Instituto Politécnico do Cávado e do Ave

Escola Superior de Tecnologia

Licenciatura em
Engenharia de Sistemas Informáticos

Trabalho Prático I

Fábio Rafael Gomes Costa

Lino Emanuel Oliveira Azevedo

Barcelos, outubro de 2025

Instituto Politécnico do Cávado e do Ave

Escola Superior de Tecnologia

Licenciatura em Engenharia de Sistemas Informáticos

Trabalho Prático I

Fábio Rafael Gomes Costa – a22997

Unidade Curricular:

Integração de Sistemas de Informação (ISI)

Docente:

Dr. Óscar Rafael da Silva Ferreira Ribeiro

Barcelos, outubro de 2025

Ficha de Identificação

Elaborado por	Fábio Rafael Gomes Costa
	Lino Emanuel Oliveira Azevedo
Contato	a22997@alunos.ipca.pt
	a23015@alunos.ipca.pt

Unidade Curricular	Integração de Sistemas de Informação
Curso	Licenciatura em Engenharia de Sistemas Informáticos
Instituição	Escola Superior de Tecnologia do Instituto Politécnico do Cávado e do Ave
Professor	Doutor Óscar Rafael da Silva Ferreira Ribeiro
Gabinete	3
Contato	oribeiro@ipca.pt

Data de início	26 de setembro de 2025
Data de conclusão	17 de Outubro de 2025

Índice

Índice de Figuras

Lista de Siglas e Acrônimo

ETL – extrair, transformar e carregar

IPCA – Instituto Politécnico do Cavado e do Ave

ISI – Integração de Sistemas de Informação

KPIs – Key Performance Indicator (Indicador-Chave de Desempenho)

UC – Unidade Curricular

1. Introdução

1.1 Contextualização

No âmbito da Unidade Curricular (UC) de Integração de Sistemas de Informação (ISI), inserida no 1º semestre do 3º ano do curso, foi proposta a realização de um trabalho prático como instrumento de avaliação, sob a orientação do docente Dr. Óscar Ribeiro.

Este projeto foca-se na aplicação prática dos conceitos lecionados, através da implementação de um fluxo de trabalho de ETL (Extract, Transform, Load), demonstrando o processo de integração de sistemas de informação ao nível dos dados.

1.2 Motivação/Pretensões e Objetivos

A motivação para este projeto surge do desafio, cada vez mais presente no mundo empresarial, de integrar sistemas de informação heterogéneos. As empresas lidam com dados provenientes de múltiplas fontes, em formatos distintos e com qualidades variáveis, tornando essencial a existência de processos que unifiquem e deem sentido a essa informação. Este trabalho simula um cenário real onde é necessário extrair, transformar e carregar dados para gerar valor e apoiar a tomada de decisão.

A principal pretensão é, portanto, projetar e implementar um processo de ETL (Extract, Transform, Load) completo e automatizado. Este processo será responsável por ler dados de um ficheiro de inventário de veículos (.csv) e de um ficheiro de registo de vendas (.xml ou .csv), limpá-los, validá-los, combiná-los e, por fim, apresentar métricas de negócio consolidadas num dashboard visual.

Para alcançar este fim, foram definidos os seguintes objetivos, em linha com o proposto no enunciado da Unidade Curricular:

- Consolidar conceitos de Integração de Sistemas de Informação através da manipulação e integração de dados.
- Analisar e especificar um cenário prático de aplicação de processos de ETL.
- Explorar ferramentas de suporte a processos de ETL, neste caso o Apache Hop e o Node-RED.
- Potenciar a experiência no desenvolvimento de software através da aplicação prática dos conceitos.

1.3 Estrutura do Documento

Este relatório foi organizado de forma a apresentar o trabalho desenvolvido de maneira clara e sequencial. Inicia-se com a capa, subcapa e Ficha de Identificação, seguida do Índice, que inclui tanto os tópicos principais como o Índice de Figuras, e a Lista de Siglas e Acrónimos.

A Introdução apresenta o enquadramento geral do projeto, incluindo a contextualização, os objetivos e a metodologia.

Na secção seguinte, Problemas, são identificados e descritos os problemas enfrentados durante o desenvolvimento do projeto.

O Desenvolvimento do projeto está dividido em duas fases: a resolução do problema com o Knime e a resolução com o Apache hup. Cada uma das fases apresenta a estratégia utilizada, um vídeo demostrativo e o repositório no github.

O relatório é concluído com as Considerações Finais, onde são apresentadas as reflexões sobre o trabalho realizado, as aprendizagens obtidas e sugestões de melhoria.

Por fim, são apresentadas as Referências Bibliográficas, onde se encontram listados todos os links consultados durante o desenvolvimento do projeto.

2. Problema

O presente trabalho foca-se na criação de um processo de Extração, Transformação e Carga (ETL) para a empresa fictícia "Stand CarExpress". O objetivo é integrar e analisar dados de duas fontes distintas para gerar informações de valor para o negócio.

As fontes de dados são:

- Um ficheiro CSV (car_information_dataset.csv) contendo um inventário geral de veículos, com detalhes técnicos como emissões de CO2, tipo de combustível e quilometragem.
- Um ficheiro XML/CSV (stand_vendas.xml ou stand_vendas.csv) que regista as vendas de veículos, incluindo data da venda e preço.

O principal desafio é que os dados estão em formatos heterogéneos e contêm inconsistências, erros e valores nulos. Por exemplo, as marcas de veículos estão escritas de formas diferentes ("vw", "volkswagen"), as matrículas podem ter formatos inválidos, e alguns registos essenciais podem estar em falta.

O objetivo deste projeto é, portanto, desenvolver um processo automatizado que:

- Extraia dados das duas fontes.
- Limpe, padronize, valide e enriqueça os dados.
- Combine as informações de inventário com as de vendas para criar uma visão unificada.
- Calcule métricas agregadas, como a média de emissões de CO2 e o valor médio de venda por marca.
- Carregue os resultados num dashboard visual para análise.
- Isole e registe os dados inválidos ou inconsistentes para auditoria posterior.

3. Knime

3.1 Estratégia utilizada

A estratégia seguiu uma lógica modular e encadeada, onde cada conjunto de nós do KNIME representa uma etapa do ciclo ETL:

3.1.1 Ferramentas

Knime: Utilizado como a principal ferramenta de ETL para desenhar e executar todo o fluxo de dados, desde a extração até à carga. Foi criado um Workflow para orquestrar a execução e a gestão de erros.

Node-RED: Utilizado para a camada de visualização. Um dashboard foi criado para receber os dados processados pelo Knime e apresentá-los de forma gráfica e intuitiva.

3.1.2 Processo de ETL

O processo foi dividido em três fases macro:

1. Extração (Extract):

- Leitura e carregamento dos dados do ficheiro car_info.csv.
- Leitura e carregamento dos dados do ficheiro sales_info.csv.

2. Transformação (Transform):

- Filtragem inicial: Utilização do nó *rule-based Row Splitter* remoção de registos com campos nulos (Marca, Modelo, Matrícula).
- Limpeza e Padronização: Aplicação de operações de *String Manipulation* para remover espaços, converter para minúsculas e remover espaços indesejados de vários campos.
- Correção de Dados: Utilização do nó *Table Creator* para normalizar nomes de marcas (ex: "vw" para " volkswagen") e tipos de combustível.
- Correção de Dados: Utilização do operador *Rule Engine* para normalizar os tipos de combustível.
- Correção de Dados: Utilização do operador *String Manipulation* para normalizar através de regex a coluna “Co2_Emission”

- Validação: Uso de Expressões Regulares para validar o formato das matrículas, separando as válidas das inválidas.
- Enriquecimento e Cálculos: Conversão de unidades de milhas para quilómetros e interpretação de múltiplos formatos de data através de um script JavaScript.
- Integração: Junção dos dois fluxos de dados (Merge Join) com base na matrícula (License_Plate) para associar os dados de venda aos dados técnicos de cada veículo.
- Agregação: Agrupamento dos dados por marca (Group By) para calcular KPIs (Key Performance Indicator (Indicador-Chave de Desempenho)) como o número de vendas, soma e média de emissões de CO2, e valores mínimo, máximo e médio de venda.

3. Carga (Load):

- Saídas de Erro: Os dados que falham nas validações são exportados para ficheiros JSON específicos com os veículos e vendas que tenham valores null (car_info_nulls.json, car_sales_nulls.json), ficheiros XML no caso dás matrículas invalidas extrai os veículos com as mesmas (Car_info_WrongPlate.xml, Sales_info_WrongPlate.xml), ficheiros CSV para as unmatched no caso do Joiner principal (Car info sem match.csv, Vendas sem match.csv), criando um registo claro dos problemas de qualidade de dados.
- Extração de Gráficos: No formato de imagem, (media_co2_por_marca.svg, receita_mensal_por_marca.svg).
- Envio para Dashboard: Os dados agregados e processados são formatados em JSON e enviados através de um cliente REST para um endpoint em Node-RED (http://localhost:1880/avg-co2).
- Visualização: O fluxo em Node-RED recebe os dados, extrai as informações relevantes e alimenta um gráfico de barras que exibe a média de CO2 por marca.

3.2 Transformações

3.2.1 Fase 1: Extração e Preparação dos Dados dos Veículos (Ficheiro CSV)

Esta fase foca-se na leitura e tratamento dos dados provenientes do ficheiro car_info.csv. O objetivo é validar, limpar e padronizar os dados do inventário de veículos.

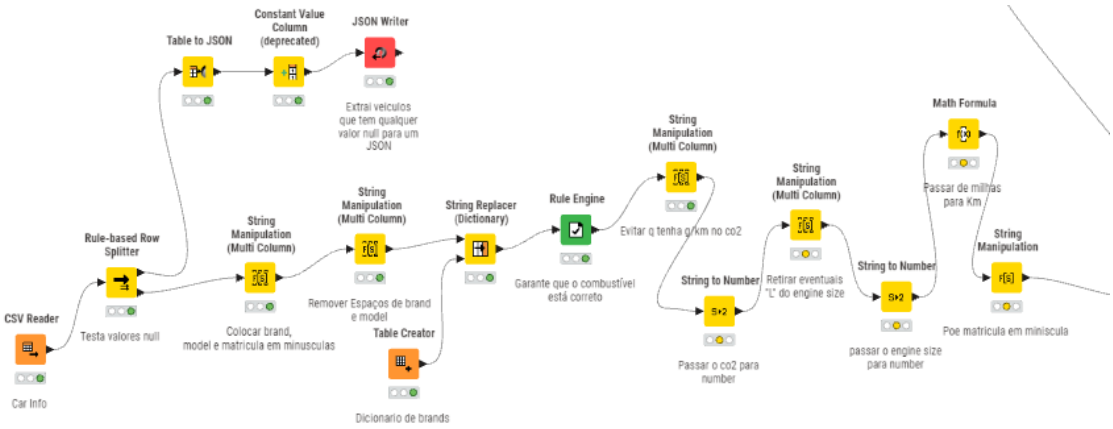


Figura 1 - Fase 1

Passos Detalhados:

1. **Extração (Input car_info.csv):** O processo inicia-se com a leitura dos dados do ficheiro CSV, que contém o inventário dos veículos.

Row ID	Brand	Model	Year	Fuel_T...	Transmi...	Mileage	Doors	Owner...	Price	Pollution	CO2_E...	Engine...	License...
Row0	Kia	Rio	2020	Diesel	Manual	289944	3	5	8501	High	168g km	4.2L	AB-12-CD
Row1	Chevrolet	Malibu	2012	Hybrid	Automatic	5356	2	3	12092	Medium	95	2.0	CD-34-EF
Row2	Mercedesbenz	GLA	2020	Gasoleo	Automatic	231440	4	2	11171	High	172	4.2	EF-56-GH
Row3	AUDI	Q5	2023	Electric	Manual	160971	2	1	11780	Low	0	2.0	GH-78-IJ
Row4	Volkswagen	Golf	2003	Hybrid	Semi-Autom...	286618	3	3	2867	Medium	105	2.6	IJ-90-KL
Row5	Toy ota	Camry	2007	Petrol	Automatic	157889	4	4	7242	High	185	2.7	KL-12-MN
Row6	Honda	Civic	2010	Electric	Automatic	139584	3	1	11208	Low	0	3.4	MN-34-OP
Row7	Kia	Sportage	2001	Electric	Semi-Autom...	157495	2	2	7950	Low	0	4.7	OP-56-QR
Row8	Kia	Sportage	2014	Hybrid	Manual	98700	3	4	9926	Medium	98	2.6	QR-78-ST
Row9	Toyota	RAV4	2005	Petrol	Manual	107724	2	5	6545	High	195	3.1	ST-90-UV
Row10	B M W	5 Series	2013	Hybrid	Automatic	296824	2	3	5863	Medium	85	1.3	UV-12-WX
Row11	Volkswagen	Golf	2009	Hybrid	Manual	42795	4	3	11444	Medium	112	4.5	WX-34-YZ
Row12	Honda	CR-V	2007	Hybrid	Automatic	132875	3	4	10842	Medium	118	4.8	YZ-56-AB
Row13	Hyundai	Elantra	2004	Hybrid	Semi-Autom...	188996	2	1	4820	Medium	93	2.3	AB-78-CD
Row14	Volkswagen	Golf	2004	Diesel	Automatic	60103	2	1	9697	High	158	3.2	CD-90-EF
Row15	Volkswagen	Golf	2007	Diesel	Automatic	60103	2	1	9697	High	158	3.2	EF-12-GH
Row16	Hyundai	Elantra	2017	Electric	Automatic	38133	5	2	14837	Low	0	2.9	GH-34-IJ
Row17	Volkswagen	Tiguan	2006	Electric	Manual	41161	2	2	11576	Low	0	4.5	IJ-56-KL
Row18	Kia	Rio	2000	Diesel	Semi-Autom...	257427	3	3	2351	High	161	3.4	KL-78-MN
Row19	Mercedes	GLA	2021	Petrol	Manual	34640	2	1	11207	High	150	1.5	MN-90-OP
Row20	Chevrolet	Equinox	2018	Hybrid	Automatic	21261	2	1	13374	Medium	88	1.8	MN-90-OQ
Row21	Toyota	RAV4	2000	Hybrid	Semi-Autom...	41814	2	2	8863	Medium	116	4.6	MN-90-OR
Row22	Toyota	RAV4	2003	Electric	Manual	136662	5	4	6266	Low	0	2.0	MN-90-OS

Figura 2 - Input car_info.csv

2. **Filtragem de Nulos (Ruled-Based Row Spliter):** O primeiro passo de validação consiste em verificar se algum dos campos contêm valores. Registos que falhem

esta validação são desviados para o ficheiro *car_info_nulls.json*, garantindo que apenas dados completos prossigam no fluxo principal.

```
MISSING $Model$          => TRUE
MISSING $Year$           => TRUE
MISSING $Fuel_Type$      => TRUE
MISSING $Transmission$   => TRUE
MISSING $Mileage$        => TRUE
MISSING $Doors$          => TRUE
MISSING $Owner_Count$    => TRUE
MISSING $Price$          => TRUE
MISSING $Pollution$      => TRUE
MISSING $CO2_Emissions$  => TRUE
MISSING $Engine_Size$    => TRUE
MISSING $License_Plate$  => TRUE
TRUE => FALSE
```

Figura 3 - Ruled-Based Row Spliter – Expression

3. **Limpeza e Padronização (String Manipulation (Multi Column)):** São aplicadas transformações para garantir a consistência dos dados.
   Isto inclui:
   - Converter o texto para minúsculas das colunas Brand e Model(lowerCase($$CURRENTCOLUMN$$)).
   - Remover espaços em branco no início e no fim dos campos com (regexReplace($$CURRENTCOLUMN$$, "\\s+", "")).
   - Remover "g km" na coluna CO2_emissions (regexReplace(replace($CO2_Emissions$, ",", "."), "[^0-9.]", "")).
   - Retirar eventuais
4. **Correção de Dados (Correct Brands & Correct Fuel):** Utiliza-se o componente Table Creater para normalizar valores. Por exemplo, a abreviatura "vw" é corrigida para " volkswagen".
   E utiliza-se o Rule Engine para normalizar o Fuel Type "Gasoleo" para "Diesel".

	column1	column2
Row0	vw	volkswagen
Row1	volks wagen	volkswagen
Row2	merc	mercedes
Row3	mercedesbenz	mercerdes

Figura 4 - Correct Brands

Expression

```
1 $Fuel_Type$ MATCHES "(?i).*(diesel|gas[óo]leo).*" => "Diesel"
2 $Fuel_Type$ MATCHES "(?i).*(gasolina|petrol|benzina).*" => "Gasoline"
3 $Fuel_Type$ MATCHES "(?i).*(hybrid|h[íi]brido|phev|hev).*" => "Hybrid"
4 $Fuel_Type$ MATCHES "(?i).*(electric|el[eé]trico|bev|ev).*" => "Electric"
5 $Fuel_Type$ MATCHES "(?i).*(lpg|gpl).*" => "LPG"
6 $Fuel_Type$ MATCHES "(?i).*(cng|gnv|gnc|metano).*" => "CNG"
7 TRUE => "Other"
```

Figura 5 - Rule Engine - Expression

5. **Enriquecimento:**
    - O campo *Mileage* (em milhas) é convertido para quilómetros através de uma fórmula matemática ($*Mileage*$* 1.60934), criando um novo campo chamado Km com recurso ao *Math Formula*.
    - *String* to *Number* é utilizado nos campos *CO2_Emissions* e no *Engine_Size*.

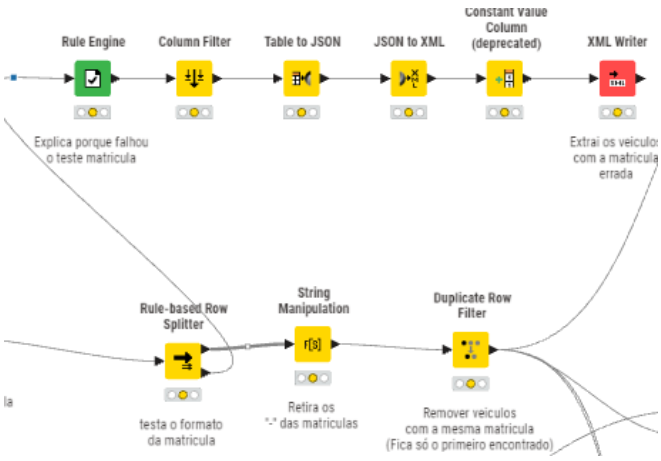


Figura 6 - Fase 1 Continuação

6. **Validação de Matrículas:** É aplicada uma Expressão Regular complexa (no Rule-based Row Splitter) para validar o formato das matrículas portuguesas e remover hífens das matrículas no String Manipulation (). Registos com matrículas inválidas ou duplicadas são filtrados e exportados para o ficheiro WrongPlate.xml, isolando-os do fluxo principal.

```
Expression
1 $License_Plate$ MATCHES "^[0-9]{2}-[a-z]{2}-[a-z]{2}$" => TRUE
2 $License_Plate$ MATCHES "^[a-z]{2}-[0-9]{2}-[a-z]{2}$" => TRUE
3 $License_Plate$ MATCHES "^[a-z]{2}-[a-z]{2}-[0-9]{2}$" => TRUE
4 $License_Plate$ MATCHES "^[0-9]{2}-[a-z]{2}-[0-9]{2}$" => TRUE
5 $License_Plate$ MATCHES "^[a-z]{2}-[0-9]{2}-[0-9]{2}$" => TRUE
6 $License_Plate$ MATCHES "^[0-9]{2}-[0-9]{2}-[a-z]{2}$" => TRUE
7 TRUE => FALSE
```

Figura 7 - Rule-based Row Splitter - Expression

```
Expression
1 lowerCase(
2   regexReplace(
3     regexReplace($License_Plate$, "[\\p{Zs}\\s]+", ""),
4   "[\\p{Pd}]+", "-")
5 )
```

Figura 8 - String Manipulation

3.2.2 Fase 2: Extração e Preparação dos Dados de Vendas (Ficheiro CSV)

De forma paralela à Fase 1, este fluxo processa os registos de vendas do ficheiro stand_vendas.xml.

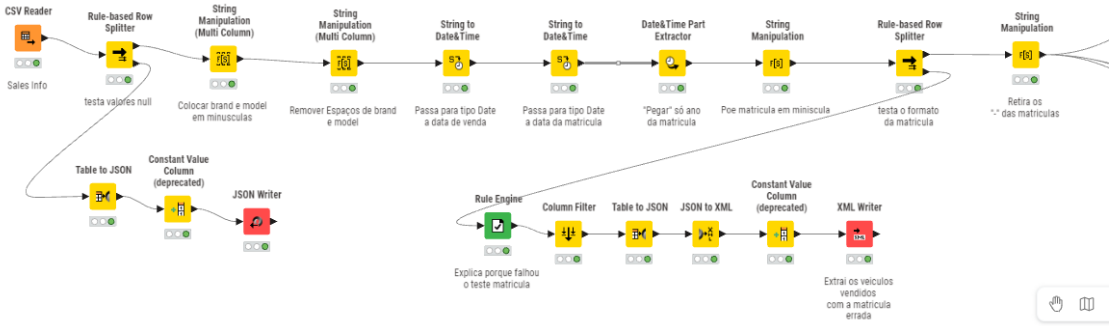


Figura 9 - Fase 2

Passos Detalhados:

1. **Extração (sales_info.csv):** O processo inicia-se com a leitura dos dados de vendas, que estão estruturados em formato CSV.

Row ID	Brand	Model	Year_a...	Date_o...	Sale_Price	License...
Row0	Kia	Rio	2020-06	2020-06-15	11250	AB-12-CD
Row1	Chevrolet	Malibu	2012-09	2020-05-22	10450	CD-34-EF
Row2	Mercedesbenz	GLA	2020-02	2020-07-11	13800	EF-56-GH
Row3	AUDI	Q5	2023-11	2020-09-03	12100	GH-78-IJ
Row4	VW	Golf	2003-03	2020-11-19	9800	IJ-90-KL
Row5	Toy ota	Camry	2007-07	2021-01-08	13200	KL-12-MN
Row6	Honda	Civic	2010-04	2021-02-25	10750	MN-34-OP
Row7	Kia	Sportage	2001-12	2021-04-14	11500	OP-56-QR
Row8	Kia	Sportage	2014-05	2021-06-02	10200	QR-78-ST
Row9	Toyota	RAV4	2005-08	2021-07-21	11800	ST-90-UV
Row10	B M W	5 Series	2013-01	2021-09-09	14500	UV-12-WX
Row11	Volkswagen	Golf	2009-02	2021-10-28	12800	WX-34-YZ
Row12	Honda	CR-V	2007-07	2021-12-16	11100	YZ-56-AB
Row13	Hyundai	Elantra	2004-10	2022-02-03	14200	AB-78-CD
Row14	Volkswagen	Golf	2004-05	2022-03-24	10600	CD-90-EF
Row15	Volkswagen	Golf	2007-03	2022-05-12	12400	EF-12-GH
Row16	Hyundai	Elantra	2017-11	2022-06-30	9900	GH-34-IJ
Row17	Volkswagen	Tiguan	2006-08	2022-08-18	11900	IJ-56-KL
Row18	Kia	Rio	2000-01	2022-10-06	10800	KL-78-MN
Row19	Mercedes	GLA	2021-06	2022-11-24	13100	MN-90-OP
Row20	BMW	5 Series	2021-06	2023-11-24	14500	AE-29-HE
Row21	Hyundai	Elantra	2004-10	?	?	?
Row22	Hyundai	Elantra	2006-09	2021-01-08	14900	MN-HH-PP

Figura 10 - Input sales_info.csv

2. **Filtragem de Nulos (Ruled-Based Row Spliter):** Tal como no fluxo do car_info, os registos com campos nulos são filtrados e enviados para um ficheiro de erro, o car_sales_nulls.json.

3. **Limpeza e Correção (String Manipulation (Multi Column) 2):** São aplicadas as mesmas lógicas de limpeza e correção de marcas vistas na Fase 1, garantindo que os dados de ambos os fluxos sejam comparáveis.

4. Usa-se o String to Date-Time para colocar a data da matriculo em tipo date-Time.

5. **Validação de Matrículas:** A mesma validação por Expressão Regular é aplicada às matrículas do ficheiro de vendas para garantir a sua integridade. Aqui não se verifica a repetição de matrículas visto que um carro pode ser vendido mais de uma vez (retorno ou devolução).

3.2.3 Fase 3: Junção dos Dados e Análise de Discrepâncias, Configuração e utilização do Joiner

Nesta fase, os dois fluxos de dados, já limpos e preparados, são combinados. O Objetivo é Unir o registo de **vendas** com o **catálogo de carros** para produzir um dataset integrado que suporte KPIs (receita, CO2) e relatórios de qualidade (linhas sem correspondência).

1. **Entradas e chaves**

Faz-se o Joiner as duas tabelas da seguinte forma:

- Left input: tabela de vendas
- Right input: tabela de carros
- Chave de junção: License_Plate ↔ License_Plate

Modo de junção: Inner Join (Matching rows only). Este modo devolve apenas os registos que existem em ambas as tabelas.

A partir deste conjunto integrado, são derivadas três análises principais:

1. Receita mensal por marca
2. Média de emissões de CO2 por marca
3. Estatísticas de preços por marca

Cada análise origina tabelas e gráficos exportados em formato imagem e JSON, permitindo integração com dashboards externos.

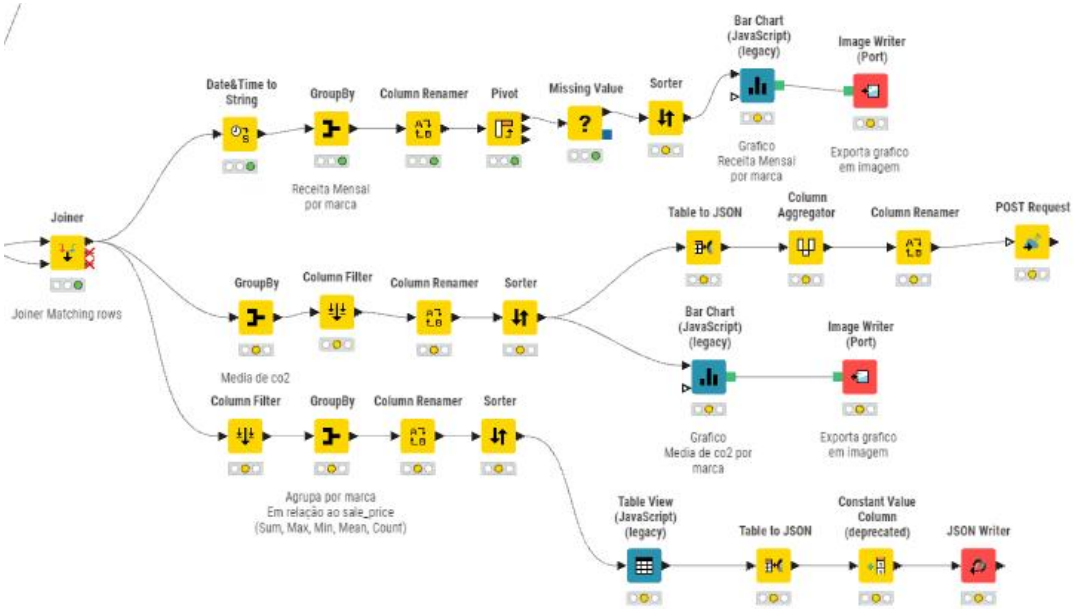


Figura 11 – Joiner

1. Receita Mensal por Marca (ramo superior)

Este ramo calcula a evolução mensal do volume de vendas por marca.

- **Datei-me to String** – converte o campo de data para formato "ano-mês" (YYYY-MM), facilitando o agrupamento temporal.
- **GroupBy** – agrega o valor total das vendas (sale_price) por marca e mês.
- **Column Renamer** – ajusta nomes das colunas para padrões legíveis (ex.: Brand, Month, Revenue).
- **Pivot** – reorganiza a estrutura dos dados, colocando os meses como colunas para facilitar a visualização comparativa entre marcas.
- **Missing Value** – preenche meses sem registos de venda com zero, garantindo consistência cronológica.
- **Sorter** – ordena os resultados, normalmente por marca ou por período.
- **Bar Chart (JavaScript)** – gera um gráfico de barras que representa a receita mensal por marca.

- **Image Writer (Port)** – exporta o gráfico gerado para ficheiro de imagem, permitindo a sua inclusão em relatórios.

Resultado: gráfico com a receita mensal de cada marca (receita_mensal_por_marca.svg)

2. Média de Emissões de CO_2 por Marca

Este ramo calcula a média de emissões e envia o resultado para o Node-RED.

- GroupBy – calcula a média (Mean) de CO_2 para cada marca.
- Column Filter – mantém apenas as colunas relevantes (Brand, Avg_CO2).
- Column Renamer – renomeia os campos para melhor legibilidade
- Sorter – ordena as marcas por valor médio de emissões.
- Bar Chart (JavaScript) – cria um gráfico de barras com a média de CO_2 por marca.
- Image Writer (Port) – exporta o gráfico como imagem.
- Table to JSON → Column Aggregator → Column Renamer → POST Request converte a tabela final em formato JSON, agrega as linhas num único objeto, renomeia o campo principal e envia o resultado via POST Request para um endpoint Node-RED, permitindo que os indicadores ambientais sejam visualizados num dashboard interativo.

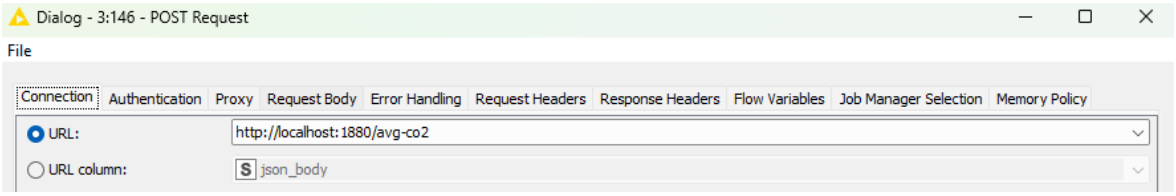


Figura 12 - POST Request

Resultado: gráfico de médias de CO_2 por marca (imagem) e envio dos dados em JSON para o Node-RED.

3. Estatísticas de Preço por Marca

O último ramo gera estatísticas descritivas sobre o preço das vendas.

- **Column Filter** – seleciona apenas as colunas Brand e sale_price.
- **GroupBy** – agrega por marca e calcula várias estatísticas: Soma, Máximo, Mínimo, Média e Contagem.
- **Column Renamer** – uniformiza os nomes das colunas resultantes.
- **Sorter** – ordena os registos por marca ou por média de preço.
- **Table View (JavaScript)** – permite inspecionar visualmente a tabela agregada.
- **Constant Value Column (deprecated)** – adiciona um campo descritivo, por exemplo metric = price_stats_by_brand.
- **Table to JSON → JSON Writer** – converte os resultados para formato JSON e grava o ficheiro localmente, criando um conjunto de dados persistente para futuras análises.

Resultado: tabela e ficheiro JSON com as principais métricas de preço por marca.

Relatórios de Qualidade

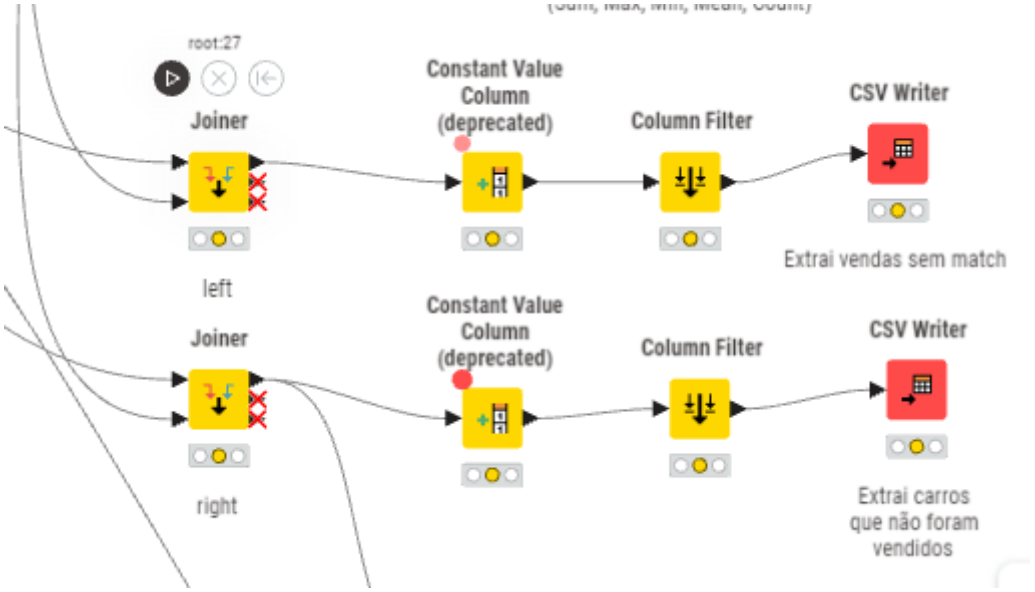


Figura 13 - Relatórios de Qualidade

1. Joiner (Left Unmatched)

Este nó recolhe todas as vendas que não possuem correspondência no dataset de informação dos carros, após a execução: é adicionada uma coluna constante (Constant Value Column) que identifica o tipo de registo, por exemplo stage = "left_only" ou observação = "venda sem correspondência". O Column Filter seleciona apenas as colunas mais relevantes (ex.: matrícula, marca, modelo, preço e data da venda).

O resultado é exportado pelo CSV Writer, originando o ficheiro "Vendas sem match.csv", que contém apenas as vendas sem veículo associado.

Resultado: ficheiro Vendas sem match.csv — lista de vendas sem correspondência no dataset de carros.

2. Joiner (Right Unmatched)

De forma análoga, este nó captura os carros que não aparecem em nenhuma venda.

O fluxo segue a mesma estrutura: Constant Value Column adiciona o campo de identificação (ex.: stage = "right_only" ou observação = "carro não vendido"). Column Filter retém as colunas essenciais (ex.: matrícula, marca, modelo, tipo de combustível, emissões, etc.).

O resultado é exportado com o CSV Writer, gerando o ficheiro "Car info sem match.csv".

Resultado: ficheiro Car info sem match.csv — lista de carros que não foram vendidos.

3.2.4 Logs

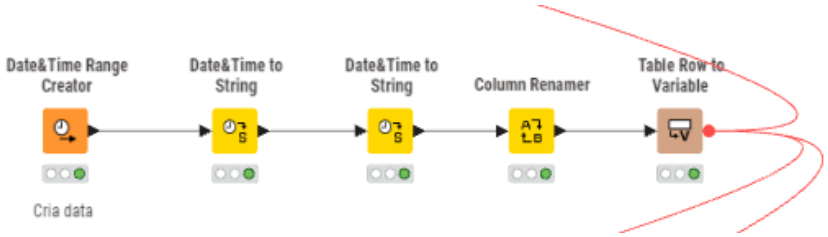


Figura 14 – Logs

Criação da Data e Variável Temporal para o Log:

Esta sequência de nós tem como objetivo gerar automaticamente a data e hora da execução do processo ETL, para que essa informação possa ser incluída nos registos do ficheiro de log (etl_log.csv).

O seu propósito é garantir que cada execução do workflow fica devidamente carimbada temporalmente, permitindo rastrear quando o processo foi realizado.

Descrição dos Nós

1. **Date&Time Range Creator**

Cria um registo contendo a data e hora atuais. Normalmente é configurado com um único valor (início = data atual, número de períodos = 1), servindo como ponto de partida temporal.Este nó gera o timestamp base do processo.

2. **Date&Time to String (1)**

Converte o valor de data/hora para o formato textual (String), permitindo manipular ou formatar a data conforme pretendido (por exemplo, yyyy-MM-dd HH:mm:ss).

3. **Date&Time to String (2)**

É utilizada uma segunda conversão quando se pretende formatar de forma diferente (por exemplo, gerar um campo apenas com a data (yyyy-MM-dd) e outro apenas com a hora (HH:mm:ss)). Desta forma, o log pode armazenar tanto a data de execução como o instante exato.

4. **Column Renamer**

Renomeia as colunas geradas, atribuindo nomes mais descritivos, como log_date e log_time. Esta etapa assegura consistência na nomenclatura das variáveis antes da sua utilização.

5. Table Row to Variable

Converte a linha de dados criada (contendo a data e hora) em variáveis de fluxo (flow variables) que são propagadas para os restantes nós do processo.

Subfluxo de log (por etapa)

1. **Extract Table Dimension** – extrai o número de linhas e colunas existentes no dataset naquele ponto do ETL.
2. **RowID** – cria um identificador de linha para permitir a transformação seguinte.
3. **Pivot** – coloca as métricas em colunas (por ex., row_count e column_count na mesma linha).
4. **Column Renamer** – padroniza os nomes das métricas (ex.: row_count, column_count).
5. **Column Resorter** – ordena as colunas para o formato final do log.

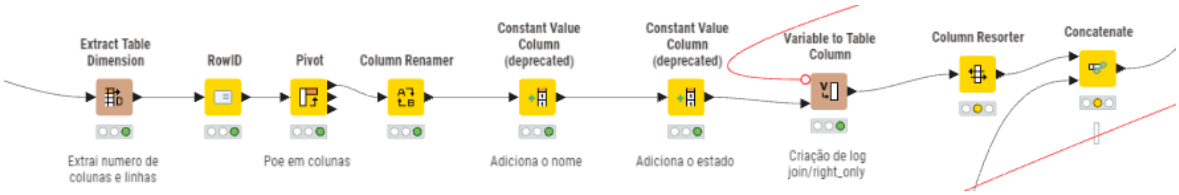


Figura 15 - Table Row to Variable

Consolidação

Os três registos (Sales Info limpo, Car Info limpo e Right Join/unmatched) são concatenados e gravados num único ficheiro etl_log.csv.

Este ficheiro permite auditoria e rastreabilidade do processo, evidenciando o efeito de cada fase (limpeza, normalização e junção) sobre o volume de dados.

3.3 Node-Red

A fase final do processo de ETL consiste em carregar os dados transformados e agregados numa plataforma de visualização. Para este fim, foi utilizado o **Node-RED**, uma ferramenta de programação baseada em fluxos que permite criar dashboard interativos de forma rápida e eficiente. Esta abordagem cumpre o critério de mais-valia de "processos de visualização dos resultados conseguidos utilizando dashboards".

O objetivo deste componente é receber os dados processados pelo KNIME e apresentá-los em múltiplos formatos gráficos para uma análise mais completa por parte do utilizador final.

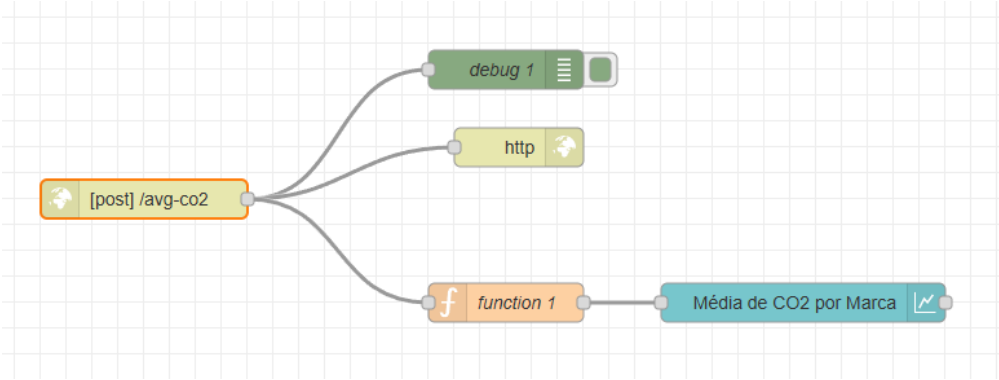


Figura 16 - Node-Red

Explicação do Fluxo (Passo a Passo):

O fluxo no Node-RED foi desenhado para receber uma única mensagem do Knime e, a partir dela, gerar um gráfico em paralelo.

1. **Processamento Paralelo (Nós de Função):** Assim que a mensagem é recebida, ela é enviada simultaneamente para dois nós function distintos. Cada nó é responsável por preparar os dados para um gráfico específico:

2. **Processamento Paralelo (Nós de Função):** Assim que a mensagem é recebida, ela é enviada o function. O mesmo é responsável por preparar os dados para um gráfico específico. Este *script* extrai a lista de dados e cria dois *arrays*: um com as marcas dos veículos (Brand) e outro com os valores da **Média de CO2** (Media_CO2). De seguida, formata estes dados para o nó de gráfico correspondente.

```
1   const dadosRecebidos = msg.payload;
2
3   const labels = [];
4   const data = [];
5
6   dadosRecebidos.forEach(function(item) {
7       labels.push(item.Brand);
8       data.push(item.avg_co2);
9   });
10
11  msg.payload = [{
12      "series": ["Média de Emissões de CO2"],
13      "data": [data],
14      "labels": labels
15  }];
16
17  return msg;
```

Figura 17 - Função

3. **Apresentação Gráfica (Nós de Gráfico):** A função envia uma mensagem, já formatada, para um nó ui_chart, rederiza um gráfico de barras com o título "Média de CO2 por Marca".

Resultado (Dashboard):

Este fluxo resulta num dashboard mais completo e eficaz, que apresenta o gráfico. Esta abordagem permite ao utilizador final analisar o desempenho ambiental de cada marca.

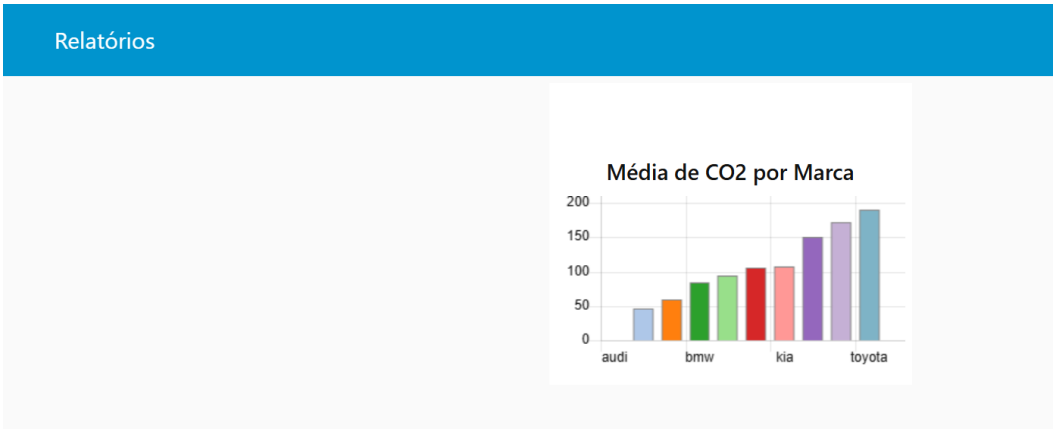


Figura 18 - Resultado (Dashboard)

3.4 Vídeo com demonstração (QR Code)

Para complementar a documentação escrita, foi produzido um vídeo que demonstra o funcionamento completo do processo de ETL. O vídeo apresenta a execução do Knime, a geração dos ficheiros de saída (incluindo os de erro) e a visualização dos resultados finais no dashboard do Node-RED.

Figura 19 - 3.4 Vídeo com demonstração (QR Code)

https://alunosipca-my.sharepoint.com/:v:/g/personal/a23015_alunos_ipca_pt/EWFqwDptrOtLlUX-K-WdLAYBgy2yDIsUbMgniom1WhJIug

3.5 Repositório GitHub

De acordo com os requisitos do enunciado, todo o material produzido durante o desenvolvimento deste trabalho foi centralizado num repositório Git. Este repositório inclui os ficheiros do projeto Knime, os ficheiros de dados de entrada, o fluxo do Node-RED (.json), e a própria documentação do projeto.

O repositório garante o controlo de versões e facilita o acesso a todos os artefactos do trabalho.

https://github.com/LinoAzevedo/Projeto_ISI_TP1

4. Apache Hop

4.1 Estratégia utilizada

Para implementar a solução, foi adotada uma estratégia baseada na utilização de duas ferramentas principais, seguindo as boas práticas de ETL e integração de sistemas.

4.1.1 Ferramentas

Apache Hop: Utilizado como a principal ferramenta de ETL para desenhar e executar todo o fluxo de dados, desde a extração até à carga. Foram criados um Pipeline (filename.hpl) para as transformações de dados e um Workflow (filename.hwf) para orquestrar a execução e a gestão de erros.

Node-RED: Utilizado para a camada de visualização. Um dashboard foi criado para receber os dados processados pelo Apache Hop e apresentá-los de forma gráfica e intuitiva.

4.1.2 Processo de ETL

O processo foi dividido em três fases macro:

1. Extração (Extract):

- Leitura e carregamento dos dados do ficheiro car_information_dataset.csv.
- Leitura e carregamento dos dados do ficheiro stand_vendas.xml.

2. Transformação (Transform):

- Filtragem inicial: Remoção de registos com campos essenciais nulos (Marca, Modelo, Matrícula).
- Limpeza e Padronização: Aplicação de operações de string para remover espaços, converter para minúsculas/maiúsculas e limpar caracteres indesejados de campos numéricos.
- Correção de Dados: Utilização do operador Value Mapper para normalizar nomes de marcas (ex: "mb" para "mercedes") e tipos de combustível.

- Validação: Uso de Expressões Regulares para validar o formato das matrículas, separando as válidas das inválidas.
- Enriquecimento e Cálculos: Conversão de unidades de milhas para quilómetros e interpretação de múltiplos formatos de data através de um script JavaScript.
- Integração: Junção dos dois fluxos de dados (Merge Join) com base na matrícula (License_Plate) para associar os dados de venda aos dados técnicos de cada veículo.
- Agregação: Agrupamento dos dados por marca (Group By) para calcular KPIs (Key Performance Indicator (Indicador-Chave de Desempenho)) como o número de vendas, soma e média de emissões de CO2, e valores mínimo, máximo e médio de venda.

3. Carga (Load):

- Saídas de Erro: Os dados que falham nas validações são exportados para ficheiros XML específicos (matriculas_invalidas.xml, null_car.xml, null_vendas.xml, carros_nao_vendidos.xml), criando um registo claro dos problemas de qualidade de dados.
- Envio para Dashboard: Os dados agregados e processados são formatados em JSON e enviados através de um cliente REST para um endpoint em Node-RED (http://localhost:1880/dados-hop).
- Visualização: O fluxo em Node-RED recebe os dados, extrai as informações relevantes e alimenta os gráficos de barras que são exibidos, sendo eles a média de CO2 por marca e a media de preço por marca.

4.2 Transformações

A lógica de transformação de dados foi implementada no pipeline filename.hpl através da ferramenta Apache Hop. Este pipeline é responsável por executar todas as operações de limpeza, validação, enriquecimento e integração dos dados.

Para uma melhor compreensão, o processo foi dividido em quatro fases lógicas, que são executadas em paralelo e sequencialmente.

Diagrama Geral do Pipeline:

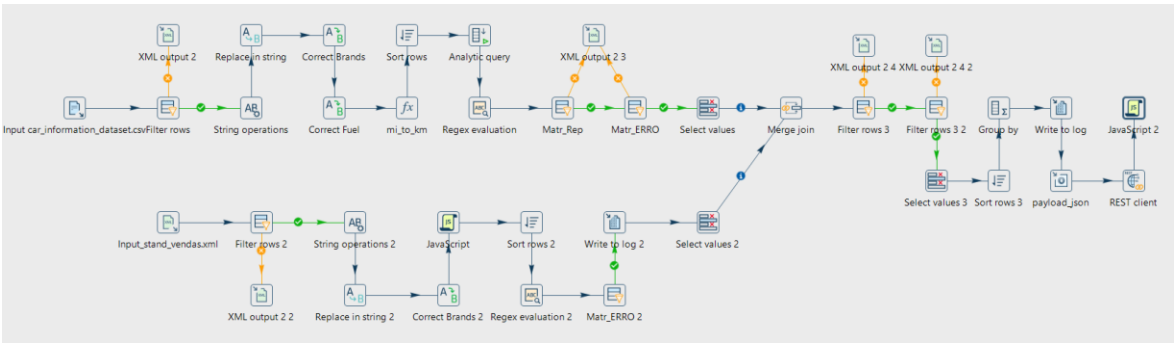


Figura 20 - Diagrama Apache Hop

4.2.1 Fase 1: Extração e Preparação dos Dados de Inventário (Ficheiro CSV)

Esta fase foca-se na leitura e tratamento dos dados provenientes do ficheiro car_information_dataset.csv. O objetivo é validar, limpar e padronizar os dados do inventário de veículos.

Diagrama da Fase 1:

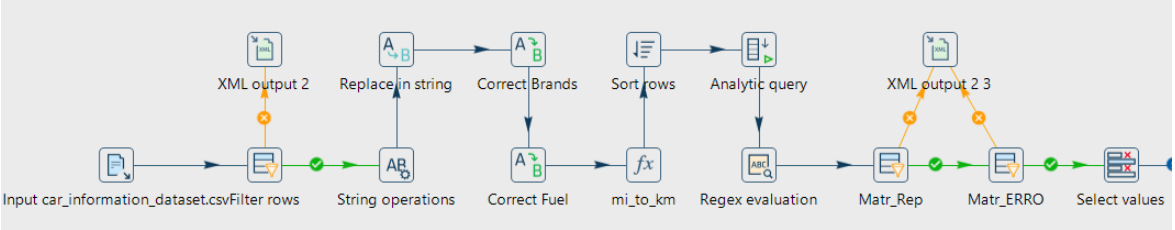


Figura 21 - Diagrama da Fase 1

Passos Detalhados:

1. **Extração (Input car_information_dataset.csv):** O processo inicia-se com a leitura dos dados do ficheiro CSV, que contém o inventário dos veículos

#	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	Brand	String		10		$	,		none
2	Model	String		8		$	,		none
3	Year	Date	yyyy	15	0	$	,		none
4	Engine_Size	String		15	1	$	.	,	none
5	Fuel_Type	String		8		$	,		none
6	Transmission	String		14		$	,		none
7	Mileage	Integer	#	15	0	$	,		none
8	Doors	Integer	#	15	0	$	,		none
9	Owner_Count	Integer	#	15	0	$	,		none
10	Price	Integer	#	15	0	$	,		none
11	Pollution	String		6		$	,		none
12	CO2_Emissions	String	#	15	0	$	,		none
13	License_Plate	String							none

Figura 22 - Input car_information_dataset.csv

2. **Filtragem de Nulos (Filter rows):** O primeiro passo de validação consiste em verificar se os campos essenciais (Brand, Model, License_Plate) contêm valores. Registos que falhem esta validação são desviados para o ficheiro null_car.xml, garantindo que apenas dados completos prossigam no fluxo principal.

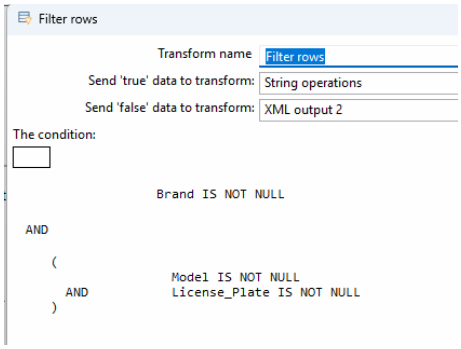
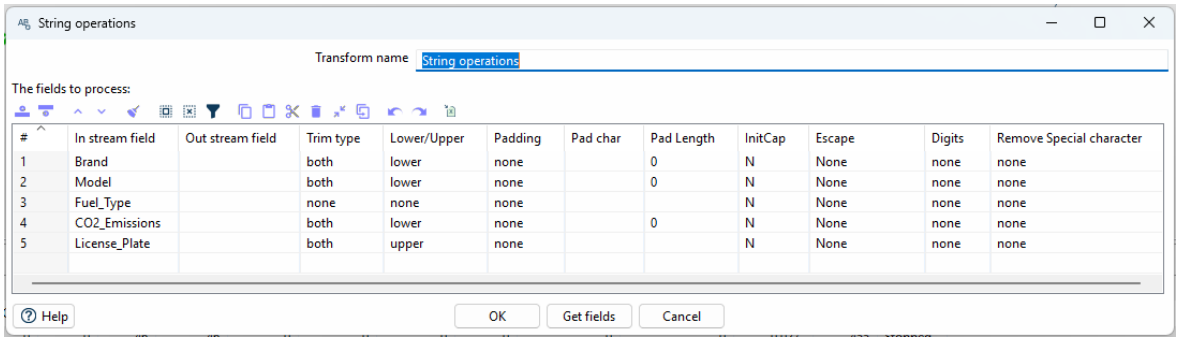


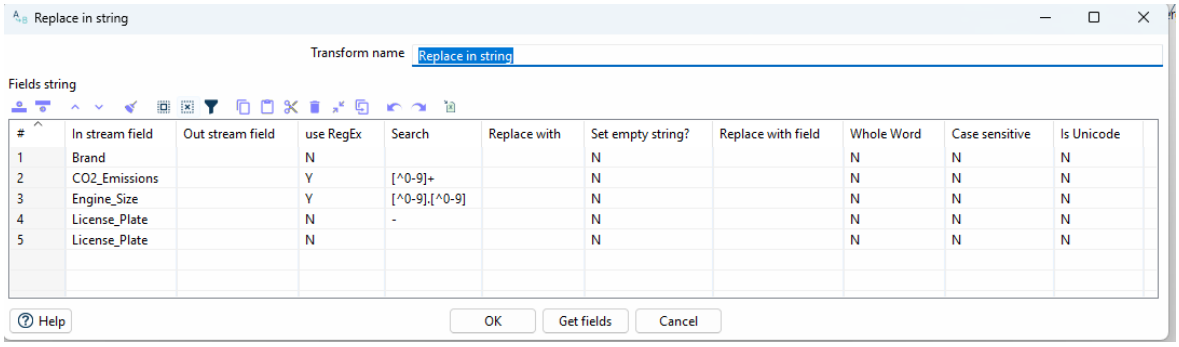
Figura 23 - Filter rows

3. **Limpeza e Padronização (String operations & Replace in string):** São aplicadas transformações para garantir a consistência dos dados. Isto inclui:
    - Remover espaços em branco no início e no fim dos campos.
    - Converter o texto para maiúsculas ou minúsculas (ex: matrículas para maiúsculas).
    - Remover caracteres não numéricos de campos como CO2_Emissions e remover hífens das matrículas.



#	In stream field	Out stream field	Trim type	Lower/Upper	Padding	Pad char	Pad Length	InitCap	Escape	Digits	Remove Special character
1	Brand		both	lower	none		0	N	None	none	none
2	Model		both	lower	none		0	N	None	none	none
3	Fuel_Type		none	none	none			N	None	none	none
4	CO2_Emissions		both	lower	none		0	N	None	none	none
5	License_Plate		both	upper	none			N	None	none	none

Figura 24 - String Operation



#	In stream field	Out stream field	use RegEx	Search	Replace with	Set empty string?	Replace with field	Whole Word	Case sensitive	Is Unicode
1	Brand		N			N		N	N	N
2	CO2_Emissions		Y	[^0-9]+		N		N	N	N
3	Engine_Size		Y	[^0-9].[^0-9]		N		N	N	N
4	License_Plate		N	-		N		N	N	N
5	License_Plate		N			N		N	N	N

Figura 25 - Replace in string

4. **Correção de Dados (Correct Brands & Correct Fuel):** Utiliza-se o componente Value Mapper para normalizar valores. Por exemplo, a abreviatura "mb" é corrigida para "mercedesbenz" e "Gasoleo" para "Diesel", criando um dicionário de dados consistente.

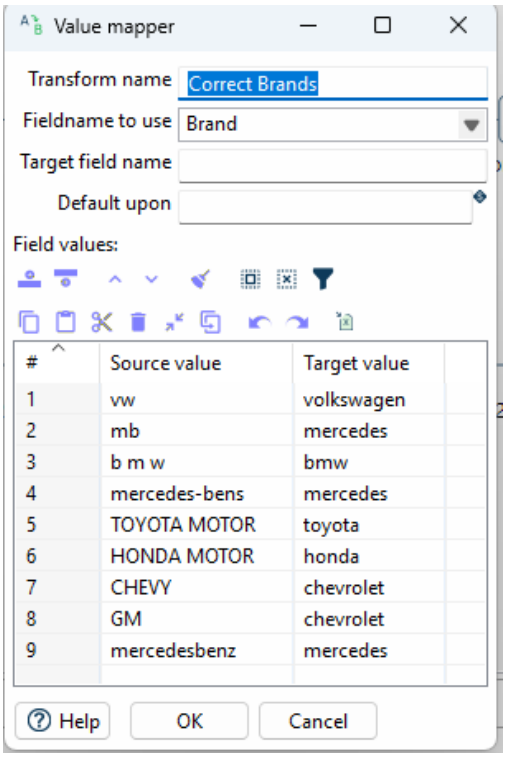


Figura 26 - Correct Brands

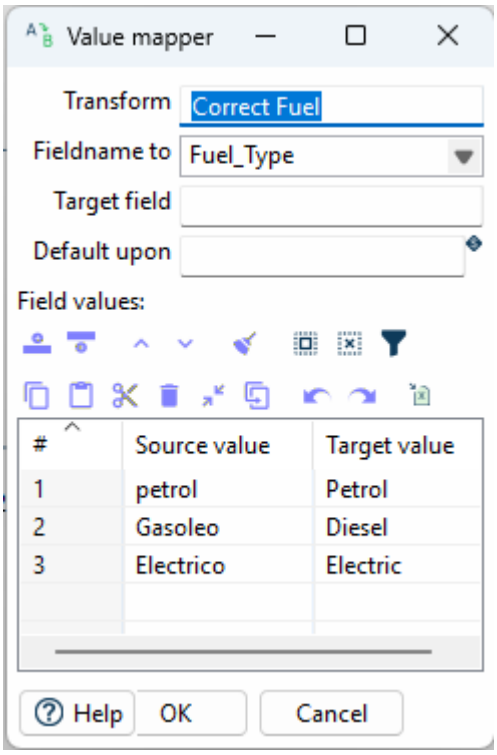


Figura 27 - Correct Fuel

5. **Enriquecimento (mi_to_km):** O campo Mileage (em milhas) é convertido para quilómetros através de uma fórmula matemática ([Mileage] * 1.60934), criando um novo campo chamado Quilometragem_KM.

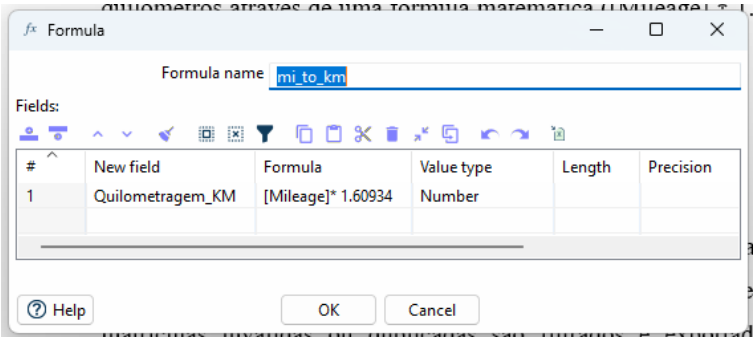


Figura 28 - mi_to_km

6. **Validação de Matrículas (Regex evaluation):** É aplicada uma Expressão Regular complexa para validar o formato das matrículas portuguesas. Registos com matrículas inválidas ou duplicadas são filtrados e exportados para o ficheiro matriculas_invalidas.xml, isolando-os do fluxo principal.

Sort rows – Ordena-se por matrícula

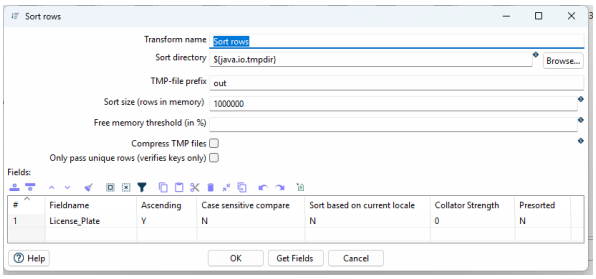


Figura 29 - Sort rows

⬇

Analytic query – Cria uma nova coluna (Matricula_Anterior) com o valor da matrícula da linha anterior (colocando a primeira como null)

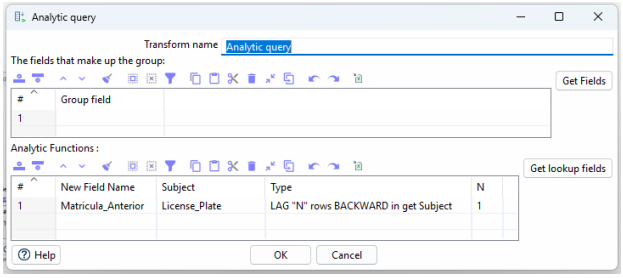


Figura 30 - Analytic query

⬇

Regex evaluation – valida se a matricula esta no formato exigido, e guarda essa informação em uma coluna boolean (formato_valido)

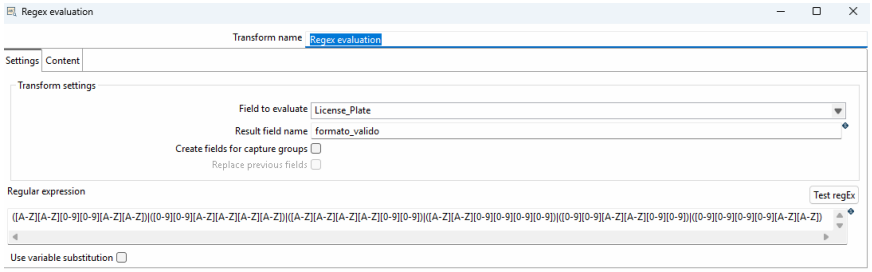


Figura 31 - Regex evaluation

⬇

Filter rows – validam-se:

Matricula_Anterior é diferente da Licence_Plate

formato_valido é igual a true

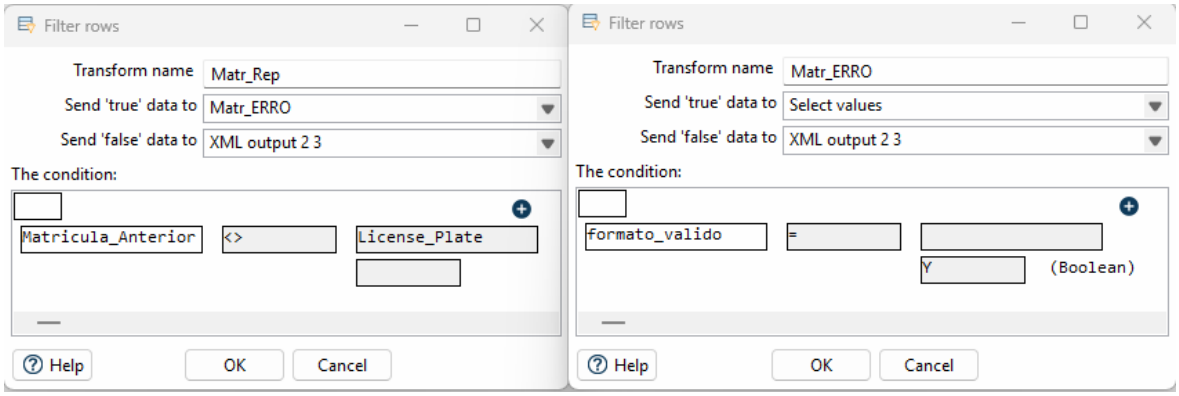


Figura 32 - Filter rows

7. **Seleção de Dados (Select values):** Remover as colunas desnecessárias (formato_valido, Matricula_Anterior e Mileage) e convertendo strings para inteiros (Engine_Size e CO2_Emissions).

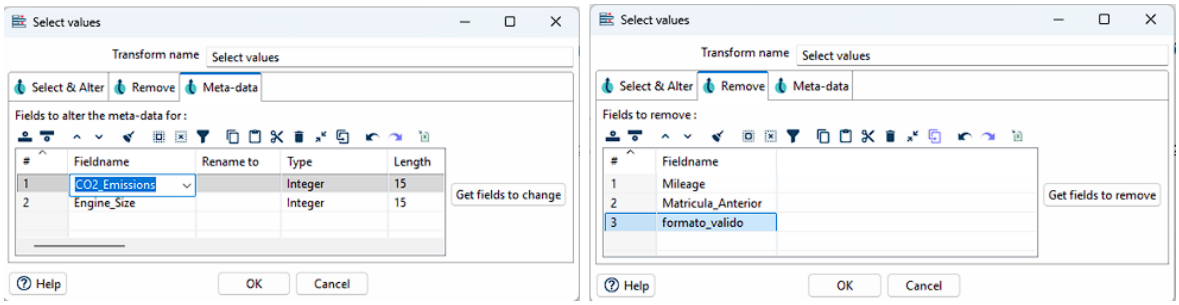


Figura 33 - Select values

4.2.2 Fase 2: Extração e Preparação dos Dados de Vendas (Ficheiro XML)

De forma paralela à Fase 1, este fluxo processa os registos de vendas do ficheiro stand_vendas.xml.

Diagrama da Fase 2:

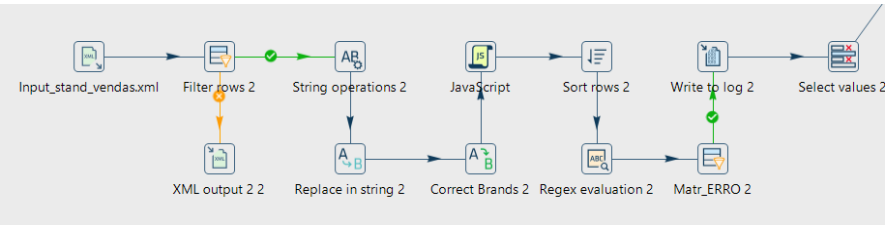
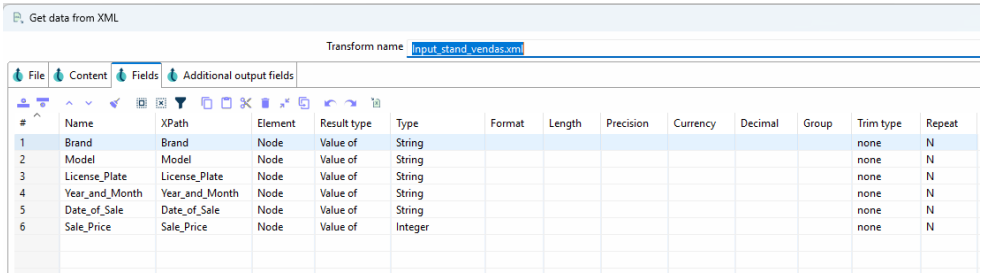


Figura 34 - Diagrama da Fase 2

Passos Detalhados:

1. **Extração (Input_stand_vendas.xml):** O processo inicia-se com a leitura dos dados de vendas, que estão estruturados em formato XML.



#	Name	XPath	Element	Result type	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type	Repeat
1	Brand	Brand	Node	Value of	String							none	N
2	Model	Model	Node	Value of	String							none	N
3	License_Plate	License_Plate	Node	Value of	String							none	N
4	Year_and_Month	Year_and_Month	Node	Value of	String							none	N
5	Date_of_Sale	Date_of_Sale	Node	Value of	String							none	N
6	Sale_Price	Sale_Price	Node	Value of	Integer							none	N

Figura 35 - Input_stand_vendas.xml

2. **Filtragem de Nulos (Filter rows 2):** Tal como no fluxo do inventário, os registos com campos essenciais nulos são filtrados e enviados para um ficheiro de erro, o null_vendas.xml.

3. **Limpeza e Correção (String operations 2, Replace in string 2, Correct Brands 2):** São aplicadas as mesmas lógicas de limpeza e correção de marcas vistas na Fase 1, garantindo que os dados de ambos os fluxos sejam comparáveis.

4. **Tratamento de Datas (JavaScript):** Um dos maiores desafios era a inconsistência no formato das datas de venda (Date_of_Sale). Foi utilizado um *script* JavaScript para analisar múltiplos formatos possíveis (Ex: "yyyy-MM-dd", "dd/MM/yyyy") e convertê-los para um formato de data único e válido.

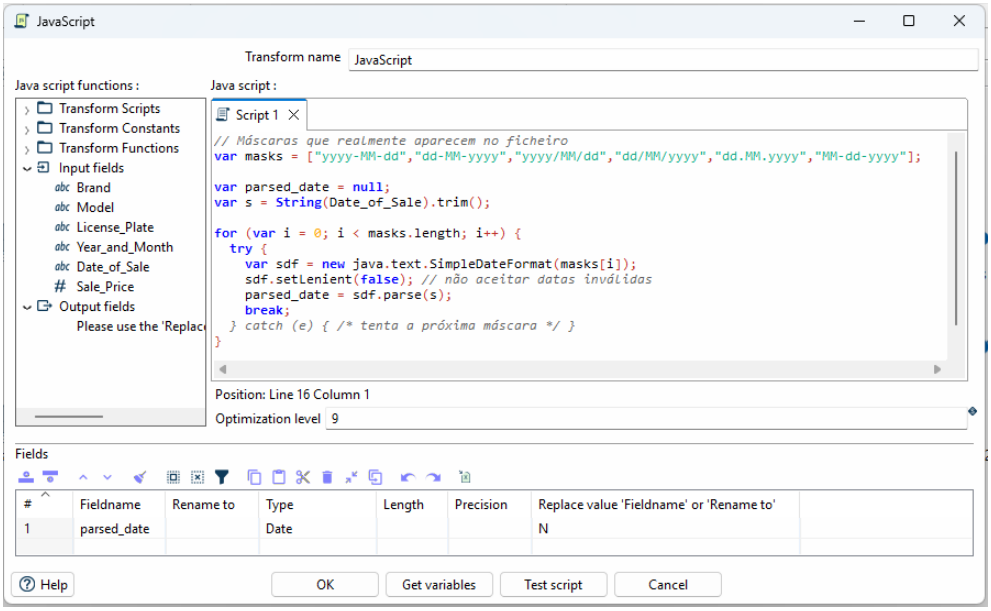


Figura 36 - JavaScript

5. **Validação de Matrículas (Regex evaluation 2):** A mesma validação por Expressão Regular é aplicada às matrículas do ficheiro de vendas para garantir a sua integridade. Aqui não se verifica a repetição de matrículas visto que um carro pode ser vendido mais de uma vez (retorno ou devolução).

6. **Seleção de Dados (Select values):** a semelhança do anterior aqui também se remove as colunas desnecessárias (formato_valido) e convertendo strings para datas.

4.2.3 Fase 3: Junção dos Dados e Análise de Discrepâncias

Nesta fase, os dois fluxos de dados, já limpos e preparados, são combinados. O objetivo é criar uma visão unificada e identificar discrepâncias entre o inventário e as vendas.

Diagrama da Fase 3:

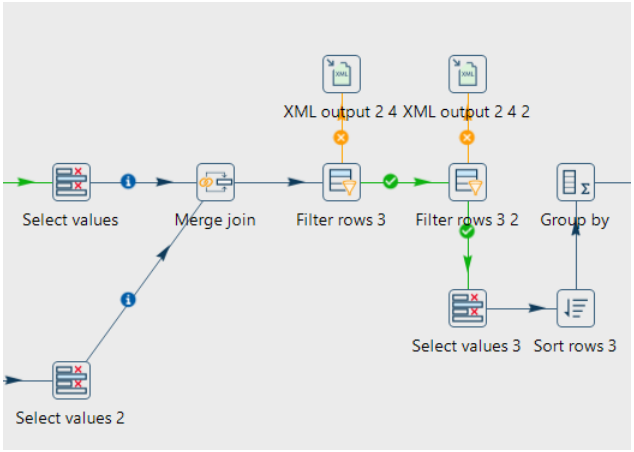


Figura 37 - Diagrama da Fase 3

Passos Detalhados:

1. **Junção (Merge join):** Os dois fluxos de dados são unidos utilizando a License_Plate (matrícula) como chave de ligação. Foi utilizado um FULL OUTER JOIN, que mantém todos os registos de ambas as fontes, mesmo que não encontrem correspondência.

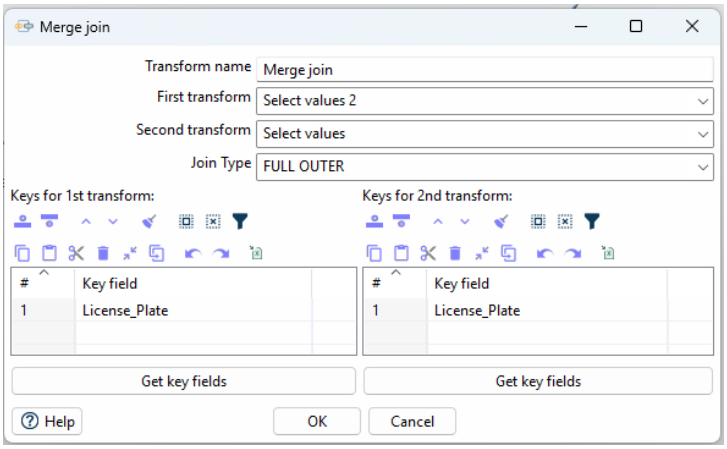


Figura 38 - Merge join

2. **Identificação de Discrepâncias (Filter rows 3 & Filter rows 3 2):** Após a junção, os filtros são usados para analisar o resultado, a semelhança dos primeiros filtros estes servem para remover linhas que ficaram nulas pós a junção:
    - **Carros não vendidos:** Registos que existem no fluxo de inventário, mas não no de vendas (identificados por terem o campo Brand do fluxo de vendas a nulo) são enviados para o ficheiro carros_nao_vendidos.xml.
    - **Vendas sem carro correspondente:** Registos que existem no fluxo de vendas, mas não no de inventário são enviados para o ficheiro vendas_sem_carros.xml.

3. **Seleção de Dados Válidos:** Apenas os registos que tiveram correspondência em ambas as fontes (ou seja, vendas de carros que existem no inventário) prosseguem para a fase final.

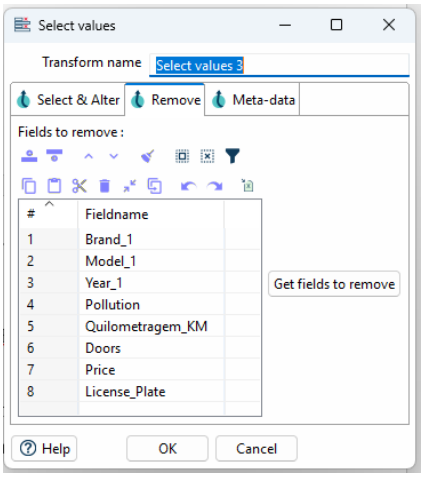


Figura 39 - Seleção de Dados Válidos

4.2.4 Fase 4: Agregação, Geração de Saídas e Carga

Esta é a fase final, onde os dados consolidados são agregados para gerar métricas de negócio e, subsequentemente, carregados no dashboard de visualização.

Diagrama da Fase 4:

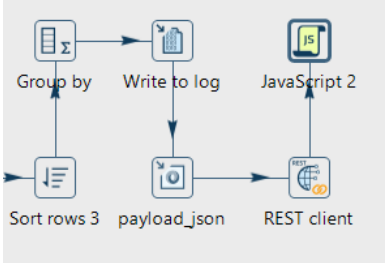


Figura 40 - Diagrama da Fase 4

Passos Detalhados:

1. **Agregação de Dados (Group by):** Os dados são agrupados por Brand (marca) para calcular métricas de alto nível, tais como:
    - Número total de vendas (COUNT_ALL).
    - Soma e média das emissões de CO2 (SUM, AVERAGE).
    - Preço de venda mínimo, máximo e médio (MIN, MAX, AVERAGE).

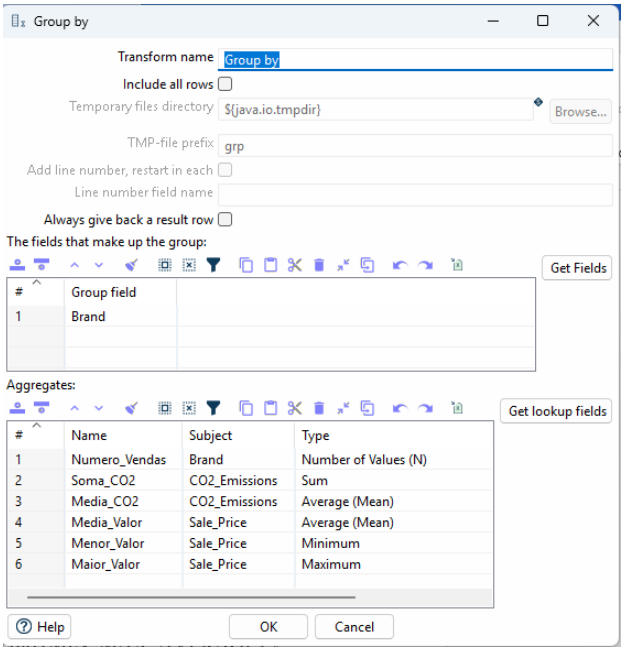


Figura 41 - Group by

2. **Formatação para JSON (payload_json):** Os resultados agregados são convertidos e exportados em formato JSON, preparando-os para serem enviados através de uma API.

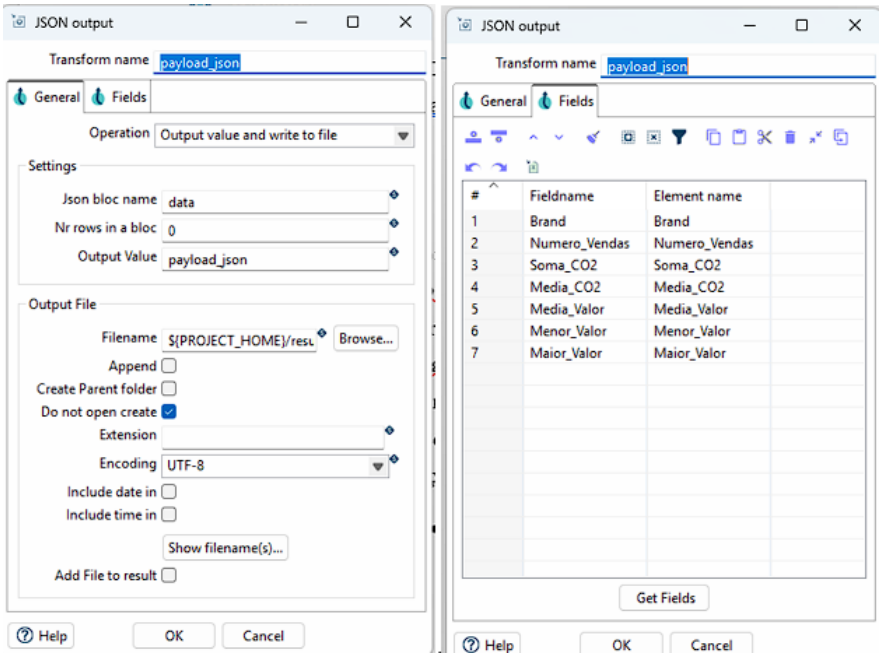


Figura 42 - payload_json

3. **Carga (REST client):** Os dados em JSON são enviados, através de um pedido POST, para um *endpoint* da API do Node-RED (http://localhost:1880/dados-hop), que está à escuta para receber esta informação.

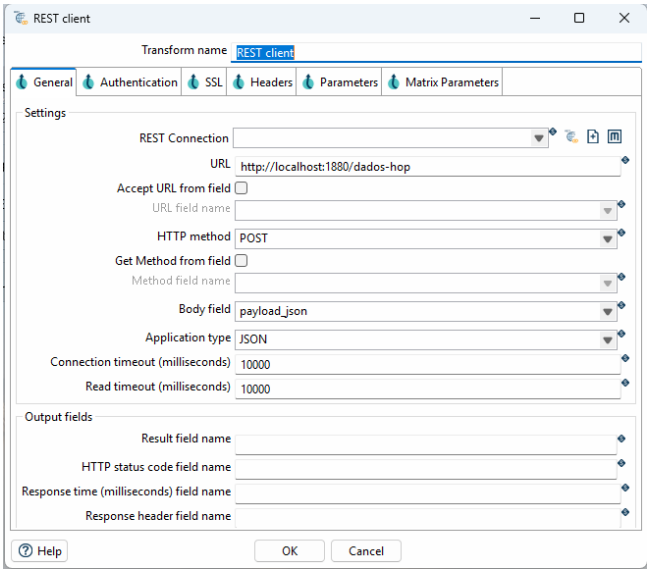


Figura 43 - REST cliente

4. **Abertura do Dashboard (JavaScript 2):** Como passo final de conveniência, um pequeno *script* JavaScript é executado para abrir automaticamente o navegador do utilizador no endereço do dashboard do Node-RED (http://localhost:1880/ui/), permitindo a visualização imediata dos resultados.

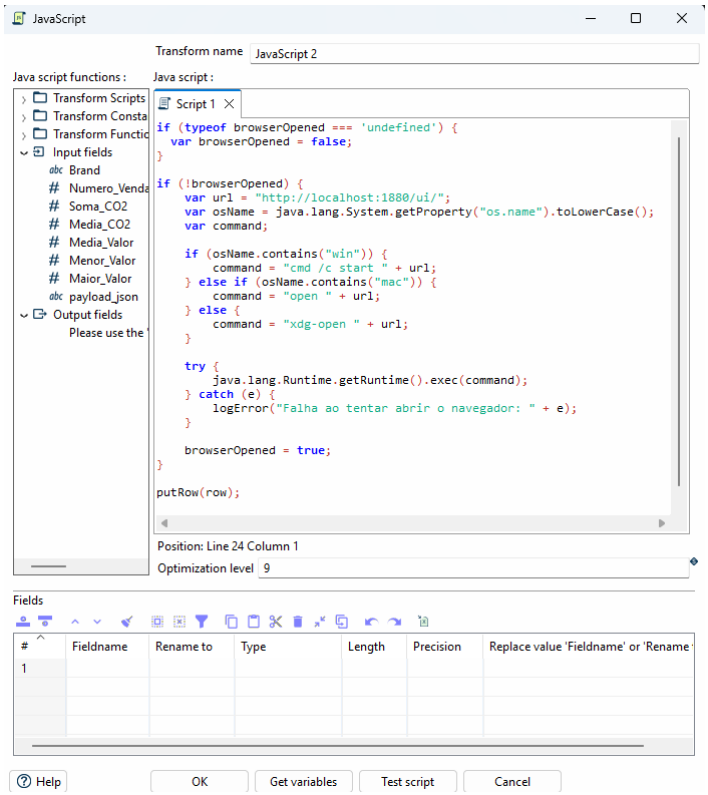
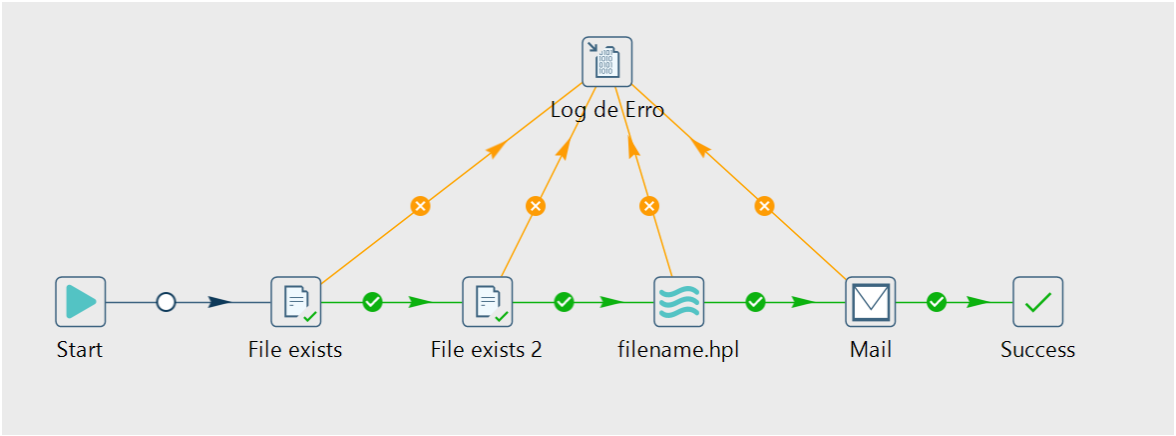


Figura 44 - JavaScript 2

4.3 Jobs

Enquanto o Pipeline se foca na transformação dos dados, o Workflow (ou Job) é responsável por orquestrar todo o processo. No ficheiro filename.hwf, foi definido um fluxo de controlo que automatiza a execução, verifica condições e lida com possíveis erros, garantindo que o processo é robusto e fiável.

Diagrama do Workflow:



Explicação do Fluxo: O workflow garante que o pipeline é executado de forma controlada e robusta:

1. **Início:** O processo é iniciado pela ação Start.

2. **Verificação de Pré-requisitos:** O workflow verifica sequencialmente se os ficheiros de entrada (car_information_dataset.csv e stand_vendas.xml) existem nos seus respetivos diretórios.

3. **Execução Condicional:**
   - Se ambos os ficheiros existirem, o workflow executa o pipeline de transformações (filename.hpl).
   - Se algum dos ficheiros não for encontrado, o fluxo é desviado para uma ação de erro.

4. **Gestão de Sucesso e Erro:**
    - Em caso de execução bem-sucedida do pipeline, é enviada uma **notificação por email** para a22997@alunos.ipca.pt a confirmar a conclusão do processo, e o workflow termina com sucesso.
    - Se ocorrer qualquer falha durante o processo (ficheiro inexistente ou erro no pipeline), o fluxo é redirecionado para a ação Log de Erro, que regista uma mensagem detalhada sobre a falha, incluindo o nome do job e do componente que falhou.

4.4 Node-Red

A fase final do processo de ETL consiste em carregar os dados transformados e agregados numa plataforma de visualização. Para este fim, foi utilizado o **Node-RED**, uma ferramenta de programação baseada em fluxos que permite criar dashboards interativos de forma rápida e eficiente. Esta abordagem cumpre o critério de mais-valia de "processos de visualização dos resultados conseguidos utilizando dashboards".

O objetivo deste componente é receber os dados processados pelo Apache Hop e apresentá-los em múltiplos formatos gráficos para uma análise mais completa por parte do utilizador final.

Diagrama do Fluxo Node-RED:

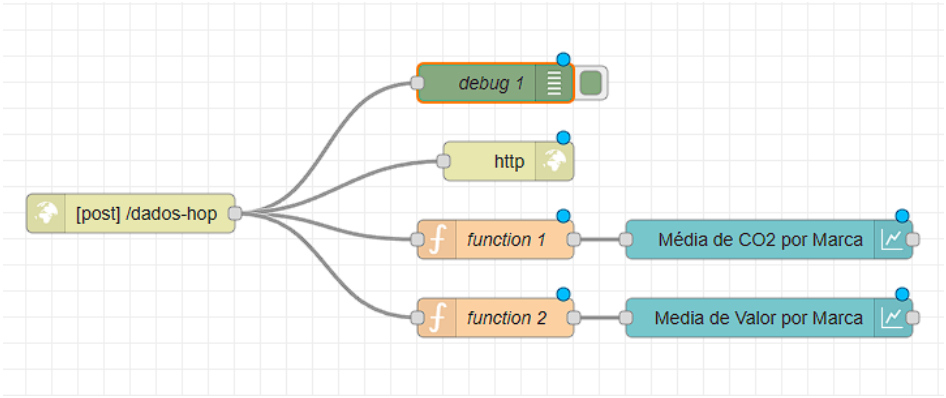


Figura 45 - Diagrama do Fluxo Node-RED

Explicação do Fluxo (Passo a Passo):

O fluxo no Node-RED foi desenhado para receber uma única mensagem do Hop e, a partir dela, gerar múltiplos gráficos em paralelo.

1. **Receção dos Dados (Endpoint HTTP):** O fluxo é iniciado pelo nó http in, que está configurado para ouvir pedidos do tipo POST no *endpoint* /dados-hop. É neste endereço que o cliente REST do Apache Hop envia os dados agregados em formato JSON.

2. **Processamento Paralelo (Nós de Função):** Assim que a mensagem é recebida, ela é enviada simultaneamente para dois nós function distintos. Cada nó é responsável por preparar os dados para um gráfico específico:

   - **Função para o Gráfico de CO2 (function 1):** Este *script* extrai a lista de dados e cria dois *arrays*: um com as marcas dos veículos (Brand) e outro com os valores da **Média de CO2** (Media_CO2). De seguida, formata estes dados para o nó de gráfico correspondente.
   - **Função para o Gráfico de Valor (function 2):** Este *script* executa uma tarefa similar, mas extrai os valores da **Média de Valor** de venda (Media_Valor), preparando os dados para o segundo gráfico.

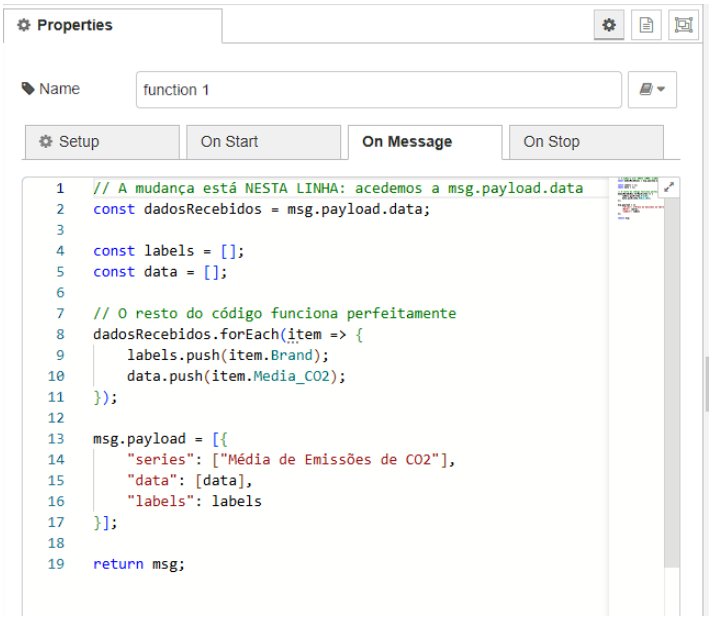


Figura 46 - function 1

3. **Apresentação Gráfica (Nós de Gráfico):** Cada nó de função envia a sua mensagem, já formatada, para um nó ui_chart dedicado:

   - O primeiro nó renderiza um gráfico de barras com o título "Média de CO2 por Marca".
   - O segundo nó renderiza outro gráfico de barras com o título "Media de Valor por Marca".
   - Ambos os gráficos são automaticamente apresentados na mesma interface do utilizador, numa aba designada "Relatórios".

Resultado (Dashboard):

Este fluxo resulta num dashboard mais completo e eficaz, que apresenta dois gráficos distintos. Esta abordagem permite ao utilizador final não só analisar o desempenho ambiental de cada marca, mas também cruzar essa informação com o valor médio de venda, permitindo uma análise de negócio mais rica e informada.

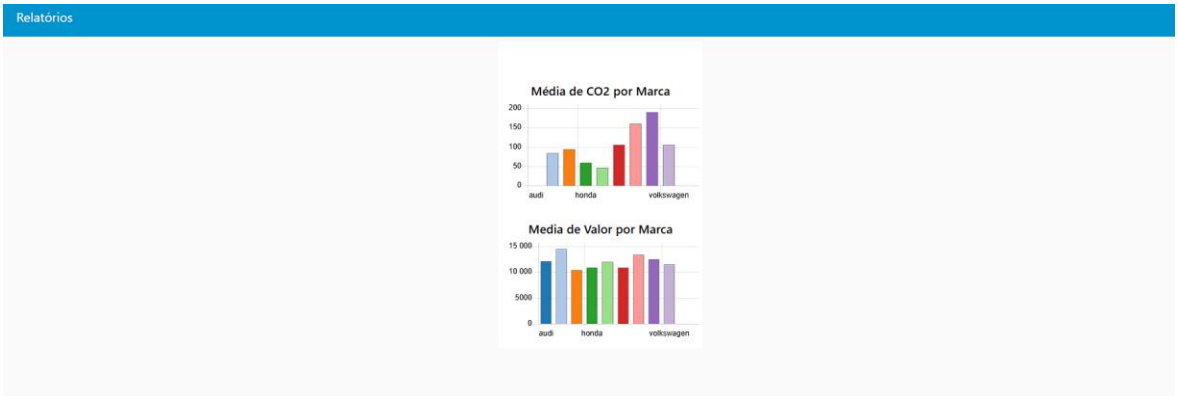


Figura 47 - Resultado (Dashboard)

4.5 Vídeo com demonstração (QR Code)

Para complementar a documentação escrita, foi produzido um vídeo que demonstra o funcionamento completo do processo de ETL. O vídeo apresenta a execução do workflow no Apache Hop, a geração dos ficheiros de saída (incluindo os de erro) e a visualização dos resultados finais no dashboard do Node-RED.

Figura 48 - QRCode Demonstração Apache Hop

https://alunosipca-my.sharepoint.com/:v:/g/personal/a22997_alunos_ipca_pt/Ea7394t0NudHl4-Iw3NyTqoBsLsvC0VknLMu7l6fM6mfwA?e=FJNZGh

4.6 Repositório GitHub

De acordo com os requisitos do enunciado, todo o material produzido durante o desenvolvimento deste trabalho foi centralizado num repositório Git. Este repositório inclui os ficheiros do projeto Apache Hop (.hpl, .hwf), os ficheiros de dados de entrada, o fluxo do Node-RED (.json), e a própria documentação do projeto.

O repositório garante o controlo de versões e facilita o acesso a todos os artefactos do trabalho.

https://github.com/fabiocosta191/Projeto_ISI_Tb1

5. Considerações Finais

A realização deste trabalho prático permitiu aplicar, de forma abrangente e aprofundada, os conceitos teóricos da Unidade Curricular de Integração de Sistemas de Informação. O objetivo principal, o desenvolvimento de um processo de ETL completo, foi alcançado com sucesso através da utilização das ferramentas como o KNIME e Apache Hop, demonstrando a capacidade de resolver o mesmo problema através de abordagens distintas.

Ao longo do projeto, foi possível simular um cenário de negócio realista, lidando com desafios como a heterogeneidade das fontes (CSV e XML) e a má qualidade dos dados. A robustez do processo foi validada pela capacidade de isolar sistematicamente diferentes tipos de anomalias em ficheiros de auditoria dedicados, como registos com dados nulos ou matrículas inválidas. É de notar, contudo, uma limitação prática: a validação de matrículas, baseada em Expressões Regulares, foi desenhada especificamente para os formatos portugueses, requerendo adaptação para outros contextos.

A integração final com o Node-RED, ao apresentar visualmente a relação entre a "Média de CO2" e a "Media de Valor" por marca, transcendeu a simples extração de dados, transformando-os em inteligência acionável. Este dashboard permite que um gestor tome decisões estratégicas sobre quais marcas promover, equilibrando o desempenho ambiental com a rentabilidade.

Como trabalhos futuros, sugere-se a integração com uma base de dados relacional para a persistência dos dados limpos, o desenvolvimento de dashboards mais interativos e a configuração dos workflows para uma execução agendada e totalmente automatizada, aproximando a solução de um ambiente de produção real.

Em suma, o projeto demonstra com sucesso como um processo de ETL bem estruturado não só garante a qualidade dos dados, mas também converte informação bruta em valor de negócio, estabelecendo uma base sólida para futuras otimizações.

6. Referencias bibliográfica

Analise de Vendas de Carros. (n.d.). Retrieved October 16, 2025, from https://www.kaggle.com/datasets/leonardobolek/analise-de-vendas-de-carros

Apache Hop - Hop. (n.d.). Retrieved October 16, 2025, from https://hop.apache.org/

apache/hop: Hop Orchestration Platform. (n.d.). Retrieved October 16, 2025, from https://github.com/apache/hop

Conjunto de dados de preços de carros. (n.d.). Retrieved October 16, 2025, from https://www.kaggle.com/datasets/mos3santos/conjunto-de-dados-de-preos-de-carros

Java Downloads | Oracle Europe. (n.d.). Retrieved October 16, 2025, from https://www.oracle.com/europe/java/technologies/downloads/

Low-code programming for event-driven applications : Node-RED. (n.d.). Retrieved October 16, 2025, from https://nodered.org/

O que é ETL? – Explicação sobre extrair, transformar e carregar – AWS. (n.d.). Retrieved October 16, 2025, from https://aws.amazon.com/pt/what-is/etl/

O que é KNIME? Um guia introdutório | DataCamp. (n.d.). Retrieved October 16, 2025, from https://www.datacamp.com/pt/blog/what-is-knime

Open for Innovation | KNIME. (n.d.). Retrieved October 16, 2025, from https://www.knime.com/