# Instagram User Analytics

- **Samruddhi Pawar**

## Project Description:

Instagram, as we all must know is one of the most used social sites in today's world.
It's really addicting right? You can share your favourite photo or video. It allows users to upload media that can be edited with various filters hashtags and geographical tagging. You also get options to share with your pre-approved followers or publicly. It is really interesting to think about where all the data gets stored when we like or comment on someone's picture. To understand all these, I created a small project that replicates some of the features of Instagram.

The aim of the project is to help the marketing team of Instagram to successfully launch an ad campaign by giving them useful insights. It also helps investors to make informed decisions about the future direction of the Instagram app.

This project aims to carry out an in-depth analysis of the user engagement process with the Instagram platform which will help the product team to launch better features for the platform. This project focuses mainly on two key aspects Marketing and Investor Metrics. Based on the user engagement and the data collected the insights need to be carried out and presented to the product team. The project will answer important questions like:

***A. Marketing***
Rewarding Most Loyal Users:
Remind Inactive Users to Start Posting
Declaring Contest Winner
Hashtag Research
Launch AD Campaign

***B. Investor Metrics:***
User Engagement
Bots & Fake Accounts

As we proceed above, I created a database by the name Instagram ig_clone. The database consists of a total of 7 tables.
1. USER TABLE
2. PHOTOS TABLE
3. COMMENTS TABLE
4. LIKES TABLE
5. FOLLOWS TABLE
6. TAGS TABLE
7. PHOTO-TAGS TABLE

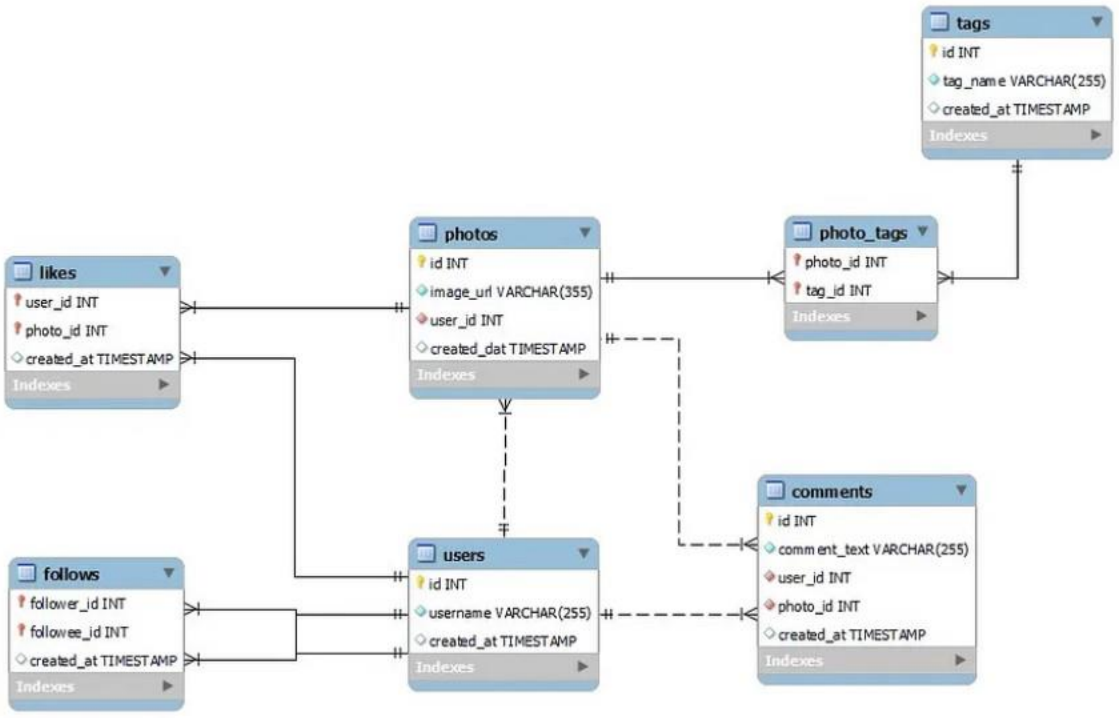
The user table consists of 3 attributes, the id being the primary. The photos table has its primary key by the name ID. Different tables were connected to it with the help of foreign keys such as likes, comments, photo_tags, and tags. There is a table called follows, which stores the followers and the following ID.

***Source of Data:***
https://docs.google.com/spreadsheets/d/1rFQWqj-lBGH_CQ9gdYr7hY9cHd3R7xPobdEO8QS5YFY/edit?usp=sharing

## Approach:

We used a methodical strategy to finish the project tasks, which included gaining access to and reviewing the Instagram information. We looked at things like the date of an Instagram user's first post, the quantity of photographs they uploaded, and the contest winner with the most likes. We also determined which day was optimal to display advertising and which hashtags were most popular. We examined the frequency of Instagram posts and the existence of phony accounts that like every single picture. We take care to maintain everything's security and privacy. We shared crucial information with the investors and made recommendations to the marketing team

Database creation: Created and inserted the values in the database using the DDL & DML SQL queries provided in the MySQL database using MySQL workbench.



## Tech-Stack Used:

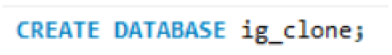
***MySQL:***
For this project, we used MySQL as the primary database management system. MySQL is a coding-based software that helps us organize and store data in a structured way. We chose

MySQL because it is widely used and has good support for working with databases. With MySQL, we can do all necessary tasks that are provided in our Task list and analyse the information we need from the Instagram database. It provided a reliable and efficient platform for managing and processing the data for our project tasks. MySQL is the only software that helped me to complete our project and allowed us to work with the Instagram data effectively.

***Canva:***
Canva is a web-based design tool that allows users to create visually appealing graphics, presentations, and social media content with ease.

## Database Generation:

**A. Create Database**

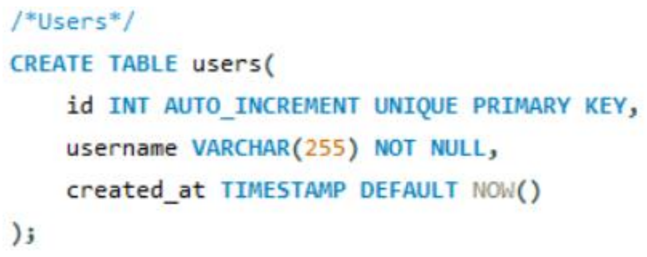


```
CREATE DATABASE ig_clone;
```

**B. Create Tables**

***1. Users table:***

This SQL code creates a table named **users** to store information about users. Here's a breakdown of each part of the code:



```
/*Users*/
CREATE TABLE users(
    id INT AUTO_INCREMENT UNIQUE PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT NOW()
);
```

*Table Name: users*
This is the name of the table you're creating to store user-related information.

*Columns:*

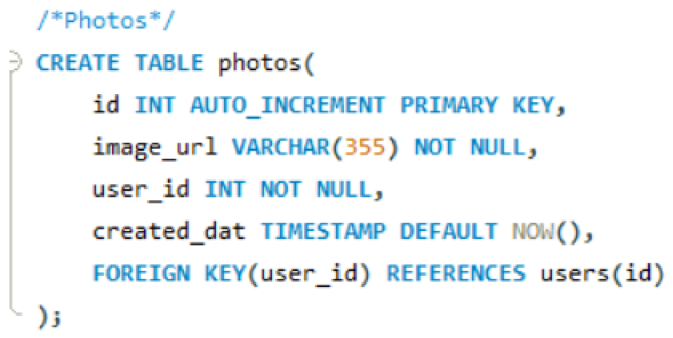
- *id INT AUTO_INCREMENT UNIQUE PRIMARY KEY:* This column is used as the primary key of the table. It's an integer (INT) data type and is set to auto-increment, meaning each new user entry will automatically get a unique ID value. The combination of *UNIQUE* and *PRIMARY KEY* ensures that each ID is unique and serves as the main identifier for each user.

- *username VARCHAR(255) NOT NULL:* This column stores the username of the user. It's a variable-length character (VARCHAR) data type with a maximum length of 255 characters. The *NOT NULL* constraint ensures that a username must be provided for every user.

- *created_at TIMESTAMP DEFAULT NOW():* This column records the timestamp when the user was created. It uses the TIMESTAMP data type and has a default value of the current timestamp (NOW()).

In summary, this SQL code creates a users table to store user information. Each user has a unique ID, a username, and a timestamp indicating when they were created. This table structure is common for managing user data in a database system.

2. ***Photos table:***
   This SQL command creates a table named **photos** to store information about photos uploaded by users. Here's a breakdown of the different parts of the code:



```
/*Photos*/
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)
);
```

*Table Name: photos*
This is the name of the table you're creating to store photo-related information.

*Columns:*

- *id INT AUTO_INCREMENT PRIMARY KEY:* This column is used as the primary key of the table. It's an integer (INT) data type and is set to auto-increment, ensuring each new photo entry gets a unique ID value. This unique ID acts as the main identifier for each photo.

- *image_url VARCHAR(355) NOT NULL:* This column stores the URL (web address) of the photo image. It's a variable-length character (VARCHAR) data type with a maximum length of 355 characters. The NOT NULL constraint ensures that an image URL must be provided for every photo.

- *user_id INT NOT NULL:* This column stores the ID of the user who uploaded the photo. It's an integer (INT) data type and cannot be left empty (NOT NULL).

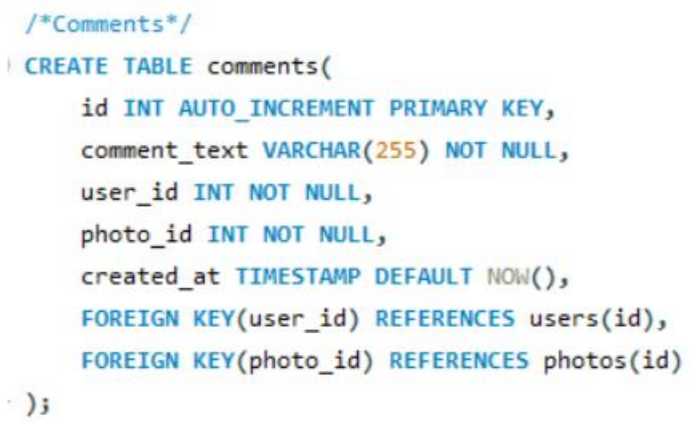
- *created_dat TIMESTAMP DEFAULT NOW():* This column records the timestamp when the photo was uploaded. It uses the TIMESTAMP data type and has a default value of the current timestamp (NOW()).

*Foreign Key:*

- *FOREIGN KEY(user_id) REFERENCES users(id):* This establishes a connection between the **user_id** column in the **photos** table and the **id** column in the **users** table. It creates a link between photos and the users who uploaded them.

In summary, the SQL code creates a **photos** table to store information about photos. Each photo has a unique ID, an image URL, the ID of the user who uploaded it, and a timestamp indicating when it was uploaded. The **user_id** column establishes a relationship with the **users** table through a foreign key constraint. This table structure is commonly used for managing photos and their associated data in a database system.

3. ***Comments table:***
This SQL code creates a table named **comments** to store information about comments made by users on photos. Here's a breakdown of the different parts of the code:



```sql
/*Comments*/
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)
);
```

*Table Name: comments*
This is the name of the table you're creating to store comment-related information.

*Columns:*

- *id INT AUTO_INCREMENT PRIMARY KEY:* This column is used as the primary key of the table. It's an integer (INT) data type and is set to auto-increment, ensuring each new comment entry gets a unique ID value. This unique ID serves as the main identifier for each comment.

- *comment_text VARCHAR(255) NOT NULL:* This column stores the text of the comment made by a user. It's a variable-length character (VARCHAR) data type with a maximum length of 255 characters. The **NOT NULL** constraint ensures that a comment text must be provided for every comment.

- *user_id INT NOT NULL:* This column stores the ID of the user who made the comment. It's an integer (INT) data type and cannot be left empty (NOT NULL).

- *photo_id INT NOT NULL:* This column stores the ID of the photo that the comment is related to. It's an integer (INT) data type and cannot be left empty (NOT NULL).

- *created_at TIMESTAMP DEFAULT NOW():* This column records the timestamp when the comment was made. It uses the TIMESTAMP data type and has a default value of the current timestamp (NOW()).

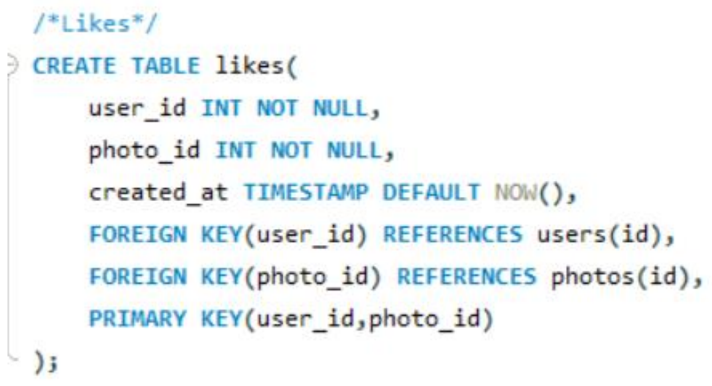
*Foreign Keys:*

- *FOREIGN KEY(user_id) REFERENCES users(id):* This establishes a connection between the **user_id** column in the **comments** table and the **id** column in the **users** table. It links comments to the users who made them.

- *FOREIGN KEY(photo_id) REFERENCES photos(id):* This creates a connection between the **photo_id** column in the **comments** table and the **id** column in the **photos** table. It associates comments with the photos they are posted on.

In summary, the SQL code creates a **comments** table to store information about comments made by users on photos. Each comment has a unique ID, the text of the comment, the ID of the user who made it, the ID of the related photo, and a timestamp indicating when it was created. The foreign keys establish relationships with the **users** and **photos** tables. This structure is commonly used for managing user comments in a database system.

4. ***Likes table***
This SQL code creates a table named **likes** to store information about likes that users give to photos. Here's an explanation of each part of the code:



```
/*Likes*/
CREATE TABLE likes(
    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),
    PRIMARY KEY(user_id,photo_id)
);
```

*Table Name: likes*
This is the name of the table you're creating to store information about likes.

*Columns:*

- *user_id INT NOT NULL:* This column stores the ID of the user who liked a photo. It's an integer (INT) data type and cannot be left empty (NOT NULL).

- *photo_id INT NOT NULL:* This column stores the ID of the photo that was liked. It's an integer (INT) data type and cannot be left empty (NOT NULL).

- *created_at TIMESTAMP DEFAULT NOW():* This column records the timestamp when the like was given. It uses the TIMESTAMP data type and has a default value of the current timestamp (NOW()).

*Foreign Keys:*

- *FOREIGN KEY(user_id) REFERENCES users(id):* This establishes a link between the **user_id** column in the **likes** table and the **id** column in the **users** table. It associates likes with the users who gave them.

- *FOREIGN KEY(photo_id) REFERENCES photos(id):* This creates a relationship between the **photo_id** column in the **likes** table and the **id** column in the **photos** table. It connects likes with the photos that were liked.

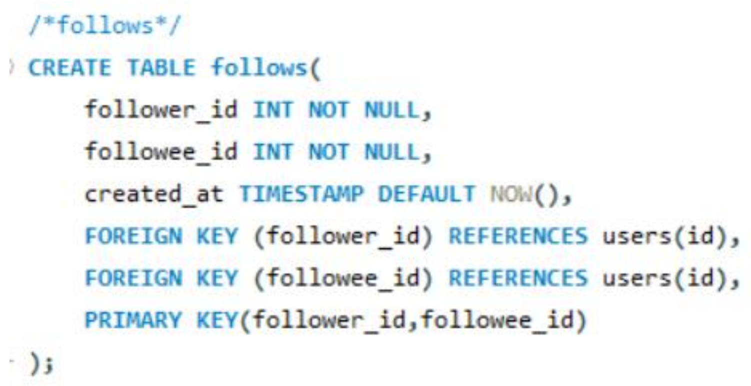
*Primary Key:*

- *PRIMARY KEY(user_id, photo_id):* This designates a composite primary key consisting of both the **user_id** and **photo_id** columns. This combination ensures that each user can only like a particular photo once, avoiding duplicate likes.

In summary, the SQL code creates a **likes** table to store information about likes. Each like is associated with a user and a photo, along with the timestamp of when the like was given. The foreign keys link the **likes** table to the **users** and **photos** tables, and the composite primary key ensures the uniqueness of like relationships. This structure is commonly used for tracking likes or similar interactions in a database system.

5. ***Follows table***

This SQL code snippet defines a table named **follows** that represents a relationship between users who follow each other on a platform. Here's a breakdown of what each part of the code does:



```
/*follows*/
CREATE TABLE follows(
    follower_id INT NOT NULL,
    followee_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT NOW(),
    FOREIGN KEY (follower_id) REFERENCES users(id),
    FOREIGN KEY (followee_id) REFERENCES users(id),
    PRIMARY KEY(follower_id,followee_id)
);
```

*Table Name: follows*
This is the name of the table that you're creating to store information about user relationships.

*Columns:*

- *follower_id INT NOT NULL:* This column stores the ID of the user who is following someone. It's an integer (INT) data type and cannot be left empty (NOT NULL).

- *followee_id INT NOT NULL:* This column stores the ID of the user who is being followed. Like the previous column, it's an integer (INT) data type and cannot be left empty (NOT NULL).

- *created_at TIMESTAMP DEFAULT NOW():* This column records the timestamp when the follow relationship was created. It uses the TIMESTAMP data type and has a default value of the current timestamp (NOW()).

*Foreign Keys:*

- *FOREIGN KEY (follower_id) REFERENCES users(id):* This indicates that the **follower_id** column is a foreign key that refers to the **id** column in the **users** table. It establishes a connection between the **follows** table and the **users** table based on the follower's ID.

- *FOREIGN KEY (followee_id) REFERENCES users(id):* Similar to the previous foreign key, this establishes a connection between the **follows** table and the **users** table based on the followee's ID.

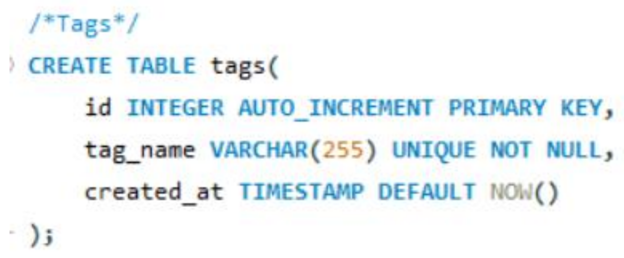
*Primary Key:*

- *PRIMARY KEY(follower_id, followee_id):* This declares a composite primary key that consists of both the **follower_id** and **followee_id** columns. This combination ensures that each follower can only follow a followee once, preventing duplicate follow relationships.

In summary, this SQL code creates a table that stores information about users following each other. The table captures the IDs of the follower and the followee, along with the timestamp of the follow action. Foreign keys link the user IDs to the **users** table, and a composite primary key ensures the uniqueness of the follow relationships. This is a common way to model social network-like interactions in a relational database.

6. ***Tags table***

This SQL code creates a table named **tags** to store information about tags associated with photos or other entities. Here's an explanation of each part of the code:



```
/*Tags*/
CREATE TABLE tags(
    id INTEGER AUTO_INCREMENT PRIMARY KEY,
    tag_name VARCHAR(255) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT NOW()
);
```

*Table Name: tags*
This is the name of the table you're creating to store tag-related information.

*Columns:*

- *id INTEGER AUTO_INCREMENT PRIMARY KEY:* This column is used as the primary key of the table. It's an integer (INTEGER) data type and is set to auto-increment, ensuring each new tag entry gets a unique ID value. This unique ID serves as the main identifier for each tag.

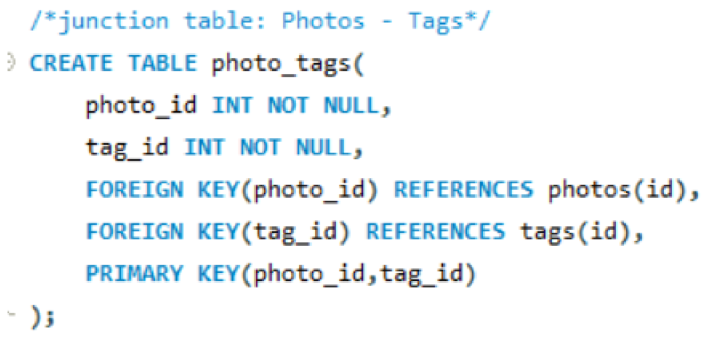
- *tag_name VARCHAR(255) UNIQUE NOT NULL:* This column stores the name of the tag. It's a variable-length character (VARCHAR) data type with a maximum length of 255 characters. The **NOT NULL** constraint ensures that a tag name must be provided for every tag. The **UNIQUE** constraint ensures that each tag name is unique, preventing duplicates.

- *created_at TIMESTAMP DEFAULT NOW():* This column records the timestamp when the tag was created. It uses the TIMESTAMP data type and has a default value of the current timestamp (NOW()).

In summary, the SQL code creates a **tags** table to store information about tags. Each tag has a unique ID, a tag name, and a timestamp indicating when it was created. The **tag_name** column has a **UNIQUE** constraint to ensure uniqueness, preventing multiple tags with the same name. This table structure is commonly used to manage tags associated with various entities, such as photos, posts, or articles.

7. ***photo_tags table***

This SQL code creates a junction table named **photo_tags** to establish a many-to-many relationship between photos and tags. This junction table is used to associate tags with photos. Here's an explanation of each part of the code:



```
/*junction table: Photos - Tags*/
CREATE TABLE photo_tags(
    photo_id INT NOT NULL,
    tag_id INT NOT NULL,
    FOREIGN KEY(photo_id) REFERENCES photos(id),
    FOREIGN KEY(tag_id) REFERENCES tags(id),
    PRIMARY KEY(photo_id,tag_id)
);
```

*Table Name: photo_tags*
This is the name of the junction table you're creating to manage the relationship between photos and tags.

*Columns:*

- *photo_id INT NOT NULL:* This column stores the ID of the photo that is associated with a tag. It's an integer (INT) data type and cannot be left empty (NOT NULL).

- *tag_id INT NOT NULL:* This column stores the ID of the tag that is associated with a photo. It's an integer (INT) data type and cannot be left empty (NOT NULL).

*Foreign Keys:*

- *FOREIGN KEY(photo_id) REFERENCES photos(id):* This establishes a relationship between the **photo_id** column in the **photo_tags** table and the **id** column in the **photos** table. It links the junction table to the photos that have been tagged.

- *FOREIGN KEY(tag_id) REFERENCES tags(id):* This creates a relationship between the **tag_id** column in the **photo_tags** table and the **id** column in the **tags** table. It links the junction table to the tags that have been assigned to photos.

*Primary Key:*

- *PRIMARY KEY(photo_id, tag_id):* This designates a composite primary key consisting of both the **photo_id** and **tag_id** columns. This combination ensures that each photo can have a tag assigned only once and prevents duplicate associations.

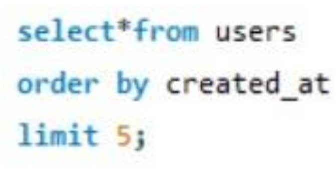
In summary, the SQL code creates a **photo_tags** junction table to manage the many-to-many relationship between photos and tags. Each row in this table indicates that a specific photo is associated with a specific tag. The foreign keys connect the junction table to the **photos** and **tags** tables, and the composite primary key ensures the uniqueness of the associations. This type of junction table is commonly used to handle many-to-many relationships in a relational database.

C. **Inserting Values into tables**
Now we will insert values into all tables.

## Insights:

***A. MARKETING*** - The team wants to launch some campaigns, and they need some help with the following questions-

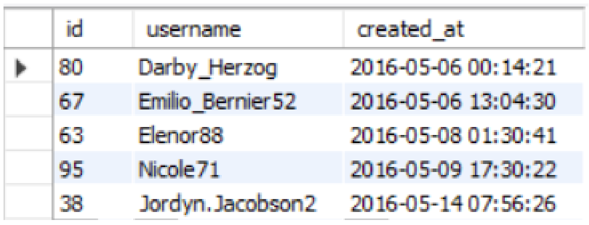
***1. Rewarding the most loyal users:*** It shows the people who have been using the platform for the longest time.
*Task* – Find the 5 oldest users of Instagram from the database provided by the team.



```
select*from users
order by created_at
limit 5;
```

SELECT – it tells your database that you want to select data.
FROM users tells the database to select data from the user table.
(*) tells the database that you want to see all columns in this table.
ORDER BY- after this expression, simply specify a column on which the data will be sorted.
LIMIT n - returns the first n rows from the result. This is much more efficient than returning all the data from the database.

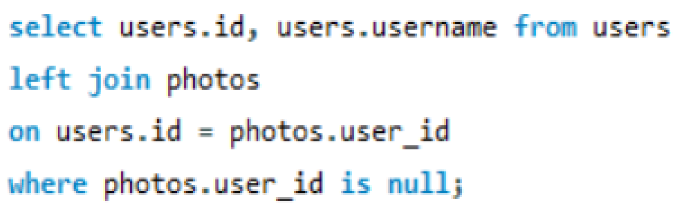
*Output:*



| | 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 |

*Results:* These are the 5 candidates who have been using the platform for the longest time.

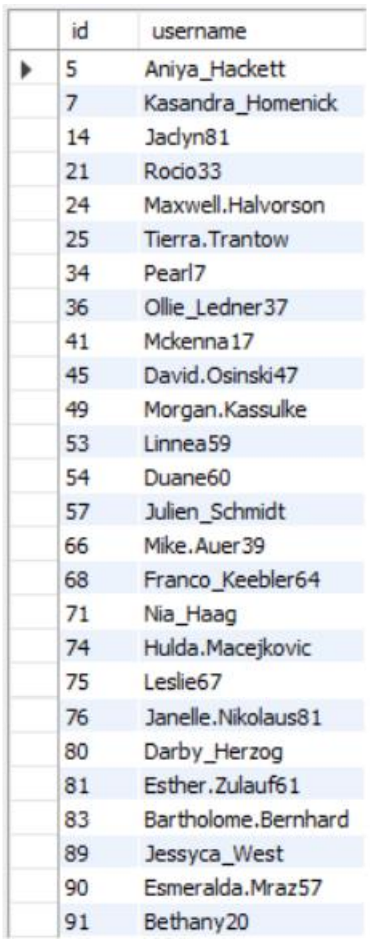
***2. Inactive users on Instagram***
*Task* - Find the users who have never posted a single photo on Instagram.



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

LEFT JOIN - works in the following way: it returns all rows from the left table (the first table in the query) plus all matching rows from the right table (the second table in the query).

*Output:*



| 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 |
| 75 | Leslie67 |
| 76 | Janelle.Nikolaus81 |
| 80 | Darby_Herzog |
| 81 | Esther.Zulauf61 |
| 83 | Bartholome.Bernhard |
| 89 | Jessyca_West |
| 90 | Esmeralda.Mraz57 |
| 91 | Bethany20 |

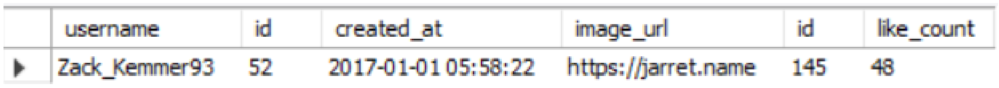
*Result:* There are a total of 100 users out of which only 74 users post the pictures on Instagram and 26 are those users who never posted a single photo on Instagram. We have to send them promotional emails to post their 1st photo. Details of those 26 users are in the given picture.

3. ***Declaring Contest Winner:***
   *Task*: Identify the winner of the contest and provide their details to the team. To do this task we need to find the most popular photo with the most likes and the user who created it.

```sql
select users.username, users.id, users.created_at, photos.image_url, photos.id,
Count(*) as like_count from users
Join photos on users.id = photos.user_id
Join likes on photos.id = likes.photo_id
group by users.id, photos.id
order by like_count desc
limit 1;
```

INNER JOIN (or JOIN, for short) only shows those rows from the two tables where there is a match between the columns. In other words, you can only see those pieces of equipment that have a room assigned and vice versa

*Output*:



| | username | id | created_at | image_url | id | like_count |
|---|---|---|---|---|---|---|
| ▶ | Zack_Kemmer93 | 52 | 2017-01-01 05:58:22 | https://jarret.name | 145 | 48 |

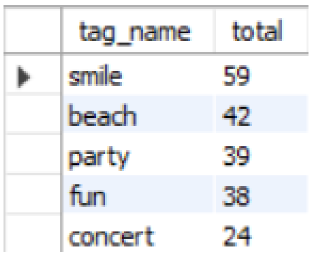
*Result*:
The contest winner is Zack_Kemmer93 whose ID is 52 gets 48 likes on a single photo.

4. ***Hashtag Researching:***
Hashtag helps the user to reach to wide range of people. It is used to draw attention, organize, promote, and connect.
*Task*- to identify the top 5 most commonly used hashtags on Instagram.

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

COUNT()- function returns the number of rows that match the specific criteria.
AS - The new name is just an alias, which means it's temporary and doesn't change the actual column name in the database. It only influences the way the column is shown in the result of the specific query. This technique is often used when there are a few columns with the same name coming from different tables.

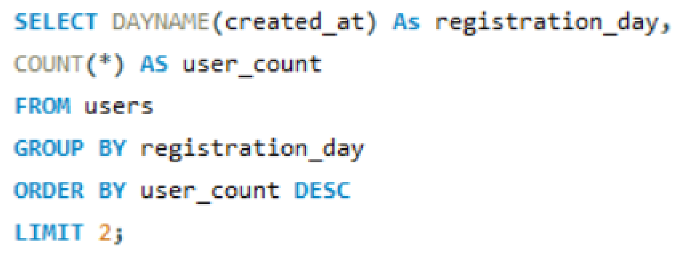
*Output*:



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

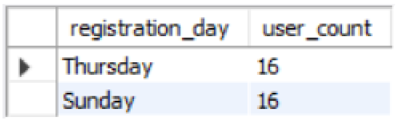
*Result*:
These are the 5 most commonly used hashtags on the platform i.e. smile, beach, party, fun, and concert. Instagram can suggest their partner brand add these to their new posts to reach out to most people who are using the platform.

5. ***Launch Ad Campaign:***
*Task* – To find out the day of the week when most users register on Instagram.



```
SELECT DAYNAME(created_at) As registration_day,
COUNT(*) AS user_count
FROM users
GROUP BY registration_day
ORDER BY user_count DESC
LIMIT 2;
```

*Output:*

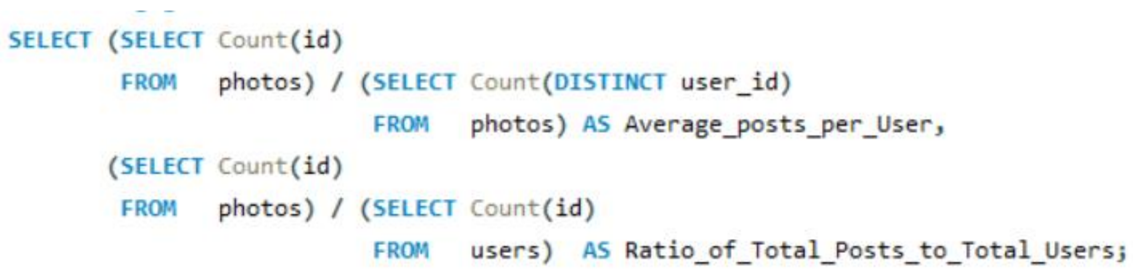


| registration_day | user_count |
|---|---|
| Thursday | 16 |
| Sunday | 16 |

*Result:*
The query result shows two days of the week when the users register which is Thursday and Sunday. According to me the most suitable day to launch the ad campaign would be Sunday because users mostly have leisure time on Sunday and more interaction happens on Sunday.

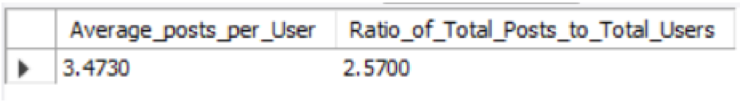
***B. INVESTOR METRICS***

6. ***User Engagement*** - Investors want to know that Instagram is not becoming redundant like Facebook, so they want to check the frequency of how much the users are engaging on the platform.
*Task* – to provide how many times an average user posts on Instagram.



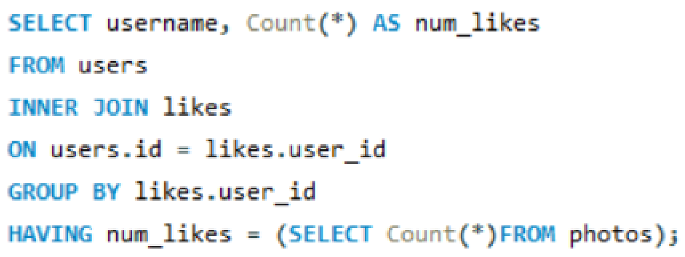
```sql
SELECT (SELECT Count(id)
        FROM   photos) / (SELECT Count(DISTINCT user_id)
                          FROM   photos) AS Average_posts_per_User,
       (SELECT Count(id)
        FROM   photos) / (SELECT Count(id)
                          FROM   users)  AS Ratio_of_Total_Posts_to_Total_Users;
```

*Output*:



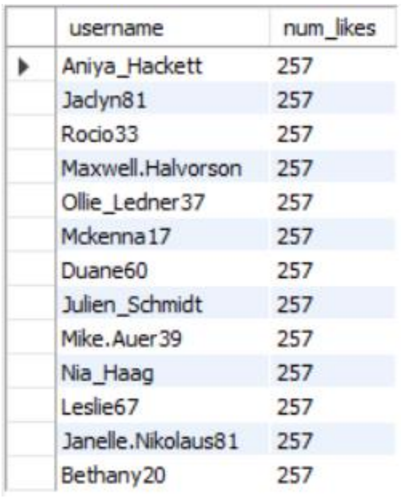
| Average_posts_per_User | Ratio_of_Total_Posts_to_Total_Users |
|---|---|
| 3.4730 | 2.5700 |

7. ***Bots and Fake Accounts*** - It is reported that there are a lot of bots and fake accounts on the platform. The investors want to know if there are fake and dummy accounts. Task – To Provide data on users(bots) who have liked every single photo on the site (normal users would not be able to do this).



```sql
SELECT username, Count(*) AS num_likes
FROM users
INNER JOIN likes
ON users.id = likes.user_id
GROUP BY likes.user_id
HAVING num_likes = (SELECT Count(*)FROM photos);
```

*Output*:



| username | num_likes |
|---|---|
| Aniya_Hackett | 257 |
| Jaclyn81 | 257 |
| Rocio33 | 257 |
| Maxwell.Halvorson | 257 |
| Ollie_Ledner37 | 257 |
| Mckenna17 | 257 |
| Duane60 | 257 |
| Julien_Schmidt | 257 |
| Mike.Auer39 | 257 |
| Nia_Haag | 257 |
| Leslie67 | 257 |
| Janelle.Nikolaus81 | 257 |
| Bethany20 | 257 |

## Summary

- The top 5 oldest customers who are considered to be the most loyal ones created their accounts in the month of May. Darby was the first one to register
- There are 5 users who have never posted a single picture on Instagram as of yet.
- The contest winner is Zack_Kemmer93 whose id is 52 gets 48 likes on a single photo.
- These are the 5 most commonly used hashtags on the platform i.e. smile, beach, party, fun, concert.
- There are 13 bots on the platform that could fill the app with spam and irrelevant content.
- It was found that Thursdays and Sundays were the days when most users registered.
- On average, a user was posting 3-4 photos in a year.

## Conclusion

From this project, I got an idea about how as a business or data analyst we work on real-time data to make any data-driven decision.
One thing I infer about this project is the dataset provided was very limited and small in respect of Rows and columns, but still, it was a very good experience working on such kind of project.
It helped me a lot to understand the analysis process well, and to provide insights for the best decision possible.

The Instagram user analytics project provided insights on marketing, User engagements, Bots, and Fake accounts. Now these insights can be used by the Instagram product team to launch new campaigns, track user engagement, and improve user experience.

- Here are the insights I found in these project

1. Top 5 Oldest Users of Instagram
2. Users who never posted photos on Instagram
3. Most liked photo on Instagram
4. Top 5 most commonly used has-tags on Instagram
5. Total number of users on Instagram
6. Total number of photos on Instagram
7. Average number of photos per user
8. Bots and fake accounts on Instagram