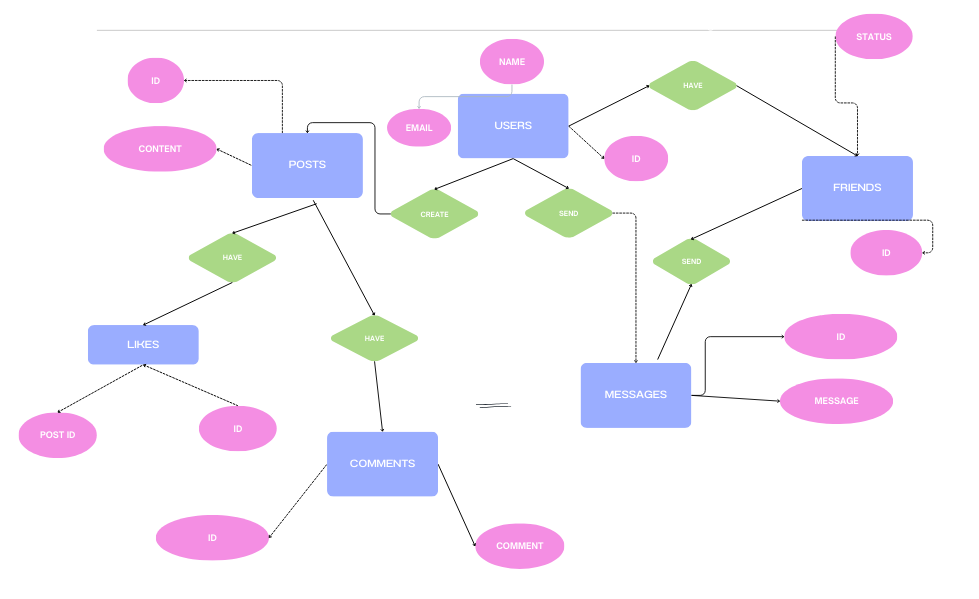
**SQL Project**

**Facebook Database**

**ER Diagram:--**



**Creating tables**

CREATE TABLE users (uid INT PRIMARY KEY, name VARCHAR(50), NOT NULL, email VARCHAR(40) UNIQUE NOT NULL, age VARCHAR(10), phone\_number VARCHAR(30) UNIQUE NOT NULL, date\_of\_birth DATE NOT NULL, gender VARCHAR(5), city VARCHAR(50), created\_at DATETIME DEFAULT current\_timestamp);

CREATE TABLE posts (p\_id INT PRIMARY KEY, uid INT, content VARCHAR(50) NOT NULL, image\_url VARCHAR(40), created\_at DATETIME DEFAULT current\_timestamp, CONSTRAINT FOREIGN KEY (uid) REFERENCES users(uid));

CREATE TABLE likes (like\_id VARCHAR(30) PRIMARY KEY, uid INT, p\_id INT, created\_at DATETIME DEFAULT current\_timestamp, CONSTRAINT FOREIGN KEY (uid) REFERENCES users(uid), CONSTRAINT FOREIGN KEY (p\_id) REFERENCES posts(p\_id));

CREATE TABLE friends (friend\_id VARCHAR(40) PRIMARY KEY, uid INT, p\_id INT, status VARCHAR(30) DEFAULT ‘Pending’, created\_at DATETIME DEFAULT current\_timestamp, CONSTRAINT FOREIGN KEY (uid) REFERENCES users(uid), CONSTRAINT FOREIGN KEY (p\_id) REFERENCES posts(p\_id));

CREATE TABLE comments (comment\_id VARCHAR(30) PRIMARY KEY, uid INT, p\_id INT, comment VARCHAR(50) NOT NULL, created\_at DATETIME DEFAULT current\_timestamp, CONSTRAINT FOREIGN KEY (uid) REFERENCES users(uid), CONSTRAINT FOREIGN KEY (p\_id) REFERENCES posts(p\_id));

CREATE TABLE messages (message\_id VARCHAR(30) PRIMARY KEY, sender\_id INT, receiver\_id INT, message VARCHAR(50) NOT NULL, created\_at DATETIME DEFAULT current\_timestamp, CONSTRAINT FOREIGN KEY (sender\_id) REFERENCES users(uid), CONSTRAINT FOREIGN KEY (receiver\_id) REFERENCES users(uid));

**Inserting data in** **tables :--**

INSERT INTO users (uid, name, email, phone\_number, date\_of\_birth, gender, city)

VALUES (1, ‘Addison Garcia’, [‘addigar@gmail.com’](mailto:‘addigar@gmail.com’), 3902843029, 09-08-2000, ‘F’, ‘New York’),

(2, ‘Ashley Davis’, [‘ashdavis@email.com’](mailto:‘ashdavis@email.com’), 578709430, 04-02-1998, ‘F’, ‘Chicago’),

(3, ‘Ethan Williams’, [‘ethan.w@gmail.com’](mailto:‘ethan.w@gmail.com’), 90450902, 17-04-1994, ‘M’, ‘Washington’),

(4, ‘Mark Smith’, [‘mksmith@mail.com’](mailto:‘mksmith@mail.com’), 75024989, 13-06-2013, ‘M’, ‘New York’),

(5, ‘Johnny Miller’, [‘millerj.y@gmail.com’](mailto:‘millerj.y@gmail.com’), 694837522, 12-09-1994, ‘M’, ‘Washington),

(6, 'Vincent Frank’, [‘vinfran23@email.com’](mailto:‘vinfran23@email.com’), 592019833, 04-12-1992, ‘M’, ‘Chicago’),

(7, ‘Candace Fischer’, [‘candypop@gmail.com’](mailto:‘candypop@gmail.com’), 43208032, 09-03-1993, ‘F’, ‘Springfield’),

(8, ‘Florence Garner’, [‘flowergarden@mail.com’](mailto:‘flowergarden@mail.com’), 897098036, 18-02-1994, ‘F’, ‘Austin’),

(9, ‘Mina Swanson’, [‘mina.swan7@email.com’](mailto:‘mina.swan7@email.com’), 239085904, 13-10-1995, ‘F’, ‘Austin’),

(10, ‘Rose Quinn’, [‘rosiequeen@gmail.com’](mailto:‘rosiequeen@gmail.com’), 7540285092, 30-12-1995', ‘F’, ‘Springfield’);

INSERT INTO posts (p\_id, uid, content)

VALUES (101, 1, ‘My best moment is yet to come’),

(102, 2, ‘Hello WORLD’),

(103, 3, ‘I am on the next level’),

(104, 3, ‘you should have fun’),

(105, 5, ‘Hey! You never walk alone’),

(106, 6, ‘please give me a remedy’),

(107, 7, ‘follow the lead of Maestro’),

(108, 2, ‘I am the one I should love in this world’),

(109, 9, ‘trust the process’),

(110, 1, ‘follow for more such content’);

INSERT INTO likes (like\_id, uid, p\_id)

VALUES

(#1, 1, 104), (#2, 2, 101), (#3, 3, 102), (#4, 4, 108), (#5, 5, 105), (#6, 6, 103), (#7, 7, 109), (#8, 8, 101), (#9, 9, 102), (#10, 10, 107), (#11, 4, 109), (#12, 1, 110), (#13, 6, 102), (#14, 8, 108), (#15, 2, 106);

INSERT INTO friends (friend\_id, uid, status)

VALUES ($1, 101, ‘Accepted’),

($2, 102, ‘Pending’),

($3, 103, ‘Pending’),

($4, 104, ‘Denied’),

($5, 105, ‘Accepted’),

($6, 106, ‘Pending’),

($7, 107, ‘Denied’),

($8, 108, ‘Denied’),

($9, 109, ‘Accepted’),

($10, 110, ‘Accepted’);

INSERT INTO comments (comment\_id, uid, p\_id, comment)

VALUES (A1, 1,101, ‘hello’),

(A2, 2, 102, ‘believe’),

(A3, 3, 103, ‘happy’),

(A4, 4, 104, ‘sweet’),

(A5, 5, 109, ‘looking good’),

(A6, 6, 107, ‘fashion street’),

(A7, 7, 106, ‘new jeans’),

(A8, 8, 105, ‘black magic’),

(A9, 9, 104, ‘true form’),

(A10, 3, 108, ‘secret garden’),

(A11, 10, 110, ‘not much’),

(A12, 8, 102, ‘friends of mine’),

(A13, 6, 103, ‘look at me’),

(A14, 9, 105, ‘mysterycase’),

(A15, 1, 108, ‘you are welcome’);

INSERT INTO messages (message\_id, sender\_id, receiver\_id, message)

VALUES (1&, 102, 103, ‘hey wassup’),

(2&, 108, 106, ‘what do you think’),

(3&, 109, 101, ‘please do my work too’) ,

(4&, 105, 107, ‘lets start a business’),

(5&, 109, 104, ‘i would like to buy this product’),

(6&, 108, 103, ‘are you free today?’),

(7&, 106, 109, ‘tell everyone to meet today’),

(8&, 110, 101, ‘i am very busy’),

(9&, 101, 109, ‘no i cannot do your work’),

(10&, 102, 104, ‘you do not remember it?’);

**Questions :--**

1. Get all users’ details from the Users table.

**Answer**: SELECT \* FROM users;

2. Get all posts created by a user with the user\_id = 1.

**Answer**: SELECT uid, p\_id FROM posts WHERE uid = 1;

3. Find all users from the city of ‘New York’

**Answer**: SELECT \* FROM users WHERE city = ‘New York’;

4. Get all friends of user with user\_id = 101 whose friendship status is “Accepted”

**Answer**: SELECT uid, friend\_id FROM friends WHERE status = “Accepted”;

5. Find all posts created between 2024-01-01 and 2024-09-01.

**Answer**: SELECT \* FROM posts WHERE created\_at BETWEEN ‘2024-01-01' AND ‘2024-09-01

6. Get details of users from the city of Chicago, Los Angeles, or San Francisco.

**Answer**: SELECT name, city FROM users WHERE name IN (‘Chicago’, ‘Los Angeles’, ‘San Francisco’);

7. Find all users whose email address ends with [‘@gmail.com’](mailto:‘@gmail.com’).

**Answer**: SELECT name, email FROM users WHERE email like ‘%@gmail.com’;

8. Get the number of posts created by each user.

**Answer**: SELECT uid, COUNT(posts.uid) FROM posts

LEFT JOIN users ON users.uid = posts.uid

GROUP BY posts.uid;

9. Get the total number of likes on each post.

**Answer**: SELECT p\_id, COUNT(likes.like\_id) FROM likes

LEFT JOIN posts ON likes.p\_id = posts.p\_id

GROUP BY p\_id;

10. Get all comments on a specific post along with the user name of the commenter.

**Answer**: SELECT p\_id, name, COUNT(comment\_id) FROM comments

LEFT JOIN users ON comments.uid = users.uid

GROUP BY comments.p\_id;

11. Get all posts liked by a specific user along with the post content and the user who

posted it.

**Answer**: SELECT p\_id, name, COUNT(like\_id) FROM likes

LEFT JOIN users ON likes.uid = users.uid

GROUP BY likes.p\_id;

12. Get the most liked post.

**Answer**: SELECT like\_id, p\_id FROM likes, MAX(COUNT(like\_id));

13. Find the users who haven&#39;t posted anything yet.

**Answer**: SELECT uid, p\_id FROM posts

RIGHT JOIN users ON posts.uid = users.uid;

14. Get the first 10 characters of each post’s content.

**Answer**: SELECT \* FROM posts, LEFT(content, 10);

15. Convert all user names to uppercase.

**Answer**: SELECT name, UPPER(name) FROM users;

16. Get the top 5 users who have received the most likes on their posts.

**Answer**: SELECT uid, like\_id, COUNT(like\_id) FROM likes

ORDER BY COUNT(like\_id) DESC

LIMIT 5;

17. Get the users who have more than 100 friends.

**Answer**: SELECT uid, friend\_id FROM friends

RIGHT JOIN users ON users.uid = friends.uid

GROUP BY uid;

18. Show the friendship status as ‘Friends’ if the status is ‘Accepted’, ‘Pending’ if its

pending, and ‘Not Friends’ for any other status.

**Answer**: SELECT friend\_id, status

CASE

WHEN status = ‘Accepted’ THEN ‘Friends’

WHEN status = ‘Pending’ THEN ‘Pending’

ELSE ‘Not Friends’

END AS status\_category

FROM friends;

19. Get all private messages between two users (user\_id = 101 and user\_id = 102).

**Answer**: SELECT \* FROM messages;

20. Categorize the post length as ‘Short’, ‘Medium’, or ‘Long’ based on character count

(e.g., < 50 characters = ‘Short’, 50-100 = ‘Medium’, >100 = ‘Long’).

**Answer**: SELECT p\_id, content

CASE

WHEN content < 50 THEN ‘Short’

WHEN content BETWEEN 50 AND 100 THEN ‘Medium’

ELSE ‘Long’

END AS post\_length

FROM posts;

21. Display the user’s age category as ‘Minor’ if their age is less than 18, ‘Adult’ if

their age is between 18 and 60, and ‘Senior’ if their age is greater than 60.

**Answer**: SELECT name, age,

CASE

WHEN age < 18 THEN ‘Minor’

WHEN age BETWEEN 18 AND 60 THEN ‘Adult’

ELSE ‘Senior’

END AS age\_category

FROM users;

Extra questions:-

1. Get user details of users who are male and live in Chicago
2. Total number of comments on posts
3. Number of posts created by users who are Senior
4. Most commented post

**Answer**s:-

1. SELECT \* FROM users WHERE gender = ‘M’ AND city = ‘Chicago’;
2. SELECT uid, COUNT(comment\_id); as total\_comments FROM comments;
3. SELECT p\_id, age FROM users

RIGHT JOIN posts ON posts.uid = users.uid

WHERE age > 60

GROUP BY uid;

1. SELECT MAX(COUNT(comment\_id)) FROM comments;