

Relatório trabalho final base de dados

Base de dados da bijuteria da Dona Elvira



Fonte: [Criador de Imagens \(bing.com\)](#)

CTeSP de Informática – Base de Dados

Professor da Disciplina – José Paulo da Costa Rêgo

Trabalho feito por Daniel Pereira

Índice

Introdução.....	6
Diagrama da base de dados	6
Construção da Base de dados.....	6
CREATE DATABASE	7
CREATE TABLE	7
TAB_pessoa.....	7
TAB_morada	7
TAB_morada_preferencial	8
TAB_email	8
TAB_email_preferencial	9
TAB_telemovel.....	9
TAB_telemovel_preferencial.....	9
TAB_password	10
TAB_reset_password.....	10
TAB_informacoes_cidadao.....	11
TAB_passaporte.....	11
TAB_cliente	12
TAB_profissao.....	12
TAB_funcionario	12
TAB_experiencia	13
TAB_unidades_tempo	13
TAB_relacoes_unidade_tempo	14
TAB_contrato	14
TAB_cargos.....	15
TAB_hierarquia	15
TAB_promocoes_cargos.....	15
TAB_tipo_instalacoes.....	16
TAB_instalacoes	16
TAB_tipos_manutencao	17
TAB_manutencao	17
TAB_feriado	18
TAB_dia_feriado	18
TAB_horario	18
TAB_horas	19
TAB_dia_semana	19

TAB_horas_semana	19
TAB_promocao	20
TAB_validade_promocao.....	20
TAB_promocao_idade_necessaria.....	21
TAB_promocao_desconto_monetario_geral.....	21
TAB_metodo_pagamento	21
TAB_promocao_metodo.....	21
TAB_promocao_cliente	22
TAB_tipos_artigos	22
TAB_fornecedores.....	22
TAB_artigos.....	23
TAB_promocao_tipo_artigo	23
TAB_promocao_artigo.....	24
TAB_tamanho	24
TAB_tamanho_artigo.....	24
TAB_stock_artigo	25
TAB_percentagem_lucro_artigo	25
TAB_transferencias	25
TAB_artigo_para_transferencia	26
TAB_venda.....	26
TAB_troca	27
EVENTS E TRIGGERS	27
verificar_aniversariantes	28
CREATE TABLE IF NOT EXISTS logs_clientes	28
TRIGGERS da tabela TAB_cliente	29
TRIGGERS da tabela TAB_pessoa	30
TRIGGERS para artigo em stock	31
Adição de regtos	31
INSERT INTO TAB_profissao.....	31
INSERT INTO TAB_tipo_instalacoes.....	32
INSERT INTO TAB_tipos_manutencao	32
INSERT INTO TAB_unidades_tempo	32
INSERT INTO TAB_relacoes_entre_unidades_tempo.....	33
INSERT INTO TAB_promocao.....	33
INSERT INTO TAB_tipos_artigos.....	33
INSERT INTO TAB_promocao_tipo_artigo	33

INSERT INTO TAB_metodo_pagamento.....	34
INSERT INTO TAB_feriado.....	34
INSERT INTO TAB_dia_feriado	35
INSERT INTO TAB_instalacoes.....	35
INSERT INTO TAB_horas.....	36
INSERT INTO TAB_dia_semana	36
INSERT INTO TAB_horario	36
INSERT INTO TAB_horas_semana	36
INSERT INTO TAB_pessoa	37
INSERT INTO TAB_funcionario	37
INSERT INTO TAB_cargos	37
INSERT INTO TAB_hierarquia	38
INSERT INTO TAB_promocoes_cargos	38
INSERT INTO TAB_experiencia.....	38
INSERT INTO TAB_tamanho.....	39
INSERT INTO TAB_fornecedores	39
INSERT INTO TAB_artigos	40
INSERT INTO TAB_tamanho_artigo	40
INSERT INTO TAB_percentagem_lucro_artigo	41
INSERT INTO TAB_stock_artigo.....	41
INSERT INTO TAB_venda	41
INSERT INTO TAB_transferencias	41
INSERT INTO TAB_artigo_para_transferencia	41
Tabelas Auxiliares.....	42
TAB_AUX_pessoa	42
TAB_AUX_cargos_profissoes	42
Adicionar dados as tabelas auxiliares	42
INSERT INTO TAB_AUX_pessoa.....	42
INSERT INTO TAB_AUX_cargos_profissoes	43
Functions e Procedures.....	44
FUNCTION obter_nome_aleatorio	44
FUNCTION obter_sobrenome_aleatorio.....	45
FUNCTION gerar_nif.....	46
FUNCTION gerar_cc.....	47
FUNCTION gerar_passaporte	48
PROCEDURE ExecutarInsercaoRegistrosMultiplosPessoas	49

CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplosClientes	50
CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplosFuncionarios	50
PROCEDURE ApagarRegistrosRepetidosClientes.....	52
FUNCTION proxima_data	53
FUNCTION dia_feriado_muda_cada_ano.....	54
PROCEDURE ExecutarInsercaoRegistrosMultiplosCidados	54
PROCEDURE ExecutarInsercaoRegistrosMultiplosEstrangeiros	55
PROCEDURE ExecutarInsercaoRegistrosMultiplosContratosAtuaisFuncionarios	56
FUNCTION obter_contratoMaisRecente	57
FUNCTION obter_definicao_hierarquicaMaisRecente	58
FUNCTION obter_promocaoMaisRecente	58
FUNCTION obter_data_termino_contrato	58
PROCEDURE ExecutarInsercaoRegistrosMultiplasPromocoes.....	59
FUNCTION obter_experienciaMaisRecente	59
PROCEDURE ExecutarInsercaoRegistrosMultiplasAbastecimentosStock	60
FUNCTION obter_artigos_em_stock	60
FUNCTION obter_valor_artigos_compra_media	61
FUNCTION obter_percentagem_lucro_artigo_atual.....	62
FUNCTION obter_data_horarioMaisRecente	63
VIEWS.....	63
CREATE VIEW VIEW_informacoes_cliente.....	64
CREATE VIEW VIEW_hierarquia_ordenada_cargos_atual	65
CREATE VIEW VIEW_informacoes_funcionario.....	65
CREATE VIEW VIEW_detalhes_artigos.....	66
CREATE VIEW VIEW_horario_atual.....	67
CREATE VIEW VIEW_informacao_instalacoes	68
Execução de procedimentos.....	68
SELECT	69
Nomes aleatórios	69
Aniversariantes.....	69
Acontecimentos na tabela TAB_cliente	71
Contagem de acontecimentos na tabela TAB_cliente	72
Feriados e datas	73
Profissões que tem salário base perto da média	73
Detalhes funcionários.....	74
Ficheiro que faz tudo!.....	74

Conclusão.....	75
Notas adicionais	75



Fonte : [Criador de Imagens \(bing.com\)](#)

Introdução

Este trabalho visa mostrar as minhas capacidades para criar uma base de dados com o objetivo de armazenar os dados e, de certa forma, também gerir uma bijuteria. Para tal para não fugir do costume decidi imaginar todos os desafios que a Dona Elvira poderia enfrentar e de tal forma preveni-los. Para tal fiz um total de 51 tabelas fundamentais (ou seja, sem contar com auxiliares e logs).

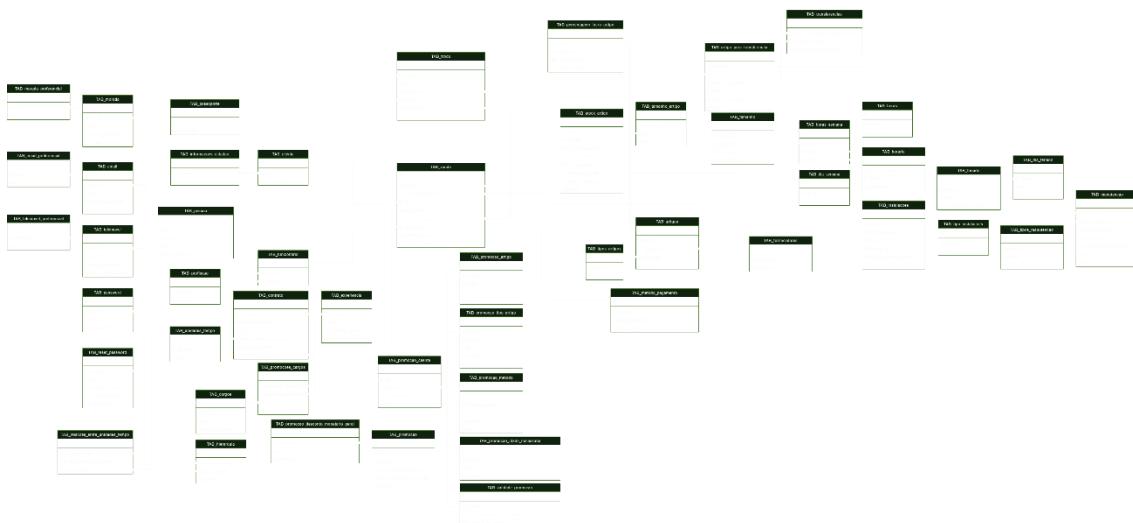
Devido à complexidade deste projeto, achei melhor ir guardando o meu progresso no GitHub, de modo a não perder a minha linha de raciocínio e obter o resultado idealizado.

github.com/Dani12600000/Trabalho_final_base_dados_Bijuteria_Dona_Elvira

Enfim, sem mais nada a dizer, comecei o projeto da seguinte forma:

Diagrama da base de dados

Primeiramente comecei por pensar minuciosamente no que a Dona Elvira poderia querer armazenar e qual a melhor forma para tal. Daí o seguinte diagrama.



Se não conseguir ver bem aqui não se preocupe, enviarei em anexo a mesma tal como pode encontrá-la nos ficheiros do meu repositório do Git.

Construção da Base de dados

Depois de ter pensado muito bem em todas as possibilidades e implementar essas ideias no diagrama passei para construção propriamente dita no MySQL Workbench

CREATE DATABASE

Começando com o comando CREATE DATABASE para criar a base de dados e logo de seguida com o comando USE começar a trabalhar nela.

CREATE TABLE

Depois da base de dados criada e pronta a ser trabalhada, adicionei as tabelas mostradas anteriormente no diagrama com o comando CREATE TABLE. Para não ficar repetitivo e trabalhoso, vou apenas explicar as que vi necessidade para tal. Caso não explique de certo será fácil entender.

TAB_pessoa

Começando com a minha tabela favorita em todas as bases de dados que envolvem pessoas, a tabela TAB_pessoa.

Pode se questionar a que pessoas me refiro, aí está, todas! Quer seja funcionário ou cliente fica registado na tabela pessoa. Porquê? Para conveniência para não haver uma redundância de dados ou chance de a pessoa se enganar. Imagine que existe o cliente Tó Zé, e que ele um dia mais tarde é contratado na bijuteria da dona Elvira. Para não ter de introduzir o Tó Zé 2 vezes apenas insere uma vez quando se regista como cliente e quando se for registar como funcionário da bijuteria basta encontrar o seu registo de pessoa. Assim a probabilidade de alguma informação vital estar errada, como aniversario, NIF, morada, será menor. Para tal fiz a seguinte tabela

```
CREATE TABLE TAB_pessoa (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome NVARCHAR(255) NOT NULL,
    sobrenome NVARCHAR (255) NOT NULL,
    NIF VARCHAR (25) NOT NULL,
    data_nascimento DATE NOT NULL DEFAULT (DATE_SUB(CURRENT_DATE(), INTERVAL 18 YEAR))
);
```

TAB_morada

Com a tabela pessoa criada faltou inserir a informação sobre a morada. Esta morada será usada para enviar faturas de compras online, e também qual a instalação do tipo venda mais próxima da morada da pessoa para definir como instalação de recolha de encomendas (clientes) ou envio de fatura de salário (funcionários). Para tal, a seguinte tabela

```
CREATE TABLE TAB_morada (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_pessoa INT NOT NULL,
    morada NVARCHAR(255) NOT NULL,
    data_hora_adicionado DATETIME NOT NULL DEFAULT (NOW()),
    data_hora_removido DATETIME,
    FOREIGN KEY (ID_pessoa) REFERENCES TAB_pessoa(ID)
);
```

TAB_morada_preferencial

Como podem ser várias moradas, decidi fazer uma tabela para guardar a morada preferencial de cada pessoa, assim, pode selecionar essa morada preferencial como morada para realizar os cálculos de distâncias.

```
CREATE TABLE TAB_morada_preferencial (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_morada INT NOT NULL,
    desde DATETIME NOT NULL DEFAULT (NOW()),
    FOREIGN KEY (ID_morada) REFERENCES TAB_morada(ID)
);
```

TAB_email

Com as moradas associadas à pessoa fiz uma tabela para armazenar os emails das pessoas, assim esse endereço de email pode ser usado na página de login e também para enviar emails para reposição de password ou de recebimento de promoções.

```
CREATE TABLE TAB_email (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_pessoa INT NOT NULL,
    email VARCHAR(150) NOT NULL,
    data_hora_associacao DATETIME NOT NULL DEFAULT (NOW()),
    data_hora_desassociacao DATETIME,
    FOREIGN KEY (ID_pessoa) REFERENCES TAB_pessoa(ID)
);
```

TAB_email_preferencial

Tal como na morada, também fiz um preferencial para os emails. Assim os emails serão enviados exclusivamente para o preferencial enquanto que todos os outros serão apenas usados para permitir o login nas plataformas digitais.

```
CREATE TABLE TAB_email_preferencial (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_email INT NOT NULL,
    desde DATETIME NOT NULL DEFAULT (NOW()),
    FOREIGN KEY (ID_email) REFERENCES TAB_email(ID)
);
```

TAB_telemovel

Depois fiz o telemóvel pois as vezes, no caso do cliente, é bom ter o número de telefone da pessoa para se poder ligar a avisar sobre promoções ou outras notificações que possam atrair a pessoa. No caso do funcionário é bom ter um número no caso de alguma emergência. Outra coisa que o número de telemóvel pode ser usado é para fazer login nas plataformas da bijuteria, ou seja se for feita uma app mobile pode ser utilizado automaticamente o número do telemóvel em que está a ser feito o login para autenticar a pessoa tal como adicionar o numero da mesma, caso o login seja feito num novo dispositivo.

```
CREATE TABLE TAB_telemovel (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_pessoa INT NOT NULL,
    telemovel VARCHAR(25) NOT NULL,
    data_hora_associacao DATETIME NOT NULL DEFAULT (NOW()),
    data_hora_desassociacao DATETIME,
    FOREIGN KEY (ID_pessoa) REFERENCES TAB_pessoa(ID)
);
```

TAB_telemovel_preferencial

Tal como nas outras tabelas fiz também um preferencial para que apenas se ligue para o numero preferencial e no caso dos funcionários, se não for atendido no preferencial, sim, ligar para os outros números

```

CREATE TABLE TAB_telemovel_preferencial (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_telemovel INT NOT NULL,
    desde DATETIME NOT NULL DEFAULT (NOW()),
    FOREIGN KEY (ID_telemovel) REFERENCES TAB_telemovel(ID)
);

```

TAB_password

Por fim, com informações como email e número de telemóvel faltou apenas a password para se poder fazer logins na plataforma. Eu, nesta abordagem, decidi fazer que o login de cliente ser o mesmo que o de funcionário. O que o distinguirá será apenas a app no qual será feito o login (APP cliente ou a APP funcionário)

```

CREATE TABLE TAB_password (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_pessoa INT NOT NULL,
    password NVARCHAR(255) NOT NULL,
    desde_data_hora DATETIME NOT NULL DEFAULT (NOW()),
    FOREIGN KEY (ID_pessoa) REFERENCES TAB_pessoa(ID)
);

```

TAB_reset_password

Como as pessoas têm uma memória biológica e às vezes se esquecem de coisas importantes como passwords fiz uma tabela para recetar as passwords.

ATENÇÃO – Isto apenas armazena os pedidos para tal, para ser usado propriamente tem que ser feito uma aplicação sobre isto para gerir toda a parte das passwords

Mas a ideia é:

- Assim que na página de reset é introduzido um dos mails ou n_telemovel, com isso ele manda um e-mail/mensagem para o mesmo introduzido com o código aleatório gerado
- Com esse código voltar na página mesma página anteriormente e introduzir o código recebido
- Com isso a pessoa é redirecionada para a página de alteração de password e assim que clica no botão de alterar esse registo fica com o campo usado = TRUE e o data_hora_usado = [A data e hora que recetou a password]

```
CREATE TABLE TAB_reset_password (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_pessoa INT NOT NULL,
    codigo VARCHAR(12) NOT NULL,
    usado BOOL NOT NULL DEFAULT (0),
    data_hora_requesitado DATE NOT NULL DEFAULT (NOW()),
    data_hora_usado DATE,
    FOREIGN KEY (ID_pessoa) REFERENCES TAB_pessoa(ID)
);
```

TAB_informacoes_cidadao

Com contas feitas fiz 2 tabelas para descrever a pessoa (cliente ou funcionário) melhor

Para isso comecei com a tabela que define as pessoas como cidadãos portugueses da forma que se observa abaixo

```
CREATE TABLE TAB_informacoes_cidadao (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_pessoa INT NOT NULL,
    CC VARCHAR(30) NOT NULL,
    FOREIGN KEY (ID_pessoa) REFERENCES TAB_pessoa(ID)
);
```

TAB_passaporte

Depois de ter os cidadãos fiz a tabela para definir os estrangeiros da forma que se observa abaixo

```
CREATE TABLE TAB_passaporte (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_pessoa INT NOT NULL,
    ID_passaporte VARCHAR(100),
    FOREIGN KEY (ID_pessoa) REFERENCES TAB_pessoa(ID)
);
```

TAB_cliente

Finalmente fiz as 2 tabelas para definir as pessoas como referido na tabela pessoa. Se são clientes ou funcionários.

Para tal comecei com a tabela cliente que se pode observar abaixo

```
CREATE TABLE TAB_cliente (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_pessoa INT NOT NULL,
    data_hora_registro DATETIME NOT NULL DEFAULT (NOW()),
    FOREIGN KEY (ID_pessoa) REFERENCES TAB_pessoa(ID)
);
```

TAB_profissao

Antes de fazer a tabela para os funcionários fiz a tabela da profissão que irá descrever qual a profissão do funcionário tal como qual o salário base de cada profissão que é usado para calculo dos salários (coisa que será explicada nas funções ou views)

```
CREATE TABLE TAB_profissao (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    designacao NVARCHAR(200),
    salario_base DECIMAL(10,2) NOT NULL DEFAULT (740.83)
);
```

TAB_funcionario

Com profissões feitas fiz a tabela que irá dizer que pessoas são funcionarias da bijuteria e quais são as suas profissões

```
CREATE TABLE TAB_funcionario (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_pessoa INT NOT NULL,
    ID_profissao INT NOT NULL,
    FOREIGN KEY (ID_pessoa) REFERENCES TAB_pessoa(ID),
    FOREIGN KEY (ID_profissao) REFERENCES TAB_profissao(ID)
);
```

TAB_experiencia

Como funcionários são mais ou menos remunerados a sua experiência por conseguirem fazer coisas melhores e mais rápido pensei em fazer a tabela TAB_experiencia que serve para isso mesmo.

Na tabela indica-se qual o funcionário que estamos a dar a experiência tal como os anos de experiência que ele tem e salário extra que receberá por tal. A parte de o salário extra ser definido aqui é pelo facto que nem todos recebem o mesmo extra pelos mesmos anos de experiência.

Para tal, fiz o seguinte

```
CREATE TABLE TAB_experiencia (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_funcionario INT NOT NULL,
    n_anos INT NOT NULL DEFAULT (1),
    salario_extra_experiencia DECIMAL(10,2) NOT NULL,
    data_hora DATETIME NOT NULL DEFAULT (NOW()),
    FOREIGN KEY (ID_funcionario) REFERENCES TAB_funcionario(ID)
);
```

TAB_unidades_tempo

Com isso fiz uma tabela com unidades de tempo, ou seja, segundos, minutos, horas, dias, semanas, meses e anos

Para tal nesta tabela defino o nome da unidade de tempo no singular e no plurar para poder ser usado nos selects para mostrar de forma mais bonito (1 dia, 2 horas, 10 minutos e 1 segundo – algo assim)

Para tal fiz o seguinte

```
CREATE TABLE TAB_unidades_tempo (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    no_singular NVARCHAR(10) NOT NULL,
    no_plural NVARCHAR(15) NOT NULL
);
```

TAB_relacoes_unidade_tempo

Com as unidades de tempo fiz as relações entre as unidades de tempo, ou seja, 60 minutos são 1 hora, 24 horas são 1 dia, 7 dias são 1 semana, 4.34812141 semanas são 1 mês e 12 meses são 1 ano

Para tal no ID_unidade_tempo colocamos a unidade tempo que referimos (por exemplo 3 – hora) e no ID_proxima_unidade_tempo a unidade de tempo a seguir (no caso 4 – dia) e no n_unidades_tempo_para_proxima (ou seja 24 – que quer dizer 24 horas)

```
CREATE TABLE TAB_relacoes_entre_unidades_tempo (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_unidade_tempo INT NOT NULL,
    n_unidades_tempo_para_proxima FLOAT NOT NULL,
    ID_proxima_unidade_tempo INT NOT NULL,
    FOREIGN KEY (ID_unidade_tempo) REFERENCES TAB_unidades_tempo(ID),
    FOREIGN KEY (ID_proxima_unidade_tempo) REFERENCES TAB_unidades_tempo(ID)
);
```

TAB_contrato

As 2 tabelas anteriores foram feitas para esta, a tabela dos contratos dos funcionários. Como no mundo do negócio tudo funciona com contratos fiz a tabela para guardar os mesmos da seguinte forma

```
CREATE TABLE TAB_contrato (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_funcionario INT NOT NULL,
    data_hora_contratado DATETIME,
    prazo_contrato INT,
    ID_unidade_tempo_prazo_contrato INT,
    data_hora_cancelado DATETIME,
    ID_funcionario_cancelou INT,
    FOREIGN KEY (ID_funcionario) REFERENCES TAB_funcionario(ID),
    FOREIGN KEY (ID_funcionario_cancelou) REFERENCES TAB_funcionario(ID),
    FOREIGN KEY (ID_unidade_tempo_prazo_contrato) REFERENCES TAB_unidades_tempo(ID)
);
```

- data_hora_contrato – data e hora que começou o contrato
- ID_unidade_tempo_prazo_contrato – usa os registos da tabela unidades tempo para definir o tempo do contrato, ou seja, se é dias, meses ou anos (não acredito que alguém colocaria menos)
- prazo_contrato – é usado em conjunto com o ID_unidade_tempo_... para dizer se se são 1, 2, 3,... anos/meses/dias
- data_hora_cancelado e ID_funcionario_cancelou – serve para na eventualidade de o contrato ser cancelado antes do término do contrato (ou seja, despedido) esses dois campos são preenchidos. A data_hora com a data e hora que ele teve o contrato rescindido e o ID_funcionario_cancelou o ID do funcionário que o

despediu (obviamente só pode ser algum superior, que é me dito através de umas tabelas feitas mais a frente)

TAB_cargos

Cada funcionário tem a sua profissão e cargo. Como profissões e cargos não tem a ver não fiz nenhuma ligação entre eles (apenas nas tabelas auxiliares para não ter registos MUITO estranhos, tal como limitando certo cargo a uma pessoa específica. Será falado na parte dedicada as tabelas auxiliares). Mas, como o mesmo cargo em diferentes pessoas tem o mesmo salário extra eu coloquei aqui para depois ser adicionado ao salário do funcionário com o determinado cargo.

```
CREATE TABLE TAB_cargos (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    designacao NVARCHAR(255) NOT NULL,
    funcoes TEXT NOT NULL,
    salario_extra_cargo DECIMAL(10,2)
);
```

TAB_hierarquia

Entre a criação das tabelas dos cargos, e como falado anteriormente na tabela dos contratos, me surgiu a ideia de fazer uma hierarquia, ou seja, ter alguém um CEO, diretores, supervisores e por fins trabalhadores regulares. Assim com isso o CEO poder despedir quem quiser, pois é CEO, os diretores todos os subordinados a eles, tal como os supervisores.

```
CREATE TABLE TAB_hierarquia (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_cargo_atribuindo INT NOT NULL,
    ID_cargo_superior INT, -- Se vazio quer dizer que não tem superior, ou seja esse cargo é o mais alto de todos
    data_hora DATETIME NOT NULL DEFAULT (NOW()),
    FOREIGN KEY (ID_cargo_superior) REFERENCES TAB_cargos(ID),
    FOREIGN KEY (ID_cargo_atribuindo) REFERENCES TAB_cargos(ID)
);
```

TAB_promocoes_cargos

Voltando aos cargos fiz a tabela que faz a promoção dos funcionários aos cargos.

Normalmente o ID_funcionario_promovedor não pode ser vazio pois esse campo indica quem o promoveu a esse cargo (ou seja, um supervisor pode promover um funcionário comum a supervisor, um diretor pode promover um funcionário comum a supervisor ou diretor e o CEO pode fazer isso tudo tal como promover ao seu cargo, ou seja, ser novo

CEO) mas podendo ser vazio pois o primeiro cargo de todos a ser promovido (CEO) não há ninguém que o promova a esse cargo se não o próprio

```
CREATE TABLE TAB_promocoes_cargos (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_funcionario_promovido INT NOT NULL,
    ID_funcionario_promovedor INT,
    ID_cargo INT NOT NULL,
    data_promovido DATE NOT NULL DEFAULT (CURRENT_DATE())
);
```

TAB_tipo_instalacoes

Com parte de RH acabada passei para o processamento das instalações com o começo de identificar os tipos de instalações com a seguinte tabela

```
CREATE TABLE TAB_tipo_instalacoes (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    designacao NVARCHAR(255) NOT NULL,
    descricao TEXT NOT NULL
);
```

TAB_instalacoes

Depois com os tipos de instalações fiz a tabela para dizer as instalações existentes da seguinte forma

```
CREATE TABLE TAB_instalacoes (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    designacao NVARCHAR (255) NOT NULL,
    ID_tipo_instalacoes INT NOT NULL,
    morada NVARCHAR(255) NOT NULL,
    localizacao_gps POINT NOT NULL,
    data_hora_primeira_abertura DATETIME NOT NULL DEFAULT (NOW()),
    data_hora_ultimo_encerramento DATETIME,
    FOREIGN KEY (ID_tipo_instalacoes) REFERENCES TAB_tipo_instalacoes(ID)
);
```

- data_hora_primeira_abertura – é a data e hora que a instalação abriu pela primeira vez ao público

- data_hora_ultimo_encerramento – é a data e hora que a instalação fechou pela ultima vez, ou seja, foi encerrada completamente ao publico

TAB_tipos_manutencao

Como coisas físicas se estragam, vão degradando, sujando, é preciso fazer manutenções

Para tal fiz a tabela TAB_tipos_manutencao que diz os tipos de manutenções que podem existir tal como se normalmente afeta o normal funcionamento da instalação que sofre da manutenção. Por exemplo, uma limpeza não afeta o normal funcionamento por isso pode continuar aberta, agora no caso de uma restauração (ou reparação a fundo) isso certamente afeta e como tal é necessário ficar fechada durante a execução da mesma.

```
CREATE TABLE TAB_tipos_manutencao (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    designacao NVARCHAR(255) NOT NULL,
    descricao TEXT NOT NULL,
    normalmente_afeta_normal_func BOOL NOT NULL DEFAULT (1)
);
```

TAB_manutencao

Com os tipos de manutenção fiz uma para registrar todas as manutenções realizadas tal como seus detalhes (data hora que começou e terminou, se afetou ou vai afetar o normal funcionamento, pode vir da tabela anterior, mas pode ser diferente dependendo da situação, e por fim o valor que a manutenção custou e os detalhes do que aconteceu)

```
CREATE TABLE TAB_manutencao (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_instalacoes INT NOT NULL,
    ID_tipo_manutencao INT NOT NULL,
    data_hora_comeco DATETIME NOT NULL DEFAULT (NOW()),
    data_hora_termino DATETIME,
    afeta_funcionamento_normal BOOL NOT NULL,
    valor DECIMAL(10,2) NOT NULL,
    detalhes TEXT NOT NULL,
    FOREIGN KEY (ID_instalacoes) REFERENCES TAB_instalacoes(ID),
    FOREIGN KEY (ID_tipo_manutencao) REFERENCES TAB_tipos_manutencao(ID)
);
```

TAB_feriado

Com instalações, RH e clientes faltou uma coisa para manter os funcionários felizes, os feriados.

Para tal fiz a tabela TAB_feriado que regista todos os feriados que existem ao longo dos anos.

Para tal fiz o seguinte

```
CREATE TABLE TAB_feriado (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    designacao NVARCHAR(255) NOT NULL,
    detalhes TEXT NOT NULL,
    repete_mesma_data.todos_anos BOOL NOT NULL DEFAULT (0)
);
```

- Esta tabela apenas grava os nomes dos feriados, detalhes e se repete todos os anos no mesmo dia, ou seja, se o campo repete_mesma_data.todos_anos for como verdadeiro existe um trigger que no começo de todos os anos encontra o último registo na tabela dia_feriado que seja deste feriado e coloca um novo na mesma data e mês para o próximo ano. Será mais bem explicado na parte de criação do trigger

TAB_dia_feriado

Com os feriados definidos fiz a tabela para armazenar os dias de cada feriado, como por exemplo no Ano Novo, 2024-01-01; 2025-01-01; 2026-01-01;

```
CREATE TABLE TAB_dia_feriado (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_feriado INT NOT NULL,
    data DATE NOT NULL,
    n_dias INT NOT NULL DEFAULT (1),
    FOREIGN KEY (ID_feriado) REFERENCES TAB_feriado(ID)
);
```

TAB_horario

Depois das tabelas dos feriados fiz a tabela para guardar os horários das instalações. Como a Dona Elvira é uma empresaria ela começou a comprar bastantes espaços novos como novas bijuterias, armazéns e escritórios para poder gerir tudo em grande escala

```

CREATE TABLE TAB_horario (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    data_entrada_vigor DATE NOT NULL DEFAULT (DATE_ADD(CURRENT_DATE(), INTERVAL 1 WEEK)),
    ID_feriado INT,
    ID_instalacoes INT NOT NULL,
    FOREIGN KEY (ID_feriado) REFERENCES TAB_feriado(ID),
    FOREIGN KEY (ID_instalacoes) REFERENCES TAB_instalacoes(ID)
);

```

- Aqui a tabela guarda os horários normais e especiais. Com especial quero dizer em feriados, para tal basta colocar o ID_feriado para ser considerado o horário para aquele feriado.

TAB_horas

Para dizer as horas eu fiz uma tabela horas que diz quando abre e fecha, depois disso usei a seguinte tabela na tabela TAB_horas_semana

```

CREATE TABLE TAB_horas (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    hora_aberto TIME NOT NULL,
    hora_fechado TIME NOT NULL
);

```

TAB_dia_semana

Nesta tabela eu indico todos os dias da semana e as horas de trabalho esperadas

```

CREATE TABLE TAB_dia_semana (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    designacao NVARCHAR(255) NOT NULL,
    n_horas_trabalho_esperado INT NOT NULL DEFAULT (7)
);

```

TAB_horas_semana

Nesta tabela eu junto 3 dados importantes as horas (TAB_horas) o dia da semana (TAB_dia_semana) e o horário ao qual faz parte (TAB_horario)

```
CREATE TABLE TAB_horas_semana (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_dia_semana INT NOT NULL,
    ID_horas INT NOT NULL,
    ID_horario INT NOT NULL,
    FOREIGN KEY (ID_dia_semana) REFERENCES TAB_dia_semana(ID),
    FOREIGN KEY (ID_horas) REFERENCES TAB_horas(ID),
    FOREIGN KEY (ID_horario) REFERENCES TAB_horario(ID)
);
```

TAB_promocao

Depois da parte das instalações feita fiz a parte para as promoções para os clientes com a seguinte tabela.

```
CREATE TABLE TAB_promocao (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome NVARCHAR(255) NOT NULL,
    descricao TEXT NOT NULL,
    pode_ser_obtido.todos_meses BOOL NOT NULL DEFAULT (1),
    quantas_vezes_pode_ser_obtido INT,
    data_criacao DATE NOT NULL DEFAULT (CURRENT_DATE())
);
```

TAB_validade_promocao

Esta tabela indica quantos dias a pessoa tem para usar a promoção depois de a receber

```
CREATE TABLE TAB_validade_promocao (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_promocao INT NOT NULL,
    n_dias_valido_depois_reenviadado INT,
    data_termino_promocao DATE,
    FOREIGN KEY (ID_promocao) REFERENCES TAB_promocao(ID)
);
```

TAB_promocao_idade_necessaria

Esta tabela diz qual a idade min e/ou max que necessita ter para poder receber a promoção

```
CREATE TABLE TAB_promocao_idade_necessaria (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_promocao INT NOT NULL,
    idade_min TINYINT,
    idade_max TINYINT,
    FOREIGN KEY (ID_promocao) REFERENCES TAB_promocao(ID),
    CONSTRAINT chk_idade_min_max_nao_sao_null CHECK (idade_min IS NOT NULL OR idade_max IS NOT NULL)
);
```

TAB_promocao_desconto_monetario_geral

Esta diz qual o desconto que a pessoa recebe a nível monetário se tiver a promoção

```
CREATE TABLE TAB_promocao_desconto_monetario_geral (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_promocao INT NOT NULL,
    valor DECIMAL(10,2),
    percentagem DECIMAL(5,2),
    FOREIGN KEY (ID_promocao) REFERENCES TAB_promocao(ID),
    CONSTRAINT chk_valor_percentagem_sao_null CHECK (valor IS NOT NULL OR percentagem IS NOT NULL)
);
```

TAB_metodo_pagamento

Esta tabela irá indicar os métodos de pagamento existentes e as suas comissões para usar

```
CREATE TABLE TAB_metodo_pagamento (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    designacao NVARCHAR(255) NOT NULL,
    comissao_percentagem DECIMAL(5,2),
    comissao_valor DECIMAL(10,2),
    CONSTRAINT chk_comissao_percentagem_valor_sao_null CHECK (comissao_percentagem IS NOT NULL OR comissao_valor IS NOT NULL)
);
```

TAB_promocao_metodo

Esta tabela de promoção usa a tabela que acabei de criar (TAB_metodo_pagamento) e caso coloque o id de um método de pagamento associado com uma promoção a pessoa pode ganhar uma promoção ou logo na altura um desconto, dependendo do campo acao

Ou seja, se tiver na acao_necessario, se a pessoa usar mbway (por exemplo) ela ganha a promoção associada.

Se por outro lado tiver a acao aplicar e a pessoa usar mbway e ela tiver a promoção associada ela ganha logo o desconto monetário

```

CREATE TABLE TAB_promocao_metodo (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_promocao INT NOT NULL,
    ID_metodo_pagamento INT NOT NULL,
    acao VARCHAR(15) NOT NULL CHECK (acao IN ('necessario', 'aplicar')),
    FOREIGN KEY (ID_promocao) REFERENCES TAB_promocao(ID),
    FOREIGN KEY (ID_metodo_pagamento) REFERENCES TAB_metodo_pagamento(ID)
);

```

TAB_promocao_cliente

Esta tabela é a tabela que atribui as promoções aos clientes

```

CREATE TABLE TAB_promocao_cliente (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_cliente INT NOT NULL,
    ID_promocao INT NOT NULL,
    data_reenvidicado DATE NOT NULL DEFAULT (CURRENT_DATE()),
    FOREIGN KEY (ID_cliente) REFERENCES TAB_cliente(ID),
    FOREIGN KEY (ID_promocao) REFERENCES TAB_promocao(ID)
);

```

TAB_tipos_artigos

Finalmente, com descontos, instalações, RH, falta o essencial : artigos

Coisa que esta tabela começa definido os tipos de artigos (pulseiras, brincos, anéis, etc...)

```

CREATE TABLE TAB_tipos_artigos (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    designacao NVARCHAR(255) NOT NULL,
    detalhes TEXT
);

```

TAB_fornecedores

Depois pensei em adicionar os fornecedores dos artigos, para poder ter informações sobre quais fornecedores

```
CREATE TABLE TAB_fornecedores (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nome_empresa NVARCHAR(255) NOT NULL,
    NIF_empresa VARCHAR(30) NOT NULL
);
```

TAB_artigos

Depois fiz a tabela para armazenar artigos

```
CREATE TABLE TAB_artigos (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_fornecedor INT NOT NULL,
    ID_tipo_artigo INT NOT NULL,
    descricao TEXT NOT NULL,
    FOREIGN KEY (ID_fornecedor) REFERENCES TAB_fornecedores(ID),
    FOREIGN KEY (ID_tipo_artigo) REFERENCES TAB_tipos_artigos(ID)
);
```

TAB_promocao_tipo_artigo

Voltando nas tabelas promoções fiz a tabela promoção para os tipos de artigos

A acao é o mesmo que a TAB_promocao_metodo_pagamento

```
CREATE TABLE TAB_promocao_tipo_artigo (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_promocao INT NOT NULL,
    ID_tipo_artigo INT NOT NULL,
    acao VARCHAR(20) CHECK (acao IN ('compra', 'desconto')),
    quantidade INT NOT NULL DEFAULT (1),
    desconto DECIMAL(5,2),
    FOREIGN KEY (ID_promocao) REFERENCES TAB_promocao(ID),
    FOREIGN KEY (ID_tipo_artigo) REFERENCES TAB_tipos_artigos(ID),
    CONSTRAINT chk_acao_desconto_artigo CHECK (
        (acao = 'desconto' AND desconto IS NOT NULL AND desconto <= 100) OR
        (acao = 'compra' AND desconto IS NULL)
    )
);
```

TAB_promocao_artigo

Depois fiz a tabela para guardar as promoções em cada artigo

```
CREATE TABLE TAB_promocao_artigo (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_promocao INT NOT NULL,
    ID_artigo INT NOT NULL,
    acao VARCHAR(20) CHECK (acao IN ('compra', 'desconto')),
    quantidade INT NOT NULL DEFAULT (1),
    desconto DECIMAL(5,2),
    FOREIGN KEY (ID_promocao) REFERENCES TAB_promocao(ID),
    FOREIGN KEY (ID_artigo) REFERENCES TAB_artigos(ID),
    CONSTRAINT chk_acao_desconto_tipo_artigo CHECK (acao = 'desconto' AND desconto IS NOT NULL OR acao = 'compra' AND desconto IS NULL)
);
```

TAB_tamanho

Com artigos fiz agora o uma tabela para poder indicar o tamanho de cada artigo, e defino o tamanho de um tipo de artigo

```
CREATE TABLE TAB_tamanho (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    designacao NVARCHAR(255) NOT NULL,
    ID_tipo_artigo INT NOT NULL,
    tamanho_min_cm FLOAT NOT NULL,
    tamanho_max_cm FLOAT NOT NULL,
    FOREIGN KEY (ID_tipo_artigo) REFERENCES TAB_tipos_artigos(ID)
);
```

TAB_tamanho_artigo

Esta tabela armazena o tamanho de cada artigo

```
CREATE TABLE TAB_tamanho_artigo (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_tamanho INT NOT NULL,
    ID_artigo INT NOT NULL,
    desde DATE NOT NULL DEFAULT (CURRENT_DATE()),
    FOREIGN KEY (ID_tamanho) REFERENCES TAB_tamanho(ID),
    FOREIGN KEY (ID_artigo) REFERENCES TAB_artigos(ID)
);
```

TAB_stock_artigo

Com artigos, fornecedores e detalhes dos artigos fiz a tabela para inserir os abastecimentos de stock, qual instalação foi abastecida, a data e hora que foi pedido o stock, a data e hora que foi enviado o stock, a data e hora que chegou e o método de pagamento ultizado para pagar o valor_total (o valor total do stock, não de cada artigo)

```
CREATE TABLE TAB_stock_artigo (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_artigo INT NOT NULL,
    quantidade INT NOT NULL DEFAULT (1),
    data_hora_requesitado DATETIME NOT NULL DEFAULT (NOW()),
    data_hora_envio DATETIME,
    data_hora_chegada DATETIME,
    ID_instalacoes_destino INT NOT NULL,
    ID_metodo_pagamento INT,
    valor_total DECIMAL(10,2), -- Valor total da compra para stock do cujo
    FOREIGN KEY (ID_artigo) REFERENCES TAB_artigos(ID),
    FOREIGN KEY (ID_instalacoes_destino) REFERENCES TAB_instalacoes(ID),
    FOREIGN KEY (ID_metodo_pagamento) REFERENCES TAB_metodo_pagamento(ID)
);
```

TAB_percentagem_lucro_artigo

Depois fiz a tabela TAB_percentagem_lucro_artigo que armazena o lucro sobre cada artigo

```
CREATE TABLE TAB_percentagem_lucro_artigo (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_artigo INT NOT NULL,
    percentagem_lucro DECIMAL(5,2) NOT NULL DEFAULT (15),
    data_hora_mudado DATETIME NOT NULL DEFAULT (NOW()),
    data_hora_aplicado DATETIME NOT NULL DEFAULT (DATE_ADD(NOW(), INTERVAL 4 DAY))
);
```

TAB_transferencias

Fiz a tabela transferência que serve para registar as transferências de produtos

Nesta tabela é registado o funcionário que fez a transferência e as instalações ao qual receberam os produtos e a data e hora que transferência foi terminada (ou seja, que os produtos chegaram)

```

CREATE TABLE TAB_transferencias (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_funcionario_responsavel_transferencia INT NOT NULL,
    ID_instalacoes_destino INT NOT NULL,
    data_hora_termino_transferencia DATETIME,
    FOREIGN KEY (ID_funcionario_responsavel_transferencia) REFERENCES TAB_funcionario(ID),
    FOREIGN KEY (ID_instalacoes_destino) REFERENCES TAB_instalacoes(ID)
);

```

TAB_artigo_para_transferencia

Esta tabela funciona com os registos da tabela transferência e serve para adicionar artigos ao “teórico” camião da transferência. Visto que por sua vez a transferência pode ser feita de qualquer produto, de qualquer quantidade e qualquer instalação

```

CREATE TABLE TAB_artigo_para_transferencia (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_transferencia INT NOT NULL,
    ID_artigo INT NOT NULL,
    ID_instalacoes_origem INT NOT NULL,
    quantidade INT NOT NULL,
    data_hora_adicionado DATETIME NOT NULL DEFAULT (NOW()),
    detalhes TEXT,
    FOREIGN KEY (ID_transferencia) REFERENCES TAB_transferencias(ID),
    FOREIGN KEY (ID_artigo) REFERENCES TAB_artigos(ID),
    FOREIGN KEY (ID_instalacoes_origem) REFERENCES TAB_instalacoes(ID)
);

```

TAB_venda

A tabela venda é a tabela que regista todas as vendas realizadas

Para isso tenho o ID do cliente que compra, o ID do artigo, o ID do método do pagamento usado para fazer a compra, um campo bool “compra_online” que caso tiver a true significa que a pessoa fez a compra online, ID_instalacoes_compra_recolha é um campo que indica a instalação que foi realizada a compra ou se foi feita online o local em que é feita a recolha. O ID do funcionário o funcionário que fez a venda

Depois as datas e horas servem para armazenar a data e hora que foi feito o pedido em data_hora_pedido, e a data_hora_recolha a data e hora da recolha do artigo.

```

CREATE TABLE TAB_venda (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_artigo INT NOT NULL,
    ID_metodo_pagamento INT NOT NULL,
    ID_cliente INT NOT NULL,
    compra_online BOOL NOT NULL,
    ID_instalacoes_compra_recolha INT NOT NULL,
    ID_funcionario INT NOT NULL,
    data_hora_pedido DATETIME NOT NULL DEFAULT (NOW()),
    data_hora_recolha DATETIME,
    FOREIGN KEY (ID_artigo) REFERENCES TAB_artigos(ID),
    FOREIGN KEY (ID_metodo_pagamento) REFERENCES TAB_metodo_pagamento(ID),
    FOREIGN KEY (ID_cliente) REFERENCES TAB_cliente(ID),
    FOREIGN KEY (ID_instalacoes_compra_recolha) REFERENCES TAB_instalacoes(ID),
    FOREIGN KEY (ID_funcionario) REFERENCES TAB_funcionario(ID)
);

```

TAB_troca

Para trocar artigos fiz a tabela TAB_troca que serve para trocar um artigo comprado na tabela TAB_venda

```

CREATE TABLE TAB_troca (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_venda INT NOT NULL, -- Venda em que foi vendido o artigo que está a ser trocado por outro
    motivo TEXT NOT NULL, -- Motivo para tal
    ID_artigo_dado INT NOT NULL, -- Artigo que foi dado na troca do recebido
    ID_funcionario INT NOT NULL, -- Funcionario que fez a troca
    ID_instalacoes INT NOT NULL, -- Instalacoes que foi feita a troca
    data_hora DATETIME NOT NULL DEFAULT (NOW()), -- as horas que aconteceu a troca
    FOREIGN KEY (ID_venda) REFERENCES TAB_venda(ID),
    FOREIGN KEY (ID_artigo_dado) REFERENCES TAB_artigos(ID),
    FOREIGN KEY (ID_funcionario) REFERENCES TAB_funcionario(ID),
    FOREIGN KEY (ID_instalacoes) REFERENCES TAB_instalacoes(ID)
);

```

EVENTS E TRIGGERS

Com as tabelas criadas fiz alguns events, triggers e tabelas de logs. Como tabelas de logs não estão no meu diagrama por não terem relações com as minhas tabelas existentes eu vou colocar a criação das tabelas de logs juntamente aqui

verificar_aniversariantes

Para tal, comecei com uma event que todos excuta todos os dias a meia-noite uma inserção na tabela de logs, a logs_aniversariantes, através dos registo na tabela pessoas e que tenham registo ligados na tabela clientes

```
DELIMITER //

CREATE EVENT verifica_aniversariantes
ON SCHEDULE EVERY 1 DAY
STARTS CURRENT_DATE() + INTERVAL 1 DAY
DO
BEGIN
    CREATE TABLE IF NOT EXISTS logs_aniversariantes (
        ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        ID_cliente INT NOT NULL,
        data_aniversario DATE NOT NULL,
        FOREIGN KEY (ID_cliente) REFERENCES TAB_cliente(ID)
    );
    INSERT INTO logs_aniversariantes (ID_cliente, data_aniversario)
    SELECT c.ID, CURDATE()
    FROM TAB_pessoa p INNER JOIN TAB_cliente c ON p.ID = c.ID_pessoa
    WHERE proxima_data(data_nascimento) = CURDATE();
END;
// 

DELIMITER ;
```

CREATE TABLE IF NOT EXISTS logs_clientes

Depois do event para fazer registo de adições, alterações e remoções de clientes fiz a seguinte tabela para guardar os mesmos

```
CREATE TABLE IF NOT EXISTS logs_clientes (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_cliente INT NOT NULL,
    data_hora DATETIME NOT NULL DEFAULT (NOW()),
    acao VARCHAR(30) NOT NULL CHECK (acao IN ('INSERT', 'DELETE', 'UPDATE')),
    detalhes TEXT NOT NULL
);
```

TRIGGERS da tabela TAB_cliente

Depois de criada a tabela que irá armazenar os logs do cliente fiz uns quantos triggers para gerir as inserções de registos no log.

Começando pelas adições de registos na tabela TAB_cliente fiz este o trigger adicao_cliente como mostra abaixo

```
CREATE TRIGGER adicao_cliente AFTER INSERT ON TAB_cliente
FOR EACH ROW
BEGIN
    DECLARE nome_pessoa NVARCHAR(255);

    SELECT CONCAT(nome, ' ', sobrenome) INTO nome_pessoa
    FROM TAB_pessoa
    WHERE ID = NEW.ID_pessoa;

    INSERT INTO logs_clientes (ID_cliente, acao, detalhes)
    VALUES (NEW.ID, 'INSERT', CONCAT('Foi inserido um novo cliente com o ID pessoa ', NEW.ID_pessoa, ' e chamada ', nome_pessoa));
END;
//
```

Depois do de adição fiz um para as remoções, ou seja, o trigger remocao_cliente como possível observar abaixo

```
CREATE TRIGGER remocao_cliente AFTER DELETE ON TAB_cliente
FOR EACH ROW
BEGIN
    DECLARE nome_pessoa NVARCHAR(255);

    SELECT CONCAT(nome, ' ', sobrenome) INTO nome_pessoa
    FROM TAB_pessoa
    WHERE ID = OLD.ID_pessoa;

    INSERT INTO logs_clientes (ID_cliente, acao, detalhes)
    VALUES (OLD.ID, 'DELETE', CONCAT('Foi removido um cliente com o ID pessoa ', OLD.ID_pessoa, ' e chamada ', nome_pessoa));
END;
//
```

Por fim fiz um para as atualizações de registos, o trigger update_cliente como mostrado abaixo

```
CREATE TRIGGER update_cliente AFTER UPDATE ON TAB_cliente
FOR EACH ROW
BEGIN
    DECLARE nome_pessoa NVARCHAR(255);

    INSERT INTO logs_clientes (ID_cliente, acao, detalhes)
    VALUES (OLD.ID, 'UPDATE', CONCAT('Foi atualizado um cliente que tinha o ID pessoa ', OLD.ID_pessoa, ' para ', NEW.ID_pessoa));
END;
//
```

CREATE TABLE IF NOT EXISTS logs_pessoas

Com os triggers para registarem as alterações nos clientes fiz uma tabela que servirá para guardar os logs da tabela TAB_pessoa, a tabela da qual a TAB_cliente tem uma foreign key desta tabela ao qual estou a fazer estes logs. Para tal fiz o seguinte

```

CREATE TABLE IF NOT EXISTS logs_pessoas (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_cliente INT NOT NULL,
    data_hora DATETIME NOT NULL DEFAULT (NOW()),
    acao VARCHAR(30) NOT NULL CHECK (acao IN ('INSERT', 'DELETE', 'UPDATE')),
    detalhes TEXT NOT NULL
);

```

TRIGGERS da tabela TAB_pessoa

Como os triggers de certa forma são iguais aos anteriores, mudando apenas a tabela de ação do trigger e a tabela de inserção dos logs. Eu não irei entrar em muitos detalhes nos triggers de adição e remoção, mas são os seguintes

```

CREATE TRIGGER adicao_pessoa AFTER INSERT ON TAB_pessoa
FOR EACH ROW
BEGIN
    INSERT INTO logs_pessoas (ID_cliente, acao, detalhes)
    VALUES (NEW.ID, 'INSERT', CONCAT('Foi inserido uma nova pessoa com o nome ', NEW.nome, ' ', NEW.sobrenome, ' e data de nascimento na data ', NEW.data_nascimento));
END;
// 

CREATE TRIGGER remocao_pessoa AFTER DELETE ON TAB_pessoa
FOR EACH ROW
BEGIN
    INSERT INTO logs_pessoas (ID_cliente, acao, detalhes)
    VALUES (OLD.ID, 'DELETE', CONCAT('Foi removida uma pessoa com o nome ', OLD.nome, ' ', OLD.sobrenome));
END;
// 

```

Quanto ao de update a situação é, diferente...

```

CREATE TRIGGER update_pessoa AFTER UPDATE ON TAB_pessoa
FOR EACH ROW
BEGIN
    DECLARE text_ocurration_description VARCHAR(255);

    IF NEW.nome <> OLD.nome AND NEW.sobrenome <> OLD.sobrenome AND NEW.data_nascimento <> OLD.data_nascimento THEN
        SET text_ocurration_description = CONCAT('Foi atualizado todos os dados de uma pessoa com o nome ', OLD.nome, ' ', OLD.sobrenome, ' e data de nascimento no dia ', OLD.data_nascimento, ' para ', NEW.nome, ' ', NEW.sobrenome, ' e data de nascimento ', NEW.data_nascimento);
    ELSE
        IF NEW.nome <> OLD.nome AND NEW.sobrenome <> OLD.sobrenome AND NEW.data_nascimento = OLD.data_nascimento THEN
            SET text_ocurration_description = CONCAT('Foi atualizado o nome completo de uma pessoa com o nome ', OLD.nome, ' ', OLD.sobrenome, ' para ', NEW.nome, ' ', NEW.sobrenome);
        ELSEIF (NEW.nome <> OLD.nome OR NEW.sobrenome <> OLD.sobrenome) AND NEW.data_nascimento <> OLD.data_nascimento THEN
            IF NEW.nome <> OLD.nome THEN
                SET text_ocurration_description = CONCAT('Foi atualizado o nome de uma pessoa com o nome ', OLD.nome, ' para ', NEW.nome);
            END IF;

            IF NEW.sobrenome <> OLD.sobrenome THEN
                SET text_ocurration_description = CONCAT('Foi atualizado o nome de uma pessoa com o sobrenome ', OLD.sobrenome, ' para ', NEW.sobrenome);
            END IF;
        ELSEIF (NEW.nome <> OLD.nome OR NEW.sobrenome <> OLD.sobrenome) AND NEW.data_nascimento = OLD.data_nascimento THEN
            IF NEW.nome <> OLD.nome THEN
                SET text_ocurration_description = CONCAT('Foi atualizado o nome e data de nascimento de uma pessoa com o nome ', OLD.nome, ' e data de nascimento ', OLD.data_nascimento, ' para ', NEW.nome, ' e data nascimento ', NEW.data_nascimento);
            END IF;
        END IF;
        IF NEW.sobrenome <> OLD.sobrenome THEN
            SET text_ocurration_description = CONCAT('Foi atualizado o nome e data de nascimento de uma pessoa com o sobrenome ', OLD.sobrenome, ' e data de nascimento ', OLD.data_nascimento, ' para ', NEW.sobrenome, ' e data nascimento ', NEW.data_nascimento);
        END IF;
    END IF;
    IF NEW.data_nascimento <> OLD.data_nascimento THEN
        SET text_ocurration_description = CONCAT('Foi atualizado a data de nascimento de uma pessoa que tinha para dia ', OLD.data_nascimento, ' para ', NEW.data_nascimento);
    END IF;
    END IF;

    INSERT INTO logs_pessoas (ID_cliente, acao, detalhes)
    VALUES (OLD.ID, 'UPDATE', text_ocurration_description);
END;
// 

DELIMITER ;

```

Como podem várias coisas atualizar, não só um ID, para ser detalhado no update fiz vários ifs dentro do trigger e caso um ou vários se aplicarem ele escreve o texto que se encontra presente no SET text_ocurration_description = CONCAT(...)

TRIGGERS para artigo em stock

Para os artigos em stock também fiz uns triggers como eles são apenas 2 (um de insert outro de update) e tem a mesma estrutura eu vou explicar apenas 1

```
CREATE TRIGGER trg_stock_artigo_before_insert
BEFORE INSERT ON TAB_stock_artigo
FOR EACH ROW
BEGIN
    -- Se data_hora_chegada não for NULL, data_hora_envio também não pode ser NULL
    IF NEW.data_hora_chegada IS NOT NULL AND NEW.data_hora_envio IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'data_hora_envio não pode ser NULL quando data_hora_chegada não é NULL';
    END IF;

    -- Se data_hora_envio for NULL, ID_metodo_pagamento deve ser NULL e valor_total pode ser NULL
    IF NEW.data_hora_envio IS NOT NULL AND (NEW.ID_metodo_pagamento IS NULL OR NEW.valor_total IS NULL) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'ID_metodo_pagamento e/ou valor_total deve ter valores quando data_hora_envio não é NULL';
    END IF;

    -- Se metodo_pagamento não for NULL e valor_total é null ou vice versa
    IF (NEW.ID_metodo_pagamento IS NOT NULL AND NEW.valor_total IS NULL) OR (NEW.ID_metodo_pagamento IS NULL AND NEW.valor_total IS NOT NULL) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'valor_total ou metodo_pagamento não pode ser NULL quando o valor_total ou metodo_pagamento não é NULL';
    END IF;
END // 

CREATE TRIGGER trg_stock_artigo_before_update
BEFORE UPDATE ON TAB_stock_artigo
FOR EACH ROW
BEGIN
    -- Se data_hora_chegada não for NULL, data_hora_envio também não pode ser NULL
    IF NEW.data_hora_chegada IS NOT NULL AND NEW.data_hora_envio IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'data_hora_envio não pode ser NULL quando data_hora_chegada não é NULL';
    END IF;

    -- Se data_hora_envio for NULL, ID_metodo_pagamento deve ser NULL e valor_total pode ser NULL
    IF NEW.data_hora_envio IS NOT NULL AND (NEW.ID_metodo_pagamento IS NULL OR NEW.valor_total IS NULL) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'ID_metodo_pagamento e/ou valor_total deve ter valores quando data_hora_envio não é NULL';
    END IF;

    -- Se metodo_pagamento não for NULL e valor_total é null ou vice versa
    IF (NEW.ID_metodo_pagamento IS NOT NULL AND NEW.valor_total IS NULL) OR (NEW.ID_metodo_pagamento IS NULL AND NEW.valor_total IS NOT NULL) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'valor_total ou metodo_pagamento não pode ser NULL quando o valor_total ou metodo_pagamento não é NULL';
    END IF;
END //
```

Como se observa nos triggers eles ao contrário de fazerem algo DEPOIS (AFTER) de inserir ou atualizar um registo eles fazem ANTES (BEFORE) e tem o objetivo de verificar a integridade dos registos inseridos ou atualizados.

As verificações que são feitas e são negadas caso ocorram são as seguintes:

- Se a data_hora_chegada não for null a data_hora_envio também não pode ser
- Se a data_hora_envio for null o ID_metodo_pagamento pode ser null tal como o valor_total
- Se o metodo_pagamento não for null e o valor_total for ou vice versa

Adição de registos

Depois de feitos os triggers e os events fiz uma quantas adições de registos iniciais em algumas tabelas, apenas as que tem um número de dados limitados e que são previsíveis

INSERT INTO TAB_profissao

Começando pela tabela das profissões inseri os seguintes registos

```

INSERT INTO TAB_profissao (designacao, salario_base)
VALUES
    ('Vendedor', 800.00),
    ('Designer de Joias', 2000.00),
    ('Ourives', 1800.00),
    ('Consultor de Vendas', 900.00),
    ('Assistente Administrativo', 750.00),
    ('Supervisor de Produção', 1300.00),
    ('Técnico em Gemologia', 2200.00),
    ('Programador', 1800.00),
    ('Analista de Sistemas', 2000.00),
    ('Administrador de Redes', 2200.00);

```

Depois fui fazendo todos os outros por ordem. Como não muda muito só vou mostrar como ficaram feitos os outros e irei apenas explicar se necessário

INSERT INTO TAB_tipo_instalacoes

```

INSERT INTO TAB_tipo_instalacoes (designacao, descricao)
VALUES
    ('Loja', 'Espaço destinado à exposição e venda de bijuterias e acessórios.'),
    ('Escritório', 'Local para atividades administrativas, gestão e atendimento aos clientes.'),
    ('Armazém', 'Espaço para armazenamento de artigos.'),
    ('Showroom', 'Ambiente para exibição de peças exclusivas e coleções especiais de bijuterias.');

```

INSERT INTO TAB_tipos_manutencao

```

INSERT INTO TAB_tipos_manutencao (designacao, descricao, normalmente_afeta_normal_func)
VALUES
    ('Limpeza', 'Remoção de sujeira e acúmulo de resíduos das instalações e equipamentos.', FALSE),
    ('Reparação', 'Correção de danos e defeitos em máquinas, equipamentos ou estruturas.', TRUE),
    ('Manutenção Preventiva', 'Inspeção e reparo periódicos para evitar falhas e prolongar a vida útil.', FALSE),
    ('Substituição de Peças', 'Troca de componentes desgastados ou danificados por novos.', TRUE),
    ('Calibração', 'Ajuste e verificação de equipamentos para garantir precisão e desempenho adequado.', FALSE);

```

INSERT INTO TAB_unidades_tempo

```

INSERT INTO TAB_unidades_tempo (no_singular, no_plural)
VALUES
    ('segundo', 'segundos'),
    ('minuto', 'minutos'),
    ('hora', 'horas'),
    ('dia', 'dias'),
    ('semana', 'semanas'),
    ('mês', 'meses'),
    ('ano', 'anos');

```

INSERT INTO TAB_relacoes_entre_unidades_tempo

```
INSERT INTO TAB_relacoes_entre_unidades_tempo (ID_unidade_tempo, n_unidades_tempo_para_proxima, ID_proxima_unidade_tempo)
VALUES
(1, 60, 2),
(2, 60, 3),
(3, 24, 4),
(4, 7, 5),
(5, 4.34812141, 6),
(6, 12, 7);
```

INSERT INTO TAB_promocao

```
INSERT INTO TAB_promocao (nome, descricao, pode_ser_obtido.todos_meses, quantas_vezes_pode_ser_obtido)
VALUES
('aniversario', 'quando for seu aniversario terá um desconto de 10% em todos os artigos na nossa bijuteria', 1, NULL),
('pulseira dos amigos', 'Ao comprar qualquer pulseira ganhe uma 100% gratis', 0, 1);
```

INSERT INTO TAB_tipos_artigos

```
INSERT INTO TAB_tipos_artigos (designacao)
VALUES
('Aneis'),
('Pulseiras'),
('Brincos'),
('Colares'),
('Tornozeleiras'),
('Broches'),
('Pingentes'),
('Chokers'),
('Alfinetes de Gravata');
```

INSERT INTO TAB_promocao_tipo_artigo

```
INSERT INTO TAB_promocao_tipo_artigo (ID_promocao, ID_tipo_artigo, acao, quantidade, desconto)
VALUES
(2, 2, 'compra', 1, NULL),
(2, 2, 'desconto', 1, 100);
```

INSERT INTO TAB_metodo_pagamento

```
INSERT INTO TAB_metodo_pagamento (designacao, comissao_percentagem, comissao_valor)
VALUES
    ('Cartão de Crédito', 1.5, NULL),
    ('Cartão de Débito', 1.0, NULL),
    ('MB Way', 1.2, NULL),
    ('Transferência Bancária', NULL, 0),
    ('Paypal', 2.9, NULL),
    ('Multibanco', 0.8, NULL),
    ('Dinheiro', NULL, 0);
```

INSERT INTO TAB_feriado

```
INSERT INTO TAB_feriado (designacao, detalhes)
VALUES
    ('Ano novo', 'Dia que comemora a passagem do ano'),
    ('Sexta-feira Santa', 'É a maior data religiosa cristã que celebra o crucifício de Jesus Cristo e sua morte no Calvário'),
    ('Páscoa', 'Dia em que se comemoram festividades religiosas e um feriado que celebra a ressurreição de Jesus ocorrida no terceiro dia após sua crucificação no Calvário, conforme o relato do Novo Testamento.'),
    ('Carnaval', 'O dia da Liberdade, é comemorado em Portugal a 25 de abril. Este é um dos 13 Feriados nacionais obrigatórios. A data celebra a revolta dos militares portugueses.'), 
    ('Dia do trabalhador', 'dia dedicado aos trabalhadores'),
    ('Corpo de Deus', 'A Solenidade do Santíssimo Sacramento do Corpo e do Sangue de Cristo ou Corpus Christi ou Corpus Domini, é generalizada em Portugal como Corpo de Deus, é uma comemoração litúrgica das Igrejas Católicas'),
    ('Dia de Portugal', 'o dia de Portugal, de Camões e das Comunidades Portuguesas celebra a data de 10 de Junho de 1139, data da morte de Conde, sendo também este o dia dedicado ao Aleijadinho de Portugal. Este é também o dia da Língua Portuguesa, dos cidadãos e das Forças Armadas.'), 
    ('Dia de Santo António', 'Santo católico nascido a 15 de agosto de 1195, em Lisboa, e falecido a 13 de Junho de 1231, em Fátima. Foi assim escolhido o dia 13 para a sua celebração.'), 
    ('Dia de São João', 'Portugal, o Dia de São João é celebrado no dia 24 de junho. São João é, tal como Santo António e São Pedro, um santo popular muito festivo em Portugal.'), 
    ('Dia da Assunção de Nossa Senhora', 'A Solenidade da Assunção do Bem-aventurada Virgem Maria é celebrada no dia 15 de agosto, desde o século V, com o significado de "nascimento para o céu" ou, segundo a tradição bilíngue, de "Assunção".'), 
    ('Dia da Restauração da Independência', 'Celebração da Restauração da Independência de Portugal, é o nome que se dá ao golpe de Estado revolucionário ocorrido a 1 de Dezembro de 1896.'), 
    ('Dia da Restauração da Independência', 'a restauração da Independência ou restauração de Portugal, é o nome que se dá ao golpe de Estado revolucionário ocorrido a 1 de Dezembro de 1896.'), 
    ('Dia de Iniciada Concílio', 'O dia da Iniciada concílio, celebrado no dia 8 de dezembro, é feriado nacional. Este dia honra a vida e a virtude de Virgem Maria, mãe de Jesus.'), 
    ('Natal', 'a data é o centro das festas de fim de ano e da temporada de férias, sendo, no cristianismo, o marco inicial do ciclo do natal, que dura doze dias.');
```

INSERT INTO TAB_dia_feriado

```
INSERT INTO TAB_dia_feriado (ID_feriado, data, n_dias)
VALUES
(1, '2024-01-01', 1),
(1, '2025-01-01', 1),
(1, '2026-01-01', 1),
(2, '2024-03-29', 1),
(2, '2025-04-18', 1),
(2, '2026-04-03', 1),
(3, '2024-03-31', 1),
(3, '2025-04-20', 1),
(3, '2026-04-05', 1),
(4, '2024-04-25', 1),
(4, '2024-04-25', 1),
(4, '2024-04-25', 1),
(5, '2024-05-01', 1),
(5, '2025-05-01', 1),
(5, '2026-05-01', 1),
(6, '2024-05-30', 1),
(6, '2025-06-19', 1),
(6, '2026-06-04', 1),
(7, '2024-06-10', 1),
(7, '2025-06-10', 1),
(7, '2026-06-10', 1),
(8, '2024-06-13', 1),
(8, '2025-06-13', 1),
(8, '2026-06-13', 1),
(9, '2024-06-24', 1),
(9, '2025-05-24', 1),
(9, '2026-05-24', 1),
(10, '2024-08-15', 1),
(10, '2025-08-15', 1),
(10, '2026-08-15', 1),
(11, '2024-10-05', 1),
(11, '2025-10-05', 1),
(11, '2026-10-05', 1),
(12, '2024-11-01', 1),
(12, '2025-11-01', 1),
(12, '2026-11-01', 1),
(13, '2024-12-01', 1),
(13, '2025-12-01', 1),
(13, '2026-12-01', 1),
(14, '2024-12-08', 1),
(14, '2025-12-08', 1),
(14, '2026-12-08', 1),
(15, '2024-12-25', 3),
(15, '2025-12-25', 3),
```

INSERT INTO TAB_instalacoes

```
INSERT INTO TAB_instalacoes (designacao, ID_tipo_instalacoes, morada, localizacao_gps, data_hora_primeira_abertura, data_hora_ultimo_encerramento)
VALUES
('Loja Central', 1, 'Rua Principal, 123', ST_GeomFromText('POINT(38.7223 -9.1393)'), '2022-01-01 09:00:00', NULL),
('Loja Filial', 1, 'Avenida Secundária, 456', ST_GeomFromText('POINT(38.7111 -9.1432)'), '2022-02-15 10:00:00', NULL),
('Escritório Administrativo', 2, 'Avenida das Empresas, 789', ST_GeomFromText('POINT(38.7199 -9.1411)'), '2022-01-20 09:30:00', NULL),
('Armazém Central', 3, 'Rua dos Armazéns, 1011', ST_GeomFromText('POINT(38.7200 -9.1399)'), '2022-01-01 08:00:00', NULL),
('Showroom', 4, 'Praça das Exibições, 1213', ST_GeomFromText('POINT(38.7250 -9.1400)'), '2022-04-05 11:00:00', NULL);
```

INSERT INTO TAB_horas

```
INSERT INTO TAB_horas (hora_aberto, hora_fechado)
VALUES
('09:00:00', '13:00:00'),
('09:00:00', '12:00:00'),
('10:00:00', '13:00:00'),
('14:00:00', '18:00:00'),
('14:00:00', '17:00:00'),
('14:00:00', '17:30:00'),
('15:00:00', '18:00:00');
```

INSERT INTO TAB_dia_semana

```
INSERT INTO TAB_dia_semana (designacao, n_horas_trabalho_esperado)
VALUES
('Domingo', 4),
('Segunda-Feira', 9),
('Terça-Feira', 9),
('Quarta-Feira', 9),
('Quinta-Feira', 9),
('Sexta-Feira', 8),
('Sábado', 0);
```

INSERT INTO TAB_horario

```
INSERT INTO TAB_horario (data_entrada_vigor, ID_feriado, ID_instalacoes)
VALUES
(DATE_ADD(CURDATE(), INTERVAL 7 DAY), NULL, 1);
```

INSERT INTO TAB_horas_semana

```
INSERT INTO TAB_horas_semana (ID_dia_semana, ID_horas, ID_horario)
VALUES
(1, 7, 1),
(2, 1, 1),
(2, 4, 1),
(3, 1, 1),
(3, 4, 1),
(4, 1, 1),
(4, 4, 1),
(5, 1, 1),
(5, 4, 1),
(6, 1, 1),
(6, 5, 1);
```

INSERT INTO TAB_pessoa

```
INSERT INTO TAB_pessoa (nome, sobrenome, nif, data_nascimento)
VALUE ('Elvira Maria', 'Silva Santos', '208581430', '1960-03-15');
```

INSERT INTO TAB_funcionario

```
INSERT INTO TAB_funcionario (ID_pessoa, ID_profissao)
VALUE ((SELECT MAX(ID) FROM TAB_pessoa), 1);
```

INSERT INTO TAB_cargos

```
INSERT INTO TAB_cargos (designacao, funcoes, salario_extra_cargo)
VALUES
('CEO', 'Responsável pela gestão geral da empresa e decisões estratégicas', 4000.00),
('Diretor Financeiro', 'Responsável pela gestão financeira da empresa', 3000.00),
('Diretor de Marketing', 'Responsável pelas estratégias de marketing e promoção', 2500.00),
('Gerente de Recursos Humanos', 'Responsável pela gestão do departamento de RH', 1500.00),
('Gerente de TI', 'Responsável pela gestão da infraestrutura de TI', 2000.00),
('Gerente de Operações', 'Responsável pela supervisão das operações diárias', 2200.00),
('Gerente de Loja', 'Responsável pela gestão e operação de uma loja específica', 500.00),
('Chefe de Armazém', 'Responsável pela supervisão do armazém e controle de estoque', 1000.00),
('Supervisor de Vendas', 'Responsável pela supervisão da equipe de vendas', 800.00),
('Assistente Executivo', 'Apóio administrativo aos executivos da empresa', 200.00),
('Coordenador de Logística', 'Responsável pela coordenação das atividades logísticas', 1100.00),
('Analista de Marketing', 'Análise e execução de campanhas de marketing', 500.00),
('Especialista em SEO', 'Optimização do site e conteúdo para motores de busca', 700.00),
('Coordenador de Vendas', 'Coordenar atividades e estratégias de vendas', 1000.00),
('Chefe de Produção', 'Supervisionar e coordenar a produção de bijuterias', 1300.00),
('Gerente de Projetos', 'Gerenciar e coordenar projetos específicos', 1500.00),
('Consultor de TI', 'Fornecer consultoria técnica e suporte de TI', 1200.00),
('Líder de Equipe de Design', 'Liderar a equipe de designers de joias', 1400.00),
('Coordenador de Segurança da Informação', 'Responsável pela segurança dos dados e informações', 1600.00),
('Chefe de Atendimento ao Cliente', 'Supervisionar e coordenar a equipe de atendimento ao cliente', 900.00),
('Ourives', 'Fabricação e design de joias', 0.00), -- Cargo base que correspondente à profissão de Ourives
('Designer de Joias', 'Design e criação de joias', 0.00), -- Cargo base que correspondente à profissão de Designer de Joias
('Vendedor', 'Venda e assistência ao cliente', 0.00), -- Cargo base que correspondente à profissão de Vendedor
('Técnico de Gemologia', 'Análise e avaliação de gemas', 0.00), -- Cargo base que correspondente à profissão de Técnico de Gemologia
('Programador Júnior', 'Desenvolvimento de software e aplicativos - Nível Júnior', 0.00), -- Cargo base que correspondente à profissão de Programador Júnior
('Programador', 'Desenvolvimento de software e aplicativos - Nível Pleno', 1200.00), -- Cargo base que correspondente à profissão de Programador
('Programador Sênior', 'Desenvolvimento de software e aplicativos - Nível Sênior', 1800.00), -- Cargo base que correspondente à profissão de Programador Sênior
('Administrador de Redes', 'Gerenciamento e manutenção de redes', 0.00); -- Cargo base que correspondente à profissão de Administrador de Redes
```

INSERT INTO TAB_hierarquia

```
INSERT INTO TAB_hierarquia (ID_cargo_atribuindo, ID_cargo_superior)
VALUES
    (1, NULL),          -- CEO
    (2, 1),            -- Diretor Financeiro -> CEO
    (3, 1),            -- Diretor de Marketing -> CEO
    (4, 1),            -- Gerente de Recursos Humanos -> CEO
    (5, 1),            -- Gerente de TI -> CEO
    (6, 1),            -- Gerente de Operações -> CEO
    (7, 6),            -- Gerente de Loja -> Gerente de Operações
    (8, 6),            -- Chefe de Armazém -> Gerente de Operações
    (9, 6),            -- Supervisor de Vendas -> Gerente de Operações
    (10, 1),           -- Assistente Executivo -> CEO
    (11, 8),           -- Coordenador de Logística -> Chefe de Armazém
    (12, 3),           -- Analista de Marketing -> Diretor de Marketing
    (13, 3),           -- Especialista em SEO -> Diretor de Marketing
    (14, 9),           -- Coordenador de Vendas -> Supervisor de Vendas
    (15, 6),           -- Chefe de Produção -> Gerente de Operações
    (16, 1),           -- Gerente de Projetos -> CEO
    (17, 5),           -- Consultor de TI -> Gerente de TI
    (18, 3),           -- Líder de Equipe de Design -> Diretor de Marketing
    (19, 5),           -- Coordenador de Segurança da Informação -> Gerente de TI
    (20, 6),           -- Chefe de Atendimento ao Cliente -> Gerente de Operações
    (21, 7),           -- Ourives -> Gerente de Loja
    (22, 7),           -- Designer de Joias -> Gerente de Loja
    (23, 7),           -- Vendedor -> Gerente de Loja
    (24, 8),           -- Técnico em Gemologia -> Chefe de Armazém
    (25, 17),          -- Programador Júnior -> Consultor de TI
    (26, 17),          -- Programador Pleno -> Consultor de TI
    (27, 17),          -- Programador Sênior -> Consultor de TI
    (28, 5),           -- Administrador de Redes -> Gerente de TI
    (22, 16);          -- Designer de Joias -> Gerente de Projetos
```

INSERT INTO TAB_promocoes_cargos

```
INSERT INTO TAB_promocoes_cargos (ID_funcionario_promovido, ID_cargo)
VALUE
    (1, 1);
```

INSERT INTO TAB_experiencia

```
INSERT INTO TAB_experiencia (ID_funcionario, n_anos, salario_extra_experiencia)
VALUE
    (1, 1, 100);
```

INSERT INTO TAB_tamanho

```
INSERT INTO TAB_tamanho (designacao, ID_tipo_artigo, tamanho_min_cm, tamanho_max_cm)
VALUES
    ('Aneis Pequenos', 1, 1.0, 2.0),
    ('Aneis Médios', 1, 2.1, 2.5),
    ('Aneis Grandes', 1, 2.6, 3.0),
    ('Pulseiras Pequenas', 2, 16.0, 18.0),
    ('Pulseiras Médias', 2, 18.1, 20.0),
    ('Pulseiras Grandes', 2, 20.1, 22.0),
    ('Brincos Pequenos', 3, 0.5, 1.0),
    ('Brincos Médios', 3, 1.1, 2.0),
    ('Brincos Grandes', 3, 2.1, 3.0),
    ('Colares Curtos', 4, 40.0, 45.0),
    ('Colares Médios', 4, 45.1, 50.0),
    ('Colares Longos', 4, 50.1, 60.0),
    ('Tornozeleiras Pequenas', 5, 19.0, 21.0),
    ('Tornozeleiras Médias', 5, 21.1, 23.0),
    ('Tornozeleiras Grandes', 5, 23.1, 25.0),
    ('Broches Pequenos', 6, 1.0, 2.0),
    ('Broches Médios', 6, 2.1, 3.5),
    ('Broches Grandes', 6, 3.6, 5.0),
    ('Pingentes Pequenos', 7, 1.0, 2.0),
    ('Pingentes Médios', 7, 2.1, 3.5),
    ('Pingentes Grandes', 7, 3.6, 5.0),
    ('Chokers Curtos', 8, 30.0, 35.0),
    ('Chokers Médios', 8, 35.1, 40.0),
    ('Chokers Longos', 8, 40.1, 45.0),
    ('Alfinetes de Gravata Pequenos', 9, 1.0, 2.0),
    ('Alfinetes de Gravata Médios', 9, 2.1, 3.5),
    ('Alfinetes de Gravata Grandes', 9, 3.6, 5.0);
```

INSERT INTO TAB_fornecedores

```
INSERT INTO TAB_fornecedores (nome_empresa, NIF_empresa)
VALUES
    ('Fornecedor A', '123456789'),
    ('Fornecedor B', '987654321'),
    ('Fornecedor C', '112233445'),
    ('Fornecedor D', '556677889'),
    ('Fornecedor E', '998877665');
```

INSERT INTO TAB_artigos

```
INSERT INTO TAB_artigos (ID_fornecedor, ID_tipo_artigo, descricao)
VALUES
    (1, 1, 'Anel de ouro 18k com diamantes'),
    (1, 2, 'Pulseira de prata esterlina'),
    (2, 3, 'Brinco de pérola natural'),
    (2, 4, 'Colar de ouro branco com safira'),
    (3, 5, 'Tornozeleira de couro trançado'),
    (3, 6, 'Broche vintage com pedras preciosas'),
    (4, 7, 'Pingente de jade esculpido'),
    (4, 8, 'Choker de renda preta com cristal'),
    (5, 9, 'Alfinete de gravata com design minimalista'),
    (5, 1, 'Anel de prata com esmeralda'),
    (1, 2, 'Pulseira de ouro rosé com rubis'),
    (2, 3, 'Brinco de prata com topázio azul');
```

INSERT INTO TAB_tamanho_artigo

```
INSERT INTO TAB_tamanho_artigo (ID_tamanho, ID_artigo)
VALUES
    (2, 1), -- Aneis Médios para o Anel de ouro 18k com diamantes
    (5, 2), -- Pulseiras Médias para a Pulseira de prata esterlina
    (8, 3), -- Brincos Médios para o Brinco de pérola natural
    (11, 4), -- Colares Médios para o Colar de ouro branco com safira
    (14, 5), -- Tornozeleiras Médias para a Tornozeleira de couro trançado
    (17, 6), -- Broches Médios para o Broche vintage com pedras preciosas
    (20, 7), -- Pingentes Médios para o Pingente de jade esculpido
    (23, 8), -- Chokers Médios para o Choker de renda preta com cristal
    (26, 9), -- Alfinetes de Gravata Médios para o Alfinete de gravata com design minimalista
    (2, 10), -- Aneis Médios para o Anel de prata com esmeralda
    (5, 11), -- Pulseiras Médias para a Pulseira de ouro rosé com rubis
    (8, 12); -- Brincos Médios para o Brinco de prata com topázio azul
```

INSERT INTO TAB_percentagem_lucro_artigo

```
INSERT INTO TAB_percentagem_lucro_artigo (ID_artigo, percentagem_lucro)
VALUES
    (1, 20.00),
    (2, 18.50),
    (3, 22.00),
    (4, 25.00),
    (5, 19.75),
    (6, 21.50),
    (7, 23.00),
    (8, 20.50),
    (9, 18.00),
    (10, 19.00),
    (11, 20.00),
    (12, 21.00);
```

INSERT INTO TAB_stock_artigo

```
INSERT INTO TAB_stock_artigo (ID_artigo, quantidade, ID_instalacoes_destino, data_hora_envio, ID_metodo_pagamento, valor_total, data_hora_chegada)
VALUES
    (1, 10, 4, '2024-06-02 20:40', 3, 100.00, '2024-06-03 10:00:00'),
    (1, 5, 4, '2024-06-02 21:40', 3, 100.00, '2024-06-03 12:00:00');
```

INSERT INTO TAB_venda

```
INSERT INTO TAB_venda (ID_artigo, ID_metodo_pagamento, ID_cliente, ID_funcionario, compra_online, ID_instalacoes_compra_recolha)
VALUES
(
    1,
    (SELECT ID FROM TAB_metodo_pagamento ORDER BY RAND() LIMIT 1),
    (SELECT ID FROM TAB_cliente ORDER BY RAND() LIMIT 1),
    (SELECT ID FROM TAB_funcionario ORDER BY RAND() LIMIT 1),
    FALSE,
    (SELECT ID FROM TAB_instalacoes WHERE designacao LIKE "%loja%" ORDER BY RAND() LIMIT 1)
);
```

INSERT INTO TAB_transferencias

```
INSERT INTO TAB_transferencias (ID_funcionario_responsavel_transferencia, ID_instalacoes_destino, data_hora_termino_transferencia)
VALUES
    ((SELECT ID FROM TAB_funcionario ORDER BY RAND() LIMIT 1), 1, '2024-06-03 19:35:00'),
    ((SELECT ID FROM TAB_funcionario ORDER BY RAND() LIMIT 1), 2, '2024-06-03 19:40:00'),
    ((SELECT ID FROM TAB_funcionario ORDER BY RAND() LIMIT 1), 1, '2024-06-10 20:00:00'),
    ((SELECT ID FROM TAB_funcionario ORDER BY RAND() LIMIT 1), 2, '2024-06-10 20:00:00');
```

INSERT INTO TAB_artigo_para_transferencia

```
INSERT INTO TAB_artigo_para_transferencia (ID_transferencia, ID_artigo, ID_instalacoes_origem, quantidade, data_hora_adicionado, detalhes)
VALUES
    (1, 1, 4, 3, '2024-06-03 19:25:00', ''),
    (2, 1, 4, 3, '2024-06-03 19:30:00', 'Houve um pequeno atraso'),
    (3, 1, 4, 2, '2024-06-03 20:00:00', 'Ta a haver uma pequena greve'),
    (4, 1, 4, 2, '2024-06-05 19:30:00', 'Ta a haver uma pequena greve');
```

Tabelas Auxiliares

Depois de feitas as adições dos registos iniciais em algumas tabelas fiz umas tabelas auxiliares na adição de mais registos, de forma aleatória, nas tabelas sem registos até ao momento.

TAB_AUX_pessoa

Começando com a tabela auxiliar para inserir registos na tabela TAB_pessoa, a TAB_AUX_pessoa que tem a seguinte estrutura

```
CREATE TABLE TAB_AUX_pessoa (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    primeiro_nome NVARCHAR(50) NOT NULL,
    segundo_nome NVARCHAR(50),
    terceiro_nome NVARCHAR(50),
    quarto_nome NVARCHAR(50) NOT NULL
);
```

TAB_AUX_cargos_profissoes

Depois das pessoas fiz outra para as profissões, que será usada para inserir dados na tabela TAB_profissão.

```
CREATE TABLE TAB_AUX_cargos_profissoes (
    ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ID_cargo INT NOT NULL,
    ID_profissao INT NOT NULL,
    FOREIGN KEY (ID_cargo) REFERENCES TAB_cargos(ID),
    FOREIGN KEY (ID_profissao) REFERENCES TAB_profissao(ID)
);
```

Adicionar dados as tabelas auxiliares

Com as tabelas auxiliares criadas adicionei os registos as duas tabelas

INSERT INTO TAB_AUX_pessoa

Começando com a tabela auxiliar das pessoas adicionei os seguintes nomes

```
INSERT INTO TAB_AUX_pessoa (primeiro_nome, segundo_nome, terceiro_nome, quarto_nome)
VALUES
    ('João', 'Pedro', 'Silva', 'Santos'),
    ('Maria', 'Clara', 'Oliveira', 'Pereira'),
    ('Carlos', 'Eduardo', 'Mendes', 'Gomes'),
    ('Ana', 'Luísa', 'Almeida', 'Fernandes'),
    ('Paulo', 'Roberto', 'Barbosa', 'Lima'),
    ('Fernanda', 'Cristina', 'Ribeiro', 'Martins'),
    ('José', 'Antonio', 'Costa', 'Sousa'),
    ('Mariana', 'Beatriz', 'Dias', 'Rocha'),
    ('Ricardo', 'Alexandre', 'Nogueira', 'Carvalho'),
    ('Luciana', 'Fernandes', 'Teixeira', 'Vieira'),
    ('Pedro', 'Henrique', 'Ferreira', 'Azevedo'),
    ('Juliana', 'Santos', 'Moreira', 'Correia'),
    ('Rafael', 'Rodrigues', 'Gonçalves', 'Melo'),
    ('Isabela', 'Freitas', 'Lopes', 'Moura'),
    ('Tiago', 'Felipe', 'Moraes', 'Silveira'),
    ('Camila', 'Fernanda', 'Pires', 'Ramos'),
    ('Gabriel', 'Duarte', 'Vieira', 'Santos'),
    ('Lorena', 'Sophia', 'Nunes', 'Silva'),
    ('Gustavo', 'Lucas', 'Carvalho', 'Martins'),
    ('Aline', 'Carolina', 'Lima', 'Alves'),
    ('Daniel', 'Oliveira', NULL, 'Pereira'),
    ('Beatriz', 'Macedo', NULL, 'Teixeira'),
    ('Liliana', 'de', 'Oliveira', 'Pratas'),
    ('Francisco', 'Rafael', 'Santos', 'Casado');
```

INSERT INTO TAB_AUX_cargos_profissoes

Quanto aos cargos de cada profissão coloquei os seguintes

```

INSERT INTO TAB_AUX_cargos_profissoes (ID_cargo, ID_profissao)
VALUES
    (2, 4), -- Diretor Financeiro
    (3, 1), -- Diretor de Marketing
    (4, 6), -- Gerente de Recursos Humanos
    (5, 9), -- Gerente de TI
    (6, 5), -- Gerente de Operações
    (7, 1), -- Gerente de Loja
    (8, 5), -- Chefe de Armazém
    (9, 1), -- Supervisor de Vendas
    (10, 5), -- Assistente Executivo
    (11, 5), -- Coordenador de Logística
    (12, 4), -- Analista de Marketing
    (13, 9), -- Especialista em SEO
    (14, 4), -- Coordenador de Vendas
    (15, 6), -- Chefe de Produção
    (16, 6), -- Gerente de Projetos
    (17, 9), -- Consultor de TI
    (18, 2), -- Líder de Equipe de Design
    (19, 9), -- Coordenador de Segurança da Informação
    (20, 1), -- Chefe de Atendimento ao Cliente
    (21, 3), -- Ourives
    (22, 2), -- Designer de Joias
    (23, 1), -- Vendedor
    (24, 7), -- Técnico de Gemologia
    (25, 8), -- Programador Júnior
    (26, 8), -- Programador
    (27, 8), -- Programador Sênior
    (28, 10); -- Administrador de Redes

```

Functions e Procedures

Com as tabelas auxiliares feitas e com registos, fiz umas quantas funções e procedimentos

FUNCTION obter_nome_aleatorio

Começando com a function obter_nome_aleatorio que devolve um nome aleatório juntando nas primeiras 2 colunas da tabela auxiliar TAB_AUX_pessoa e pegando e 2 registos diferentes

```

CREATE FUNCTION obter_nome_aleatorio()
RETURNS VARCHAR(255) READS SQL DATA
BEGIN
    DECLARE primeiro_nome_aleatorio NVARCHAR(50);
    DECLARE segundo_nome_aleatorio NVARCHAR(50);
    DECLARE nome NVARCHAR(110);

    SELECT primeiro_nome INTO primeiro_nome_aleatorio
    FROM TAB_AUX_pessoa
    ORDER BY RAND()
    LIMIT 1;

    SELECT segundo_nome INTO segundo_nome_aleatorio
    FROM TAB_AUX_pessoa
    ORDER BY RAND()
    LIMIT 1;

    IF segundo_nome_aleatorio IS NULL THEN
        SET nome = primeiro_nome_aleatorio;
    ELSE
        SET nome = CONCAT(primeiro_nome_aleatorio, ' ', segundo_nome_aleatorio);
    END IF;

    RETURN nome;
END;
//
```

FUNCTION obter_sobrenome_aleatorio

Faz praticamente o mesmo que a anterior, mas em vez de ser as primeiras 2 colunas são as 2 últimas. Devolvendo dessa forma um sobrenome.

```

CREATE FUNCTION obter_sobrenome_aleatorio()
RETURNS VARCHAR(255) READS SQL DATA
BEGIN
    DECLARE terceiro_nome_aleatorio NVARCHAR(50);
    DECLARE quarto_nome_aleatorio NVARCHAR(50);
    DECLARE sobrenome NVARCHAR(110);

    SELECT terceiro_nome INTO terceiro_nome_aleatorio
    FROM TAB_AUX_pessoa
    ORDER BY RAND()
    LIMIT 1;

    SELECT quarto_nome INTO quarto_nome_aleatorio
    FROM TAB_AUX_pessoa
    ORDER BY RAND()
    LIMIT 1;

    IF terceiro_nome_aleatorio IS NULL THEN
        SET sobrenome = quarto_nome_aleatorio;
    ELSE
        SET sobrenome = CONCAT(terceiro_nome_aleatorio, ' ', quarto_nome_aleatorio);
    END IF;

    RETURN sobrenome;
END;
//

```

FUNCTION gerar_nif

Ainda envolvendo as pessoas (mas podendo também ser aplicável as empresas) fiz a função gerar_nif para me devolver um nif aleatório.

Para elaboração disto usei o ChatGPT devido a minha necessidade de isto ser um NIF realístico

```

CREATE FUNCTION gerar_nif()
RETURNS CHAR(9)
DETERMINISTIC
BEGIN
    DECLARE nif_base CHAR(8);
    DECLARE nif CHAR(9);
    DECLARE sum INT;
    DECLARE check_digit INT;

    SET nif_base = LPAD(FLOOR(RAND() * 100000000), 8, '0');

    SET sum = 0;
    SET sum = sum + SUBSTRING(nif_base, 1, 1) * 9;
    SET sum = sum + SUBSTRING(nif_base, 2, 1) * 8;
    SET sum = sum + SUBSTRING(nif_base, 3, 1) * 7;
    SET sum = sum + SUBSTRING(nif_base, 4, 1) * 6;
    SET sum = sum + SUBSTRING(nif_base, 5, 1) * 5;
    SET sum = sum + SUBSTRING(nif_base, 6, 1) * 4;
    SET sum = sum + SUBSTRING(nif_base, 7, 1) * 3;
    SET sum = sum + SUBSTRING(nif_base, 8, 1) * 2;

    SET check_digit = 11 - (sum % 11);

    IF check_digit >= 10 THEN
        SET check_digit = 0;
    END IF;

    SET nif = CONCAT(nif_base, check_digit);

    RETURN nif;
END //

```

FUNCTION gerar_cc

Depois ainda fiz a função gerar_cc que como indica o nome gera um número de um documento de identificação de forma aleatória.

Neste também usei o GPT

```

CREATE FUNCTION gerar_cc()
RETURNS CHAR(12)
DETERMINISTIC
BEGIN
    DECLARE letras CHAR(2);
    DECLARE numeros CHAR(9);
    DECLARE cc CHAR(12);

    -- Gerar duas letras aleatórias (A-Z)
    SET letras = CHAR(FLOOR(65 + RAND() * 26));
    SET letras = CONCAT(letras, CHAR(FLOOR(65 + RAND() * 26)));

    -- Gerar nove números aleatórios (0-9)
    SET numeros = LPAD(FLOOR(RAND() * 1000000000), 9, '0');

    -- Concatenar letras e números para formar o ID do CC
    SET cc = CONCAT(letras, numeros);

    RETURN cc;
END //

```

FUNCTION gerar_passaporte

Mesma coisa que os anteriores

```

CREATE FUNCTION gerar_passaporte()
RETURNS CHAR(9)
DETERMINISTIC
BEGIN
    DECLARE letras CHAR(3);
    DECLARE numeros CHAR(6);
    DECLARE passaporte CHAR(9);

    -- Gerar três letras aleatórias (A-Z)
    SET letras = CHAR(FLOOR(65 + RAND() * 26));
    SET letras = CONCAT(letras, CHAR(FLOOR(65 + RAND() * 26)));
    SET letras = CONCAT(letras, CHAR(FLOOR(65 + RAND() * 26)));

    -- Gerar seis números aleatórios (0-9)
    SET numeros = LPAD(FLOOR(RAND() * 1000000), 6, '0');

    -- Concatenar letras e números para formar o ID do passaporte
    SET passaporte = CONCAT(letras, numeros);

    RETURN passaporte;
END //

```

PROCEDURE ExecutarInsercaoRegistrosMultiplosPessoas

Finalmente, depois de algumas functions fiz o meu primeiro procedure desta base de dados, o ExecutarInsercaoRegistrosMultiplosPessoas que como o nome indica, ele executa a inserção de várias pessoas como registo na tabela TAB_pessoa bastando indicar o número de pessoas que se pretende adicionar. Mostrarei mais a frente ele a ser chamado, por enquanto o procedure foi feito da seguinte maneira

```

CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplosPessoas(IN vezes INT)
BEGIN
    DECLARE contador INT;

    SET contador = 1;

    WHILE contador <= vezes DO
        INSERT INTO TAB_pessoa (nome, sobrenome, NIF, data_nascimento)
        VALUES (
            obter_nome_aleatorio(),
            obter_sobrenome_aleatorio(),
            gerar_nif(),
            DATE_SUB(CURRENT_DATE(), INTERVAL FLOOR(RAND() * (365 * (90 - 18) + 1) + (365 * 18)) DAY)
        );
        SET contador = contador + 1;
    END WHILE;
END;
//
```

CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplosClientes

Depois de ter um procedure para adicionar umas quantas pessoas fiz um parecido para adicionar uns quantos clientes. Ficou da seguinte forma

```
CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplosClientes(IN vezes INT)
BEGIN
    DECLARE contador INT;
    DECLARE ID_cliente_aleatorio INT;

    SET contador = 1;

    WHILE contador <= vezes DO
        SELECT ID INTO ID_cliente_aleatorio
        FROM TAB_pessoa
        WHERE ID NOT IN (SELECT ID_pessoa FROM TAB_cliente)
        ORDER BY RAND()
        LIMIT 1;

        INSERT INTO TAB_cliente (ID_pessoa, data_hora_registro)
        VALUES (
            ID_cliente_aleatorio,
            DATE_SUB(CURRENT_DATE(), INTERVAL FLOOR(RAND() * 79200) MINUTE)
        );
        SET contador = contador + 1;
    END WHILE;
END;
//
```

CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplosFuncionarios

Este procedure é igual ao anterior mudado apenas um pouco os campos ao qual vai buscar os dados para adicionar

```

CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplosFuncionarios(IN vezes INT)
BEGIN
    DECLARE contador INT;
    DECLARE ID_funcionario_aleatorio INT;
    DECLARE ID_profissao_aleatorio INT;

    SET contador = 1;

    WHILE contador <= vezes DO
        SELECT ID INTO ID_funcionario_aleatorio
            FROM TAB_pessoa
            WHERE ID NOT IN (SELECT ID_pessoa FROM TAB_funcionario)
            ORDER BY RAND()
            LIMIT 1;

        SELECT ID INTO ID_profissao_aleatorio
            FROM TAB_profissao
            ORDER BY RAND()
            LIMIT 1;

        INSERT INTO TAB_funcionario (ID_pessoa, ID_profissao)
        VALUES (
            ID_funcionario_aleatorio,
            ID_profissao_aleatorio
        );
        SET contador = contador + 1;
    END WHILE;
END;
//
```

FUNCTION existem_pessoas_com_mais_1_conta_cliente

Voltando as functions fiz uma para me dizer se existe pessoas com 1 ou mais contas de cliente.

Como a tabela pessoa guarda todas as pessoas registadas, se a mesma pessoa tiver 2 ou mais regtos na tabela cliente esta function retorna como true (1).

```

CREATE FUNCTION existem_pessoas_com_mais_1_conta_cliente(ID_pessoa_avaliar INT)
RETURNS BOOLEAN READS SQL DATA
BEGIN
    DECLARE n_pessoas_registadas_mais_que_1_cliente INT;

    -- Seleciona o número de registros de clientes para a pessoa especificada
    SELECT COUNT(*) INTO n_pessoas_registadas_mais_que_1_cliente
    FROM TAB_cliente
    WHERE ID_pessoa = ID_pessoa_avaliar;

    -- Verifica se a pessoa está registrada mais de uma vez como cliente
    IF n_pessoas_registadas_mais_que_1_cliente > 1 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;
//

```

Esta function foi útil no início que a minha adição de registo tava a fazer registo de clientes com IDs de pessoa igual

PROCEDURE ApagarRegistrosRepetidosClientes

Como comentado anteriormente, houve uma altura que tinha muitos registo de clientes repetidos, então para combater esse erro fiz este procedimento que poderia depois usar num EVENT ou mesmo um TRIGGER para remover todas as noites ou assim que fosse acrescentado um cliente novo

```

CREATE PROCEDURE ApagarRegistrosRepetidosClientes()
BEGIN
    DECLARE contador INT;
    DECLARE ID_cliente_aleatorio INT;

    -- Encontra o maior ID na tabela TAB_cliente
    SET contador = (SELECT MAX(ID) FROM TAB_cliente);

    -- Loop enquanto o contador for maior que 0
    WHILE contador > 0 DO
        -- Verifica se o cliente tem mais de uma conta
        SET ID_cliente_aleatorio = (SELECT ID_pessoa FROM TAB_cliente WHERE ID = contador);

        IF ID_cliente_aleatorio IS NOT NULL THEN
            IF existem_pessoas_com_mais_1_conta_cliente(ID_cliente_aleatorio) THEN
                -- Apaga o registro com o ID atual se for um duplicado
                DELETE FROM TAB_cliente WHERE ID = contador;
            END IF;
        END IF;

        -- Decrementa o contador
        SET contador = contador - 1;
    END WHILE;
END;
//

```

FUNCTION proxima_data

Com pessoas, temos aniversários, e aniversários não passam de uma data que acontece no mesmo dia e mês de todos os anos. Para tal fiz a function proxima_data que devolve a próxima vez que essa data vai acontecer. Ou seja, se a pessoa faz anos no dia 31/12/2000 ele vai retornar 31/12/2024. Se fizer no dia 01/01/2000 e considerando a sua data atual (06/06/2024) ele vai retornar 01/01/2025 pois a data 01/01/2024 já passou

```

CREATE FUNCTION proxima_data(data_verificar DATE)
RETURNS DATE READS SQL DATA
BEGIN
    DECLARE data_neste_ano DATE;
    DECLARE proxima_data_obtida DATE;

    SET data_neste_ano = CONCAT(DATE_FORMAT(CURDATE(), '%Y'), '-' ,DATE_FORMAT(data_verificar, '%m-%d'));

    IF data_neste_ano < CURDATE() THEN
        SET proxima_data_obtida = DATE_ADD(data_neste_ano, INTERVAL 1 YEAR);
    ELSE
        SET proxima_data_obtida = data_neste_ano;
    END IF;

    RETURN proxima_data_obtida;
END;
//

```

FUNCTION dia_feriado_muda_cada_ano

Como na tabela TAB_dia_feriado guardo todos os dias que há os feriados eu consigo através dele dizer se o feriado se muda de dia/mês que acontece a cada ano.

Para tal fiz a seguinte função

```
CREATE FUNCTION dia_feriado_muda_cada_ano(ID_feriado_inte INT)
RETURNS BOOL READS SQL DATA
BEGIN
    DECLARE n_datas_distintas INT;
    DECLARE repete BOOL;

    SELECT COUNT(DISTINCT DATE_FORMAT(df.data, '%m-%d')) INTO n_datas_distintas
        FROM TAB_dia_feriado df INNER JOIN TAB_feriado f ON df.ID_feriado = f.ID
        WHERE f.ID = ID_feriado_inte
        GROUP BY f.ID, f.designacao;

    IF n_datas_distintas > 1 THEN
        SET repete = TRUE;
    ELSE
        SET repete = FALSE;
    END IF;

    RETURN repete;
END;
//
```

PROCEDURE ExecutarInsercaoRegistrosMultiplosCidados

Novamente nos procedures, fiz um para acrescentar registo para cidadãos

Como não é muito diferente dos outros de adição de registo não vou entrar em muitos detalhes

```

CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplosCidadoses(IN vezes INT)
BEGIN
    DECLARE contador INT;
    DECLARE ID_pessoa_aleatoria INT;

    SET contador = 1;

    WHILE contador <= vezes DO
        SELECT ID INTO ID_pessoa_aleatoria
        FROM TAB_pessoa
        WHERE ID NOT IN (SELECT ID_pessoa FROM TAB_passaporte)
        ORDER BY RAND()
        LIMIT 1;

        INSERT INTO TAB_informacoes_cidadao (ID_pessoa, CC)
        VALUES (
            ID_pessoa_aleatoria,
            gerar_cc()
        );
        SET contador = contador + 1;
    END WHILE;
END;
//
```

PROCEDURE ExecutarInsercaoRegistrosMultiplosEstrangeiros

Mesma coisa para os registos da tabela TAB_passaporte que guarda as informações dos estrangeiros

```

CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplosEstrangeiros(IN vezes INT)
BEGIN
    DECLARE contador INT;
    DECLARE ID_pessoa_aleatoria INT;

    SET contador = 1;

    WHILE contador <= vezes DO
        SELECT ID INTO ID_pessoa_aleatoria
        FROM TAB_pessoa
        WHERE ID NOT IN (SELECT ID_pessoa FROM TAB_informacoes_cidadao)
        ORDER BY RAND()
        LIMIT 1;

        INSERT INTO TAB_passaporte (ID_pessoa, ID_passaporte)
        VALUES (
            ID_pessoa_aleatoria,
            gerar_passaporte()
        );
        SET contador = contador + 1;
    END WHILE;
END;
//
```

PROCEDURE

ExecutarInsercaoRegistrosMultiplosContratosAtuaisFuncionarios

Não irei também acrescentar mais nada sobre os próximos que forem muito parecidos eu não irei comentar.

Só voltarei a comentar caso tenha algo que valha a pena ser comentado

```

CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplosContratosAtuaisFuncionarios()
BEGIN
    DECLARE contador INT;
    DECLARE id_funcionario_selecionado INT;
    DECLARE random_date DATETIME;

    SET contador = 1;

    WHILE contador <= (SELECT MAX(ID) FROM TAB_funcionario) DO
        SELECT ID INTO id_funcionario_selecionado FROM TAB_funcionario WHERE ID = contador;

        IF id_funcionario_selecionado IS NOT NULL THEN

            SET random_date = DATE_SUB(NOW(), INTERVAL FLOOR(RAND() * (90 * 60 * 60) + 1) MINUTE);

            INSERT INTO TAB_contrato (ID_funcionario, data_hora_contratado, prazo_contrato, ID_unidade_tempo_prazo_contrato)
            VALUES (
                id_funcionario_selecionado,
                random_date,
                1,
                7 -- Anos, ou seja 1 ano
            );
        END IF;

        SET contador = contador + 1;
    END WHILE;
END;
//
```

FUNCTION obter_contrato_mais_recente

Mais uma vez nas functions, fiz uma que obtém o contrato mais recente de um certo funcionário especificado.

```

CREATE FUNCTION obter_contrato_mais_recente(ID_funcionario_proc INT)
RETURNS INT READS SQL DATA
BEGIN
    RETURN (SELECT c.ID
    FROM TAB_contrato c
    INNER JOIN TAB_unidades_tempo ut ON c.ID_unidade_tempo_prazo_contrato = ut.ID
    WHERE c.ID_funcionario = ID_funcionario_proc AND (data_hora_cancelado IS NULL AND ID_funcionario_cancelou IS NULL) AND data_hora_contratado <= NOW()
    ORDER BY data_hora_contratado DESC
    LIMIT 1);
END;
//
```

Como irei agora acrescentar muitas functions com a mesma tarefa, eu não irei comentar muito ou nada sobre

FUNCTION obter_definicao_hierarquica_mais_recente

```
CREATE FUNCTION obter_definicao_hierarquica_mais_recente(ID_cargo_proc INT)
RETURNS INT READS SQL DATA
BEGIN
    RETURN (SELECT ID
        FROM TAB_hierarquia
        WHERE ID_cargo_atribuindo = ID_cargo_proc AND data_hora <= NOW()
        ORDER BY data_hora DESC
        LIMIT 1);
END;
//
```

FUNCTION obter_promocao_mais_recente

```
CREATE FUNCTION obter_promocao_mais_recente(ID_funcionario_proc INT)
RETURNS INT READS SQL DATA
BEGIN
    RETURN (SELECT ID
        FROM TAB_promocoes_cargos
        WHERE ID_funcionario_promovido = ID_funcionario_proc AND data_promovido <= CURRENT_DATE()
        ORDER BY data_promovido DESC
        LIMIT 1);
END;
//
```

FUNCTION obter_data_termino_contrato

```
CREATE FUNCTION obter_data_termino_contrato(des_no_singular VARCHAR(30), data_hora_contratado_avaliando DATETIME, prazo_contrato_obt INT)
RETURNS DATE READS SQL DATA
BEGIN
    RETURN (CASE
        WHEN des_no_singular = 'ano' THEN DATE_ADD(data_hora_contratado_avaliando, INTERVAL prazo_contrato_obt YEAR)
        WHEN des_no_singular = 'mês' THEN DATE_ADD(data_hora_contratado_avaliando, INTERVAL prazo_contrato_obt MONTH)
        WHEN des_no_singular = 'semana' THEN DATE_ADD(data_hora_contratado_avaliando, INTERVAL prazo_contrato_obt WEEK)
        WHEN des_no_singular = 'dia' THEN DATE_ADD(data_hora_contratado_avaliando, INTERVAL prazo_contrato_obt DAY)
    END);
END;
//
```

PROCEDURE ExecutarInsercaoRegistrosMultiplasPromocoes

```
CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplasPromocoes()
BEGIN
    DECLARE contador INT DEFAULT 1;
    DECLARE ID_funcionario_atribuindo_cargo INT;
    DECLARE ID_cargo_selecionado INT;

    WHILE contador <= (SELECT MAX(ID) FROM TAB_funcionario) DO
        SELECT ID INTO ID_funcionario_atribuindo_cargo FROM TAB_funcionario WHERE ID = contador AND ID_pessoa != 1;

        IF ID_funcionario_atribuindo_cargo IS NOT NULL THEN
            SELECT acp.ID_cargo INTO ID_cargo_selecionado
            FROM TAB_AUX_cargos_profissoes acp
                INNER JOIN TAB_funcionario f ON acp.ID_profissao = f.ID_profissao
            WHERE f.ID = ID_funcionario_atribuindo_cargo
            ORDER BY RAND()
            LIMIT 1;

            INSERT INTO TAB_promocoes_cargos (ID_funcionario_promovido, ID_funcionario_promovedor, ID_cargo)
            VALUE (
                ID_funcionario_atribuindo_cargo,
                1,
                ID_cargo_selecionado
            );
        END IF;

        SET contador = contador + 1;
    END WHILE;
END;
//
```

FUNCTION obter_experiencia_mais_recente

```
CREATE FUNCTION obter_experiencia_mais_recente(ID_funcionario_proc INT)
RETURNS INT READS SQL DATA
BEGIN
    RETURN (SELECT ID
    FROM TAB_experiencia
    WHERE ID_funcionario = ID_funcionario_proc AND data_hora <= NOW()
    ORDER BY data_hora DESC
    LIMIT 1);
END;
//
```

PROCEDURE

ExecutarInsercaoRegistrosMultiplasAbastecimentosStock

```
CREATE PROCEDURE ExecutarInsercaoRegistrosMultiplasAbastecimentosStock(max_iterations INT)
BEGIN
    DECLARE i INT DEFAULT 0;
    DECLARE random_ID_artigo INT;
    DECLARE random_quantidade INT;
    DECLARE random_ID_instalacoes_destino INT;
    DECLARE random_ID_metodo_pagamento INT;
    DECLARE random_valor_total DECIMAL(10, 2);
    DECLARE random_data_hora_envio DATETIME;
    DECLARE random_data_hora_chegada DATETIME;
    IF max_iterations IS NULL THEN SET max_iterations = 10; END IF;

    WHILE i < max_iterations DO
        SET random_ID_artigo = FLOOR(1 + (RAND() * (SELECT MAX(ID) FROM TAB_artigos)));
        SET random_quantidade = FLOOR(1 + (RAND() * 10));
        SET random_ID_instalacoes_destino = FLOOR(1 + (RAND() * (SELECT MAX(ID) FROM TAB_instalacoes)));
        SET random_ID_metodo_pagamento = FLOOR(1 + (RAND() * (SELECT MAX(ID) FROM TAB_metodo_pagamento)));
        SET random_valor_total = ROUND(RAND() * 1000, 2);
        SET random_data_hora_envio = NOW();
        SET random_data_hora_chegada = DATE_ADD(NOW(), INTERVAL FLOOR(1 + (RAND() * 10)) DAY);

        -- Inserir o registro na tabela TAB_stock_artigo
        INSERT INTO TAB_stock_artigo (ID_artigo, quantidade, data_hora_envio, data_hora_chegada, ID_instalacoes_destino, ID_metodo_pagamento, valor_total)
        VALUES (random_ID_artigo, random_quantidade, random_data_hora_envio, random_data_hora_chegada, random_ID_instalacoes_destino, random_ID_metodo_pagamento, random_valor_total);

        -- Incrementar o contador
        SET i = i + 1;
    END WHILE;
END
//
```

FUNCTION obter_artigos_em_stock

Esta function é uma function que me deixa muito orgulhos pois tem dois campos sendo que o 2º pode ser null.

Como funciona e porque pode ser null?

Fiz assim porque imagine que quer saber o stock de um certo artigo em uma certa instalação, se quiser isso coloca preenche os dois campos com valores possíveis e assim ele lhe devolve a existência ou inexistência do mesmo no stock dessa instalação desse artigo.

Caso queira saber em geral, ou seja, de todas as instalações basta meter o segundo valor que introduz a null que ele procura em todas as instalações

```

CREATE FUNCTION obter_artigos_em_stock(ID_artigo_proc INT, ID_instalacoes_proc INT)
RETURNS INT READS SQL DATA
BEGIN
    DECLARE n_artigos_compra INT;
    DECLARE n_artigos_ano_passado INT DEFAULT 0; -- Depois adicionar
    DECLARE n_artigos_comprados INT DEFAULT 0;
    DECLARE n_artigos_vendidos INT DEFAULT 0;
    DECLARE n_artigos_transferidos_s INT DEFAULT 0;
    DECLARE n_artigos_transferidos_e INT DEFAULT 0;
    DECLARE n_artigos_devolvidos INT DEFAULT 0;
    DECLARE n_artigos_dados INT DEFAULT 0;

    IF ID_instalacoes_proc IS NULL THEN
        SELECT SUM(quantidade) INTO n_artigos_comprados
        FROM TAB_artigos a
            INNER JOIN TAB_stock_artigo sa ON a.ID = sa.ID_artigo
        WHERE sa.data_hora_chegada <= NOW() AND a.ID = ID_artigo_proc;

        SELECT COUNT(*) INTO n_artigos_vendidos
        FROM TAB_artigos a
            INNER JOIN TAB_venda v ON a.ID = v.ID_artigo
        WHERE v.data_hora_pedido <= NOW() AND a.ID = ID_artigo_proc;

        SELECT COUNT(*) INTO n_artigos_devolvidos
        FROM TAB_venda v
            INNER JOIN TAB_troca tr ON v.ID = tr.ID_venda
        WHERE v.ID_artigo = ID_artigo_proc AND tr.data_hora <= NOW();

        SELECT COUNT(*) INTO n_artigos_dados
        FROM TAB_troca tr
        WHERE tr.ID_artigo_dado = ID_artigo_proc AND tr.data_hora <= NOW();

    ELSE
        SELECT SUM(quantidade) INTO n_artigos_comprados
        FROM TAB_artigos a
            INNER JOIN TAB_stock_artigo sa ON a.ID = sa.ID_artigo
        WHERE sa.data_hora_chegada <= NOW() AND a.ID = ID_artigo_proc AND sa.ID_instalacoes_destino = ID_instalacoes_proc;

        SELECT COUNT(*) INTO n_artigos_vendidos
        FROM TAB_artigos a
            INNER JOIN TAB_venda v ON a.ID = v.ID_artigo
        WHERE v.data_hora_pedido <= NOW() AND a.ID = ID_artigo_proc AND v.ID_instalacoes_compra_recolha = ID_instalacoes_proc;
    END IF;
END

```

Como tal é uma function enorme com 75 linhas de script

FUNCTION obter_valor_artigos_compra_media

Esta function obtém o valor medio dos últimos artigos comprados

Ou seja, ele encontra a quantidade de um certo artigo que está em stock e depois encontra os últimos reabastecimentos de stock até completar o número de quantidade de stock e assim

```

CREATE FUNCTION obter_valor_artigos_compra_media(ID_artigo_proc INT)
RETURNS DECIMAL(10,2) READS SQL DATA
BEGIN
    DECLARE valor_artigo_compra DECIMAL(10,2);
    DECLARE quantidade_em_stock INT;
    DECLARE total_quantidade INT;
    DECLARE total_valor DECIMAL(10,2);

    SET quantidade_em_stock = obter_artigos_em_stock(ID_artigo_proc, NULL);

    SELECT
        SUM(sa.quantidade) AS total_quantidade,
        SUM(sa.valor_total) AS total_valor
    INTO
        total_quantidade,
        total_valor
    FROM
        TAB_stock_artigo sa
    INNER JOIN
        TAB_metodo_pagamento mp ON sa.ID_metodo_pagamento = mp.ID
    WHERE
        sa.ID_artigo = ID_artigo_proc AND
        sa.data_hora_chegada <= NOW() AND
        (SELECT SUM(quantidade)
        FROM TAB_stock_artigo AS sa2
        WHERE sa2.data_hora_chegada <= sa.data_hora_chegada AND sa2.ID_artigo = ID_artigo_proc) <= quantidade_em_stock
    ORDER BY
        sa.data_hora_chegada DESC;

    IF total_valor IS NULL THEN SET total_valor = 0; END IF;

    SET valor_artigo_compra = total_valor / total_quantidade;

    RETURN valor_artigo_compra;
END;
//

```

FUNCTION obter_percentagem_lucro_artigo_atual

```

CREATE FUNCTION obter_percentagem_lucro_artigo_atual(ID_artigo_proc INT)
RETURNS DECIMAL(5,2) READS SQL DATA
BEGIN
    RETURN (SELECT percentagem_lucro
            FROM TAB_percentagem_lucro_artigo
            WHERE ID_artigo = ID_artigo_proc AND data_hora_aplicado <= NOW()
            ORDER BY data_hora_aplicado
            LIMIT 1);
END;
//

```

```

FUNCTION obter_data_horario_mais_recente
CREATE FUNCTION obter_data_horario_mais_recente(ID_instalacoes_proc INT, ID_feriado_proc INT)
RETURNS DATE READS SQL DATA
BEGIN
    DECLARE data_mais_recente DATE;

    IF ID_feriado_proc IS NULL THEN
        SELECT data_entrada_vigor INTO data_mais_recente
        FROM TAB_horario
        WHERE ID_instalacoes = ID_instalacoes_proc
        ORDER BY data_entrada_vigor
        LIMIT 1;
    ELSE
        SELECT data_entrada_vigor INTO data_mais_recente
        FROM TAB_horario
        WHERE ID_instalacoes = ID_instalacoes_proc AND ID_feriado = ID_feriado_proc
        ORDER BY data_entrada_vigor
        LIMIT 1;
    END IF;

    RETURN data_mais_recente;
END;
//

```

VIEWS

Com as funções e procedimentos feitos fiz umas views para facilitar nos select a base de dados

CREATE VIEW VIEW_informacoes_cliente

```

CREATE VIEW VIEW_informacoes_cliente AS
SELECT c.ID AS ID_cliente,
       pe.ID AS ID_pessoa,
       pe.nome,
       pe.sobrenome,
       pe.data_nascimento,
       pe.NIF,
       DATE_FORMAT(c.data_hora_registro, "%d-%m-%Y %H:%i:%s") AS data_hora_registado_cliente,
       CASE
           WHEN ic.ID IS NULL AND pa.ID IS NOT NULL THEN "Estrangeiro"
           WHEN ic.ID IS NOT NULL AND pa.ID IS NULL THEN "Cidadão"
           ELSE "Sem dados"
       END AS estrangeiro_ou_cidadao,
       CASE
           WHEN ic.ID IS NULL AND pa.ID IS NOT NULL THEN pa.ID_passaporte
           WHEN ic.ID IS NOT NULL AND pa.ID IS NULL THEN ic.CC
           ELSE "---"
       END AS documento_identificacao
FROM TAB_cliente c
       INNER JOIN TAB_pessoa pe ON c.ID_pessoa = pe.ID
       LEFT JOIN TAB_informacoes_cidadao ic ON pe.ID = ic.ID_pessoa
       LEFT JOIN TAB_passaporte pa ON pe.ID = pa.ID_pessoa;

```

Esta view o que faz é mostrar as informações processadas sobre os clientes como se pode ver com o select abaixo

ID_cliente	ID_pessoa	nome	sobrenome	data_nascimento	NIF	data_hora_registado_cliente	estrangeiro_ou_cidadao	documento_identificacao
19	491	Isabela Felipe	Dias Silveira	1964-10-24	013872729	22-04-2024 15:09:00	Sem dados	---
20	457	Ana Luísa	Ribeiro Vieira	1964-05-10	405555288	14-04-2024 22:22:00	Sem dados	---
21	898	Tiago Duarte	Nogueira Alves	1966-08-05	034777342	01-06-2024 06:55:00	Sem dados	---
22	89	Isabela Roberto	Oliveira Martins	1946-10-01	313575304	15-04-2024 13:58:00	Sem dados	---
23	957	Liliana Fernandes	Moraes Vieira	1951-02-23	632410434	20-04-2024 15:22:00	Sem dados	---
24	236	Gustavo Beatriz	Costa Moura	1988-04-27	810573261	28-04-2024 14:06:00	Estrangeiro	WCY462184
25	15	Camila Oliveira	Ferreira Vieira	1979-06-19	404637027	10-05-2024 03:23:00	Sem dados	---
26	877	Daniel Fernandes	Nunes Pereira	1950-10-05	330563181	05-05-2024 05:04:00	Sem dados	---
27	401	Juliana Felipe	Gonçalves Silva	1940-08-27	925554375	25-05-2024 08:08:00	Sem dados	---
28	303	Francisco Carolina	Ferreira Vieira	1986-02-26	152403647	14-05-2024 11:59:00	Sem dados	---
29	902	Fernanda António	Moreira Ramos	1983-07-04	979708095	25-05-2024 18:35:00	Sem dados	---
30	915	Liliana Antonio	Lima Sousa	1947-02-04	762432713	27-04-2024 15:41:00	Sem dados	---
31	189	Lorena Henrique	Silva Gomes	2005-09-18	446725382	27-05-2024 06:07:00	Sem dados	---
32	978	Paulo Fernandes	Teixeira Sousa	1956-12-09	971160945	18-04-2024 22:11:00	Sem dados	---
33	469	Liliana Antonio	Pires Moura	1936-03-22	931572622	02-05-2024 02:50:00	Cidadão	GA215783469
34	318	Lorena Roberto	Mendes Sousa	1953-10-05	188437339	21-05-2024 02:03:00	Sem dados	---
35	786	Ricardo Duarte	Lima	1978-07-28	080567266	15-05-2024 04:43:00	Sem dados	---
36	27	Paulo Luisa	Ferreira Ferna...	1953-08-16	515008958	12-05-2024 18:46:00	Sem dados	---
37	763	Fernanda Cristina	Moreira Azevedo	1938-01-22	100316441	12-04-2024 03:52:00	Sem dados	---
...

O select executado anteriormente pode ser encontrado no ficheiro “Trabalho final - Selects teste.sql”

CREATE VIEW VIEW_hierarquia_ordenada_cargos_atual

Esta view serve para mostrar a ordem das cargos na hierarquia e a distância que cada cargo se encontra do cargo mais alto na hierarquia (ou seja, do CEO)

```
CREATE VIEW VIEW_hierarquia_ordenada_cargos_atual AS
WITH RECURSIVE HierarquiaCompleta AS (
    SELECT h.ID, h.ID_cargo_atribuindo, h.ID_cargo_superior, 0 AS distancia
    FROM TAB_hierarquia h
    WHERE h.ID_cargo_superior IS NULL AND obter_definicao_hierarquica_mais_recente(h.ID_cargo_atribuindo) = h.ID

    UNION ALL

    SELECT h.ID, h.ID_cargo_atribuindo, h.ID_cargo_superior, distancia + 1
    FROM TAB_hierarquia h
    INNER JOIN HierarquiaCompleta hc ON h.ID_cargo_superior = hc.ID_cargo_atribuindo
    WHERE obter_definicao_hierarquica_mais_recente(h.ID_cargo_atribuindo) = h.ID
)
SELECT hc.ID, hc.ID_cargo_atribuindo, ca.designacao AS des_cargo, hc.ID_cargo_superior, cs.designacao AS des_superior, hc.distancia
FROM HierarquiaCompleta hc
INNER JOIN TAB_cargos ca ON hc.ID_cargo_atribuindo = ca.ID
LEFT JOIN TAB_cargos cs ON hc.ID_cargo_superior = cs.ID
ORDER BY hc.distancia ASC, hc.ID_cargo_superior ASC;
```

Como se pode observar abaixo, ele retorna os seguintes dados

The screenshot shows a database interface with two SQL statements and a result grid.

Statement 57: `SELECT *`

Statement 58: `FROM VIEW_hierarquia_ordenada_cargos_atual;`

The Result Grid displays the following data:

ID	ID_cargo_atribuindo	des_cargo	ID_cargo_superior	des_superior	distancia
1	1	CEO	NULL	NULL	0
2	2	Diretor Financeiro	1	CEO	1
3	3	Diretor de Marketing	1	CEO	1
4	4	Gerente de Recursos Humanos	1	CEO	1
5	5	Gerente de TI	1	CEO	1
6	6	Gerente de Operações	1	CEO	1
10	10	Assistente Executivo	1	CEO	1
16	16	Gerente de Projetos	1	CEO	1
12	12	Analista de Marketing	3	Diretor de Marketing	2
13	13	Especialista em SEO	3	Diretor de Marketing	2
18	18	Líder de Equipe de Design	3	Diretor de Marketing	2
17	17	Consultor de TI	5	Gerente de TI	2
19	19	Coordenador de Segurança d...	5	Gerente de TI	2
28	28	Administrador de Redes	5	Gerente de TI	2
7	7	Gerente de Loja	6	Gerente de Operaç...	2
8	8	Chefe de Armazém	6	Gerente de Operaç...	2
9	9	Supervisor de Vendas	6	Gerente de Operaç...	2
15	15	Chefe de Produção	6	Gerente de Operaç...	2
20	20	Chefe de Atendimento ao Clie...	6	Gerente de Operaç...	2

CREATE VIEW VIEW_informacoes_funcionario

Agora que tenho a view anterior feita fiz uma view para mostrar as informações dos funcionários, como salário

```

CREATE VIEW VIEW_informacoes_funcionario AS
SELECT f.ID AS ID_funcionario,
       p.ID AS ID_pessoa,
       p.nome AS nome,
       p.sobrenome AS sobrenome,
       DATE_FORMAT(pe.data_nascimento, "%d-%m-%Y") AS data_nascimento,
       TIMESTAMPDIFF(YEAR,pe.data_nascimento, CURRENT_DATE()) AS idade,
       pe.NIF AS NIF,
       pr.profissao AS profissao,
       vhca.des_cargo,
       -- DATE_FORMAT(ct.data_hora_contratado, "%d-%m-%Y %H:%i") AS data_hora_ultimo_contrato,
       DATE_FORMAT(otbter_data_termino_contrato(ut.no_singular, co.data_hora_contratado, co.prazo_contrato), "%d-%m-%Y") AS data_termino_contrato,
       IF (obter_data_termino_contrato(ut.no_singular, co.data_hora_contratado, co.prazo_contrato) > CURRENT_DATE() AND (co.data_hora_cancelado IS NULL OR co.data_hora_cancelado > CURRENT_DATE()), DATEIFF(obter_data_termino_contrato(ut.no_singular, co.data_hora_contratado, co.prazo_contrato), CURRENT_DATE(), DAY),
       WHEN obter_data_termino_contrato(ut.no_singular, co.data_hora_contratado, co.prazo_contrato) < CURRENT_DATE() AND (co.data_hora_cancelado IS NULL OR co.data_hora_cancelado > CURRENT_DATE()) THEN "Fechado"
       WHEN (co.data_hora_cancelado IS NOT NULL OR co.data_hora_cancelado < CURRENT_DATE()) THEN "Suspensão"
       END AS status,
       COALESCE(sa.salario_extra_cargo, 0) + COALESCE(pr.salario_base, 0) + COALESCE(exp.salario_extra_experiencia, 0) AS total_salario
FROM TAB_funcionario f
INNER JOIN TAB_pessoa pe ON f.ID_pessoa = pe.ID
INNER JOIN TAB_profissao pr ON f.ID_profissao = pr.ID
INNER JOIN TAB_contrato co ON obter_contrato_mais_recente(f.ID) = co.ID
INNER JOIN TAB_unidades_tempo ut ON co.ID_unidade_tempo_prazo_contrato = ut.ID
LEFT JOIN TAB_promocoes_cargos pc ON obter_promocao_mais_recente(f.ID) = pc.ID
LEFT JOIN TAB_cargos ca ON pc.ID_cargo = ca.ID
LEFT JOIN VIEW_hierarquia_orientada_cargos vhca ON pc.ID_cargo = vhca.ID_cargo_atribuido
LEFT JOIN TAB_experiencia exp ON obter_experiencia_mais_reciente(f.ID) = exp.ID
ORDER BY ID_funcionario
;

```

Ele retorna os seguintes dados quando executado o select

237 •

238 • SELECT *
239 • FROM VIEW_informacoes_funcionario;

240 •

	ID_funcionario	ID_pessoa	nome	sobrenome	data_nascimento	idade	NIF	profissao	des_cargo	data_termino_contrato	des_para_scabar_contrato	estado	total_salario
1	1	Elvira Maria	Silva Santos	15-03-1960	64	205861430	Vendedor	CEO	04-11-2043	7091	Trabalhando	4900,00	
2	753	Gustavo Clara	Barbosa Pereira	08-04-2002	22	587239107	Analista de Sistemas	Coordenador de Segurança da Informação	24-12-2024	202	Trabalhando	3600,00	
3	287	Ana Alexandre	Costa Sousa	27-05-1966	64	754992344	Consultor de Vendas	Coordenador de Vendas	14-12-2024	192	Trabalhando	1900,00	
4	447	Ana Duarte	Mendes Gomes	25-06-1947	76	835683143	Supervisor de Produção	Chefe de Produção	06-01-2025	215	Trabalhando	2600,00	
5	243	Paulo Luís	Gonçalves Martins	05-09-1974	47	40399862	Assistente Administrativo	Coordenador de Logística	30-05-2025	359	Trabalhando	1850,00	
6	665	Gabriel Antonio	Texeira Ramos	28-09-1967	56	828694688	Supervisora de Produção	Chefe de Produção	12-05-2025	341	Trabalhando	2600,00	
7	326	Paulo Eduardo	Ribeiro Santos	04-06-1937	87	641623437	Ourives	Ourives	27-02-2025	267	Trabalhando	1800,00	
8	537	Tiago Pelipe	Gonçalves Vieira	20-08-1981	42	283940042	Designer de Jóias	Designer de Jóias	23-11-2024	171	Trabalhando	2000,00	
9	294	Júlio Roberto	Texeira Santos	15-06-1970	53	849505070	Programador	Programador Sênior	25-10-2024	142	Trabalhando	3600,00	
10	591	Fernanda Sophia	Vieira Carvalho	14-05-1974	50	27243077	Vendedor	Vendedor	12-03-2025	280	Trabalhando	800,00	
11	791	Pedro Roberto	Moraes Casado	10-09-1971	52	119530848	Administrador de Redes	Administrador de Redes	19-11-2024	167	Trabalhando	2200,00	
12	684	Beatriz Macedo	Olíveira Martins	02-09-1966	63	526017491	Técnico em Genética	Técnico de Genética	05-04-2025	304	Trabalhando	2200,00	
13	542	Mariana Carolina	Gonçalves Martins	24-02-1959	65	250647442	Programador	Programador	02-01-2025	211	Trabalhando	3000,00	
14	865	Pedro Henrique	Nunes Pratas	29-11-1997	26	939940115	Consultor de Vendas	Diretor Financeiro	17-01-2025	226	Trabalhando	3900,00	
15	397	Beatriz Sophia	Vieira Azevedo	20-08-1985	38	419098833	Ourives	Ourives	29-05-2025	358	Trabalhando	1800,00	
16	292	Tiago Roberto	Silva Teixeira	03-08-1961	62	368391135	Analista de Sistemas	Coordenador de Segurança da Informação	05-04-2025	304	Trabalhando	3600,00	
17	677	Daniel Fernandes	Hunes Ferreira	05-10-1950	73	330563181	Analista de Sistemas	Coordenador de Segurança da Informação	08-04-2025	307	Trabalhando	3600,00	
18	51	Camila Sophia	Lima Pratas	21-06-1979	44	955012465	Técnico em Genética	Técnico de Genética	23-02-2025	263	Trabalhando	2200,00	
19	948	Isabela Pedro	Santos Pratas	22-08-1945	78	466558414	Administrador de Redes	Administrador de Redes	13-02-2025	253	Trabalhando	2200,00	

CREATE VIEW VIEW_detalhes_artigos

Fiz também uma para me dar detalhes dos artigos, como preços, quantidades em stock,

etc...

```

CREATE VIEW VIEW_detalhes_artigos AS
SELECT a.ID, a.descricao AS descricao_artigo,
       tia.designacao AS des_tipo_artigo,
       t.designacao AS tamanho,
       t.tamanho_min_cm, t.tamanho_max_cm,
       f.nome_empresa AS nome_fornecedor,
       f.NIF_empresa AS NIF_fornecedor,
       obter_artigos_em_stock(a.ID, NULL) AS artigos_em_stock,
       ROUND(obter_valor_artigos_compra_media(a.ID) + obter_valor_artigos_compra_media(a.ID) * obter_percentagem_lucro_artigo_atual(a.ID) / 100, 2) AS preco
FROM TAB_artigos a
INNER JOIN TAB_fornecedores f ON a.ID_fornecedor = f.ID
INNER JOIN TAB_tipos_artigos tia ON a.ID_tipo_artigo = tia.ID
INNER JOIN TAB_tamanno_artigo taa ON a.ID = taa.ID_artigo
INNER JOIN TAB_tamanno t ON taa.ID_tamanno = t.ID
ORDER BY a.ID
;

```

Tendo estes resultados

```

61
62 •  SELECT *
63      FROM VIEW_detalhes_artigos;
64

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

ID	descricao_artigo	des_tipo_artigo	tamanho	tamanho_min_cm	tamanho_max_cm	nome_fornecedor	NIF_fornecedor	artigos_em_stock	preco
1	Anel de ouro 18k com diamantes	Aneis	Anéis Médios	2.1	2.5	Fornecedor A	123456789	13	24.00
2	Pulseira de prata esterlina	Pulseiras	Pulseiras Médias	18.1	20	Fornecedor A	123456789	10	58.56
3	Brinco de pérola natural	Brincos	Brincos Médios	1.1	2	Fornecedor B	987654321	0	NULL
4	Colar de ouro branco com safira	Colares	Colares Médios	45.1	50	Fornecedor B	987654321	0	NULL
5	Tornozeleira de couro trançado	Tornozeleiras	Tornozeleiras Médias	21.1	23	Fornecedor C	112233445	0	NULL
6	Broche vintage com pedras preciosas	Broches	Broches Médios	2.1	3.5	Fornecedor C	112233445	0	NULL
7	Pingente de jade esculpido	Pingentes	Pingentes Médios	2.1	3.5	Fornecedor D	556677889	2	NULL
8	Choker de renda preta com cristal	Chokers	Chokers Médios	35.1	40	Fornecedor D	556677889	0	NULL
9	Alfinete de gravata com design minimalista	Alfinetes de Gravata	Alfinetes de Gravata Médios	2.1	3.5	Fornecedor E	998877665	0	NULL
10	Anel de prata com esmeralda	Aneis	Anéis Médios	2.1	2.5	Fornecedor E	998877665	4	NULL
11	Pulseira de ouro rosé com rubis	Pulseiras	Pulseiras Médias	18.1	20	Fornecedor A	123456789	0	NULL
12	Brinco de prata com topázio azul	Brincos	Brincos Médios	1.1	2	Fornecedor B	987654321	0	NULL

CREATE VIEW VIEW_horario_atual

Ultimamente, fiz a view para dizer o horário atual de cada instalação

```

CREATE VIEW VIEW_horario_atual AS
SELECT ho.ID_instalacoes,
       IF(f.ID IS NULL, '---', f.ID) AS ID_feriado,
       IF(f.ID IS NULL, '---', f.designacao) AS feriado_aplicada,
       ds.ID AS ID_dia_semana,
       ds.designacao AS dia_semana,
       h.hora_aberto,
       h.hora_fechado,
       ds.n_horas_trabalho Esperado AS horas_trabalho Esperado,
       ho.data_entrada_vigor
FROM TAB_horario ho
     INNER JOIN TAB_horas_semana hs ON ho.ID = hs.ID_horario
     INNER JOIN TAB_horas h ON hs.ID_horas = h.ID
     INNER JOIN TAB_dia_semana ds ON hs.ID_dia_semana = ds.ID
     LEFT JOIN TAB_feriado f ON ho.ID_feriado = f.ID
WHERE obter_data_horario_mais_recente(ho.ID, ho.ID_feriado) = ho.data_entrada_vigor
;
```

E com ela obtenho os seguintes resultados

```

368 •  SELECT *
369      FROM VIEW_horario_atual;
370

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

ID_instalacoes	ID_feriado	feriado_aplicada	ID_dia_semana	dia_semana	hora_aberto	hora_fechado	horas_trabalho Esperado	data_entrada_vigor
1	---	---	1	Domingo	15:00:00	18:00:00	4	2024-06-09
1	---	---	2	Segunda-Feira	09:00:00	13:00:00	9	2024-06-09
1	---	---	2	Segunda-Feira	14:00:00	18:00:00	9	2024-06-09
1	---	---	3	Terça-Feira	09:00:00	13:00:00	9	2024-06-09
1	---	---	3	Terça-Feira	14:00:00	18:00:00	9	2024-06-09
1	---	---	4	Quarta-Feira	09:00:00	13:00:00	9	2024-06-09
1	---	---	4	Quarta-Feira	14:00:00	18:00:00	9	2024-06-09
1	---	---	5	Quinta-Feira	09:00:00	13:00:00	9	2024-06-09
1	---	---	5	Quinta-Feira	14:00:00	18:00:00	9	2024-06-09
1	---	---	6	Sexta-Feira	09:00:00	13:00:00	8	2024-06-09
1	---	---	6	Sexta-Feira	14:00:00	17:00:00	8	2024-06-09

CREATE VIEW VIEW_informacao_instalacoes

Por fim nas views, fiz uma para dar as informações das instalações

```
CREATE VIEW VIEW_informacao_instalacoes AS
SELECT i.ID,
       i.designacao,
       ti.designacao AS des_tipo_instalacoes,
       i.morada,
       i.localizacao_gps,
       data_hora_primeira_abertura,
       data_hora_ultimo_encerramento,
       CASE
           WHEN data_hora_primeira_abertura IS NULL THEN "Sem planos para abrir"
           WHEN data_hora_primeira_abertura > NOW() THEN "Por abrir"
           WHEN data_hora_primeira_abertura <= NOW() AND data_hora_ultimo_encerramento IS NULL THEN "Em funcionamento"
           WHEN data_hora_primeira_abertura <= NOW() AND data_hora_ultimo_encerramento > NOW() THEN "Irá ser fechado"
           WHEN data_hora_primeira_abertura <= NOW() AND data_hora_ultimo_encerramento <= NOW() THEN "Encerrado"
       END AS estado
FROM TAB_instalacoes i
INNER JOIN TAB_tipo_instalacoes ti ON i.ID_tipo_instalacoes = ti.ID
;
```

E obtenho o seguinte

371 • **SELECT ***
372 FROM VIEW_informacao_instalacoes;

ID	designacao	des_tipo_instalacoes	morada	localizacao_gps	data_hora_primeira_abertura	data_hora_ultimo_encerramento	estado
1	Loja Central	Loja	Rua Principal, 123	BL001	2022-01-01 09:00:00	HULL	Em funcionamento
2	Loja Filial	Loja	Avenida Secundária, 456	BL002	2022-02-15 10:00:00	HULL	Em funcionamento
3	Escritório Administrativo	Escritório	Avenida das Empresas, 789	BL003	2022-01-20 09:30:00	HULL	Em funcionamento
4	Armazém Central	Armazém	Rua dos Armazéns, 1011	BL004	2022-01-01 08:00:00	HULL	Em funcionamento
5	Showroom	Showroom	Praça das Exibições, 1213	BL005	2022-04-05 11:00:00	HULL	Em funcionamento

Execução de procedimentos

Por fim, executo todos os procedimentos para adição de registos

```
call ExecutarInsercaoRegistrosMultiplosPessoas(1000);
call ExecutarInsercaoRegistrosMultiplosClientes(500);
call ExecutarInsercaoRegistrosMultiplosFuncionarios(50);
call ExecutarInsercaoRegistrosMultiplosCidados(10);
call ExecutarInsercaoRegistrosMultiplosEstrangeiros(10);
call ExecutarInsercaoRegistrosMultiplosContratosAtuaisFuncionarios();
call ExecutarInsercaoRegistrosMultiplasPromocoes();
call ExecutarInsercaoRegistrosMultiplasAbastecimentosStock(NULL);
```

O primeiro, call faz 1000 novos registo de pessoas com nomes, sobrenomes e NIFs aleatórios

O segundo call insere dos registo na tabela TAB_pessoa (se só tiver executado uma vez o call anterior será possível escolher 1000 clientes criados anteriormente) 500 clientes novos que não se repetam.

O terceiro call insere, dos mesmos registo que o segundo call vai buscar, 50 funcionários

O quarto call escolhe 10 pessoas e insere informações que o tornam cidadão

O quinto call escolhe outras 10 pessoas e insere informações que o tornam estrangeiro (passaporte)

O sexto e setimo call faz contratos e promove os funcionários. No caso das promoções ele usa a tal tabela auxiliar dos cargos para ver quais cargos um funcionário com certa profissão pode ter como cargo

Por fim o oitavo call faz 10 registos de abastecimento de stock aleatórios se for inserido um valor null. Caso tenha um valor ele irá fazer essa quantidade de registos no abastecimento de stock (TAB_stock_artigo)

SELECT

Para os selects eu fiz dois ficheiros.

1. O ficheiro “Trabalho final – Selects teste.sql” que tem um monte de selects dispersos que usei apenas para testar a minha base de dados
2. O ficheiro “Trabalho final – Selects.sql” que tem os selects que pretendo que o professor avalie e que irei aqui mostrar

Nomes aleatórios

Para mostrar as minhas funções a funcionar fiz um select que tem como objetivo mostrar um nome completo aleatório através das functions obter_nome_aleatorio e obter_sobrenome_aleatorio. Como observável abaixo

```
-- Nomes aleatorios
SELECT obter_nome_aleatorio(), obter_sobrenome_aleatorio();
```

Aniversariantes

Como os aniversariantes são uma parte fundamental das promoções e a Dona Elvira parece ser algum gosto em pessoas fiz uns 3 selects simples para dar os aniversários das pessoas em 3 períodos diferentes

No final mostrei as pessoas que fizeram anos através de um select a uma das logs que já foi referida anteriormente nos triggers

Aniversariantes hoje

Este select retorna as pessoas que fazem anos na data atual

```

7      -- Aniversariantes hoje
8 •  SELECT p.* 
9      FROM TAB_pessoa p INNER JOIN TAB_cliente c ON p.ID = c.ID_pessoa
10     WHERE proxima_data(data_nascimento) = CURDATE();
11
12     -- Aniversariantes amanhã

```

Result Grid				
ID	nome	sobrenome	NIF	data_nascimento

Que no caso para hoje é nenhum (os resultados podem ser diferentes dependo do dia e das pessoas introduzidas na tabela pessoa)

Aniversariantes amanhã

O mesmo que o anterior, mas em vez de dar os da data atual dá os de amanhã (data atual + 1)

```

12     -- Aniversariantes amanhã
13 •  SELECT p.* 
14      FROM TAB_pessoa p INNER JOIN TAB_cliente c ON p.ID = c.ID_pessoa
15     WHERE proxima_data(data_nascimento) = DATE_ADD(CURDATE(), INTERVAL 1 DAY);
16

```

Result Grid				
ID	nome	sobrenome	NIF	data_nascimento

Aniversariantes próximos 7 dias

Para se saber das pessoas que fazem aniversario nos próximos 7 dias (era para ser na semana, mas isso requeria mais umas quantas coisas que não tinha tempo para fazer) eu fiz o seguinte select que obteve já alguns resultados

```

17    -- Aniversariantes proximos 7 dias
18 •  SELECT p.* 
19      FROM TAB_pessoa p INNER JOIN TAB_cliente c ON p.ID = c.ID_pessoa
20      WHERE proxima_data(data_nascimento) >= CURDATE() AND proxima_data(data_nascimento) <= DATE_ADD(CURDATE(), INTERVAL 7 DAY)
21      ORDER BY proxima_data(data_nascimento) ASC;
22

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

ID	nome	sobrenome	NIF	data_nascimento
88	Ara Eduardo	Nogueira Ramos	228885027	1958-06-08
946	Juliana Freitas	Silva Martins	661788563	1959-06-09
493	Ara Roberto	Oliveira Silveira	277397642	1947-06-11
559	Liliana Rafael	Lima	472475681	1936-06-11
828	Rafael Freitas	Ferreira Moura	304331600	1974-06-11

Como pode ver aparece apenas pessoas que fazem anos daqui a 7 dias depois da data atual (sendo no caso do exemplo 05/06/2024 a data atual ou 2024-06-05 como o MySQL formata)

Pessoas que fizeram anos e fazem anos

Por fim fiz um simples select a minha tabela logs para mostrar a Dona Elvira os aniversariantes mais atuais e antigos (caso tenha se esquecido de algum)

```

23    -- Pessoas que fizeram anos e fazem anos
24 •  SELECT *
25      FROM logs_aniversariantes;
26

```

Result Grid | Filter Rows: | Edit: |

ID	ID_cliente	data_aniversario
1	102	2024-06-04
2	76	2024-06-04
3	335	2024-06-04
4	376	2024-06-04
NULL	NULL	NULL

Acontecimentos na tabela TAB_cliente

Como a Dona Elvira tem uma equipa de informática muito boa, e funcionários para verificarem se existe algum problema a nível de regtos das tabelas fiz um select que mostre todos os acontecimentos com a tabela TAB_cliente, quer seja adições, remoções ou atualizações de clientes. Para tal fiz o seguinte select ao logs_clientes

```

27 -- Acontecimentos na tabela TAB_cliente
28 • SELECT *
29     FROM logs_clientes
30     ORDER BY data_hora DESC, ID ASC;

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

ID	ID_cliente	data_hora	acao	detalhes
208	208	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 40 e chamada Francisco Clara Dias Silva
209	209	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 429 e chamada Camila Rodrigues Silva Melo
210	210	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 69 e chamada Paula Alexandre Gonçalves Santos
211	211	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 57 e chamada Juliana Rafael Moraes Teixeira
212	212	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 374 e chamada Paulo Fernanda Nogueira Moura
213	213	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 340 e chamada Isabela Pedro Vieira Pereira
214	214	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 595 e chamada Beatriz Lucas Moreira Santos
215	215	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 708 e chamada Mariana de Nunes Santos
216	216	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 854 e chamada Gabriel Henrique Costa Santos
217	217	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 235 e chamada Pedro Beatriz Vieira Rocha
218	218	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 487 e chamada Maria Clara Moreira Carvalho
219	219	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 292 e chamada Tiago Roberto Silva Teixeira
220	220	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 389 e chamada Liliana Freitas Pires Alves
221	221	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 534 e chamada Tiago Antonio Ribeiro Vieira
222	222	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 793 e chamada João Pedro Moreira Carvalho
223	223	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 155 e chamada Lorena Henrique Pereira
224	224	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 810 e chamada Rafael Pedro Ferreira Alves
225	225	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 331 e chamada João Freitas Oliveira Pereira
226	226	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 80 e chamada Aline Santos Lopes Azevedo
227	227	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 470 e chamada Lorena Sophia Teixeira Lima
228	228	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 550 e chamada Isabela Lucas Mendes Santos
229	229	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 592 e chamada Camila Luísa Santos Gomes
230	230	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 852 e chamada Ricardo Fernandes Ribeiro Pratas
231	231	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 975 e chamada Ana Eduardo Vieira
232	232	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 567 e chamada Gabriel Alexandre Ferreira Lima
233	233	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 647 e chamada Rafael de Santos
234	234	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 627 e chamada Beatriz Henrique Moraes Lima
235	235	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 576 e chamada Ana Roberto Dias Carvalho
236	236	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 119 e chamada José Fernanda Almeida Fernan...
237	237	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 362 e chamada Rafael Santos Melo
238	238	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 889 e chamada Fernanda Santos Correia
239	239	2024-06-02 20:16:34	INSERT	Foi inserido um novo cliente com o ID pessoa 748 e chamada Maria Clara Silva Casado

Contagem de acontecimentos na tabela TAB_cliente

Para deixar mais fácil saber se está a haver mais entrada de clientes que saída fiz um select que consegue contar quantos clientes foram inseridos, removidos ou atualizados

```

32 -- Contagem de acontecimentos na tabela TAB_cliente
33 • SELECT acao, COUNT(ID)
34     FROM logs_clientes
35     GROUP BY acao;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

acao	COUNT(ID)
INSERT	500

Como só tive selects ele retorna apenas isso

Feriados e datas

Como a Dona Elvira é esquecida quando aos feriados e os funcionários dela estão sempre a avisar eu fiz um select que retorna TODOS os feriados e as suas datas tal como o número de dias que ele dura

```
37      -- Feriados e datas
38 •  SELECT df.ID, f.designacao, df.data, df.n_dias
39      FROM TAB_dia_feriado df INNER JOIN TAB_feriado f ON df.ID_feriado = f.ID
40      ORDER BY df.data ASC;
41
```

The screenshot shows a database query results grid titled "Result Grid". The grid has four columns: "ID", "designacao", "data", and "n_dias". The data is sorted by "data" in ascending order. The results are as follows:

ID	designacao	data	n_dias
1	Ano novo	2024-01-01	1
4	Sexta-feira santa	2024-03-29	1
7	Páscoa	2024-03-31	1
11	Dia da liberdade	2024-04-25	1
10	Dia da liberdade	2024-04-25	1
12	Dia da liberdade	2024-04-25	1
13	Dia do trabalhador	2024-05-01	1
16	Corpo de Deus	2024-05-30	1
19	Dia de Portugal	2024-06-10	1
22	Dia de Santo António	2024-06-13	1
25	Dia de São João	2024-06-24	1
28	Dia da Assunção de ...	2024-08-15	1
31	Dia da Implantação ...	2024-10-05	1
34	Dia de todos os Santos	2024-11-01	1
37	Dia da Restauração ...	2024-12-01	1
40	Dia de Imaculada Co...	2024-12-08	1
43	Natal	2024-12-25	3
2	Ano novo	2025-01-01	1
5	Sexta-feira santa	2025-04-18	1
8	Páscoa	2025-04-20	1
14	Dia do trabalhador	2025-05-01	1
26	Dia de São João	2025-05-24	1
20	Dia de Portugal	2025-06-10	1
23	Dia de Santo António	2025-06-13	1
17	Corpo de Deus	2025-06-19	1
29	Dia da Assunção de ...	2025-08-15	1
32	Dia da Implantação ...	2025-10-05	1
35	Dia de todos os Santos	2025-11-01	1
38	Dia da Restauração ...	2025-12-01	1
41	Dia de Imaculada Co...	2025-12-08	1
44	Natal	2025-12-25	3

Profissões que tem salário base perto da média

Como a Dona Elvira tem bastante cuidado com custos ela pediu um select para dar os salários mais perto da média. Sendo este a resposta ao pedido dela

```

42      -- Profissões que tem salário base perto da média
43 •   SELECT *
44     FROM TAB_profissao
45     WHERE ABS(salario_base - (SELECT AVG(salario_base) FROM TAB_profissao)) = (
46         SELECT ABS(salario_base - (SELECT AVG(salario_base) FROM TAB_profissao))
47         FROM TAB_profissao
48         ORDER BY ABS(salario_base - (SELECT AVG(salario_base) FROM TAB_profissao)) ASC
49         LIMIT 1)
50 ;

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

ID	designacao	salario_base
3	Ourives	1800.00
8	Programador	1800.00
NULL	NULL	NULL

Detalhes funcionários

Como a Dona Elvira quer sempre saber o que se passa com os funcionários da sua empresa eu fiz o seguinte select que simplesmente chama a view que responde a essa pergunta, a view_informacoes_funcionario

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

ID_funcionario	ID_pessoa	nome	sobrenome	data_nascimento	idade	NIF	profissao	des_cargo	data_termino_contrato	dias_para_acabar_contrato	estado	total_salario
16	292	Tiago Roberto	Silva Teixeira	03-08-1961	62	368341135	Analista de Sistemas	Coordenador de Segurança da Informação	05-04-2025	304	Trabalhando	3600.00
17	877	Daniel Fernandes	Nunes Pereira	05-10-1970	53	320561281	Analista de Sistemas	Coordenador de Segurança da Informação	08-04-2025	307	Trabalhando	3600.00
18	51	Camila Sophia	Lima Pratas	21-06-1979	44	955012465	Técnico em Gemologia	Técnico de Gemologia	23-02-2025	263	Trabalhando	2200.00
19	948	Isabel Pedro	Santos Pratas	22-08-1945	78	466559414	Administrador de Redes	Administrador de Redes	13-02-2025	253	Trabalhando	2200.00
20	143	Lucena Cristina	Dias Silveira	17-02-1998	26	852093577	Vendedor	Gerente de Loja	08-05-2025	337	Trabalhando	1300.00
21	274	Ulisses Antônio	Gonçalves Silva	10-07-2003	20	105236351	Analista de Sistemas	Gerente de TI	12-05-2025	341	Trabalhando	4000.00
22	874	Gabriel Eduardo	Moraes Lima	27-03-2001	23	704746999	Programador	Programador	02-05-2025	331	Trabalhando	3000.00
23	3	Fernando Oliveira	Almeida Oliveira	25-04-1962	62	165732200	Consultor de Vendas	Coordenador de Vendas	03-03-2025	271	Trabalhando	1900.00
24	416	Daniel de	Noqueria Carvalho	21-10-1944	79	456983228	Técnico em Gemologia	Técnico de Gemologia	15-01-2025	224	Trabalhando	2200.00
25	494	Rafael Macedo	Gonçalves Carvalho	14-03-1968	56	222566528	Vendedor	Vendedor	19-11-2024	167	Trabalhando	800.00
26	206	Juliana Henrique	Viela Casado	09-07-2002	21	816494690	Programador	Programador Júnior	09-02-2025	249	Trabalhando	1800.00
27	974	Gabril Oliveira	Santos Gomes	07-04-1987	37	698104196	Analista de Sistemas	Coordenador de Segurança da Informação	04-11-2024	156	Trabalhando	3600.00
28	14	Liliane Eduardo	Mendes Pereira	22-06-1992	31	935334143	Analista de Sistemas	Coordenador de Segurança da Informação	17-05-2025	346	Trabalhando	3600.00
29	316	Daniel Fretas	Ferreira Teixeira	10-01-1958	66	304984043	Assistente Administrativo	Coordenador de Logística	16-01-2025	225	Trabalhando	1850.00
30	572	Paulo Antônio	Lima Carvalho	24-08-1967	56	522428983	Programador	Programador	25-11-2024	173	Trabalhando	3000.00
31	332	Aline Clara	Mendes Silva	20-06-1950	73	337063702	Vendedor	Supervisor de Vendas	06-03-2025	274	Trabalhando	1600.00
32	650	Lorena Lúcia	Ferreira Pereira	21-07-1981	42	926669737	Ourives	Ourives	28-02-2025	268	Trabalhando	1800.00
33	54	Gabriel Rafael	Morreira Santa	06-05-1991	33	038778297	Programador	Programador Sênior	04-11-2024	152	Trabalhando	3600.00
34	977	Fernanda Antônio	Silva Lima	22-10-1965	58	849891525	Analista de Sistemas	Coordenador de Segurança da Informação	28-02-2025	268	Trabalhando	3600.00
35	383	Gustavo Felipe	Olivera Silva	15-02-1995	29	968693822	Vendedor	Vendedor	28-03-2025	296	Trabalhando	800.00
36	822	Liliana de	Fernandes	17-03-1983	41	217316107	Administrador de Redes	Administrador de Redes	16-04-2025	315	Trabalhando	2200.00
37	543	Tiago Carolina	Gonçalves Alves	24-06-1966	57	596359510	Administrador de Redes	Administrador de Redes	24-04-2025	323	Trabalhando	2200.00
38	137	Aline Rafael	Texeira Azevedo	23-11-1994	69	211963267	Assistente Administrativo	Assistente Executivo	11-04-2025	310	Trabalhando	950.00
39	896	José Roberto	Ferreira Azevedo	05-03-1980	44	638012015	Ourives	Ourives	12-01-2025	221	Trabalhando	1800.00
40	449	Carlos Rodrigues	Ferreira Fernandes	24-02-1980	44	598532625	Ourives	Ourives	17-02-2025	257	Trabalhando	1800.00
41	424	Paulo Alexandre	Santos Sousa	25-01-1993	31	909445156	Assistente Administrativo	Chefe de Armazém	18-02-2025	258	Trabalhando	1750.00
42	173	Pedro Santos	Viera	30-07-1990	83	476438659	Assistente Administrativo	Assistente Executivo	10-11-2024	158	Trabalhando	950.00
43	999	Maria Fernandes	Mendes Pratas	21-08-1955	68	434607410	Técnico em Gemologia	Técnico de Gemologia	27-04-2025	326	Trabalhando	2200.00
44	339	Gustavo de	Pires Moura	24-05-1964	60	268536295	Supervisor de Produção	Chefe de Produção	16-05-2025	345	Trabalhando	2600.00
45	962	Maria Henrique	Teixeira Fernandes	02-05-1987	37	604218795	Assistente Administrativo	Gerente de Operações	13-11-2024	161	Trabalhando	2950.00
46	905	Liliana Felipe	Dias Silva	14-03-1986	38	857359894	Consultor de Vendas	Dirutor Financeiro	05-04-2025	304	Trabalhando	3900.00
47	643	Mariana Cristina	Lima Moura	26-01-1998	26	488339375	Analista de Sistemas	Coordenador de Segurança da Informação	18-01-2025	227	Trabalhando	3600.00

Ficheiro que faz tudo!

Primeiramente, quando digo tudo não quero dizer tudo mesmo.

Como todo bom marketing faço só a base de dados em si e os triggers, funções, procedimentos, views e execução dos procedimentos no fim no tal ficheiro “Trabalho final – Criação geral.sql”. Como se poderá ver abaixo

```

1 • DROP DATABASE IF EXISTS DB_Bijuteria_Dona_Elvira;
2
3 • CREATE DATABASE DB_Bijuteria_Dona_Elvira;
4
5 • USE DB_Bijuteria_Dona_Elvira;
6
7 • CREATE TABLE TAB_pessoa (
8     ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
9     nome NVARCHAR(255) NOT NULL,
10    sobrenome NVARCHAR (255) NOT NULL,
11    NIF VARCHAR(25) NOT NULL,
12    data_nascimento DATE NOT NULL DEFAULT (DATE_SUB(CURRENT_DATE(), INTERVAL 18 YEAR))
13 );
14
15 • CREATE TABLE TAB_morada (
16     ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
17     ID_pessoa INT NOT NULL,
18     morada NVARCHAR(255) NOT NULL,
19     data_hora_adicionado DATETIME NOT NULL DEFAULT (NOW()),
20     data_hora_removido DATETIME,
21     FOREIGN KEY (ID_pessoa) REFERENCES TAB_pessoa(ID)
22 );
23
24 • CREATE TABLE TAB_morada_preferencial (
25
26 ...
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97 • call ExecutarInsercaoRegistrosMultiplosPessoas(1000);
98 • call ExecutarInsercaoRegistrosMultiplosClientes(500);
99 • call ExecutarInsercaoRegistrosMultiplosFuncionarios(50);
2000 • call ExecutarInsercaoRegistrosMultiplosCidades(10);
2001 • call ExecutarInsercaoRegistrosMultiplosStrangeiros(10);
2002 • call ExecutarInsercaoRegistrosMultiplosContratosAtuaisFuncionarios();
2003 • call ExecutarInsercaoRegistrosMultiplosPromocoes();
2004 • call ExecutarInsercaoRegistrosMultiplosAbastecimentosStock(NULL);

```

A parte dos selects está a parte no ficheiro como anteriormente dito “Trabalho final – Selects.sql”.

Conclusão

O trabalho foi bastante intenso, pois tive de fazer diversos trabalhos ao mesmo tempo e estudar para diversas disciplinas em pouco tempo. Mas no fim do dia consegui fazer um trabalho bem elaborado e que acredito que satisfaria a Dona Elvira.

Muitas das coisas inseridas por último na base de dados e por sua vez no relatório é tudo graças ao tempo que o professor deu. Eu senti verdadeira stress quando disse para entregar até às 20:00 horas do dia 04/06. Mas bastante aliviado quando disse que afinal podia ser até às 18:00 horas do dia 05/06.

Notas adicionais

Podia ter feito mais selects pois esta estrutura que fiz foi pensada para responder a literalmente qualquer dúvida que a Dona Elvira possa ter ou vir a ter no seu negócio e assim poder expandir de forma mais segura e confiante.

Se a Dona Elvira gostar do resultado até ao momento que é bastante funcional e quiser mais funcionalidades eu aceitaria trabalhar para a Dona Elvira full time das 08:00-13:00 e das 14:00-17:00 todos os dias exceto sexta-feira e fins de semana.

Espero que a Dona elvira goste tanto do trabalho que pague com uma nota de 20 ou no mínimo 19,99 visto que deu verdadeiramente trabalho a estruturar, contruir e pensar nas views e selects