STATS 406 F15: Lab 09
Structured Query Language (SQL) basics

1 Introduction to relational databases

- The advantage of using relational databases:
  * A very short article: `http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_9/database_design/miniweb/pg8.htm`
  * Data can be maintained and queried by different threads independently.
  * More efficient queries for objectives that only concern one subtable.
- Relational database files and their management systems:
  * A .db file is to relational database management systems (SQLite, MySQL, Oracle, etc) as a .pdf file is to PDF readers (Adobe, Foxit, Okular, etc).
  * There is one file format (.db) and many tools (management systems) you can use to manage the file.
  * Different management systems have very similar syntaxes on basic operations and return queried data in very similar forms. They differ in efficiency and other aspects, but not much grammatically.
  * A list of popular management systems: `https://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems`
  * In fact, the commands and even most code lines you learned in this class will run without any modification under most management systems.

2 Logistics: using SQLite in R

- This lab is about SQL, not R.
- Recap: install package RSQLite and connect to a database.

```
## Load the RSQLite package (if necessary, also install it first ):
if(!(require(RSQLite))){
        install.packages("RSQLite", dep=TRUE);
}
## Connect to the file
## Make sure the file is under R's working directory
driver = dbDriver("SQLite");
conn = dbConnect(driver, "baseball.db");
```

3 The order to read an SQL code

From: `http://blog.jooq.org/2014/12/04/do-you-really-understand-sqls-group-by-and-ha` and slightly modified.

1. **FROM** generates the data set.
   (a) **INNER JOIN** combines tables.
   (b) **ON** filters the combined tables.
2. **WHERE** filters the generated data set
3. **GROUP BY** aggregates the filtered data set
4. **HAVING** filters the aggregated data set
5. **SELECT** transforms the filters aggregated data set
6. **ORDER BY** sorts the transformed data set

4 SQL commands

4.1 A few notes before we start:

- All SQL keywords are case insensitive, but when calling SQL in R, capitalizing key words helps to improve readableness (especially observing a lack syntax coloring).
- SQL is designed mainly for extracting data from databases, NOT for analyzing them. SQL provides basic summarizing commands and tools, but do not expect too much. You can use R for further analysis.

4.2 Basic SQL commands:

- The very basic form of SQL queries is:

```
/* Pseudo-code */
/* Required clauses */
SELECT VariableName AS NewVariableName, Sum(VariableName) AS SumVariableName
FROM TableName
/* Optional clauses */
WHERE Conditions
GROUP BY VariableNames HAVING Conditions
ORDER BY VariableNames
```

- A more specific example:

```
/* Pseudo-code: SQLite */
/* Example from: https://en.wikipedia.org/wiki/Having_(SQL) */
/* (modified) */
SELECT DeptID AS dept, Sum(SaleAmount) TotalSalesAmount
FROM Sales
WHERE SaleDate = '01-Jan-2000'
GROUP BY DeptID HAVING Sum(SaleAmount) > 1000
ORDER BY DeptID
```

  In **FROM**:

  * In SQLite and in this course, you can understand **FROM Table1, Table2** the same as **FROM Table1 INNER JOIN Table2**.

  In **WHERE**:

  * If there are multiple conditions, they should be connected by logical connectives (**AND**, **OR** and parenthesis when needed). Conditions can also be decorated by other logical operators (**NOT**, **ANY**, etc). For a list of logical operators in SQL, see `http://www.w3resource.com/sql/boolean-operator/sql-boolean-operators.php`.

    ```
    /* Pseudo-code: SQLite */
    /* Example from: http://beginner-sql-tutorial.com/sql-logical-operators.htm */
    SELECT first_name, last_name, age, games
    FROM student_details
    WHERE age >= 10 AND age <= 15 OR NOT games = 'Football'
    ```

  For **GROUP BY**:

  * Rows that belong to the same group will only produce one row in the final output.
  * **GROUP BY** is used in combination with aggregate functions in **SELECT**.
  * (From Wikipedia) **HAVING** modifies **GROUP BY**. It is indispensable because **WHERE** does not allow aggregate functions.

In **SELECT**:

* **AS:** It can be used anywhere, not only in **SELECT**. Itself can always be omitted.
* As a consequence, the variable name should NOT contain space (Why?). If it is the case in the data, you can escape using 'var name' or [var name]. But the correct syntax of this fix needs to be double checked under different platforms.
* **AS** renames the extracted variables for convenience. It is especially useful when a. they are summarized; or b. (will see later) when they have to come with prefixes like table names.
* **Aggregate functions.** Usually used with **GROUP BY**. For a list, check, for example, `http://www.techonthenet.com/sql_server/functions/index_alpha.php`.
* NOTICE that the wording of specific functions may vary under different management systems. For example, SQLite uses "length()" instead of "len()"*.

```
/* Pseudo-code: SQLite */
SELECT Var1, Sum(Var2) AS SumVar2, Count(Var3) LengthVar3
FROM TableName
```

Quiz: Query data from Table *Teams*. Generate a table with the number of teams that won more than half of the games each year. Sort by year.

```
# Solution: see Lab_9.r
```

4.3 Inner joining tables

- Basic form (not quite "basic", look closely):

```
/* Pseudo-code */
SELECT T1.ColumnNames, T2.ColumnNames
FROM TableName1 T1 INNER JOIN TableName2 T2 ON JoiningConditions
/* Other clauses */
WHERE Conditions
/* etc */
```

- **How does inner join work?**
- What is "TableName1 T1" doing?
- **ON**: Can **ON** be completely replaced by **WHERE**? Within the range of this course, yes, but for aesthetic reasons please don't do so. For more discussion on **ON** vs **WHERE**, see, for example, `http://stackoverflow.com/questions/1018822/inner-join-on-vs-where-clause`

*But notice that the function that corresponds to the "length()" function in R is "Count()".

- **(Optional) INNER JOIN** is just the (arguably) the simplest type of joining tables. With other join types, using **ON** or **WHERE** can produce essentially different results. See, for example, `http://blog.sqlauthority.com/2014/10/13/sql-server-what-is-the-difference-between-a` Start reading from the middle of the page.

- In **SELECT** here: in examples you saw in lecture, the variable/column names that **SELECT** picked did not come with prefixes.

- **Example:** Read and analyze the following SQL code:

```
/* Goal: compare the differences in players' total salaries between teams for each year since
    1996. */
SELECT T1.yearID year, T1.teamID Team1ID, T2.teamID Team2ID, T1.SumSalary-T2.
    SumSalary SalaryDifference
FROM    (SELECT yearID, teamID, Sum(salary) SumSalary
        FROM Salaries
        GROUP BY yearID, teamID
        ORDER BY yearID, teamID
        ) T1
    INNER JOIN
        (SELECT yearID, teamID, Sum(salary) SumSalary
        FROM Salaries
        GROUP BY yearID, teamID
        ORDER BY yearID, teamID
        ) T2
    ON
        T1.yearID=T2.yearID AND T1.teamID<T2.teamID
WHERE T1.yearID>1996
-- Do we need a "GROUP BY" clause here? Why?
ORDER BY T1.yearID, T1.teamID, T2.teamID
```

- NOTICE the trick played with **AS** in **FROM**.

- What will happen if we remove the prefix "T1"/"T2" from variables in **SELECT**, **GROUP BY** or **ORDER BY**?

- What is the effect of feeding **GROUP BY** (inside **FROM**) with two variables?

- What is the effect of feeding **ORDER BY** with two variables?

5 Additional resources

There are many good resources for beginners on SQL on Internet. I just list a few examples:

- Stanford online course: `https://lagunita.stanford.edu/courses/DB/SQL/SelfPaced/courseware/ch-sql/`

- W3School – **a good dictionary for beginners**. `http://www.w3schools.com/sql/` You can "Try it yourself" but don't expect too high since the example data are sometimes too large to be illustrative.
- (Optional) Specifically, if you want to learn more types of join in the future, this is a classical illustration `http://stackoverflow.com/questions/6294778/mysql-quick-breakdown-of-the-types-of-jo`