Desafio - 07

Iuri Santos Oliveira - RA: 194610

2025-09-18

Atividade

```
library(RSQLite)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
if(!"discoCopy.db" %in% list.files("../dados/")){  # Se não houver o arquivo, ele será criado
file.copy("../dados/disco.db","../dados/discoCopy.db")} # Modificaremos esse arquivo
```

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

```
db <- dbConnect(SQLite(),"../dados/discoCopy.db")
```

```
dbListTables(db) # Lista as tabelas
```

```
##  [1] "albums"          "artists"         "customers"       "employees"
##  [5] "genres"          "invoice_items"   "invoices"        "media_types"
##  [9] "playlist_track"  "playlists"       "sqlite_sequence" "sqlite_stat1"
## [13] "tracks"
```

```
dbExecute(db,
"CREATE TABLE instruments
(AlbumId INTEGER,
TrackId INTEGER,
ElectricGuitar INTEGER,
Singer INTEGER,
Trumpet INTEGER)") # Cria uma tabela chamada "instruments" com as colunas e o seus tipos dentro dos par
```

```
## [1] 0
```

```
dbListFields(db,
'instruments') # Listar colunas da tabela
```

```
## [1] "AlbumId"        "TrackId"        "ElectricGuitar" "Singer"
## [5] "Trumpet"
```

```
dbExecute(db,
"DROP TABLE instruments") # Exclue a tabela
```

```
## [1] 0
```

```
aname = "Gilberto Gil"
sql = paste0("SELECT ArtistId FROM artists ", "WHERE Name = '", aname, "'")
aId = dbGetQuery(db, sql)
sql = paste('SELECT Title FROM albums', 'WHERE ArtistId =', aId)
dbGetQuery(db, sql)
```

```
##                                    Title
## 1                As Canções de Eu Tu Eles
## 2             Quanta Gente Veio Ver (Live)
## 3 Quanta Gente Veio ver--Bônus De Carnaval
```

```
# Aqui foram uma coluna de uma tabela com determinado nome de uma artista
# Caso utilizase {aname <- "Gilberto Gil'; DROP TABLE 'albums"} o banco de dados seria destruído
```

```
sql = paste("SELECT ArtistId FROM artists", "WHERE Name = ?")
query <- dbSendQuery(db, sql)
dbBind(query, list("Gilberto Gil"))
aId <- dbFetch(query)
dbClearResult(query)
# Segundo passo interno, não deve causar problema
sql = paste('SELECT Title FROM albums', 'WHERE ArtistId =', aId)
dbGetQuery(db, sql)
```

```
##                                    Title
## 1                As Canções de Eu Tu Eles
## 2             Quanta Gente Veio Ver (Live)
## 3 Quanta Gente Veio ver--Bônus De Carnaval
```

```
# Uma outra forma de fazer o código acima
```

```
dbExecute(db,
"CREATE TABLE instruments
(AlbumId INTEGER,
TrackId INTEGER,
ElectricGuitar INTEGER,
Singer INTEGER,
Trumpet INTEGER)")
```

```
## [1] 0
```

```
dbListFields(db, 'instruments')
```

```
## [1] "AlbumId"        "TrackId"        "ElectricGuitar" "Singer"
## [5] "Trumpet"
```

```
# Listar as colunas da tabela criada
```

```
# Eu Tu Eles: AlbumId 85,
sql = paste('SELECT TrackId, Name FROM tracks','WHERE AlbumId = 85')
dbGetQuery(db, sql) %>% head
```

```
##   TrackId                Name
## 1    1073 Óia Eu Aqui De Novo
## 2    1074      Baião Da Penha
```

```
## 3    1075 Esperando Na Janela
## 4    1076            Juazeiro
## 5    1077 Último Pau-De-Arara
## 6    1078          Asa Branca
```

```
# Selecionar 2 colunas do album 85
```

```
dbExecute(db,
"INSERT INTO instruments
VALUES ('85','1075', 0, 1, 0),('85','1078', 0, 1, 0); ")
```

```
## [1] 2
```

```
# Insere valores para cada linha, representada por parênteses, números para as colunas
```

```
dbGetQuery(db,"SELECT * FROM instruments")
```

```
##   AlbumId TrackId ElectricGuitar Singer Trumpet
## 1      85    1075              0      1       0
## 2      85    1078              0      1       0
```

```
# Seleciona a tabela "instruments'
```

```
dbWriteTable(db,"mtcars", mtcars)
dbListTables(db)
```

```
##  [1] "albums"          "artists"         "customers"       "employees"
##  [5] "genres"          "instruments"     "invoice_items"   "invoices"
##  [9] "media_types"     "mtcars"          "playlist_track"  "playlists"
## [13] "sqlite_sequence" "sqlite_stat1"    "tracks"
```

```
# Cria uma tebela chamada "mtcars"e insere o banco de dados mtcars do R (se a tabela já existe, ela ser
# Mostra os nomes das colunas
```

```
dbGetQuery(db,"SELECT * FROM mtcars") %>% head(3)
```

```
##    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## 1 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## 2 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## 3 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

```
# Selecionou a tabela mtcars e lê as 3 primeiras linhas
```

```
# Será criada uma nova tabela em que cada coluna será atribuíada um valor numérico aleatório com até 2
theAvgCar <- mtcars %>% summarise_all(function(x) round(mean(x), 2))
theAvgCar
```

```
##     mpg  cyl   disp     hp drat   wt  qsec   vs   am gear carb
## 1 20.09 6.19 230.72 146.69  3.6 3.22 17.85 0.44 0.41 3.69 2.81
```

```
# Na tabela mtcars insere a tabela theAvgCar e o 'append' coloca os dados da tabela no final
dbWriteTable(db,"mtcars", theAvgCar, append = TRUE)

# Aqui a tabela mtcars de db é selecionada e mostra as 3 últimas linhas
dbGetQuery(db, "SELECT * FROM mtcars") %>% tail(3)
```

```
##      mpg  cyl   disp     hp drat   wt  qsec   vs   am gear carb
## 31 15.00 8.00 301.00 335.00 3.54 3.57 14.60 0.00 1.00 5.00 8.00
## 32 21.40 4.00 121.00 109.00 4.11 2.78 18.60 1.00 1.00 4.00 2.00
## 33 20.09 6.19 230.72 146.69 3.60 3.22 17.85 0.44 0.41 3.69 2.81
```

```r
# Substirui a tabela mtcars para o banco de dados original e se ela já existir, será substitída pela fu
dbWriteTable(db,"mtcars", mtcars, overwrite = TRUE)

# Seleciona as 3 últimas linhas da tabela mtcars
dbGetQuery(db,"SELECT * FROM mtcars") %>% tail(3)
```

```
##     mpg cyl disp  hp drat   wt qsec vs am gear carb
## 30 19.7   6  145 175 3.62 2.77 15.5  0  1    5    6
## 31 15.0   8  301 335 3.54 3.57 14.6  0  1    5    8
## 32 21.4   4  121 109 4.11 2.78 18.6  1  1    4    2
```

```r
# Envia uma consulta SQL para selecionar todos os carros com 4 cilindros
res <- dbSendQuery(db, "SELECT * FROM mtcars WHERE cyl = 4")
# Há um loop enquanto houver resultados para buscar os resultados em chunks de 5 linhas por vez
while(!dbHasCompleted(res)){
  chunk <- dbFetch(res, n = 5)
  print(nrow(chunk))
}
```

```
## [1] 5
## [1] 5
## [1] 1
```

```r
dbClearResult(res)
```

```r
dbDisconnect(db)
# Verifica se o arquivo do banco copy existe no diretório e remove o arquivo do banco copy
if("discoCopy.db" %in% list.files("../dados/")){
file.remove("../dados/discoCopy.db")
}
```

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

```r
# Lê o arquivo CSV de aeroportos com tipos de colunas específicos
airports <- read_csv("../dados/airports.csv", col_types = "ccccdd")

# Lê o arquivo CSV de companhias aéreas com tipos de colunas específicos
airlines <- read_csv("../dados/airlines.csv", col_types = "cc")

# Conecta a um novo banco de dados SQLite chamado "air.db"
air <- dbConnect(SQLite(), dbname="../dados/air.db")

# Escreve os dados de aeroportos em uma tabela no banco
dbWriteTable(air, name = "airports", airports)

# Escreve os dados de companhias aéreas em uma tabela no banco
dbWriteTable(air, name = "airlines", airlines)

# Lista as tabelas
dbListTables(air)
```

```
## [1] "airlines" "airports"
```

```r
#  Destroe a conexão e a tabela
dbDisconnect(air)
if("air.db" %in% list.files("../dados/")){
file.remove("../dados/air.db")
```

```
}
```

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

```
library(RSQLite)
library(tidyverse)
library(dbplyr)
```

```
##
## Anexando pacote: 'dbplyr'
```

```
## Os seguintes objetos são mascarados por 'package:dplyr':
##
##     ident, sql
```

```
# Mostrando diferença entre tidyverse e dbplyr
db <- dbConnect(SQLite(),"../dados/disco.db") # original
tracks <- tbl(db,"tracks") # dplyr
tracks %>% head(3)
```

```
## # Source:   SQL [3 x 9]
## # Database: sqlite 3.46.0 [\\smb\ra194610\Documentos\ME315\Desafios\dados\disco.db]
##   TrackId Name          AlbumId MediaTypeId GenreId Composer Milliseconds  Bytes
##     <int> <chr>           <int>       <int>   <int> <chr>           <int>  <int>
## 1       1 For Those Ab~       1           1       1 Angus Y~       343719 1.12e7
## 2       2 Balls to the~       2           2       1 <NA>           342562 5.51e6
## 3       3 Fast As a Sh~       3           2       1 F. Balt~       230619 3.99e6
## # i 1 more variable: UnitPrice <dbl>
```

```
# Agrupa por AlbumId e calcula médias
meanTracks <- tracks %>%
group_by(AlbumId) %>%
summarise(AvLen = mean(Milliseconds, na.rm = TRUE),
AvCost = mean(UnitPrice, na.rm = TRUE))

meanTracks
```

```
## # Source:   SQL [?? x 3]
## # Database: sqlite 3.46.0 [\\smb\ra194610\Documentos\ME315\Desafios\dados\disco.db]
##    AlbumId   AvLen AvCost
##      <int>   <dbl>  <dbl>
##  1       1 240042.   0.99
##  2       2 342562    0.99
##  3       3 286029.   0.99
##  4       4 306657.   0.99
##  5       5 294114.   0.99
##  6       6 265456.   0.99
##  7       7 270780.   0.99
##  8       8 207638.   0.99
##  9       9 333926.   0.99
## 10      10 280551.   0.99
## # i more rows
```

```
# Exibe o código SQL equivalente que será executado no banco
meanTracks %>% show_query()
```

```
## <SQL>
## SELECT `AlbumId`, AVG(`Milliseconds`) AS `AvLen`, AVG(`UnitPrice`) AS `AvCost`
```

```
## FROM `tracks`
## GROUP BY `AlbumId`
```

```
 # Converte o resultado dbplyr para dataframe normal

mT <- meanTracks %>% collect()
mT
```

```
## # A tibble: 347 x 3
##    AlbumId   AvLen AvCost
##      <int>   <dbl>  <dbl>
##  1       1 240042.   0.99
##  2       2 342562    0.99
##  3       3 286029.   0.99
##  4       4 306657.   0.99
##  5       5 294114.   0.99
##  6       6 265456.   0.99
##  7       7 270780.   0.99
##  8       8 207638.   0.99
##  9       9 333926.   0.99
## 10      10 280551.   0.99
## # i 337 more rows
```