Working with relational databases

Why use a database?

- Database systems (DBS) are a very popular way of storing data. They are particularly efficient in dealing with large data sets.
- Think of a database (DB) as a dataset managed by a software (known as the database server). In order to access the data, you have to "talk" to the database server. The same way that web servers managed internet resources.
- As such, DB system facilitates the sharing of the same data between multiple entities. This reduces the number of copies of the data that is kept.

Why use a database?

- DBS make it easy to control who can access the data.
- DBS Allow concurrent access to data - synchronize access. Can handle transactions. Enforce the security and integrity of the data.
- Makes it easy to maintain the data.
- DB systems can accommodate very large data sets. They are highly optimized for fast retrival of the data.

What do we need to know?

Goals:

- Develop familiarity with relational database systems, and be able to map simple database terminology into statistics terms.
- Use the SELECT command of the Structured Query Language (SQL) to access data.

Relational databases

- Currently, the dominant model for database management is called "relational databases" and we will see why shortly.
- Major Relational database systems includes **Oracle**, **SQL Server**, **PostgreSQL**, **MYSQL**, **MS Access**, **SQLite**.
- **Oracle** (Oracle), **SQL Server** (MS), and **MS Access** (MS) are commercial systems.
- **SQLite**, **MYSQL**, and **PostgreSQL** are open-source. **SQLite** is a light-weight system. **PostgreSQL** is comparable to Oracle.
- There is a language for manipulating relational databases known as **SQL** (Structured Query Language). We will only learn the fraction of SQL that deals with accessing the data (the **SELECT** command).

Relational databases

- In a relational DB system, the data is represented in two dimensional **tables** (also called relations) which consist of rows (records, observations) and columns (attributes or variables). A database is a collection of such tables.
- Think of a table as a dataframe that is holding variables (or column or attributes). Each row of the table is an observation (or a case, or tuple).

Relational databases

- In DB systems, effort in put in representing the data in a way that avoids redundancy as much as possible.
- This is done by breaking the original data accross multiple tables. A process known as **normalization**.
- This is best understood through an example.

Relational databases

- Suppose that you are managing the data for a company with a number of customers, each of which will be placing multiple orders, each order has multiple items.
- If we try to keep one table to record the data, it will look like this.

CNo	Name	Address	ONo	OrderDate	ItemNo	ItemName	ItemQty	Price
1	Smith, F	101 Elm	201	”1-12-99”	001	Item1	100	5
1	Smith, F	101 Elm	201	”1-12-99”	005	Item5	250	10
2	Brown, D	17 Spruce	301	”08-27-01”	010	Item10	20	350
2	Brown, D	17 Spruce	302	”02-21-04”	031	Item31	400	12

Relational databases

- Each time an order is placed you will need to repeat the customer information, including the Customer's name, address etc...
- Worst: for each item, you need to repeat the Order information and the customer information.
- For example if a customer place two orders each with 4 items, you need 8 lines to record these two transactions.
- What if the customer send you a change of address notice?
- Difficulties of this one table model: error prone, difficult to maintain.

Relational databases

- Instead we split this data into three separate tables.
- One table for customers. If an existing customer places a new order this table does not need to be changed.
- One table for orders. Even if a new order has multiple items, that table gets only one new record.
- And one table for items.

Relational databases

CNo	Name	Address
1	Smith, F	101 Elm
2	Brown, D	17 Spruce

OrderNo	CNo	OrderDate	DelivAddr
201	1	"1-12-1999"	Address
301	2	"08-27-2001"	Address
302	2	"02-21-2004"	Address

OrderDetailNo	OrderNo	ItemName	ItemQty	ItemPrice
001	201	Item1	100	5
005	201	Item5	250	10
010	301	Item10	20	350
031	302	Item31	400	12

Relational databases

- The columns "CNo", "OrderNo" and "OrderDetailNo" are called **Primary keys**. Each table in a relational database needs one (and only one) Primary key. That is a column or a group of columns that uniquely identify each record of the table.
- Notice the presence of the column "CNo" (the Primary key of the table Customer) in the table "Order". In table "Order", the column "CNo" is called a **Foreign Key**.
- In relational databases, by creating foreign keys where needed, one create links between tables.

Relational databases

- Given a database, as a statistician, we need some basic ability to query the database and retrieve the data.
- By installing the appropriate additional library/package in R we can work with with most databases from within R.
- To do so, we need a brief introduction to working with packages in R.

R packages

- Packages in R are pieces of software that can be loaded into a R session, to acquire additional computing tools.
- They are written by members of the community and freely available online (`www.r-project.org`).
- However, we can install packages from within R, without using a browser.

R packages

- You can see the list of packages attached to your current R session with the function .packages(), as follows.

  ```
  z = .packages()
  z
  [1] "space"     "stats"     "graphics"  "grDevices" "ut
  [6] "datasets"  "methods"   "base"
  >
  ```

- You can have a description of a package with

  ```
  packageDescription("stats")
  ```

R packages

- Before we can use a package, we need to install and load it.
- We typically install a package using the function `install.packages`.
- For instance, we will need below to use the package RSQLite that will allow us to connect to SQLite databases. We can do this as follows.

  ```
  install.packages("RSQLite", dep=T)
  ```
- The argument `dep` specifies whether we would like to install at the same time all other packages on which RSQLite is built (in this case, the package RSQLite needs another important package called DBI).

R packages

- When a package is intalled, we can load it into the current R session by calling the function `library` as follows.

  ```
  library(RSQLite)# or library("RSQLite")
  ```

- You can also use the function `require`.
- When done with a Package, you can remove it from the search directory with the function `detach`, as in

  ```
  detach(package:RSQLite)
  ```