Group Project: DBMS

Shivam Yadav - 2020471
Shivanshu Nayak - 2020472
Apoorva Agarwal - 2020426

Project Scope
The scope of our project is to develop a database messaging application like WhatsApp that allows its users to log in using mobile, send or receive text messages, make voice calls, and create groups that have the same features for more effective communication.

Stakeholders
Anyone who has access to the internet and a smartphone with them can be a user of our service. Basically, users who get benefitted from instant messaging, voice calls to friends and family with access to just the internet. Businesses, where instant messaging to employees and taking orders from customers is required, are all included in stakeholders.

Views
A view is generally a virtual table based on the result-set of an SQL statement. It contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

1. **An SQL statement to create a view as MY_GROUPS using the Group and User table.**

```
Create view MY_GROUPS AS
select group_name, G.group_id
from g_roup G
left join user u
on u.user_id = G.user_id
```

2. **An SQL statement to create a view as USER_INFO using the User table where user id is more than 25**

```
Create view USER_INFO AS
select user_id, name
from user
where user_id > 25
order by user_id;
```

Grants
Grant is used to provide permissions like Select, All, Execute to users on the database objects like Tables, Views, Databases, and other objects in a SQL Server.

```
use whatsapp;

grant select
on chat_history
to 'user1';
```

SQL Queries

1. **To get sender and receiver details for both calls and chats of users whose call duration is more than an hour.**

```
SELECT c.receiver_id AS userid, call_id, c.sender_id AS caller,
duration, chat_id, ch.sender_id as sender
FROM call_history c
JOIN chat_history ch
ON ch.receiver_id= c.receiver_id
WHERE c.duration > '01:00:00'
ORDER BY userid;
```

2. **To get call_id, caller_id and name of users who received any calls and their number starts with ‘31’**

```
SELECT distinct ch.call_id , ch.sender_id as caller_id, u.name
FROM call_history ch
JOIN user u
ON u.user_id = ch.receiver_id
WHERE u.phone_number like '31%' ;
```

3. **To get the name, phone number, userid, and chat_id of users whose userid is same as chat_id and their number has ‘5641’ in it.**

```
SELECT name, phone_number, user_id, chat_id
FROM user
JOIN chat
ON user.user_id = chat.chat_id
WHERE phone_number regexp 5641;
```

4. **To get user_id, name, status, group_id, group name and status of users who are in CSD SecB group with description as Busy. And arrange in decreasing order.**

```
SELECT Distinct U.User_id, Name, S.description as
Personal_Status, U.Group_id, Group_name, SG.description as
Group_status
FROM user U
JOIN s_tatus S
ON S.user_id = U.user_id
JOIN g_roup G
ON G.group_name= 'CSD_SecB'
JOIN status_group SG
ON SG.group_id = G.group_id and SG.description= 'Busy'
ORDER BY user_id desc;
```

5. **To get userid, callid, date and Active/inactive status based on last message sent by users and order by userid.**

```
SELECT distinct u.user_id, call_id, date, 'Inactive' as Status
FROM call_history ch
JOIN user u
ON u.user_id= ch.sender_id
WHERE date < '2002-01-01'
UNION ALL
SELECT distinct u.user_id, call_id, date, 'Active' as Status
FROM call_history ch
JOIN user u
ON u.user_id= ch.sender_id
WHERE date >= '2002-01-01'
ORDER BY user_id;
```

6. **To get user_id, reciever_id, file name of group 1 users who sent documents.**

```
SELECT M.user_id, receiver_id,type, file_name
FROM media M
JOIN g_roup G
ON group_id = 1
WHERE action = 'send' AND type = 'documents'
ORDER BY M.user_id;
```

7. **To get userid, name, group_id, file name of users who sent documents with ‘plot twist’ written into then to 1,2,5,6,8 groups and order by userid.**

```
SELECT M.user_id as Sender, u.name as Sender_name,
G.group_id, file_name
FROM media M
JOIN g_roup G
ON group_id in (1,2,5,6,8)
JOIN chat_history ch
ON ch.decription = 'plot twist'
JOIN user u
ON M.user_id = u.user_id
WHERE action = 'send' and type = 'documents'
ORDER BY M.user_id;
```

8. **To get goup name and group id of users whose userid is between 100 and 500 and group have more than 100 members.**

```
SELECT group_name, G.group_id
FROM g_roup G
LEFT JOIN user u
ON u.user_id = G.group_id
WHERE G.user_id between 100 and 500 AND G.group_name =
'CSE_SecA'
HAVING COUNT(G.no_of_membes>100);
```

9. **To get all details of 3 groups having group name as Maths_SecA or CSSS_SecA and have more than 10 members.**

```
SELECT *
FROM G_roup
WHERE group_name in ('Maths_SecA', 'CSSS_SecA')
GROUP BY group_id
HAVING COUNT(no_of_membes >= 10)
LIMIT 3;
```

10. **To get user_id, name, country, and home/abroad status based on country of users from India and US.**

```
SELECT user_id, name, country, 'home' as status
FROM User
WHERE country = 'India'
UNION ALL
SELECT user_id, name, country, 'abroad' as status
FROM User
WHERE country = 'United States'
```

11.To insert details of some users with given details in users table.

```
INSERT INTO
User(user_id,name,phone_number,group_id,country)
VALUES (1007,'Chaman',7819329958,'3,1,2','India'),
      (1008,'Sam',123876958,'1,2,6','United States'),
      (1009,'Noman',7812329958,'3,4,2','Pakistan'),
       (1010,'Raman',3834529958,'1,2,5','India');
```

12. To update the users name as Spam in call_history whose phone number has 81 in it.

```
UPDATE call_history ch
JOIN c_all c
ON ch.sender_id = c.sender_id
JOIN user u
ON u.user_id = ch.receiver_id
AND u.phone_number regexp '81'
SET u.name = 'Spam';
```

13. To update the phone number as 9197247869 of users who have 7197 in their number.

```
UPDATE user
JOIN chat
ON user.user_id = chat.chat_id
AND phone_number regexp 7197
SET phone_number = 9197247869;
```

14. To get reciever_id, call_id, caller_id, chat_id, sender_id and avg(dutration) of users having reciever_id more than 100 and duration more than 3 hours.

```
SELECT c.receiver_id as userid, call_id, c.sender_id as caller,
duration, chat_id, ch.sender_id as sender, avg(duration)
FROM call_history c
JOIN chat_history ch
ON ch.receiver_id > 100
WHERE c.duration > '03:00:00'
ORDER BY duration;
```

Query Optimization

It can be defined as the iterative process of enhancing the performance of a query in terms of execution time, the number of disk accesses, and many more cost measuring criteria as data is an integral part of any application and access to the data should be in the fastest way possible to enhance the user experience while using the application.

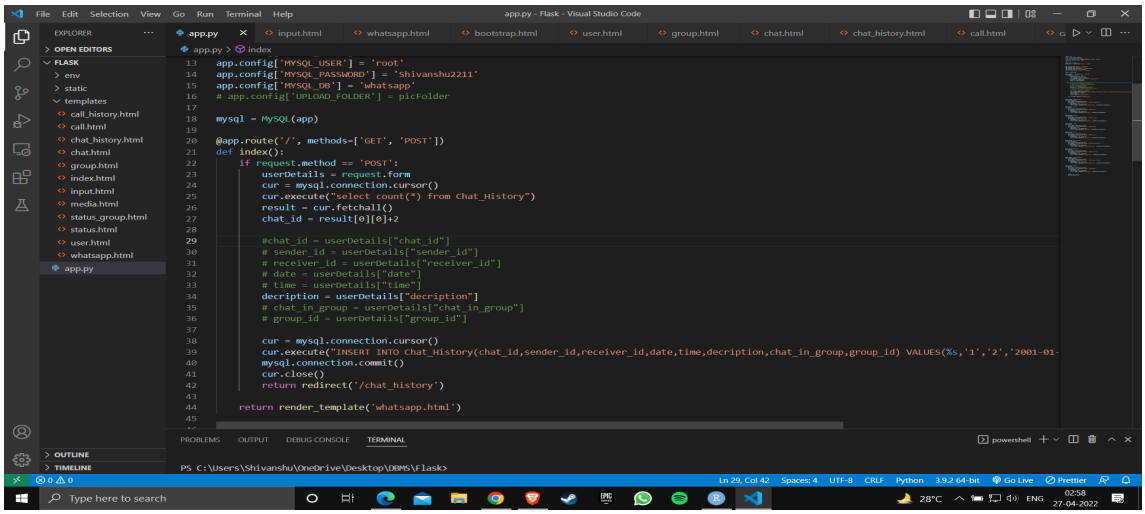
The major purposes of SQL Query optimization are:
1. Reduce Response Time
2. Reduced CPU execution time
3. Improved Throughput

We have optimized our SQL using the following measures:

- Avoid using SELECT *. Instead, we used some column names with SELECT.
- Avoid using DISTINCT with SELECT.
- Used JOINS instead of WHERE clause wherever possible. Used WHERE along with JOIN to further filter the search results.
- Used LIMIT command to limit the results.
- Avoid using HAVING clause. Used having only in case aggregate functions.
- Using EXISTS instead of IN clause.
- Using Indexes properly.
- Used UNION ALL in place of UNION.
- Used ‘Int’ datatype in fields where we knew that data would be only in integers.
- Avoided running queries in loops.

Embedded SQL Queries

Embedded SQL statements are SQL statements written inline with the program source code, of the host language.



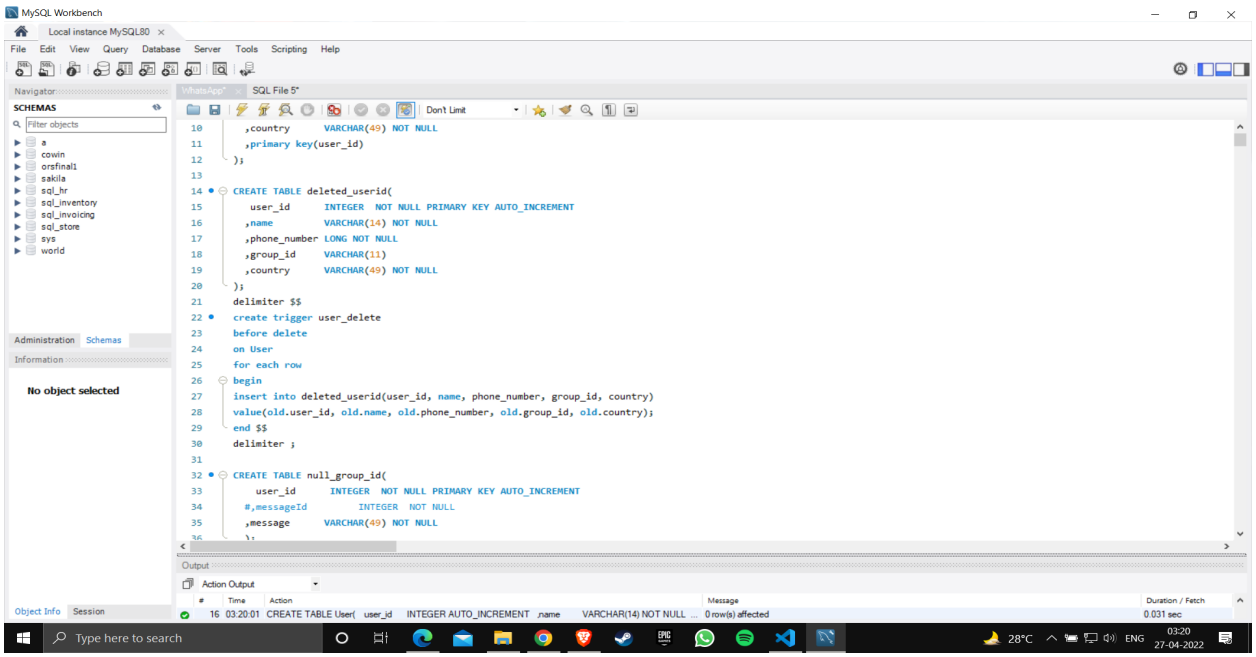

```python
@app.route('/chat_history')
def chat_history():
    cur = mysql.connection.cursor()
    resultValue = cur.execute("SELECT * FROM Chat_History")
    if resultValue > 0:
        userDetails = cur.fetchall()
        return render_template('chat_history.html', userDetails=userDetails)
```

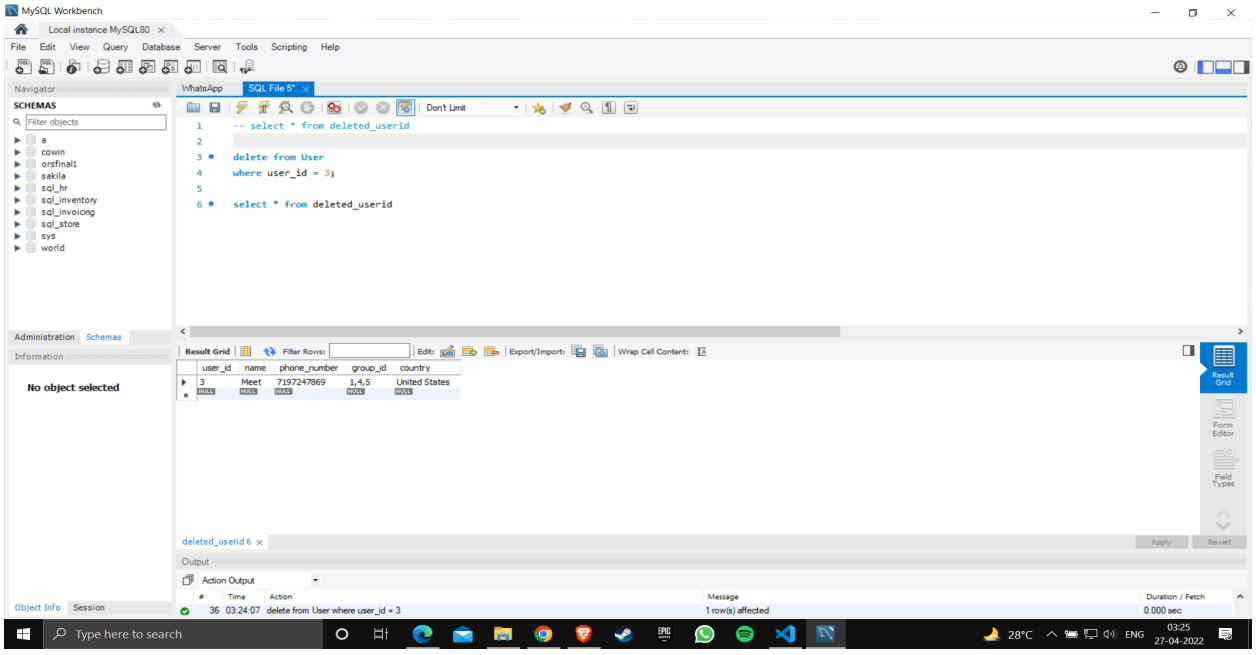
```python
@app.route('/user')
def user():
    cur = mysql.connection.cursor()
    resultValue = cur.execute("SELECT * FROM User")
    if resultValue > 0:
        userDetails = cur.fetchall()
        return render_template('user.html', userDetails=userDetails)
```

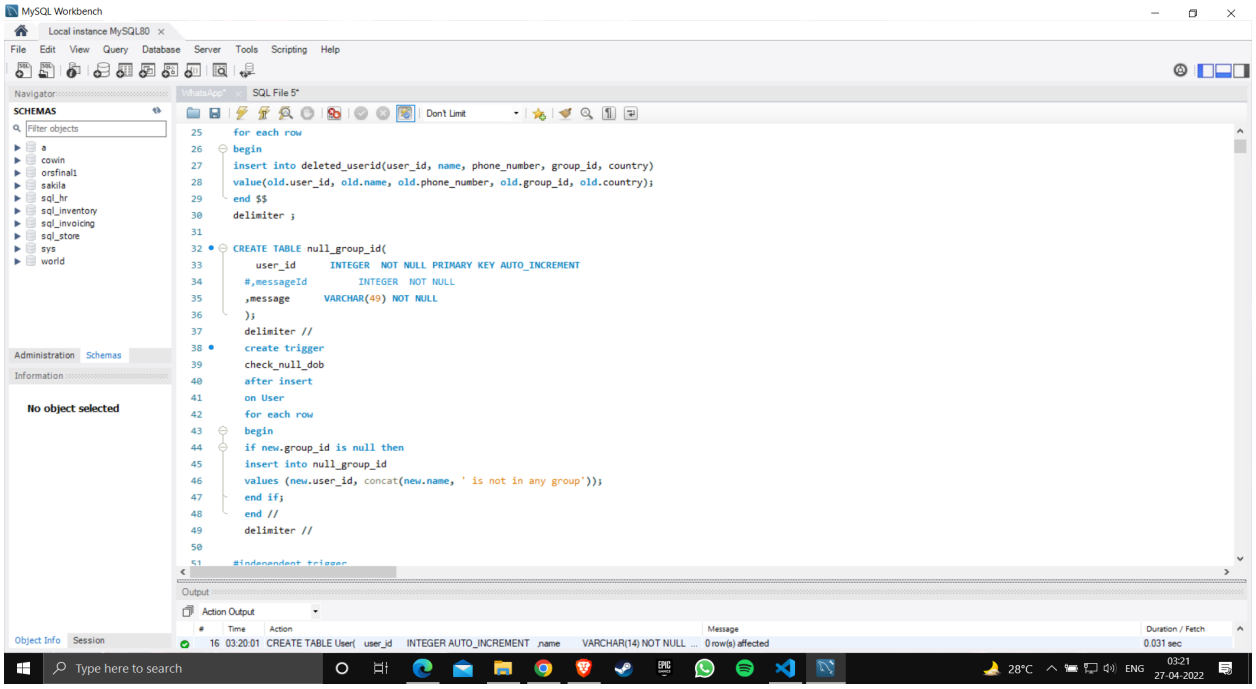
```python
@app.route('/group')
def group():
    cur = mysql.connection.cursor()
    resultValue = cur.execute("SELECT * FROM G_roup")
    if resultValue > 0:
        userDetails = cur.fetchall()
        return render_template('group.html', userDetails=userDetails)
```

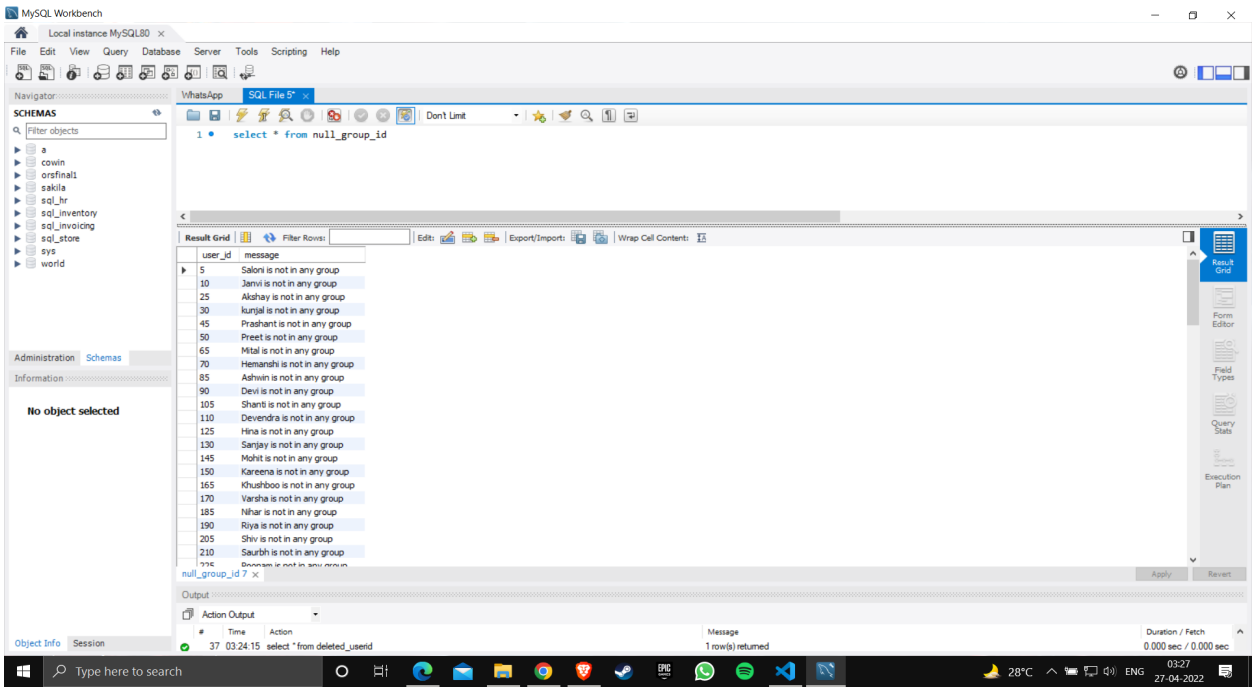
Triggers

A trigger in MySQL is a set of SQL statements that reside in a system catalog. It is a special type of stored procedure that is invoked automatically in response to an event.

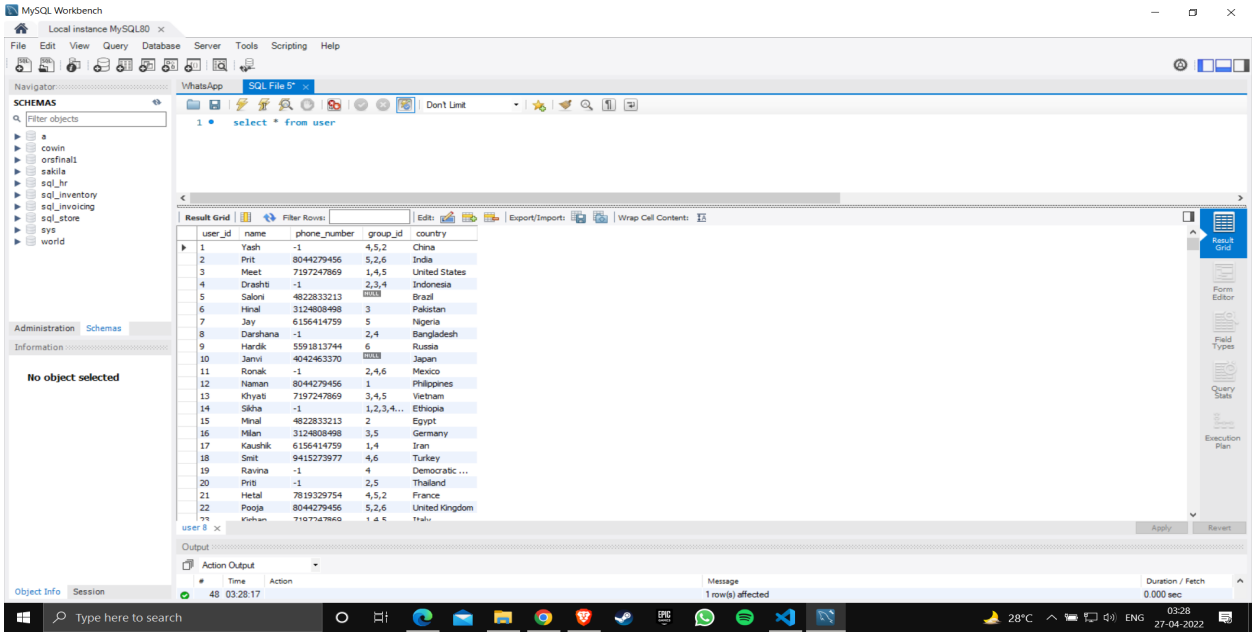


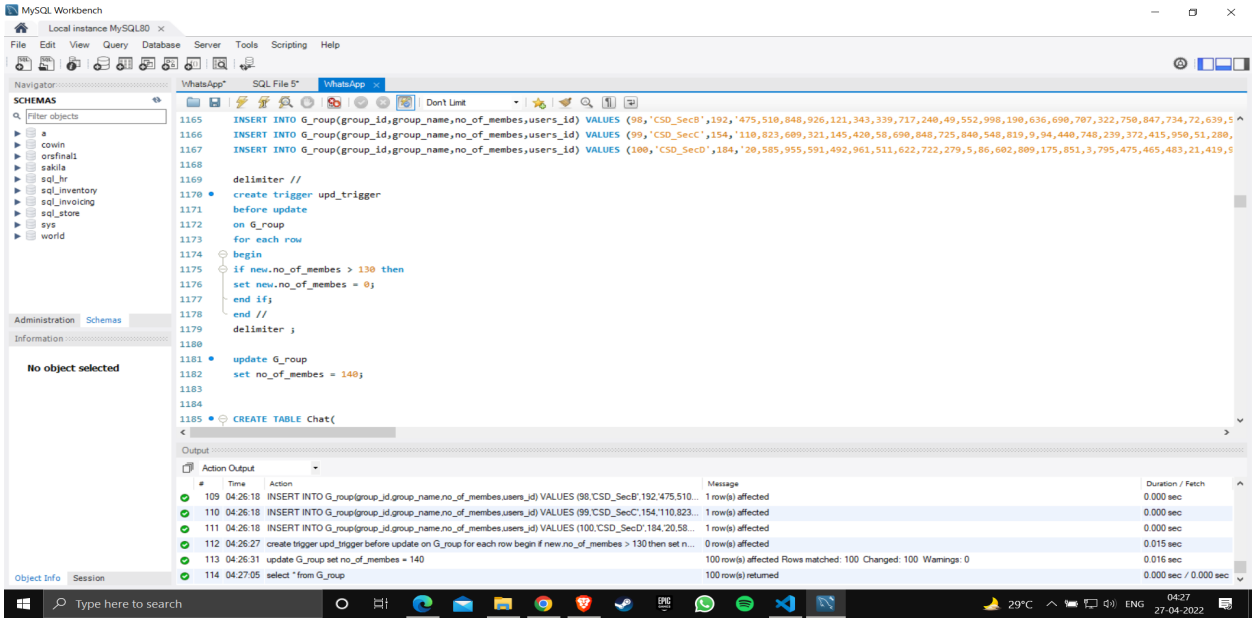


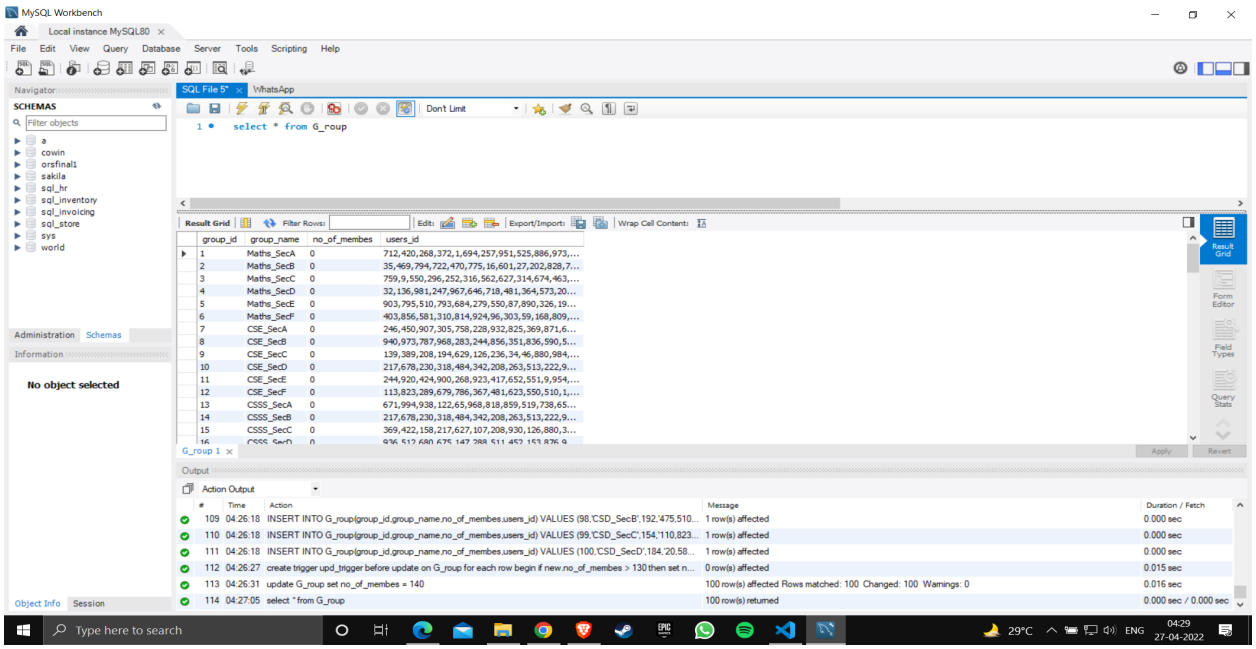





```
#independent trigger
delimiter //
create trigger check_phone_number
before insert on User
for each row
if new.phone_number < 999999999 OR new.phone_number > 9999999999 then set new.phone_number = -1;
end if; //
```


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
a
cowin
orsfinal1
sakila
sql_hr
sql_inventory
sql_invoicing
sql_store
sys
world
Administration Schemas
Information
No object selected
Object Info Session
WhatsApp
SQL File 5*
Don't Limit
1 • select * from user
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
user_id name phone_number group_id country
1 Yash -1 4,5,2 China
2 Prit 8044279456 5,2,6 India
3 Meet 7197247869 1,4,5 United States
4 Drashti -1 2,3,4 Indonesia
5 Saloni 4822833213 NULL Brazil
6 Hinal 3124808498 3 Pakistan
7 Jay 6156414759 5 Nigeria
8 Darshana -1 2,4 Bangladesh
9 Hardik 5591813744 6 Russia
10 Janvi 4042463370 NULL Japan
11 Ronak -1 2,4,6 Mexico
12 Naman 8044279456 1 Philippines
13 Khyati 7197247869 3,4,5 Vietnam
14 Sikha -1 1,2,3,4... Ethiopia
15 Minal 4822833213 2 Egypt
16 Milan 3124808498 3,5 Germany
17 Kaushik 6156414759 1,4 Iran
18 Smit 9415273977 4,6 Turkey
19 Ravina -1 4 Democratic ...
20 Priti -1 2,5 Thailand
21 Hetal 7819329754 4,5,2 France
22 Pooja 8044279456 5,2,6 United Kingdom
user 8
Apply Revert
Output
Action Output
Time Action Message Duration / Fetch
48 03:28:17 1 row(s) affected 0.000 sec
Result Grid
Form Editor
Field Types
Query Stats
Execution Plan
Type here to search
28°C ENG 03:28 27-04-2022


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
SCHEMAS
a
cowin
orsfinal1
sakila
sql_hr
sql_inventory
sql_invoicing
sql_store
sys
world
No object selected
INSERT INTO G_roup(group_id,group_name,no_of_membes,users_id) VALUES (98,'CSD_SecB',192,'475,510,848,926,121,343,339,717,240,49,552,998,190,636,690,707,322,750,847,734,72,639,5
INSERT INTO G_roup(group_id,group_name,no_of_membes,users_id) VALUES (99,'CSD_SecC',154,'110,823,609,321,145,420,58,690,848,725,840,548,819,9,94,440,748,239,372,415,950,51,280,
INSERT INTO G_roup(group_id,group_name,no_of_membes,users_id) VALUES (100,'CSD_SecD',184,'20,585,955,591,492,961,511,622,722,279,5,86,602,809,175,851,3,795,475,465,483,21,419,5
delimiter //
create trigger upd_trigger
before update
on G_roup
for each row
begin
if new.no_of_membes > 130 then
set new.no_of_membes = 0;
end if;
end //
delimiter ;
update G_roup
set no_of_membes = 140;
CREATE TABLE Chat(
Output
Action Output
109 04:26:18 INSERT INTO G_roup(group_id,group_name,no_of_membes,users_id) VALUES (98,'CSD_SecB',192,'475,510... 1 row(s) affected 0.000 sec
110 04:26:18 INSERT INTO G_roup(group_id,group_name,no_of_membes,users_id) VALUES (99,'CSD_SecC',154,'110,823... 1 row(s) affected 0.000 sec
111 04:26:18 INSERT INTO G_roup(group_id,group_name,no_of_membes,users_id) VALUES (100,'CSD_SecD',184,'20,58... 1 row(s) affected 0.000 sec
112 04:26:27 create trigger upd_trigger before update on G_roup for each row begin if new.no_of_membes > 130 then set n... 0 row(s) affected 0.015 sec
113 04:26:31 update G_roup set no_of_membes = 140 100 row(s) affected Rows matched: 100 Changed: 100 Warnings: 0 0.016 sec
114 04:27:05 select * from G_roup 100 row(s) returned 0.000 sec / 0.000 sec
Type here to search
29°C ENG 04:27 27-04-2022


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
SCHEMAS
a
cowin
orsfinal1
sakila
sql_hr
sql_inventory
sql_invoicing
sql_store
sys
world
No object selected
select * from G_roup
Result Grid
group_id group_name no_of_membes users_id
1 Maths_SecA 0 712,420,268,372,1,694,257,951,525,886,973,...
2 Maths_SecB 0 35,469,794,722,470,775,16,601,27,202,828,7...
3 Maths_SecC 0 759,9,550,296,252,316,562,627,314,674,463,...
4 Maths_SecD 0 32,136,981,247,967,646,718,481,364,573,20...
5 Maths_SecE 0 903,795,510,793,684,279,550,87,890,326,19...
6 Maths_SecF 0 403,856,581,310,814,924,96,303,59,168,809,...
7 CSE_SecA 0 246,450,907,305,758,228,932,825,369,871,6...
8 CSE_SecB 0 940,973,787,968,283,244,856,351,836,590,5...
9 CSE_SecC 0 139,389,208,194,629,126,236,34,46,880,984,...
10 CSE_SecD 0 217,678,230,318,484,342,208,263,513,222,9...
11 CSE_SecE 0 244,920,424,900,268,923,417,652,551,9,954,...
12 CSE_SecF 0 113,823,289,679,786,367,481,623,550,510,1,...
13 CSSS_SecA 0 671,994,938,122,65,968,818,859,519,738,65...
14 CSSS_SecB 0 217,678,230,318,484,342,208,263,513,222,9...
15 CSSS_SecC 0 369,422,158,217,627,107,208,930,126,880,3...
G_roup 1
Output
Action Output
109 04:26:18 INSERT INTO G_roup(group_id,group_name,no_of_membes,users_id) VALUES (98,'CSD_SecB',192,'475,510... 1 row(s) affected 0.000 sec
110 04:26:18 INSERT INTO G_roup(group_id,group_name,no_of_membes,users_id) VALUES (99,'CSD_SecC',154,'110,823... 1 row(s) affected 0.000 sec
111 04:26:18 INSERT INTO G_roup(group_id,group_name,no_of_membes,users_id) VALUES (100,'CSD_SecD',184,'20,58... 1 row(s) affected 0.000 sec
112 04:26:27 create trigger upd_trigger before update on G_roup for each row begin if new.no_of_membes > 130 then set n... 0 row(s) affected 0.015 sec
113 04:26:31 update G_roup set no_of_membes = 140 100 row(s) affected Rows matched: 100 Changed: 100 Warnings: 0 0.016 sec
114 04:27:05 select * from G_roup 100 row(s) returned 0.000 sec / 0.000 sec
Type here to search
29°C ENG 04:29 27-04-2022

Indexing

Indexing is a powerful structure in MySQL which can be leveraged to get the fastest response times from common queries. MySQL queries achieve efficiency by generating a smaller table, called an index, from a specified column or set of columns. These columns, called a key, can be used to enforce uniqueness.

```
create index user_name
on User(name);
```

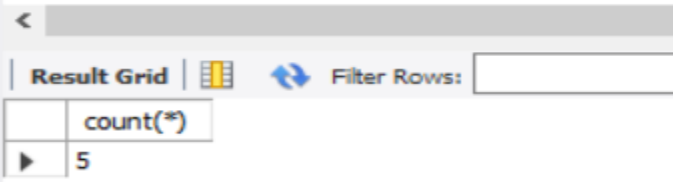
```
1 •    select count(*)
2      from User
3      where name = 'Gopal'
```

Result Grid | Filter Rows:

	count(*)
▶	3

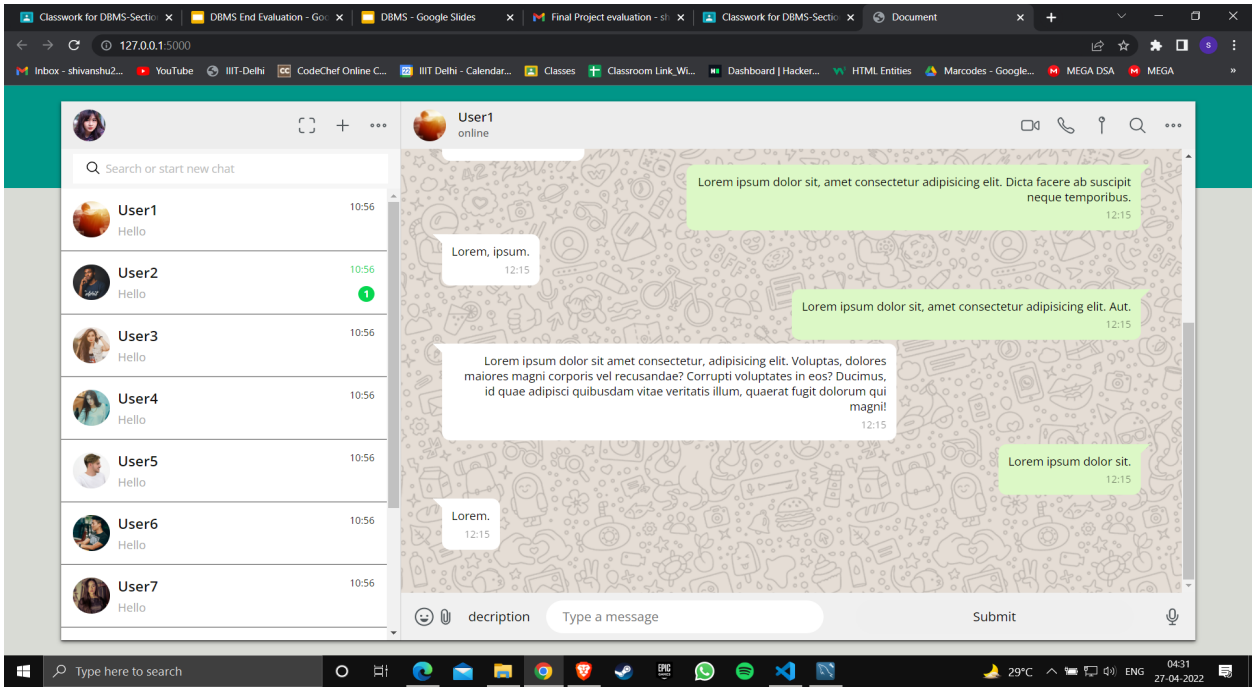
```
create index user_country
on User(country);
```

```
1 •  select count(*)
2    from User
3    where country = 'India'
```

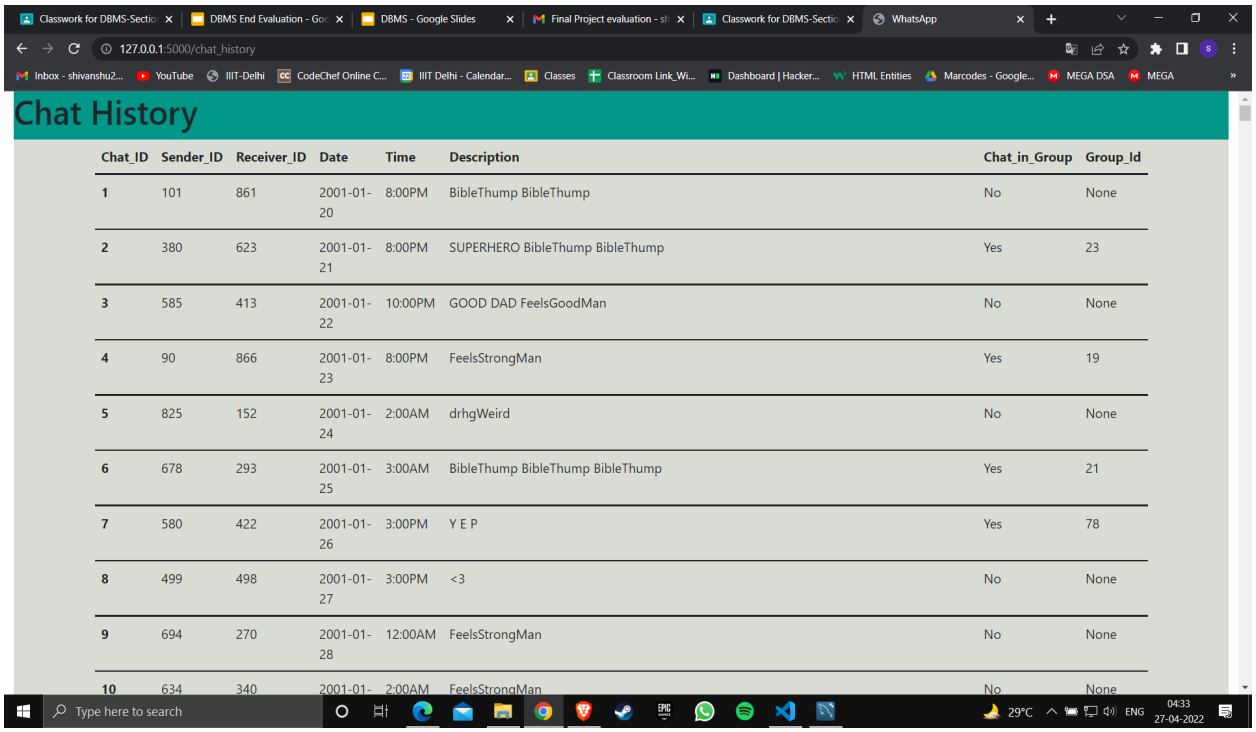



```
create index chatHistory_description
on Chat_History(decription);
```

Frontend



Database



Drive Link

https://drive.google.com/file/d/1fNAzOW0WtSTUhNu3IGVpMGBOJ2HiMv4e/view?usp=sharing