SQL Workshop
Session 1

Monday 15 July 2013

General Assembly

Welcome

- Class will focus on:
  - SQL syntax
  - Using SQL to explore and analyze data
- We will use exercises away from class to build up skill and familiarity

Who are you and why are you here?

- What sort of companies do you work for?
- What are your roles?
- What are you using for analysis?
- What are you interested in?
  - Replicating Excel analyses with SQL
  - Running queries
  - Orientation to SQL and database concepts

Who Am I

- Entrepreneur / consultant
  - Data management
  - CRM migration and integration
- Self-taught on LAMP(ython) stack
    - Plus some DNS, bash scripting, XML / XSLT
- Prior professional experience:
  - Office of Management and Budget
  - Investment banking (JP Morgan, UBS, boutique)
  - MBA, University of Chicago

Roadmap – Session 1

- Preliminaries
- Orientation to SQL databases
- Software installation
- Install sample database
- Query syntax basics

Tools

- Database Software
  - Some means of running SQL queries is needed
  - MySQL, Postgres, Sql Server, Oracle
    - Microsoft Access is NOT recommended
    - If you have questions about your SQL implementation, see me
  - MySQL is free, download from mysql.com
    - MySQL
      - Both “Community Server” and startup package
    - MySQL Workbench
- Text editor – you may or may not need one
  - Wordpad, Text Edit are very basic examples
  - Textmate, Sublime Text are more featured tools
    - Windows: Notepad++
  - Do NOT use a word processor (e.g. Word)
  - Do NOT use ‘rich text format’ / *.rtf
  - SQL code should just be ASCII text, with NO formatting information

Get A Database Program

- Some means of running SQL queries is needed
  - MySQL, Postgres, Sql Server, Oracle
    - Microsoft Access is NOT recommended
    - If you have questions about your SQL implementation, see me
  - MySQL is free, download from mysql.com
    - MySQL
      - Both “Community Server” and startup package
    - MySQL Workbench
  - Or, sign up for db4free.net

What Is SQL?

What is SQL?

- There are many SQLs
  - MySQL / Postgres / SQLite / Microsoft Access / Oracle / Sybase / etc etc etc
- Each is an implementation of a standard
  - "SQL" is basically a program's promise to behave in a defined way in response to a particular query for a particular set of data

“Client / Server”

- Most implementations are actually two programs
  - One is a front end / client, giving us access to a server and showing its outputs
    - MySQL Workbench
  - The other is a back end / server, holding the data and processing the queries
    - MySQL
  - MySQL Workbench is to MySQL as Chrome browser is to an HTTP server

Client / Server Implications

- We can access existing data
  - We don't need data dumps to do analysis
    - We do need some sensible and safe access to the existing databases
  - If your production database has a million records, you can analyze them all without downloading them all
- We can access remote data
  - If your servers are on Rackspace or Amazon you can still get at the data
- Programs can access the data
  - Languages like Ruby, Python, PHP all have tools for interacting with databases
  - They act as clients, but getting data for (say) serving web pages rather than display to an analyst

Getting Started

Install Sample Database

- Download files from https://github.com/chernevik/sql_workshop
  - Click 'Download ZIP' to get an archive of the files
  - Open / unpack archive
  - MySQL:
    - In MySQL client, run script 'setup_sampdb.sql'
  - Other implementations:
    - Tell me what they are and I'll provide scripts to load from CSV

Run a Query

- In your client, type:
  - 'SELECT 2 + 2;'
  - Run that
- Note that queries are synonymous with 'commands'
  - We use them to get information about the databases and tables available
  - We can test out some functions and expressions with them

Run A Query to retrieve data

- In your client, type:
  - ‘SELECT * FROM sampdb.student’
  - Run that

Run a Query from a file

- In the downloaded materials, find:
  - sessions/01/simple.sql
- Open this in your client's query editor
- Run it

Aside: Organizing Your Environment

- Queries are typically written in text files
  - Again, no word processing, no .doc or .rtf files
- Make a place on your computer to hold your files – data, queries, drafts
  - Your files can expand quickly, as you copy files to experiment or to make variations
  - Use of subdirectories helps keep things organized
  - Code repository tools like Git and Mercurial make a lot of sense
    - This would be a great time to begin learning and using these

Query Syntax

Simple Query

```
SELECT name, sex, student_id
FROM sampdb.student;
```

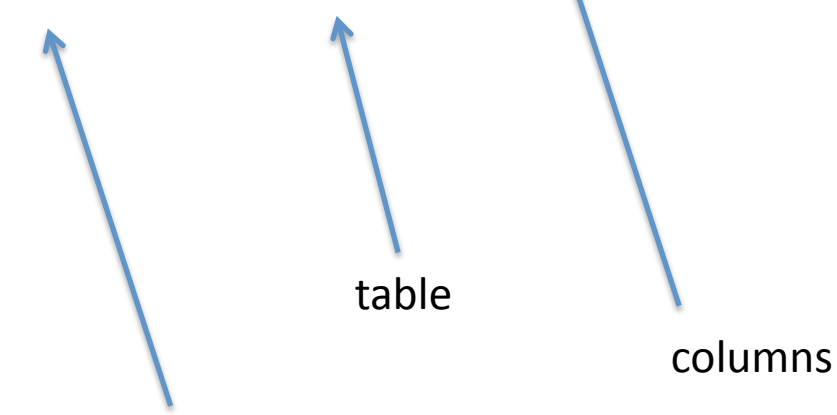
- Gets all fields of all records in table

Let's unpack that:

- "database"
  - A program concept for holding "tables"
  - Comparable to a folder in your computer's file system
- "table"
  - A program concept holding data about instances of a particular entity type – students, presidents, exams
  - These data are held in "records" or "rows"
- "record"
  - Data about a particular instance – about a particular student or particular movie
  - Data is in a collection of fields or columns
  - Comparable to a row in a spreadsheet
- "Column" / "field"
  - An attribute of an entity type about which data is collected – a student's name, or sex, or the state of origin of a president
  - Comparable to a cell in a spreadsheet row

Simple Query

```
SELECT name, sex, student_id
FROM sampdb.student;
```



We could also use:
SELECT * FROM sampdb.student
- << * >> is a wildcard meaning “all columns”

Filtering the results

```
SELECT
   *
FROM
   sampdb.student
WHERE
   sex = “F”;
```

- Notice the reorganization of the text for clarity
- WHERE clause defines conditions imposed on returned rows

An Aside on “query files”

- MySQL clients are looking for text only – e.g. files containing only ASCII or Unicode
- Don’t use Word, don’t use “rich text format”
  - These introduce characters that will confuse the MySQL client
- Use TextEdit, SublimeText, MacVim
  - vi !!!
  - A “text editor” of some sort

Filtering the results by ‘tuples’

```
SELECT
   first_name,
   last_name,
   state

FROM
   sampdb.president

WHERE
   (last_name, state) IN

   (
      ('Adams', 'MA')
   )
;
```

- A ‘tuple’ is a set of fields
  - A table is actually a set of tuples
- The IN operator allows us to see if a particular group of fields has a particular set of values

Analyzing the results

```
SELECT
    COUNT(student_id)
FROM
   sampdb.student
;
```

- COUNT() is a “function”
- Functions return information about or manipulate contents of a field
- Some functions apply only to field of single record
- Others, like COUNT, apply to the field for multiple records
- COUNT returns the number of not NULL values in the field
  - We’re ignoring “NULL” for now

Analyzing the results -- Aggregation

```
SELECT
    sex,
    COUNT(student_id),
    COUNT(sex)
FROM
   sampdb.student
GROUP BY
   sex
      ;
```

- COUNT() is an a “aggregate” function
- We can segment the records on which it works
- GROUP BY clause gathers up records by their value in a given field
- The function can be applied to any field, not just that by which records are grouped

Analyzing the results -- Prettify

```
SELECT
    sex AS Gender,
    COUNT(student_id) AS “ID Count”,
    COUNT(sex) AS gender_count
FROM
   sampdb.student
GROUP BY
   gender
   -- sex

  -- this is a comment
  -- all on this line after “--” is ignored

    # so is this, for MySQL
    /*
       And this, for MySQL
    */
;
```

- Inside the column expressions we’ve added “as [name]” clauses
- These are “alias” terms
- The results aren’t changed but their display is
- We can use the alias to specify the column

Functions

```
SELECT
   first_name,
   last_name,
   state,
   ROUND(DATEDIFF(death, birth) / 365, 1)
 AS age

FROM
   sampdb.president

ORDER BY
   age DESC

;
```

- DATEDIFF() is a defined MySQL function returning the number of days between two dates
- ROUND() controls decimal points
  - (we need this here to make some later code work)

More Functions, Aggregated

```
SELECT
  state,
  MAX(
    ROUND(DATEDIFF(death, birth) / 365, 1)
  )
 AS max_age,
  COUNT(*)

FROM
  sampdb.president

GROUP BY
  state

ORDER BY
  max_age DESC

;
```

- MAX() is an aggregate function, returing the largest of a set of values
- Here the set is the values calculated for a group of rows
- GROUP BY defines the rows
- We drop the name fields because they aren't meaningful when rows are GROUPed