The Structured Query Language (SQL) is a standard for manipulating data in relational database.

In this class we will submit SQL commands to MySQL, an open-source SQL development environment.

Login into the MySQL web page
Get Your Own MySQL

- We’ll be running MySQL and MySQL Query Browser within PortableApps. See my handout for installing MySQL and Query Browser in PortableApps.

- Though we won’t be doing this, you can install MySQL on your desktop.
  - dev.mysql.com
  - Download the latest stable version of the MySQL command line program and the MySQL Query Browser.
    - The installation files can be found in the 152-175 folder on the I: drive.
    - You should also get a copy of my database Transfer program from the I: drive. This program will allow you transport your databases between home and school.
    - See the Installing and Configuring MySQL handout on my web page for guidance on how to install, and primarily how to configure, the programs.

  - This option is not as portable as our PortableApps option.

Documentation Standards

In these notes:

- SQL keywords will appear in all caps, bold and blue
  - In SQL, keywords do not have to be capitalized. They are capitalized in these notes for clarity.
- Table names will appear in all caps, black
- Field names will appear in TitleCase
- Text entered in italics must be replaced with actual table, field names
- Objects in [square brackets] are optional.
### Notes

#### Creating a Database

- MySQL requires that a database exist before you can add tables to it.
- MySQL provides the Create Database command to do this.

```sql
CREATE DATABASE databasename;
```

- MySQL Query Browser will not display the new database in the Schemata window until you refresh (right-click in Schemata)
- All databases are stored in:
  - E:\XAMPP\MySQL\Data
  - C:\Program Files\MySQL\MySQL Server 6.0\Data if you’ve installed on the desktop.
  - Each database is stored in its own folder.
  - Unfortunately, you can NOT simply copy these folders to transfer the database to a new location.
  - See the **Backing Up and Restoring Databases** handout on my website for information on how to transfer a database from one location to another.

### MS SQL

- This command doesn't seem to create a database. Use the GUI tools to create a new database.

- Once you have created databases you can display a list of the available databases by entering:

```sql
SHOW DATABASES;
```

- MySQL comes with some databases.
  - Test is for experimenting with MySQL. Feel free to delete (drop) this database if you wish.
  - The other databases are used by the MySQL server itself to track user and databases. DO NOT delete these.

- If you no longer need a database you can delete it by entering:

```sql
DROP DATABASE databasename;
```

- There is no recovery for this command. Make sure you enter the correct database name.
- Again, you’ll need to refresh the Query Browser Schemata window before the database disappears.

### Activity

Create the Premiere Products and Henry Books databases.

Create database labs

Create database labs

Drop premiereproducts and henrybooks

Create database labs

USE labs;
### Notes

- You can also create and drop databases (schemata) using the Schemata pane in Query Browser
  - Right-click in the Schemata pane and choose Create New Schema to create a new database.
  - Right-click any database object and choose Drop Schema to delete the database.

### Activity

**Creating a Table**

```sql
CREATE TABLE tablename
(fieldname datatype
[NOT NULL][PRIMARY KEY][AUTO_INCREMENT]
fieldname datatype,
fieldname datatype,
repeat for all fields);
```

- SQL is a *free format* command language, which means you can insert extra spaces and carriage returns wherever you want. Take advantage of this to make your commands as readable as possible.
- SQL commands can be entered in any case (upper, lower, mixed)

- This command tells SQL to create a new table with the name you supplied.
- This command must also include the list of fields in the table, and the data types of those fields.
- The list of fields must be surrounded by (parenthesis)
- Most implementations of SQL require a semicolon at the end of the command.
- The **PRIMARY KEY** designates this field as the primary key for the table.
  - This technique can only be used if there is one primary key field. See below to designate multi-field primary keys.
- The **NOT NULL** clause specifies that this field may not be left blank (NULL) or changed to a NULL value.
  - **PRIMARY KEY** fields are automatically designated **NOT NULL**.
- Designating a Multi-Field Primary Key
  - List all the fields in the table, but don’t include the **PRIMARY KEY** designation.
  - After the last field, enter another comma, then enter the command **PRIMARY KEY**
  - Then, in (parenthesis), list the field names included in the key.
  - **PRIMARY KEY** *(fieldname, fieldname, etc)*
MS SQL (Visual Studio 2005 only)

- To create composite keys in MS SQL use the following syntax after the field list

  ```
  CREATE TABLE tblName
  (fldname fldtype, etc.
   fldname fldtype,
   CONSTRAINT constraintname
   PRIMARY KEY (keyfield, keyfield) )
  ```

  - The constraint name can be anything. We will not be referring to it later.

- The `AUTO_INCREMENT` clause is optional.
  - This option can only be designated on `PRIMARY KEY` fields.
  - The data type is normally designated `INTEGER`.
  - Values in this field are automatically incremented whenever new records are added to the table.
  - If records are deleted, their numbers are not reused.
  - You must use `INSERT` version 2 to add records to tables with `AUTO_INCREMENT` fields.

MS SQL

- Instead of `AUTO_INCREMENT`, use `IDENTITY(1,1)`
  - The 1,1 is optional
  - It designates the starting number and increment value

- `IDENTITY` fields must be of type Int
- **SQL Data Types**
  - **NCHAR** \((n)\)
    - text data, \(n\) represents max characters
    - use only if the field will always be full
    - (fills with spaces otherwise)
  - **VARCHAR** \((n)\)
    - text data, \(n\) represents max characters but the number of characters can be less (no fill)
  - **DATE**
    - date and time data
  - **DATETIME**
    - date and time data
  - **DATE**
    - date only (times provided are ignored)
  - **TIME**
    - time only (dates provided are ignored)
  - **SMALLINT**
    - whole number (±32K)
  - **INT** or **INTEGER**
    - whole number (±2G)
  - **DECIMAL**\((p,q)\)
    - number with decimal places
    - \(p\) designates total number of digits (point not included)
    - \(q\) designates the number of digits after decimal point
  - **BIT** or **BOOL** or **BOOLEAN**
    - Use for Yes/No True/False fields
    - Enter 1 or True; 0 or False when inserting data

- **Examples:**
  ```sql
  CREATE TABLE tblCity
  ( CityCode INT PRIMARY KEY AUTO_INCREMENT,
    CityName VARCHAR (20) NOT NULL,
    Population INT,
    NumEmployees SMALLINT,
    TaxRate DECIMAL(5,3),
    LastCensus DATE,
    PolutionProblem BOOLEAN );
  ```

- **Create tables:**
  - **COMPUTER**
    - **CompID** (auto),
    - **MfgName**(10),
    - **MfgModel**(15),
    - **ProcType**(6)
  - **PC**
    - **TagNum**(5),
    - **CompID**(4),
    - **EmpNum**(3),
    - **Location**(15)
  - **SOFTWARE**
    - **SWID**(auto),
    - **SWName**(25),
    - **SWVer** (D4,2),
    - **SWType**(20),
    - **SWCost**(D6,2)
  - **INSTALL**
    - **SWID**(4),
    - **TagNum**(5),
    - **InstDate**, 
    - **InstCost**(D6,2)
<table>
<thead>
<tr>
<th>Notes</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Listing the Tables in the Database</strong></td>
<td>Try it.</td>
</tr>
<tr>
<td>* SHOW TABLES;</td>
<td></td>
</tr>
<tr>
<td>• Lists all the tables in the database</td>
<td></td>
</tr>
<tr>
<td>• Tables are also shown in the Schemata pane (refresh if necessary)</td>
<td></td>
</tr>
<tr>
<td><strong>MS SQL</strong></td>
<td></td>
</tr>
<tr>
<td>• Tables are normally shown in the GUI</td>
<td></td>
</tr>
<tr>
<td>• Use the <code>sp_help</code> command to get a list of tables</td>
<td></td>
</tr>
<tr>
<td><strong>Showing a Table’s Structure</strong></td>
<td>Try it.</td>
</tr>
<tr>
<td>* DESCRIBE <code>tablename</code>;</td>
<td></td>
</tr>
<tr>
<td>• Shows each field’s name, data type, NULL acceptance, and</td>
<td></td>
</tr>
<tr>
<td>primary key status.</td>
<td></td>
</tr>
<tr>
<td>• Right-click, Edit on table in Schemata to see a GUI representation</td>
<td></td>
</tr>
<tr>
<td>of the table structure.</td>
<td></td>
</tr>
<tr>
<td><strong>MS SQL</strong></td>
<td></td>
</tr>
<tr>
<td>• Use the <code>sp_columns</code> <code>tablename</code> command instead.</td>
<td></td>
</tr>
<tr>
<td>• Or <code>sp_help</code> <code>tablename</code></td>
<td></td>
</tr>
</tbody>
</table>

**Removing a Table**

* DROP TABLE `tablename`;

- Deletes a table and all the data in it from the database (note, there
  is no delete verification—be careful!)
- Can also drop table from Schemata pane

Create table `junk` with a couple of junk fields.
Delete junk table.
Adding Data to a Table

`INSERT INTO tablename VALUES (‘chardata’, numdata, ‘2002-2-15’, NULL );`

- This command adds a new record to the table specified.
- Repeat the command to enter multiple records
- Data must be listed in the order the fields were entered into the database (CREATE TABLE)
- String data must be surrounded by ‘single quotes’ in MySQL
- Commas separate each data item (surrounding spaces not required)
- Note the date format for MySQL is different than listed in the text: ‘yyyy-mm-dd’
- Use hh:mm:ss format (military time) for TIME fields
  - Separate from date with a space in DATETIME fields
  - Seconds (ss) are optional
- To leave a field blank, enter the keyword NULL
  - Don’t try this on NOT NULL fields. SQL will ignore the entire record.
- SQL recognizes duplicate primary keys and ignores any record with a duplicate key.
- Data entry errors can cause SQL to ignore your command.

MS SQL

- MS SQL in Visual Studio 2005 seems to want a value for the identity fields.
  - This is not the case in SQL Server Mgt Studio
- Use Version 2 below (without the identity field) to insert data
### Notes

#### INSERT Command—Version 2
- Sometimes, you have a partial record’s data available. Instead of entering NULL a bunch of times, you can tell SQL to only fill selected fields

```sql
INSERT INTO tablename(fldname1, fldname2, fldname3, etc)
VALUES
(datafld1, datafld2, datafld3, etc);
```

- Be sure **not** to skip **NOT NULL** fields.
  - SQL actually enters the empty string into these fields (""") instead of setting them to NULL

- Inserting into **AUTO_INCREMENT** tables.
  - List all fields and values for those fields.
  - Leave the **AUTO_INCREMENT** field off the field list; SQL will automatically fill in the next number in the sequence.

#### Inserting Multiple Records
- You can insert more than one record at a time with the INSERT statement.
- After the first set of values, add a comma, then another set of values (in parenthesis), and another and another.

```sql
INSERT INTO tablename(fldname1, fldname2, fldname3, etc)
VALUES
(datafld1, datafld2, datafld3, etc),
(datafld1, datafld2, datafld3, etc),
(datafld1, datafld2, datafld3, etc),
```

**etc**

```sql
(datafld1, datafld2, datafld3, etc);
```
**Instructor’s Notes**

**Systems Implementation**

**Structured Query Language (SQL)**

<table>
<thead>
<tr>
<th>Notes</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MS SQL</strong></td>
<td></td>
</tr>
<tr>
<td>● Inserting multiple rows using this technique is not supported.</td>
<td></td>
</tr>
<tr>
<td>● Use the following command instead</td>
<td></td>
</tr>
</tbody>
</table>

```
INSERT INTO tablename 
SELECT (datafld1, datafld2, datafld3, etc), (list all fields except identity) 
UNION ALL
SELECT (datafld1, datafld2, datafld3, etc),
UNION ALL
SELECT (datafld1, datafld2, datafld3, etc),
etc
SELECT (datafld1, datafld2, datafld3, etc);
```

**Viewing Records**

```
SELECT * FROM tblname;
```

● This command shows all fields in all records.
● We will discuss the SELECT command in greater detail later

**Deleting Records**

```
DELETE FROM tablename 
WHERE fieldname = criteriavalue;
```

● This command deletes a record (or records) from the specified table that match the criteria.
● Warning, there is no way to undo this deletion and there is no delete verification.

● Example: 
```
DELETE FROM CITY 
WHERE CityName = 'Stevens Point';
```
● This command would delete all records where the city name is *Stevens Point*.

Enter a bogus record, then delete it.
Correcting Data Errors

- If there’s an error in one or more fields; values, you could delete the entire record, or, you could simply change the incorrect value(s).

```
UPDATE tablename
SET fieldname = newvalue
WHERE fieldname = criteriavalue;
```

- This command changes the value in a specified field, in a specified table, in the records that match the criteria, to a new value.
  - UPDATE tablename designates which table to make the changes to
  - SET fieldname = newvalue designates which field to change and what value to change it to (newvalue)
  - WHERE fieldname = criteriavalue designates which records to change: those where the specified field contains the appropriate value.

- Example:
  ```
  UPDATE CITY
  SET CityName = 'Stevens Point'
  WHERE CityCode = 'StPt';
  ```
  - This might be used to correct a typo (Stevns Pointe) in a record whose code is StPt.