

Instagram user Analytics Project-2

(Trainity Assignment)

SQL Task:

A) Marketing Analysis:

1. **Loyal user reward:** Identify the five oldest users on Instagram from the provided data.

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the following SQL code:


```

1 • SELECT * from ig_clone.users
2   ORDER BY created_at ASC
3   LIMIT 5;
4
      
```
- Result Grid:** Displays the results of the query, showing 5 rows of user data:

	id	username	created_at
1	62	Anya.Hackett	2016-07-01 04:39
2	67	Emilio_Bernier52	2016-05-06 13:04:30
3	63	Elenor63	2016-05-08 01:30:41
4	95	Nicole71	2016-05-09 17:30:22
5	38	Jordyn.Jacobson2	2016-05-14 07:36:26
- Action Output:** Shows the history of actions taken on the session:
 - 22 13:13:57 SELECT * from ig_clone.users WHERE created_at
 - 23 13:18:04 SELECT * from ig_clone.users ORDER BY account_created_at ASC LIMIT 1
 - 24 13:18:15 SELECT * from ig_clone.users ORDER BY created_at ASC LIMIT 1
 - 25 13:19:20 SELECT * from ig_clone.users ORDER BY created_at ASC LIMIT 5

Answer: In this task we first use the `ig_clone` database's table `users` to identify the five oldest users using select query, and arrange the data according to the `created_at` (date of account creation) in ascending order. And limit the data up to 5 because we need the 5 oldest users.

2. **Inactive User Engagement:** Identify users who have never posted a single photo on Instagram.

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the following SQL code:


```

1 • SELECT * FROM users,photos
2 • select * from users u left join photos p
3   on p.user_id = u.id
4   where p.image_url is null
5   Order by u.username;
      
```
- Result Grid:** Displays the results of the query, showing 56 rows of user data where no photos have been posted:

	id	username	created_at	id	image_url	user_id	created_at
1	5	Anya.Hackett	2016-12-07 01:04:39	NULL	NULL	NULL	NULL
2	83	Bartholome.Bernhard	2016-11-06 02:31:23	NULL	NULL	NULL	NULL
3	91	Bethany20	2016-06-03 23:31:53	NULL	NULL	NULL	NULL
4	80	Darby_Herzog	2016-05-06 00:14:21	NULL	NULL	NULL	NULL
5	45	David.Osinski5	2017-02-05 21:23:37	NULL	NULL	NULL	NULL
6	54	Duanes60	2016-12-21 04:43:38	NULL	NULL	NULL	NULL
7	90	Esmeralda.Maz57	2017-01-14 17:52:27	NULL	NULL	NULL	NULL
8	61	Felicity.Muller3	2017-01-14 17:52:27	NULL	NULL	NULL	NULL
9	68	Franco_Kehrer64	2016-11-13 20:09:27	NULL	NULL	NULL	NULL
10	74	Hulda.Macajkovic	2017-01-25 17:17:28	NULL	NULL	NULL	NULL
11	14	Jaclyn81	2017-02-06 23:29:16	NULL	NULL	NULL	NULL
12	76	Janelle.Nikolaus1	2016-07-21 09:06:09	NULL	NULL	NULL	NULL
13	89	Jessica_Vest	2016-09-14 23:47:05	NULL	NULL	NULL	NULL
14	57	Julien_Schmidt	2017-02-02 23:12:48	NULL	NULL	NULL	NULL
15	7	Kassandra_Homenick	2016-12-12 06:50:08	NULL	NULL	NULL	NULL
16	75	Leslie57	2016-09-21 05:14:01	NULL	NULL	NULL	NULL
17	53	Linnea59	2017-02-07 07:49:34	NULL	NULL	NULL	NULL
18	24	Maxwell.Halvorson	2017-04-18 02:32:44	NULL	NULL	NULL	NULL
19	41	Mckenna17	2016-07-17 17:25:45	NULL	NULL	NULL	NULL
20	66	Mike.Auer39	2016-07-01 17:36:15	NULL	NULL	NULL	NULL
21	49	Morgan.Kassuke	2016-10-30 12:42:31	NULL	NULL	NULL	NULL
22	71	Na_Haag	2016-05-14 15:38:50	NULL	NULL	NULL	NULL
23	36	Ollie_Edward37	2016-08-04 15:42:20	NULL	NULL	NULL	NULL
- Action Output:** Shows the history of actions taken on the session:
 - Result 8: Action Output

Answer: In this, task we will need the users table and photos table from database lg_clone.

Using the select query we join the user (u) and Photos (p) and photo's user_id will be equal to user's id. Using WHERE clause we will set Image_url as null (to find who have zero post) and arrange in the order by username so we will get users name who have posted zero post.

3. Contest Winner Declaration: Determine the winner of the contest and provide their details to the team. (find the most liked single photo).

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is "lg_clone".
- Query Editor:** The query is:1 select * from Likes.photos;
2 select likes.photo_id,users.username, count(likes.user_id) as nooflikes
 from likes
 inner join photos on likes.photo_id = photos.id
 inner join users on photos.user_id = users.id
 group by likes.photo_id,users.username
 order by nooflikes desc;
- Result Grid:** The results show the number of likes for each photo, ordered by likes. The data is as follows:| photo_id | username | nooflikes |
| --- | --- | --- |
| 145 | Zack_Kemmer93 | 48 |
| 127 | Malina_Streich | 43 |
| 182 | Adelle96 | 43 |
| 123 | Seth46 | 42 |
| 30 | Prestley_McClure | 41 |
| 52 | Annalise_Mckenzie16 | 41 |
| 61 | Delpha_Kuhn | 41 |
| 147 | Meagan_Doyle | 41 |
- Action Output:** The log shows the execution of the query, including the creation of temporary tables and the final result.

Answer: In this, task we will need likes, users and photos table from database.

Using Select query we will fetch Photo_id from likes & username from user's table.
using Count clause we will count the user_id as no. of likes from likes table.
now, we will inner join likes, photos and users. where from Likes's photo_id is equal to photo's id and photo's user_id equal to user's id. Now we will group them by photo_id and username and arrange them in order by no of likes in descending. To get the most liked single photo.

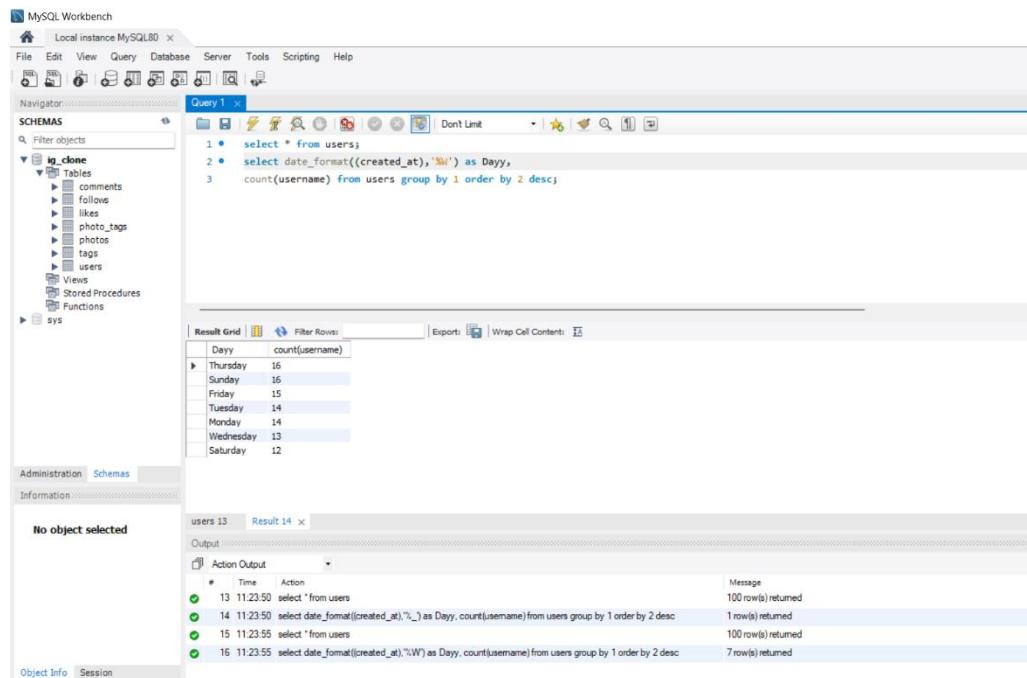
4. Hashtag Research: Identify and suggest the top five most commonly used hashtags on the platform.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is "lg_clone".
- Query Editor:** The query is:1 select * from photo_tags, tags;
2 select t.tag_name, count(p.photo_id) as ht
 from photo_tags p inner join tags t on t.id = p.tag_id
 group by t.tag_name order by ht desc
 limit 5;
- Result Grid:** The results show the top five hashtags and their counts:| tag_name | ht |
| --- | --- |
| smile | 59 |
| beach | 42 |
| party | 39 |
| fun | 38 |
| concert | 24 |
- Action Output:** The log shows the execution of the query, including the creation of temporary tables and the final result.

Answer: In this, task we will need photo_tags and tags table from the database. using the SELECT query we will get the tag_name from Tag table and count the photo_id from Photo_tags table as (ht) **Hastag**. From the **photo_tag** (p is a short name for photo_tag) and we will inner join **tags** (t is the short name for tags).
The id of (t) is equal to the tag_id of (p) and group them by tag_name from (t) and arrange them in order by (ht) **hashtag** in descending. Limit it up to 5 because we need only five tags.

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



The screenshot shows the MySQL Workbench interface. The 'Query 1' tab contains the following SQL code:

```

1 • select * from users;
2 • select date_format((created_at), "%W") as Dayy,
3   count(username) from users group by 1 order by 2 desc;

```

The 'Result Grid' shows the following data:

Dayy	count(username)
Thursday	16
Sunday	16
Friday	15
Tuesday	14
Monday	14
Wednesday	13
Saturday	12

The 'Output' section shows the history of actions:

#	Time	Action	Message
13	11:23:50	select * from users	100 row(s) returned
14	11:23:50	select date_format((created_at), "%_U") as Dayy, count(username) from users group by 1 order by 2 desc	1 row(s) returned
15	11:23:55	select * from users	100 row(s) returned
16	11:23:55	select date_format((created_at), "%W") as Dayy, count(username) from users group by 1 order by 2 desc	7 row(s) returned

Answer: In this, task we will need **USERS** table from the database. Using the Date_format (...,'%W') we Will convert the date of creation into day like (Monday, Tuesday...). And we will count the Usernames from the user table. Group all the data by the first column as **DAY** and arranging All the data order by username in Descending.

B) Investor Metrics:

1. **User Engagement:** 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.

```

MySQL Workbench
Local instance MySQL80 x
File Edit View Query Database Server Tools Scripting Help
Navigator: Schemas
SCHEMAS
  ▾ iq_clone
    ▾ Tables
      comments
      follows
      likes
      photo_tags
      photos
      tags
      users
      views
      stored Procedures
      Functions
  sys
Query 1 x
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
1 • select * from photos,users;
2 • with base as(
3   select u.id as userid,count(p.id) as photoid
4   from users u left join photos p on p.user_id = u.id group by u.id
5   select sum(photoid) as totalphotos,sum(userid) as total_users,sum(photoid)/count(userid) as photoperuser
6   from base;
7
8
Result 19 Result 20 x
No object selected
Output
Action Output
# Time Action
22 12:01:52 select * from photos,users
23 12:01:52 with base as( select u.id as userid,count(p.id) as photoid from users u left join photos p on p.user_id = u.id group by u.id )
24 12:03:31 select *from photos,users
25 12:03:31 with base as( select u.id as userid,count(p.id) as photoid from users u left join photos p on p.user_id = u.id group by u.id )
Object Info Session

```

The screenshot shows the MySQL Workbench interface with a query editor window. The query is as follows:

```

1 • select * from photos,users;
2 • with base as(
3   select u.id as userid,count(p.id) as photoid
4   from users u left join photos p on p.user_id = u.id group by u.id
5   select sum(photoid) as totalphotos,sum(userid) as total_users,sum(photoid)/count(userid) as photoperuser
6   from base;
7
8

```

The result grid displays the following data:

	totalphotos	total_users	photoperuser
	257	100	2.5700

Answer: In this, task we will need photos, users table from database. Then we will create an temporary table named as base. using select query **u.id** (user table's id column) renamed as userid and we will count the **p.id** (photo's id) and renamed as photoid from using **Left Join** we will join each user with there photos which will lead you to get every single users photos even with 0 photos and group them by user id. then we will sum all the **photo id** and named as total photos and count the user id and named as total users then we will divide the total photos with the counted users and named as photo per user from the base.

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

```

MySQL Workbench
Local instance MySQL80 x
File Edit View Query Database Server Tools Scripting Help
Navigator: Schemas
SCHEMAS
  ▾ iq_clone
    ▾ Tables
      comments
      follows
      likes
      photo_tags
      photos
      tags
      users
      views
      stored Procedures
      Functions
  sys
Query 1 x
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
1 • select * from users,likes;
2 • with base as(
3   select u.username,count(l.photo_id) as likess
4   from likes l inner join users u on u.id=user_id
5   group by username
6   select username,likess from base where likess=(select count(*) from photos)
7   order by username;
8
Result 22 Result 23 x
No object selected
Output

```

The screenshot shows the MySQL Workbench interface with a query editor window. The query is as follows:

```

1 • select * from users,likes;
2 • with base as(
3   select u.username,count(l.photo_id) as likess
4   from likes l inner join users u on u.id=user_id
5   group by username
6   select username,likess from base where likess=(select count(*) from photos)
7   order by username;
8

```

The result grid displays the following data:

username	likess
Annya_Haddett	257
Bethany20	257
Duane60	257
Jadyr81	257
Janelle_Nikolaus81	257
Julien_Schmidt	257
Leslie67	257
Maxwell_Halverson	257
Mckenna17	257
Mike_Auer39	257
Nia_Haag	257
Ollie_Ledner37	257
Rodo33	257

Answer: In this, task we will use user and likes table from database. With creating an temporary table called **base**, then we will get the username from **users** table and count the photo_id from table **likes** and name it as likess.

Later we will join the the id from user table and user_id from photos table. We will get the each likes done by a single user and group them by username and there likes.