

# PROJECT 2: INSTAGRAM USER ANALYTICS

## Introduction

This project aims to extract useful insight from raw data using MYSQL workbench and the findings could potentially influence the future development of platforms and thereby increase your knowledge in MYSQL.

## 1) Marketing Analysis

### 1. Loyal User Reward:

Task: Identify the five oldest users on Instagram from the provided data base.

Code:           SELECT \*  
                  FROM users  
  
                  ORDER BY created\_at  
  
                  LIMIT 5;

**Result:** The five oldest users selected for loyal user Reward are follow;

Sl no	# id	username	created_at
1)	80	Darby_Herzog	2016-05-06 00:14:21
2)	67	Emilio_Bernier52	2016-05-06 13:04:30
3)	63	Elenor88	2016-05-08 01:30:41
4)	95	Nicole71	2016-05-09 17:30:22
5)	38	Jordyn.Jacobson2	2016-05-14 07:56:26

The screenshot displays the MySQL Workbench interface. The top pane shows the SQL query: `use ig_clone; SELECT * FROM users ORDER BY created_at LIMIT 5;`. The bottom pane shows the 'Result Grid' with the following data:

	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:

Task: Identify users who have never posted a single photo on Instagram

```
Code: SELECT *  
      FROM users  
      LEFT JOIN photos  
      ON users.id = photo.user_id  
      WHERE photos.user_id is null;
```

**Result:** There are 26 users who have never posted a single photo on Instagram.

sno	# id	username	created_at	image_url
1.	5	Aniya_Hackett	2016-12-07 01:04:39	Null
2.	7	Kasandra_Homenick	2016-12-12 06:50:08	Null
3.	14	Jaclyn81	2017-02-06 23:29:16	Null
4.	21	Rocio33	2017-01-23 11:51:15	Null
5.	24	Maxwell.Halvorson	2017-04-18 02:32:44	Null
6.	25	Tierra.Trantow	2016-10-03 12:49:21	Null
7.	34	Pearl7	2016-07-08 21:42:01	Null
8.	36	Ollie_Ledner37	2016-08-04 15:42:20	Null
9.	41	Mckenna17	2016-07-17 17:25:45	Null
10.	45	David.Osinski47	2017-02-05 21:23:37	Null
11.	49	Morgan.Kassulke	2016-10-30 12:42:31	Null
12.	53	Linnea59	2017-02-07 07:49:34	Null
13.	54	Duane60	2016-12-21 04:43:38	Null
14.	57	Julien_Schmidt	2017-02-02 23:12:48	Null
15.	66	Mike.Auer39	2016-07-01 17:36:15	Null
16.	68	Franco_Keebler64	2016-11-13 20:09:27	Null
17.	71	Nia_Haag	2016-05-14 15:38:50	Null
18.	74	Hulda.Macejkovic	2017-01-25 17:17:28	Null
19.	75	Leslie67	2016-09-21 05:14:01	Null
20.	76	Janelle.Nikolaus81	2016-07-21 09:26:09	Null
21.	80	Darby_Herzog	2016-05-06 00:14:21	Null
22.	81	Esther.Zulauf61	2017-01-14 17:02:34	Null
23.	83	Bartholome.Bernhard	2016-11-06 02:31:23	Null
24.	89	Jessyca_West	2016-09-14 23:47:05	Null
25.	90	Esmeralda.Mraz57	2017-03-03 11:52:27	Null
26.	91	Bethany20	2016-06-03 23:31:53	Null

SQL File 5\*

users

comments

comments

photos

Limit to 1000 rows

```

10
11  # Task 2: Identify users who have never posted a single photo on Instagram.
12
13 • select *
14   from users
15  left join photos
16  on users.id = photos.user_id where photos.user_id is Null;
17

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	id	username	created_at	id	image_url	user_id	created_dat
▶	5	Aniya_Hackett	2016-12-07 01:04:39	NULL	NULL	NULL	NULL
	7	Kasandra_Homenick	2016-12-12 06:50:08	NULL	NULL	NULL	NULL
	14	Jadyn81	2017-02-06 23:29:16	NULL	NULL	NULL	NULL
	21	Rocio33	2017-01-23 11:51:15	NULL	NULL	NULL	NULL
	24	Maxwell.Halvorson	2017-04-18 02:32:44	NULL	NULL	NULL	NULL
	25	Tierra.Trantow	2016-10-03 12:49:21	NULL	NULL	NULL	NULL
	34	Pearl7	2016-07-08 21:42:01	NULL	NULL	NULL	NULL
	36	Ollie_Ledner37	2016-08-04 15:42:20	NULL	NULL	NULL	NULL
	41	Mckenna17	2016-07-17 17:25:45	NULL	NULL	NULL	NULL
	45	David.Osinski47	2017-02-05 21:23:37	NULL	NULL	NULL	NULL
	54	Duane60	2016-12-21 04:43:38	NULL	NULL	NULL	NULL
	57	Julien_Schmidt	2017-02-02 23:12:48	NULL	NULL	NULL	NULL
	66	Mike.Auer39	2016-07-01 17:36:15	NULL	NULL	NULL	NULL
	68	Franco_Keebler64	2016-11-13 20:09:27	NULL	NULL	NULL	NULL
	71	Nia_Haag	2016-05-14 15:38:50	NULL	NULL	NULL	NULL
	74	Hulda.Macejkovic	2017-01-25 17:17:28	NULL	NULL	NULL	NULL
	75	Leslie67	2016-09-21 05:14:01	NULL	NULL	NULL	NULL
	76	Janelle.Nikolaus81	2016-07-21 09:26:09	NULL	NULL	NULL	NULL
	80	Darby_Herzog	2016-05-06 00:14:21	NULL	NULL	NULL	NULL
	81	Esther.Zulauf61	2017-01-14 17:02:34	NULL	NULL	NULL	NULL
	83	Bartholome.Bernhard	2016-11-06 02:31:23	NULL	NULL	NULL	NULL
	89	Jessyca_West	2016-09-14 23:47:05	NULL	NULL	NULL	NULL
	90	Esmeralda.Mraz57	2017-03-03 11:52:27	NULL	NULL	NULL	NULL
	91	Bethany20	2016-06-03 23:31:53	NULL	NULL	NULL	NULL

Result 18

Output

Action Output

### 3 Contest Winner Declaration:

Task: Determine the winner of the contest and provide their details to the team.

Code: SELECT

```
    users.username,  
    photos.id,  
    photos.image_url,  
    count(likes.user_id) AS total_likes  
FROM  
    photos  
    INNER JOIN  
    likes ON likes.photo_id = photos.id  
    INNER JOIN  
    users ON photos.user_id = users.id  
GROUP BY photos.id  
ORDER BY total_likes DESC  
LIMIT 1;
```

**Result:** Details of the contest winner listed below,

**username :** Zack\_Kemmer93  
**id :** 145  
**image\_url :** <https://jarret.name>  
**total\_likes :** 48

The screenshot shows a SQL IDE interface with a query editor and a results panel. The query editor contains the following SQL code:

```
# 3 contest winner declaration: Determine the winner of the contest and provide their details to the team  
SELECT  
    users.username,  
    photos.id,  
    photos.image_url,  
    COUNT(likes.user_id) AS total_likes  
FROM  
    photos  
    INNER JOIN  
    likes ON likes.photo_id = photos.id  
    INNER JOIN  
    users ON photos.user_id = users.id  
GROUP BY photos.id  
ORDER BY total_likes DESC  
LIMIT 1;
```

The results panel displays a single row of data:

username	id	image_url	total_likes
Zack_Kemmer93	145	<a href="https://jarret.name">https://jarret.name</a>	48

The output panel shows the execution details:

#	Time	Action	Message
1	18:57:14	SELECT	users.username, photos.id, photos.image_url, COUNT(likes.user_id) AS total_likes FROM ... 1 row(s) returned

## 4 Hashtag Research

Task: Identify and suggest the top five most commonly used hashtags on the platform.

Code:           SELECT  
                  tags.tag\_name, count(\*) AS total\_tags  
FROM  
          photo\_tags  
          JOIN  
          tags  
ON photo\_tags.tag\_id = tags.id  
GROUP BY tags.id  
ORDER BY total\_tags DESC  
LIMIT 5;

Result: the top five most commonly used hashtags on the platform as follows;

sno	tag_name	total_tags
1)	smile	59
2)	beach	42
3)	party	39
4)	fun	38
5)	concert	24

The screenshot displays a SQL IDE interface. The top toolbar includes icons for file operations, execution, and settings. The SQL editor contains the following query:

```
41 # 4 hashtag Research: identify and suggest the top five most commonly used hashtags on the platform
42
43 • SELECT
44     tags.tag_name, COUNT(*) AS total_tags
45 FROM
46     photo_tags
47     JOIN
48     tags ON photo_tags.tag_id = tags.id
49 GROUP BY tags.id
50 ORDER BY total_tags DESC
51 LIMIT 5;
52
```

Below the editor, the 'Result Grid' shows the output of the query:

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

The 'Output' pane at the bottom shows a log of database actions:

#	Time	Action	Message
✓ 4	19:28:20	SELECT * FROM ig_clone.photo_tags LIMIT 0, 1000	501 row(s) returned
✓ 5	19:28:26	SELECT * FROM ig_clone.follows LIMIT 0, 1000	1000 row(s) returned
✓ 6	19:28:33	SELECT * FROM ig_clone.comments LIMIT 0, 1000	1000 row(s) returned
✓ 7	19:29:50	SELECT * FROM ig_clone.photo_tags LIMIT 0, 1000	501 row(s) returned
✓ 8	20:00:23	SELECT tags.tag_name, count(*) AS popular_tags from photo_tags join tags on photo_tags.tag_id = tags.id Gr...	5 row(s) returned

## 5 AD Campaign Launch

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

Code:           SELECT  
                  dayname(created\_at) AS day,  
                  count(\*) AS total\_reg  
FROM  
                  users  
GROUP BY day  
ORDER BY total\_reg DESC  
LIMIT 1;

**Result:** the schedule day for launch an ad campaign on Instagram is Thursday.

sno	day	total_reg
1	Thursday	16

The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

```
52
53 # 5 Ad Campaign Launch: day which most users register on Instagram
54
55 • SELECT
56     DAYNAME(created_at) AS day, COUNT(*) AS total_reg
57 FROM
58     users
59 GROUP BY day
60 ORDER BY total_reg DESC
61 LIMIT 1;
62
```

The result grid displays the following data:

day	total_reg
Thursday	16

Below the result grid, the 'Output' tab is selected, showing the execution log:

#	Time	Action	Message
1	20:50:38	SELECT DAYNAME(created_at) AS day, COUNT(*) AS total_reg FROM users GROUP BY day ORDER B...	1 row(s) returned
2	20:52:01	SELECT DAYNAME(created_at) AS day, COUNT(*) AS total_reg FROM users GROUP BY day ORDER B...	1 row(s) returned

## B INVESTOR METRICS

### 1 User Engagement:

**Task:** 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.

Code:

```
SELECT (
  SELECT
    count(*)
  FROM
    photos)
/ (
  SELECT
    count(*)
  FROM
    users) AS avg;
```

**Result :** Average number of posts per user on Instagram = 2.5700

The screenshot shows a SQL IDE interface with a query editor and a results panel. The query editor contains the following SQL code:

```
64
65 # INVESTOR METRICS
66 # 1 user engagement: calculate the average number of posts per user on instagram.
67
68 • SELECT
69   (SELECT
70     COUNT(*)
71   FROM
72     photos)
73   /
74   (SELECT
75     COUNT(*)
76   FROM
77     users) AS avg;
78
```

The results panel shows a single row with the value 2.5700.

avg
2.5700

The bottom panel shows the output of the query, indicating that 1 row(s) returned.

#	Time	Action	Message
1	13:18:17	SELECT (SELECT COUNT(*) FROM photos) / (SELECT COUNT(*) FROM users) AS avg;	1 row(s) returned

## 2 Bots & Fake Accounts:

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

### CODE: Query 1:

```
SELECT
    user_id, count(*) AS tot_num_likes
FROM
    likes
GROUP BY user_id
HAVING tot_num_likes
    =
    (SELECT
        count(*)
    FROM
        photos);
```

**Result:** Fake ids or potential bots are follows:

SLNO	user_id	tot_num_likes
1.	5	257
2.	14	257
3.	21	257
4.	24	257
5.	36	257
6.	41	257
7.	54	257
8.	57	257
9.	66	257
10.	71	257
11.	75	257
12.	76	257
13.	91	257



**CODE: Query 2:**

```
SELECT
    users.username, count(*) AS tot_num_likes
FROM
    users
JOIN
    likes ON users.id = likes.user_id
GROUP BY users.id
HAVING tot_num_likes
    =
    (SELECT
        count(*)
    FROM
        photos);
```

**Result :** Following are the fake users or potential bots in Instagram.

SLNO	username	tot_num_likes
1.	Aniya_Hackett	257
2.	Jaclyn81	257
3.	Rocio33	257
4.	Maxwell.Halvorson	257
5.	Ollie_Ledner37	257
6.	Mckenna17	257
7.	Duane60	257
8.	Julien_Schmidt	257
9.	Mike.Auer39	257
10.	Nia_Haag	257
11.	Leslie67	257
12.	Janelle.Nikolaus81	257
13.	Bethany20	257

project 2 insta analysis\* x photos users likes

Limit to 1000 rows

```

79
80 # 2b Bots & Fake Accounts: identify users(potential bots) who have liked every single photo on the site
81
82 • SELECT
83     user_id, COUNT(*) AS tot_num_likes
84 FROM
85     likes
86 GROUP BY user_id
87 HAVING tot_num_likes = (SELECT
88     COUNT(*)
89 FROM
90     photos);
91
92 • SELECT
93     users.username, COUNT(*) AS tot_num_likes
94 FROM
95     users
96 JOIN
97     likes ON users.id = likes.user_id
98 GROUP BY users.id
99 HAVING tot_num_likes = (SELECT
100     COUNT(*)
101 FROM
102     photos);
103

```

Result Grid

	username	tot_num_likes
▶	Aniya_Hackett	257
	Jadyn81	257
	Rocio33	257
	Maxwell.Halvorson	257
	Ollie_Ledner37	257
	Mckenna17	257
	Duane60	257
	Julien_Schmidt	257
	Mike.Auer39	257
	Nia_Haag	257
	Leslie67	257
	Janelle.Nikolaus81	257
	Bethany20	257

Result 24 Result 25 x

Output

Action Output

#	Time	Action	Message
✓ 1	16:14:04	SELECT user_id, COUNT(*) AS tot_num_likes FROM likes GROUP BY user_id HAVING tot_num_likes = (...	13 row(s) returned
✓ 2	16:14:04	SELECT users.username, COUNT(*) AS tot_num_likes FROM users JOIN likes ON users.id = likes....	13 row(s) returned

## Conclusion

This project successfully utilized SQL to uncover valuable insight about the platform. This project help to improve the experience with MYSQL, that will help me in future development of my career.