INSTAGRAM USER ANALYTICS

Project Description:

The project aims to analyze data from a social media platform- Instagram to gain insights into its user's activity. The primary goals were to 1) Identify 5 oldest users, 2) Identify inactive users, 3) Identify the winner of a competition, 4) Figure out the 5 most popular hashtags, 5) Determine the most active registration day, 6) Find the average no. of posts per user, and 7) Single out dummy accounts.

Here, I used SQL queries to analyze the data which will further help in making the user experience seamless.

Approach:

1) Identify 5 oldest users: Here, I rearranged the data of “created_at” from the “users” table in descending order to find the oldest users and set the limit to 5 to get the result of only 5 top values.
2) Identify inactive users: Here, to figure out inactive users, I performed a LEFT JOIN on “id” from “users” and “user_id” from photos. Then, to identify such users, I demanded “id and username” of all users whose “photos.user_id” is null.
3) Identify the winner of the competition: A user will win when it has a maximum number of likes on a photo. To determine this, I have done an inner join to club “photo_id” from the “likes and photos” table and “user_id” from the “users and photos” table.
4) 5 most popular hashtags: To find this, I have rearranged the “tag_name” column “count” in descending order after performing a GROUP BY operation on tag_name.

5) Most active user registration day: To find this, I have used the “DAYNAME” function to find the day of each registration. Now, I performed a GROUP BY on “created_at” from the “users” table and then rearranged the column on the basis of registration count in descending order.
6) Average no. of posts per user: For this, I counted the “id” and “unique user_id” from the “photos” table and divided them to get the average number of posts per user.
7) Single out dummy accounts: To figure this out, I performed a GROUP BY operation on “user_id” from “likes” table. Then I counted if the unique “photos_id” is equal to “total no of photos” which will give us all the dummy accounts.

Tech-Stack Used:

Here, I have used MySQL Workbench because :

1) It has a friendly user interface.
2) It has an active user community support.

Insights:

1) Identifying the oldest, inactive, and average number of posts per user helped to understand the user activity in much better way. It can further help in optimizing the user experience.
2) Identifying the winner on the basis of likes and most popular hashtags helped us figure out what is the best content for influencers.
3) Figuring out the busiest registration day will help in scheduling ad campaigns to get the maximum benefit out of them.
4) Identifying dummy accounts will help us to delete their profile thus giving us more accurate data to analyze for better insights.

Result:

With this project, I was able to work on a real-world project and get hands-on experience in the field of data analysis. The insights from

this analysis are very powerful and actions taken from these insights will be much helpful in creating a bigger impact by engaging users.

SQL Queries and their result:

A.

1)QUERY

```
USE ig_clone;
#'A'  '1'
select id from users;
SELECT id, username, created_at
FROM users
ORDER BY created_at asc
LIMIT 5;
```

RESULT:

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	NULL

2) QUERY

```
#'2'
select * from photos;
SELECT u.id, username
FROM users as u
LEFT JOIN photos On u.id = photos.user_id
WHERE photos.user_id IS NULL;
```

RESULT

id	username
5	Aniya_Hackett
7	Kasandra_Homenick
14	Jadyn81
21	Rocio33
24	Maxwell.Halvorson
25	Tierra.Trantow

id	username
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
76	Janelle.Nikolaus81
80	Darby_Herzog
81	Esther.Zulauf61
83	Bartholome.Bernhard
89	Jessyca_West
89	Jessyca_West
90	Esmeralda.Mraz57
91	Bethany20

3) QUERY

```
#'3'
#userid, username,image_url,likes
select * from likes;
select * from photos;
select * from users;
SELECT u.id, u.username, p.image_url, COUNT(*) as likes
FROM users AS u
INNER JOIN photos AS p ON p.user_id = u.id
INNER JOIN likes AS l ON p.id = l.photo_id
GROUP BY u.id, u.username, p.image_url
ORDER BY likes DESC
LIMIT 1;
```

RESULT

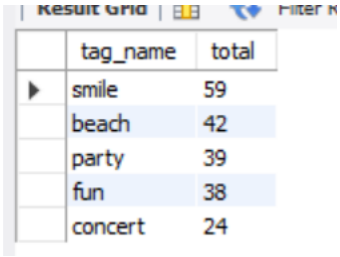
id	username	image_url	likes
52	Zack_Kemmer93	https://jarret.name	48

Zack_Kemmer93

4) QUERY

```
#'4'
select * from tags;
select * from photo_tags;
SELECT t.tag_name, COUNT(*) as total
FROM tags as t
INNER JOIN photo_tags as p
ON t.id = p.tag_id
GROUP BY t.tag_name
ORDER BY total DESC
LIMIT 5;
```

RESULT

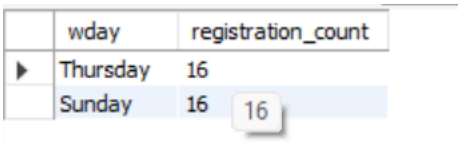


tag_name	total
smile	59
beach	42
party	39
fun	38
concert	24

5) QUERY

```
#'5'
SELECT
    DAYNAME(created_at) AS wday,
    COUNT(*) AS registration_count
FROM
    users
GROUP BY
    wday
ORDER BY
    registration_count DESC
LIMIT 2;
```

RESULT



wday	registration_count
Thursday	16
Sunday	16

B.

1) QUERY

```sql
SELECT
    user_id,
    COUNT(*) AS total_posts,
    COUNT(*) / COUNT(DISTINCT user_id) AS average_posts_per_user
FROM
    photos
GROUP BY
    user_id;

#'B'
#'1'
select COUNT(*) AS Total_Photos, COUNT(DISTINCT user_id) AS Total_Users,
       COUNT(*) / COUNT(DISTINCT user_id) AS Avg_Photos_Per_User
FROM photos;
```

RESULT:

user_id	total_posts	average_posts_per_user
1	5	5.0000
2	4	4.0000
3	4	4.0000
4	3	3.0000
6	5	5.0000
8	4	4.0000

user_id	total_posts	average_posts_per_user
9	4	4.0000
10	3	3.0000
11	5	5.0000
12	4	4.0000
13	5	5.0000
15	4	4.0000

user_id	total_posts	average_posts_per_user
16	4	4.0000
17	3	3.0000
18	1	1.0000
19	2	2.0000
20	1	1.0000
22	1	1.0000

user_id	total_posts	average_posts_per_user
23	12	12.0000
26	5	5.0000
27	1	1.0000
28	4	4.0000
29	8	8.0000
30	2	2.0000

user_id	total_posts	average_posts_per_user
32	4	4.0000
33	5	5.0000
35	2	2.0000
37	1	1.0000
38	2	2.0000
39	1	1.0000

user_id	total_posts	average_posts_per_user
40	1	1.0000
42	3	3.0000
43	5	5.0000
44	4	4.0000
46	4	4.0000
47	5	5.0000

user_id	total_posts	average_posts_per_user
48	1	1.0000
50	3	3.0000
51	5	5.0000
52	5	5.0000
55	1	1.0000
56	1	1.0000

user_id	total_posts	average_posts_per_user
58	8	8.0000
59	10	10.0000
60	2	2.0000
61	1	1.0000
62	2	2.0000
63	4	4.0000

user_id	total_posts	average_posts_per_user
64	5	5.0000
65	5	5.0000
67	3	3.0000
69	1	1.0000
70	1	1.0000
72	5	5.0000

user_id	total_posts	average_posts_per_user
73	1	1.0000
77	6	6.0000
78	5	5.0000
79	1	1.0000
82	2	2.0000
84	2	2.0000

1.0000

user_id	total_posts	average_posts_per_user
85	2	2.0000
86	9	9.0000
87	4	4.0000
88	11	11.0000
92	3	3.0000
93	2	2.0000

user_id	total_posts	average_posts_per_user
94	1	1.0000
95	2	2.0000
96	3	3.0000
97	2	2.0000
98	1	1.0000
99	3	3.0000

98	1	1.0000
99	3	3.0000
100	2	2.000 3.0000

Total_Photos	Total_Users	Avg_Photos_Per_User
257	74	3.4730

2) QUERY:

```
#"B'
#'2'
SELECT user_id
FROM likes
GROUP BY user_id
HAVING COUNT(DISTINCT photo_id) = (SELECT COUNT(*) FROM photos);
```

RESULT:

user_id
5
14
21
24
36
41

user_id
54
57
66
71
75
76

	75
	76
	91