

```

1  /*
2
3  Ejercicio 1: ¿Cuántos registros tiene cada tabla? Responder en un solo resultado.
4
5  */
6
7  SELECT 'autor' AS nombre_tabla, COUNT(*) AS cuenta FROM autor
8  UNION
9  SELECT 'categoria', COUNT(*) FROM categoria
10 UNION
11 SELECT 'inventario', COUNT(*) FROM inventario
12 UNION
13 SELECT 'libro', COUNT(*) FROM libro
14 UNION
15 SELECT 'pais', COUNT(*) FROM pais
16 UNION
17 SELECT 'prestamo', COUNT(*) FROM prestamo
18 UNION
19 SELECT 'usuario', COUNT(*) FROM usuario;
20
21
22 /*
23
24 Ejercicio 2: ¿Cuántos autores se tienen por cada nacionalidad en la biblioteca?
25
26 */
27
28 SELECT pais.nombre, COUNT(*)
29 FROM autor
30 INNER JOIN pais ON autor.nacionalidad_id = pais.id
31 GROUP BY pais.nombre
32 ORDER BY pais.nombre;
33
34
35 /*
36
37 Ejercicio 3: ¿Cuántos usuarios se tienen por cada nacionalidad en la biblioteca?
38
39 */
40 WITH usuarios_por_pais AS (
41     SELECT nacionalidad_id, COUNT(*) AS cuenta_usuarios
42     FROM usuario
43     GROUP BY nacionalidad_id
44 )
45
46 SELECT pais.nombre, usuarios_por_pais.cuenta_usuarios
47 FROM usuarios_por_pais
48 INNER JOIN pais ON usuarios_por_pais.nacionalidad_id = pais.id;
49
50 /*
51
52 Ejercicio 4: ¿Cuáles son los detalles del autor más viejo en la biblioteca? ¿Cuáles
    son los detalles del autor
53    más joven?
54
55 */
56
57 SELECT *
58 FROM autor
59 WHERE autor.fecha_nacimiento = (SELECT MIN(fecha_nacimiento) FROM autor)
60 UNION
61
62 SELECT *
63 FROM autor
64 WHERE autor.fecha_nacimiento = (SELECT MAX(fecha_nacimiento) FROM autor);
65
66

```

```

67 /*
68
69 Ejercicio 5: En promedio, ¿cuántos ejemplares se compran (por parte de la
    biblioteca) por libro?
70
71 */
72 WITH libros_agrupados AS (
73     SELECT libro_id, COUNT(*) AS cuenta_libros
74     FROM inventario
75     GROUP BY libro_id
76 )
77
78 SELECT AVG(cuenta_libros)
79 FROM libros_agrupados;
80
81
82 /*
83
84 Ejercicio 6: La biblioteca desea tener un reporte de todos los ejemplares que se
    han perdido. Incluir en el reporte
85 el ISBN, título y autor de libro así como su último usuario.
86
87 */
88 WITH CTE_prestamo AS (
89     SELECT *
90     FROM prestamo
91     WHERE prestamo.fecha_retorno IS NULL
92 )
93 SELECT libro.isbn,
94        libro.titulo,
95        autor.nombre,
96        autor.apellido,
97        usuario.nombre,
98        usuario.apellido
99 FROM CTE_prestamo
100 --INNER JOIN inventario ON inventario.libro_id = CTE_prestamo.inventario_id
101 INNER JOIN inventario ON inventario.id = CTE_prestamo.inventario_id
102 INNER JOIN libro ON inventario.libro_id = libro.id
103 INNER JOIN autor ON libro.autor_id = autor.id
104 INNER JOIN usuario ON CTE_prestamo.usuario_id = usuario.id;
105
106 /*
107
108 Ejercicio 7: ¿Cuántos préstamos se tienen por cada nacionalidad de usuario?
109
110 */
111
112 SELECT pais.nombre, COUNT(*)
113 FROM prestamo
114 INNER JOIN usuario ON prestamo.usuario_id = usuario.id
115 INNER JOIN pais ON usuario.nacionalidad_id = pais.id
116 GROUP BY pais.nombre
117 ORDER BY COUNT(*) DESC;
118
119 /*
120
121 Ejercicio 8: ¿De qué nacionalidad son los autores de los cuáles se han prestado más
    libros?
122
123 */
124
125 SELECT autor.nacionalidad_id, pais.nombre, COUNT(*)
126 FROM prestamo
127 INNER JOIN inventario ON prestamo.inventario_id = inventario.id
128 INNER JOIN libro ON inventario.libro_id = libro.id
129 INNER JOIN autor ON libro.autor_id = autor.id
130 INNER JOIN pais ON autor.nacionalidad_id = pais.id

```

```
131 GROUP BY autor.nacionalidad_id, pais.nombre
132 ORDER BY COUNT(*) DESC;
133
134 /*
135
136 Ejercicio 9: ¿Cuál es el hash de contraseña más común entre los usuarios?
137
138 */
139
140 SELECT contrasena, COUNT(*)
141 FROM usuario
142 GROUP BY contrasena
143 HAVING COUNT(*) > 1
144 ORDER BY COUNT(*) DESC;
145
146 /*
147
148 Ejercicio 10: Ejecuta un ataque de tipo rainbow table. Con los hashes precomputados
149 en la tabla 'rainbow',
150 proveer los correos de usuario y contraseñas (no hashes) de los usuarios vulnerables
151 a este hackeo.
152
153 */
154
155 SELECT usuario.correo, rainbow.contrasena
156 FROM usuario
157 INNER JOIN rainbow ON usuario.contrasena = rainbow.hash
158 ORDER BY correo;
159
160 /*
161
162 Ejercicio 11: Proporciona los detalles de los usuarios que no se encuentran
163 vulnerados por el rainbow table attack.
164 Utiliza un LEFT JOIN o RIGHT JOIN para hacerlo.
165
166 */
167
168 SELECT usuario.*
169 FROM usuario
170 LEFT JOIN rainbow ON usuario.contrasena = rainbow.hash
171 WHERE rainbow.id IS NULL
172 ORDER BY usuario.id;
173
174 /*
175
176 Ejercicio 12: La biblioteca cree que el COVID19 se puede contagiar a través de
177 libros. Proporciona una lista de todos
178 los pares de usuarios que hayan pedido en préstamo el mismo ejemplar con menos de 4
179 días de diferencia entre la
180 fecha de devolución del primer usuario y la fecha de inicio del segundo usuario.
181 Recordar que el usuario que
182 tuvo primero el libro es el potencial contagiador, el que lo pidió después es el
183 potencial contagiado. Favor de omitir
184 los casos en que el contagiador es el mismo usuario que el contagiado.
185
186 */
187
188 SELECT contagiador.inventario_id,
189        contagiador.usuario_id AS contagiador_id,
190        contagiado.usuario_id AS contagiado_id,
191        contagiador.fecha_retorno,
192        contagiado.fecha_inicio,
193        contagiado.fecha_inicio - contagiador.fecha_retorno AS diferencia_dias
194 FROM prestamo contagiador
195 INNER JOIN prestamo contagiado
196 ON contagiador.inventario_id = contagiado.inventario_id
197 AND contagiado.fecha_inicio > contagiador.fecha_retorno
```

```

191     AND (contagiado.fecha_inicio - contagiador.fecha_retorno) < 4
192     AND contagiador.usuario_id != contagiado.usuario_id
193 ORDER BY inventario_id, contagiador_id, contagiado_id;
194
195 /*
196
197 Ejercicio 13: ¿Cuál es el libro más solicitado?
198
199 */
200
201 SELECT inventario.libro_id, libro.titulo, COUNT(*)
202 FROM prestamo
203 INNER JOIN inventario ON prestamo.inventario_id = inventario.id
204 INNER JOIN libro ON inventario.libro_id = libro.id
205 GROUP BY inventario.libro_id, libro.titulo
206 ORDER BY COUNT(*) DESC;
207
208 /*
209
210 Ejercicio 14: ¿Cuál es el autor cuyos ejemplares se han perdido más? ¿Cuántos han
    sido?
211
212 */
213
214 WITH prestamos_perdidos AS (
215     SELECT *
216     FROM prestamo
217     WHERE fecha_retorno IS NULL
218           AND fecha_fin < NOW()
219 ),
220
221 recuento_prestamos_perdidos_por_autor AS (
222     SELECT autor.id, autor.nombre, autor.apellido, COUNT(*)
223     FROM prestamos_perdidos
224     INNER JOIN inventario ON prestamos_perdidos.inventario_id = inventario.id
225     INNER JOIN libro ON inventario.libro_id = libro.id
226     INNER JOIN autor ON libro.autor_id = autor.id
227     GROUP BY autor.id, autor.nombre, autor.apellido
228 )
229
230 SELECT *
231 FROM autor
232 WHERE id IN (
233     SELECT id
234     FROM recuento_prestamos_perdidos_por_autor
235     WHERE "count" = (SELECT MAX("count") FROM recuento_prestamos_perdidos_por_autor)
236 );
237
238 /*
239
240 Ejercicio 15: Obtener un reporte por usuario (usuario_id, libro_id) donde se
    especifique los libros distintos que
241 haya solicitado alguna vez.
242
243 */
244
245 SELECT DISTINCT prestamo.usuario_id, inventario.libro_id
246 FROM prestamo
247 INNER JOIN inventario ON prestamo.inventario_id = inventario.id
248 ORDER BY prestamo.usuario_id, inventario.libro_id;
249
250
251 /*
252
253 Ejercicio 16: ¿Hay algún libro que no se haya prestado ninguno de sus ejemplares?
254
255 */

```

```

256
257 SELECT *
258 FROM libro WHERE id IN (
259     SELECT libro.id
260     FROM libro
261     INNER JOIN inventario ON libro.id = inventario.libro_id
262     LEFT JOIN prestamo ON inventario.id = prestamo.inventario_id
263     GROUP BY libro.id
264     HAVING COUNT(prestamo.id) = 0
265 );
266
267 /*
268
269 Ejercicio 17: ¿De los préstamos que se regresan tarde (pero que sí se regresan)
270 cuál es el promedio, mínimo y
271 máximo de devolución tardía?
272 */
273
274 SELECT ROUND(AVG(fecha_retorno - prestamo.fecha_fin), 2),
275         MIN(fecha_retorno - prestamo.fecha_fin),
276         MAX(fecha_retorno - prestamo.fecha_fin)
277 FROM prestamo
278 WHERE fecha_retorno IS NOT NULL
279        AND fecha_retorno > fecha_fin;
280
281 /*
282
283 Ejercicio 18: ¿Cuántos ejemplares de libros se han perdido por año, usando la fecha
284 de inicio como referencia?
285 */
286
287 WITH ejemplares_no_devueltos AS (
288     SELECT *
289     FROM prestamo
290     WHERE fecha_retorno IS NULL
291           AND NOW() > fecha_fin
292 )
293
294 SELECT EXTRACT(YEAR FROM fecha_inicio) AS "año",
295        COUNT(*) AS conteo
296 FROM ejemplares_no_devueltos
297 GROUP BY EXTRACT(YEAR FROM fecha_inicio)
298 ORDER BY "año" DESC;
299
300 /*
301
302 Ejercicio 19: La biblioteca desea comenzar un sistema de recomendaciones de libros
303 . Genera un reporte por
304 usuario con libros recomendados para el (usuario_id, libro_id). Este sistema debe
305 funcionar usando como base libros
306 prestados a otros usuarios que hayan pedido en préstamo algún libro en común con el
307 usuario en cuestión. Favor de
308 omitir de la lista de recomendaciones, los libros que alguna vez el usuario ya haya
309 leído.
310 */
311
312 -- tabla extendida
313 WITH prestamo_usuario AS (
314     SELECT DISTINCT prestamo.usuario_id, inventario.libro_id
315     FROM prestamo
316     INNER JOIN inventario
317         ON prestamo.inventario_id = inventario.id
318 ),
319

```

```
317 -- sacar los pares de usuarios que tienen gustos en común
318 usuarios_gusto_comun AS (
319     SELECT DISTINCT original.usuario_id original_id,
320                     recomendador.usuario_id recomendador_id
321     FROM prestamo_usuario original
322     INNER JOIN prestamo_usuario recomendador
323         ON original.libro_id = recomendador.libro_id
324         AND original.usuario_id != recomendador.usuario_id
325     WHERE original.usuario_id < recomendador.usuario_id
326 )
327
328 SELECT DISTINCT original.usuario_id,
329                 recomendador.libro_id
330 FROM prestamo_usuario AS original
331 INNER JOIN usuarios_gusto_comun
332     ON original.usuario_id = usuarios_gusto_comun.original_id
333 INNER JOIN prestamo_usuario AS recomendador
334     ON usuarios_gusto_comun.recomendador_id = recomendador.usuario_id
335     AND recomendador.libro_id != original.libro_id
336 ORDER BY original.usuario_id, recomendador.libro_id;
337
```