

Udiddit, a social news aggregator

Introduction

Udiddit, a social news aggregation, web content rating, and discussion website, is currently using a risky and unreliable Postgres database schema to store the forum posts, discussions, and votes made by their users about different topics.

The schema allows posts to be created by registered users on certain topics, and can include a URL or a text content. It also allows registered users to cast an upvote (like) or downvote (dislike) for any forum post that has been created. In addition to this, the schema also allows registered users to add comments on posts.

Here is the DDL used to create the schema:

```
CREATE TABLE bad_posts (  
    id SERIAL PRIMARY KEY,  
    topic VARCHAR(50),  
    username VARCHAR(50),  
    title VARCHAR(150),  
    url VARCHAR(4000) DEFAULT NULL,  
    text_content TEXT DEFAULT NULL,  
    upvotes TEXT,  
    downvotes TEXT  
);  
  
CREATE TABLE bad_comments (  
    id SERIAL PRIMARY KEY,  
    username VARCHAR(50),  
    post_id BIGINT,  
    text_content TEXT  
);
```

Part I: Investigate the existing schema

As a first step, investigate this schema and some of the sample data in the project's SQL workspace. Then, in your own words, outline three (3) specific things that could be improved about this schema. Don't hesitate to outline more if you want to stand out!

1.Username and User ID Relationship:

Observation: In the "users" table, there is a "username_id" as a serial primary key and a "username" field. However, in other tables like "posts," "comments," and "votes," the relationships are established using the "username" field rather than the "username_id."

Suggestion: It would be beneficial to consistently use the "username_id" as a foreign key in related tables to establish relationships. This can enhance data consistency and make the schema more maintainable.

2.Post Votes Table Structure:

Observation: The "post_votes" table uses a separate table for upvotes and downvotes, creating two CTEs ("bad_posts_upvotes" and "bad_posts_downvotes"). This may lead to redundancy and complex queries.

Suggestion: Consider redesigning the "post_votes" table to have a single column for votes, where positive values represent upvotes and negative values represent downvotes. This could simplify queries and reduce the need for multiple CTEs.

3.User Table Indexing:

Observation: While an index is created on the "log_in" column, there is no index on the "username" column in the "users" table.

Suggestion: Create an index on the "username" column, especially since it is used in join operations with other tables. Indexing can significantly improve the performance of queries that involve filtering or joining based on the "username" field.

Part II: Create the DDL for your new schema

Having done this initial investigation and assessment, your next goal is to dive deep into the heart of the problem and create a new schema for Udiddit. Your new schema should at least reflect fixes to the shortcomings you pointed to in the previous exercise. To help you create the new schema, a few guidelines are provided to you:

1. Guideline #1: here is a list of features and specifications that Udiddit needs in order to support its website and administrative interface:
 - a. Allow new users to register:
 - i. Each username has to be unique
 - ii. Usernames can be composed of at most 25 characters
 - iii. Usernames can't be empty
 - iv. We won't worry about user passwords for this project
 - b. Allow registered users to create new topics:
 - i. Topic names have to be unique.
 - ii. The topic's name is at most 30 characters
 - iii. The topic's name can't be empty
 - iv. Topics can have an optional description of at most 500 characters.
 - c. Allow registered users to create new posts on existing topics:
 - i. Posts have a required title of at most 100 characters
 - ii. The title of a post can't be empty.
 - iii. Posts should contain either a URL or a text content, **but not both**.
 - iv. If a topic gets deleted, all the posts associated with it should be automatically deleted too.
 - v. If the user who created the post gets deleted, then the post will remain, but it will become dissociated from that user.
 - d. Allow registered users to comment on existing posts:
 - i. A comment's text content can't be empty.
 - ii. Contrary to the current linear comments, the new structure should allow comment threads at arbitrary levels.
 - iii. If a post gets deleted, all comments associated with it should be automatically deleted too.
 - iv. If the user who created the comment gets deleted, then the comment will remain, but it will become dissociated from that user.
 - v. If a comment gets deleted, then all its descendants in the thread structure should be automatically deleted too.
 - e. Make sure that a given user can only vote once on a given post:
 - i. Hint: you can store the (up/down) value of the vote as the values 1 and -1 respectively.
 - ii. If the user who cast a vote gets deleted, then all their votes will remain, but will become dissociated from the user.

- iii. If a post gets deleted, then all the votes for that post should be automatically deleted too.
2. Guideline #2: here is a list of queries that Uddidit needs in order to support its website and administrative interface. Note that you don't need to produce the DQL for those queries: they are only provided to guide the design of your new database schema.
 - a. List all users who haven't logged in in the last year.
 - b. List all users who haven't created any post.
 - c. Find a user by their username.
 - d. List all topics that don't have any posts.
 - e. Find a topic by its name.
 - f. List the latest 20 posts for a given topic.
 - g. List the latest 20 posts made by a given user.
 - h. Find all posts that link to a specific URL, for moderation purposes.
 - i. List all the top-level comments (those that don't have a parent comment) for a given post.
 - j. List all the direct children of a parent comment.
 - k. List the latest 20 comments made by a given user.
 - l. Compute the score of a post, defined as the difference between the number of upvotes and the number of downvotes
3. Guideline #3: you'll need to use normalization, various constraints, as well as indexes in your new database schema. You should use named constraints and indexes to make your schema cleaner.
4. Guideline #4: your new database schema will be composed of five (5) tables that should have an auto-incrementing id as their primary key.

Once you've taken the time to think about your new schema, write the DDL for it in the space provided here:

```
-- CREATE TABLES
-- Table for Users
CREATE TABLE users (
  user_id SERIAL PRIMARY KEY,
  username VARCHAR(25) UNIQUE NOT NULL,
  registration_date TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP,
  -- Additional user-related fields can be added in the future
);

-- Table for Topics
```

```

CREATE TABLE topics (
    topic_id SERIAL PRIMARY KEY,
    topic_name VARCHAR(30) UNIQUE NOT NULL,
    description VARCHAR(500),
    -- Additional topic-related fields can be added in the future
);

-- Table for Posts
CREATE TABLE posts (
    post_id SERIAL PRIMARY KEY,
    topic_id INTEGER REFERENCES topics(topic_id) ON DELETE CASCADE,
    user_id INTEGER REFERENCES users(user_id) ON DELETE SET NULL,
    time_stamp_post TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    title VARCHAR(100) NOT NULL,
    url VARCHAR(4000),
    text_content TEXT,
    -- Additional post-related fields can be added in the future
);

-- Table for Comments
CREATE TABLE comments (
    comment_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(user_id) ON DELETE SET NULL,
    post_id INTEGER REFERENCES posts(post_id) ON DELETE CASCADE,
    text_content TEXT NOT NULL,
    time_stamp_comment TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP,
    parent_comment_id INTEGER REFERENCES comments(comment_id) ON DELETE
CASCADE,
    -- Additional comment-related fields can be added in the future
);

-- Table for Votes
CREATE TABLE votes (
    user_id INTEGER REFERENCES users(user_id) ON DELETE SET NULL,
    post_id INTEGER REFERENCES posts(post_id) ON DELETE CASCADE,
    vote_value INTEGER CHECK(vote_value = 1 OR vote_value = -1),
    PRIMARY KEY (user_id, post_id),
    -- Additional vote-related fields can be added in the future
);

-- Indexes
CREATE INDEX idx_username ON users (username);
CREATE INDEX idx_topic_name ON topics (topic_name);
CREATE INDEX idx_post_topic_id ON posts (topic_id);
CREATE INDEX idx_post_user_id ON posts (user_id);
CREATE INDEX idx_comment_user_id ON comments (user_id);
CREATE INDEX idx_comment_post_id ON comments (post_id);
CREATE INDEX idx_vote_user_id ON votes (user_id);
CREATE INDEX idx_vote_post_id ON votes (post_id);

```


Part III: Migrate the provided data

Now that your new schema is created, it's time to migrate the data from the provided schema in the project's SQL Workspace to your own schema. This will allow you to review some DML and DQL concepts, as you'll be using INSERT...SELECT queries to do so. Here are a few guidelines to help you in this process:

1. Topic descriptions can all be empty
2. Since the bad_comments table doesn't have the threading feature, you can migrate all comments as top-level comments, i.e. without a parent
3. You can use the Postgres string function **regexp_split_to_table** to unwind the comma-separated votes values into separate rows
4. Don't forget that some users only vote or comment, and haven't created any posts. You'll have to create those users too.
5. The order of your migrations matter! For example, since posts depend on users and topics, you'll have to migrate the latter first.
6. Tip: You can start by running only SELECTs to fine-tune your queries, and use a LIMIT to avoid large data sets. Once you know you have the correct query, you can then run your full INSERT...SELECT query.
7. **NOTE:** The data in your SQL Workspace contains thousands of posts and comments. The DML queries may take at least 10-15 seconds to run.

Write the DML to migrate the current data in bad_posts and bad_comments to your new database schema:

```
-- Migrate Users
INSERT INTO users (username)
SELECT DISTINCT username FROM bad_posts;

-- Migrate Topics
INSERT INTO topics (topic_name)
SELECT DISTINCT topic FROM bad_posts;

-- Migrate Posts
INSERT INTO posts (topic_id, user_id, time_stamp_post, title, url,
text_content)
SELECT
    t.topic_id,
    u.user_id,
    bp.time_stamp_post,
    LEFT(bp.title, 100) AS title,
    bp.url,
    bp.text_content
```

```

FROM
    bad_posts bp
    JOIN users u ON bp.username = u.username
    JOIN topics t ON bp.topic = t.topic_name
WHERE
    LENGTH(TRIM(bp.title)) <= 100;

-- Migrate Comments (as top-level comments)
INSERT INTO comments (user_id, post_id, text_content,
time_stamp_comment)
SELECT
    u.user_id,
    p.post_id,
    bc.text_content,
    bc.time_stamp_comment
FROM
    bad_comments bc
    JOIN users u ON bc.username = u.username
    JOIN posts p ON bc.post_id = p.post_id;

-- Migrate Votes
INSERT INTO votes (user_id, post_id, vote_value)
WITH votes_split AS (
    SELECT
        u.user_id,
        p.post_id,
        regexp_split_to_table(p.upvotes, ',') AS upvote
    FROM
        bad_posts p
        JOIN users u ON regexp_split_to_table(p.upvotes, ',') =
u.username
)
SELECT
    v.user_id,
    v.post_id,
    1 AS vote_value
FROM
    votes_split v

UNION ALL

SELECT
    v.user_id,
    v.post_id,
    -1 AS vote_value
FROM
    votes_split v;

```


All information

Link information GitHub: <https://github.com/Helda21/Project-Udiddit-A-Social-News-Aggregator>

The screenshot displays the SQL Terminal Workspace application interface. The top navigation bar includes the UDACITY logo, "My Programs", "Discover", and "Catalog" menus, along with a search icon and user profile icons. The main content area is titled "SQL Terminal Workspace" and features a "Project Guide" on the left with instructions to start PostgreSQL and initialize the database. The central terminal window shows a PostgreSQL session with the following commands and outputs:

```
psql
DETAIL: Key (id)=(90751) already exists.
postgres=# SELECT * FROM users;
ERROR: relation "users" does not exist
LINE 1: SELECT * FROM users;
          ^
postgres=#
postgres=# CREATE TABLE bad_posts (
postgres=# id SERIAL PRIMARY KEY,
postgres=# topic VARCHAR(50),
postgres=# username VARCHAR(50),
postgres=# title VARCHAR(150),
postgres=# url VARCHAR(4000) DEFAULT NULL,
postgres=# text_content TEXT DEFAULT NULL,
postgres=# downvotes TEXT;
postgres=# );
ERROR: relation "bad_posts" already exists
postgres=# CREATE TABLE bad_comments (
postgres=# id SERIAL PRIMARY KEY,
postgres=# username VARCHAR(50),
postgres=# post_id BIGINT,
postgres=# text_content TEXT;
postgres=# );
ERROR: relation "bad_comments" already exists
postgres=# \s
postgres=# select * from bad_comments;
ERROR: syntax error at or near "\s"
LINE 1: \s
          ^
postgres=# CREATE INDEX log_in_index ON users (log_in);
ERROR: relation "users" does not exist
postgres=# CREATE TABLE users (
postgres=# -- 2.b
postgres=# username SERIAL PRIMARY KEY,
postgres=# -- 1.a.i, 1.a.ii, 1.a.iii, and 1.c
postgres=# username VARCHAR(25) CONSTRAINT required_unique_username UNIQUE NOT NULL,
postgres=# log_in TIMESTAMPTZ WITH TIME ZONE;
postgres=# );
CREATE TABLE
postgres=# CREATE INDEX log_in_index ON users (log_in);
CREATE INDEX
postgres=# CREATE INDEX username_index ON users (username VARCHAR_PATTERN_OPS);
CREATE INDEX
postgres=# CREATE TABLE topics (
postgres=# -- 2.d
postgres=# topic_id SERIAL PRIMARY KEY,
postgres=# -- 1.b.i, 1.b.ii, and 1.b.iii
postgres=# topic VARCHAR(50) CONSTRAINT required_unique_topic UNIQUE NOT NULL,
postgres=# -- 1.b.iv
postgres=# topic_description VARCHAR(500);
postgres=# );
CREATE TABLE
postgres=# CREATE INDEX topic_index ON topics (topic VARCHAR_PATTERN_OPS);
CREATE INDEX
postgres=# CREATE TABLE posts (
postgres=# -- 1.c and 1.c.iv
postgres=# topic_id INTEGER REFERENCES topics ON DELETE CASCADE,
postgres=# -- 1.c and 1.c.v
postgres=# user_id INTEGER REFERENCES users ON DELETE SET NULL,
postgres=# time_stamp_post TIMESTAMPTZ WITH TIME ZONE,
postgres=# -- 1.c.i and 1.c.ii
postgres=# title VARCHAR(150) CONSTRAINT required_title NOT NULL,
postgres=# url VARCHAR(4000) DEFAULT NULL,
postgres=# text_content TEXT DEFAULT NULL;
```

The bottom of the interface shows a Windows taskbar with the system clock at 9:14 p.m. on 17/12/2023.

SQL Terminal Workspace

[illegible]

SQL Terminal Workspace

The screenshot shows a terminal window with the following SQL commands and their outputs:

```

CREATE DATABASE posts;
CREATE TABLE posts (
  topic_id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users ON DELETE SET NULL,
  time_stamp COMMENT TIMESTAMPTZ WITH TIME ZONE,
  level INTEGER REFERENCES comments ON DELETE CASCADE,
  parent_id INTEGER REFERENCES comments ON DELETE CASCADE,
  votes INTEGER CONSTRAINT up_down_vote CHECK(votes=1 OR votes=-1)
);
CREATE INDEX latest_posts_per_topic ON posts (topic_id,time_stamp);
CREATE INDEX latest_posts_per_user ON posts (topic_id,user_id);
CREATE INDEX post_url_moderation ON posts (url VARCHAR_PATTERN_OPS);
CREATE TABLE comments (
  id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users ON DELETE SET NULL,
  post_id INTEGER REFERENCES posts ON DELETE CASCADE,
  text_content TEXT CONSTRAINT required_text_content NOT NULL,
  time_stamp COMMENT TIMESTAMPTZ WITH TIME ZONE,
  level INTEGER REFERENCES comments ON DELETE CASCADE,
  parent_id INTEGER REFERENCES comments ON DELETE CASCADE
);
CREATE INDEX level_index ON comments (level);
CREATE INDEX parent_id_index ON comments (parent_id);
CREATE TABLE votes (
  user_id INTEGER REFERENCES users ON DELETE SET NULL,
  post_id INTEGER REFERENCES posts ON DELETE CASCADE,
  vote INTEGER CONSTRAINT up_down_vote CHECK(votes=1 OR votes=-1)
);
CREATE INDEX comments_by_user ON comments (user_id,time_stamp,comment);
CREATE INDEX votes_by_user ON votes (user_id,post_id);

```

The terminal also shows the 'psql' command being used to connect to the database:

```

psql -h localhost -U postgres -d postgres
psql> \c posts
psql (15.1)
Type "help" for help.
posts=#

```

UDACITYMy ProgramsDiscoverCatalog

SQL Terminal Workspace

File Edit View Run Kernel Tabs Settings Help

Guide

Project: Guided

Click the below button to make sure the postgres service is started.

START POSTGRES

Then, click the below button to add the necessary schema to the postgres database for the project.

INIT DATABASE

You can then use the `psql` command to enter the database.

Remember, the data in postgres will be reset after 15 minutes of inactivity (including the workspace being closed). You may consider using the `scratchpad.sql` file on the tab at the top of the terminal to store your work long-term.

Template

The template for the project is not included here

Terminal 1

scratchpad.sql

```
postgres=# ;
CREATE TABLE
postgres=# CREATE INDEX score_post ON votes (vote);
CREATE INDEX
postgres=# INSERT INTO users (username)
postgres=# SELECT DISTINCT username
postgres=# FROM bad_posts;
INSERT 0 100
postgres=# SELECT * FROM users;
username_id | username
-----
1 | Raulyn25
2 | Verdie_Heathcote
3 | Jacinto78
4 | Sanny_Morar
5 | Jeffrey_Green
6 | Margarette_Kerluke42
7 | Johnny51
8 | Rickey_Oconnor48
9 | Vidal_Armstrong89
10 | Matilde08
11 | Melba_Stanon
12 | Cathy_Schulist
13 | Rolando36
14 | Domingo_Ratke
15 | Humberto_Heidenreich46
16 | Glendon3
17 | Casper_Durgan1
18 | Ignacio_Heathcote
19 | Tawana_Dougherty
20 | Fred12
21 | Rosaline_Goodwin
22 | Gertrude_Nicholas48
23 | Matilde_Bogan
24 | Jayden_Bowen
25 | Eleazar_Gislason
26 | Estrella_Boehm
27 | Jessy18
28 | Bettie_Boyer
29 | Melissa_Heimann55
```

737809
-1.03%

Buscar

ESP LAA 9:17 p. m. 17/12/2023

UDACITYMy ProgramsDiscoverCatalog

SQL Terminal Workspace

File Edit View Run Kernel Tabs Settings Help

Guide

Project: Guided

Click the below button to make sure the postgres service is started.

START POSTGRES

Then, click the below button to add the necessary schema to the postgres database for the project.

INIT DATABASE

You can then use the `psql` command to enter the database.

Remember, the data in postgres will be reset after 15 minutes of inactivity (including the workspace being closed). You may consider using the `scratchpad.sql` file on the tab at the top of the terminal to store your work long-term.

Template

The template for the project is not included here

Terminal 1

scratchpad.sql

```
70 | Gus_Komer
71 | Paris1
72 | Yasmine_Lehner
73 | Monica_Meher85
74 | Rhyna_Fourus
75 | Larue_Sayer22
76 | Carissa54
77 | Relisab39
78 | Melba_Vost82
79 | Tia_Bocco
80 | Chesley_Will122
81 | Gus32
82 | Verda_Jaskolski
83 | Luc45
84 | Vesta_Schlerin
85 | Ricardo72
86 | Mallie76
87 | Cyril_Heller23
88 | Lee_Kings3
89 | Odell_Rath77
90 | Alka_Satterfield87
91 | Fra_Willias
92 | Jaeger_Powlowski197
93 | Rebecca44
94 | Carmella_Olson
95 | Valda_McKenzie
96 | Edna9
97 | Sennie_Bednar44
98 | Chris_Ferry
99 | Jared87
100 | Aurelie16

(100 rows)

postgres=#
postgres=#
postgres=#
postgres=#
postgres=#
```

13°C
Cerca del récord

Buscar

ESP LAA 9:18 p. m. 17/12/2023



SQL Terminal Workspace

[illegible]

SQL Terminal Workspace

[illegible]

UDACITYMy ProgramsDiscoverCatalog

SQL Terminal Workspace

File Edit View Run Kernel Tabs Settings Help

Guide

Project: Database

Click the below button to make sure the postgres service is started.

START POSTGRES

Then, click the below button to add the necessary schema to the postgres database for the project.

INIT DATABASE

You can then use the `psql` command to enter the database.

Remember, the data in postgres will be reset after 15 minutes of inactivity (including the workspace being closed). You may consider using the `scratchpad.sql` file on the tab at the top of the terminal to store your work long-term.

Template

The template for the project is not included here

Terminal 1

scratchpad.sql

```
ANALYZE CREATE EXECUTE LOCK RESET START
BEGIN DEALLOCATE EXPLAIN MOVE REVOKE TABLE
CHECKPOINT DECLARE FETCH NOTIFY ROLLBACK TRUNCATE
CLOSE DELETE FROM GRANT PREPARE UNLISTEN
CLUSTER DISCARD IMPORT REASSIGN SECURITY LABEL UPDATE
COMMENT DO INSERT REFRESH SELECT VACUUM

postgres=# FROM bad_posts;
postgres=# SELECT DISTINCT upvote
postgres=# FROM tables;
postgres=# LEFT JOIN users u
postgres=# ON table1.upvote = u.username
postgres=# WHERE u.username IS NULL;
INSERT 0 994
postgres=# INSERT INTO users (username)
postgres=# WITH tables AS (SELECT REGEXP_SPLIT_TO_TABLE(downvotes, ','))
postgres=#
postgres=# COMMIT DROP LISTEN REINDEX SET VALUES
ALTER COPY END LOAD RELEASE SHOW WITH
ANALYZE CREATE EXECUTE LOCK RESET START
BEGIN DEALLOCATE EXPLAIN MOVE REVOKE TABLE
CHECKPOINT DECLARE FETCH NOTIFY ROLLBACK TRUNCATE
CLOSE DELETE FROM GRANT PREPARE UNLISTEN
CLUSTER DISCARD IMPORT REASSIGN SECURITY LABEL UPDATE
COMMENT DO INSERT REFRESH SELECT VACUUM

postgres=# COMMIT DROP LISTEN REINDEX SET VALUES
ALTER COPY END LOAD RELEASE SHOW WITH
ANALYZE CREATE EXECUTE LOCK RESET START
BEGIN DEALLOCATE EXPLAIN MOVE REVOKE TABLE
CHECKPOINT DECLARE FETCH NOTIFY ROLLBACK TRUNCATE
CLOSE DELETE FROM GRANT PREPARE UNLISTEN
CLUSTER DISCARD IMPORT REASSIGN SECURITY LABEL UPDATE
COMMENT DO INSERT REFRESH SELECT VACUUM
```

13°C Cerca del récord

Buscar

ESP LAA 9:19 p. m. 17/12/2023

UDACITYMy ProgramsDiscoverCatalog

SQL Terminal Workspace

File Edit View Run Kernel Tabs Settings Help

Guide

Project: Database

Click the below button to make sure the postgres service is started.

START POSTGRES

Then, click the below button to add the necessary schema to the postgres database for the project.

INIT DATABASE

You can then use the `psql` command to enter the database.

Remember, the data in postgres will be reset after 15 minutes of inactivity (including the workspace being closed). You may consider using the `scratchpad.sql` file on the tab at the top of the terminal to store your work long-term.

Template

The template for the project is not included here

Terminal 1

scratchpad.sql

```
COMMENT DO INSERT REFRESH SELECT VACUUM
postgres=# COMMIT DROP LISTEN REINDEX SET VALUES
ALTER COPY END LOAD RELEASE SHOW WITH
ANALYZE CREATE EXECUTE LOCK RESET START
BEGIN DEALLOCATE EXPLAIN MOVE REVOKE TABLE
CHECKPOINT DECLARE FETCH NOTIFY ROLLBACK TRUNCATE
CLOSE DELETE FROM GRANT PREPARE UNLISTEN
CLUSTER DISCARD IMPORT REASSIGN SECURITY LABEL UPDATE
COMMENT DO INSERT REFRESH SELECT VACUUM

postgres=# LEFT(bp.title,100), bp.url, bp.text_content
postgres=# FROM bad_posts bp
postgres=# JOIN topics t
postgres=# ON bp.topic = t.topic
postgres=# JOIN users u
postgres=# ON bp.username = u.username;
INSERT 0 50000
postgres=# -- Inserting data into comments database
postgres=# INSERT INTO comments (user_id, post_id, text_content, level)
postgres=# SELECT u.username, id, p.id, bc.text_content,
postgres=# ROW_NUMBER() OVER(PARTITION BY p.id)
postgres=# FROM bad_comments bc
postgres=# JOIN posts p
postgres=# ON bc.post_id = p.id
postgres=# JOIN users u
postgres=# ON bc.username = u.username;
INSERT 0 100000
postgres=# -- Inserting data into votes database
postgres=# INSERT INTO votes (user_id, post_id, vote)
postgres=# WITH tables AS (SELECT id, REGEXP_SPLIT_TO_TABLE(downvotes, ','))
postgres=#
postgres=# COMMIT DROP LISTEN REINDEX SET VALUES
ALTER COPY END LOAD RELEASE SHOW WITH
ANALYZE CREATE EXECUTE LOCK RESET START
BEGIN DEALLOCATE EXPLAIN MOVE REVOKE TABLE
CHECKPOINT DECLARE FETCH NOTIFY ROLLBACK TRUNCATE
CLOSE DELETE FROM GRANT PREPARE UNLISTEN
CLUSTER DISCARD IMPORT REASSIGN SECURITY LABEL UPDATE
```

13°C Mayorm. nublado

Buscar

ESP LAA 9:19 p. m. 17/12/2023

UDACITYMy ProgramsDiscoverCatalog

SQL Terminal Workspace

File Edit View Run Kernel Tabs Settings Help

Guide

Project: Guided

Click the below button to make sure the postgres service is started.

START POSTGRES

Then, click the below button to add the necessary schema to the postgres database for the project.

INIT DATABASE

You can then use the `psql` command to enter the database.

Remember, the data in postgres will be reset after 15 minutes of inactivity (including the workspace being closed). You may consider using the `scratchpad.sql` file on the tab at the top of the terminal to store your work long-term.

Template

The template for the project is not included here

Terminal 1

scratchpad.sql

```
ALTER COPY END LOAD RELEASE SHOW WITH
ANALYZE CREATE EXECUTE LOCK RESET START
BEGIN DEALLOCATE EXPLAIN MOVE REVOKE TABLE
CHECKPOINT DECLARE FETCH NOTIFY ROLLBACK TRUNCATE
CLOSE DELETE FROM GRANT PREPARE SAVEPOINT UNLISTEN
COMMENT DISCARD IMPORT REASSIGN SECURITY LABEL UPDATE
DO INSERT REFRESH SELECT VACUUM

postgres=# FROM bad_posts)
postgres=# SELECT u.username_id, table1.id, -1 AS vote
postgres=# FROM table1
postgres=# JOIN users u
postgres=# ON u.username = table1.downvote;

INSERT @ 249911
postgres=#
postgres=# INSERT INTO votes (user_id, post_id, vote)
postgres=# WITH table1 AS (SELECT id, response_split_20 TABLE(upvotes, ''')
postgres=#
postgres=#
ABORT COMMIT DROP LISTEN REINDEX SET VALUES
ALTER COPY END LOAD RELEASE SHOW WITH
ANALYZE CREATE EXECUTE LOCK RESET START
BEGIN DEALLOCATE EXPLAIN MOVE REVOKE TABLE
CHECKPOINT DECLARE FETCH NOTIFY ROLLBACK TRUNCATE
CLOSE DELETE FROM GRANT PREPARE SAVEPOINT UNLISTEN
COMMENT DISCARD IMPORT REASSIGN SECURITY LABEL UPDATE
DO INSERT REFRESH SELECT VACUUM

postgres=#
ABORT COMMIT DROP LISTEN REINDEX SET VALUES
ALTER COPY END LOAD RELEASE SHOW WITH
ANALYZE CREATE EXECUTE LOCK RESET START
BEGIN DEALLOCATE EXPLAIN MOVE REVOKE TABLE
CHECKPOINT DECLARE FETCH NOTIFY ROLLBACK TRUNCATE
CLOSE DELETE FROM GRANT PREPARE SAVEPOINT UNLISTEN
COMMENT DISCARD IMPORT REASSIGN SECURITY LABEL UPDATE
DO INSERT REFRESH SELECT VACUUM

postgres=#
ABORT COMMIT DROP LISTEN REINDEX SET VALUES
ALTER COPY END LOAD RELEASE SHOW WITH
ANALYZE CREATE EXECUTE LOCK RESET START
```

13°C

Mayorm. nublado

Buscar

ESP LAA

9:20 p. m. 17/12/2023

UDACITYMy ProgramsDiscoverCatalog

SQL Terminal Workspace

File Edit View Run Kernel Tabs Settings Help

Guide

Project: Guided

Click the below button to make sure the postgres service is started.

START POSTGRES

Then, click the below button to add the necessary schema to the postgres database for the project.

INIT DATABASE

You can then use the `psql` command to enter the database.

Remember, the data in postgres will be reset after 15 minutes of inactivity (including the workspace being closed). You may consider using the `scratchpad.sql` file on the tab at the top of the terminal to store your work long-term.

Template

The template for the project is not included here

Terminal 1

scratchpad.sql

```
CLUSTER DISCARD IMPORT REASSIGN SECURITY LABEL UPDATE
COMMENT DO INSERT REFRESH SELECT VACUUM

postgres=#
postgres=#
ABORT COMMIT DROP LISTEN REINDEX SET VALUES
ALTER COPY END LOAD RELEASE SHOW WITH
ANALYZE CREATE EXECUTE LOCK RESET START
BEGIN DEALLOCATE EXPLAIN MOVE REVOKE TABLE
CHECKPOINT DECLARE FETCH NOTIFY ROLLBACK TRUNCATE
CLOSE DELETE FROM GRANT PREPARE SAVEPOINT UNLISTEN
COMMENT DISCARD IMPORT REASSIGN SECURITY LABEL UPDATE
DO INSERT REFRESH SELECT VACUUM

postgres=#
ABORT COMMIT DROP LISTEN REINDEX SET VALUES
ALTER COPY END LOAD RELEASE SHOW WITH
ANALYZE CREATE EXECUTE LOCK RESET START
BEGIN DEALLOCATE EXPLAIN MOVE REVOKE TABLE
CHECKPOINT DECLARE FETCH NOTIFY ROLLBACK TRUNCATE
CLOSE DELETE FROM GRANT PREPARE SAVEPOINT UNLISTEN
COMMENT DISCARD IMPORT REASSIGN SECURITY LABEL UPDATE
DO INSERT REFRESH SELECT VACUUM

postgres=# FROM bad_posts)
postgres=# SELECT u.username_id, table1.id, 1 AS vote
postgres=# FROM table1
postgres=# JOIN users u
postgres=# ON u.username = table1.upvote;

INSERT @ 249799
postgres=#
postgres=#
postgres=# ^C
postgres=# []
```

13°C

Mayorm. nublado

Buscar

ESP LAA

9:22 p. m. 17/12/2023

UDACITYMy ProgramsDiscoverCatalog

SQL Terminal Workspace

File Edit View Run Kernel Tabs Settings Help

Guide

Project: Udiddit

Click the below button to make sure the postgres service is started.

START POSTGRES

Then, click the below button to add the necessary schema to the postgres database for the project.

INIT DATABASE

You can then use the `psql` command to enter the database.

Remember, the data in postgres will be reset after 15 minutes of inactivity (including the workspace being closed). You may consider using the `scratchpad.sql` file on the tab at the top of the terminal to store your work long-term.

Template

The template for the project is not included here

Terminal 1

scratchpad.sql

```
BEGIN;
CHECKPOINT;
CLOSE;
CLUSTER;
COMMENT;
ABORT;
ALTER;
ANALYZE;
BEGIN;
CHECKPOINT;
CLOSE;
CLUSTER;
COMMENT;
DEALLOCATE;
DELETE FROM;
IMPORT;
INSERT;
COMMIT;
COPY;
CREATE;
EXECUTE;
EXPLAIN;
FETCH;
GRANT;
IMPORT;
INSERT;
REFRESH;
LISTEN;
LOAD;
LOCK;
MOVE;
NOTIFY;
PREPARE;
REASSIGN;
REFRESH;
REINDEX;
RELEASE;
RESET;
REVOKE;
ROLLBACK;
SAVEPOINT;
SECURITY LABEL;
SELECT;
TABLE;
TRUNCATE;
UNLISTEN;
UPDATE;
VACUUM;
SET;
SHOW;
WITH;
VALUES;
```

13°C Mayorm. nublado

Buscar

ESP LAA

9:23 p. m. 17/12/2023

UDACITYMy ProgramsDiscoverCatalog

SQL Terminal Workspace

File Edit View Run Kernel Tabs Settings Help

Guide

Project: Udiddit

Click the below button to make sure the postgres service is started.

START POSTGRES

Then, click the below button to add the necessary schema to the postgres database for the project.

INIT DATABASE

You can then use the `psql` command to enter the database.

Remember, the data in postgres will be reset after 15 minutes of inactivity (including the workspace being closed). You may consider using the `scratchpad.sql` file on the tab at the top of the terminal to store your work long-term.

Terminal 1

scratchpad.sql

Migration.sql

```
postgres=# FROM "bad_posts_downvotes" bpd
postgres=#   JOIN "posts" po
postgres=#   ON bpd.title = po.title
postgres=#   JOIN "users" uu
postgres=#   ON bpd.username_downvotes = uu.username;
ERROR: relation "bad_posts_downvotes" does not exist
LINE 4: FROM "bad_posts_downvotes" bpd
          ^
postgres=# INSERT INTO "post_votes"
postgres=#   ("post_vote",
postgres=#   "voter_user_id",
postgres=#   "post_id")
postgres=# WITH "bad_posts_upvotes"
postgres=# AS (SELECT title,
postgres=#   "bad_posts_downvotes" bpd,
postgres=#   "bad_posts_downvotes"
postgres=#   FROM "bad_posts" bp)
postgres=#   Regexp_split_to_table(bp.downvotes, ',') AS username_downvotes
postgres=#   FROM "bad_posts" bp) SELECT 1 AS post_vote,
postgres=#   uu.id AS voter_user_id,
postgres=#   po.id AS post_id
postgres=# FROM "bad_posts_upvotes" bpu
postgres=#   JOIN "posts" po
postgres=#   ON bpu.title = po.title
postgres=#   JOIN "users" uu
postgres=#   ON bpu.username_upvotes = uu.username
postgres=# UNION ALL
postgres=# SELECT -1 AS post_vote,
postgres=#   uu.id AS voter_user_id,
postgres=#   po.id AS post_id
postgres=# FROM "bad_posts_downvotes" bpd
postgres=#   JOIN "posts" po
postgres=#   ON bpd.title = po.title
postgres=#   JOIN "users" uu
postgres=#   ON bpd.username_downvotes = uu.username;
ERROR: relation "post_votes" does not exist
LINE 1: INSERT INTO "post_votes"
          ^
postgres=#
```

13°C Mayorm. nublado

Buscar

ESP LAA

9:27 p. m. 17/12/2023