SQL - Exercices (Correction)

October 2020

Le vélo STAR

Nous considérons les données des stations de vélos en libre service STAR de la ville de Rennes. Une copie de la base SQLite est disponible dans le fichier `LEveloSTAR.sqlite3`. Utiliser les fonctions du package `DBI` pour répondre aux questions suivantes.

1. Se connecter à la base de données et afficher les tables. Pour chaque table, afficher les variables.

```
dbname <- file.path("data", "LEveloSTAR.sqlite3")
con <- dbConnect(RSQLite::SQLite(), dbname = dbname)
dbListTables(con)
```

```
## [1] "Etat"      "Topologie"
```

```
for(table in dbListTables(con)) {
  cat("--- Table", table, "---\n")
  print(dbListFields(con, table))
}
```

```
## --- Table Etat ---
##  [1] "id"                       "nom"
##  [3] "latitude"                 "longitude"
##  [5] "etat"                     "nb_emplacements"
##  [7] "emplacements_disponibles" "velos_disponibles"
##  [9] "date"                     "data"
## --- Table Topologie ---
##  [1] "id"                "nom"               "adresse_numero"
##  [4] "adresse_voie"      "commune"           "latitude"
##  [7] "longitude"         "id_correspondance" "mise_en_service"
## [10] "nb_emplacements"   "id_proche_1"       "id_proche_2"
## [13] "id_proche_3"       "terminal_cb"
```

2. Sélectionner l'identifiant `id`, le nom `nom` et l'identifiant de la station proche `id_proche_1` depuis la table `Topologie`.

```
query <- "
SELECT id, nom, id_proche_1
FROM Topologie
"
dbGetQuery(con, query) %>% head(3)
```

```
##   id           nom id_proche_1
## 1  1    République           2
## 2  2        Mairie           1
## 3  3 Champ Jacquet           2
```

3. Faire une jointure pour créer une table qui contient la liste des stations avec l'identifiant, le nom et le nom de la station proche associée à l'identifiant `id_proche_1` de la table `Topologie`.

```
query <- "
SELECT left.id AS id,
       left.nom AS nom,
       right.nom AS nom_proche
FROM Topologie AS left
LEFT JOIN Topologie AS right
ON (left.id_proche_1 = right.id)
"
dbGetQuery(con, query) %>% head(3)
```

```
##   id           nom nom_proche
## 1  1    République     Mairie
## 2  2        Mairie République
## 3  3 Champ Jacquet     Mairie
```

4. Ajouter à la table précédente la distance euclidienne entre la station et la station proche associée à l'identifiant `id_proche_1` de la table `Topologie`.

```
query <- "
SELECT left.id AS id,
       left.nom AS nom,
       right.nom AS nom_proche,
       (
         POWER((left.latitude - right.latitude), 2.0) +
         POWER((left.longitude - right.longitude), 2.0)
       ) AS distance
FROM Topologie AS left
LEFT JOIN Topologie AS right
ON (left.id_proche_1 = right.id)
"
dbGetQuery(con, query) %>% head(3)
```

```
##   id           nom nom_proche     distance
## 1  1    République     Mairie 3.071417e-06
## 2  2        Mairie République 3.071417e-06
## 3  3 Champ Jacquet     Mairie 3.003143e-06
```

5. Nous nous trouvons au point de coordonnées (48.1179151,-1.7028661). Créer une table avec le nom des trois stations les plus proches classées par ordre de distance et le nombre d'emplacements libres dans ces stations.

```
ma_latitude <- 48.1179151
ma_longitude = -1.7028661

query <- paste0("
SELECT nom,
       (
         POWER((latitude - ", ma_latitude, "), 2.0) +
         POWER((longitude - ", ma_longitude, "), 2.0)
       ) AS distance,
       emplacements_disponibles
FROM Etat
ORDER BY distance
LIMIT 3
")
dbGetQuery(con, query)
```

```
##                     nom     distance emplacements_disponibles
## 1               Berger 7.543101e-06                          4
## 2 Villejean-Université 1.156770e-05                         13
## 3              Marbeuf 3.864411e-05                         10
```

6. Reprendre les questions précédentes en utilisant les fonctions de `dplyr`.

```
# Question 1
db_etat <- tbl(con, "Etat")
colnames(db_etat)
```

```
##  [1] "id"                       "nom"
##  [3] "latitude"                 "longitude"
##  [5] "etat"                     "nb_emplacements"
##  [7] "emplacements_disponibles" "velos_disponibles"
##  [9] "date"                     "data"
```

```
db_topologie <- tbl(con, "Topologie")
colnames(db_topologie)
```

```
##  [1] "id"                "nom"               "adresse_numero"
##  [4] "adresse_voie"      "commune"           "latitude"
##  [7] "longitude"         "id_correspondance" "mise_en_service"
## [10] "nb_emplacements"   "id_proche_1"       "id_proche_2"
## [13] "id_proche_3"       "terminal_cb"
```

```
# Question 2
db_topologie %>%
  select(id, nom, id_proche_1) %>%
  head(3) %>%
  collect()
```

```
## # A tibble: 3 x 3
##      id nom           id_proche_1
##   <int> <chr>               <int>
## 1     1 République              2
## 2     2 Mairie                  1
## 3     3 Champ Jacquet           2
```

```
# Question 3
db_topologie %>%
  left_join(db_topologie, by = c("id_proche_1" = "id")) %>%
  rename(nom = nom.x, nom_proche = nom.y) %>%
  select(id, nom, nom_proche) %>%
  head(3) %>%
  collect()
```

```
## # A tibble: 3 x 3
##      id nom           nom_proche
##   <int> <chr>         <chr>
## 1     1 République    Mairie
## 2     2 Mairie        République
## 3     3 Champ Jacquet Mairie
```

```
# Question 4
db_topologie %>%
  left_join(db_topologie, by = c("id_proche_1" = "id")) %>%
  mutate(distance = (latitude.x - latitude.y)^2 + (longitude.x - longitude.y)^2) %>%
```

```
  rename(nom = nom.x, nom_proche = nom.y) %>%
  select(id, nom, nom_proche, distance) %>%
  head(3) %>%
  collect()
```

```
## # A tibble: 3 x 4
##      id nom           nom_proche   distance
##   <int> <chr>         <chr>           <dbl>
## 1     1 République    Mairie     0.00000307
## 2     2 Mairie        République 0.00000307
## 3     3 Champ Jacquet Mairie     0.00000300
```

```
# Question 5
db_etat %>%
  mutate(distance = (latitude - ma_latitude)^2 + (longitude - ma_longitude)^2) %>%
  arrange(distance) %>%
  select(nom, distance, emplacements_disponibles) %>%
  head(3) %>%
  collect()
```

```
## # A tibble: 3 x 3
##   nom                    distance emplacements_disponibles
##   <chr>                     <dbl>                    <int>
## 1 Berger                0.00000754                        4
## 2 Villejean-Université 0.0000116                        13
## 3 Marbeuf               0.0000386                       10
```

7. Terminer correctement en fermant la connexion à la base de données.

```
dbDisconnect(con)
```

Musique

Sur le site https://github.com/lerocha/chinook-database, nous pouvons trouver des bases de données de bibliothèques musicales. Une copie de la base SQLite est disponible dans le fichier `Chinook_Sqlite.sqlite`.

1. Se connecter à la base de données et afficher les tables. Explorer le jeu de données pour le découvrir. En particulier, étudier comment les tables `Playlist`, `PlaylistTrack` et `Track` sont liées.

```
dbname <- file.path("data", "Chinook_Sqlite.sqlite")
con <- dbConnect(RSQLite::SQLite(), dbname = dbname)
dbListTables(con)
```

```
##  [1] "Album"         "Artist"        "Customer"      "Employee"
##  [5] "Genre"         "Invoice"       "InvoiceLine"   "MediaType"
##  [9] "Playlist"      "PlaylistTrack" "Track"
```

```
db_playlist <- tbl(con, "Playlist")
db_playlist %>% head(3) %>% collect()
```

```
## # A tibble: 3 x 2
##   PlaylistId Name
##        <int> <chr>
## 1          1 Music
## 2          2 Movies
## 3          3 TV Shows
```

```
db_playlist_track <- tbl(con, "PlaylistTrack")
db_playlist_track %>% head(3) %>% collect()
```

```
## # A tibble: 3 x 2
##   PlaylistId TrackId
##        <int>   <int>
## 1          1    3402
## 2          1    3389
## 3          1    3390
```

```
db_track <- tbl(con, "Track")
db_track %>% head(3) %>% collect()
```

```
## # A tibble: 3 x 9
##   TrackId Name  AlbumId MediaTypeId GenreId Composer Milliseconds  Bytes
##     <int> <chr>   <int>       <int>   <int> <chr>           <int>  <int>
## 1       1 For ~       1           1       1 Angus Y~       343719 1.12e7
## 2       2 Ball~       2           2       1 <NA>           342562 5.51e6
## 3       3 Fast~       3           2       1 F. Balt~       230619 3.99e6
## # ... with 1 more variable: UnitPrice <dbl>
```

2. Utiliser les verbes de `dplyr` pour savoir quelles sont les playlists qui contiennent le plus de pistes.

```
db_playlist_track %>%
  group_by(PlaylistId) %>%
  summarise(n = n()) %>%
  left_join(db_playlist, by = "PlaylistId") %>%
  arrange(desc(n)) %>%
  collect()
```

```
## # A tibble: 14 x 3
##    PlaylistId     n Name
##         <int> <int> <chr>
##  1          1  3290 Music
##  2          8  3290 Music
##  3          5  1477 90's Music
##  4          3   213 TV Shows
##  5         10   213 TV Shows
##  6         12    75 Classical
##  7         11    39 Brazilian Music
##  8         17    26 Heavy Metal Classic
##  9         13    25 Classical 101 - Deep Cuts
## 10         14    25 Classical 101 - Next Steps
## 11         15    25 Classical 101 - The Basics
## 12         16    15 Grunge
## 13          9     1 Music Videos
## 14         18     1 On-The-Go 1
```

3. En utilisant `dplyr`, construire une table contenant les informations suivantes sur la playlist appelée `Classical` : le titre de chaque piste ainsi que le titre de l'album dont cette piste est tirée.

```
db_album <- tbl(con, "Album")
db_playlist_track %>%
  left_join(db_playlist, by = "PlaylistId") %>%
  rename(PlaylistName = Name) %>%
  filter(PlaylistName == "Classical") %>%
  left_join(db_track, by = "TrackId") %>%
```

```
  left_join(db_album, by = "AlbumId") %>%
  select(Name, Title) %>%
  collect()
```

```
## # A tibble: 75 x 2
##    Name                                     Title
##    <chr>                                    <chr>
##  1 "Intoitus: Adorate Deum"                 Adorate Deum: Gregorian Chant fro~
##  2 "Miserere mei, Deus"                     Allegri: Miserere
##  3 "Canon and Gigue in D Major: I. Canon"   Pachelbel: Canon & Gigue
##  4 "Concerto No. 1 in E Major, RV 269 \"Spri~ Vivaldi: The Four Seasons
##  5 "Concerto for 2 Violins in D Minor, BWV 1~ Bach: Violin Concertos
##  6 "Aria Mit 30 Veränderungen, BWV 988 \"Gol~ Bach: Goldberg Variations
##  7 "Suite for Solo Cello No. 1 in G Major, B~ Bach: The Cello Suites
##  8 "The Messiah: Behold, I Tell You a Myster~ Handel: The Messiah (Highlights)
##  9 "Solomon HWV 67: The Arrival of the Queen~ The World of Classical Favourites
## 10 "\"Eine Kleine Nachtmusik\" Serenade In G~ Sir Neville Marriner: A Celebrati~
## # ... with 65 more rows
```

4. Même question en écrivant directement la requête en SQL.

```
query <- "
SELECT Name, Title
FROM (
  SELECT Name, AlbumId
  FROM (
    SELECT TrackId
    FROM PlaylistTrack AS left
    LEFT JOIN Playlist AS right
    ON (left.PlaylistId = right.PlaylistId)
    WHERE (Name = 'Classical')
  ) AS left
  LEFT JOIN Track AS right
  ON (left.TrackId = right.TrackId)
) AS left
LEFT JOIN Album AS right
ON (left.AlbumId = right.AlbumId)
"
dbGetQuery(con, query) %>% head(3)
```

```
##                                 Name
## 1              Intoitus: Adorate Deum
## 2                  Miserere mei, Deus
## 3 Canon and Gigue in D Major: I. Canon
##                                                        Title
## 1 Adorate Deum: Gregorian Chant from the Proper of the Mass
## 2                                          Allegri: Miserere
## 3                                  Pachelbel: Canon & Gigue
```

5. Terminer correctement en fermant la connexion à la base de données.

```
dbDisconnect(con)
```