SQL Used in New Social Media

In addition to the basic functions PostGreSQL has included in the database, the following SQL code has been added into the database:

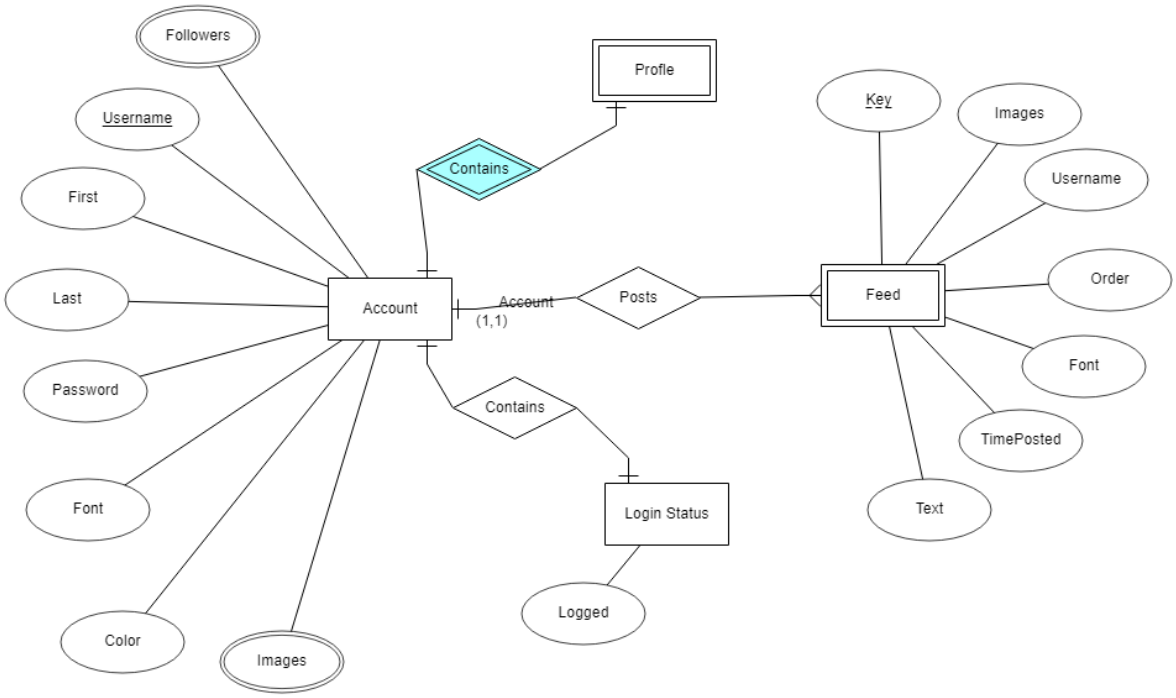
```
CREATE DATABASE postgres
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'en_US.UTF-8'
    LC_CTYPE = 'en_US.UTF-8'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;

COMMENT ON DATABASE postgres
    IS 'default administrative connection database';
```

This is to create database Postgres

Please do note that the backend code does not use SQL code but the python metadata that translates the metadata into SQL code for the database to use.

Current ER diagram:



Tables that are used

Accounts Table

```
CREATE TABLE public."Accounts"
(
    "Username" character varying COLLATE pg_catalog."default" NOT NULL,
    "First" character varying COLLATE pg_catalog."default",
    "Last" character varying COLLATE pg_catalog."default",
    "Password" character varying COLLATE pg_catalog."default",
    "Images" character varying COLLATE pg_catalog."default",
    "Followed" character varying COLLATE pg_catalog."default",
    "Font" character varying COLLATE pg_catalog."default",
    "Color" character varying COLLATE pg_catalog."default",
    CONSTRAINT "Accounts_pkey" PRIMARY KEY ("Username")
)

TABLESPACE pg_default;

ALTER TABLE public."Accounts"
    OWNER to postgres;
```

This table is designed to store all the accounts created from the frontend:

Column	Constraints	Data Type	Purpose
Username	Primary Key Cannot be NULL	Character varying	Username contains the login info a user has.
First	-	Character varying	Contains first name of the user specifically
Last	-	Character varying	Contains the last name of the user specifically
Password	-	Character varying	Password contains the authentication to make sure the input is actually the user's username.
Images	-	Character varying	Images contains the URL of the picture

Followed	-	Character varying	Followed contains the names of other users followed. It ideally stores the Usernames of the users followed.
Font	-	Character varying	Font stores the type of font the user wants to use
Color	-	Character varying	Color stores the type of color the user wants to use

Improvements:

- All fields should be character varying with size limitations:
    - Username should have a limit of 15 characters.
    - First and Last should have a limit of 20 characters.
    - Password should have a limit of 15 characters.
    - Font and Color should be a char data type.
- Images and Followed should be an array.
- Color field should be in hex.
- Followed field should be a separate relationship table consisting of two usernames columns: follower and followed:

```
CREATE TABLE public.”Followers”
{
        “Follower” character varying (15) NOT NULL
        “Followed” character varying (15) NOT NULL
        CONSTRAINT "Feed.fk" FOREIGN KEY ("Follower")
        REFERENCES public."Accounts" ("Username") MATCH SIMPLE
                ON UPDATE NO ACTION
                ON DELETE NO ACTION
                NOT VALID
        CONSTRAINT "Feed.fk" FOREIGN KEY ("Followed")
        REFERENCES public."Accounts" ("Username") MATCH SIMPLE
                ON UPDATE NO ACTION
                ON DELETE NO ACTION
                NOT VALID
}
```

Profile Table

```
CREATE TABLE public."Profile"
(
    "Username" character varying COLLATE pg_catalog."default" NOT NULL,
    "First" character varying COLLATE pg_catalog."default",
    "Last" character varying COLLATE pg_catalog."default",
    "Font" character varying COLLATE pg_catalog."default",
    "Color" character varying COLLATE pg_catalog."default",
    "Followed" character varying COLLATE pg_catalog."default",
    "Images" character varying COLLATE pg_catalog."default",
    CONSTRAINT "Profile_pkey" PRIMARY KEY ("Username")
)

TABLESPACE pg_default;

ALTER TABLE public."Profile"
    OWNER to postgres;
```

This table is designed to store all the information of profiles. All fields in this table store exactly the same information as the Accounts Table.

Improvements:

- The profile table should be removed. This table's function can be used as a View instead.
- The changes made in the Accounts Table should be implemented in this table:

CREATE VIEW public."Profile" AS
    SELECT Username, First, Last, Font, Color, Followed, Images
    FROM Accounts

SELECT FROM * public."Profile"

Feed Table

```
CREATE TABLE public."Feed"
(
    "Images" character varying COLLATE pg_catalog."default",
    "Username" character varying COLLATE pg_catalog."default" NOT NULL,
    "Order" integer,
    "Font" character varying COLLATE pg_catalog."default",
    "TimePosted" time without time zone,
    "Text" character varying COLLATE pg_catalog."default",
    CONSTRAINT "Feed_pkey" PRIMARY KEY ("Username")
)

TABLESPACE pg_default;

ALTER TABLE public."Feed"
    OWNER to postgres;
```

This table is designed to store all user posts.

Column	Constraints	Data Type	Purpose
Images	-	Character varying	Carries the URL of the image
Username	Not NULL Primary Key	Character varying	Carries the login info of the user in the same field as the field Username in Profile Table.
Order	-	integer	Does Nothing
Font	-	Character varying	Font stores the type of font the user wants to use. It's the exact same info as the field Font in the Profile Table.
TimePosted	-	Time without time zone	Stores the time that the post was made in local time the database is stored in.
Text	-	Character varying	Text stores the type of font the user wants to use. It's the exact same info as the field Text in the Profile Table.

Improvements:

- This table will not serve at its intended purpose because Username is a Primary key. Since Primary keys cannot be duplicated, each account can only make one post and one post only.
  SOLUTION: Order should be the Primary Key that auto increments each time a post is made. Username cannot be used as a Primary Key.
- Username should be set as a Foreign key.
- TimePosted should be time with time zone.
- Font and Text should be Foreign Keys.
- There should be limits on the following fields:
  - Text and Font should be character data type
  - Username should have a limit of 15 characters

Potential Improvement:

```
CREATE TABLE public."Feed"
(
    "Images" character varying COLLATE pg_catalog."default",
    "Username" character varying COLLATE pg_catalog."default" NOT NULL,
    "Order" integer,
    "Font" character varying COLLATE pg_catalog."default",
    "TimePosted" time without time zone,
    "Text" character varying COLLATE pg_catalog."default",
    "Key" integer NOT NULL DEFAULT nextval('"Feed_Key_seq"'::regclass),
    CONSTRAINT "Feed_pk" PRIMARY KEY ("Key"),
    CONSTRAINT "Feed.fk" FOREIGN KEY ("Username")
        REFERENCES public."Accounts" ("Username") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public."Feed"
    OWNER to postgres;
```

LoginStatus Table

```
CREATE TABLE public."LoginStatus"
(
    "Status" boolean,
    "Username" character varying COLLATE pg_catalog."default"
)

TABLESPACE pg_default;

ALTER TABLE public."LoginStatus"
    OWNER to postgres;
```

The table is designed to store login info of the user

Column	Constraints	Data Type	Purpose
Status	-	Boolean	Status is a boolean of the user's login status. True: The user is logged in False: The user isn't logged in
Username	-	Character varying	Username contains the user's username

Improvements:

- Should store possible user statuses such with an identifying ID:
    - Logged in
    - Logged out
    - Banned
    - Disabled
- Statuses of the User should be added to the Accounts Table.

Queries that are used

Updating username

```
UPDATE LoginStatus
SET USERNAME = “update value”
WHERE STATUS = True
```

Query where the username is changed when a user is logged in. Potential error as if multiple users log in, their usernames will be changed to the same username.

--

Searching username

```
SELECT Accounts
WHERE USERNAME = “searched name”
```

Query where the username is searched

--

Searching Login Status

```
SELECT LoginStatus
WHERE STATUS
```

Query where login status is searched.

--

Updating Login Status

```
UPDATE LoginStatus
SET STATUS = False, USERNAME = NULL
WHERE STATUS = True
```

Query where login status is changed to false and username is erased. Same potential error as updating username.

--

Return Color

```
SELECT Accounts
WHERE COLOR
```

Query where color is searched
--

Insert User
INSERT INTO Accounts(USERNAME, PASSWORD)
VAULES(‘user’, ‘encrypted’)

Query where a user is created.

--

Insert User with Followed and Color
INSERT INTO Accounts(USERNAME, PASSWORD, FOLLOWED, COLOR)
VAULES(‘user’, ‘encrypted’, ‘another user’, ‘hue’)

Same as Insert User query but also inserting followed and color

--

Insert Feed
INSERT INTO Feed(USERNAME, TEXT)
VALUES(‘user’, ‘message’)

Query where a post is created

--

Delete User
DELETE FROM Accounts
WHERE USERNAME = “chosen user”

Query where a user is deleted

--

Return Entry
SELECT ‘Table name’
WHERE USERNAME = ‘searched username’

Query where it returns everything about the username.

Queries that have been combined to be a general statement

Update Entry

UPDATE ‘Table name’
SET ‘Column name’ = ‘new value’
WHERE USERNAME = ‘searched username’

Supposed statement that takes a table name, column name, and a new value and places the new value at the listed table name and column name. This query provides a security risk where SQL injection could affect how this function can act and crash the database.