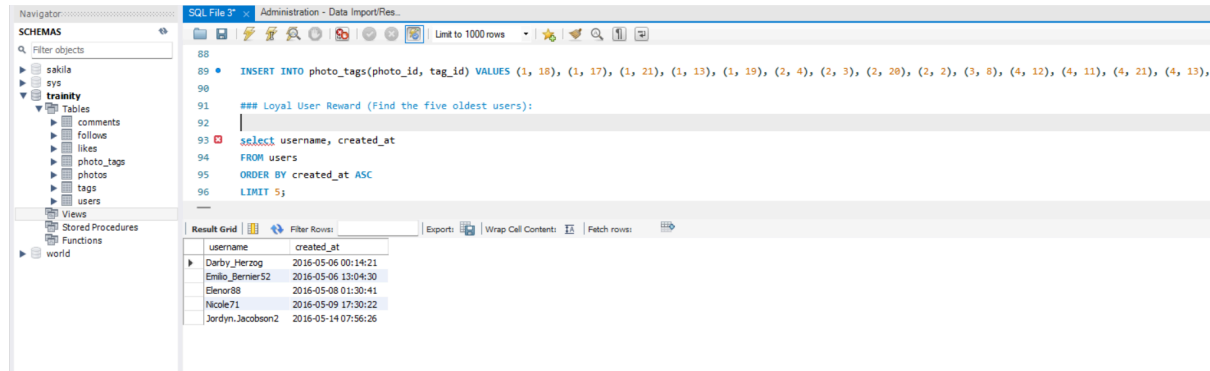


## Loyal User Reward (Find the five oldest users):

```
SELECT username, created_at
FROM users
ORDER BY created_at ASC
LIMIT 5;
```



The screenshot shows a SQL IDE interface with a schema tree on the left and a query editor on the right. The query editor contains the following SQL code:

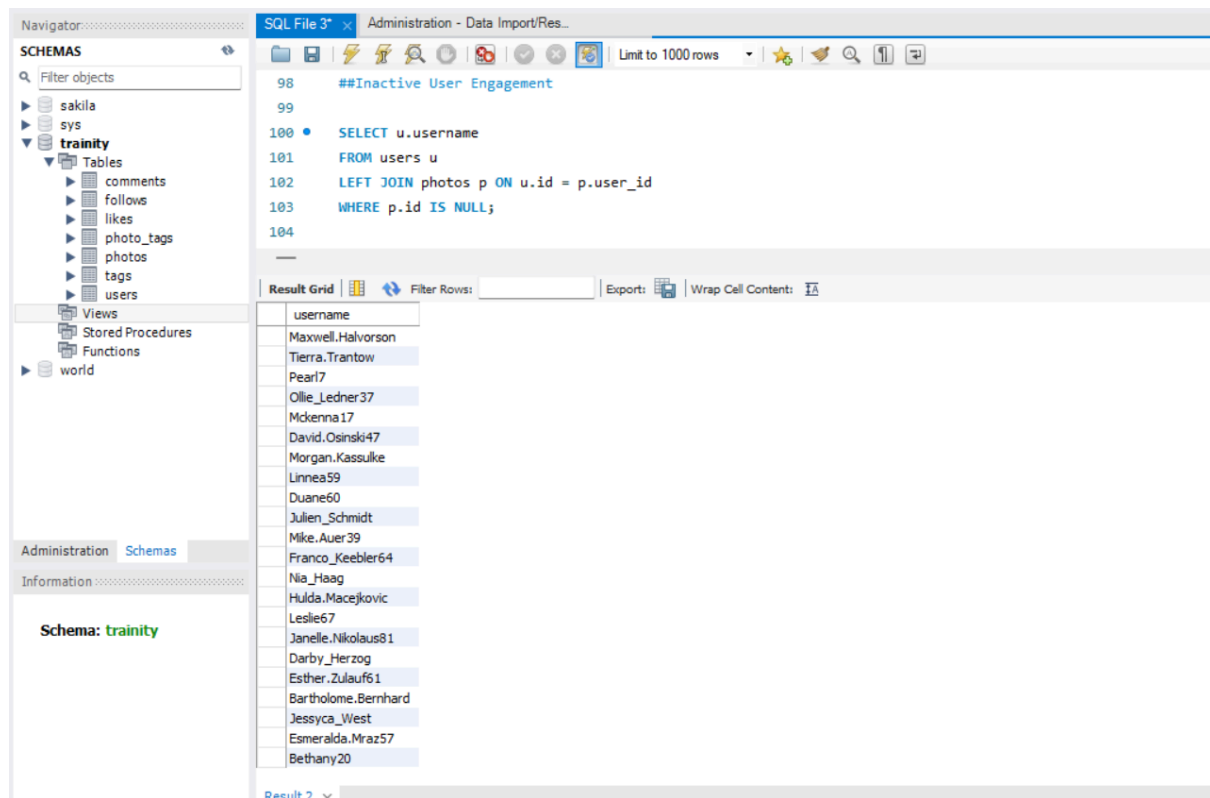
```
88 INSERT INTO photo_tags(photo_id, tag_id) VALUES (1, 18), (1, 17), (1, 21), (1, 13), (1, 19), (2, 4), (2, 3), (2, 20), (2, 2), (3, 8), (4, 12), (4, 11), (4, 21), (4, 13),
89
90
91 ### Loyal User Reward (Find the five oldest users):
92
93 select username, created_at
94 from users
95 order by created_at asc
96 limit 5;
```

The result grid below the query shows the following data:

| username         | created_at          |
|------------------|---------------------|
| Darby_Herzog     | 2016-05-06 00:14:21 |
| Emilio_Bernier52 | 2016-05-06 13:04:30 |
| Eleanor88        | 2016-05-08 01:30:41 |
| Nicole71         | 2016-05-09 17:30:22 |
| Jordyn.Jacobson2 | 2016-05-14 07:56:26 |

## Inactive User Engagement

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



The screenshot shows a SQL IDE interface with a schema tree on the left and a query editor on the right. The query editor contains the following SQL code:

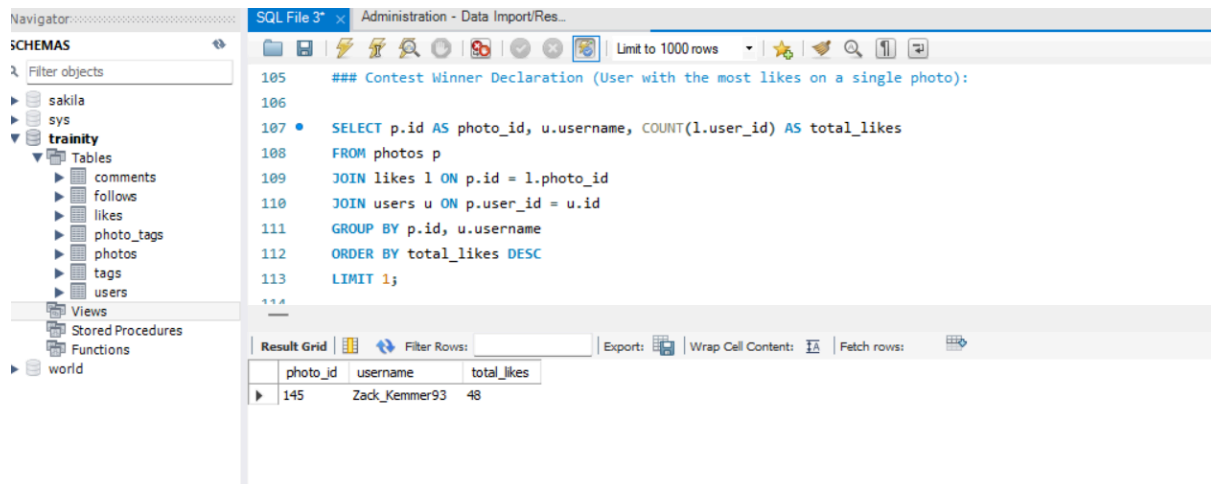
```
98 ##Inactive User Engagement
99
100 select u.username
101 from users u
102 left join photos p on u.id = p.user_id
103 where p.id is null;
104
```

The result grid below the query shows the following data:

| username            |
|---------------------|
| Maxwell.Halvorsen   |
| Tierra.Trantow      |
| Pearl7              |
| Ollie_Ledner37      |
| Mckenna17           |
| David.Osinski47     |
| Morgan.Kassulke     |
| Linnea59            |
| Duane60             |
| Julien_Schmidt      |
| Mike.Auer39         |
| Franco_Keebler64    |
| Nia_Haag            |
| Hulda.Macejkovic    |
| Leslie67            |
| Janelle.Nikolaus81  |
| Darby_Herzog        |
| Esther.Zulauf61     |
| Bartholome.Bernhard |
| Jessyca_West        |
| Esmeralda.Mraz57    |
| Bethany20           |

### Contest Winner Declaration (User with the most likes on a single photo)

```
SELECT p.id AS photo_id, u.username, COUNT(l.user_id) AS total_likes
FROM photos p
JOIN likes l ON p.id = l.photo_id
JOIN users u ON p.user_id = u.id
GROUP BY p.id, u.username
ORDER BY total_likes DESC
LIMIT 1;
```



The screenshot shows a SQL IDE interface with a left-hand sidebar displaying a database schema tree. The tree includes schemas 'sakila', 'sys', and 'trainity'. Under 'trainity', there are tables 'comments', 'follows', 'likes', 'photo\_tags', 'photos', 'tags', and 'users'. The main window displays a SQL query titled 'Contest Winner Declaration (User with the most likes on a single photo):'. The query is as follows:

```
105  ## Contest Winner Declaration (User with the most likes on a single photo):
106
107  SELECT p.id AS photo_id, u.username, COUNT(l.user_id) AS total_likes
108  FROM photos p
109  JOIN likes l ON p.id = l.photo_id
110  JOIN users u ON p.user_id = u.id
111  GROUP BY p.id, u.username
112  ORDER BY total_likes DESC
113  LIMIT 1;
```

Below the query editor, the 'Result Grid' is visible, showing the results of the query:

| photo_id | username      | total_likes |
|----------|---------------|-------------|
| 145      | Zack_Kemmer93 | 48          |

### Hashtag Research (Top five most commonly used hashtags)

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

Navigator: SCHEMAS Filter objects

- sakila
- sys
- trainity
  - Tables
    - comments
    - follows
    - likes
    - photo\_tags
    - photos
    - tags
    - users
  - Views
  - Stored Procedures
  - Functions
- world

SQL File 3\* Administration - Data Import/Res...

Limit to 1000 rows

```

115  ## Hashtag Research (Top five most commonly used hashtags):
116
117  • SELECT t.tag_name, COUNT(pt.tag_id) AS usage_count
118  FROM tags t
119  JOIN photo_tags pt ON t.id = pt.tag_id
120  GROUP BY t.tag_name
121  ORDER BY usage_count DESC
122  LIMIT 5;
123
124

```

Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows:

| tag_name | usage_count |
|----------|-------------|
| smile    | 59          |
| beach    | 42          |
| party    | 39          |
| fun      | 38          |
| concert  | 24          |

### Ad Campaign Launch ( day for launching ads based on registrations)

SELECT DAYNAME(created\_at) AS day\_of\_week, COUNT(id) AS user\_count

FROM users

GROUP BY day\_of\_week

ORDER BY user\_count DESC

LIMIT 1;

Navigator: SCHEMAS Filter objects

- sakila
- sys
- trainity
  - Tables
    - comments
    - follows
    - likes
    - photo\_tags
    - photos
    - tags
    - users
  - Views
  - Stored Procedures
  - Functions
- world

SQL File 3\* Administration - Data Import/Res...

Limit to 1000 rows

```

122  LIMIT 5;
123
124  ## Ad Campaign Launch ( day for launching ads based on registrations)
125
126  • SELECT DAYNAME(created_at) AS day_of_week, COUNT(id) AS user_count
127  FROM users
128  GROUP BY day_of_week
129  ORDER BY user_count DESC
130  LIMIT 1;
131
132

```

Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows:

| day_of_week | user_count |
|-------------|------------|
| Thursday    | 16         |

## User Engagement (Average number of posts per user and total posts divided by total users)

-- Average posts per user:

```
SELECT AVG(post_count) AS avg_posts_per_user
```

```
FROM (
```

```
    SELECT u.id, COUNT(p.id) AS post_count
```

```
    FROM users u
```

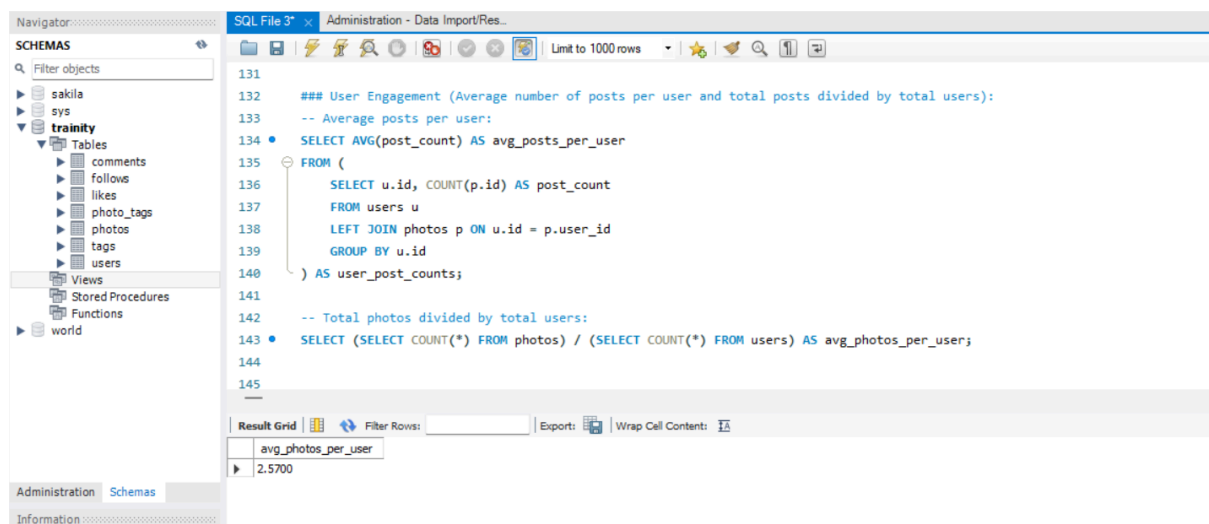
```
    LEFT JOIN photos p ON u.id = p.user_id
```

```
    GROUP BY u.id
```

```
) AS user_post_counts;
```

-- Total photos divided by total users:

```
SELECT (SELECT COUNT(*) FROM photos) / (SELECT COUNT(*) FROM users) AS  
avg_photos_per_user;
```



## Bots & Fake Accounts (Users who have liked every single photo)

```
SELECT u.username
```

```

FROM users u

WHERE NOT EXISTS (

    SELECT p.id

    FROM photos p

    WHERE NOT EXISTS (

        SELECT 1

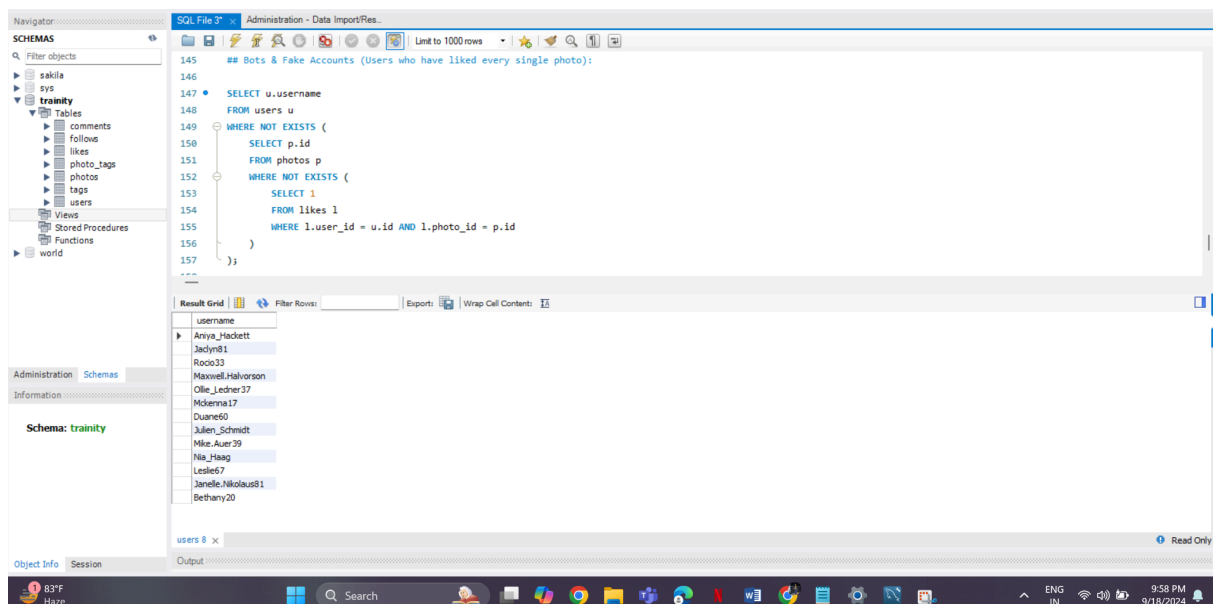
        FROM likes l

        WHERE l.user_id = u.id AND l.photo_id = p.id

    )

);

```



## Project Description >

This project aims to perform marketing and investor analysis using an Instagram-like platform's database. The goal is to provide actionable insights such as identifying loyal users, inactive users, contest winners, popular hashtags, and more. The tasks also extend to analysing metrics that can help investors gauge user engagement and the prevalence of bots on the platform. SQL queries will be used to extract this information from a pre-configured database.

## Approach

1. Data Understanding: First, I examined the schema and structure of the database, including tables for users, photos, likes, comments, hashtags, and follows. This allowed me to identify the relationships between different entities such as users, posts, and likes.

2. Data Retrieval: I executed SQL queries to answer each task based on the relationships between the tables. For instance:

- Used `JOIN` operations to combine user and photo data to find inactive users.
- Aggregated data using `GROUP BY` and `COUNT` to identify contest winners and top hashtags.
- Used conditional subqueries to detect potential bot accounts.

3. Data Analysis: For each question, the results were reviewed to extract meaningful insights, such as determining the most effective time to launch ad campaigns based on user registration data.

4. Execution: I used SQL to execute the queries, ensuring accuracy by testing each query with sample data from the provided database.

## Tech-Stack Used

MySQL Workbench (Version: 8.0): I chose this tool due to its user-friendly interface and efficient execution of SQL queries. It supports the visualization of data relationships through ER diagrams, which helps in better understanding the schema.

MySQL Database: MySQL is a powerful open-source relational database management system that allows for the efficient handling of structured data and complex queries.

## Insights

1. Loyal Users: The oldest five users were identified based on their registration date, providing a list of long-time active users who could be rewarded.

2. Inactive Users: Users who have never posted a single photo were flagged, offering an opportunity for engagement through targeted promotions.

3. Contest Winner: The user with the most likes on a single photo was identified, making it easier for the marketing team to declare the winner.

4. Popular Hashtags: The most frequently used hashtags were determined, providing valuable input to the brand partners for their social media strategy.

5. Ad Campaign Insights: The best day of the week for user registrations was identified, suggesting the optimal time to launch marketing campaigns.

## **Result**

**Achievements:** The project provided clear and actionable insights into user behavior and engagement. The analysis helped in identifying areas for growth, such as targeting inactive users and refining ad campaign timings.

**Impact:** By analysing the database using SQL, the project delivered meaningful business metrics that could assist both marketing teams and investors in making data-driven decisions.