Social Media Analysis & Warehouse

Mini Project Report -Database Lab (DSE 2241)

Department of Data Science & Computer Applications

B. Tech Data Science

4th Semester – Batch: B2 - Group:

Submitted By

By:-

Samarth Agrawal	230968358
Sai Avinash Patoju	230968356
Anchal Mogapady	230968332
Vansh Pahwa	230968346
Ashrit Reddy	230968334
Prajjnaa Ray Choudhury	230968348
Srija Chatterjee	230968392
Mudit Manas	230968384
Arnav Kumar	230968192

Mentored By

Vinayak M

Assistant Professor-Senior

DSCA, MIT

Archana H

Assistant Professor-Senior

DSCA, MIT

Date: 11th April 2025

CERTIFICATE

This is to certify that the Samarth Agrawal (230968358) , Sai Avinash Patoju (230968356) , Anchal Mogapady (230968332) , Vansh Pahwa (230968346) , Ashrit Reddy (230968334) , Prajjnaa Ray Choudhury (230968348) , Srija Chatterjee (230968392), Mudit Manas (230968384) , Arnav Kumar (230968192) have successfully executed a mini project titled **Social Media Analysis And Data Warehouse**

rightly bringing fore the competencies and skill sets they have gained during the course- Database Lab (DSE 2241), thereby resulting in the culmination of this project.

Vinayak M	**Archana H**
Assistant Professor-Senior	**Assistant Professor-Senior**
DSCA, MIT	**DSCA, MIT**

ABSTRACT

In today's digital landscape, social media plays a crucial role in shaping consumer behaviour and brand perception. Businesses are increasingly relying on social media platforms such as Twitter, Instagram, and Facebook to enhance their marketing strategies and gain a competitive edge. However, the vast amount of unstructured data generated across these platforms presents challenges in extracting meaningful insights. This project aims to develop a Social Media Analysis & Warehouse that consolidates and analyses data to provide actionable insights for businesses.

The project utilizes **SQL Plus** as the primary database for storing social media data, we have used our own data for user details. Frontend technologies like **Streamlit** are used to create dynamic user interfaces.This simplified tech stack ensures effecient data organization and query performance, focuses on collecting and storing posts, engagement metrics, and demographic data while providing capabilities for growth tracking and user identification.

Key features of the system include comprehensive management of various entities through distinct tables. The Registration table captures user sign-up information, including user ID, name, email, and registration date. The Login table manages authentication data with user credentials and timestamps of login activities. The User table contains detailed user profiles such as bio and contact information. The Post table records user-generated content, including post ID, user ID, content, The Comment table stores comments made on posts with attributes like comment ID, post ID, user ID, and content. The system also includes comprehensive features related to social networking. The Groups table manages data about different user-created groups, including group name, description, and creation date, while the Group_Members table tracks the users who are members of these groups. The Friend_Request table records friend requests sent between users, capturing details such as sender ID, receiver ID, request status . The Friends table maintains established friendships, listing user pairs with corresponding friendship start dates. These interconnected tables enhance the user's capability to foster social interactions and provide personalized experiences.

In conclusion, the **Social Media Analysis & Warehouse** provides a comprehensive solution for tracking, analysing, and reporting social media performance. The system is supported by a robust relational database structure, featuring tables for registration, login, users, posts, comments, groups, group

members, friend requests, and friends . Each table is designed to capture critical attributes, such as group details and user interactions. This modular and scalable design allows businesses to extract precise, actionable insights, empowering them to refine their social media strategies, drive user engagement, and optimize their campaigns for long-term growth.

Contents

Chapter 1
Introduction

In the era of digital transformation, social media platforms have become a central part of human interaction, communication, and content sharing. Platforms like Facebook, Instagram, Twitter, and LinkedIn have revolutionized how individuals connect, share opinions, and create communities. These platforms generate massive volumes of data every second in the form of user registrations, posts, likes, comments, friend requests, and group activities. Analyzing this wealth of data has become crucial for understanding user behavior, tracking trends, enhancing user experience, and driving decisions in marketing, product development, and public policy. This increasing reliance on data has created a need for robust systems that not only store this data but also support efficient analysis and reporting.

Existing social media platforms primarily focus on delivering interactive and engaging user experiences. However, their internal data storage mechanisms and analytics tools are not always accessible for external analysis or academic study. Furthermore, the complexity and scale of these platforms make it challenging to study their internal data structures or apply custom analytics tools. As a result, there is a growing demand for educational and research-oriented systems that simulate social media environments while providing easy access to their underlying data for analysis. These systems must support core functionalities like user registration, login, posting, commenting, sending friend requests, forming groups, and more—while storing data in a structured and retrievable format suitable for in-depth analytics.

The **Social Media Analysis & Warehouse** project has been developed with this objective in mind. It is a database-driven system designed to simulate the core functions of a social media platform while also enabling effective data storage and analysis. The backend of the system is developed using **SQL Plus**, where multiple interrelated tables such as registration, login, users, posts, comments, groups, group_members, friend_requests, and friends are created to manage and organize data efficiently. Each table is normalized to avoid redundancy and support scalability. The frontend is built using **Python Streamlit**, a modern and lightweight framework that allows the development of interactive web applications with ease. Through Streamlit, users

can interact with the system via user-friendly forms and dashboards for different functionalities such as registering, logging in, posting updates, commenting, joining groups, sending friend requests, and more.

The need for this system arises from the limitations of current social media platforms in terms of openness, flexibility, and customization for learning or research. By offering a smaller-scale but realistic version of a social media system, the Social Media Analysis Warehouse enables users, developers, and researchers to perform detailed data analysis, visualize trends, and understand social behavior patterns without dealing with the complexities of large-scale commercial platforms. This project bridges the gap between raw data generation and meaningful insights by providing a structured environment for experimenting with social media data and evaluating performance, user interaction, and network growth.

Chapter 2

Synopsis

2.1 Proposed System

This project addresses the challenge of extracting insights from the vast, unstructured data generated on social media platforms. Businesses increasingly rely on platforms like Facebook and Instagram to drive engagement, but lack tools to consolidate and analyze data effectively.
This project offers a simplified yet functional system built with **SQL Plus** for the backend and **Streamlit** for the frontend. It features a structured relational database comprising tables for **registration, login, users, posts, comments, groups, group members, friend requests, and friends**. These tables simulate social interactions while enabling efficient data storage and retrieval.
By using custom sample data, the system supports core social media functions and provides a foundation for tracking user activity, engagement trends, and network growth. The project helps businesses and researchers gain actionable insights from social data, laying the groundwork for future analytics and strategy optimization.

2.2 Objectives

The Main Objective of the work are ,,,,,,,,,,,,

- To design and implement a structured database system that simulates key functionalities of a social media platform such as user registration, login, posts, comments, likes, and friendships.
- To develop a relational data model using SQL Plus that efficiently stores and manages various entities like users, groups, and interactions while maintaining data integrity and minimizing redundancy.
- To enable real-time user interaction through a lightweight and user-friendly frontend developed in Python using Streamlit.
- To implement stored procedures and functions that automate tasks such as user authentication, friend request handling, post analytics, and group management.
- To perform analytical queries and data aggregation for tracking user engagement, identifying popular content, and measuring community growth over time.

Chapter 3
Functional Requirements

1. Procedure: User Registration

INPUT	**First Name, Last Name, Gender, DOB, Email, Password**
PROCESSING	Generate unique User ID. Insert data into User_Registration and User_Login tables. Set user status to 'Active'.
OUTPUT	"User Registered Successfully" message

2. Procedure: User Login

INPUT	**Email, Password**
PROCESSING	Check if the user exists with provided email and password. If valid, update status to 'Active'.
OUTPUT	"Login successful" or "Invalid email or password." message

3. Procedure: SendFriendRequest

INPUT	**Sender ID, Receiver ID**
PROCESSING	Generate unique Request ID. Insert friend request into Send_Request table.
OUTPUT	"Friend request sent." message

4. Procedure: AcceptFriendRequest

INPUT	**Sender ID, Receiver ID**
PROCESSING	Generate friend IDs. Add both users as friends in Friends table. Remove request from Send_Request.
OUTPUT	"Friend request accepted." message

5. Procedure: Add User to Group

INPUT	**Group ID, User ID**
PROCESSING	Generate unique Group Member ID. Insert user into Group_Members table.
OUTPUT	"User added to group successfully." message

6. Procedure: RemoveUserFromGroup

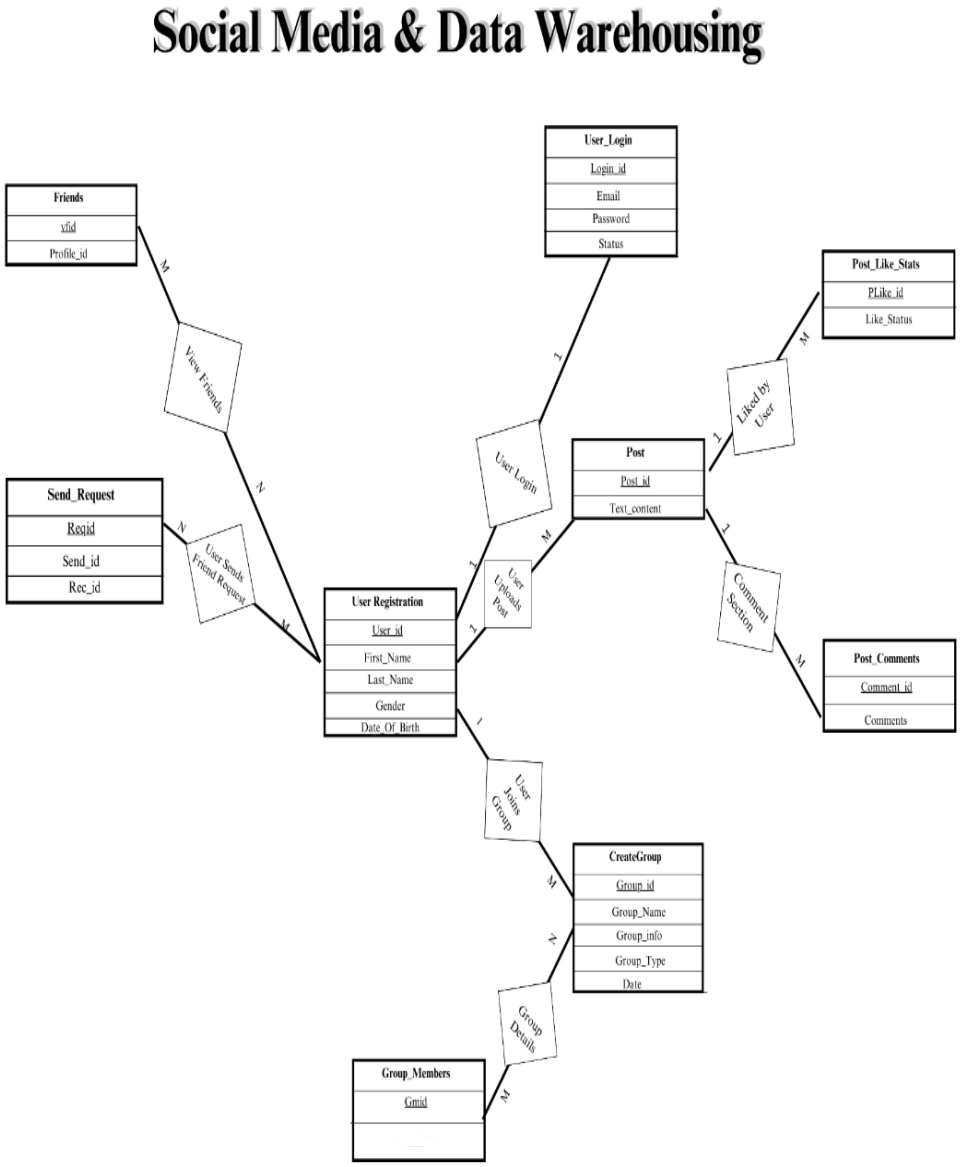
INPUT	**Group ID, User ID**
PROCESSING	Delete user from Group_Members table where both IDs match.
OUTPUT	"User removed from group." message

7. Procedure: Get Group Members

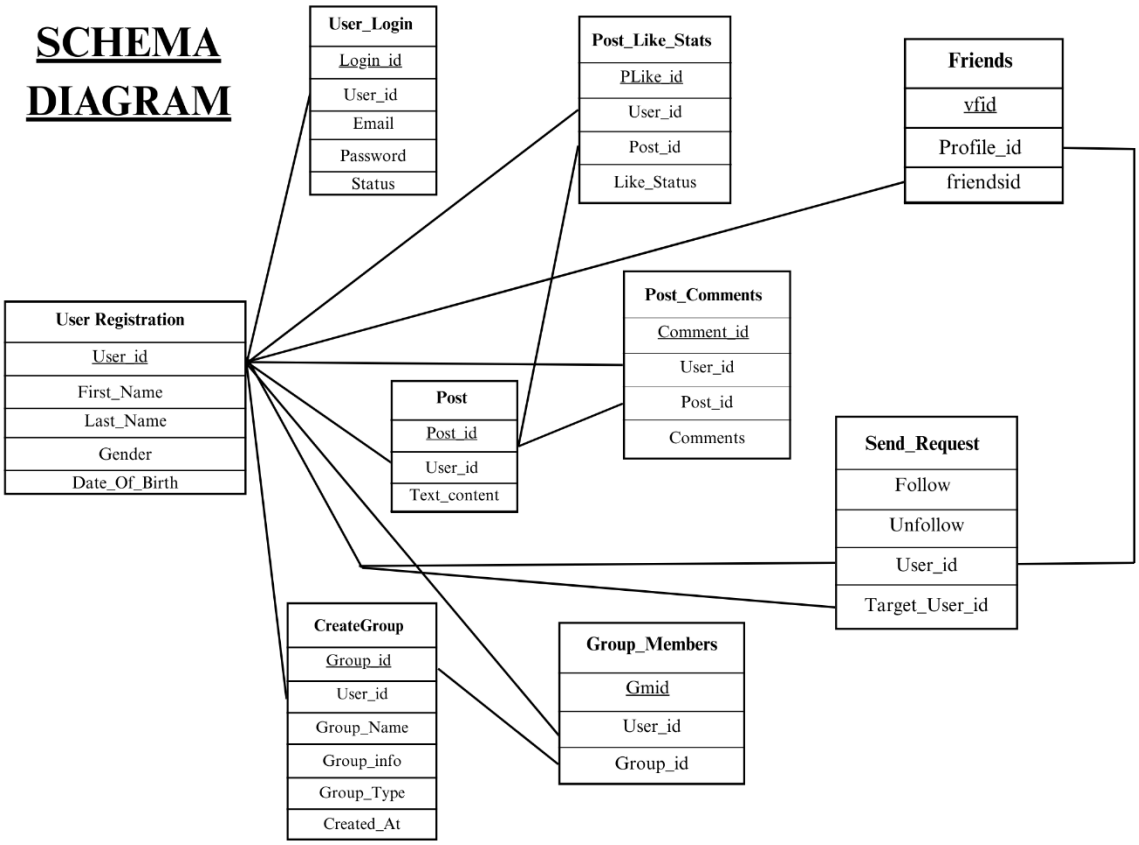
INPUT	**Group ID**
PROCESSING	Fetch all user names in the group using join between Group_Members and User_Registration.
OUTPUT	List of group members via cursor

Chapter 4
Detailed Design

4.1 ER Diagram



4.2 Schema Diagram



4.3 Relational Schema

User_Registration(User_id,First_name,Last_name,Gender,Date_Of_Birth)

User_Login(Login_id, Email,Password,Status,User_id)

Create_Group(Group_id,User_id,Group_name,Group_info,Group_type,Created_At)

Group_Members(Gmid, Group_id,User_id)

Post_Comments(Comment_id,Post_id,comments,user_id)

Send_Request(User_id, Target_User_id, Follow,Unfollow)

Friends(vfid,Profile_id,Friendsid)

Post(Post_id,Text_Content,User_id)

Post_Like_stats(Plike_id, Post_id, User_id, Like_Status)

4.4 Data Dictionary:

1. User_Registration

Column Name	Data Type	Constraint	Constraint Name
User_id	int	PRIMARY KEY	uid_pk
First_name	varchar2(20)	NOT NULL	
Last_name	varchar2(20)	NOT NULL	
Gender	varchar2(8)	NOT NULL	
Date_Of_Birth	date	NOT NULL	

2. User_Login

Column Name	Data Type	Constraint	Constraint Name
Login_id	number	PRIMARY KEY	loid_pk
Email	varchar2(20)	NOT NULL	
Password	varchar2(20)	NOT NULL	
Status	number(1)	NOT NULL	
User_id	number	REFERENCES User_Registration(User_id)	

3. Post

Column Name	Data Type	Constraint	Constraint Name
Post_id	number	PRIMARY KEY	post_pk
Text_content	clob	NOT NULL	
User_id	number	REFERENCES User_Registration(User_id)	

4. CreateGroup

Column Name	Data Type	Constraint	Constraint Name
Group_id	number	PRIMARY KEY	group_pk
User_id	number	REFERENCES User_Registration(User_id)	
Group_Name	varchar2(40)	NOT NULL	
Group_info	varchar2(200)	NOT NULL	
Group_type	varchar2(20)	NOT NULL	
Created_At	timestamp(6)	NOT NULL, DEFAULT CURRENT_TIMESTAMP	

5. Send_Request

Column Name	Data Type	Constraint	Constraint Name
User_id	number	REFERENCES User_Registration(User_id)	
Target_User_id	number	REFERENCES User_Registration(User_id)	
Follow	number(1)	DEFAULT 0	
Unfollow	number(1)	DEFAULT 0	

6. Friends

Column Name	Data Type	Constraint	Constraint Name
vfid	number	PRIMARY KEY	vfi_pk
Profile_id	number	NOT NULL	
friendsid	number	REFERENCES User_Registration(User_id)	

7. Post_like_stats

Column Name	Data Type	Constraint	Constraint Name
Plike_id	number	PRIMARY KEY	plike_pk
Post_id	number	REFERENCES Post(Post_id)	
User_id	number	REFERENCES User_Registration(User_id)	
Like_Status	number(1)	NOT NULL	

8. Post_Comments

Column Name	Data Type	Constraint	Constraint Name
Comment_id	number	PRIMARY KEY	comm_pk
Post_id	number	REFERENCES Post(Post_id)	
Comments	clob	NOT NULL	
User_id	number	REFERENCES User_Registration(User_id)	

9. Group_Members

Column Name	Data Type	Constraint	Constraint Name
Gmid	number	PRIMARY KEY	gm_pk
Group_id	number	REFERENCES CreateGroup(Group_id)	
User_id	number	REFERENCES User_Registration(User_id)	

4.5 Relational Model Implementation

```
CREATE TABLE User_Registration (
   User_id INT CONSTRAINT uid_pk PRIMARY KEY,
   First_name VARCHAR2(20) NOT NULL,
   Last_name VARCHAR2(20) NOT NULL,
   Gender VARCHAR2(8) NOT NULL,
   Date_Of_Birth DATE NOT NULL
 );
CREATE TABLE User_Login (
   Login_id NUMBER CONSTRAINT loid_pk PRIMARY KEY,
   Email VARCHAR2(40) NOT NULL,
   Password VARCHAR2(20) NOT NULL,
   Status NUMBER(1) NOT NULL,
   User_id NUMBER REFERENCES User_Registration(User_id)
 );
CREATE TABLE Post (
   Post_id NUMBER CONSTRAINT post_pk PRIMARY KEY,
   Text_content CLOB NOT NULL,
   User_id NUMBER REFERENCES User_Registration(User_id)
 );

CREATE TABLE CreateGroup (
   Group_id NUMBER CONSTRAINT group_pk PRIMARY KEY,
   User_id NUMBER REFERENCES User_Registration(User_id),
   Group_Name VARCHAR2(40) NOT NULL,
   Group_info VARCHAR2(200) NOT NULL,
```

```
    Group_type VARCHAR2(20) NOT NULL,
    Created_At TIMESTAMP DEFAULT
CURRENT_TIMESTAMP NOT NULL
  );
CREATE TABLE Send_Request (
    User_id NUMBER REFERENCES User_Registration(User_id),
    Target_User_id NUMBER REFERENCES User_Registration(User_id),
    Follow NUMBER(1) DEFAULT 0,
    Unfollow NUMBER(1) DEFAULT 0
);

CREATE TABLE Friends (
    vfid NUMBER CONSTRAINT VFI_PK PRIMARY KEY,
    Profile_id NUMBER NOT NULL,
    friendsid NUMBER REFERENCES User_Registration(User_id)
  );
CREATE TABLE Post_Like_stats (
    Plike_id NUMBER CONSTRAINT PLIKE_PK PRIMARY KEY,
    Post_id NUMBER REFERENCES Post(Post_id),
    User_id NUMBER REFERENCES User_Registration(User_id),
    Like_Status NUMBER(1) NOT NULL
  );
CREATE TABLE Post_Comments (
    Comment_id NUMBER CONSTRAINT COMM_PK PRIMARY KEY,
    Post_id NUMBER REFERENCES Post(Post_id),
    Comments CLOB NOT NULL,
    User_id NUMBER REFERENCES User_Registration(User_id)
  );
```

```
CREATE TABLE Group_Members (
   Gmid NUMBER CONSTRAINT GM_PK PRIMARY KEY,
   Group_id NUMBER REFERENCES CreateGroup(Group_id),
   User_id NUMBER REFERENCES User_Registration(User_id)
  );
```

5. Implementation

5.1 Queries

1. Q: List all users who registered before January 1, 1996.

```
SELECT *
FROM User_Registration
WHERE DATE_OF_BIRTH < TO_DATE('1996-01-01', 'YYYY-MM-DD');
```

```
   USER_ID FIRST_NAME           LAST_NAME            GENDER   DATE_OF_B
---------- -------------------- -------------------- -------- ---------
         2 Isha                 Patel                Female   05-FEB-95
         4 Priya                Patel                Female   11-AUG-95
         5 Tirth                Patel                Male     24-OCT-95
         7 Raj                  Mistry               Male     20-DEC-95
         9 Kunal                Shah                 Male     18-JUN-95
        11 Harsh                Bhatt                Male     01-DEC-95

6 rows selected.
```

---

2. Q: Retrieve the names and email addresses of users who are currently online (ostatus = 1).

```
SELECT r.first_name, r.last_name, l.email
FROM User_Registration r
JOIN User_Login l ON r.user_id = l.user_id
WHERE l.status = 1;
```

```
FIRST_NAME           LAST_NAME
-------------------- --------------------
EMAIL
----------------------------------------
Abhishek             Raval
abhishek.raval@gmail.com

Isha                 Patel
isha.patel@gmail.com

Priya                Patel
priya.patel@gmail.com


FIRST_NAME           LAST_NAME
-------------------- --------------------
EMAIL
----------------------------------------
Vishakha             Trivedi
vishakha.trivedi@gmail.com

Kunal                Shah
kunal.shah@gmail.com

Sneha                Mehta
sneha.mehta@gmail.com
```

```
FIRST_NAME           LAST_NAME
-------------------- --------------------
EMAIL
----------------------------------------
Krupa                Joshi
krupa.joshi@gmail.com


7 rows selected.
```

---

3. Q: Show the total number of friends each user has.

SELECT Profile_id, COUNT(friendsid) AS total_friends

FROM Friends

GROUP BY Profile_id;

```
PROFILE_ID TOTAL_FRIENDS
---------- -------------
         1             7
         2             5
         3             4
         4             7
         6             5
         5             4
         7             5
         8             3
         9             2
        10             2
        11             2

PROFILE_ID TOTAL_FRIENDS
---------- -------------
        12             2

12 rows selected.
```

---

4. Q: List all group names and their creators' full names.

SELECT g.group_name, r.first_name || ' ' || r.last_name AS creator_name FROM CreateGroup g JOIN User_Registration r ON g.user_id = r.user_id;

```
GROUP_NAME
----------------------------------------
CREATOR_NAME
----------------------------------------
University Selection Talks
Abhishek Raval

Quant Preparation
Abhishek Raval

Blogging Talks
Isha Patel


GROUP_NAME
----------------------------------------
CREATOR_NAME
----------------------------------------
GATE Civil Engineering
Milan Pandya

Game Designing
Priya Patel

Harvard Students
Tirth Patel
```

```
GROUP_NAME
----------------------------------------
CREATOR_NAME
----------------------------------------
CAD/CAM
Utsavi Desai

Pokemon GO FANS
Vishakha Trivedi

AI Research Group
Kunal Shah


GROUP_NAME
----------------------------------------
CREATOR_NAME
----------------------------------------
Fitness and Wellness
Sneha Mehta


10 rows selected.
```

5. Q: Retrieve all posts along with the name of the user who posted them.

SELECT p.post_id, p.text_content, r.first_name || ' ' || r.last_name AS posted_by

FROM Post p

JOIN User_Registration r ON p.user_id = r.user_id;

```
   POST_ID
----------
TEXT_CONTENT
--------------------------------------------------------------------------------
POSTED_BY
----------------------------------------
         1
rid1 - first: Hello, I am Abhishek Raval (rid1), this is my first ever post.
Abhishek Raval

         2
rid1 - second: Hello, I am Abhishek Raval (rid1), this is my second post.
Abhishek Raval

   POST_ID
----------
TEXT_CONTENT
--------------------------------------------------------------------------------
POSTED_BY
----------------------------------------

         3
rid2 - first: Hello, I am Ishan H (rid2), this is my first ever post.
Isha Patel

         4
rid3 - first: Hello, I am Milan Pandya (rid3), this is my first ever post.
```

```
rid9 - first: Hello, I am Kunal Shah (rid9), first post!
Kunal Shah

        28
rid10 - first: Hello, I am Sneha Mehta (rid10), first post!
Sneha Mehta


   POST_ID
---------
TEXT_CONTENT
-----------------------------------------------------------------------------
POSTED_BY
----------------------------------------
        29
rid11 - first: Hello, I am Harsh Bhatt (rid11), first post!
Harsh Bhatt

        30
rid12 - first: Hello, I am Krupa Joshi (rid12), first post!
Krupa Joshi

   POST_ID
---------
TEXT_CONTENT
-----------------------------------------------------------------------------
POSTED_BY
----------------------------------------


30 rows selected.
```

---

6. Q: Find users who have liked any post.

SELECT DISTINCT r.first_name, r.last_name

FROM Post_Like_stats pls

JOIN User_Registration r ON pls.user_id = r.user_id

WHERE pls.like_status = 1;

```
FIRST_NAME           LAST_NAME
-------------------- --------------------
Abhishek             Raval
Isha                 Patel
Milan                Pandya
Priya                Patel
Tirth                Patel
Utsavi               Desai
Raj                  Mistry
Vishakha             Trivedi

8 rows selected.
```

---

7. Q: Display comments on each post with commenter names.

SELECT pc.post_id, pc.comments, r.first_name || ' ' || r.last_name AS commenter

FROM Post_Comments pc

JOIN User_Registration r ON pc.user_id = r.user_id;

```
   POST_ID
----------
COMMENTS
--------------------------------------------------------------------------------
COMMENTER
----------------------------------------
         1
welcome to fb, i am ishan h
Isha Patel

         1
Thanks
Abhishek Raval

   POST_ID
----------
COMMENTS
--------------------------------------------------------------------------------
COMMENTER
----------------------------------------

         1
Milan here, good to see you.
Milan Pandya
```

```
Great post, Kunal!
Isha Patel

         21
Welcome, Sneha!
Milan Pandya


   POST_ID
----------
COMMENTS
--------------------------------------------------------------------------------
COMMENTER
----------------------------------------
         27
Nice to see you here, Harsh!
Priya Patel

         25
Hey Krupa, welcome!
Tirth Patel

   POST_ID
----------
COMMENTS
--------------------------------------------------------------------------------
COMMENTER
----------------------------------------


30 rows selected.
```

---

8. Q: Retrieving user information based on email and password entered.

SELECT User_id

FROM User_Login

WHERE Email =’ email_id_here’ AND Password =’enter_password’ and status=1;

```
   USER_ID
----------
         1
```

---

9. Q: Retrieving top 5 recent posts.

SELECT p.Post_id, p.Text_content, (u.First_name || ' ' || u.Last_name) AS full_name

FROM Post p

JOIN User_Registration u ON p.User_id = u.User_id

ORDER BY p.Post_id DESC

FETCH FIRST 5 ROWS ONLY;

```
   POST_ID
----------
TEXT_CONTENT
--------------------------------------------------------------------------------
FULL_NAME
----------------------------------------
        30
rid12 - first: Hello, I am Krupa Joshi (rid12), first post!
Krupa Joshi

        29
rid11 - first: Hello, I am Harsh Bhatt (rid11), first post!
Harsh Bhatt

   POST_ID
----------
TEXT_CONTENT
--------------------------------------------------------------------------------
FULL_NAME
----------------------------------------

        28
rid10 - first: Hello, I am Sneha Mehta (rid10), first post!
Sneha Mehta

        27
rid9 - first: Hello, I am Kunal Shah (rid9), first post!

   POST_ID
----------
TEXT_CONTENT
--------------------------------------------------------------------------------
FULL_NAME
----------------------------------------
Kunal Shah

        26
Thanks for adding me to the Quant Prep group - Vish
Vishakha Trivedi
```

---

10. Q: Counting number of user registration.

SELECT COUNT(*) FROM User_Registration;

11. Q: Counting number of posts.

SELECT COUNT(*) FROM Post;

12. Q: Counting number of groups.

SELECT COUNT(*) FROM CreateGroup;

Output of all Q10,11,12

```
SQL> SELECT COUNT(*) FROM User_Registration;

  COUNT(*)
----------
        12

SQL> SELECT COUNT(*) FROM Post;

  COUNT(*)
----------
        30

SQL> SELECT COUNT(*) FROM CreateGroup;

  COUNT(*)
----------
        10
```

---

13. Q: Retrieving the list of registered users.

SELECT User_id, First_Name || ' ' || Last_Name FROM User_Registration;

```
   USER_ID FIRST_NAME||''||LAST_NAME
---------- ----------------------------------------
         1 Abhishek Raval
         2 Isha Patel
         3 Milan Pandya
         4 Priya Patel
         5 Tirth Patel
         6 Utsavi Desai
         7 Raj Mistry
         8 Vishakha Trivedi
         9 Kunal Shah
        10 Sneha Mehta
        11 Harsh Bhatt

   USER_ID FIRST_NAME||''||LAST_NAME
---------- ----------------------------------------
        12 Krupa Joshi

12 rows selected.
```

---

14. Q: Retrieves group details and arranges it in descending order.

SELECT g.Group_id, g.Group_Name, g.Group_info, g.Group_type, u.First_name || ' ' || u.Last_name

FROM CreateGroup g JOIN User_Registration u ON g.User_id = u.User_id

ORDER BY g.Group_id DESC;

```
  GROUP_ID GROUP_NAME
--------- ------------------------------------
GROUP_INFO
--------------------------------------------------------------------------------
GROUP_TYPE          U.FIRST_NAME||''||U.LAST_NAME
------------------- ----------------------------------------
        10 Fitness and Wellness
Sharing tips for mental and physical wellness.
Public              Sneha Mehta

         9 AI Research Group
Discuss AI and ML topics.
Closed              Kunal Shah

  GROUP_ID GROUP_NAME
--------- ------------------------------------
GROUP_INFO
--------------------------------------------------------------------------------
GROUP_TYPE          U.FIRST_NAME||''||U.LAST_NAME
------------------- ----------------------------------------

         8 CAD/CAM
All about CAD/CAM.
Public              Utsavi Desai

         7 Harvard Students
All about life at Harvard and your experiences.
```

```
         4 GATE Civil Engineering
Here people share information about civil engineering, especially helpful for th
ose preparing for GATE.
Closed               Milan Pandya

  GROUP_ID GROUP_NAME
---------- ------------------------------------
GROUP_INFO
--------------------------------------------------------------------------------
GROUP_TYPE           U.FIRST_NAME||''||U.LAST_NAME
-------------------- ------------------------------------

         3 Blogging Talks
Here you can discuss about how to blog and tips to improve blog rankings.
Secret               Isha Patel

         2 Pokemon GO FANS
This group is about tricks and tips for all the Pokemon GO players.

  GROUP_ID GROUP_NAME
---------- ------------------------------------
GROUP_INFO
--------------------------------------------------------------------------------
GROUP_TYPE           U.FIRST_NAME||''||U.LAST_NAME
-------------------- ------------------------------------
Public               Vishakha Trivedi

         1 University Selection Talks
All about selecting appropriate universities for doing M.S.
Secret               Abhishek Raval


10 rows selected.
```

---

15. Q:Retrieves post details and arranges it in descending order.

SELECT p.Post_id, p.Text_content, u.First_name || ' ' || u.Last_name FROM Post p

JOIN User_Registration u ON p.User_id = u.User_id ORDER BY p.Post_id DESC;

```
   POST_ID
----------
TEXT_CONTENT
--------------------------------------------------------------------------------
U.FIRST_NAME||''||U.LAST_NAME
----------------------------------------
        30
rid12 - first: Hello, I am Krupa Joshi (rid12), first post!
Krupa Joshi

        29
rid11 - first: Hello, I am Harsh Bhatt (rid11), first post!
Harsh Bhatt

   POST_ID
----------
TEXT_CONTENT
--------------------------------------------------------------------------------
U.FIRST_NAME||''||U.LAST_NAME
----------------------------------------

        28
rid10 - first: Hello, I am Sneha Mehta (rid10), first post!
Sneha Mehta

        27
rid9 - first: Hello, I am Kunal Shah (rid9), first post!
```

```
U.FIRST_NAME||''||U.LAST_NAME
----------------------------------------
rid3 - first: Hello, I am Milan Pandya (rid3), this is my first ever post.
Milan Pandya

         3
rid2 - first: Hello, I am Ishan H (rid2), this is my first ever post.
Isha Patel


   POST_ID
----------
TEXT_CONTENT
--------------------------------------------------------------------------------
U.FIRST_NAME||''||U.LAST_NAME
----------------------------------------
         2
rid1 - second: Hello, I am Abhishek Raval (rid1), this is my second post.
Abhishek Raval

         1
rid1 - first: Hello, I am Abhishek Raval (rid1), this is my first ever post.
Abhishek Raval

   POST_ID
----------
TEXT_CONTENT
--------------------------------------------------------------------------------
U.FIRST_NAME||''||U.LAST_NAME
----------------------------------------


30 rows selected.
```

16. Q: Fetching friends for profile_id = 4.

```
SELECT DISTINCT ur.User_id, ur.First_name || ' ' || ur.Last_name AS
Friend_Name

FROM  Friends f

JOIN User_Registration ur

ON ur.User_id = CASE

        WHEN f.Profile_id = 4  THEN  f.friendsid

        WHEN f.friendsid = 4 THEN  f.Profile_id

     END

WHERE f.Profile_id = 4 OR f.friendsid = 4;
```

```
   USER_ID FRIEND_NAME
---------- ------------------------------------
         1 Abhishek Raval
         2 Isha Patel
         3 Milan Pandya
         5 Tirth Patel
         6 Utsavi Desai
         7 Raj Mistry
         8 Vishakha Trivedi
        12 Krupa Joshi

8 rows selected.
```

17. Q: Retrieving name of the individual with user_id=4.

```
SELECT First_name || ' ' || Last_name

 FROM User_Registration

 WHERE User_id = 4;
```

```
FIRST_NAME||''||LAST_NAME
----------------------------------------
Priya Patel
```

---

18. Q: Retrieving user information.

SELECT User_id, First_name, Last_name, Gender,
TO_CHAR(Date_Of_Birth, 'YYYY-MM-DD')

FROM User_Registration;

```
   USER_ID FIRST_NAME           LAST_NAME            GENDER   TO_CHAR(DA
---------- -------------------- -------------------- -------- ----------
         1 Abhishek             Raval                Male     1996-04-11
         2 Isha                 Patel                Female   1995-02-05
         3 Milan                Pandya               Male     1996-05-26
         4 Priya                Patel                Female   1995-08-11
         5 Tirth                Patel                Male     1995-10-24
         6 Utsavi               Desai                Female   1996-08-17
         7 Raj                  Mistry               Male     1995-12-20
         8 Vishakha             Trivedi              Female   1996-01-20
         9 Kunal                Shah                 Male     1995-06-18
        10 Sneha                Mehta                Female   1996-03-09
        11 Harsh                Bhatt                Male     1995-12-01

   USER_ID FIRST_NAME           LAST_NAME            GENDER   TO_CHAR(DA
---------- -------------------- -------------------- -------- ----------
        12 Krupa                Joshi                Female   1996-07-14

12 rows selected.
```

5.2 Stored Procedures

1]Overview of the ViewUserPostsWithStats Procedure

CREATE OR REPLACE PROCEDURE ViewUserPostsWithStats (

p_userId IN NUMBER,

p_cursor OUT SYS_REFCURSOR

```
)
IS
BEGIN
   OPEN p_cursor FOR
   SELECT
      P.Post_id,
      P.Text_content,
      (SELECT COUNT(*) FROM Post_Like_Stats WHERE Post_id =
P.Post_id) AS like_count,
      (SELECT COUNT(*) FROM Post_Comments WHERE Post_id =
P.Post_id) AS comment_count
   FROM Post P
   WHERE P.User_id = p_userId;
END;
/
```

2] Usage of Comment row and Comment table

```
CREATE OR REPLACE TYPE CommentRow AS OBJECT (
   Comment_id NUMBER,
   Comments   CLOB
  );
  /


 CREATE OR REPLACE TYPE CommentTable AS TABLE OF CommentRow;
/
```

3] Function to Retrieve User Comments as a Custom Object Collection

```
CREATE OR REPLACE FUNCTION GetUserComments(uId IN NUMBER)
  RETURN CommentTable PIPELINED
```

```
IS
BEGIN
  FOR rec IN (
    SELECT Comment_id, Comments
    FROM Post_Comments
    WHERE User_id = uId
  ) LOOP
    PIPE ROW (CommentRow(rec.Comment_id, rec.Comments));
  END LOOP;
 RETURN;
END;
/
```

4] Stored Procedure to Retrieve and Iterate Over Online Friends of a User

```
CREATE OR REPLACE PROCEDURE ShowOnlineFriends(p_userId IN
NUMBER) IS
BEGIN
  FOR rec IN (
    SELECT UR.User_id, UR.First_name, UR.Last_name
    FROM Friends F
    JOIN User_Registration UR ON UR.User_id = F.vfid
    JOIN User_Login UL ON UR.User_id = UL.User_id
    WHERE F.Profile_id = p_userId AND UL.Status = 1
  ) LOOP
    NULL; -- Replace this with logic to handle/display data if needed
  END LOOP;
END;
/
```

5] Stored Procedure to Retrieve Group Details with Admin Information and Member Count

```
CREATE OR REPLACE PROCEDURE GroupDetails IS
  BEGIN
    FOR rec IN (
      SELECT
        CG.Group_Name,
        CG.User_id AS Admin_id,
        COUNT(GM.User_id) AS MemberCount
      FROM CreateGroup CG
      LEFT JOIN Group_Members GM ON CG.Group_id = GM.Group_id
      GROUP BY CG.Group_Name, CG.User_id
    ) LOOP
      NULL; -- Placeholder: Add logic to process or display each row
    END LOOP;
  END;
  /
```

6] Stored Procedure to Retrieve Posts Without Any Likes Using a REF Cursor

```
CREATE OR REPLACE PROCEDURE PostsWithNoLikes(p_result OUT SYS_REFCURSOR) IS
BEGIN
    OPEN p_result FOR
    SELECT P.Post_id, P.Text_content
    FROM Post P
    LEFT JOIN Post_Like_Stats L ON P.Post_id = L.Post_id
    WHERE L.Post_id IS NULL;
END;
```

/

7] Stored Procedure to Identify Posts Without Comments

```
CREATE OR REPLACE PROCEDURE PostsWithNoComments IS
  BEGIN
    FOR rec IN (
      SELECT P.Post_id, P.Text_content
      FROM Post P
      LEFT JOIN Post_Comments C ON P.Post_id = C.Post_id
      WHERE C.Post_id IS NULL
    ) LOOP
      NULL; -- Placeholder: add logic to process or display each row if needed
    END LOOP;
  END;
```

8] Stored Procedure to Retrieve Posts Liked by Friends of a Specific User

```
CREATE OR REPLACE PROCEDURE FriendLikedUserPosts (
    p_user_id IN NUMBER,
    p_result OUT SYS_REFCURSOR
) AS
BEGIN
    OPEN p_result FOR
        SELECT DISTINCT
            p.Post_id,
            DBMS_LOB.SUBSTR(p.Text_content, 4000, 1) AS Text_content,
            ur.First_name || ' ' || ur.Last_name AS Friend_Name
        FROM Post p
        JOIN Post_Like_stats pls ON p.Post_id = pls.Post_id
        JOIN User_Registration ur ON pls.User_id = ur.User_id
```

```
    WHERE p.User_id = p_user_id
      AND pls.User_id IN (
          SELECT friendsid
          FROM Friends
          WHERE Profile_id = p_user_id
      )
      AND pls.Like_Status = 1;
END;
/
```

9] Stored Procedure to Authenticate User Login and Update Online Status

```
CREATE OR REPLACE PROCEDURE UserLogin (
    p_email IN VARCHAR2,
    p_password IN VARCHAR2
 ) AS
    v_count NUMBER;
 BEGIN
    SELECT COUNT(*) INTO v_count
    FROM User_Login
    WHERE Email = p_email AND Password = p_password;

    IF v_count = 1 THEN
      UPDATE User_Login
      SET Status = 'Active'
      WHERE Email = p_email;

      DBMS_OUTPUT.PUT_LINE('Login successful.');
    ELSE
```

```
        DBMS_OUTPUT.PUT_LINE('Invalid email or password.');
    END IF;
  END;
  /
```

10] Stored Procedure to Register a New User and Initialize Login Credentials

```
CREATE OR REPLACE PROCEDURE RegisterUser (
    p_first_name IN VARCHAR2,
    p_last_name IN VARCHAR2,
    p_gender IN VARCHAR2,
    p_dob IN DATE,
    p_email IN VARCHAR2,
    p_password IN VARCHAR2
  ) AS
    v_user_id NUMBER;
  BEGIN
    -- Generate next user ID
    SELECT NVL(MAX(User_id), 0) + 1 INTO v_user_id FROM
User_Registration;

    -- Insert into User_Registration table
    INSERT INTO User_Registration (User_id, First_Name, Last_Name,
Gender, Date_Of_Birth)
    VALUES (v_user_id, p_first_name, p_last_name, p_gender, p_dob);

    -- Insert into User_Login table
    INSERT INTO User_Login (Login_id, Email, Password, Status)
    VALUES (v_user_id, p_email, p_password, 'Active');
```

```
    DBMS_OUTPUT.PUT_LINE('User Registered Successfully');
  END;
  /
```

11] Stored Procedure to Send a Friend Request by Inserting a New Record

```
CREATE OR REPLACE PROCEDURE SendFriendRequest (
    p_sender_id IN NUMBER,
    p_receiver_id IN NUMBER
  ) AS
    v_req_id NUMBER;
  BEGIN
    -- Generate the next Reqid
    SELECT NVL(MAX(Reqid), 0) + 1 INTO v_req_id FROM Send_Request;

    -- Insert the new friend request
    INSERT INTO Send_Request (Reqid, Send_id, Rec_id)
    VALUES (v_req_id, p_sender_id, p_receiver_id);

    DBMS_OUTPUT.PUT_LINE('Friend request sent.');
  END;
  /
```

12] Stored Procedure to Accept a Friend Request and Establish Mutual Friendship

```
CREATE OR REPLACE PROCEDURE AcceptFriendRequest (
    p_sender_id IN NUMBER,
    p_receiver_id IN NUMBER
  ) AS
    v_max_id NUMBER;
  BEGIN
```

```
    -- Get current maximum vfid
    SELECT NVL(MAX(vfid), 0) INTO v_max_id FROM Friends;

    -- Insert both entries: one for each user
    INSERT INTO Friends (vfid, Profile_id, friendsid)
    VALUES (v_max_id + 1, p_sender_id, p_receiver_id);

    INSERT INTO Friends (vfid, Profile_id, friendsid)
    VALUES (v_max_id + 2, p_receiver_id, p_sender_id);

    -- Delete the request
    DELETE FROM Send_Request
    WHERE Send_id = p_sender_id AND Rec_id = p_receiver_id;

    DBMS_OUTPUT.PUT_LINE('Friend request accepted.');
  END;
  /
```

13] Stored Procedure to Add a User to a Group by Inserting Membership Record

```
CREATE OR REPLACE PROCEDURE AddUserToGroup (
    p_group_id IN NUMBER,
    p_user_id IN NUMBER
  ) AS
    v_gmid NUMBER;
  BEGIN
    -- Get the next available Gmid
    SELECT NVL(MAX(Gmid), 0) + 1 INTO v_gmid FROM
Group_Members;
```

```
    -- Insert the user into the group
    INSERT INTO Group_Members (Gmid, Group_id, User_id)
    VALUES (v_gmid, p_group_id, p_user_id);

    DBMS_OUTPUT.PUT_LINE('User added to group successfully.');
  END;
  /
```

14] Stored Procedure to Remove a User from a Group

```
CREATE OR REPLACE PROCEDURE RemoveUserFromGroup (
    p_group_id IN NUMBER,
    p_user_id IN NUMBER
  ) AS
  BEGIN
    DELETE FROM Group_Members
    WHERE Group_id = p_group_id AND User_id = p_user_id;

    DBMS_OUTPUT.PUT_LINE('User removed from group.');
  END;
  /
```

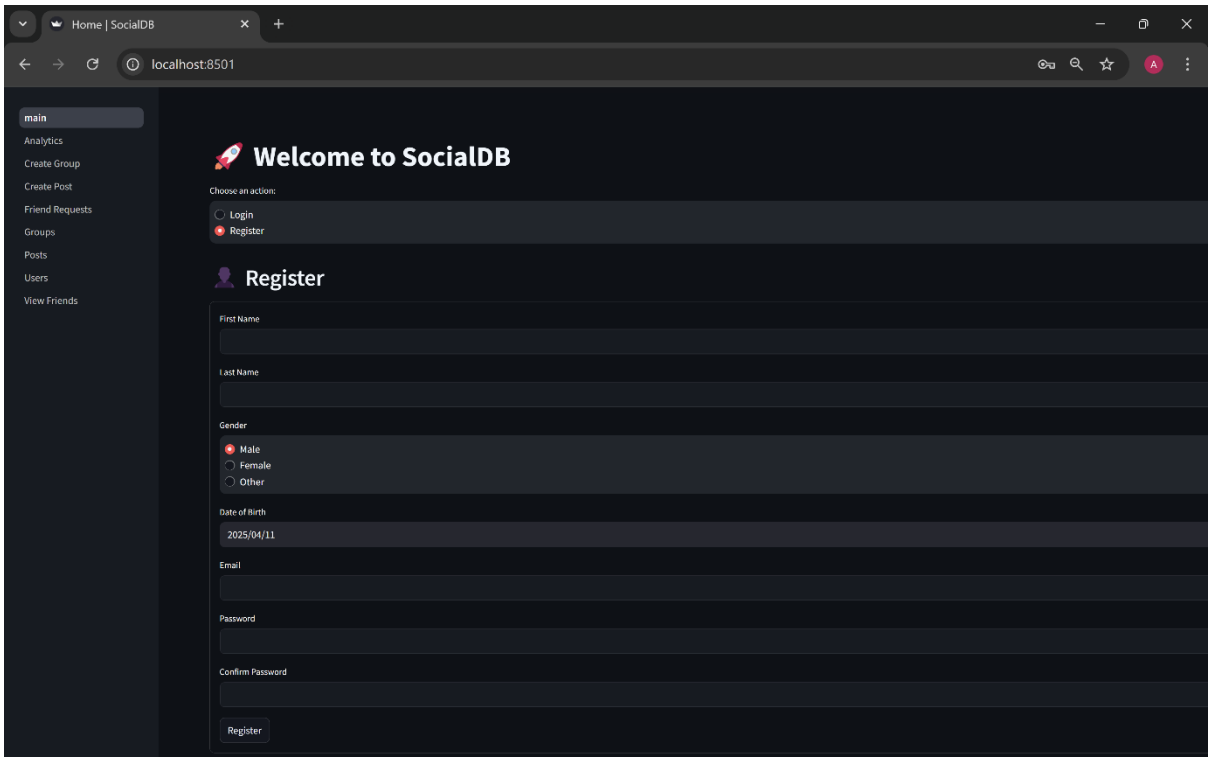
15] Stored Procedure to Retrieve All Members of a Specific Group Using a REF Cursor

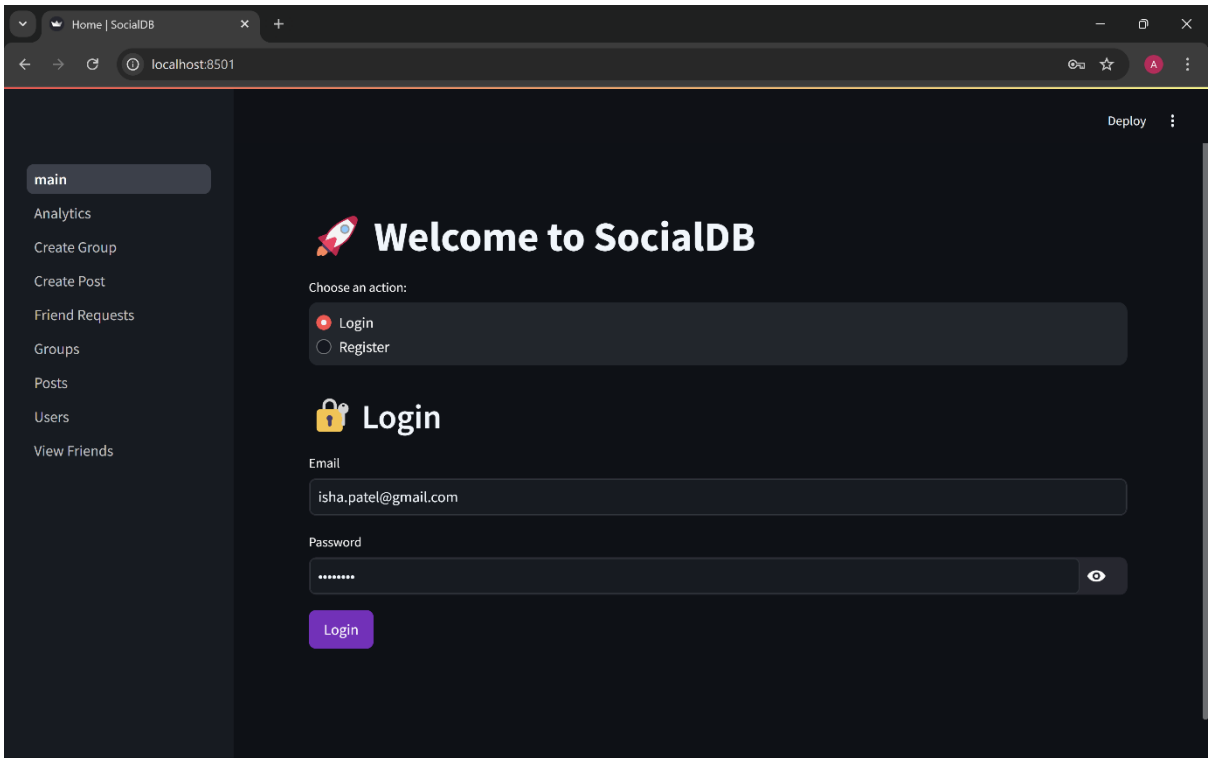
```
CREATE OR REPLACE PROCEDURE GetGroupMembers (
   p_group_id IN NUMBER,
   p_cursor OUT SYS_REFCURSOR
) AS
BEGIN
   OPEN p_cursor FOR
```

```
    SELECT ur.User_id, ur.First_name || ' ' || ur.Last_name AS FullName
    FROM Group_Members gm
    JOIN User_Registration ur ON gm.User_id = ur.User_id
    WHERE gm.Group_id = p_group_id;
END;
/
```

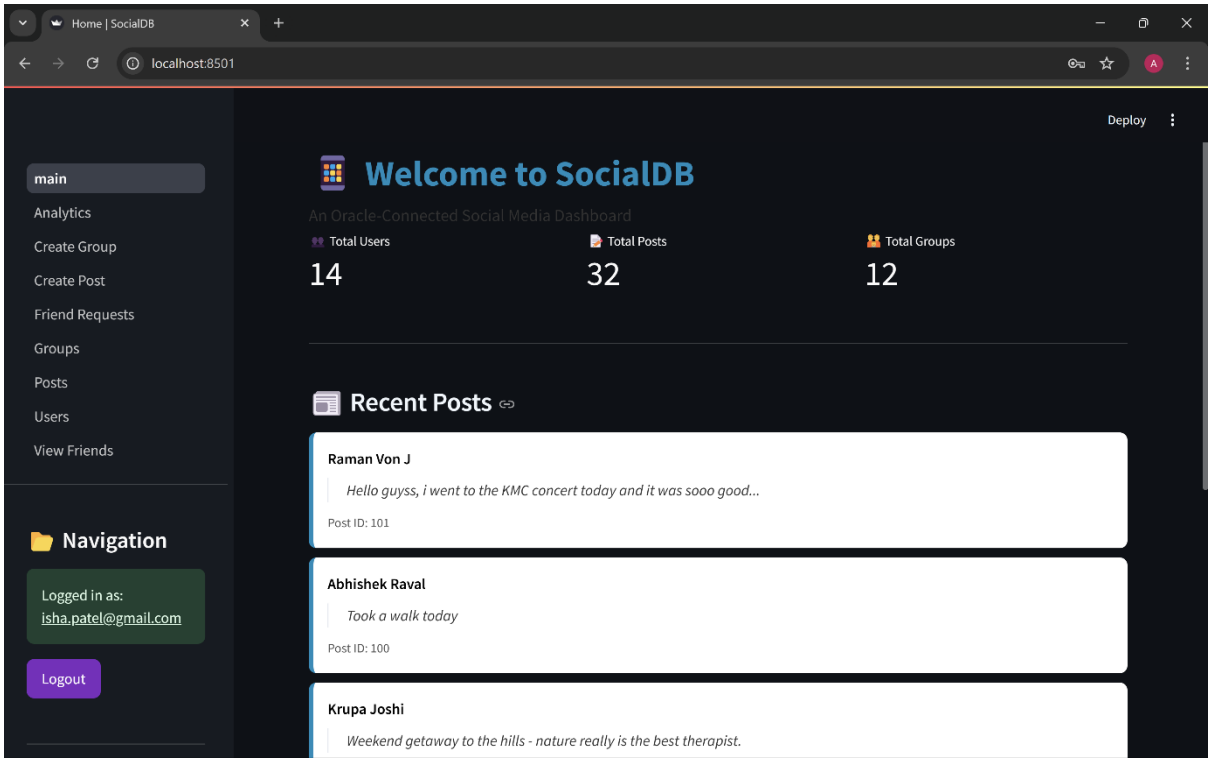
RESULTS

1. Login and Registration Page

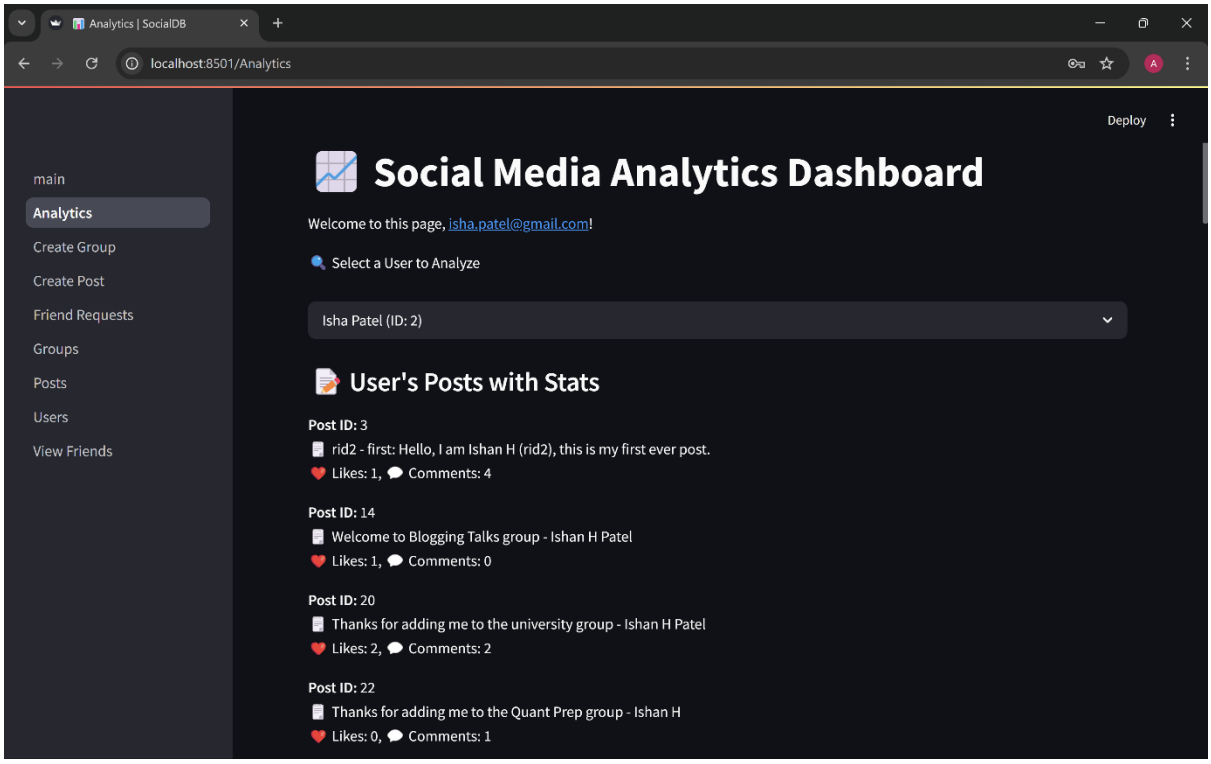




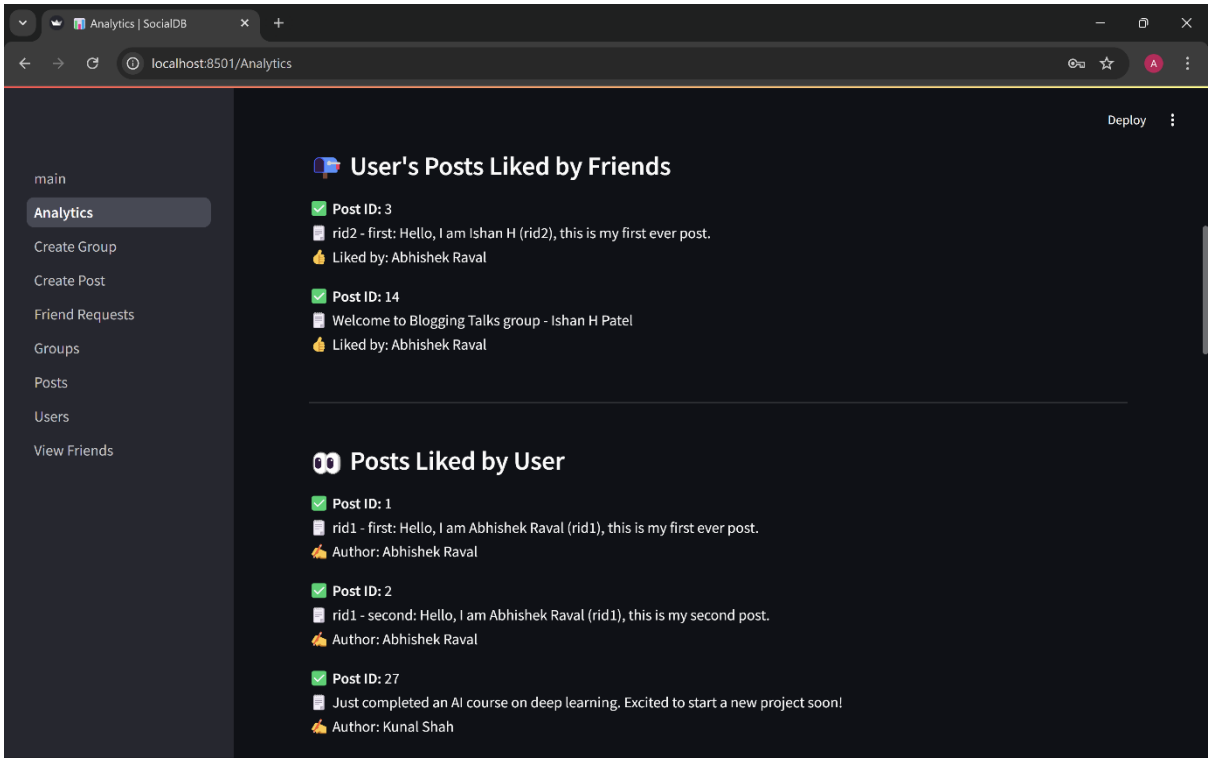
2. Home Page



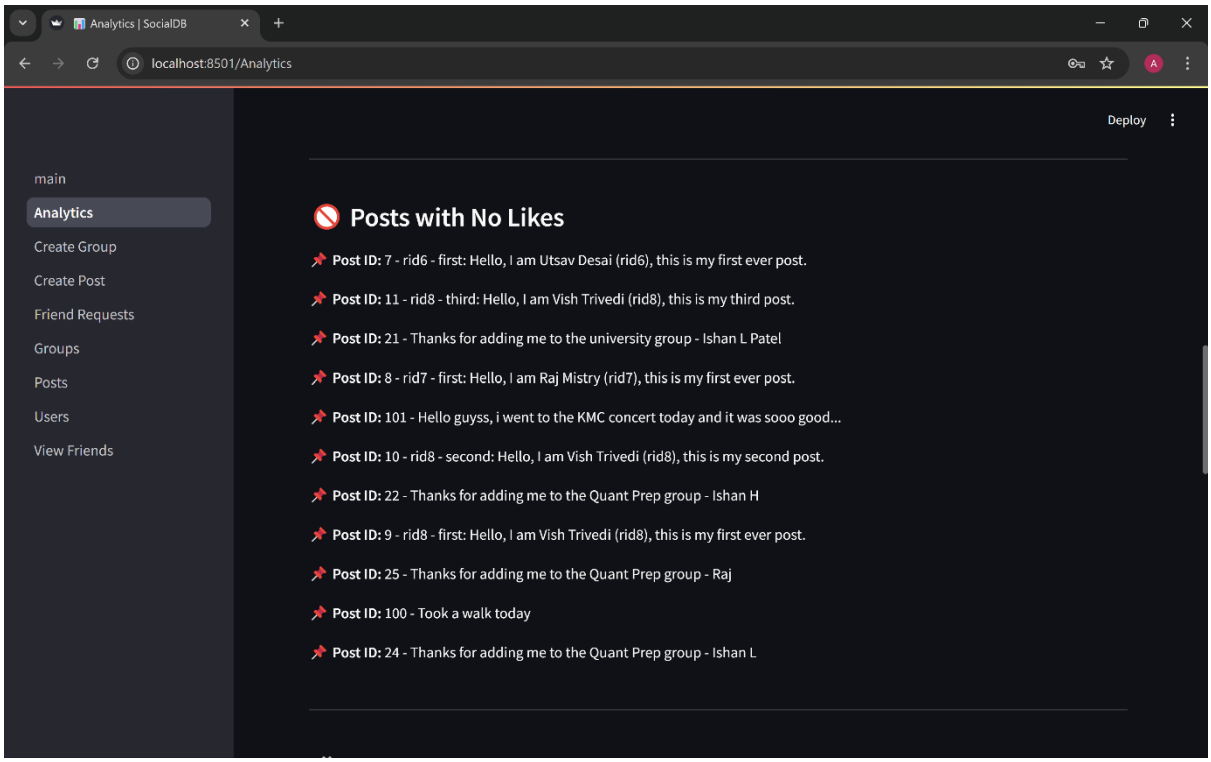
3. Analytics Dashboard Page



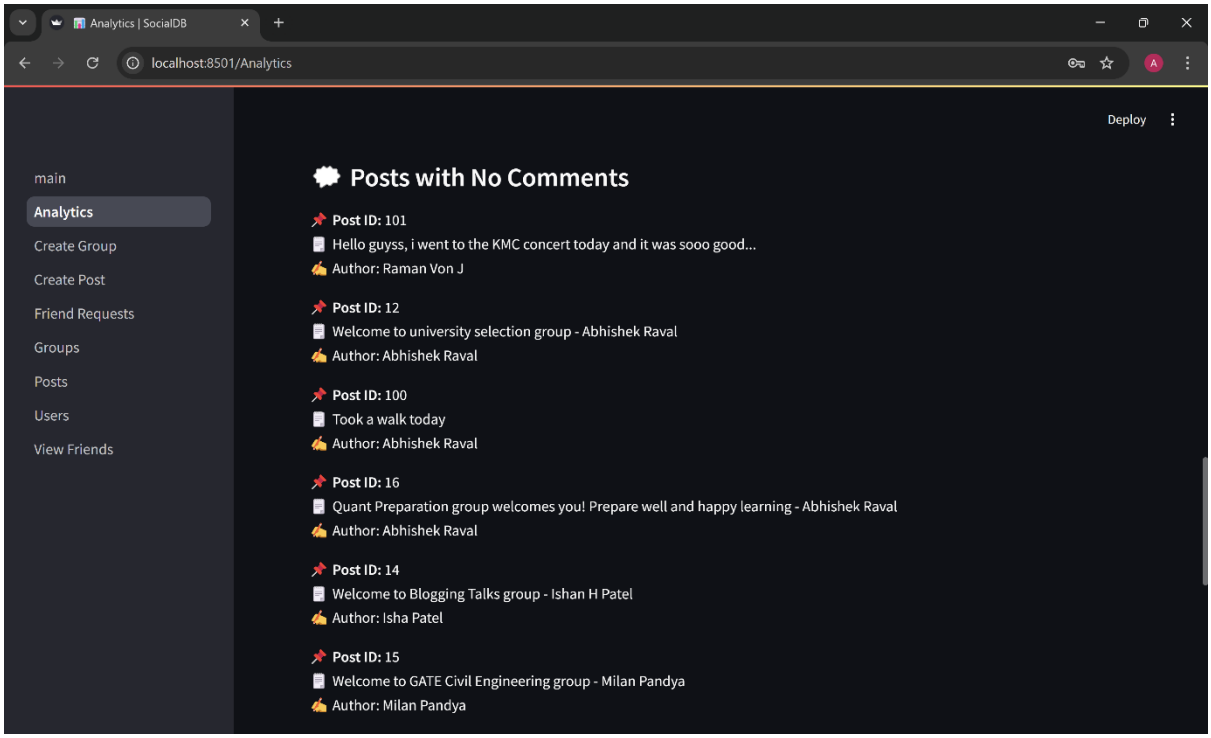
3.a: User Posts Liked by Their Friends & Posts Liked by User



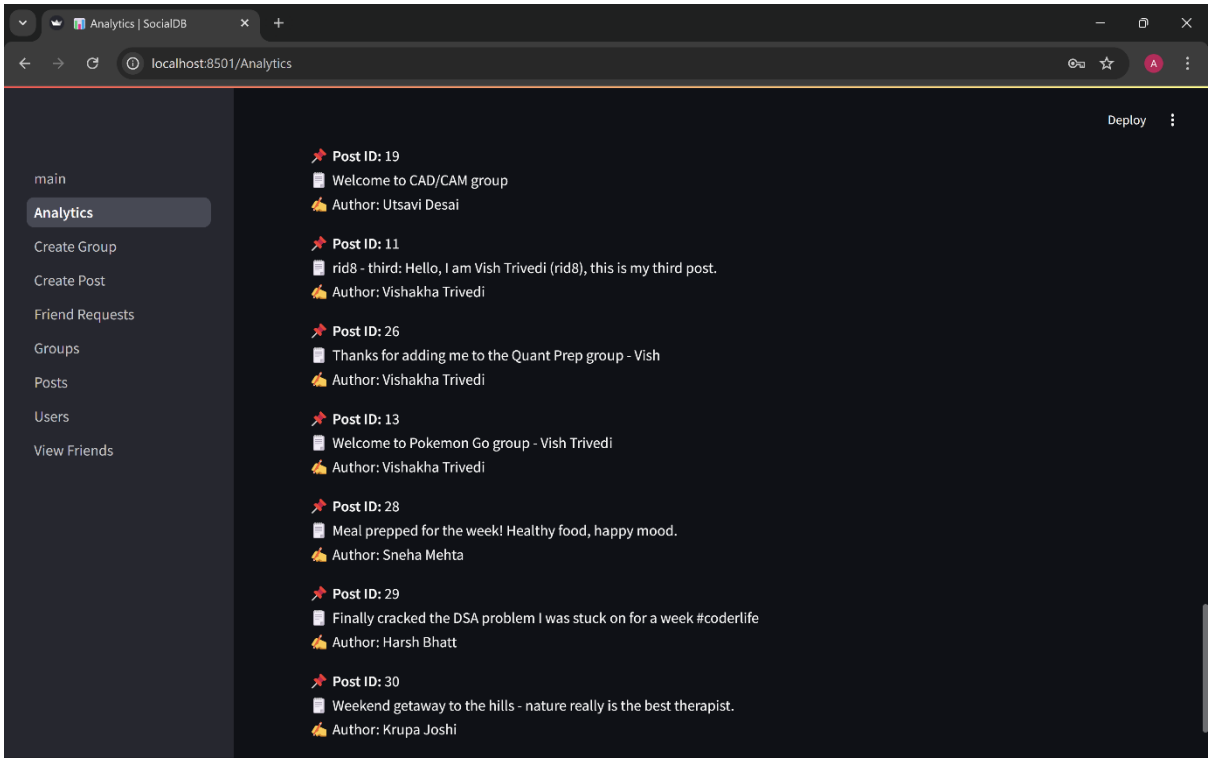
3.b: Posts with No Likes



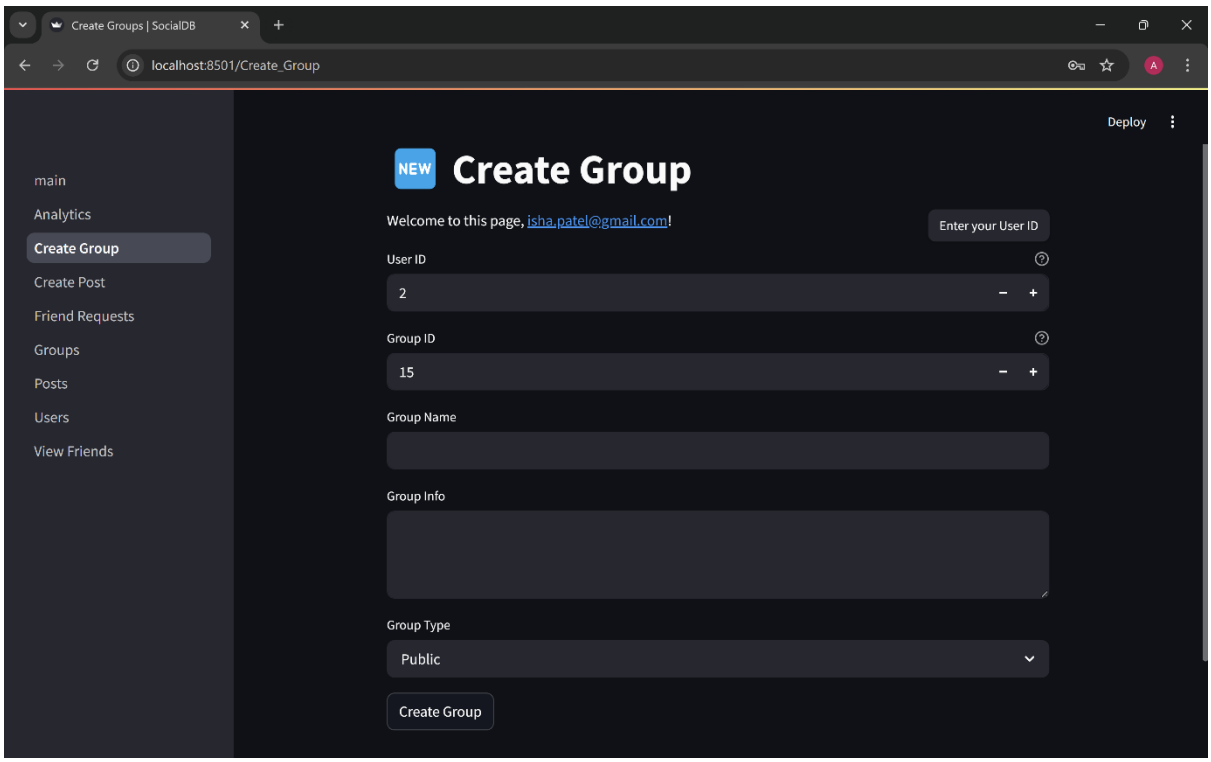
3.c: Posts Without any comments



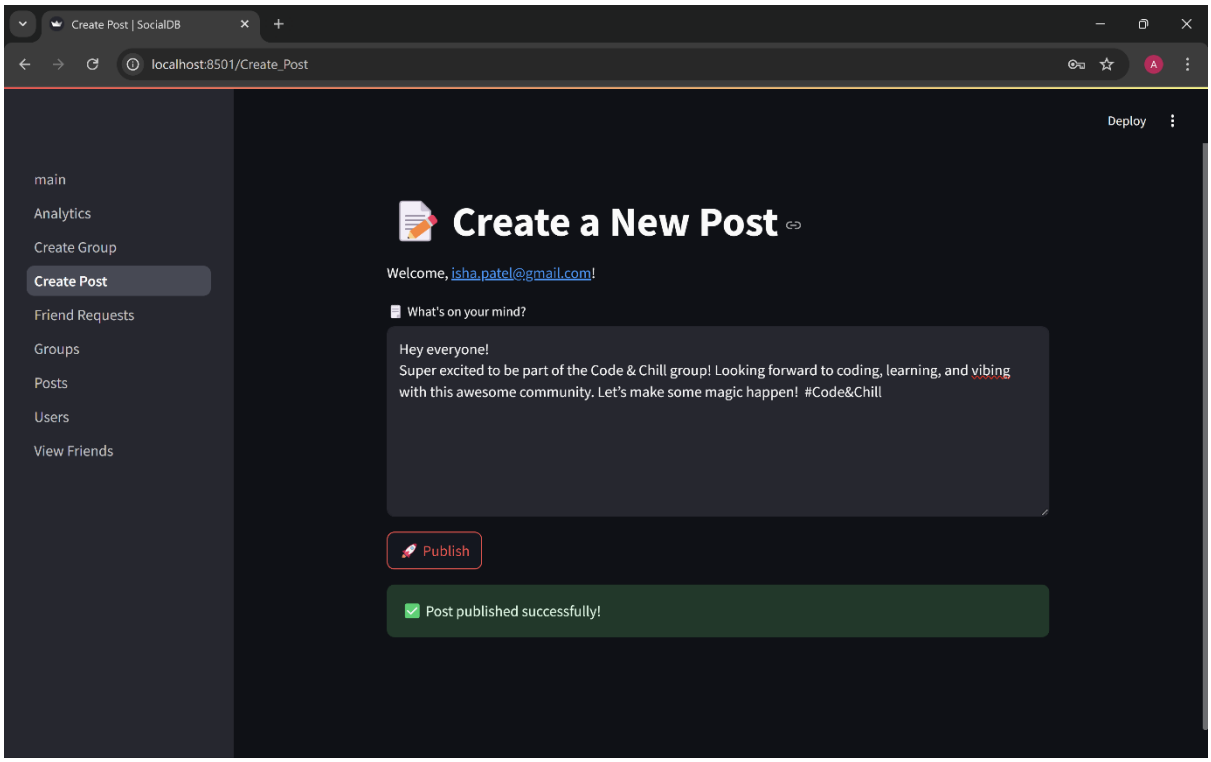
3.d: All posts uploaded by user and his friends.



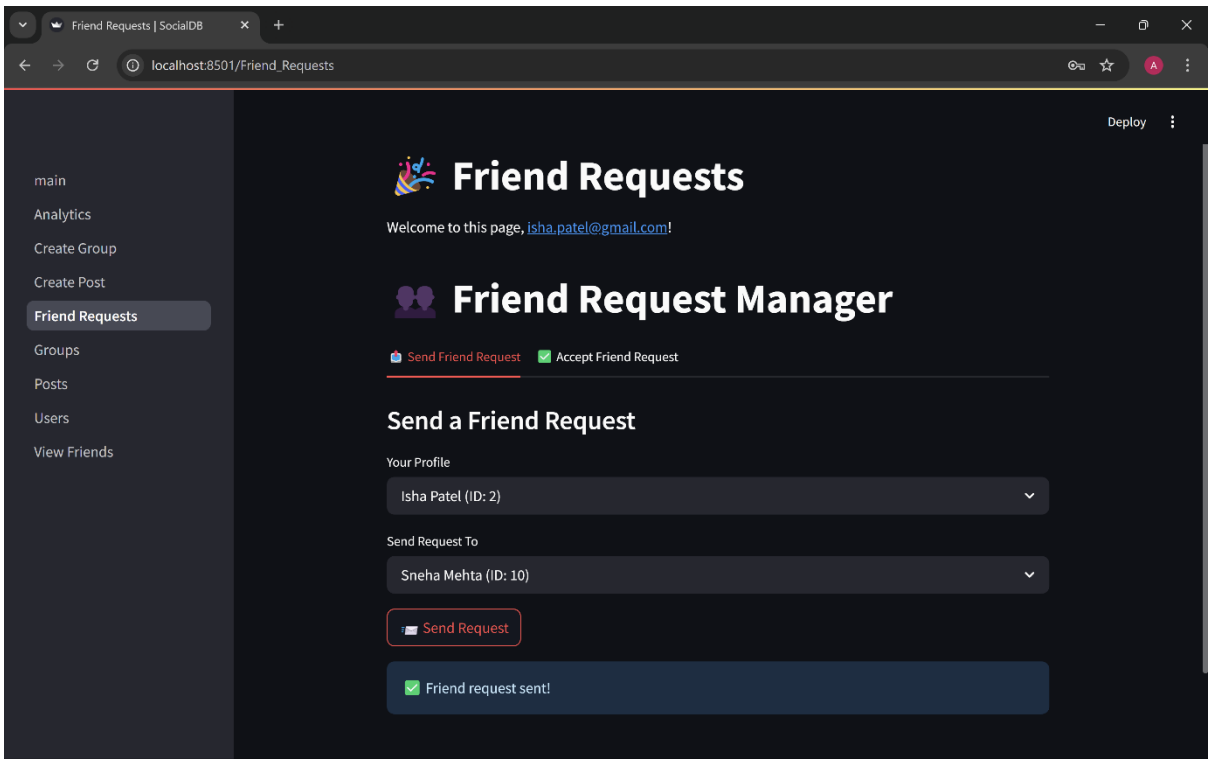
4. Create Group Page : creates group by taking the necessary information from user.



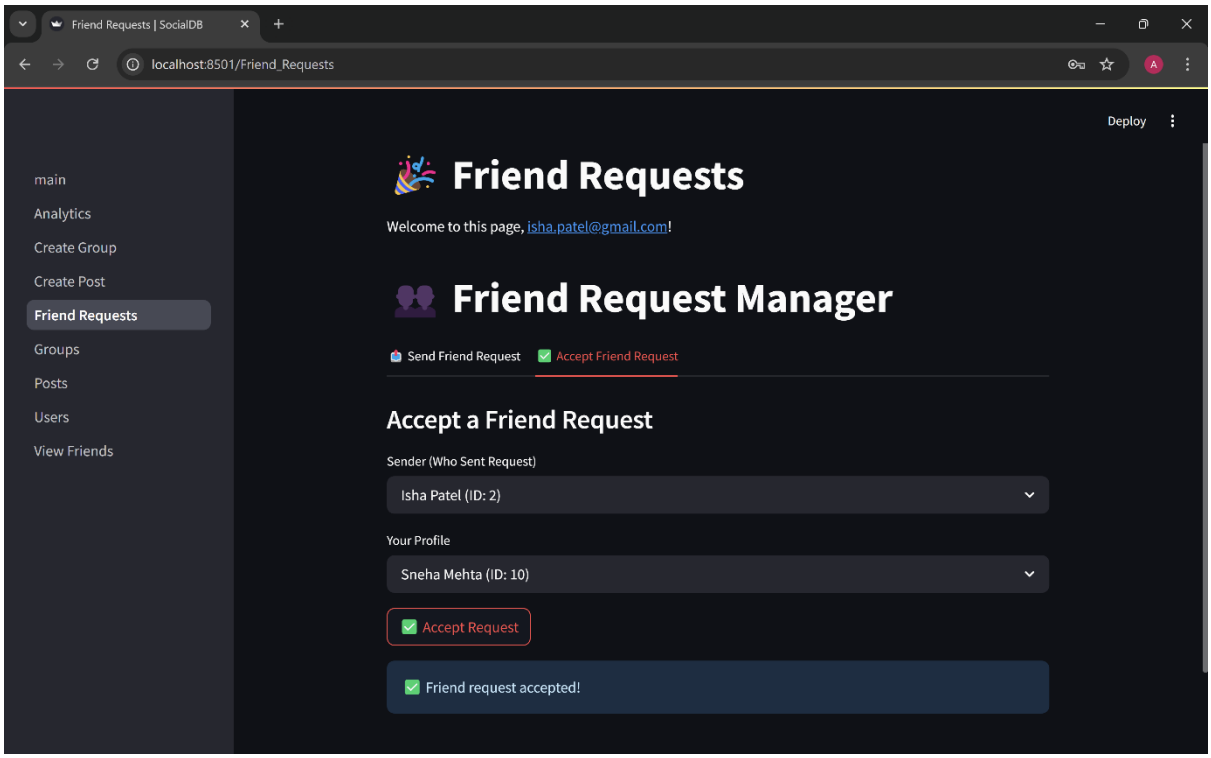
5. Create Post Page : creates a post by taking the necessary information from user and publishes it.



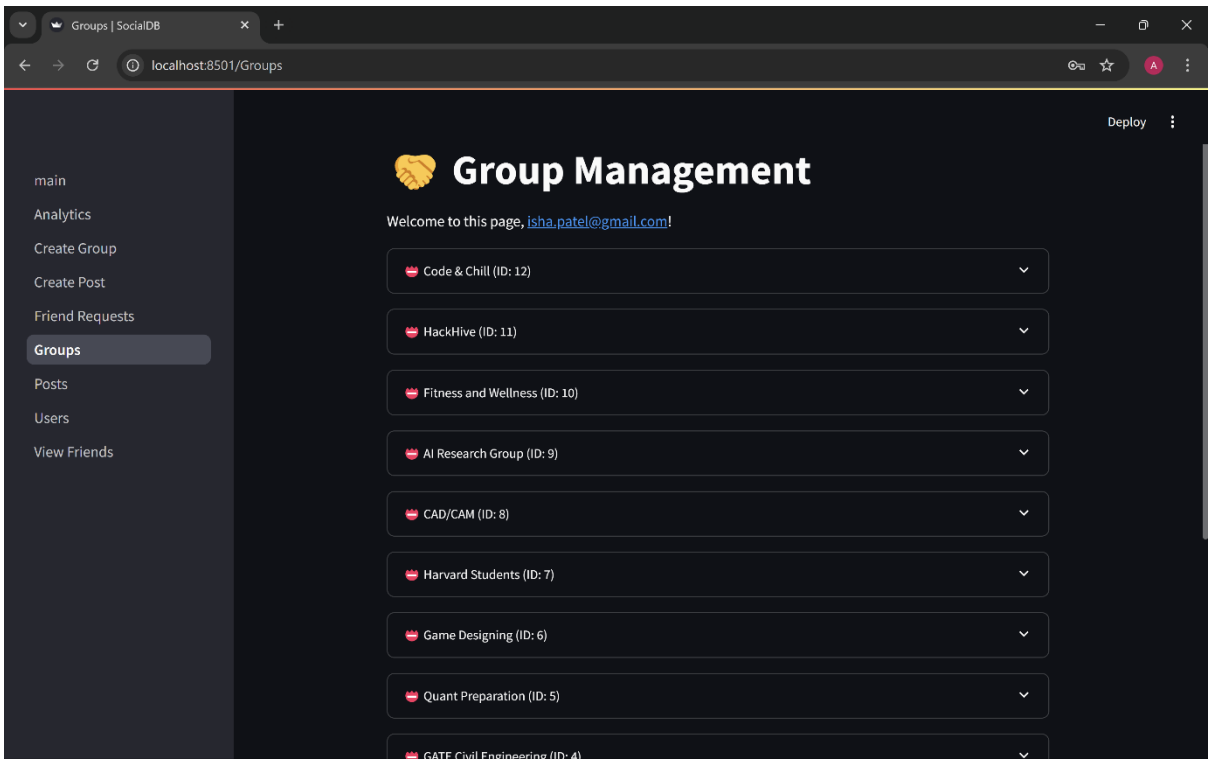
6.a: Send Request Page :Take user_id and target_id(friend's id) from user input to send a friend Request.

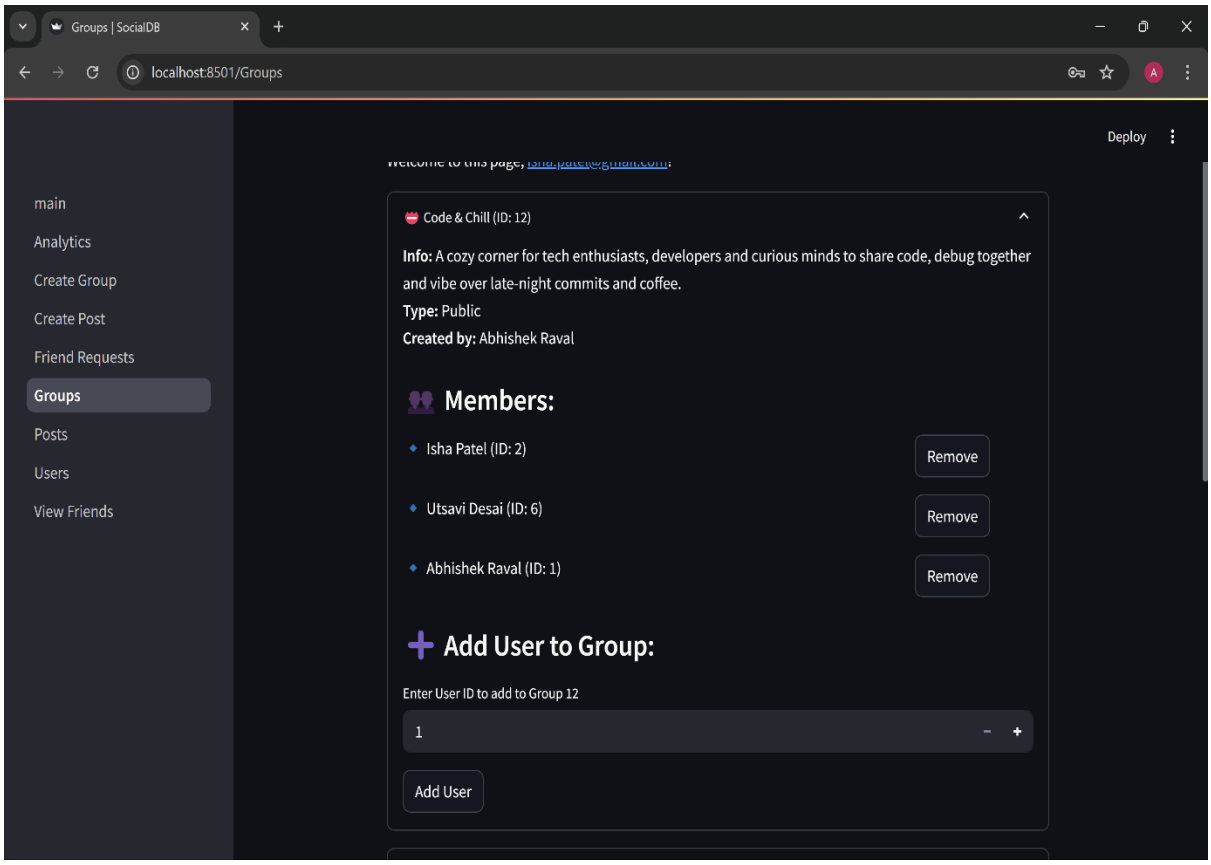


6.b:Accept Request Page :Take sender_id(who sent the request) and user_id(own id) from user input to accept a friend Request.

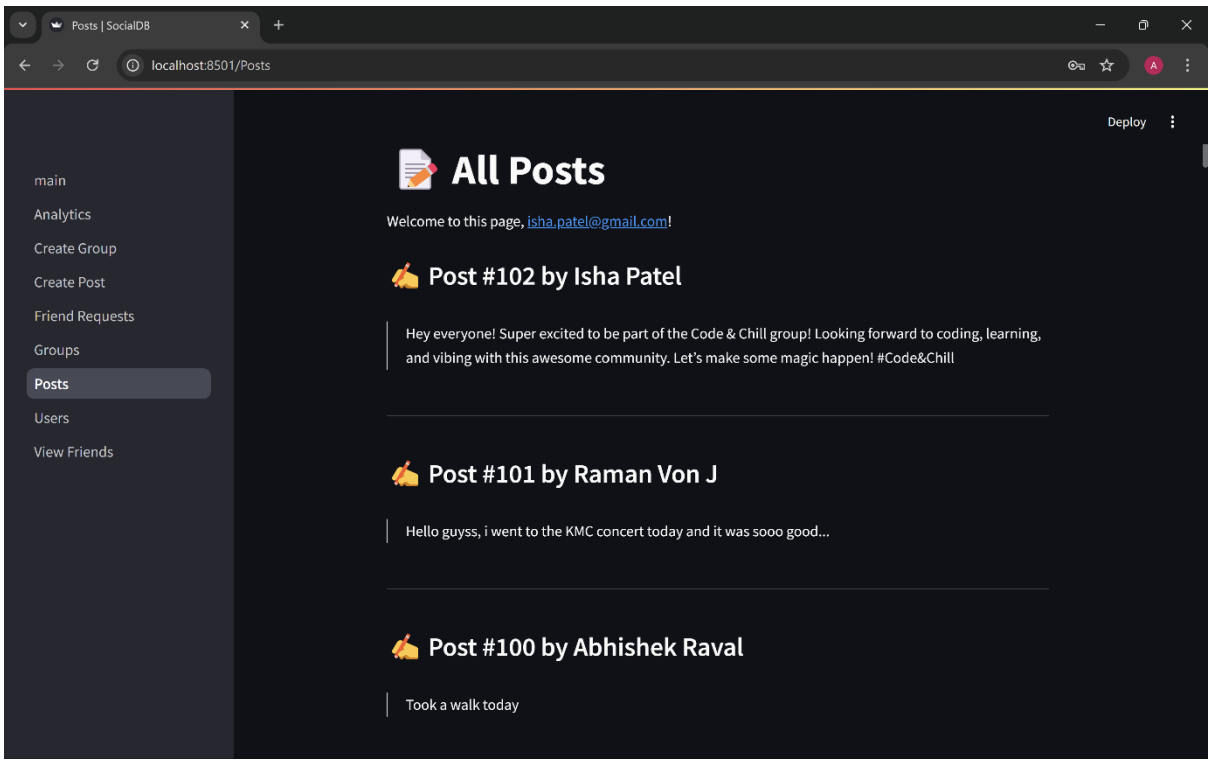


7. Group Page: it shows all the groups created till now, and shows all the necessary details of each group when click that respective group.

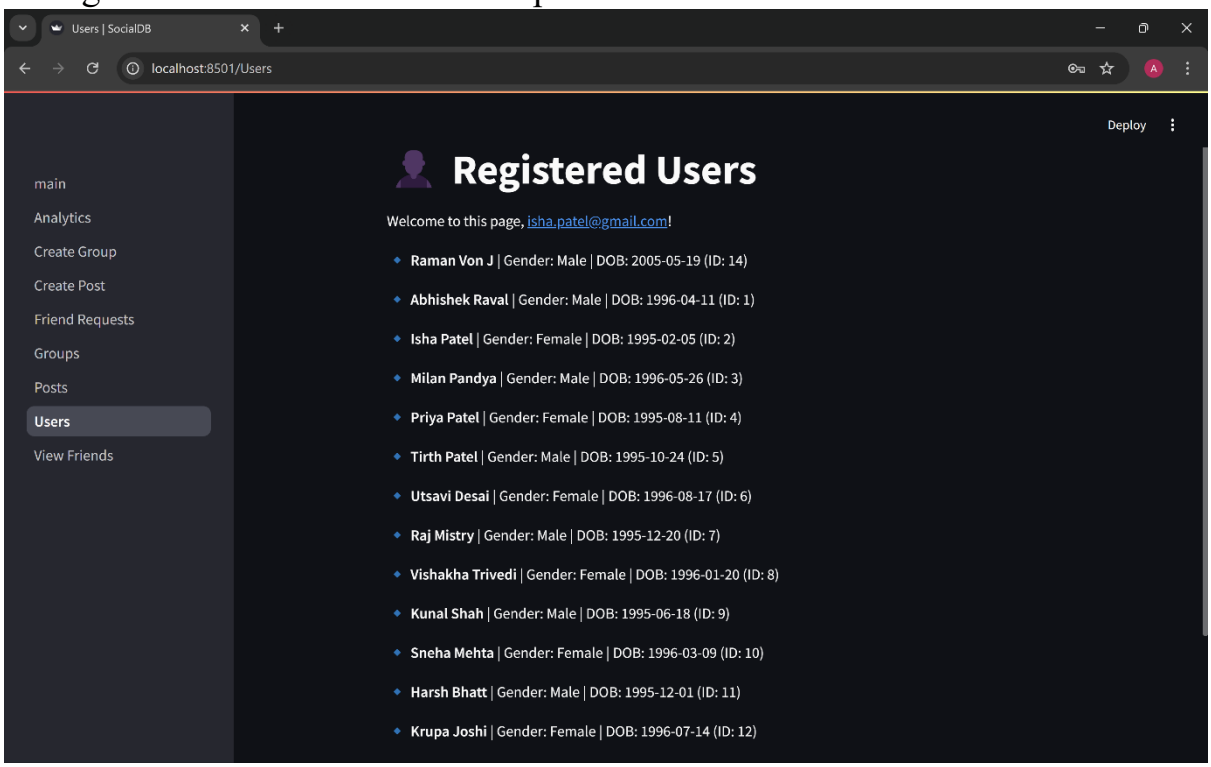




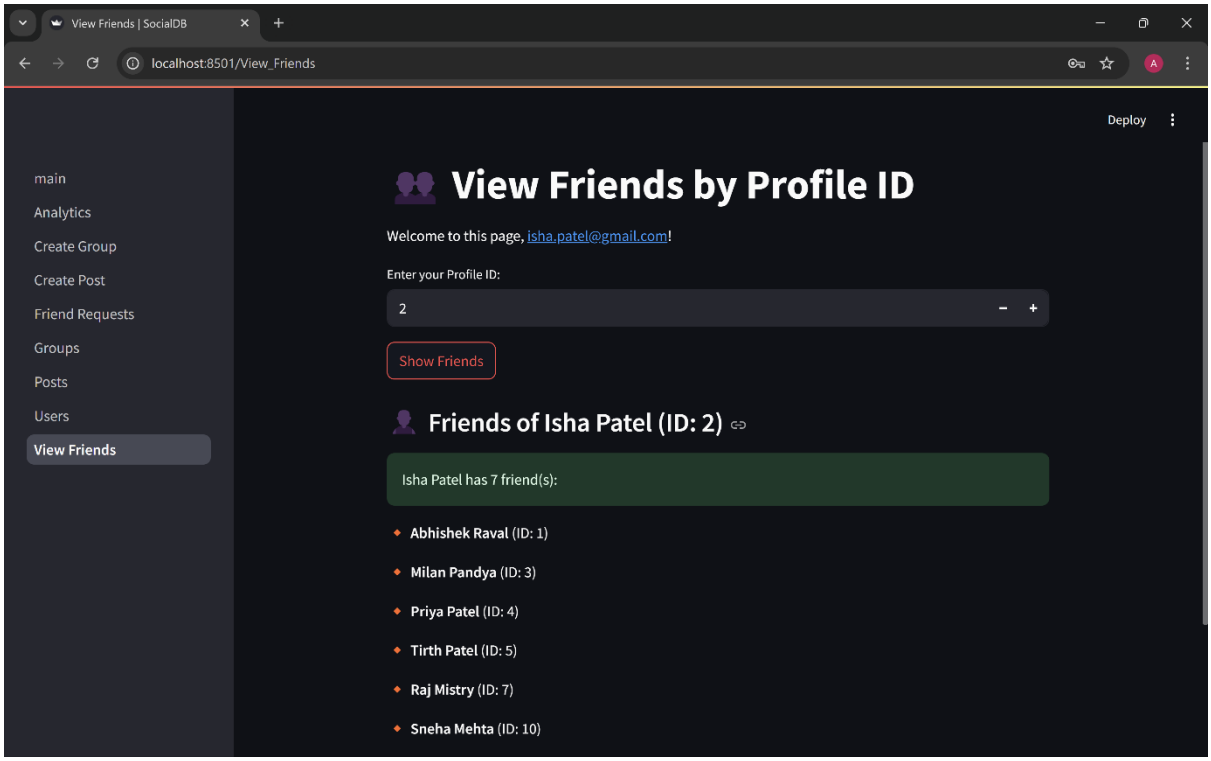
8. Posts Page: it shows all the post uploaded till now, and shows the necessary details of each post like user who uploaded and post content, post id .



9. Users Page: it shows all the users that have registered till now and it display's their gender date of birth and other personal details.



10. View Friends Page: it takes profile id as the input from user and shows all the friends that the particular person has along with friend's name and user id.


View Friends | SocialDB
localhost:8501/View_Friends
Deploy
main
Analytics
Create Group
Create Post
Friend Requests
Groups
Posts
Users
View Friends
View Friends by Profile ID
Welcome to this page, isha.patel@gmail.com!
Enter your Profile ID:
2
Show Friends
Friends of Isha Patel (ID: 2)
Isha Patel has 7 friend(s):
Abhishek Raval (ID: 1)
Milan Pandya (ID: 3)
Priya Patel (ID: 4)
Tirth Patel (ID: 5)
Raj Mistry (ID: 7)
Sneha Mehta (ID: 10)

Conclusion

This project brings to life a student-friendly social media platform designed to connect, engage, and support interaction among users in a simple and meaningful way. With features like user registration, login, posting updates, liking and commenting, managing friend requests, and joining groups—students get all the familiar social tools they need in one place.

The use of PL/SQL code helps handle these features smoothly behind the scenes, from tracking who liked a post to managing group memberships, the procedures and functions are built to be efficient, easy to understand, and adaptable for future improvements.

We used Oracle's powerful tools like cursors, custom object types, and functions, the platform not only works well but is also designed with flexibility and performance in mind.

In the end, this is a social media platform which is student friendly. This platform has created a space for students to express their views and has a means to interact with fellow peers and have fun!

Scope for future work

This project can be improved and expanded in many useful ways. In the future, we can add smart features like showing post or friend suggestions based on user interest, or checking the mood of posts using simple AI tools. We can also show trends—like who is most active or which posts are most liked—on user-friendly dashboards. For better communication, chat features like direct messaging and even voice chat options can be added so users can connect in real-time.

To make the platform safer, we can use methods like two-factor authentication (2FA) and password encryption to protect user accounts. Storing data on cloud platforms will also help in handling more users and growing the project easily. Connecting this system with real social media platforms like Twitter or Instagram could allow comparison and real-world analysis.

More pages and tools can be included to help users manage their profiles, groups, and interactions better. Students can test new features and perform data analysis. Researchers can use the platform to understand online behaviour. Overall, this system has great potential for learning, research, and even future startup development.

Contributions:

Prajjnaa Ray Choudhury	Connected frontend and backend through streamlit and assembled the website
Mudit Manas	Created Send friend request and create group pages and debugged errors
Srija Chatterjee	Created user registration and login page and debugged errors
Sai Avinash Patoju	Created View Friends and Posts page
Ashrit Reddy	Created Create Post page and Groups pages
Arnav Kumar	Created and modified report
Samarth Agrawal	Worked in create table and insertion of records, procedures and function
Anchal Mogapady	Worked in create table and insertion of records, procedures and function
Vansh Pahwa	Modified report and created Users page