

**CENTRO UNIVERSITÁRIO CARIOCA**

**ENGENHARIA DA COMPUTAÇÃO**

**YURI CONCEIÇÃO DE BRITO SCARANNI**

**UMA ABORDAGEM DIDÁTICA DO ECOSSISTEMA DA INFRAESTRUTURA DE CIÊNCIA DE DADOS**

**RIO DE JANEIRO**

**2021**

**YURI CONCEIÇÃO DE BRITO SCARANNI**

**UMA ABORDAGEM DIDÁTICA DO ECOSSISTEMA DA INFRAESTRUTURA DE CIÊNCIA DE DADOS**

Trabalho de Conclusão de Curso apresentado ao Centro Universitário Carioca, como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: Prof. D. Sc. Sérgio Assunção Monteiro

**RIO DE JANEIRO**

**2021**

|  |
| --- |
| Yuri Conceição de Brito Scaranni  Uma Abordagem Didática do Ecossistema da Infraestrutura de Ciência de Dados / Yuri Conceição de Brito Scaranni. - Rio de Janeiro, 2021.  46 f.  Orientador: Sérgio Assunção Monteiro.  TCC (Graduação em Engenharia da Computação) – Centro universitário Unicarioca. Rio de Janeiro, 2021.  1. ETL. 2. Análise de dados. 3. Python. 4. Didático. 5. Bancos de dados. |

**YURI CONCEIÇÃO DE BRITO SCARANNI**

**UMA ABORDAGEM DIDÁTICA DO ECOSSISTEMA DA INFRAESTRUTURA DE CIÊNCIA DE DADOS**

Trabalho de Conclusão de Curso apresentado ao Centro Universitário Carioca, como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

BANCA EXAMINADORA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Sérgio Monteiro Assunção, D.Sc - Orientador

Centro Universitário Carioca

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Neury Nunes Cardoso, D. Sc - Coordenador

Centro Universitário Carioca

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Alberto Tavares da Silva, D.Sc

Centro Universitário Carioca

**AGRADECIMENTOS**

Agradecemos a todos que me apoiaram desde o início deste sonho, aos meus amigos, aos meus professores, ao meu orientador prof. Sergio Monteiro Assunção por todo suporte no desenvolvimento deste trabalho e em especial a minha namorada e companheira de todos os momentos Evlyn Moraes.

**RESUMO**

A cada dia que passa observamos um aumento considerável no volume de dados que somos capazes de produzir. Seu armazenamento, antes com propósito único de gerar um histórico, ganhou mais significado quando o ramo de análise de dados cresceu e pôde mostrar informações extremamente úteis na tomada de decisão que antes passavam despercebidas. No meio desse processo, entre a geração do dado e a compreensão do mesmo fica a infraestrutura responsável por organizar esses dados se utilizando de processos automatizados para isso e é neste ponto que profissionais voltados para a ciência de dados se destacam, sendo os encarregados pela montagem de toda a engenharia envolvida no processo. Nesse contexto, o presente trabalho trata de explicar como as diversas ferramentas para processamento e análise de dados se contextualizam formando o ecossistema que viabiliza desenvolver aplicações para ciência de dados. Faremos uma abordagem tanto teórica, como prática.

**Palavras chave:** ETL, Análise de dados, Python, Didático, Bancos de dados.

**ABSTRACT**

With each passing day we observe a considerable increase in the volume of data we are able to produce. Its storage, previously with the sole purpose of generating a history, gained more significance when the data analysis industry grew and was able to show extremely useful information in decision making that previously went unnoticed. In the middle of this process, between data generation and understanding, is the infrastructure responsible for organizing this data using automated processes, and it is at this point that data science professionals stand out, being in charge of assembling all the engineering involved in the process. In this context, this paper aims to explain how the various tools for data processing and analysis are contextualized forming the ecosystem that enables the development of applications for data science. We will take a both theoretical and practical approach.

**Keywords:** ETL, Data analysis, Python, Didactic, Databases.

**LISTA DE ILUSTRAÇÕES**

[Figura 1 – Disciplinas envolvidas na ciência de dados retirado do artigo Data Science in Action 1](#_Toc58093865)9

[Figura 2 – Arquivo csv dos dados da covid19 2](#_Toc58093865)9

[Figura 3 – Download ferramenta WAMP server. 3](#_Toc58093866)1

[Figura 4 – WAMP server em execução na porta localhost 3](#_Toc58093867)1

[Figura 5 – Página inicial do phpMyAdmin 32](#_Toc58093869)

[Figura 6 – Consulta a tabela criada dentro do banco de dados 3](#_Toc58093870)3

[Figura 7 – Retorno da consulta feita sobre a tabela covid19 3](#_Toc58093870)4

[Figura 8 – Média de novos casos mensal por região (continente) 3](#_Toc58093870)5

[Figura 9 – Gráfico de mortes totais covid 3](#_Toc58093870)6

[Figura 10 – Gráfico de relação óbitos x infectados](#_Toc58093870) 38

[Figura 11 – Gráfico de total de infectados covid](#_Toc58093870) 40

[Figura 12 – Cluster em funcionamento no Databricks](#_Toc58093870) 41

[Figura 13 – Notebook simples de apresentação 41](#_Toc58093870)

[Figura 14 – Tela de upload de arquivos do databricks](#_Toc58093870) 42

[Figura 15 – Tela criação de tabela via UI](#_Toc58093870) 42

[Figura 16 – Métodos de acesso aos dados do databricks](#_Toc58093870) 43

[Figura 17 – Plotagem de gráfico no databricks](#_Toc58093870) 43

**LISTA DE CÓDIGOS**

[Código 1 – Exemplo de códigos de seleção de registros](#_Toc58093865) 17

[Código 2 – Exemplo de código de inserção de registros. 1](#_Toc58093866)7

[Código 3 – Exemplo de código de update de registros.](#_Toc58093866) 17

[Código 4 – Exemplo de código de deleção de registros.](#_Toc58093867) 18

[Código 5 – Exemplo de código de criação de tabela.](#_Toc58093867) 18

[Código 6 – Exemplo de código de deleção de tabela](#_Toc58093867) 18

[Código 7 – Exemplo de código de alteração de nome de tabela.](#_Toc58093867) 18

[Código 8 – Exemplo de código de concessão de permissão](#_Toc58093867) 18

[Código 9 – Exemplo de código de remoção de permissão](#_Toc58093867) 18

[Código 10 – Exemplo de lista em Python 2](#_Toc58093867)2

[Código 11 – Exemplo de dicionário em Python](#_Toc58093867) 23

[Código 12 – Exemplo de tupla em Python](#_Toc58093865) 24

[Código 13 – Exemplo de importação de bibliotecas em Python 2](#_Toc58093866)4

[Código 14 – Exemplo de if em Python 2](#_Toc58093866)4

[Código 15 – Exemplo de if/else em Python 2](#_Toc58093867)4

[Código 16 – Exemplo de if/elif/else em Python 2](#_Toc58093867)5

[Código 17 – Exemplo de while em Python. 2](#_Toc58093867)5

[Código 18 – Exemplo de for em Python 2](#_Toc58093867)5

[Código 19 – Exemplo de função em Python](#_Toc58093867) 25

[Código 20 – Implementação extração covid-19 ourworldindata](#_Toc58093865) 28

[Código 21 – Renomeando campos e excluindo colunas com Pandas](#_Toc58093866) 30

[Código 22 – Criando base de dados 3](#_Toc58093866)2

[Código 23 – Inserindo arquivo em uma tabela no banco de dados 32](#_Toc58093867)

[Código 24 – Query para top 10 países no ranking de casos por covid 33](#_Toc58093867)

[Código 25 – Query de média de novos casos por continente](#_Toc58093867) 34

[Código 26 – Importação de bibliotecas python 35](#_Toc58093867)

[Código 27 – Código gráfico mortes totais por covid](#_Toc58093867) 36

[Código 28 – Código gráfico infectados covid 37](#_Toc58093867)

[Código 29 – Código gráfico relação mortes x infectados 39](#_Toc58093867)

[Código 30 – Código gráfico databricks](#_Toc58093867) 43

**LISTA DE TABELAS**

[Tabela 1 – Principais funções de tratamento de strings em Python](#_Toc58093865) 20

[Tabela 2 – Operadores numéricos](#_Toc58093865) 21

[Tabela 3 – Operadores de comparação](#_Toc58093865) 22

[Tabela 4 – Operadores lógicos](#_Toc58093865) 22

[Tabela 5 – Funções de listas](#_Toc58093865) 22

[Tabela 6 – Funções de dicionários](#_Toc58093865) 23

SUMÁRIO

[**1 INTRODUÇÃO** 12](#_Toc58367016)

[**1.1 Objetivo**](#_Toc58367019) 12

[**1.2 Organização**](#_Toc58367019) 12

[**2 REVISÃO BIBLIOGRÁFICA**](#_Toc58367043) 13

[**3 INTRODUÇÃO A CIÊNCIA DE DADOS**](#_Toc58367044) 16

[**3.1 Banco de dados**](#_Toc58367045) 16

[3.1.1 SQL 1](#_Toc58367046)6

[3.1.2 MySQL 1](#_Toc58367046)8

[3.1.3 phpMyAdmin 1](#_Toc58367046)8

[3.1.4 WAMP 1](#_Toc58367046)8

[**3.2 Python**](#_Toc58367045) 19

[3.2.1 Introdução Python 1](#_Toc58367046)9

[3.2.2 Pandas](#_Toc58367046) 26

[3.2.3 Matplotlib](#_Toc58367046) 26

[3.2.4 Seaborn 26](#_Toc58367046)

[**3.3 ETL**](#_Toc58367045) 26

[3.3.1 Extract 26](#_Toc58367046)

[3.3.3 Transform 27](#_Toc58367046)

[3.3.3 Load 27](#_Toc58367046)

[**3.4 Databricks**](#_Toc58367045) 27

[**4 ESTUDO DE CASO**](#_Toc58367044) 28

[**4.1 Integração com ETL**](#_Toc58367045) 28

[4.1.1 Extração dos dados](#_Toc58367046) 28

[4.1.2 Transformação 2](#_Toc58367046)9

[4.1.3 Carga 30](#_Toc58367046)

[**4.2 Análise de dados via cliente sql**](#_Toc58367045) 33

[**4.3 Análise de dados gráfica com python**](#_Toc58367045) 35

[**4.4 Análise de dados com databricks**](#_Toc58367045) 40

[**CONCLUSÃO**](#_Toc58367044) 44

[**BIBLIOGRAFIA**](#_Toc58367044) 45

# Introdução

A ciência de dados surgiu nos anos 1970 como um ramo da tecnologia que combinava a estatística com métodos científicos e análise de dados com o objetivo de extrair informações a partir de dados, desde então a ciência de dados teve, além de um crescimento muito grande, uma transformação muito grande nos últimos anos muito por conta da facilidade e suporte a tomadas de decisão que proporcionou, deixando de ser um ramo somente acadêmico se tornando muito comum no meio corporativo. E um dos principais motivos que corrobora esse crescimento e a necessidade da aplicação da ciência de dados nas empresas é a chance de enxergar os dados com outros olhos, partindo daquilo que é real e palpável como um volume de vendas e alcançar insights cada vez mais conclusivos e até preditivos com o auxílio do aprendizado de máquina, o que acaba sendo um diferencial para a empresa em relação aos concorrentes.

Então para empresa olhar para os dados se tornou a estrada do sucesso, ajudando no controle de todos os setores e nas tomadas de decisão para seu crescimento e correção de falhas, porém, com a expansão da área a demanda por profissionais qualificados só aumentou nos últimos anos e a profissão de engenheiro(a)/cientista de dados tem sido citada como as mais promissoras do futuro então é fundamental que a sociedade tenha mais profissionais voltadas para esse seguimento.

1.1 Objetivo

Compreendendo a necessidade de mercado para atuantes na área de ciência de dados este trabalho tem por objetivo jogar luz sobre o ramo e mostrar de forma didática e prática os primeiros passos para conhecer e se envolver no ramo. Explicando as ferramentas utilizadas e toda a infraestrutura/arquitetura envolvida para os primeiros passos na área, partindo desde a explicação do motivo da utilização de determinada ferramenta até a execução prática de todo o processo.

1.2 Organização

O trabalho foi dividido em cinco capítulos, sendo este primeiro capítulo uma introdução ao tema juntamente com o objetivo e organização do trabalho.

O segundo capítulo mergulha na história da ciência de dados, descrevendo eventos relevantes que a popularizaram.

O terceiro capítulo visa fazer uma introdução as tecnologias que estão diretamente envolvidas com a ciência de dados além de descrever o seu funcionamento.

O quarto capítulo é um estudo de caso em que utilizamos as ferramentas descritas no capítulo três para realizar uma análise e armazenamento das informações sobre o coronavírus no mundo.

O quinto capítulo apresenta as conclusões do trabalho.

# Revisão bibliográfica

Ciência de dados, *Data Science*, se tornou uma disciplina importante no cenário global nos últimos anos. Composta por disciplinas que já eram consolidadas há tempos como estatística, bancos de dados, mineração e análise de dados, a combinação desses elementos possibilitou transformar uma imensa quantidade de dados em informações valiosas pra empresas, pessoas e a sociedade como um todo (van der Aalst W. 2016).

Com o avanço tecnológico o mundo produziu mais dados, um exemplo disso se dá no estudo da IDC Digital Universe Study, de abril de 2014 que estimava que a quantidade de informação digital armazenada em 2014 ultrapassava a barreira dos 4 Zettabytes e cresceria 10 vezes mais nos próximos seis anos. Um valor que é representado por 2^70 bytes, para termos de comparação 1 MB representa 2^20 bytes (van der Aalst W. 2016). Em meio a essa grande quantidade de dados temos aqueles gerados no nosso dia a dia, como redes sociais, voos, estudos, pesquisas etc., basicamente tudo que fazemos se torna dado e boa parte deles é tida como dados não estruturados, comumente conhecida como BIG DATA, e a fim de tratar estes dados a ciência de dados emerge.

Definida como um campo interdisciplinar a ciência de dados sempre visou, e ainda visa, transformar dados em valores úteis, sejam eles dados estruturados ou não estruturados, grandes ou pequenos, estáticos ou transmitidos em tempo real. Estes valores úteis que os dados geram podem vir em forma de predições, decisões automatizadas ou qualquer tipo de visualização que entregue *insights.*

A ciência de dados inclui no seu processo a extração, preparação, exploração, transformação, armazenamento e recuperação de dados, inclui também a infraestrutura computacional, vários tipos de mineração e a apresentação desses resultados considerando os aspectos éticos, sociais, legais e comerciais (van der Aalst W. 2016).

*Data science* evoluiu de forma geral quando se destaca da estatística comum para ser algo além no que diz respeito a análise de dados, como disse John Wilder Tukey (químico, estatístico e topólogo que desempenhou um papel fundamental no desenvolvimento e no estudo da estatística em meados do século XX, campo este que se beneficiou enormemente de suas contribuições) em 1962 no livro "The Future of Data Analysis", “por muito tempo acreditei ser um estatístico, mas na realidade tinha um interesse maior por analisar dados do que fazer inferências estatísticas”. Tukey menciona também que a análise de dados e as partes das estatísticas que a compõem devem assumir características da ciência e não da matemática, era o começo da disseminação da ciência de dados, sendo Tukey um dos primeiros a alçar luz sobre o tema e mesmo que seus textos tenham mais de 50 anos eles ainda retratam a realidade dos dias atuais tendo em vista que mais do que nunca precisamos analisar tudo a nossa volta.

Passados alguns anos, já em 1974 Peter Naur (astrônomo e cientista da computação dinamarquês e vencedor do concurso 2005 A.M. Prêmio Turing, a maior homenagem em ciência da computação) publica "Concise Survey of Computer Methods" na Suécia e nos Estados Unidos. Neste livro Naur lista os principais métodos contemporâneos de processamento de dados que eram úteis para muitas aplicações. Em seu livro Naur utiliza o conceito de dados definido pela International Federation for Information Processing (IFIP) em "Guide to Concepts and Terms in Data Processing" que afirma que o dado é uma representação de fatos ou ideias de uma maneira formalizada, capaz de ser comunicada ou manipulada por algum processo. A partir deste conceito Naur utiliza em seu livro o termo 'ciência de dados' algumas vezes além de oferecer uma definição para ela: “A ciência de lidar com dados, uma vez que tenham sido estabelecidos, enquanto a relação dos dados com o que eles representam é delegada a outros campos e ciências”.

Tal como a ciência da computação há alguns anos teve como raiz muitas áreas sendo a principal delas a matemática, a ciência de dados segue caminho parecido. No caso da ciência da computação ela surgiu devido a disponibilidade de recursos de computação e a necessidade de profissionais voltados para isto, a ciência de dados surge por conta da imensa quantidade de dados e a necessidade de quem trabalhe esses dados de forma que eles agreguem algum valor as companhias.

A ciência de dados se tornou então uma junção de várias disciplinas como bem mostra a figura 1. Sendo que dependendo da área e da empresa na qual o cientista vai atuar algumas disciplinas tendem a se sobressair mais, logo, alguém que trabalhará com persistência de dados tende a usar mais bancos de dados e alguém que trabalhe mais com pesquisa tende a usar mais a estatística e ambos continuam sendo cientistas de dados em suas raízes. Algumas das principais técnicas são:

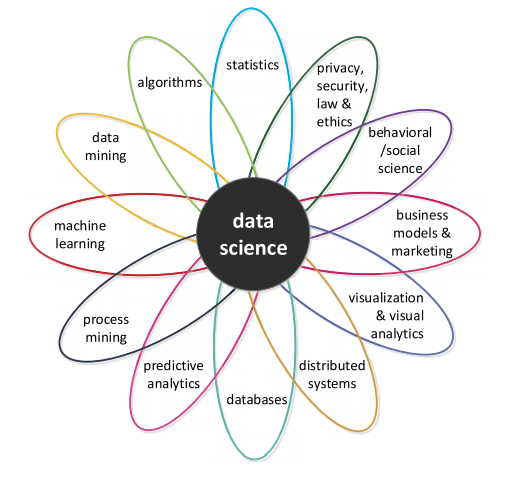


Figura 1 – Disciplinas envolvidas na ciência de dados retirado do artigo Data Science in Action.

* A estatística pode é enxergada muitas vezes, inclusive pelos autores citados neste trabalho, como a origem da ciência de dados, trazendo para dentro do ramo muitos dos seus princípios. Levando em conta que a disciplina é tipicamente dividida em duas partes, sendo uma a estatística descritiva (para resumir os dados da amostra usando noções como média, desvio padrão e frequência) e estatística inferencial (usando dados de amostra para estimar as características de todos os dados ou testar uma hipótese), comportamento que se repete nas análises de dados atualmente.
* Algoritmos é outra disciplina crucial em qualquer abordagem de análise de dados. Quando os conjuntos de dados ficam maiores, a complexidade dos algoritmos se torna uma preocupação já que em muitos casos a performance é levada em conta e algo não bem elaborado por ruir a análise. Outras disciplinas que surgiram ao longo dos anos e complementam a ciência de dados são a mineração de dados, que pode ser definida como “a análise de conjuntos de dados (geralmente grandes) para encontrar relacionamentos insuspeitados e resumir os dados de novas maneiras que sejam compreensíveis e úteis para o proprietário dos dados”.
* O aprendizado de máquina, que se preocupa com a questão de como construir um computador, uma inteligência capaz de aprender com a própria experiência e melhorar seu desempenho, sendo notadamente um campo derivado da Inteligência Artificial. Bancos de dados, que são usados ​​para armazenar dados e é um dos pilares da ciência de dados.
* Visualização e análise visual são elementos-chave da ciência de dados, até porque as pessoas precisam interpretar os resultados e orientar a análise ainda porque esta análise depende fortemente do julgamento humano e da interação direta com os dados.
* Modelos de negócios e marketing porque a ciência de dados trata da transformação de dados em valor, incluindo valor comercial.
* As ciências sociais e comportamentais, pois, a maioria dos dados são (indiretamente) gerados por pessoas e os resultados da análise são frequentemente utilizados ​​para influenciar pessoas (por exemplo, orientar o cliente para um produto ou incentivar um gerente a eliminar o desperdício).
* Privacidade, segurança, lei e ética são ingredientes essenciais para proteger os indivíduos e organizações de práticas “ruins” de ciência de dados. Privacidade refere-se à capacidade de isolar informações confidenciais. A privacidade muitas vezes depende de mecanismos de segurança que visam garantir a confidencialidade, integridade e disponibilidade dos dados.

Hoje podemos notar no nosso dia a dia o quão necessária e poderosa pode ser a ciência de dados empregada inclusive em governos, como em 2012 quando o então presidente dos EUA, Obama, contratou cientistas de dados que fizeram diversos estudos como uma forma de identificar os eleitores que precisavam de uma atenção a mais, otimizar programas e recursos para captação de fundos de doadores específicos e focando esforços para votos onde eles seriam úteis (Joel Grus, 2015, p. 31), esse conjunto de atitudes serviram como base para a reeleição do presidente, o que nos leva a crer que a ciência de dados estará cada vez mais envolvida em diversas facetas da nossa sociedade.

# Introdução à Ciência de Dados

A ciência de dados por ser uma área que junta diversas disciplinas dentro dela acaba por dar maior destaque aos profissionais que conseguem mesclar o conhecimento nessas disciplinas somado ao conhecimento técnico para exercer de forma performática toda a teoria que envolve essas disciplinas. Então uma boa forma de começar é entendendo todas as tecnologias que estão ali envolvidas e as principais ferramentas do mercado. Este trabalho lida com todas as tecnologias disponíveis em versões gratuitas.

3.1 Banco de dados

Boa parte dos dados que precisamos e acessamos nos dias de hoje estarão em bases de dados, utilizando os sistemas SGBD (Sistemas Gerenciadores de Bancos de Dados) para armazenar e consultar os dados de forma eficiente. A maioria dos sistemas conhecidos são de bancos de dados relacionais como por exemplo o MySQL, Oracle e SQL Server, bancos estes que armazenam os dados em tabelas e são consultados através da linguagem SQL (Structured Query Language), uma linguagem exclusivamente utilizada para consulta de dados. Os SGBD através do SQL oferecem ao usuário uma forma estruturada, organizada e segura de armazenar seus dados, além de uma interface para consultas de forma simples e direta.

3.1.1 SQL

Structured Query Language, é uma linguagem de programação em bancos de dados padronizada e universal que surgiu na década de 70 e foi criada por E. F. Codd. Idealizada para atender ao modelo relacional de bancos de dados o SQL evoluiu e adquiriu no caminho diversos dialetos com pequenas variações de termos ou sintaxe entre eles. SQL é uma linguagem utilizada para consultar (essas consultas são chamadas de Query) e visualizar dados dentro do banco de dados, além de alterar estrutura de tabelas, criar bases de dados, inserir e atualizar valores, excluir tabelas e registros, em resumo, SQL é a linguagem que se comunica com o banco de dados executando todos os comandos dentro dele. A linguagem SQL pode ser dividida em quatro enfoques, que são eles DML, DDL, DCL e DQL.

DML (Data Manipulation Language) é um subconjunto da linguagem SQL, utilizada para selecionar (SELECT), inserir (INSERT), atualizar (UPDATE) eapagar(DELETE), essas quatro operações são conhecidas também como CRUD (Create, Read, Update e Delete).

DDL (Data Definition Language) são comandos voltados para manipulação de tabelas e elementos associados, como chave primária e chaves estrangeira, índices etc. Os principais comandos são CREATE, DROP, ALTER (em algumas situações).

DCL (Data Control Language) são comandos voltados para autorização de dados e licenças de usuários para manipulação dentro do banco de dados. Alguns comandos comuns são GRANT (garante privilégios para usuários), REVOKE (revoga privilégios de usuários), COMMIT (realiza a gravação de dados no banco que estão aguardando em sessão) e ROLLBACK (descarta dados existentes desde o último COMMIT).

DQL (Data Query Language) possui somente um comando, SELECT, sendo que ele está também no grupo DML, além do comando no DQL estão contidas diversas cláusulas (FROM, WHERE, GROUP BY, HAVING, ORDER BY, DISTINT), operadores lógicos (AND, OR, NOT), Operadores de Comparação (<, >, <>, <=, =, >=, BETWEEIN, LIKE) e funções de Soma (AVG, COUNT, SUM, MIN, MAX).

Alguns exemplos de comandos:

/\* Retorna todos os campos \*/

SELECT \* FROM database.tabela ;

/\* Retorna somente os campos campo\_1 e campo\_2 \*/

SELECT campo\_1, campo\_2 FROM database.tabela;

/\* Retorna todos os campos onde campo\_1 é igual a 1 \*/

SELECT \* FROM database.tabela WHERE campo\_1 = 1;

/\* Retorna o campo\_1 e a quantidade de registros agrupados pelo campo\_1 \*/

SELECT campo\_1, COUNT(\*) FROM database.tabela GROUP BY 1;

/\* Retorna o campo\_1 e a soma dos valores do campo\_1 agrupados \*/

SELECT campo\_1, SUM(\*) FROM database.tabela GROUP BY 1;

/\* Retorna o campo\_2 quando campo\_1 é igual a 1 e o campo\_2 tem algum registro que inicie com ‘Yur’ \*/

SELECT campo\_2 FROM database.tabela WHERE campo\_1 = 1 AND campo\_2 LIKE 'Yur%';

Código fonte 1 – Exemplo de códigos de seleção de registros.

INSERT INTO database.tabela (campo\_1, campo\_2) VALUES (‘1’, ‘Yuri’);

Código fonte 2 – Exemplo de código de inserção de registros.

UPDATE database.tabela SET campo\_1 = 2;

Código fonte 3 – Exemplo de código de update de registros.

Código fonte 4 – Exemplo de código de deleção de registros.

CREATE TABLE database.tabela\_nova (campo\_1 INT, campo\_2 VARCHAR(255) );

DELETE FROM database.tabela WHERE campo\_1 = 2;

Código fonte 5 – Exemplo de código de criação de tabela.

DROP TABLE database.tabela\_nova;

Código fonte 6 – Exemplo de código de deleção de tabela.

ALTER TABLE database.tabela RENAME TO tabela\_2;

Código fonte 7 – Exemplo de código de alteração de nome de tabela.

GRANT SELECT ON database.tabela TO usuario;

Código fonte 8 – Exemplo de código de concessão de permissão.

REVOKE SELECT ON database.tabela FROM usuário;

Código fonte 9 – Exemplo de código de remoção de permissão.

3.1.2 MySQL

Um dos Sistemas de gerenciamento de bancos de dados mais utilizados no mundo hoje, MySQL utiliza a sintaxe SQL padrão. Popularizou-se por ser uma aplicação do tipo open source e por ter uma fácil integração com muitas linguagens de programação.

3.1.3 PhpMyAdmin

PhpMyAdmin é uma ferramenta gratuita escrita na linguagem PHP utilizada para administração do banco de dados MySQL através da Web. Consegue prover uma interface gráfica que facilita a iteração com o banco de forma que é possível criar e deletar tabelas, bases de dados, além de conceder permissões e consultar tabelas.

3.1.4 WAMP

WAMP, também conhecido como WAMP Server, é um acrônimo para Windows Apache MySQL Php (Para sistemas Linux é conhecido como LAMP – Linux Apache MySQL Php). Ele é um ambiente de desenvolvimento Web do Windows. WAMP permite que você crie aplicativos web com Apache2, PHP e um banco de dados MySQL. WAMP possibilita a criação de um servidor local e executa automaticamente o banco MySQL na máquina, além disso ele já traz instalado o phpMyAdmin que é um cliente web para bancos de dados, desta forma é possível executar um sistema completo de banco de dados localmente.

3.2 Python

O Python é uma linguagem de programação de alto nível criada em 1991 por Guido van Rossum, simples e objetiva a linguagem se popularizou nos últimos anos por isso além da sua imensa flexibilidade, disponibilidade de bibliotecas e grande comunidade. Em Python é multiparadigma, seja ele funcional, orientado a objetos, estruturada, possui tipagem dinâmica e é open source.

Python pode ser usado para desenvolver e administrar grandes sistemas e aplicações, um dos seus pontos fortes é a legibilidade dos códigos, principalmente pelo fato do Python usar indentação como forma de separar blocos de código, o que retira a necessidade do uso de chaves, colchetes e palavras reservadas de inicio e fim como BEGIN e END. Python também é muito utilizado no ramo da ciência de dados por ser robusto para programação de forma geral e possuir muitas bibliotecas de apoio para análise, estatística, matemática, e tratamento e visualização de dados (ex.: Numpy, Pandas, Matplotlib e Scipy).

Com Python é possível fazer desde a extração, tratamento, até a inserção dos dados em um banco de dados, tornando uma linguagem extremamente poderosa. O python possui nativamente diversos recursos que facilitam a vida do programador como tipos números, de ponto flutuante, string, listas, tuplas, dicionários e funções.

3.2.1 Introdução Python

1) Tipos básicos

Python possui alguns tipos básicos de dados sendo eles inteiro (voltado para números inteiros), float (voltado para números de ponto flutuante/decimais) e string (conjunto de caracteres). Além disso o Python possui diversas funções voltadas para a manipulação de strings e dos demais tipos, na figura 2 estão as principais funções para tratamento de strings.

|  |  |  |
| --- | --- | --- |
| Método | Descrição | Exemplo |
| len() | Retorna o tamanho da string. | teste = “Apostila de Python” |
| len(teste) |
| 18 |
| capitalize() | Retorna a string com a primeira letra maiúscula. | a = "python" |
| a.capitalize() |
| 'Python' |
| count() | Informa quantas vezes um caractere (ou uma sequência de caracteres) aparece na string. | b = "Linguagem Python" |
| b.count("n") |
| 2 |
| startswith() | Verifica se uma string inicia com uma determinada sequência. | c = "Python" |
| c.startswith("Py") |
| True |
| endswith() | Verifica se uma string termina com uma determinada sequência. | d = "Python" |
| d.endswith("Py") |
| False |
| isalnum() | Verifica se a string possui algum conteúdo alfanumérico (letra ou número). | e = "!@#$%" |
| e.isalnum() |
| False |
| isalpha() | Verifica se a string possui apenas conteúdo alfabético. | f = "Python" |
| f.isalpha() |
| True |
| islower() | Verifica se todas as letras de uma string são minúsculas. | g = "pytHon" |
| g.islower() |
| False |
| isupper() | Verifica se todas as letras de uma string são maiúsculas. | h = "# PYTHON 12" |
| h.isupper() |
| True |
| lower() | Retorna uma cópia da string trocando todas as letras para minúsculo. | i = "#PYTHON 3" |
| i.lower() |
| '#python 3' |
| upper() | Retorna uma cópia da string trocando todas as letras para maiúsculo. | j = "Python" |
| j.upper() |
| 'PYTHON' |
| swapcase() | Inverte o conteúdo da string (Minúsculo / Maiúsculo). | k = "Python" |
| k.swapcase() |
| 'pYTHON' |
| title() | Converte para maiúsculo todas as primeiras letras de cada palavra da string. | l = "apostila de python" |
| l.title() |
| 'Apostila De Python' |
| split() | Transforma a string em uma lista, utilizando os espaços como referência. | m = "cana de açúcar" |
| m.split() |
| ['cana', 'de', 'açúcar'] |
| replace(S1, S2) | Substitui na string o trecho S1 pelo trecho S2. | n = "Apostila teste" |
| n.replace("teste", "Python") |
| Apostila Python' |
| find() | Retorna o índice da primeira ocorrência de um determinado caractere na string. Se o caractere não estiver na string retorna -1. | o = "Python" |
| o.find("h") |
| 3 |
| ljust() | Ajusta a string para um tamanho mínimo, acrescentando espaços à direita se necessário. | p = " Python" |
| p.ljust(15) |
| "Python " |
| rjust() | Ajusta a string para um tamanho mínimo, acrescentando espaços à esquerda se necessário. | q = "Python" |
| q.rjust(15) |
| " Python" |
| center() | Ajusta a string para um tamanho mínimo, acrescentando espaços à esquerda e à direita, se necessário. | r = "Python" |
| r.center(10) |
| " Python " |
| lstrip() | Remove todos os espaços em branco do lado esquerdo da string. | s = " Python " |
| s.lstrip() |
| "Python " |
| rstrip() | Remove todos os espaços em branco do lado direito da string. | t = " Python " |
| t.rstrip() |
| " Python" |
| strip() | Remove todos os espaços em branco da string. | u = " Python " |
| u.strip() |
| "Python" |

Tabela 1 – Principais funções de tratamento de strings em Python

2) Operadores

Python ainda possui alguns operadores e caracteres de comparação muito utilizados quando estamos falando de números, sejam eles do tipo inteiro (int), longos (long), decimais (float) ou complexos (complex). Na tabela 2, 3 e 4 abaixo estão listados os operadores numéricos, de comparação e lógicos, respectivamente.

|  |  |  |
| --- | --- | --- |
| Operador | Descrição | Exemplo |
| + | Soma | 1 + 1 = 2 |
| - | Subtração | 1 - 1 = 0 |
| \* | Multiplicação | 1 \* 1 = 1 |
| / | Divisão | 1 / 1 = 1 |
| % | Resto da divisão | 3 % 2 = 1 |
| \*\* | Potência | 2 \*\* 3 = 8 |

Tabela 2 – Operadores numéricos

|  |  |  |
| --- | --- | --- |
| Operador | Descrição | Exemplo |
| < | Menor que | 2 < 3 |
| <= | Menor ou igual | 3 <= 3 |
| > | Maior que | 3 > 2 |
| >= | Maior ou igual | 3 >= 3 |
| == | Igual | 3 == 3 |
| != | Diferente | 3 != 3 |

Tabela 3 – Operadores de comparação

|  |  |  |
| --- | --- | --- |
| Operador | Descrição | Exemplo |
| not | Não | not A |
| and | E | (1+1) and (10 > 1) |
| or | Ou | (1+1) or (10 > 1) |

Tabela 4 – Operadores lógicos

3) Estruturas

Temos ainda as estruturas de dados presentes em Python, sendo a principal delas a LISTA. Uma lista é um conjunto sequencial, indexado, de valores iniciado pelo index 0, identificado pelo caractere [] (colchete). Uma lista pode armazenar todo tipo de dado, de string a float e outras listas, dicionários etc. Abaixo temos algumas das funções mais utilizadas em listas.

A = [“Yuri”, 1, “Tcc”, 2000]

Código fonte 10 – Exemplo de lista em Python.

|  |  |  |
| --- | --- | --- |
| Função | Descrição | Exemplo |
| len | Retorna tamanho da lista. | A = [1,2,3] / len(A) / 3 |
| min | Retorna menor valor da lista. | A = [1,2,3] / min(A) / 1 |
| max | Retorna maior valor da lista | A = [1,2,3] / max(A) / 3 |
| sum | Retorna soma dos elementos da lista | A = [1,2,3] / sum(A) / 6 |
| append | Adiciona novo elemento no final da lista | A = [1,2,3] / A.append(4) / A = [1,2,3,4] |
| extend | Insere uma lista em outra lista | A = [1,2,3] / A.extend([4,5,6]) / A = [1,2,3,4,5,6] |
| del | Remove um elemento da lista através do índice | A = [1,2,3] / del[0] / A = [2,3] |
| in | Averigua se elemento pertence a lista | A = [1,2,3] / 1 in A / True |
| sort() | Ordena a lista | A = [3,1,2] / A.sort() / A = [1,2,3] |
| reverse() | Inverte elementos da lista | A = [1,2,3] / A.reverse() / A = [3,2,1] |

Tabela 5 – Funções de listas

A próxima estrutura de dados é o dicionário. Um dicionário é um conjunto de valores onde o valor é associado a uma chave de acesso e ele é identificado pelo caractere {} (chaves).

A = {“nome”: “Yuri”, “idade”: 25}

Código fonte 11 – Exemplo de dicionário em Python.

Dicionários são estruturas muito utilizadas em Python e abaixo estão algumas de suas principais funções.

|  |  |  |
| --- | --- | --- |
| Função | Descrição | Exemplo (A = {"nome": "Python", "idade": 30} ) |
| clear() | Remove todos os elementos do dicionário | A.clear() / A = {} |
| copy() | Retorna uma cópia do dicionário | B = A.copy() / B = {"nome": "Python", "idade": 30} |
| get() | Retorna o valor de uma chave específica | A.get("nome") / Python |
| items() | Retorna a lista contendo uma tupla para cada chave/valor | A.items() / dict\_items([('nome', 'Python'), ('idade', 30)]) |
| keys() | Retorna a lista contendo as chaves do dicionário | A.keys() / dict\_keys(['nome', 'idade']) |
| pop() | Remove um elemento com chave específica | A.pop("nome") / {'idade': 30} |
| update() | Realiza update no dicionário com a chave-valor passado | A.update({"time": "Flamengo"}) / {'nome': 'Python', 'idade': 30, 'time': 'Flamengo'} |
| values() | Retorna a lista contendo os valores do dicionário | A.values() / dict\_values(['Python', 30]) |

Tabela 6 – Funções de dicionários

Outra estrutura de dados bastante utilizada em Python são as TUPLAS, uma tupla é um conjunto sequencial de valores, também indexado, e imutável, identificado pelo caractere () (parêntereses).

A = (10, 20, 30)

Código fonte 12 – Exemplo de tupla em Python.

4) Bibliotecas

Python conta com diversas bibliotecas e para fazer uso de algum de seus métodos/funções é necessário importá-la. Boa parte das bibliotecas que o desenvolvedor precisará já virão pré-instaladas.

import datetime

print(datetime.datetime.now())

Código fonte 13 – Exemplo de importação de bibliotecas em Python.

5) Estruturas de decisão

Python conta com uma estrutura de domada de decisão clássica, com IF, ELSE e ELIF.

IF, indica que caso uma condição seja verdade o bloco abaixo será executado.

ELSE, indica que caso a condição de IF seja falsa, o bloco contido em ELSE será executado.

ELIF, elif é executado sempre que if é falso, porém, diferentemente do else, elif aceita uma condição a ser passada, ou seja, ele funciona com a lógica de “caso contrário”.

A = 10

If A > 10:

print(“estou no if”)

Código fonte 14 – Exemplo de if em Python.

A = 10

If A > 10:

print(“estou no if”)

elif A = 10:

print(“estou em elif”)

Código fonte 15 – Exemplo de if/else em Python.

A = 10

If A > 10:

print(“estou no if”)

elif A = 10:

print(“estou em elif”)

else:

print(“estou em else”)

Código fonte 16 – Exemplo de if/elif/else em Python.

6) Laços de repetição

Python conta com dois laços de repetição (loop), sendo eles while e for, onde em while o laço se mantém enquanto for verdade e em for o laço se mantém enquanto não chegou ao fim de uma condição passada, como uma contagem por exemplo.

contador = 0

while contador < 10:

print(f"ainda não cheguei a 10, estou em {contador}")

contador += 1

print("cheguei a 10")

Código fonte 17 – Exemplo de while em Python.

contador = 10

for i in range(contador):

print(f"ainda não cheguei a 10, estou em {i}")

print("cheguei a 10")

Código fonte 18 – Exemplo de for em Python.

7) Funções

Funções em Python são trechos de código reutilizáveis a partir do momento em que são chamados novamente em qualquer lugar do código. A palavra reservada para criar funções é def. E a estrutura simples de uma função é:

def soma(a, b):

print(f"O resultado da soma é {a + b}")

soma(1, 3)

Código fonte 19 – Exemplo de função em Python.

3.2.2 Pandas

O Pandas é um dos pacotes mais utilizados no que diz respeito ao tratamento de dados, ele é um pacote Python que fornece estruturas de dados rápidas, flexíveis e expressivas projetadas para tornar o trabalho com dados estruturados (tabulares, multidimensionais, potencialmente heterogêneos) e de séries temporais fácil e intuitivo. O principal objetivo do pacote é ser o bloco de construção fundamental de alto nível para fazer análises de dados práticas e do mundo real em Python. Além disso, tem o objetivo mais amplo de se tornar a ferramenta de análise / manipulação de dados de código aberto mais poderosa e flexível disponível em qualquer idioma. O Pandas, portanto, colabora muito com milhares de métodos e funções para tratar e estruturar dados, facilitando a vida do programador.

3.2.3 Matplotlib

Matplotlib é uma biblioteca da linguagem Python utilizada na criação de gráficos e visualização de dados. Ela é uma biblioteca bastante conhecida pela comunidade da linguagem e devido a facilidade na criação e manipulação de gráficos se tornou extremamente popular sendo utilizada inclusive por outras bibliotecas como base.

3.2.4 Seaborn

Seaborn é uma biblioteca da linguagem Python muito utilizada na visualização de dados, ela é baseada em outra biblioteca, Matplotlib, com o diferencial de apresentar gráficos de alto nível de forma mais atrativa.

3.3 ETL

ETL é o nome dado a um dos processos mais comuns de exploração e integração de dados, onde o dado atravessa algumas etapas até que possa ser utilizado em algum DataWarehouse, banco de dados tradicional ou visualizador de dados.

3.3.1 Extract

Diz respeito a parte de extração de um ETL, a obtenção do dado em si. Várias ferramentas podem estar envolvidas variando de acordo com cada processo. A forma mais comum de se ver uma extração é através de uma linguagem de programação que se conecta com uma origem ou simplesmente faz o download dessa origem, dentre as linguagens mais conhecidas que realizam essa operação o Python é de longe a mais consolidada no processo, muito devido a sua facilidade em lidar com dados e seu tratamento além da imensa gama de bibliotecas que ele possui.

3.3.2 Transform

A transformação abrange toda a etapa de limpeza e manipulação do dado, uma vez que ele pode estar em uma codificação ruim para leitura, com mais colunas do que as necessárias, não traduzido etc. A tecnologia responsável pela transformação ainda é a linguagem de programação, geralmente também Python. Ao fim desta etapa é comum a geração de um novo arquivo pronto para leitura imediata ou inserção em uma base de dados.

3.3.3 Load

Nesta etapa, de carga, o dado extraído já está pronto para ser repassado ao seu destino para então ser consultado, a linguagem de programação continua atuando ao abrir conexão com bancos de dados para então fazer a inserção dos dados no banco, ou agindo para fazer upload do arquivo em alguma plataforma que disponibilize a leitura/consulta.

3.4 Databricks

Tem crescido no mercado de ciência de dados o número de ferramentas com a capacidade de sintetizar tudo que o ramo precisa, são conhecidas como ferramentas de ETL ou Ambiente de Data Science, dentre eles tem se destacado nos últimos anos o Databricks. O databricks é uma ferramenta completa voltada para cientistas e engenheiros de dados desenvolvida pelos mesmos criados do Apache Spark. Cada vez mais utilizada nas grandes empresas o Databricks entrega agilidade, poder de computação e facilidade na hora de realizar as principais tarefas como extrair dados e plotar gráficos.

O Databricks tem seu funcionamento bem parecido com a ferramenta já conhecida, o Jupyter Notebook. Através de notebooks é possível escrever blocos de código que são executados de forma independente, com o diferencial que o Databricks suporta a escrita de diversas linguagens de programação dentro do mesmo notebook, possibilitando a sinergia entre Python, SQL, Scala, Shell Script e Markdown.

Outro grande ponto positivo dentro do Databricks é sua grande escalabilidade devido ao uso da engine Spark de forma nativa. Através do Spark o Databricks consegue realizar um processamento paralelo, o que acelera o desenvolvimento e programação já que grandes massas de dados podem ser lidas de forma paralela rapidamente.

# Estudo de caso

Os códigos utilizados em todo o processo de ETL estão disponíveis no repositório: *https://github.com/yuri-scaranni/Analise\_covid.git*

4.1 Integração com ETL

A arquitetura de dados baseada no modelo clássico de ETL tem por objetivo dividir em camadas as etapas bem definidas o ciclo de vida do dado. É uma das formas mais tradicionais de se trabalhar com dados por ser efetivo, simples e claro quando se faz necessário uma revisão dos passos ou correção de erros.

4.1.1 Extração dos dados

Um processo de ETL bem estruturado garante uma boa massa de dados, que por sua vez garantem uma análise mais real e melhor, este tipo de processo é mais utilizado para inserção dos dados em uma base para futuras análises. O primeiro passo neste tipo de integração é analisar bem a fonte dos dados e escolher a melhor forma de extrai-lo. Uma das formas mais comuns, e que utilizaremos neste trabalho, é a extração de arquivos direto da Web, para isso utilizaremos a linguagem Python com a biblioteca requests, instalada através do comando pip install requests,e a extração será dos dados da covid-19 disponibilizados diariamente em arquivo csv pelo site *ourworldindata.org*, um site que disponibiliza dados empíricos sobre todo tipo de assunto e é mantido sob a tutela da Universidade de Oxford.

Código fonte 20 – Implementação extração covid-19 ourworldindata.

import requests

name\_dataset = 'covid19.csv'

source\_url = "https://covid.ourworldindata.org/data/owid-covid-data.csv"

print('>>>> Extração iniciada <<<<')

data = requests.get(source\_url)

if data.status\_code != 200:

print('Erro ao extrair!')

with open(f'download/{name\_dataset}', 'w', encoding='utf-8') as f:

f.write(data.text)

print('>>>> Extração finalizada <<<<')

Após a extração podemos ver o arquivo salvo na pasta ‘download’ do nosso projeto como mostra a Figura 2.

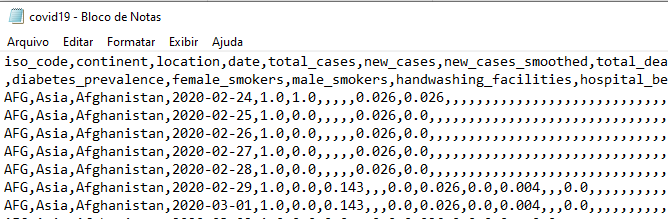


Figura 2 – Arquivo csv dos dados da covid19.

4.1.2 Transformação

A transformação dos dados é a etapa na qual devemos ter em mente aquilo que pretendemos carregar para nossa futura análise, alguns dados precisam ser tratados, traduzidos ou até removidos. Uma das ferramentas mais utilizadas para isso é a biblioteca Pandas do Python. Com a capacidade de transformar dados em uma estrutura organizada, denominado Dataframe, o Pandas, que pode ser instalado através do comando pip install pandas, é um aliado fantástico e leve para o tratamento de dados. Faremos alguns tratamentos nestes dados como traduzir o nome dos campos e excluir algumas colunas que não farão parte da análise, mantendo apenas as mais relevantes.

import pandas as pd

campos\_nome = {

"iso\_code":"codigo\_iso",

"continent":"continente",

"location":"pais",

"date":"data",

"total\_cases":"casos\_totais",

"new\_cases":"novos\_casos",

"total\_deaths":"mortes\_totais",

"new\_deaths":"novas\_mortes",

"total\_cases\_per\_million":"total\_casos\_por\_milhao",

"new\_cases\_per\_million":"novos\_casos\_por\_milhao",

"total\_deaths\_per\_million":"total\_mortes\_por\_milhao",

"new\_deaths\_per\_million":"novas\_mortes\_por\_milhao",

"reproduction\_rate":"indice\_reproducao",

"new\_tests":"novos\_testes",

"total\_tests":"total\_testes",

"total\_tests\_per\_thousand":"total\_testes\_por\_mil",

"new\_tests\_per\_thousand":"novos\_testes\_por\_mil",

"total\_vaccinations":"total\_vacinas",

"people\_vaccinated":"pessoas\_vacinadas",

"people\_fully\_vaccinated":"pessoas\_vacinadas\_dose\_completa",

"new\_vaccinations":"novas\_vacinas\_aplicadas",

"total\_vaccinations\_per\_hundred":"total\_vacinacao\_por\_centena",

"people\_vaccinated\_per\_hundred":"pessoas\_vacinadas\_por\_centena",

"people\_fully\_vaccinated\_per\_hundred":"pessoas\_vacinadas\_dose\_completa\_por\_centena",

"population":"populacao",

"population\_density":"densidade\_populacional",

"median\_age":"media\_idade",

"aged\_65\_older":"idade\_acima\_65",

"aged\_70\_older":"idade\_acima\_70",

"gdp\_per\_capita":"pib\_per\_capita",

"extreme\_poverty":"pobreza\_extrema",

"life\_expectancy":"expectativa\_vida"

}

df = pd.read\_csv(f'download/covid19.csv', sep=',', encoding='utf-8')

df = df.filter(campos\_nome.keys()) # Filtrando colunas

df = df.rename(columns=campos\_nome) # Renomeando colunas

df.to\_csv('download/covid19\_renamed.csv', sep=',', index=False)

Código fonte 21 – Renomeando campos e excluindo colunas com Pandas.

4.1.3 Carga

Após os dados estarem extraídos e tratados é o momento de inseri-los no banco de dados. O banco escolhido aqui é o MySQL, utilizado através do servidor local criado pela ferramenta WAMP server. A interface com os dados será feita pelo phpMyAdmin também disponibilizado pelo WAMP. Para efetuar a carga precisamos antes criar a base de dados e então através do Pandas iremos inserir os dados do arquivo em uma tabela gerada também pelo Pandas.

Então iremos baixar e executar o WAMP server, disponível em *wampserver.com/en/*, como mostra a Figura 3.

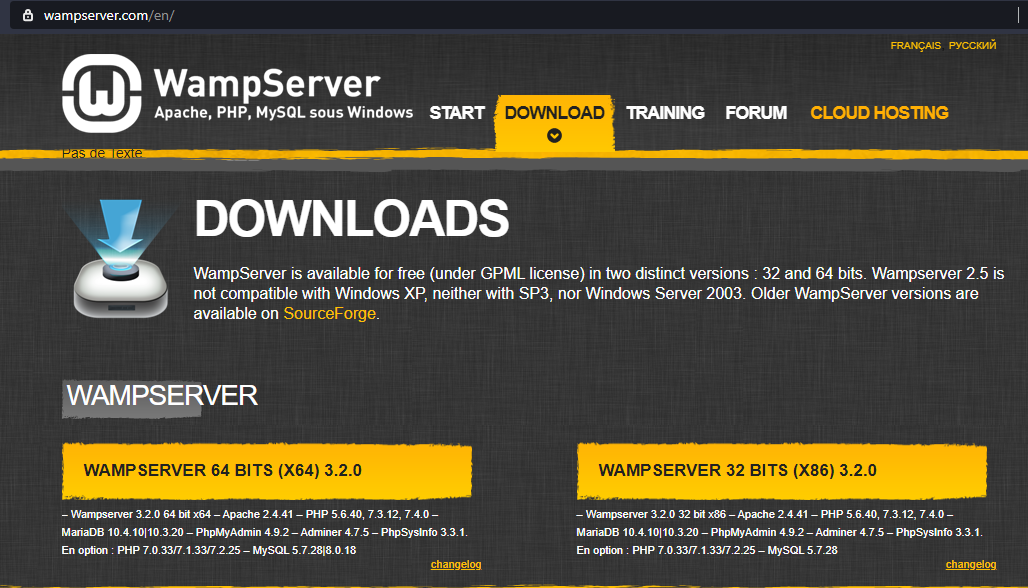


Figura 3 – Download ferramenta WAMP server.

Terminado o download e a instalação, o WAMP server, figura 4, está pronto para administrar nosso servidor local de MySQL.

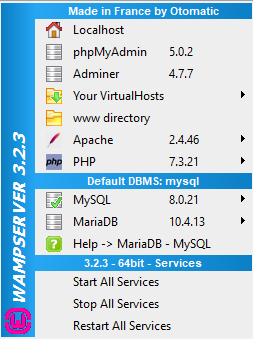


Figura 4 – WAMP server em execução na porta localhost.

Agora podemos acessar pelo browser a URL *localhost/phpmyadmin/* e fazer login na base de dados MySQL. Por padrão o login de acesso é ***root*** e o password em branco. Na aba superior é possível ver as opções gerais de administração e escrita de SQL e a esquerda as bases de dados existentes, como na figura 5.



Figura 5 – Página inicial do phpMyAdmin.

Na parte superior em SQL, podemos escrever nosso primeiro código SQL para criar uma base de dados que abrigará nossa tabela com os dados que geramos nos passos de extração e transformação.

CREATE DATABASE IF NOT EXISTS analise\_covid;

Código fonte 22 – Criando base de dados.

Com a base de dados criada é possível agora executar o script Python que realiza a inserção dos dados. A inserção é feita utilizando duas bibliotecas, Sqlalchemy, instalada através do comando pip install sqlalchemy, e a biblioteca PyMySQL, instalada através do comando pip install pymysql.

from sqlalchemy import create\_engine

import pandas as pd

# Dados de conexão local

conn = {'host': 'localhost', 'user': 'root', 'pass': '', 'database':'analise\_covid'}

# Criando conexão

engine = create\_engine(f'mysql+pymysql://{conn["user"]}:{conn["pass"]}@{conn["host"]}/{conn["database"]}')

# Abrindo arquivo de dados transformados

data = pd.read\_csv('download/covid19\_renamed.csv', sep=',', encoding='utf-8', index\_col=False)

# Insere os dados

data.to\_sql(name='covid\_19', con=engine, schema=conn["database"], if\_exists='replace', index=False)

Código fonte 23 – Inserindo arquivo em uma tabela no banco de dados.

Com a carga dos dados feita é possível realizar consultas SQL e analisá-los através do phpMyAdmin, como mostra a figura 6.

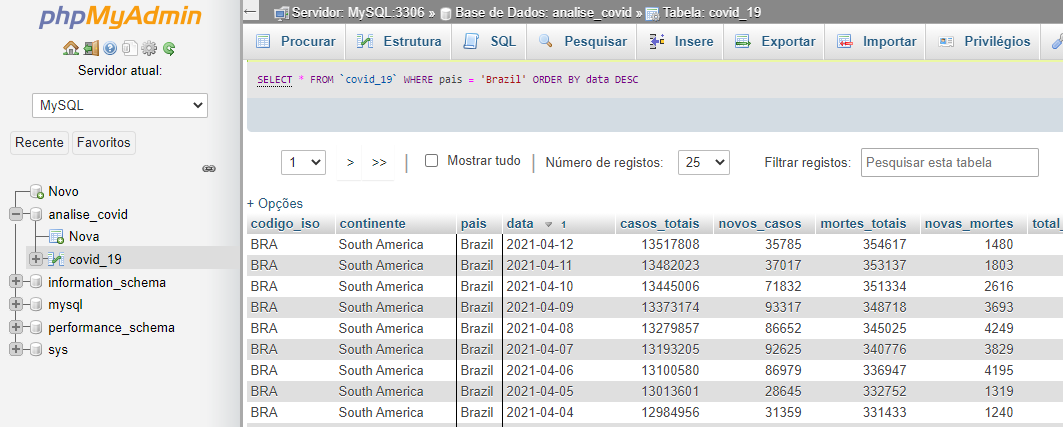


Figura 6 – Consulta a tabela criada dentro do banco de dados.

4.2 Análise de dados via cliente SQL

A análise dos dados inseridos em um banco de dados é extremamente comum e vital em profissões ligadas a ciência de dados e para isso é necessário a utilização da linguagem SQL para que seja possível obter algumas análises interessantes em cima dos dados. Agora que temos nossos dados dentro da base de dados MySQL, o que precisamos fazer é utilizar o editor de texto SQL do PhpMyAdmin para realizar nossas consultas, ou querys.

Para começar, iremos até nosso PhpMyAdmin e iremos explorar os dados que lá estão, abrindo o editor de SQL e inserindo a query:

SELECT

data

,pais

,casos\_totais

,mortes\_totais

FROM

analise\_covid.covid\_19

WHERE

data = (CURRENT\_DATE - INTERVAL(1) DAY)

AND continente IS NOT NULL

ORDER BY 3 DESC

LIMIT 10;

Código fonte 24 – Query para top 10 países no ranking de casos por covid.

Esta query trará como resultado os países com maior número de casos e mortos por covid no até um dia antes da data atual, se limitando apenas a países e ignorando continentes e ordenando pelo número de casos como indica a figura 7.

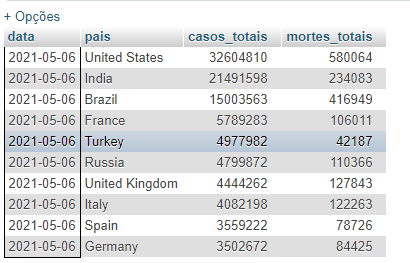


Figura 7 – Retorno da consulta feita sobre a tabela covid19.

Uma análise através do banco nos dá de forma simples e rápida valores exatos e uma noção mais realista dos dados, nesta query a seguir faremos uma análise temporal, olhando a média de novos casos de covid em cada região(continente) ignorando os valores globais (World e International).

SELECT

DATE\_FORMAT(data,'%Y-%m') AS Data,

pais AS Pais,

ROUND(AVG(novos\_casos), 2) AS media\_novos\_casos\_por\_continente

FROM analise\_covid.covid\_19

WHERE continente IS NULL AND pais NOT IN ('World', 'International')

GROUP BY 2,1 ORDER BY 1 DESC

Código fonte 25 – Query de média de novos casos por continente.

O que podemos notar a partir dos dados é que atualmente a região mais controlada é a Oceania seguida da África, com menor índice de novos casos em maio de 2021, em contrapartida, Ásia e América do Sul estão entre os piores (figura 8), o que vai diretamente de encontro com o que acompanhamos no mundo atualmente, onde os EUA, embora líderes de casos, controlaram a doença e não apresentam grande incidência de novos casos enquanto Brasil e Índia que completam o pódio elevam os números devido a situação não controlada que os dois países vivenciam.

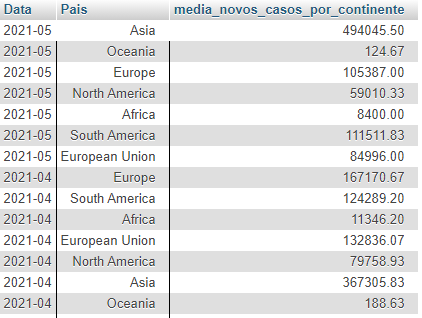


Figura 8 – Média de novos casos mensal por região (continente).

Este é apenas um pequeno exemplo do tipo de insight que olhar os dados de forma direta traz, e a grande vantagem por trás desse método de análise é a praticidade que a linguagem SQL trás, poder manipular dados de diversas formas e com inúmeras funções torna a análise numérica simples e direta.

4.3 Análise de dados gráfica com Python

A análise através de gráficos dá uma dimensão mais visual dos dados que lidamos e muitas vezes permitem insights mais diretos, apenas ao olhar já é possível formular conclusões. Neste sentido o Python é de longe a melhor linguagem para realizar este tipo de trabalho pois a linguagem possui muitas bibliotecas de visualização extremamente poderosas como o Seaborn, Matplotlib entre outras.

As análises gráficas que se seguem neste tópico fazem uso do arquivo csv com dados sobre a covid19 disponibilizado pela ourworldindata.org já previamente extraído e transformado no tópico anterior. Para isso utilizaremos as seguintes bibliotecas.

import seaborn as sns

from datetime import timedelta

import pandas as pd

import matplotlib.pyplot as plt

Código fonte 26 – Importação de bibliotecas python.

Mortes totais:

Nesta análise iremos olhar dentro da nossa massa de dados por colunas que representem o índice de mortes totais. Organizando pela quantidade de óbitos totais, para uma demonstração mais visual usaremos os primeiros 50 resultados.

def mortes\_covid\_top\_50():

# Abrindo arquivo e estabelecendo 'data' como tipo datetime

df = pd.read\_csv('download/covid19\_renamed.csv', sep=',')

df['data'] = pd.to\_datetime(df['data'])

# Extraindo somente o último dia disponível

yesterday = max(df['data']) - timedelta(days=1)

df = df.loc[df['data'] == yesterday]

# Removendo linhas onde o continente

# Não está preenchido

df = df.dropna(subset=['continente'])

# Ordenando pelo campo 'mortes\_totais' e limitando

# Aos 50 primeiros registros

df = df.sort\_values(by='mortes\_totais', ascending=False).head(50)

# Plotagem do gráfico de barras

sns.set\_theme(style="darkgrid")

palette = sns.color\_palette("Wistia", 50)

sns.barplot(data=df,

x='pais',

y='mortes\_totais',

palette=palette)

plt.xlabel("Países")

plt.ylabel("Mortes totais")

plt.title("Ranking mortes covid")

plt.xticks(rotation=90)

plt.show()

Código fonte 27 – Código gráfico mortes totais por covid.

Com a execução do código fonte 7 obtemos este gráfico da figura 9.

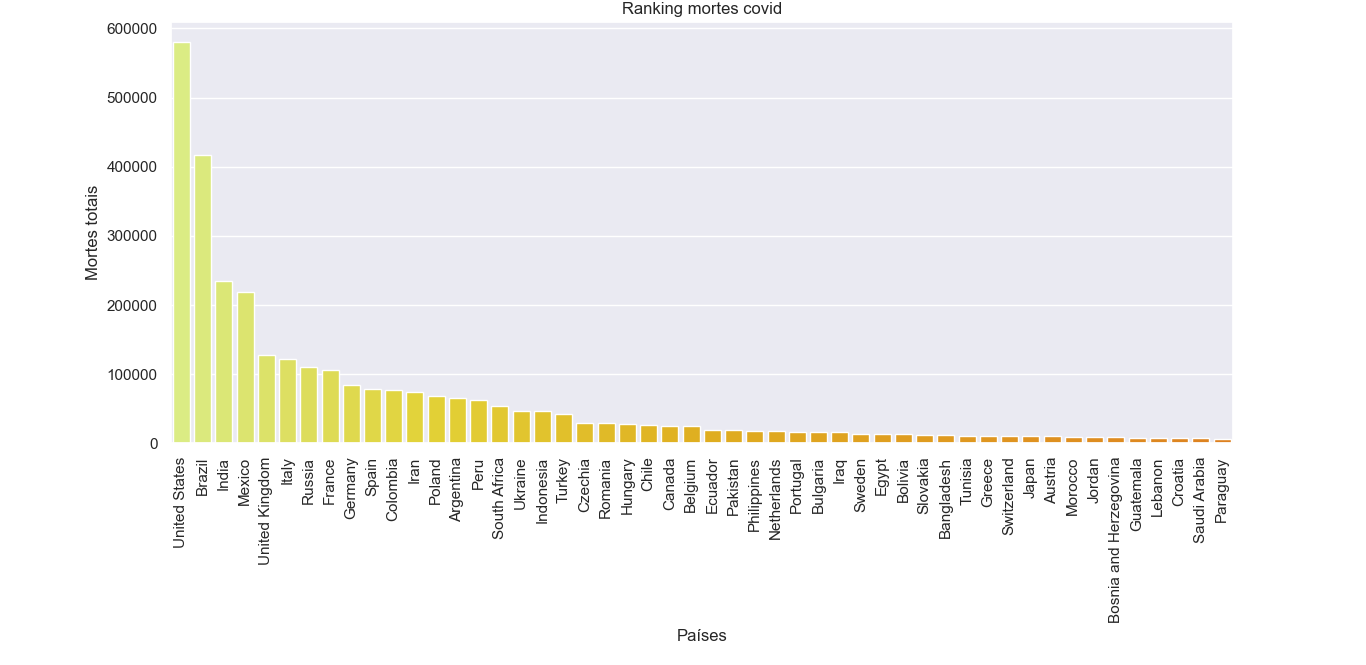


Figura 9 – Gráfico de mortes totais covid.

De forma bem clara é possível enxergar toda a situação global em um instante. E ver quais são os países mais afetados pela.

Recordistas de casos de covid:

Nesta segunda análise voltamos nossa visão para os países com maior índice de infectados pelo covid19 ao longo dos meses, utilizamos na visualização apenas 5 países para ter uma noção mais nítida das linhas.

Código fonte 28 – Código gráfico infectados covid.

def casos\_totais\_ao\_longo\_do\_tempo():

# Abrindo arquivo e estabelecendo 'data' como tipo datetime

df = pd.read\_csv('download/covid19\_renamed.csv', sep=',')

df['data'] = pd.to\_datetime(df['data'])

# Removendo linhas onde o continente

# Não está preenchido

df = df.dropna(subset=['continente'])

# Selecionando paises que até a data válida

# mais próxima eram os top 5 recordistas de casos

yesterday = max(df['data']) - timedelta(days=1)

df\_drop = df.loc[df['data'] == yesterday]

df\_drop = df\_drop.sort\_values(by='casos\_totais', ascending=False).head(5)

df\_drop = df\_drop['pais'].drop\_duplicates()

df = df[df['pais'].isin(df\_drop)].sort\_values(by='data')

# Transformando data comum em apenas ano e mês

df['data'] = df['data'].dt.to\_period('M').apply(str)

# Plotagem do gráfico de linhas

sns.set\_theme(style="whitegrid")

grap = sns.lineplot(data=df, x='data', y='casos\_totais', hue='pais')

# Criando unidade Mi(milhões) no eixo Y

from matplotlib.ticker import FuncFormatter

f = lambda x, pos: f'{x / 10 \*\* 6:,.0f} Mi'

grap.yaxis.set\_major\_formatter(FuncFormatter(f))

plt.xlabel("Ano-Mês")

plt.ylabel("Total de casos")

plt.title("Linha temporal do total de casos de covid")

plt.show()

O resultado é o gráfico da figura 10 que mostra os 5 países recordistas em número de infectados.

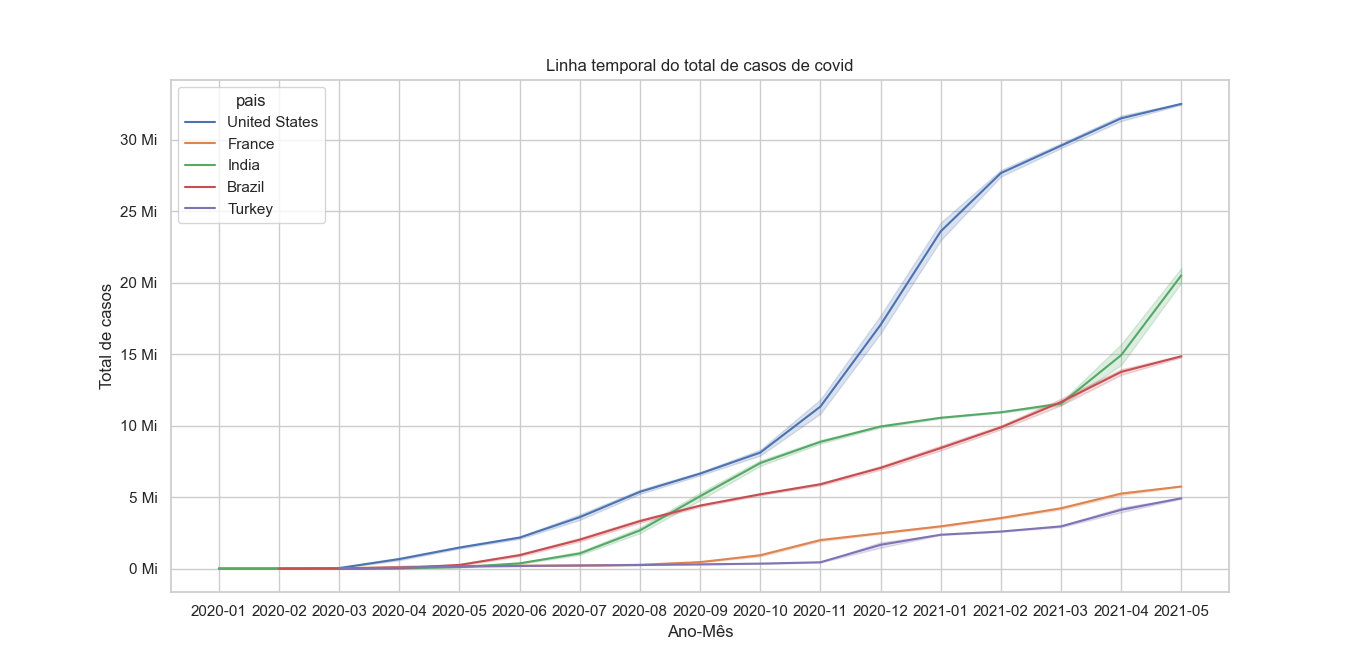


Figura 10 – Gráfico de total de infectados covid.

Um dado importante disso é que nem todos 5 primeiros no ranking de mortos estão entre os 5 primeiros no ranking de infectados. O que nos leva a elaboração de mais um gráfico para responder a este questionamento e nos levar a uma certeza, o número de infectados está ou não diretamente relacionado ao número de vítimas fatais?

Para isso iremos usar um gráfico de relação, que segue o código a seguir:

def relacao\_casos\_x\_mortes():

# Abrindo arquivo e estabelecendo 'data' como tipo datetime

df = pd.read\_csv('download/covid19\_renamed.csv', sep=',')

df['data'] = pd.to\_datetime(df['data'])

# Removendo linhas onde o continente

# Não está preenchido

df = df.dropna(subset=['continente'])

# Selecionando paises que até a data válida

# mais próxima eram os top 5 recordistas de casos

yesterday = max(df['data']) - timedelta(days=1)

df\_drop = df.loc[df['data'] == yesterday]

df\_drop = df\_drop.sort\_values(by='casos\_totais', ascending=False)

df\_drop = df\_drop['pais'].drop\_duplicates()

df = df[df['pais'].isin(df\_drop)].sort\_values(by='data')

# Filtrando colunas

yesterday = max(df['data']) - timedelta(days=1)

df = df.loc[df['data'] == yesterday]

df = df.filter(['data', 'pais', 'mortes\_totais', 'casos\_totais'])

# Plotando relação

sns.set\_theme(style="ticks")

grap = sns.scatterplot(data=df, x='mortes\_totais', y='casos\_totais')

# Criando unidade Mi(milhões) no eixo Y

from matplotlib.ticker import FuncFormatter

f = lambda x, pos: f'{x / 10 \*\* 6:,.0f} Mi'

grap.yaxis.set\_major\_formatter(FuncFormatter(f))

plt.xlabel("Total de mortos")

plt.ylabel("Total de casos")

plt.title("Relação Casos x Mortes")

plt.show()

Código fonte 29 – Código gráfico relação mortes x infectados.

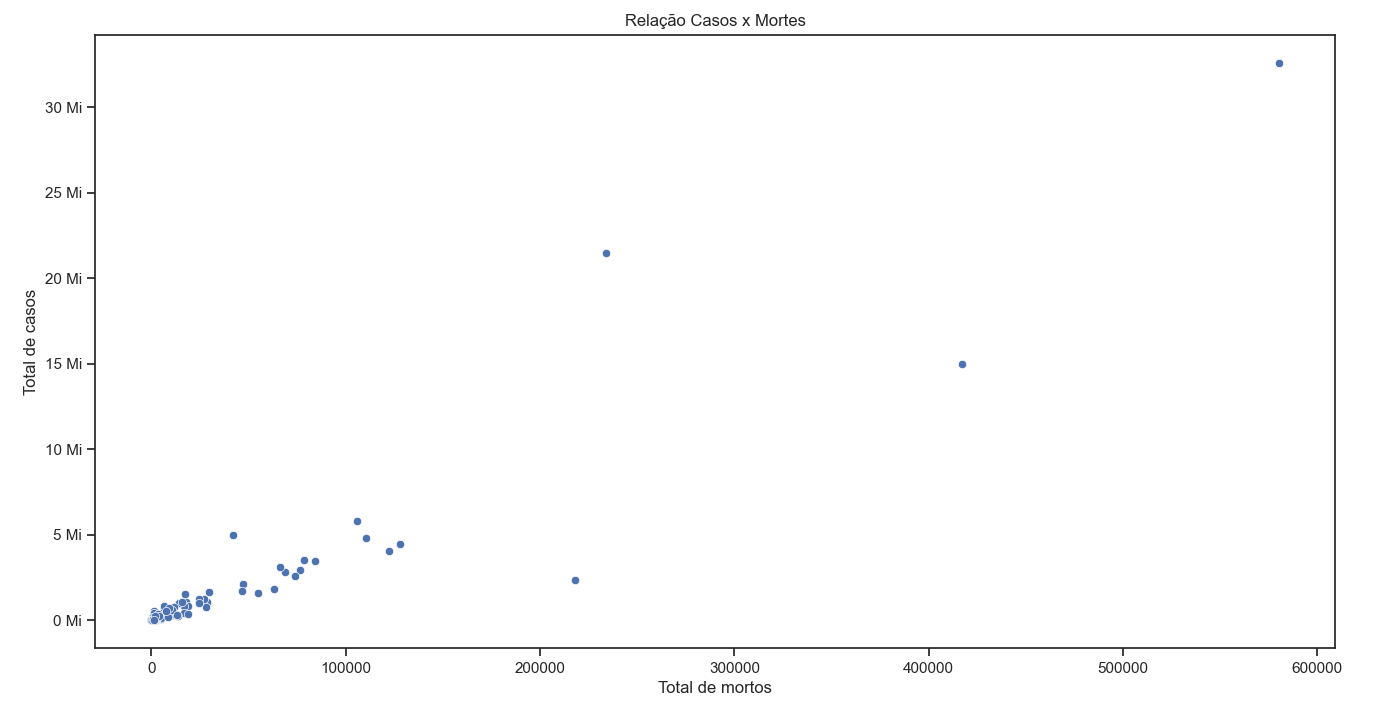


Figura 11 – Gráfico de relação óbitos x infectados.

O resultado nos mostra que, sim, há uma relação linear entre o número de infectados e o número de mortos, há alguns pontos fora da curva, mas boa parte dos países onde o número de casos é baixo, o número de mortos também é e o efeito contrário acontece nos demais países, onde quanto maior o número de infectados maior o número de óbitos como mostra a figura 11. Isso nos traz a algumas conclusões como, evitar a infecção em massa é a medida fundamental já que a relação mortos x infectados se mostrou linear, conseguimos concluir também que países com populações maiores como EUA, Índia e Brasil sofreram mais com a doença e que há pontos em comum entre estes países que sofreram mais como notícias já divulgadas em 2021 sobre atrasos na vacinação e falta de cumprimento de medidas sanitárias para evitar a contaminação.

4.4 Análise de dados com Databricks

Com o Databricks todo o trabalho desde a origem do dado até a sua visualização fica contido no mesmo lugar o que é um ponto extremamente relevante quando falamos de tecnologia, pois, torna mais simples o entendimento do pipeline do dado. Para acessá-lo é necessário criar uma conta no site [*https://databricks.com/try-databricks*](https://databricks.com/try-databricks)e a partir disso um e-mail será enviado para o usuário e então estará apto a utilizar a ferramenta.

O Databricks é uma aplicação na nuvem que cria para o usuário um cluster de dados (figura 12), dentro deste cluster são criadas bases de dados e notebooks.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Figura 12 – Cluster em funcionamento no Databricks.

Com o cluster ativo podemos começar a digitar nossos códigos no notebook e executá-los. Para criar um notebook é preciso acessar a aba lateral esquerda *Workspace* e clicar em *users*, o e-mail do usuário irá ser exibido e ao clicar na seta na direita terá a opção de criar notebook. O databricks permite que você utilize múltiplas linguagens no mesmo notebook, mas você deve optar por uma linguagem nativa para o notebook, essa linguagem será a que o notebook irá assumir à priori, caso queira fazer a transição de uma linguagem para outra, basta abrir outro bloco no notebook e digitar o comando mágico % seguido da linguagem (%sql, %python, %py, %md, %shell) como na figura 13 e executar o código através do comando Ctrl+Enter.

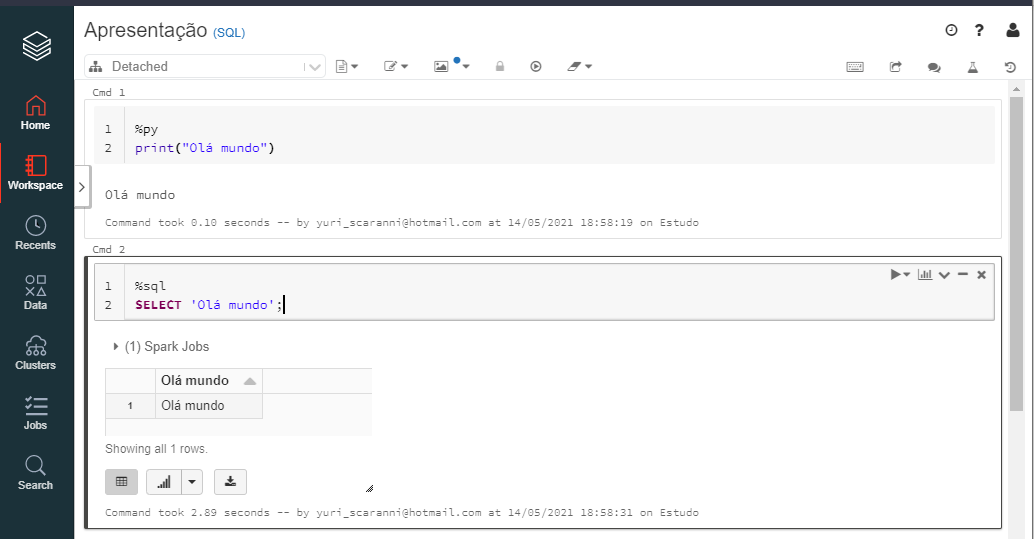


Figura 13 – Notebook simples de apresentação.

O databricks nos permite também trabalhar arquivos de forma simples, através de um upload arquivos xlsx ou csv se tornam tabelas prontas para serem manipuladas. Para isso vamos acessar a barra lateral esquerda, clicando em Data serão exibidas as bases de dados existentes (inicialmente será exibida apenas a base default) e então na parte superior clicar no botão *Create Table.* Agora basta arrastar um arquivo para o box de upload ou procurá-lo no computador como exemplifica a figura 14.

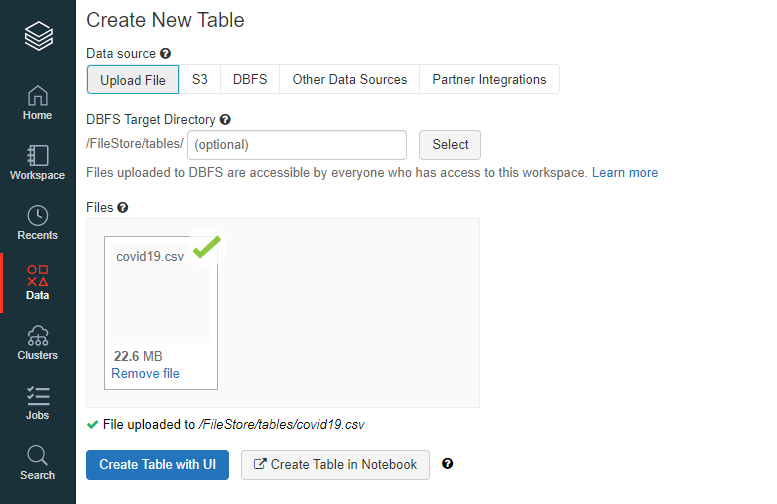


Figura 14 – Tela de upload de arquivos do databricks.

O próximo passo pode ser feito de duas formas, ao clicar em *Create Table with UI* o databricks disponibiliza uma tela para alteração de colunas, nome de tabela, entre outras opções (figura 15), ao clicar em *Create Table in Notebook* o databricks iniciará um novo notebook contendo a sequência de códigos necessária para gerar a tabela no banco de dados. Neste momento é possível definir se a primeira linha será cabeçalho, se o databricks assumirá a tipagem dos dados automaticamente dentre outras opções. Caso escolha a opção UI, basta clicar em *create table* e caso opte pela criação com notebook basta clicar no botão *Run all* na parte superior para realizar a criação da tabela.

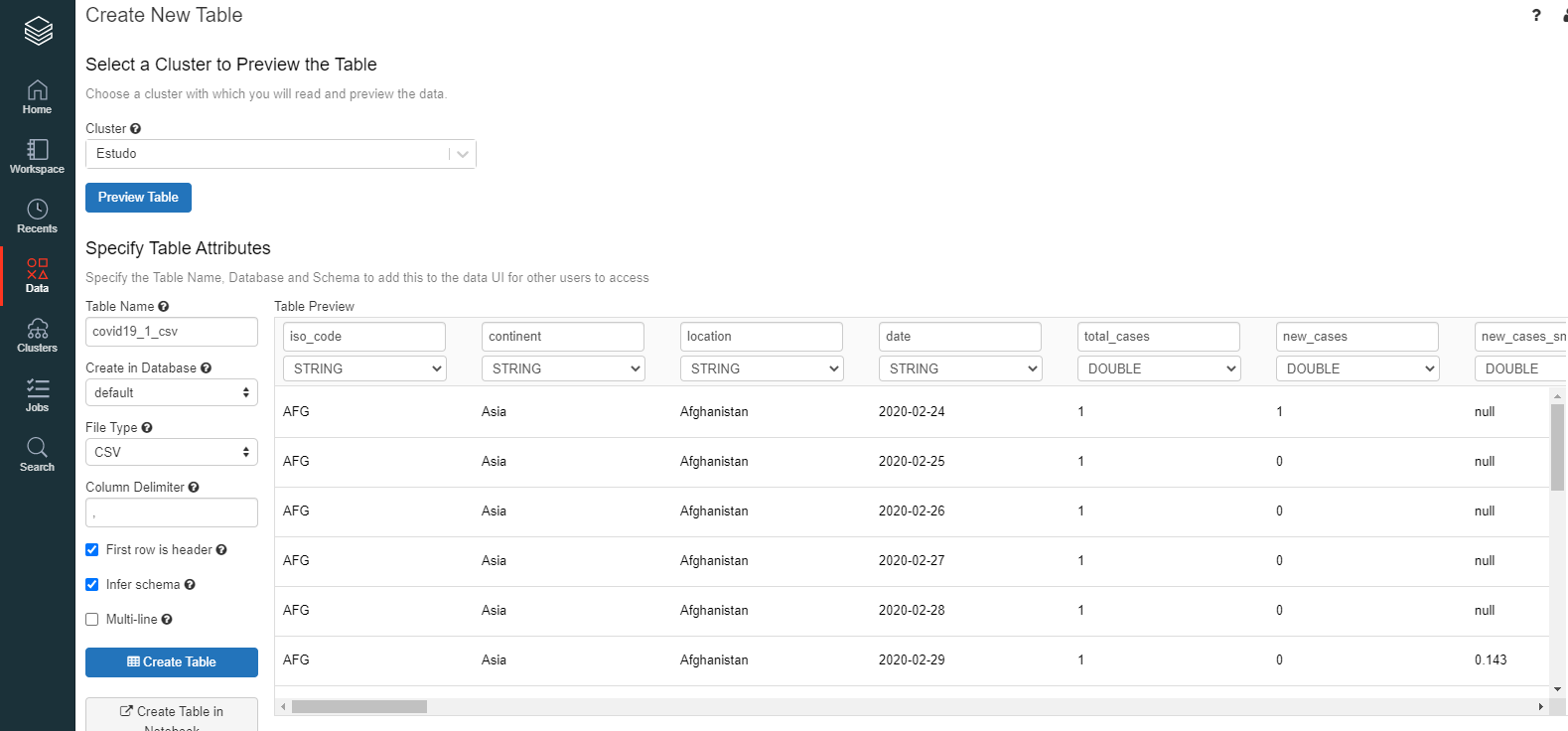


Figura 15 – Tela criação de tabela via UI.

Com o arquivo inserido e tabelas geradas, o trabalho a partir de então se resume em explorar os dados da forma que for mais conveniente para o cientista/engenheiro, seja utilizando SQL, pyspark ou pandas.

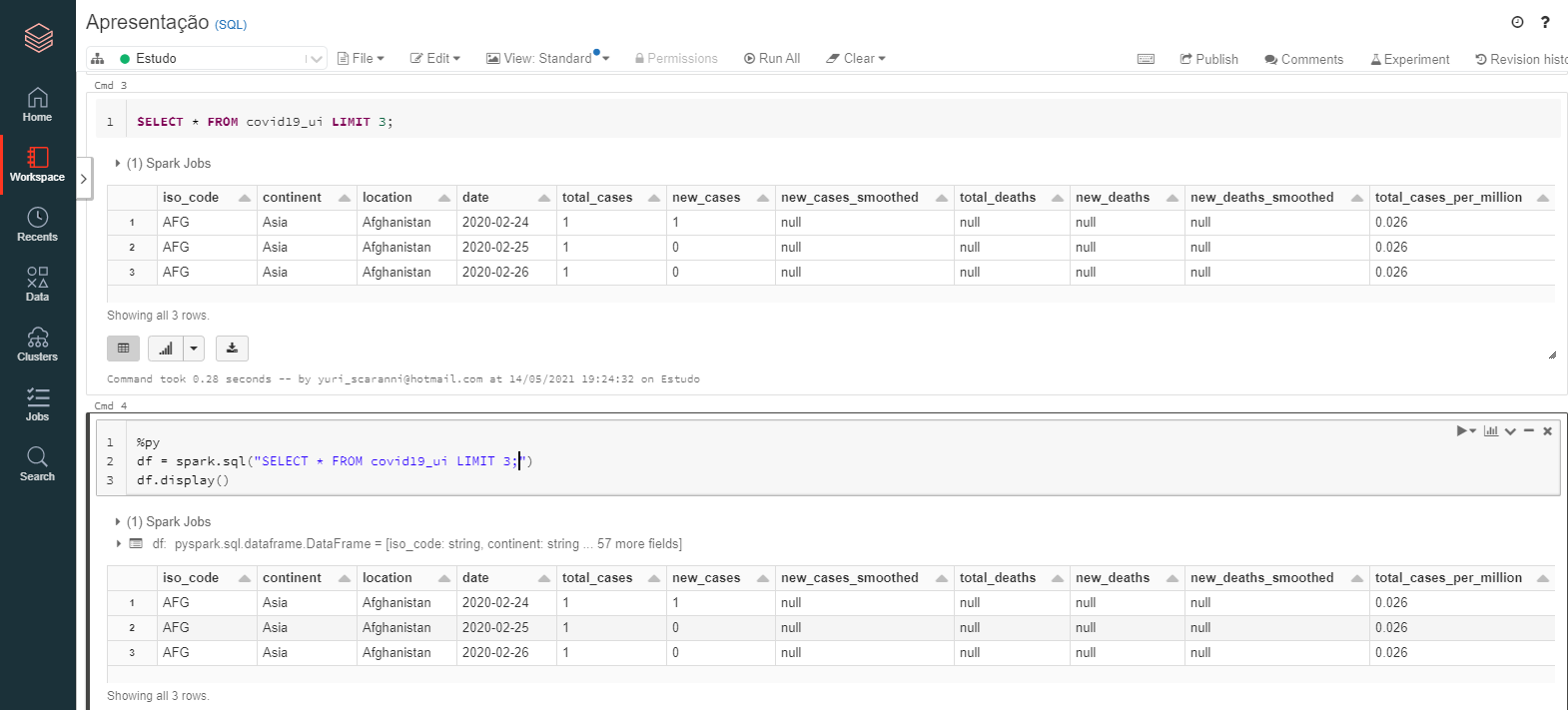


Figura 16 – Métodos de acesso aos dados do databricks.

O databricks agora está pronto para gerar informações e insights para você, abaixo (figura 16) é feita uma consulta a tabela para saber os continentes com maior índice de óbitos do mundo e com 2 comandos está pronto um gráfico na forma que o usuário bem entender. E como é possível ver, a Oceania é o continente que melhor se destaca, somando menos de 2000 mortes ao longo da pandemia, não chega a somar nem 1% do total dos óbitos ao redor do mundo.

SELECT

continent AS continente, SUM(total\_deaths) AS obitos

FROM covid\_19

WHERE date = CURRENT\_DATE - 1 AND continent IS NOT NULL

GROUP BY 1;

Código fonte 30 – Código gráfico databricks

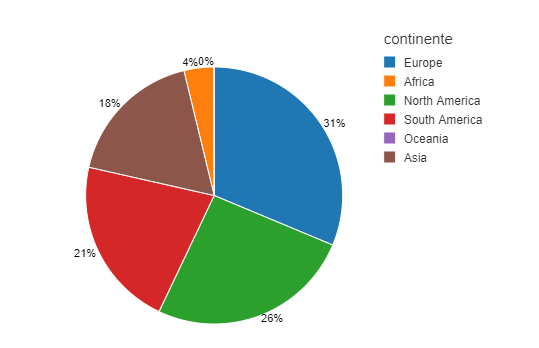


Figura 17 – Plotagem de gráfico no databricks.

Conclusão

Este trabalho teve por objetivo demonstrar de forma prática e didática a infraestrutura da ciência de dados, desde a explicação do seu surgimento até as principais ferramentas utilizadas por engenheiros e cientistas de dados.

Explicação de como foi feito

Para uma compreensão melhor do leitor, foi feita uma explicação detalhada do funcionamento de cada uma das etapas dos principais processos de dados, com ênfase principalmente no modelo de trabalho ETL que é hoje o formato mais tradicional de trabalho adotado nas grandes empresas.

Foi possível introduzir também novas ferramentas que vem se consolidando no mercado, como o Databricks, e explicar o porquê de python e SQL serem as principais linguagens de programação do ramo.

Para estudos futuros pode se continuar a análise e explicação em volta do Databricks, principalmente com a utilização de Datalakes através do Delta Databricks. Além da utilização de algoritmos de machine leaning para serem aplicados nos dados como forma de se obter análises preditivas através dos dados coletados.

# Bibliografia

**van der Aalst W. (2016) Data Science in Action. In: Process Mining. Springer, Berlin, Heidelberg.** [**https://doi.org/10.1007/978-3-662-49851-4\_1**](https://doi.org/10.1007/978-3-662-49851-4_1)

**FORBES. A VERY SHORT HISTORY OF DATA SCIENCE. DISPONÍVEL EM: https://www.forbes.com/sites/gilpress/2013/05/28/a-very-short-history-of-data-science/?sh=1e363ef455cf. ACESSO EM: 13 MAR. 2021.**

**GRUS, Joel. Data Science do Zero: Primeiras Regras Com o Python. 1. ed. Rio de Janeiro: Alta Books, 2015. p. 1-471.**

**PICHARILLO, J. et al. INTRODUÇÃO À PROGRAMAÇÃO COM PYTHON . Programa de Educação Tutorial, São Paulo. Disponível em: http://antigo.scl.ifsp.edu.br/portal/arquivos/2016.05.04\_Apostila\_Python\_-\_PET\_ADS\_S%C3%A3o\_Carlos.pdf. Acesso em: 20 mai. 2021.**