SQL: the basics

About this bootcamp
- 4 weeks, 1.5 hrs per week
- NOT a regular course...
  - way of communication
  - enrollment
  - course eval
- SQL intro for an analyst
  - queries: SELECT basics
  - queries: JOINs
  - subqueries
  - making changes to the database

Why SQL?
- widely used
- hospitals, universities, businesses
- a powerful technology (for analysts)
- easy to audit and replicate
- a solid technology (for engineers)
- arguably easier to learn than a programming language

Open req. for a financial and strategic analyst position: primary responsibilities data analytics and spreadsheet modeling.

Seeking resumes of recent graduate with experience in database program/software SQL.

Preferred Candidate:
- Finance and Statistics backgrounds.
- MBA w/ <5yrs work experience.
- Early Leader (optional)

Deadline: Please forward potential candidate resumes by Tuesday, March 18, 2014.

“SQL is essential for a marketer”

David Togerson
Director of Information Strategy at gogo
US Cellular, Terradata, Diamond Management Consultants

“SQL is essential for a marketer”

“SQL is essential for a marketer”
About me

• Yina Jin
• PhD student in Computer Science
• http://www.cs.rochester.edu/~yjin18/
• yinajin@rochester.edu

A database

• A database is an organized collection of data
• in Excel — spreadsheets
• in relational databases — tables
• CRUD: create, read, update and delete

The Relational Model of Data

• Data is represented as a collection of two dimensional tables
  • A relation is a named two dimensional table
  • An attribute is a named column
  • A tuple is an unnamed row
    • Order of row and columns is undefined
    • Rows are unnamed
• Values of attributes are drawn from a domain
  • Eg: {0,1,2,...}, {Male, Female}, {string of characters}
• In practice: data types are limited by implementation

Questions?
### SQL Select Command

```
SELECT [DISTINCT] sel_expression [, sel_expression ...] FROM table_references
[WHERE condition]
[GROUPBY column [,column...]
[[HAVING condition]]
[ORDER BY columns [ASC|DESC]];```

### SELECT and FROM (vertical slices)
- selecting a column:
  ```sql
  SELECT price FROM car;
  ```
- selecting multiple columns:
  ```sql
  SELECT model, man, price FROM car;
  ```
- selecting all columns:
  ```sql
  SELECT * FROM car;
  ```
- Getting a discount:
  ```sql
  SELECT price*0.95 FROM car;
  ```
- Getting the length of model names:
  ```sql
  SELECT model, char_length(model) FROM car;
  ```
Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/</code></td>
<td>Division</td>
</tr>
<tr>
<td><code>-</code></td>
<td>Minus</td>
</tr>
<tr>
<td><code>not</code></td>
<td>Logical not</td>
</tr>
<tr>
<td><code>*</code></td>
<td>Multiplication</td>
</tr>
<tr>
<td><code>and</code></td>
<td>Logical and</td>
</tr>
<tr>
<td><code>like</code></td>
<td>Simple pattern matching</td>
</tr>
<tr>
<td><code>+</code></td>
<td>Addition</td>
</tr>
<tr>
<td><code>or</code></td>
<td>Logical or</td>
</tr>
<tr>
<td><code>regexp</code></td>
<td>Regular expression pattern matching</td>
</tr>
</tbody>
</table>


String and Numerical Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>char_length()</code></td>
<td>Number of characters in a string</td>
</tr>
<tr>
<td><code>concat()</code></td>
<td>Concatenate strings</td>
</tr>
<tr>
<td><code>lower()</code></td>
<td>Convert to lower case</td>
</tr>
<tr>
<td><code>upper()</code></td>
<td>Convert to upper case</td>
</tr>
<tr>
<td><code>ltrim()</code></td>
<td>Remove leading spaces</td>
</tr>
<tr>
<td><code>rtrim()</code></td>
<td>Remove trailing spaces</td>
</tr>
<tr>
<td><code>locate()</code></td>
<td>Find the first location of substring</td>
</tr>
<tr>
<td><code>substr()</code></td>
<td>Extract a substring</td>
</tr>
<tr>
<td><code>left()</code></td>
<td>Return a number of leftmost characters</td>
</tr>
<tr>
<td><code>right()</code></td>
<td>Return a number of rightmost characters</td>
</tr>
<tr>
<td><code>log()</code>, <code>exp()</code></td>
<td>Log and exponent</td>
</tr>
<tr>
<td><code>round()</code></td>
<td>Round off decimals</td>
</tr>
</tbody>
</table>


Formatting conventions

- The semicolon
  ```sql
  SELECT * FROM car;
  SELECT * FROM car
  ```
- The capitalized
  ```sql
  SELECT * FROM car;
  select * FROM car;
  ```
- The space
  ```sql
  SELECT * FROM car;
  ```

AS & LIMIT & making comments

```sql
SELECT Price AS p, Weight AS w FROM car;
SELECT * FROM car LIMIT 3
SELECT price FROM car; -- get prices for all cars
/* Here's my super long comment that doesn’t fit into one single line. But it’s fine. */
```
**Selections: Horizontal Slices**

```sql
SELECT sel_expression
FROM table_references
WHERE condition
```

- Only rows satisfying the `condition` are used
- `Condition` is a Boolean expression, and is evaluated for each row in turn
  - Cannot use values from multiple rows
  - All rows are used if there is no where clause

**Expressions**

```
Expr:
expr OR expr
expr AND expr
NOT expr
```

Comparison operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&lt;=</td>
<td>&lt;=</td>
</tr>
<tr>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>&gt;=</td>
<td>&gt;=</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>!= or &lt;&gt;</td>
<td>!= or &lt;&gt;</td>
</tr>
</tbody>
</table>

Comparison operators:

- LIKE, IN, NOT IN
- IS NULL or IS NOT NULL

**Example**

```
My budget for a plug-in car is under 35,000. Can I afford any of the cars?
```

```sql
SELECT * FROM car WHERE price <= 35000;
```
Simple String Matching - LIKE
expr LIKE pattern

- Pattern: all characters except % and _ stand for themselves
- % is a wild card character for 0 or more characters
- _ is a wild card character that matches exactly one character
- Use \% and \_ if you want to match those characters
- Not case sensitive

Examples of LIKE pattern matching

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>'cam'</td>
<td>'cam' 'Cam' 'CAM' 'cAm'</td>
</tr>
<tr>
<td>'ca%'</td>
<td>'ca' 'cam' 'canada'</td>
</tr>
<tr>
<td>'ca_'</td>
<td>'cam' but not 'ca' or 'canada'</td>
</tr>
<tr>
<td>'_ ca%'</td>
<td>'escape' 'pica' but not 'scape'</td>
</tr>
<tr>
<td>'_ c%a'</td>
<td>'scrap' 'oct is a nice month'</td>
</tr>
</tbody>
</table>

Use single ‘ or double “ to delimit string literals

Comparison operators - IN

- I want to know car sales in California and Washington
  SELECT * FROM sales WHERE st IN ('ca', 'wa');

- I want to know the models of the cars that don’t have battery size 22, 24 or 27
  SELECT models FROM car WHERE battery NOT IN (22, 24, 27);

Ordering your results

- ORDER BY expr [ASC | DESC]
- ordering alphabetically for strings
  SELECT * FROM sales ORDER BY st;
- default is ascending — can add DESC for descending
  SELECT * FROM sales ORDER BY qty DESC;
- order by multiple columns
  SELECT * FROM car ORDER BY battery, weight;
  SELECT * FROM car ORDER BY weight, battery;
- can be used together with LIMIT to see top results
Questions?

Practices...

• Find the state code (two letter st value) for states with names more than 6 letters long

SELECT st FROM state WHERE length(name) > 6;

Practices...

• For each manufacturer, find the battery power to weight ratio (reported as kwh_per_lb) and the MPGe in order of decreasing ratio
For each manufacturer, find the battery power to weight ratio (reported as kwh_per_lb) and the MPGe in order of decreasing ratio.

<table>
<thead>
<tr>
<th>man</th>
<th>kwh_per_lb</th>
<th>mpge</th>
</tr>
</thead>
</table>

• find the most expensive car

```sql
SELECT * FROM car
ORDER BY price DESC
LIMIT 1;
```

What about ties? What if we want to return all cars that are tagged the most expensive?
Aggregate Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count()</td>
<td>Number of rows</td>
</tr>
<tr>
<td>Count(distinct )</td>
<td>Number of different rows</td>
</tr>
<tr>
<td>avg(expr)</td>
<td>Average if numerical or else error</td>
</tr>
<tr>
<td>min(expr)</td>
<td>minimum</td>
</tr>
<tr>
<td>max(expr)</td>
<td>maximum</td>
</tr>
<tr>
<td>std(expr)</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>sum(expr)</td>
<td>sum</td>
</tr>
</tbody>
</table>

Examples of aggregations

- how many car models can I afford with my budget of $35,000?
  ```
  SELECT COUNT(*) AS NumAfford FROM car WHERE price <= 35000;
  ```

- What’s the weight difference between the heaviest car and the lightest car?
  ```
  SELECT MAX(weight) - MIN(weight) FROM car;
  ```

Aggregating within categories

GROUP BY

- GROUP BY expr [, expr]...
- Breaks the rows that satisfy the WHERE clause into as many groups as there are different values for expr, expr
- Separate data into groups which can be aggregated independent of one another
- Can only report aggregate values and the expr in the group by clause

Examples of aggregations with GROUP BY

- How many cars are sold (in total) at each state?
  ```
  SELECT st, SUM(qty) FROM sales GROUP BY st ORDER BY sum(qty) DESC;
  ```
Examples of aggregations with GROUP BY

- Group by multiple columns

```
SELECT st, SUM(qty)
FROM sales
GROUP BY st, county
ORDER BY sum(qty) DESC;
```

HAVING expr

- WHERE doesn’t allow you to filter on aggregate columns
- HAVING — Aggregate functions can be used in the HAVING clause
- Only groups that satisfy the HAVING expression are reported
- Do not use HAVING if a WHERE clause would suffice
  * Much higher overhead

Examples of aggregations with GROUP BY and HAVING

- What states have sales of at least two different models of cars?

```
SELECT st
FROM sales
GROUP BY st
HAVING COUNT(*) >= 2
ORDER BY SUM(qty) DESC;
```

Questions?
Putting it altogether…
the SQL Select Command

```
SELECT [DISTINCT]
    sel_expression [, sel_expression ...]
FROM table_references
[WHERE condition]
[GROUP BY column [, column ...]
    [[HAVING condition]]
[ORDER BY columns [ASC|DESC]];
```

Practices…

• Group the cars into two groups: one that has the letter 'B' in the manufacturer's name, and others that do not. For these two groups find the average prices in dollars.

```
SELECT man LIKE '%b%', ROUND(AVG(price))
FROM car
GROUP BY man LIKE '%b%';
```

Stay tuned for next week

• will talk about SQL joins
• ... and maybe subqueries
Accessing the instructional database

Windows users —

http://www.toadworld.com/freeware/552.aspx
From Dell Computers

Setup Information

Username: publicsql
PW: here4SQL (case sensitive)
Database name: car
SSH host: mysql.simonnet.rochester.edu
Mac users —

Sequel Pro
MySQL database management for Mac OS X

http://www.sequelpro.com/

1: fill the form as shown

here4SQL

here4SQL

The authenticity of host 'mysql.simonnet.rochester.edu (128.161.238.163)' can't be established. RSA key fingerprint is ab:ac:49:9d:ad:36:67:0a:af:bf:bd:3f:df:9b:97:ba:8e. Are you sure you want to continue connecting (yes/no)?

No  Yes
How am I doing?

What works? What doesn’t?

U S Census Bureau

http://www.census.gov/acs/www/

The American Community Survey (ACS) is an ongoing statistical survey that samples a small percentage of the population every year -- giving communities the information they need to plan.

Over 400 Databases from Single Portal

http://alpha.data.gov/

A collection of open data from government, companies, and non-profits including:

• Commerce - GPS, Weather, Census Data
• Energy - Outlook and Analysis, Energy Usage, Energy Star, Your Car’s Data Stream, Renewable Energy + See all energy datasets and tools
• Health - Recalls and Safety Alerts, Medicare Plans, Hospital Quality Data, Public Health Data, Clinical Trials, Health Indicators, Your Health Record + See all health datasets & tools
• Education - Open Badges, Academic Data, Learning Registry, Education Data Standards, Higher Education Datasets + See all education datasets & tools

Factfinder2 Database

http://factfinder2.census.gov/faces/nav/jsf/pages/index.xhtml

Find popular facts about your community by using Community Facts. Enter a state, county, city, town, or zip code in the Community Facts text box, then click GO. The Community Facts page displays,
Health Data for Epidemiology

http://chir.asu.edu/data

A key activity of CHiR from its outset has been obtaining health data being collected routinely by various health care and public health entities and making these data available for use for

Data for Urban Planning

http://www.cityforward.org/

City Forward data catalog is a listing of the data sets currently available for use from publicly available data sources about cities and metropolitan areas around the world. Please use the links back to the data source and their terms of use.

Database of Technology Companies

http://www.crunchbase.com/

CrunchBase is the free database of technology companies, people, and investors that anyone can edit.

Earth Policies

http://www.earthpolicy.org/books/fpep/fpep_data

• Population
• Food production
• Weather
European Central Bank


- Monetary data
- Balance of payments
- Economic data

EU Statistical Bureau

http://epp.eurostat.ec.europa.eu/portal/page/portal/eurostat/home/

- Eurostat is the statistical office of the European Union situated in Luxembourg. Its task is to provide the European Union with statistics at European level that enable comparisons between countries and regions.
- One example: for an accurate picture of EU unemployment it is important that unemployed people in Finland or Portugal are counted or measured in the same way as in Ireland or Germany. So Eurostat works with Member States to define common methodology on unemployment or asks Member States to include appropriate questions when gathering national data. These EU data are then sent to Eurostat so we can publish EU-wide unemployment data, which can then be used to compare unemployment rates between countries.

Google DB Search Engine

www.google.com/publicdata/directory

- Useful search engine
- You can save to “my data”
- Google maintains over 100 million curated pages of data

Amazon Public Data Sets

http://aws.amazon.com/datasets

- AWS Amazon Web Services
- Hadoop friendly files
- Can use seamlessly with Amazon cloud computing utility EC2 Amazon ‘Elastic Cloud Compute’