**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?

**Finding Duplicate:**

SELECT username, COUNT(\*) AS count

FROM users

GROUP BY username

HAVING COUNT(\*) > 1;

SELECT user\_id, image\_url, COUNT(\*) AS count

FROM photos

GROUP BY user\_id, image\_url

HAVING COUNT(\*) > 1;

SELECT image\_url, COUNT(\*) AS count

FROM photos

GROUP BY image\_url

HAVING COUNT(\*) > 1;

SELECT user\_id, photo\_id, comment\_text, COUNT(\*) AS count

FROM comments

GROUP BY user\_id, photo\_id, comment\_text

HAVING COUNT(\*) > 1;

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

FROM likes

GROUP BY user\_id, photo\_id

HAVING COUNT(\*) > 1;

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

FROM follows

GROUP BY follower\_id, followee\_id

HAVING COUNT(\*) > 1;

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

FROM photo\_tags

GROUP BY photo\_id, tag\_id

HAVING COUNT(\*) > 1;

**Approach:**

**Detecting Duplicate Records Across Key Tables**

* **Used GROUP BY + HAVING COUNT(\*) > 1** to check for duplicate records.
* Each query is designed to focus on **logical uniqueness** based on the table’s purpose and expected data constraints.
* Duplicate entries can distort analytics (e.g., inflated like/comment counts, repeated followers).
* Using GROUP BY + HAVING COUNT(\*) > 1 provides visibility into how often a combination appears, making it easier to debug or clean the data.
* Chosen GROUP BY + HAVING COUNT(\*) > 1 over DISTINCT because it retains duplicates and gives a count, which DISTINCT does not show.
* Applied to the following tables:
  + users, photos, comments, likes, follows, photo\_tags

**Finding Null:**

**-- Users table**

SELECT \* FROM users

WHERE username IS NULL OR created\_at IS NULL;

**-- Photos table**

SELECT \* FROM photos

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

**-- Comments table**

SELECT \* FROM comments

WHERE comment\_text IS NULL OR user\_id IS NULL OR photo\_id IS NULL OR created\_at IS NULL;

**-- Likes table**

SELECT \* FROM likes

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

**-- Follows table**

SELECT \* FROM follows

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

**-- Tags table**

SELECT \* FROM tags

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

**-- Photo\_Tags table**

SELECT \* FROM photo\_tags

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

**Approach:**

**Explanation: Null Value Checks**

* Nulls in relational fields (e.g., user\_id, photo\_id) can break foreign key relationships, cause join failures, or lead to missing data in reports.
* Not all columns are checked — only those that:
  + Are not strictly enforced by NOT NULL or PRIMARY KEY
  + Are not foreign keys with enforced referential integrity
* This approach improves data trustworthiness.
* This avoids unnecessary checks and focuses on fields where missing data could affect functionality.

**TRIM WHITESPACES:**

**-- User table**

UPDATE users

SET username = TRIM(username)

WHERE username != TRIM(username);

**-- photos Table**

UPDATE photos

SET image\_url = TRIM(image\_url)

WHERE image\_url != TRIM(image\_url);

**-- Comments Table**

UPDATE comments

SET comment\_text = TRIM(comment\_text)

WHERE comment\_text != TRIM(comment\_text);

**-- Tags Table**

UPDATE tags

SET tag\_name = TRIM(tag\_name)

WHERE tag\_name != TRIM(tag\_name);

**Approach:**

* Removed the White Spaces using the TRIM() and used WHERE the there are white space are TRIM() not applied.

**Updating Column name for Consistency:**

ALTER TABLE photos

CHANGE COLUMN created\_dat

created\_at TIMESTAMP DEFAULT NOW();

**Approach:**

* Since created\_at in all the table is same, for consistent approach modified the column name to created\_at from created\_dat in photos table.

**Standardizing text format:**

**-- User Table**

UPDATE users

SET username = LOWER(username);

**-- Tags Table**

UPDATE tags

SET tag\_name = LOWER(tag\_name);

**-- Comments Table**

UPDATE comments

SET comment\_text = LOWER(comment\_text);

**Approach:**

* Modified the text data in the table is to standard format, using the UPDATE command and set the text to lower case.

**Verifying Foreign Key Consistency:**

**-- Comments Table**

SELECT \* FROM comments

WHERE user\_id NOT IN (SELECT id FROM users)

OR photo\_id NOT IN (SELECT id FROM photos);

**-- Photos Table**

SELECT \* FROM photos

WHERE user\_id NOT IN (SELECT id FROM users);

**-- Likes Table**

SELECT \* FROM likes

WHERE user\_id NOT IN (SELECT id FROM users)

OR photo\_id NOT IN (SELECT id FROM photos);

**-- Follows Table**

SELECT \* FROM follows

WHERE follower\_id NOT IN (SELECT id FROM users)

OR followee\_id NOT IN (SELECT id FROM users);

**-- Photo\_tags Table**

SELECT \* FROM photo\_tags

WHERE photo\_id NOT IN (SELECT id FROM photos)

OR tag\_id NOT IN (SELECT id FROM tags);

**Approach:**

* Verified the foreign key is consistent to the primary key using “where” clause and Subqueries in the above code.

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

**Code:**

select

U.id,

U.username,

count(distinct P.id) as Number\_of\_posts,

count(distinct L.photo\_id) as Number\_of\_likes,

count(distinct C.id) as Number\_of\_Comments

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

group by U.id,U.username;

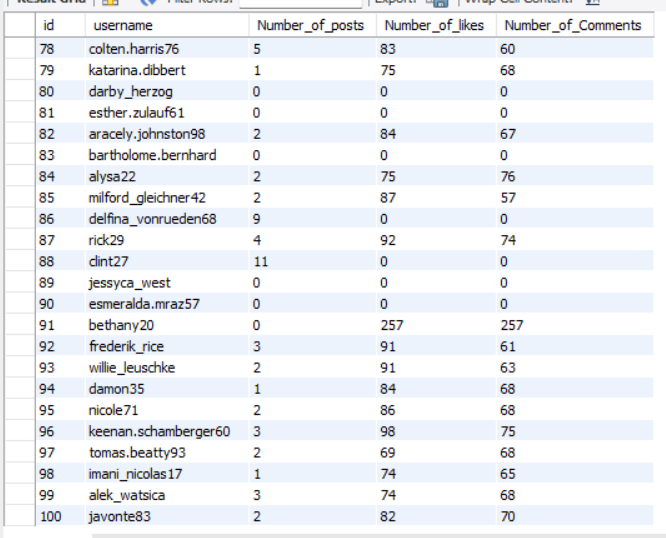
**Approach:**

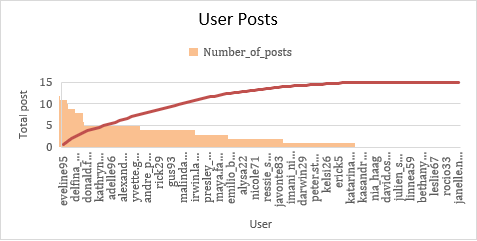
1. **User Engagement Summary**
   * Joined users, photos, likes, and comments to measure user activity across:
     + Posts created
     + Photos liked
     + Comments made
2. **LEFT JOINs**
   * Ensures all users are included, even with no activity.
3. **COUNT(DISTINCT …)**
   * Avoids double-counting and ensures accuracy.

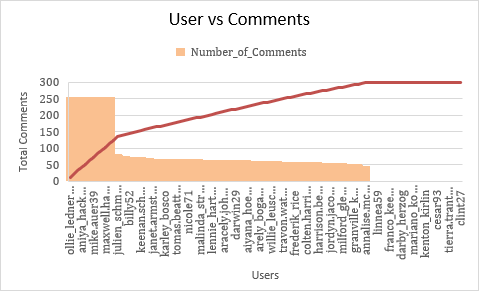
**Explanation:**

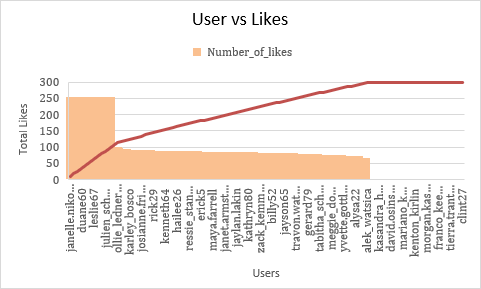
This query gives a per-user summary of:

* How many posts they made.
* How many photos they liked.
* How many comments they wrote.

****







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

**Code:**

SELECT AVG(tag\_count) AS avg\_tags\_per\_photo

FROM (

SELECT P.id, COUNT(PT.tag\_id) AS tag\_count

FROM photos P

LEFT JOIN photo\_tags PT ON P.id = PT.photo\_id

GROUP BY P.id

) AS tag\_counts;

**Approach:**

1. **Average tags per post**

* Initially did left join photos table with photo\_tags table to ensure all the post from photos table are listed.
* Then, using aggregate function “count” calculated total number of photo\_tags for each photo.
* Then considered the above result as a table calculated the average tags per post using aggregate function “AVG”.

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

**Code:**

with Engagement\_rank as (

select

id,

username,

coalesce(round(((Total\_Comments+Total\_Likes)/Total\_Post),2),0) as Engagement\_rate

from (

select

U.id,

U.username,

count(photo\_id) Total\_Post,

sum(number\_of\_Comments) as Total\_Comments,

sum(number\_of\_Likes) as Total\_Likes

from users U

left join (

select

P.user\_id,

P.id as Photo\_id,

count(distinct C.id) as number\_of\_Comments,

count(distinct L.user\_id) as number\_of\_Likes

from photos P

left join comments C on P.id = C.photo\_id

left join likes L on P.id = L.photo\_id

group by P.id, user\_id

) X on U.id = X.user\_id

group by U.id

) RN

)

select

\*,

dense\_rank() over (order by Engagement\_rate desc) as `Rank`

from Engagement\_rank;

**Engagement Rate & Ranking – Approach:**

1. **Photo-Level Metrics via Subquery**  
   • Left joined photos with comments and likes to count total comments and likes per photo.  
   • Grouped by photo\_id and user\_id to get per-photo engagement.
2. **User-Level Aggregation**  
   • Joined users with the subquery to sum up all likes, comments, and post count per user.  
   • Grouped by user to get total metrics.
3. **Engagement Rate Calculation (CTE)**  
   • Calculated engagement rate as:  
   (Total\_Comments + Total\_Likes) / Total\_Post  
   • Used COALESCE to avoid nulls or division by zero.
4. **User Ranking**  
   • Applied DENSE\_RANK() to rank users based on highest engagement rate.

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

**Code:**

select

U.id,

U.username,

count(distinct F1.follower\_id) as Total\_follower,

count(distinct F.followee\_id) as Total\_followee

from users U

left join follows F on U.id = F.follower\_id

left join follows F1 on U.id = F1.followee\_id

group by U.id,U.username

order by Total\_followee desc,Total\_follower desc;

**Follower vs. Followee Count – Approach:**

1. **Table Joins for Follower/Followee Mapping**  
   • Joined users with follows (aliased as F) to get whom the user follows (followee\_id).  
   • Joined users again with follows (aliased as F1) to get who follows the user (follower\_id).
2. **Aggregation**  
   • Used COUNT(DISTINCT) to calculate:  
   – Total\_follower: how many unique users follow each user.  
   – Total\_followee: how many unique users each user is following.
3. **Result Structuring**  
   • Grouped by user to get counts per user.  
   • Ordered the results by Total\_followee and Total\_follower in descending order to list the most socially connected users first.
4. Calculate the average engagement rate (likes, comments) per post for each user.

**Code:**

with Engagements as

(

select

P.user\_id,

P.id as Photo\_id,

(count(distinct C.id) + count(distinct L.user\_id)) as Total\_Engagement

from photos P

left join comments C on P.id = C.photo\_id

left join likes L on P.id = L.photo\_id

group by P.id, user\_id

)

select

E.user\_id,

U.username,

round(avg(Total\_Engagement),2) as AVG\_Engagement\_per\_post

from Engagements E

join users U on E.user\_id = U.id

group by user\_id

order by AVG\_Enagement\_per\_post desc;

**Average Engagement per Post – Approach:**

1. **Calculate Engagements per Photo**  
   • Used a **CTE (Engagements)** to join the photos table with comments and likes tables.  
   • For each photo, computed Total\_Engagement as the sum of:  
   – Distinct comments (C.id)  
   – Distinct likes (L.user\_id)  
   • Grouped by photo\_id and user\_id to get engagement per post.
2. **Aggregate at User Level**  
   • Joined CTE with users table to get usernames.  
   • Calculated the **average engagement per post per user** using AVG(Total\_Engagement).
3. **Result Presentation**  
   • Grouped by user\_id to compute per-user averages.  
   • Sorted the results by AVG\_Engagement\_per\_post in descending order to highlight the most engaging content creators.

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

Code:

select

id,

username

from users

where id not in (

select user\_id from likes);

**Identifying Users Who Haven’t Liked Any Photos – Approach:**

1. **Subquery to Collect Active Likers**  
   • Selected all user\_ids from the likes table — these represent users who have liked at least one photo.
2. **Filter Non-Likers**  
   • Queried the users table to select id and username.  
   • Applied a **WHERE id NOT IN (...)** condition using the subquery to exclude users present in the likes table.
3. **Final Result**  
   • Returned a list of users who have **never liked** any photo on the platform.  
   • Helps in identifying **passive or inactive users** for potential re-engagement strategies.
4. How can you leverage user-generated content (posts, hashtags, photo tags) to create more personalized and engaging ad campaigns?

**Code1:**

with Popular\_tags as (

select T.id,T.tag\_name, count(photo\_id) as total\_photos

from photo\_tags PT

join tags T on PT.tag\_id = T.id

group by tag\_id

having total\_photos >= 20

)

select distinct U.id, U.username

from users U

join photos P on U.id = P.user\_id

join photo\_tags PT on P.id = PT.photo\_id

where PT.tag\_id in (select id from popular\_tags);

**Identifying Users Posting with Popular Tags – Approach**

1. **CTE: Popular\_tags**  
   • Joined the photo\_tags table with the tags table to get tag names.  
   • Counted how many photos were tagged with each tag (count(photo\_id)).  
   • Applied a HAVING clause to filter for tags used in **at least 20 photos**, marking them as popular.
2. **Main Query**  
   • Started from the users table and joined:  
   – photos to connect users with their posts.  
   – photo\_tags to associate those posts with their tags.  
   • Filtered using a WHERE clause to retain only those posts tagged with **popular tags** (using IN with the CTE result).
3. **Final Output**  
   • Selected **distinct user IDs and usernames** of users who posted content with popular tags.  
   • Useful for finding active users aligned with trending or widely followed topics — potential candidates for targeted promotions or community features.

**Code2:**

with Popular\_tags as (

select T.id,T.tag\_name, count(photo\_id) as total\_photos

from photo\_tags PT

join tags T on PT.tag\_id = T.id

group by tag\_id,T.tag\_name

having total\_photos >= 20

)

select distinct C.user\_id

from comments C

join photos P on C.photo\_id = P.id

join photo\_tags PT on P.id = PT.photo\_id

where PT.tag\_id in (select id from popular\_tags)

union

select distinct L.user\_id

from Likes L

join photos P on L.photo\_id = P.id

join photo\_tags PT on P.id = PT.photo\_id

where PT.tag\_id in (select id from popular\_tags)

order by user\_id;

**Approach: Identifying Users Engaging with Popular Tags:**

1. **CTE: Popular\_tags**  
   • Joined photo\_tags with tags to retrieve tag names.  
   • Grouped by tag\_id and tag\_name to count the number of **photos** each tag appears in.  
   • Applied a HAVING clause to filter only **popular tags** (used in 20 or more photos).
2. **Main Query (Union of Commenters and Likers)**

• **First SELECT**:

I) Joined comments with photos and photo\_tags to link each comment to the

tags used in the associated photo.  
II) Filtered using IN (SELECT id FROM popular\_tags) to find users who

**commented** on photos with popular tags.

• Second SELECT (after UNION):  
 I) Followed a similar pattern but with the likes table instead of

comments.  
 II) Retrieved users who liked photos tagged with popular tags.

1. **UNION and Ordering**  
   • Used UNION to combine the two user groups (commenters and likers), removing duplicates automatically.  
   • Final result gives a **distinct list of users** who have either commented on or liked photos with popular tags, **sorted by user\_id**.

1. 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?

**Code:**

select

T.id,

T.tag\_name,

count(distinct P.id) as Posts\_count,

count(distinct C.id) as Total\_Comments,

count(distinct L.user\_id,L.photo\_id) as Total\_likes,

round((count(distinct C.id) + count(distinc L.user\_id,L.photo\_id))/count(distinct P.id),2) as

AVG\_tag\_engagement

FROM likes L

join photos P on L.photo\_id = P.id

join comments C on P.id = C.photo\_id

JOIN photo\_tags PT ON P.id = PT.photo\_id

JOIN tags T ON PT.tag\_id = T.id

group by T.id,T.tag\_name

order by AVG\_tag\_engagement desc;

**Tag-Wise Engagement Analysis:**

• Started by **joining the likes table with photos** to connect each like to the

corresponding photo.  
• Then, **joined comments table with photos** to count total comments made on each

photo.  
• Followed by **joining photo\_tags with photos** to map each photo to its associated

tags.  
• Finally, **joined tags table** to get the tag\_name and tag\_id for each tag.

• Used COUNT(DISTINCT P.id) to calculate total number of unique posts under each

tag.  
• Used COUNT(DISTINCT C.id) to get the number of unique comments on those

posts.  
• Attempted to use COUNT(DISTINCT L.user\_id, L.photo\_id) to calculate total likes,

to properly count each like per photo-user combination.

• Calculated **average engagement per post** for each tag.

• Grouped the result by T.id and T.tag\_name to analyze metrics per tag.  
• Ordered the output by AVG\_tag\_engagement in descending order to list the **most**

**engaging tags first**.

1. Calculate the total number of likes, comments, and photo tags for each user.

**Code:**

with Userwise\_engagement as (

select

P.user\_id,

P.id as Photo\_id,

count(distinct C.id) as number\_of\_Comments,

count(distinct L.user\_id) as number\_of\_Likes

from photos P

left join comments C on P.id = C.photo\_id

left join likes L on P.id = L.photo\_id

group by P.id, user\_id

),Userwise\_posts\_engagement as (

select

user\_id,

count(photo\_id) as Total\_Post,

sum(number\_of\_Comments) as

Total\_Comments\_received,

sum(number\_of\_Likes) as Total\_Likes\_received

from Userwise\_engagement

group by user\_id

),user\_commented as (

select

user\_id,

count(id) as User\_total\_commented

from comments

group by user\_id

),user\_likes as (

select

user\_id,

count(photo\_id) as User\_total\_liked

from likes

group by user\_id

)

select

U.id,

U.username,

coalesce(Total\_Post,0) as Total\_Post,

coalesce(Total\_Comments\_received,0) as Total\_Comments\_received,

coalesce(Total\_Likes\_received,0) as Total\_Likes\_received,

coalesce(User\_total\_commented,0) as User\_total\_commented,

coalesce(User\_total\_liked,0) as User\_total\_liked

from users U

left join Userwise\_posts\_engagement UPE on U.id = UPE.user\_id

left join user\_commented UC on U.id = UC.user\_id

left join user\_likes UL on U.id = UL.user\_id;

**Approach:**

**1. Create user-wise photo engagement summary (CTE: Userwise\_engagement)**

* Joined:
  + photos with comments to count how many comments each photo received.
  + photos with likes to count how many likes each photo received.
* Grouped by photo\_id and user\_id to get per-photo engagement for each user.

**2. Aggregate engagement received by users (CTE: Userwise\_posts\_engagement)**

* Summed up:
  + Number of photos posted (Total\_Post)
  + Total comments received across all user photos
  + Total likes received across all user photos.

**3. Count comments made by each user (CTE: user\_commented)**

* Grouped the comments table by user\_id to find out how many comments each user has made on other photos.

**4. Count likes given by each user (CTE: user\_likes)**

* Grouped the likes table by user\_id to find out how many times each user liked a photo.

**5. Final Output (Main SELECT)**

* Joined the users table with all CTEs:
  + Userwise\_posts\_engagement → to include total posts, likes received, comments received.
  + user\_commented → to include number of comments made by the user.
  + user\_likes → to include number of likes made by the user.
* Used LEFT JOIN to ensure **even inactive users** (who never liked/commented/posted) are included.
* Used COALESCE() to **replace NULL with 0** for users who didn’t perform any activity.

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

**Code:**

select

engagement\_month,

id,

username,

Total\_Engagement,

dense\_rank() over ( partition by engagement\_month order by

Total\_Engagement desc) as RNK

from (

select

U.id,

U.username,

(count(distinct C.id) + count(distinct L.photo\_id)) as Total\_Engagement,

DATE\_FORMAT(COALESCE(l.created\_at, c.created\_at), '%Y-%m')

as engagement\_month

from users U

left join comments C on U.id = C.user\_id

left join likes L on U.id = L.user\_id

group by U.id,U.username,engagement\_month

having Total\_Engagement > 0

)as UserEngagement;

**Approach:**

**1. Join User Activity Tables**

* users table was **left joined** with:
  + comments (on U.id = C.user\_id)
  + likes (on U.id = L.user\_id)
* Purpose: To collect all likes and comments made **by** each user.

**2. Calculate Monthly Engagement**

* Used DATE\_FORMAT() on the created\_at from likes and comments to extract the **engagement\_month** (format: YYYY-MM).
* Used COALESCE() to handle cases where only one of the timestamps exists.
* Counted:
  + DISTINCT C.id for number of comments made.
  + DISTINCT L.photo\_id for number of photos liked.

**3. Group by User and Month**

* Grouped results by:
  + user\_id, username, and engagement\_month to get engagement per user per month.

**4. Filter Active Users**

* Used HAVING Total\_Engagement > 0 to exclude users with no monthly activity.

**5. Assign Monthly Rankings**

* In the outer query:
  + Applied DENSE\_RANK() using PARTITION BY engagement\_month to rank users **within each month** based on Total\_Engagement (highest first).

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.

**Code:**

with Highest\_avg\_likes as (

select

T.id,

T.tag\_name,

round(avg(likes\_count),2) avg\_likes,

rank() over (order by round(avg(likes\_count),2) desc) as RNK

from tags T

join photo\_tags PT on T.id = PT.tag\_id

join photos P on P.id = PT.photo\_id

join (select photo\_id,count(user\_id) as likes\_count

from likes group by photo\_id

) as LikesCount on P.id = LikesCount.likes\_count

group by T.id,T.tag\_name

)

select

id,

tag\_name,

avg\_likes

from Highest\_avg\_likes

where RNK = 1;

**Approach:**

**1. Calculate Likes per Post**

* Created a subquery Likes Count that:
  + Groups the likes table by photo\_id
  + Counts how many users liked each photo → likes\_count

**2. Join Tags with Photo Data**

* tags joined with:
  + photo\_tags (to get which tag is used on which photo)
  + photos (to associate photo IDs)
  + Likes Count (to get like counts per photo)

**3. Calculate Average Likes per Tag**

* For each tag:
  + Calculated AVG(likes\_count) rounded to 2 decimals
  + Used RANK() to rank tags based on their average likes in descending order

**4. Filter for Top Tag**

* In the final SELECT, filtered to show only the tag(s) with **rank = 1**, i.e., highest average likes across posts.

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

**Code:**

select

F2.follower\_id AS user\_id,

F2.followee\_id AS followed\_person\_id,

F1.created\_at AS followed\_first\_at,

F2.created\_at AS followed\_second\_at

FROM

follows AS F1 -- Represents the initial follow (Person B follows Person A)

INNER JOIN

follows AS F2 ON F1.follower\_id = F2.followee\_id -- B follows A, and A follows B

AND F1.followee\_id = F2.follower\_id

WHERE

F2.created\_at > F1.created\_at --User A followed Person B AFTER Person B followed User A

ORDER BY

F2.created\_at ASC;

**Approach:**

**1. Self-Join on follows Table**

* F1 represents the **first follow** (e.g., B → A).
* F2 represents the **second follow** (e.g., A → B).
* Joined where:
  + F1.follower\_id = F2.followee\_id → B followed A
  + F1.followee\_id = F2.follower\_id → A followed B

**2. Filter on Follow Order**

* Used F2.created\_at > F1.created\_at to ensure A followed B **after** B already followed A.

**3. Select Relevant Info**

* F2.follower\_id = User A (the person who followed second)
* F2.followee\_id = User B (the person who followed first)
* F1.created\_at = First follow timestamp (B → A)
* F2.created\_at = Second follow timestamp (A → B)

**4. Ordering**

* Sorted by F2.created\_at ASC to see earliest mutual connections first.

**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?

**Code:**

with User\_age as (

select

U.id,

timestampdiff(day,U.created\_at,now()) as user\_age\_days

from users U

) ,

Engagement\_score as (

select

distinct U.id,

U.username,

UA.user\_age\_days,

count(distinct C.id)\*2 as Comments\_score, -- assigning 2points for comment ,1

point for likes , 3 point for posts

count(distinct P.id)\*3 as Posts\_score,

count(distinct L.photo\_id) as Likes\_score

from users U

left join comments C on U.id = C.user\_id

left join likes L on U.id = L.user\_id

left join User\_age UA on U.id = UA.id

left join Photos P on U.id = P.user\_id

group by U.id

),

Ranked\_Users AS (

SELECT

id,

username,

user\_age\_days,

COALESCE(Comments\_score, 0) + COALESCE(Posts\_score, 0) + COALESCE(Likes\_score, 0) AS Total\_engage\_score

FROM Engagement\_score

)

SELECT

Id, username, user\_age\_days, Total\_engage\_score,

DENSE\_RANK() OVER (ORDER BY Total\_engage\_score DESC, user\_age\_days DESC) AS

engagement\_rank

FROM Ranked\_Users

limit 10;

**Approach:**

**1. User Age Calculation (CTE: User\_age)**

* Used TIMESTAMPDIFF(DAY, created\_at, NOW()) to compute **number of days since user joined**.
* Output: Each user’s user\_age\_days.

**2. Engagement Score Assignment (CTE: Engagement\_score)**

* Joined the users table with:
  + comments (for comment activity)
  + likes (for likes given)
  + photos (for posts made)
  + User\_age (for account age).
* Calculated a **weighted engagement score**:
  + Comments\_score = count(distinct comment\_id) × 2
  + Posts\_score = count(distinct photo\_id) × 3
  + Likes\_score = count(distinct liked photo\_id) × 1.

**3. Summing Total Engagement (CTE: Ranked\_Users)**

* Added the three weighted scores using COALESCE to avoid nulls:
  + Total\_engage\_score = Comments\_score + Posts\_score + Likes\_score.

**4. Ranking Logic**

* Used DENSE\_RANK() to assign engagement rank:
  + Ordered by Total\_engage\_score DESC
  + In case of tie, older users (user\_age\_days DESC) get higher rank.

**5. Final Selection**

* Retrieved Top 10 most engaged users by limiting output using LIMIT 10.

**Top loyal users (top 5–10) can be rewarded with:**

* **Badges**: “Top Contributor”, “Engaged User”
* **Priority features**: Early access to new features or beta tools
* **Recognition**: Public leaderboard
* **Monetary or in-kind rewards**: Amazon vouchers, shout-outs
* **Engagement roles**: Allow them to moderate communities or promote content
* We could also create a **view** or a **dashboard** that tracks engagement trends over time.

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

**Code:**

with inactive\_user\_score as (

select

U.id,

U.username,

count(distinct C.id)\*2 as Comments\_score, -- assigning 2points for comment

,1 point for likes , 3 point for posts

count(distinct P.id)\*3 as Posts\_score,

count(distinct L.photo\_id) as Likes\_score

from users U

left join comments C on U.id = C.user\_id

left join likes L on U.id = L.user\_id

left join Photos P on U.id = P.user\_id

group by U.id

),

Ranked\_Inative\_Users AS (

SELECT

id,

username,

COALESCE(Comments\_score, 0) + COALESCE(Posts\_score, 0) + COALESCE(Likes\_score, 0) AS Total\_engage\_score

FROM inactive\_user\_score

)

SELECT

id,

username,

Total\_engage\_score,

DENSE\_RANK() OVER (ORDER BY Total\_engage\_score ASC) AS inactive\_user\_rank

FROM Ranked\_Inative\_Users

limit 20;

**Approach:**

**1. Score Calculation (CTE: inactive\_user\_score)**

* Joined users table with:
  + comments → to capture user comments
  + likes → to capture likes given
  + photos → to capture posts made
* Assigned engagement weights:
  + 2 points per comment
  + 3 points per post
  + 1 point per like
* Aggregated counts using COUNT(DISTINCT ...) for accuracy and deduplication.

**2. Total Engagement Score (CTE: Ranked\_Inative\_Users)**

* Combined all scores using COALESCE() to avoid null issues.

“Total\_engage\_score = Comments\_score + Posts\_score + Likes\_score”

**3. Ranking Inactive Users**

* Used DENSE\_RANK() to order users by lowest to highest engagement.
* Focused on least engaged users:
  + Those with zero or minimal activity
* Limited output to bottom 20 users using LIMIT 20.

**Strategies to re-engage users:**

**1. Personalized Email or In-App Messages**

* “Hey [name], we miss you!”
* “People you follow have posted recently”.

**2. Gamification**

* Badges for returning
* "Streaks" for 3-day or 7-day activity.

**3. Incentives**

* Contests: "Post your best photo today!"
* Small rewards: coupons, digital stickers.

**4. Social Proof**

* Show them what their connections are doing
* "Your friend @alex posted a new photo".

**5. Feature Highlights**

* “New filters & editing tools just launched”
* "Now you can tag multiple friends in photos".

**6. Push Notifications (if app-based)**

* Remind when tagged, liked, or commented

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

**Code:**

WITH tag\_engagement AS (

SELECT

t.id AS tag\_id,

t.tag\_name,

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

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

COUNT(DISTINCT pt.photo\_id) AS total\_photos

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.id, t.tag\_name

),

engagement\_rate AS (

SELECT

tag\_name,

total\_likes,

total\_comments,

total\_photos,

(total\_likes + total\_comments) / total\_photos AS engagement\_per\_photo

FROM tag\_engagement

WHERE total\_photos > 0

)

SELECT \*

FROM engagement\_rate

ORDER BY engagement\_per\_photo DESC

LIMIT 10;

**Approach:**

**1. Step 1: Tag-Level Aggregation (tag\_engagement CTE)**

• Joined tags with photo\_tags, photos, likes, and comments to calculate:

* **Total likes** per tag → COUNT(DISTINCT l.user\_id)
* **Total comments** per tag → COUNT(DISTINCT c.id)
* **Total photos** associated with the tag → COUNT(DISTINCT pt.photo\_id)

**2. Step 2: Engagement Rate Calculation (engagement\_rate CTE)**

• Calculated **engagement per photo** for each tag:

“engagement\_per\_photo = (total\_likes + total\_comments) / total\_photos”

• Filtered out tags that have **zero photos** to avoid division errors.

**3. Step 3: Ranking and Output**

• Sorted all tags in **descending order** of engagement per photo.  
 • Returned **top 10 tags** with the highest average interaction.

**Hashtags based strategy:**

| **Use Case** | **How to Apply Top Hashtag Data** |
| --- | --- |
| **Content Strategy** | Promote content using high-engagement hashtags. |
| **User Recommendations** | Suggest popular hashtags during photo uploads. |
| **Influencer Campaigns** | Identify creators posting with top-performing tags. |
| **Ad Campaigns** | Run ads around high-performing topics (e.g. #fitness, #travel). |
| **Feature Hashtags** | Boost discoverability by trending hot tags on explore pages. |

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?

**Code:**

-- Hour based engagement:

WITH photo\_stats AS (

SELECT

p.id AS photo\_id,

HOUR(p.created\_at) AS post\_hour

FROM photos p

),

likes\_count AS (

SELECT

photo\_id,

COUNT(\*) AS total\_likes

FROM likes

GROUP BY photo\_id

),

comments\_count AS (

SELECT

photo\_id,

COUNT(\*) AS total\_comments

FROM comments

GROUP BY photo\_id

)

SELECT

ps.post\_hour,

COUNT(DISTINCT ps.photo\_id) AS total\_posts,

COALESCE(SUM(lc.total\_likes), 0) AS total\_likes,

COALESCE(SUM(cc.total\_comments), 0) AS total\_comments,

ROUND(

(COALESCE(SUM(lc.total\_likes), 0) + COALESCE(SUM(cc.total\_comments), 0)) /

COUNT(DISTINCT ps.photo\_id), 2 ) AS avg\_engagement\_per\_post

FROM photo\_stats ps

LEFT JOIN likes\_count lc ON ps.photo\_id = lc.photo\_id

LEFT JOIN comments\_count cc ON ps.photo\_id = cc.photo\_id

GROUP BY ps.post\_hour

ORDER BY post\_hour;

**Approach: Hourly Post Engagement Analysis**

**Objective:**  
Analyse **user engagement (likes & comments)** by the **hour of day** a post was created, and identify the best posting hours.

**1. Extract Post Hour (photo\_stats CTE):**

* Selected photo\_id and extracted HOUR(created\_at) from the photos table.
* This helps group engagement data based on the **time of day** each photo was posted.

**2. Aggregate Likes per Photo (likes\_count CTE):**

* Counted total number of likes per photo\_id.

**3. Aggregate Comments per Photo (comments\_count CTE):**

* Counted total number of comments per photo\_id.

**4. Join Engagements with Hour Data (Main Query):**

* Joined photo\_stats with likes\_count and comments\_count using photo\_id.
* Grouped by post\_hour to calculate metrics per hour:
  + total\_posts → how many photos were posted at that hour
  + total\_likes and total\_comments → summed from joined tables
  + avg\_engagement\_per\_post = (likes + comments) / total posts

**5. Handled Missing Data:**

* Used COALESCE() to treat missing likes/comments as zero, ensuring no null errors in calculations.

**6. Ordering:**

* Ordered results by post\_hour (0–23) to get a **chronological view of engagement trends**.

**Code:**

**-- weekday based engagement**

WITH photo\_stats AS (

SELECT

p.id AS photo\_id,

DAYNAME(p.created\_at) AS post\_day

FROM photos p

),

likes\_count AS (

SELECT

photo\_id,

COUNT(\*) AS total\_likes

FROM likes

GROUP BY photo\_id

),

comments\_count AS (

SELECT

photo\_id,

COUNT(\*) AS total\_comments

FROM comments

GROUP BY photo\_id

)

SELECT

ps.post\_day,

COUNT(DISTINCT ps.photo\_id) AS total\_posts,

COALESCE(SUM(lc.total\_likes), 0) AS total\_likes,

COALESCE(SUM(cc.total\_comments), 0) AS total\_comments,

ROUND(

(COALESCE(SUM(lc.total\_likes), 0) + COALESCE(SUM(cc.total\_comments), 0)) /

COUNT(DISTINCT ps.photo\_id),

2

) AS avg\_engagement\_per\_post

FROM photo\_stats ps

LEFT JOIN likes\_count lc ON ps.photo\_id = lc.photo\_id

LEFT JOIN comments\_count cc ON ps.photo\_id = cc.photo\_id

GROUP BY ps.post\_day

ORDER BY ps.post\_day;

**Approach- Weekday Engagement Analysis:**

**Objective:**  
Determine which **day of the week** generates the **highest average engagement per post** (likes + comments).

**1. Extract Post Day (photo\_stats CTE):**

* From the photos table, selected each photo\_id and extracted the **weekday name** using DAYNAME(created\_at).

**2. Aggregate Likes (likes\_count CTE):**

* Calculated the total number of **likes per photo**.

**3. Aggregate Comments (comments\_count CTE):**

* Calculated the total number of **comments per photo**.

**4. Join and Aggregate Engagement by Day (Main Query):**

* Joined the photo\_stats table with both likes and comments CTEs using photo\_id.
* Grouped data by post\_day to compute:
  + total\_posts → number of posts on that day
  + total\_likes and total\_comments → total interactions
  + avg\_engagement\_per\_post = (likes + comments) / posts

**5. Handled Missing Data:**

* Used COALESCE() to replace nulls with 0, ensuring accurate totals when likes or comments are missing.

**6. Sorted by Day Name:**

* Used ORDER BY ps.post\_day, which will sort by **day name alphabetically** (not week order unless customized).

| **Insight** | **Actionable Strategy** |
| --- | --- |
| Highest engagement hours of a day | Schedule content/ad drops in this time slot |
| Highest engagement week  Days (weekdays, weekends) | Launch campaigns or contests on weekends |

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?

**Code:**

WITH followers\_count AS (

SELECT

followee\_id AS user\_id,

COUNT(follower\_id) AS total\_followers

FROM follows

GROUP BY followee\_id

),

posts\_count AS (

SELECT

user\_id,

COUNT(\*) AS total\_posts

FROM photos

GROUP BY user\_id

),

likes\_count AS (

SELECT

p.user\_id,

COUNT(l.user\_id) AS total\_likes

FROM photos p

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

GROUP BY p.user\_id

),

comments\_count AS (

SELECT

p.user\_id,

COUNT(c.id) AS total\_comments

FROM photos p

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

GROUP BY p.user\_id

)

SELECT

u.id,

u.username,

COALESCE(fc.total\_followers, 0) AS followers,

COALESCE(pc.total\_posts, 0) AS posts,

COALESCE(lc.total\_likes, 0) AS likes,

COALESCE(cc.total\_comments, 0) AS comments,

ROUND(

(COALESCE(lc.total\_likes, 0) + COALESCE(cc.total\_comments, 0)) /

NULLIF(pc.total\_posts, 0), 2

) AS engagement\_rate

FROM users u

LEFT JOIN followers\_count fc ON u.id = fc.user\_id

LEFT JOIN posts\_count pc ON u.id = pc.user\_id

LEFT JOIN likes\_count lc ON u.id = lc.user\_id

LEFT JOIN comments\_count cc ON u.id = cc.user\_id

WHERE COALESCE(fc.total\_followers, 0) > 50 -- influencer threshold

AND pc.total\_posts >=5 -- influencer threshold

ORDER BY engagement\_rate DESC, followers DESC

limit 20;

**Approach-Top Potential Influencers Based on Followers and Engagement:**

**Objective:**  
Identify users with influencer potential by analyzing those who have:

* A significant follower base (>50 followers)
* At least 5 posts
* A high engagement rate (likes + comments per post)

**1. Follower Count (followers\_count CTE):**

* Counted how many followers each user (followee) has from the follows table.
* Output: user\_id, total\_followers.

**2. Post Count (posts\_count CTE):**

* Counted total posts made by each user from the photos table.
* Output: user\_id, total\_posts.

**3. Likes per User (likes\_count CTE):**

* Joined photos and likes to count how many likes each user’s photos received.
* Output: user\_id, total\_likes.

**4. Comments per User (comments\_count CTE):**

* Joined photos and comments to count total comments received on each user’s posts.
* Output: user\_id, total\_comments.

**5. Final Selection & Filtering (Main Query):**

* Joined all the CTEs with the users table to build a complete engagement profile.
* Used COALESCE() to safely handle any nulls (missing data).
* Calculated engagement rate as: “(likes + comments) / posts”
* using NULLIF() to avoid division by zero.
* Filtered:
  + followers > 50 → basic threshold for influencer reach.
  + posts >= 5 → ensures enough content exists for meaningful engagement.

**6. Ordering & Limiting:**

* Sorted by engagement\_rate DESC, then followers DESC.
* Limited to top 20 users with highest potential.

**Target the Right Influencers:**

* High engagement\_rate + strong follower base = **high trust & ROI**
* Niche-based (we can add tag filters later to target by hashtags: travel, food, etc.)

**How to Collaborate:**

1. **DM or email them** from brand/influencer account reachout.
2. **Propose clear value**: offer free products, affiliate commission, or sponsored pay,
3. **Offer value :** early access, giveaways, or co-branded content opportunities.
4. **Use engagement analytics**: Share how their current reach aligns with your campaign
5. **Track campaigns**: Use hashtags to track performance

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

**Code:**

WITH followers\_count AS (

SELECT

followee\_id AS user\_id,

COUNT(\*) AS followers

FROM follows

GROUP BY followee\_id

),

posts\_count AS (

SELECT

user\_id,

COUNT(\*) AS posts

FROM photos

GROUP BY user\_id

),

likes\_count AS (

SELECT

user\_id,

COUNT(\*) AS likes

FROM likes

GROUP BY user\_id

),

comments\_count AS (

SELECT

user\_id,

COUNT(\*) AS comments

FROM comments

GROUP BY user\_id

),

engagement\_data AS (

SELECT

u.id,

u.username,

COALESCE(f.followers, 0) AS followers,

COALESCE(p.posts, 0) AS posts,

COALESCE(l.likes, 0) AS likes,

COALESCE(c.comments, 0) AS comments

FROM users u

LEFT JOIN followers\_count f ON u.id = f.user\_id

LEFT JOIN posts\_count p ON u.id = p.user\_id

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

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

),

segmented\_users AS (

SELECT \*,(likes + comments) / NULLIF(posts, 0) as engage,

CASE

WHEN followers >= 50 AND posts >= 5 AND (likes + comments) / NULLIF(posts, 0) > 20 THEN 'Influencer'

WHEN posts < 2 AND (likes + comments) >= 50 THEN 'Liker & Commenter'

WHEN posts <= 4 AND (likes + comments) >= 50 THEN 'Active Creator'

ELSE 'Passive User'

END AS user\_segment

FROM engagement\_data

)

SELECT

user\_segment,

COUNT(\*) AS user\_count

FROM segmented\_users

GROUP BY user\_segment

ORDER BY user\_count DESC;

**Approach- User Segmentation Based on Engagement & Activity:**

**Objective:**  
 Categorize users into **distinct behavioural segments**—Influencers, Active Creators, Likers & Commenters, and Passive Users—based on their activity data.

**1. CTEs for Aggregating Metrics:**

* followers\_count
  + From the follows table, counts how many followers each user has (followee\_id grouped).
  + Output: user\_id, followers.
* posts\_count
  + From the photos table, counts total posts made by each user.
  + Output: user\_id, posts.
* likes\_count
  + From the likes table, counts total likes performed by each user.
  + Output: user\_id, likes.
* comments\_count
  + From the comments table, counts total comments made by each user.
  + Output: user\_id, comments.

**2. engagement\_data CTE: Merging User Stats**

* Joins all the metrics above to the users table using LEFT JOIN.
* Uses COALESCE() to replace missing data with 0.
* Creates a consolidated table with all key engagement metrics per user:
  + followers, posts, likes, comments

**3. segmented\_users CTE: Apply Segmentation Logic**

* Calculates engagement per post: (likes + comments) / posts
* Applies conditional logic using CASE to assign users into segments:

| Segment Name | Criteria |
| --- | --- |
| Influencer | followers ≥ 50, posts ≥ 5, engagement rate > 20 |
| Liker & Commenter | posts < 2, total likes + comments ≥ 50 |
| Active Creator | posts ≤ 4, total likes + comments ≥ 50 |
| Passive User | All others |

**4. Final Query: Segment Count Summary**

* Counts how many users fall into each user\_segment.
* Sorted in descending order by user\_count.

**User\_Segments:**

| **Segment** | **Campaign Ideas** |
| --- | --- |
| **Influencers** | Brand partnerships, product launches, paid promotions |
| **Active Creators** | Creator incentives (badges, feature spotlights, analytics tools) |
| **Likers & Commenters** | Encourage them to start posting (“Your likes matter — share your story!”) |
| **Passive Users** | Re-engagement push: story suggestions, friend activity, post reminders |

**Insights and Recommendations:**

**1)Improve Posting Behaviour**

**Insights:**

* High percentage of users have 0 posts.
* Significant drop-off observed after users make 5 posts.
* Very few power users with more than 6 posts.
* Moderate number of users post 1–2 times, showing initial interest.

**Recommendations:**

* Onboard campaigns for 0-post users.
* Gamify post streaks to push beyond 5 posts.
* Offer creator tools to boost high-frequency posting.
* Promote trending content to boost 1–2 post users.

**2. Improve Users Commenting Behavior:**

**Insights:**

* Peak user count found in the 71–80 comment range.
* Many users fall under 40 total comments.
* A small set of users have made 251–260 comments.
* Sparse activity observed in the 41–60 comment range.

**Recommendations:**

* Celebrate 71–80 comment users with public recognition.
* Nudge <40 commenters with targeted conversation alerts.
* Involve 251+ commenters in moderation or featured roles.
* Use small milestone goals to push 41–60 commenters further.

**3. Improve Users Likes Given and received**

**Insights:**

* Majority of users are in the 91–100 likes range.
* A set of super users liked 251–260 posts.
* 23 users have <1 like, indicating low engagement.
* Drop observed in the 61–70 and 101–110 like ranges.

**Recommendations:**

* Reward 91–100 range users with badges or exclusive features.
* Spotlight 251+ like users as power contributors.
* Onboard <1 like users with content suggestions.
* Push 81–100 users using streak or reward challenges.
* Investigate low-like zones for potential UX improvements.

**4. Improve Hashtag Usage**

**Insights:**

* #smile and #beach are the most used tags.
* Emotion and fun-based tags like #fun and #lol dominate.
* Lifestyle tags such as #food and #beauty show steady use.
* Users show varied tag behavior, suggesting diverse interests.

**Recommendations:**

* Promote #smile and #beach in trending suggestions.
* Recommend emotional tags to boost early engagement.
* Build events around lifestyle categories.
* Encourage multi-tag use for discovery.

**5. User Segmentation strategy**

**Insights:**

* Active Creators form the largest user group (35%).
* Likers & Commenters are the second largest (31%).
* Passive Users make up a significant portion (23%).
* Influencers are the smallest segment (11%) but highly valuable.

**Recommendations:**

* Reward Active Creators with visibility and creator tools.
* Motivate Likers & Commenters to start posting.
* Onboard Passive Users with follow/content suggestions.
* Partner with Influencers for feature promos or campaigns.

**6. Overall Engagement Strategy**

**Retention**

* Badge system, post reminders, personalized content nudges.

**Engagement**

* Boost posts with trending tags, prompt posts during peak hours.

**Acquisition**

* Enhance onboarding with "suggested users," promote popular tags.

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

**Key Metrics to Measure Effectiveness:**

| **Metric** | **Formula** | **Why It Matters** |
| --- | --- | --- |
| **Click-Through Rate (CTR)** | clicks / impressions | Measures how compelling your ad is. High CTR means your ad is engaging and relevant. |
| **Conversion Rate (CVR)** | conversions / clicks | Tells you how effective your landing page or funnel is after users click the ad. |
| **Cost Per Click (CPC)** | spend / clicks | Helps control cost efficiency. Lower CPC means you're getting more traffic for less. |
| **Cost Per Conversion (CPA)** | spend / conversions | Shows how expensive it is to acquire one customer or action. Lower is more efficient. |
| **Return on Ad Spend (ROAS)** | revenue\_generated / spend | Indicates profitability of your campaign. ROAS > 1 means positive return. |
| **Engagement per Impression** | (clicks + conversions) / impressions | Combines two key actions to gauge overall interaction per view. Higher is better. |

**Optimization Strategy**

**1. Identify high CTR campaigns**

* What visuals, copy, or CTA performed well?
* Replicate themes from successful campaigns

**2. Improve low CVR campaigns**

* If CTR is high but conversions are low:
  + Look at landing page experience
  + Consider A/B testing for post-click funnel

**3. Cut campaigns with high CPA**

* Prioritize cost-effective campaigns
* Redirect budget to high-ROAS ads

**4. Day-of-Week & Time-Based Trends**

* Analyse which time periods get better conversions

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

### **Key Metrics to Identify Advocates**

| Metric | Description |
| --- | --- |
| **Engagement Rate** | (Likes + Comments) / Posts – reflects how interactive their audience is |
| **Post Frequency** | Consistent content creation shows commitment and reliability |
| **Follower Count** | Indicates reach and potential for visibility |
| **Engagement per Follower** | Normalizes engagement to account for follower size |
| **Tag Usage** | Frequent tagging may suggest active participation in trends or challenges |
| **Mentions of Instagram Campaigns** | Indicates alignment with Instagram's initiatives or values |

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

**Step-by-Step Approach: If Questions Were Not Given**

**Step 1: Understand the Business Model**

* What is the platform's goal? (e.g., increase engagement, grow user base, promote ads)
* Who are the users? What actions can they take? (posts, comments, likes, follows, etc.)

**Step 2: Explore the Schema & Relationship:**

Examine the existing tables:

* users, photos, comments, likes, follows, tags, photo\_tags
* Ask: What relationships exist? What metrics can I derive from them?

**Step 3: Define Key Areas of Interest:**

Based on the schema, define **buckets** for analysis:

| **Area** | **Questions You Can Create Yourself** |
| --- | --- |
| **User Engagement** | Who are the most active users? Least? Who posts or likes most? |
| **Content Performance** | Which photos get the most likes/comments? Are some tags more popular? |
| **Network Growth** | Who has the most followers? Who gains/loses followers fast? |
| **Platform Health** | How many users are inactive? What % of users contribute 80% of posts? |
| **Timing Trends** | Which hours/days have most activity? |

**Step 4: Design Metrics to Track:**

Even without questions, you can create metrics like:

* Average likes per photo
* Comments-to-post ratio
* Engagement per user
* Top 10 hashtags
* Growth in user activity by week/month

**Step 5: Segment the Users:**

Divide users into:

|  |
| --- |
| **Influencers** |
| **Active Creators** |
| **Likers & Commenters** |
| **Passive Users** |

Each group can be analysed for patterns and targeted strategies.

1. 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?

UPDATE User\_Interactions

SET Engagement\_Type = 'Heart'

WHERE Engagement\_Type = 'Like';

**Approach:**

UPDATE User\_Interactions: updates the table.

SET Engagement\_Type = 'Heart': the new value.

WHERE Engagement\_Type = 'Like': ensures only

"Like" values are changed to “Heart”.