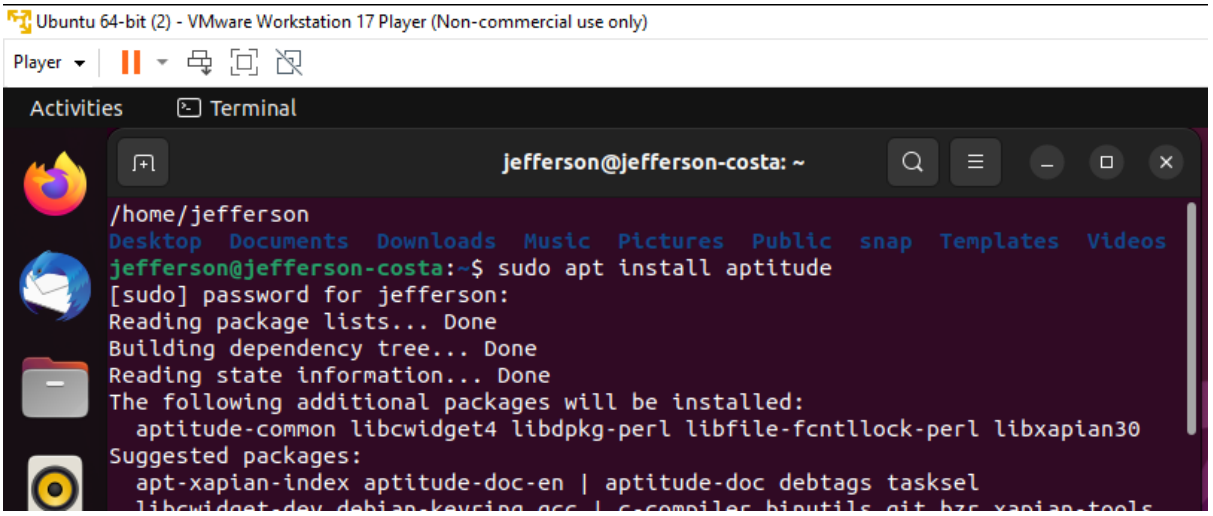
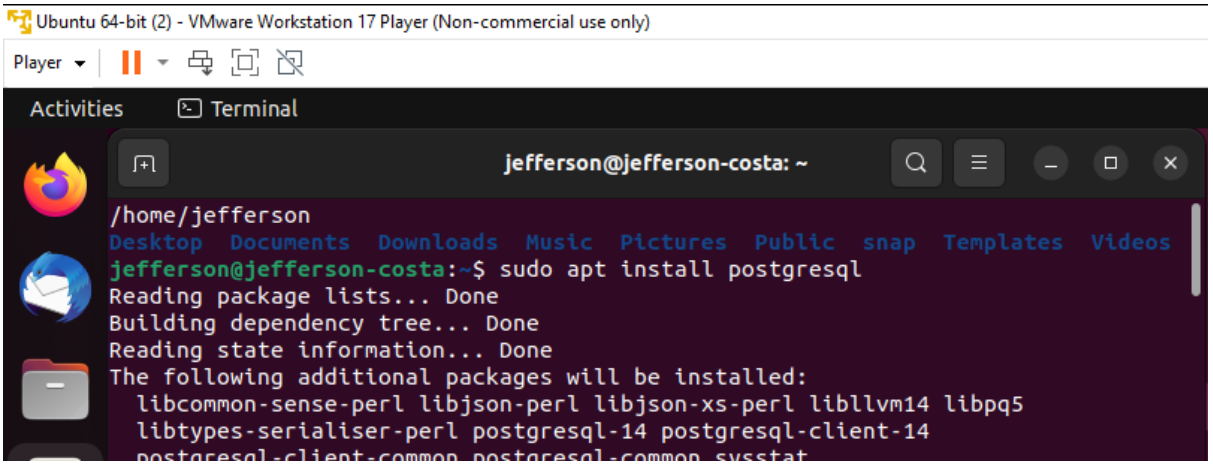
Instalação do Postgres, To join or not to Join e funções
Jefferson Cristino da Costa - 11821BSI227

1.1 sudo apt install aptitude

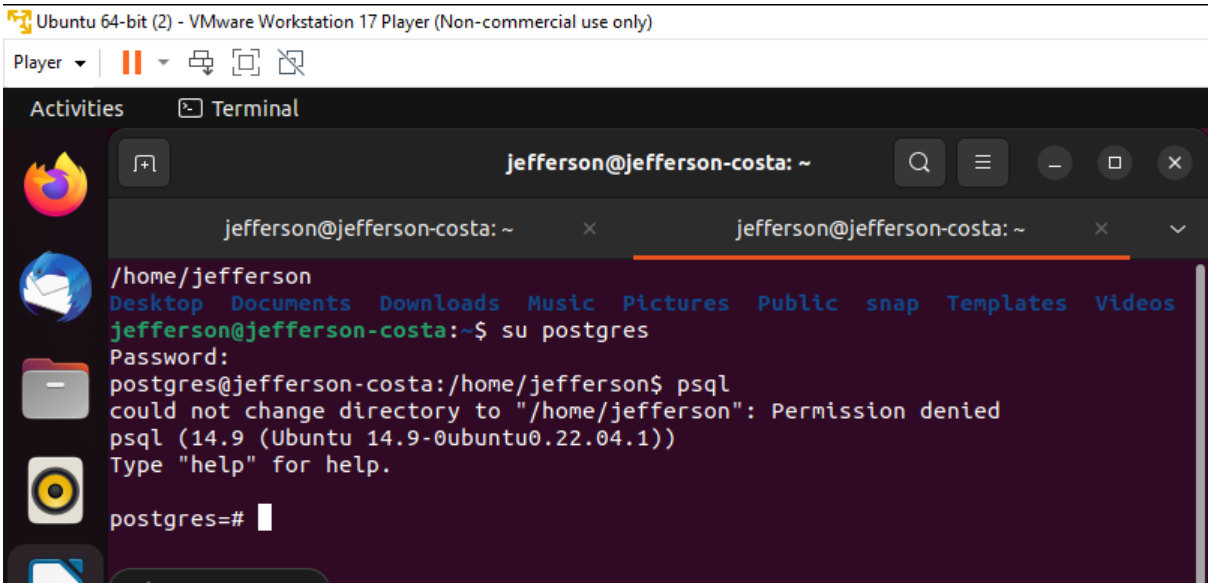


2.1 sudo apt install postgresq

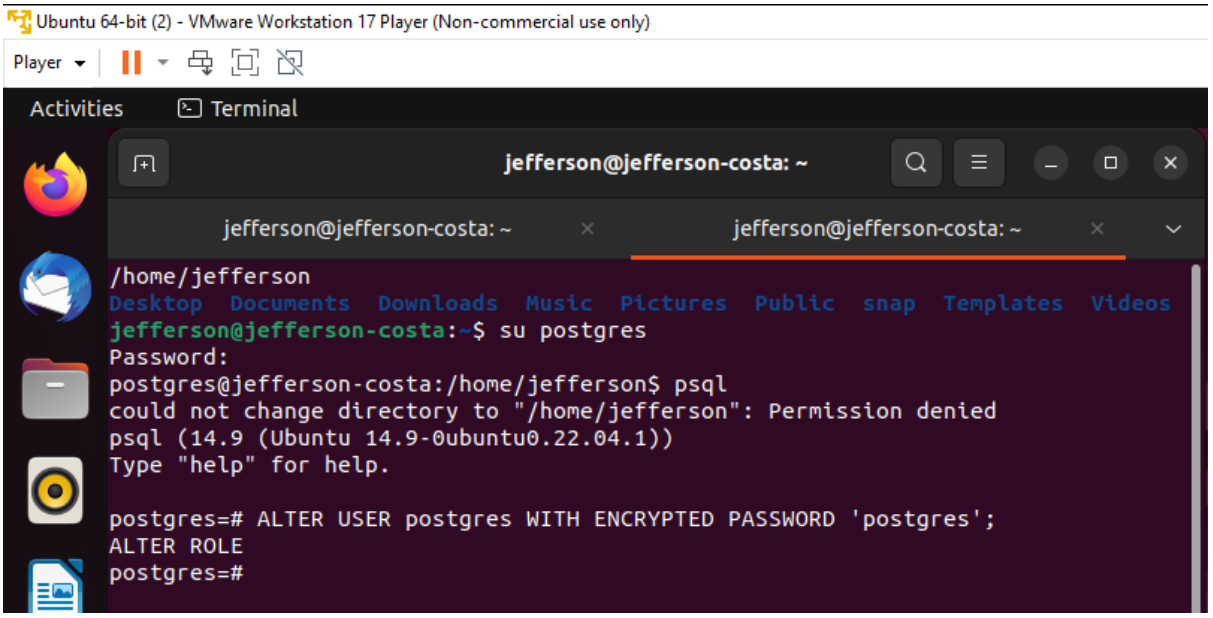


3.1 sudo passwd postgres

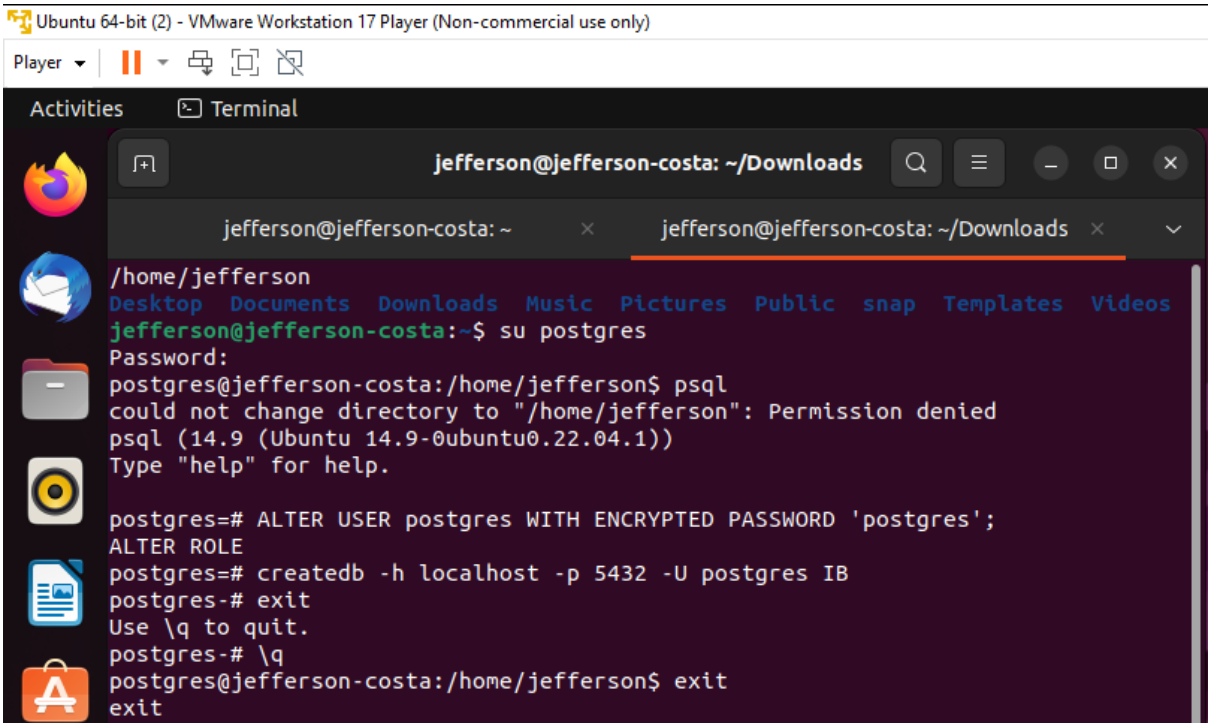
4.1 su postgres (digite a senha criada no passo anterior)
4.2 psql



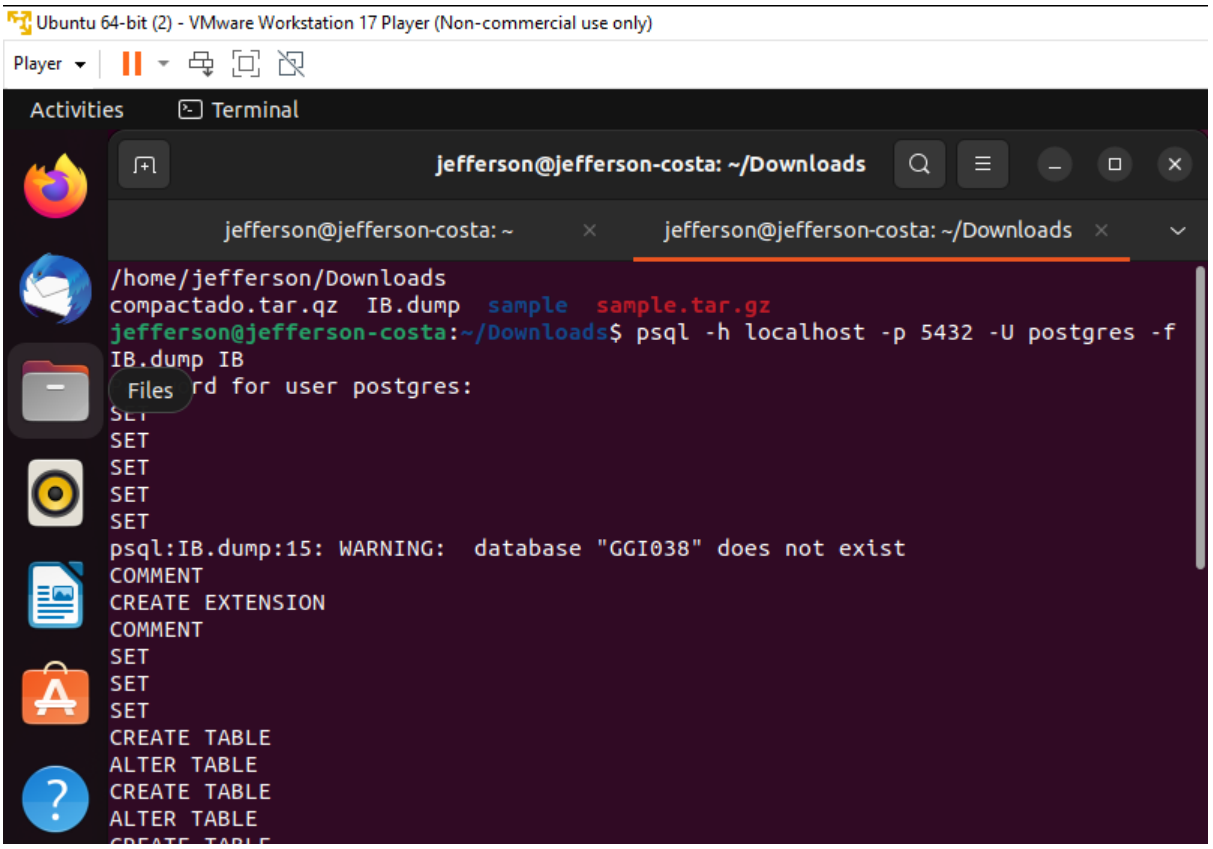
5 Devemos alterara senha do usuário postgres para o mesmo valor utilizando o seguinte comando: `ALTER USER postgres WITH ENCRYPTED PASSWORD 'postgres';`



6 createdb -h localhost -p 5432 -U postgres IB

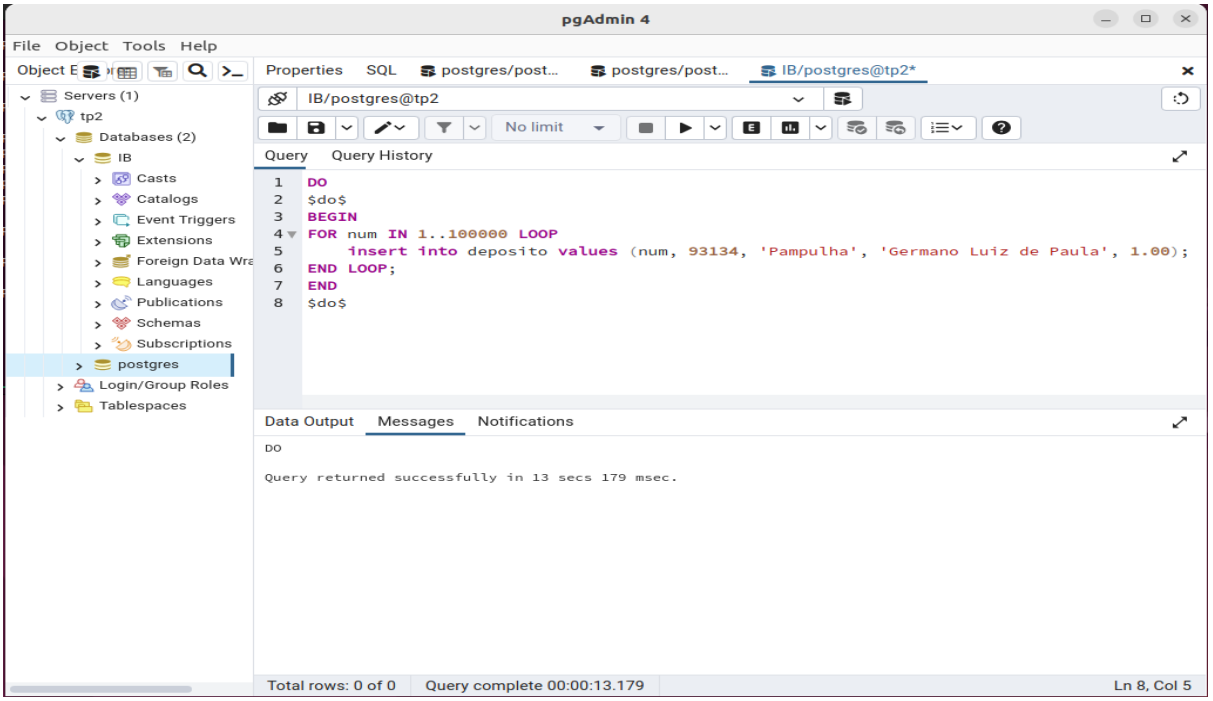


6.3 Agora posicione-se no diretório no qual o download do arquivo IB.dump foi efetuado, por exemplo, com 'cd /home/login_usuário/Downloads' e digite o comando: psql -h localhost -p 5432 -U postgres -f IB.dump IB

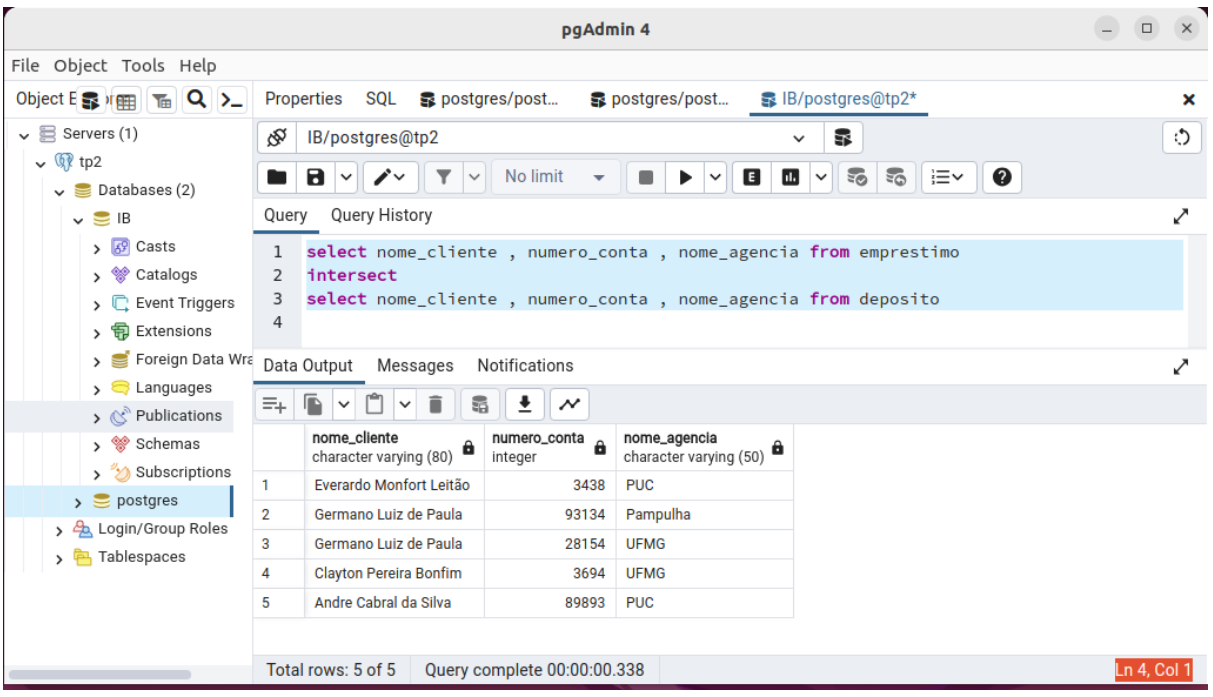


6.4 criar esta conexão via o pgadmin3 ao banco IB.

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.

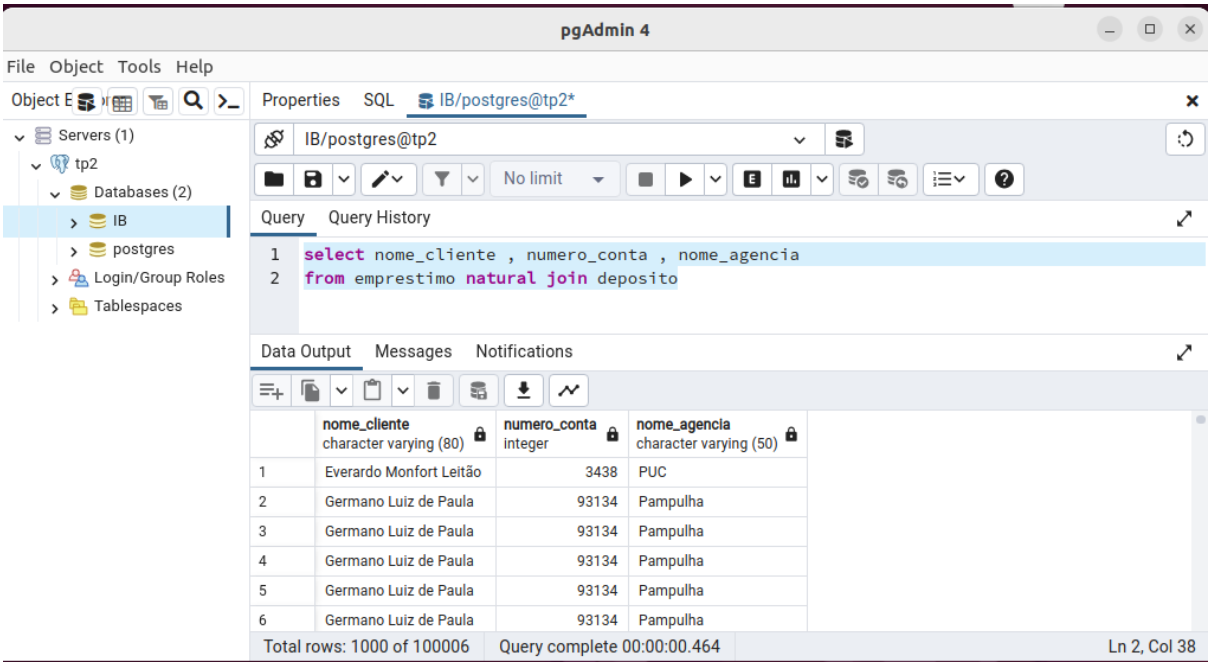


11.1 Selecione os nomes dos clientes e seus respectivos números de conta e nome de agência que fizeram depósitos e empréstimos ao mesmo tempo. Este código em SQL resolve esta consulta sem fazer uso da cláusula JOIN:



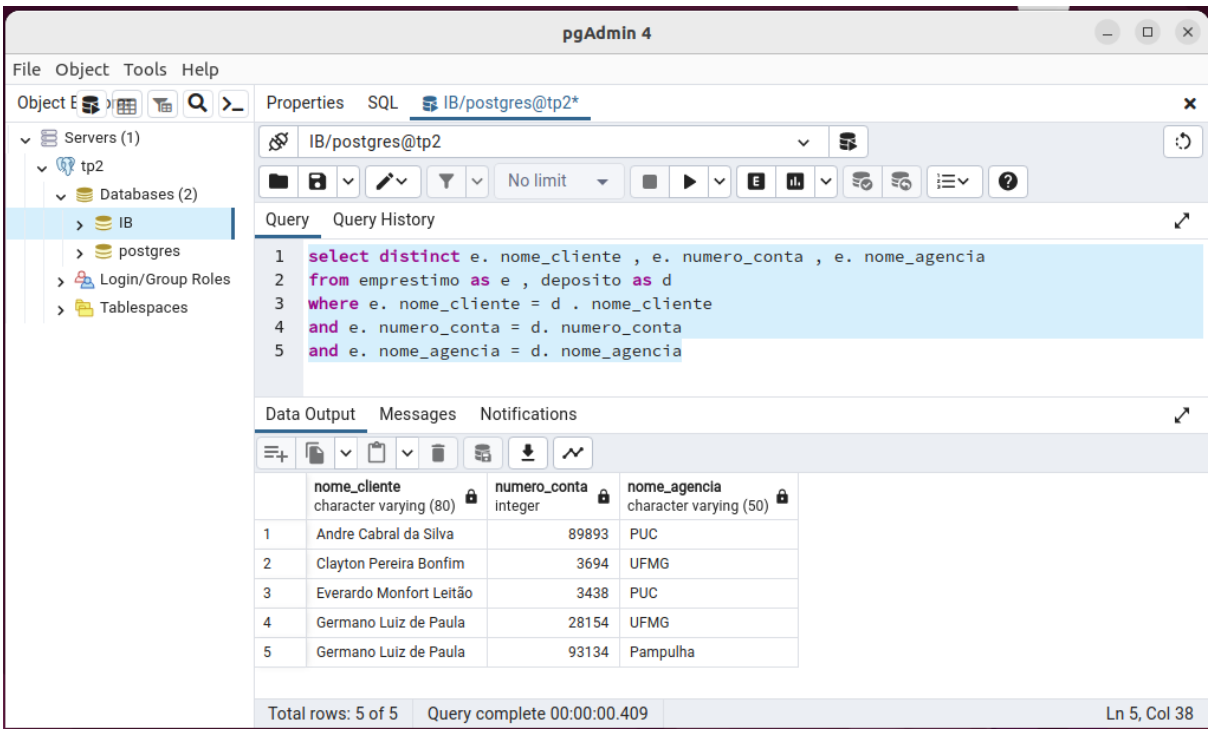
11.2 Construa a consulta equivalente a este exemplo utilizando a cláusula JOIN.

```
select nome_cliente , numero_conta , nome_agencia
from emprestimo natural join deposito
```



11.3 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
from emprestimo as e , deposito as d
where e. nome_cliente = d . nome_cliente
and e. numero_conta = d. numero_conta
and e. nome_agencia = d. nome_agencia
```



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 naplanilha 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.

INTERSECT	JOIN	DISTINCT
0,34	0,46	0,19
0,3	0,45	0,279
0,273	0,467	0,17
0,25	0,626	0,279
0,293	0,466	0,22
0,271	0,58	0,233
0,315	0,566	0,191
0,293	0,441	0,224
0,289	0,47	0,206
0,255	0,476	0,345
0,276	0,539	0,35
0,273	0,455	0,542
0,291	0,613	0,447
0,279	0,479	0,351

	0,27	0,488	0,22
	0,355	0,474	0,19
	0,313	0,385	0,335
	0,263	0,45	0,257
	0,257	0,431	0,271
	0,311	0,339	0,402
	0,233	0,47	0,266
	0,319	0,455	0,311
	0,244	0,34	0,542
	0,3	0,652	0,544
	0,239	0,445	0,345
	0,345	0,477	0,331
	0,266	0,555	0,205
	0,271	0,595	0,192
	0,27	0,46	0,225
	0,261	0,412	0,289
MÉDIA =	0,2838333333	0,4839	0,2984

13 **Cenário 1)**

```
1
2 --A DEFINICAO DA FUNCAO GETDADOSCLIENTE
3 DROP FUNCTION getDadosCliente ( character varying );
4
5 CREATE OR REPLACE FUNCTION getDadosCliente ( p_nome_cliente
character
varying )
6 RETURNS character varying AS
7 $BODY$
8 DECLARE finished INTEGER DEFAULT 0;
9 DECLARE
10 dados_agencia character varying ;
11 dados_conta int ;
12 dados_cliente character varying ;
13 conta character varying ;
14 cursor_relatorio cursor FOR SELECT nome_cliente , nome_agencia ,
numero_conta FROM conta
15 WHERE nome_cliente = p_nome_cliente ;
16 BEGIN
17 OPEN cursor_relatorio ;
```

```
18 conta = ”;
19 LOOP
20 FETCH cursor_relatorio into dados_cliente , dados_agencia ,
dados_conta ;
21 IF FOUND THEN
22 conta = conta || dados_agencia || ’ - ’ || dados_conta || ’
, ’;
23 END IF ;
24 IF NOT FOUND THEN
25 CLOSE cursor_relatorio ;
26 RETURN conta ;
27 END IF ;
28 END LOOP ;
29 END
30 $BODY$
31 LANGUAGE plpgsql VOLATILE
32 COST 100;
33 ALTER FUNCTION getDadosCliente ( character varying )
34 OWNER TO postgres ;
35
36 select nome_cliente , getDadosCliente ( nome_cliente ) from conta
```

Cenário 2)

```
1
2 CREATE OR REPLACE function getClassificacao ( p_numero_conta integer ,
p_nome_agencia character varying , p_nome_cliente character
varying )
3 returns character varying as
4 $BODY$
5 declare
6 soma_deposito float ;
7 classificacao character varying ;
8 cursor_relatorio cursor for select sum( d. saldo_deposito ) as
total_dep
9 from conta c natural left outer join deposito d
10 where c. nome_cliente = p_nome_cliente
11 and c. nome_agencia = p_nome_agencia
12 and c. numero_conta = p_numero_conta
13 group by c. nome_cliente , c. nome_agencia , c. numero_conta ;
14 begin
15 open cursor_relatorio ;
16 fetch cursor_relatorio into soma_deposito ;
```

```
17 if found then
18 if soma_deposito is null then soma_deposito = 0;
end if ;
19 if soma_deposito > 6000 then classificacao = 'A';
end if ;
20 if soma_deposito between 6000 and 4000 then
classificacao = 'B'; end if ;
21 if soma_deposito < 4000 then classificacao = 'C';
end if ;
22 end if ;
23 close cursor_relatorio ;
24 return classificacao ;
25 end
26 $BODY$
27 language plpgsql volatile
28 cost 100;
29 alter function getClassificacao ( integer , character varying ,
character varying )
30 owner to postgres ;
31
32 select numero_conta , nome_agencia , nome_cliente , getClassificacao (
numero_conta , nome_agencia , nome_cliente ) from conta c
```