

Trabalho 1 BD2

Integrantes:

Maycon Douglas Batista dos Santos 11921BSI209

&

Matheus Costa Monteiro 12111BSI281

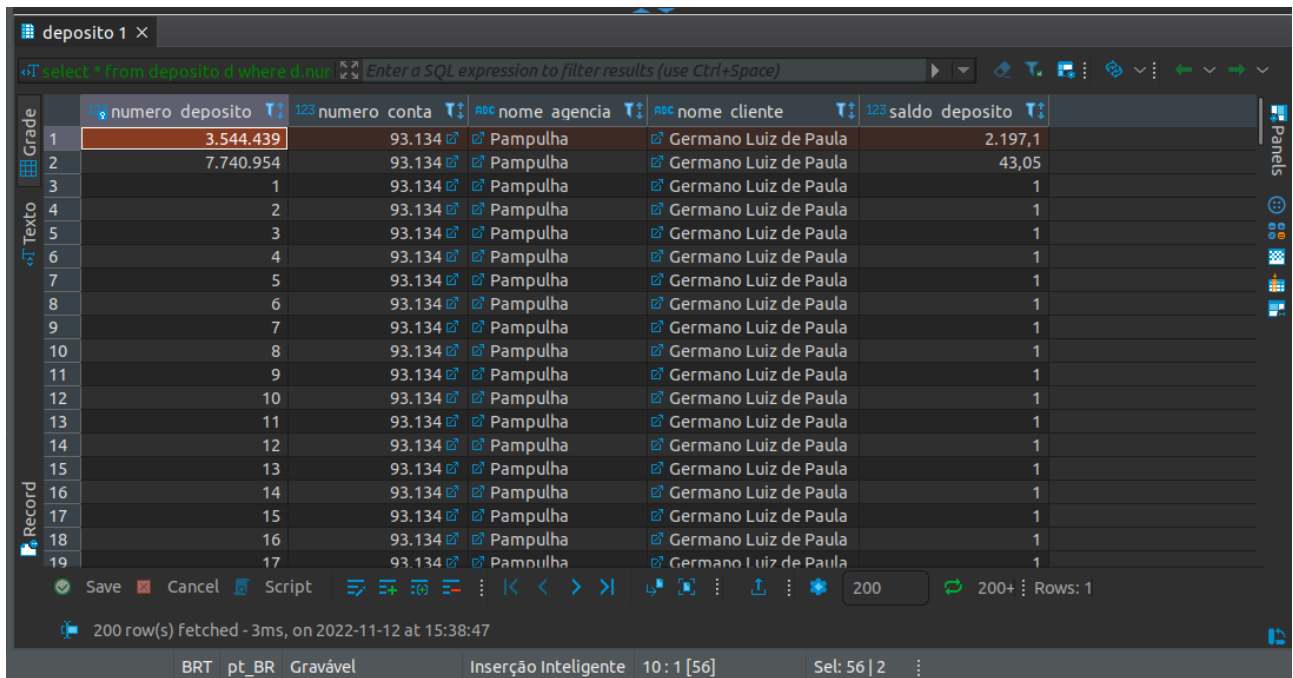
Instalação do Postgres, To join or not to Join e funções

Aviso: como foi discutido em aula e concordado em conjunto com o professor foi omitido a parte do relatório sobre a instalação e configuração do banco de dados postgresql.

9) Acrescente, por exemplo, 100.000 depósitos de R\$ 1,00 (Um Real) na conta do cliente 'Germano Luiz de Paula', na agência 'Pampulha', na conta 93134. Você deve executar o código abaixo em uma janela de comandos do PostgreSQL, quando o banco de dados selecionado para consultas for o nosso banco IB.

```
do
$do$
begin
    for num in 1..100000 loop
        insert into deposito (numero_deposito, numero_conta, nome_agencia, nome_cliente, saldo_deposito)
            values (num, 93134, 'Pampulha', 'Germano Luiz de Paula', 1.00);
    end loop;
end;
$do$;
```

Como pedido o resultado após rodar o código sql acima temos:



	numero deposito	numero conta	nome agencia	nome cliente	saldo deposito
1	3.544.439	93.134	Pampulha	Germano Luiz de Paula	2.197,1
2	7.740.954	93.134	Pampulha	Germano Luiz de Paula	43,05
3	1	93.134	Pampulha	Germano Luiz de Paula	1
4	2	93.134	Pampulha	Germano Luiz de Paula	1
5	3	93.134	Pampulha	Germano Luiz de Paula	1
6	4	93.134	Pampulha	Germano Luiz de Paula	1
7	5	93.134	Pampulha	Germano Luiz de Paula	1
8	6	93.134	Pampulha	Germano Luiz de Paula	1
9	7	93.134	Pampulha	Germano Luiz de Paula	1
10	8	93.134	Pampulha	Germano Luiz de Paula	1
11	9	93.134	Pampulha	Germano Luiz de Paula	1
12	10	93.134	Pampulha	Germano Luiz de Paula	1
13	11	93.134	Pampulha	Germano Luiz de Paula	1
14	12	93.134	Pampulha	Germano Luiz de Paula	1
15	13	93.134	Pampulha	Germano Luiz de Paula	1
16	14	93.134	Pampulha	Germano Luiz de Paula	1
17	15	93.134	Pampulha	Germano Luiz de Paula	1
18	16	93.134	Pampulha	Germano Luiz de Paula	1
19	17	93.134	Pampulha	Germano Luiz de Paula	1

11.1

Selecione os nomes dos clientes e seus respectivos números de conta e nome de agência que fizeram depósitos e empréstimos ao mesmo tempo.

Este código em SQL resolve esta consulta sem fazer uso da cláusula JOIN:

```
select e.nome_cliente, e.numero_conta, e.nome_agencia from emprestimo e
intersect
select d.nome_cliente, d.numero_conta, d.nome_agencia from deposito d;
```

Resultado após realizar a consulta acima!.

```
-- questão 11.1
/*
 * Selecione os nomes dos clientes e seus respectivos números de conta e nome de
 * agência que fizeram depósitos e empréstimos ao mesmo tempo.
 * Este código em SQL resolve esta consulta sem fazer uso da cláusula JOIN:
 */

select e.nome_cliente, e.numero_conta, e.nome_agencia from emprestimo e
intersect
select d.nome_cliente, d.numero_conta, d.nome_agencia from deposito d;
```

Results 1 x

select e.nome_cliente, e.numero_conta, e.nome_agencia

	nome_cliente	numero_conta	nome_agencia
1	Clayton Pereira Bonfim	3.694	UFMG
2	Everardo Monfort Leitão	3.438	PUC
3	Germano Luiz de Paula	93.134	Pampulha
4	Andre Cabral da Silva	89.893	PUC
5	Germano Luiz de Paula	28.154	UFMG

5 row(s) fetched - 111ms, on 2022-11-12 at 23:20:39

BRT pt_BR Gravável Inserção Inteligente 34:1 [154] Sel: 154 | 3

Agora vem a sua tarefa:

11.2

Construa a consulta equivalente a este exemplo utilizando a cláusula JOIN.

Resposta: como pedido nossa consulta ficou assim!

```
select e.nome_cliente, e.numero_conta, e.nome_agencia
       from emprestimo e natural inner join deposito d
       group by e.nome_cliente, e.numero_conta, e.nome_agencia;
```

Resultado após rodar a consulta acima.

-- 11.2 Construa a consulta equivalente a este exemplo utilizando a cláusula JOIN.

```
select e.nome_cliente, e.numero_conta, e.nome_agencia
from emprestimo e natural inner join deposito d
group by e.nome_cliente, e.numero_conta, e.nome_agencia;
```

emprestimo 1 x

Enter a SQL expression to filter results (use Ctrl+Space)

Grade	nome_cliente	numero_conta	nome_agencia
1	Andre Cabral da Silva	89.893	PUC
2	Clayton Pereira Bonfim	3.694	UFMG
3	Everardo Monfort Leitão	3.438	PUC
4	Germano Luiz de Paula	28.154	UFMG
5	Germano Luiz de Paula	93.134	Pampulha

5 row(s) fetched - 117ms, on 2022-11-12 at 23:54:05

BRT pt_BR Gravável Inserção Inteligente 40: 1 [160] Sel: 160 | 3

11.3

Construa a consulta equivalente a este exemplo utilizando a SELECT DISTINCT e sem o JOIN.

Resposta: como pedido nossa consulta ficou assim!

```
select distinct(e.nome_cliente || ' ' || e.numero_conta || ' ' || e.nome_agencia) as nome_conta_agencia
from emprestimo e, deposito d
where e.nome_cliente = d.nome_cliente
and e.numero_conta = d.numero_conta
and e.nome_agencia = d.nome_agencia;
```

Resultado após rodar a consulta acima.

-- Construa a consulta equivalente a este exemplo utilizando a SELECT DISTINCT e sem o JOIN.

```
select distinct(e.nome_cliente || ' | ' || e.numero_conta || ' | ' || e.nome_agencia) as nome_conta_agencia
from emprestimo e, deposito d
where e.nome_cliente = d.nome_cliente
and e.numero_conta = d.numero_conta
and e.nome_agencia = d.nome_agencia;
```

Results 1 x

select distinct(e.nome_cliente || ' | ' || e.numero_conta || ' | ' || e.nome_agencia) as nome_conta_agencia

nome_conta_agencia
Andre Cabral da Silva 89893 PUC
Clayton Pereira Bonfim 3694 UFMG
Everardo Monfort Leitão 3438 PUC
Germano Luiz de Paula 28154 UFMG
Germano Luiz de Paula 93134 Pampulha

5 row(s) fetched - 156ms, on 2022-11-12 at 23:58:00

BRT pt_BR Gravável Inserção Inteligente 46:1 [253] Sel: 253 | 5

12) Construa uma tabela em uma planilha (Programa Calc do Libre Office) com três colunas: intersect, distinct e join. Execute as três consultas pelo menos 30 vezes e registre na planilha o tempo de execução em milissegundos para a conclusão de cada uma das três versões da consulta. O tempo de execução de cada consulta é exibido no canto inferior direito da tela que executou uma consulta. Ao final, tire a média de cada coluna e conclua qual versão da consulta foi mais rápida.

Resposta:

G6						
	A	B	C	D	E	F
1	Intersect	Distinct	Join			
2	129	188	151			
3	113	123	178		Média	
4	134	138	225		Intersec	88,5
5	85	122	122		Distinct	140,5
6	128	167	116		Join	161
7	125	128	186			
8	100	148	151			
9	95	129	147			
10	84	160	140			
11	119	113	181			
12	52	128	157			
13	68	114	161			
14	49	150	149			
15	117	160	165			
16	53	141	181			
17	93	161	184			
18	105	163	158			
19	54	121	137			
20	81	103	192			
21	58	101	117			
22	70	105	142			
23	52	109	182			
24	118	155	191			
25	103	160	161			
26	64	172	181			
27	68	140	136			
28	71	134	189			
29	81	156	182			

Bom a consulta usando Intersect foi a mais rápida bem a frente dos outros demais com quase metade dos outros dois que vem logo em seguidas quase empatados, concluindo que join tem o maior custo e seria bom evitar o seu uso e sempre que possível usar intersect.

13.1

Implemente primeiro (por ser mais simples) uma função em PL/pgSQL para retornar o relatório descrito no cenário 2 a partir desta consulta SQL;

Aviso! Como no trecho do problema 2 não foi especificado o que a função retornaria caso a faixa do cliente estivesse menor que 1000, então tomei a liberdade de colocar D.

Contexto!

Cenário 2) Novamente, você precisa executar uma consulta em SQL para retornar dados de clientes. Os dados armazenados te permitem inferir o quão interessantes são estes clientes para receberem um novo tipo de cartão, com melhores taxas e mais crédito. Para este propósito foram criadas três faixas de clientes: A , B e C cujas as somas dos valores depositados ultrapassem seis mil, quatro mil e um mil Reais, respectivamente. O foco principal são os clientes classes A, mas existe a possibilidade de que uma parte dos novos clientes deste cartão seja oriunda de clientes do tipo B. Desta forma solicita-se que no

relatório final conste apenas o nome do cliente (e outros dados de contato, por exemplo) acompanhado da letra que denomina a faixa de classificação do cliente, as somas das quantias depositadas não devem ser exibidas no relatório. Novamente eu te pergunto: como você vai codificar em letras as faixas de depósitos dos clientes em uma única linha de consulta SQL?

Possível solução do Cenário 2) Implementar uma função que, utilizando os dados identificadores de um cliente, pesquisa na tabela de depósitos especificamente por registros de um único cliente: o cliente cuja linha esteja sendo escrita como saída do relatório final. Todos os depósitos deste cliente único seriam somados em uma variável. Em seguida esta variável seria submetida a estruturas de decisão do tipo if-then-else para determinar se a letra que deve ser retornada pela função é A, B ou C. A letra retornada pela função ocuparia então a coluna de classificação na linha do cliente especificamente consultado.

Resposta:

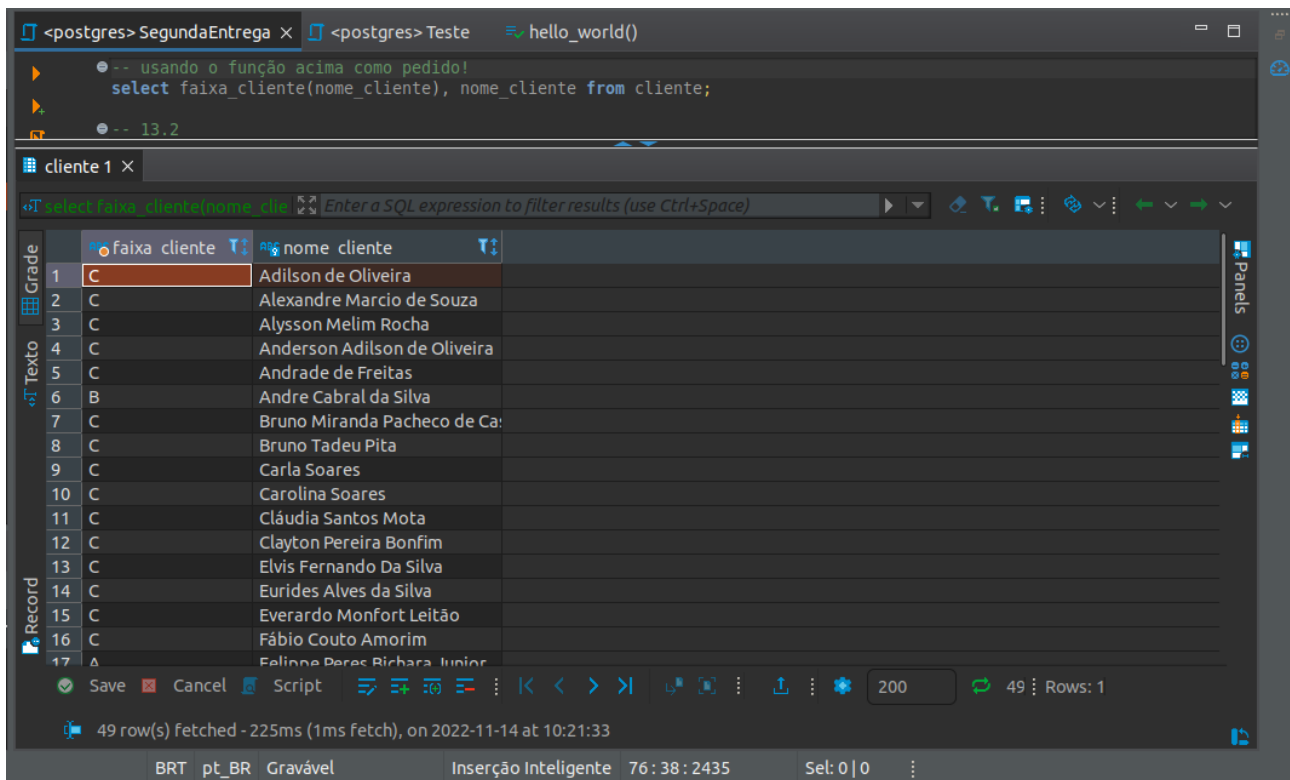
```
create or replace function faixa_cliente(nome_cliente2 varchar(80)) returns character as
$body$
declare
    soma_depositos float;
begin
    -- soma todos os depósitos do cliente que foi passado e coloca na variável soma_depositos
    select sum(d.saldo_deposito) as total from deposito d where d.nome_cliente = nome_cliente2 into
soma_depositos;

    if (soma_depositos > 6000) then
        return 'A';
    elsif soma_depositos > 4000 then
        return 'B';
    else
        return 'C'; -- como indicado pelo professor!
    end if;

end;
$body$ LANGUAGE plpgsql; -- ok funcionando!

select faixa_cliente(nome_cliente), nome_cliente from cliente;
```

Usando a função abaixo obtemos:



13.2

Implemente uma função em PL/pgSQL para retornar o relatório descrito no cenário

Contexto!

Cenário 1) Você precisa executar uma consulta em SQL para retornar os dados de clientes e para cada cliente será criada uma única linha em um relatório. Até aqui podemos resolver facilmente com uma simples consulta em SQL. Entretanto, foi solicitado que os números de todas as contas de um cliente, de cada agência, sejam exibidos nesta mesma linha destinada aos dados do cliente. Neste caso, haverá uma coluna no relatório que vai juntar o nome da agência ao número da conta e este par de dados será exibido como um só dado (exemplo: “Central-12345”). Quantas contas e agências tiver um cliente devem ser exibidas nesta coluna separado por vírgula (exemplo: “Central-12345”, “Pampulha-67890”, ...). E agora? Como você vai retornar todas as contas do cliente em uma única consulta, de modo que essa lista passe a constituir uma das colunas de dados a serem retornadas?

Possível solução do Cenário 1) Implementar uma função que, utilizando os dados identificadores de um cliente, pesquisa na tabela de contas especificamente por registros de um único cliente: o cliente cuja linha esteja sendo escrita como saída do relatório final. Utilizando um loop para iterar entre todos os resultados da consulta realizada somente aos dados deste único cliente, todos os números de conta e suas respectivas agências seriam retornados, um a um, para uma variável de texto que concatenasse os dados na forma “Nome da Agência-Número da Conta”. Em seguida esta variável de texto deveria também ser concatenada a uma outra variável de texto com o propósito de criar uma lista de contas de cada cliente. Quando o loop terminasse a sua execução, por não haver mais registros a serem retornados para um determinado cliente, então todas as contas deste cliente estariam presentes na variável lista e esta seria retornada pela função. Ao receber o valor de retorno da função a nossa consulta inicial utilizará este valor para ocupar a coluna de contas para o cliente da vez, ou seja, o cliente cujos dados estão sendo escritos no relatório final.

Resposta:

```
create or replace function contas_cliente (nome_cliente2 varchar(80)) returns varchar as
$body$
```



```

declare
    lista varchar;
    dados varchar;

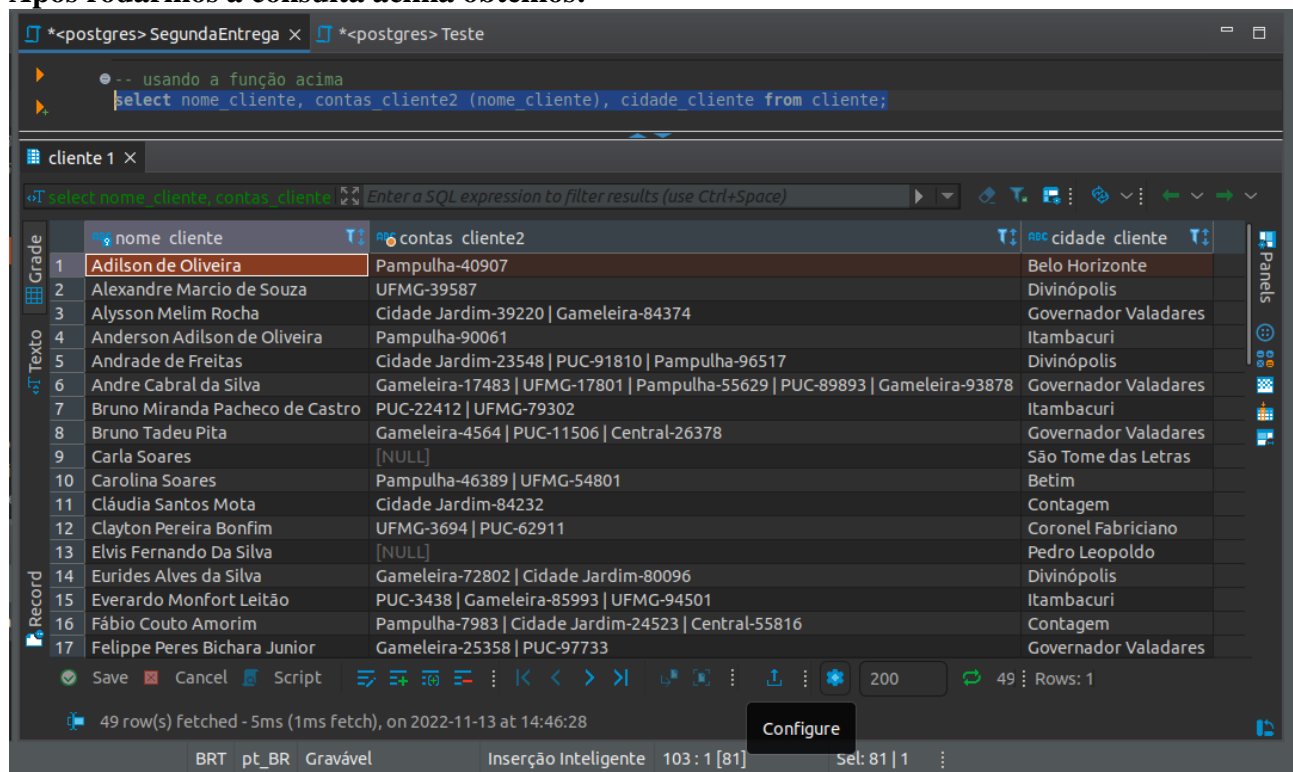
begin
    for dados in select c.nome_agencia || '-' || c.numero_conta from conta c where c.nome_cliente =
nome_cliente2 loop
        if lista is null then
            lista := dados;
        else
            lista := lista || '-' || dados;
        end if;
    end loop;

    return lista;
end;
$body$ LANGUAGE plpgsql;

```

select nome_cliente, contas_cliente (nome_cliente), cidade_cliente from cliente;

Após rodarmos a consulta acima obtemos:



The screenshot shows a PostgreSQL client window with a query executed. The query is: `select nome_cliente, contas_cliente2 (nome_cliente), cidade_cliente from cliente;`. The result is displayed in a table with 17 rows. The table has three columns: 'nome cliente', 'contas_cliente2', and 'cidade_cliente'. The data is as follows:

nome cliente	contas_cliente2	cidade_cliente
Adilson de Oliveira	Pampulha-40907	Belo Horizonte
Alexandre Marcio de Souza	UFMG-39587	Divinópolis
Alysson Melim Rocha	Cidade Jardim-39220 Gameleira-84374	Governador Valadares
Anderson Adilson de Oliveira	Pampulha-90061	Itambacuri
Andrade de Freitas	Cidade Jardim-23548 PUC-91810 Pampulha-96517	Divinópolis
Andre Cabral da Silva	Gameleira-17483 UFMG-17801 Pampulha-55629 PUC-89893 Gameleira-93878	Governador Valadares
Bruno Miranda Pacheco de Castro	PUC-22412 UFMG-79302	Itambacuri
Bruno Tadeu Pita	Gameleira-4564 PUC-11506 Central-26378	Governador Valadares
Carla Soares	[NULL]	São Tome das Letras
Carolina Soares	Pampulha-46389 UFMG-54801	Betim
Cláudia Santos Mota	Cidade Jardim-84232	Contagem
Clayton Pereira Bonfim	UFMG-3694 PUC-62911	Coronel Fabriciano
Elvis Fernando Da Silva	[NULL]	Pedro Leopoldo
Eurides Alves da Silva	Gameleira-72802 Cidade Jardim-80096	Divinópolis
Everardo Monfort Leitão	PUC-3438 Gameleira-85993 UFMG-94501	Itambacuri
Fábio Couto Amorim	Pampulha-7983 Cidade Jardim-24523 Central-55816	Contagem
Felippe Peres Bichara Junior	Gameleira-25358 PUC-97733	Governador Valadares

The interface also shows a status bar at the bottom indicating '49 row(s) fetched - 5ms (1ms fetch), on 2022-11-13 at 14:46:28' and a 'Configure' button.