
DOCUMENTAÇÃO

Projeto Final ECONOMIA



REQUISITOS:

- ✓ Obrigatoriamente os datasets devem ter formatos diferentes (CSV / Json / Parquet / Sql / NoSql) e 1 deles obrigatoriamente tem que ser em CSV;
- ✓ Operações com Pandas (limpezas, transformações e normalizações);
- ✓ Operações usando PySpark com a descrição de cada uma das operações;
- ✓ Operações utilizando o SparkSQL com a descrição de cada umas das operações;
- ✓ Os datasets utilizados podem ser em lingua estrangeira, mas devem ao final terem seus dados/colunas exibidos na lingua PT-BR;
- ✓ os datasets devem ser salvos e operados em armazenamento cloud obrigatoriamente dentro da plataforma GCP (não pode ser usado Google drive ou armazenamento alheio ao google);
- ✓ os dados tratados devem ser armazenados também em GCP, mas obrigatoriamente em um datalake(Gstorage), DW(BigQuery) ou em ambos;
- ✓ Deve ser feito análises dentro do Big Query utilizando a linguagem padrão SQL com a descrição das consultas feitas;
- ✓ Deve ser criado no datastudio um dash board simples para exibição gráfica dos dados tratados trazendo insights importantes;
- ✓ E deve ser demonstrado em um workflow simples (gráfico) as etapas de ETL.

ESCOLHA DOS DATA SETS:

Apesar de as notícias sobre um 'possível vírus' terem começado a circular em meados de novembro de 2019, apenas em 11 de março de 2020 é que a OMS (Organização Mundial da Saúde) decretou estado de 'Pandemia Mundial por Coronavírus'. Dentre as medidas de segurança, recomendou o 'isolamento social', conhecido popularmente por 'Quarentena' e consequentemente houve uma brusca desaceleração da atividade econômica em todo o mundo.

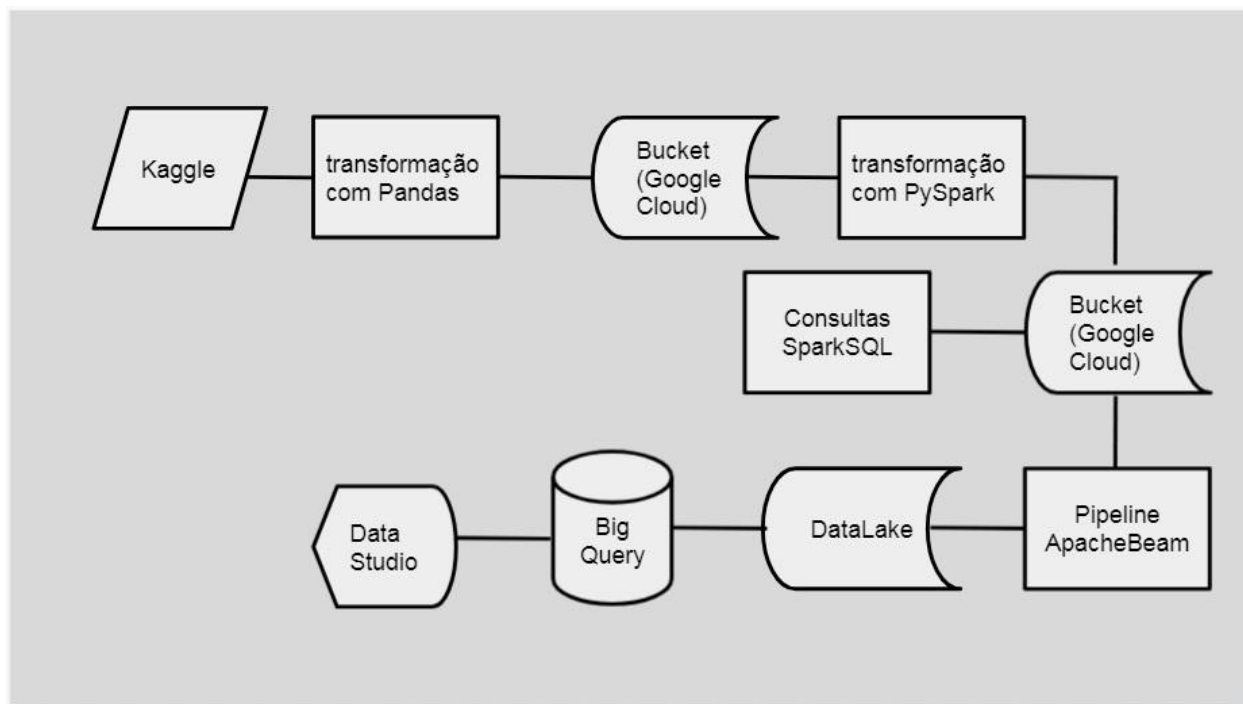
Com base nesse fato, procuramos identificar o comportamento do setor econômico diante desta situação.

Primeiramente buscamos por um dataset com dados sobre o movimento das ações da Bolsa de Valores brasileira, que contemplasse esse período pandêmico, ou seja, entre novembro de 2019 e março de 2020 e, para garantir uma visão mais completa, selecionamos também outros 2 (dois) datasets, estes, contendo dados sobre o mercado de Criptomoedas. Mercado este que sofreu oscilações ainda mais protuberantes durante o período correspondente ao decreto de Pandemia.

FERRAMENTAS UTILIZADAS:



WORKFLOW ETL:



DataSet 'ACOES'

Conexão Google Cloud

- Upload do DataSet bruto para a Cloud Storage
Autenticando o acesso ao Projeto e ao Bucket.

```
from google.colab import auth

project_id = "projeto-grupo2-economia"
bucket_name = "economia-dados-g2"

auth.authenticate_user()

!gcloud config set project {project_id}
```

Backup do DataSet 'Acoes_Bruto.csv' antes dos processos de limpeza direto para a pasta 'entrada' do Bucket.

```
!gsutil cp /content/drive/MyDrive/dados/PROJETO_FINAL/AcoesBruto.csv gs://{bucket_name}/entrada/Acoes_Bruto.csv

Copying file:///content/drive/MyDrive/dados/PROJETO_FINAL/AcoesBruto.csv [Content-Type=text/csv]...
|
Operation completed over 1 objects/42.0 MiB.
```

- Download do DataSet para o Google Colab

```
!gsutil cp gs://{bucket_name}/entrada/Acoes_Bruto.csv /content/drive/MyDrive/dados/PROJETO_FINAL/AcoesBruto.csv

Copying gs://economia-dados-g2/entrada/Acoes_Bruto.csv...
- [1 files][ 42.0 MiB/ 42.0 MiB]
Operation completed over 1 objects/42.0 MiB.
```

ETL com PYTHON e PANDAS

- Instalação Pandera e bibliotecas

```
pip install pandera

import pandas as pd
import pandera as pa
```

- Montando o Dataframe .csv

```
df_acoes = pd.read_csv(r"/content/drive/MyDrive/dados/PROJETO_FINAL/AcoesBruto.csv",
                      sep=';', encoding='ISO8859-1',
                      parse_dates=['ref.date'],
                      dayfirst=True, decimal=',')
```

Comentário 01: Arquivo no formato .csv, o separador neste caso é ';' (ponto e vírgula), uso do encoding = 'ISO8859-1' devido existência de caractere especial nos registros, coluna 'ref.date' definida para o tipo data, as colunas de valores não inteiros tem como separador decimal a virgula.

- Resumo estatístico do Dataframe

df_acoes.describe()								
	price.open	price.high	price.low	price.close	volume	price.adjusted	ret.adjusted.prices	ret.closing.prices
count	298861.000000	298861.000000	298861.000000	298861.000000	2.988610e+05	2.988610e+05	298486.000000	298486.000000
mean	24.864123	25.128140	24.593715	24.865624	2.459531e+06	-6.927078e+03	0.004401	0.003353
std	39.034601	39.191963	38.906102	39.065673	1.109202e+07	5.106503e+05	0.679534	0.451878
min	0.460000	0.480000	0.430000	0.440000	0.000000e+00	-3.087797e+07	-1.079758	-0.990477
25%	6.290000	6.380000	6.170000	6.280000	1.000000e+02	5.990000e+00	-0.009130	-0.009231
50%	14.350000	14.620000	14.050000	14.350000	3.260000e+04	1.367000e+01	0.000000	0.000000
75%	28.610000	29.000000	28.150000	28.610000	1.241855e+06	2.702000e+01	0.008746	0.008596
max	1290.000000	1344.000000	1234.000000	1341.000000	9.395928e+08	1.693193e+06	277.247600	109.555600

- Quantidade de linhas e colunas com descrição

df_acoes.shape	df_acoes.index
(298861, 15)	RangeIndex(start=0, stop=298861, step=1)

```
df_acoes.columns

Index(['price.open', 'price.high', 'price.low', 'price.close', 'volume',
      'price.adjusted', 'ref.date', 'ticker', 'ret.adjusted.prices',
      'ret.closing.prices', 'Empresa', 'Setor', 'Subsetor', 'Tipo',
      'classificação'],
      dtype='object')
```

- Quantidade de dados válidos em cada coluna e o respectivo tipo

df_acoes.count()		df_acoes.dtypes	
price.open	298861	price.open	float64
price.high	298861	price.high	float64
price.low	298861	price.low	float64
price.close	298861	price.close	float64
volume	298861	volume	float64
price.adjusted	298861	price.adjusted	float64
ref.date	298861	ref.date	datetime64[ns]
ticker	298861	ticker	object
ret.adjusted.prices	298486	ret.adjusted.prices	float64
ret.closing.prices	298486	ret.closing.prices	float64
Empresa	298861	Empresa	object
Setor	298861	Setor	object
Subsetor	298861	Subsetor	object
Tipo	298861	Tipo	object
Classificação	298861	Classificação	object
dtype: int64		dtype: object	

Comentário 02: A coluna 'volume' está definida com o tipo 'float64' ao invés de 'int'.

- Cast da coluna 'volume' para o tipo 'inteiro'

```
df_acoes["volume"] = df_acoes["volume"].astype(int)

df_acoes.dtypes

price.open      float64
price.high      float64
price.low       float64
price.close     float64
volume          int64
price.adjusted  float64
ref.date        datetime64[ns]
ticker          object
ret.adjusted.prices float64
ret.closing.prices float64
Empresa         object
Setor           object
Subsetor        object
Tipo            object
Classificação   object
dtype: object
```


- Renomeando as colunas

```
df_acoes.rename(columns={"price.open":"Preco_Abertura",
                        "price.high":"Maior_Precio",
                        "price.low":"Menor_Precio",
                        "price.close":"Preco_Fechamento",
                        "volume":"Volume",
                        "price.adjusted":"Preco_Ajustado",
                        "ref.date":"Data_Referencia",
                        "ticker":"Codigo_Acao",
                        "ret.adjusted.prices":"Ret_Precio_Ajustado",
                        "ret.closing.prices":"Ret_Precio_Fechamento",
                        "Classificação":"Classificacao"},inplace = True)
```

(verificando se as colunas foram devidamente re-nomeadas e se os dados estão com o tipo definido)

```
df_acoes.dtypes
```

```
Preco_Abertura      float64
Maior_Precio        float64
Menor_Precio        float64
Preco_Fechamento    float64
Volume              int64
Preco_Ajustado      float64
Data_Referencia     datetime64[ns]
Codigo_Acao         object
Ret_Precio_Ajustado  float64
Ret_Precio_Fechamento float64
Empresa            object
Setor              object
Subsetor           object
Tipo              object
Classificacao      object
dtype: object
```

- Contagem de registros 'nulos'

```
df_acoes.isnull().sum()
```

```
Preco_Abertura      0
Maior_Preco         0
Menor_Preco         0
Preco_Fechamento   0
Volume              0
Preco_Ajustado      0
DataReferencia      0
Codigo_Acao         0
Ret_Preco_Ajustado  375
Ret_Preco_Fechamento 375
Empresa             0
Setor               0
Subsetor            0
Tipo                0
Classificacao       0
dtype: int64
```

- Contagem de registros 'NA'

```
df_acoes.isna().sum()
```

```
Preco_Abertura      0
Maior_Preco         0
Menor_Preco         0
Preco_Fechamento   0
Volume              0
Preco_Ajustado      0
DataReferencia      0
Codigo_Acao         0
Ret_Preco_Ajustado  375
Ret_Preco_Fechamento 375
Empresa             0
Setor               0
Subsetor            0
Tipo                0
Classificacao       0
dtype: int64
```

- Filtrando os registros 'nulos' e 'NA'

```
filter_Fechamento = df_acoes.Ret_Preco_Fechamento.isnull()
filter_Ajustado = df_acoes.Ret_Preco_Ajustado.isnull()
df_acoes.loc[filter_Ajustado]
df_acoes.loc[filter_Fechamento]
```

- Contando registros únicos nas colunas mais relevantes

```
pd.unique(df_acoes['Empresa'])
```

(exibe o nome de todos os registros unicos na coluna 'Empresa')

```
print('--'*35)
cont = 0
for nome in (df_acoes['Empresa'].unique()):
    cont += 1
print(f'Total de {cont} empresas distintas.')
```

(utilizando laço de repetição para contabilizar o total de registros unicos na coluna 'Empresa')

```
-----
Total de 283 empresas distintas.
```

```
pd.unique(df_acoes['Setor'])
```

```
pd.unique(df_acoes['Subsetor'])
```

```
pd.unique(df_acoes['Tipo'])
```

```
pd.unique(df_acoes['Classificacao'])
```

(exibe os registros únicos das demais colunas)

Pandera

- Validando o schema da estrutura do dataset com Pandera

```
schema_df = {
    'Preco_Abertura':pa.Column(pa.Float),
    'Maior_Preco':pa.Column(pa.Float),
    'Menor_Preco':pa.Column(pa.Float),
    'Preco_Fechamento':pa.Column(pa.Float),
    'Volume':pa.Column(pa.Int),
    'Preco_Ajustado':pa.Column(pa.Float),
    'Data_Referencia':pa.Column(pa.DateTime),
    'Codigo_Acao':pa.Column(pa.String),
    'Ret_Preco_Ajustado':pa.Column(pa.Float, nullable=True),
    'Ret_Preco_Fechamento':pa.Column(pa.Float, nullable=True),
    'Empresa':pa.Column(pa.String),
    'Setor':pa.Column(pa.String),
    'Subsetor':pa.Column(pa.String),
    'Tipo':pa.Column(pa.String),
    'Classificacao':pa.Column(pa.String)
}
```

```
schema = pa.DataFrameSchema(columns=schema_df)
schema.validate(df_acoes)
```

Load do Dataset tratado para GCP

```
df_acoes.to_csv('/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pandas.csv', index = False)
```

```
!gsutil cp /content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pandas.csv
gs://{bucket_name}/saida/dados_pandas/acoes_pandas.csv
```

```
Copying file:///content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pandas.csv [Content-Type=text/csv]...
-
Operation completed over 1 objects/40.0 MiB.
```

ETL com PYSPARK

- Instalação do Pyspark

```
pip install pyspark
```

- Importando as bibliotecas necessárias

```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import *
import datetime
from pyspark.sql.window import Window
```

- Ingestão do Dataset previamente tratado com Pandas, direto do bucket

```
!gsutil cp gs://{bucket_name}/saida/dados_pandas/acoes_pandas.csv
/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_tratado_pandas.csv
```

```
Copying gs://economia-dados-g2/saida/dados_pandas/acoes_pandas.csv...
- [1 files][ 40.0 MiB/ 40.0 MiB]
Operation completed over 1 objects/40.0 MiB.
```

- Configurando Sparksession para leitura do Dataset

```
spark = (SparkSession.builder
        .master("local")
        .appName("ProjetoFinalG2Economia")
        .config("spark.ui.port", "4050")
        .getOrCreate())
```

- Lendo o Dataset com o seu próprio schema

```
df = spark.read.csv(r"/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_tratado_pandas.csv",
                    header=True, inferSchema=True)
```

```
df.printSchema()
```

```
root
|-- Preco_Abertura: double (nullable = true)
|-- Maior_Precio: double (nullable = true)
|-- Menor_Precio: double (nullable = true)
|-- Preco_Fechamento: double (nullable = true)
|-- Volume: integer (nullable = true)
|-- Preco_Ajustado: double (nullable = true)
|-- Data_Referencia: string (nullable = true)
|--Codigo_Acao: string (nullable = true)
|-- Ret_Precio_Ajustado: double (nullable = true)
|-- Ret_Precio_Fechamento: double (nullable = true)
|-- Empresa: string (nullable = true)
|-- Setor: string (nullable = true)
|-- Subsetor: string (nullable = true)
|-- Tipo: string (nullable = true)
|-- Classificacao: string (nullable = true)
```

(algumas colunas vieram com o tipo diferente do desejado)

- Redefinindo o schema da estrutura do Dataset com StructType

```
schema = StructType([
    StructField("Preco_Abertura",FloatType(),True),
    StructField("Maior_Precio",FloatType(),True),
    StructField("Menor_Precio",FloatType(),True),
    StructField("Preco_Fechamento",FloatType(),True),
    StructField("Volume",IntegerType(),True),
    StructField("Preco_Ajustado",FloatType(),True),
    StructField("Data_Referencia",DateType(),True),
    StructField("Codigo_Acao",StringType(),True),
    StructField("Ret_Precio_Ajustado",FloatType(),True),
    StructField("Ret_Precio_Fechamento",FloatType(),True),
    StructField("Empresa",StringType(),True),
    StructField("Setor",StringType(),True),
    StructField("Subsetor",StringType(),True),
    StructField("Tipo",StringType(),True),
    StructField("Classificacao",StringType(),True)])

local=("/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_tratado_pandas.csv")
df_acoes_struct = spark.read.load(local, format="csv", header="true", schema=schema)
```

```
df_acoes_struct.printSchema()
```

```
root
|-- Preço_Abertura: float (nullable = true)
|-- Maior_Preco: float (nullable = true)
|-- Menor_Preco: float (nullable = true)
|-- Preço_Fechamento: float (nullable = true)
|-- Volume: integer (nullable = true)
|-- Preço_Ajustado: float (nullable = true)
|-- Data_Referencia: date (nullable = true)
|--Codigo_Acao: string (nullable = true)
|-- Ret_Preco_Ajustado: float (nullable = true)
|-- Ret_Preco_Fechamento: float (nullable = true)
|-- Empresa: string (nullable = true)
|-- Setor: string (nullable = true)
|-- Subsetor: string (nullable = true)
|-- Tipo: string (nullable = true)
|-- Classificacao: string (nullable = true)
```

(estrutura já re-definida)

- Estatísticas do Dataset

```
df_acoes_struct.count()
```

```
298861
```

(298.861 linhas com registros)

```
df_acoes_struct.describe().show()
```

(exibe um resumo estatístico do Dataset)

- Contabilizando os registros 'nulos' no Dataset

```
df_acoes_struct.select([F.count(F.when(F.isNull(c), c)).alias(c)
for c in df_acoes_struct.columns]).show()
```

```

+-----+-----+-----+-----+
|Data_Referencia|Codigo_Acao|Ret_Precio_Ajustado|Ret_Precio_Fechamento|
+-----+-----+-----+-----+
|              0|              0|              375|              375|
+-----+-----+-----+-----+

```

(localizados 375 registros 'nulos' nas colunas Ret_Precio_Ajustado e Ret_Precio_Fechamento)

- Redefinindo os registros com valor 'nulo' como valor 0 (zero)

```

df_acoes_struct = df_acoes_struct.fillna(
    {'Ret_Precio_Ajustado':0, 'Ret_Precio_Fechamento':0})

```

```

+-----+-----+-----+-----+
|Data_Referencia|Codigo_Acao|Ret_Precio_Ajustado|Ret_Precio_Fechamento|
+-----+-----+-----+-----+
|              0|              0|              0|              0|
+-----+-----+-----+-----+

```

(agora não existem mais registros com valor 'nulo' nas colunas)

Load do Dataset tratado para GCP

```

df_acoes_struct_pd = df_acoes_struct.toPandas()

```

(convertendo o Dataset para Pandas antes de fazer o load para a GCP)

```

df_acoes_struct_pd.to_csv('/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pyspark.csv', index = False)

```

```

!gsutil cp /content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pyspark.csv
gs://{bucket_name}/saida/dados_pyspark/

```

```

Copying file:///content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pyspark.csv [Content-Type=text/csv]...
-
Operation completed over 1 objects/39.5 MiB.

```


Extraindo informações do Dataset com Pyspark

- Consulta que insere duas colunas no Dataset, chamadas 'Valorizou_No_Dia' e 'Valorização' e exibe as ações que mais valorizaram em todo período contemplado pelo Dataset (01/02/2018 a 30/03/2021) em ordem decrescente da coluna 'Valorizacao'

```
df1=(df_acoes_struct.withColumn('Valorizou_No_Dia',
                                F.when(F.col("Preco_Fechamento")>=F.col("Preco_Abertura"),
                                      F.lit("SIM")).otherwise("NÃO")))

df1=(df1.withColumn('Valorizacao',F.lit(F.col("Preco_Fechamento")-(F.col("Preco_Abertura")))))

df1.select("Valorizou_No_Dia",
           "Preco_Abertura",
           "Preco_Fechamento",
           "Valorizacao","Data_Referencia",
           "Codigo_Acao","Empresa","Setor").orderBy("Valorizacao",ascending=False).show(10)
```

Valorizou_No_Dia	Preco_Abertura	Preco_Fechamento	Valorizacao	Data_Referencia	Codigo_Acao	Empresa	Setor
SIM	162.74	294.1	131.36	2019-08-26	TELB3.SA	TELEBRAS	Comunicações
SIM	80.01	200.0	119.99	2019-08-16	MAPT4.SA	CEMEPE	Outros
SIM	1082.0	1194.0	112.0	2021-03-17	BPAN4.SA	BANCO PAN	Financeiro
SIM	182.88	258.48	75.600006	2018-12-27	GSHP3.SA	GENERALSHOPP	Financeiro
SIM	1267.0	1341.0	74.0	2021-02-23	BPAN4.SA	BANCO PAN	Financeiro
SIM	7.0	68.64	61.64	2019-09-19	GPAR3.SA	CELGPAR	Utilidade Pública
SIM	50.1	111.01	60.910004	2019-08-13	MAPT4.SA	CEMEPE	Outros
SIM	68.0	125.0	57.0	2020-08-24	SNSY3.SA	SANSUY	Materiais Básicos
SIM	40.01	95.01	55.000004	2019-08-13	MAPT3.SA	CEMEPE	Outros
SIM	173.0	228.0	55.0	2020-01-10	CGAS3.SA	COMGAS	Utilidade Pública

only showing top 10 rows

- Consulta que exibe as 10 ações que mais valorizaram ou desvalorizaram em uma data específica. A exibição se dá em ordem decrescente dos registros da coluna 'Valorizacao'. O código recebe a informação da 'data' por input e também o tipo de movimentação, sendo [1] para VALORIZACAO' ou [2] para DESVALORIZACAO.

```
data = input('Digite data no formato aaaa-mm-dd: ')
movimento = int(input('Digite:
[ 1 ] para VALORIZAÇÃO
[ 2 ] para DESVALORIZAÇÃO
: '))
if movimento == 1:
    valorizacao = "SIM"
elif movimento == 2:
    valorizacao = "NÃO"

(df1.select("Codigo_Acao", "Preco_Abertura", "Preco_Fechamento",
            "Valorizou_No_Dia", "Valorizacao")
 .where((F.col("Data_Referencia")==data) & (F.col("Valorizou_No_Dia")==valorizacao))
 .orderBy("Valorizacao",ascending=False).show(10))
```

```
Digite data no formato aaaa-mm-dd: 2020-03-03
Digite:
[ 1 ] para VALORIZAÇÃO
[ 2 ] para DESVALORIZAÇÃO
: 2
```

Codigo_Acao	Preco_Abertura	Preco_Fechamento	Valorizou_No_Dia	Valorizacao
OIBR3.SA	1.02	1.01	NÃO	-0.00999999
MMXM3.SA	2.06	2.05	NÃO	-0.00999999
JB DU4.SA	2.2	2.19	NÃO	-0.00999999
EMBR3.SA	16.92	16.91	NÃO	-0.010000229
ALUP3.SA	12.0	11.99	NÃO	-0.010000229
CAML3.SA	9.01	9.0	NÃO	-0.010000229
ABEV3.SA	14.75	14.74	NÃO	-0.010000229
STBP3.SA	6.42	6.41	NÃO	-0.010000229
BRIV4.SA	9.04	9.03	NÃO	-0.010000229
REDE3.SA	9.85	9.84	NÃO	-0.010000229

only showing top 10 rows

(neste exemplo, o usuario escolhe visualizar as ações que 'desvalorizaram' no dia 03/03/2020)

- Consulta para conferir o movimento de uma ação específica em um dia ou em um período. A variável 'ticker' recebe o Código_Acao por input e também se a consulta será de apenas um dia ou período. Caso o usuário escolha visualizar apenas um dia, ele digita separadamente o dia, o mês e o ano, mas caso escolha um período, deverá digitar a data de início e também a data final no formato 'aaaa-mm-dd'

```

ticker = str(input('Digite Código_Acao: ')).upper().strip()
tempo = int(input('Digite
[ 1 ] pesquisar UNICO DIA ou
[ 0 ] pesquisar por PERIODO '''))
if tempo == 1:
    dia = int(input('Digite Dia com 2 digitos (dd): '))
    mes = int(input('Digite mês com 2 digitos (mm): '))
    ano = int(input('Digite ano com 4 digitos (aaaa): '))
    data = f'{ano}-{mes}-{dia}'
    print()
    print('--'*20)

    df1.select(
        F.col("Data_Referencia"),
        F.col("Codigo_Acao"),
        F.col("Preco_Abertura"),
        F.col("Preco_Fechamento"),
        F.col("Valorizou_No_Dia")).filter(F.col("Codigo_Acao")==ticker).filter(F.col("Data_Referencia")==data).show(10)

else:
    data_inicio = input('Digite data inicial no formato aaaa-mm-dd: ')
    data_fim = input('Digite data final do período no formato aaaa-mm-dd: ')
    print()

```

```

df1.select(
    F.col("Data_Referencia"),
    F.col("Codigo_Acao"),
    F.col("Preco_Abertura"),
    F.col("Preco_Fechamento"),
    F.col("Valorizou_No_Dia")).filter(F.col("Codigo_Acao")==ticker).where(
        (F.col("Data_Referencia")>=data_inicio) & (F.col("Data_Referencia")<=data_fim)).show()

```

```

Digite Codigo_Acao: ITUB4.SA
Digite
[ 1 ] pesquisar UNICO Dia ou
[ 0 ] pesquisar por PERIODO 1
Digite Dia com 2 digitos (dd): 23
Digite mês com 2 digitos (mm): 03
Digite ano com 4 digitos (aaaa): 2020

```

```

-----
+-----+-----+-----+-----+-----+
|Data_Referencia|Codigo_Acao|Preco_Abertura|Preco_Fechamento|Valorizou_No_Dia|
+-----+-----+-----+-----+-----+
|    2020-03-23|   ITUB4.SA|        22.26|         20.52|          NÃO|
+-----+-----+-----+-----+-----+

```

(neste exemplo o usuario escolheu visualizar o movimento da ação 'ITUB4.SA' apenas no dia 23-03-2020)

- Consulta para verificar a quantidade de Codigo_Acao distintos agrupados por Setor

```
df1.groupBy("Setor").agg(F.count_distinct("Codigo_Acao").alias("Qtde_Acoes")).orderBy("Qtde_Acoes",ascending=False).show()
```

```

+-----+-----+
|          Setor|Qtde_Acoes|
+-----+-----+
|  Consumo Cíclico|        81|
|    Financeiro|        70|
|  Bens Industriais|        59|
| Utilidade Pública|        59|
|  Materiais Básicos|        37|
| Consumo não Cíclico|        22|
|        Saúde|        18|
|Petróleo, Gás e B...|        11|
|        Outros|         8|
|    Comunicações|         6|
|Tecnologia da Inf...|         4|
+-----+-----+

```

SparkSQL

- Ingestão do Dataset previamente tratado com Pyspark, direto do bucket

```
!gsutil cp gs://{bucket_name}/saida/dados_pyspark/acoes_pyspark.csv  
/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pyspark.csv
```

```
Copying gs://economia-dados-g2/saida/dados_pyspark/acoes_pyspark.csv...  
- [1 files][ 39.5 MiB/ 39.5 MiB]  
Operation completed over 1 objects/39.5 MiB.
```

- Leitura do Dataset, definindo suas options e criando a tabela temporária chamada 'tabela_acoes'

```
df_acoes_sql = (spark  
    .read  
    .format("csv")  
    .option("header", "true")  
    .option("inferSchema", "true")  
    .option("delimiter", ",")  
    .load("/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pyspark.csv")  
    .createOrReplaceTempView("tabela_acoes"))
```

- Conferindo a estrutura da tabela temporaria 'tabela-acoes'

```
spark.sql(
'''
describe tabela_acoes
'''
).show()
```

col_name	data_type	comment
Preco_Abertura	double	null
Maior_Preco	double	null
Menor_Preco	double	null
Preco_Fechamento	double	null
Volume	int	null
Preco_Ajustado	double	null
Data_Referencia	string	null
Codigo_Acao	string	null
Ret_Preco_Ajustado	double	null
Ret_Preco_Fechamento	double	null
Empresa	string	null
Setor	string	null
Subsetor	string	null
Tipo	string	null
Classificacao	string	null

- Consulta para exibir a data do 1º registro e do ultimo registro

```
spark.sql('''SELECT MIN(Data_Referencia) AS Data_Inicial,
MAX(Data_Referencia) AS Data_Final
FROM tabela_acoes
''').show()
```

```
+-----+-----+
|Data_Inicial|Data_Final|
+-----+-----+
| 2018-01-02|2021-03-30|
+-----+-----+
```

- Consulta que exibe as ações com maior média de valorização de todo o periodo

```
spark.sql('''SELECT DISTINCT Codigo_Acao,  
MEAN(Ret_Preco_Fechamento) AS Media_Valorizacao  
FROM tabela_acoes  
GROUP BY Codigo_Acao  
ORDER BY Media_Valorizacao DESC  
''').show(10)
```

```
+-----+-----+  
|Codigo_Acao|  Media_Valorizacao|  
+-----+-----+  
|  BPAN4.SA| 0.37760877316135505|  
|  PFRM3.SA| 0.3700864396097865|  
|  PPLA11.SA| 0.01819454997214554|  
|  GPAR3.SA|0.014244492693099117|  
|  SNSY3.SA|0.012566410547929742|  
|  MAPT4.SA| 0.00907143868770389|  
|  NORD3.SA| 0.00869473979385194|  
|  IGBR3.SA|0.007730840931994...|  
|  MMXM3.SA|0.007329312337013802|  
|  MERC3.SA|0.007206785348682558|  
+-----+-----+  
only showing top 10 rows
```

- Consulta que exibe as ações com maior média de desvalorização de todo o periodo

```
spark.sql('''SELECT DISTINCT Codigo_Acao,  
MEAN(Ret_Preco_Fechamento) AS Media_Desvalorizacao  
FROM tabela_acoes  
GROUP BY Codigo_Acao  
ORDER BY Media_Desvalorizacao DESC  
''').show(10)
```

```
+-----+-----+  
|Codigo_Acao|Media_Desvalorizacao|  
+-----+-----+  
|BDLL3.SA|-0.00205103147791719|  
|CIEL3.SA|-0.00176718887565...|  
|BPHA3.SA|-0.00162101381053...|  
|SLED4.SA|-0.00146668667126...|  
|COGN3.SA|-0.00124123654542...|  
|JBUD4.SA|-0.00120848628193...|  
|BBRK3.SA|-0.00104089222459...|  
|JBUD3.SA|-9.36546947302383...|  
|RNEW3.SA|-8.42819104893348...|  
|GFS3.SA|-8.30587054705144...|  
+-----+-----+  
only showing top 10 rows
```


- Valor médio de fechamento da bolsa entre os dias 01 e 30 de março de 2019

```
spark.sql('''SELECT Data_Referencia, MEAN(Ret_Preco_Fechamento)
FROM tabela_acoes
WHERE (Data_Referencia >= '2019-03-01' AND Data_Referencia <= '2019-03-30')
GROUP BY Data_Referencia
ORDER BY Data_Referencia
''').show()
```

```
+-----+-----+
|Data_Referencia|mean(Ret_Preco_Fechamento)|
+-----+-----+
|2019-03-01| -0.00334903054799...|
|2019-03-04| 0.0|
|2019-03-05| 0.0|
|2019-03-06| 0.0|
|2019-03-07| -0.00471158876133333|
|2019-03-08| 0.005693498069866664|
|2019-03-11| 0.007823408223733336|
|2019-03-12| 6.483792698666668E-4|
|2019-03-13| 0.002005324732799999|
|2019-03-14| 0.001167123254133...|
|2019-03-15| 0.005169732322506667|
|2019-03-18| 0.0062430539288|
|2019-03-19| -8.53773535999999...|
|2019-03-20| -0.00137325998586...|
|2019-03-21| -0.00796891678106...|
|2019-03-22| -0.01032503691199...|
|2019-03-25| -0.00165564428399...|
|2019-03-26| 0.005911535798666661|
|2019-03-27| -0.02122064322016...|
|2019-03-28| 0.010349898013066667|
+-----+-----+
```

- Valor médio de fechamento da bolsa entre os dias 01 e 30 de março de 2020

```
spark.sql('''SELECT Data_Referencia, MEAN(Ret_Preco_Fechamento)
FROM tabela_acoes
WHERE (Data_Referencia >= '2020-03-01' AND Data_Referencia <= '2020-03-30')
GROUP BY Data_Referencia
ORDER BY Data_Referencia
''').show()
```

```
+-----+-----+
|Data_Referencia|mean(Ret_Preco_Fechamento)|
+-----+-----+
| 2020-03-02 | 0.024836280333599987 |
| 2020-03-03 | -0.00445821360346... |
| 2020-03-04 | 0.00759844385712 |
| 2020-03-05 | -0.03586061367013332 |
| 2020-03-06 | -0.03012584688594666 |
| 2020-03-09 | -0.08570029700533337 |
| 2020-03-10 | 0.04549420326933334 |
| 2020-03-11 | -0.04351318606266668 |
| 2020-03-12 | -0.10320864221973333 |
| 2020-03-13 | 0.059518492781866675 |
| 2020-03-16 | -0.0893158755072 |
| 2020-03-17 | 0.007738628940266674 |
| 2020-03-18 | -0.06313440892521678 |
| 2020-03-19 | 0.018104871441839997 |
| 2020-03-20 | -0.00483178056383... |
| 2020-03-23 | -0.04311139114639... |
| 2020-03-24 | 0.05044575783866666 |
| 2020-03-25 | 0.05912728126106667 |
| 2020-03-26 | 0.04204567402346666 |
| 2020-03-27 | -0.01542235223093... |
+-----+-----+
only showing top 20 rows
```

- Consulta que exibe as 10 ações com melhor fechamento do dia 10/03/2020 (um dia antes de ser decretado Pandemia Mundial por Corona Vírus)

```
spark.sql(''' SELECT Preco_Abertura, Preco_Fechamento, Ret_Preco_Fechamento,
Codigo_Acao, Empresa, Subsetor
FROM tabela_acoes
WHERE Data_Referencia = "2020-03-10"
ORDER BY Ret_Preco_Fechamento DESC
''').show(10)
```

Preco_Abertura	Preco_Fechamento	Ret_Preco_Fechamento	Codigo_Acao	Empresa	Subsetor
3.8	4.5	0.3636364	BSEV3.SA	BIOSEV	Alimentos Process...
1.33	1.61	0.248062	TPIS3.SA	TRIUNFO PART	Transporte
4.14	4.59	0.2249999	ETER3.SA	ETERNIT	Construção e Enge...
10.95	11.62	0.2129436	VVAR3.SA	VIAVAREJO	Comércio
41.9	44.81	0.1845096	VALE3.SA	VALE	Mineração
13.17	14.9	0.1732283	CCRO3.SA	CCR SA	Transporte
11.25	11.75	0.1642716	MGLU3.SA	MAGAZ LUIZA	Comércio
5.55	5.82	0.164	LPSB3.SA	LOPES BRASIL	Exploração de Imó...
6.25	6.71	0.1608997	EUCA4.SA	EUCATEX	Madeira e Papel
16.13	17.88	0.1602855	UGPA3.SA	ULTRAPAR	Petróleo, Gás e B...

only showing top 10 rows

- Consulta que exibe as 10 ações com melhor fechamento no dia 11/03/2020 (dia que foi decretado Pandemia Mundial por Corona Vírus)

```
spark.sql(''' SELECT Preco_Abertura, Preco_Fechamento, Ret_Preco_Fechamento,
Codigo_Acao, Empresa, Subsetor
FROM tabela_acoes
WHERE Data_Referencia = "2020-03-11"
ORDER BY Ret_Preco_Fechamento DESC
''').show(10)
```

Preco_Abertura	Preco_Fechamento	Ret_Preco_Fechamento	Codigo_Acao	Empresa	Subsetor
6.0	6.7	0.2407407	BTTL3.SA	BATTISTELLA	Comércio
30.24	32.75	0.1909091	JOPA3.SA	JOSAPAR	Alimentos Process...
1.38	1.31	0.07377049	OIBR4.SA	OI	Telecomunicações
1.04	0.93	0.04494382	OIBR3.SA	OI	Telecomunicações
6.4	7.3	0.04285714	GPCP3.SA	GPC PART	Químicos
2.9	2.88	0.02857143	JBDU3.SA	J B DUARTE	Outros
13.64	14.37	0.02642857	BMEB3.SA	MERC BRASIL	Intermediários Fi...
3.6	3.6	0.01983003	NORD3.SA	NORDON MET	Máquinas e Equipa...
12.58	12.8	0.01748814	BIOM3.SA	BIOMM	Medicamentos e Ou...
29.38	28.89	0.01340209	BPAC3.SA	BTGP BANCO	Intermediários Fi...

only showing top 10 rows

- Consulta que exibe as 10 ações com melhor fechamento no dia 12/03/2020 (um dia depois de ser decretado Pandemia Mundial por Corona Vírus)

```
spark.sql(''' SELECT Preco_Abertura, Preco_Fechamento, Ret_Preco_Fechamento,
Codigo_Acao, Empresa, Subsetor
FROM tabela_acoes
WHERE DataReferencia = "2020-03-12"
ORDER BY Ret_Preco_Fechamento DESC
''').show(10)
```

Preco_Abertura	Preco_Fechamento	Ret_Preco_Fechamento	Codigo_Acao	Empresa	Subsetor
1.97	2.23	0.1238739	AZEV3.SA	AZEVEDO	Construção e Enge...
50.0	50.0	0.1111111	MTIG4.SA	METAL IGUACU	Embalagens
10.0	11.49	0.09428571	MERC4.SA	MERC FINANC	Intermediários Fi...
11.2	11.77	0.05089286	ENGI3.SA	ENERGISA	Energia Elétrica
14.5	14.26	0.04852941	EUCA3.SA	EUCATEX	Madeira e Papel
2.51	2.79	0.02573529	RNEW4.SA	RENOVA	Energia Elétrica
4.57	4.31	0.01891253	RNEW3.SA	RENOVA	Energia Elétrica
4.85	5.06	0.007968127	EALT4.SA	ACO ALTONA	Máquinas e Equipa...
5.0	5.0	0.0	BAUH3.SA	EXCELSIOR	Alimentos Process...
29.86	29.86	0.0	AHEB3.SA	SPTURIS	Viagens e Lazer

only showing top 10 rows

- Consulta que usa método Like para selecionar todas as movimentações das ações ocorridas no mês de março de 2020

```
spark.sql('''
SELECT DataReferencia, Codigo_Acao, Setor, Preco_Abertura, Preco_Fechamento
FROM tabela_acoes
WHERE DataReferencia LIKE "2020-03%"
''').show(22)
```

DataReferencia	Codigo_Acao	Setor	Preco_Abertura	Preco_Fechamento
2020-03-02	AALR3.SA	Saúde	19.0	20.08
2020-03-03	AALR3.SA	Saúde	20.69	20.16
2020-03-04	AALR3.SA	Saúde	20.75	20.0
2020-03-05	AALR3.SA	Saúde	20.21	18.83
2020-03-06	AALR3.SA	Saúde	18.5	17.52
2020-03-09	AALR3.SA	Saúde	15.12	15.12
2020-03-10	AALR3.SA	Saúde	16.0	15.94
2020-03-11	AALR3.SA	Saúde	15.75	13.91
2020-03-12	AALR3.SA	Saúde	12.0	12.91
2020-03-13	AALR3.SA	Saúde	14.0	13.73
2020-03-16	AALR3.SA	Saúde	12.23	11.75
2020-03-17	AALR3.SA	Saúde	12.01	12.25
2020-03-18	AALR3.SA	Saúde	11.48	9.99
2020-03-19	AALR3.SA	Saúde	9.6	10.57
2020-03-20	AALR3.SA	Saúde	10.81	10.76
2020-03-23	AALR3.SA	Saúde	10.41	10.1
2020-03-24	AALR3.SA	Saúde	11.0	9.6
2020-03-25	AALR3.SA	Saúde	9.6	9.75
2020-03-26	AALR3.SA	Saúde	9.8	10.65
2020-03-27	AALR3.SA	Saúde	10.0	9.69
2020-03-30	AALR3.SA	Saúde	9.9	9.29
2020-03-31	AALR3.SA	Saúde	9.59	9.0

only showing top 22 rows

ETL com Apache Beam

- Instalando apache-beam e importando a biblioteca

```
pip install apache-beam[interactive]
```

```
import apache_beam as beam
```

- Pipe apenas para leitura do arquivo

```
p1 = beam.Pipeline ()

acoes = (
    p1
    | 'Extrair os dados' >> beam.io.ReadFromText('/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pyspark.csv',
                                                skip_header_lines= 0)
    | 'Separador' >> beam.Map(lambda record: record.split(','))
    #| 'Saída de dados' >> beam.Map(print)
    #| 'Gravar resultados' >> beam.io.WriterToText('resultado_acoes.txt')
)
p1.run()
```

```
WARNING:root:Make sure that locally built Python SDK docker image has Python 3.7 interpreter.
<apache_beam.runners.portability.fn_api_runner.fn_runner.RunnerResult at 0x7ff2060a7250>
```

- Pipe que recebe o Codigo_Acao e Data por input do usuario e exibe na tela apenas os registros que obedecem ao filtro.

```
cod_acao = input('Digite Código da Ação: ').upper().strip()
data = input('Digite a data de interesse no formato aaaa-mm-dd: ')

p2 = beam.Pipeline()

filtrar_acoes = (
    p2
    | 'Extrair os dados' >> beam.io.ReadFromText('/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pyspark.csv',
                                                skip_header_lines= 0)
    | 'Separador' >> beam.Map(lambda record: record.split(','))
    | 'Data de interesse' >> beam.Filter(lambda record : (record[6]) == data)
    | 'Filtrar por Codigo_Acao' >> beam.Filter(lambda record : record[7] == cod_acao)
    | 'Saída de Dados' >> beam.Map(print)
    #| 'Gravar resultado' >> beam.io.WriteToText('resultado_acoes.txt')
)
p2.run()
```

```

Digite Código da Ação: itub4.sa
Digite a data de interesse no formato aaaa-mm-dd: 2019-03-11
WARNING:root:Make sure that locally built Python SDK docker image has Python 3.7 interpreter.
['35.85', '37.02', '35.83', '36.92', '19995500', '34.86', '2019-03-11', 'ITUB4.SA', '0.03272715', '0.03272722', 'ITAUUNIBANCO',
<apache_beam.runners.portability.fn_api_runner.fn_runner.RunnerResult at 0x7ff20623fe90>

```

(neste exemplo o usuário digitou o Código_Ação 'ITUB4.SA' e data 11 de março de 2019)

- Pipe que filtra os registros apenas com data de 01/03/2019 e exibe na tela apenas os registros das colunas 'Data_Referencia', 'Preco_Fechamento', 'Ret_Preco_Fechamento', 'Subsetor'.

```

p3 = beam.Pipeline()

acoes_subsetor = (
    p3
    | 'Extrair os dados' >> beam.io.ReadFromText('/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pyspark.csv',
                                              skip_header_lines= 0)
    | 'Separador' >> beam.Map(lambda record: record.split(','))
    | 'Data de interesse' >> beam.Filter(lambda record : (record[6]) == '2019-03-01') #select na coluna Data_Referencia = 01/03/2019
    | 'Agregação de colunas' >> beam.Map(lambda record: (record[7],float(record[0]),float(record[9]),(record[12]))) #select na coluna Ret_Preco_Fechamento
    #| 'Combinar os dados' >> beam.CombinePerKey(sum) #tipo sum Ret_Preco_Fechamento + groupBy Subsetor
    | 'Saída de Dados' >> beam.Map(print)
    #| 'Gravar resultado' >> beam.io.WriteToText('resultado_acoes.txt')
)

p3.run()

```

```

WARNING:root:Make sure that locally built Python SDK docker image has Python 3.7 interpreter.
('AALR3.SA', 15.77, -0.0012658228, 'Análises e Diagnósticos')
('ABCB4.SA', 18.3, -0.0005471006, 'Intermediários Financeiros')
('ABEV3.SA', 17.14, -0.027278004, 'Bebidas')
('AFLT3.SA', 5.0, 0.030927835, 'Utilidade Pública')
('AGRO3.SA', 15.87, 0.008832808, 'Agropecuária')
('AHEB3.SA', 59.7, 0.0, 'Viagens e Lazer')
('ALPA3.SA', 15.94, 0.00511509, '"Tecidos')
('ALPA4.SA', 15.36, 0.03962461, '"Tecidos')
('ALSO3.SA', 30.68, -0.002913564, 'Exploração de Imóveis')
('ALUP3.SA', 8.0, 0.06666667, 'Energia Elétrica')
('ALUP4.SA', 7.34, -0.02536716, 'Energia Elétrica')
('ALUP11.SA', 22.95, -0.02183406, 'Energia Elétrica')
('AMAR3.SA', 6.19, 0.03896104, 'Comércio')
('ANIM3.SA', 6.26, -0.02457276, 'Diversos')
('APER3.SA', 23.46, -0.0102041, 'Previdência e Seguros')
('ARZZ3.SA', 53.97, -0.01982581, 'Comércio')
('ATOM3.SA', 2.34, -0.02542373, 'Outros')
('AZEVE3.SA', 8.95, 0.0, 'Construção e Engenharia')
('AZEVE4.SA', 4.28, -0.009090918, 'Construção e Engenharia')
('AZUL4.SA', 37.52, -0.02390443, 'Transporte')
('B3SA3.SA', 32.66, -0.02775242, 'Serviços Financeiros Diversos')
('BAHI3.SA', 118.29, 0.0, 'Diversos')
('BALM3.SA', 11.0, 0.0, 'Equipamentos')
('BALM4.SA', 11.06, -0.1485758, 'Equipamentos')
('BAUH3.SA', 5.0, 0.0, 'Alimentos Processados')

```

- Pipe que filtra apenas as ações que tiveram movimento negativo na data 01/03/2019 e agrupa por Subsetor

```
p4 = beam.Pipeline()

acoes_negat_subsetor = (
    p4
    | 'Extrair os dados' >> beam.io.ReadFromText('/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pyspark.csv',
                                                skip_header_lines= 0)
    | 'Separador' >> beam.Map(lambda record: record.split(','))
    | 'Data de interesse' >> beam.Filter(lambda record : (record[6]) == '2019-03-01') #select na coluna Data_Refere
    | 'Acoes com desempenho negativo' >> beam.Filter(lambda record: float(record[9]) < 0)
    | 'Agregação de colunas' >> beam.Map(lambda record: (record[12],float(record[9]))) #select Ret_Preco_Fechamento
    | 'Combinar os dados' >> beam.CombinePerKey(sum) #tipo sum Ret_Preco_Fechamento + groupBy Subsetor
    | 'Saída de Dados' >> beam.Map(print)
    | 'Gravar resultado' >> beam.io.WriteToText('resultado_acoes.txt')
)
p4.run()
```

```
WARNING:root:Make sure that locally built Python SDK docker image has Python 3.7 interpreter.
('Análises e Diagnósticos', -0.0620464168)
('Intermediários Financeiros', -0.3253004615999999)
('Bebidas', -0.027278004)
('Exploração de Imóveis', -0.116406625)
('Energia Elétrica', -0.5113239445000001)
('Diversos', -0.09646934700000001)
('Previdência e Seguros', -0.040353243)
('Comércio', -0.09856730400000001)
('Outros', -0.02542373)
('Construção e Engenharia', -0.060277218)
('Transporte', -0.12467983)
('Serviços Financeiros Diversos', -0.048343102)
('Equipamentos', -0.1485758)
('Máquinas e Equipamentos', -0.17910516399999998)
('Medicamentos e Outros Produtos', -0.01049463)
('Mineração', -0.020770725)
(' Gás e Biocombustíveis"', -0.18123810099999998)
('Alimentos Processados', -0.041555464)
('Tecidos', -0.17253884)
('Comércio e Distribuição', -0.06462290200000001)
('Água e Saneamento', -0.024825905000000002)
('Viagens e Lazer', -0.008561644)
('Construção Civil', -0.151999345)
('Madeira e Papel', -0.096353444)
('Material de Transporte', -0.043950064999999996)
('Siderurgia e Metalurgia', -0.102746263)
('Químicos', -0.011748848)
('Hoteis e Restaurantes', -0.059336730000000004)
('Telecomunicações', -0.0773748)
```


- Pipe que contabiliza e exibe na tela quantas ações distintas por Subsetor tiveram movimento negativo da data 01/03/2019

```
p5 = beam.Pipeline()

ocorrencias_negat_subsetor = (
    p5
    | 'Extrair os dados' >> beam.io.ReadFromText('/content/drive/MyDrive/dados/PROJETO_FINAL/acoes_pyspark.csv',
                                              skip_header_lines= 0)
    | 'Separador' >> beam.Map(lambda record: record.split(','))
    | 'Data de interesse' >> beam.Filter(lambda record : (record[6]) == '2019-03-01') #select na coluna Data_Refer
    | 'Acoes com desepenho negativo' >> beam.Filter(lambda record: float(record[9]) < 0)
    | 'Agregação de colunas' >> beam.Map(lambda record: (record[12],float(record[9]))) #select Ret_Preco_Fechament
    #| 'Combinar os dados' >> beam.CombinePerKey(sum) #tipo sum Ret_Preco_Fechamento + groupBy Subsetor
    | 'Ocorrencias de valor negativo por Subsetor' >> beam.combiners.Count.PerKey()
    | 'Saída de Dados' >> beam.Map(print)
    #| 'Gravar resultado' >> beam.io.WriteToText('resultado_acoes.txt')
)
p5.run()
```

```
WARNING:root:Make sure that locally built Python SDK docker image has Python 3.7 interpreter.
('Análises e Diagnósticos', 5)
('Intermediários Financeiros', 14)
('Bebidas', 1)
('Exploração de Imóveis', 8)
('Energia Elétrica', 21)
('Diversos', 6)
('Previdência e Seguros', 4)
('Comércio', 7)
('Outros', 1)
('Construção e Engenharia', 4)
('Transporte', 9)
('Serviços Financeiros Diversos', 3)
('Equipamentos', 1)
('Máquinas e Equipamentos', 8)
('Medicamentos e Outros Produtos', 1)
('Mineração', 3)
(' Gás e Biocombustíveis"', 8)
('Alimentos Processados', 5)
('Tecidos', 6)
('Comércio e Distribuição', 4)
('Água e Saneamento', 3)
('Viagens e Lazer', 1)
('Construção Civil', 8)
('Madeira e Papel', 6)
('Material de Transporte', 3)
('Siderurgia e Metalurgia', 7)
('Químicos', 2)
('Hoteis e Restaurantes', 2)
('Telecomunicações', 4)
```

DataSet 'ETHEREUM' e 'BITCOIN'

Conexão Google Cloud

Upload do DataSet bruto para a Cloud Storage

Autenticando o acesso ao Projeto e ao Bucket.

```
from google.colab import auth

project_id = "projeto-grupo2-economia"
bucket_name = "economia-dados-g2"

auth.authenticate_user()

!gcloud config set project {project_id}
```

Backup do DataSet 'ethereum.json' antes dos processos de limpeza direto para a pasta 'entrada' do Bucket.

```
#Enviando o DATASET bruto para o bucket de entrada
!gsutil cp /content/drive/MyDrive/ProjetoFinal/crypto/ethereum.json gs://{bucket_name}/entrada/Ethereum_Bruto.json
```

```
#Enviando os dados brutos para o Bucket
!gsutil cp /content/bitcoin.json gs://{bucket_name}/entrada/
```

Download do Dataset da bucket para o drive para poder usá-lo no Colab

```
!gsutil cp gs://{bucket_name}/entrada/Ethereum_Bruto.json /content/drive/MyDrive/ProjetoFinal/crypto/ethereum_entrada.json
```

Download do Dataset da bucket para o ambiente de trabalho local do Colab

```
#Download dos dados para área de trabalho local
!gsutil cp gs://{bucket_name}/entrada/bitcoin.json /content/
```

ETL com PYTHON e PANDAS

Instalação da biblioteca Pandas com o apelido 'pd'

```
import pandas as pd
```

Fazendo a ingestão de dados no colab com o arquivo 'json'

```
df = pd.read_json('/content/drive/MyDrive/ProjetoFinal/crypto/ethereum_entrada.json')
```

```
#Abrindo o arquivo json  
df = pd.read_json("/content/bitcoin.json")
```

Realizando a tradução das colunas com o comando 'rename(columns)' e utilizando o parâmetro de 'inplace=True' para sobrescrever no próprio DataFrame.

```
df.rename(columns={  
    'SNo':'Id',  
    'Symbol':'Sigla',  
    'Date':'Data',  
    'Name':'Moeda',  
    'Open':'Preco_Abertura',  
    'Close':'Preco_Fechamento',  
    'Low':'Menor_Preco',  
    'High':'Maior_Preco',  
    'Volume':'Volume',  
    'Marketcap':'Capitalizacao_Mercado'  
}, inplace=True)
```

Comentário 01: o parâmetro inplace ajuda a decidir como você deseja afetar os dados na qual você está trabalhando. Se desejar fazer uma alteração no objeto dataframe e substituir o que estava lá antes, usa-se juntamente com o 'True'. Caso você deseje fazer uma cópia do dataframe e atribuí-lo a uma variável diferente para que possa modificar os dados originais usa-se junto ao 'False'.

Verificando os tipos de dados que existem dentro de cada coluna do DataFrame.

```
df.dtypes
Id                int64
Moeda             object
Sigla             object
Data             datetime64[ns]
Maior_Preco      float64
Menor_Preco      float64
Preco_Abertura   float64
Preco_Fechamento float64
Volume           float64
Capitalizacao_Mercado float64
dtype: object
```

Formatando a notação científica do DataFrame para float.

```
pd.options.display.float_format = '{:,.2f}'.format
```

Volume	Marketcap
6.741880e+05	4.548689e+07
5.321700e+05	4.239957e+07
4.052830e+05	4.281836e+07

Antes

Volume	Capitalizacao_Mercado
674,188.00	45,486,894.24
532,170.00	42,399,573.50
405,283.00	42,818,364.39
1,463,100.00	64,569,288.43

Depois

Comentário 02: O pandas tem um sistema de 'options' que permite personalizar alguns aspectos de seu comportamento, sendo as 'options' relacionadas à exibição aquelas que o usuário provavelmente ajustará. A formatação da notação científica dos tipos de dados float resulta em um float com uma quantidade específica de dígitos após a vírgula decimal no Pandas.

Usando 'pandas.to_datetime' formatação da coluna 'Data' excluindo os caracteres correspondentes a hora, deixando apenas 'ano-mês-dia' para compor os registros de data.

```
df['Data'] = pd.to_datetime(df['Data']).dt.normalize()
```

Date	Data
2018-04-29 23:59:59	2018-04-29
2018-04-30 23:59:59	2018-04-30
2018-05-01 23:59:59	2018-05-01
2018-05-02 23:59:59	2018-05-02
2018-05-03 23:59:59	2018-05-03

Antes

Depois

Transformando os dados que antes estavam no formato 'json' para 'csv' e fazendo upload no Drive.

```
df.to_csv("/content/drive/MyDrive/ProjetoFinal/crypto/ethereum_pandas.csv", index=True)
```

```
df.to_csv("/content/bitcoin_pandas.csv", index=False)
```

Comentário 03: No comando acima existe um método 'index=True', ele é utilizado quando o DataFrame possui uma coluna que há uma sequência imutável usada para indexação e alinhamento. O objeto básico que armazena rótulos de eixo para todos os objetos pandas.

Fazendo o upload do DataFrame normalizado para a bucket da google cloud platform

```
!gsutil cp /content/drive/MyDrive/ProjetoFinal/crypto/ethereum_pandas.csv gs://{bucket_name}/saida/dados_pandas/
```

```
!gsutil cp /content/bitcoin_pandas.csv gs://{bucket_name}/saida/dados_pandas/
```

Plotagem com PANDAS

A plotagem é utilizada para visualizar de forma rápida alguns resultados em gráficos.

```
colunas_selecionadas_plotagem = ['Data', 'Preco_Fechamento']

df1=df.filter(items=colunas_selecionadas_plotagem)

df1 = df1.query("Data >= '2018-03-01' & Data <= '2018-03-31'")
```

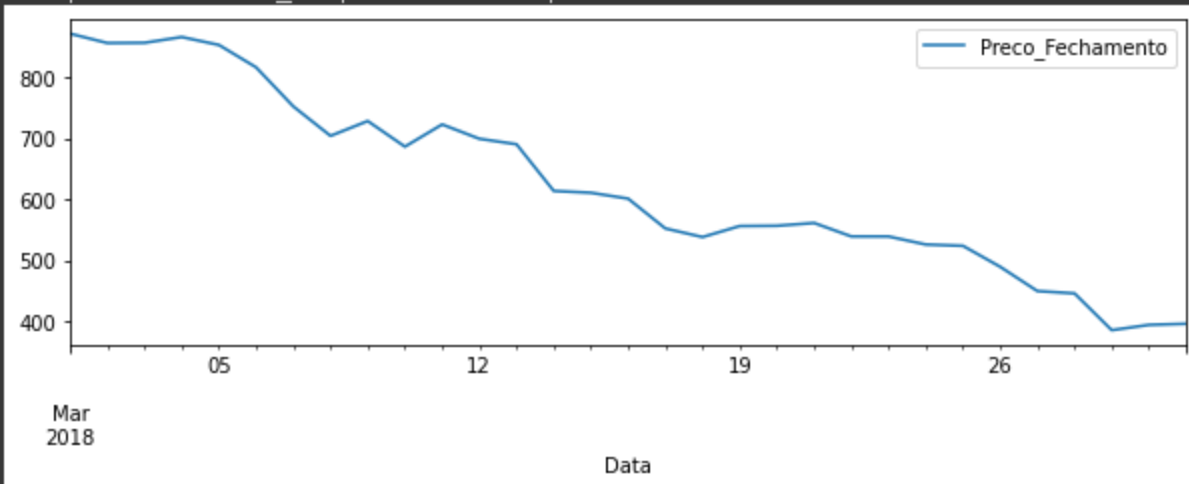
Comentário 04: Na primeira linha de código foi criado uma lista contendo duas colunas do DataFrame ethereum intitulado como 'df'. Já na segunda linha de código está sendo criado um novo DataFrame chamado 'df1' onde ele recebe o DataFrame original só que filtrando apenas as duas colunas que estão contidas na lista 'colunas_selecionadas_plotagem'. Na terceira linha de código é utilizado o método de consulta 'query' onde está sendo buscando os resultados no intervalo de data do dia 01/03/2018 ao dia 31/03/2018.

df1		
	Data	Preco_Fechamento
Id		
937	2018-03-01	872.20
938	2018-03-02	856.85
939	2018-03-03	857.22
940	2018-03-04	866.68
941	2018-03-05	853.68
942	2018-03-06	816.95
943	2018-03-07	752.83
944	2018-03-08	704.60
945	2018-03-09	728.92

Aqui está sendo feita a plotagem com o df1 criado anteriormente contendo no eixo X a coluna 'Data' e no eixo Y a coluna 'Preco_Fechamento'. o "kind='line'" diz respeito ao tipo de gráfico que será mostrado, no caso abaixo foi escolhido o de linhas, já o figsize trata da dimensão da imagem.

```
df1.plot(x='Data',y='Preco_Fechamento',kind='line',figsize=(10, 3))
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f329c098b10>



Instalações e Configurações do Pyspark

Importação de Bibliotecas com suas funções para fazer uso nas consultas.

```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, DateType, FloatType, DoubleType
from pyspark.sql.window import Window
from pyspark.sql.functions import *
```

Configuração da SparkSession

```
spark = (SparkSession.builder\
        .master("local")\
        .appName("dataframe.ethereum")\
        .config("spark.ui.port", "4050")\
        .getOrCreate())
```

Comentário 05: A primeira coisa a se fazer ao querer usar qualquer funcionalidade do Spark é configurar a classe `SparkSession`. Nessa configuração existem alguns parâmetros como o `builder` que é um atributo de classe que literalmente constrói nossas instâncias `SparkSession`. O `master` define uma URL mestre para rodar a instância. O `appName` define um nome para o aplicativo que será mostrado na IU da web do spark. O `config` define as opções de configuração, no caso do código acima sinaliza a porta que será utilizada. O `getOrCreate` obtém um existente `SparkSession` ou, se não houver, cria um novo com base nas opções definidas neste builder.

Montando toda a estrutura do DataFrame com as classes `StructType` e `StructField`

```
schema= StructType([
    StructField('Id',IntegerType(),True),
    StructField('Moeda',StringType(),True),
    StructField('Sigla',StringType(),True),
    StructField('Data',DateType(),True),
    StructField('Maior_Preco',FloatType(),True),
    StructField('Menor_Preco',FloatType(),True),
    StructField('Preco_Abertura',FloatType(),True),
    StructField('Preco_Fechamento',FloatType(),True),
    StructField('Volume',FloatType(),True),
    StructField('Capitalizacao_Mercado',FloatType(),True)])
```

Comentário 06: As classes PySpark `StructType` e `StructField` são usadas para especificar o esquema para o `DataFrame` e criar colunas complexas. `StructType` é uma coleção de `StructField` que define o nome da coluna, tipo de dados da coluna, booleano para especificar se o campo pode ser anulável ou não, etc.

Verificando as colunas com o `printSchema`, método no DataFrame que mostra colunas StructType como "struct".

```
df_ethereum.printSchema()
```

```
root
|-- Id: integer (nullable = true)
|-- Moeda: string (nullable = true)
|-- Sigla: string (nullable = true)
|-- Data: date (nullable = true)
|-- Maior_Preco: float (nullable = true)
|-- Menor_Preco: float (nullable = true)
|-- Preco_Abertura: float (nullable = true)
|-- Preco_Fechamento: float (nullable = true)
|-- Volume: float (nullable = true)
|-- Capitalizacao_Mercado: float (nullable = true)
```

Consultas no Pyspark

Criando uma coluna nova com a função `withColumn` verificando as vezes na qual a moeda ethereum fechou o dia com valorização utilizando as verificações `When` e `Otherwise`, ambas verificações funcionam como o 'if' e 'else' da programação python onde retorna um valor quando a primeira condição é verdadeira e quando é falsa retorna o valor da segunda condição.

```
df_ethereum1 = (df_ethereum.withColumn("Valorizou_No_Dia",
    F.when(F.col("Preco_Fechamento") > F.col("Preco_Abertura"), F.lit("SIM"))
    .otherwise("NAO")))
df_ethereum1.show()
```

	Id	Moeda	Sigla	Data	Maior_Preco	Menor_Preco	Preco_Abertura	Preco_Fechamento	Volume	Capitalizacao_Mercado	Valorizou_No_Dia
1	Ethereum	ETH	2015-08-08	2.7988100051879883	0.7147250175476074	2.793760061264038	0.7533249855041504	674188.0	4.54868942408E7	NAO	
2	Ethereum	ETH	2015-08-09	0.8798899756240845	0.629190981388092	0.7061359882354736	0.7018970251083374	532170.0	4.23995734991E7	NAO	
3	Ethereum	ETH	2015-08-10	0.7298539876937866	0.6365460157394409	0.7139890193039209	0.7084479928016663	405283.0	4.28183643045E7	NAO	
4	Ethereum	ETH	2015-08-11	1.131410002708435	0.6632350007165833	0.7080870270720065	1.0678600072860718	1463100.0	6.45692884328E7	SIM	
5	Ethereum	ETH	2015-08-12	1.2899399995803833	0.8836079835891724	1.058750033378601	1.2174400091171265	2150620.0	7.36450109063E7	SIM	
6	Ethereum	ETH	2015-08-13	1.9650700092315674	1.1719900369644165	1.2222399711608887	1.8276699781417847	4068680.0	1.10607191674E8	SIM	
7	Ethereum	ETH	2015-08-14	2.2618799209594727	1.7547500133514404	1.810920000076294	1.8278700113296509	4637030.0	1.10672321811E8	SIM	
8	Ethereum	ETH	2015-08-15	1.8772399425506592	1.5709799528121948	1.8028899431228638	1.6888999938964844	2554360.0	1.02303608467E8	NAO	
9	Ethereum	ETH	2015-08-16	1.6952400207519531	1.0898100137710571	1.6843500137329102	1.5660300254821777	3550790.0	9.49010053503E7	NAO	
10	Ethereum	ETH	2015-08-17	1.5811899900436401	1.1853400468826294	1.5811899900436401	1.2036099433898926	1942830.0	8.72953665007E7	NAO	
11	Ethereum	ETH	2015-08-18	1.3311599493026733	1.087049961090088	1.2152999639511108	1.087049961090088	1485680.0	7.8868413078E7	NAO	
12	Ethereum	ETH	2015-08-19	1.3179899454116821	1.1669299602508545	1.1669299602508545	1.2588599920272827	1486240.0	9.13663914393E7	SIM	

Utilizando a função `select` para selecionar as colunas **Data**, **Sigla**, **Preço Abertura** e **Preço Fechamento** onde na coluna **Valorizou No Dia** corresponder a SIM.

```
df_ethereum1.select(F.col('Data'),F.col('Sigla'),F.col('Preco_Abertura'),F.col('Preco_Fechamento')).filter(F.col('Valorizou_No_Dia')== 'SIM').show(10)
```

Data	Sigla	Preco_Abertura	Preco_Fechamento
2015-08-11	ETH	0.788870270729065	1.0678600072860718
2015-08-12	ETH	1.058750033378601	1.2174400091171265
2015-08-13	ETH	1.2222399711608887	1.8276699781417847
2015-08-14	ETH	1.810920000076294	1.8278700113296509
2015-08-19	ETH	1.1669299602508545	1.2588599920272827
2015-08-20	ETH	1.2511800527572632	1.4649200439453125
2015-08-26	ETH	1.132789969444275	1.159980058670044
2015-08-28	ETH	1.1476600170135498	1.1913800239562988
2015-08-30	ETH	1.1829899549484253	1.3192700147628784
2015-08-31	ETH	1.3213800191879272	1.358240008354187

only showing top 10 rows

Criação de Colunas no Pyspark

No Dataset de Bitcoin foram criadas duas colunas para além das originais. A primeira chamada "Valorizou_No_Dia" para facilitar a verificação de momentos em que a moeda esteve valorizada.

```
df = df.withColumn("Valorizou_No_Dia", F.when(F.col("Preco_Fechamento") > F.col("Preco_Abertura"), \
F.lit("SIM")).otherwise("NAO"))
```

Resultando na seguinte coluna.

Valorizou_No_Dia
SIM
NAO
NAO
NAO
NAO
SIM
SIM
NAO
NAO
SIM

A segunda coluna criada foi a "Variacao_Preco", que criada a partir de uma subtração simples entre o preço de fechamento e o preço de abertura de cada dia, de forma que é possível entender a variação ao longo do tempo.

```
df = df.withColumn("Variacao_Preco", F.col("Preco_Fechamento")-F.col("Preco_Abertura"))
```

	Variacao_Preco
10.095993041992188	
-5.0	
-22.01000213623047	
-11.169998168945312	
-8.5	

Spark SQL

Fazendo a ingestão dos dados já normalizados pelo pandas e pelo pyspark da bucket para o drive.

```
!gsutil cp gs://{bucket_name}/saida/dados_pyspark/ethereum_pyspark.csv /content/drive/MyDrive/ProjetoFinal/crypto/ethereum_spark_sql.csv
```

Configurando o leitor de arquivo csv do spark SQL e fazendo o load do arquivo do drive, por último apelidando o 'df_ethereum_sql' como tabela_ethereum, e o arquivo 'bitcoin_spark_sql.csv' como tabela_bitcoin.

```
df_ethereum_sql = (spark
    .read
    .format("csv")
    .option("header", "true")
    .option("inferSchema", "true")
    .option("delimiter", ",")
    .load("/content/drive/MyDrive/ProjetoFinal/crypto/ethereum_spark_sql.csv")
    .createOrReplaceTempView("tabela_ethereum"))
```

```
df_sql = (spark
    .read
    .format("csv")
    .option("header", "true")
    .option("inferSchema", "true")
    .option("delimiter", ",")
    .load("/content/dados_sparksql/bitcoin_spark_sql.csv")
    .createOrReplaceTempView("tabela_bitcoin"))
```

Realizando um 'Select' em todo o Data Frame

spark.sql(''''SELECT * FROM tabela_ethereum''').show()										
	Id	Moeda	Sigla	Data	Maior_Preco	Menor_Preco	Preco_Abertura	Preco_Fechamento	Volume	Capitalizacao_Mercado
1	Ethereum	ETH	2015-08-08	2.7988100051879883	0.7147250175476074	2.793760061264038	0.7533249855041504	674188.0	4.54868942408E7	
2	Ethereum	ETH	2015-08-09	0.8798099756240845	0.629190981388092	0.7061359882354736	0.7018970251083374	532170.0	4.23995734991E7	
3	Ethereum	ETH	2015-08-10	0.7298539876937866	0.6365460157394409	0.7139890193939209	0.7084479928016663	405283.0	4.28183643945E7	
4	Ethereum	ETH	2015-08-11	1.131410002708435	0.6632350087165833	0.7080870270729065	1.0678600072860718	1463100.0	6.45692884328E7	
5	Ethereum	ETH	2015-08-12	1.289939995803833	0.8836079835891724	1.058750033378601	1.2174400091171265	2150620.0	7.36450109863E7	
6	Ethereum	ETH	2015-08-13	1.9650700092315674	1.1719900369644165	1.2222399711608887	1.8276699781417847	4068680.0	1.10607191674E8	
7	Ethereum	ETH	2015-08-14	2.2618799209594727	1.7547500133514404	1.810920000076294	1.8278700113296509	4637030.0	1.10672321811E8	
8	Ethereum	ETH	2015-08-15	1.8772399425506592	1.5709799528121948	1.8028899431228638	1.6888999938964844	2554360.0	1.02303608467E8	
9	Ethereum	ETH	2015-08-16	1.6952400207519531	1.0898100137710571	1.6843500137329102	1.5660300254821777	3550790.0	9.49010053503E7	
10	Ethereum	ETH	2015-08-17	1.5811899900436401	1.1853400468826294	1.5811899900436401	1.2036099433898926	1942830.0	8.72953665007E7	
11	Ethereum	ETH	2015-08-18	1.3311599493026733	1.087049961090088	1.2152999639511108	1.087049961090088	1485680.0	7.8868413078E7	
12	Ethereum	ETH	2015-08-19	1.3179899454116821	1.1669299602508545	1.1669299602508545	1.2588599920272827	1486240.0	9.13663914393E7	
13	Ethereum	ETH	2015-08-20	1.5333000421524048	1.248329997062683	1.2511800527572632	1.4649200439453125	2843760.0	1.06351420178E8	
14	Ethereum	ETH	2015-08-21	1.5564199686050415	1.3528000116348267	1.4775199890136719	1.3952900171279907	2020970.0	1.01331855597E8	
15	Ethereum	ETH	2015-08-22	1.4764100313186646	1.352679967880249	1.3962899446487427	1.37923002243042	948310.0	1.00201825662E8	
16	Ethereum	ETH	2015-08-23	1.4097000360488892	1.2977700233459473	1.375	1.3525899648666382	1589300.0	9.83003513461E7	
17	Ethereum	ETH	2015-08-24	1.362779974937439	1.2312699556350708	1.3455899953842163	1.2312699556350708	924920.0	8.95152565525E7	
18	Ethereum	ETH	2015-08-25	1.241819977760315	1.1286499500274658	1.2286100387573242	1.1401900053024292	1307180.0	8.29226329052E7	
19	Ethereum	ETH	2015-08-26	1.2024799585342407	1.0618300437927246	1.132789969444275	1.159980058670044	1056750.0	8.43909235384E7	
20	Ethereum	ETH	2015-08-27	1.188830018043518	1.1372900009155273	1.1698100566864014	1.1476999521255493	686662.0	8.35245853858E7	

only showing top 20 rows

Selecionando a menor e a maior data contida no Data Frame utilizando as funções 'min' e 'max' na coluna 'Data'

```
spark.sql('SELECT MIN(Data) AS Data_Inicial,
MAX(Data) AS Data_Final
FROM tabela_ethereum
').show()
```

Data_Inicial	Data_Final
2015-08-08	2021-07-06

Novamente com a função 'min' e 'max' foi capturado o menor e maior preço de capitalização de mercado da ethereum entre as datas de 08/08/2015 a 06/07/2021

```
spark.sql('SELECT MIN(Capitalizacao_Mercado) AS Capitalizacao_Minima,
MAX(Capitalizacao_Mercado) AS Capitalizacao_Maxima
FROM tabela_ethereum
').show()
```

Capitalizacao_Minima	Capitalizacao_Maxima
3.22136261748E7	4.8288190049093E11

Selecionando as colunas Sigla, Data, Maior_Preco da tabela_ethereum usando a função ORDER BY na coluna Maior_Preco de forma descendente.

```
spark.sql(''' SELECT Sigla, Data, Maior_Preco
FROM tabela_ethereum
ORDER BY Maior_Preco DESC
''').show()
```

Sigla	Data	Maior_Preco
ETH	2021-05-12	4362.3505418
ETH	2021-05-10	4197.47327657
ETH	2021-05-11	4178.20881516
ETH	2021-05-14	4171.01696316
ETH	2021-05-15	4129.18552263
ETH	2021-05-13	4032.5635388
ETH	2021-05-09	3981.259091

Selecionando na tabela_bitcoin informações sobre os dias em que a moeda esteve valorizada.

```
spark.sql('''
SELECT Data, Maior_Preco, Menor_Preco, Preco_Abertura, Preco_Fechamento, Variacao_Preco
FROM tabela_bitcoin
WHERE Valorizou_No_Dia == "SIM"
''').show()
```

Na tabela_bitcoin, foram selecionadas informações sobre a variação de preços da moeda, ordenando das maiores variações positivas para as maiores variações negativas

```
spark.sql('''
SELECT Variacao_Preco, Preco_Abertura, Preco_Fechamento, Data
FROM tabela_bitcoin
ORDER BY Variacao_Preco DESC
''').show()
```

Variacao_Precos	Precos_Abertura	Precos_Fechamento	Data
7309.636429489998	38886.82728995	46196.46371944	2021-02-08
4943.96242435	49077.79236302	54021.75478737	2021-04-26
4471.738318240001	45159.50305253	49631.24137077	2021-03-01
4212.152397079997	51675.98128513	55888.13368221	2021-02-19
4181.5137619399975	53568.66358369	57750.17734563	2021-04-30
4171.841186879996	53252.16476125	57424.00594813	2021-05-05
4029.0685544299995	36753.66970802	40782.73826245	2021-05-20
4005.61506923	34700.363568	38705.97863723	2021-05-24
3928.14363531	33416.97785088	37345.12148619	2021-06-09
3899.71451813	57343.37024739	61243.08476552	2021-03-13
3633.599609375	14266.099609375	17899.69921875	2017-12-07
3613.440139860002	59890.01779033	63503.45793019	2021-04-13
3542.0707562499956	35555.79014042	39097.86089667	2021-06-13
3454.303636079996	51683.0100897	55137.31372578	2021-03-26
3401.239818730006	33915.11958124	37316.35939997	2021-01-13
3025.0545414100015	30441.0418168	33466.09635821	2021-01-28
3010.619583340005	44898.71161149	47909.33119483	2021-02-11
2941.731103539998	49207.27643233	52149.00753587	2021-02-17
2810.7498770899983	34013.614533	36824.36441009	2021-01-06
2750.81210508	29376.45583414	32127.26793922	2021-01-02

Ainda na tabela_bitcoin foram selecionadas informações sobre a moeda em março de 2020.

```
spark.sql(''
SELECT *
FROM tabela_bitcoin
WHERE Data LIKE "2020-03%"
'').show()
```

Data	Maior_Precos	Menor_Precos	Precos_Abertura	Precos_Fechamento	Volume	Capitalizacao_Mercado
2020-01-01	7254.33061134	7174.94415256	7194.89197053	7200.17439274	1.85656649967884E10	1.30580829149587E11
2020-01-02	7212.15525252	6935.26997193	7202.55112207	6985.47000061	2.08020834653292E10	1.26699395235204E11
2020-01-03	7413.71509934	6914.99590793	6984.4286123	7344.88418341	2.81114810319377E10	1.33233444755489E11
2020-01-04	7427.38579435	7309.5140117	7345.37527527	7410.65656642	1.84442712747607E10	1.34442464030264E11
2020-01-05	7544.49687224	7400.5355609	7410.45169373	7411.31732676	1.97250740945446E10	1.34469548249076E11

Concatenando os DataSets

‘ACOES’, ‘ETHEREUM’ e ‘BITCOIN’

ETL com PYTHON e PANDAS

- Ingestão dos Datasets, que foram tratados com Pyspark, direto da GCP via !gsutil

```
1 #Dataset Acoes
2
3 !gsutil cp gs://{bucket_name}/saida/dados_pyspark/acoes_pyspark.csv /content/drive/MyDrive/dados/JOIN/acoes_join.csv

Copying gs://economia-dados-g2/saida/dados_pyspark/acoes_pyspark.csv...
- [1 files][ 39.5 MiB/ 39.5 MiB]
Operation completed over 1 objects/39.5 MiB.

1 #Dataset Ethereum
2
3 !gsutil cp gs://{bucket_name}/saida/dados_pyspark/bitcoin_pyspark.csv /content/drive/MyDrive/dados/JOIN/bitcoin_join.csv

Copying gs://economia-dados-g2/saida/dados_pyspark/bitcoin_pyspark.csv...
/ [1 files][409.8 KiB/409.8 KiB]
Operation completed over 1 objects/409.8 KiB.

1 #Dataset Bitcoin
2
3 !gsutil cp gs://{bucket_name}/saida/dados_pyspark/ethereum_pyspark.csv /content/drive/MyDrive/dados/JOIN/ethereum_join.csv

Copying gs://economia-dados-g2/saida/dados_pyspark/ethereum_pyspark.csv...
/ [1 files][254.2 KiB/254.2 KiB]
Operation completed over 1 objects/254.2 KiB.
```

- Leitura dos Datasets:

‘Acoes’

```
df_acoes_join = pd.read_csv(r"/content/drive/MyDrive/dados/JOIN/acoes_join.csv",
                             sep=',',
                             parse_dates=[ 'Data Referencia'],
                             dayfirst=True)
```

- Drop das colunas excedentes

```
lista_drop_acoes=['Volume', 'Preco_Ajustado', 'Ret_Preco_Ajustado', 'Ret_Preco_Fechamento',
                  'Setor', 'Tipo','Classificacao']
df_acoes_join = df_acoes_join.drop(lista_drop_acoes,axis = 1)
df_acoes_join.rename(columns={"Data_Referencia":"Data"},inplace = True)
```

- Reorganizando as colunas do Dataset

```
cols = list(df_acoes_join.columns.values)
df_acoes_join = df_acoes_join[['Empresa','Codigo_Acao','Data','Maior_Preco','Menor_Preco',
                               'Preco_Abertura','Preco_Fechamento','Subsetor']]
df_acoes_join.head(3)
```

'Bitcoin'

```
df_bitcoin_join = pd.read_csv(r"/content/drive/MyDrive/dados/JOIN/bitcoin_join.csv",
                              sep=',',
                              parse_dates=['Data'],
                              dayfirst=True)
```

- Drop das colunas excedentes, renomeando as colunas 'Sigla' e 'Moeda' para 'Codigo_Acao' e 'Empresa' respectivamente, adicionando nova coluna 'Subsetor'

```
lista_drop_bitcoin=['Id', 'Volume', 'Capitalizacao_Mercado', 'Valorizou_No_Dia', 'Variacao_Preco']
df_bitcoin_join = df_bitcoin_join.drop(lista_drop_bitcoin,axis = 1)

df_bitcoin_join.rename(columns={"Sigla":"Codigo_Acao",
                                "Moeda":"Empresa",
                                },inplace = True)

df_bitcoin_join['Subsetor'] = 'Criptomoeda'
```


'Ethereum'

```
df_ethereum_join = pd.read_csv(r"/content/drive/MyDrive/dados/JOIN/ethereum_join.csv",  
                                sep=',',  
                                parse_dates=['Data'],  
                                dayfirst=True)
```

- Drop das colunas excedentes, renomeando as colunas 'Sigla' e 'Moeda' para 'Codigo_Acao' e 'Empresa' respectivamente, adicionando nova coluna 'Subsetor'

```
lista_drop_ethereum = ['Id', 'Volume', 'Capitalizacao_Mercado']  
df_ethereum_join = df_ethereum_join.drop(lista_drop_ethereum, axis = 1)  
  
df_ethereum_join.rename(columns={"Sigla": "Codigo_Acao",  
                                "Moeda": "Empresa",  
                                }, inplace = True)  
  
df_ethereum_join['Subsetor'] = 'Criptomoeda'
```

- Abrindo os Datasets:

'df_acoes_join'

```
1 df_acoes_join.head(5)
```

	Empresa	Codigo_Acao	Data	Maior_Preco	Menor_Preco	Preco_Abertura	Preco_Fechamento	Subsetor
0	ALLIAR	AALR3.SA	2018-01-02	15.16	14.70	14.94	14.89	Análises e Diagnósticos
1	ALLIAR	AALR3.SA	2018-01-03	15.32	14.79	14.89	14.96	Análises e Diagnósticos
2	ALLIAR	AALR3.SA	2018-01-04	15.16	14.81	15.00	15.09	Análises e Diagnósticos
3	ALLIAR	AALR3.SA	2018-01-05	15.17	14.95	15.05	15.05	Análises e Diagnósticos
4	ALLIAR	AALR3.SA	2018-01-08	15.10	14.62	15.05	14.79	Análises e Diagnósticos

'df_bitcoin_join'

```
2 df_bitcoin_join.head(5)
```

	Empresa	Codigo_Acao	Data	Maior_Preco	Menor_Preco	Preco_Abertura	Preco_Fechamento	Subsetor
0	Bitcoin	BTC	2013-04-29	147.488007	134.000000	134.444000	144.539993	Criptomoeda
1	Bitcoin	BTC	2013-04-30	146.929993	134.050003	144.000000	139.000000	Criptomoeda
2	Bitcoin	BTC	2013-05-01	139.889999	107.720001	139.000000	116.989998	Criptomoeda
3	Bitcoin	BTC	2013-05-02	125.599998	92.281898	116.379997	105.209999	Criptomoeda
4	Bitcoin	BTC	2013-05-03	108.127998	79.099998	106.250000	97.750000	Criptomoeda

'df_ethereum_join'

```
1 df_ethereum_join.head(5)
```

	Empresa	Codigo_Acao	Data	Maior_Preco	Menor_Preco	Preco_Abertura	Preco_Fechamento	Subsetor
0	Ethereum	ETH	2015-08-08	2.798810	0.714725	2.793760	0.753325	Criptomoeda
1	Ethereum	ETH	2015-08-09	0.879810	0.629191	0.706136	0.701897	Criptomoeda
2	Ethereum	ETH	2015-08-10	0.729854	0.636546	0.713989	0.708448	Criptomoeda
3	Ethereum	ETH	2015-08-11	1.131410	0.663235	0.708087	1.067860	Criptomoeda
4	Ethereum	ETH	2015-08-12	1.289940	0.883608	1.058750	1.217440	Criptomoeda

- Concatenando os Datasets 'df_acoes_join' + 'df_bitcoin_join' + 'df_ethereum_join'

```
frames = [df_acoes_join, df_bitcoin_join, df_ethereum_join]
df_economia = pd.concat(frames)
```

- Conferindo a estrutura do novo Dataset 'df_economia'

```
1 df_economia.dtypes
```

```
Empresa          object
Codigo_Acao      object
Data             datetime64[ns]
Maior_Preco      float64
Menor_Preco      float64
Preco_Abertura   float64
Preco_Fechamento float64
Subsetor         object
dtype: object
```

```
1 df_economia.count()
```

```
Empresa          304012
Codigo_Acao      304012
Data             304012
Maior_Preco      304012
Menor_Preco      304012
Preco_Abertura   304012
Preco_Fechamento 304012
Subsetor         304012
dtype: int64
```

- Validando a estrutura do novo Dataset com Pandera

```
schema_df = {
    'Empresa':pa.Column(pa.String),
    'Codigo_Acao':pa.Column(pa.String),
    'Data':pa.Column(pa.DateTime),
    'Maior_Preco':pa.Column(pa.Float),
    'Menor_Preco':pa.Column(pa.Float),
    'Preco_Abertura':pa.Column(pa.Float),
    'Preco_Fechamento':pa.Column(pa.Float),
    'Subsetor':pa.Column(pa.String)
}
```

```
schema = pa.DataFrameSchema(columns=schema_df)
schema.validate(df_economia)
```

- Visão geral do Dataset 'df_economia'

	Empresa	Codigo_Acao	Data	Maior_Preco	Menor_Preco	Preco_Abertura	Preco_Fechamento	Subsetor
0	ALLIAR	AALR3.SA	2018-01-02	15.160000	14.700000	14.940000	14.890000	Análises e Diagnósticos
1	ALLIAR	AALR3.SA	2018-01-03	15.320000	14.790000	14.890000	14.960000	Análises e Diagnósticos
2	ALLIAR	AALR3.SA	2018-01-04	15.160000	14.810000	15.000000	15.090000	Análises e Diagnósticos
3	ALLIAR	AALR3.SA	2018-01-05	15.170000	14.950000	15.050000	15.050000	Análises e Diagnósticos
4	ALLIAR	AALR3.SA	2018-01-08	15.100000	14.620000	15.050000	14.790000	Análises e Diagnósticos
...
2155	Ethereum	ETH	2021-07-02	2155.596496	2021.824808	2109.892677	2150.040364	Criptomoeda
2156	Ethereum	ETH	2021-07-03	2237.567155	2117.590013	2150.835025	2226.114282	Criptomoeda
2157	Ethereum	ETH	2021-07-04	2384.286857	2190.837703	2226.550382	2321.724112	Criptomoeda
2158	Ethereum	ETH	2021-07-05	2321.922836	2163.041394	2321.922836	2198.582464	Criptomoeda
2159	Ethereum	ETH	2021-07-06	2346.294874	2197.919385	2197.919385	2324.679449	Criptomoeda

304012 rows × 8 columns

- Conferindo se não há registros 'nulos'

```
1 df_economia.isna().sum()

Empresa      0
Codigo_Acao  0
Data         0
Maior_Preco  0
Menor_Preco  0
Preco_Abertura  0
Preco_Fechamento  0
Subsetor     0
dtype: int64
```

Load do Dataset concatenado 'df_economia' para GCP

```
df_economia.to_csv('/content/drive/MyDrive/dados/PROJETO_FINAL/df_final_economia.csv', index = False)
```

```
#load para a pasta SAIDA
```

```
!gsutil cp /content/drive/MyDrive/dados/PROJETO_FINAL/df_final_economia.csv  
gs://{bucket_name}/saida/dados_final/df_final_economia.csv
```

```
Copying file:///content/drive/MyDrive/dados/PROJETO_FINAL/df_final_economia.csv [Content-Type=text/csv]...  
-  
Operation completed over 1 objects/21.1 MiB.
```

```
#LOAD para a pasta SAIDA para ser ingerido na PIPELINE com DATAFLOW
```

```
!gsutil cp /content/drive/MyDrive/dados/PROJETO_FINAL/df_final_economia.csv  
gs://{bucket_name}/entrada/df_economia.csv
```

BIG QUERY

Uma vez que os dataframes já estavam normalizados e as alterações já estavam no Bucket apropriado e já havíamos tirado nossos insights dos dataframes usando as ferramentas já mencionadas, utilizamos o BigQuery para criar tabelas que nos permitissem demonstrar as formas com que a pandemia afetou os mercados de ações e criptomoedas.

1.

```
CREATE OR REPLACE TABLE Economia_g2.tabela_economia_mar19 AS (SELECT *  
FROM Economia_g2.tabela_economia  
WHERE Data>="2019-03-01" and Data<="2019-03-30");  
  
CREATE OR REPLACE TABLE Economia_g2.tabela_economia_mar20 AS (SELECT *  
FROM Economia_g2.tabela_economia  
WHERE Data>="2020-03-01" and Data<="2020-03-30");
```

Essas duas queries foram feitas a partir do dataframe de dados concatenados, onde selecionamos todas as informações do dataframe, mas apenas no mês de março dos anos de 2020 e 2019, para demonstrar a diferença entre flutuações comuns e a grande queda que aconteceu em março de 2020.

2.

```
CREATE OR REPLACE TABLE Economia_g2.tabela_economia_nov_mar20 AS (SELECT *  
FROM Economia_g2.tabela_economia  
WHERE Data>="2019-11-01" and Data<="2020-03-31");  
  
CREATE OR REPLACE TABLE Economia_g2.tabela_economia_nov_mar21 AS (SELECT *  
FROM Economia_g2.tabela_economia  
WHERE Data>="2020-11-01" and Data<="2021-03-31");
```

Essas duas queries foram retiradas do mesmo dataframe de dados concatenados, no entanto dessa vez delineando as variações do mercado desde os primeiros casos em novembro de 2019 até a declaração da pandemia de Covid-19 em março de 2020, e o mesmo período de 2020 a 2021.

3.

```
CREATE OR REPLACE TABLE Economia_g2.variacao_nov_mar20 AS (
SELECT Empresa, Subsetor, Codigo_Acao, Data, Preco_Abertura, Preco_Fechamento,
(Preco_Fechamento - Preco_Abertura) AS Variacao
FROM Economia_g2.tabela_economia_nov_mar20);

CREATE OR REPLACE TABLE Economia_g2.variacao_nov_mar21 AS (
SELECT Empresa, Subsetor, Codigo_Acao, Data, Preco_Abertura, Preco_Fechamento,
(Preco_Fechamento - Preco_Abertura) AS Variacao
FROM Economia_g2.tabela_economia_nov_mar21);
```

Essas queries são focadas na variação diária do preço das ações de empresas e das criptomoedas. Essas são retiradas das tabelas que criamos mostrando os períodos de novembro de 2019 a março de 2020 e de novembro de 2020 a março de 2021.

4.

```
CREATE OR REPLACE TABLE Economia_g2.media_variacao_nov_mar20 AS (
SELECT Subsetor, AVG(Variacao) AS Variacao_Media
FROM Economia_g2.variacao_nov_mar20
GROUP BY Subsetor);

CREATE OR REPLACE TABLE Economia_g2.media_variacao_nov_mar21 AS (
SELECT Subsetor, AVG(Variacao) AS Variacao_Media
FROM Economia_g2.variacao_nov_mar21
GROUP BY Subsetor);
```

Essas queries mostram a variação média do preço das ações por setor, durante o período de novembro de 2019 a março de 2020 e de novembro de 2020 a março de 2021. Mostrando o quanto as empresas foram afetadas pelas incertezas que a pandemia trouxe para o mercado.

5.

```
SELECT E.Data, E.Capitalizacao_Mercado AS Capitalizacao_Ethereum,
B.Capitalizacao_Mercado AS Capitalizacao_Bitcoin
from `Economia_g2.dados_bitcoin` B
RIGHT JOIN `Economia_g2.dados_ethereum` E
ON E.Data = B.Data
WHERE E.Data between '2019-11-01' and '2021-06-30'
ORDER BY Data
```

Essa query mostra a variação da capitalização de mercado das criptomoedas, desde os primeiros relatos de Covid-19, até junho de 2021, demonstrando a forma como a pandemia afetou o mercado de criptomoedas.

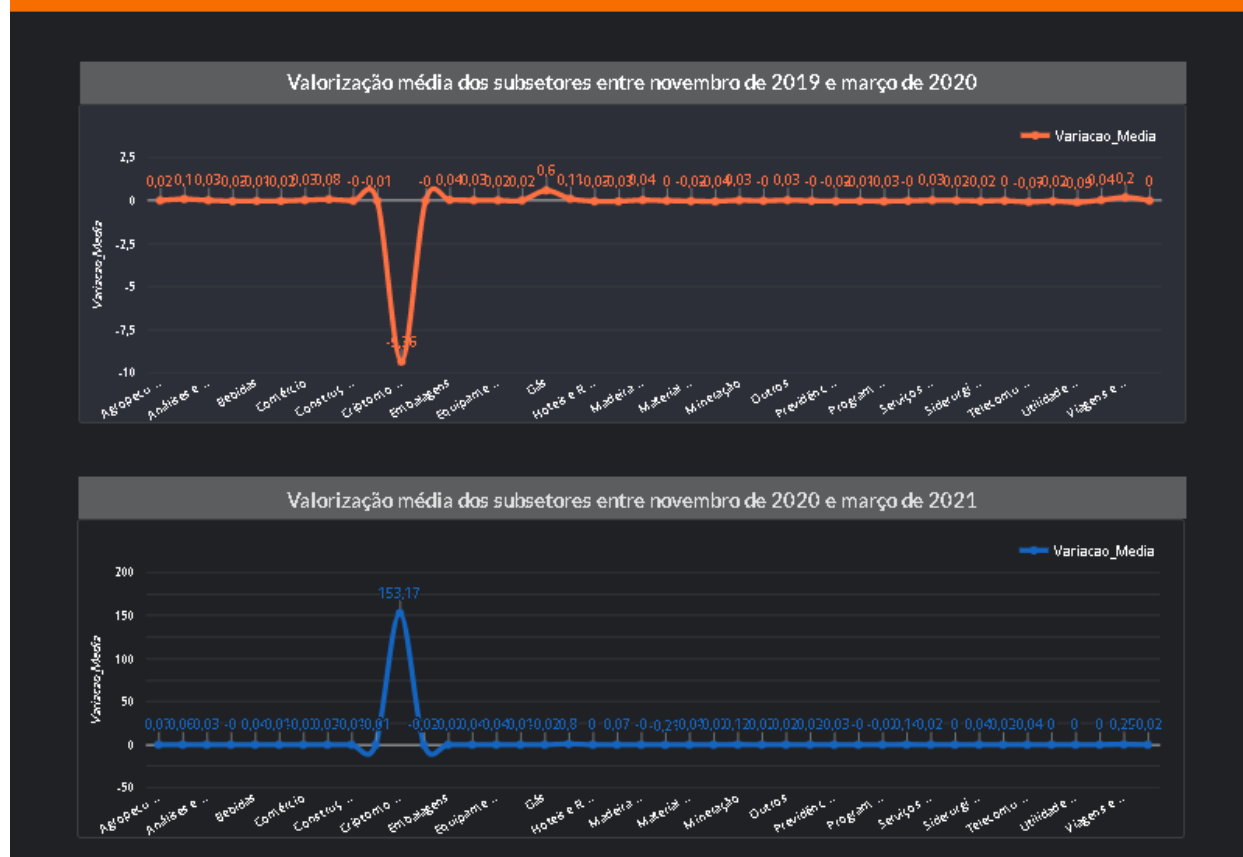
6.

```
SELECT E.Data, SAFE_SUBTRACT(E.Preco_Fechamento, E.Preco_Abertura) AS Variacao_Ethereum,  
B.Variacao_Preco AS Variacao_Bitcoin  
from `Economia_g2.dados_bitcoin` B  
RIGHT JOIN `Economia_g2.dados_ethereum` E  
ON E.Data = B.Data  
WHERE E.Data between '2019-11-01' and '2020-06-30'  
ORDER BY Data
```

Essa query mostra a variação de preço das criptomoedas, desde os primeiros relatos de Covid-19, até junho de 2021.

DATA STUDIO

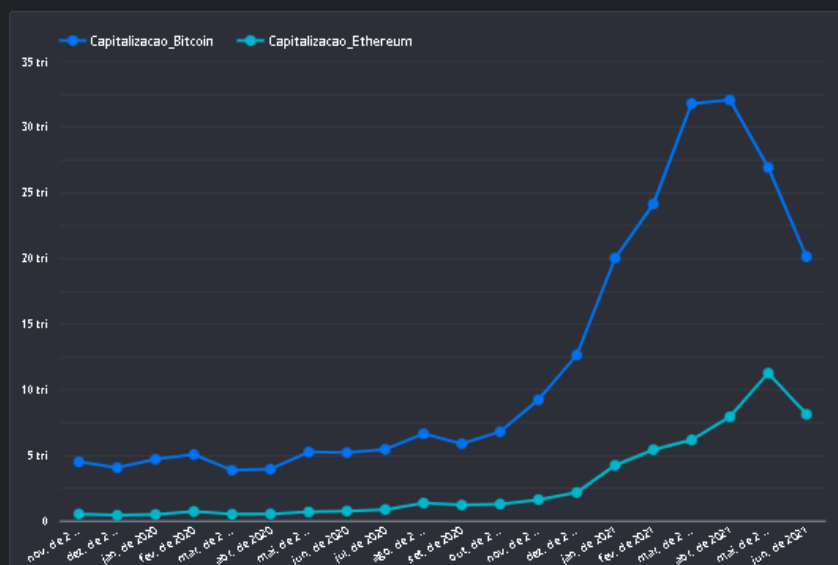
Análise comportamental do mercado de ações e criptomoedas pela pandemia



O primeiro gráfico nos traz um recorte dos subsetores entre o período de novembro de 2019 a março de 2020, onde conseguimos observar uma disparidade na desvalorização de quase 10% das criptomoedas, em contrapartida os outros subsetores que fazem parte da bolsa de valores houve também uma desvalorização, entretanto bem menor.


No segundo gráfico está sendo representado os mesmos subsetores só que 1 ano após a pandemia onde a vacinação já estava em andamento em todo o mundo e mais uma vez o subsetor da criptomoeda se destaca, mas desta vez com uma super valorização e os demais subsetores também apresentaram uma alta, mas não tão significativa.

Análise comportamental do mercado de ações e criptomoedas pela pandemia



Nesse terceiro gráfico o recorte traz a capitalização de mercado das duas criptomoedas trabalhadas no projeto que foi a ethereum e o bitcoin entre o período de novembro de 2019 a março de 2021. Podemos observar que em março de 2020 com a pandemia declarada ambas as moedas representaram uma queda na sua capitalização, entretanto já quase no final do ano de 2020 com a pandemia mais estabilizada seus preços subiram muito, inclusive o bitcoin registrou sua maior alta em abril de 2021. Também podemos ver analisando o gráfico que na metade do ano de 2021 ambas as moedas têm registrado uma baixa, mas agora devido a outros fatores como crise política.

Análise geral: Qualquer analista a partir dos tratamentos e das normalizações feitas pelo grupo seria capaz de garantir insights como esses. Claramente o mercado de criptomoeda tem uma alta volatilidade se comparada com as ações da bolsa brasileira. Um exemplo de utilidade desses insights representados pelo dashboard é que um investidor poderia facilmente decidir onde investir seu dinheiro de acordo com o seu perfil, se há um momento de crise talvez a melhor solução seja a bolsa pois representa menos riscos de perda, já se um período de crise estiver perto do fim ele deveria apostar suas fichas no



mercado de criptomoedas tendo em vista que ela pode alcançar uma supervalorização em um curto espaço de tempo.