User class

Most of the examples below assume you have the following class, to represent the structure of a user node:

```
public class User
{
    public long Id { get; set; }
    public string Name { get; set; }
    public int Age { get; set; }
    public string Email { get; set; }
}
```

Get all users by label

```
MATCH (user:User)
RETURN user
```

Get specific user

```
MATCH (user:User)
WHERE user.Id = 1234
RETURN user
```

Get a user, and the count of their friends

```
OPTIONAL MATCH (user:User)-[FRIENDS_WITH]-(friend:User)
WHERE user.Id = 1234
RETURN user, count(friend) AS NumberOfFriends
```

Get a user, and all their friends

```
OPTIONAL MATCH (user:User)-[FRIENDS_WITH]-(friend:User)
WHERE user.Id = 1234
RETURN user, collect(friend) AS NumberOfFriends
```

Create a user

```
CREATE (user:User { Id: 456, Name: 'Jim' })
```

Should use parameters:

```
CREATE (user:User {newUser})
```

Create a user, only if they don't already exist

```
MERGE (user:User { Id: 456 })
ON CREATE user
SET user.Name = 'Jim'
```

Should use parameters:

```
CREATE (user:User { Id: {id} })
ON CREATE user
SET user = {newUser}
```

Create a user and relate them to an existing one

```
MATCH (invitee:User)
WHERE invitee.Id = 123
CREATE invitee-[:INVITED]->(invited:User {newUser})
```

Relate two existing users

```
MATCH (user1:User), (user2:User)
WHERE user1.Id = 123, user2.Id = 456
CREATE user1-[:FRIENDS_WITH]->user2
```

Relate two existing users, only if they aren't related already

```
MATCH (user1:User), (user2:User)
WHERE user1.Id = 123, user2.Id = 456
CREATE UNIQUE user1-[:FRIENDS_WITH]->user2
```

Update a single property on a user

```
MATCH (user:User)
WHERE user.Id = 123
SET user.Age = 25
```

Replace all the properties on a user

```
MATCH (user:User)
WHERE user.Id = 123
SET user = { Id: 123, Age: 25, Email: 'tatham@oddie.com.au' }
```

Delete a user

```
MATCH (user:User)
WHERE user.Id = 123
DELETE user
```

Delete a user and all inbound relationships

```
OPTIONAL MATCH (user:User)<-[r]-()
WHERE user.Id = 123
DELETE r, user
```

Get all labels for a specific user

```
MATCH (user:User)
WHERE user.Id = 1234
RETURN labels(user)
```

Get all labels for a specific user, and still the user too

```
MATCH (user:User)
WHERE user.Id = 1234
RETURN user, labels(user)
```

Get a user, count their friends then add this number to the user and return.

Note: This is an example of using Neo4j 3.0 Stored Procedures.
There are other ways of adding a property to an object, this is just an example of CALL and YIELD, using apoc Neo4j Stored Procedures

```
MATCH (user:User)
WHERE user.Id = 1234
WITH user, size((user)-[:IS_FRIENDS_WITH]->(:Friend)) as numberOfFriends
CALL apoc.map.setKey(user, 'numberOfFriends', numberOfFriends) YIELD value
AS userWithFriends
RETURN userWithFriends
```