**SHRADDHA THAKUR**

**MYSQL PROJECT**

**INSTAGRAM ANALYSIS**

**Project Title:** Instagram Data Analysis with MySQL

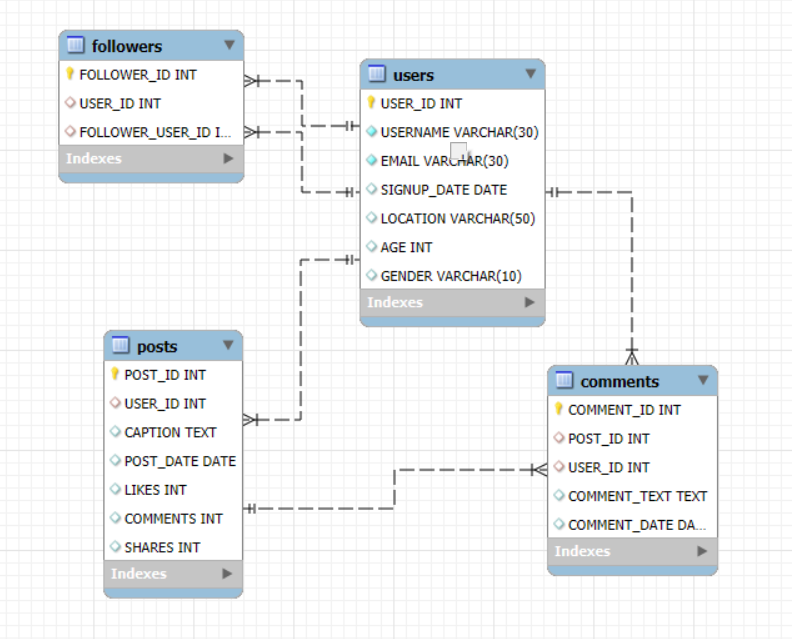
**Objective:** The primary objective of this project is to analyse user engagement and interactions on Instagram by structured data stored in a MySQL database. This analysis will provide insights into user behaviour, content popularity, and the dynamics of social connections on the platform.

**Project Description:**

This project involves designing and implementing a MySQL database to store and analyse data from an Instagram-like social media platform.

**The Entity-Relationship (ER):**

The Entity-Relationship (ER) diagram for the social media platform analysis project.



--- CREATE DATABASE :

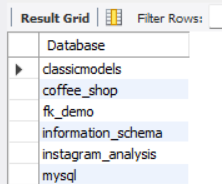
SYNTAX:

CREATE DATABASE INSTAGRAM\_ANALYSIS;

--- TO SEE CREATED DATABASES

SYNTAX:

SHOW DATABASES;



--- TO SELECT/USE DATABASE

SYNTAX:

USE INSTAGRAM\_ANALYSIS;

--- TO CREATE TABLE INTO DATABASE

CREATING 5 TABLES INTO INSTAGRAM\_ANALYSIS DATABASE

SYNTAX:

TABLE 1: USERS

CREATE TABLE USERS ( USER\_ID INT PRIMARY KEY, USERNAME VARCHAR(30) UNIQUE NOT NULL, EMAIL VARCHAR(30) NOT NULL, SIGNUP\_DATE DATE, LOCATION VARCHAR(50),AGE INT,GENDER VARCHAR(10));

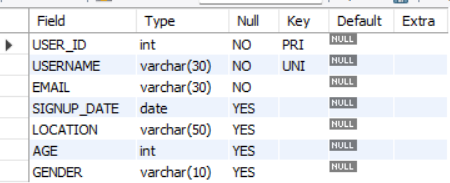


TABLE 2: POSTS

CREATE TABLE POSTS (POST\_ID INT PRIMARY KEY AUTO\_INCREMENT,USER\_ID INT,CAPTION TEXT,POST\_DATE DATE,LIKES INT,COMMENTS INT,SHARES INT,FOREIGN KEY (USER\_ID) REFERENCES USERS(USER\_ID));

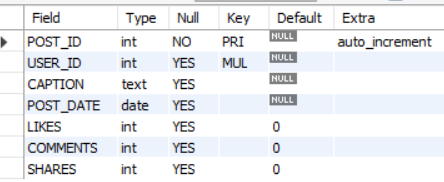


TABLE 3: COMMENTS

CREATE TABLE COMMENTS (COMMENT\_ID INT PRIMARY KEY AUTO\_INCREMENT,POST\_ID INT,USER\_ID INT,COMMENT\_TEXT TEXT, COMMENT\_DATE DATE,FOREIGN KEY (POST\_ID) REFERENCES POSTS(POST\_ID),FOREIGN KEY (USER\_ID) REFERENCES USERS(USER\_ID));

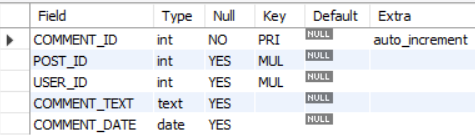


TABLE 4: FOLLOWERS

CREATE TABLE FOLLOWERS (FOLLOWER\_ID INT PRIMARY KEY AUTO\_INCREMENT,USER\_ID INT,FOLLOWER\_USER\_ID INT,FOREIGN KEY (USER\_ID) REFERENCES USERS(USER\_ID),FOREIGN KEY (FOLLOWER\_USER\_ID) REFERENCES USERS(USER\_ID));

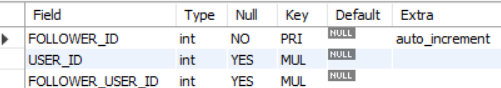
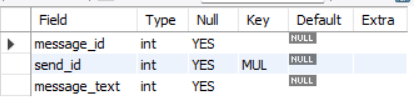


TABLE 5: MESSAGES

CREATE TABLE MESSAGES ( MESSAGE\_ID INT, SEND\_ID INT, MESSAGE\_TEXT INT, FOREIGN KEY (SEND\_ID) REFERENCES USERS (USER\_ID));



ALTER TABLE

--- ADD DEFAULT CONSTRAINT IN POST TABLE TO COLUMNS LIKES, COMMENTS AND SHARE TO BE 0

SYNTAX:

* ALTER TABLE POSTS ALTER LIKES SET DEFAULT 0;
* ALTER TABLE POSTS ALTER COMMENTS SET DEFAULT 0;
* ALTER TABLE POSTS ALTER SHARES SET DEFAULT 0;

--- CHANGE ATTRIBUTE NAME IN TABLE MESSAGES FROM SEND\_ID TO SENDER\_ID AND ALSO ADD FOREIGN KEY CONSTRAINT REFERENCING TABLE USERS (USER\_ID

SYNTAX:

* ALTER TABLE MESSAGES CHANGE SEND\_ID SENDER\_ID INT;
* ALTER TABLE MESSAGES

ADD CONSTRAINT fk\_user\_id

FOREIGN KEY (SENDER\_ID) REFERENCES USERS(USER\_ID);

--- ADD COLUMN RECEIVER ID IN TABLE MESSAGES BY REFERENCING TABLE USERS (USER\_ID)

SYNTAX:

* ALTER TABLE MESSAGES ADD COLUMN RECEIVER\_ID INT AFTER SENDER\_ID;
* ALTER TABLE MESSAGES

ADD CONSTRAINT FK\_RECEIVER\_ID

FOREIGN KEY (RECEIVER\_ID) REFERENCES USERS(USER\_ID);

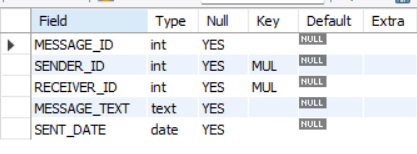
--- ADD COLUMN SENT\_DATE IN TABLE MESSAGES

ALTER TABLE MESSAGES ADD COLUMN SENT\_DATE DATE;

--- CHANGE DATA TYPE OF MESSAGE TEXT TO TEXT IN TABLE MESSAGES

SYNTAX:

ALTER TABLE MESSAGES MODIFY MESSAGE\_ID TEXT;



INSERT

ADDING RECORDS IN ALL TABLES

TABLE 1: USERS

INSERT INTO USERS ( USER\_ID,USERNAME,EMAIL,SIGNUP\_DATE,LOCATION,AGE,GENDER) VALUES (1, 'arjun\_sharma', 'arjun.sharma@gmail.com', '2020-01-15', 'Delhi', 25, 'M'),

(2, 'priya\_kapoor', 'priya.kapoor@gmail.com', '2020-02-20', 'Mumbai', 28, 'F'),

(3, 'rahul\_verma', 'rahul.verma@gmail.com', '2020-03-05', 'Delhi', 32, 'M'),

(4, 'sneha\_patel', 'sneha.patel@gmail.com', '2020-04-10', 'Ahmedabad', 26, 'F');

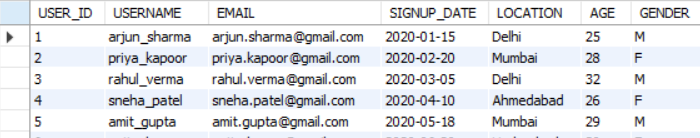


TABLE 2: POSTS

INSERT INTO POSTS (USER\_ID, CAPTION, POST\_DATE, LIKES, COMMENTS, SHARES) VALUES

(10,'Exploring the beauty of Delhi!', '2023-01-15', 120, 15, 10),

(20,'Love the vibe in Mumbai!', '2023-01-20', 200, 30, 20),

(3,'Had an amazing time in Bangalore.', '2023-02-05', 180, 22, 18),

(6 ,'Sunset in Ahmedabad was breathtaking.', '2023-02-15', 95, 12, 8),

(12,'Chennai, you never disappoint!', '2023-02-25', 210, 34, 25);

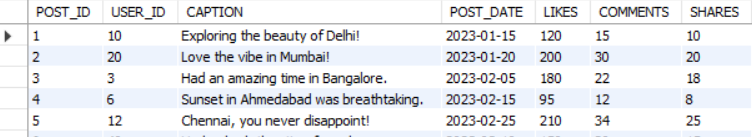


TABLE 3: COMMENTS

INSERT INTO COMMENTS (POST\_ID, USER\_ID, COMMENT\_TEXT, COMMENT\_DATE) VALUES

(1, 2, 'Great post! Really enjoyed the view.', '2023-01-16'),

(1, 3, 'The photo is stunning, where was this taken?', '2023-01-17'),

(2, 1, 'Mumbai is fantastic! I miss it.', '2023-01-21'),

(2, 4, 'Love the energy of this city!', '2023-01-22'),

(3, 5, 'Bangalore looks amazing in this light.', '2023-02-06'),

(3, 6, 'I need to visit Bangalore soon.', '2023-02-07');

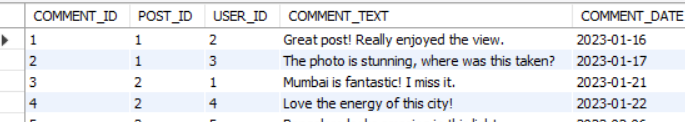


TABLE 4: FOLLOWERS

INSERT INTO Followers (user\_id, follower\_user\_id) VALUES (1,2),(1,3),(1,4),(1,5),(2,1),(2,3),(2,4),(2,5),(3,1),(3,2),(3,4),(3,5);

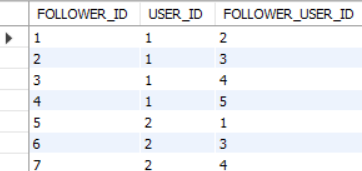


TABLE 5: MESSAGES

INSERT INTO MESSAGES (MESSAGE\_ID, SENDER\_ID, RECEIVER\_ID, MESSAGE\_TEXT, SENT\_DATE) VALUES

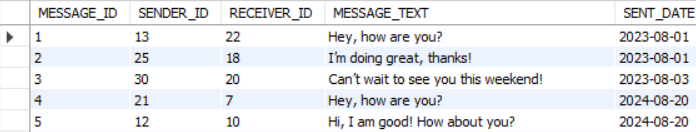
(1,13,22, 'Hey, how are you?', '2023-08-01'),

(2,25, 18, 'I’m doing great, thanks!', '2023-08-01'),

(3,30, 20, 'Can’t wait to see you this weekend!', '2023-08-03'),

(4,21, 7, 'Hey, how are you?', '2024-08-20'),

(5,12, 10, 'Hi, I am good! How about you?', '2024-08-20');



--- DELETE ALL RECORDS FROM TABLE MESSAGES

SYNTAX:

TRUNCATE TABLE MESSAGES;

--- RENAME TABLE MESSAGES TO SMS

SYNTAX:

RENAME TABLE MESSAGES TO SMS;

--- DELETE DATA AS WELL AS STRUCTURE OF THE SMS TABLE

SYNTAX:

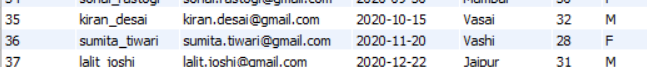
DROP TABLE SMS;

UPDATE

--- UPDATE USERS EMAIL AS “ Sumita.tiwari1@gmail.com” WHERE USER ID IS 36.

SYNTAX:

UPDATE USERS SET EMAIL='Sumita.tiwari1@gmail.com' where user\_id = 36;





--- UPDATE LIKES COUNT BY 10 FOR POST ID IS 12

SYNTAX:

UPDATE POSTS SET LIKES = LIKES + 10 WHERE POST\_ID = 12;





DELETE

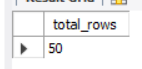
--- DELETE RECORD FROM COMMENTS WHOSE COMMENT ID IS 8

SYNTAX:

DELETE FROM COMMENTS WHERE COMMENT\_ID = 8;

--- CALCULATE TOTAL NUMBER OF ROWS IN THE TABLES

* SELECT COUNT(\*) AS TOTAL\_ROWS FROM USERS;
* SELECT COUNT(\*) AS TOTAL\_ROWS FROM POSTS;
* SELECT COUNT(\*) AS TOTAL\_ROWS FROM COMMENTS;
* SELECT COUNT(\*) AS TOTAL\_ROWS FROM FOLLOWERS;



--- CALCULATE TOTAL NUMBER OF COLUMNS INT THE TABLES

SELECT COUNT(\*) AS total\_columns

FROM INFORMATION\_SCHEMA.COLUMNS

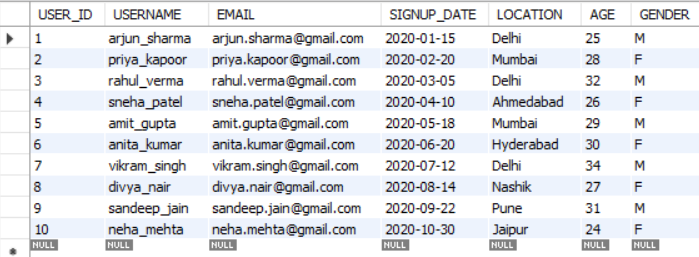
WHERE table\_name = 'USERS' AND table\_schema = 'instagram\_analysis';



NOTE:[CHANGE TABLE NAME FOR PARTICULAR TABLES]

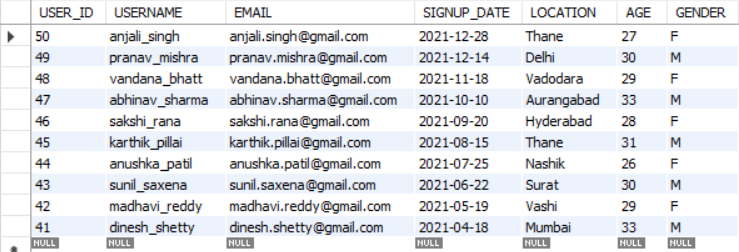
-- TO SEE FIRST 10 RECORDS FROM TABLE USERS

SELECT \* FROM USERS LIMIT 10;



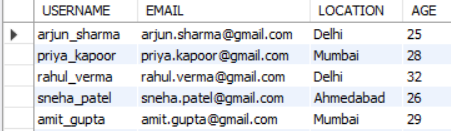
-- TO SEE LAST 10 RECORDS FROM TABLE USERS

SELECT \* FROM USERS ORDER BY USER\_ID DESC LIMIT 10;



-- TO DISPLAY ONLY USERNAME,EMAIL,LOCATION,AGE

SELECT USERNAME,EMAIL,LOCATION,AGE FROM USERS;



-- DISPLAY DIFFERENT LOCATIONS IN TABLE USERS

SELECT DISTINCT LOCATION FROM USERS;

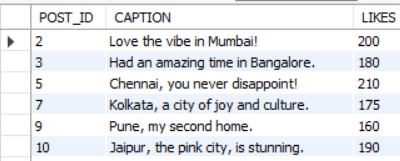


-- CHECK NULL VALUE IN TABLE FOLLOWER WHERE FOLLOWER ID IS NOT NULL

SELECT COUNT(\*) FROM FOLLOWERS WHERE FOLLOWER\_ID IS NOT NULL;

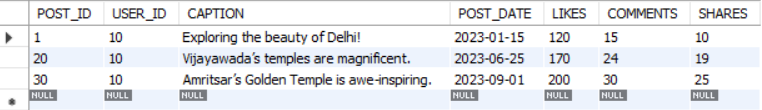
-- QUERY TO FIND POST THAT HAVE RECEIVED MORE THAM 150 LIKE

SELECT POST\_ID,CAPTION,LIKES FROM POSTS WHERE LIKES > 150;



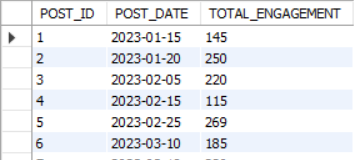
-- FIND POSTS WHERE LIKES ARE MORE THAN 100 AND LIKED BY USER\_ID 10

SELECT \* FROM POSTS WHERE USER\_ID = 10 AND LIKES > 100;



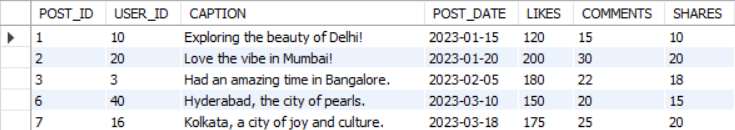
-- CALCULATE THE TOTAL ENGAGEMENT FOR EACH POST BY ADDING LIKES,COMMENTS,SHARES

SELECT POST\_ID,POST\_DATE,(LIKES + COMMENTS + SHARES) AS TOTAL\_ENGAGEMENT FROM POSTS;



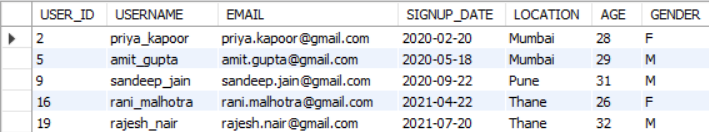
-- DISPLAY VALUES FROM TABLE POSTS WHERE POSTS HAS LIKES BETWEEN 100-200

SELECT \* FROM POSTS WHERE LIKES BETWEEN 100 AND 200;



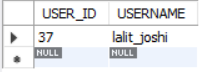
-- DISPLAY VALUES FROM TABLE USERS WHERE USERS BELONGS FROM THANE,MUMBAI,PUNE

SELECT \* FROM USERS WHERE LOCATION IN ('THANE','MUMBAI','PUNE');



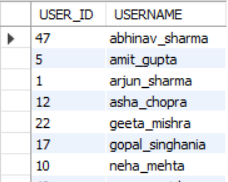
-- FIND USERS WHOSE NAME STARTS WITH LETTER "L"

SELECT USER\_ID,USERNAME FROM USERS WHERE USERNAME LIKE 'L%';



-- FIND USERWHOSE NAME ENDS WITH LETTER "A"

SELECT USER\_ID,USERNAME FROM USERS WHERE USERNAME LIKE '%A';



-- CALCULATE MINIMUM,MAXIMUM,AVERAGE LIKES OF THE POSTS

SELECT MIN(LIKES) AS MINIMUM\_LIKES,

MAX(LIKES) AS MAXIMUM\_LIKES,

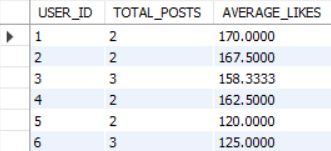
AVG(LIKES) AS AVERAGE\_LIKES

FROM POSTS;



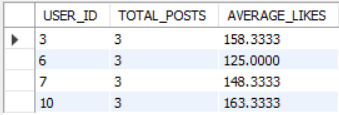
-- CALCULATE TOTAL NUMBER OF POSTS AND AVERAGE LIKES PER USER

SELECT USER\_ID, COUNT(POST\_ID) AS TOTAL\_POSTS, AVG(LIKES) AS AVERAGE\_LIKES FROM POSTS GROUP BY USER\_ID;



-- FIND USERS WHO HAVE MADE MORE THAN 2 POSTS AND HAVE AVERAGE OF 100 LIKES PER POST

SELECT USER\_ID, COUNT(POST\_ID) AS TOTAL\_POSTS, AVG(LIKES) AS AVERAGE\_LIKES FROM POSTS GROUP BY USER\_ID HAVING COUNT(POST\_ID) > 2 AND AVG(LIKES) >100;

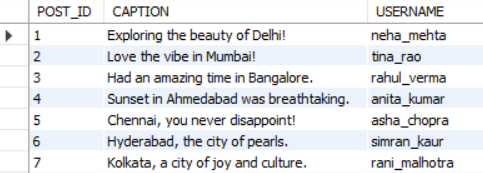


-- INNER JOIN

-- RETRIEVE POSTS ALONG WITH USERNAME OF THE AUTHOR

SELECT P.POST\_ID,P.CAPTION,U.USERNAME

FROM



POSTS AS P

JOIN

USERS AS U

ON

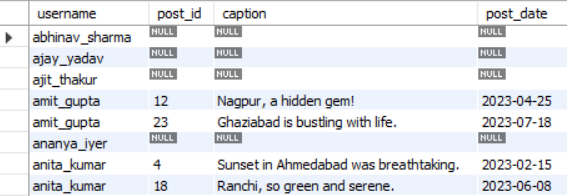
P.USER\_ID=U.USER\_ID;

-- LEFT JOIN

-- RETIEVE ALL USERS AND THERE POSTS (IS ANY)

SELECT U.username, P.post\_id, P.caption, P.post\_date

FROM



Users AS U

LEFT JOIN

Posts AS P

ON

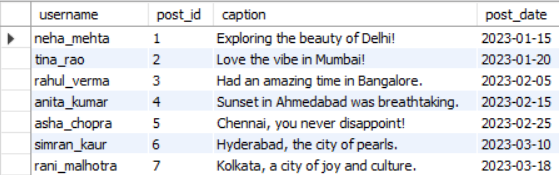
U.user\_id = P.user\_id;

-- RIGHT JOIN

-- RETRIEVE ALL POSTS AND THERE USERNAME WHO CREATED POSTS

SELECT U.username, P.post\_id, P.caption,P.post\_date

FROM



Users AS U

RIGHT JOIN

Posts AS P

ON

U.user\_id = P.user\_id;

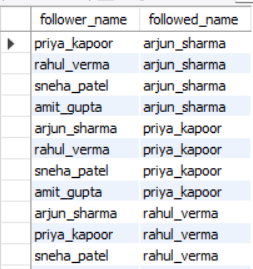
-- SELF JOIN

-- RETRIEVE ALL FOLLOWERS RELATION, SHOWING WHO FOLLOWS WHOM

SELECT u1.username AS follower\_name, u2.username AS followed\_name

FROM

Followers AS F



JOIN

Users u1

ON

F.follower\_user\_id = u1.user\_id

JOIN

Users u2

ON

F.user\_id = u2.user\_id;

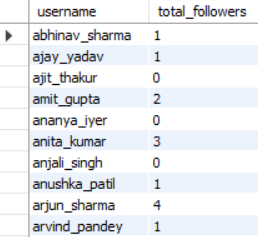
-- RETRIEVE EACH USERS USERNAME ALONG WITH THE TOTAL NUMBER OF FOLLOWERS THEY HAVE

SELECT username, (SELECT COUNT(\*) FROM Followers

WHERE Followers.user\_id = Users.user\_id) AS total\_followers

FROM

Users;



-- FIND ALL POSTS CREATED BY USERS WHO HAVE AT LEAST ONE FOLLOWER

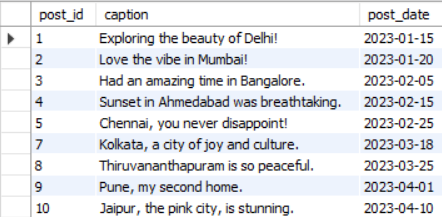
SELECT post\_id, caption, post\_date

FROM

Posts

WHERE

user\_id IN (SELECT DISTINCT user\_id FROM Followers);



-- CREATE A VIEW AS POPULARPOSTS SHOWING POST\_ID WITH MORE THAN 50 LIKES,CAPTION,USERNAME IN IT

CREATE VIEW PopularPosts AS

SELECT p.post\_id,p.caption, p.likes, u.username

FROM

Posts p

INNER JOIN

Users u ON p.user\_id = u.user\_id

WHERE

p.likes > 50;

-- VIEW POPULARPOSTS

select \* from popularposts;

