09

# O total de contratos válidos de cada cliente

Na aula anterior vimos como fazer o cruzamento de bases utilizando um *data step* com a opção *merge*, pois queríamos cruzar a base de clientes com a base de operações agrupada por CPF para obtermos quantos jogos cada cliente alugou.

Antes do cruzamento fizemos os seguintes tratamentos nas bases:

- Sumarizamos a base de operações pela variável CPF, construindo nisso a variável `Total_contratos` ,
- Criamos a variável `CPF_RAIZ` na base de clientes à partir da variável `CPF` , para que tenhamos o CPF raiz em ambas as bases.

O código desses dois tratamentos segue abaixo:

```
PROC SQL;
    create table contratos_cpf as
    select CPF, count(*) as Total_contratos
    from alura.operacoes_201709
    group by 1
;QUIT;

DATA clientes_cpf;
set alura.cadastro_cliente_v3;
CPF_RAIZ = input(substr(CPF,1,11),COMMAX11.0);
RUN;
```

Podemos cruzar estas duas bases usando um *PROC SQL* ao invés do *data step*, e para isso usamos alguns comandos diferentes na lógica SQL que já conhecemos:

- Da mesma forma que colocamos apelidos usando o "in=" no data merge, fazemos algo muito parecido aqui no SQL. Após declararmos a base, colocaremos que ela seja "referenciada como" um outro nome, mais curto, para simplificar o código. Fazemos isso com o comando `as` . Vamos chamar a base `clientes_cpf` de `A` e a base `contratos_cpf` de `B` .
- Como queremos fazer um cruzamento à esquerda, usamos o comando `left join` , seguido pela outra base que participará do cruzamento (a base de contratos).
- Precisamos declarar de qual base cada variável provém, mas podemos usar os apelidos que usamos para simplificar o código.
- Finalmente, diremos em quais chaves ele irá procurar a igualdade de cruzamento após o comando `on` . No caso queremos que a chave "cpf_raiz" da primeira base seja igual a chave "cpf" da segunda.

O código em SQL que faz este cruzamento segue abaixo:

```
PROC SQL;
    create table cadastro_cliente_jogos as
    select a.*, b.Total_contratos
    from clientes_cpf as A
    left join contratos_cpf as B
    on a.CPF_RAIZ = b.CPF
;QUIT;
```

Mas vimos também que podemos "pular" as bases intermediárias que criamos antes de fazer o cruzamento, colocando dentro do próprio SQL que faz o cruzamento, da seguinte forma:

- Partindo diretamente da base de clientes, fazemos o cruzamento usando não diretamente uma variável, mas a função que usamos para transformar o CPF em CPF raiz, o que torna possível comparar estas duas chaves de cruzamento.
- Ao invés de chamarmos a base `contratos_cpf` como nossa base `B`, colocamos entre parênteses o processo que cria essa base (pois ela é uma base que também foi criada com um processo em SQL). Isso muitas vezes é chamado de um "sub-processo" ou um "processo interno" ("subquery" ou "inner query", em inglês).

Todos os códigos que construímos anteriormente podem ser trocados pelo código abaixo:

```
PROC SQL;
    create table cadastro_cliente_jogos as
    select a.*, b.Total_contratos
    from alura.cadastro_cliente_v3 as A
    left join (
        select CPF, count(*) as Total_contratos
        from alura.operacoes_201709
        group by 1
        ) as B
    on input(substr(a.CPF,1,11),COMMAX11.0) = b.CPF
;QUIT;
```

Mas nem todos os jogos alugados deveriam ser considerados na ação promocional da *AluraPlay*. Precisamos de uma variável que nos diga quantos jogos que o cliente alugou que são válidos na promoção.

Usando o código SQL acima, construa a variável `Contratos_validos`, que contém a quantidade de contratos de aluguel de jogos de cada cliente que podem ser considerados válidos na promoção. Considere como "válido" os contratos em que não houve atraso (ou seja, que o cliente não ficou com o jogo por mais de 30 dias) e nem dano ao produto (ou seja, o custo de reparo é zero).