

Tuesday, 16th May 2023

Trainity

STUDENT

MARJIBA

marjibajamir9@gmail.com

INSTAGRAM USER ANALYTICS

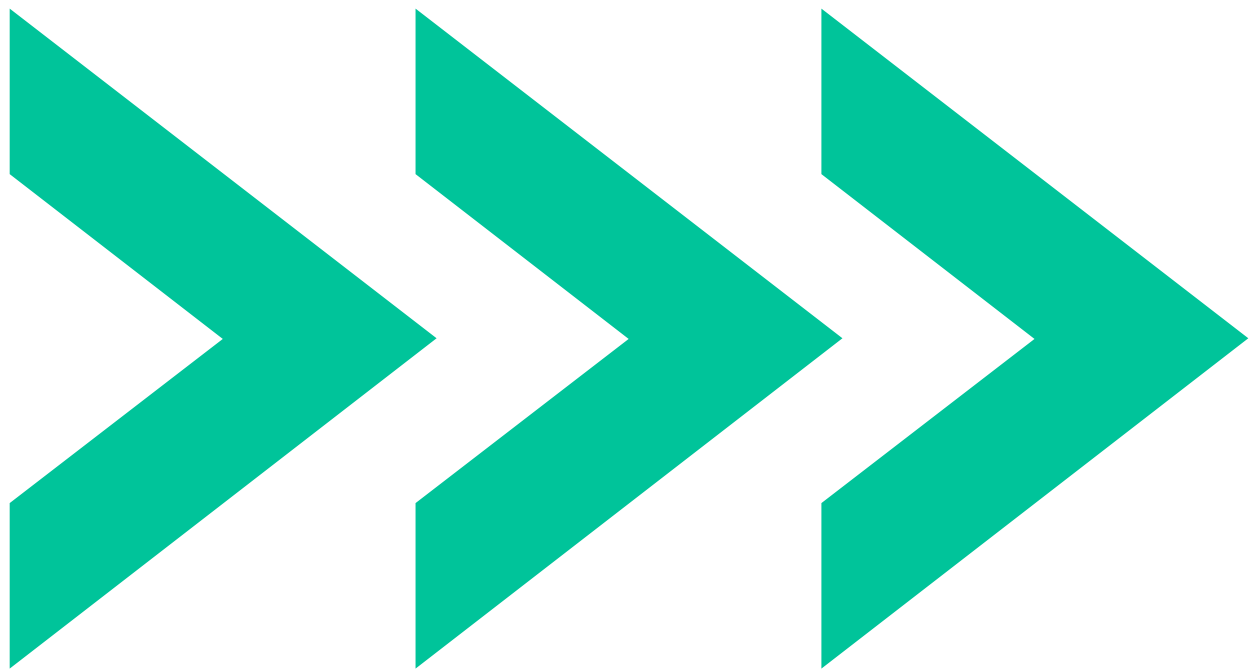


Table of
Contents

Page

I	Problem Statement	3
II	MARKETING	7
1	Rewarding Most Loyal Users:	7
2	Remind Inactive Users to Start Posting:	8
3	Declaring Contest Winner:	10
4	Hashtag Researching:	13
5	Launch AD Campaign:	16
III	INVESTOR METRICS	18
1	User Engagement:	18
2	Bots & Fake Accounts:	20
IV	INSIGHTS	21
V	RESULT	23

I Problem Statement

Project Description:

This project aims to address marketing and investor concerns related to Instagram. The project will handle various tasks such as identifying the oldest users, reminding inactive users to start posting, declaring contest winners, researching popular hashtags, determining the best day to launch ad campaigns, assessing user engagement, and investigating the presence of bots and fake accounts. Through this project, valuable insights and recommendations will be provided to enhance marketing strategies and evaluate Instagram's performance.

Approach:

The project utilizes SQL Workbench as the tech-stack to handle the data analysis tasks. The provided database will be queried to identify the oldest users, users who have never posted, contest winners, popular hashtags, and user engagement metrics. Additionally, registration patterns will be analyzed to determine the optimal day for ad campaign launches. The presence of bots and fake accounts will be assessed by identifying users who have liked every single photo. SQL Workbench provides the necessary functionality for efficient data querying and analysis, making it an ideal choice for this project.

Tech Stack used:

Tech-Stack Used: SQL Workbench SQL Workbench is used for data querying and analysis in this project. Its version 8.0 CE was employed to extract relevant information from the provided database. SQL Workbench is a widely used tool for managing and working with relational databases. Its purpose in this project is to perform queries and operations on the Instagram database to extract insights, identify trends, and provide the required information for marketing and investor metrics.

Brief overview of what we are going to do next:

This project focuses on addressing marketing and investor concerns for Instagram. It is divided into two main sections: Marketing and Investor Metrics.

A) Marketing:

1. Rewarding Most Loyal Users:

- Task: Identify the 5 oldest users on Instagram.

2. Remind Inactive Users to Start Posting:

- Task: Find users who have never posted a photo and send them promotional emails to encourage their first post.

3. Declaring Contest Winner:

- Task: Identify the contest winner based on the user with the most likes on a single photo.

4. Hashtag Researching:

- Task: Identify the top 5 most commonly used hashtags on Instagram.

5. Launch AD Campaign:

- Task: Determine the best day of the week to launch ADs based on user registration patterns.

B) Investor Metrics:

1. User Engagement:

- Task: Provide the average number of posts made by Instagram users and calculate the ratio of total photos to total users.

2. Bots & Fake Accounts:

- Task: Gather data on users (bots) who have liked every single photo on Instagram.

This project aims to provide valuable insights and recommendations to enhance Instagram's marketing strategies and assess its performance for investors.

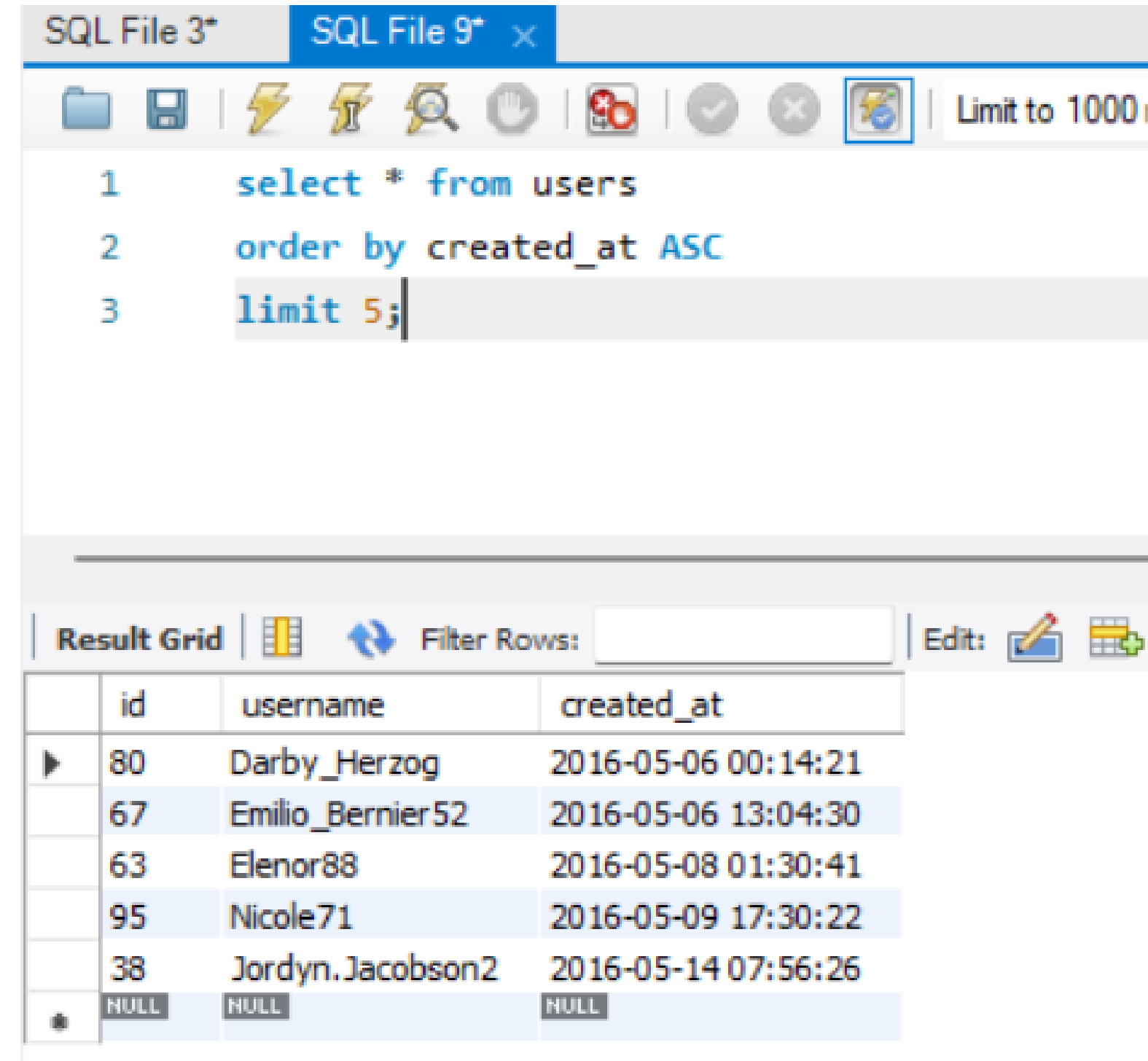
II MARKETING

- **Rewarding Most Loyal Users:** People who have been using the platform for the longest time.

Find the 5 oldest users of the Instagram from the database provided

To find the 5 oldest users of Instagram, we can look at the date of account creation for each user in the database. By sorting the users based on their account creation date, we can identify the 5 oldest users.

```
SELECT *  
FROM users_table_name  
ORDER BY account_creation_date_column  
LIMIT 5;
```



The screenshot shows a SQL IDE interface with two tabs: 'SQL File 3*' and 'SQL File 9*'. The active tab 'SQL File 9*' contains the following SQL query:

```
1 select * from users  
2 order by created_at ASC  
3 limit 5;
```

Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The grid has four columns: 'id', 'username', and 'created_at'. The results are as follows:

	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

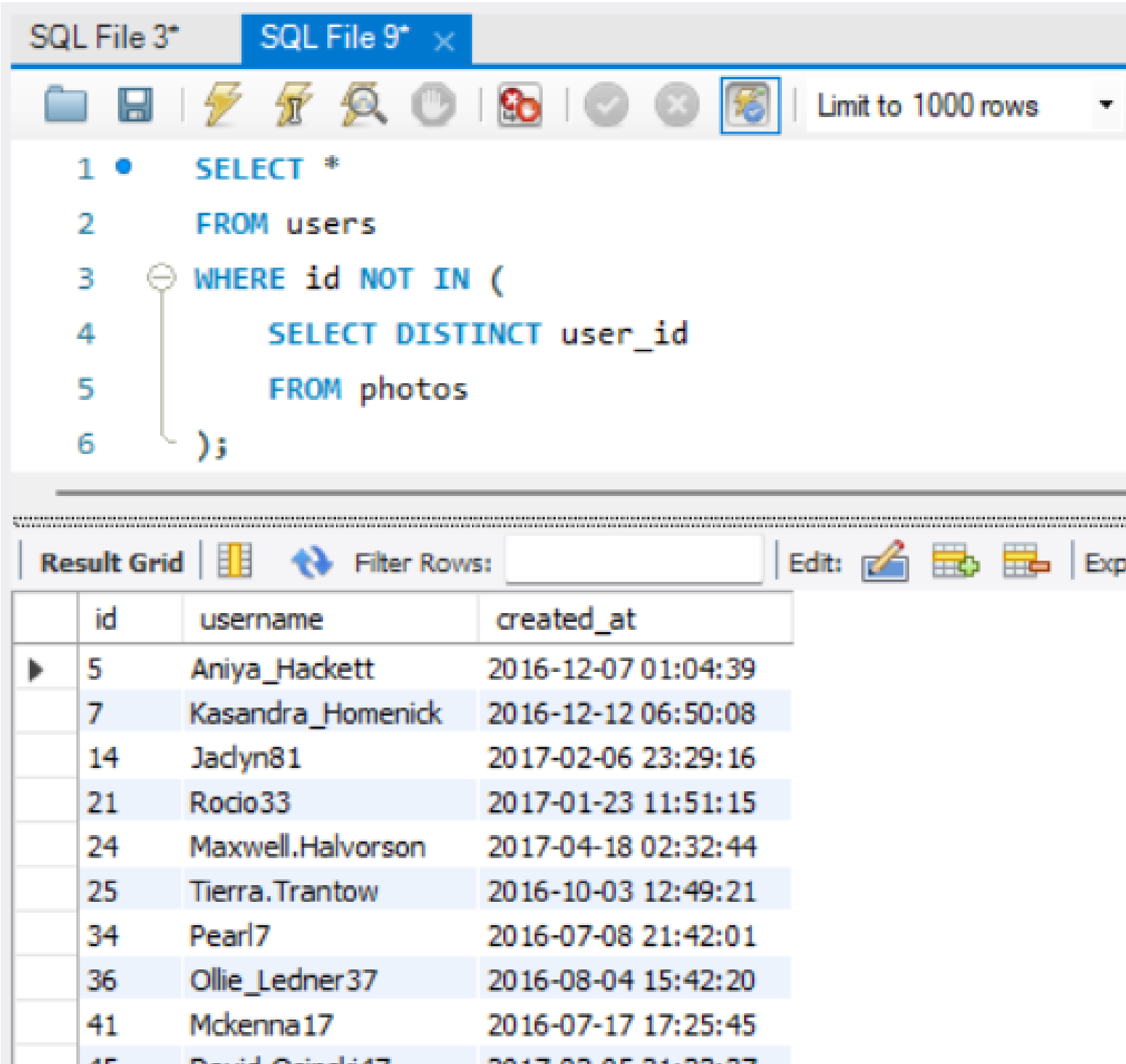
- **Remind Inactive Users to Start Posting:** By sending the promotional emails to post their 1st photo.

Find the users who have never posted a single photo on Instagram

To find the users who have never posted a single photo on Instagram, we need to analyze the database and identify users who do not have any photo uploads associated with their accounts. By filtering out these users, we can compile a list of inactive users who have not posted any photos.

```
SELECT *
FROM users
WHERE user_id NOT IN (
    SELECT DISTINCT user_id
    FROM photos
);
```

Same user have posted multiple photos, hence we use Distinct keyword



- Before we proceed let us have a look at some tables on which we will perform queries

users table

The screenshot shows a SQL query editor with the query `select * from users;` and a result grid below it. The result grid contains 17 rows of user data.

	id	username	created_at
▶	1	Kenton_Kirlin	2017-02-16 18:22:11
	2	Andre_Purdy85	2017-04-02 17:11:21
	3	Harley_Lind18	2017-02-21 11:12:33
	4	Arely_Bogan63	2016-08-13 01:28:43
	5	Aniya_Hackett	2016-12-07 01:04:39
	6	Travon.Waters	2017-04-30 13:26:14
	7	Kasandra_Homenick	2016-12-12 06:50:08
	8	Tabitha_Schamberger11	2016-08-20 02:19:46
	9	Gus93	2016-06-24 19:36:31
	10	Presley_McClure	2016-08-07 16:25:49
	11	Justina.Gaylord27	2017-05-04 16:32:16
	12	Dereck65	2017-01-19 01:34:14
	13	Alexandro35	2017-03-29 17:09:02
	14	Jacyn81	2017-02-06 23:29:16
	15	Billy52	2016-10-05 14:10:20
	16	Annalise.McKenzie16	2016-08-02 21:32:46
	17	Norbert_Carroll35	2017-02-06 22:05:43

photos table

The screenshot shows a SQL query editor with the query `select * from photos;` and a result grid below it. The result grid contains 17 rows of photo data.

	id	image_url	user_id	created_at
▶	1	http://elijah.biz	1	2023-05-16 16:25:28
	2	https://shanon.org	1	2023-05-16 16:25:28
	3	http://vicky.biz	1	2023-05-16 16:25:28
	4	http://oleta.net	1	2023-05-16 16:25:28
	5	https://jennings.biz	1	2023-05-16 16:25:28
	6	https://quinn.biz	2	2023-05-16 16:25:28
	7	https://selina.name	2	2023-05-16 16:25:28
	8	http://malvina.org	2	2023-05-16 16:25:28
	9	https://branson.biz	2	2023-05-16 16:25:28
	10	https://elenor.name	3	2023-05-16 16:25:28
	11	https://marcelino.com	3	2023-05-16 16:25:28
	12	http://felicity.name	3	2023-05-16 16:25:28
	13	https://fred.com	3	2023-05-16 16:25:28
	14	https://gerhard.biz	4	2023-05-16 16:25:28
	15	https://sherwood.net	4	2023-05-16 16:25:28
	16	https://maudie.org	4	2023-05-16 16:25:28
	17	http://annamae.name	6	2023-05-16 16:25:28

likes table

The screenshot shows a SQL query editor with the query `select * from likes;` and a result grid below it. The result grid contains 17 rows of like data.

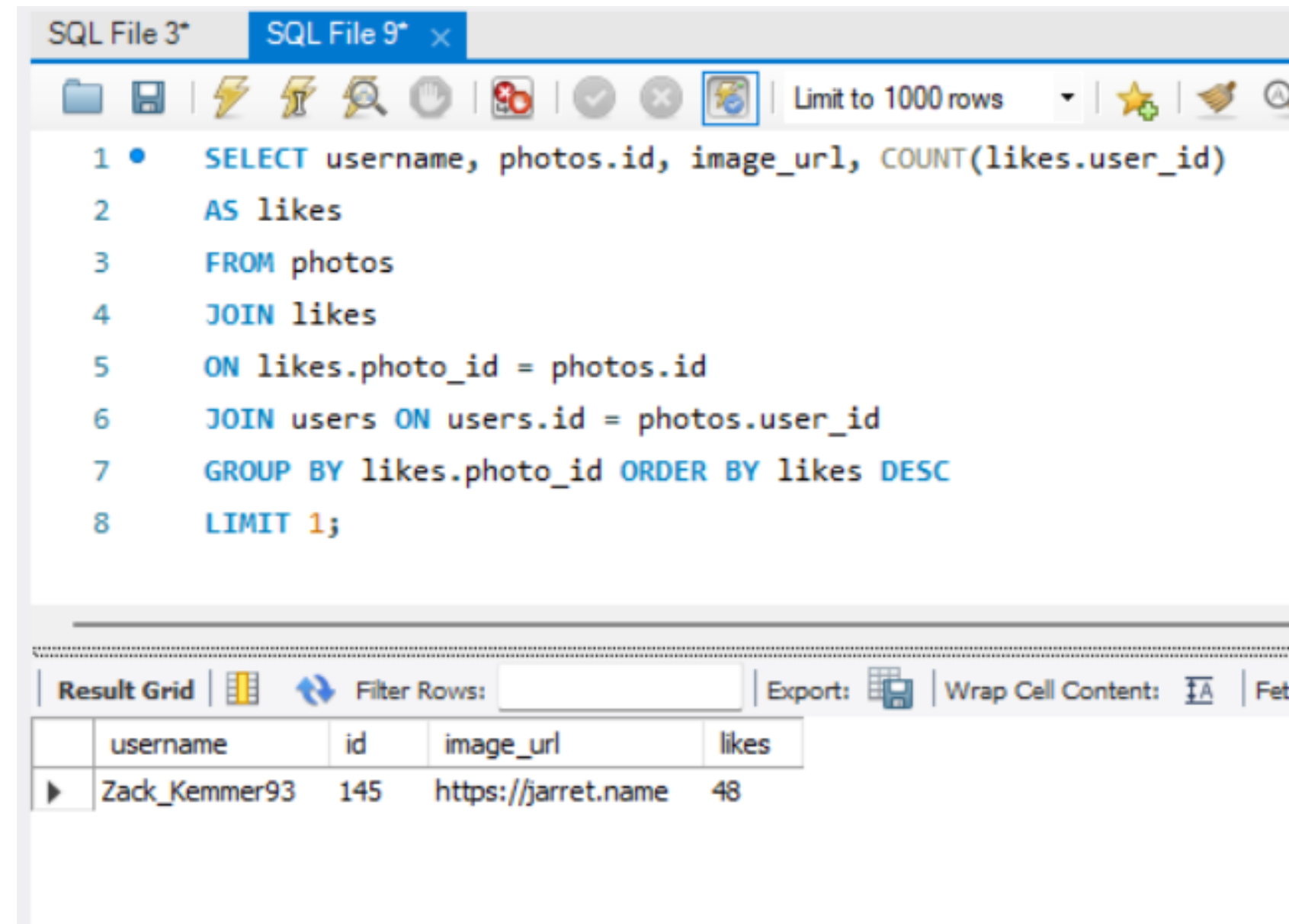
	user_id	photo_id	created_at
▶	2	1	2023-05-16 16:25:29
	2	4	2023-05-16 16:25:29
	2	8	2023-05-16 16:25:29
	2	9	2023-05-16 16:25:29
	2	10	2023-05-16 16:25:29
	2	11	2023-05-16 16:25:29
	2	12	2023-05-16 16:25:29
	2	13	2023-05-16 16:25:29
	2	15	2023-05-16 16:25:29
	2	23	2023-05-16 16:25:29
	2	25	2023-05-16 16:25:29
	2	27	2023-05-16 16:25:29
	2	30	2023-05-16 16:25:29
	2	34	2023-05-16 16:25:29
	2	36	2023-05-16 16:25:29
	2	39	2023-05-16 16:25:29
	2	41	2023-05-16 16:25:29

- 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.

Identify the winner of the contest and provide their details to the team

To identify the winner of the contest who received the most likes on a single photo, we need to analyze the contest entries and determine the photo with the highest number of likes. Once we have identified the photo, we can retrieve the details of the user who posted it and declare them as the winner.

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



The screenshot shows a SQL IDE window with two tabs: 'SQL File 3*' and 'SQL File 9*'. The active tab 'SQL File 9*' contains the following SQL query:

```
1 • SELECT username, photos.id, image_url, COUNT(likes.user_id)  
2 AS likes  
3 FROM photos  
4 JOIN likes  
5 ON likes.photo_id = photos.id  
6 JOIN users ON users.id = photos.user_id  
7 GROUP BY likes.photo_id ORDER BY likes DESC  
8 LIMIT 1;
```

Below the query editor, the 'Result Grid' tab is active, displaying the results of the query in a table format:

username	id	image_url	likes
Zack_Kemmer93	145	https://jarret.name	48

The IDE interface includes various icons for file operations, execution, and settings. A 'Limit to 1000 rows' dropdown is visible in the top right corner of the query editor.

- Lets Break down the query to see how we got our result:

```
SELECT username, photos.id, image_url,  
COUNT(likes.user_id)  
AS likes
```

We retrieve the username from the users table, the photo ID and image URL from the photos table, and the count of likes from the likes table. We use the AS keyword to alias the count of likes as "likes" for better readability in the result.

```
FROM photos
```

This specifies the table from which we are selecting data, in this case, the photos table.

```
JOIN likes  
ON likes.photo_id = photos.id
```

This is a join condition that links the photos table with the likes table.

```
JOIN users ON users.id = photos.user_id
```

This is another join condition that links the photos table with the users table.

```
GROUP BY likes.photo_id
```

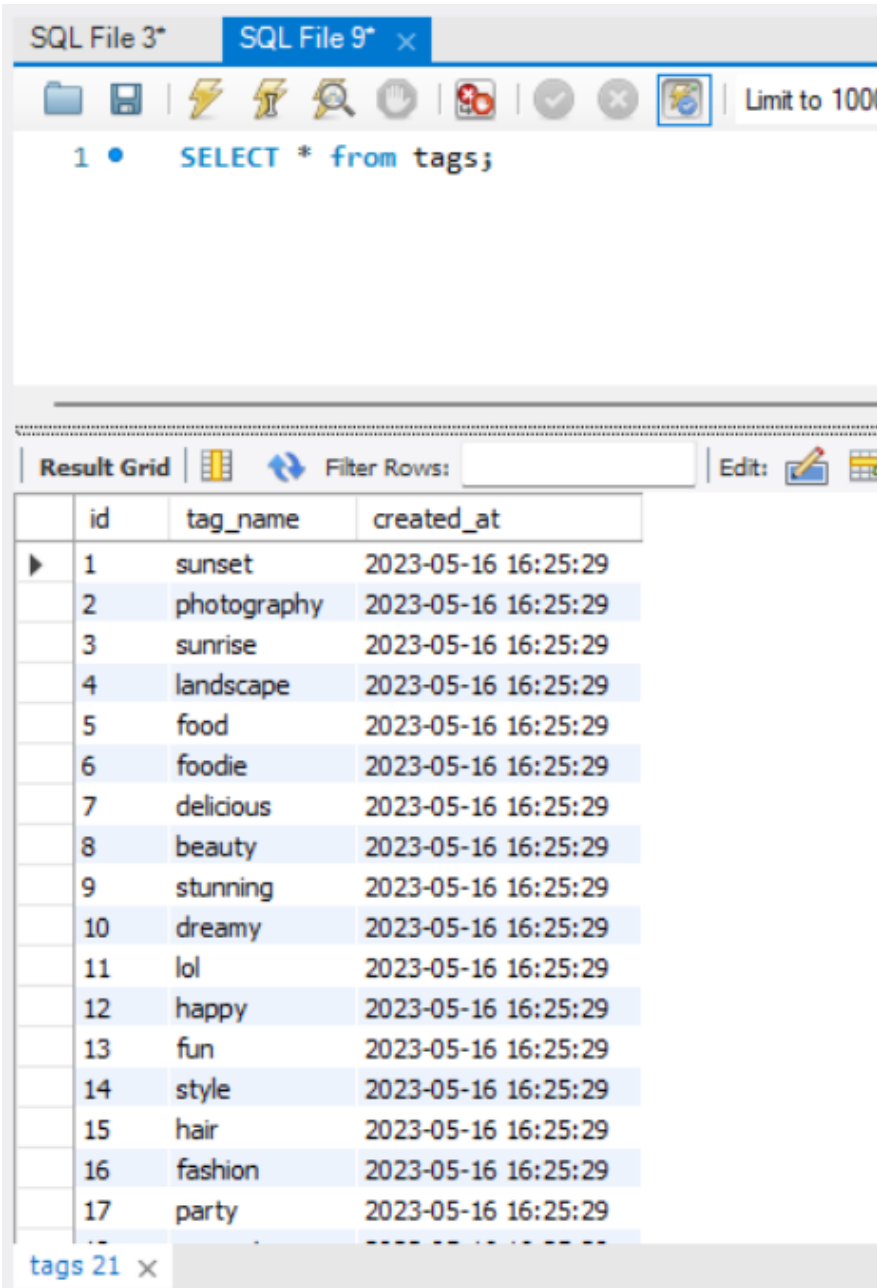
This groups the result by the photo_id column from the likes table. It ensures that the count of likes is calculated per photo.

```
ORDER BY likes DESC  
LIMIT 1;
```

This orders the result in descending order based on the count of likes.

- Before we proceed let us have a look at some more tables we will be using

tags table



SQL File 3* SQL File 9* x

1 • `SELECT * from tags;`

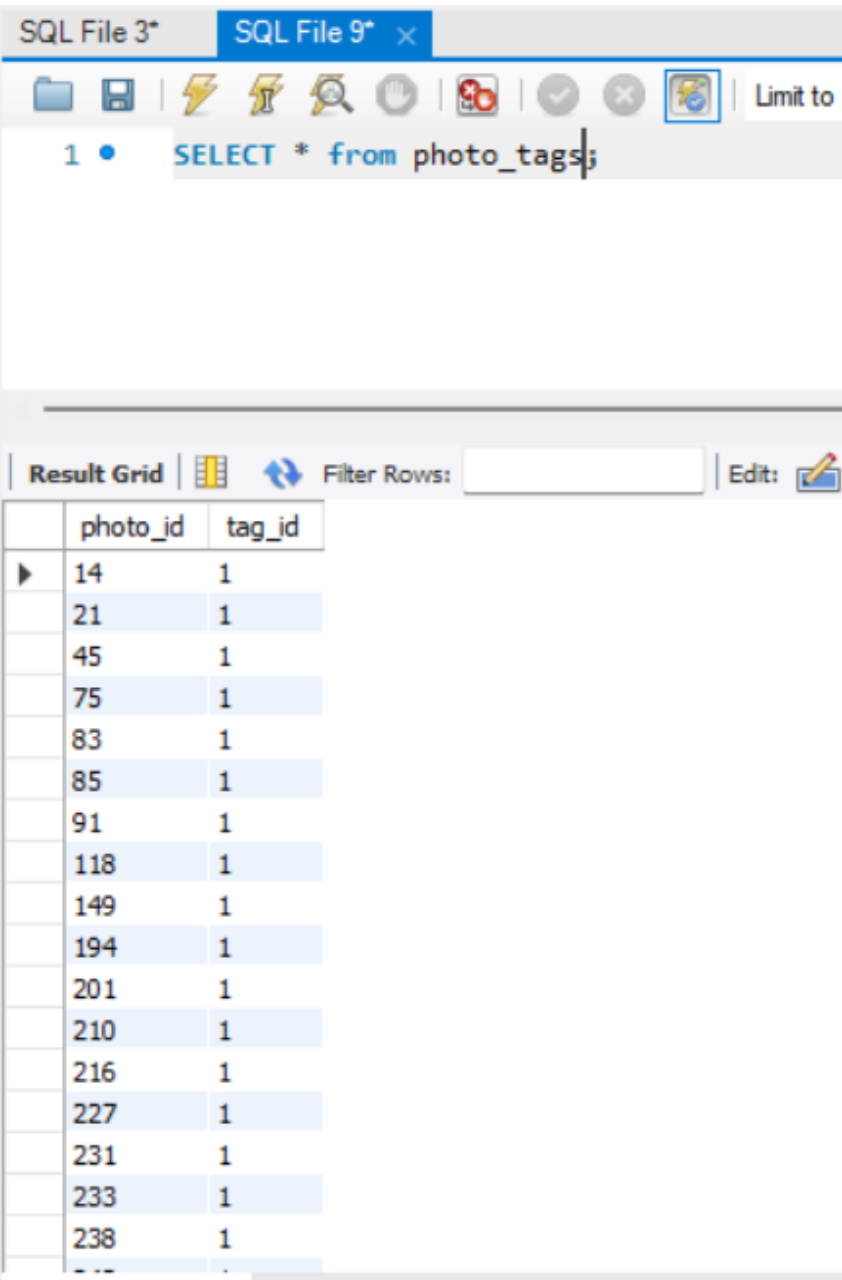
Limit to 1000

Result Grid Filter Rows: Edit:

	id	tag_name	created_at
▶	1	sunset	2023-05-16 16:25:29
	2	photography	2023-05-16 16:25:29
	3	sunrise	2023-05-16 16:25:29
	4	landscape	2023-05-16 16:25:29
	5	food	2023-05-16 16:25:29
	6	foodie	2023-05-16 16:25:29
	7	delicious	2023-05-16 16:25:29
	8	beauty	2023-05-16 16:25:29
	9	stunning	2023-05-16 16:25:29
	10	dreamy	2023-05-16 16:25:29
	11	lol	2023-05-16 16:25:29
	12	happy	2023-05-16 16:25:29
	13	fun	2023-05-16 16:25:29
	14	style	2023-05-16 16:25:29
	15	hair	2023-05-16 16:25:29
	16	fashion	2023-05-16 16:25:29
	17	party	2023-05-16 16:25:29

tags 21 x

photo_tags table



SQL File 3* SQL File 9* x

1 • `SELECT * from photo_tags;`

Limit to

Result Grid Filter Rows: Edit:

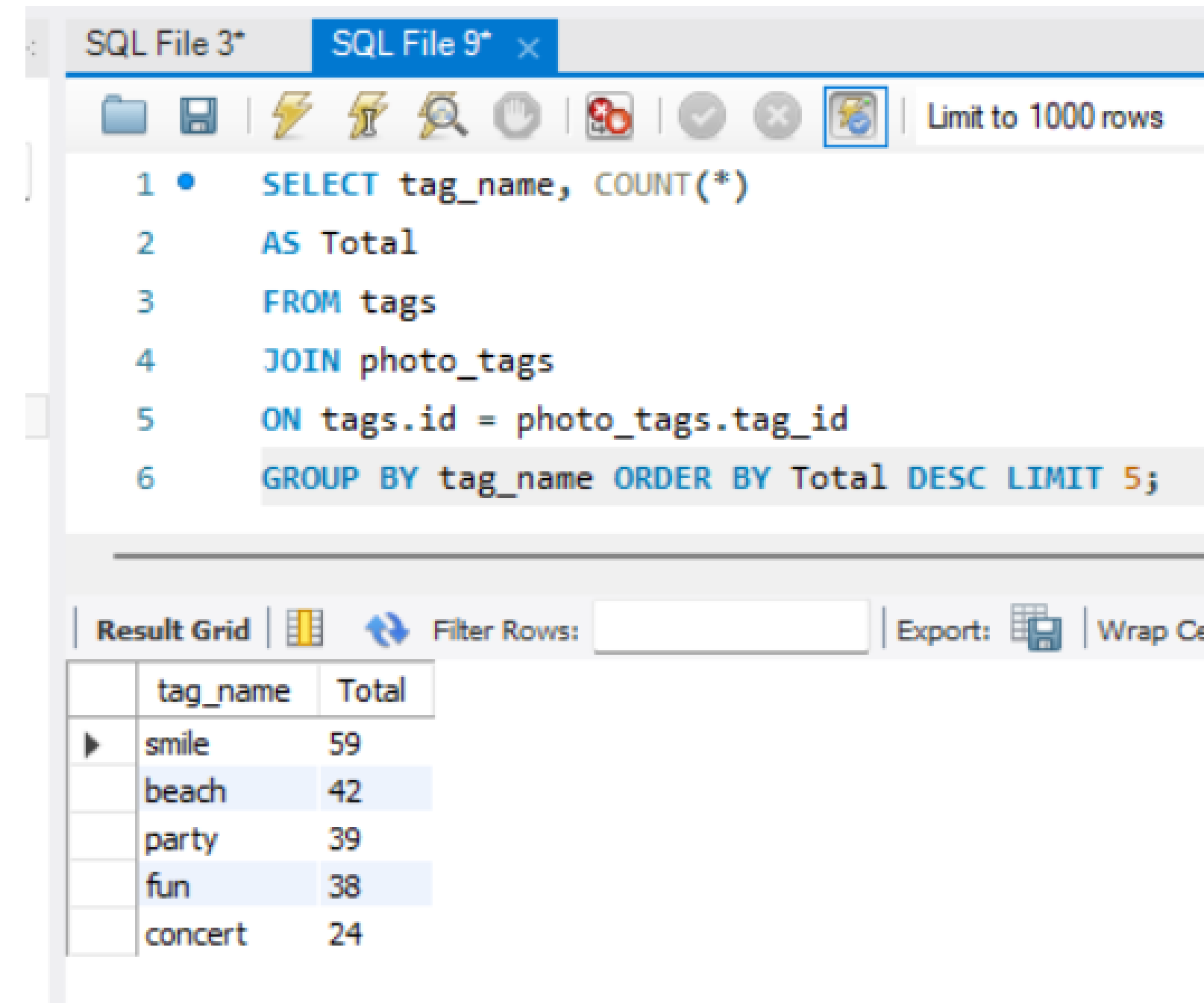
	photo_id	tag_id
▶	14	1
	21	1
	45	1
	75	1
	83	1
	85	1
	91	1
	118	1
	149	1
	194	1
	201	1
	210	1
	216	1
	227	1
	231	1
	233	1
	238	1

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

Identify and suggest the top 5 most commonly used hashtags on the platform

To identify the top 5 most commonly used hashtags on Instagram, we can analyze the posts in the database and extract the hashtags associated with each post. By counting the frequency of each hashtag, we can determine the most commonly used ones and suggest them to the partner brand.

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



The screenshot shows a SQL IDE interface with two tabs: 'SQL File 3*' and 'SQL File 9*'. The active tab 'SQL File 9*' contains the following SQL query:

```
1 • SELECT tag_name, COUNT(*)  
2 AS Total  
3 FROM tags  
4 JOIN photo_tags  
5 ON tags.id = photo_tags.tag_id  
6 GROUP BY tag_name ORDER BY Total DESC LIMIT 5;
```


Below the query editor, the 'Result Grid' tab is active, displaying the results of the query in a table:

	tag_name	Total
▶	smile	59
	beach	42
	party	39
	fun	38
	concert	24


The interface also includes a toolbar with various icons (file, save, run, search, etc.) and a 'Limit to 1000 rows' option.

- Lets Break down the query to see how we got our result:


SELECT tag_name, **COUNT(*)** **AS** Total This part of the query selects the column tag_name from the tags table and counts the number of occurrences of each tag. The COUNT(*) function counts the number of rows in the result set for each distinct tag_name. The AS Total renames the resulting count column as Total.



FROM tags
JOIN photo_tags **ON** tags.id = photo_tags.tag_id

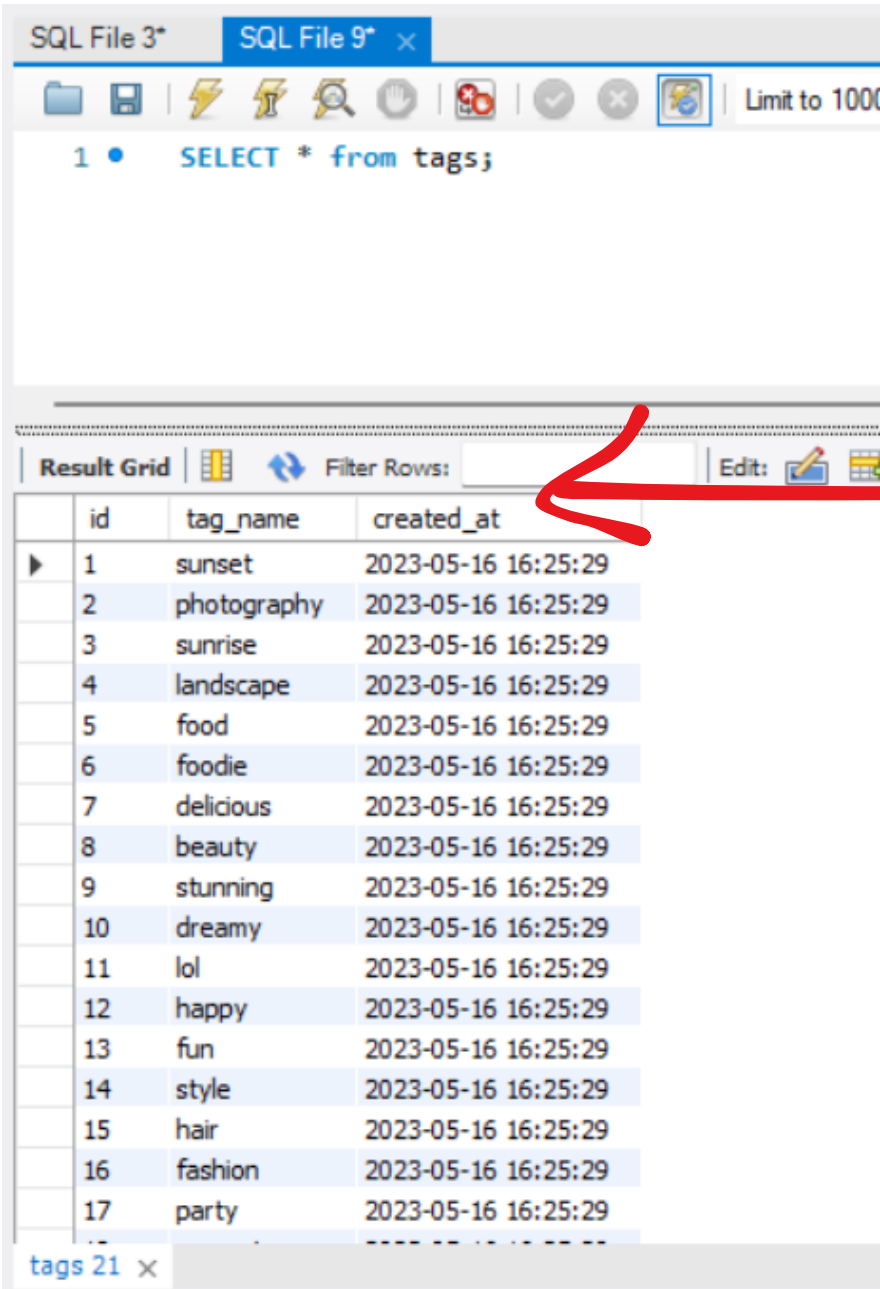
 This part specifies the tables involved in the query. It joins the tags table with the photo_tags table using the condition tags.id = photo_tags.tag_id. This join combines the rows from both tables where the tag_id in photo_tags matches the id in tags.

GROUP BY tag_name
ORDER BY Total **DESC**
LIMIT 5

 This clause groups the result set by the tag_name column. It ensures that the COUNT(*) function is applied to each distinct tag name separately and orders the result set in descending order based on the Total column. The highest counts will appear first.

- Before we proceed let us have a look at some more tables we will be using

users table



The screenshot shows a database client interface with two tabs: 'SQL File 3*' and 'SQL File 9*'. The active tab 'SQL File 9*' contains the SQL query: `SELECT * from tags;`. Below the query editor, the 'Result Grid' tab is active, displaying the results of the query. The results are shown in a table with three columns: 'id', 'tag_name', and 'created_at'. A red arrow points from the text 'we want to convert the dates into day_names and count it so find out the day with highest number of registrations' to the 'created_at' column header. The table contains 17 rows of data, all with a 'created_at' value of '2023-05-16 16:25:29'. The bottom of the window shows a tab labeled 'tags 21 x'.

id	tag_name	created_at
1	sunset	2023-05-16 16:25:29
2	photography	2023-05-16 16:25:29
3	sunrise	2023-05-16 16:25:29
4	landscape	2023-05-16 16:25:29
5	food	2023-05-16 16:25:29
6	foodie	2023-05-16 16:25:29
7	delicious	2023-05-16 16:25:29
8	beauty	2023-05-16 16:25:29
9	stunning	2023-05-16 16:25:29
10	dreamy	2023-05-16 16:25:29
11	lol	2023-05-16 16:25:29
12	happy	2023-05-16 16:25:29
13	fun	2023-05-16 16:25:29
14	style	2023-05-16 16:25:29
15	hair	2023-05-16 16:25:29
16	fashion	2023-05-16 16:25:29
17	party	2023-05-16 16:25:29

we want to convert the dates into day_names and count it so find out the day with highest number of registrations











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

What day of the week do most users register on? Provide insights on when to schedule an ad campaign

To determine the best day to launch ADs, we can analyze the registration data in the database and identify the day of the week with the highest number of user registrations. This information will provide insights into when users are most active and engaged, making it an optimal time to schedule the ad campaign.

```
SELECT DAYNAME(created_at)
AS Day,
COUNT(username) AS Total
FROM users GROUP BY Day
ORDER BY Total DESC LIMIT 2;
```

SQL File 3* SQL File 9* x SQL File 10*

  |     |  |    | Lin

1

SELECT DAYNAME(created_at)

2

AS Day, COUNT(username)


3


AS Total FROM users GROUP BY Day

4

ORDER BY Total DESC LIMIT 2;

Result Grid





Filter Rows:

Expo

	Day	Total
▶	Thursday	16
	Sunday	16

- Lets Break down the query to see how we got our result:

SELECT DAYNAME(created_at) **AS** Day



This part selects the day of the week (in text format) from the created_at column of the users table and aliases it as "Day". The DAYNAME() function is used to extract the day of the week from the given date.

COUNT(username) **AS** Total



This counts the number of occurrences of the username column in the users table and aliases it as "Total". It represents the total number of users registered on each day.

FROM users



This specifies the table from which the data is being retrieved, in this case, the users table.

GROUP BY Day



This clause groups the data by the "Day" column, so that we can calculate the count for each unique day of the week.

III Investor Metrics:

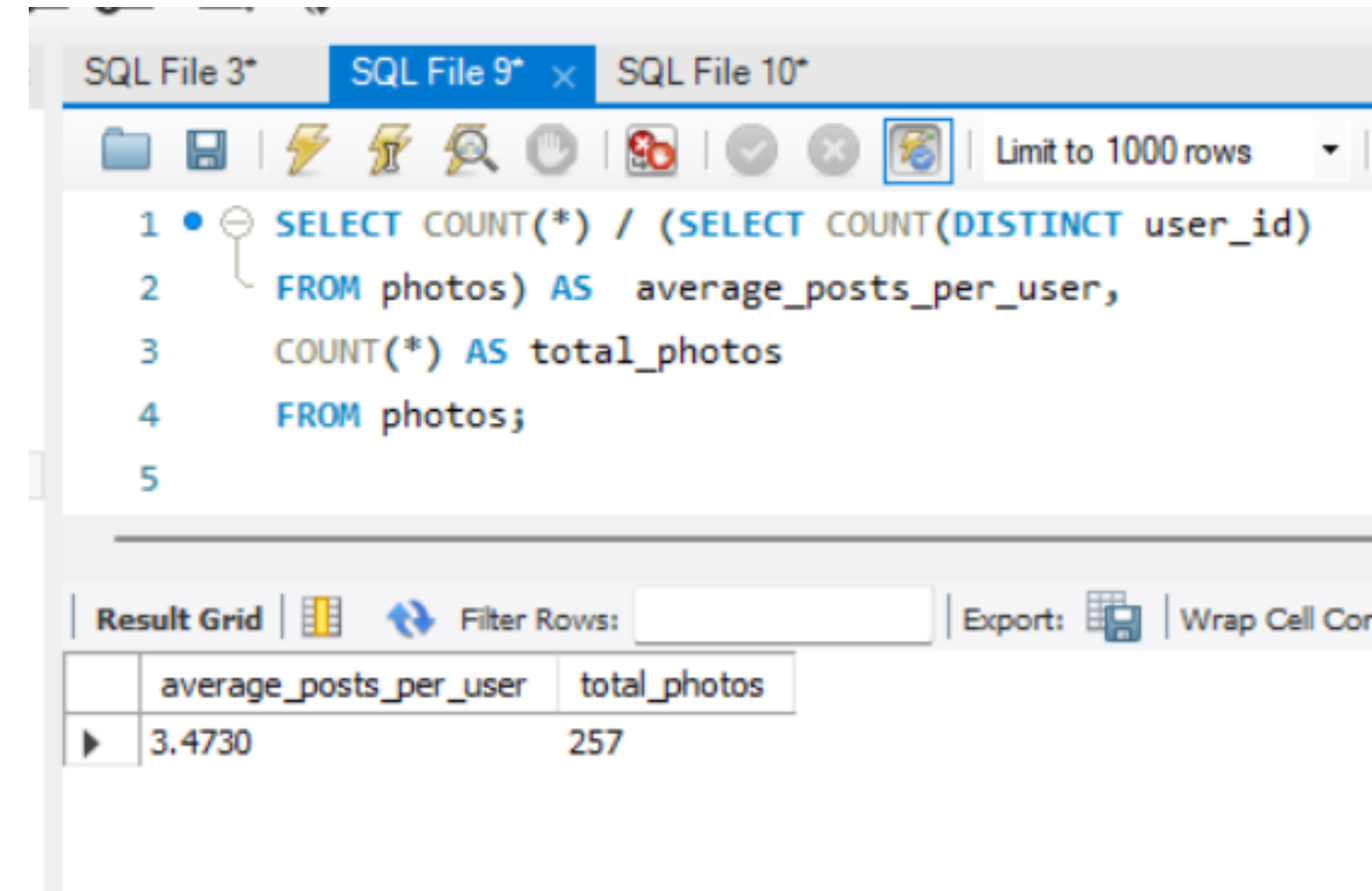
- **Bots & Fake Accounts:** The investors want to know if the platform is crowded with fake and dummy accounts

Provide data on users (bots) who have liked every single photo on the site (since any normal user would not be able to do this)

To calculate the average posts per user, we divide the total count of photos by the count of distinct users. By dividing these values, we get the average number of posts per user. This gives us an understanding of how active users are on the platform.

Also, by counting all the rows in the photos table, we obtain the total number of photos available on Instagram.

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



- Lets Break down the query to see how we got our result:

`SELECT COUNT(*) /`



This part calculates the count of all rows in the photos table. It counts every row in the table, regardless of the values in specific columns.

`(SELECT COUNT(DISTINCT user_id) FROM photos)`



This is a subquery that calculates the count of distinct user_id values in the photos table. The DISTINCT keyword ensures that only unique user_id values are counted.

`AS average_posts_per_user`



This is an alias given to the result of the division operation. It renames the column as "average_posts_per_user" for better readability and understanding.

`COUNT(*) AS total_photos`



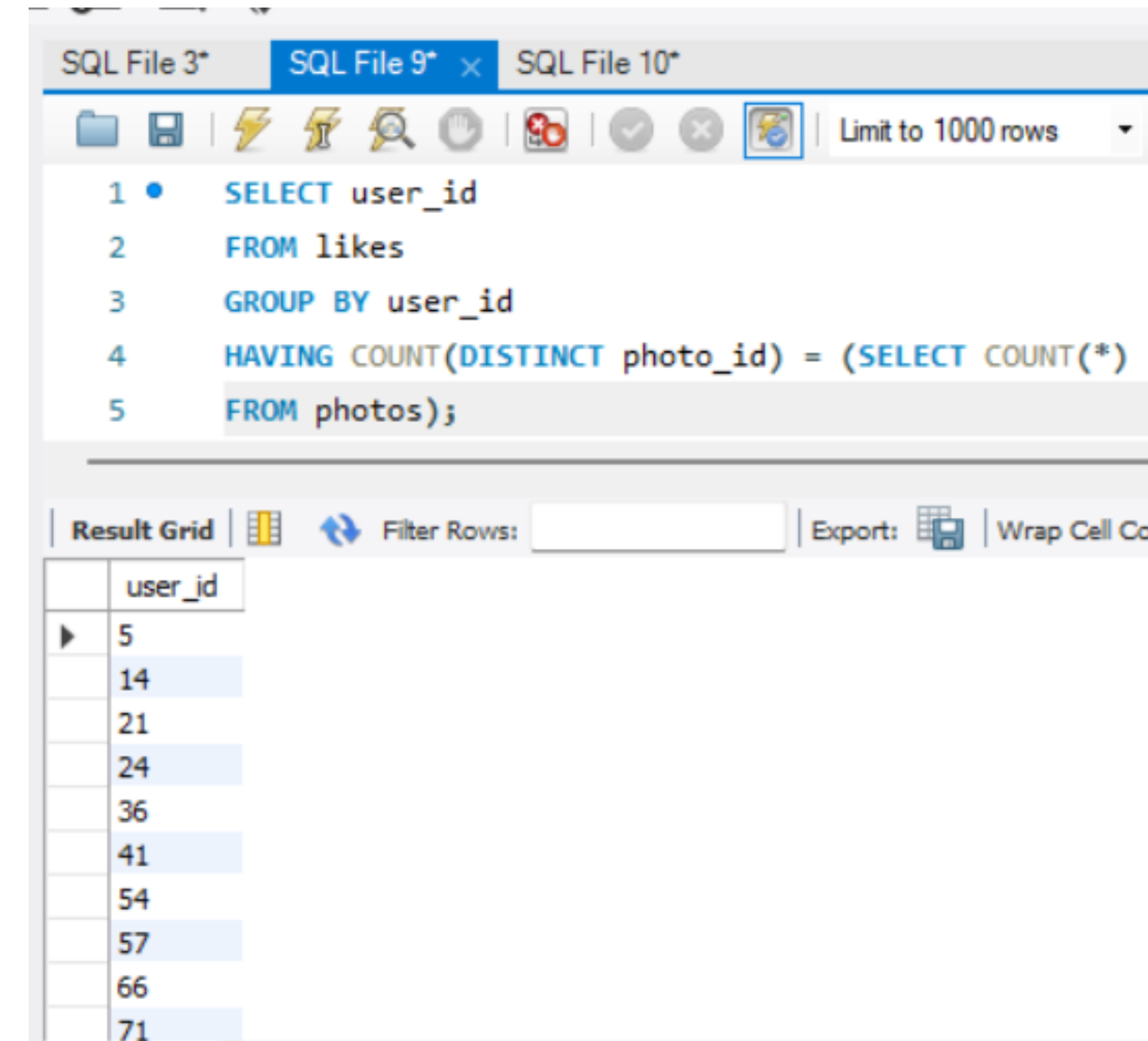
This calculates the count of all rows in the photos table, similar to step 1. It also assigns an alias "total_photos" to the result.

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

Provide data on users (bots) who have liked every single photo on the site (since any normal user would not be able to do this)

The approach here is to group the rows in the likes table by the user_id and check if the count of distinct photo_id values for each user matches the total count of photos in the photos table. Users who have liked every single photo would have a distinct photo_id count equal to the total count of photos in the database.

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



IV Insights

Based on the analysis of the provided data, here are some key insights, that might be important for the business developers of instagram:

- **Most Loyal Users:** Identifying the 5 oldest users of Instagram allows the business to reward and acknowledge their loyalty. These users could be valuable brand advocates and influencers who have been using the platform for a significant amount of time.
- **Inactive Users:** Identifying users who have never posted a single photo on Instagram provides an opportunity to re-engage them. By sending promotional emails or targeted campaigns encouraging them to start posting, the business can increase user activity and generate more content on the platform.
- **Contest Winner:** Determining the winner of the contest based on the most likes on a single photo allows the business to recognize and reward users who have created engaging and popular content. This highlights the importance of user-generated content and its impact on user engagement.
- **Top Hashtags:** Identifying the top 5 most commonly used hashtags on Instagram helps the business and partner brands reach a wider audience and increase the visibility of their content. Using popular hashtags can attract more users, increase engagement, and enhance brand exposure on the platform.

- Optimal Day for Ad Campaign: Analyzing the registration data and identifying the day of the week with the highest number of user registrations provides valuable insights for scheduling ad campaigns. Launching ads on the most active day can maximize reach and engagement, increasing the effectiveness of the campaign.
- User Engagement: Calculating the average number of posts per user allows the business to assess user engagement on Instagram. If the average number of posts per user is decreasing, it may indicate a decline in user activity and engagement, which requires attention to keep users actively using the platform.
- Bots & Fake Accounts: Identifying potential bot or fake accounts based on their liking behavior is crucial for maintaining a genuine and trustworthy user base. Detecting and addressing these accounts helps maintain the integrity of the platform, ensures fair engagement metrics, and fosters a positive user experience.



V RESULT

- Successfully used SQL queries to analyze user engagement and identify potential bots on Instagram.
- Developed practical skills in retrieving specific data, analyzing trends, and joining tables.
- Recognized the importance of rewarding the oldest users on Instagram and tapping into their influence.
- identified users who haven't posted any photos to encourage them to start sharing and increase activity on the platform.
- Determined the contest winner based on the most likes, showcasing the power of user-generated content.
- Researched the most commonly used hashtags to enhance brand visibility and connect with a wider audience.
- Detected potential bot accounts by identifying users who were liking every single photo.
- Gained hands-on experience in data analytics and SQL, boosting confidence for future projects.

Tuesday, 16th May 2023

Trinity

STUDENT

MARJIBA

marjibajamir9@gmail.com

**Thank you
@Trinity**