Emily Greven and Swastik Satyam

Assumptions:

- One-way friendship relation (like followers on instagram)
- Photo must be in an album, photo is in only one album (total participation)
- Each comment must have a user and each comment is written by one user only (total participation)
- Each photo can have multiple tags and each tag can be on multiple photos (many to many)
- Each photo can have multiple comments but each comment can only be on one photo (total participation.
- Each user can have multiple albums but each album can only have one user (total participation)

New Assumptions:

- Email is unique for each user
- Album name is unique for each album
- Added Album_name to Photos table to allow for easier queries
- User_id not unique in Comments to allow anonymous users to leave comments
- Only users that own their photos and albums can delete those albums and photos
- Comments are not null

Limitations:

- Anonymous users are unable to like photos due to limitations in the data table “Likes” since the primary key requires user_id to be not null
- Users cannot delete their own profiles
- No authentication of email addresses or ability to reset passwords

- Not easily scalable due to the large amount of data stored for each photo

Updated Schema based on example solution given:

```
CREATE DATABASE IF NOT EXISTS photoshare;
USE photoshare;


DROP TABLE IF EXISTS Friends CASCADE;
DROP TABLE IF EXISTS Tagged CASCADE;
DROP TABLE IF EXISTS Comments CASCADE;
DROP TABLE IF EXISTS Likes CASCADE;


DROP TABLE IF EXISTS Tags CASCADE;
DROP TABLE IF EXISTS Photos CASCADE;
DROP TABLE IF EXISTS Albums CASCADE;
DROP TABLE IF EXISTS Users CASCADE;

CREATE TABLE Users(
user_id INTEGER AUTO_INCREMENT,
first_name VARCHAR(100) NOT NULL,
last_name VARCHAR(100) NOT NULL,
email VARCHAR(100) UNIQUE NOT NULL,
birth_date DATE,
hometown VARCHAR(100),
gender VARCHAR(100),
password VARCHAR(100) NOT NULL,
PRIMARY KEY (user_id)
);

CREATE TABLE Friends(
user_id1 INTEGER,
user_id2 INTEGER,
PRIMARY KEY (user_id1, user_id2),
FOREIGN KEY (user_id1)
REFERENCES Users(user_id),
FOREIGN KEY (user_id2)
REFERENCES Users(user_id)
);

CREATE TABLE Albums(
albums_id INTEGER AUTO_INCREMENT,
```

```sql
album_name VARCHAR(100) UNIQUE,
date DATE,
user_id INTEGER NOT NULL,
PRIMARY KEY (albums_id),
FOREIGN KEY (user_id)
REFERENCES Users(user_id)
);

CREATE TABLE Tags(
tag_id INTEGER AUTO_INCREMENT,
name VARCHAR(100),
PRIMARY KEY (tag_id)
);

CREATE TABLE Photos(
photo_id INTEGER AUTO_INCREMENT,
caption VARCHAR(100),
data LONGBLOB,
albums_id INTEGER NOT NULL,
album_name VARCHAR(100),
user_id INTEGER NOT NULL,
PRIMARY KEY (photo_id),
FOREIGN KEY (albums_id) REFERENCES Albums (albums_id) ON DELETE CASCADE,
FOREIGN KEY (user_id) REFERENCES Users (user_id)
);

CREATE TABLE Tagged(
photo_id INTEGER,
tag_id INTEGER,
PRIMARY KEY (photo_id, tag_id),
FOREIGN KEY(photo_id)
REFERENCES Photos (photo_id),
FOREIGN KEY(tag_id)
REFERENCES Tags (tag_id)
);

CREATE TABLE Comments(
comment_id INTEGER AUTO_INCREMENT,
user_id INTEGER,
photo_id INTEGER NOT NULL,
text VARCHAR (100),
date DATE,
PRIMARY KEY (comment_id),
FOREIGN KEY (user_id)
REFERENCES Users (user_id),
FOREIGN KEY (photo_id)
REFERENCES Photos (photo_id)
);

CREATE TABLE Likes(
photo_id INTEGER,
user_id INTEGER,
PRIMARY KEY (photo_id,user_id),
FOREIGN KEY (photo_id)
REFERENCES Photos (photo_id),
FOREIGN KEY (user_id)
REFERENCES Users (user_id)
);
```