Interfacing SQL with R/Python

Interfacing SQL in Python :

Python's *pandas* library can do the majority of the work done in SQL. This is similar to *dplyr* library of R.

The *sqlite3* package provides a straightforward interface to extract data from sqlite databases using standard SQL commands. We need to import is with command:

```
import sqlite3
```

Now let's understand the interfacing with an example:

1. Step-1 Basically we import the libraries.
   code:
   ```
   import sqlite3
   Import pandas as pd
   ```

2. Step-2: Then, we create a sqlite3 database in RAM. name it 'con'.
   code:
   ```
   con = sqlite3.connect(":memory:")   # create a sqlite3 database in RAM
   ```

3. Step-3: Now we transfer columns of flights, weather, airlines,airports, planes to our database 'con'.
   code:
   ```
   flights.to_sql('flights', con=con)
   weather.to_sql('weather', con=con)
   airlines.to_sql('airlines', con=con)
   airports.to_sql('airports', con=con)
   planes.to_sql('planes', con=con)
   ```

4. Step-4: Now we are ready to write a SQL query. Here for the purpose of an example, we create a query to select top-10 flights
   Code:
   ```
   query = "SELECT * FROM flights LIMIT 10"  #
   ```

5. Step-5: Interfacing SQL with python and printing the results.
   code:
   ```
   top10 = pd.read_sql(query, con)  #this command does interfacing
   top10                            #this prints the results of previous step
   ```

Interfacing SQL & R

There are two ways to interface the SQL and R. One is through database interface (DBI) and the other one is through the *dplyr* library. Below, we provide the usual steps:

Step-1: Load the driver, connect to the right database
Step-2: R sends an SQL query to the DBMS
Step-3:SQL executes the query, sending back a manageably small dataframe
Step-4: R does the actual statistics and data analysis
Step-5: Close the connection when you're done

If the dplyr library of R is so good then why do we need SQL?

The goal of dplyr is not to replace every SQL function with an R function; that would be difficult and error prone. Instead, dplyr only generates SELECT statements, the SQL you write most often as an analyst. Also, SQL is a simple language which gives us speed. Sometimes in big-data, the speed is crucial therefore the simpler language like SQL may end-up doing a good job.