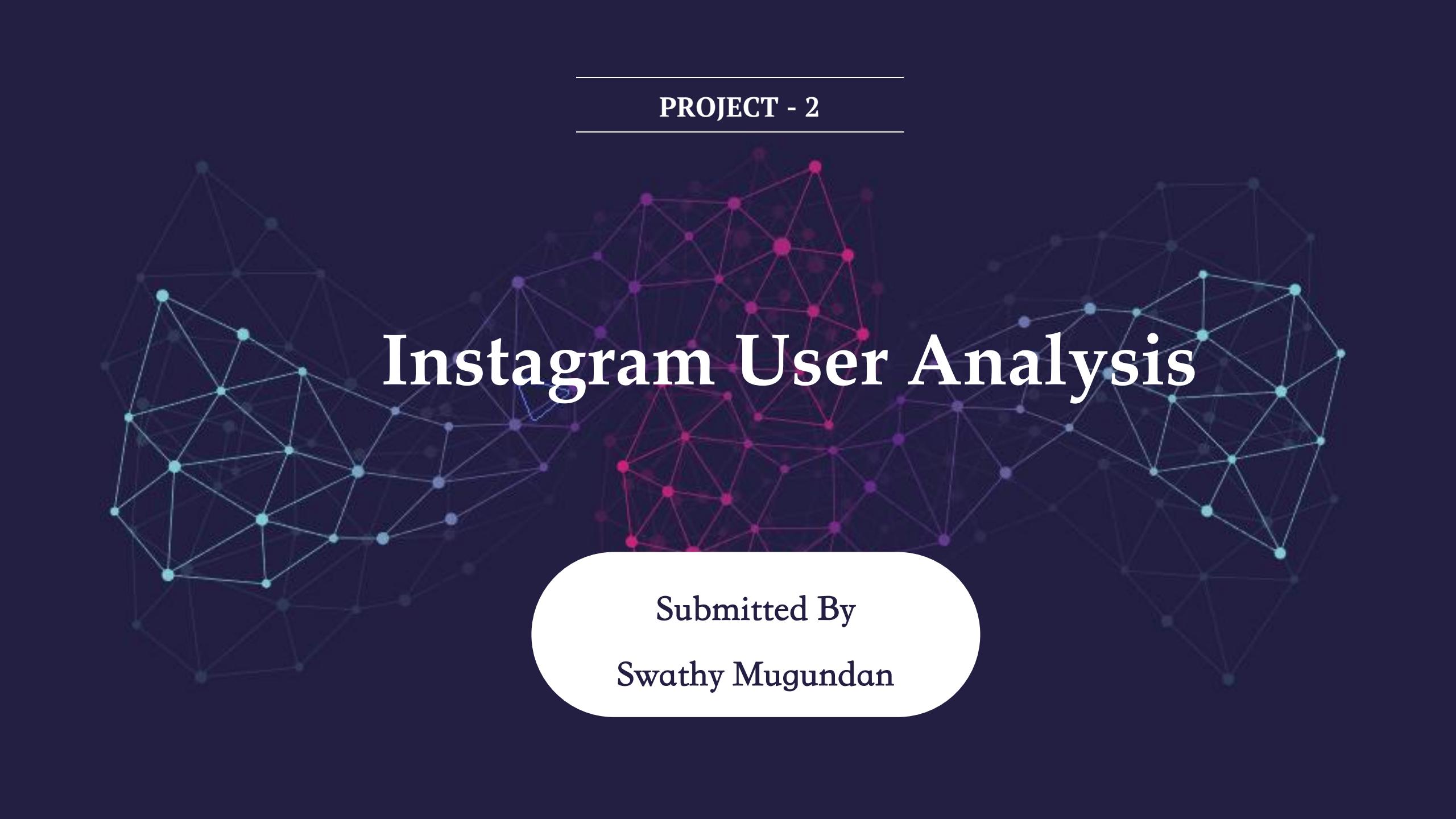


---

PROJECT - 2

---

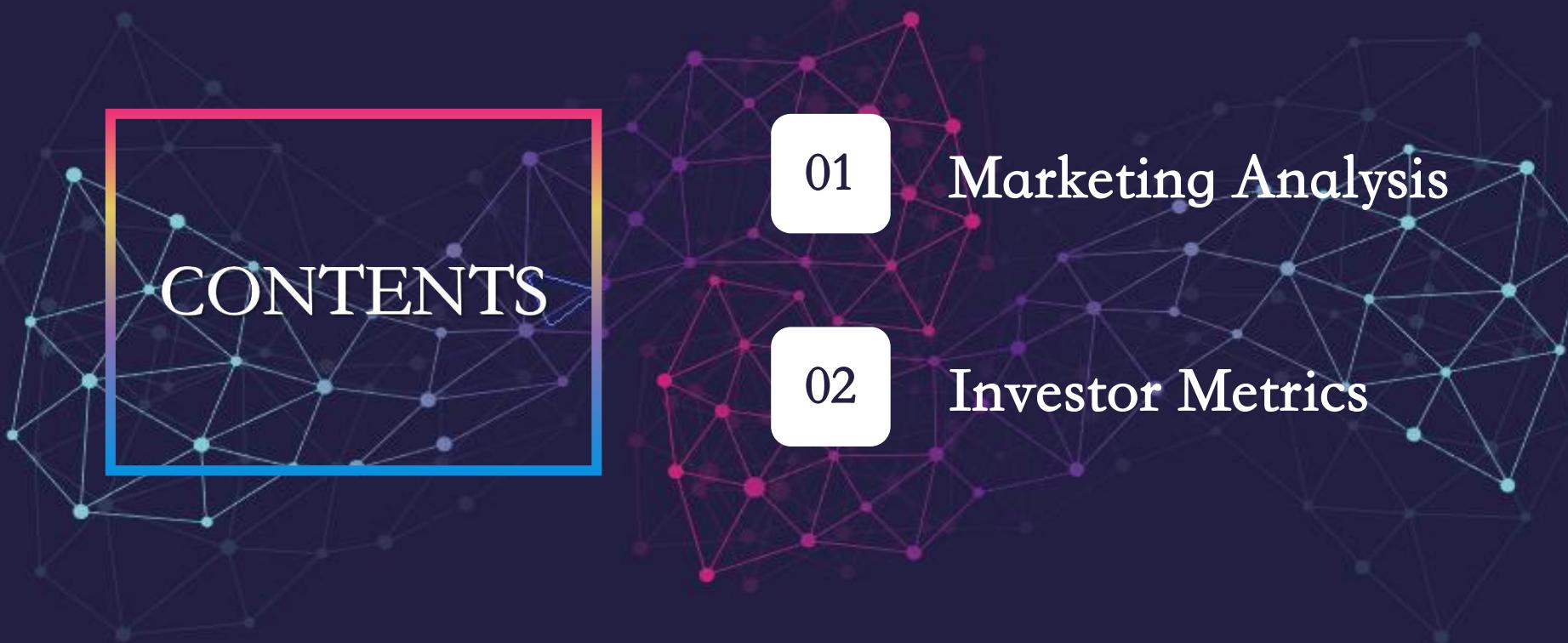
A dark purple background featuring three abstract network graphs composed of colored dots (nodes) and connecting lines (edges). One graph is teal on the left, one is magenta in the center, and one is light blue on the right.

# Instagram User Analysis

Submitted By

Swathy Mugundan

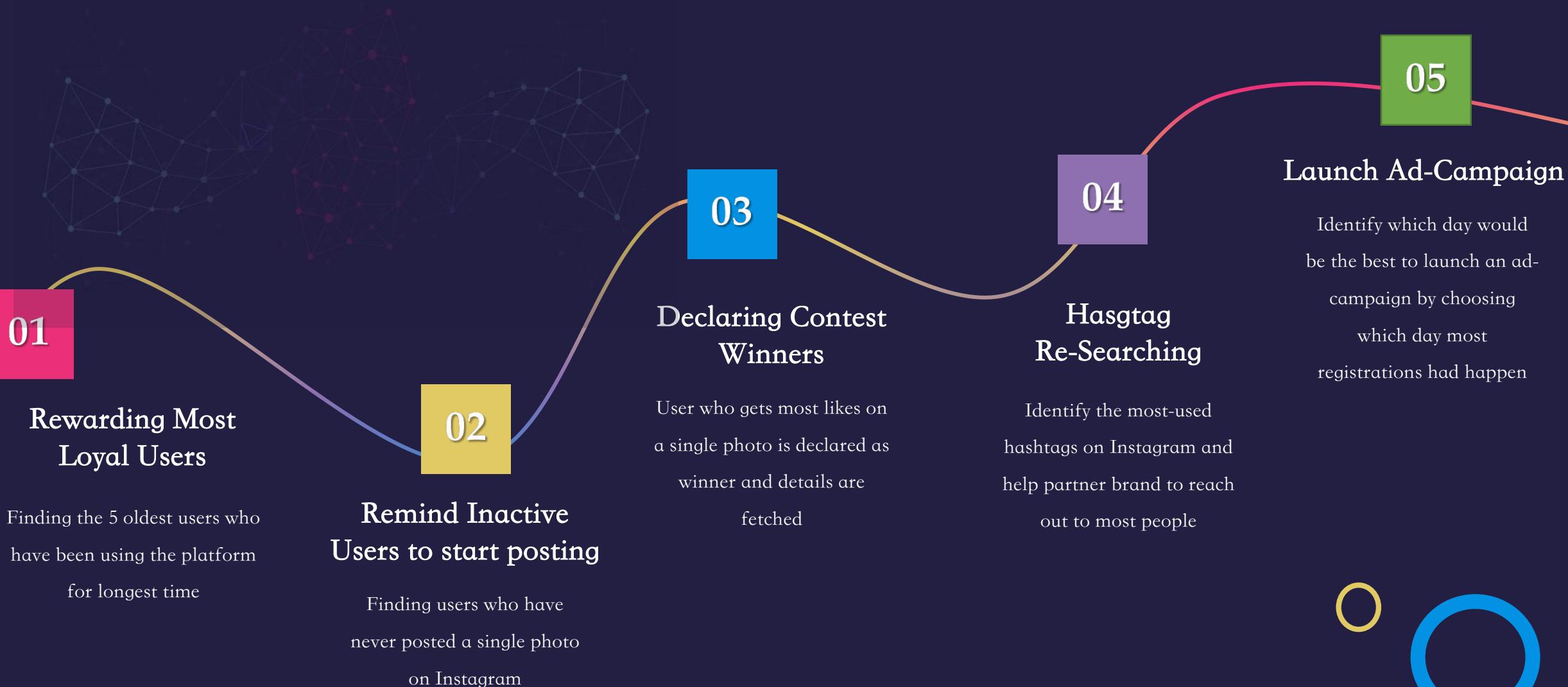
# Project 2 : Instagram User Analysis



# Problem Statement : Instagram User Analysis

- User analysis is the process by which we track how users engage and interact with our digital product (software or mobile application) in an attempt to derive business insights for marketing, product & development teams.
- These insights are then used by teams across the business to launch a new marketing campaign, decide on features to build for an app, track the success of the app by measuring user engagement and improve the experience altogether while helping the business grow.
- You are working with the product team of Instagram and the product manager has asked you to provide insights on the questions asked by the management team.

# Project Description : Marketing Analysis



# Project Description : Investor Metrics



## User Engagement

Find the activity status of the users by checking their average posts, total number of photos and total number of users

## Bots & Fake Accounts

Find the users who have liked all the posts (possibly a bot) :  
This helps to find the fake/dummy accounts on the platform

# Project Approach : Instagram Analysis



## DB Data

Download the dataset from the site, run the queries one by one and form the database with all the required tables.

## Problem Def'n

Analyse the tables and its attributes and understand the references between tables. Read the question properly.

## Query Formation

Figure out the related tables and write the suitable SQL Statement for fetching the results

## Result Analysis

Analyse and understand the result set and capture the inference of the problem statement,

# Tech Stack Used

## MySQL Server

User for the Relational Database Management System. To create database, To create Tables.

## MySQL Workbench

It is a GUI Based tool for Querying the Database. It is a visual-based DB tool to observe the databases , its tables, its attributes and how they are being stored in the system.

## Teampaper Snap

To take screenshots of the result-grid of the SQL Statements from SQL Workbench

# Project 2 : Instagram User Analysis

## Database Creation Screenshots

The screenshot shows a database management interface with a sidebar on the left containing a tree view of databases: companydb, ig\_clone (selected), Tables, Views, Stored Procedures, Functions, and sys. A yellow circle highlights the 'ig\_clone' database node. The main panel displays a command-line interface with the following session history:

```
1 *  CREATE DATABASE ig_clone;
```

The Action Output pane shows the results of the database creation and subsequent SELECT statements. A blue box highlights the last command in the history:

Action	Response	Duration / Fetch Time
select * from employee LIMIT 0, 1000	8 row(s) returned	0.00048 sec / 0.000...
SELECT SUM(salary) from employee...	1 row(s) returned	0.00054 sec / 0.000...
SELECT SUM(salary) from employee...	1 row(s) returned	0.00052 sec / 0.000...
SELECT SUM(salary) from employee...	1 row(s) returned	0.00051 sec / 0.000...
SELECT SUM(salary) from employee...	1 row(s) returned	0.00051 sec / 0.000...
SELECT SUM(salary) from employee...	1 row(s) returned	0.00069 sec / 0.000...
SELECT MIN(salary) from employee...	1 row(s) returned	0.00057 sec / 0.000...
SELECT MIN(salary) as MIN, MAX(s...	1 row(s) returned	0.00058 sec / 0.000...
SELECT sum(salary) from employee...	3 row(s) returned	0.0017 sec / 0.00001...
SELECT dno,sum(salary) from empl...	3 row(s) returned	0.00017 sec / 0.000...
CREATE DATABASE ig_clone	1 row(s) affected	0.0033 sec

# Project 2 : DB-Tables Creation Screenshots

**Users Table**

```
1  /*Users*/
2  CREATE TABLE users(
3      id INT AUTO_INCREMENT UNIQUE PRIMARY KEY,
4      username VARCHAR(255) NOT NULL,
5      created_at TIMESTAMP DEFAULT NOW()
6  );
7
8 * INSERT INTO users (username, created_at) VALUES ('Kenton_Kirlin', '2017-02-16 18:22:10.846'), ('Andre_Purdy85', '2017-04-02 17:11:21.417'), ('Ha
9
10 * select * from users;
```

Result Grid    Filter Rows:  Search    Edit: Export/Import:

	id	username	created_at
1	1	Kenton_Kirlin	2017-02-16 18:22:11
2	2	Andre_Purdy85	2017-04-02 17:11:21
3	3	Harley_Lind18	2017-02-21 11:12:33
4	4	Arely_Bogan63	2016-08-13 01:28:43
5	5	Aniya_Hackett	2016-12-07 01:04:39
6	6	Travon.Waters	2017-04-30 13:26:14
7	7	Kassandra_Homenick	2016-12-12 06:50:08
8	8	Tabitha_Schamberger11	2016-08-20 02:19:46
9	9	Gus93	2016-06-24 19:36:31
10	10	Presley_McClure	2016-08-07 16:25:49
11	11	Justina.Gaylord27	2017-05-04 16:32:16
12	12	Dereck65	2017-01-19 01:34:14
13	13	Alexandro35	2017-03-29 17:09:02
14	14	Jaclyn81	2017-02-06 23:29:16

users 19    Apply Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
1	13:42:24	CREATE TABLE users( id INT AUTO_INCREMENT UNIQUE PRIMARY KEY, username VARCHAR(255) NOT NULL, creat...	0 row(s) affected	0.011 sec
2	13:42:48	INSERT INTO users (username, created_at) VALUES ('Kenton_Kirlin', '2017-02-16 18:22:10.846'), ('Andre_Purdy85', '2... 100 row(s) affected Records: 100 Duplicates: 0 War...	100 row(s) affected Records: 100 Duplicates: 0 War...	0.0054 sec
3	13:43:06	select * from users LIMIT 0, 1000	100 row(s) returned	0.00080 sec / 0.000...

# Project 2 : DB-Tables Creation Screenshots

**Photos Table**

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

- Schemas:** companydb, ig\_clone (selected), users, sys.
- Tables:** photos (selected).
- Script Editor:** Shows the SQL code for creating the photos table and inserting 14 rows of data.
- Result Grid:** Displays the 14 rows of data inserted into the photos table.
- Action Output:** Shows the log of actions taken, including the creation of the table and the insertion of 14 rows.
- Right Panel:** Contains icons for Result Grid, Form Editor, Field Types, and Query Stats.

```
CREATE TABLE photos(
    id INT AUTO_INCREMENT PRIMARY KEY,
    image_url VARCHAR(355) NOT NULL,
    user_id INT NOT NULL,
    created_dat TIMESTAMP DEFAULT NOW(),
    FOREIGN KEY(user_id) REFERENCES users(id)
);

INSERT INTO photos(image_url, user_id) VALUES ('http://elijah.biz', 1), ('https://shanon.org', 1), ('http://vicky.biz', 1), ('http://oleta.net', 1), ('https://jennings.biz', 1), ('https://quinn.biz', 2), ('https://selina.name', 2), ('http://malvina.org', 2), ('https://branson.biz', 2), ('https://elenor.name', 3), ('https://marcelino.name', 3), ('http://felicity.name', 3), ('https://fred.com', 3), ('https://gerhard.biz', 4)
```

id	image_url	user_id	created_dat
1	http://elijah.biz	1	2023-07-20 13:47:19
2	https://shanon.org	1	2023-07-20 13:47:19
3	http://vicky.biz	1	2023-07-20 13:47:19
4	http://oleta.net	1	2023-07-20 13:47:19
5	https://jennings.biz	1	2023-07-20 13:47:19
6	https://quinn.biz	2	2023-07-20 13:47:19
7	https://selina.name	2	2023-07-20 13:47:19
8	http://malvina.org	2	2023-07-20 13:47:19
9	https://branson.biz	2	2023-07-20 13:47:19
10	https://elenor.name	3	2023-07-20 13:47:19
11	https://marcelino...	3	2023-07-20 13:47:19
12	http://felicity.name	3	2023-07-20 13:47:19
13	https://fred.com	3	2023-07-20 13:47:19
14	https://gerhard.biz	4	2023-07-20 13:47:19

Action Output:

Time	Action	Response	Duration / Fetch Time
13:47:17	CREATE TABLE photos( id INT AUTO_INCREMENT PRIMARY KEY, image_url VARCHAR(355) NOT NULL, user_id INT...	0 row(s) affected	0.013 sec
13:47:19	INSERT INTO photos(image_url, user_id) VALUES ('http://elijah.biz', 1), ('https://shanon.org', 1), ('http://vicky.biz', 1), ('http://oleta.net', 1), ('https://jennings.biz', 1), ('https://quinn.biz', 2), ('https://selina.name', 2), ('http://malvina.org', 2), ('https://branson.biz', 2), ('https://elenor.name', 3), ('https://marcelino.name', 3), ('http://felicity.name', 3), ('https://fred.com', 3), ('https://gerhard.biz', 4)	257 row(s) affected Records: 257 Duplicates: 0 Warnings: 0	0.0071 sec
13:47:20	select * from photos LIMIT 0, 1000	257 row(s) returned	0.0011 sec / 0.00006...

# Project 2 : DB-Tables Creation Screenshots

**Comments Table**

The screenshot shows the MySQL Workbench interface with a dark theme. The left sidebar displays the 'Schemas' tree, where the 'ig\_clone' schema is selected. Under 'Tables', the 'comments' table is highlighted. The main pane contains the SQL code for creating the table and inserting data. The 'Result Grid' pane shows the 14 rows of data inserted into the 'comments' table. The bottom pane shows the execution history and action output.

```
3 id INT AUTO_INCREMENT PRIMARY KEY,
4 comment_text VARCHAR(255) NOT NULL,
5 user_id INT NOT NULL,
6 photo_id INT NOT NULL,
7 created_at TIMESTAMP DEFAULT NOW(),
8 FOREIGN KEY(user_id) REFERENCES users(id),
9 FOREIGN KEY(photo_id) REFERENCES photos(id)
10 );
11
12 • INSERT INTO comments(comment_text, user_id, photo_id) VALUES ('unde at dolorem', 2, 1), ('quae ea ducimus', 3, 1), ('alias a voluptatum', 5, 1),
13
14 • select * from comments;
15
16
```

	id	comment_text	user_id	photo_id	created_at
1	1	unde at dolorem	2	1	2023-07-20 14:01:18
2	2	quae ea ducimus	3	1	2023-07-20 14:01:18
3	3	alias a voluptatum	5	1	2023-07-20 14:01:18
4	4	facere suscipit sunt	14	1	2023-07-20 14:01:18
5	5	totam eligendi quaerat	17	1	2023-07-20 14:01:18
6	6	vitae quia aliquam	21	1	2023-07-20 14:01:18
7	7	exercitationem occaecati neque	24	1	2023-07-20 14:01:18
8	8	sint ad fugiat	31	1	2023-07-20 14:01:18
9	9	nesciunt aut nesciunt	36	1	2023-07-20 14:01:18
10	10	laudantium ut nostrum	41	1	2023-07-20 14:01:18
11	11	omnis aut asperiores	52	1	2023-07-20 14:01:18
12	12	et eum molestias	54	1	2023-07-20 14:01:18
13	13	alias paratur neque	55	1	2023-07-20 14:01:18
14	14	amet iure adipisci	57	1	2023-07-20 14:01:18

Object Info: Schema: ig\_clone

Action Output:

Time	Action	Response	Duration / Fetch Time
14:01:07	select * from comments LIMIT 0, 1000	Error Code: 1146. Table 'ig_clone.comments' doesn't exist	0.000 sec
14:01:14	CREATE TABLE comments( id INT AUTO_INCREMENT PRIMARY KEY, comment_text VARCHAR(255) NOT NULL, user_id INT NOT NULL, photo_id INT NOT NULL, created_at TIMESTAMP DEFAULT NOW(), FOREIGN KEY(user_id) REFERENCES users(id), FOREIGN KEY(photo_id) REFERENCES photos(id) );	0 row(s) affected	0.022 sec
14:01:18	INSERT INTO comments(comment_text, user_id, photo_id) VALUES ('unde at dolorem', 2, 1), ('quae ea ducimus', 3, 1), ('alias a voluptatum', 5, 1);	7488 row(s) affected Records: 7488 Duplicates: 0 Warnings: 0	0.096 sec
14:01:20	select * from comments LIMIT 0, 1000	1000 row(s) returned	0.0017 sec / 0.00029...

# Project 2 : DB-Tables Creation Screenshots

**Likes Table**

The screenshot shows a database management interface with a dark theme. On the left, the 'SCHEMAS' sidebar lists several databases, with 'ig\_clone' selected. Under 'ig\_clone', the 'Tables' section is expanded, showing 'comments' and 'likes'. The 'Columns' section for 'likes' is also expanded, listing 'user\_id', 'photo\_id', and 'created\_at'. Other sections like 'Indexes', 'Foreign Keys', and 'Triggers' are present but collapsed.

The main area displays the SQL code for creating the 'likes' table and inserting data:

```
2  CREATE TABLE likes(
3      user_id INT NOT NULL,
4      photo_id INT NOT NULL,
5      created_at TIMESTAMP DEFAULT NOW(),
6      FOREIGN KEY(user_id) REFERENCES users(id),
7      FOREIGN KEY(photo_id) REFERENCES photos(id),
8      PRIMARY KEY(user_id,photo_id)
9  );
10
11 •   INSERT INTO likes(user_id,photo_id) VALUES (2, 1), (5, 1), (9, 1), (10, 1), (11, 1), (14, 1), (19, 1), (21, 1), (24, 1), (35, 1), (36, 1), (41, 1);
12
13 •   select * from likes;
14
```

The 'Result Grid' tab is active, showing the data inserted into the 'likes' table:

user_id	photo_id	created_at
2	1	2023-07-20 14:04:20
2	4	2023-07-20 14:04:20
2	8	2023-07-20 14:04:20
2	9	2023-07-20 14:04:20
2	10	2023-07-20 14:04:20
2	11	2023-07-20 14:04:20
2	12	2023-07-20 14:04:20
2	13	2023-07-20 14:04:20
2	15	2023-07-20 14:04:20
2	23	2023-07-20 14:04:20
2	25	2023-07-20 14:04:20
2	27	2023-07-20 14:04:20
2	30	2023-07-20 14:04:20
2	34	2023-07-20 14:04:20

The bottom section shows the 'Action Output' log:

Action	Time	Response	Duration / Fetch Time
CREATE TABLE likes( user_id INT NOT NULL, photo_id INT NOT NULL, created_at TIMESTAMP DEFAULT NOW(), FO...	14:04:18	0 row(s) affected	0.016 sec
INSERT INTO likes(user_id,photo_id) VALUES (2, 1), (5, 1), (9, 1), (10, 1), (11, 1), (14, 1), (19, 1), (21, 1), (24, 1), (35, 1), (36, 1), (41, 1);	14:04:20	8782 row(s) affected Records: 8782 Duplicates: 0 W...	0.098 sec
select * from likes LIMIT 0, 1000	14:04:22	1000 row(s) returned	0.0016 sec / 0.00022...

# Project 2 : DB-Tables Creation Screenshots

## Follows Table

The screenshot shows the MySQL Workbench interface with a pink border. On the left, the Schemas tree shows the ig\_clone schema selected. The Tables node is expanded, and the follows table is selected. The central pane displays the SQL code for creating the follows table:

```
3 |     follower_id INT NOT NULL,
4 |     followee_id INT NOT NULL,
5 |     created_at TIMESTAMP DEFAULT NOW(),
6 |     FOREIGN KEY (follower_id) REFERENCES users(id),
7 |     FOREIGN KEY (followee_id) REFERENCES users(id),
8 |     PRIMARY KEY(follower_id,followee_id)
9 |
10|
11|
12 •   INSERT INTO follows(follower_id, followee_id) VALUES (2, 1), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (2, 8), (2, 9), (2, 10), (2, 11), (2, 12),
13|
14 •   select * from follows;
```

The Result Grid below shows 14 rows of data inserted into the follows table:

follower_id	followee_id	created_at
2	1	2023-07-20 14:22:32
2	3	2023-07-20 14:22:32
2	4	2023-07-20 14:22:32
2	5	2023-07-20 14:22:32
2	6	2023-07-20 14:22:32
2	7	2023-07-20 14:22:32
2	8	2023-07-20 14:22:32
2	9	2023-07-20 14:22:32
2	10	2023-07-20 14:22:32
2	11	2023-07-20 14:22:32
2	12	2023-07-20 14:22:32
2	13	2023-07-20 14:22:32
2	14	2023-07-20 14:22:32
2	15	2023-07-20 14:22:32

The Action Output pane at the bottom shows the history of operations:

Action	Time	Response	Duration / Fetch Time
CREATE TABLE follows( follower_id INT NOT NULL, followee_id INT NOT NULL, created_at TIMESTAMP DEFAULT NO...)	14:22:30	0 row(s) affected	0.017 sec
INSERT INTO follows(follower_id, followee_id) VALUES (2, 1), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (2, 8), (2, 9), (2, 10), (2, 11), (2, 12)	14:22:32	7623 row(s) affected Records: 7623 Duplicates: 0 W... 1000 row(s) returned	0.082 sec
select * from follows LIMIT 0, 1000	14:22:34		0.0016 sec / 0.00018...

# Project 2 : DB-Tables Creation Screenshots

MySQL Workbench

Administration   Schemas   Query 1

Limit to 1000 rows

1 /\*Tags\*/  
2 CREATE TABLE tags(  
3     id INTEGER AUTO\_INCREMENT PRIMARY KEY,  
4     tag\_name VARCHAR(255) UNIQUE NOT NULL,  
5     created\_at TIMESTAMP DEFAULT NOW()  
6 );  
7  
9 INSERT INTO tags(tag\_name) VALUES ('sunset'), ('photography'), ('sunrise'), ('landscape'), ('food'), ('foodie'), ('delicious'), ('beauty'), ('stunning'), ('dreamy'), ('lol'), ('happy'), ('fun'), ('lstyle');  
10  
11 select \* from tags;

Result Grid   Filter Rows: Search   Edit: Export/Import:

	id	tag_name	created_at
1	1	sunset	2023-07-20 14:06:07
2	2	photography	2023-07-20 14:06:07
3	3	sunrise	2023-07-20 14:06:07
4	4	landscape	2023-07-20 14:06:07
5	5	food	2023-07-20 14:06:07
6	6	foodie	2023-07-20 14:06:07
7	7	delicious	2023-07-20 14:06:07
8	8	beauty	2023-07-20 14:06:07
9	9	stunning	2023-07-20 14:06:07
10	10	dreamy	2023-07-20 14:06:07
11	11	lol	2023-07-20 14:06:07
12	12	happy	2023-07-20 14:06:07
13	13	fun	2023-07-20 14:06:07
14	14	lstyle	2023-07-20 14:06:07

Object Info   Session   Schema: ig\_clone

tags 23   Apply   Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
1	14:06:05	CREATE TABLE tags( id INTEGER AUTO_INCREMENT PRIMARY KEY, tag_name VARCHAR(255) UNIQUE NOT NULL,...	0 row(s) affected	0.012 sec
2	14:06:07	INSERT INTO tags(tag_name) VALUES ('sunset'), ('photography'), ('sunrise'), ('landscape'), ('food'), ('foodie'), ('delicious'), ('beauty'), ('stunning'), ('dreamy'), ('lol'), ('happy'), ('fun'), ('lstyle');	21 row(s) affected Records: 21 Duplicates: 0 Warnings: 0	0.0026 sec
3	14:06:09	select * from tags LIMIT 0, 1000	21 row(s) returned	0.00072 sec / 0.0000...

Result Grid   Form Editor   Field Types   Query Stats

## Tags Table

# Project 2 : DB-Tables Creation Screenshots

**Photo Tags Table**

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

- Schemas:** companydb, ig\_clone (selected).
- Tables:** comments, follows, likes, photos, photo\_tags (selected).
- photo\_tags Columns:** photo\_id, tag\_id.
- Script Area:** SQL code for creating the table and inserting data.
- Result Grid:** Shows the inserted data.
- Action Output:** Shows the execution history and results of the queries.

**SQL Script:**

```
2 CREATE TABLE photo_tags(
3     photo_id INT NOT NULL,
4     tag_id INT NOT NULL,
5     FOREIGN KEY(photo_id) REFERENCES photos(id),
6     FOREIGN KEY(tag_id) REFERENCES tags(id),
7     PRIMARY KEY(photo_id,tag_id)
8 );
9
10
11 • INSERT INTO photo_tags(photo_id, tag_id) VALUES (1, 18), (1, 17), (1, 21), (1, 13), (1, 19), (2, 4), (2, 3), (2, 20), (2, 2), (3, 8), (4, 12), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10), (11, 11), (12, 12), (13, 13), (14, 14), (15, 15), (16, 16), (17, 17), (18, 18), (19, 19), (20, 20), (21, 21), (22, 22), (23, 23), (24, 24), (25, 25);
12
13 • select * from photo_tags;
```

**Result Grid:**

photo_id	tag_id
14	1
21	1
45	1
75	1
83	1
85	1
91	1
118	1
149	1
194	1
201	1
210	1
216	1
227	1

**Action Output:**

Action	Time	Response	Duration / Fetch Time
CREATE TABLE photo_tags( photo_id INT NOT NULL, tag_id INT NOT NULL, FOREIGN KEY(photo_id) REFERENCES...	14:24:51	0 row(s) affected	0.015 sec
INSERT INTO photo_tags(photo_id, tag_id) VALUES (1, 18), (1, 17), (1, 21), (1, 13), (1, 19), (2, 4), (2, 3), (2, 20), (2, 2), (3, 8), (4, 12), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10), (11, 11), (12, 12), (13, 13), (14, 14), (15, 15), (16, 16), (17, 17), (18, 18), (19, 19), (20, 20), (21, 21), (22, 22), (23, 23), (24, 24), (25, 25);	14:24:53	501 row(s) affected Records: 501 Duplicates: 0 Warn... 0.011 sec	
select * from photo_tags LIMIT 0, 1000	14:24:55	501 row(s) returned	0.00088 sec / 0.000...

# Marketing Analysis : Loyal User Reward

```
select id, username  
from users  
order by created_at  
LIMIT 5;
```

Result Grid Filter Rows:

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

# Marketing Analysis : Inactive Users (Zero Photos Posted)

```
select users.id, username  
from users left join photos  
on users.id = photos.user_id  
where image_url is null;
```

id	username
5	Aniya_Hackett
7	Kasandra_Homenick
14	Jaclyn81
21	Rocio33
24	Maxwell.Halvorson
25	Tierra.Trantow
34	Pearl7
36	Ollie_Ledner37
41	Mckenna17
45	David.Osinski47
49	Morgan.Kassulke
53	Linnea59
54	Duane60
57	Julien_Schmidt
66	Mike.Auer39
68	Franco_Keebler64
71	Nia_Haag
74	Hulda.Macejkovic
74	Hulda.Macejkovic
75	Leslie67
76	Janelle.Nikolaus81
80	Darby_Herzog
81	Esther.Zulauf61
83	Bartholome.Bernhard
89	Jessyca_West
90	Esmeralda.Mraz57
91	Bethany20

# Marketing Analysis : Contest Winner Declaration

```
select U.id, U.username, P.photo_id, P.count as Likes  
from users U  
join  
(select photos.user_id, photo_id, count(photo_id) as count  
     from likes left join photos  
   on likes.photo_id = photos.id  
group by likes.photo_id  
order by count desc) P  
on U.id = P.user_id  
order by likes desc;
```

**CONCLUSION :** Based on the result set, we can conclude that Zack\_Kemmer93 is the winner for securing most likes in single photo

Result Grid				
	id	username	photo_id	Likes
	52	Zack_Kemmer93	145	48
	46	Malinda_Streich	127	43
	65	Adelle96	182	43
	44	Seth46	123	42
	10	Presley_McClure	30	41
	16	Annalise.McKenzie16	52	41
	20	Delpha.Kihn	61	41
	55	Meggie_Doyle	147	41
	63	Elenor88	174	41
	72	Kathryn80	192	41
	100	Javonte83	256	41
	3	Harley_Lind18	13	40
	32	Irwin.Larson	97	40
	58	Aurelie71	153	40
	59	Cesar93	161	40
	94	Damon35	244	40
	13	Alexandro35	44	39
	22	Kenneth64	62	39
	23	Eveline95	66	39
	33	Yvette.Gottlieb91	100	39
	37	Yazmin_Mills95	107	39

# Marketing Analysis : Hastag Research

```
select id, tag_name, count(tag_name) as count  
from tags left join photo_tags  
on tags.id = photo_tags.tag_id  
group by id  
order by count desc ;
```

## CONCLUSION

Based on the result set, we can conclude that most of the users used the "SMILE" hashtag.

id	tag_name	count
21	smile	59
20	beach	42
17	party	39
13	fun	38
5	food	24
11	lol	24
18	concert	24
..	..	..

# Marketing Analysis : Ad-Campaign

```
select weekday(created_at) as day,  
       count(weekday(created_at)) as count  
  from users  
 group by day;
```

## CONCLUSION

Based on the result set, we can conclude that most of the users have registered to the platform on Wednesday (3) and Saturday (6). Hence, ad campaign can be planned on those days.

Result Grid Filter Rows:

day	count
3	16
6	16
1	14
5	12
2	13
0	14
4	15

# Investor Metrics : Average Analysis (1)

```
select user_id as ID,  
       count(user_id) as No_of_Posts,  
       257/count(user_id) as Average  
  from photos  
 group by user_id;
```

## NOTE

The total number of posts  
is 257 and users is 100.

ID	No_of_Posts	Average
1	5	51.4000
2	4	64.2500
3	4	64.2500
4	3	85.6667
6	5	51.4000
8	4	64.2500
9	4	64.2500
10	3	85.6667
11	5	51.4000
12	4	64.2500
13	5	51.4000
15	4	64.2500
16	4	64.2500
17	3	85.6667
18	1	257.0000
19	2	128.5000
20	1	257.0000
22	1	257.0000
23	12	21.4167
26	5	51.4000
27	1	257.0000

ID	No_of_Posts	Average
27	1	257.0000
28	4	64.2500
29	8	32.1250
30	2	128.5000
31	1	257.0000
32	4	64.2500
33	5	51.4000
35	2	128.5000
37	1	257.0000
38	2	128.5000
39	1	257.0000
40	1	257.0000
42	3	85.6667
43	5	51.4000
44	4	64.2500
46	4	64.2500
47	5	51.4000
48	1	257.0000
50	3	85.6667
51	5	51.4000

# Investor Metrics : Average Analysis (2)

```
select user_id as ID,  
       count(user_id) as No_of_Posts,  
       257/count(user_id) as Average  
  from photos  
 group by user_id;
```

## NOTE

The total number of posts  
is 257 and users is 100.

ID	No_of_Posts	Average
52	5	51.4000
55	1	257.0000
56	1	257.0000
58	8	32.1250
59	10	25.7000
60	2	128.5000
61	1	257.0000
62	2	128.5000
63	4	64.2500
64	5	51.4000
65	5	51.4000
67	3	85.6667
69	1	257.0000
70	1	257.0000
72	5	51.4000

ID	No_of_Posts	Average
72	5	51.4000
73	1	257.0000
77	6	42.8333
78	5	51.4000
79	1	257.0000
82	2	128.5000
84	2	128.5000
85	2	128.5000
86	9	28.5556
87	4	64.2500
88	11	23.3636
92	3	85.6667
93	2	128.5000
94	1	257.0000
95	2	128.5000
96	3	85.6667
97	2	128.5000
98	1	257.0000
99	3	85.6667
100	2	128.5000

# Investor Metrics : Bots & Fake Accounts

```
select U.id as ID, U.username as Uname, L.count as Likes  
from users U  join  
(select user_id, count(user_id) as count from likes  
group by user_id  
order by count desc) L  
on U.id = L.user_id  
where L.count = 257  
group by U.id;
```

## NOTE

The total number of posts is 257.  
If a user is liking all 257 posts  
then it can possibly be a bot.

ID	Uname	Likes
5	Aniya_Hackett	257
14	Jaclyn81	257
21	Rocio33	257
24	Maxwell.Halvorson	257
36	Ollie_Ledner37	257
41	Mckenna17	257
54	Duane60	257
57	Julien_Schmidt	257
66	Mike.Auer39	257
71	Nia_Haag	257
75	Leslie67	257
76	Janelle.Nikolaus81	257
91	Bethany20	257

# Project 2 : Result Analysis & Insights

- The required database is formed and the tables are populated. SQL Statements were ran on the DB to fetch the results of the required problem statements.
- Marketing Analysis steps helps us to understand the users and their respective actions on the Instagram platform.
- Investor Analysis helps us to observe the average user posts and the fake bots available with the instagram platform
- Thus, using a simple SQL Query we were able to analyse the data and figured out certain metrics and analytics upon it.



# Thank You

Submitted By

Swathy Mugundan