ESCOLA SUPERIOR DE TECNOLOGIA - IPS
ANO LETIVO 2015 / 2016

ENGENHARIA DE INFORMÁTICA
BASES DE DADOS
PROFº CLÁUDIO SAPATEIRO

DESENVOLVIMENTO DE SISTEMA DE SOFTWARE POS – ESTAÇÃO DE SERVIÇO

MIGUEL FURTADO 120221006
PEDRO RIBEIRO 140221043

Índice

Introdução

A pedido do docente Cláudio Sapateiro para a unidade curricular de Base de Dados, de engenharia informática do Instituto Politécnico de Setúbal da Escola Superior de Tecnologias, irá ser desenvolvido um modelo de dados de suporte a um sistema de software Point-Of-Sale.

Os objetivos do desenvolvimento deste é aplicar os conhecimentos obtidos ao longo do semestre, e desenvolver capacidades para que no futuro seja possível aplicar estes sem dificuldades.

O tema escolhido irá ser o desenvolvimento de um sistema de software POS para uma estação de serviço, que visa integrar todos os aspetos que inclui a faturação do serviço para o cliente, gestão de stock, registo de clientes, fornecedores, colaboradores, volume de vendas por parte da estação de serviço entre outros.

Dado isto espera-se que o desenvolvimento deste tema seja um sucesso e que todas as dificuldades encontradas sejam superadas.

POS- O que é?

POS é uma sigla que significa Point-Of-Sale ou em português ponto de venda, é o momento e lugar onde é efetuada uma transação. É o ponto em que o cliente faz o pagamento ao vendedor em troca de bens ou um serviço. No ponto de venda, o vendedor irá preparar a fatura para o cliente ou calcular o valor a pagar pelo cliente. Neste ponto são oferecidas algumas opções aos clientes de como pretendem pagar. Após o pagamento é emitido o recibo de compra.

Um sistema POS, é um sistema usado para melhor servir as necessidades do vendedor, com hardware e software customizado. Este podem incluir scanners, ecrãs touch, registadores eletrónicas ou manuais, terminais EFTPOS, entre outros. Diferentes tipos de negócios irão ter sistemas POS diferentes.

O POS também é referido como o ponto de serviço pois não é apenas um ponto de venda, mas também um ponto de retorno.

Alguns dos requisitos gerais da informação gerida por um sistema de software POS, podem passar por ter o valor dos artigos a vender, informações sobre os colaboradores da empresa onde o software está inserido, média das vendas. Algumas características de um sistema de software POS de uma estação de serviço, passariam por incluir, os dados do cliente, sendo que estes incluiriam os pontos pessoais do mesmo, outras caraterísticas seriam, acesso aos preçários dos combustíveis, média dos combustíveis vendidos diariamente, semanalmente, mensalmente, inventário sobre todos os artigos da loja.

Alguns dos softwares existentes são:

- WinRest
- Magnifinance
- InvoiceExpress

Domínio da Aplicação dos POS

O tema que o nosso grupo escolheu foi o desenvolvimento de um sistema de software POS para uma estação de serviço, apenas para o serviço de abastecimento de combustível, gestão de clientes, gestão de colaboradores e gestão de stocks. O nome escolhido para o tema é GasLineExpress.

Alguns dos requisitos necessários para a implementação do POS:

- Efetuar registo de clientes
- Efetuar registo de colaboradores
- Efetuar registo de fornecedores
- Efetuar venda de combustíveis
- Efetuar cálculo para recibos
- Gerir pontos de clientes
- Informação sobre os combustíveis armazenados
- Gestão de stock de combustíveis
- Efetuar registo de uma oferta

- Categorizar oferta

Com esta informação irá ser possível, criar e visualizar uma ficha de clientes, colaboradores e dos diversos fornecedores existentes e ainda alterar informações destes caso seja necessário. Será possível calcular a média de vendas de combustíveis, verificar o stock existente, saber mais sobre os clientes e sobre as promoções existentes que podem ser oferecidas a estes, conseguir identificar as necessidade de stock e que fornecedores são precisos contatar para repor o stock. Conseguir identificar o tipo de cliente existente e relacionar a quantidade monetária que este despende em cada abastecimento. Perceber quais são os combustíveis de maior rotação. Conseguir gerir as ofertas aos clientes com um sistema de pontos. Conseguir realizar todos os cálculos necessários para uma venda. Categorizar os diferentes tipos de combustíveis. Permitir efetuar uma gestão das ofertas existentes que podem ser oferecidas ao cliente.

1ª Fase

Identificação das Entidades

- **Cliente** (cli_id, cli_nome, cli_data_registo, cli_telefone, cli_localidade, cli_pontos, cli_empresa, cli_nif, cli_data_nascimento, cli_rua, cli_nr, cli_andar, cli_porta, cli_codigo_postal, cli_estado)
    - A entidade cliente, foi escolhida para conseguirmos categorizar um dos tipos de pessoas existentes e para que seja facilitado o acesso à informação apenas do cliente. A **chave primária** desta entidade é o cli_id.

- **Colaborador** (col_id, col_nome, col_data_registo, col_telefone, col_localidade, col_rua, col_nr, col_andar, col_porta, col_codigo_postal, col_data_nascimento, col_estado);
    - A entidade colaborador, foi escolhida para conseguirmos categorizar um dos tipos de pessoas existentes e para que seja facilitado o acesso à informação apenas do colaborador. A **chave primária** desta entidade é o col_id.

- **Fornecedor (**for_id, for_nome, for_data_registo, for_telefone, for_localidade, for_empresa, for_rua, for_nr, for_andar, for_porta, for_codigo_postal, for_estado);
    - A entidade fornecedora, tal como a cliente e colaborador, tem generalização da entidade Pessoa. Foi feita uma entidade única para o fornecedor pois este irá trazer outros atributos novos que serão relacionados. A **chave primária** desta entidade é o for_id.

- **Combustivel** (co_id, co_custo_litro, co_nome, co_data_inicio, co_data_fim);
    - A entidade combustível foi criada com o intuito de se identificar facilmente o produto que irá ser vendido na estação de serviço. Esta tabela irá estar associado ao fornecedor e também à entidade vendas. A **chave primária** desta entidade é o co_id.

- **Vendas** (ve_id, ve_id_combustivel, ve_taxa_iva, ve_litros, ve_custo, ve_data_venda);
    - A entidade vendas foi criada para ser possível registar todas as vendas ocorridas e mais tarde relacionar, por exemplo, o volume de vendas efetuado num determinado espaço de tempo. A **chave primária** desta entidade é o ve_id que identifica a venda.

- **Ofertas** (of_id, of_nome, of_pontos, of_data_inicio, of_data_fim, of_descricao, of_categoria, of_numero_ofertas, of_litros);
    - A entidade ofertas, é a entidade que vai servir para mostrar as ofertas existentes das quais os clientes podem usufruir, caso tenham os pontos necessários para a adquirir. A **chave primária** é o of_id.

- **Faturacao** (fa_id, fa_custo_total, fa_data, fa_volume_vendas, fa_iva);
    - A entidade faturação foi criada para que haja apenas uma tabela onde se guarda a informação final após se efetuar uma venda. Esta vai servir para obter informação sobre volume de vendas, entre outros dados para se enviar à autoridade tributária. A **chave primária** é o fa_id para se identificar a tabela.

Relacionamentos

- **Fornecedor-Combustível:**
  Cardinalidade: N:N; **Participação:** total;
  Combustível-Fornecedor:
  Cardinalidade: 1:N; **Participação:** total;
    - Esta relação diz-nos que vários fornecedores têm que fornecer 1 ou mais combustíveis e que um combustível pode ter 1 ou mais fornecedores.

- **Colaborador-Vendas:**
  Cardinalidade: 1:N; **Participação:** total;
  Vendas-Colaborador:
  Cardinalidade: N:N; **Participação:** total;
    - Esta relação indica-nos que n colaboradores podem fazer n vendas e vice-versa a participação é total pois só o colaborador pode fazer vendas. Esta relação permite-nos identificar quantas vendes têm sido feitas por cada colaborador individualmente.

- **Combustível-Vendas:**
  Cardinalidade: 1:N; **Participação:** total;
  Vendas-Combustível:
  Cardinalidade: N:1; **Participação:** total;

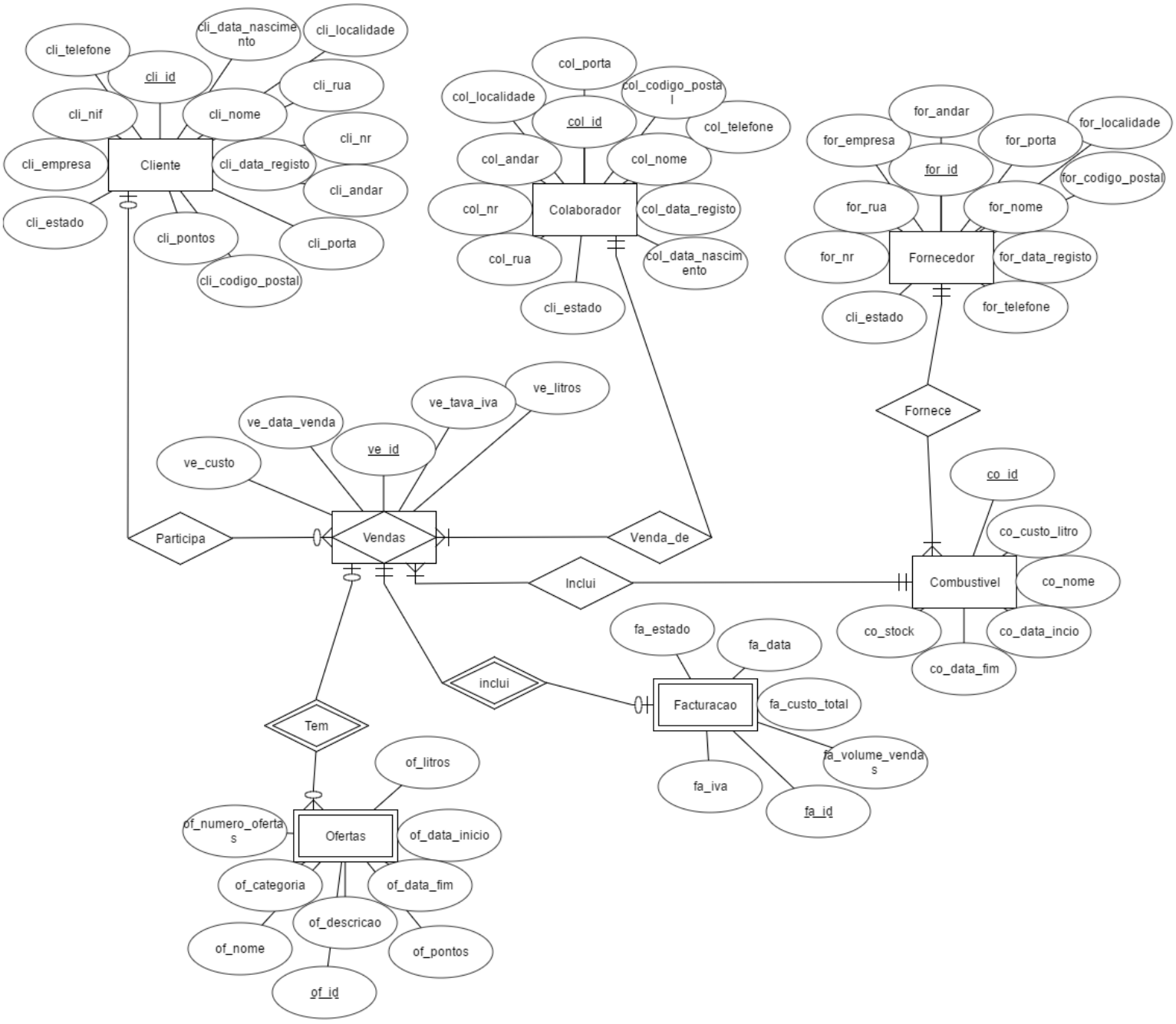
- Esta relação foi criada pois nas vendas é preciso ter o combustível que se vai vender. Tendo isto em conta cada combustível pode participar em n vendas e n vendas podem ter apenas um combustível. Em ambas a participação é total.

- **Cliente-Vendas:**
  Cardinalidade: 1:N; **Participação**: parcial;
  Vendas-Cliente:
  Cardinalidade: 1:1 **Participação:** parcial;
  - Esta relação foi criada pois um cliente pode participar em n vendas, mas uma venda só pode ter um cliente. Em ambas a participação é total.

- **Vendas-Ofertas:**
  Cardinalidade: 1-N; **Participação:** Parcial;
  Ofertas-Vendas:
  Cardinalidade: N:N; **Participação:** Parcial;
  - Esta relação foi criada pois um venda pode ter 1 ou n ofertas e n ofertas pode participar em n vendas. Em ambas a participação é parcial.

- **Vendas-Faturacao**
  Cardinalidade: N:N; **Participação:** Parcial;
  Faturacao-Vendas:
  Cardinalidade: N:N; **Participação:** Total;
  - Esta relação foi criada pois n vendas tem n linhas de faturação e n linhas de faturação podem ter n vendas. Em ambas a participação é total.

Diagrama Entidade Relação



Modelo Relacional

Cliente (cli_id INTEGER PRIMARY KEY, cli_nome VARCHAR (20), cli_data_registo DATE, cli_telefone INTEGER, cli_localidade VARCHAR (25), cli_pontos INTEGER, cli_empresa VARCHAR (20), cli_nif INTEGER, cli_data_nascimento DATE, cli_rua VARCHAR (25), cli_nr INTEGER, cli_andar VARCHAR (10), cli_porta VARCHAR (10), cli_codigo_postal INTEGER, cli_estado VARCHAR(20));

Colaborador (col_id INTEGER PRIMARY KEY, col_nome VARCHAR (20), col_data_registo DATE, col_telefone INTEGER, col_localidade VARCHAR (25), col_rua VARCHAR (25), col_nr INTEGER, col_andar VARCHAR (10), col_porta VARCHAR (10), col_codigo_postal INTEGER, col_data_nascimento DATE, col_estado VARCHAR(20));

Fornecedor (for_id INTEGER PRIMARY KEY, for_nome VARCHAR (20), for_data_registo DATE, for_telefone INTEGER, for_localidade VARCHAR (25), for_empresa VARCHAR (20), for_rua VARCHAR (25), for_nr INTEGER, for_andar VARCHAR (10), for_porta VARCHAR (10), for_codigo_postal INTEGER, for_estado VARCHAR(20));

Combustivel (co_id INTEGER PRIMARY KEY, co_custo_litro DOUBLE (16,4), co_nome VARCHAR (20), co_data_inicio DATE, co_data_fim DATE co_stock INTEGER, co_fornecedor_id INTEGER FOREIGN KEY);

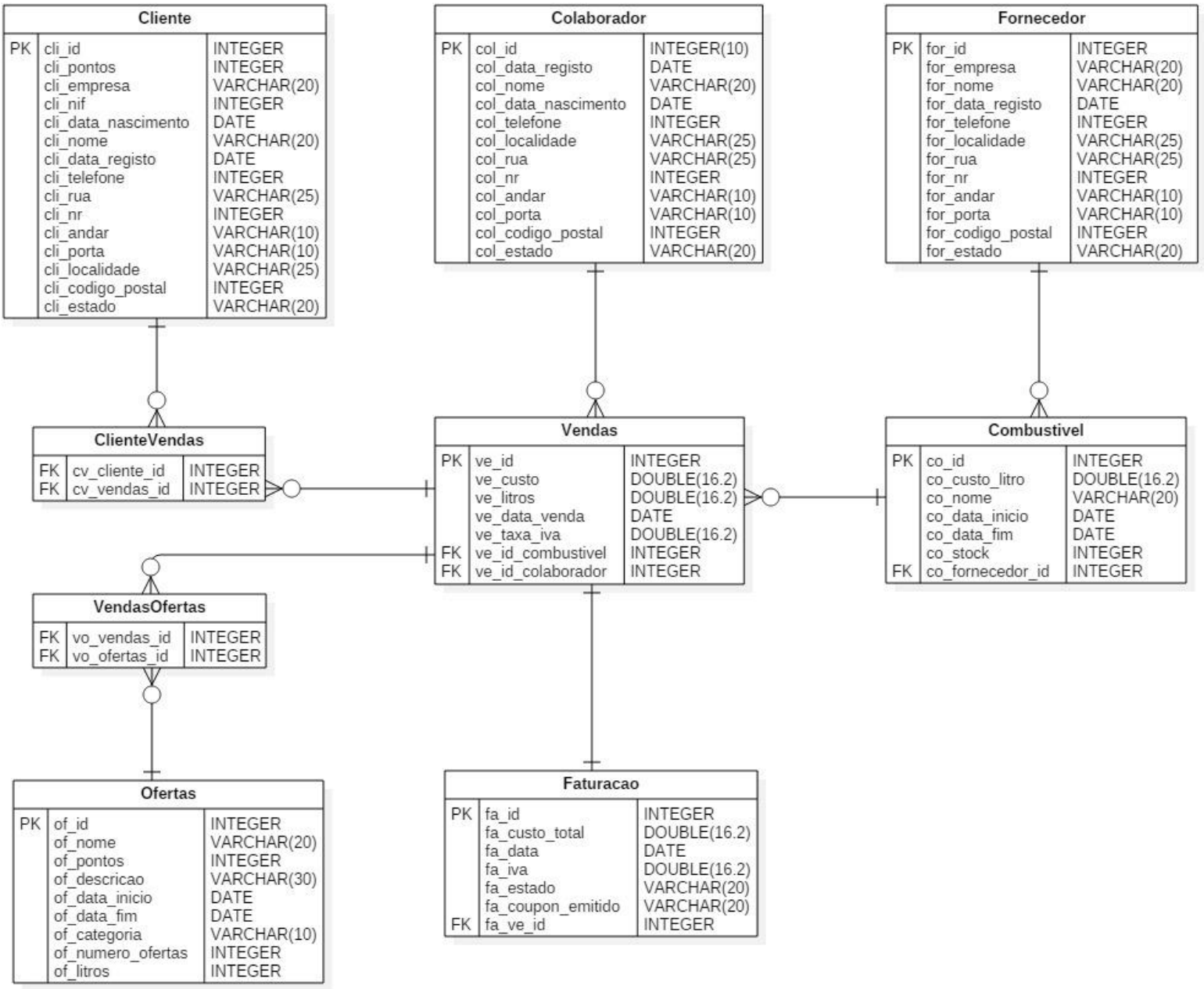
Ofertas (of_id INTEGER PRIMARY KEY, of_nome VARCHAR (20), of_pontos INTEGER, of_data_inicio DATE, of_data_fim DATE, of_descricao VARCHAR (30), of_categoria VARCHAR (20), of_numero_ofertas INTEGER, of_litros DOUBLE);

Vendas (ve_id INTEGER PRIMARY KEY, ve_id_combustivel INTEGER FOREIGN KEY, ve_taxa_iva DOUBLE (16,4), ve_litros DOUBLE (16,4), ve_custo DOUBLE (16,4), ve_data_venda DATE, ve_id_colaborador INTEGER FOREIGN KEY);

Faturacao (fa_id INTEGER PRIMARY KEY, fa_custo_total DOUBLE (16,4), fa_data DATE, fa_volume_vendas DOUBLE (16,4), fa_iva DOUBLE (16,4), fa_ve_id INT FOREING KEY);

ClienteVendas (cv_cliente_id INTEGER FOREIGN KEY, cv_vendas_id INTEGER FOREIGN KEY);

VendasOfertas (vo_vendas_id INTEGER FOREIGN KEY, vo_ofertas_id INTEGER FOREIGN KEY);


Cliente
PK cli_id INTEGER
cli_pontos INTEGER
cli_empresa VARCHAR(20)
cli_nif INTEGER
cli_data_nascimento DATE
cli_nome VARCHAR(20)
cli_data_registo DATE
cli_telefone INTEGER
cli_rua VARCHAR(25)
cli_nr INTEGER
cli_andar VARCHAR(10)
cli_porta VARCHAR(10)
cli_localidade VARCHAR(25)
cli_codigo_postal INTEGER
cli_estado VARCHAR(20)
Colaborador
PK col_id INTEGER(10)
col_data_registo DATE
col_nome VARCHAR(20)
col_data_nascimento DATE
col_telefone INTEGER
col_localidade VARCHAR(25)
col_rua VARCHAR(25)
col_nr INTEGER
col_andar VARCHAR(10)
col_porta VARCHAR(10)
col_codigo_postal INTEGER
col_estado VARCHAR(20)
Fornecedor
PK for_id INTEGER
for_empresa VARCHAR(20)
for_nome VARCHAR(20)
for_data_registo DATE
for_telefone INTEGER
for_localidade VARCHAR(25)
for_rua VARCHAR(25)
for_nr INTEGER
for_andar VARCHAR(10)
for_porta VARCHAR(10)
for_codigo_postal INTEGER
for_estado VARCHAR(20)
ClienteVendas
FK cv_cliente_id INTEGER
FK cv_vendas_id INTEGER
Vendas
PK ve_id INTEGER
ve_custo DOUBLE(16.2)
ve_litros DOUBLE(16.2)
ve_data_venda DATE
ve_taxa_iva DOUBLE(16.2)
FK ve_id_combustivel INTEGER
FK ve_id_colaborador INTEGER
Combustivel
PK co_id INTEGER
co_custo_litro DOUBLE(16.2)
co_nome VARCHAR(20)
co_data_inicio DATE
co_data_fim DATE
co_stock INTEGER
FK co_fornecedor_id INTEGER
VendasOfertas
FK vo_vendas_id INTEGER
FK vo_ofertas_id INTEGER
Ofertas
PK of_id INTEGER
of_nome VARCHAR(20)
of_pontos INTEGER
of_descricao VARCHAR(30)
of_data_inicio DATE
of_data_fim DATE
of_categoria VARCHAR(10)
of_numero_ofertas INTEGER
of_litros INTEGER
Faturacao
PK fa_id INTEGER
fa_custo_total DOUBLE(16.2)
fa_data DATE
fa_iva DOUBLE(16.2)
fa_estado VARCHAR(20)
fa_coupon_emitido VARCHAR(20)
FK fa_ve_id INTEGER

Exemplo de Informação a conter na Base de Dados

Tabela Clientes

cli_id	cli_nome	cli_data_registo	cli_telefone	cli_localidade	cli_pontos	cli_empresa	cli_nif	cli_data_nascimento	cli_rua	cli_nr	cli_andar	cli_porta	cli_codigo_postal
1	Jonas	2016-01-01	915152789	Lisboa	0	Sadin	24587915	1992-05-12	Rua Alves Santos	5	1º	A	2500
2	Rui	2015-05-12	916243897	Setubal	245	TerraCotaa	45135798	1989-03-20	Rua Jacinto Andrade	7	NULL	NULL	2900
3	Andre	2016-03-11	911892543	Lisboa	458	Landa	29874561	1995-06-04	Rua Moreira OL	10	2º	B	5500
4	Lopes	2015-07-20	963879534	Almada	0	Grouje	24587915	1993-10-25	Avenida Europa	4	3º	C	2910
5	Paulo	2014-12-01	934587326	Cascais	78	Kool	13458794	1990-05-07	Avenida Portal	5	NULL	NULL	5470
6	Pedro	2013-11-20	925554123	Faro	90	TicTravel	154222	1988-11-20	Avenida Trip	6	NULL	NULL	451
7	Jose	2016-01-11	910000458	Porto	545	Moveit	14587988	1995-10-25	Rua Jose DelWhiskey	41	NULL	NULL	2541
8	Diana	2014-02-21	920125487	Seixal	1425	JabbaTaxi	15548798	1980-10-05	Rua do Cavaco	7	NULL	NULL	2840
9	Miguel	2015-03-25	920111523	Corroios	5000	DingDong	14457821	1996-05-14	Praceta Afonso Henriques	8	NULL	NULL	2694
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

A tabela clientes vai permitir-nos guardar todos os dados relativamente aos clientes, os campos cli_andar e cli_porta encontram-se a null, pois alguns clientes podem não ter um andar ou porta na morada. Mais tarde quando a inserção de dados for com um stored procedured isto já não ficara a null.

Tabela Colaboradores

col_id	col_nome	col_data_registo	col_telefone	col_localidade	col_rua	col_nr	col_andar	col_porta	col_codigo_postal	col_data_nascimento
1	Kore	2015-03-14	91458752	Setubal	Rua Super Homem	23	5º	G	2900	1990-05-17
2	Yann	2016-02-03	934516879	Montijo	Avenida Plural	5	NULL	NULL	4500	1985-10-16
3	Karlos	2014-10-22	924571284	Almada	Rua dos Peixes	8	1º	Esquerdo	5400	1991-05-14
4	Lara	2015-12-20	914758234	Setubal	Rua Destruida	1	NULL	NULL	2910	1993-04-07
5	Mario	2016-04-01	916289341	Lisboa	Avenida 5 Espanha	5	4º	A	3550	1991-01-08
6	Carlos	2014-05-06	921458221	Setubal	Avenida Peixinho Dourado	8	2º	B	2456	1992-02-12
7	Cecilia	2014-11-15	921114589	Seixal	Rua D.Dinis	5	NULL	NULL	2458	1991-01-31
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

A tabela colaboradores vai permitir-nos guardar todos os dados relativamente aos colaboradores, os campos cli_andar e cli_porta encontram-se a null, pois alguns colaboradores podem não ter um andar ou porta na morada. Mais tarde quando a inserção de dados for com um stored procedured isto já não ficara a null.

Tabela Fornecedores

for_id	for_nome	for_data_registo	for_telefone	for_localidade	for_empresa	for_rua	for_nr	for_andar	for_porta	for_codigo_postal
1	Luis	2010-05-18	916458237	Setubal	CombusAll	Rua dos combustiveis	10	NULL	NULL	3150
2	Pedro	2012-12-04	935761592	Almada	EnergiasVivas	Rua Sofle	2	NULL	NULL	4500
3	Torin	2011-07-30	916732851	Setubal	LigaLeve	Rua Joaquem Almeida	5	2º	C	5500
4	Holha	2015-05-08	963482516	Almada	SabesCombustivel	Avenida Mal Me Quer	6	3º	D	3690
5	Hefe	2015-08-06	964829537	Montijo	MundoVerde	Avenida Qualquer	8	NULL	NULL	2350
6	Ze	2010-11-24	924587112	Amora	DieselLife	Rua Zeca Afonso	8	NULL	NULL	2451
7	Diogo	2009-03-22	914567852	Sesimbra	TreeHug	Rua Mane	1	NULL	NULL	2548
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

A tabela fornecedores vai permitir-nos guardar todos os dados relativamente aos fornecedores, os campos cli_andar e cli_porta encontram-se a null, pois alguns fornecedores podem não ter um andar ou porta na morada. Mais tarde quando a inserção de dados for com um stored procedured isto já não ficara a null.

Tabela Combustiveis

co_id	co_custo_litro	co_nome	co_data_inicio	co_data_fim	co_fornecedor_id	co_stock
1	1.24	Gasolina 95	2016-04-10	NULL	1	400
2	1.00	Gasoleo	2016-03-20	NULL	2	200
3	1.20	Gasolina 95	2016-03-15	2016-04-10	1	0
4	1.40	Gasolina 98	2016-03-20	NULL	3	50
NULL	NULL	NULL	NULL	NULL	NULL	NULL

A tabela combustível vai permitir-nos guardar todos os dados relativamente aos combustiveis, o campo co_data_fim é null pois significa que o combustível ainda está em vigor. Quando um combustível acaba ou é inserido um novo combustível do mesmo tipo, ou seja, mesmo nome, a data_fim é mudade para a data em que é registado o novo combustível na tabela.

Tabela Ofertas

of_id	of_nome	of_pontos	of_data_inicio	of_data_fim	of_descricao	of_categoria	of_numero_ofertas
1	Bicicleta 110	5000	2016-01-05	2016-05-20	Bicicleta todo terreno	Material	20
2	Apito	100	2016-03-20	2016-06-20	Faz barulho	Material	100
3	5 Litros	300	2016-01-01	2016-05-01	Gasolina 95 5 Litros	Descontos	NULL
4	10 Litros	800	2016-02-01	2016-06-01	Gasoleo 10 Litros	Descontos	NULL
5	10 Litros	900	2016-01-15	2016-06-01	Gasolina 95 10 Litros	Descontos	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

A tabela ofertas guarda toda a informação relativamente às ofertas, caso as ofertas sejam do tipo descontos, como não tem número de ofertas este campo permanecerá a null.

Tabela Vendas

ve_id	ve_id_combustivel	ve_taxa_iva	ve_litros	ve_custo	ve_data_venda	ve_id_colaborador
1	1	0.35	15.00	18.60	2016-04-10	1
2	2	0.15	15.00	NULL	2016-03-20	2
3	4	0.35	NULL	10.00	2016-03-20	3
4	1	0.35	20.00	NULL	2016-04-10	1
5	1	0.35	25.00	NULL	2046-03-21	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL

A tabela vendas guarda toda a informação sobre as vendas. Os campos ve_litros e ve_custo encontram-se a null pois neste momento toda a informação foi introduzida com um insert. Mais tarde será inserida com um stored procedures que irá calcular estes valores e assim já não irá haver nulls.

Tabela Faturacao

fa_id	fa_custo_total	fa_data	fa_iva	fa_ve_id
1	25.11	2016-04-10	6.45	1
2	17.22	2016-03-20	2.24	2
3	10.00	2016-03-20	3.50	3
NULL	NULL	NULL	NULL	NULL

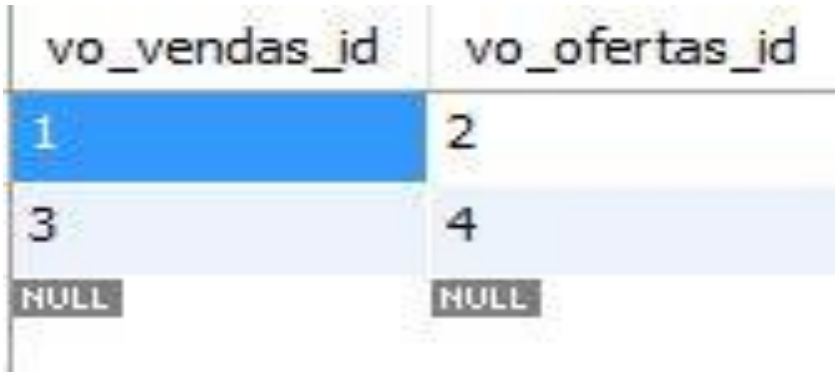
A tabela faturacao irá guardar toda a informação sobre as faturas.

Tabela ClienteVendas

cv_cliente_id	cv_vendas_id
1	1
2	2
3	3
4	4
NULL	NULL

A tabela clienteVendas é muito importante pois com esta é que será possível identificar quando um cliente está associado a uma venda.

Tabela VendasOfertas



vo_vendas_id	vo_ofertas_id
1	2
3	4
NULL	NULL

A tabela vendasOfertas será para associar uma venda a uma oferta.

Agregação de Informação para Análise

Vistas que Permitirão Comunicar à AT

```
CREATE OR REPLACE VIEW AutoridadeTributariaLoja AS
SELECT SUM(fa_custo_total) 'Lucro Com Iva', MONTHNAME(fa_data) Mes, SUM(fa_iva) 'Total
Iva', SUM(fa_custo_total - fa_iva) 'Lucro Sem Iva'
FROM Faturacao
WHERE YEAR(fa_data) = YEAR(CURRENT_DATE)
AND MONTH(fa_data) = MONTH(CURRENT_DATE - INTERVAL 1 MONTH);


CREATE OR REPLACE VIEW AutoridadeTributariaCliente AS
SELECT cli_nome 'Cliente Nome', fa_iva 'Iva Pago', (fa_custo_total - fa_iva) 'Custos Sem Iva',
fa_custo_total 'Custo Com Iva', MONTHNAME(fa_data) Mes
FROM Cliente
INNER JOIN ClienteVendas
ON cli_id = cv_cliente_id
INNER JOIN Vendas
ON cv_vendas_id = ve_id
INNER JOIN Faturacao
ON ve_id = fa_ve_id
WHERE YEAR(fa_data) = YEAR(CURRENT_DATE)
AND MONTH(fa_data) = MONTH(CURRENT_DATE - INTERVAL 1 MONTH);
```

Itens Necessários para Análise em Benefício do Negócio

- Volume de vendas
- Número de ofertas adquiridas pelos clientes
- Ofertas mais vendidas por categoria
- Oferta mais procurada (Top 5)
- Combustível mais vendido

- Combustível menos vendido
- Colaborador que efetuou mais vendas
- Cliente registado que fez mais compras
- Número de vendas sem cliente registado
- Número de vendas com cliente registado

View Volume de Vendas

```
CREATE OR REPLACE VIEW VolumeDeVendas AS
SELECT COUNT(ve_id) 'Numero de Vendas', SUM(ve_custo) 'Lucro', SUM(ve_litros) 'Litros
Vendidos', MONTHNAME(ve_data_venda) Mes
FROM vendas
GROUP BY YEAR(ve_data_venda) = YEAR(CURRENT_DATE )
AND MONTH(ve_data_venda) = MONTH(CURRENT_DATE - INTERVAL 1 MONTH);
```

View Número de Ofertas Adquiridas pelos Clientes

```
CREATE OR REPLACE VIEW NumeroOfertasPorCliente AS
SELECT cli_nome Nome, COUNT(vo_ofertas_id) 'Numero Ofertas' FROM cliente
INNER JOIN ClienteVendas
ON cli_id = cv_cliente_id
INNER JOIN Vendas
ON cv_vendas_id = ve_id
INNER JOIN VendasOfertas
ON ve_id = vo_vendas_id
INNER JOIN Ofertas
ON vo_ofertas_id = of_id
GROUP BY cli_nome;
```

View Ofertas mais Vendidas por Categoria

```
CREATE OR REPLACE VIEW OfertasMaisVendidasCategoria AS
SELECT of_categoria Categoria, of_nome Nome, COUNT(vo_ofertas_id)'Numero de Ofertas
Vendidas'
FROM Ofertas
INNER JOIN VendasOfertas
ON of_id = vo_ofertas_id
GROUP BY vo_ofertas_id;
```

View Ofertas Mais Procurada (Top 5)

```
CREATE OR REPLACE VIEW OfertaMaisProcurada AS
 SELECT of_nome 'Nome', COUNT(vo_ofertas_id) 'Numero de Ofertas Vendidas'
 FROM Ofertas
 INNER JOIN VendasOfertas
 ON of_id = vo_ofertas_id
```

```
GROUP BY vo_ofertas_id
ORDER BY vo_ofertas_id DESC
LIMIT 5;
```

View Combustível Mais Vendido

```
CREATE OR REPLACE VIEW CombustivelMaisVendido AS
SELECT co_nome 'Combustível', COUNT(ve_id) 'Numero de Vendas'
FROM Combustivel
INNER JOIN  Vendas
ON co_id = ve_id_combustivel
GROUP BY co_nome
HAVING COUNT(*) = (SELECT MAX(ve_id) FROM
                                        (SELECT COUNT(*) ve_id
                                                FROM Vendas
                                                GROUP BY ve_id_combustivel)n);
```

View Combustível Menos Vendido

```
CREATE OR REPLACE VIEW CombustivelMenosVendido AS
SELECT co_nome 'Combustível', COUNT(ve_id) 'Numero de Vendas'
FROM Combustivel
INNER JOIN  Vendas
ON co_id = ve_id_combustivel
GROUP BY co_nome
HAVING COUNT(*) = (SELECT MIN(ve_id) FROM
                                        (SELECT COUNT(*) ve_id
                                                FROM Vendas
                                                 GROUP BY ve_id_combustivel)n);
```

View Colaborador que Efetuou Mais Vendas

```
CREATE OR REPLACE VIEW NVendasColaborador AS
SELECT  DISTINCT col_nome Colaborador , COUNT(ve_id_colaborador) 'Numero de Vendas'
FROM Colaborador
INNER JOIN Vendas
ON col_id = ve_id_colaborador
GROUP BY ve_id_colaborador;
```

View Cliente Registado que Fez Mais Compras

```
CREATE OR REPLACE VIEW ClienteComMaisCompras AS
SELECT cli_nome Nome, COUNT(cv_cliente_id) 'Numero de Compras'
FROM Cliente
INNER JOIN ClienteVendas
ON cli_id = cv_cliente_id
GROUP BY cli_nome
```

```
HAVING COUNT(*) = (SELECT MAX(cv_cliente_id)  FROM
                                                (SELECT COUNT(*) cv_cliente_id
                                                        FROM ClienteVendas
                                                        GROUP BY cv_cliente_id) n);
```

View Número de Vendas Sem Cliente Registado

```
CREATE OR REPLACE VIEW NVendasSemCliente AS
SELECT COUNT(ve_id) 'Numero de Vendas Sem Cliente'
FROM Vendas
INNER JOIN ClienteVendas
ON ve_id = cv_vendas_id
WHERE ve_id =  (SELECT COUNT(cv_vendas_id)
                                    FROM ClienteVendas);
```

View Número de Vendas Com Cliente Registado

```
CREATE OR REPLACE VIEW NVendasComCliente AS
SELECT COUNT(cv_cliente_id) 'Numero de Vendas Com cliente'
FROM ClienteVendas;
```

2ª Fase

Views

View TabelaCliente

ID	Nome	Data Registo	Telefone	Localidade	Pontos	Empresa	Nif	Data de Nascimento	Rua	Nr	Andar	Porta	Codigo_Postal	Estado
1	Jonas	2016-01-01	915152789	Lisboa	437	Sadin	24587915	1992-05-12	Rua Alves Santos	5	1º	A	2500	Ativo
2	Rui	2015-05-12	916243897	Setubal	43	TerraCotaa	45135798	1989-03-20	Rua Jacinto Andrade	7	Não Tem	Não Tem	2900	Ativo
3	Andre	2016-03-11	911892543	Lisboa	414	Landa	29874561	1995-06-04	Rua Moreira OL	10	2º	B	5500	Ativo
4	Lopes	2015-07-20	963879534	Almada	526	Grouje	24587915	1993-10-25	Avenida Europa	4	3º	C	2910	Ativo
5	Paulo	2014-12-01	934587326	Cascais	40	Kool	13458794	1990-05-07	Avenida Portal	5	Não Tem	Não Tem	5470	Ativo
6	Pedro	2013-11-20	925554123	Faro	50	TicTravel	154222	1988-11-20	Avenida Trip	6	Não Tem	Não Tem	451	Ativo
7	Jose	2016-01-11	910000458	Porto	545	Moveit	14587988	1995-10-25	Rua Jose DelWhiskey	41	Não Tem	Não Tem	2541	Ativo
8	Diana	2014-02-21	920125487	Seixal	1425	JabbaTaxi	15548798	1980-10-05	Rua do Cavaco	7	Não Tem	Não Tem	2840	Ativo
9	Miguel	2015-03-25	920111523	Corroios	5000	DingDong	14457821	1996-05-14	Praceta Afonso Henriques	8	Não Tem	Não Tem	2694	Ativo
10	Luis	2013-02-09	920113243	Setubal	409	DingDong	55612456	1994-08-14	Algum lado	3	Não Tem	Não Tem	2234	Ativo
11	Petra	2015-12-25	920111523	Angola	5000	Praias	2468123	1992-05-20	Rua em Africa	2	Não Tem	Não Tem	3	Ativo
12	Neo	1999-03-25	920111523	Matrix	5000	Zion	548913256	1978-05-14	Unknown Street	0	Não Tem	Não Tem	9999	Ativo
13	Ruben	2013-10-04	920111523	Lisboa	5000	Empresa X	4549412	1993-06-24	Rua Malmeqer Nao é	14	Não Tem	Não Tem	8432	Ativo
14	Carvas	2012-03-23	920111523	Algures	4474	Empresa Y	14457821	1991-01-11	Planeta Escondido	1	Não Tem	Não Tem	2002	Ativo
15	Polus	2011-02-25	920111523	Loonar	5000	Street	1466221	1992-05-13	Korner Surfer	10	Não Tem	Não Tem	2694	Ativo
16	Romolus	2011-02-25	920111523	Loonar	5028	Street	10000021	1992-05-14	Korner Surfer	11	Não Tem	Não Tem	2694	Ativo

Com esta View conseguimos ver toda a informação relativamente aos clientes, nesta já conseguimos ver que a coluna Andar e Porta já não se encontram a null, pois como a inserção na tabela é feita com um stored procedures este confirma se os campos estão a null ou não. Exemplo encontra-se no script teste.sql, linha 21.

View TabelaColaborador

ID	Nome	Data Registo	Telefone	Localidade	Rua	Nr	Andar	Porta	Codigo Postal	Data de Nascimento	Estado
1	Kore	2015-03-14	91458752	Setubal	Rua Super Homem	23	5º	G	2900	1990-05-17	Ativo
2	Yann	2016-02-03	934516879	Montijo	Avenida Plural	5	Não Tem	Não Tem	4500	1985-10-16	Ativo
3	Karlos	2014-10-22	924571284	Almada	Rua dos Peixes	8	1º	Esquerdo	5400	1991-05-14	Ativo
4	Lara	2015-12-20	914758234	Setubal	Rua Destruida	1	Não Tem	Não Tem	2910	1993-04-07	Ativo
5	Mario	2016-04-01	916289341	Lisboa	Avenida 5 Espanha	5	4º	A	3550	1991-01-08	Ativo
6	Carlos	2014-05-06	921458221	Setubal	Avenida Peixinho Dourado	8	2º	B	2456	1992-02-12	Ativo
7	Cecilia	2014-11-15	921114589	Seixal	Rua D.Dinis	5	Não Tem	Não Tem	2458	1991-01-31	Ativo
8	Maró	2012-11-15	932312319	Setubal	Rua D.Dinas	2	Não Tem	Não Tem	2233	1991-01-04	Ativo
9	Kool	2014-12-12	921177489	Lisboa	Rua Repetida	5	Não Tem	Não Tem	2458	1991-01-31	Ativo
10	kola	2014-01-16	923232323	Setubal	Rua Sao Jaquim	39	Não Tem	Não Tem	2758	1991-03-31	Ativo
11	kolo	2014-02-13	921132389	Montijo	Rua Alentejo	10	Não Tem	Não Tem	8768	1991-05-31	Ativo
12	kole	2014-06-14	92232389	New York	Avenida Ar	2	Não Tem	Não Tem	2776	1991-04-30	Ativo
13	Paulo	2012-10-15	92311145	Seixal	Marques Ines	10	Não Tem	Não Tem	6558	1992-03-31	Ativo
14	Caló	2013-09-11	921114589	Lisboa	Rua Lopes Andrade	7	Não Tem	Não Tem	2468	1991-04-30	Ativo
15	Nocas	2015-12-15	921352558	OLE	Rua De Deus	1	Não Tem	Não Tem	1458	1993-03-09	Ativo
16	Tildas	2014-11-15	921114589	Seixal	Margem Sul	2	Não Tem	Não Tem	2348	1995-05-31	Ativo
17	Trapla	2012-02-27	943252239	Setubal	Rua Baixa	1	Não Tem	Não Tem	6666	1991-10-10	Ativo

Com esta View conseguimos ver toda a informação relativamente aos colaboradores, nesta já conseguimos ver que a coluna Andar e Porta já não se encontram a null, pois como a inserção na tabela é feita com um stored procedures este confirma se os campos estão a null ou não. Exemplo encontra-se no script teste.sql, linha 24.

View TabelaFornecedor

ID	Nome	Data Registo	Telefone	Localidade	Empresa	Rua	Nr	Andar	Porta	Codigo Postal	Estado
1	Luis	2010-05-18	916458237	Setubal	CombusAll	Rua dos combustiveis	10	Não Tem	Não Tem	3150	Ativo
2	Pedro	2012-12-04	935761592	Almada	EnergiasVivas	Rua Sofie	2	Não Tem	Não Tem	4500	Ativo
3	Torin	2011-07-30	916732851	Setubal	LigaLeve	Rua Joaquem Almeida	5	2º	C	5500	Ativo
4	Holha	2015-05-08	963482516	Almada	SabesCombustivel	Avenida Mal Me Quer	6	3º	D	3690	Ativo
5	Hefe	2015-08-06	964829537	Montijo	MundoVerde	Avenida Qualquer	8	Não Tem	Não Tem	2350	Ativo
6	Ze	2010-11-24	924587112	Amora	DieselLife	Rua Zeca Afonso	8	Não Tem	Não Tem	2451	Ativo
7	Diogo	2009-03-22	914567852	Sesimbra	TreeHug	Rua Mane	1	Não Tem	Não Tem	2548	Ativo
8	Yah	2008-04-21	97453735	Setubal	TreeMongered	Rua Kebab	10	Não Tem	Não Tem	952	Ativo
9	Han Solo	2007-06-16	93423522	Lisboa	TomSavior	Rua vida	2	Não Tem	Não Tem	1092	Ativo
10	Lalanja Boy	2013-04-09	91242852	Algures	AlguresCA	Rua Perdida	5	Não Tem	Não Tem	9367	Ativo
11	Carto	2010-02-10	932527852	Cevilha	SpanCompany	La Rua Rua	34	Não Tem	Não Tem	1528	Ativo
12	Yur	2011-08-25	939478278	Pertinho	Kiirmon	Rua Francesa	54	Não Tem	Não Tem	2111	Ativo
13	Yomcha	2007-01-10	96323232	Cidade	Polis	Rua Polis	14	Não Tem	Não Tem	1248	Ativo
14	Goku	2010-05-10	9163473	Pinhal Novo	World	Rua World	19	Não Tem	Não Tem	4923	Ativo
15	Freeza	2001-10-22	91434645	Nave Esp...	Freeza Company	Desconhecida	0	Não Tem	Não Tem	1001	Ativo
16	Satan	2004-06-16	92131852	Nameque	SatanEmprsa	Rua Nameque	12	Não Tem	Não Tem	1123	Ativo
17	Trolha	2014-01-21	914522852	Sesimbra	Pelota	Rua Pelota	11	Não Tem	Não Tem	1156	Ativo

Com esta View conseguimos ver toda a informação relativamente aos fornecedores, nesta já conseguimos ver que a coluna Andar e Porta já não se encontram a null, pois como a inserção na tabela é feita com um stored procedures este confirma se os campos estão a null ou não. Exemplo encontra-se no script teste.sql, linha 27.

View TabelaCombustivel

ID	Custo por Litro	Nome	Data de Inicio	Data de Fim	ID Fornecedor	Stock
1	1.24	Gasolina 95	2016-06-01	2016-06-01	1	0
2	0.99	Gasoleo	2016-06-01	NULL	2	89860
3	1.20	Gasolina 95	2016-06-01	NULL	5	90162
4	1.40	Gasolina 98	2016-06-01	NULL	3	89889

Com esta view conseguimos ver toda a informação relativamente aos combustíveis. Nesta vemos que a colune Data de Fim tem alguns valores a null, isto deve-se a que, sempre que

um combustível está disponível ainda não tem Data de Fim, o que já não acontece no combustível com o id 1, que ficou com uma data de fim, pois foi inserido um novo combustível com o mesmo nome. Ao adicionar um novo combustível se o combustível antigo tiver stock, este será removido e adicionado ao stock do novo combustível adicionado. Exemplo encontra-se no script teste.sql, linha 30.

View TabelaOfertas

ID	Nome	Pontos	Data de Inicio	Data de Fim	Descricao	Categoria	Numero de Ofertas	Litros
15	5 Litros	450	2016-01-01	2016-05-01	Gasolina 95 5 Litros	Descontos	NULL	5
16	10 Litros	900	2016-01-15	2016-06-01	Gasolina 95 10 Litros	Descontos	NULL	10
17	10 Litros	900	2016-01-15	2016-06-01	Gasoleo 5 Litros	Descontos	NULL	5
18	10 Litros	800	2016-02-01	2016-06-01	Gasoleo 10 Litros	Descontos	NULL	10
19	10 Litros	900	2016-01-15	2016-06-01	Gasolina 98 10 Litros	Descontos	NULL	10
20	5 Litros	450	2016-01-15	2016-06-01	Gasolina 98 5 Litros	Descontos	NULL	5
1	Bicicleta 110	5000	2016-01-05	2016-05-20	Bicicleta todo terreno	Material	20	NULL
2	Apito	100	2016-03-20	2016-05-20	Faz barulho	Material	97	NULL
3	Porta Chaves	150	2016-03-20	2016-05-20	Decoracao	Material	99	NULL
4	Peluche	800	2016-03-20	2016-05-20	Fofo	Material	98	NULL
5	DVD	400	2016-03-20	2016-05-20	Filme à escolha	Material	100	NULL
6	Livro	400	2016-03-20	2016-05-20	Livro à escolha	Material	100	NULL
7	Cafe	50	2016-03-20	2016-05-20	Café	Material	9991	NULL
8	Pequeno-Al...	200	2016-03-20	2016-05-20	Pequeno Almoço	Material	1000	NULL
9	Gelado	150	2016-03-20	2016-05-20	Gelado à escolha	Material	999	NULL
10	Cenas	990	2016-03-20	2016-05-20	Cenas	Material	198	NULL
11	Magic Sword	99999	2016-03-20	2016-05-20	Espada Magica!	Material	1	NULL

Com esta View conseguimos visualizar toda a informação referente às ofertas. Nesta view vão sempre existir alguns campos a null sendo estes o Numero de Ofertas e Litros. O Numero de Ofertas vai estar sempre a null, quando a oferta é da categoria Descontos, já para o campo Litros este vai estar sempre a null quando a oferta é da categoria Material, Isto porque um material não tem litros, mas tem número de ofertas disponíveis. Os Descontos vão ter os litros pois é o número de litros associado a esta oferta e quantos litros este pode descontar. Exemplo encontra-se no script teste.sql, linha 33.

View TabelaVendas

ID	ID Combustivel	Taxa de Iva	Litros	Custo	Data de Venda	Id Colaborador
1	2	0.23	15.00	18.00	2016-06-01	1
2	2	0.23	14.00	17.00	2016-06-01	2
3	4	0.23	5.50	10.00	2016-06-01	3
4	3	0.23	20.00	30.00	2016-06-01	1
5	4	0.23	5.50	10.00	2016-06-01	2
6	3	0.23	9.63	15.00	2016-06-01	10
7	3	0.23	8.00	12.00	2016-06-01	15
8	4	0.23	10.00	17.00	2016-06-01	13
9	3	0.23	32.08	50.00	2016-06-01	9
10	2	0.23	27.22	35.00	2016-06-01	19
11	3	0.23	19.00	28.00	2016-06-01	8
12	4	0.23	5.00	9.00	2016-06-01	7
13	2	0.23	13.22	17.00	2016-06-01	9
14	3	0.23	19.00	28.00	2016-06-01	17
15	4	0.23	15.95	29.00	2016-06-01	5
16	3	0.23	48.00	71.00	2016-06-01	18
17	3	0.23	26.00	38.00	2016-06-01	15

Com esta View conseguimos ver toda a informação referente às vendas. Nesta conseguimos ver que os Litros e o Custo já não se encontram a null em nenhum caso, isto deve-se ao facto de ser inserido com um stored procedures e este calcula com base no custo ou nos litros dados qual os valores, seja dos litros ou do custo. Exemplo encontra-se no script teste.sql, linha 36.

View TabelaFaturacao

ID	Custo Total	Data	Iva	ID Venda	Estado
1	18.00	2016-06-01	4.14	1	Fechada
2	17.00	2016-06-01	3.91	2	Fechada
3	10.00	2016-06-01	2.30	3	Fechada
4	30.00	2016-06-01	6.90	4	Fechada
5	10.00	2016-06-01	2.30	5	Fechada
6	15.00	2016-06-01	3.45	6	Fechada
7	50.00	2016-06-01	11.50	9	Fechada
8	35.00	2016-06-01	8.05	10	Fechada
9	28.00	2016-06-01	6.44	11	Fechada
10	9.00	2016-06-01	2.07	12	Fechada
11	71.00	2016-06-01	16.33	16	Fechada
12	10.00	2016-06-01	2.30	18	Fechada
13	50.00	2016-06-01	11.50	20	Fechada
14	24.00	2016-06-01	5.52	22	Fechada
15	20.00	2016-06-01	4.60	23	Fechada
16	40.00	2016-06-01	9.20	25	Fechada
17	12.00	2016-06-01	2.76	27	Fechada

Com esta view é possível ver toda a informação referente à tabela faturacao. O estado de cada fatura quando é criada é por omissão Fechada. Para se abrir existe um stored procedures que será falado posteriormente em que se pode abrir uma fatura para alterar valores. Exemplo encontra-se no script teste.sql, linha 39.

View TabelaClienteVenda

Id Cliente	Id Venda
1	1
1	25
2	2
2	11
2	28
3	3
4	4
4	16
4	27
5	5
5	18
6	6
10	9
10	10
14	12
14	22
16	20

Com esta View é possível toda a informação referente à tabela clienteVendas, ordenada pelo id do cliente. Exemplo encontra-se no script teste.sql, linha 42.

View TabelaVendaOfertas

Id Venda	Id Oferta
1	2
2	3
2	2
3	7
4	7
5	7
6	7
7	4
8	4
8	13
9	7
10	2
10	9
10	7
12	7
15	10
17	14

Com esta View é possível toda a informação referente à tabela vendaOfertas , ordenada pelo id da venda. Exemplo encontra-se no script teste.sql, linha 45.

View nr_fatura_cliente

Nome	Numero de Faturas	Total Com Iva	Total Sem Iva
Andre	1	10.00	7.70
Carvas	2	33.00	25.41
Gajo	1	27.00	20.79
Holis	1	20.00	15.40
Jonas	2	58.00	44.66
Lopes	3	113.00	87.01
Luis	2	85.00	65.45
Paulo	2	20.00	15.40
Pedro	1	15.00	11.55
Romolus	1	50.00	38.50
Rui	3	67.00	51.59

View que mostra todos os clientes com faturas associadas e quantas faturas lhes estão associadas. Para além disse calcula o total com iva e sem iva de todas as faturas associadas. Exemplo encontra-se no script teste.sql, linha 48.

View OfertaMenosProcurada

Nome	Numero de Ofertas Vendidas
10 Litros	1
Porta Chaves	1
Gelado	1
Jogo	1
Peluche	2

View que mostra as 5 ofertas menos procuradas. Exemplo encontra-se no script teste.sql, linha 51.

Stored Procedures

Stored Procedure sp_registar_cliente

Antes de registar, temos 20 clientes.

16	Romolus	2011-02-25	920111523	Loonar	5028	Street	10000021	1992-05-14	Korner Surfer	11	Não Tem	Não Tem	2694	Ativo
17	Holis	2016-04-25	92434433	Luna	4961	Empresa X	100443321	1995-06-14	BLAAAA	2	Não Tem	Não Tem	1294	Ativo
18	Coru	2016-05-20	920124223	Loonar	5000	Street	10000021	1994-05-14	Korner Ridge	3	Não Tem	Não Tem	2645	Ativo
19	Gajo	2015-04-03	923231523	Place	248	QUALQA	1342400...	1996-02-04	Barker	10	Não Tem	Não Tem	2694	Ativo
20	20Ultimo	2016-05-30	91123829	CabouEste	5000	Bica	10000021	1991-01-14	Vista de Setubal	1	Não Tem	Não Tem	1900	Ativo

Após a chamada do stored procedures, temos 21. Exemplo encontra-se no script teste.sql, linha 54.

18	Coru	2016-05-20	920124223	Loonar	5000	Street	10000021	1994-05-14	Korner Ridge	3	Não Tem	Não Tem	2645	Ativo
19	Gajo	2015-04-03	923231523	Place	248	QUALQA	1342400...	1996-02-04	Barker	10	Não Tem	Não Tem	2694	Ativo
20	20Ultimo	2016-05-30	91123829	CabouEste	5000	Bica	10000021	1991-01-14	Vista de Setubal	1	Não Tem	Não Tem	1900	Ativo
21	Rui	2015-05-12	916243897	Setubal	245	TerraCotaa	45135798	1989-03-20	Rua Jacinto Andrade	7	Não Tem	Não Tem	2900	Ativo

Stored Procedure sp_registar_colaborador

Antes de registar, temos 20 colaboradores.

18	Pinhal	2014-02-19	918201489	Montijo	Rua Algures	7	Não Tem	Não Tem	4348	1992-02-19	Ativo
19	Priscila	2014-11-15	921114589	Pinhal Novo	Rua Soma	6	Não Tem	Não Tem	2425	1993-11-12	Ativo
20	Raminho	2014-11-15	921114589	Algures	Rua Vida	3	Não Tem	Não Tem	5555	1994-08-24	Ativo

Após a chamada do stored procedures, temos 21. Exemplo encontra-se no script teste.sql, linha 58.

18	Pinhal	2014-02-19	918201489	Montijo	Rua Algures	7	Não Tem	Não Tem	4348	1992-02-19	Ativo
19	Priscila	2014-11-15	921114589	Pinhal Novo	Rua Soma	6	Não Tem	Não Tem	2425	1993-11-12	Ativo
20	Raminho	2014-11-15	921114589	Algures	Rua Vida	3	Não Tem	Não Tem	5555	1994-08-24	Ativo
21	Kore	2015-03-14	91458752	Setubal	Rua Super Homem	23	5º	G	2900	1990-05-17	Ativo

Stored Procedure sp_registar_fornecedor

Antes de registar, temos 20 fornecedores.

18	Robulha	2012-10-10	95353852	Montijo	Arbore	Rua Arbore	12	Não Tem	Não Tem	2382	Ativo
19	Marques	2011-12-01	9525353	Setubal	Flore	Rua Flore	13	Não Tem	Não Tem	2313	Ativo
20	Fornecedor	2010-12-17	91213852	Alcacer	Penoi	Rua Penoi	16	Não Tem	Não Tem	1252	Ativo

Após a chamada do stored procedures, temos 21. Exemplo encontra-se no script teste.sql, linha 63.

20	Fornecedor	2010-12-17	91213852	Alcacer	Penoi	Rua Penoi	16	Não Tem	Não Tem	1252	Ativo
21	Luis	2010-05-18	916458237	Setubal	CombusAll	Rua dos combustiveis	10	Não Tem	Não Tem	3150	Ativo

Stored Procedure sp_registar_combustivel

Antes de registar, temos 4 combustíveis.

ID	Custo por Litro	Nome	Data de Inicio	Data de Fim	ID Fornecedor	Stock
1	1.24	Gasolina 95	2016-06-01	2016-06-01	1	0
2	0.99	Gasoleo	2016-06-01	NULL	2	89860
3	1.20	Gasolina 95	2016-06-01	NULL	5	90162
4	1.40	Gasolina 98	2016-06-01	NULL	3	89889

Após a chamada do stored procedure, temos 5. Exemplo encontra-se no script teste.sql, linha 67.

ID	Custo por Litro	Nome	Data de Inicio	Data de Fim	ID Fornecedor	Stock
1	1.24	Gasolina 95	2016-06-01	2016-06-01	1	0
2	0.99	Gasoleo	2016-06-01	NULL	2	89860
3	1.20	Gasolina 95	2016-06-01	2016-06-01	5	0
4	1.40	Gasolina 98	2016-06-01	NULL	3	89889
5	1.24	Gasolina 95	2016-06-01	NULL	1	90562

Stored Procedure sp_registar_oferta

Antes de registar, temos 20 ofertas.

18	10 Litros	800	2016-02-01	2016-06-01	Gasoleo 10 Litros	Descontos	NULL	10
19	10 Litros	900	2016-01-15	2016-06-01	Gasolina 98 10 Litros	Descontos	NULL	10
20	5 Litros	450	2016-01-15	2016-06-01	Gasolina 98 5 Litros	Descontos	NULL	5

Após a chamada do stored procedures, temos 21. Exemplo encontra-se no script teste.sql, linha 73.

19	10 Litros	900	2016-01-15	2016-06-01	Gasolina 98 10 Litros	Descontos	NULL	10
20	5 Litros	450	2016-01-15	2016-06-01	Gasolina 98 5 Litros	Descontos	NULL	5
21	5 Litros	300	2016-06-01	2016-05-01	Gasolina 95 5 Litros	Descontos	NULL	5

Stored Procedure sp_registar_fatura

Antes de registar temos 20 faturas. A Partir do id da venda e de um cliente conseguimos registar uma fatura.

18	22.00	2016-06-01	5.06	28	Fechada
19	27.00	2016-06-01	6.21	29	Fechada
20	30.00	2016-06-01	6.90	26	Fechada

Após a chamada do stored procedures, ficamos com 21 faturas. Exemplo encontra-se no script teste.sql, linha 77.

19	27.00	2016-06-01	6.21	29	Fechada
20	30.00	2016-06-01	6.90	26	Fechada
21	17.00	2016-06-01	3.91	8	Fechada

Stored Procedure sp_relacionar_clienteVenda

Antes de relacionar um cliente com uma venda, temos 21 relações. Este relacionar occore quando se abre uma nova fatura.

Id Cliente	Id Venda
1	1
1	25
2	2
2	11

Após relacionar, neste caso o cliente 1 com a venda 5, ficamos com 22 relações. Exemplo encontra-se no script teste.sql, linha 81.

Id Cliente	Id Venda
1	1
1	25
1	5

Stored Procedure sp_relacionar_vendaOferta

Antes de relacionar a tabela tem 22 registos. Este procedures serve para quando um cliente quer uma oferta e vai tentar associar a oferta pretendida a uma venda. Só funciona se o cliente tiver pontos suficientes para usufruir da oferta.

Id Venda	Id Oferta
1	2
2	3
2	2
3	7
4	7

Após relacionar vamos ter 23 registos, neste caso em particular vamos relacionar mais uma vez a venda 1 com a oferta 2. Exemplo encontra-se no script teste.sql, linha 85.

Id Venda	Id Oferta
1	2
1	2
2	3
2	2
3	7

Stored Procedure sp_produtos_fatura

Este mostra os produtos associados a uma fatura através do seu id. No exemplo vamos visualizar os produtos da fatura 2. Exemplo encontra-se no script teste.sql, linha 90.

Id Fatura	Combustivel	Litros	Oferta
2	Gasoleo	14.00	Porta Chaves
2	Gasoleo	14.00	Apito

Stored Procedure sp_registar_venda

Com esta conseguimos registar uma venda. Neste momento temos 29 registos. Este stored procedures é muito importante pois este vai calcular com base nos litros ou custo inserido qual os litros ou custo associado a esta venda internamente, o que não era possível anteriormente com um insert.

27	3	0.23	8.00	12.00	2016-06-01	15
28	3	0.23	15.00	22.00	2016-06-01	19
29	3	0.23	18.00	27.00	2016-06-01	16

Após a chamada do stored procedures ficamos com 30 registos. Exemplo encontra-se no script teste.sql, linha 93.

28	3	0.23	15.00	22.00	2016-06-01	19
29	3	0.23	18.00	27.00	2016-06-01	16
30	4	0.23	10.00	17.00	2016-06-01	3

Stored Procedure sp_listar_ofertas_categoria

Com este procedure conseguimos ver todas as ofertas que já foram incluídas numa venda a partir da sua categoria. Exemplo encontra-se no script teste.sql, linha 98.

Categoria	Nome	Numero de Ofertas Vendidas
Material	Cafe	9
Material	Apito	4
Material	Cenas	2
Material	Peluche	2
Material	Portal	2
Material	Jogo	1
Material	Gelado	1
Material	Porta Chaves	1

Stored Procedure sp_resumo_fatura

Com este procedure conseguimos ver o resumo de uma fatura a partir do seu id, neste exemplo vamos ver o resumo da fatura com o id 2. Exemplo encontra-se no script teste.sql, linha 101.

Id Fatura	Nome	Combustivel	Litros	Numero Ofertas	Total Com Iva	Total Sem Iva	Estado
2	Rui	Gasoleo	14.00	2	34.00	26.18	Fechada

Stored Procedure sp_volume_anual_mes

Este stored procedure, mostra o volume anual dividido pelos meses. A informação para este está associado às vendas com fatura. Para este stored procedure é preciso enviar uma data com o ano relativo ao que pretendemos. Nos resultados vamos apenas conseguir visualizar os dados relativos ao mês de junho pois os registos da faturas, a data, é feita com um CURDATE() o que leva a que apenas existam dados relativos ao mês corrente.

Numero Vendas	Iva Pago	Total Sem Iva	Total Com Iva	Mes	Nif
22	127.65	427.35	555.00	June	24587915

Stored Procedure sp_remover_venda

Este procedure remove uma venda e caso haja uma fatura associada também apaga a sua fatura. Vamos apagar a venda 1. Exemplo encontra-se no script teste.sql, linha 109.

ID	ID Combustivel	Taxa de Iva	Litros	Custo	Data de Venda	Id Colaborador
2	2	0.23	14.00	17.00	2016-06-01	2
3	4	0.23	5.50	10.00	2016-06-01	3
4	3	0.23	20.00	30.00	2016-06-01	1

Stored Procedure sp_fechar_fatura

Com este procedure conseguimos fechar uma fatura a partir do seu id e devolve os detalhes da mesma. No exemplo vamos fechar a fatura 5. Exemplo encontra-se no script teste.sql, linha 117.

Id Fatura	Nome	Combustivel	Litros	Numero Ofertas	Total Com Iva	Total Sem Iva	Estado
5	Paulo	Gasolina 98	5.50	2	20.00	15.40	Aberta

Após fechar fatura 5.

Id Fatura	Nome	Combustivel	Litros	Numero Ofertas	Total Com Iva	Total Sem Iva	Estado
5	Paulo	Gasolina 98	5.50	2	20.00	15.40	Fechado

Stored Procedure sp_abrir_fatura

Com este procedure conseguimos abrir uma fatura a partir do seu id e devolve os detalhes da mesma. No exemplo vamos abrir a fatura 5. Exemplo encontra-se no script teste.sql, linha 121.

Id Fatura	Nome	Combustivel	Litros	Numero Ofertas	Total Com Iva	Total Sem Iva	Estado
5	Paulo	Gasolina 98	5.50	2	20.00	15.40	Fechado

Após abrir a fatura 5.

Id Fatura	Nome	Combustivel	Litros	Numero Ofertas	Total Com Iva	Total Sem Iva	Estado
5	Paulo	Gasolina 98	5.50	2	20.00	15.40	Aberta

Stored Procedure sp_adicionar_combustivel

Com este procedure conseguimos adicionar um combustível a uma fatura, caso esta esteja com o estado de aberto. Neste exemplo vamos adicionar à venda 3, mais 5 litros de combustível. O Exemplo encontra-se no script teste.sql, linha 126.

Antes de adicionar:

ID	ID Combustivel	Taxa de Iva	Litros	Custo	Data de Venda	Id Colaborador
2	2	0.23	14.00	17.00	2016-06-01	2
3	4	0.23	5.50	10.00	2016-06-01	3

Depois de adicionar:

ID	ID Combustivel	Taxa de Iva	Litros	Custo	Data de Venda	Id Colaborador
2	2	0.23	14.00	17.00	2016-06-01	2
3	4	0.23	10.50	17.00	2016-06-01	3

Stored Procedure sp_alterar_stock_combustivel

Com este procedure conseguimos adicionar ao stock de um combustível uma quantidade de combustível, na tabela de combustíveis

ID	Custo por Litro	Nome	Data de Inicio	Data de Fim	ID Fornecedor	Stock
1	1.24	Gasolina 95	2016-06-01	2016-06-01	1	0
2	0.99	Gasoleo	2016-06-01	NULL	2	89875
3	1.20	Gasolina 95	2016-06-01	NULL	5	90162
4	1.40	Gasolina 98	2016-06-01	NULL	3	89874

Após se alterar o stock do combustível com o id 2. Exemplo encontra-se no script teste.sql, linha 130.

ID	Custo por Litro	Nome	Data de Inicio	Data de Fim	ID Fornecedor	Stock
1	1.24	Gasolina 95	2016-06-01	2016-06-01	1	0
2	0.99	Gasoleo	2016-06-01	NULL	2	90075
3	1.20	Gasolina 95	2016-06-01	NULL	5	90162
4	1.40	Gasolina 98	2016-06-01	NULL	3	89874

Stored Procedure sp_remove_combustivel_fatura

Com este stored procedure conseguimos remover uma quantidade de um combustivel de uma fatura. Antes de remover combustivel:

Id Fatura	Nome	Combustivel	Litros	Numero Ofertas	Total Com Iva	Total Sem Iva	Estado
3	Andre	Gasolina 98	5.50	1	10.00	7.70	Fechada

Após remover uma quantidade de combustivel. Exemplo encontra-se no script teste.sql, linha 134.

Id Fatura	Nome	Combustivel	Litros	Numero Ofertas	Total Com Iva	Total Sem Iva	Estado
3	Andre	Gasolina 98	0.50	1	10.00	7.70	Fechada

Stored Procedure sp_remover_oferta

Com este procedure conseguimos remover uma oferta da tabela de oferta. Neste exemplo vamos remover a oferta 19. Exemplo encontra-se no script teste.sql, linha 141.

16	10 Litros	900	2016-01-15	2016-06-01	Gasolina 95 10 Litros	Descontos	NULL	10
17	10 Litros	900	2016-01-15	2016-06-01	Gasoleo 5 Litros	Descontos	NULL	5
18	10 Litros	800	2016-02-01	2016-06-01	Gasoleo 10 Litros	Descontos	NULL	10
20	5 Litros	450	2016-01-15	2016-06-01	Gasolina 98 5 Litros	Descontos	NULL	5

Stored Procedure sp_alterar_oferta_nome

Com este procedure conseguimos alterar o nome de uma oferta, neste exemplo vamos alterar o nome da oferta 1 para OLEEEE. Exemplo encontra-se no script teste.sql, linha 146.

ID	Nome	Pontos	Data de Inicio	Data de Fim	Descricao	Categoria	Numero de Ofertas	Litros
1	OLEEEE	5000	2016-01-05	2016-05-20	Bicicleta todo terreno	Material	20	NULL

Stored Procedure sp_alterar_oferta_pontos

Com este procedure conseguimos alterar os pontos de uma oferta, neste exemplo vamos alterar os pontos da oferta 1 para 500. Exemplo encontra-se no script teste.sql, linha 150.

ID	Nome	Pontos	Data de Inicio	Data de Fim	Descricao	Categoria	Numero de Ofertas	Litros
1	OLEEEE	500	2016-01-05	2016-05-20	Bicicleta todo terreno	Material	20	NULL

Stored Procedure sp_alterar_oferta_data_inicio

Com este procedure conseguimos alterar a data de início de uma oferta, sendo que esta nunca pode ser maior que a data de fim. Vamos alterar a data de início da oferta 1. Exemplo encontra-se no script teste.sql, linha 154.

ID	Nome	Pontos	Data de Inicio	Data de Fim	Descricao	Categoria	Numero de Ofertas	Litros
1	OLEEEE	500	2016-05-19	2016-05-20	Bicicleta todo terreno	Material	20	NULL

Stored Procedure sp_alterar_oferta_data_fim

Com este procedure conseguimos alterar a data de fim de uma oferta, sendo que esta nunca pode ser menor que a data de início. Vamos alterar a data de fim da oferta 1. Exemplo encontra-se no script teste.sql, linha 158.

ID	Nome	Pontos	Data de Inicio	Data de Fim	Descricao	Categoria	Numero de Ofertas	Litros
1	OLEEEE	500	2016-05-19	2016-05-22	Bicicleta todo terreno	Material	20	NULL

Stored Procedure sp_alterar_oferta_descricao

Com este procedure conseguimos alterar a descrição de uma oferta. Neste exemplo vamos alterar a descrição da oferta 1. Exemplo encontra-se no script teste.sql, linha 162.

ID	Nome	Pontos	Data de Inicio	Data de Fim	Descricao	Categoria	Numero de Ofertas	Litros
1	OLEEEE	500	2016-05-19	2016-05-22	Cenas do Mexixo	Material	20	NULL

Stored Procedure sp_alterar_oferta_categoria

Com este procedure conseguimos alterar a categoria da oferta, para Descontos ou Material. Neste exemplo vamos alterar a categoria da oferta 1 para Descontos. Exemplo encontra-se no script teste.sql, linha 166.

ID	Nome	Pontos	Data de Inicio	Data de Fim	Descricao	Categoria	Numero de Ofertas	Litros
1	OLEEEE	500	2016-05-19	2016-05-22	Cenas do Mexixo	Descontos	20	NULL

Stored Procedure sp_alterar_oferta_numeroOfertas

Com este procedure conseguimos alterar o número de ofertas de uma oferta caso esta seja do tipo Material. Neste exemplo vamos alterar a oferta 1 para que o número de ofertas seja de 50. Exemplo encontra-se no script teste.sql, linha 170.

ID	Nome	Pontos	Data de Inicio	Data de Fim	Descricao	Categoria	Numero de Ofertas	Litros
1	OLEEEE	500	2016-05-19	2016-05-22	Cenas do Mexixo	Material	50	NULL

Stored Procedure sp_alterar_oferta_litros

Com este procedure conseguimos alterar os litros de uma oferta, caso esta seja do tipo Descontos. Neste exemplo vamos alterar a oferta 15 para que passe a ter 10 litros. Exemplo encontra-se no script teste.sql, linha 174.

ID	Nome	Pontos	Data de Inicio	Data de Fim	Descricao	Categoria	Numero de Ofertas	Litros
15	5 Litros	450	2016-01-01	2016-05-01	Gasolina 95 5 Litros	Descontos	NULL	10

Stored Procedure sp_remover_fatura

Com este procedure conseguimos remover uma fatura a partir do seu id. Neste exemplo vamos remover a fatura com o id 4. Exemplo encontra-se no script teste.sql, linha 178.

ID	Custo Total	Data	Iva	ID Venda	Estado
1	18.00	2016-06-01	4.14	1	Fechada
2	17.00	2016-06-01	3.91	2	Fechada
3	10.00	2016-06-01	2.30	3	Fechada
5	10.00	2016-06-01	2.30	5	Fechada

Stored Procedure sp_remover_oferta_fatura

Com este procedure conseguimos remover uma oferta de uma fatura, aberta, a partir do id da fatura e do nome da oferta em questão. Neste exemplo vamos remover da fatura 8, o café. Exemplo encontra-se no script teste.sql, linha 184.

Id Fatura	Combustivel	Litros	Oferta
8	Gasoleo	27.22	Apito
8	Gasoleo	27.22	Gelado

Stored Procedure sp_desativar_cliente

Com este procedure podemos desativar um cliente, ou seja, mudar o seu estado para inativo. Neste exemplo vamos mudar o estado do cliente com o id 1 para inativo. Exemplo encontra-se no script teste.sql, linha 188.

ID	Nome	Data Registo	Telefone	Localidade	Pontos	Empresa	Nif	Data de Nascimento	Rua	Nr	Andar	Porta	Codigo_Postal	Estado
1	Jonas	2016-01-01	915152789	Lisboa	437	Sadin	24587915	1992-05-12	Rua Alves Santos	5	1º	A	2500	Inativo

Stored Procedure sp_ativar_cliente

Com este procedure podemos ativar um cliente, ou seja, mudar o seu estado para ativo. Neste exemplo vamos mudar o estado do cliente com o id 1 para ativo. Exemplo encontra-se no script teste.sql, linha 192.

ID	Nome	Data Registo	Telefone	Localidade	Pontos	Empresa	Nif	Data de Nascimento	Rua	Nr	Andar	Porta	Codigo_Postal	Estado
1	Jonas	2016-01-01	915152789	Lisboa	437	Sadin	24587915	1992-05-12	Rua Alves Santos	5	1º	A	2500	Ativo

.

Stored Procedure sp_desativar_colaborador

Com este procedure podemos desativar um colaborador, ou seja, mudar o seu estado para inativo. Neste exemplo vamos mudar o estado do colaborador com o id 2 para inativo. Exemplo encontra-se no script teste.sql, linha 196.

2	Yann	2016-02-03	934516879	Montijo	Avenida Plural	5	Não Tem	Não Tem	4500	1985-10-16	Inativo

Stored Procedure sp_ativar_colaborador

Com este procedure podemos ativar um colaborador, ou seja, mudar o seu estado para ativo. Neste exemplo vamos mudar o estado do colaborador com o id 2 para ativo. Exemplo encontra-se no script teste.sql, linha 200.

2	Yann	2016-02-03	934516879	Montijo	Avenida Plural	5	Não Tem	Não Tem	4500	1985-10-16	Ativo

Stored Procedure sp_desativar_fornecedor

Com este procedure podemos desativar um fornecedor, ou seja, mudar o seu estado para inativo. Neste exemplo vamos mudar o estado do fornecedor com o id 1 para inativo. Exemplo encontra-se no script teste.sql, linha 204.

ID	Nome	Data Registo	Telefone	Localidade	Empresa	Rua	Nr	Andar	Porta	Codigo Postal	Estado
1	Luis	2010-05-18	916458237	Setubal	CombusAll	Rua dos combustiveis	10	Não Tem	Não Tem	3150	Inativo

Stored Procedure sp_ativar_fornecedor

Com este procedure podemos ativar um fornecedor, ou seja, mudar o seu estado para ativo. Neste exemplo vamos mudar o estado do fornedor com o id 1 para ativo. Exemplo encontra-se no script teste.sql, linha 208

ID	Nome	Data Registo	Telefone	Localidade	Empresa	Rua	Nr	Andar	Porta	Codigo Postal	Estado
1	Luis	2010-05-18	916458237	Setubal	CombusAll	Rua dos combustiveis	10	Não Tem	Não Tem	3150	Ativo

.

Stored Procedure sp_alterar_venda_combustivel

Com este procedure conseguimos alterar o combustivel de uma venda. Neste exemplo vamos alterar na venda com o id 1 para o combustivel para o id 4. Exemplo encontra-se no script teste.sql, linha 212.

ID	ID Combustivel	Taxa de Iva	Litros	Custo	Data de Venda	Id Colaborador
1	4	0.23	15.00	18.00	2016-06-01	1

Stored Procedure sp_alterar_venda_colaborador

Com este procedure conseguimos alterar o colaborador de uma venda. Neste exemplo vamos alterar na venda com o id 1 para o colaborador com o id 19. Exemplo encontra-se no script teste.sql, linha 216.

ID	ID Combustivel	Taxa de Iva	Litros	Custo	Data de Venda	Id Colaborador
1	4	0.23	15.00	18.00	2016-06-01	19

Stored Procedure sp_alterar_venda_litros

Com este procedure conseguimos alterar os litros de uma venda. Neste exemplo vamos alterar na venda com o id 1 para 19 litros. Ao alterarmos os litros vamos por consequência alterar o custo associado a esta venda. Exemplo encontra-se no script teste.sql, linha 220.

ID	ID Combustivel	Taxa de Iva	Litros	Custo	Data de Venda	Id Colaborador
1	4	0.23	19.00	23.00	2016-06-01	19

Stored Procedure sp_alterar_venda_custo

Com este procedure conseguimos alterar o custo de uma venda. Neste exemplo vamos alterar na venda com o id 1 para que o custo seja de 40. Ao alterarmos o custo vamos por consequência alterar os litros associados a esta venda. Exemplo encontra-se no script teste.sql, linha 225.

ID	ID Combustivel	Taxa de Iva	Litros	Custo	Data de Venda	Id Colaborador
1	4	0.23	22.00	40.00	2016-06-01	19

Stored Procedure sp_alterar_cliente_nome

Com este procedure vamos alterar o nome de um cliente. Neste exemplo vamos alterar o cliente com o id 1 para que o nome seja Olee. Exemplo encontra-se no script teste.sql, linha 233.

ID	Nome	Data Registo	Telefone	Localidade	Pontos	Empresa	Nif	Data de Nascimento	Rua	Nr	Andar	Porta	Codigo_Postal	Estado
1	Olee	2016-01-01	915152789	Lisboa	437	Sadin	24587915	1992-05-12	Rua Alves Santos	5	1º	A	2500	Ativo

Stored Procedure sp_alterar_cliente_telefone

Com este procedure vamos alterar o telefone de um cliente. Neste exemplo vamos alterar o cliente com o id 1 para que o telefone seja 111111111. Exemplo encontra-se no script teste.sql, linha 237.

ID	Nome	Data Registo	Telefone	Localidade	Pontos	Empresa	Nif	Data de Nascimento	Rua	Nr	Andar	Porta	Codigo_Postal	Estado
1	Olee	2016-01-01	1111111	Lisboa	437	Sadin	24587915	1992-05-12	Rua Alves Santos	5	1º	A	2500	Ativo

Stored Procedure sp_alterar_cliente_localidade

Com este procedure vamos alterar a localidade de um cliente. Neste exemplo vamos alterar o cliente com o id 1 para o a localidade seja VIDA. Exemplo encontra-se no script teste.sql, linha 241.

ID	Nome	Data Registo	Telefone	Localidade	Pontos	Empresa	Nif	Data de Nascimento	Rua	Nr	Andar	Porta	Codigo_Postal	Estado
1	Olee	2016-01-01	1111111	VIDA	437	Sadin	24587915	1992-05-12	Rua Alves Santos	5	1º	A	2500	Ativo

Stored Procedure sp_alterar_cliente_empresa

Com este procedure vamos alterar a empresa de um cliente. Neste exemplo vamos alterar o cliente com o id 1 para que a empresa seja Chato. Exemplo encontra-se no script teste.sql, linha 245.

ID	Nome	Data Registo	Telefone	Localidade	Pontos	Empresa	Nif	Data de Nascimento	Rua	Nr	Andar	Porta	Codigo_Postal	Estado
1	Olee	2016-01-01	1111111	VIDA	437	Chato	24587915	1992-05-12	Rua Alves Santos	5	1º	A	2500	Ativo

Stored Procedure sp_alterar_cliente_nif

Com este procedure vamos alterar o nif de um cliente. Neste exemplo vamos alterar o cliente com o id 1 para que o nif seja 222222. Exemplo encontra-se no script teste.sql, linha 249.

ID	Nome	Data Registo	Telefone	Localidade	Pontos	Empresa	Nif	Data de Nascimento	Rua	Nr	Andar	Porta	Codigo_Postal	Estado
1	Olee	2016-01-01	1111111	VIDA	437	Chato	222222	1992-05-12	Rua Alves Santos	5	1º	A	2500	Ativo

Stored Procedure sp_fatura_cliente_data

Com este procedure conseguimos visualizar o número de faturas associadas a um cliente a partir do seu id, num dado espaço de tempo, intervalo de 1 mês, a partir de uma data. Neste exemplo vamos visualizar a informação do cliente 1 relativamente ao mês de junho. Exemplo encontra-se no script teste.sql, linha 254.

Nome	Numero de Faturas	Total Com Iva	Total Sem Iva
Olee	2	80.00	66.66

Stored Procedure sp_alterar_colaborador_nome

Com este procedure conseguimos alterar o nome do colaborador. Neste exemplo vamos alterar o nome do colaborador com o id 1 para OLEE. Exemplo encontra-se no script teste.sql, linha 262.

ID	Nome	Data Registo	Telefone	Localidade	Rua	Nr	Andar	Porta	Codigo Postal	Data de Nascimento	Estado
1	OLEE	2015-03-14	91458752	Setubal	Rua Super Homem	23	5º	G	2900	1990-05-17	Ativo

Stored Procedure sp_alterar_colaborador_telefone

Com este procedure conseguimos alterar o telefone de um colaborador. Neste exemplo vamos alterar o telefone do colaborador com o id 1 para 11111. Exemplo encontra-se no script teste.sql, linha 266.

ID	Nome	Data Registo	Telefone	Localidade	Rua	Nr	Andar	Porta	Codigo Postal	Data de Nascimento	Estado
1	OLEE	2015-03-14	111111	Setubal	Rua Super Homem	23	5º	G	2900	1990-05-17	Ativo

Stored Procedure sp_alterar_colaborador_localidade

Com este procedure conseguimos alterar a localidade de um colaborador. Neste exemplo vamos alterar a localidade do colaborador 1 para CHATO. Exemplo encontra-se no script teste.sql, linha 270.

ID	Nome	Data Registo	Telefone	Localidade	Rua	Nr	Andar	Porta	Codigo Postal	Data de Nascimento	Estado
1	OLEE	2015-03-14	111111	CHATO	Rua Super Homem	23	5º	G	2900	1990-05-17	Ativo

Stored Functions

Stored Function fu_existeCombustivel

Com esta function é possível identificar o id de um combustível a partir do seu nome e retorna o seu id.

Stored Function fu_calculaPontos

Com esta function é possível calcular quantos pontos existem para se dar a um cliente a partir do id de uma venda. Esta a partir do id vai buscar os litros da venda, calcula os pontos e retorna esse valor.

Stored Function fu_encontrarClienteId

Com esta function é possível encontrar o id de um cliente a partir do id de uma venda.

Stored Function fu_calcula_custo

Com esta function é possível calcular qual o custo da venda através dos litros, custo por litro e da taxa da iva. Após calcular o custo retorna esse valor.

Stored Function fu_calcula_litros

Com esta function é possível calcular quantos litros estão associados a uma venda através do custo, custo por litro e taxa de iva. Após calcular os litros retorna esse valor.

Trigger

Trigger tr_associar_pontos

Este trigger após um insert na tabela clienteVendas faz update aos pontos do cliente usando a função calculaPontos. Neste exemplo vamo registar a venda com id 24 no cliente com o id 1. Exemplo encontra-se no script teste.sql, linha 276.

Antes de registar a fatura:

ID	Nome	Data Registo	Telefone	Localidade	Pontos	Empresa	Nif	Data de Nascimento	Rua	Nr	Andar	Porta	Codigo_Postal	Estado
1	Olee	2016-01-01	1111111	VIDA	437	Chato	222222	1992-05-12	Rua Alves Santos	5	1º	A	2500	Ativo

Depois de registar a fatura:

ID	Nome	Data Registo	Telefone	Localidade	Pontos	Empresa	Nif	Data de Nascimento	Rua	Nr	Andar	Porta	Codigo_Postal	Estado
1	Olee	2016-01-01	1111111	VIDA	439	Chato	222222	1992-05-12	Rua Alves Santos	5	1º	A	2500	Ativo

Conclusão

No desenvolvimento do projeto de um sistema de software point-of-sale para a estação de serviço, foram identificadas as entidades e respetivos atributos, assim como as relações entre as entidades. Consequentemente foi possível a criação do Diagrama Entidade Relacionamento e, posteriormente, o Modelo Relacional. Estes deram origem à base de dados, que contém as respetivas relações e as views com a informação necessária para a gestão da estação de serviço e também os respetivos stored procedures, stored fuctions e triggers necessários.

Na primeira fase foram cumpridos todos os objetivos propostos e apesar de sentidas algumas dificuldades, estes foram realizados com sucesso.

As dificuldades sentidas na primeira fase surgiram maioritariamente no desenvolvimento do Diagrama Entidade Relacionamento e do Modelo Relacional devido à grande importância e impacto que estes iriam ter, mais tarde, no desenvolvimento da base dados. Nomeadamente no Diagrama Entidade Relacionamento, as dificuldades foram na implementação correta das relações entre as entidades; bem como no Modelo Relacional. Contudo, como já foi referido, todas as dificuldades foram ultrapassadas com êxito.

O desenvolvimento da primeira fase deste projeto foi fundamental, pois permitiu-nos consolidar os novos conhecimentos que nos foram transmitidos e aperfeiçoar as nossas competências relativas ao desenvolvimento de bases de dados.

Na segunda fase foram cumpridos, tal como na primeira fase, todos os objetivos propostos e nesta última fase não foram sentidas grandes dificuldades.

A grande dificuldade e provavelmente a única foi no desenvolvimento do trigger, mas esta deveu-se à falta de prática. Independentemente das dificuldades foi implementado com sucesso.

O desenvolvimento da segunda fase deste projeto, permitiu-nos controlar o funcionamento completo da base de dados. Este controlo levou a que possa ser efetuada praticamente qualquer tipo de ação sobre as diversas tabelas existentes com o fim de que o sistema pos seja um sucesso.

Em suma o desenvolvimento do sistema de software point-of-sale para a estação de serviço foi um grande sucesso e foram alcançados todos os objetivos propostos. Ao alcançar estes objetivos foi possível conhecer muito mais sobre o mundo de base de dados e agora, com este conhecimento, qualquer desafio, dentro destes conhecimentos desenvolvidos é possível de realizar.

Anexo A - create.sql

```
DROP DATABASE IF EXISTS pos; /*Apagar a base de dados se já existir */

CREATE DATABASE pos;/*criar a base dados*/

USE pos;/* usar a base de dados*/

/*Criação da tabela dos Clientes
*Com esta vamos conseguir obter as informações necessárias sobre um cliente que esteja registo na empresa
*/
CREATE TABLE Cliente
(
        cli_id INT AUTO_INCREMENT PRIMARY KEY,
   cli_nome VARCHAR (20) NOT NULL,
   cli_data_registo DATE NOT NULL,
   cli_telefone INT NOT NULL,
   cli_localidade VARCHAR(25) NOT NULL,
   cli_pontos INT DEFAULT 0 NOT NULL,
   cli_empresa VARCHAR (20) NOT NULL,
   cli_nif INT NOT NULL,
   cli_data_nascimento DATE NOT NULL,
   cli_rua VARCHAR (25) NOT NULL,
   cli_nr INT NOT NULL,
   cli_andar VARCHAR (10),
   cli_porta VARCHAR (10),
   cli_codigo_postal INT NOT NULL,
   cli_estado VARCHAR(20) NOT NULL DEFAULT 'Ativo'
);

/*Criação da tabela dos Colaborares da empresa
*Com estava vamos conseguir obter todas as informações nessárias sobre um colaborador que trabalha estação de serviço
*/
CREATE TABLE Colaborador
(
        col_id INT AUTO_INCREMENT PRIMARY KEY,
   col_nome VARCHAR (20) NOT NULL,
   col_data_registo DATE NOT NULL,
   col_telefone INT NOT NULL,
   col_localidade VARCHAR (25) NOT NULL,
   col_rua VARCHAR (25) NOT NULL,
   col_nr INT NOT NULL,
   col_andar VARCHAR (10),
   col_porta VARCHAR (10),
   col_codigo_postal INT NOT NULL,
```

```
   col_data_nascimento DATE NOT NULL,
   col_estado VARCHAR (20) NOT NULL DEFAULT 'Ativo'
);

/*Criação da tabela dos Fornecedores
*Com estava vamos conseguir obter todas as informações nessárias sobre um fornecedor de
combustivel
*/
CREATE TABLE Fornecedor
(
        for_id INT AUTO_INCREMENT PRIMARY KEY,
   for_nome VARCHAR (20) NOT NULL,
   for_data_registo DATE NOT NULL,
   for_telefone INT NOT NULL,
   for_localidade VARCHAR (25) NOT NULL,
   for_empresa VARCHAR (20) NOT NULL,
   for_rua VARCHAR (25) NOT NULL,
   for_nr INT NOT NULL,
   for_andar VARCHAR (10),
   for_porta VARCHAR (10),
   for_codigo_postal INT NOT NULL,
   for_estado VARCHAR(20) NOT NULL DEFAULT 'Ativo'
);

/*Criação da tabela Combustível
*Com esta tabela vamos saber tudo sobre o combustivel que é vendido na estação de serviço
*/
CREATE TABLE Combustivel
(
        co_id INT AUTO_INCREMENT PRIMARY KEY,
   co_custo_litro DOUBLE(16,2) NOT NULL,
   co_nome VARCHAR (20) NOT NULL,
   co_data_inicio DATE NOT NULL,
   co_data_fim DATE,
   co_fornecedor_id INT NOT NULL,
   co_stock INT NOT NULL,
   FOREIGN KEY(co_fornecedor_id)
                REFERENCES Fornecedor (for_id)
);

/*Criação da tabela Ofertas
*Com esta vamos saber tudo sobre as ofertas ofericidas aos clientes registados
*/
CREATE TABLE Ofertas
(
        of_id INT AUTO_INCREMENT PRIMARY KEY,
   of_nome VARCHAR (20) NOT NULL,
   of_pontos INT NOT NULL,
```

```
   of_data_inicio DATE NOT NULL,
   of_data_fim DATE,
   of_descricao VARCHAR (30) NOT NULL,
   of_categoria VARCHAR (20) NOT NULL,
   of_numero_ofertas INT,
   of_litros INT
);

/*Criação da tabela Vendas
*com esta tabela vamos saber tudo sobre as vendas efectuadas na estação de serviço
*/
CREATE TABLE Vendas
(
        ve_id INT AUTO_INCREMENT PRIMARY KEY,
   ve_id_combustivel INT NOT NULL,
   ve_taxa_iva DOUBLE(16,2) NOT NULL,
   ve_litros DOUBLE(16,2),
   ve_custo DOUBLE(16,2),
   ve_data_venda DATE NOT NULL,
   ve_id_colaborador INT NOT NULL,
        FOREIGN KEY(ve_id_combustivel)
                REFERENCES Combustivel(co_id),
        FOREIGN KEY(ve_id_colaborador)
                REFERENCES Colaborador(col_id)
);

/*Criacação da tabela Faturacao
*Com esta tabela vamos saber tudo sobre a faturação da estação de serviço
*/
CREATE TABLE Faturacao
(
        fa_id INT AUTO_INCREMENT PRIMARY KEY,
   fa_custo_total DOUBLE (16,2) NOT NULL,
   fa_data DATE NOT NULL,
   fa_iva DOUBLE (16,2) NOT NULL,
   fa_ve_id INT UNIQUE NOT NULL,
   fa_estado VARCHAR(20) DEFAULT 'Fechada',
   FOREIGN KEY (fa_ve_id)
                REFERENCES Vendas (ve_id)
        ON DELETE CASCADE

);

/*Criação da tabela Cliente
*Com esta tabela vamos saber que vendas têm cliente associado
*/
CREATE TABLE ClienteVendas
(
```

```
        cv_cliente_id INT NOT NULL,
   cv_vendas_id INT NOT NULL,
   FOREIGN KEY(cv_cliente_id)
                REFERENCES Cliente (cli_id),
        FOREIGN KEY(cv_vendas_id)
                REFERENCES Vendas (ve_id)
        ON DELETE CASCADE
);

/*Criacao da Tabela VendasOfertas
*Com esta vamos conseguir identificar as vendas que têm ofertas
*/
CREATE TABLE VendasOfertas
(
        vo_vendas_id INT NOT NULL,
   vo_ofertas_id INT NOT NULL,
   FOREIGN KEY(vo_vendas_id)
                REFERENCES Vendas (ve_id),
        FOREIGN KEY(vo_ofertas_id)
                REFERENCES Ofertas (of_id)
        ON DELETE CASCADE
);
```

Anexo B - logic.sql

```
USE pos;


/*View AutoridadeTributariaLoja
*Com esta view temos a informação necessária para enviar à autoridade tributaria
mensalmente
*/
CREATE OR REPLACE VIEW AutoridadeTributariaLoja AS
SELECT SUM(fa_custo_total) 'Lucro Com Iva', MONTHNAME(fa_data) Mes, SUM(fa_iva) 'Total
Iva', SUM(fa_custo_total - fa_iva) 'Lucro Sem Iva'
FROM Faturacao
WHERE YEAR(fa_data) = YEAR(CURRENT_DATE)
AND MONTH(fa_data) = MONTH(CURRENT_DATE - INTERVAL 1 MONTH);

/*View AutoridadeTributariaCliente
*Com esta view temos a informação necessária para enviar à autoridade tributaria
mensalmente sobre os clientes
*/
CREATE OR REPLACE VIEW AutoridadeTributariaCliente AS
SELECT cli_nome 'Cliente Nome', fa_iva 'Iva Pago', (fa_custo_total - fa_iva) 'Custos Sem Iva',
fa_custo_total 'Custo Com Iva', MONTHNAME(fa_data) Mes, cli_nif Nif
FROM Cliente
INNER JOIN ClienteVendas
ON cli_id = cv_cliente_id
INNER JOIN Vendas
ON cv_vendas_id = ve_id
INNER JOIN Faturacao
ON ve_id = fa_ve_id
WHERE YEAR(fa_data) = YEAR(CURDATE() )
AND MONTH(fa_data) = MONTH(CURDATE() - INTERVAL 1 MONTH);

/*View VolumeDeVendas
*Com esta view vamos conseguir identificar o volume de vendas mensal e também de todos os
mes em que houve vendas.
*/
CREATE OR REPLACE VIEW VolumeDeVendas AS
SELECT DISTINCT COUNT(ve_id) 'Numero de Vendas', SUM(ve_custo) 'Lucro', SUM(ve_litros)
'Litros Vendidos', MONTHNAME(ve_data_venda) Mes
FROM vendas
GROUP BY YEAR(ve_data_venda) = YEAR(CURRENT_DATE - INTERVAL 1 MONTH)
AND MONTH(ve_data_venda) = MONTH(CURRENT_DATE - INTERVAL 1 MONTH);

/*View NumeroOfertasPorCliente
*Com esta view vai ser possível ver o numero de ofertas adquiridas por cliente
*/
```

```
CREATE OR REPLACE VIEW NumeroOfertasPorCliente AS
SELECT cli_nome Nome, COUNT(vo_ofertas_id) 'Numero Ofertas' FROM cliente
INNER JOIN ClienteVendas
ON cli_id = cv_cliente_id
INNER JOIN Vendas
ON cv_vendas_id = ve_id
INNER JOIN VendasOfertas
ON ve_id = vo_vendas_id
INNER JOIN Ofertas
ON vo_ofertas_id = of_id
GROUP BY cli_nome;

/*View OfertasMaisVendidasCategoria
*Com esta view vai ser possível identificar as ofertas mais vendidas nas 2 categorias que
existem
*/
CREATE OR REPLACE VIEW OfertasMaisVendidasCategoria AS
SELECT DISTINCT of_categoria Categoria, of_nome Nome, COUNT(vo_ofertas_id)'Numero de
Ofertas Vendidas'
FROM Ofertas
INNER JOIN VendasOfertas
ON of_id = vo_ofertas_id
GROUP BY vo_ofertas_id
HAVING COUNT(vo_ofertas_id) = (SELECT MAX(vo_ofertas_id) FROM
                                                    (SELECT COUNT(*)
vo_ofertas_id
                                                    FROM vendasofertas
                                                    GROUP BY vo_ofertas_id) n);

/*View OfertasMaisProcuradasTop5
*Com esta view é possível saber quais as 5 ofertas mais procuradas pelos clientes
*/
CREATE OR REPLACE VIEW OfertaMaisProcurada AS
SELECT DISTINCT of_nome 'Nome', COUNT(vo_ofertas_id) 'Numero de Ofertas Vendidas'
FROM Ofertas
INNER JOIN VendasOfertas
ON of_id = vo_ofertas_id
GROUP BY vo_ofertas_id
ORDER BY COUNT(vo_ofertas_id) DESC
LIMIT 5;


/*View CombustivelMaisVendido
*Com esta view é possível saber qual o combústivel que foi mais vendido até hoje
*/
CREATE OR REPLACE VIEW CombustivelMaisVendido AS
SELECT co_nome 'Combustível', COUNT(ve_id) 'Numero de Vendas'
FROM Combustivel
```

```
INNER JOIN  Vendas
ON co_id = ve_id_combustivel
GROUP BY co_nome
HAVING COUNT(*) = (SELECT MAX(ve_id) FROM
                                        (SELECT COUNT(*) ve_id
                                                FROM Vendas
                GROUP BY ve_id_combustivel)n);

/*View CombustivelMenosVendido
*Com esta view é possível saber qual o combústivel que foi menos vendido até hoje
*/
CREATE OR REPLACE VIEW CombustivelMenosVendido AS
SELECT co_nome 'Combustível', COUNT(ve_id) 'Numero de       Vendas'
FROM Combustivel
INNER JOIN  Vendas
ON co_id = ve_id_combustivel
GROUP BY co_nome
HAVING COUNT(*) = (SELECT MIN(ve_id) FROM
                                        (SELECT COUNT(*) ve_id
                                                FROM Vendas
                GROUP BY ve_id_combustivel)n);

/*View NVendasColaborador
*Com esta view vai ser possível identificar o colaborador que efetuou mais vendas
*/
CREATE OR REPLACE VIEW NVendasColaborador AS
SELECT  DISTINCT col_nome Colaborador , COUNT(ve_id_colaborador) 'Numero de Vendas'
FROM Colaborador
INNER JOIN Vendas
ON col_id = ve_id_colaborador
GROUP BY col_nome;

/*View ClienteComMaisCompras
*Com esta view vai ser possível identificar o cliente que fez mais compras
*/
CREATE OR REPLACE VIEW ClienteComMaisCompras AS
SELECT cli_nome Nome, COUNT(cv_cliente_id) 'Numero de Compras'
FROM Cliente
INNER JOIN ClienteVendas
ON cli_id = cv_cliente_id
GROUP BY cli_nome
HAVING COUNT(*) = (SELECT MAX(cv_cliente_id)  FROM
                                        (SELECT COUNT(*) cv_cliente_id
                                                FROM ClienteVendas
                                                GROUP BY cv_cliente_id) n);


/*View NVendasSemmCliente
```

```
*Com esta view vai ser possível identificar quantas vendas efetuadas não têm cliente
*/
CREATE OR REPLACE VIEW NVendasSemCliente AS
SELECT COUNT(ve_id)'Numero de Vendas Sem Cliente'
FROM Vendas
LEFT JOIN clienteVendas
ON ve_id = cv_vendas_id
WHERE cv_vendas_id IS NULL;



/*View NVendasComCliente
*Com esta view vai ser possível identificar quantas vendas efetuadas têm um cliente registado
*/
CREATE OR REPLACE VIEW NVendasComCliente AS
SELECT COUNT(cv_cliente_id) 'Numero de Vendas Com cliente'
FROM ClienteVendas;


/* Fase 2 */

/**********************************VIEWS**********************************
********/

/*View TabelaCliente
*Com esta view vai ser possível mostrar a informação sobre todos os clientes
*/
CREATE OR REPLACE VIEW TabelaCliente AS
SELECT cli_id ID, cli_nome Nome, cli_data_registo 'Data Registo',
cli_telefone Telefone, cli_localidade Localidade, cli_pontos Pontos,
cli_empresa Empresa, cli_nif Nif, cli_data_nascimento 'Data de Nascimento',
cli_rua Rua, cli_nr Nr, cli_andar Andar, cli_porta Porta, cli_codigo_postal 'Codigo_Postal',
cli_estado Estado
FROM Cliente;

/*View TabelaColaborador
*Com esta view vamos conseguir visualizar todas as informações sobre os colaboradores
*/
CREATE OR REPLACE VIEW TabelaColaborador AS
SELECT col_id ID, col_nome Nome, col_data_registo 'Data Registo',
col_telefone Telefone, col_localidade Localidade, col_rua Rua, col_nr Nr,
col_andar Andar, col_porta Porta, col_codigo_postal 'Codigo Postal',
col_data_nascimento 'Data de Nascimento', col_estado Estado
FROM Colaborador;

/*View TabelaFornecedor
*Com esta view vamos conseguir visualizar todas as informações sobre os fornecedores
*/
```

```
CREATE OR REPLACE VIEW TabelaFornecedor AS
SELECT for_id ID, for_nome Nome, for_data_registo 'Data Registo',
for_telefone Telefone, for_localidade Localidade, for_empresa Empresa,
for_rua Rua, for_nr Nr, for_andar Andar, for_porta Porta, for_codigo_postal 'Codigo Postal',
for_estado Estado
FROM Fornecedor;

/*View TabelaCombustivel
*Com esta view vamos conseguir visualizar todas as informações sobre os combustiveis
*/
CREATE OR REPLACE VIEW TabelaCombustivel AS
SELECT co_id ID, co_custo_litro 'Custo por Litro',
co_nome Nome, co_data_inicio 'Data de Inicio',
co_data_fim 'Data de Fim', co_fornecedor_id 'ID Fornecedor',
co_stock Stock
FROM Combustivel;

/*View TabelaOfertas
*Com esta view vamos ser possível visualizar toda a informação sobre a tabela ofertas
*/
CREATE OR REPLACE VIEW TabelaOfertas AS
SELECT of_id ID, of_nome Nome, of_pontos Pontos,
of_data_inicio 'Data de Inicio', of_data_fim 'Data de Fim',
of_descricao Descricao, of_categoria Categoria,
of_numero_ofertas 'Numero de Ofertas', of_litros 'Litros'
FROM Ofertas
ORDER BY Categoria;

//*View TabelaVendas
*Com esta view é possível visualizar a informação sobre as vendas
*/
CREATE OR REPLACE VIEW TabelaVendas AS
SELECT ve_id ID, ve_id_Combustivel 'ID Combustivel',
ve_taxa_iva 'Taxa de Iva', ve_litros 'Litros',
ve_custo 'Custo', ve_data_venda 'Data de Venda', ve_id_colaborador 'Id Colaborador'
FROM Vendas;

/*View TabelaFaturacao
*Com esta view é possível visualizar as informações sobre as faturações
*/
CREATE OR REPLACE VIEW TabelaFaturacao AS
SELECT fa_id ID, fa_custo_total 'Custo Total',
fa_data Data, fa_iva Iva, fa_ve_id 'ID Venda', fa_estado 'Estado'
FROM Faturacao;

/*View TabelaClienteVenda
*Com esta view é possível visualizar as informações sobre a tabela cliente vendas
*/
```

```
CREATE OR REPLACE VIEW TabelaClienteVenda AS
SELECT cv_cliente_id 'Id Cliente', cv_vendas_id'Id Venda'
FROM clientevendas
ORDER BY cv_cliente_id;

/*View TabelaVendaOferta
*Com esta view é possível visualizar as informações sobre a tabela venda Oferta
*/
CREATE OR REPLACE VIEW TabelaVendaOferta AS
SELECT vo_vendas_id 'Id Venda', vo_ofertas_id'Id Oferta'
FROM vendasofertas
ORDER BY vo_vendas_id;

/*View OfertaMenosProcurada
*Com esta view é possível visualizar as ofertas menos vendidas
*/
CREATE OR REPLACE VIEW OfertaMenosProcurada AS
SELECT DISTINCT of_nome 'Nome', COUNT(vo_ofertas_id) 'Numero de Ofertas Vendidas'
FROM Ofertas
INNER JOIN VendasOfertas
ON of_id = vo_ofertas_id
GROUP BY vo_ofertas_id
HAVING COUNT(*) > 0
ORDER BY COUNT(vo_ofertas_id) ASC
LIMIT 5;

 /*View nr_oferta_cliente
*Com esta view é possível visualizar todos os clientes, quantas faturas existem em seu nome, o
total com iva e sem iva.
*/
CREATE OR REPLACE VIEW nr_fatura_cliente AS
 SELECT cli_nome 'Nome', COUNT(cv_cliente_id) 'Numero de Faturas', SUM(fa_custo_total)
'Total Com Iva', (SUM(fa_custo_total) - SUM(fa_iva)) AS 'Total Sem Iva'
 FROM cliente
 INNER JOIN clientevendas
 ON cli_id = cv_cliente_id
 INNER JOIN faturacao
 ON fa_ve_id = cv_vendas_id
 GROUP BY cli_nome
 ORDER BY cli_nome ASC;

/**********************STORED
PROCEDURES************************************/

/*Procedure registar_cliente
*Stored Procedure que permite registar clientes de modo a que nenhum valor fique a null
*/
DROP PROCEDURE IF EXISTS sp_registar_cliente;
```

```
DELIMITER //
CREATE PROCEDURE sp_registar_cliente (cli_nome VARCHAR(20), cli_data_registo DATE,
cli_telefone INT, cli_localidade VARCHAR(25), cli_pontos int, cli_empresa VARCHAR(20),
cli_nif INT, cli_data_nascimento DATE, cli_rua VARCHAR(25), cli_nr INT, cli_andar
VARCHAR(10), cli_porta VARCHAR(10), cli_codigo_postal INT)
BEGIN

IF(ISNULL(cli_andar)) THEN
SET cli_andar = 'Não Tem';
END IF;

IF(ISNULL(cli_porta)) THEN
SET cli_porta = 'Não Tem';
END IF;

INSERT INTO Cliente (cli_nome, cli_data_registo, cli_telefone, cli_localidade, cli_pontos,
cli_empresa, cli_nif, cli_data_nascimento, cli_rua, cli_nr, cli_andar, cli_porta,
cli_codigo_postal) values (cli_nome, cli_data_registo, cli_telefone, cli_localidade, cli_pontos,
cli_empresa, cli_nif, cli_data_nascimento, cli_rua, cli_nr, cli_andar, cli_porta,
cli_codigo_postal);

END//
DELIMITER ;

/*Procedure registar colaborador
*Stored Procedure que permite registar colaboradores de modo a que nenhum valor fique a
null
*/
DROP PROCEDURE IF EXISTS sp_registar_colaborador;
DELIMITER //
CREATE PROCEDURE sp_registar_colaborador (col_nome VARCHAR(20), col_data_registo
DATE, col_telefone INT, col_localidade VARCHAR(25), col_rua VARCHAR(25), col_nr INT,
col_andar VARCHAR(10), col_porta VARCHAR(10), col_codigo_postal INT,
col_data_nascimento DATE)
BEGIN

IF(ISNULL(col_andar)) THEN
SET col_andar = 'Não Tem';
END IF;

IF(ISNULL(col_porta)) THEN
SET col_porta = 'Não Tem';
END IF;

INSERT INTO Colaborador (col_nome, col_data_registo, col_telefone, col_localidade, col_rua,
col_nr, col_andar, col_porta, col_codigo_postal, col_data_nascimento) values (col_nome,
col_data_registo, col_telefone, col_localidade, col_rua, col_nr, col_andar, col_porta,
col_codigo_postal, col_data_nascimento) ;
```

```
END//
DELIMITER ;

/*Procedure registar_forncedor
*Com este procedure conseguimos registar fornecedores
*/
DROP PROCEDURE IF EXISTS sp_registar_fornecedor;
DELIMITER //
CREATE PROCEDURE sp_registar_fornecedor(for_nome VARCHAR(20), for_data_registo DATE,
for_telefone INT, for_localidade VARCHAR(25), for_empresa VARCHAR(20), for_rua
VARCHAR(25), for_nr INT, for_andar VARCHAR(10), for_porta VARCHAR(10), for_codigo_postal
INT)
BEGIN

IF(ISNULL(for_andar)) THEN
SET for_andar = 'Não Tem';
END IF;

IF(ISNULL(for_porta)) THEN
SET for_porta = 'Não Tem';
END IF;

INSERT INTO Fornecedor (for_nome, for_data_registo, for_telefone, for_localidade,
for_empresa, for_rua, for_nr, for_andar, for_porta, for_codigo_postal) values (for_nome,
for_data_registo, for_telefone, for_localidade, for_empresa, for_rua, for_nr, for_andar,
for_porta, for_codigo_postal);

END //
DELIMITER ;


/*Procedure registar_combustivel
*Com este procedure conseguimos registar combustiveis
*/
DROP PROCEDURE IF EXISTS sp_registar_combustivel;
DELIMITER //
CREATE PROCEDURE sp_registar_combustivel(co_custo_litro DOUBLE(16,2), co_nome
VARCHAR(20), co_fornecedor_id INT, co_stockRecebido INT)
BEGIN

DECLARE ID INT;
DECLARE stock INT;
DECLARE estado VARCHAR(20);

SELECT for_estado INTO estado FROM fornecedor WHERE for_id = co_fornecedor_id;

IF(estado = 'Inativo' OR ISNULL(estado)) THEN
```

```
SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Fornecedor Inativo ou inexistente, não pode adicionar fatura`,
   MYSQL_ERRNO = `erron`;
END IF;

SET ID = fu_existeCombustivel(co_nome);
SELECT co_stock INTO stock FROM combustivel WHERE co_id = ID;

IF (ISNULL(ID) OR ID = '') THEN

   INSERT INTO Combustivel (co_custo_litro, co_nome, co_data_inicio, co_fornecedor_id,
co_stock) VALUES (co_custo_litro, co_nome, CURDATE(), co_fornecedor_id,
co_stockRecebido);

ELSE

        SET co_stockRecebido = co_stockRecebido + stock;

        INSERT INTO Combustivel (co_custo_litro, co_nome, co_data_inicio, co_fornecedor_id,
co_stock) VALUES (co_custo_litro, co_nome, CURDATE(), co_fornecedor_id,
co_stockRecebido);

        UPDATE Combustivel
        SET co_data_fim = curdate(),
        co_stock = 0
        WHERE co_id = ID;

END IF;

END//
DELIMITER ;

/*Procedure registar_oferta
*Com este procedure conseguimos registar combustiveis
*/
DROP PROCEDURE IF EXISTS sp_registar_oferta;
DELIMITER //
CREATE PROCEDURE sp_registar_oferta(of_nome VARCHAR(20), of_pontos INT, of_data_inicio
DATE, of_data_fim DATE, of_descricao VARCHAR(30), of_categoria VARCHAR(30),
of_numero_ofertas INT, of_litros INT)
BEGIN

IF(ISNULL(of_litros)) THEN
```

```
        INSERT INTO Ofertas (of_nome, of_pontos, of_data_inicio, of_data_fim, of_descricao,
of_categoria, of_numero_ofertas) VALUES (of_nome, of_pontos, of_data_inicio, of_data_fim,
of_descricao, of_categoria, of_numero_ofertas);

ELSE
        INSERT INTO Ofertas (of_nome, of_pontos, of_data_inicio, of_data_fim, of_descricao,
of_categoria, of_litros) VALUES (of_nome, of_pontos, of_data_inicio, of_data_fim,
of_descricao, of_categoria, of_litros);

END IF;
END//
DELIMITER ;

/*Procedure registar venda
* Com este procedure conseguimos registar uma venda
*/
DROP PROCEDURE IF EXISTS sp_registar_venda;
DELIMITER //
CREATE PROCEDURE sp_registar_venda(ve_id_combustivel INT, ve_litros DOUBLE(16,2),
ve_custo DOUBLE(16,2), ve_id_colaboradorEntrada INT)
BEGIN

DECLARE precoLitro double;
DECLARE ve_taxa_iva DOUBLE;
DECLARE data_fim DATE;
DECLARE estado VARCHAR(20);

SET ve_taxa_iva = 0.23;

SELECT co_custo_litro INTO precoLitro FROM Combustivel WHERE co_id = ve_id_combustivel;
SELECT co_data_fim INTO data_fim FROM Combustivel WHERE co_id = ve_id_combustivel;
SELECT col_estado INTO estado FROM colaborador WHERE col_id =
ve_id_colaboradorEntrada;

IF (estado = 'Inativo' OR ISNULL(estado)) THEN
SIGNAL SQLSTATE
  'ERR0R'
SET
  MESSAGE_TEXT = `Colaborador inválido ou inexistente, escolha outro`,
  MYSQL_ERRNO = `erron`;

END IF;

IF(data_fim IS NOT NULL) THEN
SIGNAL SQLSTATE
  'ERR0R'
SET
  MESSAGE_TEXT = `Combustivel inválido, insira outro id`,
```

```
   MYSQL_ERRNO = `erron`;
END IF;

IF(ISNULL(ve_litros) AND ISNULL(ve_custo)) THEN
SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Insira custo ou litros para continuar`,
   MYSQL_ERRNO = `erron`;
END IF;

IF(ISNULL(ve_litros)) THEN

        SET ve_litros = fu_calcula_litros(ve_custo, precoLitro, ve_taxa_iva);

END IF;

IF(ISNULL(ve_custo)) THEN
        SET ve_custo =fu_calcula_custo(ve_litros, precoLitro , ve_taxa_iva) ;

END IF;

UPDATE combustivel
set co_stock = co_stock - ve_litros
WHERE co_id = ve_id_combustivel;

INSERT INTO Vendas(ve_id_combustivel, ve_taxa_iva, ve_litros, ve_custo, ve_data_venda,
ve_id_colaborador) VALUES(ve_id_combustivel, ve_taxa_iva, ve_litros, ve_custo, CURDATE(),
ve_id_colaboradorEntrada);

END //
DELIMITER ;


/*Procedure registar_fatura
*Com este procedure conseguimos registar uma fatura
*/
DROP PROCEDURE IF EXISTS sp_registar_fatura;
DELIMITER //
CREATE PROCEDURE sp_registar_fatura(fa_ve_idEntrada INT, cliente_id INT)
BEGIN

DECLARE fa_custo_total DOUBLE;
DECLARE iva DOUBLE;
DECLARE vendas_id INT;
DECLARE estado VARCHAR(20);

SELECT cli_estado INTO estado FROM cliente WHERE cli_id = cliente_id;
```

```
SELECT ve_custo INTO fa_custo_total FROM vendas WHERE ve_id = fa_ve_idEntrada;
SELECT ve_taxa_iva INTO iva FROM vendas WHERE ve_id = fa_ve_idEntrada;
SELECT ve_id INTO vendas_id FROM vendas WHERE fa_ve_idEntrada = ve_id;

IF(estado = 'Inativo' OR ISNULL(estado)) THEN

SIGNAL SQLSTATE
  'ERR0R'
SET
  MESSAGE_TEXT = `Cliente Inativo ou inexistente, não pode adicionar fatura`,
  MYSQL_ERRNO = `erron`;
END IF;

IF(ISNULL(vendas_id)) THEN

  SIGNAL SQLSTATE
  'ERR0R'
SET
  MESSAGE_TEXT = `Não existe venda com esse id`,
  MYSQL_ERRNO = `erron`;

END IF;

SET iva = iva * fa_custo_total;

INSERT INTO Faturacao (fa_custo_total, fa_data, fa_iva, fa_ve_id) values(fa_custo_total,
CURDATE(), iva, fa_ve_idEntrada);

CALL sp_relacionar_clienteVenda(cliente_id, fa_ve_idEntrada);

END//
DELIMITER ;

/*Procedure relacionar_clienteVenda
*Com este procedure conseguimos relacionar um cliente com uma venda
*/
DROP PROCEDURE IF EXISTS sp_relacionar_clienteVenda;
DELIMITER //
CREATE PROCEDURE sp_relacionar_clienteVenda(cv_cliente_id INT, cv_vendas_id INT)
BEGIN

INSERT INTO ClienteVendas (cv_cliente_id, cv_vendas_id) values(cv_cliente_id, cv_vendas_id);

END//
DELIMITER ;

/*Procedure relacionar_vendaOferta
*Com este procedure conseguimos relacionar uma venda com uma oferta
```

```
*/
DROP PROCEDURE IF EXISTS sp_relacionar_vendaOferta;
DELIMITER //
CREATE PROCEDURE sp_relacionar_vendaOferta(vo_vendas_id INT, vo_ofertas_id INT)
BEGIN

DECLARE tipo VARCHAR(20);
DECLARE combVenda INT;
DECLARE litros INT;
DECLARE cliente_pontos INT;
Declare pontos INT;
DECLARE cliente_id INT;
DECLARE nr_ofertas INT;

SELECT of_categoria INTO tipo FROM ofertas WHERE vo_ofertas_id = of_id;
SELECT co_id INTO combVenda FROM combustivel INNER JOIN vendas ON ve_id =
vo_vendas_id WHERE ve_id_combustivel = co_id;
SELECT of_litros INTO litros FROM ofertas WHERE vo_ofertas_id = of_id;
SELECT cli_id INTO cliente_id FROM cliente INNER JOIN vendas ON vo_vendas_id = ve_id
INNER JOIN clientevendas ON cv_vendas_id = ve_id WHERE cv_cliente_id = cli_id;
SELECT cli_pontos INTO cliente_pontos FROM Cliente WHERE cli_id = cliente_id;
SELECT of_pontos INTO pontos FROM ofertas WHERE vo_ofertas_id = of_id;
SELECT of_numero_ofertas INTO nr_ofertas FROM ofertas WHERE vo_ofertas_id = of_id;

IF((cliente_pontos - pontos) < 0) THEN
        SIGNAL SQLSTATE
                'ERR0R'
        SET
                MESSAGE_TEXT = `Não tem pontos suficientes`,
                MYSQL_ERRNO = `erron`;

ELSEIF(nr_ofertas = 0) THEN
        SIGNAL SQLSTATE
                'ERROR'
        SET
                MESSAGE_TEXT = `Não há stock dessa oferta`,
                MYSQL_ERRNO = `erron`;
ELSE

        UPDATE cliente
        SET cli_pontos = cli_pontos - pontos
        WHERE cli_id = cliente_id;

INSERT INTO VendasOfertas (vo_vendas_id, vo_ofertas_id) values(vo_vendas_id,
vo_ofertas_id);

        IF(tipo = 'Descontos') THEN
                UPDATE combustivel
```

```
                SET co_stock = co_stock - litros
                WHERE co_id = combVenda;

        ELSE
                UPDATE ofertas
                SET of_numero_ofertas = of_numero_ofertas - 1
                WHERE vo_ofertas_id = of_id;

        END IF;


END IF;

END//
DELIMITER ;


/*Procedure produtos_fatura
*Com este procedure conseguimos visualizar os detalhes de uma fatura a partir do seu id
*/
DROP PROCEDURE IF EXISTS sp_produtos_fatura;
DELIMITER //
CREATE PROCEDURE sp_produtos_fatura(fatura_id INT)
BEGIN

DECLARE vendas_id INT;
DECLARE vo_oferta_id INT;
DECLARE fa_idGuarda INT;

SELECT fa_id INTO fa_idGuarda FROM faturacao WHERE fa_id = fatura_id;

IF(ISNULL(fa_idGuarda) OR fa_idGuarda = '') THEN

        SIGNAL SQLSTATE
                'ERROR'
        SET
                MESSAGE_TEXT = `fatura inexistente`,
                MYSQL_ERRNO = `erron`;

END IF;

SELECT ve_id INTO vendas_id FROM vendas INNER JOIN faturacao ON fa_ve_id = ve_id WHERE
fa_id = fatura_id;
SELECT COUNT(vo_ofertas_id) INTO vo_oferta_id FROM vendasOfertas WHERE vo_vendas_id
= vendas_id;

IF(vo_oferta_id > 0) THEN
```

```
        SELECT fa_id 'Id Fatura', co_nome Combustivel, ve_litros Litros, of_nome Oferta
        FROM faturacao
        INNER JOIN vendas
        ON fa_ve_id = ve_id
        INNER JOIN Combustivel
        ON ve_id_combustivel = co_id
        INNER JOIN VendasOfertas
        ON vo_vendas_id = ve_id
        INNER JOIN Ofertas
        ON vo_ofertas_id = of_id
        WHERE fa_id = fatura_id;

 ELSE

        SELECT fa_id 'Id Fatura', co_nome Combustivel, ve_litros Litros
        FROM faturacao
        INNER JOIN vendas
        ON fa_ve_id = ve_id
        INNER JOIN Combustivel
        ON ve_id_combustivel = co_id
        WHERE fa_id = fatura_id;

END IF;

END//
DELIMITER ;

/*Procedure listar_ofertaS_categoria
*Com este procedure conseguimos listar as ofertas por ordem de mais vendida a partir de uma
categoria
*/
DROP PROCEDURE IF EXISTS sp_listar_ofertas_categoria;
DELIMITER //
CREATE PROCEDURE sp_listar_ofertas_categoria(categoria VARCHAR(20))
BEGIN
SELECT DISTINCT of_categoria Categoria, of_nome Nome, COUNT(vo_ofertas_id)'Numero de
Ofertas Vendidas'
FROM Ofertas
INNER JOIN VendasOfertas
ON of_id = vo_ofertas_id
WHERE of_categoria = categoria
GROUP BY vo_ofertas_id
ORDER BY COUNT(vo_ofertas_id) DESC;

END //
DELIMITER ;
```

```
/*Procedure produtos_fatura
*Com este procedure conseguimos visualizar os detalhes de uma fatura a partir do seu id
*/
DROP PROCEDURE IF EXISTS sp_resumo_fatura;
DELIMITER //
CREATE PROCEDURE sp_resumo_fatura(fatura_id INT)
BEGIN

	SELECT fa_id 'Id Fatura',cli_nome Nome, co_nome Combustivel, ve_litros Litros,
COUNT(vo_ofertas_id) 'Numero Ofertas', SUM(fa_custo_total) 'Total Com Iva',
(SUM(fa_custo_total) - SUM(fa_iva)) 'Total Sem Iva', fa_estado Estado
	FROM faturacao
	INNER JOIN vendas
	ON fa_ve_id = ve_id
	INNER JOIN Combustivel
	ON ve_id_combustivel = co_id
  INNER JOIN vendasOfertas
  ON vo_vendas_id = ve_id
  INNER JOIN clientevendas
  ON cv_vendas_id = ve_id
  INNER JOIN cliente
  ON cli_id = cv_cliente_id
	WHERE fa_id = fatura_id;

END//
DELIMITER ;

/*Procedure volume_anual_mes
*Com este procedure conseguimos visualizar os detalhes de todos os meses do ultimo ano
*/
DROP PROCEDURE IF EXISTS sp_volume_anual_mes;
DELIMITER //
CREATE PROCEDURE sp_volume_anual_mes(ano DATE)
BEGIN

SELECT DISTINCT COUNT(ve_id) 'Numero Vendas', SUM(fa_iva) 'Iva Pago',
(SUM(fa_custo_total) - SUM(fa_iva)) 'Total Sem Iva', SUM(fa_custo_total) 'Total Com Iva',
MONTHNAME(fa_data) Mes, cli_nif Nif
FROM Cliente
INNER JOIN ClienteVendas
ON cli_id = cv_cliente_id
INNER JOIN Vendas
ON cv_vendas_id = ve_id
INNER JOIN Faturacao
ON ve_id = fa_ve_id
WHERE YEAR(fa_data) = YEAR(ano)
GROUP BY Mes;
```

```
END//
DELIMITER ;

/*Procedure remover_venda
*Com este procedure conseguimos remover uma venda e fatura se estiver associada
*/
DROP PROCEDURE IF EXISTS sp_remover_venda;
 DELIMITER //
CREATE PROCEDURE sp_remover_venda(ve_idEntrada INT)
BEGIN

DECLARE Litros DOUBLE;
DECLARE Combustivel_id INT;
DECLARE fatura_id INT;

SELECT ve_litros INTO Litros FROM Vendas WHERE ve_id = ve_idEntrada;
SELECT co_id INTO Combustivel_id FROM Combustivel INNER JOIN vendas ON co_id =
ve_id_combustivel WHERE ve_id = ve_idEntrada;
SELECT fa_id INTO fatura_id FROM faturacao INNER JOIN vendas ON fa_ve_id = ve_id WHERE
ve_id = ve_idEntrada;

IF(ISNULL(ve_idEntrada)) THEN
SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Insira um id valido`,
   MYSQL_ERRNO = `erron`;
END IF;


UPDATE combustivel
SET co_stock = co_stock + Litros
WHERE Combustivel_id = co_id;

SET FOREIGN_KEY_CHECKS = 0;
DELETE FROM vendas WHERE ve_id = ve_idEntrada;
SET FOREIGN_KEY_CHECKS = 1;

IF (fatura_id IS NOT NULL) THEN

SET FOREIGN_KEY_CHECKS = 0;
DELETE FROM faturacao WHERE fa_id = fatura_id;
DELETE FROM vendasofertas WHERE ve_idEntrada = vo_vendas_id;
DELETE FROM clienteVendas WHERE ve_idEntrada = cv_vendas_id;
SET FOREIGN_KEY_CHECKS = 1;

END IF;
```

```
END //
DELIMITER ;

/*Procedure fechar_fatura
*Com este procedure conseguimos fechar uma fatura a partir do id
*/
DROP PROCEDURE IF EXISTS sp_fechar_fatura;
DELIMITER //
CREATE PROCEDURE sp_fechar_fatura(fatura_id INT)
BEGIN

UPDATE faturacao
SET fa_estado = 'Fechado'
WHERE fa_id = fatura_id;

CALL sp_resumo_fatura(fatura_id);

END //
DELIMITER ;

/*Procedure abrir_fatura
*Com este procedure conseguimos abrir uma fatura a partir do id
*/
DROP PROCEDURE IF EXISTS sp_abrir_fatura;
DELIMITER //
CREATE PROCEDURE sp_abrir_fatura(fatura_id INT)
BEGIN

UPDATE faturacao
SET fa_estado = 'Aberta'
WHERE fa_id = fatura_id;

CALL sp_resumo_fatura(fatura_id);

END //
DELIMITER ;

/*Procedure adcionar_combustivel
*Com este procedure conseguimos adicionar mais combustivel ao que já existe */
DROP PROCEDURE IF EXISTS sp_adicionar_combustivel;
DELIMITER //
CREATE PROCEDURE sp_adicionar_combustivel(fatura_idEntrada INT, quantidade INT)
BEGIN

DECLARE estado VARCHAR(20);
DECLARE fatura_vendas_id INT;
DECLARE preco_litro double;
DECLARE taxa_iva double;
```

```
DECLARE combustivel INT;

SELECT fa_estado INTO estado FROM faturacao WHERE fa_id = fatura_idEntrada;

IF(estado = 'Fechado') THEN
SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `fatura fechada`,
   MYSQL_ERRNO = `erron`;
END IF;

SELECT fa_ve_id INTO fatura_vendas_id FROM faturacao WHERE fa_id = fatura_idEntrada;
SELECT co_custo_litro INTO preco_litro FROM Combustivel INNER JOIN vendas ON ve_id =
fatura_vendas_id WHERE ve_id_combustivel = co_id;
SELECT ve_taxa_iva INTO taxa_iva FROM vendas WHERE ve_id = fatura_vendas_id;
SELECT co_id INTO combustivel FROM combustivel INNER JOIN vendas ON ve_id =
fatura_vendas_id WHERE ve_id_combustivel = co_id;

UPDATE vendas
SET ve_litros = ve_litros + quantidade,
ve_custo = fu_calcula_custo(ve_litros, preco_litro, taxa_iva)
WHERE ve_id = fatura_vendas_id;

UPDATE combustivel
SET co_stock = co_stock - quantidade
WHERE co_id = combustivel;

END //
DELIMITER ;

/*Procedure alterar_stock_combustivel
*Com este procedure conseguimos adicionar stock a um combustivel */
DROP PROCEDURE IF EXISTS sp_alterar_stock_combustivel;
DELIMITER //
CREATE PROCEDURE sp_alterar_stock_combustivel(combustivel_id INT, quantidade INT)
BEGIN

DECLARE data DATE;
SELECT co_data_fim INTO data FROM combustivel WHERE co_id = combustivel_id;

IF(data IS NOT NULL) THEN

SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Combustivel inválido`,
   MYSQL_ERRNO = `erron`;
```

```
END IF;

UPDATE combustivel
SET co_stock = co_stock + quantidade
WHERE co_id = combustivel_id;

END //
DELIMITER ;

/*Procedure remove_combustivel_fatura
*Com este procedure conseguimos remover uma quantidade de combustivel de uma fatura */
DROP PROCEDURE IF EXISTS sp_remove_combustivel_fatura;
DELIMITER //
CREATE PROCEDURE sp_remove_combustivel_fatura(fatura_idEntrada INT, quantidade INT)
BEGIN

DECLARE estado VARCHAR(20);
DECLARE fatura_vendas_id INT;
DECLARE preco_litro double;
DECLARE taxa_iva double;
DECLARE combustivel INT;
DECLARE litros INT;

SELECT fa_estado INTO estado FROM faturacao WHERE fa_id = fatura_idEntrada;


IF(estado = 'Fechado') THEN
SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `fatura fechada`,
   MYSQL_ERRNO = `erron`;

ELSEIF(estado = '' OR ISNULL(estado)) THEN

SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `fatura inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

SELECT fa_ve_id INTO fatura_vendas_id FROM faturacao WHERE fa_id = fatura_idEntrada;
SELECT co_custo_litro INTO preco_litro FROM Combustivel INNER JOIN vendas ON ve_id =
fatura_vendas_id WHERE ve_id_combustivel = co_id;
SELECT ve_taxa_iva INTO taxa_iva FROM vendas WHERE ve_id = fatura_vendas_id;
```

```
SELECT co_id INTO combustivel FROM combustivel INNER JOIN vendas ON ve_id =
fatura_vendas_id WHERE ve_id_combustivel = co_id;
SELECT ve_litros INTO litros FROM vendas WHERE ve_id = fatura_vendas_id;

IF((litros - quantidade) <= 0) THEN
SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Quantia inválida, quantidade fica menor que 0`,
   MYSQL_ERRNO = `erron`;
END IF;

UPDATE vendas
SET ve_litros = ve_litros - quantidade,
ve_custo = fu_calcula_custo(ve_litros, preco_litro, taxa_iva)
WHERE ve_id = fatura_vendas_id;

UPDATE combustivel
SET co_stock = co_stock + quantidade
WHERE co_id = combustivel;

END //
DELIMITER ;

/*Procedure remover_oferta
*Com este procedure conseguimos remover uma oferta */
DROP PROCEDURE IF EXISTS sp_remover_oferta;
DELIMITER //
CREATE PROCEDURE sp_remover_oferta(oferta_id INT)
BEGIN

        SET FOREIGN_KEY_CHECKS = 0;
   DELETE FROM ofertas WHERE of_id = oferta_id;
   SET FOREIGN_KEY_CHECKS = 1;

END //
DELIMITER ;

/*Procedure alterar_oferta_nome
*Com este procedure conseguimos alterar o nome de uma oferta */
DROP PROCEDURE IF EXISTS sp_alterar_oferta_nome;
DELIMITER //
CREATE PROCEDURE sp_alterar_oferta_nome(oferta_id INT, of_nomeEntrada VARCHAR(20))
BEGIN

        UPDATE ofertas
   SET of_nome = of_nomeEntrada
   WHERE of_id = oferta_id;
```

```
END //
DELIMITER ;

/*Procedure alterar_oferta_pontos
*Com este procedure conseguimos alterar os pontos uma oferta */
DROP PROCEDURE IF EXISTS sp_alterar_oferta_pontos;
DELIMITER //
CREATE PROCEDURE sp_alterar_oferta_pontos(oferta_id INT, oferta_pontos INT)
BEGIN

        UPDATE ofertas
  SET of_pontos = oferta_pontos
  WHERE of_id = oferta_id;

END //
DELIMITER ;

/*Procedure alterar_oferta_data_inicio
*Com este procedure conseguimos alterar a data inicio de uma oferta */
DROP PROCEDURE IF EXISTS sp_alterar_oferta_data_inicio;
DELIMITER //
CREATE PROCEDURE sp_alterar_oferta_data_inicio(oferta_id INT, oferta_data_inicio DATE)
BEGIN

DECLARE data_fim DATE;

SELECT of_data_fim INTO data_fim FROM ofertas WHERE of_id = oferta_id;

IF(oferta_data_inicio < data_fim) THEN

        UPDATE ofertas
  SET of_data_inicio = oferta_data_inicio
  WHERE of_id = oferta_id;

ELSE

SIGNAL SQLSTATE
  'ERR0R'
SET
  MESSAGE_TEXT = `Data de inicio maior que data de fim`,
  MYSQL_ERRNO = `erron`;

END IF;

END //
DELIMITER ;
```

```
/*Procedure alterar_oferta_data_fim
*Com este procedure conseguimos alterar a data fim de uma oferta */
DROP PROCEDURE IF EXISTS sp_alterar_oferta_data_fim;
DELIMITER //
CREATE PROCEDURE sp_alterar_oferta_data_fim(oferta_id INT, oferta_data_fim DATE)
BEGIN

DECLARE data_inicio DATE;

SELECT of_data_inicio INTO data_inicio FROM ofertas WHERE of_id = oferta_id;

IF(oferta_data_fim > data_inicio) THEN

        UPDATE ofertas
  SET of_data_fim = oferta_data_fim
  WHERE of_id = oferta_id;

ELSE
SIGNAL SQLSTATE
  'ERR0R'
SET
  MESSAGE_TEXT = `Data de fim menor que data de inicio`,
  MYSQL_ERRNO = `erron`;

END IF;

END //
DELIMITER ;

/*Procedure alterar_oferta_descricao
*Com este procedure conseguimos alterar a descricao de uma oferta */
DROP PROCEDURE IF EXISTS sp_alterar_oferta_descricao;
DELIMITER //
CREATE PROCEDURE sp_alterar_oferta_descricao(oferta_id INT, oferta_descricao
VARCHAR(25))
BEGIN

        UPDATE ofertas
  SET of_descricao = oferta_descricao
  WHERE of_id = oferta_id;

END //
DELIMITER ;

/*Procedure alterar_oferta_categoria
*Com este procedure conseguimos alterar a categoria de uma oferta */
DROP PROCEDURE IF EXISTS sp_alterar_oferta_categoria;
DELIMITER //
```

```
CREATE PROCEDURE sp_alterar_oferta_categoria(oferta_id INT, oferta_categoria
VARCHAR(25))
BEGIN
IF(oferta_categoria = 'Material' OR oferta_categoria = 'Descontos') THEN
        UPDATE ofertas
  SET of_categoria = oferta_categoria
  WHERE of_id = oferta_id;

ELSE
SIGNAL SQLSTATE
  'ERR0R'
SET
  MESSAGE_TEXT = `Categoria Inválida`,
  MYSQL_ERRNO = `erron`;

END IF;

END //
DELIMITER ;

/*Procedure alterar_oferta_numeroOfertas
*Com este procedure conseguimos alterar o numero de ofertas de uma oferta */
DROP PROCEDURE IF EXISTS sp_alterar_oferta_numeroOfertas;
DELIMITER //
CREATE PROCEDURE sp_alterar_oferta_numeroOfertas(oferta_id INT, oferta_numero INT)
BEGIN

DECLARE categoria VARCHAR(20);
SELECT of_categoria INTO categoria FROM ofertas WHERE of_id = oferta_id;

IF(categoria = 'Material') THEN

        UPDATE ofertas
  SET of_numero_ofertas = oferta_numero
  WHERE of_id = oferta_id;

ELSE

SIGNAL SQLSTATE
  'ERR0R'
SET
  MESSAGE_TEXT = `Oferta inválida`,
  MYSQL_ERRNO = `erron`;

END IF;

END //
DELIMITER ;
```

```
/*Procedure alterar_oferta_litros
*Com este procedure conseguimos alterar os litros de uma oferta */
DROP PROCEDURE IF EXISTS sp_alterar_oferta_litros;
DELIMITER //
CREATE PROCEDURE sp_alterar_oferta_litros(oferta_id INT, oferta_litros INT)
BEGIN

DECLARE categoria VARCHAR(20);
SELECT of_categoria INTO categoria FROM ofertas WHERE of_id = oferta_id;

IF(categoria = 'Descontos') THEN

        UPDATE ofertas
   SET of_litros = oferta_litros
   WHERE of_id = oferta_id;

ELSE
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Não pode alterar os litros desta oferta`,
   MYSQL_ERRNO = `erron`;

END IF;

END //
DELIMITER ;

/*Procedure remover_fatura
*Com este procedure conseguimos remover uma oferta */
DROP PROCEDURE IF EXISTS sp_remover_fatura;
DELIMITER //
CREATE PROCEDURE sp_remover_fatura(fatura_id INT)
BEGIN

DECLARE vendas_id INT;
SELECT ve_id INTO vendas_id FROM vendas INNER JOIN faturacao ON fa_ve_id = ve_id WHERE
fa_id = fatura_id;

        SET FOREIGN_KEY_CHECKS = 0;
   DELETE FROM faturacao WHERE fa_id = fatura_id;
   DELETE FROM clientevendas WHERE vendas_id = cv_vendas_id;
   SET FOREIGN_KEY_CHECKS = 1;

END //
DELIMITER ;
```

```
/*Procedure remover_oferta_fatura
*Com este procedure conseguimos remover uma oferta de uma fatura */
DROP PROCEDURE IF EXISTS sp_remover_oferta_fatura;
DELIMITER //
CREATE PROCEDURE sp_remover_oferta_fatura(fatura_id INT, oferta_nome VARCHAR(25))
BEGIN

DECLARE vendas_id INT;
DECLARE oferta_id VARCHAR(25);

SELECT ve_id INTO vendas_id FROM vendas INNER JOIN faturacao ON fa_ve_id = ve_id WHERE
fa_id = fatura_id;
SELECT of_id INTO oferta_id FROM ofertas WHERE of_nome = oferta_nome;


        SET FOREIGN_KEY_CHECKS = 0;
  DELETE FROM vendasOfertas WHERE vo_ofertas_id = oferta_id AND vo_vendas_id =
vendas_id;
  SET FOREIGN_KEY_CHECKS = 1;

END //
DELIMITER ;

/*Procedure desativar_cliente
*Com este procedure conseguimos desativar um cliente registado*/
DROP PROCEDURE IF EXISTS sp_desativar_cliente;
DELIMITER //
CREATE PROCEDURE sp_desativar_cliente(cli_idEntrada INT)
BEGIN

DECLARE cliente_id INT;
SELECT cli_id INTO cliente_id FROM cliente WHERE cli_id = cli_idEntrada;

IF(ISNULL(cliente_id) OR cliente_id = '') THEN
  SIGNAL SQLSTATE
  'ERR0R'
SET
  MESSAGE_TEXT = `Cliente inexistente`,
  MYSQL_ERRNO = `erron`;

END IF;

UPDATE cliente
SET cli_estado = 'Inativo'
WHERE cli_id = cli_idEntrada;

END //
DELIMITER ;
```

```
/*Procedure ativar_cliente
*Com este procedure conseguimos ativar um cliente registado */
DROP PROCEDURE IF EXISTS sp_ativar_cliente;
DELIMITER //
CREATE PROCEDURE sp_ativar_cliente(cli_idEntrada INT)
BEGIN

DECLARE cliente_id INT;
SELECT cli_id INTO cliente_id FROM cliente WHERE cli_id = cli_idEntrada;

IF(ISNULL(cliente_id) OR cliente_id = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Cliente inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE cliente
SET cli_estado = 'Ativo'
WHERE cli_id = cli_idEntrada;

END //
DELIMITER ;

/*Procedure desativar_colaborador
*Com este procedure conseguimos desativar um colaborador*/
DROP PROCEDURE IF EXISTS sp_desativar_colaborador;
DELIMITER //
CREATE PROCEDURE sp_desativar_colaborador(col_idEntrada INT)
BEGIN

DECLARE colaborador_id INT;
SELECT col_id INTO colaborador_id FROM colaborador WHERE col_id = col_idEntrada;

IF(ISNULL(colaborador_id) OR colaborador_id = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Colaborador inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE colaborador
SET col_estado = 'Inativo'
```

```
WHERE col_id = col_idEntrada;

END //
DELIMITER ;

/*Procedure ativar_colaborador
*Com este procedure conseguimos ativar um colaborador registado */
DROP PROCEDURE IF EXISTS sp_ativar_colaborador;
DELIMITER //
CREATE PROCEDURE sp_ativar_colaborador(col_idEntrada INT)
BEGIN

DECLARE colaborador_id INT;
SELECT col_id INTO colaborador_id FROM colaborador WHERE col_id = col_idEntrada;

IF(ISNULL(colaborador_id) OR colaborador_id = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Colaborador inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE colaborador
SET col_estado = 'Ativo'
WHERE col_id = col_idEntrada;

END //
DELIMITER ;

/*Procedure desativar_fornecedor
*Com este procedure conseguimos desativar um fornecedor*/
DROP PROCEDURE IF EXISTS sp_desativar_fornecedor;
DELIMITER //
CREATE PROCEDURE sp_desativar_fornecedor(for_idEntrada INT)
BEGIN

DECLARE fornecedor_id INT;
SELECT for_id INTO fornecedor_id FROM fornecedor WHERE for_id = for_idEntrada;

IF(ISNULL(fornecedor_id ) OR fornecedor_id  = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Fornecedor inexistente`,
   MYSQL_ERRNO = `erron`;
```

```
END IF;

UPDATE fornecedor
SET for_estado = 'Inativo'
WHERE for_id = for_idEntrada;

END //
DELIMITER ;

/*Procedure ativar_fornecedor
*Com este procedure conseguimos ativar um fornecedor*/
DROP PROCEDURE IF EXISTS sp_ativar_fornecedor;
DELIMITER //
CREATE PROCEDURE sp_ativar_fornecedor(for_idEntrada INT)
BEGIN

DECLARE fornecedor_id INT;
SELECT for_id INTO fornecedor_id FROM fornecedor WHERE for_id = for_idEntrada;

IF(ISNULL(fornecedor_id ) OR fornecedor_id  = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Fornecedor inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE fornecedor
SET for_estado = 'Ativo'
WHERE for_id = for_idEntrada;

END //
DELIMITER ;

/*Procedure alterar_venda_combustivel
*Com este procedure é possível alterar o combustivel de uma venda a partir do seu id
*/
DROP PROCEDURE IF EXISTS sp_alterar_venda_combustivel;
DELIMITER //
CREATE PROCEDURE sp_alterar_venda_combustivel(venda_id INT, combustivel_id INT)
BEGIN

DECLARE data DATE;
DECLARE v_id INT;
SELECT co_data_fim INTO data FROM combustivel WHERE co_id = combustivel_id;
SELECT ve_id INTO v_id FROM vendas WHERE ve_id = venda_id;
```

```
IF(ISNULL(v_id)) THEN
 SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `venda Invalida`,
   MYSQL_ERRNO = `erron`;
END IF;

IF(data IS NOT NULL) THEN
 SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Combustivel Invalido`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE vendas
SET ve_id_combustivel = combustivel_id
WHERE ve_id = venda_id;

END//
DELIMITER ;

/*Procedure alterar_venda_colaborador
*Com este procedure é possível alterar o colaborador de uma venda a partir do seu id
*/
DROP PROCEDURE IF EXISTS sp_alterar_venda_colaborador;
DELIMITER //
CREATE PROCEDURE sp_alterar_venda_colaborador(venda_id INT, colaborador_id INT)
BEGIN

DECLARE c_id INT;
DECLARE v_id INT;

SELECT col_id INTO c_id FROM colaborador WHERE col_id = colaborador_id;
SELECT ve_id INTO v_id FROM vendas WHERE ve_id = venda_id;

IF(ISNULL(v_id)) THEN
 SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `venda Invalida`,
   MYSQL_ERRNO = `erron`;
END IF;

IF(ISNULL(c_id) OR c_id = '') THEN
 SIGNAL SQLSTATE
```

```
   'ERR0R'
SET
   MESSAGE_TEXT = `Colaborador Inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE vendas
SET ve_id_colaborador = colaborador_id
WHERE ve_id = venda_id;

END//
DELIMITER ;

/*Procedure alterar_venda_litros
*Com este procedure é possível alterar a quantidade de litros de combustivel de uma venda a
partir do seu id
*/
DROP PROCEDURE IF EXISTS sp_alterar_venda_litros;
DELIMITER //
CREATE PROCEDURE sp_alterar_venda_litros(venda_id INT, litros INT)
BEGIN

DECLARE v_id INT;
DECLARE preco_litros INT;
DECLARE fatura_id INT;
DECLARE custo DOUBLE;
DECLARE iva DOUBLE;

SELECT ve_id INTO v_id FROM vendas WHERE ve_id = venda_id;
SELECT co_custo_litro INTO preco_litros FROM combustivel INNER JOIN vendas ON ve_id =
venda_id WHERE ve_id_combustivel = co_id;
SELECT fa_id INTO fatura_id FROM faturacao WHERE fa_ve_id = venda_id;
SELECT ve_taxa_iva INTO iva FROM vendas WHERE ve_id = venda_id;
SET custo = fu_calcula_custo(litros, preco_litros, iva);

IF(ISNULL(v_id)) THEN
 SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `venda Invalida`,
   MYSQL_ERRNO = `erron`;
END IF;


UPDATE vendas
SET ve_litros = litros, ve_custo = custo
WHERE ve_id = venda_id;
```

```
IF(fatura_id IS NOT NULL) THEN

UPDATE faturacao
SET fa_custo_total = custo
WHERE fa_id = fatura_id;

END IF;

END//
DELIMITER ;

/*Procedure alterar_venda_custo
*Com este procedure é possível alterar o custo de uma venda a partir do seu id
*/
DROP PROCEDURE IF EXISTS sp_alterar_venda_custo;
DELIMITER //
CREATE PROCEDURE sp_alterar_venda_custo(venda_id INT, custo INT)
BEGIN

DECLARE v_id INT;
DECLARE preco_litros DOUBLE;
DECLARE iva DOUBLE;
DECLARE litros INT;
DECLARE fatura_id INT;

SELECT ve_id INTO v_id FROM vendas WHERE ve_id = venda_id;
SELECT co_custo_litro INTO preco_litros FROM combustivel INNER JOIN vendas ON ve_id =
venda_id WHERE ve_id_combustivel = co_id;
SELECT fa_id INTO fatura_id FROM faturacao WHERE fa_ve_id = venda_id;
SELECT ve_taxa_iva INTO iva FROM vendas WHERE ve_id = venda_id;
SET litros = fu_calcula_litros(custo, preco_litros, iva);

IF(ISNULL(v_id)) THEN
 SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `venda Invalida`,
   MYSQL_ERRNO = `erron`;
END IF;


UPDATE vendas
SET ve_litros = litros, ve_custo = custo
WHERE ve_id = venda_id;

IF(fatura_id IS NOT NULL) THEN
```

```
UPDATE faturacao
SET fa_custo_total = custo
WHERE fa_id = fatura_id;

END IF;

END//

DELIMITER ;

/*Procedure alterar_cliente_nome
*Com este procedure é possível alterar o nome de um cliente
*/
DROP PROCEDURE IF EXISTS sp_alterar_cliente_nome;
DELIMITER //
CREATE PROCEDURE sp_alterar_cliente_nome(cli_idEntrada INT, nome VARCHAR(20))
BEGIN

DECLARE cliente_id INT;
SELECT cli_id INTO cliente_id FROM cliente WHERE cli_id = cli_idEntrada;

IF(ISNULL(cliente_id) OR cliente_id = '') THEN
  SIGNAL SQLSTATE
  'ERR0R'
SET
  MESSAGE_TEXT = `Cliente inexistente`,
  MYSQL_ERRNO = `erron`;

END IF;

UPDATE cliente
SET cli_nome = nome
WHERE cli_id = cli_idEntrada;

END//
DELIMITER ;

/*Procedure alterar_cliente_telefone
*Com este procedure é possível alterar o telefone de um cliente
*/
DROP PROCEDURE IF EXISTS sp_alterar_cliente_telefone;
DELIMITER //
CREATE PROCEDURE sp_alterar_cliente_telefone(cli_idEntrada INT, telefone INT)
BEGIN

DECLARE cliente_id INT;
SELECT cli_id INTO cliente_id FROM cliente WHERE cli_id = cli_idEntrada;
```

```
IF(ISNULL(cliente_id) OR cliente_id = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Cliente inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE cliente
SET cli_telefone = telefone
WHERE cli_id = cli_idEntrada;

END//
DELIMITER ;

/*Procedure alterar_cliente_localidade
*Com este procedure é possível alterar a localidade de um cliente
*/
DROP PROCEDURE IF EXISTS sp_alterar_cliente_localidade;
DELIMITER //
CREATE PROCEDURE sp_alterar_cliente_localidade(cli_idEntrada INT, localidade VARCHAR(20))
BEGIN

DECLARE cliente_id INT;
SELECT cli_id INTO cliente_id FROM cliente WHERE cli_id = cli_idEntrada;

IF(ISNULL(cliente_id) OR cliente_id = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Cliente inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE cliente
SET cli_localidade = localidade
WHERE cli_id = cli_idEntrada;

END//
DELIMITER ;

/*Procedure alterar_cliente_empresa
*Com este procedure é possível alterar a empresa de um cliente
*/
DROP PROCEDURE IF EXISTS sp_alterar_cliente_empresa;
DELIMITER //
```

```
CREATE PROCEDURE sp_alterar_cliente_empresa(cli_idEntrada INT, empresa VARCHAR(20))
BEGIN

DECLARE cliente_id INT;
SELECT cli_id INTO cliente_id FROM cliente WHERE cli_id = cli_idEntrada;

IF(ISNULL(cliente_id) OR cliente_id = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Cliente inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE cliente
SET cli_empresa = empresa
WHERE cli_id = cli_idEntrada;

END//
DELIMITER ;

/*Procedure alterar_cliente_nif
*Com este procedure é possível alterar o nif de um cliente
*/
DROP PROCEDURE IF EXISTS sp_alterar_cliente_nif;
DELIMITER //
CREATE PROCEDURE sp_alterar_cliente_nif(cli_idEntrada INT, nif INT)
BEGIN

DECLARE cliente_id INT;
SELECT cli_id INTO cliente_id FROM cliente WHERE cli_id = cli_idEntrada;

IF(ISNULL(cliente_id) OR cliente_id = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Cliente inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE cliente
SET cli_nif = nif
WHERE cli_id = cli_idEntrada;

END//
DELIMITER ;
```

```
/*Procedure alterar_colaborador_nome
*Com este procedure é possível alterar o nome de um cliente
*/
DROP PROCEDURE IF EXISTS sp_alterar_colaborador_nome;
DELIMITER //
CREATE PROCEDURE sp_alterar_colaborador_nome(col_idEntrada INT, nome VARCHAR(20))
BEGIN

DECLARE colaborador_id INT;
SELECT col_id INTO colaborador_id FROM colaborador WHERE col_id = col_idEntrada;

IF(ISNULL(colaborador_id) OR colaborador_id = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Colaborador inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE colaborador
SET col_nome = nome
WHERE col_id = col_idEntrada;

END//
DELIMITER ;

/*Procedure alterar_colaborador_telefone
*Com este procedure é possível alterar o telefone de um cliente
*/
DROP PROCEDURE IF EXISTS sp_alterar_colaborador_telefone;
DELIMITER //
CREATE PROCEDURE sp_alterar_colaborador_telefone(col_idEntrada INT, telefone INT)
BEGIN

DECLARE colaborador_id INT;
SELECT col_id INTO colaborador_id FROM colaborador WHERE col_id = col_idEntrada;

IF(ISNULL(colaborador_id) OR colaborador_id = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Colaborador inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;
```

```
UPDATE colaborador
SET col_telefone = telefone
WHERE col_id = col_idEntrada;

END//
DELIMITER ;

/*Procedure alterar_colaborador_localidade
*Com este procedure é possível alterar a localidade de um colaborador
*/
DROP PROCEDURE IF EXISTS sp_alterar_colaborador_localidade;
DELIMITER //
CREATE PROCEDURE sp_alterar_colaborador_localidade(col_idEntrada INT, localidade
VARCHAR(20))
BEGIN

DECLARE colaborador_id INT;
SELECT col_id INTO colaborador_id FROM colaborador WHERE col_id = col_idEntrada;

IF(ISNULL(colaborador_id) OR colaborador_id = '') THEN
   SIGNAL SQLSTATE
   'ERR0R'
SET
   MESSAGE_TEXT = `Colaborador inexistente`,
   MYSQL_ERRNO = `erron`;

END IF;

UPDATE colaborador
SET col_localidade = localidade
WHERE col_id = col_idEntrada;

END//
DELIMITER ;


/*View fatura_cliente_data
*Com esta view é possível visualizar uma fatura de um cliente, num mes, a partir do seu id e de
uma data, quantas faturas existem em seu nome, o total com iva e sem iva.
*/
DROP PROCEDURE IF EXISTS sp_fatura_cliente_data;
DELIMITER //
CREATE PROCEDURE sp_fatura_cliente_data(data DATE, cliente_id INT)
BEGIN

SELECT cli_nome 'Nome', COUNT(cv_cliente_id) 'Numero de Faturas', SUM(fa_custo_total)
'Total Com Iva', (SUM(fa_custo_total) - SUM(fa_iva)) AS 'Total Sem Iva'
FROM cliente
```

```
INNER JOIN clientevendas
ON cli_id = cv_cliente_id
INNER JOIN faturacao
ON fa_ve_id = cv_vendas_id
WHERE YEAR(fa_data) = YEAR(data)
AND MONTH(fa_data) = MONTH(DATA)
AND cli_id = cliente_id;

END //
DELIMITER ;

/**********************STORED
FUNCTIONS************************************/

/*Function existeCombustivel
*Retorna o id do combustivel a partir do nome
*/
DROP FUNCTION IF EXISTS fu_existeCombustivel;
DELIMITER $$
CREATE FUNCTION fu_existeCombustivel(co_nomeRecebido VARCHAR(20))
RETURNS INT
BEGIN

DECLARE ID INT;

SELECT co_id INTO ID FROM COMBUSTIVEL WHERE co_nomeRecebido = co_nome AND
ISNULL(co_data_fim);

RETURN ID;
END $$
DELIMITER ;

/*Function calculaPontos
*Retorna quantos pontos foram ganhos na venda
*/
DROP FUNCTION IF EXISTS fu_calculaPontos;
DELIMITER $$
CREATE FUNCTION fu_calculaPontos(ve_idRecebido INT)
RETURNS INT
BEGIN

DECLARE pontos INT;
DECLARE litros DOUBLE;

SELECT ve_litros INTO litros FROM vendas WHERE ve_id = ve_idRecebido;

SET pontos = litros;
```

```
RETURN pontos;
END $$
DELIMITER ;

/*Function encontrarIdCliente
*Retorna Id do cliente
*/
DROP FUNCTION IF EXISTS fu_encontrarClienteId;
DELIMITER $$
CREATE FUNCTION fu_encontrarClienteId(ve_idRecebido INT)
RETURNS INT
BEGIN

DECLARE Id INT;
SELECT cv_cliente_id INTO Id FROM clienteVendas WHERE cv_vendas_id = ve_idRecebido;


RETURN Id;
END $$
DELIMITER ;

/*Function calcula_sem_custo
*Calcular custo dos litros e retorna esse custo
*/
DROP FUNCTION IF EXISTS fu_calcula_custo;
DELIMITER $$
CREATE FUNCTION fu_calcula_custo(ve_litros int, co_custo_litro double, ve_taxa_iva double)
RETURNS DECIMAL(10,2)
BEGIN

DECLARE custo DECIMAL;
SET custo = ((ve_litros * co_custo_litro) * ve_taxa_iva) + (ve_litros * co_custo_litro);
RETURN custo;

END$$

   DELIMITER ;

/*Function calcula_sem_litros
* Calcular litros por custo
*/
DROP FUNCTION IF EXISTS fu_calcula_litros;
DELIMITER $$
CREATE FUNCTION fu_calcula_litros(ve_custo double, co_custo_litro double , ve_taxa_iva
double)
RETURNS DECIMAL(10,2)
BEGIN
```

```
RETURN((ve_custo - (ve_custo * ve_taxa_iva)) / co_custo_litro);

END$$

DELIMITER ;

/************************TRIGGERS*************************************/


DROP TRIGGER IF EXISTS tr_associar_pontos;
DELIMITER $$
CREATE TRIGGER tr_associar_pontos AFTER INSERT ON clienteVendas
FOR EACH ROW
BEGIN

DECLARE pontos DOUBLE;

SET pontos = fu_calculaPontos(NEW.cv_vendas_id);

UPDATE Cliente
SET cli_pontos = cli_pontos + pontos
WHERE cli_id = NEW.cv_cliente_id;

END $$
DELIMITER ;
```

Anexo C - populate.sql

```
USE pos;

/*Insererir dados na tabela Cliente*/
CALL sp_registar_cliente ('Jonas', str_to_date('2016.01.01','%Y.%m.%d'), 915152789, 'Lisboa',
500, 'Sadin', 24587915, str_to_date('1992.05.12', '%Y.%m.%d'), "Rua Alves Santos", 5,
'1º','A',2500);
CALL sp_registar_cliente ('Rui', str_to_date('2015.05.12','%Y.%m.%d'), 916243897, 'Setubal',
245, 'TerraCotaa', 45135798, str_to_date('1989.03.20', '%Y.%m.%d'), "Rua Jacinto Andrade", 7,
null, null, 2900);
CALL sp_registar_cliente ('Andre', str_to_date('2016.03.11','%Y.%m.%d'), 911892543, 'Lisboa',
458,'Landa', 29874561, str_to_date('1995.06.04', '%Y.%m.%d'), "Rua Moreira OL", 10,
'2º','B',5500);
CALL sp_registar_cliente ('Lopes', str_to_date('2015.07.20','%Y.%m.%d'), 963879534, 'Almada',
500, 'Grouje', 24587915, str_to_date('1993.10.25', '%Y.%m.%d'), "Avenida Europa", 4,
'3º','C',2910);
CALL sp_registar_cliente ('Paulo', str_to_date('2014.12.01','%Y.%m.%d'), 934587326, 'Cascais',
78, 'Kool', 13458794, str_to_date('1990.05.07', '%Y.%m.%d'), "Avenida Portal", 5, null, null,
5470);
CALL sp_registar_cliente ('Pedro', str_to_date('2013.11.20', '%Y.%m.%d'), 925554123, 'Faro' ,
90,'TicTravel', 154222,  str_to_date('1988.11.20', '%Y.%m.%d'), "Avenida Trip" , 6 , null, null,
451);
CALL sp_registar_cliente ('Jose', str_to_date('2016.01.11','%Y.%m.%d'), 910000458, 'Porto' ,
545 , 'Moveit', 14587988, str_to_date('1995.10.25','%Y.%m.%d'), "Rua Jose DelWhiskey", 41,
null, null, 2541 );
CALL sp_registar_cliente ('Diana', str_to_date('2014.02.21','%Y.%m.%d'), 920125487, 'Seixal',
1425, 'JabbaTaxi', 15548798, str_to_date('1980.10.05','%Y.%m.%d'), "Rua do Cavaco", 7, null,
null, 2840);
CALL sp_registar_cliente ('Miguel', str_to_date('2015.03.25', '%Y.%m.%d'), 920111523,
'Corroios', 5000, 'DingDong', 14457821, str_to_date('1996.05.14','%Y.%m.%d'), "Praceta
Afonso Henriques", 8 , null, null, 2694);
CALL sp_registar_cliente ('Luis', str_to_date('2013.02.09', '%Y.%m.%d'), 920113243, 'Setubal',
700, 'DingDong', 55612456, str_to_date('1994.08.14','%Y.%m.%d'), "Algum lado", 3 , null, null,
2234);
CALL sp_registar_cliente ('Petra', str_to_date('2015.12.25', '%Y.%m.%d'), 920111523, 'Angola',
5000, 'Praias', 2468123, str_to_date('1992.05.20','%Y.%m.%d'), "Rua em Africa", 2 , null, null,
0003);
CALL sp_registar_cliente ('Neo', str_to_date('1999.03.25', '%Y.%m.%d'), 920111523, 'Matrix',
5000, 'Zion', 548913256, str_to_date('1978.05.14','%Y.%m.%d'), "Unknown Street", 0 , null,
null, 9999);
CALL sp_registar_cliente ('Ruben', str_to_date('2013.10.04', '%Y.%m.%d'), 920111523, 'Lisboa',
5000, 'Empresa X', 4549412, str_to_date('1993.06.24','%Y.%m.%d'), "Rua Malmeqer Nao é", 14
, null, null, 8432);
CALL sp_registar_cliente ('Carvas', str_to_date('2012.03.23', '%Y.%m.%d'), 920111523,
'Algures', 4999, 'Empresa Y', 14457821, str_to_date('1991.01.11','%Y.%m.%d'), "Planeta
Escondido", 01 , null, null, 2002);
```

```
CALL sp_registar_cliente ('Polus', str_to_date('2011.02.25', '%Y.%m.%d'), 920111523, 'Loonar',
5000, 'Street' , 1466221, str_to_date('1992.05.13','%Y.%m.%d'), "Korner Surfer", 10 , null, null,
2694);
CALL sp_registar_cliente ('Romolus', str_to_date('2011.02.25', '%Y.%m.%d'), 920111523,
'Loonar', 5000, 'Street' , 10000021, str_to_date('1992.05.14','%Y.%m.%d'), "Korner Surfer", 11 ,
null, null, 2694);
CALL sp_registar_cliente ('Holis', str_to_date('2016.04.25', '%Y.%m.%d'), 92434433, 'Luna',
5000, 'Empresa X' , 100443321, str_to_date('1995.06.14','%Y.%m.%d'), "BLAAAA", 2 , null, null,
1294);
CALL sp_registar_cliente ('Coru', str_to_date('2016.05.20', '%Y.%m.%d'), 920124223, 'Loonar',
5000, 'Street' , 10000021, str_to_date('1994.05.14','%Y.%m.%d'), "Korner Ridge", 3 , null, null,
2645);
CALL sp_registar_cliente ('Gajo', str_to_date('2015.04.03', '%Y.%m.%d'), 923231523, 'Place',
230, 'QUALQA' , 1342400021, str_to_date('1996.02.04','%Y.%m.%d'), "Barker", 10 , null, null,
2694);
CALL sp_registar_cliente ('20Ultimo', str_to_date('2016.05.30', '%Y.%m.%d'), 91123829,
'CabouEste', 5000, 'Bica' , 10000021, str_to_date('1991.01.14','%Y.%m.%d'), "Vista de
Setubal", 01 , null, null, 1900);

/*Inserir dados na tabela Colaborador*/
CALL sp_registar_colaborador ('Kore', str_to_date('2015.03.14','%Y.%m.%d'), 91458752,
'Setubal', 'Rua Super Homem', 23, '5º', 'G', 2900, str_to_date('1990.05.17','%Y.%m.%d'));
CALL sp_registar_colaborador ('Yann', str_to_date('2016.02.03','%Y.%m.%d'), 934516879,
'Montijo','Avenida Plural', 5, null, null, 4500, str_to_date('1985.10.16','%Y.%m.%d'));
CALL sp_registar_colaborador ('Karlos', str_to_date('2014.10.22','%Y.%m.%d'),
924571284,'Almada','Rua dos Peixes', 8, '1º','Esquerdo', 5400,
str_to_date('1991.05.14','%Y.%m.%d'));
CALL sp_registar_colaborador ('Lara', str_to_date('2015.12.20','%Y.%m.%d'), 914758234,
'Setubal', 'Rua Destruida', 1, null, null, 2910, str_to_date('1993.04.07','%Y.%m.%d'));
CALL sp_registar_colaborador ('Mario', str_to_date('2016.04.01','%Y.%m.%d'), 916289341,
'Lisboa', 'Avenida 5 Espanha', 5, '4º', 'A', 3550, str_to_date('1991.01.08','%Y.%m.%d'));
CALL sp_registar_colaborador ('Carlos', str_to_date('2014.05.06','%Y.%m.%d'), 921458221,
'Setubal', 'Avenida Peixinho Dourado', 8, '2º', 'B', 2456, str_to_date('1992.02.12','%Y.%m.%d'));
CALL sp_registar_colaborador ('Cecilia', str_to_date('2014.11.15','%Y.%m.%d'), 921114589,
'Seixal', 'Rua D.Dinis', 5, null, null, 2458, str_to_date('1991.01.31','%Y.%m.%d'));
CALL sp_registar_colaborador ('Maró', str_to_date('2012.11.15','%Y.%m.%d'), 932312319,
'Setubal', 'Rua D.Dinas', 2, null, null, 2233, str_to_date('1991.01.04','%Y.%m.%d'));
CALL sp_registar_colaborador ('Kool', str_to_date('2014.12.12','%Y.%m.%d'), 921177489,
'Lisboa', 'Rua Repetida', 5, null, null, 2458, str_to_date('1991.01.31','%Y.%m.%d'));
CALL sp_registar_colaborador ('kola', str_to_date('2014.01.16','%Y.%m.%d'), 923232323,
'Setubal', 'Rua Sao Jaquim', 39, null, null, 2758, str_to_date('1991.03.31','%Y.%m.%d'));
CALL sp_registar_colaborador ('kolo', str_to_date('2014.02.13','%Y.%m.%d'), 921132389,
'Montijo', 'Rua Alentejo', 10, null, null, 8768, str_to_date('1991.05.31','%Y.%m.%d'));
CALL sp_registar_colaborador ('kole', str_to_date('2014.06.14','%Y.%m.%d'), 92232389, 'New
York', 'Avenida Ar', 2, null, null, 2776, str_to_date('1991.04.30','%Y.%m.%d'));
CALL sp_registar_colaborador ('Paulo', str_to_date('2012.10.15','%Y.%m.%d'), 92311145,
'Seixal', 'Marques Ines', 10, null, null, 6558, str_to_date('1992.03.31','%Y.%m.%d'));
```

```
CALL sp_registar_colaborador ('Caló', str_to_date('2013.09.11','%Y.%m.%d'), 921114589,
'Lisboa', 'Rua Lopes Andrade', 7, null, null, 2468, str_to_date('1991.04.30','%Y.%m.%d'));
CALL sp_registar_colaborador ('Nocas', str_to_date('2015.12.15','%Y.%m.%d'), 921352558,
'OLE', 'Rua De Deus', 1, null, null, 1458, str_to_date('1993.03.09','%Y.%m.%d'));
CALL sp_registar_colaborador ('Tildas', str_to_date('2014.11.15','%Y.%m.%d'), 921114589,
'Seixal', 'Margem Sul', 2, null, null, 2348, str_to_date('1995.05.31','%Y.%m.%d'));
CALL sp_registar_colaborador ('Trapla', str_to_date('2012.02.27','%Y.%m.%d'), 943252239,
'Setubal', 'Rua Baixa', 1, null, null, 6666, str_to_date('1991.10.10','%Y.%m.%d'));
CALL sp_registar_colaborador ('Pinhal', str_to_date('2014.02.19','%Y.%m.%d'), 918201489,
'Montijo', 'Rua Algures', 7, null, null, 4348, str_to_date('1992.02.19','%Y.%m.%d'));
CALL sp_registar_colaborador ('Priscila', str_to_date('2014.11.15','%Y.%m.%d'), 921114589,
'Pinhal Novo', 'Rua Soma', 6, null, null, 2425, str_to_date('1993.11.12','%Y.%m.%d'));
CALL sp_registar_colaborador ('Raminho', str_to_date('2014.11.15','%Y.%m.%d'), 921114589,
'Algures', 'Rua Vida', 3, null, null, 5555, str_to_date('1994.08.24','%Y.%m.%d'));

/*Inserir dados na tabela Fornecedor*/
CALL sp_registar_fornecedor ('Luis', str_to_date('2010.05.18','%Y.%m.%d'), 916458237,
'Setubal', 'CombusAll', 'Rua dos combustiveis', 10, null, null, 3150);
CALL sp_registar_fornecedor ('Pedro', str_to_date('2012.12.04','%Y.%m.%d'), 935761592,
'Almada', 'EnergiasVivas', 'Rua Sofle', 2, null, null, 4500);
CALL sp_registar_fornecedor ('Torin', str_to_date('2011.07.30','%Y.%m.%d'), 916732851,
'Setubal', 'LigaLeve', 'Rua Joaquem Almeida', 5,'2º','C', 5500);
CALL sp_registar_fornecedor ('Holha', str_to_date('2015.05.08','%Y.%m.%d'), 963482516,
'Almada', 'SabesCombustivel', 'Avenida Mal Me Quer', 6, '3º', 'D', 3690);
CALL sp_registar_fornecedor ('Hefe', str_to_date('2015.08.06','%Y.%m.%d'), 964829537,
'Montijo', 'MundoVerde', 'Avenida Qualquer', 8, null, null, 2350);
CALL sp_registar_fornecedor ('Ze', str_to_date('2010.11.24','%Y.%m.%d'), 924587112, 'Amora'
, 'DieselLife', 'Rua Zeca Afonso', 8, null, null, 2451);
CALL sp_registar_fornecedor ('Diogo', str_to_date('2009.03.22','%Y.%m.%d'), 914567852,
'Sesimbra', 'TreeHug', 'Rua Mane', 1, null, null, 2548);
CALL sp_registar_fornecedor ('Yah', str_to_date('2008.04.21','%Y.%m.%d'), 97453735,
'Setubal', 'TreeMongered', 'Rua Kebab', 10, null, null, 0952);
CALL sp_registar_fornecedor ('Han Solo', str_to_date('2007.06.16','%Y.%m.%d'), 93423522,
'Lisboa', 'TomSavior', 'Rua vida', 2, null, null, 1092);
CALL sp_registar_fornecedor ('Lalanja Boy', str_to_date('2013.04.09','%Y.%m.%d'), 91242852,
'Algures', 'AlguresCA', 'Rua Perdida', 5, null, null, 9367);
CALL sp_registar_fornecedor ('Carto', str_to_date('2010.02.10','%Y.%m.%d'), 932527852,
'Cevilha', 'SpanCompany', 'La Rua Rua', 34, null, null, 1528);
CALL sp_registar_fornecedor ('Yur', str_to_date('2011.08.25','%Y.%m.%d'), 939478278,
'Pertinho', 'Kiirmon', 'Rua Francesa', 54, null, null, 2111);
CALL sp_registar_fornecedor ('Yomcha', str_to_date('2007.01.10','%Y.%m.%d'), 96323232,
'Cidade', 'Polis', 'Rua Polis', 14, null, null, 1248);
CALL sp_registar_fornecedor ('Goku', str_to_date('2010.05.10','%Y.%m.%d'), 9163473, 'Pinhal
Novo', 'World', 'Rua World', 19, null, null, 4923);
CALL sp_registar_fornecedor ('Freeza', str_to_date('2001.10.22','%Y.%m.%d'), 91434645,
'Nave Espacial', 'Freeza Company', 'Desconhecida', 0, null, null, 1001);
CALL sp_registar_fornecedor ('Satan', str_to_date('2004.06.16','%Y.%m.%d'), 92131852,
'Nameque', 'SatanEmprsa', 'Rua Nameque', 12, null, null, 1123);
```

```
CALL sp_registar_fornecedor ('Trolha', str_to_date('2014.01.21','%Y.%m.%d'), 914522852,
'Sesimbra', 'Pelota', 'Rua Pelota', 11, null, null, 1156);
CALL sp_registar_fornecedor ('Robulha', str_to_date('2012.10.10','%Y.%m.%d'), 95353852,
'Montijo', 'Arbore', 'Rua Arbore', 12, null, null, 2382);
CALL sp_registar_fornecedor ('Marques', str_to_date('2011.12.01','%Y.%m.%d'), 9525353,
'Setubal', 'Flore', 'Rua Flore', 13, null, null, 2313);
CALL sp_registar_fornecedor ('Fornecedor', str_to_date('2010.12.17','%Y.%m.%d'), 91213852,
'Alcacer', 'Penoi', 'Rua Penoi', 16, null, null, 1252);

/*Inserir dados na tabela Combustivel*/
CALL sp_registar_combustivel (1.24, 'Gasolina 95', 1, 400);
CALL sp_registar_combustivel (0.99, 'Gasoleo', 2, 90000);
CALL sp_registar_combustivel (1.20, 'Gasolina 95', 5, 90000);
CALL sp_registar_combustivel (1.40, 'Gasolina 98', 3, 90000);

/*Inserir dados na tabela Ofertas*/
CALL sp_registar_oferta ('Bicicleta 110', 5000, str_to_date('2016.01.05','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Bicicleta todo terreno', 'Material', 20, null);
CALL sp_registar_oferta ('Apito', 100, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Faz barulho', 'Material', 100, null);
CALL sp_registar_oferta ('Porta Chaves', 150, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Decoracao', 'Material', 100, null);
CALL sp_registar_oferta ('Peluche', 800, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Fofo', 'Material', 100, null);
CALL sp_registar_oferta ('DVD', 400, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Filme à escolha', 'Material', 100, null);
CALL sp_registar_oferta ('Livro', 400, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Livro à escolha', 'Material', 100, null);
CALL sp_registar_oferta ('Cafe', 50, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Café', 'Material', 10000, null);
CALL sp_registar_oferta ('Pequeno-Almoço', 200, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Pequeno Almoço', 'Material', 1000, null);
CALL sp_registar_oferta ('Gelado', 150, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Gelado à escolha', 'Material', 1000, null);
CALL sp_registar_oferta ('Cenas', 990, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Cenas', 'Material', 200, null);
CALL sp_registar_oferta ('Magic Sword', 99999, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Espada Magica!', 'Material', 1, null);
CALL sp_registar_oferta ('Carro', 99999999, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Megan Prateado', 'Material', 5, null);
CALL sp_registar_oferta ('Portal', 500, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Portal para local', 'Material', 2, null);
CALL sp_registar_oferta ('Jogo', 15000, str_to_date('2016.03.20','%Y.%m.%d'),
str_to_date('2016.05.20','%Y.%m.%d'), 'Jogo à escolha', 'Material', 1000, null);
CALL sp_registar_oferta ('5 Litros',  450, str_to_date('2016.01.01','%Y.%m.%d'),
str_to_date('2016.05.01','%Y.%m.%d'), 'Gasolina 95 5 Litros', 'Descontos', null, 5);
CALL sp_registar_oferta ('10 Litros', 900, str_to_date('2016.01.15','%Y.%m.%d'),
str_to_date('2016.06.01','%Y.%m.%d'), 'Gasolina 95 10 Litros', 'Descontos', null, 10);
```

```
CALL sp_registar_oferta ('10 Litros', 900, str_to_date('2016.01.15','%Y.%m.%d'),
str_to_date('2016.06.01','%Y.%m.%d'), 'Gasoleo 5 Litros', 'Descontos', null, 5);
CALL sp_registar_oferta ('10 Litros', 800, str_to_date('2016.02.01','%Y.%m.%d'),
str_to_date('2016.06.01','%Y.%m.%d'), 'Gasoleo 10 Litros', 'Descontos', null, 10);
CALL sp_registar_oferta ('10 Litros', 900, str_to_date('2016.01.15','%Y.%m.%d'),
str_to_date('2016.06.01','%Y.%m.%d'), 'Gasolina 98 10 Litros', 'Descontos', null, 10);
CALL sp_registar_oferta ('5 Litros', 450, str_to_date('2016.01.15','%Y.%m.%d'),
str_to_date('2016.06.01','%Y.%m.%d'), 'Gasolina 98 5 Litros', 'Descontos', null, 5);

/*Inserir dados na tabela Vendas*/
CALL sp_registar_venda (2, 15, null, 1);
CALL sp_registar_venda (2, 14, null, 2);
CALL sp_registar_venda (4, null, 10.0, 3);
CALL sp_registar_venda (3, 20, null, 1);
CALL sp_registar_venda (4, null, 10, 2);
CALL sp_registar_venda (3, null, 15, 10);
CALL sp_registar_venda (3, 8, null, 15);
CALL sp_registar_venda (4, 10, null, 13);
CALL sp_registar_venda (3, null, 50, 9);
CALL sp_registar_venda (2, null, 35, 19);
CALL sp_registar_venda (3, 19, null, 8);
CALL sp_registar_venda (4, 5, null, 7);
CALL sp_registar_venda (2, null, 17, 9);
CALL sp_registar_venda (3, 19, null, 17);
CALL sp_registar_venda (4, null, 29, 5);
CALL sp_registar_venda (3, 48, null, 18);
CALL sp_registar_venda (3, 26, null, 15);
CALL sp_registar_venda (4, null, 10, 11);
CALL sp_registar_venda (2, 18, null, 12);
CALL sp_registar_venda (4, null, 50, 17);
CALL sp_registar_venda (3, 15, null, 10);
CALL sp_registar_venda (2, 20, null, 9);
CALL sp_registar_venda (4, null, 20, 10);
CALL sp_registar_venda (4, 2, null, 15);
CALL sp_registar_venda (4, null, 40, 12);
CALL sp_registar_venda (2, null, 30, 17);
CALL sp_registar_venda (3, 8, null, 15);
CALL sp_registar_venda (3, 15, null, 19);
CALL sp_registar_venda (3, 18, null, 16);

/*Inserir dados na tabela Faturacao e na tabela ClienteVendas*/
CALL sp_registar_fatura (1, 1);
CALL sp_registar_fatura (2, 2);
CALL sp_registar_fatura (3, 3);
CALL sp_registar_fatura (4, 4);
CALL sp_registar_fatura (5, 5);
CALL sp_registar_fatura (6, 6);
CALL sp_registar_fatura (9, 10);
```

```
CALL sp_registar_fatura (10, 10);
CALL sp_registar_fatura (11, 2);
CALL sp_registar_fatura (12, 14);
CALL sp_registar_fatura (16, 4);
CALL sp_registar_fatura (18, 5);
CALL sp_registar_fatura (20, 16);
CALL sp_registar_fatura (22, 14);
CALL sp_registar_fatura (23, 17);
CALL sp_registar_fatura (25, 1);
CALL sp_registar_fatura (27, 4);
CALL sp_registar_fatura (28, 2);
CALL sp_registar_fatura (29, 19);
CALL sp_registar_fatura (26, 9);
```

```
/*Inserir dados na tabela Vendas Ofertas*/
CALL sp_relacionar_vendaOferta (10, 2);
CALL sp_relacionar_vendaOferta (2,3);
CALL sp_relacionar_vendaOferta (1, 2);
CALL sp_relacionar_vendaOferta (2, 2);
CALL sp_relacionar_vendaOferta (3, 7);
CALL sp_relacionar_vendaOferta (4, 7);
CALL sp_relacionar_vendaOferta (5, 7);
CALL sp_relacionar_vendaOferta (6, 7);
CALL sp_relacionar_vendaOferta (7, 4);
CALL sp_relacionar_vendaOferta (8, 4);
CALL sp_relacionar_vendaOferta (8, 13);
CALL sp_relacionar_vendaOferta (9, 7);
CALL sp_relacionar_vendaOferta (10, 9);
CALL sp_relacionar_vendaOferta (10, 7);
CALL sp_relacionar_vendaOferta (12, 7);
CALL sp_relacionar_vendaOferta (15, 10);
CALL sp_relacionar_vendaOferta (17, 14);
CALL sp_relacionar_vendaOferta (19, 16);
CALL sp_relacionar_vendaOferta (22, 13);
CALL sp_relacionar_vendaOferta (23, 7);
CALL sp_relacionar_vendaOferta (24, 10);
CALL sp_relacionar_vendaOferta (24, 7);
```

Anexo D - teste.sql

```
use pos;
/* Fase 1*/
SELECT * FROM AutoridadeTributariaLoja; /* Aparece vazio porque a faturas têm a data do mes de maio e suposto aparecer a info do mes anterior*/
SELECT * FROM AutoridadeTributariaCliente; /* Aparece vazio porque a faturas têm a data do mes de maio e suposto aparecer a info do mes anterior*/
SELECT * FROM VolumeDeVendas;
SELECT * FROM NumeroOfertasPorCliente;
SELECT * FROM OfertasMaisVendidasCategoria;
SELECT * FROM OfertaMaisProcurada;
SELECT * FROM CombustivelMaisVendido;
SELECT * FROM CombustivelMenosVendido;
SELECT * FROM NVendasColaborador;
SELECT * FROM ClienteComMaisCompras;
SELECT * FROM NVendasSemCliente;
SELECT * FROM NVendasComCliente;

/* Fase 2*/

/* VIEWS */

/*Ver tabela cliente*/
SELECT * FROM TabelaCliente;

/*Ver tabela Colaborador*/
SELECT * FROM TabelaColaborador;

/*Ver tabela Fornecedor*/
SELECT * FROM TabelaFornecedor;

/*Ver tabela Combustivel*/
SELECT * FROM TabelaCombustivel;

/*Ver tabela ofertas*/
SELECT * FROM TabelaOfertas;

/*Ver tabela Vendas*/
SELECT * FROM TabelaVendas;

/*Ver tabela faturacao*/
SELECT * FROM tabelafaturacao;

/*Ver tabela clienteVendas*/
SELECT * FROM TabelaClienteVenda;
```

```
/*Ver tabela vendasOfertas*/
SELECT * FROM TabelaVendaOferta;

/*Ver clientes e quantas faturas existem e nome deles*/
SELECT * FROM nr_fatura_cliente;

/* Ver quais as ofertas menos vendidas*/
SELECT * FROM OfertaMenosProcurada;

/*  STORED PROCEDURE */


/*Registar um Cliente*/
CALL sp_registar_cliente('Rui', str_to_date('2015.05.12','%Y.%m.%d'), 916243897, 'Setubal',
245, 'TerraCotaa', 45135798, str_to_date('1989.03.20', '%Y.%m.%d'), "Rua Jacinto Andrade", 7,
null, null, 2900);
SELECT * FROM TabelaCliente;

/*Registar um Colaborador*/
CALL sp_registar_colaborador('Kore', str_to_date('2015.03.14','%Y.%m.%d'), 91458752,
'Setubal', 'Rua Super Homem', 23, '5º', 'G', 2900,  str_to_date('1990.05.17','%Y.%m.%d'));
CALL sp_registar_colaborador('Kore', str_to_date('2015.03.14','%Y.%m.%d'), 91458752,
'Setubal', 'Rua Super Homem', 23, null, null, 2900,  str_to_date('1990.05.17','%Y.%m.%d'));
SELECT * FROM TabelaColaborador;

/*Registar um Fornecedor*/
CALL sp_registar_fornecedor('Luis', str_to_date('2010.05.18','%Y.%m.%d'), 916458237,
'Setubal', 'CombusAll', 'Rua dos combustiveis', 10, null, null, 3150);
SELECT * FROM TabelaFornecedor;

/*Registar um Combustivel*/
CALL sp_registar_combustivel(1.24, 'Gasolina 95', 1, 400);
CALL sp_registar_combustivel(1.24, 'Gasolina 95', 1, 400);
CALL sp_registar_combustivel(1.24, 'Gasolina 95', 1, 200);
SELECT * FROM TabelaCombustivel;

/*Registar uma Ofertas*/
CALL sp_registar_oferta('5 Litros', 300, curdate(), str_to_date('2016.05.01','%Y.%m.%d'),
'Gasolina 95 5 Litros', 'Descontos', null, 5);
SELECT * FROM TabelaOfertas;

/*Registar uma Fatura*/
CALL sp_registar_fatura (8, 5);
SELECT * FROM TabelaFaturacao;

/*Relacionar um cliente com uma venda*/
CALL sp_relacionar_clienteVenda(1,5);
SELECT * FROM ClienteVendas;
```

```
/*Relacionar uma venda com uma oferta*/
CALL sp_relacionar_vendaOferta(1, 2);
CALL sp_relacionar_vendaOferta(4, 7);
SELECT * FROM VendasOfertas;

/*Mostra os produtos associados a um id de uma fatura*/
CALL sp_produtos_fatura(2);

/*Registar uma venda*/
CALL sp_registar_venda(4, 10, null,  3);
CALL sp_registar_venda(2, null, 20, 2);
SELECT * FROM TabelaVendas;

/*Listar as categorias por ofertas*/
CALL sp_listar_ofertas_categoria('Material');

/*Detalhes de uma fatura a partir do seu id*/
CALL sp_resumo_fatura(2);
CALL sp_resumo_fatura(3);
CALL sp_resumo_fatura(8);

/* Volume anual de vendas divido por meses*/
CALL sp_volume_anual_mes(str_to_date('2016.06.30','%Y.%m.%d'));

/*Remover uma venda a partir do seu id*/
CALL sp_remover_venda(1);
CALL sp_remover_venda(2);
SELECT * FROM tabelavendas;
SELECT * FROM clientevendas;
SELECT * FROM tabelafaturacao;
SELECT * FROM vendasofertas;

/*Fechar uma fatura a partir do seu id*/
CALL sp_fechar_fatura(5);
CALL sp_fechar_fatura(6);

/*Abrir uma fatura a partir do seu id*/
CALL sp_abrir_fatura(5);
CALL sp_abrir_fatura(6);
SELECT * FROM tabelafaturacao;

/*Adicionar quantidade de um combustivel a uma fatura e por consequencia a uma venda,
recebe id da fatura e a quantidade de combustivel*/
CALL sp_adicionar_combustivel(3, 5);
SELECT * FROM tabelavendas;
```

```
/*Alterar o stock de um combustivel, dá-se o id e a quantidade. Só funciona se a data_fim =
null*/
CALL sp_alterar_stock_combustivel(2, 200);
SELECT * FROM tabelaCombustivel;

//*Remove uma quantidade de combustivel de uma fatura, recebe o id da fatura e a
quantidade a remover*/
CALL sp_remove_combustivel_fatura(3, 5);
CALL sp_resumo_fatura(3);

CALL sp_remove_combustivel_fatura(5, 2);
CALL sp_resumo_fatura(5);

/*Remover uma oferta, recebe o id da oferta*/
CALL sp_remover_oferta(20);
CALL sp_remover_oferta(19);
SELECT * FROM tabelaOfertas;

/*Alterar nome de uma oferta*/
CALL sp_alterar_oferta_nome(1, 'OLEEEE');
SELECT * FROM tabelaOfertas;

/*Alterar pontos de uma oferta*/
CALL sp_alterar_oferta_pontos(1, 500);
SELECT * FROM tabelaOfertas;

/*Alterar data inicio de uma oferta*/
CALL sp_alterar_oferta_data_inicio(1, str_to_date('2016.05.19', '%Y.%m.%d'));
SELECT * FROM tabelaOfertas;

/*Alterar data fim de uma oferta*/
CALL sp_alterar_oferta_data_fim(1, str_to_date('2016.05.22', '%Y.%m.%d'));
SELECT * FROM tabelaOfertas;

/*Alterar desricao de uma oferta*/
CALL sp_alterar_oferta_descricao(1, 'Cenas do Mexixo');
SELECT * FROM tabelaOfertas;

/*Alterar categoria de uma oferta*/
CALL sp_alterar_oferta_categoria(1, 'Material');
SELECT * FROM tabelaOfertas;

/*Alterar o numero de ofertas de uma oferta*/
CALL sp_alterar_oferta_numeroOfertas(1, 50);
SELECT * FROM tabelaOfertas;

/*Alterar o numero de litros de uma oferta*/
CALL sp_alterar_oferta_litros(15, 10);
```

```
SELECT * FROM tabelaOfertas;

/*Remove uma fatura, recebe o id da fatura*/
CALL sp_remover_fatura(4);
SELECT * FROM tabelafaturacao;
SELECT * FROM tabelaVendas;
SELECT * FROM clienteVendas;

/*Remove uma oferta de uma fatura, recebe o id e o nome da oferta a remover*/
CALL sp_remover_oferta_fatura(8, 'Café');
CALL sp_produtos_fatura(8);

/*Muda o estado de um cliente para inativo*/
CALL sp_desativar_cliente(1);
SELECT * FROM tabelaCliente;

/*Muda o estado de um cliente para ativo*/
CALL sp_ativar_cliente(1);
SELECT * FROM tabelaCliente;

/*Muda o estado de um colaborador para inativo*/
CALL sp_desativar_colaborador(2);
SELECT * FROM tabelacolaborador;

/*Muda o estado de um colaborador para ativo*/
CALL sp_ativar_colaborador(2);
SELECT * FROM tabelacolaborador;

/*Muda o estado de um fornecedor para inativo*/
CALL sp_desativar_fornecedor(1);
SELECT * FROM tabelafornecedor;

/*Muda o estado de um fornecedor para inativo*/
CALL sp_ativar_fornecedor(1);
SELECT * FROM tabelaFornecedor;

/*Alterar o combustivel de uma venda*/
CALL sp_alterar_venda_combustivel(1, 4);
SELECT * FROM tabelaVEndas;

/*Alterar o colaborador de uma venda*/
CALL sp_alterar_venda_colaborador(1, 19);
SELECT * FROM tabelaVendas;

/*Alterar litros de uma venda*/
CALL sp_alterar_venda_litros(1, 19);
SELECT * FROM tabelaVendas;
SELECT * FROM tabelaFaturacao;
```

```
/*Alterar custo de uma venda*/
CALL sp_alterar_venda_custo(1, 40);
SELECT * FROM tabelaVEndas;
SELECT * FROM tabelaFaturacao;


/**********ALTERAR CLIENTE***********/

/*Alterar o nome de um cliente*/
CALL sp_alterar_cliente_nome(1, 'Olee');
SELECT * FROM tabelaCliente;

/*Alterar o telefone de um cliente*/
CALL sp_alterar_cliente_telefone(1, 1111111);
SELECT * FROM tabelaCliente;

/*Alterar localidade de um cliente*/
CALL sp_alterar_cliente_localidade(1, 'VIDA');
SELECT * FROM tabelaCliente;

/*Alterar empresa de um cliente*/
CALL sp_alterar_cliente_empresa(1, 'Chato');
SELECT * FROM tabelaCliente;

/*Alterar nif de um cliente*/
CALL sp_alterar_cliente_nif(1, 222222);
SELECT * FROM tabelacliente;


/*Visualizar o numero de faturas de um cliente, a partir do seu id e de uma data*/
CALL sp_fatura_cliente_data(str_to_date('2016.06.02', '%Y.%m.%d'), 1);
CALL sp_registar_fatura(28, 1);
CALL sp_fatura_cliente_data(str_to_date('2016.06.02', '%Y.%m.%d'), 1);


/********Alterar Colaborador ***********/

/*Alterar o nome de um colaborador*/
CALL sp_alterar_colaborador_nome(1, 'OLEE');
SELECT * FROM tabelaColaborador;

/*Alterar o telefone de um colaborador*/
CALL sp_alterar_colaborador_telefone(1, 111111);
SELECT * FROM tabelacolaborador;

/*Altera a localidade de um colaborador*/
CALL sp_alterar_colaborador_localidade(1, 'CHATO');
```

```
SELECT * FROM tabelacolaborador;


/* Testar Trigger */

CALL sp_registar_fatura (24, 1);
SELECT * FROM TabelaCliente;
SELECT * FROM TabelaVendas;
SELECT * FROM tabelafaturacao;
SELECT * FROM ClienteVendas;

/**********************************/
```