POLITÉCNICO DE LEIRIA | ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Gestão Inteligente de Despesas de Casa

Mestrado em Ciência de Dados

Mickael Vieira da Silva

Leiria, março de 2024

POLITÉCNICO DE LEIRIA | ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Gestão Inteligente de Despesas de Casa

Mestrado em Ciência de Dados

Mickael Vieira da Silva

Projeto de Mestrado realizado sob a orientação da Doutora Eugénia Moreira Bernardino, Professora da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria e coorientação da Doutora Anabela Moreira Bernardino, Professora da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, março de 2024

Originalidade e Direitos de Autor

O presente relatório de estágio é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Agradecimentos

Quero agradecer a todas as pessoas que me apoiaram, direta ou indiretamente, ao longo de todo este percurso que foi o Mestrado. Agradeço ao Instituto Politécnico de Leiria por me ter acolhido na sua instituição e, através dos seus docentes e funcionários, me deram as condições para adquirir os conhecimentos técnicos que agora utilizo constantemente no meu percurso profissional. Um agradecimento especial para os docentes do departamento de Engenharia Eletrotécnica e Computadores, onde tirei a Licenciatura, e aos docentes do departamento de Engenharia Informática, nomeadamente os que lecionaram neste Mestrado.

Agradeço também à Doutora Anabela Moreira Bernardino e à Doutora Eugénia Moreira Bernardino, docentes nesta Instituição e orientadoras deste projeto, que tornaram possível a realização deste trabalho que já ambicionava há alguns anos.

Por último, mas não menos importante, quero agradecer aos meus pais que sempre me apoiaram, seja no meu percurso académico, seja em qualquer etapa da minha vida. Obrigado também à minha namorada, que sem ela não teria ingressado neste Mestrado e de quem não me faltou apoio.

À minha família

Resumo

O presente relatório é parte integrante do projeto realizado no âmbito da Unidade Curricular (UC) Dissertação/Projeto/Estágio do Mestrado em Ciência de Dados, lecionado na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria. O objetivo deste relatório é descrever todo o trabalho desenvolvido ao longo do projeto.

O projeto visa oferecer uma solução a um problema real através de técnicas utilizadas em *Business Intelligence* – uma das grandes áreas da Ciência de Dados que tem muito ênfase neste Mestrado. O objetivo deste projeto é responder a várias perguntas que geralmente ficam sem resposta aquando a elaboração de orçamentos familiares, como por exemplo: "*quanto dinheiro gastamos nas compras dos supermercados*?", "*em que categorias de produto se despendeu mais dinheiro?*", ou "*qual é a loja mais cara para os produtos comprados com mais frequência*?".

O projeto iniciou-se com a construção de uma Base de Dados (BD) SQL de raiz, o que incluiu a elaboração do dicionário de dados, do modelo conceptual (diagrama de entidade e relacionamento) e do modelo lógico. Esta BD foi de seguida preenchida com dados que foram extraídos a partir de faturas eletrónicas emitidas por várias lojas da cadeia de supermercados Continente, faturas essas datadas de 2019 a 2024. Foi necessário criar vários algoritmos com a linguagem de programação *Python* para conseguir recolher as faturas que se encontravam num servidor de e-mail, extrair dessas faturas os dados relevantes e introduzir os mesmos na BD, isto tudo de forma automática. Outras faturas referentes a outras lojas foram introduzidas manualmente através de *queries* SQL.

A partir da BD já preenchida, foi feito um estudo para a elaboração de um *Data Warehouse* (DW) igualmente de raiz, incluindo a conceção do modelo dimensional, criação das tabelas e automatização do processo ETL (*Extract*, *Transform* and *Load*).

Por fim, foram elaborados vários *dashboards*. Os gráficos e as tabelas desses *dashboards* são gerados diretamente a partir dos dados que foram carregados no DW, o que permite analisar sob diversas formas as despesas obtidas com base nas faturas disponíveis.

Palavras-chave: Base de dados, *Business Intelligence*, *Dashboard*, *Data Warehouse*, *Gestão de despesas*

Abstract

This report is an integral part of the master's degree project in Data Science, from the School of Technology and Management of the Polytechnique Institute of Leiria. The aim of this report is to describe all the work carried out during the project.

The project aims to provide a solution to a real problem through techniques used in Business Intelligence – one of the main branches of data science. The aim of this project is to answer various questions that are often left unanswered when drawing up household budgets, such as: "how much money do we spend on supermarket?", "which product categories do we spend the most money on?", or "which is the most expensive store for the products we buy most often?". The food retail market is very confusing due to the wide variety of stores and the manipulation of prices. The large number of promotions, loyalty cards and the marketing itself means that you can't really get a clear idea of how much a particular product actually costs or where you're actually spending the most money.

The project began by building an SQL database from scratch, which included drawing up the data dictionary, the logical model and the conceptual model. This database was then populated with data that was extracted from electronic invoices issued by several stores of the Continente supermarket chain, which are dated from 2019 to 2024. Several algorithms had to be created using Python programming language to collect the invoices from an e-mail server, extract the relevant data from these invoices and insert them into the database, all automatically. Other invoices from other stores were entered manually using SQL queries.

From the database already created, a study was carried out to create a data warehouse from scratch, including designing the dimensional model (star schema), creating the tables, and automating the ETL (Extract, Transform and Load) process.

Finally, the project includes the creation of various dashboards. The graphs in these dashboards are generated directly from the data that has been uploaded to the data warehouse, allowing us to analyse the expenses in various ways.

Keywords: Database, Business Intelligence, Dashboard, Data Warehouse, Expense Management

Índice

Originalidade e Direitos de Autor ... **iii**
Agradecimentos ... **iv**
Resumo ... **vi**
Abstract ... **vii**
Índice ... **viii**
Lista de Figuras ... **xi**
Lista de Tabelas ... **xiii**
Lista de Siglas e Acrónimos ... **xiv**
1. Introdução ... **1**
1.1. Objetivos ... **2**
1.2. Motivação ... **2**
1.3. Cronograma do projeto ... **3**
1.4. Estrutura do documento ... **6**
2. Enquadramento ... **7**
2.1. Conceitos básicos ... **7**
2.1.1. Business Intelligence ... 7
2.1.2. Processo ETL ... 8
2.1.3. Processo analítico *online* (OLAP) ... 9
2.1.4. Data Warehouse ... 11
2.1.5. Dashboard ... 13
2.2. Ferramentas e linguagens ... **13**
2.2.1. Linguagens de programação ... 13
2.2.2. SQL *Server Management Studio* ... 14
2.2.3. *Visual Studio 2022 & SQL Server Integration Services* ... 14
2.2.4. *Power BI* ... 15
2.2.5. *Draw IO* ... 15
2.3. Metodologia de desenvolvimento ... **15**
3. Análise inicial ... **18**
3.1. *User Stories* ... **18**
3.2. Funcionalidades do projeto ... **19**
3.3. Modelo de dados ... **21**

3.4. Modelo conceptual **21**
3.5. Tabelas da BD **22**
4. Desenvolvimento **24**
4.1. Criação da base de dados **24**
4.2. Download das faturas eletrónicas existentes no e-mail **25**
4.3. Recolha dos dados a partir das faturas eletrónicas **27**
4.3.1. Extração do texto 28
4.3.2. Recolha dos dados 30
4.3.2.1. Identificação da loja 31
4.3.2.2. Identificação do número da fatura e da data 31
4.3.2.3. Remoção das linhas inferiores 32
4.3.2.4. Remoção das linhas superiores 33
4.3.2.5. Remoção das linhas "desconto em cartão" 33
4.3.2.6. Extração dos dados das linhas de venda 34
4.3.2.7. Tratamento do valor total e do IVA 37
4.3.2.8. IVA 0% bens essenciais 39
4.4. Preenchimento automático das tabelas da base de dados **41**
4.5. Data Warehouse **46**
4.6. População do Data *Warehouse* – Processo ETL **49**
4.7. *Dashboard* **54**
4.7.1. Produtos mais comprados 57
4.7.2. Produtos mais dispendiosos 57
4.7.3. Preços unitários dos produtos 59
4.7.4. Preços médios dos produtos por loja 60
4.7.5. Despesa por categoria 61
4.7.6. Despesa por loja 61
4.7.7. Despesas totais 62
4.8. Análise crítica **63**
5. Conclusão **64**
5.1. Trabalho futuro **65**
6. Referências **67**
Anexos **72**
Anexo I. Relacionamentos entre as tabelas da BD **72**
Anexo II. Modelo lógico **74**

Anexo III. Dicionário de dados.. 75

Lista de Figuras

Figura 1 – Cronograma do projeto 5
Figura 2 – Diagrama com os *inputs* e *outputs* de um DW [11] 11
Figura 3 – Ilustração de um modelo dimensional em estrela [15] 13
Figura 4 – Diagrama típico da metodologia *Scrum* [28] 16
Figura 5 – Diagrama da arquitetura do projeto 20
Figura 6 – Diagrama Entidade Relacionamento (DER) 22
Figura 7 – Modelo da BD gerado através do *Reverse Engineering* do SSMS 25
Figura 8 – Código para importação das faturas eletrónicas a partir do e-mail: dados introduzidos 26
Figura 9 – Código para importação das faturas eletrónicas a partir do e-mail: download dos ficheiros 27
Figura 10 – *Output* obtido pelo algoritmo no fim de descarregar todos os ficheiros do e-mail 27
Figura 11 – Código criado para extração e apresentação do texto presente nas faturas eletrónicas 29
Figura 12 – Código criado para dividir a *string* resultante da extração de texto numa lista de *strings* 29
Figura 13 – Padrão encontrado nas faturas eletrónicas que permite ao algoritmo retirar os dados 30
Figura 14 – Código criado para identificação da loja 31
Figura 15 – Código criado para identificação do número da fatura e da data da compra 31
Figura 16 – Código criado para a criação da BD auxiliar dadosFatura 32
Figura 17 – Código criado para remoção das últimas linhas, irrelevantes para a BD 32
Figura 18 – Código criado para remoção das primeiras linhas, agora irrelevantes para a BD 33
Figura 19 – Código criado para remoção das linhas com eventuais descontos em cartão 34
Figura 20 – Código criado para a criação da BD temporária *lista_produtos* 34
Figura 21 – Código criado para o início da análise das linhas de venda das faturas eletrónicas 35
Figura 22 – Código criado para identificação da categoria 35
Figura 23 – Exemplo de uma linha venda de uma fatura eletrónica com quantidade 1 36
Figura 24 – Exemplo de uma linha venda de uma fatura eletrónica com quantidade diferente de 1 36
Figura 25 – Código criado para extrair a descrição, preço, subtotal e IVA, quando a quantidade é 1. 36
Figura 26 – Código alternativo para extração dos dados, quando a quantidade é diferente de 1. 36
Figura 27– Exemplo de uma linha venda de uma fatura eletrónica com desconto direto. 37
Figura 28 – Código criado para extração do desconto direto, caso existe. 37
Figura 29 – Código criado para o acréscimo dos dados extraídos na BD temporária *lista_produtos*. 37
Figura 30 – Código criado para o cálculo do total da fatura 38
Figura 31 – Código criado para transformar o campo extraído da taxa do IVA num campo numérico 38
Figura 32 – Tabela final típica obtida após extração dos dados de uma fatura eletrónica 39
Figura 33 – Código criado para gravação dos dados extraídos das faturas eletrónicas em ficheiros CSV 39
Figura 34 – Exemplo de uma fatura com isenção de IVA (com desconto e quantidade diferente de 1) 40
Figura 35 – Exemplo de uma fatura com isenção de IVA (sem desconto e quantidade igual a 1) 40
Figura 36 – Exemplo de uma fatura com isenção de IVA (com desconto e quantidade igual a 1) 40
Figura 37 – Código criado para identificar os produtos com isenção temporária de IVA 40
Figura 38 – Código alterado para poder identificar a letra ‘D’ como sendo IVA a 0% 41

Figura 39 – Código criado para importar as bibliotecas e os ficheiros CSV 42
Figura 40 – Código criado para efetuar ligação do *script* à DB SQL 42
Figura 41 – Código criado para a proteção da BD contra a introdução de faturas duplicadas 42
Figura 42 – Código criado para preenchimento da tabela *invoice* na BD SQL 43
Figura 43 – Código criado para inserção da categoria e do produto na BD SQL 44
Figura 44 – Código criado para inserção dos dados referentes às linhas de venda na BD SQL 44
Figura 45 – Output obtido pelo algoritmo no fim de introduzir os dados das linhas de venda na BD SQL 44
Figura 46 – Verificação da criação da fatura exemplo na tabela *invoice* da BD SQL 45
Figura 47 – Verificação da criação das linhas de venda na tabela *invoice_line* da BD SQL 45
Figura 48 – Amostra da tabela *county* da BD, obtido diretamente a partir do SSMS 46
Figura 49 – *Query* SQL para criar manualmente a loja "CBD Marrazes" na tabela *store* da BD 46
Figura 50 – Modelo do DW gerado através do *Reverse Engineering* do SSMS 49
Figura 51 – *Control Flow* da solução criada em *Visual Studio* 50
Figura 52 – Blocos criados no *Data Flow time* para o seu processo ETL 51
Figura 53 – Blocos criados no *Data Flow time, item* e *loja* para o seu processo ETL 52
Figura 54 – Blocos criados no *Data Flow fact* para o seu processo ETL 52
Figura 55 – Construção da *query* para a extração dos dados da BD original para a tabela de factos 53
Figura 56 – *Query* gerada automaticamente pelo construtor do bloco *OLE DB Source* 53
Figura 57 – *Query* para determinar qual a última linha de venda introduzida no DW 54
Figura 58 – *Output* obtido após uma execução do processo ETL 54
Figura 59 – Modelo de dados gerados a partir do *Power BI Desktop* 55
Figura 60 – Página 1 do *dashboard*: Produtos mais comprados 57
Figura 61 – Página 2 do *dashboard*: Produtos mais dispendiosos 58
Figura 62 – Gráfico temporal do produto que gerou a maior despesa 58
Figura 63 – Página 3 do *dashboard*: Preços dos produtos 59
Figura 64 – Gráfico temporal com a evolução do preço médio do produto mais adquirido 59
Figura 65 – Página 4 do *dashboard*: Preços dos produtos por loja 60
Figura 66 – Gráfico temporal com evolução do preço médio, em cada loja, do produto mais adquirido 60
Figura 67 – Página 5 do *dashboard*: Despesa por categoria 61
Figura 68 – Página 6 do *dashboard*: Despesa por loja 62
Figura 69 – Página 7 do *dashboard*: Despesas totais 62
Figura 70 – Diagrama relacionamento entre as tabelas *invoice* e *invoice_line* 72
Figura 71 – Diagrama relacionamento entre as tabelas *invoice_line* e *item* 72
Figura 72 – Diagrama relacionamento entre as tabelas *item* e *category* 73
Figura 73 – Diagrama relacionamento entre as tabelas *invoice* e *store* 73
Figura 74 – Diagrama relacionamento entre as tabelas *store* e *county* 73
Figura 75 – Modelo lógico da BD 74

Lista de Tabelas

Tabela 1 – Bus Matrix – Identificação da tabela de factos e dimensões 47
Tabela 2 – Descrição dos atributos da dimensão *store* 47
Tabela 3 – Descrição dos atributos da dimensão *item* 47
Tabela 4 – Descrição dos atributos da dimensão *county* 47
Tabela 5 – Descrição dos atributos da dimensão *time* 48
Tabela 6 – Descrição dos atributos da tabela de factos 49
Tabela 7 – Tabela *Invoice* 75
Tabela 8 – Tabela *store* 76
Tabela 9 – Tabela *county* 76
Tabela 10 – Tabela *item* 76
Tabela 11 – Tabela *invoice_line* 77
Tabela 12 – Tabela *category* 77

Lista de Siglas e Acrónimos

API	*Application Programming Interface*
BD	Base de dados
BI	*Business Intelligence*
CRM	*Customer Relationship Management*
CSV	*Comma-Separated Values*
DAX	*Data Analysis Expressions*
DER	Diagrama Entidade e Relacionamento
DW	*Data Warehouse*
ERP	*Enterprise Resource Planning*
IA	Inteligência Artificial
IDE	*Integrated Development Environment*
IMAP	*Internet Message Access Protocol*
IPYNB	*Interactive Python Notebook*
IVA	Imposto sobre Valor Acrescentado
KPI	*Key Performance Indicator*
ML	*Machine Learning*
OLAP	Processo analítico *online*
OLTP	Processamento de Transações em Tempo Real
PK	*Primary Key* – Chave Primária
RegEx	Expressões Regulares
SQL	*Structured Query Language*
SSMS	*SQL Server Management Studio*
SSO	*Single Sign-On*
TSV	*Tab-Separated Values*
UC	Unidade Curricular
US	*User Storie*

1. Introdução

O presente relatório descreve o trabalho desenvolvido ao longo do projeto no âmbito da Unidade Curricular (UC) Dissertação/Projeto/Estágio do Mestrado em Ciência de Dados, lecionado na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, nomeadamente, a metodologia de desenvolvimento, a descrição das tarefas desenvolvidas no projeto e as respetivas tecnologias utilizadas. Trata-se de um projeto desenvolvido ao longo de 13 meses (setembro 2022 a setembro 2023), no âmbito do ano letivo 2022/2023.

Trata-se de um projeto principalmente focado em *Business Intelligence* (BI), uma das grandes áreas da Ciência de Dados. BI é um conjunto de processos e tecnologias que permite a análise de dados de forma a fornecer informações úteis que ajudam executivos, gerentes e funcionários a tomar decisões de negócios informadas. Como parte do processo de BI, as organizações recolhem dados de sistemas internos e fontes externas, preparam os mesmos para análise e criam visualizações de dados tais como *dashboards* ou relatórios para disponibilizar os resultados de análise aos diferentes *Stakeholders* para tomada de decisões operacionais e planeamento estratégico [1].

Com a quantidade de dados disponível atualmente e com a tecnologia existente que agora permite recolher e armazenar grandes quantidades de dados, qualquer empresa ou organização que pretende um crescimento grande e a longo prazo deve adotar uma estratégia baseada em dados (*data-driven decision-making*). O objetivo do BI é melhorar as operações e as estratégias de uma organização através da utilização de dados relevantes. As empresas que utilizam eficazmente ferramentas e técnicas de BI podem traduzir os dados recolhidos em informações valiosas sobre os seus processos e estratégias empresariais. Estas informações podem ser utilizadas para tomar melhores decisões de negócio que aumentem a produtividade e as receitas, conduzindo a um crescimento acelerado do negócio e a lucros mais elevados. Sem o BI, as organizações não podem tirar partido da tomada de decisões baseada em dados. Em vez disso, os executivos e os trabalhadores são obrigados a basear as decisões empresariais noutros fatores, como o conhecimento acumulado, as experiências anteriores, a intuição e o instinto. Embora estes métodos possam resultar em boas decisões, também estão propícias a erros que impedem o crescimento da empresa [1].

"*Se olhar para a sua organização e vir equipas a tomar decisões sem esforço porque estão a utilizar os dados, então já percebeu o verdadeiro valor dos seus dados*." [2]

1.1. Objetivos

O presente relatório é então parte integrante de um projeto cujos principais objetivos são os seguintes:

- Complementar as componentes de formação académica, através da implementação de soluções tecnológicas que permitam a resolução de problemas reais, proporcionando uma formação prática na área de BI;
- Aplicação prática dos conhecimentos e de competências teórico-práticas adquiridos ao longo do curso;
- Ensaiar práticas ajustadas a problemas reais;
- Criar e desenvolver hábitos de trabalho e sentido de responsabilidade;
- Criação de um sistema para armazenamento de dados relativo a despesas obtidas com a aquisição de bens essenciais;
- Arranjar forma de conseguir descarregar automaticamente todas as faturas eletrónicas existentes numa pasta do correio eletrónico;
- Realizar a extração automática dos dados relevantes a partir das faturas eletrónicas emitidas pelas lojas da cadeia de supermercados Continente;
- Execução de um processo *Extract, Transform and Load* (ETL) de forma a preencher automaticamente um *Data Warehouse* (DW);
- Geração de gráficos e de tabelas a partir dos dados exportados para o DW, que permitam analisar os dados extraídos das faturas eletróncias.

1.2. Motivação

O motivo que levou ao desenvolvimento deste projeto prende-se pelo facto de não se conseguir encontrar um sistema que permita resumir as despesas realizadas em supermercados para o dia a dia. Nos dias de hoje, em Portugal e na Europa, o aumento da

inflação obriga a um controlo mais minucioso das despesas da casa e uma boa parte do orçamento familiar é gasto nos supermercados para a compra de bens essenciais. No entanto, o mercado do retalho alimentar tornou-se num mundo muito complexo. Entre o minimercado da aldeia e o hipermercado do centro comercial, entre promoções e cartões de fidelização e entre variedade de produtos e variedade de qualidade, torna-se praticamente impossível saber de forma pormenorizada as despesas obtidas com as compras efetuadas para a casa e identificar formas de diminuir essa despesa. Existe a necessidade de poder responder a questões tais como "Qual é a loja que vende mais barato os produtos que mais uso?", "Em que categoria de produtos gasto mais dinheiro?", ou "A que preço comprei este produto da última vez?".

Com este projeto pretende-se iniciar uma aplicação que permita auxiliar na elaboração de orçamentos familiares e ajudar a perceber onde se está a gastar mais dinheiro.

1.3. Cronograma do projeto

O projeto teve início em setembro de 2022 e ficou concluído em setembro de 2023. A data inicial prevista para a entrega do projeto era setembro de 2023, mas devido a impedimentos profissionais, foi necessário solicitar uma prorrogação do prazo de entrega (31 de março de 2024), uma vez que o processo de escrita do relatório ainda estava num estado muito inicial. A Figura 1 representa o cronograma final do projeto, que se foi atualizando com tarefas não previstas inicialmente e com essa prorrogação de prazo.

O primeiro mês de trabalho foi praticamente dedicado a pesquisas sobre as necessidades do projeto consoante os objetivos a cumprir, nomeadamente conseguir responder a perguntas relacionadas com as despesas realizadas com os supermercados. Essas perguntas foram convertidas em *User Stories* (US) nesta etapa inicial do projeto. De seguida foi desenvolvido o modelo de dados, o diagrama lógico e o Diagrama Entidade e Relacionamento (DER), que permitiu criar as tabelas da Base de Dados (BD).

A fase mais demorada do projeto foi o desenvolvimento de algoritmos em *Python*. Esses algoritmos permitem obter as faturas eletrónicas em formato PDF armazenadas numa conta de e-mail. Posteriormente esses documentos são lidos pelo algoritmo de forma a extrair os dados relevantes e armazenar os mesmos diretamente em tabelas previamente criadas. Para armazenar os dados das faturas foi utilizada a linguagem *Structured Query Language* (SQL).

Após a BD ter sido preenchida com dados, iniciou-se a fase da conceção do DW. Começou-se com o desenvolvimento da *Bus Matrix* e o estudo sobre as tabelas de facto e as dimensões que se podem criar de forma a conseguir responder às US. Foi de seguida necessário criar uma nova BD para o DW, com as tabelas de facto e as dimensões que resultaram desse estudo. Por fim, foi feita a população do DW de forma automática através de um processo ETL, com o auxílio do *Visual Studio Integration Services*.

A última fase do projeto foi criar os *dashboards* a partir dos quais é possível obter respostas às perguntas estabelecidas nas US.

Ao longo de todo o projeto foram realizadas reuniões periódicas com as orientadoras e o presente relatório foi-se desenvolvendo à medida da evolução do projeto.

Em abril de 2023, uma medida do Governo português permitiu que uma seleção de produtos essenciais fosse temporariamente vendida com uma taxa de Imposto sobre Valor Acrescentado (IVA) de 0%, como forma de tentar travar a inflação existente nessa altura [3]. Essa medida originou algumas alterações nas faturas do Continente que obrigaram a uma atualização do algoritmo desenvolvido para a leitura dos dados dessas faturas eletrónicas. Essa atualização foi realizada em setembro de 2023.

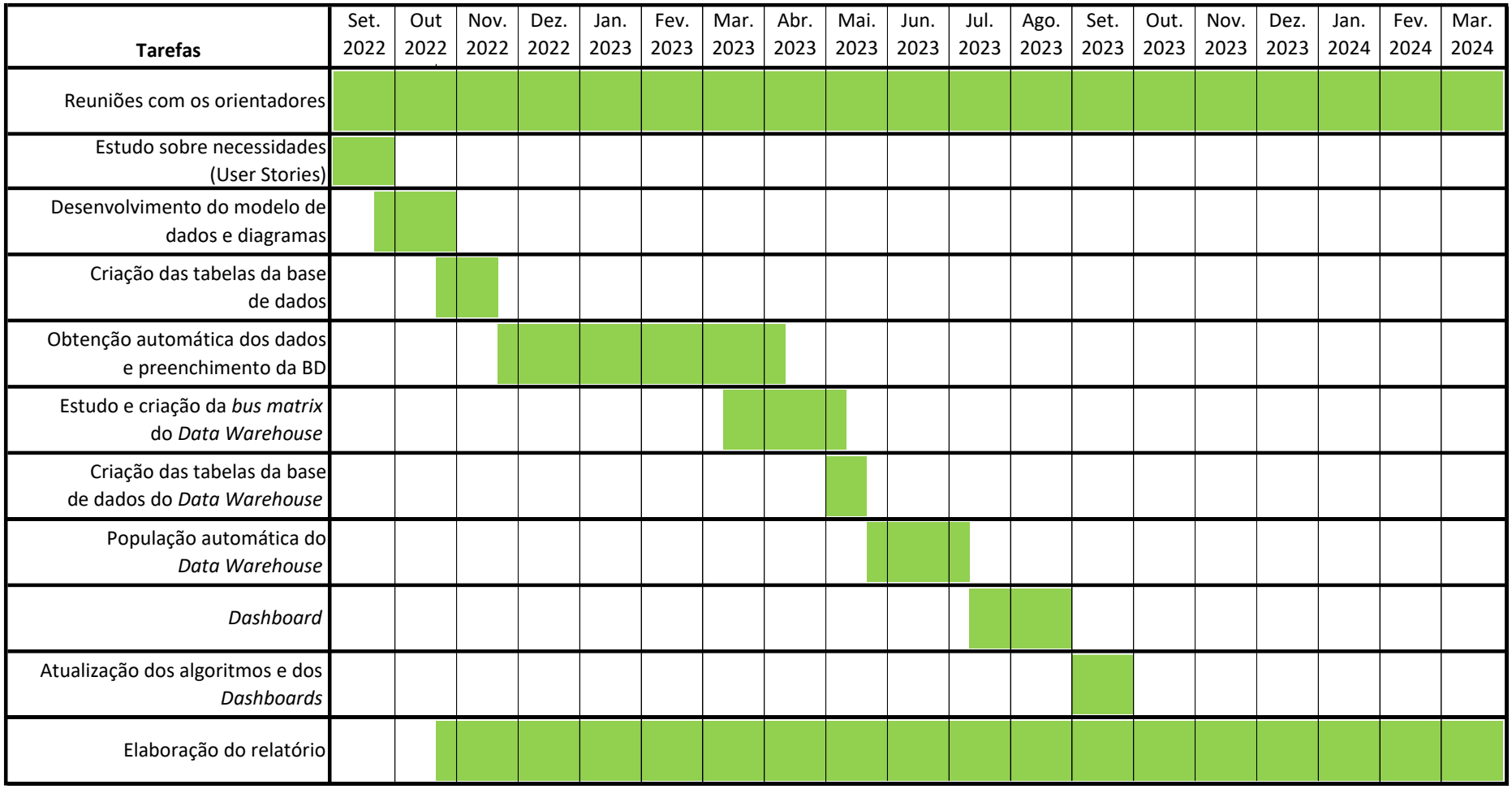


Figura 1 – Cronograma do projeto

1.4. Estrutura do documento

O presente relatório encontra-se estruturado em 6 capítulos, sendo que no capítulo 2 (Enquadramento) é feita uma introdução aos conceitos teóricos que foram necessários para o desenvolvimento do projeto. Neste capítulo também é abordada a metodologia utilizada para desenvolver este projeto.

No capítulo 3 (Análise inicial) é feita uma contextualização do projeto, expondo as perguntas às quais se procuram respostas através de US, bem como as funcionalidades providenciadas pelo projeto. Além disso, este capítulo também inclui a descrição da BD, com a apresentação do modelo de dados.

O capítulo 4 (Desenvolvimento) apresenta a forma de como cada etapa do projeto foi desenvolvida, incluindo os testes efetuados e os resultados obtidos. Nesse capítulo encontra-se também a apresentação dos gráficos e das tabelas obtidas através dos diversos *dashboards* criados, bem como a respetiva análise.

Por fim, o capítulo 5 (Conclusão) apresenta as conclusões finais bem como algumas propostas de melhorias para trabalho futuro.

2. Enquadramento

Neste capítulo é feita uma contextualização aos vários tópicos do projeto, começando por uma breve abordagem ao longo da primeira secção sobre os conceitos teóricos utilizados, desde o BI, Processo ETL, Processo analítico *online* (OLAP), DW e também sobre *Dashboards* (secção 2.1). De seguida são apresentadas as ferramentas utilizadas fazendo uma breve descrição das mesmas e em que etapa do projeto foram utilizadas (secção 2.2). Por fim, a última secção refere-se à metodologia de desenvolvimento utilizada ao longo do projeto (secção 2.3).

2.1. Conceitos básicos

Ao longo da presente secção é feita uma introdução genérica aos diversos conceitos teóricos que foram colocados em prática neste projeto, nomeadamente, BI (secção 2.1.1), Processo ETL (secção 2.1.2), OLAP (secção 2.1.3), DW (secção 2.1.4) e *Dashboard* (secção 2.1.5).

2.1.1. Business Intelligence

No cenário empresarial atual onde a competitividade é cada vez maior, as organizações enfrentam o desafio de transformar grandes volumes de dados em informações úteis para a tomada de decisões estratégicas. Nesse contexto, o BI surge como uma abordagem fundamental que permite tirar o maior valor possível dos dados.

BI é um conjunto de processos, tecnologias e ferramentas que permitem às organizações recolher, organizar, analisar e interpretar dados para orientar decisões estratégicas e operacionais. Esses dados podem ser provenientes tanto de fontes internas como externas e abrangem as mais diversas áreas. Neste sentido, o BI permite a criação de visualizações, *dashboards* e indicadores de acordo com as necessidades do utilizador, tudo de forma rápida graças ao seu sistema de armazenamento por DW [4].

O BI torna-se assim numa ferramenta valiosa e imprescindível para auxiliar as empresas em tomar decisões mais informadas, fornecendo dados atuais e históricos dentro do contexto empresarial. As técnicas de análise e as ferramentas existentes possibilitam comparações de desempenho e concorrência, ajudando a empresa a operar de maneira mais eficiente. Além disso, o BI facilita a identificação de tendências de mercado de forma a impulsionar as vendas e os respetivos lucros. Quando aplicados corretamente, os dados certos podem

contribuir para diversas áreas do negócio, desde a conformidade até ao recrutamento. O BI engloba tanto a análise de dados, como a análise empresarial, integrando-as como partes essenciais de um processo completo. Enquanto que os cientistas de dados exploram aspetos específicos dos dados através de técnicas avançadas de estatística e análise preditiva para identificar padrões em dados históricos para assim poder prever tendências futuras, o BI auxilia os utilizadores na extração de conclusões por meio da análise de dados, análise essa que procura perceber a razão pelos quais certos eventos ocorreram e o que é esperado para o futuro [5].

O BI foi concebido para responder a questões específicas e fornecer análises rápidas para orientar a gestão e as tomadas de decisões. No entanto, as empresas podem utilizar os processos de análise para formular questões cada vez mais refinadas, otimizando o processo o tornando-o iterativo. A análise empresarial não deve ser vista como um processo linear: responder a uma pergunta provavelmente levará ao surgimento de novas questões complementares. Por isso, o processo deve ser concebido como um ciclo contínuo de acesso, descoberta, exploração e partilha de informações baseadas em dados [5].

Embora grandes empresas tenham frequentemente equipas dedicadas e sistemas sofisticados de BI, muitas das técnicas e abordagens de BI podem ser adaptadas para uso por particulares, mesmo em escala menor, nomeadamente para projetos pessoais, como é o caso deste projeto. Existem várias ferramentas e plataformas disponíveis que podem ajudar na recolha, análise e visualização de dados, que permitem ajudar a tomar decisões mais informadas.

2.1.2. Processo ETL

Um dos problemas em comum que as organizações enfrentam cada vez mais é a necessidade de extrair dados de várias fontes e em diversos formatos, e transferir os mesmos para outros locais. Nestas situações, o destino pode não ser do mesmo tipo de armazenamento que a origem – o formato pode ser diferente e pode haver necessidade de transformar ou limpar os dados, antes de carregar os mesmos para o local de destino [6].

O processo ETL consiste num sistema de integração de dados que funciona em 3 etapas: extração, transformação e carregamento, sendo utilizado para combinar dados de diversas fontes. Trata-se de um conjunto de ferramentas informáticas cuja função é a extração de dados de diversas origens e formatos, transformação dos dados conforme regras de negócios

e, por fim, carregamento dos dados para um local destino que pode ser, por exemplo, um DW [7].

O processo ETL resume-se então às seguintes 3 fases:

- *Extract*: a extração é a etapa inicial do processo ETL. Os dados são recolhidos das suas fontes originais e colocados de forma temporária numa zona do sistema denominada de *staging area*, onde serão posteriormente trabalhados. As fontes de dados podem ser, entre outras, servidores SQL, sistemas de *Customer Relationship Management* (CRM) ou *Enterprise Resource Planning* (ERP), ficheiros do tipo *flat file* (por exemplo: *Comma-Separated Values* (CSV) ou *Tab-Separated Values* (TSV)), e-mail ou páginas *Web*;

- *Transform*: Nesta fase, os dados extraídos podem ou não ser alvos de transformação de forma a serem compatíveis com a utilização e o local a que se destinam. Essa transformação pode ser a nível de formatação, limpeza, cálculos, filtragem, encriptação, aplicação de regras de negócio, entre outros;

- *Load*: Por fim, é nesta fase do carregamento que os dados já trabalhados e limpos são importados para a sua BD de destino. É a partir dessa BD que os dados são utilizados conforme a respetiva necessidade [8].

As soluções ETL melhoram a qualidade dos dados ao efetuar a limpeza dos mesmos no momento da transformação. Além disso, este processo permite realizar a extração de apenas parte dos dados, nomeadamente os que ainda não foram extraídos em processos de extração anteriores. Por outras palavras, o processo ETL pode ser desenhado de forma a que sejam extraídos, transformados e carregados somente dados novos, fazendo assim uma simples atualização da BD de destino. Isso torna o processo ETL numa solução mais rápida, já que não precisa de trabalhar todos os dados sempre que são adicionados novos dados nas fontes de origem [9].

2.1.3. Processo analítico *online* (OLAP)

Um sistema OLAP é uma ferramenta crucial para a análise de grandes quantidades de dados no seio das organizações. Ao contrário das BD com Processamento de Transações em Tempo Real (OLTP), as BD OLAP são otimizadas para efetuar consultas e criar relatórios e gráficos dinâmicos. Os dados são provenientes de sistemas OLTP como, por exemplo, um

DW e são organizados em estruturas hierárquicas e multidimensionais chamadas de cubos, em vez de tabelas.

Num servidor OLAP é possível efetuar cálculos que resumem os dados, como somas ou médias, permitindo assim uma obtenção mais rápida dos dados devido à redução da quantidade de informação a transferir. Uma BD OLAP possui essencialmente os seguintes componentes:

- **Medida**: É um conjunto de valores, normalmente numéricos, que está armazenado numa coluna na tabela de factos. São os valores que são alvo de análise e que é necessário agrupar por dimensões (agrupar, por exemplo as vendas por loja, por cliente, por categoria ou por espaços temporais) [10];

- **Campo calculado**: muitas vezes, é necessário apresentar informações que não são retiradas diretamente da BD, mas sim calculadas com base nos campos existentes nela. Para tal, cria-se um novo campo calculado cujo valor é calculado com auxílio de uma expressão. Os valores de um novo campo calculado podem derivar dos valores de outros campos calculados. Por exemplo, pode haver a necessidade de criar um campo calculado "lucro de venda", através da subtração do valor do campo "custo de aquisição" ao valor do campo "valor da venda" [10];

- **Dimensão**: É um conjunto de atributos que descreve os aspetos de interesse de uma empresa nos quais se pretende efetuar a análise dos dados. As dimensões são essenciais para entender e interpretar os dados, pois representam as diferentes perspetivas pelas quais as informações podem ser analisadas. Por exemplo, num DW de vendas, as dimensões podem incluir tempo, produto, cliente e localização [10];

- **Hierarquia**: Trata-se de uma estrutura organizacional que permite ao utilizador explorar os dados em diferentes níveis de agregação, organizando os dados em diferentes níveis de detalhe. Ajuda a fornecer uma visão hierárquica dos elementos dentro de uma dimensão. Por exemplo, numa dimensão de tempo, uma hierarquia permite agrupar os dados pelos níveis: ano, trimestre, mês e dia. Cada nível subsequente representa uma granularidade maior dos dados temporais. As hierarquias também podem ser utilizadas para realizar cálculos agregados em diferentes níveis, proporcionando uma análise mais completa e precisa dos dados.

Por exemplo, é possível calcular o total de vendas por ano, trimestre ou mês, dependendo do nível de detalhe desejado [10];

- **Cubo**: trata-se de uma estrutura multidimensional de dados que permite a combinação das várias medidas através das hierarquias existentes nas diversas dimensões. Com as combinações criadas no seio de um cubo, é possível gerar gráficos e tabelas dinâmicas para a análise dos dados [10].

2.1.4. Data Warehouse

Um DW é um sistema que permite agrupar dados de origens e formatos diferentes num único local de armazenamento. Conforme ilustrado na Figura 2, trata-se de uma BD centralizada e que possui os dados uniformizados para suportar diversas tarefas tais como análise de dados, processos de *data mining*, Inteligência Artificial (IA) ou *Machine Learning* (ML). Com um sistema de DW, uma organização pode executar análises de grandes volumes de dados de forma mais simplificada e focada nas verdadeiras necessidades do negócio.

Figura 2 – Diagrama com os *inputs* e *outputs* de um DW [11]

Os sistemas de DW são soluções utilizadas em BI há mais de três décadas. No entanto, com o aparecimento de novos tipos de dados e em quantidades muito superiores, mas também com a evolução das tecnologias de armazenamento, estes sistemas tiveram igualmente de evoluir. Tradicionalmente, um DW era armazenado em computadores locais e a sua funcionalidade resumia-se em extrair dados de outras origens, limpar e preparar os mesmos e carregar e manter os dados numa BD relacional. Atualmente, um DW pode ser instalado num servidor dedicado ou algures na *cloud*. Alguns DW têm também recursos integrados que permitam análise e visualização de dados [12].

Um DW é a base de qualquer sistema de BI ou de outras funções analíticas. A sua principal função consiste em otimizar os relatórios, *dashboards* e outras ferramentas analíticas, fornecendo as informações necessárias para as tomadas de decisões da organização. Um DW tem muitos benefícios, nomeadamente:

- Melhores funções analíticas: Com o DW, os administradores das organizações têm acesso simplificado a dados provenientes de várias fontes, o que reduz as tomadas de decisões baseadas em informações incompletas [13];

- Consultas mais rápidas: Os DW são dimensionados para que o acesso aos dados seja mais rápido, tornando a respetiva análise também menos demorada. Com um DW, é possível consultar muito rapidamente grandes quantidades de dados consolidados e já trabalhados, com pouco ou até nenhuma intervenção do departamento de TI [13];

- Melhoria da qualidade dos dados: Antes de serem carregados para o DW, os dados passam pelo processo ETL, o que faz com que esses já estejam filtrados, limpos, trabalhados e uniformes, melhorando consideravelmente a qualidade dos mesmos [13];

- Dados históricos: Ao armazenar dados históricos num DW, torna-se possível a criação de algoritmos que possam criar modelos de previsão de tendências futuras, promovendo assim a melhoria contínua da empresa [13].

Um DW pode ser dimensionado através da modelação dimensional dos dados. Trata-se de um modelo que deixa os dados organizados para uma mais fácil compreensão e uso dos mesmos. Isso permite um acesso mais rápido e eficiente aos dados através de *queries* mais simples, traduzindo-se também numa maior performance do sistema. Além disso é um modelo escalável e que consegue adaptar-se a eventuais novos requisitos de negócio [14]. Um exemplo de um modelo dimensional é o esquema em estrela (utilizado neste projeto): é um sistema baseado em tabelas de facto (que contém as métricas ou as medidas alvo de análise) e em dimensões (tabelas que contém as informações que permitam calcular ou descrever as medidas ou as métricas das tabelas de factos).

Na Figura 3 encontra-se ilustrado um exemplo de um esquema em estrela, cuja tabela de factos se encontra no centro e que está relacionada com 5 tabelas dimensões. Esses relacionamentos são criados entre as chaves estrangeiras da tabela de factos e as chaves primárias das tabelas dimensões.

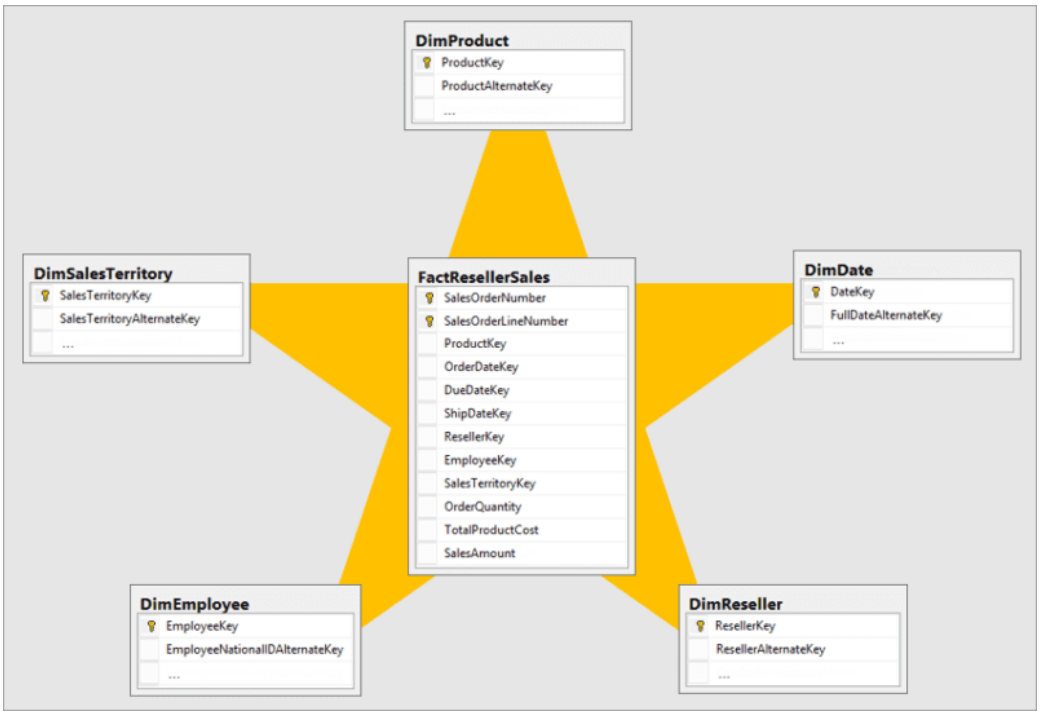


Figura 3 – Ilustração de um modelo dimensional em estrela [15]

2.1.5. Dashboard

Um *dashboard*, de forma genérica, é uma forma de apresentar vários tipos de dados num único local. Trata-se de uma ferramenta que transmite informações claras e que possam ser interpretadas por pessoas que, geralmente, não são profissionais da área dos dados. Os *dashboards* podem, por exemplo, ser utilizados para apresentar gráficos ou tabelas aos membros da administração de uma empresa, de forma a que esses possam tomar decisões de negócios baseados em dados reais e atualizados [16].

2.2. Ferramentas e linguagens

O projeto foi desenvolvido com recurso a diversas ferramentas informáticas. Nesta secção são descritas de forma resumida as ferramentas e as linguagens de programação que foram utilizadas e em que parte do projeto foram essenciais.

2.2.1. Linguagens de programação

Uma linguagem de programação é uma linguagem computacional utilizada por programadores para comunicar com computadores. Trata-se de um conjunto de instruções escritas numa linguagem específica de forma a serem executadas tarefas [17].

Existem várias partes do projeto em que foi necessário realizar uma tarefa muito específica e tal só foi possível com recurso ao desenvolvimento de um algoritmo informático. A linguagem de programação escolhida foi o *Python*. Trata-se de uma linguagem de alto nível, orientada a objetos [18], bastante utlizada em Ciência de Dados. Existem já inúmeras bibliotecas desenvolvidas em *Python* que permitam a pessoas com apenas conhecimentos básicos de programação criarem algoritmos para uma grande variedade de finalidades. A linguagem de programação *Python* torna-se por isso numa escolha intuitiva para quem queira programar sem ter conhecimentos avançados sobre programação.

Para este projeto, os algoritmos em *Python* foram desenvolvidos em ambiente *Visual Studio Code* (V.1.77.3), com o *Python* V3.8.9.

Com a linguagem de programação *Python* foram desenvolvidos os algoritmos que permitiram ir buscar as faturas eletrónicas em formato PDF armazenadas em pastas de correio eletrónico, bem como ler essas faturas de forma a extrair e armazenar numa BD as informações relevantes.

2.2.2. SQL *Server Management Studio*

A BD foi integralmente criada com recurso à ferramenta *SQL Server Management Studio* (SSMS), da *Microsoft*. O SSMS possui um ambiente integrado que permite, entre outras funções, aceder, configurar, gerir, administrar e desenvolver componentes de um servidor SQL [19]. A maioria das tabelas foram criadas diretamente a partir do SSMS, mas o preenchimento das mesmas foi feito com recurso à integração de outras ferramentas. Para este projeto foi utilizada a versão 19.0.2 do SSMS.

2.2.3. *Visual Studio 2022 & SQL Server Integration Services*

O *Visual Studio*, também da *Microsoft*, possui um ambiente de desenvolvimento integrado (IDE) que permite o desenvolvimento de *Softwares* [20]. Uma das extensões que é possível instalar no *Visual Studio* é o *SQL Server Integration Services*. Essa extensão permite o desenvolvimento de soluções do tipo ETL no âmbito de projetos de DW [21]. É com recurso a esta ferramenta que neste projeto se preenchem automaticamente todas as tabelas do DW. Todo o processo ETL pode ser gerido através desta ferramenta e as configurações podem ser feitas por blocos, sem a necessidade de conhecimentos avançados sobre programação ou sobre a linguagem SQL [22].

2.2.4. *Power BI*

Para o presente projeto, foi utilizada a ferramenta *Power BI Desktop* para a criação dos *dashboards*. *Microsoft Power BI* é uma ferramenta dedicada à área do BI. Trata-se de uma plataforma que permite integrar, analisar, visualizar e partilhar dados [23]. Para este projeto foi utilizada a versão 2.118 (junho de 2023).

Existem, no entanto, outras ferramentas como *Tableau* [24] ou *Grafana* [25] para o desenvolvimento de *dashboards* semelhantes aos que foram criados com o *Power BI*. A escolha do *Power BI* para este projeto foi pelo simples facto de já ter sido utilizado no âmbito de outros projetos do Mestrado e também por ser uma ferramenta bastante utilizada no seio das organizações.

2.2.5. *Draw IO*

Foi utilizada a ferramenta *Draw IO* para desenhar o modelo lógico e o DER. Trata-se um uma ferramenta *online* e gratuita que permite desenhar uma grande variedade de diagramas [26].

2.3. Metodologia de desenvolvimento

A metodologia de suporte ao desenvolvimento do projeto foi baseada em *Scrum*. Trata-se de uma metodologia ágil normalmente utilizada como base para a gestão do desenvolvimento de projetos de *software* [27].

Esta metodologia consiste em dividir as diferentes tarefas inerentes ao desenvolvimento do projeto em etapas denominadas *sprints*, e podem existir os seguintes *stackolders*:

- *Product Owner*: é a pessoa responsável por definir as tarefas e as respetivas prioridades a serem realizadas ao longo das diferentes *sprints*;
- *Scrum Master*: é responsável tanto em assegurar que a equipa siga e respeite os valores e as práticas da metodologia *Scrum*, como em garantir que as tarefas a realizar ao longo da *sprint* não sejam excessivas em relação ao que a equipa tem a capacidade de realizar;
- *Scrum Team*: é a equipa de desenvolvimento que trata da execução das tarefas ao longo das *sprints*.

Consoante as necessidades do projeto, podem existir outros intervenientes e a quantidade de membros que forma a *Scrum Team* pode variar. A duração das *sprints* também varia conforme o projeto, mas essa é tipicamente de 1, 2 ou 4 semanas.

A metodologia *Scrum* também prevê diversas reuniões, cada uma com periodicidades e objetivos próprias. As reuniões são essencialmente as seguintes:

- *Sprint Planning Meeting*: esta reunião ocorre antes do início de cada *sprint* e é durante a qual o *Product Owner* apresenta as tarefas que devem ser realizadas com maior prioridade. Conforme essas prioridades, os membros da *Scrum Team* definem quais as tarefas que irão ser realizadas nesta nova *sprint*;

- *Sprint Review Meeting*: esta reunião acontece sempre no final de cada *sprint* e é durante a qual os membros da *Scrum Team* mostram o que foi alcançado ao longo desta *sprint*, muitas vezes sob a forma de demonstração. É então avaliado se os objetivos determinados durante a *sprint planning* foram ou não alcançados;

- *Daily Scrum*: é uma reunião diária que acontece ao longo da *sprint* e preferencialmente de manhã. Tem como objetivo dar a conhecer o que foi realizado no dia anterior e definir as tarefas a realizar ao longo do dia que se inicia. Tipicamente, os impedimentos à realização das tarefas previstas não são discutidos nesta reunião. Essas dificuldades são normalmente comunicadas ao *Scrum Master* em reuniões próprias.

A Figura 4 ilustra um diagrama típico com as diferentes etapas da metodologia *scrum*.

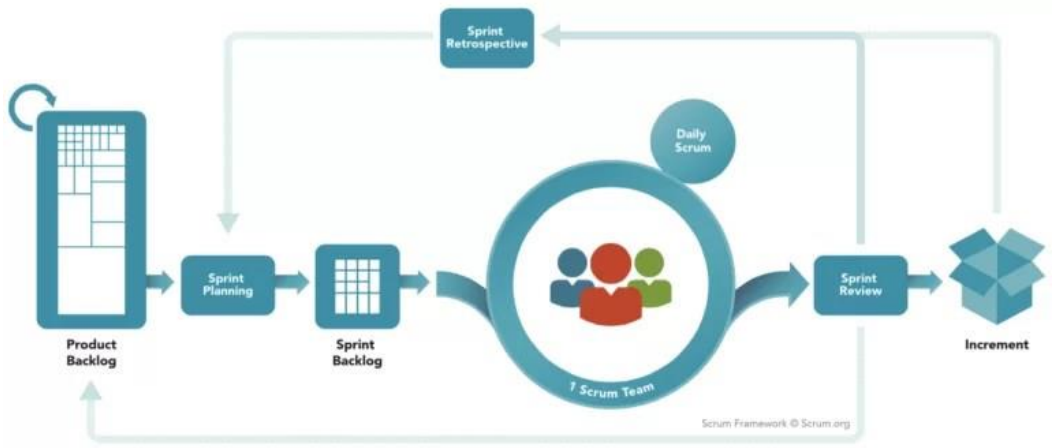


Figura 4 – Diagrama típico da metodologia *Scrum* [28]

Para o presente projeto, esta metodologia foi adaptada para poder funcionar com os *stakeholders* disponíveis. Neste caso, a *Scrum Team* era constituída por apenas um membro – o autor do projeto, enquanto que os papéis de *Product Owner* e *Scrum Master* foram assegurados pelas professoras orientadoras do projeto. As *sprints* tiveram uma duração habitual de 2 semanas, e foram sempre concluídas com reuniões online (*via Microsoft Teams*). Nessas reuniões realizava-se tanto a s*print Review* (mostrar os resultados obtidos durante a última *sprint*), bem como a *sprint Planning* (planeamento da próxima *sprint*).

3. Análise inicial

Neste capítulo é feita uma análise inicial, começando com a secção das US (3.1), onde são apresentadas as principais necessidades do projeto. Na secção 3.2 é feita uma listagem com as funcionalidades do projeto, incluindo a apresentação do respetivo diagrama da arquitetura. Por fim, é feito referencia à parte do dimensionamento da BD, nomeadamente a exposição do modelo de dados na secção 3.3, o modelo conceptual e o DER na secção 3.4 e as tabelas da BD na secção 3.5.

3.1. *User Stories*

O normal desenvolvimento de projetos de BI passa sempre por uma fase de entrevistas durante as quais se procura perceber as reais necessidades da empresa. É com base nessas necessidades que os dados devem ser explorados, trabalhados e apresentados para que possam ser extraídas as informações necessárias e úteis para as tomadas de decisão.

Neste caso em particular, o projeto não foi desenvolvido para responder a perguntas de uma empresa em particular, mas sim para fazer face a uma necessidade própria, pelo que esta fase das entrevistas foi diferente do habitual: foi necessário procurar saber o que realmente é necessário saber sobre os dados disponíveis. Foram assim identificadas diversas necessidades que foram convertidas nas US da listagem abaixo. Estas US podem ser consideradas como as funcionalidades do projeto bem como os objetivos que se pretende alcançar com ele.

US1: Identificar os produtos mais comprados. Pretende-se identificar quais são os produtos mais comprados em termos de quantidade e com que frequência. Poderá ser interessante saber em que loja os produtos que são comprados com maior frequência são mais baratos.

US2: Identificar os produtos mais dispendiosos. Pretende-se identificar quais são os produtos mais caros e com que frequência se costumam comprar. Poder-se-á assim procurar por eventuais produtos equivalentes mais baratos ou até mesmo questionar se realmente existe necessidade de comprar alguns desses produtos.

US3: Analisar variações de preços ao longo do tempo. Pretende-se analisar as variações dos preços dos produtos mais comprados e mais dispendiosos ao longo do

tempo e assim tentar identificar eventuais sazonalidades. Com esta análise espera-se perceber se existe alguma altura do ano ou do mês em que os preços desses produtos atingem algum pico máximo ou mínimo.

US4: Analisar variações de preços entre lojas. Pretende-se analisar a diferença de preços dos produtos mais comprados e mais dispendiosos entre as diferentes lojas e assim perceber onde fica em média mais barato comprar esses produtos.

US5: Analisar os gastos totais por categorias de produtos ao longo do tempo. Com esta análise pretende-se saber em que categoria de produtos se tem gastado mais dinheiro, verificar eventuais tendências de crescimento da despesa por categoria ao longo do tempo e identificar possíveis sazonalidades. O objetivo será estudar a possibilidade de comprar os produtos das categorias mais dispendiosas numa altura do ano em que possam eventualmente ser mais baratos.

US6: Analisar os gastos totais por lojas ao longo do tempo. Pretende-se aqui identificar se existe alguma sazonalidade quanto à loja escolhida para fazer as compras, e analisar se é realmente a melhor loja considerando o estudo da US4.

US7: Analisar os gastos totais ao longo do tempo. Com esta análise pretende-se essencialmente perceber quais foram os gastos com os supermercados nos últimos anos. Com esta informação é possível elaborar um orçamento familiar para o ano seguinte mais realista.

3.2. Funcionalidades do projeto

Com este projeto pretende-se obter gráficos e visualizações sobre as compras feitas para a aquisição de bens essenciais do dia a dia. Para tal, o projeto foi desenvolvido de forma a fornecer as seguintes funcionalidades:

- Armazenamento dos dados relativos a despesas obtidas com a aquisição de bens essenciais numa BD. Esta BD permite armazenar informações sobre produtos (como a categoria, o preço unitário, a taxa de IVA ou os descontos aplicados), lojas, localizações e datas das compras;
- *Download* automático de todos os ficheiros existentes numa pasta do correio eletrónico. Após indicar o endereço de e-mail, a palavra-passe e a pasta onde se

encontram os ficheiros, o sistema descarrega automaticamente cada um dos ficheiros existentes nessa pasta (caso as definições de segurança da conta de e-mail assim o permitam). Essa funcionalidade é utilizada para descarregar as faturas eletrónicas que se encontram armazenadas numa conta de e-mail;

- Extração automática dos dados relevantes para a análise pretendida que se encontram nas faturas eletrónicas. Essa funcionalidade está disponível para as faturas eletrónicas emitidas pelas lojas da cadeia de supermercados Continente;

- Introdução dos dados na BD. Esta funcionalidade pode ser efetuada automaticamente para os dados extraídos a partir das faturas eletrónicas do ponto anterior. Para as restantes faturas, é possível introduzir os respetivos dados na BD de forma manual através de *queries* SQL;

- Execução de um processo ETL de forma a preencher um DW que irá permitir uma análise mais facilitada dos dados introduzidos na BD;

- Geração de gráficos e de tabelas a partir dos dados exportados para o DW, que permitam analisar os dados e tirar conclusões que vão ao encontro da resposta às questões que foram levantadas por este projeto.

Todas as funcionalidades do projeto encontram-se resumidas no diagrama da Figura 5. Os ícones utilizados para criar a figura foram retirados de [29].

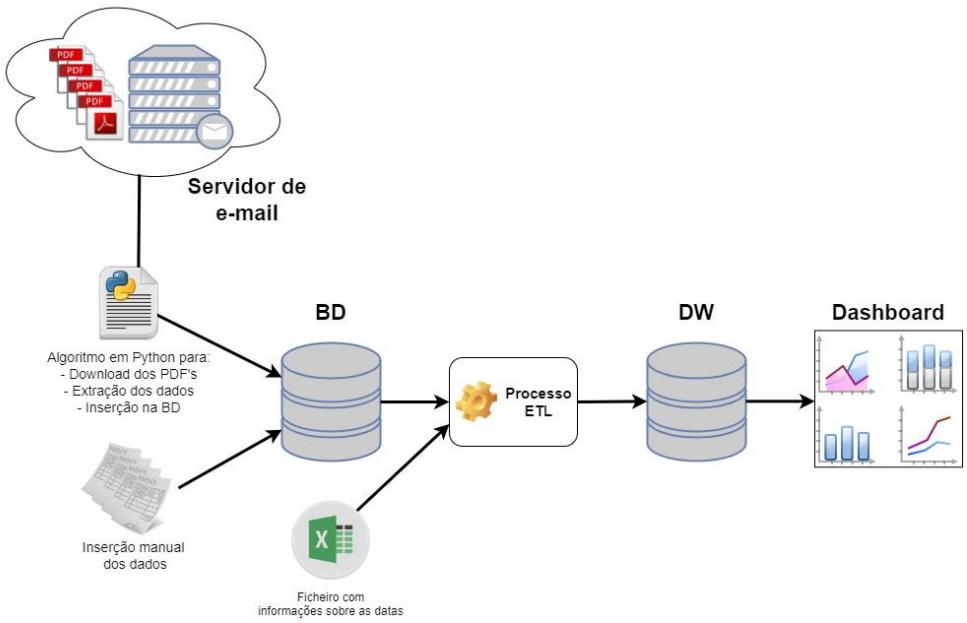


Figura 5 – Diagrama da arquitetura do projeto

3.3. Modelo de dados

Nesta secção aborda-se a lógica da conceção da BD. Com o auxílio do DER, do dicionário de dados e do modelo lógico é explicado como foram dimensionadas as diversas tabelas da BD e como elas se relacionam entre si para conseguir armazenar eficazmente todos os dados presentes nas faturas que possam ser úteis para o projeto.

A criação das tabelas seguiu a seguinte lógica: sempre que um dado pode ser igual entre as diferentes faturas, então existirá uma tabela para esse dado. Por exemplo, todas as informações referentes a uma loja onde foi efetuada uma determinada compra pode ser igual em várias faturas. Para evitar a repetição de todos os dados referentes às lojas, foi criada uma tabela que permite caracterizar cada estabelecimento e foi utilizado um relacionamento para interligar cada compra à sua respetiva loja. O mesmo acontece com os produtos que podem ser referenciados em várias faturas, ou com os Concelhos e os Distritos que se podem repetir pelas diferentes lojas.

3.4. Modelo conceptual

A Figura 6 apresenta o DER da BD. Esse diagrama permite ter uma visão global da estrutura da BD, com cada uma das tabelas que possui, os respetivos campos e relacionamentos.

Nesta BD, todos os relacionamentos tem cardinalidade 1:N e com participação sempre obrigatória do lado N. Consequentemente, as tabelas da BD terão todas a mesma estrutura, isto é, uma coluna para a PK, as colunas necessárias para as chaves estrangeiras e as restantes colunas para os respetivos atributos.

De forma geral, a base de dados vai permitir guardar os dados das faturas (*invoice*) e das linhas das faturas (*invoice_line*). Cada fatura está associada a uma loja (*store*) e uma loja está associado a um concelho (*county*). Cada linha da fatura está relacionada com um produto (*item*), que por sua vez pertence a uma categoria (*category*).

O anexo I do presente relatório possui mais pormenores sobre cada um dos relacionamentos da BD, enquanto que o anexo II corresponde ao modelo lógico do DER, já com uma estrutura muito semelhante às tabelas que foram criadas para a BD.

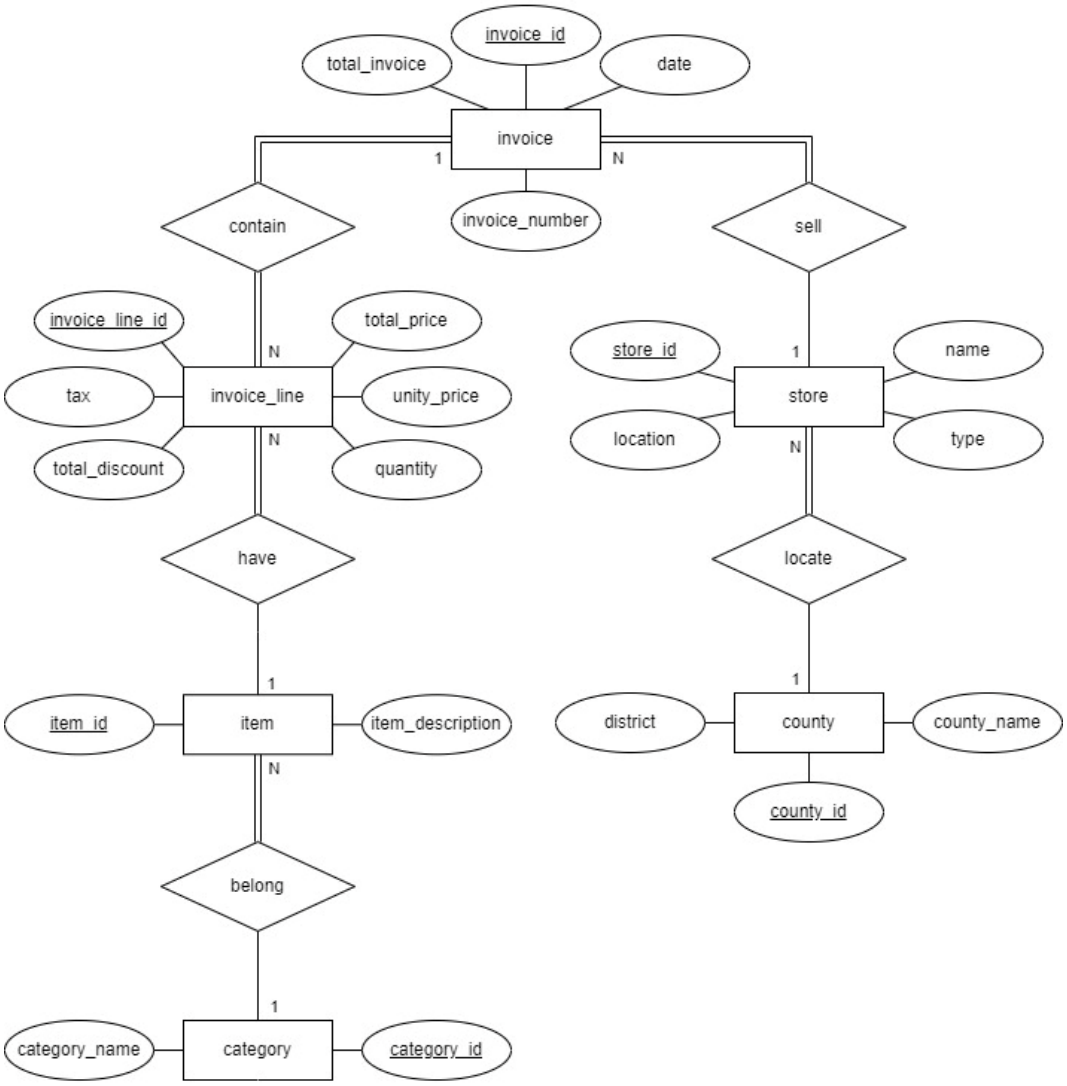


Figura 6 – Diagrama Entidade Relacionamento (DER)

3.5. Tabelas da BD

Nesta secção estão discriminadas cada uma das 6 tabelas da BD.

Segue uma listagem com a descrição de cada tabela e com os respetivos atributos:

- **Tabela *invoice***: Esta é a tabela central da BD. É a tabela que possui os dados relativos a cada uma das faturas, nomeadamente a identificação da loja, da data da compra, do montante total e do número da fatura. São essencialmente dados sobre o cabeçalho da fatura que se encontram nesta tabela e existe uma linha por fatura;

- **Tabela *store***: Nesta tabela encontram-se os detalhes sobre as lojas em si. Os dados que interessam sobre as lojas são o nome, a localização e o tipo de estabelecimento

(que pode ser hipermercado, minimercado, talho, frutaria, farmácia, entre outros) e existe uma linha por loja;

- **Tabela *county***: Aqui encontram-se todos os Concelhos de Portugal à data de hoje, incluindo ilhas, bem como o Distrito ao qual cada Concelho pertence. Existe uma linha por Concelho;

- **Tabela *invoice_line***: A tabela *invoice_line* é a tabela mais extensa da BD por ser a que possui mais amostras e mais atributos. Nela encontram-se os pormenores das linhas de venda de todas as faturas identificadas na tabela *invoice*. Por cada produto comprado em cada uma das faturas é criada uma linha nova nesta tabela (isto é, se uma fatura tiver 20 produtos diferentes, serão criadas 20 linhas novas). Se um mesmo produto voltar a ser comprado noutra fatura, então será criada uma nova linha nesta tabela relacionando esse produto com essa nova fatura. Cada linha tem informações como a identificação do produto comprado, a respetiva quantidade, preço unitário, taxa de IVA e eventual desconto aplicado;

- **Tabela *item***: Esta tabela foi criada para armazenar todos os diferentes produtos que foram comprados e para poder categorizar os mesmos. Existe uma linha por produto e cada linha de venda da tabela *invoice_line* relaciona-se com um destes produtos. Além disso, cada produto está diretamente relacionado com uma das categorias da tabela *category*;

- **Tabela *category***: Por fim, esta tabela permite agrupar os produtos armazenados na tabela *item* por categoria.

Uma descrição mais detalhada de cada tabela encontra-se no anexo III (dicionário de dados).

4. Desenvolvimento

No presente capítulo é descrita a forma como cada uma das etapas do projeto foram desenvolvidas. Em cada secção é descrito todo o procedimento realizado, os testes efetuados e os resultados obtidos. A primeira parte a ser concebida foi a BD e todo o processo utilizado encontra-se descrito na secção 4.1. A secção 4.2 refere-se ao procedimento utilizado para efetuar o *download* automático de todas as faturas eletrónicas que se encontram no correio eletrónico. A secção 4.3 descreve o algoritmo criado de raiz que permite extrair os dados dessas faturas, enquanto que na secção 4.4 está a descrição do algoritmo criado para a introdução desses mesmos dados na BD. De seguida, a secção 4.5 descreve a criação do DW, que inclui o dimensionamento e a criação das tabelas e na secção seguinte é explicado como o processo ETL foi desenvolvido para poder preencher automaticamente o DW. A secção 4.7, por sua vez, aborda o tema do *dashboard*, nomeadamente como o mesmo foi desenvolvido e também a apresentação de alguns gráficos resultantes e respetivas conclusões. Por fim, a última seção deste capítulo expõe uma análise crítica sobre as conclusões retiradas com a análise dos gráficos.

4.1. Criação da base de dados

A BD foi criada com o *Microsoft* SSMS num computador local. A primeira fase consistiu em criar as tabelas com o respetivo nome. De seguida foram criadas as colunas de cada tabela com auxílio da opção *Design*, sempre em concordância com o definido no dicionário de dados (anexo III), isto é, definindo o tipo de campo, se é ou não campo obrigatório, qual é a PK e configurando as restrições. As chaves estrangeiras foram configuradas criando os respetivos relacionamentos. Esta forma de criação de tabelas é facilitado para técnicos com pouca experiência nessa área, já que permite criar tabelas de forma intuitiva sem recorrer às tradicionais *queries* SQL.

Com as tabelas criadas e com os relacionamentos devidamente configurados, foi possível gerar o modelo da Figura 7 através do *reverse engineering*.

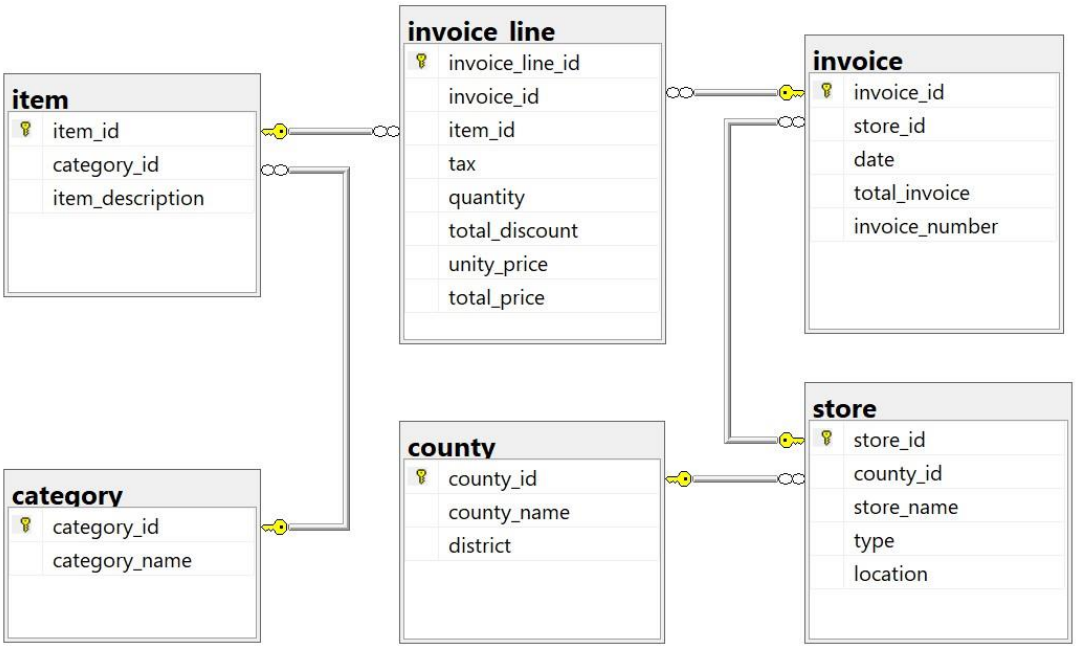


Figura 7 – Modelo da BD gerado através do *Reverse Engineering* do SSMS

4.2. Download das faturas eletrónicas existentes no e-mail

As tabelas da BD acabadas de criar devem ser agora preenchidas com dados de faturas de supermercados. Para tal, é necessário extrair um conjunto de faturas eletrónicas que se encontram guardadas numa caixa de correio eletrónico pessoal, para posteriormente retirar dessas faturas os dados relevantes.

Trata-se de cerca de uma centena de e-mails contendo uma ou mais faturas eletrónicas, faturas essas que devem ser descarregadas para tratamento futuro. Considerando a grande quantidade de documentos a descarregar, esta tarefa encontra-se sujeita a enganos para ser realizada manualmente – por exemplo, engano como não descarregar um ou outro documento ou descarregar várias vezes o mesmo. Além disso, esta tarefa torna-se monótona e repetitiva. Para contornar essa situação, resolveu-se procurar uma solução que realizasse essa tarefa de forma automática. Essa solução foi encontrada no artigo de Sanket Doshi [30], e consiste num pequeno *script* desenvolvido em *Python* que, depois de devidamente configurado, acede à(s) pasta(s) especificada(s) do correio eletrónico e descarrega os ficheiros desejados. Para tal, o algoritmo utiliza as seguintes bibliotecas:

- *imaplib*: Esta biblioteca define classes que permitem criar ligações a servidores de *Internet Message Access Protocol* (IMAP) [31]. IMAP é um protocolo que armazena mensagens de e-mail num servidor e permite a manipulação das mesmas tal como se estivessem armazenadas no próprio computador (ou outro dispositivo) [32];

- *os*: Com esta biblioteca, é possível manusear algumas funcionalidades do sistema operativo, nomeadamente na manipulação de pastas e ficheiros [33];

- *email*: Esta biblioteca permite a gestão de mensagens de e-mail [34].

Para facilitar o acesso do *script* à conta Google onde se encontra o correio eletrónico, foi criada uma *password* para aplicações. Trata-se de uma *password* que pode ser utilizada por aplicações que não permitam a autenticação *Single Sign-On* (SSO) da Google [35]. Essa *password*, configurada diretamente nas configurações da conta da Google, é diferente da utilizada para se autenticar nessa conta e pode ser eliminada a qualquer momento.

Na Figura 8 encontra-se ilustrado o código de importação das bibliotecas, bem como o código que define a localização da pasta de destino (onde irão ficar armazenados os ficheiros), o endereço de e-mail e a respetiva *password* de acesso (*password* aqui omitida por razões de segurança).

```
1   import imaplib
2   import os
3   import email
4
5   pasta_destino ="/Users/micka/Dropbox/projeto/notebooks/anexos"
6   email_user = "mickael.v.silva@gmail.com"
7   email_pass = "???" # Palavra passe omitida
8   mail = imaplib.IMAP4_SSL("imap.gmail.com",993)
9   mail.login(email_user, email_pass)
10  mail.select('Faturas_continente')
11
```

Figura 8 – Código para importação das faturas eletrónicas a partir do e-mail: dados introduzidos

Para além dos dados de *login*, foi igualmente necessário colocar no *script* o endereço do servidor IMAP da *Google* (*imap.gmail.com*) e a respetiva porta de comunicação para IMAP4_SSL (por defeito: 993). Por fim, na linha 10 foi introduzido o nome da pasta onde o algoritmo tem de ir buscar as faturas eletrónicas (neste caso, trata-se de uma pasta existente no e-mail e denominada "*Faturas_continente*").

Na Figura 9 ilustra-se o código que efetua o *download* dos ficheiros. A única alteração que foi efetuada em relação ao algoritmo original foi o acréscimo de uma variável auxiliar chamada "*contador*" (linhas 16, 37 e 38). Essa variável é inicializada com o valor 0 e é incrementada em 1 valor, sempre que é descarregado um novo documento, de forma a conseguir apresentar no final a quantidade de documentos descarregados.

```python
contador = 0
for num in data[0].split():
    typ, data = mail.fetch(num, '(RFC822)' )
    raw_email = data[0][1]
# converts byte literal to string removing b''
    raw_email_string = raw_email.decode('utf-8')
    email_message = email.message_from_string(raw_email_string)
# downloading attachments
    for part in email_message.walk():
        if part.get_content_maintype() == 'multipart':
            continue
        if part.get('Content-Disposition') is None:
            continue
        fileName = part.get_filename()
        if bool(fileName):
            filePath = os.path.join(pasta_destino, fileName)
            if not os.path.isfile(filePath):
                fp = open(filePath, 'wb')
                fp.write(part.get_payload(decode=True))
                fp.close()
            subject = str(email_message).split("Subject: ", 1)[1].split("\nTo:", 1)[0]
        contador += 1
print(f"foram descarregado {contador} ficheiro(s)")
```

Figura 9 – Código para importação das faturas eletrónicas a partir do e-mail: download dos ficheiros

Como resultado final, obtiveram-se 95 ficheiros descarregados na pasta especificada em menos de 30 segundos. A Figura 10 mostra o *Output* obtido.

```
PS C:\Users\micka\Dropbox\projeto\notebooks> py importar_anexos_email_COMpass.py
Foram descarregados 95 ficheiros
```

Figura 10 – *Output* obtido pelo algoritmo no fim de descarregar todos os ficheiros do e-mail

4.3. Recolha dos dados a partir das faturas eletrónicas

O próximo passo é retirar os dados existentes nas faturas eletrónicas (documentos em formato PDF) para posteriormente preencher a BD. Considerando a grande quantidade de faturas (cerca de 100 documentos) e, consequentemente, a grande quantidade de dados a extrair das mesmas, resolveu-se criar um algoritmo para a realização desta tarefa. Utilizar um algoritmo para a extração dos dados tem as seguintes vantagens:

- Mais rapidez na execução;
- Grande diminuição da possibilidade de erros;
- Eliminação de uma tarefa repetitiva e monótona;
- Permite deixar os dados já formatados para serem inseridos na BD.

Este algoritmo foi criado de raiz e igualmente com linguagem *Python*, sendo que as bibliotecas utilizadas foram as seguintes:

- *PyPDF2*: Esta biblioteca *open source* permite manipular ficheiros PDF. As suas funções permitem realizar operações como dividir ou unir ficheiros, extrair imagens, extrair texto, reduzir tamanho, entre outras funcionalidades básicas e avançadas [36];

- *Re*: Trata-se de uma biblioteca que permite a utilização de Expressões Regulares (*RegEx*) similares às utilizadas em *PERL*. *PERL* é uma linguagem de programação utilizada para a manipulação de texto e as *RegEx* permitam determinar se uma dada *string* possui ou não um determinado padrão (*pattern*) [37];

- *Pandas:* biblioteca *open source* para análise de dados [38].

Nas subsecções que se seguem são explicadas as várias etapas que foram criadas no algoritmo para conseguir extrair os dados necessários das faturas eletrónicas. Para demonstrar os resultados das várias etapas, foi utilizada uma das faturas, doravante denominada *fatura exemplo*, fatura essa datada de maio 2019 e que possui vários produtos de várias categorias, valores diferentes de IVA, vários descontos diretos e também descontos em cartão. O algoritmo foi desenvolvido num ficheiro do tipo *Interactive Python Notebook* (IPYNB). Esse tipo de ficheiros é utilizado pelo *Jupyther Notebook* e permite guardar e partilhar documentos com algoritmos, equações matemáticas, gráficos, tabelas e texto simples [39]. O facto de utilizar um ficheiro IPYNB facilitou o desenvolvimento do código, já que esse permite a execução do mesmo por parcelas e a introdução de texto livre ao longo do ficheiro, o que permite ter o mesmo bem organizado e explicado.

Nos exemplos aqui facultados, alguns dados encontram-se ocultados por se tratar de dados pessoais e não utilizados no projeto.

4.3.1. Extração do texto

O primeiro passo consistiu em extrair e apresentar o texto do ficheiro PDF. A leitura do ficheiro é feita com a função *PdfReader()* e a extração do respetivo texto através da função *extract_text()*. Como resultado, obteve-se o apresentado na Figura 11, isto é, uma *string* contendo todos os caracteres seguidos existentes no ficheiro, e onde as mudanças de linhas são representadas pelo caractere especial "\n". Para o caso específico da fatura exemplo, a *string* possui um total de 5.330 caracteres.

```
## Leitura do ficheiro PDF
reader = PdfReader("fatura1.pdf")
page = reader.pages[0]# Ler página 1 do ficheiro. As faturas do Continente apenas tem 1 página
✓ 0.0s

#Extração do texto presente no ficheiro PDF
#Resulta numa string única com todo o texto presente no ficheiro
text = page.extract_text()

print(len(text))
text
✓ 0.0s

5330

'CBD Marrazes\nCONTINENTE HIPERMERCADOS, S.A.\nEstrada da Outurela n.118 Edif. Immopolis Bloco D \n2790-114 Carnaxi
```

Figura 11 – Código criado para extração e apresentação do texto presente nas faturas eletrónicas

Para facilitar a manipulação do texto extraído, dividiu-se a *string* numa lista de *strings* denominada *text_splitted* através da linha de código "*text_splitted = re.split(r"[\n]", texto)*". A função *re.split()* cria um novo item na lista *text_splitted* sempre que encontrar o caracter especial "\n" (mudança de linha). A variável *text_splitted* é então uma lista onde cada item corresponde a uma linha de texto da fatura. A Figura 12 ilustra o resultado obtido nesta etapa com a fatura exemplo, sendo que, neste caso, a lista possui 201 itens, correspondentes às 201 linhas de texto da fatura.

```
#Dividir a string sempre que encontrar "\n" (mudança de linha)
#O resultado é uma lista de strings
text_splitted = re.split(r"[\n]", text)

print(f"Quantidade de itens na lista: {len(text_splitted)}")
text_splitted
✓ 0.0s

Quantidade de itens na lista: 201

['CBD Marrazes',
 'CONTINENTE HIPERMERCADOS, S.A.',
 'Estrada da Outurela n.118 Edif. Immopolis Bloco D ',
 '2790-114 Carnaxide',
 'Registada CRC Porto sob nº 501591109',
 'NIF: PT501591109',
 'C.S: 66.500.000,00 EUR SIRPEEE: PT000251',
 'Fatura Original',
 'Nro: ET INQ008/000001 22/05/2019 16:14',
 'Cliente: Mickael Vieira da Silva',
 'NIF: [redacted]',
 'Morada: [redacted]',
 'Local: [redacted]',
 'C. Postal: [redacted]',
 'IVADESCRICAO VALOR',
...
 '[redacted] - Disponivel desde 23/05/2019',
 '[redacted] - ',
 'SALDO NO CARTAO [redacted]',
 '[redacted] - Disponivel a partir de 23/05/2019',
 'Ja ganhou com o cartao [redacted]']
```

Figura 12 – Código criado para dividir a *string* resultante da extração de texto numa lista de *strings*

4.3.2. Recolha dos dados

A presente etapa consiste em retirar os dados que interessam do texto acabado de extrair. Para tal, teve-se de encontrar um padrão que se repete em todas as faturas e que permite ao algoritmo identificar as linhas que correspondem a informações úteis. Na Figura 13 encontra-se ilustrado a fatura exemplo (recortada no meio e no final devido ao seu grande tamanho) com os padrões destacados.

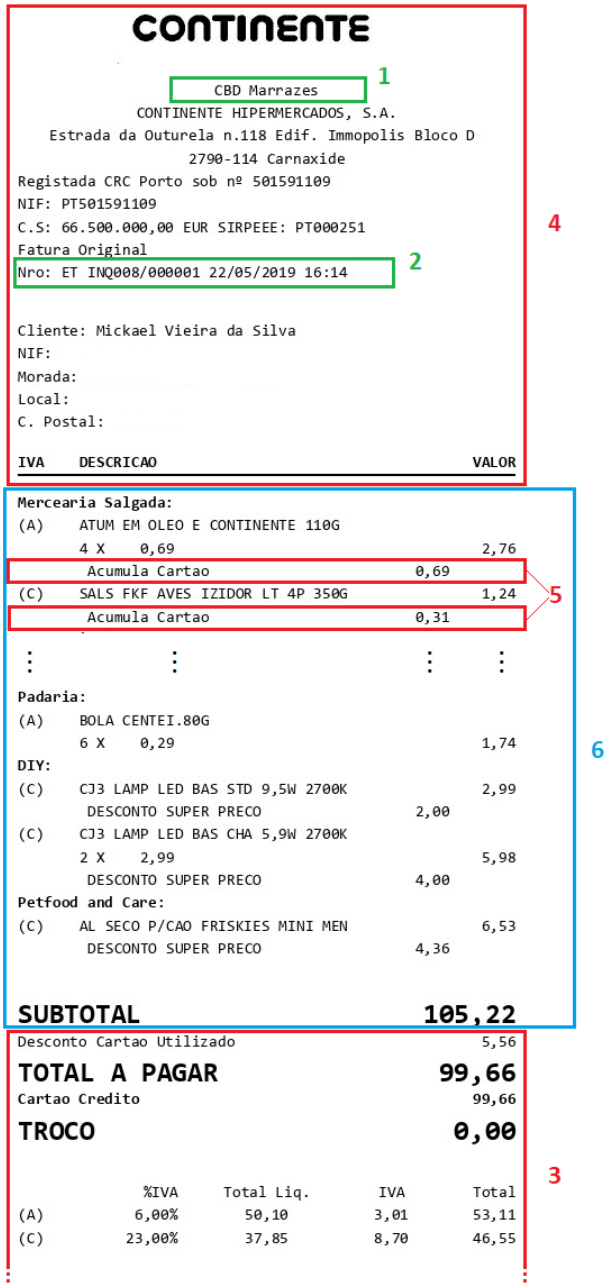


Figura 13 – Padrão encontrado nas faturas eletrónicas que permite ao algoritmo retirar os dados

Nas subsecções seguintes estão explicadas de que forma os dados foram retirados de cada um dos padrões identificados na Figura 13.

4.3.2.1. Identificação da loja

O primeiro dado a ser recolhido é o nome da loja. Essa informação encontra-se na primeira linha das faturas, pelo que a sua recolha é efetuada através do código da Figura 14.

Identificação da loja

A primeira linha permite identificar a loja

```
1  loja = text_splitted[0]
2  loja
```

✓ 0.0s

```
'CBD Marrazes'
```

Figura 14 – Código criado para identificação da loja

4.3.2.2. Identificação do número da fatura e da data

Em todas as faturas eletrónicas, o número da fatura e a data da compra encontram-se na mesma linha, linha essa que se inicia sempre por "*Nro:*". No entanto, a posição dessa linha varia em algumas faturas. Para extrair o número da fatura e a data, foi necessário em primeiro lugar implementar um ciclo repetitivo "FOR" que percorresse todas as linhas analisando os primeiros 4 caracteres de cada uma. Se for encontrada uma linha que se inicia com os caracteres "*Nro:*", então o número da fatura (que corresponde à segunda e à terceira palavra dessa linha) é guardado na variável *num_fatura* e a data (que corresponde à quarta palavra) fica armazenada na variável *data*. O código referente a essa tarefa encontra-se ilustrado na Figura 15.

Identificação da fatura e da data

Estas informações são retiradas da linha que começa por "Nro:"

```
1  for i in range(0, len(text_splitted)):  #Ciclo 'for' para procurar a linha que começa por "Nro:"
2      aux=text_splitted[i]
3      if aux[0:4]=="Nro:":
4          listAux = re.split("\s+", text_splitted[i]) # Copiar cada palavra da string (separados
5                                                      # por 1 ou mais espaços) para uma lista
6          num_fatura=listAux[1]+" "+listAux[2] # Retira o nº da fatura, que corresponde à 2a e 3a string
7          dataAux=listAux[3] # Retira a data, que corresponde à 4a string
8          break # Para forçar o fim do ciclo 'for'. Não vale a pena continuar
9
10 print(f"número da fatura: {num_fatura}")
11 data=dataAux[6:10]+'-'+dataAux[3:5]+'-'+dataAux[0:2]
12 print(f"Data: {data}")
```

✓ 0.0s

```
número da fatura: ET INQ008/000001
Data: 2019-05-22
```

Figura 15 – Código criado para identificação do número da fatura e da data da compra

Os dados adquiridos até agora (loja, data e número da fatura) são introduzidos numa BD do tipo *Pandas* denominada *dadosFatura* através do código da Figura 16. O objetivo dessa BD auxiliar é ser futuramente exportada para a BD do projeto.

Criação da BD dadosFatura

Criação da BD e respetivo preenchimento com os dados já adquiridos (loja, data e número da fatura)

```
1  dadosFatura = pd.DataFrame(data={'store':[loja],
2                                   'date':[data],
3                                   'invoice_number':[num_fatura]})
4  dadosFatura
```

✓ 0.0s

	store	date	invoice_number
0	CBD Marrazes	2019-05-22	ET INQ008/000001

Figura 16 – Código criado para a criação da BD auxiliar dadosFatura

4.3.2.3. Remoção das linhas inferiores

Outro padrão que foi encontrado no seio de todas as faturas eletrónicas é que abaixo da linha que apresenta o total da fatura não existem informações relevantes para a BD. Por isso, procedeu-se à remoção dessas linhas através do código da Figura 17.

```
1  #Encontra a primeira linha com a string "TOTAL" ou "SUBTOTAL, e apaga todas as linhas seguintes
2  for i in range(0, len(text_splitted)):
3      aux=re.search("\ATOTAL",text_splitted[i]) # \A Matches only at the start of the string
4      if aux != None:  #parar o ciclo quando encontrar a palavra "total" no inicio da string
5          break
6      aux=re.search("SUBTOTAL",text_splitted[i])
7      if aux != None:  #parar o ciclo quando encontrar a palavra "subtotal"
8          break
9
10 #Proteção para o caso de não encontrar a string pretendida
11 if i == len(text_splitted): # i: número do indice onde "total" foi encontrado"
12     print("as strings 'TOTAL' e 'SUBTOTAL' não foram encontradas")
13 else:
14     text_splitted=text_splitted[:i+1]
15
16 text_splitted
```

✓ 0.0s

```
['CBD Marrazes',
 'CONTINENTE HIPERMERCADOS, S.A.',
 'Estrada da Outurela n.118 Edif. Immopolis Bloco D ',
 '2790-114 Carnaxide',
 'Registada CRC Porto sob nº 501591109',
 'NIF: PT501591109',
 'C.S: 66.500.000,00 EUR SIRPEEE: PT000251',
 'Fatura Original',
 'Nro: ET INQ008/000001 22/05/2019 16:14',
 'Cliente: Mickael Vieira da Silva',
 'NIF: PT247458570',
 'Morada: Travessa dos Gagos n16',
 'Local: Bidoeira de Baixo',
 'C. Postal: 2415-001',
 'IVADESCRICAO VALOR',
...
 'DESCONTO SUPER PRECO 4,00',
 'Petfood and Care:',
 '(C)AL SECO P/CAO FRISKIES MINI MEN 6,53',
 'DESCONTO SUPER PRECO 4,36',
 'SUBTOTAL 105,22']
```

Figura 17 – Código criado para remoção das últimas linhas, irrelevantes para a BD

4.3.2.4. Remoção das linhas superiores

À semelhança das linhas inferiores, existem linhas no início do documento que são agora irrelevantes para a BD. Como já foram extraídos os dados sobre a loja, a data e o número da fatura, todo o cabeçalho da fatura (até a linha que tem escrito "IVA DESCRICAO VALOR") deixou de ter informações de interesse para a BD. Essas linhas são então removidas através do código da Figura 18.

```
#Encontra a string "DESCRICAO VALOR", e apaga desde o início até essa linha
for i in range(0, len(text_splitted)-1):
    aux=re.search("DESCRICAO VALOR",text_splitted[i])
    if aux != None: #para o ciclo quando encontrar as palavras "DESCRICAO VALOR"
        break

#Proteção para o caso de não encontrar a string pretendida
if i == len(text_splitted): # i: número do indice onde "total" foi encontrado
    print("a string 'DESCRICAO VALOR' não foi encontrada")
else:
    text_splitted=text_splitted[i+1:]

text_splitted
```

✓ 0.0s

```
['Mercearia Salgada:',
 '(A)ATUM EM OLEO E CONTINENTE 110G',
 '4 X    0,69 2,76',
 'Acumula Cartao 0,69',
 '(C)SALS FKF AVES IZIDOR LT 4P 350G 1,24',
 'Acumula Cartao 0,31',
 '(C)FEIJAO ENCARN CONTINENTE 830G 0,90',
 '(C)GRAO BICO CONTINENTE 830G 0,90',
 '(C)MILHO DOCE CONTINENTE 3*150G 1,29',
 'Mercearia Doce:',
 '(C)PREP BRANCA NEVE BOLO CHOC 450G',
 '2 X    2,29 4,58',
 '(C)BOL.ARROZ CONT.EQUIL.C/SAL 130G 0,89',
 'Acumula Cartao 0,22',
 'Higiene:',
...
 'DESCONTO SUPER PRECO 4,00',
 'Petfood and Care:',
 '(C)AL SECO P/CAO FRISKIES MINI MEN 6,53',
 'DESCONTO SUPER PRECO 4,36',
 'SUBTOTAL 105,22']
```

Figura 18 – Código criado para remoção das primeiras linhas, agora irrelevantes para a BD

4.3.2.5. Remoção das linhas "desconto em cartão"

A lista *texto_splitted* ficou agora somente com as linhas de venda. No entanto, podem ainda existir linhas referentes a um desconto em cartão que o produto excecionalmente possa oferecer na data dessa compra. Como no âmbito deste projeto não são considerados os descontos em cartão, essas linhas tronam-se irrelevantes, pelo que se utilizou o código da Figura 19 para eliminar essas linhas.

```python
#apagar as linhas que contenham simultaneamente "ACUMULA" e "CARTAO"
for i in range(0, len(text_splitted)-1):# Procura em cada linha a existencia de "acumula" e "cartao"
                                        # (?i) para não ser sensivel a letras minusculas ou maiusculas

    aux=re.search(r'(?m)(?i)^(?=.*acumula)(?=.*cartao).+$',text_splitted[i])
    if aux != None:
        text_splitted.pop(i) #Se encontrar "ACUMULA" e "CARTAO" na mesma linha, apaga essa mesma linha
    if i==len(text_splitted)-1: #Ao ir apagando linhas, o len(text_splitted) vai variando.
        break         # quando i for igual a len(text_splitted), conclui o ciclo

text_splitted
```

✓ 0.0s

```
['Mercearia Salgada:',
 '(A)ATUM EM OLEO E CONTINENTE 110G',
 '4 X    0,69 2,76',
 '(C)SALS FKF AVES IZIDOR LT 4P 350G 1,24',
 '(C)FEIJAO ENCARN CONTINENTE 830G 0,90',
 '(C)GRAO BICO CONTINENTE 830G 0,90',
 '(C)MILHO DOCE CONTINENTE 3*150G 1,29',
 'Mercearia Doce:',
 '(C)PREP BRANCA NEVE BOLO CHOC 450G',
 '2 X    2,29 4,58',
 '(C)BOL.ARROZ CONT.EQUIL.C/SAL 130G 0,89',
 'Higiene:',
 '(C)DEO SPRAY REXONA BIORYTHM 200ML',
 '2 X    1,85 3,69',
 'DESCONTO SUPER PRECO 3,69',
...
 'DESCONTO SUPER PRECO 4,00',
 'Petfood and Care:',
 '(C)AL SECO P/CAO FRISKIES MINI MEN 6,53',
 'DESCONTO SUPER PRECO 4,36',
 'SUBTOTAL 105,22']
```

Figura 19 – Código criado para remoção das linhas com eventuais descontos em cartão

4.3.2.6. Extração dos dados das linhas de venda

Esta última fase consiste em extrair os dados relativos às linhas de venda. É nas linhas de venda que se encontram a descrição dos produtos que foram comprados, a respetiva categoria, quantidade, preço unitário, subtotal e taxa de IVA.

Para ir armazenando os dados que se vão extraindo nesta fase, criou-se uma BD temporária em formato *Pandas* com o nome *lista_produtos*, conforme pode ser visualizado na Figura 20.

```python
# Criação da base de dados temporária vazia
lista_produtos = pd.DataFrame(columns=['category_name',
                                       'item_description',
                                       'tax',
                                       'quantity',
                                       'total_discount',
                                       'total_price',
                                       'unity_price'])
```

Figura 20 – Código criado para a criação da BD temporária *lista_produtos*

Para identificar e extrair essas informações, o algoritmo foi desenvolvido para percorrer cada uma das linhas de venda das faturas eletrónicas (assinalado em azul na Figura 13) através de um ciclo de repetição *WHILE*. Com o auxílio das variáveis *indice* (para navegar através dos índices da lista "*text_splitted*") e *num_produto* (para ir numerando os diferentes produtos que se forem encontrando na fatura), esse ciclo *WHILE* começa por guardar na lista *texto_linha_atual* cada uma das palavras que existem na primeira linha e na lista *texto_linha_seguinte* cada uma das palavras que existem na linha seguinte. Além disso, retém também na variável *is_num* o primeiro caracter da linha seguinte (vai ser útil nos próximos passos saber se esse caracter é ou não um número). Por fim, o algoritmo vai analisar o que se encontra armazenado nas variáveis *texto_linha_atual* e *texto_linha_seguinte* e, consoante o padrão encontrado nas *strings* dessas variáveis, procede à extração dos respetivos dados. O código ilustrado na Figura 21 é o que vai iniciar esse ciclo *WHILE*.

```
# a variavel indice vai ser usada para navegar através dos indices da string 'text_splitted'
indice=0
# a variável numProduto vai ser usada para numerar os diferentes produtos da fatura
numProduto=1

# A forma como a informação é apresentada depende de vários fatores, e isso
# condiciona a forma como devemos extrair os dados:

# Inicio do ciclo repetitivo para correr todas as linhas de produtos
while indice+2<=len(text_splitted):
    # copiar cada palavra da string (separados por 1 ou mais espaços) para uma lista
    texto_linha_atual = re.split("\s+", text_splitted[indice])
    # copiar cada palavra da string seguinte para uma outra lista
    texto_linha_seguinte = re.split("\s+", text_splitted[indice+1])
    # Reter o primeiro caracter da linha seguinte (para identificar a seguir se
    is_num=texto_linha_seguinte[0]                          # é ou não um numero)
```

Figura 21 – Código criado para o início da análise das linhas de venda das faturas eletrónicas

Os produtos estão agrupados por categoria. A linha que identifica o início de uma nova categoria tem sempre o seguinte formato: "*NOME_DA_CATEGORIA:*". Ou seja, sempre que se identificar o caracter ":" no final da *string*, então é porque daqui em diante, a categoria é a que está indicada nessa linha. A identificação da categoria é realizada através do código da Figura 22.

```
# Quando a linha corresponde a uma nova categoria
    if text_splitted[indice][-1]==":":
        category_name=text_splitted[indice]
        category_name=category_name[:-1]
        indice+=1
```

Figura 22 – Código criado para identificação da categoria

Se a linha em análise não for a identificação de uma nova categoria, então é porque se trata de um produto. Existem agora 2 situações possíveis: quando a quantidade desse produto é igual a 1, então a linha de venda tem o aspeto da Figura 23. Caso a quantidade seja diferente de 1 (inclusive quando a quantidade é um número decimal entre 0 e 1 – o que acontece quando, por exemplo, se compra menos de 1 kg de fruta), então a linha de venda tem o aspeto da Figura 24.

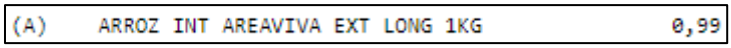


Figura 23 – Exemplo de uma linha venda de uma fatura eletrónica com quantidade 1

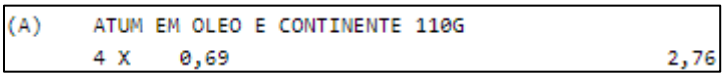


Figura 24 – Exemplo de uma linha venda de uma fatura eletrónica com quantidade diferente de 1

Para fazer face a cada uma dessas situações, foram criados os algoritmos apresentados na Figura 25 e na Figura 26. São eles que retiram as informações necessárias das linhas de venda, isto é, a descrição do produto, a quantidade, o preço unitário, o subtotal e a taxa do IVA.

```
# Quando a quantidade comprada é 1
    else:
        texto_linha_seguinte = re.split("\s+", text_splitted[indice])
        quantity=1
        unity_price=texto_linha_seguinte[-1]
        #substituir "," por "." de forma a que esta variável possa ser interpretada como float
        unity_price=float(unity_price.replace(",","."))
        total_price=texto_linha_seguinte[-1]
        #substituir "," por "." de forma a que esta variável possa ser interpretada como float
        total_price=float(total_price.replace(",","."))
        item_description = ' '.join(texto_linha_seguinte[:-1])
        item_description=item_description[3:]
        tax=texto_linha_atual[0]
        tax=tax[1]
        total_discount=0.0
```

Figura 25 – Código criado para extrair a descrição, preço, subtotal e IVA, quando a quantidade é 1.

```
# Quando a quantidade comprada é diferente de 1
#(inclusive quando se compra menos de 1kg de fruta por exemplo)
    elif is_num[0]=="1" or is_num[0]=="2" or is_num[0]=="3" or is_num[0]=="4" or is_num[0]=="5" or \
        is_num[0]=="6" or is_num[0]=="7" or is_num[0]=="8" or is_num[0]=="9" or is_num[0]=="0":
        quantity=texto_linha_seguinte[0]
        #substituir "," por "." para que esta variável possa ser interpretada como float na BD SQL
        quantity=float(quantity.replace(",","."))
        unity_price=texto_linha_seguinte[2]
        unity_price=float(unity_price.replace(",","."))
        total_price=texto_linha_seguinte[3]
        total_price=float(total_price.replace(",","."))
        item_description=text_splitted[indice]
        item_description=item_description[3:]
        total_discount=0.0
        tax=texto_linha_atual[0]
        tax=tax[1]
```

Figura 26 – Código alternativo para extração dos dados, quando a quantidade é diferente de 1.

Independentemente da quantidade comprada, pode ainda existir uma linha adicional correspondente ao desconto direto. Na Figura 27 é apresentada uma linha de venda típica das faturas eletrónicas que possui o referido desconto direto. O respetivo código utilizado para identificar e extrair essa informação encontra-se na Figura 28.

```
(A)     MASSA NINHOS C/OVOS CNT 500G                    1,29
         DESCONTO SUPER PRECO                  0,40
```

Figura 27– Exemplo de uma linha venda de uma fatura eletrónica com desconto direto.

```
41      # Se houver um desconto direto
42          if indice+2<len(text_splitted): # Proteção do código caso seja a última linha de venda da fatura
43              texto_linha_atual = re.split("\s+", text_splitted[indice+2])
44
45              if texto_linha_atual[0]=="POUPANCA" or texto_linha_atual[0]=="DESCONTO":
46                  total_discount=texto_linha_atual[-1]
47                  #substituir "," por "." para que esta variável possa ser interpretada como float na BD SQL
48                  total_discount=float(total_discount.replace(",","."))
49                  indice+=3
50              else:
51                  indice+=2
52          else:
53              indice+=2
```

Figura 28 – Código criado para extração do desconto direto, caso existe.

No fim do algoritmo ter recolhido todas as informações relevantes de uma linha de código, o mesmo acrescenta esses dados à BD *listaProdutos* (através do código ilustrado na Figura 29), e reinicia o ciclo *WHILE* com a próxima linha de venda, atualizando as variáveis *num_produto* e *indice*.

```
91          lista_produtos.loc[str(num_produto)]=[category_name,
92                                                item_description,
93                                                tax,
94                                                quantity,
95                                                total_discount,
96                                                total_price,
97                                                unity_price]
98
99          num_produto+=1
```

Figura 29 – Código criado para o acréscimo dos dados extraídos na BD temporária *lista_produtos*.

Quando for detetado que a última linha de venda já foi analisada, então o ciclo *WHILE* é terminado e o algoritmo prossegue para a próxima etapa.

4.3.2.7. Tratamento do valor total e do IVA

Antes de começar a introduzir os dados recolhidos na BD SQL, ainda é necessário calcular o custo total da fatura e transformar o campo "*tax*", de forma a que esse corresponda às definições da BD.

O valor total da fatura é calculado fazendo o simples somatório dos subtotais de cada linha de venda. Poder-se-ia também extrair esse dado diretamente da fatura eletrónica, mas calcular esse campo permite comparar esse valor com o total que está escrito na fatura e assim, se ambos os valores forem iguais, então é porque a extração aparenta ter sido bem feita. O total da fatura é calculado e acrescentado à BD *Pandas dados_fatura* através do código da Figura 30.

```
# calcular o valor total da fatura

total_invoice=lista_produtos['total_price'].sum().round(2)
print(total_invoice)

dados_fatura['total'] = [total_invoice]
dados_fatura
```

✓ 0.0s

```
105.22
```

	store	date	invoice_number	total
0	CBD Marrazes	2019-05-22	ET INQ008/000001	105.22

Figura 30 – Código criado para o cálculo do total da fatura

Além de calcular o total da fatura, também é necessário transformar o campo correspondente à taxa do IVA. O valor extraído das faturas eletrónicas para esse campo pode ser “A”, B” ou “C”, consoante se a taxa do IVA daquele produto for 6%, 13% ou 23% respetivamente. Conforme o respetivo dicionário de dados (anexo III), esse campo deve ser numérico e com 2 casas decimais. É então necessário transformar esse campo de forma a substituir a letra *A* por *0.06*, a letra *B* por *0.13* e a letra *C* por *0.23*. Caso o valor do campo *tax* não seja nenhuma dessas letras, o mesmo será substituído por 0. Na Figura 31 é apresentado o código criado para realização desta operação.

```
# converter a letra da variavel tax para o respetivo numero
# (A:0.06, B:0.13, C:0.23)

for indice in range(0,len(lista_produtos.index)):
    if lista_produtos.tax[indice]=='A':
        lista_produtos.tax[indice]='0.06'
    elif lista_produtos.tax[indice]=='B':
        lista_produtos.tax[indice]='0.13'
    elif lista_produtos.tax[indice]=='C':
        lista_produtos.tax[indice]='0.23'
    else: lista_produtos.tax[indice]='0'
```

Figura 31 – Código criado para transformar o campo extraído da taxa do IVA num campo numérico

No final desta etapa, os dados presentes na BD *Pandas lista_produtos* já se encontram prontos a serem exportados para o SQL. Na Figura 32 apresenta-se um excerto da BD referente à extração dos dados da fatura exemplo.

```
lista_produtos.iloc[[0,1,2,3,4,51,52]]
✓ 0.0s
```

	category_name	item_description	tax	quantity	total_discount	total_price	unity_price
1	Mercearia Salgada	ATUM EM OLEO E CONTINENTE 110G	0.06	4.0	0.00	2.76	0.69
2	Mercearia Salgada	SALS FKF AVES IZIDOR LT 4P 350G	0.23	1.0	0.00	1.24	1.24
3	Mercearia Salgada	FEIJAO ENCARN CONTINENTE 830G	0.23	1.0	0.00	0.90	0.90
4	Mercearia Salgada	GRAO BICO CONTINENTE 830G	0.23	1.0	0.00	0.90	0.90
5	Mercearia Salgada	MILHO DOCE CONTINENTE 3*150G	0.23	1.0	0.00	1.29	1.29
52	DIY	CJ3 LAMP LED BAS CHA 5,9W 2700K	0.23	2.0	4.00	5.98	2.99
53	Petfood and Care	AL SECO P/CAO FRISKIES MINI MEN	0.23	1.0	4.36	6.53	6.53

Figura 32 – Tabela final típica obtida após extração dos dados de uma fatura eletrónica

Os dados acabados de extrair são, por fim, gravados em ficheiros do tipo CSV de forma a que possam ser importados pelo *script* que irá tratar do preenchimento da BD SQL (secção 4.4). Os ficheiros CSV são criados através do código da Figura 33.

```
1  #Grava a base de dados pandas num ficheiro CSV
2  lista_produtos.to_csv(r'C:/Users/micka/Dropbox/projeto/notebooks/listaProdutos.csv', index=False)
3  dados_fatura.to_csv(r'C:/Users/micka\Dropbox/projeto/notebooks/dadosFatura.csv', index=False)
✓ 0.0s
```

Figura 33 – Código criado para gravação dos dados extraídos das faturas eletrónicas em ficheiros CSV

4.3.2.8. IVA 0% bens essenciais

Com a entrada em vigor da Lei nº 17/2023 de 14 de abril, o formato das faturas eletrónicas do Continente sofreu alterações que obrigou a uma atualização do algoritmo. A referida Lei engloba uma medida excecional que prevê a isenção temporária do valor do IVA num conjunto de produtos alimentares de forma a fazer face ao aumento extraordinário dos preços desses produtos [40].

Essa Lei, nos dias de hoje, já não se encontra em vigor, no entanto, as faturas eletrónicas que contêm produtos com essa isenção de IVA passaram a ter o aspeto da Figura 34 (para o caso do produto ser comprado com quantidade diferente de 1 e possuir um desconto direto), ou da Figura 35 (para o caso do produto ser comprado com quantidade igual a 1 e não possuir desconto), ou da Figura 36 (no caso de ser adquirido 1 quantidade e existir desconto). Essa alteração nas faturas fez com que o algoritmo deixasse de funcionar, pelo que foi necessário acrescentar o código da Figura 37 ao algoritmo.

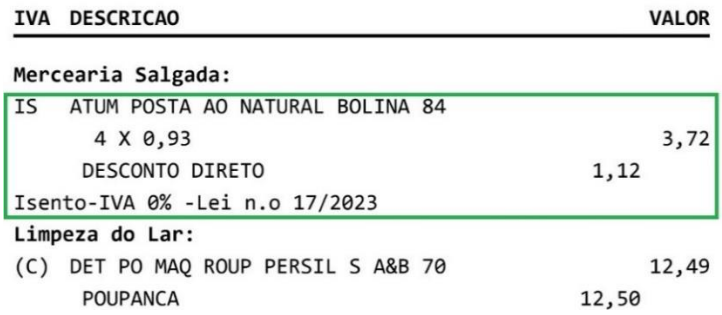


Figura 34 – Exemplo de uma fatura com isenção de IVA (com desconto e quantidade diferente de 1)

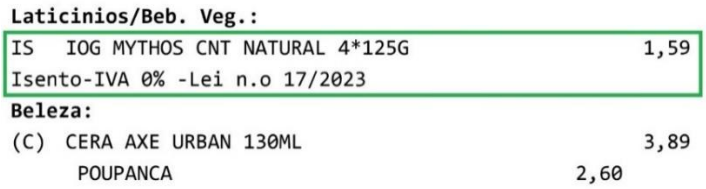


Figura 35 – Exemplo de uma fatura com isenção de IVA (sem desconto e quantidade igual a 1)

Figura 36 – Exemplo de uma fatura com isenção de IVA (com desconto e quantidade igual a 1)

```
#Indentificar as linhas com IVA 0 e aliminar as
#respetivas linhas desnecessárias
i=0
while i<len(text_splitted):
    if text_splitted[i].startswith("Isento-IVA"):
        if text_splitted[i-1].startswith("IS"):
            text_splitted[i-1] = "(D)" + text_splitted[i-1][2:]
            del text_splitted[i]
        elif text_splitted[i-2].startswith("IS"):
            text_splitted[i-2] = "(D)" + text_splitted[i-2][2:]
            del text_splitted[i]
        else:
            text_splitted[i-3] = "(D)" + text_splitted[i-3][2:]
            del text_splitted[i]
    else:
        i+=1
#text_splitted
```

Figura 37 – Código criado para identificar os produtos com isenção temporária de IVA

O código da Figura 37 procura pelas linhas que contenham a *string* "Isento-IVA", elimina essa linha por se tratar de uma informação não relevante, localiza a respetiva *string* "IS" (que

se pode encontrar 1 ou 2 linhas mais em cima) e substitui essa por "(D)", que será a identificação mais à frente do IVA a 0%.

Por fim, o algoritmo da Figura 31 foi alterado de forma a ficar como na Figura 38, para que a letra 'D' do campo *tax* possa ser transformada para o valor '0', correspondente ao IVA 0%. Se eventualmente o algoritmo não encontrar nenhuma das letras A, B, C ou D no campo *tax*, então armazena nesse campo o valor '-1', de forma a perceber que algo correu mal com a extração do valor do IVA.

```
# converter a letra da variavel tax para o respetivo numero
# (A:0.06, B:0.13, C:0.23, D:0)

for indice in range(0,len(lista_produtos.index)):
    if lista_produtos.tax[indice]=='A':
        lista_produtos.tax[indice]='0.06'
    elif lista_produtos.tax[indice]=='B':
        lista_produtos.tax[indice]='0.13'
    elif lista_produtos.tax[indice]=='C':
        lista_produtos.tax[indice]='0.23'
    elif lista_produtos.tax[indice]=='D':
        lista_produtos.tax[indice]='0'
    else: lista_produtos.tax[indice]='-1'
```

Figura 38 – Código alterado para poder identificar a letra 'D' como sendo IVA a 0%

Com esta alteração ao algoritmo, o mesmo voltou a funcionar corretamente.

4.4. Preenchimento automático das tabelas da base de dados

Nesta secção é explicado de que forma os dados extraídos na secção anterior foram escritos na BD.

Os dados recolhidos pelo intermédio do algoritmo explicado na secção anterior foram introduzidos na BD igualmente de forma automática através de um outro *script*, também esse desenvolvido em *Python*. Desta vez foram utilizadas as seguintes bibliotecas:

- *Pandas*: biblioteca *open source* para análise de dados [20] (já foi referida na secção 4.3);
- *Pyodbc*: biblioteca igualmente *open source* que facilita o acesso a bases de dados ODBC [41]. ODBC é a sigla de *Open Database Connectivity* e trata-se de uma *Application Programming Interface* (API) que permite estabelecer ligações entre uma aplicação e uma BD [42].

O algoritmo começa então por importar essas bibliotecas através do código ilustrado na Figura 39, bem como os 2 ficheiros CSV criados na secção anterior que contêm os dados a serem introduzidos na BD.

```
import pyodbc
import pandas as pd

listaProdutos=pd.read_csv(r'C:/Users/(...))/listaProdutos.csv')
dadosFatura=pd.read_csv(r'C:/Users/(...))/dadosFatura.csv')
```

Figura 39 – Código criado para importar as bibliotecas e os ficheiros CSV

De seguida, é feita a ligação à BD através do código da Figura 40, sendo *UID* e *PWD* respetivamente o nome de utilizador e a password de acesso à BD.

```
# Ligação à base de dados local
cnxn = pyodbc.connect('DRIVER={SQL Server Native Client 11.0};'
                      'SERVER=MICKAEL-PC\MSSQLSERVER2;'
                      'DATABASE=projeto;'
                      'UID=[redacted];'
                      'PWD=[redacted];')
cursor = cnxn.cursor()
```

Figura 40 – Código criado para efetuar ligação do *script* à DB SQL

Antes de iniciar a introdução dos dados, as linhas de código da Figura 41 protegem a BD contra a introdução de faturas duplicadas. Ao verificar que o número da fatura já existe na tabela *invoice*, então o algoritmo simplesmente imprime o texto "*Fatura já foi introduzida*", e não executa mais nenhum comando.

```
# Proteção para o caso da fatura selecionada já ter sido inserida:
query= "SELECT 1 FROM invoice WHERE invoice_number='"+dadosFatura.invoice_number[0]+"';"
#dados=cursor.execute(query)
dados = pd.read_sql(query, cnxn)
if dados.empty == 0: #Mudar aqui para 0 para que esta proteção possa funcionar
    print("Fatura já foi introduzida")
else:

```

Figura 41 – Código criado para a proteção da BD contra a introdução de faturas duplicadas

Caso a fatura ainda não tenha sido criada na BD, então é executado o código da Figura 42. Esse código executa a criação de uma nova linha da tabela *invoice* com os dados da fatura nova.

```
else:

    # Preenchimento da tabela invoice:
    count_invoice = cursor.execute("IF NOT EXISTS (SELECT 1 FROM invoice \
                                   WHERE invoice_number='"+dadosFatura.invoice_number[line]+"') \
                                    INSERT INTO invoice (store_id,date,total_invoice,invoice_number) \
                                    VALUES ((SELECT store_id \
                                            FROM store \
                                            WHERE store_name = '"+dadosFatura.store[line]+"'),\
                                   '"+dadosFatura.date[line]+"',\
                                   "+str(dadosFatura.total[line])+",\
                                   '"+dadosFatura.invoice_number[line]+"');").rowcount
    cnxn.commit()
```

Figura 42 – Código criado para preenchimento da tabela *invoice* na BD SQL

De seguida foi criado um ciclo de repetição *FOR* que introduz na BD SQL cada um dos dados referentes às linhas de venda, dados esses que se encontram agora na BD provisória *listaprodutos* (conforme criado com o código da Figura 39). Para cada linha de venda é realizado o seguinte processo:

- Criação da categoria (na tabela *category*), caso esta ainda não tenha sido criada (Figura 43);
- Criação do produto (na tabela *item*), caso este ainda não tenha sido criado, com ligação à respetiva categoria (igualmente Figura 43);
- Preenchimento das linhas de venda (na tabela *invoice_line*), com ligação à respetiva fatura (na tabela *invoice*) e aos respetivos produtos (na tabela *item*), através do código da Figura 44. Os dados inseridos são os seguintes:
    - tax;
    - quantity;
    - total_discount;
    - unity_price;
    - total_price.

No fim de ter percorrido todas as linhas de venda, o algoritmo apresenta as informações da Figura 45.

Os resultados apresentados na Figura 45 são referentes à fatura exemplo, sendo que todos os produtos e categorias nela contida já existiam na BD, pelo que não foi necessário criar de novo esses dados.

```
#introdução de cada linha de venda
for line in range(0, len(listaProdutos)):

    # Introduzir a categoria, caso essa não esteja já introduzida:
    count = cursor.execute("IF NOT EXISTS (SELECT 1 FROM category \
                           WHERE category_name='"+listaProdutos.category_name[line]+"') \
                           INSERT INTO category (category_name) \
                           VALUES ('"+listaProdutos.category_name[line]+"');").rowcount
    cnxn.commit()
    if count > 0:
        count_categories += count

    # introduzir o item, caso esse não esteja já introduzido:
    count = cursor.execute("IF NOT EXISTS (SELECT 1 FROM item \
                           WHERE item_description='"+listaProdutos.item_description[line]+"') \
                            INSERT INTO item (category_id,item_description) \
                            VALUES ((SELECT category_id \
                                FROM category \
                                WHERE category_name = '"+listaProdutos.category_name[line]+"'),\
                                    '"+listaProdutos.item_description[line]+"');").rowcount
    cnxn.commit()
    if count > 0:
        count_itens += count
```

Figura 43 – Código criado para inserção da categoria e do produto na BD SQL

```
    # preencher a tabela invoice_line:
    count = cursor.execute("INSERT INTO invoice_line (invoice_id,item_id,tax,quantity,\
                                                      total_discount,unity_price,total_price) \
                           VALUES ((SELECT invoice_id \
                                    FROM invoice \
                                    WHERE invoice_number = '"+dadosFatura.invoice_number[0]+"'),\
                           (SELECT item_id \
                                    FROM item \
                                    WHERE item_description = '"+listaProdutos.item_description[line]+"'),\
                           '"+str(listaProdutos.tax[line])+"',\
                           '"+str(listaProdutos.quantity[line])+"',\
                           '"+str(listaProdutos.total_discount[line])+"',\
                           '"+str(listaProdutos.unity_price[line])+"',\
                           '"+str(listaProdutos.total_price[line])+"');").rowcount
    cnxn.commit()
    if count > 0:
        count_invoice_line += count
```

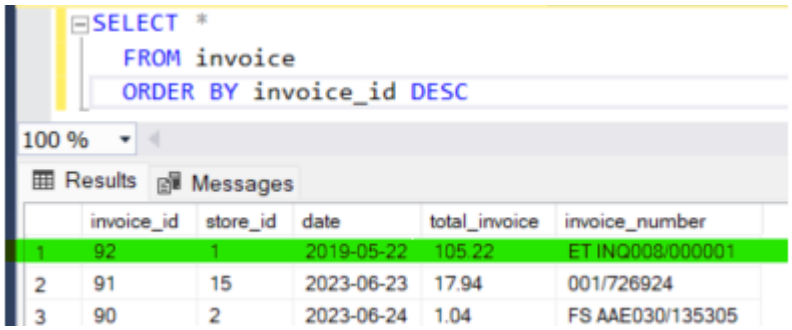
Figura 44 – Código criado para inserção dos dados referentes às linhas de venda na BD SQL

```
    print('fatura(s) criada(s): ' + str(count_invoice))
    print('Novos produtos introduzidos: ' + str(count_itens))
    print('Novas categorias introduzidas: ' + str(count_categories))
    print('Qt. de linhas de vendas criadas: ' + str(count_invoice_line))


fatura(s) criada(s): 1
Novos produtos introduzidos: 0
Novas categorias introduzidas: 0
Qt. de linhas de vendas criadas: 53
```

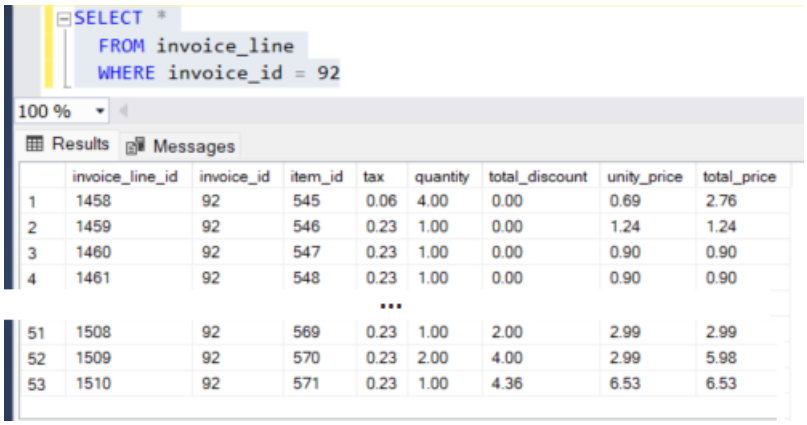
Figura 45 – Output obtido pelo algoritmo no fim de introduzir os dados das linhas de venda na BD SQL

É possível verificar a correta importação dos dados através de alguns comandos SQL introduzidos diretamente no *Microsoft SSMS*. Na Figura 46 verifica-se a correta criação da fatura na tabela *invoice*, enquanto que na Figura 47 encontra-se ilustrado parte das linhas de venda criadas na tabela *invoice_line*.



	invoice_id	store_id	date	total_invoice	invoice_number
1	92	1	2019-05-22	105.22	ET INQ008/000001
2	91	15	2023-06-23	17.94	001/726924
3	90	2	2023-06-24	1.04	FS AAE030/135305

Figura 46 – Verificação da criação da fatura exemplo na tabela *invoice* da BD SQL



	invoice_line_id	invoice_id	item_id	tax	quantity	total_discount	unity_price	total_price
1	1458	92	545	0.06	4.00	0.00	0.69	2.76
2	1459	92	546	0.23	1.00	0.00	1.24	1.24
3	1460	92	547	0.23	1.00	0.00	0.90	0.90
4	1461	92	548	0.23	1.00	0.00	0.90	0.90
...								
51	1508	92	569	0.23	1.00	2.00	2.99	2.99
52	1509	92	570	0.23	2.00	4.00	2.99	5.98
53	1510	92	571	0.23	1.00	4.36	6.53	6.53

Figura 47 – Verificação da criação das linhas de venda na tabela *invoice_line* da BD SQL

O objetivo da tabela *county*, por sua vez, é existir na BD uma lista com todos os Concelhos de Portugal, incluindo ilhas, e com a identificação dos respetivos Distritos. Ligar as lojas a essa tabela pode permitir filtrar os resultados apresentados nos *dashboards* por Concelhos ou por Distritos. Essa filtragem não é realizada neste projeto, já que a grande maioria das compras foram realizadas no Concelho de Leiria, mas assim a BD já está preparada para permitir análises por Concelhos ou por Distritos, caso essa venha a expandir-se nesse sentido.

O preenchimento da tabela *county* foi realizado através da importação direta de um ficheiro *Excel*. Este ficheiro, originalmente adquirido através do Portal das Finanças [43], contém a lista completa de todas as Freguesias de Portugal, bem como os respetivos Concelhos e Distritos, entre outras informações. Antes de ser importado para a BD, o ficheiro foi trabalhado, de forma a que nele ficassem somente os dados relevantes para o projeto (neste caso, os Concelhos e Distritos). Esta importação resultou numa tabela com 308 linhas correspondentes aos 308 Concelhos de Portugal e com os respetivos Distritos identificados. Na Figura 48 encontra-se ilustrada uma amostra dessa tabela.

```
SELECT *
  FROM county
```

100 %

Results | Messages

	county_id	county_name	district
131	131	vila nova de foz coa	guarda
132	132	alcobaça	leiria
133	133	alvaiazere	leiria
134	134	ansião	leiria
135	135	batalha	leiria
136	136	bombarral	leiria
137	137	caldas da rainha	leiria
138	138	castanheira de pera	leiria
139	139	figueiro dos vinhos	leiria
140	140	leiria	leiria
141	141	marinha grande	leiria
142	142	nazare	leiria
143	143	obidos	leiria
144	144	pedrogão grande	leiria
145	145	peniche	leiria
146	146	pombal	leiria
147	147	porto de mos	leiria
148	148	alenquer	lisboa
149	149	arruda dos vinhos	lisboa

Figura 48 – Amostra da tabela *county* da BD, obtido diretamente a partir do SSMS

A tabela *store* da BD permite identificar não só em que cidade a loja se encontra, (incluindo o respetivo Concelho através do relacionamento com a tabela *county*), mas também a natureza do estabelecimento (supermercado, minimercado, farmácia, frutaria, entre outros). Trata-se de informações que não são muito claras nas faturas eletrónicas, impossibilitando esta tabela de ser preenchida de forma automática, como acontece com as restantes. Por isso, foi necessário preencher a tabela *store* de forma manual, diretamente com comandos a partir do SSMS. Na Figura 49 encontra-se ilustrada a *query* utilizada para criar a loja da fatura exemplo, sendo que para as restantes lojas, apenas foi necessário alterar os campos a vermelho.

```
/*CBD Marrazes*/
IF not exists (SELECT 1 FROM store WHERE store_name = 'CBD Marrazes')
INSERT INTO store (county_id,store_name,"type","location")
VALUES  ((SELECT county_id FROM county WHERE county_name = 'leiria'),
         'CBD Marrazes',
         'supermercado',
         'Marrazes');
```

Figura 49 – *Query* SQL para criar manualmente a loja "CBD Marrazes" na tabela *store* da BD

4.5. Data Warehouse

A BD criada na secção anterior possui os dados sobre a maioria das compras realizadas desde 2019 até ao início de 2024. A próxima etapa do projeto passa agora pela criação do DW. Nesta secção é explicado como o DW foi implementado, desde a criação da tabela de factos

e dimensões até à criação das tabelas no SSMS. A BD criada na secção anterior será doravante denominada BD original.

Para o presente projeto, foi identificado uma tabela de factos com as métricas relativas às despesas, bem como 4 dimensões. Na Tabela 1 estão identificados os campos que são necessários na tabela de factos, bem como as dimensões com as quais esses campos se interligam.

Tabela 1 – Bus Matrix – Identificação da tabela de factos e dimensões

	Factos	**Dimensões**			
		store	item	county	time
Despesa	unity_price	**1**	**1**	**1**	**1**
	total_discount	**1**	**1**		**1**
	Tax		**1**		**1**
	Quantity	**1**	**1**	**1**	**1**

o número 1 indica que existe relacionamento entre a dimensão e a medida

Para a tabela de factos e para cada dimensão foi criada uma tabela no DW. Tudo começou com a elaboração de uma lista dos atributos que irão compor cada uma das tabelas. De seguida apresentam-se os atributos das dimensões *store* (Tabela 2), *item* (Tabela 3) e *county* (Tabela 4), bem como a identificação da origem de cada um deles.

Tabela 2 – Descrição dos atributos da dimensão *store*

		Tabela/Atributo Original		
Nome	**Descrição**	**Tabela**	**Atributo**	**Relacionamento**
id_store	id original da loja	store	store_id	
store_name	nome da loja	store	store_name	
type	tipo de loja	store	Type	

Tabela 3 – Descrição dos atributos da dimensão *item*

		Tabela/Atributo Original		
Nome	**Descrição**	**Tabela**	**Atributo**	**Relacionamento**
id_item	id original do produto	item	item_id	
item_description	designação original do produto	item	item_description	
category_name	designação da categoria do produto	category	category_name	item.category_id = category.category_id

Tabela 4 – Descrição dos atributos da dimensão *county*

		Tabela/Atributo Original		
Nome	**Descrição**	**Tabela**	**Atributo**	**Relacionamento**
id_county	id do local	county	county_id	
county_name	designação do Concelho	county	county_name	
district	designação do Distrito	county	District	

A dimensão *time* (Tabela 5), por sua vez, tem um conceito um pouco diferente das restantes dimensões. A principal função da dimensão *time* é poder fornecer informações detalhadas sobre as datas, bem como efetuar pesquisas e apresentar resultados com base em intervalos temporais. Por exemplo, ao saber a data de uma determinada fatura, é possível saber, consultando a dimensão *time*, em que dia da semana foi efetuada essa compra, em que semestre ou trimestre do ano se encontra essa data, se aquele dia em específico era ou não um dia útil, entre outros.

Tabela 5 – Descrição dos atributos da dimensão *time*

		Tabela/Atributo Original		
Nome	**Descrição**	**Tabela**	**Atributo**	**Relacionamento**
id_data_key	Chave da data			
dia_da_semana	Dia da semana			
semana	Nº da semana			
mes	Nº do Mês			
mês_extenso	Mês escrito por extenso			
trimestre	Trimestre			
trimestre_extenso	Trimestre extenso			
semestre	Semestre			
semestre_extenso	Semestre extenso			
ano	Ano			
diaUtil	Se é ou não dia útil			

Por isso, ao contrário das restantes dimensões, a dimensão *time* não foi buscar os seus dados à BD original, mas sim a um ficheiro *Excel* devidamente preparado para o efeito. É por essa razão que os atributos da dimensão *time* não se encontram relacionados com nenhum campo da BD original. O único relacionamento que existe na dimensão *time* é o que interliga a mesma com a tabela de factos (através da PK *data_key*).

A tabela de factos, por sua vez, vai buscar os seus dados maioritariamente à tabela *invoice_line* da BD original. Os seus atributos são apresentados na Tabela 6. As tabelas do DW foram criadas no SSMS, através do mesmo procedimento utilizado na criação das tabelas da BD original, isto é, após criar as tabelas em si e atribuir-lhes os respetivos nomes, foram criadas as colunas com auxílio da opção *Design*. De seguida foram criados os respetivos relacionamentos entre tabelas que permitiram gerar o modelo da Figura 50, através de *reverse engineering*.

Tabela 6 – Descrição dos atributos da tabela de factos

		Tabela/Atributo Original		
Nome	**Descrição**	**Tabela**	**Atributo**	**Relacionamento**
id_item	id original do produto	item	item_id	
id_store	id original da loja	store	store_id	
id_county	id original do Concelho	county	county_id	
data_key	data original	invoice	date	
id_invoice_line	id original linha de venda	invoice_line	invoice_line_id	
unity_price	Preço unitário do produto	invoice_line	unity_price	
total_discount	Desconto aplicado	invoice_line	total_discount	
Tax	IVA do produto	invoice_line	tax	
Quantity	Quantidade de produto	invoice_line	quantity	

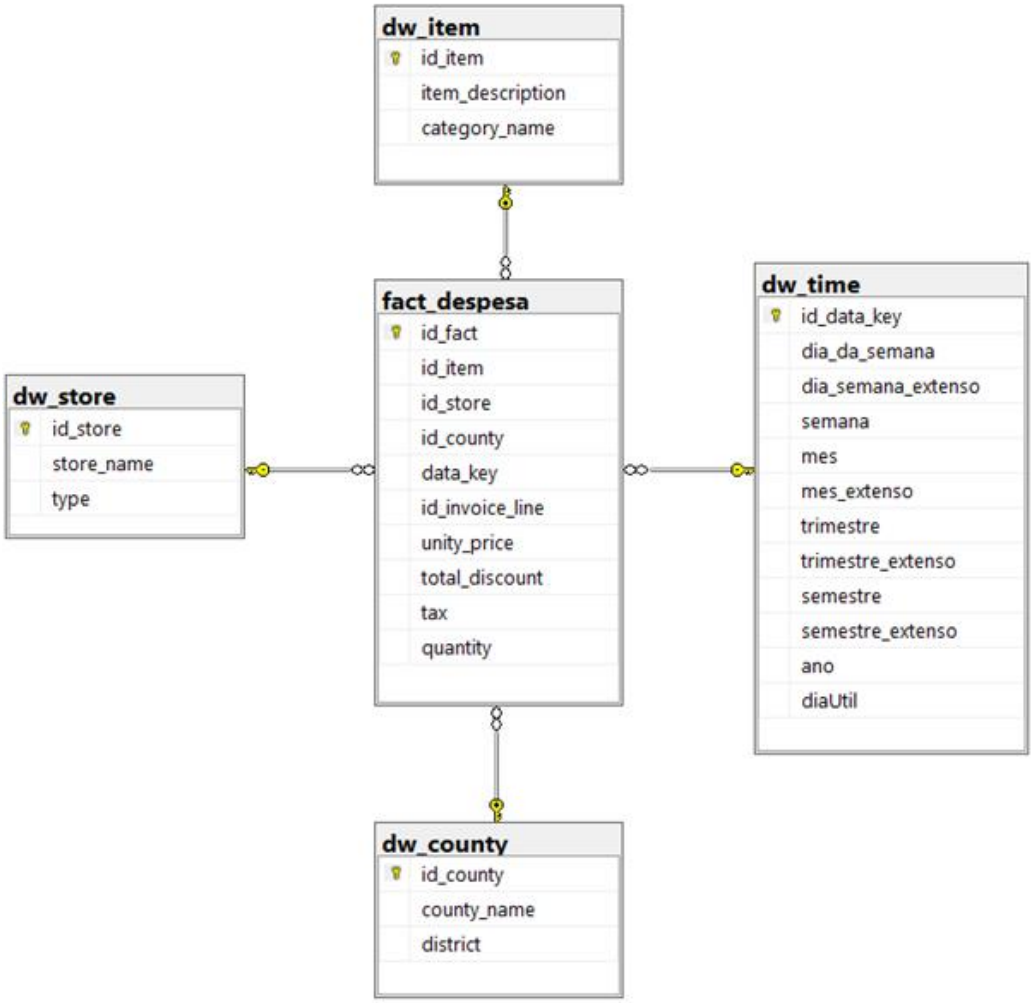


Figura 50 – Modelo do DW gerado através de *Reverse Engineering* do SSMS

4.6. População do Data *Warehouse* – Processo ETL

O DW encontra-se agora criado com as suas tabelas e configurações. O próximo passo é então preencher essas tabelas com os dados da BD original. Esse processo tem o nome de

ETL, e foi realizado de forma automática através de uma solução criada na ferramenta *Visual Studio* da *Microsoft* e a extensão *SQL Server Integration Services*.

Para preencher as tabelas das dimensões *county*, *item* e *store* e também a tabela de factos, a solução criada efetua uma ligação com a BD original para extrair dela os dados necessários, e outra ligação com o DW para carregar nessas tabelas os dados extraídos. O procedimento só foi diferente para a dimensão *time*, pois o sistema vai preencher esta com dados extraídos não da BD original, mas sim de um ficheiro *Excel* criado para o efeito. A esse ficheiro foi igualmente criada uma ligação para que o sistema pudesse aceder ao mesmo.

A vista geral (*control flow*) da solução do *Visual Studio* ficou com o aspeto apresentado na Figura 51. Na parte inferior dessa figura encontram-se as referidas ligações.

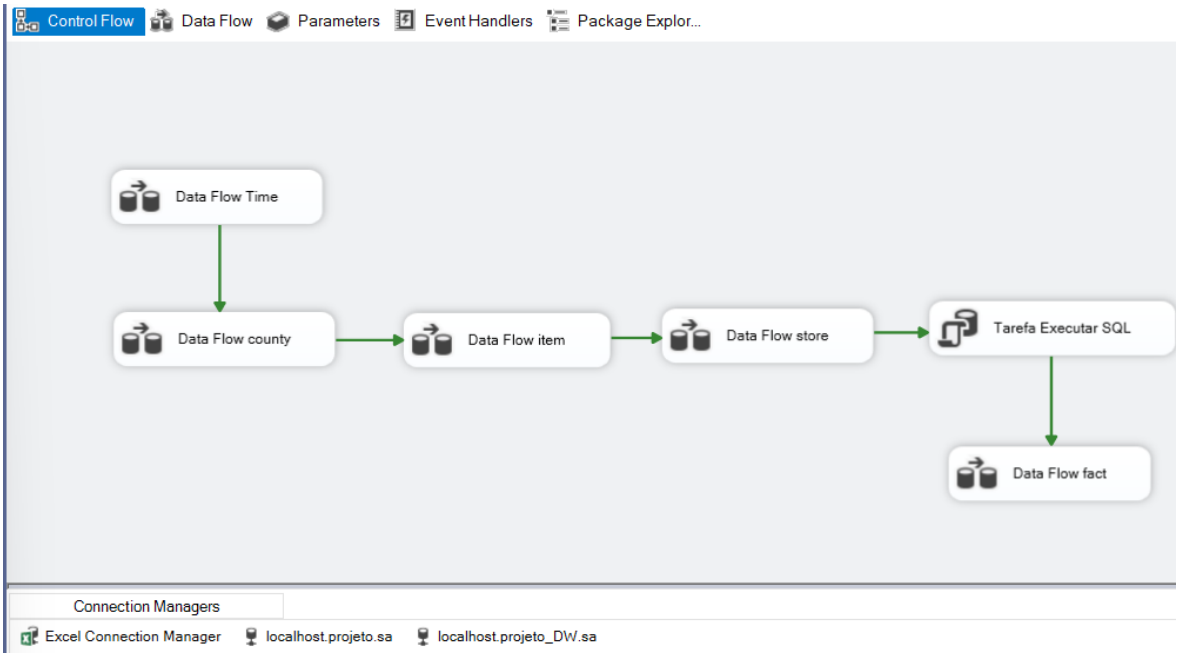


Figura 51 – *Control Flow* da solução criada em *Visual Studio*

O processo ETL foi realizado com auxílio de um bloco *Data Flow* para cada uma das tabelas do DW.

Os dados necessários para o preenchimento da tabela da dimensão *time* encontram-se num ficheiro *Excel* criado para o efeito. Esse ficheiro possui os seguintes campos referentes a todos os dias entre 1 de janeiro 2019 e 31 de dezembro 2039:

- Data (no formato dd/mm/aaaa);
- Número do dia da semana (“1” para domingo, “2” para segunda-feira, etc.);
- Dia da semana escrito por extenso;

- Número da semana;
- Número do mês;
- Mês escrito por extenso;
- Número do trimestre;
- Trimestre escrito por extenso;
- Número do semestre;
- Semestre escrito por extenso;
- Ano;
- Dia útil (“sim se for um dia útil, “não” se não for um dia útil).

A Figura 52 ilustra os blocos que foram criados no seio do *Data Flow time*.

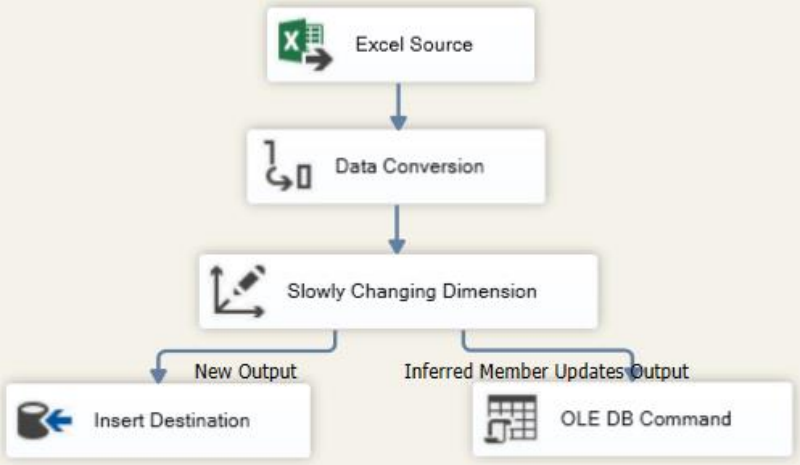


Figura 52 – Blocos criados no *Data Flow time* para o seu processo ETL

O bloco *Excel Source* trata de extrair todos os dados do ficheiro *Excel*, enquanto que o bloco *Data Conversion* converte cada um dos campos extraídos para o formato das respetivas colunas destino da tabela do DW. O bloco *Slowly Changing Dimension*, de uma forma genérica, é um bloco que permite atualizar os campos das dimensões de um DW. Ele tanto atualiza os campos existentes como cria campos novos quando esses ainda não existem. Por fim, o bloco *Insert Destination* foi configurado para inserir novas linhas na dimensão *time*, enquanto que o bloco *OLE DB Command* trata de atualizar as linhas existentes.

Os blocos *Data Flow county*, *Data Flow item* e *Data Flow store* (da Figura 51) são idênticos em termos de processo. Os dados a transferir para as dimensões *county*, *item* e *store* já se encontram devidamente formatados, não existindo assim a etapa da Transformação para esses. Isso faz com que a única diferença entre esses 3 blocos seja os mapeamentos entre as tabelas origem e destino. A Figura 53 mostra o pormenor dos 3 blocos *Data Flow* referidos.

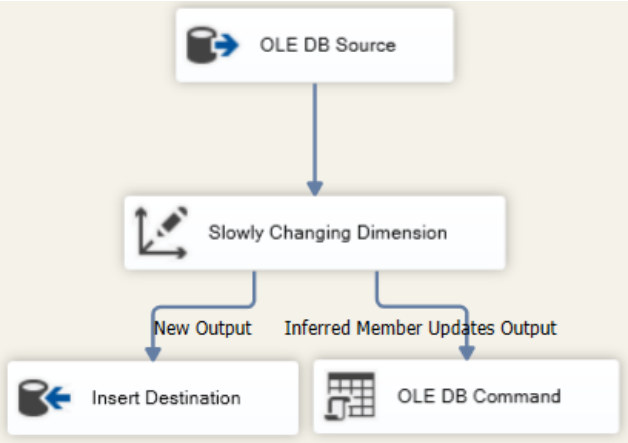


Figura 53 – Blocos criados no *Data Flow time, item* e *loja* para o seu processo ETL

O bloco *Data Flow fact*, por sua vez, foi configurado para preencher a tabela de factos. Neste caso, como se trata de uma cópia direta dos dados da BD original, os únicos blocos que existem são os mapeamentos entre campos de origem e de destino. A Figura 54 mostra a constituição interna do *Data Flow fact*.

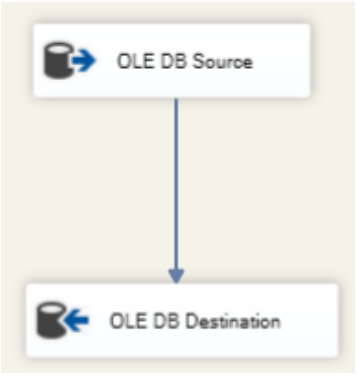


Figura 54 – Blocos criados no *Data Flow fact* para o seu processo ETL

É no bloco *OLE DB Source* onde se construiu a *query* utilizada para extrair os dados da BD original. Essa *query* foi construída visualmente, através da seleção dos campos pretendidos, conforme pode ser visualizado na Figura 55. Depois foi apenas necessário atribuir novos nomes a alguns campos e, com o auxílio dos relacionamentos da própria BD original, o sistema criou automaticamente a *query* completa apresentada na Figura 56.

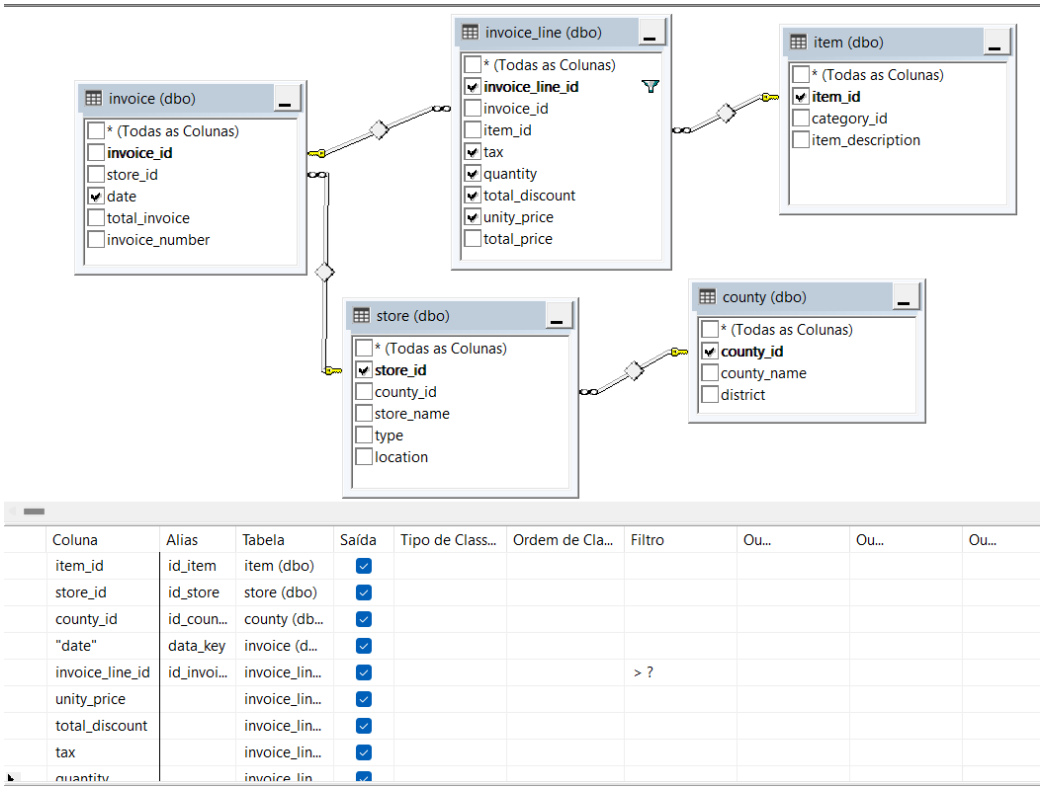


Figura 55 – Construção da *query* para a extração dos dados da BD original para a tabela de factos

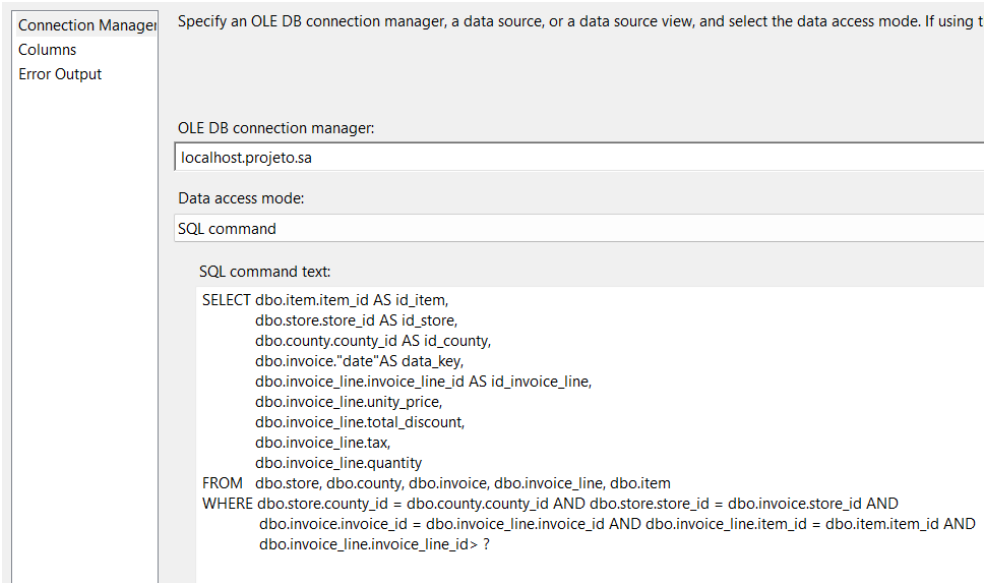


Figura 56 – *Query* gerada automaticamente pelo construtor do bloco *OLE DB Source*

Por fim, ainda existe o bloco *Tarefa Executar SQL* (Figura 51) que é responsável por manter o rasto da última linha de venda introduzida no DW. Ao acrescentar dados na tabela de factos, não é necessário ir buscar à BD original todos os dados que já foram introduzidos. Por isso, foi criado esse bloco que, através da *query* da Figura 57, vai determinar qual o *id* da última linha de venda existente na tabela de factos e guardar essa informação na variável *LAST_DESPESA*. Essa variável foi introduzida na *query* da Figura 56, de forma a que apenas

sejam extraídos os dados da BD original que tenham um *id* superior a essa, evitando assim extrair e processar dados já introduzidos.

```
SELECT
ISNULL(MAX(id_invoice_line), 0) AS LAST_DESPESA
FROM dbo.fact_despesa
```

Figura 57 – *Query* para determinar qual a última linha de venda introduzida no DW

Com esta solução de ETL, as tabelas dimensões tanto podem receber novas linhas como atualizar linhas existentes, enquanto que na tabela de factos apenas é possível acrescentar novas linhas. Após a execução da solução ETL, obtêm-se como *output* a mensagem apresentada na Figura 58, da qual se destacam as 53 linhas escritas no DW correspondentes às 53 linhas de venda introduzidas naquela execução em específico.

```
Information: 0x4004300B at Data Flow fact, SSIS.Pipeline: "OLE DB Destination" wrote 53 rows.
Information: 0x40043009 at Data Flow fact, SSIS.Pipeline: Cleanup phase is beginning.
SSIS package "C:\Users\micka\Dropbox\projeto\SSIS20-07-2023\projeto_SSIS_DW\projeto_SSIS_DW\Package.dtsx" finished: Success.
O programa "[18964] DtsDebugHost.exe: DTS" foi fechado com o código 0 (0x0).
```

Figura 58 – *Output* obtido após uma execução do processo ETL

4.7. *Dashboard*

Ao longo desta secção são apresentados os gráficos e as tabelas criadas a partir dos dados importados para o DW (Secção 4.5). Além disso, também são procuradas nesses gráficos as respostas às perguntas que deram origem a este projeto, perguntas essas listadas na secção 3.1.

Em primeiro lugar, foi necessário criar um novo modelo no *Power BI Desktop* e ligar o mesmo aos dados do DW. Essa operação foi realizada de forma simples através da opção *Get Data / Analysis Services*. O correto acesso aos dados foi verificado através da geração automática do modelo de dados apresentado na Figura 59.

Além dos dados existentes no DW, foi necessário criar algumas medidas. Trata-se de campos calculados com base nos dados e que permitam determinar, por exemplo, o total gasto por categoria de produto ou por loja. As medidas foram criadas com o auxílio de expressões DAX (*Data Analysis Expressions*). DAX é um conjunto de funções com sintaxe própria que permite calcular novas informações com base nos dados já existentes no modelo [44].

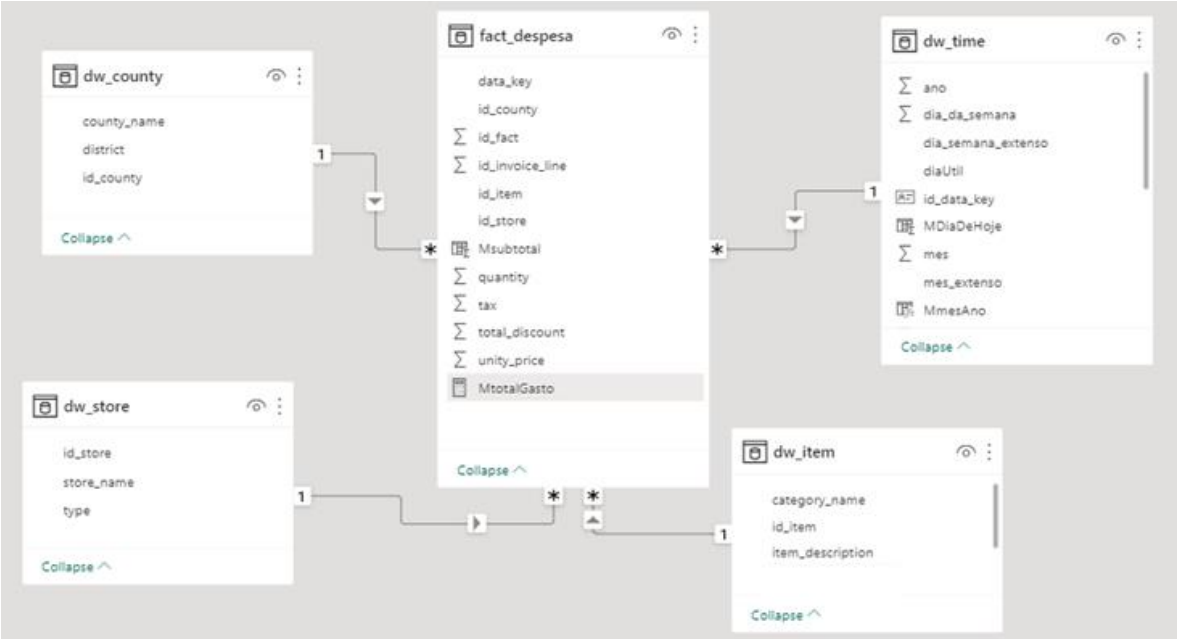


Figura 59 – Modelo de dados gerados a partir do *Power BI Desktop*

As medidas criadas para este projeto bem como as respetivas fórmulas DAX foram as seguintes:

MCategoria: Campo com a lista de todas as categorias existentes.

```
MCategoria = DISTINCT(dw_item[category_name])
```

MLojas: Campo com a lista de todas as lojas existentes.

```
MLojas = DISTINCT(dw_store[store_name])
```

MMediaLoja: Cálculo do preço médio de um determinado produto numa determinada loja.

```
MMediaLoja = CALCULATE(AVERAGE(fact_despesa[unity_price]),
ALLEXCEPT(dw_store,dw_store[store_name]))
```

MMediaPrecoUnitario: Cálculo do preço médio de um determinado produto.

```
MMediaPrecoUnitario = CALCULATE(AVERAGE(fact_despesa[unity_price]),
ALLEXCEPT(dw_item,dw_item[item_description]))
```

MTotalCat: Cálculo do total gasto por categoria de produto.

```
MTotalCat = CALCULATE(SUM(fact_despesa[Msubtotal]),
ALLEXCEPT(dw_item,dw_item[category_name]))
```

MTotalLoja: Cálculo do total gasto por loja.

```
MTotalLoja = CALCULATE(SUM(fact_despesa[Msubtotal]),
ALLEXCEPT(dw_store,dw_store[store_name]))
```

MTotalProduto: Cálculo do total gasto por produto.

```
MTotalProduto = CALCULATE(SUM(fact_despesa[Msubtotal]),
ALLEXCEPT(dw_item,dw_item[item_description]))
```

MTotalProdutoQuant: Cálculo do total da quantidade de um dado produto comprado.

```
MTotalProdutoQuant = CALCULATE(SUM(fact_despesa[quantity]),
ALLEXCEPT(dw_item,dw_item[item_description]))
```

MGastosTotais: Cálculo do total gasto.

```
MGastosTotais = SUMX('fact_despesa', 'fact_despesa'[Msubtotal])
```

MGastosPassado: Cálculo do total gasto no ano anterior em relação a um ano de referência.

```
MGastosPassado = CALCULATE(SUMX('fact_despesa','fact_despesa'[Msubtotal]),
DATESINPERIOD(dw_time[id_data_key],
DATE(YEAR(MAX(dw_time[id_data_key])), 1, 1)-1, -1, YEAR))
```

TaxaCrescimentoAno: Cálculo da taxa de crescimento da despesa total num dado ano em relação ao ano anterior.

```
TaxaCrescimentoAno = CALCULATE(1-[MGastosPassado]/[MGastosTotais],
FILTER(Medidas, ISNUMBER([MGastosPassado])))
```

KPITaxaCrescimentoAno: *Key Performance Indicator* (KPI) que indica se num dado ano a despesa total foi mais elevada ou mais baixa em relação ao ano anterior.

```
KPITaxaCrescimentoAno = IF([TaxaCrescimentoAno]>0, 1, -1)
```

O próximo passo consistiu em criar os gráficos e as tabelas para apresentar os dados. Foram criadas 7 páginas para poder apresentar os dados de forma a conseguir responder especificamente às 7 US inicialmente identificadas. Estas 7 páginas são apresentadas nas subsecções que se seguem.

4.7.1. Produtos mais comprados

A primeira página mostra a tabela com a lista dos produtos mais comprados em termos de quantidade. Ao selecionar um produto (ou mais) nessa tabela, o gráfico apresenta a respetiva quantidade comprada repartida pelo tempo. No caso do leite (produto comprado com maior quantidade), conforme apresentado na Figura 60, é possível verificar que é um produto comprado regularmente ao longo do ano, podendo existir uns picos mais elevados, mas sem conseguir identificar alguma tendência.

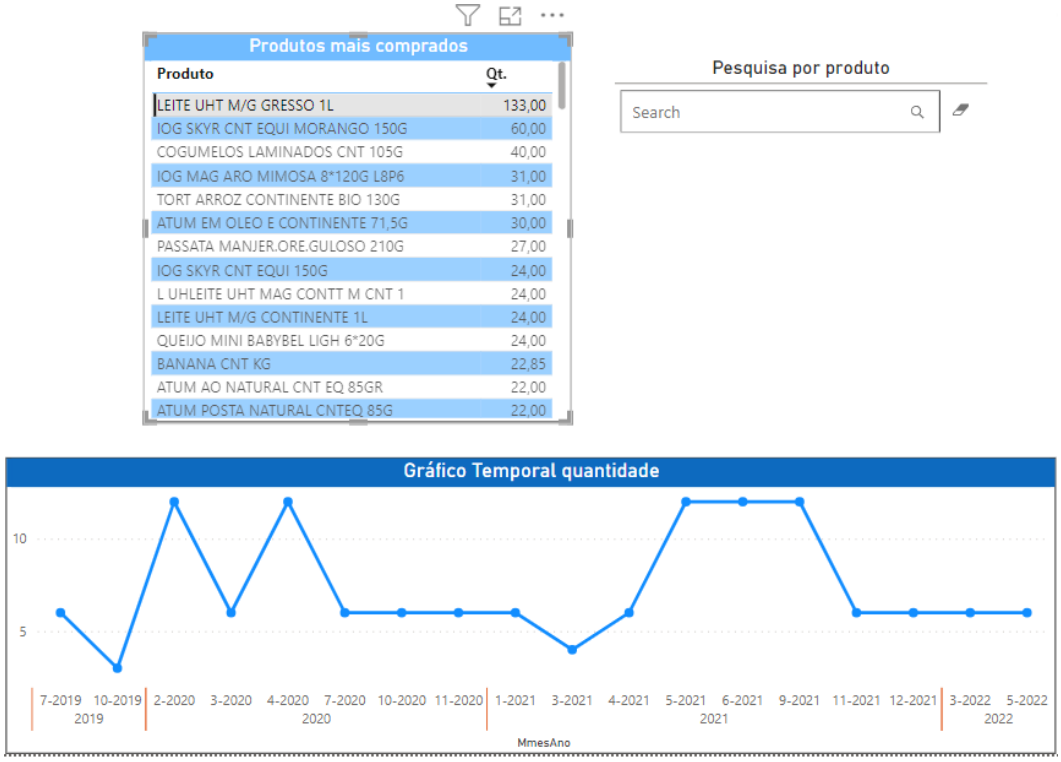


Figura 60 – Página 1 do *dashboard*: Produtos mais comprados

4.7.2. Produtos mais dispendiosos

Nesta segunda página do *dashboard* é possível verificar quais são os produtos que causam as maiores despesas. A página é apresentada na Figura 61, e nela pode verificar-se que o leite, por ser o produto mais comprado em termos de quantidade (conclusão retirada do gráfico anterior), também se encontra nesta lista, representando 3,36% da despesa total.

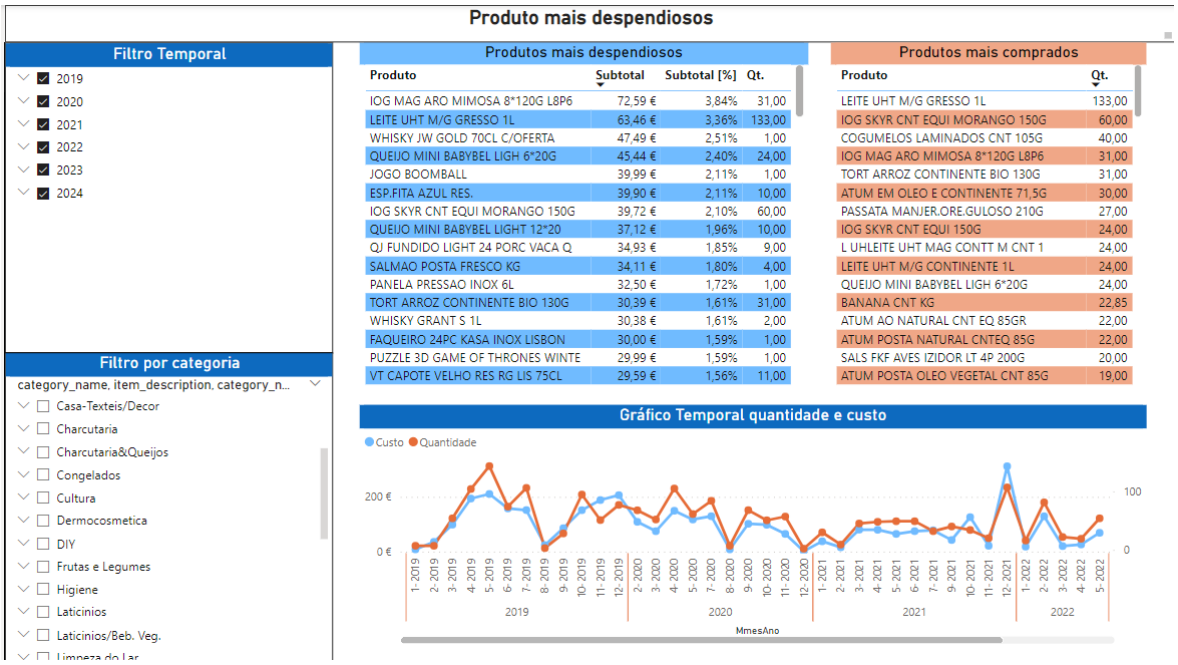


Figura 61 – Página 2 do *dashboard*: Produtos mais dispendiosos

Esta lista é liderada pelo iogurte "*IOG MAG ARO MIMOSA 8*120G L8P6*", que, no entanto, possui uma quantidade comprada muito inferior ao leite. Ao selecionar esse produto na lista, é gerado o gráfico temporal apresentado na Figura 62, e do qual se pode notar que já existe uma diminuição na compra deste produto. Existem muitos outros produtos no mercado equivalentes a este e com valor mais baixo, possibilitando assim uma poupança a longo prazo se substituir esse produto por outro de valor mais baixo.

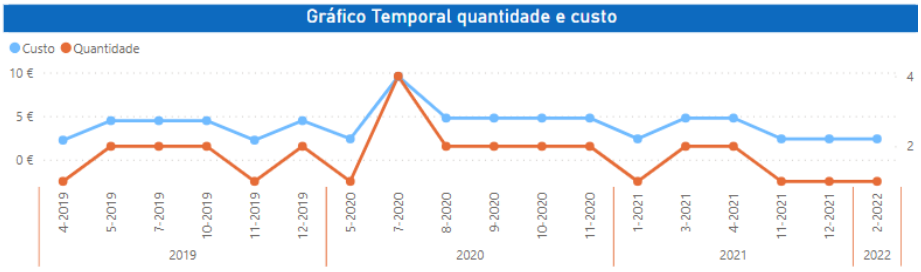


Figura 62 – Gráfico temporal do produto que gerou a maior despesa

Também são de destacar alguns produtos que foram comprados somente uma vez, mas que mesmo assim encontram-se na lista dos produtos que causam as maiores despesas. É exemplo disso o produto "*WHISKY JW GOLD 70CL*" que, apesar de ter sido comprado somente uma vez, encontra-se no terceiro lugar da lista representando 2,51% da despesa total.

4.7.3. Preços unitários dos produtos

A terceira página do *dashboard* permite não só listar os produtos cujo preço médio é mais elevado, como permite analisar a evolução do preço unitário de qualquer produto ao longo do tempo. A Figura 63 mostra esta lista sem filtros aplicados, a partir da qual se pode naturalmente concluir que os produtos com preço unitário médio mais elevados são os que são menos vezes comprados (neste caso, foram comprados só uma vez).

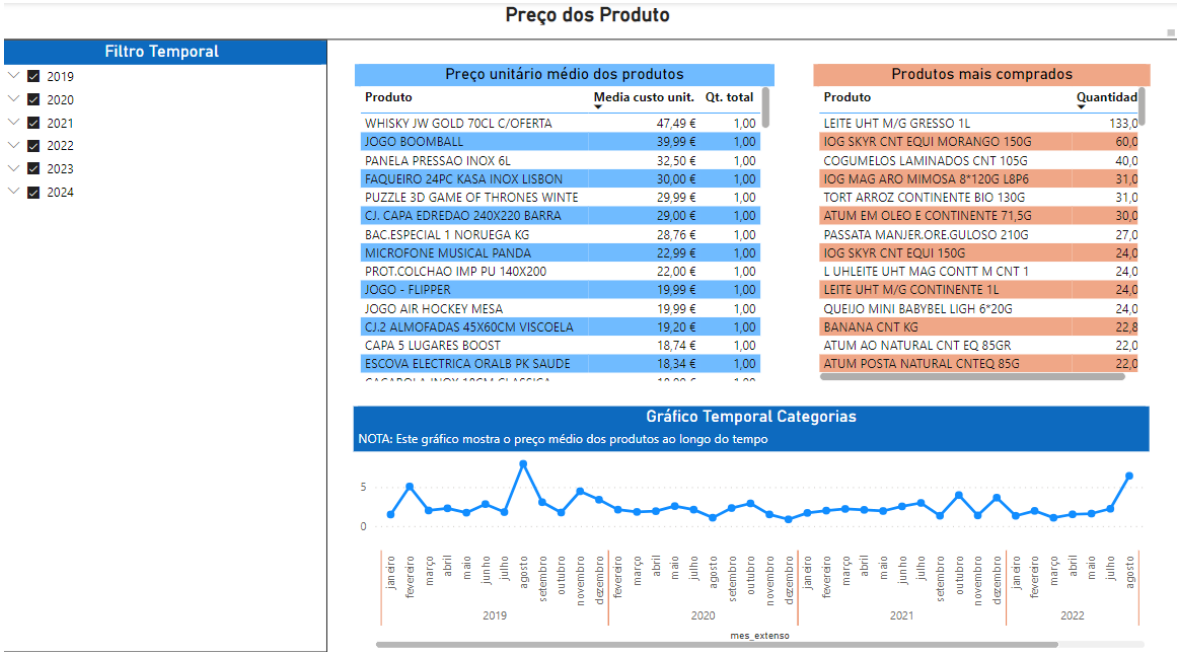


Figura 63 – Página 3 do *dashboard*: Preços dos produtos

No caso do leite, que é o produto mais adquirido em termos de quantidade, é possível ver através da Figura 64 que entre 2019 e 2022 não existe grande variação do seu preço médio.

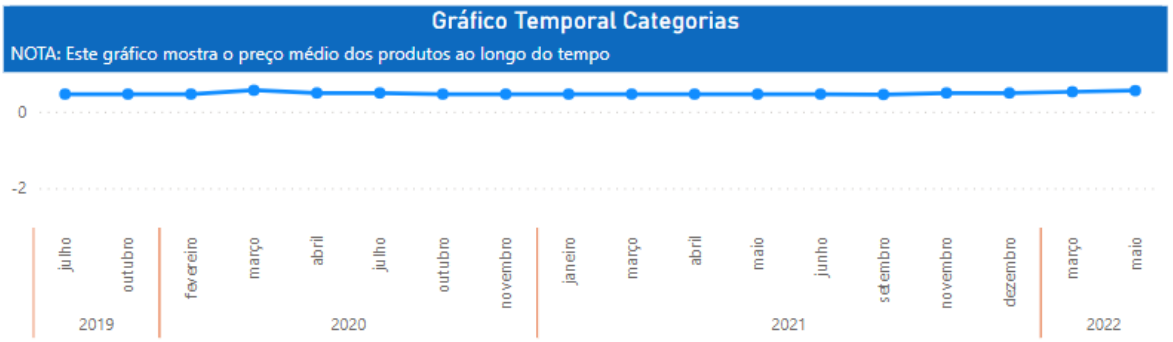


Figura 64 – Gráfico temporal com a evolução do preço médio do produto mais adquirido

4.7.4. Preços médios dos produtos por loja

Esta página, apresentada na Figura 65, permite comparar os preços dos mesmos produtos nas diversas lojas através do gráfico temporal. Além disso, a tabela situada do lado direito faz o resumo do total gasto por loja. Ao selecionar um ou vários produtos em específico, essa tabela apresenta o total gasto por loja para esse(s) produto(s).

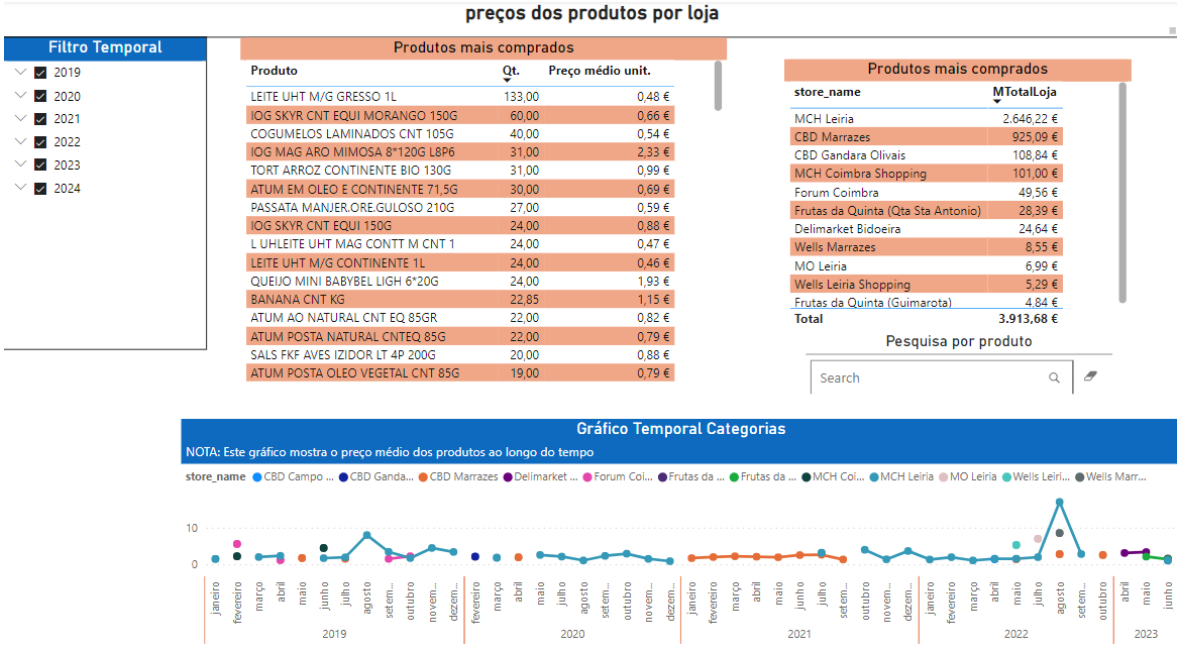


Figura 65 – Página 4 do *dashboard*: Preços dos produtos por loja

Na Figura 66 encontra-se o gráfico temporal quando se seleciona somente o leite. Trata-se de um produto que apenas foi adquirido em 3 lojas: "*CBD Marrazes*", "*MCH Leiria*" e "*CBD Gândara*", lojas essas todas pertencentes ao grupo Continente. O gráfico em si permite afirmar que o preço médio deste produto não difere entre essas 3 lojas. Essa situação verificou-se de igual forma com os restantes produtos que se encontram no topo da lista dos produtos mais adquiridos, pelo que é possível concluir que os preços não variam entre as diferentes lojas desta cadeia de supermercados.

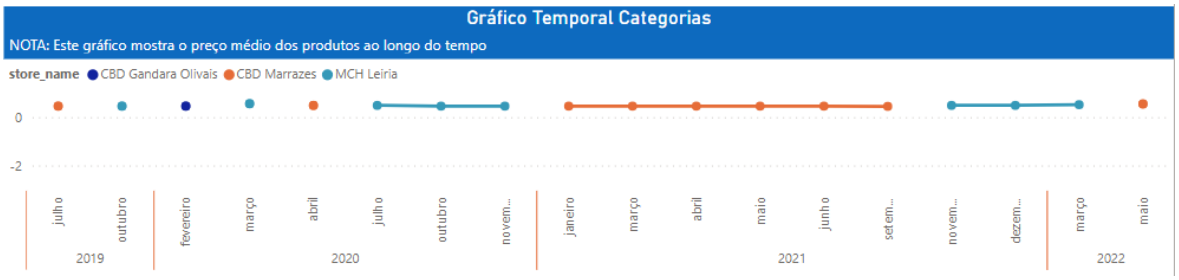


Figura 66 – Gráfico temporal com evolução do preço médio, em cada loja, do produto mais adquirido

4.7.5. Despesa por categoria

Através dos gráficos desta página, é possível perceber quais são as categorias de produtos onde se gastou mais dinheiro. Conforme a Figura 67, a categoria que originou a maior despesa foi *Frutas e Legumes* com 11,03% da despesa total, seguido de muito perto pela categoria *Mercearia Salgada* que representa 10,19% da despesa total. Com isto, conclui-se que a maior despesa é originada pelos bens essenciais do dia a dia.

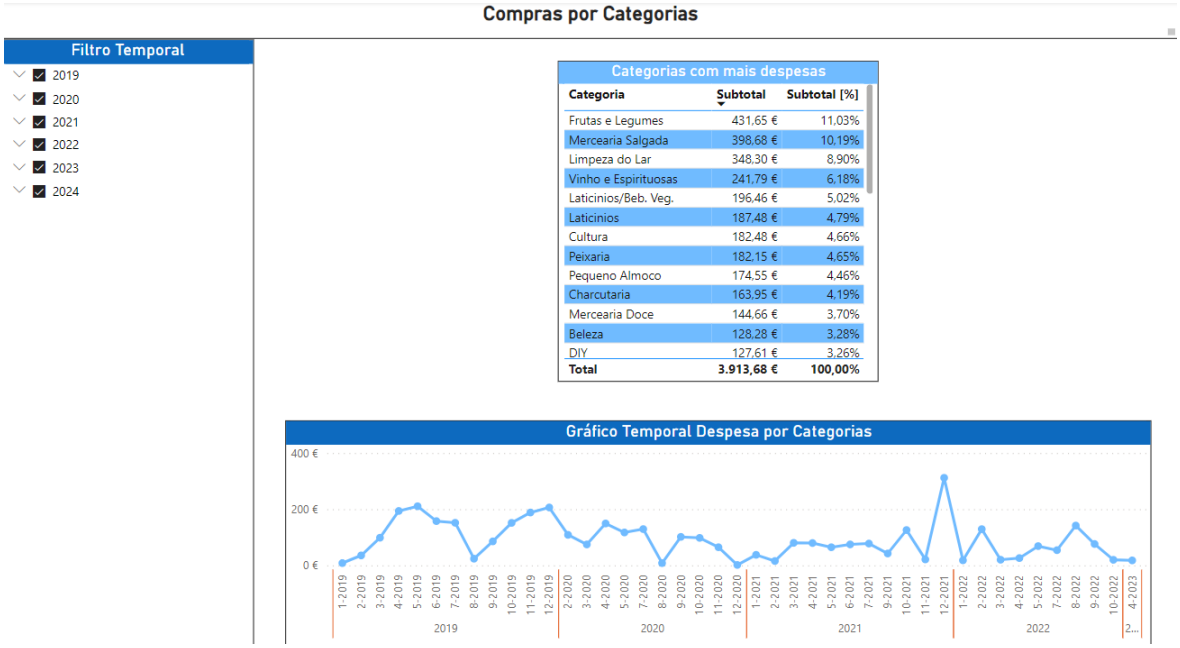


Figura 67 – Página 5 do *dashboard*: Despesa por categoria

4.7.6. Despesa por loja

Nesta sexta página encontra-se a lista das lojas onde ocorreram as maiores despesas. Foi na loja *Modelo Continente Hipermercado Leiria* onde foi realizado mais de metade das compras (67,61%). Esta página também reflete o quanto a BD se encontra baseada em faturas da cadeia de supermercados Continente: segundo a tabela apresentada na Figura 68, as 5 lojas com maior volume de faturação pertencem a essa cadeia, totalizando 97,88% do total faturado.

Na realidade, as despesas desde 2019 não se concentraram apenas nessas lojas, ao contrário do sugerido nos dados apresentados. Existem outras despesas semelhantes que foram realizadas em outras lojas equivalentes, mas que apenas fornecem a opção de faturas em papel. As faturas em papel são muitas vezes descartadas (ou perdidas) após as compras efetuadas, e por essa razão, apenas foi registada na BD uma pequena quantidade de faturas em papel.

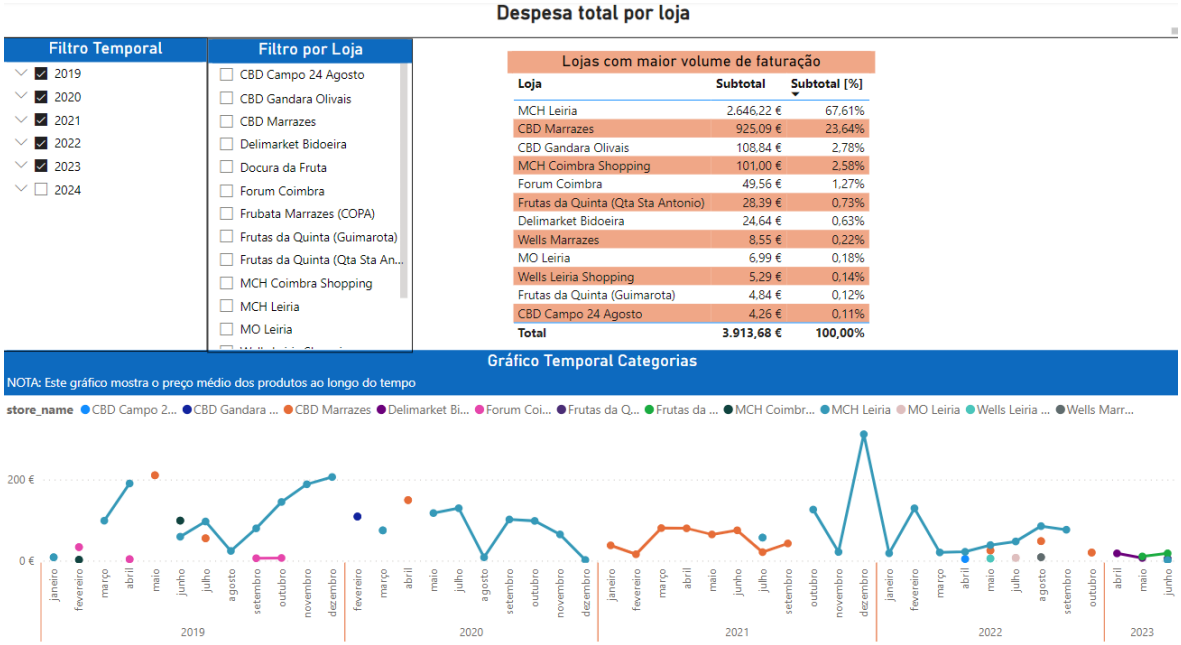


Figura 68 – Página 6 do *dashboard*: Despesa por loja

4.7.7. Despesas totais

Por fim, é nesta última página onde se obtém uma visão global da despesa. Além de possuir o valor total da despesa em destaque, a tabela central mostra o KPI da variação dessa despesa ano após ano. O gráfico temporal permite visualizar essa variação da despesa até ao nível dos meses. Essa página encontra-se ilustrada na Figura 69.

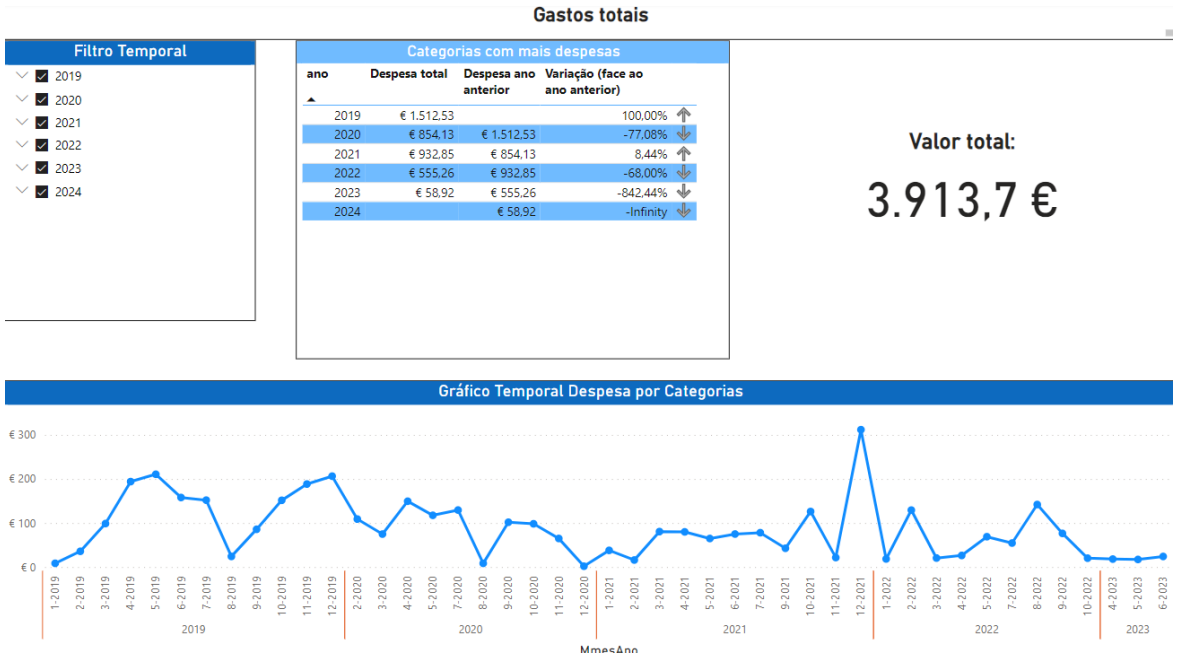


Figura 69 – Página 7 do *dashboard*: Despesas totais

4.8. Análise crítica

Através do projeto desenvolvido, e considerando as faturas utilizadas é possível responder à maioria das US definidas. No entanto, relativamente à análise das diferentes lojas, apenas é possível tirar conclusões sobre os diferentes estabelecimentos pertencentes ao grupo Continente na zona de Leiria.

Os gráficos da Figura 68 mostram que a grande maioria dos dados são provenientes da cadeia de supermercados Continente, o que não reflete a realidade das despesas obtidas com os supermercados desde 2019. Houve muitas outras despesas semelhantes a essas que foram realizadas em outras lojas que facultam apenas as faturas em papel, faturas essas que não foram guardadas e que não é possível recuperar. Por essa razão, a grande conclusão que se consegue retirar com a análise dos dados disponíveis é que estão em falta muitos outros dados.

Para conseguir responder corretamente a todas as US, é necessário incluir os dados referente às faturas emitidas pelas outras lojas. É sempre possível introduzir faturas de outras lojas que estejam em formato papel através da criação de *queries* SQL, tal como foi realizado com algumas faturas de mercearias e de outras lojas que não oferecem faturas eletrónicas, mas trata-se de um processo demorado e sujeito a uma taxa mais elevada de erros.

No entanto, com os dados disponíveis, é possível tirar conclusões sobre os produtos que geram a maior despesa e que se compram com mais frequência, mas também realizar uma análise sobre as diferenças de preços entre as diferentes lojas.

A BD e a análise dos respetivos dados encontram-se essencialmente focados nas compras realizadas em supermercados, mas facilmente pode ser adaptado para incluir faturas de eletricidade, gás, telecomunicações, crédito habitação, água e saneamento, entre outras. Hoje em dia, essas faturas são enviadas aos clientes via correio eletrónico, pelo que essas faturas já existem em formato digital. Isso possibilita a criação de algoritmos parecidos com os criados neste projeto para conseguir a extração dos dados dessas faturas e assim poder incluir essas despesas na BD e na análise dos dados. Tal evolução do projeto permitiria incluir mais despesas fixas relativas à casa, tornando possível uma análise mais minuciosa desses dados e uma elaboração mais precisa do orçamento familiar.

5. Conclusão

Este projeto permitiu desenvolver uma solução para um problema real, nomeadamente, gestão das despesas de casa, através de um DW, um sistema amplamente utilizado no mundo do BI. Após uma análise inicial durante a qual se definiram os objetivos do projeto através da definição de US, foi dimensionada e implementada uma BD de raiz, incluindo a conceção do modelo de dados, DER e dicionário de dados. Essa BD foi de seguida preenchida maioritariamente através de algoritmos desenvolvidos com a linguagem de programação *Python*, criados para o efeito. Estes algoritmos descarregam os ficheiros correspondentes às faturas eletrónicas, ficheiros esses armazenados numa conta de e-mail. Além disso, os algoritmos vão ler e extrair os dados existentes nessas faturas eletrónicas e preencher a BD, tudo de forma autónoma e automática. Foi igualmente criado um DW com um esquema em estrela totalmente de raiz, incluindo a definição das tabelas de facto e dimensões. Esse DW foi depois preenchido através de um processo ETL. Por fim, os dados carregados para o DW foram alvo de análise através dos diferentes *dashboards* criados e, com base nessas visualizações, foi possível tirar conclusões sobre os dados recolhidos a partir das faturas eletrónicas.

A realização deste projeto tornou-se num desafio bastante positivo e repleto de aprendizagens. Foi possível colocar em prática conhecimentos teóricos adquiridos ao longo do Mestrado através da implementação de uma solução tecnológica com o intuito de resolver um problema real. A UC *Business Intelligence* lecionada no Mestrado foi fundamental nesse sentido, visto que foi nela que foram abordados os temas relativos ao BI, nomeadamente os conceitos do DW, sistema OLAP, processo ETL e *dashboards*. A UC Integração de Dados também foi importante pela introdução dada às BD SQL, bem como pelos conceitos de integração de dados provenientes de diversas fontes que foram abordados. As demais UC, como *Deep Learning*, *Data Mining* e *Big Data*, apesar de não terem conceitos teóricos aplicados neste projeto, permitiram através dos seus trabalhos práticos descobrir e treinar a linguagem de programação *Python*, que foi essencial para o desenvolvimento dos algoritmos deste projeto. O desenvolvimento deste projeto permitiu aprofundar os conhecimentos técnicos relativos ao BI, perceber melhor os conceitos que existem em torno desta matéria e poder enfrentar obstáculos e dificuldades reais que se podem encontrar em projetos semelhantes. Estes conhecimentos estão agora a ser aplicados na minha atividade profissional e são primordiais para a minha evolução nesta área.

A principal dificuldade encontrada no desenvolvimento do projeto foi escrever o algoritmo que permitiu extrair os dados das faturas eletrónicas. A linguagem de programação *Python* tem a grande vantagem de possuir bibliotecas que facilitaram imenso o desenvolvimento desse *script*, o que torna essa linguagem de programação numa ferramenta muito útil para técnicos pouco experientes nessa matéria. As restantes ferramentas utilizadas também permitiram manusear mais facilmente a BD e criar código SQL, mesmo possuindo pouca experiência também nesse domínio. O *Microsoft SSMS* permitiu criar a BD de raiz, enquanto que o *Visual Studio* com *SQL Server Integration Services* permitiu executar o processo ETL de forma intuitiva.

Quanto à análise feita aos dados em si, foi possível tirar conclusões sobre quais são os produtos que se compram com mais frequência, mas também quais são as categorias que mais despesas originam ou estudar a evolução dos preços de qualquer um dos produtos. Relativamente à análise sobre a diferença de preços entre as diferentes lojas, apenas é possível tirar conclusões sobre os diferentes estabelecimentos pertencentes ao grupo Continente na zona de Leiria, já que a grande maioria dos dados disponíveis para este projeto são relativas a essas lojas.

5.1. Trabalho futuro

Conforme foi referido na secção 4.7.6, a grande maioria das despesas apresentadas são relativas às compras realizadas na cadeia de supermercado Continente. De forma a tornar a BD mais completa, é necessário conseguir incluir todas as despesas de todas as lojas. Para este projeto, as faturas em papel tiveram de ser introduzidas manualmente através das *queries* do SSMS. Como proposta de melhoria, sugere-se a criação de, por exemplo, uma aplicação web que permita introduzir as faturas em papel diretamente para a BD, através do preenchimento de um formulário.

Outra proposta de melhoria para este projeto é a inclusão de outras despesas que existem no dia a dia. Além dos supermercados, o sistema poderia igualmente armazenar despesas como eletricidade, água, telecomunicações, crédito habitação, entre outras. Hoje em dia, a maioria dessas faturas já são enviadas por correio eletrónico, pelo que é possível criar um *script* semelhante ao que foi criado neste projeto de forma a extrair os respetivos dados e armazenar os mesmos na BD. Assim, tornar-se-ia possível analisar todas as despesas fixas que existem

em torno de uma habitação, com o objetivo de encontrar eventuais formas de poupanças, bem como auxiliar na elaboração de orçamentos familiares.

O projeto desenvolvido pode igualmente seguir para uma vertente de predição de despesas futuras, através de técnicas de *Forecasting*. O *Forecasting*, igualmente estudado neste Mestrado, é uma técnica utilizada em Ciência de Dados para predição de tendências futuras com base em dados históricos. Com o registo das despesas que este projeto permite realizar, é possível criar modelos que possam prever quais serão as despesas da casa ao longo do ano e, assim, tornar ainda mais preciso a elaboração de um orçamento familiar.

Por último, uma forma muito mais moderna e eficiente para a extração de informações de ficheiros PDF seria com recurso à IA e a técnicas de ML. Ao invés de tentar encontrar padrões de forma manual nesses documentos com o intuito de extrair os dados, é possível com a tecnologia de hoje criar um modelo que "aprende" a ler esses dados com o auxílio de um conjunto de faturas de treino, para depois poder extrair os dados relevantes em faturas novas.

6. Referências

[1] R. Sharda, D. Delen e E. Turban, Business Intelligence, Analytics, and Data Science, 4 ed., Pearson, 2021.

[2] M. Nelson, “Beyond The Buzzword: What Does Data-Driven Decision-Making Really Mean?,” 23 set. 2022. [Online]. Available: https://www.forbes.com/sites/tableau/2022/09/23/beyond-the-buzzword-what-does-data-driven-decision-making-really-mean/. [Acedido em 11 11 2023].

[3] “eportugal,” 18 04 2023. [Online]. Available: https://eportugal.gov.pt/noticias/cabaz-de-44-produtos-alimentares-a-iva-zero-ja-disponivel. [Acedido em 09 02 2024].

[4] N. Jukic, S. Vrbsky, S. Nestorov e A. Sharma, Database Systems: Introduction to Databases and Data Warehouses, 2 ed., Prospect Press, 2021.

[5] F. V. Primak, Decisões com B.I. (Business Intelligence), Ciência Moderna, 2020.

[6] R. Sherman, Business Intelligence Guidebook, Amsterdam: Morgan Kaufmann, 2015.

[7] R. Kimball e J. Caserta, The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data, Kimball, 2004.

[8] D. M. Tayade, “International Research Journal of Engineering and Technology (IRJET),” *Comparative Study of ETL and E-LT in Data Warehousing*, pp. 2803-2807, 06 2019.

[9] “ETL (Extract, Transform, Load),” IBM, [Online]. Available: https://www.ibm.com/topics/etl. [Acedido em 25 01 2023].

[10] A. Berson e S. j. Smith, “Data Warehousing, Data Mining, and OLAP (Data Warehousing/Data Management),” 1997.

[11] "O que é um data warehouse?," IBM, [Online]. Available: https://www.ibm.com/br-pt/topics/data-warehouse. [Acedido em 20 02 2024].

[12] R. Kimball e M. Ross, The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, Wiley, 2013.

[13] W. Inmon, Building the Data Warehouse, Wiley, 2008.

[14] C. Adamson, em *Star Schema The Complete Reference*, McGraw Hill, 2010.

[15] T. d. M. Dutra, "Do The Math," 04 03 2021. [Online]. Available: https://blog.math.group/metodologia-star-schema-de-modelagem-dimensional. [Acedido em 18 01 2024].

[16] J. Berengueres, M. Sandell e A. Fenwick, Introduction to Data Visualization & Storytelling: A Guide For The Data Scientist (Visual Thinking), 3 ed., Independently Published, 2019.

[17] M. Scott, Programming Language Pragmatics, Morgan Kaufmann, 2015.

[18] E. Matthes, Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming, 2 ed., No Starch Press, 2019.

[19] D. Petkovic, Microsoft Sql Server 2019: A Beginner'S Guide, Seventh Edition, MCGRAW-HILL EDUCATION, 2020.

[20] O. j. Preez , Visual Studio 2022 In-Depth: Explore the Fantastic Features of Visual Studio 2022, 2 ed., BPB PUBN, 2022.

[21] "marquetplace," [Online]. Available: https://marketplace.visualstudio.com/items?itemName=SSIS.SqlServerIntegrationServicesProjects. [Acedido em 11 07 2023].

[22] "What Are SQL Server Integration Services?," Solarwinds, [Online]. Available: https://www.solarwinds.com/resources/it-glossary/ssis-sql-server-integration-services. [Acedido em 25 01 2024].

[23] S. Rodriguez e J. Swinnen, Business Intelligence with Power BI, 1 ed., Intersentia, 2023.

[24] D. G. Murray, Tableau Your Data!: Fast and Easy Visual Analysis with Tableau Software, Wiley, 2013.

[25] A. R. Yeruva e V. B. Ramu, End-to-End Observability with Grafana: A comprehensive guide to observability and performance visualization with Grafana, BPB Publications, 2023.

[26] "Draw.io," [Online]. Available: https://app.diagrams.net/. [Acedido em 2023].

[27] J. Sutherland, Scrum: The Art of Doing Twice the Work in Half the Time, Random House, 2014.

[28] "Scrum," Scrum Portugal, [Online]. Available: https://www.scrumportugal.pt/scrum/. [Acedido em 21 02 2024].

[29] icon archive, [Online]. Available: https://www.iconarchive.com/. [Acedido em 2024 02 22].

[30] S. Doshi, "Medium," 13 10 2018. [Online]. Available: https://medium.com/@sdoshi579/to-read-emails-and-download-attachments-in-python-6d7d6b60269. [Acedido em 11 2022].

[31] P. S. Foundation, "PYTHON," 19 12 2023. [Online]. Available: https://docs.python.org/3/library/imaplib.html#module-imaplib. [Acedido em 19 12 2023].

[32] R. Awati, "TechTarget," 09 2021. [Online]. Available: https://www.techtarget.com/whatis/definition/IMAP-Internet-Message-Access-Protocol. [Acedido em 19 12 2023].

[33] P. S. Foundation, "PYTHON," 19 12 2023. [Online]. Available: https://docs.python.org/3/library/os.html. [Acedido em 20 12 2023].

[34] P. S. Foundation, “PYTHON,” 19 12 2023. [Online]. Available: https://docs.python.org/3/library/email.html#module-email. [Acedido em 20 12 2023].

[35] “GOOGLE Support,” [Online]. Available: https://support.google.com/mail/answer/185833?hl=en. [Acedido em 20 12 2023].

[36] M. Fenniak, “PyPDF2,” 2008. [Online]. Available: https://pypdf2.readthedocs.io/en/latest/index.html. [Acedido em 12 11 2022].

[37] P. S. Foundation, “PYTHON,” [Online]. Available: https://docs.python.org/3/library/re.html#module-re. [Acedido em 11 2022].

[38] “PANDAS Python,” [Online]. Available: https://pandas.pydata.org/. [Acedido em 07 01 2024].

[39] D. Toomey, Jupyter for Data Science: Exploratory analysis, statistical modeling, machine learning, and data visualization with Jupyter, Packt Publishing, 2017.

[40] “Lei n.º 17/2023, de 14 de abril,” Diário da República, [Online]. Available: https://diariodarepublica.pt/dr/detalhe/lei/17-2023-211783460. [Acedido em 14 03 2024].

[41] “PyPI,” Python Package Index, [Online]. Available: https://pypi.org/project/pyodbc/. [Acedido em 14 01 2024].

[42] A. Hughes, “techtarget,” 10 2017. [Online]. Available: https://www.techtarget.com/searchoracle/definition/Open-Database-Connectivity. [Acedido em 14 01 2023].

[43] P. d. Finanças. [Online]. Available: https://info.portaldasfinancas.gov.pt/pt/apoio_contribuinte/Documents/ListaFreguesiasVigentes_SFs.xls. [Acedido em 31 01 2023].

[44] A. Ferrari e M. Russo, The Definitive Guide to Dax Business Intelligence With Microsoft Excel, Sql Server Analysis Services, And Power Bi, 2 ed., MICROSOFT PRESS, 2019.

[45] “sqlservertutorial,” [Online]. Available: https://www.sqlservertutorial.net/sql-server-basics/sql-server-decimal/. [Acedido em 17 07 2023].

[46] P. S. Foundation, “PYTHON,” 19 12 2023. [Online]. Available: https://docs.python.org/3/library/base64.html#module-base64. [Acedido em 20 12 2023].

[47] “Microsoft Learn,” 23 08 2023. [Online]. Available: https://learn.microsoft.com/en-us/sql/connect/python/pyodbc/step-1-configure-development-environment-for-pyodbc-python-development?view=sql-server-ver16&tabs=windows.

[48] “Microsoft Learn,” 11 01 2023. [Online]. Available: https://learn.microsoft.com/en-us/sql/connect/python/pyodbc/step-3-proof-of-concept-connecting-to-sql-using-pyodbc?view=sql-server-ver16.

Anexos

Anexo I. Relacionamentos entre as tabelas da BD

Este anexo apresenta com mais pormenores os relacionamentos existentes entre todas as tabelas da BD.

Relacionamento entre as tabelas *invoice* e *invoice_line*:

Figura 70 – Diagrama relacionamento entre as tabelas *invoice* e *invoice_line*

- Cardinalidade: 1:N
- Participação: obrigatória dos dois lados;
- Resumo: cada fatura tem de ter pelo menos 1 linha de venda e cada linha de venda tem de pertencer a 1 só fatura

Relacionamento entre as tabelas *invoice_line* e *item*:

Figura 71 – Diagrama relacionamento entre as tabelas *invoice_line* e *item*

- Cardinalidade: 1:N
- Participação: obrigatória do lado N;
- Resumo: cada linha de venda tem de ter 1 só produto e os produtos podem estar em 0 ou mais linhas de venda.

Relacionamento entre as tabelas *item* e *category*:

Figura 72 – Diagrama relacionamento entre as tabelas *item* e *category*

- Cardinalidade: 1:N
- Participação: obrigatória do lado N;
- Resumo: cada produto tem de pertencer a 1 categoria e as categorias podem estar associados a 0 ou mais produtos.

Relacionamento entre as tabelas *invoice* e *store*:

Figura 73 – Diagrama relacionamento entre as tabelas *invoice* e *store*

- Cardinalidade: 1:N
- Participação: obrigatória do lado N;
- Resumo: cada fatura tem de ser vendida por 1 loja e cada loja pode ter emitido 0 ou mais faturas.

Relacionamento entre as tabelas *store* e *county*:

Figura 74 – Diagrama relacionamento entre as tabelas *store* e *county*

- Cardinalidade: 1:N
- Participação: obrigatória do lado N;
- Resumo: cada loja tem de estar localizada num Concelho e cada Concelho pode ter 0 ou mais lojas.

Anexo II. Modelo lógico

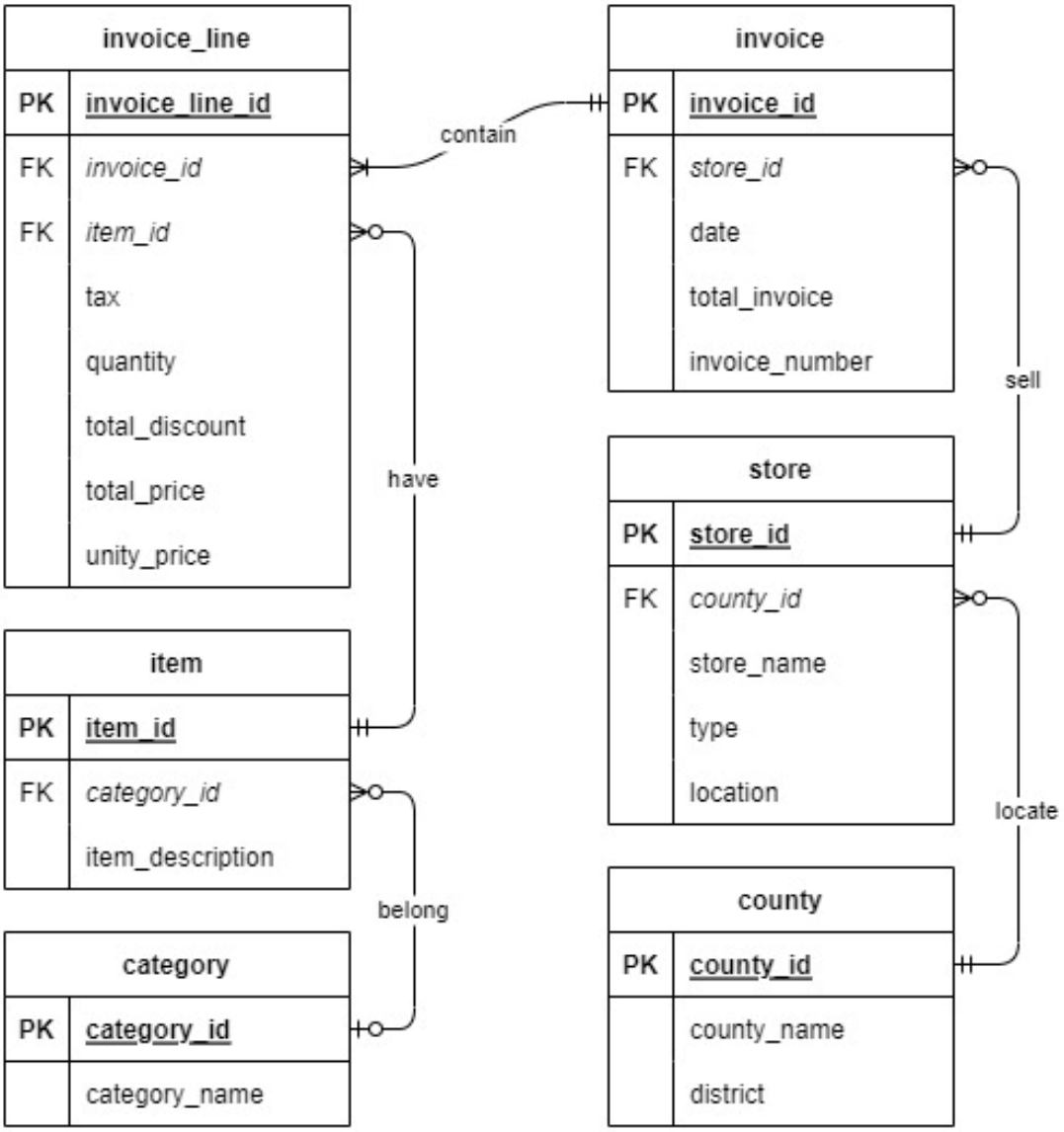


Figura 75 – Modelo lógico da BD

Anexo III. Dicionário de dados

Neste anexo estão discriminados cada um dos atributos existentes nas 6 tabelas pelo intermédio de dicionários de dados organizados da seguinte forma:

- Tipo: categoriza o tipo de variável. O caso desta BD, as variáveis podem ser do tipo:
    - **int:** inteiro;
    - **date:** Data no formato YYYY-MM-DD;
    - **decimal(*precision*, *scale*):** número decimal cujo *precision* representa a quantidade total de dígitos e *scale* a quantidade de dígitos à direita da vírgula (por exemplo: o número 123,45 tem *precision* = 5 e *scale* = 2) [45];
    - **varchar(n):** *string* que pode ter até *n* caracteres.
- **Obrig.:** se o campo é ou não de preenchimento obrigatório. No caso específico desta BD, todos os campos de todas as tabelas são de preenchimento obrigatório;
- **PK?:** Se o campo é ou não uma chave primária (PK). Nesta BD, todas as chaves primárias estão identificadas com o sufixo "_id";
- **Domínio**: Indica se existe restrições quanto à gama de valores que a variável pode ou não assumir.

Tabela 7 – Tabela *Invoice*

Coluna	Tipo	Obrig.	PK?	Domínio	Notas
invoice_id	int	Sim	Sim	Números inteiros sequenciais a começar em 1 e com incremento 1	
store_id	int	Sim	Não	store.store_id	
date	date	Sim	Não	≤ data e hora atual	
total_invoice	decimal (7,2)	Sim	Não	≥ 0	valor total da fatura
invoice_number	varchar (20)	Sim	Não		número da fatura

Tabela 8 – Tabela *store*

Coluna	Tipo	Obrig.	PK?	Domínio	Notas
store_id	int	Sim	Sim	Números inteiros sequenciais a começar em 1 e com incremento 1	
county_id	int	Sim	Não		id do concelho da localidade da loja
store_name	varchar (50)	Sim	Não		nome ou designação da loja
type	varchar (50)	Sim	Não		tipo de loja
location	varchar (25)	Sim	Não		localidade

Tabela 9 – Tabela *county*

Coluna	Tipo	Obrig.	PK?	Domínio	Notas
county_id	int	Sim	Sim	Números inteiros sequenciais a começar em 1 e com incremento 1	
county_name	varchar (30)	Sim	Não		nome do Concelho
district	varchar (17)	Sim	Não		Distrito

Tabela 10 – Tabela *item*

Coluna	Tipo	Obrig.	PK?	Domínio	Notas
item_id	int	Sim	Sim	Números inteiros sequenciais a começar em 1 e com incremento 1	
category_id	int	Sim	Não	category.category_id	categoria do produto
item_description	varchar (50)	Sim	Não		descrição do produto

Tabela 11 – Tabela *invoice_line*

Coluna	Tipo	Obrig.	PK?	Domínio	Notas
invoice_line_id	int	Sim	Sim	Números inteiros sequenciais a começar em 1 e com incremento 1	
invoice_id	int	Sim	Não	invoice.invoice_id	
item_id	int	Sim	Não	item.item_id	ID do produto
tax	decimal (3,2)	Sim	Não		taxa do IVA
quantity	decimal (5,2)	Sim	Não	> 0	quantidade de produto comprado
total_discount	decimal (5,2)	Sim	Não	Por omissão = 0	valor do desconto total
unity_price	decimal (7,2)	Sim	Não	≥ 0	preço unitário
total_price	decimal (7,2)	Sim	Não	≥ 0	preço total

Tabela 12 – Tabela *category*

Coluna	Tipo	Obrig.	PK?	Domínio	Notas
category_id	int	Sim	Sim	Números inteiros sequenciais a começar em 1 e com incremento 1	
category_name	varchar (25)	Sim	Não		nome da categoria