and ordering by the number of likes in descending order, limiting the result to one row. The 'Result Grid' at the bottom shows the output of the query, displaying the user\_id, username, photo\_id, and no\_of\_likes for the winning photo.

```
21
22 -- CONTEST WINNER
23 • Select
24     photos.user_id,
25     users.username,
26     likes.photo_id,
27     count(*) as no_of_likes
28 From likes
29 Left Join photos
30     On photos.id = likes.photo_id
31 Left Join users
32     On users.id = photos.user_id
33 Group By photo_id
34 Order By no_of_likes Desc
35 Limit 1;
36
```

user_id	username	photo_id	no_of_likes
52	Zack_Kemmer93	145	48

We have joined the tables photos, likes and users. We found the number of likes each photo received using Group By and Count for photo\_id.

4. **Hashtag Researching:** Identify and suggest the top 5 most commonly used hashtags on the platform.

Select

tag\_id,  
tags.tag\_name,  
COUNT(\*) as tag\_count

From photo\_tags

Join tags

On photo\_tags.tag\_id = tags.id

Group By tag\_id

Order By tag\_count Desc

Limit 7;

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'ig\_clone' expanded, showing tables like 'comments', 'follows', 'likes', 'photo\_tags', 'photos', 'tags', and 'users'. The main editor window contains a SQL query to find the most common hashtags. The 'Result Grid' at the bottom shows the output of the query, listing the top 7 tags by count.

```
-- Most Common Hashtags
Select
tag_id,
tags.tag_name,
COUNT(*) as tag_count
From photo_tags
Join tags
On photo_tags.tag_id = tags.id
Group By tag_id
Order By tag_count Desc
Limit 7;
```

tag_id	tag_name	tag_count
21	smile	59
20	beach	42
17	party	39
13	fun	38
18	concert	24
11	lol	24
5	food	24

We used Group By and Count for tag\_id to count the number of times a tag was used. The tables photo\_tags and tags were joined to get the names of tags. The 5<sup>th</sup> position is shared by three hashtags, therefore we have set the limit to 7.

5. **Launch AD Campaign:** What day of the week do most users register on? Provide insights on when to schedule an ad campaign.

Select

DAYNAME(created\_at) As Registration\_Day,

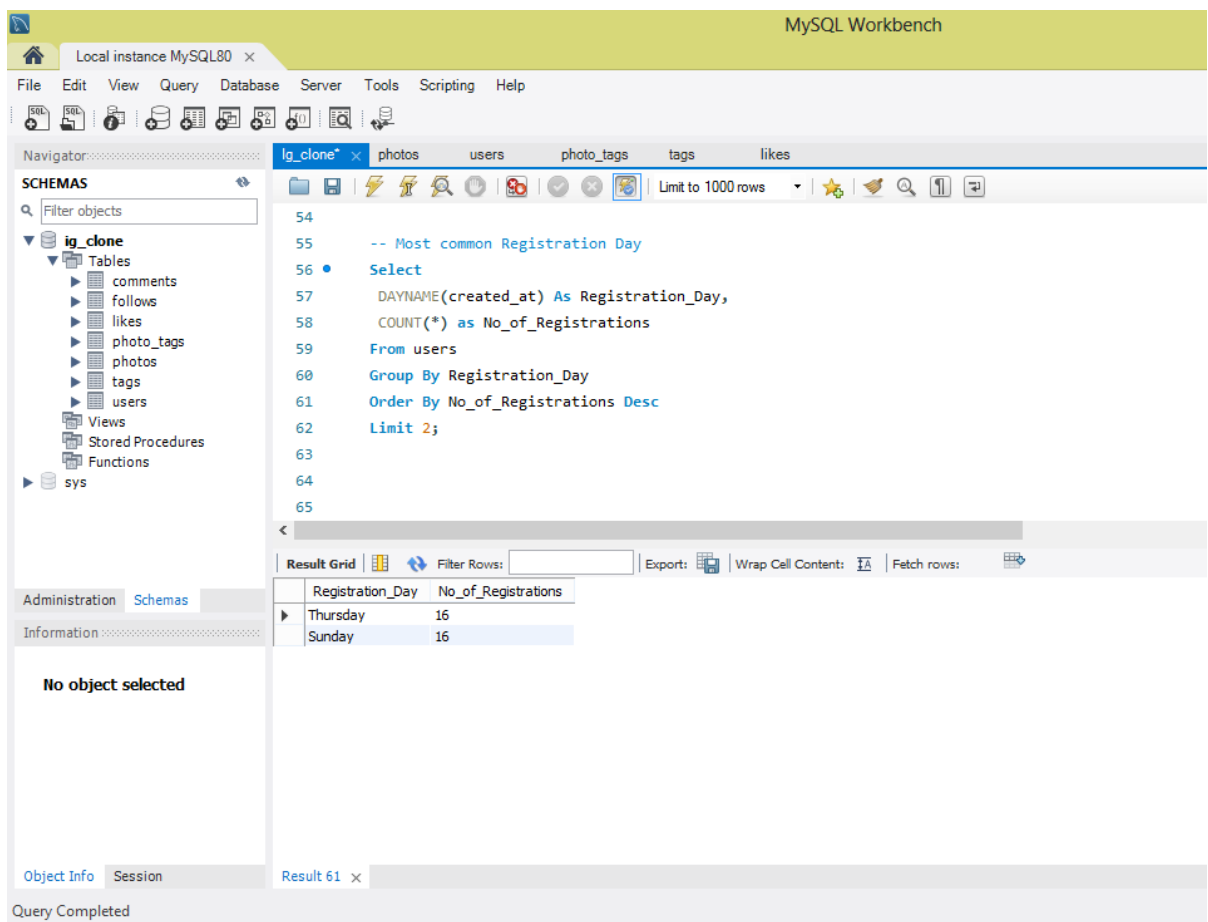
COUNT(\*) as No\_of\_Registrations

From users

Group By Registration\_Day

Order By No\_of\_Registrations Desc

Limit 2;



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'ig\_clone' selected. The main editor shows a SQL query:   
-- Most common Registration Day  
Select  
DAYNAME(created\_at) As Registration\_Day,  
COUNT(\*) as No\_of\_Registrations  
From users  
Group By Registration\_Day  
Order By No\_of\_Registrations Desc  
Limit 2;  
The 'Result Grid' at the bottom shows the following data:

Registration_Day	No_of_Registrations
Thursday	16
Sunday	16

The status bar at the bottom indicates 'Query Completed'.

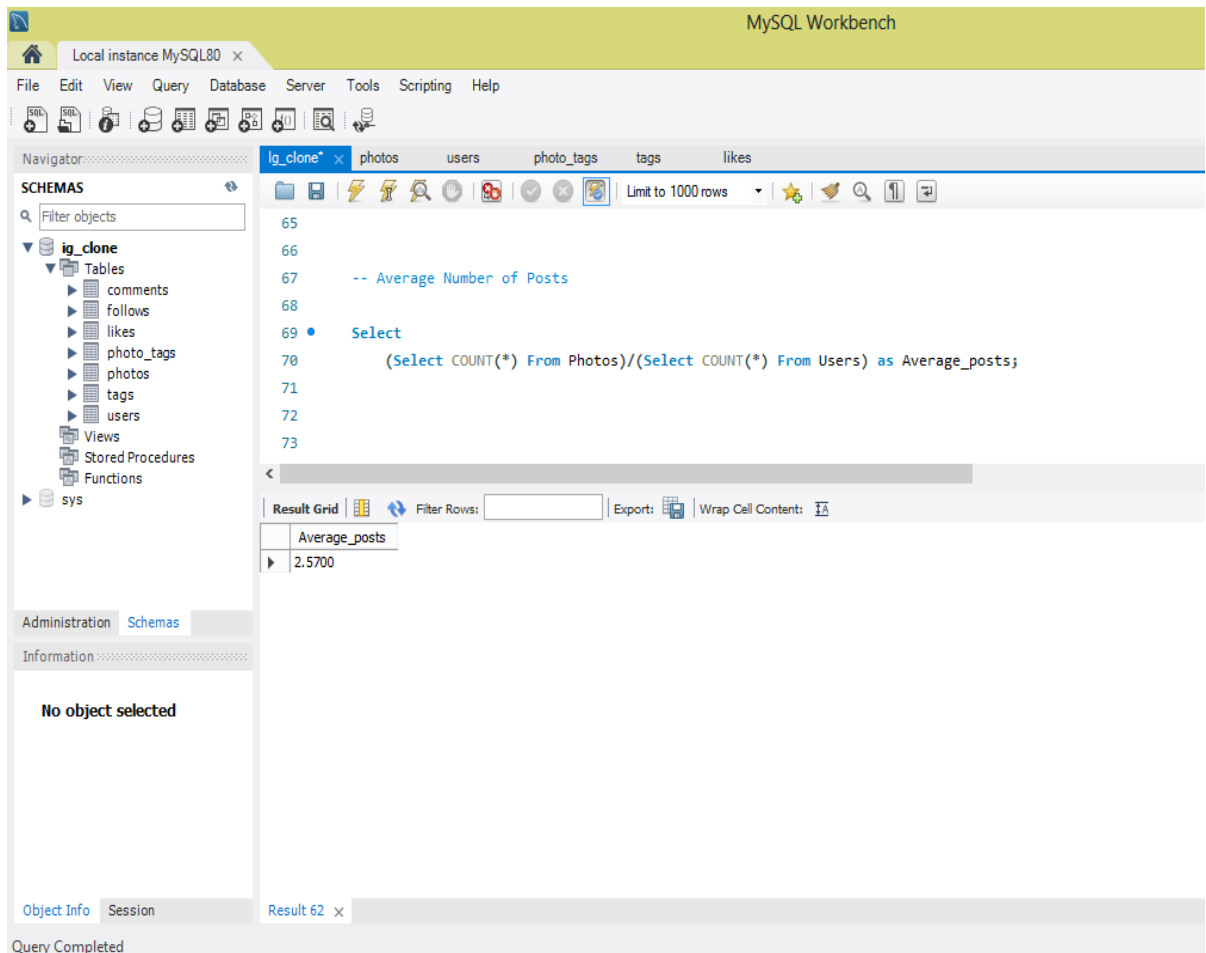
Both Thursday and Sunday have the maximum number of registrations, therefore we have set the limit to 2.

## B) Investor Metrics:

**User Engagement:** Provide how many times does average user posts on Instagram. Also, provide the total number of photos on Instagram/total number of users.

Select

(Select COUNT(\*) From Photos)/(Select COUNT(\*) From Users) as Average\_posts;



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

```
65
66
67 -- Average Number of Posts
68
69 Select
70 (Select COUNT(*) From Photos)/(Select COUNT(*) From Users) as Average_posts;
71
72
73
```

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

Average_posts
2.5700

The status bar at the bottom indicates 'Query Completed'.

Subquery was used to find the average number of posts. The average number of posts for this dataset is 2.57.

**Bots & Fake Accounts:** Provide data on users (bots) who have liked every single photo on the site (since any normal user would not be able to do this).

Select

users.id,  
users.username,  
COUNT(\*) As number\_of\_likes

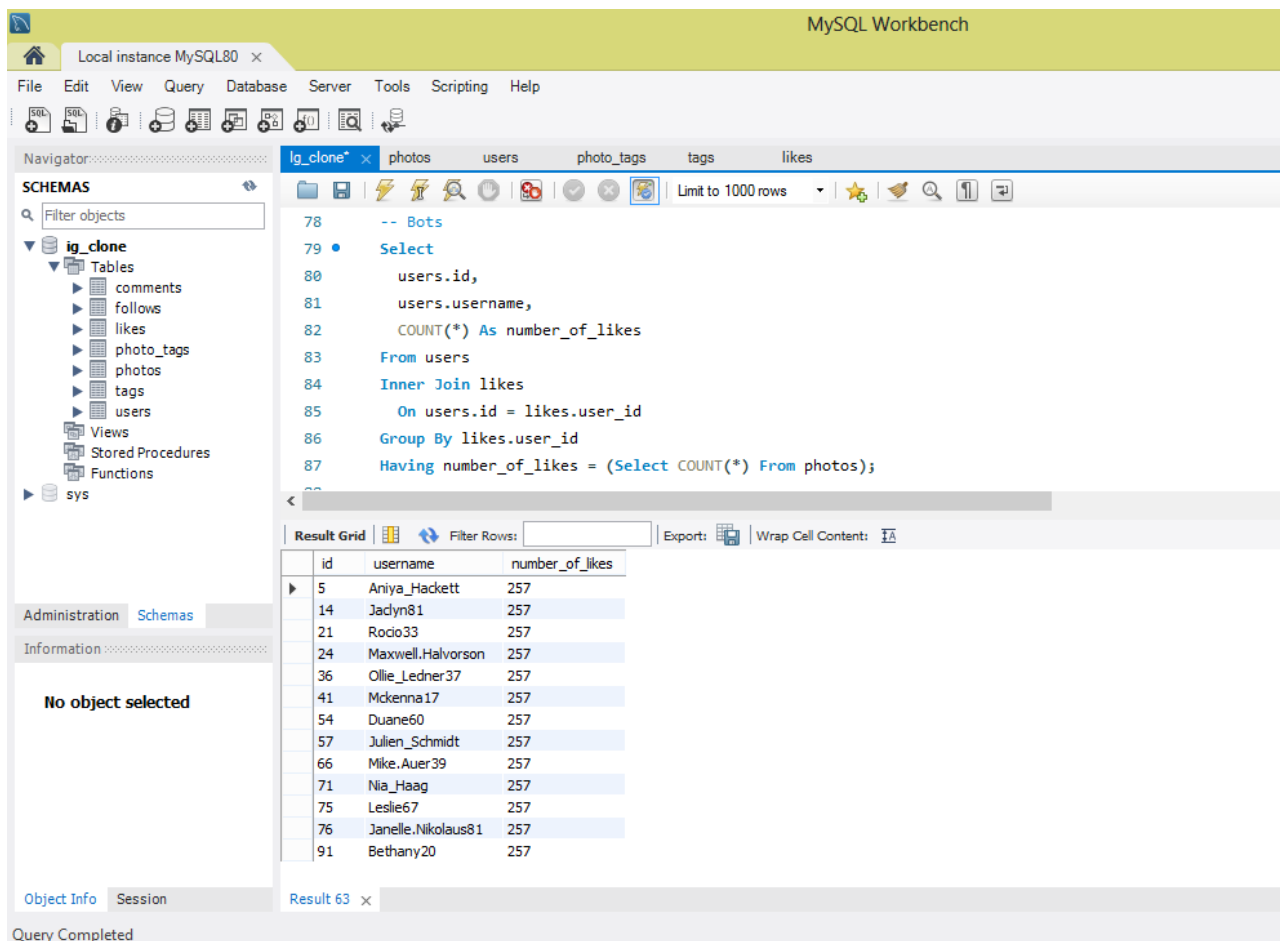
From users

Inner Join likes

On users.id = likes.user\_id

Group By likes.user\_id

Having number\_of\_likes = (Select COUNT(\*) From photos);



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'ig\_clone' selected, showing tables like comments, follows, likes, photo\_tags, photos, tags, and users. The main editor window contains a SQL query that selects user information and the count of likes, filtered by users who have liked all photos. The 'Result Grid' at the bottom shows the output of the query, listing user IDs, usernames, and the number of likes (all 257).

```
-- Bots
Select
  users.id,
  users.username,
  COUNT(*) As number_of_likes
From users
Inner Join likes
  On users.id = likes.user_id
Group By likes.user_id
Having number_of_likes = (Select COUNT(*) From photos);
```

id	username	number_of_likes
5	Aniya_Hackett	257
14	Jadyn81	257
21	Rocio33	257
24	Maxwell_Halvorson	257
36	Ollie_Ledner37	257
41	Mckenna17	257
54	Duane60	257
57	Julien_Schmidt	257
66	Mike_Auer39	257
71	Nia_Haag	257
75	Leslie67	257
76	Janelle.Nikolaus81	257
91	Bethany20	257

The number of likes for each user\_id was counted and compared to the total number of photos. The tables users and likes were joined to retrieve user information for the bots that had liked all the photos.