

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ - UTFPR

CAMPUS DOIS VIZINHOS

ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS

BRUNA MATTIOLI DE OLIVEIRA

GABRIEL ANDRADE VARGA

**PROJETO INTEGRADOR 3**

DOIS VIZINHOS

OUTUBRO / 2021

## SUMÁRIO

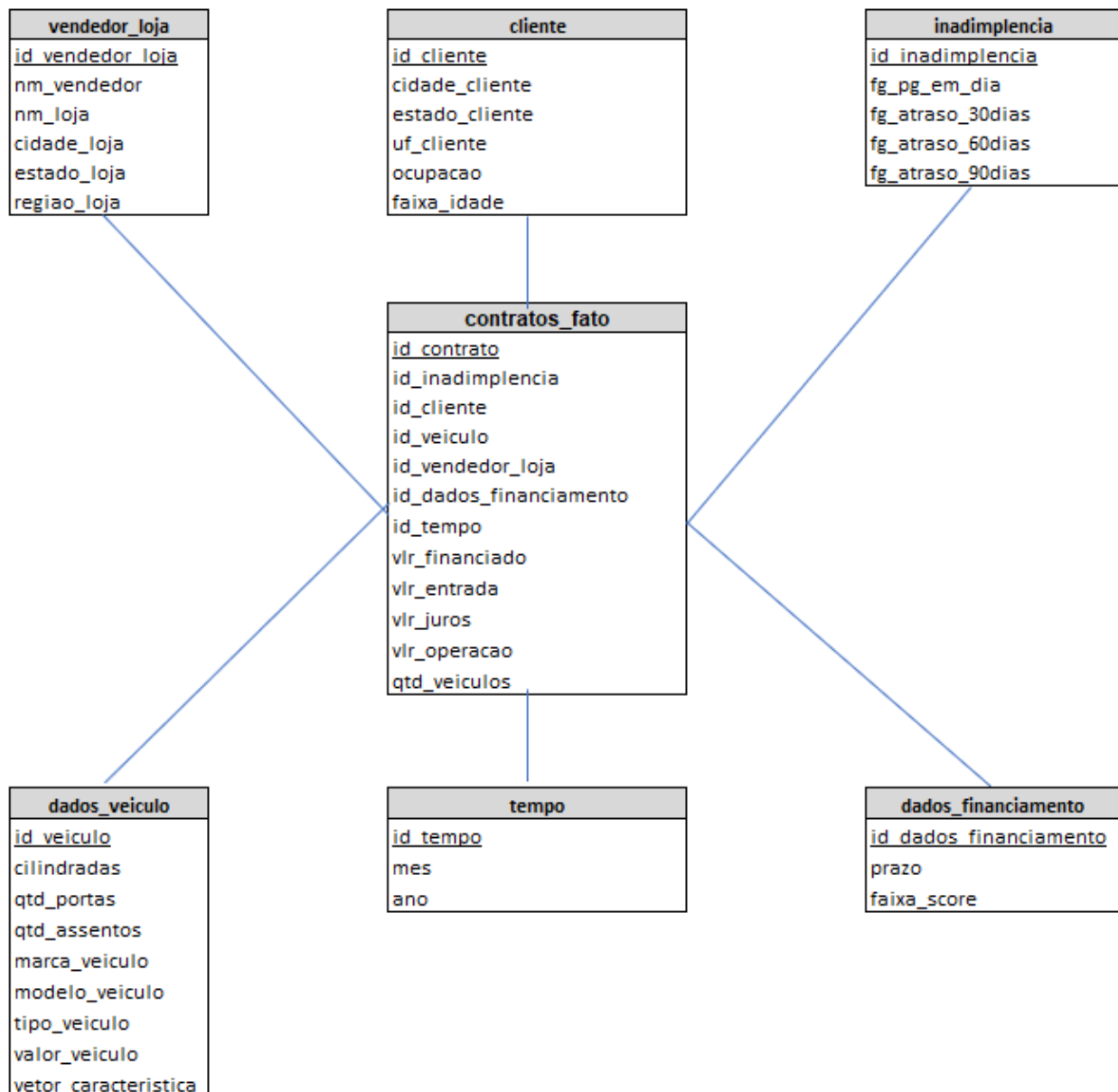
PROCESSAMENTO ANALÍTICO DE DADOS	3
RECUPERAÇÃO DE INFORMAÇÃO BASEADA EM CONTEÚDO	13
INTRODUÇÃO AO BIG DATA	17

# 1. Processamento Analítico de Dados

## a. Dados utilizados

Os dados utilizados para a elaboração deste data warehouse são informações geradas com base num hipercubo de dados de uma empresa financeira que faz contratos de financiamentos de veículos de luxo.

O data warehouse foi projetado conforme o projeto lógico do modelo dimensional abaixo:



O formato utilizado foi o esquema estrela, onde temos a tabela fato no centro do diagrama (contratos\_fato) e um total de seis dimensões não normalizadas (vendedor\_loja, cliente, inadimplencia, dados\_veiculo, tempo e dados\_financiamento) complementando as informações necessárias para a análise do negócio.

Foram gerados aleatoriamente 1.800 contratos de financiamentos no período de Agosto de 2020 a Agosto de 2021 (13 meses). Estes contratos podem ter um ou mais veículos financiados. A estrutura desse data warehouse é muito similar a de um grande banco privado brasileiro que faz aproximadamente 20% de todos os financiamentos de veículos do país.

O projeto físico foi desenvolvido para implantação no PostgreSQL e está descrito abaixo:

```
-- Script de criação das bases de dados do Data Warehouse no PostgreSQL
```

```
-- Dimensão vendedor_loja
```

```
CREATE TABLE vendedor_loja (  
    id_vendedor_loja SERIAL PRIMARY KEY,  
    nm_vendedor VARCHAR,  
    nm_loja VARCHAR,  
    cidade_loja VARCHAR,  
    estado_loja VARCHAR,  
    regioao_loja VARCHAR  
);
```

```
-- Dimensão cliente
```

```
CREATE TABLE cliente (  
    id_cliente SERIAL PRIMARY KEY,  
    cidade_cliente VARCHAR,  
    estado_cliente VARCHAR,  
    uf_cliente VARCHAR,  
    faixa_idade VARCHAR,  
    ocupacao VARCHAR  
);
```

```
-- Dimensão inadimplencia
```

```
CREATE TABLE inadimplencia (  
    id_inadimplencia SERIAL PRIMARY KEY,  
    fg_pg_em_dia INTEGER,  
    fg_atraso_30dias INTEGER,  
    fg_atraso_60dias INTEGER,  
    fg_atraso_90dias INTEGER  
);
```

```
-- Dimensão dados_veiculo
```

```
CREATE TABLE dados_veiculo (  
    id_veiculo SERIAL PRIMARY KEY,
```

```

    cilindradas NUMERIC(30,2),
    qtd_portas INTEGER,
    qtd_assentos INTEGER,
    marca_veiculo VARCHAR,
    modelo_veiculo VARCHAR,
    tipo_veiculo VARCHAR,
    valor_veiculo NUMERIC(30,2),
    vetor_caracteristica FLOAT[]
);

-- Dimensão tempo
CREATE TABLE tempo (
    id_tempo SERIAL PRIMARY KEY,
    mes VARCHAR,
    ano INTEGER
);

-- Dimensão dados_financiamento
CREATE TABLE dados_financiamento (
    id_dados_financiamento SERIAL PRIMARY KEY,
    prazo VARCHAR,
    faixa_score VARCHAR
);

-- Tabela Fato contratos_fato
CREATE TABLE contratos_fato (
    id_contrato SERIAL PRIMARY KEY,
    id_inadimplencia INTEGER,
    id_vendedor_loja INTEGER,
    id_dados_financiamento INTEGER,
    id_tempo INTEGER,
    id_cliente INTEGER,
    id_veiculo INTEGER,
    vlr_financiado NUMERIC(30,2),
    vlr_entrada NUMERIC(30,2),
    vlr_juros NUMERIC(30,2),
    vlr_operacao NUMERIC(30,2),
    qtd_veiculos INTEGER
);

-- Criação das chaves estrangeiras na tabela Fato contratos_fato

```

```

ALTER TABLE contratos_fato
  ADD FOREIGN KEY (id_inadimplencia)
  REFERENCES inadimplencia (id_inadimplencia);

ALTER TABLE contratos_fato
  ADD FOREIGN KEY (id_vendedor_loja)
  REFERENCES vendedor_loja (id_vendedor_loja);

ALTER TABLE contratos_fato
  ADD FOREIGN KEY (id_dados_financiamento)
  REFERENCES dados_financiamento (id_dados_financiamento);

ALTER TABLE contratos_fato
  ADD FOREIGN KEY (id_tempo)
  REFERENCES tempo (id_tempo);

ALTER TABLE contratos_fato
  ADD FOREIGN KEY (id_cliente)
  REFERENCES cliente (id_cliente);

ALTER TABLE contratos_fato
  ADD FOREIGN KEY (id_veiculo)
  REFERENCES dados_veiculo (id_veiculo);

```

## b. Hierarquias identificadas

As hierarquias identificadas são:

- **Dimensão vendedor\_loja:** a hierarquia desta dimensão é definida por (all) <= (regiao\_loja) <= (estado\_loja) <= (cidade\_loja) <= (nm\_loja) <= (nm\_vendedor).
- **Dimensão cliente:** esta dimensão tem três hierarquias diferentes. A primeira é definida por (all) <= (uf\_cliente) <= (estado\_cliente) <= (cidade\_cliente). A segunda é definida por (all) <= (ocupacao) e a terceira é definida por (all) <= (faixa\_idade).
- **Dimensão tempo:** a hierarquia desta dimensão é definida por (all) <= (ano) <= (mes).
- **Dimensão inadimplência:** esta dimensão tem quatro hierarquias diferentes. A primeira é definida por (all) <= (fg\_pg\_em\_dia), a segunda por (all) <= (fg\_atraso\_30\_dias), a terceira por (all) <= (fg\_atraso\_60\_dias) e a quarta por (all) <= (fg\_atraso\_90\_dias).
- **Dimensão dados\_financiamento:** esta dimensão tem duas hierarquias. A primeira é definida por (all) <= (faixa\_score). A segunda é definida por (all) <= (prazo).

- **Dimensão dados\_veiculo:** a hierarquia desta dimensão é definida por (all) <= (tipo\_veiculo) <= (marca\_veiculo) <= (modelo\_veiculo) <= (vetor\_caracteristica)

### c. Medidas identificadas

As medidas identificadas são:

- **Vlr\_financiado:** esta medida representa o valor financiado de cada contrato de financiamento. Em resumo, ela mostra quanto o cliente financiou na operação. Esta medida pode ser agregada utilizando as funções avg e sum.

- **Vlr\_entrada:** esta medida representa o valor de entrada de cada contrato de financiamento. Em resumo, ela mostra quanto o cliente pagou de entrada na operação. Esta medida pode ser agregada utilizando as funções avg e sum.

- **Vlr\_juros:** esta medida representa o valor de juros de cada contrato de financiamento. Em resumo, ela mostra quanto o cliente teve que pagar de juros na operação. Cada cliente paga o valor de juros conforme o risco dele ficar inadimplente. Esta medida pode ser agregada utilizando as funções avg e sum.

- **Vlr\_operação:** esta medida representa o valor total da operação de cada contrato de financiamento. Ela é a soma do valor financiado com o valor dos juros. Esta medida pode ser agregada utilizando as funções avg e sum.

- **Qtd\_veiculos:** esta medida representa a quantidade de veículos que o cliente comprou em cada operação de financiamento. Cada cliente pode comprar mais de um carro por operação. Esta medida pode ser agregada utilizando as funções avg e sum.

### d. 4 grupos de consultas analíticas no datawarehouse e suas características

- **Grupo de consultas 1:** Este grupo de consulta tem como objetivo verificar a soma total de valor financiado por determinadas lojas de diversas cidades durante os anos de 2020 e de 2021.

Consulta 1.1 – Verificar a soma do valor financiado das lojas da cidade de São Paulo no mês de Outubro de 2020.

```
SELECT v.nm_loja, sum(vlr_financiado) AS sum_vlr_financiado
FROM vendedor_loja as v, contratos_fato as fato, tempo
WHERE v.id_vendedor_loja = fato.id_vendedor_loja AND fato.id_tempo = tempo.id_tempo
AND v.cidade_loja = 'São Paulo' AND tempo.mes = 'Outubro' AND tempo.ano = 2020
GROUP BY nm_loja;
```

	nm_loja character varying	sum_vlr_financiado numeric
1	Armando Veículos	486389.70
2	Auto Shopping Cristal	732899.00
3	AutoStar	1212950.20
4	Caraigá	416665.60
5	Mercadocar	572545.00
6	Viamar	978565.80

Consulta 1.2 – Verificar a soma do valor financiado das lojas Simcauto, Eurobarra e Robmar Automóveis (ambas da cidade do Rio de Janeiro) nos meses de Janeiro, Fevereiro e Março de 2021.

```
SELECT v.nm_loja, v.cidade_loja, sum(vlr_financiado) as sum_vlr_financiado
FROM vendedor_loja as v, contratos_fato as fato, tempo
WHERE v.id_vendedor_loja = fato.id_vendedor_loja AND fato.id_tempo = tempo.id_tempo
AND v.cidade_loja = 'Rio de Janeiro' AND (tempo.mes = 'Janeiro' OR tempo.mes = 'Fevereiro'
OR tempo.mes = 'Março')
AND tempo.ano = 2021 AND (v.nm_loja = 'Simcauto' OR v.nm_loja = 'Eurobarra' OR v.nm_loja
= 'Robmar Automóveis')
GROUP BY v.nm_loja, v.cidade_loja;
```

	nm_loja character varying	cidade_loja character varying	sum_vlr_financiado numeric
1	Eurobarra	Rio de Janeiro	6291400.50
2	Robmar Automóveis	Rio de Janeiro	5709624.60
3	Simcauto	Rio de Janeiro	3561316.50

Consulta 1.3 – Verificar a soma do valor financiado por vendedor e por loja no estado do Paraná no mês de Outubro de 2020 com o objetivo de ver qual o vendedor que teve o maior montante financiado.

```
SELECT v.nm_vendedor, v.nm_loja, v.estado_loja, sum(vlr_financiado) as sum_vlr_financiado
FROM vendedor_loja as v, contratos_fato as fato, tempo
WHERE v.id_vendedor_loja = fato.id_vendedor_loja AND fato.id_tempo = tempo.id_tempo
AND v.estado_loja = 'Paraná' AND tempo.mes = 'Outubro' AND tempo.ano = 2020
GROUP BY v.nm_vendedor, v.nm_loja, v.estado_loja
ORDER BY sum_vlr_financiado DESC;
```



	nm_vendedor character varying	nm_loja character varying	estado_loja character varying	sum_vlr_financiado numeric
1	Vitor Gabriel Ribeiro	Barigui Seminovos	Paraná	688887.00
2	Vitória Novaes	CCV	Paraná	618570.00
3	Pedro Moraes	Metrosul	Paraná	501437.50
4	Luiz Henrique Rodrigues	Barigui Seminovos	Paraná	363002.50
5	Rodrigo da Paz	Barigui Seminovos	Paraná	294236.00
6	Ana Luiza Melo	7place	Paraná	188091.00
7	Yago Barbosa	Barigui Seminovos	Paraná	158944.00
8	Rafael Farias	Metrosul	Paraná	151590.00
9	Pedro Caldeira	Valesul	Paraná	141495.00
10	Lucas Gabriel Moraes	Metrosul	Paraná	135992.00
11	Júlia Fernandes	Valesul	Paraná	111580.00

- **Grupo de consultas 2:** Este grupo de consulta tem como objetivo verificar a quantidade de veículos vendidos de determinadas fabricantes e suas características, observando a região de venda e informações do cliente.

Consulta 2.1 - Verificar quantos veiculos financiados os clientes com ocupação Profissional Liberal e que residem no Estado do Mato Grosso do Sul tiveram no período completo da base.

```
SELECT cliente.ocupacao, sum(fato.qtd_veiculos) as sum_qtd_veiculos
FROM vendedor_loja as v, contratos_fato as fato, cliente
WHERE v.id_vendedor_loja = fato.id_vendedor_loja AND fato.id_cliente = cliente.id_cliente
AND cliente.estado_cliente = 'Mato Grosso do Sul' AND cliente.ocupacao = 'Profissional Liberal'
GROUP BY cliente.ocupacao;
```

	ocupacao character varying	sum_qtd_veiculos bigint
1	Profissional Liberal	31

Consulta 2.2 - Verificar quantos veiculos financiados os clientes que residem no estado de São Paulo e que tem idade acima de 45 anos tiveram no período completo da base.

```
SELECT cliente.faixa_idade, cliente.estado_cliente, sum(fato.qtd_veiculos) as
sum_qtd_veiculos
FROM vendedor_loja as v, contratos_fato as fato, cliente
WHERE v.id_vendedor_loja = fato.id_vendedor_loja AND fato.id_cliente = cliente.id_cliente
AND cliente.estado_cliente = 'São Paulo'
AND cliente.faixa_idade IN ('06. 46 a 50 anos', '07. 51 a 55 anos', '08. 56 a 60 anos',
'09. 61 a 65 anos', '10. Acima de 65 anos')
```

```
GROUP BY cliente.faixa_idade, cliente.estado_cliente
ORDER BY cliente.faixa_idade;
```

	faixa_idade character varying	estado_cliente character varying	sum_qtd_veiculos bigint
1	06. 46 a 50 anos	São Paulo	34
2	07. 51 a 55 anos	São Paulo	44
3	08. 56 a 60 anos	São Paulo	24
4	09. 61 a 65 anos	São Paulo	50
5	10. Acima de 65 anos	São Paulo	49

Consulta 2.3 - Verificar quantos veiculos financiados as lojas da região do Sudeste fizeram no período completo da base para os clientes com o seguinte perfil: ocupação Funcionário Público ou Militar com idade entre 18 e 30 anos.

```
SELECT cliente.ocupacao, v.regiao_loja, sum(fato.qtd_veiculos) as sum_qtd_veiculos
FROM vendedor_loja as v, contratos_fato as fato, cliente
WHERE v.id_vendedor_loja = fato.id_vendedor_loja AND fato.id_cliente = cliente.id_cliente
AND v.regiao_loja = 'Sudeste'
AND cliente.faixa_idade IN ('01. 18 a 24 anos', '02. 26 a 30 anos')
AND cliente.ocupacao IN ('Funcionário Público', 'Militar')
GROUP BY cliente.ocupacao, v.regiao_loja
ORDER BY v.regiao_loja, cliente.ocupacao;
```

	ocupacao character varying	regiao_loja character varying	sum_qtd_veiculos bigint
1	Funcionário Público	Sudeste	22
2	Militar	Sudeste	18

- **Grupo de consultas 3:** Este grupo de consulta tem como objetivo verificar o perfil de inadimplência dos clientes que atrasaram alguma parcela do financiamento em mais de 30 dias, observando diversas informações que ajudem a entender este público.

Consulta 3.1 – Verificar quantos clientes com a ocupação Aposentado das lojas da cidade de São Paulo tiveram atraso de mais de 30 dias no pagamento de alguma parcela do contrato.

```
SELECT v.cidade_loja, d.prazo, sum(i.fg_atraso_30dias) as sum_qtd_atraso_30d
```

```

FROM contratos_fato as fato, vendedor_loja as v, cliente, inadimplencia as i,
dados_financiamento as d
WHERE fato.id_vendedor_loja = v.id_vendedor_loja AND fato.id_cliente = cliente.id_cliente
AND fato.id_inadimplencia = i.id_inadimplencia
AND fato.id_dados_financiamento = d.id_dados_financiamento AND i.fg_atraso_30dias = 1 AND
cliente.ocupacao = 'Aposentado' AND v.cidade_loja = 'São Paulo'
GROUP BY v.cidade_loja, d.prazo
ORDER BY d.prazo;

```

	cidade_loja character varying	prazo character varying	sum_qtd_atraso_30d bigint
1	São Paulo	12	1

Consulta 3.2 – Verificar quantos clientes que residem no estado do Rio de Janeiro, com idade menor que 65 anos, atrasaram mais de 30 dias no pagamento de alguma parcela do contrato. Mostrar esta quantidade agrupada por Faixa de Score.

```

SELECT cliente.estado_cliente, df.faixa_score, sum(i.fg_atraso_30dias) as
sum_qtd_atraso_30d
FROM contratos_fato as fato, cliente, inadimplencia as i, dados_financiamento as df
WHERE cliente.id_cliente = fato.id_cliente AND i.id_inadimplencia = fato.id_inadimplencia
AND df.id_dados_financiamento = fato.id_dados_financiamento
AND i.fg_atraso_30dias = 1 AND cliente.estado_cliente = 'Rio de Janeiro' AND
cliente.faixa_idade NOT IN ('10. Acima de 65 anos')
GROUP BY cliente.estado_cliente, df.faixa_score;

```

	estado_cliente character varying	faixa_score character varying	sum_qtd_atraso_30d bigint
1	Rio de Janeiro	01_EXCELENTE	4
2	Rio de Janeiro	02_BOM	1
3	Rio de Janeiro	03_RUIM	6
4	Rio de Janeiro	04_PESSIMO	4

- **Grupo de consultas 4:** Este grupo de consulta tem como objetivo verificar a soma do montante do valor de juros cobrados de diversas regiões, marcas de veículos diferentes, períodos e dados do financiamento distintos.

Consulta 4.1 – Verificar o valor total de juros cobrado nos clientes que financiaram um ou mais veículos da marca Audi nos meses de Setembro, Outubro e Novembro de 2020. Mostrar esta soma agrupada pelo prazo do financiamento.

```

SELECT dv.marca_veiculo, df.prazo, sum(fato.vlr_juros) as sum_vlr_juros
FROM contratos_fato as fato, dados_veiculo as dv, tempo, dados_financiamento as df
WHERE fato.id_veiculo = dv.id_veiculo AND fato.id_tempo = tempo.id_tempo AND
fato.id_dados_financiamento = df.id_dados_financiamento
AND dv.marca_veiculo = 'Audi' AND tempo.mes in ('Setembro', 'Outubro', 'Novembro') AND
tempo.ano = 2020
GROUP BY dv.marca_veiculo, df.prazo;

```

	marca_veiculo character varying	prazo character varying	sum_vlr_juros numeric
1	Audi	12	90523.24
2	Audi	24	193976.15
3	Audi	36	249969.56
4	Audi	48	74425.49
5	Audi	60	44433.51

Consulta 4.2 - Verificar o valor total de juros cobrado nos clientes que financiaram um ou mais veículos do tipo SUV no ano de 2021. Mostrar esta soma agrupada pela faixa de score.

```

SELECT dv.tipo_veiculo, df.faixa_score, sum(fato.vlr_juros) as sum_vlr_juros
FROM contratos_fato as fato, dados_veiculo as dv, tempo, dados_financiamento as df
WHERE fato.id_veiculo = dv.id_veiculo AND fato.id_tempo = tempo.id_tempo AND
fato.id_dados_financiamento = df.id_dados_financiamento
AND dv.tipo_veiculo = 'SUV' AND tempo.ano = 2021
GROUP BY dv.tipo_veiculo, df.faixa_score;

```

	tipo_veiculo character varying	faixa_score character varying	sum_vlr_juros numeric
1	SUV	01_EXCELENTE	1276574.71
2	SUV	02_BOM	1846632.69
3	SUV	03_RUIM	260925.89
4	SUV	04_PESSIMO	331926.40

## 2. Recuperação de Informação Baseada em Conteúdo

### a. Descrever como armazenar as imagens (ou outros dados complexos) no data warehouse projetado e implementado

Os dados complexos utilizados para a elaboração deste Projeto Integrador foram tirados do dataset The Comprehensive Cars (CompCars). Este dataset contém 5 atributos extraídos de 164.344 imagens de veículos do carro todo, 27.618 imagens das peças do veículo e 50.000 imagens de câmeras de segurança com a frente do veículo. Os atributos do vetor de característica são: velocidade máxima do veículo, cilindradas, quantidade de portas, quantidade de assentos e o tipo do veículo. O dataset está disponível em: [http://mmlab.ie.cuhk.edu.hk/datasets/comp\\_cars/](http://mmlab.ie.cuhk.edu.hk/datasets/comp_cars/). Para este data warehouse, foi feita a seleção de 106 veículos com base nos modelos que tem comercialização no Brasil e que estavam disponíveis em sua versão 0KM no mercado nacional no ano de 2020.

Como este dataset com as informações das imagens dos veículos está diretamente ligado a tabela Fato (contratos de financiamentos de veículos), foi criada uma coluna que armazena os vetores de características da imagem de cada carro selecionado dentro da dimensão dados\_veiculo, juntos com outras informações úteis para as análises do negócio.

### b. Armazenar os vetores de características no data warehouse mantido no PostgreSQL.

Os vetores de características foram armazenados na dimensão dados\_veiculos. Os dados desta dimensão estão armazenados no arquivo tabela\_dados\_veiculos.csv. A imputação dos dados no SGBD foi feita com ajuda de um looping no Python. O código para execução do armazenamento dos dados está descrito no PDF do Jupyter Notebook Script\_Criacao\_Banco\_Dados que está dentro da pasta do Projeto Integrador.

### c. Criar consultas por similaridade range query e kNN (aceita-se o valor fixo para k, ex: os 5 mais similares) sobre o data warehouse que mantém os vetores de características

Para executar as consultas por similaridade range query e kNN no data warehouse, é necessário criar a função de distância e a criação das funções para a execução dos modelos. A função de distância utilizada será a L2, que foi vista em aula durante o módulo. Os códigos necessários para a realização desta etapa estão descritos abaixo:

```
-- Criação da função de distância L2
CREATE OR REPLACE FUNCTION l2(elem1 FLOAT[], elem2 FLOAT[]) RETURNS FLOAT AS $$
DECLARE
    size INTEGER;
    somat FLOAT;
```

```

BEGIN
SELECT CARDINALITY(vetor_caracteristica) INTO size FROM dados_veiculo LIMIT 1;
somat := 0;
FOR i IN 1..SIZE LOOP
    somat := somat + (ABS(elem1[i] - elem2[i])*ABS(elem1[i] - elem2[i]));
END LOOP;
RETURN SQRT(somat);
END $$
LANGUAGE plpgsql;

-- Criação do RangeQuery utilizando a função de distância L2
CREATE OR REPLACE FUNCTION RangeQueryl2(qc FLOAT[], radius FLOAT)
RETURNS TABLE (id_veiculo INTEGER, distancia FLOAT, marca_veiculo VARCHAR, modelo_veiculo
VARCHAR) AS $$
BEGIN
RETURN QUERY
    SELECT dados_veiculo.id_veiculo, l2(vetor_caracteristica, qc) AS distancia,
dados_veiculo.marca_veiculo, dados_veiculo.modelo_veiculo
FROM dados_veiculo
WHERE l2(vetor_caracteristica, qc) <= radius
ORDER BY l2(vetor_caracteristica, qc);

END $$
LANGUAGE plpgsql;

-- Criação do KNN utilizando a função de distância L2
CREATE OR REPLACE FUNCTION KNN_l2(qc FLOAT[], k INTEGER)
RETURNS TABLE (id_veiculo INTEGER, distancia FLOAT, marca_veiculo VARCHAR, modelo_veiculo
VARCHAR) AS $$
BEGIN
RETURN QUERY
    SELECT dados_veiculo.id_veiculo, l2(vetor_caracteristica, qc) AS distancia,
dados_veiculo.marca_veiculo, dados_veiculo.modelo_veiculo
FROM dados_veiculo
ORDER BY l2(vetor_caracteristica, qc) LIMIT k;

END $$
LANGUAGE plpgsql;





```

- **Consulta 1:** Quais são os veículos mais similares com raio 3 ao Volkswagen Polo Hatch:

O vetor de características do Polo Hatch é: [182,1.4,5,5,4]. O código para executar a consulta é:

```
SELECT * FROM RangeQuery12(ARRAY[182,1.4,5,5,4], 3);
```





Temos como resultado: Volkswagen Jetta, Toyota Prius, Honda CR-V, Fiat 500, Chevrolet Cruze Hatch e o Citroen C4 Aircross.

	 id_veiculo integer	 distancia double precision	 marca_veiculo character varying	 modelo_veiculo character varying
1	57	0	Volkswagen	Polo hatch
2	63	1.7320508075688772	Volkswagen	Jetta
3	95	2.039607805437114	Toyota	Prius
4	48	2.08806130178211	Honda	Honda CR-V
5	69	2.23606797749979	Fiat	FIAT 500
6	105	2.835489375751565	Chevrolet	Cruze hatch
7	92	2.891366458960192	Citroen	C4 Aircross

- **Consulta 2:** Quais são os 3 vizinhos mais similares ao Audi A3, não incluindo ele mesmo:

O vetor de características do Audi A3 é: [235,1.8,5,5,4]. O código para executar a consulta é:

```
SELECT * FROM KNN_12(ARRAY[235,1.8,5,5,4], 4);
```





	 id_veiculo integer	 distancia double precision	 marca_veiculo character varying	 modelo_veiculo character varying
1	1	0	Audi	Audi A3
2	27	1.7435595774162693	BMW	BMW 4 Series
3	12	2.118962010041709	Audi	Audi A7
4	78	2.4576411454889016	Mini	MINI JCW

Temos como resultado: BMW 4 Series, Audi A7 e o Mini JCW.

- **Consulta 3:** Considerando um raio 7, qual é o veículo menos similar ao Mitsubishi Pajero:

O vetor de características do Mitsubishi Pajero é: [170,3,5,5,2]. O código para executar a consulta é:

```
SELECT * FROM RangeQuery12(ARRAY[170,3,5,5,2], 7);
```

	 id_veiculo integer	 distancia double precision	 marca_veiculo character varying	 modelo_veiculo character varying
1	101	0	Mitsubishi	Pajero
2	88	1.4142135623730951	Kia	Sportage
3	74	2.4413111231467406	Mini	MINI COUNTRYMAN
4	89	2.4413111231467406	Kia	Soul
5	97	3.3000000000000003	Toyota	Yaris

Temos como resultado: Toyota Yaris.

#### d. Anexo: Vetores de Característica

```
id_veiculo  vetor_caracteristica
```

```
1  {235, 1.8, 5, 5, 4}
2  {220, 1.8, 4, 5, 3}
3  {228, 2.0, 4, 5, 3}
4  {230, 2.0, 5, 5, 2}
5  {234, 3.0, 5, 5, 2}
6  {242, 1.8, 4, 5, 3}
7  {250, 4.2, 2, 4, 3}
8  {250, 4.0, 5, 4, 6}
9  {238, 2.0, 5, 5, 7}
10 {248, 2.0, 2, 4, 3}
11 {242, 2.0, 5, 4, 6}
12 {235, 2.5, 5, 5, 6}
13 {222, 3.0, 5, 5, 2}
14 {300, 4.2, 2, 2, 10}
15 {210, 1.6, 4, 5, 3}
16 {200, 2.0, 5, 5, 2}
17 {225, 2.0, 4, 5, 3}
18 {250, 3.0, 2, 4, 3}
19 {250, 3.0, 4, 5, 3}
20 {250, 3.0, 2, 4, 3}
21 {250, 4.4, 4, 5, 3}
22 {250, 4.4, 5, 5, 2}
23 {250, 4.4, 5, 4, 2}
24 {213, 2.0, 2, 4, 3}
25 {230, 2.0, 2, 4, 3}
26 {229, 2.0, 5, 5, 6}
27 {236, 2.0, 4, 5, 3}
28 {240, 2.0, 5, 5, 6}
29 {250, 3.0, 4, 5, 3}
30 {210, 2.0, 5, 5, 2}
```



31 {212, 2.0, 5, 5, 2}  
32 {230, 3.0, 5, 5, 2}  
33 {240, 3.0, 5, 4, 2}  
34 {232, 2.0, 2, 2, 9}  
35 {226, 1.8, 4, 5, 3}  
36 {240, 2.0, 2, 4, 10}  
37 {250, 5.5, 4, 4, 3}  
38 {250, 6.2, 4, 5, 3}  
39 {210, 5.5, 5, 5, 2}  
40 {250, 5.5, 5, 5, 2}  
41 {250, 2.0, 4, 5, 3}  
42 {130, 2.1, 3, 3, 5}  
43 {202, 1.6, 5, 5, 4}  
44 {240, 2.0, 4, 5, 3}  
45 {240, 2.0, 2, 4, 3}  
46 {250, 3.0, 4, 5, 3}  
47 {199, 1.8, 4, 5, 3}  
48 {182, 2.0, 5, 5, 2}  
49 {202, 1.6, 5, 5, 2}  
50 {262, 2.7, 2, 2, 12}  
51 {223, 2.0, 5, 5, 2}  
52 {262, 3.0, 5, 4, 10}  
53 {283, 3.4, 2, 4, 10}  
54 {239, 3.0, 5, 5, 2}  
55 {242, 3.0, 5, 5, 2}  
56 {185, 1.6, 5, 5, 11}  
57 {182, 1.4, 5, 5, 4}  
58 {200, 1.4, 4, 5, 3}  
59 {185, 1.4, 5, 5, 2}  
60 {200, 1.4, 5, 5, 4}  
61 {230, 3.0, 5, 5, 2}  
62 {204, 1.4, 5, 5, 4}  
63 {181, 1.4, 4, 5, 3}  
64 {200, 5.7, 4, 5, 8}  
65 {195, 1.6, 5, 5, 4}  
66 {195, 1.6, 4, 5, 3}  
67 {195, 1.0, 5, 5, 4}  
68 {260, 3.7, 2, 4, 10}  
69 {182, 1.4, 3, 4, 4}  
70 {260, 3.0, 2, 2, 10}  
71 {241, 2.0, 4, 5, 3}  
72 {209, 2.0, 4, 5, 3}

73 {179, 1.6, 3, 4, 4}  
74 {168, 1.6, 5, 5, 2}  
75 {198, 1.6, 3, 2, 10}  
76 {192, 1.6, 2, 2, 12}  
77 {191, 1.6, 2, 4, 12}  
78 {236, 1.6, 3, 4, 4}  
79 {223, 1.6, 5, 5, 2}  
80 {238, 1.6, 3, 4, 4}  
81 {315, 3.8, 2, 4, 10}  
82 {190, 2.0, 5, 5, 2}  
83 {195, 3.0, 5, 5, 2}  
84 {210, 3.0, 5, 5, 2}  
85 {217, 2.0, 5, 5, 2}  
86 {210, 3.0, 5, 5, 2}  
87 {186, 1.6, 4, 5, 3}  
88 {171, 2.0, 5, 5, 2}  
89 {170, 1.6, 5, 5, 4}  
90 {190, 2.2, 5, 5, 2}  
91 {255, 2.5, 4, 5, 3}  
92 {184, 2.0, 5, 5, 2}  
93 {202, 1.6, 3, 5, 4}  
94 {185, 2.0, 5, 5, 2}  
95 {180, 1.8, 5, 5, 4}  
96 {190, 2.0, 4, 5, 3}  
97 {168, 1.3, 5, 5, 4}  
98 {230, 2.0, 4, 5, 3}  
99 {230, 2.0, 5, 5, 7}  
100 {210, 2.0, 5, 5, 2}  
101 {170, 3.0, 5, 5, 2}  
102 {190, 2.0, 5, 5, 2}  
103 {190, 2.0, 5, 5, 2}  
104 {200, 1.4, 4, 5, 3}  
105 {180, 1.6, 5, 5, 6}  
106 {250, 3.6, 2, 4, 10}

### 3. Introdução ao Big Data

#### a. Qual é o NoSQL escolhido e por que da sua escolha?

O NoSQL escolhido é o MongoDB, pois ele utiliza um formato de dados semi-estruturado JSON, que consiste em um formato moderno, de fácil desserialização e de fácil utilização. A estrutura de armazenamento e processamento de dados do banco orientado a documentos é a que se adequa melhor ao nosso database, uma vez que estamos lidando com dados de contratos de financiamento.

#### b. Qual é o modelo de dados do NoSQL escolhido?

O modelo de dados do NoSQL escolhido é o orientado a documentos.

#### c. Por que seria interessante migrar do PostgreSQL ao NoSQL escolhido? Detalhar a motivação.

Seria interessante a migração na situação em que o volume dos dados fosse muito alto e no qual fosse necessário ter alta disponibilidade e desempenho na escrita e leitura dos dados. Seria interessante também pois o banco de dados orientado a documentos tem maior flexibilidade nos dados armazenados, de forma que cada contrato de financiamento da base de dados poderia ter mais informações do que as descritas no nosso modelo, de forma que cada objeto dessa ou de outras tabelas pudesse ter atributos diferentes, além dos que já pertencem a nossa base.

#### d. Como implantar as tabelas de dimensão e de fato no NoSQL escolhido?

Para implementar as tabelas dimensão e de fato, é necessário criar um database chamado "dw\_financiamento", e criar cada coleção e documentos dentro destas, conforme abaixo:

```
#Conectando o cliente ao localhost
client = pymongo.MongoClient("mongodb://localhost:27017/")

#Criação do database
db = client["dw_financiamento"]

#Criação das coleções
db.create_collection('contratos_fato')
db.create_collection('vendedor_loja')
db.create_collection('cliente')
db.create_collection('inadimplencia')
db.create_collection('dados_financiamento')
db.create_collection('tempo')
db.create_collection('dados_veiculo')

#Criação de um documento de exemplo em cada coleção
```

```
db.vendedor_loja.insert_one({
    'id_vendedor_loja': 21,
    'nm_vendedor': "Emanuely da Luz",
    'nm_loja': "Viamar",
    'cidade_loja': "São Paulo",
    'estado_loja': "São Paulo",
    'uf_loja': "SP",
    'regiao_loja': "Sudeste",
})

db.dados_veiculo.insert_one({
    'id_veiculo': 42,
    'velocidade_max': 130,
    'cilindradas': 2.1,
    'qtd_portas': 3,
    'qtd_assentos': 3,
    'modelo_veiculo': "Sprinter",
    'marca_veiculo': "Mercedes-Benz",
    'tipo_veiculo': "utilitário",
    'tipo_veiculo': 178000,
    'vetor_caracteristica': [130, 2.1, 3, 3, 5],
    'Id_tipo': 5
})

db.cliente.insert_one({
    'id_cliente': 347,
    'cidade_cliente': "São Paulo",
    'estado_cliente': "São Paulo",
    'uf_cliente': "SP",
    'ocupacao': "Aposentado",
    'faixa_idade': "10. Acima de 65 anos"
})

db.inadimplencia.insert_one({
    'id_inadimplencia': 4,
    'fg_pg_em_dia': 1,
    'fg_atraso_30dias': 0,
    'fg_atraso_60dias': 0,
    'fg_atraso_90dias': 0
})

db.dados_financiamento.insert_one({
```

```

        'id_dados_financiamento': 9,
        'prazo': 48,
        'faixa_score': "02_BOM"
    })

db.tempo.insert_one({
    'id_tempo': 12,
    'mes': 'Julho',
    'ano': 2021
})

db.contratos_fato.insert_one({
    'id_contrato': 1,
    'id_inadimplencia': 4,
    'id_cliente': 347,
    'id_veiculo': 42,
    'id_vendedor_loja': 21,
    'id_dados_financiamento': 9,
    'Id_tempo': 12,
    'vlr_financiado': 320400,
    'vlr_entrada': 35600,
    'vlr_juros': 16020,
    'vlr_operacao': 336420,
    'qtd_veiculos': 2
})

```

**e. Qual seria a sintaxe das consultas analíticas no NoSQL escolhido? Apresente exemplos.**

- Retornar todos os contratos de financiamento no qual o carro financiado o id\_veiculo = 42 (modelo\_veiculo = "Sprinter"):

```
list(db.contratos_fato.find({'id_veiculo': 42}))
```

- Retornar os contratos de financiamento no qual a data inicial foi em 2021, ou id\_tempo igual ou menor a 5:

```
list(db.contratos_fato.find({'id_tempo': {'$lte': 5}}))
```

- Retornar os contratos de financiamento no qual o valor dos juros é maior que R\$5.000:

```
list(db.contratos_fato.find({'vlr_juros': {'$gt': 5000}}))
```

- Retornar os contratos de financiamento no qual o valor financiado é igual a R\$116.991:

```
list(db.contratos_fato.find({'vlr_financiado': {'$eq': 116991}}))
```

- Retornar os contratos de financiamento no qual a quantidade de veículos é diferente de 1:

```
list(db.contratos_fato.find({'qtd_veiculos': {'$ne': 1}}))
```

- Retornar os contratos de financiamento no qual o valor dos juros seja igual a R\$5.849,55 ou R\$2.478,96:

```
list(db.contratos_fato.find({'vlr_juros': {'$in': [5849.55, 2478.96]}}))
```

- Retornar os contratos de financiamento no qual o valor da operação é maior ou igual a R\$300 mil:

```
list(db.contratos_fato.find({'vlr_operacao': {'$gte': 300000}}))
```

- Retornar os contratos de financiamento no qual o valor financiado é menor que R\$15 mil:

```
list(db.contratos_fato.find({'vlr_financiado': {'$lt': 15000}}))
```

- Retornar os contratos de financiamento no qual o valor de entrada é diferente de R\$35.600, R\$360.000 e R\$523.000:

```
list(db.contratos_fato.find({'vlr_entrada': {'$nin': [35600, 360000 e 523000]}}))
```