Nome: Matheus Rocha Monteiro

Trainee Data Science - Lapisco
Módulo 1 - Atividade 1

INSTITUTO FEDERAL
Ceará

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.

Questão 1

Insira 3 registros para cada tabela criada.

Solução

```sql
INSERT INTO tbAtor
VALUES
(1,"Matheus Rocha",20,"Fortaleza",2000.0,"M"),
(2,"Roberto",40,"Fortaleza",15000.0,"M"),
(3,"Iagson",30,"Fortaleza",10000.0,"M");

INSERT INTO tbNovela
VALUES
(1,"Avenida Brasil",'2002-08-16',"2100-08-16",8),
(2,"Chocolate Com Pimenta",'2002-08-16',"2100-08-16",10),
(3,"O Rei do Gado",'2002-08-16',"2100-08-16",7);


INSERT INTO tbCapitulo
VALUES
(1,"Capitulo 1","2010-08-16",1),
(2,"Capitulo 1","2010-08-16",2),
(3,"Capitulo 1","2010-08-16",3);

INSERT INTO tbPersonagem
VALUES
(1,"Personagem A",15,"Rico",1),
(2,"Personagem B",20,"Rico Demais",2),
(3,"Personagem C",30,"Pobre",3);

INSERT INTO tbNovelaPersonagem
VALUES
(1,1),
(2,2),
(3,3);
```

Questão 2

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

Solução

```sql
SELECT *
FROM tbNovela t
WHERE t.horario_exibicao IS NULL;
```

Questão 3

Selecione o nome de todos os atores que morem em cidades que comecem com a letra "M".

Solução

```sql
SELECT *
FROM tbAtor t
WHERE t.cidade_ator LIKE "M%";
```

Questão 4

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

Solução

```sql
SELECT *
FROM tbPersonagem t
ORDER BY t.nome_personagem ASC;
```

Questão 5

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

Solução

```sql
SELECT n.nome_novela, COUNT(*) as qtd_capitulos
FROM tbNovela n
INNER JOIN tbCapitulo c
ON n.codigo_novela = c.codigo_novela
GROUP BY c.codigo_novela;
```

Questão 6

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

Solução

```sql
SELECT n.nome_novela
FROM tbNovela n
INNER JOIN tbCapitulo c
ON n.codigo_novela = c.codigo_novela
GROUP BY c.codigo_novela
HAVING COUNT(*) > 40;
```

Pandas

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

Questão 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.

Solução

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from IPython.display import display
warnings.filterwarnings("ignore")

iris = sns.load_dataset('iris')
iris.species = iris.species.astype('category')
iris.head()

cols = ['sepal_length','sepal_width','petal_length','petal_width']
for index, col in enumerate(cols):
    print(f"--- Outliers para a coluna: {col} ---")

    upper_limit = iris[col].mean() + np.abs(3 * iris[col].std())
    lower_limit = iris[col].mean() - np.abs(3 * iris[col].std())

    query = f"{col} > {upper_limit:.2f} | {col} < {lower_limit:.2f}"

    outliers = iris.query(query)

    # se nao encontrei outliers, crio eles
    if outliers.empty:
        print("\nOutliers nao encontrados, gerando outliers\n")

        #criando outlier
        new_row = {}
        for new_col in cols:
```

```
            new_row[new_col] = 0

        new_row[col] = upper_limit + 1.0
        new_row['species'] = 'setosa'

        #inserindo outlier no dataframe
        iris = iris.append(new_row, ignore_index = True)

        outliers = iris.query(query)

    display(outliers)
    print("\n\n\n")
```

Questão 2

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

Solução

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from IPython.display import display
warnings.filterwarnings("ignore")

df1 = pd.DataFrame({
    'a': [1,2,3],
    'b': [10,11,12]
})

df2 = pd.DataFrame({
    'a': [4,5,6],
    'b': [13,14,15]
})

df1['data_from'] = 'dataframe 1'
df2['data_from'] = 'dataframe 2'


df3 = pd.concat([df1,df2], ignore_index=True)
df3.species = df3.data_from.astype('category')

display(df3)
```

Resultado

É gerado um dataframe com os dados de ambos dataframes anteriores e também com um campo adicional, contendo a informação de onde tal informação veio:

Tabela 1: Dataset gerado através da concatenação dos datasets 1 e 2

	a	b	data_from
0	1	10	dataframe 1
1	2	11	dataframe 1
2	3	12	dataframe 1
3	4	13	dataframe 2
4	5	14	dataframe 2
5	6	15	dataframe 2

Questão 3

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

Solução

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from IPython.display import display
warnings.filterwarnings("ignore")

titanic = sns.load_dataset('titanic')
titanic.survived = titanic.survived.astype('boolean')
display(titanic.head())

q3 = titanic.query('age > 27 and survived == True')
display(q3.head())
```

Resultado

Tabela 2: 5 primeiros passageiros que possuem mais de 27 anos e sobreviveram ao acidente

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
1	True	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
3	True	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
11	True	1	female	58.0	0	0	26.5500	S	First	woman	False	C	Southampton	yes	True
15	True	2	female	55.0	0	0	16.0000	S	Second	woman	False	NaN	Southampton	yes	True
21	True	2	male	34.0	0	0	13.0000	S	Second	man	True	D	Southampton	yes	True

Questão 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

Solução

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from IPython.display import display
warnings.filterwarnings("ignore")

titanic = sns.load_dataset('titanic')
titanic.survived = titanic.survived.astype('boolean')
display(titanic.head())

def person_type(age: int) -> str:
    if age < 18:
        return 'Child'
    if age < 65:
        return 'Adult'
    return 'Old'

titanic['person_type'] = [person_type(age) for age in titanic.age]
sample = titanic.sample(n=10, random_state=1)

display(sample)
```

Resultado

Tabela 3: Amostra de 10 passageiros após classificação baseada em idade

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone	person_type
862	True	1	female	48.0	0	0	25.9292	S	First	woman	False	D	Southampton	yes	True	Adult
223	False	3	male	NaN	0	0	7.8958	S	Third	man	True	NaN	Southampton	no	True	Old
84	True	2	female	17.0	0	0	10.5000	S	Second	woman	False	NaN	Southampton	yes	True	Child
680	False	3	female	NaN	0	0	8.1375	Q	Third	woman	False	NaN	Queenstown	no	True	Old
535	True	2	female	7.0	0	2	26.2500	S	Second	child	False	NaN	Southampton	yes	False	Child
623	False	3	male	21.0	0	0	7.8542	S	Third	man	True	NaN	Southampton	no	True	Adult
148	False	2	male	36.5	0	2	26.0000	S	Second	man	True	F	Southampton	no	False	Adult
3	True	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False	Adult
34	False	1	male	28.0	1	0	82.1708	C	First	man	True	NaN	Cherbourg	no	False	Adult
241	True	3	female	NaN	1	0	15.5000	Q	Third	woman	False	NaN	Queenstown	yes	False	Old

Questão 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

Solução

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from IPython.display import display
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split

iris = sns.load_dataset('iris')
iris.species = iris.species.astype('category')
iris.info()

X = iris.copy()
X = X.drop(['species'], axis = 1)
y = iris.species.copy()

#usando 20% dos dados para teste e 80% para treino
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)
```