**Instagram User Analytics**

This project uses SQL to analyze user interactions and engagement data on Instagram to provide insights for the product team to improve the app.

**Tech-Stack Used** : MySQL 8.0 CE

1. User Friendly
2. Performance
3. Open Source

The tasks encompass marketing analysis, including the identification of loyal users, encouragement of inactive users, declaration of contest winners, and exploration of popular hashtags. Furthermore, investor metrics are scrutinized to evaluate user engagement levels and identify possible instances of fake accounts.

* Loyal Users
* Inactive Users
* Contest Winner
* Hashtag Research
* Ad Campaign
* User Engagement
* Bots & Fake Accounts

**Marketing Analysis**

**1.Loyal Users**

The marketing team wants to reward the most loyal users, i.e., those who have been using the platform for the longest time.

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

**Analysis**

The task is to display the five oldest instagram users. We may approach the question by selecting the users based on the date of creation in ascending order which would display the oldest at the top and only show the top 5, ie. the oldest 5.

**QUERY**

SELECT

id, username, created\_at

FROM

users

ORDER BY created\_at

LIMIT 5;

A screenshot of a computer

Description automatically generated

**2.Inactive User Engagement**

The team wants to encourage inactive users to start posting by sending them promotional emails.

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

**Analysis**

The required attributes are scattered across more than one table for this task. We have to approach the problem by utilising inner joins to avoid the errors caused by Non aggregrated columns during group by. We find the Id of the photo with the highest like count by using count and group by and joins the username and user id of the user from the users table using inner joins.

**QUERY**

SELECT

users.id AS `Inactive IDs`,

users.username AS `Inactive Usernames`

FROM

users

LEFT JOIN

photos ON photos.user\_id = users.id

WHERE

photos.id IS NULL

GROUP BY users.id;

A screenshot of a computer

Description automatically generated

**3.Contest Winner Declaration**

The team has organized a contest where the user with the most likes on a single photo wins.

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

**Analysis**

The required attributes are scattered across more than one table for this task. We have to approach the problem by utilising inner joins to avoid the errors caused by Non aggregrated columns during group by. We find the Id of the photo with the highest like count by using count and group by and joins the username and user id of the user from the users table using inner joins.

**QUERY**

SELECT

users.username AS Winner, users.id, likes.photo\_id

FROM

likes

INNER JOIN

photos ON photos.id = likes.photo\_id

INNER JOIN

users ON users.id = photos.user\_id

GROUP BY likes.photo\_id

ORDER BY COUNT(photo\_id) DESC

LIMIT 1;

A screenshot of a computer

Description automatically generated

**4.Hashtag Research**

A partner brand wants to know the most popular hashtags to use in their posts to reach the most people.

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

**Analysis**

To identify the top five most used hashtags on the platform, we use Common Table Expression. The CTE also uses a nested query that counts the occurrences of each hash tag in the table and assigns a rank to each tag based on its count or repetitions, with the highest count receiving a rank of 1. The outer query then filters the results to include only the top five ranked tags.

**QUERY**

WITH CTE AS (

SELECT tag\_id,ranking

FROM(

SELECT tag\_id,

RANK() OVER(ORDER BY count(tag\_id) DESC) AS Ranking

FROM photo\_tags

GROUP BY tag\_id

) AS RankingTable

WHERE ranking <= 5

)

SELECT CTE.ranking, tags.tag\_name

FROM CTE

INNER JOIN tags

ON CTE.tag\_id=tags.id

ORDER BY ranking;

A screenshot of a computer

Description automatically generated

**5.Ad Campaign Launch**

The team wants to know the best day of the week to launch ads.

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

**Analysis**

To find the day with the most number of registrations we may arrange the value in descending order by the count or number of registrations per each day of the week. Limiting the value to show only 1, ie. only the largest count or highest number of registrations will give us the best day for and Ad Campaign.

**QUERY**

SELECT

DAYNAME(created\_at) AS `Best day for Ads`

FROM

users

GROUP BY `Best day for Ads`

ORDER BY COUNT(DAYNAME(created\_at)) DESC

LIMIT 1;

A screenshot of a computer

Description automatically generated

**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.

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.

**Analysis**

In order to find the Average number of posts per user we may divide the total number of posts by total number of users. But the problem we run into is that we do not have the table indicating the number of photos per user Id but instead as who posted which photo. So we have to find the sum of number of photos per user Id, but we cannot find the sum of the count of photos per used Id in one single statement leading to the use of a derived table wherein we first calculate the count or number of posts per user Id and then find its sum.

**QUERY**

SELECT

SUM(posts) / COUNT(posts) AS `Average number of posts per user`

FROM

(SELECT

COUNT(user\_id) AS posts

FROM

photos

GROUP BY user\_id) AS t1;

A screenshot of a computer

Description automatically generated

**2.Bots & Fake Accounts**

Investors want to know if the platform is crowded with fake and dummy accounts.

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

**Analysis**

We are given the guideline to mark user Ids who have liked every single photo as such a feat is not possible by a regular user. We may filter the user Ids with number of occurences in the photos, indicating the number or count of likes, that are equal to the maximum value of photo Id which should be the value of the last photo id, so logically it should give the number of photos in total.

**QUERY**

SELECT

user\_id AS `BOT Ids`

FROM

likes

GROUP BY user\_id

HAVING COUNT(photo\_id) = MAX(photo\_id);

A screenshot of a computer

Description automatically generated

**INSIGHTS**

Accomplished all assigned tasks in the project, delivering actionable insights for both the marketing and investor teams. The analysis provides valuable data for decision-making, including strategies for retaining users, promotional approaches, and maintaining platform integrity. The project deepens comprehension of user interactions on Instagram, opening avenues for data-driven decisions and potential enhancements to the application.

**RESULT**

* Deep understanding of the topic
* Improved problem-solving skills
* Increased confidence