Desafio08ME315

Gabriella de Oliveira Argenton 255677

Invalid Date

1

```r
library(RSQLite)

# caminho relativo (mesma pasta do projeto)
db_file <- file.path("dados", "database.sqlite3")

conn <- dbConnect(SQLite(), dbname = db_file)

# conferir tabelas
dbListTables(conn)
```

```
 [1] "course_offerings"    "courses"             "grade_distributions"
 [4] "instructors"         "rooms"               "schedules"
 [7] "sections"            "subject_memberships" "subjects"
[10] "teachings"
```

2

```r
# Professores de STAT via nome do oferecimento (ex.: "STAT 302 ...")
sql_prof_stat_fallback <- "
WITH stat_sections AS (
  SELECT sec.uuid AS section_uuid
  FROM course_offerings co
  JOIN sections sec ON sec.course_offering_uuid = co.uuid
  WHERE UPPER(co.name) LIKE 'STAT %'   -- <- chave do fallback
)
SELECT DISTINCT i.name AS professor
FROM stat_sections ss
JOIN teachings t   ON t.section_uuid = ss.section_uuid
JOIN instructors i ON i.id          = t.instructor_id
WHERE i.name IS NOT NULL AND TRIM(i.name) <> ''
ORDER BY i.name;
"

prof_stat <- dbGetQuery(conn, sql_prof_stat_fallback)
n_prof <- nrow(prof_stat)
cat("Número de professores que lecionaram STAT:", n_prof, "\n")
```

```
Número de professores que lecionaram STAT: 128
```

```
print(prof_stat)
```

```
                       professor
1                ABIGAIL BENZINE
2                 ADRIENNE WOOD
3                AKICHIKA OZEKI
4                ALEXANDER TAHK
5                  AMANDA EGGEN
6                    AMY ATWOOD
7                 ANNE BRUCKNER
8                    ANRU ZHANG
9                    BAS ROKERS
10                 BRET HANLON
11                 BRET LARGET
12            BRYAN SEAN KELLER
13                CAROL ECKERLY
14                   CECILE ANE
15                   CHEN JING
16              CHRISTOPHER COX
17  CHRISTOPHER MICHAEL SWOBODA
18                   COLE COOK
19              COLLEEN F MOORE
20               COURTNEY HALL
21                      DAN SU
22                DANIEL ADAMS
23              DANIEL BRADFORD
24               DANIEL WRIGHT
25                DAVID WEIMER
26                  DEBRAJ DAS
27                  DEREK BEAN
```

28	DONALD PORTER
29	DONGGYU KIM
30	EDWARD ERKER
31	ERICA LEE DEADMAN
32	FAN GAO
33	FAN YANG
34	GONZALO CONTADOR
35	GUANNAN SUN
36	GUN WOONG PARK
37	HAN CHEN
38	HAO CHEN
39	HAO ZHENG
40	HAO ZHOU
41	HEATHER MARIE BRAZEAU
42	HYEBIN SONG
43	JEE YEON KIM
44	JESSE KAYE
45	JIAJIE CHEN
46	JINGJIANG PENG
47	JIWEI ZHAO
48	JODI WOLLACK
49	JOHN J. CURTIN
50	JOSEPH RYAN NEWTON
51	JOSHUA BRANDT SCHIFFMAN
52	JUN ZHU
53	KATHERINE KORTENKAMP
54	KENNETH POTTER
55	KRISTINA KELLETT
56	LIFAN YU
57	LILUN DU
58	LU YANG
59	LUWAN ZHANG
60	MARKUS BRAUER
61	MARTIN ZETTERSTEN
62	MELANIE FUHRMANN
63	MENG SONG
64	MICHAEL AMATO
65	MICHAEL NOH
66	MICHELE VOLBRECHT
67	MITCHELL CAMPBELL
68	MOO K CHUNG
69	MURRAY CLAYTON
70	NICHOLAS STEPHEN KEULER

71	NORBERT BINKIEWICZ
72	PAIGE MISSION
73	PATRICK SCHNARRENBERGER
74	PAUL SAVARIAPPAN
75	PEIGEN ZHOU
76	PERLA REYES
77	PHOEBE JORDAN
78	POOJA SIDNEY
79	QI JIANG
80	QI TANG
81	QIAN ZHIGUANG
82	QUEFENG LI
83	RACHEL SALK
84	REBECCA ANN MCGILL
85	RICHARD J. CHAPPELL
86	ROBERT WAYNE GREEN
87	RONALD SERLIN
88	RUIFENG XU
89	RUOSI GUO
90	RYAN HANKE
91	SARAH ANN DEPAOLI
92	SARAH BROWN
93	SEHO PARK
94	SEO YOUNG LEE
95	SEUNGBONG HAN
96	SHENGJI JIA
97	SHUANG HUANG
98	SIJING LI
99	SONG WANG
100	SOOJIN PARK
101	SU-YOUNG KIM
102	TAMMI KRAL
103	TAO YU
104	THOMAS COOK
105	TIMOTHY SEAN GRANT
106	TING YE
107	WEI-YIN LOH
108	WELLINGTON AMARAL
109	WENWEN ZHANG
110	XIAO GUO
111	XIAO NIE
112	XIAOMING MA
113	XIN ZHANG

```
114                XINXIN YU
115               XINYU SONG
116                    XU HE
117                    XU XU
118                YALI WANG
119                 YAN CHEN
120                YAOYAO XU
121                  YI CHAI
122                YIFAN MEI
123               YING ZHANG
124              YONGNAM KIM
125           YOUNG MIN PARK
126         YOUNGDEOK HWANG
127                ZHUANG WU
128            ZUOFENG SHANG
```

3

```
sql_resumo <- "
WITH grade_points(letter, pts) AS (
  VALUES ('A',4.0), ('AB',3.5), ('B',3.0), ('BC',2.5), ('C',2.0), ('D',1.0), ('F',0.0)
),
stat_offerings AS (
  SELECT co.uuid AS course_offering_uuid, co.name AS offering_name
  FROM course_offerings co
  WHERE UPPER(co.name) LIKE 'STAT %'
),
-- notas em formato longo (converte TEXT -> INTEGER; ignora conceitos não numéricos)
long_counts AS (
  SELECT gd.course_offering_uuid, gd.section_number, 'A'  AS letter, CAST(NULLIF(TRIM(gd.a_c
  UNION ALL SELECT gd.course_offering_uuid, gd.section_number, 'AB', CAST(NULLIF(TRIM(gd.ab_
  UNION ALL SELECT gd.course_offering_uuid, gd.section_number, 'B' , CAST(NULLIF(TRIM(gd.b_c
  UNION ALL SELECT gd.course_offering_uuid, gd.section_number, 'BC', CAST(NULLIF(TRIM(gd.bc_
  UNION ALL SELECT gd.course_offering_uuid, gd.section_number, 'C' , CAST(NULLIF(TRIM(gd.c_c
  UNION ALL SELECT gd.course_offering_uuid, gd.section_number, 'D' , CAST(NULLIF(TRIM(gd.d_c
  UNION ALL SELECT gd.course_offering_uuid, gd.section_number, 'F' , CAST(NULLIF(TRIM(gd.f_c
),
-- GPA por seção STAT (só se houver alunos em A..F)
gpa_section AS (
  SELECT s.uuid AS section_uuid,
         so.offering_name,
         SUM(gp.pts * COALESCE(lc.n,0)) * 1.0 / SUM(COALESCE(lc.n,0)) AS gpa,
         SUM(COALESCE(lc.n,0)) AS total_n
  FROM long_counts lc
  JOIN grade_points gp ON gp.letter = lc.letter
  JOIN sections s
    ON s.course_offering_uuid = lc.course_offering_uuid
   AND s.number               = lc.section_number
  JOIN stat_offerings so ON so.course_offering_uuid = lc.course_offering_uuid
  GROUP BY s.uuid
  HAVING SUM(COALESCE(lc.n,0)) > 0
),
-- GPA médio por professor (STAT)
/* adicione HAVING COUNT(*) >= 3 para exigir mínimo de seções, se quiser */
prof_gpa AS (
  SELECT i.name AS professor,
         AVG(gs.gpa) AS gpa_media,
         COUNT(*)    AS n_sections
  FROM gpa_section gs
  JOIN teachings t   ON t.section_uuid = gs.section_uuid
  JOIN instructors i ON i.id          = t.instructor_id
  WHERE i.name IS NOT NULL AND TRIM(i.name) <> ''
  GROUP BY i.name
),
-- GPA médio por disciplina (STAT) usando o nome do offering
disc_gpa AS (
  SELECT gs.offering_name AS disciplina,
         AVG(gs.gpa) AS gpa_media,
         COUNT(*)    AS n_sections
  FROM gpa_section gs
  GROUP BY gs.offering_name
)
-- quatro respostas em um resultado só (4 linhas)
SELECT 'prof_mais_dificil' AS tipo, professor AS nome, gpa_media, n_sections
FROM ( SELECT professor, gpa_media, n_sections FROM prof_gpa ORDER BY gpa_media ASC,  n_secti
UNION ALL
SELECT 'prof_mais_facil', professor, gpa_media, n_sections
FROM ( SELECT professor, gpa_media, n_sections FROM prof_gpa ORDER BY gpa_media DESC, n_secti
UNION ALL
SELECT 'disc_mais_dificil', disciplina, gpa_media, n_sections
FROM ( SELECT disciplina, gpa_media, n_sections FROM disc_gpa ORDER BY gpa_media ASC,  n_sect
UNION ALL
SELECT 'disc_mais_facil', disciplina, gpa_media, n_sections
FROM ( SELECT disciplina, gpa_media, n_sections FROM disc_gpa ORDER BY gpa_media DESC, n_sect
```

```r
"

resumo <- dbGetQuery(conn, sql_resumo)
resumo
```

```
               tipo                       nome gpa_media n_sections
1 prof_mais_dificil              DONALD PORTER  3.275017          5
2   prof_mais_facil                YONGNAM KIM  4.000000          1
3 disc_mais_dificil      Stat Expermntl Design  3.331165         11
4   disc_mais_facil Stat Meth-Applied to Ed II  3.866745         36
```

4

```r
dbDisconnect(conn)
```

```r
cat("Relatório gerado em:", format(Sys.time(), "%d/%m/%Y %H:%M:%S"))
```

```
Relatório gerado em: 25/09/2025 11:23:26
```