ACCESS SQL WORKSHOP I: INTRODUCTION TO SQL

What is a relational database?
What is a database?

- A **database** is an organized collection of **data**, stored and accessed electronically.
- This **data** requires interpretation to become information we can analyze.
What is a relational database?

- A relational database is a digital database based on the relational model of data.
- Virtually all relational database systems use SQL (Structured Query Language) for querying and maintaining the database.
  - SQL is pronounced “S-Q-L” or “sequel”.
What is the **relational model**?

- Data is organized in **tables** of **columns** ("attributes") and **rows** ("records"), with a unique **key** ("primary key") identifying each row.

- Rows in a table can be **linked** to rows in other tables by adding a columns for the unique key of the linked row ("foreign keys").
I’m not a computer scientist. What does this mean for an actuary?

- This abstract data stuff has more applications to actuarial work than you may think!
- Consider the most basic example: policies and claims.
### Example: Policies and Claims

**Primary Key: PolicyNumber**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>PolicyNumber</td>
<td>PolicyStartDate</td>
<td>AnnualMilesDriven</td>
<td>County</td>
</tr>
<tr>
<td>2</td>
<td>P100000</td>
<td>1/1/2014</td>
<td>5,000</td>
</tr>
<tr>
<td>3</td>
<td>P100001</td>
<td>1/1/2014</td>
<td>31,000</td>
</tr>
<tr>
<td>4</td>
<td>P100002</td>
<td>1/1/2014</td>
<td>14,000</td>
</tr>
<tr>
<td>5</td>
<td>P100003</td>
<td>1/1/2014</td>
<td>26,000</td>
</tr>
<tr>
<td>6</td>
<td>P100004</td>
<td>1/1/2014</td>
<td>9,000</td>
</tr>
<tr>
<td>7</td>
<td>P100005</td>
<td>1/1/2014</td>
<td>6,000</td>
</tr>
<tr>
<td>8</td>
<td>P100006</td>
<td>1/1/2014</td>
<td>13,000</td>
</tr>
<tr>
<td>9</td>
<td>P100007</td>
<td>1/1/2014</td>
<td>8,000</td>
</tr>
<tr>
<td>10</td>
<td>P100008</td>
<td>1/1/2014</td>
<td>12,000</td>
</tr>
<tr>
<td>11</td>
<td>P100009</td>
<td>1/1/2014</td>
<td>14,000</td>
</tr>
</tbody>
</table>

**Primary Key: ClaimNumber**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClaimNumber</td>
<td>PolicyNumber</td>
<td>ClaimAmount</td>
<td>LossDate</td>
<td>ReportDate</td>
</tr>
<tr>
<td>C900302</td>
<td>P100001</td>
<td>6,615</td>
<td>3/2/2014</td>
<td>1/18/2015</td>
</tr>
<tr>
<td>C902408</td>
<td>P100009</td>
<td>1,591</td>
<td>11/2/2014</td>
<td>7/1/2016</td>
</tr>
</tbody>
</table>

**Foreign Key: PolicyNumber**
Example: Policies and Claims

- We can see that the table for Claims can be linked to the table for Policies through the key PolicyNumber.
- That’s a brief introduction to what we’re doing and why you should care.
- In fact, this workshop will focus entirely on this seemingly-simple example of only two* tables.

*This isn't quite true. We'll create a few more later to play with, though that’s an issue for a later time.
Microsoft Access / SQL

- Now that you know what we’re working with, let’s jump in!
The Microsoft Access Environment
The Microsoft Access Environment

- Data can be entered into data tables manually, and data tables can be viewed in their entirety manually.
What we care about: Queries
This user-friendly interface is good enough for some simple tasks...
…but for more advanced tasks, we’ll want to use SQL.
Regardless of what view you choose, click Run to run the query.
- This may not make any sense now, but that’s what this workshop is for.
- When we begin writing simple queries, you can always go back to design view to see the user-friendly depiction of what’s going on.
- That’s enough introductory stuff. Let’s begin coding!
Getting our first tables: Importing from CSVs
SELECT `column1`, `column2`, ... FROM `table_name`;
SELECT * FROM `table_name`;

- The first line selects all rows, but only the specified columns, from `table_name`.
- The second line selects all rows and all columns from `table_name`.
SELECT column1, column2, ... FROM table_name
WHERE condition;

- Only selects rows from `table_name` where `condition` is true (and the appropriate columns)
- `condition` can be a simple boolean:
  - WHERE `column_name` = “Value”
  - WHERE `column_name` <> 0
- Or an IN statement:
  - WHERE `column_name` IN (value1, value2, value3)
- Or a complex statement:
  - WHERE (column_name = 0 OR column_name = 1) AND another_column IN (“value1”, “value2”)
SELECT DISTINCT column1, column2, ...
FROM table_name;

- Selects appropriate rows and columns, but removes duplicates
```
SELECT column1, column2, ... FROM table_name
ORDER BY column1, column2, ... [DESC]
```

- Selects appropriate rows and columns, then orders them in the given order.
- By default, sorts in ascending order.
- Use `DESC` to sort in descending order.
Exercises

- Select all policies with low driving frequency in LA or Orange county. How many are there? Note that each policy gets a new entry upon renewal—be sure to account for this.

- Select all policies with that have low or medium driving frequency or are not located in LA, Orange, or San Bernardino county.
SQL Variables

- Any “unrecognized” text (that isn’t a reserved SQL keyword) is interpreted as a variable.
- Suppose our columns are “col1”, “col2”, and “col3”:
  - SELECT col1, col2 FROM table WHERE col1 <> 0 AND col3 = my_variable;
- Since my_variable isn’t a column name or a keyword, SQL interprets it as a variable.
- You will be prompted to input a value for my_variable before the script runs.
Aggregate Functions: SUM, COUNT, AVG, ...

- Aggregates numerical data in groups
- Must be accompanied by a `GROUP BY` clause

```sql
SELECT SUM(Claim_Amount), ReportYear FROM claims GROUP BY ReportYear;
```

- This will give the total claims for each year, since the aggregate data is grouped by report year.
Exercises

- What is the total amount of claims reported in each year?
- What is the total amount of claims occurring in each year?
- Consider only claims that occurred in 2014. How much was reported in each year?
WHERE vs. HAVING

**WHERE**
- Used to test variables in a single “cell” of the data table
- Cannot be used with aggregate functions

**HAVING**
- The “equivalent” of WHERE for aggregate functions
- Comes after the `GROUP BY` clause
  - We need to know what to group by before we know which groups to show!
Exercises

- Use the previous example to determine how many claims each policy would have if we implemented a $1,000 policy deductible.
  - Hint: A $1,000 eliminates all policies with total claims under $1,000. For policies with claims under $1,000, their total claim amount will be reduced by $1,000.

- How many policies had total claims over $60,000? What are their policy numbers?

- Use the results from the previous part to determine how many claims each of these policies had. Which policy has the highest severity? What is it?
To be continued...

- A preview of next time:
Next time, we’ll see how powerful SQL is by learning how to link related data.

<table>
<thead>
<tr>
<th>PolicyNumber</th>
<th>PolicyStartDate</th>
<th>AnnualMilesDriven</th>
<th>County</th>
</tr>
</thead>
<tbody>
<tr>
<td>P100000</td>
<td>1/1/2014</td>
<td>5,000</td>
<td>Orange</td>
</tr>
<tr>
<td>P100001</td>
<td>1/1/2014</td>
<td>31,000</td>
<td>Ventura</td>
</tr>
<tr>
<td>P100002</td>
<td>1/1/2014</td>
<td>14,000</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>P100003</td>
<td>1/1/2014</td>
<td>26,000</td>
<td>San Bernadino</td>
</tr>
<tr>
<td>P100004</td>
<td>1/1/2014</td>
<td>9,000</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>P100005</td>
<td>1/1/2014</td>
<td>6,000</td>
<td>San Bernadino</td>
</tr>
<tr>
<td>P100006</td>
<td>1/1/2014</td>
<td>13,000</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>P100007</td>
<td>1/1/2014</td>
<td>8,000</td>
<td>Orange</td>
</tr>
<tr>
<td>P100008</td>
<td>1/1/2014</td>
<td>12,000</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>P100009</td>
<td>1/1/2014</td>
<td>14,000</td>
<td>Orange</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ClaimNumber</th>
<th>PolicyNumber</th>
<th>ClaimAmount</th>
<th>LossDate</th>
<th>ReportDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>C900302</td>
<td>P100001</td>
<td>6,615</td>
<td>3/2/2014</td>
<td>1/18/2015</td>
</tr>
<tr>
<td>C902408</td>
<td>P100009</td>
<td>1,591</td>
<td>11/2/2014</td>
<td>7/1/2016</td>
</tr>
</tbody>
</table>