

Project Description:

In this project, we will use Instagram user data to analyze user interactions and their engagement with the Instagram app to provide valuable insights that can help the business grow by taking data driven decisions. User analysis involves tracking how users engage with a digital product, such as a software application or mobile app. The insights derived from the analysis can be used by various teams within the business. For example, 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 user experience.

We will use MySQL Workbench as the tool to analyze the Instagram user data and to answer the questions posed by the management team.

Tech-Stack Used:

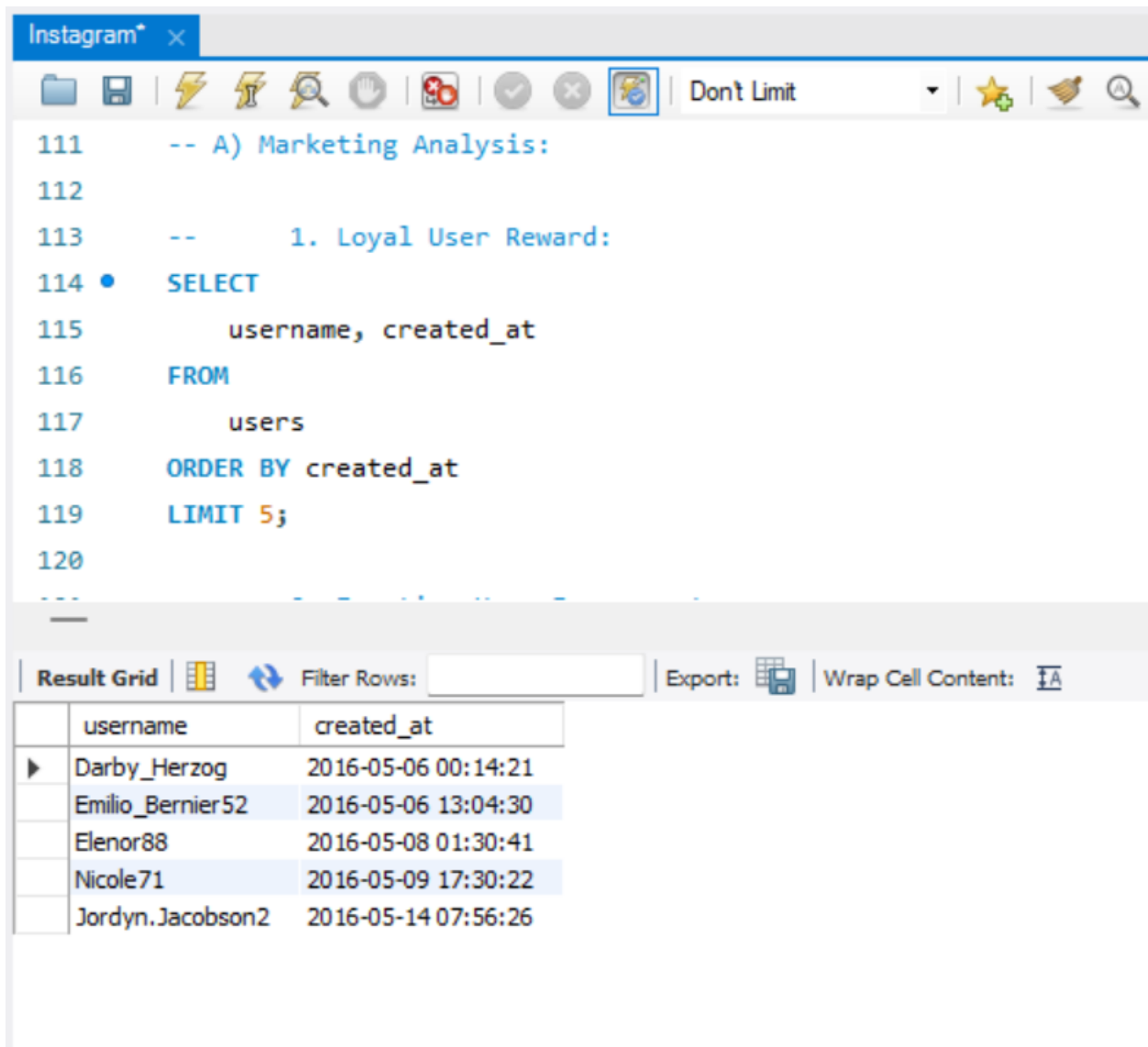
For this project, we used MySQL Workbench 8.0 to analyze the user data provided and get the insights from it. We used this tool as it is open-source, user-friendly, easy to install, it is secure and most widely used database management system in the world.

Approach:

The dataset provided was first loaded in MySQL Workbench and database was created. Then, the database was selected with **use** and checked the tables present using **select** inside the data. Checked for any missing values with **describe**. Then, we used types of **joins (inner, left)** and **with** to join the tables to fetch the required records to answer the questions posed by the management team.

A) Marketing Analysis:

1. Loyal User Reward:



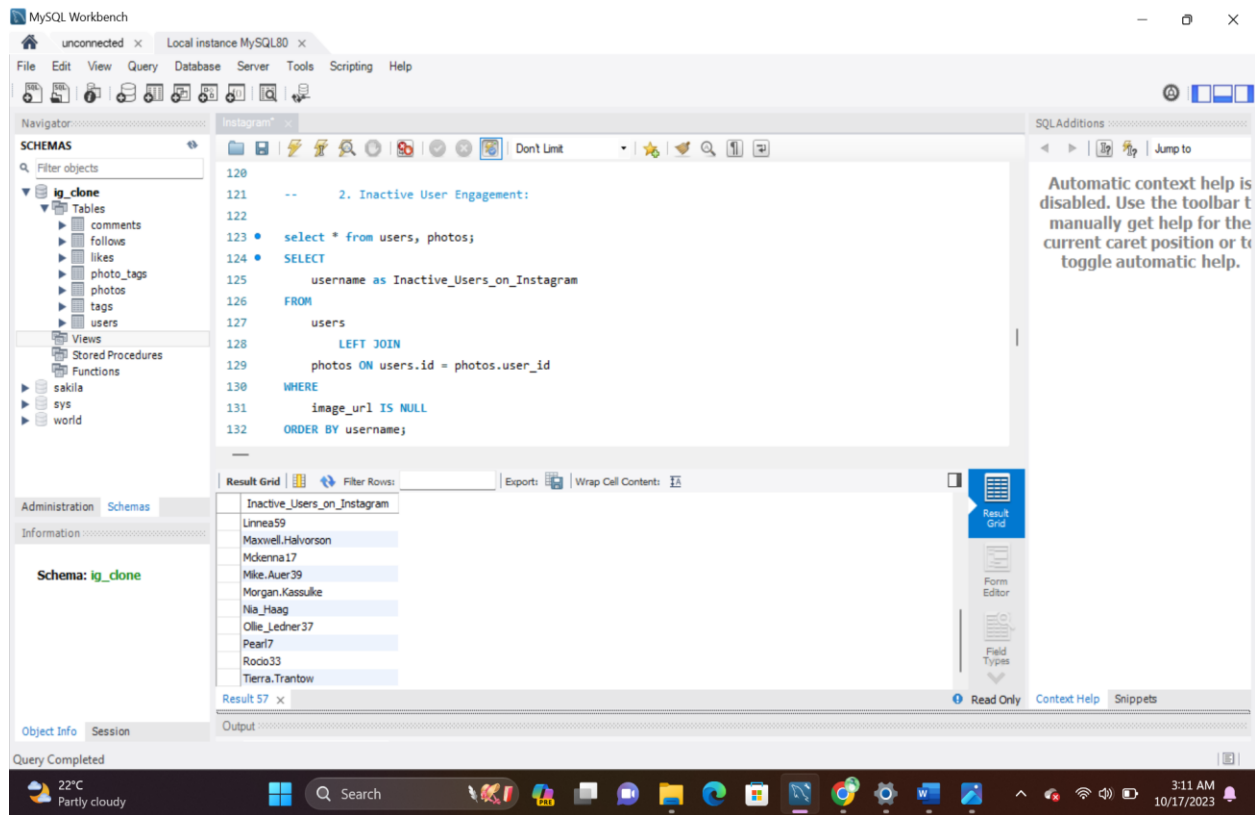
```
111 -- A) Marketing Analysis:
112
113 --      1. Loyal User Reward:
114 • SELECT
115     username, created_at
116 FROM
117     users
118 ORDER BY created_at
119 LIMIT 5;
120
...
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	username	created_at
▶	Darby_Herzog	2016-05-06 00:14:21
	Emilio_Bernier52	2016-05-06 13:04:30
	Elenor88	2016-05-08 01:30:41
	Nicole71	2016-05-09 17:30:22
	Jordyn.Jacobson2	2016-05-14 07:56:26

Here, we ordered the data with respect to **created_at** to find out top 5 oldest users on Instagram and found that the first user that registered was on 2016-05-06 00:14:21.

2. Inactive User Engagement:



The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view with 'ig_clone' expanded, listing tables like 'comments', 'follows', 'likes', 'photo_tags', 'photos', 'tags', and 'users'. The central editor window contains the following SQL query:

```
120
121 -- 2. Inactive User Engagement:
122
123 • select * from users, photos;
124 • SELECT
125     username as Inactive_Users_on_Instagram
126 FROM
127     users
128     LEFT JOIN
129     photos ON users.id = photos.user_id
130 WHERE
131     image_url IS NULL
132 ORDER BY username;
```

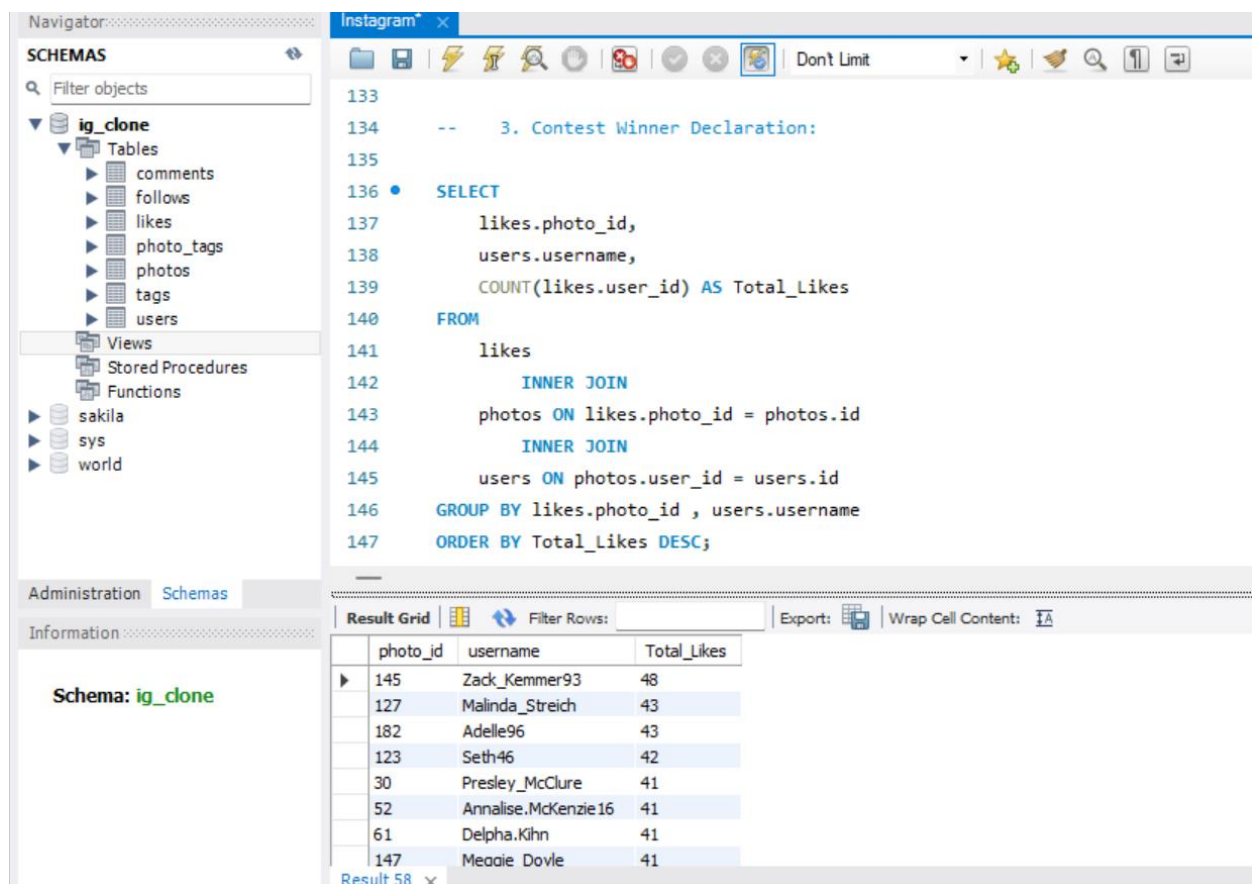
Below the query editor, the 'Result Grid' shows the results of the query, with the column header 'Inactive_Users_on_Instagram'. The results list the following usernames:

Inactive_Users_on_Instagram
Linnea59
Maxwell.Halvorson
McKenna17
Mike.Auer39
Morgan.Kassulke
Nia_Haag
Ollie_Ledner37
Pearl7
Rocio33
Tierra.Trantow

On the right side of the interface, a 'SQLAdditions' pane displays a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

In this, we joined users and photos table with **left join** to find out the users who have never posted a single photo on Instagram. We used to filter the records here based on **where image_url is null** to get the desired result.

3. Contest Winner Declaration:



The screenshot shows a database IDE with a left sidebar containing a 'SCHEMAS' tree. The 'ig_clone' schema is expanded, showing tables like 'comments', 'follows', 'likes', 'photo_tags', 'photos', 'tags', and 'users'. The main editor displays a SQL query for '3. Contest Winner Declaration:'. The query selects 'likes.photo_id', 'users.username', and 'COUNT(likes.user_id) AS Total_Likes' from the 'likes' table, joined with 'photos' and 'users' using 'INNER JOIN'. The results are grouped by 'likes.photo_id' and 'users.username', and ordered by 'Total_Likes' in descending order.

```
133
134 -- 3. Contest Winner Declaration:
135
136 • SELECT
137     likes.photo_id,
138     users.username,
139     COUNT(likes.user_id) AS Total_Likes
140 FROM
141     likes
142     INNER JOIN
143     photos ON likes.photo_id = photos.id
144     INNER JOIN
145     users ON photos.user_id = users.id
146 GROUP BY likes.photo_id , users.username
147 ORDER BY Total_Likes DESC;
```

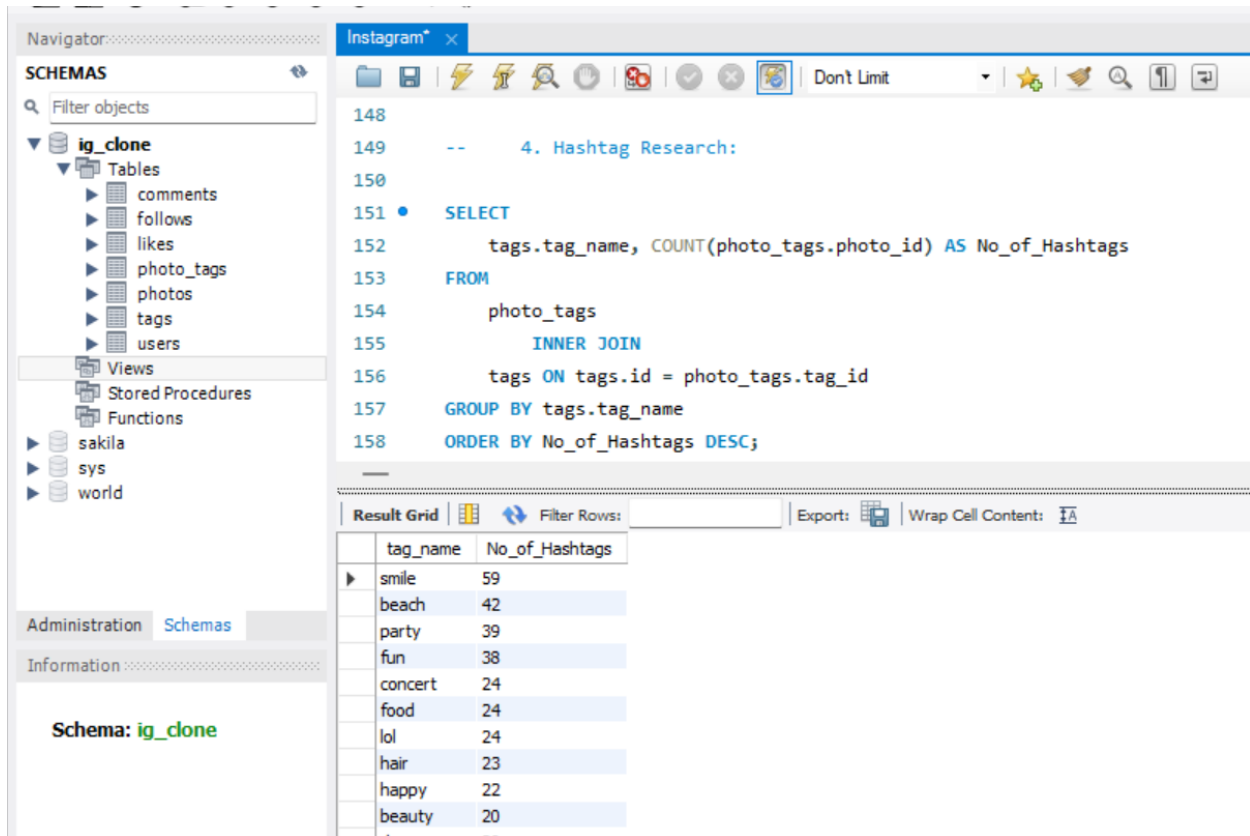
The bottom of the IDE shows the 'Result Grid' with the following data:

photo_id	username	Total_Likes
145	Zack_Kemmer93	48
127	Malinda_Streich	43
182	Adelle96	43
123	Seth46	42
30	Presley_McClure	41
52	Annalise.McKenzie16	41
61	Delpha.Kihn	41
147	Meggie_Dovle	41

In this part, we needed to find out the user who has the most number of likes on a single photo. So, we joined three tables here which are likes, photos, and users using **inner join**. We counted the number of likes by counting the **user_id** in likes table .

We found out that the user with the most likes had a total of **48 likes**.

4. Hashtag Research:



The screenshot displays a database management interface with a left-hand sidebar showing a schema named 'ig_clone' containing tables like 'comments', 'follows', 'likes', 'photo_tags', 'photos', 'tags', and 'users'. The main window shows a SQL query titled '4. Hashtag Research:' which uses an inner join between 'photo_tags' and 'tags' to count the number of hashtags per tag name. The results are shown in a table with columns 'tag_name' and 'No_of_Hashtags'.

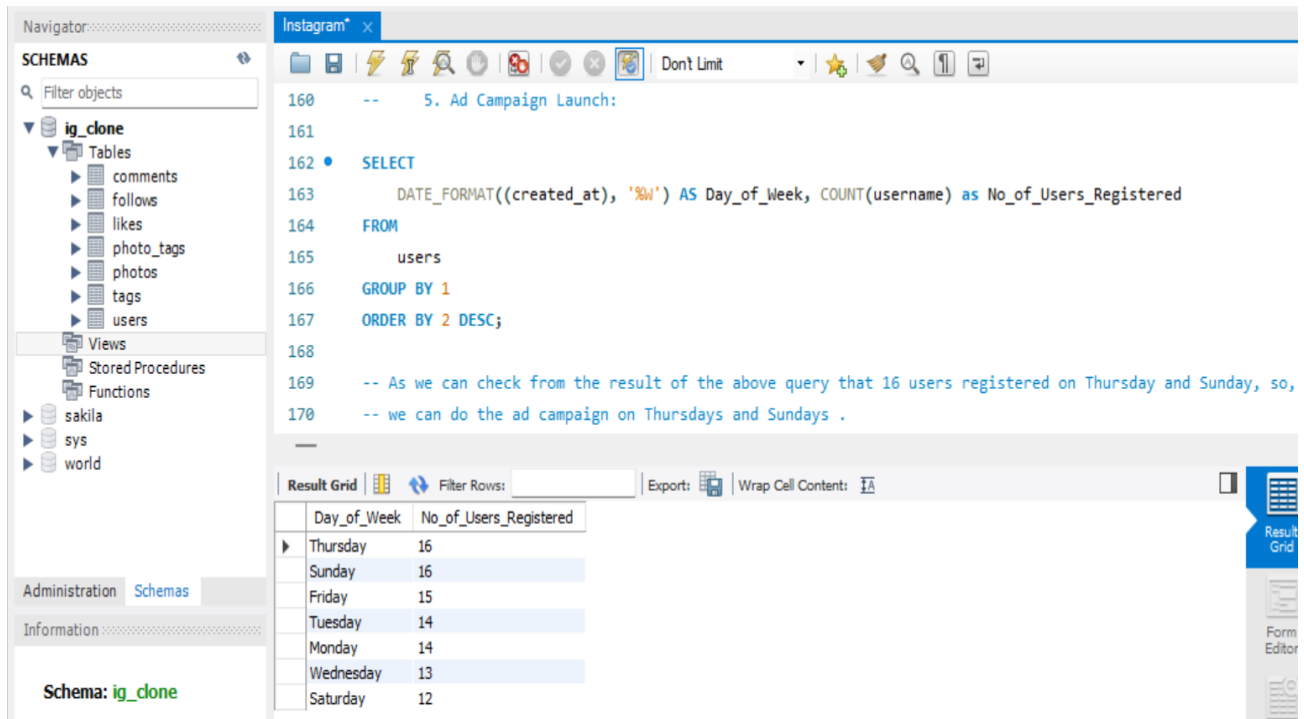
```
148
149 -- 4. Hashtag Research:
150
151 • SELECT
152     tags.tag_name, COUNT(photo_tags.photo_id) AS No_of_Hashtags
153 FROM
154     photo_tags
155     INNER JOIN
156     tags ON tags.id = photo_tags.tag_id
157 GROUP BY tags.tag_name
158 ORDER BY No_of_Hashtags DESC;
```

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

Here, we found out the most popular hashtags that could be used in posts to reach to the most people. To find out the popular hashtags, we counted the number of times a particular hashtag has been used in a post. For this, we used **photo_tags** and **tags** table and joined using **inner join**. Then, the records were grouped by the **tag_name** and ordered by number of times it has been used in descending order.

We found out that **smile, beach, party, fun, concert**, are some of the most popular and common hashtags that people use in their posts.

5. Ad Campaign Launch:



The screenshot displays a database management interface with a left-hand sidebar showing a schema tree for 'ig_clone'. The main window shows a SQL query in a text editor, and below it, a 'Result Grid' showing the output of the query. The query is a SELECT statement that formats the 'created_at' date into a day of the week and counts the number of users registered on each day, ordered by the count in descending order.

```
160 -- 5. Ad Campaign Launch:
161
162 • SELECT
163     DATE_FORMAT((created_at), '%W') AS Day_of_Week, COUNT(username) as No_of_Users_Registered
164 FROM
165     users
166 GROUP BY 1
167 ORDER BY 2 DESC;
168
169 -- As we can check from the result of the above query that 16 users registered on Thursday and Sunday, so,
170 -- we can do the ad campaign on Thursdays and Sundays .
```

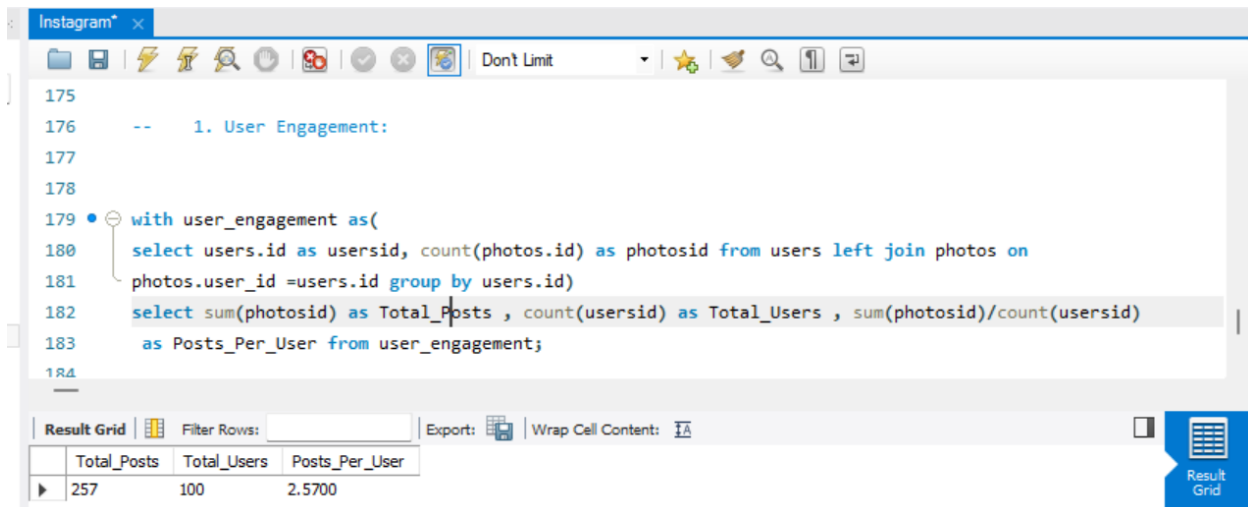
Day_of_Week	No_of_Users_Registered
Thursday	16
Sunday	16
Friday	15
Tuesday	14
Monday	14
Wednesday	13
Saturday	12

Here, we wanted to find out the best day of the week to launch a Ad campaign. To get the result, we used **DATE_FORMAT()** and fetched the day from the **created_at** column in the users table. Then, we counted the total number of users that were registered on each day of week and arranged the result in descending order in order to find out on which day most users register themselves on Instagram.

Based upon the finding, we see that on **Thursday** and **Sunday** most users registered themselves on Instagram. So, a ad-campaign should be organized on these days.

B) Investor Metrics:

1. User Engagement:



The screenshot shows a SQL query editor window titled "Instagram". The query is as follows:

```
175
176  -- 1. User Engagement:
177
178
179 with user_engagement as(
180   select users.id as usersid, count(photos.id) as photosid from users left join photos on
181   photos.user_id =users.id group by users.id)
182   select sum(photosid) as Total_Posts , count(usersid) as Total_Users , sum(photosid)/count(usersid)
183   as Posts_Per_User from user_engagement;
```

Below the query editor, the "Result Grid" is displayed with the following data:

	Total_Posts	Total_Users	Posts_Per_User
▶	257	100	2.5700

Here, the investors wanted to find out if the users are still active and posting on Instagram or if they are making fewer posts. So, we found out the average number of posts per user i.e., on an average how many posts a user is doing. To get this result, we needed to calculate the total users on Instagram and the total number of posts that have been made.

We used **with** and joined tables users and photos using **left join** and to calculate the total posts we used **sum()** and for total users we used **count()**. To find out the average no of posts per user we divided the total posts by total users.

Based on the result, we see that on an average a user makes around **2.7** posts or we can round off and say **3** posts where the total users are **100** and total posts done are **257**.

2. Bots & Fake Accounts:

The screenshot shows a database management interface with a left sidebar containing a 'SCHEMAS' tree. The 'ig_clone' schema is selected, showing tables like 'comments', 'follows', 'likes', 'photo_tags', 'photos', 'tags', and 'users'. The main area displays a SQL query in a text editor. The query includes a subquery for 'Total_Posts' and 'Total_Users', followed by a section titled '2. Bots & Fake Accounts:' which defines a CTE 'bots' and then selects usernames where the number of likes equals the total number of photos.

```
181 photos.user_id =users.id group by users.id)
182 select sum(photosid) as Total_Posts , count(usersid) as Total_Users , sum(photosid)/count(usersid)
183 as Posts_Per_User from user_engagement;
184
185 -- 2. Bots & Fake Accounts:
186
187
188 with bots as (
189 select username, count(photo_id) as No_of_Likes from users inner join likes on users.id=likes.user_id
190 group by username)
191 select username as Potential_Bots, No_of_Likes from bots where No_of_Likes=(select count(*) from photos)
192 order by No_of_Likes;
193
194
```

Below the query editor, a 'Result Grid' shows the output of the query. It has two columns: 'Potential_Bots' and 'No_of_Likes'. The results list 13 usernames, all with a value of 257 in the 'No_of_Likes' column.

Potential_Bots	No_of_Likes
Aniya_Hackett	257
Jadyn81	257
Rocio33	257
Maxwell_Halvorson	257
Ollie_Ledner37	257
Mckenna17	257
Duane60	257
Julien_Schmidt	257
Mike_Auer39	257

Here, we wanted to find out if there are any users registered on Instagram as bots which are fake and dummy accounts. In order to achieve this, we checked if there are any users who have liked every single photo on the website as this is typically not possible for a normal user.

To get the desired result, we joined the tables **users** and **likes** using **inner join** and counted the total number of likes from **count()** on **photo_id**. Then, we counted the total number of records present in **photos** table and filtered our records as the **username** of the user whose total number of likes is equal to the total number of photos.

Analyzing this, we found out there are **13** such users registered which could be a potential bot or a fake and dummy account.

Insights:

- We found out that the first user that registered on Instagram was on **2016-05-06 00:14:21**.
- We found out that there is a total of **26** such users who have never posted a single photo.
- We found out that the most number of likes on a single photo for a user is **48**.
- We found out that the most common/popular hashtags used on posts by the users are **smile, beach, party, fun, concert**.
- We found out that on **Thursdays** and **Sundays**, maximum number of users are registered on the website.
- We found out that on an average a user posts around **3 photos**.
- We found out that there is a total of **13** such users who could be a possible bot/fake/dummy account.

Result:

Based upon the findings, we are able to detect that we need more engagement from the users on the platform as the number of posts per user seems to be less. In order to achieve this, we can start a ad-campaign on the days that most users are registered. Also, we found out that out of a total of 100 users, 13 of them are not real users. So, we should either deactivate them or does not count them as a user for metrics.

We saw the most popular hashtags that people use on their posts, so, we can use them to reach a larger group of people while doing any promotions. Also, there are 26 users who have never posted a single photo on Instagram, so, we can offer these users with some sort of promotion to increase the engagement or target them with some specific ads. Also, we checked the oldest users on the app, so, we can do promotions by giving them some sort of benefits or services for free for a period of time.