SQL & R - Extra Credit

Jawaid Hakim

2022-09-10

Introduction

Here we do a basic exploration of SQL Window Function.

From PostGreSQL introduction to Windows Function:

```
A window function performs a calculation across a set of table rows that are somehow related to the curr
```

Absenteeism Dataset

The [Absenteeism] (https://archive.ics.uci.edu/ml/datasets/Absenteeism+at+work) database was created with records of absenteeism at work from July 2007 to July 2010 at a courier company in Brazil. For this assigment dataset was loaded into a AWS RDS MySQL database. Let's connect to the database:

```
db <- dbConnect(RMySQL::MySQL(), user="guest", password="guestpass", host="cuny-ds.c5iiratvieki.us-east-
dbListTables(db)
```

```
## [1] "Absenteeism_at_work"
```

Let's query the Absenteeism table.

```
qry <- "SELECT * FROM Absenteeism_at_work ORDER BY ID, 'Month of absence', 'Absenteeism time in hours'"
rs <- dbSendQuery(db, qry)      # Send query for execution
rows <- dbFetch(rs, n=-1)       # Fetch query results
dbClearResult(rs)               # Clear results cache
```

```
## [1] TRUE
```

```
str(rows)
```

```
## 'data.frame':    740 obs. of  21 variables:
##  $ ID                             : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Reason for absence             : num  22 23 26 7 13 23 19 26 18 25 ...
##  $ Month of absence               : num  7 8 12 4 6 8 8 10 11 11 ...
##  $ Day of the week                : num  2 5 4 6 6 3 5 2 4 5 ...
##  $ Seasons                        : num  1 1 4 3 3 1 1 4 4 4 ...
##  $ Transportation expense         : num  235 235 235 235 235 235 235 235 235 235 ...
##  $ Distance from Residence to Work: num  11 11 11 11 11 11 11 11 11 11 ...
```

```
##  $ Service time                   : num  14 14 14 14 14 14 14 14 14 14 ...
##  $ Age                            : num  37 37 37 37 37 37 37 37 37 37 ...
##  $ Work load Average day          : num  240 206 261 326 378 ...
##  $ Hit target                     : num  97 92 97 96 94 94 94 88 97 97 ...
##  $ Disciplinary failure           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Education                      : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ Son                            : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Social drinker                 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Social smoker                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Pet                            : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Weight                         : num  88 88 88 88 88 88 88 88 88 88 ...
##  $ Height                         : num  172 172 172 172 172 172 172 172 172 172 ...
##  $ Body mass index                : num  29 29 29 29 29 29 29 29 29 29 ...
##  $ Absenteeism time in hours      : num  8 4 8 3 16 1 8 4 1 2 ...
```

Having selected this data set, and well into doing the assignment, I realized there was no *year* column in the dataset. This was problematic because without the year it is impossible to accurately sequence the observations. For example, it is impossible to identify the year in which observations **ID = 2** were made. The first observation could be from year 2007 or from 2008 or from 2009!

Furthermore, not every month is represented in the observations. For example, observations for ID = 2 are only available for the months of April (4), June (6), July(7), and August(8).

To get around these issues for this assignment, we make some simplifying assumptions:

1. Observations for all IDs start on the same year (2007) and month (1)
2. Observations for a given ID are for consecutive months (1/2007, 2/2007, . . . , 12, 1/2008, 2/2008, . . .)

```
qry <- "SELECT * FROM Absenteeism_at_work WHERE ID = 2 ORDER BY ID, 'Month of absence', 'Absenteeism ti
rs <- dbSendQuery(db, qry)
rows <- dbFetch(rs, n=-1)
dbClearResult(rs)
```

```
## [1] TRUE
```

```
str(rows)
```

```
## 'data.frame':    6 obs. of  21 variables:
##  $ ID                             : num  2 2 2 2 2 2
##  $ Reason for absence             : num  18 18 28 0 23 0
##  $ Month of absence               : num  8 8 8 4 7 6
##  $ Day of the week                : num  5 2 6 2 2 2
##  $ Seasons                        : num  1 1 1 3 1 3
##  $ Transportation expense         : num  235 235 235 235 235 235
##  $ Distance from Residence to Work: num  29 29 29 29 29 29
##  $ Service time                   : num  12 12 12 12 12 12
##  $ Age                            : num  48 48 48 48 48 48
##  $ Work load Average day          : num  206 206 206 326 230 ...
##  $ Hit target                     : num  92 92 92 96 92 96
##  $ Disciplinary failure           : int  0 0 0 1 0 1
##  $ Education                      : num  1 1 1 1 1 1
##  $ Son                            : num  1 1 1 1 1 1
##  $ Social drinker                 : int  0 0 0 0 0 0
```

```
##  $ Social smoker                 : int  1 1 1 1 1 1
##  $ Pet                           : num  5 5 5 5 5 5
##  $ Weight                        : num  88 88 88 88 88 88
##  $ Height                        : num  163 163 163 163 163 163
##  $ Body mass index               : num  33 33 33 33 33 33
##  $ Absenteeism time in hours     : num  8 8 8 0 1 0
```

Since the *Month of Absence* column is irrelevant under these assumptions, we can drop this column:

```
rows <- rows %>% mutate(`Month of absence` = NULL)
str(rows)
```

```
## 'data.frame':    6 obs. of  20 variables:
##  $ ID                            : num  2 2 2 2 2 2
##  $ Reason for absence            : num  18 18 28 0 23 0
##  $ Day of the week               : num  5 2 6 2 2 2
##  $ Seasons                       : num  1 1 1 3 1 3
##  $ Transportation expense        : num  235 235 235 235 235 235
##  $ Distance from Residence to Work: num  29 29 29 29 29 29
##  $ Service time                  : num  12 12 12 12 12 12
##  $ Age                           : num  48 48 48 48 48 48
##  $ Work load Average day         : num  206 206 206 326 230 ...
##  $ Hit target                    : num  92 92 92 96 92 96
##  $ Disciplinary failure          : int  0 0 0 1 0 1
##  $ Education                     : num  1 1 1 1 1 1
##  $ Son                           : num  1 1 1 1 1 1
##  $ Social drinker                : int  0 0 0 0 0 0
##  $ Social smoker                 : int  1 1 1 1 1 1
##  $ Pet                           : num  5 5 5 5 5 5
##  $ Weight                        : num  88 88 88 88 88 88
##  $ Height                        : num  163 163 163 163 163 163
##  $ Body mass index               : num  33 33 33 33 33 33
##  $ Absenteeism time in hours     : num  8 8 8 0 1 0
```

With that preamble out of the way, the 2 columns of interest for this assignment are **ID** (identifying the subject), and target column **Absebteeism time in hours**.

We start with computing the running average of *Absebteeism time in hours* for each subject using plain old GROUP BY function:

```
qry <- "select DISTINCT ID,
            avg(`Absenteeism time in hours`) AS 'Average'
            FROM Absenteeism_at_work
            GROUP BY ID
            ORDER BY ID"
rs <- dbSendQuery(db, qry)
rows <- dbFetch(rs, n=-1)
dbClearResult(rs)
```

```
## [1] TRUE
```

```
head(rows)
```

```
##   ID Average
## 1  1  5.2609
## 2  2  4.1667
## 3  3  4.2655
## 4  4  0.0000
## 5  5  5.4737
## 6  6  9.0000
```

Now we do the same computation using Windows Function:

```
qry <- "select DISTINCT ID,
            avg(`Absenteeism time in hours`) OVER(PARTITION BY ID) AS 'Average'
            FROM Absenteeism_at_work
            ORDER BY ID"
rs <- dbSendQuery(db, qry)
rows <- dbFetch(rs, n=-1)
dbClearResult(rs)
```

```
## [1] TRUE
```

```
head(rows)
```

```
##   ID Average
## 1  1  5.2609
## 2  2  4.1667
## 3  3  4.2655
## 4  4  0.0000
## 5  5  5.4737
## 6  6  9.0000
```

Finally, we compute the 6-month and 3-month running average using Window Function. As expected, the aggregation is applied to to each row.

```
qry <- "select DISTINCT ID,
                ROUND(avg(`Absenteeism time in hours`) OVER(PARTITION BY ID ROWS BETWEEN 5 PRECEDING AN
                ROUND(avg(`Absenteeism time in hours`) OVER(PARTITION BY ID ROWS BETWEEN 2 PRECEDING AN
                FROM Absenteeism_at_work
                ORDER BY ID, 6_Month_Average, 3_Month_Average;"
rs <- dbSendQuery(db, qry)
rows <- dbFetch(rs, n=-1)
head(rows)
```

```
##   ID 6_Month_Average 3_Month_Average
## 1  1             2.2             2.3
## 2  1             2.7             3.0
## 3  1             3.2             2.0
## 4  1             3.3             2.3
## 5  1             3.3             4.3
## 6  1             3.8             5.3
```

Conclusion

This concludes a basic exploration of SQL Window Functions. They are a powerful tool and data scientists should be familiar with them.