

## Data Analytics Project: Instagram User Analytics

### Project Description:

Instagram User Analytics allows to track how users engage and interact with Instagram. This helps in deriving business insights for marketing, product & development teams. This project insights are to be used by the management team, marketing team and investors.

### Approach:

Based on the provided dataset, and the questions asked by the management team, firstly, the tables were studied, to understand the relations between each table. Multiple tables were linked and used. Also, multiple queries were run to get the necessary insights from the data.

### Tech-Stack Used:

MySQL Workbench 8.0 CE was used to execute this project. As it provided enough useful set of functions. Also, the connectivity, speed and security of MySQL was suitable for accessing given dataset and run the queries.

### Insights:

Analysis Performed:

#### A) Marketing:

##### 1. Rewarding Most Loyal Users:

- For this task, we first found the active users, who have been using Instagram for a long time
- Below query was used to find the oldest users who have liked, commented on the posts at least once

Query:

Use ig\_clone;

create table ig\_clone.active\_users

SELECT u.id, u.username, u.created\_at "user\_joining\_date", l.photo\_id "liked\_photo",  
l.created\_at "liked\_at",

c.photo\_id "commented\_photo", c.created\_at "commented\_at"

FROM ig\_clone.users u, ig\_clone.likes l, ig\_clone.comments c

where u.id=l.user\_id and u.id=c.user\_id

order by u.created\_at asc, l.created\_at desc, c.created\_at desc;

The output:

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Fetch rows:

	id	username	user_joining_date	liked_photo	liked_at	commented_photo	commented_at
▶	67	Emilio_Bernier52	2016-05-06 13:04:30	216	2023-05-19 14:24:13	247	2023-05-19 14:24:12
	67	Emilio_Bernier52	2016-05-06 13:04:30	252	2023-05-19 14:24:13	247	2023-05-19 14:24:12
	67	Emilio_Bernier52	2016-05-06 13:04:30	250	2023-05-19 14:24:13	247	2023-05-19 14:24:12
	67	Emilio_Bernier52	2016-05-06 13:04:30	248	2023-05-19 14:24:13	247	2023-05-19 14:24:12
	67	Emilio_Bernier52	2016-05-06 13:04:30	244	2023-05-19 14:24:13	247	2023-05-19 14:24:12
	67	Emilio_Bernier52	2016-05-06 13:04:30	243	2023-05-19 14:24:13	247	2023-05-19 14:24:12
	67	Emilio_Bernier52	2016-05-06 13:04:30	242	2023-05-19 14:24:13	247	2023-05-19 14:24:12
	67	Emilio_Bernier52	2016-05-06 13:04:30	241	2023-05-19 14:24:13	247	2023-05-19 14:24:12
	67	Emilio_Bernier52	2016-05-06 13:04:30	234	2023-05-19 14:24:13	247	2023-05-19 14:24:12
	67	Emilio_Bernier52	2016-05-06 13:04:30	227	2023-05-19 14:24:13	247	2023-05-19 14:24:12
	67	Emilio_Bernier52	2016-05-06 13:04:30	224	2023-05-19 14:24:13	247	2023-05-19 14:24:12
	67	Emilio_Bernier52	2016-05-06 13:04:30	221	2023-05-19 14:24:13	247	2023-05-19 14:24:12

Result 2

- From this table of active users, we found the 5 oldest users, using below query:

Query:

```
SELECT distinct id, username, user_joining_date, liked_at, commented_at
```

```
FROM ig_clone.active_users
```

```
limit 5;
```

The final output:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
id	username	user_joining_date	liked_at	commented_at
67	Emilio_Bernier52	2016-05-06 13:04:30	2023-05-19 14:24:13	2023-05-19 14:24:12
63	Elenor88	2016-05-08 01:30:41	2023-05-19 14:24:13	2023-05-19 14:24:12
95	Nicole71	2016-05-09 17:30:22	2023-05-19 14:24:13	2023-05-19 14:24:12
38	Jordyn.Jacobson2	2016-05-14 07:56:26	2023-05-19 14:24:13	2023-05-19 14:24:12
71	Nia_Haag	2016-05-14 15:38:50	2023-05-19 14:24:13	2023-05-19 14:24:12

Final result: User ID: 67,63,95,38,71 are the 5 oldest and active users.

## 2. Remind Inactive Users to Start Posting:

For this we found the users who have never posted a single photo, using the query given below.

Query:

```
SELECT u.id "user_id", u.username, u.created_at "user_joining_date"
FROM ig_clone.users u
where u.id not in (select p.user_id "active_id" from ig_clone.photos p);
```

The output:

Result Grid	Filter Rows:	Exp
user_id	username	user_joining_date
5	Aniya_Hackett	2016-12-07 01:04:39
7	Kasandra_Homenick	2016-12-12 06:50:08
14	Jadyn81	2017-02-06 23:29:16
21	Rocio33	2017-01-23 11:51:15
24	Maxwell.Halvorson	2017-04-18 02:32:44
25	Tierra.Trantow	2016-10-03 12:49:21
34	Pearl7	2016-07-08 21:42:01
36	Ollie_Ledner37	2016-08-04 15:42:20
41	Mckenna17	2016-07-17 17:25:45
45	David.Osinski47	2017-02-05 21:23:37
49	Morgan.Kassulke	2016-10-30 12:42:31
53	Linnea59	2017-02-07 07:49:34
54	Duane60	2016-12-21 04:43:38
57	Julien_Schmidt	2017-02-02 23:12:48
66	Mike.Auer39	2016-07-01 17:36:15
68	Franco_Keebler64	2016-11-13 20:09:27
71	Nia_Haag	2016-05-14 15:38:50

The Final Result: 26 users have never posted a single photo

### 3. Declaring Contest Winner:

The team started a contest and the user who gets the most likes on a single photo will win the contest now they wish to declare the winner.

For this task, following query was used:

Query:

```
use ig_clone;
```

```
SELECT u.id "User ID", u.username, p.id "Photo ID", p.image_url, count(*) as likes
```

```
from ig_clone.photos p
```

```
inner join ig_clone.likes l
```

```
on p.id=l.photo_id
```

```
inner join ig_clone.users u on u.id=p.user_id
```

```
group by l.photo_id
```

```
order by likes desc
```

```
limit 1;
```

The output:

Result Grid						Filter Rows:	Export:	Wri
	User ID	username	Photo ID	image_url	likes			
►	52	Zack_Kemmer93	145	https://jarret.name	48			

The Final Result: User Zack\_Kemmer93 is the winner of the contest with 48 likes

4. Hashtag Researching: A partner brand wants to know, which hashtags to use in the post to reach the most people on the platform.

For this task, we identified the top 5 most commonly used hashtags on the platform by using the query below:

Query:

```
SELECT distinct t.id "Tag_ID", t.tag_name, count(p.photo_id) as totalcount
FROM ig_clone.tags t, ig_clone.photo_tags p
where t.id=p.tag_id
group by tag_id
order by count(photo_id) desc
limit 5;
```

The output:



The screenshot shows a database interface with a 'Result Grid' tab selected. The grid displays the results of the SQL query, showing the top 5 most commonly used hashtags. The columns are 'Tag\_ID', 'tag\_name', and 'totalcount'. The rows are ordered by 'totalcount' in descending order.

	Tag_ID	tag_name	totalcount
▶	21	smile	59
	20	beach	42
	17	party	39
	13	fun	38
	5	food	24

The Final Result: These 5 tags shown in the image above are the most commonly used hashtags

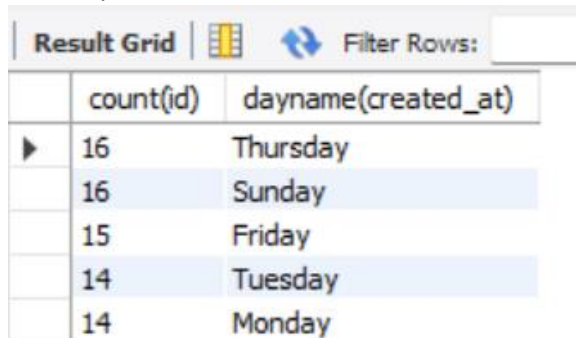
5. Launch AD Campaign: The team wants to know, which day would be the best day to launch ADs.

For this task, we found the day of the week most users registered on by running the query below

Query:

```
SELECT count(id),dayname(created_at) from ig_clone.users  
group by dayname(created_at)  
order by count(id) desc  
limit 5;
```

The output:



The screenshot shows a database interface with a 'Result Grid' tab. It displays the results of a SQL query. The table has two columns: 'count(id)' and 'dayname(created\_at)'. The data is sorted in descending order of the count. The first two rows, Thursday and Sunday, both have a count of 16. The next row is Friday with a count of 15. The last two rows are Tuesday and Monday, both with a count of 14. The rows are alternatingly highlighted in white and light blue.

	count(id)	dayname(created_at)
▶	16	Thursday
	16	Sunday
	15	Friday
	14	Tuesday
	14	Monday

The Final Result: Based on the data shown in the image above, Thursday and Sunday are the days of the week when most users registered, so the ad campaign should be scheduled for these 2 days to attract a greater number of users

## B. Investor Metrics:

To assess the performance and redundancy of the app, following tasks are performed:

### 1. User Engagement:

To know how many times does average user posts on Instagram, we first get the count number of every user's post, using the query below:

Query:

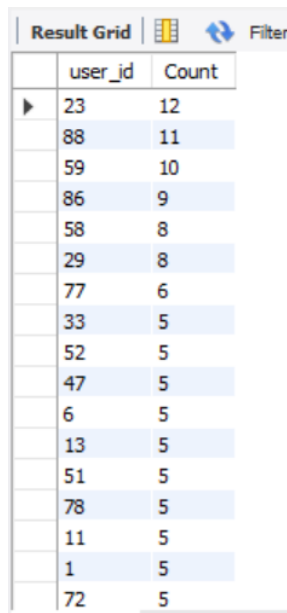
```
create table ig_clone.user_post
```

```
SELECT user_id, count(distinct id) "Count" from ig_clone.photos
```

```
group by user_id
```

```
order by count(distinct id) desc;
```

The output:



The screenshot shows a 'Result Grid' window with a table containing two columns: 'user\_id' and 'Count'. The table lists 20 rows of data, sorted by 'Count' in descending order. The first row has user\_id 23 and Count 12. The second row has user\_id 88 and Count 11. The third row has user\_id 59 and Count 10. The fourth row has user\_id 86 and Count 9. The fifth row has user\_id 58 and Count 8. The sixth row has user\_id 29 and Count 8. The seventh row has user\_id 77 and Count 6. The eighth row has user\_id 33 and Count 5. The ninth row has user\_id 52 and Count 5. The tenth row has user\_id 47 and Count 5. The eleventh row has user\_id 6 and Count 5. The twelfth row has user\_id 13 and Count 5. The thirteenth row has user\_id 51 and Count 5. The fourteenth row has user\_id 78 and Count 5. The fifteenth row has user\_id 11 and Count 5. The sixteenth row has user\_id 1 and Count 5. The seventeenth row has user\_id 72 and Count 5. The table is displayed in a light blue and white striped format.

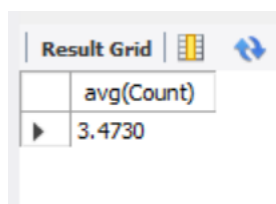
user_id	Count
23	12
88	11
59	10
86	9
58	8
29	8
77	6
33	5
52	5
47	5
6	5
13	5
51	5
78	5
11	5
1	5
72	5

Based on the derived table, we get the average of the count of user's post, using below query:

Query:

```
SELECT avg(Count) FROM ig_clone.user_post;
```

The output:



The screenshot shows a 'Result Grid' window with a table containing one column: 'avg(Count)'. The table has one row with the value 3.4730. The table is displayed in a light blue and white striped format.

avg(Count)
3.4730

Another question asked is: Also, provide the total number of photos on Instagram/total number of users

Query:

```
select count(distinct p.id) "Total Photos", count(distinct u.id) "Total Users",  
(select count(distinct id) from ig_clone.photos)/(select count(distinct id) from ig_clone.users) as  
Average  
from ig_clone.users u, ig_clone.photos p  
;
```

The output:

Result Grid			
Filter Rows:			
	Total Photos	Total Users	Average
▶	257	100	2.5700

The Final Result: On an average a user posts 3 times a day, total number of users is 100, total number of photos posted is 257 and total number of photos on Instagram/total number of users is 2.57



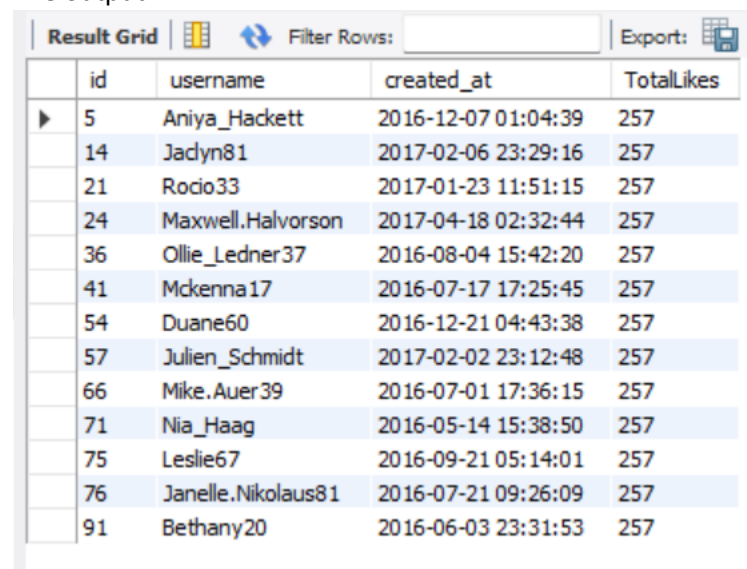
2. Bots & Fake Accounts: The investors want to know if the platform is crowded with fake and dummy accounts

For this we find the user who has liked every single photo.

Query:

```
select u.id, u.username, u.created_at, count(*) as TotalLikes from ig_clone.users u
inner join ig_clone.likes l
on
l.user_id=u.id
group by l.user_id
having TotalLikes=(select count(p.id) from ig_clone.photos p)
order by count(photo_id) desc
;
```

The output:



The screenshot shows a database query result grid with the following columns: id, username, created\_at, and TotalLikes. There are 13 rows of data, all showing a TotalLikes value of 257. The interface includes a 'Result Grid' tab, a 'Filter Rows' search bar, and an 'Export' button.

	id	username	created_at	TotalLikes
▶	5	Aniya_Hackett	2016-12-07 01:04:39	257
	14	Jadyn81	2017-02-06 23:29:16	257
	21	Rocio33	2017-01-23 11:51:15	257
	24	Maxwell.Halvorson	2017-04-18 02:32:44	257
	36	Ollie_Ledner37	2016-08-04 15:42:20	257
	41	Mckenna17	2016-07-17 17:25:45	257
	54	Duane60	2016-12-21 04:43:38	257
	57	Julien_Schmidt	2017-02-02 23:12:48	257
	66	Mike.Auer39	2016-07-01 17:36:15	257
	71	Nia_Haag	2016-05-14 15:38:50	257
	75	Leslie67	2016-09-21 05:14:01	257
	76	Janelle.Nikolaus81	2016-07-21 09:26:09	257
	91	Bethany20	2016-06-03 23:31:53	257

The Final Result: As shown in image above, total 13 accounts are detected to be bots/fake accounts.

### Result:

This project has helped me learn how to sort the dataset, link tables, and how to get insights from the data. It has helped me practice the queries and get hands on experience on analysing the data.