**Database project Phase 1, 2, 3 and 4 Combined Submission**

**Project Idea: portfolio terrain – A dedicated social media platform for freelance and remote developers**



**Submitted by:**

**Usman Shahid (L1F22BSCS1057) &**

**Ali Hamza Sehole (L1F21BSCS0791)**

**Section: D-8**

**Submitted to:**

**Sir Afham Nazir**

**Database project idea:**

social media platform dedicated for freelance and remote workers – Portfolio Terrain

**project description:**

this project will be a dedicated social media platform for freelance and remote developers that will allow developers to showcase there projects, skills and endorsements to secure freelance work and clients all over the globe. The idea is to provide users with interactive UI that is easy to use for both buyer and seller. The unique thing in this project would be that it will allow buyers to independently select desired freelancers for their projects with right skill set and portfolio projects. There will not be any bidding system although there will be a feature for buyers that they can post their project or job with their requirements and freelancers can apply to those projects. There will be a commission fee of 1% on each project that freelancer would pay to platform in order to use the services of this platform. Freelancers and buyers both will be given the ability to follow accounts of other users in order to increase the number of users of platform. Real time message communication feature shall also be provided to both freelancers and buyers in order to contact with their relevant people. The platform itself will play the role of a middle man and will connect buyer with seller and ensure that the buyer pays the payment to the platform and after successful confirmation of project completion from client and freelancer, the platform will pay the amount to its related freelance user and deduct it’s commission from the total amount

**Overview of Entities required in this project**

1. **User\_Account:**

Stores login credentials for users (freelancers or remote workers or buyers/clients).

1. **Porfolio Projects:**

Users will post these projects to attract buyers.

1. **Likes:**

Clients can like a portfolio project.

1. **Reviews:**

After completion of a certain project, a client can review the work of a freelance/remote worker.

1. **Payments:**

Buyers would pay Users on the completion of tasks.

1. **Projects\_done:**

Users who have completed projects on the site would have the facility to showcase their previous work on their profile.

1. **User\_Profile:**

Stores information about the skills, profile picture, and about info of the user.

1. **Buyer\_Profile:**

Stores information about the payment details, profile picture, and about info of the buyer.

1. **Categories:**

Provides buyers with the list of categories of freelance and remote work.

1. **Messages:**

Users and buyers will have the facility to chat with each other regarding projects.

1. **Notifications:**

For providing users with notifications in regard of their work

1. **Contracts:**

For storing contract related information if any, in between freelancer and buyer

1. **Attachments:**

For storing the filenames that users upload either those files can be of freelancers or buyers

1. **Invoices:**

To keep track of payments and provide user and freelancer with proof of payment transfer

1. **Admin\_Account:**

For administrative functionalities

1. **Management\_Account:**

For management purposes, solely for approving posts of users, either freelancer or buyer

**Attributes and Relations of entities with each other**

**User\_Account:**

1. Attributes: user\_id (Primary Key), username, email, password, role (freelancer/buyer), last\_login
2. Relationships:
3. One-to-One with User\_Profile (user\_id as Foreign Key)
4. One-to-Many with Messages (sender\_user\_id and receiver\_user\_id as Foreign Keys)

**Porfolio Projects:**

1. Attributes: project\_id (Primary Key), user\_id (Foreign Key), title, description, creation\_date, visibility
2. Relationships:
3. Many-to-One with User\_Account (user\_id as Foreign Key)
4. One-to-Many with Likes (project\_id as Foreign Key)
5. One-to-Many with Reviews (project\_id as Foreign Key)

**Likes:**

1. Attributes: like\_id (Primary Key), project\_id (Foreign Key), user\_id (Foreign Key), like\_date
2. Relationships:
3. Many-to-One with Porfolio Projects (project\_id as Foreign Key)
4. Many-to-One with User\_Account (user\_id as Foreign Key)

**Reviews:**

1. Attributes: review\_id (Primary Key), project\_id (Foreign Key), reviewer\_user\_id (Foreign Key), review\_text, rating, review\_date
2. Relationships:
3. Many-to-One with Porfolio Projects (project\_id as Foreign Key)
4. Many-to-One with User\_Account (reviewer\_user\_id as Foreign Key)

**Payments:**

1. Attributes: payment\_id (Primary Key), project\_id (Foreign Key), buyer\_user\_id (Foreign Key), amount, payment\_date
2. Relationships:
3. Many-to-One with Porfolio Projects (project\_id as Foreign Key)
4. Many-to-One with User\_Account (buyer\_user\_id as Foreign Key)
5. Projects\_done:
6. Attributes: project\_done\_id (Primary Key), user\_id (Foreign Key), project\_id (Foreign Key)
7. Relationships:
8. Many-to-One with User\_Account (user\_id as Foreign Key)
9. Many-to-One with Porfolio Projects (project\_id as Foreign Key)

**User\_Profile:**

1. Attributes: user\_id (Primary Key), full\_name, bio, profile\_picture\_url, skills
2. Relationships:
3. One-to-One with User\_Account (user\_id as Foreign Key)

**Buyer\_Profile:**

1. Attributes: user\_id (Primary Key), payment\_details, company\_name, company\_logo\_url
2. Relationships:
3. One-to-One with User\_Account (user\_id as Foreign Key)
4. Categories:
5. Attributes: category\_id (Primary Key), category\_name
6. Relationships: None

**Messages:**

1. Attributes: message\_id (Primary Key), sender\_user\_id (Foreign Key), receiver\_user\_id (Foreign Key), message\_text, timestamp, is\_read
2. Relationships:
3. Many-to-One with User\_Account (sender\_user\_id and receiver\_user\_id as Foreign Keys)
4. Notifications:
5. Attributes: notification\_id (Primary Key), user\_id (Foreign Key), notification\_text, notification\_date
6. Relationships:
7. Many-to-One with User\_Account (user\_id as Foreign Key)

**Contracts:**

1. Attributes: contract\_id (Primary Key), project\_id (Foreign Key), freelancer\_user\_id (Foreign Key), buyer\_user\_id (Foreign Key), terms, start\_date, end\_date
2. Relationships:
3. Many-to-One with Porfolio Projects (project\_id as Foreign Key)
4. Many-to-One with User\_Account for freelancer (freelancer\_user\_id as Foreign Key)
5. Many-to-One with User\_Account for buyer (buyer\_user\_id as Foreign Key)
6. Attachments:
7. Attributes: attachment\_id (Primary Key), project\_id (Foreign Key), filename, file\_url
8. Relationships:
9. Many-to-One with Porfolio Projects (project\_id as Foreign Key)

**Invoices:**

1. Attributes: invoice\_id (Primary Key), project\_id (Foreign Key), buyer\_user\_id (Foreign Key), freelancer\_user\_id (Foreign Key), amount, issue\_date, due\_date
2. Relationships:
3. Many-to-One with Porfolio Projects (project\_id as Foreign Key)
4. Many-to-One with User\_Account for buyer (buyer\_user\_id as Foreign Key)
5. Many-to-One with User\_Account for freelancer (freelancer\_user\_id as Foreign Key)

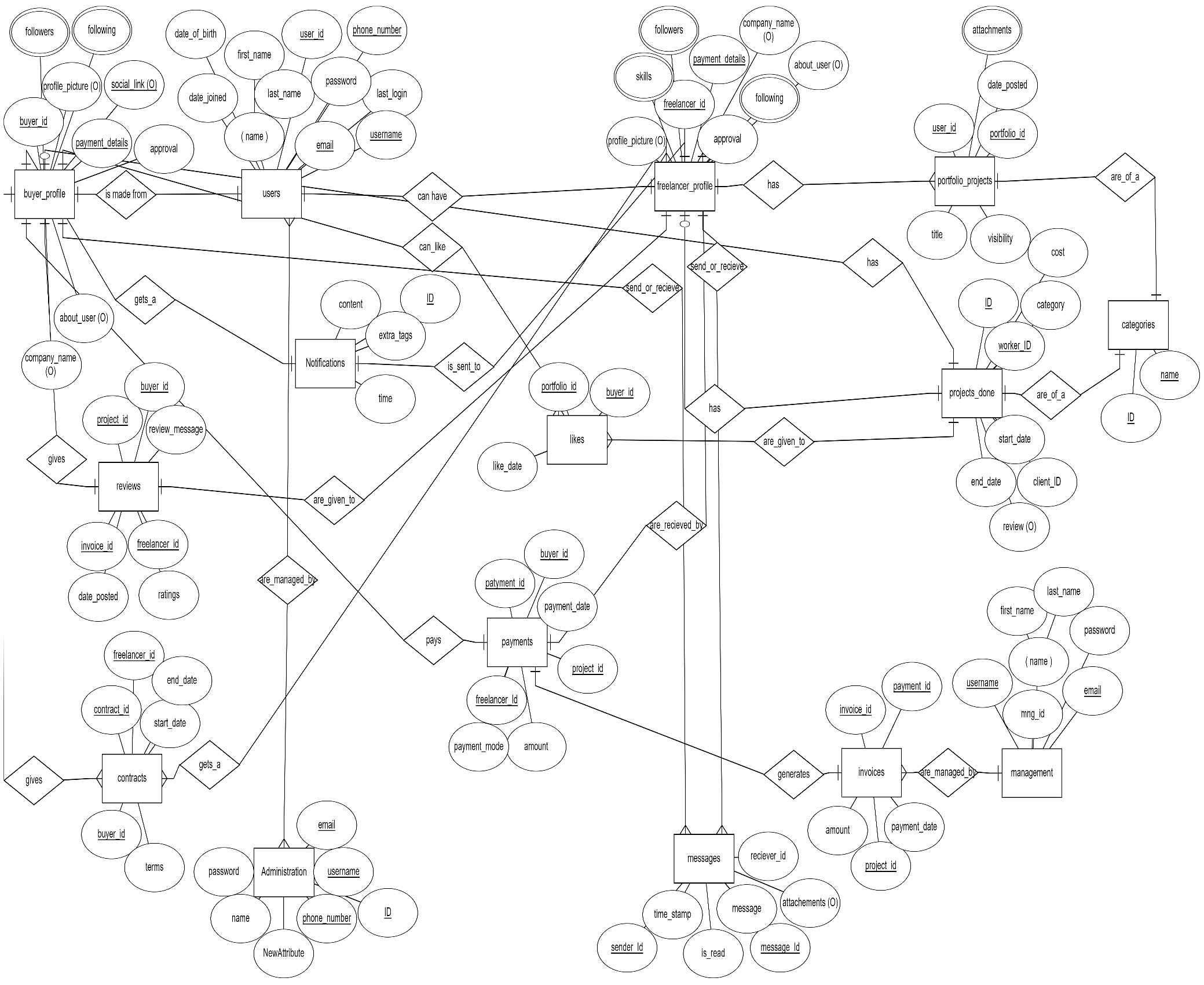
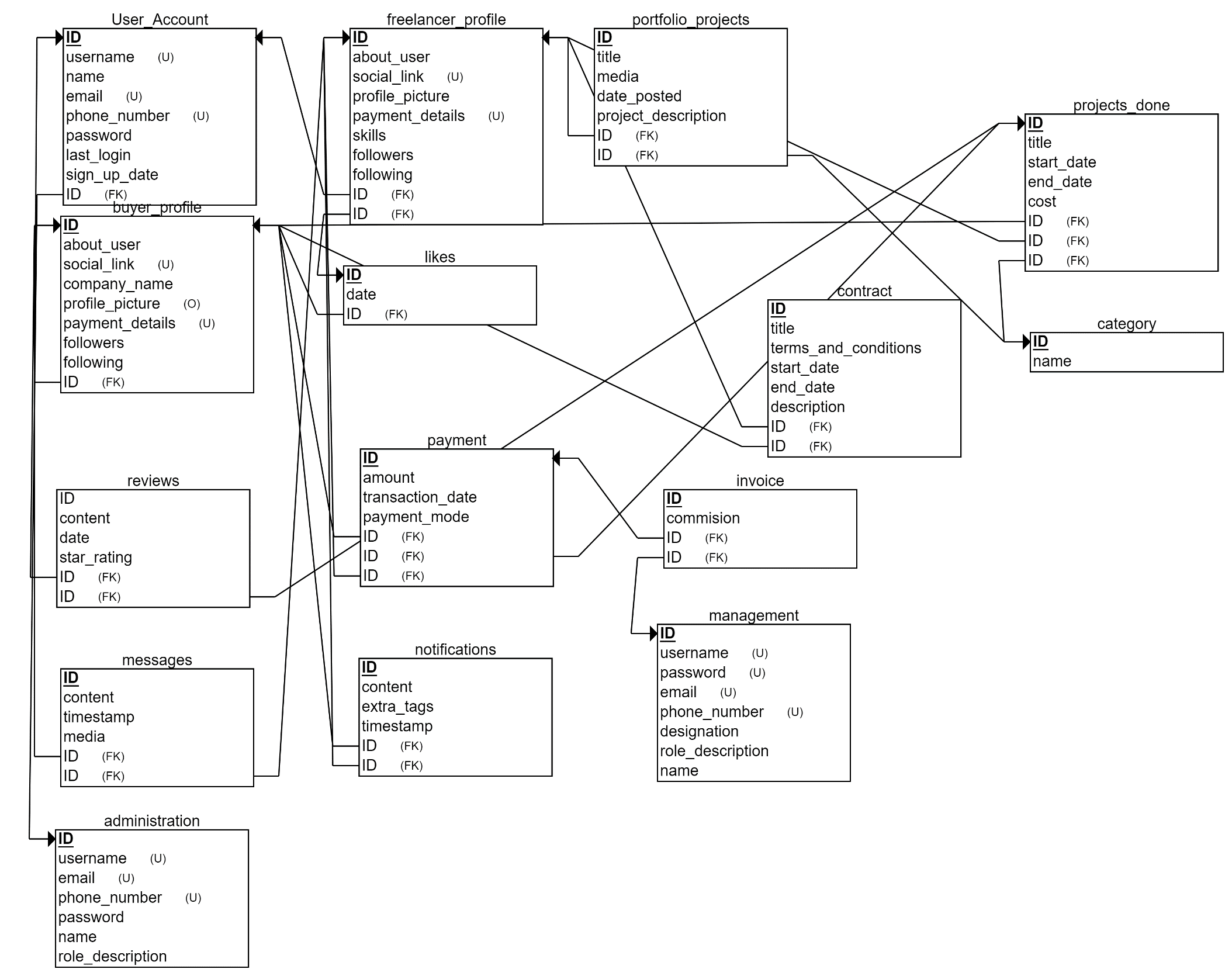
**Admin\_Account:**

1. Attributes: admin\_id (Primary Key), username, email, password
2. Relationships: None

**Management\_Account:**

1. Attributes: management\_id (Primary Key), username, email, password
2. Relationships: None

**ERD Diagram And Relational Schema:**

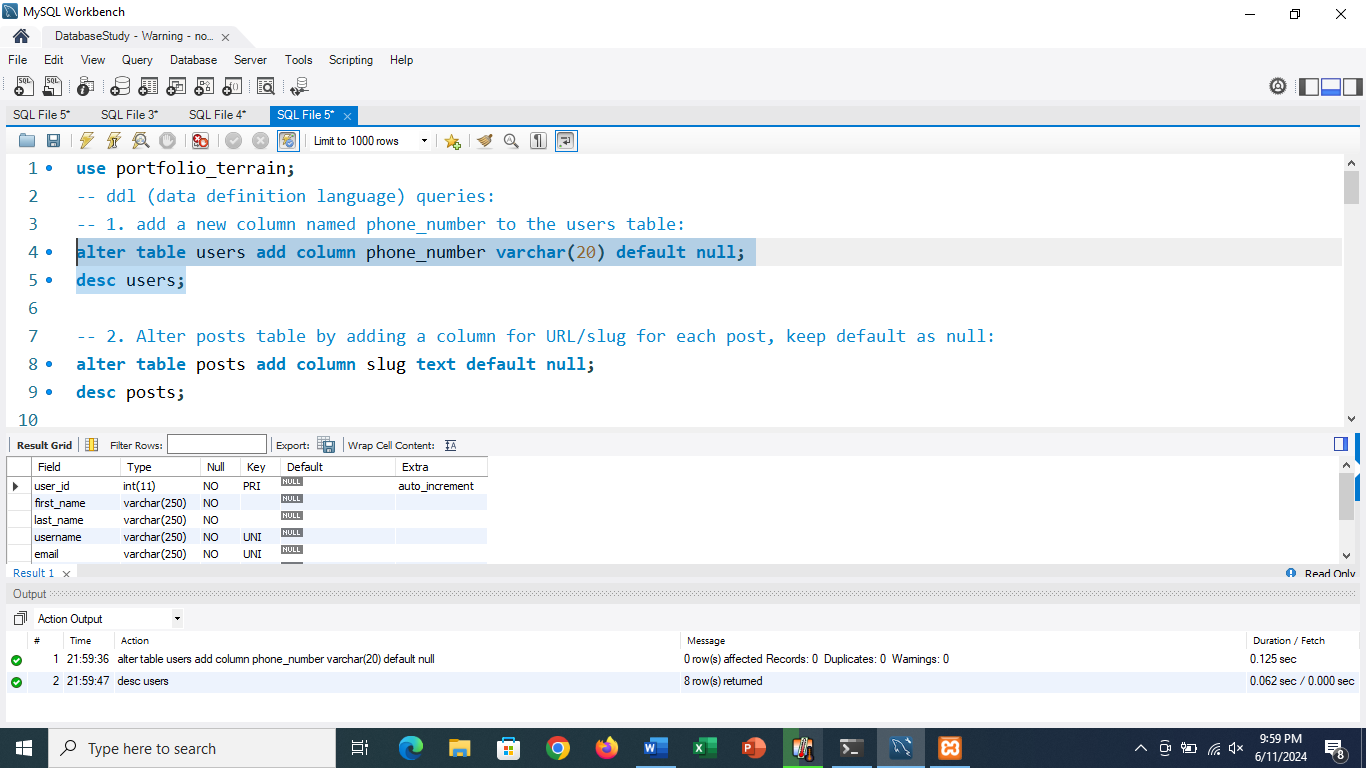
1. 
2. 

use portfolio\_terrain;

**DDL (data definition language) queries:**

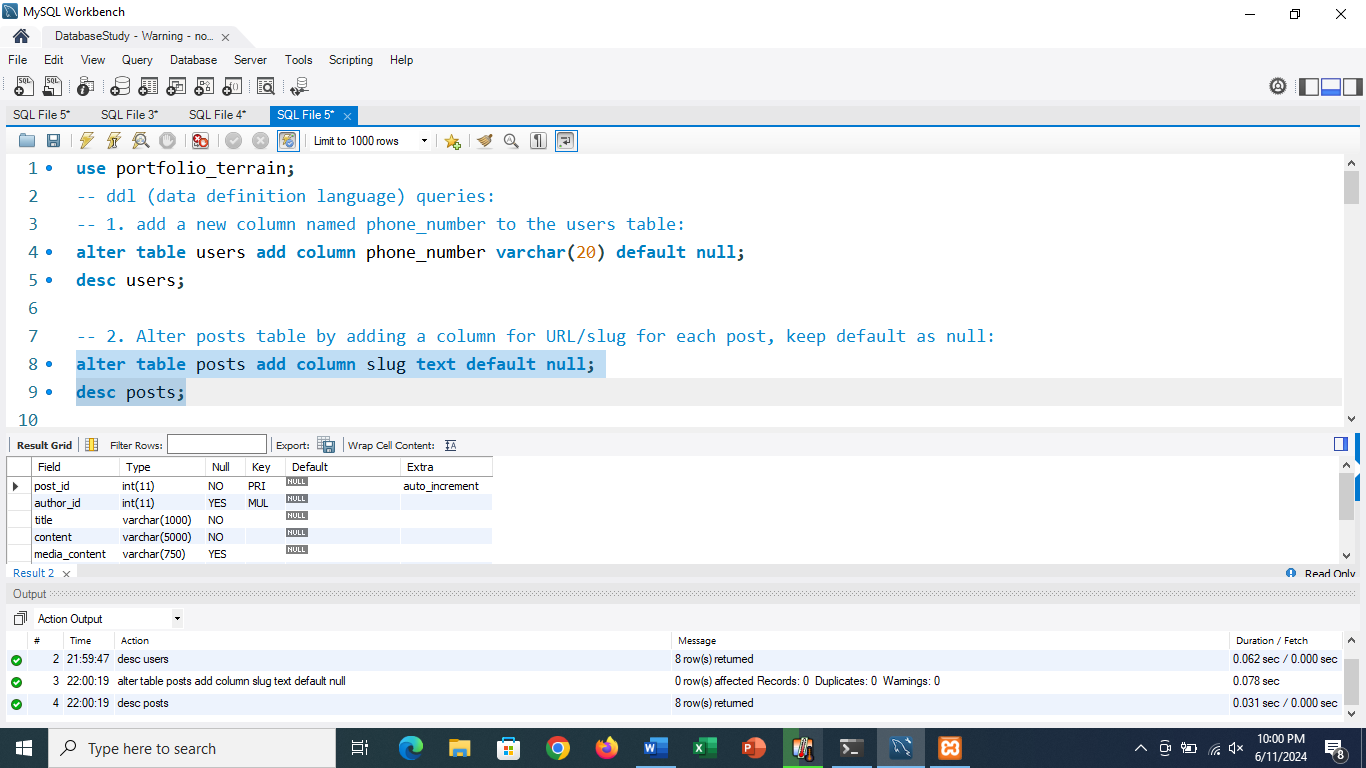
**1. add a new column named phone\_number to the users table:**

alter table users add column phone\_number varchar(20) default null;



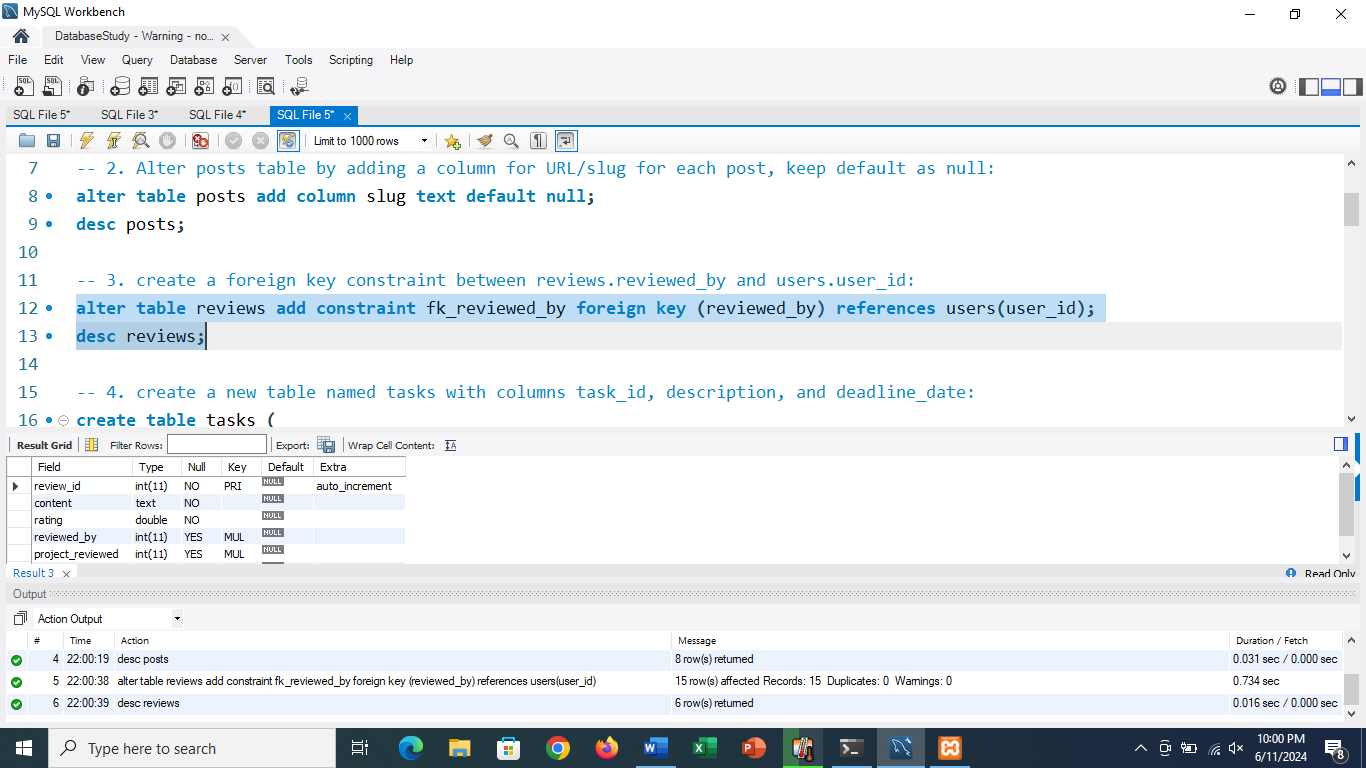
**2. Alter posts table by adding a column for URL/slug for each post, keep default as null:**

alter table posts add column slug text default null;



**3. create a foreign key constraint between reviews.reviewed\_by and users.user\_id:**

alter table reviews add constraint fk\_reviewed\_by foreign key (reviewed\_by) references users(user\_id);



**4. create a new table named tasks with columns task\_id, description, and deadline\_date:**

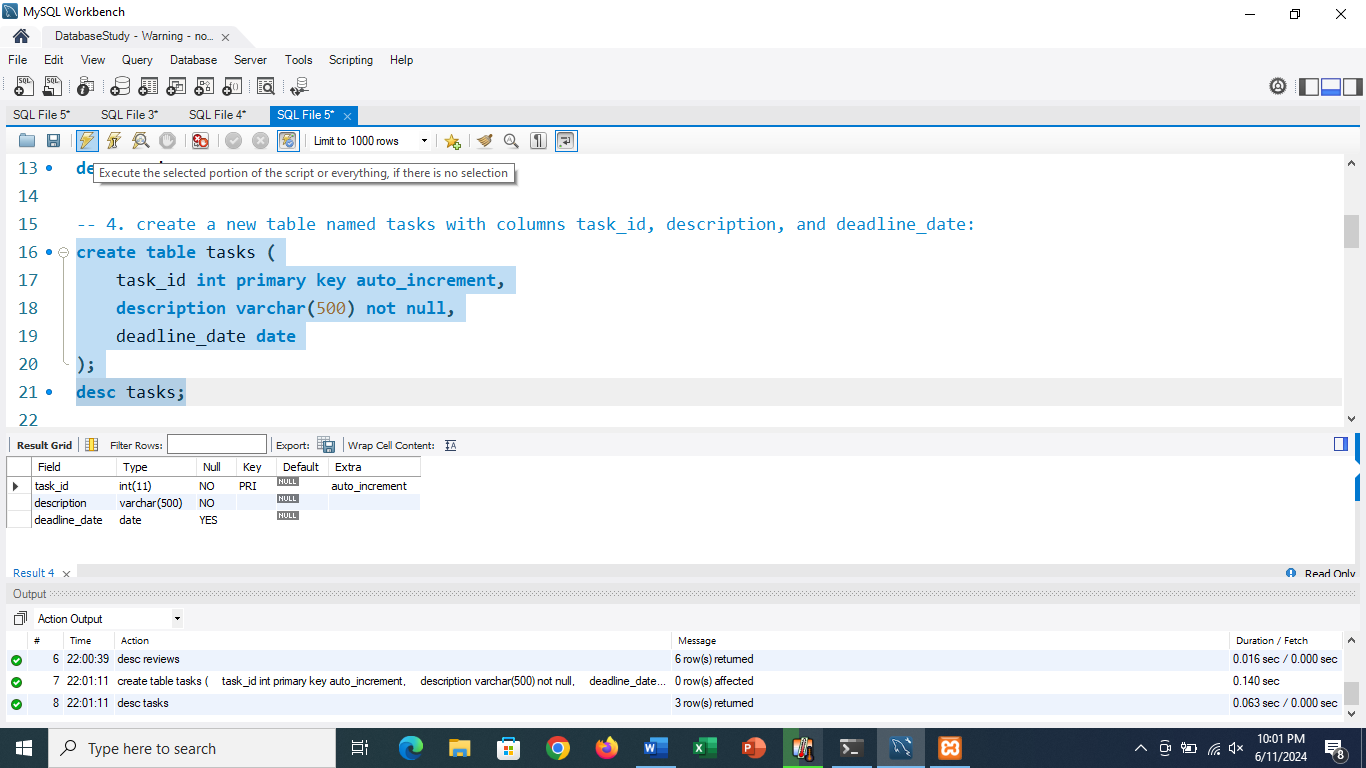
create table tasks (

task\_id int primary key auto\_increment,

description varchar(500) not null,

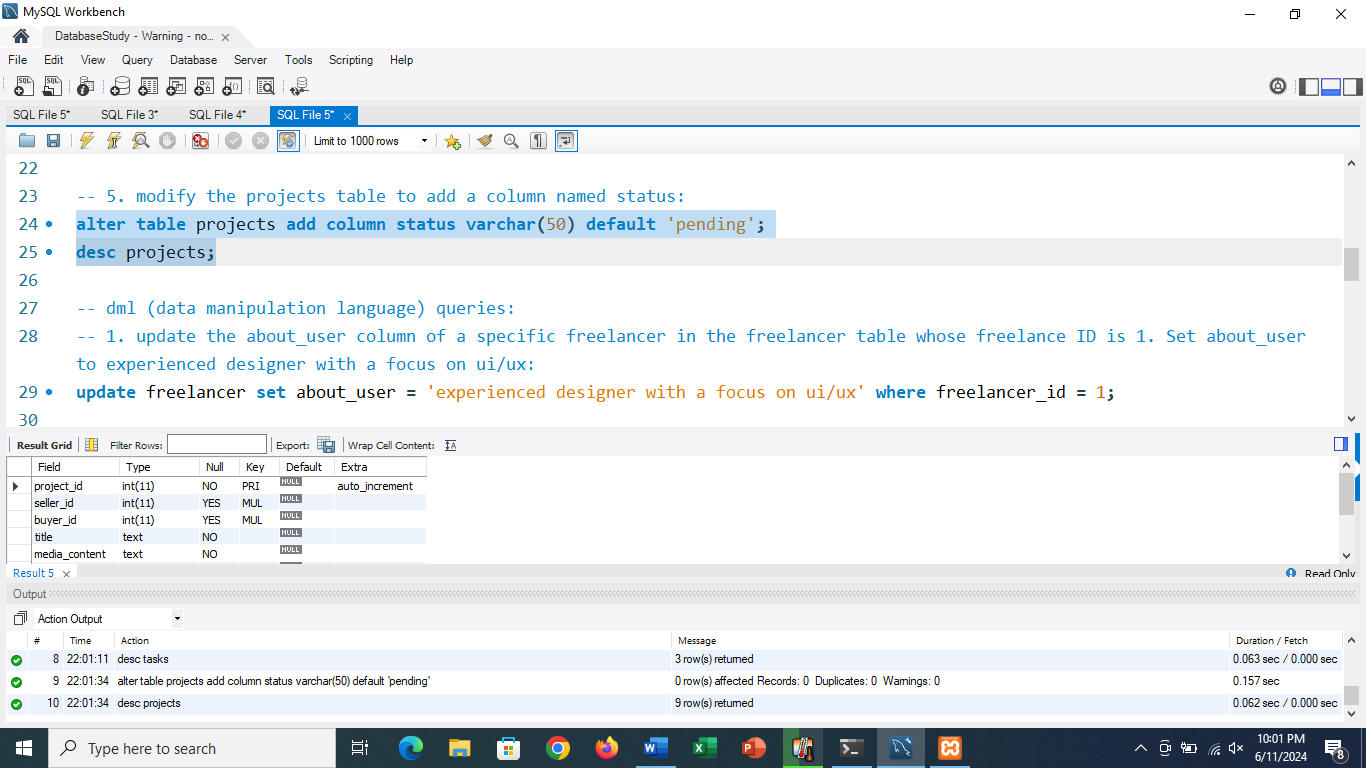
deadline\_date date

);



**5. modify the projects table to add a column named status:**

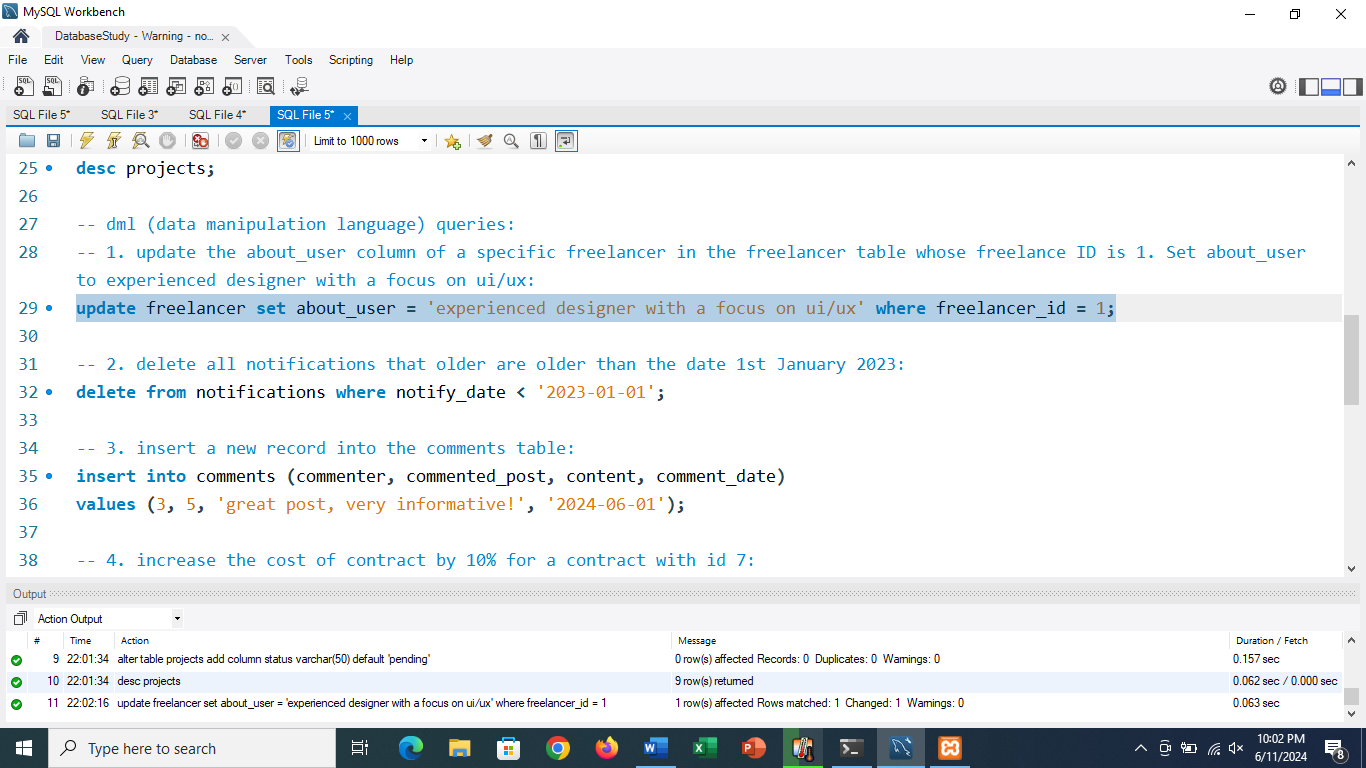
alter table projects add column status varchar(50) default 'pending';



**DML (data manipulation language) queries:**

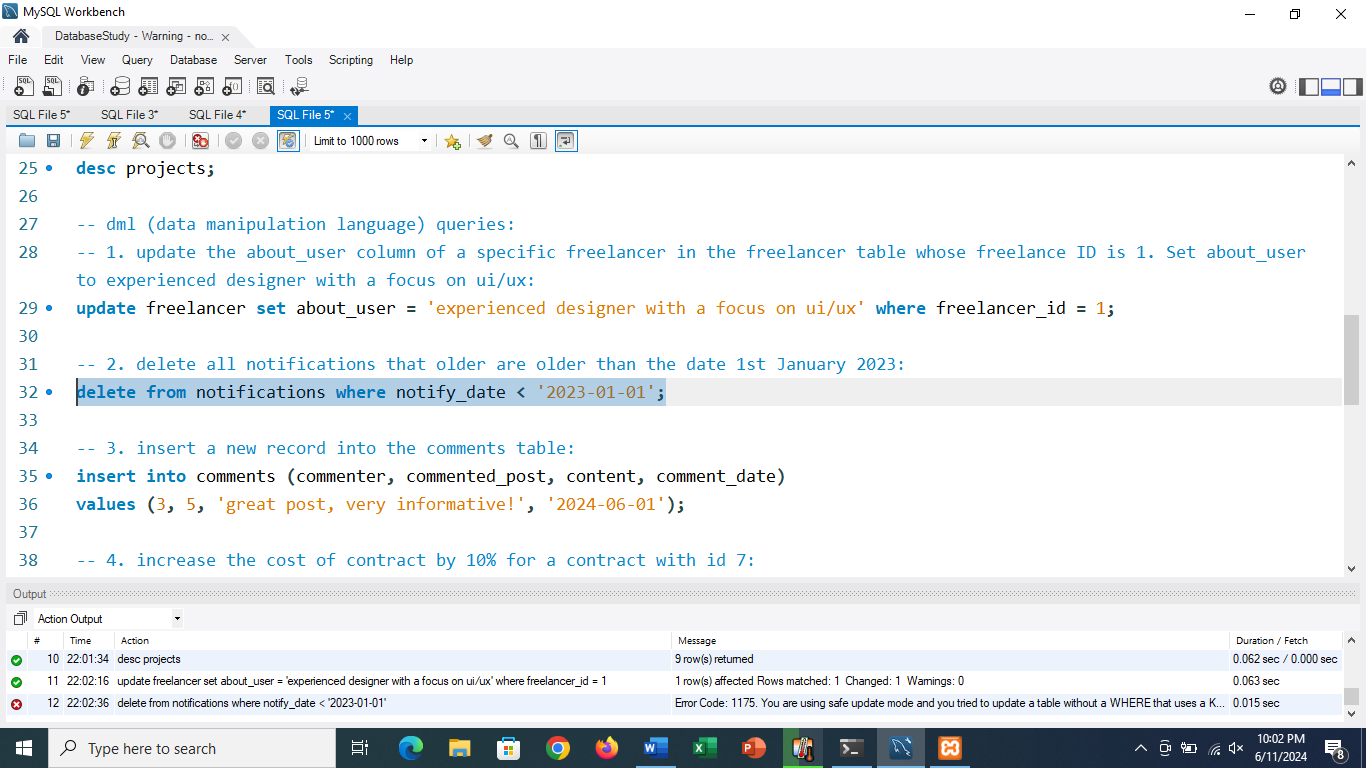
**1. update the about\_user column of a specific freelancer in the freelancer table whose freelance ID is 1. Set about\_user to experienced designer with a focus on ui/ux:**

update freelancer set about\_user = 'experienced designer with a focus on ui/ux' where freelancer\_id = 1;



**2. delete all notifications that older are older than the date 1st January 2023:**

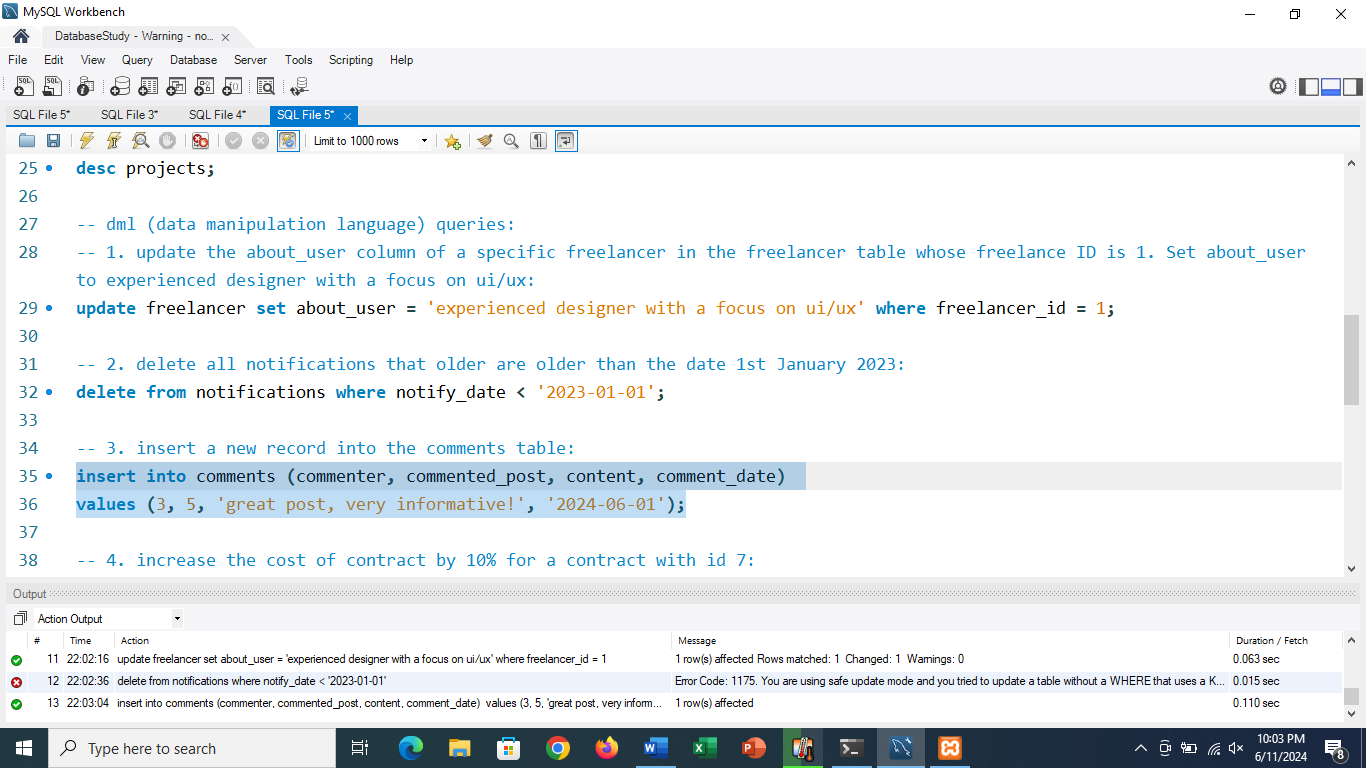
delete from notifications where notify\_date < '2023-01-01';



**3. insert a new record into the comments table:**

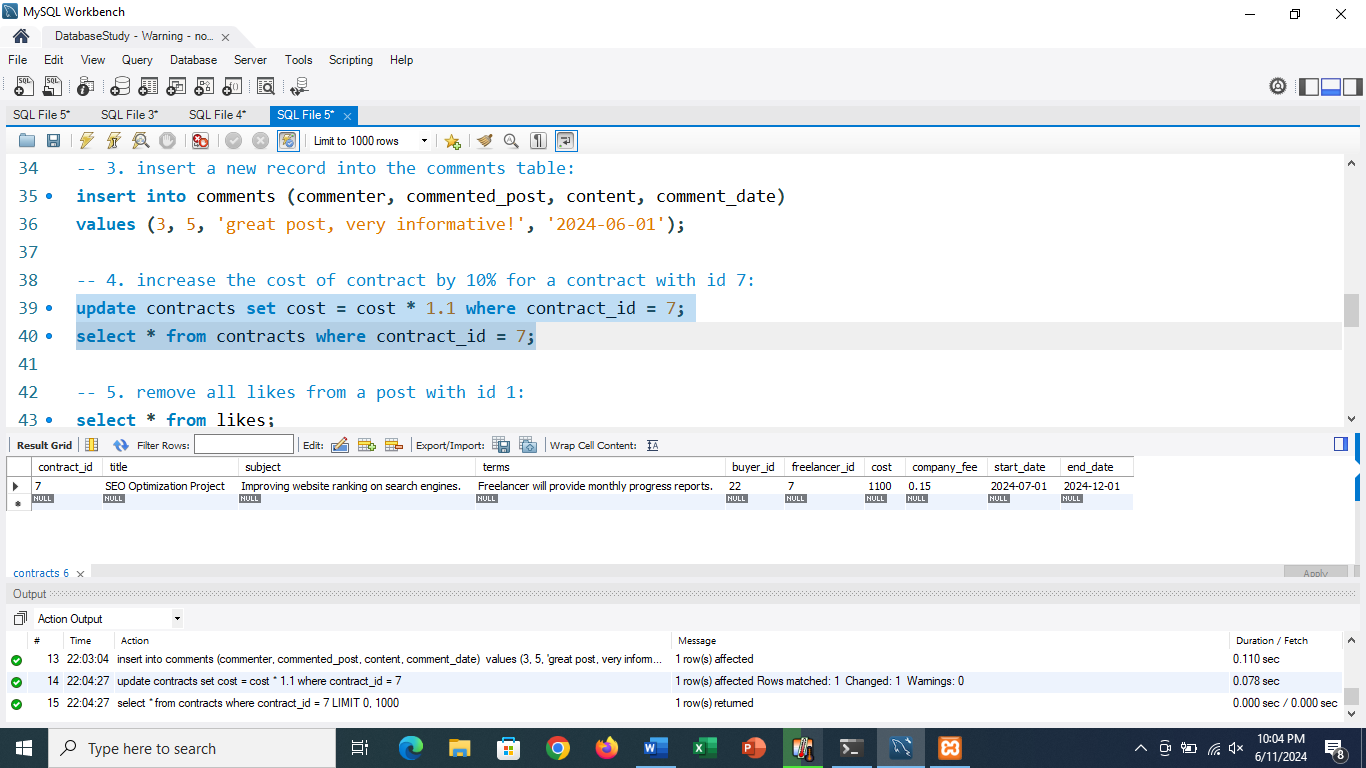
insert into comments (commenter, commented\_post, content, comment\_date)

values (3, 5, 'great post, very informative!', '2024-06-01');



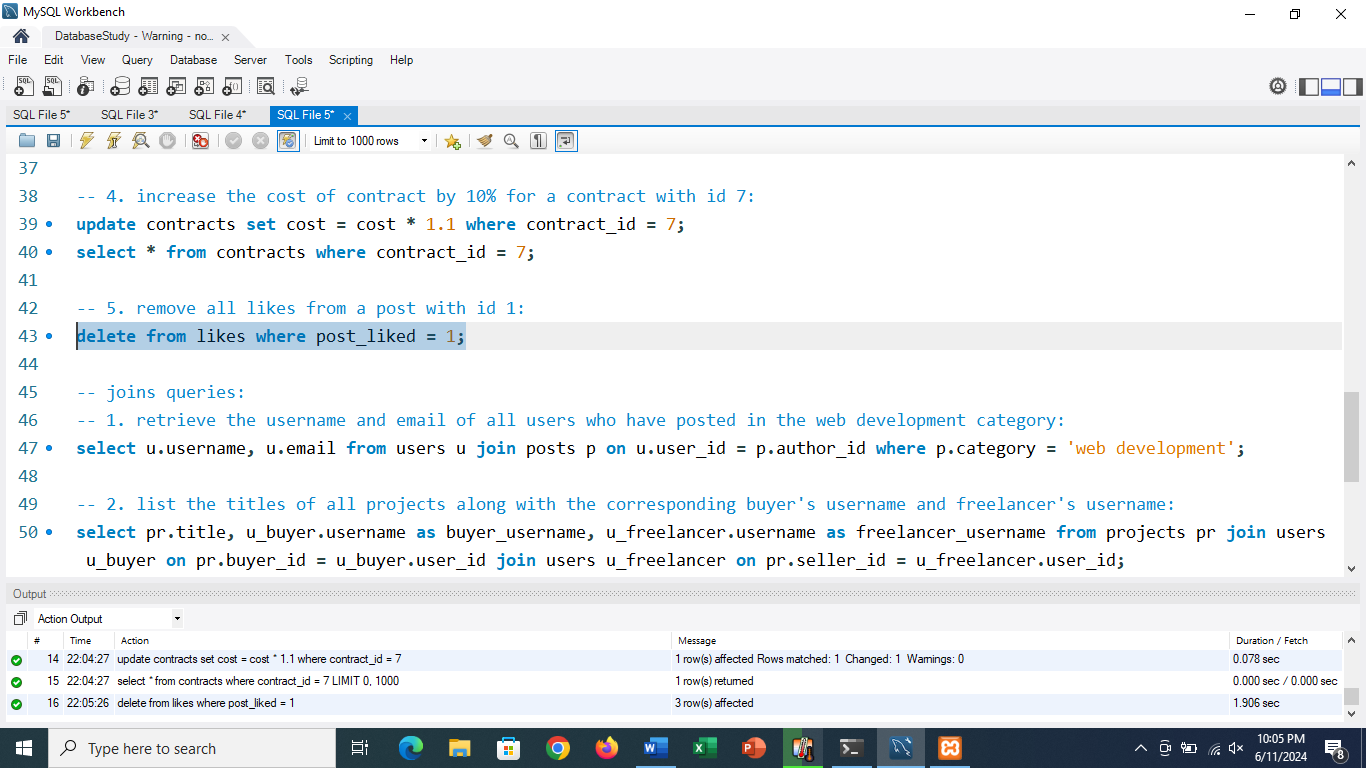
**4. increase the cost of contract by 10% for a contract with id 7:**

update contracts set cost = cost \* 1.1 where contract\_id = 7;



**5. remove all likes from a post with id 1:**

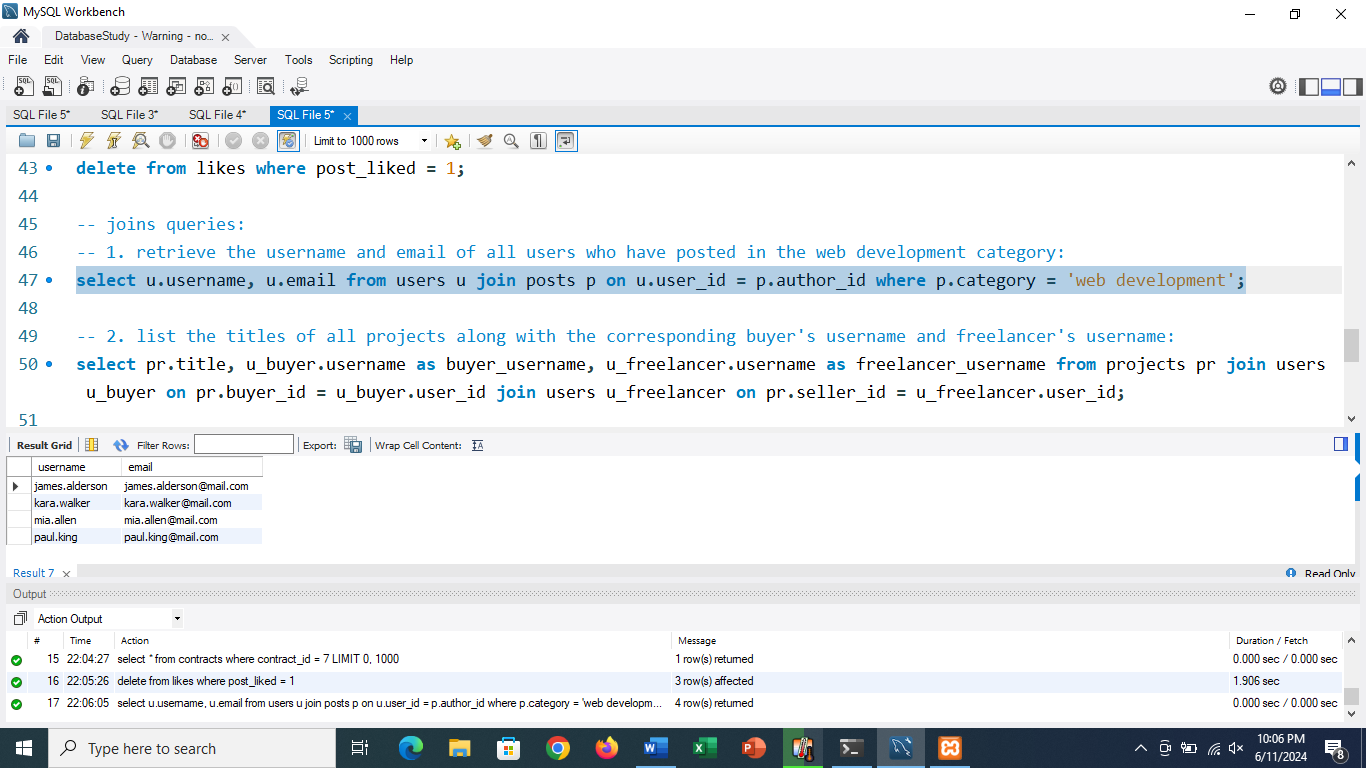
delete from likes where post\_liked = 1;



**joins queries:**

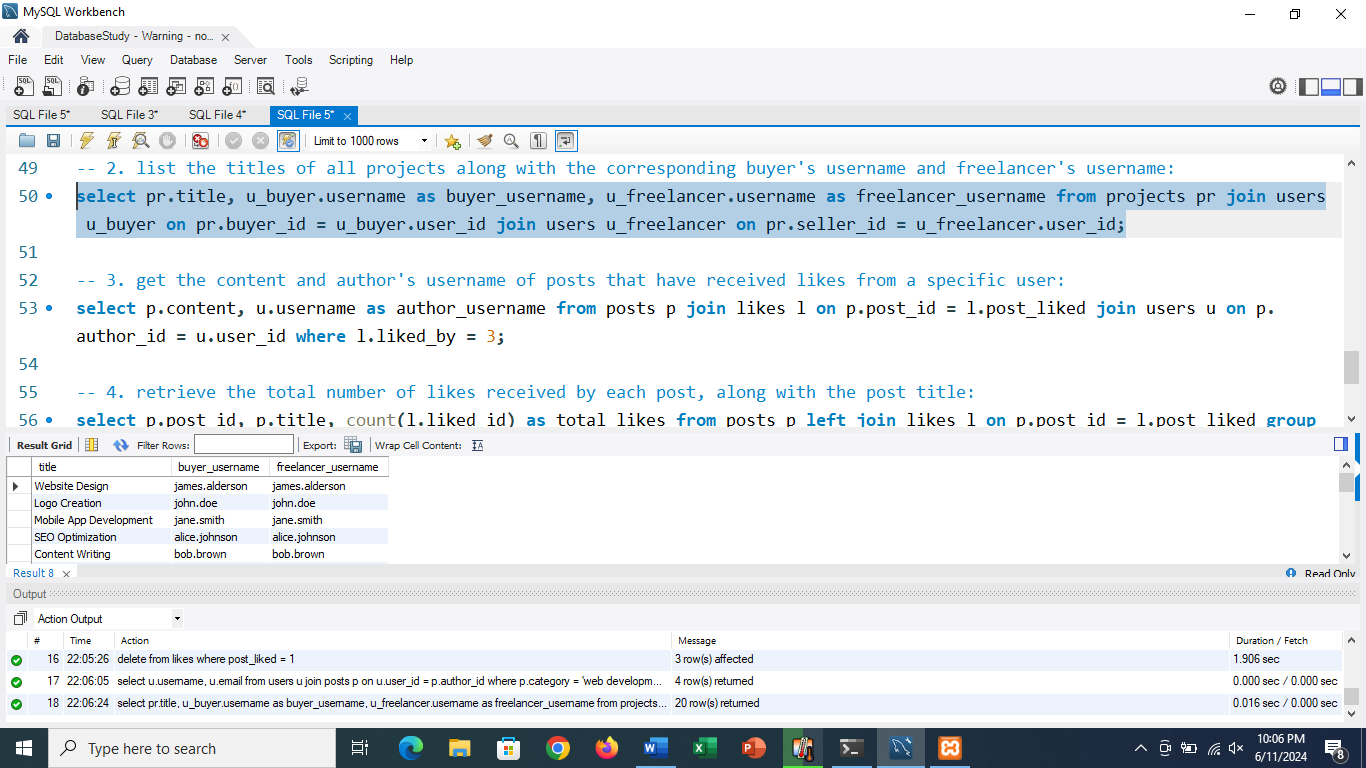
**1. retrieve the username and email of all users who have posted in the web development category:**

select u.username, u.email from users u join posts p on u.user\_id = p.author\_id where p.category = 'web development';



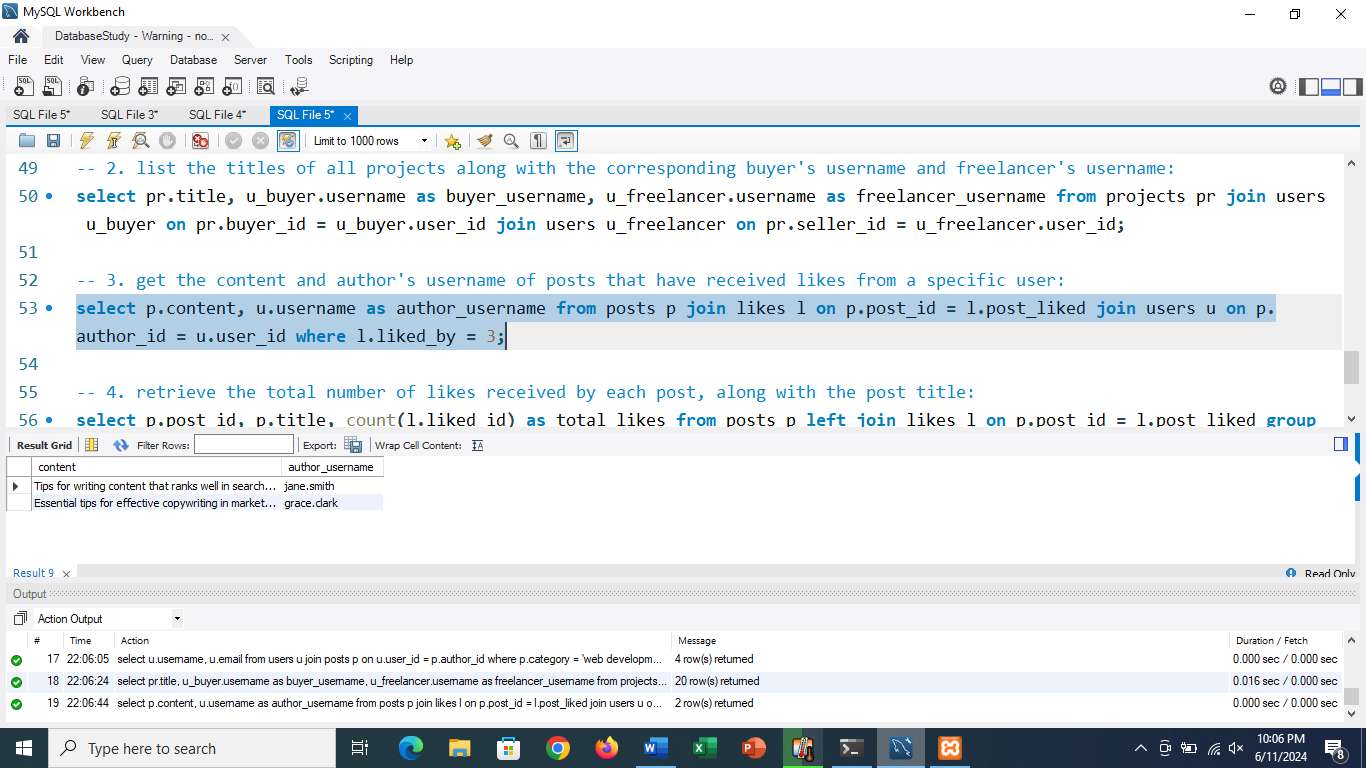
**2. list the titles of all projects along with the corresponding buyer's username and freelancer's username:**

select pr.title, u\_buyer.username as buyer\_username, u\_freelancer.username as freelancer\_username from projects pr join users u\_buyer on pr.buyer\_id = u\_buyer.user\_id join users u\_freelancer on pr.seller\_id = u\_freelancer.user\_id;



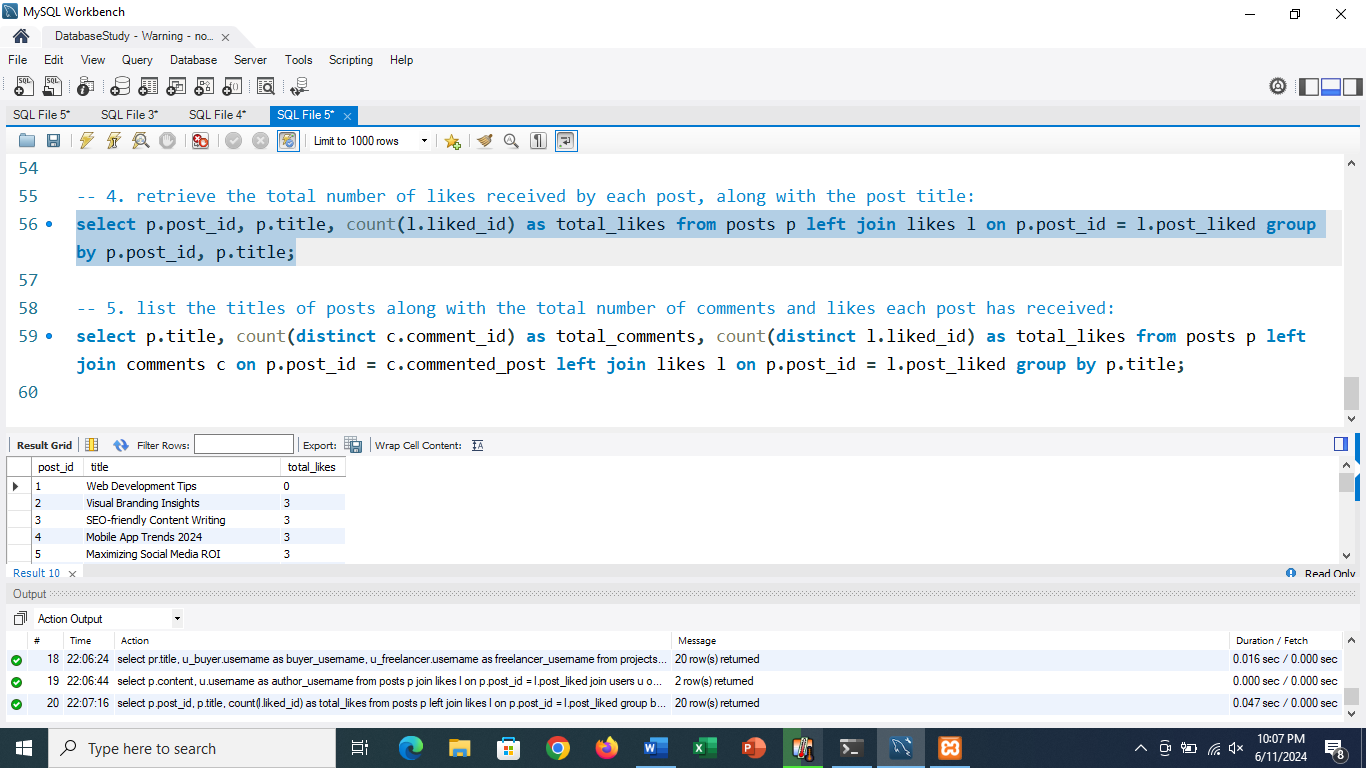
**3. get the content and author's username of posts that have received likes from a specific user:**

select p.content, u.username as author\_username from posts p join likes l on p.post\_id = l.post\_liked join users u on p.author\_id = u.user\_id where l.liked\_by = 3;



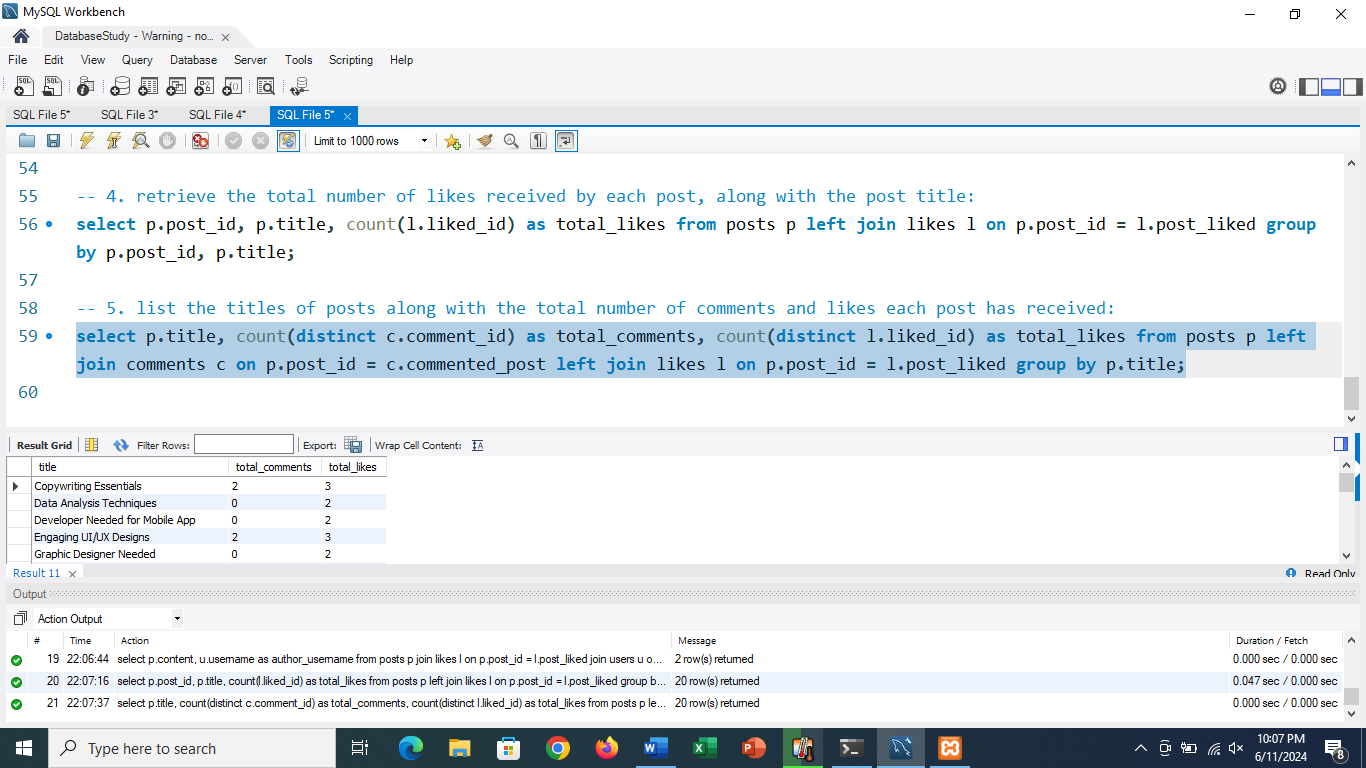
**4. retrieve the total number of likes received by each post, along with the post title:**

select p.post\_id, p.title, count(l.liked\_id) as total\_likes from posts p left join likes l on p.post\_id = l.post\_liked group by p.post\_id, p.title;



**5. list the titles of posts along with the total number of comments and likes each post has received:**

select p.title, count(distinct c.comment\_id) as total\_comments, count(distinct l.liked\_id) as total\_likes from posts p left join comments c on p.post\_id = c.commented\_post left join likes l on p.post\_id = l.post\_liked group by p.title;



use portfolio\_terrain;

-- nested queries

-- 1. retrieve the latest post content from each user:

select u.username, p.title, p.content

from users u

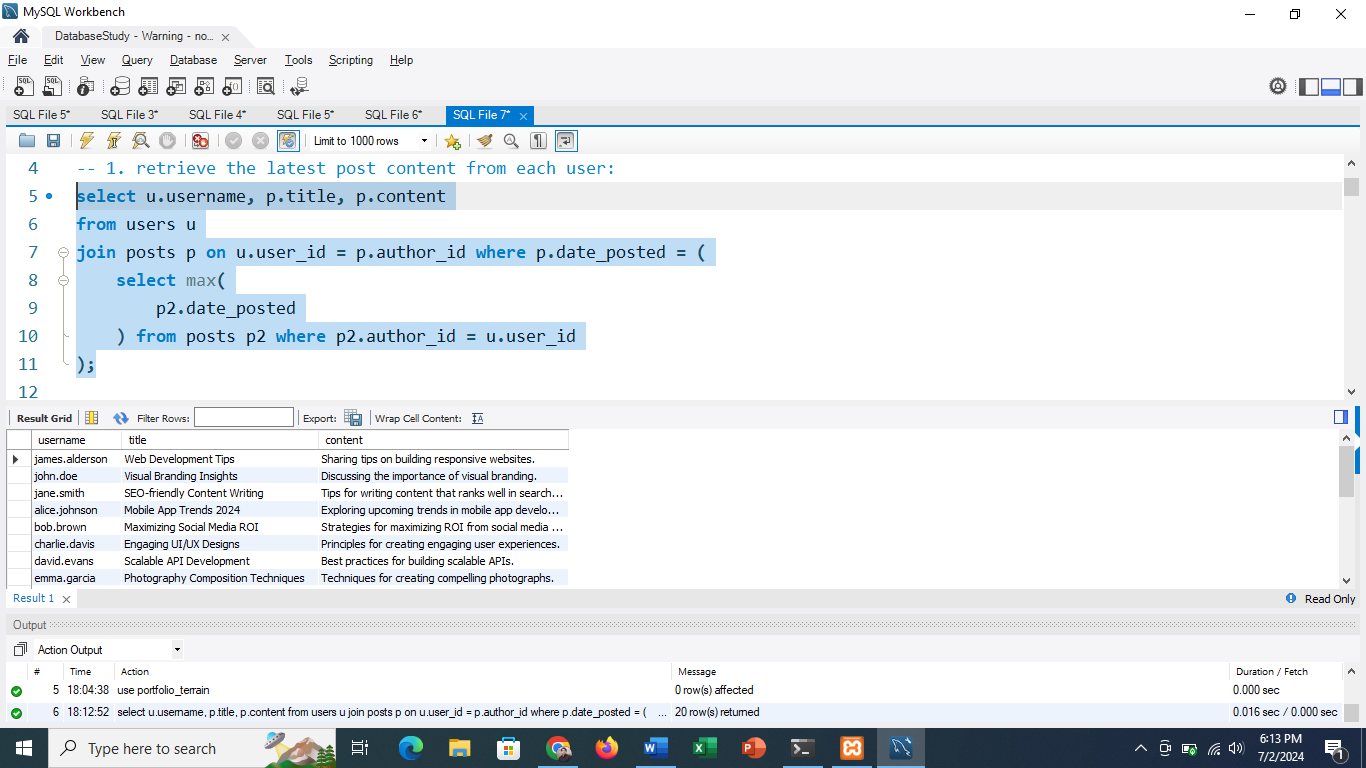
join posts p on u.user\_id = p.author\_id where p.date\_posted = (

select max(

p2.date\_posted

) from posts p2 where p2.author\_id = u.user\_id

);



-- 2. find users who have posted in the 'web development' category and the date of their latest post in that category:

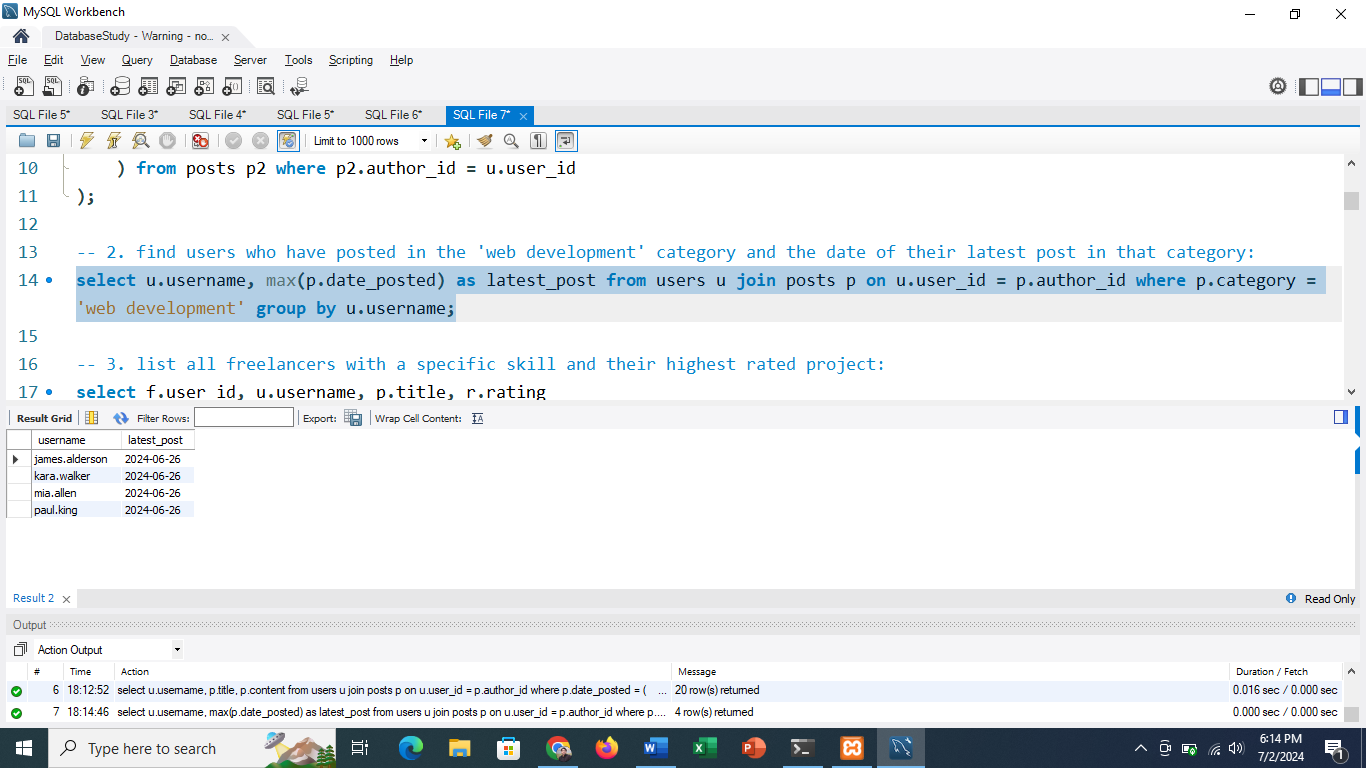
select u.username, max(p.date\_posted) as latest\_post

from users u

join posts p on u.user\_id = p.author\_id

where p.category = 'web development'

group by u.username;



-- 3. list all freelancers with a specific skill and their highest rated project:

select f.user\_id, u.username, p.title, r.rating from freelancer f join users u on f.user\_id = u.user\_id join projects p on f.freelancer\_id = p.seller\_id join reviews r on p.project\_id = r.project\_reviewed

where f.skills like '%python%'

and r.rating = (

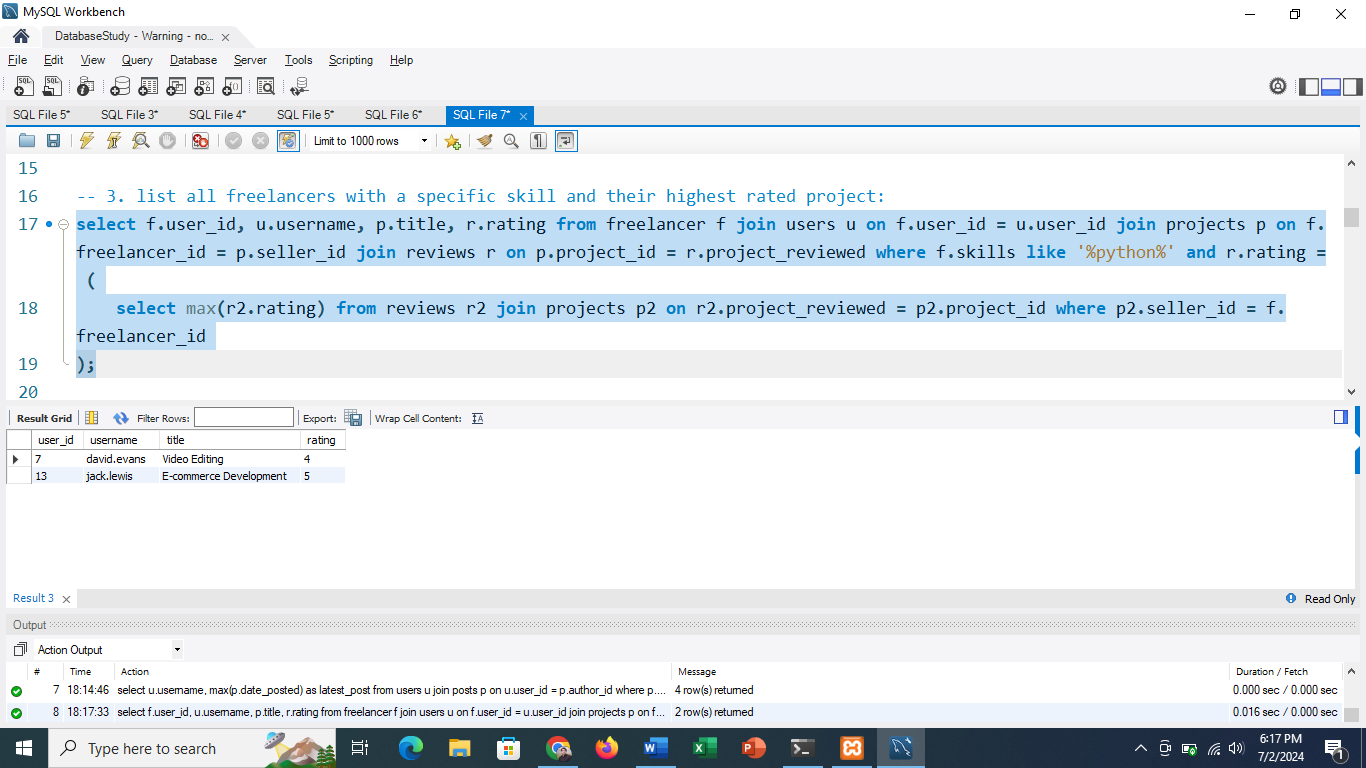
select max(r2.rating)

from reviews r2

join projects p2 on r2.project\_reviewed = p2.project\_id

where p2.seller\_id = f.freelancer\_id

);



-- 4. find the buyer who has spent the most on contracts:

select b.user\_id, u.username, sum(c.cost) as total\_spent

from buyer b

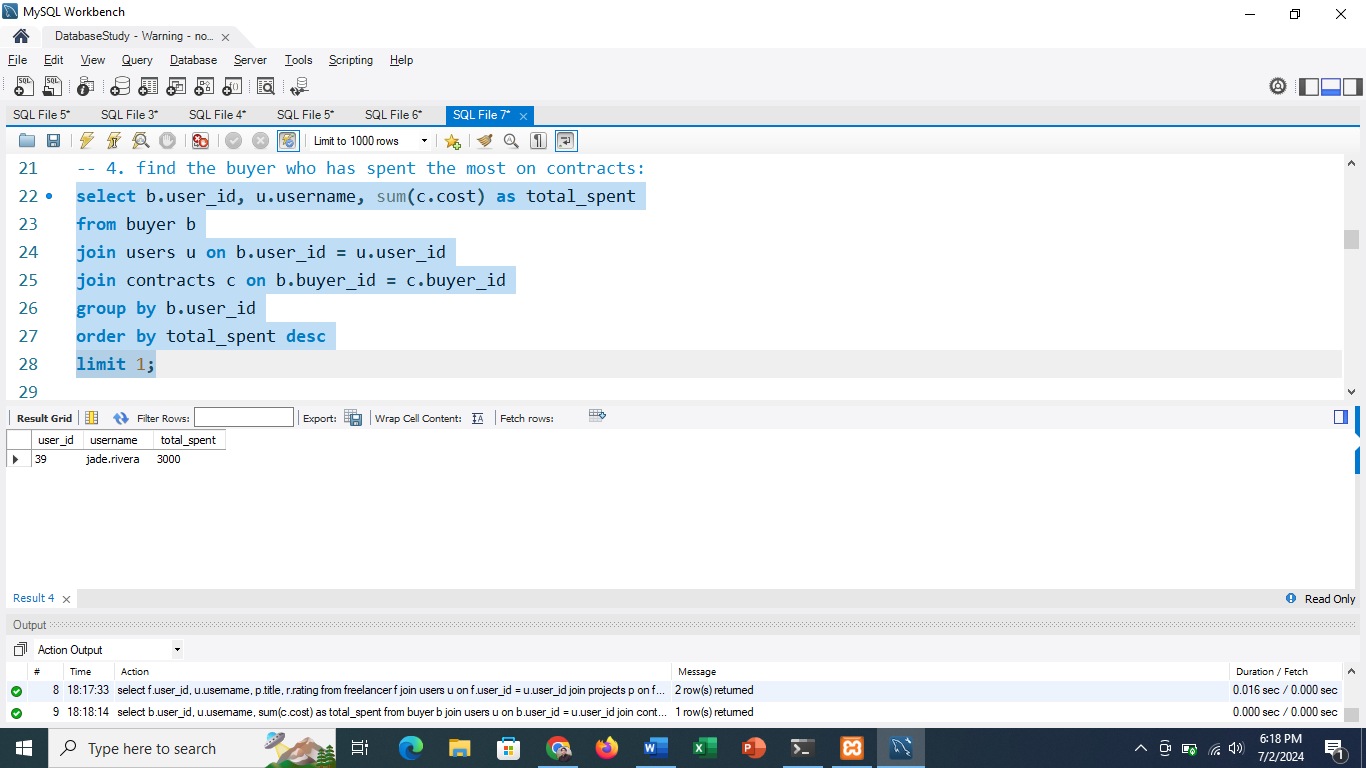
join users u on b.user\_id = u.user\_id

join contracts c on b.buyer\_id = c.buyer\_id

group by b.user\_id

order by total\_spent desc

limit 1;



-- 5. retrieve comments on the most liked post:

select c.content, u.username, c.comment\_date

from comments c

join users u on c.commenter = u.user\_id

where c.commented\_post = (

select l.post\_liked

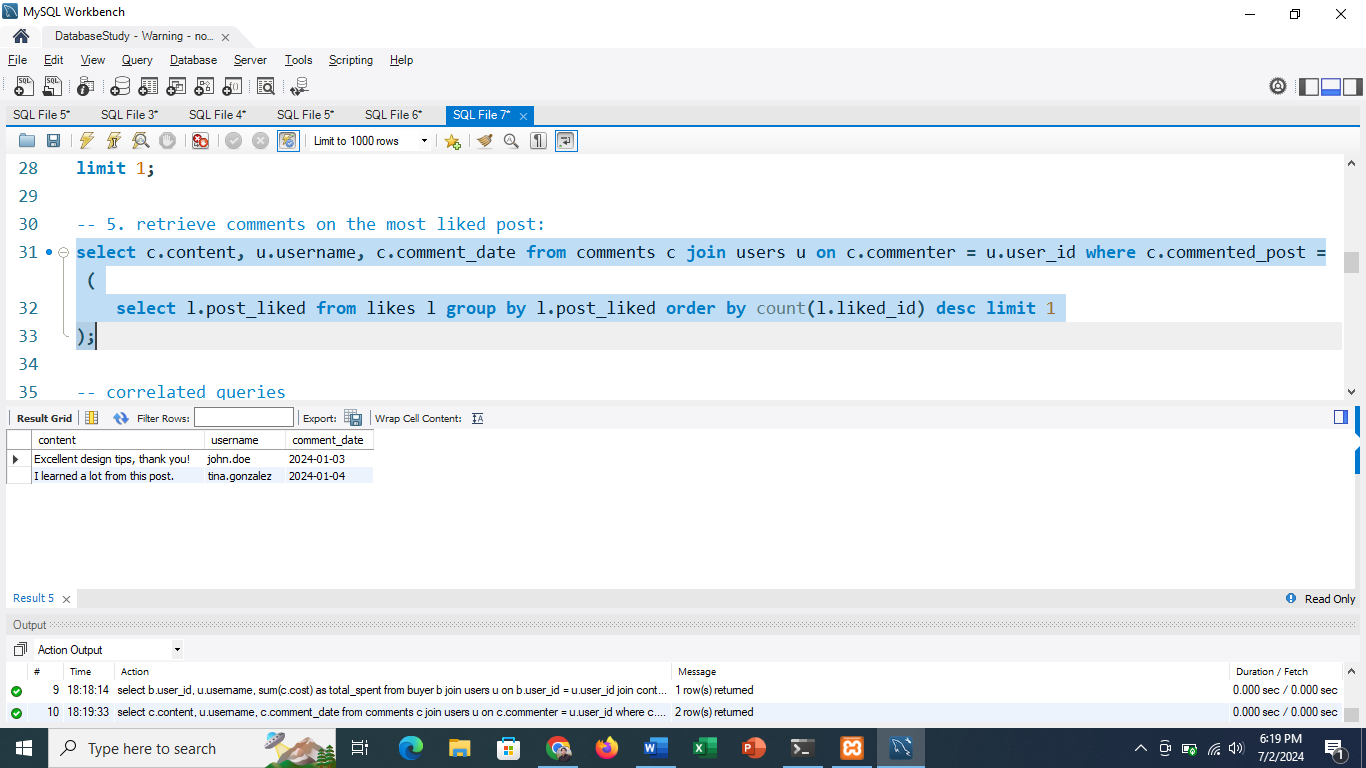
from likes l

group by l.post\_liked

order by count(l.liked\_id) desc

limit 1

);



-- correlated queries

-- 1. retrieve all posts along with the number of comments they have received:

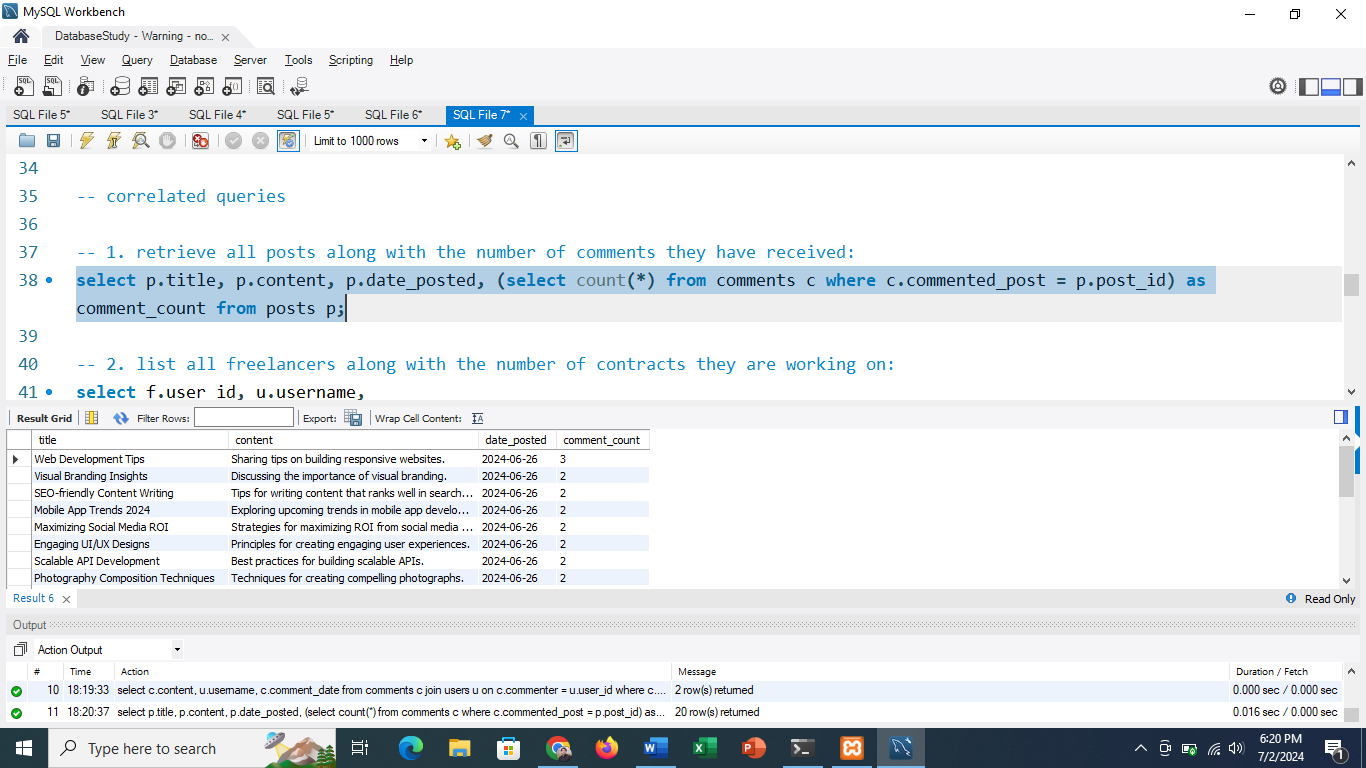
select p.title, p.content, p.date\_posted,

(select count(\*)

from comments c

where c.commented\_post = p.post\_id) as comment\_count

from posts p;



-- 2. list all freelancers along with the number of contracts they are working on:

select f.user\_id, u.username,

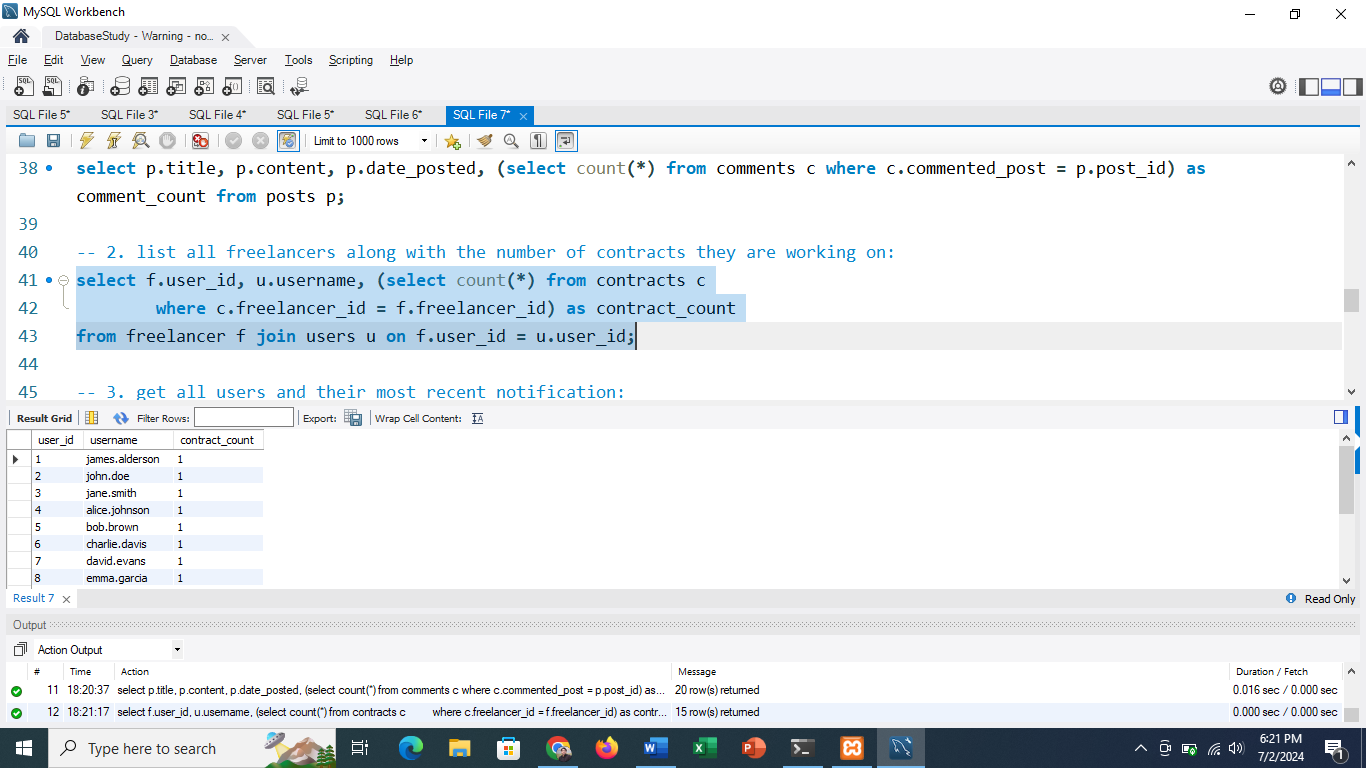
(select count(\*)

from contracts c

where c.freelancer\_id = f.freelancer\_id) as contract\_count

from freelancer f

join users u on f.user\_id = u.user\_id;



-- 3. get all users and their most recent notification:

select u.username, n.content, n.notify\_date

from users u

left join notifications n on u.user\_id = n.user\_notified

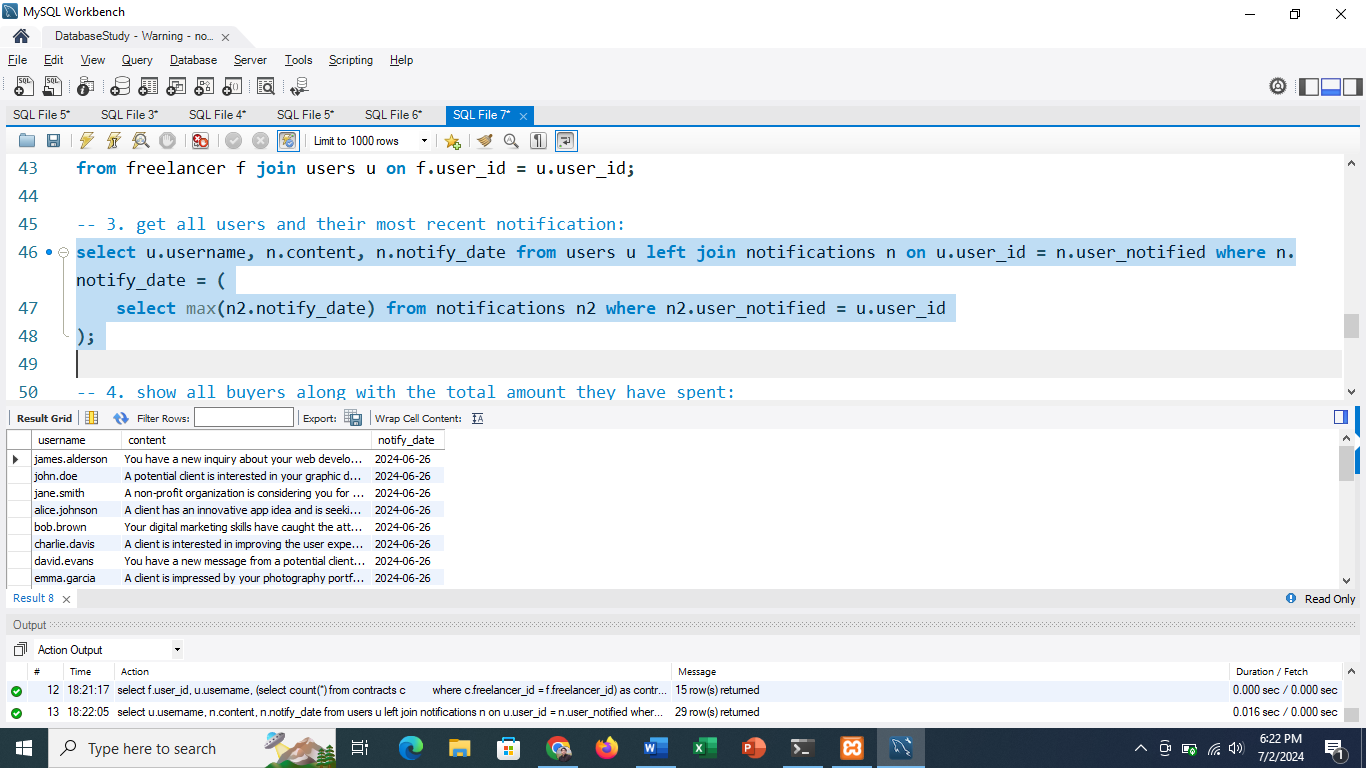
where n.notify\_date = (

select max(n2.notify\_date)

from notifications n2

where n2.user\_notified = u.user\_id

);



-- 4. show all buyers along with the total amount they have spent:

select b.user\_id, u.username,

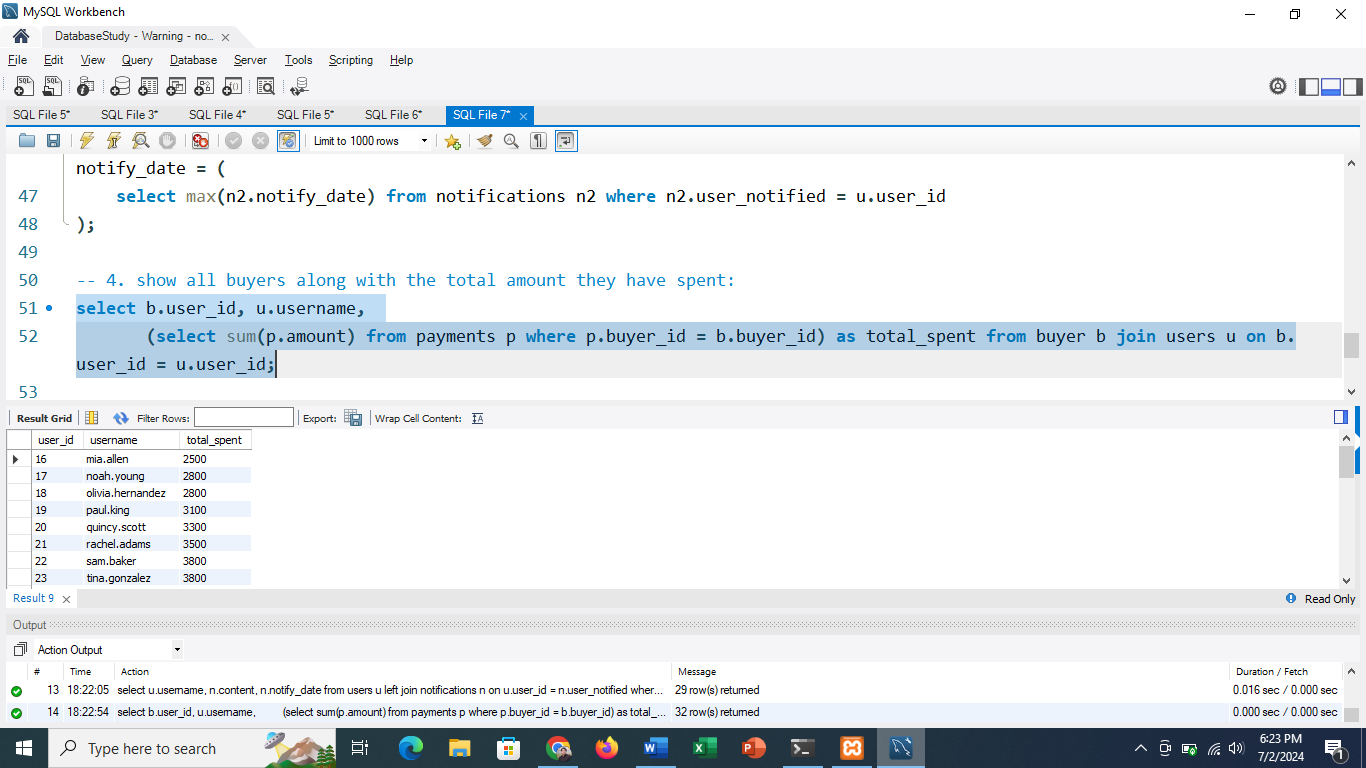
(select sum(p.amount)

from payments p

where p.buyer\_id = b.buyer\_id) as total\_spent

from buyer b

join users u on b.user\_id = u.user\_id;



-- 5. list all projects along with the highest rating they received in reviews:

select p.title, p.category, (select max(r.rating) from reviews r where r.project\_reviewed = p.project\_id) as highest\_rating

from projects p;

