
SOCIAL
MEDIA
MARKETING
TRENDING
TECHNOLOGY
CONTENT
CREATIVE
ADVERTISING

PROJECT NAME

SOCIAL MEDIA MANAGEMENT SYSTEM

NAME: SHRUSHTY KISHOR VANDRE

INTRODUCTION

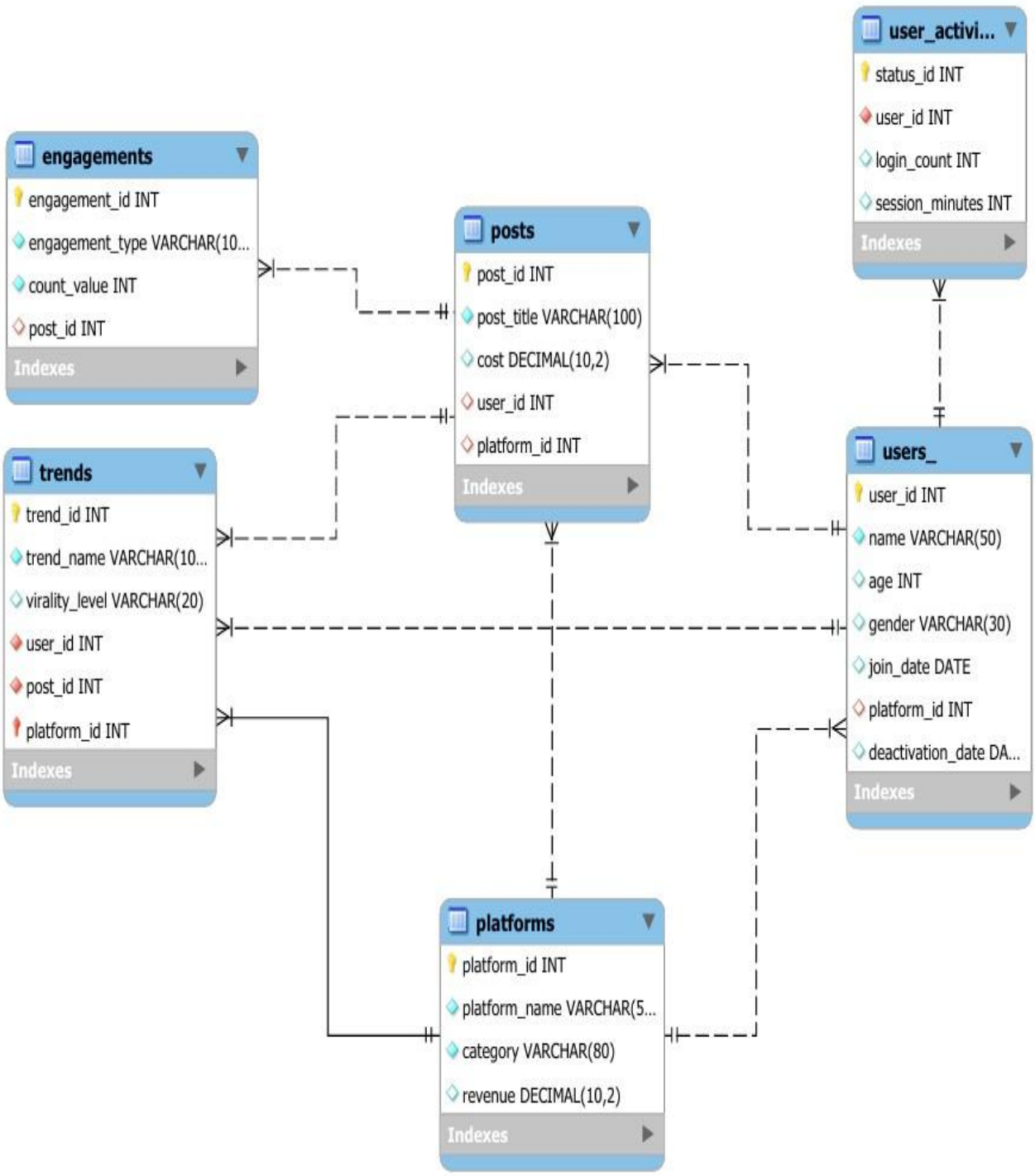
This project is about creating a social media analytics database that manages data about platforms, users, posts, engagements, and trends. The goal is to organize information in a structured way so that user activity, content performance, and platform growth can be analyzed effectively.

The database is built using MySQL and includes six main tables. Data definition commands are used to create and alter tables with primary and foreign keys. Data manipulation queries handle insert, update, and delete operations with sample records. Query language commands such as select, where, group by, and having are used to analyze engagement and revenue. Joins and subqueries provide insights such as costly posts or active users, while window functions like rank and row number help in ranking posts and trends. This setup demonstrates how social media platforms can track and analyze user and content behavior.

In the future, this project can be extended by connecting with live APIs for real-time updates, using predictive analytics to forecast trends, creating dashboards in Power BI or Tableau, scaling for big data, and building recommendation systems for personalized content. These improvements can turn the database into a complete social media monitoring and decision-making tool.

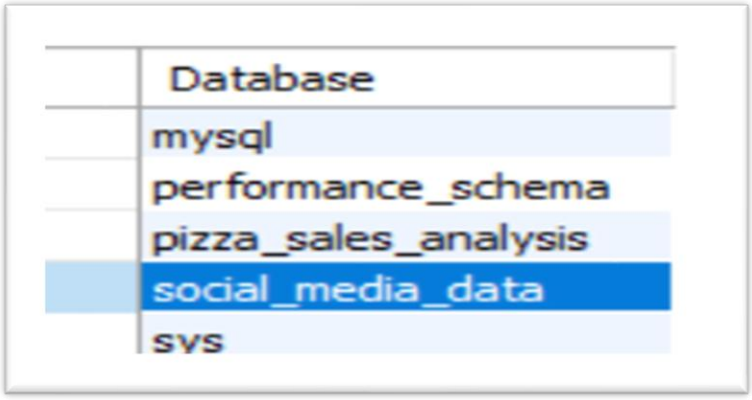
Overall, the project highlights the importance of structured databases in the digital era, where vast amounts of social media data are generated daily. By organizing and analyzing this data efficiently, businesses and creators can gain meaningful insights, improve strategies, and enhance user engagement across platforms.

ER Diagram


user_activi...
status_id INT
user_id INT
login_count INT
session_minutes INT
Indexes
engagements
engagement_id INT
engagement_type VARCHAR(10...
count_value INT
post_id INT
Indexes
posts
post_id INT
post_title VARCHAR(100)
cost DECIMAL(10,2)
user_id INT
platform_id INT
Indexes
trends
trend_id INT
trend_name VARCHAR(10...
virality_level VARCHAR(20)
user_id INT
post_id INT
platform_id INT
Indexes
users_
user_id INT
name VARCHAR(50)
age INT
gender VARCHAR(30)
join_date DATE
platform_id INT
deactivation_date DA...
Indexes
platforms
platform_id INT
platform_name VARCHAR(5...
category VARCHAR(80)
revenue DECIMAL(10,2)
Indexes

- **Databases:**

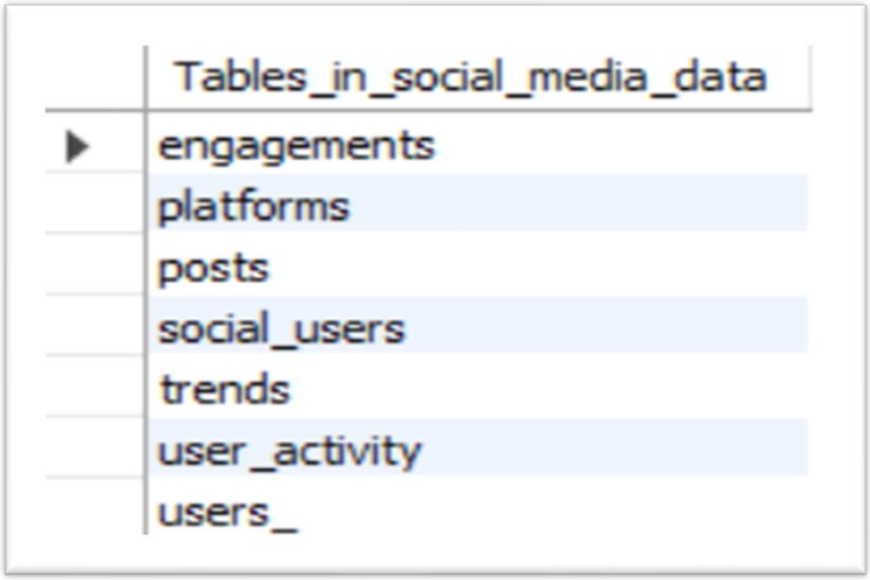
```
CREATE DATABASE Social_media_data;
USE Social_media_data;
show databases;
```



Database
mysql
performance_schema
pizza_sales_analysis
social_media_data
sys

- **Tables in social_media_data Database:**

```
show tables;
```



Tables_in_social_media_data
engagements
platforms
posts
social_users
trends
user_activity
users_

DATA DEFINATION LANGUAGE (DDL):

1) Creating Tables:

A)
```
CREATE TABLE IF NOT EXISTS Platforms(
 platform_id INT PRIMARY KEY,
  platform_name VARCHAR(50) NOT NULL,
 category VARCHAR(80) NOT NULL,
  revenue DECIMAL(10,2) CHECK (revenue > 0));
desc platforms;
```

Field	Type	Null	Key	Default	Extra
platform_id	int	NO	PRI	NULL	
platform_name	varchar(50)	NO		NULL	
category	varchar(80)	NO		NULL	
revenue	decimal(10,2)	YES		NULL	

B)
```
CREATE TABLE IF NOT EXISTS Users_(
 user_id INT PRIMARY KEY,
name VARCHAR(50) NOT NULL,
 age INT CHECK (age > 0),
 gender VARCHAR(30), join_date DATE,
 platform_id INT,deactivation_date DATE, FOREIGN KEY (platform_id) REFERENCES
Platforms(platform_id));
desc Users_;
```

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	
name	varchar(50)	NO		NULL	
age	int	YES		NULL	
gender	varchar(30)	YES		NULL	
join_date	date	YES		NULL	
platform_id	int	YES	MUL	NULL	
deactivation_date	date	YES		NULL	

C) CREATE TABLE IF NOT EXISTS Posts(

post_id INT PRIMARY KEY,

post_title VARCHAR(100) NOT NULL,

cost DECIMAL(10,2),

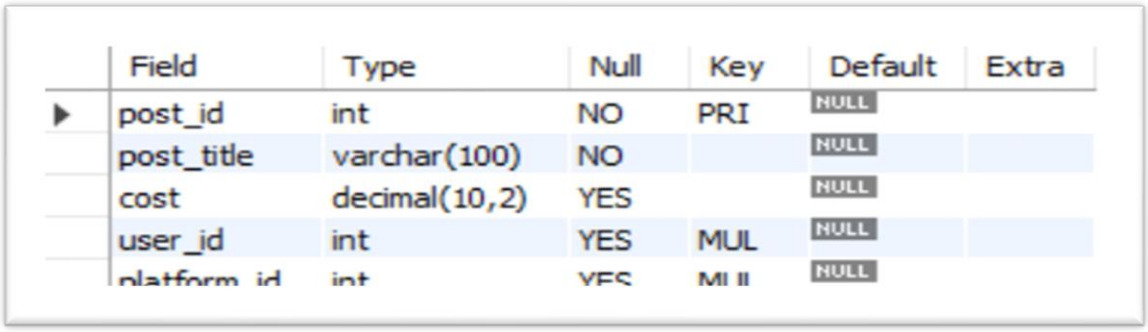
user_id INT,

platform_id INT,

FOREIGN KEY (user_id) REFERENCES Users(user_id),

FOREIGN KEY (platform_id) REFERENCES Platforms(platform_id));

desc posts;



Field	Type	Null	Key	Default	Extra
post_id	int	NO	PRI	NULL	
post_title	varchar(100)	NO		NULL	
cost	decimal(10,2)	YES		NULL	
user_id	int	YES	MUL	NULL	
platform_id	int	YES	MUL	NULL	

D) CREATE TABLE IF NOT EXISTS Engagements(

engagement_id INT PRIMARY KEY,

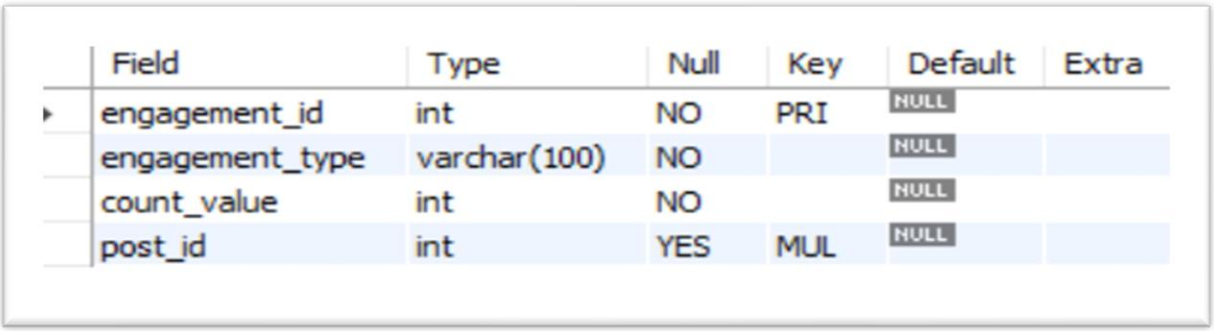
engagement_type VARCHAR(100) NOT NULL,

count_value INT NOT NULL,

post_id INT,

FOREIGN KEY (post_id) REFERENCES Posts(post_id));

desc Engagements;



Field	Type	Null	Key	Default	Extra
engagement_id	int	NO	PRI	NULL	
engagement_type	varchar(100)	NO		NULL	
count_value	int	NO		NULL	
post_id	int	YES	MUL	NULL	

E)
```
CREATE TABLE IF NOT EXISTS User_Activity(
 status_id INT PRIMARY KEY,
 user_id INT NOT NULL,
 login_count INT,
 session_minutes INT CHECK (session_minutes > 0),
 FOREIGN KEY (user_id) REFERENCES Users(user_id));
desc User_activity;
```

Field	Type	Null	Key	Default	Extra
status_id	int	NO	PRI	NULL	
user_id	int	NO	MUL	NULL	
login_count	int	YES		NULL	
session_minutes	int	YES		NULL	

2) Alter Table:

- **Alter Table: Add Column**

a) add column at first position

```
alter table users_ add column phone bigint first;
desc users_;
```

Field	Type	Null	Key	Default	Extra
phone	bigint	YES		NULL	
user_id	int	NO	PRI	NULL	
name	varchar(50)	NO		NULL	
age	int	YES		NULL	
gender	varchar(30)	YES		NULL	
join_date	date	YES		NULL	
platform_id	int	YES	MUL	NULL	
deactivation_date	date	YES		NULL	

b) add column after a specific column (after name)

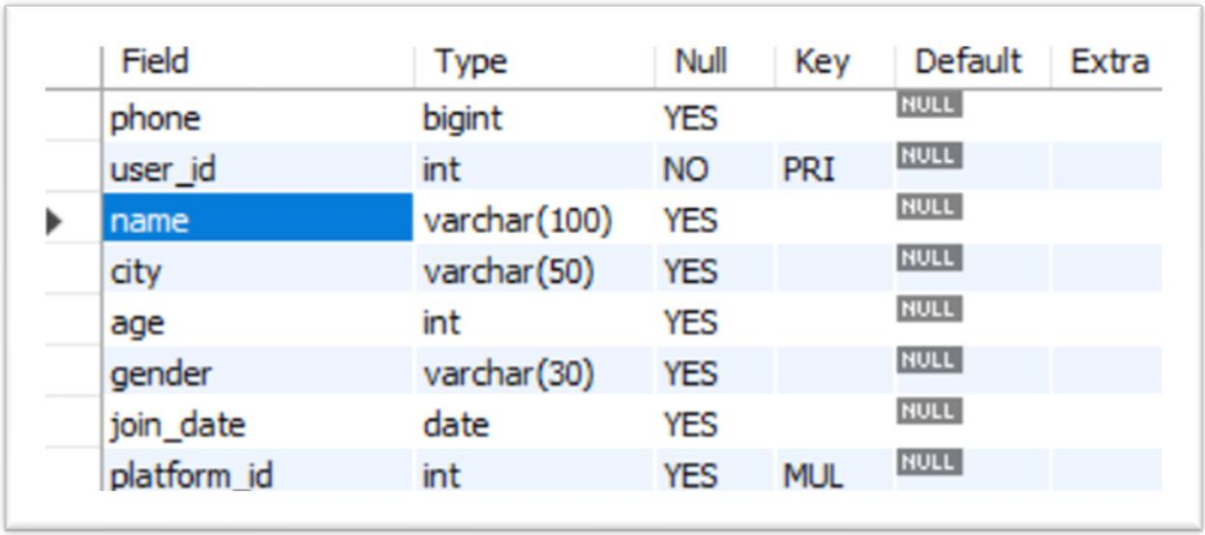
```
alter table users_ add column city varchar(50) after name;
```

Field	Type	Null	Key	Default	Extra
phone	bigint	YES		NULL	
user_id	int	NO	PRI	NULL	
name	varchar(50)	NO		NULL	
city	varchar(50)	YES		NULL	
age	int	YES		NULL	
gender	varchar(30)	YES		NULL	
join_date	date	YES		NULL	
platform_id	int	YES	MUL	NULL	

- **Alter Table : Modify column (datatype)**

a) modify column (change datatype or size)

alter table users_ modify column name varchar(100);

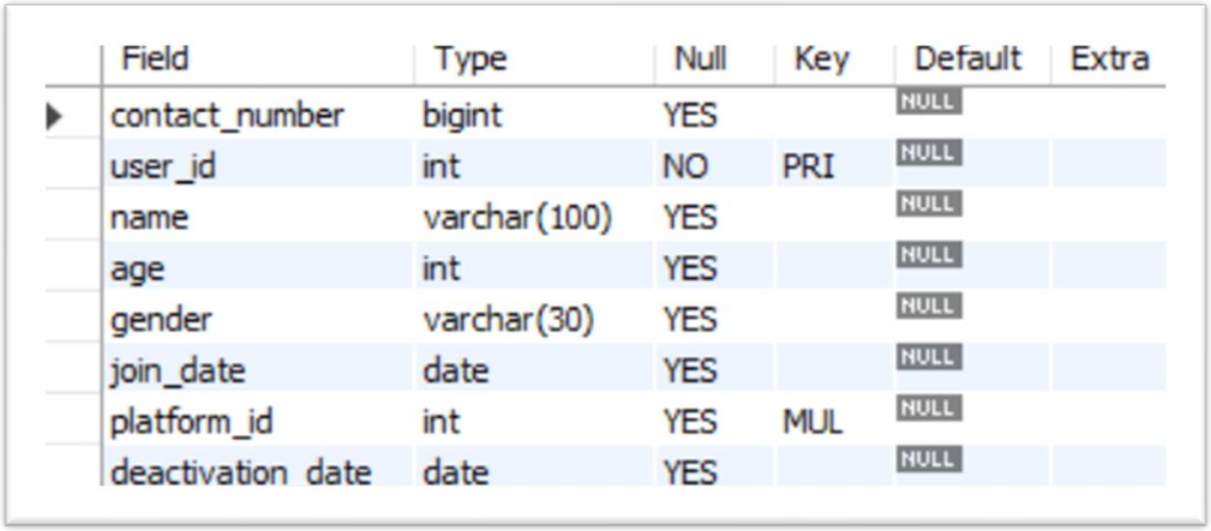


Field	Type	Null	Key	Default	Extra
phone	bigint	YES		NULL	
user_id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
city	varchar(50)	YES		NULL	
age	int	YES		NULL	
gender	varchar(30)	YES		NULL	
join_date	date	YES		NULL	
platform_id	int	YES	MUL	NULL	

- **Alter Table: Drop Column**

b) drop column city

alter table users_ drop column city;

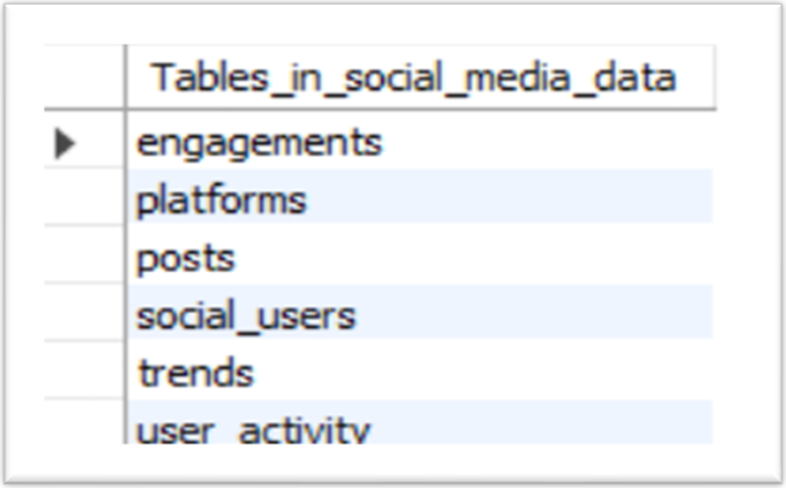


Field	Type	Null	Key	Default	Extra
contact_number	bigint	YES		NULL	
user_id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
age	int	YES		NULL	
gender	varchar(30)	YES		NULL	
join_date	date	YES		NULL	
platform_id	int	YES	MUL	NULL	
deactivation_date	date	YES		NULL	

- **Alter Table: Rename Table**

c) rename table

```
rename table users_ to social_users;
```

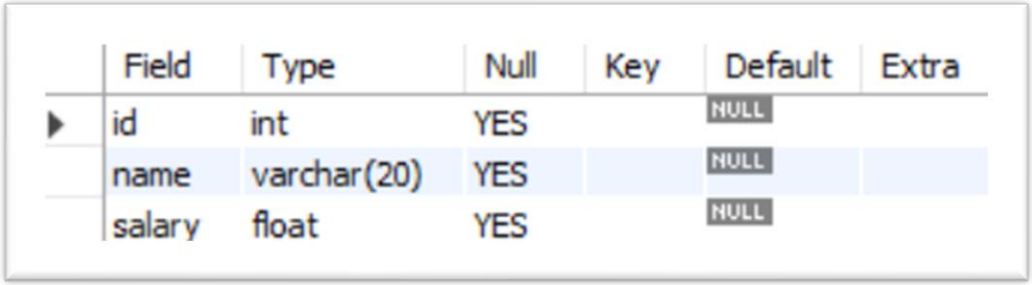


Tables_in_social_media_data
engagements
platforms
posts
social_users
trends
user_activity

3) Truncate Table:

- **Truncate table users**

```
truncate table users;
```

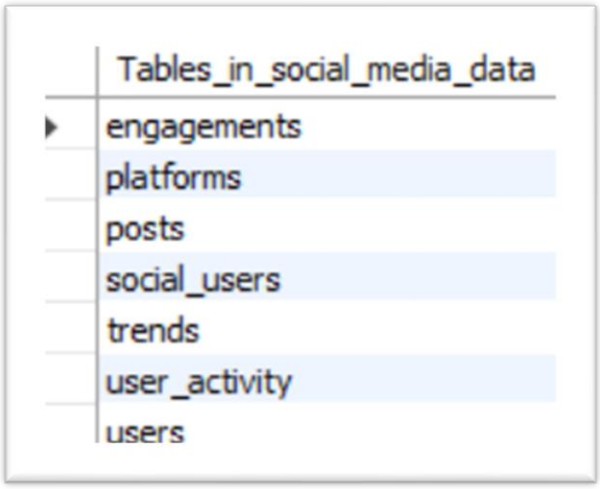


Field	Type	Null	Key	Default	Extra
id	int	YES		NULL	
name	varchar(20)	YES		NULL	
salary	float	YES		NULL	

4) Drop Table:

- **Drop Table users**

```
drop table users;
```

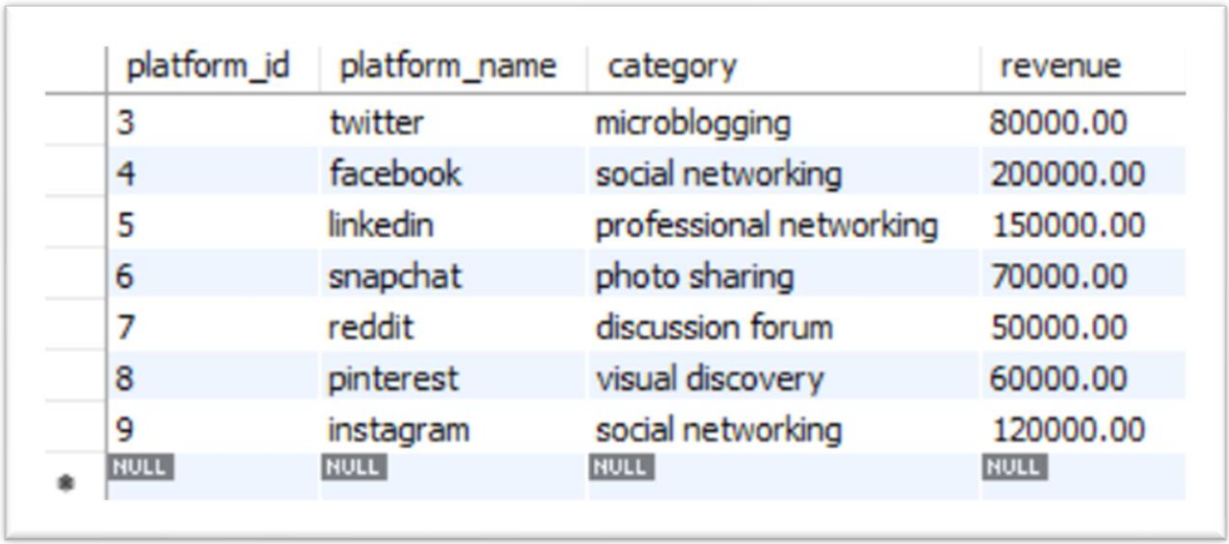


Tables_in_social_media_data
engagements
platforms
posts
social_users
trends
user_activity
users

2) Data Manipulation Language (DML):

Insert Into Table:

```
insert into platforms values (1, 'instagram', 'social networking', 120000.00);
```



platform_id	platform_name	category	revenue
3	twitter	microblogging	80000.00
4	facebook	social networking	200000.00
5	linkedin	professional networking	150000.00
6	snapchat	photo sharing	70000.00
7	reddit	discussion forum	50000.00
8	pinterest	visual discovery	60000.00
9	instagram	social networking	120000.00
NULL	NULL	NULL	NULL

Update into Table:

```
update users_ set age = 26 where user_id = 101;
```

user_id	name	age	gender	join_date	platform_id	deactivation_date
101	aarav mehta	26	male	2024-01-10	1	NULL
102	riya sharma	22	female	2024-02-05	2	NULL
103	kabir singh	28	male	2024-03-01	3	NULL
104	ananya gupta	30	female	2024-03-12	4	NULL
105	ishaan patel	26	male	2024-04-20	5	NULL
106	diya nair	24	female	2024-05-15	6	NULL
107	arjun reddy	27	male	2024-06-01	7	NULL
108	sneha rov	29	female	2024-06-25	8	NULL

Delete From Table:

```
delete from platforms where platform_id = 9;
```

platform_id	platform_name	category	revenue
1	instagram	social networking	120000.00
2	youtube	video sharing	250000.00
3	twitter	microblogging	80000.00
4	facebook	social networking	200000.00
5	linkedin	professional networking	150000.00
6	snapchat	photo sharing	70000.00
7	reddit	discussion forum	50000.00
8	pinterest	visual discovery	60000.00

3) Data Query Language (DQL):

- **Select Query:**

1. **Select Query For Entire Data**

```
select*from users_;
```

user_id	name	age	gender	join_date	platform_id	deactivation_date
101	aarav mehta	26	male	2024-01-10	1	NULL
102	riya sharma	22	female	2024-02-05	2	NULL
103	kabir singh	28	male	2024-03-01	3	NULL
104	ananya gupta	30	female	2024-03-12	4	NULL
105	ishaan patel	26	male	2024-04-20	5	NULL
106	diya nair	24	female	2024-05-15	6	NULL
107	arjun reddy	27	male	2024-06-01	7	NULL
108	sneha roy	29	female	2024-06-25	8	NULL

2. **Select Specific Data (only female users from the users_ table)**

```
select user_id, name, age, gender
from users_
where gender = 'female';
```

user_id	name	age	gender
102	riya sharma	22	female
104	ananya gupta	30	female
106	diya nair	24	female
108	sneha roy	29	female
NULL	NULL	NULL	NULL

3. **Select Query With Changing Column Name (with Alias name):**

display id, username, and user_age instead of the original column names where age should be greater than 25.

```
select user_id as id,  name as username,   age as user_age from users_ u where age > 25;
```

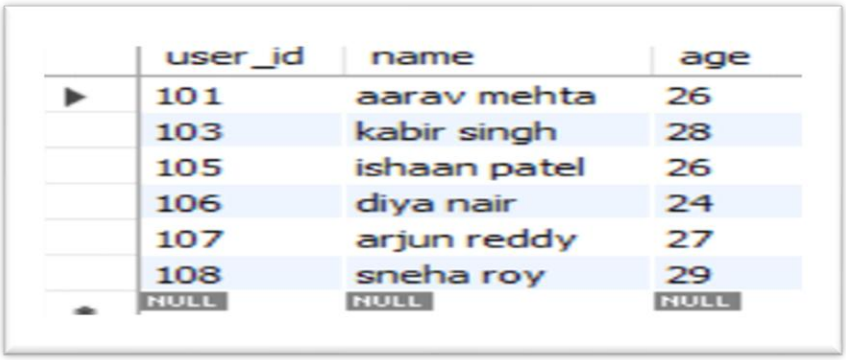
id	username	user_age
101	aarav mehta	26
103	kabir singh	28
104	ananya gupta	30
105	ishaan patel	26
107	arjun reddy	27
108	sneha roy	29

Where Clause:

- **Comparison Operators**

  show users whose age is greater than 22, less than 30, and not equal to 25

  select user_id, name, age from users_ where age > 22 and age < 30 and age != 25;



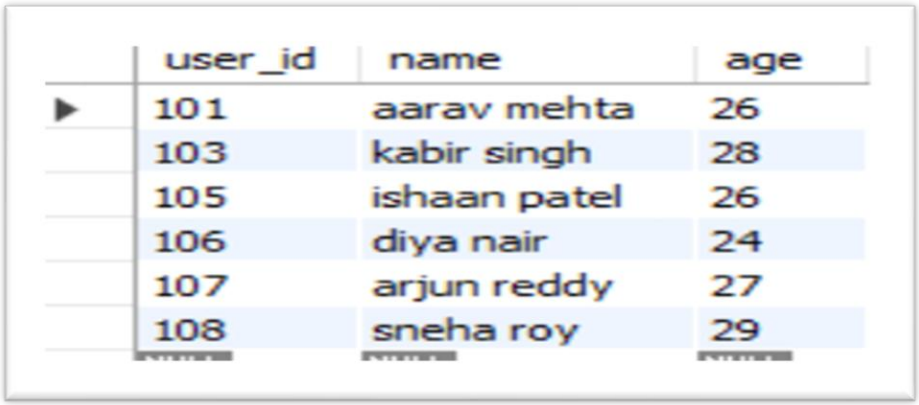
user_id	name	age
101	aarav mehta	26
103	kabir singh	28
105	ishaan patel	26
106	diya nair	24
107	arjun reddy	27
108	sneha roy	29
NULL	NULL	NULL

Logical Operators:

- **AND Operator:**

show users whose age is greater than 25 AND gender is female

select user_id, name, age, gender from users_ where age > 25 and gender = 'female';

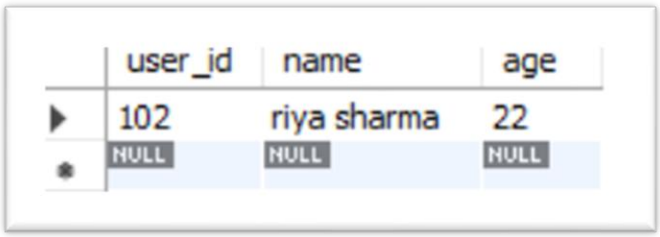


user_id	name	age
101	aarav mehta	26
103	kabir singh	28
105	ishaan patel	26
106	diya nair	24
107	arjun reddy	27
108	sneha roy	29

- **OR Operator:**

  show users whose age is less than 23 OR greater than 30

  select user_id, name, age from users_ where age < 23 or age > 30;

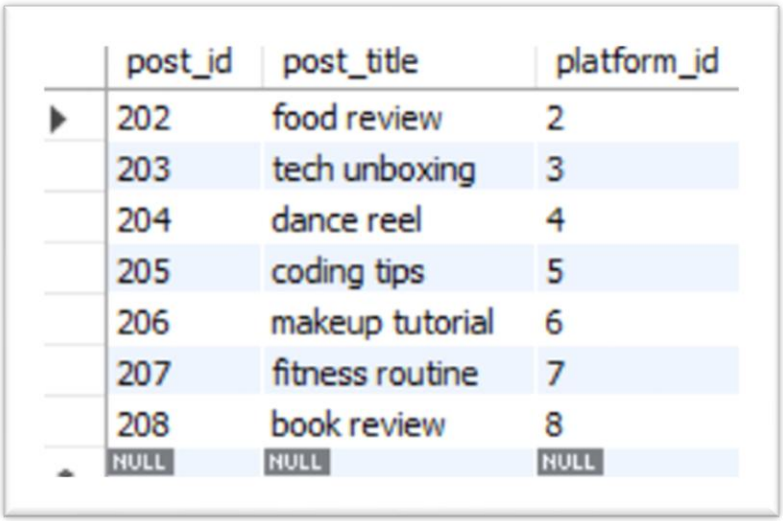


user_id	name	age
102	riya sharma	22
NULL	NULL	NULL

- **NOT Operator:**

  show posts that are NOT from platform_id 1

  select post_id, post_title, platform_id from posts where not platform_id = 1;



post_id	post_title	platform_id
202	food review	2
203	tech unboxing	3
204	dance reel	4
205	coding tips	5
206	makeup tutorial	6
207	fitness routine	7
208	book review	8
NULL	NULL	NULL

- **NOT NULL Operator:**

  show users where deactivation_date is not null

  select user_id, name, deactivation_date from users_ where deactivation_date is not null;

user_id	name	deactivation_date
105	ishaan patel	2024-12-31
NULL	NULL	NULL

- **Range Operator (Between Operator):**

  show posts where cost between 1000 and 2500

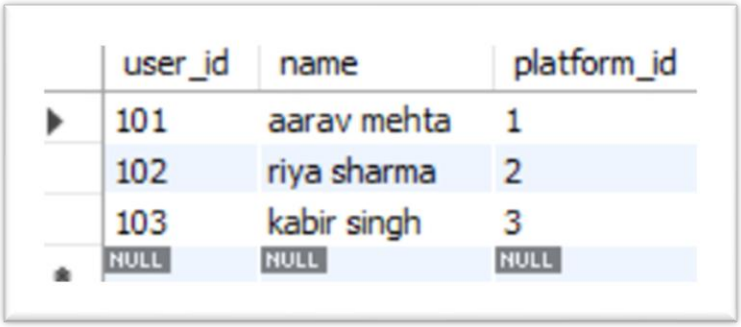
  select post_id, post_title, cost from posts where cost between 1000 and 2500;

post_id	post_title	cost
201	travel vlog	1500.00
202	food review	1200.00
205	coding tips	2500.00
206	makeup tutorial	2200.00
207	fitness routine	1800.00
208	book review	1100.00
NULL	NULL	NULL

- **OR Operator:**

  show users whose platform_id is in the list (1, 2, 3)

  ```
  select user_id, name, platform_id  from users_  where
  platform_id in (1, 2, 3);
  ```

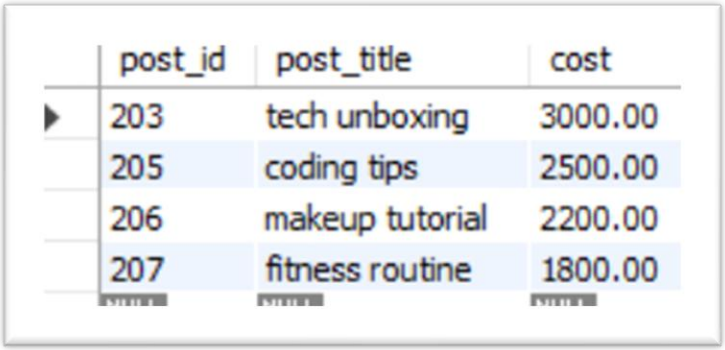


user_id	name	platform_id
101	aarav mehta	1
102	riya sharma	2
103	kabir singh	3
NULL	NULL	NULL

- **ANY Operator:**

show posts whose cost is greater than any cost of platform_id =1

```
select post_id, post_title, cost from posts where cost > any (select
cost from posts where platform_id = 1);
```

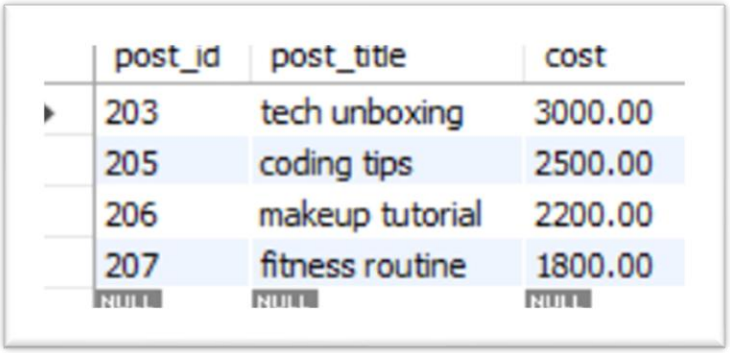


post_id	post_title	cost
203	tech unboxing	3000.00
205	coding tips	2500.00
206	makeup tutorial	2200.00
207	fitness routine	1800.00

- **ALL Operator:**

show posts whose cost is greater than all costs of platform_id = 1

```
select post_id, post_title, cost from posts where cost > all (select
cost from posts where platform_id = 1);
```



post_id	post_title	cost
203	tech unboxing	3000.00
205	coding tips	2500.00
206	makeup tutorial	2200.00
207	fitness routine	1800.00
NULL	NULL	NULL

1. **String Functions:**

- **Like Operator**

find all posts that contain the word 'vlog' in their title

```
select post_id, post_title from posts where post_title like
'%vlog%';
```

post_id	post_title
201	travel vlog
NULL	NULL

- **Concat Operator**

  show user name along with gender in one column

  ```
  select concat(name, ' (', gender, ')') as user_details
  from users_;
  ```

user_details
aarav mehta (male)
riya sharma (female)
kabir singh (male)
ananya gupta (female)
ishaan patel (male)
diya nair (female)
arjun reddy (male)
sneha roy (female)

- **Replace and Reverse Operator:**

  show post titles after replacing 'vlog' with 'video log' and also display them reversed

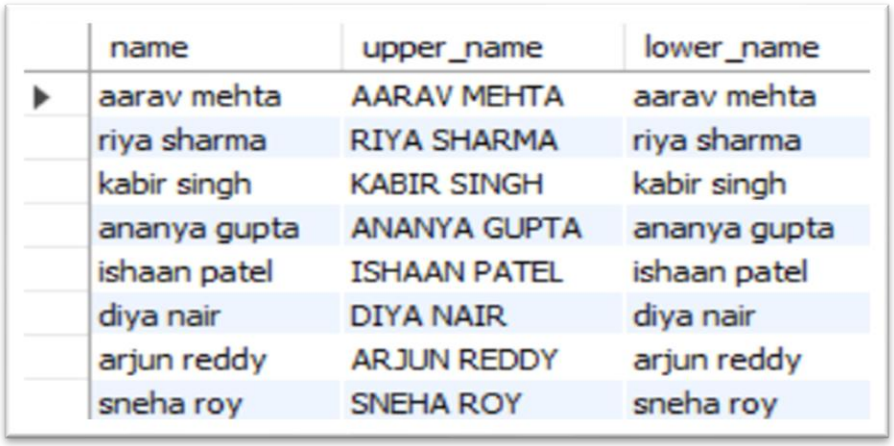
  ```
  select post_id, post_title as original_title, replace(post_title,
  'vlog', 'video log') as replaced_title,  reverse(post_title) as
  reversed_title from posts;
  ```

post_id	original_title	replaced_title	reversed_title
201	travel vlog	travel video log	golv levart
202	food review	food review	weiver doof
203	tech unboxing	tech unboxing	gnixobnu hcet
204	dance reel	dance reel	leer ecnad
205	coding tips	coding tips	spit gnidoc
206	makeup tutorial	makeup tutorial	lairotut puekam
207	fitness routine	fitness routine	enituor ssentif
208	book review	book review	weiver koob

- **Upper and Lower Operator:**

  show user names in both uppercase and lowercase

  select name, upper(name) as upper_name, lower(name) as lower_name from users_;



name	upper_name	lower_name
aarav mehta	AARAV MEHTA	aarav mehta
riya sharma	RIYA SHARMA	riya sharma
kabir singh	KABIR SINGH	kabir singh
ananya gupta	ANANYA GUPTA	ananya gupta
ishaan patel	ISHAAN PATEL	ishaan patel
diya nair	DIYA NAIR	diya nair
arjun reddy	ARJUN REDDY	arjun reddy
sneha roy	SNEHA ROY	sneha roy

- **SubString (Ltrim,Rtrim,Trim):**

clean unwanted spaces in usernames using ltrim, rtrim, and trim

select name as original_name, ltrim(name) as left_trimmed, rtrim(name) as right_trimmed, trim(name) as fully_trimmed from users_;

original_name	left_trimmed	right_trimmed	fully_trimmed
aarav mehta	aarav mehta	aarav mehta	aarav mehta
riya sharma	riya sharma	riya sharma	riya sharma
kabir singh	kabir singh	kabir singh	kabir singh
ananya gupta	ananya gupta	ananya gupta	ananya gupta
ishaan patel	ishaan patel	ishaan patel	ishaan patel
diya nair	diya nair	diya nair	diya nair
arjun reddy	arjun reddy	arjun reddy	arjun reddy
sneha roy	sneha roy	sneha roy	sneha roy

Mathematical Functions:

- **abs() → Absolute Value**

show absolute value of session_minutes difference from 300

```
select user_id, session_minutes, abs(session_minutes - 300) as abs_difference from user_activity;
```

user_id	session_minutes	abs_difference
101	340	40
102	260	40
103	420	120
104	310	10
105	580	280
106	330	30
107	400	100
108	290	10

- **mod() → remainder of division**

  find remainder when session_minutes is divided by 60 (hours)

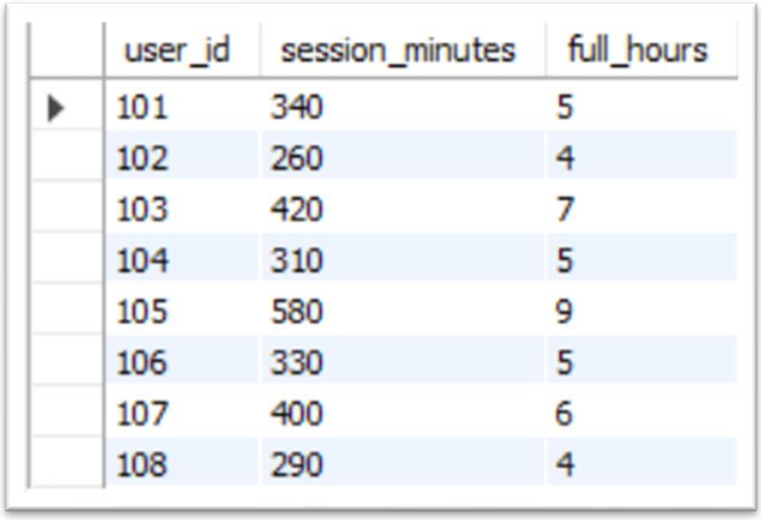
```
select user_id, session_minutes, mod(session_minutes, 60) as remaining_minutes from user_activity;
```

user_id	session_minutes	remaining_minutes
101	340	40
102	260	20
103	420	0
104	310	10
105	580	40
106	330	30
107	400	40
108	290	50

- **floor() → round down:**

  convert session_minutes into whole hours (rounded down)

  select user_id, session_minutes, floor(session_minutes / 60) as full_hours from user_activity;

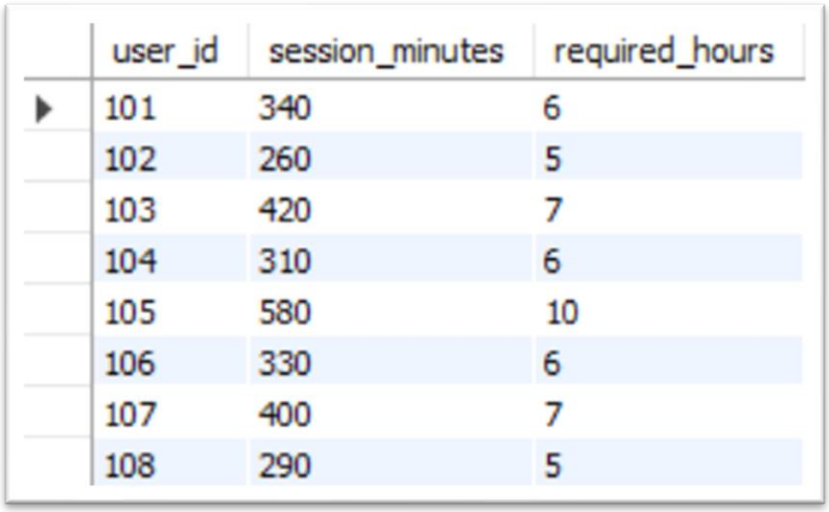


user_id	session_minutes	full_hours
101	340	5
102	260	4
103	420	7
104	310	5
105	580	9
106	330	5
107	400	6
108	290	4

- **ceil() → round up:**

  convert session_minutes into whole hours (rounded up)

  select user_id, session_minutes, ceil(session_minutes / 60) as required_hours from user_activity;

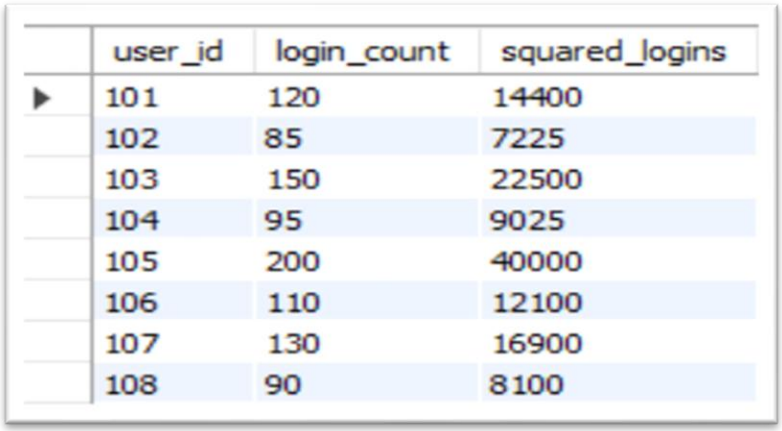


user_id	session_minutes	required_hours
101	340	6
102	260	5
103	420	7
104	310	6
105	580	10
106	330	6
107	400	7
108	290	5

- **pow(x,y) → power:**

  square the login_count of users

  select user_id, login_count, pow(login_count, 2) as squared_logins from user_activity;

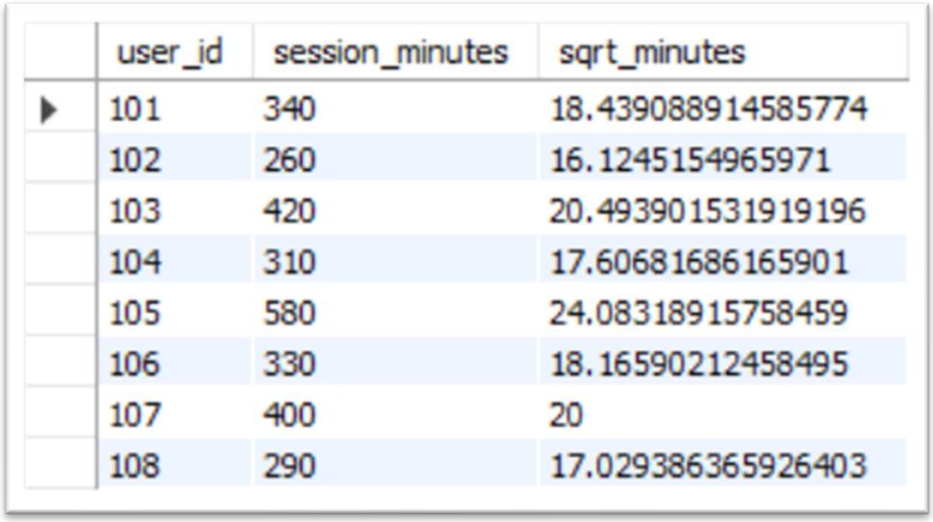


user_id	login_count	squared_logins
101	120	14400
102	85	7225
103	150	22500
104	95	9025
105	200	40000
106	110	12100
107	130	16900
108	90	8100

- **sqrt() → square root:**

  show square root of session_minutes

  select user_id, session_minutes, sqrt(session_minutes) as sqrt_minutes from user_activity;



user_id	session_minutes	sqrt_minutes
101	340	18.439088914585774
102	260	16.1245154965971
103	420	20.493901531919196
104	310	17.60681686165901
105	580	24.08318915758459
106	330	18.16590212458495
107	400	20
108	290	17.029386365926403

- **Aggregate functions (min, max, sum, avg, count):**

```
summary of post costs
select
    min(cost) as min_post_cost,
    max(cost) as max_post_cost,
    sum(cost) as total_cost,
    avg(cost) as avg_cost,
    count(post_id) as total_posts
from posts;
```

min_post_cost	max_post_cost	total_cost	avg_cost	total_posts
800.00	3000.00	14100.00	1762.500000	8

2. Date Functions:

- **curdate() → current date :**

  show today's date along with each user

```
select user_id, name, curdate() as today_date
from users_;
```

user_id	name	today_date
101	aarav mehta	2025-09-14
102	riya sharma	2025-09-14
103	kabir singh	2025-09-14
104	ananya gupta	2025-09-14
105	ishaan patel	2025-09-14
106	diya nair	2025-09-14
107	arjun reddy	2025-09-14
108	sneha roy	2025-09-14

- **now() → current date and time:**

  show current date-time when fetching posts

```
select post_id, post_title, now() as fetched_time
from posts;
```

post_id	post_title	fetched_time
201	travel vlog	2025-09-14 14:05:10
202	food review	2025-09-14 14:05:10
203	tech unboxing	2025-09-14 14:05:10
204	dance reel	2025-09-14 14:05:10
205	coding tips	2025-09-14 14:05:10
206	makeup tutorial	2025-09-14 14:05:10
207	fitness routine	2025-09-14 14:05:10
208	book review	2025-09-14 14:05:10

- **datediff() → difference between two dates (in days):**

  show how many days each user has been on the platform

  select user_id, name, join_date, datediff(curdate(), join_date) as days_on_platform from users_;

user_id	name	join_date	days_on_platform
101	aarav mehta	2024-01-10	613
102	riya sharma	2024-02-05	587
103	kabir singh	2024-03-01	562
104	ananya gupta	2024-03-12	551
105	ishaan patel	2024-04-20	512
106	diya nair	2024-05-15	487
107	arjun reddy	2024-06-01	470
108	sneha roy	2024-06-25	446

- **date_format() → format the date into custom style:**

  show join_date in dd-mm-yyyy format

  select user_id, name,date_format(join_date, '%d-%m-%Y') as formatted_join_date from users_;

user_id	name	formatted_join_date
101	aarav mehta	10-01-2024
102	riya sharma	05-02-2024
103	kabir singh	01-03-2024
104	ananya gupta	12-03-2024
105	ishaan patel	20-04-2024
106	diya nair	15-05-2024
107	arjun reddy	01-06-2024
108	sneha roy	25-06-2024

- **date_add() → add interval to a date:**

  show date when each user will complete 100 days after joining

  ```
  select user_id, name, join_date,  date_add(join_date, interval 100 day) as milestone_date from users_;
  ```

user_id	name	join_date	milestone_date
101	aarav mehta	2024-01-10	2024-04-19
102	riya sharma	2024-02-05	2024-05-15
103	kabir singh	2024-03-01	2024-06-09
104	ananya gupta	2024-03-12	2024-06-20
105	ishaan patel	2024-04-20	2024-07-29
106	diya nair	2024-05-15	2024-08-23
107	arjun reddy	2024-06-01	2024-09-09
108	sneha roy	2024-06-25	2024-10-03

1. **Limit Query:**

   show the first 5 posts only

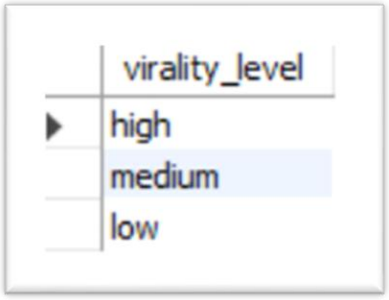
   ```
   select post_id, post_title, cost
   from posts
   limit 5;
   ```

post_id	post_title	cost
201	travel vlog	1500.00
202	food review	1200.00
203	tech unboxing	3000.00
204	dance reel	800.00
205	coding tips	2500.00

2. Distinct Query:

show distinct virality levels from trends table

```
select distinct virality_level
from trends;
```

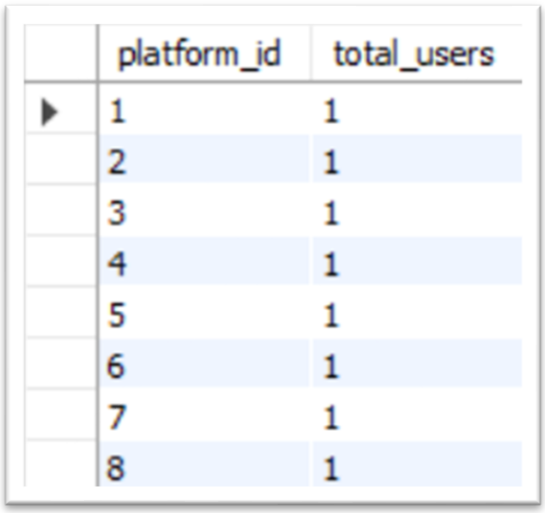


virality_level
high
medium
low

3. Group by Clause:

count how many users are there in each platform

```
select platform_id, count(user_id) as total_users from
users_ group by platform_id;
```

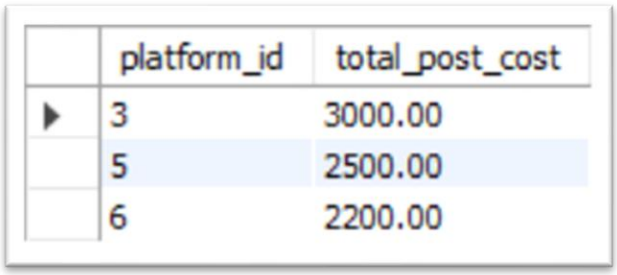


platform_id	total_users
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1

4. Having Clause:

show platforms where total post cost is greater than 5000

```
select platform_id, sum(cost) as total_post_cost from
posts group by platform_id having sum(cost) > 5000;
```



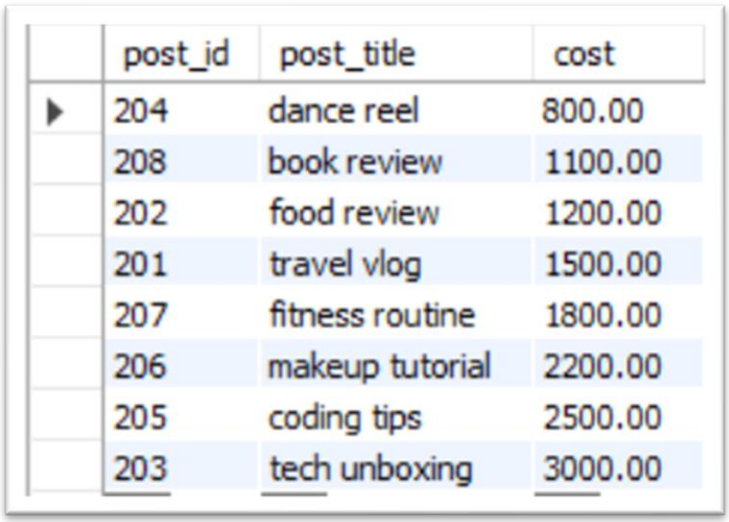
platform_id	total_post_cost
3	3000.00
5	2500.00
6	2200.00

5. Order by Clause:

- **Ascending**

list posts ordered by cost in ascending order

```
select post_id, post_title, cost from posts order by cost
asc;
```



post_id	post_title	cost
204	dance reel	800.00
208	book review	1100.00
202	food review	1200.00
201	travel vlog	1500.00
207	fitness routine	1800.00
206	makeup tutorial	2200.00
205	coding tips	2500.00
203	tech unboxing	3000.00

- **Descending**

  list users ordered by age in descending order

  select user_id, name, age from users_ order by age desc;

user_id	name	age
104	ananya gupta	30
108	sneha roy	29
103	kabir singh	28
107	arjun reddy	27
101	aarav mehta	26
105	ishaan patel	26
106	diya nair	24
102	riya sharma	22

6. Sub-Query:

- **Example**

  show posts whose cost is above the average cost of all posts

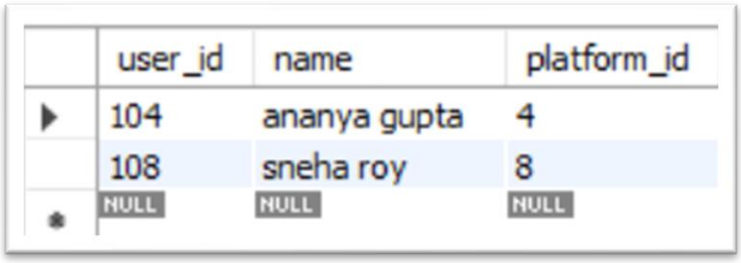
  select post_id, post_title, cost from posts where cost > (select avg(cost) from posts);

post_id	post_title	cost
203	tech unboxing	3000.00
205	coding tips	2500.00
206	makeup tutorial	2200.00
207	fitness routine	1800.00
NULL	NULL	NULL

- **Multi-Row Subquery:**

  find users who are on the same platforms as users older than 28

  select user_id, name, platform_id from users_ where platform_id in (select platform_id from users_ where age > 28);



user_id	name	platform_id
104	ananya gupta	4
108	sneha roy	8
NULL	NULL	NULL

- **Multi-Column Subquery:**

  find posts where (user_id, platform_id) pair exists in users table

  select post_id, post_title, user_id, platform_id from posts where (user_id, platform_id) in (select user_id, platform_id from users_);

post_id	post_title	user_id	platform_id
201	travel vlog	101	1
202	food review	102	2
203	tech unboxing	103	3
204	dance reel	104	4
205	coding tips	105	5
206	makeup tutorial	106	6
207	fitness routine	107	7
208	book review	108	8

- **Multi-Table Subquery:**

  show users who created posts on high-revenue

  select user_id, name from users_ where platforms (revenue > 150000) select user_id, name from users_ where platform_id in (select platform_id from platforms where revenue > 150000);

user_id	name
102	riya sharma
104	ananya gupta
NULL	NULL

7. JOINS:

- **INNER JOIN (only matching rows):**

list posts along with their user names

```
select p.post_id, p.post_title, u.name as user_name
from posts p  inner join users_ u on p.user_id = u.user_id;
```

post_id	post_title	user_name
201	travel vlog	aarav mehta
202	food review	riya sharma
203	tech unboxing	kabir singh
204	dance reel	ananya gupta
205	coding tips	ishaan patel
206	makeup tutorial	diya nair
207	fitness routine	arjun reddy
208	book review	sneha roy

- **LEFT JOIN (all rows from left, matching from right):**

list all users and their posts (include users with no posts)

```
select u.user_id, u.name, p.post_title from users_ u
left join posts p on u.user_id = p.user_id;
```

	user_id	name	post_title
▶	101	aarav mehta	travel vlog
	102	riya sharma	food review
	103	kabir singh	tech unboxing
	104	ananya gupta	dance reel
	105	ishaan patel	coding tips
	106	diya nair	makeup tutorial
	107	arjun reddy	fitness routine
	108	sneha roy	book review

- **FULL OUTER JOIN (all rows from both, using union):**

list all platforms and their posts (include platforms without posts and posts without platforms)

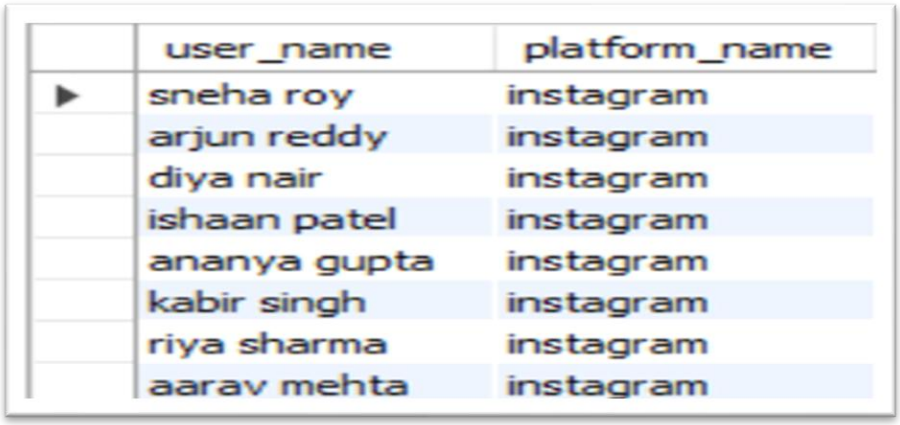
```
select pl.platform_name, p.post_title
from platforms pl
left join posts p on pl.platform_id = p.platform_id
union
select pl.platform_name, p.post_title
from platforms pl
right join posts p on pl.platform_id = p.platform_id;
```

platform_name	post_title
instagram	travel vlog
youtube	food review
twitter	tech unboxing
facebook	dance reel
linkedin	coding tips
snapchat	makeup tutorial
reddit	fitness routine
pinterest	book review

- **CROSS JOIN (Cartesian product, all combinations):**

  show all possible user and platform combinations

```
select u.name as user_name, pl.platform_name
from users_ u
cross join platforms pl;
```



	user_name	platform_name
▶	sneha roy	instagram
	arjun reddy	instagram
	diya nair	instagram
	ishaan patel	instagram
	ananya gupta	instagram
	kabir singh	instagram
	riya sharma	instagram
	aarav mehta	instagram

Output will be continued as same for rest of the platforms.

- **SELF JOIN:**

  find pairs of users who are on the same platform

```
select u1.name as user1, u2.name as user2, u1.platform_id
from users_ u1
join users_ u2 on u1.platform_id = u2.platform_id
where u1.user_id < u2.user_id;
```

	user1	user2	platform_id
▶	aarav mehta	riya sharma	1

8. WINDOWS FUNCTIONS:

- **rownumber() – without partition:**

  assign row numbers to posts ordered by cost (highest first)

```
select post_id, post_title, platform_id, cost,
row_number() over(order by cost desc) as row_num
from posts;
```

post_id	post_title	platform_id	cost	row_num
203	tech unboxing	3	3000.00	1
205	coding tips	5	2500.00	2
206	makeup tutorial	6	2200.00	3
207	fitness routine	7	1800.00	4
201	travel vlog	1	1500.00	5
202	food review	2	1200.00	6
208	book review	8	1100.00	7
204	dance reel	4	800.00	8

- **rownumber() with partition:**

  assign row numbers to posts within each platform (highest cost first)

```
select post_id, post_title, platform_id, cost,
row_number() over(partition by platform_id order by cost desc)
as row_num
from posts;
```

post_id	post_title	platform_id	cost	row_num
201	travel vlog	1	1500.00	1
202	food review	2	1200.00	1
203	tech unboxing	3	3000.00	1
204	dance reel	4	800.00	1
205	coding tips	5	2500.00	1
206	makeup tutorial	6	2200.00	1
207	fitness routine	7	1800.00	1
208	book review	8	1100.00	1

- **RANK ():**

1. **Without partition:**

rank posts by cost across all platforms

select post_id, post_title, platform_id, cost, rank() over(order by cost desc) as rank_num from posts;

post_id	post_title	platform_id	cost	rank_num
203	tech unboxing	3	3000.00	1
205	coding tips	5	2500.00	2
206	makeup tutorial	6	2200.00	3
207	fitness routine	7	1800.00	4
201	travel vlog	1	1500.00	5
202	food review	2	1200.00	6
208	book review	8	1100.00	7
204	dance reel	4	800.00	8

2. **With Partition:**

rank posts by cost within each platform

select post_id, post_title, platform_id, cost, rank() over(partition by platform_id order by cost desc) as rank_num from posts;

post_id	post_title	platform_id	cost	rank_num
201	travel vlog	1	1500.00	1
202	food review	2	1200.00	1
203	tech unboxing	3	3000.00	1
204	dance reel	4	800.00	1
205	coding tips	5	2500.00	1
206	makeup tutorial	6	2200.00	1
207	fitness routine	7	1800.00	1
208	book review	8	1100.00	1

- **DENSERANK():**

  1. Without Partition:

  dense rank posts by cost across all platforms

  select post_id, post_title, platform_id, cost,
  dense_rank() over(order by cost desc) as
  dense_rank_num from posts;

post_id	post_title	platform_id	cost	dense_rank_num
203	tech unboxing	3	3000.00	1
205	coding tips	5	2500.00	2
206	makeup tutorial	6	2200.00	3
207	fitness routine	7	1800.00	4
201	travel vlog	1	1500.00	5
202	food review	2	1200.00	6
208	book review	8	1100.00	7
204	dance reel	4	800.00	8

  2. With Partition:

  dense rank posts by cost within each platform

  select post_id, post_title, platform_id, cost,
  dense_rank() over(partition by platform_id
  order by cost desc) as dense_rank_num from posts;

post_id	post_title	platform_id	cost	dense_rank_num
201	travel vlog	1	1500.00	1
202	food review	2	1200.00	1
203	tech unboxing	3	3000.00	1
204	dance reel	4	800.00	1
205	coding tips	5	2500.00	1
206	makeup tutorial	6	2200.00	1
207	fitness routine	7	1800.00	1
208	book review	8	1100.00	1

- **first_value():**

  Q: find the cheapest post on each platform

  select post_id, post_title, platform_id, cost, first_value(post_title) over(partition by platform_id order by cost asc) as cheapest_post from posts;

post_id	post_title	platform_id	cost	cheapest_post
201	travel vlog	1	1500.00	travel vlog
202	food review	2	1200.00	food review
203	tech unboxing	3	3000.00	tech unboxing
204	dance reel	4	800.00	dance reel
205	coding tips	5	2500.00	coding tips
206	makeup tutorial	6	2200.00	makeup tutorial
207	fitness routine	7	1800.00	fitness routine
208	book review	8	1100.00	book review

- **last_value():**

  Q: find the most expensive post overall

  select post_id, post_title, cost, last_value(post_title) over(order by cost asc rows between unbounded preceding and unbounded following) as most_expensive_post from posts;

post_id	post_title	cost	most_expensive_post
204	dance reel	800.00	tech unboxing
208	book review	1100.00	tech unboxing
202	food review	1200.00	tech unboxing
201	travel vlog	1500.00	tech unboxing
207	fitness routine	1800.00	tech unboxing
206	makeup tutorial	2200.00	tech unboxing
205	coding tips	2500.00	tech unboxing
203	tech unboxing	3000.00	tech unboxing

- **lead()**

  Q: show each post and the next higher cost post

  select post_id, post_title, cost, lead(post_title) over(order by cost asc) as next_higher_cost_post from posts;

post_id	post_title	cost	next_higher_cost_post
204	dance reel	800.00	book review
208	book review	1100.00	food review
202	food review	1200.00	travel vlog
201	travel vlog	1500.00	fitness routine
207	fitness routine	1800.00	makeup tutorial
206	makeup tutorial	2200.00	coding tips
205	coding tips	2500.00	tech unboxing
203	tech unboxing	3000.00	NULL

- **lag():**

  Q: Show each user's login count and also display the previous user's login count to compare activity trends.

  select user_id, login_count,lag(login_count, 1, 0) over(order by user_id) as previous_user_login_count from user_activity;

user_id	login_count	previous_user_login_count
101	120	0
102	85	120
103	150	85
104	95	150
105	200	95
106	110	200
107	130	110
108	90	130

- **nth_value():**

  Q: show the 2nd cheapest post overall

  select post_id, post_title, cost, nth_value(post_title, 2) over(order by cost asc rows between unbounded preceding and unbounded following) as second_cheapest_post from posts;

post_id	post_title	cost	second_cheapest_post
204	dance reel	800.00	book review
208	book review	1100.00	book review
202	food review	1200.00	book review
201	travel vlog	1500.00	book review
207	fitness routine	1800.00	book review
206	makeup tutorial	2200.00	book review
205	coding tips	2500.00	book review
203	tech unboxing	3000.00	book review

- **ntile():**

  Q: divide posts into 4 groups based on cost (quartiles)

  select post_id, post_title, cost, ntile(4) over(order by cost desc) as cost_quartile from posts;

post_id	post_title	cost	cost_quartile
203	tech unboxing	3000.00	1
205	coding tips	2500.00	1
206	makeup tutorial	2200.00	2
207	fitness routine	1800.00	2
201	travel vlog	1500.00	3
202	food review	1200.00	3
208	book review	1100.00	4
204	dance reel	800.00	4