



UNIVERSIDADE PITÁGORAS UNOPAR

GESTÃO DA TECNOLOGIA DA INFORMAÇÃO

NOME DO(S) AUTOR(ES) EM ORDEM ALFABÉTICA
JOÃO BOSCO FERREIRA DUARTE DA SILVEIRA

Gerenciamento e Desenvolvimento em Banco de Dados

NOME DO(S) AUTOR(ES) EM ORDEM ALFABÉTICA

JOAO BOSCO FERREIRA DUARTE DA SILVEIRA

Gerenciamento e Desenvolvimento em Banco de Dados

Trabalho da matéria Gerenciamento e Desenvolvimento em Banco de Dados, onde iremos elaborar um Banco de Dados no software SQL Fiddle para uma loja virtual de eletrônicos, apresentado à Universidade Pitágoras Unopar, como requisito parcial para a obtenção de média bimestral na disciplina de Gerenciamento e Desenvolvimento em Banco de Dados.

Orientador:

Prof. Marco Gilberto Fernandes Junior

Macaé - RJ
2023

Sumário

1	INTRODUÇÃO	3
2	DESENVOLVIMENTO	4
	2.1 Proposta da atividade.....	4
	2.2 Resolução da proposta.....	4
3	CONCLUSÃO	21
4	REFERÊNCIAS.....	22

1 INTRODUÇÃO

Esta atividade foi dividida em 3 etapas. A primeira, foi criar o banco de dados e as tabelas com as entidades “Cliente, Produto e Compras” e os atributos de cada uma delas, como iremos ver no decorrer do exercício.

A segunda etapa foi inserir os dados para cada entidade, exatamente como foi pedido no exercício.

A última etapa foi utilizar o comando `SELECT` para realizar as consultas propostas e exigidas pelo enunciado.

Neste relatório, será mostrado com explicações e imagens, toda a trajetória até a conclusão do exercício, com os detalhes de como foi feita cada etapa.

2 DESENVOLVIMENTO

2.1 PROPOSTA DA ATIVIDADE

Explorar a ferramenta SQL Fiddle e realizar a criação de um banco de dados relacional, bem como a realização da inserção e consulta dos dados.

2.2 RESOLUÇÃO DA PROPOSTA

Primeiramente, entrei na Plataforma SQL Fiddle, como solicitado, para começar pela criação do Database, antes de criar as tabelas em si, contudo, não consegui gerar o Banco de Dados e, após pesquisar sobre, conforme informado no seguinte link (<https://stackoverflow.com/questions/34909908/sql-fiddle-threw-create-database-permission-denied-in-database-master>), o SQL FIDDLE 5.6 não permitiu que fosse criado o banco de dados, portanto, segue-se com o exercício para a criação das tabelas e atributos.

Nesta fase, deve-se construir as tabelas com seus atributos, sendo assim, por ordem, a primeira tabela é a “tabela Cliente” com os atributos “código”, “nome”, “estado”, “cidade” e “telefone”, com o código sendo chave primária.

Partindo para a construção da tabela e atributos da entidade Cliente, o primeiro passo é “Create Table Cliente”, código integer auto_increment, pois foi especificado no exercício que o código teria que ser auto incremento. Logo depois o Nome do cliente, onde nele programamos como varchar, ou seja, “nome varchar (100)”, pois “varchar” é o tipo para caracteres do tipo textos e o “(100)” se trata da quantidade limite destes caracteres no nome.

Então seguindo as características técnicas do nome (varchar, com 100 caracteres), foi feito restante dos atributos, sendo eles “estado varchar (100)”,

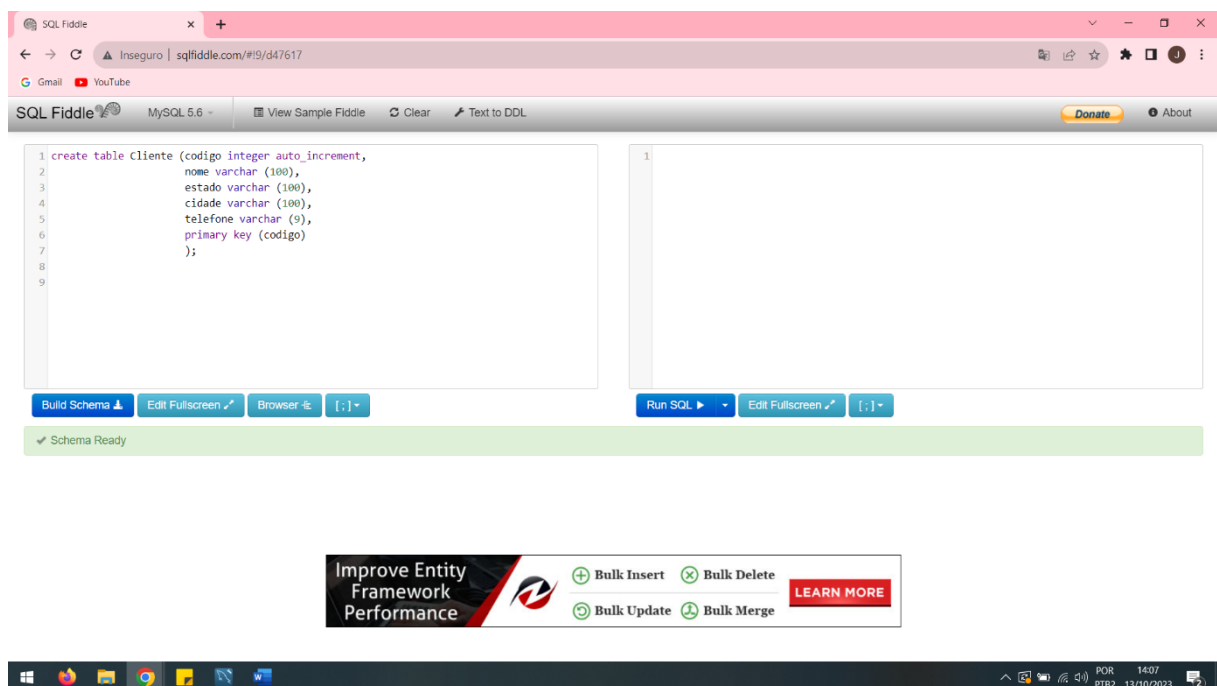
“cidade varchar (100)” e “telefone varchar (9)”.

Depois de terminar a linha do telefone, salta-se para próxima e, para que fique organizado e padronizado, faz-se então a chave primaria para o código do cliente, escrevendo no final “primary key (código)”. Optei por programar a chave primaria no final de cada tabela, pois pessoalmente, fica mais organizado para mim seguir esse padrão.

Então após o “primary key (código)”, não há mais atributos a serem colocados na tabela Cliente, portanto o que resta a fazer é se certificar que não houve nenhum erro na sintaxe e que todos a pontuação está correta.

Com tudo preenchido e analisado, agora clica-se em “Build Schema”, no SQL Fiddle para verificar se a tabela foi criada com sucesso ou se houve algum erro durante a programação. E caso não haja erros de digitação, sintaxe ou pontuação, a programação deverá ficar como demonstrado na **imagem 1** abaixo:

Imagem 1



Como visto na imagem acima, o “Schema” da entidade Cliente, está pronto para rodar, portanto agora será feita a tabela da entidade Produto.

Para a programação da tabela Produto, as regras seguidas serão as mesmas, pois como dito no exercício, a mesma também é chave primaria com auto incremento, sem mudanças radicais, em relação a tabela Cliente.

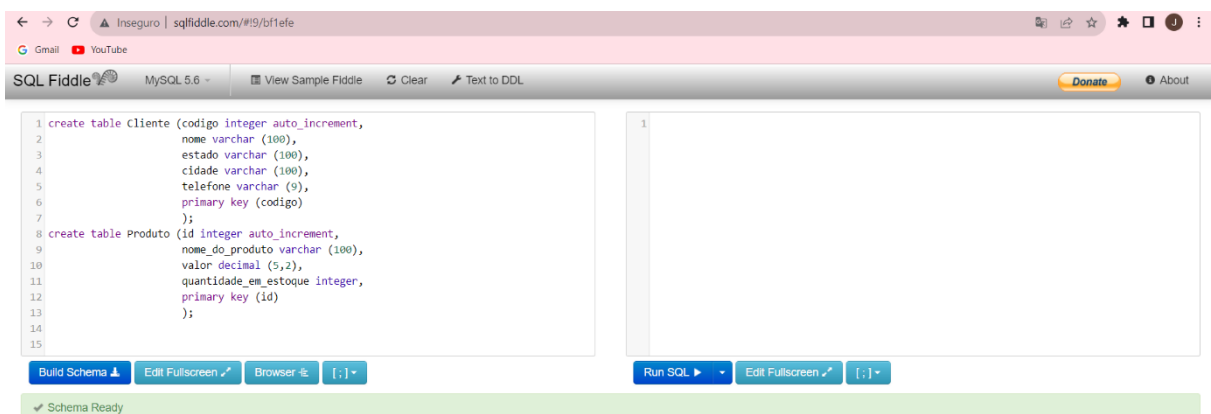
Primeiramente, é criado a tabela com os comandos “create table Produto com o id inteiro e auto incremento, sendo o comando “id integer auto_increment”.

Logo após, será constituída a tabela com o restante dos atributos, sendo eles o nome do produto, valor e quantidade em estoque. Na linguagem dos comandos, será “nome_do_produto varchar (100)”, “valor decimal (5,2)” e “quantidade_em_estoque integer”. A diferença na programação desta tabela está no valor. Perceba que o valor está programado para “decimal (5,2). Isso significa que, na tabela, o valor deverá ser digitado somente em algarismos e o significado de “5,2” é que, no ato da inserção dos dados, o limite é de cinco algarismos inteiros e dois decimais, como por exemplo “25000,50”, este seria o modelo no momento da inserção e o valor pode ser qualquer um, desde que respeite a regra de, no máximo, cinco algarismos e dois decimais.

Após a programação dos atributos na tabela, termina-se com “primary key (id) ;”, pois como informado acima, nessa programação, por razões organizacionais o “primary key” é inserido na última linha.

Feita a verificação dos algarismos, sintaxe e pontuação, clique em “Build Schema” e ficará exatamente como mostrado na **imagem 2** abaixo:

Imagem 2



Com as tabelas Cliente e Produto elaboradas, agora resta a programação da tabela Compras. No caso desta tabela, alguns requisitos diferentes, foram estabelecidos, entre eles está a chave estrangeira no código do cliente e no id do produto. Como nas tabelas anteriores a chave primária foi programada na última linha, será mantido esta organização e começa-se a programação normalmente.

Primeiro define-se a tabela com o “create table Compra”, depois, define-se a programação dos atributos. Nesse caso, os atributos desta tabela são o número, a data da compra, o código do cliente, o id do produto, a quantidade comprada e o valor da compra, lembrando que o código do cliente e o id do produto devem ser cadastrados com chave estrangeira, sendo assim, a programação ficará:

numero integer, que significa que será um numero inteiro, data_da_compra datetime, onde datetime é especificamente a ordem de programação para a inserção de datas na tabela, codigo_do_cliente integer, que por ser um código numeral, também pode-se por inteiro, id_do_produto integer, quantidade_comprada integer, valor_da_compra decimal (8,2), onde coloca-se decimal, visto que se trata de algo com valor, nesse caso com limite de 8 algarismos e duas decimais (11111111,11).

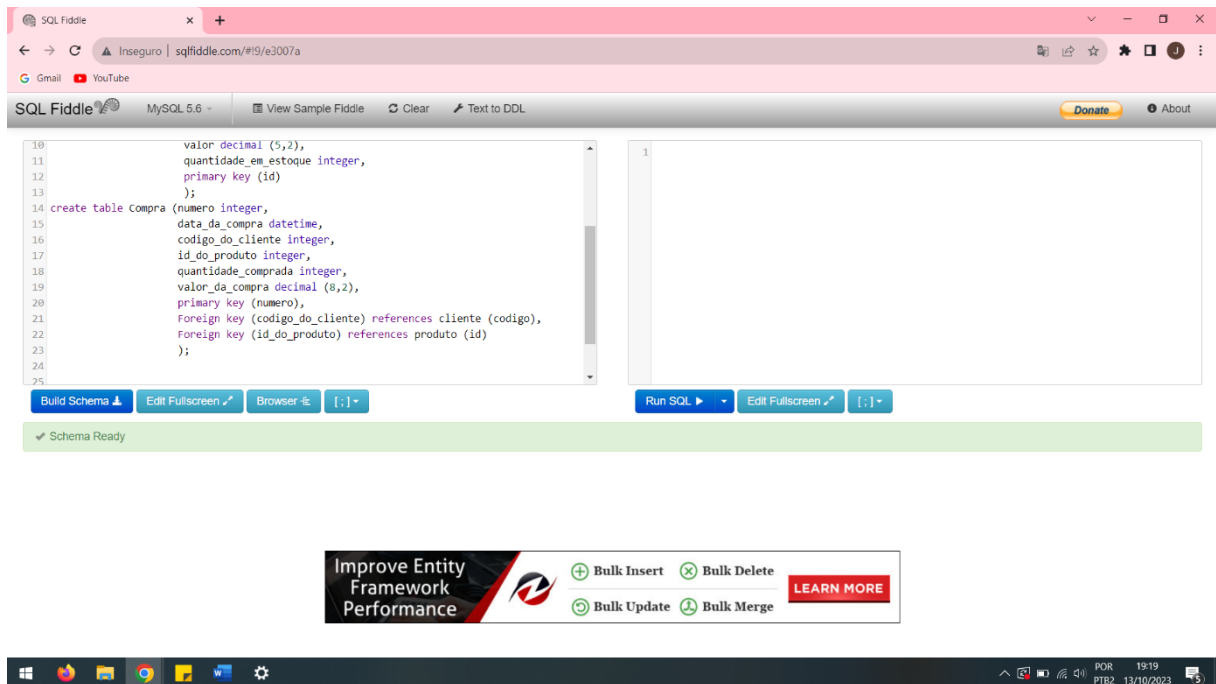
E agora, nas últimas linhas, cadastra-se tanto a chave primaria, quanto a estrangeira. A primária será o número “primary key (numero)” e quanto a chave estrangeira, para cadastra-la, é preciso não so programar ela por si mesma, como também programar algo na qual ela irá evidenciar. Para isso, é preciso cadastra-la e logo após inserir o comando “references” seguido pelo que quer que aquela chave referencie ou evidencie. Nesse caso, o enunciado ordena que o código do cliente seja referenciado pela chave estrangeira, além do id do produto. Portanto, a programação deverá ser escrita da seguinte forma:

“Foreign key (codigo_do_cliente) references cliente (código), Foreign key (id_do_produto) references produto (id));”.

Percebe-se que ao deixar para as linhas finais o cadastramento das chaves, desde o início, a programação sai mais fluida e organizada e essa era, precisamente, a intenção.

Após, novamente, verificar toda sintaxe e pontuação, é pressionado o “Build Schema” e a tabela é criada com sucesso. Para uma melhor ilustração, veja a imagem 3 abaixo:

Imagem 3



Finalizada a criação das três tabelas, agora a segunda etapa do exercício é estabelecida.

Com as tabelas programadas, para que sejam visíveis e úteis, agora é necessário que os dados sejam inseridos.

O exercício, por sua vez, forneceu os dados nos quais deverão ser inseridos em todas as três entidades.

Para a tabela Cliente, estão todos os nomes, estados, cidades e telefones já propostos pelo enunciado.

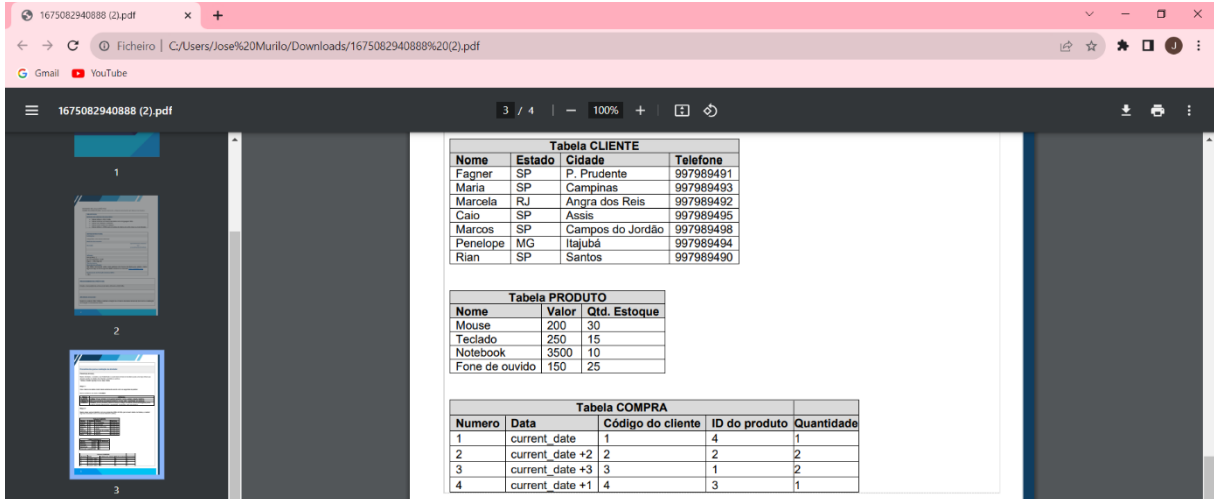
Para a tabela Produto, estão os produtos nos quais devem se cadastrados, os valores, as quantidades, etc.

E, não diferente, para a tabela Compra, também há as informações, contudo, o exercício informa que deverá ser programado a data como "current_date", sem a modificação da mesma.

Nas imagens 4 e 5, é possível verificar, com mais detalhes, quais os dados que o enunciado deseja que sejam inseridos:

Imagem 4

9



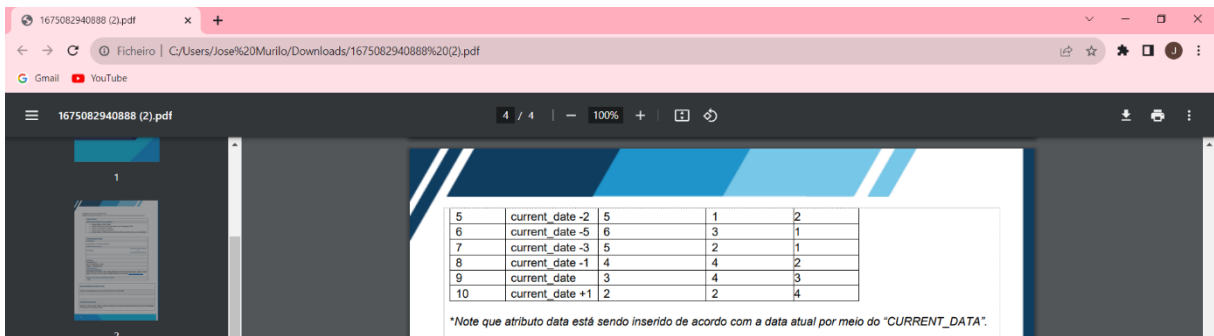
The screenshot shows a PDF document with three tables. The first table, 'Tabela CLIENTE', lists client information. The second table, 'Tabela PRODUTO', lists product information. The third table, 'Tabela COMPRA', lists purchase transactions.

Nome	Estado	Cidade	Telefone
Fagner	SP	P. Prudente	997989491
Maria	SP	Campinas	997989493
Marcela	RJ	Angra dos Reis	997989492
Caio	SP	Assis	997989495
Marcos	SP	Campos do Jordão	997989498
Penelope	MG	Itajubá	997989494
Rian	SP	Santos	997989490

Nome	Valor	Qtd. Estoque
Mouse	200	30
Teclado	250	15
Notebook	3500	10
Fone de ouvido	150	25

Numero	Data	Código do cliente	ID do produto	Quantidade
1	current_date	1	4	1
2	current_date +2	2	2	2
3	current_date +3	3	1	2
4	current_date +1	4	3	1

Imagem 5



The screenshot shows a PDF document with a table containing 5 columns and 10 rows of data. The table is followed by a note in Portuguese.

5	current_date -2	5	1	2
6	current_date -5	6	3	1
7	current_date -3	5	2	1
8	current_date -1	4	4	2
9	current_date	3	4	3
10	current_date +1	2	2	4

*Note que atributo data está sendo inserido de acordo com a data atual por meio do "CURRENT_DATE".

Sendo assim, seguindo tais diretrizes, para começar a inserção dos dados, o comando deverá ser escrito exatamente na ordem de preenchimento. O comando usado será "insert into Cliente (nome, estado, cidade, telefone) e utilizando o comando "values", segue-se com a inserção, de acordo com os dados propostos no exercício, e a sintaxe ficará: ("Fagner", "SP", "P. Prudente", "997989491)". Todos os valores devem ser inseridos na respectiva ordem em que foram colocados anteriormente, nesse caso, primeiro o nome Fagner, depois o estado SP, seguido da cidade e até chegar no último que é o telefone.

Após inserir tais dados, um mecanismo que pode acelerar o processo de inserção é copiar e colar a linha e editá-la depois.

Se nenhum erro foi cometido, os dados da tabela cliente foram inseridos e ficarão como na imagem a seguir.

Nota-se também, na Imagem 7 que ao usar o comando SELECT, já é

possível ver a tabela Cliente pronta com os dados inseridos:

Imagem 6

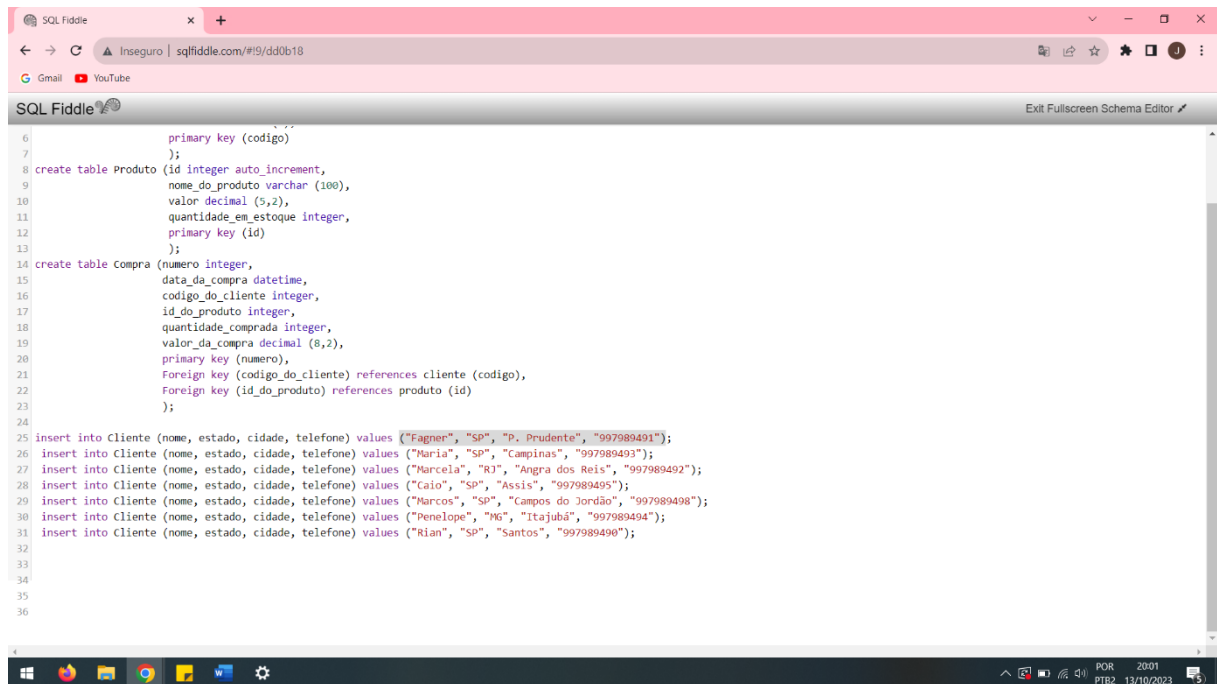
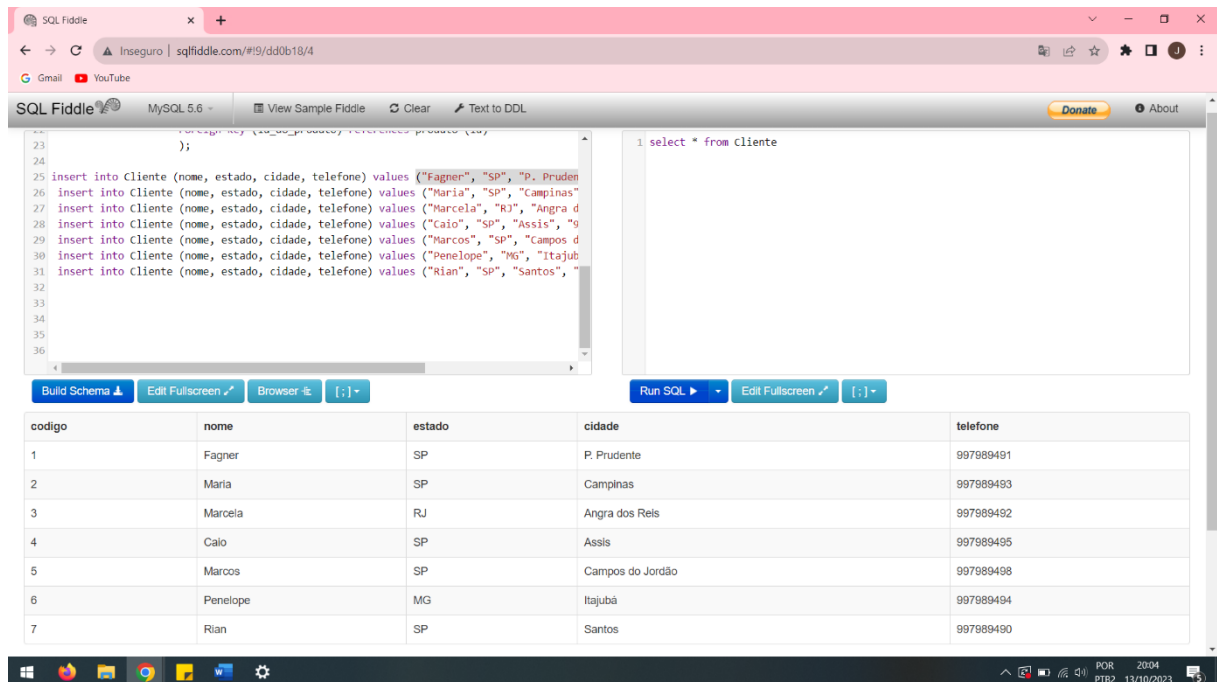


Imagem 7



Com a tabela Cliente já preenchida, o próximo passo é preencher a tabela Produto e, seguindo as mesmas regras da tabela anterior, usamos o comando insert into Produto, normalmente, seguido pelo nome do produto, valor e quantidade em estoque, nessa ordem, entre parênteses: “(nome_do_produto, valor, quantidade_em_estoque)”.

Após o script, agora é preciso inserir os dados, como na tabela anterior, com o comando “values” e os dados ditos no exercício entre parênteses, sempre respeitando a pontuação e a ordem: “values (“Mouse”, 200, 30);”.

Analizando com cautela, percebe-se que enquanto todos os outros nomes possuem aspas, os Algarismos que representam os valores 200 e 30 não possuem. E isso é dado exatamente por serem e representarem Algarismos. Especificamente nesse caso, não deve-se colocar aspas nos valores, pois os mesmos estão representando quantidade e preços.

Seguindo a dica da tabela anterior, sobre copiar, colar e depois editar os dados, rapidamente, é possível finalizar a inserção de todos os dados e ao executar e dar o comando “SELECT * FROM PRODUTO”, já é possível ver a tabela pronta, como mostra a imagem 9.

Imagem 8

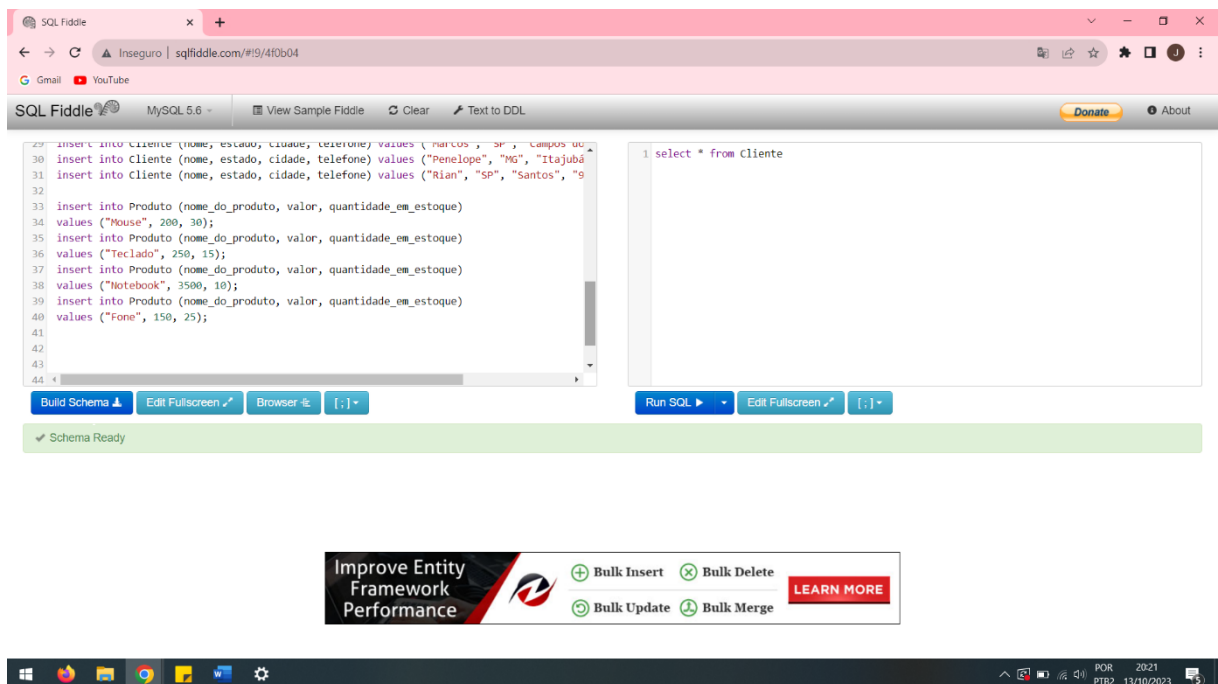
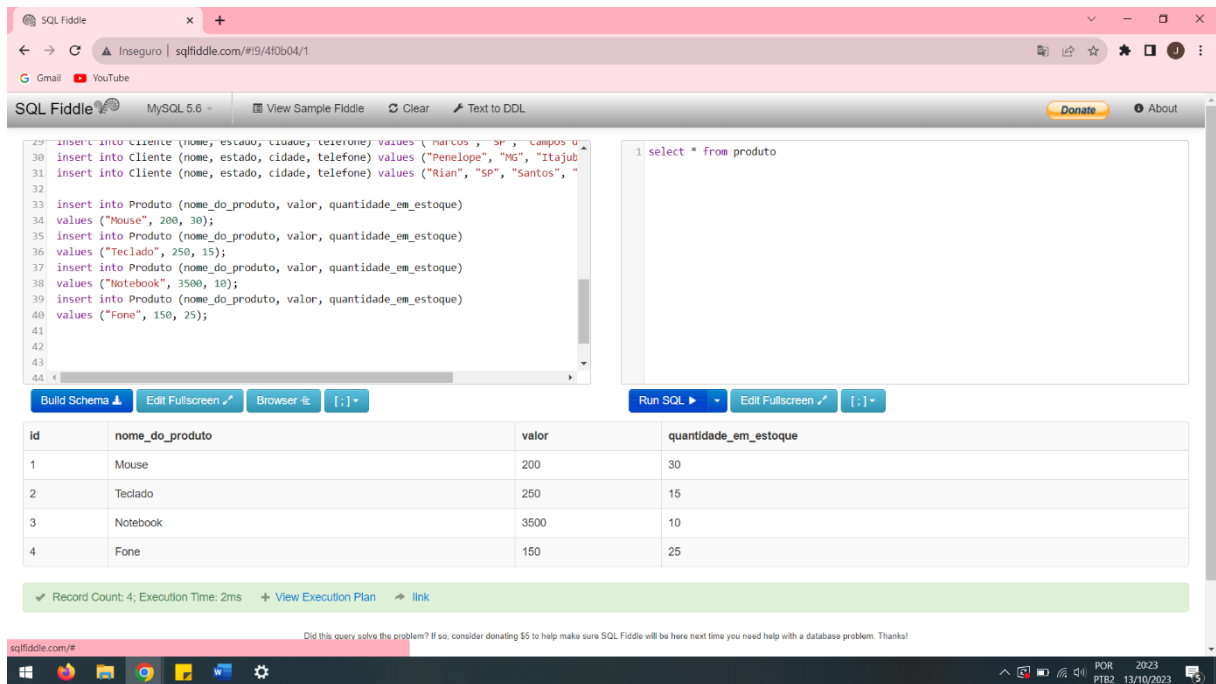


Imagem 9



Por fim, agora deve-se fazer a inserção dos dados da tabela Compras.

Para isso, será inserido o número, data da compra, onde o exercício ordena que seja feita na data o "current_date", código do cliente, id do produto, quantidade comprada e valor da compra. Para essa inserção, como nas anteriores, deve-se respeitar a ordem dos atributos no momento da inserção.

Para a inserção usa-se o comando "insert into Compra" e entre parênteses ficará "(numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_comprada, valor_da_compra)". Exatamente deste modo, foi definido o padrão, sendo assim, agora deve-se definir os valores.

Para isso, utiliza-se o comando "values" seguido dos valores definidos entre parênteses, que ficará exatamente: "values (1, current_date, 1, 4, 1, 150);".

Os valores acima foram dados e se referem a primeira linha da tabela dada pelo exercício para inserção.

Agora, basta preencher o restante dos dados, usando novamente, o método de copiar, colar e editar nas outras linhas e ao terminar, ficará como visto na imagem 10 abaixo:

Imagem 10

```

42 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_comprada, valor_da_compra)
43 values (1, current_date, 1, 4, 1, 150);
44
45 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_comprada, valor_da_compra)
46 values (2, current_date +2, 2, 2, 2, 500);
47
48 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_comprada, valor_da_compra)
49 values (3, current_date +3, 3, 1, 2, 400);
50
51 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_comprada, valor_da_compra)
52 values (4, current_date +1, 4, 3, 1, 3500);
53
54 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_comprada, valor_da_compra)
55 values (5, current_date -2, 5, 1, 2, 400);
56
57 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_comprada, valor_da_compra)
58 values (6, current_date -5, 6, 3, 1, 3500);
59
60 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_comprada, valor_da_compra)
61 values (7, current_date -3, 5, 2, 1, 250);
62
63 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_comprada, valor_da_compra)
64 values (8, current_date -1, 4, 4, 2, 300);
65
66 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_comprada, valor_da_compra)
67 values (9, current_date, 3, 4, 3, 450);
68
69 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_comprada, valor_da_compra)
70 values (10, current_date +1, 2, 2, 4, 1000);
71
72
73

```

Um adendo ao `current_date`, ao olhar para alguns como nas linhas 55 e 58, percebe-se que há um número negativo logo após a palavra. Isso se dá devido o cliente ter feito a compra x dias antes da data atual. Ou seja, nos casos acima, o cliente fez a compra dois dias e cinco dias, respectivamente, antes da data atual programada.

Ao analisar com atenção, também percebe-se a alteração nos valores, como na linha 55, onde o último valor, que se refere a quantidade gasta, é 400. Durante a programação dos valores, me atentei ao número de produtos que o cliente comprou. Nesse caso, foram dois Mouses de R\$200,00. Por isso, durante a programação já multipliquei os valores.

Com todos os dados inseridos, foram feitas algumas consultas aleatórias, utilizando o comando `SELECT`, como solicitado no exercício. A seguir, algumas imagens com as legendas das consultas realizadas.

Imagem 11

SQL Fiddle | MySQL 5.6 | View Sample Fiddle | Clear | Text to DDL | Donate | About

```

63 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantida
64 values (8, current_date -1, 4, 4, 2, 300);
65
66 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantida
67 values (9, current_date, 3, 4, 3, 450);
68
69 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantid
70 values (10, current_date +1, 2, 2, 4, 1000);
71
72
73
74
75
76

```

```

1 select * from produto

```

id	nome_do_produto	valor	quantidade_em_estoque
1	Mouse	200	30
2	Teclado	250	15
3	Notebook	3500	10
4	Fone	150	25

Record Count: 4; Execution Time: 8ms | View Execution Plan | link

Did this query solve the problem? If so, consider donating \$5 to help make sure SQL Fiddle will be here next time you need help with a database problem. Thanks!

Nessa imagem, foi utilizado o comando “SELECT * from Produto”. Com esse comando, é possível ver as tabelas que se referem ao produto. Nesse caso, podemos ver o id, nome, valor, e quantidade em estoque.

Imagem 12

SQL Fiddle | MySQL 5.6 | View Sample Fiddle | Clear | Text to DDL | Donate | About

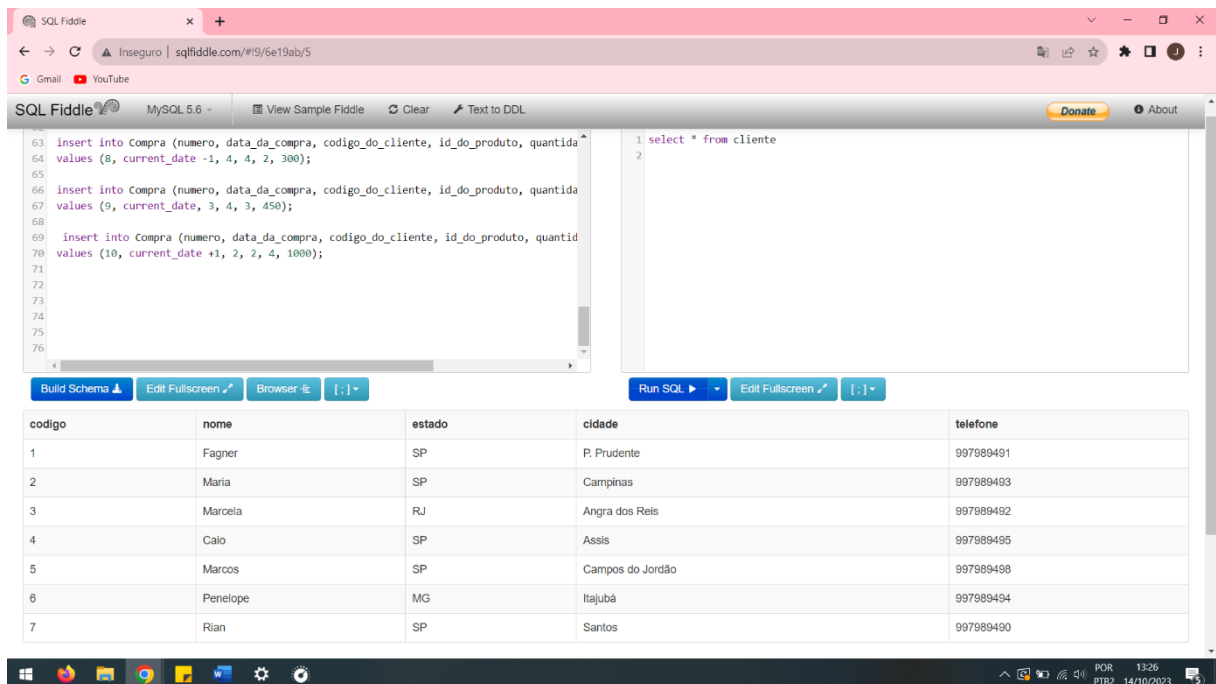
numero	data_da_compra	codigo_do_cliente	id_do_produto	quantidade_comprada	valor_da_compra
1	2023-10-14T00:00:00Z	1	4	1	150
2	2023-10-16T00:00:00Z	2	2	2	500
3	2023-10-17T00:00:00Z	3	1	2	400
4	2023-10-15T00:00:00Z	4	3	1	3500
5	2023-10-12T00:00:00Z	5	1	2	400
6	2023-10-09T00:00:00Z	6	3	1	3500
7	2023-10-11T00:00:00Z	5	2	1	250
8	2023-10-13T00:00:00Z	4	4	2	300
9	2023-10-14T00:00:00Z	3	4	3	450
10	2023-10-15T00:00:00Z	2	2	4	1000

Record Count: 10; Execution Time: 8ms | View Execution Plan | link

Did this query solve the problem? If so, consider donating \$5 to help make sure SQL Fiddle will be here next time you need help with a database problem. Thanks!

Nessa imagem, foi utilizado o comando “SELECT * from Compras”. Com esse comando, é possível ver a tabela Compra e seus dados. Nota-se que as datas estão atualizadas e batem com as compras realizadas na data atual e antes ou depois da data atual.

Imagem 13



Nessa imagem, foi utilizado o comando “SELECT * from Cliente”. Com esse comando, é possível ver a tabela cliente. Nesse caso, com todos os atributos preenchidos e já organizado pelo número, do primeiro cliente ao sétimo.

Com as etapas 1 e 2 cumpridas e as tabelas preenchidas e funcionais, resta apenas a ultima etapa do exercício, onde a proposta agora é treinar as junções Join.

Antes de pratica-las, é importante lembrar do que se trata. As funções Join no SQL, de maneira ilustrada, são exatamente como conjuntos matemáticos.

Os comandos que serão utilizados no terceiro exercício são o left join e inner join.

Imagine um conjunto matemático, onde há dois círculos interligados, o da direita e da esquerda. Como o próprio nome “left” diz, esse join, especificamente irá juntar, desses dois círculos, tudo que está à esquerda e dentro da interceção, enquanto tudo que está à direita não será mesclado. Em resumo, o left join retorna todos os casos que estao a tabela esquerda e todos que fizerem par com a direita. Se alguém da esquerda não possuir par, ele retornará como null, como veremos no exercício a seguir.

O Inner Join, por outro lado é relativamente mais simples. Ele é usado apenas para selecionar as interseções dos conjuntos, ou seja, o comando mesclará

cada uma das tabelas e vai retornar apenas o que existem em todos os lados. Todos em comum retornarão, enquanto que o restante não aparecerá.

Para começar a terceira etapa do exercício, em seu primeiro tópico, o enunciado pede que, utilizando os comandos join, utilizemos um comando SELECT que retorne o número da venda, nome do cliente, nome do produto e valor da compra, ordenando a saída pelo número da compra.

No entanto, é importante salientar que durante essa resolução, é possível verificar, que o enunciado pede que faça uma consulta que retorne ao número da venda e da compra, além dos outros atributos citados. Porém, o exercício não especificou que era para realizar a programação de uma tabela de vendas, visto que nas etapas anteriores não há nenhum atributo "vendas" em nenhuma das entidades, portanto, será seguido o exercício normalmente, desconsiderando então, o "número da venda" do enunciado.

Neste primeiro tópico, será usado o comando INNER JOIN, para que ele mescle as tabelas e mostra apenas os atributos citados. O enunciado também pede que mostre a tabela ordenada pelo número da venda.

Para isso, o comando completo usado será “select numero, nome, nome_do_produto, valor_da_compra from Cliente inner join Compra on codigo = codigo_do_cliente

```
inner join Produto on id_do_produto = id
order by numero asc”.
```

Percebe-se que a função inner join está relacionada a tabela Compra. Usa-se também o comando on, para citar o código e com o “=”, é possível programar para igualar ao código do cliente, para depois usarmos inner join.

Por fim, para atender a todas as solicitações e ordenar os números de id, usa-se o comando “order by” relacionando o Produto onde o id_do_produto será o id e para isso, utiliza-se novamente o “=” para correlaciona-los.

Ao certificar-se que não há nenhum erro de sintaxe ou digitação, então basta pressionar o “Run SQL” e ficará exatamente como na imagem abaixo.

The screenshot shows the SQL Fiddle interface with the following SQL queries:

```

63 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade)
64 values (8, current_date -1, 4, 4, 2, 300);
65
66 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade)
67 values (9, current_date, 3, 4, 3, 450);
68
69 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade)
70 values (10, current_date +1, 2, 2, 4, 1000);
71
72
73
74
75
76

```

The result table shows the following data:

numero	nome	nome_do_produto	valor_da_compra
1	Fagner	Fone	150
2	Maria	Teclado	500
3	Marcela	Mouse	400
4	Caio	Notebook	3500
5	Marcos	Mouse	400
6	Penelope	Notebook	3500
7	Marcos	Teclado	250
8	Caio	Fone	300
9	Marcela	Fone	450
10	Maria	Teclado	1000

Record Count: 10; Execution Time: 2ms

Esta imagem é a junção de dois prints, pois não foi possível capturar ao mesmo tempo, tanto a programação, quanto a tabela completa. Por esse motivo, o print foi tirado até p 7º nome e, após isso, tirado novamente um print, porem com o cursor mais embaixo, para que possibilite ver o restante dos 3 nomes. Ambos foram tirados exatamente na mesma hora.

Seguindo para o próximo tópico, o enunciado pede para que seja dado um SELECT que demonstre o nome do(s) cliente(s), e o código da compra, onde o código deverá ser preenchido como “NULL” para os clientes que não compraram na loja.

Para este exercício, como foi visto acima, será usado o Left Join, pois ele, além de separar os pares, também é o responsável por preencher automaticamente como null, os dados que não se correlacionam entre tabelas.

Portanto, seguindo as instruções do enunciado, primeiro dá-se o comando “select nome”, em seguida, buscamos com “numero from cliente left join compra” fazendo a coligação com o comando “on” para “codigo = codigo_do_cliente;”.

O resultado é a tabela já mesclada com o cliente que não comprou na loja já associado como “null”, como pode-se ver na imagem abaixo.

Imagem 14

The screenshot shows the SQL Fiddle interface with the following SQL query:

```

1 select nome, numero from cliente left join compra on codigo = codigo_do_cliente;
2
3
4

```

The results table displays the following data:

nome	numero
Fagner	1
Maria	2
Maria	10
Marcela	3
Marcela	9
Caio	4
Caio	8
Marcos	5
Penelope	6
Rian	(null)

Record Count: 11; Execution Time: 1ms

Assim como a Imagem 13, esta imagem é a junção de dois prints, pois não foi possível capturar ao mesmo tempo, tanto a programação, quanto a tabela completa. Por esse motivo, o print foi tirado até o nome Marcos e, após isso, tirado novamente um print, porém com o cursor mais embaixo, para que possibilite ver o restante dos 2 nomes. Ambos foram tirados exatamente na mesma hora.

Nota-se na imagem acima, que diferente dos outros clientes, o cliente Rian não efetuou nenhuma compra, mas com este comando, a o sistema consegue mostra-lo, preenchendo seu lugar na coluna “numero”, como “null”, assim como foi solicitado no exercício.

É importante salientar a importância do comando left join e de colocar a tabela associada à esquerda, pois caso a mesma estivesse na direita, o comando não reconheceria e não daria o resultado programado, pois como já explicado, o Left Join foca na chamada “Left Table”, que traduzida, significa tabela esquerda. Por esse motivo, é tão importante atentar-se no lado e comando correto.

Com a resolução dos dois primeiros tópicos, agora basta finalizar a tarefa com o último.

De acordo com o enunciado, a solicitação agora é uma consulta com o comando SELECT com o nome do cliente (sem que se repitam, caso o cliente tenha feito mais de uma compra), e o telefone, dos clientes que tenham comprado na loja antes da data atual.

Para esta tarefa, o comando que será utilizado é o termo SELECT DISTINCT, que tem como função, especificamente, distinguir os atributos de acordo com a logica "if...where", ou seja, é feito o comando, definida a proposição e executado, para achar o resultado na tabela que foi solicitado no enunciado, que nesse caso é não haver repetições dos clientes que fizeram mais de uma compra e o telefone dos clientes que compraram com antecedência.

Para mostrar essa tabela, o comando começa com "select distinct nome, telefone" para que o comando entenda que o selecionado para consulta é o nome e telefone, atentando-se na vírgula.

Assim segue-se com "from cliente inner join compra on codigo = codigo_do_cliente", que com o comando "from" e "inner join" seguido do símbolo "=", consegue-se especificar o código da compra e o código do cliente são semelhantes.

Então, completa-se com o comando where, seguindo a lógica de programação condicional para que a data da compra seja menor que a data atual, onde para isso, utiliza-se o conectivo "<" (where data_da_compra < current_date).

Após se certificar que está tudo escrito da maneira correta, clique em Run SQL e terá a tabela pronta seguindo as ordens do enunciado, como pode-se ver na imagem abaixo.

Imagem 15

The screenshot shows the SQL Fiddle interface with the following SQL code in the left pane:

```
37 insert into Produto (nome_do_produto, valor, quantidade_em_estoque)
38 values ("Notebook", 3500, 10);
39 insert into Produto (nome_do_produto, valor, quantidade_em_estoque)
40 values ("Fone", 150, 25);
41
42 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_compra)
43 values (1, current_date, 1, 4, 1, 150);
44
45 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_compra)
46 values (2, current_date + 2, 2, 2, 2, 500);
47
48 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_compra)
49 values (3, current_date + 3, 3, 1, 2, 400);
50
51 insert into Compra (numero, data_da_compra, codigo_do_cliente, id_do_produto, quantidade_compra)
52 values (4, current_date + 1, 4, 3, 1, 3500);
53
54 insert into Compra (numero, data da compra, codigo do cliente, id do produto, quantidade comprada)
55 values (5, current_date + 4, 5, 2, 1, 100);
```

The right pane contains the following query:

```
1 select distinct nome, telefone from cliente inner join compra on codigo = codigo_do_cliente
2 where data_da_compra < current_date;
```

The results pane shows the following table:

nome	telefone
Caio	997089495
Marcos	997969496
Penelope	997089494

Record Count: 3, Execution Time: 1ms. View Execution Plan link.

Did this query solve the problem? If so, consider donating \$5 to help make sure SQL Fiddle will be here next time you need help with a database problem. Thank!

Improve Entity Framework Bulk Insert Bulk Delete LEARN MORE

Como visto na imagem acima, a tabela já sai pronta, sem repetir nenhum nome e os telefones dos clientes que fizeram a compra antes da data atual.

3 CONCLUSÃO

Com isso, todas as tarefas dadas pelos problemas propostos foram atendidas.

Primeiramente, atribuímos todas as tabelas das três entidades que nos foi dada, sendo elas Cliente, Produto e Compras. As tabelas foram criadas e com isso, foram definidos seus atributos, como solicitado no exercício.

Após a definição dos atributos e programação de todas as três tabelas, com toda atenção direcionada para o modo no qual escrevemos e as regras que devemos seguir.

Então, definimos as chaves, primárias e estrangeiras para os devidos atributos aos quais o exercício exigiu.

Com as tabelas prontas, bastou realizar a inserção de dados e após isso, realizar as consultas usando os comandos SELECT e JOIN.

O exercício foi de extrema ajuda para que eu conseguisse entender na prática a programação. Houveram momentos difíceis, erros de sintaxe, erros de digitação, mas este é o lado bom, pois com nossos erros no exercício prático, nós aprendemos.

4 REFERÊNCIAS

Ramakrishnan Raghu, Sistemas de Gerenciamento de Banco de Dados, McGrawHill, 2008.

Sistema Utilizado: SQL Fiddle.

