HuskyGram Sample Queries.

```sql
USE HuskyGram;

-- Give the image url of all images uploaded by user Nick.

SELECT image_url
FROM users u
        JOIN photos p ON p.user_id = u.id
WHERE u.username = 'Nick';


-- Which users follow user Nick?
-- Give the names of their followers.
SELECT u.username
FROM follows f
   JOIN users u ON u.id = f.follower_id
   JOIN users followee ON followee.id = f.followee_id
WHERE followee.username = 'Nick';

-- Give the user names of any user that has
-- tagged a photo with #huskylife.

SELECT distinct u.username
FROM users u
   JOIN photos p on u.id = p.user_id
   JOIN photo_tags pt ON p.id = pt.photo_id
   JOIN tags t ON pt.tag_id = t.id
WHERE t.tag_name = '#NEU';


-- Provide a list of tag names associated with
-- any photo that has been liked by SillyCat123.

SELECT distinct (t.tag_name)
FROM users u
   JOIN likes l ON u.id = l.user_id
   JOIN photo_tags pt on l.photo_id = pt.photo_id
   JOIN tags t on pt.tag_id = t.id
WHERE u.username = 'Mark';
```

```sql
-- #1 Provide a list of all users with all of each users'
followers.
-- The result should contain two columns: one named userName and
the other
-- named followerName, and it should be sorted by userName then by
followerName for each user.

SELECT u.username AS userName, uf.username AS followerName
FROM users u
        JOIN follows f ON u.id = f.followee_id
        JOIN users uf ON uf.id = f.follower_id
ORDER BY u.username, uf.username;


-- #2 Provide a count of the number of photos that have more than 3
-- comments and 2 or more tags. Your query should return a single
-- column named numPhotos with a single row containing the count.

SELECT COUNT(*) AS numPhotos
FROM photos p
WHERE (SELECT COUNT(*) FROM comments c WHERE c.photo_id = p.id) > 3
 AND (SELECT COUNT(*) FROM photo_tags pt WHERE pt.photo_id = p.id) >=
2;


-- #3 What are the three most popular tags?  A tag's popularity is
defined as the
-- number of photos to which a particular tag is applied.  The
result of your
-- query should contain the tag name (aliased as tagName) and
number of photos
-- that it has been applied to (aliased as numPhotos) sorted in
descending order
-- based on numPhotos (most popular tag at the top of the list).
-- Simplifying assumptions:  Assume that all tags are applied to a
-- different number of photos.



SELECT t.tag_name AS tagName, COUNT(pt.photo_id) AS numPhotos
FROM tags t
    JOIN photo_tags pt ON t.id = pt.tag_id
GROUP BY tagName
```

```
ORDER BY numPhotos DESC
LIMIT 3;


-- #4 How many comments have been written about each photo?  The
resulting
-- table should contain the photo's id number, URL of the photo,
and number
-- of comments.  The results should be in reverse chronological
order based
-- on when the photo was uploaded (created_at attribute).


SELECT p.id, p.image_url, COUNT(c.comment_text) AS numComments
FROM photos p
        JOIN comments c ON p.id = c.photo_id
GROUP BY p.id, p.image_url, p.created_at
ORDER BY p.created_at DESC;


-- #5 Which users have uploaded photos that have not yet been
"liked"
-- but have received comments? The resulting table should include a
-- list of usernames sorted alphabetically.


SELECT DISTINCT u.username
FROM users u
        JOIN photos p ON u.id = p.user_id
        JOIN comments c ON p.id = c.photo_id
        LEFT JOIN likes l ON p.id = l.photo_id
WHERE l.photo_id IS NULL
ORDER BY u.username;


-- #6  How many photos have been tagged with both
-- #NEU and #BU? The resulting table should have a
-- single column named "num_photos".

SELECT COUNT(DISTINCT pt1.photo_id) AS num_photos
FROM photo_tags pt1
        JOIN tags t1 ON pt1.tag_id = t1.id
        JOIN photo_tags pt2 ON pt1.photo_id = pt2.photo_id
```

```
        JOIN tags t2 ON pt2.tag_id = t2.id
WHERE t1.tag_name = '#NEU'
 AND t2.tag_name = '#BU';

-- #7 Provide a list of all comments that contain the substring
-- "college".  The search should be case insensitive.  The results
-- should contain the name of the commenter and the text of the
-- comment,  and the data should be sorted in ascending order by
-- the date the comment was made.

SELECT u.username, c.comment_text
FROM comments c
        JOIN users u ON c.user_id = u.id
WHERE c.comment_text LIKE '%college%'
ORDER BY c.created_at;

-- #8 Which users have not uploaded any photos?  The resulting
-- table should contain a list of user names sorted alphabetically
-- as well as when they joined the platform (create_at field in the
-- users table).  Use appropriate column names in the result.


SELECT u.username AS user_name, u.created_at AS joined_at
FROM users u
        LEFT JOIN photos p ON u.id = p.user_id
WHERE p.id IS NULL
ORDER BY u.username;

SELECT *
FROM users u
WHERE u.id NOT IN (SELECT DISTINCT user_id FROM photos);
```