SQL

Para resolver os exercícios, utilize a descrição do modelo relacional abaixo e os comandos SQL aprendidos. Se houver a necessidade de utilizar mais de uma tabela em uma consulta, utilize o INNER JOIN, em vez do produto cartesiano.

novela (codigo_novela: inteiro, nome_novela: caracter(30), data_primeiro_capitulo: date, data_ultimo_capitulo: date, horario_exibicao: time)
novelaPersonagem (codigo_novela: inteiro, cod_personagem: inteiro) cod_novela **referencia** novela cod_personagem **referencia** personagem
personagem (cod_personagem: inteiro, nome_companhia: caracter(50), idade_personagem: inteiro, situacao_financeira_personagem: caracter(20), codigo_ator: inteiro) cod_ator **referencia** ator
ator (codigo_ator: inteiro, nome_ator: caracter(20), idade: inteiro, cidade_ator: caracter(20), salario_ator: real, sexo_ator: caracter(1))
capitulos (codigo_capitulo: inteiro, nome_capitulo: caracter(50), data_exibicao_capitulo: date, codigo_novela: inteiro) codigo_novela **referencia** novela

Comandos para criação da tabela:

```
CREATE TABLE tbNovela
(codigo_novela INT,
nome_novela VARCHAR(30) NOT NULL,
data_primeiro_capitulo DATE,
data_ultimo_capitulo DATE,
horario_exibicao TIME,
CONSTRAINT pk_tbNovela PRIMARY KEY (codigo_novela)
);

CREATE TABLE tbAtor
(codigo_ator INT,
nome_ator VARCHAR(20) NOT NULL,
idade INT,
cidade_ator VARCHAR(20),
salario_ator FLOAT,
sexo_ator CHAR(1),
CONSTRAINT pk_tbAtor PRIMARY KEY (codigo_ator)
);

CREATE TABLE tbCapitulo
(codigo_capitulo INT,
nome_capitulo VARCHAR(50) NOT NULL,
data_exibicao_capitulo DATE,
```

```
codigo_novela INT,
CONSTRAINT pk_tbCapitulo PRIMARY KEY (codigo_capitulo),
CONSTRAINT fk_tbCapitulo_tbNovela FOREIGN KEY (codigo_novela)
        REFERENCES tbNovela (codigo_novela) ON DELETE CASCADE
        ON UPDATE CASCADE
);


CREATE TABLE tbPersonagem
(codigo_personagem INT,
nome_personagem VARCHAR(50) NOT NULL,
idade_personagem INT,
situacao_fnanceira_personagem VARCHAR(20),
codigo_ator INT,
CONSTRAINT pk_tbPersonagem PRIMARY KEY (codigo_personagem),
CONSTRAINT fk_tbPersonagem_tbAtor FOREIGN KEY (codigo_ator)
        REFERENCES tbAtor (codigo_ator) ON DELETE CASCADE ON UPDATE CASCADE
);


CREATE TABLE tbNovelaPersonagem
(codigo_personagem INT,
codigo_novela INT,
CONSTRAINT pk_tbNovelaPersonagem PRIMARY KEY (codigo_personagem,codigo_novela),
CONSTRAINT fk_tbNovelaPersonagem_tbPersonagem FOREIGN KEY (codigo_personagem)
        REFERENCES tbPersonagem (codigo_personagem) ON DELETE CASCADE
        ON      UPDATE CASCADE,

CONSTRAINT fk_tbNovelaPersonagem_tbNovela FOREIGN KEY (codigo_novela)
        REFERENCES tbNovela (codigo_novela) ON DELETE CASCADE
        ON UPDATE  CASCADE
);
```

Questões:

1) Insira 3 registros para cada tabela criada.

2) Encontre todas as novelas que tenham o valor do horário de exibição vazio.

3) Selecione o nome de todos os atores que morem em cidades que comecem com a letra “M”.

4) Selecione todos os campos da tabela tbPersonagem ordenados por nome em ordem crescente.

5) Selecione quantos capítulos existem por novela, retorne o nome da novela e a quantidade de capítulos para a novela.

6) Encontre o nome de todas as novelas que tem mais de 40 capítulos.

Pandas

Nas questões seguintes, utilizar as seguintes bases de dados: titanic, iris

1) Verifique a presença de outliers (e.g., valor maior que 3 vezes o desvio padrão da base) na base de dados iris ou outra base de dados. Caso não exista, insira uma ou mais linhas e selecione elas utilizando a regra mencionada.

2) Faça um merge de dois dataframes e crie uma nova coluna para indicar a que dataframe pertence cada linha.

3) Selecione e mostre a quantidade de passageiros do titanic que possuem mais de 27 anos e que sobreviveram ao acidente.

4) Transforme os dados contínuos de idade do titanic em dados categóricos de acordo com a seguinte regra:

0 a 18 anos → Criança
18 a 65 anos → Adulto
maior de 65 → Idoso

5) Aplique o hold out utilizando diferentes porcentagens de divisão para separar a base iris em treino e teste (e.g. 60/40, 70/30, 80/20, 90/10). Crie vetores para armazenar os dados da divisão de treino e teste da seguinte forma:

X_train → armazena os dados de treino
X_test → armazena os dados de teste
y_train → armazena a label correspondente dos dados de treino
y_test → armazena a label correspondente dos dados de teste

Esse padrão é extremamente adotado em problemas supervisionados de machine learning.