

Instagram User Analytics

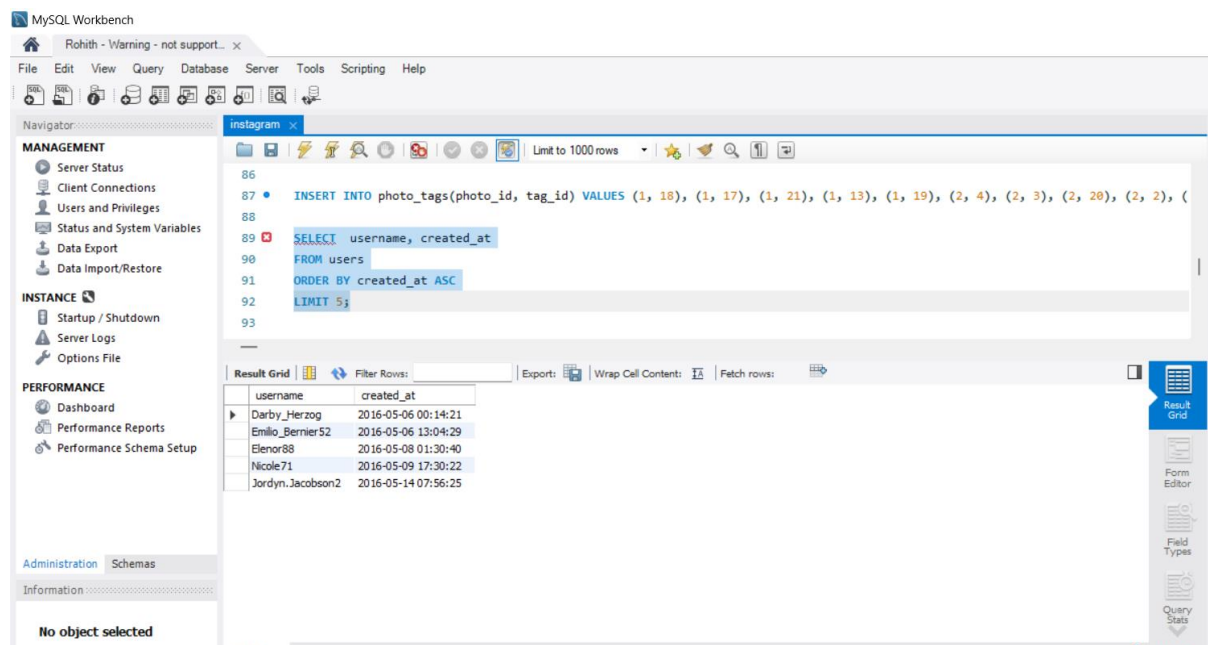
Install MySQL workbench and then import the Given Database.

SQL Tasks :

A) Marketing Analysis:

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.

Your Task: Identify the five oldest users on Instagram from the provided database.



```
SELECT username, created_at
```

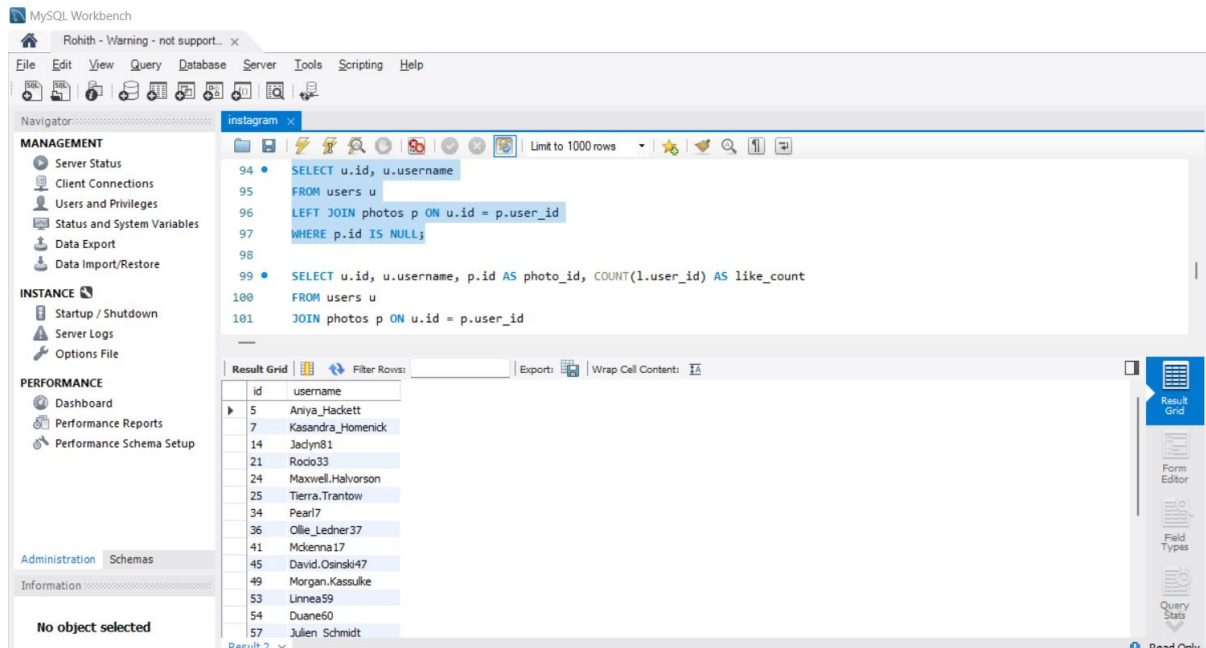
```
FROM users
```

```
ORDER BY created_at ASC
```

```
LIMIT 5;
```

B) Inactive User Engagement: The team wants to encourage inactive users to start posting by sending them promotional emails.

Your Task: Identify users who have never posted a single photo on Instagram.

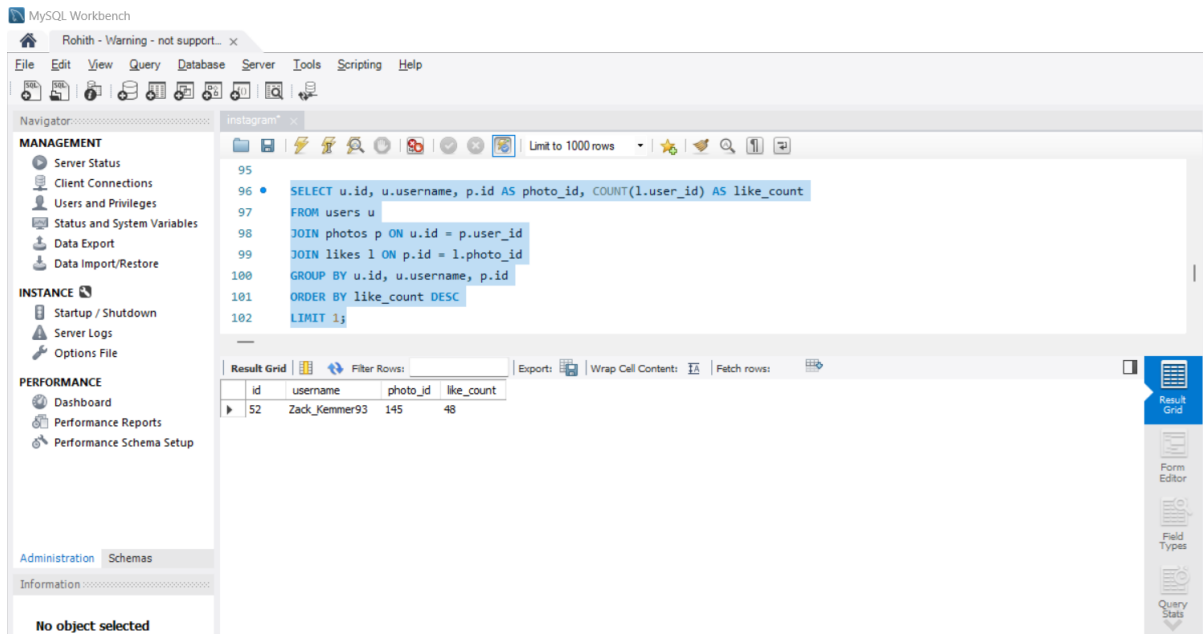


```
SELECT u.id, u.username
FROM users u
LEFT JOIN photos p ON u.id = p.user_id
WHERE p.id IS NULL;
```

3) Contest Winner Declaration: The team has organized a contest where the user with the most likes on a single photo wins.

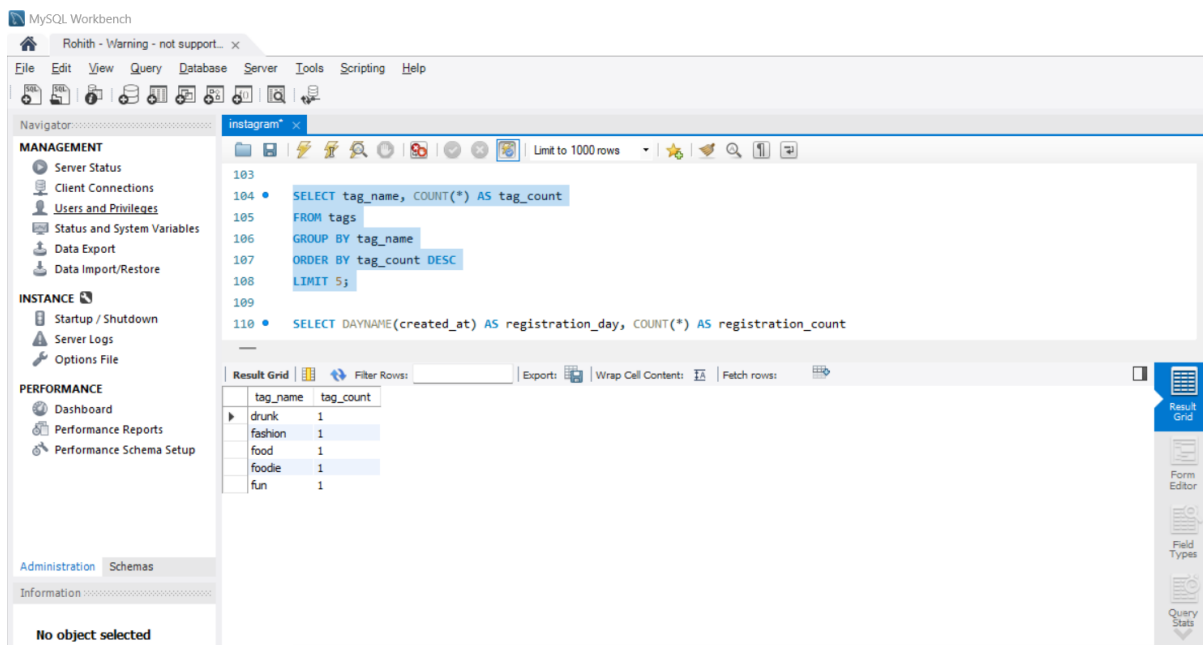
Your Task: Determine the winner of the contest and provide their details to the team.

```
SELECT u.id, u.username, p.id AS photo_id, COUNT(l.user_id) AS like_count
FROM users u
JOIN photos p ON u.id = p.user_id
JOIN likes l ON p.id = l.photo_id
GROUP BY u.id, u.username, p.id
ORDER BY like_count DESC
LIMIT 1;
```



4) Hashtag Research: A partner brand wants to know the most popular hashtags to use in their posts to reach the most people.

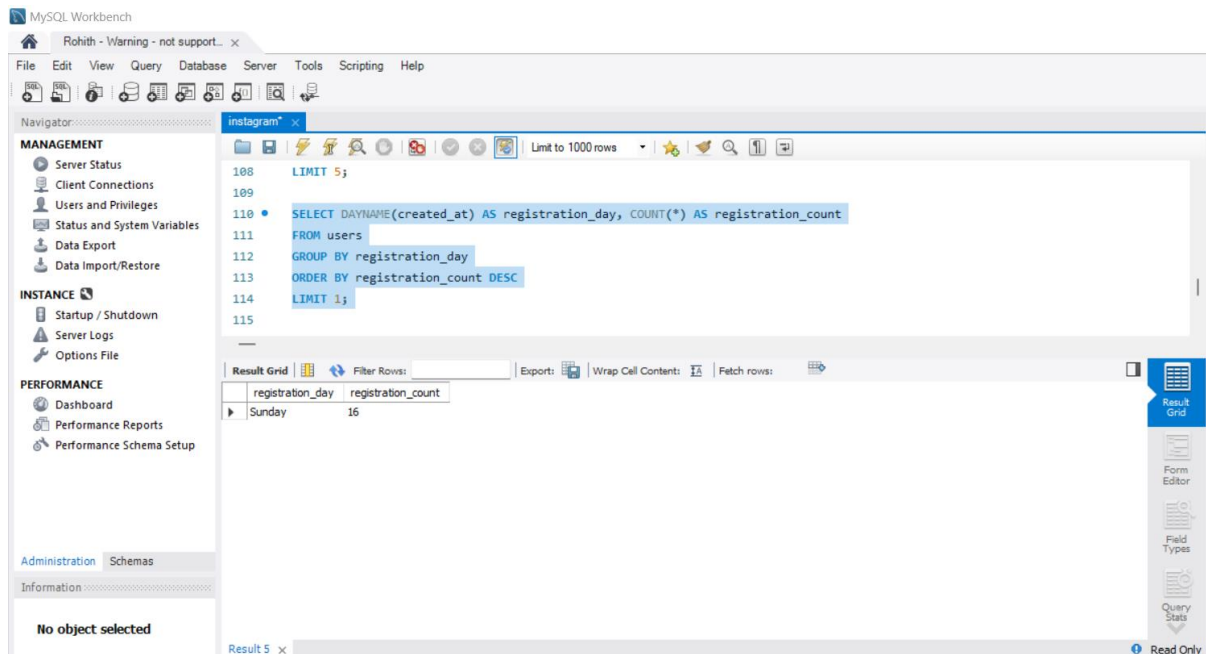
Your Task: Identify and suggest the top five most commonly used hashtags on the platform.



```
SELECT tag_name, COUNT(*) AS tag_count  
FROM tags  
GROUP BY tag_name  
ORDER BY tag_count DESC  
LIMIT 5;
```

5) Ad Campaign Launch: The team wants to know the best day of the week to launch ads.

Your Task: Determine the day of the week when most users register on Instagram. Provide insights on when to schedule an ad campaign.

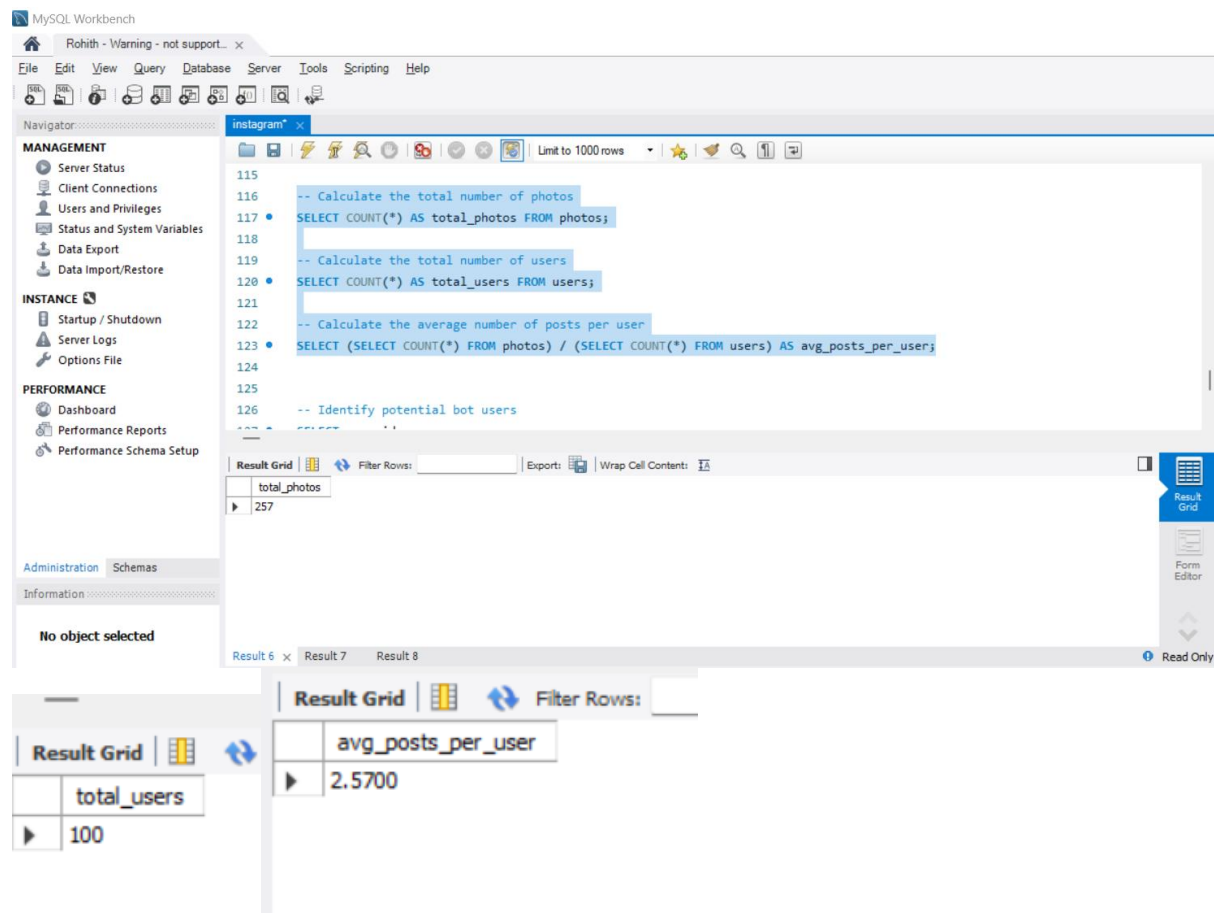


```
SELECT DAYNAME(created_at) AS registration_day, COUNT(*) AS  
registration_count  
FROM users  
GROUP BY registration_day  
ORDER BY registration_count DESC  
LIMIT 1;
```

B) 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.

Your Task: 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.



The screenshot shows the MySQL Workbench interface. The left sidebar contains the 'Navigator' pane with sections for 'MANAGEMENT' (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore), 'INSTANCE' (Startup / Shutdown, Server Logs, Options File), and 'PERFORMANCE' (Dashboard, Performance Reports, Performance Schema Setup). The main editor window displays the following SQL queries:

```
115
116 -- Calculate the total number of photos
117 • SELECT COUNT(*) AS total_photos FROM photos;
118
119 -- Calculate the total number of users
120 • SELECT COUNT(*) AS total_users FROM users;
121
122 -- Calculate the average number of posts per user
123 • SELECT (SELECT COUNT(*) FROM photos) / (SELECT COUNT(*) FROM users) AS avg_posts_per_user;
124
125
126 -- Identify potential bot users
```

Below the queries, the 'Result Grid' shows the results of the first two queries:

total_photos
257

total_users
100

At the bottom, another 'Result Grid' shows the result of the third query:

avg_posts_per_user
2.5700

-- Calculate the total number of photos

```
SELECT COUNT(*) AS total_photos FROM photos;
```

-- Calculate the total number of users

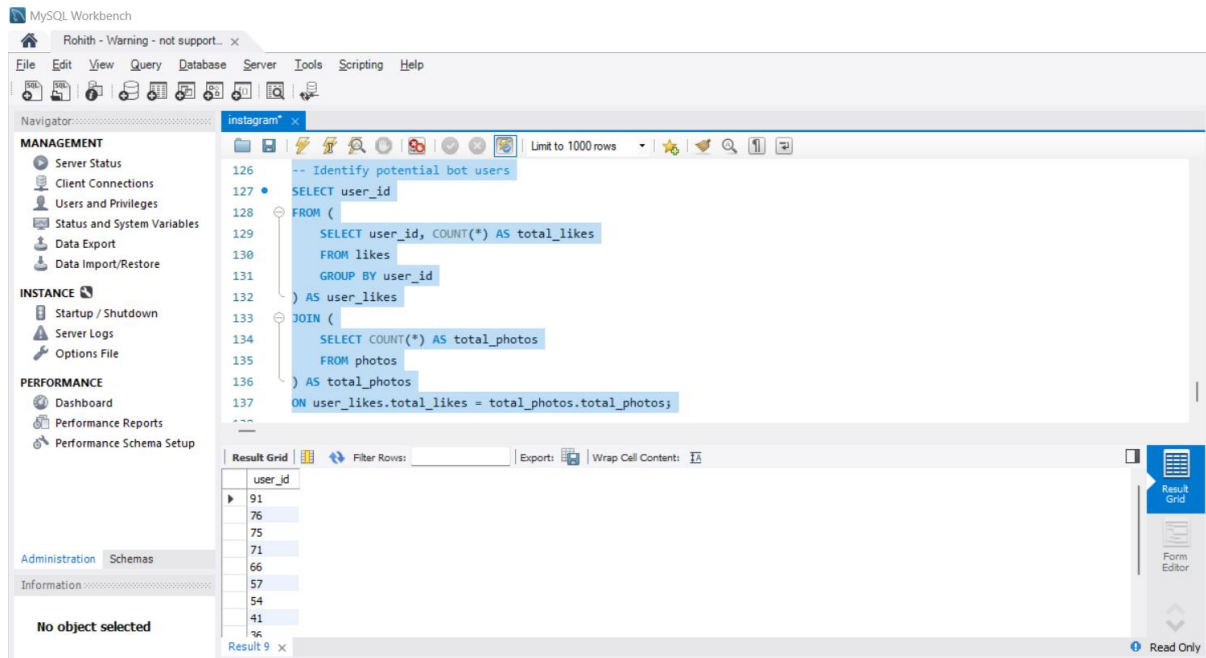
```
SELECT COUNT(*) AS total_users FROM users;
```

-- Calculate the average number of posts per user

```
SELECT (SELECT COUNT(*) FROM photos) / (SELECT COUNT(*) FROM users) AS  
avg_posts_per_user;
```

2.Bots & Fake Accounts: Investors want to know if the platform is crowded with fake and dummy accounts.

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



-- Identify potential bot users

SELECT user_id

FROM (

SELECT user_id, COUNT(*) AS total_likes

FROM likes

GROUP BY user_id

) AS user_likes

JOIN (

SELECT COUNT(*) AS total_photos

FROM photos

) AS total_photos

ON user_likes.total_likes = total_photos.total_photos;