STATS 406 Fall 2016: Lab 04

1 Introduction to relational databases

- The advantage of using relational databases:
  * Data can be maintained and queried by different threads independently.
  * More efficient queries for objectives that only concerns 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.
  * In fact, the commands and even most code lines you learned in this class will run without modification under most management systems.

2 Using SQLite in R

- We can work with databases within R by installing related packages.
- Install package RSQLite and connect to a database.

```
## Load the RSQLite package (install it first if haven't yet)
install.packages("RSQLite", dep=TRUE);
library("RSQLite");
## Make sure the file is under the working directory and then connect to it
driver = dbDriver("SQLite");
conn = dbConnect(driver, "baseball.db");
## Check the tables in a database
dbListTables(conn);
## Check the variables in a table
dbListFields(conn, "Teams");
```

3 SQL commands

3.1 A few notes

- SQL keywords are not case sensitive, but when calling SQL in R, capitalizing key words helps to improve readability.
- 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. We can use R for further analysis.

3.2 Basic SQL commands:

- The very basic form of SQL queries is:

```
/* Pseudo-code */
/* Required clauses */
SELECT ColumnNames
FROM TableName
/* Optional clauses */
WHERE Conditions
GROUP BY VariableNames HAVING Conditions
ORDER BY VariableNames
```

  In **SELECT**:

  * **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.
  * **AS** can often be omitted. Thus the variable name should NOT contain space (Why?).
  * Aggregate functions. For a list, check, for example, `http://www.techonthenet.com/sql_server/functions/index_alpha.php`.

```
/* Pseudo-code: SQLite */
SELECT Var1, MIN(Var2) AS MinVar2, AVG(Var3) AS AvgVar3
FROM TableName
```

  In **FROM**:

  * In this course, unless we combine tables, otherwise we only select from one table.

  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**, 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'
```

Quiz 1: Extract data from table *Master*. Generate a table with all variables in Master for players born in 1980s and sort by their names.

```
# Solution: see Lab4.R
```

- For **GROUP BY**:
  * **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.

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

Quiz 2: Query data from table *Teams*. Generate a table showing the number of teams that won more than half of the games each year since 1950. Sort by year.

```
# Solution: see Lab4.R
```

3.3 Inner joining tables

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

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

- What are the different roles of Table "T1" and Table "T2"?
- **ON**: Specify which rows to combine from the two tables
- In **SELECT** here: specify table names as prefixes for variable names when needed.

- **Example:**

```
SELECT Schools.schoolID, schoolName, schoolCity, schoolState, playerID
FROM Schools INNER JOIN SchoolsPlayers ON Schools.schoolID = SchoolsPlayers.schoolID
WHERE schoolState = 'MI'
```

  * If we omit "Schools" in the first "Schools.schoolID", there will be an error.
  * In general, DO NOT omit table names if causing ambiguity.

Quiz 3: Join table *Schools* with table *SchoolsPlayers*. Generate a table showing the number of players from each school in Michigan. Sort by the number of players in decreasing order.

```
# Solution: see Lab4.R
```