We add a “cpoint” attribute in Users Table, it helps to calculate an users’ contribution point whenever he/she uploads a photo or leaves a comment.

We set user_id, album_id, photo_id, comment_id, and tag_id all to AUTO_INCREASE, so these ids will generate automatically when they are created.

We set Likes, Comments, Tagged to ON DELETE CASCADE so they will be deleted when an associated photo was deleted. Same thing has done to Albums and Photos, photos would be deleted when their Album was deleted.

Constraints like “Users cannot comment on their own photos” cannot be done by using assertion because MySQL doesn’t support it. So we wrote python code to make sure the constraint can still be applied to our project. For example: (details can be found in comment( ) function in app.py)

```
if isUser():
    # cannot comment own photo
    if selfComment(pid):
        flash("Cannot comment on your own photo")
        return redirect(url_for('comment', photo_id=pid))
    else:
        uid = getUserIdFromEmail(flask_login.current_user.id)
        cursor = conn.cursor()
        cursor.execute("INSERT INTO Comments(user_id, photo_id, text, date)\
                        VALUES ({0},{1},'{2}','{3}')".format(uid, pid, comment, today))
        conn.commit()
        addpoint(uid)
```

Another constraint: Anonymous cannot like photos

```
# The person liked the photo
else:
    if isUser():
        uid = getUserIdFromEmail(flask_login.current_user.id)
        cursor = conn.cursor()
        cursor.execute("INSERT INTO Likes(photo_id, user_id) VALUES('{0}','{1}')".format(pid, uid))
        conn.commit()
        flash("You liked this photo!!")
        return redirect(url_for('comment', photo_id=pid))
    else:
        flash("You cannot like this photo as an anonymous user")
        return redirect(url_for('comment', photo_id=pid))
```

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

CREATE TABLE Users(
 user_id INTEGER AUTO_INCREMENT,
 first_name VARCHAR(100) NOT NULL,
 last_name VARCHAR(100) NOT NULL,
 email VARCHAR(100) NOT NULL,
 birth_date DATE NOT NULL,
 hometown VARCHAR(100),
 gender VARCHAR(100),
 cpoint INTEGER,
 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,
  name VARCHAR(100) NOT NULL,
  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,
  word VARCHAR(100),
  PRIMARY KEY (tag_id)
 );
```

```
CREATE TABLE Photos(
 photo_id INTEGER AUTO_INCREMENT,
 caption VARCHAR(100),
 data LONGBLOB,
 albums_id INTEGER NOT NULL,
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)
);
SELECT * FROM Tagged;
CREATE TABLE Tagged(
 photo_id INTEGER,
 tag_id INTEGER,
 PRIMARY KEY (photo_id, tag_id),
 FOREIGN KEY(photo_id)
 REFERENCES Photos (photo_id) ON DELETE CASCADE,
 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) ON DELETE CASCADE
);
```

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