INSTAGRAM

Project description

The project involves the analysis of user interaction data on Instagram. Various attributes, including the number of users on the app, user posts, tags used, and likes on others' posts, have been meticulously collected. The primary objective is to derive meaningful insights from this data to inform the development of a well-targeted marketing plan and the enhancement of app features.
The methodology involves employing SQL queries on a MySQL Workbench database. Each query is carefully designed to extract specific information relevant to the management's inquiries. The focus is on crafting clean, optimized queries to ensure efficiency in data retrieval.

Approach Undertaken

In executing the project, a meticulous and systematic approach was adopted, ensuring clarity in understanding each query posed by the management. The process involved the following key steps:

1.Formulation of Entity Relationship Diagram
2. Step by step breakdown of the problem
3. Formulation of query step by step

Key Benefits of the Approach:

1. Clarity in Understanding
2. Precise Query Interpretation
3. Optimized Data Retrieval
4. Insightful Analysis
5. Alignment with Objectives

TECH STACK USED

MySQL Workbench was used to carry out the operations. It is an open-source RDBMS.

INSIGHTS

The marketing plan of awarding the most loyal users and making arrangements for involvement of inactive users will help in increasing the popularity of the app.

Engagement Levels:
Identify the level of engagement by analyzing the number of comments on photos. High comment counts may indicate active and engaged followers.

Content Popularity:
Understand which photos receive the most comments. This can help identify popular content and trends.

*User Sentiment***:**

Conduct sentiment analysis on comments to gauge how users feel about the content. Positive or negative sentiments can provide valuable feedback.

Top Commenters:
Identify users who frequently comment on photos. This could help identify influencers or highly engaged community members.

Topic Analysis:
Analyze the content of comments to identify recurring topics or themes. This can inform content creation strategies.

Engagement by User Type:
Categorize users based on their commenting behavior (e.g., frequent commenters, occasional commenters). This segmentation can inform targeted engagement strategies.

Campaign Impact:
Analyze comments related to specific campaigns or promotions to assess their impact on user engagement.

RESULT
The project helped in clear understanding of real-life projects and the formulation of queries and the thought-process required to understand the demand of the question.

#MARKETING ANALYSIS

#LOYAL USER AWARD 5 OLDEST USERS

```
SELECT username, created_at
FROM users
ORDER BY created_at
LIMIT 5;
```

OUTPUT

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

#INACTIVE USERS ENGAGEMENT

```
SELECT username
FROM users
WHERE users.id NOT IN (
SELECT user_id
FROM photos);
```

OUTPUT

username
Aniya_Hackett
Kasandra_Homenick
Jaclyn81
Rocio33
Maxwell.Halvorson
Tierra.Trantow
Pearl7
Ollie_Ledner37
Mckenna17
David.Osinski47
Morgan.Kassulke
Linnea59
Duane60
Julien_Schmidt
Mike.Auer39
Franco_Keebler64
Nia_Haag
Hulda.Macejkovic
Leslie67
Janelle.Nikolaus81
Darby_Herzog
Esther.Zulauf61
Bartholome.Bernhard
Jessyca_West
Esmeralda.Mraz57
Bethany20

```
#USER WITH THE MOST LIKES ON SINGLE PHOTO

Select u.id, u.username, img_url  From photos p, users u
where p.user_id = u.id
AND p.id = (SELECT photo_id FROM LIKES
                GROUP BY photo_id
                                HAVING count(user_id)=(SELECT MAX(u) FROM (SELECT
COUNT(user_id) u FROM LIKES
GROUP BY photo_id) T
```

));

OUTPUT

id	username	img_url
52	Zack_Kemmer93	https://jarret.name

```
SELECT u.* FROM users u, photos p
WHERE p.id = (
                                SELECT photo_id FROM likes
                                GROUP BY photo_id
                                HAVING
                                COUNT(user_id) = (SELECT COUNT(user_id) u
                                FROM likes
                                GROUP BY photo_id
                                ORDER BY u DESC
                                LIMIT 1))
AND p.user_id = u.id;
```

OUTPUT

id	username	created_at
52	Zack_Kemmer93	2017-01-01 05:58:22

```
WITH likes_on_photo AS(
                        SELECT COUNT(*) cnt, photo_id
                        FROM likes
                        GROUP BY photo_id)
SELECT username, user_id, cnt
FROM (photos p INNER JOIN likes_on_photo lp
ON p.id= lp.photo_id) INNER JOIN users u
ON p.user_id =u.id
WHERE cnt = (SELECT MAX(cnt) FROM likes_on_photo);
```

OUTPUT

username	user_id	cnt
Zack_Kemmer93	52	48

```
#MOST POPULAR 5 HASHTAGS

SELECT COUNT(1) cnt, tag_name
FROM photo_tags pt INNER JOIN tags
ON pt.tag_id= tags.id
GROUP BY tag_name
ORDER BY cnt DESC
LIMIT 5;
```

cnt	tag_name

59	smile
42	beach
39	party
38	fun
24	concert

#AD CAMPAIGN LAUNCH

```
WITH Joining_day AS(
SELECT DATE_FORMAT(created_at,'%W') AS Day_of_joining
FROM users)
SELECT COUNT(Day_of_joining) cnt, Day_of_joining
FROM Joining_day
GROUP BY Day_of_joining
ORDER BY cnt DESC
LIMIT 1;
```

OUTPUT

cnt	Day_of_joining
16	Thursday

#INVESTOR METRICS

#AVERAGE POSTS PER USER

```
WITH Count_of_photos AS(
SELECT COUNT(id) cnt, user_id
FROM photos
GROUP BY user_id)
SELECT AVG(cnt)
FROM count_of_photos;
```

```
SELECT count(id)/count(distinct user_id) AS AVG
FROM photos;
```

OUTPUT
3.4865

#USER WHO HAS LIKED EVERY SINGLE PHOTO

```
SELECT user_id
FROM likes
GROUP BY user_id
HAVING COUNT(photo_id)=(
SELECT COUNT(id)
FROM PHOTOS);
```

OUTPUT

-