

A
Project Report
on
**Social Media Database
Management System**

Developed by

**KASHYAP GORASIYA(IT-037)
GAUTAM GANDHI(IT-033)**

**Guided By
Internal Guide:
Prof. Archana N Vyas
Department of Information Technology
Faculty of Technology
DD University**



**Department of Information Technology
Faculty of Technology, Dharmasinh Desai University
College Road, Nadiad-387001
October-2019**

ACKNOWLEDGEMENT

We would like to give our sincere acknowledgement to everybody responsible for the successful completion of our project “SOCIAL MEDIA DATABASE MANAGEMENT SYSTEM”.

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of this project.

We owe our deep gratitude to our project guide Prof. Archana Vyas, who took been interest on our project work and guided us all along till the completion of our project work by providing all the necessary help for developing a good Database System.

We would also like to thank all our lecturers.

Finally we convey our acknowledgement to all our friends and family members who directly or indirectly associated with us in the successful completion of the project. We thank one and all.

TABLE OF CONTENTS

I. Certificate.....	I
II. Acknowledgement.....	II
1. SYSTEM OVERVIEW	
1.1 Current system	1
1.2 Objectives of the Proposed System	1
1.3 Advantages of the Proposed system (over current)	1
2. E-R DIAGRAM.....	6
3. DATA DICTIONARY	7
4. SCHEMA DIAGRAM.....	10
5. DATABASE IMPLEMENTATION.....	
5.1 Create Schema	11
5.2 Insert Data values	14
5.3 Queries (Based on group by, having, constrain)	24
5.4 Queries (joins, self joins, sub query).....	28
5.5 PL/SQL.....	34
5.6 Functions	35
5.7 Triggers	36
5.8 Cursors	40
6. FUTURE ENHANCEMENTS OF THE SYSTEM	41
7. BIBLIOGRAPHY	42

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT



CERTIFICATE

This is to certify that the project entitled “**Social Media Database Management System**” is a bonafide report of the work carried out by

- 1) **Mr. GANDHI GAUTAM**, Student ID No : **19ITUOF015**
- 2) **Mr. GORASIYA KASHYAP**, Student ID No : **19ITUEF014**

of Department of Information Technology, semester V, under the guidance and supervision for the subject Database Management System. They were involved in Project training during academic year 2020-2021.

Prof. Archana N Vyas.

(Project Guide)

Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

Prof. Vipul Dabhi

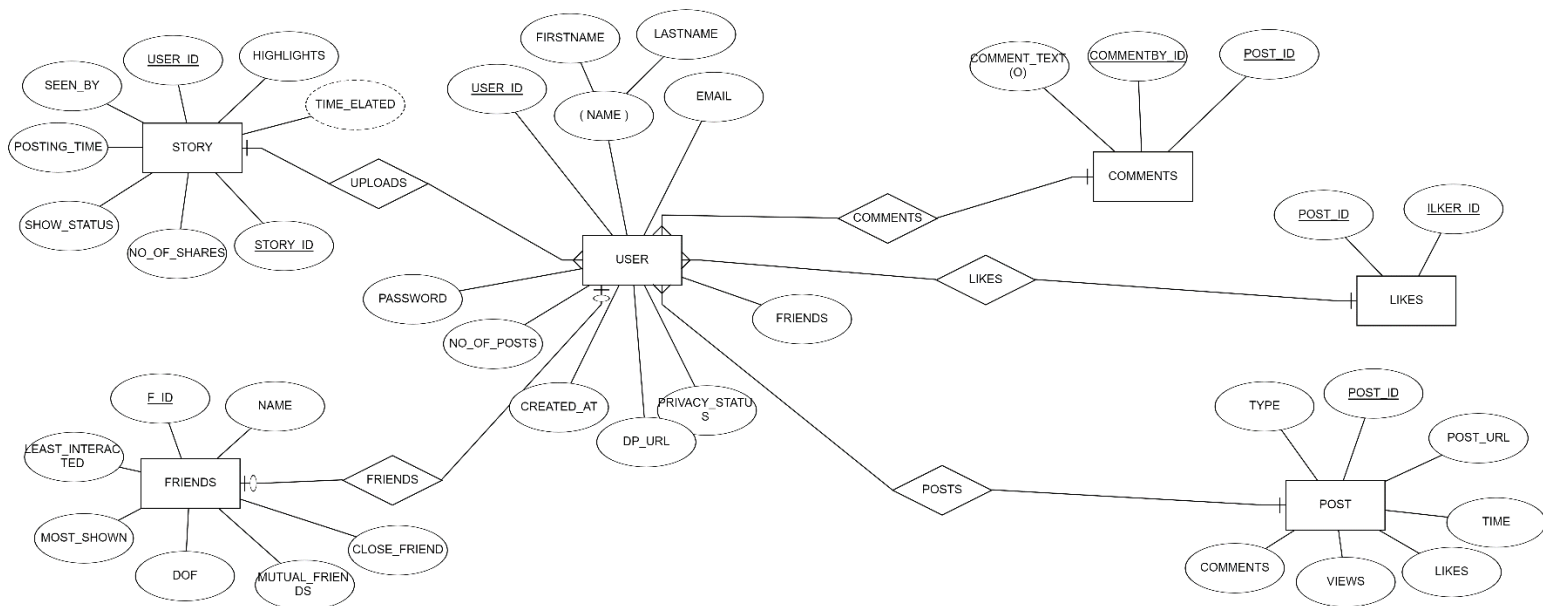
Head , Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

1. **SYSTEM OVERVIEW**

1)Social Media Database System keeps track of almost everything a real life social media deals with. (eg: Facebook).

2)It keeps the track of users, their friends ,stories, posts and much more.

3)All the basic queries as well as complex pl/sql , cursors and triggers have been implemented in order to prove the effectiveness of the database Management system.

2.E-R DIAGRAM

3. DATA DICTIONARY

3.1 USERS

```
SOCIALMEDIA=# \d users

Table "public.users"
  Column      |      Type      | Collation | Nullable |      Default
-----+-----+-----+-----+-----
 user_id      | character varying(50) |           | not null |
 password     | character varying(50) |           | not null |
 firstname    | character varying(50) |           | not null |
 lastname     | character varying(50) |           | not null |
 email        | character varying(50) |           | not null |
 dp_url       | character varying(50) |           |          | NULL::character varying
 no_of_post   | integer          |           |          | 0
 privacy_status | boolean          |           |          | false
 friends      | integer          |           |          | 0
 created_at   | timestamp without time zone |           |          | now()

Indexes:
    "users_pkey" PRIMARY KEY, btree (user_id)
    "users_email_key" UNIQUE CONSTRAINT, btree (email)

Check constraints:
    "users_email_check" CHECK (email::text ~~ '%__@__%.__%'::text)

Referenced by:
    TABLE "story" CONSTRAINT "fk10" FOREIGN KEY (user_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "friends" CONSTRAINT "fk3" FOREIGN KEY (user_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "friends" CONSTRAINT "fk4" FOREIGN KEY (friend_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "post" CONSTRAINT "fk5" FOREIGN KEY (user_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "likes" CONSTRAINT "fk7" FOREIGN KEY (liker_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "comments" CONSTRAINT "fk9" FOREIGN KEY (commentby_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
```

3.2 FRIENDS

```
SOCIALMEDIA=# \d friends

Table "public.friends"
  Column      |      Type      | Collation | Nullable |      Default
-----+-----+-----+-----+-----
 user_id      | character varying(50) |           | not null |
 friend_id    | character varying(50) |           | not null |
 f_name       | character varying(50) |           | not null |
 close_friend | boolean          |           |          | false
 date_of_following | date            |           | not null |
 mutual_friends | integer          |           |          | 0
 most_shown   | boolean          |           |          |
 least_interacted | boolean          |           |          |

Indexes:
    "friends_pkey" PRIMARY KEY, btree (user_id, friend_id)

Foreign-key constraints:
    "fk3" FOREIGN KEY (user_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
    "fk4" FOREIGN KEY (friend_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
```

3.3 POST

```
SOCIALMEDIA=# \d post
          Table "public.post"
  Column |          Type          | Collation | Nullable |          Default
-----+-----+-----+-----+-----
 post_id | character varying(50)  |           | not null |
 user_id | character varying(50)  |           | not null |
 post_url | character varying(50)  |           |          | NULL::character varying
 posting_time | timestamp without time zone |           |          |
 likes   | integer                |           |          | 0
 view    | integer                |           |          | 0
 comments | integer                |           |          | 0
 type    | character varying(9)   |           |          |
Indexes:
    "post_pkey" PRIMARY KEY, btree (post_id)
Check constraints:
    "post_type_check" CHECK (type::text = ANY (ARRAY['photo'::character varying, 'video'::character varying, 'audio'::character varying, 'story'::character varying]::text[]))
Foreign-key constraints:
    "fk5" FOREIGN KEY (user_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
Referenced by:
    TABLE "likes" CONSTRAINT "fk6" FOREIGN KEY (post_id) REFERENCES post(post_id) ON UPDATE CASCADE ON DELETE CASCADE
    TABLE "comments" CONSTRAINT "fk8" FOREIGN KEY (post_id) REFERENCES post(post_id) ON UPDATE CASCADE ON DELETE CASCADE
```

3.4 LIKES

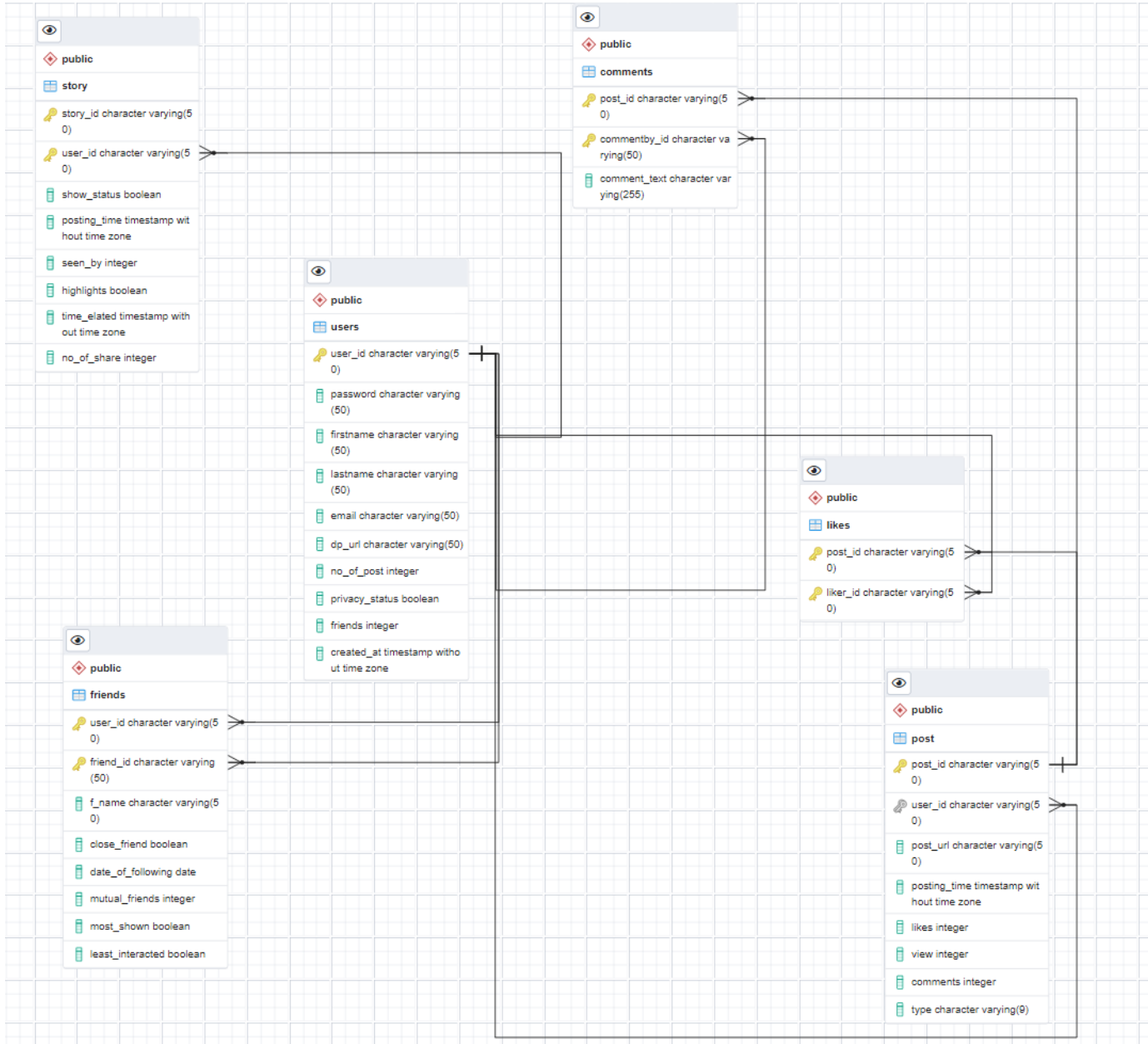
```
SOCIALMEDIA=# \d likes
          Table "public.likes"
  Column |          Type          | Collation | Nullable |          Default
-----+-----+-----+-----+-----
 post_id | character varying(50)  |           | not null |
 liker_id | character varying(50)  |           | not null |
Indexes:
    "likes_pkey" PRIMARY KEY, btree (post_id, liker_id)
Foreign-key constraints:
    "fk6" FOREIGN KEY (post_id) REFERENCES post(post_id) ON UPDATE CASCADE ON DELETE CASCADE
    "fk7" FOREIGN KEY (liker_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
```


3.5 COMMENT

```
SOCIALMEDIA=# \d comments
Table "public.comments"
  Column      |          Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
post_id       | character varying(50)  |           | not null |
commentby_id  | character varying(50)  |           | not null |
comment_text  | character varying(255) |           |          |
Indexes:
    "comments_pkey" PRIMARY KEY, btree (post_id, commentby_id)
Foreign-key constraints:
    "fk8" FOREIGN KEY (post_id) REFERENCES post(post_id) ON UPDATE CASCADE ON DELETE CASCADE
    "fk9" FOREIGN KEY (commentby_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
```

3.6 STORY

```
SOCIALMEDIA=# \d story
Table "public.story"
  Column      |          Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
story_id      | character varying(50)  |           | not null |
user_id       | character varying(50)  |           | not null |
show_status   | boolean                 |           |          | false
posting_time  | timestamp without time zone |         |          |
seen_by       | integer                 |           |          | 0
highlights    | boolean                 |           |          | false
time_elated   | timestamp without time zone |         |          |
no_of_share   | integer                 |           |          | 0
Indexes:
    "story_pkey" PRIMARY KEY, btree (user_id, story_id)
Foreign-key constraints:
    "fk10" FOREIGN KEY (user_id) REFERENCES users(user_id) ON UPDATE CASCADE ON DELETE CASCADE
```

4.**SCHEMA DIAGRAM**

5. DATABASE IMPLEMENTATION

5.1 CREATE SCHEMA

5.1.1 USERS

```
CREATE TABLE users (  
    user_id VARCHAR(50) NOT NULL,  
    password VARCHAR(50) NOT NULL,  
    firstname VARCHAR(50) NOT NULL,  
    lastname VARCHAR(50) NOT NULL,  
    email VARCHAR(50) CHECK(email LIKE '%____@____%.____%') UNIQUE NOT NULL,  
    dp_url VARCHAR(50) DEFAULT NULL,  
    no_of_post INT DEFAULT 0,  
    privacy_status BOOLEAN DEFAULT FALSE,  
    friends INT DEFAULT 0,  
    created_at TIMESTAMP DEFAULT NOW(),  
  
    PRIMARY KEY(user_id)  
  
);
```

5.1.2 FRIENDS

```
CREATE TABLE friends(  
    user_id VARCHAR(50) NOT NULL,  
    friend_id VARCHAR(50) NOT NULL,  
    f_name VARCHAR(50) NOT NULL,  
    close_friend BOOLEAN DEFAULT FALSE,  
    date_of_following DATE NOT NULL,  
    mutual_friends INT DEFAULT 0,  
    most_shown BOOLEAN,  
    least_interacted BOOLEAN,  
  
    PRIMARY KEY(user_id, friend_id),  
    CONSTRAINT fk3 FOREIGN KEY(user_id) REFERENCES users(user_id) ON DELETE CASCADE ON UPDATE  
CASCADE,  
    CONSTRAINT fk4 FOREIGN KEY(friend_id) REFERENCES users(user_id) ON DELETE CASCADE ON UPDATE  
CASCADE  
  
);
```

5.1.3 POST

```
CREATE TABLE post(  
    post_id VARCHAR(50) NOT NULL ,  
    user_id VARCHAR(50) NOT NULL,  
    post_url VARCHAR(50) NULL DEFAULT NULL ,  
    posting_time TIMESTAMP ,  
    likes INT DEFAULT 0,  
    view INT DEFAULT 0,  
    comments INT DEFAULT 0,  
    type VARCHAR(9) check(type IN ('photo', 'video','audio','story')),  
    PRIMARY KEY(post_id),  
    CONSTRAINT fk5 FOREIGN KEY(user_id) REFERENCES users(user_id) ON DELETE CASCADE  
    ON UPDATE CASCADE  
  
);
```

5.1.4 LIKES

```
CREATE TABLE likes(  
    post_id VARCHAR(50) NOT NULL,  
    liker_id VARCHAR(50) NOT NULL,  
  
    PRIMARY KEY(post_id,liker_id),  
    CONSTRAINT fk6 FOREIGN KEY(post_id) REFERENCES post(post_id) ON DELETE  
    CASCADE ON UPDATE CASCADE,  
    CONSTRAINT fk7 FOREIGN KEY(liker_id) REFERENCES users(user_id) ON  
    DELETE CASCADE ON UPDATE CASCADE  
  
);
```

5.1.5 COMMENTS

```
CREATE TABLE comments(  
    post_id VARCHAR(50) NOT NULL,  
    commentby_id VARCHAR(50) NOT NULL,  
    comment_text VARCHAR(255) ,  
  
    PRIMARY KEY(post_id,commentby_id),  
    CONSTRAINT fk8 FOREIGN KEY(post_id) REFERENCES post(post_id) ON DELETE  
    CASCADE ON UPDATE CASCADE,  
    CONSTRAINT fk9 FOREIGN KEY(commentby_id) REFERENCES users(user_id) ON  
    DELETE CASCADE ON UPDATE CASCADE  
  
);
```











5.1.6 STORY

```
CREATE TABLE story(  
    story_id VARCHAR(50) NOT NULL,  
    user_id VARCHAR(50) NOT NULL,  
    show_status BOOLEAN DEFAULT FALSE,  
    posting_time TIMESTAMP ,  
    seen_by INT DEFAULT 0,  
    highlights BOOLEAN DEFAULT FALSE,  
    time_elated TIMESTAMP ,  
    no_of_share INT DEFAULT 0,  
  
    PRIMARY KEY(user_id,story_id),  
    CONSTRAINT fk10 FOREIGN KEY(user_id) REFERENCES users(user_id) ON DELETE CASCADE ON  
UPDATE CASCADE  
);
```

5.2 INSERT DATA VALUE

5.2.1 USERS

```
INSERT INTO users(user_id ,      password ,      firstname , lastname ,      email ,
                  dp_url ,      no_of_post , privacy_status , friends ,created_at )VALUES
('USR01','Pass@USR1','Raj','Sharma','rajsharma@gmail.com','IC/DP/USRDP01.png',2,FAL
SE,4,'2010-06-02 01:10:25-03'),
('USR02','Pass@USR2','Yash','Varma','yashvarma@gmail.com','IC/DP/USRDP02.png',1,TRU
E,4,'2019-07-22 05:05:25-02'),
('USR03','Pass@USR3','Pooja','Maheta','poojamaheta@gmail.com','IC/DP/USRDP03.png',n
ull,FALSE,null,'2020-08-08 09:10:25-07'),
('USR04','Pass@USR4','Riya','Kotak','riyakotak@gmail.com','IC/DP/USRDP04.png',3,TRU
E,4,'2011-05-22 10:02:25-01'),
('USR05','Pass@USR5','Parth','Bhatt','parthbhatt@gmail.com','IC/DP/USRDP05.png',nul
l,FALSE,3,'2016-06-15 02:06:25-06'),
('USR06','Pass@USR6','Alice','Shah','aliceshah@gmail.com','IC/DP/USRDP06.png',null,
TRUE,2,'2018-06-16 08:13:25-04'),
('USR07','Pass@USR7','Akash','Benarji','akashbenarji@gmail.com','IC/DP/USRDP07.png'
,1,FALSE,5,'2017-04-06 19:15:25-07'),
('USR08','Pass@USR8','Juli','Kapoor','julikapoor@gmail.com','IC/DP/USRDP08.png',1,F
ALSE,3,'2018-09-12 03:56:25-05'),
('USR09','Pass@USR9','Rahul','Agrawal','rahulagrawal@gmail.com','IC/DP/USRDP09.png'
,1,TRUE,2,'2017-06-07 04:10:25-09'),
('USR010','Pass@USR10','Kajal','Adani','kajaladani@gmail.com','IC/DP/USRDP010.png',
null,FALSE,1,'2018-06-22 17:20:25-08'),
('USR011','Pass@USR11','Pratik','Gandhi','pratikgandhi@gmail.com','IC/DP/USRDP011.p
ng',1,TRUE,null,'2019-06-04 19:57:25-07'),
('USR012','Pass@USR12','Jitu','Kumar','jitukumar@gmail.com','IC/DP/USRDP012.png',nu
ll,FALSE,1,'2019-12-25 16:27:25-07'),
('USR013','Pass@USR13','Vaibhav','Roshan','vaibhavroshan@gmail.com','IC/DP/USRDP013
.png',null,TRUE,null,'2019-10-31 18:40:25-05'),
('USR014','Pass@USR14','Minal','Gada','minalgada@gmail.com','IC/DP/USRDP014.png',2,
FALSE,null,'2017-06-22 06:51:25-04'),
('USR015','Pass@USR15','Vartika','Hathi','vartikahathi@gmail.com','IC/DP/USRDP015.p
ng',null,TRUE,2,'2016-04-30 12:10:25-04'),
('USR016','Pass@USR16','Meena','Aiyar','meenaaiyar@gmail.com','IC/DP/USRDP016.png',
null,FALSE,null,'2016-03-14 21:37:25-06'),
('USR017','Pass@USR17','Shivangi','Joshi','shivangijoshi@gmail.com','IC/DP/USRDP017
.png',2,FALSE,1,'2019-11-13 07:49:25-07'),
('USR018','Pass@USR18','Harsh','Patel','harshpatel@gmail.com','IC/DP/USRDP018.png',
1,FALSE,null,'2015-06-17 22:28:25-01'),
('USR019','Pass@USR19','Deep','Chavda','deepchavda@gmail.com','IC/DP/USRDP019.png',
null,TRUE,null,'2016-02-20 11:48:25-08'),
('USR020','Pass@USR20','Ashtha','Modi','ashthamodi@gmail.com','IC/DP/USRDP020.png',
1,FALSE,1,'2017-01-10 23:30:25-09')
;
```

	 user_id	 password	 first_name	 last_name	 email	 dp_url	 no_of_post	 privacy_status	 friends	 created_at
	[PK] character	character varying (50)	character (50)	character varying (50)	character varying (50)	character varying (50)	integer	boolean	integer	timestamp without time zone
1	USR01	Pass@USR1	Raj	Sharma	rajsharma@gmail.com	IC/DP/USRDP01.png	2	false		2010-06-02 01:10:25
2	USR010	Pass@USR10	Kajal	Adani	kajaladani@gmail.com	IC/DP/USRDP010.png	[null]	false	1	2018-06-22 17:20:25
3	USR011	Pass@USR11	Pratik	Gandhi	pratikgandhi@gmail.com	IC/DP/USRDP011.png	1	true	[null]	2019-06-04 19:57:25
4	USR012	Pass@USR12	Jitu	Kumar	jitikumar@gmail.com	IC/DP/USRDP012.png	[null]	false	1	2019-12-25 16:27:25
5	USR013	Pass@USR13	Vaibhav	Roshan	vaibhavroshan@gmail.com	IC/DP/USRDP013.png	[null]	true	[null]	2019-10-31 18:40:25
6	USR014	Pass@USR14	Minal	Gada	minalgada@gmail.com	IC/DP/USRDP014.png	2	false	[null]	2017-06-22 06:51:25
7	USR015	Pass@USR15	Vartika	Hathi	vartikahathi@gmail.com	IC/DP/USRDP015.png	[null]	true	2	2016-04-30 12:10:25
8	USR016	Pass@USR16	Meena	Aliyar	meenaaliyar@gmail.com	IC/DP/USRDP016.png	[null]	false	[null]	2016-03-14 21:37:25
9	USR017	Pass@USR17	Shivangi	Joshi	shivangijoshi@gmail.com	IC/DP/USRDP017.png	2	false	1	2019-11-13 07:49:25
10	USR018	Pass@USR18	Harsh	Patel	harshpatel@gmail.com	IC/DP/USRDP018.png	1	false	[null]	2015-06-17 22:28:25
11	USR019	Pass@USR19	Deep	Chavda	deepchavda@gmail.com	IC/DP/USRDP019.png	[null]	true	[null]	2016-02-20 11:48:25
12	USR02	Pass@USR2	Yash	Varma	yashvarma@gmail.com	IC/DP/USRDP02.png	1	true	4	2019-07-22 05:05:25
13	USR020	Pass@USR20	Ashtha	Modi	ashthamodi@gmail.com	IC/DP/USRDP020.png	1	false	1	2017-01-10 23:30:25
14	USR03	Pass@USR3	Pooja	Maheta	poojamaheta@gmail.com	IC/DP/USRDP03.png	[null]	false	[null]	2020-08-08 09:10:25
15	USR04	Pass@USR4	Riya	Kotak	riyakotak@gmail.com	IC/DP/USRDP04.png	3	true	4	2011-05-22 10:02:25
16	USR05	Pass@USR5	Parth	Bhatt	parthbhatt@gmail.com	IC/DP/USRDP05.png	[null]	false	3	2016-06-15 02:06:25
17	USR06	Pass@USR6	Alice	Shah	aliceshah@gmail.com	IC/DP/USRDP06.png	[null]	true	2	2018-06-16 08:13:25
18	USR07	Pass@USR7	Akash	Benarji	akashbenarji@gmail.com	IC/DP/USRDP07.png	1	false	5	2017-04-06 19:15:25
19	USR08	Pass@USR8	Juli	Kapoor	julikapoor@gmail.com	IC/DP/USRDP08.png	1	false	3	2018-09-12 03:56:25
20	USR09	Pass@USR9	Rahul	Agrawal	rahulagrawal@gmail.com	IC/DP/USRDP09.png	1	true	2	2017-06-07 04:10:25

5.2.2 FRIENDS

```
INSERT INTO friends(      user_id ,      friend_id , f_name ,      close_friend ,
      date_of_following , mutual_friends , most_shown , least_interacted)VALUES
('USR01','USR02','Yash Varma',TRUE,'2019-08-23',1,FALSE,TRUE),
('USR01','USR020','Ashtha Modi',FALSE,'2018-09-02',0,FALSE,TRUE),
('USR01','USR012','Jitu Kumar',FALSE,'2020-01-02',1,TRUE,FALSE),
('USR01','USR04','Riya Kotak',TRUE,'2012-03-15',1,TRUE,FALSE),

('USR02','USR01','Raj Sharma',TRUE,'2019-10-30',1,TRUE,FALSE),
('USR02','USR03','Pooja Maheta',FALSE,'2020-12-16',0,FALSE,TRUE),
('USR02','USR04','Riya Kotak',TRUE,'2020-06-07',0,FALSE,TRUE),
('USR02','USR010','Kajal Adani',FALSE,'2021-04-05',0,FALSE,TRUE),

('USR04','USR02','Yash Varma',FALSE,'2020-06-07',0,FALSE,TRUE),
('USR04','USR05','Parth Bhatt',FALSE,'2017-07-17',0,FALSE,TRUE),
('USR04','USR08','Parth Bhatt',FALSE,'2019-09-19',0,TRUE,FALSE),
('USR04','USR07','Akash Benarji',TRUE,'2018-12-02',1,TRUE,FALSE),

('USR05','USR01','Raj Sharma',FALSE,'2016-06-22',1,FALSE,TRUE),
('USR05','USR04','Riya Kotak',FALSE,'2016-06-22',0,FALSE,TRUE),
('USR05','USR010','Kajal Adani',TRUE,'2019-01-30',0,TRUE,FALSE),

('USR06','USR09','Rahul Agrawal',FALSE,'2018-09-19',0,TRUE,FALSE),
('USR06','USR011','Pratik Gandhi',FALSE,'2020-10-20',0,FALSE,TRUE),

('USR07','USR02','Yash Varma',TRUE,'2019-10-10',0,TRUE,FALSE),
('USR07','USR012','Jitu Kumar',TRUE,'2020-04-06',1,FALSE,TRUE),
('USR07','USR013','Vaibhav Roshan',FALSE,'2021-08-12',0,TRUE,FALSE),
('USR07','USR018','Kajal Adani',FALSE,'2017-04-06',1,FALSE,TRUE),
('USR07','USR020','Ashtha Modi',TRUE,'2017-11-02',0,FALSE,TRUE),

('USR08','USR01','Raj Sharma',FALSE,'2018-06-22',1,TRUE,FALSE),
('USR08','USR04','Riya Kotak',FALSE,'2019-09-09',0,FALSE,TRUE),
('USR08','USR016','Meena Aiyar',TRUE,'2020-06-20',0,FALSE,TRUE),

('USR09','USR05','Parth Bhatt',TRUE,'2016-06-06',0,TRUE,FALSE),
('USR09','USR014','Mina Gada',TRUE,'2018-06-16',0,TRUE,FALSE),

('USR010','USR06','Alice Shah',TRUE,'2020-01-01',0,TRUE,FALSE),

('USR012','USR02','Yash Varma',TRUE,'2019-12-25',0,FALSE,TRUE),

('USR015','USR010','Kajal Adani',TRUE,'2019-05-22',0,TRUE,FALSE),
('USR015','USR09','Rahul Agrawal',FALSE,'2019-05-25',0,TRUE,FALSE),

('USR017','USR05','Parth Bhatt',TRUE,'2019-10-19',0,TRUE,FALSE),

('USR020','USR018','Harsh Patel',TRUE,'2017-02-20',0,FALSE,TRUE)
;
```


	user_id [PK] character varying (50)	friend_id [PK] character varying (50)	f_name character varying (50)	close_friend boolean	date_of_following date	mutual_friends integer	most_shown boolean	least_interacted boolean
1	USR01	USR012	Jitu Kumar	false	2020-01-02		1 true	false
2	USR01	USR02	Yash Varma	true	2019-08-23		1 false	true
3	USR01	USR020	Ashtha Modi	false	2018-09-02		0 false	true
4	USR01	USR04	Riya Kotak	true	2012-03-15		1 true	false
5	USR010	USR06	Alice Shah	true	2020-01-01		0 true	false
6	USR012	USR02	Yash Varma	true	2019-12-25		0 false	true
7	USR015	USR010	Kajal Adani	true	2019-05-22		0 true	false
8	USR015	USR09	Rahul Agrawal	false	2019-05-25		0 true	false
9	USR017	USR05	Parth Bhatt	true	2019-10-19		0 true	false
10	USR02	USR01	Raj Sharma	true	2019-10-30		1 true	false
11	USR02	USR010	Kajal Adani	false	2021-04-05		0 false	true
12	USR02	USR03	Pooja Maheta	false	2020-12-16		0 false	true
13	USR02	USR04	Riya Kotak	true	2020-06-07		0 false	true
14	USR020	USR018	Harsh Patel	true	2017-02-20		0 false	true
15	USR04	USR02	Yash Varma	false	2020-06-07		0 false	true
16	USR04	USR05	Parth Bhatt	false	2017-07-17		0 false	true
17	USR04	USR07	Akash Benarji	true	2018-12-02		1 true	false
18	USR04	USR08	Parth Bhatt	false	2019-09-19		0 true	false
19	USR05	USR01	Raj Sharma	false	2016-06-22		1 false	true
20	USR05	USR010	Kajal Adani	true	2019-01-30		0 true	false
21	USR05	USR04	Riya Kotak	false	2016-06-22		0 false	true
22	USR06	USR011	Pratik Gandhi	false	2020-10-20		0 false	true
23	USR06	USR09	Rahul Agrawal	false	2018-09-19		0 true	false
24	USR07	USR012	Jitu Kumar	true	2020-04-06		1 false	true
25	USR07	USR013	Vaibhav Roshan	false	2021-08-12		0 true	false
26	USR07	USR018	Kajal Adani	false	2017-04-06		1 false	true
27	USR07	USR02	Yash Varma	true	2019-10-10		0 true	false
28	USR07	USR020	Ashtha Modi	true	2017-11-02		0 false	true
29	USR08	USR01	Raj Sharma	false	2018-06-22		1 true	false
30	USR08	USR016	Meena Aiyar	true	2020-06-20		0 false	true
31	USR08	USR04	Riya Kotak	false	2019-09-09		0 false	true
32	USR09	USR014	Mina Gada	true	2018-06-16		0 true	false
33	USR09	USR05	Parth Bhatt	true	2016-06-06		0 true	false

5.2.3 POST

```
INSERT INTO post( post_id ,   user_id ,   post_url , posting_time ,
                likes ,      view ,      comments , type )VALUES
('USR01PST01','USR01','IC/POST/POSTUSR01/POST01USR01.png','2011-05-02
01:10:25-03',1,8,1,'photo'),
('USR01PST02','USR01','IC/POST/POSTUSR01/POST02USR01.png','2020-03-20
10:00:25-03',2,7,0,'video'),

('USR02PST01','USR02','IC/POST/POSTUSR02/POST01USR01.png','2019-07-23
05:01:03-02',1,6,1,'photo'),

('USR04PST01','USR04','IC/POST/POSTUSR04/POST01USR04.png','2012-05-22
11:059:20-01',1,5,0,'photo'),
('USR04PST02','USR04','IC/POST/POSTUSR04/POST02USR04.png','2015-11-02
12:010:05-01',2,4,1,'audio'),
('USR04PST03','USR04','IC/POST/POSTUSR04/POST03USR04.png','2019-12-20
13:02:00-01',1,3,0,'video'),

('USR07PST01','USR07','IC/POST/POSTUSR07/POST01USR07.png','2017-09-01
23:50:25-07',2,2,2,'photo'),

('USR08PST01','USR08','IC/POST/POSTUSR08/POST01USR08.png','2019-09-12
03:06:00-05',1,1,0,'photo'),

('USR09PST01','USR09','IC/POST/POSTUSR09/POST01USR09.png','2018-09-04
02:10:56-09',2,8,1,'photo'),

('USR11PST01','USR011','IC/POST/POSTUSR11/POST01USR11.png','2019-12-25
10:52:25-07',1,4,0,'video'),

('USR14PST01','USR014','IC/POST/POSTUSR14/POST01USR14.png','2018-06-22
06:52:26-04',2,7,1,'audio'),
('USR14PST02','USR014','IC/POST/POSTUSR14/POST02USR14.png','2020-06-22
06:50:27-04',1,8,0,'photo'),

('USR17PST01','USR017','IC/POST/POSTUSR17/POST01USR17.png','2019-12-25
10:52:45-07',1,9,1,'video'),
('USR17PST02','USR017','IC/POST/POSTUSR17/POST02USR17.png','2021-08-22
19:10:55-07',2,2,0,'photo'),

('USR18PST01','USR018','IC/POST/POSTUSR18/POST01USR18.png','2016-03-04
18:30:15-08',1,1,1,'video'),

('USR20PST01','USR020','IC/POST/POSTUSR20/POST01USR20.png','2017-09-10
07:30:25-09',3,1,0,'audio')
;
```

	post_id [PK] character varying (50)	user_id character varying (50)	post_url character varying (50)	posting_time timestamp without time zone	likes integer	view integer	comments integer	type character varying (9)
1	USR01PST01	USR01	IC/POST/POSTUSR01/POST01USR01.png	2011-05-02 01:10:25	1	8	1	photo
2	USR01PST02	USR01	IC/POST/POSTUSR01/POST02USR01.png	2020-03-20 10:00:25	2	7	0	video
3	USR02PST01	USR02	IC/POST/POSTUSR02/POST01USR01.png	2019-07-23 05:01:03	1	6	1	photo
4	USR04PST01	USR04	IC/POST/POSTUSR04/POST01USR04.png	2012-05-22 11:59:20	1	5	0	photo
5	USR04PST02	USR04	IC/POST/POSTUSR04/POST02USR04.png	2015-11-02 12:10:05	2	4	1	audio
6	USR04PST03	USR04	IC/POST/POSTUSR04/POST03USR04.png	2019-12-20 13:02:00	1	3	0	video
7	USR07PST01	USR07	IC/POST/POSTUSR07/POST01USR07.png	2017-09-01 23:50:25	2	2	2	photo
8	USR08PST01	USR08	IC/POST/POSTUSR08/POST01USR08.png	2019-09-12 03:06:00	1	1	0	photo
9	USR09PST01	USR09	IC/POST/POSTUSR09/POST01USR09.png	2018-09-04 02:10:56	2	8	1	photo
10	USR11PST01	USR011	IC/POST/POSTUSR11/POST01USR11.png	2019-12-25 10:52:25	1	4	0	video
11	USR14PST01	USR014	IC/POST/POSTUSR14/POST01USR14.png	2018-06-22 06:52:26	2	7	1	audio
12	USR14PST02	USR014	IC/POST/POSTUSR14/POST02USR14.png	2020-06-22 06:50:27	1	8	0	photo
13	USR17PST01	USR017	IC/POST/POSTUSR17/POST01USR17.png	2019-12-25 10:52:45	1	9	1	video
14	USR17PST02	USR017	IC/POST/POSTUSR17/POST02USR17.png	2021-08-22 19:10:55	2	2	0	photo
15	USR18PST01	USR018	IC/POST/POSTUSR18/POST01USR18.png	2016-03-04 18:30:15	1	1	1	video
16	USR20PST01	USR020	IC/POST/POSTUSR20/POST01USR20.png	2017-09-10 07:30:25	3	1	0	audio

5.2.4 LIKES

```
INSERT INTO likes(    post_id ,    liker_id )VALUES
('USR01PST01','USR01'),
('USR01PST02','USR011'),
('USR01PST02','USR014'),

('USR02PST01','USR01'),

('USR04PST01','USR02'),
('USR04PST02','USR07'),
('USR04PST02','USR08'),
('USR04PST03','USR05'),

('USR07PST01','USR011'),
('USR07PST01','USR012'),

('USR08PST01','USR013'),

('USR09PST01','USR014'),
('USR09PST01','USR05'),

('USR11PST01','USR016'),

('USR14PST01','USR017'),
('USR14PST01','USR018'),
('USR14PST02','USR015'),

('USR17PST01','USR016'),
('USR17PST02','USR01'),
('USR17PST02','USR017'),

('USR18PST01','USR017'),

('USR20PST01','USR019'),
('USR20PST01','USR09'),
('USR20PST01','USR015')
;
```

	post_id [PK] character varying (50)	 liker_id [PK] character varying (50)
1	USR01PST01	USR01
2	USR01PST02	USR011
3	USR01PST02	USR014
4	USR02PST01	USR01
5	USR04PST01	USR02
6	USR04PST02	USR07
7	USR04PST02	USR08
8	USR04PST03	USR05
9	USR07PST01	USR011
10	USR07PST01	USR012
11	USR08PST01	USR013
12	USR09PST01	USR014
13	USR09PST01	USR05
14	USR11PST01	USR016
15	USR14PST01	USR017
16	USR14PST01	USR018
17	USR14PST02	USR015
18	USR17PST01	USR016
19	USR17PST02	USR01
20	USR17PST02	USR017
21	USR18PST01	USR017
22	USR20PST01	USR015
23	USR20PST01	USR019
24	USR20PST01	USR09

5.2.5 COMMENT

```
INSERT INTO comments( post_id, commentby_id , comment_text )
VALUES
('USR01PST01','USR020','WOWWWWWW'),

('USR02PST01','USR04','Awesome'),

('USR04PST02','USR08','Great'),




('USR07PST01','USR09','Good'),
('USR07PST01','USR020','Superb'),

('USR09PST01','USR05','Nice'),

('USR14PST01','USR011','Alas'),

('USR17PST01','USR012','Best wishes'),

('USR18PST01','USR019','Happy Birthday')
;
```

	 post_id [PK] character varying (50)	 commentby_id [PK] character varying (50)	 comment_text character varying (255)
1	USR01PST01	USR020	WOWWWWWW
2	USR02PST01	USR04	Awesome
3	USR04PST02	USR08	Great
4	USR07PST01	USR020	Superb
5	USR07PST01	USR09	Good
6	USR09PST01	USR05	Nice
7	USR14PST01	USR011	Alas
8	USR17PST01	USR012	Best wishes
9	USR18PST01	USR019	Happy Birthday

5.2.6 STORY

```
INSERT INTO story( story_id , user_id , show_status ,
                   posting_time , seen_by , highlights, no_of_share )VALUES
('STORYUSR01','USR020',TRUE,'2021-06-22 04:10:25-07',3,TRUE,1),
('STORYUSR03','USR01',TRUE,'2021-06-22 06:10:34-07',9,TRUE,0),
('STORYUSR04','USR07',TRUE,'2021-06-22 23:10:10-07',7,FALSE,3),
('STORYUSR02','USR018',TRUE,'2021-06-22 05:10:50-07',3,FALSE,5),
('STORYUSR011','USR019',TRUE,'2021-06-22 01:10:43-07',6,TRUE,7),
('STORYUSR019','USR016',TRUE,'2021-06-22 13:10:58-07',9,TRUE,2),
('STORYUSR013','USR013',TRUE,'2021-06-22 00:00:40-07',3,FALSE,1)
;
```

	story_id [PK] character varying (50)	user_id [PK] character varying (50)	show_status boolean	posting_time timestamp without time zone	seen_by integer	highlights boolean	time_elated timestamp without time zone	no_of_share integer
1	STORYUSR01	USR020	true	2021-06-22 04:10:25	3	true	[null]	1
2	STORYUSR011	USR019	true	2021-06-22 01:10:43	6	true	[null]	7
3	STORYUSR013	USR013	true	2021-06-22 00:00:40	3	false	[null]	1
4	STORYUSR019	USR016	true	2021-06-22 13:10:58	9	true	[null]	2
5	STORYUSR02	USR018	true	2021-06-22 05:10:50	3	false	[null]	5
6	STORYUSR03	USR01	true	2021-06-22 06:10:34	9	true	[null]	0
7	STORYUSR04	USR07	true	2021-06-22 23:10:10	7	false	[null]	3

5.3 BASIC QUERIES

1) Display information of posts between 2019 and 2021.

```
select * from post where posting_time between '2019-01-01' AND '2021-12-31';
```

Query Editor									
<pre>1 --1) Display information of posts between 2019 and 2021. 2 select * from post where posting_time between '2019-01-01' AND '2021-12-31';</pre>									
Data Output Explain Query History Notifications									
	post_id [PK] character varying (50)	user_id character varying (50)	post_url character varying (50)	posting_time timestamp without time zone	likes integer	view integer	comments integer	type character varying (9)	
1	USR01PST02	USR01	IC/POST/POSTUSR01/POST...	2020-03-20 10:00:25	2	7	0	video	
2	USR02PST01	USR02	IC/POST/POSTUSR02/POST...	2019-07-23 05:01:03	1	6	1	photo	
3	USR04PST03	USR04	IC/POST/POSTUSR04/POST...	2019-12-20 13:02:00	1	3	0	video	
4	USR08PST01	USR08	IC/POST/POSTUSR08/POST...	2019-09-12 03:06:00	1	1	0	photo	
5	USR11PST01	USR11	IC/POST/POSTUSR11/POST...	2019-12-25 10:52:25	1	4	0	video	
6	USR14PST02	USR14	IC/POST/POSTUSR14/POST...	2020-06-22 06:50:27	1	8	0	photo	
7	USR17PST01	USR17	IC/POST/POSTUSR17/POST...	2019-12-25 10:52:45	1	9	1	video	
8	USR17PST02	USR17	IC/POST/POSTUSR17/POST...	2021-08-22 19:10:55	2	2	0	photo	

2) Display sum of likes from post table.

```
select sum(likes) as likes from post;
```

Query Editor		
<pre>1 --2) Display sum of likes from post table. 2 select sum(likes) as likes from post;</pre>		
Data Output Explain Query History Notifications		
	likes bigint	
1	24	

3)Display average of likes from post table.

```
select avg(likes) as likes from post;
```

Query Editor	
1	--3)Display average of likes from post table.
2	select avg(likes) as likes from post;

Data Output	Explain	Query History	Notifications
	likes		
	numeric		
1	1.5000000000000000		

4)Display average posts and email id of users.

```
select email,avg(no_of_post) from users group by email;
```

Data Output		Query Editor	
	email character varying (50)	avg numeric	
1	pratikgandhi@gmail.com	1.0000000000000000	
2	vaibhavroshan@gmail.com	[null]	
3	shivangijoshi@gmail.com	2.0000000000000000	
4	minalgada@gmail.com	2.0000000000000000	
5	deepchavda@gmail.com	[null]	
6	rajsharma@gmail.com	2.0000000000000000	
7	jitukumar@gmail.com	[null]	
8	harshpatel@gmail.com	1.0000000000000000	
9	yashvarma@gmail.com	1.0000000000000000	
10	riyakotak@gmail.com	3.0000000000000000	
11	aliceshah@gmail.com	[null]	
12	parthbhatt@gmail.com	[null]	
13	poojamaheta@gmail.com	[null]	
14	akashbenarji@gmail.com	1.0000000000000000	
15	rahulagrawal@gmail.com	1.0000000000000000	
16	ashthamodi@gmail.com	1.0000000000000000	
17	julikapoor@gmail.com	1.0000000000000000	
18	meenaaiyar@gmail.com	[null]	
19	kajaladani@gmail.com	[null]	
20	vartikahathi@gmail.com	[null]	

Explain	Query History	Notifications
Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can use the command "EXPLAIN (FORMAT JSON) [QUERY]".		

5) Display user id and date of following from followers table where user id should not be USR01 or USR02.

```
select user_id,date_of_following from friends where user_id not
in('USR01','USR02') order by user_id;
```

Data Output

	user_id character varying (50)	date_of_following date
1	USR010	2020-01-01
2	USR012	2019-12-25
3	USR015	2019-05-25
4	USR015	2019-05-22
5	USR017	2019-10-19
6	USR020	2017-02-20
7	USR04	2020-06-07
8	USR04	2017-07-17
9	USR04	2019-09-19
10	USR04	2018-12-02
11	USR05	2016-06-22
12	USR05	2019-01-30
13	USR05	2016-06-22
14	USR06	2018-09-19
15	USR06	2020-10-20
16	USR07	2020-04-06
17	USR07	2021-08-12
18	USR07	2017-04-06
19	USR07	2017-11-02
20	USR07	2019-10-10
21	USR08	2020-06-20
22	USR08	2019-09-09
23	USR08	2018-06-22
24	USR09	2018-06-16
25	USR09	2016-06-06

Query Editor

1

--5)Display user id and date of following from followers table where user id should not be USR01 or USR02.

2

select user_id,date_of_following from friends where user_id not in('USR01','USR02') order by user_id;

Explain

Query History

Notifications

Use Explain/Explain analyze button to generate the plan for a query. Alternatively, you can also execute "EXPLAIN (FORMAT JSON) [QUERY]".

Messages

Successfully run. Total query runtime: 51 msec.
25 rows affected.

6) No. of friend of a particular user. user_id= USR08

```
select COUNT(friends.friend_id) from friends WHERE friends.user_id =
'USR08';
```

Query Editor							
	<pre>1 --6)No. of friend of a particular user. user_id= USR08 2 select COUNT(friends.friend_id) from friends WHERE friends.user_id = 'USR08';</pre>						
Explain	Query History Notifications Data Output						
	<table> <tr> <th>count</th><th></th></tr> <tr> <td>bigint</td><td></td></tr> <tr><td>1</td><td>3</td></tr> </table>	count		bigint		1	3
count							
bigint							
1	3						

7) Most liked post of a particular user.

```
SELECT post_id, count(likier_id) as count FROM likes GROUP BY post_id ORDER BY count desc LIMIT 1;
```

Query Editor			
1	--7) Most liked post of a particular user.		
2	SELECT post_id, count(likier_id) as count FROM likes GROUP BY post_id ORDER BY count desc LIMIT 1;		
Explain Query History Notifications Data Output			
	post_id character varying (50)	count bigint	
1	USR20PST01	3	

5.4 Queries (self join & join & subqueries)

1) liker_id of post id = USR07PST01.

```
SELECT post.post_id, post.likes, likes.liker_id FROM post inner join likes on post.post_id = likes.post_id;
```

Query Editor

```
1 --1)liker_id of post id = USR07PST01.
2 SELECT post.post_id, post.likes, likes.liker_id FROM post inner join likes on post.post_id = likes.post_id;
3
```

Data Output Explain Query History Notifications

	post_id character varying (50)	likes integer	liker_id character varying (50)
1	USR01PST01		1 USR01
2	USR01PST02		2 USR011
3	USR01PST02		2 USR014
4	USR02PST01		1 USR01
5	USR04PST01		1 USR02
6	USR04PST02		2 USR07
7	USR04PST02		2 USR08
8	USR04PST03		1 USR05
9	USR07PST01		2 USR011
10	USR07PST01		2 USR012
11	USR08PST01		1 USR013
12	USR09PST01		2 USR014
13	USR09PST01		2 USR05
14	USR11PST01		1 USR016
15	USR14PST01		2 USR017
16	USR14PST01		2 USR018
17	USR14PST02		1 USR015
18	USR17PST01		1 USR016
19	USR17PST02		2 USR01
20	USR17PST02		2 USR017
21	USR18PST01		1 USR017
22	USR20PST01		3 USR019
23	USR20PST01		3 USR09
24	USR20PST01		3 USR015

Messages

Successfully run. Total query runtime: 55 msec.
24 rows affected.

2) Display name of friends of user='USR07' who had posted story?

```
select friends.friend_id from friends inner join story on friends.user_id = story.user_id WHERE story.user_id in (select friends.friend_id from friends where friends.user_id= 'USR07');
```

SOCIALMEDIA/postgres@PostgreSQL 13

Query Editor

```

1 select friends.friend_id from friends inner join story on friends.user_id = story.user_id
2 WHERE story.user_id in (select friends.friend_id from friends where friends.user_id= 'USR07');
3

```

Data Output Explain Query History Notifications

friend_id
USR018

3) Display story information of story which has been posted at the earliest.

select * from story where posting_time=(select posting_time from story order by posting_time limit 1);

Query Editor

```

1 --3)Display story information of story which has been posted at the earliest.
2 select * from story where posting_time=(select posting_time from story order by posting_time limit 1);
3

```

Data Output Explain Query History Notifications

story_id	user_id	show_status	posting_time	seen_by	highlights	time_elated	no_of_share
STORYUSR013	USR013	true	2021-06-22 00:00:40	3	false	[null]	1

4) Display post_id ,likes from post table where no. of likes of one is greater than another.

select a.post_id,a.likes,b.post_id,b.likes from post a, post b where a.likes>b.likes;



post_id	likes	post_id	likes
USR01PST02	2	USR01PST01	1
USR01PST02	2	USR02PST01	1
USR01PST02	2	USR04PST01	1
USR01PST02	2	USR04PST03	1
USR01PST02	2	USR08PST01	1
USR01PST02	2	USR11PST01	1
USR01PST02	2	USR14PST02	1
USR01PST02	2	USR17PST01	1
USR01PST02	2	USR18PST01	1
USR04PST02	2	USR01PST01	1
USR04PST02	2	USR02PST01	1
USR04PST02	2	USR04PST01	1
USR04PST02	2	USR04PST03	1
USR04PST02	2	USR08PST01	1
USR04PST02	2	USR11PST01	1
USR04PST02	2	USR14PST02	1
USR04PST02	2	USR17PST01	1
USR04PST02	2	USR18PST01	1
USR07PST01	2	USR01PST01	1
USR07PST01	2	USR02PST01	1
USR07PST01	2	USR04PST01	1
USR07PST01	2	USR04PST03	1
USR07PST01	2	USR08PST01	1
USR07PST01	2	USR11PST01	1
USR07PST01	2	USR14PST02	1
USR07PST01	2	USR17PST01	1
USR07PST01	2	USR18PST01	1
USR09PST01	2	USR01PST01	1
USR09PST01	2	USR02PST01	1
USR09PST01	2	USR04PST01	1
USR09PST01	2	USR04PST03	1
USR09PST01	2	USR08PST01	1
USR09PST01	2	USR11PST01	1
USR09PST01	2	USR14PST02	1
USR09PST01	2	USR17PST01	1
USR09PST01	2	USR18PST01	1
USR14PST01	2	USR01PST01	1
USR14PST01	2	USR02PST01	1
USR14PST01	2	USR04PST01	1
USR14PST01	2	USR04PST03	1
USR14PST01	2	USR08PST01	1
USR14PST01	2	USR11PST01	1
USR14PST01	2	USR14PST02	1
USR14PST01	2	USR17PST01	1
USR14PST01	2	USR18PST01	1
USR17PST02	2	USR01PST01	1
USR17PST02	2	USR02PST01	1
USR17PST02	2	USR04PST01	1
USR17PST02	2	USR04PST03	1
USR17PST02	2	USR08PST01	1
USR17PST02	2	USR11PST01	1
USR17PST02	2	USR14PST02	1
USR17PST02	2	USR17PST01	1
USR17PST02	2	USR18PST01	1
USR20PST01	3	USR01PST01	1
USR20PST01	3	USR01PST02	2
USR20PST01	3	USR02PST01	1
USR20PST01	3	USR04PST01	1
USR20PST01	3	USR04PST02	2
USR20PST01	3	USR04PST03	1
USR20PST01	3	USR07PST01	2
USR20PST01	3	USR08PST01	1
USR20PST01	3	USR09PST01	2
USR20PST01	3	USR11PST01	1
USR20PST01	3	USR14PST01	2
USR20PST01	3	USR14PST02	1
USR20PST01	3	USR17PST01	1
USR20PST01	3	USR17PST02	2
USR20PST01	3	USR18PST01	1

(69 rows)

5)Display first_name of user who has commented on any Post .

```
select users.firstname from (post inner join comments on
comments.post_id=post.post_id) inner join users on
users.user_id=comments.commentby_id;
```

```
1 --5)Display first_name of user who has commented on any Post .
2 select users.firstname from (post inner join comments on comments.post_id=post.post_id)
3 inner join users on users.user_id=comments.commentby_id;
4
```

Explain	Query History	Notifications	Data Output
	firstname character varying (50)		
1	Ashtha		
2	Riya		
3	Juli		
4	Rahul		
5	Ashtha		
6	Parth		
7	Pratik		
8	Jitu		
9	Deep		

6)Most liked post .

```
SELECT post_id, COUNT(DISTINCT liker_id) FROM likes GROUP BY post_id
ORDER BY COUNT DESC LIMIT 1 ;
```

Query Editor

```
1 --6)Most liked post .
2 SELECT post_id, COUNT(DISTINCT liker_id) FROM likes GROUP BY post_id ORDER BY COUNT DESC LIMIT 1 ;
3
4
```

Explain	Query History	Notifications	Data Output
	<div> <div>post_id</div> <div>character varying (50)</div> </div>	<div> <div>count</div> <div>bigint</div> </div>	
1	USR20PST01	3	



7) Display mutual_friends of USR01 and USR02.

```
SELECT f1.friend_id as Mutual_friend,f1.f_name FROM friends f1 INNER JOIN friends f2 ON f1.friend_id = f2.friend_id WHERE f1.user_id = 'USR01' AND f2.user_id = 'USR02';
```

Query Editor

```
1 --7) Display mutual_friends of USR01 and USR02.
2 SELECT f1.friend_id as Mutual_friend,f1.f_name FROM friends f1 INNER JOIN friends f2
3 ON f1.friend_id = f2.friend_id WHERE f1.user_id = 'USR01' AND f2.user_id = 'USR02';
4
```

Explain Query History Notifications Data Output

	 mutual_friend character varying (50)	 f_name character varying (50)	
1	USR04	Riya Kotak	


8) Display friends of each other from friends table .

```
SELECT f1.user_id , f2.user_id FROM friends f1 INNER JOIN friends f2 ON f1.user_id =f2.friend_id WHERE f1.friend_id =f2.user_id ;
```

Query Editor

```
1 --8) Display friends of each other from friends table .
2 SELECT f1.user_id , f2.user_id FROM friends f1 INNER JOIN friends f2
3 ON f1.user_id =f2.friend_id WHERE f1.friend_id =f2.user_id ;
4
```

Explain Query History Notifications Data Output

	 user_id character varying (50)	 user_id character varying (50)	
1	USR01	USR02	
2	USR02	USR01	
3	USR02	USR04	
4	USR04	USR02	
5	USR04	USR05	
6	USR04	USR08	
7	USR05	USR04	
8	USR08	USR04	

9)searching for users that aren't already friends.

```
SELECT f2.user_id ,COUNT(*) as friends_in_common FROM friends f1 LEFT  
JOIN friends f2   ON f1.friend_id = f2.friend_id  
WHERE f1.user_id = 'USR07' GROUP BY f2.user_id ORDER BY  
friends_in_common DESC ;
```

Query Editor			
1	--9)searching for users that aren't already friends.		
2	SELECT f2.user_id ,COUNT(*) as friends_in_common FROM friends f1 LEFT JOIN friends f2 ON f1.friend_id = f2.friend_id		
3	WHERE f1.user_id = 'USR07' GROUP BY f2.user_id ORDER BY friends_in_common DESC ;		
4			
Explain Query History Notifications Data Output			
	user_id character varying (50)	friends_in_common bigint	
1	USR07	5	
2	USR01	3	
3	USR012	1	
4	USR020	1	
5	USR04	1	

5.5 PL/SQL

1) Create a view that displays story_id and the no. of times it is shared.

```
create view newstory as
select story_id,no_of_share from story;

select * from newstory;
```

Query Editor

1

2

3

4

5

--2)Create a view that displays story_id and the no. of times it is shared.
create view newstory as
select story_id,no_of_share from story;

select * from newstory;

Explain

Query History

Notifications

Data Output

	story_id character varying (50)	no_of_share integer	
1	STORYUSR01	1	
2	STORYUSR03	0	
3	STORYUSR04	3	
4	STORYUSR02	5	
5	STORYUSR011	7	
6	STORYUSR019	2	
7	STORYUSR013	1	

5.6 Functions

1) Create a function gives the total number of private accounts.

```
create function get_privacy_status()
returns int
language plpgsql
as
$$
Declare
    user_count integer;
Begin
    select count(*)
    into user_count
    from users
    where privacy_status='true';
    return user_count;
End;
$$;
```

```
select get_privacy_status();
```

Query Editor

```
1  --1)Create a function gives the total number of private accounts.
2  create function get_privacy_status()
3  returns int
4  language plpgsql
5  as
6  $$
7  Declare
8      user_count integer;
9  Begin
10     select count(*)
11     into user_count
12     from users
13     where privacy_status='true';
14     return user_count;
15 End;
16 $$;
17
18 select get_privacy_status();
```

Explain Query History Notifications Data Output

	get_privacy_status	
	integer	🔒
1		8

5.7 Triggers

1) Create a trigger which displays an error message if someone tries to enter invalid E-mail.

```
create function check_email() returns trigger as $$
BEGIN
if NEW.email LIKE '%____@____%.____%' then
return NEW;
end if;
raise exception 'Your E-mail format must be like
'***@**.*';
END;
$$
LANGUAGE plpgsql;
```

```
create trigger email_check
BEFORE INSERT OR UPDATE
ON users
FOR EACH ROW
EXECUTE PROCEDURE check_email();
```

```
INSERT INTO users(user_id , password ,    firstname ,
                lastname ,    email ,    dp_url ,no_of_post ,
                privacy_status , friends ,created_at
)VALUES('USR021','Pass@USR21','Shyam','Sharma','shyamsharma
gmail.com','IC/DP/USRDP021.png',2,FALSE,4,'2010-06-02
01:10:25-03');
```

Messages

ERROR: Your E-mail format
must be like **@**.*
CONTEXT: PL/pgSQL function
check_email() line 6 at
RAISE
SQL state: P0001

Query Editor

```
1  --1)Create a trigger which displays an error message if someone tries to enter invalid E-mail.
2  create function check_email() returns trigger as $$
3  BEGIN
4  if NEW.email LIKE '%__@__%.__%' then
5  return NEW;
6  end if;
7  raise exception 'Your E-mail format must be like **@**.* ';
8  END;
9  $$
10 LANGUAGE plpgsql;
11
12 create trigger email_check
13 BEFORE INSERT OR UPDATE
14 ON users
15 FOR EACH ROW
16 EXECUTE PROCEDURE check_email();
17
18 INSERT INTO users(user_id , password , firstname , lastname , email , dp_url ,
19 no_of_post , privacy_status , friends ,created_at )VALUES('USR021','Pass@USR21','Shyam','Sharma
20 , 'shyamsharmamagmail.com', 'IC/DP/USRDP021.png',2,FALSE,4, '2010-06-02 01:10:25-03');
21
```

2)Create a trigger which inserts the new tuples on insertion and old tuples on deletion into a new table called friend_audit.

```
create or replace function do_friend_audit()
returns trigger as $$
begin
if(TG_OP='DELETE') then insert into friend_audit
select 'D',now(),OLD.user_id,OLD.f_name;
return OLD;

elseif(TG_OP='UPDATE') then insert into friend_audit
select 'U',now(),NEW.user_id,NEW.f_name;
return NEW;

elseif(TG_OP='INSERT') then insert into friend_audit
select 'I',now(),NEW.user_id,NEW.f_name;
return NEW;

end if;
end;

$$
language plpgsql;

create trigger friend_audit
after insert or update or delete on friends
for each row
execute procedure do_friend_audit();

create table friend_audit(OP varchar(10) not null,
stamp timestamp not null,
user_id varchar(10) not null,
friend_name varchar(10) not null);

INSERT INTO users(user_id , password ,    firstname ,
                lastname ,    email ,    dp_url , no_of_post ,
                privacy_status , friends ,    created_at )VALUES
('USR022','Pass@USR22','Mukund','Raj','mukundraj@gmail.com'
```

```
, 'IC/DP/USRDP022.png',2,FALSE,4,'2021-06-02 01:10:25-03');
```

```
INSERT INTO friends( user_id , friend_id , f_name ,
close_friend , date_of_following , mutual_friends ,
most_shown , least_interacted)VALUES
('USR022','USR02','Yash Varma',TRUE,'2021-08-
23',1,FALSE,TRUE);
```

```
UPDATE friends set most_shown = 'TRUE' where user_id =
'USR022';
```

```
SELECT * from friend_audit;
```

```
Query Editor
1 create or replace function do_friend_audit()
2 returns trigger as $$
3 begin
4     if(TG_OP='DELETE') then insert into friend_audit
5         select 'D',now(),OLD.user_id,OLD.f_name;
6         return OLD;
7     elseif(TG_OP='UPDATE') then insert into friend_audit
8         select 'U',now(),NEW.user_id,NEW.f_name;
9         return NEW;
10    elseif(TG_OP='INSERT') then insert into friend_audit
11        select 'I',now(),NEW.user_id,NEW.f_name;
12        return NEW;
13    end if;
14 end;
15 $$
16 language plpgsql;
17
18 create trigger friend_audit
19 after insert or update or delete on friends for each row
20 execute procedure do_friend_audit();
21
22 create table friend_audit(OP varchar(10) not null, stamp timestamp not null, user_id varchar(10) not null, friend_name varchar(10) not null);
23
24
25 INSERT INTO users(user_id , password , firstname , lastname , email , dp_url , no_of_post , privacy_status , friends , created_at
26 ('USR022','Pass@USR022','Mukund','Raj','mukundraj@gmail.com','IC/DP/USRDP022.png',2,FALSE,4,'2021-06-02 01:10:25-03');
```

```
Query Editor
27
28
29 user_id varchar(10) not null,
30 friend_name varchar(10) not null);
31
32
33 INSERT INTO users(user_id , password , firstname , lastname , email , dp_url , no_of_post , privacy_status , friends , created_at
34 ('USR022','Pass@USR022','Mukund','Raj','mukundraj@gmail.com','IC/DP/USRDP022.png',2,FALSE,4,'2021-06-02 01:10:25-03');
35
36 INSERT INTO friends( user_id , friend_id , f_name , close_friend , date_of_following , mutual_friends , most_shown , least_interacted)VALU
37 ('USR022','USR02','Yash Varma',TRUE,'2021-08-23',1,FALSE,TRUE);
38
39 UPDATE friends set most_shown = 'TRUE' where user_id = 'USR022';
40
41 SELECT * from friend_audit;
42
```

Data Output				
	op character varying (10)	stamp timestamp without time zone	user_id character varying (10)	friend_name character varying (10)
1	I	2021-10-21 09:55:26.835675	USR022	Yash Varma
2	U	2021-10-21 09:55:26.835675	USR022	Yash Varma

5.8 Cursor

1) Create a cursor which displays the the respective story's information iff it is included in highlights. Otherwise it should not display anything.

create or replace function get_story_info(str_id varchar(50)) returns text as \$\$

declare name text default '';

storyrec record;

cur_str cursor(str_id varchar(50))

for select user_id,story_id

from story

where story.highlights='TRUE' and str_id=story_id;

begin

open cur_str(str_id);

fetch cur_str into storyrec;

exit when not found;

name := storyrec.user_id || ' , ' || storyrec.story_id;

close cur_str;

return name;

end; \$\$

language plpgsql;

SELECT get_story_info('STORYUSR01');

SELECT get_story_info('STORYUSR02');

```
postgres=# SELECT get_story_info('STORYUSR01');
get_story_info
-----
USR020 , STORYUSR01
(1 row)

postgres=# SELECT get_story_info('STORYUSR02');
get_story_info
-----
(1 row)
```


6. FUTURE ENHANCEMENTS OF THE SYSTEM

- We will design Front-end Design in HTML , CSS , JavaScript and Develop Bank-end in Python.
- For security purpose New Registration is done using OTP.
- We will make database more consistent and We are making this database efficient and easy to implement with huge data capacity.
- Methods and user data input will be lot easy after the implement of GUI.
- We will also add some extra features so that the users can get answer for their complaints as fast as possible.

7.

BIBLIOGRAPHY

- For the successful implementation of this project we referred to many websites and books.
- We created the ER Diagram and Schema Diagram on “erdplus.com”.
- Mostly we referred the online material for syntax of procedures, triggers, Exception and cursors.

Reference book:

Data Base System Concepts

-Henry F. Korth & A. Silberschatz 2nd Ed. McGraw-Hill 1991

Reference Websites:

- <https://www.stackoverflow.com/>
- <https://erdplus.com>