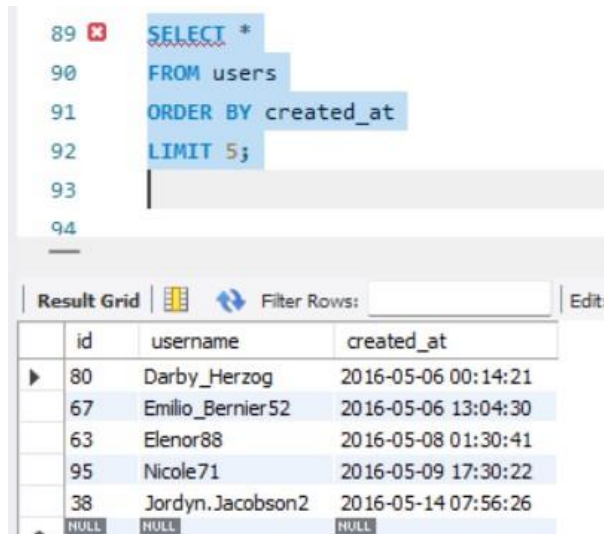



# Instagram User Analytics



## A) Marketing Analysis:

### 1. Loyal User Reward:

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



89  `SELECT *`  
90 `FROM users`  
91 `ORDER BY created_at`  
92 `LIMIT 5;`  
93  
94

Result Grid   Filter Rows:  Edit

	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:

```
SELECT users.id, users.username  
FROM users  
LEFT JOIN photos ON users.id = photos.user_id  
WHERE photos.id IS NULL;
```

```
90 SELECT users.id, users.username  
91 FROM users  
92 LEFT JOIN photos ON users.id = photos.user_id  
93 WHERE photos.id IS NULL;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Co
	id	username		
▶	5	Aniya_Hackett		
	7	Kassandra_Homenick		
	14	Jacyn81		
	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		
	75	Leslie67		
	76	Janelle.Nikolaus81		
	80	Darby_Herzog		
	81	Esther.Zulauf61		
	83	Bartholome.Bernhard		
	89	Jessyca_West		
	90	Esmeralda.Mraz57		
	91	Bethany20		

### 3. Contest Winner Declaration

```
SELECT users.id AS user_id, users.username, MAX(likes_count) AS max_likes
FROM users
JOIN photos ON users.id = photos.user_id
LEFT JOIN (
    SELECT photo_id, COUNT(*) AS likes_count
    FROM likes
    GROUP BY photo_id
) AS photo_likes ON photos.id = photo_likes.photo_id
GROUP BY users.id, users.username
ORDER BY max_likes DESC
LIMIT 1;
```

```
90 SELECT users.id AS user_id, users.username, MAX(likes_count) AS max_likes
91 FROM users
92 JOIN photos ON users.id = photos.user_id
93 LEFT JOIN (
94     SELECT photo_id, COUNT(*) AS likes_count
95     FROM likes
96     GROUP BY photo_id
97 ) AS photo_likes ON photos.id = photo_likes.photo_id
98 GROUP BY users.id, users.username
99 ORDER BY max_likes DESC
100 LIMIT 1;
101
102
103
```

Result Grid			
Filter Rows:			
Export: Wrap Cell Content: Fetch rows:			
	user_id	username	max_likes
▶	52	Zack_Kemmer93	48

#### 4. Hashtag Research:

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

```
90 ✖ SELECT tags.tag_name, COUNT(*) AS tag_count
91 FROM tags
92 JOIN photo_tags ON tags.id = photo_tags.tag_id
93 GROUP BY tags.tag_name
94 ORDER BY tag_count DESC
95 LIMIT 5;
```

100

Result Grid		Filter Rows:	Export:	Wrap Cell C
tag_name	tag_count			
smile	59			
beach	42			
party	39			
fun	38			
concert	24			

## 5. Ad Campaign Launch:

```
SELECT DAYNAME(created_at) AS registration_day, COUNT(*) AS registration_count
FROM users
GROUP BY registration_day
ORDER BY registration_count DESC
LIMIT 1;
```

```
90 SELECT DAYNAME(created_at) AS registration_day, COUNT(*) AS registration_count
91 FROM users
92 GROUP BY registration_day
93 ORDER BY registration_count DESC
94 LIMIT 1;
```

95  
96  
97  
98  
99  
100

Result Grid			Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	registration_day	registration_count				
▶	Thursday	16				

## B) Investor Metrics:

### 1. User Engagement:

- Average number of Posts per user

```
SELECT AVG(posts_per_user) AS average_posts_per_user
FROM (
  SELECT user_id, COUNT(*) AS posts_per_user
  FROM photos
  GROUP BY user_id
) AS user_posts;
```

The screenshot shows a SQL query editor with line numbers 90 to 96. The query is: `SELECT AVG(posts_per_user) AS average_posts_per_user FROM ( SELECT user_id, COUNT(*) AS posts_per_user FROM photos GROUP BY user_id ) AS user_posts;`. Below the editor is a toolbar with 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content' buttons. The 'Result Grid' is active, showing a table with one column 'average\_posts\_per\_user' and one row with the value '3.4730'.

average_posts_per_user
3.4730

- Total number of photos on Instagram divided by the total number of users

```
SELECT COUNT(*) AS total_photos, COUNT(DISTINCT user_id) AS total_users,  
       COUNT(*) / COUNT(DISTINCT user_id) AS photos_per_user_ratio  
FROM photos;
```

```
90 SELECT COUNT(*) AS total_photos, COUNT(DISTINCT user_id) AS total_users,  
91     COUNT(*) / COUNT(DISTINCT user_id) AS photos_per_user_ratio  
92 FROM photos;  
93  
94  
95  
96  
97  
98  
99  
100
```

Result Grid | | Filter Rows:  | Export: | Wrap Cell Content:

	total_photos	total_users	photos_per_user_ratio
▶	257	74	3.4730

## 2. Bots & Fake Accounts:

```
SELECT user_id
FROM (
    SELECT DISTINCT user_id, photo_id
    FROM likes
) AS unique_likes
GROUP BY user_id
HAVING COUNT(DISTINCT photo_id) = (SELECT COUNT(*) FROM photos);
```

```
91 SELECT user_id
92 FROM (
93     SELECT DISTINCT user_id, photo_id
94     FROM likes
95 ) AS unique_likes
96 GROUP BY user_id
97 HAVING COUNT(DISTINCT photo_id) = (SELECT COUNT(*) FROM photos);
or
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	user_id			
▶	5			
	14			
	21			
	24			
	36			
	41			
	54			
	57			
	66			
	71			
	75			
	76			
	91			