**Learners have to come up with a Report to support the answers to the following questions and suggestions**

Objective Questions

1. Are there any tables with duplicate or missing null values? If so, how would you handle them?

=> -- 1.

SELECT id , COUNT(\*)

FROM users

GROUP BY id

HAVING COUNT(\*) > 1;

SELECT \*

FROM users

WHERE id IS NULL OR username IS NULL OR created\_at IS NULL;

-- 2.

SELECT id, COUNT(\*)

FROM photos

GROUP BY id

HAVING COUNT(\*) > 1;

SELECT \*

FROM photos

WHERE user\_id IS NULL OR image\_url IS NULL;

-- 3.

SELECT id, COUNT(\*)

FROM comments

GROUP BY id

HAVING COUNT(\*) > 1;

SELECT \*

FROM comments

WHERE id IS NULL OR user\_id IS NULL OR photo\_id IS NULL;

-- 4.

SELECT user\_id, photo\_id, COUNT(\*)

FROM likes

GROUP BY user\_id, photo\_id

HAVING COUNT(\*) > 1;

SELECT \*

FROM likes

WHERE user\_id IS NULL OR photo\_id IS NULL;

-- 5.

SELECT follower\_id,followee\_id, COUNT(\*)

FROM follows

GROUP BY follower\_id,followee\_id

HAVING COUNT(\*) > 1;

SELECT \*

FROM follows

WHERE follower\_id IS NULL OR followee\_id IS NULL;

-- 6.

SELECT id , COUNT(\*) AS duplicate\_count

FROM tags

GROUP BY id

HAVING COUNT(\*) > 1;

SELECT \*

FROM tags

WHERE id IS NULL OR created\_at IS NULL OR tag\_name IS NULL;

-- 7.

SELECT photo\_id,tag\_id, COUNT(\*)

FROM photo\_tags

GROUP BY photo\_id,tag\_id

HAVING COUNT(\*) > 1;

SELECT \*

FROM photo\_tags

WHERE photo\_id IS NULL OR tag\_id IS NULL;

-- After identifying i would have used below method for deleting it:

DELETE FROM users

WHERE username IS NULL;

-- Canreplace or update it by:

UPDATE users

SET username = default\_value

WHERE id IS NULL;

#### **Approach**:

* Identify duplicates:
* The query SELECT photo\_id, tag\_id, COUNT(\*) FROM photo\_tags GROUP BY photo\_id, tag\_id HAVING COUNT(\*) > 1; finds duplicates in the photo\_tags table.
* Identify missing values:

The query SELECT \* FROM photo\_tags WHERE photo\_id IS NULL OR tag\_id IS NULL; finds rows with missing photo\_id or tag\_id.

#### Calculation/Analysis:

* Duplicate entries for the same photo-tag combination may lead to redundant relationships.
* Missing photo\_id or tag\_id indicates incomplete tagging.

#### Conclusion:

* For duplicates:

Remove the redundant entries

1. What is the distribution of user activity levels (e.g., number of posts, likes, comments) across the user base?

=> -- User table is connected to others tables like photos, comments, likes

-- photos

select u.id, u.username, count(p.user\_id) as count\_photo\_id

from users u left join photos p

on u.id = p.user\_id

group by u.id, u.username;

-- comments

select u.id, u.username, count(c.user\_id) as count\_comment\_id

from users u left join comments c

on u.id = c.user\_id

group by u.id, u.username;

-- likes

select u.id, u.username, count(l.user\_id) as count\_like\_id

from users u left join likes l

on u.id = l.user\_id

group by u.id, u.username;

-- final query

select u.id, u.username, COALESCE(p.count\_photo\_id, 0) AS count\_photo\_id, COALESCE(c.count\_comment\_id, 0) AS count\_comment\_id, COALESCE(l.count\_like\_id, 0) AS count\_like\_id

from users u

left join (select user\_id, count(\*) as count\_photo\_id from photos group by user\_id) p

ON u.id = p.user\_id

left join (select user\_id, count(\*) as count\_comment\_id from comments group by user\_id) c

ON u.id = c.user\_id

left join (select user\_id, count(\*) as count\_like\_id from likes group by user\_id) l

ON u.id = l.user\_id

group by u.id, u.username;

### **Approach**

The SQL query provided aims to gather a comprehensive view of user activity levels across various interaction types in a social media context. Here's a breakdown of the approach taken in the query:

1. **User Table Selection**: The query starts by selecting the user ID and username from the users table, which serves as the primary source of user information.
2. **Left Joins for Activity Counts**:
   1. **Photos**: A left join is performed with a subquery that counts the number of photos uploaded by each user. This subquery groups the results by user\_id, ensuring that each user's total photo count is calculated.
   2. **Comments**: Another left join is executed with a subquery that counts the number of comments made by each user, similarly grouping by user\_id.
   3. **Likes**: A final left join is made with a subquery that counts the number of likes given by each user, again grouping by user\_id.
3. **Use of COALESCE**: The COALESCE function is used to handle cases where a user has not engaged in a particular activity. If a user has no photos, comments, or likes, the count will default to zero rather than returning a NULL value.
4. **Final Grouping**: The results are grouped by user ID and username to ensure that each user appears only once in the final output.

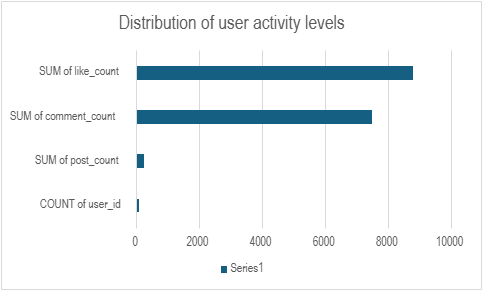
### **Conclusion**

The resulting dataset provides a clear overview of user engagement across the platform by displaying the total counts of photos, comments, and likes associated with each user. This information can be useful for several analytical purposes:

* **User Engagement Analysis**: By examining the counts, stakeholders can identify active users versus inactive ones, allowing for targeted engagement strategies.
* **Content Strategy**: Insights into what types of activities are more common can inform content creation strategies.
* **Feature Development**: Understanding user behavior can guide the development of new features aimed at enhancing user interaction.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | username | count\_photo\_id | count\_comment\_id | count\_like\_id |
| 1 | Kenton\_Kirlin | 5 | 0 | 0 |
| 2 | Andre\_Purdy85 | 4 | 66 | 94 |
| 3 | Harley\_Lind18 | 4 | 67 | 79 |
| 4 | Arely\_Bogan63 | 3 | 64 | 93 |
| 5 | Aniya\_Hackett | 0 | 257 | 257 |
| 6 | Travon.Waters | 5 | 62 | 82 |
| 7 | Kasandra\_Homenick | 0 | 0 | 0 |
| 8 | Tabitha\_Schamberger11 | 4 | 61 | 79 |
| 9 | Gus93 | 4 | 60 | 85 |
| 10 | Presley\_McClure | 3 | 63 | 87 |
| 11 | Justina.Gaylord27 | 5 | 49 | 89 |
| 12 | Dereck65 | 4 | 68 | 77 |
| 13 | Alexandro35 | 5 | 58 | 93 |
| 14 | Jaclyn81 | 0 | 257 | 257 |
| 15 | Billy52 | 4 | 77 | 84 |
| 16 | Annalise.McKenzie16 | 4 | 52 | 103 |
| 17 | Norbert\_Carroll35 | 3 | 83 | 78 |
| 18 | Odessa2 | 1 | 53 | 82 |
| 19 | Hailee26 | 2 | 60 | 90 |
| 20 | Delpha.Kihn | 1 | 67 | 87 |
| 21 | Rocio33 | 0 | 257 | 257 |
| 22 | Kenneth64 | 1 | 67 | 91 |
| 23 | Eveline95 | 12 | 0 | 0 |
| 24 | Maxwell.Halvorson | 0 | 257 | 257 |
| 25 | Tierra.Trantow | 0 | 0 | 0 |
| 26 | Josianne.Friesen | 5 | 69 | 94 |
| 27 | Darwin29 | 1 | 67 | 79 |
| 28 | Dario77 | 4 | 59 | 77 |
| 29 | Jaime53 | 8 | 0 | 0 |
| 30 | Kaley9 | 2 | 65 | 81 |
| 31 | Aiyana\_Hoeger | 1 | 66 | 88 |
| 32 | Irwin.Larson | 4 | 56 | 91 |
| 33 | Yvette.Gottlieb91 | 5 | 61 | 77 |
| 34 | Pearl7 | 0 | 0 | 0 |
| 35 | Lennie\_Hartmann40 | 2 | 67 | 92 |
| 36 | Ollie\_Ledner37 | 0 | 257 | 257 |
| 37 | Yazmin\_Mills95 | 1 | 63 | 84 |
| 38 | Jordyn.Jacobson2 | 2 | 58 | 85 |
| 39 | Kelsi26 | 1 | 67 | 89 |
| 40 | Rafael.Hickle2 | 1 | 68 | 85 |
| 41 | Mckenna17 | 0 | 257 | 257 |
| 42 | Maya.Farrell | 3 | 54 | 87 |
| 43 | Janet.Armstrong | 5 | 72 | 86 |
| 44 | Seth46 | 4 | 60 | 86 |
| 45 | David.Osinski47 | 0 | 0 | 0 |
| 46 | Malinda\_Streich | 4 | 68 | 88 |
| 47 | Harrison.Beatty50 | 5 | 59 | 76 |
| 48 | Granville\_Kutch | 1 | 55 | 75 |
| 49 | Morgan.Kassulke | 0 | 0 | 0 |
| 50 | Gerard79 | 3 | 69 | 81 |
| 51 | Mariano\_Koch3 | 5 | 0 | 0 |
| 52 | Zack\_Kemmer93 | 5 | 56 | 85 |
| 53 | Linnea59 | 0 | 0 | 0 |
| 54 | Duane60 | 0 | 257 | 257 |
| 55 | Meggie\_Doyle | 1 | 66 | 78 |
| 56 | Peter.Stehr0 | 1 | 68 | 81 |
| 57 | Julien\_Schmidt | 0 | 257 | 257 |
| 58 | Aurelie71 | 8 | 0 | 0 |
| 59 | Cesar93 | 10 | 0 | 0 |
| 60 | Sam52 | 2 | 72 | 86 |
| 61 | Jayson65 | 1 | 58 | 83 |
| 62 | Ressie\_Stanton46 | 2 | 58 | 88 |
| 63 | Elenor88 | 4 | 80 | 83 |
| 64 | Florence99 | 5 | 0 | 0 |
| 65 | Adelle96 | 5 | 60 | 96 |
| 66 | Mike.Auer39 | 0 | 257 | 257 |
| 67 | Emilio\_Bernier52 | 3 | 76 | 86 |
| 68 | Franco\_Keebler64 | 0 | 0 | 0 |
| 69 | Karley\_Bosco | 1 | 69 | 97 |
| 70 | Erick5 | 1 | 69 | 88 |
| 71 | Nia\_Haag | 0 | 257 | 257 |
| 72 | Kathryn80 | 5 | 64 | 85 |
| 73 | Jaylan.Lakin | 1 | 63 | 86 |
| 74 | Hulda.Macejkovic | 0 | 0 | 0 |
| 75 | Leslie67 | 0 | 257 | 257 |
| 76 | Janelle.Nikolaus81 | 0 | 257 | 257 |
| 77 | Donald.Fritsch | 6 | 0 | 0 |
| 78 | Colten.Harris76 | 5 | 60 | 83 |
| 79 | Katarina.Dibbert | 1 | 68 | 75 |
| 80 | Darby\_Herzog | 0 | 0 | 0 |
| 81 | Esther.Zulauf61 | 0 | 0 | 0 |
| 82 | Aracely.Johnston98 | 2 | 67 | 84 |
| 83 | Bartholome.Bernhard | 0 | 0 | 0 |
| 84 | Alysa22 | 2 | 76 | 75 |
| 85 | Milford\_Gleichner42 | 2 | 57 | 87 |
| 86 | Delfina\_VonRueden68 | 9 | 0 | 0 |
| 87 | Rick29 | 4 | 74 | 92 |
| 88 | Clint27 | 11 | 0 | 0 |
| 89 | Jessyca\_West | 0 | 0 | 0 |
| 90 | Esmeralda.Mraz57 | 0 | 0 | 0 |
| 91 | Bethany20 | 0 | 257 | 257 |
| 92 | Frederik\_Rice | 3 | 61 | 91 |
| 93 | Willie\_Leuschke | 2 | 63 | 91 |
| 94 | Damon35 | 1 | 68 | 84 |
| 95 | Nicole71 | 2 | 68 | 86 |
| 96 | Keenan.Schamberger60 | 3 | 75 | 98 |
| 97 | Tomas.Beatty93 | 2 | 68 | 69 |
| 98 | Imani\_Nicolas17 | 1 | 65 | 74 |
| 99 | Alek\_Watsica | 3 | 68 | 74 |
| 100 | Javonte83 | 2 | 70 | 82 |

|  |  |  |  |
| --- | --- | --- | --- |
| COUNT of user\_id | SUM of post\_count | SUM of comment\_count | SUM of like\_count |
| 100 | 257 | 7488 | 8782 |



1. Calculate the average number of tags per post (photo\_tags and photos tables).

=> SELECT AVG(tag\_count) AS average\_tags

FROM (

SELECT p.id , COUNT(t.tag\_id) AS tag\_count

FROM photos p

LEFT JOIN photo\_tags t ON p.id = t.photo\_id

GROUP BY p.id

) AS photo\_tag\_counts;

### **Approach:**

To calculate the **average number of tags per post** (where a post is represented by a row in the photos table), I am using a two-step process:

1. **Count the Tags for Each Photo:**
   1. First, I join the photos table (p) with the photo\_tags table (t) using a LEFT JOIN. This ensures that even photos without tags are included in the result.
   2. The COUNT(t.tag\_id) counts the number of tags associated with each photo (i.e., each photo\_id). If a photo has no tags, the count will be zero.
   3. I then group the results by p.id (the photo\_id), so we get a count of tags for each photo.
2. **Calculate the Average:**
   1. After calculating the number of tags for each photo, I calculate the average of these tag counts using the outer query SELECT AVG(tag\_count) AS average\_tags.
   2. This gives the average number of tags per photo across the entire dataset.

### **Conclusion:**

* The final query effectively computes the **average number of tags per photo**.
  + **If all photos had the same number of tags**, the result would be that number.
  + **If some photos had no tags**, those would be included in the average, so it could lower the overall average if a significant number of photos have no tags.

1. Identify the top users with the highest engagement rates (likes, comments) on their posts and rank them.

=> select u.id as Users, MAX(c.comment\_count) as Max\_comment, Max(l.like\_count) as Max\_likes

from users u

left join

(select user\_id, count(id) as comment\_count from comments group by user\_id) c

ON u.id = c.user\_id

left join

(select user\_id, count(photo\_id) as like\_count from likes group by user\_id) l

ON u.id = l.user\_id

group by Users

order by Max\_comment desc, Max\_likes desc Limit 2;

**Approach:**

To identify the top users with the highest engagement rates based on **likes** and **comments** on their posts, we need to look at two things:

1. **Comments Per User:**
   1. We need to count how many comments each user has received. This is done by counting the id (comment IDs) from the comments table, grouped by user\_id.
2. **Likes Per User:**
   1. Similarly, we need to count how many likes each user has received on their posts. This is done by counting the photo\_id (like entries) from the likes table, grouped by user\_id.

Finally, we need to rank users by **both the maximum number of comments and likes** they've received and return the top users.

### **Conclusion:**

* This query ranks the top users based on the maximum number of **comments** and **likes** they have received on their posts.
  + **Sorting Logic:** Users are first ranked by the number of comments (highest first), and in the case of a tie, they are then ranked by the number of likes.
  + The LIMIT 2 ensures that only the top 2 users are returned.
* This query provides a clear view of which users are most engaged with others based on the interactions their posts receive.

1. Which users have the highest number of followers and followings?

=> SELECT

u.id AS user\_id,

u.username AS username,

COUNT(f.followee\_id) AS followings\_count,

COUNT(fw.follower\_id) AS followers\_count

FROM

users u

LEFT JOIN

follows f ON u.id = f.followee\_id

LEFT JOIN

follows fw ON u.id = fw.follower\_id

GROUP BY

u.id, u.username

ORDER BY

followings\_count DESC

LIMIT 1;

### **Approach:**

To determine which user has the highest number of followers and followings, we need to analyze the follows table, which stores the relationships between users. The follows table typically contains two important columns: follower\_id (the user following) and followee\_id (the user being followed).

### **Steps to Calculate:**

1. **Followers Count:**
   1. We can find how many users are following a particular user by counting the number of times a user appears as a followee\_id in the follows table. This count represents the number of followers that user has.
2. **Followings Count:**
   1. Similarly, to find out how many users a particular user is following, we count the number of times a user appears as a follower\_id in the follows table. This gives the number of followings for the user.

### **Conclusion:**

* This query identifies the **user with the highest number of followings** by counting how many people each user is following.
* The ORDER BY followings\_count DESC ensures that we rank users by their followings, and LIMIT 1 returns only the top user with the most followings.
* The query also returns the corresponding followers\_count for the user, but the primary ranking is based on the number of people they are following.

Thus, the result will show the user who is following the most people, along with the count of their followers and followings.

1. Calculate the average engagement rate (likes, comments) per post for each user.

=> select u.id, u.username , AVG(l.photo\_id) as avg\_likes, AVG(c.id) as avg\_comments

from users u

left join likes l

ON u.id = l.user\_id

left join comments c

ON u.id = c.user\_id

group by u.id;

### **Approach:**

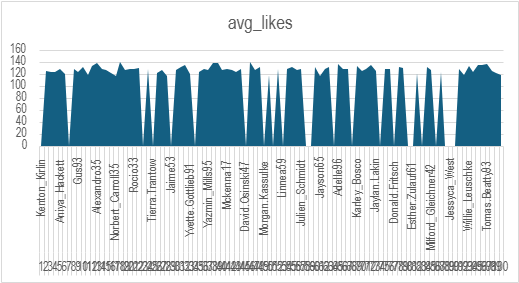
The objective here is to calculate the **average engagement rate per post** for each user based on the number of **likes** and **comments**. We can define engagement as the sum of likes and comments on a post. Here's how the query works:

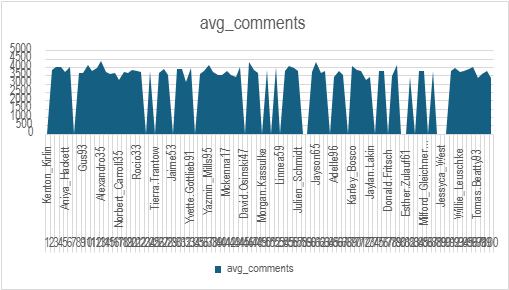
1. **Join Users with Likes and Comments:**
   1. The users table is joined with the likes and comments tables using LEFT JOIN.
      1. likes table is joined on u.id = l.user\_id to count the number of likes.
      2. comments table is joined on u.id = c.user\_id to count the number of comments.
   2. This ensures that each user is included in the result, even if they have no likes or comments (because of the LEFT JOIN).
2. **Average Calculation:**
   1. AVG(l.photo\_id) calculates the average number of likes per post for each user. This works because each row in the likes table corresponds to a like on a photo (or post).
   2. AVG(c.id) calculates the average number of comments per post for each user.
   3. Both values are averaged across all posts that the user has published.
3. **Grouping:**
   1. The GROUP BY u.id groups the results by user, ensuring each user has one row with their calculated average likes and comments per post.

### **Conclusion:**

* The query calculates the **average number of likes** and **comments** per post for each user.
  + The avg\_likes column shows the average number of likes each user’s posts have received.
  + The avg\_comments column shows the average number of comments each user’s posts have received.
* This helps in understanding the **engagement rate per user**, allowing us to compare users based on how much interaction (likes and comments) their posts are generating.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | username | avg\_likes | avg\_comments |  |
| 1 | Kenton\_Kirlin |  |  |  |
| 2 | Andre\_Purdy85 | 126.3191 | 3844.9697 |  |
| 3 | Harley\_Lind18 | 124.3797 | 4015.1194 |  |
| 4 | Arely\_Bogan63 | 123.6989 | 3993.0156 |  |
| 5 | Aniya\_Hackett | 129.0000 | 3728.3074 |  |
| 6 | Travon.Waters | 120.2317 | 3998.2258 |  |
| 7 | Kasandra\_Homenick |  |  |  |
| 8 | Tabitha\_Schamberger11 | 128.8481 | 3648.6066 |  |
| 9 | Gus93 | 124.6941 | 3637.1000 |  |
| 10 | Presley\_McClure | 131.6207 | 4109.1587 |  |
| 11 | Justina.Gaylord27 | 118.7865 | 3761.7755 |  |
| 12 | Dereck65 | 133.4805 | 3921.1471 |  |
| 13 | Alexandro35 | 138.7957 | 4341.2414 |  |
| 14 | Jaclyn81 | 129.0000 | 3730.9455 |  |
| 15 | Billy52 | 126.8333 | 3582.8571 |  |
| 16 | Annalise.McKenzie16 | 122.8641 | 3658.0962 |  |
| 17 | Norbert\_Carroll35 | 117.0769 | 3232.6867 |  |
| 18 | Odessa2 | 139.9146 | 3673.0755 |  |
| 19 | Hailee26 | 126.5667 | 3670.9667 |  |
| 20 | Delpha.Kihn | 129.5402 | 3849.0448 |  |
| 21 | Rocio33 | 129.0000 | 3733.4708 |  |
| 22 | Kenneth64 | 130.3187 | 3690.7612 |  |
| 23 | Eveline95 |  |  |  |
| 24 | Maxwell.Halvorson | 129.0000 | 3734.7315 |  |
| 25 | Tierra.Trantow |  |  |  |
| 26 | Josianne.Friesen | 122.5957 | 3644.5652 |  |
| 27 | Darwin29 | 127.5063 | 3900.5522 |  |
| 28 | Dario77 | 116.9351 | 3546.2373 |  |
| 29 | Jaime53 |  |  |  |
| 30 | Kaley9 | 127.6296 | 3884.9692 |  |
| 31 | Aiyana\_Hoeger | 131.8977 | 3901.3485 |  |
| 32 | Irwin.Larson | 135.9341 | 3080.3214 |  |
| 33 | Yvette.Gottlieb91 | 120.7532 | 3929.5410 |  |
| 34 | Pearl7 |  |  |  |
| 35 | Lennie\_Hartmann40 | 123.8913 | 3560.1194 |  |
| 36 | Ollie\_Ledner37 | 129.0000 | 3737.7160 |  |
| 37 | Yazmin\_Mills95 | 126.7500 | 4134.8413 |  |
| 38 | Jordyn.Jacobson2 | 139.4353 | 3676.5690 |  |
| 39 | Kelsi26 | 138.2472 | 3500.5522 |  |
| 40 | Rafael.Hickle2 | 126.6941 | 3504.9706 |  |
| 41 | Mckenna17 | 129.0000 | 3739.7121 |  |
| 42 | Maya.Farrell | 128.0690 | 3537.9074 |  |
| 43 | Janet.Armstrong | 123.6395 | 3398.5556 |  |
| 44 | Seth46 | 129.0233 | 3988.7000 |  |
| 45 | David.Osinski47 |  |  |  |
| 46 | Malinda\_Streich | 139.9773 | 4306.7353 |  |
| 47 | Harrison.Beatty50 | 127.3026 | 3824.8983 |  |
| 48 | Granville\_Kutch | 132.2000 | 3620.3636 |  |
| 49 | Morgan.Kassulke |  |  |  |
| 50 | Gerard79 | 118.3333 | 3818.1884 |  |
| 51 | Mariano\_Koch3 |  |  |  |
| 52 | Zack\_Kemmer93 | 128.3647 | 4024.0536 |  |
| 53 | Linnea59 |  |  |  |
| 54 | Duane60 | 129.0000 | 3742.6304 |  |
| 55 | Meggie\_Doyle | 132.3974 | 4040.9242 |  |
| 56 | Peter.Stehr0 | 126.9383 | 3939.5588 |  |
| 57 | Julien\_Schmidt | 129.0000 | 3744.1518 |  |
| 58 | Aurelie71 |  |  |  |
| 59 | Cesar93 |  |  |  |
| 60 | Sam52 | 132.4651 | 3697.3056 |  |
| 61 | Jayson65 | 116.8193 | 4323.9138 |  |
| 62 | Ressie\_Stanton46 | 128.8068 | 3627.6379 |  |
| 63 | Elenor88 | 131.6988 | 3768.9375 |  |
| 64 | Florence99 |  |  |  |
| 65 | Adelle96 | 137.8542 | 3373.9833 |  |
| 66 | Mike.Auer39 | 129.0000 | 3746.4280 |  |
| 67 | Emilio\_Bernier52 | 128.6628 | 3534.3684 |  |
| 68 | Franco\_Keebler64 |  |  |  |
| 69 | Karley\_Bosco | 133.7010 | 4075.3768 |  |
| 70 | Erick5 | 126.2045 | 3848.7681 |  |
| 71 | Nia\_Haag | 129.0000 | 3748.2607 |  |
| 72 | Kathryn80 | 135.8941 | 3220.7500 |  |
| 73 | Jaylan.Lakin | 126.4535 | 3375.9841 |  |
| 74 | Hulda.Macejkovic |  |  |  |
| 75 | Leslie67 | 129.0000 | 3749.7549 |  |
| 76 | Janelle.Nikolaus81 | 129.0000 | 3750.7549 |  |
| 77 | Donald.Fritsch |  |  |  |
| 78 | Colten.Harris76 | 131.8193 | 3466.2833 |  |
| 79 | Katarina.Dibbert | 130.9067 | 4112.2794 |  |
| 80 | Darby\_Herzog |  |  |  |
| 81 | Esther.Zulauf61 |  |  |  |
| 82 | Aracely.Johnston98 | 121.6548 | 3375.6866 |  |
| 83 | Bartholome.Bernhard |  |  |  |
| 84 | Alysa22 | 131.9467 | 3787.1579 |  |
| 85 | Milford\_Gleichner42 | 127.9655 | 3790.0175 |  |
| 86 | Delfina\_VonRueden68 |  |  |  |
| 87 | Rick29 | 124.5761 | 3778.7703 |  |
| 88 | Clint27 |  |  |  |
| 89 | Jessyca\_West |  |  |  |
| 90 | Esmeralda.Mraz57 |  |  |  |
| 91 | Bethany20 | 129.0000 | 3753.3191 |  |
| 92 | Frederik\_Rice | 118.4945 | 3947.6230 |  |
| 93 | Willie\_Leuschke | 133.6923 | 3708.1587 |  |
| 94 | Damon35 | 124.5833 | 3788.3824 |  |
| 95 | Nicole71 | 135.2791 | 3885.5588 |  |
| 96 | Keenan.Schamberger60 | 135.1837 | 4000.1867 |  |
| 97 | Tomas.Beatty93 | 136.7536 | 3350.1471 |  |
| 98 | Imani\_Nicolas17 | 125.2027 | 3577.8000 |  |
| 99 | Alek\_Watsica | 121.8649 | 3772.2206 |  |
| 100 | Javonte83 | 118.9268 | 3338.3286 |  |





1. Get the list of users who have never liked any post (users and likes tables)

=> select id as user\_id, username from users

where id NOT IN (select user\_id from likes);

### **Approach:**

To identify the users who have **never liked any post**, we can follow these steps:

1. **Check the Likes Table:**
   1. The likes table contains records of users liking posts. Specifically, the user\_id in the likes table represents the user who liked a post.
2. **Find Users Who Are Not in the Likes Table:**
   1. We can use a NOT IN subquery to identify users who do not appear in the likes table. The subquery SELECT user\_id FROM likes gives us a list of users who have liked at least one post. Using NOT IN with this subquery in the main query will return users who have never liked a post.

### **Conclusion:**

* This query returns a list of users who have **never liked any post**.
  + It does so by identifying users whose user\_id does not appear in the likes table.
  + The result will include all users from the users table who have not engaged in any liking activity on posts.

1. How can you leverage user-generated content (posts, hashtags, photo tags) to create more personalized and engaging ad campaigns?

=> SELECT t.tag\_name, COUNT(\*) AS total\_posts, AVG(COALESCE(likes\_count.likes, 0) + COALESCE(comments\_count.comments, 0)) AS avg\_engagement

FROM tags t

JOIN photo\_tags pt ON t.id = pt.tag\_id

JOIN photos p ON pt.photo\_id = p.id

LEFT JOIN

(SELECT photo\_id, COUNT(\*) AS likes FROM likes GROUP BY photo\_id) AS likes\_count ON p.id = likes\_count.photo\_id

LEFT JOIN

(SELECT photo\_id, COUNT(\*) AS comments FROM comments GROUP BY photo\_id) AS comments\_count ON p.id = comments\_count.photo\_id

GROUP BY t.tag\_name

having avg\_engagement >= 64

ORDER BY avg\_engagement DESC;

### **Approach:**

To create more **personalized and engaging ad campaigns** using **user-generated content** (UGC), such as posts, hashtags, and photo tags, we can focus on analyzing the performance of posts based on their engagement levels (likes and comments) tied to specific **hashtags**. Here's how the query works:

1. **Tag Performance:**
   1. The tags table is used to get the most popular tags that users are associating with their posts. Tags are often used to categorize content and can be used to identify trending topics or themes.
2. **Engagement Metrics:**
   1. The query calculates the **total number of posts** that are associated with each tag.
   2. It also calculates the **average engagement** for each tag, which is the sum of likes and comments (calculated as likes\_count.likes + comments\_count.comments) for each post associated with that tag.
   3. **COALESCE** is used to ensure that even posts with zero likes or comments are considered.
3. **Filter for High Engagement:**
   1. The HAVING avg\_engagement >= 64 condition filters out tags that have an average engagement below 64, focusing on tags that generate relatively higher engagement.
4. **Ordering for Insights:**
   1. The query orders the tags by average engagement in descending order, giving a ranking of tags with the highest engagement.

### **Approach:**

To create more **personalized and engaging ad campaigns** using **user-generated content** (UGC), such as posts, hashtags, and photo tags, we can focus on analyzing the performance of posts based on their engagement levels (likes and comments) tied to specific **hashtags**. Here's how the query works:

1. **Tag Performance:**
   1. The tags table is used to get the most popular tags that users are associating with their posts. Tags are often used to categorize content and can be used to identify trending topics or themes.
2. **Engagement Metrics:**
   1. The query calculates the **total number of posts** that are associated with each tag.
   2. It also calculates the **average engagement** for each tag, which is the sum of likes and comments (calculated as likes\_count.likes + comments\_count.comments) for each post associated with that tag.
   3. **COALESCE** is used to ensure that even posts with zero likes or comments are considered.
3. **Filter for High Engagement:**
   1. The HAVING avg\_engagement >= 64 condition filters out tags that have an average engagement below 64, focusing on tags that generate relatively higher engagement.
4. **Ordering for Insights:**
   1. The query orders the tags by average engagement in descending order, giving a ranking of tags with the highest engagement.

### **Conclusion:**

This query allows you to **leverage user-generated content (UGC)** to create more **personalized and engaging ad campaigns** by:

1. **Identifying High-Engagement Tags:** The query helps pinpoint which hashtags are driving the most engagement (likes and comments) in user posts. This allows you to target specific themes or interests that resonate with your audience.
2. **Focus on Relevant Topics:** By selecting tags with high engagement, you can create ads that align with trending topics or interests, increasing the chances of user interaction and brand recall.
3. **Content Personalization:** Knowing which tags are generating the most interaction helps you tailor your ads around those specific themes, ensuring your ad content is relevant to the audience who is already engaging with related posts.
4. Are there any correlations between user activity levels and specific content types (e.g., photos, videos, reels)? How can this information guide content creation and curation strategies?

=> SELECT

u.username,

COUNT(p.id) AS total\_photos,

COUNT(l.photo\_id) AS total\_likes,

COUNT(c.id) AS total\_comments,

AVG(CASE WHEN l.photo\_id IS NOT NULL THEN 1 ELSE 0 END) AS avg\_likes\_per\_photo,

AVG(CASE WHEN c.id IS NOT NULL THEN 1 ELSE 0 END) AS avg\_comments\_per\_photo

FROM users u

LEFT JOIN photos p ON u.id = p.user\_id

LEFT JOIN likes l ON p.id = l.photo\_id

LEFT JOIN comments c ON p.id = c.photo\_id

GROUP BY u.id

ORDER BY total\_photos DESC, total\_likes DESC, total\_comments DESC;

### **Approach:**

To understand the correlation between **user activity levels** and **specific content types** (such as photos, videos, and reels), we can focus on analyzing the engagement levels (likes and comments) based on the **content type** that users are posting. The query you provided calculates user activity metrics across different content types, specifically **photos** in this case, and provides insights on how these activities correlate with engagement (likes and comments). Here's a breakdown of the approach:

### **Steps to Calculate:**

1. **Content and Engagement Analysis:**
   1. The query counts how many **photos** each user has posted (COUNT(p.id)).
   2. It then calculates how many **likes** (COUNT(l.photo\_id)) and **comments** (COUNT(c.id)) each photo has received.
2. **Average Engagement Per Post:**
   1. The query calculates the **average number of likes per photo** (AVG(CASE WHEN l.photo\_id IS NOT NULL THEN 1 ELSE 0 END)), showing how many photos a user has received likes on, relative to the total number of their photos.
   2. Similarly, the **average number of comments per photo** is calculated (AVG(CASE WHEN c.id IS NOT NULL THEN 1 ELSE 0 END)), indicating how often photos have received comments relative to the total posts.
3. **Grouping by User:**
   1. The GROUP BY u.id ensures that we calculate these metrics for each user individually, giving us a clear understanding of how their activity on the platform correlates with engagement.
4. **Sorting by Activity Levels:**
   1. The results are ordered by **total photos**, **total likes**, and **total comments** in descending order to show the most active users in terms of content creation and interaction.

### **Conclusion:**

1. **Identifying Active Users**:
   1. The query allows us to identify users who are **most active** in terms of posting photos. These users can be analyzed for their content type and style to understand what type of posts generate more engagement.
2. **Understanding Engagement**:
   1. By calculating both **total likes/comments** and **average likes/comments per post**, we can discern how engaging a user's content is. A high number of posts but low engagement might suggest that their content is not resonating well, while high engagement per post suggests their content is particularly popular.
3. **Content Strategy Insights**:
   1. **Content Creation**: If certain users receive high engagement (e.g., lots of likes/comments per photo), their style or topic could be analyzed and replicated for more content creation.
   2. **Content Curation**: If certain content types (like photos, videos, or reels) tend to perform better, the strategy could be adjusted to focus on these content types. For example, if videos have more engagement than photos, you might curate more video content for the audience.
4. **Improving Engagement**:
   1. By understanding the relationship between activity (how much content is being posted) and engagement (likes/comments per post), platforms or brands can tailor content strategies more effectively. If users with a high post count are getting fewer interactions, content optimization can be considered to improve those metrics.
5. Calculate the total number of likes, comments, and photo tags for each user.

=> select u.id as User\_id, u.username as username, count(distinct l.photo\_id) as like\_count,

count(distinct c.id) as comment\_count, count(distinct pt.tag\_id) as tag\_count

from users u

left join photos p

ON u.id = p.user\_id

left join likes l

ON u.id = l.user\_id

left join comments c

ON u.id = c.user\_id

left join photo\_tags pt

ON p.user\_id = pt.photo\_id

group by User\_id;

### **Approach:**

To calculate the **total number of likes**, **comments**, and **photo tags** for each user, we need to join relevant tables and perform **counting** operations for each of these activities. Here’s how we approach this:

1. **Data from Users and Photos:**
   1. We begin by selecting user information (id and username) from the users table.
   2. The query links the users table to the photos table (using p.user\_id = u.id) to retrieve each user's posts.
2. **Likes Count:**
   1. The likes table is then joined with the photos table (l.photo\_id = p.id). We count how many **distinct likes** each photo has received by using COUNT(DISTINCT l.photo\_id). This gives the total number of unique photos liked by each user.
3. **Comments Count:**
   1. Similarly, we join the comments table (c.user\_id = u.id) to count the **distinct comments** made by each user on their photos. The COUNT(DISTINCT c.id) counts all unique comments per user.
4. **Tags Count:**
   1. The photo\_tags table is joined to the photos table (pt.photo\_id = p.id). We count the **distinct tags** applied to each user's photos using COUNT(DISTINCT pt.tag\_id).
5. **Group by User:**
   1. The GROUP BY u.id ensures that we calculate these counts for each individual user.

### **Conclusion:**

This query provides a comprehensive breakdown of each user's **engagement activities** by counting:

* **Likes:** The total number of distinct photos that have been liked by the user.
* **Comments:** The total number of distinct comments the user has made on their photos.
* **Tags:** The number of unique tags associated with the user’s photos.

1. Rank users based on their total engagement (likes, comments, shares) over a month.

=> With Engagment AS

(select u.id as User\_Id, u.username as Username,

c.comment\_count as comment\_count, l.like\_count as like\_count ,

(c.comment\_count + l.like\_count) as engagment\_count

from users u

left join

(select user\_id, COUNT(id) as comment\_count from comments

where created\_at between '2024-10-1' and '2024-10-30'

group by user\_id) c

ON u.id = c.user\_id

left join

(select user\_id, COUNT(photo\_id) as like\_count from likes

where created\_at between '2024-10-1' and '2024-10-30'

group by user\_id)l

ON u.id = l.user\_id )

select User\_id, username, comment\_count,like\_count, engagment\_count,

Dense\_rank() over (order by engagment\_count desc) as Engangment\_rank

from Engagment

order by Engangment\_rank;

### **Approach:**

To rank users based on their total **engagement** (likes and comments) over a specific period (e.g., the month of October 2024), we need to:

1. **Join the Users Table with Comments and Likes:**
   1. We first retrieve users (u.id, u.username) from the users table.
   2. Then, we join this table with the comments and likes tables, filtered for the date range between **October 1, 2024**, and **October 30, 2024**.
2. **Count Comments and Likes:**
   1. We calculate the total number of comments (comment\_count) and likes (like\_count) for each user within this time period.
   2. The COUNT() function is used to count the distinct comments and likes for each user.
3. **Calculate Total Engagement:**
   1. Total **engagement** is the sum of **likes** and **comments** (engagement\_count), which gives a measure of overall user interaction with the content.
4. **Ranking the Users:**
   1. We use DENSE\_RANK() to rank users based on their **engagement count**. The users with the highest engagement will be ranked first.
   2. The DENSE\_RANK() function ensures that if multiple users have the same engagement, they receive the same rank without gaps.
5. **Display the Results:**
   1. The final result will display the user ID, username, comment count, like count, total engagement, and their rank, ordered by engagement in descending order.

### **Conclusion:**

This query calculates and ranks users based on their engagement over a given time period. By analyzing the **engagement count** (likes + comments), this helps identify which users are the most active and engaged. This information can be useful for **targeted marketing campaigns**, **content recommendations**, or **recognition** of top contributors in the community.

1. Retrieve the hashtags that have been used in posts with the highest average number of likes. Use a CTE to calculate the average likes for each hashtag first.

=> WITH HashtagLikes AS (

SELECT ht.tag\_name, COUNT(l.photo\_id) AS total\_likes, COUNT(DISTINCT p.id) AS total\_posts

FROM tags ht

JOIN photo\_tags pt ON ht.id = pt.tag\_id

JOIN photos p ON pt.photo\_id = p.id

LEFT JOIN likes l ON p.id = l.photo\_id

GROUP BY ht.tag\_name

),

AverageLikesPerHashtag AS (

SELECT tag\_name, ROUND((CAST(total\_likes AS FLOAT) / total\_posts),2) AS avg\_likes

FROM HashtagLikes

)

SELECT tag\_name, avg\_likes

FROM AverageLikesPerHashtag

group by tag\_name

having avg\_likes >= 34.5

ORDER BY avg\_likes DESC;

### **Approach:**

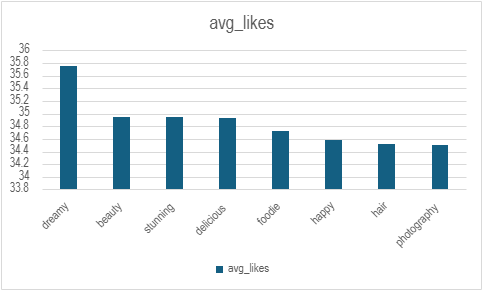
To retrieve the hashtags that have been used in posts with the highest average number of likes, we will:

1. **Calculate Total Likes and Posts Per Hashtag:**
   1. First, we'll join the tags table (which contains hashtag data) with the photo\_tags table (which links hashtags to photos), and the photos table (which holds the actual posts).
   2. Using the LEFT JOIN with the likes table, we'll count the number of likes each post has received and group this information by the hashtag (tag\_name).
2. **Calculate the Average Likes per Hashtag:**
   1. We'll create a second Common Table Expression (CTE) to calculate the **average number of likes per post** for each hashtag. This is done by dividing the **total likes** by the **total number of posts** using the formula: *avg\_likes=total\_likestotal\_posts\text{avg\\_likes} = \frac{\text{total\\_likes}}{\text{total\\_posts}}*avg\_likes=total\_poststotal\_likes
   2. We'll also round the result to two decimal places for clarity.
3. **Filter Hashtags by Average Likes:**
   1. We will then filter the hashtags that have an average like count of **34.5** or more.
4. **Return the Results:**
   1. The final query will return the hashtags (tag\_name) along with their average number of likes (avg\_likes), ordered by the average likes in descending order.

### **Conclusion:**

This query provides a list of hashtags that have the highest average number of likes per post, helping identify popular and engaging content. The results can be used to guide content strategy, target specific hashtags for advertising, or optimize posts to include high-engagement hashtags.

|  |  |
| --- | --- |
| tag\_name | avg\_likes |
| dreamy | 35.75 |
| beauty | 34.95 |
| stunning | 34.94 |
| delicious | 34.93 |
| foodie | 34.73 |
| happy | 34.59 |
| hair | 34.52 |
| photography | 34.5 |



1. Retrieve the users who have started following someone after being followed by that person

=> SELECT

f1.follower\_id AS user\_id,

u.username

FROM

follows f1

JOIN

follows f2 ON f1.follower\_id = f2.followee\_id

JOIN

users u ON f1.follower\_id = u.id

WHERE

f1.created\_at > f2.created\_at

GROUP BY

f1.follower\_id, u.username;

### **Approach:**

To retrieve the users who have started following someone **after** being followed by that person, we need to:

1. **Self-Join the follows Table:**
   1. We'll perform a self-join on the follows table.
      1. f1 will represent the follower-followee relationship where a user is following another.
      2. f2 will represent the reverse relationship, where the same user is followed by the other person.
   2. The follows table will be aliased as f1 and f2, where:
      1. f1.follower\_id is the user who follows the person (f2.followee\_id).
      2. f2.follower\_id is the user who originally followed f1.
2. **Compare created\_at Timestamps:**
   1. We compare the created\_at timestamps of the two follow events:
      1. The user in f1 (follower\_id) should have started following the user in f2 **after** the user in f2 followed them.
      2. The condition is f1.created\_at > f2.created\_at, which ensures that the follow event in f1 occurred after the event in f2.
3. **Retrieve User Details:**
   1. Once the condition is satisfied, we retrieve the user ID (f1.follower\_id) and their username (u.username) by joining the users table to get the usernames of the users.
4. **Group by User ID and Username:**
   1. We'll group the results by f1.follower\_id and u.username to ensure distinct entries for users.

### **Conclusion:**

This query identifies users who have followed someone **after** being followed by that same person. This can provide insights into user interactions and reciprocal following patterns, which could be useful for analyzing social behavior or building engagement strategies.

Subjective Questions

1. Based on user engagement and activity levels, which users would you consider the most loyal or valuable? How would you reward or incentivize these users?

=> WITH TotalLikes AS (

SELECT u.id, COUNT(distinct l.photo\_id) AS total\_likes

FROM users u

LEFT JOIN likes l ON u.id = l.user\_id

GROUP BY u.id

),

TotalComments AS (

SELECT u.id, COUNT(distinct c.photo\_id) AS total\_comments

FROM users u

LEFT JOIN comments c ON u.id = c.user\_id

GROUP BY u.id

),

PhotosPosted AS (

SELECT user\_id, COUNT(id) AS total\_photos\_posted

FROM photos

GROUP BY user\_id

),

Followers AS (

SELECT followee\_id AS user\_id, COUNT(follower\_id) AS total\_followers

FROM follows

GROUP BY followee\_id

),

UniqueTags AS (

SELECT p.user\_id, COUNT(DISTINCT pt.tag\_id) AS unique\_tags\_used

FROM photos p

LEFT JOIN photo\_tags pt ON p.id = pt.photo\_id

GROUP BY p.user\_id

)

SELECT u.id AS user\_id, u.username,

COALESCE(tl.total\_likes, 0) AS total\_likes,

COALESCE(tc.total\_comments, 0) AS total\_comments,

COALESCE(pp.total\_photos\_posted, 0) AS total\_photos\_posted,

COALESCE(f.total\_followers, 0) AS total\_followers,

COALESCE(ut.unique\_tags\_used, 0) AS unique\_tags\_used,

(COALESCE(tl.total\_likes, 0) + COALESCE(tc.total\_comments, 0)) AS total\_engagement

FROM users u

LEFT JOIN TotalLikes tl ON u.id = tl.id

LEFT JOIN TotalComments tc ON u.id = tc.id

LEFT JOIN PhotosPosted pp ON u.id = pp.user\_id

LEFT JOIN Followers f ON u.id = f.user\_id

LEFT JOIN UniqueTags ut ON u.id = ut.user\_id

GROUP BY u.id

having total\_photos\_posted >0

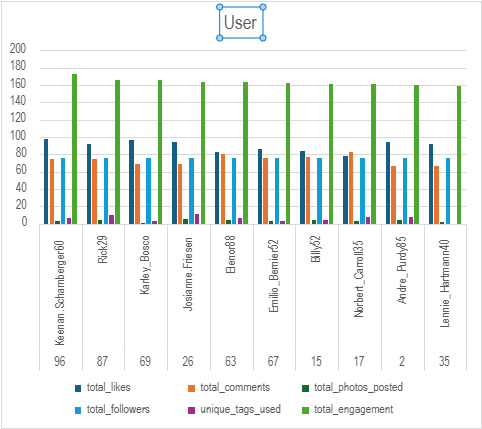
ORDER BY total\_engagement DESC, total\_followers DESC, total\_photos\_posted DESC

LIMIT 10;

#### **Approach to Reward or Incentivize Loyal or Valuable Users**

* **Exclusive Features or Badges**: Award users with high engagement or contribution (e.g., top 10 list) exclusive badges or profile recognition to highlight their contribution.
* **Increased Exposure**: Promote their posts or profiles in feeds or featured sections to give them more visibility within the community.
* **Early Access or VIP Privileges**: Provide these users with early access to new features, updates, or premium content to make them feel valued.
* **Discounts or Gifts**: If applicable, offer discounts or free merchandise as incentives for their loyalty or activity.
* **Personalized Feedback**: Offer recognition in the form of personalized messages or shout-outs for their creative and engaging contributions.
* **Access to Analytics**: For users with high influence (e.g., followers, likes), provide insights into their performance and engagement levels on the platform, rewarding them with deeper engagement tools.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| user\_id | username | total\_likes | total\_comments | total\_photos\_posted | total\_followers | unique\_tags\_used | total\_engagement |
| 96 | Keenan.Schamberger60 | 98 | 75 | 3 | 76 | 6 | 173 |
| 87 | Rick29 | 92 | 74 | 4 | 76 | 10 | 166 |
| 69 | Karley\_Bosco | 97 | 69 | 1 | 76 | 3 | 166 |
| 26 | Josianne.Friesen | 94 | 69 | 5 | 76 | 11 | 163 |
| 63 | Elenor88 | 83 | 80 | 4 | 76 | 6 | 163 |
| 67 | Emilio\_Bernier52 | 86 | 76 | 3 | 76 | 3 | 162 |
| 15 | Billy52 | 84 | 77 | 4 | 76 | 4 | 161 |
| 17 | Norbert\_Carroll35 | 78 | 83 | 3 | 76 | 7 | 161 |
| 2 | Andre\_Purdy85 | 94 | 66 | 4 | 76 | 7 | 160 |
| 35 | Lennie\_Hartmann40 | 92 | 67 | 2 | 76 | 0 | 159 |



1. For inactive users, what strategies would you recommend to re-engage them and encourage them to start posting or engaging again?

=> Activity-Based Rewards: Introduce a system where users can earn badges for participating more actively. For instance, after sharing a certain number of photos or posts, they could receive a Bronze, Silver, or Gold badge, which will be shown on their profile. This visual recognition can motivate users to return and stay active on the platform.

Exclusive Access with Free Subscriptions: Provide users with a 30-day complimentary subscription. This will allow them to explore premium features and could spark their interest to re-engage with the platform.

Tailored Reminders and Notifications: Send customized notifications to remind users of popular posts, updates on their content (like comments or new followers), or upcoming events they might enjoy. By drawing attention to the social interactions and content that matter most to them, we can encourage users to revisit the platform and reconnect with their network.

1. Which hashtags or content topics have the highest engagement rates? How can this information guide content strategy and ad campaigns?

=> SELECT t.tag\_name,

COUNT(DISTINCT p.id) AS photo\_count,

COUNT(DISTINCT l.user\_id) AS likes\_count,

COUNT(DISTINCT c.id) AS comments\_count,

(COUNT(DISTINCT l.user\_id) + COUNT(DISTINCT c.id)) AS engagement\_score

FROM tags t

JOIN photo\_tags pt ON t.id = pt.tag\_id

JOIN photos p ON pt.photo\_id = p.id

LEFT JOIN likes l ON p.id = l.photo\_id

LEFT JOIN comments c ON p.id = c.photo\_id

GROUP BY t.tag\_name

ORDER BY engagement\_score DESC

LIMIT 10;

### **Insights and Approach for Identifying Hashtags/Content Topics with Highest Engagement**

#### **Key Metrics to Evaluate Hashtag Engagement:**

* **Photo Count**: Number of photos using the hashtag, indicating its popularity.
* **Likes Count**: Total likes on photos tagged with the hashtag, reflecting user approval.
* **Comments Count**: Total comments, showing deeper interaction.
* **Engagement Score**: Combined measure of likes and comments, representing overall engagement.

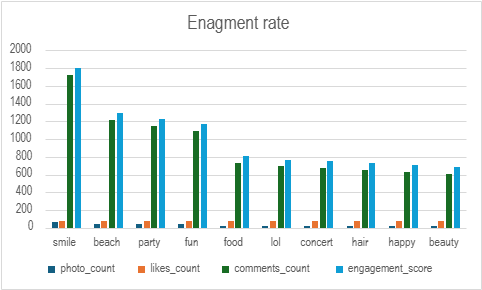
#### **Top Hashtags with Highest Engagement**

* **#smile**:
  + **Photo Count**: 59 photos.
  + **Likes Count**: 77 likes.
  + **Comments Count**: 1,725 comments.
  + **Engagement Score**: 1,802.

#### **Insights:**

* **High Engagement on #smile**: This hashtag shows a very high level of interaction, with a significant number of comments relative to likes, indicating active discussions and strong engagement.
* **Positive and Relatable Content**: The high engagement suggests that emotional, positive, and relatable content (like smiling) resonates well with the audience.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| tag\_name | photo\_count | likes\_count | comments\_count | engagement\_score |
| smile | 59 | 77 | 1725 | 1802 |
| beach | 42 | 77 | 1216 | 1293 |
| party | 39 | 77 | 1151 | 1228 |
| fun | 38 | 77 | 1089 | 1166 |
| food | 24 | 77 | 727 | 804 |
| lol | 24 | 77 | 691 | 768 |
| concert | 24 | 77 | 676 | 753 |
| hair | 23 | 77 | 652 | 729 |
| happy | 22 | 77 | 625 | 702 |
| beauty | 20 | 77 | 604 | 681 |



1. Are there any patterns or trends in user engagement based on demographics (age, location, gender) or posting times? How can these insights inform targeted marketing campaigns?

=> SELECT

HOUR(p.created\_dat) AS post\_hour,

DAYOFWEEK(p.created\_dat) AS post\_day,

COUNT(DISTINCT p.id) AS total\_photos\_posted,

COUNT(DISTINCT l.photo\_id) AS total\_likes\_received,

COUNT(DISTINCT c.id) AS total\_comments\_made

FROM photos p

JOIN likes l ON p.id = l.photo\_id

JOIN comments c ON p.id = c.photo\_id

GROUP BY post\_hour, post\_day;

### **Insights and Approach for Analyzing User Engagement Based on Posting Times**

#### **Key Metrics to Evaluate Engagement by Posting Time:**

* **Post Hour**: The time of day when photos are posted, indicating when users are most active.
* **Post Day**: The day of the week when photos are posted, showing trends in user activity over time.
* **Total Photos Posted**: The number of photos posted during each specific hour and day.
* **Total Likes Received**: The number of likes photos received during each specific hour and day.
* **Total Comments Made**: The number of comments made on photos during each specific hour and day.

#### **Query Results Breakdown (Hypothetical Results)**

* **Post Hour**: Identify the most popular hours for posting photos and receiving engagement (likes and comments).
* **Post Day**: Identify which days of the week see the highest levels of user activity and engagement.

1. Based on follower counts and engagement rates, which users would be ideal candidates for influencer marketing campaigns? How would you approach and collaborate with these influencers?

=> SELECT

u.id AS user\_id,

u.username,

COALESCE(tl.total\_likes, 0) AS Total\_Likes,

COALESCE(tc.total\_comments, 0) AS Total\_Comments,

COALESCE(pp.total\_photos\_posted, 0) AS Total\_Posts,

COALESCE(f.total\_followers, 0) AS Total\_Followers,

ROUND((COALESCE(tl.total\_likes, 0) + COALESCE(tc.total\_comments, 0)) / (COALESCE(pp.total\_photos\_posted, 0)),2) AS Engagement\_Rate

FROM users u

LEFT JOIN (

SELECT u.id, COUNT(DISTINCT l.photo\_id) AS total\_likes

FROM users u

LEFT JOIN likes l ON u.id = l.user\_id

GROUP BY u.id

) tl ON u.id = tl.id

LEFT JOIN (

SELECT u.id, COUNT(DISTINCT c.photo\_id) AS total\_comments

FROM users u

LEFT JOIN comments c ON u.id = c.user\_id

GROUP BY u.id

) tc ON u.id = tc.id

LEFT JOIN (

SELECT user\_id, COUNT(id) AS total\_photos\_posted

FROM photos

GROUP BY user\_id

) pp ON u.id = pp.user\_id

LEFT JOIN (

SELECT followee\_id AS user\_id, COUNT(follower\_id) AS total\_followers

FROM follows

GROUP BY followee\_id

) f ON u.id = f.user\_id

GROUP BY u.id

HAVING Total\_Posts > 0

ORDER BY Engagement\_Rate DESC, Total\_Followers DESC, Total\_Posts DESC

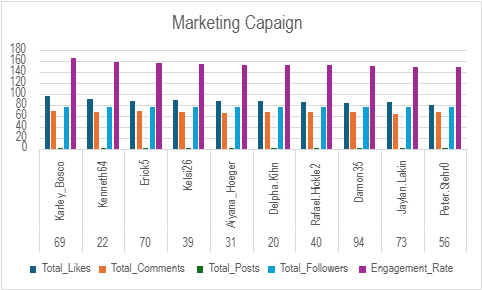
LIMIT 10;

### **Insights and Approach for Identifying Ideal Influencers for Marketing Campaigns**

#### **Key Metrics for Influencer Evaluation:**

* **Total Likes**: The total number of likes received on posts, indicating user approval and engagement.
* **Total Comments**: Reflects deeper engagement and interaction with content.
* **Total Posts**: The number of posts shared, showing content consistency.
* **Total Followers**: The size of the influencer's audience, indicating their reach.
* **Engagement Rate**: Calculated by the sum of likes and comments divided by the total number of posts, reflecting the level of interaction per post.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| user\_id | username | Total\_Likes | Total\_Comments | Total\_Posts | Total\_Followers | Engagement\_Rate |
| 69 | Karley\_Bosco | 97 | 69 | 1 | 76 | 166 |
| 22 | Kenneth64 | 91 | 67 | 1 | 76 | 158 |
| 70 | Erick5 | 88 | 69 | 1 | 76 | 157 |
| 39 | Kelsi26 | 89 | 67 | 1 | 76 | 156 |
| 31 | Aiyana\_Hoeger | 88 | 66 | 1 | 76 | 154 |
| 20 | Delpha.Kihn | 87 | 67 | 1 | 76 | 154 |
| 40 | Rafael.Hickle2 | 85 | 68 | 1 | 76 | 153 |
| 94 | Damon35 | 84 | 68 | 1 | 76 | 152 |
| 73 | Jaylan.Lakin | 86 | 63 | 1 | 76 | 149 |
| 56 | Peter.Stehr0 | 81 | 68 | 1 | 76 | 149 |



1. Based on user behavior and engagement data, how would you segment the user base for targeted marketing campaigns or personalized recommendations?

=> -- 1)For User Type(OLD/NEW USER)

SELECT

id AS user\_id,

username,

created\_at AS signup\_date,

CASE

WHEN DATEDIFF('2017-05-04', created\_at) <= 60 THEN 'New User'

ELSE 'Old User'

END AS user\_type

FROM

users;

### **Insights and Approach for Segmenting User Base for Targeted Marketing Campaigns**

#### **1. Segmentation by User Type (New vs. Old Users)**

* **New User**: Identified as users who signed up within the last 60 days.
* **Old User**: Users who signed up more than 60 days ago.

#### **Key Insights:**

* **New Users**: Engage in onboarding content, tutorials, and promotions that help them get familiar with the platform.
* **Old Users**: They are already familiar with the platform and may respond better to loyalty programs, advanced features, or exclusive offers.

-- 2) for user\_category(HIGHLY/MODERATE/LESS ACTIVE)

SELECT

u.id AS user\_id,

u.username,

p.total\_post,

COALESCE(l.total\_likes, 0) + COALESCE(c.total\_comments, 0) AS total\_engagement,

CASE

WHEN COALESCE(l.total\_likes, 0) + COALESCE(c.total\_comments, 0) > 150 THEN 'Highly Active'

WHEN COALESCE(l.total\_likes, 0) + COALESCE(c.total\_comments, 0) BETWEEN 50 AND 150 THEN 'Moderately Active'

ELSE 'Less Active'

END AS user\_category

FROM users u

LEFT JOIN (

SELECT user\_id, COUNT(\*) AS total\_likes

FROM ig\_clone.likes

GROUP BY user\_id

) l ON u.id = l.user\_id

LEFT JOIN (

SELECT user\_id, COUNT(\*) AS total\_comments

FROM comments

GROUP BY user\_id

) c ON u.id = c.user\_id

LEFT JOIN (

SELECT user\_id, count(id) AS total\_post

FROM photos

GROUP BY user\_id

) p ON u.id = p.user\_id

group by u.id

having p.total\_post >0

ORDER BY user\_id DESC;

#### **Segmentation by User Activity (Highly Active, Moderately Active, Less Active)**

* **Highly Active**: Users with a high number of likes and comments (150+ total engagements).
* **Moderately Active**: Users with moderate engagement (50-150 total engagements).
* **Less Active**: Users with low engagement (less than 50 total engagements).

#### **Key Insights:**

* **Highly Active Users**: These users have high engagement and content interaction. They are likely content creators or super-users.
* **Moderately Active Users**: They engage regularly but are not as consistent. They might be more passive users or lurkers.
* **Less Active Users**: These users have low engagement and fewer posts. They may need encouragement or incentives to interact more.

1. If data on ad campaigns (impressions, clicks, conversions) is available, how would you measure their effectiveness and optimize future campaigns?

=> To measure how well our ad campaigns are performing, we could use this approach:

SELECT   
campaign\_id,   
impressions,   
clicks,   
conversions,   
(clicks \* 1.0 / impressions) AS click\_through\_rate,   
(conversions \* 1.0 / clicks) AS conversion\_rate  
FROM AdCampaigns  
WHERE (conversions \* 1.0 / clicks) < targetConversionRate;

Here’s what we’re looking at:

1. Click-Through Rate (CTR): This metric shows how often people clicked on the ad after seeing it. To calculate it, we divide the total number of clicks by the total number of impressions. If the CTR is high, it usually means the ad is grabbing people’s attention.
2. Conversion Rate: This tells us how many of those clicks actually led to the desired action—like a purchase or sign-up. We figure this out by dividing the number of conversions by the number of clicks. This helps us gauge whether the ad is actually driving meaningful actions.

Looking at the impression data gives us another layer of insight. We can see which types of ads users are engaging with the most, and from there, we can figure out what’s working and what’s not. This helps us pinpoint areas that need improvement.

1. How can you use user activity data to identify potential brand ambassadors or advocates who could help promote Instagram's initiatives or events?

=> To identify potential brand ambassadors or advocates from user activity data, we can use several metrics to pinpoint engaged and influential users:

User Engagement (Likes and Comments): We can identify users who receive a higher number of likes and comments on their posts. This typically reflects the quality and appeal of the content they share.

sql

Copy code

SELECT   
u.id AS user\_id,   
COALESCE(SUM(l.user\_id IS NOT NULL), 0) AS total\_likes,   
COALESCE(SUM(c.user\_id IS NOT NULL), 0) AS total\_comments  
FROM users u   
LEFT JOIN likes l ON u.id = l.user\_id   
LEFT JOIN comments c ON u.id = c.user\_id   
GROUP BY u.id;

Followers Count: Users with a large following are valuable because they have a wider reach and can influence a larger audience.

sql

Copy code

SELECT   
followee\_id AS user\_id,   
COUNT(follower\_id) AS total\_followers   
FROM follows   
GROUP BY followee\_id;

Content Frequency: By analyzing how often users post—whether it’s weekly or monthly—we can gauge their level of interest and commitment to the platform.

sql

Copy code

SELECT   
p.user\_id,   
EXTRACT(MONTH FROM p.created\_dat) AS post\_month,   
COUNT(p.id) AS total\_posts\_per\_month   
FROM photos p   
 GROUP BY p.user\_id, post\_month;

Brand Advocacy: We can also identify users who actively promote or support the platform’s initiatives. Their consistent enthusiasm and engagement can be a strong indicator that they genuinely care about the brand and are likely to advocate for it.

1. How would you approach this problem, if the objective and subjective questions weren't given?

=> If the question had not provided a specific approach, I would consider tackling this problem with the following strategies:

1. **Analyze User Activity by Time and Age Group**:
   1. Start by tracking user activity patterns—focusing on likes, comments, and the times they post. Group this data by age brackets to identify when different demographics are most engaged.
   2. This will help us schedule marketing content and ad campaigns at the times when users are most active, maximizing visibility and engagement for each age group.
2. **Segment Users by Interests and Behaviors**:
   1. We could categorize users based on their interests and behaviors, such as the type of content they interact with most (fashion, fitness, tech, etc.) or how often they engage with posts.
   2. This segmentation allows for more personalized marketing campaigns tailored to each group’s preferences, resulting in higher relevance and better engagement rates.
3. **Leverage Influencers and High-Engagement Users**:
   1. Look for users with large followings and high levels of engagement (lots of likes, comments, and shares). These influencers have a broader reach and can help enhance credibility and visibility for your brand.
   2. Partnering with these users for collaborations, sponsored content, or even ambassador roles can amplify your messaging and connect you to new audiences.
4. **Optimize Content Strategy Based on Engagement**:
   1. Study which types of content (photos, videos, stories) are generating the most engagement and impressions. This analysis helps us understand what resonates with the audience and allows us to focus resources on creating more of the content that performs well.
   2. By consistently prioritizing high-performing content, we can ensure that our campaigns are more effective and drive higher user interaction.
5. **Evaluate Past Campaigns and Refine Future Strategies**:
   1. Measure past campaign performance through metrics like impressions, clicks, and conversions. This data gives us valuable insight into what worked and what didn’t.
   2. We can use this historical data to optimize future campaigns, allocate marketing budgets more efficiently, and focus on strategies that have proven successful.
6. **Re-engage Inactive Users**:
   1. Segment users into active and inactive groups. For inactive users, we could send personalized messages or offers, such as exclusive promotions or memberships, to bring them back onto the platform.
   2. For active users, we could offer loyalty rewards or new features to keep them engaged and deepen their connection with the platform.
7. **Competitive Analysis**:
   1. Compare user behavior and engagement on our platform with that of competitors. This will help identify gaps in the market and uncover opportunities for differentiation.
   2. Based on this analysis, we can develop strategies that capitalize on market trends, positioning our brand to stand out in a competitive landscape.
8. **Targeted Advertising**:
   1. Improve ad performance by ensuring that ads are reaching users who are most likely to be interested in our products or services. This can be achieved by using detailed user data, such as their previous interactions with content, demographics, and purchase behavior.
   2. By delivering ads to the right audience at the right time, we increase the chances of higher engagement and conversions.
9. Assuming there's a "User\_Interactions" table tracking user engagements, how can you update the "Engagement\_Type" column to change all instances of "Like" to "Heart" to align with Instagram's terminology?

=> -- Disable safe updates temporarily

SET SQL\_SAFE\_UPDATES = 0;

-- Perform the update operation

UPDATE User\_Interactions

SET Engagement\_Type = 'Heart'

WHERE Engagement\_Type = 'Likes'

AND id IS NOT NULL;

-- Re-enable safe updates after the operation

SET SQL\_SAFE\_UPDATES = 1;

### **Perform the Update Operation:**

* The UPDATE statement will change the Engagement\_Type from "Likes" to "Heart".
* The WHERE clause ensures the change only applies to rows where Engagement\_Type = 'Likes' and the id is not null, ensuring you don't inadvertently update unwanted records.

### **Important Considerations:**

* **Data Integrity**: This query ensures only rows with the Engagement\_Type = 'Likes' are updated, preserving other types of engagement (e.g., comments, shares) if they exist.
* **Performance**: If the table has a large number of rows, ensure that there are indexes on the Engagement\_Type and id columns to optimize performance.

This query should effectively align the terminology in your database with Instagram's "Heart" terminology.