Project: Data Modeling

Erickson Figueroa

Report

Executive Summary:

The project focused on managing unstructured data from a CSV file in PostgreSQL. Tasks included conceptual model construction, table creation, and development of a physical ERD. Configuring permissions to read the file and importing the data records for users, posts, and comments were integral to organizing and managing data effectively within the PostgreSQL environment.

Objective:

Create a logical data modelling starting from an unstructured CSV file, then transfer that logic to physical tables within the PostgreSQL database engine, create a physical modelling or ERD diagram, and import the necessary records from the CSV file.

Brief functionality of the process:

In the project, there was a CSV file containing unstructured data. After understanding the provided CSV file, the construction of a conceptual model diagram began. Subsequently, initiating the process involved creating tables in PostgreSQL and developing the physical diagram or ERD based on the logic of the conceptual diagram. Once the physical ERD diagram was ready, the configuration of necessary permissions for the PostgreSQL user to read the CSV file in the directory where it took place.

Following the process, a new table named blog_data inside PostgreSQL was created to import the data from the CSV file. When the imported data was ready in the blog_data table, the insertion of records for the users, posts, and comments table commenced.

The biggest challenge of the process:

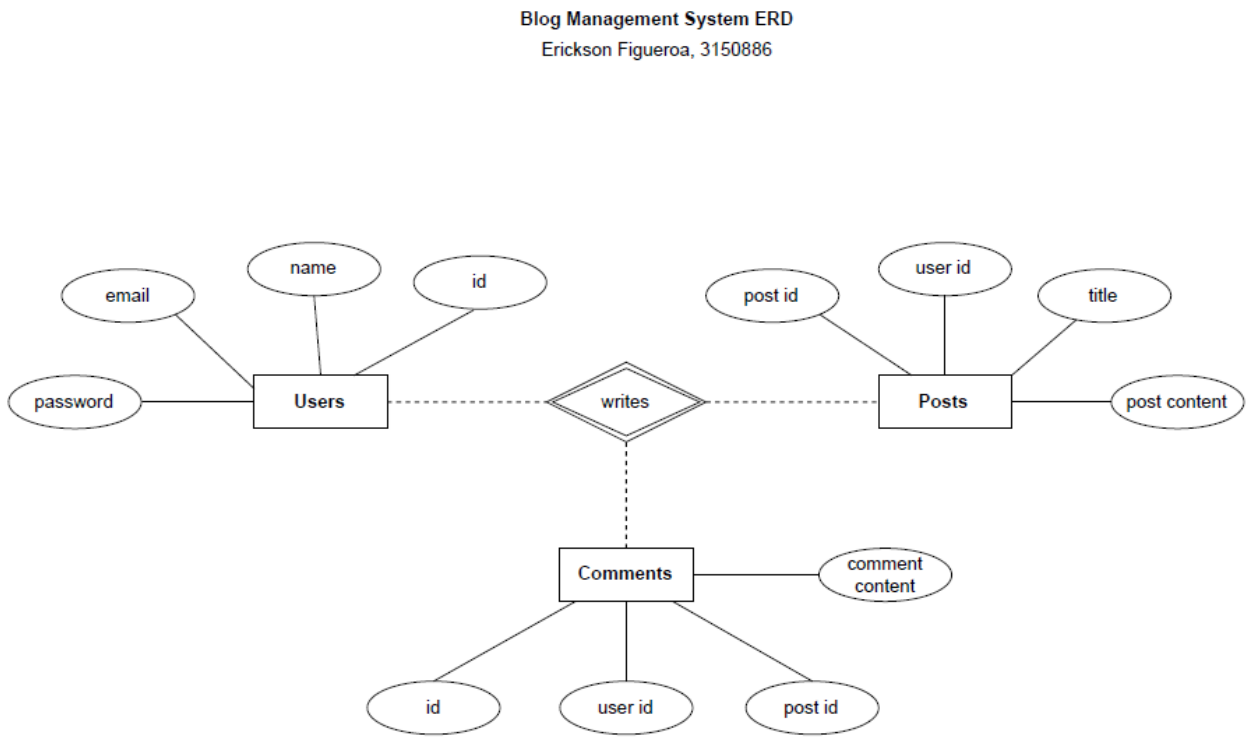
The most challenging aspect involved inserting records into the comments table, requiring the creation of a complex SQL query to split, clean, and insert the data accurately. The task

demanded meticulous attention to detail to ensure the correct separation and insertion of multiple comments within a single string for a user into the final comments table.

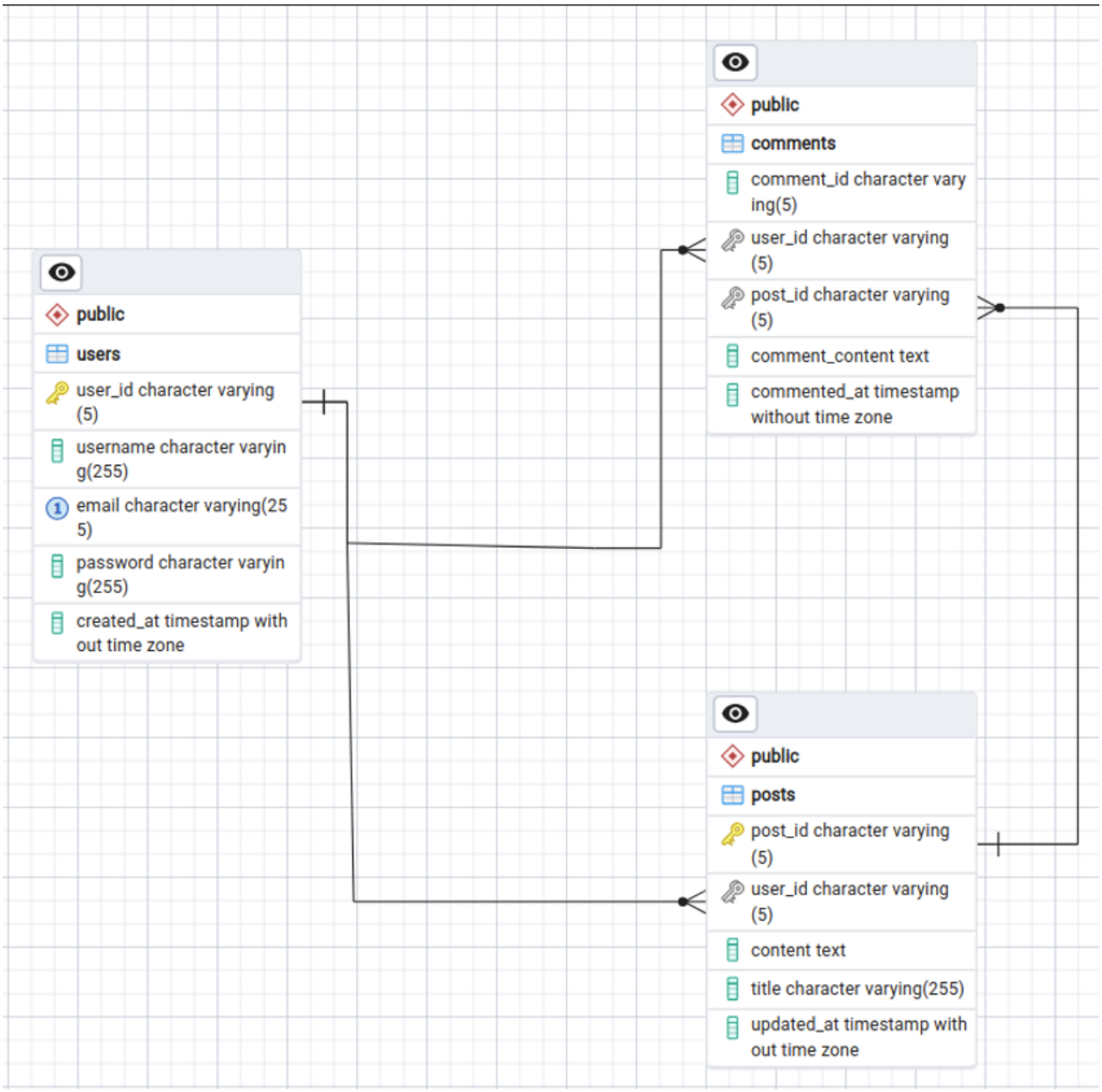
Additional tables:

blog_data: To import the records from the CSV file without any transformation.
temp_comment_string: To split the comments for those users who made more than one comment on the same post.

Conceptual Model:



Physical Model:



Unstructured CSV file:

A	B	C	D	E	F	G	H	I	J
user_id	username	email	password	created_at	post_id	content	title	upated_at	comments
uid1	sarah_c	sarah@example.com	sarapass	2024-01-16 8:10	pid3	JavaScript is a popular ...	JavaScript Basics	2024-01-16 9:30	JavaScript is amazing! 2024-01-16 08:30 cid5
uid2	emily_j	emily@example.com	secure123	2024-01-16 17:30	pid5	Build web applications using ...	Python Web Development	2024-01-16 18:00	Thanks for the tutorial! 2024-01-16 18:00 cid7
uid3	jane_s	jane@example.com	passw0rd	2024-01-14 14:45	pid1	Python is a verstile and ...	Getting Started with Python	2024-01-15 10:20	I learned a lot. 2024-01-14 16:00 cid2 | Thanks for the tutorial! 2024-01-14 15:30 cid1
uid4	michael_b	michael@example.com	p@ssw0rd	2024-01-16 12:20	pid4	Learn about various data ...	Data Structures in C++	2024-01-16 15:45	Great explaination! 2024-01-16 13:00 cid6
uid5	john_d	john@example.com	password123	2024-01-15 9:30	pid2	MongoDB is a highly scalable ...	Introduction to MongoDB	2024-01-15 9:35	Well explaind. 2024-01-15 10:00 cid4 | Great article! 2024-01-15 09:45 cid3

Granting Permission to CSV for user 'postgres' and Importing the Data:

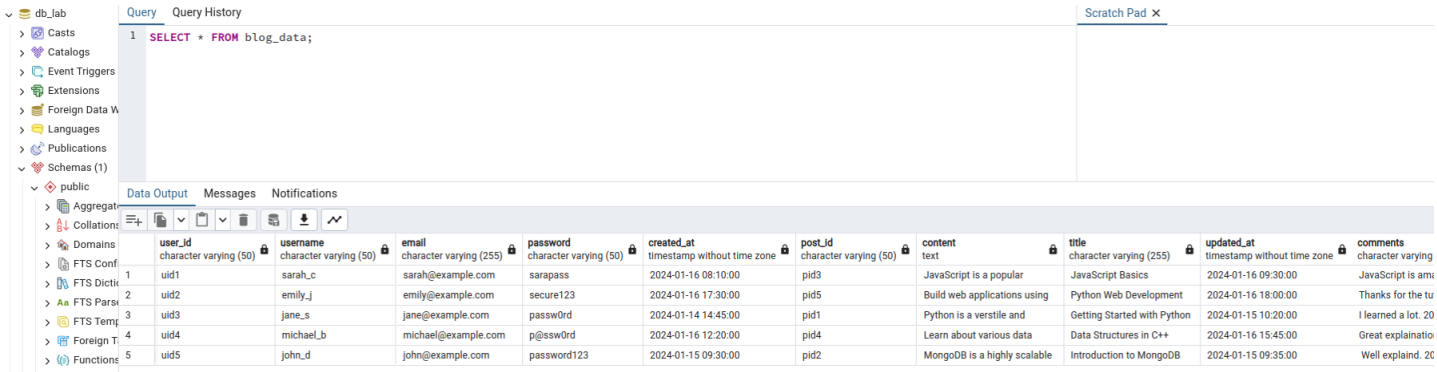
```
adminuser@erickson-lab:~$ sudo setfacl -m u:postgres:r /home/adminuser/Downloads/blogging_
dataset.csv
adminuser@erickson-lab:~$
adminuser@erickson-lab:~$
adminuser@erickson-lab:~$ psql -h localhost -U postgres -d db_lab
Password for user postgres:
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.10.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

db_lab=#
db_lab=#
db_lab=#
db_lab=# \COPY blog_data FROM '/home/adminuser/Downloads/blogging_dataset.csv' WITH DELIMI
TER ',' CSV HEADER;
COPY 5
```

Data Import Result: (blog_data table)

```
db_lab=# \x auto
Expanded display is used automatically.
db_lab=# SELECT * FROM blog_data;
```

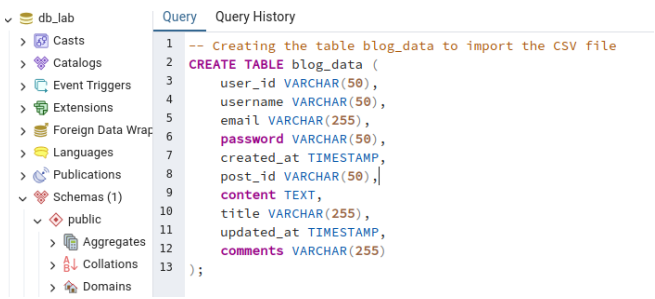
```
-[ RECORD 1 ]-----------------------------------------------------------------------
----------
user_id    | uid1
username   | sarah_c
email      | sarah@example.com
password   | sarapass
created_at | 2024-01-16 08:10:00
post_id    | pid3
content    | JavaScript is a popular
title      | JavaScript Basics
updated_at | 2024-01-16 09:30:00
comments   | JavaScript is amazing! 2024-01-16 08:30 cid5
-[ RECORD 2 ]-----------------------------------------------------------------------
----------
user_id    | uid2
username   | emily_j
email      | emily@example.com
password   | secure123
created_at | 2024-01-16 17:30:00
post_id    | pid5
content    | Build web applications using
title      | Python Web Development
updated_at | 2024-01-16 18:00:00
comments   | Thanks for the tutorial! 2024-01-16 18:00 cid7
-[ RECORD 3 ]-----------------------------------------------------------------------
----------
user_id    | uid3
username   | jane_s
email      | jane@example.com
password   | passw0rd
created_at | 2024-01-14 14:45:00
[illegible] | pid1
[illegible] | Python is a verstile and
title      | Getting Started with Python
updated_at | 2024-01-15 10:20:00
comments   | I learned a lot. 2024-01-14 16:00 cid2 | Thanks for the tutorial! 2024-01-14
15:30 cid1
-[ RECORD 4 ]-----------------------------------------------------------------------
----------
user_id    | uid4
username   | michael_b
email      | michael@example.com
password   | p@ssw0rd
created_at | 2024-01-16 12:20:00
post_id    | pid4
content    | Learn about various data
```

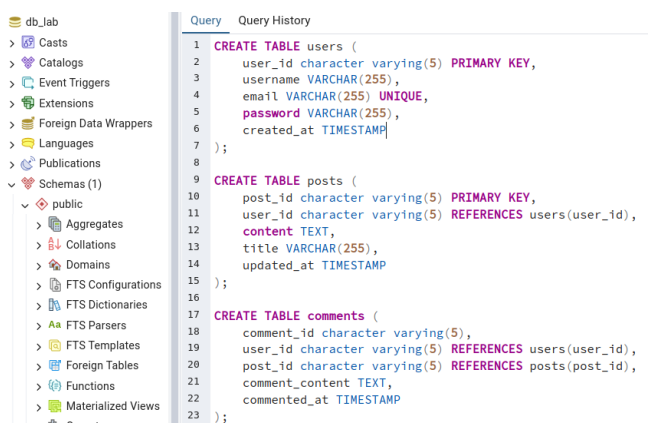
Verifying blog_data **table result in Pgadmin:**

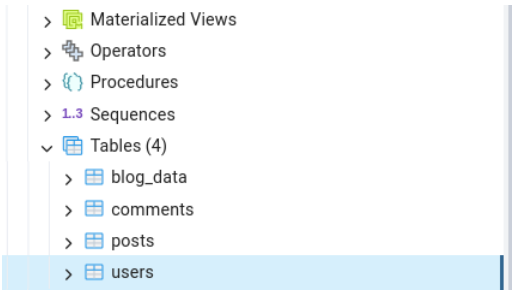


	user_id character varying (50)	username character varying (50)	email character varying (255)	password character varying (50)	created_at timestamp without time zone	post_id character varying (50)	content text	title character varying (255)	updated_at timestamp without time zone	comments character varying
1	uid1	sarah_c	sarah@example.com	sarapass	2024-01-16 08:10:00	pid3	JavaScript is a popular	JavaScript Basics	2024-01-16 09:30:00	JavaScript is ama
2	uid2	emily_j	emily@example.com	secure123	2024-01-16 17:30:00	pid5	Build web applications using	Python Web Development	2024-01-16 18:00:00	Thanks for the tu
3	uid3	jane_s	jane@example.com	passw0rd	2024-01-14 14:45:00	pid1	Python is a verstile and	Getting Started with Python	2024-01-15 10:20:00	I learned a lot. 20
4	uid4	michael_b	michael@example.com	p@ssw0rd	2024-01-16 12:20:00	pid4	Learn about various data	Data Structures in C++	2024-01-16 15:45:00	Great explainatio
5	uid5	john_d	john@example.com	password123	2024-01-15 09:30:00	pid2	MongoDB is a highly scalable	Introduction to MongoDB	2024-01-15 09:35:00	Well explaind. 20

Tables Creation Process:







Inserting Data Process:

Query Query History

```sql
-----------------------------------------------------------------
-- Queries to populate, clean and split the data in each table
-----------------------------------------------------------------

-- Inserting data to each table based on "blog_data table"
-- The blog_data table previosly created has the CSV data rows

-- Insert into users table
INSERT INTO users (user_id, username, email, password, created_at)
SELECT user_id, username, email, password, created_at FROM blog_data;


-- Insert into posts table
INSERT INTO posts (post_id, user_id, content, title, updated_at)
SELECT post_id, user_id, content, title, updated_at FROM blog_data;


-- Create a temporary table to clean and get the comments string by user_id
CREATE TEMPORARY TABLE temp_comment_string (
    original_comment VARCHAR(255) NULL,
    comment_id VARCHAR(50) NULL,
    user_id VARCHAR(50) NULL,
    post_id VARCHAR(50) NULL,
    comment_content VARCHAR(255) NULL,
    commented_at VARCHAR(50) NULL
);
```

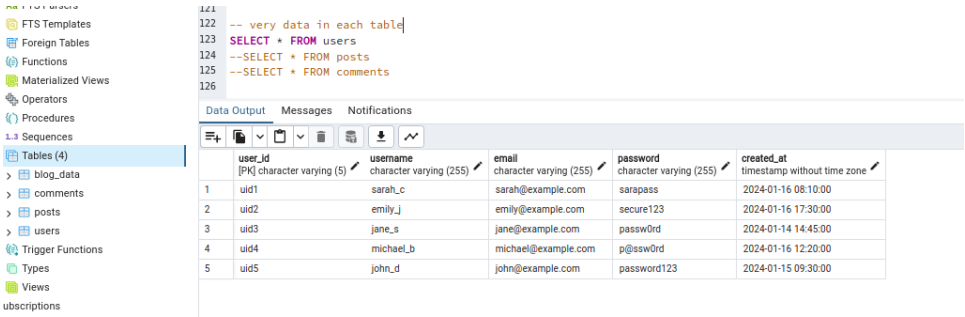
```sql
-- Insert data with more than user_id in the same comment string
INSERT INTO temp_comment_string (original_comment, comment_id, user_id, post_id, comment_content, commented_at)
WITH SplitComments AS (
    SELECT
        comments,
        user_id,
        post_id,
        SPLIT_PART(comments, '|', 1) AS comment_part_1,
        SPLIT_PART(comments, '|', 2) AS comment_part_2
    FROM
        blog_data
    WHERE
        comments LIKE '%|%'

)
SELECT
    comments AS original_comment,
    'cid' || SPLIT_PART(comment_part_1, 'cid', 2) AS comment_id,
    user_id,
    post_id,
    REGEXP_REPLACE(comment_part_1, '2024.*$', '') AS comment_content,
    REGEXP_REPLACE(comment_part_1, '^.*?(\d{4}-\d{2}-\d{2} \d{2}:\d{2}).*$', '\1') AS commented_at
FROM
    SplitComments

UNION ALL

SELECT
    comments AS original_comment,
    'cid' || SPLIT_PART(comment_part_2, 'cid', 2) AS comment_id,
    user_id,
    post_id,
    REGEXP_REPLACE(comment_part_2, '2024.*$', '') AS comment_content,
    REGEXP_REPLACE(comment_part_2, '^.*?(\d{4}-\d{2}-\d{2} \d{2}:\d{2}).*$', '\1') AS commented_at
FROM
    SplitComments;
```

```sql
-- Insert data with only one user_id in the same comment string
INSERT INTO temp_comment_string (original_comment, comment_id, user_id, post_id, comment_content, commented_at)

WITH SplitComments AS (
    SELECT
        comments,
        user_id,
        post_id,
        SPLIT_PART(comments, '|', 1) AS comment_part_1,
        SPLIT_PART(comments, '|', 2) AS comment_part_2
    FROM
        blog_data
    WHERE
        comments NOT LIKE '%|%'

)
SELECT
    comments AS original_comment,
    'cid' || SPLIT_PART(comment_part_1, 'cid', 2) AS comment_id,
    user_id,
    post_id,
    REGEXP_REPLACE(comment_part_1, '2024.*$', '') AS comment_content,
    REGEXP_REPLACE(comment_part_1, '^.*?(\d{4}-\d{2}-\d{2} \d{2}:\d{2}).*$', '\1') AS commented_at
FROM
    SplitComments

UNION ALL

SELECT
    comments AS original_comment,
    'cid' || SPLIT_PART(comment_part_2, 'cid', 2) AS comment_id,
    user_id,
    post_id,
    REGEXP_REPLACE(comment_part_2, '2024.*$', '') AS comment_content,
    REGEXP_REPLACE(comment_part_2, '^.*?(\d{4}-\d{2}-\d{2} \d{2}:\d{2}).*$', '\1') AS commented_at
FROM
    SplitComments;
```

```sql
-- Delete rows where comment_content and comented_at is empty after insert
DELETE FROM temp_comment_string WHERE comment_content = '' OR commented_at = ''


-- Finally, insert in the comments table the clean data
INSERT INTO comments (comment_id, user_id, post_id, comment_content, commented_at)
SELECT
    comment_id,
    user_id,
    post_id,
    TRIM(BOTH ' ' FROM comment_content) AS comment_content,
    commented_at::timestamp AS commented_at --> casting to avoid datetime error convertion
FROM temp_comment_string;
```

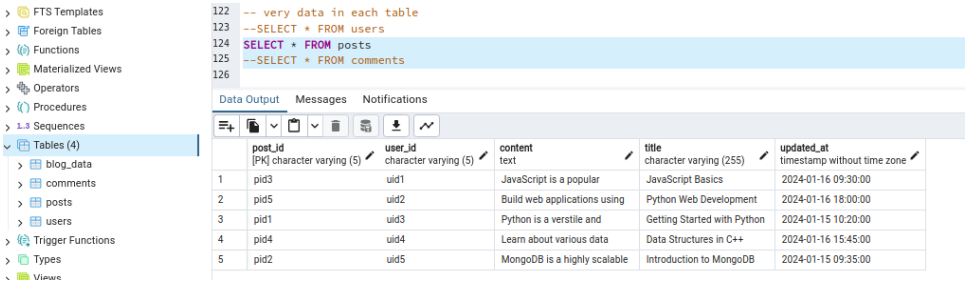
Making the following queries:

a) Retrieve all users



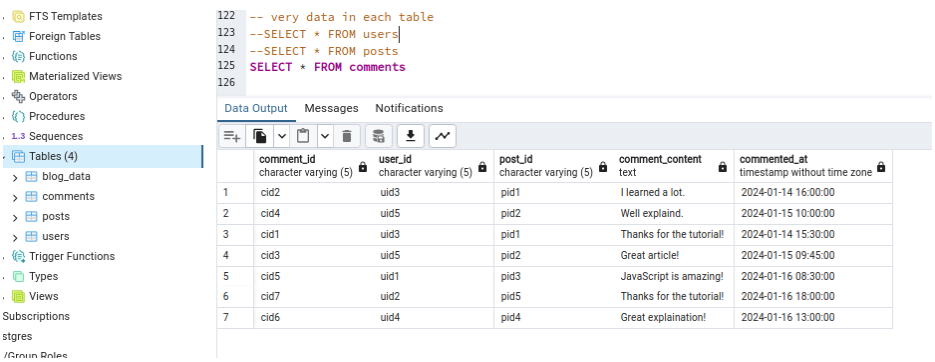
	user_id [PK] character varying (5)	username character varying (255)	email character varying (255)	password character varying (255)	created_at timestamp without time zone
1	uid1	sarah_c	sarah@example.com	sarapass	2024-01-16 08:10:00
2	uid2	emily_j	emily@example.com	secure123	2024-01-16 17:30:00
3	uid3	jane_s	jane@example.com	passw0rd	2024-01-14 14:45:00
4	uid4	michael_b	michael@example.com	p@ssw0rd	2024-01-16 12:20:00
5	uid5	john_d	john@example.com	password123	2024-01-15 09:30:00

b) Retrieve all posts



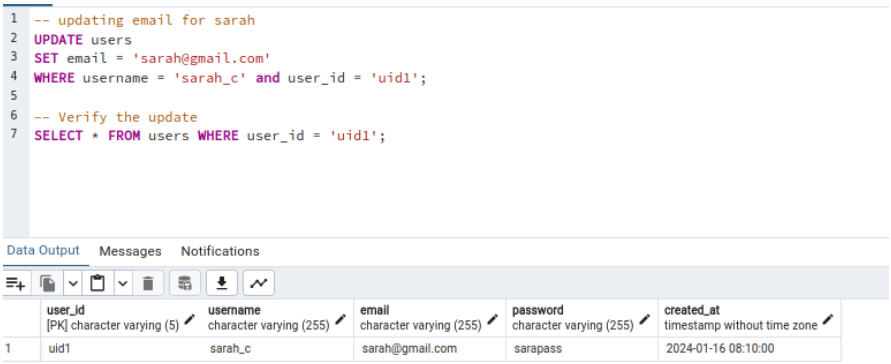
	post_id [PK] character varying (5)	user_id character varying (5)	content text	title character varying (255)	updated_at timestamp without time zone
1	pid3	uid1	JavaScript is a popular	JavaScript Basics	2024-01-16 09:30:00
2	pid5	uid2	Build web applications using	Python Web Development	2024-01-16 18:00:00
3	pid1	uid3	Python is a verstile and	Getting Started with Python	2024-01-15 10:20:00
4	pid4	uid4	Learn about various data	Data Structures in C++	2024-01-16 15:45:00
5	pid2	uid5	MongoDB is a highly scalable	Introduction to MongoDB	2024-01-15 09:35:00

c) Retrieve all comments



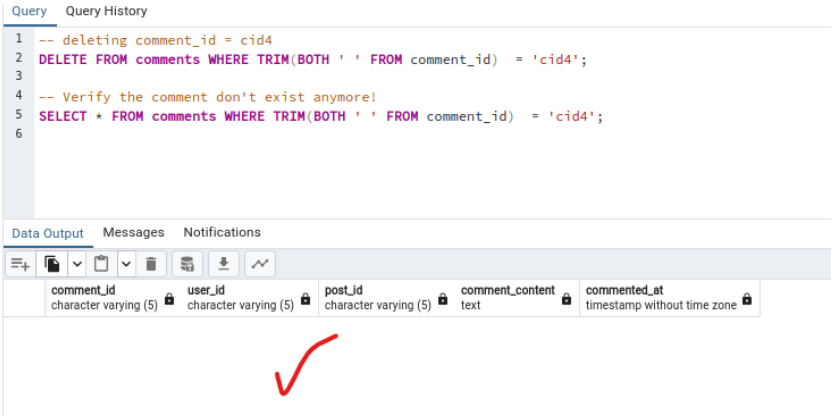
	comment_id character varying (5)	user_id character varying (5)	post_id character varying (5)	comment_content text	commented_at timestamp without time zone
1	cid2	uid3	pid1	I learned a lot.	2024-01-14 16:00:00
2	cid4	uid5	pid2	Well explaind.	2024-01-15 10:00:00
3	cid1	uid3	pid1	Thanks for the tutorial!	2024-01-14 15:30:00
4	cid3	uid5	pid2	Great article!	2024-01-15 09:45:00
5	cid5	uid1	pid3	JavaScript is amazing!	2024-01-16 08:30:00
6	cid7	uid2	pid5	Thanks for the tutorial!	2024-01-16 18:00:00
7	cid6	uid4	pid4	Great explaination!	2024-01-16 13:00:00

d) Update the user email of sarah_c to sarab@gmail.com



	user_id [PK] character varying (5)	username character varying (255)	email character varying (255)	password character varying (255)	created_at timestamp without time zone
1	uid1	sarah_c	sarah@gmail.com	sarapass	2024-01-16 08:10:00

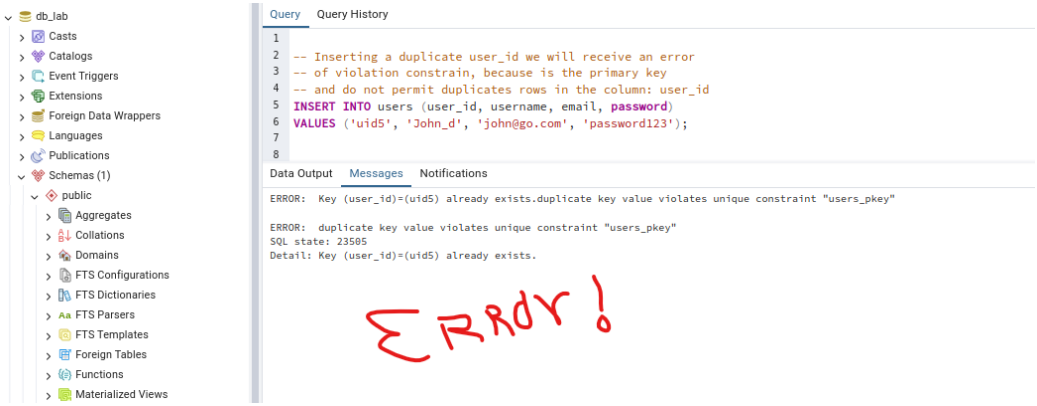
e) Delete comment id cid4



f) What happens if you insert the data below into the user table?

User-id: uid5, Username: John_d, Email: john@go.com , Password: password123

Error of primary key violation, because duplicate **userid = uid5**



The SQL source code

ALL SQL Queries.sql

Dataset

blogging_dataset.csv