



Digital Transformation
Z Academy

DATA SCIENCE & BUSINESS INTELLIGENCE

PROF. ANTÔNIO FERREIRA DOS SANTOS JÚNIOR
PROF. MARCELO CHAMY MACHADO



LG

inova
POLO DE INOVAÇÃO IFAM

 INSTITUTO FEDERAL
Amazonas
Campus Manaus Zona Leste

 **FAEPI**

SUMÁRIO

APRESENTAÇÃO	3
Módulo 1: Fundamentos de Dados	6
Módulo 2: Técnicas de Data Science	16
NUMPY	17
Objetos array multidimensionais	19
Atributos	22
Índices	22
Fatiamento (slicing)	24
Redimensionamento (reshaping)	25
Junção e separação de arrays	26
PANDAS	27
Conceitos básicos	27
Método isin()	34
Métodos loc e iloc	34
Análise de frequência	39
Exercícios:	40
Realizando merge em dataframes	41
Leitura de dados em páginas web:	42
Técnicas de limpeza e preparação de dados	47
Módulo 3: Business Intelligence	49
Conceitos básicos de Business Intelligence	50
Conceitos-chave de Business Intelligence (BI)	50
Dados estruturados, não estruturados e semiestruturados	52
Visualizações	57
O que é o Power BI ?	57
HANDS-ON - Exploração da interface gráfica da ferramenta Power BI	61
Fase 02 – Modelagem e visualização dos modelos e dados associados:	62
Fase 03: Visualização de relatórios e dashboards:	63
Fase 04: Publicação / Compartilhamento de relatórios e dashboards	63
Estrutura geral da interface de visualização do Power BI:	64
Exercícios práticos de fixação – Descoberta, análise, transformações e visualizações de dados, a partir de dois conjuntos de dados não conhecidos previamente:	88
Gerando informações visuais com o seaborn	89
Correlação	91
Módulo 4: Comunicação e Storytelling com Dados (10h/aula)	94
RESUMO MÓDULO	100
ATIVIDADES	101
REFERÊNCIAS BIBLIOGRÁFICAS	102



APRESENTAÇÃO

Car@ aprendiz,

Seja bem-vind@ ao curso **Data Science & Business Intelligence** do ZL Academy – Digital Transformation.

Este curso surgiu da necessidade de desenvolver, com os alunos e os parceiros do ZL Academy, conhecimentos e competências relacionadas à geração de informações valiosas, a partir de insights importantes para os negócios das mais diversas áreas e que auxiliem na tomada de decisões estratégicas nas organizações.

Seu objetivo é orientar e apoiar os profissionais envolvidos nas áreas de Data Science & Business Intelligence. Para tanto, serão desenvolvidas/aprimoradas as seguintes competências referentes à ciência de dados:

- Obtenção de dados
- Organização

- Exploração
- Apresentação

Para alcançar essas competências, conhiceremos os seguintes tópicos:

- **Módulo 1: Fundamentos de Dados**
 - Conceitos de Big Data, Data Warehouse e Data Lake.
 - Processo de *Knowledge Discovery in Databases* (KDD).
 - Modelagem e qualidade de dados.
 - Gerenciamento de Banco de Dados.
- **Módulo 2: Técnicas de Data Science**
 - Introdução à programação para Data Science (Python libraries: NumPy, Pandas).
 - Técnicas de Limpeza e Preparação de Dados.
 - Análise Exploratória de Dados (EDA).
 - Visualização de Dados avançada (técnicas interativas e storytelling).
 - Estatística Descritiva e Inferencial para Data Science (com foco em testes de hipóteses e análise de variância).
- **Módulo 3: Business Intelligence**
 - Conceitos e fundamentos de Business Intelligence (BI).
 - Sistemas e ferramentas de BI (Dashboards, OLAP).
 - Processo de Business Intelligence: coleta, transformação e análise de dados para geração de relatórios e painéis gerenciais.
 - Mineração de Dados (Data Mining) introdutória (técnicas de agregação e associação sem uso de algoritmos de Machine Learning).
 - Técnicas de Business Analytics para suporte à decisão (análises financeiras, de marketing, etc.).
- **Módulo 4: Comunicação e Storytelling com Dados (10h/aula)**
 - Princípios de design para visualização de dados eficazes.
 - Técnicas de narrativa baseada em dados (data storytelling) para comunicar insights a diferentes públicos.
 - Ética e responsabilidade no uso de dados para tomada de decisão.

- **Metodologia:** A disciplina será ministrada de forma teórica e prática, combinando aulas expositivas, atividades em grupo, laboratórios, estudos de caso e projetos.
 - **Aulas Expositivas:** Introdução aos conceitos fundamentais de *Data Science* e *Business Intelligence* sem *Machine Learning*.
 - **Hands On:** aprendizagem a partir da prática ativa. Os alunos participam diretamente de atividades seguindo orientações do instrutor, ou passos indicados na apostila.

- **Atividades em Grupo:** Discussões e resolução de problemas em grupo para estimular a análise **crítica** e o trabalho colaborativo.
 - **Laboratórios:** Aplicação prática de técnicas de Data Science e BI utilizando ferramentas computacionais.
 - **Estudos de Caso:** Análise de cenários reais de aplicação de Data Science e BI para empresas, com foco em soluções baseadas em estatística e análise de dados.
 - **Projetos:** Desenvolvimento de projetos que integrem os conceitos de Data Science e BI para resolver problemas específicos e gerar insights de valor, utilizando técnicas estatísticas e analíticas.
- **Avaliação de Aprendizagem:** A avaliação será contínua e composta por:
- **Participação nas Atividades em Sala de Aula (20%):** Avaliação da assiduidade, colaboração nas atividades em grupo e interesse nas discussões em sala de aula.
 - **Desempenho nos Laboratórios (30%):** Avaliação da habilidade em utilizar ferramentas de Data Science e BI para realizar tarefas práticas de limpeza, exploração e análise de dados.
 - **Qualidade e Apresentação do Projeto Final (50%):** Avaliação da capacidade de aplicar os conceitos de Data Science e BI para solucionar um problema, realizar análises estatísticas, gerar insights e comunicar os resultados de forma efetiva.
- **Recursos Didáticos:**
- **Apostila:** Material didático contendo os conceitos teóricos, exemplos práticos e exercícios.
 - **Slides:** Apresentações em PowerPoint ou similar para auxiliar nas aulas expositivas.
 - **Softwares:** Ambientes de desenvolvimento e ferramentas de Data Science e BI (ex.: Jupyter Notebook, RStudio, softwares de visualização de dados).

Ao final da disciplina, é esperado que os alunos estejam preparados para desafios voltados para data science e Business Intelligence do mundo real, podendo aplicar os conhecimentos adquiridos com confiança, e buscando aprimoramento constante e se aprofundando nas técnicas aqui abordadas, assim como em outras que não fizeram parte do escopo ou novas técnicas que surjam.

MÓDULO 1: FUNDAMENTOS DE DADOS

- Introdução Ciência de Dados
- Conceitos de Big Data, Data Warehouse e Data Lake.
- Processo de *Knowledge Discovery in Databases* (KDD).
- Modelagem e qualidade de dados.
- Gerenciamento de Banco de Dados.

Introdução à Ciência de Dados

Com a popularização da internet, temos visto crescer a disponibilidade de dados numa magnitude jamais vista e a tendência é crescer ainda mais, à medida que mais pessoas produzem novos conteúdos nessa grande rede. Com essa quantidade de dados disponível, **como podemos extrair informações que sejam úteis para alcançarmos nossos objetivos?** O primeiro ponto aqui é diferenciar dados de informações e para isso vamos usar um jardim como exemplo, para isso o leitor deve se imaginar passando por um jardim com flores, plantas, grama, insetos, talvez alguns anões de jardim, pense nesse jardim.

Dados são fatos, ou seja, verdades que fazem parte do mundo real e são representados de alguma forma percebida pelo seu sistema sensorial. No caso do jardim, tudo que você visualiza, sente com o tato, ouve, sente o gosto ou o cheiro são fatos, são verdades sobre o jardim.

Informações são interpretações, opiniões ou conclusões que são extraídas a partir dos dados. Neste caso, as pessoas que passam pelo jardim terão interpretações diferentes. Umas podem passar por ele e “achar bonito” ou até “cheiroso” se uma planta exalar um cheiro agradável, outras o acharão “mau-cuidado”, há as que “sentirão falta de algo” e as que serão indiferentes, pois estão com pressa ou têm outros objetivos, dentre outras interpretações. Quanto ao jardineiro, este chegará a conclusões de que precisa “cortar a grama”, “plantar, podar ou retirar alguma planta”, “retirar as folhas que caíram”. Enfim, as informações que ele interpreta terão o objetivo de tornar o jardim bonito.

No caso da computação, e, consequentemente, da internet, os dados estão disponíveis nas mais diversas formas tais como textos, vídeos, áudios, tabelas, imagens em arquivos, bancos de dados estruturados, bancos de dados não estruturados, páginas web, sistemas web, dentre outras, numa quantidade que para uma pessoa comum é infinita, mas que, de acordo com KALINOWSKI et. al (2023), em 2021 foi estimada em cerca de 79 zettabytes (79 trilhões de gigabytes), em dados gerados de aplicativos, web sites, redes sociais, bancos, e-commerce, sistemas pessoais e de empresas, de alto volume, alta variedade e alta velocidade. Dito isso, volta-se agora à pergunta feita anteriormente. Para respondê-la, faremos um panorama das tecnologias de coleta e de análise de dados conforme KELLEHER e TIERNEY (2018).

Na década de 70, um marco importante é a criação dos bancos de dados relacionais por Edgard F. Codd, determinando como os dados são armazenados, indexados e obtidos de uma base de dados a partir de um meio simples de consulta que culminou na SQL, sigla em inglês para Linguagem Estruturada de Consulta. Esse conceito é usado até os dias de hoje. Na década de 1990, as empresas perceberam que, mesmo tendo acumulado uma quantidade significativa de dados, elas têm dificuldades em analisá-los. Então surgiu o Data Warehouse, onde os dados são obtidos em toda a organização e integrados de forma que se tenha um conjunto de dados mais comprehensível para análise. Já nas duas últimas décadas, foram introduzidos os dispositivos móveis, com integração de dados com outros dispositivos através da Internet, gerando, além da variedade de dados já descrita, os metadados, que descrevem a estrutura e

propriedades dos dados, ou seja, dados sobre dados, trazendo assim uma nova forma de analisá-los, denominada big data, na qual tem-se um volume de dados extremos e variados, que requer alta velocidade de processamento e veracidade, para gerar valor. Isso trouxe o desenvolvimento de outras tecnologias, referindo-se a elas o termo NoSQL, que armazenam dados como objetos e seus atributos, e de frameworks para o processamento de dados, tais como o MapReduce e o Hadoop.

Tradicionalmente, nas áreas de banco de dados e sistemas de informação, os Sistemas de Gerenciamento de Banco de Dados (SGBD) juntamente com aplicações desktop ou web e, mais recentemente, os aplicativos para dispositivos móveis processam dados e são capazes de organizá-los para apresentá-los em forma de informação. Um exemplo é o sistema de avaliação do ENEM, ao fornecer o Cartão de Confirmação, contendo os dados da escola onde o candidato fará a prova, sala, data e hora, bem como outras recomendações. Os dados dos candidatos e das escolas são armazenados no SGBD e o sistema apresenta o cartão de confirmação com as informações necessárias para que façam sua prova.

Com a introdução das ciências de dados - cujo objetivo é usar dados para obter percepção e compreensão - esse conceito foi expandido para a pirâmide DIKW (ver figura 1), onde dados são processados e transformados em informação, que é transformada em conhecimento e, finalmente, em sabedoria, todos usados para a tomada de decisões a partir do processamento de uma quantidade de dados imensas e com várias variáveis envolvidas.

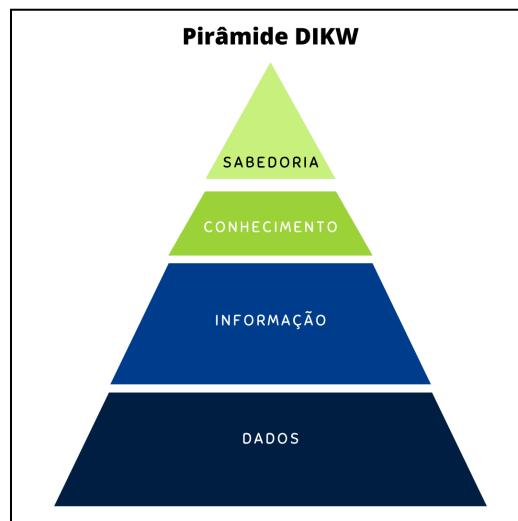


Figura 1. A Pirâmide DIKW (sigla em inglês para dados, informação, conhecimento e sabedoria)

Considerando a pirâmide DIKW de acordo com KELLEHER e TIERNEY (2018), o tamanho de cada item na pirâmide representa a quantidade desse item envolvida. Assim, os dados são criados a partir de abstrações ou medidas trazidas do mundo real, cuja quantidade é a maior; as informações são dados processados, estruturados ou contextualizados de forma a terem algum significado para as pessoas; conhecimento é informação interpretada e com

significado para alguém, de tal forma que este possa agir nela se necessário; sabedoria é a ação sobre o conhecimento de forma apropriada.

Para exemplificar, considerando ainda o ENEM, o INEP (2024) coleta e disponibiliza uma série de dados a respeito deste exame e os torna público. Já vimos que o sistema do ENEM fornece as informações necessárias para que o candidato faça sua prova e, além disso, apresenta a nota obtida e, através de outro sistema, o SISU, possibilita ao candidato submeter sua nota ao processo seletivo de qualquer instituição de ensino superior no Brasil. Com essas informações, já não os dados, pode-se perguntar: qual o perfil do candidato que foi selecionado para um determinado curso de uma universidade? Nesse sentido, será necessário obter informações tanto do SISU quanto do ENEM para buscar tanto informações sobre quem passou nesse curso (pelo SISU), quanto dados sobre quem é esse candidato (pelo ENEM). Com certeza, virão informações sobre a última escola em que o candidato estudou, ou nas que estudou no ensino médio.

E se forem necessárias informações sobre a trajetória do aluno? Então será necessário obter informações a partir do censo escolar, que é outro sistema do MEC, que possui informações sobre o aluno em todas as escolas em que estudou, assim estaremos na parte da pirâmide DIKW que trata do conhecimento, onde, segundo KELLEHER e TIERNEY (2018), são usadas técnicas de agregação e de exploração de dados.

E se for necessário indicar qual a probabilidade de um aluno, considerando sua trajetória escolar, passar num determinado curso no ENEM, sem que ele tenha feito a prova? Nesse sentido, usa-se técnicas de aprendizagem de máquina para que o sistema aprenda sobre as características dos alunos que passaram no ENEM e preveja, com base no conhecimento obtido a partir do censo escolar, aqueles com maior ou menor probabilidade de passar em cada curso do ENEM, levando-se à sabedoria da pirâmide DIKW.

Ciência de Dados - Conceitos

De acordo com ZINOVIEV (2016) Ciência de Dados é a disciplina de extração de conhecimento a partir dos dados, advinda da ciência da computação, da estatística e de conhecimento de domínio. De acordo com KELLEHER e TIERNEY (2018), ela engloba um conjunto de princípios, definição de problemas, algoritmos e processos, advindos campos tais como mineração de dados e aprendizagem de máquina, para extrair padrões não tão óbvios, mas úteis, de um grande conjunto de dados.

Ela se refere à coleta de dados de várias fontes para fins de análise, com o objetivo de apoiar a tomada de decisões, utilizando geralmente grandes quantidades de dados de forma sistemática, sendo a etapa de preparação dos dados brutos, limpeza e análise de dados fundamental para a etapa seguinte, de análise descritiva ou diagnóstica (envolvendo business intelligence) ou análise preditiva (envolvendo algoritmos de machine learning), permitindo, a partir daí uma análise prescritiva (KALINOWSKI et. al, 2023).

ZINOVIEV (2016) apresenta as seguintes áreas tratadas pela ciência de dados para extrair conhecimento a partir de dados:

- **Bancos de dados**, que proveem armazenamento e integração de dados.
- **Análise de texto e processamento de linguagem natural**, que traduz textos qualitativos em variáveis quantitativas.
- **Análise numérica de dados e mineração de dados**, que realiza buscas de padrões consistentes e relacionamentos entre variáveis.
- **Análise de redes complexas**, que trata de coleções de entidades arbitrárias interconectadas.
- **Visualização de dados**, que trata da apresentação das informações através de tabelas e gráficos, que podem mudar de forma dinâmica.
- **Aprendizagem de máquina** (incluindo agrupamento, árvores de decisão, classificação e redes neurais), que trata da realização de previsões baseadas em amostras de dados.
- **Processamento de séries temporais**, que possui uma série de ferramentas para analistas do mercado de ações, economistas e pesquisadores de áudio e vídeo.
- **Big Data**, que se refere à análise de dados não estruturados (texto, áudio e vídeo), com tamanho em terabytes, produzidos e obtidos em pouco tempo

Ciência de Dados - Procedimentos

KELLEHER e TIERNEY (2018) apresentam o ciclo de vida denominado CRISP-DM usado como procedimento para extrair conhecimento a partir de dados, consistindo de 6 estágios: compreensão de negócio, compreensão de dados, preparação de dados, modelagem, avaliação e entrega. São estágios lineares, no entanto, o cientista de dados pode retornar a estágios anteriores.

Para ZINOVIEV (2016), um estudo de análise de dados típico envolve a questão a ser respondida e o tipo de análise a ser empregada, onde a mais simples é a descritiva, na qual o conjunto de dados é descrito através da apresentação de suas medidas agregadas, geralmente em formato visual, para depois realizar uma análise de dados exploratória, onde se encontram novas relações entre as variáveis existentes. Se a amostra de dados for pequena, pode-se usar análise inferencial para aumentá-la. Caso se deseje prever o futuro a partir de dados do passado, pode-se realizar uma análise preditiva. A análise casual identifica variáveis que afetam umas às outras, com a análise mecanicista explorando como elas afetam umas às outras.

A análise é sempre tão boa quanto os dados sendo usados, destaca ZINOVIEV (2016), que apresenta uma série de ferramentas da linguagem Python que, considerando a experiência do cientista de dados, podem ser utilizadas primeiro para obter dados a partir da web e de outras fontes, em seguida para limpar e organizar estatisticamente estes dados. Então é feita uma análise descritiva e exploratória. Terminando em gráficos de dispersão, histogramas e resumos estatísticos, que ajudam especialmente quando o conjunto de dados possui muitas dimensões. Agora entra a predição de dados, com ferramentas de treinamento de modelos de dados, que permitem aprender com o passado para prever o futuro.

Para KALINOWSKI et. al. (2023), as etapas para um projeto básico de ciência de dados envolvem a definição do problema, coleta e análise de dados, pré-processamento, modelagem e inferência, pós-processamento, apresentação de resultados e implantação do modelo e geração de valor.

Para iniciar a análise de dados, ZINOVIEV (2016) apresenta o pipeline de aquisição de dados, que consiste em obter dados a partir de uma variedade de fontes (source), tais como a internet, arquivos e bancos de dados, convertendo-os em formatos mais adequados para processamento com formato desestruturado (unstructured format), conteúdo texto em linguagem natural ou formato estruturado (structured format), incluindo CSV, dados tabulares, HTML/XML (vindos de páginas da web), JSON, vindos de APIs REST. As representações, considerando a linguagem Python, podem ser estruturas nativas tais como listas, tuplas, conjuntos e dicionários, estruturas do NumPy tais como vetores (array) ou matrizes (matrix) ou estruturas do pandas, tais como data frames and series conforme a figura 2. Busca-se criar um pipeline automatizado com obtenção, limpeza e transformação de dados; análise descritiva e exploratória; e modelagem de dados com predição. Essas técnicas serão vistas no capítulo 2.

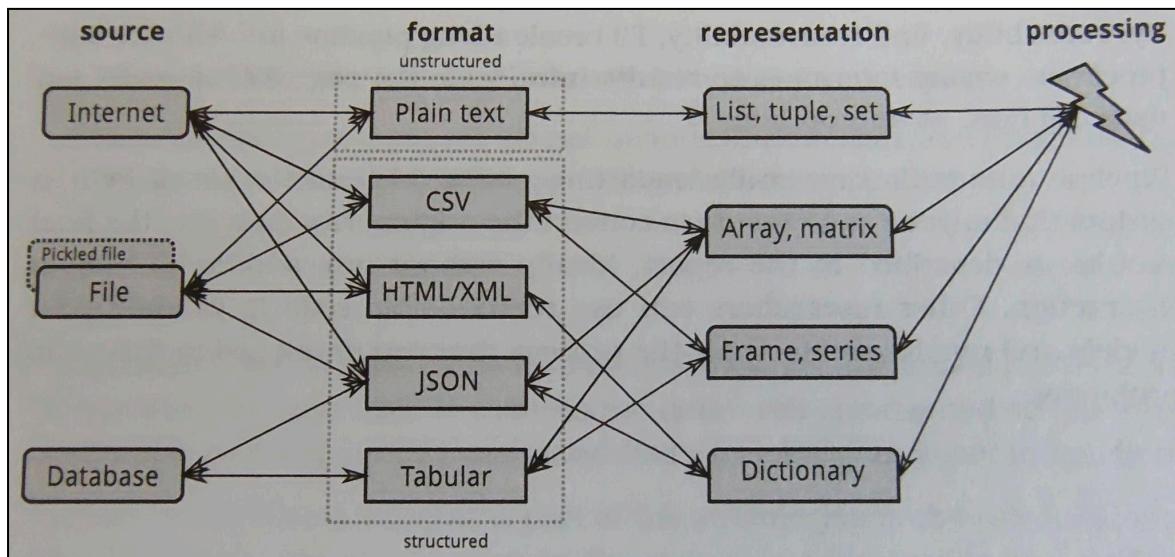


Figura 2. Pipeline de aquisição de dados (ZINOVIEV, 2016)

Big Data

Segundo RUSSOM et al (2011), a necessidade da manipulação de grandes massas de dados era um problema que ficou mais evidente no início dos anos 2000, dado que os recursos computacionais disponíveis à época, não tinham a capacidade de lidar com quantidade de dados que atingiam volumes da ordem de terabytes, gerando uma crise da escalabilidade dos dados. Posteriormente, entretanto, as empresas perceberam que era possível descobrir conjuntos de fatos relevantes para seus negócios, por meio da exploração e análise dessa grande quantidade de dados, também conhecidos como “Big Data”. Na realidade atual, um maior entendimento de qualquer área de negócios passa pela

capacidade das organizações serem capazes de manipular grandes volumes de dados, sendo estes dados os mais detalhados possíveis.

Dessa forma, Big Data pode ser definido como “tecnologias e práticas emergentes que possibilitam a seleção, processamento, armazenamento e geração de insights de grandes volumes de dados estruturados e não estruturados de maneira rápida, efetiva e a um custo acessível. Big Data pode ser considerado como um conjunto de dados que cresce exponencialmente e necessita de habilidades além das quais as ferramentas típicas de gerenciamento e processamento de informações dispõem” (DEVMEDIA, 2020). Sendo as etapas de concepção, coleta de dados, pré-processamento, mineração de dados, análise de conteúdo, visualização de informações e integração de dados, fundamentais para obter a sabedoria, conforme vemos na figura 3.



Figura 3. Etapas para análise de big data (RUSSOM et al, 2011)

Knowledge Discovery in Databases (KDD)

KDD pode ser definido como um processo de várias etapas, não trivial, interativo e iterativo, para identificação de padrões comprehensíveis, válidos, novos e potencialmente úteis a partir de grandes conjuntos de dados. Em KDD, pode-se dividir o processo de construção dos modelos em duas grandes etapas, que são conhecidas por diversos nomes na literatura. Costuma-se dizer que se está aprendendo – ou então treinando, construindo, formulando, induzindo um modelo de conhecimento – a partir de um conjunto de dados quando se procura por padrões nestes dados, em geral utilizando um ou mais algoritmos de Data Mining ou Machine Learning. Finalmente, quando se faz uma estimativa – ou então teste, predição – dos valores desconhecidos para atributos do conjunto de dados, diz-se que o modelo está sendo aplicado (KALINOWSKI et al, 2023)

Introdução à Distribuição Anaconda

Anaconda Distribution é uma distribuição gratuita de ciência de dados Python/R que contém:

- conda - um pacote e um gerenciador de ambiente para sua interface de linha de comando
- Anaconda Navigator - um aplicativo de desktop construído em conda, com opções para lançar outros aplicativos de desenvolvimento a partir de seus ambientes gerenciados
- mais de 250 pacotes científicos e de aprendizagem de máquina

A Anaconda Distribution é gratuita, fácil de instalar e oferece suporte gratuito à comunidade. Para saber mais, acesse começar com a Distribuição Anaconda <<https://docs.anaconda.com/free/anaconda/getting-started/index.html>>.

Ferramentas para Ciência de Dados: Pandas, Numpy e SciPy

- Leitura e escrita de dados
- Integração numérica
- Interpolação
- Álgebra linear

Ferramentas para Machine Learning: TensorFlow, PyTorch e Keras

- Classificação, regressão e agrupamento
- Análise preditiva
- Construir e treinar modelos de aprendizagem profunda

Ferramentas para visualização de dados: matplotlib, bokeh, plotly e Streamlit

- Visualização
- Dashboards
- Widgets

Anaconda

- Conda, gerenciador de pacotes através de linha de comando
- Anaconda navigator, IDE para criar aplicações e gerenciar pacotes, ambientes e canais
- Contém a última versão do python da distribuição Anaconda
- Contém em torno de 300 pacotes para ciência de dados e aprendizagem de máquina.

Miniconda

- Conda, gerenciador de pacotes através de linha de comando
- Contém a última versão do python da distribuição Anaconda

Vamos usar o Miniconda, por ser mais leve e, à medida que precisarmos os pacotes, podemos realizar a instalação.

Ele permite criar um ambiente de programação e compartilhar; instalar pacotes; e utilizar uma IDE.

Conda é uma ferramenta para gerenciar repositórios de pacotes, suas dependências e criar ambientes de programação para diversas linguagens.

Uma vez que um ambiente foi criado, pode-se usar arquivos .yml para compartilhar, por exemplo, ambientes para ML/AI, ETL, Dashboard, etc.

Para baixar e instalar o Miniconda:

- Acesse o site oficial do **Miniconda** <<https://www.anaconda.com/download>>.
- Escolha a versão adequada para o seu sistema operacional (Windows 10).
- Inclua a pasta condabin contida na pasta do miniconda na variável de ambiente path.

Para criar um ambiente de programação, faça os seguintes passos

	Acesse o CMD e execute os seguintes comandos: <ol style="list-style-type: none">1. Criar um ambiente chamado ds (data science) C:\> conda create -n ds python=3.92. Ver os ambientes disponíveis C:\> conda info --env3. Ativar o ambiente ds C:\> conda activate ds4. Instalar pacotes C:\> conda install jupyter
---	---

Jupyter Notebook

É um ambiente que nos permite escrever código e ver os resultados, além de plotar gráficos e tabelas.



Conda passo a passo para acessar o jupyter lab:

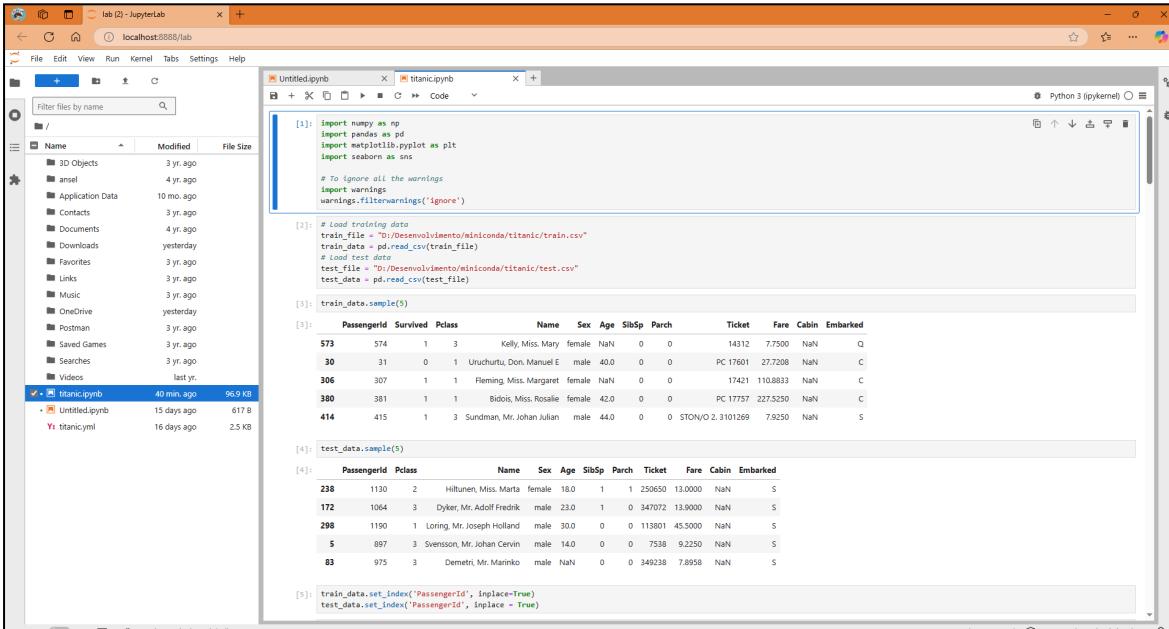
1. Ativar o ambiente

C:\> conda activate ds

2. Executar o Jupyter Lab

(ds) C:\> jupyter lab

3. Acessar o browser, como na figura a seguir



```

[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# To ignore all the warnings
import warnings
warnings.filterwarnings('ignore')

[2]: # Load training data
train_file = "D:/Desenvolvimento/miniconda/titanic/train.csv"
train_data = pd.read_csv(train_file)
# Load test data
test_file = "D:/Desenvolvimento/miniconda/titanic/test.csv"
test_data = pd.read_csv(test_file)

[3]: train_data.sample(5)

[3]:
  PassengerId  Survived  Pclass      Name     Sex   Age SibSp  Parch     Ticket     Fare Cabin Embarked
  573         1        3  Kelly, Miss. Mary  female   NaN    0     0   14312  7.7500   NaN     Q
   30         1        0  Uruchurtu, Don. Manuel E  male  40.0    0     0   PC 17601 27.7208   NaN      C
  306         1        1  Fleming, Miss. Margaret  female   NaN    0     0   17421 110.8833   NaN      C
  380         1        1  Bidois, Miss. Rosalie  female  42.0    0     0   PC 17757 227.5250   NaN      C
  414         1        3  Sundman, Mr. Johan Julian  male  44.0    0     0  STON/O 2.3101269 7.9250   NaN      S

[4]: test_data.sample(5)

[4]:
  PassengerId  Pclass      Name     Sex   Age SibSp  Parch     Ticket     Fare Cabin Embarked
  238         2        2  Hiltunen, Miss. Marta  female  18.0    1    1  250650 13.0000   NaN      S
  172         3        3  Dyker, Mr. Adolf Fredrik  male  23.0    1    0  347072 13.0000   NaN      S
  298         1        1  Loring, Mr. Joseph Holland  male  30.0    0    0  113801 45.5000   NaN      S
   5         3        3  Svensson, Mr. Johan Cervin  male  14.0    0    0   7538  9.2250   NaN      S
   83         3        3  Demetri, Mr. Mariniko  male   NaN    0    0  349238 7.8958   NaN      S

[5]: train_data.set_index('PassengerId', inplace=True)
test_data.set_index('PassengerId', inplace=True)

```

Figura 4. Jupyter Lab

MÓDULO 2: TÉCNICAS DE DATA SCIENCE

- Introdução à programação para Data Science (Python libraries: NumPy, Pandas).
- Técnicas de Limpeza e Preparação de Dados.
- Análise Exploratória de Dados (EDA).
- Visualização de Dados avançada (técnicas interativas e storytelling).
- Estatística Descritiva e Inferencial para Data Science (com foco em testes de hipóteses e análise de variância).

NUMPY

Numerical Python é a biblioteca mais utilizada para processamento com finalidades científicas.

Utiliza basicamente objetos array para troca de dados, e a maioria dos pacotes computacionais com funcionalidades científicas utilizam essa biblioteca para manipulação de dados.

Permite usar alguns recursos como:

- ndarray: array multidimensional otimizado que permite operações aritméticas rápidas, orientadas a arrays.
- funções matemáticas para operações rápidas em arrays de dados inteiros, sem necessidade utilização de laços.
- ferramentas para leitura e escrita de dados um array no disco, além de trabalhar com arquivos em memória.
- recursos para álgebra linear, geração aleatória de números e transformadas de Fourier.
- possibilidade de conectar o NumPy a outras bibliotecas escritas em C, C++ ou FORTRAN.

Realizando algumas operações com arrays, é possível entender a capacidade e velocidade de processamento do NumPy:

Para carregar o NumPy, é necessário fazer a importação:

```
[1]: import numpy as np
```

Através do módulo timeit, é possível identificar o tempo necessário para processar determinadas rotinas. No exemplo da figura abaixo, pode-se comprovar a eficiência do NumPy ao executar a rotina de gerar o quadrado de cada índice, comparando o tempo necessário para realizar a mesma operação com uma rotina via programação tradicional.

Algumas operações com arrays criados a partir do NumPy:

```
m1 = np.array([1, 2, 3, 4, 5, 6])
m2 = np.array([7, 8, 9, 10, 11, 12])
m1 + m2

array([ 8, 10, 12, 14, 16, 18])

m1 * m2

array([ 7, 16, 27, 40, 55, 72])
```

```
[3]: print([n*n for n in range(100)])
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1024
, 1089, 1156, 1225, 1296, 1369, 1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209, 2304, 2401, 2500, 2601, 2704, 2809, 2916, 3025, 3136, 3249,
3364, 3481, 3600, 3721, 3844, 3969, 4096, 4225, 4356, 4489, 4624, 4761, 4900, 5041, 5184, 5329, 5476, 5625, 5776, 5929, 6084, 6241, 6400, 6561, 6724, 6
889, 7056, 7225, 7396, 7569, 7744, 7921, 8100, 8281, 8464, 8649, 8836, 9025, 9216, 9409, 9604, 9801]

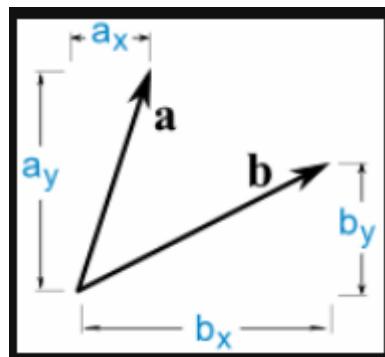
[2]: %timeit [n*n for n in range(1000)]
117 µs ± 6.2 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)

[3]: %timeit np.arange(1000)**2
4.98 µs ± 359 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

A importância do NumPy para processamentos de dados numéricos é devido ao seu projeto, especificamente voltado para eficácia em arrays contendo muitos dados.

Internamente, NumPy faz com que os dados sejam armazenados em um bloco contíguo de memória, seja qual for os tipos de objetos embutidos. O conjunto de algoritmos foram escritos na linguagem C e conseguem acessar a memória utilizada, sem verificação de tipos ou qualquer outro *overhead*.

Outro exemplo, que normalmente é realizado em diversas operações de Data Science e Machine Learning é o processamento vetorial de dados. Ao fazer um produto escalar de dois arrays e comparando a forma tradicional (criando uma função) e o processamento com o NumPy, é possível verificar a diferença na performance das duas estratégias:



O produto escalar é dado pela fórmula:

Para dois vetores $\mathbf{a} = (a_x, a_y)$ e $\mathbf{b} = (b_x, b_y)$:

$$\mathbf{a} \cdot \mathbf{b} = a_x \times b_x + a_y \times b_y$$

Criando uma função para refletir a fórmula e executando a função a partir dos arrays m1 e m2 anteriormente definidos, computando o tempo para geração do resultado:

```

def dot(m1, m2):
    s = 0
    for i in range(1000):
        s += m1[i] * m2[i]
    return s

%timeit dot(m1, m2)
348 µs ± 12.2 µs per loop (mean ± std. dev. of 7 runs, 1,000 loops each)

```

Realizando a comparação com a função existente no NumPy np.dot:

```

%timeit np.dot(m1, m2)
1.29 µs ± 18.8 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops each)

```

Realizando operações mais pesadas:

Vamos fazer uma verificação do desempenho de um array NumPy e uma lista equivalente em Python, contendo 5 milhões de inteiros:

```

[4]: arr_np = np.arange(5000000)
[5]: lst = list(range(5000000))
[6]: %time for _ in range(10): arr2 = arr_np * 2
CPU times: total: 141 ms
Wall time: 147 ms
[7]: %time for _ in range(10): lst2 = [x * 2 for x in lst]
CPU times: total: 5.56 s
Wall time: 6.15 s

```

Na média, os algoritmos do NumPy são de 10 a 100 vezes mais rápidos que o Python puro, e ainda, consumindo bem menos memória.

Os array NumPy utilizam muito menos memória do que as demais sequências embutidas do Python.

As operações perfazem processamentos complexos em arrays inteiros sem utilizar laços for.

OBJETOS ARRAY MULTIDIMENSIONAIS

Um dos grandes diferenciais do NumPy é sua capacidade de lidar com arrays multidimensionais, através de seu objeto ndarray.

Os arrays NumPy são coleções de objetos do mesmo tipo de dados.

Uma das formas de inicializar arrays multidimensionais usando lista de listas pode ser feita da seguinte forma:

```
np.array([range(i, i + 7) for i in [2, 4, 6]])  
  
array([[ 2,  3,  4,  5,  6,  7,  8],  
       [ 4,  5,  6,  7,  8,  9, 10],  
       [ 6,  7,  8,  9, 10, 11, 12]])
```

Especificamente para arrays grandes, é mais eficiente criar arrays a partir de rotinas prontas em NumPy. Vários exemplos podem ser gerados para ilustrar as funcionalidades que podem ser aplicadas:

Criação de um array de inteiros preenchido com zeros, com 20 posições.

```
np.zeros(20, dtype=int)  
  
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Criação de um array de 5x7 do tipo float, preenchido com 1s.

```
np.ones((5, 7), dtype=float)  
  
array([[1., 1., 1., 1., 1., 1., 1.],  
       [1., 1., 1., 1., 1., 1., 1.],  
       [1., 1., 1., 1., 1., 1., 1.],  
       [1., 1., 1., 1., 1., 1., 1.],  
       [1., 1., 1., 1., 1., 1., 1.]])
```

Criação de um array 5x7 preenchido com 3.14.

```
: np.full((5, 7), 3.14)  
  
: array([[3.14, 3.14, 3.14, 3.14, 3.14, 3.14, 3.14],  
        [3.14, 3.14, 3.14, 3.14, 3.14, 3.14, 3.14],  
        [3.14, 3.14, 3.14, 3.14, 3.14, 3.14, 3.14],  
        [3.14, 3.14, 3.14, 3.14, 3.14, 3.14, 3.14],  
        [3.14, 3.14, 3.14, 3.14, 3.14, 3.14, 3.14]])
```

Criação de um array preenchido com uma sequência linear iniciando com o índice 0 até o limite com 40, com passo 3.

```
np.arange(0, 40, 3)  
  
array([ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39])
```

Criação de um array com 6 valores com o mesmo intervalo de distância entre 0 e 1.

```
np.linspace(0, 1, 6)  
array([0. , 0.2, 0.4, 0.6, 0.8, 1. ])
```

Criação de um array 5x5 de inteiros aleatórios no intervalo [0, 10).

```
np.random.randint(0, 10, (5, 5))  
array([[1, 4, 8, 6, 4],  
       [8, 1, 4, 4, 6],  
       [3, 8, 0, 5, 9],  
       [8, 9, 6, 3, 4],  
       [4, 8, 0, 6, 0]])
```

Criação de uma matriz identidade de 5x5:

```
np.eye(5)  
array([[1., 0., 0., 0., 0.],  
       [0., 1., 0., 0., 0.],  
       [0., 0., 1., 0., 0.],  
       [0., 0., 0., 1., 0.],  
       [0., 0., 0., 0., 1.]])
```

Em relação aos tipos de dados, é importante sempre especificar o que for mais adequado para otimização do uso de memória. Os tipos de dados padrão do NumPy são:

- bool_: Boolean (True or False), armazenado como byte;
- int_: tipo inteiro padrão (mesmo que o long da linguagem C, normalmente int64 ou int32);
- intc: idêntico ao int do C (normalmente int32 ou int64);
- intp: inteiro utilizado para indexação (mesmo que tipo ssize_t do C; normalmente int32 ou int64);
- int8: Byte (-128 a 127);
- int16: inteiro (-32768 a 32767);
- int32: inteiro (-2147483648 a 2147483647);
- int64: inteiro (-9223372036854775808 a 9223372036854775807);
- uint8: inteiro positivo (0 a 255);
- uint16: inteiro positivo (0 a 65535);
- uint32: inteiro positivo (0 a 4294967295);
- uint64: inteiro positivo (0 a 18446744073709551615);
- float_ alias para float64;
- float16: float com metade da precisão: bit de sinal, 5 bits de expoente, 10 bits para mantissa;
- float32: float com precisão simples: bit de sinal, 8 bits de expoente, 23 bits para mantissa;
- float64: float de dupla precisão: bit de sinal, 11 bits de expoente, 52 bits para mantissa;

- complex_ alias para complex128;
- complex64 números complexos, representados por 2 floats de 32-bit;
- complex128 números complexos, representados por 2 floats de 64-bit.

A manipulação de dados em Python é quase um sinônimo de manipulação de arrays utilizando NumPy. É possível acessar dados ou subarrays, separar, mudar o formato / dimensões e juntar arrays.

Algumas categorias de manipulações básicas de arrays envolvem:

- Atributos: determinação do tamanho, formato, consumo de memória e os tipos de dados;
- Índices: leitura ou escrita de valores de elementos individuais nos arrays;
- Slicing (fatiamento): geração de arrays menores a partir de um array;
- Redimensionamento (reshape): alteração de dimensões dos arrays;
- Junção ou separação: combinação de múltiplos arrays em um único, ou separando um array em vários.

ATRIBUTOS

Para exemplificar os atributos dos arrays, serão criados 4 arrays, respectivamente de 1, 2, 3 e 4 dimensões, e verificados os atributos:

```
import numpy as np

np.random.seed(0)

x1 = np.random.randint(10, size=6) # array de 1 dimensão
x2 = np.random.randint(10, size=(3, 4)) # array de 2 dimensões
x3 = np.random.randint(10, size=(3, 4, 5)) # array de 3 dimensões
x4 = np.random.randint(10, size=(3, 4, 5, 6)) # array de 4 dimensões

print("dim x4: ", x4.ndim)
print("shape x4:", x4.shape)
print("tamanho x4: ", x4.size)

dim x4: 4
shape x4: (3, 4, 5, 6)
tamanho x4: 360
```

```
print("dtype:", x4.dtype)
dtype: int32

print("itemsize:", x4.itemsize, "bytes")
print(" nbytes:", x4.nbytes, "bytes")

itemsize: 4 bytes
nbytes: 1440 bytes
```

ÍNDICES

A indexação nos arrays é bastante similar à indexação padrão de listas. Cada array inicia com o índice 0, e pode-se acessar o n-ésimo valor utilizando colchetes, como nos exemplos da figura a seguir.

```
x1  
array([5, 0, 3, 3, 7, 9])  
x1[0]  
5  
x1[3]  
3  
x1[-1]  
9  
x1[-2]  
7
```

Já nos arrays multidimensionais, é necessário indicar os índices dependendo do formato (dimensões):

```
x2  
array([[3, 5, 2, 4],  
       [7, 6, 8, 8],  
       [1, 6, 7, 7]])  
x2[0, 0]  
3  
x2[2, 0]  
1  
x2[0, 2]  
2  
x2[2, -1]  
7
```

É possível também atribuir valores:

```
x2[0, 0] = 30  
x2  
array([[30, 5, 2, 4],  
       [7, 6, 8, 8],  
       [1, 6, 7, 7]])
```

FATIAMENTO (SLICING)

Pode-se utilizar colchetes para acessar subarrays, incluindo a notação de fatiamento (*slicing*) em conjunto com o caractere “`:`”

`x[início:fim:passo]`

Os exemplos ilustram respectivamente, a geração de um array de 10 posições; os 5 primeiros elementos; os elementos após o índice 5; extração de uma subarray:

```
x = np.arange(10)

x
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

x[:5]
array([0, 1, 2, 3, 4])

x[5:]
array([5, 6, 7, 8, 9])

x[4:7]
array([4, 5, 6])
```

Os próximos exemplos ilustram a utilização do terceiro parâmetro (passo) na realização de fatiamentos:

```

# Todos os elementos a partir do início, com passo 2
x[::2]

array([0, 2, 4, 6, 8])

# Todos os elementos a partir do início, com passo 3
x[::3]

array([0, 3, 6, 9])

# Todos os elementos a partir do índice 2, com passo 2
x[2::2]

array([2, 4, 6, 8])

# Todos os elementos com inversão do array
x[::-1]

array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])

# Todos os elementos a partir do índice 5 até o primeiro, com passo 2
x[5::-2]

array([5, 3, 1])

```

REDIMENSIONAMENTO (RESHAPING)

Outra operação importante e bastante útil é o redimensionamento de arrays. A forma mais flexível de fazer esse procedimento é utilizar o método **reshape()**.

Como exemplo, para um array de 15 elementos, é possível redimensionar de várias maneiras, desde que o produto do número de colunas por número de linhas seja igual a 15.

```

x0 = np.arange(0, 15)

x0
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])

x0_r1 = x0.reshape(3, 5)

x0_r1
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])

x0_r2 = x0.reshape(5, 3)

x0_r2
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11],
       [12, 13, 14]])

```

JUNÇÃO E SEPARAÇÃO DE ARRAYS

É possível juntar múltiplos arrays, a partir da concatenação de forma horizontal ou vertical. Da mesma forma, é possível converter uma matriz em várias. Esses procedimentos serão exemplificados assim.

```
x = np.array([1, 2, 3, 4])
y = np.array([4, 3, 2, 1])
np.concatenate([x, y])

array([1, 2, 3, 4, 4, 3, 2, 1])
```

É possível concatenar mais de 2 arrays:

```
z = [99, 99, 99, 99]
print(np.concatenate([x, y, z]))

[ 1  2  3  4  4  3  2  1 99 99 99 99]
```

Podem ser concatenados arrays com n dimensões:

```
a1 = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 7, 8]])

np.concatenate([a1, a1])

array([[1, 2, 3],
       [4, 5, 6],
       [7, 7, 8],
       [1, 2, 3],
       [4, 5, 6],
       [7, 7, 8]])
```

Uma outra forma de concatenar horizontalmente ou verticalmente é utilizando os métodos **vstack** (junção vertical) e **hstack** (junção horizontal).

```

x = np.array([1, 2, 3])
a2 = np.array([[9, 8, 7],
              [6, 5, 4]])

np.vstack([x, a2])

array([[1, 2, 3],
       [9, 8, 7],
       [6, 5, 4]])

y = np.array([[99],
              [99]])

np.hstack([a2, y])

array([[ 9,  8,  7, 99],
       [ 6,  5,  4, 99]])

```

PANDAS

CONCEITOS BÁSICOS

O módulo Pandas é o principal do Python utilizado para realizar tarefas com dados.

Uma de suas grandes vantagens é o uso de estruturas chamadas *DataFrames* e séries, que são estruturas de dados em 2 dimensões rápidas e flexíveis. O pandas oferece um conjunto de recursos que facilitam a leitura e manipulação de dados, e ainda, métodos eficientes para realizar processos como limpeza, padronização e análise exploratória.

A partir do momento que conseguirmos realizar a leitura de quaisquer dados pelo Pandas, já é possível atribuir o resultado da leitura para um *dataframe*.

Para a leitura de um arquivo com colunas separadas por vírgulas (csv) , é necessário apenas indicar o caminho para a função **read_csv**. Ao ler e atribuir o resultado para uma variável o Python já irá entender que o tipo da variável é um DataFrame. O arquivo em questão foi gerado a partir do site do IBGE, contendo a relação de todos os municípios do Brasil, com as colunas (codigo_ibge, nome, latitude, longitude, capital (0 indica False e 1 True) e codigo_uf, com o código da unidade da federação que será utilizada em outro arquivo. Além das colunas, o pandas cria para cada DataFrame um índice automático, a partir de um sequencial gerado iniciando do zero até o número de linhas -1 do arquivo lido.

Para visualizar os dados armazenados na variável, basta digitar seu nome e executar a célula. Ao final, o pandas irá indicar o total de linhas x total de colunas.

f

Para confirmar o tipo de dados da variável, pode-se utilizar a função **type**:

```
print(type(df_mun))  
<class 'pandas.core.frame.DataFrame'>
```

O método head() pode ser usado para mostrar as 5 primeiras linhas do dataframe, possibilitando assim uma visualização mais rápida e com menos dados na tela:

df_mun.head()						
	codigo_ibge	nome	latitude	longitude	capital	codigo_uf
0	5200050	Abadia de Goiás	-16.75730	-49.4412	0	52
1	3100104	Abadia dos Dourados	-18.48310	-47.3916	0	31
2	5200100	Abadiânia	-16.19700	-48.7057	0	52
3	3100203	Abaeté	-19.15510	-45.4444	0	31
4	1500107	Abaetetuba	-1.72183	-48.8788	0	15

O método tail() mostra as 5 últimas linhas do dataframe:

df_mun.tail()						
	codigo_ibge	nome	latitude	longitude	capital	codigo_uf
5565	2933604	Xique-Xique	-10.82300	-42.7245	0	29
5566	2517407	Zabelê	-8.07901	-37.1057	0	25
5567	3557154	Zacarias	-21.05060	-50.0552	0	35
5568	2114007	Zé Doca	-3.27014	-45.6553	0	21
5569	4219853	Zortéa	-27.45210	-51.5520	0	42

Caso seja necessário visualizar mais linhas ao utilizar um desses métodos, basta indicar um número no parâmetro de entrada da função:

df_mun.head(15)						
	codigo_ibge	nome	latitude	longitude	capital	codigo_uf
0	5200050	Abadia de Goiás	-16.75730	-49.4412	0	52
1	3100104	Abadia dos Dourados	-18.48310	-47.3916	0	31
2	5200100	Abadiânia	-16.19700	-48.7057	0	52
3	3100203	Abaeté	-19.15510	-45.4444	0	31
4	1500107	Abaetetuba	-1.72183	-48.8788	0	15
5	2300101	Abaiara	-7.34588	-39.0416	0	23
6	2900108	Abaíra	-13.24880	-41.6619	0	29
7	2900207	Abaré	-8.72073	-39.1162	0	29
8	4100103	Abatiá	-23.30490	-50.3133	0	41
9	4200051	Abdon Batista	-27.61260	-51.0233	0	42
10	1500131	Abel Figueiredo	-4.95333	-48.3933	0	15
11	4200101	Abelardo Luz	-26.57160	-52.3229	0	42
12	3100302	Abre Campo	-20.29960	-42.4743	0	31
13	2600054	Abreu e Lima	-7.90072	-34.8984	0	26
14	1700251	Abreulândia	-9.62101	-49.1518	0	17

O objeto DataFrame possui a propriedade **shape**, que retorna o número de linhas e colunas sem ter que mostrar uma grande quantidade de dados na tela.

```
print(df_mun.shape)  
(5570, 6)
```

A partir disso, é possível, por exemplo, atribuir essas propriedades para variáveis, ou diretamente, ou realizando o fatiamento pela posição desejada.

```
num_linhas, num_colunas = df_mun.shape

num_linhas

5570

num_colunas

6

nlinhas = df_mun.shape[0]

nlinhas

5570
```

Outra propriedade útil é a **columns**, que retorna o nome de todas as colunas do DataFrame em uma lista.

```
print(df_mun.columns)

Index(['codigo_ibge', 'nome', 'latitude', 'longitude', 'capital', 'codigo_uf'], dtype='object')
```

A propriedade **dtypes** retorna o tipo de cada coluna do DataFrame.

```
print(df_mun.dtypes)

codigo_ibge      int64
nome            object
latitude       float64
longitude      float64
capital        int64
codigo_uf       int64
dtype: object
```

Sempre que um DataFrame for aberto, é importante entender a estrutura dos dados e seu conteúdo. Esse é o ponto inicial da análise exploratória dos dados, etapa importante para que os dados sejam entendidos. O método **info()** mostra uma visão geral do DataFrame, retornando: número de linhas, número de colunas, tipos de dados das colunas, memória utilizada para armazenar os dados (espaço em disco) e quantidade de valores nulos existentes em cada coluna.

```

print(df_mun.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5570 entries, 0 to 5569
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   codigo_ibge  5570 non-null   int64  
 1   nome         5570 non-null   object  
 2   latitude     5570 non-null   float64 
 3   longitude    5570 non-null   float64 
 4   capital      5570 non-null   int64  
 5   codigo_uf    5570 non-null   int64  
dtypes: float64(2), int64(3), object(1)
memory usage: 261.2+ KB

```

Um outro método bastante útil é o **describe()**, que retorna informações estatísticas a partir das colunas numéricas. As informações são: contagem de registros, média, desvio padrão, valor mínimo, valor máximo e os percentis, que são medidas referentes à distribuição dos dados, indicando a medida de assimetria da distribuição de probabilidade em relação à média.

df_mun.describe()					
	codigo_ibge	latitude	longitude	capital	codigo_uf
count	5.570000e+03	5570.000000	5570.000000	5570.000000	5570.000000
mean	3.253591e+06	-16.449144	-46.231003	0.004847	32.377738
std	9.849103e+05	8.287237	6.408539	0.069461	9.833862
min	1.100015e+06	-33.686600	-72.899700	0.000000	11.000000
25%	2.512126e+06	-22.843875	-50.878525	0.000000	25.000000
50%	3.146280e+06	-18.094300	-46.523200	0.000000	31.000000
75%	4.119190e+06	-8.496445	-41.410775	0.000000	41.000000
max	5.300108e+06	4.603140	-32.410700	1.000000	53.000000

Uma forma de realizar consultas / aplicar filtros nos dados do DataFrame é utilizar o método **query()**.

df_mun.query("nome=='Manaus'")					
	codigo_ibge	nome	latitude	longitude	capital
2885	1302603	Manaus	-3.11866	-60.0212	1

É possível também gerar outros dataframes com subconjuntos de dados baseados em determinadas colunas originalmente lidas. Isso otimiza o uso de memória, no caso de armazenar resultados em variáveis, além de possibilitar uma

melhor visualização em conjuntos de dados com grande número de colunas. Para isso, é necessário indicar uma lista de listas, com colchetes duplos:

subs1 = df_mun[["codigo_ibge", "nome", "capital"]]			
subs1.head(10)			
	codigo_ibge	nome	capital
0	5200050	Abadia de Goiás	0
1	3100104	Abadia dos Dourados	0
2	5200100	Abadiânia	0
3	3100203	Abaeté	0
4	1500107	Abaetetuba	0
5	2300101	Abaíara	0
6	2900108	Abaíra	0
7	2900207	Abaré	0
8	4100103	Abatiá	0
9	4200051	Abdon Batista	0

O método query pode ser usado de diversas formas, combinando critérios de diversas formas, conforme os exemplos a seguir:

df_mun.query(("codigo_uf==13 & capital==1"))						
	codigo_ibge	nome	latitude	longitude	capital	codigo_uf
2885	1302603	Manaus	-3.11866	-60.0212	1	13

df_mun.query(("codigo_uf==13") and ("capital==1"))						
3947	4314902	Porto Alegre	-30.031800	-51.2065	1	43
3977	1100205	Porto Velho	-8.760770	-63.8999	1	11
4110	2611606	Recife	-8.046660	-34.8771	1	26
4193	1200401	Rio Branco	-9.974990	-67.8243	1	12
4207	3304557	Rio de Janeiro	-22.912900	-43.2003	1	33
4332	2927408	Salvador	-12.971800	-38.5011	1	29
4810	2111300	São Luís	-2.538740	-44.2825	1	21
4853	3550308	São Paulo	-23.532900	-46.6395	1	35
5209	2211001	Teresina	-5.091940	-42.8034	1	22
5533	3205309	Vitória	-20.315500	-40.3128	1	32

df_mun.loc[df_mun.latitude > 0]						
	codigo_ibge	nome	latitude	longitude	capital	codigo_uf
141	1400050	Alto Alegre	2.988580	-61.3072	0	14
184	1400027	Amajari	3.645710	-61.3692	0	14
186	1600105	Amapá	2.052670	-50.7957	0	16
642	1400100	Boa Vista	2.823840	-60.6753	1	14
707	1400159	Bonfim	3.361610	-59.8333	0	14
904	1600204	Calçoene	2.504750	-50.9512	0	16
1038	1400175	Cantá	2.609940	-60.6058	0	14
1086	1400209	Caracaraí	1.827660	-61.1304	0	14
1145	1400233	Caroebe	0.884203	-59.6959	0	14
1525	1600212	Cutias	0.970761	-50.8005	0	16
1780	1600238	Ferreira Gomes	0.857256	-51.1795	0	16
2256	1400282	Iracema	2.183050	-61.0415	0	14
2423	1600253	Itaubal	0.602185	-50.6996	0	16
2831	1600303	Macapá	0.034934	-51.0694	1	16

df_mun.query('capital == 1 & latitude > 0')						
	codigo_ibge	nome	latitude	longitude	capital	codigo_uf
642	1400100	Boa Vista	2.823840	-60.6753	1	14
2831	1600303	Macapá	0.034934	-51.0694	1	16

Retornando o total de municípios por UF:

```
df_mun.groupby(by='codigo_uf').size()

  codigo_uf
  11      52
  12      22
  13      62
  14      15
  15     144
  16      16
  17     139
  21     217
  22     224
  23     184
  24     167
  25     223
  26     185
  27     102
  28      75
  29     417
  31     853
  32      78
  33      92
  35     645
  41     399
  42     295
  43     497
  50      79
  51     141
  52     246
  53      1
  dtype: int64
```

MÉTODO ISIN()

Pode ser utilizado para passar listas de valores e realizar filtros, conforme o exemplo:

df_mun.loc[df_mun['nome'].isin(['Manaus', 'Itacoatiara'])]						
	codigo_ibge	nome	latitude	longitude	capital	codigo_uf
2300	1301902	Itacoatiara	-3.13861	-58.4449	0	13
2885	1302603	Manaus	-3.11866	-60.0212	1	13

MÉTODOS LOC E ILOC

Os métodos **loc** e **iloc** permitem o acesso baseado em linhas ou colunas do dataframe.

O método **loc** é baseado nos rótulos das colunas. No exemplo, o método é utilizado para filtrar somente algumas colunas, em conjunto com a seleção das 10 primeiras linhas:

	nome	codigo_ibge
0	Abadia de Goiás	5200050
1	Abadia dos Dourados	3100104
2	Abadiânia	5200100
3	Abaeté	3100203
4	Abaetetuba	1500107
5	Abaíara	2300101
6	Abaíra	2900108
7	Abaré	2900207
8	Abatiá	4100103
9	Abdon Batista	4200051
10	Abel Figueiredo	1500131

É possível também retornar todos os dados de determinado índice. Neste caso, é utilizado o índice gerado automaticamente pelo pandas ao ler o arquivo.

```
print(df_mun.loc[1050])  
codigo_ibge      3510203  
nome            Capão Bonito  
latitude        -24.0113  
longitude       -48.3482  
capital          0  
codigo_uf        35  
Name: 1050, dtype: object
```

Pode-se também retornar faixas (intervalos) de linhas, as partir do índice automático gerado no DataFrame:

```

print(df_mun.loc[2880:2900])

   codigo_ibge      nome  latitude  longitude  capital  codigo_uf
2880      4311734  Mampituba -29.21360 -49.9311    0       43
2881     1302504  Manacapuru -3.29066 -60.6216    0       13
2882     2509008     Manaíra -7.70331 -38.1523    0       25
2883     1302553  Manaquiri -3.44078 -60.4612    0       13
2884     2609154     Manari -8.96490 -37.6313    0       26
2885     1302603    Manaus -3.11866 -60.0212    1       13
2886    1200336  Mâncio Lima -7.61657 -72.8997    0       12
2887    4114104  Mandaguáçu -23.34580 -52.0944    0       41
2888    4114203  Mandaguari -23.54460 -51.6710    0       41
2889    4114302  Mandirituba -25.77700 -49.3282    0       41
2890    3528601     Manduri -23.00560 -49.3202    0       35
2891    4114351  Manfrinópolis -26.14410 -53.3113    0       41
2892    3139300      Manga -14.75290 -43.9391    0       31
2893    3302601  Mangaratiba -22.95940 -44.0409    0       33
2894    4114401  Mangueirinha -25.94210 -52.1743    0       41
2895    3139409    Manhuaçu -20.25720 -42.0280    0       31
2896    3139508  Manhumirim -20.35910 -41.9589    0       31
2897    1302702    Manicoré -5.80462 -61.2895    0       13
2898    2205904  Manoel Emídio -8.01234 -43.8755    0       22
2899    4114500  Manoel Ribas -24.51440 -51.6658    0       41
2900    1200344  Manoel Urbano -8.83291 -69.2679    0       12

```

Diferentemente do fatiamento em dados com o comportamento de lista, ao fatiar utilizando o loc o limite superior é incluído no resultado.

Outra possibilidade é indicar determinados índices para retornar as linhas:

```

print(df_mun.loc[[2881, 2883, 2885, 2897]])

   codigo_ibge      nome  latitude  longitude  capital  codigo_uf
2881     1302504  Manacapuru -3.29066 -60.6216    0       13
2883     1302553  Manaquiri -3.44078 -60.4612    0       13
2885     1302603    Manaus -3.11866 -60.0212    1       13
2897    1302702  Manicoré -5.80462 -61.2895    0       13

```

Uma prática que pode ser vantajosa é utilizar um índice baseado em valor de coluna, em vez de utilizar o índice gerado pelo pandas. Com isso, passa a ser possível realizar filtros pelos valores dessa coluna.

```

df_mun2 = df_mun.set_index("nome")

df_mun2.head()

   codigo_ibge  latitude  longitude  capital  codigo_uf
nome
Abadia de Goiás      5200050 -16.75730 -49.4412    0       52
Abadia dos Dourados    3100104 -18.48310 -47.3916    0       31
Abadiânia            5200100 -16.19700 -48.7057    0       52
Abaeté              3100203 -19.15510 -45.4444    0       31
Abaetetuba          1500107 -1.72183 -48.8788    0       15

```

Ao indicar para o pandas qual a coluna que deve ser utilizada como índice, o DataFrame df_mun2 passa a ser acessado pelos nomes dos municípios.

```
df_mun2.loc["Manaus"]
```

codigo_ibge	1.302603e+06
latitude	-3.118660e+00
longitude	-6.002120e+01
capital	1.000000e+00
codigo_uf	1.300000e+01
Name:	Manaus, dtype: float64

Um detalhe importante é que ao utilizar um campo do tipo string como índice os filtros passam a utilizar o critério de case sensitive, e é necessário ter o cuidado de digitar os nomes dos municípios da mesma forma que no arquivo original. Caso contrário, será gerado um erro.

```
df_mun2.loc[['Manaus', 'Parintins', 'Itacoatiara', 'Presidente Figueiredo']]
```

```
KeyError: "[ 'Presidente Figueiredo' ] not in index"
```

Ao resolver a questão da letra “f” que deveria estar maiúscula, os dados são retornados:

```
df_mun2.loc[['Manaus', 'Parintins', 'Itacoatiara', 'Presidente Figueiredo']]
```

	codigo_ibge	latitude	longitude	capital	codigo_uf
nome					
Manaus	1302603	-3.11866	-60.0212	1	13
Parintins	1303403	-2.63741	-56.7290	0	13
Itacoatiara	1301902	-3.13861	-58.4449	0	13
Presidente Figueiredo	1303536	-2.02981	-60.0234	0	13

O método **iloc** possibilita a seleção por números inteiros de linhas.

Seleção da primeira linha do DataFrame:

```
df_mun.iloc[0]
```

codigo_ibge	5200050
nome	Abadia de Goiás
latitude	-16.7573
longitude	-49.4412
capital	0
codigo_uf	52
Name:	0, dtype: object

Da mesma forma, qualquer índice pode ser indicado.

```
print(df_mun.iloc[35])
```

codigo_ibge	2600104
nome	Afogados da Ingazeira
latitude	-7.74312
longitude	-37.631
capital	0
codigo_uf	26
Name:	35, dtype: object

```
print(df_mun.iloc[-1])
```

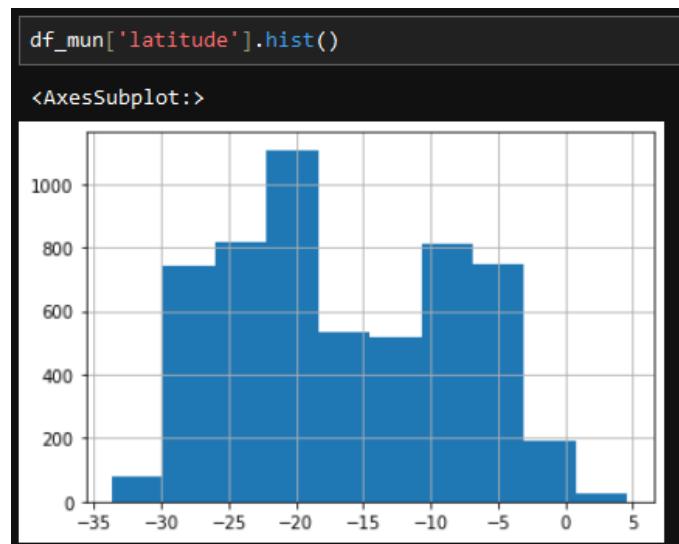
codigo_ibge	4219853
nome	Zortéa
latitude	-27.4521
longitude	-51.552
capital	0
codigo_uf	42
Name:	5569, dtype: object

Realizando ao mesmo tempo a seleção de linhas e colunas utilizando fatiamento em conjunto com iloc:

df_mun.iloc[0:10, 0:2]		
	codigo_ibge	nome
0	5200050	Abadia de Goiás
1	3100104	Abadia dos Dourados
2	5200100	Abadiânia
3	3100203	Abaeté
4	1500107	Abaetetuba
5	2300101	Abaiara
6	2900108	Abaíra
7	2900207	Abaré
8	4100103	Abatiá
9	4200051	Abdon Batista

ANÁLISE DE FREQUÊNCIA

Um outro método importante é o `hist()`, que permite de forma bem rápida verificar a distribuição dos dados a partir da frequência, demonstrando graficamente os histogramas. O eixo X representa intervalos de valores encontrados na coluna, e no eixo y a contagem / frequência. O gráfico apresenta, portanto, a contagem de municípios de cada faixa existente de latitude.



O dataset `london.csv` passará a ser utilizado para a realização de filtros e exercícios.

Para criar o arquivos london.csv, acesse o arquivo `athlete_events.csv` no link a seguir:

<https://www.kaggle.com/datasets/heesoo37/120-years-of-olympic-history-at-athletes-and-results?resource=download>

Em seguida, utilize os seguintes comandos para gerar o arquivo london.csv.

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

athletes_file = "D:/Desenvolvimento/miniconda/london/athlete_events.csv"
athletes_data = pd.read_csv(athletes_file)

london = athletes_data.query("City=='London'")[["Sport", "Age", "Height", "Weight", "Sex"]]

london.to_csv('D:/Desenvolvimento/miniconda/london/london.csv', index=False)
```

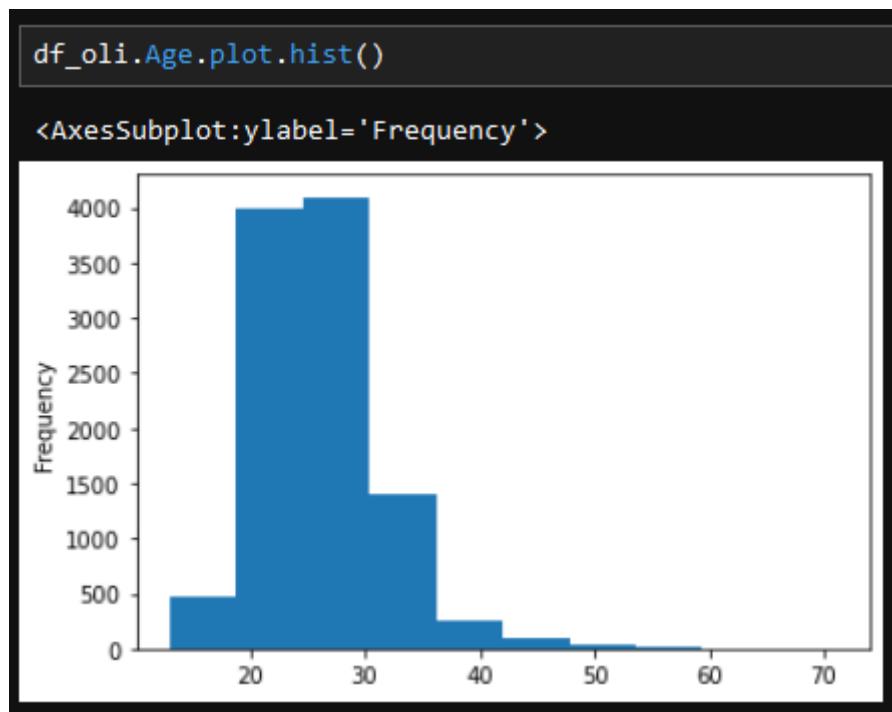
Observe que você poderá salvar na pasta de sua preferência. Então siga os passos a seguir com o arquivo london.csv salvo anteriormente:

```
df_oli = pd.read_csv('datasets/london.csv')

df_oli.head()
```

	Total	Sport	Age	Height	Weight	Sex
0	0	Judo	23	170.0	60.0	M
1	0	Athletics	33	193.0	125.0	M
2	0	Athletics	30	187.0	76.0	M
3	0	Boxing	24	NaN	NaN	M
4	0	Athletics	26	178.0	85.0	F

Gerando um histograma que ilustra o total de atletas baseado na idade:



EXERCÍCIOS:

1. Mostre um histograma baseado na altura dos jogadores de basquete.
2. Faça um refinamento, filtrando para o histograma somente jogadores de basquete do sexo masculino.

O dataset Titanic é um exemplo clássico de um problema de classificação, no qual o objetivo é prever uma variável de resultado categórica (neste caso, se um passageiro sobreviveu ou não) com base em um conjunto de características de entrada (como idade, sexo, classe do passageiro, etc). Um dos principais desafios deste dataset é a fase de transformação e limpeza de dados, lidando com valores ausentes e decidir como transformá-los ou eliminá-los da análise. Como o foco da disciplina não é a aplicação de aprendizado de máquina, podemos trabalhar até o pré-processamento de dados. As instruções e os dados poderão ser obtidos a partir deste link: <https://www.kaggle.com/code/manishkc06/learn-basics-of-data-science-using-titanic-dataset/notebook>

REALIZANDO MERGE EM DATAFRAMES

Normalmente, podemos complementar dados de diversas fontes para tornar mais abrangente e representar de uma forma mais adequada os filtros que podem ser realizados. O DataFrame de municípios contém uma coluna `codigo_uf` que representa a unidade da federação, e em bancos de dados normalmente bem

normalizados, essa coluna será representada como uma **chave estrangeira**, e a descrição estará contida em outra tabela. No caso, vamos utilizar um novo arquivo csv chamado “estados.csv” para realizar o merge (junção) e possibilitar dessa forma a representação do nome da unidade da federação tanto em filtros como nos resultados das consultas realizadas.

	codigo_uf	uf	nome
0	11	RO	Rondônia
1	12	AC	Acre
2	13	AM	Amazonas
3	14	RR	Roraima
4	15	PA	Pará
5	16	AP	Amapá
6	17	TO	Tocantins
7	21	MA	Maranhão
8	22	PI	Piauí
9	23	CE	Ceará

O método Merge possibilita a junção dos 2 dataframes, sendo necessário indicar a coluna que serve de ligação (chave primária / chave estrangeira) nos dois arquivos:

df_mun.merge(df_est, how='inner', on = "codigo_uf")								
	codigo_ibge	nome_x	latitude	longitude	capital	codigo_uf	uf	nome_y
0	5200050	Abadia de Goiás	-16.7573	-49.4412	0	52	GO	Goiás
1	5200100	Abadiânia	-16.1970	-48.7057	0	52	GO	Goiás
2	5200134	Acreúna	-17.3960	-50.3749	0	52	GO	Goiás
3	5200159	Adelândia	-16.4127	-50.1657	0	52	GO	Goiás
4	5200175	Água Fria de Goiás	-14.9778	-47.7823	0	52	GO	Goiás
...
5565	3306107	Valença	-22.2445	-43.7129	0	33	RJ	Rio de Janeiro
5566	3306156	Varre-Sai	-20.9276	-41.8701	0	33	RJ	Rio de Janeiro
5567	3306206	Vassouras	-22.4059	-43.6686	0	33	RJ	Rio de Janeiro
5568	3306305	Volta Redonda	-22.5202	-44.0996	0	33	RJ	Rio de Janeiro
5569	5300108	Brasília	-15.7795	-47.9297	1	53	DF	Distrito Federal
5570 rows × 8 columns								

LEITURA DE DADOS EM PÁGINAS WEB:

O pandas possibilita a leitura de tabelas estruturadas em páginas web, a partir da utilização do **método read_html()**.

Para exemplificar, será utilizado um link contendo dados do estado do Amazonas, em: <[https://en.wikipedia.org/wiki/Amazonas_\(Brazilian_state\)](https://en.wikipedia.org/wiki/Amazonas_(Brazilian_state))>.

```
import pandas as pd

tab_am = pd.read_html('https://en.wikipedia.org/wiki/Amazonas_(Brazilian_state)')

tab_am

23                               GDP
24                               • Year
25                               • Total
26                               • Per capita
27                               HDI
28                               • Year
29                               • Category
30                               Time zones
31                               NaN
32                               Postal Code
33                               ISO 3166 code
34                               License Plate Letter Sequence
35                               Website

                                Amazonas.1
0                           State
1   State of AmazonasEstado de Amazonas (Portuguese)
2 .mw-parser-output .ib-settlement-cols{text-align...
```

A leitura trouxe todas as tabelas identificadas no link, conforme o resultado abaixo:

```
print(f'Total de tabelas: {len(tab_am)}')

Total de tabelas: 21
```

Ajustando para filtrar somente a tabela de interesse, que são os dados de crescimento populacional por ano (usando a propriedade match):

```
tab_am_pop = pd.read_html('https://en.wikipedia.org/wiki/Amazonas_(Brazilian_state)',  
                           match = 'Historical population')  
  
tab_am_pop  
  
[  
 0    Year      Pop.      ±%  
 1  1872     57610      -  
 2  1890    147915  +156.8%  
 3  1900    249756  +68.9%  
 4  1920    363166  +45.4%  
 5  1940    438008  +20.6%  
 6  1950    514099  +17.4%  
 7  1960    721215  +40.3%  
 8  1970   960934  +33.2%  
 9  1980  1449135  +50.8%  
10  1991  2102901  +45.1%  
11  2000  2817252  +34.0%  
12  2010  3483985  +23.7%  
13  2022  3941613  +13.1%  
14 Source:[1]  Source:[1]  Source:[1]]
```

O resultado é uma lista, e contém ainda alguns dados indesejados, além de não ser ainda um DataFrame. Para realizar os ajustes e deixar o dado da forma que é esperada, podemos filtrar somente o índice 0 dos dados:

df_am_pop = tab_am_pop[0]			
df_am_pop			
	Year	Pop.	±%
0	1872	57610	—
1	1890	147915	+156.8%
2	1900	249756	+68.9%
3	1920	363166	+45.4%
4	1940	438008	+20.6%
5	1950	514099	+17.4%
6	1960	721215	+40.3%
7	1970	960934	+33.2%
8	1980	1449135	+50.8%
9	1991	2102901	+45.1%
10	2000	2817252	+34.0%
11	2010	3483985	+23.7%
12	2022	3941613	+13.1%
13	Source:[1] Source:[1] Source:[1]		

O DataFrame ainda contém dados indesejados na última linha, e suas colunas estão do tipo object, conforme figura abaixo.

```
df_am_pop.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Year     14 non-null    object 
 1   Pop.    14 non-null    object 
 2   ±%      14 non-null    object 
dtypes: object(3)
memory usage: 464.0+ bytes
```

Para resolver o primeiro problema, basta fatiar todas as linhas com exceção da última:

df_am_pop[:12]			
	Year	Pop.	±%
0	1872	57610	—
1	1890	147915	+156.8%
2	1900	249756	+68.9%
3	1920	363166	+45.4%
4	1940	438008	+20.6%
5	1950	514099	+17.4%
6	1960	721215	+40.3%
7	1970	960934	+33.2%
8	1980	1449135	+50.8%
9	1991	2102901	+45.1%
10	2000	2817252	+34.0%
11	2010	3483985	+23.7%

df_am_pop_ = df_am_pop[:12]			
	Year	Pop.	±%
0	1872	57610	—
1	1890	147915	+156.8%
2	1900	249756	+68.9%
3	1920	363166	+45.4%
4	1940	438008	+20.6%
5	1950	514099	+17.4%
6	1960	721215	+40.3%
7	1970	960934	+33.2%
8	1980	1449135	+50.8%
9	1991	2102901	+45.1%
10	2000	2817252	+34.0%
11	2010	3483985	+23.7%

Resta agora resolver o problema dos tipos das colunas, realizando a conversão dos dados para inteiro.

```
df_am_pop_.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Year     12 non-null    object  
 1   Pop.    12 non-null    object  
 2   ±%      12 non-null    object  
dtypes: object(3)
memory usage: 416.0+ bytes

df_am_pop_.dtypes
Year    object
Pop.   object
±%    object
dtype: object
```

E após isso, transformar para um dataframe:

TÉCNICAS DE LIMPEZA E PREPARAÇÃO DE DADOS

Assim como em qualquer sistema de informação, é importantíssimo que os dados estejam íntegros, sem redundância ou inconsistências. Um dos principais problemas que causam a geração de informações incorretas é a utilização de dados sem critérios. Alguns dos problemas que devem ser verificados são:

- Valores incompatíveis com tipo de dados da coluna;
- Valores valores muito pequenos (ou muito grandes) em relação ao valor esperado;
- Valores ausentes;

Em relação aos valores ausentes, é possível de forma rápida identificar a quantidade de linhas apresentando essa situação por coluna do DataFrame. O dataset utilizado nos exemplos é o “london.csv” anteriormente aberto.

```

import pandas as pd
import matplotlib.pyplot as plt

ausentes = df_oli.isna().sum()
print("Valores ausentes por coluna:")
print(ausentes)

Valores ausentes por coluna:
Total      0
Sport      0
Age        0
Height     561
Weight    1280
Sex        0
dtype: int64

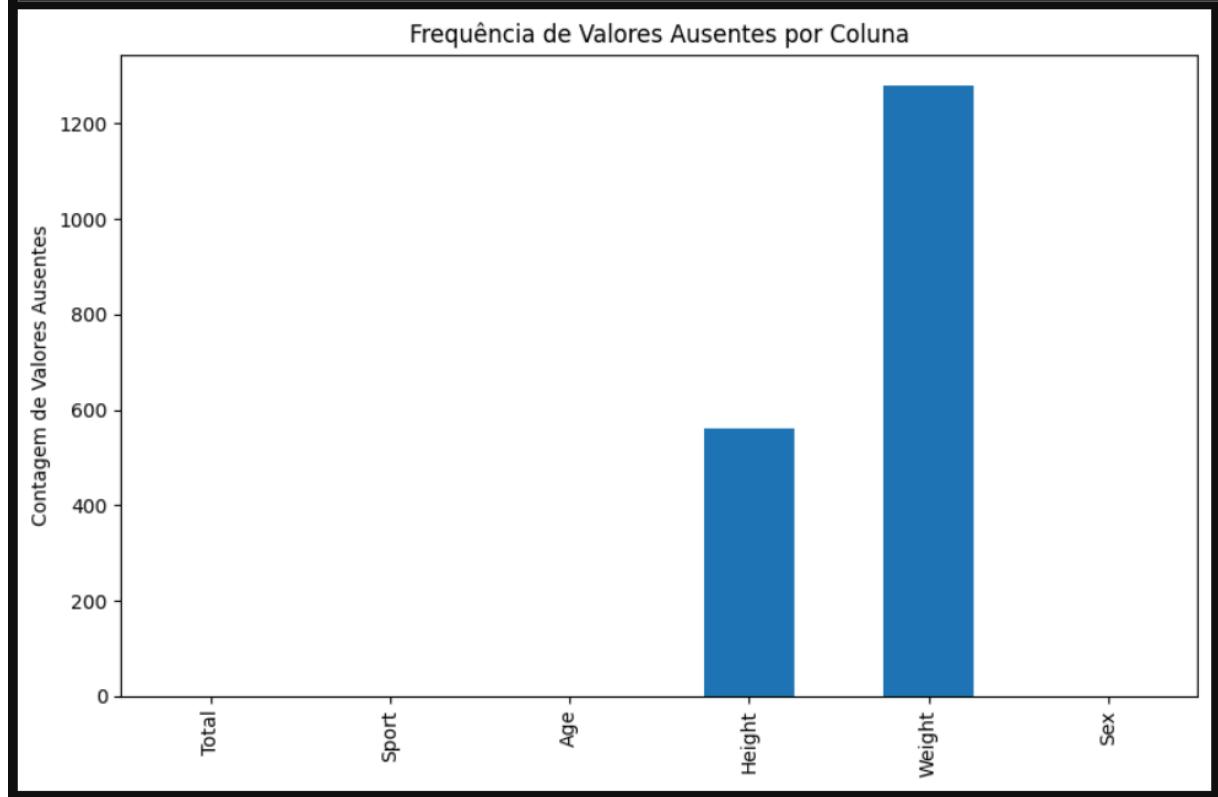
```

De forma gráfica, podemos visualizar utilizando o um gráfico de barras simples:

```

missing_values.plot(kind='bar', figsize=(10, 6), title='Frequência de Valores Ausentes por Coluna')
plt.ylabel('Contagem de Valores Ausentes')
plt.show()

```



Algumas das técnicas para lidar com valores nulos são:

Como o número de linhas

```
df_oli['Height'].fillna(df_oli['Height'].mean(), inplace=True)
```

```
df_oli['Weight'].fillna(df_oli['Weight'].mean(), inplace=True)
```

MÓDULO 3: BUSINESS INTELLIGENCE

- Conceitos e fundamentos de Business Intelligence (BI).
- Sistemas e ferramentas de BI (Dashboards, OLAP).
- Processo de Business Intelligence: coleta, transformação e análise de dados para geração de relatórios e painéis gerenciais.
- Mineração de Dados (Data Mining) introdutória (técnicas de agregação e associação sem uso de algoritmos de Machine Learning).
- Técnicas de Business Analytics para suporte à decisão (análises financeiras, de marketing, etc.).

CONCEITOS BÁSICOS DE BUSINESS INTELLIGENCE

Estamos em um mundo em constante **transformação**, principalmente na área de negócios. É mais do que nunca necessário que as empresas estejam capacitadas em ferramentas e técnicas atualizadas, como o **Business Intelligence (BI)**, para que a **tomada de decisão** seja realizada de forma mais **ágil e inteligente** possível.

BI está associado com a utilização de dados para tomar melhores decisões. Isso não necessariamente está restrito aos negócios, mas em nossa vida familiar nós podemos também tomar melhores decisões. Como exemplo, se formos realizar a reforma de um quarto, podemos ter diversos orçamentos de diferentes fornecedores. Os valores envolvidos dos itens são dados que nos permitem tomar decisões de forma embasada, sobre qual empresa escolher. Alguns produtos podem até ser pesquisados online, o que traz mais dados para suportar a nossa decisão.

CONCEITOS-CHAVE DE BUSINESS INTELLIGENCE (BI)

No contexto organizacional, BI envolve tomar melhores decisões de negócio. As organizações se preocupam em como tornar seus negócios mais eficazes, eficientes e lucrativos.

Algumas perguntas que os gestores se fazem constantemente são:

- Como a empresa pode atrair novos clientes ?
- Como a empresa pode reter mais clientes ?
- Quem são os concorrentes e como eles se comparam ?
- O que impulsiona o lucro ?
- Quais custos e despesas podem ser reduzidos?

Para auxiliar nas respostas para essas perguntas e criar decisões estratégicas e operacionais efetivas, é necessário que tenhamos dados associados e estudados com técnicas e ferramentas de BI.

Os conceitos-chave de BI ajudam a responder a todas as perguntas acima, e podem ser agrupados em cinco diferentes áreas:

1. Domínio

2. Dados

3. Modelo

4. Análise

5. Visualização

Domínio

É o contexto no qual BI é aplicado.

A maioria dos negócios possuem funções e áreas ou departamentos, tais como:

- Vendas
- Marketing
- Manufatura / produção
- Logística
- Pesquisa e Desenvolvimento
- Compras
- Recursos Humanos
- Contabilidade
- Financeiro

Cada uma dessas áreas representa um domínio no qual o BI pode ser utilizado para auxiliar nas respostas para diversas perguntas e ajudar em uma melhor tomada de decisão.

Ao especificar um domínio, estreitamos o **FOCO** visando ser mais específicos com as perguntas e respostas, o que melhora ainda mais o planejamento.

Por exemplo, na área de **vendas**, a gestão pode querer saber quais **vendedores** têm **melhores e piores resultados**. BI pode prover esses insights, assim como ajudar a determinar **quais atividades** permitem a certos profissionais superarem outros. Essa informação pode ser utilizada para treinar e fazer mentorias para os profissionais que estão tendo resultados menos atrativos.

Na área de **marketing**, podemos usar BI para determinar quais campanhas de marketing, tais como e-mail, rádio, folhetos, televisão e internet são mais efetivas para atrair novos clientes. Consequentemente isso mostra onde o orçamento de marketing pode ser mais concentrado.

Na **produção**, podemos utilizar o BI para determinar o *mean time between failure* (MTBF) dos equipamentos utilizados na fabricação dos produtos. Essa informação pode ser utilizada para definir com que frequência podem ou devem ser realizadas manutenções preventivas.

Dados

Uma vez que o domínio seja definido, o próximo passo é identificar e adquirir os dados que estão relacionados a este domínio. Isso quer dizer **identificar as fontes de dados relevantes**, que podem ser **internas** ou **externas** à organização e podem ser **estruturadas**, **não estruturadas** ou **semiestruturadas**.

Dados internos são gerados na organização, em seus processos de negócio e operações. Podem ser gerados enormes volumes de dados sob a forma de receita líquida, vendas a clientes, aquisição de novos clientes, volume de negócios, unidades produzidas, custos de matérias-primas e diversos outros, como séries temporais ou informações transacionais. Os dados atuais e históricos são muito valiosos para as empresas, pois através deles é possível identificar padrões e tendências, assim como realizar previsões e planejamentos de ações futuras.

Organizações possuem múltiplas fontes de dados relevantes.

A combinação dos dados que a organização já detém (dados internos) com **dados externos** torna o business intelligence mais efetivo.

Dados externos são dados gerados fora da organização, tais como performance global das empresas concorrentes, informações censitárias, preços dos produtos concorrentes.

Cada domínio e questão terá dados internos e externos relevantes para a resposta à questão formulada.

Mesmo quando escolhemos como domínio um setor, como por exemplo, **Produção**, é muito provável que teremos fontes de dados relevantes em outros domínios, tais como **Vendas e Marketing**.

DADOS ESTRUTURADOS, NÃO ESTRUTURADOS E SEMIESTRUTURADOS

Dados estruturados são aqueles com uma especificação formal, contendo linhas e colunas com colunas bem definidas de determinados tipos de dados.

Como exemplo, vamos imaginar uma planilha com as colunas `venda_id`, `cliente`, `quantidade comprada` e `preço unitário`. Cada linha representa uma venda.

Fontes de dados estruturadas são as mais fáceis para que ferramentas de BI consigam consumir e analisar os dados.

Normalmente os dados transacionais de uma organização são armazenados em sistemas de bancos de dados relacionais, tais como SQL Server, Oracle, MySQL, DB2, Informix, Sybase, PostgreSQL e outros.

Dados não estruturados

O conceito é exatamente o oposto ao dos dados estruturados. Os não estruturados não contém nenhuma organização lógica que permita a organização em tabelas com linhas e colunas. Tais dados envolvem imagens, vídeos, áudio e texto, como arquivos do word, e-mails, publicações em redes sociais e páginas web.

Estes dados são os mais complexos para que ferramentas de BI consigam consumir e analisar. Normalmente, estes tipos de dados são armazenados em tipos como *Binary Large Objects* (BLOBS) ou sistemas de arquivo como *New Technology File System* (NTFS) ou Hadoop *Distributed File System* (HDFS).

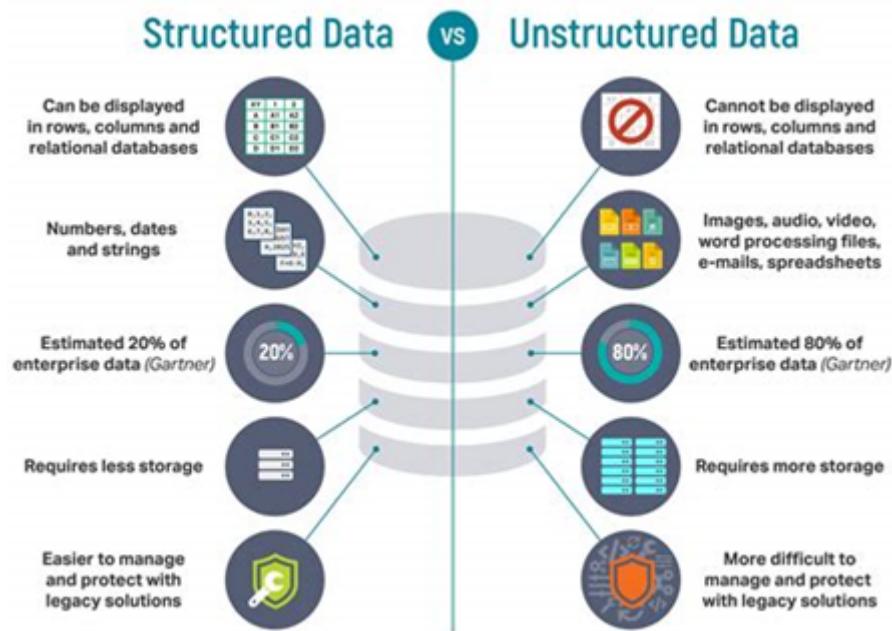
Dados não estruturados também são armazenados em sistemas de bancos de dados NoSQL, como: MongoDB, Microsoft Azure Cosmos DB, Neo4j, Redis, Cassandra, HBase e outros.

Dados semiestruturados

São dados que não representam a estrutura formal de dados estruturados, tais como tabelas com linhas e colunas, mas contém uma estrutura definida.

Exemplos destes dados são arquivos textos delimitados, XML, HTML, JSON e outros. Também podem ser incluídos nessa categoria protocolos de acesso a dados como *Open Data Protocol* (OData) e *Representational State Transfer* (REST) APIs.

Estes tipos de dados são utilizados por fontes de dados como Microsoft Sharepoint, Exchange, Microsoft Active Directory, redes sociais como Twitter e Facebook, outras plataformas online como Google Analytics, Github e vários outros.



Modelo

Um modelo de dados é a forma na qual um ou mais fontes de dados são organizadas visando permitir análises e visualizações.

Modelos são criados através da limpeza e transformação dos dados, auxiliando na definição dos tipos de dados contidos nas fontes de dados, assim como na definição de categorias de dados para tipos específicos.

Organização dos modelos

A organização pode ser extremamente simples, envolvendo somente uma única tabela com colunas e linhas, mas no caso de Business Intelligence, normalmente envolve múltiplas tabelas de dados provenientes de diferentes fontes de dados.

Quando mais fontes de dados e tabelas precisam ser combinadas de forma coerente, mais complexo se torna o modelo.

Transformação e Limpeza

Ao criar um modelo de dados, é frequentemente necessário limpar e transformar os dados de origem. Partimos da premissa de que **os dados nunca estão limpos, no primeiro momento**. Sempre há necessidade de ajustar dados ou descartar dados ruins. Um exemplo disso é a existência de **dados duplicados de clientes** com nomes digitados de formas diferentes (alguma letra por erro de entrada). Alguns outros problemas são: **dados ausentes, formatações inconsistentes**, e até mesmo algo simples como espaços em locais inadequados.

Todas essas situações causam problemas ao realizar análises com ferramentas de BI.

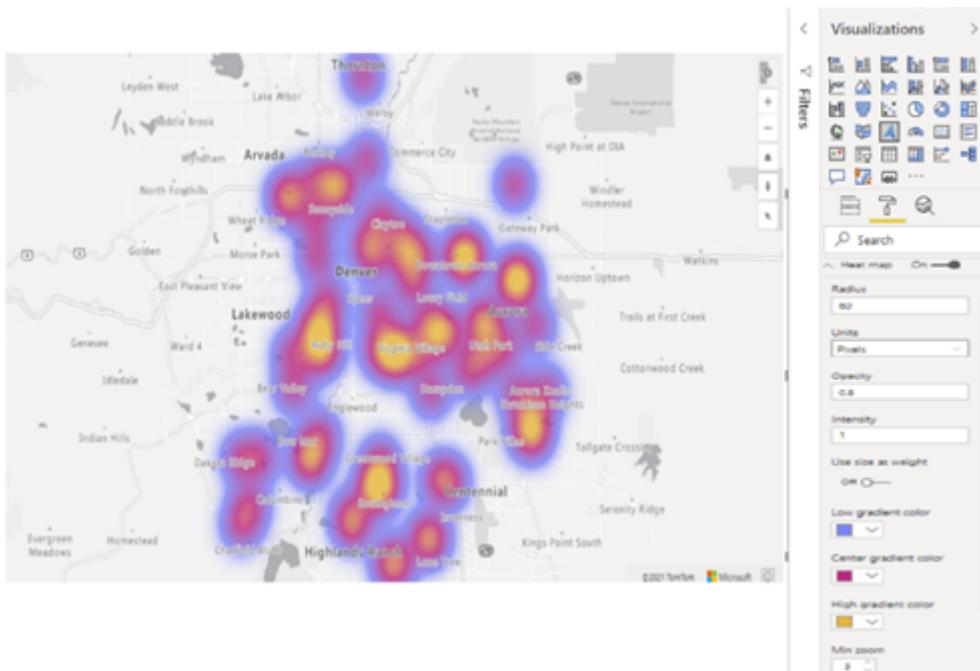
Técnicas de transformação e limpeza são normalmente referenciadas como extract, transform and load (**ETL**), ferramentas existentes em diversos produtos como Microsoft SQL Server Integration Services (SSIS), Azure Data Factory, Oracle Data Integrator, Pentaho Data Integration, Sybase ETL e muitos outros.

Definindo e categorizando tipos de dados

Modelos de dados também definem de maneira formal os tipos de dados de cada tabela. Os tipos geralmente incluem formatos como texto, número decimal, número inteiro, porcentagem, data, hora, data e hora, duração, verdadeiro/falso e binário.

É muito importante definir de forma correta os dados que estão sendo selecionados, pois isso é crucial para o tipo de análise que poderá ser feita posteriormente. Como exemplo, não tem sentido somar ou fazer média de dados textuais.

Outra questão é que dados possuem categorias, por exemplo, mesmo sendo numérico, um tipo de dado pode representar um CEP. Com isso, é possível determinar que essa categoria de dado numérico pode ser utilizada, por exemplo, para ser plotada em um mapa.



Análise

Uma vez que as etapas anteriores: seleção do domínio e combinação de fontes de dados em um modelo, o próximo passo é realizar a análise dos dados.

A análise é um processo chave do BI, e é nela que tentamos responder questões relevantes ao negócio, utilizando os dados internos e externos.

Ter simplesmente dados sobre vendas não é totalmente útil para um negócio. Para fazer a previsão de receitas de vendas em um ponto no futuro, é importante que os dados sejam agregados e analisados de algumas formas.

Por exemplo, pode ser considerada **a venda média de um produto, a frequência de compras e quais clientes compram com mais frequência**. Essas são informações que permitem uma melhor tomada de decisão.

Análises de dados podem ter diversas formas, tais como a agregação de dados, como somas, contagens, médias, assim como a criação de cálculos mais complexos, identificação de tendências, correlações e previsão.

Muitas vezes, a saúde ou performance dos negócios são monitoradas através de indicadores como **key performance indicators (KPIs)**. Podem incluir questões como taxa de retenção de funcionários, novas aquisições de clientes por mês, margem bruta, lucro antes de juros, impostos, depreciação e amortização.

As agregações e cálculos são chamadas de **métricas** ou **medidas**, e são utilizadas para identificar tendências ou padrões que são utilizadas na tomada de decisão das organizações.

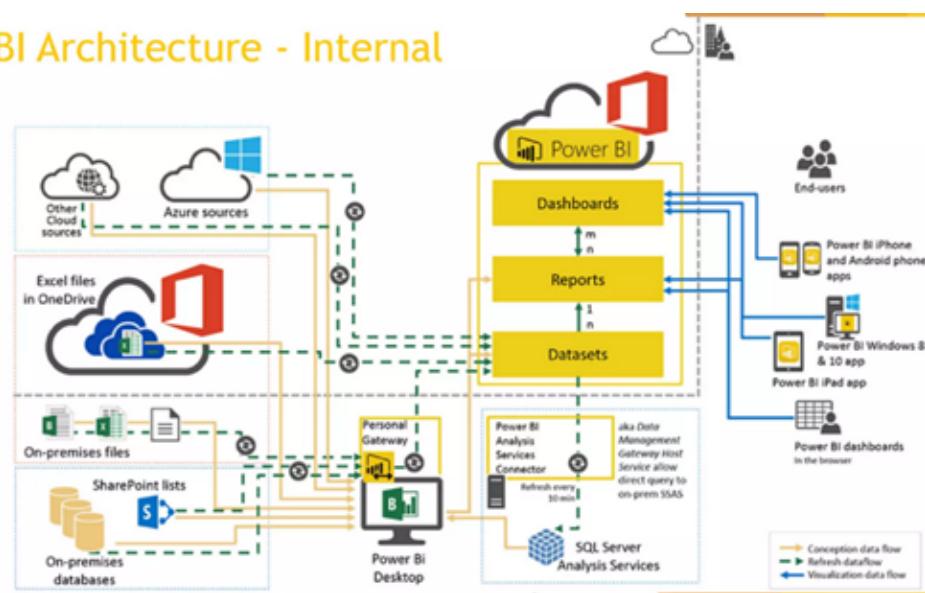
VISUALIZAÇÕES

O último conceito de BI é a visualização, ou a apresentação real das análises que estão sendo realizadas.

Ao demonstrar as análises realizadas através de gráficos, relatórios e dashboards, as pessoas conseguem visualizar melhor e de forma mais intuitiva os resultados.

Visualizações permitem ao analista ou autor do relatório contar uma história através dos dados. Essa história deve responder às questões que foram originalmente propostas ao definir o domínio, e portanto, entregar os *insights* que permitam à organização tomar melhores decisões.

Power BI Architecture - Internal



O QUE É o POWER BI ?

Considerando sua maior abrangência e quantidade de funcionalidades, é uma tecnologia baseada em nuvem para relatórios e análise de dados.

Foi elaborado de uma forma que não é somente útil para desenvolvedores, mas também para usuários treinados e analistas de negócios.

Power BI utiliza um ambiente simples e fácil de utilizar, amigável para elaborar relatórios. Apesar das facilidades, é baseado em diversos componentes poderosos que podem ser utilizados para gerar relatórios tanto para soluções simples quanto para cenários bem complexos.

Cada componente é responsável por uma parte específica de tecnologias, tais como:

- Realizar conexão com fontes de dados
- Executar cálculos analíticos
- Criação de relatórios
- Compartilhamento de relatórios

Algumas boas práticas recomendadas ao utilizar o Power BI de uma forma mais eficaz:

1. Preparação de dados: Fazer a limpeza e preparar os dados antes de importar para o Power BI, para garantir que as visualizações realmente tenham o significado adequado e preciso.
2. Modelagem: Criar um modelo de dados bem projetado que suporte as necessidades dos relatórios e dashboards.
3. Visualizações: Escolher a visualização correta para demonstrar de forma efetiva os *insights* obtidos com dados utilizados.
4. Interatividade: Utilizar filtros, buscas hierárquicas e outras funcionalidades interativas para permitir que os usuários explorem e analisem os dados de forma mais flexível e por conta própria.
5. Dashboards: Criar dashboards interativos que disponibilizem uma única visão de indicadores e métricas.
6. Colaboração: Compartilhar relatórios e dashboards com outros na organização utilizando as funcionalidades colaborativas, tais como workspaces e comentários.
7. Otimização de performance: Monitorar com regularidade e otimizar a performance dos relatórios e dashboards, para garantir uma melhor experiência dos usuários.

A fase de **preparação de dados** é um passo crucial para utilizar o Power BI de forma mais efetiva. Nessa fase, é necessário atenção nos seguintes detalhes:

1. Fontes de dados (Data Sources): Identificar as fontes de dados que precisam ser importadas com o Power BI e garantir que estejam em um

formato que seja compatível (exemplo: CSV, Excel, SQL Server, XML, JSON e muitos outros). Fontes de dados são os dados brutos que serão importados, e podem ser originados a partir de várias fontes, tais como bancos de dados, planilhas, sistemas baseados em nuvem ou APIs.

2. Limpeza de dados (Data Cleaning): Limpar os dados removendo qualquer parte de dados irrelevante, duplicada ou inconsistente, e realizando transformações em formatos que tornem mais fácil o trabalho para geração de informações gerenciais. Pode também envolver a remoção de linhas ou colunas vazias, corrigir erros de entrada de dados e padronizar determinados dados em um formato a ser utilizado. Essa fase também é muito importante, pois irá garantir que as informações visualizadas serão geradas a partir de dados com a acurácia e significado necessários.
3. Normalização de dados: Normalizar os dados garante que eles terão consistência e que serão eliminadas as redundâncias. Isso pode envolver a combinação de dados de fontes múltiplas ou transformá-los em um formato padrão, como por exemplo uma coluna com o formato data.
4. Transformação de dados: Transformar (ou converter) os dados em um formato que seja adequado para analisar e criar relatórios, tais como transformar uma tabela em uma tabela pivot (pivot table), ou gerando novas colunas a partir de cálculos.
5. Integração de dados: Integrar dados de diferentes fontes em um único conjunto de dados no PowerBI. Pode incluir a realização do merge de tabelas ou criar relacionamentos entre tabelas.
6. Validação de dados: Validar os dados para garantir que são precisos e completos. Pode incluir a verificação da existência de dados ausentes ou inconsistentes.
7. Qualidade de dados: Aprimorar a qualidade dos dados através da identificação e correção de quaisquer erros ou imprecisões. Pode incluir a necessidade de utilizar técnicas de limpeza de dados, tais como preenchimento ou padronização de dados. A fase de **modelagem de dados** é o processo de criação de um modelo de dados bem projetado que suporte as necessidades dos relatórios e painéis

As etapas envolvidas na fase de modelagem são:

1. Estrutura de dados: determinação da melhor estrutura para os dados existentes, com base nos tipos de análises e visualizações planejadas. Isso pode envolver a criação de tabelas, colunas e relacionamentos no PowerBI.
2. Tipos de dados: atribuição do tipo de dados correto a cada coluna em seu modelo de dados. Isso garantirá que o PowerBI possa analisar e visualizar

os dados com precisão. Alguns tipos de dados comuns incluem texto, número, data/hora e moeda.

3. Relacionamentos: definição de relacionamentos entre tabelas no modelo de dados. Com isso, o PowerBI conseguirá juntar dados de diferentes tabelas e criar visualizações mais complexas. Os relacionamentos podem ser estabelecidos com base em uma coluna comum entre as tabelas, como um relacionamento de chave primária/chave estrangeira.
4. Medidas: criação de medidas para agrupar e resumir dados em seu modelo. Medidas são valores calculados que podem ser usados em visualizações e relatórios. Alguns exemplos de medidas incluem cálculos como: soma, contagem, média e máximo.
5. Colunas calculadas: definição de colunas calculadas para derivar novos dados de colunas existentes em seu modelo de dados. As colunas calculadas podem ser usadas em visualizações e relatórios. Por exemplo, você pode criar uma coluna calculada para calcular a diferença entre duas colunas em seus dados, ou o lucro (receitas – despesas) envolvidos em determinado processo.
6. Fórmulas DAX: Utilização de DAX (Data Analysis eXpressions - expressões de análise de dados) para executar cálculos e manipulações mais avançados em seus dados. As fórmulas DAX podem ser usadas para criar medidas, colunas calculadas e outras transformações de dados.
7. Atualização de dados: configuração de agendamentos de atualização de dados para garantir que o modelo de dados do PowerBI esteja sempre atualizado. Isso garantirá também que as visualizações e relatórios sejam baseados nos dados mais atuais.

Projetos com o powerbi consistem de 3 fases:

1. Data discovery

- a. Data Extraction
- b. Transform

Conectar com uma ou mais fontes (datasources) para obter todos os dados necessários.

Realizar a limpeza dos dados, eliminando valores indesejados ou inapropriados contidos nos campos (Power Query – criar regras de negócios).

2. Data Modeling / mashup

Criar relacionamentos entre as fontes de dados existentes. Por exemplo, para obter o lucro de uma empresa, é possível que seja necessário utilizar 4 ou 5 diferentes fontes de dados para obter despesas, vendas, salários, etc. Tais fontes de dados podem existir em diferentes origens, como bancos de dados (Oracle, Sql Server), planilhas, arquivos de texto, etc e podem ser relacionados.

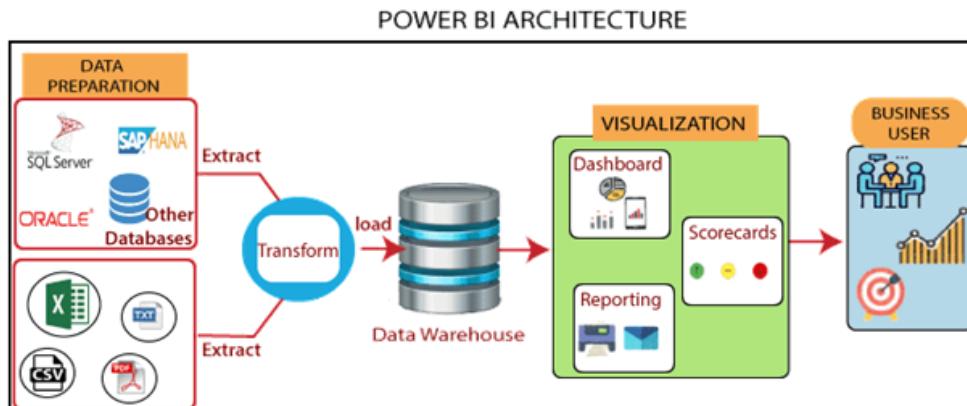
Utilização da DAX (Data Analysis eXpression language) para realizar cálculos e criar expressões para gerar dados finais a serem mostrados.

3. Data Visualization

Criar relatórios, dashboards e filtros para mostrar de forma visual e dinâmica os dados finais a serem apresentados aos usuários finais.

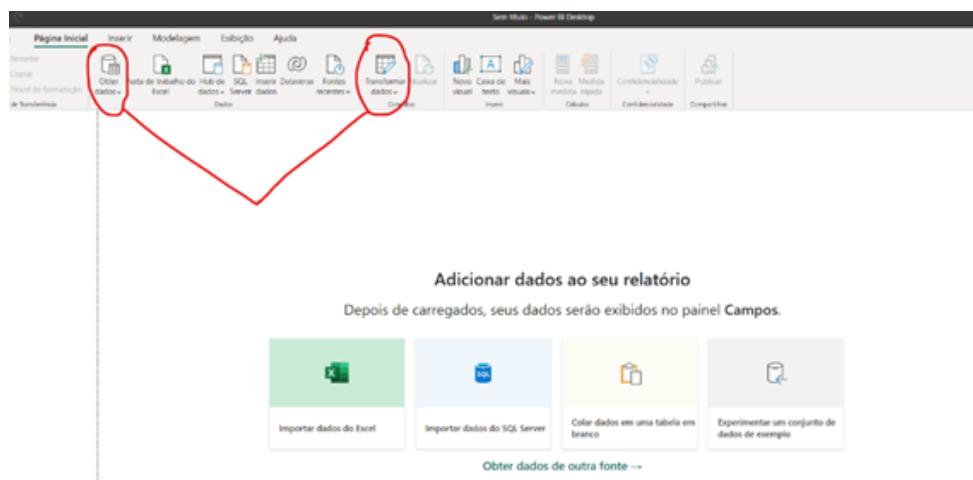
Para fazer o download do Power Bi Desktop (necessário criar uma conta):

https://go.microsoft.com/fwlink/?LinkId=874445&pbi_source=websignup_uNav

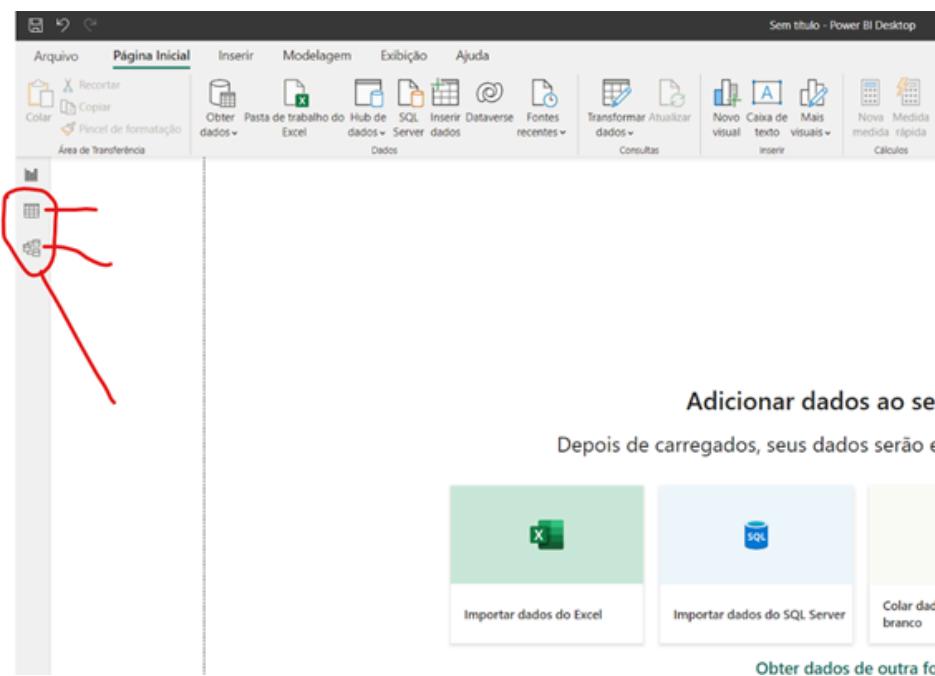


HANDS-ON - EXPLORAÇÃO DA INTERFACE GRÁFICA DA FERRAMENTA POWER BI

Fase I – Obtenção, limpeza e transformação de dados:



FASE 02 – MODELAGEM E VISUALIZAÇÃO DOS MODELOS E DADOS ASSOCIADOS:



FASE 03: VISUALIZAÇÃO DE RELATÓRIOS E DASHBOARDS:

The screenshot shows the Microsoft Power BI Desktop interface. The ribbon at the top has tabs for Arquivo, Página Inicial (selected), Inserir, Modelagem, Exibição, and Ajuda. The Página Inicial tab has several sections: Colar, Recortar, Copiar, Pincel de formatação, Área de Transferência, Dados (with sub-options like Obtener datos, Pasta de trabalho do Excel, Hub de datos, SQL Server dados, Inserir Dataverse, Fontes recentes, Transformar dados, Atualizar dados, Consultas), and Visualizações (with sub-options like Novo visual, Caixa de texto, Mais visuais, Inserir, Nova medida rápida, Círculos). A red circle and arrow highlight the 'Obter datos' icon in the Dados section.

Adicionar dados ao seu relatório
Depois de carregados, seus dados serão exibidos no painel Campos.

Importar dados do Excel Importar dados do SQL Server Colar dados brancos

Obter dados de outra fonte

FASE 04: PUBLICAÇÃO / COMPARTILHAMENTO DE RELATÓRIOS E DASHBOARDS

The screenshot shows the Microsoft Power BI Desktop interface with the ribbon. The Página Inicial tab is selected. The ribbon includes sections for Arquivo, Página Inicial (selected), Inserir, Modelagem, Exibição, and Ajuda. The Página Inicial tab has sections for Colar, Recortar, Copiar, Pincel de formatação, Área de Transferência, Dados (with sub-options like Obtener datos, Pasta de trabajo do Excel, Hub de datos, SQL Server dados, Inserir Dataverse, Fontes recentes, Transformar dados, Atualizar dados, Consultas), and Visualizações (with sub-options like Novo visual, Caixa de texto, Mais visuais, Inserir, Nova medida rápida, Círculos, Confidencialidade, Publicar, Compartilhar). A red circle highlights the 'Publicar' icon in the Visualizações section.

Adicionar dados ao seu relatório
Depois de carregados, seus dados serão exibidos no painel Campos.

XLSX SQL Server PDF PNG

ESTRUTURA GERAL DA INTERFACE DE VISUALIZAÇÃO DO POWER BI:

Visões: Ao lado esquerdo, cada botão permite visualizar os relatórios, os dados e o modelo.

Painéis: Ao lado direito os painéis são os de filtro, visualizações e campos.

Guias: Nos botões na parte de cima, permitem diversas interações mais específicas.

Prática 01: Nesta prática, será feita a conexão do Power BI com uma fonte de dados pública referente a bancos falidos.

Será acessado um dataset diretamente de um site na web – um conjunto de dados de instituições bancárias falidas, com a finalidade de entender os melhores locais para construir novas agências em determinados estados dos Estados Unidos.

Para isso, é necessário abrir no navegador o endereço: data.gov, que contém uma grande variedade de conjuntos de dados de diversas finalidades.

Na lista de busca, digitar “fdic failed banks”

The screenshot shows the 'FDIC Failed Bank List' dataset page. At the top, it says 'Metadata Updated: November 12, 2020'. Below that, a note states: 'The FDIC is often appointed as receiver for failed banks. This list includes banks which have failed since October 1, 2000.' Under the heading 'Access & Use Information', there are two items: 'Public: This dataset is intended for public access and use.' and 'License: No license information was provided. If this work was prepared by an officer or employee of the United States government as part of that person's official duties it is considered a U.S. Government Work.' Below this, under 'Downloads & Resources', there are two options: 'Comma Separated Values File' (banklist.csv) with 435 views and a 'Download' button, and 'Web Page' (index.html) with 48 views and a 'Visit page' button.

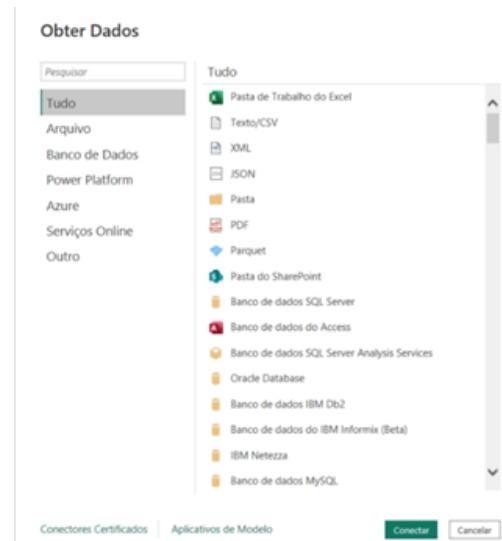
Apesar da possibilidade de fazer o download do arquivo CSV, será feito o vínculo da fonte de dados a partir do endereço web. Dessa forma, toda vez que houver uma atualização no conteúdo do arquivo, o dashboard criado será automaticamente atualizado, e dessa forma, garantindo que sempre será gerada a informação de forma mais atualizada.

The screenshot shows a dataset page titled "FDIC Failed Bank List". The page includes metadata updated on November 12, 2020, and a note about the FDIC being appointed as receiver for failed banks. It features sections for "Access & Use Information" and "Downloads & Resources", where a CSV file named "banklist.csv" is listed with 436 views. A context menu is open over the page, with the option "Copiar link" highlighted by a red arrow.

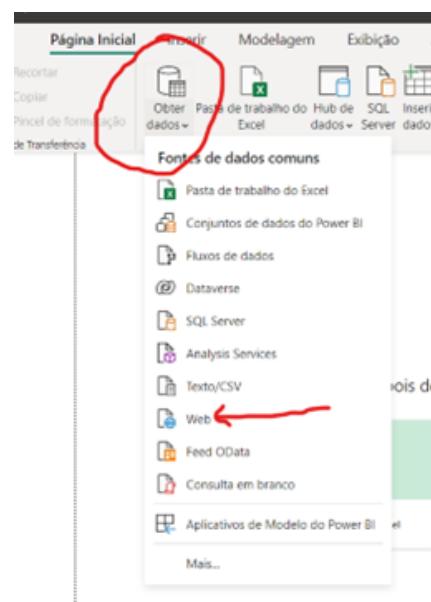
Acessando na guia “Página Inicial”, será selecionada a opção “Obter dados”, e depois a opção “Mais”:

The screenshot shows the Microsoft Power BI ribbon with the "Página Inicial" tab selected. The "Obter dados" button is highlighted. A dropdown menu is open under "Obter dados", showing various data source options like Excel, Power BI Data Model, and others. The "Mais..." option at the bottom of the list is circled with a red box. To the right, there's a preview area for "Adicionar dados" (Add data) showing "Excel" and "Importar dados do SQL Server" (Import data from SQL Server).

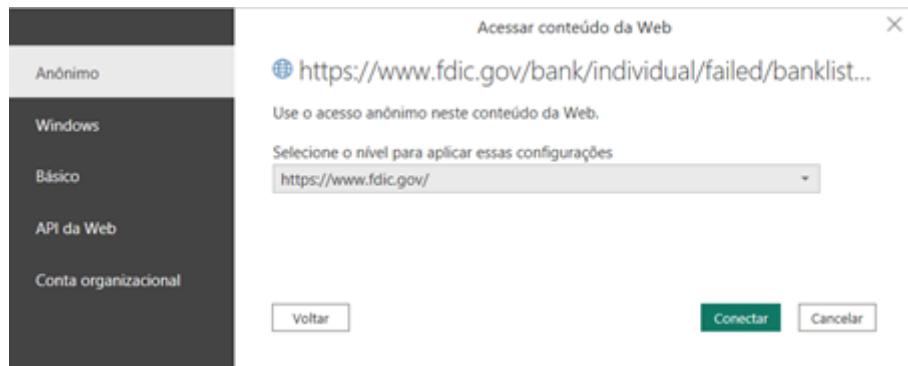
As opções indicam a grande quantidade de opções de fontes de dados que podem ser utilizadas no Power Bi (Arquivo, Banco de Dados, Power Platform, Azure, Serviços Online e Outros).



Para fazer a conexão com a fonte de dados do link de internet com o arquivo que contém a lista dos bancos, é necessário clicar com o botão direito sobre o botão “download” e copiar o endereço do link:



Na primeira vez que é realizada a conexão com a fonte de dados, é mostrada uma tela de conexão para uma possível autenticação. Isso será necessário no caso de utilizar algum sistema ou link que não seja público, e seja necessário informar as credenciais de acesso autorizado.



Como o site que será utilizado fornece datasets de forma pública, não será necessário usar nenhuma credencial / autenticação para acessar os dados. Basta clicar no botão “conectar” para acessar os dados de forma anônima, o que é permitido pelo site.

Bank Name	City	State	Cert	Acquiring Institution	Closing Date	Fund
Almena State Bank	Almena	KS	25426	Equity Bank	23/10/2020	205
First City Bank of Florida	Fort Walton Beach	FL	26748	United Fidelity Bank, fsb	16/10/2020	205
The First State Bank	Barbourville	WV	14362	MVB Bank, Inc.	03/04/2020	205
Ericson State Bank	Ericson	NE	28265	Farmers and Merchants Bank	14/02/2020	205
City National Bank of New Jersey	Newark	NJ	21111	Industrial Bank	01/11/2019	205
Resolute Bank	Maumee	OH	58317	Buckeye State Bank	25/10/2019	205
Louisia Community Bank	Louisia	KY	58112	Kentucky Farmers Bank Corporation	25/10/2019	205
The Enloe State Bank	Cooper	TX	20736	Legend Bank, N. A.	31/05/2019	205
Washington Federal Bank for Savings	Chicago	IL	30570	Royal Savings Bank	15/12/2017	205
The Farmers and Merchants State Bank of Argonia	Argonia	KS	17729	Conway Bank	13/10/2017	205
Fayette County Bank	Saint Elmo	IL	1802	United Fidelity Bank, fsb	26/05/2017	205
Guaranty Bank, (d/b/a Bestbank in Georgia & Michigan)	Milwaukee	WI	30003	First-Citizens Bank & Trust Company	05/05/2017	205
First NBC Bank	New Orleans	LA	58302	Whitney Bank	28/04/2017	205
Profilco Bank	Cottonwood Heights	UT	35493	Cache Valley Bank	03/03/2017	205
Seaway Bank and Trust Company	Chicago	IL	29328	State Bank of Texas	27/03/2017	205
Harvest Community Bank	Pennsville	NJ	34951	First-Citizens Bank & Trust Company	13/01/2017	205
Allied Bank	Mulberry	AR	92	Today's Bank	23/08/2016	205
The Woodbury Banking Company	Woodbury	GA	11297	United Bank	19/08/2016	205
First CornerStone Bank	King of Prussia	PA	35312	First Citizens Bank & Trust Company	06/05/2016	205
Trust Company Bank	Memphis	TN	9956	The Bank of Fayette County	28/04/2016	205

O botão “Carregar” realiza a carga de todos os dados do link / arquivo vinculados com a conexão para obter os dados e cria um novo modelo.

Conexão com a fonte de dados em:

<https://catalog.data.gov/dataset/fdic-failed-bank-list>

<https://www.fdic.gov/bank/individual/failed/banklist.csv>

Em alguns sites, por questões de segurança, é exigida a autenticação dos usuários (login e senha) para obter os dados. Esse site que estamos utilizando permite acesso anônimo, então basta avançar na primeira tela.

Vamos utilizar o botão “Transformar Dados”, que vai iniciar uma outra janela, com o Power Query e permitir que sejam feitos todos os ajustes (limpezas e transformações) necessários nos dados a serem utilizados.

Algumas vezes, quando é possível realizar todos os pré-processamento nos dados previamente, não é necessário abrir o botão “**Transformar dados**”, e com isso, basta carregar os dados para iniciar o uso.

A situação mais usual é que quase sempre tenhamos alguma coisa para ajustar, como limpar, padronizar e transformar nos dados, ou mesmo gerar campos novos, como vamos ver em alguns dos exemplos nas etapas a seguir.

<https://www.fdic.gov/bank/individual/failed/banklist.csv>

The screenshot shows the 'Data Load Preview' window in Power BI. At the top, there are three tabs: 'Origem do Arquivo' (File Source), 'Delimitador' (Delimiter), and 'Detecção de Tipo de Dados' (Type Detection). The 'Delimitador' tab is selected, showing '1252: Europeu Ocidental (Windows)' as the encoding and 'Vírgula' (Comma) as the delimiter. The 'Detecção de Tipo de Dados' tab has 'Com base em um conjunto de dad...' (Based on a sample) selected. Below these tabs is a preview of a table with columns: Name, City, State, Cert, Acquiring Institution, Closing Date, and Fund. The table contains several rows of data, mostly from Illinois (IL). At the bottom of the preview, there are three buttons: 'Extrair a Tabela Usando Exemplos' (Extract Table Using Examples), 'Carregar' (Load), and 'Transformar Dados' (Transform Data). The 'Transformar Dados' button is circled in red.

O botão de transformar inicia o editor do power query em uma nova janela, que permite fazer qualquer tipo de limpeza ou transformação de dados necessárias.

The screenshot shows the Power Query Editor interface with a table titled "Bancos Falidos". The table has the following columns and data:

	Bank Name	City	State	County	Acquiring Institution	Closing Date	Hand
1	Almeria State Bank	Almeria	KS		25429_Equity Bank	28/3/2019	
2	First City Bank of Florida	Fort Walton Beach	FL		26748_United Purity Bank, Inc.	19/10/2019	
3	The First State Bank	Bethanyville	WV		24362_MVVB Bank, Inc.	03/04/2019	
4	Ericson State Bank	Ericson	NE		28003_Farmers and Merchants Bank	24/02/2019	
5	City National Bank of New Jersey	Newark	NJ		21112_Industrial Bank	01/11/2019	
6	Resolute Bank	Massee	OH		28117_Buckeye State Bank	29/10/2019	
7	Louisa Community Bank	Louisa	KY		28122_Kentucky Farmers Bank Corporation	25/10/2019	
8	The Enco State Bank	Cooper	TX		30716_Legend Bank, N.A.	30/09/2019	
9	Washington Federal Bank for Savings	Chicago	IL		30919_Wisegel Savings Bank	19/10/2019	
10	The Farmers and Merchants State Bank of Arizona	Argonne	KS		37739_Currency Bank	13/03/2019	
11	Payette County Bank	Saint Elmo	IL		1802_United Purity Bank, Inc.	26/05/2019	
12	Grocery Bank, McWayne (Georgia & Michigan)	Milwaukee	WI		30003_First-Citizens Bank & Trust Company	05/09/2019	
13	First NBC Bank	New Orleans	LA		28102_Whitney Bank	28/04/2019	
14	Proactive Bank	Cottonwood Heights	UT		87485_Calico Valley Bank	05/05/2019	
15	Seaway Bank and Trust Company	Chicago	IL		28108_State Bank of Texas	25/05/2019	
16	Harvest Community Bank	Pensacola	FL		24932_First Citizens Bank & Trust Company	22/04/2019	
17	Altad Bank	Butteberry	AR		90_Today's Bank	23/09/2019	
18	The Woodbury Banking Company	Woodbury	CA		21107_United Bank	19/05/2019	
19	First CornerStone Bank	King of Prussia	PA		28112_First-Citizens Bank & Trust Company	09/05/2019	

Caso essa janela do Power Query seja fechada por acidente, é possível reabri-la escolhendo o botão “Transformar Dado” da guia “Página Inicial”. Tudo que for feito e salvo no Power Query é mantido no mesmo arquivo do Power BI, com formato “.pbix”.

Uma boa prática a ser feita é renomear a consulta Power Query com um nome mais apropriado, pois é possível que estejamos trabalhando com várias sessões simultâneas do Power BI ativas, e se não utilizarmos um nome adequado, pode gerar confusão ou até a transformação errada de dados. Vamos salvar como “Bancos Falidos”.

Isso pode ser feito tanto pela área de Consultas (lado esquerdo da tela), como pela Propriedade da Configuração da Consulta (lado direito da tela), como mostra a figura abaixo:

The screenshot shows the Power Query Editor with the configuration pane open on the right side. The "Configurações" section is expanded, showing the "PROPRIEDADES" tab with the "Nome" field set to "Bancos Falidos". A red arrow points to this field. Below it, under "ETAPAS APPLICADAS", there are two entries: "Fonte" and "Cabeçalhos Promovidos".

Caso o painel de “Config. Consulta” seja fechado, é possível abrir novamente pelo menu “Exibição”, opção de “Config. Consulta”, como na figura abaixo:

The screenshot shows the Power Query Editor interface. The ribbon at the top has tabs: Arquivo, Página Inicial, Transformar, Adicionar Coluna, **Exibição**, Ferramentas, and Ajuda. The 'Exibição' tab is highlighted with a red circle. Below the ribbon, there are sections for Layout, Visualização dos Dados, and Colunas. A red arrow points to the 'Config. Consulta' icon in the Layout section. The main area shows a table with one row: 'Bancos Falidos' containing 'Almena State Bank'. The formula bar above the table shows: = Table.TransformColumns(#"Colunas Renomeadas", {{"

Cada coluna existente reflete o tipo de dados identificado de forma automática pelo Power Query no momento da leitura de uma amostragem dos dados existentes. Os campos textuais são indicados com letras “ABC”, campos de data com o ícone de calendário, e campos numéricicos com “123”.

The screenshot shows the Power Query Editor with a table. The columns are labeled: Bank Name, City, State, Text, Acquiring Institution, and Closing Date. Red circles highlight the 'Text' column header and the 'Closing Date' column header. The table data is as follows:

	Bank Name	City	State	Text	Acquiring Institution	Closing Date	Fund
1	Almena State Bank	Almena	KS		25426 Equity Bank	28/10/2020	20538
2	First City Bank of Florida	Fort Walton Beach	FL		26748 United Fidelity Bank, fob	28/10/2020	20537
3	The First State Bank	Barbourville	WV		24362 MVB Bank, Inc.	03/04/2020	20536

Caso seja necessário alterar algum tipo de dados, basta clicar sobre o símbolo referente ao tipo automaticamente identificado, e escolher o tipo de dados adequado, conforme a figura abaixo:

The screenshot shows a dropdown menu for data type selection. The 'Text' option is selected, indicated by a red circle. Other options include Número Decimal, Número decimal fixo, Número Inteiro, Percentual, Data/hora, Data, Hora, Data/Hora/Fuso Horário, Duração, Texto, Verdadeiro/Falso, and Binário. The menu also includes 'Usando a Localidade...' at the bottom. The background shows a table with columns: Cert, Acquiring Institut, and several rows of data.

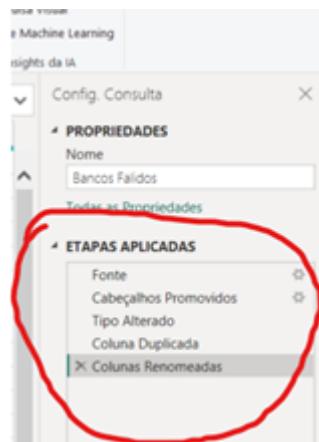
É uma boa prática remover do conjunto de dados todas as colunas que não contém dados utilizáveis. Isso diminui o volume de dados a ser manipulado, consequentemente melhorando a performance das consultas pelo menor uso de memória.

Pode ser necessário alterar os nomes das colunas (campos), para deixar mais claro o significado.

Podemos também duplicar uma determinada coluna para fazer testes de transformações nos dados.

Observe que para cada tipo de dados, o Power BI disponibiliza automaticamente diversos tipos de transformações. Clique em colunas com diferentes tipos (texto, números inteiros, data) e perceba a diferença nas respectivas funções que aparecem.

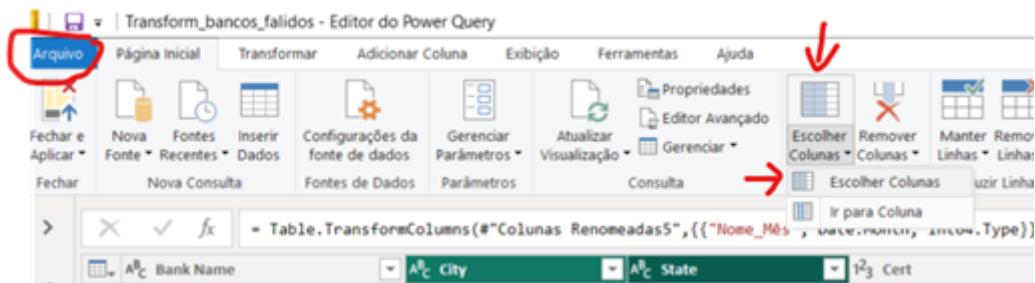
O editor do Power Query mantém um histórico de ações – “Etapas Aplicadas”, na aba “Config. Consulta”, permitindo desfazer caso o resultado da ação não seja o adequado.



HANDS-ON - Atividade prática 1:

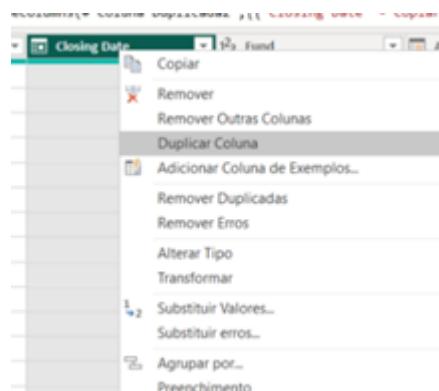
Vamos fazer as seguintes operações iniciais, como primeira atividade prática:

1. Remover a coluna “CERT” utilizando a opção: -> Arquivo -> Escolher Colunas -> Escolher colunas, e desmarcar a referida coluna. Essa é a opção mais otimizada, principalmente para grandes conjuntos de dados com muitas colunas e muitos registros.

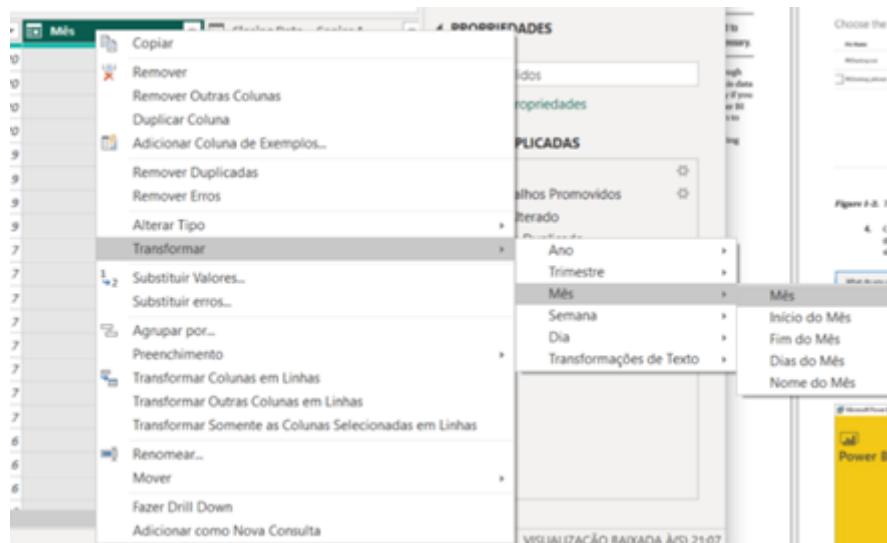


A outra forma de remover colunas é simplesmente clicando com o botão direito sobre o nome da coluna e escolhendo a opção “Remover”. Essa opção é mais lenta do que a de escolher os campos.

2. Duplicação da coluna de data “Closing Date” 4 (quatro) vezes. Sempre antes de fazer alguma transformação nos dados, é aconselhável manter o dado (coluna) original. Use o botão direito do mouse sobre a coluna para exibir as opções:



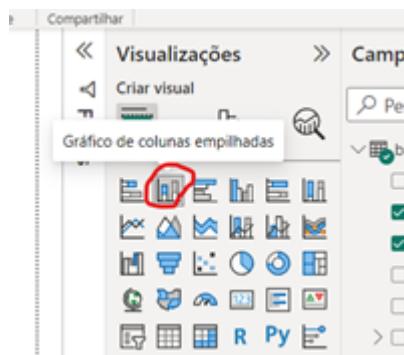
3. Alterar o nome da primeira cópia para “Mês”.
4. Alterar o nome da segunda cópia para “Trimestre”.
5. Alterar o nome da terceira cópia para “Ano”.
6. Alterar o nome da quarta cópia para “Nome_Mês”.
7. Transformar a data da coluna renomeada no passo 2 (Mês), para que fique somente com o número do mês:



8. Transformar a data da coluna renomeada no passo 3 (Trimestre), para que fique somente com o número do Trimestre.
9. Transformar a data da coluna renomeada no passo 4 (Ano), para que fique somente com o número do Ano.
10. Transformar a data da coluna Nome_Mês para o nome do respectivo mês.
11. Transforme o nome de forma que as iniciais fiquem todas maiúsculas.
12. Criar uma nova coluna combinando os valores das colunas “City” e “ST”.
 - a. Isso é útil quando temos cidades com o mesmo nome em estados diferentes, o que acontece em várias situações.
 - i. Selecione as 2 colunas clicando em cada uma com o CTRL pressionado.
 - ii. Abra a guia “Adicionar Coluna”.
 - iii. Selecione a opção “Coluna de Exemplos” e “De Todas as Colunas”
 - iv. Ao ser gerada uma nova coluna, clique e digite em qualquer registro o nome da cidade e do estado, no seguinte formato: Cidade (Estado) e aperte ENTER.
 - v. Verifique que o Power BI utilizou seu exemplo e replicou em todas as demais linhas o formato que você digitou.

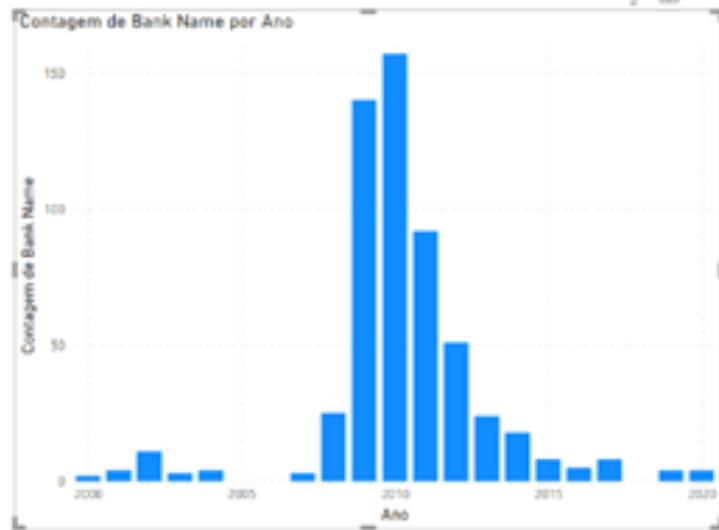
Ao clicar e aplicar, na guia “Página Inicial”, todos os dados são importados para o modelo do Power BI, e os campos transformados ficam disponíveis no painel “Campos”.

Vamos criar os primeiros exemplos de visualização, com 2 gráficos. O primeiro será um gráfico de barras empilhadas:

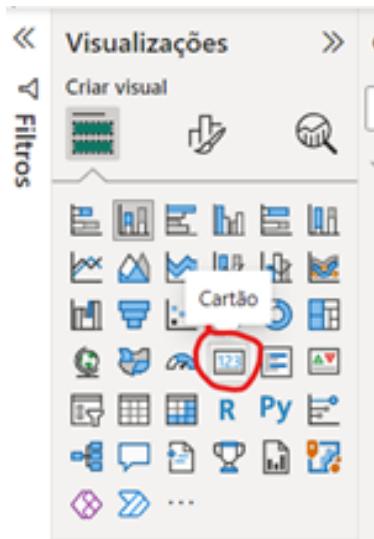


No painel de campos, clique e arraste o campo “Ano” para “Eixo X” do gráfico, e o campo “Bank Name” para o Eixo Y.

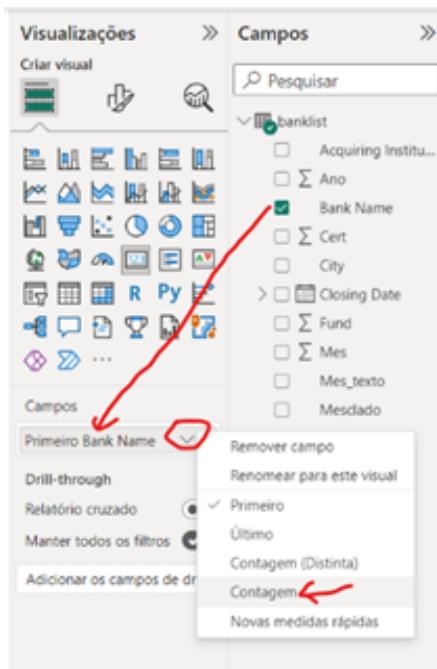
O gráfico irá ser gerado conforme a figura abaixo:



O segundo componente gráfico será um cartão:



Clique em uma área livre do dashboard para posicionar o cartão. Após isso, arraste o campo “Bank Name” para “Campos” da visualização do cartão, clique na setinha para abrir as opções e selecione a função “Contagem”, para mostrar o total de bancos existentes na tabela.



HANDS-ON - Atividade prática 2:

Vamos criar uma tabela de calendário utilizando códigos DAX, para poder analisar dados por períodos.

1. Selecionar a visão de dados.
2. Clicar no botão “Nova tabela”.

Bank Name	City	State	Cert	Acquiring Institution	
National Bank	Lino Lakes	MN	23306	Farmers & Merchants Savings Bank	sexta-j
First Southern Bank	Batesville	AR	58052	Southern Bank	sexta-j
United Americas Bank, N.A.	Atlanta	GA	35065	State Bank and Trust Company	sexta-j
Appalachian Community Bank, FSB	McCaysville	GA	58495	Peoples Bank of East Tennessee	sexta-j
Chestatee State Bank	Dawsonville	GA	34578	Bank of the Ozarks	sexta-j
The Bank of Miami,N.A.	Coral Gables	FL	19040	1st United Bank	sexta-j
Earthstar Bank	Southampton	PA	35561	Polonia Bank	sexta-j
Paramount Bank	Farmington Hills	MI	34673	Level One Bank	sexta-j

3. No campo que é mostrado, vamos criar 3 colunas, digitando os valores abaixo e finalizando com ENTER em cada um.
 - 1ª coluna: Calendario = CALENDARAUTO()

Clique no botão “Nova coluna” e digite o comando abaixo. Faça isso para todas as 4 etapas descritas.

- 2^a coluna: Ano = `YEAR('Calendario'[Date])`

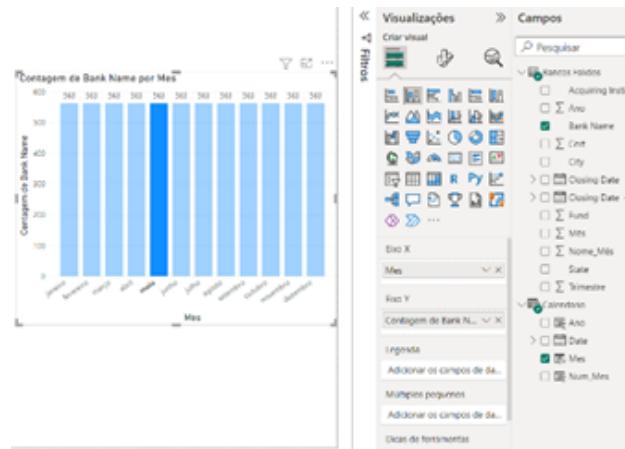
O nome da tabela ‘Calendario’ é opcional.

- 3^a coluna: Num_Mes = `MONTH([Date])`
- 4^a coluna: Mes = `FORMAT([Date], "mmmm")`.

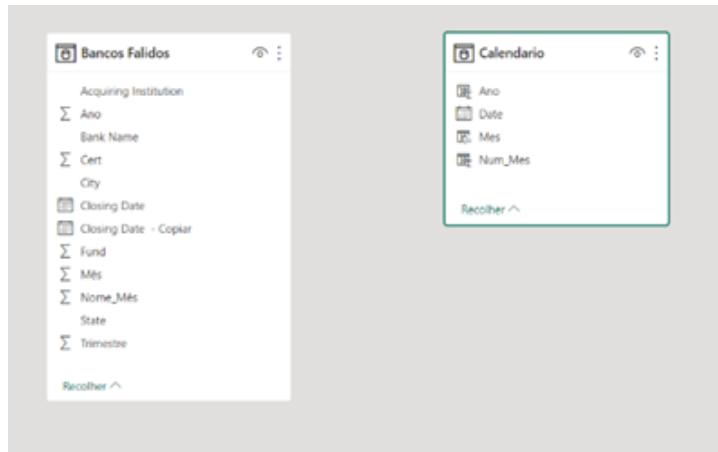
Após criado, altere o segundo parâmetro, variando para “mmm”, “mmm-yyyy”, “ddd”, “ddd” entenda o significado gerado por cada alteração.

4. Após isso, criar um gráfico de colunas empilhadas. Colocar o campo “Mês” no Eixo X e “Bank Name” o Eixo Y

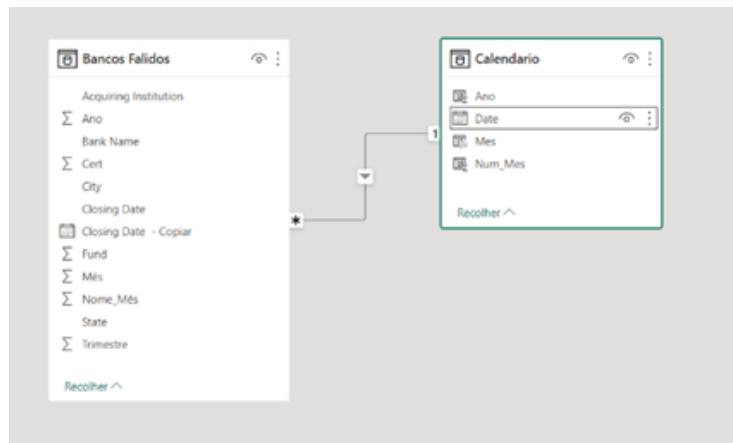
Verifique no gráfico, que para cada mês a contagem de bancos falidos se repete (563).



Isso é causado por conta de o modelo não estar adequadamente configurado. As duas tabelas foram criadas e estão “soltas”, sem o relacionamento necessário entre elas.

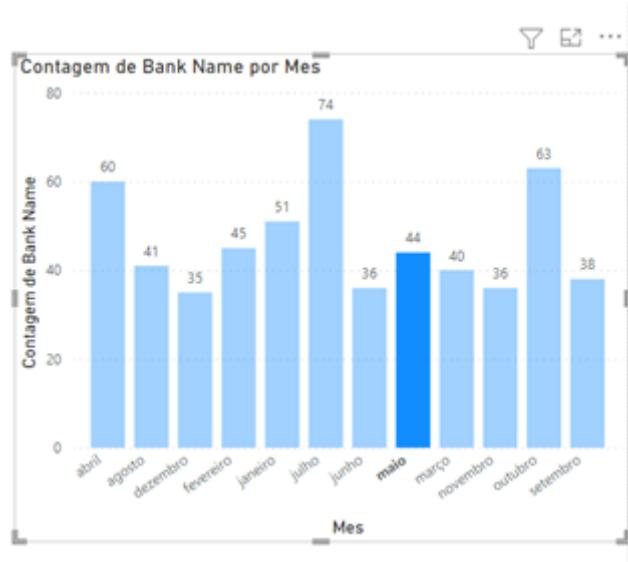


- As tabelas devem ser relacionadas para que a informação mostrada seja apropriada. Devemos relacionar as 2 tabelas utilizando a coluna data (Date → Closing Date)

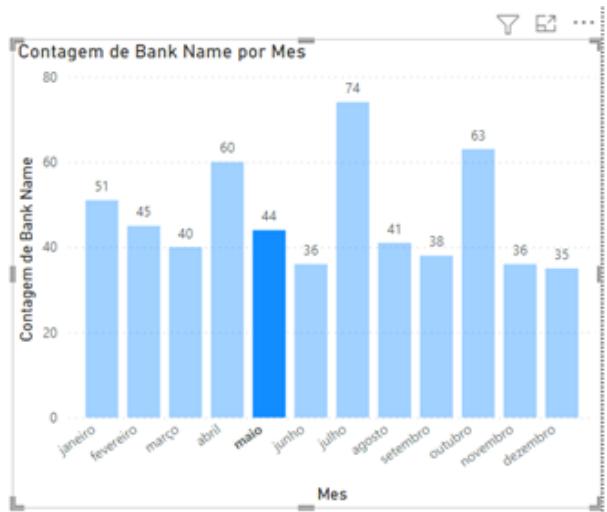


Com isso, fica identificado que foi criado um relacionamento 1-* (lê-se “1 para muitos”).

Vamos retornar para a visualização e verificar se resolveu.



6. Em relação ao número total por mês, foi resolvido. Mas ainda temos o problema da ordem dos meses de forma alfabética. Para resolver isso, vamos realizar os seguintes procedimentos:
 - a. Clique para selecionar na tabela calendário o campo Mes.
 - b. Selecione “Ferramenta de colunas”
 - c. Selecione “Classificar por colunas” e depois “Num_Mes”. Verifique que agora está correto o gráfico.



HANDS-ON - Atividade prática 3:

Vamos criar uma nova medida, com o total de bancos, e após isso, criar um novo gráfico de barras empilhadas, que contenha o total de bancos falidos por estado.

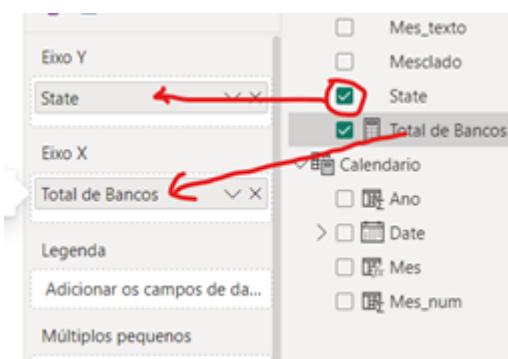
1. Clique com o botão direito sobre o nome da tabela “banklist” e escolha “nova medida”.

2. Em seguida, digite o código DAX:

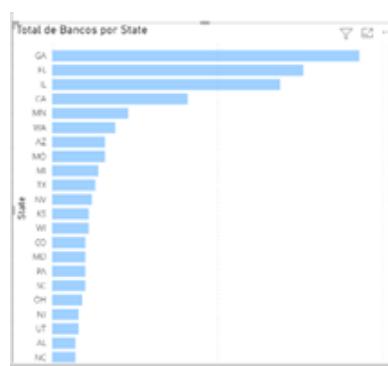
Total de Bancos = COUNTROWS('banklist')

3. Crie um novo gráfico – “Barras empilhadas”

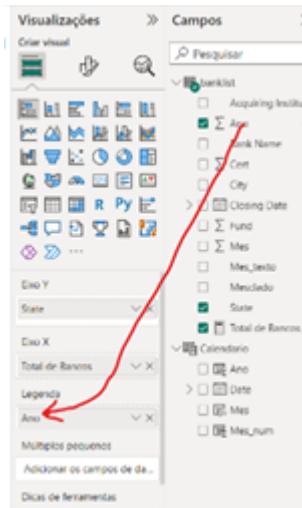
4. Marque a checkbox da coluna “State”, que deve ficar no Eixo Y.



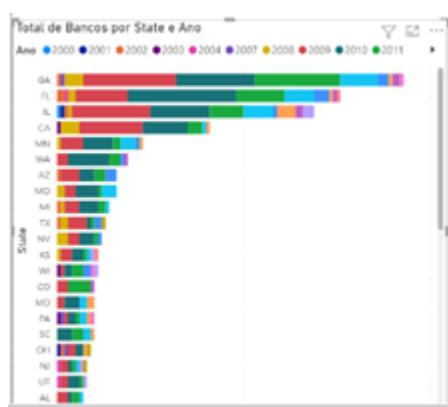
5. O gráfico resultante deve ficar da seguinte forma:



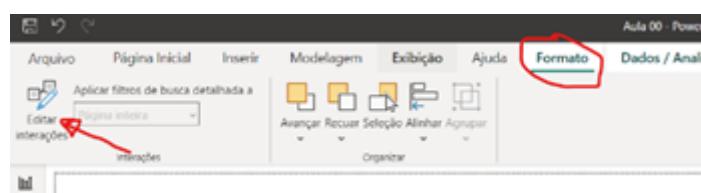
6. Podemos “fatiar” ainda a informação por estado, clicando e arrastando a coluna “Ano” para a coluna de legenda.



7. O gráfico deve ficar da seguinte forma, possibilitando agora o filtro de forma visual, ao clicar em qualquer parte do gráfico para selecionar um ano, ou vários, mantendo o CTRL pressionado.



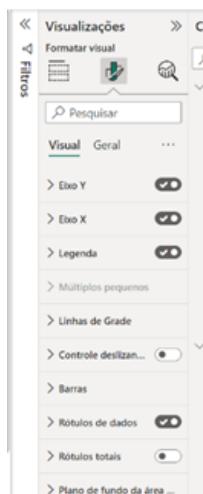
8. Agora, vamos configurar a interação do gráfico na guia “Formato” e botão “Editar interações”.



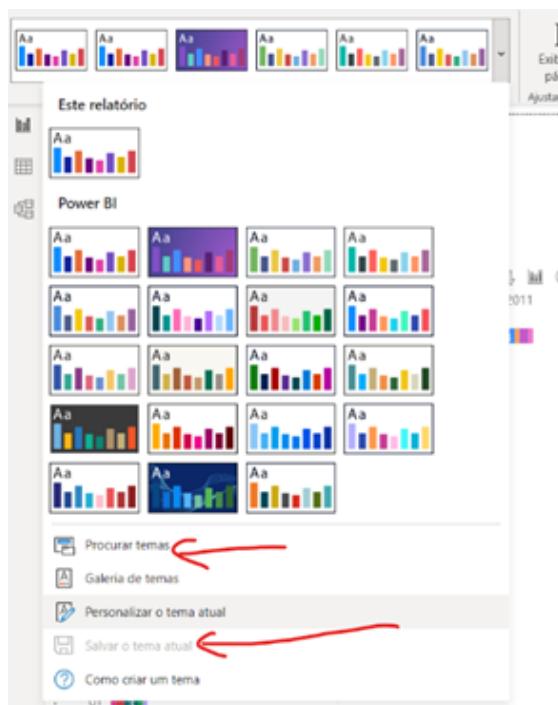
9. Esse botão permite configurar como um gráfico irá interagir com outro, através de filtros, de “highlights” dos valores, ou sem interação, conforme os símbolos da figura abaixo:



10. Ainda é possível adicionar rótulo de dados, no painel de visualizações, ao habilitar “Rótulo de dados”, como na figura abaixo:



11. No menu “exibição”, é possível criar um tema específico. Para fazer isso basta expandir o botão de temas e escolher a opção “personalizar o tema atual”. É possível personalizar cores dos gráficos, tipos, cores e tamanhos de fontes e muitas outras características, como bordas e outros., e depois salvar o tema para uso posterior. O tema é salvo em um arquivo do tipo .JSON, que pode ser compartilhado com outros usuários para manter padrões de cores da organização (a opção procurar temas permite abrir um arquivo .json).



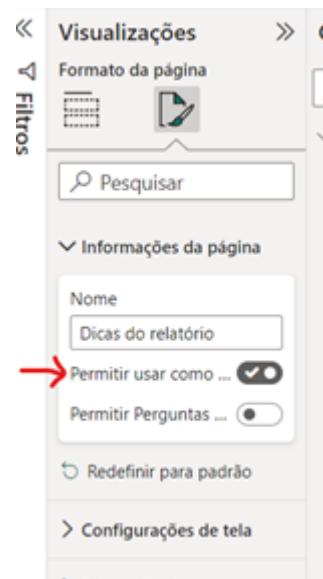
HANDS-ON - Atividade prática 4:

Contando uma história com os dados, de uma forma mais avançada.

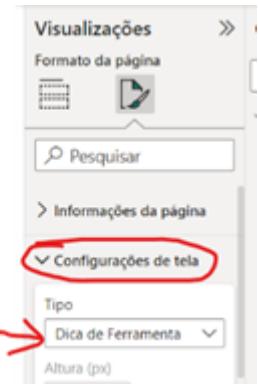
1. Vamos criar uma nova aba de visualização:



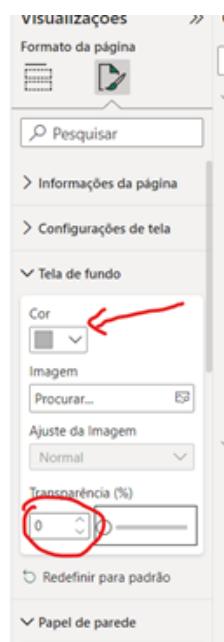
2. Vamos renomear a página para “Dicas do relatório”, e após isso, ocultar a aba. Em seguida, acesse no painel de visualização a opção de formatar a página do relatório, e marque “Permitir usar como uma dica de ferramenta”.



3. Em seguida, vamos alterar o tamanho da página, deixando como “Dica de ferramenta”:



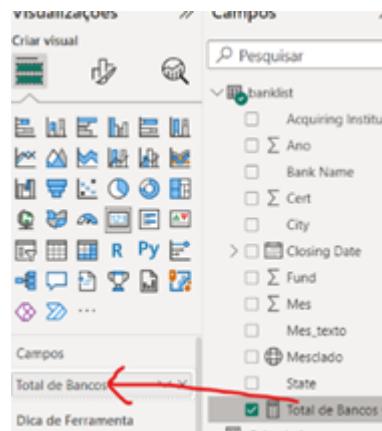
4. As próximas mudanças são na tela de fundo, deixando com uma cor cinza e transparência de 0%:



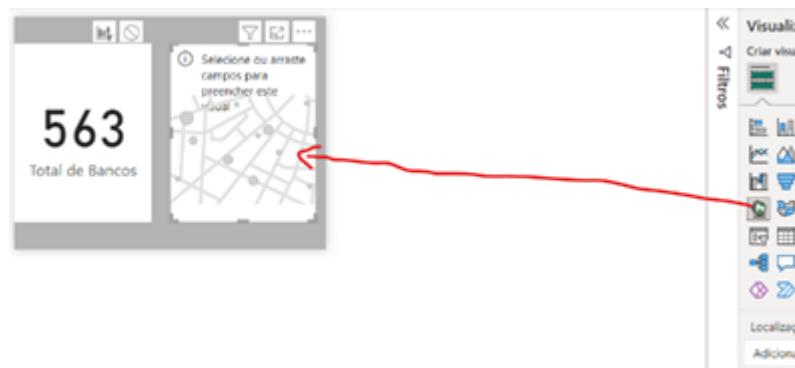
5. Adicione um cartão no visual desse novo relatório, e em seguida diminua a sua largura para que fique com aproximadamente menos da metade do tamanho.



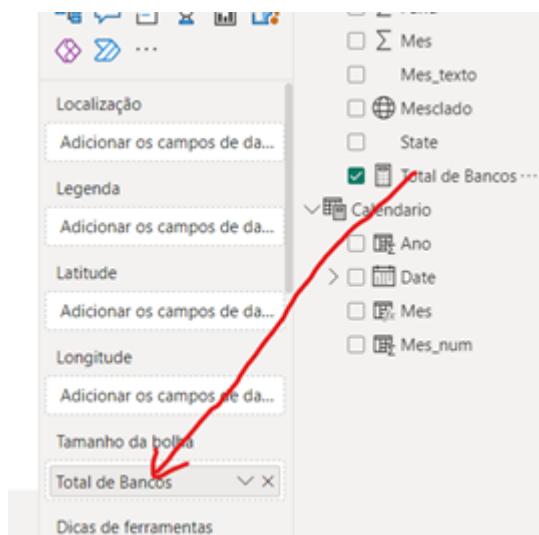
6. Adicione o Total de Campos para os campos desse cartão:



7. O próximo passo é adicionar um mapa ao lado do cartão:



8. Em seguida, adicione o campo “Total de bancos” ao campo “Tamanho da bolha”:



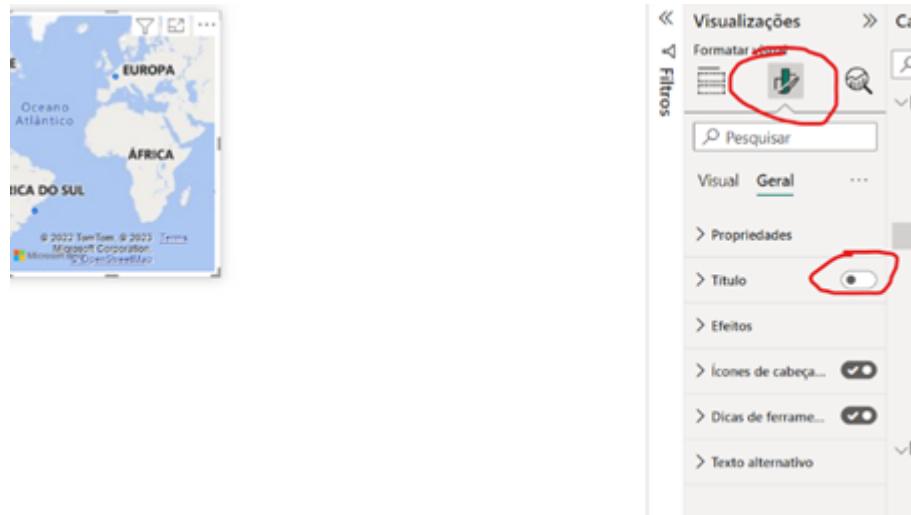
9. Clique no campo “Cidade – Estado” e arraste para o campo “Localização”:

The screenshot shows the Power BI Fields pane. In the 'Localização' section, there is a dropdown menu with 'Cidade - Estado' selected. A red arrow points from this selection to the 'Cidade - Estado' field listed under the 'banklist' category in the main pane.

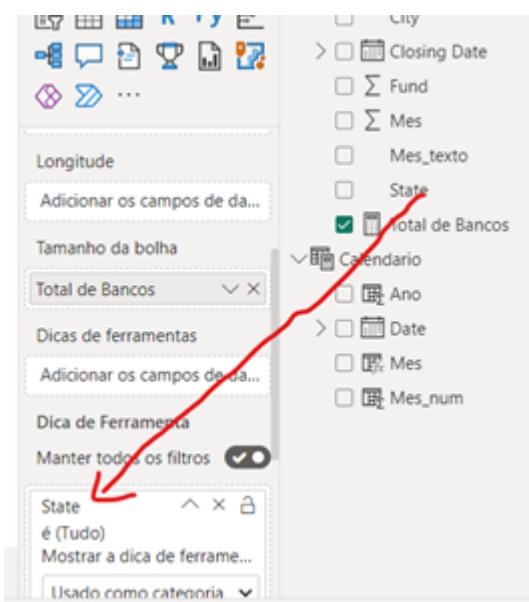
10. Selecione o campo “Cidade – Estado”, em seguida a guia “Ferramenta de coluna” e no campo “Categoria de dados”, selecione a opção “Local”.

The screenshot shows the Power BI ribbon with the 'Ferramentas de coluna' tab selected. On the left, a map visualization displays '63 de Bancos'. In the Fields pane, the 'Cidade - Estado' field is highlighted with a red circle. This indicates that the 'Cidade - Estado' field has been selected for modification.

11. Desative o título do componente de mapa.



12. Clique e arraste o campo “Estado” para o campo “Dica de ferramenta”



13. Por fim, volte ao dashboard e visualize como ficou a dica, ao colocar o ponteiro do mouse nos componentes de dados apresentados.

EXERCÍCIOS PRÁTICOS DE FIXAÇÃO – DESCOBERTA, ANÁLISE, TRANSFORMAÇÕES E VISUALIZAÇÕES DE DADOS, A PARTIR DE DOIS CONJUNTOS DE DADOS NÃO CONHECIDOS PREVIAMENTE:

Utilizando os dois endereços abaixo, vincule os arquivos a partir das url e crie pelo menos 4 gráficos diferentes.

Crie 2 (dashboards) diferentes, vinculando cada um com um dos endereços abaixo:

<<https://data.ct.gov/api/views/5mzw-sjtu/rows.csv?accessType=DOWNLOAD>>

<http://factfinder.census.gov/faces/affhelp/jsf/pages/metadata.xhtml?lang=en&type=table&id=table.en.DEC_10_SF1_SF1DP1#main_content>

Dicas:

1. Siga o roteiro realizado nas atividades práticas (*hands-on*). O primeiro passo é verificar os dados existentes, entendendo o significado de cada coluna, e realizar as transformações e limpezas dos campos, eliminando o que for desnecessário e gerando novas colunas que sejam importantes para a construção das visualizações e filtros.
2. Uma vez que os dados sejam importados para o modelo do Power BI, concentre-se na elaboração das perguntas que devem ser feitas sobre o domínio dos dados. Crie pelo menos 5 perguntas.
3. Observando cada pergunta elaborada, crie o(s) gráfico(s) que responda(m) a cada uma delas.

O dashboard só estará finalizado quando for possível responder a todas as perguntas elaboradas.

GERANDO INFORMAÇÕES VISUAIS COM O SEABORN

O **seaborn** é baseado no Matplotlib, e permite a criação de gráficos estatísticos que permite identificar a relação entre dados e outliers. Isso auxilia na seleção de atributos de datasets desconhecidos, conforme será demonstrado em exemplos de sua aplicação a seguir:

A regressão linear entre duas variáveis é representada por uma linha reta no gráfico. Isso permite identificar a relação entre 2 variáveis

Vamos utilizar primeiramente como exemplo o uso da regressão linear para associar o total da conta (total_bill) de cada cliente em um restaurante com a gorjeta (tip) deixada, no dataset "Tips", existente no seaborn.

```
import seaborn as sns
from matplotlib import pyplot as plt
df = sns.load_dataset('tips')
df
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

Realizando uma exploração inicial para entender o dataset:

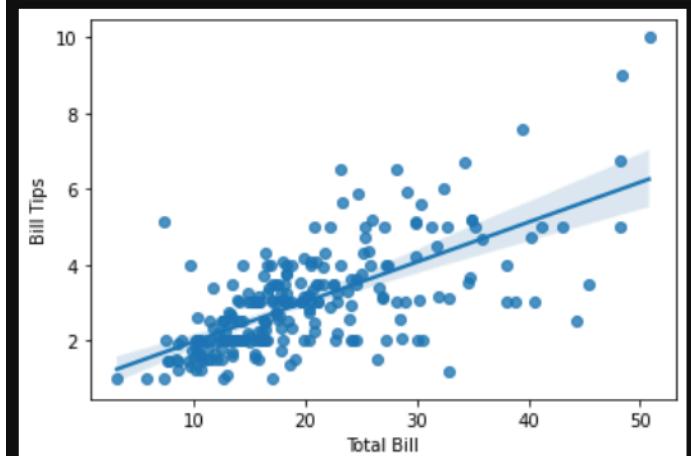
```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64  
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB
```

```
df.describe()

      total_bill      tip      size
count  244.000000  244.000000  244.000000
mean   19.785943   2.998279   2.569672
std    8.902412   1.383638   0.951100
min    3.070000   1.000000   1.000000
25%   13.347500   2.000000   2.000000
50%   17.795000   2.900000   2.000000
75%   24.127500   3.562500   3.000000
max    50.810000  10.000000   6.000000
```

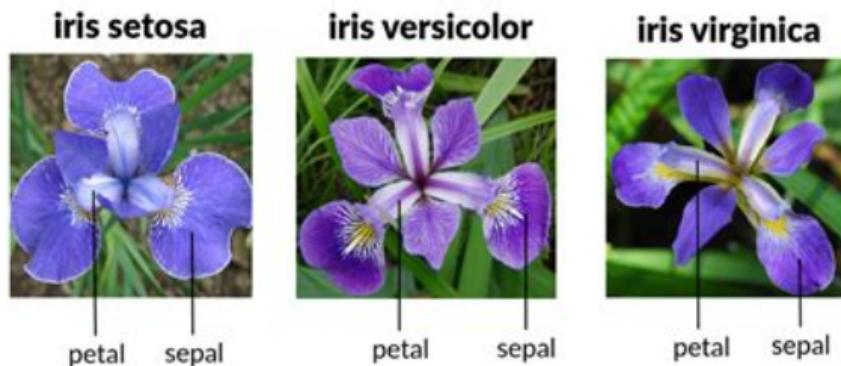
```
sns.regplot(x = "total_bill", y = "tip", data = df)
plt.xlabel('Total Bill')
plt.ylabel('Bill Tips')
plt.show()
```



CORRELAÇÃO

Refere-se a um relacionamento estatístico com dependência entre dois conjuntos de dados, como por exemplo, a correlação entre o preço de um produto e seu volume de vendas.

Como exemplo, utilizaremos o dataset **Iris** disponível na biblioteca Seaborn para tentar inferir a correlação entre o comprimento e a largura das sépalas e pétalas das 3 espécies da flor Iris.



```
import matplotlib.pyplot as plt
import seaborn as sns
df = sns.load_dataset('iris')
df
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

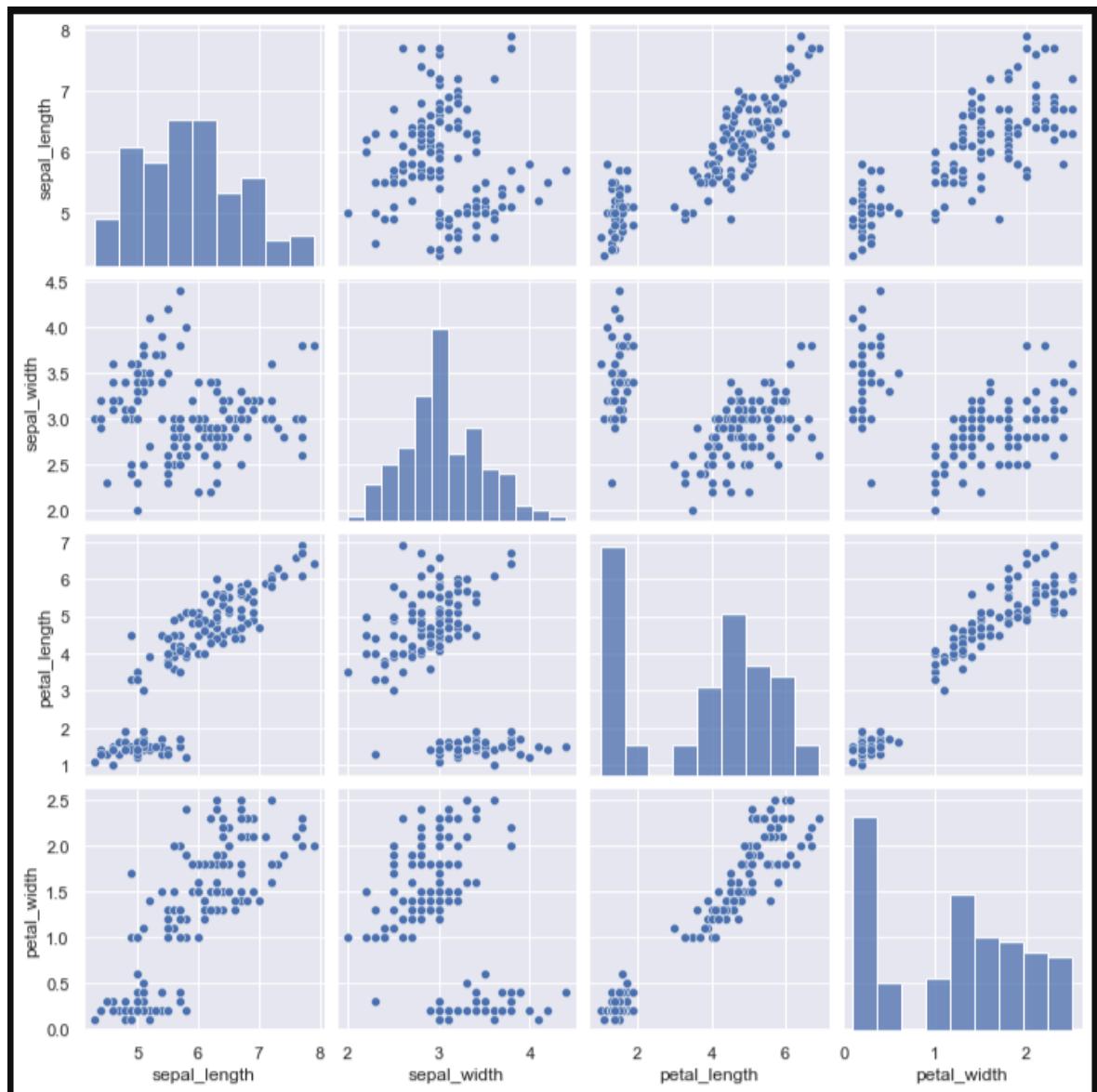
150 rows × 5 columns

O método corr() mostra que os valores mais próximos de 1 representam as maiores correlações. Podemos entender que, para identificar um elemento entre

as 3 espécies, a melhor combinação de campos (maior correlação) é entre os campos petal_length e petal_width.

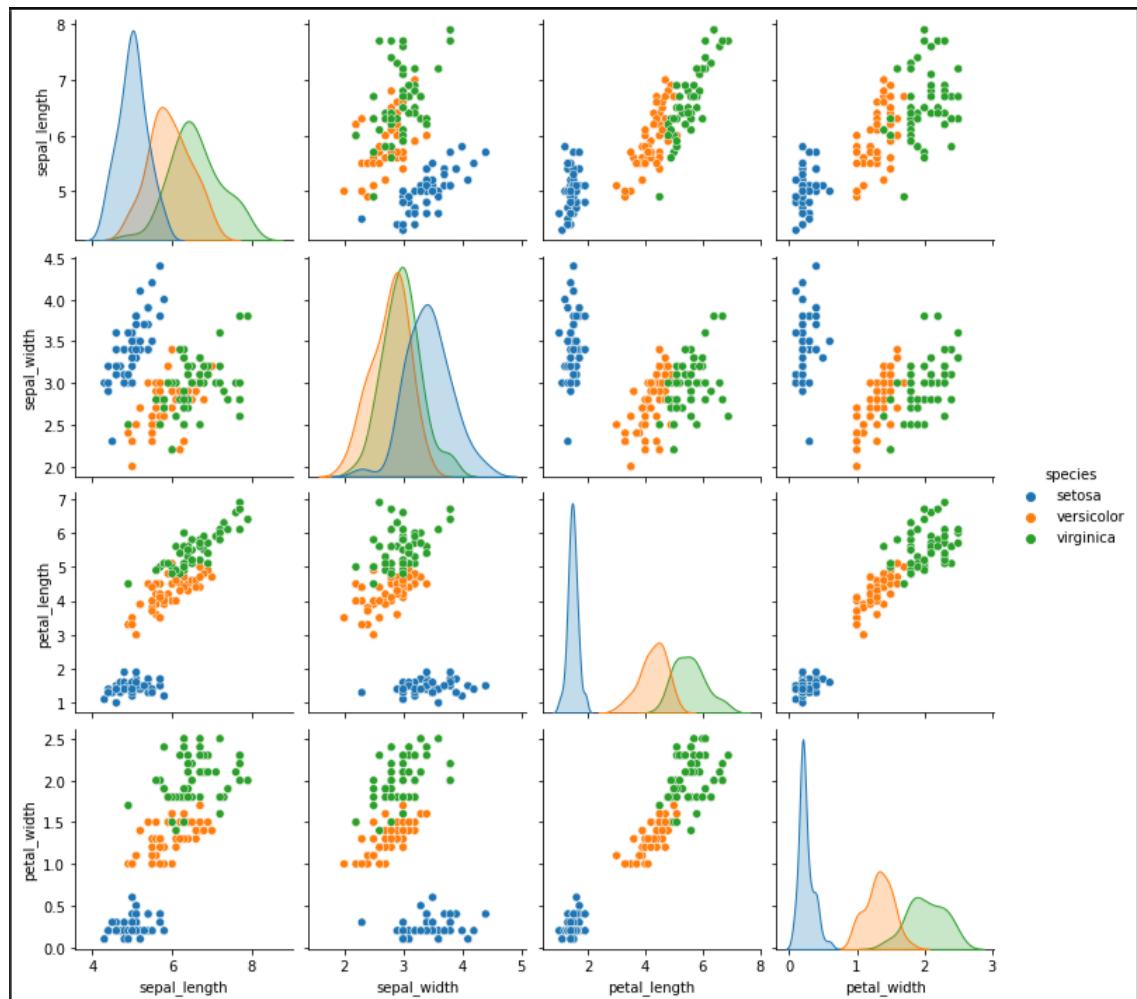
df.corr()				
	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.117570	0.871754	0.817941
sepal_width	-0.117570	1.000000	-0.428440	-0.366126
petal_length	0.871754	-0.428440	1.000000	0.962865
petal_width	0.817941	-0.366126	0.962865	1.000000

```
sns.pairplot(df, kind="scatter")  
plt.show()
```



O gráfico pode ser bem melhorado, com a inclusão do parâmetro "hue" para diferenciar na plotagem, permitindo a visualização dos *data points* das 3 espécies existentes no dataset.

```
sns.pairplot(df, kind="scatter", hue = "species")
plt.show()
```



Outros tipos de gráficos serão demonstrados com a utilização do dataset “dados_marketing.csv”.

MÓDULO 4: COMUNICAÇÃO E STORYTELLING COM DADOS (10H/AULA)

- Princípios de design para visualização de dados eficazes.
- Técnicas de narrativa baseada em dados (data storytelling) para comunicar insights a diferentes públicos.
- Exemplo usando o Power B.I.

Data Storytelling

Data storytelling dados é a estratégia de usar dados para embasar argumentos, contar histórias e envolver pessoas (ALURA, 2024).

Através da combinação de elementos visuais e uma narrativa envolvente, o storytelling com dados transforma informações complexas em insights claros e acionáveis, promovendo uma comunicação clara e eficaz, facilitando a interpretação dos dados por parte de audiências diversas, independentemente de seu nível de familiaridade com o conteúdo técnico. Por fim, melhorando na tomada de decisões informada.

Os dados podem ser quantitativos ou qualitativos e devem ser precisos, relevantes e significativos. É necessário coletá-los, prepará-los, limpá-los e selecionar bem os que estarão na narrativa.

A seleção cuidadosa dos dados a serem incluídos na história é essencial para garantir que eles suportem e enriqueçam a narrativa.

A narrativa estrutura os dados em uma sequência lógica e coerente, ajudando a audiência a entender a história por trás dos números, tendo um início, meio e fim.

A visualização de dados traduz números e estatísticas em representações visuais intuitivas, como gráficos, mapas e infográficos.

Como criar um Storytelling

De acordo com PESSOA (2023), os passos para se criar um Storytelling são: obter os dados, visualizar a história contida neles e criar narrativas atraentes. A seguir, cada um desses é apresentado.

1. Obter os dados

Utilize ferramentas de acesso a dados, determine o período e extraia para uma planilha. Sabendo o que está buscando faça uma análise explanatória. Caso contrário, inicie uma análise exploratória até encontrar algo interessante para iniciar a construção do storytelling.

2. Visualizar a história contida nos dados

Deve-se focar em como transformar esses dados em algo mais “pálpavel”. Neste caso, utilizar a visualização de dados, a partir das técnicas mostradas nos módulos 2 e 3 acima. Identifique a história que cada dado conta e, a partir disso, construa os gráficos. No entanto, deve-se tomar cuidado com quais gráficos escolher. Apesar do storytelling ser, por vezes, falado, os gráficos também precisam comunicar a mensagem certa.

3. Criar narrativas atraentes

Com os dados e gráficos em mãos, estrutura-se como a história será contada. Para começar a construir o data storytelling, é necessário:

- Entender como a análise de dados pode comprovar o ponto ou ideia
- Definir o público e a linguagem mais adequada para se comunicar com eles
- Preparar a apresentação de uma forma que tenha um início (contexto), meio (resultados) e fim (conclusões)

A partir disso, ficará mais fácil construir uma narrativa que chame a atenção do público. Podem ser usados diferentes formatos de apresentar a narrativa, como um relatório, dados em uma página em branco, gráficos e imagens, vídeos animados ou infográficos.

Observa-se que os passos 1 e 2 foram apresentados nos módulos anteriores. Desta forma, o foco deste módulo será no passo 3.

Ferramentas úteis para Data Storytelling

Pessoa (2023) apresenta algumas ferramentas úteis para storytelling, dentre elas o Google Search Console, Google Trends, Google Analytics, Tableau e Power B.I.

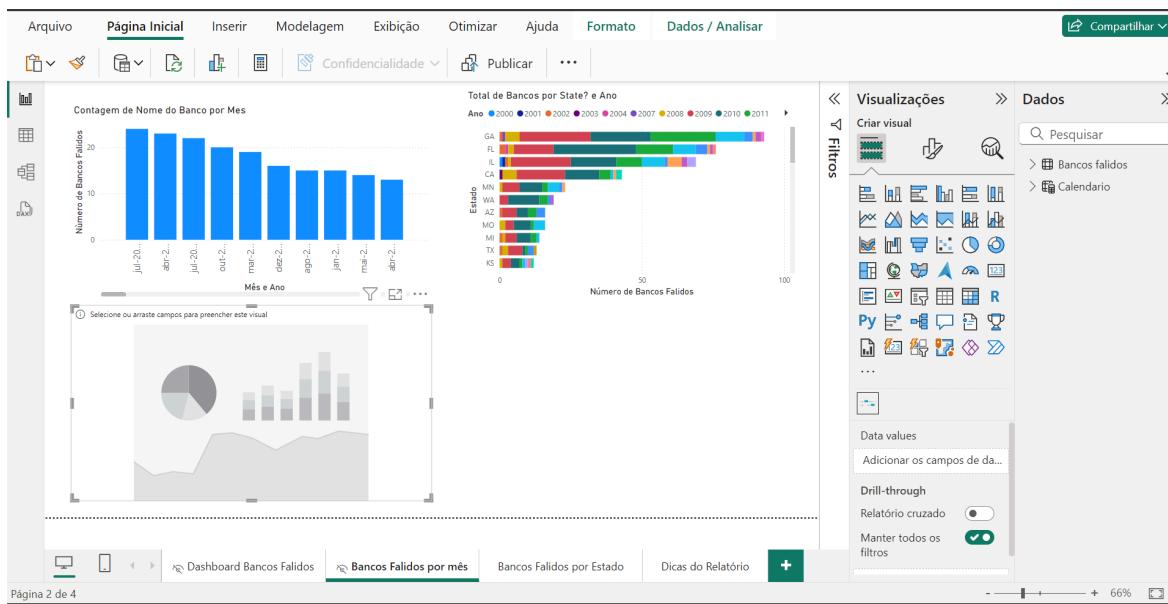
- O Google Search Console é uma ferramenta gratuita que ajuda os proprietários de sites e profissionais de SEO a monitorar e solucionar problemas com o mecanismo de busca.
- O Google Trends analisa a popularidade de termos de pesquisa específicos em várias regiões e idiomas do mundo, fornecendo insights sobre tendências de mercado, comportamento do consumidor e interesse do público em tópicos específicos ao longo do tempo. Esses dados podem ser utilizados para criar narrativas convincentes sobre tendências e mudanças no interesse do público.
- O Google Analytics é uma das principais ferramentas mais populares para análise de comportamento de sites. Ele permite que você acompanhe e analise dados detalhados sobre visitantes do site, comportamento do usuário, conversões, canais de aquisição de tráfego, e muito mais. Pode-se usar esses dados para criar histórias sobre como os usuários interagem com o seu site, quais canais de marketing são mais eficazes e como diferentes segmentos de público se comportam.
- Tableau e Power BI : permitem que se transforme os dados em gráficos e visualizações interativas.

A seguir será mostrado como criar um storytelling a partir dos dados de bancos falidos usados no módulo 2, com o Power B.I. Nesse sentido, vamos escrever um texto com dizeres tais como “Em Jan-2023, 300 bancos faliram”.

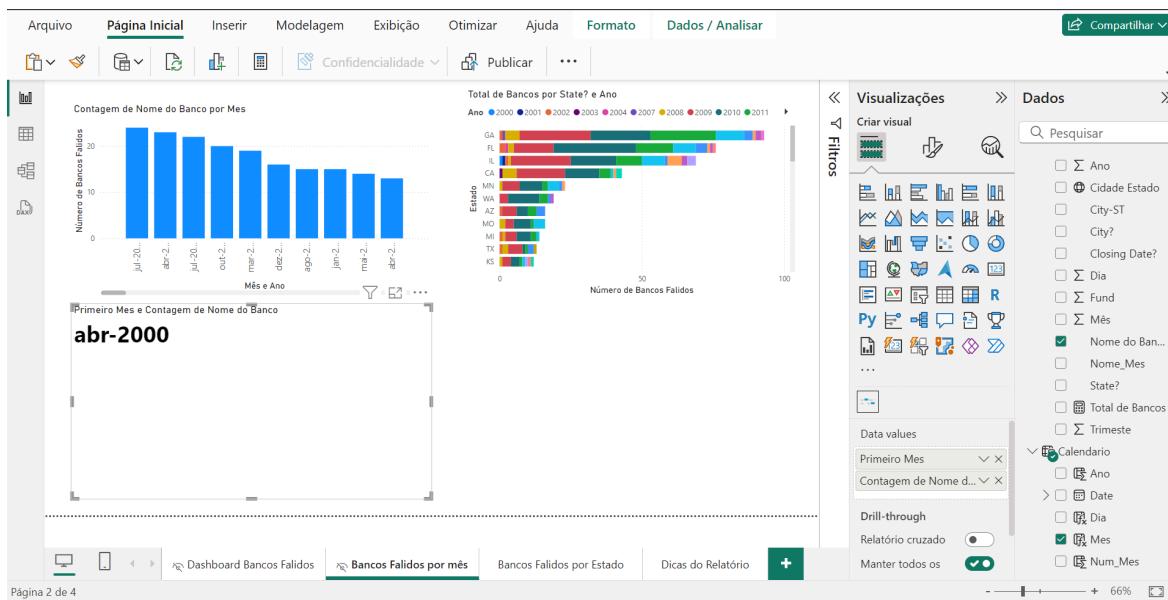
Para começar, acesse a página Bancos Falidos por Mês criada anteriormente, depois instale um visual de storytelling do Power BI. Para isso, conforme a figura abaixo, clique na opção “...” e, em seguida, clique em “obter mais visuais”.

Será apresentada a tela da imagem abaixo, denominada Visuais do Power BI. Em “Filtrar por”, selecione “Todos” e no campo de pesquisa digite “data story”. Será apresentada a opção Enlighten Data Story, da Enlighten Designs. Clique nesta opção e, em seguida, na tela deste visual, clique no botão “Adicionar”. Após a importação, o Enlighten Data Story irá aparecer no menu Visualizações, conforme o ícone abaixo (à direita).

Agora, estando na página Bancos Falidos Por Mês, clique no ícone do Enlighten Data Story, então será apresentado o visual correspondente, conforme a figura a seguir.

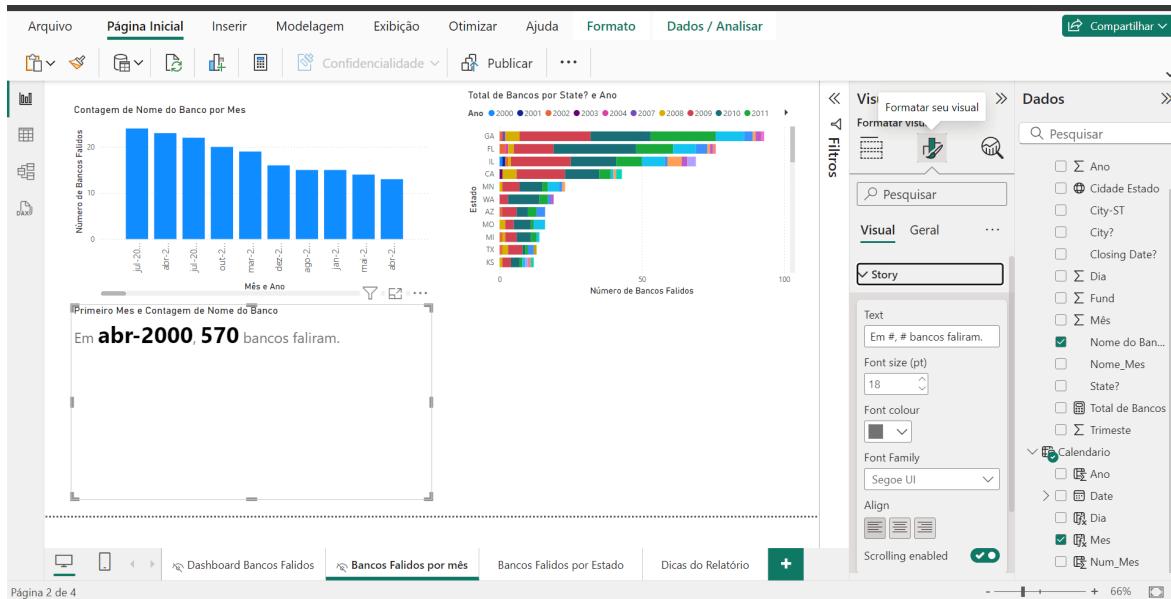


Agora vamos arrastar os campos Calendário [Mês] e Bancos Falidos [Nome do Banco] para “Data values”, tal como na figura a seguir. Os campos deverão estar na ordem de sua aparição no texto. No caso de nosso texto, a ordem é “Mes”, seguido de “Nome do Banco”. Não se preocupe com as informações que aparecerão no gráfico, pois logo serão editadas.

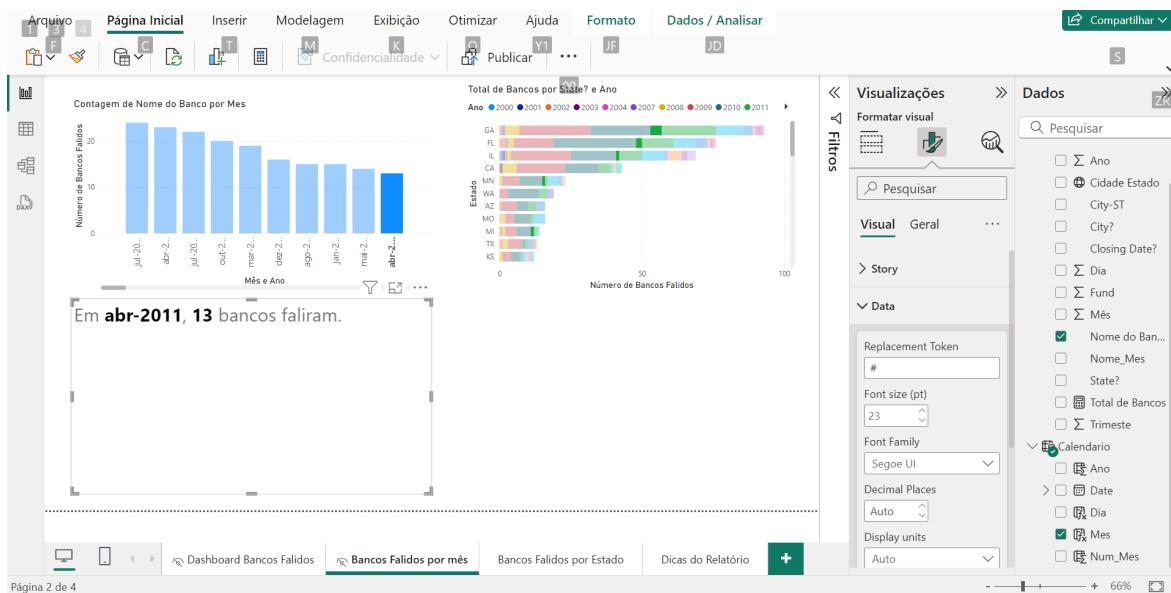


Agora, no menu de Visualizações, clique na aba “Formatar seu Visual”, conforme a imagem abaixo. Nas opções apresentadas, clique em story. Depois, em “Text”, escreva o seguinte texto: “Em #, # bancos faliram.”. Os símbolos, # (hashtag) correspondem aos campos inseridos anteriormente em Data values (Mes e Nome

do Banco), na ordem em que estão, ou seja, o primeiro # corresponde ao mês e o segundo # corresponde à contagem de bancos.



Talvez o texto não tenha ficado tão atraente. Caso queira, por exemplo, deixar o tamanho da fonte uniforme, tanto em “>Story”, quanto em “>Data”, mude o tamanho da fonte (Font size (pt)) para o mesmo tamanho, que pode ser 23. Para retirar o texto “Primeiro Mes e Contagem de Nome de Banco”, basta ir na aba “Geral” (abaixo de pesquisar) e desmarcar “Título”. Observe como ficou, na figura a seguir. Clique em algum mês do gráfico de barras “Contagem de Nome do Banco por Mes” e verá que o texto muda.



EXERCÍCIO

Considerando os outros gráficos disponíveis, crie um storytelling como quiser para cada um deles, explicando o conteúdo de cada gráfico.

RESUMO MÓDULO

Neste módulo foram vistos os conceitos de data storytelling e foi mostrado como fazer isso nos dashboard do Power B.I.

REFERÊNCIAS BIBLIOGRÁFICAS

Colocar todas as referências que foram utilizadas para a construção do conteúdo (bibliográficas, sites, links, documentos oficiais, vídeos, etc). E fique atento aos direitos autorais e as normas da ABNT.

ALURA. Storytelling com dados: transforme seus dados em narrativas envolventes. Acessado em novembro de 2024. Disponível em <<https://www.alura.com.br/artigos/storytelling-com-dados?srsltid=AfmBOopQwd9MMSqHLtPLszsiR6ZnulEVMidFnVQOvgE0c6DnrS4oa2HZ>>

CHAUDHARY, M. Learn Basics of Data Science using Titanic Dataset. Disponível em <<https://www.kaggle.com/code/manishkc06/learn-basics-of-data-science-using-titanic-dataset/notebook>>. Acessado em novembro de 2024.

DEVMEDIA. Big Data Tutorial: Como Trabalhar com Big Data na Prática. Disponível em <<https://www.devmedia.com.br/big-data-tutorial/30918>>. Acessado em novembro de 2024.

GRUS, J. Data Science do Zero - Primeiras Regras com o Python. Alta Books. ISBN: 9788576089988. Rio de Janeiro, Brasil. 2016.

INEP. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira <<https://www.gov.br/inep/pt-br>>. Acessado em novembro de 2024

KALINOWSKI, M.; ESCOVEDO T.; VILLAMIZAR, H.; e LOPES, H. Engenharia de Software para Ciência de Dados - Um guia de boas práticas com ênfase na construção de sistemas de machine learning. Casa do Código. ISBN 9788555193347. São Paulo, Brasil. 2023.

KELLEHER, J. & BRENDAN, T. Data Science. The MIT Press. ISBN 9780262535434. London, UK. 2018.

MCKINNEY, W. Python para Análise de Dados - Tratamento de dados com pandas, NumPy e Jupyter. O'Reilly - Novatec. ISBN: 9788575228418. Brasil. 3a Edição. 2023.

PANDAS. Getting Started - Pandas. Disponível em <https://pandas.pydata.org/docs/getting_started/index.html>. Acessado em novembro de 2024.

PESSOA, M. Data storytelling: o que é, qual a importância e mais Disponível em <<https://www.conversion.com.br/blog/data-storytelling-o-que-e-qual-a-importancia-e-mais/>>. Publicado em 2023 e acessado em novembro de 2024.

POWERBI. Documentação do Power BI. Disponível em <<https://learn.microsoft.com/pt-br/power-bi/>>. Acessado em setembro de 2024.

ROCKCONTENT. Guia do Data Storytelling. Disponível em <<https://rockcontent.com/br/blog/data-storytelling/>>. Acessado em novembro de 2024.

RUSSOM, P. (2011) Big Data Analytics. TDWI Best Practices Report, Fourth Quarter, 19, 1-34

SEABORN. Statistical Data Visualization. Disponível em <<https://seaborn.pydata.org/>>. Acessado em julho de 2024.

ZINOVIEV, D. Data Science Essentials in Python - Collect, Organize, Explore Predict Value. The Pragmatic Programmers. ISBN: 9781680501841. USA. 2016.

GLOSSÁRIO

Lista alfabética de conceitos e termos, com suas respectivas definições que aparecerão no texto em negrito e quando o aprendiz posicionar o mouse sobre ela, abrirá uma caixa textual com sua descrição.