



Projeto Final BC17

Engenharia de Dados - SoulCode Academy

Combustíveis

Equipe



Danilo Ferrari



Lilia Bakker



Emilson Cardoso



Stéphanie Pirajá

Agenda

- Objetivo
- Escopo
- Processos
- Pipelines
- Análises
- Agradecimentos

Objetivo

Analisar as métricas dos combustíveis no Brasil nos segundos semestres dos anos 2019, 2020 e 2021.*

*Período que representa o cenário antes e durante a Pandemia de Covid-19

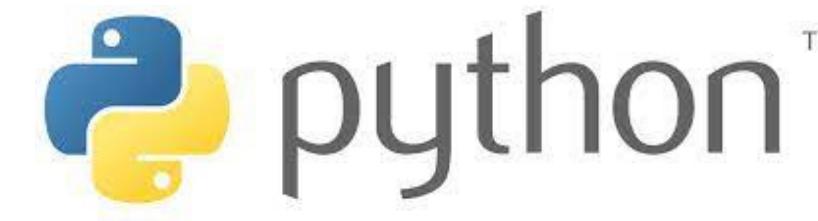
Escopo

- ◆ Iniciar com, no mínimo, 2 datasets em formatos diferentes
- ◆ Realizar o projeto em ambiente de nuvem
- ◆ Ter os datasets originais em um banco de dados SQL
- ◆ Utilizar a linguagem Python e suas bibliotecas Pandas e PySpark
- ◆ Enviar o dataset tratado para um banco de dados NoSQL
- ◆ Realizar os insights através do DataStudio

Ferramentas Utilizadas



Google Cloud



Processos

ETL

Análises

Carregamento

Transformação

Extração

Fonte de dados

Workflow



Coleta de Dados

▼ Leitura dos datasets

Combustiveis 2019-02

```
[ ] 1 df2019 = pd.read_parquet("gs://projeto-final-equipe1/Originais/combustiveis-2019-02.pq")
```

Combustiveis 2020-02

```
[ ] 1 df2020 = pd.read_parquet("gs://projeto-final-equipe1/Originais/combustiveis-2020-02.pq")
```

Combustiveis 2021-02

```
[ ] 1 df2021 = pd.read_csv("gs://projeto-final-equipe1/Originais/combustiveis-2021-02.csv",encoding="unicode_escape", sep=";")
```

Banco de Dados SQL

```
1 # Realizando conexão com MySQL no GCP
2 conexao = mysql.connector.connect(host='35.233.109.131', user='root', password='4d9Uu.2ymgZ0PzbD', db='projeto-final')
3 cursor = conexao.cursor()
4 engine = create_engine("mysql+pymysql://root:4d9Uu.2ymgZ0PzbD@35.233.109.131/projeto-final")

● ⓧ CREATE TABLE combustiveis_log(
    id_log int auto_increment PRIMARY KEY,
    data_log timestamp
)j

    DELIMITER $$

● USE `projeto-final`$$
● CREATE DEFINER = CURRENT_USER TRIGGER `projeto-final`.`combustiveis_AFTER_INSERT` AFTER INSERT ON `combustiveis` FOR EACH ROW
    BEGIN
        INSERT INTO `projeto-final`.`combustiveis_log` (`data_log`) VALUES (current_timestamp);
    END$$
    DELIMITER ;

● select * from combustiveis;
● select * from combustiveis_log;
```

Banco de Dados SQL

Inserção dos dados no MySQL

```
[ ] 1 #Renomeando as colunas para padronizar na inserção do MySQL
2 df2019.columns = ['Regiao_Sigla', 'Estado_Sigla', 'Municipio', 'Revenda', 'CNPJ_Revenda', 'Rua', 'Numero',
3 | | | | | | 'Complemento', 'Bairro', 'CEP', 'Produto', 'Data_Coleta', 'Valor_Venda', 'Valor_Compra',
4 | | | | | | 'Unidade_Medida', 'Bandeira']
```

```
[ ] 1 # Inserindo os dados no banco MySQL
2 df2019.to_sql('combustiveis', engine, if_exists='append', index= False)
```

✓	17	19:34:14	select * from combustiveis	1202792 row(s) returned
✓	18	19:38:16	select * from combustiveis_log	1202792 row(s) returned

Banco de Dados SQL

Leitura dos dados do MySQL

```
[ ] 1 # Leitura dos dados MySQL em Dataframe  
2 dfproj = pd.read_sql("select * from combustiveis", conexao)
```

```
[ ] 1 # Verificando a leitura dos dados  
2 dfproj.head(5)
```

Pré-análise dos Dados

```
[ ] 1 #Verificando informações do dataset  
2 dfproj.info()
```

```
[ ] 1 #Verificando valores nulos  
2 dfproj.isnull().sum()
```

```
[ ] 1 #Verificando inconsistencias na coluna "Regiao_Sigla"  
2 dfproj["Regiao_Sigla"].unique()
```

Tratamentos

- Dropar as colunas: ID, Rua, Bairro, Numero, Complemento, por não serem consideradas importantes para nosso objetivo (Insights finais), e por ter o CEP como informação redundante
- Dropar a coluna Valor_Compra por aproximadamente 82% dos valores serem nulos
- Trocar as "," por "." na coluna Valor_Venda

Tratamento dos Dados com Pandas

```
[ ] 1 #Backup do Dataframe  
2 dfproj_backup = dfproj.copy()  
  
[ ] 1 #Drop de colunas  
2 dfproj.drop(columns=["id", "Rua", "Numero", "Complemento", "Bairro", "Valor_Compra"], inplace=True)  
  
[ ] 1 #Verificando se existem valores duplicados  
2 dfproj.duplicated().sum()  
  
[ ] 1 #Renomeando as colunas  
2 dfproj.columns=["Regiao", "Estado", "Municipio", "Revenda", "CNPJ", "CEP", "Produto", "Data", "Valor_Venda", "Unidade_Medida", "Bandeira"]  
  
[ ] 1 #Alterando "," por "." na coluna Valor_Venda  
2 dfproj["Valor_Venda"] = dfproj["Valor_Venda"].str.replace(",",".")  
  
[ ] 1 #Verificando alterações  
2 dfproj.head(5)  
  
[ ] 1 #Salvando um arquivo parquet com os dados pré-tratados no GCP (Bucket)  
2 dfproj.to_parquet("gs://projeto-final-equipe1/Pre-Tratado/Combustiveis.pq", index=False)
```

Leitura dos Dados - PySpark

```
[ ] 1 #Faz a conexão com a sparksession (conexão entre pyspark e spark), já informando a API que faz conexão com o GCP
2 spark = (
3     SparkSession.builder
4         .master ('local')
5         .appName('projeto-final')
6         .config('spark.ui.port','4050')
7         .config('spark.jars','https://storage.googleapis.com/hadoop-lib/gcs/gcs-connector-hadoop2-latest.jar')
8         .getOrCreate()
9 )
```

```
[ ] 1 #Leitura dos dados diretamente do bucket GCP para um Dataframe
2 dfSpark = (spark.read.format("parquet")
3             .option("header", True)
4             .option("delimiter", ",")
5             .load("gs://projeto-final-equipe1/Pre-Tratado/Combustiveis.pq", schema=my_schema)
6 )
```

Tratamento dos Dados - PySpark

```
[ ] 1 #Verificar o tipo de dados de cada coluna  
2 dfSpark.printSchema()
```

```
[ ] 1 #Verificar os dados lidos  
2 dfSpark.show(5)
```

```
[ ] 1 #Converter o tipo da coluna "Valor_Venda" de StringType para FloatType  
2 dfSpark = dfSpark.withColumn("Valor_Venda" ,dfSpark["Valor_Venda"].cast(FloatType()))
```

```
[ ] 1 #Transformara coluna "Data" de StringType pra DateType  
2 dfSpark = dfSpark.withColumn("Data", F.to_date(F.col('Data'), 'dd/MM/yyyy'))
```

Carregamento

Load para o GCP

```
[ ] 1 #Salvar o dataframe em um arquivo Parquet no Cloud Storage (bucket)
  2 (dfSpark.write.format("parquet").option("header",True).
  3 | save("gs://projeto-final-equipe1/Tratados/Combustiveis-Tratados.pq",mode="overwrite"))
```

Buckets > projeto-final-equipe1 > Tratados 

UPLOAD FILES

UPLOAD FOLDER

CREATE FOLDER

MANAGE HOLDS

DOWNLOAD

DELETE

Filter by name prefix only ▾

 Filter Filter objects and folders

 Show

<input type="checkbox"/>	Name	Size	Type	Created 	Storage class	Last mod
<input type="checkbox"/>	Combustiveis-Tratados.pq/	—	Folder	—	—	—

Banco de Dados NoSQL

Conexão com o MongoDB Atlas

```
[ ] 1 CONNECTION_STRING = "mongodb+srv://soulcode:a1b2c3@projetopessoalbc17.blwk4.mongodb.net/?retryWrites=true&w=majority"

[ ] 1 client = MongoClient(CONNECTION_STRING)

[ ] 1 dbname = client['Projeto_Final']

[ ] 1 collection_name = dbname['Combustíveis']
```

Load para o MongoDB Atlas

```
[ ] 1 # Transformando o DataFrame em dicionário
2 dados = df_tratado.to_dict('records')

[ ] 1 # Inserindo os dados no banco MongoDB
2 collection_name.insert_many(dados)
```

Projeto_Final

DATABASE SIZE: 355.82MB INDEX SIZE: 35.36MB TOTAL COLLECTIONS: 1

[CREATE COLLECTION](#)

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
Combustíveis	1202792	355.82MB	311B	1	35.36MB	35.36MB

Pipelines

Configuração da Pipeline

```
#Opções para configuração da criação do modelo de pipeline no Cloud Storage
pipeline_options = {
    "project": "projeto-final-352219",
    "runner": "DataflowRunner",
    "region": "Multi-region",
    "staging_location": "gs://projeto-final-equipe1/Staging",
    "temp_location": "gs://projeto-final-equipe1/Staging",
    "template_location": "gs://projeto-final-equipe1/Models/modelo_batch_anos",
}
pipeline_options = PipelineOptions.from_dictionary(pipeline_options)
```

```
#Criação do schema para salvar o arquivo em parquet
schema_map = pyarrow.schema ({
    "Regiao": pyarrow.string(),
    "Estado": pyarrow.string(),
    "Municipio": pyarrow.string(),
    "Revenda": pyarrow.string(),
    "CNPJ": pyarrow.string(),
    "CEP": pyarrow.string(),
    "Produto":pyarrow.string(),
    "Data": pyarrow.date64(),
    "Valor_Venda": pyarrow.float64(),
    "Unidade_Medida": pyarrow.string(),
    "Bandeira": pyarrow.string(),
    "Ano": pyarrow.int64(),
    "Mes": pyarrow.int64()
})
```

Pipelines

```
#Criação da Pipeline
p1 = beam.Pipeline(options=pipeline_options)

combustiveis2019=
    p1
    |"01.Leitura dos datasets">>> beam.io.ReadFromParquet(path)
    |"02.Filtrar por ano">>> beam.Filter(lambda record: record["Ano"] == 2019)
    |'03.Salvar resultado'>> beam.io.WriteToParquet("gs://projeto-final-equipe1/Pipeline/CombustiveisAno2019",schema=schema_map)
)
combustiveis2020=
    p1
    |"11.Leitura dos datasets">>> beam.io.ReadFromParquet(path)
    |"12.Filtrar por ano">>> beam.Filter(lambda record: record["Ano"] == 2020)
    |'13.Salvar resultado'>> beam.io.WriteToParquet("gs://projeto-final-equipe1/Pipeline/CombustiveisAno2020",schema=schema_map)
)
combustiveis2021=
    p1
    |"21.Leitura dos datasets">>> beam.io.ReadFromParquet(path)
    |"22.Filtrar por ano">>> beam.Filter(lambda record: record["Ano"] == 2021)
    |'23.Salvar resultado'>> beam.io.WriteToParquet("gs://projeto-final-equipe1/Pipeline/CombustiveisAno2021",schema=schema_map)
)
p1.run()
```

Pipelines

[Pipeline-Final](#) [CLONE](#) [+ IMPORT AS PIPELINE](#) [SHARE](#)

[JOB GRAPH](#) [EXECUTION DETAILS](#) [JOB METRICS](#) [RECOMMENDATIONS](#)

Job steps view
Graph view ▾

CLEAR SELECTION

[12]: 21Lei...s datasets ✓ Succeeded 18 sec 1 of 1 stage succeeded	[12]: 11Lei...s datasets ✓ Succeeded 29 sec 1 of 1 stage succeeded	[12]: 01Lei...s datasets ✓ Succeeded 17 sec 1 of 1 stage succeeded
[12]: 22Fil...ar por ano ✓ Succeeded 1 sec 1 of 1 stage succeeded	[12]: 12Fil...ar por ano ✓ Succeeded 2 sec 1 of 1 stage succeeded	[12]: 02Fil...ar por ano ✓ Succeeded 1 sec 1 of 1 stage succeeded
[12]: 23Sal... resultado ✓ Succeeded 15 sec 5 of 5 stages succeeded	[12]: 13Sal... resultado ✓ Succeeded 15 sec 5 of 5 stages succeeded	[12]: 03Sal... resultado ✓ Succeeded 17 sec 5 of 5 stages succeeded

Job info

Job name	Pipeline-Final
Job ID	2022-06-08_07_50_21-14617154053738238354
Job type	Batch
Job status	✓ Succeeded
SDK version	Apache Beam Python 3.7 SDK 2.39.0
Job region	us-central1
Worker location	us-central1-b
Current workers	0
Latest worker status	Worker pool stopped.
Start time	June 8, 2022 at 11:50:21 AM GMT-3
Elapsed time	7 min 33 sec
Encryption type	Google-managed key

Resource metrics

Current vCPUs	1
Total vCPU time	0.109 vCPU hr
Current memory	3.75 GB

Análises

Filtros e agrupamentos com Pyspark

```
[ ] 1 #Filtrar os dados do Ano de 2019, agrupar por Produtos e fazer a média, o máximo e o mínimo de cada produto
2 ano2019 = dfSpark["Ano"] == 2019
3 (dfSpark.filter(ano2019).groupBy("Produto").agg(F.mean("Valor_Venda")).alias("Media_Preco"),
4 ||| F.max("Valor_Venda").alias("Preco_Max"),F.min("Valor_Venda").alias("Preco_Min")).show()
```

Produto	Media_Preco	Preco_Max	Preco_Min
DIESEL S10	3.732812434513274	5.089	2.999
DIESEL	3.6475548843968655	4.99	2.999
ETANOL	3.161922439504206	5.47	2.099
GNV	3.2474904105460367	4.559	2.489
GASOLINA	4.4251792166388055	5.859	3.559

SparkSQL

Bandeiras mais presentes em cada região

```
[ ] 1 spark.sql("WITH n AS (SELECT Bandeira, Regiao, count(Bandeira) AS Frequencia FROM combustiveis \
2 WHERE Regiao = 'N' GROUP BY Bandeira, Regiao ORDER BY Frequencia DESC LIMIT 10), \
3 ne AS (SELECT Bandeira, Regiao, count(Bandeira) AS Frequencia FROM combustiveis \
4 WHERE Regiao = 'NE' GROUP BY Bandeira, Regiao ORDER BY Frequencia DESC LIMIT 10), \
5 s AS (SELECT Bandeira, Regiao, count(Bandeira) AS Frequencia FROM combustiveis \
6 WHERE Regiao = 'S' GROUP BY Bandeira, Regiao ORDER BY Frequencia DESC LIMIT 10), \
7 se AS (SELECT Bandeira, Regiao, count(Bandeira) AS Frequencia FROM combustiveis \
8 WHERE Regiao = 'SE' GROUP BY Bandeira, Regiao ORDER BY Frequencia DESC LIMIT 10), \
9 co AS (SELECT Bandeira, Regiao, count(Bandeira) AS Frequencia FROM combustiveis \
10 WHERE Regiao = 'CO' GROUP BY Bandeira, Regiao ORDER BY Frequencia DESC LIMIT 10) \
11 \
12 SELECT n.Bandeira, n.Regiao, n.Frequencia, ne.Regiao, ne.Frequencia, s.Regiao, s.Frequencia, se.Regiao, se.Frequencia, \
13 co.Regiao, co.Frequencia FROM n INNER JOIN ne ON n.Bandeira = ne.Bandeira INNER JOIN s ON ne.Bandeira = s.Bandeira \
14 INNER JOIN se ON s.Bandeira = se.Bandeira INNER JOIN co ON se.Bandeira = co.Bandeira ORDER BY n.Frequencia DESC").show(truncate = False)
```

Bandeira	Regiao Frequencia											
	Regiao	Frequencia	Regiao	Frequencia	Regiao	Frequencia	Regiao	Frequencia	Regiao	Frequencia	Regiao	Frequencia
BRANCA	N	15886	NE	81164	S	56841	SE	225338	CO	44397		
PETROBRAS DISTRIBUIDORA S.A.	N	13544	NE	43511	S	24680	SE	76971	CO	17670		
IPIRANGA	N	12961	NE	20428	S	51963	SE	104825	CO	13672		
VIBRA ENERGIA	N	7831	NE	25337	S	14765	SE	45559	CO	9339		
RAIZEN	N	1651	NE	37717	S	24706	SE	112799	CO	14778		
ALESAT	N	423	NE	7169	S	2760	SE	14920	CO	2095		

Big Query - GCP

The screenshot shows a BigQuery query editor. On the left, a sidebar displays pinned projects, with 'projeto-final-352219' expanded to show 'combustiveis' and 'combustiveis' selected. The main area contains a complex SQL query with numbered lines:

```
1 WITH n AS (SELECT Produto, Regiao, avg(Valor_Venda) AS Media_Valor FROM `projeto-final-352219.combustiveis.combustiveis` WHERE Regiao = 'N' GROUP BY Produto, Regiao ORDER BY Media_Valor DESC),
2 ne AS (SELECT Produto, Regiao, avg(Valor_Venda) AS Media_Valor FROM `projeto-final-352219.combustiveis.combustiveis` WHERE Regiao = 'NE' GROUP BY Produto, Regiao ORDER BY Media_Valor DESC),
3 s AS (SELECT Produto, Regiao, avg(Valor_Venda) AS Media_Valor FROM `projeto-final-352219.combustiveis.combustiveis` WHERE Regiao = 'S' GROUP BY Produto, Regiao ORDER BY Media_Valor DESC),
4 se AS (SELECT Produto, Regiao, avg(Valor_Venda) AS Media_Valor FROM `projeto-final-352219.combustiveis.combustiveis` WHERE Regiao = 'SE' GROUP BY Produto, Regiao ORDER BY Media_Valor DESC),
5 co AS (SELECT Produto, Regiao, avg(Valor_Venda) AS Media_Valor FROM `projeto-final-352219.combustiveis.combustiveis` WHERE Regiao = 'CO' GROUP BY Produto, Regiao ORDER BY Media_Valor DESC)
6
7 SELECT n.Produto, n.Regiao, n.Media_Valor, ne.Regiao, ne.Media_Valor, s.Regiao, s.Media_Valor, se.Regiao, se.Media_Valor,
8 co.Regiao, co.Media_Valor FROM n INNER JOIN ne ON n.Produto = ne.Produto INNER JOIN s ON ne.Produto = s.Produto
9 INNER JOIN se ON s.Produto = se.Produto INNER JOIN co ON se.Produto = co.Produto ORDER BY n.Media_Valor DESC
```

A status bar at the top right indicates: "This query will process 25.52 MB when run."

Query results

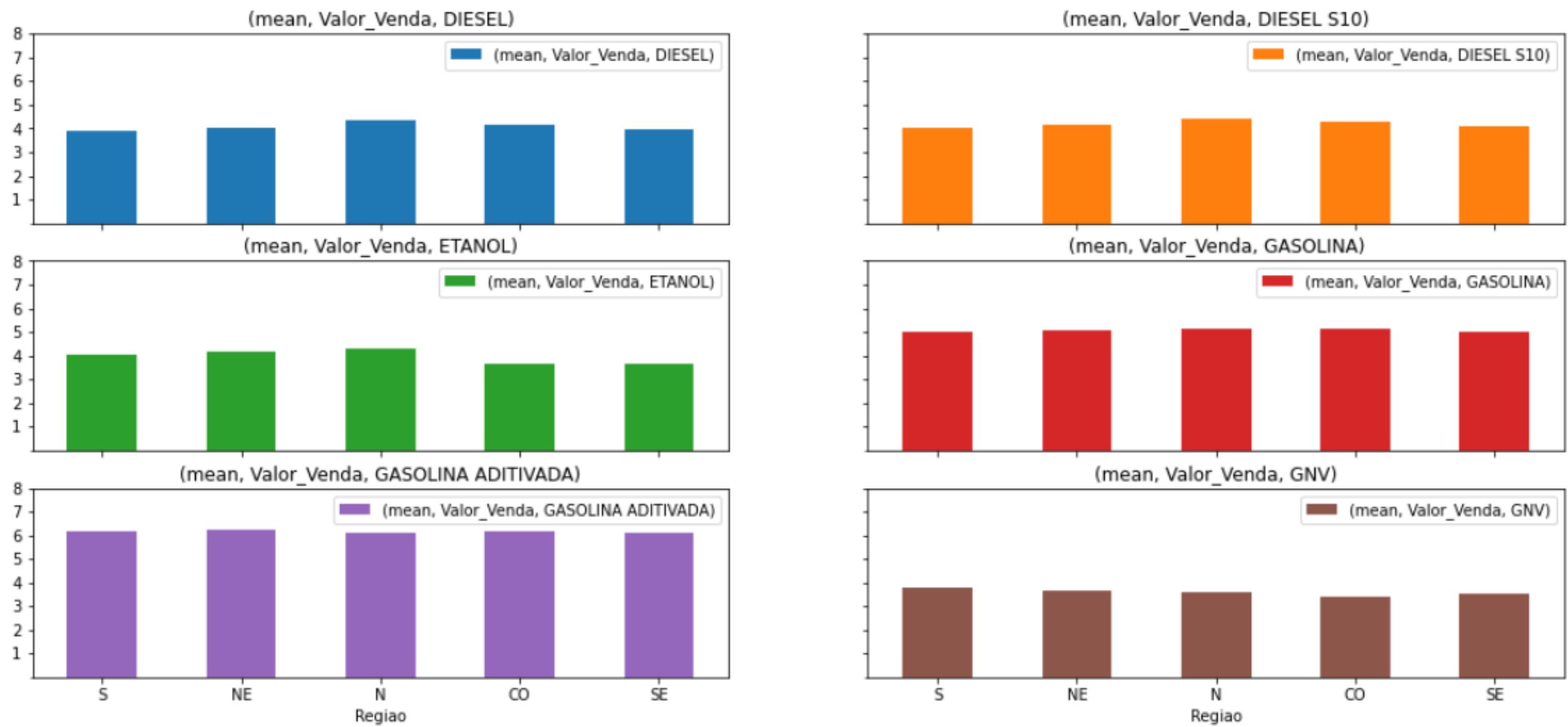
[SAVE RESULTS](#) [EXPLORE DATA](#)

JOB INFORMATION		RESULTS		JSON		EXECUTION DETAILS					
Row	Produto	Regiao	Media_Valor	Regiao_1	Media_Valor_1	Regiao_2	Media_Valor_2	Regiao_3	Media_Valor_3	Regiao_4	Media_Valor_4
1	GASOLINA ADITIVADA	N	6.0783930200939738	NE	6.2239482995626583	S	6.1832845159561254	SE	6.1067921298246262	CO	6.11
2	GASOLINA	N	5.1091532623117439	NE	5.0941801874711166	S	4.99109162535654	SE	5.0342782566230051	CO	5.11
3	DIESEL S10	N	4.4048265863342433	NE	4.1704705616630457	S	4.0012634499643429	SE	4.1159976501966273	CO	4.21
4	DIESEL	N	4.3601426812616317	NE	4.0519950546407557	S	3.91884440298943	SE	3.9732053720461287	CO	4.11
5	ETANOL	N	4.2854152957701039	NE	4.1483918907407435	S	4.0297637656120111	SE	3.6641345598432342	CO	3.61
6	GNV	N	3.5916000366210938	NE	3.6677979404805026	S	3.762021594330005	SE	3.5482135434415243	CO	3.31

Plotagem com Pandas

```
[ ] 1 df_pivotado4 = pd.pivot_table(df,index=["Regiao"],values=["Valor_Venda"],columns=["Produto"], aggfunc=[np.mean], fill_value=0, sort=False)  
[ ] 1 df_pivotado4.plot(kind = 'bar', subplots=True, layout=(6,2), sharey=True, figsize = (14,14), title= "Média de valor de venda de cada combustível por região", rot=0)
```

Média de valor de venda de cada combustível por região



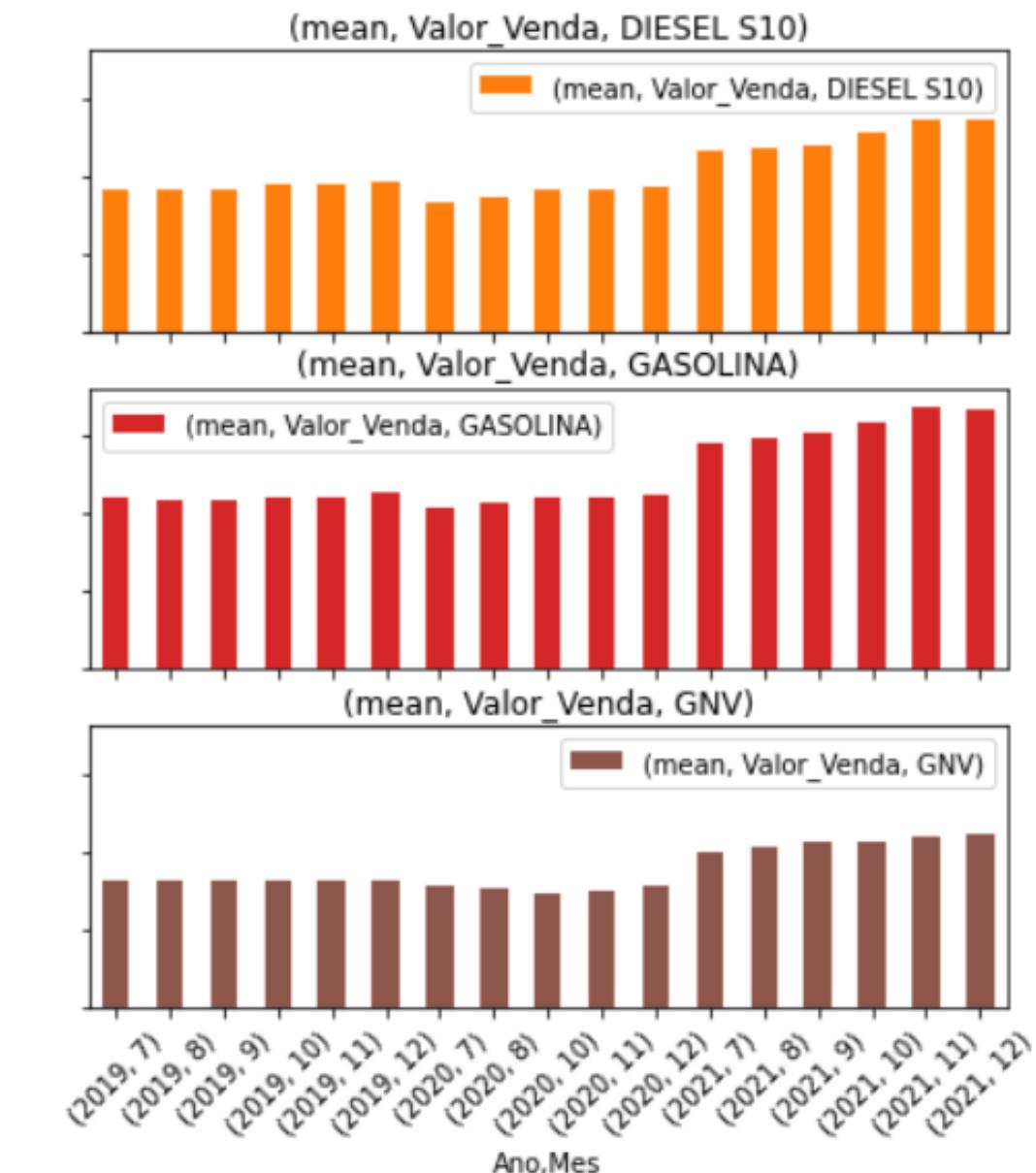
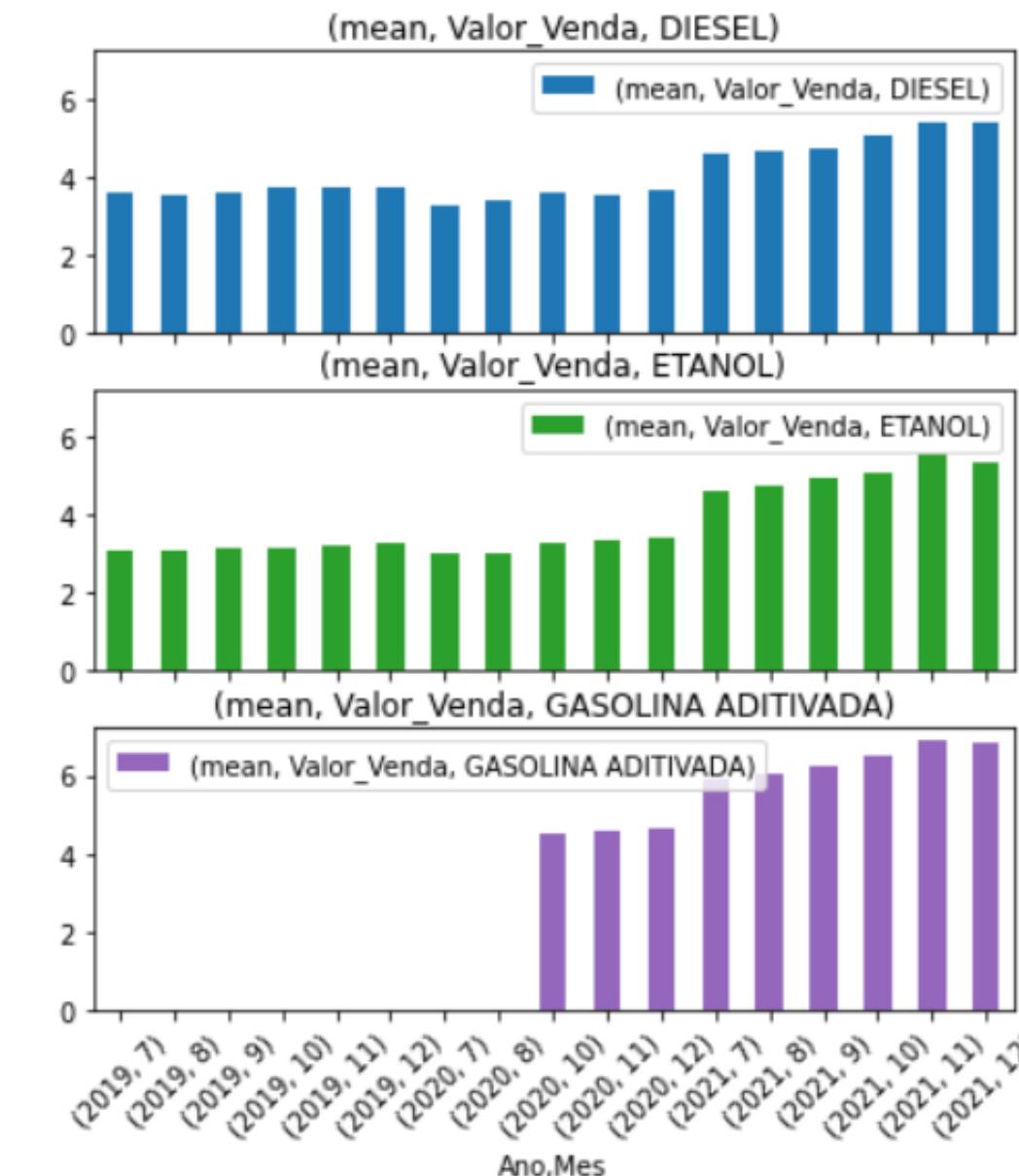
Plotagem com Pandas

```
[ ] 1 # Pivot table p média da variação de venda de combustível por mes/ano  
2 df_variacao = pd.pivot_table(df,index=["Ano","Mes"],values=["Valor_Venda"],columns=["Produto"], aggfunc=[np.mean], fill_value=0, sort=False)
```

```
[ ] 1 df_variacao.plot(kind = 'bar', subplots=True, layout=(6,2), sharey=True, figsize = (14,14),\n2 | | | | | title= "Variação do valor de venda de combustíveis ao longo do 2º semestre dos anos 2019, 2020 e 2021",\n3 | | | | | rot=45)
```

Obs:

- **Registros referentes a Gasolina Adivitada tiveram início em 10/2020.**
 - **No mês de setembro de 2020 não houve registros.**



Combustíveis na Mídia



Ministério de Minas e Energia

Órgãos do Governo Acesso à Informação Legislação Acessibilidade



Entrar

≡ Agência Nacional do Petróleo, Gás Natural e Biocombustíveis

O que você procura?



[Home](#) > [Canais de Atendimento](#) > [Imprensa](#) > [Notícias e comunicados](#) > Comercialização de combustíveis em 2020 teve queda de 5,97% na comparação com 2019 devido à pandemia

Comercialização de combustíveis em 2020 teve queda de 5,97% na comparação com 2019 devido à pandemia

Publicado em 06/04/2021 17h23 | Atualizado em 12/04/2021 09h03

Compartilhe:



https://www.gov.br/anp/pt-br/canais_atendimento/imprensa/noticias-comunicados/comercializacao-de-combustiveis-em-2020-teve-queda-de-5-97-na-comparacao-com-2019-devido-a-pandemia

Dashboard Data Studio

SOUL CODE

Projeto Final - BC17

1 de jul. de 2019 - 31 de dez. de 2019

Combustíveis

Gasolina

R\$ 4,43

Gasolina
Aditivada

0

Etanol

R\$ 3,16

Diesel

R\$ 3,65

Diesel S10

R\$ 3,65

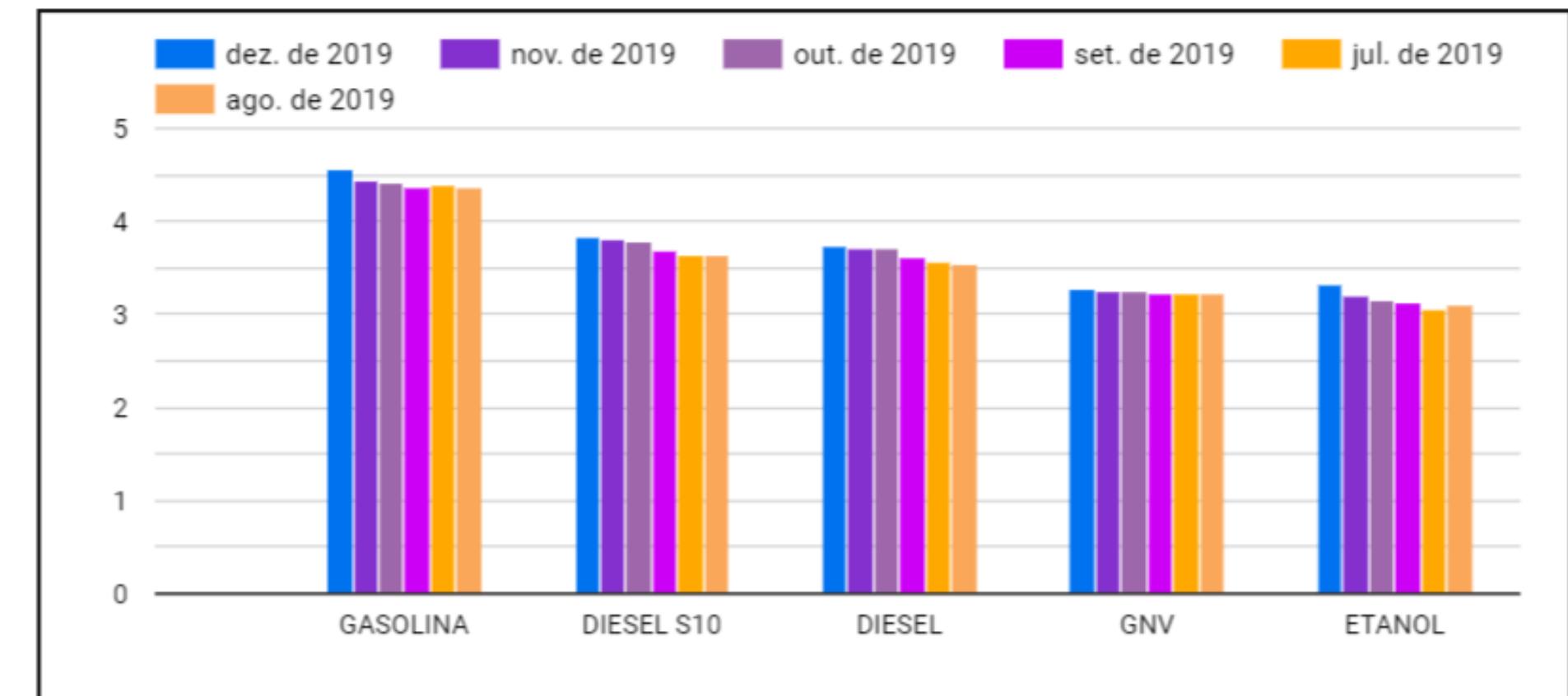
GNV

R\$ 3,65

Incidência das Bandeiras

BRANCA	34,6%
PETROBRAS DISTRIBUIDORA S.A.	24,52%
IPIRANGA	16,82%
RAIZEN	15,37%
ALESAT	2,34%
SABBÁ	1,16%
RAIZEN MIME	0,55%
ATEM'S	0,45%
SP	0,42%
EQUADOR	0,32%

Variação da média dos preços dos combustíveis nos meses



Dashboard Data Studio

SOUL CODE

Projeto Final - BC17

1 de jul. de 2020 - 31 de dez. de 2020

Combustíveis

Gasolina
R\$ 4,28

Gasolina Aditivada
R\$ 4,59

Etanol
R\$ 3,14

Diesel
R\$ 3,38

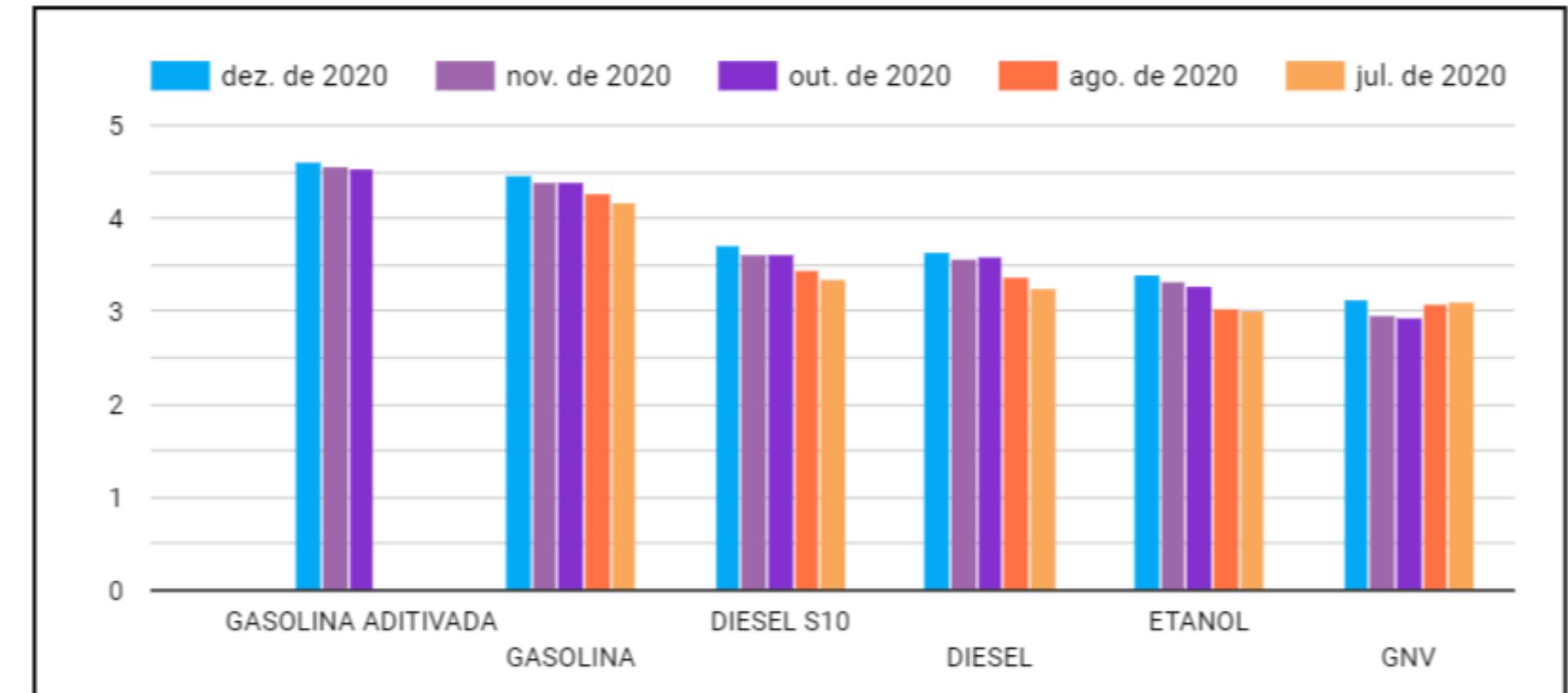
Diesel S10
R\$ 3,38

GNV
R\$ 3,38

Incidência das Bandeiras

BRANCA	34,07%
PETROBRAS DISTRIBUIDORA S.A.	23,35%
IPIRANGA	17,25%
RAIZEN	16,67%
ALESAT	2,34%
SABBÁ	1,34%
ATEM'S	0,75%
RAIZEN MIME	0,53%
SP	0,39%
EQUADOR	0,35%

Variação da média dos preços dos combustíveis nos meses



Dashboard Data Studio

SOUL CODE

Projeto Final - BC17

1 de jul. de 2021 - 31 de dez. de 2021

Gasolina

R\$ 6,27

Gasolina
Aditivada

R\$ 6,42

Etanol

R\$ 5,06

Diesel

R\$ 4,98

Diesel S10

R\$ 4,98

GNV

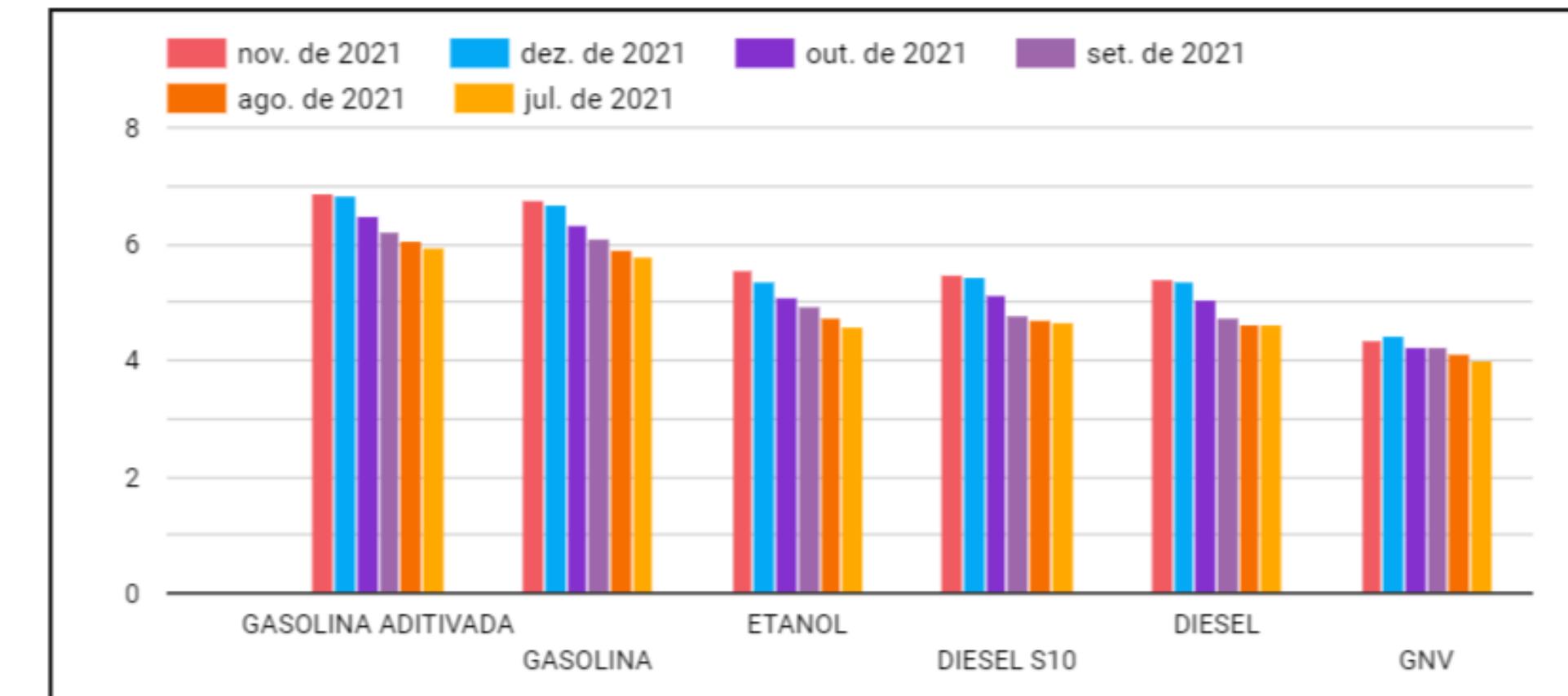
R\$ 4,98

Combustíveis

Incidência das Bandeiras

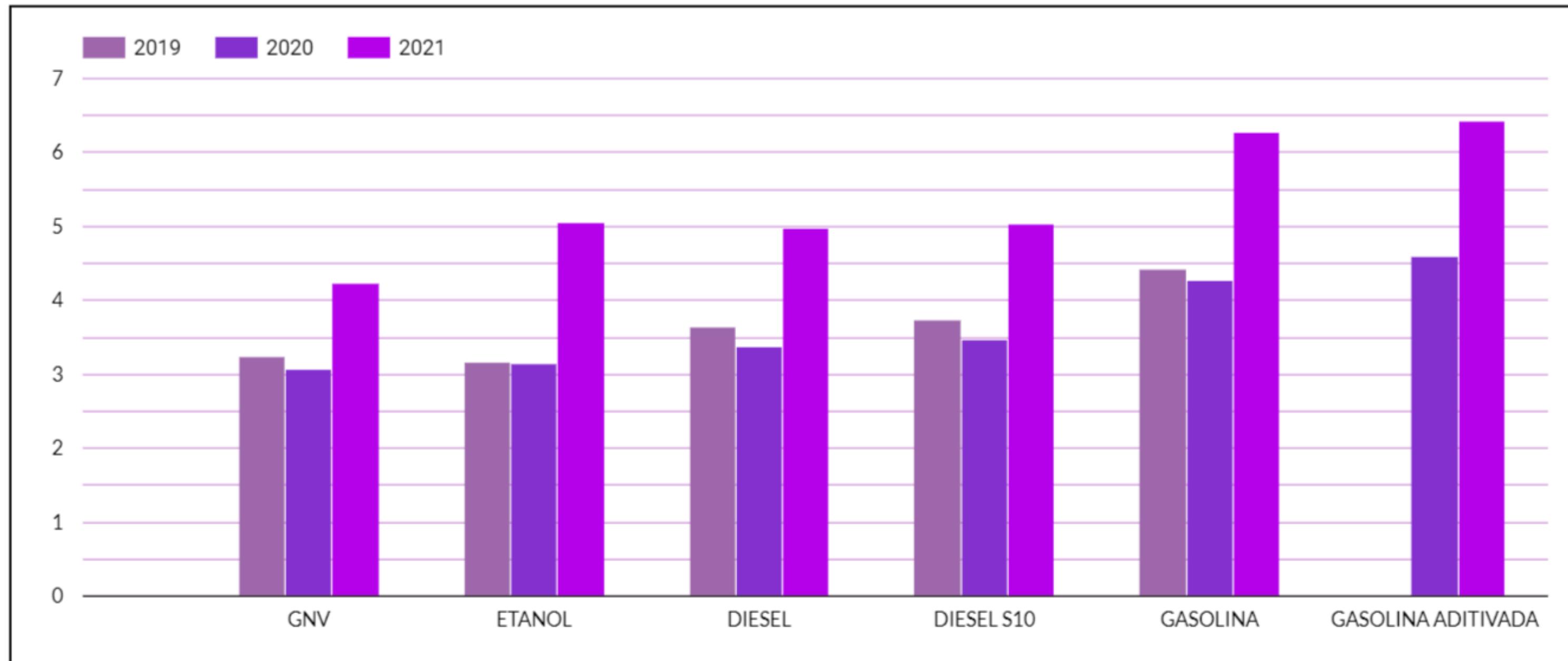
BRANCA	36,43%
VIBRA ENERGIA	21,75%
IPIRANGA	16,94%
RAIZEN	16,19%
ALESAT	2,18%
SABBÁ	1,12%
RAIZEN MIME	0,53%
RODOIL	0,48%
ATEM'S	0,47%
EQUADOR	0,38%

Variação da média dos preços dos combustíveis nos meses



Dashboard Data Studio

Valor médio dos combustíveis ao longo dos segundos semestres de 2019, 2020 e 2021



Combustíveis na Mídia

Gasolina e diesel ficam quase 45% mais caros em 2021; preço do etanol subiu 60%

Mesmo com a última redução, o aumento da gasolina deve ser cerca de quatro vezes maior do que a inflação do ano

Por Larissa Albuquerque (com Cauê Lira)

30/12/2021 12h33 · Atualizado há 5 meses



<https://autoesporte.globo.com/seu-bolso/noticia/2021/12/gasolina-e-diesel-ficam-quase-45percent-mais-caros-em-2021-preco-do-etanol-subiu-60percent.ghml>

**"Em toda dificuldade existe uma
oportunidade!"**

Albert Einstein

Agradecimentos

- SoulCode Academy
- Instituto Ser +
- Lingopass
- Parceiros
- Colegas da turma

Agradecemos a todos!