*Teste Prático – Desenvolvedor Back-End Pleno*

***Teste de SQL***

Dada a estrutura abaixo, gere os scripts em Postgres ou T-SQL:



**1°:** Crie o script de criação das tabelas com os respectivos relacionamentos.

**2)** Gere os scripts de inserção dos registros:

**Tabela**: Owner

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **Name** | | Adam Smith | | Scott Johnson | | Kimberly Parker | |
|  |

**Tabela:** Cats

|  |  |  |
| --- | --- | --- |
| **Name** | **Age** | **OwnerId** |
| Lily | 5 | 1 |
| Chloe | 2 | 3 |
| Charlie | 3 | 2 |

**Tabela:** Dogs

|  |  |  |
| --- | --- | --- |
| **Name** | **Age** | **OwnerId** |
| Maggie | 1 | 2 |
| Duke | 7 | 1 |
| Buddy | 4 | 2 |

3) Monte uma query que selecione apenas os nomes da tabela Cats que começem com 'c', e os nomes da tabela Dogs que terminem com 'e'.

Se possível retorne os dados em um único resultset.

4) Selecione o registro na tabela Dogs com menor Age

5) Selecione a soma da coluna Age da tabela Cats

6) Selecione os nomes dos Owner e seus respectivos Cats

7) Selecione os registros da tabela Dogs com maior Age separados por OwnerId

8) Retorne o nome do Owner e a respectiva quantidade de animais total

***Teste C# e ASP.NET Core***

Após a migração para o PostgreSQL, a empresa precisa de uma API para inserir, atualizar, consultar e apagar os registros do banco.

Criar uma Web API em .NET Core 3.1 que faça o CRUD do diagrama anterior para as tabelas Cats e Dogs. A tabela Owner será apenas para leitura.

Os dados podem ser salvos/armazendos em uma das 3 formas abaixo:

- Em memória;

- Cache Redis;

- Postgres;

Como Entregar o Teste Prático

Enviar todos os arquivos em um arquivo zip utilizando a separação abaixo, ou utilizando um repositório git público.

Pastas:

* Database: Esta pasta deve conter os scripts do teste de SQL.
* WebAPI: O fonte do projeto C#.