E-COMMERCE

COMPARATIVO OLIST (BRASIL) E EUA

EQUIPE DO PROJETO

NICOLE SIQUEIRA

ANDRÉ LUIS

MARTA MARUBAYASHI

OSNI RAMILTON

PHILIPPE MAGNO

NOSSA EMPRESA

I.T DATA SEARCH

A EMPRESA I.T DATA SEARCH FOI CRIADA, PELOS ALUNOS DA SOULCODE COM INTUITO DE AJUDAR OS CLIENTES A ENCONTRAR ATRAVÉS DOS DADOS DAS SUAS EMPRESAS OS POTENCIAIS PROBLEMAS E, TAMBÉM, ATRAVÉS DELES, A SOLUÇÃO.

O PROJETO

REQUISITOS DO PROJETO

- OBRIGATORIAMENTE OS DATASETS DEVEM TER FORMATOS DIFERENTES.
- OPERAÇÕES COM PANDAS E PYSPARK (LIMPEZAS , TRANSFORMAÇÕES E NORMALIZAÇÕES)
- OS DATASETS DEVEM SER SALVOS E OPERADOS EM ARMAZENAMENTO CLOUD NA PLATAFORMA GCP
- OS DADOS TRATADOS DEVEM SER ARMAZENADOS OBRIGATORIAMENTE EM UM DATALAKE(GSTORAGE) , DW(BIGQUERY) OU EM AMBOS.
- OS DATASETS ORIGINAIS DEVEM SER ARMAZENADOS EM MYSQL
- OS DATAFRAME(S) RESULTANTE(S) DEVE(M) ESTAR EM UMA COLEÇÃO DO MONGODB ATLAS
- ANÁLISES DENTRO DO BIG QUERY UTILIZANDO A LINGUAGEM PADRÃO SQL
- DASHBOARD NO LOOKERSTUDIO PARA EXIBIÇÃO GRÁFICA DOS DADOS TRATADOS

SOBRE OS DADOS

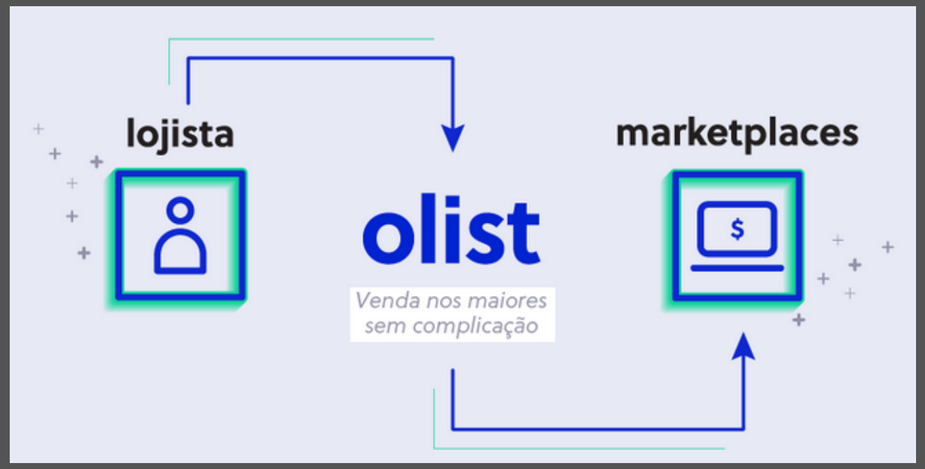
- ECONOMIA (FOCADA NA UTILIZAÇÃO DE MEIOS ELETRÔNICOS)
- DADOS E-COMMERCE OLIST STORE.
- OS DADOS PÚBLICOS FORAM DISPONIBILIZADOS PELA OLIST NA PLATAFORMA KAGGLE
- O DATASET US E-COMMERCE TAMBÉM FOI ENCONTRADO NA MESMA PLATAFORMA, E ELES ESTÃO PRESENTES NO PROJETO COM O INTUITO DE UMA BÁSICA ANÁLISE COMPARATIVA

SOBRE A OLIST

TRATA-SE DE UM UMA STARTUP BRASILEIRA QUE ATUA NO SEGMENTO DE E-COMMERCE.
DE UM LADO, O OLIST CONCENTRA VENDEDORES QUE DESEJAM ANUNCIAR EM MARKETPLACES COMO:

viavarejo · mercado livre · CASAS BAHIA · amazon · americanas

DO OUTRO LADO, CONCENTRA OS PRODUTOS DE TODOS OS VENDEDORES EM UMA LOJA ÚNICA QUE FICA VISÍVEL AO CONSUMIDOR FINAL.

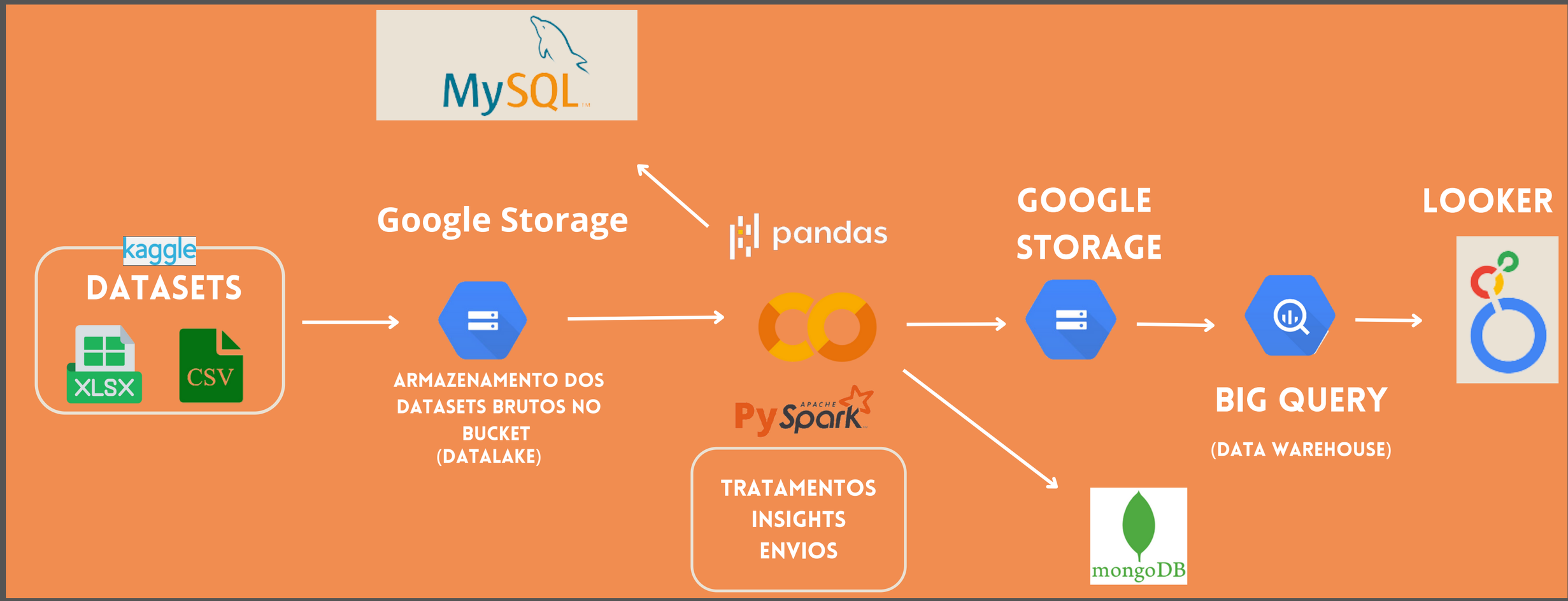


OBJETIVO DO PROJETO

O NOSSO OBJETIVO COM A ESCOLHA DESSES DATASETS, FOI DEMONSTRAR COMO O E-COMMERCE BRASILEIRO, É UM DOS PRINCIPAIS MEIOS DE CONSUMO E ENTENDER COMO FUNCIONAM ESSAS COMPRAS

O PROCESSO

WORKFLOW


MySQL
Google Storage
pandas
GOOGLE STORAGE
LOOKER
kaggle
DATASETS
XLSX
CSV
ARMAZENAMENTO DOS DATASETS BRUTOS NO BUCKET (DATALAKE)
PySpark
BIG QUERY
(DATA WAREHOUSE)
TRATAMENTOS
INSIGHTS
ENVIOS
mongoDB

OS DATASETS

DF1 = OLIST_CUSTOMERS_DATASET

DF2 = OLIST_GEOLOCATION_DATASET

DF3 = OLIST_ORDER_ITEMS_DATASET

DF4 = OLIST_ORDER_PAYMENTS_DATASET

DF5 = OLIST_ORDER_REVIEWS_DATASET

DF6 = OLIST_ORDERS_DATASET

DF7 = OLIST_PRODUCTS_DATASET

DF8 = OLIST_SELLERS_DATASET

DF9 = PRODUCT_CATEGORY_NAME_TRANSLATION

EXTRAÇÃO

```
[7]  df1.count()
```

```
99441
```

```
[8]  df2.count()
```

```
1000163
```

```
[9]  df3.count()
```

```
112650
```

```
[10]  df4.count()
```

```
103886
```

```
[11]  df5.count()
```

```
104162
```

```
[12]  df6.count()
```

```
99441
```

```
[13]  df7.count()
```

```
32951
```

```
[14]  df8.count()
```

```
3095
```

```
[15]  df9.count()
```

```
71
```

MYSQL

```
mydb = create_engine('mysql+pymysql://' + user + ':' + passw + '@' + host + ':' + str(port) + '/' + database , echo=False)

directory = r'gs://e-commerce-brazil/bases-originais-olist/'  # path of csv file
csvFileName = 'olist_customers_dataset.csv'
csvFileName2 = 'olist_geolocation_dataset.csv'
csvFileName3 = 'olist_order_items_dataset.csv'
csvFileName4 = 'olist_order_payments_dataset.csv'
csvFileName5 = 'olist_order_reviews_dataset.csv'
csvFileName6 = 'olist_orders_dataset.csv'
csvFileName7 = 'olist_products_dataset.csv'
csvFileName8 = 'olist_sellers_dataset.csv'
csvFileName9 = 'product_category_name_translation.csv'
```

MYSQL

```
df = pd.read_csv(os.path.join(directory, csvFileName ))
df2 = pd.read_csv(os.path.join(directory, csvFileName2 ))
df3 = pd.read_csv(os.path.join(directory, csvFileName3 ))
df4 = pd.read_csv(os.path.join(directory, csvFileName4 ))
df5 = pd.read_csv(os.path.join(directory, csvFileName5 ))
df6 = pd.read_csv(os.path.join(directory, csvFileName6 ))
df7 = pd.read_csv(os.path.join(directory, csvFileName7 ))
df8 = pd.read_csv(os.path.join(directory, csvFileName8 ))
df9 = pd.read_csv(os.path.join(directory, csvFileName9 ))
#Enviando para Mysql
df.to_sql(name=csvFileName[:-4], con=mydb, if_exists = 'replace', index=False)
df2.to_sql(name=csvFileName2[:-4], con=mydb, if_exists = 'replace', index=False)
df3.to_sql(name=csvFileName3[:-4], con=mydb, if_exists = 'replace', index=False)
df4.to_sql(name=csvFileName4[:-4], con=mydb, if_exists = 'replace', index=False)
#df5.to_sql(name=csvFileName5[:-4], con=mydb, if_exists = 'replace', index=False) <-- Erro (Incorrect string value)
df6.to_sql(name=csvFileName6[:-4], con=mydb, if_exists = 'replace', index=False)
df7.to_sql(name=csvFileName7[:-4], con=mydb, if_exists = 'replace', index=False)
df8.to_sql(name=csvFileName8[:-4], con=mydb, if_exists = 'replace', index=False)
df9.to_sql(name=csvFileName9[:-4], con=mydb, if_exists = 'replace', index=False)
```

PRÉ-ANALISE

order_id
112650
null
null
00010242fe8c5a6d1...
null
null
null
fffe41c64501cc87c...

order_id
103886
null
null
00010242fe8c5a6d1...
null
null
null
fffe41c64501cc87c...

PRÉ-ANALISE

Order Date	Unnamed: 1	Row ID	Order ID	Ship Mode	Customer ID	Segment	Country	City	State	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales
2020-01-01 00:00:00	NaN	849	CA-2017-107503	Standard Class	GA-14725	Consumer	United States	Lorain	Ohio	44052	East	FUR-FU-10003878	Furniture	Furnishings	Linden 10" Round Wall Clock, Black	48.896
2020-01-01 00:00:00	NaN	4010	CA-2017-144463	Standard Class	SC-20725	Consumer	United States	Los Angeles	California	90036	West	FUR-FU-10001215	Furniture	Furnishings	Howard Miller 11-1/2" Diameter Brentwood Wall ...	474.430
2020-01-01 00:00:00	NaN	6683	CA-2017-154466	First Class	DP-13390	Home Office	United States	Franklin	Wisconsin	53132	Central	OFF-BI-10002012	Office Supplies	Binders	Wilson Jones Easy Flow II Sheet Lifters	3.600
2020-01-01 00:00:00	NaN	8070	CA-2017-151750	Standard Class	JM-15250	Consumer	United States	Huntsville	Texas	77340	Central	OFF-ST-10002743	Office Supplies	Storage	SAFCO Boltless Steel Shelving	454.560
2020-01-01 00:00:00	NaN	8071	CA-2017-151750	Standard Class	JM-15250	Consumer	United States	Huntsville	Texas	77340	Central	FUR-FU-10002116	Furniture	Furnishings	Tenex Carpeted, Granite-Look or Clear Contempo...	141.420
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
30-12-20	NaN	908	CA-2017-143259	Standard Class	PO-18865	Consumer	United States	New York City	New York	10009	East	TEC-PH-10004774	Technology	Phones	Gear Head AU3700S Headset	90.930
30-12-20	NaN	909	CA-2017-143259	Standard Class	PO-18865	Consumer	United States	New York City	New York	10009	East	OFF-BI-10003684	Office Supplies	Binders	Wilson Jones Legal Size Ring Binders	52.776
30-12-20	NaN	1297	CA-2017-115427	Standard Class	EB-13975	Corporate	United States	Fairfield	California	94533	West	OFF-BI-10002103	Office Supplies	Binders	Cardinal Slant-D Ring Binder, Heavy Gauge Vinyl	13.904
30-12-20	NaN	1298	CA-2017-115427	Standard Class	EB-13975	Corporate	United States	Fairfield	California	94533	West	OFF-BI-10004632	Office Supplies	Binders	GBC Binding covers	20.720
30-12-20	NaN	5092	CA-2017-156720	Standard Class	JM-15580	Consumer	United States	Loveland	Colorado	80538	West	OFF-FA-10003472	Office Supplies	Fasteners	Bagged Rubber Bands	3.024

ANÁLISE

```
[16] df1.dtypes
```

```
[('customer_id', 'string'),
 ('customer_unique_id', 'string'),
 ('customer_zip_code_prefix', 'string'),
 ('customer_city', 'string'),
 ('customer_state', 'string')]
```

```
[17] df2.dtypes
```

```
[('geolocation_zip_code_prefix', 'string'),
 ('geolocation_lat', 'string'),
 ('geolocation_lng', 'string'),
 ('geolocation_city', 'string'),
 ('geolocation_state', 'string')]
```

```
[18] df3.dtypes
```

```
[('order_id', 'string'),
 ('order_item_id', 'string'),
 ('product_id', 'string'),
 ('seller_id', 'string'),
 ('shipping_limit_date', 'string'),
 ('price', 'string'),
 ('freight_value', 'string')]
```

ANÁLISE

summary	review_id	order_id	review_score
count	104161	101926	101782
mean	4.5	0.0	4.087204087597126
stddev	0.7071067811865476	0.0	1.375004642397186
min	"		"
25%	4.0	0.0	4.0
50%	4.0	0.0	5.0
75%	5.0	0.0	5.0
max	👍🏽👍🏽"	visando sempre o ...	seria mais coeren...

review_comment_title	review_comment_message	review_creation_date	review_answer_timestamp
12005	41083	95398	95377
3.1654349958806 96E10	8.172413793103448	null	null
5.6254347509089 47E11	3.1175650470615843	null	null
		FOI A MINHA PRIM...	POIS NÃO CUMPREM...
5.0	8.0	null	null
10.0	10.0	null	null
10.0	10.0	null	null
🔟	😡😡😡😡😡👎👎👎👎👎	veio bem embalada...	70 + R$15

ANÁLISE

summary	product_id	product_category_name	product_name_lenght	product_description_lenght
count	32951	32341	32341	32341
mean	null	null	48.476948764725 89	771.4952846232337
stddev	null	null	10.245740725237287	635.1152246349538
min	00066f42aeeb9f300...	agro_industria_e_...	10	100
25%	null	null	42.0	339.0
50%	null	null	51.0	595.0
75%	null	null	57.0	972.0
max	fffe9eeff12fcbd74...	utilidades_domest...	9	999

ANÁLISE

product_photos_qty	product_weight_g	product_length_cm	product_height_cm	product_width_cm
32341	32949	32949	32949	32949
2.1889861166939797	2276.4724877841513	30.815077847582 63	16.937661234028347	23.196728277034204
1.736765637931 5435	4282.038730977024	16.914458054065953	13.637554061749569	12.079047453227794
1	0	10	10	10
1.0	300.0	18.0	8.0	15.0
1.0	700.0	25.0	13.0	20.0
3.0	1900.0	38.0	21.0	30.0
9	998	99	99	98

TRATAMENTO

```
[ ]  #RENOMENADO AS COLUNAS
     df1= (df1
             .withColumnRenamed('customer_id','id_cliente')
             .withColumnRenamed('customer_unique_id','unico_id_cliente')
             .withColumnRenamed('customer_zip_code_prefix','prefixo_codigo_postal_cliente')
             .withColumnRenamed('customer_city','cidade_cliente')
             .withColumnRenamed('customer_state','estado_cliente')
     )
```

```
[ ]  df2= (df2
             .withColumnRenamed('geolocation_zip_code_prefix','prefixo_codigo_postal_geolocalização')
             .withColumnRenamed('geolocation_lat','geolocalizacao_lat')
             .withColumnRenamed('geolocation_lng','geolocalizacao_lng')
             .withColumnRenamed('geolocation_city','geolocalizacao_cidade')
             .withColumnRenamed('geolocation_state','geolocalizacao_estado')
     )
```

TRATAMENTO

```
[ ]  df2 = df2.drop(F.col('prefixo_codigo_postal_geolocalização'))
     df2 = df2.drop(F.col('geolocalizacao_lat'))
     df2 = df2.drop(F.col('geolocalizacao_lng'))
```

```
[ ]  df1 = df1.drop(F.col('unico_id_cliente'))
     df1 = df1.drop(F.col('prefixo_codigo_postal_cliente'))
```

TRATAMENTO

```
[ ]  #Removendo os duplicados campo ID
     df1.select(F.count('id_cliente')).show()
```

```
[ ]  df3 = ( \
                df3.drop(F.col('data_limite_envio'))\
                .drop(F.col('data_limite_envio'))
                )
```

TRATAMENTO

payment_type
credit_card
credit_card
credit_card
credit_card
credit_card
credit_card
credit_card
credit_card
credit_card
boleto
credit_card
credit_card
credit_card
credit_card
credit_card
credit_card
credit_card
credit_card
credit_card
credit_card

TRATAMENTO

```
df4 = (df4.withColumn('tipo_de_pagamento', F.regexp_replace('tipo_de_pagamento','credit_card','cartao_credito')
df4 = (df4.withColumn('tipo_de_pagamento', F.regexp_replace('tipo_de_pagamento','voucher','cupom')))
df4 = (df4.withColumn('tipo_de_pagamento', F.regexp_replace('tipo_de_pagamento','debit_card','cartao_debito')))
```

TRATAMENTO

id_pedido	count
null	2236
2018-05-15 00:00:00	16
2017-12-19 00:00:00	16
2017-12-16 00:00:00	15
2018-02-24 00:00:00	15
2018-04-11 00:00:00	14
2018-08-18 00:00:00	14
2017-12-06 00:00:00	14
2018-08-28 00:00:00	13
2018-08-17 00:00:00	12
2018-03-30 00:00:00	12
2018-06-21 00:00:00	11
2018-04-13 00:00:00	11
2018-01-27 00:00:00	11
2018-03-21 00:00:00	11
2018-08-10 00:00:00	11
2018-08-04 00:00:00	11
2017-12-13 00:00:00	11
2018-03-02 00:00:00	11
2018-05-23 00:00:00	11

only showing top 20 rows

TRATAMENTO

```
df5 = df5.withColumn("id_pedido",F.when(F.col("id_pedido").rlike("2018"),F.lit("nao_informado")).otherwise(F.col("id_pedido")))
df5 = df5.withColumn("id_pedido",F.when(F.col("id_pedido").rlike("2017"),F.lit("nao_informado")).otherwise(F.col("id_pedido")))
```

```
df5 = df5.withColumn("id_pedido",F.when(F.col("id_pedido").rlike(" "),F.lit("nao_informado")).otherwise(F.col("id_pedido")))
```

```
df5 = df5.withColumn("id_pedido",F.when(F.col("id_pedido").rlike(" "),F.lit("nao_informado")).otherwise(F.col("id_pedido")))
```

UNIFICAÇÃO DOS DATASETS

```
[ ]  # df1 > df6 : id_cliente
     df_final = df1.join(df6, on = ['id_cliente'], how = 'inner')
```

```
[ ]  # df3 > df8 : id_vendedor
     df_final2 = df3.join(df8, on = ['id_vendedor'], how = 'inner')
```

```
[ ]  # df4 > df5 : id_pedido
     df_final3 = df4.join(df5, on =['id_pedido'], how = 'inner')
```

```
[ ]  #Usando df_final4 para agrupar todos os 3 dataframes finais ao dataframe da olist completo
     df_final4 = df_final.join(df_final2, on=['id_pedido'])
```

```
[ ]  # recebendo todos os dados em dataframe "DF_OLIST"
     df_olist = df_final4.join(df_final3, on=['id_pedido'], how = 'inner')
```

```
[ ]  #Verificando dataframe final
     df_olist.dtypes
```

PANDAS

```
[ ]  # Converta o dataframe do PySpark para o Pandas
     #df_olist_pd = df_final.toPandas()

     df_olist_pd = df_olist.toPandas()
     # Exiba o dataframe do Pandas
     df_olist_pd.head(5)
```

PANDAS

```
def whitespace_remover(dataframe):
    # percorrendo cada coluna
    for i in dataframe.columns:
        # checando o tipo de cada coluna
        if dataframe[i].dtype == 'object':
            # aplicando o metodo strip
            dataframe[i] = dataframe[i].map(str.strip)
        else:
            # caso coluna diferente de object não fazer nada.
            pass

whitespace_remover(df_olist_pd)
```

DATAFRAME FINAL

```
df_olist_pd.shape
```

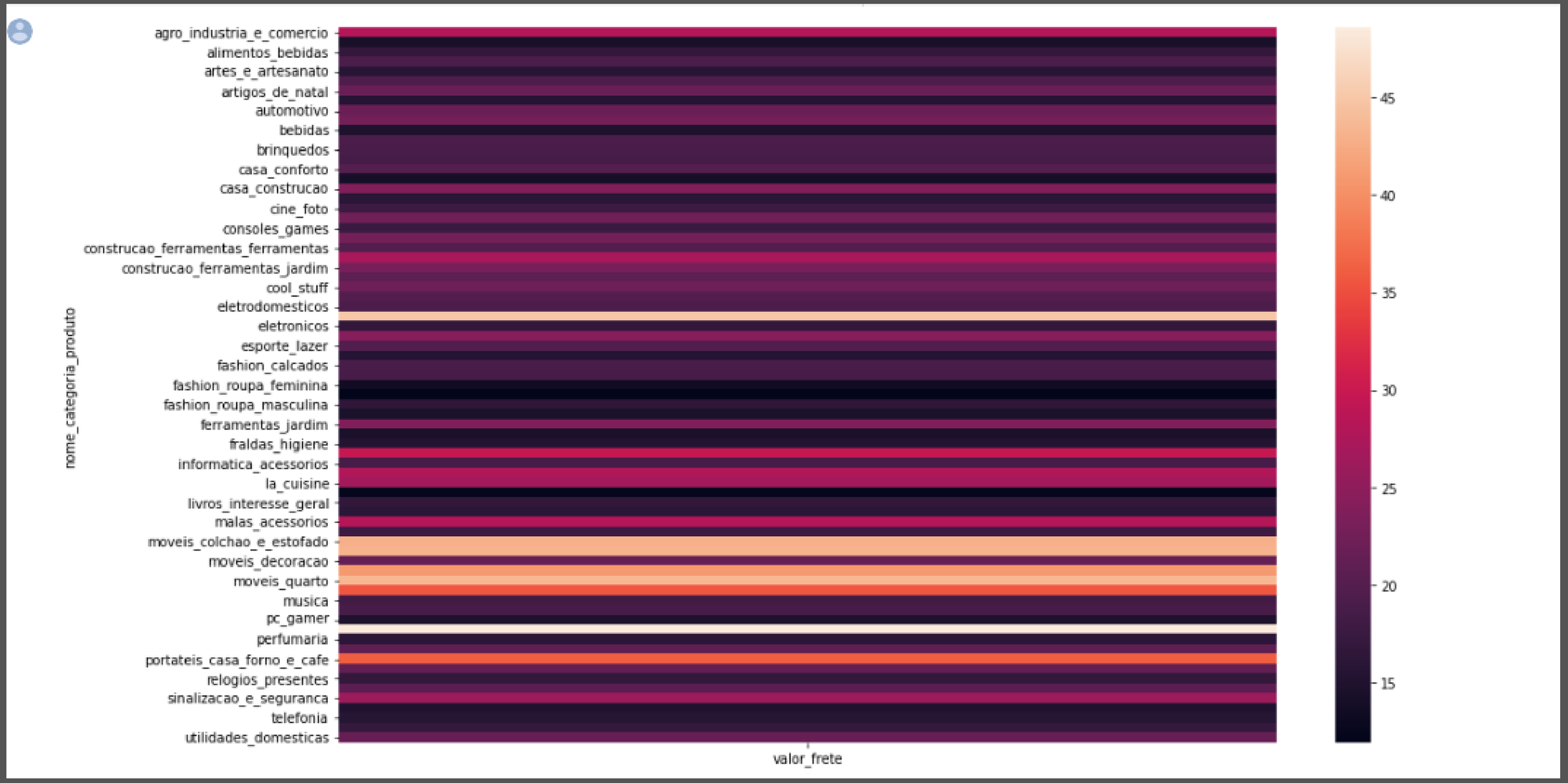
```
(97824, 29)
```

PLOTAGEM

PLOTAGEM

```
###Analisando nome da categoria com valor de frete
#
plt.figure(figsize = (16,10))
sns.heatmap(df_olist_pd.pivot_table(index=['nome_categoria_produto'],values=['valor_frete'],aggfunc='mean'));
```

PLOTAGEM

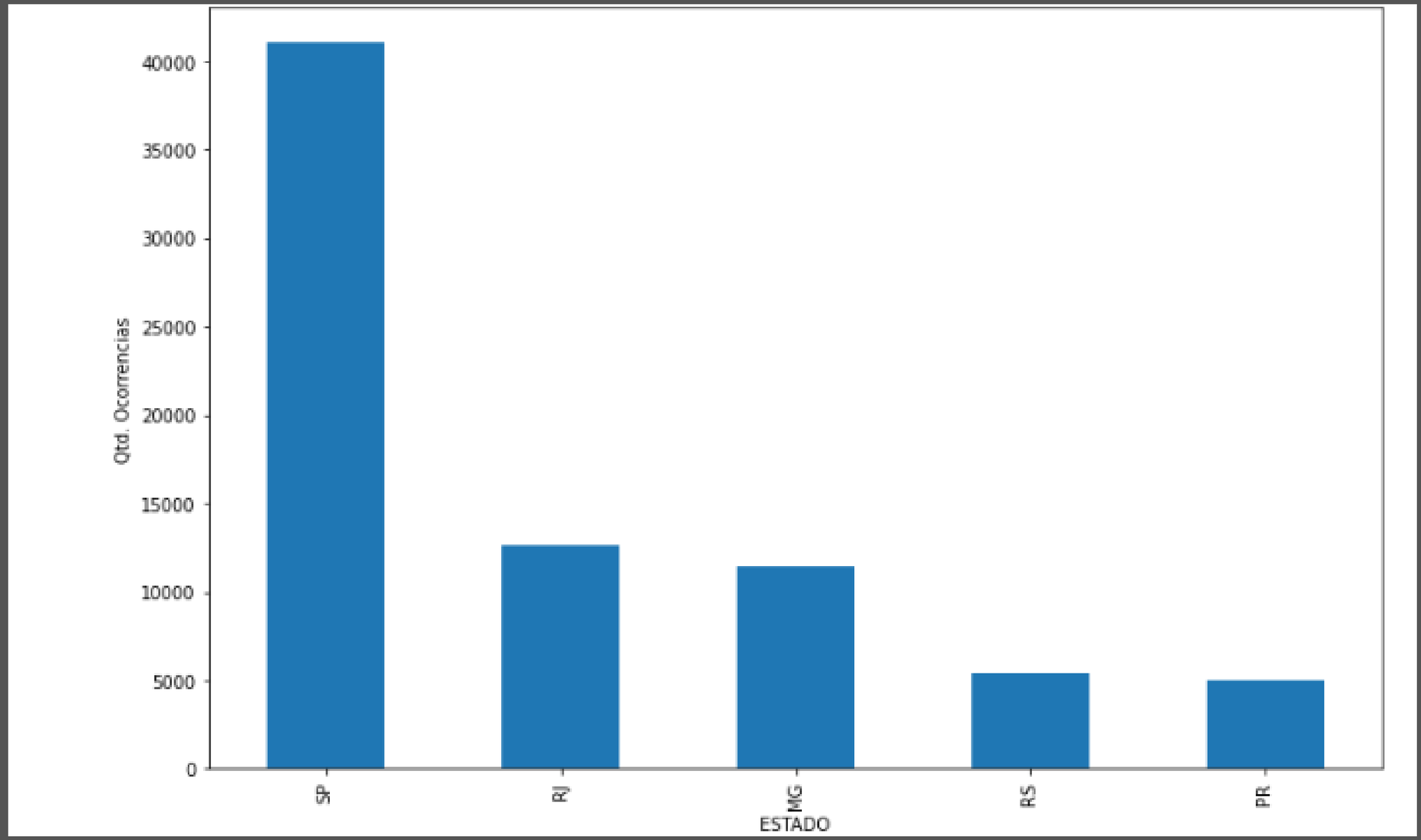

agro_industria_e_comercio
alimentos_bebidas
artes_e_artesanato
artigos_de_natal
automotivo
bebidas
brinquedos
casa_conforto
casa_construcao
cine_foto
consoles_games
construcao_ferramentas_ferramentas
construcao_ferramentas_jardim
cool_stuff
eletrodomesticos
eletronicos
esporte_lazer
fashion_calcados
fashion_roupa_feminina
fashion_roupa_masculina
ferramentas_jardim
fraldas_higiene
informatica_acessorios
la_cuisine
livros_interesse_geral
malas_acessorios
moveis_colchao_e_estofado
moveis_decoracao
moveis_quarto
musica
pc_gamer
perfumaria
portateis_casa_forno_e_cafe
relogios_presentes
sinalizacao_e_seguranca
telefonia
utilidades_domesticas
nome_categoria_produto
valor_frete
45
40
35
30
25
20
15

PLOTAGEM

```
#Plotagem grárfico de barra
df_olist_pd.groupby(['estado_cliente'],dropna=False).size().sort_values(ascending=False).head(5).plot.bar(figsize=(12,8),xlabel='ESTADO',ylabel='Qtd. Ocorrencias')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0c04d81370>
```

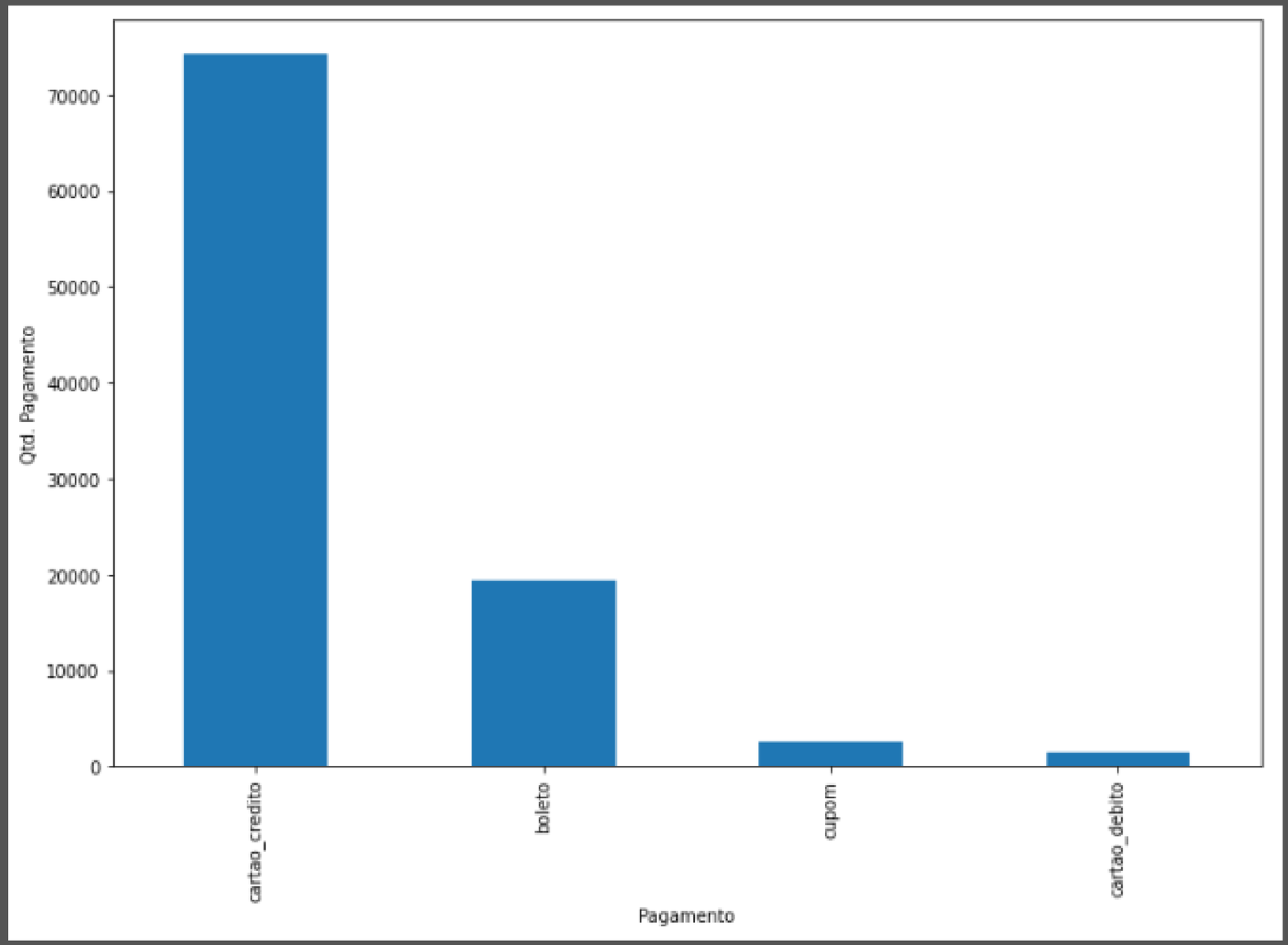
PLOTAGEM


40000
35000
30000
25000
20000
15000
10000
5000
0
Qtd. Ocorrencias
SP
RJ
MG
RS
PR
ESTADO

PLOTAGEM

```
#Plotagem | Verificando meios de pagamento mais usado
df_olist_pd.groupby(['tipo_de_pagamento'],dropna=False).size().sort_values(ascending=False).head(5).plot.bar(figsize=(12,8),xlabel='Pagamento',ylabel='Qtd. Pagamento')
```

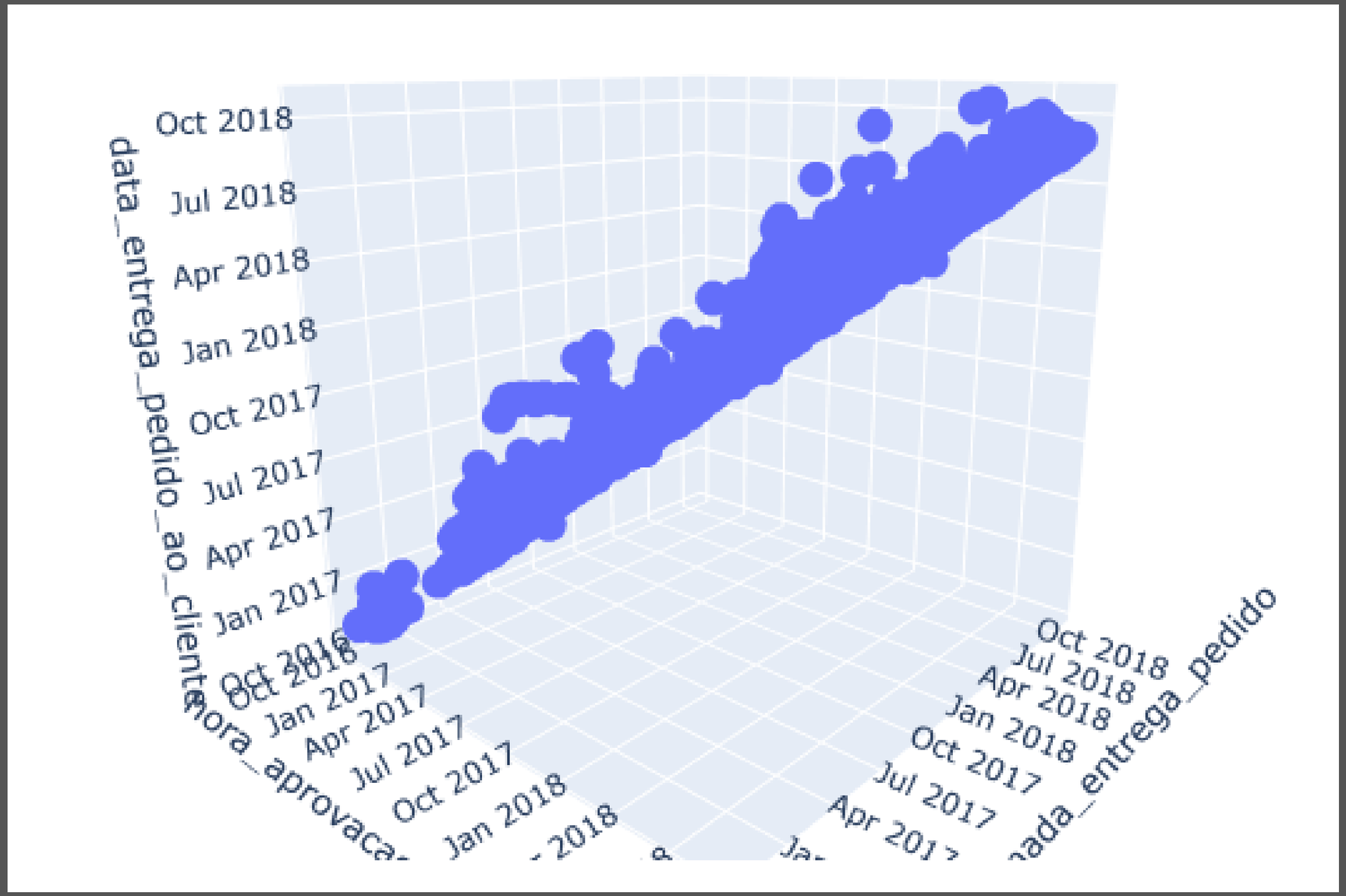
PLOTAGEM


70000
60000
50000
40000
30000
20000
10000
0
Qtd. Pagamento
cartao_credito
boleto
cupom
cartao_debito
Pagamento

PLOTAGEM

```
#Verificando data estimada com data de entrega
fig = px.scatter_3d(df_olist_pd, x='hora_aprovacao_pedido', y='data_estimada_entrega_pedido', z='data_entrega_pedido_ao_cliente')
fig.show()
```

PLOTAGEM


Oct 2018
Jul 2018
Apr 2018
Jan 2018
Oct 2017
Jul 2017
Apr 2017
Jan 2017
data_entrega_pedido_ao_cliente
data_aprovacao
data_entregada_entrega_pedido

PLOTAGEM

```
#Verificando coluna status_pedido

df_olist_pd.groupby('status_pedido')['status_pedido'].count().apply(lambda x: x / len(df_olist_pd) * 100).round(2).astype(str) + '%'
```

PLOTAGEM

```
status_pedido
aprovado                0.0%
cancelado              0.45%
em_processamento        0.3%
entregue               97.87%
enviado                 1.05%
faturado                0.31%
indisponivel            0.01%
Name: status_pedido, dtype: object
```

PLOTAGEM

```
#Iniciando o tratamento da coluna para representar na plotagem
words = ' '.join((i for i in df_olist_pd.nome_categoria_produto.dropna
#Setando as configs da nuvem de plotagem
wc = WordCloud( width=800, height=400, background_color='black').gener
#Plotando os dados
plt.figure(figsize=(16,8))
plt.axis("off")
plt.grid(False)
plt.imshow(wc);
```

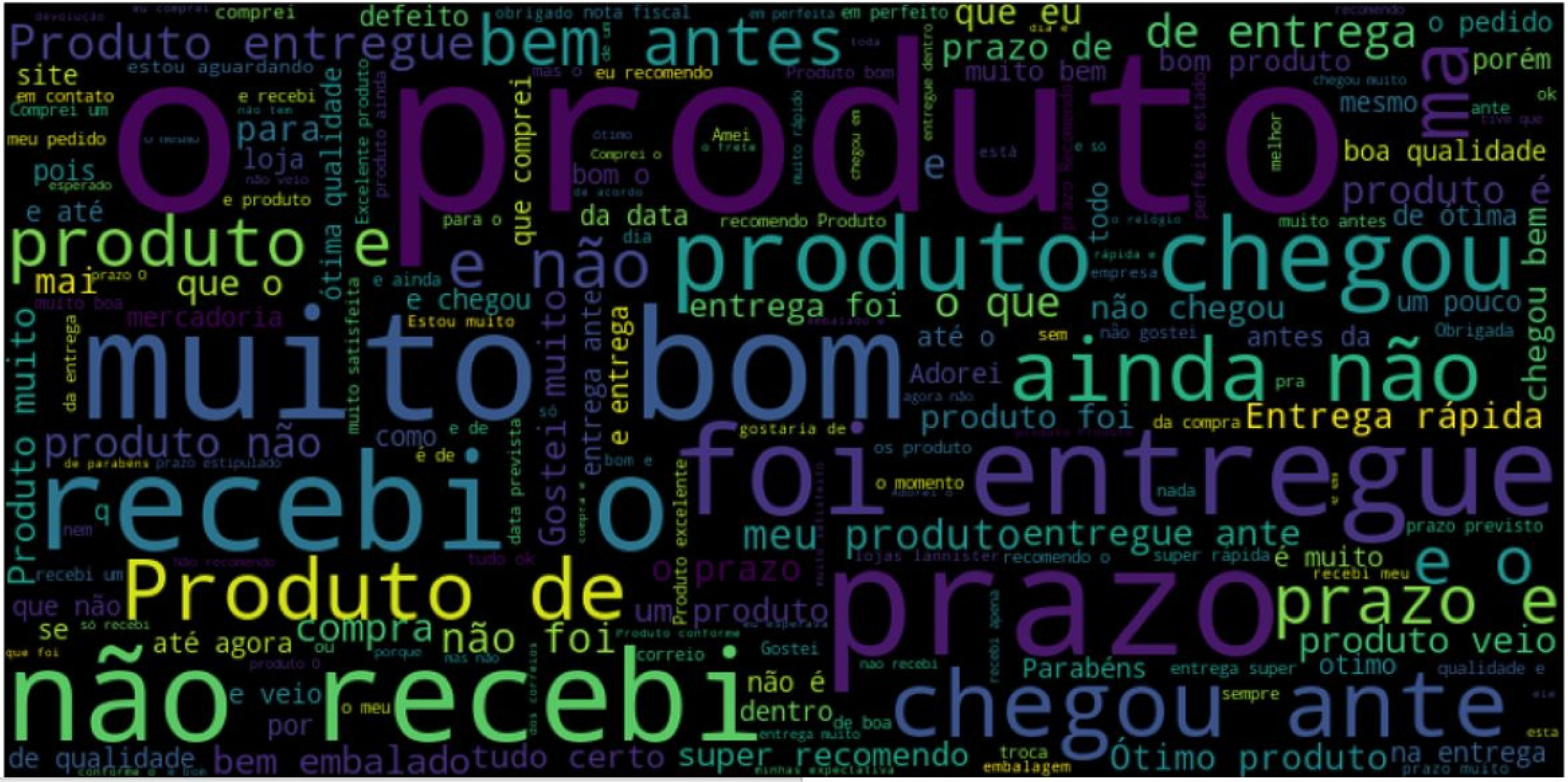
PLOTAGEM


utilidades_domesticas
consoles_games
livros_interesse_geral
fashion_roupa_masculina
construcao_ferramentas_jardim
pet_shop
cama_mesa_banho
artes
moveis_escritorio
fashion_calcados
construcao_ferramentas_construcao
relogios_presentes
eletrodomesticos
telefonia
perfumaria
esporte_lazer
alimentos
audio
ferramentas_jardim
livros_tecnicos
construcao_ferramentas_iluminacao
papelaria
moveis_decoracao
industria_comercio_e_negocios
malas_acessorios
eletrodomesticos_2
automotivo
cool_stuff
moveis_sala
climatizacao
bebes
brinquedos
beleza_saude
eletroportateis
casa_conforto
eletronicos
construcao_ferramentas_ferramentas
instrumentos_musicais
moveis_cozinha_area_de_servico_jantar_e_jardim
pcs
artigos_de_natal
sinalizacao_e_seguranca
fashion_underwear_e_moda_praia
cine_foto
informatica_acessorios
fashion_bolsas_e_acessorios
agro_industria_e_comercio
moveis_quarto
casa_construcao

PLOTAGEM

```
#Iniciando o tratamento da coluna para representar na plotagem
words = ' '.join((i for i in df_olist_pd.mensagem_da_avaliacao.dropna().str.replace(';'
#Setando as configs da nuvem de plotagem
wc = WordCloud( width=800, height=400, background_color='black').generate(words)
#Plotando os dados
plt.figure(figsize=(16,8))
plt.axis("off")
plt.grid(False)
plt.imshow(wc);
```

PLOTAGEM


o produto
muito bom
foi entregue
prazo
recebi o
não recebi
produto chegou
ainda não
Produto de
chegou ante
Produto entregue
bem antes
de entrega
produto e
e não
prazo e
produto veio
meu produtoentregue ante
Entrega rápida
produto não
bem embalado
tudo certo
super recomendo
Ótimo produto
na entrega
de qualidade
não foi
compra
um produto
o prazo
entrega foi
o que
não chegou
boa qualidade
produto é
de ótima
um pouco
antes da
produto foi
Adorei
Parabéns
otimo
dentro
não é
até agora
que não
e veio
por
que eu
prazo de
muito bem
bom produto
o pedido
porém
mesmo
site
para
loja
pois
e até
mai
que o
mercadoria
e chegou
da data
defeito
comprei
estou aguardando
eu recomendo
Gostei muito
ótima qualidade
que comprei
entrega ante
e entrega
Produto muito
muito satisfeita
Excelente produto
Produto excelente
data prevista
chegou bem
como
nada
prazo previsto
super rápida
é muito
recomendo o
correio
Gostei
e o
de boa
troca
embalagem
sempre
entrega super
qualidade e
Obrigada
pra
sem
até o
todo
melhor
Amei
recomendo Produto
gostaria de
o momento
os produto
meu pedido
em contato
Comprei um
e recebi
e produto
para o
dia
Estou muito
e ainda
se
recebi um
que foi
nem
ou
porque
o meu
da compra
muito antes
ok
ante
obrigado nota fiscal
em perfeito

MONGODB

```
[ ]  dbt = client['tratado']
     colecaot = dbt['base_tratada']
```

```
[ ]  # inserção do DF pandas com o dataframe tratado, para o mongoDB, em foma de dicionário,
     # assim como foi feito o upload dos dataframes com os dados originais
     # o DF estava grande demais para ser enviado para o Mongo então será enviado em pedaços ("chunks")
     chunk_size = 1000
     # definido o tamanho do chunk, calcula-se a quantidade de chunks
     num_chunks = df_olist_pd.shape[0] // chunk_size + 1

     # depois itera sobre os chunks
     for i in range(num_chunks):
         # e cria um .loc em cima dele
         chunk = df_olist_pd.loc[i*chunk_size:(i+1)*chunk_size-1]

         # aqui converte-se cada chunk para uma lista de dicionários
         chunk_dict = chunk.to_dict('records')

         # e aqui eles são inseridos no MongoDB
         colecaot.insert_many(chunk_dict)
```

MONGODB

```
[ ]  #montar o banco de dados para a base tratada
     db = client['BaseTratada']
```

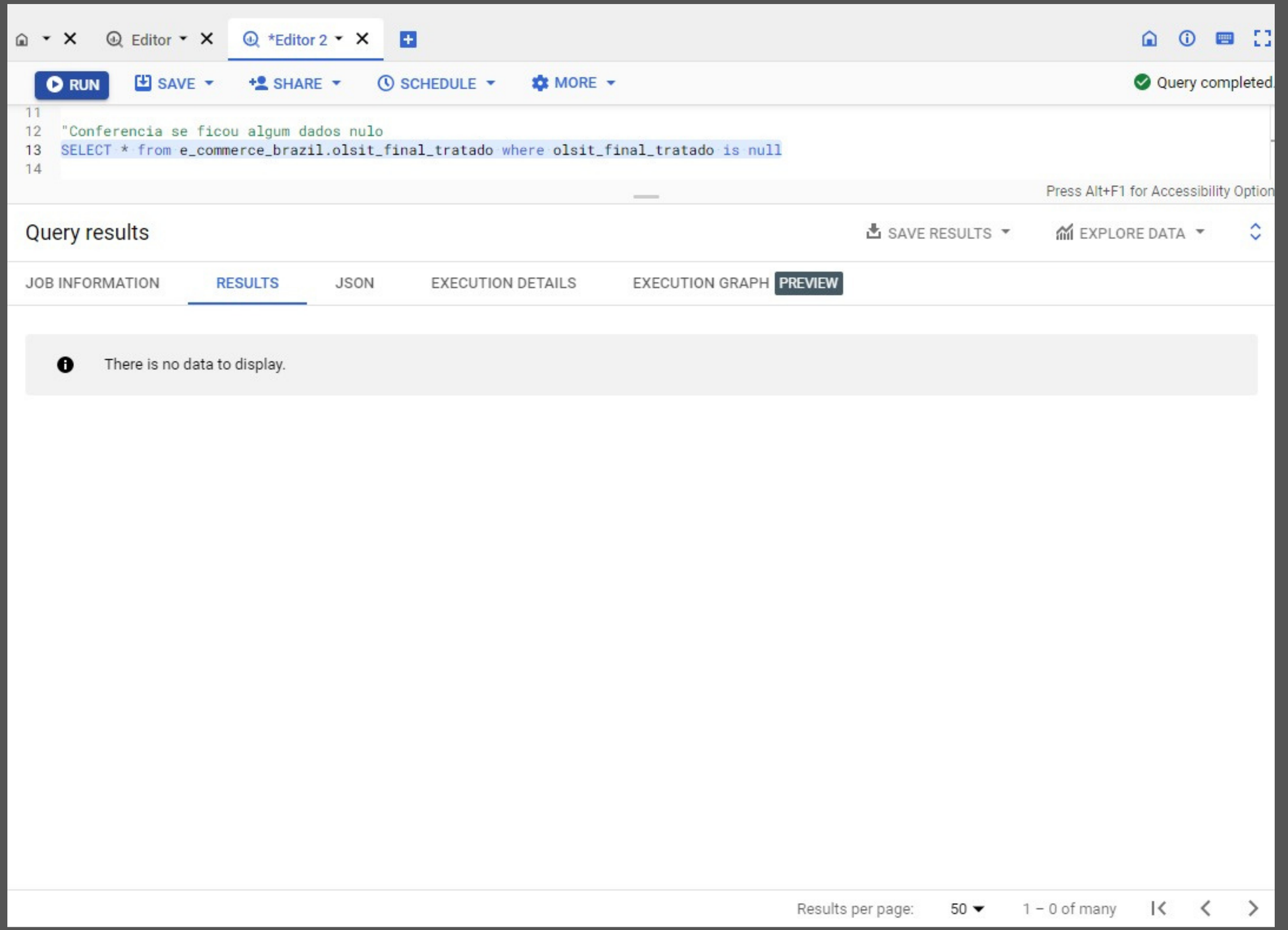
```
[ ]  #criando a coleção para a base tratada
     col_df_olist = db['olist_2016_2018_tratado']
     col_df_olist.count_documents({})
```

```
[ ]  #Gerando arquivo csv e fazendo o upload para GSC
     df_olist_pd.to_csv('olsit_final_tratado',index=False)
```

```
[ ]  df_olist_pd.head(10)
```

BIGQUERY

BIGQUERY


Editor
*Editor 2
RUN
SAVE
SHARE
SCHEDULE
MORE
Query completed
"Conferencia se ficou algum dados nulo
SELECT * from e_commerce_brazil.olsit_final_tratado where olsit_final_tratado is null
Press Alt+F1 for Accessibility Option
Query results
SAVE RESULTS
EXPLORE DATA
JOB INFORMATION
RESULTS
JSON
EXECUTION DETAILS
EXECUTION GRAPH
PREVIEW
There is no data to display.
Results per page:
50
1 – 0 of many

BIGQUERY

```
17  "Mudança dos tipos dos dados para consulta"
18  SELECT CAST('2021-19-20 12:52:01' AS string) AS data_hora_compra_pedido
19  SELECT CAST('2021-19-20 12:52:01' AS string) AS data_estimada_entrega_pedido
20
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	data_hora_compra_pedido
1	2021-19-20 12:52:01

BIGQUERY

```
6  "seleção de vendas dos 27 estados brasileiros"
7  SELECT estado_cliente as estado, COUNT(*) as qtde_por_estado FROM `it-data-search.e_commerce_brazil.olsit_final_tratado` group by  estado
   LIMIT 27
8
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	estado	qtde_por_estado
1	MG	11460
2	PR	4967
3	ES	1994
4	PE	1631
5	RO	246
6	SP	41063
7	RS	5411
8	RJ	12587
9	SC	3579
10	BA	3316
11	MT	895
12	TO	277
13	PA	955
14	CE	1316
15	PI	487
16	DF	2114
17	SE	344
18	GO	1991

Results per page: 50 | 1 – 27 of 27

BIGQUERY

```
15  "Consulta estimada de prazos"
16  SELECT data_hora_compra_pedido, data_entrega_pedido_ao_cliente, data_estimada_entrega_pedido FROM e_commerce_brazil.olsit_final_tratado
17
```

Press Alt+F1 for Accessibility Options

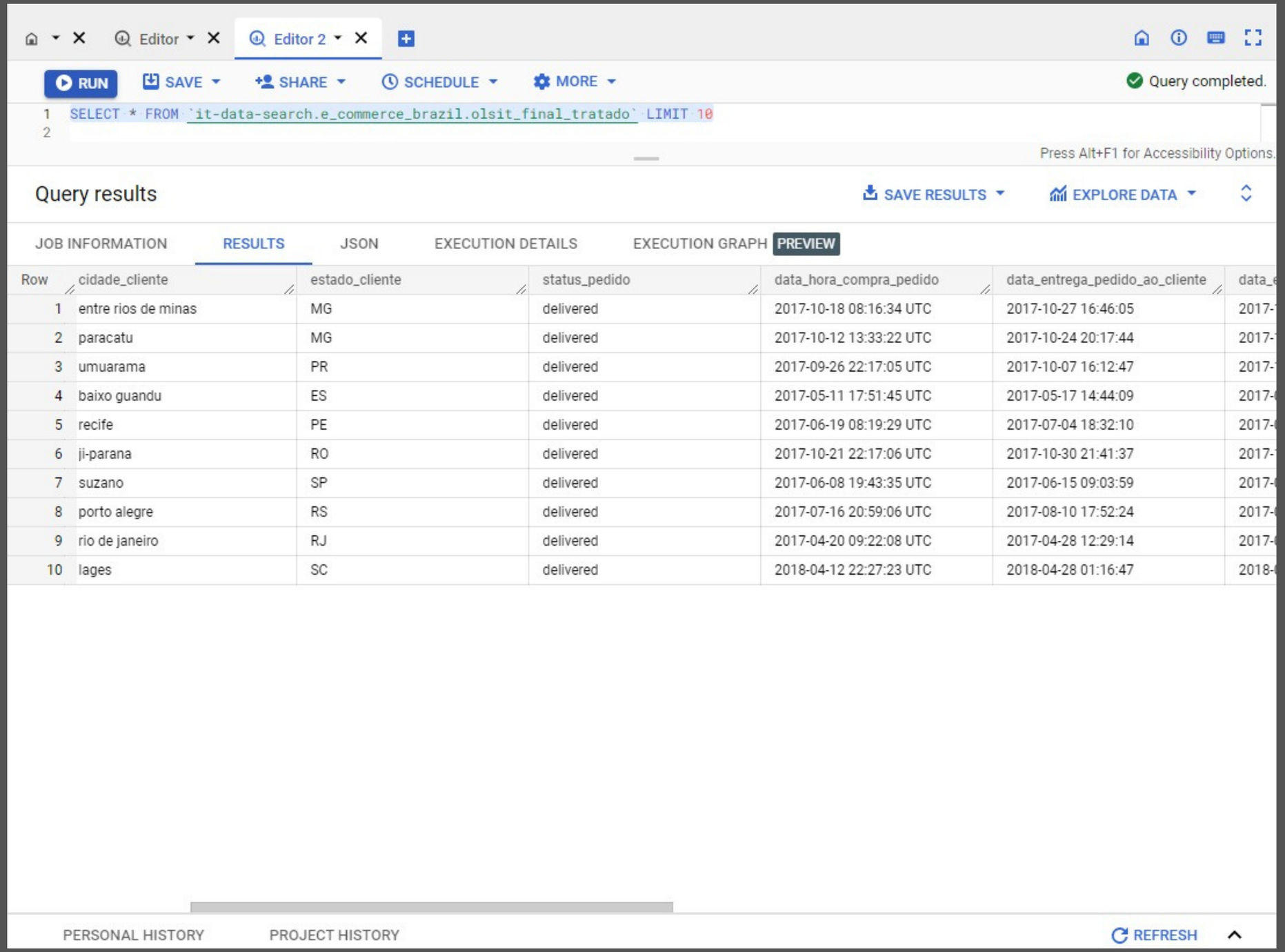
Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

Row	data_hora_compra_pedido	data_entrega_pedido_ao_cliente	data_estimada_e
1	2017-10-18 08:16:34 UTC	2017-10-27 16:46:05	2017-11-09
2	2017-10-12 13:33:22 UTC	2017-10-24 20:17:44	2017-11-06
3	2017-09-26 22:17:05 UTC	2017-10-07 16:12:47	2017-10-30
4	2017-05-11 17:51:45 UTC	2017-05-17 14:44:09	2017-06-20
5	2017-06-19 08:19:29 UTC	2017-07-04 18:32:10	2017-07-19
6	2017-10-21 22:17:06 UTC	2017-10-30 21:41:37	2017-11-29
7	2017-06-08 19:43:35 UTC	2017-06-15 09:03:59	2017-07-10
8	2017-07-16 20:59:06 UTC	2017-08-10 17:52:24	2017-08-16
9	2017-04-20 09:22:08 UTC	2017-04-28 12:29:14	2017-05-12
10	2018-04-12 22:27:23 UTC	2018-04-28 01:16:47	2018-05-08
11	2017-05-24 08:57:47 UTC	2017-05-31 13:05:28	2017-06-22
12	2017-06-01 19:58:20 UTC	2017-06-16 13:43:07	2017-06-30
13	2018-03-29 20:53:48 UTC	2018-04-24 16:10:44	2018-04-24
14	2018-03-18 19:26:01 UTC	2018-04-11 20:47:49	2018-04-16
15	2017-05-16 16:14:09 UTC	2017-05-22 14:46:40	2017-06-07
16	2017-06-03 00:29:40 UTC	2017-06-08 11:28:28	2017-06-28
17	2017-05-14 18:35:12 UTC	2017-05-23 12:45:29	2017-06-02
18	2017-06-06 23:08:04 UTC	2017-06-14 14:42:13	2017-06-29
19	2017-11-25 10:50:33 UTC	2017-12-14 15:12:40	2017-12-22

BIGQUERY



Row	cidade_cliente	estado_cliente	status_pedido	data_hora_compra_pedido	data_entrega_pedido_ao_cliente	data_e
1	entre rios de minas	MG	delivered	2017-10-18 08:16:34 UTC	2017-10-27 16:46:05	2017-
2	paracatu	MG	delivered	2017-10-12 13:33:22 UTC	2017-10-24 20:17:44	2017-
3	umuarama	PR	delivered	2017-09-26 22:17:05 UTC	2017-10-07 16:12:47	2017-
4	baixo guandu	ES	delivered	2017-05-11 17:51:45 UTC	2017-05-17 14:44:09	2017-
5	recife	PE	delivered	2017-06-19 08:19:29 UTC	2017-07-04 18:32:10	2017-
6	ji-parana	RO	delivered	2017-10-21 22:17:06 UTC	2017-10-30 21:41:37	2017-
7	suzano	SP	delivered	2017-06-08 19:43:35 UTC	2017-06-15 09:03:59	2017-
8	porto alegre	RS	delivered	2017-07-16 20:59:06 UTC	2017-08-10 17:52:24	2017-
9	rio de janeiro	RJ	delivered	2017-04-20 09:22:08 UTC	2017-04-28 12:29:14	2017-
10	lages	SC	delivered	2018-04-12 22:27:23 UTC	2018-04-28 01:16:47	2018-

CUSTOS DO PROJETO

CUSTOS DO PROJETO

Cloud Storage

1x Standard Storage

Location: Paris

Total Amount of Storage: 10 GiB	BRL 1.38
Always Free usage included: No	

BRL 1.38

BigQuery

olist-final-project

Location: Paris

Active Storage 5 GiB
Long-term Storage 0 GiB
Queries 3 TiB

BRL 74.94

CUSTOS DO PROJETO

CUSTO TOTAL: R$ 170,10

Cloud SQL for MySQL

ARMAZENAMENTO DE DADOS BRUTOS

# of instances: 1

Instance type: db-standard-1

Location: Paris

12.0 total hours per month

SSD Storage: 10.0 GiB

Backup: 0.0 GiB

BRL 17.45

Total Estimated Cost: BRL 93.78 per 1 month

Estimate Currency

BRL - Brazilian Real

DASHBOARD

SIQUEIRASNI

PHILIPPEMAGNO

ANDRE-LUÍS-ENGENHARIA-DE-DADOS

MARTAMARUBAYASHI

OSNIMUNIZ