Desafio 8

```r
library(RSQLite)
conn = dbConnect(SQLite(), 'database.sqlite3')
```

```r
query_professores_stat <- "
SELECT DISTINCT i.name AS professor
FROM instructors i
INNER JOIN teachings t ON i.id = t.instructor_id
INNER JOIN sections s ON t.section_uuid = s.uuid
INNER JOIN subject_memberships sm ON sm.course_offering_uuid = s.course_offering_uuid
INNER JOIN subjects sub ON sm.subject_code = sub.code
WHERE sub.abbreviation = 'STAT';
"

professores_stat <- dbGetQuery(conn, query_professores_stat)
print(professores_stat)  # printando resultados
```

```
                       professor
1                    MINJING TAO
2                  DONALD PORTER
3                     SHENG WANG
4                  KUNLING HUANG
5                       DONG XIA
6                JENNIFER NGUYEN
7                    SEULKEE YUN
8                     DUZHE WANG
9                    YONGFENG WU
10                   JARED BROWN
11                   YUCHEN ZHOU
12                    VICTOR LUO
13                GUN WOONG PARK
14                   BROOK LUERS
15                    DUY NGUYEN
16                  JOHN GILLETT
17                     QUOC TRAN
18                     YAOYAO XU
19                     JIE ZHANG
20                     LONG PHAN
21                  WESLEY CHANG
22                       TONG LI
23                  SHANE HUBLER
24                      FAN YANG
25                    JU HEE CHO
26                     JEEA CHOI
27                    SHIXUE LIU
28                   WEI-YIN LOH
29                       BIN DAI
30                    XIPEI YANG
31               DOUGLAS M. BATES
32           MICHAEL GEORGE ILTIS
```

33	LIAM JOHNSTON
34	CRYSTAL CHEN
35	YIFAN MEI
36	RUI CHEN
37	JILI WANG
38	LUWAN ZHANG
39	GINA BENNINGER
40	JIE SONG
41	XIWEN MA
42	HAODA FU
43	DEYUAN JIANG
44	JINGJIANG PENG
45	ERICA LEE DEADMAN
46	AKICHIKA OZEKI
47	MENG SONG
48	HAO TENG
49	DONGGYU KIM
50	COLIN LONGHURST
51	FREDERICK BOEHM
52	YONGSU LEE
53	GONZALO CONTADOR
54	HAO CHEN
55	XIN ZHANG
56	BO YANG
57	RUNGANG HAN
58	SEAN KENT
59	DEBRAJ DAS
60	YI LI
61	CLAIRE BOBST
62	CHAN PARK
63	SIJING LI
64	SHENGJI JIA
65	YUCHANG WU
66	ZHUANG WU
67	XIAOMAO LI
68	HYEBIN SONG
69	YOURAN QI
70	YOUNG MIN PARK
71	BIN ZHANG
72	QIURONG CUI
73	SEHO PARK
74	SHENG ZHANG
75	HUIKUN ZHANG
76	ALYSSA DIGILIO
77	YUQING XU
78	STEPHEN BERG
79	PEIGEN ZHOU
80	YAN CHEN
81	QIONG ZHANG
82	YOUNGDEOK HWANG
83	HAN CHEN
84	NICHOLAS STEPHEN KEULER
85	ZHENGXIAO WU

86	YUJIN CHUNG
87	WENWEN ZHANG
88	PERLA REYES
89	QI TANG
90	CHIA-CHIEH LIN
91	JIALE XU
92	YANG ZHAO
93	XINWEI DENG
94	MING XIE
95	JINGCI MENG
96	ISMOR FISCHER
97	JIAJIE CHEN
98	ZUOFENG SHANG
99	HAO ZHENG
100	XU XU
101	BIN ZHU
102	QUEFENG LI
103	YU QIU LIU
104	JUN LI
105	FAN GAO
106	VICTORIA MANSFIELD
107	ANQI SHI
108	KYLE HEBERT
109	XIUYU MA
110	CHENLIANG XU
111	CHANHAN HSU
112	YUANZHI LI
113	KAM-WAH TSUI
114	XINYU SONG
115	TIMOTHY IDOWU
116	XIAO NIE
117	CHEN CHENG
118	ANDREW LESLIE
119	YAOGUO XIE
120	KEVIN PACKARD
121	YI LIU
122	WENZHI CAO
123	JOSEPH DEUTSCH
124	LAN LUO
125	RYAN ZEA
126	ROBERT WARDROP
127	SANGBUM CHOI
128	XIN LI
129	SEUNGBONG HAN
130	ZHIGUO XIAO
131	RICHARD A. JOHNSON
132	RUI TANG
133	IAN BRANSTAD RILEY
134	BO HUANG
135	HEATHER MARIE BRAZEAU
136	XU HE
137	GARY HOWARD SCHROEDER
138	JUN ZHANG
139	CHENXI LI

[illegible]	[illegible]
140	SONA Z SWANSON
141	YUAN JIANG
142	JUNHEE HAN
143	LILUN DU
144	XIRAN WANG
145	ZIFENG ZHAO
146	CUIZE HAN
147	YING ZHANG
148	JOHN DAVIS
149	CHEN JING
150	JINGLAN LI
151	YI CHAI
152	THU LE
153	ALAN HUANG
154	HAOYANG FAN
155	BOWEN HU
156	TZU HSIANG HUNG
157	TUN LEE NG
158	ABIGAIL BENZINE
159	ALEXANDER COVINGTON
160	CHELSEY GREEN
161	MICHAEL KUTZLER
162	LILI ZHENG
163	REBECCA POST
164	LU YANG
165	RUOSI GUO
166	XUN ZHAO
167	MIN NIU
168	JIAN WU
169	WEI ZHENG
170	XINJIE HE
171	JIE WEI
172	CLAUDIA SOLIS LEMUS
173	XUEYAO CHEN
174	MANJUSHA KANCHARLA
175	MITCHELL PAUKNER
176	XIAOWU DAI
177	CHENGNING ZHANG
178	GINA OH
179	LUXI CAO
180	SONG WANG
181	LAM HO
182	TIEN VO
183	SHIZHEN WANG
184	SOHEIL SADEGHI
185	BINGYING XIE
186	GENG LI
187	JENNIFER BIRSTLER
188	MARIA KAMENETSKY
189	KYLE HERBET
190	JIANCHANG HU
191	TRAM TA
192	YUAN WANG

193	KAZUHIKO SHINKI
194	SIJIAN WANG
195	CHIEN-WEI CHEN
196	BEHZAD AALIPUR
197	DEREK BEAN
198	BEN ADAM HAALAND
199	ELOISA D CHAVAS
200	YALI WANG
201	VARSHA KULKARNI
202	SCOTT JOSEPH HETZEL
203	MIN JUNG LEE
204	FANG FANG
205	KRISTEN CYFFKA
206	TING-LI LIN
207	BRET LARGET
208	KATHERINE GOODE
209	ERIK NORDHEIM
210	NING FAN
211	ANRU ZHANG
212	GARVESH RASKUTTI
213	KRISHNAKUMAR BALASUBRAMANIAN
214	YAZHEN WANG
215	CHAOYANG YU
216	YILIN ZHANG
217	ZHENGJUN ZHANG
218	XIUFENG SHAO
219	YUAN LI
220	BRET HANLON
221	LILI LAN
222	CECILE ANE
223	ZHANG CHUNMING
224	LEI XU
225	MUHONG GAO
226	AUGUST JENSEN
227	XIAO GUO
228	THOMAS G. KURTZ
229	NICHOLAS HENDERSON
230	CHENSHENG KUANG
231	TAERI UHM
232	NORBERT BINKIEWICZ
233	KARL ROHE
234	JUNGWON MUN
235	JOSEP GINEBRA
236	TAO YU
237	XINXIN YU
238	HAO ZHOU
239	LIE XIONG
240	QIAN ZHIGUANG
241	SHANG WU
242	SHAN LU
243	STEPHEN AARON STANHOPE
244	LANCINE KONATE
245	GUILHERME VIEIRA NUNES LUDWIG
[illegible]	[illegible]

246	PAUL SAVARIAPPAN
247	CUICUI QI
248	KJELL DOKSUM
249	DEREK NORTON
250	KEEGAN KORTHAUER
251	KAILEI CHEN
252	THEVAASIINEN CHANDERENG
253	JOHN KANE
254	SOKOL VAKO
255	QING LI
256	YOUNG LEE
257	HYUNSEUNG KANG
258	BEHZAD AALIPUR HAFSHEJANI
259	SHULEI WANG
260	ZHONGJIE YU
261	LIN QI
262	JIWEI ZHAO
263	SUNDUZ KELES
264	SHUYUN YE
265	HUI WANG
266	KEVIN HASEGAWA ENG
267	XIAODAN WEI
268	MARIA ROJO
269	RUIFENG XU
270	YONGJOON KIM
271	NELLIE LAUGHLIN
272	MICHAEL HOGAN
273	KARL BROMAN
274	REBECCA KOSCIK
275	RONGJUN ZHU
276	BI CHENG WU
277	YUNONG LIN
278	ZHIGENG GENG
279	XIAOPING FENG
280	SHUANG HUANG
281	GUANNAN SUN
282	JEE YEON KIM
283	GREGORY SHINAULT
284	TONGHAI YANG
285	TIMO SEPPALAINEN
286	BENEDEK VALKO
287	DAVID GRIFFEATH
288	DAVID ANDERSON
289	PHILIP WOOD
290	FLORIAN BERTRAND
291	JASON RICHARD SWANSON
292	ALEXANDER KISELEV
293	SAMUEL STECHMANN
294	WAI TONG FAN
295	JUN YIN
296	STEFFEN LEMPP
297	RUIFANG SONG
298	JAMES D KUELBS
299	ANATOLE BECK

300	DONGHYUN LEE
301	ANDREJ ZLATOS
302	DANIELE CAPPELLETTI
303	MIHAELA IFRIM
304	SEBASTIEN ROCH
305	SCOTT HOTTOVY
306	JONATHON PETERSON
307	STEPHEN WAINGER
308	ALEXANDER FISH
309	GREGORIO MORENO-FLORES
310	SUKHENDU MEHROTRA
311	DIETRICH UHLENBROCK
312	LEV BORISOV
313	RICHARD A BRUALDI
314	ARNOLD MILLER
315	MATTHEW BALLARD
316	ALBRECHT KLEMM
317	PAUL M TERWILLIGER
318	STEVEN SAM
319	KEN ONO
320	ROBERT HARRON
321	YANNAN QIU
322	JOHN WILTSHIRE-GORDON
323	KYUNGMANN KIM
324	CHRISTOPHER WAGNER
325	PAUL RATHOUZ
326	STEPHEN WRIGHT
327	BENJAMIN RECHT
328	MICHAEL FERRIS
329	JESSE THOMAS HOLZER
330	HUILIN HU
331	ALBERTO DEL PIA
332	ROBERT R MEYER
333	SHI JIN
334	MARY LINDSTROM
335	ADIN-CRISTIAN ANDREI
336	MICHAEL LIOU
337	CHAOQUN MEI
338	CHRISTINA M. KENDZIORSKI
339	RONALD GANGNON
340	ROBERT WAYNE GREEN
341	COLE COOK
342	MURRAY CLAYTON
343	QI JIANG
344	JUN ZHU
345	ANNE BRUCKNER
346	TING YE
347	JUN SHAO
348	HUAIBAO FENG
349	ZIJIAN NI
350	THOMAS COOK
351	RICHARD J. CHAPPELL
352	MOO K CHUNG

```
353                 MICHAEL NEWTON
354                  BRIAN YANDELL
355                MICHELLE HARRIS
356                  NORMAN DRAPER
357                    YU MENGGANG
358                      MING YUAN
359                  FANGFANG WANG
360                      JUNHO LEE
361                    GRACE WAHBA
362                   JASON P FINE
363           MICHAEL RENE KOSOROK
364                      YAJUAN SI
365                         LU MAO
366                   DAVID DEMETS
367                 JAMES ANDERSON
368                   EDWARD ERKER
369                     SHUAI CHEN
370                     MARI PALTA
371                   GUANHUA CHEN
372  SRIKANTHMADHAVAN ARAVAMUTHAN
373          NATALIA DE LEON GATTI
374                 GUILHERME ROSA
375                  YANBING ZHENG
376             CHRISTINE SORKNESS
377                MARIAN R FISHER
```

```
query_count_professores_stat <- "
SELECT COUNT(DISTINCT i.name) AS num_professores
FROM instructors i
INNER JOIN teachings t ON i.id = t.instructor_id
INNER JOIN sections s ON t.section_uuid = s.uuid
INNER JOIN subject_memberships sm ON sm.course_offering_uuid = s.course_offering_uuid
INNER JOIN subjects sub ON sm.subject_code = sub.code
WHERE sub.abbreviation = 'STAT';
"

num_professores_stat <- dbGetQuery(conn, query_count_professores_stat)
print(num_professores_stat)
```

```
  num_professores
1             377
```

```
get_professor_by_gpa <- function(conn, type = c("min", "max")) {
  type <- match.arg(type)
  query <- sprintf("
WITH gpa_calculations AS (
  SELECT co.uuid AS course_offering_uuid,
         (CAST(gd.a_count AS INTEGER)*4 + CAST(gd.ab_count AS INTEGER)*3.5 +
          CAST(gd.b_count AS INTEGER)*3 + CAST(gd.bc_count AS INTEGER)*2.5 +
          CAST(gd.c_count AS INTEGER)*2 + CAST(gd.d_count AS INTEGER)*1 +
          CAST(gd.f_count AS INTEGER)*0) /
         NULLIF((CAST(gd.a_count AS INTEGER) + CAST(gd.ab_count AS INTEGER) +
                 CAST(gd.b_count AS INTEGER) + CAST(gd.bc_count AS INTEGER) +
                 CAST(gd.b_count AS INTEGER) + CAST(gd.bc_count AS INTEGER) +
                 CAST(gd.c_count AS INTEGER) + CAST(gd.d_count AS INTEGER) +
                 CAST(gd.f_count AS INTEGER)), 0) AS gpa
  FROM course_offerings co
  JOIN grade_distributions gd ON co.uuid = gd.course_offering_uuid
  JOIN subject_memberships sm ON co.uuid = sm.course_offering_uuid
  JOIN subjects sub ON sm.subject_code = sub.code
  WHERE sub.abbreviation = 'STAT'
    AND (CAST(gd.a_count AS INTEGER) + CAST(gd.ab_count AS INTEGER) +
         CAST(gd.b_count AS INTEGER) + CAST(gd.bc_count AS INTEGER) +
         CAST(gd.c_count AS INTEGER) + CAST(gd.d_count AS INTEGER) +
         CAST(gd.f_count AS INTEGER)) > 0
),
professor_gpas AS (
  SELECT i.id, i.name AS professor,
         AVG(gc.gpa) AS gpa_medio,
         COUNT(DISTINCT gc.course_offering_uuid) AS num_disciplinas
  FROM gpa_calculations gc
  JOIN sections s ON gc.course_offering_uuid = s.course_offering_uuid
  JOIN teachings t ON s.uuid = t.section_uuid
  JOIN instructors i ON t.instructor_id = i.id
  WHERE gc.gpa IS NOT NULL
  GROUP BY i.id, i.name
  HAVING COUNT(DISTINCT gc.course_offering_uuid) >= 1
),
agg_gpa AS (
  SELECT %s(gpa_medio) AS gpa_medio_val FROM professor_gpas
)
SELECT pg.professor,
       ROUND(pg.gpa_medio,4) AS gpa_medio,
       pg.num_disciplinas
FROM professor_gpas pg
JOIN agg_gpa ag ON pg.gpa_medio = ag.gpa_medio_val
ORDER BY pg.professor;
", ifelse(type == "min", "MIN", "MAX"))

  dbGetQuery(conn, query)
}

professor_dificil <- get_professor_by_gpa(conn, "min")
professor_facil <- get_professor_by_gpa(conn, "max")

print(professor_dificil)
```

```
       professor gpa_medio num_disciplinas
1 JAMES D KUELBS    2.5987               1
```

```
print(professor_facil)
```

```
                     professor gpa_medio num_disciplinas
1                 GUANHUA CHEN         4               1
2 SRIKANTHMADHAVAN ARAVAMUTHAN         4               1
3                    YAJUAN SI         4               1
```

```
get_course_by_gpa <- function(conn, type = c("min", "max")) {
  type <- match.arg(type)
  query <- sprintf("
WITH gpa_calculations AS (
  SELECT co.uuid AS course_offering_uuid,
         co.course_uuid AS course_uuid,
         (CAST(gd.a_count AS INTEGER)*4 + CAST(gd.ab_count AS INTEGER)*3.5 +
          CAST(gd.b_count AS INTEGER)*3 + CAST(gd.bc_count AS INTEGER)*2.5 +
          CAST(gd.c_count AS INTEGER)*2 + CAST(gd.d_count AS INTEGER)*1 +
          CAST(gd.f_count AS INTEGER)*0) /
         NULLIF((CAST(gd.a_count AS INTEGER) + CAST(gd.ab_count AS INTEGER) +
                 CAST(gd.b_count AS INTEGER) + CAST(gd.bc_count AS INTEGER) +
                 CAST(gd.c_count AS INTEGER) + CAST(gd.d_count AS INTEGER) +
                 CAST(gd.f_count AS INTEGER)), 0) AS gpa
  FROM course_offerings co
  JOIN grade_distributions gd ON co.uuid = gd.course_offering_uuid
  JOIN subject_memberships sm ON co.uuid = sm.course_offering_uuid
  JOIN subjects sub ON sm.subject_code = sub.code
  WHERE sub.abbreviation = 'STAT'
    AND (CAST(gd.a_count AS INTEGER) + CAST(gd.ab_count AS INTEGER) +
         CAST(gd.b_count AS INTEGER) + CAST(gd.bc_count AS INTEGER) +
         CAST(gd.c_count AS INTEGER) + CAST(gd.d_count AS INTEGER) +
         CAST(gd.f_count AS INTEGER)) > 0
),
course_gpas AS (
  SELECT c.number AS course_number,
         c.name AS course_name,
         AVG(gc.gpa) AS gpa_medio,
         COUNT(DISTINCT gc.course_offering_uuid) AS num_ofertas
  FROM gpa_calculations gc
  JOIN courses c ON gc.course_uuid = c.uuid
  WHERE gc.gpa IS NOT NULL
  GROUP BY c.number, c.name
  HAVING COUNT(DISTINCT gc.course_offering_uuid) >= 1
),
agg_gpa AS (
  SELECT %s(gpa_medio) AS gpa_val FROM course_gpas
)
SELECT cg.course_number,
       cg.course_name,
       ROUND(cg.gpa_medio,4) AS gpa_medio,
       cg.num_ofertas
FROM course_gpas cg
JOIN agg_gpa ag ON cg.gpa_medio = ag.gpa_val;
", ifelse(type == "min", "MIN", "MAX"))

  dbGetQuery(conn, query)
}

disciplina_dificil <- get_course_by_gpa(conn, "min")
disciplina_facil <- get_course_by_gpa(conn, "max")
```

```r
print(disciplina_dificil)
```

```
  course_number                           course_name gpa_medio num_ofertas
1           431 Introduction to the Theory of Probability    2.8916          22
```

```r
print(disciplina_facil)
```

```
  course_number                                           course_name gpa_medio
1           628                              Data Science Practicum         4
2           811                        Sample Survey Theory and Method         4
3           834                         Empir Proc&Semiparmtrc Infernc         4
4           841 Nonparametric Statistics and Machine Learning Methods         4
  num_ofertas
1           1
2           4
3           1
4           1
```

```r
dbDisconnect(conn)
```