# Lecture #2

NEWM N510: Web-Database Concepts

## MySQL (1)

kharrazi@iupui.edu
http://www.info510.com

## Review Last Lecture

- Networking Overview
- Web Server
- Static Languages - HTML
- Server Side Languages – PHP
- Databases - MySQL
- HTML/PHP/MySQL Integration
- Course Project

## Lecture in a Nutshell

1. Database Overview
2. Relational Databases
3. Installing MySQL
4. Command line MySQL
5. MySQL GUI Tools
6. SQL Introduction
7. SQL: SELECT
8. SQL: WHERE

## 1. Database Overview

- Different types of Database structures (Hierarchical, Relational, Temporal) are based on the way they store the data on Hard Disk Drive and how they read from the stored data.

- Famous Relational Databases: Oracle, MS SQL (Microsoft), DB2 (IBM), MySQL, mSQL, Postgre SQL and etc.

- MySQL is an open source relational database management system (RDBMS) that uses Structured Query Language (SQL), the most popular language for adding, accessing, and processing data in a database.

## 2. Relational Databases

- RDBMS (Relational Database Management System)
- RDBMSs can provide faster access to data than flat files.
- RDBMSs can be easily queried (SQL Language) to extract sets of data that fit certain criteria.
- RDBMSs have built-in mechanisms for dealing with concurrent access so that you as a programmer don't have to worry about it.
- RDBMSs have built-in privilege systems.

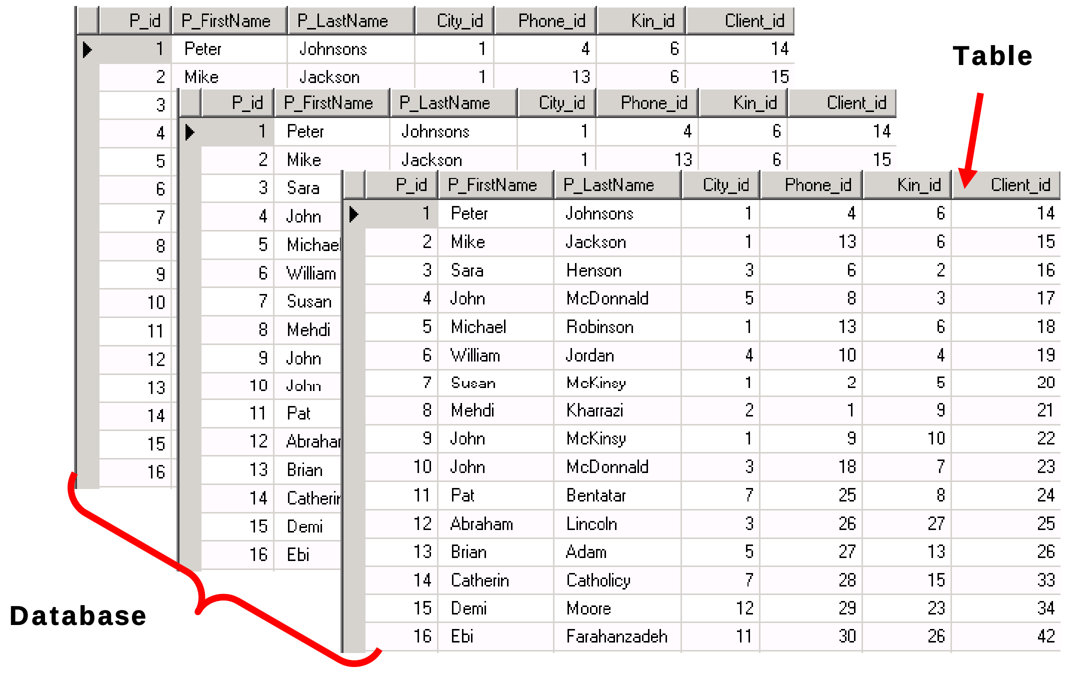
## *Relational Databases (cont.)*

- Relational databases are made up of relations, more commonly called tables.

- A table is exactly what it sounds like a table of data. If you've used an electronic spreadsheet (Excel), you've already used a relational table.

- A database usually consists of several tables.

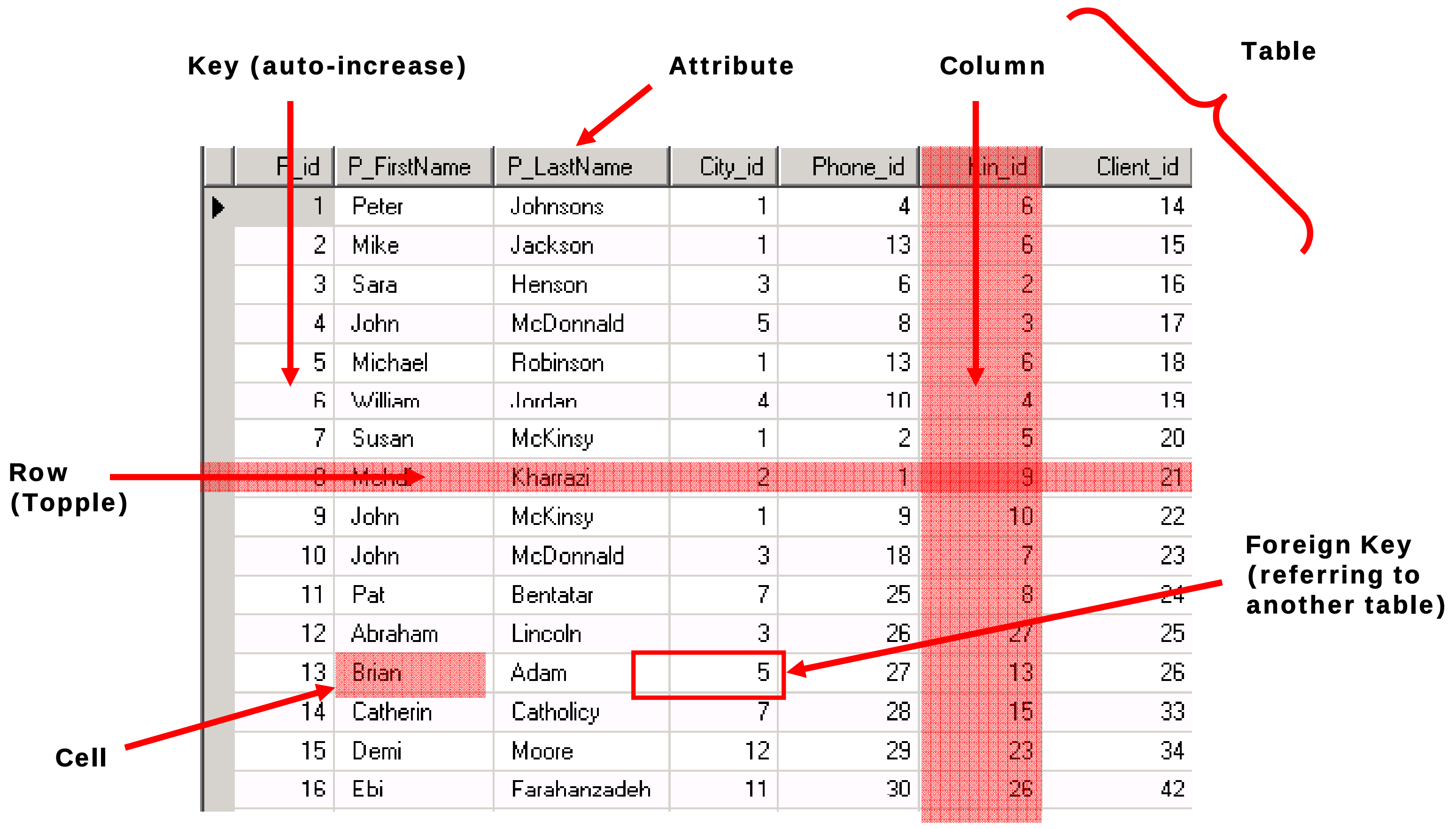
- MySQL can handle thousands of databases.



| P_id | P_FirstName | P_LastName | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 |
| 2 | Mike | Jackson | 1 | 13 | 6 | 15 |
| 3 | Sara | Henson | 3 | 6 | 2 | 16 |
| 4 | John | McDonnald | 5 | 8 | 3 | 17 |
| 5 | Michael | Robinson | 1 | 13 | 6 | 18 |
| 6 | William | Jordan | 4 | 10 | 4 | 19 |
| 7 | Susan | McKinsy | 1 | 2 | 5 | 20 |
| 8 | Mehdi | Kharrazi | 2 | 1 | 9 | 21 |
| 9 | John | McKinsy | 1 | 9 | 10 | 22 |
| 10 | John | McDonnald | 3 | 18 | 7 | 23 |
| 11 | Pat | Bentatar | 7 | 25 | 8 | 24 |
| 12 | Abraham | Lincoln | 3 | 26 | 27 | 25 |
| 13 | Brian | Adam | 5 | 27 | 13 | 26 |
| 14 | Catherin | Catholicy | 7 | 28 | 15 | 33 |
| 15 | Demi | Moore | 12 | 29 | 23 | 34 |
| 16 | Ebi | Farahanzadeh | 11 | 30 | 26 | 42 |

## *Relational Databases (cont.)*

- Elements of the relational database table:



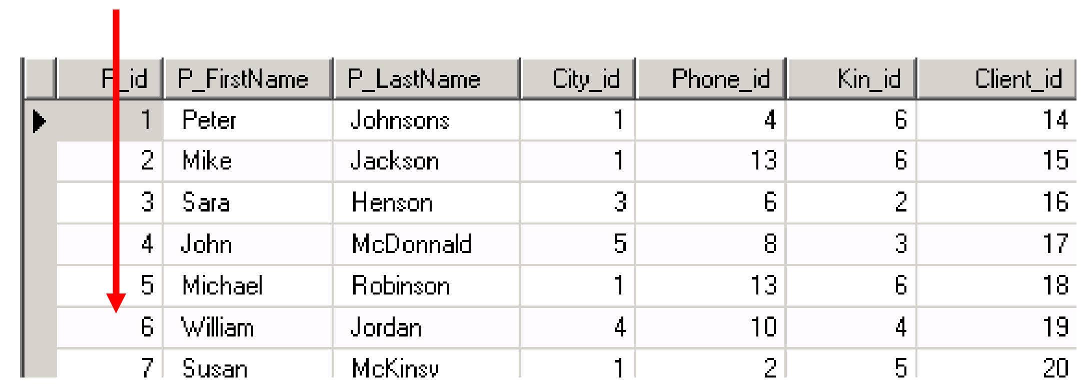
| P_id | P_FirstName | P_LastName | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 |
| 2 | Mike | Jackson | 1 | 13 | 6 | 15 |
| 3 | Sara | Henson | 3 | 6 | 2 | 16 |
| 4 | John | McDonnald | 5 | 8 | 3 | 17 |
| 5 | Michael | Robinson | 1 | 13 | 6 | 18 |
| 6 | William | Jordan | 4 | 10 | 4 | 19 |
| 7 | Susan | McKinsy | 1 | 2 | 5 | 20 |
| 8 | Mehdi | Kharrazi | 2 | 1 | 9 | 21 |
| 9 | John | McKinsy | 1 | 9 | 10 | 22 |
| 10 | John | McDonnald | 3 | 18 | 7 | 23 |
| 11 | Pat | Bentatar | 7 | 25 | 8 | 24 |
| 12 | Abraham | Lincoln | 3 | 26 | 27 | 25 |
| 13 | Brian | Adam | 5 | 27 | 13 | 26 |
| 14 | Catherin | Catholicy | 7 | 28 | 15 | 33 |
| 15 | Demi | Moore | 12 | 29 | 23 | 34 |
| 16 | Ebi | Farahanzadeh | 11 | 30 | 26 | 42 |

## *Relational Databases (cont.)*

- **<u>Keys</u>**

  - Keys are special fields
  - Keys are defined on table creation
  - Keys tie tables together
  - Keys are unique: no two records have same value of the key
  - Primary key: Unique and links two tables → e.g. SSN number
  - Only one primary key per table

**Key (auto-increase)**

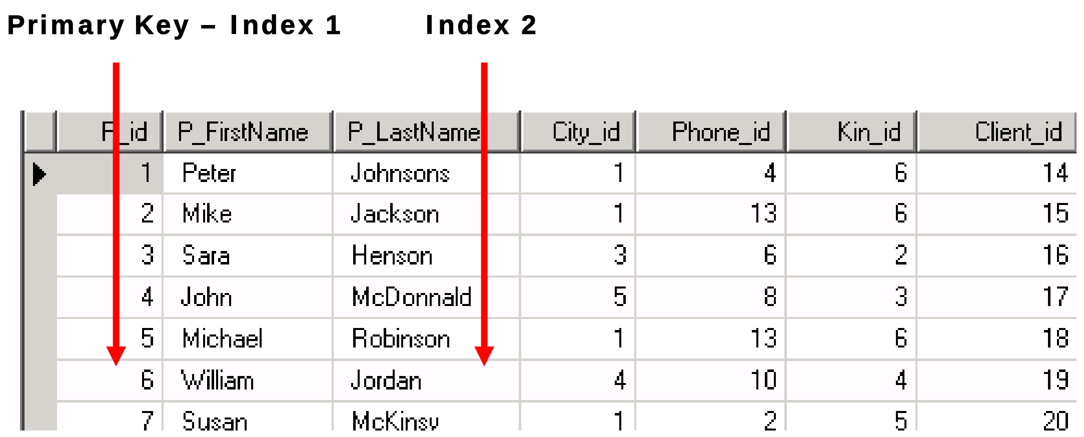


| P_id | P_FirstName | P_LastName | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 |
| 2 | Mike | Jackson | 1 | 13 | 6 | 15 |
| 3 | Sara | Henson | 3 | 6 | 2 | 16 |
| 4 | John | McDonnald | 5 | 8 | 3 | 17 |
| 5 | Michael | Robinson | 1 | 13 | 6 | 18 |
| 6 | William | Jordan | 4 | 10 | 4 | 19 |
| 7 | Susan | McKinsy | 1 | 2 | 5 | 20 |

## ***Relational Databases (cont.)***

- **<u>Indexes</u>**

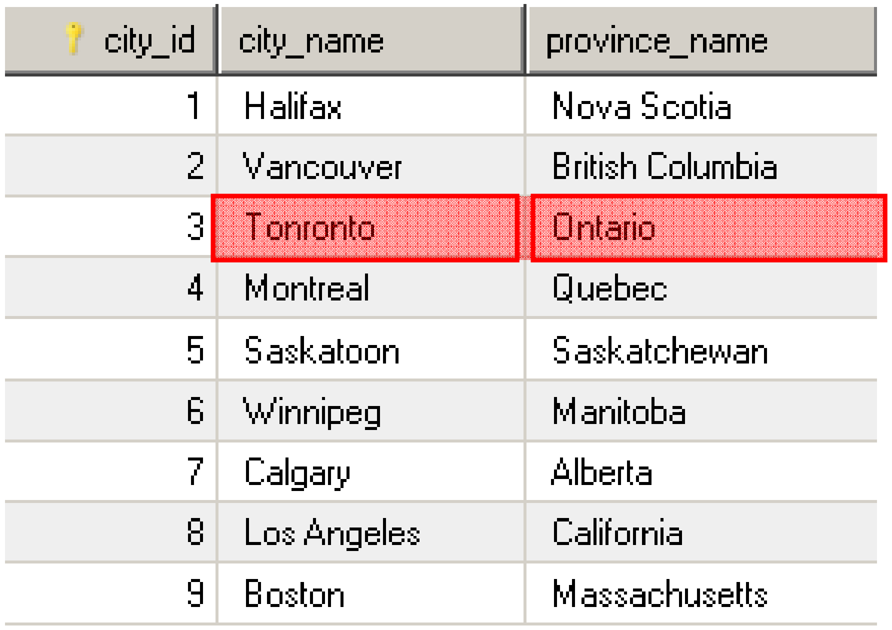
▪ Similar to the index of a book
▪ MySQL automatically <u>creates an index for each primary key</u>
▪ Indexes make it a lot faster to retrieve results
▪ User can define additional indexes



| P_id | P_FirstName | P_LastName | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 |
| 2 | Mike | Jackson | 1 | 13 | 6 | 15 |
| 3 | Sara | Henson | 3 | 6 | 2 | 16 |
| 4 | John | McDonnald | 5 | 8 | 3 | 17 |
| 5 | Michael | Robinson | 1 | 13 | 6 | 18 |
| 6 | William | Jordan | 4 | 10 | 4 | 19 |
| 7 | Susan | McKinsy | 1 | 2 | 5 | 20 |

## ***Relational Databases (cont.)***

- Internal Key (1 to 1 relationship):

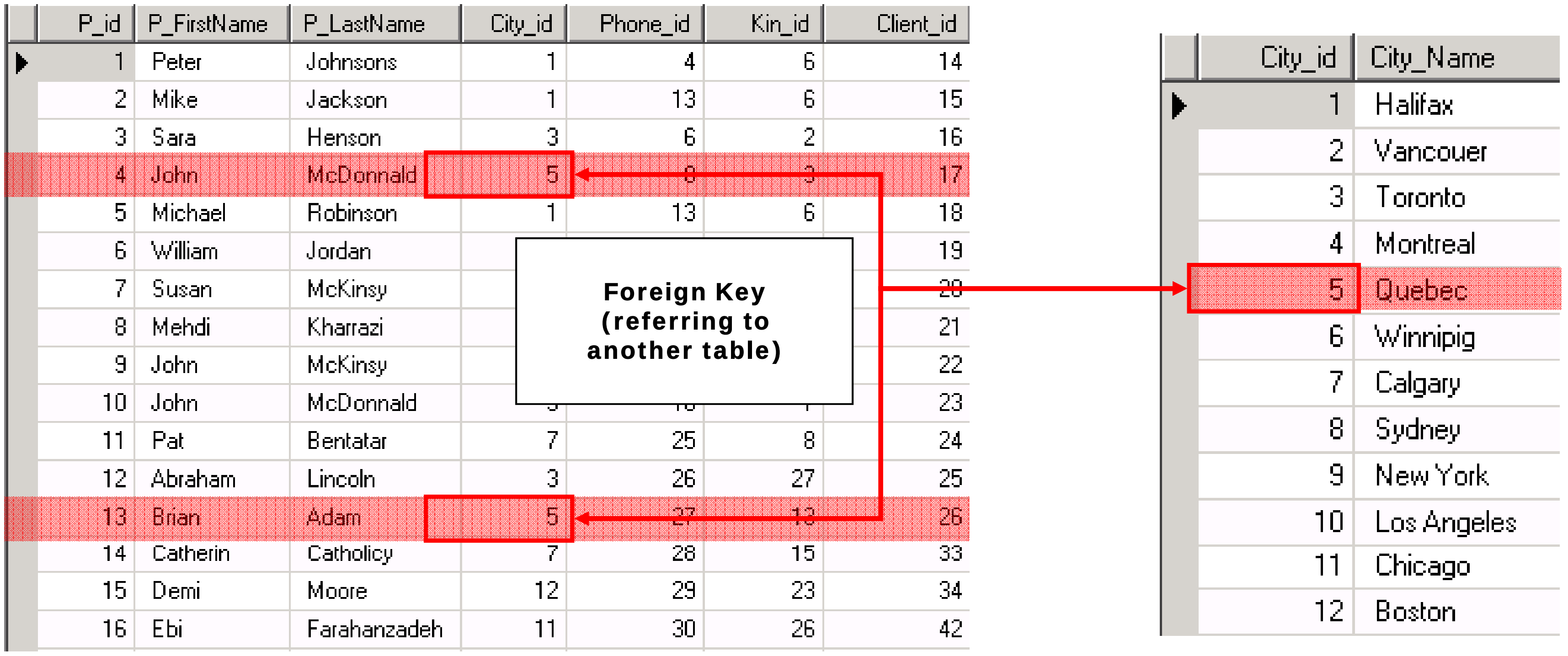


| city_id | city_name | province_name |
|---|---|---|
| 1 | Halifax | Nova Scotia |
| 2 | Vancouver | British Columbia |
| 3 | Tonronto | Ontario |
| 4 | Montreal | Quebec |
| 5 | Saskatoon | Saskatchewan |
| 6 | Winnipeg | Manitoba |
| 7 | Calgary | Alberta |
| 8 | Los Angeles | California |
| 9 | Boston | Massachusetts |

**Table: City_information (ONE-ONE)**

## *Relational Databases (cont.)*

- Foreign Key (1 to Many relationship):



| P_id | P_FirstName | P_LastName | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 |
| 2 | Mike | Jackson | 1 | 13 | 6 | 15 |
| 3 | Sara | Henson | 3 | 6 | 2 | 16 |
| 4 | John | McDonnald | 5 | 8 | 3 | 17 |
| 5 | Michael | Robinson | 1 | 13 | 6 | 18 |
| 6 | William | Jordan | [illegible] | [illegible] | [illegible] | 19 |
| 7 | Susan | McKinsy | [illegible] | [illegible] | [illegible] | 20 |
| 8 | Mehdi | Kharrazi | [illegible] | [illegible] | [illegible] | 21 |
| 9 | John | McKinsy | [illegible] | [illegible] | [illegible] | 22 |
| 10 | John | McDonnald | [illegible] | [illegible] | [illegible] | 23 |
| 11 | Pat | Bentatar | 7 | 25 | 8 | 24 |
| 12 | Abraham | Lincoln | 3 | 26 | 27 | 25 |
| 13 | Brian | Adam | 5 | 27 | 13 | 26 |
| 14 | Catherin | Catholicy | 7 | 28 | 15 | 33 |
| 15 | Demi | Moore | 12 | 29 | 23 | 34 |
| 16 | Ebi | Farahanzadeh | 11 | 30 | 26 | 42 |

**Foreign Key (referring to another table)**

| City_id | City_Name |
|---|---|
| 1 | Halifax |
| 2 | Vancouer |
| 3 | Toronto |
| 4 | Montreal |
| 5 | Quebec |
| 6 | Winnipig |
| 7 | Calgary |
| 8 | Sydney |
| 9 | New York |
| 10 | Los Angeles |
| 11 | Chicago |
| 12 | Boston |

**Table: Patient_information (MANY)**

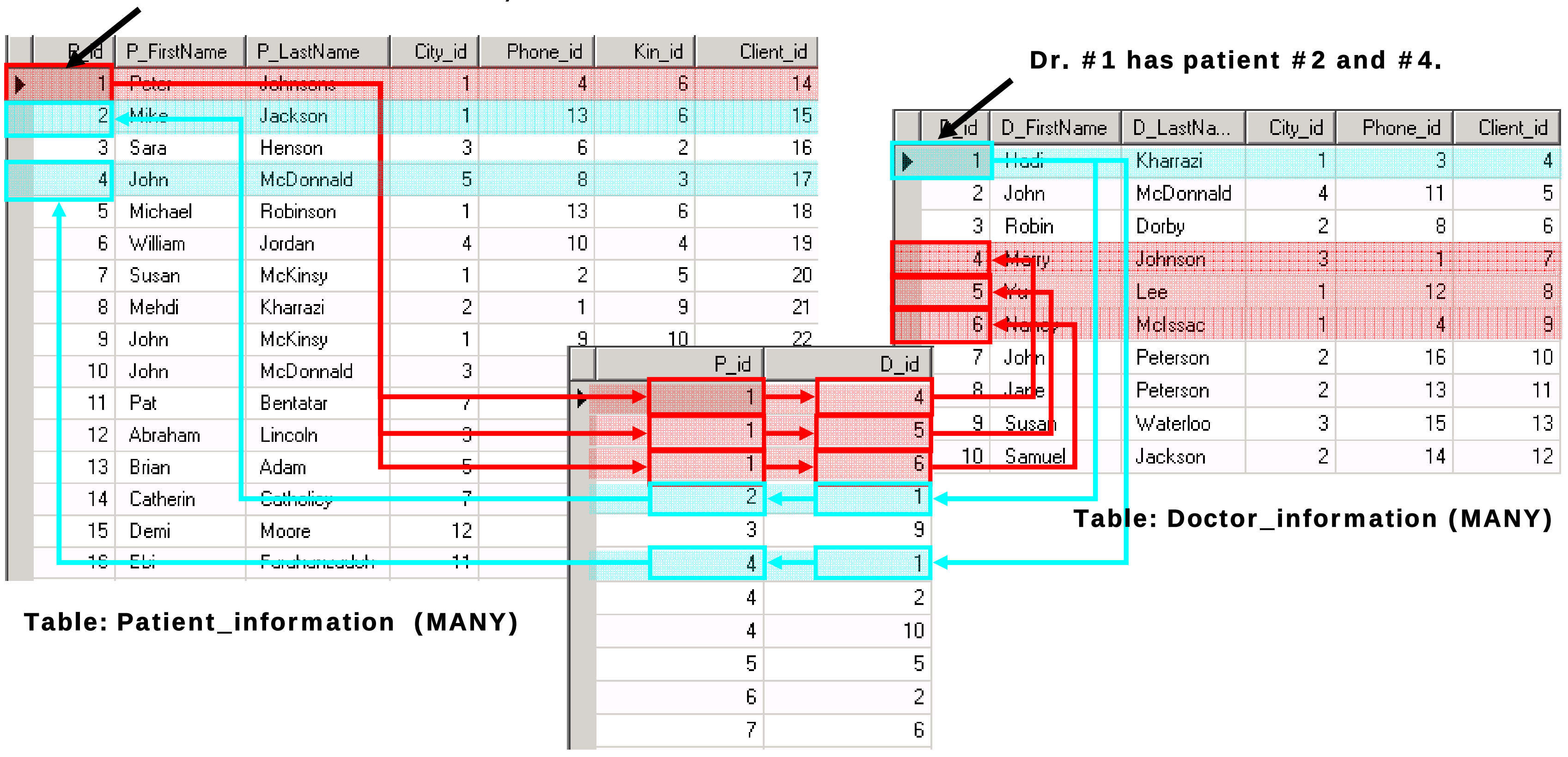
**Table: City_information (ONE)**

## *Relational Databases (cont.)*

- Foreign Key (Many to Many relationship):

**Patient #1 has doctor #4, #5 and #6.**

**Dr. #1 has patient #2 and #4.**



| P_id | P_FirstName | P_LastName | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 |
| 2 | Mike | Jackson | 1 | 13 | 6 | 15 |
| 3 | Sara | Henson | 3 | 6 | 2 | 16 |
| 4 | John | McDonnald | 5 | 8 | 3 | 17 |
| 5 | Michael | Robinson | 1 | 13 | 6 | 18 |
| 6 | William | Jordan | 4 | 10 | 4 | 19 |
| 7 | Susan | McKinsy | 1 | 2 | 5 | 20 |
| 8 | Mehdi | Kharrazi | 2 | 1 | 9 | 21 |
| 9 | John | McKinsy | 1 | 9 | 10 | 22 |
| 10 | John | McDonnald | 3 | | | |
| 11 | Pat | Bentatar | 7 | | | |
| 12 | Abraham | Lincoln | 3 | | | |
| 13 | Brian | Adam | 5 | | | |
| 14 | Catherin | Catholicy | 7 | | | |
| 15 | Demi | Moore | 12 | | | |
| 16 | Ebi | Farahanizadeh | 11 | | | |

**Table: Patient_information (MANY)**

| P_id | D_id |
|---|---|
| 1 | 4 |
| 1 | 5 |
| 1 | 6 |
| 2 | 1 |
| 3 | 9 |
| 4 | 1 |
| 4 | 2 |
| 4 | 10 |
| 5 | 5 |
| 6 | 2 |
| 7 | 6 |

**Table: Patient_Doctor_realationship**

| D_id | D_FirstName | D_LastNa... | City_id | Phone_id | Client_id |
|---|---|---|---|---|---|
| 1 | Hadi | Kharrazi | 1 | 3 | 4 |
| 2 | John | McDonnald | 4 | 11 | 5 |
| 3 | Robin | Dorby | 2 | 8 | 6 |
| 4 | Marry | Johnson | 3 | 1 | 7 |
| 5 | Yu | Lee | 1 | 12 | 8 |
| 6 | Nancy | McIssac | 1 | 4 | 9 |
| 7 | John | Peterson | 2 | 16 | 10 |
| 8 | Jane | Peterson | 2 | 13 | 11 |
| 9 | Susan | Waterloo | 3 | 15 | 13 |
| 10 | Samuel | Jackson | 2 | 14 | 12 |

**Table: Doctor_information (MANY)**

***Relational Databases (cont.)***

## ***Database Normalization***

In the field of relational database design, normalization is a systematic way of ensuring that a database structure is suitable for general-purpose querying and free of certain undesirable characteristics — insertion, update, and deletion anomalies — that could lead to a loss of data integrity

Atomic data → smallest piece of data that can't or shouldn't be divided. The decision to consider a piece of information as atomic or not depends on the context and decision of the database designer:

Pizza delivery: order_id, address (includes house_number and street_name)

Real estate agent: mls_id, house_number, street_name

*(Real estate agent may want to know the houses on sale on one street)*

## ***Relational Databases (cont.)*** (1NF)

**1NF** → First normal form sets the basic rules for a database:

- There's no top-to-bottom ordering to the rows
- There's no left-to-right ordering to the columns
- There are no duplicate rows
- All columns are regular [i.e. rows have no hidden components such as row IDs, object IDs, or hidden timestamps]

- Some approaches to 1NF:
  - Eliminate duplicative columns from the same table.
  - Create separate tables for each group of related data and identify each row with a unique column or set of columns (primary key).

- Example:
  - Suppose a novice designer wishes to record the names and diagnosis of patients in a table.

## ***Relational Databases (cont.)*** (1NF)

- The table can be initially defined as:

| **p_id** | patient_name | patient_diagnosis |
|---|---|---|
| 1 | Mike | Diabetes |
| 2 | Sara | Asthma |
| 3 | Peter | Migraine |
| 4 | Brian | Arthritis |

- The designer then becomes aware of a requirement to record multiple diagnosis for some patients:

| **p_id** | patient_name | patient_diagnoses |
|---|---|---|
| 1 | Mike | Diabetes |
| 2 | Sara | Asthma<br>Multiple Sclerosis |
| 3 | Peter | Migraine<br>Diabetes<br>Chronic Fatigue |
| 4 | Brian | Arthritis |

## ***Relational Databases (cont.)*** (1NF)

- The designer might attempt to get around this restriction by repeating groups across columns

| **p_id** | patient_name | patient_diagnosis_1 | patient_diagnosis_2 | patient_diagnosis_3 |
|---|---|---|---|---|
| 1 | Mike | Diabetes | | |
| 2 | Sara | Asthma | Multiple Sclerosis | |
| 3 | Peter | Migraine | Diabetes | Chronic Fatigue |
| 4 | Brian | Arthritis | | |

- The designer might attempt to get around this restriction by Repeating groups within columns

| **p_id** | patient_name | patient_diagnoses |
|---|---|---|
| 1 | Mike | Diabetes |
| 2 | Sara | Asthma, Multiple Sclerosis |
| 3 | Peter | Migraine, Diabetes, Chronic Fatigue |
| 4 | Brian | Arthritis |

## ***Relational Databases (cont.)*** (1NF)

- The designer normalized the table (1NF) by adding a row for each diagnosis.

| **p_id** | patient_name | patient_diagnosis |
|---|---|---|
| 1 | Mike | Diabetes |
| 2 | Sara | Asthma |
| 2 | Sara | Multiple Sclerosis |
| 3 | Peter | Migraine |
| 3 | Peter | Diabetes |
| 3 | Peter | Chronic Fatigue |
| 4 | Brian | Arthritis |

## ***Relational Databases (cont.)*** (1NF)

- A design that is unambiguously in 1NF makes use of two tables: a Patient Name table and a Patient Diagnosis table.

| **patient_id** | patient_name |
|---|---|
| 1 | Mike |
| 2 | Sara |
| 3 | Peter |
| 4 | Brian |

| **patient_id** | patient_diagnosis |
|---|---|
| 1 | Diabetes |
| 2 | Asthma |
| 2 | Multiple Sclerosis |
| 3 | Migraine |
| 3 | Diabetes |
| 3 | Chronic Fatigue |
| 4 | Arthritis |

## ***Relational Databases (cont.)*** (2NF)

**2NF** → Second normal form further addresses the concept of removing duplicative data:

- Meet all the requirements of the first normal form.
- Eliminate redundancy from columns
- No non-key attributes should depend on a portion of the primary key

- <u>Some approaches to 2NF:</u>

o Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
o Create relationships between these new tables and their predecessors through the use of foreign keys.

- <u>Example:</u>

o Further normalizing the last example.

## *Relational Databases (cont.)* (2NF)

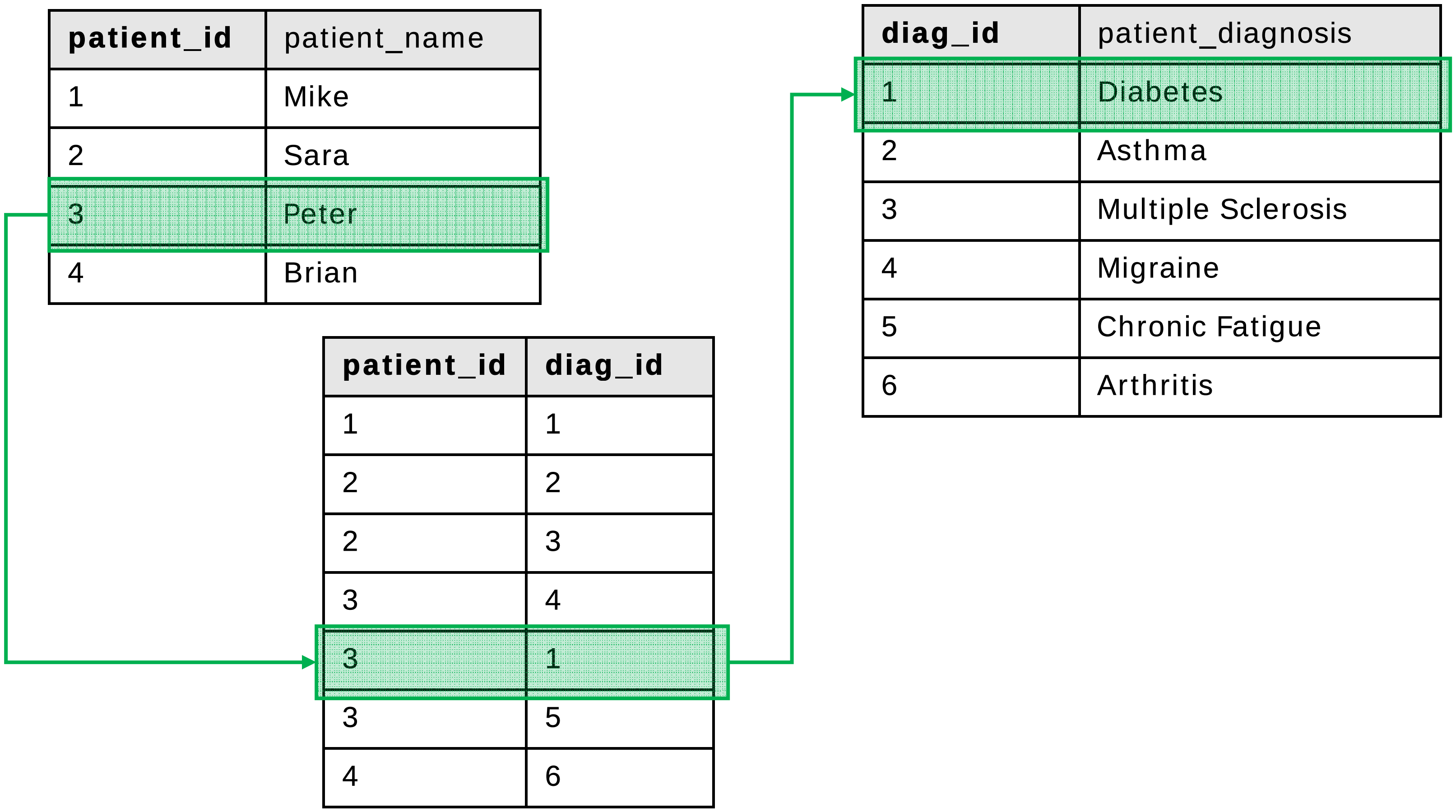
- Consider the following 1NF tables:

| **patient_id** | patient_name |
|---|---|
| 1 | Mike |
| 2 | Sara |
| 3 | Peter |
| 4 | Brian |

| **patient_id** | **diag_id** | patient_diagnosis |
|---|---|---|
| 1 | 1 | Diabetes |
| 2 | 2 | Asthma |
| 2 | 3 | Multiple Sclerosis |
| 3 | 4 | Migraine |
| 3 | 1 | Diabetes |
| 3 | 5 | Chronic Fatigue |
| 4 | 6 | Arthritis |

## ***Relational Databases (cont.)*** (2NF)

- A design that is unambiguously in 2NF makes use of multiple tables: a Patient Name table, a Diagnosis table and a Relational table.



| **patient_id** | patient_name |
|---|---|
| 1 | Mike |
| 2 | Sara |
| 3 | Peter |
| 4 | Brian |

| **patient_id** | **diag_id** |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 2 | 3 |
| 3 | 4 |
| 3 | 1 |
| 3 | 5 |
| 4 | 6 |

| **diag_id** | patient_diagnosis |
|---|---|
| 1 | Diabetes |
| 2 | Asthma |
| 3 | Multiple Sclerosis |
| 4 | Migraine |
| 5 | Chronic Fatigue |
| 6 | Arthritis |

## ***Relational Databases (cont.)*** (3NF)

**3NF** → Third normal form goes one large step further:

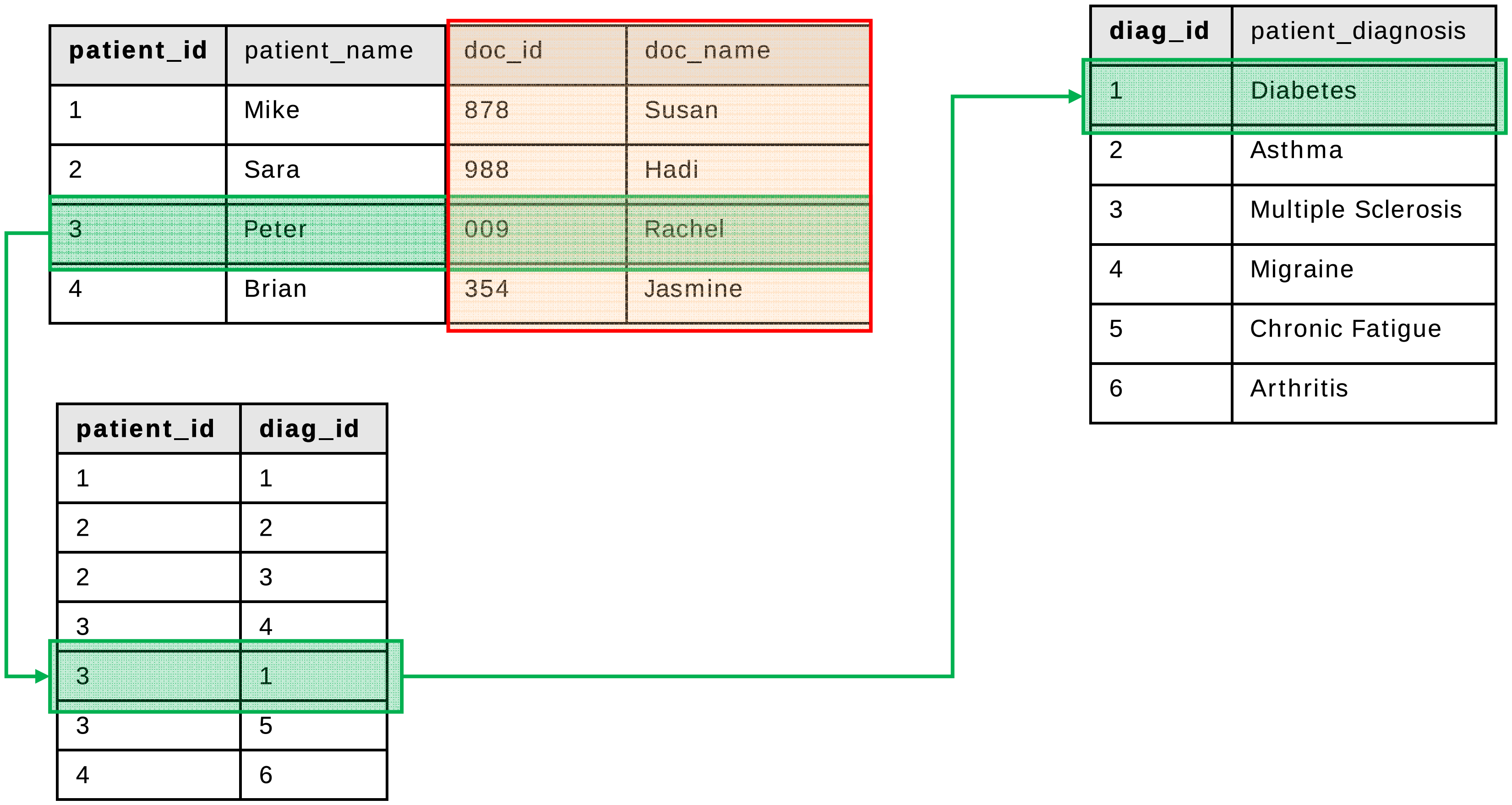
- Meet all the requirements of the second normal form.
- No attributes depend on other non-key attributes.

- Some approaches to 3NF:
  - Remove columns that are not dependent upon the primary key.

- Example:
  - Further normalizing the extended version of the last example.

## ***Relational Databases (cont.)*** (3NF)

- The patient table can be initially defined as:



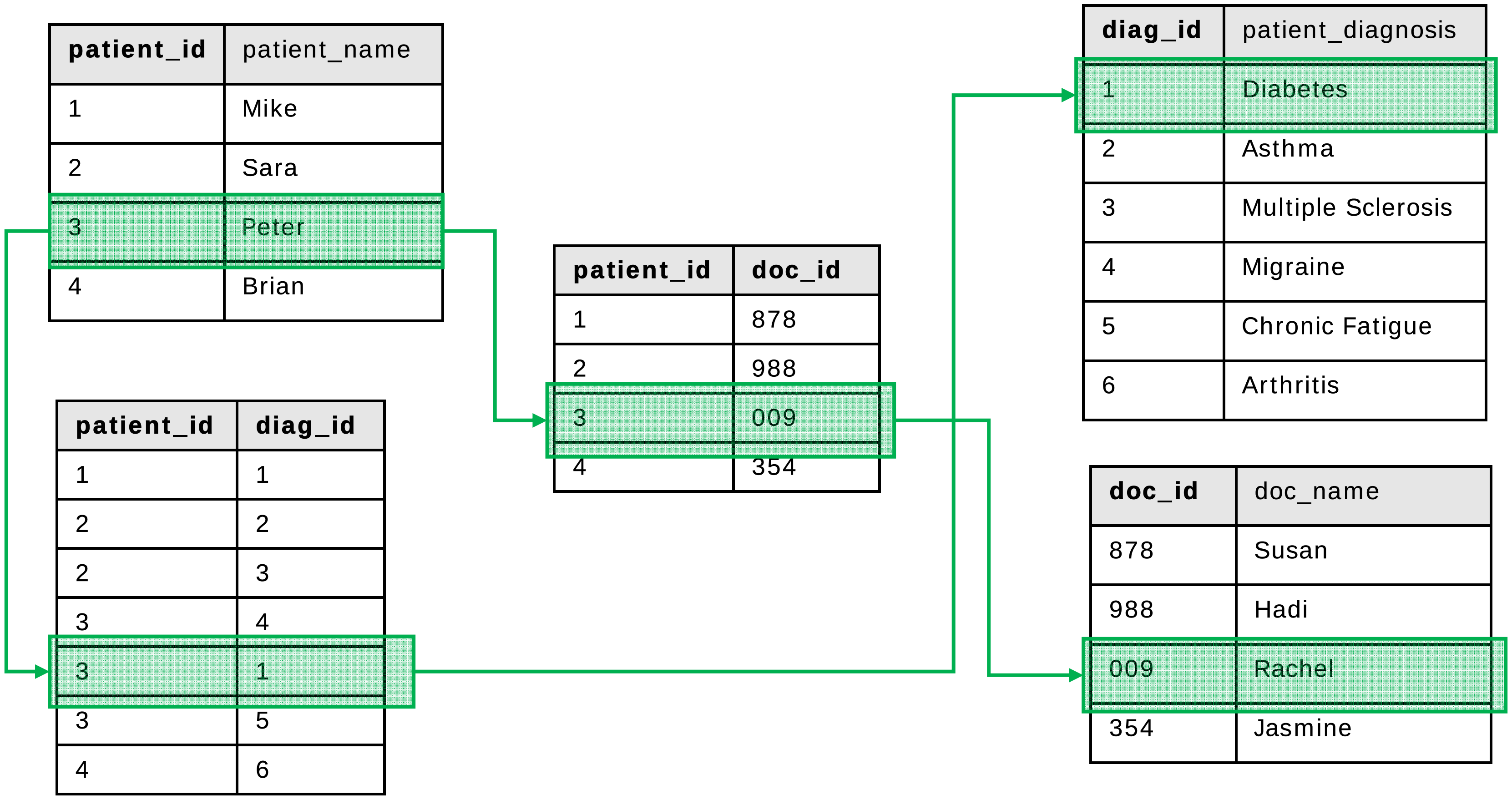
| **patient_id** | patient_name | doc_id | doc_name |
|---|---|---|---|
| 1 | Mike | 878 | Susan |
| 2 | Sara | 988 | Hadi |
| 3 | Peter | 009 | Rachel |
| 4 | Brian | 354 | Jasmine |

| **patient_id** | **diag_id** |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 2 | 3 |
| 3 | 4 |
| 3 | 1 |
| 3 | 5 |
| 4 | 6 |

| **diag_id** | patient_diagnosis |
|---|---|
| 1 | Diabetes |
| 2 | Asthma |
| 3 | Multiple Sclerosis |
| 4 | Migraine |
| 5 | Chronic Fatigue |
| 6 | Arthritis |

## ***Relational Databases (cont.)*** (3NF)

- Columns that are not dependent upon the primary key are removed:



| **patient_id** | patient_name |
|---|---|
| 1 | Mike |
| 2 | Sara |
| 3 | Peter |
| 4 | Brian |

| **patient_id** | **diag_id** |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 2 | 3 |
| 3 | 4 |
| 3 | 1 |
| 3 | 5 |
| 4 | 6 |

| **patient_id** | **doc_id** |
|---|---|
| 1 | 878 |
| 2 | 988 |
| 3 | 009 |
| 4 | 354 |

| **diag_id** | patient_diagnosis |
|---|---|
| 1 | Diabetes |
| 2 | Asthma |
| 3 | Multiple Sclerosis |
| 4 | Migraine |
| 5 | Chronic Fatigue |
| 6 | Arthritis |

| **doc_id** | doc_name |
|---|---|
| 878 | Susan |
| 988 | Hadi |
| 009 | Rachel |
| 354 | Jasmine |

## ***Relational Databases (cont.)*** (xNF)

**4NF** → Fourth normal form has one additional requirement:

- Meet all the requirements of the third normal form.
- A relation is in 4NF if it has no multi-valued dependencies.

**5NF** → Fifth normal form is sometimes seen and won't be discussed.

**6NF** → Fifth normal form is very rarely seen and won't be discussed.

## 3. Installing MySQL

- MySQL is already installed on the server; therefore there is no need to install it on your desktops.

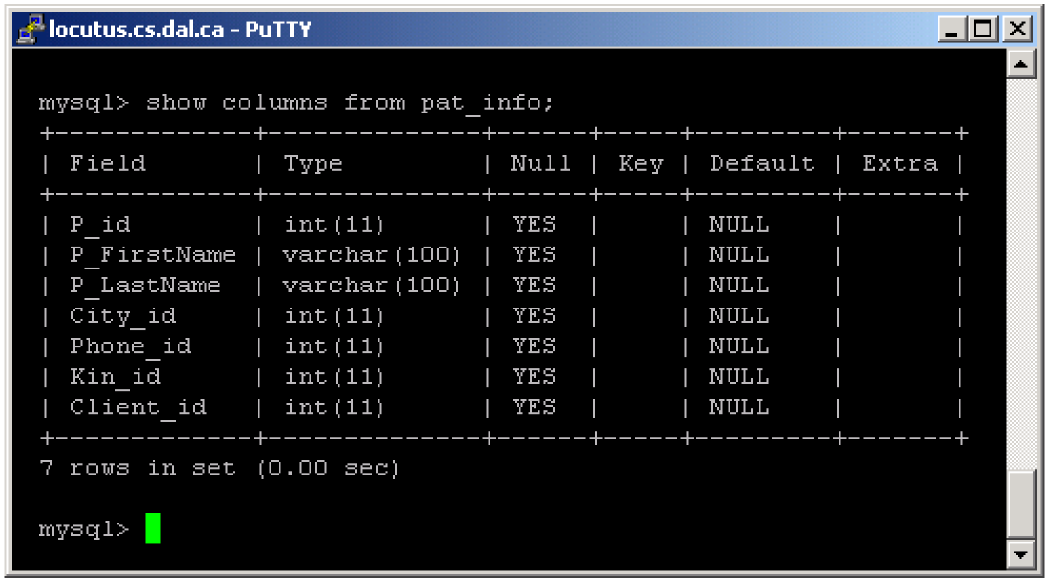
- In case you want to install a web server (IIS, Apache) on your desktop or laptop and then install MySQL server, you can find more information at:

  http://dev.mysql.com/doc/mysql/en/installing.html

- Some open source packages include the Apache server, PHP engine and MySQL server all together and installing them would install all of them together.

## 4. Command line MySQL

- There are 2 ways to communicate with the available MySQL server on the server:
  - Through a telnet (SSH) client such as PuTTY which will show everything in a command line style.

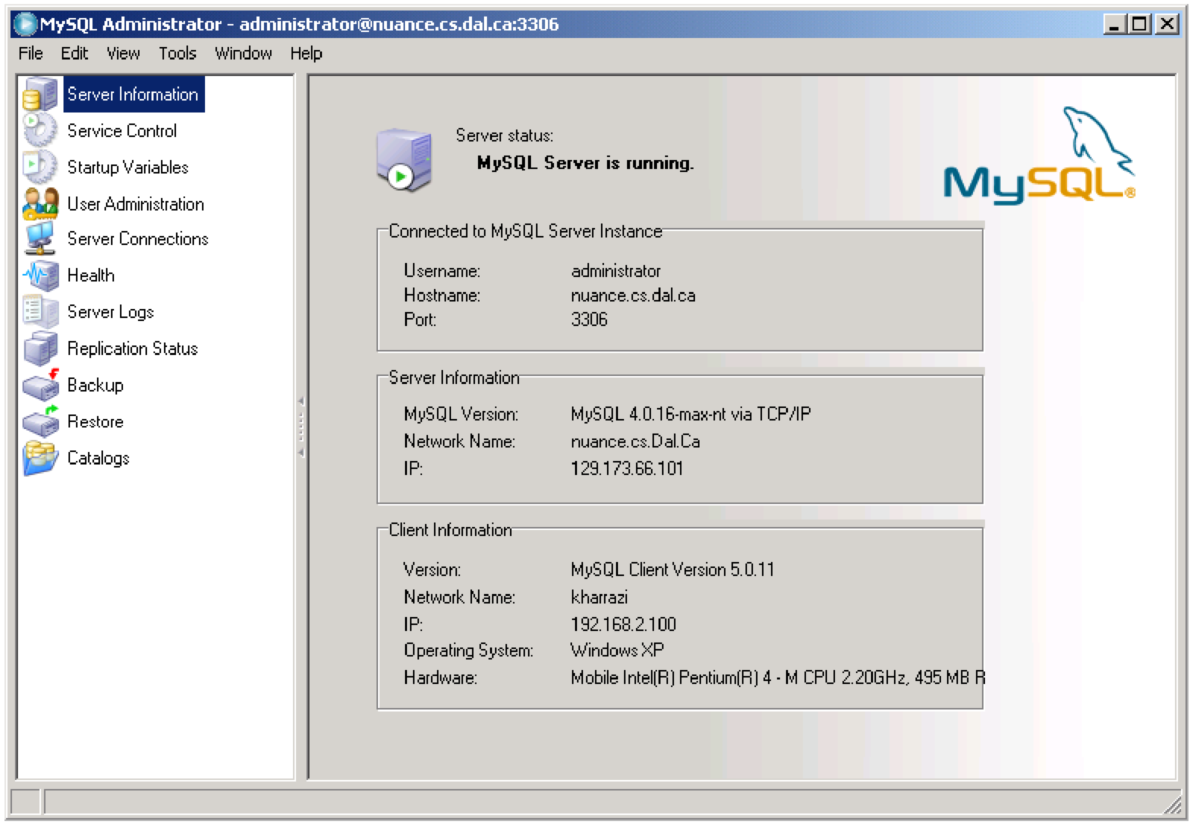


```
mysql> show columns from pat_info;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| P_id        | int(11)      | YES  |     | NULL    |       |
| P_FirstName | varchar(100) | YES  |     | NULL    |       |
| P_LastName  | varchar(100) | YES  |     | NULL    |       |
| City_id     | int(11)      | YES  |     | NULL    |       |
| Phone_id    | int(11)      | YES  |     | NULL    |       |
| Kin_id      | int(11)      | YES  |     | NULL    |       |
| Client_id   | int(11)      | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql>
```

  - Through one of the available Graphical User Interfaces made by MySQL.

## 5. MySQL GUI Tools

- MySQL Administrator:



Server status:
**MySQL Server is running.**

Connected to MySQL Server Instance

| | |
|---|---|
| Username: | administrator |
| Hostname: | nuance.cs.dal.ca |
| Port: | 3306 |

Server Information

| | |
|---|---|
| MySQL Version: | MySQL 4.0.16-max-nt via TCP/IP |
| Network Name: | nuance.cs.Dal.Ca |
| IP: | 129.173.66.101 |

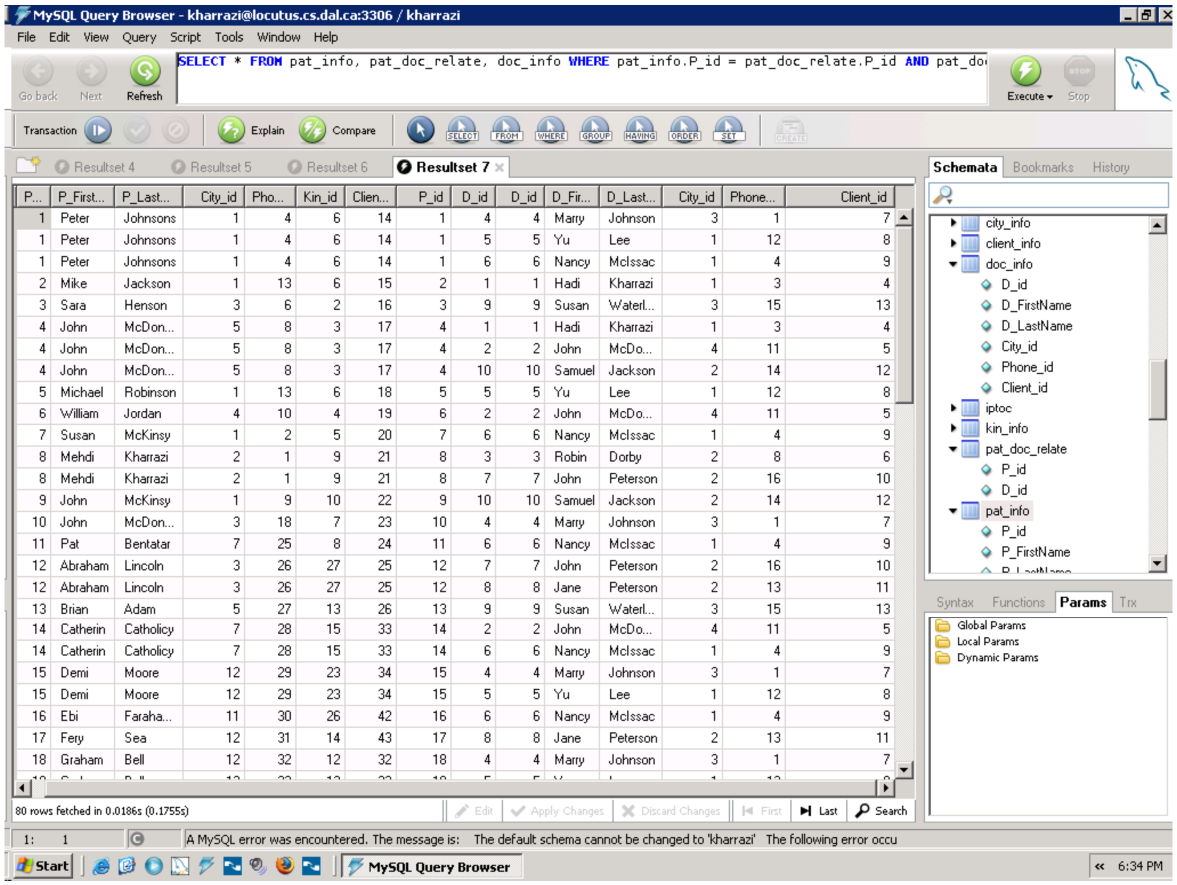
Client Information

| | |
|---|---|
| Version: | MySQL Client Version 5.0.11 |
| Network Name: | kharrazi |
| IP: | 192.168.2.100 |
| Operating System: | Windows XP |
| Hardware: | Mobile Intel(R) Pentium(R) 4 - M CPU 2.20GHz, 495 MB R |

**You don't have administrative privileges for the MySQL server.**

## ***MySQL GUI Tools (cont.)***

- MySQL Query Browser:


MySQL Query Browser - kharrazi@locutus.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Go back Next Refresh
Execute Stop
Transaction Explain Compare
SELECT FROM WHERE GROUP HAVING ORDER SET CREATE
Resultset 4 Resultset 5 Resultset 6 Resultset 7
Schemata Bookmarks History
city_info
client_info
doc_info
D_id
D_FirstName
D_LastName
City_id
Phone_id
Client_id
iptoc
kin_info
pat_doc_relate
P_id
D_id
pat_info
P_id
P_FirstName
Syntax Functions Params Trx
Global Params
Local Params
Dynamic Params
Edit Apply Changes Discard Changes First Last Search
Start
MySQL Query Browser
6:34 PM

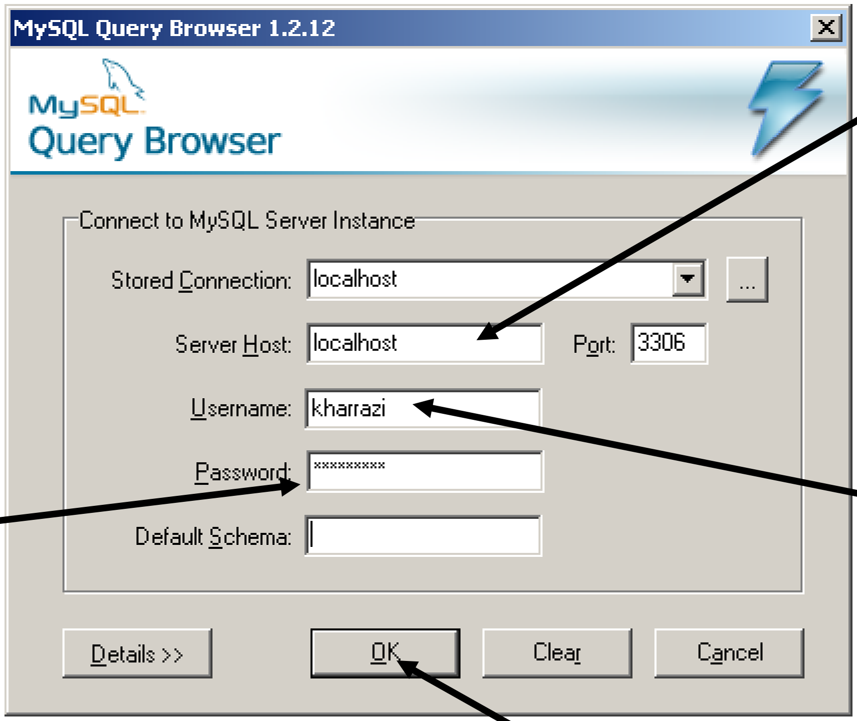
```
SELECT * FROM pat_info, pat_doc_relate, doc_info WHERE pat_info.P_id = pat_doc_relate.P_id AND pat_do
```

| P... | P_First... | P_Last... | City_id | Pho... | Kin_id | Clien... | P_id | D_id | D_id | D_Fir... | D_Last... | City_id | Phone... | Client_id |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 | 1 | 4 | 4 | Marry | Johnson | 3 | 1 | 7 |
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 | 1 | 5 | 5 | Yu | Lee | 1 | 12 | 8 |
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 | 1 | 6 | 6 | Nancy | McIssac | 1 | 4 | 9 |
| 2 | Mike | Jackson | 1 | 13 | 6 | 15 | 2 | 1 | 1 | Hadi | Kharrazi | 1 | 3 | 4 |
| 3 | Sara | Henson | 3 | 6 | 2 | 16 | 3 | 9 | 9 | Susan | Waterl... | 3 | 15 | 13 |
| 4 | John | McDon... | 5 | 8 | 3 | 17 | 4 | 1 | 1 | Hadi | Kharrazi | 1 | 3 | 4 |
| 4 | John | McDon... | 5 | 8 | 3 | 17 | 4 | 2 | 2 | John | McDo... | 4 | 11 | 5 |
| 4 | John | McDon... | 5 | 8 | 3 | 17 | 4 | 10 | 10 | Samuel | Jackson | 2 | 14 | 12 |
| 5 | Michael | Robinson | 1 | 13 | 6 | 18 | 5 | 5 | 5 | Yu | Lee | 1 | 12 | 8 |
| 6 | William | Jordan | 4 | 10 | 4 | 19 | 6 | 2 | 2 | John | McDo... | 4 | 11 | 5 |
| 7 | Susan | McKinsy | 1 | 2 | 5 | 20 | 7 | 6 | 6 | Nancy | McIssac | 1 | 4 | 9 |
| 8 | Mehdi | Kharrazi | 2 | 1 | 9 | 21 | 8 | 3 | 3 | Robin | Dorby | 2 | 8 | 6 |
| 8 | Mehdi | Kharrazi | 2 | 1 | 9 | 21 | 8 | 7 | 7 | John | Peterson | 2 | 16 | 10 |
| 9 | John | McKinsy | 1 | 9 | 10 | 22 | 9 | 10 | 10 | Samuel | Jackson | 2 | 14 | 12 |
| 10 | John | McDon... | 3 | 18 | 7 | 23 | 10 | 4 | 4 | Marry | Johnson | 3 | 1 | 7 |
| 11 | Pat | Bentatar | 7 | 25 | 8 | 24 | 11 | 6 | 6 | Nancy | McIssac | 1 | 4 | 9 |
| 12 | Abraham | Lincoln | 3 | 26 | 27 | 25 | 12 | 7 | 7 | John | Peterson | 2 | 16 | 10 |
| 12 | Abraham | Lincoln | 3 | 26 | 27 | 25 | 12 | 8 | 8 | Jane | Peterson | 2 | 13 | 11 |
| 13 | Brian | Adam | 5 | 27 | 13 | 26 | 13 | 9 | 9 | Susan | Waterl... | 3 | 15 | 13 |
| 14 | Catherin | Catholicy | 7 | 28 | 15 | 33 | 14 | 2 | 2 | John | McDo... | 4 | 11 | 5 |
| 14 | Catherin | Catholicy | 7 | 28 | 15 | 33 | 14 | 6 | 6 | Nancy | McIssac | 1 | 4 | 9 |
| 15 | Demi | Moore | 12 | 29 | 23 | 34 | 15 | 4 | 4 | Marry | Johnson | 3 | 1 | 7 |
| 15 | Demi | Moore | 12 | 29 | 23 | 34 | 15 | 5 | 5 | Yu | Lee | 1 | 12 | 8 |
| 16 | Ebi | Faraha... | 11 | 30 | 26 | 42 | 16 | 6 | 6 | Nancy | McIssac | 1 | 4 | 9 |
| 17 | Fery | Sea | 12 | 31 | 14 | 43 | 17 | 8 | 8 | Jane | Peterson | 2 | 13 | 11 |
| 18 | Graham | Bell | 12 | 32 | 12 | 32 | 18 | 4 | 4 | Marry | Johnson | 3 | 1 | 7 |

```
80 rows fetched in 0.0186s (0.1755s)
1:  1   A MySQL error was encountered. The message is:  The default schema cannot be changed to 'kharrazi'  The following error occu
```

## *MySQL GUI Tools / Query Browser (cont.)*

- Connecting to MySQL:



MySQL Query Browser 1.2.12

Connect to MySQL Server Instance

Stored Connection: localhost

Server Host: localhost

Port: 3306

Username: kharrazi

Password: *********

Default Schema:

**1** **Server's Domain Name (IP Address) for example 'localhost'**

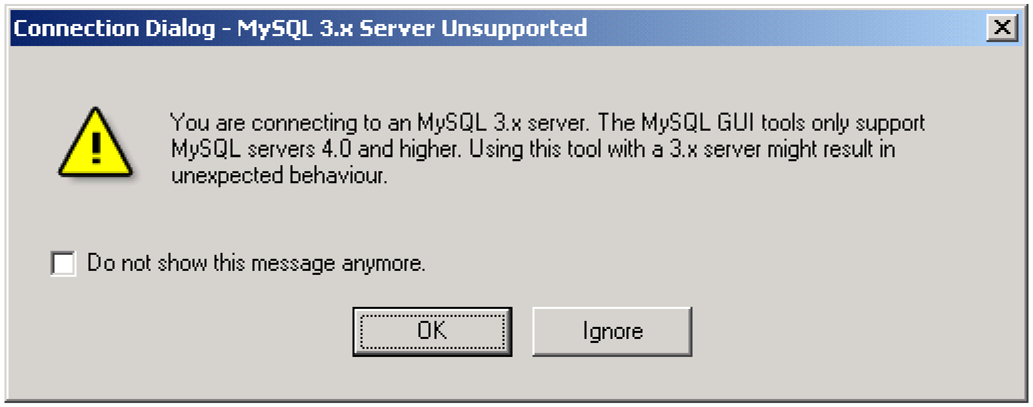
**2** **MySQL Username (Should be the same as your OnCourse account)**

**3** **MySQL Password (The one that you just set by PuTTY)**

**4** **Connecting to the Server**

## *MySQL GUI Tools / Query Browser (cont.)*

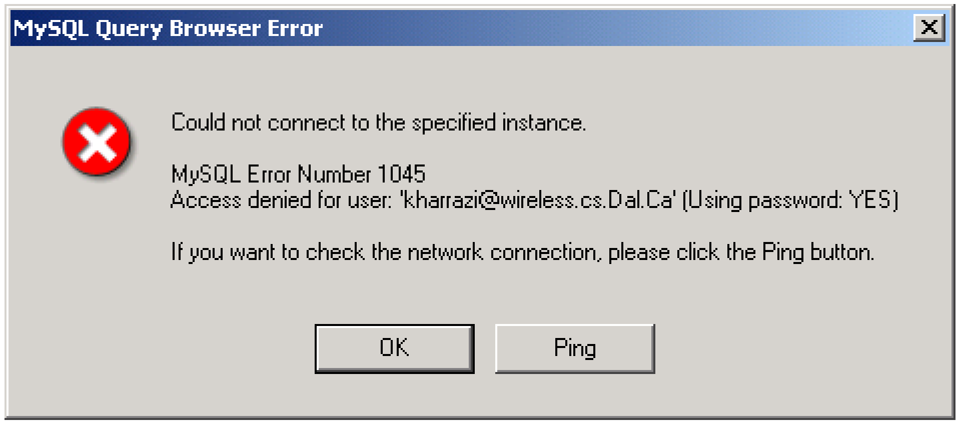
- Connection Errors:



You are connecting to an MySQL 3.x server. The MySQL GUI tools only support MySQL servers 4.0 and higher. Using this tool with a 3.x server might result in unexpected behaviour.

☐ Do not show this message anymore.

**Your MySQL database is old (less than version 4.1)**



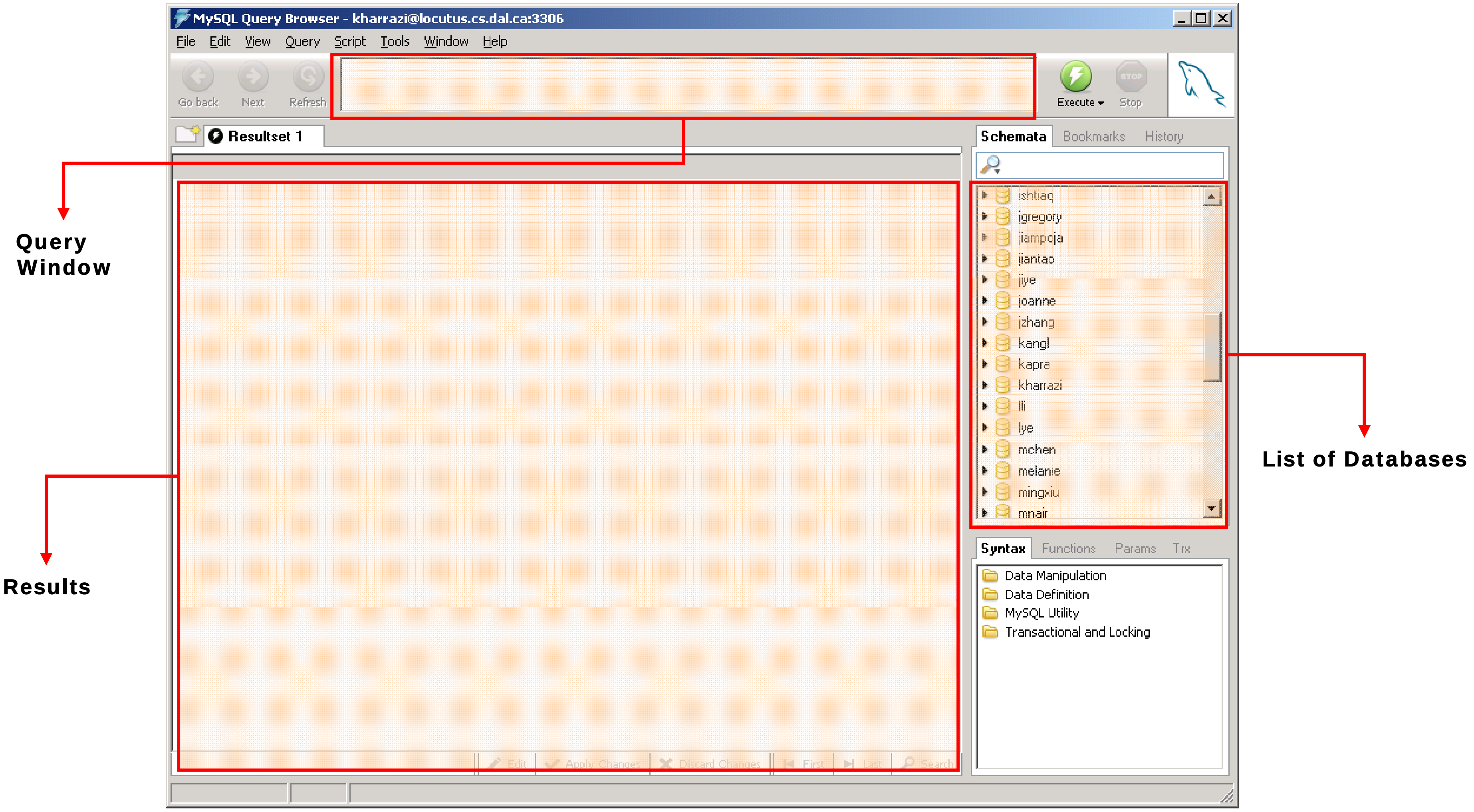
Could not connect to the specified instance.

MySQL Error Number 1045
Access denied for user: 'kharrazi@wireless.cs.Dal.Ca' (Using password: YES)

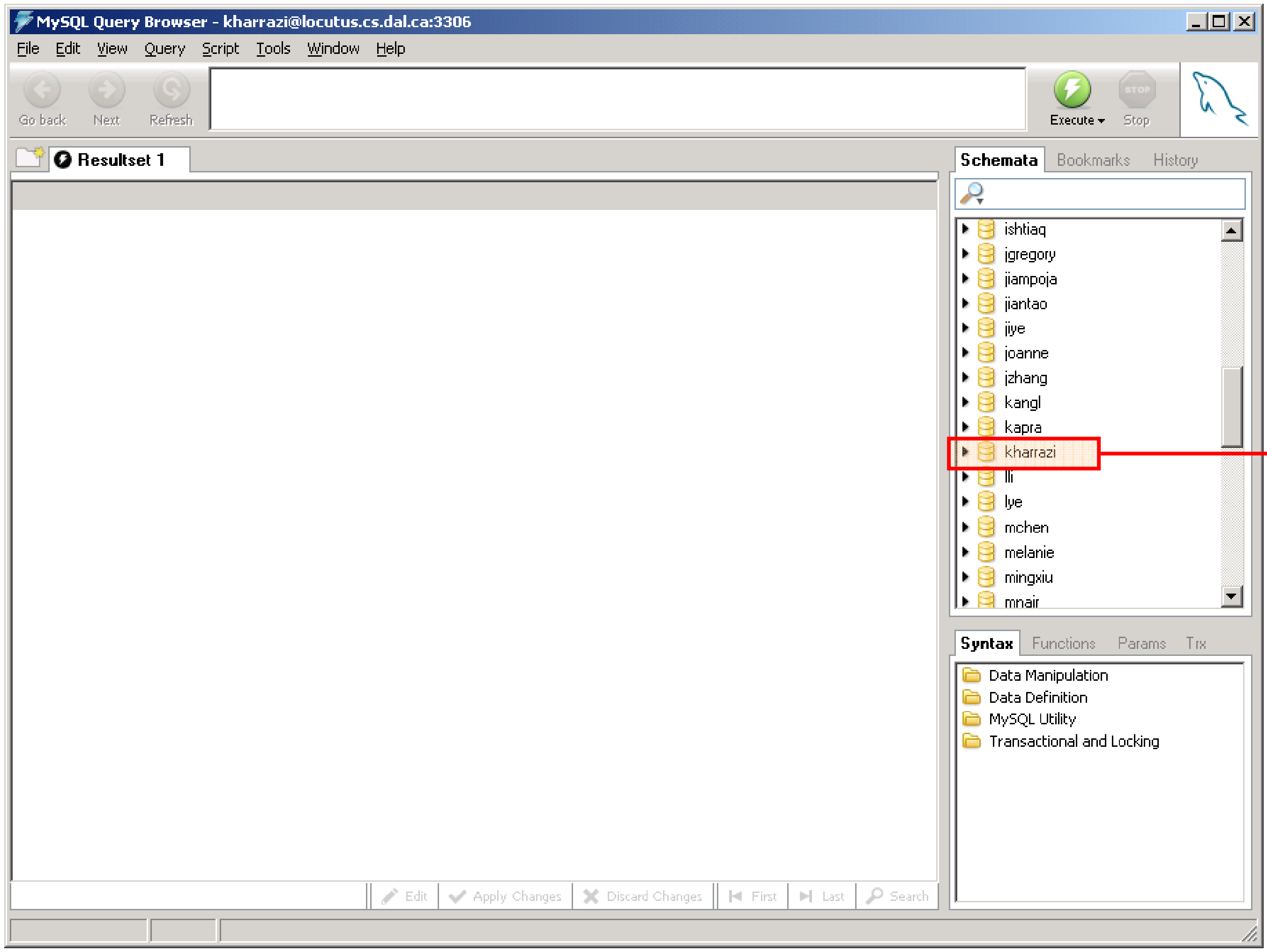
If you want to check the network connection, please click the Ping button.

**Connection failure (User / Pass wrong, Web Server is down, MySQL is down, …)**

## ***MySQL GUI Tools / Query Browser (cont.)*** (Browsing)

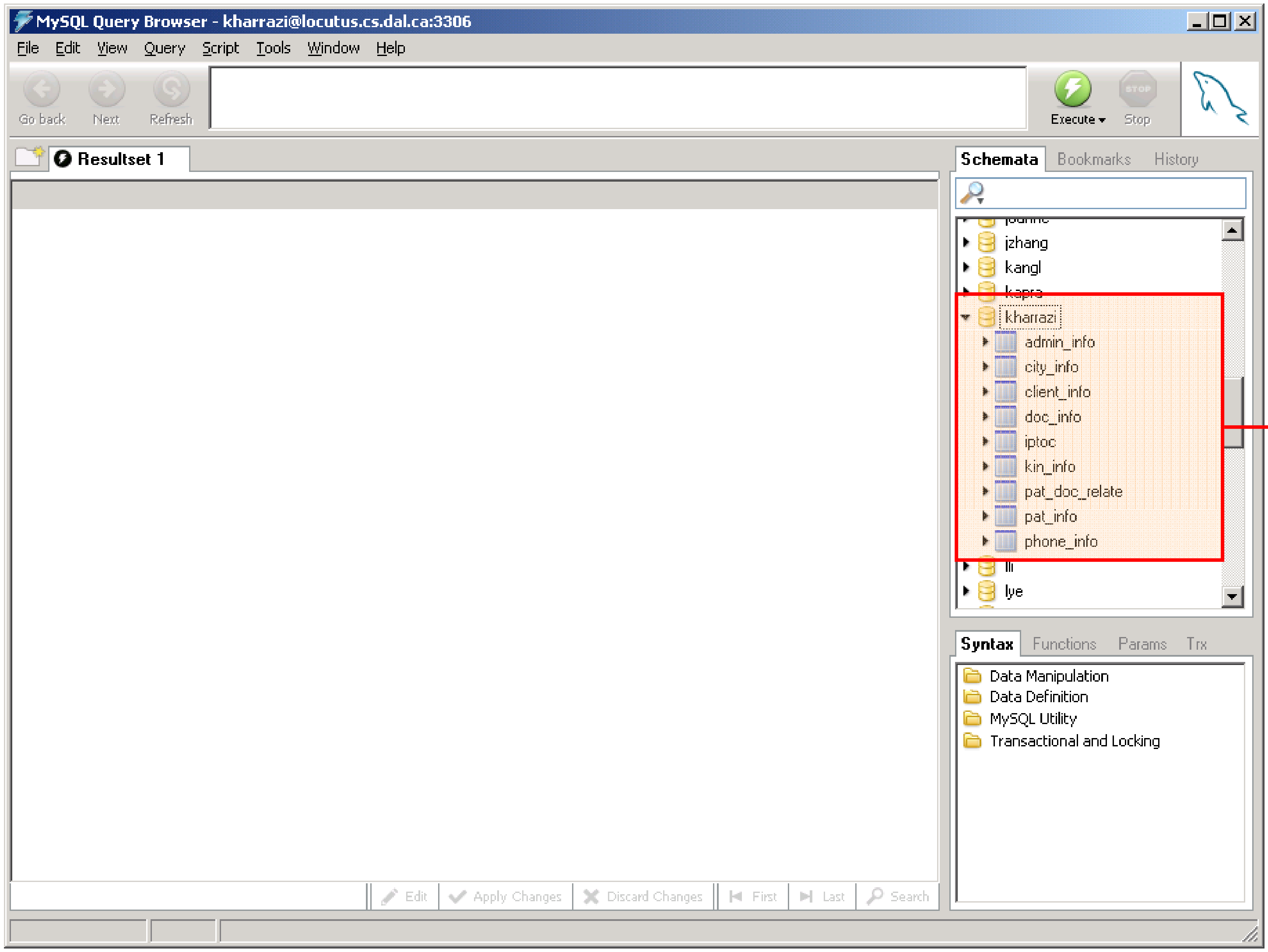


## ***MySQL GUI Tools / Query Browser (cont.)*** (Fetching Database)


MySQL Query Browser - kharrazi@locutus.cs.dal.ca:3306
File Edit View Query Script Tools Window Help
Go back Next Refresh
Execute Stop
Resultset 1
Schemata Bookmarks History
ishtiaq
jgregory
jiampoja
jiantao
jiye
joanne
jzhang
kangl
kapra
kharrazi
lli
lye
mchen
melanie
mingxiu
mnair
Syntax Functions Params Trx
Data Manipulation
Data Definition
MySQL Utility
Transactional and Locking
Edit Apply Changes Discard Changes First Last Search

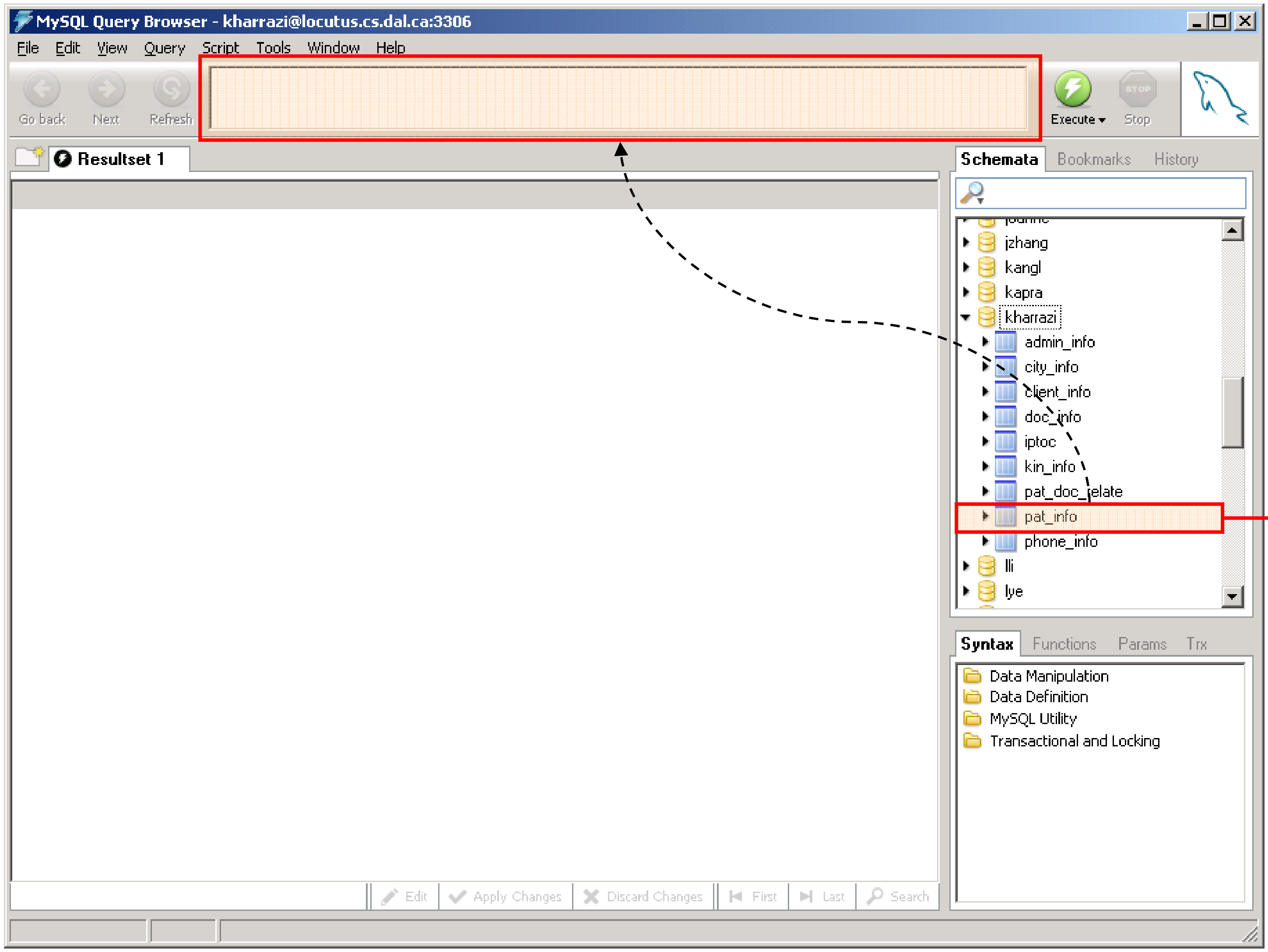
**Click on your DB**

## *MySQL GUI Tools / Query Browser (cont.)* (Fetching Database)



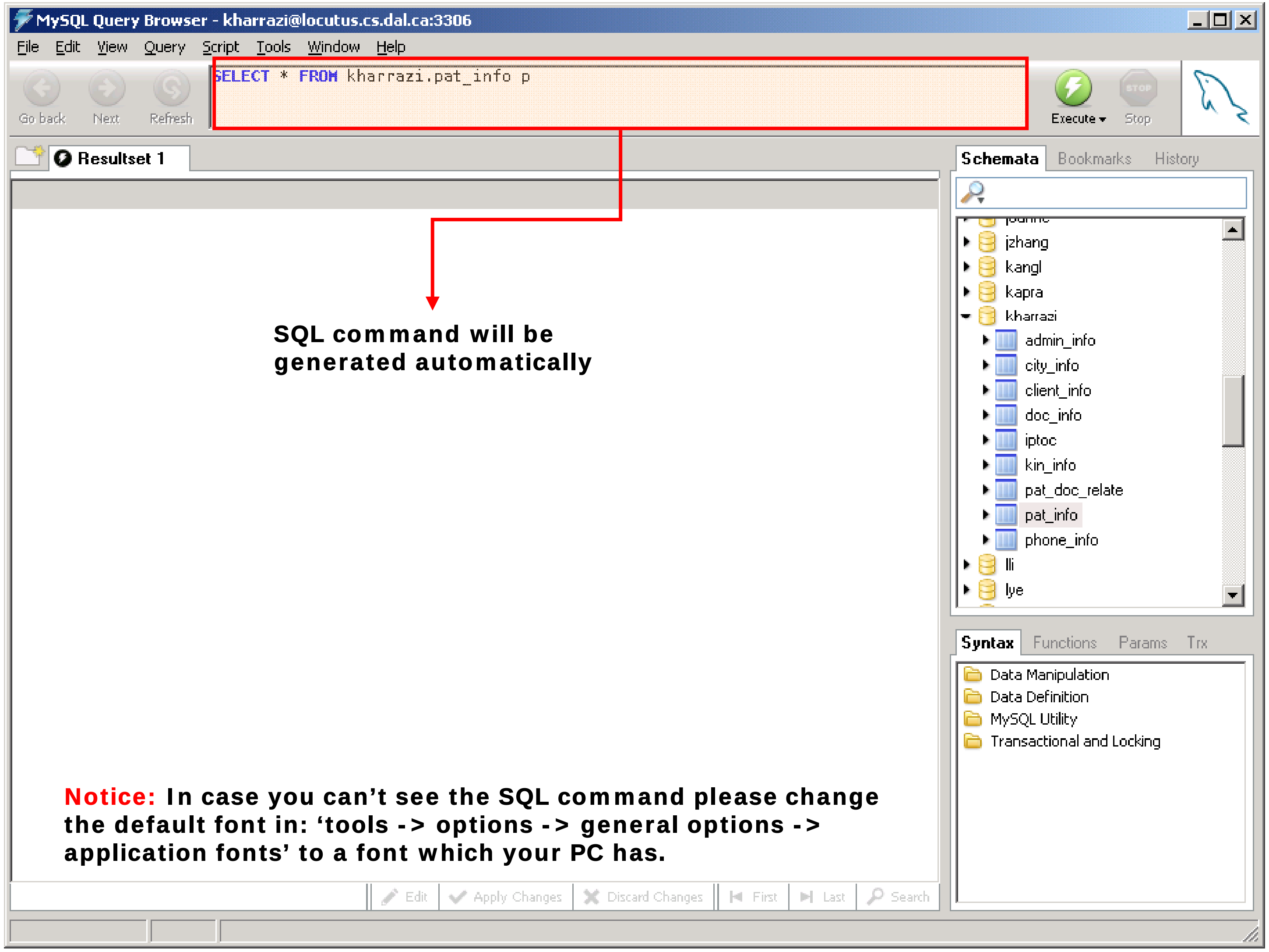
**If you had tables you would see them here**

## ***MySQL GUI Tools / Query Browser (cont.)*** (Querying a Table)



**Drag one of your table and drop it in the SQL area**

## ***MySQL GUI Tools / Query Browser (cont.)*** (Querying a Table)

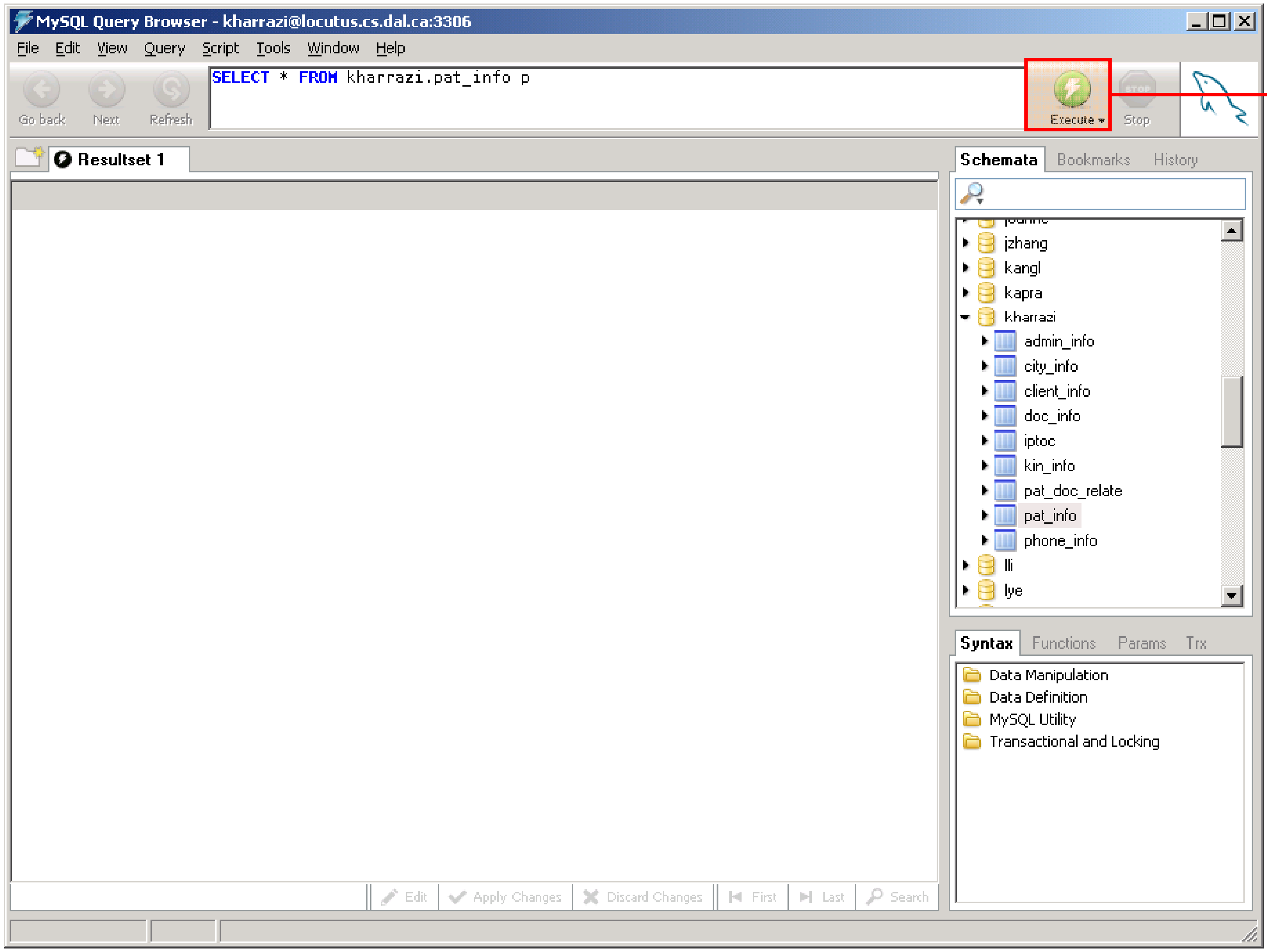


```
SELECT * FROM kharrazi.pat_info p
```

**SQL command will be generated automatically**

**Notice: In case you can't see the SQL command please change the default font in: 'tools -> options -> general options -> application fonts' to a font which your PC has.**

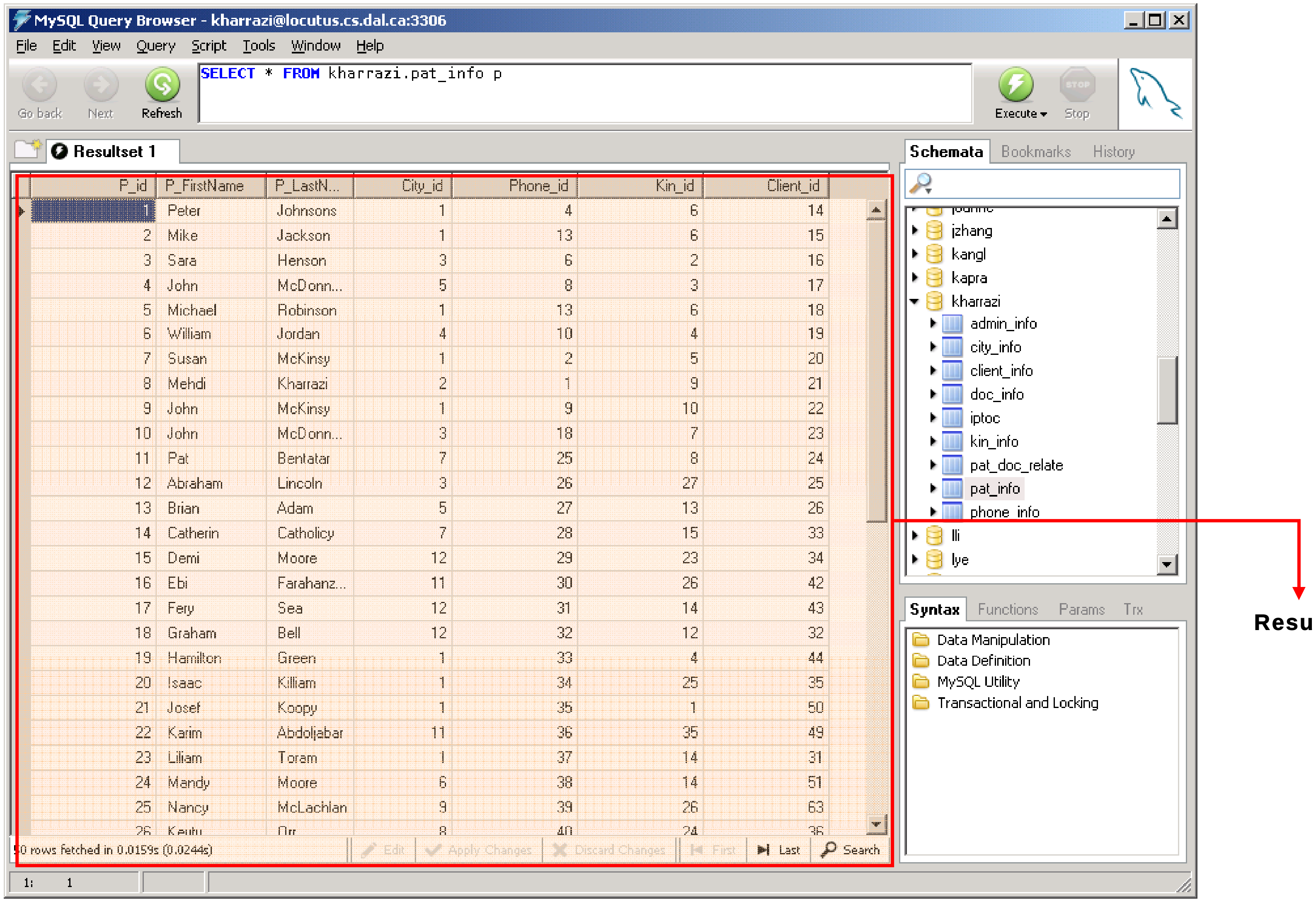
## ***MySQL GUI Tools / Query Browser (cont.)*** (Querying a Table)



```
SELECT * FROM kharrazi.pat_info p
```

**Click on Execute**

## ***MySQL GUI Tools / Query Browser (cont.)*** (Querying a Table)


MySQL Query Browser - kharrazi@locutus.cs.dal.ca:3306
File Edit View Query Script Tools Window Help
Go back Next Refresh
Execute Stop
Resultset 1
Schemata Bookmarks History
joanne
jzhang
kangl
kapra
kharrazi
admin_info
city_info
client_info
doc_info
iptoc
kin_info
pat_doc_relate
pat_info
phone_info
lli
lye
Syntax Functions Params Trx
Data Manipulation
Data Definition
MySQL Utility
Transactional and Locking
Edit Apply Changes Discard Changes First Last Search

```
SELECT * FROM kharrazi.pat_info p
```

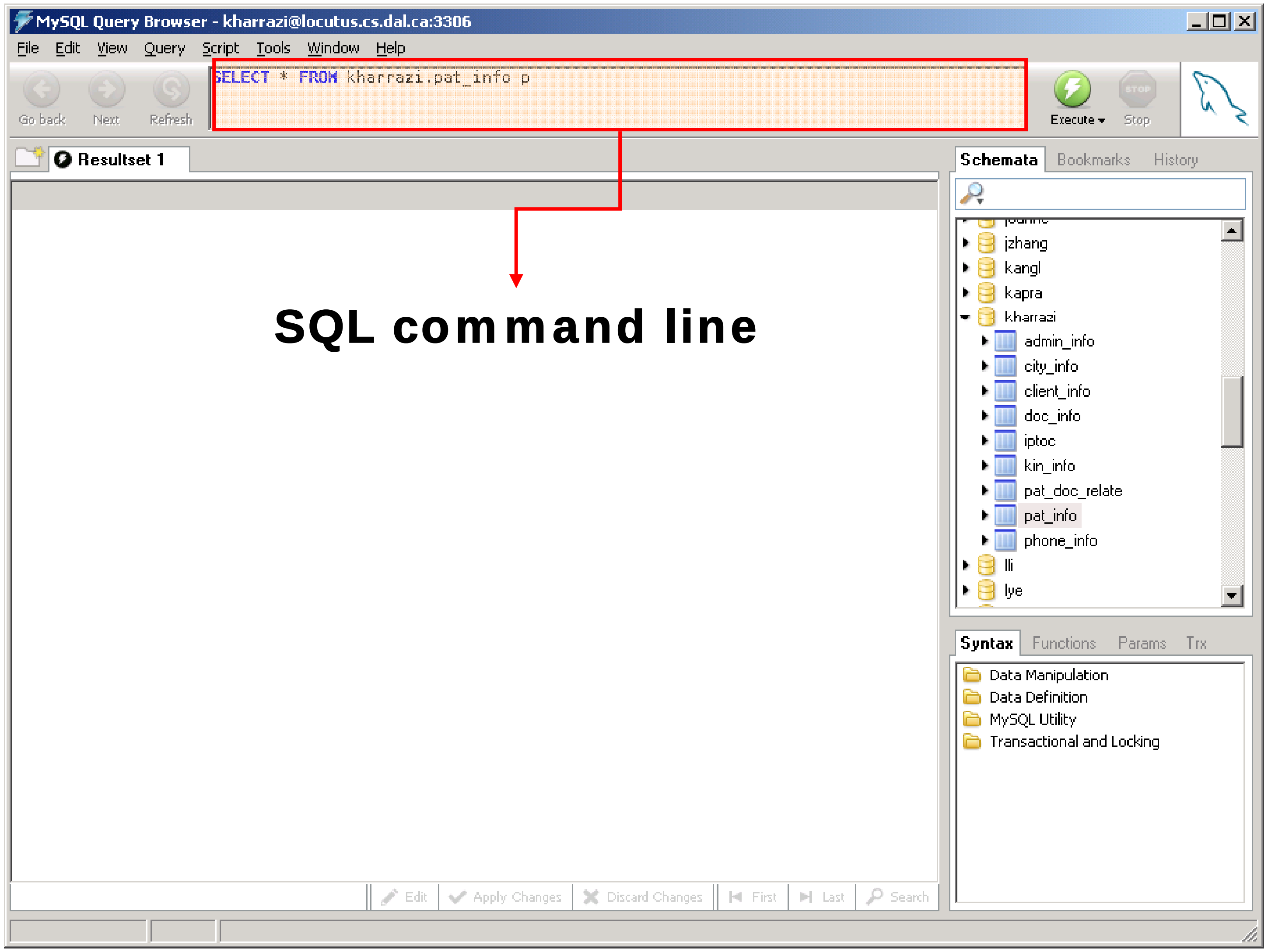
| P_id | P_FirstName | P_LastN... | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 |
| 2 | Mike | Jackson | 1 | 13 | 6 | 15 |
| 3 | Sara | Henson | 3 | 6 | 2 | 16 |
| 4 | John | McDonn... | 5 | 8 | 3 | 17 |
| 5 | Michael | Robinson | 1 | 13 | 6 | 18 |
| 6 | William | Jordan | 4 | 10 | 4 | 19 |
| 7 | Susan | McKinsy | 1 | 2 | 5 | 20 |
| 8 | Mehdi | Kharrazi | 2 | 1 | 9 | 21 |
| 9 | John | McKinsy | 1 | 9 | 10 | 22 |
| 10 | John | McDonn... | 3 | 18 | 7 | 23 |
| 11 | Pat | Bentatar | 7 | 25 | 8 | 24 |
| 12 | Abraham | Lincoln | 3 | 26 | 27 | 25 |
| 13 | Brian | Adam | 5 | 27 | 13 | 26 |
| 14 | Catherin | Catholicy | 7 | 28 | 15 | 33 |
| 15 | Demi | Moore | 12 | 29 | 23 | 34 |
| 16 | Ebi | Farahanz... | 11 | 30 | 26 | 42 |
| 17 | Fery | Sea | 12 | 31 | 14 | 43 |
| 18 | Graham | Bell | 12 | 32 | 12 | 32 |
| 19 | Hamilton | Green | 1 | 33 | 4 | 44 |
| 20 | Isaac | Killiam | 1 | 34 | 25 | 35 |
| 21 | Josef | Koopy | 1 | 35 | 1 | 50 |
| 22 | Karim | Abdoljabar | 11 | 36 | 35 | 49 |
| 23 | Liliam | Toram | 1 | 37 | 14 | 31 |
| 24 | Mandy | Moore | 6 | 38 | 14 | 51 |
| 25 | Nancy | McLachlan | 9 | 39 | 26 | 63 |
| 26 | Keutu | Orr | 8 | 40 | 24 | 36 |

50 rows fetched in 0.0159s (0.0244s)

1: 1

**Results**

## ***MySQL GUI Tools / Query Browser (cont.)*** (Querying a Table)



```
SELECT * FROM kharrazi.pat_info p
```

## *MySQL GUI Tools / Query Browser (cont.)*

### Restoring a dumped database

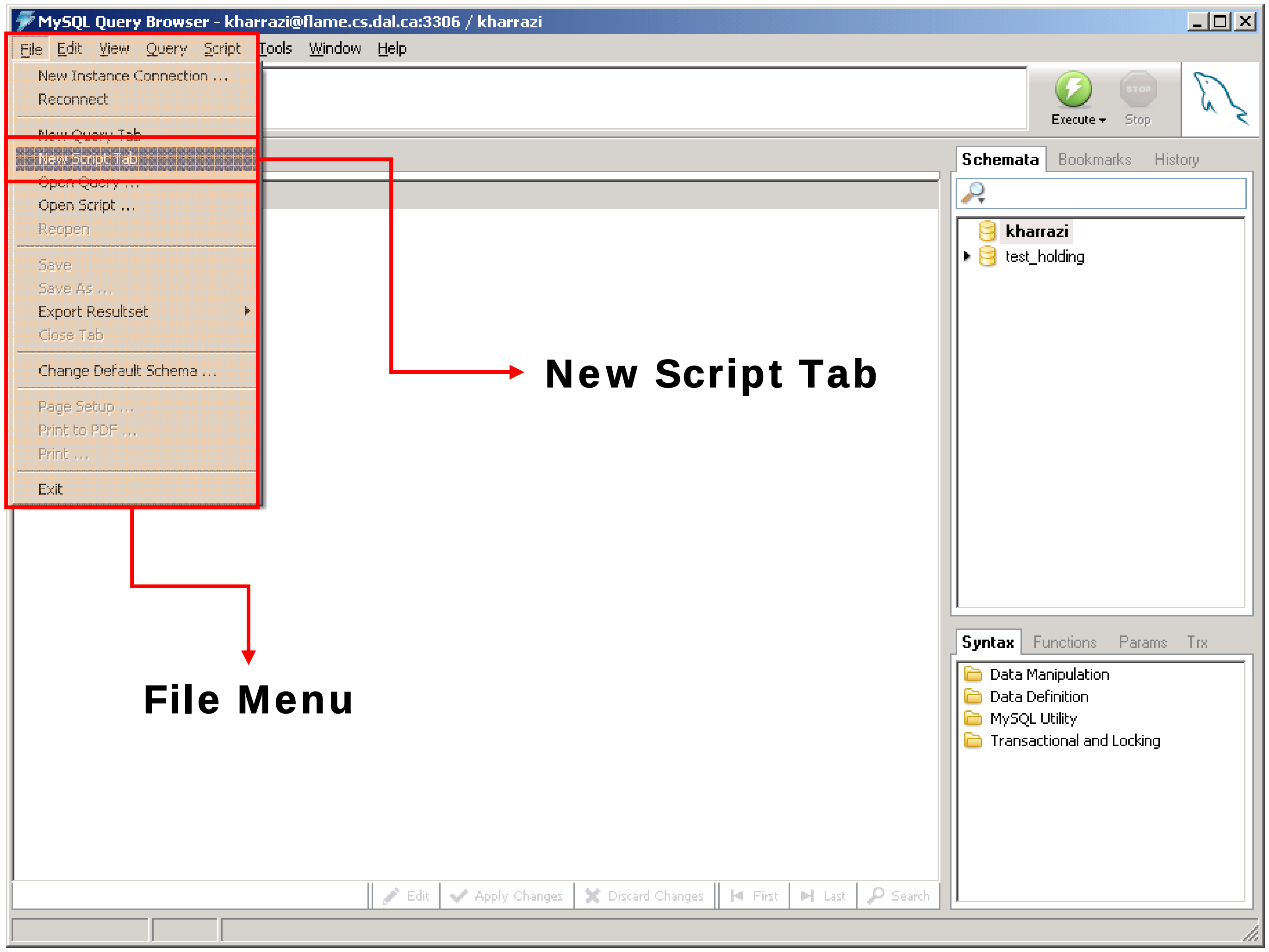
- Before starting the SQL languages we should have tables and data to test the commands on them.

- In the *Resources* folder that you downloaded at the beginning of this session a file named *sample_data.sql* exists that contains a dumped (stored) version of a sample database created beforehand by the tutor.

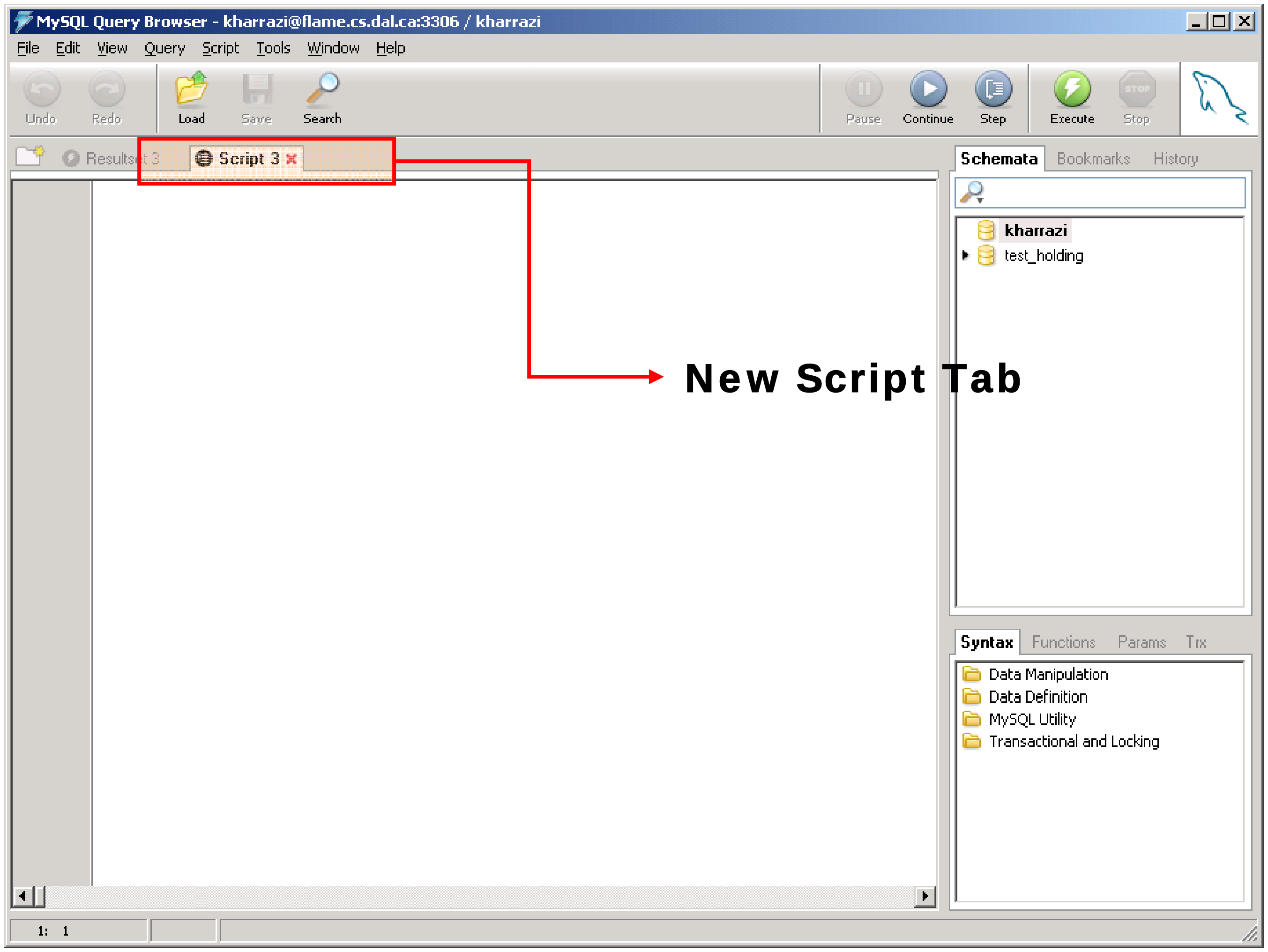
- In the next couple of slides we will restore the dumped version of the class database, which is now basically a file, into our databases on the faculty server.

- Creating tables and databases, dumping a database and restoring a database from a dumped file will be discussed later in future tutorials.

## *MySQL GUI Tools / Query Browser (cont.)* (Restoring DB)

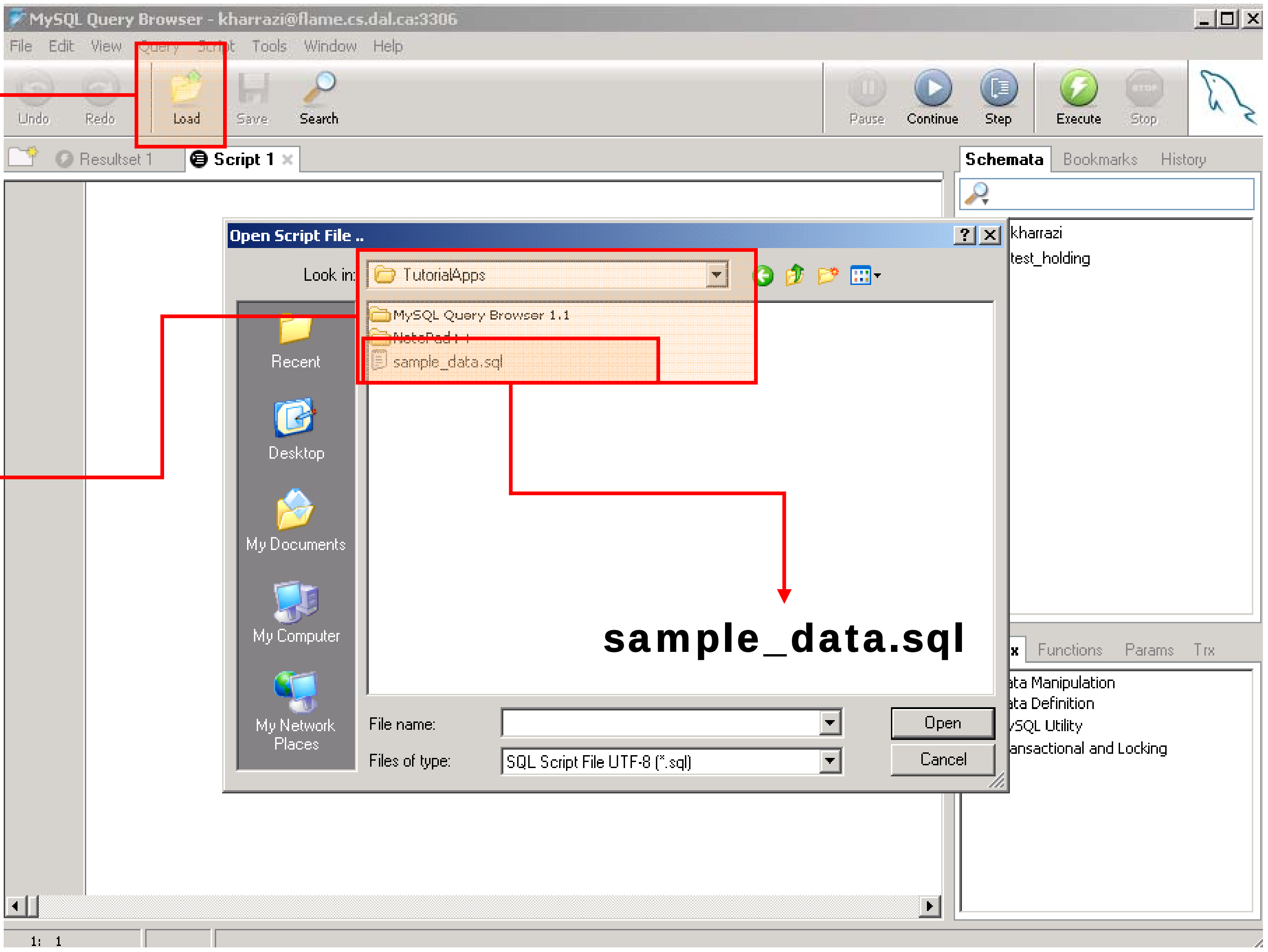

MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
New Instance Connection ...
Reconnect
New Query Tab
New Script Tab
Open Query ...
Open Script ...
Reopen
Save
Save As ...
Export Resultset
Close Tab
Change Default Schema ...
Page Setup ...
Print to PDF ...
Print ...
Exit
Execute
Stop
Schemata Bookmarks History
kharrazi
test_holding
Syntax Functions Params Trx
Data Manipulation
Data Definition
MySQL Utility
Transactional and Locking
Edit Apply Changes Discard Changes First Last Search
New Script Tab
File Menu

## *MySQL GUI Tools / Query Browser (cont.)* (Restoring DB)


MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Undo Redo Load Save Search
Pause Continue Step Execute Stop
Resultset 3
Script 3
Schemata Bookmarks History
kharrazi
test_holding
Syntax Functions Params Trx
Data Manipulation
Data Definition
MySQL Utility
Transactional and Locking
1: 1

**New Script Tab**

## *MySQL GUI Tools / Query Browser (cont.)* (Restoring DB)

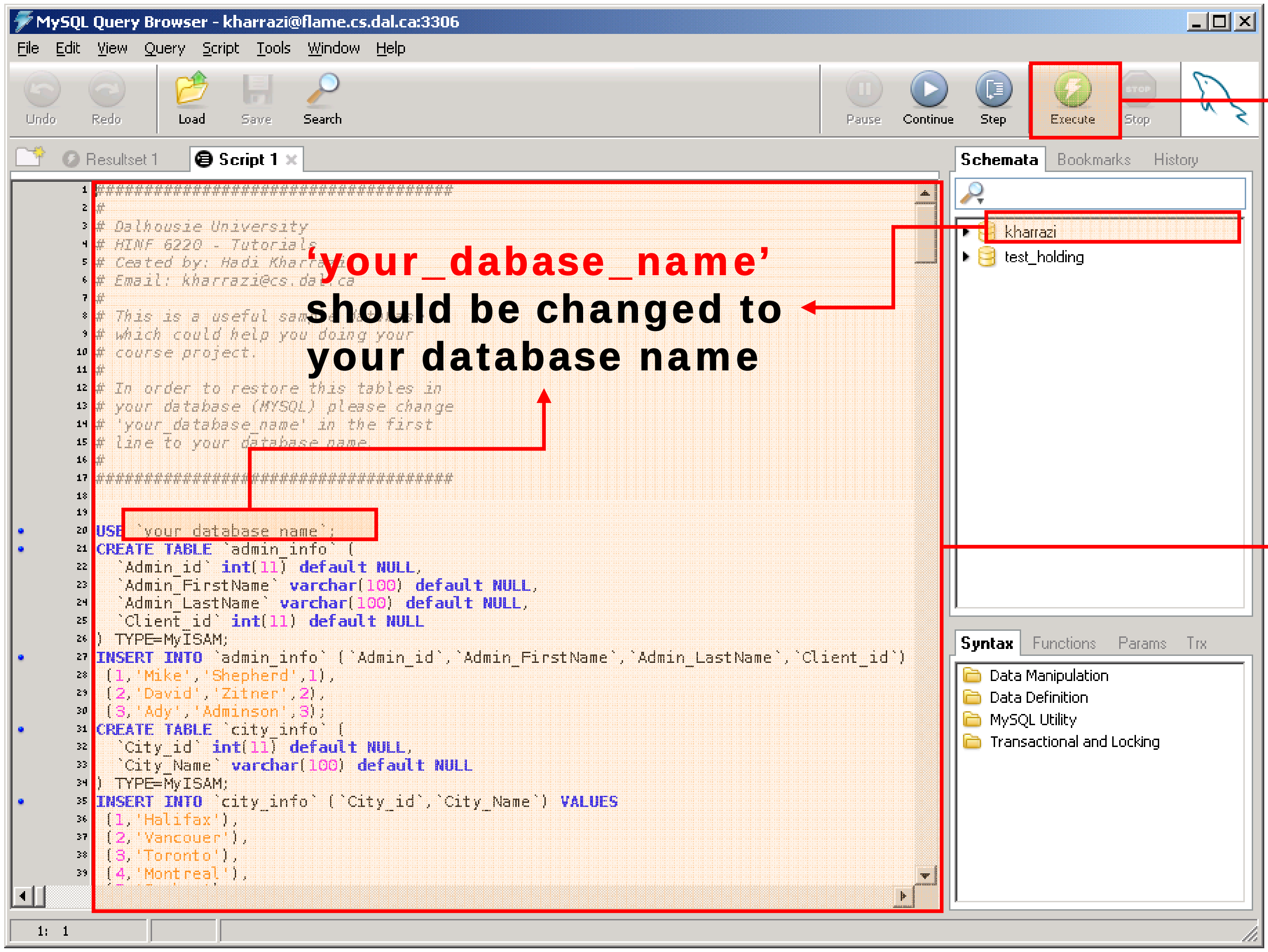


**Load**

**Tutorial Apps Folder**

**sample_data.sql**

## *MySQL GUI Tools / Query Browser (cont.)* (Restoring DB)


MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306
File Edit View Query Script Tools Window Help
Undo Redo Load Save Search
Pause Continue Step Execute Stop
Resultset 1
Script 1
Schemata Bookmarks History
kharrazi
test_holding
Syntax Functions Params Trx
Data Manipulation
Data Definition
MySQL Utility
Transactional and Locking
1: 1

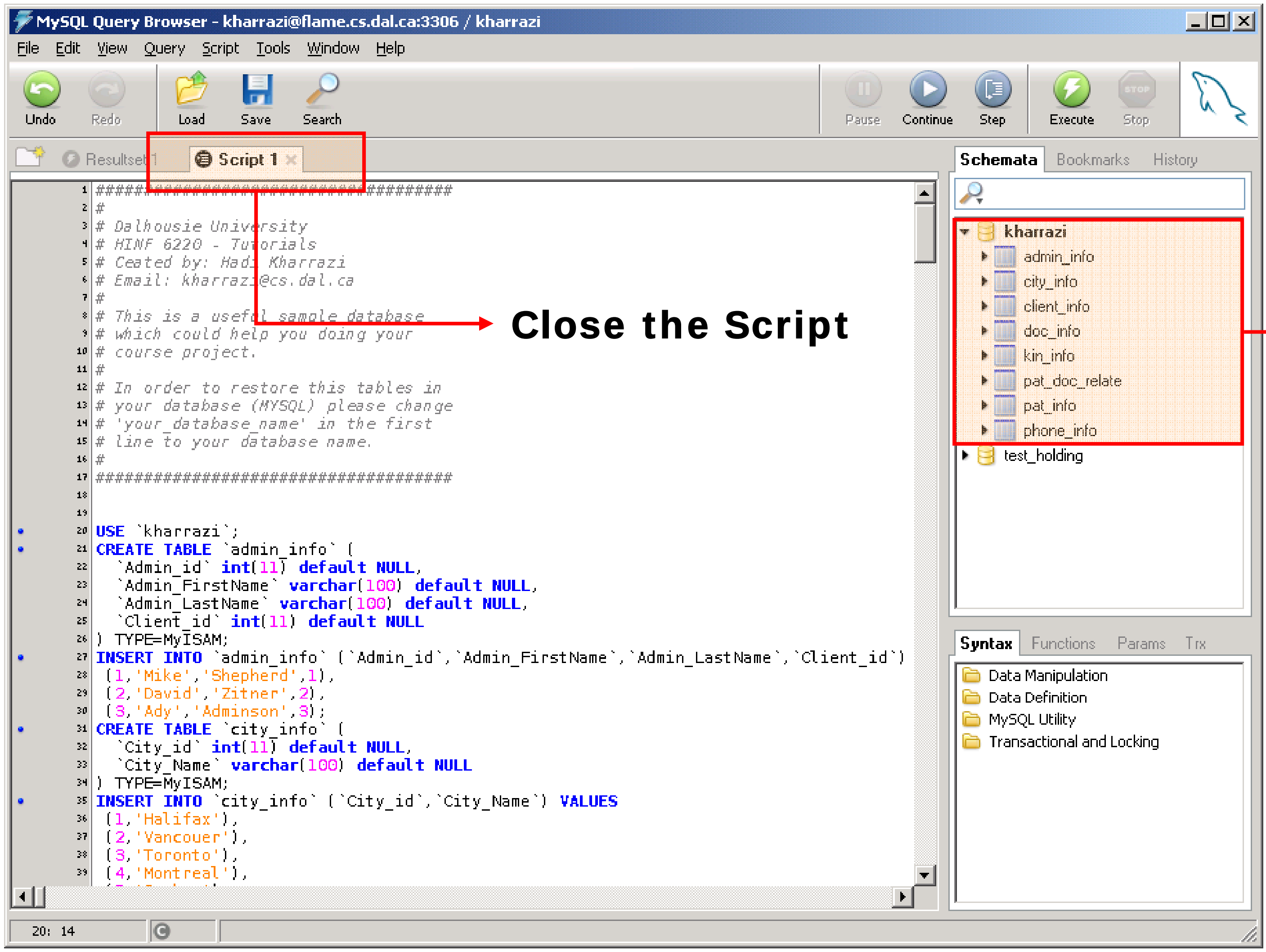
```
#####################################
#
# Dalhousie University
# HINF 6220 - Tutorials
# Ceated by: Hadi Kharrazi
# Email: kharrazi@cs.dal.ca
#
# This is a useful sample database
# which could help you doing your
# course project.
#
# In order to restore this tables in
# your database (MYSQL) please change
# 'your_database_name' in the first
# line to your database name.
#
#####################################


USE `your_database_name`;
CREATE TABLE `admin_info` (
  `Admin_id` int(11) default NULL,
  `Admin_FirstName` varchar(100) default NULL,
  `Admin_LastName` varchar(100) default NULL,
  `Client_id` int(11) default NULL
) TYPE=MyISAM;
INSERT INTO `admin_info` (`Admin_id`,`Admin_FirstName`,`Admin_LastName`,`Client_id`)
 (1,'Mike','Shepherd',1),
 (2,'David','Zitner',2),
 (3,'Ady','Adminson',3);
CREATE TABLE `city_info` (
  `City_id` int(11) default NULL,
  `City_Name` varchar(100) default NULL
) TYPE=MyISAM;
INSERT INTO `city_info` (`City_id`,`City_Name`) VALUES
 (1,'Halifax'),
 (2,'Vancouer'),
 (3,'Toronto'),
 (4,'Montreal'),
```

**'your_dabase_name' should be changed to your database name**

**Execute the restore**

**SQL Dump Code**

## ***MySQL GUI Tools / Query Browser (cont.)*** (Restoring DB)


MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Undo Redo Load Save Search
Pause Continue Step Execute Stop
Resultset 1
Script 1
Schemata Bookmarks History
kharrazi
admin_info
city_info
client_info
doc_info
kin_info
pat_doc_relate
pat_info
phone_info
test_holding
Syntax Functions Params Trx
Data Manipulation
Data Definition
MySQL Utility
Transactional and Locking
20: 14

```
#####################################
#
# Dalhousie University
# HINF 6220 - Tutorials
# Ceated by: Hadi Kharrazi
# Email: kharrazi@cs.dal.ca
#
# This is a useful sample database
# which could help you doing your
# course project.
#
# In order to restore this tables in
# your database (MYSQL) please change
# 'your_database_name' in the first
# line to your database name.
#
#####################################


USE `kharrazi`;
CREATE TABLE `admin_info` (
  `Admin_id` int(11) default NULL,
  `Admin_FirstName` varchar(100) default NULL,
  `Admin_LastName` varchar(100) default NULL,
  `Client_id` int(11) default NULL
) TYPE=MyISAM;
INSERT INTO `admin_info` (`Admin_id`,`Admin_FirstName`,`Admin_LastName`,`Client_id`)
 (1,'Mike','Shepherd',1),
 (2,'David','Zitner',2),
 (3,'Ady','Adminson',3);
CREATE TABLE `city_info` (
  `City_id` int(11) default NULL,
  `City_Name` varchar(100) default NULL
) TYPE=MyISAM;
INSERT INTO `city_info` (`City_id`,`City_Name`) VALUES
 (1,'Halifax'),
 (2,'Vancouer'),
 (3,'Toronto'),
 (4,'Montreal'),
```

**Close the Script**

**New tables created (restored)**

## 6. SQL Introduction

- SQL is a standard computer language for accessing and manipulating databases.

- What is SQL?

SQL stands for **Structured Query Language**
SQL allows you to **access** a database
SQL is an **ANSI** standard computer language
SQL can **execute queries** against a database
SQL can **retrieve** data from a database
SQL can **insert** new records in a database
SQL can **delete** records from a database
SQL can **update** records in a database
SQL is **easy** to learn

## *SQL Introduction (cont.)*

- SQL is an ANSI (American National Standards Institute) standard computer language for accessing and manipulating database systems.
- SQL statements are used to retrieve and update data in a database. SQL works with database programs like **MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase, etc**.
- Unfortunately, there are many different versions of the SQL language, but to be in compliance with the ANSI standard, they must support the same **major keywords** in a similar manner (such as SELECT, UPDATE, DELETE, INSERT, WHERE, and others).

## ***SQL Introduction (cont.)***

- SQL Data Manipulation Language (**DML**):

  * SELECT - extracts data from a database table
  * UPDATE - updates data in a database table
  * DELETE - deletes data from a database table
  * INSERT INTO - inserts new data into a database table

- SQL Data Definition Language (**DDL**):

  # CREATE TABLE - creates a new database table
  # ALTER TABLE - alters (changes) a database table
  # DROP TABLE - deletes a database table
  # CREATE INDEX - creates an index (search key)
  # DROP INDEX - deletes an index

## *SQL Introduction (cont.)*

***SQL in a Nutshell***

1. SQL Introduction
2. SQL: **SELECT** Statement
3. SQL: **WHERE** (BETWEEN/LIKE/LIMIT) Clause
4. SQL: **AND** & **OR**
5. SQL: **IN**
6. SQL: **ORDER BY** Clause
7. SQL: **INSERT INTO** Statement
8. SQL: **UPDATE/SET** Statement
9. SQL: **DELETE** Statement
10. SQL: *Joining and Keys (Inner Join)*
11. SQL: **LEFT JOIN/ON** *(Outer Join)*
12. SQL: **GROUP BY** & **HAVING**
13. SQL: **FUNCTIONS**
14. SQL: **CREATE** Database, Table, and Index
15. SQL: **DROP** Index, Table and Database
16. SQL: **ALTER** Table

## 7. SQL: SELECT Statement

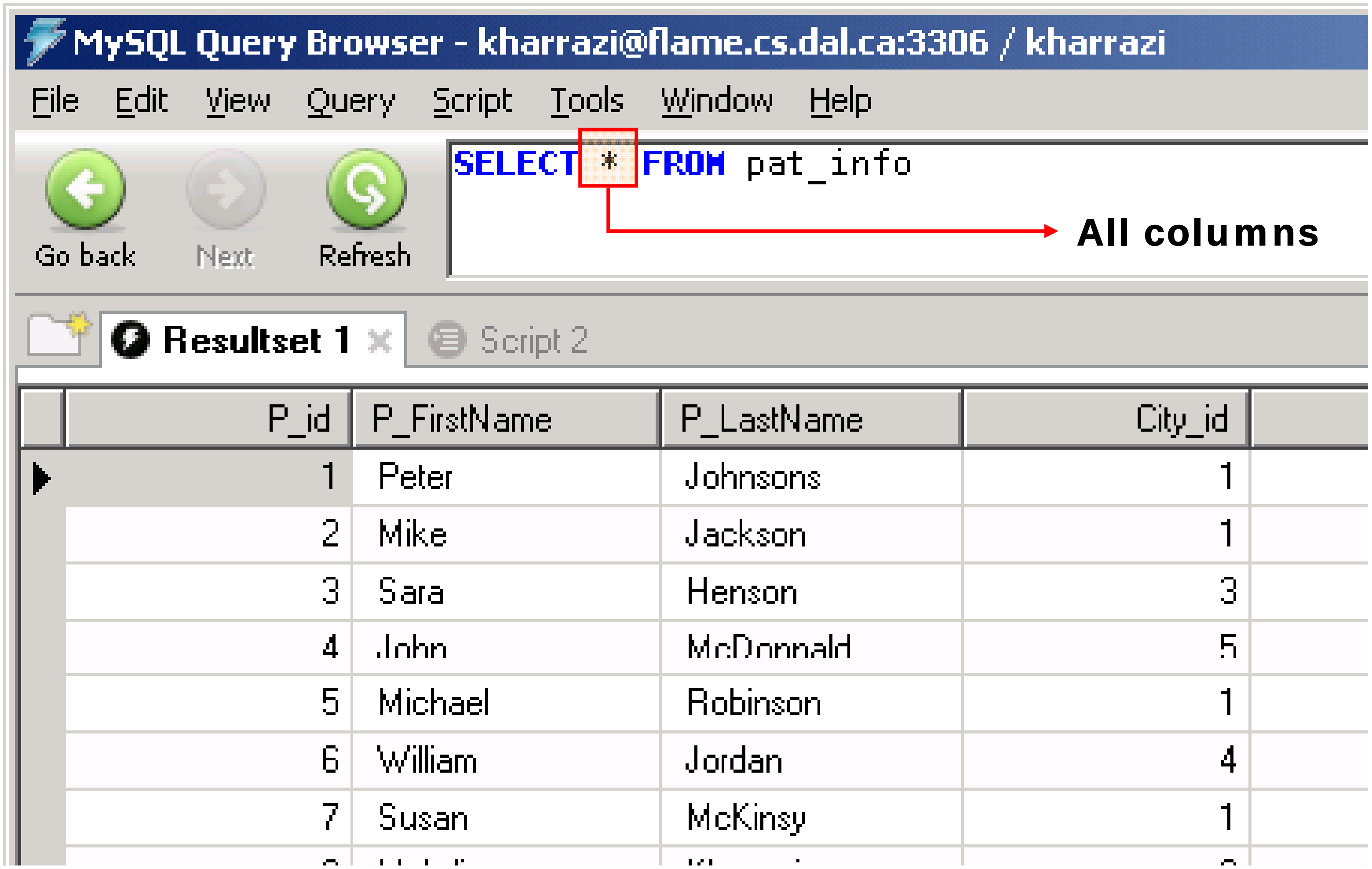
- The **SELECT** statement is used to select data from a table. The tabular result is stored in a result table.
- Syntax:

```
SELECT column_name(s) FROM table_name
```

- Examples:

```
SELECT * FROM pat_info

SELECT P_FirstName FROM pat_info

SELECT P_FirstName, P_LastName FROM pat_info

SELECT DISTINCT city_id FROM pat_info
```

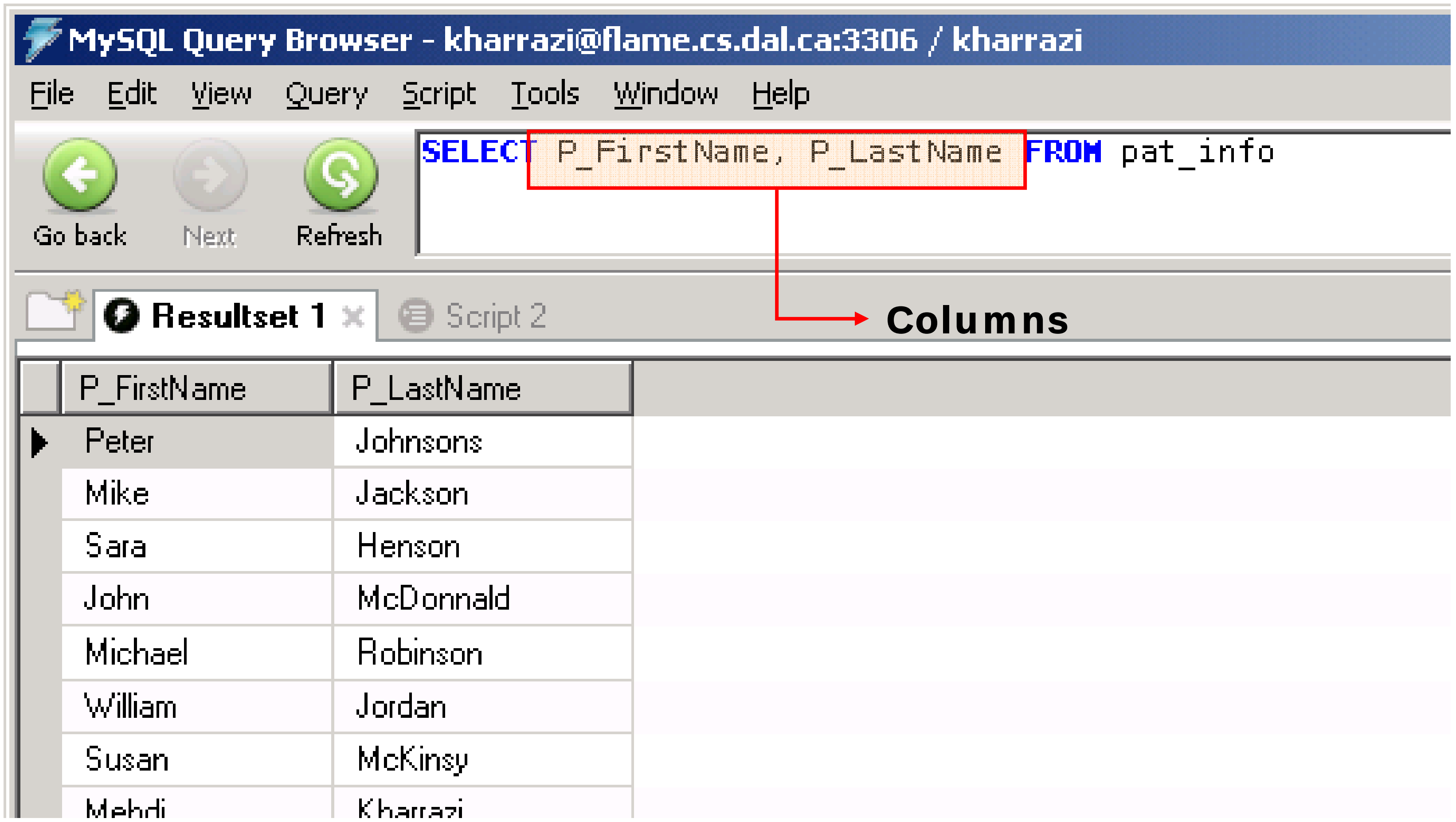
## SQL: SELECT Statement (cont.)



```
SELECT * FROM pat_info
```

| P_id | P_FirstName | P_LastName | City_id |
|---|---|---|---|
| 1 | Peter | Johnsons | 1 |
| 2 | Mike | Jackson | 1 |
| 3 | Sara | Henson | 3 |
| 4 | John | McDonnald | 5 |
| 5 | Michael | Robinson | 1 |
| 6 | William | Jordan | 4 |
| 7 | Susan | McKinsy | 1 |

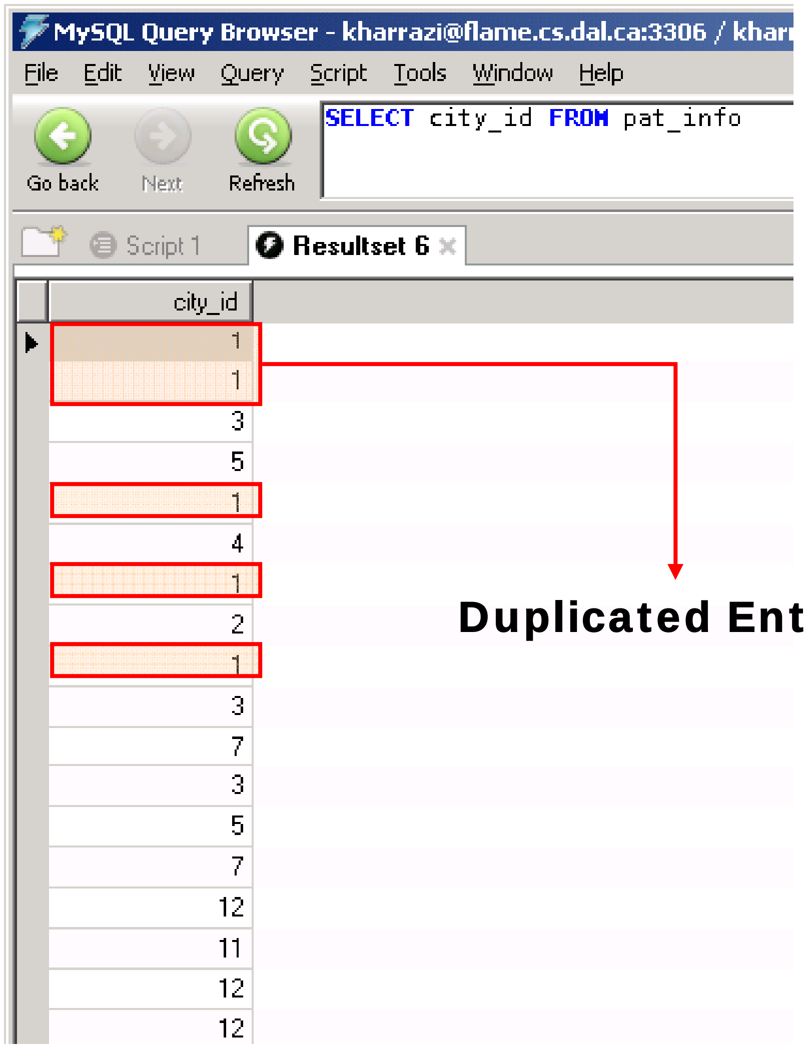
## SQL: SELECT Statement (cont.)


MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Resultset 1
Script 2
Columns

```
SELECT P_FirstName, P_LastName FROM pat_info
```

| P_FirstName | P_LastName |
|---|---|
| Peter | Johnsons |
| Mike | Jackson |
| Sara | Henson |
| John | McDonnald |
| Michael | Robinson |
| William | Jordan |
| Susan | McKinsy |
| Mehdi | Kharrazi |

## SQL: SELECT Statement (cont.)

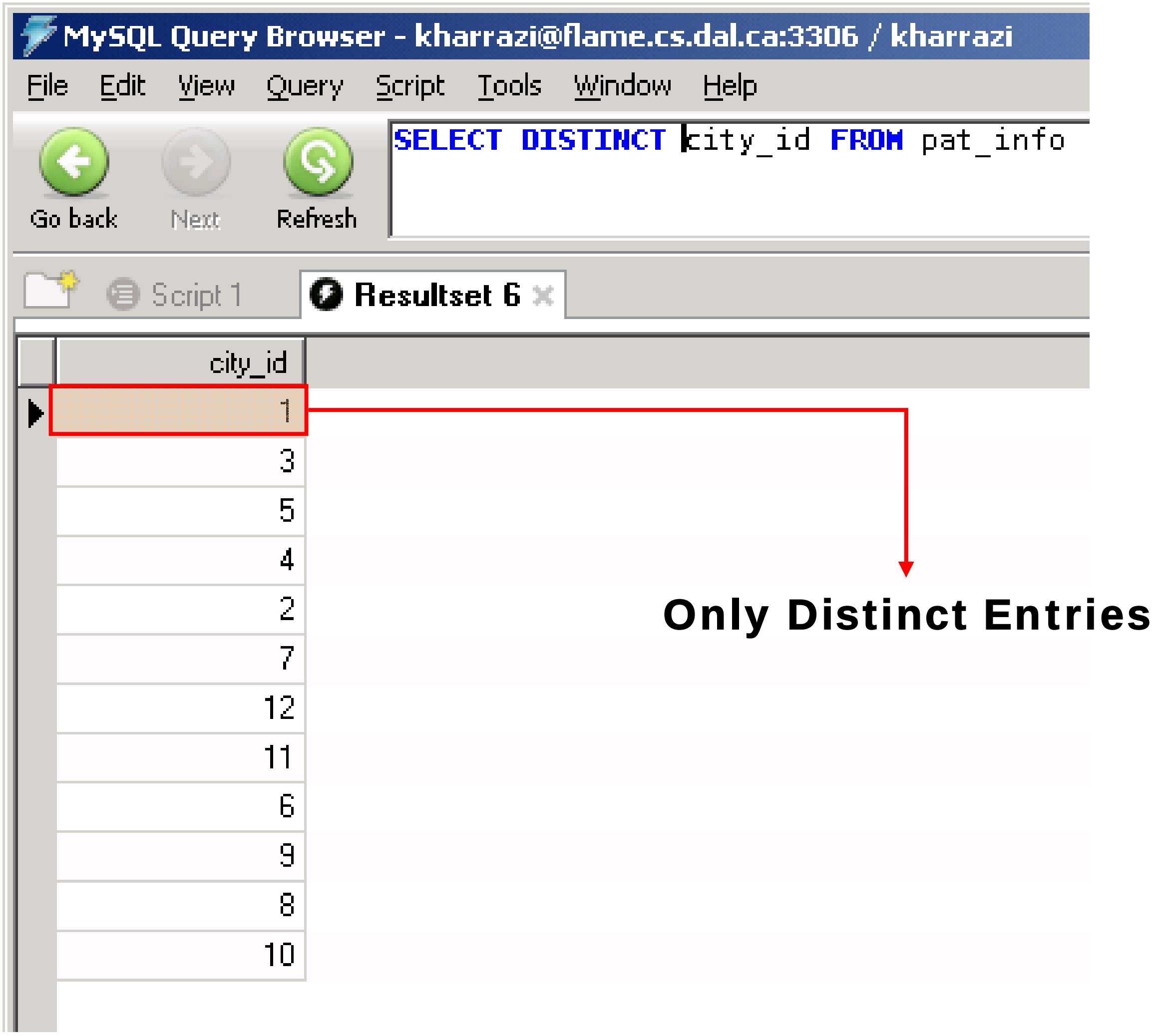

MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / khar
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Script 1
Resultset 6

```
SELECT city_id FROM pat_info
```

```
city_id
      1
      1
      3
      5
      1
      4
      1
      2
      1
      3
      7
      3
      5
      7
     12
     11
     12
     12
```

**Duplicated Entries**

## SQL: SELECT Statement (cont.)


MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Script 1
Resultset 6
Only Distinct Entries

```
SELECT DISTINCT city_id FROM pat_info
```

| city_id |
|---|
| 1 |
| 3 |
| 5 |
| 4 |
| 2 |
| 7 |
| 12 |
| 11 |
| 6 |
| 9 |
| 8 |
| 10 |

## 8. SQL: WHERE Statement

- To conditionally select data from a table, a **WHERE** clause can be added to the **SELECT** statement.
- Syntax:

```
SELECT column FROM table WHERE column operator value
```

- Examples:

```
SELECT * FROM pat_info WHERE P_FirstName='Mike'

SELECT * FROM pat_info WHERE P_id<6

SELECT * FROM pat_info WHERE P_id<6 LIMIT 2

SELECT * FROM pat_info WHERE P_id BETWEEN 6 AND 9

SELECT * FROM pat_info WHERE P_FirstName LIKE 'h%'
```

## ***SQL: WHERE Statement (cont.)***

| **Operator** | **Description** |
|---|---|
| = | Equal |
| <> | Not equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |
| BETWEEN | Between an inclusive range |
| LIKE | Search for a pattern |

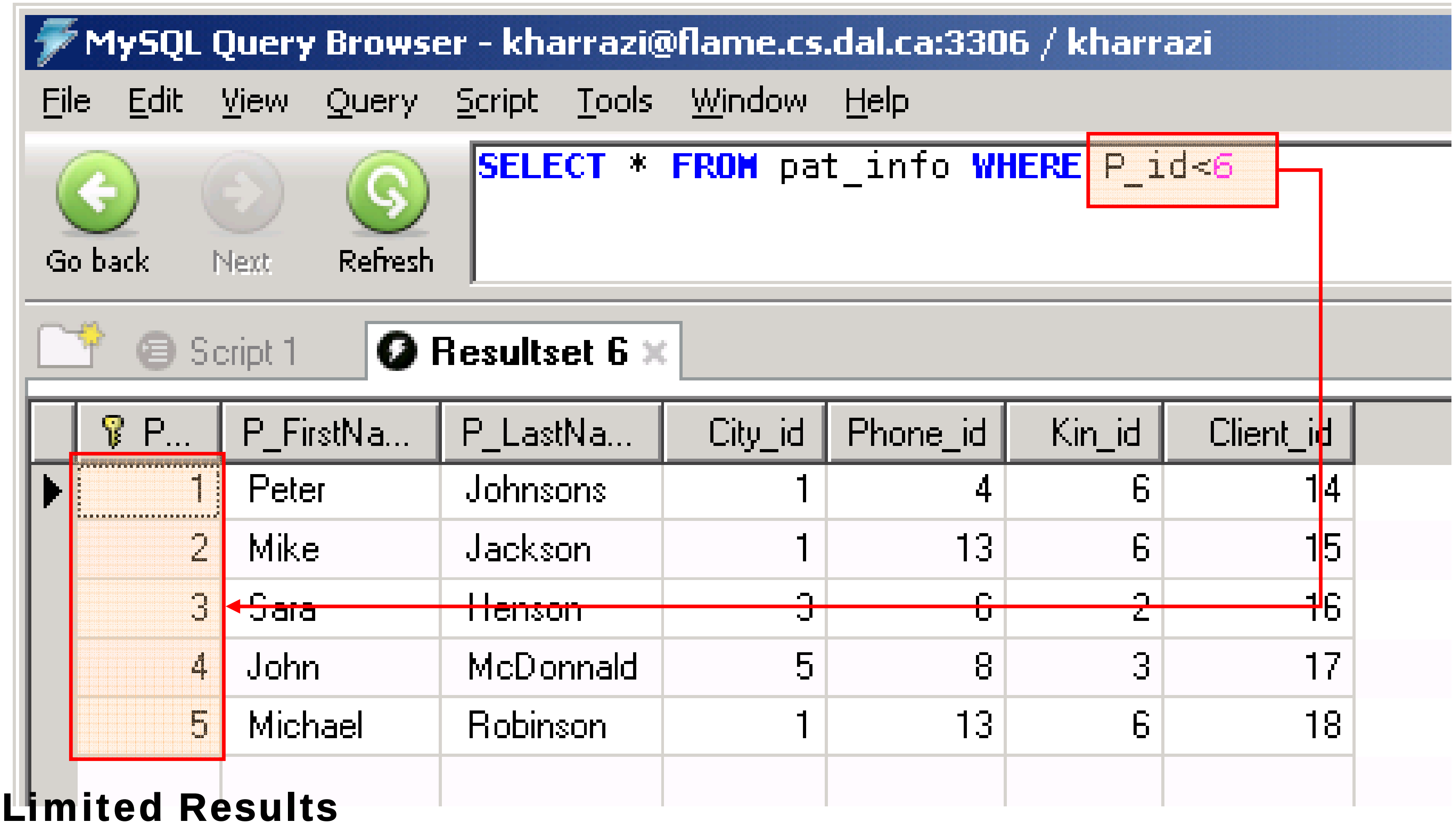
## ***SQL: WHERE Statement (cont.)***

- **NOT** clause could be combined with the WHERE statement in order to invert the selection range.

- "**`SELECT * FROM pat_info WHERE P_id BETWEEN 6 AND 9`**" will select only those P_id data that vary between 6 and 9 but "**`SELECT * FROM pat_info WHERE P_id NOT BETWEEN 6 AND 9`**" will select any P_id except those P_id data that vary between 6 and 9.

- Queries with boolean expressions can get very sophisticated (beyond the scope of the course).

## SQL: WHERE Statement (cont.)

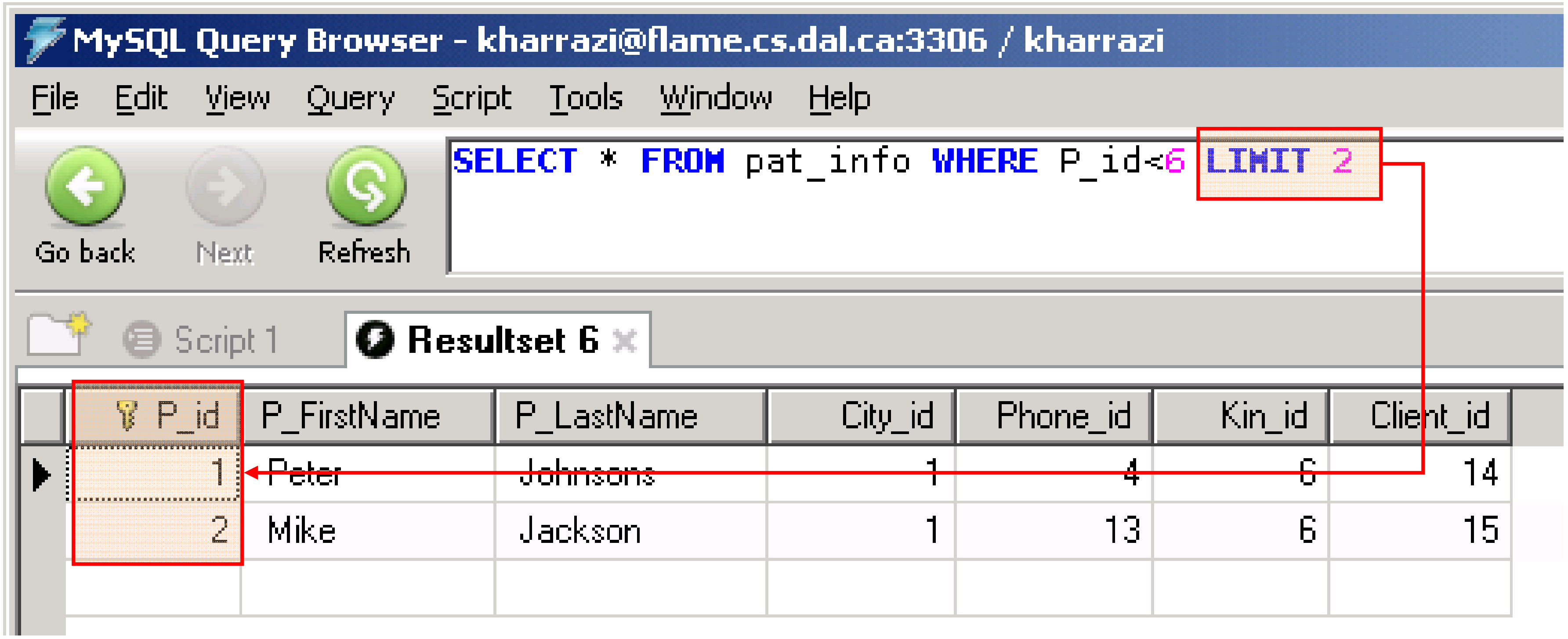

MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Script 1
Resultset 6

```
SELECT * FROM pat_info WHERE P_id<6
```

| P... | P_FirstNa... | P_LastNa... | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 |
| 2 | Mike | Jackson | 1 | 13 | 6 | 15 |
| 3 | Sara | Henson | 3 | 6 | 2 | 16 |
| 4 | John | McDonnald | 5 | 8 | 3 | 17 |
| 5 | Michael | Robinson | 1 | 13 | 6 | 18 |

**Limited Results**

## SQL: WHERE Statement (cont.)

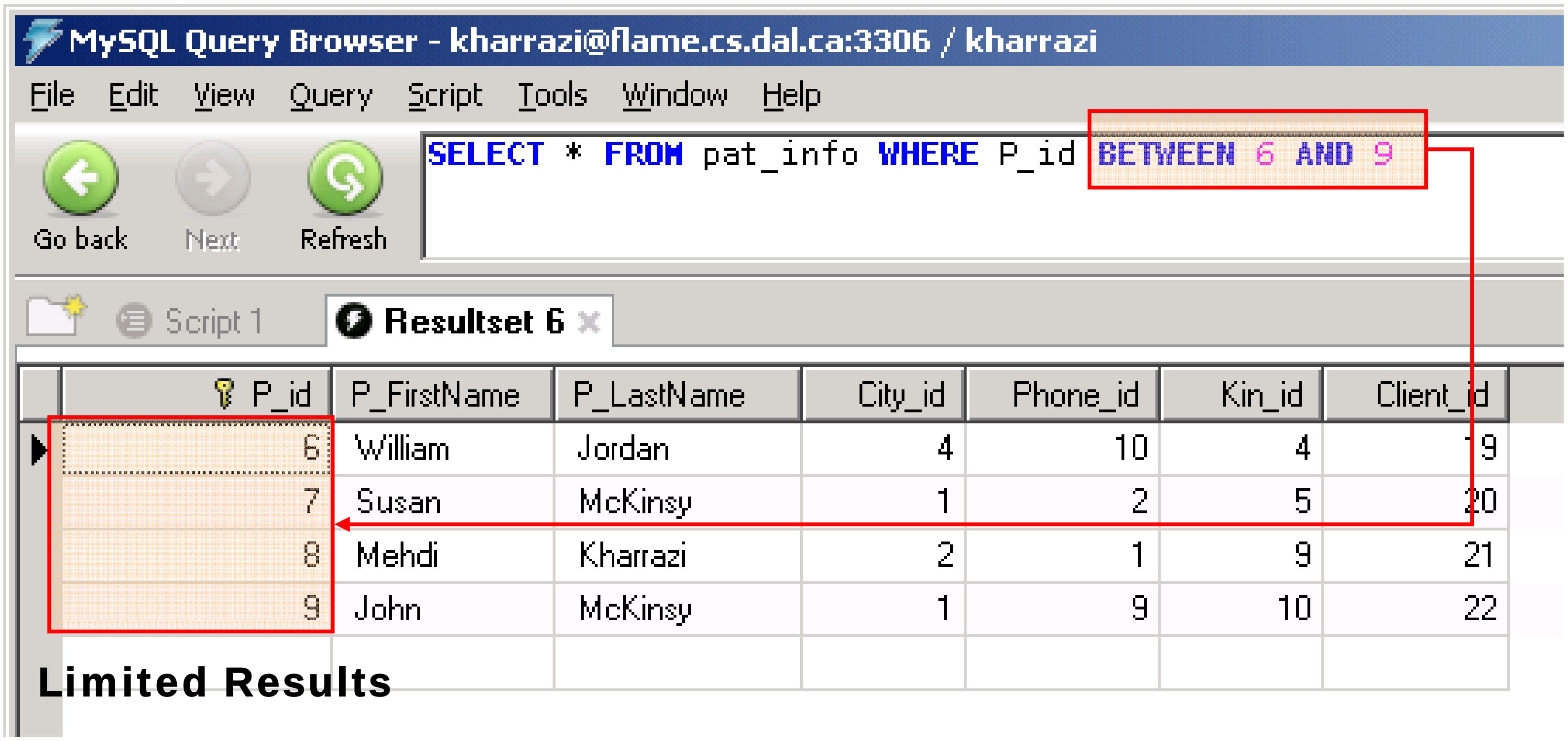

MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Script 1
Resultset 6

```
SELECT * FROM pat_info WHERE P_id<6 LIMIT 2
```

| P_id | P_FirstName | P_LastName | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 1 | Peter | Johnsons | 1 | 4 | 6 | 14 |
| 2 | Mike | Jackson | 1 | 13 | 6 | 15 |

**Only the first 2 results is fetched**

## SQL: WHERE Statement (cont.)

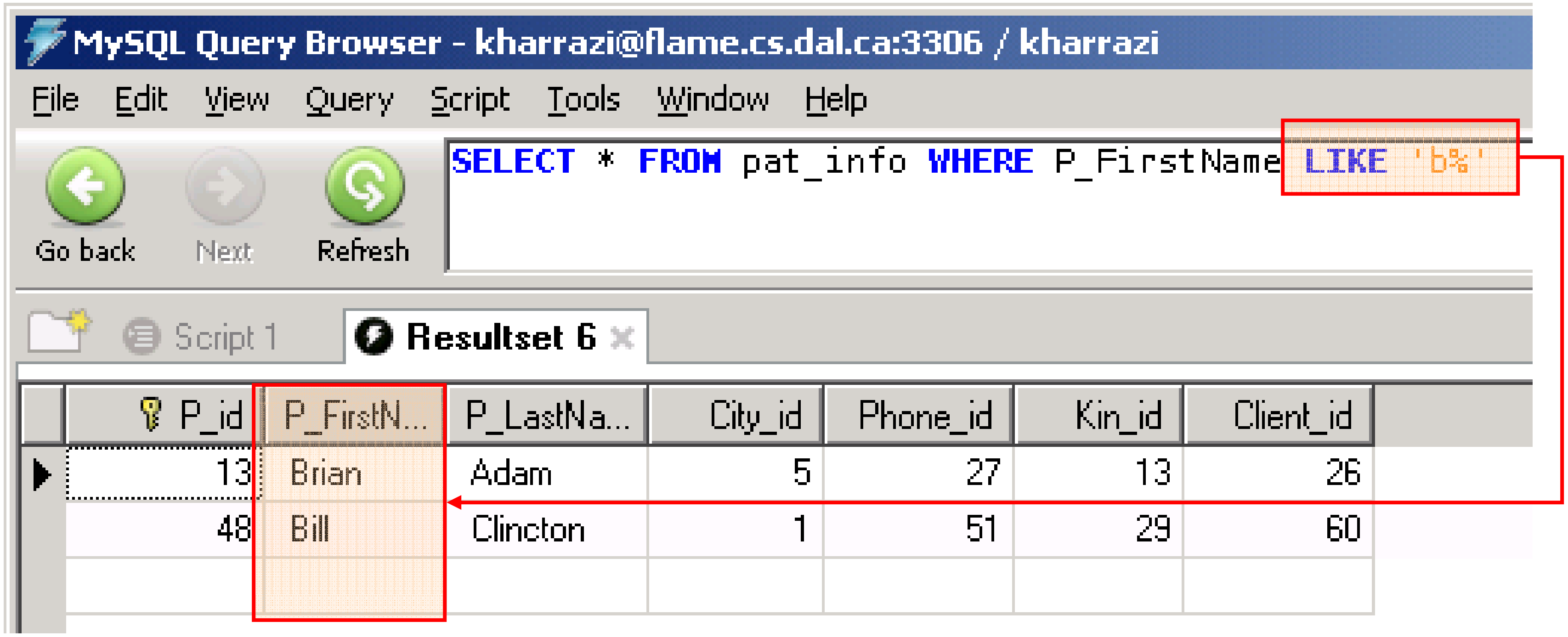

MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Script 1
Resultset 6

```
SELECT * FROM pat_info WHERE P_id BETWEEN 6 AND 9
```

| P_id | P_FirstName | P_LastName | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 6 | William | Jordan | 4 | 10 | 4 | 19 |
| 7 | Susan | McKinsy | 1 | 2 | 5 | 20 |
| 8 | Mehdi | Kharrazi | 2 | 1 | 9 | 21 |
| 9 | John | McKinsy | 1 | 9 | 10 | 22 |

**Limited Results**

## SQL: WHERE Statement (cont.)


MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Script 1
Resultset 6

```
SELECT * FROM pat_info WHERE P_FirstName LIKE 'b%'
```

| P_id | P_FirstN... | P_LastNa... | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 13 | Brian | Adam | 5 | 27 | 13 | 26 |
| 48 | Bill | Clincton | 1 | 51 | 29 | 60 |

**All start with 'B'**

## SQL: WHERE Statement (cont.)

- {‘Bob’; ‘Bill’; ‘Brian’; ‘Barnaby’; ‘Barclay’; ‘Barb’; ‘Gabriel’; ‘Jacob’}

o LIKE ‘b’ → null

o LIKE ‘b%’ → {‘Bob’; ‘Bill’; ‘Brian’; ‘Barnaby’; ‘Barclay’; ‘Barb’}

o LIKE ‘%b’ → {‘Bob’; ‘Barb’; ‘Jacob’}

o LIKE ‘%b%’ →

{‘Bob’; ‘Bill’; ‘Brian’; ‘Barnaby’; ‘Barclay’; ‘Barb’; ‘Gabriel’; ‘Jacob’}

o LIKE ‘b%b’ → {‘Bob’; ‘Barb’}

o LIKE ‘%b%b%’ → {‘Bob’; ‘Barnaby’; ‘Barb’}

o LIKE ‘%bb%’ → null

## 9. SQL: AND & OR Clause

- AND & OR join two or more conditions in a WHERE clause. The AND operator displays a row if ALL conditions listed are true. The OR operator displays a row if ANY of the conditions listed are true.

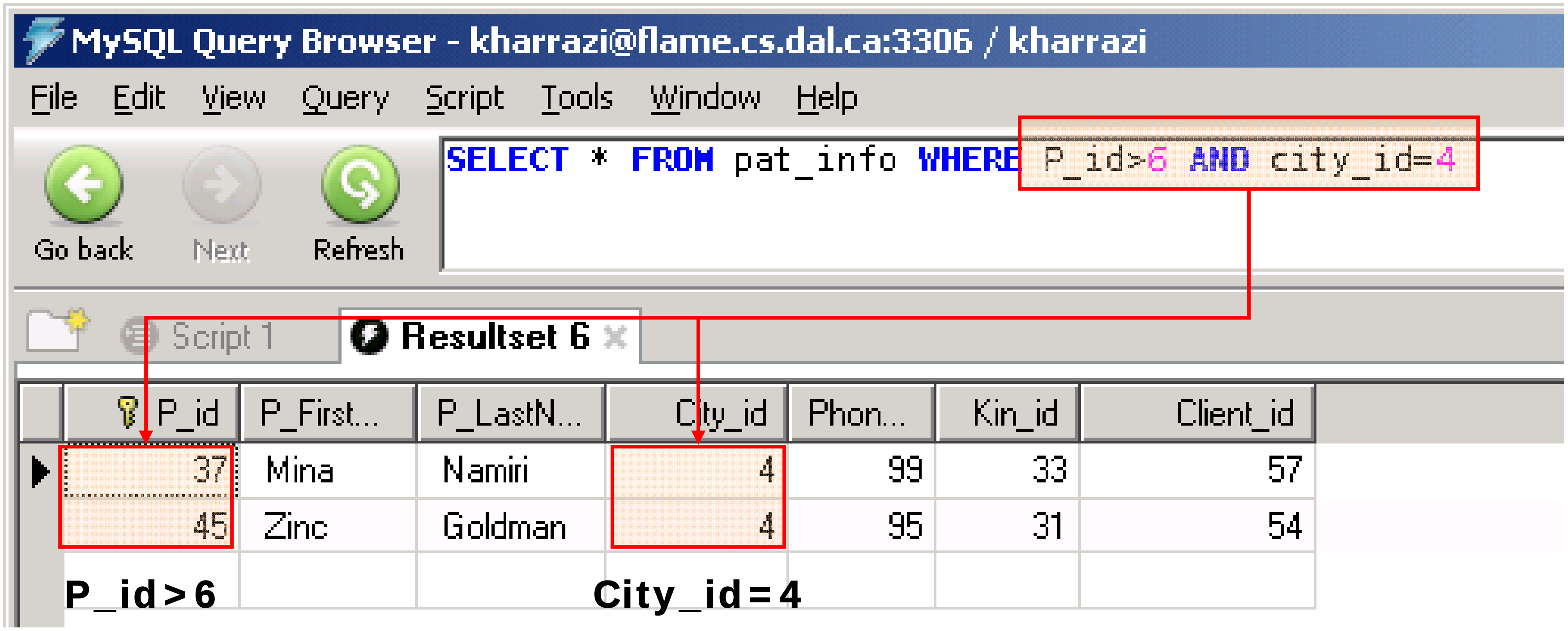
- Syntax:

```
SELECT column FROM table WHERE column operator value
AND column operator value OR column operator value
```

- Examples:

```
SELECT * FROM pat_info WHERE P_id>6 AND City_id=4

SELECT * FROM pat_info WHERE City_id=3 OR City_id=4
```

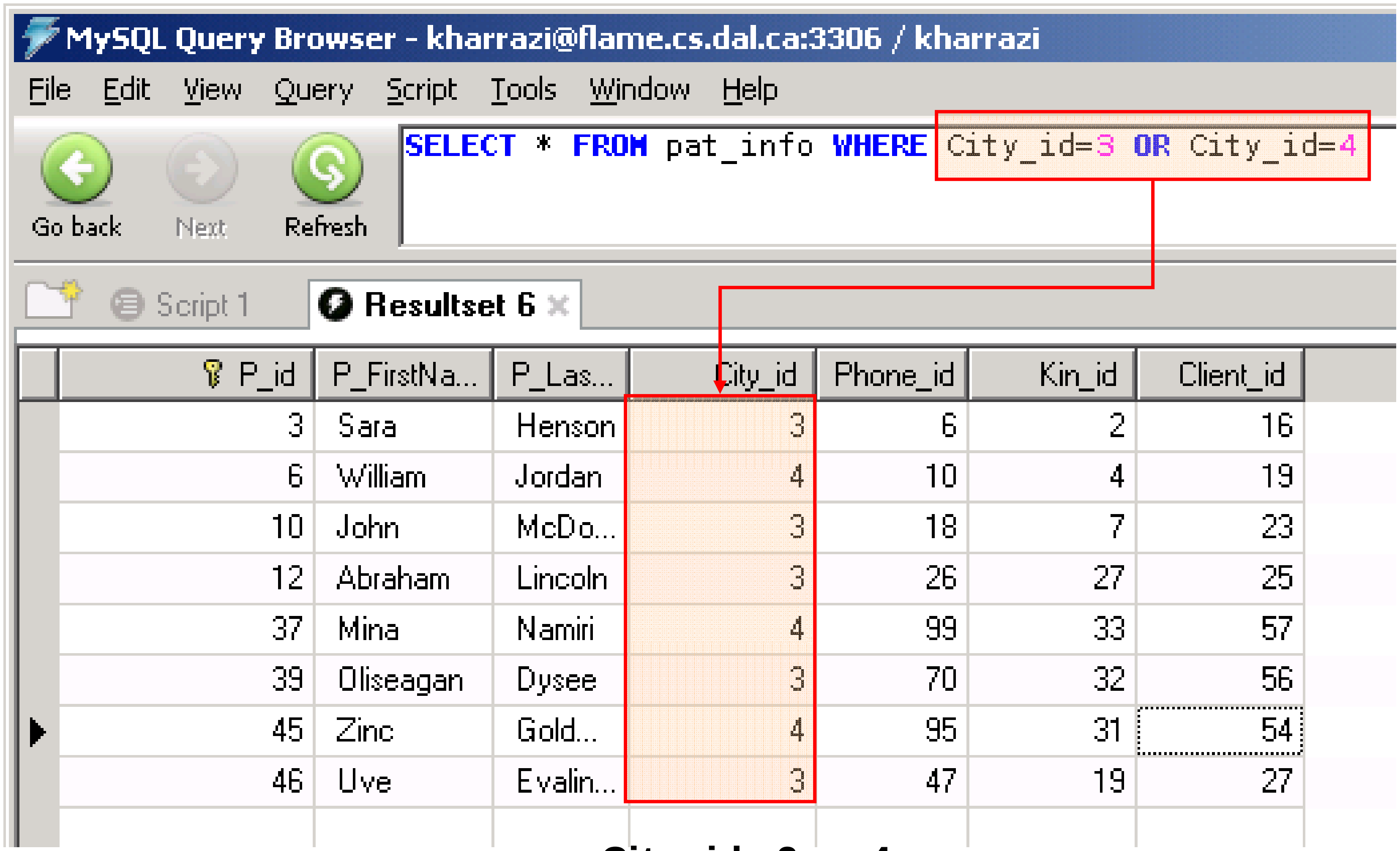
## *SQL: AND & OR (cont.)*


MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Script 1
Resultset 6
P_id>6
City_id=4

```
SELECT * FROM pat_info WHERE P_id>6 AND city_id=4
```

| P_id | P_First... | P_LastN... | City_id | Phon... | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 37 | Mina | Namiri | 4 | 99 | 33 | 57 |
| 45 | Zinc | Goldman | 4 | 95 | 31 | 54 |

## *SQL: AND & OR (cont.)*


MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Script 1
Resultset 6

```
SELECT * FROM pat_info WHERE City_id=3 OR City_id=4
```

| P_id | P_FirstNa... | P_Las... | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 3 | Sara | Henson | 3 | 6 | 2 | 16 |
| 6 | William | Jordan | 4 | 10 | 4 | 19 |
| 10 | John | McDo... | 3 | 18 | 7 | 23 |
| 12 | Abraham | Lincoln | 3 | 26 | 27 | 25 |
| 37 | Mina | Namiri | 4 | 99 | 33 | 57 |
| 39 | Oliseagan | Dysee | 3 | 70 | 32 | 56 |
| 45 | Zinc | Gold... | 4 | 95 | 31 | 54 |
| 46 | Uve | Evalin... | 3 | 47 | 19 | 27 |

**City_id=3 or 4**

## 10. SQL: IN Clause

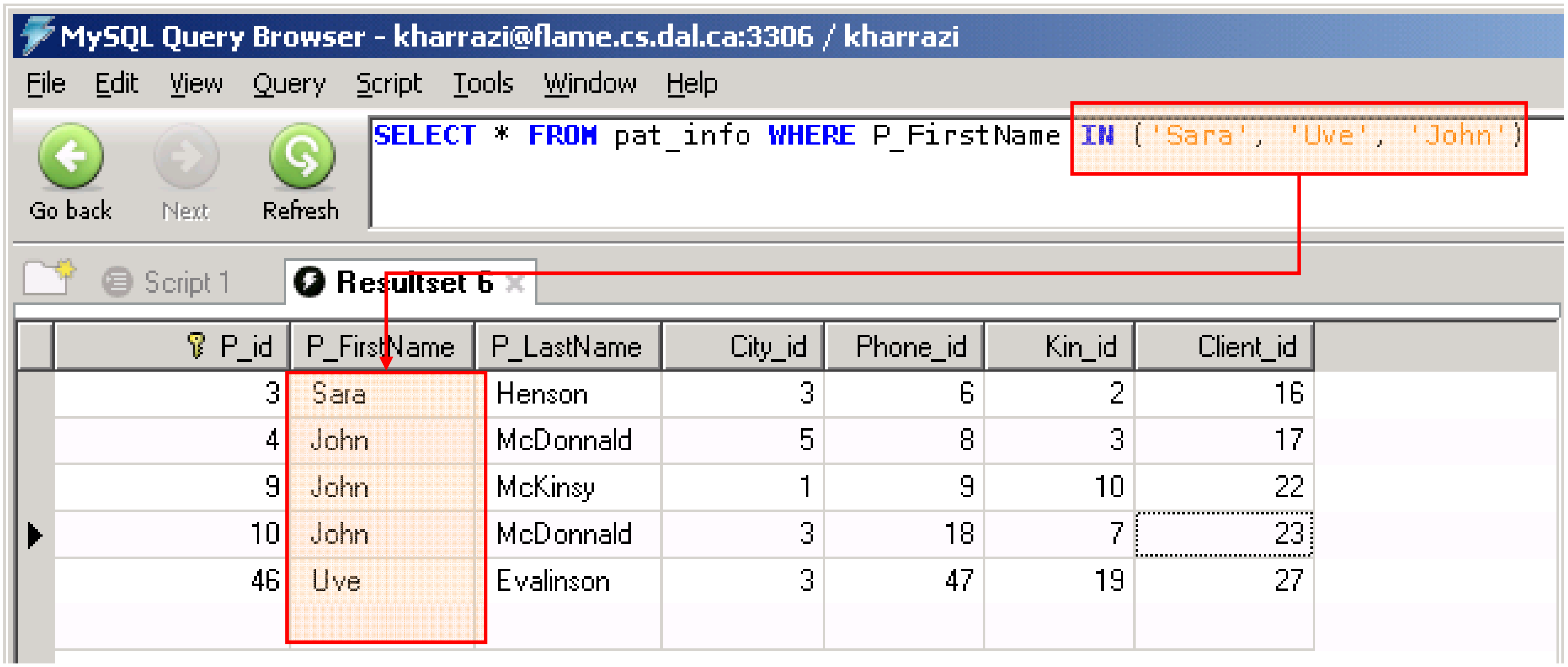
- The IN operator may be used if you know the exact value you want to return for at least one of the columns.
- Syntax:

```
SELECT column FROM table
WHERE column IN (value1, value2,… )
```

- Examples:

```
SELECT * FROM pat_info WHERE P_FirstName
IN ('Sara', 'Uve', 'John')
```

## SQL: IN (cont.)


MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Script 1
Resultset 6

```
SELECT * FROM pat_info WHERE P_FirstName IN ('Sara', 'Uve', 'John')
```

| P_id | P_FirstName | P_LastName | City_id | Phone_id | Kin_id | Client_id |
|---|---|---|---|---|---|---|
| 3 | Sara | Henson | 3 | 6 | 2 | 16 |
| 4 | John | McDonnald | 5 | 8 | 3 | 17 |
| 9 | John | McKinsy | 1 | 9 | 10 | 22 |
| 10 | John | McDonnald | 3 | 18 | 7 | 23 |
| 46 | Uve | Evalinson | 3 | 47 | 19 | 27 |

**P_FirstName is either:**
**‘Sara’, ‘Uve’ or ‘John’**

## 11. SQL: ORDER BY Clause

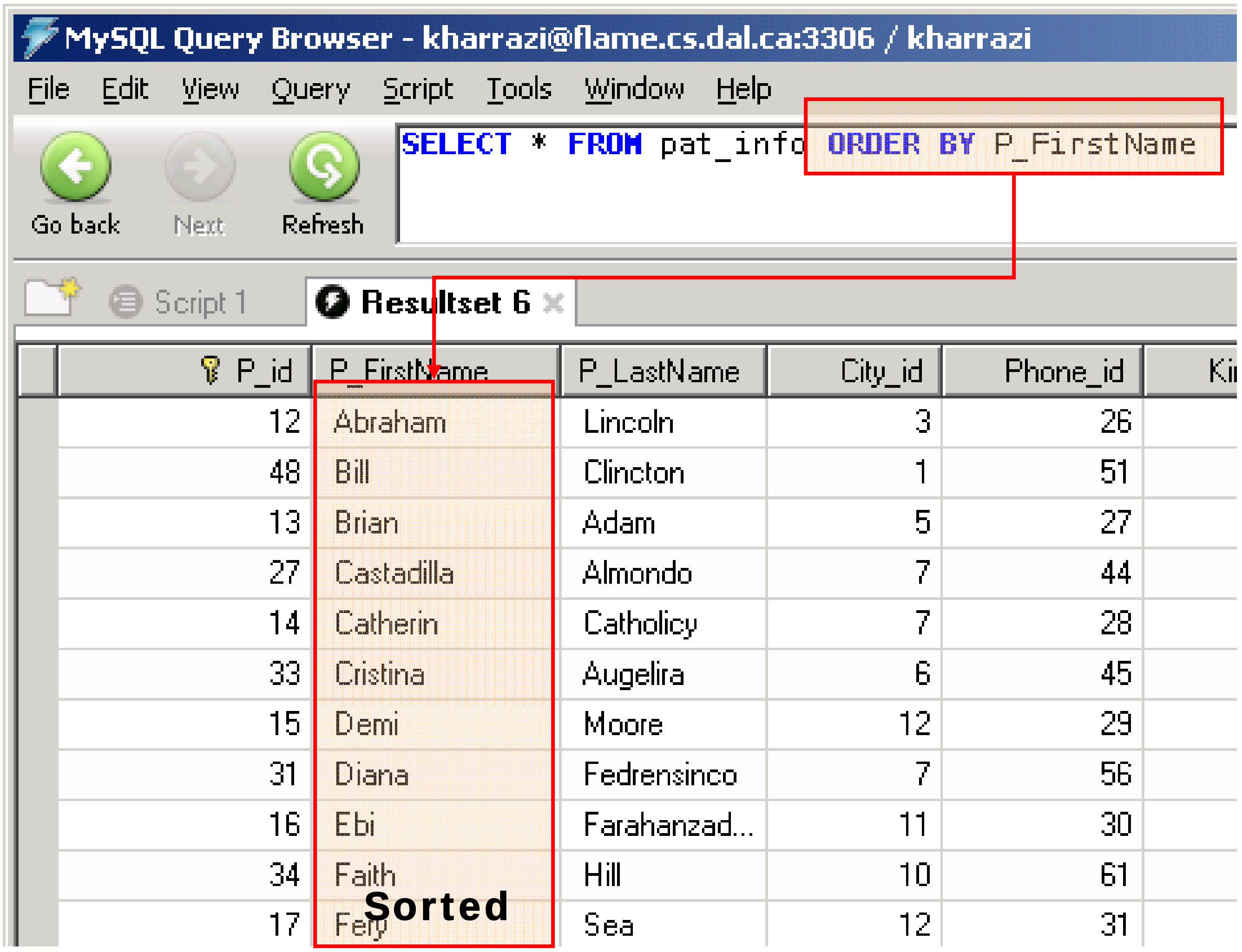
- The ORDER BY clause is used to sort the rows.
- Syntax:

```
SELECT column FROM table ORDERED BY column DESC/ASC
```

- Examples:

```
SELECT * FROM pat_info ORDER BY P_FirstName

SELECT * FROM pat_info ORDER BY P_FirstName DESC
```

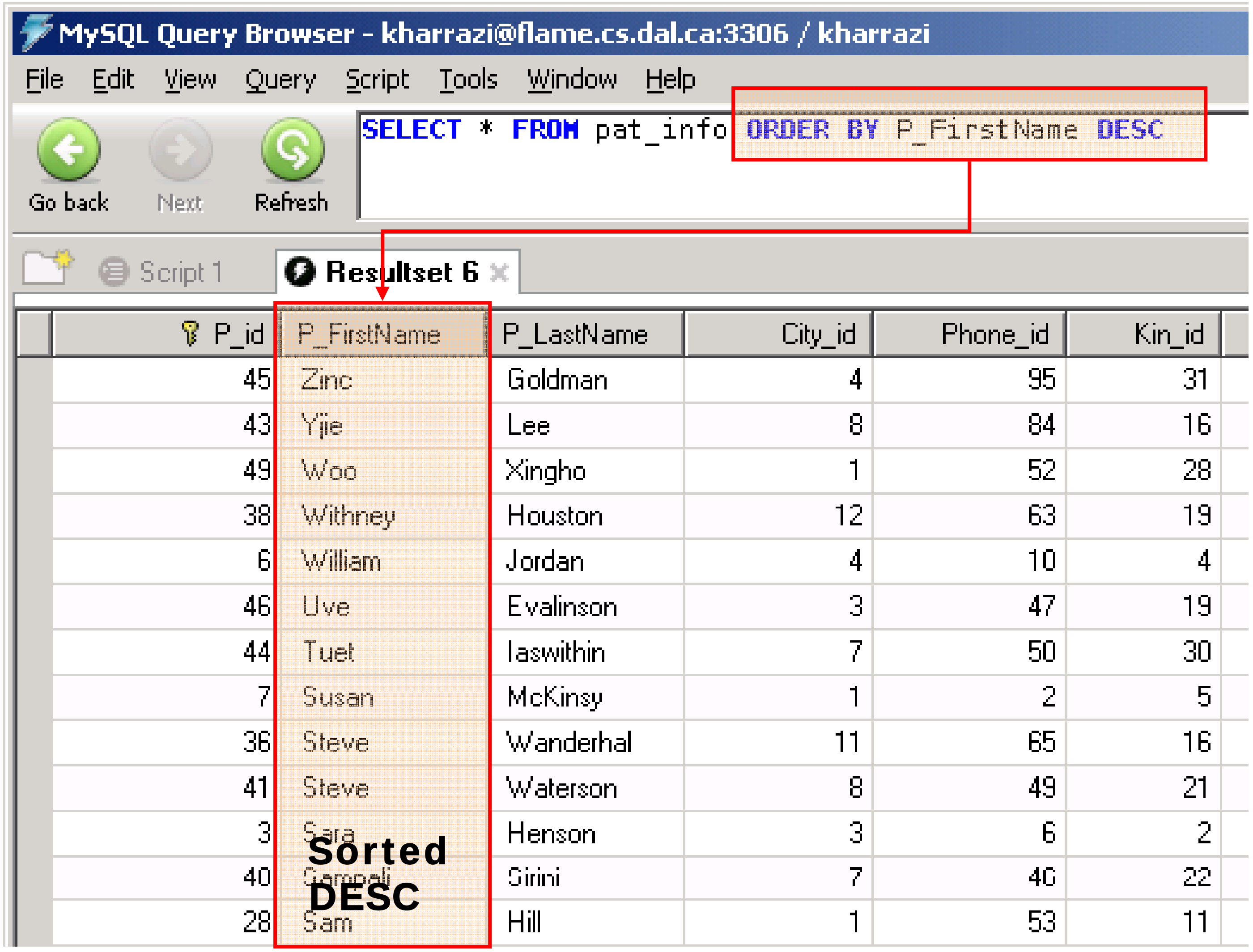
## *SQL: ORDER BY (cont.)*



```
SELECT * FROM pat_info ORDER BY P_FirstName
```

| P_id | P_FirstName | P_LastName | City_id | Phone_id | Ki |
|---|---|---|---|---|---|
| 12 | Abraham | Lincoln | 3 | 26 | |
| 48 | Bill | Clincton | 1 | 51 | |
| 13 | Brian | Adam | 5 | 27 | |
| 27 | Castadilla | Almondo | 7 | 44 | |
| 14 | Catherin | Catholicy | 7 | 28 | |
| 33 | Cristina | Augelira | 6 | 45 | |
| 15 | Demi | Moore | 12 | 29 | |
| 31 | Diana | Fedrensinco | 7 | 56 | |
| 16 | Ebi | Farahanzad... | 11 | 30 | |
| 34 | Faith | Hill | 10 | 61 | |
| 17 | Fery | Sea | 12 | 31 | |

## SQL: ORDER BY (cont.)


MySQL Query Browser - kharrazi@flame.cs.dal.ca:3306 / kharrazi
File Edit View Query Script Tools Window Help
Go back
Next
Refresh
Script 1
Resultset 6
Sorted DESC

```
SELECT * FROM pat_info ORDER BY P_FirstName DESC
```

| P_id | P_FirstName | P_LastName | City_id | Phone_id | Kin_id |
|---|---|---|---|---|---|
| 45 | Zinc | Goldman | 4 | 95 | 31 |
| 43 | Yjie | Lee | 8 | 84 | 16 |
| 49 | Woo | Xingho | 1 | 52 | 28 |
| 38 | Withney | Houston | 12 | 63 | 19 |
| 6 | William | Jordan | 4 | 10 | 4 |
| 46 | Uve | Evalinson | 3 | 47 | 19 |
| 44 | Tuet | laswithin | 7 | 50 | 30 |
| 7 | Susan | McKinsy | 1 | 2 | 5 |
| 36 | Steve | Wanderhal | 11 | 65 | 16 |
| 41 | Steve | Waterson | 8 | 49 | 21 |
| 3 | Sara | Henson | 3 | 6 | 2 |
| 40 | Campali | Sirini | 7 | 40 | 22 |
| 28 | Sam | Hill | 1 | 53 | 11 |

## Summary

- Database Overview
- Relational Databases
- Installing MySQL
- Command line MySQL
- MySQL GUI Tools
- SQL Introduction
- SQL: SELECT
- SQL: WHERE

## Next Session

- SQL: INSERT
- SQL: UPDATE
- SQL: DELETE
- SQL: Joining and Keys (Inner/Left/Right Join)
- SQL: GROUP BY & HAVING
- SQL: Functions

## Exercise

- Please refer to the available text file in the slides section for this session on the course website:
- http://info510.com/core/public_page.php?page_name=slides