## Instructions

| *Read through this document, studying the tables and ERD. Then, write SQL queries that answer the questions listed in the* [*SQL Queries*](#_wf5f22f6eq6k) *section. Finally, submit your answers.* |
| --- |

The following questions contain a series of tables taken from the social media site FindingFastFriends.

## Tables Overview

You can see all the tables and their descriptions listed below. Each of these tables contain 1M+ rows of data. You can see the first three rows of each table to get a sense of what they contain. The tables necessary to answer each individual question will be repeated throughout the assessment for you to reference.

| users | | | |
| --- | --- | --- | --- |
| user\_id | username | email | friend\_count |
| 3437315 | janedoe2.0 | janedoe2.0@email.com | 5 |
| 3437316 | princessluv8996 | princessluv8996@email.com | 600017 |
| 3437317 | sk8ter4l!f3 | sk8ter4l!f3@email.com | 583 |

**Table Name:** *users*

**user\_id:** *Id of user*

**username:** *name of user*

**email:** *email of user*

**friend\_count:** *user’s number of friends*

***Table Description:*** *New rows are added as a new user is created.*

| friend\_requests | | | | |
| --- | --- | --- | --- | --- |
| action\_id | requester\_id | requestee\_id | action\_timestamp | action\_taken |
| 1 | 1037392 | 3437315 | 2015-03-15 00:01:05 | Requested |
| 2 | 2138102 | 5438443 | 2015-03-15 00:01:07 | Accepted |
| 3 | 2331234 | 1231232 | 2015-03-15 00:01:08 | Rejected |

**Table Name:** *friend\_requests*

**action\_id:** *Id of action*

**requester\_id:** *Id of user who took action*

**requestee\_id:** *Id of user who had action taken on*

**action\_timestamp:** *Timestamp of user A action*

**action\_taken:** *Type of action user took (Requested, Accepted, Rejected)*

**Table Description:** Each action receives its own unique *action\_id* within this table. For example, after a friend request is sent and then accepted that means two rows of *action\_id*s exist in this table for these two actions.

| **messages** | | | | | |
| --- | --- | --- | --- | --- | --- |
| **message\_id** | **from\_user\_id** | **to\_user\_id** | **date\_sent** | **date\_read** | **message** |
| 1 | 3437317 | 2138102 | 2015-03-15 00:02:17 | 2015-03-15 00:03:05 | “Hi!!! Wanted to know how u were doing?! Miss you !! ...” |
| 2 | 1438443 | 5937440 | 2015-03-15 00:02:24 | NULL | “Don’t forget to like and comment on my new pic...” |
| 3 | 2331234 | 1231232 | 2015-03-15 00:02:25 | 2015-04-01 00:11:08 | “Let’s hang out!” |

**Table Name:** *messages*

**message\_id:** *Id of message*

**from\_user\_id:** *Id of user sending the message*

**to\_user\_id:** *Id of user receiving the message*

**date\_sent:** *date that the message was sent*

**date\_read:** *date that the message was read. NULL if message has not been read.*

**message:** *text of the message*

***Table Description:*** *New rows are added as new messages are sent.*

## ERD

Here is the ERD of the database for your reference.

The left-most folder is titled "friend_requests", the middle is "users", and the right-most folder is "messages".

friend_requests table has a primary key: action_id. Friend_requests has two foreign keys: userA_id and userB_id that reference the users table. Users table has a primary key: user_id. messages table has a primary key: message_id. Messages table has 2 foreign keys: from_user_id and to_user_id that reference the users table

## SQL Queries

You have a users table containing 1M+ rows of user information. Below are the first 3 rows.

| users | | | |
| --- | --- | --- | --- |
| user\_id | username | email | friend\_count |
| 3437315 | janedoe2.0 | janedoe2.0@email.com | 5 |
| 3437316 | princessluv8996 | princessluv8996@email.com | 600017 |
| 3437317 | sk8ter4l!f3 | sk8ter4l!f3@email.com | 583 |

### **Question 1: Write a SQL query that returns the email address and friend count of the user with the most friends.**

HINT: What if more than one user has the same "max" friend count?

| SELECT  email, friend\_count  FROM users  WHERE friend\_count = (SELECT MAX(friend\_count) FROM users); |
| --- |

You have a users table containing 1M+ rows of user information and a friend\_requests table containing 1M+ rows of friend request information. Below are the first 3 rows of each table.

| users | | | |
| --- | --- | --- | --- |
| user\_id | username | email | friend\_count |
| 3437315 | janedoe2.0 | janedoe2.0@email.com | 5 |
| 3437316 | princessluv8996 | princessluv8996@email.com | 600017 |
| 3437317 | sk8ter4l!f3 | sk8ter4l!f3@email.com | 583 |

| friend\_requests | | | | |
| --- | --- | --- | --- | --- |
| action\_id | requester\_id | requestee\_id | action\_timestamp | action\_taken |
| 1 | 1037392 | 3437315 | 2015-03-15 00:01:05 | Requested |
| 2 | 2138102 | 5438443 | 2015-03-15 00:01:07 | Accepted |
| 3 | 2331234 | 1231232 | 2015-03-15 00:01:08 | Rejected |

### **Question 2: Write a SQL query that returns the three users who have sent the most friend requests. Your query should return the username and number of requests sent.**

| SELECT  u.username,  COUNT(fr.requester\_id) AS num\_requests\_sent  FROM users u  JOIN friend\_requests fr ON u.user\_id = fr.requester\_id  WHERE fr.action\_taken = ‘Requested’  GROUP BY u.user\_id, username  ORDER BY num\_requests\_sent DESC  LIMIT 3; |
| --- |

Consider the friend\_requests table again. It contains a column action\_taken that shows whether a friend request was: Requested, Accepted or Rejected.

| friend\_requests | | | | |
| --- | --- | --- | --- | --- |
| action\_id | requester\_id | requestee\_id | action\_timestamp | action\_taken |
| 1 | 1037392 | 3437315 | 2015-03-15 00:01:05 | Requested |
| 2 | 2138102 | 5438443 | 2015-03-15 00:01:07 | Accepted |
| 3 | 2331234 | 1231232 | 2015-03-15 00:01:08 | Rejected |

### **Question 3A: Write a query to determine the number of Accepted friend requests.**

| SELECT  COUNT(\*)  FROM friend\_requests  WHERE action\_taken = ‘Accepted’; |
| --- |

### **Question 3B: Write a second query to determine the percentage of requests that are Accepted.**

HINT: Reading the table description (in Table Overview) is important in understanding how this table is populated.

| SELECT  (COUNT(CASE WHEN action\_taken = ‘Accepted’ THEN 1 END) / COUNT(\*)) \* 100 AS percentage\_accepted  FROM friend\_requests; |
| --- |

The database also has a third table messages, that includes 1M+ rows of information related to the messages sent between users. Here are the first three rows.

| **messages** | | | | | |
| --- | --- | --- | --- | --- | --- |
| **message\_id** | **from\_user\_id** | **to\_user\_id** | **date\_sent** | **date\_read** | **message** |
| 1 | 3437317 | 2138102 | 2015-03-15 00:02:17 | 2015-03-15 00:03:05 | “Hi!!! Wanted to know how u were doing?! Miss you !! ...” |
| 2 | 1438443 | 5937440 | 2015-03-15 00:02:24 | NULL | “Don’t forget to like and comment on my new pic...” |
| 3 | 2331234 | 1231232 | 2015-03-15 00:02:25 | 2015-04-01 00:11:08 | “Let’s hang out!” |

### **Question 4:** **Write a SQL query to count the number of messages that include the following phrase: “Miss you” (Note: You should account for a capital “M” and lowercase “m”.)**

| SELECT  COUNT(\*)  FROM messages  WHERE LOWER(message) LIKE ‘%miss you%’; |
| --- |

Consider the following users and messages tables to answer the question below.

| users | | | |
| --- | --- | --- | --- |
| user\_id | username | email | friend\_count |
| 3437315 | janedoe2.0 | janedoe2.0@email.com | 5 |
| 3437316 | princessluv8996 | princessluv8996@email.com | 600017 |
| 3437317 | sk8ter4l!f3 | sk8ter4l!f3@email.com | 583 |

| **messages** | | | | | |
| --- | --- | --- | --- | --- | --- |
| **message\_id** | **from\_user\_id** | **to\_user\_id** | **date\_sent** | **date\_read** | **message** |
| 1 | 3437317 | 2138102 | 2015-03-15 00:02:17 | 2015-03-15 00:03:05 | “Hi!!! Wanted to know how u were doing?! Miss you !! ...” |
| 2 | 1438443 | 5937440 | 2015-03-15 00:02:24 | NULL | “Don’t forget to like and comment on my new pic...” |
| 3 | 2331234 | 1231232 | 2015-03-15 00:02:25 | 2015-04-01 00:11:08 | “Let’s hang out!” |

### **Question 5: Write a SQL query to determine which users have more than 10 unread messages.**

| SELECT  from\_user\_id,  COUNT(\*) AS unread\_messages\_count  FROM messages  WHERE date\_read IS NILL  GROUP BY from\_user\_id  HAVING COUNT(\*) > 10; |
| --- |