**ABSTRACT**

**TITLE:**  **Social Networking Site**

The Social Networking Site (SNS) is a popular online platform where users can connect with each other and share information. As part of a Database Management System (DBMS) project, designing and implementing an SNS database can be a challenging task. The database schema for an SNS includes tables for users, posts, comments, messages, and other entities. The DBMS must manage the storage, retrieval, and manipulation of this data while ensuring data integrity and security. To build an efficient SNS DBMS, the project team must optimize the database schema for fast query execution and efficient data storage. The DBMS should also incorporate features such as data backups, security protocols, and scalability options to accommodate the platform's growth. Additionally, the project can leverage advanced data analysis techniques, such as machine learning and data mining, to extract valuable insights from the SNS database. These insights can help improve user experience, identify trends in user behaviour, and inform marketing strategies.

In conclusion, designing and implementing an SNS database as part of a DBMS project can provide valuable experience in database design, optimization, and data analysis. The project offers opportunities to explore advanced technologies and develop solutions to complex challenges. By building a robust and scalable SNS DBMS, the project team can create a platform that supports social interaction and helps businesses leverage valuable insights to improve their strategies.

# CHAPTER 1. INTRODUCTION

**1.1 Introduction to the topic**

A social networking site DBMS project involves developing a robust database management system to support the functionalities of a social networking platform. This project aims to provide users with a seamless online experience, allowing them to connect, interact, and share content with others. The DBMS serves as the backbone of the platform, handling the storage, retrieval, and management of user profiles, posts, relationships, and other pertinent data.

In the first paragraph, introduce the concept of social networking sites and their widespread popularity in the digital era. Emphasize the need for an efficient and well-designed DBMS to handle the vast amount of data generated by users on these platforms. Explain that the project's objective is to create a reliable and scalable DBMS that enables smooth user interactions, facilitates content sharing, and maintains the integrity of user data.

The second paragraph should highlight the key features and functionalities of the social networking site DBMS project. Discuss aspects such as user registration, authentication, and profile management, emphasizing the importance of ensuring data security and privacy. Mention features like friend connections, news feeds, messaging systems, and content sharing capabilities that contribute to a vibrant social networking experience. Conclude by underlining the significance of a robust DBMS in supporting these features, enabling efficient data storage, retrieval, and organization to deliver a seamless user experience.

**1.2 Problem Statement**

Problem Statement:

The objective of this social networking site DBMS project is to design and implement a scalable and efficient database management system to support the functionalities of a social networking platform. The current challenge lies in managing and organizing the vast amount of user-generated data, ensuring smooth user interactions, and maintaining data integrity and security.

The existing social networking platforms struggle with issues such as slow response times, data inconsistencies, and difficulties in handling increasing user loads. The primary problem to address in this project is to develop a DBMS that can effectively store, retrieve, and manage user profiles, posts, comments, friend connections, and other relevant data. The system should be designed to handle a large number of concurrent users, while providing a seamless and personalized user experience.

Furthermore, the project needs to address privacy concerns and implement robust security measures to protect user data. The DBMS should have mechanisms to authenticate and authorize users, enforce privacy settings, and safeguard sensitive information. Additionally, the system should incorporate efficient algorithms for news feed generation, personalized recommendations, and content filtering, to enhance user engagement and satisfaction.

By addressing these challenges, the social networking site DBMS project aims to create a reliable and scalable platform that ensures fast and secure access to user data, enhances user interactions, and delivers an immersive and personalized social networking experience.

**1.3 Objectives**

The primary objective of the social networking site DBMS project is to develop a scalable and efficient database management system that addresses the challenges of managing and organizing a large volume of user-generated data. The project aims to ensure smooth user interactions by implementing fast data retrieval and storage mechanisms, optimizing query performance, and minimizing response times. Additionally, the objective is to enhance the user experience by incorporating personalized recommendations, news feed algorithms, and privacy controls.

Furthermore, the project seeks to prioritize data integrity and security. The objective is to design and implement robust mechanisms for data validation, backup and recovery, and user authentication. The DBMS should enforce strict privacy settings, encrypt sensitive user information, and detect and prevent unauthorized access attempts or malicious activities. By achieving these objectives, the social networking site DBMS project aims to create a reliable and secure platform that delivers a seamlessand engaging user experience while ensuring the privacy and integrity of user data.

# CHAPTER 2. DATABASE DESIGN

**2.1 List of Attributes, entities and relationship**

## Entity Name: Customer

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| cid | int(10) |
| cname | varchar(50) |
| cemail | varchar(100) |
| password | varchar(100) |
| Registration\_date | date |

## Entity Name: Profile

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| cid | int(10) |
| Full\_name | varchar(100) |
| date\_of\_birth | date |
| gender | varchar(10) |

## Entity Name: Friends

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| cid | int(10) |
| friend\_id | int |
| status | Varchar(20) |

## Entity Name: Followers

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| cid | int(10) |
| follower\_id | int(10) |

## Entity Name: Post

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| Post\_id | int(10) |
| cid | int(10) |
| post\_text | varchar(255) |
| Post\_date | date |

## Entity Name: Comments

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| Comment\_id | int(10) |
| post\_id | int |
| cid | Int(10) |
| content | varchar(500) |

## 

## Entity Name: PostLikes

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| like\_id | int(10) |
| Post\_id | int(10) |
| cid | int(10) |

## 

## Entity Name: CommentLikes

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| like\_id | int(10) |
| Comment\_id | int(10) |
| cid | Int(10) |

## Entity Name: Conversations

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| conversation\_id | int(10) |
| user1\_id | Int(10) |
| user2\_id | Int(10) |

## Entity Name: Messages

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| ***message\_id*** | int(10) |
| conversation\_id | int(10) |
| Sender\_id | Int(10) |
| Reciver\_id | Int(10) |
| content | varchar(1000) |

## Entity Name: Notifications

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| Notification\_id | int(10) |
| cid | Int(10) |
| Nitification\_text | text |
| notification\_date | date |

## Entity Name: Groups

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| Group\_id | int(10) |
| Group\_name | varchar(100) |
| Group\_description | text |

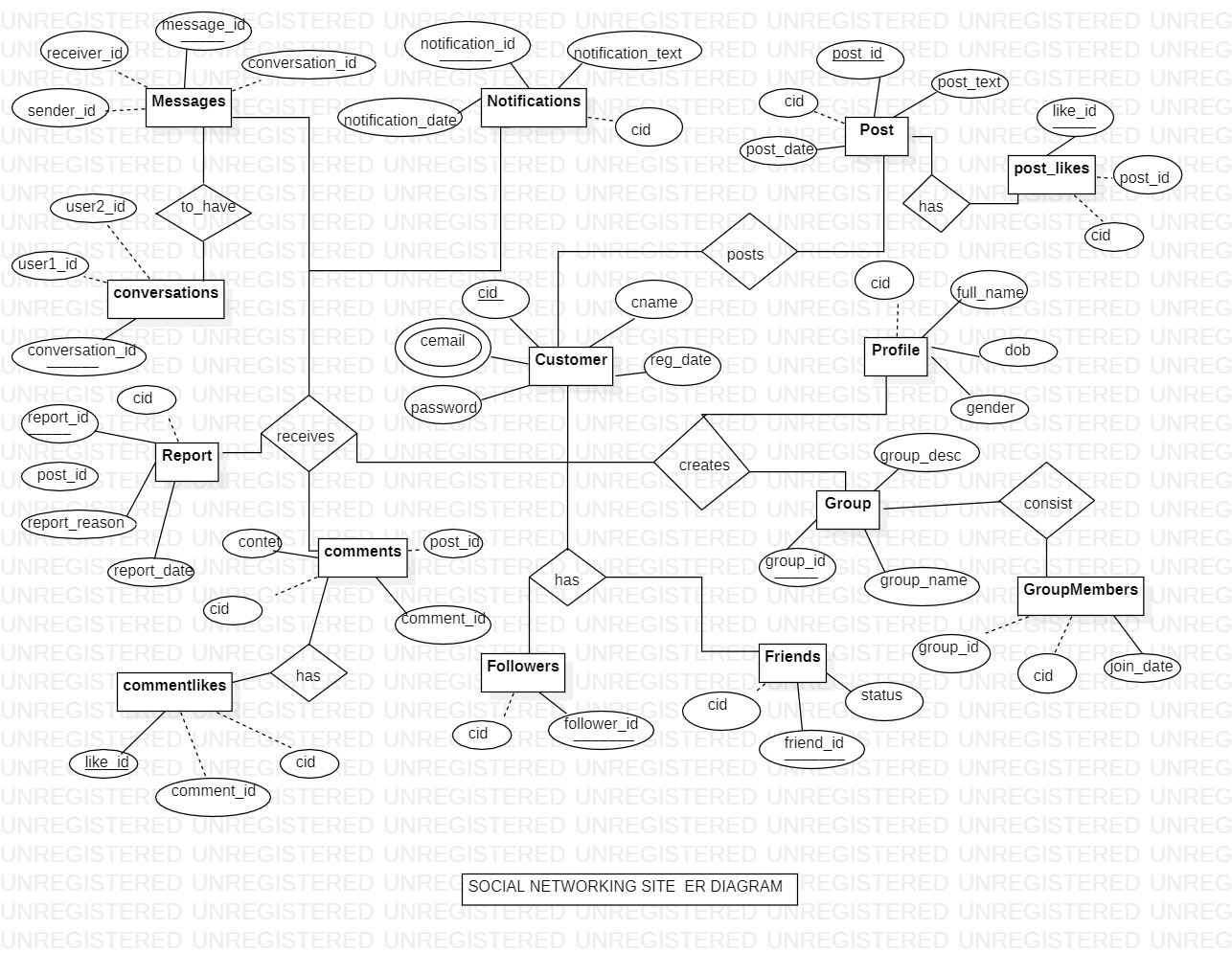
## Entity Name: GroupMembers

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| Group\_id | int(10) |
| cid | Int(10) |
| Join\_date | date |

# 14.Entity Name:Report

|  |  |
| --- | --- |
| **Attributes** | **Type** |
| Report\_id | Int(10) |
| cid | Int(10) |
| Post\_id | Int(10) |
| Report\_reason | text |
| Report\_date | date |

# 2.2 E-R Diagram



**CHAPTER 3. RELATIONAL MODEL**

# 3.1 Database languages

Four categories of database languages :

**1.** **Data definition language (DDL)**

Data definition language (DDL) creates the framework of the database by specifying the database schema, which is the structure that represents the organization of data. Its common uses include the creation and alteration of tables, files, indexes and columns within the database. This language also allows users to rename or drop the existing database or its components.

Here's a list of DDL statements:

• CREATE: Creates a new database or object, such as a table, index or column.

• ALTER: Changes the structure of the database or object.

• DROP: Deletes the database or existing objects.

• RENAME: Renames the database or existing objects.

**2.** **Data manipulation language (DML)**

Data manipulation language (DML) provides operations that handle user requests, offering a way to access and manipulate the data that users store within a database. Its common functions include inserting, updating and retrieving data from the database.

Here's a list of DML statements:

• INSERT: Adds new data to the existing database table.

• UPDATE: Changes or updates values in the table.

• DELETE: Removes records or rows from the table.

• SELECT: Retrieves data from the table or multiple tables.

**3. Data control language (DCL)**

Data control language (DCL) controls access to the data that users store within a database. Essentially, this language controls the rights and permissions of the database system. It allows users to grant or revoke privileges to the database.

Here's a list of DCL statements:

• GRANT: Gives a user access to the database.

• REVOKE: Removes a user's access to the database.

**4. Transaction control language (TCL)**

Transaction control language (TCL) manages the transactions within a database. Transactions group a set of related tasks into a single, executable task. All the tasks must succeed in order for the transaction to work. Here's a list of TCL statements:

• COMMIT: Carries out a transaction.

• ROLLBACK: Restores a transaction if any tasks fail to execute.

**3.2 Table Description**

Following are the tables along with constraints used in Social Networking Site database.

**1.Customer:** The "Customer" table stores customer information with ID, name, email, password, and registration date, utilizing integer, varchar, and date data types. The ID column is the primary key.

**Constraint:** The "Customer" table enforces constraints such as primary key for "cid" and non-null requirements for "cname", "cemail", "password", and "registration\_date" columns.

**2. Profile:** The "Profile" table stores additional information about customers such as full name, date of birth, and gender, with the customer ID (cid) serving as the primary key and a foreign key constraint referencing the "Customer" table.

**Constraint:** The "Profile" table includes a primary key constraint on "cid" and a foreign key constraint referencing "cid" in the "Customer" table, maintaining data integrity and establishing a relationship.

**3.Friends:** The "Friends" table represents the friendship relationship between customers using their IDs, with a composite primary key and foreign key constraints referencing the "Customer" table for data integrity.

**Constraint:** The "Friends" table enforces constraints including a composite primary key on "cid" and "friend\_id", and foreign key constraints referencing the "Customer" table for maintaining data integrity and establishing the friendship relationship.

**4.Followers:** The "Followers" table represents the followership relationship between customers using their IDs, with a composite primary key and foreign key constraints referencing the "Customer" table for data integrity.

**Constraint:** The "Followers" table enforces constraints including a composite primary key on "cid" and "follower\_id", and foreign key constraints referencing the "Customer" table for maintaining data integrity and establishing the followership relationship.

**5.Post:** The table "Post" stores posts with a unique identifier, category ID, text content, and date of creation.

**Constraint:** The "Post" table has constraints that enforce the primary key on the "post\_id" column, the non-null constraint on the "cid" column, the non-null constraint on the "post\_text" column, and the non-null constraint on the "post\_date" column.

**6.Comments:** The "Comments" table stores comments with a unique identifier, associated post ID, customer ID, content, and foreign key constraints referencing the "Post" and "Customer" tables.

**Constraint:** The "Comments" table has constraints that enforce the primary key on the "comment\_id" column and foreign key constraints on the "post\_id" and "cid" columns, referencing the "Post" and "Customer" tables, respectively.

**7.PostLikes:** **T**he "PostLikes" table stores likes on posts with a unique identifier, associated post ID, customer ID, and foreign key constraints referencing the "Post" and "Customer" tables.

**Constraint:** The "PostLikes" table has constraints that enforce the primary key on the "like\_id" column and foreign key constraints on the "post\_id" and "cid" columns, referencing the "Post" and "Customer" tables, respectively.

**8. CommentLikes:**The "CommentLikes" table stores likes on comments with a unique identifier, associated comment ID, customer ID, and foreign key constraints referencing the "Comments" and "Customer" tables.

**Constraints:** The "CommentLikes" table has constraints that enforce the primary key on the "like\_id" column and foreign key constraints on the "comment\_id" and "cid" columns, referencing the "Comments" and "Customer" tables, respectively.

**9. Conversations:**The "Conversations" table stores conversations with a unique identifier, and foreign key constraints referencing the "Customer" table for user1 and user2 IDs.

**Constraints:** The "Conversations" table has constraints that enforce the primary key on the "conversation\_id" column and foreign key constraints on the "user1\_id" and "user2\_id" columns, referencing the "Customer" table for customer IDs.

**10. Messages:** The "Messages" table stores messages with a unique identifier, associated conversation ID, sender ID, receiver ID, content, and foreign key constraints referencing the "Conversations" and "Customer" tables.

**Constraints:** The "Messages" table has constraints that enforce the primary key on the "message\_id" column and foreign key constraints on the "conversation\_id", "sender\_id", and "receiver\_id" columns, referencing the "Conversations" and "Customer" tables, respectively.

**11. Notifications:** The "Notifications" table stores notifications with a unique identifier, customer ID, notification text, notification date, and a foreign key constraint referencing the "Customer" table.

**Constraints:** The "Notifications" table has constraints that enforce the primary key on the "notification\_id" column and a foreign key constraint on the "cid" column, referencing the "Customer" table.

**12.Groups:** The "Groups" table stores groups with a unique identifier, group name, and group description.

**Constraints:** The "Groups" table has a primary key constraint on the "group\_id" column, ensuring its uniqueness.

**13.GroupMembers:** The "GroupMembers" table stores group membership information with group ID, customer ID, join date, and constraints including a composite primary key on group\_id and cid, and foreign key constraints referencing the "Groups" and "Customer" tables.

**Constraint:** The "GroupMembers" table has constraints that enforce a composite primary key on the "group\_id" and "cid" columns, as well as foreign key constraints on the "group\_id" and "cid" columns, referencing the "Groups" and "Customer" tables, respectively.

**14.Report:** The "Report" table stores reports with a unique identifier, customer ID, post ID, report reason, report date, and foreign key constraints referencing the "Customer" and "Posts" tables.

**Constraints:** The "Report" table has constraints that enforce the primary key on the "report\_id" column, and foreign key constraints on the "cid" and "post\_id" columns, referencing the "Customer" and "Posts" tables, respectively.

**3.3 Relational Database Scheme**

The relational database schema for Social Networking Site database is as follows:

1. Customer(cid,cname,cemail,password,registration\_date)
2. Profile(cid,full\_name,date\_of\_birth,gender)
3. Friends(cid,friend\_id,status)
4. Followers(cid,follower\_id)
5. Post(post\_id,cid,post\_text,post\_date)
6. Comments(comment\_id,cid,content)
7. PostLikes(like\_id,post\_id,cid)
8. CommentsLikes(like\_id,comment\_id,cid)
9. Conversations(conversation\_id,user1\_id,user2\_id)
10. Messages(message\_id,conversation\_id,sender\_id,receiver\_id,content)
11. Notifications(notification\_id,cid,notification\_text,notification\_date)
12. Groups(group\_id,group\_name,group\_description)
13. GroupMembers(group\_id,cid,join\_date)

Report(report\_id,cid,post\_id,repot\_date)

**3.4 Relational Queries**

/\* create a table customer\*/

CREATE TABLE Customer (

cid INT PRIMARY KEY,

cname VARCHAR(50) NOT NULL,

cemail VARCHAR(100) NOT NULL,

password VARCHAR(100) NOT NULL,

registration\_date DATE NOT NULL

);

INSERT INTO Customer

VALUES

(1,'swathi','swathi@gmail.com','s10101','2020-09-19' ),

(2,'samuel','samuel@gmail.com','sam0140','2021-12-06'),

(3, 'john\_doe', 'john@example.com', 'password123', '2023-01-19'),

(4, 'jane\_smith', 'jane@example.com', 'secret456', '2022-02-01'),

(5,'sid','sid@gmail.com','sid53','2020-08-30'),

select \* from customer

**Output:**

**sno** **cid cname cmobile cage Crating**

1 101 Swathi 8074942456 17 10

2 102 Samuel 6304634778 20 9

3 103 Sid Sriram 33333333 28 8

4 104 Anirudh 444444444 23 9

5 105 Jonita 555555555 24 8

6 106 Arman Malik 666666666 26 7

7 107 Arjith Singh 777777777 32 8

8 108 SPB 8888888888 21 9

9 109 Sunitha 99999999 23 9

10 110 DSP 8925674397 28 7

11 111 Chitra 1313131313 26 8

12 112 Ram Miriyala 1414141414 29 9

13 113 Anurag Kulakarni 1515151515 34 9

14 114 SS Thaman 7895461259 35 8

15 115 Dhvani Bhanushali 1717171717 32 8

16 116 Neha Kakkar 1818181818 37 9

/\*create a table Profile\*/

CREATE TABLE Profile (

cid INT PRIMARY KEY,

full\_name VARCHAR(100),

date\_of\_birth DATE,

gender VARCHAR(10),

FOREIGN KEY (cid) REFERENCES Customer(cid)

);

INSERT INTO Profile

VALUES

(1,'swathipr','2005-09-19','Female'),

(2,'samuelpr','2001-12-06','Male'),

(3, 'john\_doe','2003-01-19','Male'),

(4, 'jane\_smith', '1996-02-01','Male'),

(5,'sid sriram','1986-08-30','Male'),

(6,'anirudh ravichander','1999-02-24','Male');

select \* from Profile

**Output:**

**cid full\_name date\_of\_birth gender**

1 swathipr 2005-09-19 Female

2 samuelpr 2001-12-06 Male

3 john\_doe 2003-01-19 Male

4 jane\_smith 1996-02-01 Male

5 sid sriram 1986-08-30 Male

6 anirudh ravichander 1999-02-24 Male

7 sunny komarla 1982-04-18 Male

8 manu komarla 2002-01-05 Male

9 bhuvana sura 2001-03-24 Female

10 siri sivapuram 1996-09-17 Female

11 mahi komarla 1986-09-14 Female

12 anjalipr 1997-09-26 Female

13 rayudupr 1994-05-16 Male

14 ms dhoni 1989-03-26 Male

15 ravindra jadeja 1995-07-20 Male

16 rohit sharma 1989-05-31 Male

/\*create a table Friends\*/

CREATE TABLE Friends (

cid INT,

friend\_id INT,

status VARCHAR(20) NOT NULL,

PRIMARY KEY (cid, friend\_id),

FOREIGN KEY (cid) REFERENCES Customer(cid),

FOREIGN KEY (friend\_id) REFERENCES Customer(cid),

);

INSERT INTO Friends (cid, friend\_id, status)

VALUES

(1, 2, 'Accepted'),

(2, 1, 'Pending'),

(3, 4, 'Accepted'),

(4, 3, 'Accepted'),

(5, 6, 'Pending'),

Select \* from friends

**Output:**

**cid friend\_id status**

1 2 Accepted

2 1 Pending

3 4 Accepted

4 3 Accepted

5 6 Pending

6 5 Accepted

7 8 Accepted

8 7 Accepted

9 10 Accepted

10 9 Accepted

11 12 Accepted

12 11 Accepted

13 14 Pending

/\* create a table Followers\*/

CREATE TABLE Followers (

cid INT,

follower\_id INT,

PRIMARY KEY (cid, follower\_id),

FOREIGN KEY (cid) REFERENCES Customer(cid),

FOREIGN KEY (follower\_id) REFERENCES Customer(cid)

);

INSERT INTO Followers (cid, follower\_id)

VALUES

(41, 2),

(22, 1),

(33, 5),

(14, 1),

(35, 2),

Select \* from followers

**Output:**

**cid follower\_id**

1 20

2 21

2 30

3 29

3 41

4 15

5 36

6 17

7 28

8 6

8 19

9 7

9 10

10 8

10 11

10 38

11 12

12 23

12 28

/\* create a table Post\*/

CREATE TABLE Post(

post\_id int PRIMARY KEY,

cid INT NOT NULL,

post\_text VARCHAR(255) NOT NULL,

post\_date DATE NOT NULL

);

insert into Post values

(101,19,'Hello','2023-03-29'),

(110, 10, 'Hello World!', '2023-01-01'),

(121, 12, 'This is a test.', '2023-01-02'),

(102, 14, 'Hello world!', '2023-05-18 '),

(103, 27, 'Having a great day!', '2023-05-18 '),

(133, 23, 'Check out this amazing photo!', '2023-05-18'),

(104,11, 'Excited for the weekend!', '2023-05-18'),

(105, 34, 'Just finished a challenging project!', '2023-05-19 '),

Select \* from post

**Output:**

post\_id cid post\_text post\_date

101 19 Hello 2023-03-29

102 14 Hello world! 2023-05-18

103 27 Having a great day! 2023-05-18

104 11 Excited for the weekend! 2023-05-18

105 34 Just finished a challenging project! 2023-05-19

106 20 What's your favorite movie? 2023-05-19

107 13 Looking for recommendations new books to read. 2023-05-19

108 40 Feeling motivated today! 2023-05-19

109 12 Just visited an amazing art exhibition! 2023-05-19

110 10 Hello World! 2023-01-01

111 22 Happy birthday to my dear friend! 2023-05-19

112 41 After long Time!! 2021-09-19

/\*create table Comments\*/

CREATE TABLE Comments (

comment\_id INT PRIMARY KEY,

post\_id INT,

cid INT,

content VARCHAR(500),

FOREIGN KEY (post\_id) REFERENCES Post(post\_id),

FOREIGN KEY (cid) REFERENCES Customer(cid)

);

INSERT INTO Comments

VALUES (201, 101, 1, 'Super Post!'),

(202, 101, 2, 'Nice post!'),

(203, 102, 3, 'I have a question regarding this post.'),

(204, 102, 1, 'Thanks for sharing!'),

(205, 103, 3, 'Great job!'),

(206, 104, 2, 'Interesting topic.'),

select \* from comments

**Output:**

**comment\_id post\_id cid content**

201 101 1 Super Post!

202 101 2 Nice post!

203 102 3 I have a question regarding this post.

204 102 1 Thanks for sharing!

205 103 3 Great job!

206 104 2 Interesting topic.

207 104 1 I completely agree.

208 105 3 This post helped me a lot.

209 106 2 Looking forward to more content like this.

/\*create table PostLikes\*/

CREATE TABLE PostLikes (

like\_id INT PRIMARY KEY,

post\_id INT,

cid INT,

FOREIGN KEY (post\_id) REFERENCES Post(post\_id),

FOREIGN KEY (cid) REFERENCES Customer(cid)

);

INSERT INTO PostLikes

VALUES (301, 101, 1),

(302, 101, 2),

(303, 102, 3),

(304, 102, 1),

(305, 103, 3),

(306, 104, 2),

Select \* from postlikes

**OUTPUT:**

**like\_id post\_id cid**

301 101 1

302 101 2

303 102 3

304 102 1

305 103 3

306 104 2

307 104 1

308 105 3

309 106 2

310 107 1

311 107 3

312 108 2

313 109 1

314 109 3

315 110 2

316 111 1

317 112 3

319 103 1

320 109 3

/\*create table commentlikes

CREATE TABLE CommentLikes (

like\_id INT PRIMARY KEY,

comment\_id INT,

cid INT,

FOREIGN KEY (comment\_id) REFERENCES Comments(comment\_id),

FOREIGN KEY (cid) REFERENCES Customer(cid)

);

INSERT INTO CommentLikes

VALUES (301, 201, 1),

(302, 201, 2),

(303, 202, 3),

(304, 202, 1),

(305, 203, 3),

(306, 204, 2),

select \* from commentlikes

**Output:**

**like\_id comment\_id cid**

301 201 1

302 201 2

303 202 3

304 202 1

305 203 3

306 204 2

307 204 1

308 205 3

309 206 2

/\*create table conversations\*/

CREATE TABLE Conversations (

conversation\_id INT PRIMARY KEY,

user1\_id INT,

user2\_id INT,

FOREIGN KEY (user1\_id) REFERENCES Customer(cid),

FOREIGN KEY (user2\_id) REFERENCES Customer(cid)

);

INSERT INTO Conversations

VALUES (401, 1, 2),

(402, 1, 3),

(403, 1, 4),

(404, 1, 5),

(405, 1, 6),

(406, 1, 7),

Select \* from conversations

**OUTPUT:**

**conversation\_id user1\_id user2\_id**

401 1 2

402 1 3

403 1 4

404 1 5

405 1 6

406 1 7

407 1 8

408 1 9

409 1 10

/\*create table Messages\*/

CREATE TABLE Messages (

message\_id INT PRIMARY KEY,

conversation\_id INT,

sender\_id INT,

receiver\_id INT,

content VARCHAR(1000),

FOREIGN KEY (conversat2ion\_id) REFERENCES Conversations(conversation\_id),

FOREIGN KEY (sender\_id) REFERENCES Customer(cid),

FOREIGN KEY (receiver\_id) REFERENCES Customer(cid)

);

INSERT INTO Messages

VALUES (1, 401, 1, 20, 'Hello, how are you?'),

(2, 402, 2, 11, ' Iam doing well, thank you! How about you?'),

(3, 403, 31, 32, ' Iam doing great too! Just enjoying the day.'),

(4, 404, 29, 41, 'That good to hear! Anything interesting happening?'),

(5, 405, 18, 22, 'Not much, just catching up on some work.'),

Select \* from messages

**OUTPUT:**

**message\_id conversation\_id sender\_id receiver\_id content**

1 401 1 20 Hello, how are you?

2 402 2 11 Iam doing well, thank you! How about you?

3 403 31 32 Iam doing great too! Just enjoying the day.

4 404 29 41 That good to hear! Anything interesting happening?

5 405 18 22 Not much, just catching up on some work.

6 406 21 17 Soundsproductive! Iamplanning go for hike

7 407 12 32 That sounds like a great plan! Enjoy your hike.

8 408 20 19 Thanks! I will share some pictures with you afterward.

9 409 11 2 I will be looking forward to it!

/\*create table Notifications\*/

CREATE TABLE Notifications (

notification\_id INT PRIMARY KEY,

cid INT,

notification\_text TEXT,

notification\_date date not null,

FOREIGN KEY (cid) REFERENCES Customer(cid)

);

INSERT INTO Notifications

VALUES

(601, 1, 'You have a new message.', '2023-05-18'),

(602, 2, 'Your post has been liked.', '2023-02-18'),

(603, 3, 'Someone commented on your post.', '2023-05-18'),

(604, 4, 'Your friend request has been accepted.', '2023-04-11'),

(605, 5, 'You have been mentioned in a post.', '2023-01-28'),

Select \* from Notifications

**OUTPUT:**

**notification\_id cid notification\_text notification\_date**

601 1 You have a new message. 2023-05-18

603 3 Someone commented on your post. 2023-05-18

604 4 Your friend request has been accepted. 2023-04-11

605 5 You have been mentioned in a post. 2023-01-28

606 6 A new event has been created. 2022-05-30

607 7 You have a new friend suggestion. 2021-01-17

608 8 Your post has been shared. 2023-01-06

609 9 You have received a new friend request. 2022-09-19

610 10 Your profile has been updated. 2023-01-06

/\*create table group\*/

CREATE TABLE Groups (

group\_id INT PRIMARY KEY,

group\_name VARCHAR(100),

group\_description TEXT

);

INSERT INTO Groups (group\_id, group\_name, group\_description)

VALUES

(801, 'Sports Enthusiasts', 'A group for passionate sports fans'),

(802, 'Food Lovers', 'Exploring the world of culinary delights'),

(803, 'Travel Junkies', 'Sharing travel stories and tips'),

(804, 'Tech Geeks', 'Discussing the latest technology trends'),

(805, 'Bookworms United', 'For avid readers and book lovers'),

Select \* from groups

**OUTPUT:**

**group\_id group\_name group\_description**

801 Sports Enthusiasts A group for passionate sports fans

802 Food Lovers Exploring the world of culinary delights

803 Travel Junkies Sharing travel stories and tips

804 Tech Geeks Discussing the latest technology trends

805 Bookworms United For avid readers and book lovers

806 Fitness Freaks Motivating each othertowards healthy lifestyle

807 Music Addicts All about music and sharing favorite tunes

808 Photography Enthusiasts Capturing the world through lenses

809 Fashionistas Fashion trends, styles, and inspirations

/\*create table GroupMembers\*/

CREATE TABLE GroupMembers (

group\_id INT,

cid INT,

join\_date DATE,

PRIMARY KEY (group\_id, cid),

FOREIGN KEY (group\_id) REFERENCES Groups(group\_id),

FOREIGN KEY (cid) REFERENCES Customer(cid)

);

INSERT INTO GroupMembers (group\_id, cid, join\_date) VALUES

(801, 1, '2020-01-01'),

(802, 2, '2022-02-02'),

(803, 7, '2023-01-07'),

(804, 10, '2023-01-10'),

(805, 5, '2020-07-05'),

Select \* from GroupMembers

**OUTPUT:**

**group\_id cid join\_date**

801 1 2020-01-01

802 2 2022-02-02

803 7 2023-01-07

804 10 2023-01-10

805 5 2020-07-05

806 6 2021-06-06

807 7 2021-11-10

808 8 2023-12-30

809 9 2022-01-01

810 10 2020-02-21

/\*create table Report\*/

CREATE TABLE Report(

report\_id INT PRIMARY KEY,

cid INT,

post\_id INT,

report\_reason TEXT,

report\_date date not null,

FOREIGN KEY (cid) REFERENCES Customer(cid),

FOREIGN KEY (post\_id) REFERENCES Posts(post\_id)

);

INSERT INTO Report(report\_id, cid, post\_id, report\_reason, report\_date) VALUES

(701, 41, 134, 'Inappropriate content', '2023-05-18'),

(702, 2, 135, 'Spamming', '2022-03-18'),

(703, 3, 136, 'Harassment', '2021-04-28'),

(704, 4, 137, 'Fake account', '2020-02-08'),

(705, 5, 138, 'Violation of terms', '2023-03-30'),

Select \* from Report

**OUTPUT:**

**report\_id cid post\_id report\_reason report\_date**

701 41 134 Inappropriate content 2023-05-18

702 2 135 Spamming 2022-03-18

703 3 136 Harassment 2021-04-28

704 4 137 Fake account 2020-02-08

705 5 138 Violation of terms 2023-03-30

706 6 139 Cyberbullying 2022-07-06

707 7 140 Impersonation 2021-08-27

708 8 141 Privacy infringement 2020-09-19

709 9 142 Offensive content 2023-01-20

**SQL QUERIES:**

**1.Find out the customers who are the most popular users in the database?**

**Command:**

SELECT cid, COUNT(\*) AS total\_likes

FROM PostLikes

GROUP BY cid

ORDER BY total\_likes DESC;

**Output:**

**cid total\_likes**

1 7

3 7

2 5

**2.Update customer relation by adding age as an attribute and data type as int with null constraint.**

**Command:**

alter table customer

add cage int null;

**Output:**

**cid cname cemail registration\_date cage**

1 swathi swathi@gmail.com 2020-09-19 17

2 samuel samuel@gmail.com 2021-12-06 21

3 john\_doe john@example.com 2023-01-19 32

4 jane\_smith jane@example.com 2022-02-01 26

5 sid sid@gmail.com 2020-08-30 20

6 anirudh anirudh@gmail.com 2021-02-24 29

7 sunny sunny@gmail.com 2019-04-18 39

8 manu manu@gmail.com 2016-01-05 41

9 bhuvana bhuvana@gmail.com 2014-03-24 32

10 siri siri@gmail.com 2014-09-17 26

11 mahi mahi@gmail.com 2016-09-14 37

12 anjali anjali@gmail.com 2023-09-26 22

13 rayudu rayudu@gmail.com 2019-05-16 30

14 dhoni dhoni@gmail.com 2017-03-26 45

15 jadeja jadeja@gmail.com 2019-07-20 23

16 rohit rohit@gmail.com 2018-05-31 42

17 hardik hardik@gmail.com 2021-03-01 NULL

18 faf faf@gmail.com 2023-06-14 NULL

19 gill gill@gmail.com 2022-09-19 NULL

**3.Find out the average age of the customer?**

**Command:**

SELECT AVG(cage) AS average\_age

FROM Customer;

**Output:**

**average\_age**

30

**4.How many users are there in the database?**

**Command:**

SELECT COUNT(\*) AS total\_customer FROM Customer;

**Output:**

**total\_customer**

41

**5.Find out the 5 customers who are the most active users in the database?**

**Command:**

SELECT top(5) cid, COUNT(\*) AS total\_posts

FROM Post

GROUP BY cid

ORDER BY total\_posts DESC;

**Output:**

**cid total\_posts**

12 2

22 2

23 2

27 2

34 2

**6.Find posts of customer where cid=19,10 and 27.**

**Command:**

SELECT \* FROM post WHERE cid IN (19, 10,27);

**Output:**

**post\_id cid post\_text post\_date**

101 19 Hello 2023-03-29

103 27 Having a great day! 2023-05-18

110 10 Hello World! 2023-01-01

140 27 Exploring new places... 2021-01-20

**7.Find full name ,gender,date of birth of a customer where cid=1,2 and 19.**

**Command:**

select full\_name,gender,date\_of\_birth from profile where cid in(1,2,19)

**Output:**

**full\_name gender date\_of\_birth**

swathipr Female 2005-09-19

samuelpr Male 2001-12-06

shubmangill Male 2000-09-19

**8.Find out the total number of customers who registred last year.**

**Command:**

Select count(\*) as registred \_lastyear from customer where

registration\_date>=DATEADD(year,-1,GETDATE())

AND registration\_date<GETDATE();

**Output:**

**registred\_lastyear**

**5**

**9.Create a temporary table for sender\_id .**

**Command:**

select \* into Chat from Messages where sender\_id=21

select \* from Chat

**Output:**

**message\_id conversation\_id sender\_id receiver\_id content**

6 406 21 17 Sounds productive!

13 413 21 23 Iam doing well too.

**10.Create a view that contains list of content where comment\_id is 217.**

**Command:**

create view review as(

select \* from comments where comment\_id=217)

select \* from review

**Output:**

**comment\_id post\_id cid content**

217 112 3 I enjoyed reading this.

**11.Find out the status,cid and friend\_id where friend\_id not in 14,20,8,15.**

**Command:**

select cid,status,friend\_id from friends where friend\_id not in (14,20,8,15)

**Output:**

**cid status friend\_id**

1 Accepted 2

2 Pending 1

3 Accepted 4

4 Accepted 3

5 Pending 6

6 Accepted 5

8 Accepted 7

9 Accepted 10

10 Accepted 9

11 Accepted 12

12 Accepted 11

14 Accepted 13

15 Accepted 16

17 Accepted 18

18 Accepted 17

20 Accepted 19

**12. Create a simple relation that contains report id integer and primary key,cid int foreign key,report date date with not null constraint,report reason reason as text data type and post id as foreign key make this relation as Report.**

**Command:**

CREATE TABLE Report(

report\_id INT PRIMARY KEY,

cid INT,

post\_id INT,

report\_reason TEXT,

report\_date date not null,

FOREIGN KEY (cid) REFERENCES Customer(cid),

FOREIGN KEY (post\_id) REFERENCES Posts(post\_id)

);

select \* from Report

**Output:**

**report\_id cid post\_id report\_reason report\_date**

**13.Insert the records in Report relation.**

**Command:**

INSERT INTO Report

VALUES

(701, 41, 134, 'Inappropriate content', '2023-05-18'),

(702, 2, 135, 'Spamming', '2022-03-18'),

(703, 3, 136, 'Harassment', '2021-04-28'),

(704, 4, 137, 'Fake account', '2020-02-08'),

(705, 5, 138, 'Violation of terms', '2023-03-30'),

**Output:**

**report\_id cid post\_id report\_reason report\_date**

701 41 134 Inappropriate content 2023-05-18

702 2 135 Spamming 2022-03-18

703 3 136 Harassment 2021-04-28

704 4 137 Fake account 2020-02-08

705 5 138 Violation of terms 2023-03-30

706 6 139 Cyberbullying 2022-07-06

707 7 140 Impersonation 2021-08-27

708 8 141 Privacy infringement 2020-09-19

709 9 142 Offensive content 2023-01-20

**15.Find the customer who have zero post likes.**

**Command:**

SELECT Cust.cid, Cust.cname

FROM Customer Cust

LEFT JOIN Posts P ON Cust.cid = P.cid

LEFT JOIN PostLikes PL ON P.post\_id = PL.post\_id

GROUP BY Cust.cid, Cust.cname

HAVING COUNT(PL.like\_id) = 0;

**Output:**

**cid cname**

19 gill

20 rahul

21 krunal

22 jaiswal

23 virat

24 chahhal

25 russel

26 DK

27 chahhar

28 pollard

29 smriti

30 harmanpreet

31 chitra

**16.Find the average commentlikes who cid is 3.**

**Command:**

SELECT AVG(like\_id) AS avg\_comment\_likes

FROM CommentLikes cl

JOIN Customer cu ON cl.cid = cu.cid

WHERE cl.cid = 3;

**Output:**

**avg\_comment\_likes**

311

**17.Find the details of customer age between 20 and 30.**

**Command:**

SELECT \* FROM Customer WHERE cage BETWEEN 20 AND 30

**Output:**

**cid cname cemail password registration\_date cage**

2 samuel samuel@gmail.com sam0140 2021-12-06 21

4 jane\_smith jane@example.com secret456 2022-02-01 26

5 sid sid@gmail.com sid53 2020-08-30 20

6 anirudh anirudh@gmail.com anirudh25 2021-02-24 29

10 siri siri@gmail.com siri57 2014-09-17 26

**18.find out the total number of reports are there in report table.**

**Command:**

SELECT COUNT(\*) AS total\_reports

FROM Report;

**Output:**

**total\_reports**

20

**19.Find the like\_id of commentlikes where commentid=205.**

**Command:**

SELECT like\_id FROM commentlikes WHERE comment\_id=205

**Output:**

**like\_id**

308

**20.Find the total number of receivers that are in messages.**

**Command:**

SELECT COUNT(DISTINCT receiver\_id) as receivers FROM messages

**Output:**

**receivers**

16

**21.Update the notification of a particular customer.**

**Command:**

UPDATE notifications

SET notification\_id = '608'

WHERE cid = '05'

select \* from notifications

**Output:**

**notification\_id cid notification\_text notification\_date**

601 1 You have a new message. 2023-05-18

603 3 Someone commented on your post. 2023-05-18

604 4 Your friend request has been accepted. 2023-04-11

608 5 You have been mentioned in a post. 2023-01-28

606 6 A new event has been created. 2022-05-30

607 7 You have a new friend suggestion. 2021-01-17

608 8 Your post has been shared. 2023-01-06

609 9 You have received a new friend request. 2022-09-19

610 10 Your profile has been updated. 2023-01-06

611 11 You have a new follower. 2022-10-18

613 13 Your event invitation has been accepted. 2022-07-06

614 14 You have a new notification. 2020-09-08

**22.Find the name of the group whose name starts with S and having atleast five Characters.**

**Command:**

SELECT group\_name FROM groups WHERE group\_name LIKE 'T\_%'

**Output:**

**group\_name**

Travel Junkies

Tech Geeks

**23.Create a view from GroupMembers which include group\_id,cid.**

**Command:**

create view group\_members as

SELECT group\_id,cid

FROM GroupMembers

select \* from group\_members

**Output:**

**group\_id cid**

801 1

802 2

803 7

804 10

805 5

806 6

807 7

808 8

809 9

810 10

811 11

812 12

813 13

814 14

815 15

816 16

817 17

818 18

819 19

820 20

**24.Display the top five report\_id of reports.ind**

**Command:**

select top(5) report\_id from report

**Output:**

**report\_id**

701

702

703

704

705

**25.Find the report\_reason of the customer who's cid=16.**

**Command:**

SELECT report\_reason

FROM Report

WHERE cid = 16;

**Output:**

**report\_reason**

Unauthorized access

**26.Create a temporary table for gender and for 5 customers.**

**Command:**

select top (5) \* into gender from Profile where gender='Male'

select \* from gender

**Output:**

**cid full\_name date\_of\_birth gender**

2 samuelpr 2001-12-06 Male

3 john\_doe 2003-01-19 Male

4 jane\_smith 1996-02-01 Male

5 sid sriram 1986-08-30 Male

6 anirudh ravichander 1999-02-24 Male

**27.Find the total number of posts poste by the age group is 20 & 30 .**

**Command:**

SELECT COUNT(DISTINCT post\_id) AS total\_posts

FROM posts p

JOIN customer c ON p.cid = c.cid

WHERE cage IN (20, 30);

**Output:**

**total\_posts**

23

**28.List all customers registered on or after 2022-01-01.**

**Command:**

SELECT \* FROM Customer WHERE registration\_date >= '2022-01-01';

**Output:**

**cid cname cemail password registration\_date cage**

3 john\_doe john@example.com pasword123 2023-01-19 32

4 jane\_smith jane@example.com secret456 2022-02-01 26

12 anjali anjali@gmail.com anjali547 2023-09-26 22

18 faf faf@gmail.com faf89 2023-06-14 NULL

19 gill gill@gmail.com gill09 2022-09-19 NULL

22 jaiswal jaiswal@gmail.com jaiswal64 2023-04-16 NULL

31 chitra chitra@gmail.com chitra93 2023-04-15 NULL

33 Alice Johnson alice@gmail.com alice67 2022-03-12 NULL

41 Alex Turner alex@gmail.com turner30 2023-02-14 NULL

**29. Find the customers who registered before 2017-01-01 and have the letter 'a' in their email.**

**Command:**

SELECT \* FROM Customer WHERE registration\_date < '2017-01-01' AND cemail LIKE '%a%';

**Output:**

**cid cname cemail password registration\_date cage**

8 manu manu@gmail.com manu36 2016-01-05 41

9 bhuvana bhuvana@gmail.com bhuvana68 2014-03-24 32

10 siri siri@gmail.com siri57 2014-09-17 26

11 mahi mahi@gmail.com mahi59 2016-09-14 37

26 DK dk@gmail.com dk36 2016-03-28 NULL

28 pollard pollard2gmail.com pollard69 2016-05-16 NULL

34 Anderson anderson@gmail.com bob29 1988-11-05 NULL

40 John Doe john@gmail.com doe96 2016-05-15 NULL

**30. Find the customers whose names start with the letter 's' and have registered in the year 2020.**

**Command:**

SELECT \* FROM Customer WHERE cname LIKE 's%' AND YEAR(registration\_date) = 2020;

**Output:**

**cid cname cemail password registration\_date cage**

1 swathi swathi@gmail.com s10101 2020-09-19 17

5 sid sid@gmail.com sid53 2020-08-30 20

**31. List the top 5 customers and their followers in alphabetical order of their names.**

**Command:**

SELECT top (5)c.cname, f.follower\_id

FROM Customer c

JOIN Followers f ON c.cid = f.cid

ORDER BY c.cname;

**Output:**

**cname follower\_id**

Alex Turner 2

Alex Turner 27

Alice Johnson 5

Alice Johnson 31

anirudh 17

**32. List the top 10 customers and their post counts in descending order of post counts.**

**Command:**

SELECT top(10) c.cid, c.cname, COUNT(p.post\_id) AS post\_count

FROM Customer c

LEFT JOIN Post p ON c.cid = p.cid

GROUP BY c.cid, c.cname

ORDER BY post\_count DESC;

**Output:**

**cid cname post\_count**

12 anjali 2

22 jaiswal 2

23 virat 2

27 chahhar 2

34 Bob Anderson 2

40 John Doe 2

11 mahi 1

10 siri 1

3 john\_doe 1

13 rayudu 1

**33.Find the customers who have a friend named 'Swathi'.**

**Command:**

SELECT c.\*

FROM Customer c

JOIN Friends f ON c.cid = f.cid

JOIN Customer friend ON f.friend\_id = friend.cid

WHERE friend.cname = 'Swathi';

**Output:**

**cid cname cemail password registration\_date cage**

2 samuel samuel@gmail.com sam0140 2021-12-06 21

**34.How many male customers are registered on the social media site?**

**Command:**

SELECT COUNT(\*) as Male\_Count FROM Profile WHERE gender = 'Male';

**Output:**

**Male\_Count**

30

**35.List the customers who have a pending friend request.**

**Command:**

SELECT Customer.cid, Customer.cname FROM Customer

INNER JOIN Friends ON Customer.cid = Friends.cid

WHERE Friends.status = 'Pending';

**Output:**

**cid cname**

2 samuel

5 sid

13 rayudu

16 rohit

**36.List the customers who registered before the year 2015.**

**Command:**

SELECT c.cname FROM Customer c WHERE YEAR(c.registration\_date) < 2015;

**Output:**

**cname**

bhuvana

siri

Bob Anderson

**37.List the five customers who have posted on the social media site after the year 2022.**

**Command:**

SELECT top(5) c.cname FROM Customer c JOIN Post p ON c.cid = p.cid WHERE YEAR(p.post\_date) > 2022;

**Output:**

**cname**

gill

dhoni

chahhar

mahi

Bob Anderson

**38. How many customers have both friends and followers?**

**Command:**

SELECT COUNT(\*) as Common\_Friends FROM (SELECT DISTINCT cid FROM Friends)

AS friends\_list JOIN (SELECT DISTINCT cid FROM Followers)

AS followers\_list ON friends\_list.cid = followers\_list.cid;

**Output:**

**Common\_Friends**

20

**39. List the customers who have the same full name in their profile.**

**Command:**

SELECT p.full\_name FROM Profile p GROUP BY p.full\_name HAVING COUNT(\*) > 1;

**Output:**

**full\_name**

**40.What is the most recent registration date?**

**Command:**

SELECT MAX(registration\_date) FROM Customer;

**Output:**

**(No column name)**

2023-09-26

**41.List the 10 customers who have accepted friend requests.**

**Command:**

SELECT top(10) Customer.cid, Customer.cname FROM Customer

INNER JOIN Friends ON Customer.cid = Friends.cid

WHERE Friends.status = 'Accepted';

**Output:**

**cid cname**

1 swathi

3 john\_doe

4 jane\_smith

6 anirudh

7 sunny

8 manu

9 bhuvana

10 siri

11 mahi

12 anjali

**42.List the customers who have made a post on a specific date, say '2023-05-18'.**

**Command:**

SELECT DISTINCT cid FROM Post WHERE post\_date = '2023-05-18';

**Output:**

**cid**

11

14

21

23

27

**43.List the customers who have not made any posts.**

**Command:**

SELECT cid,cname FROM Customer WHERE cid NOT IN (SELECT DISTINCT cid FROM Post);

**Output:**

**cid cname**

1 swathi

2 samuel

4 jane\_smith

5 sid

6 anirudh

7 sunny

8 manu

9 bhuvana

15 jadeja

16 rohit

17 hardik

18 faf

**44.List the customers whose names start with the letter 'S'.**

**Command:**

SELECT \* FROM Customer WHERE cname LIKE 'S%';

**Output:**

**cid cname cemail password registration\_date cage**

1 swathi swathi@gmail.com s10101 2020-09-19 17

2 samuel samuel@gmail.com sam0140 2021-12-06 21

5 sid sid@gmail.com sid53 2020-08-30 20

7 sunny sunny@gmail.com sunny54 2019-04-18 39

10 siri siri@gmail.com siri57 2014-09-17 26

29 smriti smriti@gmail.com smriti47 2021-01-10 NULL

36 Sam Wilson sam@gmail.com sam888 2021-09-02 NULL

39 Sara Miller miller@gmail.com sara32 2017-06-29 NULL

**45.List the customers whose names end with the letter 'l'.**

**Command:**

SELECT \* FROM Customer WHERE cname LIKE '%l';

**Output:**

**cid cname cemail password registration\_date cage**

2 samuel samuel@gmail.com sam0140 2021-12-06 21

19 gill gill@gmail.com gill09 2022-09-19 NULL

20 rahul rahul@gmail.com rahul23 2018-04-23 NULL

21 krunal krunal@gail.com krunal86 2019-06-17 NULL

22 jaiswal jaiswal@gmail.com jaiswal64 2023-04-16 NULL

24 chahhal chahhal@gmail.com chahhal14 2019-02-12 NULL

25 russel russel@gmail.com russel41 2017-08-30 NULL

**46.List the customers in the order of their registration date, starting from the latest.**

**Command:**

SELECT \* FROM Customer ORDER BY registration\_date DESC;

**Output:**

**cid cname cemail password registration\_date cage**

12 anjali anjali@gmail.com anjali547 2023-09-26 22

18 faf faf@gmail.com faf89 2023-06-14 NULL

22 jaiswal jaiswal@gmail.com jaiswal64 2023-04-16 NULL

31 chitra chitra@gmail.com chitra93 2023-04-15 NULL

41 Alex Turner alex@gmail.com turner30 2023-02-14 NULL

3 john\_doe john@example.com password123 2023-01-19 32

19 gill gill@gmail.com gill09 2022-09-19 NULL

33 Alice Johnson alice@gmail.com alice67 2022-03-12 NULL

4 jane\_smith jane@example.com secret456 2022-02-01 26

2 samuel samuel@gmail.com sam0140 2021-12-06 21

**47.List the customers who have the highest registration date.**

**Command:**

SELECT \* FROM Customer WHERE registration\_date = (SELECT MAX(registration\_date) FROM Customer);

**Output:**

**cid cname cemail password registration\_date cage**

12 anjali anjali@gmail.com anjali547 2023-09-26 22

**48.List the customer ID, customer name, and date of birth of customers who were born in the month is 9.**

**Command:**

SELECT c.cid, c.cname, p.date\_of\_birth

FROM Customer c JOIN Profile p ON c.cid = p.cid

WHERE MONTH(p.date\_of\_birth) = 9;

**Output:**

**cid cname date\_of\_birth**

1 swathi 2005-09-19

10 siri 1996-09-17

11 mahi 1986-09-14

12 anjali 1997-09-26

19 gill 2000-09-19

32 Jane Smith 1995-09-20

36 Sam Wilson 1991-09-02

**49.Find the top 5 customers with their cid, cname, and the count of their friends, including customers who may not have any friends, sorted by the friend count in descending order."**

**Command:**

SELECT top(5) c.cid, c.cname, COUNT(f.friend\_id) AS friend\_count

FROM Customer c LEFT JOIN Friends f ON c.cid = f.cid

GROUP BY c.cid, c.cname

**Output:**

**cid cname friend\_count**

1 swathi 1

2 samuel 1

3 john\_doe 1

4 jane\_smith 1

5 sid 1

**50.Find the top 11 customers with their customer ID, customer name, and the count of their posts, sorted by the post count in descending order."**

**Command:**

SELECT top(11) c.cid, c.cname, COUNT(\*) AS post\_count

FROM Customer c JOIN Post p ON c.cid = p.cid

GROUP BY c.cid, c.cname;

**Output:**

**cid cname post\_count**

3 john\_doe 1

10 siri 1

11 mahi 1

12 anjali 2

13 rayudu 1

14 dhoni 1

19 gill 1

20 rahul 1

21 kruna 1

22 jaiswal 2

23 virat 2

**CHAPTER 4. CONCLUSION AND FUTUREWORK**

**4.1 Conclusion**

In conclusion, the development of a social networking site using a database management system (DBMS) is a complex and challenging project. Throughout the project, various aspects of database design, implementation, and management were addressed to create a robust and efficient system.

The project aimed to provide users with a platform to connect and interact with others, share content, and engage in social activities.

**Key components of the project included:**

**1. Database Design:** A well-structured database schema was designed, taking into account the various entities, attributes, and relationships within the social networking site. This design allowed for efficient data storage and retrieval.

**2. User Management:** The DBMS facilitated the creation and management of user accounts, authentication, and authorization mechanisms. It ensured that users could securely access the site and perform actions based on their permissions.

**3. Content Storage and Retrieval:** The DBMS efficiently stored and retrieved user-generated content, such as posts, photos, videos, and comments. Techniques such as indexing, caching, and data partitioning were utilized to optimize content retrieval speed and enhance user experience.

**4. Relationship Management:** The DBMS managed connections and relationships between users, enabling features like friend requests, followers, and mutual connections. These functionalities were implemented to facilitate social interactions and network growth.

**5. Scalability and Performance:** As the social networking site aimed to accommodate a large user base, scalability and performance were crucial considerations. The DBMS architecture was designed to handle increasing data volumes and concurrent user activity, ensuring a smooth and responsive user experience.

**6. Data Privacy and Security: The** DBMS played a vital role in implementing data privacy and security measures. User data was protected through encryption, access controls, and other security mechanisms to safeguard personal information and prevent unauthorized access.

In conclusion, the successful implementation of a social networking site using a DBMS required careful consideration of database design, user management, content storage, scalability, and security. The project aimed to provide users with a seamless, engaging, and secure platform for social interactions, and the DBMS played a critical role in achieving these goals.

**4.2 FUTURE WORK**

In future work for a database management system (DBMS) project for a social networking site, a key area of focus could be on implementing artificial intelligence (AI) and machine learning (ML) algorithms to enhance user experiences. By leveraging AI and ML, the DBMS can provide personalized content recommendations, intelligent search capabilities, sentiment analysis, and user behavior predictions. These technologies can significantly improve user engagement, satisfaction, and the overall quality of interactions within the social networking site, making it more dynamic and tailored to individ