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

S277a Scaranni, Yuri Conceição de Brito

Uma abordagem didática do ecossistema da infraestrutura de ciência de dados / Yuri Conceição de Brito Scaranni.- Rio de Janeiro, 2021.

53 f.

Orientador: Sérgio Assunção Monteiro

Trabalho de Conclusão de Curso (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. 6. Databricks. I. Monteiro, Sérgio Assunção, prof. orient. II. Titulo.

CDD 005.1

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, M. Sc - Coordenador
Centro Universitário Carioca

_____

Prof. Alberto Tavares da Silva, D.Sc
Centro Universitário Carioca

AGRADECIMENTOS

Agradeço a todos que me apoiaram desde o início deste sonho de concluir um curso de engenharia em uma área que sou apaixonado, então agradeço muito aos meus amigos, em especial ao Rodrigo Rodrigues que viu tudo isso de perto, aos meus professores, ao meu orientador prof. Sergio Monteiro Assunção por todo suporte no desenvolvimento deste trabalho, a minha família e o último agradecimento, mas de uma importância enorme e muito 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 para 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 de conclusão de curso 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, Databricks.

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 undergraduate thesis 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, Databricks.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS FONTE

LISTA DE TABELAS

SUMÁRIO

1 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. Com o mundo globalizado que vivemos e o avanço da tecnologia aprendemos a explorar e descobrir segredos por trás destes dados. Das redes sociais até a posição que tal produto estará na prateleira do mercado, tudo isso passa pelas mãos dos cientistas de dados, que devem entender, armazenar, traduzir, encontrar padrões e prever comportamentos através de análises. Dados se tornaram o novo ouro, mas, para dar suporte a tal arquitetura é necessária uma grande infraestrutura e que funcione de forma coesa, simples e segura.

1.1 Objetivo

O objetivo deste trabalho é jogar luz sobre a área de ciência de dados e mostrar de forma didática e prática os primeiros passos. Explicando as ferramentas que podem ser utilizadas, formas de tratamento e persistência de dados, análises gráficas e toda a infraestrutura/arquitetura envolvida.

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 sua evolução e áreas que a compõem.

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

O quarto capítulo é um estudo de caso em que utilizamos as ferramentas descritas no capítulo três para realizar diversas análises utilizando as informações sobre o coronavírus no mundo disponibilizado pela Our World In Data.

O quinto capítulo apresenta as conclusões do trabalho.

2 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 para 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, 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 aparece.

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).

A ciência de dados cresce e evolui quando se destaca da estatística comum para ser algo além no que diz respeito a análise de dados, como disse John Wilder Tukey, conhecido estatístico estadunidense, 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, 1962). Tukey menciona também que a análise de dados e as partes da estatística 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 dar relevância ao tema (FORBES, 2013).

Passados alguns anos, já em 1974 Peter Naur (astrônomo e cientista da computação dinamarquês vencedor do concurso 2005 A.M. Prêmio Turing) 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. 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"(NAUR, 1974).

Tal como a ciência da computação, que 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.

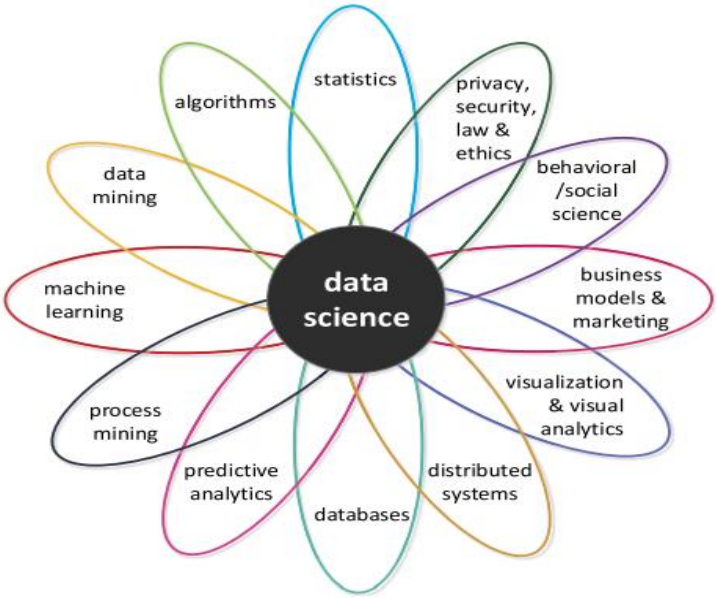

Figura 1 – Disciplinas envolvidas na ciência de dados retirado do artigo Data Science in Action.

Dentre as disciplinas contidas na ciência de dados a estatística é enxergada muitas vezes como a figura principal, trazendo para o 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), esta divisão entre a estatística 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 áreas envolvidas são 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 também estão diretamente envolvidos pois a ciência de dados trata da transformação de dados em valor, incluindo valor comercial. As ciências sociais e comportamentais também fazem parte 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 (VAN DER AALST W. 2016).

Hoje podemos notar no nosso dia a dia a quão necessária e poderosa pode ser a ciência de dados. Um exemplo disto ocorreu 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.

Ciência de dados é um ramo em pleno crescimento, com aplicação em quase tudo que conhecemos e as profissões de engenheiro de dados e cientista de dados são tidas como as profissões do futuro, sendo que hoje passamos por uma escassez de profissionais no mercado para atender a tamanha demanda.

3 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. Neste modelo (existem outros modelos de bancos de dados, como o NoSQL), os dados são vistos como tabelas cuja suas operações são feitas baseadas em alguma relação algébrica.

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) e apagar (DELETE) registros, 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 chave 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 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. O grupo DQL (Select) possui diversas cláusulas

disponíveis para realizar consultas como FROM, WHERE, GROUP BY, HAVING, ORDER BY, DISTINT, possui também operadores lógicos AND, OR, NOT, Operadores de Comparação <, >, <>, <=, =, >=, BETWEEIN, LIKE e funções de agregação AVG, COUNT, SUM, MIN, MAX.

Logo abaixo, no código fonte 1 é possível observar exemplos de comandos Select. No comando 1 obtemos o retorno da tabela com todos os registros e colunas, no comando 2 retornamos apenas as colunas/campos que queremos, logo, retornará apenas campo_1 e campo_2, no comando 3 realizamos um filtro na tabela e só retornamos registros que correspondem a cláusula passada, no quarto comando é feita uma contagem de registros agrupados pelo campo_1, ou seja, contará a quantidade de linhas para cada valor distinto no campo_1, no comando 5 é retornado a soma de todos os valores do campo_1 e por fim, no comando 6 remos o campo_2 quando campo_1 é igual a 1 e o campo_2 tem algum registro que inicie com 'Yur'.

```
/* 1 */
SELECT * FROM database.tabela ;

/* 2 */
SELECT campo_1, campo_2 FROM database.tabela;

/* 3 */
SELECT * FROM database.tabela WHERE campo_1 = 1;

/* 4 */
SELECT campo_1, COUNT(*) FROM database.tabela GROUP BY 1;

/* 5 */
SELECT campo_1, SUM(*) FROM database.tabela GROUP BY 1;

/* 6 */
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.

No código fonte 2 temos exemplos dos demais comandos que compõem o grupo DML/CRUD, realizando inserção, atualização e deleção dos dados.

```
INSERT INTO database.tabela (campo_1, campo_2) VALUES ('1', 'Yuri');

UPDATE database.tabela SET campo_1 = 2;

DELETE FROM database.tabela WHERE campo_1 = 2;
```

Código fonte 2 – Exemplo de demais comandos do grupo CRUD.

No código fonte 3 exemplifico a utilização de comandos DDL, que tem por intenção realizar alterações na tabela no nível estrutural, além de criar e deletar tabelas.

```
CREATE TABLE database.tabela_nova;

DROP TABLE database.tabela_nova;

ALTER TABLE database.tabela RENAME TO tabela_2;
```

Código fonte 3 – Exemplo de código DDL.

Por fim, no código fonte 4 temos exemplos de comandos DCL que mexem com os níveis e tipos de permissão em bases de dados.

```
GRANT SELECT ON database.tabela TO usuario;

REVOKE SELECT ON database.tabela FROM usuário;
```

Código fonte 4 – Exemplo de código DCL.

3.1.2 Join

Joins são comandos utilizados em SQL para combinação de tabelas, na figura 2 estão todos os exemplos de join utilizados normalmente. A lógica por trás do join tem a intenção de unir as tabelas por dados em comum entre elas, podendo ser divididos em alguns tipos, o comando INNER JOIN por exemplo une as tabelas e seleciona os dados quando há interseção, ou seja, quando o dado comparado existe nas duas tabelas que estão sendo unidas.

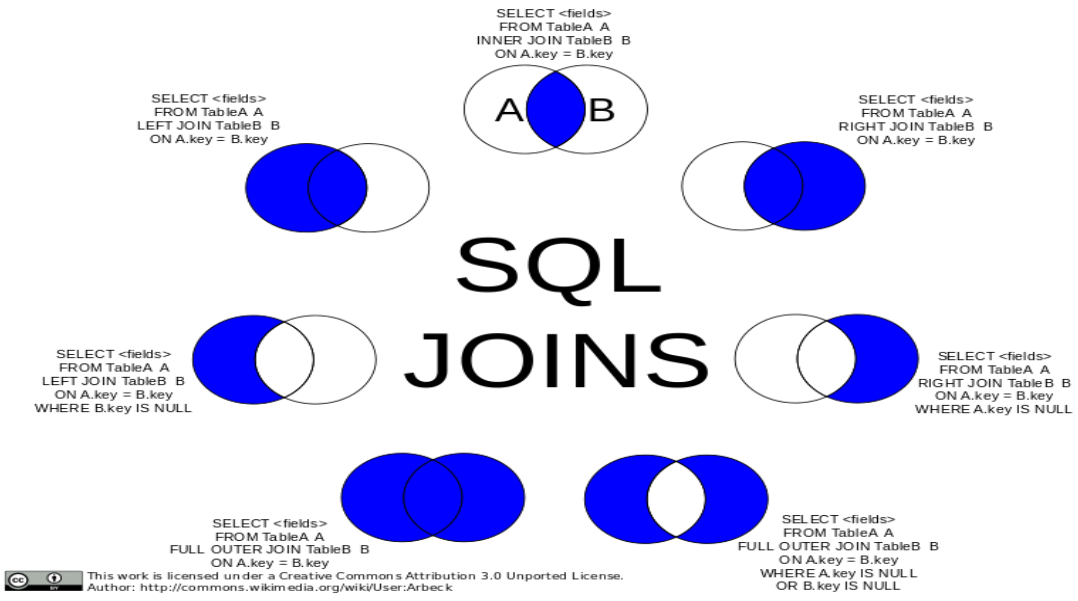

Figura 2 – Tipos de Join em SQL extraído de https://commons.wikimedia.org/wiki/File:SQL_Joins.svg.

Já o LEFT JOIN, por exemplo, poderia ser usado para unir duas tabelas A e B contando somente que o dado em A precisa existir, ou seja, caso na tabela B o campo chave que buscamos seja nulo, retornaremos o registro de A e o que seria o registro de B virá nulo.

3.1.3 Chaves

O conceito de chave é utilizado na busca de um item, ou seja, um dado é utilizado como referência na consulta no banco de dados. É um conceito lógico que só faz sentido para a aplicação e não existe fisicamente no banco de dados (SIQUEIRA, on-line).

Podemos dividir o conceito de chave em 4 partes, primária (PK), secundária, candidata e estrangeira (FK).

A chave primária é utilizada para identificar unicamente um registro de uma tabela, ela pode possuir vários atributos ou ser um único atributo. Um exemplo simples é o nosso CPF, que é uma chave primária em uma tabela de pessoas pois cada pessoa só tem 1 CPF e 1 CPF só pode pertencer a uma pessoa.

A chave secundária segue um princípio parecido com o da chave primária, com o diferencial de que ela não identifica de forma única apenas um registro e sim um conjunto de registros, como um CEP por exemplo, ele apesar de ser um dado único (não há CEP igual) ao mesmo tempo ele não é atrelado a uma residência só e sim a um conjunto.

Uma chave candidata indica uma chave que não é primária, porém, ela é identificadora tal qual a primária, por exemplo, se temos em uma tabela as colunas Matrícula e RG, a chave desta tabela é Matrícula, pois ele é um número único gerado automaticamente e para uso interno, porém, RG também é um dado único, então RG se torna uma chave candidata.

A chave estrangeira nada mais é do que um atributo que é chave primária em outra tabela, servindo assim para estabelecer relacionamentos entre as tabelas de um banco de dados. Um exemplo, podemos ter na tabela FUNCIONARIOS uma chave estrangeira COD_DEPTO que estabelece um relacionamento entre a tabela FUNCIONARIOS e a tabela DEPTOS, sendo que na tabela DEPTOS a coluna COD_DEPTO é a chave primária (SIQUEIRA, on-line).

3.1.4 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.

3.1.5 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.

3.1.6 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. Python é multiparadigma, sendo possível desenvolver de forma funcional, orientado a objetos ou 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 início 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, 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. Python possui nativamente diversos recursos que facilitam a vida do programador como listas, tuplas, dicionário, tipos inteiros, ponto flutuante, string 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, nas tabelas 1 e 2 estão as principais funções para tratamento de strings.

Tabela 1 – Principais funções de tratamento de strings em Python (1)

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'

Fonte: PICHARILLO, J. et al.

Tabela 2 – Principais funções de tratamento de strings em Python (2)

Método	Descrição	Exemplo
upper()	Retorna uma cópia da string trocando todas as letras para maiúsculo.	j = "Python" j.upper() '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 "
swapcase()	Inverte o conteúdo da string (Minúsculo / Maiúsculo).	k = "Python" k.swapcase() '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"

Fonte: PICHARILLO, J. et al.

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). Nas tabelas 3, 4 e 5 abaixo estão listados os operadores numéricos, de comparação e lógicos, respectivamente.

Operadores numéricos, tabela 3, em Python podem ser utilizados em tipos numéricos, mas em alguns casos, como o operador '+', podemos utilizá-lo também em strings. O operador '+' em strings é utilizado para concatenação.

Tabela 3 – Operadores numéricos

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

Fonte: PICHARILLO, J. et al.

Os operadores, tabela 4, de comparação são muito utilizados quando lidamos com condições, em comandos de tomada de decisão e laços de repetição.

Tabela 4 – Operadores de comparação

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

Fonte: PICHARILLO, J. et al.

Os operadores lógicos da tabela 5 também são muito utilizados em condições e tomadas de decisão no código.

Tabela 5 – Operadores lógicos

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

Fonte: PICHARILLO, J. et al.

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), como indicam os exemplos da tabela 6. Uma lista pode armazenar todo tipo de dado, de string a float e até outras listas, dicionários etc. Abaixo temos algumas das funções mais utilizadas em listas.

Tabela 6 – Funções de listas

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]

Fonte: PICHARILLO, J. et al.

A próxima estrutura de dados é o dicionário, exemplificado na tabela 7. Um dicionário é um conjunto de valores onde o valor é associado a uma chave de acesso e ele

é identificado pelo caractere {} (chaves). Dicionários são estruturas muito utilizadas em Python e abaixo estão algumas de suas principais funções.

Tabela 7 – Funções de dicionários

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])
popitem()	Remove a última chave-valor inserido.	A.popitem() / A = {"nome": "Python"}

Fonte: PICHARILLO, J. et al.

Por fim, a última estrutura são as TUPLAS, exemplificada no código fonte 5, uma tupla é um conjunto sequencial de valores, também indexado, e imutável, identificado pelo caractere () (parêntereses).

```
A = (10, 20, 30)
```

Código fonte 5 – 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, o código fonte 6 exemplifica a importação. Boa parte das bibliotecas que o desenvolvedor precisará já virão pré-instaladas.

```
import datetime
print(datetime.datetime.now())
```

Código fonte 6 – Exemplo de importação de bibliotecas em Python.

5) Estruturas de decisão

Python conta com uma estrutura de domada de decisão baseada em IF, ELSE e ELIF. IF indica que caso uma condição seja verdade o bloco abaixo será executado como mostra o código fonte 7, o print “estou no if” será disparado pois a condição em IF é atendida.

```
A = 10
if A >= 10:
      print(“estou no if”)
```

Código fonte 7 – Exemplo de if em Python.

ELSE indica que caso a condição de IF seja falsa, o bloco contido em ELSE será executado. No código fonte 8 temos a condição de IF sendo falsa pois A é igual a 10 e não maior que 10, logo, o bloco de código a ser executado será o que está em ELSE com o print “estou em else”.

```
A = 10
if A > 10:
      print(“estou no if”)
else:
      print(“estou em else”)
```

Código fonte 8 – Exemplo de if/else em Python.

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”. No código fonte 9 podemos observar que a primeira condição não pode ser atendida, logo ele descerá para o bloco ELIF, que por sua vez é verdadeiro, logo, o bloco de código de ELIF será executado, resultado no print “estou em elif”.

```
A = 10
if A > 10:
      print(“estou no if”)
elif A == 10:
      print(“estou em elif”)
else:
      print(“estou em else”)
```

Código fonte 9 – 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 previamente passada, como uma contagem por exemplo.

No código fonte 10, exemplificamos o uso de um while simples, onde ele realiza uma contagem que parte de contador = 0 até 9, pois a condição de parada é que contador seja menor que 10 e não igual a 10 e por fim, do lado de fora do bloco, inserimos um print apenas para indicar a chegada ao valor 10.

```
contador = 0
while contador < 10:
      print(f"ainda não cheguei a 10, estou em {contador}")
      contador += 1
print("cheguei a 10")
```

Código fonte 10 – Exemplo de while em Python.

O laço for possui um funcionamento muito parecido e é exemplificado no código fonte 11, onde desta vez indicamos o tamanho do nosso laço, que irá até 10 e sempre começando por 0.

```
contador = 10
for i in range(contador):
    print(f"ainda não cheguei a 10, estou em {i}")
print("cheguei a 10")
```

Código fonte 11 – 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. Funções, como a do código fonte 12, são extremamente úteis quando falamos de um código organizado, pois, com elas podemos utilizar o mesmo trecho de código quantas vezes quisermos.

```
def soma(a, b):
    print(f"O resultado da soma é {a + b}")
soma(1, 3)
```

Código fonte 12 – 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 (Extract, Transform e Load) é o nome dado ao processos comum 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. Através da separação de etapas que o ETL se baseia podemos identificar mais claramente as camadas (bronze, prata e ouro) envolvidas, onde no nível mais baixo, bronze, ocorre a etapa de extração e o dado está igual ou bem semelhante ao que se encontra na origem, já na segunda camada, a de transformação ou prata, o dado passa pelos primeiros métodos de limpeza e por fim na camada de ouro, equivalente a etapa de carga, ele será persistido com as regras e métodos necessários para o cliente final.

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 uma extração acontecer é utilizando uma linguagem de programação que se conecta com uma origem ou através de um download da web/sftp/ftp etc. O objetivo maior dessa etapa é ter sob o controle do desenvolvedor, em seu servidor ou máquina, uma cópia confiável da origem. Dentre as linguagens mais conhecidas que realizam essa operação o Python é de longe a mais consolidada no processo, muito devido a sua imensa gama de bibliotecas para este fim.

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, com chaves nulas 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 intermediária.

3.3.3 Load

Nesta etapa, de carga, espera-se que os dados a serem carregados já estejam limpos e prontos para uso. O processo aqui tem como foco abrir conexão com o banco de dados de destino (ou qualquer outro tipo de sistema/repositório) e gravar os dados tratados nesta base, de forma que eles possam ser consultados com total confiança.

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 e foi 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 Spark, ou Apache Spark, utilizado por baixo dos panos pelo Databricks é um mecanismo muito poderoso de processamento. Sendo código aberto e construído em torno de velocidade, facilidade de utilização, e análises sofisticadas. Ele foi originalmente desenvolvido na Universidade de Berkeley em 2009. O framework Spark é 100% open source, hospedado no Apache Software Foundation independente de fornecedor.

O Databricks tem seu funcionamento bem parecido com outra 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.

4 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 modularizar as etapas do processo, estabelecendo uma clara hierarquia e divisão em camadas ou níveis.

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, código fonte 13, para isso utilizaremos a linguagem Python com a biblioteca *requests*, instalada através do comando *pip install requests*, extrairemos 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.

```
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 <<<<')
```

Código fonte 13 – Implementação extração covid-19 ourworldindata.

Após a extração podemos ver o arquivo salvo na pasta ‘download’ do nosso projeto como mostra a Figura 3.

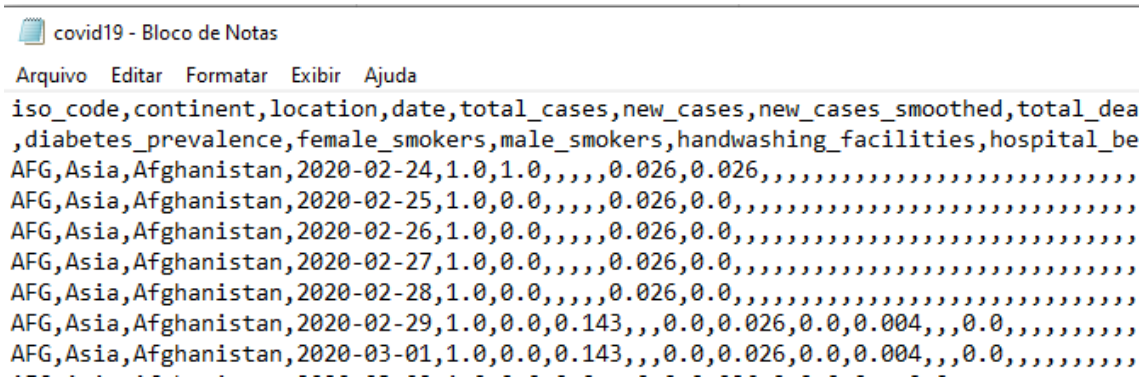

Figura 3 – Arquivo csv dos dados da covid19.

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 como mostra o código fonte 14.

```
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":"taxa_transmissao",
"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_comple
ta_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())
df = df.rename(columns=campos_nome)
df.to_csv('download/covid19_renamed.csv', sep=',', index=False)
```

Código fonte 14 – Renomeando campos e excluindo colunas com Pandas.

Com o código fonte 14 realizamos a leitura dos dados com o comando pd.read_csv e em seguida filtramos as colunas que queremos utilizando uma função filter junto de um dicionário, neste dicionário selecionamos apenas as chaves, ou seja, ele irá deixar no arquivo apenas as colunas passadas no comando, em seguida renomeamos as colunas desta vez utilizando o dict como um todo pois as chaves(key) são os valores antigos e no valor(value) estão os novos nomes. Por fim, salvamos um novo arquivo CSV renomeado.

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 4.

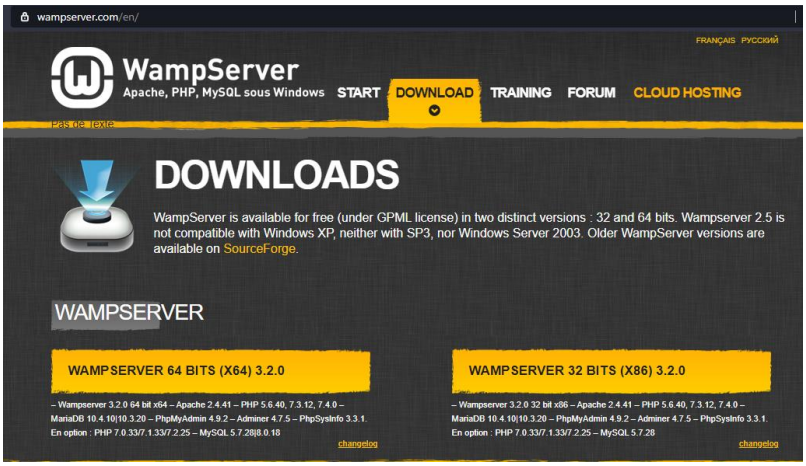

Figura 4 – Download ferramenta WAMP server.

Terminado o download e a instalação, o WAMP server, figura 5, está pronto para administrar nosso servidor local de MySQL.

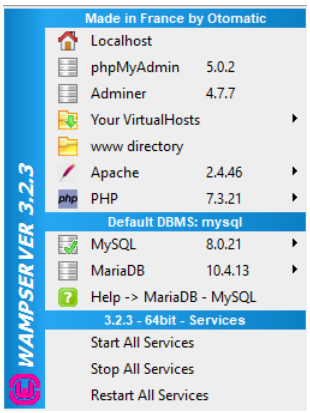

Figura 5 – WAMP server em execução na porta localhost.

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 6.

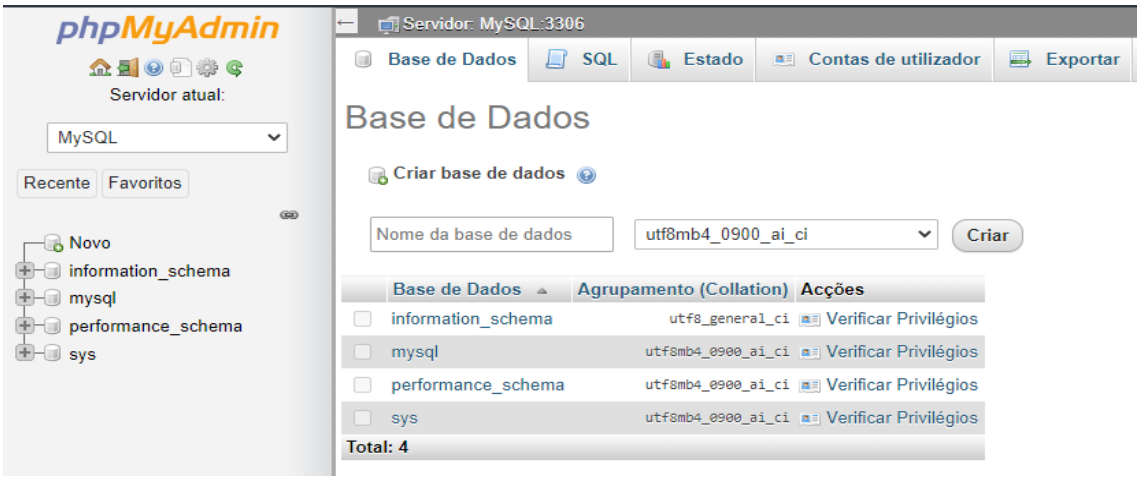

Figura 6 – Página inicial do phpMyAdmin.

Na parte superior na aba SQL, podemos escrever nosso primeiro código SQL, código fonte 15, 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 15 – 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*. O código responsável por isso está logo abaixo no código fonte 16 onde começamos importando as bibliotecas, depois passando os dados de conexão como host, usuário, password e base de dados, após isso, passamos a função que será responsável por fazer a nossa conexão, no nosso caso a função do sqlalchemy, create_engine com os devidos parâmetros e por fim abrimos nosso arquivo CSV tratado, instanciando no dataframe 'data' e inserindo na base de dados através da biblioteca do Pandas 'to_sql'.

```
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 16 – Inserindo arquivo em uma tabela no banco de dados.

Com a carga dos dados feita é possível realizar consultas SQL, código fonte 17, e analisá-los através do phpMyAdmin, como mostra a figura 7. Esta query tem por objetivo retornar os dados da tabela 'covid_19', filtrando pelo país 'Brazil' e ordenando os valores pela 'data' em ordem decrescente.

```
SELECT * FROM analise_covid.covid_19 WHERE pais = 'Brazil' ORDER BY
data DESC
```

Código fonte 17 – Inserindo arquivo em uma tabela no banco de dados.

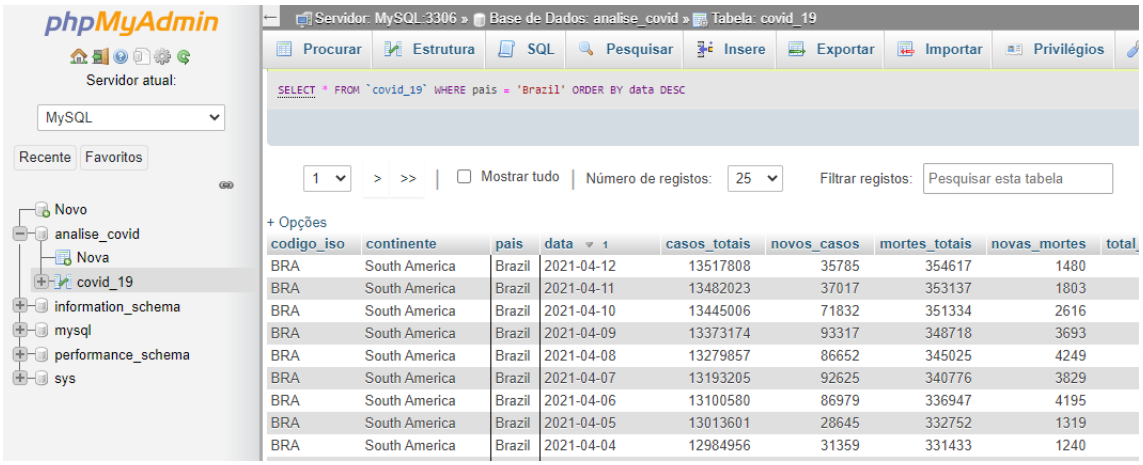

Figura 7 – Consulta a tabela criada dentro do banco de dados.

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 do código fonte 18:

```
SELECT
      data
      ,pais
      ,casos_totais
      ,mortes_totais
FROM
      analise_covid.covid_19
WHERE
      data = '2021-05-30'
      AND continente IS NOT NULL
ORDER BY 3 DESC
LIMIT 10;
```

Código fonte 18 – 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 na data 30/05/2021 e onde o campo ‘continente’ não seja nulo, ordenando pelo número de casos como indica a figura 8.

data	pais	casos_totais	mortes_totais
2021-05-30	United States	33261731	594443
2021-05-30	India	28047534	329100
2021-05-30	Brazil	16515120	461931
2021-05-30	France	5728418	109562
2021-05-30	Turkey	5242911	47405
2021-05-30	Russia	5005171	119130
2021-05-30	United Kingdom	4499937	128043
2021-05-30	Italy	4216003	126046
2021-05-30	Argentina	3753609	77456
2021-05-30	Germany	3687715	88431

Figura 8 – 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, código fonte 19, 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 Continente,
      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 19 – Query de média de novos casos por continente.

A query do código fonte 19 retorna os continentes agrupados por mês, juntamente da média de novos casos neste período e o valor é arredondado para até 2 casas decimais, isto onde o campo ‘continente’ seja nulo e o campo ‘pais’ não corresponda aos valores ‘World’ e ‘International’, ordenado pela data de forma decrescente.

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 9);

2021-05	Asia	378695.39
2021-05	North America	44107.16
2021-05	Africa	9121.32
2021-05	South America	126086.03
2021-05	Oceania	180.74
2021-05	European Union	46053.13
2021-05	Europe	65294.10
2021-04	North America	79791.27
2021-04	Asia	367305.83
2021-04	Europe	167170.67
2021-04	Oceania	188.63
2021-04	South America	124289.20

Figura 9 – Média de novos casos mensal por região (continente).

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 a seguir 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 gerar nossos gráficos utilizaremos as seguintes bibliotecas (código fonte 20).

```
import seaborn as sns
from datetime import timedelta
import pandas as pd
import matplotlib.pyplot as plt
```

Código fonte 20 – Importação de bibliotecas python.

Primeiro analisaremos o número total de óbitos. Nesta análise iremos olhar dentro da nossa massa colunas que representem o número de mortes totais. Organizando pela quantidade de óbitos totais, para uma demonstração mais visual nos limitaremos a observar os 30 primeiros países. No código fonte 21 está o código utilizado para esta visualização, nele buscamos primeiro pelo nosso arquivo tratado e em seguida buscamos a maior data presente no arquivo e subtraímos 1 (um) dia, isto é feito pois nem todos os dados obtidos pelo ourworldindata são captados no mesmo instante, logo, buscamos um dia anterior para que todos os países estejam com seus dados registrados.

```
def mortes_covid_top_30():
    df = pd.read_csv('download/covid19_renamed.csv', sep=',')
    df['data'] = pd.to_datetime(df['data'])
    df = df.dropna(subset=['continente'])
    yesterday = max(df['data']) - timedelta(days=1)
    df = df.loc[df['data'] == yesterday]
    df = df.sort_values(by='mortes_totais',
ascending=False).head(30)
    sns.set_theme(style="darkgrid")
    palette = sns.color_palette("Wistia", 30)
    sns.barplot(data=df,
                x='mortes_totais',
                y='pais',
                orient='h',
                palette=palette)
    plt.xlabel("Óbitos totais")
    plt.ylabel("Países")
    plt.title("Ranking óbitos covid")
    plt.xticks(rotation=90)
    plt.show()
```

Código fonte 21 – Código gráfico mortes totais por covid.

Em seguida ordenamos os dados pelo campo ‘mortes_totais’ de forma decrescente e limitamos aos 30 primeiros, fixamos o estilo de fundo para ser escuro, escolhemos a paleta de cores, conjunto “Wistia” em 30 tons distintos, instanciamos um gráfico de barras utilizando como dataset o dataframe ‘df’ onde os eixos x e y são ‘mortes_totais’ e ‘pais’ respectivamente, orientamos o gráfico para horizontal e por fim colocamos os nomes nos eixos, título e alteramos os dados do eixo x para uma rotação de 90º. O resultado pode ser visto na figura 10.

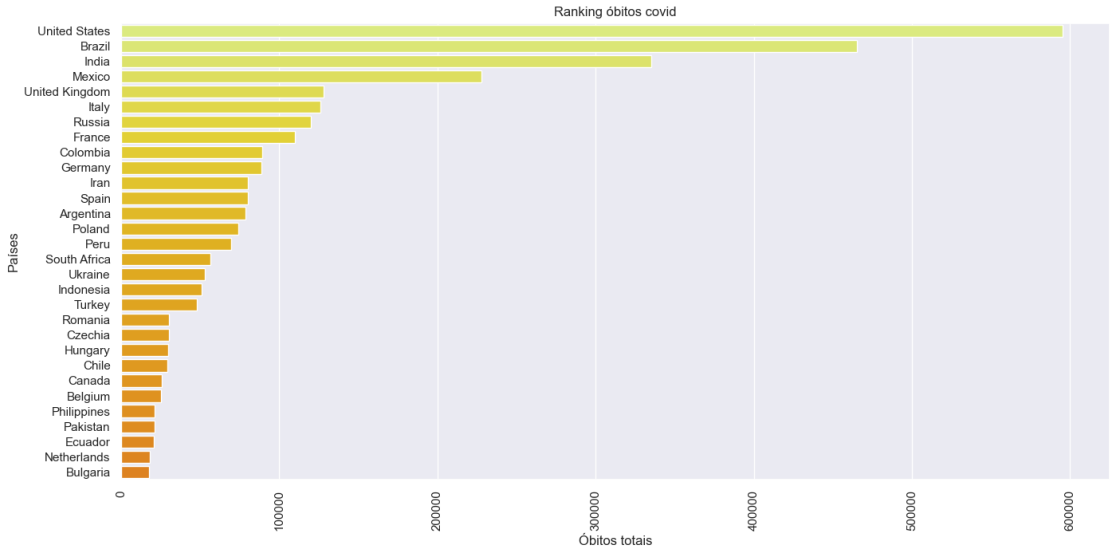

Figura 10 – Gráfico de mortes totais covid.

A primeira observação que pode ser feita com o gráfico da figura 10 em mãos é que os 3 países com maior número de óbitos são países com uma população grande. Brasil, EUA e Índia têm atualmente 212mi, 331mi, 1.380mi de habitantes respectivamente. Prosseguindo com as análises, vamos nos concentrar nos 5 países (EUA, Brasil, Índia, Mexico e Reino Unido) com maior número de óbitos e observá-los em outros aspectos. O primeiro ponto que podemos observar nesse conjunto são os números de infectados, para isso utilizaremos o código fonte 22.

```
def casos_totais_ao_longo_do_tempo():
    df = pd.read_csv('download/covid19_renamed.csv', sep=',')
    df['data'] = pd.to_datetime(df['data'])
    df = df.dropna(subset=['continente'])
    paises_selecionados = ['United States', 'Brazil', 'India',
'Mexico', 'United Kingdom']
    df = df.loc[df['pais'].isin(paises_selecionados)]
    sns.set_theme(style="darkgrid")
    grap = sns.lineplot(data=df,
                        x='data',
                        y='casos_totais',
                        hue='pais',
                        palette="CMRmap_r")
    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()
```

Código fonte 22 – Código gráfico infectados covid.

Neste código seguimos os passos do código anterior, porém criamos uma lista com os países que queremos observar e em seguida usamos o comando ‘loc’ para selecionar apenas eles, fixamos o tema de fundo, inserimos as opções de dataset, eixo e paleta e chamamos a função ‘FuncFormatter’ da biblioteca ‘matplotlib’ em junto de uma função anônima para editar os valores exibidos no eixo y para a unidade MI (milhões) e por fim plotamos o gráfico. Gráfico este que pode ser observado na figura 11.

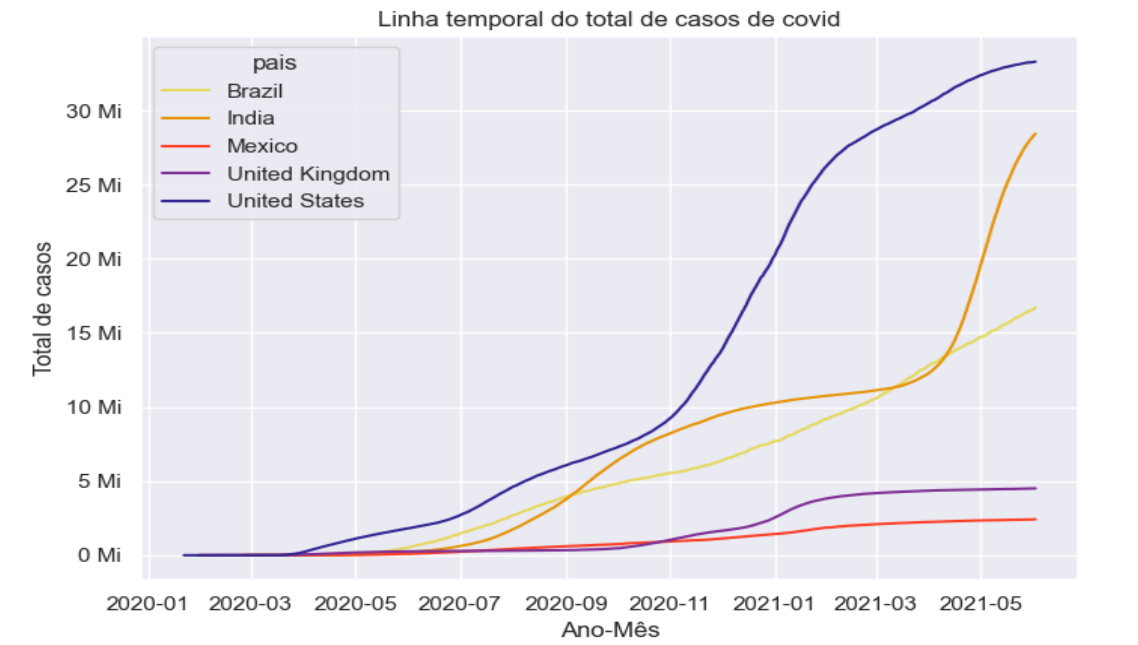

Figura 11 – Gráfico de total de infectados covid.

As primeiras observações que podemos fazer a respeito do gráfico da figura 11 é que a Índia teve um aumento significativo de casos entre março de 2021 e maio de 2021, formando uma linha praticamente vertical, enquanto Reino Unido e México demonstram uma estabilidade a partir de janeiro de 2021, já que a linha de infectados segue um comportamento horizontal. O Brasil, infelizmente, segue com a linha de casos crescendo, sem demonstrar previsão de queda, diferentemente dos EUA que embora sejam o país com maior número de casos demonstram uma melhora significativa tendo em vista que a linha do país indica uma tendência de estabilidade se olharmos o período de maio de 2021 em diante.

Continuando a análise em cima do nosso top 5 temos mais um dado que é muito interessante de se observar de forma gráfica que é a ‘taxa_transmissao’ que representa a taxa de transmissão do vírus. Caso este índice seja igual a 2 por exemplo, significa que uma pessoa infectada tende a transmitir o vírus da covid para outras 2 pessoas, caso o índice seja de 0,5 isso significa que 2 (duas) pessoas infectadas tendem a transmitir o vírus para 1 (uma) pessoa. Para isso iremos usar um gráfico do código fonte 23 a seguir:

```
def indice_reproducao():
    df = pd.read_csv('download/covid19_renamed.csv', sep=',')
    df['data'] = pd.to_datetime(df['data'])
    df = df.dropna(subset=['continente'])
    paises_selecionados = ['United States', 'Brazil', 'India',
'Mexico', 'United Kingdom']
    df = df.loc[df['pais'].isin(paises_selecionados)]
    sns.set_theme(style='darkgrid')
    sns.lineplot(data=df,
                 x='data',
                 y='taxa_transmissao',
                 hue='pais',
                 linewidth=1.1,
                 palette='tab10')
    plt.xlabel("Ano-Mês")
    plt.ylabel("Taxa de reprodução")
    plt.title("Taxa de transmissão ao longo do temppo")
    plt.show()
```

Código fonte 23 – Código gráfico relação mortes x infectados.

No código 23 vamos gerar um novo gráfico de linhas, começando por instanciar nosso arquivo em um dataframe, após isso selecionamos apenas os 5 países que estão na lista ‘paises_selecionados’ que estamos estudando e em seguida começamos a plotar o gráfico, desta vez passamos como um dos parâmetros a espessura da linha (linewidth). O gráfico resultante deste código é o da figura 12.

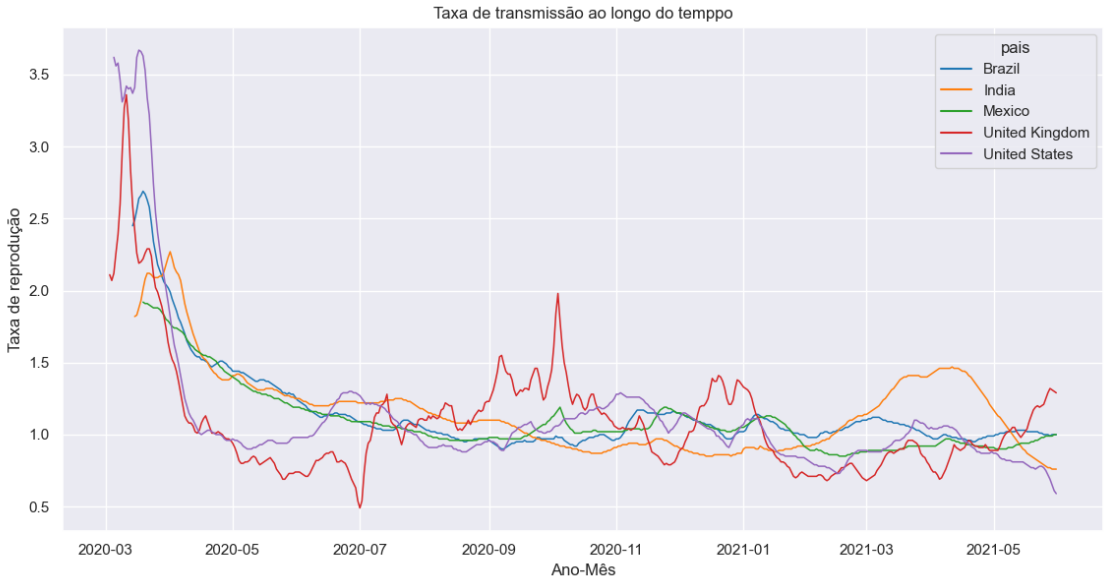

Figura 12 – Gráfico da taxa de transmissão.

O primeiro ponto que podemos observar no gráfico da figura 12 são os momentos em que há um aumento na taxa de transmissão, observando a Índia por exemplo, entre os

meses de janeiro de 2021 e março de 2021 a taxa de transmissão sobe de um patamar menor que 1 para quase 1,5. Esta subida causa um impacto muito grande no número de casos e podemos observar isto retornando a figura 11, nela é possível ver que de março de 2021 em diante houve um aumento significativo no número de casos de covid no país.

Olhando para os EUA é possível ver o inverso acontecer, comparando os gráficos das figuras 11 e 12 é observado que na figura 12 a partir de março de 2021 a taxa de transmissão no país cai de pouco mais de 1 para quase 0,5. No mesmo período na figura 11 é observado que a linha do número de casos passa a se estabilizar, adotando um comportamento mais horizontal.

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* e a partir disso um e-mail será enviado para o usuário que então estará apto a utilizar a ferramenta.

O Databricks é uma aplicação em nuvem que cria para o usuário um cluster de dados (figura 13), dentro deste cluster são criadas bases de dados, tudo controlado e gerenciado dentro do Databricks.

Para iniciarmos nosso trabalho, vamos navegar até ‘clusters’ e em seguida clicar em ‘create cluster’, figura 13, para criar nosso cluster.

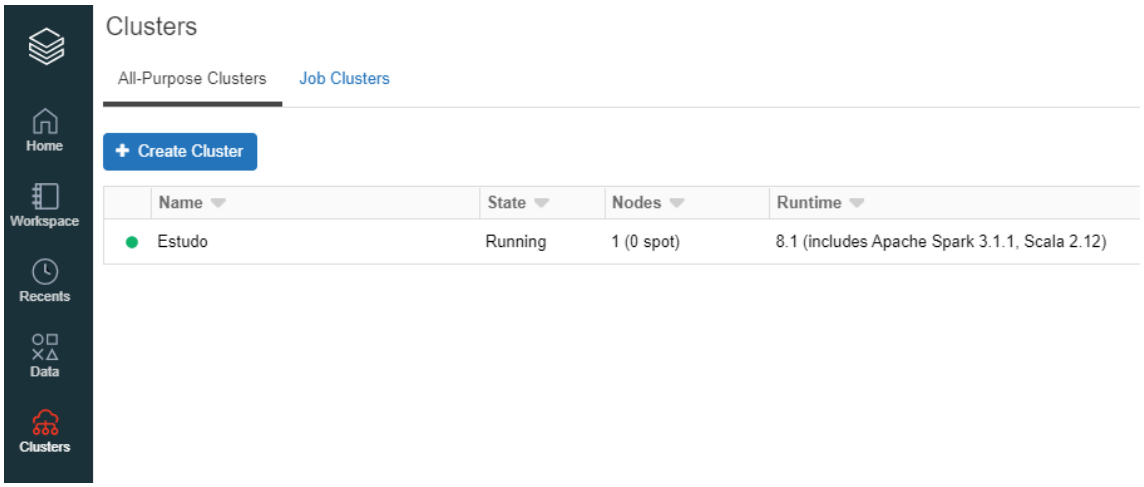

Figura 13 – Cluster em funcionamento no Databricks.

Você será direcionado a uma tela de criação, agora basta inserir um nome para o cluster e escolher a versão do databricks que deseja utilizar, sendo sempre recomendável utilizar a versão mais recente, após isso basta confirmar e o cluster será criado e em

seguida ativado. Vale lembrar que após 2 horas de inatividade o cluster é automaticamente desligado, mas isso não é um problema, pois você pode clonar o cluster inativo e assim ativar um novo cluster com as mesmas bases de dados que você estava anteriormente trabalhando. A reativação de cluster desativado é uma opção exclusiva da versão paga do Databricks.

Com o cluster ativo podemos começar a digitar nossos códigos no notebook e executá-los. Os notebooks no Databricks se assemelham aos notebooks utilizados no Jupyter Notebook, onde cada célula é independente e o desenvolvedor pode escolher rodar o notebook inteiro em sequência ou apenas as células que desejar.

Para criar um notebook é preciso acessar a aba lateral esquerda *Workspace* e clicar em *users*, o e-mail do usuário será exibido e ao clicar na seta na direita aparecerá a opção de criar notebook, com o notebook criado é preciso atrelar ele ao cluster, clicando em *attach* no canto superior esquerdo e selecionando o cluster.

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 outra célula no notebook e digitar o comando mágico % seguido da linguagem (%sql, %python, %py, %md, %shell) como na figura 14 e executar o código através do comando Ctrl+Enter.

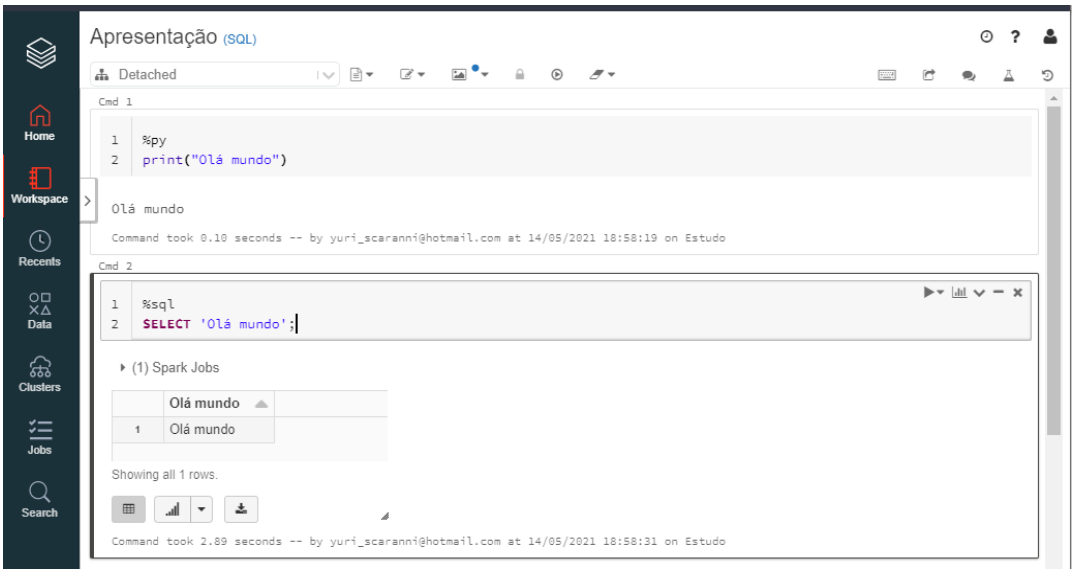

Figura 14 – Notebook simples de apresentação.

O Databricks é uma ferramenta muito boa para lidar com o trabalho de converter arquivos em tabelas, pois, com sua interface de criação simples e prática é possível criar uma tabela em segundos. Para isso vamos fazer o upload do nosso arquivo covid19_renamed.csv para dentro do cluster clicando primeiro em *data* na aba esquerda e em seguida *create table*, desta forma seremos direcionados a tela de upload. Nesta tela você pode escolher o caminho que deseja salvar o arquivo e arrastar o mesmo para dentro do Databricks como na figura 15.

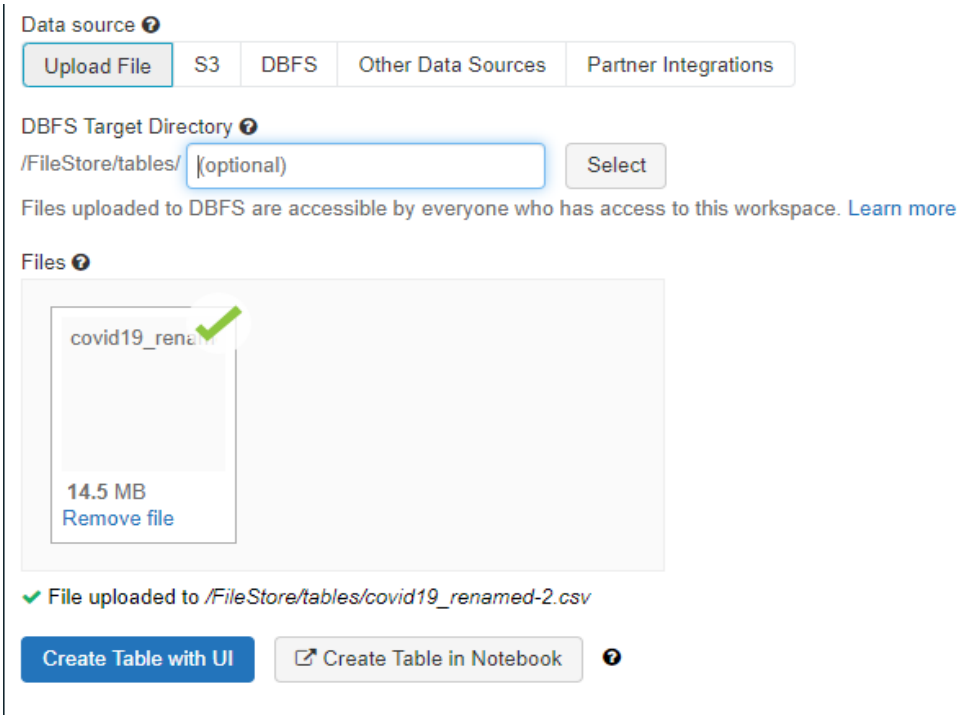

Figura 15 – Tela de upload de arquivos do databricks.

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 16), ao clicar em *Create Table in Notebook* o databricks iniciará um novo notebook contendo a sequência de códigos necessários para gerar a mesma tabela no banco de dados.

Adotaremos aqui o modelo utilizando a UI. Clicando em *Create Table with UI*, em seguida selecionando o cluster e clicando em *preview table*, será aberta a edição dos dados na mesma tela, neste momento é possível definir se a primeira linha será cabeçalho, a base de dados, se o databricks assumirá a tipagem dos dados automaticamente dentre outras opções e em seguida basta clicar em *create table* como mostra a figura 16. Caso opte pela criação com notebook basta clicar no botão *Run all* na parte superior para realizar a criação da tabela.

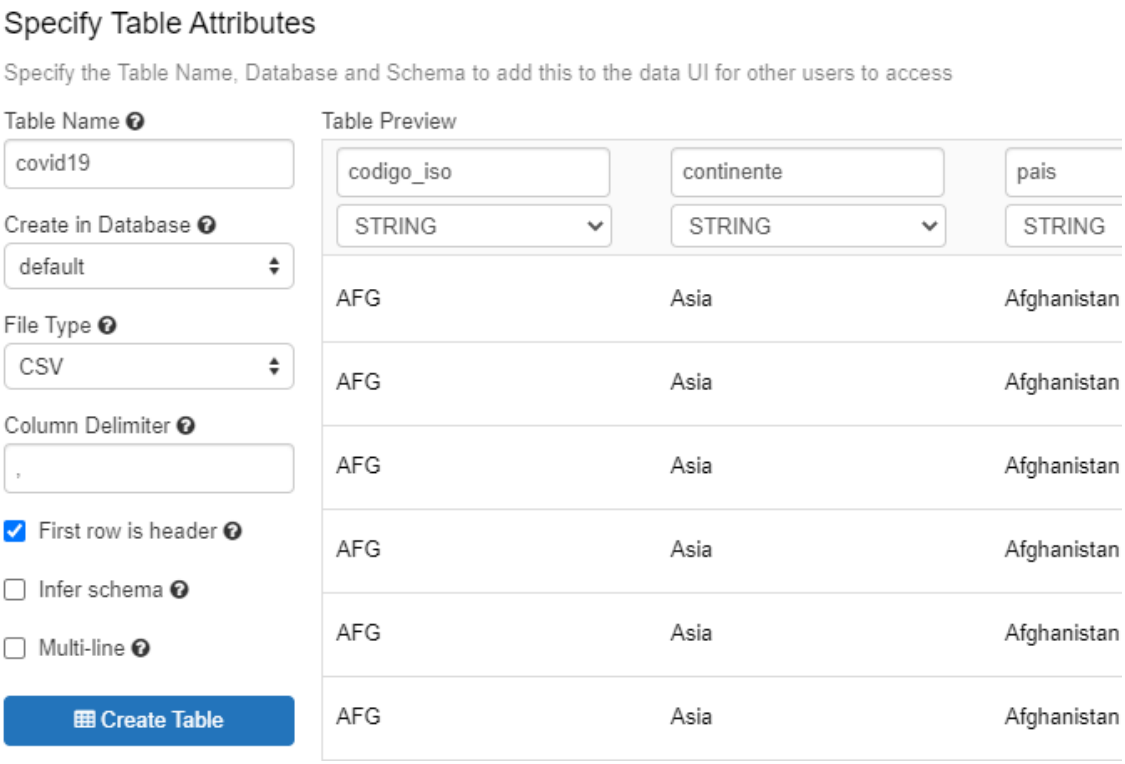

Figura 16– Tela criação de tabela via UI.

Com o arquivo inserido e tabela a gerada, 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 (spark) ou pandas. Então vamos abrir um notebook (clicando em *workspace*) e fazer alguns códigos.

Primeiro abriremos uma célula para inserir nossa marcação de texto (markdown) para facilitar futuras buscas, código fonte 24.

```
%md
# Estudos Covid
```

Código fonte 24 – Código de markdown.

Na célula seguinte vamos continuar nossa análise de dados, porém, utilizaremos a linguagem SQL para construir nossa lógica e então gerar gráficos dentro do próprio Databricks. Para o primeiro gráfico utilizaremos dados de vacinados totais e vacinados dose completa, como mostra o código fonte 25.

```
SELECT
   CAST(populacao AS INT) populacao
  ,CAST(pessoas_vacinadas_dose_completa AS INT) dose_completa
  ,CAST(pessoas_vacinadas AS INT) vacinados
  ,pais
FROM default.covid19
WHERE data = '2021-05-30'
  AND pessoas_vacinadas_dose_completa IS NOT NULL
  AND pessoas_vacinadas IS NOT NULL
  AND populacao IS NOT NULL
ORDER BY populacao DESC
LIMIT 15;
```

Código fonte 25 – Código gráfico databricks barras

A intenção deste código é mostrar através de 3 colunas os números absolutos de vacinados em relação a população total, garantindo que os dados que queremos estão preenchidos (infelizmente há casos em que os dados do país não estão disponíveis como a China por exemplo). Limitando a uma data recente, dia 30 de maio, e extraindo somente 15 países e ordenando pelo tamanho da população. O resultado podemos acompanhar na figura 17.

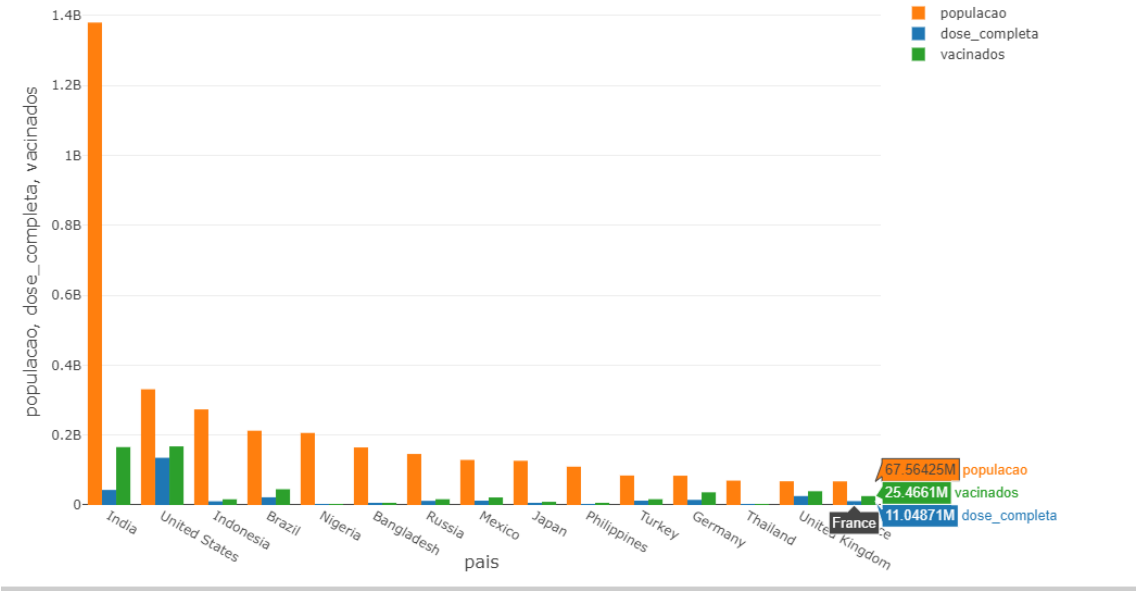

Figura 17 – Gráfico de vacinados x população Databricks.

Para gerar e manipular este gráfico basta clicar na parte abaixo dos resultados da query e depois selecionar os campos que deseja que estejam no gráfico como mostra a figura 18.

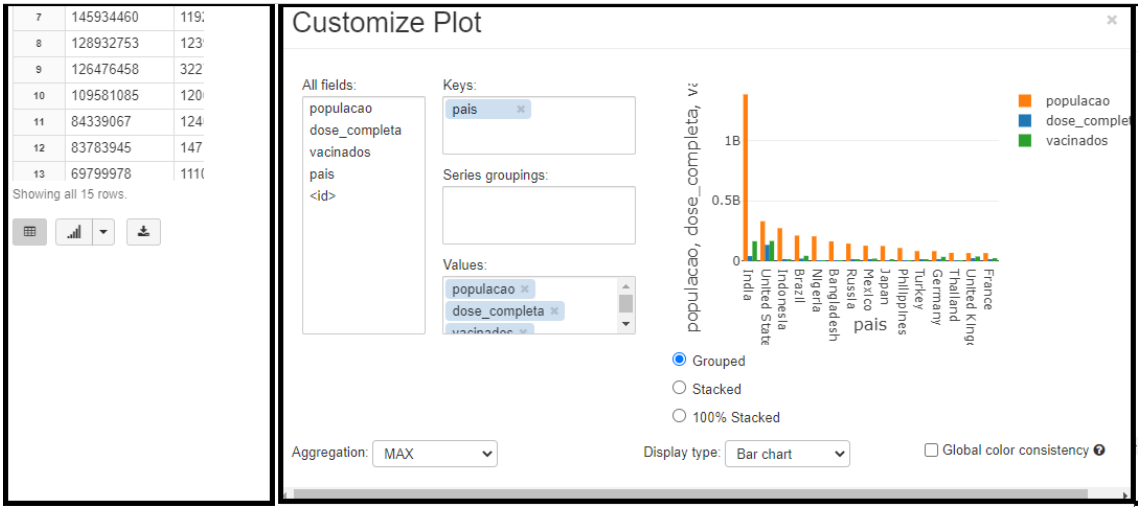

Figura 18 – Manipulando gráficos Databricks.

O primeiro insight que podemos ter com este gráfico da figura 17 é a relação entre números absolutos de população e vacinados. A Índia por exemplo está com um número baixo de vacinados em relação a população, enquanto os EUA possuem quase metade dos cidadãos na condição de vacinados com doses totais. Isso evidencia o comportamento dos países durante a pandemia, já que os EUA realizaram a compra e distribuição de vacinas em volume maior que os demais países.

Outro gráfico interessante que podemos fazer com extrema facilidade no Databricks diz respeito ao total de óbitos por covid e casos de covid agrupados por continente, semelhante ao que fizemos no tópico 4.2, dividindo através de continentes como mostra o código fonte 26. Somando o total de casos e óbitos por continente.

```
SELECT
  continente, SUM(mortes_totais) AS obitos, SUM(casos_totais) AS
casos
FROM default.covid19
WHERE data = '2021-05-30'
  AND continente IS NOT NULL
GROUP BY 1;
```

Código fonte 26 – Código gráfico databricks pizza

Embora os gráficos gerados no seaborn ofereçam muitas opções de personalização e edição, os gráficos feitos pelo Databricks são feitos de forma mais simples, com bem menos linhas de código, já que toda a parte gráfica que o cientista precisaria programar já está pronta, entregando muita agilidade no processo. O resultado podemos acompanhar na figura 19.

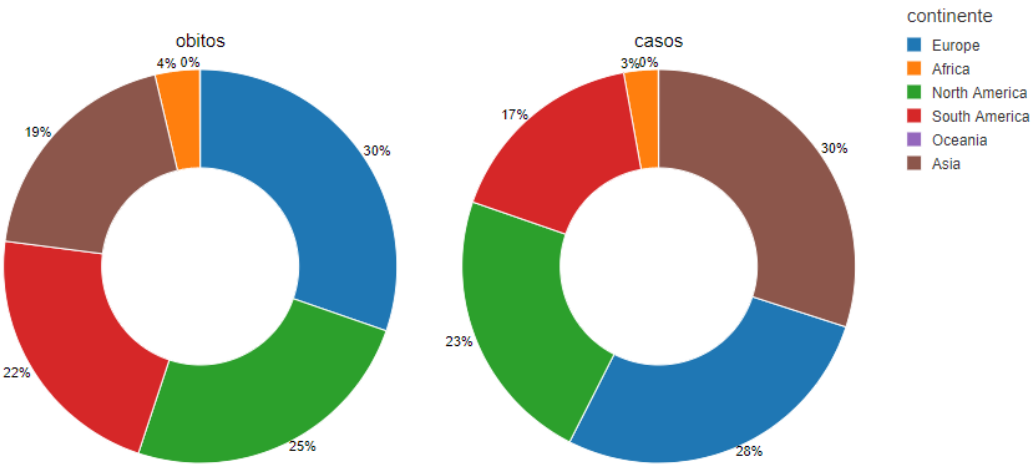

Figura 19 – Plotagem de gráfico de pizza no databricks.

Neste gráfico mostramos a divisão de óbitos e casos, e é notório certo equilíbrio no número total, isto pode ser causado devido ao fato de que em cada continente pelo menos um país despontou como epicentro, e para evidenciar isso podemos exemplificar com Índia na Asia, EUA na América do Norte e Brasil na América do Sul.

Por fim, faremos uma nova análise, mas dessa vez ela se baseará nos dados da query do Databricks, vamos vincular o 'pib_per_capita' com o número de óbitos por covid19 do país. Para isso vamos primeiro selecionar os países, ordenar por valor de 'pib_per_capita' e criar um gerador de ordenação como exemplifica o código fonte 27 a seguir.

```
SELECT
      ROW_NUMBER() OVER (
        ORDER BY
          PIB DESC
      ) AS posicao,
      *
    FROM(
        SELECT
          DISTINCT pais,
          CAST(pib_per_capita AS INT) AS PIB
        FROM
          default.covid19
        WHERE
          pib_per_capita IS NOT NULL
        ORDER BY
          PIB DESC
      )
```

Código fonte 27 – Ranking de pib_per_capita

Em seguida faremos uma seleção que já utilizamos antes no Python, com os países e o número de óbitos por covid19 até a data 30 de maio de 2021. O exemplo está abaixo no código fonte 28.

```
SELECT
  DISTINCT pais,
  CAST(mortes_totais AS INT) AS obitos
FROM
  default.covid19
WHERE
   mortes_totais IS NOT NULL
  AND data = '2021-05-30'
ORDER BY obitos DESC
```

Código fonte 28 – Óbitos por covid19 até fim de maio.

Agora vamos unir estas duas análises através do campo ‘pais’ que existe nos dois através de um INNER JOIN pois neste caso é interessante que ambas as tabelas tenham os valores que buscamos. O código fonte 29 mostra o código completo. Nele fazemos ainda um pequeno ajuste, concatenando o dado de posição com o caractere ‘°’ para indicar posição no ranking de pib per capita. Uma observação que podemos fazer é que o ranking é bem misto, contando com países com um pib per capita alto como os EUA e Itália, mas também com países que ocupam o fim do ranking como a Índia, o que nos dá uma dimensão de que o número de mortos pode não estar diretamente ligado ao pib_per_capita do país. O resultado pode ser visto na figura

	pais	obitos	ranking_pib_per_capita
1	United States	594443	13°
2	Brazil	461931	86°
3	India	329100	127°
4	Mexico	223507	71°
5	United Kingdom	128043	28°
6	Italy	126046	37°
7	Russia	119130	55°
8	France	109562	30°
9	Germany	88431	22°
10	Colombia	88282	92°

Figura 20 – Relação pib per capita x total de óbitos.

```
SELECT
  paises_obitos.*,
  paises_pib.posicao || '°' AS ranking_pib_per_capita
FROM
  (
    SELECT
      DISTINCT pais,
      CAST(mortes_totais AS INT) AS obitos
    FROM
      default.covid19
    WHERE
      mortes_totais IS NOT NULL
      AND data = '2021-05-30'
    ORDER BY
      obitos DESC
  ) paises_obitos
  INNER JOIN (
    SELECT
      ROW_NUMBER() OVER (
        ORDER BY
          PIB DESC
      ) AS posicao,
      *
    FROM(
        SELECT
          DISTINCT pais,
          CAST(pib_per_capita AS INT) AS PIB
        FROM
          default.covid19
        WHERE
          pib_per_capita IS NOT NULL
        ORDER BY
          PIB DESC
      )
  ) paises_pib ON paises_obitos.pais = paises_pib.pais
ORDER BY
  paises_obitos.obitos DESC
```

Código fonte 29 – Relação pib per capita x total de óbitos.

5 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.

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

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.

SIQUEIRA, Fernando De. DB2 - Banco de dados - Modelo Relacional. Disponível em: <http://www.cadcobol.com.br/db2_novo_modelo_relacional.htm>. Acesso em: 28 maio. 2021.