Instagram Database Management System

INTRODUCTION

Instagram, a social media giant, relies heavily on its database management system to store and manage user data, posts, comments, likes, and followers. Effective database design is crucial for ensuring data consistency, scalability, and performance. This project aims to design and implement a simplified database management system for Instagram using SQL.

Objective:

The primary objective of this project is to create a relational database management system that can efficiently store and manage Instagram's user data, posts, comments, likes, and followers. The system should support basic functionalities such as:

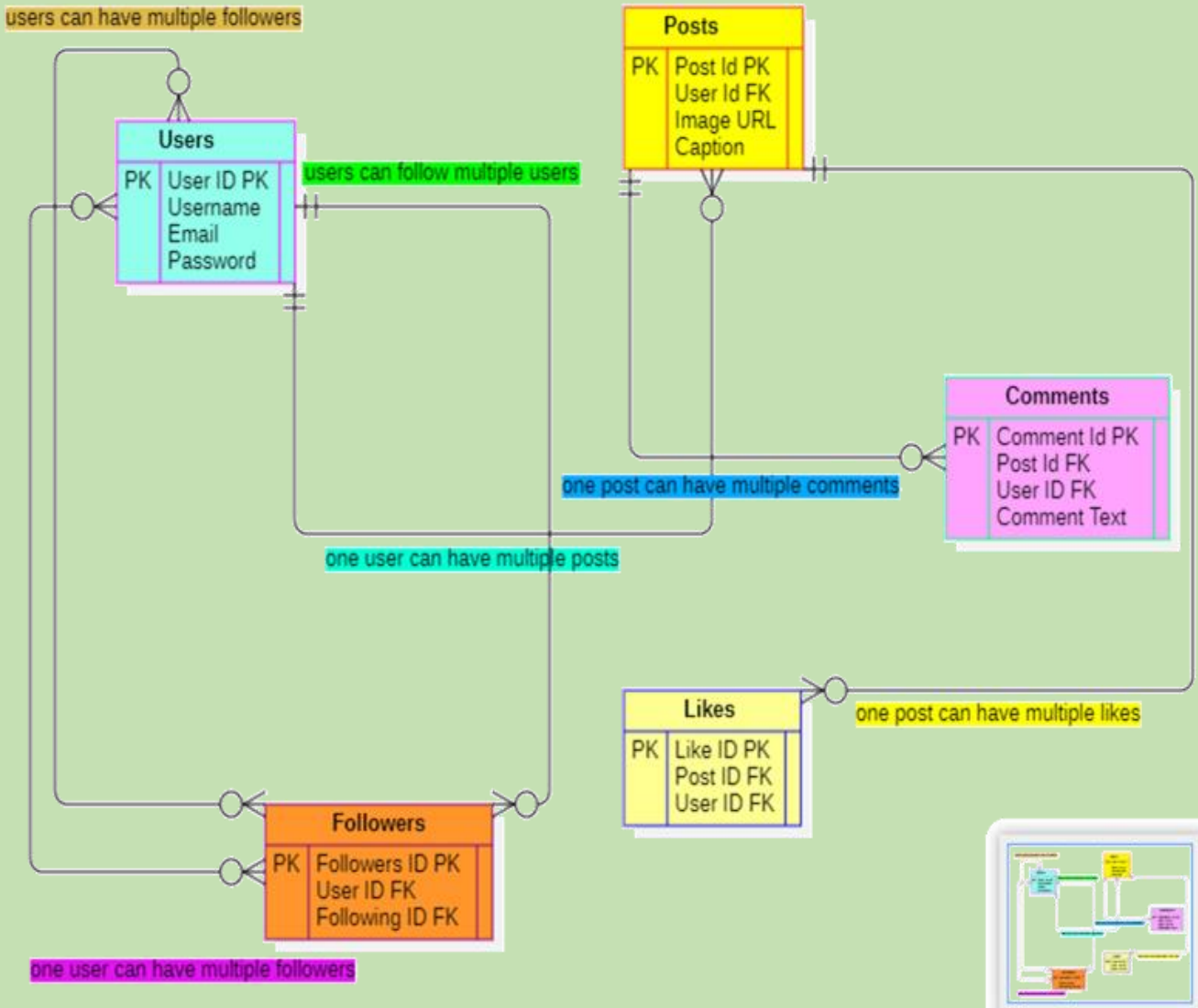
- User registration and login
- Posting and commenting
- Liking and following
- Data retrieval and analysis

Database Requirements:

The database management system will be designed to meet the following requirements:

- Store user information (username, email, password)
- Manage posts (image URL, caption)
- Handle comments (text, user ID, post ID)
- Track likes (user ID, post ID)
- Maintain follower relationships (user ID, following ID)

ER DIAGRAM



Vaishnavi Leharkar

Database Design

Databases: Instagram Database Management System

Tables:

1) **Users**
2) **Posts**
3) **Comments**
4) **Likes**
5) **Followers**

Vaishnavi Leharkar

Create database
CREATE DATABASE Instagram;

Use database
USE Instagram;

Creating Table

a) User_ID INT PRIMARY KEY,
Username VARCHAR(255) UNIQUE,
Email VARCHAR(255) UNIQUE,
Password VARCHAR(255));

```
CREATE DATABASE Instagram;
USE Instagram;

CREATE TABLE Users (
  User_ID INT PRIMARY KEY,
  Username VARCHAR(255) UNIQUE,
  Email VARCHAR(255) UNIQUE,
  Password VARCHAR(255));
```

b) CREATE TABLE Posts (
Post_ID INT PRIMARY KEY,
User_ID INT,
Image_URL VARCHAR(255),
Caption TEXT,
FOREIGN KEY (User_ID) REFERENCES Users(User_ID));

```
CREATE TABLE Posts (
Post_ID INT PRIMARY KEY,
User_ID INT,
Image_URL VARCHAR(255),
Caption TEXT,
FOREIGN KEY (User_ID) REFERENCES Users(User_ID));
```

c) CREATE TABLE Comments (
Comment_ID INT PRIMARY KEY,
Post_ID INT,
User_ID INT,
Comment_Text TEXT,
FOREIGN KEY (Post_ID) REFERENCES Posts(Post_ID),
FOREIGN KEY (User_ID) REFERENCES Users(User_ID));

```
CREATE TABLE Comments (
Comment_ID INT PRIMARY KEY,
Post_ID INT,
User_ID INT,
Comment_Text TEXT,
FOREIGN KEY (Post_ID) REFERENCES Posts(Post_ID),
FOREIGN KEY (User_ID) REFERENCES Users(User_ID));
```

d) CREATE TABLE Followers (
Follower_ID INT PRIMARY KEY,
User_ID INT,
Following_ID INT,
FOREIGN KEY (User_ID) REFERENCES Users(User_ID),
FOREIGN KEY (Following_ID) REFERENCES Users(User_ID));

```
CREATE TABLE Followers (
Follower_ID INT PRIMARY KEY,
User_ID INT,
Following_ID INT,
FOREIGN KEY (User_ID) REFERENCES Users(User_ID),
FOREIGN KEY (Following_ID) REFERENCES Users(User_ID));
```

e) CREATE TABLE Likes (
Like_ID INT PRIMARY KEY,
Post_ID INT,
User_ID INT,
FOREIGN KEY (Post_ID) REFERENCES Posts(Post_ID),
FOREIGN KEY (User_ID) REFERENCES Users(User_ID));

```
CREATE TABLE Likes (
Like_ID INT PRIMARY KEY,
Post_ID INT,
User_ID INT,
FOREIGN KEY (Post_ID) REFERENCES Posts(Post_ID),
FOREIGN KEY (User_ID) REFERENCES Users(User_ID)
);
```

Tables in databases

```
show databases;
show tables;
```

Result Grid | Filter Rows: | Export:

Tables_in_instagram
comments
followers
likes
posts
users

Data Definition language (DDL)

1) Creating Tables:

a) Users:

```
611 •    Desc users;
612
```

Result Grid | Filter Rows: | Export: | Wr

Field	Type	Null	Key	Default	Extra
User_ID	int	NO	PRI	NULL	
Username	varchar(255)	YES	UNI	NULL	
Email	varchar(255)	YES	UNI	NULL	
Password	varchar(255)	YES		NULL	

b) Posts:

```
612 •    Desc posts;
```

Result Grid | Filter Rows: | Export: | Wra

Field	Type	Null	Key	Default	Extra
Post_ID	int	NO	PRI	NULL	
User_ID	int	YES	MUL	NULL	
Image_URL	varchar(255)	YES		NULL	
Caption	text	YES		NULL	

c) Comments:

```
613 •  Desc comments;
```

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
Comment_ID	int	NO	PRI	NULL	
Post_ID	int	YES	MUL	NULL	
User_ID	int	YES	MUL	NULL	
Comment_Text	text	YES		NULL	

d) Likes:

```
614 •  Desc Likes;
615
616
```

Result Grid | Filter Rows: | Expo

Field	Type	Null	Key	Default	Extra
Like_ID	int	NO	PRI	NULL	
Post_ID	int	YES	MUL	NULL	
User_ID	int	YES	MUL	NULL	

e) Followers:

```
615 •  desc followers;
616
```

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
Follower_ID	int	NO	PRI	NULL	
User_ID	int	YES	MUL	NULL	
Following_ID	int	YES	MUL	NULL	

2) Alter Table:

a) Alter table add column:

```
ALTER TABLE Users ADD COLUMN Phone VARCHAR(20);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

Field	Type	Null	Key	Default	Extra
User_ID	int	NO	PRI	NULL	
Username	varchar(255)	YES	UNI	NULL	
Email	varchar(255)	YES	UNI	NULL	
Password	varchar(255)	YES		NULL	
Phone	varchar(20)	YES		NULL	

b) Alter table modify column:

```
ALTER TABLE Users modify Phone VARCHAR(11);
```

Result Grid | Filter Rows: | Export: | Wrap C

Field	Type	Null	Key	Default	Extra
User_ID	int	NO	PRI	NULL	
Username	varchar(255)	YES	UNI	NULL	
Email	varchar(255)	YES	UNI	NULL	
Password	varchar(255)	YES		NULL	
Phone	varchar(11)	YES		NULL	

c) Alter table rename column:

```
ALTER TABLE Users RENAME COLUMN Username TO FullName;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
User_ID	int	NO	PRI	NULL	
FullName	varchar(255)	YES	UNI	NULL	
Email	varchar(255)	YES	UNI	NULL	
Password	varchar(255)	YES		NULL	
Phone	varchar(11)	YES		NULL	

d) Alter table drop column:

```
ALTER TABLE Users DROP COLUMN Phone;
```

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
User_ID	int	NO	PRI	NULL	
FullName	varchar(255)	YES	UNI	NULL	
Email	varchar(255)	YES	UNI	NULL	
Password	varchar(255)	YES		NULL	

e) Rename table:

```
RENAME TABLE Users TO UserAccounts;
Desc UserAccounts;
```

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
User_ID	int	NO	PRI	NULL	
FullName	varchar(255)	YES	UNI	NULL	
Email	varchar(255)	YES	UNI	NULL	
Password	varchar(255)	YES		NULL	

f) Truncate table:

```
CREATE TABLE Users1 (
    User_ID INT PRIMARY KEY,
    Username VARCHAR(255) UNIQUE,
    Email VARCHAR(255) UNIQUE,
    Password VARCHAR(255));

Truncate users1;
```

g) Drop table:

```
CREATE TABLE Users1 (
    User_ID INT PRIMARY KEY,
    Username VARCHAR(255) UNIQUE,
    Email VARCHAR(255) UNIQUE,
    Password VARCHAR(255));

desc Users1;
Drop table Users1;
```

Data Manipulation language (DML)

1) Insert into table

```
INSERT INTO UserAccounts (User_ID, FullName, Email, Password) VALUES
(1, 'John Doe', 'john.doe@example.com', 'password123'),
(2, 'Jane Doe', 'jane.doe@example.com', 'password456'),
(3, 'Bob Smith', 'bob.smith@example.com', 'password789'),
(4, 'Alice Johnson', 'alice.johnson@example.com', 'password1011'),
(5, 'Mike Brown', 'mike.brown@example.com', 'password1213');
```

2) Update into table

```
UPDATE UserAccounts SET FullName = 'John Doe Jr.'
WHERE User_ID = 1;
```

3) Delete into table

```
DELETE FROM UserAccounts WHERE User_ID = 5;
```

Data Query Language (DQL)

1) Select query:

```
701 •   select * from UserAccounts;
```

Result Grid | Filter Rows: | Edit:

User_ID	FullName	Email	Password
1	John Doe Jr.	john.doe@example.com	password123
2	Jane Doe	jane.doe@example.com	password456
3	Bob Smith	bob.smith@example.com	password789
4	Alice Johnson	alice.johnson@example.com	password1011
5	Mike Brown	mike.brown@example.com	password1213
NULL	NULL	NULL	NULL

a) Order by query ASC:

```
702 •   SELECT * FROM UserAccounts ORDER BY FullName ASC;
```

Result Grid | Filter Rows: | Edit:

User_ID	FullName	Email	Password
4	Alice Johnson	alice.johnson@example.com	password1011
3	Bob Smith	bob.smith@example.com	password789
2	Jane Doe	jane.doe@example.com	password456
1	John Doe Jr.	john.doe@example.com	password123
5	Mike Brown	mike.brown@example.com	password1213
NULL	NULL	NULL	NULL

b) Order by query DESC:

```
702 •   SELECT * FROM UserAccounts ORDER BY FullName DESC;
```

Result Grid | Filter Rows: | Edit:

User_ID	FullName	Email	Password
5	Mike Brown	mike.brown@example.com	password1213
1	John Doe Jr.	john.doe@example.com	password123
2	Jane Doe	jane.doe@example.com	password456
3	Bob Smith	bob.smith@example.com	password789
4	Alice Johnson	alice.johnson@example.com	password1011
NULL	NULL	NULL	NULL

c) Limit query:

```
702 •  SELECT * FROM UserAccounts limit 3;
```

Result Grid | Filter Rows: | Edit:

User_ID	FullName	Email	Password
1	John Doe Jr.	john.doe@example.com	password123
2	Jane Doe	jane.doe@example.com	password456
3	Bob Smith	bob.smith@example.com	password789
NULL	NULL	NULL	NULL

d) Select query with specific column:

```
703 •  SELECT User_ID, FullName, Email FROM UserAccounts;
```

Result Grid | Filter Rows: | Edit: | Expo

User_ID	FullName	Email
1	John Doe Jr.	john.doe@example.com
2	Jane Doe	jane.doe@example.com
3	Bob Smith	bob.smith@example.com
4	Alice Johnson	alice.johnson@example.com
5	Mike Brown	mike.brown@example.com
NULL	NULL	NULL

e) Distinct query

```
706 •  SELECT DISTINCT Caption FROM Posts;
```

Result Grid | Filter Rows: | Expo

Caption
Beautiful sunset!
Delicious cake!
Amazing mountains!
Cute puppy!
Exciting city!

2) Using where clause

a) With Comparision Operator

```
SELECT *
FROM UserAccounts
WHERE Fullname='Jane Doe';
```

Result Grid | Filter Rows: | Edit:

User_ID	FullName	Email	Password
2	Jane Doe	jane.doe@example.com	password456
NULL	NULL	NULL	NULL

b) Greater Than (>)

```
SELECT *
FROM UserAccounts
WHERE Age>30;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

User_ID	FullName	Email	Password	Age	City
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan
NULL	NULL	NULL	NULL	NULL	NULL

c) Greater Than or Equal (≥)

```
SELECT *
FROM UserAccounts
WHERE Age>=30;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

User_ID	FullName	Email	Password	Age	City
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan
2	Jane Doe	jane.doe@example.co	jane.doe@example.com		USA
3	Bob Smith	bob.smith@example.com	password789	30	USA
NULL	NULL	NULL	NULL	NULL	NULL

3) Using Logical Operator

a) Find all users from the USA who are older than or equal to 30. (Using AND operator)

```
SELECT * FROM UserAccounts WHERE Country='USA' AND Age>=30;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

User_ID	FullName	Email	Password	Age	Country
2	Jane Doe	jane.doe@example.com	password456	30	USA
3	Bob Smith	bob.smith@example.com	password789	30	USA
NULL	NULL	NULL	NULL	NULL	NULL

b) Find all users from the USA or Canada who are older than 25. (Using AND/ OR operator)

```
SELECT * FROM UserAccounts
WHERE (Country='USA' OR Country='South Africa') AND Age>25;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

User_ID	FullName	Email	Password	Age	Country
2	Jane Doe	jane.doe@example.com	password456	30	USA
3	Bob Smith	bob.smith@example.com	password789	30	USA
NULL	NULL	NULL	NULL	NULL	NULL

c) Find all users whose age is between 25 and 35. (Using between clause)

```
SELECT *
FROM UserAccounts
WHERE Age BETWEEN 25 AND 35;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

User_ID	FullName	Email	Password	Age	Country
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan
2	Jane Doe	jane.doe@example.com	password456	30	USA
3	Bob Smith	bob.smith@example.com	password789	30	USA
4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa
NULL	NULL	NULL	NULL	NULL	NULL

d) Find all users from USA, Canada, or Mexico.(Using IN clause)

```
SELECT *
FROM UserAccounts
WHERE Country IN ('USA', 'South Africa');
```

Result Grid | Filter Rows: | Edit: | Export/Import:

User_ID	FullName	Email	Password	Age	Country
2	Jane Doe	jane.doe@example.com	password456	30	USA
3	Bob Smith	bob.smith@example.com	password789	30	USA
4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa
NULL	NULL	NULL	NULL	NULL	NULL

4) Aggregate function:

a) Find the total number of users.(Using Count)

```
SELECT COUNT(*)
FROM UserAccounts;
```

Result Grid | Filter Rows:

COUNT(*)
5

b) Calculate the average age of all users.(Using Average)

```
SELECT AVG(Age)
FROM UserAccounts;
```

Result Grid | Filter Rows:

AVG(Age)
29.2500

c) Find the maximum age of all users. (Using MAX)

```
SELECT MAX(Age)
FROM UserAccounts;
```

Result Grid | Filter Rows:

MAX(Age)
32

d) Find the minimum age of all users.(Using MIN)

```
SELECT MIN(Age)
FROM UserAccounts;
```

Result Grid | Filter Rows:

MIN(Age)
25

e) Calculate the total age of all users.(Using Sum)

```
SELECT SUM(Age)
FROM UserAccounts;
```

Result Grid | Filter Rows:

SUM(Age)
117

5) Group By clause:

a) Find the total number of posts for each user.

```
SELECT User_ID, COUNT(Post_ID) as Total_Posts
FROM Posts
GROUP BY User_ID;
```

Result Grid | Filter Rows: | Export: | Wrap

User_ID	Total_Posts
1	1
2	1
3	1
4	1
5	1

b) Find the total number of likes for each post.

```
SELECT Post_ID, COUNT(Like_ID) as Total_Likes
FROM Likes
GROUP BY Post_ID;
```

Result Grid | Filter Rows: | Export: | Wrap

Post_ID	Total_Likes
1	3
2	2
3	1
4	1
5	3

6) Like operator:

a) Find all comments starting with "Y".

```
SELECT *
FROM Comments
WHERE Comment_Text LIKE 'Y%';
```

Result Grid | Filter Rows: | Edit:

Comment_ID	Post_ID	User_ID	Comment_Text
3	2	1	Yum!
NULL	NULL	NULL	NULL

b) Find all users with usernames starting with "A”.

```
SELECT *
FROM useraccounts
WHERE FullName LIKE 'J%';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

User_ID	FullName	Email	Password	Age	Country
2	Jane Doe	jane.doe@example.com	password456	30	USA
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan
NULL	NULL	NULL	NULL	NULL	NULL

7) Union:

a) Find all posts that have likes or comments.

```
SELECT Post_ID FROM Likes
UNION
SELECT Post_ID FROM Comments;
```

Result Grid | Filter Rows:

Post_ID
1
2
3
4
5

b) Find all post IDs that have likes from users in the USA or South Africa.

```
SELECT Post_ID FROM Comments;SELECT Post_ID FROM Likes WHERE User_ID IN
(SELECT User_ID FROM UserAccounts WHERE Country='USA')
UNION SELECT Post_ID FROM Likes WHERE User_ID IN (SELECT User_ID
FROM UserAccounts WHERE Country='South Africa');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Post_ID
1
5
3

c) Find all user IDs who have posted or commented.

```
SELECT User_ID FROM Posts
UNION
SELECT User_ID FROM Comments;
```

Result Grid | Filter Rows:

User_ID
1
2
3
4
5

8) Joins:

1) INNER JOIN

a) Find all posts with their corresponding user information.

```
SELECT *
FROM Posts
INNER JOIN UserAccounts
ON Posts.User_ID = UserAccounts.User_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Post_ID	User_ID	Image_URL	Caption	User_ID	FullName	Email	Password	Age	Country
1	1	image1.jpg	Beautiful sunset!	1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan
2	2	image2.jpg	Delicious cake!	2	Jane Doe	jane.doe@example.com	password456	30	USA
3	3	image3.jpg	Amazing mountains!	3	Bob Smith	bob.smith@example.com	password789	30	USA
4	4	image4.jpg	Cute puppy!	4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa
5	5	image5.jpg	Exciting city!	5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL

2) LEFT JOIN (or LEFT OUTER JOIN)

b) Find all users with their follower information.

```
SELECT *
FROM UserAccounts
LEFT JOIN Followers
ON UserAccounts.User_ID = Followers.User_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

User_ID	FullName	Email	Password	Age	Country	Follower_ID	User_ID	Following_ID
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	1	1	2
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	6	1	4
2	Jane Doe	jane.doe@example.com	password456	30	USA	2	2	1
2	Jane Doe	jane.doe@example.com	password456	30	USA	7	2	5
3	Bob Smith	bob.smith@example.com	password789	30	USA	3	3	1
3	Bob Smith	bob.smith@example.com	password789	30	USA	8	3	1
4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa	4	4	2
4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa	9	4	3
5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL	5	5	3
5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL	10	5	2

3) RIGHT JOIN (or RIGHT OUTER JOIN)

c) Find all followers with their corresponding user information.

```
SELECT *
FROM Followers
RIGHT JOIN UserAccounts
ON Followers.User_ID = UserAccounts.User_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Follower_ID	User_ID	Following_ID	User_ID	FullName	Email	Password	Age	Country
1	1	2	1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan
6	1	4	1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan
2	2	1	2	Jane Doe	jane.doe@example.com	password456	30	USA
7	2	5	2	Jane Doe	jane.doe@example.com	password456	30	USA
3	3	1	3	Bob Smith	bob.smith@example.com	password789	30	USA
8	3	1	3	Bob Smith	bob.smith@example.com	password789	30	USA
4	4	2	4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa
9	4	3	4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa
5	5	3	5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL
10	5	2	5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL

4) CROSS JOIN

d) Find all possible combinations of users and posts.

```
SELECT *
FROM UserAccounts
CROSS JOIN Posts;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

User_ID	FullName	Email	Password	Age	Country	Post_ID	User_ID	Image_URL	Caption
5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL	1	1	image1.jpg	Beautiful sunset!
4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa	1	1	image1.jpg	Beautiful sunset!
3	Bob Smith	bob.smith@example.com	password789	30	USA	1	1	image1.jpg	Beautiful sunset!
2	Jane Doe	jane.doe@example.com	password456	30	USA	1	1	image1.jpg	Beautiful sunset!
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	1	1	image1.jpg	Beautiful sunset!
5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL	2	2	image2.jpg	Delicious cake!
4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa	2	2	image2.jpg	Delicious cake!
3	Bob Smith	bob.smith@example.com	password789	30	USA	2	2	image2.jpg	Delicious cake!
2	Jane Doe	jane.doe@example.com	password456	30	USA	2	2	image2.jpg	Delicious cake!
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	2	2	image2.jpg	Delicious cake!
5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL	3	3	image3.jpg	Amazing mount...
4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa	3	3	image3.jpg	Amazing mount...
3	Bob Smith	bob.smith@example.com	password789	30	USA	3	3	image3.jpg	Amazing mount...
2	Jane Doe	jane.doe@example.com	password456	30	USA	3	3	image3.jpg	Amazing mount...

5) SELF JOIN

e) Find all users who follow themselves.

```
SELECT *
FROM UserAccounts
JOIN Followers
ON UserAccounts.User_ID = Followers.Following_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

User_ID	FullName	Email	Password	Age	Country	Follower_ID	User_ID	Following_ID
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	2	2	1
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	3	3	1
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	8	3	1
2	Jane Doe	jane.doe@example.com	password456	30	USA	1	1	2
2	Jane Doe	jane.doe@example.com	password456	30	USA	4	4	2
2	Jane Doe	jane.doe@example.com	password456	30	USA	10	5	2
3	Bob Smith	bob.smith@example.com	password789	30	USA	5	5	3
3	Bob Smith	bob.smith@example.com	password789	30	USA	9	4	3
4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa	6	1	4
5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL	7	2	5

6) MULTIPLE JOIN

f) Find all posts with their corresponding user, comment, and like information.

```
SELECT * FROM Posts
INNER JOIN UserAccounts ON Posts.User_ID = UserAccounts.User_ID
INNER JOIN Comments ON Posts.Post_ID = Comments.Post_ID
INNER JOIN Likes ON Posts.Post_ID = Likes.Post_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Post_ID	User_ID	Image_URL	Caption	User_ID	FullName	Email	Password	Age	Country	Comment_ID	Post_ID	User_ID	Comment_Text	Like_ID	Post_ID	User_ID
1	1	image1.jpg	Beautiful sunset!	1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	1	1	2	Love this photo!	1	1	2
1	1	image1.jpg	Beautiful sunset!	1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	1	1	2	Love this photo!	2	1	3
1	1	image1.jpg	Beautiful sunset!	1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	1	1	2	Love this photo!	9	1	4
1	1	image1.jpg	Beautiful sunset!	1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	2	1	3	Stunning view!	1	1	2
1	1	image1.jpg	Beautiful sunset!	1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	2	1	3	Stunning view!	2	1	3
1	1	image1.jpg	Beautiful sunset!	1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan	2	1	3	Stunning view!	9	1	4
2	2	image2.jpg	Delicious cake!	2	Jane Doe	jane.doe@example.com	password456	30	USA	3	2	1	Yum!	3	2	1
2	2	image2.jpg	Delicious cake!	2	Jane Doe	jane.doe@example.com	password456	30	USA	3	2	1	Yum!	10	2	5
3	3	image3.jpg	Amazing mountains!	3	Bob Smith	bob.smith@example.com	password789	30	USA	4	3	4	Breathtaking!	4	3	4
4	4	image4.jpg	Cute puppy!	4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa	5	4	5	Adorable!	5	4	5
5	5	image5.jpg	Exciting city!	5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL	6	5	2	City life!	6	5	2
5	5	image5.jpg	Exciting city!	5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL	6	5	2	City life!	7	5	3
5	5	image5.jpg	Exciting city!	5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL	6	5	2	City life!	8	5	1
5	5	image5.jpg	Exciting city!	5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL	7	5	3	Skyscrapers!	6	5	2
5	5	image5.jpg	Exciting city!	5	Mike Brown	mike.brown@example.com	password1213	NULL	NULL	7	5	3	Skyscrapers!	7	5	3

9) Subquery

1) Simple Subqueries:

a) Find all posts written by users who have more than or equal to 3 followers.

```
SELECT * FROM Posts WHERE User_ID IN (SELECT Following_ID FROM Followers
GROUP BY Following_ID HAVING count(*) >= 3);
select * from followers;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell C

Post_ID	User_ID	Image_URL	Caption
1	1	image1.jpg	Beautiful sunset!
2	2	image2.jpg	Delicious cake!
NULL	NULL	NULL	NULL

2) Subqueries with IN

b) Find all users who have liked a post written by user 1.

```
SELECT * FROM UserAccounts WHERE User_ID IN (
SELECT User_ID FROM Likes WHERE Post_ID IN (
SELECT Post_ID FROM Posts WHERE User_ID = 1));
```

Result Grid | Filter Rows: | Edit: | Export/Import:

User_ID	FullName	Email	Password	Age	Country
2	Jane Doe	jane.doe@example.com	password456	30	USA
3	Bob Smith	bob.smith@example.com	password789	30	USA
4	Alice Johnson	alice.johnson@example.com	password1011	25	South Africa
NULL	NULL	NULL	NULL	NULL	NULL

3) Subqueries with NOT IN

c) 1. Find all users who have not liked any posts.

```
SELECT * FROM UserAccounts
WHERE User_ID NOT IN (SELECT User_ID FROM Likes);
```

Result Grid | Filter Rows: | Edit: | Export/Ir

User_ID	FullName	Email	Password	Age	Country
NULL	NULL	NULL	NULL	NULL	NULL

4) Subqueries with ANY

d) Find all posts that have been liked by at least one user who has more than 10 followers.

```
SELECT * FROM Posts WHERE Post_ID = ANY (
SELECT Post_ID FROM Likes WHERE User_ID IN ( SELECT Following_ID
FROM Followers  GROUP BY Following_ID HAVING COUNT(*) <= 5));
```

Result Grid | Filter Rows: | Edit: | Export/Import:

Post_ID	User_ID	Image_URL	Caption
1	1	image1.jpg	Beautiful sunset!
2	2	image2.jpg	Delicious cake!
3	3	image3.jpg	Amazing mountains!
4	4	image4.jpg	Cute puppy!
5	5	image5.jpg	Exciting city!
NULL	NULL	NULL	NULL

5) Subqueries with ALL

e) Find all users who have more followers than the average.

```
SELECT * FROM UserAccounts WHERE User_ID IN (SELECT Following_ID
FROM Followers GROUP BY Following_ID HAVING COUNT(*) > (SELECT AVG(Count)FROM
(SELECT Following_ID, COUNT(*) as Count FROM Followers GROUP BY Following_ID) as AvgFollowers));
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

User_ID	FullName	Email	Password	Age	Country
1	John Doe Jr.	john.doe@example.com	password123	32	Manhattan
2	Jane Doe	jane.doe@example.com	password456	30	USA
NULL	NULL	NULL	NULL	NULL	NULL