

# Franquia de Lojas de Penhores

## Base de Dados

**Licenciatura em Engenharia Informática e Computação**

**Docentes:**

Prof. Carla Lopes, [ctl@fe.up.pt](mailto:ctl@fe.up.pt)

Prof. Michel Ferreira, [mpferrei@fc.up.pt](mailto:mpferrei@fc.up.pt)

Prof. Mariana Dias, [mfdias@fe.up.pt](mailto:mfdias@fe.up.pt)

**Discentes:**

Carolina Mosqueiro, [up202303637@fe.up.pt](mailto:up202303637@fe.up.pt)

Mafalda Dias, [up202307365@fe.up.pt](mailto:up202307365@fe.up.pt)

Maria Catarina Moreira, [up202306603@fe.up.pt](mailto:up202306603@fe.up.pt)

Turma 5, Grupo 2

Porto, 30 de novembro de 2024

# Índice

Descrição do Tema .....	4
Modelo Conceptual Refinado .....	5
Esquema Relacional .....	6
Esquema Relacional Inicial .....	6
Integração de IA.....	6
Esquema Relacional Final .....	7
Dependências Funcionais e Formas Normais.....	8
Análise Inicial .....	9
Integração de IA.....	10
Análise Final .....	10
Utilização de IA no <i>create.sql</i> .....	12
Restrições de Validação .....	12
Resposta da IA .....	12
Prompt .....	12
Resposta da IA .....	12
Prompt .....	13
Resposta da IA .....	13
Decisão .....	13
Otimizações de Consulta .....	14
Prompt .....	14
Resposta da IA .....	14
Decisão .....	14
Campos Opcionais ou Nulos.....	14
Prompt .....	14
Resposta da IA .....	14
Decisão .....	15
Utilização de IA no <i>populate.sql</i> .....	16
Adição de Dados .....	16
Prompt .....	16
Integração de IA .....	16
Decisão .....	17
Irregularidades .....	18

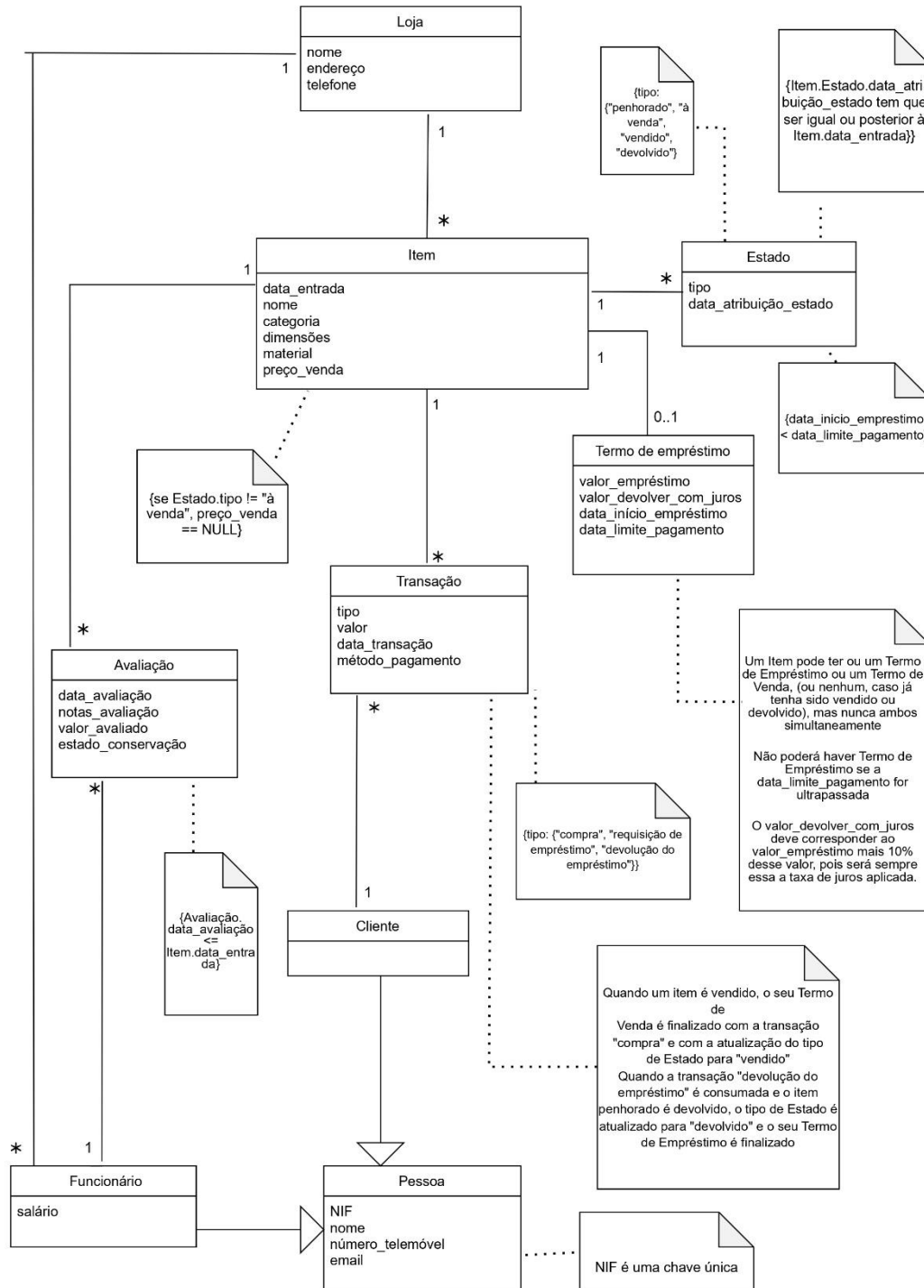
Prompt .....	18
Integração de IA .....	18
Prompt .....	18
Resposta da IA .....	18
Decisão .....	19
Transferências Problemáticas .....	19
Prompt .....	19
Resposta da IA .....	20
Decisão .....	20
Preços da Loja.....	20
Prompt .....	20
Resposta da IA .....	20
Decisão .....	21
Integração de IA – Análise Final .....	21

## **Descrição do Tema**

A Base de Dados expressa neste relatório pelos membros do grupo 502 pretende simular um sistema de franchising aplicado ao típico funcionamento de lojas de penhores. Como tal, todos os estabelecimentos associados a esta franquia estão estruturados de maneira similar e, no que diz respeito ao seu funcionamento, seguem um conjunto fixo de regras e padrões, nomeadamente em relação aos empréstimos que fornecem aos seus clientes, às taxas de juros associadas e aos tipos de artigos que podem penhorar.

No que diz respeito à definição básica de lojas de penhores, estas são estabelecimentos que não só vendem certos tipos de artigos – tais como eletrodomésticos, joias, antiguidades, entre outros –, mas também os adquirem de clientes que, em troca, recebem um empréstimo por cada um dos seus artigos – geralmente em dinheiro e que não lhes foi possível obter através de meios tradicionais. Estes artigos penhorados não ficam imediatamente na posse da loja, pois é concedido um intervalo de tempo aos clientes para devolverem o valor emprestado – agora com juros adicionados – de forma a recuperarem os seus itens. Caso não consigam fazê-lo até ao fim do prazo outrora estabelecido, estes passam a pertencer à loja e podem ser vendidos normalmente a entidades interessadas. Os detalhes inerentes aos itens são estabelecidos pelos funcionários das lojas.

# Modelo Conceptual Refinado



# Esquema Relacional

## Esquema Relacional Inicial

Loja (idLoja, nome, endereco, telefone)

Item (idItem, nome, dataEntrada, categoria, dimensoes, material, precoVenda, idLoja -> Loja)

Estado (idEstado, tipo, dataAtribuicaoEstado, idItem -> Item)

TermoEmprestimo (idEmprestimo, valorEmprestimo, valorDevolverComJuros, dataInicioEmprestimo, dataLimitePagamento, idItem -> Item)

Transacao (idItem -> Item, nifCliente -> Cliente, tipo, valor, dataTransacao, metodoPagamento)

Avaliacao (idItem -> Item, nifFuncionario -> Funcionario, dataAvaliacao, notasAvaliacao, valorAvaliado, estadoConservacao)

Pessoa (nif, nome, telefone, email)

Funcionario (nifFuncionario -> Pessoa, salario, idLoja -> Loja)

Cliente (nifCliente -> Pessoa)

## Integração de IA

Escolheu-se, entre vários modelos LLM, o Claude.ai (versão 3.5 Haiku). Esta decisão baseou-se na capacidade do Claude 3.5 Haiku de compreender e gerar respostas detalhadas sobre tópicos complexos, como sistemas de bases de dados e modelos UML.

Ao requerer-se auxílio na criação do esquema relacional correspondente ao modelo conceptual previamente apresentado, o Claude.ai solicitou o esclarecimento de algumas dúvidas que achou pertinentes para responder com maior precisão. Particularmente, para além daquelas que exigiam somente um “sim” ou “não” como resposta, perguntou qual estratégia de herança deveria usar em relação à classe “Pessoa” e às suas subclasses, “Funcionário” e “Cliente”. Pediu-se, então, entre as opções que ele expôs, que apresentasse o esquema relacional com a estratégia *Joined Table*, a qual vai de encontro ao que foi feito no esquema inicial.

O Claude.ai, portanto, expôs uma primeira versão. No entanto, esta apresentava dois erros óbvios: “Avaliação” não estava ligada de nenhuma forma a “Funcionário” – apesar de as avaliações dos itens serem realizadas pelos funcionários da loja – e “Termo de Empréstimo” possuía o NIF do cliente como chave estrangeira – embora um não esteja diretamente relacionado com o outro.

Confrontado, o Claude.ai corrigiu-se e, finalmente, foi possível obter o seguinte esquema relacional:

Loja (idLoja, nome, endereco, telefone)

Item (idItem, nome, dataEntrada, categoria, dimensoes, material, precoVenda, idLoja -> Loja)

Estado (idEstado, tipo, dataAtribuicaoEstado, idItem -> Item)

TermoEmprestimo (idEmprestimo, valorEmprestimo, valorDevolverComJuros, dataInicioEmprestimo, dataLimitePagamento, idItem -> Item)

Transacao (idTransacao, tipo, valor, dataTransacao, metodoPagamento, idItem -> Item, nifCliente -> Cliente)

Avaliacao (idAvaliacao, dataAvaliacao, notasAvaliacao, valorAvaliado, estadoConservacao, idItem -> Item, nifFuncionario -> Funcionario)

Pessoa (nif, nome, telefone, email)

Funcionario (nifFuncionario -> Pessoa, salario, idLoja -> Loja)

Cliente (nifCliente -> Pessoa)

## Esquema Relacional Final

Inicialmente, pretendia manter-se o esquema inicial e ignorar a adição de IDs que a inteligência artificial propôs, visto que as chaves estrangeiras nas tabelas “Transação” e “Avaliação”, teoricamente, serviriam perfeitamente de chave primária (composta). No entanto, esta escolha acabou por causar alguns problemas posteriormente, nomeadamente na tabela “Transação”, uma vez que um cliente pode requisitar um empréstimo e, depois,

devolvê-lo, ambas transações associadas a um mesmo item. Assim, adicionou-se um ID a “Transação” para que não houvesse *tuples* diferentes com a mesma chave primária e também a “Avaliação”, com o intuito de manter o esquema uniforme.

Loja (idLoja, nome, endereco, telefone)

Item (idItem, nome, dataEntrada, categoria, dimensoes, material, precoVenda, idLoja -> Loja)

Estado (idEstado, tipo, dataAtribuicaoEstado, idItem -> Item)

TermoEmprestimo (idEmprestimo, valorEmprestimo, valorDevolverComJuros, dataInicioEmprestimo, dataLimitePagamento, idItem -> Item)

Transacao (idTransacao, tipo, valor, dataTransacao, metodoPagamento, idItem -> Item, nifCliente -> Cliente)

Avaliacao (idAvaliacao, dataAvaliacao, notasAvaliacao, valorAvaliado, estadoConservacao, idItem -> Item, nifFuncionario -> Funcionario)

Pessoa (nif, nome, telefone, email)

Funcionario (nifFuncionario -> Pessoa, salario, idLoja -> Loja)

Cliente (nifCliente -> Pessoa)

## Dependências Funcionais e Formas Normais

### Loja:

idLoja -> nome, endereco, telefone

endereco -> idLoja, nome, telefone

telefone -> idLoja, nome, endereco

### Item:

idItem -> nome, dataEntrada, categoria, dimensoes, material, precoVenda, idLoja



**Estado:**

idEstado -> tipo, dataAtribuicaoEstado, idItem

**TermoEmprestimo:**

idEmprestimo -> valorEmprestimo, valorDevolverComJuros, dataInicioEmprestimo, dataLimitePagamento, idItem

**Transacao:**

idTransacao -> tipo, valor, dataTransacao, metodoPagamento, idItem, nifCliente

**Avaliacao:**

idAvaliacao -> dataAvaliacao, notasAvaliacao, valorAvaliado, estadoConservacao, idItem, nifFuncionario

**Pessoa:**

nif -> nome, telefone, email

telefone -> nif, nome, email

email -> nif, nome, telefone

**Funcionario:**

nifFuncionario -> salario, idLoja

**Análise Inicial**

Para uma dependência funcional não trivial estar na Forma Normal de Boyce-Codd, o seu lado esquerdo deve ser uma superchave. Visto que a chave (primária ou candidata) de uma relação é também superchave, pode afirmar-se que todas as relações apresentadas estão na Forma Normal de Boyce-Codd.

Uma dependência funcional não trivial está na Terceira Forma Normal quando o seu lado esquerdo é uma superchave ou o seu lado direito consiste somente em atributos primos. Uma vez que todas as dependências funcionais identificadas cumprem uma ou ambas as condições, é possível afirmar que todas as relações apresentadas estão na Terceira Forma Normal.

## Integração de IA

Solicitou-se ao Claude.ai, primeiramente, que apresentasse todas as dependências funcionais presentes no esquema relacional final. Ele apresentou aquelas cuja chave primária da relação se encontra no lado esquerdo – reforçando diversas vezes que o ID de cada uma das relações determina todos os outros atributos da mesma –, mas falhou em obter as dependências funcionais com chaves alternativas.

Ao ser questionado acerca da existência ou não de alguma violação da Terceira Forma Normal ou da Forma Normal de Boyce-Codd, o Claude.ai constatou que:

“Para estar em 3NF:

1. Deve estar em 2NF (todos os atributos não-chave devem depender totalmente da chave);
2. Não deve ter dependências transitivas de atributos não-chave.

Para estar em BCNF:

1. Para cada dependência funcional não-trivial  $X \rightarrow Y$ ,  $X$  deve ser uma superchave.

Todas as relações estão na Forma Normal de Boyce-Codd (BCNF) e, conseqüentemente, também estão na 3NF, pois:

1. Todas as DFs têm superchaves como determinantes;
2. Não existem dependências transitivas;
3. Não existem dependências parciais.”

## Análise Final

As dependências funcionais presentes em cada relação são, portanto, as seguintes:

**Loja:**

idLoja -> nome, endereco, telefone

endereco -> idLoja, nome, telefone

telefone -> idLoja, nome, endereco

**Item:**

idItem -> nome, dataEntrada, categoria, dimensoes, material, precoVenda, idLoja

**Estado:**

idEstado -> tipo, dataAtribuicaoEstado, idItem

**TermoEmprestimo:**

idEmprestimo -> valorEmprestimo, valorDevolverComJuros, dataInicioEmprestimo, dataLimitePagamento, idItem

**Transacao:**

idTransacao -> tipo, valor, dataTransacao, metodoPagamento, idItem, nifCliente

**Avaliacao:**

idAvaliacao -> dataAvaliacao, notasAvaliacao, valorAvaliado, estadoConservacao, idItem, nifFuncionario

**Pessoa:**

nif -> nome, telefone, email

telefone -> nif, nome, email

email -> nif, nome, telefone

**Funcionario:**

nifFuncionario -> salario, idLoja

Conclui-se que não existem violações da Terceira Forma Normal ou da Forma Normal de Boyce-Codd, visto que todos os atributos de cada uma das relações são determinados por uma superchave.

## Utilização de IA no *create.sql*

Com o intuito de melhorar a primeira versão da base de dados, presente no ficheiro *create1.sql*, perguntou-se ao Claude.ai que adições e mudanças ele faria, particularmente, em relação a restrições, referências e validações que seriam pertinentes. Este sugeriu que se incluíssem restrições de validação, otimizações de consulta e campos nulos/opcionais.

### Restrições de Validação

#### Resposta da IA

1. O campo “email” deveria incluir uma validação de formato adequado;
2. O campo “telefone” requer validações mais rigorosas para assegurar a consistência dos dados;
3. Algumas restrições podem ser definidas de forma mais específica para atender a regras de negócio.

#### Prompt

“Como posso melhorar a validação de campos como ‘email’ e ‘telefone’?”

#### Resposta da IA

Para melhorar a validação de “email” e “telefone” no *SQLite*, pode usar:

Para “email”:

```
CREATE TABLE Pessoa (  
    -- Outras colunas existentes  
    email TEXT NOT NULL CHECK (  
        email LIKE '%_@__%.__%' AND  
        email LIKE '%@%.%'  
    )  
);
```

Para “telefone” (Portugal):

```
CREATE TABLE Pessoa (  
    -- Outras colunas existentes  
    telefone TEXT NOT NULL CHECK (  
        length(telefone) = 9 AND  
        telefone REGEXP '^([1236][0-9]{7})$'  
    )  
);
```

### Prompt

“E que tipo de restrições posso adicionar?”

### Resposta da IA

Principalmente no que diz respeito aos NIFs, pode adicionar uma restrição de verificação (“CHECK”) para garantir que o “nif” seja um número válido, ou seja, que consiste em nove dígitos numéricos, por exemplo:

```
CHECK (nif REGEXP '^([0-9]{9})$')
```

### Decisão

Acataram-se as sugestões relacionadas a “email” e “telefone”, pois são, de facto, pertinentes para a inclusão de dados válidos na base de dados, bem como o “CHECK” para garantir que o “nif” tenha nove números, pelo mesmo motivo. No entanto, como o *SQLite* não utiliza “REGEXP” diretamente no “CHECK”, recorreu-se a uma alternativa para implementar a restrição pretendida:

```
CHECK (length(nif) = 9 AND nif GLOB '[0-9]*'),  
CHECK (length(telefone) = 9 AND telefone GLOB '[0-9]*')
```

## Otimizações de Consulta

### Prompt

“Que otimizações de consulta tinhas em mente?”

### Resposta da IA

Maioritariamente, sugeria que evitasse o uso de “SELECT\*”:

```
-- Mau (traz todas as colunas)
SELECT * FROM Item;

-- Melhor (só traz colunas necessárias)
SELECT idItem, nome, precoVenda FROM Item;
```

### Decisão

Em momento algum do código existia um “SELECT \*” ou semelhante. A sugestão da IA foi infundada e, portanto, rejeitada.

## Campos Opcionais ou Nulos

### Prompt

“E em relação aos Campos Nulos/Opcionais?”

### Resposta da IA

Sugeria:

1. O uso de “DEFAULT”:

```
-- Definir valores padrão
ALTER TABLE Item
MODIFY categoria TEXT DEFAULT 'outro';
```

```
-- Material com valores padrão
ALTER TABLE Item
MODIFY material TEXT DEFAULT 'não especificado';
```

## 2. O uso de restrições de validação:

```
CREATE TABLE Item (
    categoria TEXT CHECK (
        categoria IN ('eletrônica', 'joia', 'arte', 'outro') OR
        categoria IS NULL
    )
);
```

### Decisão

Aceitou-se a primeira sugestão da IA, mas, como a instrução “MODIFY” não é válida em SQL para definir valores padrão após a criação da tabela, utilizou-se uma notação alternativa que atende ao mesmo propósito:

```
ALTER TABLE Item
    ALTER COLUMN categoria SET DEFAULT 'Não Especificado',
    ALTER COLUMN material SET DEFAULT 'Não Especificado';
```

Aceitou-se, também, a sugestão do uso de restrições de validação em “categoria”:

```
categoria TEXT NOT NULL CHECK (
    categoria IN ('Joias e Acessórios',
        'Arte', 'Instrumentos Musicais',
        'Colecionáveis e Antiguidades',
        'Louça e Cerâmica',
        'Tecnologia e Informática',
        'Desportos e Lazer', 'Outros')
);
```

Assim, ambas as sugestões do Claude.ai foram aceites. Quanto ao uso de “DEFAULT”, embora, inicialmente, não se tenham ponderado as situações em que a categoria de um dado item fosse inválida, é prudente considerar a ocorrência de exceções. Além disso, a limitação do número de categorias, ideia inicialmente rejeitada para não limitar o intrínseco funcionamento das lojas de penhores, foi, depois, aceite, uma vez que, sem essa restrição, havia espaço para categorias redundantes e dificultaria futuras consultas. Curiosamente, a adição de um conjunto limitado de valores acabou por ser necessária durante o preenchimento do *populate.sql*, sendo, portanto, mais um motivo para implementar essa sugestão.

## Utilização de IA no *populate.sql*

Tencionava-se, agora, aumentar o número de entradas do ficheiro *populate1.sql*, adicionando-se mais itens, e, consequentemente, novos clientes e funcionários.

### Adição de Dados

#### Prompt

“Cria algumas possíveis novas entradas para o ficheiro *populate*, de acordo com as restrições da base de dados em *create2.sql*. Tem especial atenção às datas (não vendas um item antes de o avaliares, por exemplo) e às transferências (não podes ter uma venda e uma devolução de empréstimo para o mesmo item, uma venda antes da data máxima para pagamento do empréstimo associado, etc.). Tenta usar valores (preços e valores avaliados) realistas para os vários itens que criares.”

#### Integração de IA

O Claude.ai criou dados para três novos itens de várias categorias que, numa primeira análise, pareceram corretos. No entanto, após um maior escrutínio, tornaram-se evidentes várias lacunas na lógica da IA, mesmo com os avisos adicionais. Como exemplo disto tem-se a máquina fotográfica *vintage*, cujo empréstimo era devolvido, mas esta, posteriormente, vendida.



```

INSERT INTO Item (idItem, nome, dataEntrada,
                 categoria, dimensoes, material,
                 precoVenda, idLoja) VALUES
(14, 'Máquina Fotográfica Leica Vintage',
 '2024-12-10', 'Colecionáveis e Antiguidades',
 '15 x 10 x 8 cm', 'Metal e Couro', NULL, 5);

[...]

INSERT INTO Transacao (idTransacao, idItem,
                      nifCliente, tipo, valor, dataTransacao,
                      metodoPagamento) VALUES
(23, 14, 777888999, 'devolução de empréstimo',
 2640.00, '2025-01-10', 'dinheiro');

INSERT INTO Estado (idEstado, tipo,
                   dataAtribuicaoEstado, idItem) VALUES
(29, 'à venda', '2025-01-11', 14);

```

Outros detalhes remetem a “dataInicioEmprestimo” em “Termo de Empréstimo” e “dataAtribuicaoEstado” na tabela “Estado” (quando o estado é “penhorado”), que deveriam ser iguais, de acordo com o resto dos itens. No entanto, os *tuples* novos tinham, como data de início de empréstimo, o dia seguinte à data de atribuição de estado “penhorado”.

## Decisão

Mesmo modificando os dados fornecidos pelo Claude, seria, ainda, necessário alterar itens já existentes (cujo prazo de empréstimo teria acabado, vindas as datas fornecidas pela IA). Visto que os dados fornecidos não inspiraram muita confiança, decidiu-se não aplicar estas adições. No entanto, ao rever os dados criados pela IA, foram identificadas algumas irregularidades na tabela “Transação” do ficheiro *populate.sql* original, em particular nos “idCliente”.

## Irregularidades

### Prompt

“Analisa os dados da tabela ‘Transação’ no *populate.sql*; parecem existir algumas irregularidades quanto aos ‘idCliente’. Por exemplo, um cliente que devolveu um empréstimo de 330 € realizou uma compra de quase 4000 € passado um mês. Existe também um empréstimo a um cliente que, no passado, não devolveu o valor emprestado. Adicionalmente, após deixar passar o prazo para pagar o primeiro empréstimo, este mesmo cliente compra de volta o próprio item (agora 1200 € mais caro).

Determina o que faz ou não sentido e corrige o que achares necessário. Podes criar clientes (adicionando sempre os seus dados às tabelas ‘Pessoa’ e ‘Cliente’).”

### Integração de IA

A IA sugeriu que fossem criados novos clientes, no entanto, a solução em concreto era muito complexa, por isso descartou-se esta tentativa e decidiu-se alterar apenas dois “idCliente” (transferências 10 e 19) de modo que as transações fizessem mais sentido. De seguida, continuou-se com um *prompt* mais básico.

### Prompt

“Altere o ‘idCliente’ das transferências 10 e 19, no entanto, de momento, ainda existem duas transferências pouco usuais: as realizadas pelo cliente 666777888.”

### Resposta da IA

Para resolver esta inconsistência, sugiro criar-se um novo cliente com perfil financeiro mais adequado, por isso:

1. Criei um novo cliente (Ricardo Fernandes Oliveira) com NIF 654321987;
2. Transferi as duas transações problemáticas do cliente 666777888 para o novo cliente;
3. Mantive os valores e datas originais das transações.

```
-- Adicionar nova pessoa
INSERT INTO Pessoa (nif, nome, telefone, email) VALUES
(654321987, 'Ricardo Fernandes Oliveira', 919876543,
'ricardo.oliveira@email.com');

-- Adicionar novo cliente
INSERT INTO Cliente (nifCliente) VALUES
(654321987);

-- Atualizar transações problemáticas
UPDATE Transacao
SET nifCliente = '654321987'
WHERE nifCliente = '666777888' AND (
    (idItem = 8 AND tipo = 'compra') OR
    (idItem = 9 AND tipo = 'requisição de empréstimo')
);
```

## Decisão

A solução sugerida não resolveria o problema, já que substitui o “idCliente” em ambas as transações. Mesmo assim, adaptou-se a sugestão da criação do novo cliente, trocando o “idCliente” em apenas uma das transações. No entanto, existe ainda um par de transferências possivelmente problemáticas, as quais remetem ao cliente que compra um item que tinha anteriormente penhorado.

## Transferências Problemáticas

### Prompt

“Quanto às transferências associadas ao cliente 111222333 e item 4 (isqueiro antigo), achas que faz sentido o cliente comprar um item que penhorou anteriormente? Mesmo que seja pouco usual, valerá a pena manter esta ‘imperfeição’, já que oferece mais variedade na base de dados?”

## **Resposta da IA**

Na verdade, esta transação pode ser interessante e realista. Existem cenários plausíveis onde um cliente pode comprar de volta um item que anteriormente penhorou:

1. Valor sentimental: o isqueiro pode ter um significado especial para o cliente;
2. Arrependimento: o cliente pode ter-se arrependido de penhorar o item;
3. Melhoria financeira: o cliente pode estar em uma situação financeira melhor e querer recuperar o objeto.

A minha recomendação seria manter a transação como está. [...] A única observação que poderia ser feita é que o valor de compra (420 €) parece um pouco alto para um isqueiro antigo, considerando que a avaliação inicial foi de 300 €.

## **Decisão**

Achou-se a sugestão acertada e, por isso, baixou-se ligeiramente o valor da avaliação e de venda do isqueiro para 250 e 370 euros, respetivamente. Para terminar a interação com a IA, decidiu realizar-se outra análise geral, desta vez focada nos preços praticados pela loja.

## **Preços da Loja**

### **Prompt**

“No geral, o que achaste dos valores de revenda praticados pela loja? Demasiado altos? E os valores com juros a devolver?”

## **Resposta da IA**

Tenho algumas observações gerais:

1. Os valores de empréstimo são geralmente entre 30-50% do valor de avaliação;
2. A taxa de juros fixa de 10% parece razoável para um estabelecimento de penhores;
3. As revendas têm margens variáveis, mas não parecem abusivas.

Como tal, os valores parecem realistas e seguem uma lógica de negócio típica de uma casa de penhores:

1. Empréstimo menos do que o valor real do item;
2. Aplicar uma taxa de juros moderada;
3. Ter flexibilidade na revenda, dependendo do item e do mercado.

Não diria que os valores são demasiado altos. Representam uma estratégia de negócio comum em casas de penhores, onde se protegem financeiramente, emprestando menos que o valor real e tendo margem para revenda.

Realço somente, quanto a revenda, que alguns itens têm margens significativas. Exemplo do conjunto de tacos de golf:

1. Empréstimo: 2100 €;
2. Venda: 3790 € (quase 80% de acréscimo).

## **Decisão**

Concordou-se que o preço de alguns itens estaria demasiado alto e, consequentemente, baixou-se o valor de venda do conjunto de tacos de golf (cujo acréscimo era exagerado).

## **Integração de IA – Análise Final**

A constante utilização de inteligência artificial durante o projeto reforçou, mais uma vez, que esta possui limitações e que não deve ser confiada cegamente. De um modo geral, existem áreas em que esta é mais útil e outras em que é menos útil, pelo que cabe ao seu utilizador ter senso crítico o suficiente para discernir essas áreas e não aceitar qualquer resposta. A inteligência artificial, no presente momento, não consegue substituir totalmente o trabalho humano e, embora, de facto, seja capaz de poupar imenso tempo ao utilizador numa tarefa repetitiva, por ser necessária cautela na sua utilização, muitas vezes, perder-se-ia menos tempo a fazer essa tarefa do zero do que a confirmar que ela não cometeu nenhum erro espontaneamente durante a sua execução.

Um dos pontos fortes da inteligência artificial, como já foi mencionado, é a sua capacidade de executar tarefas repetitivas – trivial, computadores são bons nisso. Como tal, esperava-se que a IA fosse especialmente útil durante a população da base de dados, uma vez que é capaz de apresentar uma grande diversidade de valores em poucos segundos, algo que demoraria ao seu utilizador muito mais tempo para replicar.

Entretanto, fugindo às expectativas outrora estabelecidas, ela deixou muito a desejar no seu uso para aperfeiçoar o *populate.sql*, já que parecia ter dificuldades em manter os dados consistentes. Talvez tivesse sido mais útil numa base de dados menos desenvolvida, na qual houvesse menos probabilidade de o seu *output* “chocar” com informação pré-existente, ou até para iniciar a criação do *populate.sql*.

Ademais, possivelmente pelo quão abstratos modelos conceptuais eventualmente se tornam, a inteligência artificial desiluiu na passagem deste para um modelo relacional, apresentando alguns erros inconcebíveis, o que demonstra, mais uma vez, a sua falta de credibilidade para uma grande diversidade de temas.

Não obstante, mostrou-se particularmente conveniente no aperfeiçoamento do *create.sql*, dando algumas sugestões pertinentes e auxiliando com a sua sintaxe, outros dois pontos fortes da mesma.

Todos estes aspetos indicam, em suma, que a inteligência artificial, por não ter conseguido apresentar um único *output* inteiramente fiável, não era especialmente necessária em nenhuma parte deste projeto, sendo meramente uma ferramenta de auxílio para tarefas menos complexas.