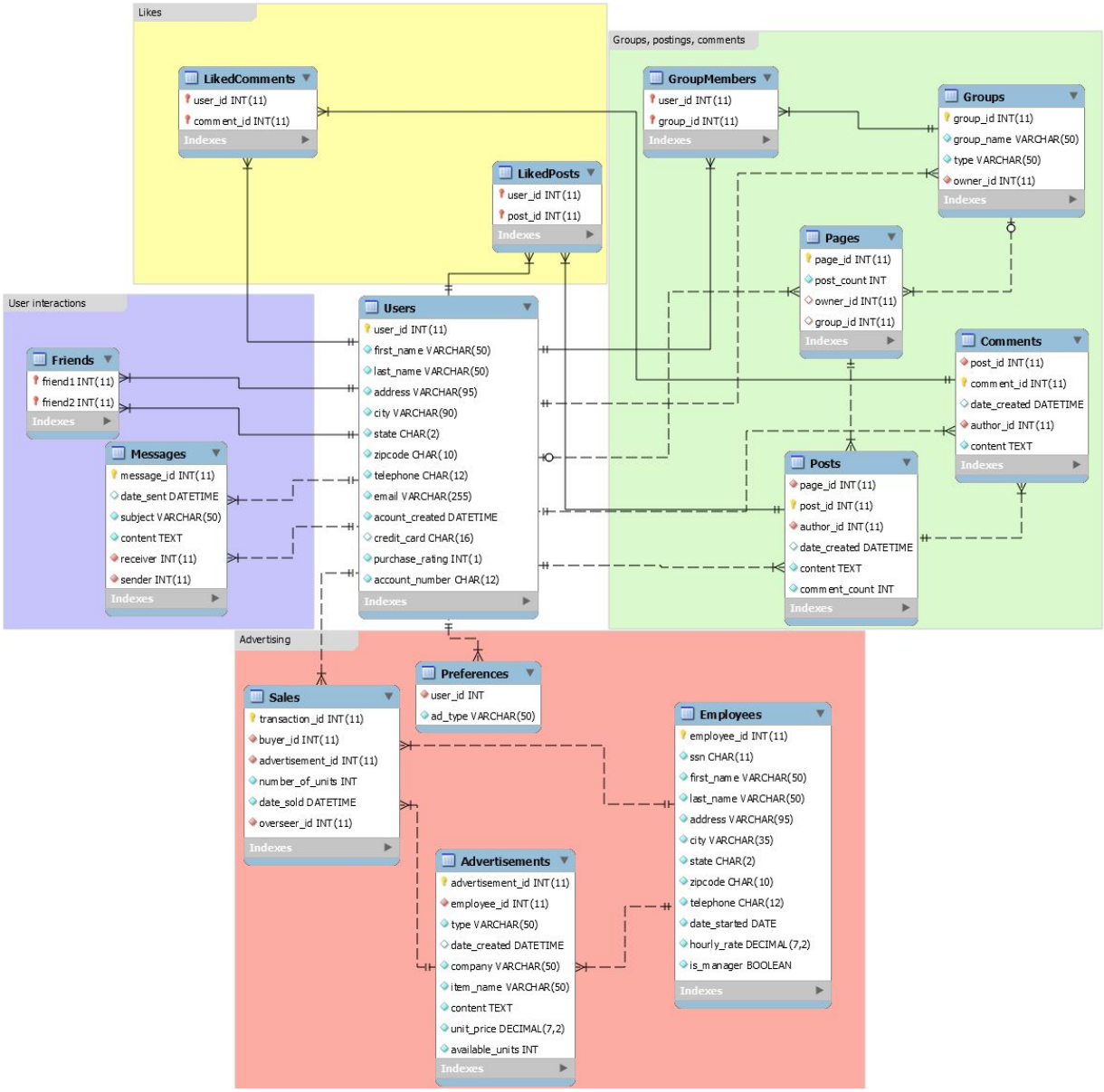
Wolfiebook

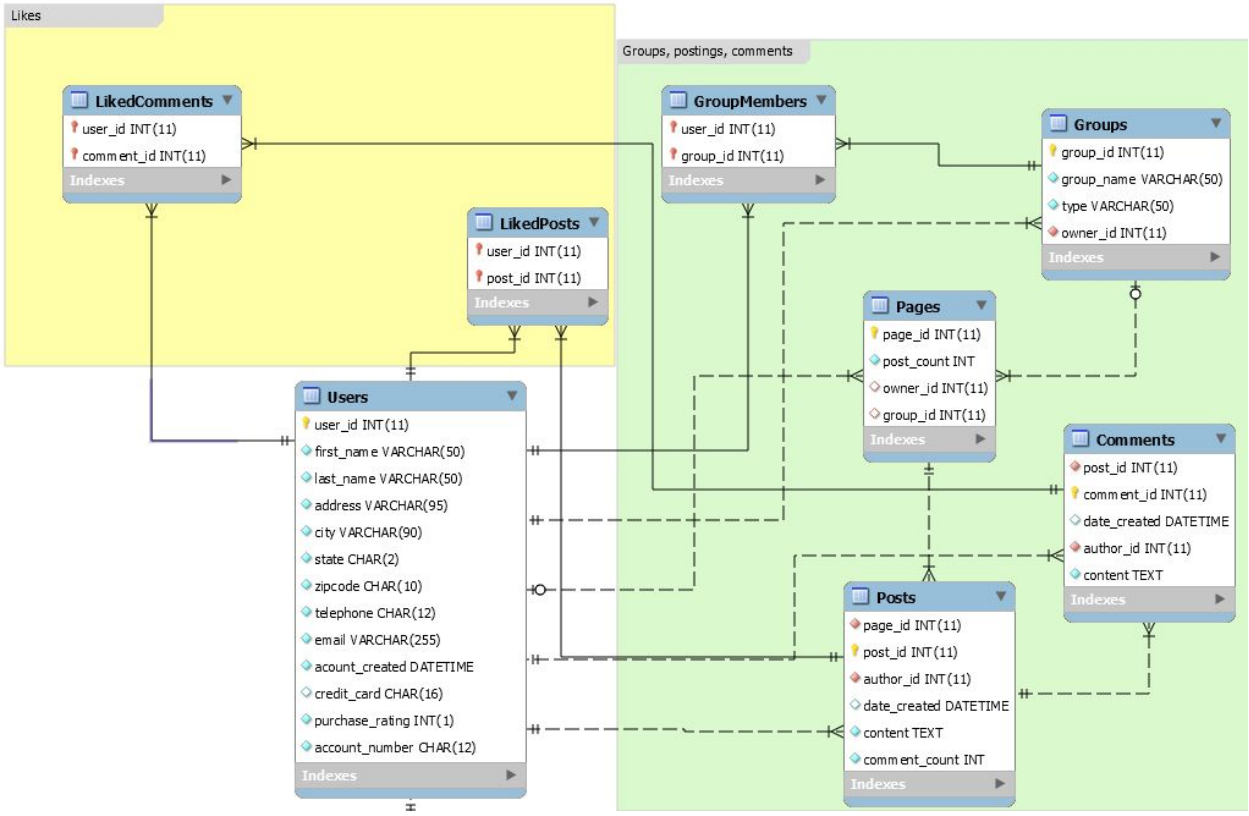
CSE305.01 Fall 2016
Project Assignment 1

Chaerin Kim - Chaerin.Kim@stonybrook.edu
Jeonghoon Kim - Jeonghoon.Kim@stonybrook.edu
Bryan Koelbel - Bryan.Koelbel@stonybrook.edu

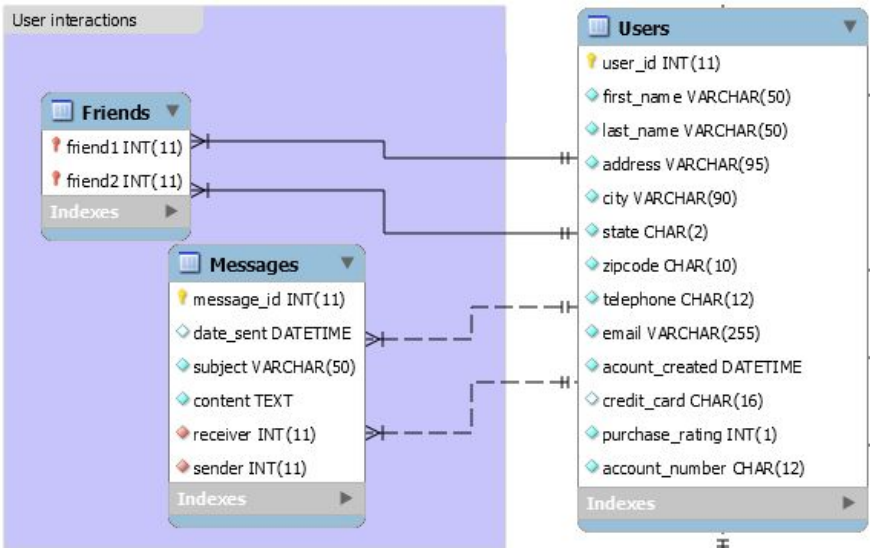
I. ER Diagram



Project Overview



“Groups, postings, comments” and “Likes” isolated



“User interactions” isolated

“Advertising” isolated

Above is the ER Diagram for the design of the database system. User account information is stored in the **Users** relation.

User interactions:

Friendships are handled by the **Friends** relation; there should be exactly one Friends relation for each pair of friends. There can be any number of **Messages** for each **sender** and **receiver**.

Groups, postings, comments:

A User can be the owner (**owner_id**) of any number of **Groups**. Group memberships are stored in the **GroupMembers** relation; a user can be a member of a certain group exactly once, but can join any number of groups. The **Pages** relation handles both User pages (Wall posts) and Group pages. Depending on whether the Page is a group page or a user page, exactly one of the two **owner_id** and **group_id** attributes should be non-null. Each Page can have any number of **Posts**, and a user can write (**author_id**) any number of posts. Each Post can have any number of **Comments**, and a user can write (**author_id**) any number of comments.

Likes:

A User can “like” any number of comments or posts, represented by **LikedPosts** and **LikedComments**. Note that a User may not like the same comment or post multiple times (in other words, both **user_id** and **post_id** are primary keys).

Advertising:

A user can have any number of **Preferences** for advertisements. The **Advertisements** relation represents each ad in the system. An Advertisement may be associated with any number of **Sales**, which represents a unique User purchase (note that a User may purchase the same product from an ad any number of times). Advertisements and Sales are overseen by one **Employee** (an Employee can handle any number of Sales and Advertisements). Note that Managers and Employees are stored in the same table, distinguished by the **is_manager** attribute.

II. Relational Model

```
CREATE TABLE IF NOT EXISTS Users (
	user_id int(11) NOT NULL AUTO_INCREMENT,
	first_name varchar(50) NOT NULL,
	last_name varchar(50) NOT NULL,
	address varchar(95) NOT NULL,
	city varchar(35) NOT NULL,
	state char(2) NOT NULL,
	zipcode varchar(10) NOT NULL,
	telephone char(12) NOT NULL,
	email varchar(255) NOT NULL,
	account_number char(12) NOT NULL,
	account_created datetime DEFAULT CURRENT_TIMESTAMP,
	credit_card char(16), # keep account history
	purchase_rating int(1) NOT NULL, # active status in terms of making purchases
	PRIMARY KEY (user_id)
	#Can connect with other users
	#Can post message on page
	#Can follow up on Wall
	#Can like or comment on post
	#Can create group
	#Can make purchase
);

CREATE TABLE IF NOT EXISTS Preferences (
	user_id int(11) NOT NULL,
	ad_type varchar(50) NOT NULL,
	FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

```
CREATE TABLE IF NOT EXISTS Groups (
	group_id int(11) NOT NULL AUTO_INCREMENT,
	group_name varchar(50) NOT NULL,
	type varchar(50) NOT NULL,
	owner_id int(11) NOT NULL,
	PRIMARY KEY (group_id),
	FOREIGN KEY (owner_id) REFERENCES Users(user_id)
);

CREATE TABLE IF NOT EXISTS Pages (
	page_id int(11) NOT NULL AUTO_INCREMENT,
	owner_id int(11),
	group_id int(11),
	post_count int DEFAULT 0,
	PRIMARY KEY (page_id),
	FOREIGN KEY (owner_id) REFERENCES Users(user_id),
	FOREIGN KEY (group_id) REFERENCES Groups(group_id)
);

CREATE TABLE IF NOT EXISTS Posts (
	page_id int(11) NOT NULL,
	post_id int(11) NOT NULL AUTO_INCREMENT,
	author_id int(11) NOT NULL,
	date_created datetime DEFAULT CURRENT_TIMESTAMP,
	content text,
	comment_count int DEFAULT 0,
	PRIMARY KEY (post_id),
	FOREIGN KEY (page_id) REFERENCES Pages (page_id),
	FOREIGN KEY (author_id) REFERENCES Users (user_id)
);

CREATE TABLE IF NOT EXISTS Comments (
	post_id int(11) NOT NULL,
	comment_id int(11) NOT NULL AUTO_INCREMENT,
	author_id int(11) NOT NULL,
	date_created datetime DEFAULT CURRENT_TIMESTAMP,
	content text NOT NULL,
	PRIMARY KEY (comment_id),
	FOREIGN KEY (author_id) REFERENCES Users(user_id),
	FOREIGN KEY (post_id) REFERENCES Posts (post_id)
);

CREATE TABLE IF NOT EXISTS Messages (
```

```
	message_id int(11) NOT NULL AUTO_INCREMENT,
	date_sent datetime DEFAULT CURRENT_TIMESTAMP,
	subject varchar(50) NOT NULL,
	content text NOT NULL,
	sender int(11) NOT NULL,
	receiver int(11) NOT NULL,
	PRIMARY KEY (message_id),
	FOREIGN KEY (sender) REFERENCES Users(user_id),
	FOREIGN KEY (receiver) REFERENCES Users(user_id)
);

CREATE TABLE IF NOT EXISTS Friends (
	friend1 int(11) NOT NULL,
	friend2 int(11) NOT NULL,
	PRIMARY KEY (friend1, friend2),
	FOREIGN KEY (friend1) REFERENCES Users(user_id),
	FOREIGN KEY (friend2) REFERENCES Users(user_id)
);

CREATE TABLE IF NOT EXISTS LikedPosts (
	user_id int(11) NOT NULL,
	post_id int(11) NOT NULL,
	PRIMARY KEY (user_id, post_id),
	FOREIGN KEY (user_id) REFERENCES Users(user_id),
	FOREIGN KEY (post_id) REFERENCES Posts(post_id)
);

CREATE TABLE IF NOT EXISTS LikedComments (
	user_id int(11) NOT NULL,
	comment_id int(11) NOT NULL,
	PRIMARY KEY (user_id, comment_id),
	FOREIGN KEY (user_id) REFERENCES Users(user_id),
	FOREIGN KEY (comment_id) REFERENCES Comments(comment_id)
);

CREATE TABLE IF NOT EXISTS GroupMembers (
	user_id int(11) NOT NULL,
	group_id int(11) NOT NULL,
	PRIMARY KEY (user_id, group_id),
	FOREIGN KEY (user_id) REFERENCES Users(user_id),
	FOREIGN KEY (group_id) REFERENCES Groups(group_id)
);
```

```
CREATE TABLE IF NOT EXISTS Employees (
	employee_id int(11) NOT NULL,
	ssn char(11) NOT NULL,
	first_name varchar(50) NOT NULL,
	last_name varchar(50) NOT NULL,
	address varchar(95) NOT NULL,
	city varchar(35) NOT NULL,
	state char(2) NOT NULL,
	zipcode varchar(10) NOT NULL,
	telephone char(12) NOT NULL,
	date_started date NOT NULL, # no default exists for date in mysql
	hourly_rate decimal(7,2) NOT NULL,
	is_manager BOOLEAN NOT NULL,
	PRIMARY KEY (employee_id)
);

CREATE TABLE IF NOT EXISTS Advertisements (
	advertisement_id int(11) NOT NULL AUTO_INCREMENT,
	employee_id int(11) NOT NULL,
	type varchar(50) NOT NULL, #(e.g. clothing, computers)
	date_created datetime DEFAULT CURRENT_TIMESTAMP,
	company varchar(50) NOT NULL, #(e.g. Ford, Gap, Google)
	item_name varchar(50) NOT NULL, #(e.g. particular car, article of clothing, smartphone)
	content text NOT NULL,
	unit_price decimal(7,2) NOT NULL,
	available_units int NOT NULL,
	PRIMARY KEY (advertisement_id),
	FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
);

CREATE TABLE IF NOT EXISTS Sales (
	transaction_id int(11) NOT NULL AUTO_INCREMENT,
	buyer_id int(11),
	date_sold datetime DEFAULT CURRENT_TIMESTAMP,
	advertisement_id int(11) NOT NULL,
	number_of_units int NOT NULL,
	overseer_id int(11),
	PRIMARY KEY (transaction_id),
	FOREIGN KEY (advertisement_id) REFERENCES Advertisements(advertisement_id),
	FOREIGN KEY (buyer_id) REFERENCES Users (user_id),
	FOREIGN KEY (overseer_id) REFERENCES Employees (employee_id)
);
```

Above is the MySQL implementation of the CREATE tables for the project, based on the ER Diagram model. Note that some participation constraints, such as each user having exactly one page (their Wall), will be enforced at a later date using triggers or restricted by the software. Also note that AUTO_INCREMENT is used for key IDs, such as user IDs, post IDs, and comment IDs. This will ensure that, for example, each user ID is unique. Constraints for parameters such as password and user id's can be implemented in the backend or UI before reaching the database.

III. Collaboration Plan

- **Chaerin** will focus on groups, pages, postings, and comments.
- **Jeonghoon** will focus on messaging, searching, liking and unliking postings and comments, and group-related privacy issues.
- **Bryan** will focus on all aspects of targeted advertising and sales.

Wolfiebook

CSE305.01 Fall 2016
Project Assignment 2

Chaerin Kim - Chaerin.Kim@stonybrook.edu
Jeonghoon Kim - Jeonghoon.Kim@stonybrook.edu
Bryan Koelbel - Bryan.Koelbel@stonybrook.edu

User-Level Transactions

User-Level - General transactions:

Register

SQL Statement

INSERT INTO users (user_password, first_name, last_name, address, city, state, zipcode, telephone, email, account_created, credit_card, purchase_rating) VALUES (‘?user_password’, ‘?first_name’, ‘?last_name’, ‘?address’, ‘?city’, ‘?state’, ‘?zipcode’, ‘?telephone’, ‘?email’, ‘?account_created’, ‘?credit_card’, ‘?purchase_rating’);

Parameter types/definition

User_id int: id that identifies the user, unique and auto-increment
User_password char(40): hashed password of user
First_name varchar(50): first name of user
last_name varchar(50): last name of user
address varchar(95): address of user
city varchar(35): city user resides in
state char(2): state user resides in
zipcode varchar(10): zipcode user resides in
telephone char(12): phone number of user
email varchar(255): email address of user, unique
account_created datetime: date and time the user was created
credit_card char(16): credit card number of user
purchase_rating int(1): active status in terms of making purchases

Execution

INSERT INTO Users (user_password, first_name, last_name, address, city, state, zipcode, telephone, email, account_created, credit_card, purchase_rating)
VALUES (SHA2(“salt”, "password"), "Bryan", "Koelbel", "123 South Dr", "Stony Brook", "NY", "11790", "631-123-4567", "bryan.koelbel@stonybrook.edu", NOW(), "1947234500008264", 1);

Output

user_id	user_password	first_name	last_name	address	city	state	zipcode	telephone	email	account_created	credit_card	purchase_rating
1	63479ad69a090b258277ec8fba6f99419a2ffb2489815106 57c944ccd1148e97	Bryan	Koelbel	123 South Dr	Stony Brook	NY	11790	631-123-4567	bryan.koelbel@stonybrook.edu	2016-04-27 10:25:32	1947234500008264	1

Sign-in and Sign-out

SQL Statement

Sign-in: SELECT * FROM Users WHERE email = ‘?email’ AND user_password= SHA2(‘salt’, ‘’?user_password’);
Sign-out: Not necessary.

Parameter type/definition

email varchar(255): email address of user, unique
user_password varchar(64): password of user

Execution

SELECT * FROM Users WHERE email=”bryan.koelbel@stonybrook.edu” AND user_password=SHA2(“salt”, “password”);

Output

```
mysql> SELECT * FROM users WHERE email='bryan.koelbel@stonybrook.edu' AND user_password=SHA2('salt', 'password');
+---------+----------------------------------------------------------------+------------+-----------+-----------+-------------+-------+---------+-------------+-------------------------+---------------------+---------------------+---------------+
| user_id | user_password                                                  | first_name | last_name | address   | city        | state | zipcode | telephone   | email                   | account_created     | credit_card         | purchase_rating |
+---------+----------------------------------------------------------------+------------+-----------+-----------+-------------+-------+---------+-------------+-------------------------+---------------------+---------------------+---------------+
|       1 | 63479ad69a090b258277ec8fba6f99419a2ffb240981510657c944ccd1140e97 | Bryan      | Koelbel   | 123 South Dr | Stony Brook | NY    | 11790   | 631-123-4567 | bryan.koelbel@stonybrook.edu | 2016-04-27 10:25:32 | 1947234500000264    |             1 |
+---------+----------------------------------------------------------------+------------+-----------+-----------+-------------+-------+---------+-------------+-------------------------+---------------------+---------------------+---------------+
```

Post messages in their personal pages

SQL Statement

INSERT INTO Posts (page_id, author_id, date_created, content)
SELECT p.page_id, ‘?author_id’, NOW(), ‘?content’
FROM Pages p
WHERE p.owner_id = ‘?author_id’;
UPDATE Pages SET post_count = post_count + 1 WHERE owner_id = ‘?author_id’;

Parameter type/definition

page_id int(11): page id of the personal page
post_id int(11): auto-incremented post id
author_id int(11): author id
date_created datetime: created date
content text: contents of the post
post_count int: post_count parameter from Pages table

Execution

INSERT INTO Posts (page_id, author_id, date_created, content)
SELECT p.page_id, ‘1’, NOW(), ‘Posting on personal page’
FROM Pages p
WHERE p.owner_id = ‘1’;
UPDATE Pages SET post_count = post_count + 1 WHERE owner_id = ‘1’;

Output

SELECT * FROM posts WHERE page_id = 1;

page_id	post_id	author_id	date_created	content	comment_count
1	1	1	2016-09-02 09:25:33	Hello, World!	2
1	11	1	2016-11-04 16:15:39	Posting on personal page	0

SELECT * FROM pages WHERE owner_id = 1;

page_id	owner_id	group_id	post_count
1	1	NULL	2

Send and Receive a message

SQL Statement

INSERT INTO messages (date_sent, subject, content, sender, receiver)
VALUES (NOW(), ‘?subject’, ‘?content’, ‘?sender’, ‘?receiver’);

Parameter type/definition

message_id int(11): auto-incremented message id
date_sent datetime: time when the message is sent
subject varchar(50): subject of the message
content text: content of the message
sender int(11): user_id of the sender
receiver int(11): user_id of the receiver

Execution

INSERT INTO messages (date_sent, subject, content, sender, receiver)
VALUES (NOW(), ‘Hey! You!’, ‘Get off of my cloud’, ‘1’, ‘2’);

Output

message_id	date_sent	subject	content	sender	receiver
11	2016-11-04 16:54:09	Hey! You!	Get off of my cloud	1	2

The message data will be stored only once when the message is sent.

Delete a message

SQL Statement

DELETE FROM messages WHERE message_id = '?message_id';

Parameter type/definition

Same as 'Send and Receive a messages'

Execution

DELETE FROM messages WHERE message_id = '11';

Output

Message with message_id=11 deleted from the table.

User Level - Regard to their own groups:

Create a group

SQL Statement

INSERT INTO Groups(?group_name, ?type, ?owner_id)
VALUES ('?group_name', '?type', '?owner_id');

Parameter type/definition

group_id int(11): id of group
group_name varchar(50): name of group
type varchar(50): type of group (club, organization, etc)
owner_id int(11): id of User who created group

Execution

INSERT INTO book.Groups(group_name, type, owner_id)
VALUES("Test Group", "Club", 1);

Output

group_id	group_name	type	owner_id
1	Test Group	Club	1

Search for a user and add him/her to a group

SQL Statement

```
INSERT INTO GroupMembers(user_id, group_id)
        SELECT u.user_id, '?group_id'
        FROM Users u
        WHERE MATCH(u.first_name, u.last_name)
        AGAINST ('keyword' IN NATURAL LANGUAGE MODE)
        OR (u.email='keyword')
        LIMIT 1;
```

Parameter type/definition

user_id int(11): user associated to group
group_id int(11): group id

Execution

```
INSERT INTO GroupMembers(user_id, group_id)
        SELECT u.user_id, 6
        FROM Users u
        WHERE MATCH(u.first_name, u.last_name)
        AGAINST ('Jeonghoon Kim' IN NATURAL LANGUAGE MODE)
        OR (u.email='Jeonghoon Kim')
        LIMIT 1;
```

Output

user_id	group_id
2	6

Make a post

Same as ‘Post messages in their personal pages’.

SQL Statement

Parameter type/definition

Execution

Output

SQL Statement

```
INSERT INTO Posts (page_id, author_id, date_created, content)
        SELECT p.page_id, ‘?author_id’, NOW(), ‘?content’
        FROM Pages p
        WHERE p.owner_id = ‘?author_id’;
UPDATE Pages SET post_count = post_count + 1 WHERE owner_id = ‘?author_id’;
```

Parameter type/definition

page_id int(11): page id of the personal page
post_id int(11): auto-incremented post id
author_id int(11): author id
date_created datetime: created date
content text: contents of the post
post_count int: post_count parameter from Pages table

Execution

```
INSERT INTO Posts (page_id, author_id, date_created, content)
        SELECT p.page_id, ‘1’, NOW(), ‘Posting on personal page’
        FROM Pages p
        WHERE p.owner_id = ‘1’;
UPDATE Pages SET post_count = post_count + 1 WHERE owner_id = ‘1’;
```

Output

SELECT * FROM posts WHERE page_id = 1;

page_id	post_id	author_id	date_created	content	comment_count
1	1	1	2016-09-02 09:25:33	Hello, World!	2
1	11	1	2016-11-04 16:15:39	Posting on personal page	0

SELECT * FROM pages WHERE owner_id = 1;

page_id	owner_id	group_id	post_count
1	1	NULL	2

Comment on a post (On User or Group Page)

SQL Statement

```
INSERT INTO Comments(post_id, author_id, date_created, content)
VALUES(‘?post_id’, ‘?author_id’, NOW(), ‘?content’);
UPDATE Posts SET comment_count = comment_count + 1 WHERE post_id = ‘?post_id’;
```

Parameter type/definition

post_id int(11) : Id of post
comment_id int(11) : id of comment
author_id int(11) : id of author

date_created datetime : date and time comment was created
content text : content of the comment

Execution

INSERT INTO Comments(post_id, author_id, date_created, content)
VALUES('11', '2', NOW(), 'For the win');
UPDATE Posts SET comment_count = comment_count + 1 WHERE post_id = '11';

Output

SELECT * FROM Comments WHERE comment_id = 11;

post_id	comment_id	author_id	date_created	content
11	11	2	2016-11-04 23:14:39	For the win

SELECT * FROM Posts WHERE post_id = 11;

page_id	post_id	author_id	date_created	content	comment_count
1	11	1	2016-11-04 16:15:39	Posting on personal page	2

Like a post (On User or Group page)

SQL Statement

INSERT IGNORE INTO LikedPosts (user_id, post_id) VALUES('?user_id', '?post_id');

Parameter type/definition

user_id int(11): user associated to post
post_id int(11): post id

Execution

INSERT IGNORE INTO LikedPosts (user_id, post_id) VALUES('2', '1');

Output

user_id	post_id
2	1

Like a comment (On User or Group page)

SQL Statement

INSERT IGNORE INTO LikedComments (user_id, comment_id) VALUES('?user_id', '?comment_id');

Parameter type/definition

user_id int(11): user associated to comment
comment_id int(11): comment id

Execution

INSERT IGNORE INTO LikedComments (user_id, comment_id) VALUES('4', '4');

Output

user_id	comment_id
4	4

Remove a user from a group & Unjoin Group

SQL Statement

DELETE FROM GroupMembers WHERE user_id='?user_id'

Parameter type/definition

user_id int(11) : id of user we want to remove
group_id int(11): group id

Execution

DELETE FROM GroupMembers WHERE user_id=1

Output

The user data deleted from the GroupMembers table.

Remove a post (on user or group page)

SQL Statement

```
DELETE FROM Posts WHERE post_id='?post_id';
UPDATE Pages SET post_count = post_count - 1
WHERE page_id = (SELECT page_id
                 FROM Posts
                 WHERE post_id = '?post_id');
```

Parameter type/definition

Post_id int(11): id of post we want to delete
page_id int(11): page that post is in.

Execution

```
DELETE FROM Posts WHERE post_id=2;
UPDATE Pages SET post_count = post_count - 1
WHERE page_id = (SELECT page_id
                 FROM Posts
                 WHERE post_id = 2);
```

Output

The post data deleted and post_count will be decremented.

Remove a comment (on user or group page)

SQL Statement

```
DELETE FROM Comments WHERE comment_id = '?comment_id';
UPDATE Posts SET comment_count = comment_count - 1
WHERE post_id = (SELECT post_id
                 FROM Comments
                 WHERE comment_id = '?comment_id');
```

Parameter type/definition

Comment_id int(11) : id of comment we want to remove
post_id int(11): post that comment is in.

Execution

```
DELETE FROM Comments WHERE comment_id = 2;
UPDATE Posts SET comment_count = comment_count - 1
WHERE post_id = (SELECT post_id
                 FROM Comments
                 WHERE comment_id = 2;
```

Output

The comment data deleted and comment_count will be decremented.

Unlike a post (On User or Group page)

SQL Statement

```
DELETE FROM LikedPosts WHERE user_id = '?user_id' AND post_id = '?post_id';
```

Parameter type/definition

user_id int(11): user id
post_id int(11): post to unlike

Execution

```
DELETE FROM LikedPosts WHERE user_id = 1 AND post_id = 1;
```

Output

User data is deleted from the LikedPosts table.

Unlike a comment (On User or Group page)

SQL Statement

DELETE FROM LikedComments WHERE user_id = ‘?user_id AND comment_id = ‘?comment_id’’;

Parameter type/definition

user_id int(11): user id

comment_id int(11): comment to unlike

Execution

DELETE FROM LikedComments WHERE user_id = ‘1’ AND comment_id = ‘1’;

Output

User data is deleted from the LikedComment table.

Modify a post (on user or group page)

SQL Statement

UPDATE Posts SET content=’?content’ WHERE post_id=’?post_id’;

Parameter type/definition

Content text : content of post

Post_id int(11): id of post

Execution

UPDATE Posts SET content=’After modify’ WHERE post_id=11;

Output

page_id	post_id	author_id	date_created	content	comment_count
1	11	1	2016-11-04 16:15:39	Before modify	2

page_id	post_id	author_id	date_created	content	comment_count
1	11	1	2016-11-04 16:15:39	After modify	2

Modify a comment (on user or group page)

SQL Statement

UPDATE Comments SET content=’?content’ WHERE comment_id=’?comment_id’;

Parameter type/definition

Content text : content of post

Post_id int(11): id of post

Execution

UPDATE Comments SET content=’After modify’ WHERE comment_id=2;

Output

post_id	comment_id	author_id	date_created	content
11	11	2	2016-11-04 23:14:39	Before modify

post_id	comment_id	author_id	date_created	content
11	11	2	2016-11-04 23:14:39	After modify

Delete a group

SQL Statement

DELETE FROM Groups WHERE group_id = ‘?group_id’ AND owner_id = (SELECT user_id FROM Users WHERE user_id = ‘?user_id’);

Parameter type/definition

owner_id int(11): owner of group we want to delete

group_id int(11): group we want to delete

user_id int(11): owner of the group

Execution

DELETE FROM Groups WHERE group_id = 2 AND owner_id = (SELECT user_id FROM Users WHERE user_id = 7);

Output

Group data is deleted from the group table.

Rename a group

SQL Statement

UPDATE Groups SET group_name='?group_name' WHERE group_id='?group_id' AND owner_id = (SELECT user_id FROM Users WHERE user_id = '?user_id');

Parameter type/definition

Group_name VARCHAR(50) : name we want to change
Group_id int(11): id of group we want to rename
Owner_id int(11): owner of the group
User_id int(11): user id

Execution

UPDATE Groups SET group_name='New group name' WHERE group_id=1 AND owner_id = (SELECT user_id FROM Users WHERE user_id = '2');

Output

group_id	group_name	type	owner_id
12	group to rename	team	10

group_id	group_name	type	owner_id
12	New group name	team	10

User-Level - regard to other users' groups:

Join a group

SQL Statement

INSERT INTO GroupMembers(?user_id, ?group_id);

Parameter type/definition

user_id int(11) : id of user we want to add to group
group_id int(11) : id of group user wants to join

Execution

INSERT INTO GroupMembers(1, 1)

Output

user_id	group_id
1	1

Unjoin a group

SQL Statement

DELETE FROM Groups WHERE user_id = 'user_id' AND group_id = '?group_id';

Parameter type/definition

user_id int(11) : id of user we want to add to group
group_id int(11) : id of group user wants to unjoin

Execution

DELETE FROM Groups WHERE user_id = 1 AND group_id = 1;

Output

User data from a group table is deleted.

Make a post on a group page

SQL Statement

INSERT INTO Posts (page_id, author_id, date_created, content, comment_count)
VALUES (SELECT p.page_id, '?author_id', NOW(), '?content', comment_count
FROM Pages p WHERE p.group_id = '?group_id');
UPDATE Pages SET post_count = post_count + 1 WHERE group_id = '?group_id';

Parameter type/definition

page_id int(11): group page
post_id int(11): post created
author_id int(11): author of the post

date_created datetime: the time post created
content text: content of the post
comment_count int: comment count of the post
group_id int(11): group of the page
post_count int: post count of the page

Execution

```
INSERT INTO Posts (page_id, author_id, date_created, content)
        SELECT p.page_id, 1, NOW(), "group page post"
        FROM Pages p WHERE p.group_id = 1;
UPDATE Pages SET post_count = post_count + 1 WHERE group_id = 1;
```

Output

page_id	post_id	author_id	date_created	content	comment_count
1	2	1	2016-11-02 17:10:37	group page post	0

Post_count is incremented.

Manager-Level Transactions

Manager-Level - General transactions:

Add information for an employee

SQL Statement

INSERT INTO Employees (employee_password, ssn, first_name, last_name, address, city, state, zipcode, telephone, date_started, hourly_rate, is_manager) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);

Parameter type/definition

password : CHAR(64), ssn : CHAR(11), first_name : VARCHAR(50), last_name : VARCHAR(50), address : VARCHAR(95), city : VARCHAR(35), state : CHAR(2), zipcode : VARCHAR(10), telephone : CHAR(12), date_started : DATE, hourly_rate : DECIMAL(7,2), is_manager : BOOLEAN

Execution

password = "5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8", ssn = “034-03-0341”, first_name = “John”, last_name = “Smith”, address = “123 Default St”, city = “Somewhere”, state = “NY”, zipcode = “12345”, telephone = “123-555-5555”, date_started = “2016-11-02”, hourly_rate = 9.50, is_manager = FALSE:

Output

employee_id	employee_password	ssn	first_name	last_name	address	city	state	zipcode	telephone	date_started	hourly_rate	is_manager
1	5E884898DA28047151D0E56F...	123-45-6789	John	Doe	7746 South Young St	Halethorpe	MD	21227	123-456-7890	2012-06-14	9.25	1
2	5E884898DA28047151D0E56F...	246-83-6790	Jane	Doe	7746 North Young St	Halethorpe	MD	21227	123-456-7890	2012-06-18	10.25	1
3	5E884898DA28047151D0E56F...	111-34-2020	Angus	McDonald	717 Lincoln St	Dayton	OH	45420	222-328-7712	2016-08-03	3.30	0
4	5E884898DA28047151D0E56F...	222-45-3131	Oliver	Williams	200 Lincoln St	Dayton	OH	45420	913-123-0924	2016-03-23	9.00	0
5	5E884898DA28047151D0E56F...	123-54-9813	Lisa	Hollands	81 Second Ave	Matawan	NJ	07747	346-912-8701	1993-10-13	45.00	1
6	5E884898DA28047151D0E56F...	213-13-3568	Marcus	Smith	401 Magnolia St	Aliquippa	PA	15001	121-984-3256	2014-11-25	12.00	0
7	5E884898DA28047151D0E56F...	074-26-1348	Joe	Schmoe	123 North St	Aliquippa	PA	15001	121-234-1349	2014-11-25	10.00	0
8	5E884898DA28047151D0E56F...	213-13-3568	Jenna	Sanders	38 Somewhere Dr	Aliquippa	PA	15001	121-434-2382	2014-11-23	10.15	0
9	5E884898DA28047151D0E56F...	012-86-2016	Sofia	Stocio	7114 Stonybrook Ct	Central Islip	NY	11722	631-983-3999	2015-04-03	11.50	0
10	5E884898DA28047151D0E56F...	312-31-4311	Kevin	Nadeau	880 Winchester Rd	Concord	NH	03301	121-872-0894	2013-06-01	11.00	0
11	5E884898DA28047151D0E56F...	034-03-0341	John	Smith	123 Default St	Somewhere	NY	12345	123-555-5555	2016-11-02	9.50	0

Edit and Delete information for an employee

SQL Statement

UPDATE Employees
SET employee_password = ?, ssn = ?, first_name = ?, last_name = ?, address = ?, city = ?, state = ?, zipcode = ?, telephone = ?, date_started = ?, hourly_rate = ?, is_manager = ?
WHERE employee_id = ?

Parameter type/definition

employee_password : CHAR(64), ssn : CHAR(11), first_name : VARCHAR(50), last_name : VARCHAR(50), address : VARCHAR(95), city : VARCHAR(35), state : CHAR(2), zipcode : VARCHAR(10), telephone : CHAR(12), date_started : DATE, hourly_rate : DECIMAL(7,2), is_manager : BOOLEAN, employee_id : INT

Execution

employee_password = "5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8", ssn = “034-03-0341”, **first_name = “Joey”**, last_name = “Doe”, address = “7746 South Young St”, city = “Halethorpe”, state = “MD”, zipcode = “21227”, telephone = “123-456-7890”, date_started = “2012-06-14”, hourly_rate = 9.55, is_manager = TRUE, **employee_id = 1**

Output

employee_id	employee_password	ssn	first_name	last_name	address	city	state	zipcode	telephone	date_started	hourly_rate	is_manager
1	5E884898DA28047151D0E56F...	123-45-6789	Joey	Doe	7746 South Young St	Halethorpe	MD	21227	123-456-7890	2012-06-14	9.25	1
2	5E884898DA28047151D0E56F...	246-83-6790	Jane	Doe	7746 North Young St	Halethorpe	MD	21227	123-456-7890	2012-06-18	10.25	1
3	5E884898DA28047151D0E56F...	111-34-2020	Angus	McDonald	717 Lincoln St	Dayton	OH	45420	222-328-7712	2016-08-03	3.30	0
4	5E884898DA28047151D0E56F...	222-45-3131	Oliver	Williams	200 Lincoln St	Dayton	OH	45420	913-123-0924	2016-03-23	9.00	0
5	5E884898DA28047151D0E56F...	123-54-9813	Lisa	Hollands	81 Second Ave	Matawan	NJ	07747	346-912-8701	1993-10-13	45.00	1
6	5E884898DA28047151D0E56F...	213-13-3568	Marcus	Smith	401 Magnolia St	Aliquippa	PA	15001	121-984-3256	2014-11-25	12.00	0
7	5E884898DA28047151D0E56F...	074-26-1348	Joe	Schmoe	123 North St	Aliquippa	PA	15001	121-234-1349	2014-11-25	10.00	0
8	5E884898DA28047151D0E56F...	213-13-3568	Jenna	Sanders	38 Somewhere Dr	Aliquippa	PA	15001	121-434-2382	2014-11-23	10.15	0
9	5E884898DA28047151D0E56F...	012-86-2016	Sofia	Stocio	7114 Stonybrook Ct	Central Islip	NY	11722	631-983-3999	2015-04-03	11.50	0
10	5E884898DA28047151D0E56F...	312-31-4311	Kevin	Nadeau	880 Winchester Rd	Concord	NH	03301	121-872-0894	2013-06-01	11.00	0

Note that employee data should be non-null, so values should not be deleted. But a value like “N/A” is acceptable. When the user wishes to alter employee information, the application should load existing values into a form to allow them to be modified or remain unchanged. That way the user does not have to re-enter unchanged data.

Obtain a sales report for a particular month

SQL Statement

SELECT * FROM Sales WHERE YEAR(date_sold) = ? AND MONTH(date_sold) = ?

Parameter type/definition

year : INT, month : INT

Execution

year = 2016, month = 10

Output

transaction_id	buyer_id	date_sold	advertisement_id	number_of_units	overseer_id
5	8	2016-10-03 10:20:00	9	5000	8
6	8	2016-10-04 04:31:00	10	1	10
7	4	2016-10-20 10:30:00	8	1	3
8	4	2016-10-20 19:30:00	8	3	3
9	3	2016-10-22 10:30:00	4	2	7
10	5	2016-10-25 10:30:00	6	5	6

Produce a comprehensive listing of all items being advertised on the site

SQL Statement

SELECT item_name FROM Advertisements

Parameter type/definition

None

Execution

Output

item_name
2016 Ford Somethingorother
2016 Subaru Outback
Levi's Boot Cut Jeans
Subway Sandwiches
Apple Macbook Pro
Samsung Galaxy Note 7
Doctor Strange
Database Systems for Dummies
Furby
2017 Honda Accord

Produce a list of transactions by item name

SQL Statement

SELECT transaction_id, buyer_id, date_sold, S.advertisement_id, number_of_units, overseer_id
FROM Sales S, Advertisements A
WHERE A.item_name = ? AND A.advertisement_id = S.advertisement_id

Parameter type/definition

item_name : VARCHAR(50)

Execution

item_name = “Apple Macbook Pro”

Output

transaction_id	buyer_id	date_sold	advertisement	number_of_units	overseer_id
3	9	2016-1...	5	5	6
4	6	2016-1...	5	500	6

Produce a list of transactions by user name

SQL Statement

SELECT transaction_id, buyer_id, date_sold, S.advertisement_id, number_of_units, overseer_id
FROM Sales S, Users U
WHERE S.buyer_id = U.user_id AND U.first_name = ? AND U.last_name = ?

Parameter type/definition

first_name : VARCHAR(50), last_name : VARCHAR(50)

Execution

first_name = “Paul”, last_name = “Fodor”

Output

transaction_id	buyer_id	date_sold	advertisement	number_of_units	overseer_id
7	4	2016-1...	8	1	3
8	4	2016-1...	8	3	3

Produce a summary listing of revenue generated by a particular item

SQL Statement

SELECT S.transaction_id, S.buyer_id, S.date_sold, S.advertisement_id, S.overseer_id, S.number_of_units, S.charge_amount AS ‘revenue’
FROM Sales S
WHERE S.advertisement_id = ?

Parameter type/definition

item_id : INT

Execution

item_id = 4

Output

transaction_id	buyer_id	date_sold	advertisement_id	overseer_id	number_of_units	revenue
9	3	2016-10-22 10:30:00	4	7	2	10.00

Produce a summary listing of revenue generated by a particular item type

SQL Statement

SELECT S.transaction_id, S.buyer_id, S.date_sold, S.advertisement_id, S.overseer_id, S.number_of_units, (S.charge_amount) AS ‘revenue’
FROM Sales S, Advertisements A
WHERE A.type = ? AND A.advertisement_id = S.advertisement_id

Parameter type/definition

item_type : VARCHAR(50)

Execution

item_type = “cars”

Output

transaction_id	buyer_id	date_sold	advertisement_id	overseer_id	number_of_units	revenue
1	1	2016-07-14 10:30:00	1	1	1	25000.00
2	7	2016-07-18 10:30:00	2	3	250	5500000.00
6	8	2016-10-04 04:31:00	10	10	1	22355.00

Produce a summary listing of revenue generated by a particular customer

SQL Statement

SELECT S.transaction_id, S.buyer_id, S.date_sold, S.advertisement_id, S.overseer_id, S.number_of_units, (S.charge_amount) AS 'revenue'
FROM Sales S
WHERE S.buyer_id = ?

Parameter type/definition

buyer_id : INT

Execution

buyer_id = 4

Output

transaction_id	buyer_id	date_sold	advertisement_id	overseer_id	number_of_units	revenue
7	4	2016-10-20 10:30:00	8	3	1	29.99
8	4	2016-10-20 19:30:00	8	3	3	89.97

Determine which customer representative generated the most total revenue

SQL Statement

SELECT S.overseer_id, SUM(S.charge_amount) as 'revenue'
FROM Sales S
GROUP BY overseer_id
ORDER BY revenue DESC LIMIT 1

Parameter type/definition

None

Execution

Output

overseer_id	revenue
3	5500119.96

Determine which customer generated the most total revenue

SQL Statement

SELECT S.buyer_id, SUM(S.charge_amount) as 'revenue'
FROM Sales S
GROUP BY buyer_id

ORDER BY revenue DESC LIMIT 1

Parameter type/definition

None

Execution

Output

buyer_id	revenue
7	5500000.00

Produce a list of most active items

SQL Statement

SELECT A.item_name, COUNT(A.item_name) as number_of_transactions
FROM Sales S, Advertisements A
WHERE A.advertisement_id = S.advertisement_id
GROUP BY item_name
ORDER BY number_of_transactions DESC, item_name DESC

Parameter type/definition

None

Execution

Output

item_name	number_of_transactions
Database Systems for Dummies	2
Apple Macbook Pro	2
Subway Sandwiches	1
Samsung Galaxy Note 7	1
Furby	1
2017 Honda Accord	1
2016 Subaru Outback	1
2016 Ford Somethingorother	1

Produce a list of all customers who have purchased a particular item

SQL Statement

SELECT U.user_id, U.first_name, U.last_name
FROM Sales S, Advertisements A, Users U
WHERE S.advertisement_id = A.advertisement_id AND S.buyer_id = U.user_id AND A.item_name = ?

Parameter type/definition

item_name : VARCHAR(50)

Execution

item_name = “Apple Macbook Pro”

Output

user_id	first_name	last_name
9	Melissa	Randomperson
6	Tim	Cook

Produce a list of all items for a given company

SQL Statement

SELECT item_name FROM Advertisements WHERE company = ?

Parameter type/definition

company_name : VARCHAR(50)

Execution

company_name = “Apple”

Output

item_name
Apple Macbook Pro

Customer Representative-Level Transactions

Customer Representative-Level - Sales agent:

Create an advertisement

SQL Statement

INSERT INTO Advertisements (employee_id, type, date_created, company, item_name, content, unit_price, available_units) VALUES (?, ?, NOW(), ?, ?, ?, ?, ?)

Parameter type/definition

employee_id : INT, ad_type : VARCHAR(50), company : VARCHAR(50), item_name : VARCHAR(50), content: TEXT, unit_price : DECIMAL, available_units : INT

Execution

employee_id = 1, ad_type = 'books', company = 'McGraw Hill', item_name = 'MySQL Guide', content = 'Book for learning how to use MySQL database systems', unit_price = 24.99, available_units = 100

Output

advertisement_id	employee_id	type	date_created	company	item_name	content	unit_price	available_units
1	1	cars	2016-06-14 10:30:00	Ford	2016 Ford Somethingorother	A car with 4 wheel drive!	25000.00	1000
2	5	cars	2016-06-14 10:30:00	Subaru	2016 Subaru Outback	The best car you'll ever drive.	22000.00	2000
3	5	clothing	2016-06-14 10:30:00	Levi's	Levi's Boot Cut Jeans	Comfy jeans!	59.99	135
4	7	food	2016-07-03 10:30:00	Subway	Subway Sandwiches	Eat fresh(tm).	5.00	100000
5	6	computers	2016-07-15 10:30:00	Apple	Apple Macbook Pro	A fancy new computer.	999.99	50000
6	6	computers	2016-07-15 10:30:00	Samsung	Samsung Galaxy Note 7	Hope it doesn't explode.	999.99	50000
7	4	movies	2016-09-05 10:30:00	Disney	Doctor Strange	Buy tickets for the new Marvel ...	9.99	1000000
8	3	books	2016-09-07 10:30:00	Random House	Database Systems for Dummies	Learn MySQL with this new book.	29.99	250000
9	8	toys	2016-09-08 10:30:00	Tiger Electronics	Furby	Own a weird robot pet!	59.99	123000
10	10	cars	2016-09-13 10:30:00	Honda	2017 Honda Accord	A reliable car.	22355.00	67000
11	1	books	2016-11-02 16:04:08	McGraw Hill	MySQL Guide	Book for learning how to use M...	24.99	100

Delete an advertisement

SQL Statement

DELETE FROM Advertisements WHERE advertisement_id = ?

Parameter type/definition

advertisement_id : INT

Execution

advertisement_id = 7

Output

advertisement_id	employee_id	type	date_created	company	item_name	content	unit_price	available_units
1	1	cars	2016-06-14 10:30:00	Ford	2016 Ford Somethingorother	A car with 4 wheel drive!	25000.00	1000
2	5	cars	2016-06-14 10:30:00	Subaru	2016 Subaru Outback	The best car you'll ever drive.	22000.00	2000
3	5	clothing	2016-06-14 10:30:00	Levi's	Levi's Boot Cut Jeans	Comfy jeans!	59.99	135
4	7	food	2016-07-03 10:30:00	Subway	Subway Sandwiches	Eat fresh(tm).	5.00	100000
5	6	computers	2016-07-15 10:30:00	Apple	Apple Macbook Pro	A fancy new computer.	999.99	50000
6	6	computers	2016-07-15 10:30:00	Samsung	Samsung Galaxy Note 7	Hope it doesn't explode.	999.99	50000
8	3	books	2016-09-07 10:30:00	Random House	Database Systems for Dummies	Learn MySQL with this new book.	29.99	250000
9	8	toys	2016-09-08 10:30:00	Tiger Electronics	Furby	Own a weird robot pet!	59.99	123000
10	10	cars	2016-09-13 10:30:00	Honda	2017 Honda Accord	A reliable car.	22355.00	67000

Note that advertisements that have resulted in a purchase cannot be deleted (for database integrity).

Record a transaction

SQL Statement

```
SET @advertisement_id = ?;
SET @num_units = ?;
SET @buyer_id = ?;
INSERT INTO Sales (buyer_id, card_number, date_sold, advertisement_id, number_of_units, overseer_id, charge_amount)
VALUES (@buyer_id, ?, NOW(), @advertisement_id, @num_units, ?,
        (SELECT @num_units * unit_price
        FROM Advertisements A
        WHERE A.advertisement_id = @advertisement_id));
# increment user purchase rating
```

UPDATE Users
SET purchase_rating = purchase_rating + @num_units
WHERE user_id = @buyer_id;
decrement available units
UPDATE Advertisements
SET available_units = available_units - @num_units
WHERE advertisement_id = @advertisement_id;

Parameter type/definition

advertisement_id : INT, number_of_units : INT, buyer_id : INT, card_number : CHAR(16), overseer_id : INT

Execution

advertisement_id = 1, number_of_units = 1, buyer_id = 1, card_number = “1234567812345678”, overseer_id = 1

Output

transaction_id	buyer_id	card_number	date_sold	advertisement_id	number_of_units	overseer_id	charge_amount
1	1	1947234500008264	2016-07-14 10:30:00	1	1	1	25000.00
2	7	4444999912345555	2016-07-18 10:30:00	2	250	3	5500000.00
3	9	1234123412341234	2016-08-01 11:20:00	5	5	6	4999.95
4	6	9000900080006000	2016-08-19 10:30:00	5	500	6	499995.00
5	8	3511927539490047	2016-10-03 10:20:00	9	5000	8	299950.00
6	8	3511927539490047	2016-10-04 04:31:00	10	1	10	22355.00
7	4	1111111111111111	2016-10-20 10:30:00	8	1	3	29.99
8	4	1111111111111111	2016-10-20 19:30:00	8	3	3	89.97
9	3	9999444477772222	2016-10-22 10:30:00	4	2	7	10.00
10	5	2234983513786645	2016-10-25 10:30:00	6	5	6	4999.95
11	1	1234567812345678	2016-11-02 19:48:52	1	1	1	25000.00

Add information for a customer

Equivalent to User-level - General transaction - Register.

Edit and Delete information for a customer

SQL Statement

UPDATE Users
SET user_password = ?, first_name = ?, last_name = ?, address = ?, city = ?, state = ?, zipcode = ?, telephone = ?, email = ?, credit_card = ?, purchase_rating = ?
WHERE user_id = ?

Parameter type/definition

user_password : CHAR(64), first_name : VARCHAR(50), last_name : VARCHAR(50), address : VARCHAR(95), city : varchar(35), state : char(2), zipcode : VARCHAR(10), telephone : CHAR(12), email : VARCHAR(255), credit_card : CHAR(16), purchase_rating : INT, user_id : INT

Execution

user_password = "5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8", first_name = "Bryan", last_name = "Koelbel", address = "123 South Dr", city = "Stony Brook", state = "NY", zipcode = "11790", telephone = "631-123-4567", email = "bryan.koelbel@stonybrook.edu", **credit_card = “1234567812345678”**, purchase_rating = 1, user_id = 1

Output

After execution, result of SELECT user_id, first_name, last_name, credit_card FROM Users

user_id	first_name	last_name	credit_card
1	Bryan	Koelbel	1234567812345678
2	Jeonghoon	Kim	NULL
3	Chaerin	Kim	9999444477772222
4	Paul	Fodor	1111111111111111
5	Jennifer	Wong	2234983513786645
6	Tim	Cook	NULL
7	Mark	Zuckerberg	4444999912345555
8	Ben	Carson	3511927539490047
9	Melissa	Randomperson	1234123412341234
10	Joe	Randomperson	NULL

Note that most User (customer) data should be non-null, so values should not be deleted. But a value like “N/A” is acceptable. When the employee wishes to alter customer information, the application should load existing values into a form to allow them to be modified or remain unchanged. That way the user does not have to re-enter unchanged data.

Produce customer mailing lists

SQL Statement

SELECT email FROM Users U

Parameter type/definition

None

Execution

Output

email
bryan.koelbel@stonybrook.edu
jeonghoon.kim@stonybrook.edu
chaerin.kim@stonybrook.edu
paul.fodor@stonybrook.edu
jwong@cs.sunysb.edu
tcook@apple.com
zuck@facebook.com
benny1248@yahoo.com
somerando636@gmail.com
someotherrando636@gmail.com

Produce a list of item suggestions for a given customer (based on that customer's past transactions)

SQL Statement

```
SELECT I.item_name, I.content
FROM Advertisements I, (
        SELECT A.type
        FROM Advertisements A, Sales S
        WHERE S.buyer_id = ? AND S.advertisement_id = A.advertisement_id) F WHERE I.type = F.type
```

Parameter type/definition

user_id : INT

Execution

user_id = 1

Output

item_name	content
2016 Ford Somethingorother	A car with 4 wheel drive!
2016 Subaru Outback	The best car you'll ever drive.
2017 Honda Accord	A reliable car.

Customer Representative-Level - Customer:

Purchase one or more copies of an advertised item

SQL Statement

```
SET @advertisement_id = ?;
SET @num_units = ?;
SET @buyer_id = ?;
INSERT INTO Sales (buyer_id, card_number, date_sold, advertisement_id, number_of_units, overseer_id, charge_amount)
VALUES (@buyer_id, ?, NOW(), @advertisement_id, @num_units, ?,
        (SELECT @num_units * unit_price
        FROM Advertisements A
        WHERE A.advertisement_id = @advertisement_id));
# increment user purchase rating
UPDATE Users
SET purchase_rating = purchase_rating + @num_units
WHERE user_id = @buyer_id;
# decrement available units
```

UPDATE Advertisements
SET available_units = available_units - @num_units
WHERE advertisement_id = @advertisement_id;

Parameter type/definition

advertisement_id : INT, number_of_units : INT, buyer_id : INT, card_number : CHAR(16), overseer_id : INT

Execution

advertisement_id = 1, number_of_units = 1, buyer_id = 1, card_number = “1234567812345678”, overseer_id = 1
After execution, the result of SELECT * FROM Sales

Output

transaction_id	buyer_id	card_number	date_sold	advertisement_id	number_of_units	overseer_id	charge_amount
1	1	1947234500008264	2016-07-14 10:30:00	1	1	1	25000.00
2	7	4444999912345555	2016-07-18 10:30:00	2	250	3	5500000.00
3	9	1234123412341234	2016-08-01 11:20:00	5	5	6	4999.95
4	6	9000900080006000	2016-08-19 10:30:00	5	500	6	499995.00
5	8	3511927539490047	2016-10-03 10:20:00	9	5000	8	299950.00
6	8	3511927539490047	2016-10-04 04:31:00	10	1	10	22355.00
7	4	1111111111111111	2016-10-20 10:30:00	8	1	3	29.99
8	4	1111111111111111	2016-10-20 19:30:00	8	3	3	89.97
9	3	9999444477772222	2016-10-22 10:30:00	4	2	7	10.00
10	5	2234983513786645	2016-10-25 10:30:00	6	5	6	4999.95
11	1	1234567812345678	2016-11-02 19:48:52	1	1	1	25000.00

A Customer’s current groups

SQL Statement

SELECT G.group_id, G.group_name
FROM Groups G, GroupMembers M
WHERE M.user_id = ? AND M.group_id = G.group_id

Parameter type/definition

user_id : INT

Execution

user_id = 1

Output

group_id	group_name
1	CSE305 Project
4	Car Fans

For each of a customer’s accounts, the account history

SQL Statement

SELECT S.transaction_id, S.date_sold, A.item_name, S.charge_amount, S.overseer_id
FROM Sales S, Advertisements A
WHERE S.card_number = ? AND S.advertisement_id = A.advertisement_id

Parameter type/definition

account_number : CHAR(16)

Execution

account_number = "1111111111111111"

Output

transaction_id	date_sold	item_name	charge_amount	overseer_id
7	2016-10-20 10:30:00	Database Systems for Dummies	29.99	3
8	2016-10-20 19:30:00	Database Systems for Dummies	89.97	3

Best-seller list of items

SQL Statement

SELECT A.item_name, SUM(S.number_of_units) as units_sold
FROM Sales S, Advertisements A
WHERE A.advertisement_id = S.advertisement_id
GROUP BY item_name

ORDER BY units_sold DESC, item_name DESC

Parameter type/definition

None

Execution

Output

item_name	units_sold
Furby	5000
Apple Macbook Pro	505
2016 Subaru Outback	250
Samsung Galaxy Note 7	5
Database Systems for Dummies	4
Subway Sandwiches	2
2017 Honda Accord	1
2016 Ford Somethingorother	1

Personalized item suggestion list

SQL Statement

SELECT A.advertisement_id, A.item_name, A.content
FROM Advertisements A, Preferences P
WHERE P.user_id = ? AND P.ad_type = A.type

Parameter type/definition

user_id : INT

Execution

user_id = 1

Output

advertisement_id	item_name	content
1	2016 Ford Somethingorother	A car with 4 wheel drive!
2	2016 Subaru Outback	The best car you'll ever drive.
3	Levi's Boot Cut Jeans	Comfy jeans!
10	2017 Honda Accord	A reliable car.

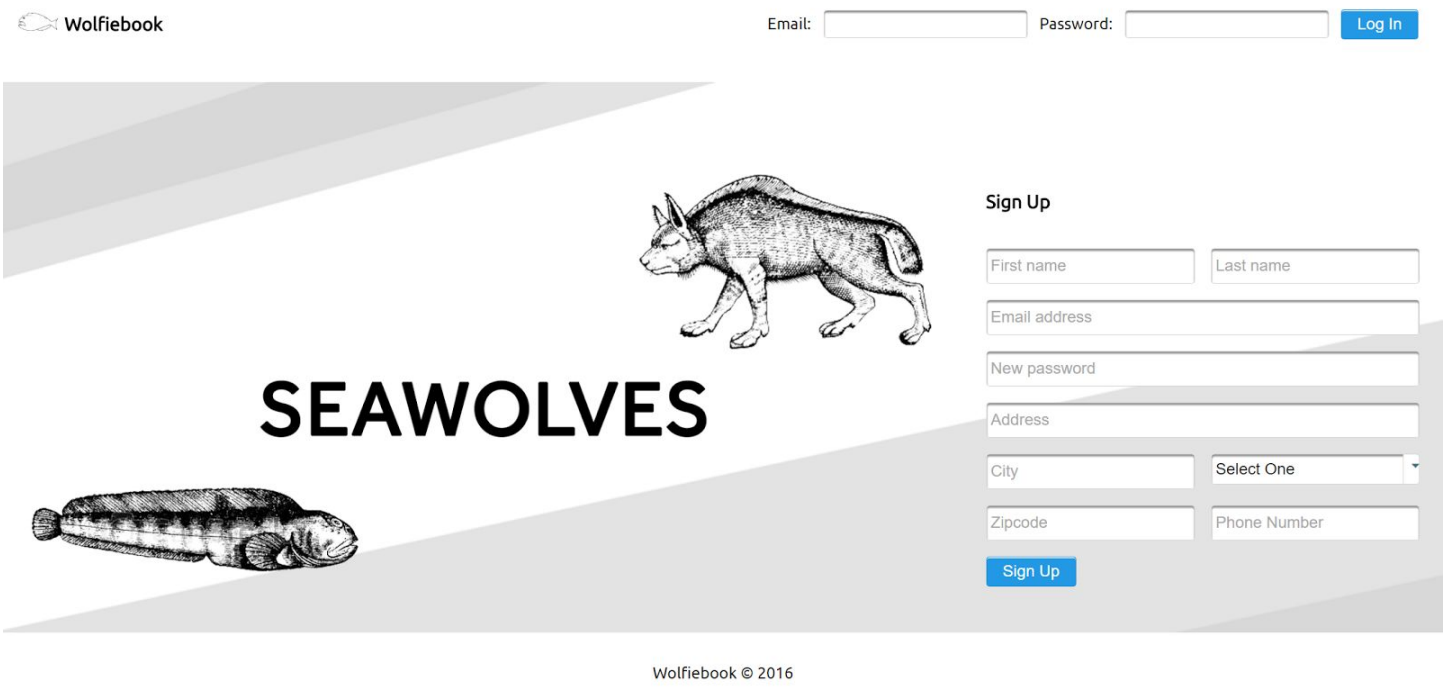
Wolfiebook

CSE305.01 Fall 2016
Project Assignment 3

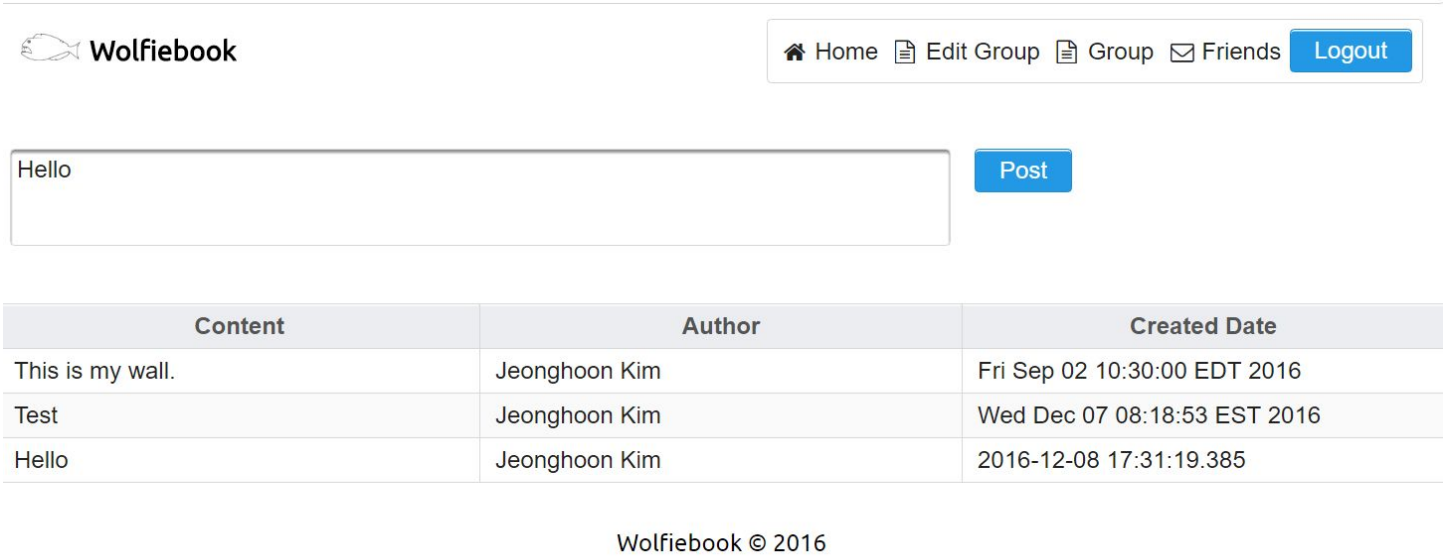
Chaerin Kim - Chaerin.Kim@stonybrook.edu
Jeonghoon Kim - Jeonghoon.Kim@stonybrook.edu
Bryan Koelbel - Bryan.Koelbel@stonybrook.edu

User Site

Home Page:



Posting on Wall:



Friends List:

Wolfiebook

Home | Edit Group | Group | Friends | Logout

Search users: Enter user name, email or phone number

1

First Name	Last Name
Bryan	Koelbel
Jeonghoon	Kim
Chaerin	Kim
Paul	Fodor
Jennifer	Wong
Tim	Cook
Mark	Zuckerberg
Ben	Carson
Melissa	Randomperson
Joe	Randomperson

1

_View

Wolfiebook © 2016

Messaging a Friend:

Wolfiebook

Home | Edit Group | Group | Friends | Logout

Search users: Enter user name, email or phone number

Chaerin Kim

Messages

Hello!

How are you?

1

Subject	Content	From
Hello!	How are you?	Jeonghoon Kim

1

Delete

Send | Close

Viewing Groups:

Wolfiebook

Home | Edit Group | Group | Friends | Logout

All Groups

1

Group Name	Owner	Type
CSE305 Project	Chaerin	team
Stony Brook CS Department	Jennifer	organization
Illuminati	Tim	organization
Car Fans	Jennifer	club
Volunteer Club	Melissa	club
Stony Brook Swim Team	Paul	team
Pen Pals	Melissa	club
Republican National Committee	Ben	organization
ABC Club	Jeonghoon	club
123 Club	Chaerin	club
blah	Jeonghoon	test

1

Join

Joined Groups

1

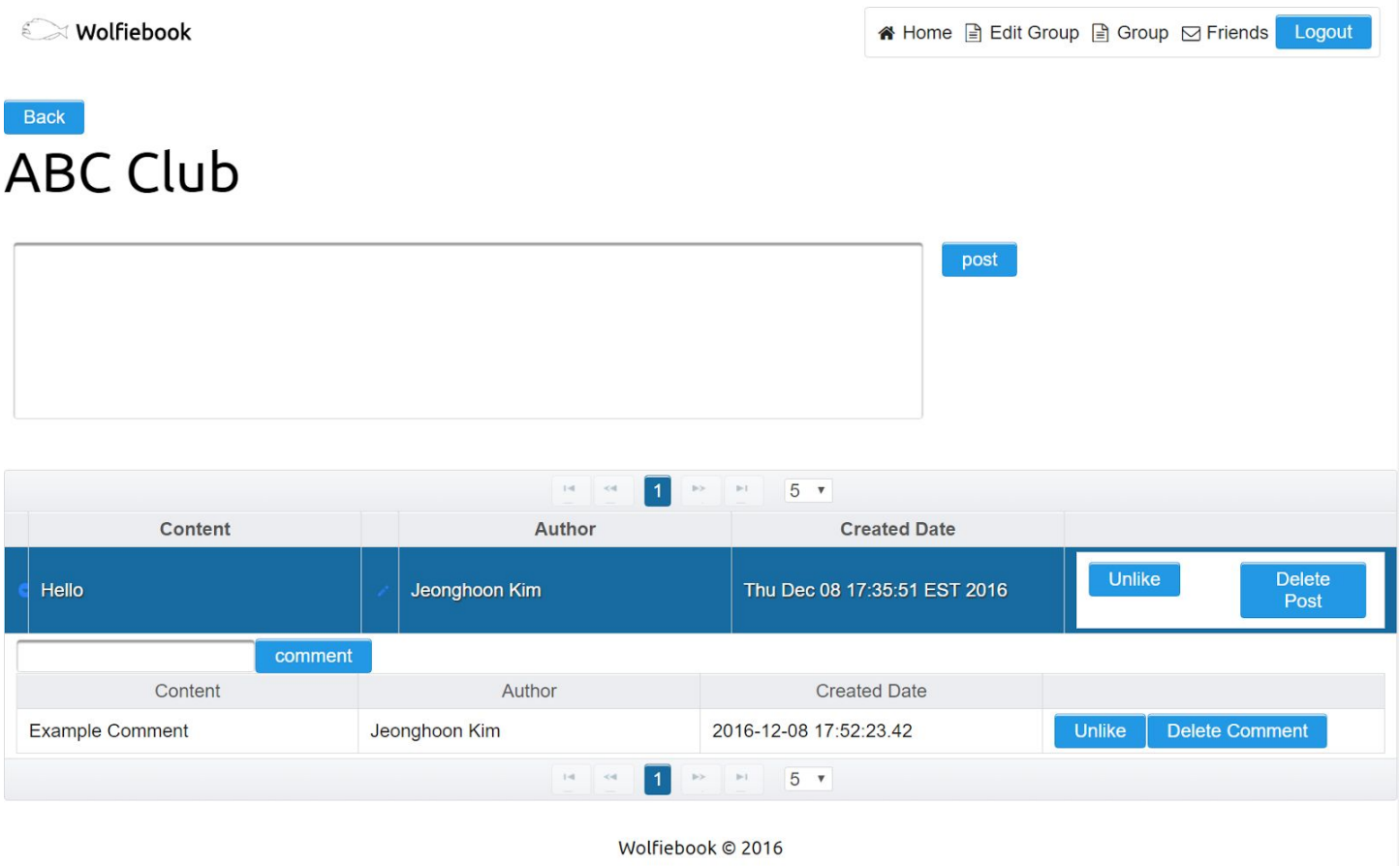
Group Name	Owner	Type
ABC Club	Jeonghoon	club
CSE305 Project	Chaerin	team

1

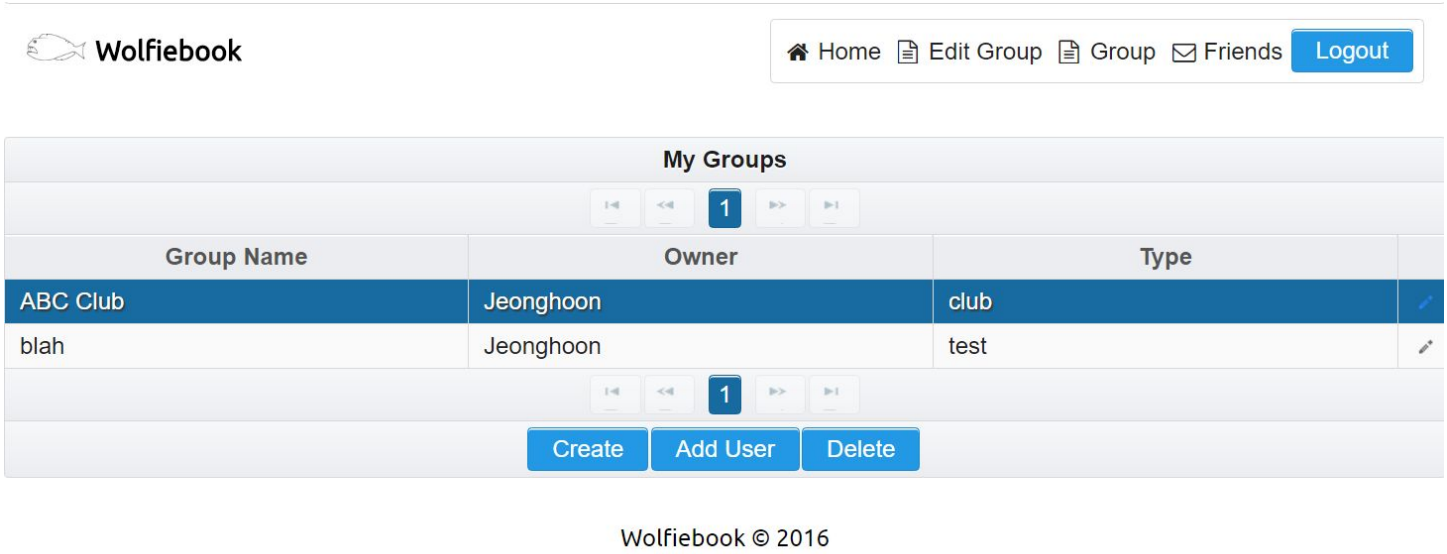
View

Wolfiebook © 2016

Posting and Commenting in a Group:

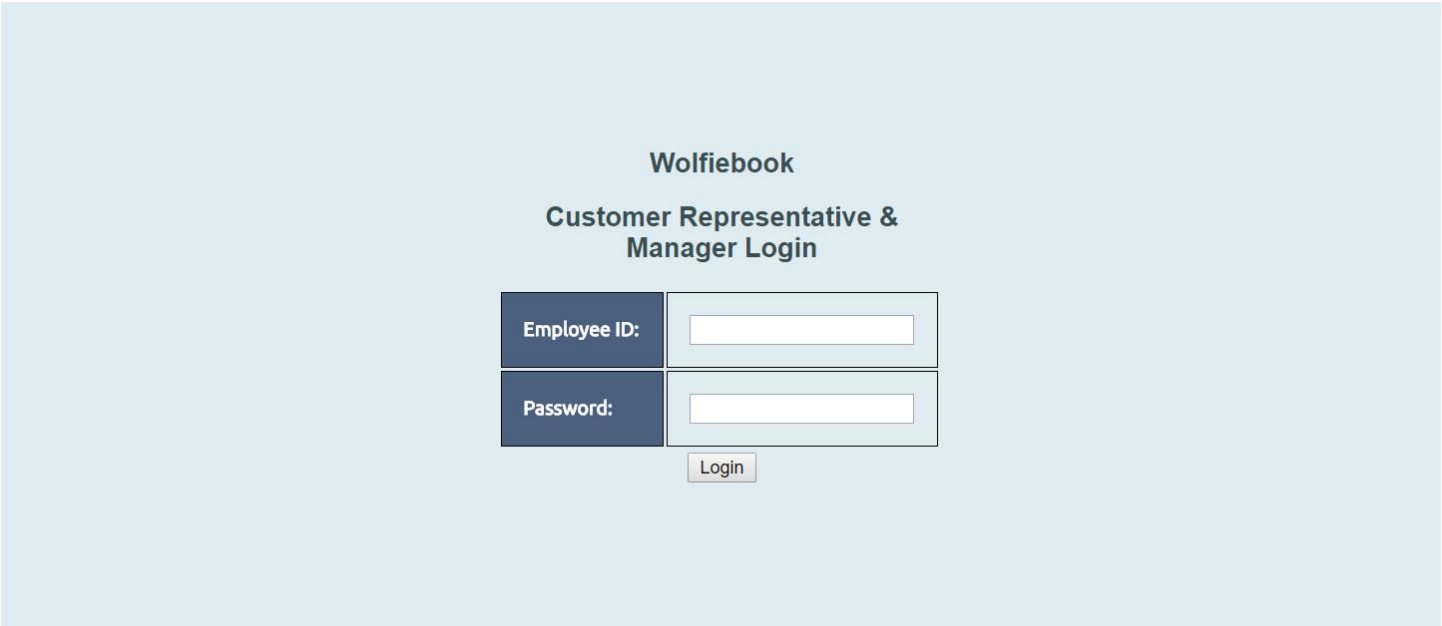


Editing Your Groups:



Manager Site

Login:



Home Page:

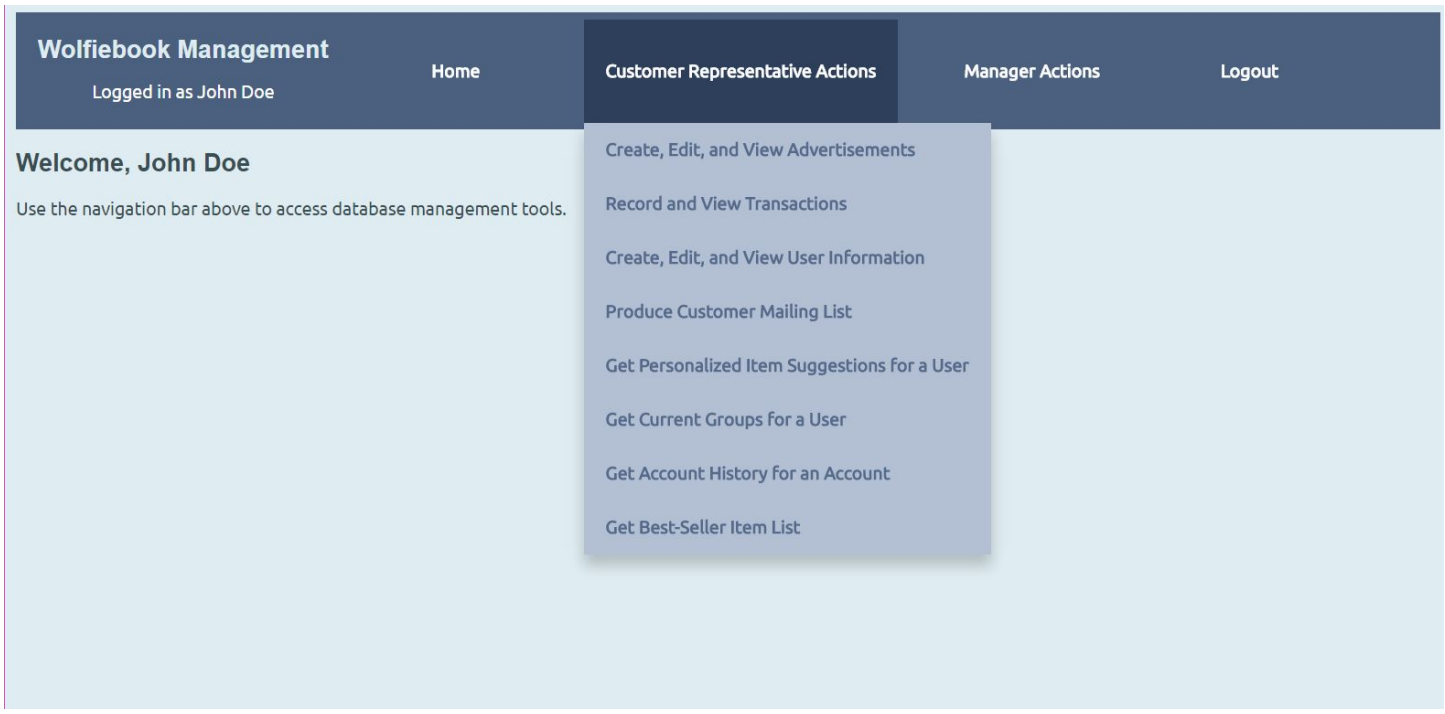


Table View:

Wolfiebook Management

Logged in as John Doe

Home | Customer Representative Actions | Manager Actions | Logout

Advertisements

Ad ID	Employee ID	Ad Type	Date Created	Company	Item Name	Description	Unit Price	Available Units	Tools
1	1	cars	2016-06-14 10:30:00.0	Ford	2016 Ford Somethingorother	A car with 4 wheel drive!	$25000.00	1000	Edit Delete
2	5	cars	2016-06-14 10:30:00.0	Subaru	2016 Subaru Outback	The best car you'll ever drive.	$22000.00	2000	Edit Delete
3	5	clothing	2016-06-14 10:30:00.0	Levi's	Levi's Boot Cut Jeans	Comfy jeans!	$59.99	135	Edit Delete
4	7	food	2016-07-03 10:30:00.0	Subway	Subway Sandwiches	Eat fresh(tm).	$5.00	100000	Edit Delete
5	6	computers	2016-07-15 10:30:00.0	Apple	Apple Macbook Pro	A fancy new computer.	$999.99	50000	Edit Delete
6	6	computers	2016-07-15 10:30:00.0	Samsung	Samsung Galaxy Note 7	Hope it doesn't explode.	$999.99	50000	Edit Delete

Table Item Create:

Wolfiebook Management

Logged in as John Doe

Home | Customer Representative Actions | Manager Actions | Logout

Record a Transaction

Enter information below.

Buyer ID:

Bank Account Number (if blank, defaults to User's current account number):

Advertisement ID:

Number of Units:

Submit

Table Item Edit:

Wolfiebook Management

Logged in as John Doe

Home | Customer Representative Actions | Manager Actions | Logout

Edit Advertisement

Advertisement #1

Created by Employee #1 at 2016-06-14 10:30:00.0

Ad Type:

cars

Company Name:

Ford

Item Name:

2016 Ford Somethingorothe

Description:

A car with 4 wheel drive!

Unit Price:

25000.00

Number of Available Units:

1000

Update

Table Item Delete:

Wolfiebook Management

Logged in as John Doe

Home | Customer Representative Actions | Manager Actions | Logout

Delete Advertisement

Advertisement successfully deleted!

View All Advertisements

Misc. Transaction Example:

Wolfiebook Management

Logged in as John Doe

Home | Customer Representative Actions | Manager Actions | Logout

View Which Representative Generated the Most Revenue

Employee #3 (Angus McDonald) has generated the most revenue.

They have generated $5500119.96 to date.

Transaction ID	Buyer ID	Card Number	Date Sold	Ad ID	Units	Overseer ID	Revenue
2	7	4444999912345555	2016-07-18 10:30:00.0	2	250	3	$5500000.00
7	4	1111111111111111	2016-10-20 10:30:00.0	8	1	3	$29.99
8	4	1111111111111111	2016-10-20 19:30:00.0	8	3	3	$89.97

(all other transactions follow a similar format)