

Instagram User Analytics

Project Description: User Engagement and Interaction Analysis for Instagram

As a data analyst at Instagram, this project focuses on analysing user interactions and engagement with the Instagram app to generate actionable insights for business growth. By examining how users engage with key features of the app, the analysis provides valuable data that informs strategic decisions across multiple teams. These insights help the marketing team optimize campaigns, guide the product team in developing new features, and support the development team in enhancing the overall user experience. The ultimate goal is to leverage data to improve user satisfaction, drive engagement, and support the continuous growth of the platform

Approach:

Collaborated with the product team to define key business questions and metrics. Analysed Instagram user data to identify relationships and trends. Extracted and cleaned relevant user interaction and engagement data from internal databases. Conducted targeted analyses to provide actionable insights, including:

1. Identifying loyal users with long-term engagement.
2. Highlighting users with minimal interactions for re-engagement opportunities.
3. Recognizing contest winners based on the highest likes on a single photo.
4. Conducting hashtag research to identify the most commonly used hashtags.
5. Determining the optimal day for campaign launches.

Delivered findings to the product and marketing teams, enabling data-driven decision-making to improve user engagement and drive growth.

Tech-Stack Used:

MySQL Workbench

Version 8.0.31 build 2235049 CE (64 bits)



Insights:

1. **User Loyalty:** Identified the most loyal users with the longest app usage, providing insights into retention strategies.
2. **Inactive Users:** Flagged users with minimal or no activity, enabling re-engagement campaigns.
3. **Contest Winners:** Determined users with the highest likes on a single post, valuable for promoting contests and rewards programs.
4. **Popular Hashtags:** Revealed the most frequently used hashtags, aiding content creation and marketing strategies.
5. **Optimal Ad Launch Day:** Identified the best day of the week for ad campaigns, maximizing visibility and engagement.

User Engagement:

- Analysed posts per user and determined the average number of posts, providing a benchmark for user activity.

Bots and Fake Accounts:

- Detected patterns indicative of bot or fake accounts, enhancing data integrity and platform security.

These insights provided actionable strategies to improve user retention, engagement, and marketing effectiveness.

Achievements:

Through this project, I successfully delivered actionable insights that helped the product and marketing teams improve user retention, engagement, and campaign effectiveness.

This analysis enabled Instagram to make data-driven decisions, optimize user experience, and target growth opportunities effectively. Personally, the project enhanced my skills in user behaviour analysis, while reinforcing the importance of leveraging data to drive business outcomes.

Results:

A) Marketing Analysis:

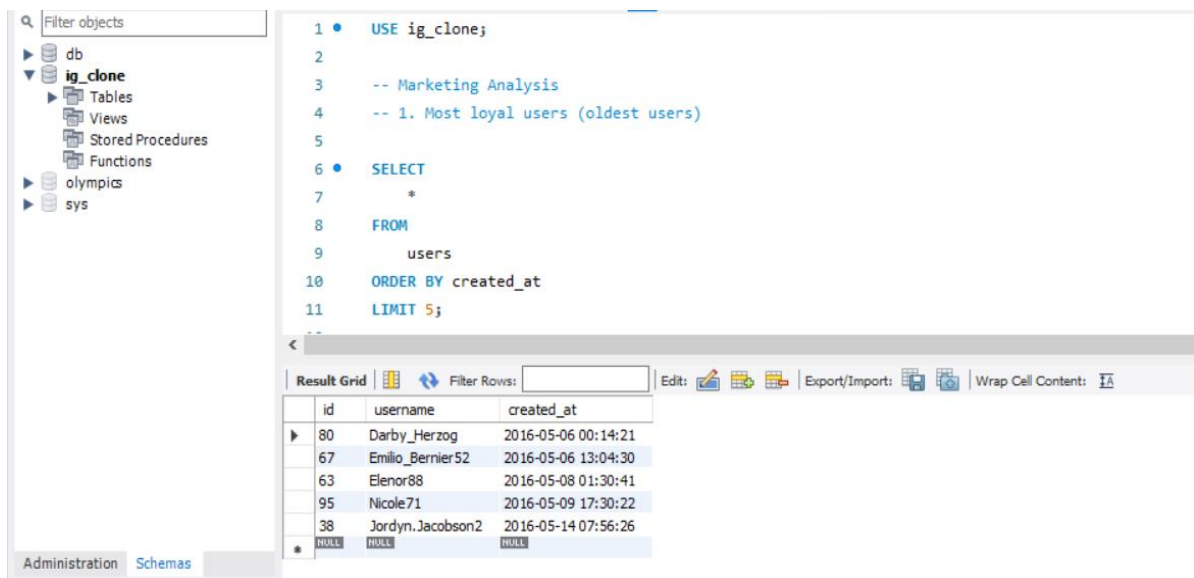
1. Loyal User Reward:

Users who have been using the platform for the longest time.

1. Darby_Herzog
2. Emilio_Bernier52
3. Elenor88
4. Nicole71
5. Jordyn.Jacobson2

SQL Query:

```
SELECT
    *
FROM
    users
ORDER BY created_at
LIMIT 5;
```



The screenshot shows a database management interface. On the left, a 'Filter objects' pane lists databases: db, ig_clone, olympics, and sys. The 'ig_clone' database is selected, showing its contents: Tables, Views, Stored Procedures, and Functions. The main area displays a SQL query in a text editor:

```
1 • USE ig_clone;
2
3 -- Marketing Analysis
4 -- 1. Most loyal users (oldest users)
5
6 • SELECT
7     *
8 FROM
9     users
10 ORDER BY created_at
11 LIMIT 5;
```

Below the query editor, the 'Result Grid' shows the execution results. It includes a toolbar with icons for Edit, Filter Rows, Export/Import, and Wrap Cell Content. The results are displayed in a table with columns 'id', 'username', and 'created_at'.

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

2. Inactive User Engagement

Users who have never posted a single photo on Instagram.

SQL Query:

```
SELECT
    users.id,
    users.username,
    COUNT(DISTINCT (photos.image_url)) AS num_of_photos
FROM
    photos
    RIGHT JOIN
    users ON users.id = photos.user_id
GROUP BY users.id
HAVING num_of_photos = 0;
```

The screenshot shows a database management interface. On the left, a sidebar displays a tree view of database objects, including 'db', 'ig_clone', 'olympics', and 'sys'. The 'ig_clone' database is selected. The main area displays the SQL query for finding inactive users. Below the query editor, a 'Result Grid' shows the output of the query, which is a table with three columns: 'id', 'username', and 'num_of_photos'. The table contains 15 rows of data, all with a 'num_of_photos' value of 0.

```
13  -- 2. Inactive user
14
15  *  SELECT
16      users.id,
17      users.username,
18      COUNT(DISTINCT (photos.image_url)) AS num_of_photos
19  FROM
20      photos
21      RIGHT JOIN
22      users ON users.id = photos.user_id
23  GROUP BY users.id
24  HAVING num_of_photos = 0;
```

id	username	num_of_photos
5	Aniya_Hackett	0
7	Kassandra_Homenick	0
14	Jadyn81	0
21	Rocio33	0
24	Maxwell.Halvorson	0
25	Tierra.Trantow	0
34	Pearl7	0
36	Ollie_Ledner37	0
41	Mckenna17	0
45	David.Osinski47	0
49	Morgan.Kassulke	0
53	Linnea59	0
54	Duane60	0
57	Julien_Schmidt	0

3. Contest Winner Declaration:

Winner of the contest - User with the most likes on a single photo

SQL Query:

```
SELECT * FROM (  
SELECT  
    users.id AS user_id,  
    users.username,  
    likes.photo_id,  
    COUNT(likes.photo_id) AS photo_likes  
FROM  
    photos  
    RIGHT JOIN  
    likes ON likes.photo_id = photos.id  
    LEFT JOIN  
    users ON users.id = photos.user_id  
GROUP BY photos.user_id , likes.photo_id  
ORDER BY photo_likes DESC  
    ) as contest_winner  
HAVING MAX(contest_winner.photo_likes);
```

The screenshot shows a database management tool interface. On the left, there is a sidebar with a search bar and a list of objects: db, ig_clone, olympics, and sys. The main area displays the SQL query for finding the contest winner. Below the query, there is a 'Result Grid' tab showing the results of the query. The results are as follows:

user_id	username	photo_id	photo_likes
52	Zack_Kemmer93	145	48

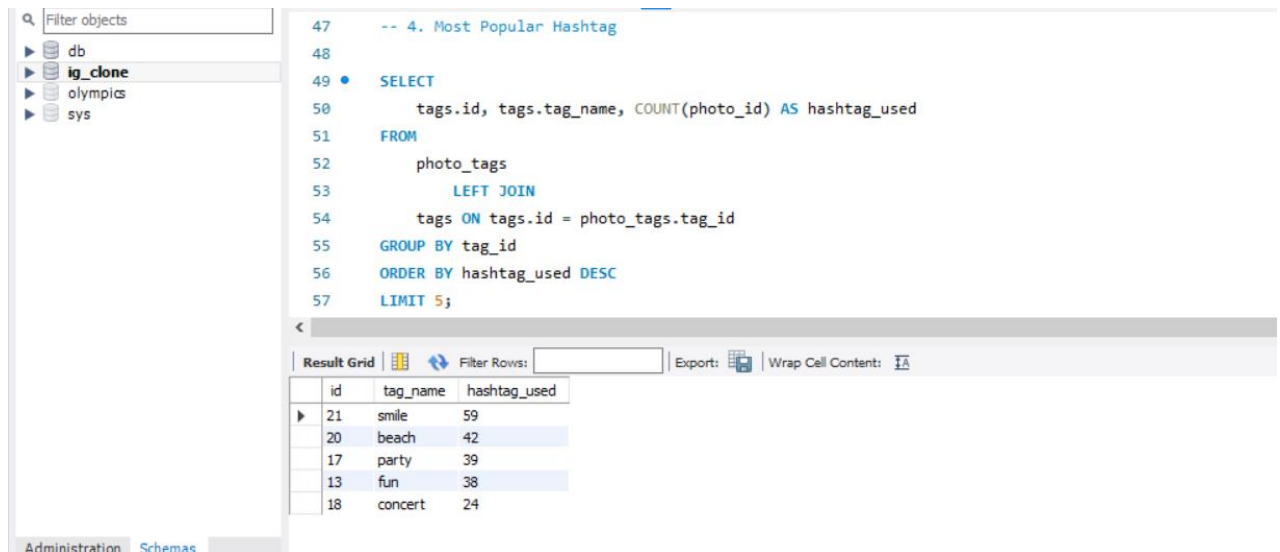
The schema is identified as 'ig_clone'.

4. Hashtag Research:

Most popular hashtags: smile, beach, party, fun, concert

SQL Query:

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



The screenshot shows a database management interface. On the left, a sidebar lists database objects: 'db', 'ig_clone', 'olympics', and 'sys'. The main area displays the SQL query from the previous block, with line numbers 47 to 57. Below the query, a 'Result Grid' shows the output of the query. The grid has three columns: 'id', 'tag_name', and 'hashtag_used'. The results are as follows:

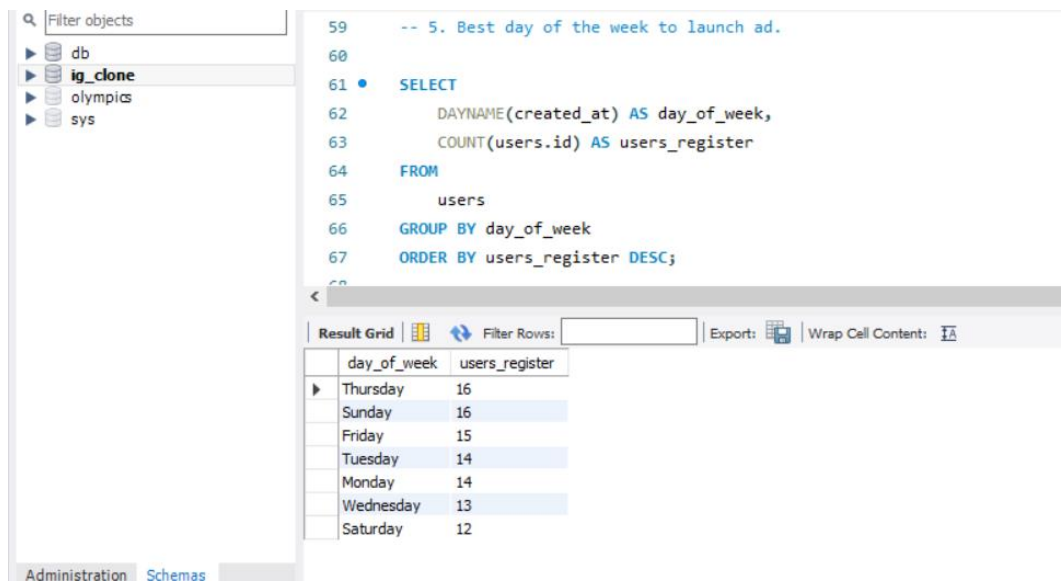
id	tag_name	hashtag_used
21	smile	59
20	beach	42
17	party	39
13	fun	38
18	concert	24

5. Ad Campaign Launch

The best day of the week to launch ads: Thursday and Sunday

SQL Query:

```
SELECT
    DAYNAME(created_at) AS day_of_week,
    COUNT(users.id) AS users_register
FROM
    users
GROUP BY day_of_week
ORDER BY users_register DESC;
```



The screenshot shows a SQL IDE interface. On the left, a 'Filter objects' pane lists databases: db, ig_clone, olympics, and sys. The main editor displays the SQL query from the previous block. Below the editor, a 'Result Grid' shows the output of the query. The grid has two columns: 'day_of_week' and 'users_register'. The results are ordered by 'users_register' in descending order.

day_of_week	users_register
Thursday	16
Sunday	16
Friday	15
Tuesday	14
Monday	14
Wednesday	13
Saturday	12

B) Investor Metrics:

1. User Engagement:

User posts fewer posts but they are active on Instagram

SQL Query:

- Posts per user

```
SELECT
    photos.user_id, users.username, COUNT(photos.id) AS posts
FROM
    photos
    LEFT JOIN
        users ON users.id = photos.user_id
GROUP BY photos.user_id
ORDER BY posts DESC;
```

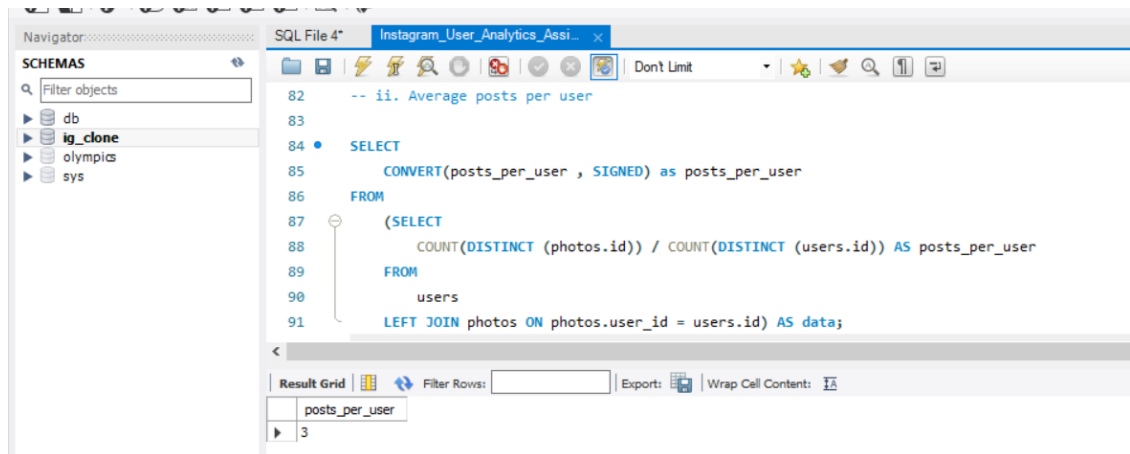
The screenshot shows a database management interface with a left sidebar containing a tree view of databases: 'db', 'ig_clone', 'olympics', and 'sys'. The 'ig_clone' database is selected. The main area displays a SQL query in a text editor, which is the same query as shown in the previous block. Below the query editor, there is a 'Result Grid' tab. The results are displayed in a table with three columns: 'user_id', 'username', and 'posts'. The table contains 15 rows of data, sorted by 'posts' in descending order. The first row shows user_id 23, username 'Eveline95', and 12 posts. The last row shows user_id 11, username 'Justina.Gaylord27', and 5 posts.

user_id	username	posts
23	Eveline95	12
88	Clint27	11
59	Cesar93	10
86	Delfina_VonRueden68	9
58	Aurelie71	8
29	Jaime53	8
77	Donald.Fritsch	6
33	Yvette.Gottlieb91	5
52	Zack_Kemmer93	5
47	Harrison.Beatty50	5
6	Travon.Waters	5
13	Alexandro35	5
51	Mariano_Koch3	5
78	Colten.Harris76	5
11	Justina.Gaylord27	5

- Average posts per user

SQL Query:

```
SELECT
    CONVERT(posts_per_user , SIGNED) as posts_per_user
FROM
    (SELECT
        COUNT(DISTINCT (photos.id)) / COUNT(DISTINCT (users.id)) AS posts_per_user
    FROM
        users
    LEFT JOIN photos ON photos.user_id = users.id) AS data;
```



2. Bots & Fake Accounts

There are 13 potential Bots and Fake accounts on the platform

SQL Query:

```
SELECT
    *
FROM
    (SELECT
        users.id AS user_id,
        users.username,
        COUNT(DISTINCT (likes.photo_id)) AS likes_by_user
    FROM
        users
    RIGHT JOIN likes ON likes.user_id = users.id
    GROUP BY users.id , users.username
    ORDER BY likes_by_user DESC) AS bot_detector
HAVING likes_by_user = (SELECT
    COUNT(photos.id)
FROM
    photos)
ORDER BY bot_detector.user_id;
```

Filter objects

db

ig_clone

olympics

sys

Administration

Schemas

Information

Schema: ig_clone

```

93  -- 2. Bots or Fake accounts
94
95  • SELECT
96      *
97  FROM
98      (
99          (SELECT
100              users.id AS user_id,
101              users.username,
102              COUNT(DISTINCT (likes.photo_id)) AS likes_by_user
103          FROM
104              users
105          RIGHT JOIN likes ON likes.user_id = users.id
106          GROUP BY users.id , users.username
107          ORDER BY likes_by_user DESC) AS bot_detector
108      HAVING likes_by_user = (SELECT
109          COUNT(photos.id)
110          FROM
111              photos)
112      ORDER BY bot_detector.user_id;

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	user_id	username	likes_by_user
▶	5	Aniya_Hackett	257
	14	Jadyn81	257
	21	Rocio33	257
	24	Maxwell_Halvorson	257
	36	Ollie_Ledner37	257
	41	Melissa17	257