

Instagram User Analytics

SQL Fundamentals

Project Description:

The project report describes analysis of user interactions and engagement with the Instagram app. We're able to find various highlights about user interactions which are used to develop different strategies for app improvement. This will help provide valuable insights to developers and investors that can help the business grow.

Approach:

The tasks have been divided as information for the developer team and investors.

We start by creating the database in MySQL using MySQL workbench.

Various SQL commands have been used to retrieve information from the database according to our requirement.

We start by creating our database in MySQL.

Next we run a command to use the said database that has been created.

The data is stored in a tabular format. These tables are created using the CREATE command.

INSERT command is used to insert data in the tables.

We use various DDL and DML queries like SELECT, FROM, GROUP BY, ORDER BY, etc to retrieve and show information from the database.

Tech-Stack Used:

Intel Core i5

Windows 10

MySql 8.0 CE Workbench

Local host MySQL80

Insights:

Creating a database in MySQL:

CREATE DATABASE ig_clone;

Command has been used to create a database named as ig_clone.

USE ig_clone;

USE database command is run to select the database that we work on.

Creating a table:

Creating a table named users to save information related to users of instagram app.

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

CREATE TABLE command is used to create the table.

We create three columns in our table, namely

1) Id

```
id INT AUTO_INCREMENT UNIQUE PRIMARY KEY,
```

The query receives a unique integer value as input. We define the ID column as the primary key to reference other foreign key tables.

Primary key values are unique and distinct.

2) username

```
username VARCHAR(255) NOT NULL,
```

The query receives a string as input. **NOT NULL** specifies the input value can't be null.

3) created_at

```
created_at TIMESTAMP DEFAULT NOW()
```

This column specifies the date and time on which the user id was created.

Similarly, we create our other tables for

Photos uploaded on the app

```
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)  
);
```

Foreign Key is used to reference the primary key which in our case is the **id** column.

Database for all comments on the app

```
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)  
);
```

For 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)
```

For 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)  
);
```

For hashtags

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

Then we create photo_tags table as junction table

```
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)  
);
```

Inserting values in table columns:

Inserting values in table/columns we have created, we use

INSERT INTO tablename (column1, column2,..column n) VALUES

(expression1,expression2,.....expression n);

where tablename is the table we want to add values to, column refers to the columns in the table and expressions are the values that we want to add.

Result:

In order to visualize the data in tables we use query

SELECT * FROM tablename;

To visualize data from users table we use command

select * from users;

we get output as

The screenshot shows a database management interface. On the left, a sidebar lists database objects: m, _clone, Tables, Views, Stored Procedures, Functions, kila, s, and prfd. The main area displays a SQL query in a text editor:

```
148 SELECT
149     username, COUNT(*) AS all_likes
150 FROM
151     users u
152 JOIN
153     likes l ON u.id = l.user_id
```

Below the query editor, the 'Result Grid' is visible, showing a table with three columns: id, username, and created_at. The table contains 17 rows of data:

id	username	created_at
1	Kenton_Kirln	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	Ardy_Bogen63	2016-06-13 01:28:43
5	Aniya_Hackett	2016-12-07 01:04:39
6	Travon_Waters	2017-04-30 13:26:14
7	Kassandra_Homenick	2016-12-12 06:50:08
8	Tabitha_Schamberger11	2016-08-20 02:19:46
9	Gu93	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	Derek65	2017-01-19 01:34:14
13	Alexandro35	2017-03-29 17:09:02
14	Jacyln81	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

At the bottom, there is an 'Output' section with a dropdown menu set to 'Action Output'. The Windows taskbar at the very bottom shows the time as 16:56 on 24-10-2024.

Marketing Analysis

1.Loyal User Reward:

The marketing team wants to reward the most loyal users, i.e., those who have been using the platform for the longest time.

Your Task: Identify the five oldest users on Instagram from the provided database.

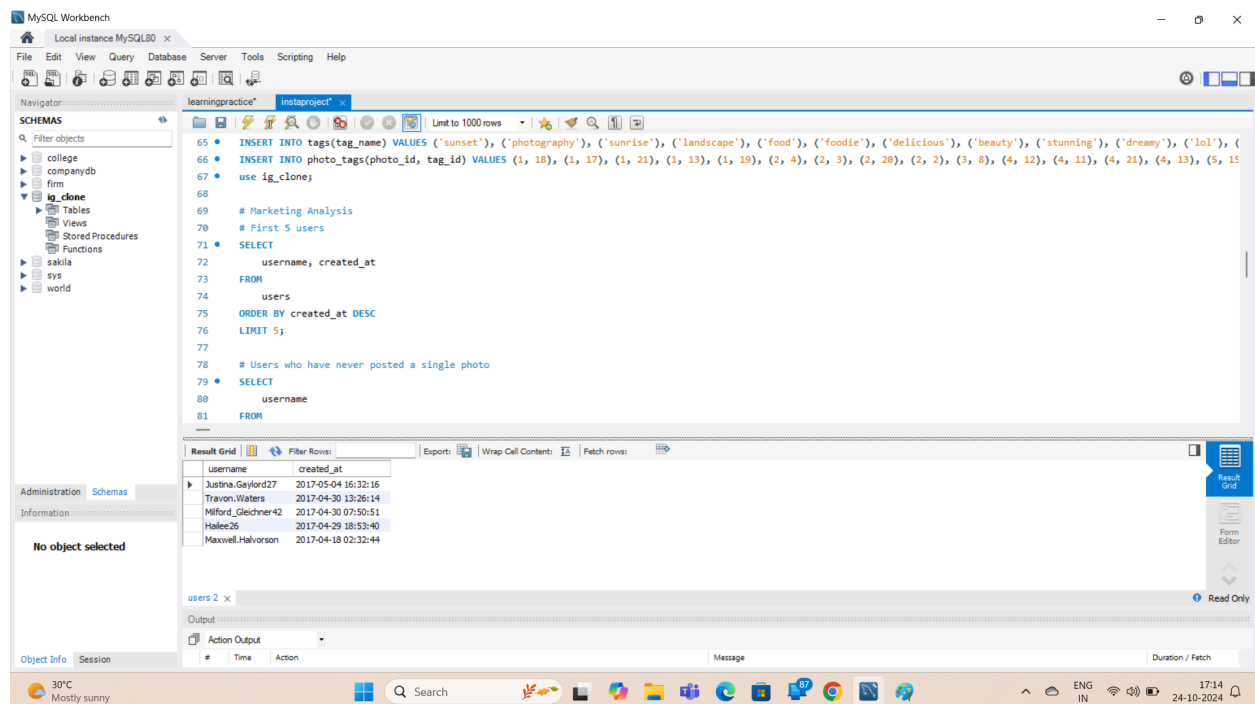
Query:

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

Explanation:

We select columns username and created_at from users table and order the entries from created_at in descending order. We limit the number of rows to 5.

Result:



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

```
65 • INSERT INTO tags(tag_name) VALUES ('sunset'), ('photography'), ('sunrise'), ('landscape'), ('food'), ('foodie'), ('delicious'), ('beauty'), ('stunning'), ('dreamy'), ('lol'), (
66 • 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), (5, 15
67 • use ig_clone);
68
69 # Marketing Analysis
70 # First 5 users
71 • SELECT
72     username, created_at
73 FROM
74     users
75 ORDER BY created_at DESC
76 LIMIT 5;
77
78 # Users who have never posted a single photo
79 • SELECT
80     username
81 FROM
```

The Results tab shows the output of the query:

username	created_at
Justina.Gaylord27	2017-05-04 16:32:16
Travon.Waters	2017-04-30 13:36:14
Milford_Gleichner42	2017-04-30 07:50:51
Hallee26	2017-04-29 18:53:40
Maxwell.Halvorson	2017-04-18 02:32:44

Conclusion:

Justina.Gaylord27, Travon.Waters, Milford_Gliechner42, Hailee26, Maxwell.Halvorson are the first five users of instagram app.

2. Inactive User Engagement:

The team wants to encourage inactive users to start posting by sending them promotional emails.

Your Task: Identify users who have never posted a single photo on Instagram.

Query:

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

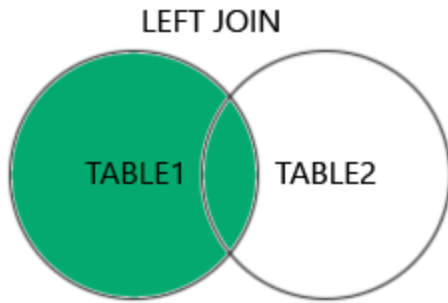
Explanation:

Here we use **JOINT** clauses to join photos on users. To find records where users id having null join records in photos are given as output.

The **WHERE** clause is used to filter records. It is used to extract only those records that fulfill a specified condition.

LEFT JOIN

The LEFT JOIN command returns all rows from the left table, and the matching rows from the right table.



Result:

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

```

77
78 # Users who have never posted a single photo
79 • SELECT
80     username
81 FROM
82     users
83     LEFT JOIN
84     photos ON users.id = photos.user_id
85 WHERE
86     photos.id IS NULL;
87
88 # Details of contest winner for most liked photo
89 • SELECT

```

The Result Grid displays the following usernames:

username
Aniya_Hackett
Kassandra_Homenick
Jacyln81
Rocio33
Maxwell_Halvorson
Tierra_Tranter
Peat7
Olle_Ledner37
Mckenna17
David_Osinski47
Morgan_Kassulke
Linnea59
Travis401

Conclusion:

The result table displays the usernames of IDs who have never uploaded a single photo on instagram.

3. Contest Winner Declaration:

The team has organized a contest where the user with the most likes on a single photo wins. Your Task: Determine the winner of the contest and provide their details to the team.

Query:

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

Explanation:

SELECT command is used to show selected columns.

COUNT(*) is used to count the number of entries in the photos table.

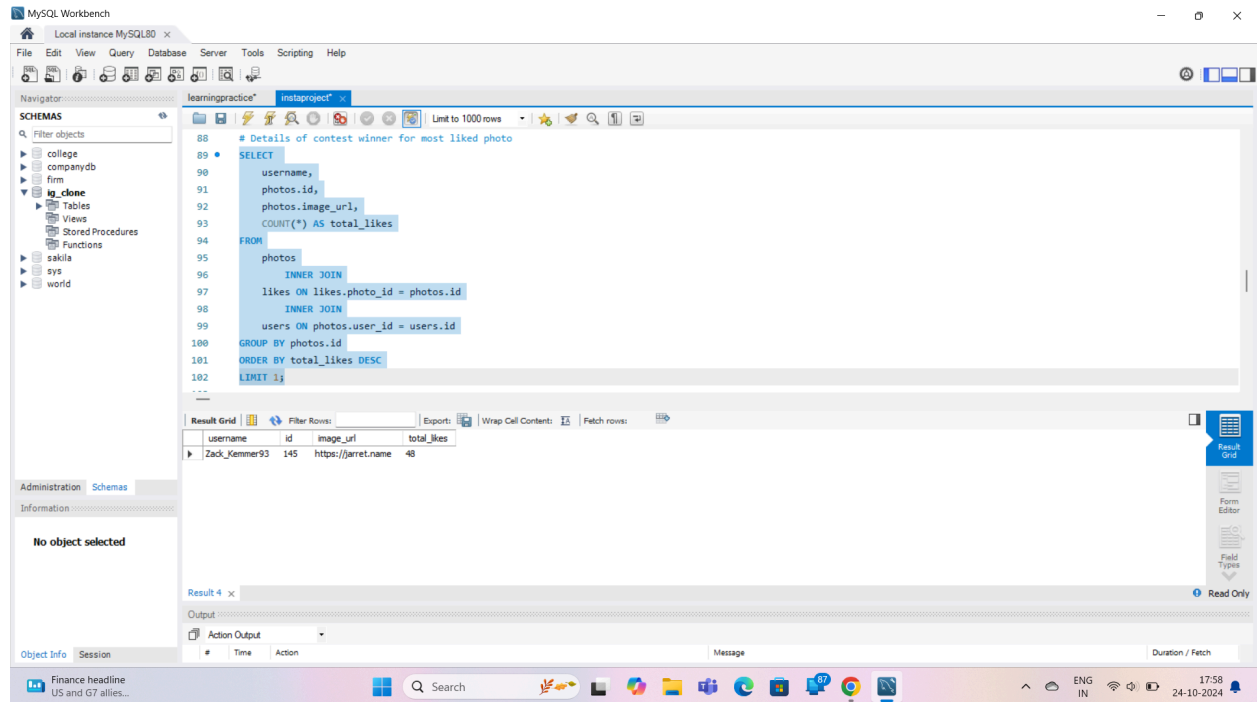
INNER JOIN

The INNER JOIN clause in SQL is used to combine multiple tables and fetch records that have the same values in the common columns.

First inner join is applied as a likes on likes (photos_id) column. Another is applied as users on photos(user_id) column.

The result is given in descending order using GROUP BY and DESC. Entries are limited to 1.

Result:



Conclusion:

User Zack_Kemmer93 is the winner of the contest with 48 likes which is most number of likes on a single post.

4. Hashtag Research:

A partner brand wants to know the most popular hashtags to use in their posts to reach the most people.

Your Task: Identify and suggest the top five most commonly used hashtags on the platform.

Query:

SELECT

tag_name, COUNT(tag_name) AS most_popular

FROM

tags

JOIN

photo_tags ON tags.id = photo_tags.tag_id

GROUP BY tags.id

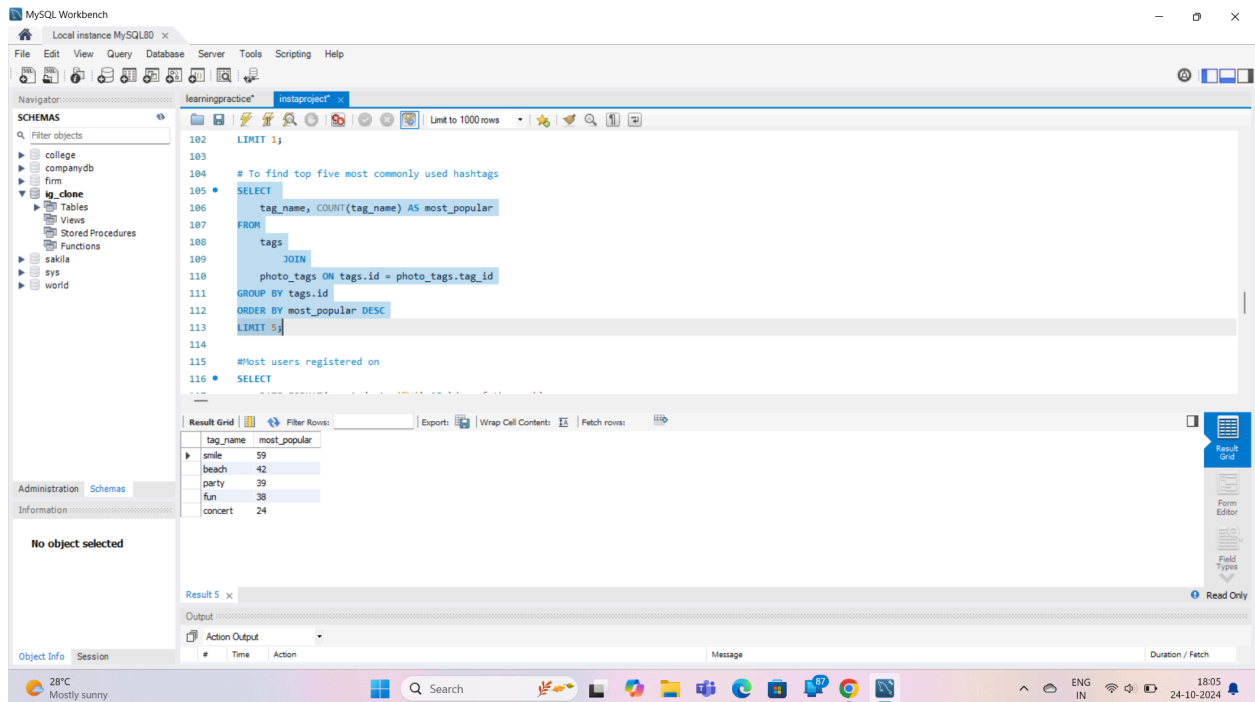
ORDER BY most_popular DESC

LIMIT 5;

Explanation:

In COUNT(tag_name) AS most_popular query, AS is used as an alias to give name to created aliases.

Result:



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

```
102 LIMIT 1;
103
104 # To find top five most commonly used hashtags
105 • SELECT
106     tag_name, COUNT(tag_name) AS most_popular
107 FROM
108     tags
109 JOIN
110     photo_tags ON tags.id = photo_tags.tag_id
111 GROUP BY tags.id
112 ORDER BY most_popular DESC
113 LIMIT 5;
114
115 #Most users registered on
116 • SELECT
```

The Results tab shows the following data:

tag_name	most_popular
smile	59
beach	42
party	39
fun	38
concert	24

The bottom of the screenshot shows the Windows taskbar with the date 24-10-2024 and time 18:05.

Conclusion:

Smile, Beach, Party, Fun, Concert are the most popular hashtags used by users on instagram in a descending order.

5. Ad Campaign Launch:

The team wants to know the best day of the week to launch ads.

Your Task: Determine the day of the week when most users register on Instagram. Provide insights on when to schedule an ad campaign.

Query:

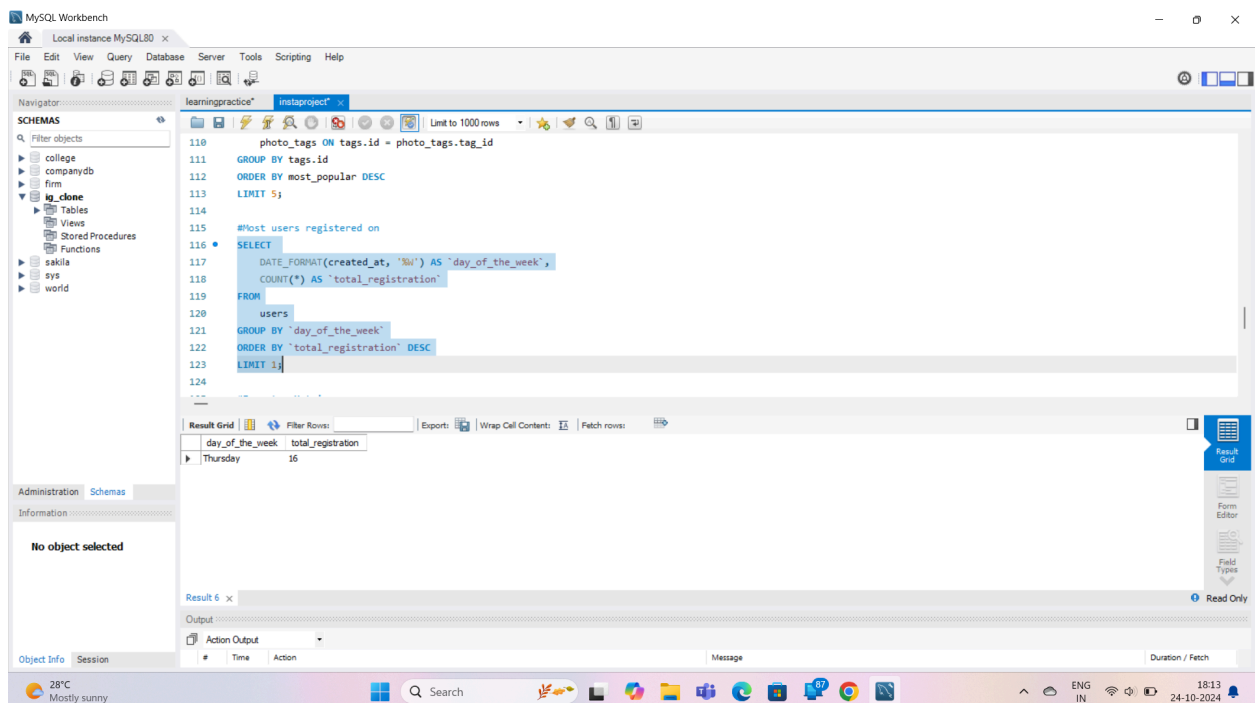
```
SELECT
    DATE_FORMAT(created_at, '%W') AS `day_of_the_week`,
    COUNT(*) AS `total_registration`
FROM
    users
GROUP BY `day_of_the_week`
ORDER BY `total_registration` DESC
LIMIT 1;
```

Explanation:

The DATE_FORMAT() function formats a date as specified.

%W is a parameter used to give weekday names in full (Sunday to Saturday).

Result:



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

```
photo_tags ON tags.id = photo_tags.tag_id
GROUP BY tags.id
ORDER BY most_popular DESC
LIMIT 5;

#Most users registered on
SELECT
    DATE_FORMAT(created_at, '%W') AS `day_of_the_week`,
    COUNT(*) AS `total_registration`
FROM
    users
GROUP BY `day_of_the_week`
ORDER BY `total_registration` DESC
LIMIT 1;
```

The result grid shows the following data:

day_of_the_week	total_registration
Thursday	16

The bottom status bar indicates the system temperature is 28°C, mostly sunny, and the time is 18:13 on 24-10-2024.

Conclusion:

16 registrations which is the most out of all weeks has been done on a thursday. Hence, the ad campaign should be scheduled on a thursday.

B) Investor Metrics:

1. User Engagement:

Investors want to know if users are still active and posting on Instagram or if they are making fewer posts.

Your Task: Calculate the average number of posts per user on Instagram. Also, provide the total number of photos on Instagram divided by the total number of users.

Query:

```
SELECT
    AVG(no_of_posts) AS avg_posts_per_user
FROM
    (SELECT
        user_id, COUNT(*) AS no_of_posts
    FROM
        photos
    GROUP BY user_id
    ORDER BY no_of_posts DESC) AS avg_posts;
```

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

Explanation:

Nested query has been used where an independent query is nested inside a dependent query.

Execution of outer query is dependent on inner query.

/ is used as a mathematical operator to perform division.

Result:

MySQL Workbench screenshot showing a query for average posts per user.

```
124
125 #Investor Metrics
126 # User engagement
127 • SELECT
128     AVG(no_of_posts) AS avg_posts_per_user
129 FROM
130     (SELECT
131         user_id, COUNT(*) AS no_of_posts
132     FROM
133         photos
134     GROUP BY user_id
135     ORDER BY no_of_posts DESC) AS avg_posts;
136
137 # Total number of photos divided by total number of users on instagram
138 • SELECT
```

Result Grid:

avg_posts_per_user
3.4730

MySQL Workbench screenshot showing a query for the ratio of total photos to total users.

```
133     photos
134     GROUP BY user_id
135     ORDER BY no_of_posts DESC) AS avg_posts;
136
137 # Total number of photos divided by total number of users on instagram
138 • SELECT
139     (SELECT
140         COUNT(*)
141     FROM
142         photos) / (SELECT
143         COUNT(*)
144     FROM
145         users) AS avg_posts;
146
147 # Bots and fake accounts. Accounts that have liked every single post on instagram
```

Result Grid:

avg_posts
2.5700

Conclusion:

Average number of posts per user on instagram is 3.4730.

Total number of photos divided by the total number of users on instagram is 2.5700.

2. Bots & Fake Accounts:

Investors want to know if the platform is crowded with fake and dummy accounts.

Your Task: Identify users (potential bots) who have liked every single photo on the site, as this is not typically possible for a normal user.

Query:

```
SELECT
  username, COUNT(*) AS all_likes
FROM
  users u
  JOIN
    likes l ON u.id = l.user_id
GROUP BY l.user_id
HAVING all_likes = (SELECT
  COUNT(*)
FROM
  photos);
```

Explanation:

The HAVING clause was introduced in SQL to allow the filtering of query results based on aggregate functions and groupings.

Result:

SQL Query:

```

144 FROM
145     users) AS avg_posts;
146
147 # Bots and fake accounts. Accounts that have liked every single post on instagram
148 SELECT
149     username, COUNT(*) AS all_likes
150 FROM
151     users u
152 JOIN
153     likes l ON u.id = l.user_id
154 GROUP BY l.user_id
155 HAVING all_likes = (SELECT
156     COUNT(*)
157 FROM
158     photos);

```

Result Grid:

username	all_likes
Arinya_Hackett	257
Jadyn81	257
Rocio33	257
MaxwellHalvorson	257
Olke_Ledner37	257
McKenna17	257
Duane60	257
Julien_Schmidt	257
Mike.Auer 39	257
Nia_Haag	257

Conclusion:

Multiple IDs that have liked all the photos on the app could potentially be bots as its humanly not possible. The result table displays such IDs.