

Instagram User Analytics

Project Aim

This project is based on the user interactions and engagement analytics using SQL as part of Instagram database with MySQL workbench. The goal is to provide insights so that they can use the given insights as part of their decision making. It is based on advanced analysis of loyal users, user activity and helps detect better marketing campaigning strategies. It will help answer specific business questions asked by client management team and enable their decision making process.

Approach

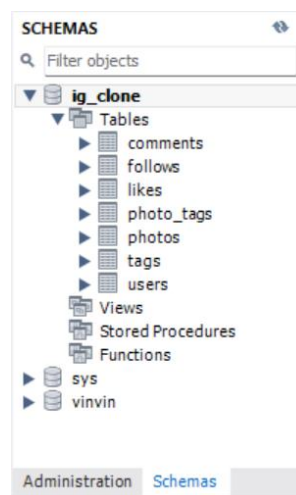
There were several steps to achieve this on the project:

Setting Up the Environment:

1. We installed and configured MySQL workbench to control the database.
2. The data was then loaded into tables (users, photos, comments, likes, follows, tags, photo_tags) by creating a schema to structure the data which in turn created using given SQL commands.

Data Ingestion:

INSERT statements were used to import the provided dataset into them. This consisted of making sure the data was correctly loaded and maintaining relationships between tables using foreign keys.



Tech-Stack Used

1. SQL (Structured Query Language): SQL as a language is used generally to interact with databases and perform analyses.
2. MySQL Workbench 8.0.38: MySQL Workbench has strong relational database management and querying capabilities. It offers a user-friendly interface for building schemas, running SQL queries, and effectively managing data.

Insights

A) Marketing Analysis:

1. Loyal User Reward:

The query gave the marketing team targets for rewards by figuring out who the five oldest users on the network were, which would increase user retention.

Code Used:

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

The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view with 'ig_clone' selected, containing tables like 'comments', 'follows', 'likes', 'photo_tags', 'photos', 'tags', and 'users'. The main editor shows a SQL query: `SELECT id, username, created_at FROM users ORDER BY created_at ASC LIMIT 5;`. Below the query, the 'Result Grid' shows the results of the query, displaying columns 'id', 'username', and 'created_at' for five users. The 'Output' pane at the bottom shows the execution log, including the query and the message '5 row(s) returned'.

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

Conclusion:

We got the result as:

	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. Inactive User Engagement:

The query discovered a subset of users who had never posted a photo, providing a chance for re-engagement via targeted promotions.

Code Used:

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

The screenshot displays a database management tool interface. On the left, a 'SCHEMAS' pane shows a tree view of the database structure, including tables like 'comments', 'follows', 'likes', 'photo_tags', 'photos', 'tags', 'users', 'views', 'stored_procedures', 'functions', 'sys', and 'vinvin'. The 'ig_clone' schema is selected. The main area shows a query editor with the following SQL code:

```
1 • SELECT u.id, u.username
2 FROM users u
3 LEFT JOIN photos p ON u.id = p.id
4 WHERE p.id IS NULL;
5
```

Below the query editor, a 'Result Grid' shows the results of the query. The grid has two columns: 'id' and 'username'. The results are as follows:

id	username
80	Darby_Herzog
67	Emilio_Bernier52
63	Elenor88
95	Nicole71
38	Jordyn.Jacobson2
NULL	NULL

At the bottom, an 'Output' pane shows the execution log. It includes a table with columns for '#', 'Time', 'Action', and 'Message'. The log shows two entries:

#	Time	Action	Message
61	19:53:02	SELECT u.id, u.username FROM users u LEFT JOIN image_url i ON u.id = i.id WH...	Error Code: 1146. Table 'ig_clone.image_url' doesn't exist
62	19:54:17	SELECT u.id, u.username FROM users u LEFT JOIN photos p ON u.id = p.id WHE...	0 row(s) returned

Conclusion:

We got the result as:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
id	username		

3. Contest Winner Declaration:

The query identified the user who received the most likes on a single photo, allowing the contest winner to be declared and providing important input for future engagement activities.

Code Used:

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

The screenshot shows a database management interface with a query editor and a results pane. The query editor contains the following SQL code:

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

The results pane displays a single row of data:

user_id	username	photo_id	like_count
52	Zack_Kemmer93	145	48

The interface also shows a schema tree on the left with tables like comments, follows, likes, photo_tags, photos, tags, and users. The bottom pane shows the execution log with the following entries:

#	Time	Action	Message
63	20:30:29	SELECT * FROM ig_clone.likes	8782 row(s) returned
64	20:40:35	SELECT p.user_id, u.username, p.id AS photo_id, COUNT(l.user_id) AS like_count ...	1 row(s) returned

Conclusion:

We get the result as:

	user_id	username	photo_id	like_count
▶	52	Zack_Kemmer93	145	48

4. Hashtag Research:

The top five most regularly used hashtags had been identified, providing strategic recommendations for the partner brand to maximize their exposure.

Code Used:

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

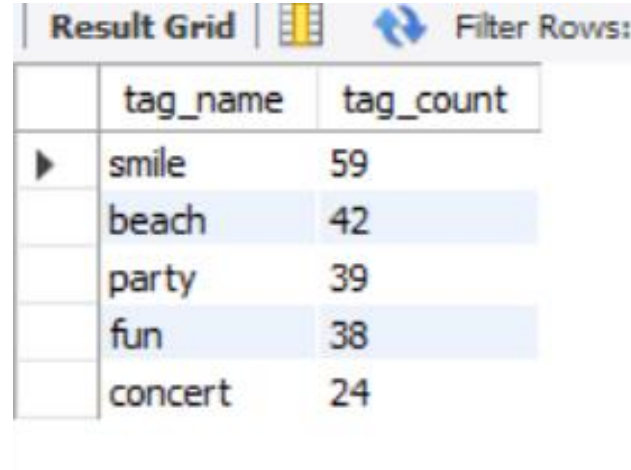
The screenshot displays a database management interface with a sidebar on the left showing a schema named 'ig_clone' containing tables like comments, follows, likes, photo_tags, photos, tags, and users. The main window shows a SQL query in 'Query 1' (mtask4) that selects the top 5 tags by count. The 'Result Grid' at the bottom shows the following data:

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

Below the result grid, the 'Output' section shows the execution log with two entries: one for a previous query (64) and one for the current query (65) which returned 5 rows.

Conclusion:

We get the result as:



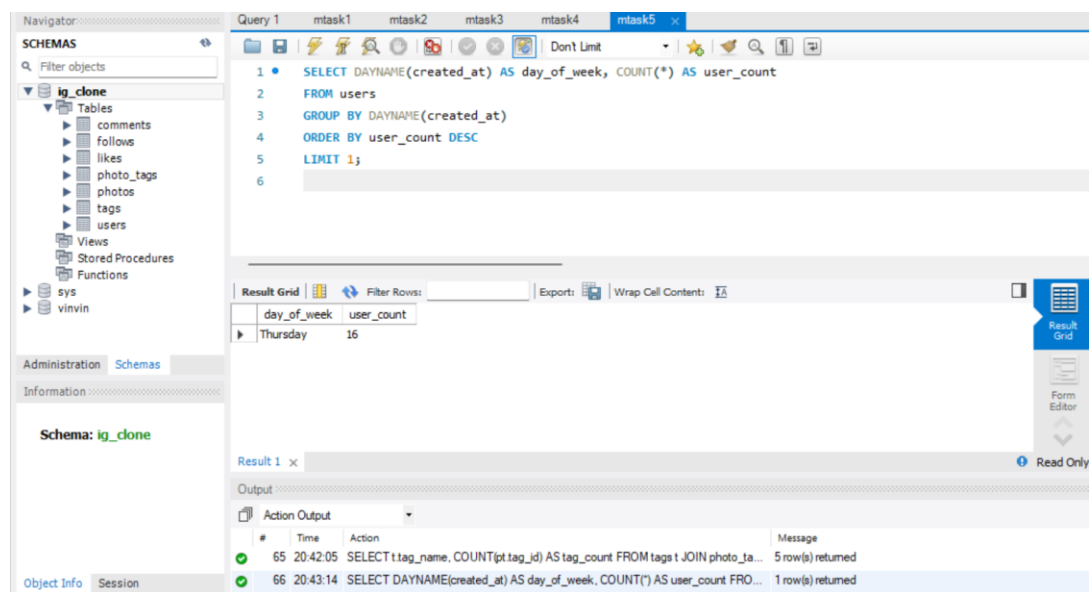
	tag_name	tag_count
▶	smile	59
	beach	42
	party	39
	fun	38
	concert	24

5. Ad Campaign launch:

By monitoring user registration statistics, we were able to predict the ideal day of the week to debut ads, providing insights for improving campaign performance.

Code Used:

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



The screenshot shows a database management tool interface. On the left is a 'Navigator' pane with a tree view of the 'ig_clone' schema, including tables like comments, follows, likes, photo_tags, photos, tags, and users. The main area displays 'Query 1' with the following SQL code:

```
1 • SELECT DAYNAME(created_at) AS day_of_week, COUNT(*) AS user_count
2 FROM users
3 GROUP BY DAYNAME(created_at)
4 ORDER BY user_count DESC
5 LIMIT 1;
6
```

Below the query editor is a 'Result Grid' showing the output of the query:

day_of_week	user_count
Thursday	16

At the bottom, the 'Output' pane shows a log of actions, including the execution of the query and the number of rows returned.

Conclusion:

We get the result as:

Result Grid		Filter Rows:
	day_of_week	user_count
▶	Thursday	16

B) Investor metrics:

1. User Engagement:

The query revealed the average amount of postings per user. This information is critical for assessing the general health of the platform and user satisfaction, as well as guiding future product development and marketing tactics.

Code Used:

```
SELECT AVG(post_count) AS average_posts_per_user
FROM (
  SELECT user_id, COUNT(*) AS post_count
  FROM photos
  GROUP BY user_id
) AS user_posts;
```

-- Total number of photos divided by the total number of users

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

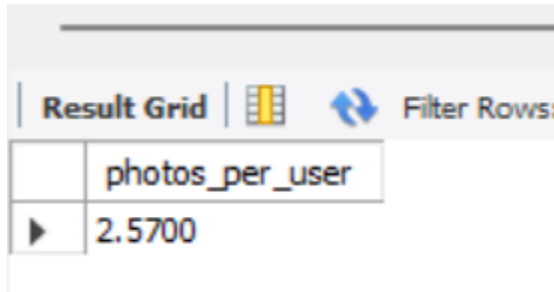
The screenshot displays a database management interface with a left-hand sidebar containing a 'SCHEMAS' tree and a bottom 'Information' panel. The main window shows a query editor with two SQL queries. The first query calculates the average number of posts per user, and the second query calculates the total number of photos divided by the total number of users. Below the editor, a 'Result Grid' shows the output of the first query, with a single row for 'photos_per_user' having a value of 2.5700. At the bottom, an 'Output' panel shows the execution log with two entries: one for the first query (line 67) and one for the second query (line 68), both indicating successful execution and the number of rows returned.

Result Grid	Filter Rows:	Exports:	Wrap Cell Content:
photos_per_user			
▶ 2.5700			

Result 1	Result 2
Output	
Action Output	
#	Time
67	20:44:33
SELECT AVG(post_count) AS average_posts_per_user FROM (SELECT user_id,... 1 row(s) returned	
68	20:44:33
SELECT (SELECT COUNT(*) FROM photos) / (SELECT COUNT(*) FROM users) A... 1 row(s) returned	

Conclusion:

We get the results as:



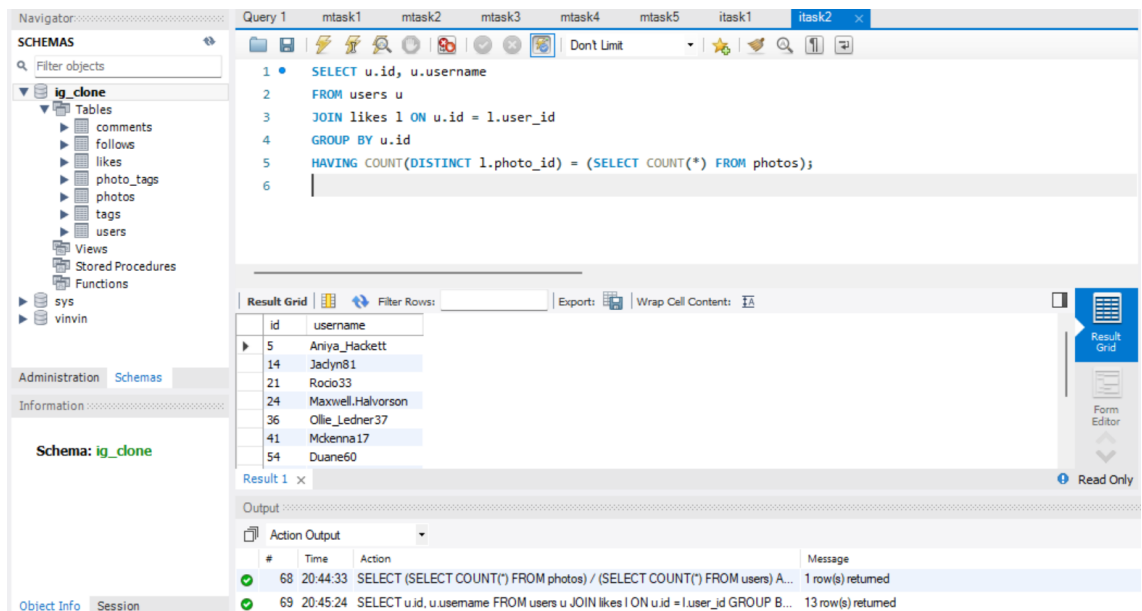
	photos_per_user
▶	2.5700

2. Bots and Fake Accounts:

The query identified suspected bot accounts, allowing the team to protect the platform's integrity through the detection of users with unusual engagement patterns.

Code Used:

```
SELECT u.id, u.username
FROM users u
JOIN likes l ON u.id = l.user_id
GROUP BY u.id
HAVING COUNT(DISTINCT l.photo_id) = (SELECT COUNT(*) FROM photos);
```



The screenshot shows a database management tool interface. On the left is a 'SCHEMAS' sidebar with a tree view of database objects including 'ig_clone' (Tables, Views, Stored Procedures, Functions) and 'sys'. The main area displays 'Query 1' with the following SQL code:

```
1 • SELECT u.id, u.username
2 FROM users u
3 JOIN likes l ON u.id = l.user_id
4 GROUP BY u.id
5 HAVING COUNT(DISTINCT l.photo_id) = (SELECT COUNT(*) FROM photos);
6
```

Below the query is a 'Result Grid' showing the results of the query. The grid has two columns: 'id' and 'username'. The results are as follows:

id	username
5	Aniya_Hackett
14	Jadyn81
21	Rocio33
24	Maxwell.Halvorson
36	Ollie_Ledner37
41	Mckenna17
54	Duane60

At the bottom, the 'Output' section shows the execution log. The first entry is:


```
68 20:44:33 SELECT (SELECT COUNT(*) FROM photos) / (SELECT COUNT(*) FROM users) A... 1 row(s) returned
```

The second entry is:

```
69 20:45:24 SELECT u.id, u.username FROM users u JOIN likes l ON u.id = l.user_id GROUP B... 13 row(s) returned
```


Conclusion:

We get the result as:

Result Grid				 Filter Rows:
	id	username		
▶	5	Aniya_Hackett		
	14	Jadyn81		
	21	Rocio33		
	24	Maxwell.Halvorson		
	36	Ollie_Ledner37		
	41	Mckenna17		
	54	Duane60		
	57	Julien_Schmidt		
	66	Mike.Auer39		
	71	Nia_Haag		
	75	Leslie67		
	76	Janelle.Nikolaus81		
	91	Bethany20		

Result

Overall, the project was effective in delivering several actionable items that could be immediately situated and implemented by the product and marketing teams at Instagram. The analysis also helps the platform boost repeat usage by developing reward activities for those users who were most consistent. The decision to analyze inactive users would allow for finding ways to re-activate said users and thus, improve activity rates on the site. Information on some of the trending hashtags in the market and best time to post the ad campaign was helpful for strategic marketing approach towards improving the reach and engagement. Furthermore, the identification of abnormal engagement patterns proved useful in identifying bots or fake accounts that compromised the integrity of the platform. In summary, the project provided a valuable service in providing an extensive insight into user behavior relevant to the future enhancement of features, marketing strategies and subsequent user management methods.