

# Curso de Especialização em Data Science e Estatística Aplicada

## Banco de Dados - Atividade Avaliativa

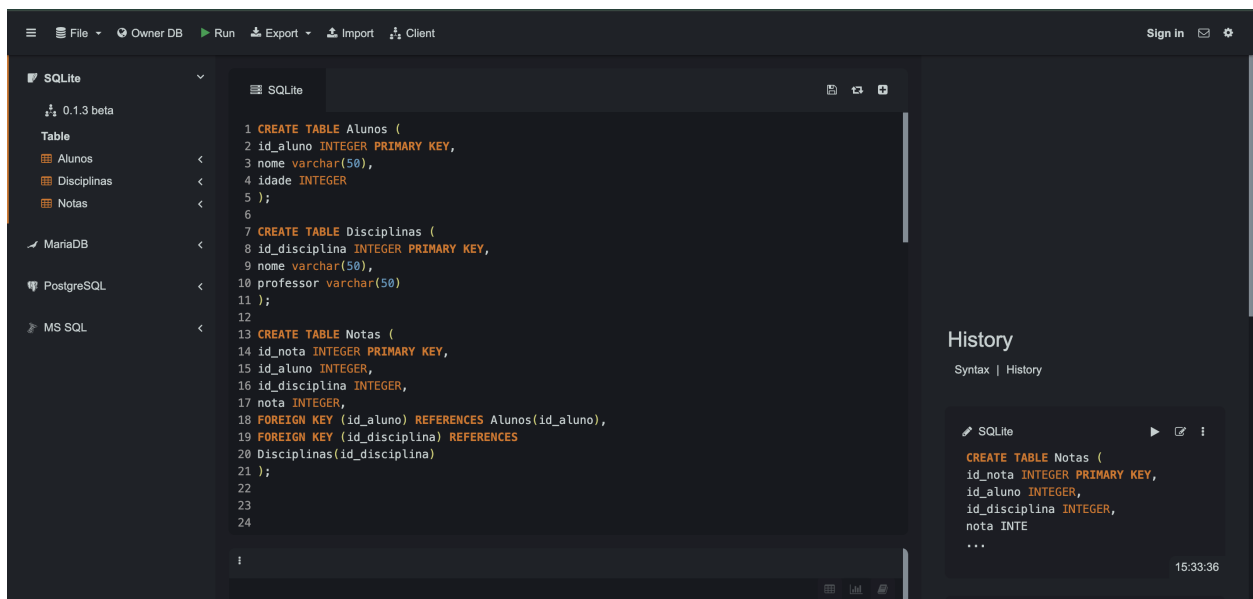
Ana Maria Alves da Silva

2024-06-08

**Questão 1.** Na ferramenta SQLite IDE, crie e execute os scripts do BD disponíveis em google drive.

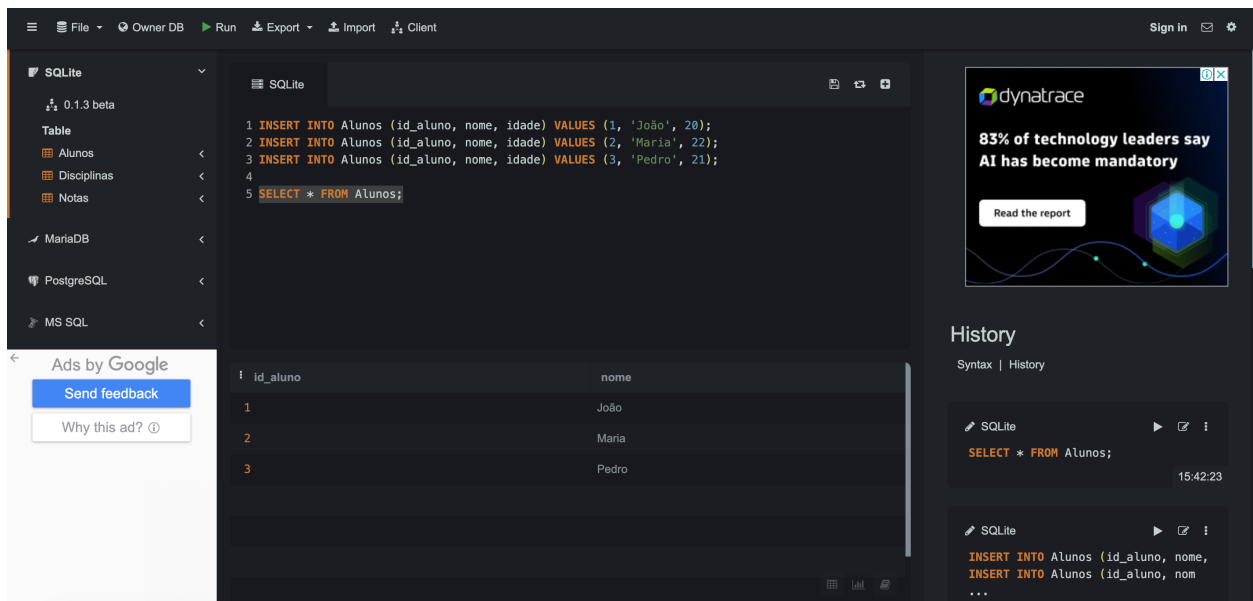
**Solução 1:**

- Passo 1: Criar as tabelas.

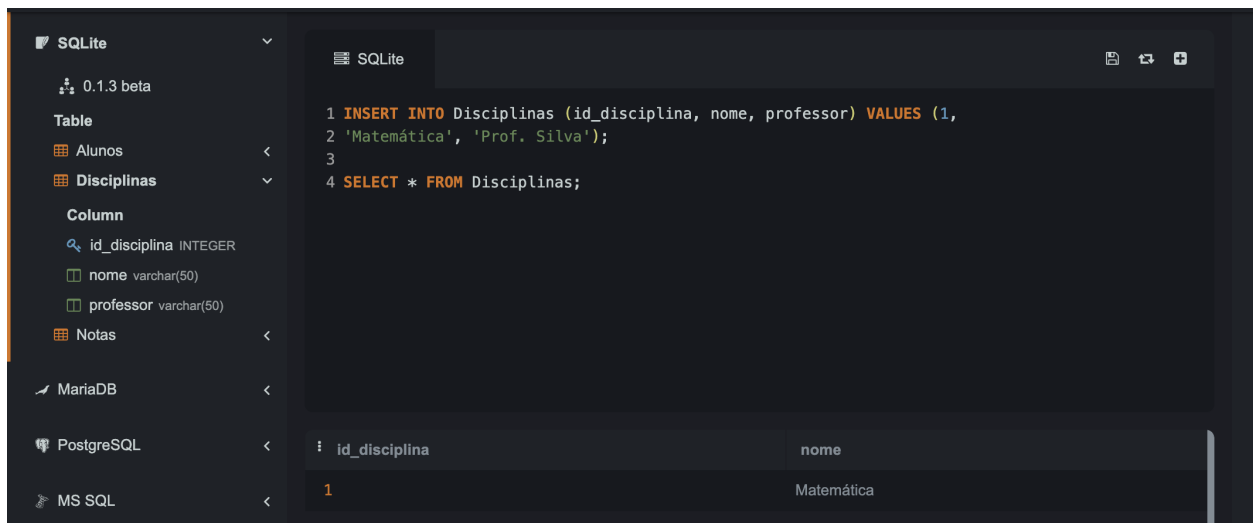


```
1 CREATE TABLE Alunos (  
2 id_aluno INTEGER PRIMARY KEY,  
3 nome varchar(50),  
4 idade INTEGER  
5 );  
6  
7 CREATE TABLE Disciplinas (  
8 id_disciplina INTEGER PRIMARY KEY,  
9 nome varchar(50),  
10 professor varchar(50)  
11 );  
12  
13 CREATE TABLE Notas (  
14 id_nota INTEGER PRIMARY KEY,  
15 id_aluno INTEGER,  
16 id_disciplina INTEGER,  
17 nota INTEGER,  
18 FOREIGN KEY (id_aluno) REFERENCES Alunos(id_aluno),  
19 FOREIGN KEY (id_disciplina) REFERENCES  
20 Disciplinas(id_disciplina)  
21 );  
22  
23  
24
```

- Passo 2: Inserir dados na tabela Alunos.



- Passo 3: Inserir dados na tabela Disciplina.



- Passo 4: Inserir dados na tabela Notas.

The screenshot shows a database management interface with a sidebar on the left containing a tree view of the database structure. The main area displays SQL queries and a table view for the 'Notas' table.

**Database Structure (Left Sidebar):**

- SQLite 0.1.3 beta
  - Table
    - Alunos
    - Disciplinas
    - Notas
      - Column
        - id\_nota INTEGER
        - id\_aluno INTEGER
        - id\_disciplina INTEGER
        - nota INTEGER

**SQL Queries (Main Area):**

```

1 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (1,1, 1, 8);
2 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (2,1, 1, 7);
3 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (3,1, 1, 9);
4
5 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (4,2, 1, 6);
6 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (5,2, 1, 8);
7 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (6,2, 1, 7);
8
9 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (7,3, 1, 9);
10 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (8,3, 1, 7);
11 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (9, 3, 1, 8);
12
13 SELECT * FROM Notas;
  
```

**Table View (Main Area):**

id_nota	id_aluno	id_disciplina
1	1	1
2	1	1
3	1	1
4	2	1
5	2	1
6	2	1
7	3	1

- Passo 5: Update de dados na tabela Alunos.

The screenshot shows a database management interface with a sidebar on the left containing a tree view of the database structure. The main area displays SQL queries and a table view for the 'Alunos' table.

**Database Structure (Left Sidebar):**

- SQLite 0.1.3 beta
  - Table
    - Alunos
      - Column
        - id\_aluno INTEGER
        - nome varchar(50)
        - idade INTEGER
    - Disciplinas
    - Notas

**SQL Queries (Main Area):**

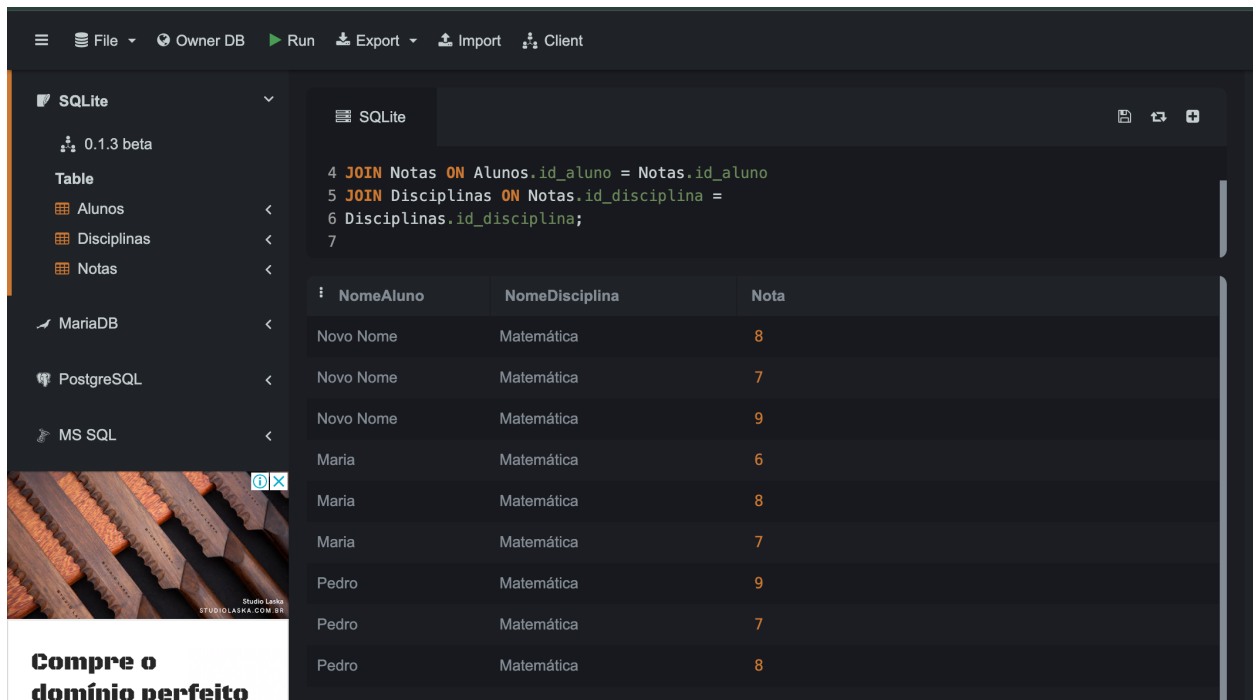
```

1 UPDATE Alunos SET nome = 'Novo Nome' WHERE id_aluno = 1;
2
3 SELECT * FROM alunos;
  
```

**Table View (Main Area):**

id_aluno	nome
1	Novo Nome
2	Maria
3	Pedro

- Passo 6: Visualizando as tabelas interligadas.



Questão 2. Na ferramenta SQLite IDE, insira linhas em todas as tabelas do BD criado.

Solução 2:

- Passo 1: Insert na tabela Alunos.

SQLite

```
1 INSERT INTO Alunos (id_aluno, nome, idade) VALUES (4, 'José', 25);
2 INSERT INTO Alunos (id_aluno, nome, idade) VALUES (5, 'Paula', 19);
3 INSERT INTO Alunos (id_aluno, nome, idade) VALUES (6, 'Lira', 26);
4
5 SELECT * FROM Alunos;
```

id_aluno	nome
1	Novo Nome
2	Maria
3	Pedro
4	José
5	Paula
6	Lira

- Passo 2: Insert na tabela Disciplinas.

SQLite

```
1 INSERT INTO Disciplinas (id_disciplina, nome, professor) VALUES (2,
2 'Português', 'Prof. Sousa');
3 INSERT INTO Disciplinas (id_disciplina, nome, professor) VALUES (3,
4 'Português', 'Prof. McGonagall');
5
6 SELECT * FROM Disciplinas;
```

id_disciplina	nome	professor
1	Matemática	Prof. Silva
2	Português	Prof. Sousa
3	Português	Prof. McGonagall

- Passo 3: Insert na tabela Notas.

SQLite

```
1 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (10,3, 2, 6);
2 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (11,3, 2, 5);
3 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (12, 3, 3, 4);
4 INSERT INTO Notas (id_nota, id_aluno, id_disciplina, nota) VALUES (13, 3, 3, 8);
5
6 SELECT * FROM notas;
```

id_nota	id_aluno	id_disciplina
3	1	1
4	2	1
5	2	1
6	2	1
7	3	1
8	3	1
9	3	1
10	3	2
11	3	2
12	3	3
13	3	3

- Passo 3: Insert na tabela Alunos.

SQLite

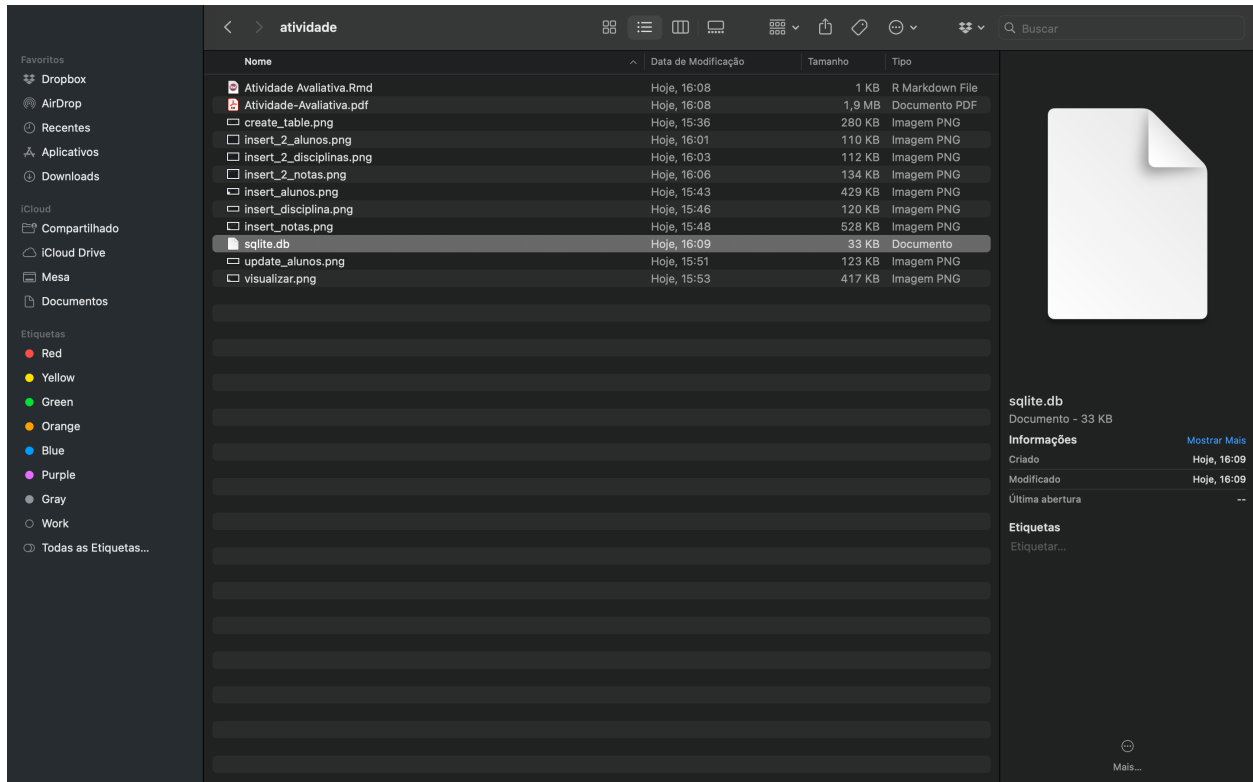
```
1 INSERT INTO Alunos (id_aluno, nome, idade) VALUES (4, 'José', 25);
2 INSERT INTO Alunos (id_aluno, nome, idade) VALUES (5, 'Paula', 19);
3 INSERT INTO Alunos (id_aluno, nome, idade) VALUES (6, 'Lira', 26);
4
5 SELECT * FROM Alunos;
```

id_aluno	nome
1	Novo Nome
2	Maria
3	Pedro
4	José
5	Paula
6	Lira



Questão 3. Na ferramenta SQLite, gere o BD na forma de um arquivo e armazene-o em uma pasta/diretório no seu computador.

Solução 3:



Questão 4. No R, faça a conexão com o BD usando o arquivo gerado no enunciado 3.

Solução 4:

```
library("RSQLite")
setwd("/Users/anamaria/especializacao/modulo_2/atividade/")
conexao <- RSQLite::dbConnect(RSQLite::SQLite(), dbname = "sqlite.db")
```

```
DBI::dbListTables(conexao)
```

```
## [1] "Alunos"      "Disciplinas" "Notas"
```

Questão 5. No R, construa três consultas SQL selecionando diretamente do BD linhas das tabelas utilizando a cláusula WHERE.

Solução 5:

```
-- Consulta na tabela Alunos
SELECT * FROM Alunos where nome ='Paula';
```

Table 1: 1 records

id_aluno	nome	idade
5	Paula	19

```
-- Consulta na tabela Disciplina
SELECT * FROM Disciplinas where nome ='Português';
```

Table 2: 2 records

id_disciplina	nome	professor
2	Português	Prof. Sousa
3	Português	Prof. McGonagall

```
-- Consulta na tabela Notas
SELECT * FROM Notas where nota > 8;
```

Table 3: 2 records

id_nota	id_aluno	id_disciplina	nota
3	1	1	9
7	3	1	9

**Questão 6.** No R, faça a importação das tabelas para data frames.

**Solução 6:**

- Dataframe Alunos:

```
library("dplyr")
```

```
##
```

```
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
```

```
##
```

```
## filter, lag
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
## intersect, setdiff, setequal, union
```

```
library("tibble")
alunos_tbl <- dplyr::tbl(conexao,"Alunos")
alunos_df <- dplyr::collect(alunos_tbl)
```

```
alunos_df
```

```
## # A tibble: 6 x 3
##   id_aluno nome      idade
##   <int> <chr>    <int>
## 1      1 Novo Nome    20
## 2      2 Maria      22
## 3      3 Pedro      21
## 4      4 José       25
## 5      5 Paula      19
## 6      6 Lira       26
```

- Dataframe Disciplinas:

```
disciplinas_tbl <- dplyr::tbl(conexao,"Disciplinas")
disciplinas_df <- dplyr::collect(disciplinas_tbl)
```

```
disciplinas_df
```

```
## # A tibble: 3 x 3
##   id_disciplina nome      professor
##   <int> <chr>    <chr>
## 1      1 Matemática Prof. Silva
## 2      2 Português Prof. Sousa
## 3      3 Português Prof. McGonagall
```

- Dataframe Notas:

```
notas_tbl <- dplyr::tbl(conexao,"Notas")
notas_df <- dplyr::collect(notas_tbl)
notas_df
```

```
## # A tibble: 13 x 4
##   id_nota id_aluno id_disciplina nota
##   <int>   <int>      <int> <int>
## 1      1      1          1      8
## 2      2      2          1      7
## 3      3      3          1      9
## 4      4      4          2      6
## 5      5      5          2      8
## 6      6      6          2      7
## 7      7      7          3      9
## 8      8      8          3      7
## 9      9      9          3      8
## 10     10     10          3      6
## 11     11     11          3      5
## 12     12     12          3      4
## 13     13     13          3      8
```

Questão 7. No R, faça consultas utilizando `select()` do pacote `dplyr` nos objetos `tibble` correspondentes aos data frames gerados no enunciado 6.

Solução 7:

```
alunos1 <- dplyr::sql("SELECT * FROM Alunos WHERE idade > 22")
alunos_select <- dplyr::tbl(conexao, alunos1)
alunos_db_select <- dplyr::collect(alunos_select)
alunos_db_select
```

```
## # A tibble: 2 x 3
##   id_aluno nome   idade
##   <int> <chr> <int>
## 1      4 José    25
## 2      6 Lira    26
```

```
disciplinas1 <- dplyr::sql("SELECT * FROM Disciplinas WHERE nome != 'Português'")
disciplinas_select <- dplyr::tbl(conexao, disciplinas1)
disciplinas_db_select <- dplyr::collect(disciplinas_select)
disciplinas_db_select
```

```
## # A tibble: 1 x 3
##   id_disciplina nome      professor
##   <int> <chr>      <chr>
## 1      1 Matemática Prof. Silva
```

```
notas1 <- dplyr::sql("SELECT * FROM Notas WHERE nota <= 6")
notas_select <- dplyr::tbl(conexao, notas1)
notas_db_select <- dplyr::collect(notas_select)
notas_db_select
```

```
## # A tibble: 4 x 4
##   id_nota id_aluno id_disciplina  nota
##   <int>   <int>      <int> <int>
## 1      4      2          1      6
## 2     10      3          2      6
## 3     11      3          2      5
## 4     12      3          3      4
```

Questão 8.(opcional) Realize o mesmo processo (enunciados 2 até 7) para o BD “Animais de uma Fazenda”, disponível em google drive (páginas 22 a 25, apostila de SQLite).

Solução 8:

Para questão de otimização do relatório, irei omitir os prints da criação do BD. O BD estará anexado no arquivo compactado que está anexado.

- Conectando no BD

```
conexao_animal <- RSQLite::dbConnect(RSQLite::SQLite(), dbname = "animal.db")
```

- Realizar consultas utilizando a cláusula *Where*

```
-- Consulta na tabela Animal
SELECT * FROM Animal where peso_desm <= 200;
```

Table 4: 3 records

id_animal	nome	mae	pai	data_nasc	data_desm	peso_nasc	peso_desm	faz_orig
1	Animal 1	Mãe 1	Pai 1	2022-01-01	2022-06-01	2.5	200	1
3	Animal 3	Mãe 3	Pai 3	2022-03-01	2022-08-01	2.8	190	2
4	Animal 4	Mãe 4	Pai 4	2022-04-01	2022-09-01	2.2	180	2

```
-- Consulta na tabela Fazenda
SELECT * FROM Fazenda where id_faz = 1;
```

Table 5: 1 records

id_faz	nome	proprietario	telefone
1	Fazenda 1	Proprietário 1	123456789

```
-- Consulta na tabela Vacina
SELECT * FROM Vacina where data_venc <= '2024-02-01';
```

Table 6: 2 records

id_vacina	nome	tipo	data_venc	fabricante
1	Vacina 1	Tipo 1	2024-01-01	Fabricante 1
2	Vacina 2	Tipo 2	2024-02-01	Fabricante 2

```
-- Consulta na tabela Vacinacao
SELECT * FROM Vacinacao where data_vacinacao <= '2022-09-15';
```

Table 7: 7 records

id_vacinacao	data_vacinacao	id_animal	id_vacina	nome_aplicador
1	2022-07-15	1	1	Aplicador 1
2	2022-07-20	1	2	Aplicador 2
3	2022-08-01	1	3	Aplicador 3
4	2022-08-15	2	1	Aplicador 1
5	2022-08-20	2	2	Aplicador 2
6	2022-09-01	2	3	Aplicador 3
7	2022-09-15	3	1	Aplicador 1

- Importação das tabelas para dataframes.

```
animal_tbl <- dplyr::tbl(conexao_animal,"Animal")
animal_df <- dplyr::collect(animal_tbl)
animal_df
```

```
## # A tibble: 5 x 9
##   id_animal nome      mae pai data_nasc data_desm peso_nasc peso_desm faz_orig
##   <int> <chr>    <chr> <chr> <chr> <chr> <dbl> <dbl> <int>
## 1      1 Animal~ Mãe 1 Pai 1 2022-01-~ 2022-06-~ 2.5 200 1
## 2      2 Animal~ Mãe 2 Pai 2 2022-02-~ 2022-07-~ 3 220 1
## 3      3 Animal~ Mãe 3 Pai 3 2022-03-~ 2022-08-~ 2.8 190 2
## 4      4 Animal~ Mãe 4 Pai 4 2022-04-~ 2022-09-~ 2.2 180 2
## 5      5 Animal~ Mãe 5 Pai 5 2022-05-~ 2022-10-~ 2.6 210 2
```

```
fazenda_tbl <- dplyr::tbl(conexao_animal,"Fazenda")
fazenda_df <- dplyr::collect(fazenda_tbl)
fazenda_df
```

```
## # A tibble: 2 x 4
##   id_faz nome      proprietario telefone
##   <int> <chr>    <chr> <chr>
## 1      1 Fazenda 1 Proprietário 1 123456789
## 2      2 Fazenda 2 Proprietário 2 987654321
```

```
vacina_tbl <- dplyr::tbl(conexao_animal,"Vacina")
vacina_df <- dplyr::collect(vacina_tbl)
vacina_df
```

```
## # A tibble: 3 x 5
##   id_vacina nome      tipo data_venc fabricante
##   <int> <chr>    <chr> <chr> <chr>
## 1      1 Vacina 1 Tipo 1 2024-01-01 Fabricante 1
## 2      2 Vacina 2 Tipo 2 2024-02-01 Fabricante 2
## 3      3 Vacina 3 Tipo 3 2024-03-01 Fabricante 3
```

```
vacinacao_tbl <- dplyr::tbl(conexao_animal,"Vacinacao")
vacinacao_df <- dplyr::collect(vacinacao_tbl)
vacinacao_df
```

```
## # A tibble: 15 x 5
##   id_vacinacao data_vacinacao id_animal id_vacina nome_aplicador
##   <int> <chr> <int> <int> <chr>
## 1      1 2022-07-15 1 1 Aplicador 1
## 2      2 2022-07-20 1 2 Aplicador 2
## 3      3 2022-08-01 1 3 Aplicador 3
## 4      4 2022-08-15 2 1 Aplicador 1
## 5      5 2022-08-20 2 2 Aplicador 2
## 6      6 2022-09-01 2 3 Aplicador 3
## 7      7 2022-09-15 3 1 Aplicador 1
## 8      8 2022-09-20 3 2 Aplicador 2
## 9      9 2022-10-01 3 3 Aplicador 3
## 10     10 2022-10-15 4 1 Aplicador 1
```

```
## 11      11 2022-10-20      4      2 Aplicador 2
## 12      12 2022-11-01      4      3 Aplicador 3
## 13      13 2022-11-15      5      1 Aplicador 1
## 14      14 2022-11-20      5      2 Aplicador 2
## 15      15 2022-12-01      5      3 Aplicador 3
```

- Faça consultas utilizando `select()` do pacote `dplyr` nos objetos `tibble` correspondentes aos data frames gerados no item anterior.

```
animais1 <- dplyr::sql("SELECT * FROM Animal WHERE data_nasc <= '2022-03-01'")
animais_select <- dplyr::tbl(conexao_animal, animais1)
animais_db_select <- dplyr::collect(animais_select)
animais_db_select
```

```
## # A tibble: 3 x 9
##   id_animal nome      mae pai  data_nasc data_desm peso_nasc peso_desm faz_orig
##   <int> <chr>    <chr> <chr> <chr>      <chr>      <dbl>      <dbl>    <int>
## 1         1 Animal~ Mãe 1 Pai 1 2022-01-- 2022-06--      2.5        200        1
## 2         2 Animal~ Mãe 2 Pai 2 2022-02-- 2022-07--      3          220        1
## 3         3 Animal~ Mãe 3 Pai 3 2022-03-- 2022-08--      2.8        190        2
```

```
vacina1 <- dplyr::sql("SELECT * FROM vacina WHERE id_vacina <= 2")
vacina_select <- dplyr::tbl(conexao_animal, vacina1)
vacina_db_select <- dplyr::collect(vacina_select)
vacina_db_select
```

```
## # A tibble: 2 x 5
##   id_vacina nome      tipo  data_venc fabricante
##   <int> <chr>    <chr> <chr>      <chr>
## 1         1 Vacina 1 Tipo 1 2024-01-01 Fabricante 1
## 2         2 Vacina 2 Tipo 2 2024-02-01 Fabricante 2
```

```
vacinacao1 <- dplyr::sql("SELECT * FROM vacinacao WHERE nome_aplicador = 'Aplicador 1'")
vacinacao_select <- dplyr::tbl(conexao_animal, vacinacao1)
vacinacao_db_select <- dplyr::collect(vacinacao_select)
vacinacao_db_select
```

```
## # A tibble: 5 x 5
##   id_vacinacao data_vacinacao id_animal id_vacina nome_aplicador
##   <int> <chr>      <int>      <int> <chr>
## 1         1 2022-07-15          1          1 Aplicador 1
## 2         4 2022-08-15          2          1 Aplicador 1
## 3         7 2022-09-15          3          1 Aplicador 1
## 4        10 2022-10-15          4          1 Aplicador 1
## 5        13 2022-11-15          5          1 Aplicador 1
```

**Questão 9.(opcional)** Realize o mesmo processo (enunciados 2 até 7) para o BD cujo modelo está disponível no slide 37 disponível em google drive

**Solução 9:**

A solução é igual a solução do item 8, mudando apenas as tabelas do BD e sendo necessário criar os scripts para gerar o bd similar ao que foi realizado na questão 10.

**Questão 10.** Considere o BD descrito no Link. Realize o mesmo processo (enunciados 2 até 7) para este BD. Além disso, execute as dez operações propostas (que estão nos exercícios de integridade) e discuta as restrições de integridade violadas por cada operação (se houver alguma, dada uma operação ela viola alguma coisa?), e as diferentes maneiras de lidar com essas restrições.

**Solução 10:**

- Passo 1: criar as tabelas do banco de dados. As tabelas, bem como o banco de dados, serão criados no sqllite mas para manter o arquivo organizado, iremos importar apenas o BD gerado no Sqlite e colocaremos os comandos usados no Sqlite para gerar o BD.

1. Criação das tabelas:

```
-- Tabela Funcionário
CREATE TABLE FUNCIONARIO (
    Cpf VARCHAR(11) PRIMARY KEY,
    Pnome VARCHAR(255),
    Minicial CHAR(1),
    Unome VARCHAR(255),
    Datanasc DATE,
    Endereco VARCHAR(255),
    Sexo CHAR(1),
    Salario DECIMAL(10, 2),
    Cpf_supervisor VARCHAR(11),
    Dnr INT,
    FOREIGN KEY (Cpf_supervisor) REFERENCES FUNCIONARIO(Cpf),
    FOREIGN KEY (Dnr) REFERENCES DEPARTAMENTO(Dnumero)
);
```

```
-- Tabela Departamento
CREATE TABLE DEPARTAMENTO (
    Dnumero INT PRIMARY KEY,
    Dnome VARCHAR(255),
    Cpf_gerente VARCHAR(11),
    Data_inicio_gerente DATE,
    FOREIGN KEY (Cpf_gerente) REFERENCES FUNCIONARIO(Cpf)
);
```

```
-- Tabela Localizações
CREATE TABLE LOCALIZACOES_DEP (
    Dnumero INT,
    Dlocal VARCHAR(255),
    PRIMARY KEY (Dnumero, Dlocal),
    FOREIGN KEY (Dnumero) REFERENCES DEPARTAMENTO(Dnumero)
);
```

```
-- Tabela Projeto
CREATE TABLE PROJETO (
    Projnumero INT PRIMARY KEY,
    Projnome VARCHAR(255),
    Projlocal VARCHAR(255),
```



```

Dnum INT,
FOREIGN KEY (Dnum) REFERENCES DEPARTAMENTO(Dnumero)
);

```

```

-- Tabela Trabalha
CREATE TABLE TRABALHA_EM (
    Fcpf VARCHAR(11),
    Pnr INT,
    Horas DECIMAL(5, 2),
    PRIMARY KEY (Fcpf, Pnr),
    FOREIGN KEY (Fcpf) REFERENCES FUNCIONARIO(Cpf),
    FOREIGN KEY (Pnr) REFERENCES PROJETO(Projnumero)
);

```

```

-- Tabela Dependente
CREATE TABLE DEPENDENTE (
    Fcpf VARCHAR(11),
    Nome_dependente VARCHAR(255),
    Sexo CHAR(1),
    Datanasc DATE,
    Parentesco VARCHAR(255),
    PRIMARY KEY (Fcpf, Nome_dependente),
    FOREIGN KEY (Fcpf) REFERENCES FUNCIONARIO(Cpf)
);

```

2. Inserção dos dados nas tabelas correspondentes:

```

-- Inserção de dados na tabela funcionário:
INSERT INTO FUNCIONARIO (Cpf, Pnome, Minicial, Unome, Datanasc, Endereco,
Sexo, Salario, Cpf_supervisor, Dnr) VALUES
('12345678966', 'João', 'B', 'Silva', '1965-01-09', 'Rua das Flores, 751, São Paulo, SP',
'M', 30000, '33344555587', 5),
('33344555587', 'Fernando', 'T', 'Wong', '1955-12-08', 'Rua da Lapa, 34, São Paulo, SP',
'M', 40000, '88866555576', 5),
('9988777767', 'Alice', 'J', 'Zelaya', '1968-01-19', 'Rua Souza Lima, 35, Curitiba, PR',
'F', 25000, '98765432168', 4),
('98765432168', 'Jennifer', 'S', 'Souza', '1941-06-20', 'Av. Arthur de Lima, 54, Santo André, SP',
'F', 43000, '88866555576', 4),
('66688444476', 'Ronaldo', 'K', 'Lima', '1962-09-15', 'Rua Rebouças, 65, Piracicaba, SP',
'M', 38000, '33344555587', 5),
('45345345376', 'Joice', 'A', 'Leite', '1972-07-31', 'Av. Lucas Obes, 74, São Paulo, SP',
'F', 25000, '33344555587', 5),
('98798798733', 'André', 'V', 'Pereira', '1969-03-29', 'Rua Timbira, 35, São Paulo, SP',
'M', 25000, '98765432168', 4),
('88866555576', 'Jorge', 'E', 'Brito', '1937-10-11', 'Rua do Horto, 35, São Paulo, SP',
'M', 55000, NULL, 1);

```

```

-- Inserção Tabela Departamento
INSERT INTO DEPARTAMENTO (Dnumero, Dnome, Cpf_gerente, Data_inicio_gerente) VALUES
(1, 'Matriz', '88866555576', '1988-06-19'),
(4, 'Administração', '98765432168', '1995-01-01'),
(5, 'Pesquisa', '33344555587', '1988-05-22');

```

```
-- Inserção na Tabela de Localizacao_dep
INSERT INTO LOCALIZACOES_DEP (Dnumero, Dlocal) VALUES
(1, 'São Paulo'),
(4, 'Mauá'),
(5, 'Santo André'),
(5, 'Itu'),
(5, 'São Paulo');
```

```
-- Inserção na Tabela TRABALHA_EM
INSERT INTO TRABALHA_EM (Fcpf, Pnr, Horas) VALUES
('12345678966', 1, 32.5),
('12345678966', 2, 7.5),
('66688444476', 3, 40.0),
('45345345376', 1, 20.0),
('45345345376', 2, 20.0),
('33344555587', 2, 10.0),
('33344555587', 3, 10.0),
('33344555587', 10, 10.0),
('33344555587', 20, 10.0),
('99988777767', 30, 30.0),
('99988777767', 10, 10.0),
('98798798733', 10, 35.0);
```

```
-- Inserção na Tabela Projeto
INSERT INTO PROJETO (Projnumero, Projnome, Projlocal, Dnum) VALUES
(1, 'ProdutoX', 'Santo André', 5),
(2, 'ProdutoY', 'Itu', 5),
(3, 'ProdutoZ', 'São Paulo', 5),
(10, 'Informatização', 'Mauá', 4),
(20, 'Reorganização', 'São Paulo', 1),
(30, 'Novosbenefícios', 'Mauá', 4);
```

```
-- Inserção na Tabela Dependente
INSERT INTO DEPENDENTE (Fcpf, Nome_dependente, Sexo, Datanasc, Parentesco) VALUES
('33344555587', 'Alicia', 'F', '1986-05-04', 'Filha'),
('33344555587', 'Tiago', 'M', '1983-10-25', 'Filho'),
('33344555587', 'Janaina', 'F', '1958-03-05', 'Esposa');
```

- Passo 2: exportar o Banco de Dados com as tabelas acima em um arquivo .db e realizar testes e consultas SQL.

1. Fazer a conexão do banco de dados usando o arquivo gerado através da exportação no SQLite.

```
conexao_restricao <- RSQLite::dbConnect(RSQLite::SQLite(), dbname = "restricao.db")
```

2. Construir três consultas SQL selecionando diretamente do BD linhas das tabelas utilizando a cláusula WHERE.

```
select Pnome, Datanasc, Endereco from FUNCIONARIO where datanasc >= '1965-01-01';
```

Table 8: 4 records

Pnome	Datanasc	Endereco
João	1965-01-09	Rua das Flores, 751, São Paulo, SP
Alice	1968-01-19	Rua Souza Lima, 35, Curitiba, PR
Joice	1972-07-31	Av. Lucas Obes, 74, São Paulo, SP
André	1969-03-29	Rua Timbira, 35, São Paulo, SP

```
select * from DEPENDENTE where Parentesco in ('Filho', 'Filha');
```

Table 9: 2 records

Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco
33344555587	Alicia	F	1986-05-04	Filha
33344555587	Tiago	M	1983-10-25	Filho

```
select Pnome, Minicial, Unome, Salario from FUNCIONARIO where salario > 30000;
```

Table 10: 4 records

Pnome	Minicial	Unome	Salario
Fernando	T	Wong	40000
Jennifer	S	Souza	43000
Ronaldo	K	Lima	38000
Jorge	E	Brito	55000

3. Fazer a importação das tabelas do BD criado para data frames.

```
departamento_tbl <- dplyr::tbl(conexao_restricao, "Departamento")
departamento_df <- dplyr::collect(departamento_tbl)
departamento_df
```

```
## # A tibble: 3 x 4
##   Dnumero Dnome      Cpf_gerente Data_inicio_gerente
##   <int> <chr>      <chr>      <chr>
## 1      1 Matriz      88866555576 1988-06-19
## 2      4 Administração 98765432168 1995-01-01
## 3      5 Pesquisa    33344555587 1988-05-22
```

```
dependente_tbl <- dplyr::tbl(conexao_restricao, "Dependente")
dependente_df <- dplyr::collect(dependente_tbl)
dependente_df
```

```
## # A tibble: 3 x 5
##   Fcpf      Nome_dependente Sexo  Datanasc  Parentesco
##   <chr>      <chr>      <chr> <chr>      <chr>
## 1 33344555587 Alicia      F     1986-05-04 Filha
## 2 33344555587 Tiago      M     1983-10-25 Filho
## 3 33344555587 Janaina    F     1958-03-05 Esposa
```

```
funcionario_tbl <- dplyr::tbl(conexao_restricao,"Funcionario")
funcionario_df <- dplyr::collect(funcionario_tbl)
funcionario_df
```

```
## # A tibble: 8 x 10
##   Cpf      Pnome Minicial Unome Datanasc Endereco Sexo  Salario Cpf_supervisor
##   <chr>    <chr> <chr>   <chr> <chr>   <chr>   <chr>   <int> <chr>
## 1 123456789~ João  B      Silva 1965-01~ Rua das~ M      30000 33344555587
## 2 333445555~ Fern~ T      Wong  1955-12~ Rua da ~ M      40000 88866555576
## 3 9988777767 Alice J      Zela~ 1968-01~ Rua Sou~ F      25000 98765432168
## 4 987654321~ Jenn~ S      Souza 1941-06~ Av. Art~ F      43000 88866555576
## 5 666884444~ Rona~ K      Lima  1962-09~ Rua Reb~ M      38000 33344555587
## 6 453453453~ Joice A      Leite 1972-07~ Av. Luc~ F      25000 33344555587
## 7 987987987~ André V      Pere~ 1969-03~ Rua Tim~ M      25000 98765432168
## 8 888665555~ Jorge E      Brito 1937-10~ Rua do ~ M      55000 <NA>
## # i 1 more variable: Dnr <int>
```

```
localizacao_tbl <- dplyr::tbl(conexao_restricao,"Localizacoes_Dep")
localizacao_df <- dplyr::collect(localizacao_tbl)
localizacao_df
```

```
## # A tibble: 5 x 2
##   Dnumero Dlocal
##   <int> <chr>
## 1      1 São Paulo
## 2      4 Mauá
## 3      5 Santo André
## 4      5 Itu
## 5      5 São Paulo
```

```
projeto_tbl <- dplyr::tbl(conexao_restricao,"Projeto")
projeto_df <- dplyr::collect(projeto_tbl)
projeto_df
```

```
## # A tibble: 6 x 4
##   Projnumero Projnome      Projlocal  Dnum
##   <int> <chr>      <chr>      <int>
## 1      1 ProdutoX      Santo André    5
## 2      2 ProdutoY      Itu            5
## 3      3 ProdutoZ      São Paulo      5
## 4     10 Informatização Mauá            4
## 5     20 Reorganização São Paulo      1
## 6     30 Novosbeneficios Mauá            4
```

```
trabalha_tbl <- dplyr::tbl(conexao_restricao,"Trabalha_EM")
trabalha_df <- dplyr::collect(trabalha_tbl)
trabalha_df
```

```
## # A tibble: 12 x 3
##   Fcpf      Pnr Horas
##   <chr>    <int> <dbl>
```

```
## 1 12345678966      1 32.5
## 2 12345678966      2  7.5
## 3 66688444476      3 40
## 4 45345345376      1 20
## 5 45345345376      2 20
## 6 33344555587      2 10
## 7 33344555587      3 10
## 8 33344555587     10 10
## 9 33344555587     20 10
## 10 99988777767     30 30
## 11 99988777767     10 10
## 12 98798798733     10 35
```

4. Faça consultas utilizando `select()` do pacote `dplyr` nos objetos tibble correspondentes aos data frames gerados no item anterior.

*Obs:* Com o objetivo de ser mais sucinta, realizei uma única consulta realacionando as tabelas do BD entre si.

```
sql_rest <- dplyr::sql("SELECT
  F.Cpf,
  F.Pnome,
  F.Unome,
  D.Dnome AS Departamento,
  P.Projnome AS Projeto,
  TE.Horas AS Horas_Trabalhadas,
  Dep.Nome_dependente AS Dependente,
  Dep.Parentesco
FROM
  FUNCIONARIO F
LEFT JOIN
  DEPARTAMENTO D ON F.Dnr = D.Dnumero
LEFT JOIN
  TRABALHA_EM TE ON F.Cpf = TE.Fcpf
LEFT JOIN
  PROJETO P ON TE.Pnr = P.Projnumero
LEFT JOIN
  DEPENDENTE Dep ON F.Cpf = Dep.Fcpf
")
restricao_select <- dplyr::tbl(conexao_restricao, sql_rest)
restricao_select_df <- dplyr::collect(restricao_select)
restricao_select_df
```

```
## # A tibble: 21 x 8
##   Cpf      Pnome    Unome Departamento Projeto  Horas_Trabalhadas Dependente
##   <chr>    <chr>    <chr> <chr>          <chr>          <dbl> <chr>
## 1 12345678966 João     Silva Pesquisa  ProdutoX         32.5 <NA>
## 2 12345678966 João     Silva Pesquisa  ProdutoY          7.5 <NA>
## 3 33344555587 Fernando Wong  Pesquisa  ProdutoY         10 Alicia
## 4 33344555587 Fernando Wong  Pesquisa  ProdutoY         10 Janaina
## 5 33344555587 Fernando Wong  Pesquisa  ProdutoY         10 Tiago
## 6 33344555587 Fernando Wong  Pesquisa  ProdutoZ         10 Alicia
## 7 33344555587 Fernando Wong  Pesquisa  ProdutoZ         10 Janaina
```

```
## 8 33344555587 Fernando Wong Pesquisa ProdutoZ 10 Tiago
## 9 33344555587 Fernando Wong Pesquisa Informa~ 10 Alicia
## 10 33344555587 Fernando Wong Pesquisa Informa~ 10 Janaina
## # i 11 more rows
## # i 1 more variable: Parentesco <chr>
```

- Passo 3: Executar as 10 operações propostas, discutir as restrições de integridade violadas por cada operação e as diferentes maneiras de lidar com essas restrições.

1. Inserir <'Roberto', 'F', 'Santos', '94377554355', '21-06-1972', 'Rua Benjamin, 34, Santo André, SP', M, 58.000, '88866555576', 1> em FUNCIONARIO.

```
Insert into FUNCIONARIO values
('Roberto', 'F', 'Santos', '94377554355', '21-06-1972', 'Rua Benjamin, 34, Santo André,
'SP', M, 58.000, '88866555576', 1);
```

### Discussão:

- i. Da maneira que os dados estão formatados, haverá problema com a correspondência das tabelas, isso é, inserir o dado correto na tabela correta. Para corrigir esse problema, devemos dizer a sequência correta das tabelas nas quais os dados serão inseridos.
- ii. Observamos também que o formato da data está diferente do formato da data que está na tabela FUNCIONARIO, sendo necessário realizar o ajuste da data para o mesmo formato da tabela. Será necessário ajustar também o salário para a formatação correta.
- iii. Com a discussão acima, a inserção correta, sem violar as restrições seria:

```
INSERT INTO FUNCIONARIO
(Pnome, Minicial, Unome, Cpf, Datanasc, Endereco, Sexo, Salario, Cpf_supervisor, Dnr)
VALUES ('Roberto', 'F', 'Santos', '94377554355', '1972-06-21',
'Rua Benjamin, 34, Santo André, SP', 'M', 58000, '88866555576', 1);
```

2. Inserir <'ProdutoA', 4, 'Santo André', 2> em PROJETO.

```
INSERT INTO PROJETO VALUES ('ProdutoA', 4, 'Santo André', 2);
```

### Discussão:

- i. Da maneira que os dados estão formatados, haverá problema com a correspondência das tabelas, isso é, inserir o dado correto na tabela correta. Para corrigir esse problema, devemos dizer a sequência correta das tabelas nas quais os dados serão inseridos.
- ii. Inserção ajustada:

```
INSERT INTO PROJETO (projnome, projnumero, projlocal, dnum) VALUES
('ProdutoA', 4, 'Santo André', 2);
```

3. Inserir <'Producao', 4, '94377554355', '01-10-2007'> em DEPARTAMENTO.

```
INSERT INTO DEPARTAMENTO VALUES ('Producao', 4, '94377554355', '01-10-2007');
```

#### Discussão:

- i. Da maneira que os dados estão formatados, haverá problema com a correspondência das tabelas, isso é, inserir o dado correto na tabela correta. Para corrigir esse problema, devemos dizer a sequência correta das tabelas nas quais os dados serão inseridos ou alterar a ordem dos dados a serem inseridos para que corresponda a ordem das colunas das tabelas.
- ii. Nessa inserção temos violação de chave primária uma vez que Dnumero é a chave primária da tabela e o valor 4 já está sendo usado, isso é, atribuído a outro departamento. Para corrigir esse problema deveremos mudar o valor da chave primária que estamos inserindo.
- iii. Formatação da data na inserção não corresponde a formatação da data na tabela DEPARTAMENTO sendo necessário realizar a correção.
- iv. Inserção corrigida:

```
INSERT INTO DEPARTAMENTO values (2, 'Producao', '94377554355', '2007-10-01');
```

4. Inserir <'67767898944', NULL, '40,0'> em TRABALHA\_EM.

```
INSERT INTO TRABALHA_EM VALUES ('67767898944', NULL, '40,0');
```

#### Discussão

- i. Temos violação de chave secundária, pois ela não pode ser nula. Para corrigir, inserir um valor.
- ii. A formatação das horas estão incoerentes com a formatação da coluna da tabela.
- iii. Correção:

```
INSERT INTO TRABALHA_EM VALUES ('67767898944', 4, 40.0);
```

5. Inserir <'45345345376', 'João', 'M', '12-12-1990', 'marido'> em DEPENDENTE.

```
INSERT INTO DEPENDENTE VALUES ('45345345376', 'João', 'M', '12-12-1990', 'marido');
```

#### Discussão

- i. Da maneira que os dados estão formatados, haverá problema com a correspondência das tabelas, isso é, inserir o dado correto na tabela correta. Para corrigir esse problema, devemos dizer a sequência correta das tabelas nas quais os dados serão inseridos ou alterar a ordem dos dados a serem inseridos para que corresponda a ordem das colunas das tabelas.
- ii. A formatação da data está incoerente com a formatação da coluna correspondente na tabela.
- iii. Correção:

```
INSERT INTO DEPENDENTE VALUES ('45345345376', 'João', 'M', '1990-12-12', 'Marido');
```

6. Excluir as linhas de TRABALHA\_EM com Fcpf = '33344555587'.

```
DELETE FROM TRABALHA_EM where Fcpf = '33344555587';
```

#### Discussão

- i. Nesse caso, não há problemas pois da maneira que criamos a tabela TRABALHA\_EM, Fcpf é uma chave estrangeira exportada da tabela FUNCIONARIO. Haveria problemas se Fcpf fosse uma chave primária que é referenciada em outras tabelas.
7. Excluir a linha de FUNCIONARIO com Cpf = '98765432168'.

```
DELETE FROM FUNCIONARIO WHERE Cpf = '98765432168';
```

#### Discussão

- i. Como Cpf é a chave primária da tabela FUNCIONARIO, haverá problemas ao excluir os registros se, e somente se, esse registro for usado em alguma das outras tabelas como chave secundária.
8. Excluir a linha de PROJETO com Projnome = 'ProdutoX'.

```
DELETE FROM PROJETO WHERE Projnome = 'ProdutoX';
```

#### Discussão

- i. Nesse caso, não há problemas pois da maneira que criamos a tabela PROJETO, Projnome não é chave primária.
9. Modificar Cpf\_gerente e Data\_inicio\_gerente da linha DEPARTAMENTO com Dnumero = 5 para '12345678966' e '01-10-2007', respectivamente. Modificar o atributo Cpf\_supervisor da linha FUNCIONARIO com Cpf = '99988777767' para '94377554355'.

```
UPDATE DEPARTAMENTO
SET Cpf_gerente = '12345678966', Data_inicio_gerente = '2007-10-01'
WHERE Dnumero = 5;

UPDATE FUNCIONARIO
SET Cpf_supervisor = '94377554355'
WHERE Cpf = '99988777767';
```

#### Discussão:

- i. Haveria problema em realizar essas modificações se o CPF\_gerente ou CPF\_supervisor informados já estivessem atribuídos a outra pessoa na tabela de FUNCIONARIOS, uma vez que CPF é a chave primária desta tabela e chave secundária nas demais tabelas.
10. Modificar o atributo Horas da linha TRABALHA\_EM com Fcpf = '99988777767' e Pnr = 10 para '5,0'.



```
UPDATE TRABALHA_EM  
SET Horas = 5.0  
WHERE Fcpf = '99988777767' AND Pnr = 10;
```

#### Discussão:

- i. Não há restrição sendo violada ao realizar essa atualização na tabela.

**Questão 11.** Considere os data sets imdb disponíveis google drive. Quais as diferenças entre imdb.csv, imdb.sqlite e imdb.rds?

#### Solução 11:

Os três formatos informados são maneiras diferentes de armazenar dados relacionados ao IMDb.

O formato **csv** (Comma-Separated Values) armazena os dados de maneira tabular, onde cada linha do arquivo é um registro de dados, e cada campo é separado por uma vírgula ou outro delimitador especificado mas não suporta relações entre tabelas além de não ser eficiente para grandes volumes de dados. É comumente usado para armazenar dataframes que são uma estrutura de dados que organiza os dados em uma tabela bidimensional de linhas e colunas, como uma planilha.

O formato **sqlite** é um arquivo de banco de dados SQLite. Diferente do formato csv, o sqlite é um sistema de gerenciamento de banco de dados relacional contido em uma biblioteca de programação C. Um arquivo .sqlite contém o banco de dados inteiro, isto é, inclui todas as informações daquele banco de dados como tabelas, índices, triggers e outras informações. Além disso, o sqlite suporta relações complexas entre dados sendo uma solução portátil que não necessita de um servidor de banco de dados separado.

O formato **rds** (R Data File) é um arquivo específico do R sendo utilizado para armazenar objetos de dados R, como dataframes, listas, modelos estatísticos, etc preservando a estrutura dos dados e metadados. Este formato permite que os usuários de R salvem e carreguem objetos em ou para o ambiente R de maneira eficiente mas só possui compatibilidade com o R, diferentemente dos outros dois formatos mencionados que possui compatibilidade com diversas ferramentas.

**Questão 12.** Qual dos 3 imdb (csv, sqlite, rds), você consegue abrir e manipular na ferramenta SQLite IDE? Por quê?

#### Solução 12:

Apenas o arquivo no formato rds não é possível abrir e manipular na ferramenta SQLite IDE pois arquivos nesse formato são compatíveis apenas com o R, conforme mencionado na questão anterior.

Como o SQLite IDE é projetado especificamente para interagir com bancos de dados SQLite, permitindo visualizar, editar, e gerenciar dados armazenados em arquivos .sqlite, o arquivo .sqlite permite abrir e manipular nessa ferramenta.

Embora seja possível importar e manipular arquivos .csv no SQLite IDE vale resaltar que neste caso teremos uma tabela de um banco de dados e não o banco de dados completo.

**Questão 13.** Quais dos 3 imdb (csv, sqlite, rds), você consegue importar para o R e trabalhar como data frames ou mesmo tibbles? Escreva os scripts necessários para realizar a importação de cada um deles. Mostre ainda instruções de manipulação dos data frames e tibbles resultantes da importação.

**Solução 13:**

No ambiente R, podemos importar e manipular dados em qualquer um dos três formatos imdb.csv, imdb.sqlite, e imdb.rds. No entanto, cada formato necessita de uma abordagem diferente para importação e manipulação subsequente como dataframes ou tibbles.

- Abordagem para csv:

```
library(readr)
imdb_csv <- read_csv("/Users/anamaria/especializacao/modulo_2/atividade/imdb.csv")

## Rows: 28490 Columns: 20
## -- Column specification -----
## Delimiter: ","
## chr (11): id_filme, titulo, data_lancamento, generos, pais, idioma, direcao,...
## dbl (9): ano, duracao, orcamento, receita, receita_eua, nota_imdb, num_aval...
##
## i Use 'spec()' to retrieve the full column specification for this data.
## i Specify the column types or set 'show_col_types = FALSE' to quiet this message.
```

```
imdb_csv[2,3]
```

```
## # A tibble: 1 x 1
##   ano
##   <dbl>
## 1  1945
```

- Abordagem para SQLite:

```
conexao_sqlite <- dbConnect(RSQLite::SQLite(),
  dbname = "/Users/anamaria/especializacao/modulo_2/atividade/imdb.sqlite")

imdb_sqlite <- dbReadTable(conexao_sqlite, "imdb")
```

```
SELECT titulo, duracao, diretor, ano FROM imdb where duracao > 190;
```

Table 11: Displaying records 1 - 10

titulo	duracao	diretor	ano
Titanic	194	James Cameron	1997
Iron Man 3	195	Shane Black	2013
Troy	196	Wolfgang Petersen	2004
Watchmen	215	Zack Snyder	2009
Kingdom of Heaven	194	Ridley Scott	2005

titulo	duracao	diretor	ano
The Wolf of Wall Street	240	Martin Scorsese	2013
Gangs of New York	216	Martin Scorsese	2002
The Lord of the Rings: The Return of the King	192	Peter Jackson	2003
Wyatt Earp	212	Lawrence Kasdan	1994
Gods and Generals	280	Ron Maxwell	2003

- Abordagem para rds:

```
imdb_rds <- readRDS("/Users/anamaria/especializacao/modulo_2/atividade/imdb.rds")
names(imdb_rds)
```

```
## [1] "id_filme"          "titulo"            "ano"
## [4] "data_lancamento"  "generos"           "duracao"
## [7] "pais"             "idioma"            "orcamento"
## [10] "receita"          "receita_eua"       "nota_imdb"
## [13] "num_avaliacoes"   "direcao"           "roteiro"
## [16] "producao"         "elenco"            "descricao"
## [19] "num_criticas_publico" "num_criticas_critica"
```

**Questão 14.** Considere o data set Melanoma.xlsx disponível google drive a) Faça a importação do data set para o R; b) realize operações de manipulação do data frame e do tibble; e c) faça consultas usando SQL.

**Solução 14:**

- Parte a: importação dos dados

```
library(readxl)
melanoma_data <- read_excel("/Users/anamaria/especializacao/modulo_2/atividade/Melanoma.xlsx")
head(melanoma_data)
```

```
## # A tibble: 6 x 7
##   time status sex age year thickness ulcer
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 10      3      1 76 1972      6.76      1
## 2 30      3      1 56 1968      0.65      0
## 3 35      2      1 41 1977      1.34      0
## 4 99      3      0 71 1968      2.9       0
## 5 185     1      1 52 1965     12.1      1
## 6 204     1      1 28 1971      4.84      1
```

- Parte b: operações de manipulação

```
library("dplyr")
melanoma_tibble <- as_tibble(melanoma_data)
melanoma_selected <- select(melanoma_tibble, sex, age)
melanoma_selected
```

```
## # A tibble: 205 x 2
##   sex age
##   <dbl> <dbl>
## 1     1  76
## 2     1  56
## 3     1  41
## 4     0  71
## 5     1  52
## 6     1  28
## 7     1  77
## 8     0  60
## 9     1  49
## 10    0  68
## # i 195 more rows
```

```
melanoma_filtered <- filter(melanoma_tibble, age >= 68)
melanoma_filtered
```

```
## # A tibble: 41 x 7
##   time status sex age year thickness ulcer
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1    10     3     1  76 1972     6.76     1
## 2    99     3     0  71 1968     2.9      0
## 3   210     1     1  77 1972     5.16     1
## 4   279     1     0  68 1971     7.41     1
## 5   386     1     0  68 1965     3.87     1
## 6   493     3     1  72 1971    12.6      1
## 7   621     1     1  72 1972     7.06     1
## 8   629     1     1  95 1968     5.48     1
## 9   667     1     0  89 1968    13.8      1
## 10  793     1     1  68 1970     4.84     1
## # i 31 more rows
```

```
melanoma_arranged <- arrange(melanoma_tibble, age)
melanoma_arranged
```

```
## # A tibble: 205 x 7
##   time status sex age year thickness ulcer
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  3385     2     0   4 1968     2.74     0
## 2  3776     2     1  12 1967     7.09     1
## 3   469     1     0  14 1969     2.42     1
## 4  1710     2     1  15 1973     0.58     0
## 5   858     1     0  16 1967     3.56     0
## 6  1062     1     1  19 1966     3.87     1
## 7  4479     2     0  19 1965     1.13     1
## 8  1654     2     1  20 1973     0.97     0
## 9  3441     2     0  20 1968     0.65     0
## 10  3776     2     0  21 1967     1.29     1
## # i 195 more rows
```

- Parte c: Consultas usando SQL

```
library(sqldf)
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
## Warning in doTryCatch(return(expr), name, parentenv, handler): unable to load shared object '/Library/Frameworks/R.framework/Resources/modules/R_X11.so, 0x0006): Library not loaded: /  
##   dlopen(/Library/Frameworks/R.framework/Resources/modules/R_X11.so, 0x0006): Library not loaded: /  
##   Referenced from: <9A3F5E83-2A35-33C3-9C5A-5255B116A1BE> /Library/Frameworks/R.framework/Versions/4  
##   Reason: tried: '/opt/X11/lib/libSM.6.dylib' (no such file), '/System/Volumes/Preboot/Cryptexes/OS/
```

```
## tcltk DLL is linked to '/opt/X11/lib/libX11.6.dylib'
```

```
## Could not load tcltk. Will use slower R code instead.
```

```
result_sql <- sqldf("SELECT * FROM melanoma_tibble WHERE age > 85 AND status = 3")  
head(result_sql)
```

```
##   time status sex age year thickness ulcer  
## 1   826      3  0  86 1965      8.54     1
```

Questão 15 (opcional). Considere o data set imdb.csv disponível google drive.

a) Faça a importação do data set para o R; b) realize operações de manipulação do data frame e do tibble; e c) faça consultas usando SQL.

Solução 15:

- Parte a: observe que na questão 12 esse data set já foi importado como imdb\_csv. Portanto, não farei uma nova importação.
- Parte b: operações de manipulação

```
imdb_subset <- imdb_csv[imdb_csv$ano > 2019, 2]  
imdb_subset
```

```
## # A tibble: 277 x 1  
##   titulo  
##   <chr>  
## 1 Shirley  
## 2 Birds of Prey: And the Fantabulous Emancipation of One Harley Quinn  
## 3 Cry Havoc  
## 4 The Orchard  
## 5 Equal Standard  
## 6 A Fall from Grace  
## 7 Minyan  
## 8 Expectant  
## 9 A Mother Knows Worst  
## 10 Carrion  
## # i 267 more rows
```

```
library("dplyr")
imdb_tibble <- as_tibble(imdb_csv)
imdb_tibble_filter <- filter(imdb_tibble, generos == 'Drama')
imdb_tibble_filter
```

```
## # A tibble: 2,565 x 20
##   id_filme  titulo    ano data_lancamento generos duracao pais  idioma orcamento
##   <chr>    <chr>  <dbl> <chr>          <chr>    <dbl> <chr> <chr>    <dbl>
## 1 tt0216204 The S~  1999 2000-03-01    Drama      83 USA  Engli~   400000
## 2 tt0049571 On th~  1956 1956-04-30    Drama      98 USA  Engli~  1505000
## 3 tt2822280 Confe~  2015 2015-03-24    Drama      90 USA  Engli~    NA
## 4 tt3703836 Henry~  2015 2016-01-08    Drama      87 USA  Engli~    NA
## 5 tt1142798 The F~  2008 2008-09-12    Drama     111 USA  Engli~    NA
## 6 tt0395561 State~  2005 2014-04-23    Drama     128 USA  Engli~   800000
## 7 tt0468458 Broth~  2006 2006-04-29    Drama      91 USA  Engli~  1000000
## 8 tt0109514 Curse~  1994 1995-05-05    Drama     102 USA  Engli~ 12577385
## 9 tt2099788 Welco~  2012 2013-03-01    Drama      81 USA  Engli~    NA
## 10 tt1996346 Woman~  2012 2012-04-13    Drama     101 USA  Engli~    NA
## # i 2,555 more rows
## # i 11 more variables: receita <dbl>, receita_eua <dbl>, nota_imdb <dbl>,
## #   num_avaliacoes <dbl>, direcao <chr>, roteiro <chr>, producao <chr>,
## #   elenco <chr>, descricao <chr>, num_criticas_publico <dbl>,
## #   num_criticas_critica <dbl>
```

- Parte c: Consultas usando SQL

```
library(sqldf)
result_imdb_sql <- sqldf("SELECT titulo, ano, generos, idioma from imdb_tibble
WHERE idioma != 'English' and generos == 'Comedy' and ano > 2015")
result_imdb_sql
```

```
##           titulo  ano generos          idioma
## 1      Banana Split 2018  Comedy      English, German
## 2      Booksmart 2019  Comedy      English, Mandarin, Spanish
## 3    No manches Frida 2019  Comedy      Spanish
## 4    An American Pickle 2020  Comedy      English, Hebrew
## 5      Copycat 2016  Comedy      English, Spanish
## 6    Daddy's Home 2 2017  Comedy      English, Spanish
## 7  Make America White Again 2020  Comedy      English, Mandarin, Russian, Spanish
## 8    3 Weeks in Yerevan 2016  Comedy      Armenian
## 9      Drunk Parents 2019  Comedy      English, Spanish, German
## 10  Office Christmas Party 2016  Comedy      English, Russian
## 11      Night School 2018  Comedy      English, Spanish, French, Arabic
```

Questão 16(opcional). Considere o data set cancer\_cerebro.csv. a) Faça a importação do data set para o R; b) realize operações de manipulação do data frame e do tibble; e c) faça consultas usando SQL.

Solução 16:

- Parte a: importação dos dados

```
library(readr)
cancer_cerebro <- read_csv("/Users/anamaria/especializacao/modulo_2/atividade/cancer_cerebro.csv")
```

```
## Rows: 2805 Columns: 11
## -- Column specification -----
## Delimiter: ","
## chr (3): TRATHOSP, FAIXAETAR, TOPOGRUP
## dbl (8): ID, ESCOLARI, SEXO, CATEATEND, ANODIAG, DIAGTRAT, TEMPO, CENSURA
##
## i Use 'spec()' to retrieve the full column specification for this data.
## i Specify the column types or set 'show_col_types = FALSE' to quiet this message.
```

```
colnames(cancer_cerebro)
```

```
## [1] "ID"          "ESCOLARI"    "SEXO"        "CATEATEND"   "ANODIAG"     "TRATHOSP"
## [7] "DIAGTRAT"    "FAIXAETAR"   "TEMPO"       "CENSURA"    "TOPOGRUP"
```

- Parte b: operações e manipulações

```
cancer_subset <- cancer_cerebro[cancer_cerebro$FAIXAETAR == '30-39', ]
cancer_subset
```

```
## # A tibble: 291 x 11
##       ID ESCOLARI SEXO CATEATEND ANODIAG TRATHOSP DIAGTRAT FAIXAETAR TEMPO
##   <dbl>   <dbl> <dbl>   <dbl>   <dbl> <chr>      <dbl> <chr>      <dbl>
## 1     3       4     2       2    2011 G         67 30-39      7.07
## 2     8       5     1       9    2010 A          0 30-39      5.35
## 3     9       9     1       9    2010 A          0 30-39      5.78
## 4    11       5     2       9    2010 A          0 30-39      5.16
## 5    21       5     2       1    2014 B          6 30-39      1.32
## 6    23       5     2       1    2014 A          0 30-39      1.45
## 7    24       9     1       3    2014 A          0 30-39      1.25
## 8    37       9     1       1    2013 G          0 30-39      2.38
## 9    38       5     1       3    2014 F         16 30-39      1.01
## 10   44       5     1       1    2013 G          6 30-39      1.84
## # i 281 more rows
## # i 2 more variables: CENSURA <dbl>, TOPOGRUP <chr>
```

```
library("dplyr")
cerebro_tibble <- as_tibble(cancer_cerebro)
cerebro_tibble_filter <- filter(cerebro_tibble, ANODIAG == 2014 & TRATHOSP == 'B')
cerebro_tibble_filter
```

```
## # A tibble: 49 x 11
##       ID ESCOLARI SEXO CATEATEND ANODIAG TRATHOSP DIAGTRAT FAIXAETAR TEMPO
##   <dbl>   <dbl> <dbl>   <dbl>   <dbl> <chr>      <dbl> <chr>      <dbl>
## 1    21       5     2       1    2014 B          6 30-39      1.32
## 2    36       9     1       3    2014 B         15 20-29      1.70
## 3    56       4     1       1    2014 B          7 10-19      0.986
## 4   219       2     2       2    2014 B         13 00-09      0.592
```

```
## 5 222 2 1 1 2014 B 19 00-09 0.773
## 6 335 1 1 3 2014 B 33 00-09 0.195
## 7 613 2 2 2 2014 B 116 50-59 0.321
## 8 743 9 2 2 2014 B 164 60-69 5.21
## 9 790 1 1 2 2014 B 66 60-69 0.501
## 10 792 1 2 2 2014 B 90 50-59 0.899
## # i 39 more rows
## # i 2 more variables: CENSURA <dbl>, TOPOGRUP <chr>
```

- Parte c: Consultas usando SQL

```
library(sqldf)
result_cerebro_sql <- sqldf("SELECT CATEATEND, ANODIAG, TRATHOSP, DIAGTRAT, FAIXAETAR, TEMPO
  from cerebro_tibble WHERE ANODIAG == 2010 and FAIXAETAR == '20-29' and DIAGTRAT != 0")
result_cerebro_sql
```

```
## CATEATEND ANODIAG TRATHOSP DIAGTRAT FAIXAETAR TEMPO
## 1 2 2010 G 163 20-29 2.5972603
## 2 9 2010 A 77 20-29 0.8876712
## 3 9 2010 G 154 20-29 1.1287671
## 4 9 2010 B 116 20-29 0.3863014
## 5 9 2010 E 1 20-29 0.9780822
## 6 9 2010 F 49 20-29 4.0684932
## 7 9 2010 D 39 20-29 1.8164384
## 8 9 2010 F 31 20-29 2.1232877
## 9 9 2010 F 20 20-29 1.5424658
## 10 2 2010 G 101 20-29 7.4410959
## 11 2 2010 F 57 20-29 7.2876712
## 12 9 2010 F 69 20-29 2.0054795
```