



Criptomoeda

Projeto Final BC17

Grupo 2

Equipe de Engenheiros de Dados



**Aldreks
Albuquerque**



Carlos Bahia



Jalvo Alef



**Marco Aurélio
Menezes**

Índice

- ➔ **Escopo do Projeto**
- ➔ **Dados Brutos**
- ➔ **Introdução ao Tema**
- ➔ **Tecnologias**
- ➔ **WorkFlow**
- ➔ **Ambiente Cloud**
- ➔ **ETL**
- ➔ **Insights / Data Studio**
- ➔ **Análise de Custo**
- ➔ **Requisitos Desejáveis**

01

Escopo do Projeto

Tecnologias

Google Cloud Platform

Python

Pandas

PySpark

SparkSQL

Data Studio

Big Query

SQL e NoSQL



**16 Requisitos
técnicos obrigatórios**



**+ 6 Requisitos
técnicos desejáveis**

02

Dados brutos

Dataset original

10 colunas x 2,5 milhões de linhas x 200 MB

	ticker	TokenName	Date	Open	High	Low	Close	Volume	Market Cap
1	AVA	Travala.com	Apr 30, 2022	1.28	1.28	1.14		8157497.0	58101831.0
2	AVA	AVA	Apr 29, 2022	1.2	1.43	1.2	1.28	26864987.0	65578069.0
3	AVA	Travala.com	Apr 28, 2022	1.2	1.21	1.17	1.2	4166128.0	61461049.0
4		Travala.com	Apr 27, 2022	1.17	1.21		1.2	3792108.0	61046830.0
5	AVA	Travala.com	Apr 26, 2022	1.26	1.27	1.15	1.17	5080377.0	59743409.0

Dataset Complementar

.JSON



```
g_dsl - Bloco de Notas
Arquivo Editar Formatar Colar Ajuda
[{"TRONPAD":"TRONPAD"}, {"VITEVITE":"VITE"}, {"HYCONHYC":"HYCON"}, {"USDKUSD"}, {"PAXEXPAXEX":"PAXEX"}, {"STRAKSSTAK":"STRAKS"}, {"XDNAXDNA":"XDNA"}, {"H"}, {"RKRMRK":"RMRK"}, {"XMONXMON":"XMON"}, {"NFTXNFTX":"NFTX"}, {"WHALEWHALE":"WH"}, {"EDOSE":"DOSE"}, {"DOGAM\00cdDOGA":"DOGAMI"}, {"UMI":"UMI"}, {"BTC":"Bitcoin"}, {"HT":"Huobi Token"}, {"BSV":"Bitcoin SV"}, {"GRT":"The Graph"}, {"XEC"}, {"2"}, {"ROSE":"Oasis Network"}, {"TFUEL":"Theta Fuel"}, {"IOTX":"IoTeX"}, {"LD"}, {"a Token"}, {"NU":"NuCypher"}, {"REN":"Ren"}, {"MX":"MX TOKEN"}, {"XNO":"Nano"}, {"BNB"}, {"FRAX":"Frax"}, {"HBTC":"Huobi BTC"}, {"BTTCOLD":"BitTorrent"}, {"DFI"}, {"nic"}, {"FX":"Function X"}, {"STEEM":"Steem"}, {"REP":"Augur"}, {"METIS":"Met"}]
```

.CSV



Date	Open	High	Low	Close	Adj Close	Volume
2013-04-29	54254.000000	55336.000000	54254.000000	54887.000000	54887.0000	
2013-04-30	54888.000000	55910.000000	54585.000000	55910.000000	55910.0000	
2013-05-02	55919.000000	55919.000000	55104.000000	55322.000000	55322.0000	
2013-05-03	55330.000000	56366.000000	55330.000000	55488.000000	55488.0000	
2013-05-06	55488.000000	55513.000000	55488.000000	55488.000000	55488.0000	

Date	Open	High	Low	Close	Adj Close	Volume
2013-04-29	9200.559570	9258.889648	9195.589844	9245.219727	9245.219727	28
2013-04-30	9240.259766	9276.879883	9205.620117	9276.879883	9276.879883	37
2013-05-01	9248.450195	9248.450195	9169.780273	9175.780273	9175.780273	35

BOVESPA
Bolsa de Valores do Brasil

03

Introdução ao Tema

O que é Criptomoeda?

Como funciona a Criptomoeda?

O que é "blockchain"?

04

Tecnologias

Tecnologías utilizadas



Trello Áreas de trabalho Recente Marcado com estrela Modelos Criar


Pesquisar

Quadro **Gestão de Projetos** Apresentação Grupo 2 Visível à Área de trabalho

Placker - Sign-up Power-up do Calendário Power-Ups Automação

Recursos para Projetos

Aqui, colocaremos as ferramentas adicionais e seus links que usaremos para o projeto, como o GCP, Mongo, Gannt, Colab, ou qualquer outro link ou arquivo externo



Senha GCP SQL Jalvo Alef

projeto_final.ipynb

Esquema do Projeto

MySQL/Postgres

- 6 - colunas exibidos na lingua PT-BR
- 7 - Converter e normalizar
- 10 - análises dentro do Big Query
- 11 - Utilizar o banco NoSQL
- 12 - dados tratados armazenados datalake(Gstorage) , DW(BigQuery)
- 13 - Dataframe(s) resultante(s)

A fazer

- Buscar e escolher Dataset Adicional
- Acessos ao GCP - MySQL
- 01 - triggers e procedures para o banco SQL
- 2 - salvos e operados obrigatoriamente dentro da plataforma GCP
- 9 - Operações utilizando o SparkSQL

Em Produção

- 4 - datasets devem ter formatos diferentes
- 8 - Operações usando PySpark
- Tratamento da base de dados
- Traduzir colunas do DataSet Original
- 5 - Operações com Pandas

Concluido

- 16 - Entregar todos os scripts (DDL/DML)
- relatório completo com os insights de ETL
- Capacidade de argumentação
- Habilidade de interpretação e análise de dados.
- Capacidade de implementação linguagem SQL.
- Capacidade de organização e grupo

Trello

Ferramenta de gestão demandas



Soulcode - Projeto Final

Adicionar descrição do quadro

Padrão

Gantt

+ Adicionar Visualização

Última visualização Convidar / 4 + Adicionar ao quadro ...

Integrar

Automatizar



Novo elemento

Busca... Pessoas Filtro Ordenar

Requisitos Obrigatórios

Subelementos

Pessoa

Status

Data

Trello Card



1. Haver utilização de triggers e procedures para o banco SQL



↳ 1

JS

Feito

jun 8

1.1 trigger



1

JS

Feito

2. Datasets (salvos e tratados) em GCP, não pode usar Google drive



↳ 1

JS

Feito

jun 9

3. Os datasets salvos e MySQL/Postgres



↳ 1

JS

Feito

jun 10

4. datasets devem ter formatos diferentes (CSV / Json / Parquet / Sql / NoSql)



↳ 1

JS

Em progresso

jun 3

5. Utilizar o Pandas (limpezas, transformações e normalizações)



↳ 2

JS

Feito

6. Traduzir os datasets (dados/colunas) para a língua PT-BR



1

JS

Feito

7. Converter e normalizar os dados via SPARK (csv/parquet)



1

JS

Feito

jun 8

8. Utilizar o PySpark com a descrição de cada uma das operações



1

JS

Feito

9. Utilizar o SparkSQL com a descrição de cada uma das operações



1

JS

Feito

10. Análises em Big Query SQL com a descrição das consultas feitas



1

JS

Iniciar

11. Utilizar o banco NoSQL (MongoDB ou Cassandra) como um data lake



1

JS

Feito

12. Datasets (tratados) em GCP, data lake Google Storage e/ou BigQuery



1

JS

Feito

13. Criar o usuário (soulcode) e senha (a1b2c3) no MongoDB Atlas e informar



1

JS

Feito

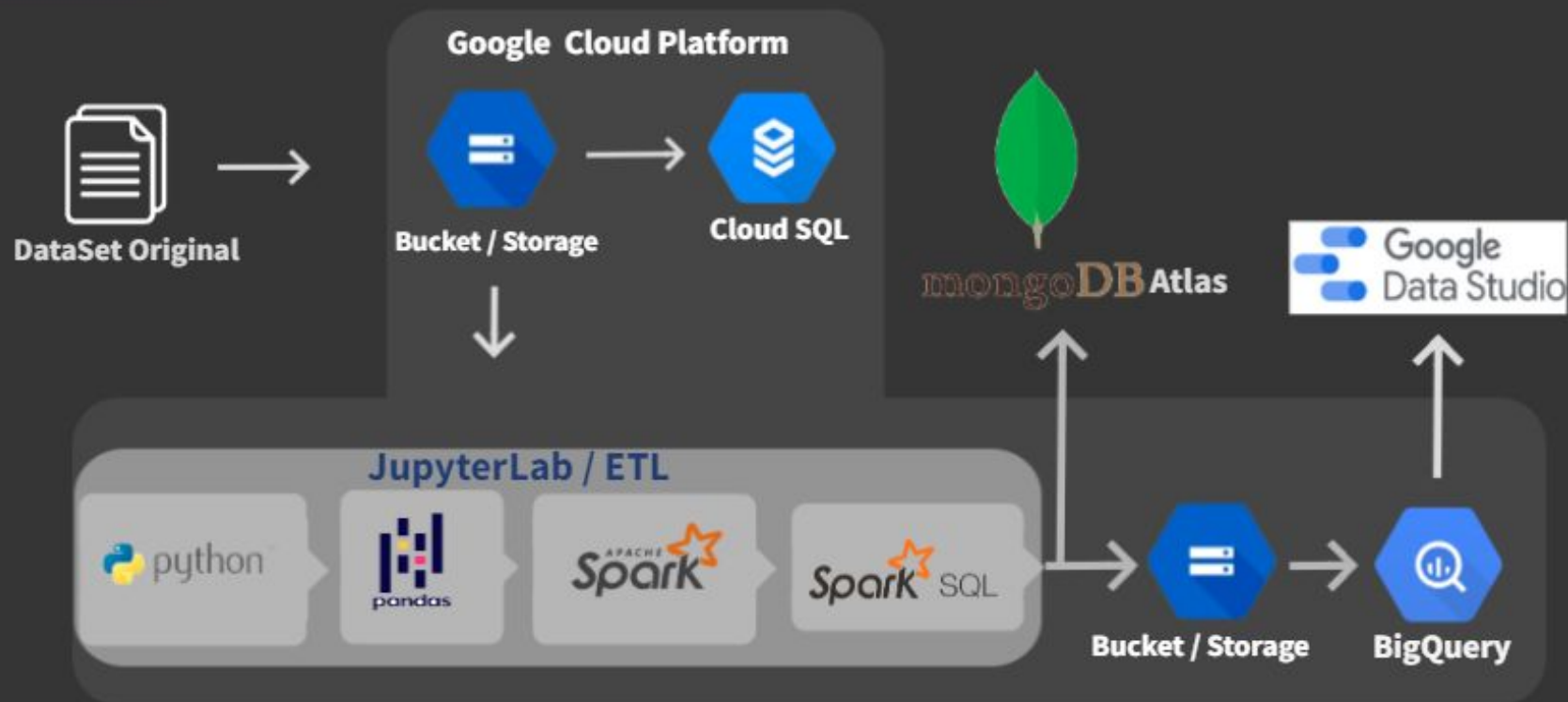
Monday

Ferramenta de gestão Processos Diagrama de Gantt

05

WorkFlow

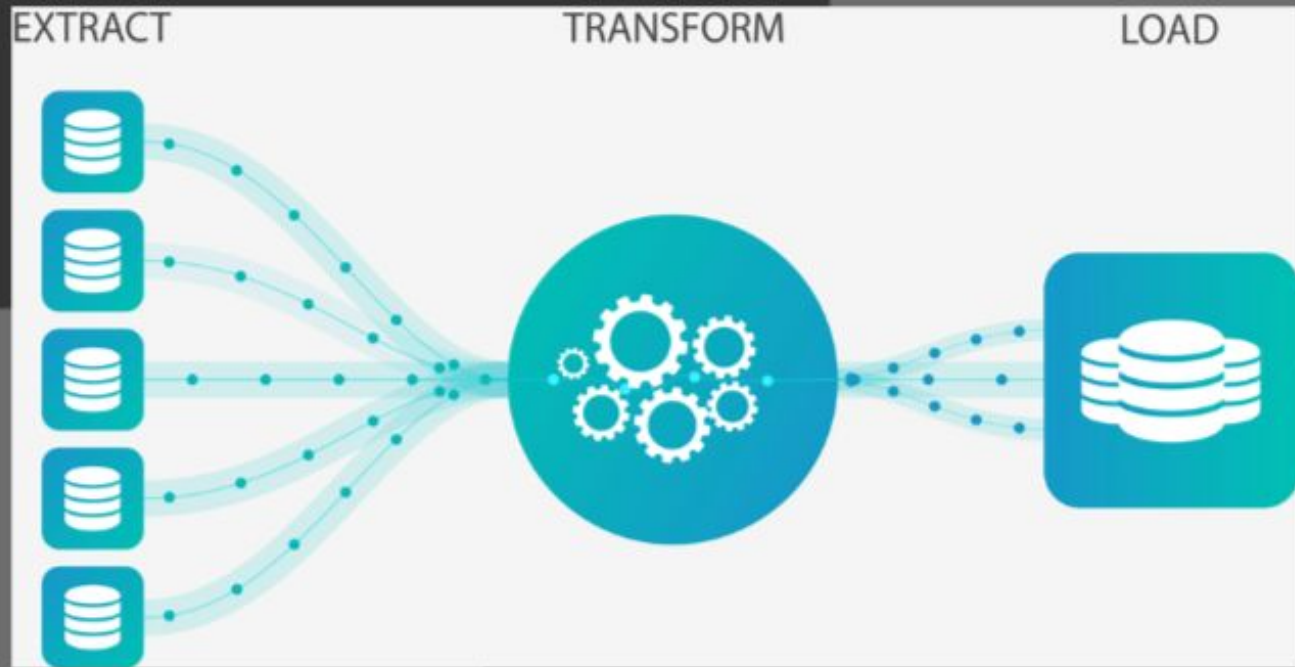
Fluxograma do Processo de ETL



06

Ambiente Cloud

ETL



Cloud Storage

Google Cloud Platform BC17 Search Products, resources, docs (/)

Cloud Storage

← Bucket details

criptomoeda

Location

us-east1 (South Carolina)

Storage class

Standard

Public access

Not public

Protection

None

OBJECTS CONFIGURATION PERMISSIONS PROTECTION LIFECYCLE

Buckets > criptomoeda

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DOWNLOAD DELETE

Filter by name prefix only Filter Filter objects and folders

GCP

Banco Relacional - SQL

Google Cloud Platform BC17 Search Products, resources, docs (/)

SQL

PRIMARY INSTANCE

Overview

Connections

Users

Databases

Backups

Replicas

Operations

Release Notes

Overview

EDIT

IMPORT

EXPORT

RESTART

STOP

DELETE

CLONE

All instances > mycriptomoeda

mycriptomoeda

MySQL 8.0

1 hour 6 hours 1 day 7 days 30 days Custom

Chart

CPU utilization

4%

2%

0

UTC-3 8:00 AM 10:00 AM 12:00 PM 2:00 PM 4:00 PM 6:00 PM 8:00 PM 10:00 PM Jun 16 2:00 AM 4:00 AM 6:00 AM

Connect to this instance

Public IP address

104.155.120.253

Connection name

central-point-349020:us-central1:mycriptomoeda

Configuration

vCPUs

4

Memory

26 GB

SSD storage

100 GB

Database version is MySQL 8.0.26

Banco Relacional - SQL

The screenshot displays the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar shows the 'SCHEMAS' panel with a tree view containing 'criptomeda' (Tables, Views, Stored Procedures, Functions) and 'sys'. The main query editor shows a SQL query: `SELECT * FROM criptomeda_origen LIMIT 1000;`. The right sidebar shows the 'SQL Assistant' panel with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.' The bottom panel shows the 'Result Grid' with columns: id, ticker, TokenName, Date, Open, High, Low, Close, Volume, and Market_Cap. The data is filtered to show 1000 rows. The bottom status bar shows 'Object Info' and 'Session'.

MySQL Workbench

central-point-349022-us-central-1

File Edit View Query Database Server Tools Scripting Help

Navigator

Query 1

Limit to 1000 rows

SQL Assistant

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid

	id	ticker	TokenName	Date	Open	High	Low	Close	Volume	Market_Cap
0					0.00	0.00	0.00	0.00	0.00	0.00
1	AVK	Travelo.com	Apr 20, 2022	1.38	1.38	1.14	1.14	8157497.00	58121831.80	
2	AVK	Travelo.com	Apr 20, 2022	1.20	1.40	1.20	1.20	20864987.00	65578069.00	
3	AVK	Travelo.com	Apr 26, 2022	1.26	1.21	1.17	1.20	4184128.00	61461049.00	
4	AVK	Travelo.com	Apr 27, 2022	1.17	1.21	1.16	1.20	3792108.00	61046830.00	
5	AVK	Travelo.com	Apr 26, 2022	1.26	1.27	1.15	1.17	5086377.00	56743409.00	
6	AVK	Travelo.com	Apr 25, 2022	1.26	1.26	1.20	1.26	5614731.00	64200715.00	
7	AVK	Travelo.com	Apr 24, 2022	1.29	1.29	1.25	1.26	3962797.00	64026141.80	
8	AVK	Travelo.com	Apr 23, 2022	1.33	1.33	1.28	1.29	4540066.00	65828257.00	
9	AVK	Travelo.com	Apr 22, 2022	1.37	1.37	1.33	1.33	4147223.00	67689237.80	
10	AVK	Travelo.com	Apr 21, 2022	1.41	1.44	1.35	1.37	3838261.00	66519165.00	
11	AVK	Travelo.com	Apr 20, 2022	1.45	1.46	1.40	1.41	3766846.00	71827230.00	
12	AVK	Travelo.com	Apr 19, 2022	1.48	1.48	1.36	1.45	4181051.00	74617948.00	

criptomeda_origen

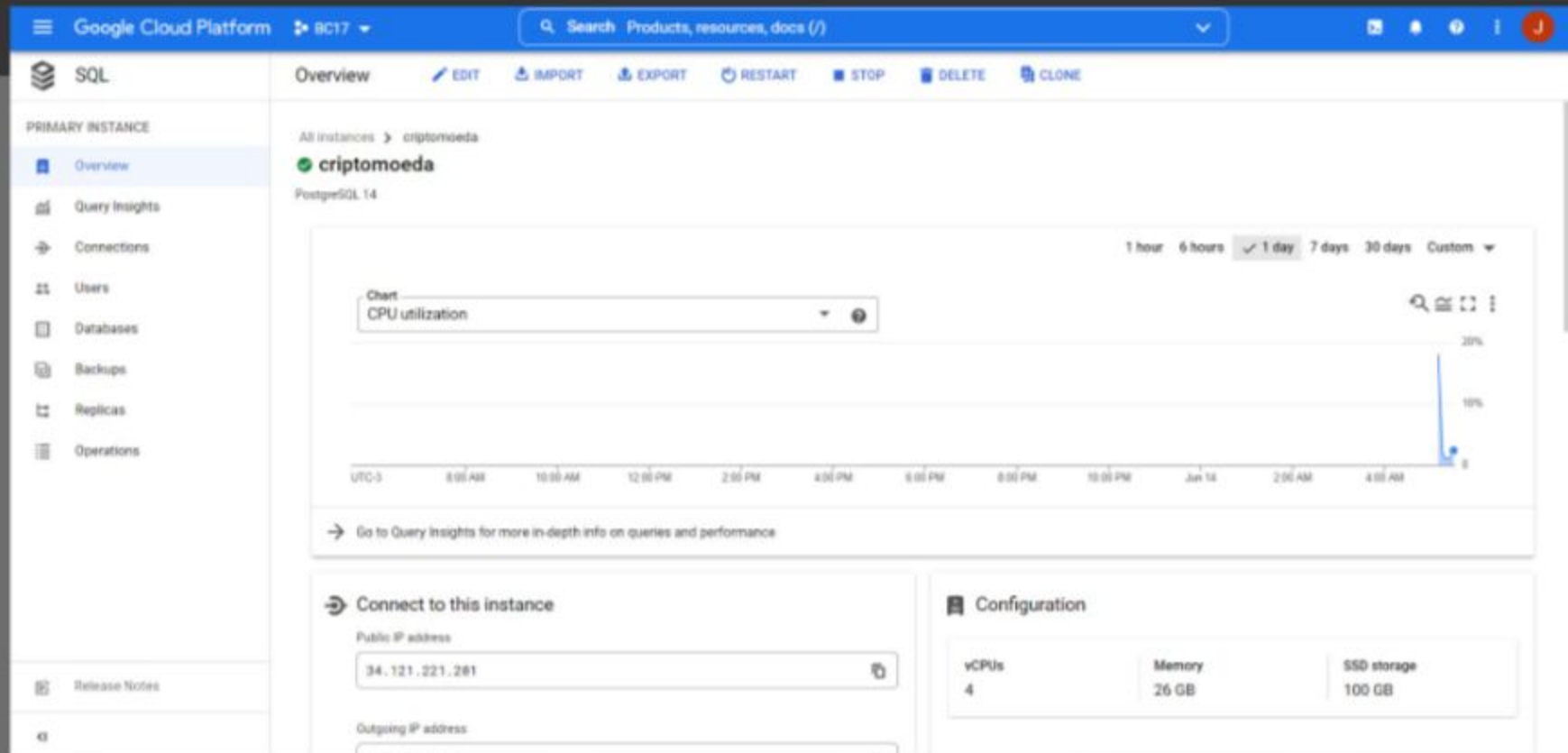
Output

Action Output

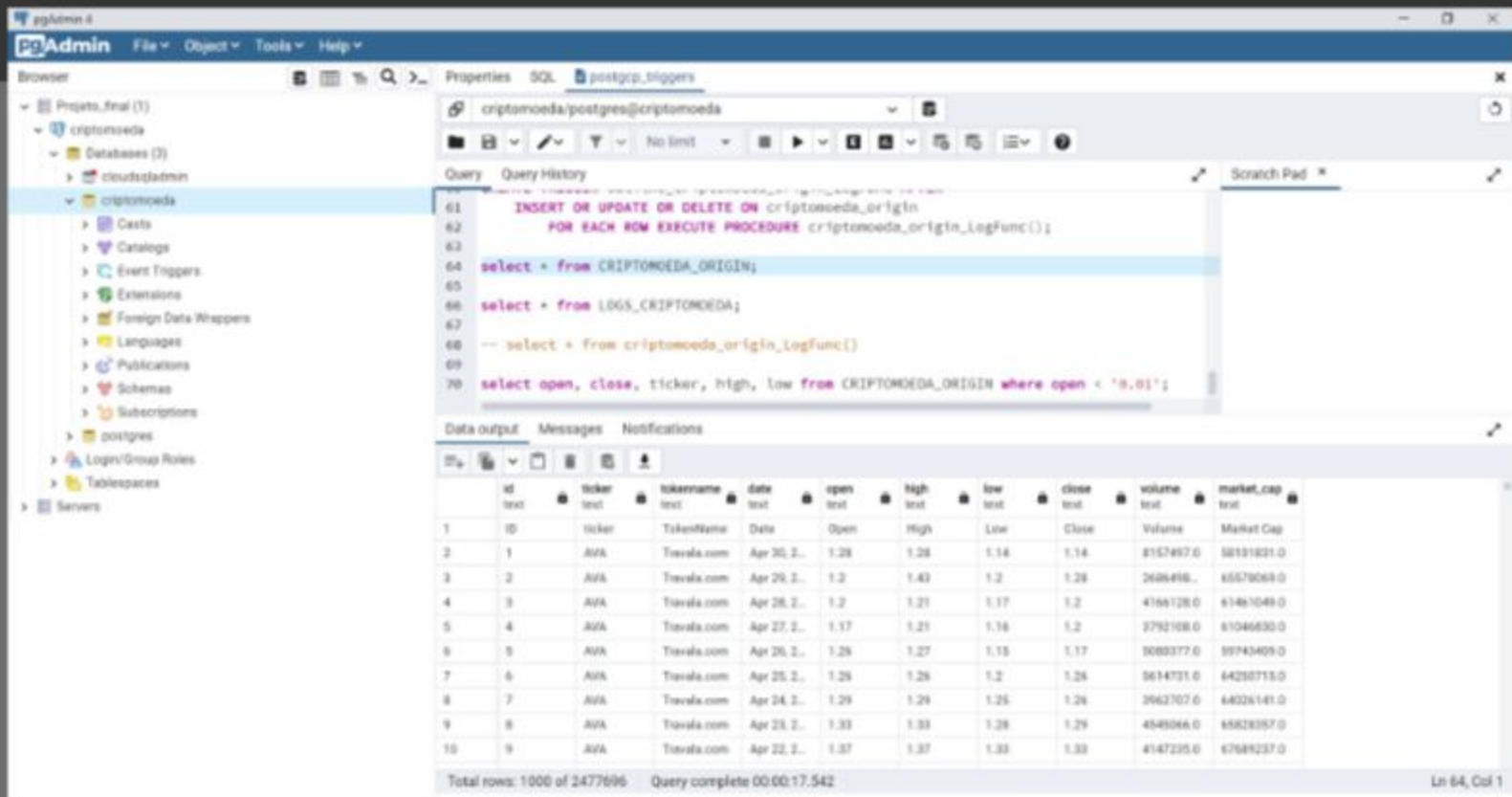
#	Time	Action	Message	Duration / Fetch
2	06:16:30	use criptomeda	0 row(s) affected	0.141 sec
3	06:16:33	show tables	0 row(s) returned	0.156 sec / 0.000 sec
4	06:16:45	CREATE TABLE IF NOT EXISTS criptomeda_origen (id integer, ticker varchar(30), TokenName varchar(100))	0 row(s) affected	0.172 sec
5	06:16:52	show tables	1 row(s) returned	0.156 sec / 0.000 sec
6	06:20:32	SELECT * FROM criptomeda_origen	Error Code: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1	0.141 sec
7	06:20:38	SELECT * FROM criptomeda_origen LIMIT 0, 1000	1000 row(s) returned	0.156 sec / 0.156 sec

Object Info Session

Banco Relacional - SQL



Banco Relacional - SQL



The screenshot shows the pgAdmin 4 web interface. On the left, the 'Browser' pane displays a tree structure of the database 'criptomoeda', with 'CriptoMoeda' selected. The main pane shows a SQL query in the 'Query' tab, which is a trigger function for 'criptomoeda_origin'. The query includes an 'INSERT OR UPDATE OR DELETE' trigger definition and a 'select' statement that filters for 'open < '0.01''. Below the query, the 'Data output' tab displays a table with 11 columns: id, ticker, tokenname, date, open, high, low, close, volume, and market_cap. The table contains 10 rows of data. The status bar at the bottom indicates 'Total rows: 1000 of 2477696' and 'Query complete 00:00:17.542'.

```
61 INSERT OR UPDATE OR DELETE ON criptoMoeda_origin
62 FOR EACH ROW EXECUTE PROCEDURE criptoMoeda_origin_LogFunc();
63
64 select * from CRIPTOMOEDA_ORIGIN;
65
66 select * from LOGS_CRIPTOMOEDA;
67
68 -- select * from criptoMoeda_origin_LogFunc();
69
70 select open, close, ticker, high, low from CRIPTOMOEDA_ORIGIN where open < '0.01';
```

	id	ticker	tokenname	date	open	high	low	close	volume	market_cap
1	ID	ticker	TokenName	Date	Open	High	Low	Close	Volume	Market Cap
2	1	AVA	Toraxa.com	Apr 20, 2...	1.28	1.28	1.14	1.14	8157497.0	58191821.0
3	2	AVA	Toraxa.com	Apr 20, 2...	1.2	1.43	1.2	1.28	2608416...	65579069.0
4	3	AVA	Toraxa.com	Apr 26, 2...	1.2	1.21	1.17	1.2	4166128.0	61481049.0
5	4	AVA	Toraxa.com	Apr 27, 2...	1.17	1.21	1.16	1.2	2792108.0	61046630.0
6	5	AVA	Toraxa.com	Apr 26, 2...	1.28	1.27	1.15	1.17	9083377.0	59743409.0
7	6	AVA	Toraxa.com	Apr 25, 2...	1.28	1.28	1.2	1.26	5614731.0	64258715.0
8	7	AVA	Toraxa.com	Apr 24, 2...	1.29	1.29	1.25	1.26	2962707.0	64028141.0
9	8	AVA	Toraxa.com	Apr 23, 2...	1.33	1.33	1.28	1.29	4545046.0	65828357.0
10	9	AVA	Toraxa.com	Apr 22, 2...	1.37	1.37	1.33	1.33	4147235.0	67689237.0

Total rows: 1000 of 2477696 Query complete 00:00:17.542

Ln 64, Col 1

Comparativo

Tecnologias	PostgreSQL	MySQL
GCP	≈ 30 s 810 MB	≈ 30 s 2 GB
Tipagem	Otimizada	Precisa de ajustes

200 MB e 2,5 Mi de linhas

07



ETL

Instalação de bibliotecas no JupyterLab GCP

```
1 ## Instalação da Biblioteca Gerenciador de Arquivos do GCP
2 !pip install gcsfs
```

```
1 ## Instalação da Biblioteca para uso do MongoDB Atlas
2 !pip install pymongo[srv]
```

```
1 ## Instalação da Biblioteca pySpark
2 !pip install pyspark
```

```
1 ## Instalação da Biblioteca parquet
2 !pip install pyarrow
```

Carregamento das bibliotecas no JupyterLab GCP

```
1 # Carregamento biblioteca Pandas
2 import pandas as pd
3
4 # Carregamento biblioteca Mongo
5 import pymongo
6 from pymongo import MongoClient
7
8 #Uso em Parquet
9 import pyarrow
10
11 # Carregamento biblioteca GCP
12 import gcsfs #acessar GCP
13 from google.cloud import storage
14 import os
15
16 # Carregamento biblioteca (tratamento de arquivos)
17 from bson.json_util import dumps, loads
18 import csv
19
```

Conexão com GCP e preparação para importação do DataSet

```
1 #CONFIGURAÇÃO DA CHAVE DE SEGURANÇA DO GCP (ACESSO)
2 serviceAccount = 'central-point-349020-90861e3e3455.json'
3 os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = serviceAccount
4
5 #Cria conexão com bucket GCP
6 client = storage.Client()
7
8 #Define pasta bucket para normalização do Dataset (Cloud Storage/GCP)
9 folder_bucket = client.get_bucket('criptomoeda')
10
11 #Define arquivo a ser extraído do bucket/GCP
12 folder_bucket.blob('Tratada_SparkValorMercadoMaiorZero.csv')
13
14 #Cria Path do local de origem do arquivo a ser extraído do Bucket/GCP (gsutil URI)
15 path_tratados = 'gs://criptomoeda/Tratados/Tratada_SparkValorMercadoMaiorZero.csv'
16
```

Importação base de dados, traduz os rótulos das colunas de inglês para PT-BR e cria Dataframe em arquivo PARQUET

```
1
2 # Preparando tradução dos rótulos das colunas do dataset de inglês para PT-BR via Pandas
3 Rotulo_Ingles = ([ 'ticker', 'TokenName', 'Date', 'Open', 'High', 'Low', 'Close', 'Volume',
4 Rotulo_PortBR = ([ 'Cod_Empresa', 'Empresa', 'Dt_Negociacao', 'Abertura', 'Max', 'Min', 'Fe
5                     'Volume_Negociado', 'Valor_Mercado_Empresa'])
6
7 # Importa base CSV do Bucket/GCP e cria Dataframe em Pandas
8 print(">>> Carregar Base CSV.")
9 df_pandas = pd.read_csv(path_CSV_bucket, sep=',', usecols = Rotulo_Ingles)
10
11 # Aplica a atualização da tradução dos rótulos das colunas
12 df_pandas.columns = Rotulo_PortBR
13
14 # Cria cópia do DF para normalização futura
15 df_pd = df_pandas.copy()
16
17 #Converte DF Pandas em arquivo Parquet
18 print(">>> Salva DF_Pandas em Parquet.")
19 df_pd.to_parquet('df.parquet')
20
21 print(">>> Carrega Base Parquet para DF.")
22 df_pd = pd.read_parquet('df.parquet')
23
24 print(">>> Base Parquet carregada com sucesso.")
```

Leitura e análise dos dados (início e fim)

```
1 ## Verifica a qtd de Colunas e linhas (tamanho da base de dados)
2 df_pd.shape
```

```
(1249976, 10)
```

```
1 # Ordena o DataFrame por Data e demais colunas abaixo
2 df_pd.sort_values(by=['Dt_Negociacao', 'Cod_Empresa', 'Empresa', 'Volume_Negociado'], inplace=True)
```

```
1 ## Checa os primeiros registros
2 df_pd.head(10)
```

	Cod_Empresa	Empresa	Dt_Negociacao	Abertura	Max	Min	Fechamento	Volume_Negociado	Valor_Mercado_Empresa
0	ANC	Anoncoin	2013-12-27	5.21000000	5.39000000	4.64000000	4.81000000	16,203.00000000	2,975,246.00000000
1	BTC	Bitcoin	2013-12-27	783.28000000	777.51000000	713.60000000	735.07000000	46,862,700.00000000	8,955,395,000.00000000
2	DMD	Diamond	2013-12-27	2.29000000	2.50000000	1.49000000	1.75000000	8,360.00000000	341,344.00000000
3	DOGE	Dogecoin	2013-12-27	0.00060300	0.00062820	0.00049690	0.00052190	477,422.00000000	8,016,604.00000000
4	LTC	Litecoin	2013-12-27	24.81000000	25.27000000	22.26000000	23.27000000	31,112,200.00000000	566,088,060.00000000
5	TRC	Terracoin	2013-12-27	0.50380000	0.52510000	0.45000000	0.47780000	31,077.00000000	2,268,590.00000000

Checa e exclui duplicidade de registros

```
[ ] 1 # Verifica total de linhas duplicadas  
2 df_pd.duplicated().sum()
```

0

```
[ ] 1 # Exclui registros duplicados  
2 df_pd = df_pd.drop_duplicates()  
3  
4 ## Verifica tamanho da base de dados após dropagem  
5 df_pd.shape
```

(1249976, 10)

Análise da estrutura do DataFrame

```
1 # Obtém informações detalhadas da estrutura do DF como:
2 # tipos de dados por campo, e qtos possuem dados NÃO NULOS, qtde de linhas e de colunas
3 df_pd.info()
4
5 # insight da análise: "Dt_Negociacao deve ser convertida de Object para DateTime"
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1249976 entries, 0 to 1249975
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Cod_Empresa            1235941 non-null object
1   Empresa                1147140 non-null object
2   Dt_Negociacao          1249976 non-null object
3   Abertura               1249976 non-null float64
4   Max                    1249976 non-null float64
5   Min                    1249976 non-null float64
6   Fechamento            1249976 non-null float64
7   Volume_Negociado       1249976 non-null float64
8   Valor_Mercado_Empresa  1249976 non-null float64
9   Ano                    1249976 non-null int64
dtypes: float64(6), int64(1), object(3)
memory usage: 104.9+ MB
```


Tratamento da coluna Data

```
1 #Converte coluna de object para DateTime (formato Y-M-D)
2 df_pd['Dt_Negociacao'] = pd.to_datetime(df_pd['Dt_Negociacao'], format="%Y-%m-%d", \
3                                         errors='coerce')
```

```
1 #Altera a ordem da coluna Dt_Negociacao
2 list_columns = list(df_pd.columns)
3 list_columns.remove('Dt_Negociacao')
4 new_list_columns = ['Dt_Negociacao'] + list_columns
5
6 #Refaz o DF na ordem correta de colunas
7 df_pd = df_pd.reindex(new_list_columns, axis=1)
```

```
1 # Visualiza DF atualizado
2 df_pd.head()
```

	Dt_Negociacao	Cod_Empresa	Empresa	Abertura	Max	Min	Fechamento
0	2013-12-27	ANC	Anoncoin	5.21000000	5.39000000	4.64000000	4.81000000
1	2013-12-27	BTC	Bitcoin	763.28000000	777.51000000	713.60000000	735.07000000
2	2013-12-27	DMD	Diamond	2.29000000	2.50000000	1.49000000	1.75000000
3	2013-12-27	DOGE	Dogecoin	0.00060300	0.00062820	0.00049690	0.00052190

Estatística de campos de valor

```
# Apresenta estatística no ano determinado
df_2019 = df_pd[["Abertura", "Min", "Max", "Fechamento", "Volume_Negociado", \
                "Valor_Mercado_Empresa", "Ano" ]].query('Ano == 2019 ')

df_2019[["Abertura", "Min", "Max", "Fechamento", "Volume_Negociado", \
        "Valor_Mercado_Empresa" ]].describe()
```

	Abertura	Min	Max	Fechamento	Volume_Negociado	Valor_Mercado_Empresa
count	187,280.000	187,280.000	187,280.000	187,280.000	187,280.000	187,280.000
mean	24.478	23.881	25.204	24.482	108,612,499.007	412,507,469.112
std	421.795	411.507	440.743	421.759	1,238,108,753.815	6,288,329,620.598
min	0.000	0.000	0.000	0.000	1.000	1,039.000
25%	0.005	0.004	0.005	0.005	10,663.000	1,162,226.500
50%	0.029	0.027	0.031	0.028	179,460.500	5,682,177.500
75%	0.185	0.172	0.197	0.185	1,600,123.250	22,069,310.000
max	13,017.120	12,233.260	40,826.500	13,016.230	53,509,130,000.000	231,462,110,000.000

Spark / SparkSQL

Conexão da Session

```
1 #SEÇÃO GCP
2 spark = (
3     SparkSession.builder
4         .master ('local')
5         .appName('proj_final') #nome da session para o projeto
6         .config('spark.ui.port','4050')
7         .config('spark.jars','https://storage.googleapis.com/hadoop-lib/gcs/gcs-c
8         .getOrCreate()
9 )
10
11 #testa se a conexao com spark foi realizada com sucesso, e ver versão
12 spark
```

Criação do Schema

```
1 #Formando a Estrutura do DataFrame em pySpark
2 esquema = {
3     StructType([
4         StructField('Cod_Empresa', StringType(), True),
5         StructField('Empresa', StringType(), True),
6         StructField('Dt_Negociacao', DateType(), False),
7         StructField('Abertura', FloatType(), True),
8         StructField('Max', FloatType(), True),
9         StructField('Min', FloatType(), True),
10        StructField('Fechamento', FloatType(), True),
11        StructField('Volume_Negociado', FloatType(), True),
12        StructField('Valor_Mercado_Empresa', FloatType(), True)
13    ])
14 }
```

Cria DataFrame

```
1 # Cria DataFrame com StructType do pySpark, captando a base CSV do GCP/Bue
2 df_spark = (
3     spark.read.format('csv')
4         .option('header', True)
5         .option('delimiter', ',')
6         .option('inferSchema', True)
7         .load(path_tratados, schema=esquema)
8 )
```

Cria arquivo Parquet

```
1 # SPARK - CRIA Arquivo Parquet
2
3 # Converter um DataFrame Pandas, num DataFrame Spark
4 spark.conf.set("spark.sql.execution.arrow.enabled", "true")
5 df_spark = spark.createDataFrame(df_spark)
6
7 # Salvo DF Spark em Parquet
8 df_spark.write.parquet(path_parquet)
9
```

```
1 # SPARK - Carrega arquivo Parquet em DataFrame Spark
2 df_spark = spark.read.parquet(path_parquet)
```

Comparativo entre arquivo CSV e Parquet (databricks.com)

Formato	Espaço utilizado	Tempo de execução	Escaneado
CSV	1 TB	236 seg	1.15 TB
Parquet	130 GB	6.78 seg	2.51 GB
RESUMO DO DESEMPENHO PARQUET			
Redução do Espaço			87%
Velocidade (vezes mais rápido)			34

Visualização DataFrame Spark

```
: #Visualiza o DataFrame pySpark  
df_spark.show(15)
```

Cod_Empresa	Empresa	Dt_Negociacao	Abertura	Max	Min	Fechamento	Volume_Negociado	Valor_Mercado_Empresa	Ano
RSR	Reserve Rights	2021-01-04	0.03263	0.03521	0.02686	0.03066	2.31849744E8	2.86691712E8	2021
RUFF	Ruff	2021-01-04	0.005884	0.00857	0.005709	0.007571	3069577.0	7423676.0	2021
RUNE	THORChain	2021-01-04	1.48	1.59	1.31	1.59	4.1093944E7	2.51942752E8	2021
SAFE	Safe	2021-01-04	0.2123	0.2379	0.1805	0.2153	102844.0	4484890.0	2021
SALTSALT	null	2021-01-04	0.3795	0.4356	0.3708	0.4356	152177.0	3.4262344E7	2021
SAN	Santiment Network...	2021-01-04	0.09095	0.09384	0.08313	0.08369	39297.0	5293463.0	2021
SAND	The Sandbox	2021-01-04	0.04102	0.04242	0.03696	0.04049	7083381.0	2.5178244E7	2021
SAPP	Sapphire	2021-01-04	0.0693	0.07074	0.05985	0.067	118050.0	3.1080598E7	2021
SBD	Steem Dollars	2021-01-04	2.84	2.91	2.54	2.71	3398754.0	1.6045217E7	2021
SC	Siacoin	2021-01-04	0.004796	0.004853	0.003945	0.004396	3.7314148E7	1.98966784E8	2021
SCC	StakeCubeCoin	2021-01-04	0.5368	0.5945	0.477	0.5917	20535.0	4765194.0	2021
SCP	ScPrime	2021-01-04	0.1278	0.1304	0.09174	0.09928	8524.0	3147831.0	2021
SCR	Scorum Coins	2021-01-04	0.007771	0.0131	0.005407	0.007096	2841.0	207659.0	2021
SCRT	Secret	2021-01-04	0.5577	0.6214	0.3629	0.6148	809616.0	3.4674248E7	2021
SENSOSENSO	null	2021-01-04	0.3018	0.3037	0.2962	0.3011	967257.0	3686039.0	2021

only showing top 15 rows

Algumas análises

```
[74]: # Verificando o maior Volume_Negociado
df_spark.select(F.max("Volume_Negociado").alias("Maior_Volume_Negociado_US$")).show()
```

```
+-----+
|Maior_Volume_Negociado_US$|
+-----+
|          3.50967955E11|
+-----+
```

```
[76]: # Verificando o maior Valor_Mercado_Empresa
df_spark.select(F.max("Valor_Mercado_Empresa").alias("Maior_Valor_Mercado_Empresa_US$")).show()
```

```
+-----+
|Maior_Valor_Mercado_Empresa_US$|
+-----+
|          6.5313069E13|
+-----+
```

```
[75]: # Verificando o menor Volume$ Negociado
df_spark.select(F.min("Volume_Negociado").alias("Menor_Volume_Negociado_US$")).show()
```

```
+-----+
|Menor_Volume_Negociado_US$|
+-----+
|              1.0|
+-----+
```


Algumas análises

```
# Lista registros com valor de Abertura <= Zero (0)
df_spark.select( F.col('Cod_Empresa'), F.col('Empresa'), round(F.col('Abertura'),5), round(F.col('Fechamento'),5)
    .where(F.col('Abertura') <= 0) \
    .orderBy( F.col('Abertura').desc()).show(20)
```

```
# Agrupa informações, conta e Ordena decrescente
df_spark.groupBy( F.col("Empresa") ).count().orderBy(F.col("Empresa").desc()).show(10);
```

```
# SPARK (SELECT, DISTINCT, ORDERBY, ASC, DESC, F.COL) SELECIONANDO VALORES DISTINTOS ORDENADOS ALFABETICAMENTE
df_spark.select( 'Empresa', 'Valor_Mercado_Empresa').distinct().orderBy(F.col('Valor_Mercado_Empresa').asc()).
```

```
# Algumas Análises sobre o Min e Max
```

```
df_spark.select( F.col('Cod_Empresa'), F.col('Empresa'), F.col('Abertura'), F.col('Max'), \
    F.col('Min'), F.col('Fechamento'), F.col('Volume_Negociado'), F.col('Valor_Mercado_Empresa') ) \
    .orderBy(F.col('Min'), F.col('Max')).show(5)
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
|Cod_Empresa|Empresa|Abertura|    Max|    Min|Fechamento|Volume_Negociado|Valor_Mercado_Empresa|
+-----+-----+-----+-----+-----+-----+-----+-----+
|      SPOR|  Spore| 2.7E-11|3.3E-11|2.5E-11|   2.9E-11|          3310.0|        1042488.0|
|      SPOR|  Spore| 3.2E-11|3.2E-11|2.7E-11|   2.7E-11|          1765.0|        967044.0|
|      SPOR|  Spore| 2.9E-11|3.1E-11|2.9E-11|   3.1E-11|          3281.0|       1097600.0|
|      SPOR|  Spore| 3.1E-11|3.2E-11|3.0E-11|   3.2E-11|          2249.0|       1126016.0|
|      SPOR|  Spore| 3.2E-11|3.3E-11|3.0E-11|   3.1E-11|          2551.0|       1087170.0|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

Selecionando colunas e filtrando registros

```
# Seleciona período de data_negociação e código_empresa específicas com condicional (WHERE)
df_spark.select( F.col("Dt_Negociacao"), F.col("Cod_Empresa"), F.col("Empresa"), \
                 F.col("Volume_Negociado"), F.col("Valor_Mercado_Empresa") ) \
    .where( ((F.col("Dt_Negociacao") >= '2022-01-01') & (F.col("Dt_Negociacao") <= '2022-05-31')
            & (F.col("Cod_Empresa") < 'BBB') ) \
    .orderBy(F.col('Dt_Negociacao').desc()).show(10)
```

[Stage 49:>

(0 + 4) / 4]

Dt_Negociacao	Cod_Empresa	Empresa	Volume_Negociado	Valor_Mercado_Empresa
2022-05-01	AAVE	Aave	2.1493816E8	2.00780134E9
2022-05-01	ADD	Add.xyz	2.0	829701.0
2022-05-01	ABT	Arcblock	967788.0	1.3024856E7
2022-05-01	ACA	Acala Token	1.5200707E7	3.59167968E8
2022-05-01	ACENTACE	null	1318497.0	5906972.0
2022-05-01	ACK	AcknoLedger	14140.0	959150.0
2022-05-01	ACM	AC Milan Fan Token	1.8003992E7	1.8066448E7
2022-05-01	ACT	Achain	783204.0	5079028.0
2022-05-01	ADA	Cardano	9.2844826E8	2.66761544E10
2022-05-01	ADAPAD	ADAPad	142999.0	3519057.0

only showing top 10 rows

Algumas Seleções e filtros

#Identifica o início de possíveis datas nulas para posterior dropagem

```
df_spark.select(F.col('Dt_Negociacao'), F.col('Cod_Empresa'), F.col('Empresa'), F.col('Abertura'), \
                F.col('Fechamento'), F.col('Volume_Negociado'), F.col('Valor_Mercado_Empresa'), F.col('Ano')) \
                .orderBy(F.col('Dt_Negociacao'), F.col('Empresa')).show(5)
```

[Stage 59:=====>

(1 + 3) / 4]

Dt_Negociacao	Cod_Empresa	Empresa	Abertura	Fechamento	Volume_Negociado	Valor_Mercado_Empresa	Ano
2013-12-27	XRP	null	0.02443	0.02708	148422.0	2.11674064E8	2013
2013-12-27	ANC	Anoncoin	5.21	4.81	16203.0	2975246.0	2013
2013-12-27	BTC	Bitcoin	763.28	735.07	4.68627E7	8.9553951E9	2013
2013-12-27	DMD	Diamond	2.29	1.75	8360.0	341344.0	2013
2013-12-27	DOGE	Dogecoin	6.03E-4	5.219E-4	477422.0	8016604.0	2013

only showing top 5 rows

#Conta registros NULL nas colunas especificadas

```
qtde = df_spark.filter(F.col('Dt_Negociacao').isNull() | F.col('Abertura').isNull()).count()
```

```
print(qtde)
```

0

Upload do DataFrame tratado

```
#Exporta base de dados (DataFrame SPARK/PARQUET) Tratada para o Bucket/GCP (com Valor Mercado Maior Zero)  
df_SparkValorMercadoMaiorZero.toPandas().to_csv(path_tratados+'Base_Tratada_SparkValorMercadoMaiorZeroF.csv',  
          index=False)  
  
print(">>> Exportação do DF Spark para Bucket/GCP, concluído com sucesso.")
```

Média, Máximo, Mínimo

8.3-SPARK_SQL (GROUPBY, AGG, ORDERBY, SUM, MEAN, MAX, ROUND)

```
# MOSTRAR ALGUNS ÍNDICES DOS DADOS: Abertura, Fechamento, Min, Max, Volume_Negociado, Valor_Mercado_Empresa
df_SparkSql.groupBy(F.col('Empresa')).agg( round(F.mean('Abertura'),3), round(F.mean('Fechamento'),3), round(F.mean('Min'),3), round(F.mean('Max'),3), \
                                           F.max('Volume_Negociado'), \
                                           F.max('Valor_Mercado_Empresa'), \
                                           ).orderBy(F.col('Empresa')).show(10)
```

```
[Stage 70: <=====> (133 + 4) / 200]
```

Empresa	round(avg(Abertura), 3)	round(avg(Fechamento), 3)	round(avg(Min), 3)	round(avg(Max), 3)	max(Volume_Negociado)	max(Valor_Mercado_Empresa)	
null	244.255		244.56	228.029	269.795	3.6955177E10	1.3085347E11
Chain	0.023	0.023	0.021	0.025	1.21326822E9	2.38228432E8	
Coin	0.131	0.131	0.119	0.143	5574248.0	3.2283772E7	
Datalink	0.443	0.443	0.413	0.48	3.17522176E8	7.6608136E7	
Exchange Token	0.301	0.302	0.294	0.306	1628974.0	1340704.0	
Fan Token	4.989	4.986	4.661	5.438	1.02298336E8	2.374796E7	
Finance	0.357	0.354	0.335	0.379	4.2214396E7	3.52402848E8	
Gaming	0.0	0.0	0.0	0.0	6.9875024E7	9.6041594E8	
Group	0.014	0.014	0.012	0.016	3.5096768E7	1.026628E8	
Lend	0.338	0.334	0.311	0.403	5893220.0	1.3906682E7	

only showing top 10 rows

Trabalhando com View

```
## Criando uma VIEW do DataSet Tratado para processamento de análises mais rápido
df = (spark
      .read
      .format("csv")
      .option("header", "true")
      .option("inferSchema", "true")
      .option("delimiter", ",")
      .load(path_tratados)
      .createOrReplaceTempView("VIEW_Spark_Tratada"))
```

```
# VIEW - Filtra registros por período de Data_Negociação e ordena por Data + Cod_Empresa
spark.sql('''SELECT Ano, Cod_Empresa, Volume_Negociado, Valor_Mercado_Empresa
            FROM VIEW_Spark_Tratada
            WHERE Dt_Negociacao >= "2020-01-01" AND Dt_Negociacao <= "2020-01-31"
            ORDER BY Dt_Negociacao ASC, Cod_Empresa DESC''').show(5)
```

[Stage 79:=====> (3 + 1) / 4]

Ano	Cod_Empresa	Volume_Negociado	Valor_Mercado_Empresa
2020	ZSC	5457.0	282529.0
2020	ZRX	1.0396732E7	1.10582024E8
2020	ZPT	30.0	599404.0
2020	ZNT	12066.0	14756.0
2020	ZIL	4366624.0	4.4445668E7

only showing top 5 rows

Exportação para MongoDB Atlas (cloud)

```
#Cria Conexão com o Servidor MongoDB Atlas
print('Conexão Servidor MongoDB Atlas.')
myurl = "mongodb+srv://soulcode:a1b2c3@cluster-proj-final.uj7gz.mongodb.net/db_criptomoeda.Criptomoeda_Sp
client = MongoClient(myurl)

#STATUS DO SERVIDOR CLIENT
print('Status do servidor Client do MongoDB Atlas.')
print(client.stats )

#Conectando com o Banco de Dados
print('Conecta ao banco de dados.')
db = client.db_criptomoeda

# converte de pyspark df para pandas df
print('Converte DF de pySpark para DF Pandas.')
df_pd_tratado = df_SparkSql.toPandas()

#Converte Dt_Negociacao em string
df_pd_tratado['Dt_Negociacao'] = df_pd_tratado['Dt_Negociacao'].astype('datetime64[ns]')
df_pd_tratado['Dt_Negociacao'] = df_pd_tratado['Dt_Negociacao'].dt.strftime('%Y-%m-%d')

#Converte de DF para Dicionário
print('Converte DF para Dict.')
data_dict = df_pd_tratado.to_dict(orient='records')

#Insere coleção (json) no MongoDB
print('Cria coleção no banco MongoDB.')
db.Criptomoeda_SparkSQL_Tratada.insert_many(data_dict)

print('\n>>> Dados inseridos com sucesso no banco MongoDB Atlas.')
```

Base tratada - Exportada para o MongoDB Atlas

The screenshot displays the MongoDB Atlas web interface. The top navigation bar includes the user profile 'Aldreks Albu...', 'Access Manager', and 'Billing'. The main header shows 'proj_final' and 'Atlas' tabs. The left sidebar contains sections for 'DEPLOYMENT', 'Database', 'Data Lake', 'DATA SERVICES', and 'SECURITY'. The main content area is titled 'Cluster-Proj-Final' and shows the 'Collections' tab. The collection 'db_criptomoeda.Criptomoeda_SparkSQL_Tratada' is selected, displaying its size (3.14 MB) and document count (14,996). A search bar is visible, and a query result is shown at the bottom.

Database: **Cluster-Proj-Final** (Version: 5.0.9, Region: AWS Sao Paulo (sa-east-1))

Overview | Real Time | Metrics | **Collections** | Search | Profiler | Performance Advisor | Online Archive | Cmd Line Tools

DATABASES: 1 COLLECTIONS: 2

+ Create Database

Search namespaces

db_criptomoeda

- Criptomoeda_Pandas_...
- Criptomoeda_SparkS...**

db_criptomoeda.Criptomoeda_SparkSQL_Tratada

SIZE: 3.14 MB | TOTAL DOCUMENTS: 14,996 | INDEXES: 1 (TOTAL SIZE: 373.9 KB)

Find | Indexes | Schema And Patterns | Aggregation | Search Indexes

INSERT DOCUMENT

QUERY RESULTS: 1-20 OF MANY

```
{
  "_id": ObjectId("62a29f12bf78a4b1a3c1a5"),
  "Cod_Empresa": "ABC",
  "Empresa": "Artemis",
  "Dt_Registracao": "2013-11-27",
  "Abertura": 5.7300000000000003
}
```


BigQuery / SQL

Google Cloud Platform BC17

Search big query

Explorer

Type to search

Viewing pinned projects.

- central-point-349020
 - dataset2
 - criptomoeda
- bigquery-public-data

MORE RESULTS

tabcripto..._eda x *Unsaved...y 2 x

RUN SAVE SHARE SCHEDULE MORE

```
1 SELECT Dt_Negociacao, Cod_Empresa, Volume_Negociado, Valor_Mercado_Empresa
2 FROM `central-point-349020.dataset2.criptomoeda`
3 WHERE Ano >= 2022
4 ORDER BY Ano, Empresa, Volume_Negociado DESC LIMIT 10;
```

Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	Dt_Negociacao	Cod_Empresa	Volume_Negociado	Valor_Mercado_Empresa
1	2022-02-08	XRP	6159382500.0	42006170000.0
2	2022-02-24	XRP	5146799000.0	33385746000.0
3	2022-01-07	BNB	4613172000.0	74691630000.0
4	2022-01-11	BNB	4600616000.0	77315390000.0
5	2022-01-06	BNB	4400229000.0	78942910000.0
6	2022-01-22	BNB	4226932500.0	59127680000.0
7	2022-02-07	XRP	4197403100.0	39596605000.0

08

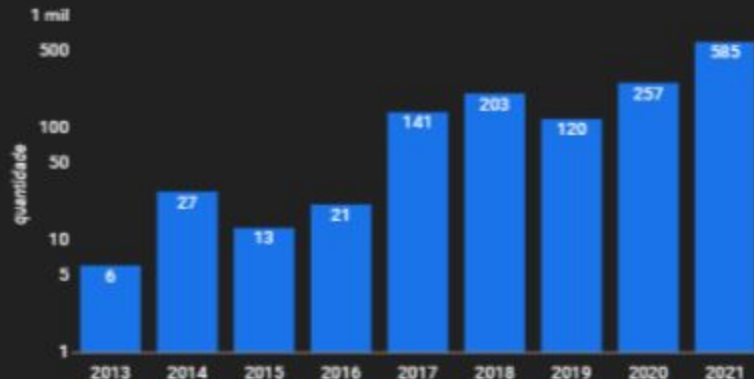
Insights / Datastudio

Insights

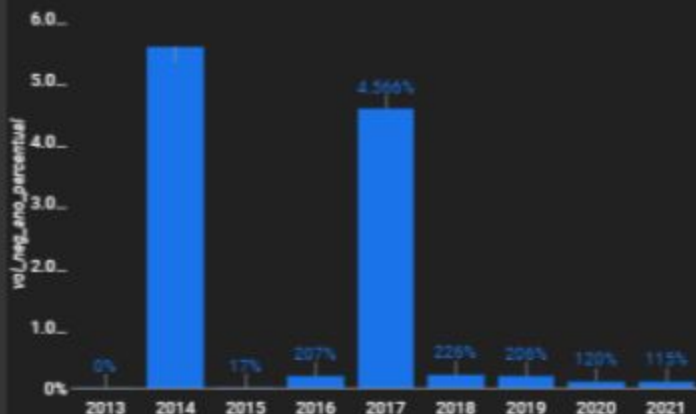
Volume Negociado das Criptomoedas / Ano



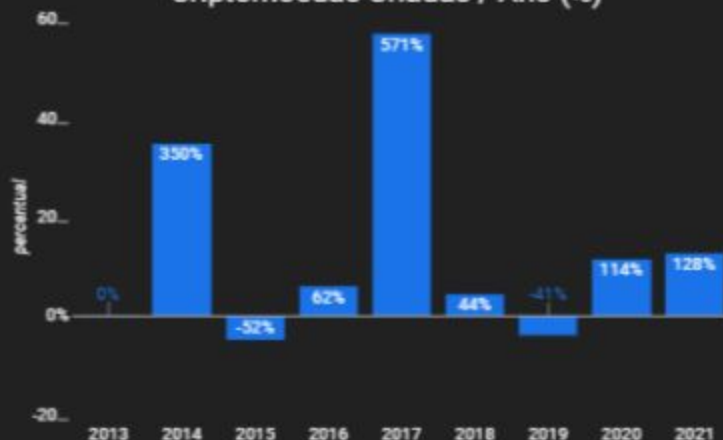
Criptomoedas Criadas / Ano



Volume Negociado das Criptomoedas / Ano (%)



Criptomoedas Criadas / Ano (%)



Rentabilidade



Volatilidade



Vantagens

Diversificação
Segurança
Descentralização
Alta liquidez
Volatilidade



VS

Desvantagens

Baixa aceitação
Falta de regulamentação
Mercado novo
Volatilidade



09

Análise de Custo

Estimativa

Tecnologia	Valores
SQL Cloud	USD 270.46
BigQuery	USD 0.00
Bucket Storage	USD 0.20 / GB
MongoDB	BRL 0.06 / h
JupyterLab	USD 202.09
Estimativa Mensal: USD 516	

10

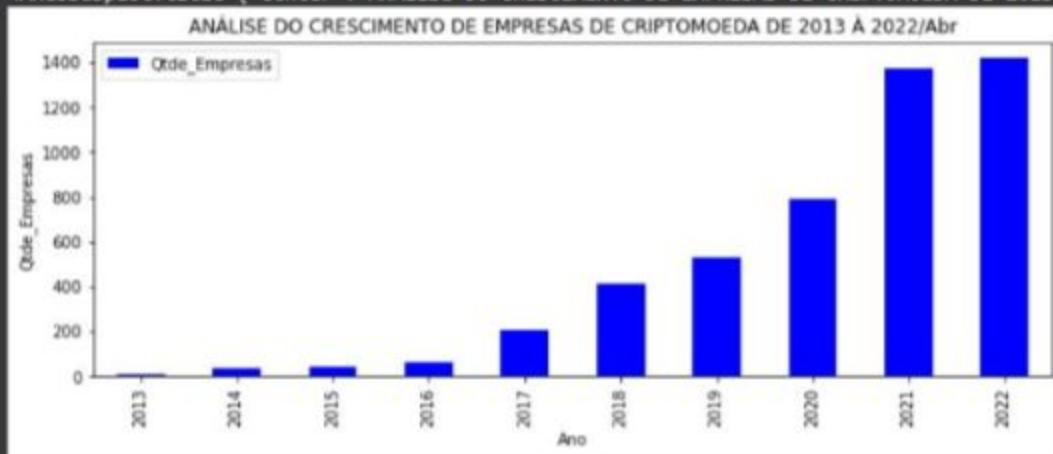
Requisitos Desejáveis

Requisito Desejável 3

Plotagem Pandas

```
1 #MODELO 3
2 #Monta Serie com as duas colunas, para plotar evolução de empresas por ano
3 serie_ano_empresa_count = df_pd[ ["Ano", "Empresa"] ].groupby(['Ano', 'Empresa'])['Empresa'].count()
4 serie_ano_empresa_count = serie_ano_empresa_count.rename('count')
5 df_ano_empresa_count = serie_ano_empresa_count.to_frame().reset_index()
6 groupby_empresa_count = df_ano_empresa_count[["Ano", "Empresa"]].groupby(['Ano'])['Ano']
7 serie_empresa_count = groupby_empresa_count.count().rename('Qtde_Empresas')
8 df_empresa_count = serie_empresa_count.to_frame()
9 #GRAFICO
10 df_empresa_count.plot.bar(title='ANÁLISE DO CRESCIMENTO DE EMPRESAS DE CRIPTOMOEDA DE 2013 À 2022/Abr', \
11                             figsize=(11,4), xlabel='Ano', ylabel='Qtde_Empresas', color='blue')
```

<AxesSubplot:title={'center':'ANÁLISE DO CRESCIMENTO DE EMPRESAS DE CRIPTOMOEDA DE 2013 À 2022/Abr'}, xlabel='Ano', ylabel='Qtde_Empresas', color='blue', figsize=(11,4)>



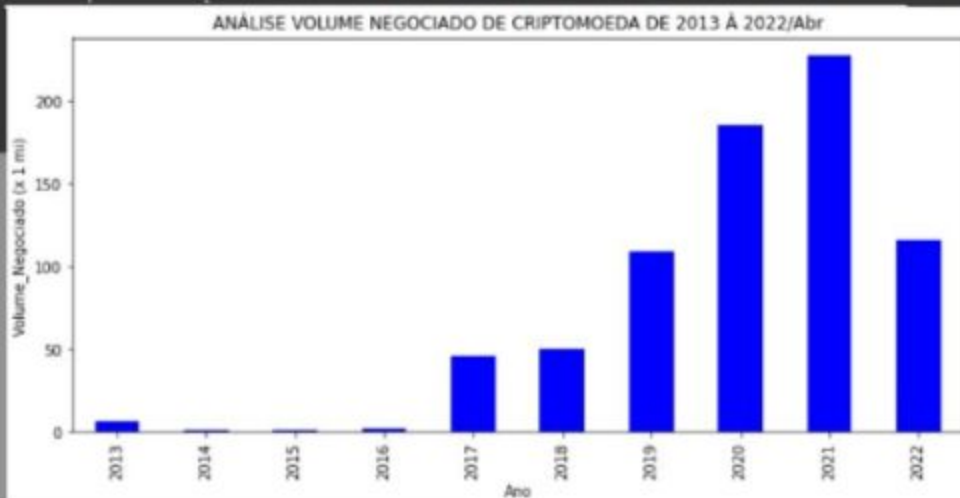
Acumulado

Requisito Desejável 3

Plotagem Pandas

```
1 # MODELO 1
2 #Monta Serie com as duas colunas, para plotar Valor_Mercado_Empresa por ano (X 1.000.000)
3 df_valor_Mercado_Empresa = df_pd[ ["Ano", "Valor_Mercado_Empresa"] ].groupby('Ano')['Valor_Mercado_Empresa'].mean()
4 df_valor_Mercado_Empresa_by_miliao = df_valor_Mercado_Empresa/(10**6)
5
6 # GRÁFICO
7 df_valor_Mercado_Empresa_by_miliao.plot.bar( title='ANÁLISE TOTAL VALOR MERCADO DAS EMPRESAS DE CRIPTOMOEDA DE 2013 À 2022/Abr', \
8                                              figsize=(11,5), xlabel='Ano', ylabel='Valor_Mercado_Empresa (x 1 mi)', color='blue')
```

<AxesSubplot:title=('center': 'ANÁLISE TOTAL VALOR MERCADO DAS EMPRESAS DE CRIPTOMOEDA DE 2013 À 2022/Abr'), xlabel='Ano', ylabel='Valor_Mercado_Empresa (x 1 mi)'



Requisito Desejável 5

Foi realizado o referido item e disponibilizado na pasta ETL do bucket/GCP, e no Classroom.

Obrigado!

