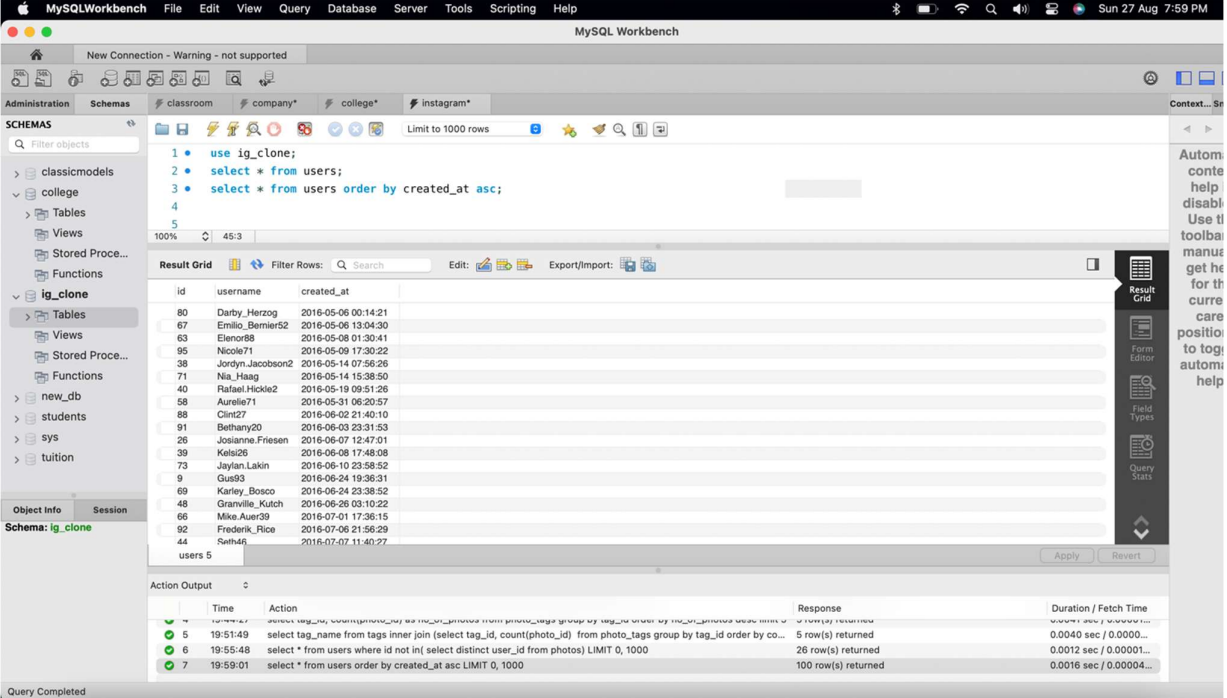


Project Description: The main purpose of this project is to find those data from Instagram through which we can find regular, real user accounts to give them incentives and to find those users who seems to be less active should be motivated through some lucrative ideas and if they are not active for a long period of time, then the account should be deactivated. In short our main focus will be company's growth.

The approach we'll take to handle this project by collecting data in our software and will perform queries on them according to our target and find the result to derive the insights.

Approach:

- 1) Performed order by on created_at column in users table.



The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'Schemas' panel with 'ig_clone' selected. The central query editor contains the following SQL query:

```
1 • use ig_clone;
2 • select * from users;
3 • select * from users order by created_at asc;
4
```

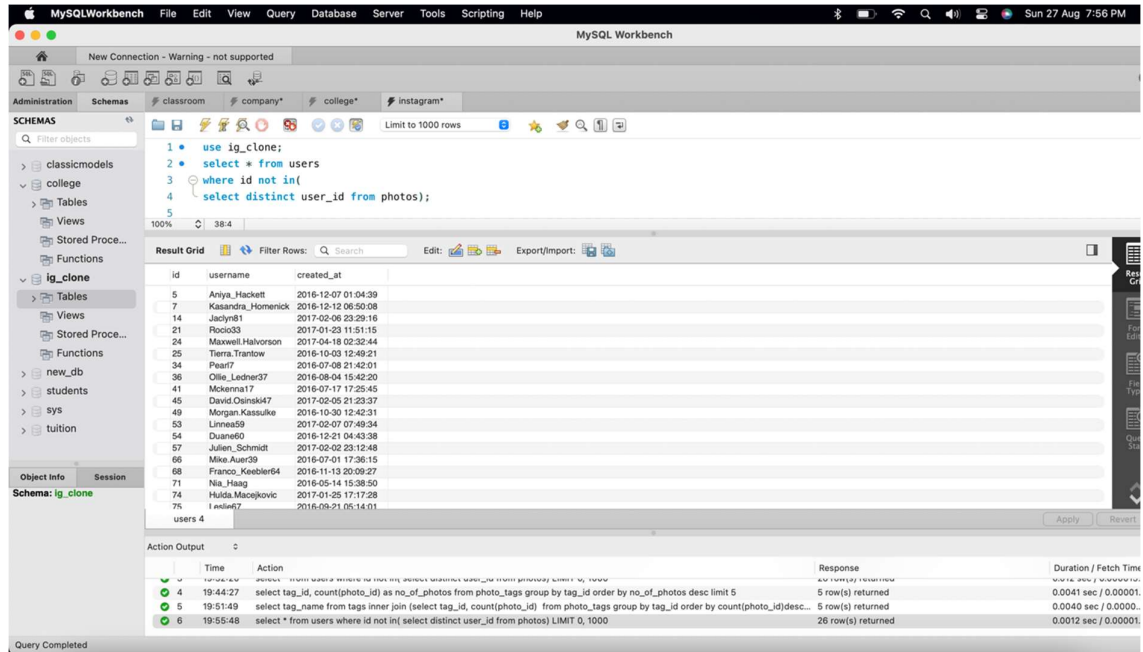
The 'Result Grid' shows the execution results for the query. The columns are 'id', 'username', and 'created_at'. The data is sorted by 'created_at' in ascending order. The first few rows are:

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
71	Nia_Haag	2016-05-14 15:38:50
40	Rafael_Hickie2	2016-05-19 09:51:26
58	Aurelie71	2016-05-31 06:20:57
88	Clin227	2016-06-02 21:40:10
91	Bethany20	2016-06-03 23:31:53
26	Josianne_Friesen	2016-06-07 12:47:01
39	Kelsi26	2016-06-08 17:48:08
73	Jaylan_Lakin	2016-06-10 23:58:52
9	Gus93	2016-06-24 19:36:31
69	Karley_Bosco	2016-06-24 23:38:52
48	Granville_Kutch	2016-06-26 03:10:22
66	Mike_Auer39	2016-07-01 17:36:15
92	Frederik_Rice	2016-07-06 21:56:29
44	Ruth46	2016-07-07 11:40:27

The 'Action Output' panel at the bottom shows the query execution details:

Time	Action	Response	Duration / Fetch Time
19:51:49	select tag_name from tags inner join (select tag_id, count(photo_id) from photo_tags group by tag_id order by co...	5 row(s) returned	0.0040 sec / 0.0000...
19:55:48	select * from users where id not in(select distinct user_id from photos) LIMIT 0, 1000	26 row(s) returned	0.0012 sec / 0.00001...
19:59:01	select * from users order by created_at asc LIMIT 0, 1000	100 row(s) returned	0.0016 sec / 0.00004...

- 2) Selected those user id which were not present in photos column through subqueries.



3) First, I viewed the likes table then I noticed that by grouping that data of photo id with counting of user id we can find the number of likes on a single photo and can order them accordingly. So by grouping the data I made a table.

```
1 • use ig_clone;
2 • select * from likes;
3 • select photo_id, count(user_id) as photo_likes
4 • from likes
5 • group by photo_id;
```

Using these queries , I grouped the table

photo_id	photo_likes
1	25
2	36
3	38
4	38
5	31
6	31
7	38
8	27
9	31
10	30
11	33
12	29
13	40
14	35
15	34
16	37
17	36
18	31
19	35
20	35
21	35
22	28
23	38
24	35
25	36
26	30
27	31
28	36
29	33

Then I found the maximum value of photo_likes using subqueries

```

2 • select max(photo_likes)
3   from (select photo_id, count(user_id) as photo_likes
4         from likes
5        group by photo_id)
6   as MaxphotoLikes;
7

```

max(photo_lik...
48

After finding the maximum value we found the photo id according to that by using that group by from table and using that maximum values query in where clause with photo_likes so that we can find the photo_id where the likes are maximum. Following query we used.

```

7 • select photo_id
8   from (select photo_id, count(user_id) as photo_likes
9         from likes
10        group by photo_id
11       ) as total_likes
12  where photo_likes = (select max(photo_likes)
13                     from (select photo_id, count(user_id) as photo_likes
14                           from likes
15                          group by photo_id
16                         ) as total_likes);
17
18
19
20
21

```

photo_id
145

Now we are going to use this query as subquery in another query to find the user id of that person who posted that photo. So, we just copied the previous command of photo_id in where clause for finding the users id of that photo id in users table.

```

18 • select user_id from photos
19  where id= ( select photo_id
20            from (select photo_id, count(user_id) as photo_likes
21                  from likes
22                 group by photo_id
23                ) as total_likes
24            where photo_likes = (select max(photo_likes)
25                                from (select photo_id, count(user_id) as photo_likes
26                                      from likes
27                                     group by photo_id
28                                    ) as total_likes));

```

user_id
52

Now, using this user_id we will print the users information by copying that query in users table as a subquery.

In where clause of id of pasted the query through which we found the user id.

```
2 • select * from users
3   where id = (select user_id from photos
4   where id = (select photo_id
5   from(select photo_id,count(user_id) as photo_likes
6   from likes
7   group by photo_id
8   ) as total_likes
9   where photo_likes = (select max(photo_likes)
10  from(select photo_id,count(user_id) as photo_likes
11  from likes
12  group by photo_id
13  ) as total_likes
14  )));
15
16
```

0% 5:14

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

id	username	created_at
52	Zack_Kemmer93	2017-01-01 05:58:22
users 8		

4) In the photo_tag table data I found that we can count the number of photo and group by it with tag_id to find the max tag used. Through this approach we made a query which prints the top 5 tags used through order by and limit :

```
2 • select tag_id, count(photo_id) as no_of_photos
3   from photo_tags
4   group by tag_id
5   order by no_of_photos desc
6   limit 5;
7
8
9
10
```

0% 9:6

Result Grid Filter Rows: Search Export: Fetch rows:

tag_id	no_of_phot...
21	59
20	42
17	39
13	38
18	24

And then to find the tag name of that id we inner joined the table with tags table

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```

1 • use ig_clone;
2 • select * from tags;
3 • select tag_name from tags
4 • inner join (select tag_id, count(photo_id) from photo_tags
5 • group by tag_id
6 • order by count(photo_id) desc
7 • limit 5) as top5
8 • on tags.id= top5.tag_id;
9

```

The Result Grid shows the following data:

tag_name
smile
beach
party
fun
concert

The Action Output table shows the execution details:

Time	Action	Response	Duration / Fetch Time
19:55:48	select * from users where id not in (select distinct user_id from photos) LIMIT 0, 1000	26 row(s) returned	0.0012 sec / 0.00001...
19:55:01	select * from users order by created_at asc LIMIT 0, 1000	100 row(s) returned	0.0016 sec / 0.00004...
20:05:29	select tag_name from tags inner join (select tag_id, count(photo_id) from photo_tags group by tag_id order by cou...	5 row(s) returned	0.0015 sec / 0.00000...

5) I grouped the data of day with count of users id that were there for that day.

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```

1 • use ig_clone;
2 • select * from users;
3 • select count(id), weekday(created_at) as day_
4 • from users group by day_
5 • order by count(id) desc;
6

```

The Result Grid shows the following data:

count(id)	day_
16	3
16	6
15	4
14	1
14	0
13	2
12	5

The Action Output table shows the execution details:

Time	Action	Response	Duration / Fetch Time
20:13:39	select user_id, count(photo_id) as no_of_likes from likes group by user_id having no_of_likes = (select count...	13 row(s) returned	0.013 sec / 0.000010...
20:24:36	select * from users where id = (select user_id from photos where id = (select photo_id from(select photo_id...	1 row(s) returned	0.010 sec / 0.000008...
20:50:57	select count(id), weekday(created_at) as day_, from users group by day_ order by count(id) desc LIMIT 0, 10...	7 row(s) returned	0.0031 sec / 0.00000...

6) Found total number of photos by counting it.

MySQL Workbench interface showing a query execution. The query is:

```
1 • use ig_clone;
2 • select count(id) from photos;
```

The result grid shows a single row with the value 257.

Time	Action	Response	Duration / Fe
15	22:10:27	select user_id from photos where id= (select photo_id from (select photo_id, count(user_id) as photo_likes from likes group by photo_id) as...	1 row(s) returned
16	22:18:42	select tag_id, count(photo_id) as no_of_photos from photo_tags group by tag_id order by no_of_photos desc limit 5	5 row(s) returned
17	22:27:20	select count(id) from photos LIMIT 0, 1000	1 row(s) returned

Dividing it by number of id's.

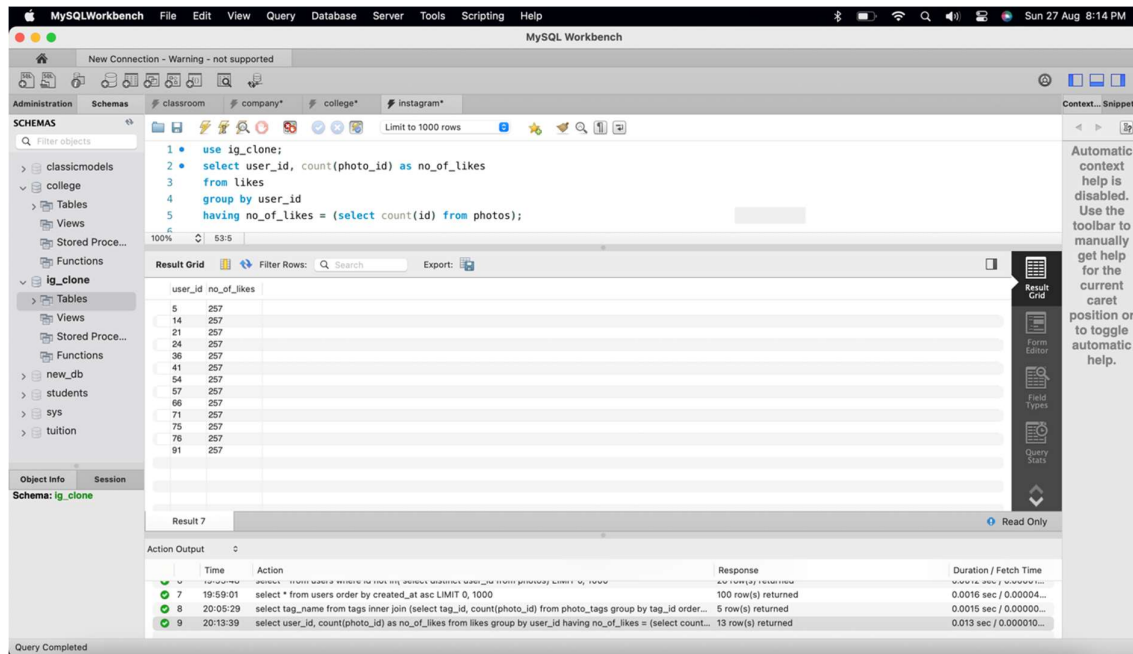
MySQL Workbench interface showing a query execution. The query is:

```
1 • use ig_clone;
2 • select count(id)/(select count(id) from users) from photos;
```

The result grid shows a single row with the value 2.5700.

Time	Action	Response	Duration / Fe
16	22:18:42	select user_id from photos where id= (select photo_id from (select photo_id, count(user_id) as photo_likes from likes group by photo_id) as...	1 row(s) returned
17	22:27:20	select count(id) from photos LIMIT 0, 1000	1 row(s) returned
18	22:28:59	select count(id)/(select count(id) from users) from photos LIMIT 0, 1000	1 row(s) returned

7) Grouped the data of user_id with count of photo_id having those rows where number of likes is equal to the total number of photos which gives those users who liked every photo.



Tech-Stack Used: MySQL Workbench.

My SQL Workbench is an easy to use tool for SQL through which we can perform different queries on database and work on them

Insights:

1)List of users on Instagram according to joining data. User joined first is on top.

MySQL Workbench interface showing a query result grid for the 'instagram' database. The query is: `select * from users order by created_at asc LIMIT 0, 1000`. The result grid displays 1000 rows of user data, including columns for id, username, and created_at. The first row is for user 80, Darby_Herzog, created at 2016-05-06 00:14:21. The last row is for user 4, Anely_Rocam3, created at 2016-06-13 01:28:43.

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
71	Nia_Haag	2016-05-14 15:38:50
40	Rafael_Hickle2	2016-05-19 09:51:26
58	Aunelle71	2016-05-31 06:20:57
88	Clint27	2016-06-02 21:40:10
91	Bethany20	2016-06-03 23:31:53
26	Josanne_Friesen	2016-06-07 12:47:01
39	Kelsi26	2016-06-08 17:48:08
73	Jaylan_Lakin	2016-06-10 23:58:52
9	Quis93	2016-06-24 19:36:31
69	Karley_Bosco	2016-06-24 23:38:52
48	Granville_Kutch	2016-06-26 03:10:22
66	Mike_Auer39	2016-07-01 17:36:15
92	Frederik_Rice	2016-07-06 21:56:29
44	Sebi46	2016-07-07 11:40:27
34	Pearl7	2016-07-08 21:42:01
46	Malinda_Streich	2016-07-09 21:37:08
41	McKenna17	2016-07-17 17:25:45
76	Janelle_Nikolaus81	2016-07-21 09:26:09
82	Aracely_Johnston...	2016-07-25 18:49:10
37	Yazmin_Mills95	2016-07-27 00:58:44
16	Annalisa_McKen...	2016-08-02 21:32:46
36	Olis_Ledner37	2016-08-04 15:42:20
10	Prestley_McClure	2016-08-07 16:25:49
4	Anely_Rocam3	2016-06-13 01:28:43

2) List of users who never posted any photo.

MySQL Workbench interface showing a query result grid for the 'instagram' database. The query is: `select * from users where id not in (select distinct user_id from photos) LIMIT 0, 1000`. The result grid displays 1000 rows of user data, including columns for id, username, and created_at. The first row is for user 5, Aniya_Hackett, created at 2016-12-07 01:04:39. The last row is for user 91, Bethany20, created at 2016-06-03 23:31:53.

id	username	created_at
5	Aniya_Hackett	2016-12-07 01:04:39
7	Kassandra_Homenick	2016-12-12 08:50:08
14	Jaclyn81	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_Tranlow	2016-10-03 12:49:21
34	Pearl7	2016-07-08 21:42:01
36	Olis_Ledner37	2016-08-04 15:42:20
41	McKenna17	2016-07-17 17:25:45
45	David_Olsinski47	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
74	Hilda_Macgiovic	2017-01-25 17:17:28
75	Leslie67	2016-09-21 05:14:01
76	Janelle_Nikolaus81	2016-07-21 09:26:09
80	Darby_Herzog	2016-05-06 00:14:21
81	Esther_Zulauf61	2017-01-14 17:02:34
83	Bartholome_Bernhard	2016-11-06 02:31:23
69	Jessica_West	2016-09-14 23:47:05
90	Esmeralda_Mraz57	2017-03-03 11:52:27
91	Bethany20	2016-06-03 23:31:53

3) Winner of contest with 48 likes on a single photo.

The screenshot shows the MySQL Workbench interface. The 'Schemas' panel on the left lists databases including 'college' and 'ig_clone'. The 'Result Grid' displays a single row of data for a user with ID 52, username 'Zack_Kemmer93', and created_at '2017-01-01 05:58:22'. The 'Action Output' panel at the bottom shows the execution of three queries. The third query, executed at 20:24:36, selects the user ID from the 'users' table based on the number of likes on a specific photo, returning 1 row.

id	username	created_at
52	Zack_Kemmer93	2017-01-01 05:58:22

Time	Action	Response	Duration / Fetch Time
20:05:29	select tag_name from tags inner join (select tag_id, count(photo_id) from photo_tags group by tag_id order by count(photo_id) desc) as tag_counts on tags.tag_id = tag_counts.tag_id	5 row(s) returned	0.0015 sec / 0.00000...
20:13:39	select user_id, count(photo_id) as no_of_likes from likes group by user_id having no_of_likes = (select count(photo_id) from photo_tags group by tag_id order by count(photo_id) desc) limit 1	13 row(s) returned	0.013 sec / 0.000010...
20:24:36	select * from users where id = (select user_id from photos where id = (select photo_id from (select photo_id, count(photo_id) as no_of_likes from photo_tags group by photo_id order by no_of_likes desc) as photo_likes limit 1))	1 row(s) returned	0.010 sec / 0.000008...

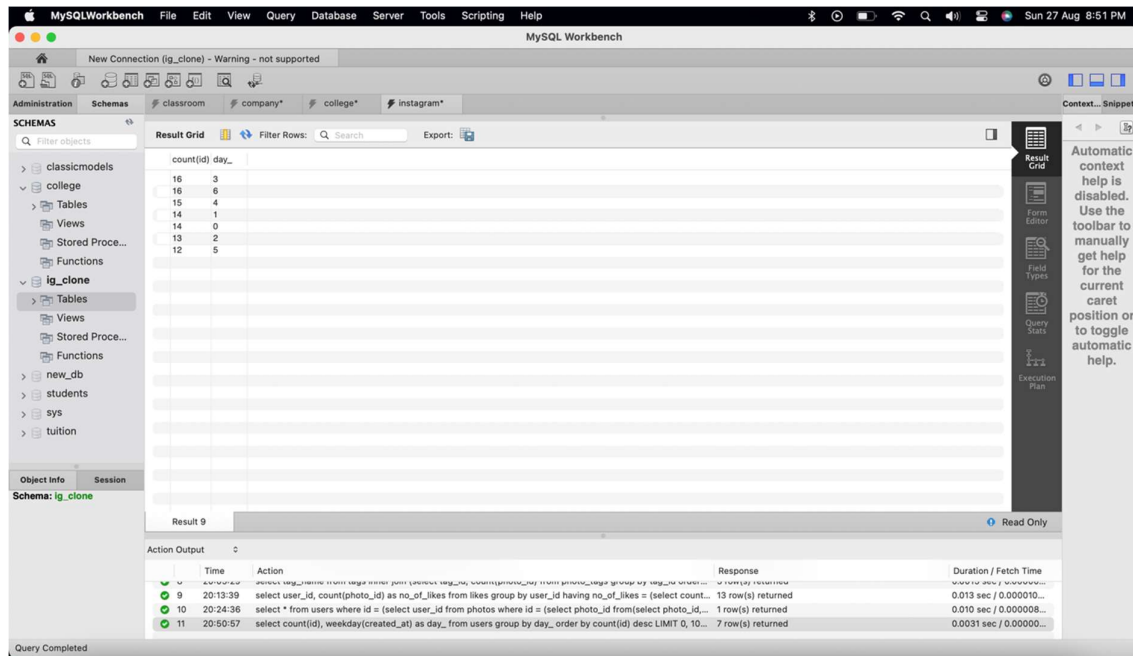
4) List of top 5 trending tags.

The screenshot shows the MySQL Workbench interface. The 'Schemas' panel on the left lists databases including 'college' and 'ig_clone'. The 'Result Grid' displays a list of five tags: 'smile', 'beach', 'party', 'fun', and 'concert'. The 'Action Output' panel at the bottom shows the execution of three queries. The third query, executed at 20:05:29, selects the top 5 trending tags based on the number of photos they appear in, returning 5 rows.

tag_name
smile
beach
party
fun
concert

Time	Action	Response	Duration / Fetch Time
19:55:48	select tag_name from tags inner join (select tag_id, count(photo_id) from photo_tags group by tag_id order by count(photo_id) desc) as tag_counts on tags.tag_id = tag_counts.tag_id	5 row(s) returned	0.0012 sec / 0.00001...
19:59:01	select * from users where id not in (select distinct user_id from photos) LIMIT 0, 1000	26 row(s) returned	0.0016 sec / 0.00004...
20:05:29	select tag_name from tags inner join (select tag_id, count(photo_id) from photo_tags group by tag_id order by count(photo_id) desc) as tag_counts on tags.tag_id = tag_counts.tag_id	5 row(s) returned	0.0015 sec / 0.00000...

5) List of users joining according to the day. 0 means Monday.



MySQL Workbench interface showing a query result for the 'ig_clone' schema. The query is: `select count(id) day_ from users group by day_order by count(id) desc LIMIT 0, 10;`

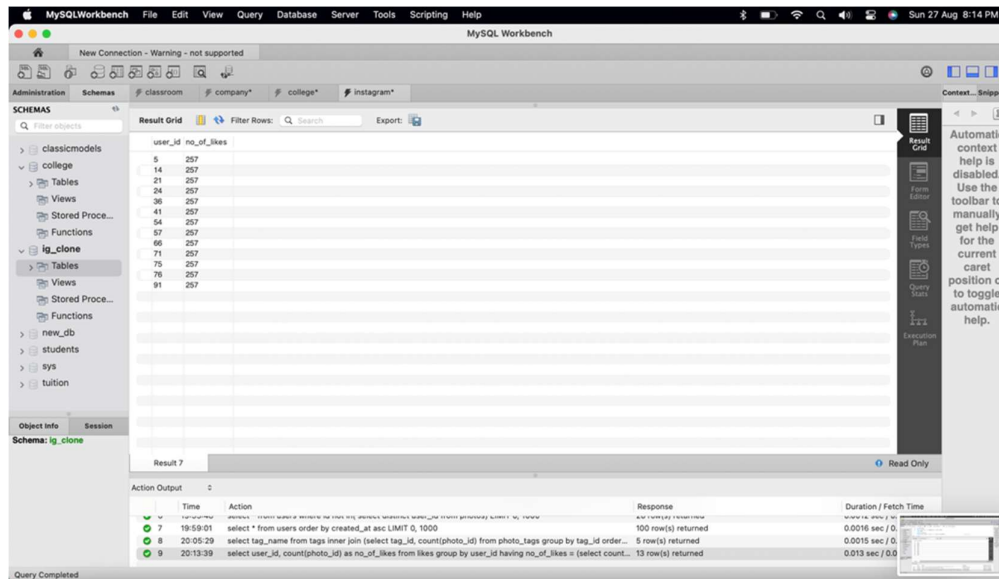
count(id)	day_
16	3
16	6
15	4
14	1
14	0
13	2
12	5

Result 9

Time	Action	Response	Duration / Fetch Time
20:13:39	select user_id, count(photo_id) as no_of_likes from likes group by user_id having no_of_likes = (select count...	13 row(s) returned	0.013 sec / 0.000010...
20:24:36	select * from users where id = (select user_id from photos where id = (select photo_id from(select photo_id...	1 row(s) returned	0.010 sec / 0.000008...
20:50:57	select count(id), weekday(created_at) as day_ from users group by day_order by count(id) desc LIMIT 0, 10...	7 row(s) returned	0.0031 sec / 0.00000...

6) The total number of photos is 257 and total number of users is 100. Average of photos per users is 2.57.

7) List of people who liked every single photo.



MySQL Workbench interface showing a query result for the 'ig_clone' schema. The query is: `select user_id, count(photo_id) as no_of_likes from likes group by user_id having no_of_likes = (select count...`

user_id	no_of_likes
5	257
14	257
21	257
24	257
36	257
41	257
54	257
57	257
66	257
71	257
75	257
78	257
91	257

Result 7

Time	Action	Response	Duration / Fetch Time
19:59:01	select * from users order by created_at asc LIMIT 0, 1000	100 row(s) returned	0.0016 sec / 0...
20:05:29	select tag_name from tags inner join (select tag_id, count(photo_id) from photo_tags group by tag_id order...	5 row(s) returned	0.0015 sec / 0...
20:13:39	select user_id, count(photo_id) as no_of_likes from likes group by user_id having no_of_likes = (select count...	13 row(s) returned	0.013 sec / 0.0...

Result:

A) Market Analysis:

1. The most loyal or five oldest users of Instagram are:

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	

2. The users who have never posted any photo are:

id	username	created_at	
5	Aniya_Hackett	2016-12-07 01:04:39	
7	Kasandra_Homenick	2016-12-12 06:50:08	
14	Jaclyn81	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	
74	Hulda.Macejkovic	2017-01-25 17:17:28	
75	Leslie67	2016-09-21 05:14:01	
76	Janelle.Nikolaus81	2016-07-21 09:26:09	
80	Darby_Herzog	2016-05-06 00:14:21	
81	Esther.Zulauf61	2017-01-14 17:02:34	
83	Bartholome.Bernhard	2016-11-06 02:31:23	
89	Jessyca_West	2016-09-14 23:47:05	
90	Esmeralda.Mraz57	2017-03-03 11:52:27	
91	Bethany20	2016-06-03 23:31:53	

3. The winner of contest of most likes on a single photo is:

id	username	created_at	
52	Zack_Kemmer93	2017-01-01 05:58:22	

4. The most popular hashtags are:

tag_name	Resets all sorted columns
smile	
beach	
party	
fun	
concert	

- The best days to launch ads on Instagram are Wednesdays and Sundays as most user registers on Wednesdays and Sundays. So, we can schedule our ad campaign on next Wednesday and Sunday that is 30th august and 3rd of September.

B) Investor Metrics:

- The average of posts per user on Instagram is 2.57
And total number of photos on Instagram are 257.
- The list of bots and fake accounts is given below:

user_id	no_of_likes	
5	257	
14	257	
21	257	
24	257	
36	257	
41	257	
54	257	
57	257	
66	257	
71	257	
75	257	
76	257	
91	257	