**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS  
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**

**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

**Sandro José Lins Santos**

**Classificação da aprovação ou rejeição dos produtos vendidos no e-comerce associados a região e ao idh**

Belo Horizonte

2021

**Nome do(a) Autor(a)**

**TÍTULO DO PROJETO**

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Belo Horizonte

ano **SUMÁRIO**

[1. Introdução 4](#_Toc43156067)

[1.1. Contextualização 4](#_Toc43156068)

[1.2. O problema proposto 4](#_Toc43156069)

[2. Coleta de Dados 4](#_Toc43156070)

[3. Processamento/Tratamento de Dados 5](#_Toc43156071)

[4. Análise e Exploração dos Dados 5](#_Toc43156072)

[5. Criação de Modelos de Machine Learning 5](#_Toc43156073)

[6. Apresentação dos Resultados 5](#_Toc43156074)

[7. Links 6](#_Toc43156075)

[REFERÊNCIAS 7](#_Toc43156076)

[APÊNDICE 8](#_Toc43156077)

# 1. Introdução

## 1.1. Contextualização

O e-comerce é uma importante modalidade de vendas e representa uma forma de comerciantes conseguirem alcançar maiores consumidores sem precisarem possuir lojas físicas nos diversos pontos do país. E para que os produtos alcancem de maneira eficaz o público-alvo, um dos elementos de análise é o nível de satisfação dos consumidores. Não adianta comercializar um produto para uma região sendo que o preço, ou frete esteja acima do percebido como positivo por parte dos consumidores. Com base nisso os comerciantes poderão avaliar melhor suas vendas.

## 1.2. O problema proposto

O problema é importante para poder classificar, conforme as variações das variáveis aleatórias usadas no modelo, a venda vai ser aprovada ou rejeitada na opinião do comprador. Então o vendedor terá uma ferramenta de gestão capaz de ajudar na tomada de decisão sobre os preços que podem ser praticados em determinada região.

Os dados analisados são:

* Dataset publicado pela Olist Store, que é uma empresa de alavancagem de vendas na internet. Esta base se encontra no site da keggle
* Base é a base publicada pelo Atlas de Desenvolvimento Humano no Brasil. Que traz o censo de 1991 a 2010 dos municípios do Brasil.
* Base do IBGE que contém as siglas para os códigos dos estados.

Nesta analise vai ser estudado se para determinada configuração de variáveis, como preço, frete, município, idh, parcelamento, afetam a nota de satisfação do consumidor.

# 2. Coleta de Dados

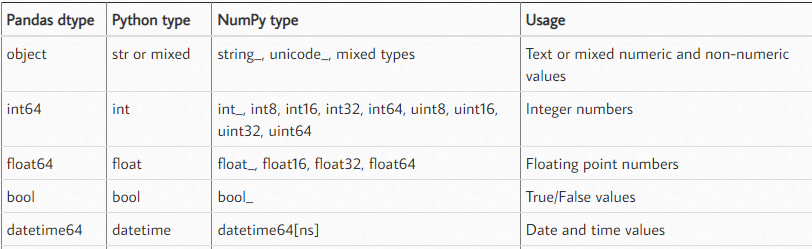
Nessa seção você deve deixar claro onde obteve os dados, o formato e estrutura dos datasets, o relacionamento entre os datasets utilizados, etc. Caso os dados tenham sido obtidos na internet, informe a data e o link em que os dados foram obtidos. Sugere-se que você crie uma tabela com a descrição de cada campo/coluna do seu dataset conforme o exemplo a seguir:

Linguagem de programação e bibliotecas utilizadas:

A linguagem de programação que foi utilizada neste projeto foi o Python.

O python possui a biblioteca pandas que transforma dados tabulares nos chamados DataFrames que lembram muito o excel no seu formato. Neste projeto converteremos os arquivso do formato csv para o formato de DataFrames utilizando a biblioteca pandas para poder manipular e extrair informações sobre as variáveis que constam dos arquivos e assim fazer os estudos necessários e montar o modelo de arvore de decisão.

Como será utilizado o dataframe, há uma particularidade para os tipos das variáveis. Estas têm nomes diferentes das do python. No site <https://pbpython.com/pandas_dtypes.html> mostra um quadro comparativo que colo aqui abaixo:



Ao listarmos os tipos das variáveis das colunas dos dataframes, teremos que observar esta tabela.

Base\_1

* nome: Brazilian E-Commerce Public Dataset by Olist
* site: <https://www.kaggle.com/olistbr/brazilian-ecommerce>

Base 2

* nome: Censo Demográfico (1991, 2000 e 2010)
* site: <http://www.atlasbrasil.org.br/acervo/biblioteca>

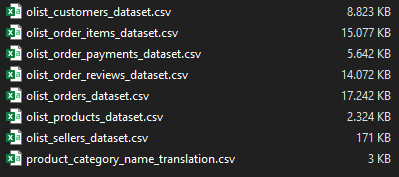
Base 3

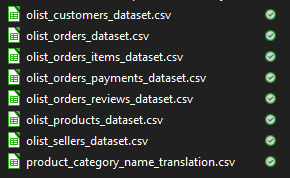
O arquivo fica em <https://www.ibge.gov.br/> -> estatisticas -> downloads -> Perfil\_Estados -> Base\_ETADIC\_2019.xlsx (<https://www.ibge.gov.br/estatisticas/downloads-estatisticas.html>)

Da Base 3 vai-se aproveitar a ultima aba de planilhas que contém os códigos de uf, as siglas dos estados e a região a qual pertencem.

A Base 1 possui 8 arquivos no formato csv conforme listados na figura abaixo:

Para evitar digitação errada ao longo do script, foi alterado o termo “order” para “orders”.



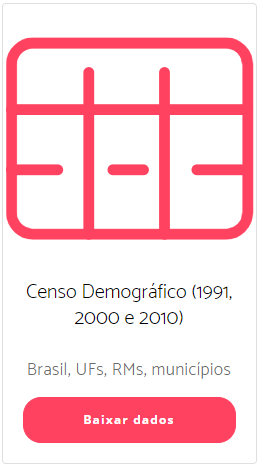


O arquivo “product\_category\_name\_translation.csv” não será utilizado porque se trata apenas de um arquivo que traduz do português para o inglês a lista dos nomes dos produtos. E estes nomes já estão em português no arquivo “olist\_products\_dataset.csv”.

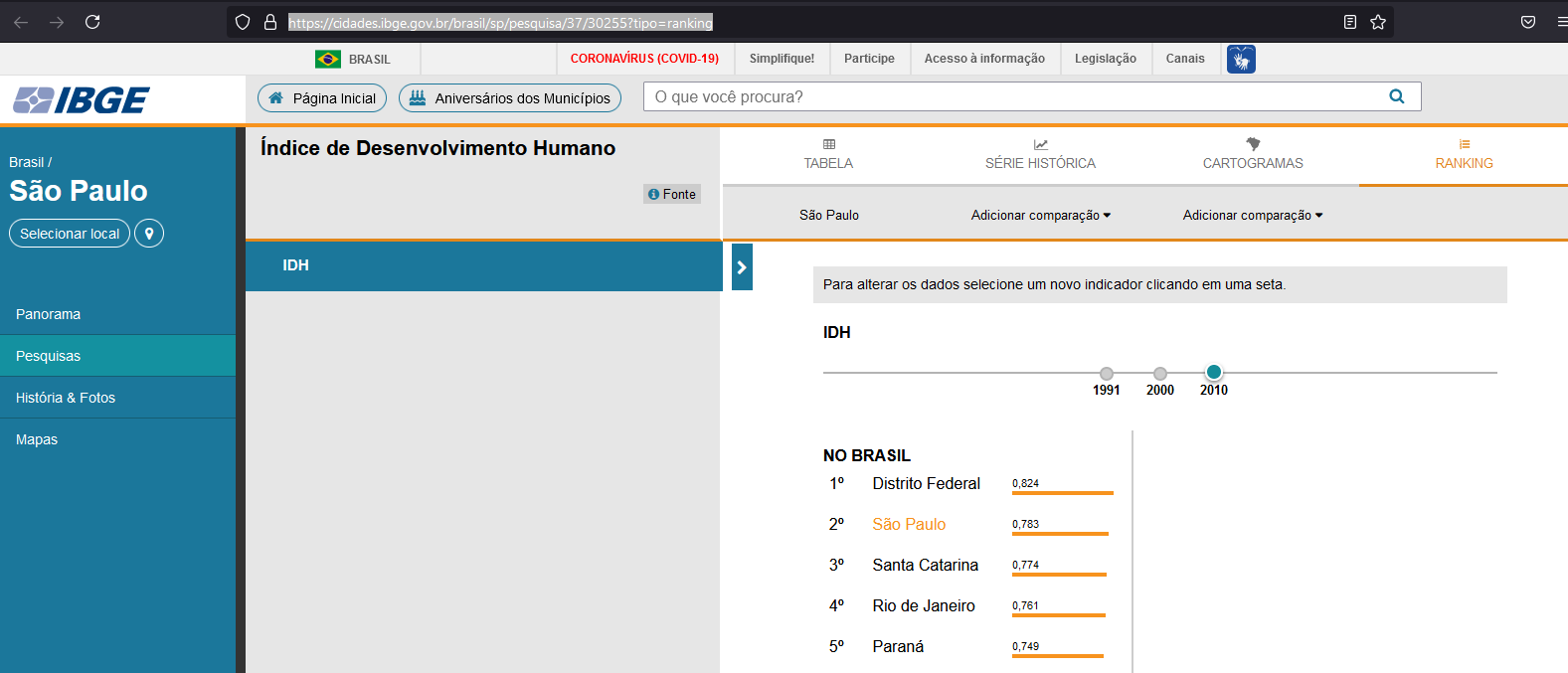
A Base 2 possui dois arquivos como segue abaixo:



Ambos os arquivos fazerm referencia à 2010 e este é o arquivo mais recente da base encontrado:



No site do IBGE, (<https://cidades.ibge.gov.br/brasil/sp/pesquisa/37/30255?tipo=ranking>), temos que o ano de referencia do IDH é de 1991, 2000 e 2010. Que coincide com os anos da pesquisa da Base 2.



Para a Base 1 temos os valores para os anos de 2017 e 2018.

Os arquivos possuem as seguintes colunas e formatos:

Base 1:



arquivo da base 1: olist\_customers\_dataset.csv

customer\_id object

customer\_unique\_id object

customer\_zip\_code\_prefix int64

customer\_city object

customer\_state object

arquivo da base 1: olist\_orders\_dataset.csv

order\_id object

customer\_id object

order\_status object

order\_purchase\_timestamp object

order\_approved\_at object

order\_delivered\_carrier\_date object

order\_delivered\_customer\_date object

order\_estimated\_delivery\_date object

arquivo da base 1: olist\_order\_items\_dataset.csv

order\_id object

order\_item\_id int64

product\_id object

seller\_id object

shipping\_limit\_date object

price float64

freight\_value float64

arquivo da base 1: olist\_products\_dataset.csv

product\_id object

product\_category\_name object

product\_name\_lenght float64

product\_description\_lenght float64

product\_photos\_qty float64

product\_weight\_g float64

product\_length\_cm float64

product\_height\_cm float64

product\_width\_cm float64

arquivo da base 1: olist\_order\_payments\_dataset.csv

order\_id object

payment\_sequential int64

payment\_type object

payment\_installments int64

payment\_value float64

arquivo da base 1: olist\_order\_reviews\_dataset.csv

review\_id object

order\_id object

review\_score int64

review\_comment\_title object

review\_comment\_message object

review\_creation\_date object

review\_answer\_timestamp object

arquivo da base 1: olist\_sellers\_dataset.csv

seller\_id object

seller\_zip\_code\_prefix int64

seller\_city object

seller\_state object

arquivo da base 1: product\_category\_name\_translation.csv

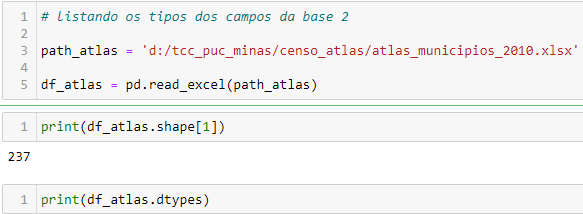
product\_category\_name object

product\_category\_name\_english object

Base 2:

A base completa contempla os anos de 1991, 2000 e 2010. Então foi copiado apenas o ano de 2010 para um arquivo separado para poder ler com mais velocidade

A base 2 possui 237 colunas



ANO int64

UF int64

Codmun6 int64

Codmun7 int64

Município object

ESPVIDA float64

FECTOT float64

MORT1 float64

MORT5 float64

RAZDEP float64

SOBRE40 float64

SOBRE60 float64

T\_ENV float64

E\_ANOSESTUDO float64

T\_ANALF11A14 float64

T\_ANALF15A17 float64

T\_ANALF15M float64

T\_ANALF18A24 float64

T\_ANALF18M float64

T\_ANALF25A29 float64

T\_ANALF25M float64

T\_ATRASO\_0\_BASICO float64

T\_ATRASO\_0\_FUND float64

T\_ATRASO\_0\_MED float64

T\_ATRASO\_1\_BASICO float64

T\_ATRASO\_1\_FUND float64

T\_ATRASO\_1\_MED float64

T\_ATRASO\_2\_BASICO float64

T\_ATRASO\_2\_FUND float64

T\_ATRASO\_2\_MED float64

T\_FBBAS float64

T\_FBFUND float64

T\_FBMED float64

T\_FBPRE float64

T\_FBSUPER float64

T\_FLBAS float64

T\_FLFUND float64

T\_FLMED float64

T\_FLPRE float64

T\_FLSUPER float64

T\_FREQ0A3 float64

T\_FREQ11A14 float64

T\_FREQ15A17 float64

T\_FREQ18A24 float64

T\_FREQ25A29 float64

T\_FREQ4A5 float64

T\_FREQ4A6 float64

T\_FREQ5A6 float64

T\_FREQ6 float64

T\_FREQ6A14 float64

T\_FREQ6A17 float64

T\_FREQFUND1517 float64

T\_FREQFUND1824 float64

T\_FREQFUND45 float64

T\_FREQMED1824 float64

T\_FREQMED614 float64

T\_FREQSUPER1517 float64

T\_FUND11A13 float64

T\_FUND12A14 float64

T\_FUND15A17 float64

T\_FUND16A18 float64

T\_FUND18A24 float64

T\_FUND18M float64

T\_FUND25M float64

T\_MED18A20 float64

T\_MED18A24 float64

T\_MED18M float64

T\_MED19A21 float64

T\_MED25M float64

T\_SUPER25M float64

CORTE1 float64

CORTE2 float64

CORTE3 float64

CORTE4 float64

CORTE9 float64

GINI float64

PIND float64

PINDCRI float64

PMPOB float64

PMPOBCRI float64

PPOB float64

PPOBCRI float64

PREN10RICOS float64

PREN20 float64

PREN20RICOS float64

PREN40 float64

PREN60 float64

PREN80 float64

PRENTRAB float64

R1040 float64

R2040 float64

RDPC float64

RDPC1 float64

RDPC10 float64

RDPC2 float64

RDPC3 float64

RDPC4 float64

RDPC5 float64

RDPCT float64

RIND float64

RMPOB float64

RPOB float64

THEIL float64

CPR float64

EMP float64

P\_AGRO float64

P\_COM float64

P\_CONSTR float64

P\_EXTR float64

P\_FORMAL float64

P\_FUND float64

P\_MED float64

P\_SERV float64

P\_SIUP float64

P\_SUPER float64

P\_TRANSF float64

REN0 float64

REN1 float64

REN2 float64

REN3 float64

REN5 float64

RENOCUP float64

T\_ATIV float64

T\_ATIV1014 float64

T\_ATIV1517 float64

T\_ATIV1824 float64

T\_ATIV18M float64

T\_ATIV2529 float64

T\_DES float64

T\_DES1014 float64

T\_DES1517 float64

T\_DES1824 float64

T\_DES18M float64

T\_DES2529 float64

THEILtrab float64

TRABCC float64

TRABPUB float64

TRABSC float64

T\_AGUA float64

T\_BANAGUA float64

T\_DENS float64

T\_LIXO float64

T\_LUZ float64

AGUA\_ESGOTO float64

PAREDE float64

T\_CRIFUNDIN\_TODOS float64

T\_FORA4A5 float64

T\_FORA6A14 float64

T\_FUNDIN\_TODOS float64

T\_FUNDIN\_TODOS\_MMEIO float64

T\_FUNDIN18MINF float64

T\_M10A14CF float64

T\_M15A17CF float64

T\_MULCHEFEFIF014 float64

T\_NESTUDA\_NTRAB\_MMEIO float64

T\_OCUPDESLOC\_1 float64

T\_RMAXIDOSO float64

T\_SLUZ float64

HOMEM0A4 int64

HOMEM10A14 int64

HOMEM15A19 int64

HOMEM20A24 int64

HOMEM25A29 int64

HOMEM30A34 int64

HOMEM35A39 int64

HOMEM40A44 int64

HOMEM45A49 int64

HOMEM50A54 int64

HOMEM55A59 int64

HOMEM5A9 int64

HOMEM60A64 int64

HOMEM65A69 int64

HOMEM70A74 int64

HOMEM75A79 int64

HOMEMTOT int64

HOMENS80 int64

MULH0A4 int64

MULH10A14 int64

MULH15A19 int64

MULH20A24 int64

MULH25A29 int64

MULH30A34 int64

MULH35A39 int64

MULH40A44 int64

MULH45A49 int64

MULH50A54 int64

MULH55A59 int64

MULH5A9 int64

MULH60A64 int64

MULH65A69 int64

MULH70A74 int64

MULH75A79 int64

MULHER80 int64

MULHERTOT int64

PEA int64

PEA1014 int64

PEA1517 int64

PEA18M int64

peso1 int64

PESO1114 int64

PESO1113 int64

PESO1214 int64

peso13 int64

PESO15 int64

peso1517 int64

PESO1524 int64

PESO1618 int64

PESO18 int64

Peso1820 int64

PESO1824 int64

Peso1921 int64

PESO25 int64

peso4 int64

peso5 int64

peso6 int64

PESO610 int64

Peso617 int64

PESO65 int64

PESOM1014 int64

PESOM1517 int64

PESOM15M int64

PESOM25M int64

pesoRUR int64

pesotot int64

pesourb int64

PIA int64

PIA1014 int64

PIA1517 int64

PIA18M int64

POP int64

POPT int64

I\_ESCOLARIDADE float64

I\_FREQ\_PROP float64

IDHM float64

IDHM\_E float64

IDHM\_L float64

IDHM\_R float64

Base 3

REGIAO object

COD UF int64

UF object

NOME UF object

# 3. Processamento/Tratamento de Dados

Cada arquivo possui um conjunto de campos (colunas) que serão transformados em um dataframe. Após esta transformação, haverá a fusão de um dataframe ao outro utilzando uma chave em comum da mesma maneira que se faz em um left join no sql.

A cada fusão será testado se houve linhas duplicadas e/ou linhas com elementos nulos. Os elementos duplicados e nulos serão tratados conforme analise do contexto e seu impacto no dataframe final.

Ao final da analise o resultado será um dataframe contendo as colunas de todos os outros dataframes.

O metodo disponivel no pandas para fazer a operação de concatenar um dataframe em outro, chama-se merge (<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html>)

Para a biblioteca pandas os elementos nulos ou faltantes são considereros como NaN (Not a Number) (<https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html>)

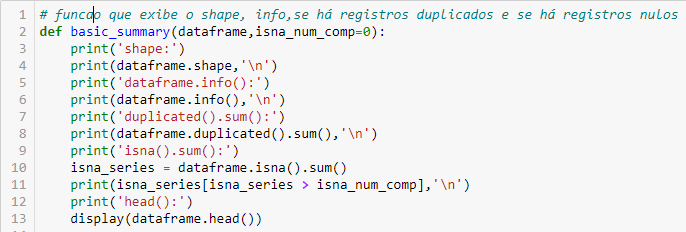
No pandas temos o comando dataframe.duplicated() que retorna uma series contendo todos os indices das linhas e ao lado False para não duplicado e True para duplicado. Como mostra na documentação (<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.duplicated.html>), pode-se mostrar o primeiro duplicado, o ultimo duplicado ou manter todos os duplicados.

Estes argumentos são importante no caso de querer tratar os casos duplicados em separado.

O comando dataframe.drop\_duplicates() (<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html>), retira as linhas duplicadas.

Como serão craidos muitos dataframes e depois os dataframes serão unidos uns aos outros criando novos dataframes, foi criada uma função que contém alguns métodos para poder fazer uma análise rápida do dataframe e com isso poder tomar algumas decisões sobre seus elementos e lidar com os casos nulos e/ou faltantes, e linhas duplicadas.

A função foi chamada de basic\_summary() como demonstrado abaixo:



Nesta função temos:

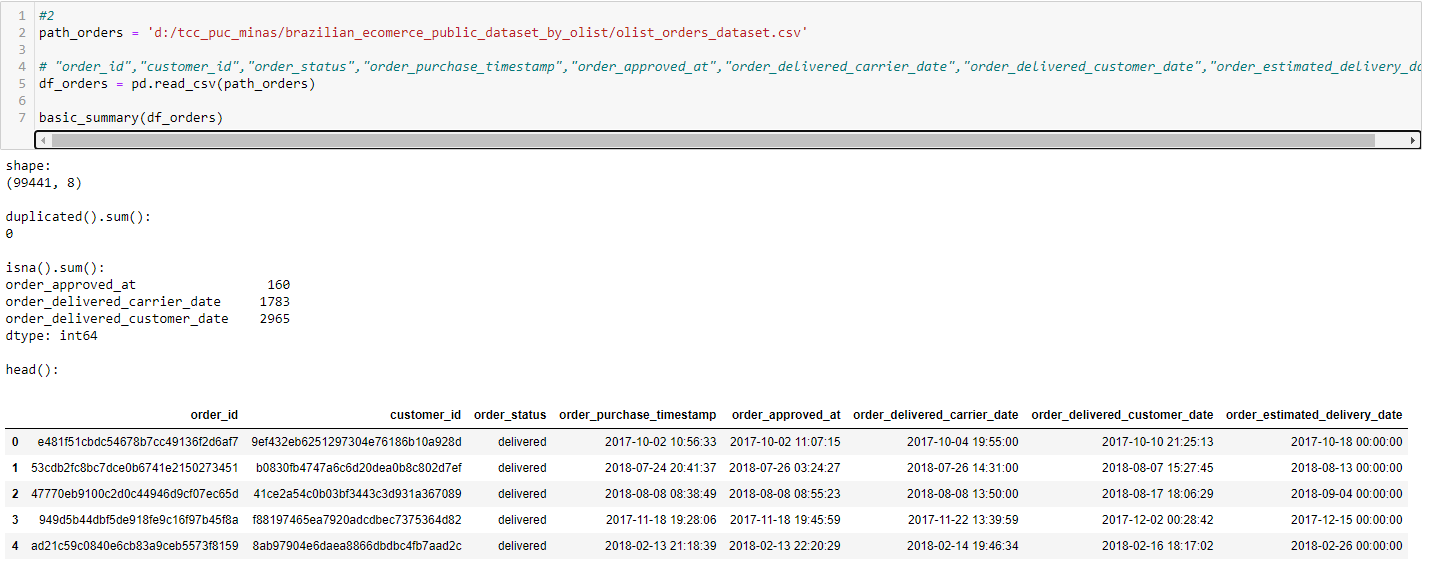
* print(dataframe.shape) – mostra o numero de linhas e colunas do dataframe
* print(dataframa.duplicated().sum()) – traz o somatório de linhas duplicadas no dataframe
* isna\_series = dataframe.isna().sum() – como mostrado acima, traz uma series contendo o nome das colunas e o somatório de valores nulos e/ou faltantes para estas. Neste caso o valor é armazenado na variável isna\_series.
* print(isna\_series[isna\_series > isna\_num\_comp]) – traz as colunas do dataframe que possuem as somas de ocorrências dos valores nulos ou faltantes maiores que o isna\_num\_comp. O valor default é zero. Então qualquer valor maior que zero sastifaz a inequação e será exibido apenas as colunas que possuem valores maiores que zero.
* display(dataframe.head()) – mostra as 5 primeiras linhas do dataframe

Depois da explicação dos métodos que serão usados, agora será demonstrada a análise de cada dataframe para a Base 1

* 1. - Análise de olist\_customers\_dataset.csv transformado no dataframe df\_customers



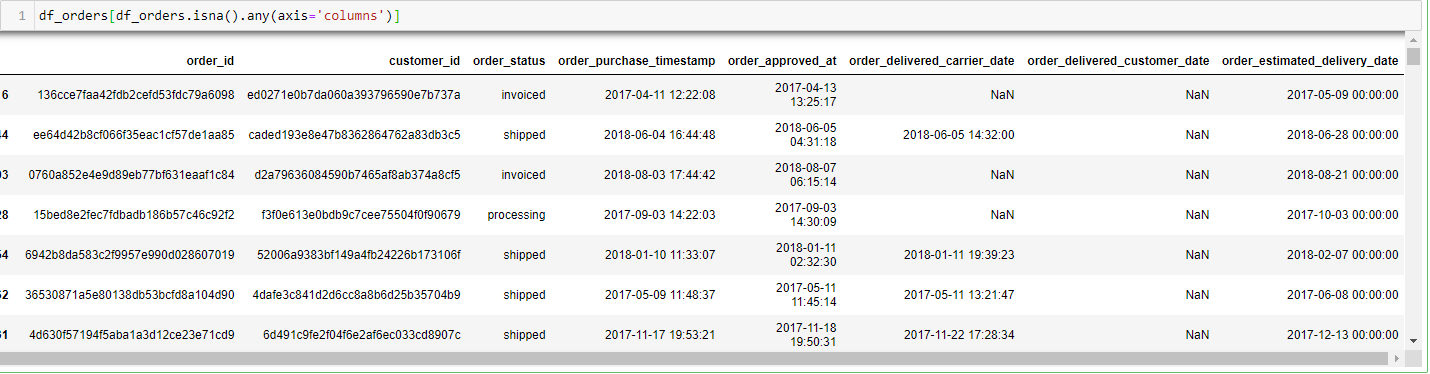
* 1. - Análise de olist\_orders\_dataset.csv transformado em df\_orders



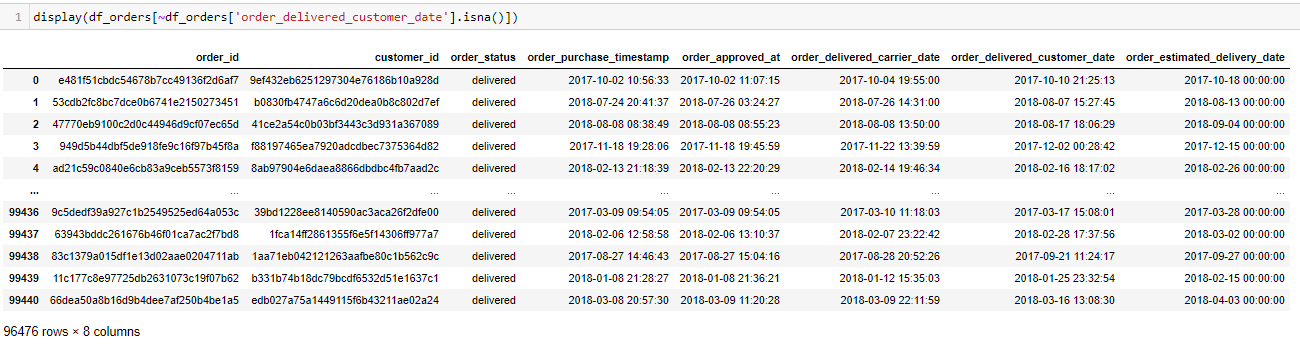
Como podemos perceber, há elementos NaN em algumas colunas de df\_orders. Então temos que analisar aonde estão este NaN e que decisão tomar quanto a estes dados faltantes.

No resumo já podemos averiguar que os valores NaN estão nas colunas de datas deste dataframe.

Neste caso, ao usar comando “df\_orders[df\_orders.isna().any(axis='columns')]”, pode-se analisar as linhas que possuem pelo menos um valor de NaN (ou NA) como mostrado abaixo:

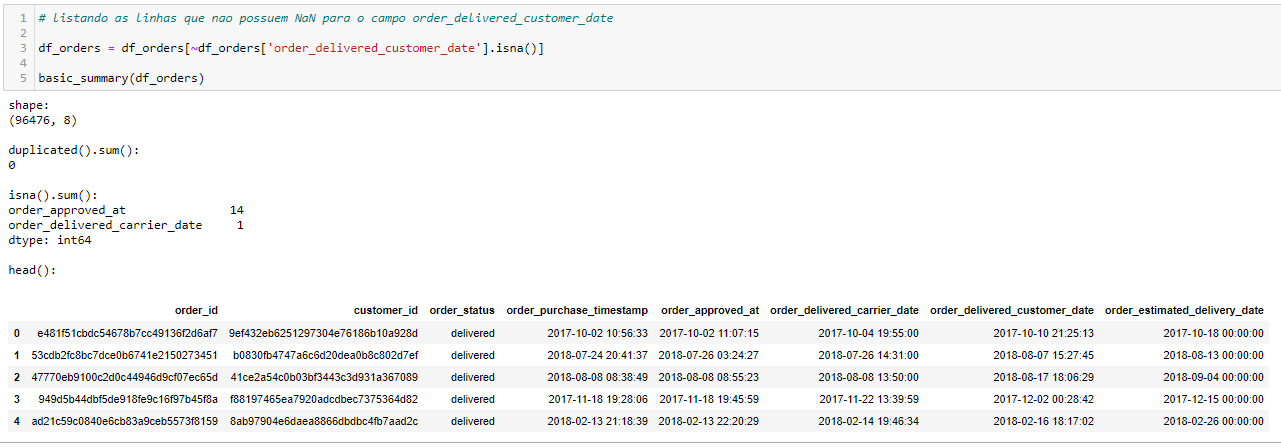


Analisando o campo “order\_delivered\_customer\_date” e listando quantas linhas existem para os elementos não NaN.

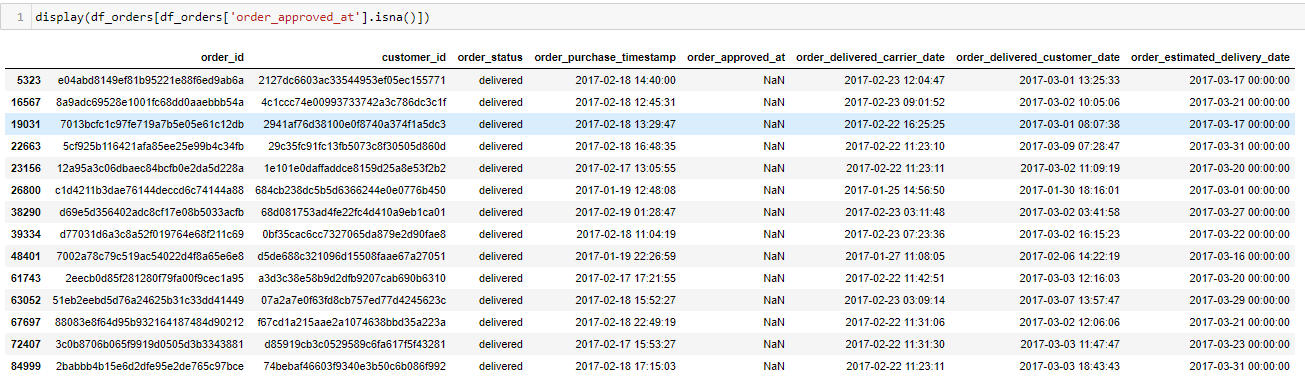


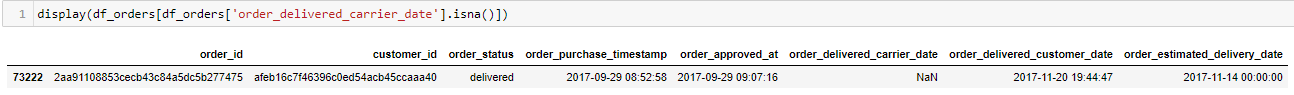
Estas linhas representam as mercadorias que foram entregues aos consumidores.

Então estas linhas serão o novo dataframe “df\_orders”



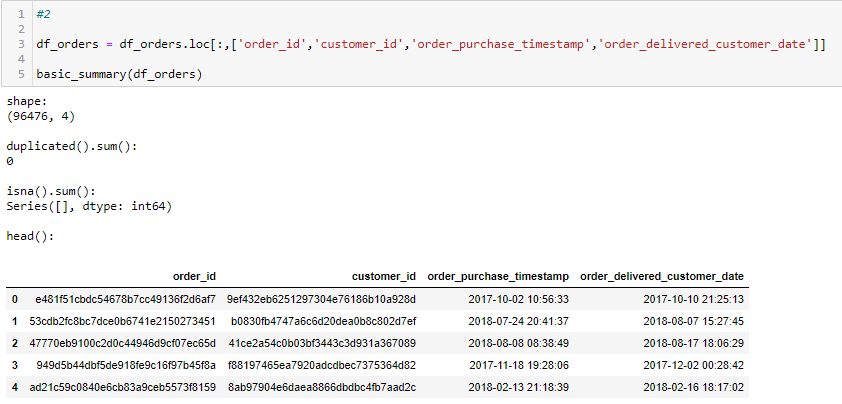
Ainda pode-se observar campos NaN neste novo dataframe.





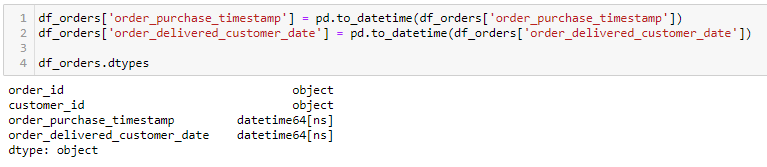
Mas os campos das datas de interesse que são “order\_purchase\_timestamp” e “order\_delivered\_customer\_date”, não possuem NaN.

Removendo os campos e mantendo os de interesse ficou-se com o seguinte dataframe:

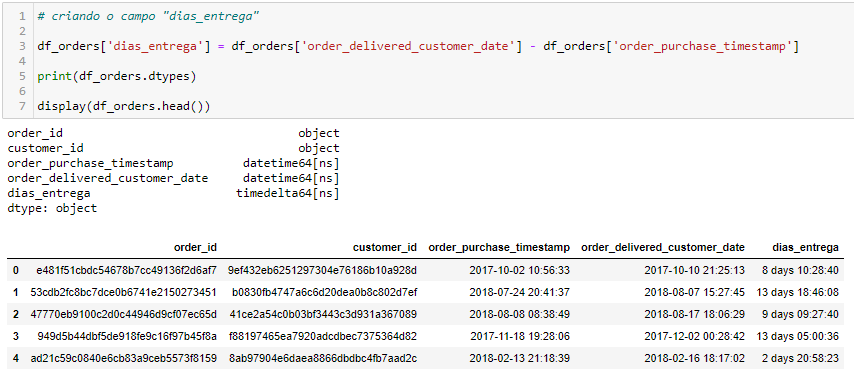


Foi criado o campo “dias\_entrega” que é a diferença, em dias, da data da compra até a data da entrega ao destinatário.

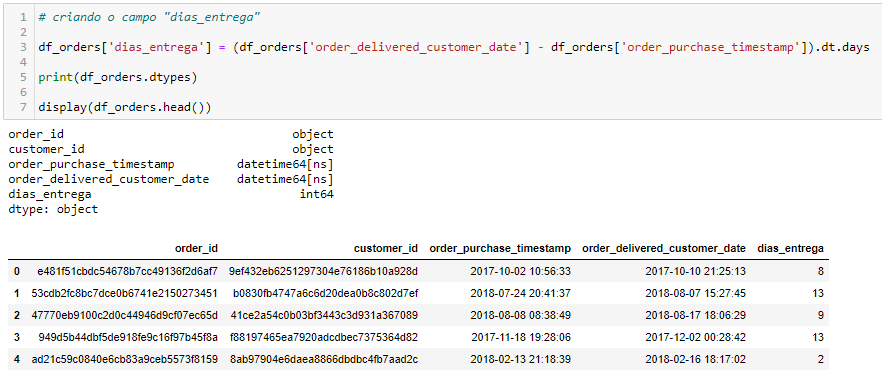
Primeiro passou-se os campos de data para o formato datetime:



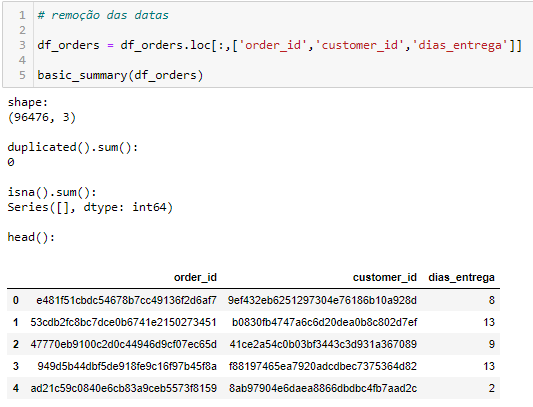
Então fez-se a coluna com a diferença das datas:



Fazendo o coluna direto mostrando somente os dias:



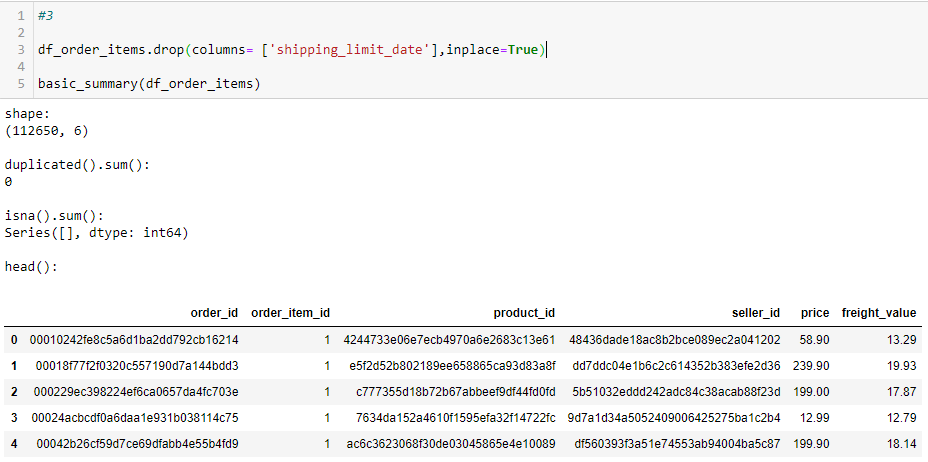
As datas foram removidas e o dataframe final ficou como segue:



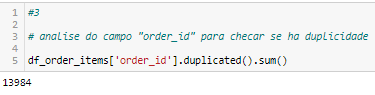
* 1. - Analise de olist\_order\_items\_dataset.csv transformado no dataframe df\_order\_items



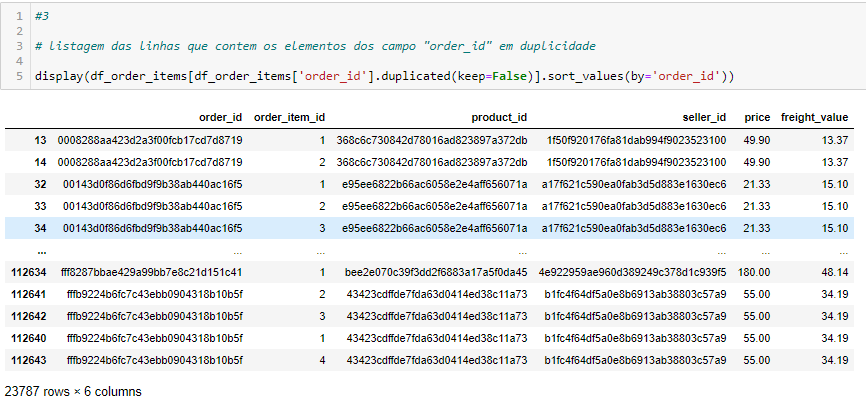
Neste dataframe o campo “shipping\_limit\_date” não será utilizado e foi removido conforme figura abaixo:



Fazendo a análise dos valores repetidos para o campo “order\_id” através de:

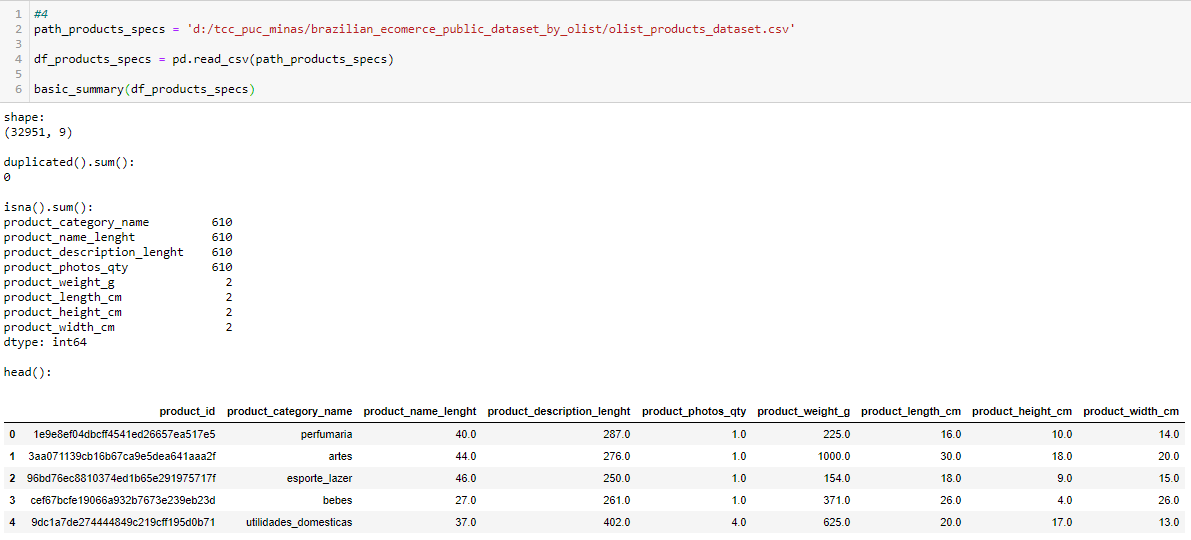


E listando as linhas que possuem valores duplicados, conforme mostrado abaixo:

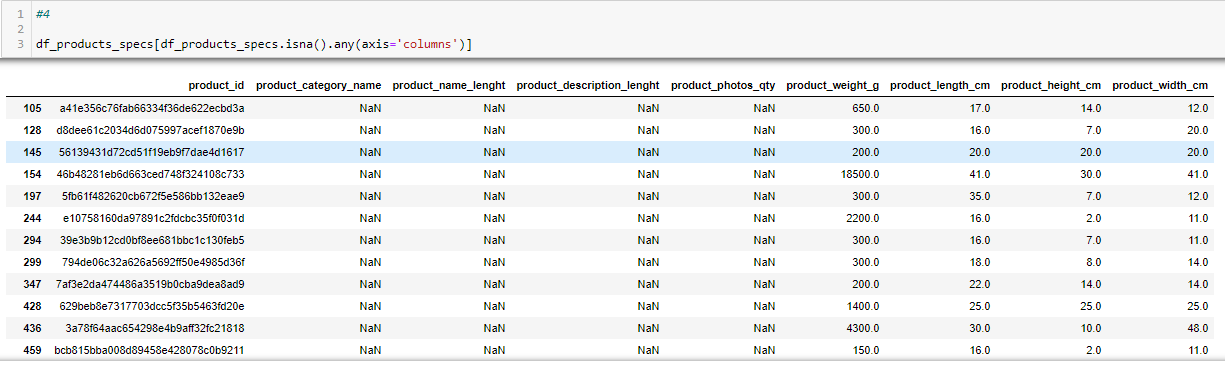


Para este dataframe temos que ,para um mesmo “order\_id”, termos mais de um “product\_id”.

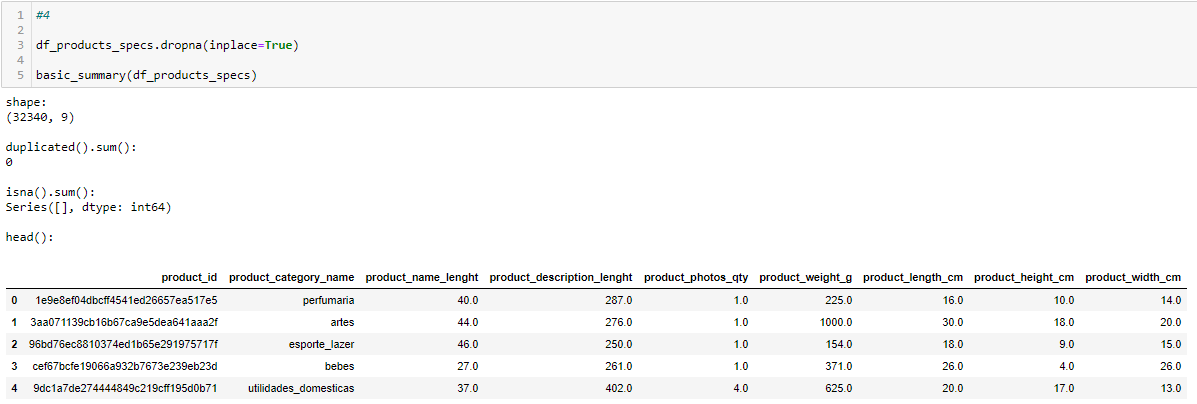
* 1. - Análise de olist\_products\_dataset.csv transformado em df\_products\_specs



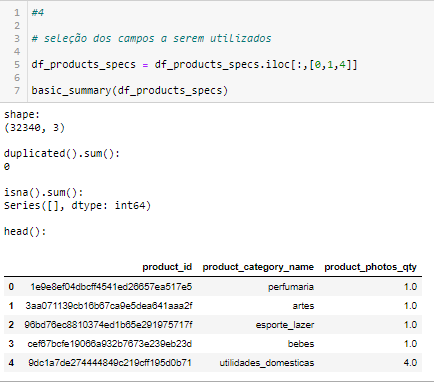
Como foi observado, temos elementos NaN em algumas colunas do dataframe. Então vamos analisar as linhas para termos uma visão geral das ocorrencias.



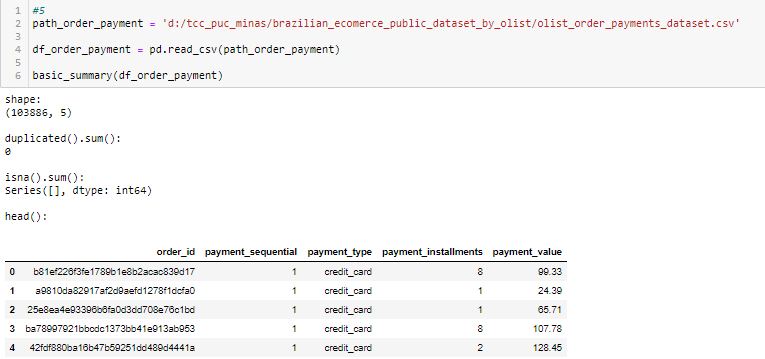
As linhas nas quais estes campos estão com valores NaN não serão aproveitadas e então procedeu-se à remoção das linhas



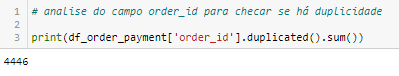
Os campos que serão aproveitados deste dataframe são “product\_id”, product\_category\_name” e “product\_photos\_qty”. Então foi criado um novo dataframe apenas com os campos a serem utilizados.



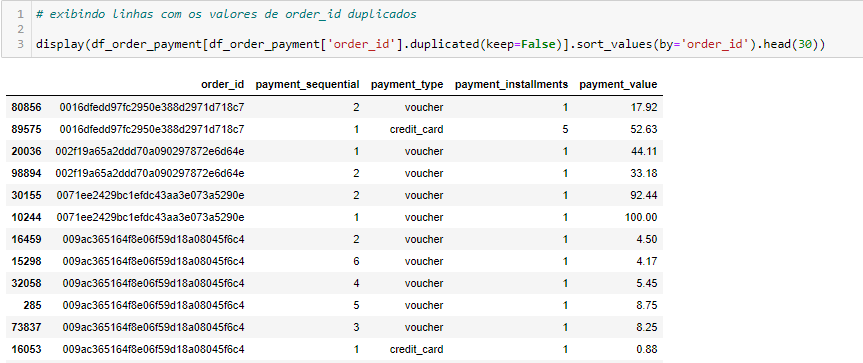
* 1. - Análise de olist\_order\_payments\_dataset.csv transformado em df\_orders\_payment



Pelo resumo não há linhas duplicadas e nem elementos NaN em qualquer coluna. Foi feita uma análise da ocorrência de duplicidade em order\_id.



Há valores duplicados em “order\_id” no dataframe. Vamos exibir as linhas duplicadas para poder estudar melhor o dataframe.

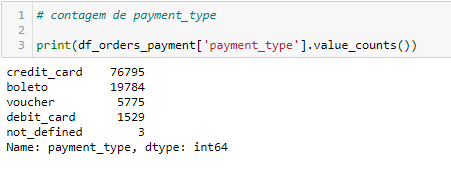


Pode-se observar que para uma mesmo valor de “order\_id” temos mais de um valor para “payment\_type”.

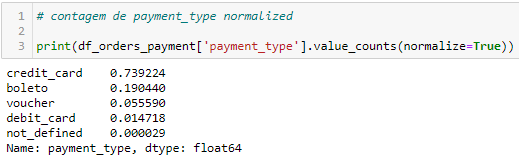
Os valores unicos que aparecem no campo “payment\_type são:



Fez-se um contagem de quantas vezes os valores unicos estão presentes em todo o dataframe:

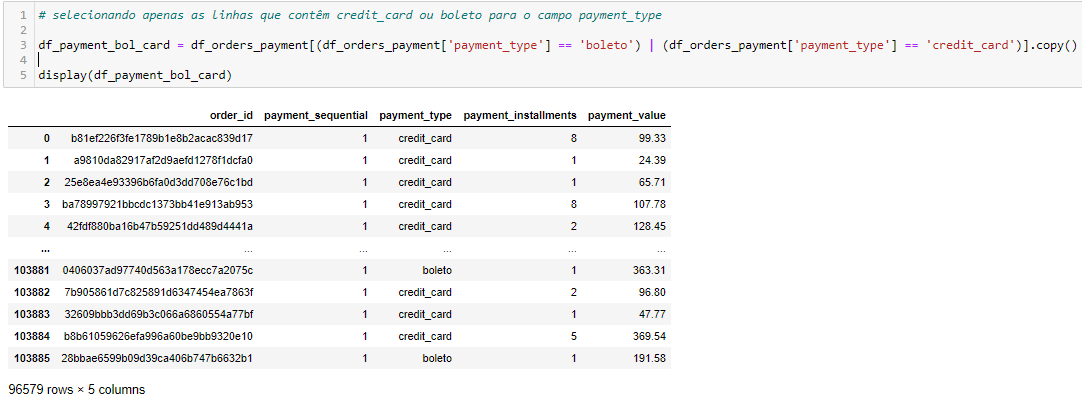


E tambem a contagem normalizada. Na qual os items aparecem como percentual do todo.

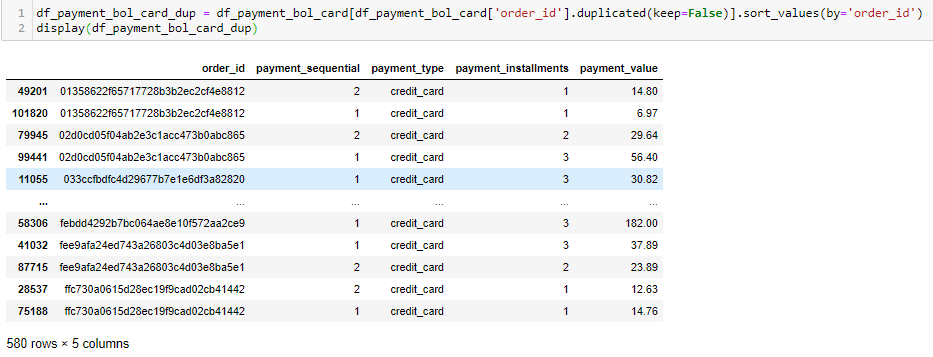


Pode-se obeservar que os tipos “credit\_card” e “boleto” representam mais de 80% do numero de ocorrencias no dataframe. Então decidiu-se usar estes dois items para o classificador.

Então foi criado um dataframe que continha apenas os elementos “credit\_card” e “boleto” no campo “payment\_type”.

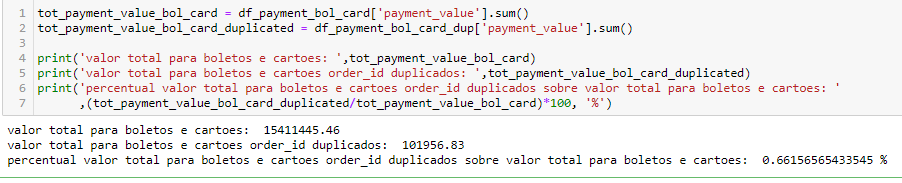


A partir deste dataframe se fez uma análise dos elementos de “order\_id” repetidos.

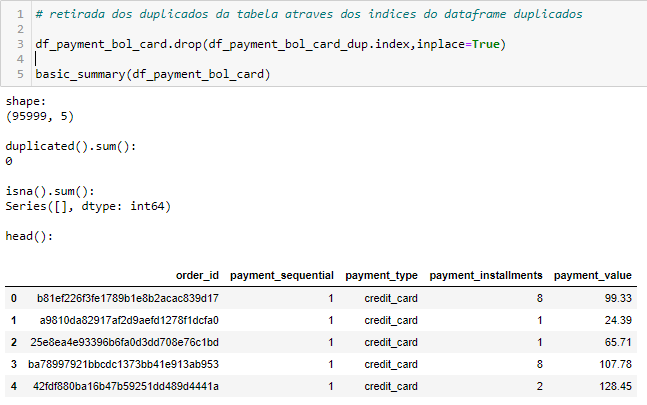


Pode-se observar que para um mesmo elemento de “order\_id”, tem-se uma repetição de um elemento em “payment\_type”.

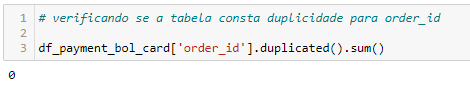
Fez-se então um levantamento de, quanto do valor total de “payment\_value” do dataframe, os elementos repetidos representam.



O percentual é de aproximadamente 0,66% do total. Então decidiu-se por excluir estas linhas do dataframe.



Mais uma vez se fez o teste para os elementos duplicados em “order\_id”.



Renomeou-se o dataframe para o nome “df\_orders\_payment”



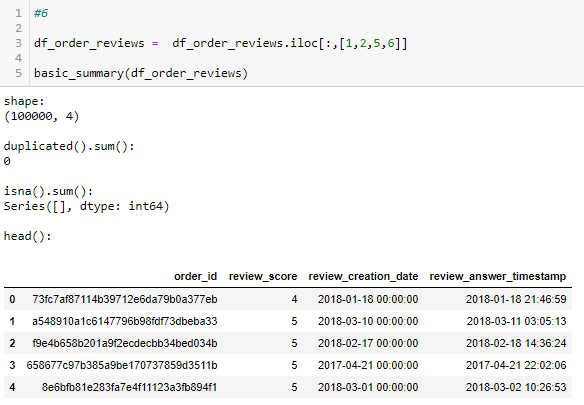
3.6 - Análise de olist\_order\_reviews\_dataset.csv transformado em df\_order\_reviews



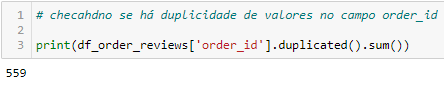
O resumo mostra que o dataframa possui NaN em duas colunas: “review\_comment\_title” e “review\_comment\_message”.

O campo “review\_id” é um campo de identificação e não será utilizado porque não possui qualquer referência em outro dataframe.

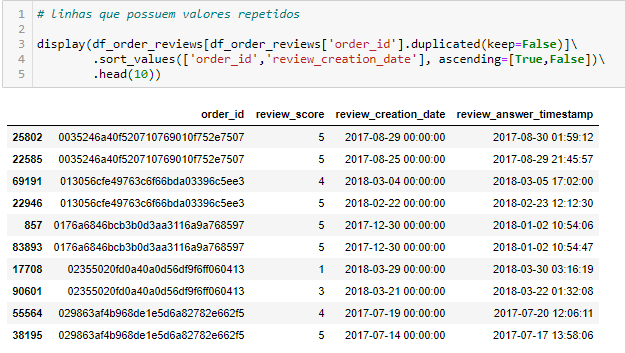
Os campos que não serão utilizados foram excluidos para o prosseguimento da análise no dataframe resultante.



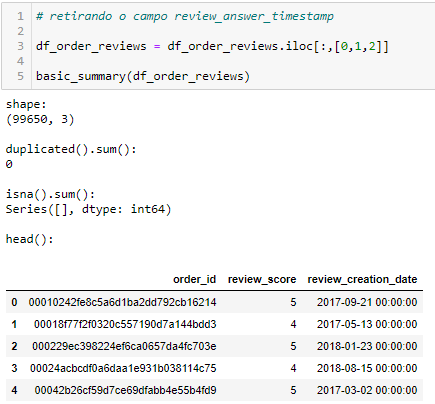
Fez-se então a verificação de valores repetidos no campo “order\_id”.



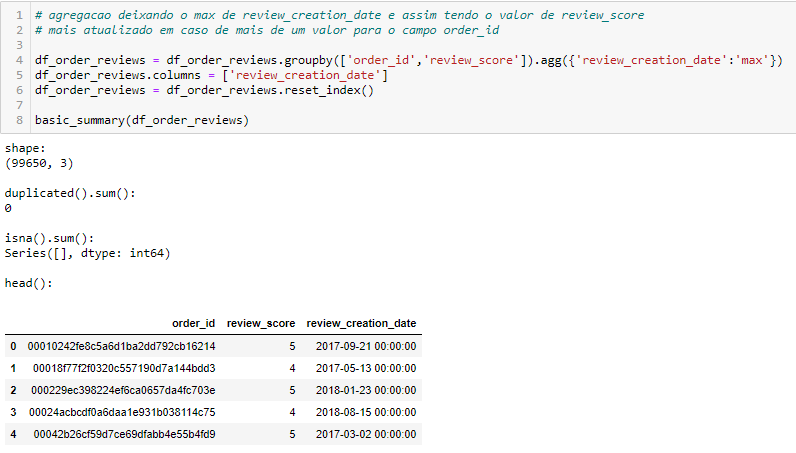
Como há valores repetidos, listou-se as linhas que possuem valores repetidos.



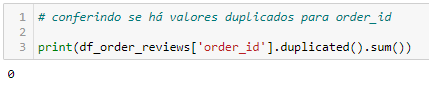
Nesta etapa, o campo “review\_answer\_timestamp” será removido porque não será usado.



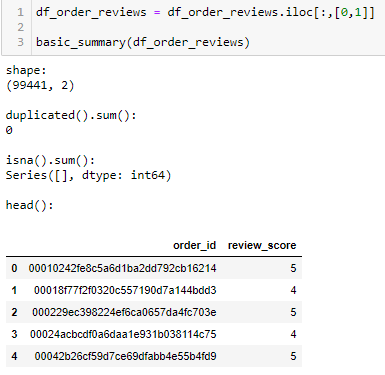
Neste caso, como para um mesmo valor de “order\_id” acontece de ter mais de um valor de “review\_score”, optou-se por deixar o valor mais recente através da agregaçao por maior valor para “review\_creation\_date”.



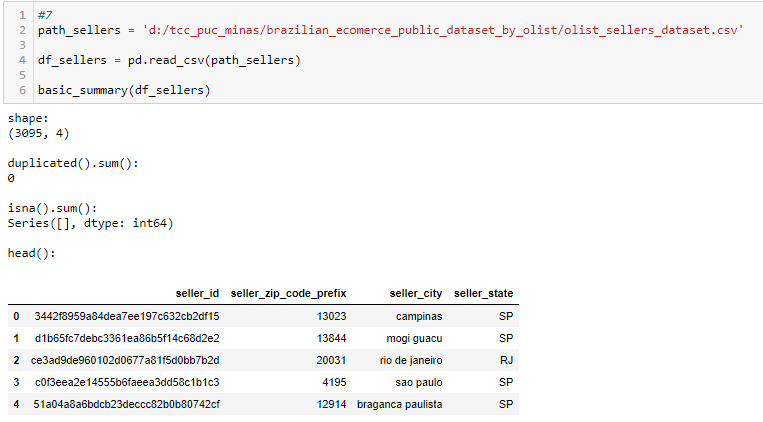
Faz-se de novo a conferencia para ver se há mais valores duplicados para o campo “order\_id”



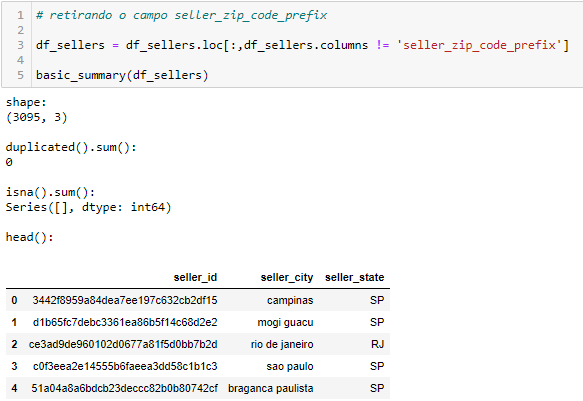
Neste caso o campo “review\_creation\_date” já perdeu sua função e foi excluido.



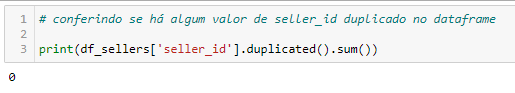
3.7 - Análise de olist\_sellers\_dataset.csv transformado em df\_sellers



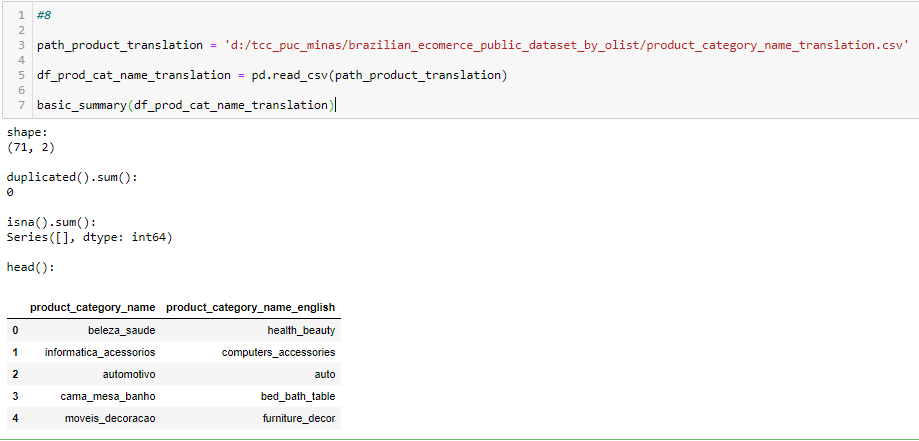
Neste dataframe o campo “seller\_zip\_code\_prefix” será excluído por já se ter a informação do municipio e estado.



Verificou-se se há valores duplicados para o campo “seller\_id”.

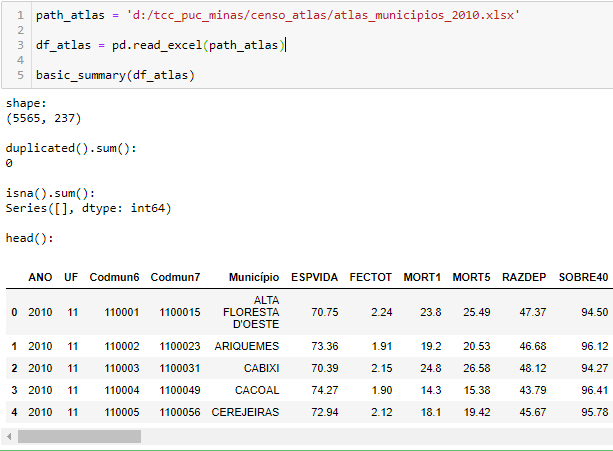


3.8 - Análise de product\_category\_name\_translation transformado em df\_prod\_cat\_name\_translation

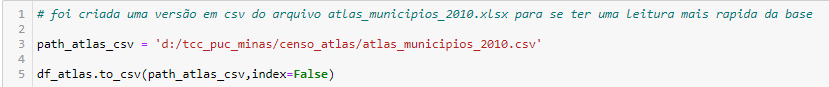


Este dataframe não será utilizado no projeto porque faz apenas uma tradução do português para o inglês dos nomes das categorias dos produtos. E no dataframe “df\_products\_specs” o valor de “product\_category\_name” já está em português.

3.9 - Análise da base atlas que foi transfomada no dataframe df\_atlas



O dataframe foi convertido para um arquivo no formato csv para a leitura mais rapida

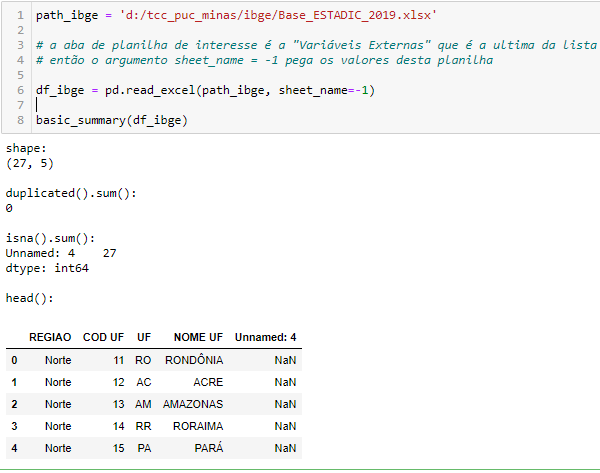


O dataframe “df\_atlas” possui 237 campos e nem todos serão utilizados no classificador. Mais a frente, haverá uma análise da importancia dos campos para o classificador e da correlações entre os campos. De posse dessa informação será feita uma análise do significado do campo e se será utilizado no classificador.

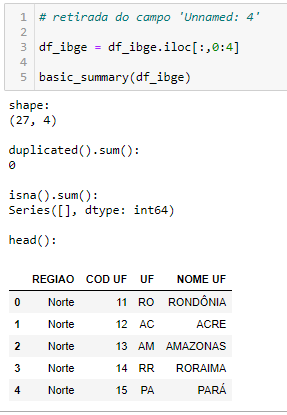
3.10 - Análise da base ibge que foi transfomada no dataframe df\_ibge

No arquivos Base\_ESTADIC\_2019.xlsx temos que aba de planilhas de interesse é a "Variáveis Externas" porque vamos utilizar para cruzar com o dataframe “df\_atlas” e associas a cada valor do campo “UF” a sigla do estado e a região a qual pertence.

Como esta aba é a ultima, o argumento sheet\_name = -1 faz com que o pandas leia os dados desta aba.



No resumo e na exibição do dataframe percebe-se que o campo “Unnamed: 4” possui todos os valores NaN. Então este será retirado do dataframe “df\_ibge”.

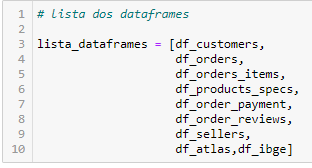


3.11 – Análise dos dataframes e seus campos no contexto de formar um dataframe único.

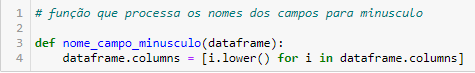
Os dataframes da olist são provenientes de arquivos distintos e possuem campos comuns em alguns arquivos. Estes campos comuns servirão como chave para montarmos um dataframe da olist único que, então, será unido com o dataframe do altas e, após esta junção, será unido com o dataframe do ibge. Compondo um dataframe que terá todos os campos de interesse em uma base. E esta base única será usada no classificador.

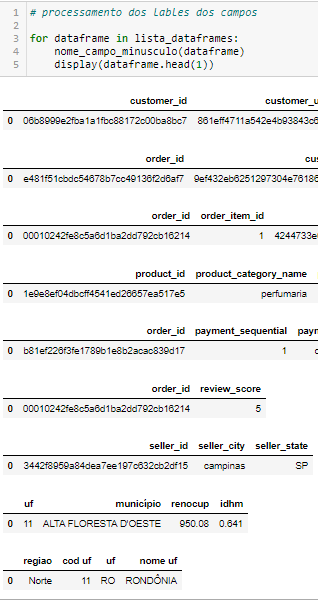
Percebe-se que há dataframes com os nomes dos campos em maiúsculo e outros dataframes com os nomes em minúsculo. Essa diferença já gera errro na hora de fazer a junção porque o Python é sensível para nomes de varáveis maiúsculas e minúsculas.

Antes fez-se uma lista com os nomes dos dataframes para poder processar em lote.



Uma função foi criada para passar o nome dos campos para minúsculo. Os nomes que já estiverem em minúsculo continuarão.

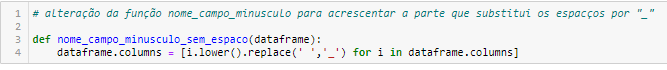




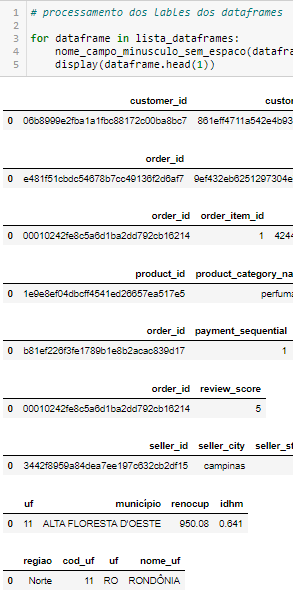
Pela observação dos campos de df\_atlas e df\_ibge, já temos campos em comum com nomes diferentes, mas que represtentam a mesma coisa, o codigo dos municipios.

Observa-se que que alguns campos do df\_ibge possuem espaços. Estes campos terão os espaços substituídos por “\_” (underscore).

A função anterior foi modificada para poder transformar os nomes dos campos em minúsculo e substituir os espaços por “\_”



Abaixo é mostrado o resultado do novo processamento. O anterior ficará comentado com ‘’’ (três aspas) no começo.

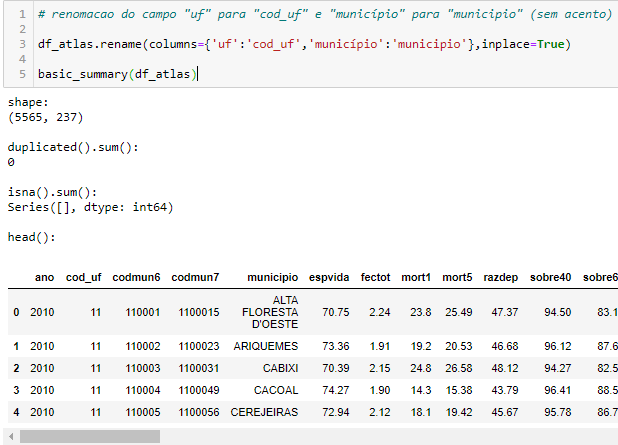


Comparando os campos de df\_atlas com df\_ibge, percebe-se que os valores do campo “uf” de df\_atlas não corresponde aos valores do campo em df\_ibge.

O campo “uf” de df\_atlas corresponde ao campo “cod\_uf” do dataframe df\_ibge.

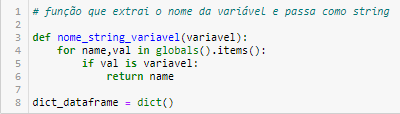
Observou-se, também, que o campo “município” está com acento e, então será renomeado para “municipio” (sem o acento)

Com base nisso, renomeou-se o campo “uf” para “cod\_uf” em df\_atlas.

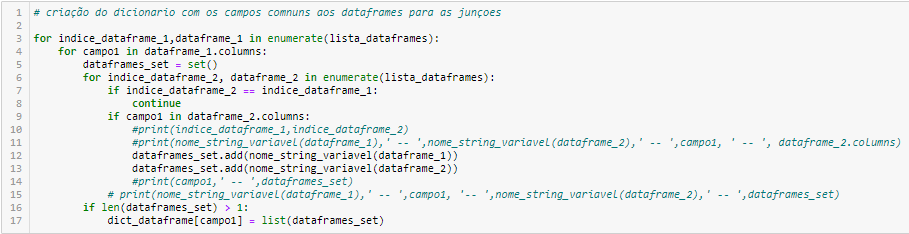


Foi criado um dicinario que contem como chave o nome do campo e como valor uma lista com os dataframes que possuem este campo em comum.

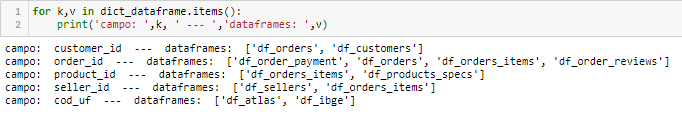
Para isso foi nessário criar uma uma função que faz lê o nome da varável e a passa como string para ser usada na listagem dos dataframes. Isso porque se colocarmos print(df\_ibge), por exemplo, o notebook vai listar o conteúdo do dataframe e não o nome em si. Esta função foi feita para evitar ficar criando uma nova lista somente com os nomes dos dataframes como string.



Então, depois de criada a função fez-se a criação do dicionario.



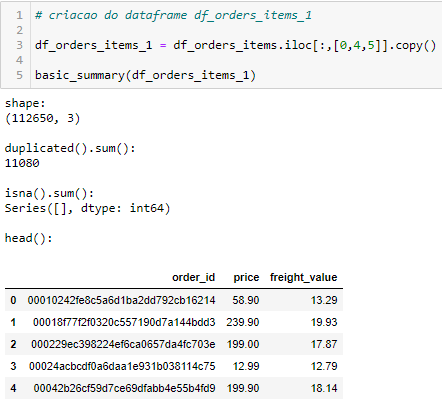
Então listou os elementos do dicionario para poder dar inicio ao merge dos dataframes.



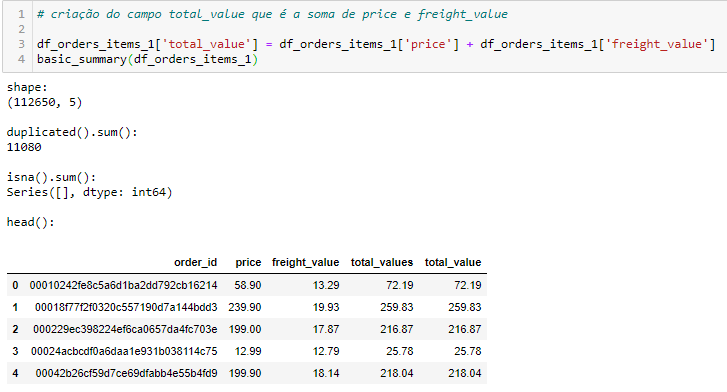
Observou-se que o dataframes “df\_order\_payment” traz o campo payment\_value e que o dataframe “df\_order\_items” traz traz os campos “price” e “freight\_value” que são relacionados ao campo “oder\_id”.

Então procedeu-se a uma conferência para verificar se as tabelas apresentam valores somados iguais para um mesmo “order\_id” nos dois dataframes

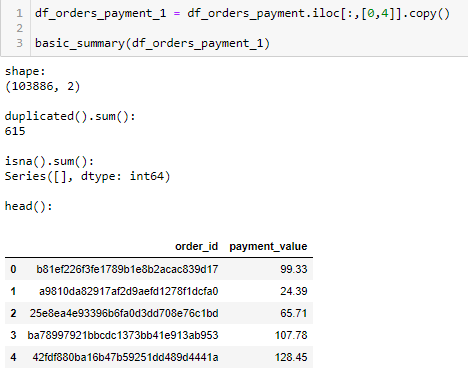
Para o dataframe “df\_orders\_items”, foi criado um outro dataframe os campos “order\_id”, “price” e “freight\_value”.



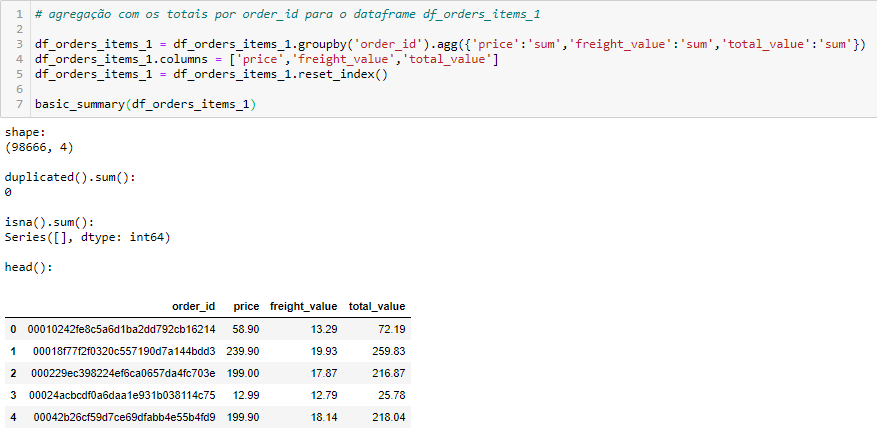
Foi criado um campo “total\_value” que é a soma de “price” + “freight\_value”.



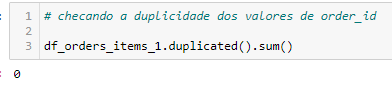
E foi criado um novo dataframe do “df\_orders\_payment” com os campos “order\_id” e “payment\_value”.



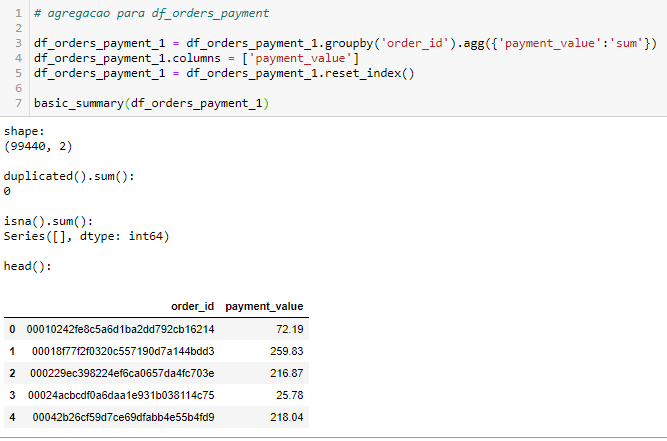
Foi feita a agregação com os totais para cada “order\_id” do dataframe “df\_orders\_items\_1”



Pode-se obeservar que, com a agregação, as linhas duplicadas zeraram. Abaixo fez-se o teste para ver se ainda havia elementos duplicados no campo “order\_id”.



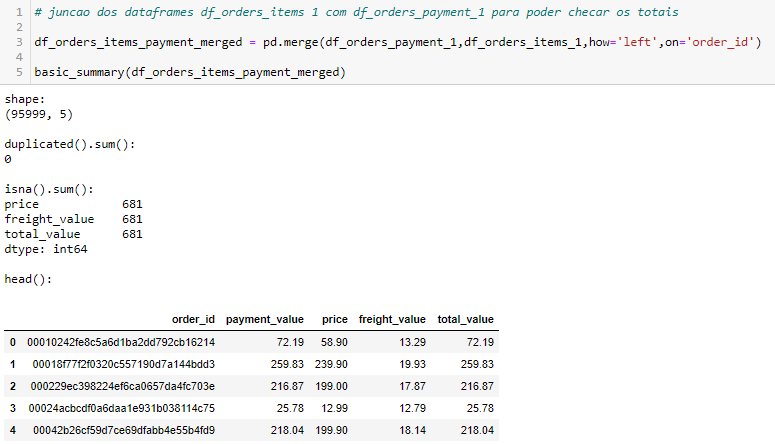
O mesmo foi feito para o dataframe “df\_orders\_payment\_1”.



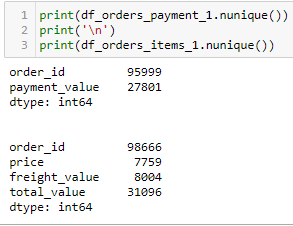
Fez-se então uma junção dos dois dataframes para poder checar as somas totais para os elementos de “order\_id”

Verifica-se que os valores de “payment\_value” e “total\_value” são iguais. Então temos que as bases refletem igualmente os totais dos elementos de “order\_id”.

Não houve correspondencia de “df\_orders\_items\_1” sobre “df\_orders\_payment\_1” em 681 ocorrencias.



Contagem dos valores unicos para cada campo dos dataframes:



Ou seja, existem 681 elementos no campo “order\_id” a mais para o “df\_orders\_payment\_1”.

A proposta neste momento foi comprovar que, no caso de mesmo elemento de “order\_id” a soma dos totais é a mesma.

A diferença entre os elementos unicos de cada dataframe será tratada no momento da junção.

3.12 – União dos dataframes

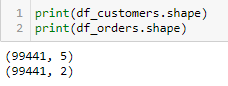
Os dataframes que foram vistos serão unidos através do método merge do pandas (<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html>).

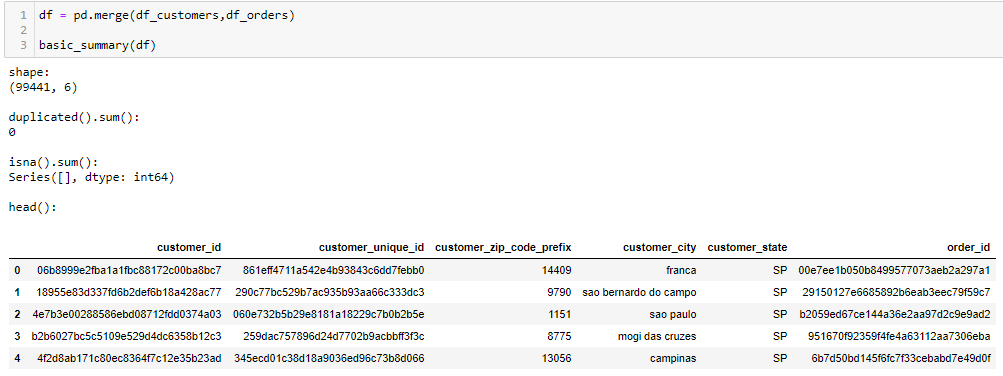
A medida que os merges são feitos, há uma análise das linhas duplicadas, e casos de NaN que possam ocorrer porque os elementos de uma base não tiveram correspondência na outra.

Converncionou-se fazer o merge do tipo left join utilizando na coluna da esquerda o dataframe com o maior numero de linhas.

3.12.1 – União dos dataframes “df\_customers” e “df\_orders”

O campo usado para a junção foi o campo “order\_id”



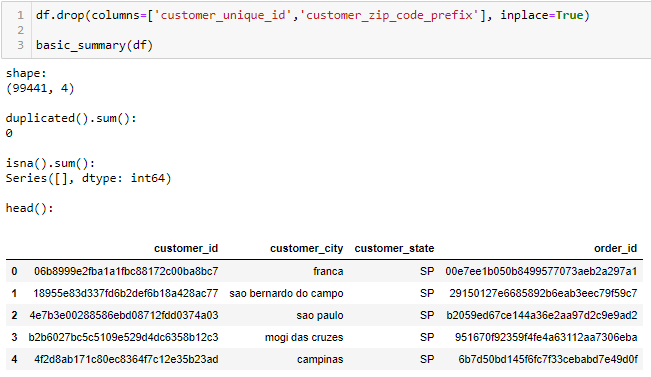


Para o dataframe final, os campos “customer\_unique\_id” e “customer\_zip\_code\_prefix” não serão utilizados.

O campo “customer\_unique\_id” é um campo de indentificação única que não possui outra correspondencia nas bases e está sendo redundante para o dataframe porque já temos um campo de identificação que é o “customer\_id”.

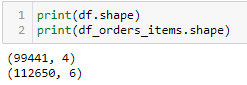
No caso campo “customer\_zip\_code\_prefix” , já se tem a informação da cidade e do estado de cada “order\_id” e as outras junções serão feitas por estes elementos.

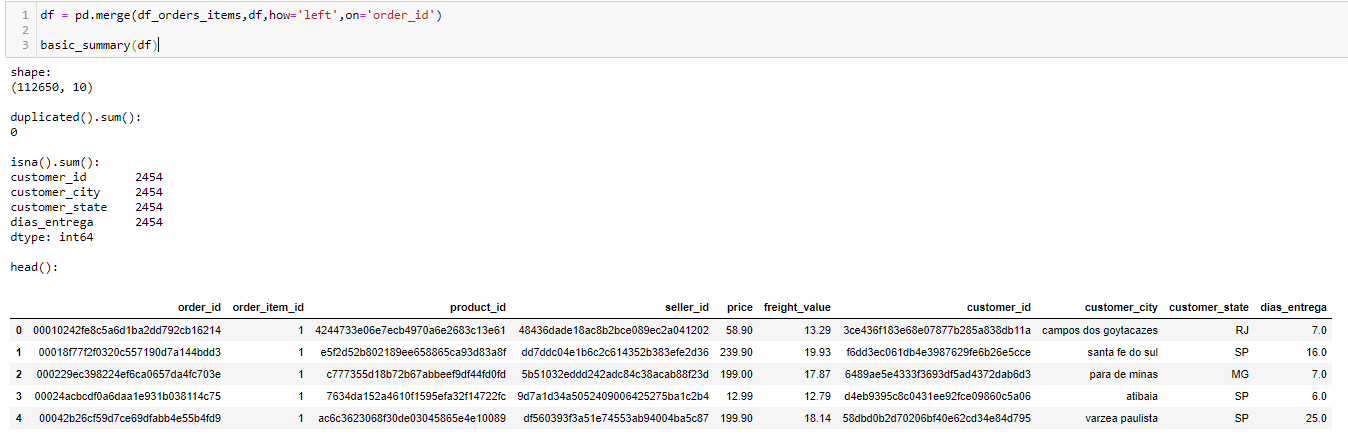
Então removeu-se os campos e o dataframe ficou como segue:



3.12.2 – União dos dataframes “df” e “df\_orders\_items”

O campo usado na junção foi o “order\_id”





Como algumas linhas do dataframe “df\_orders\_items” foram excluídas, já eram previstas linhas com NaN

Então estas foram removidas.

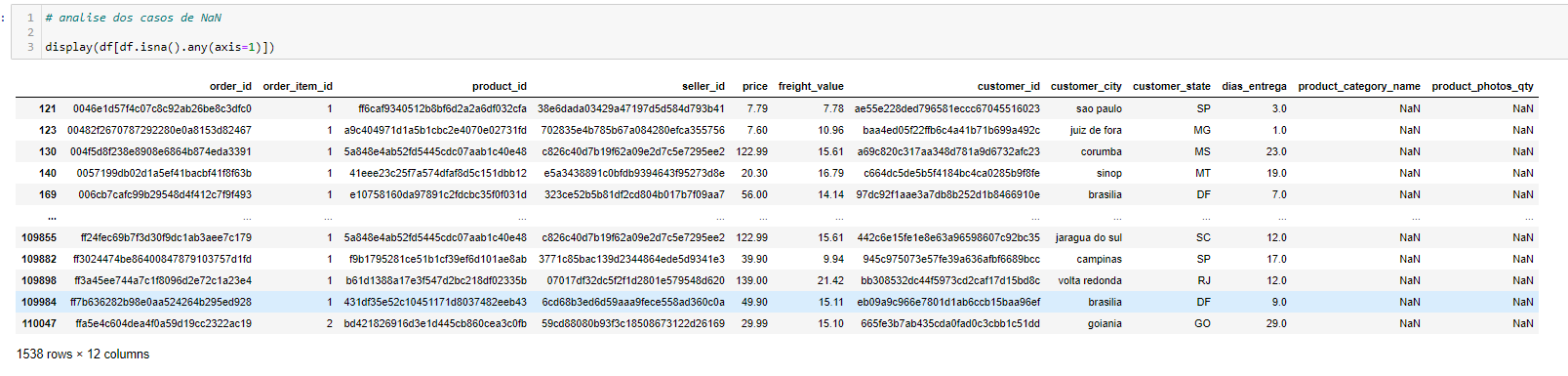
3.12.3 – União dos dataframes “df” e “df\_product\_specs”

O campo utilizado na junção foi o campo “product\_id”

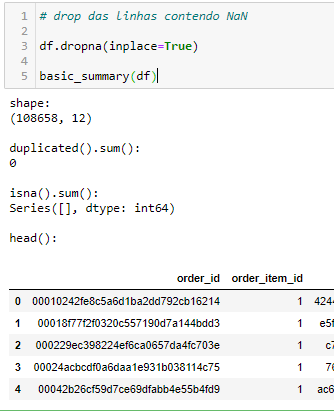


Pelo resumo tivemos que não hove 1604 correspondencia do dataframe “df\_products\_specs” sobre o dataframe “df” no campo “product\_id”.

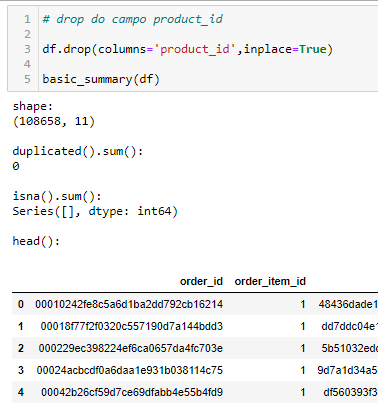
Agora procede-se à análise dos casos de NaN



Neste caso as linhas serão excluídas do dataframe “df”.



Temos agora o campo “product\_id” que perdeu sua utilidade, haja vista o campo “product\_category\_name” que informa qual é a categoria de cada “product\_id”. Então este campo foi removido do dataframe “df”



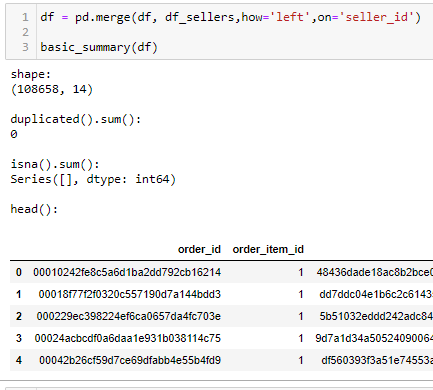
3.12.4 – União dos dataframes “df” e “df\_orders\_review”

O campo de junção foi o “order\_id”



3.12.5 – União dos dataframes “df” e “df\_sellers”

O campo de junção é o “seller\_id”



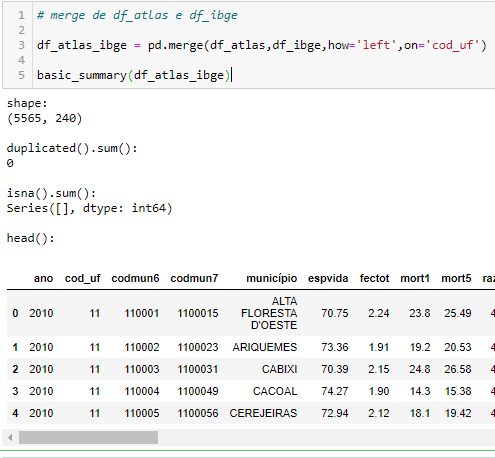
Esta junção ressaltou um erro que há na base da olist no nome do município e do estado a qual pertence.



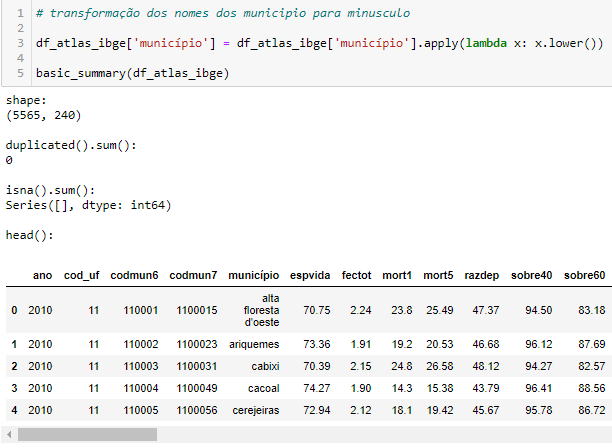
Neste caso, Volta Redonda pertence ao estado do Rio de Janeiro e não à São Paulo.

Para correção deste problema, primeiro fez-se a união dos dataframes “df\_atlas” e “df\_ibge”.

A junção destas bases se deu com o campo “cod\_uf”



Os nomes dos municípios foram transformados em minúsculo.



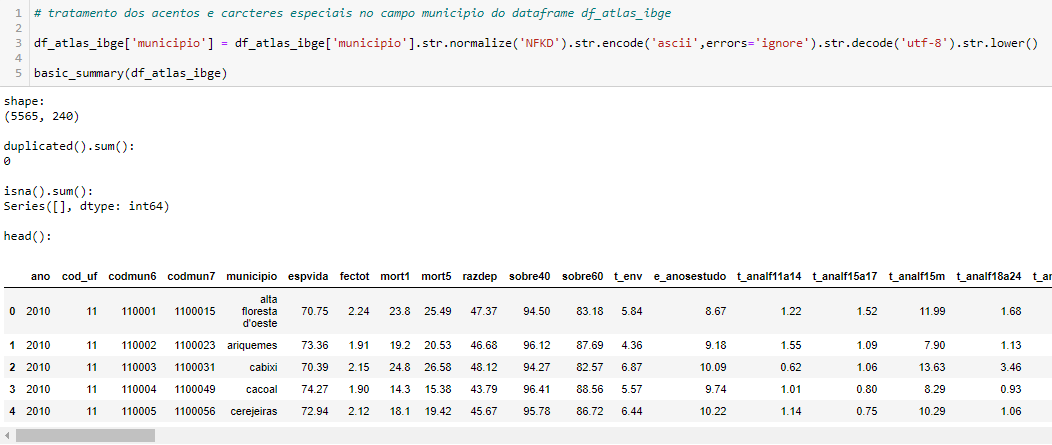
Os nomes dos municípios contém acentos e caracteres especiais, como “ç”. Na base olist, os nomes dos municípios estão sem esses elementos.

Para isso faz-se uma decomposição dos caracteres especiais em dois elementos distintos. Por exemplo, o “Á” (letra A com acento) é decomposto no código Unicode da letra a mais o código Unicode do acento.

A partir desta decomposição passa-se para a codificação ASCII, que contém os caracteres sem o acento. Então ser codificado, no caso do exemplo da letra “A” com o acento vai sair do decodificador como apenas A e o acento geraria um erro pois não existe o código do acento na tabela ASCII. Mas o argumento errors = ignore descarta esses erros.

Nesta parte temos uma letra A em bytes. Precisa-se então decodificar esta letra para o formato utf-8. O formato utf-8 é o mais comum e utizado nos sistemas atuais. Então esta decodificação do ascii para o formato utf-8 garante uma compatibilidade maior.

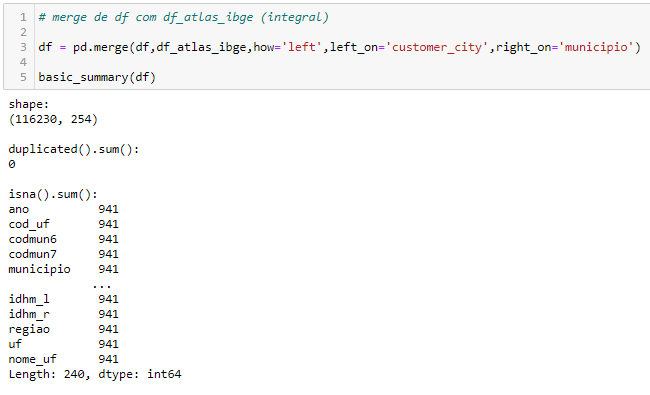
Abaixo é mostrado o resultado da retirada do acento e caracteres especiais.



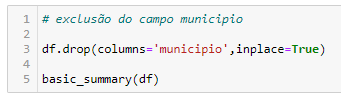
As colunas que contém as siglas dos estados serão removidas do dataframe df e serão efetuados dois merges. O primeiro vai ser entre “df” e o “df\_atlas\_ibge” integral. Usando como junção os campos “customer\_city” e “município”. E depois o campo “uf” será renomado para “customer\_state”.

O outro merge será feito entre o dataframe atualizado e o df\_atlas\_ibge com uma versão modificada temporariamente através de slicing para trazer apenas o nome do município e o código do estado. Esta junção será feita pelos campos “seller\_city” e “município”. E os campos serão altarados para que fique conforme os nomes originais.

O primeiro merge foi feito conforme mostrado abaixo:

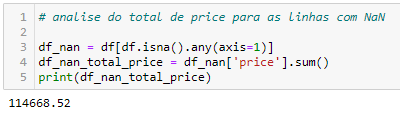


O o campo “município” foi excluído porque traz a mesma informaçõ que “customer\_city”.

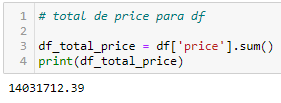


Neste caso temos a ocorrência de NaN, então alguns municípios de “df\_atlas\_ibge” não encontraram correspondência em “df”. Procedeu-se então para a análise dos NaN

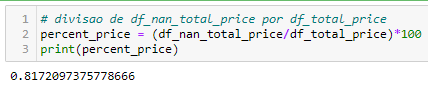
Fez-se então uma análise de quanto dos valores de “price” estes NaN representam.



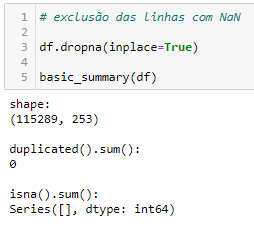
Agora fez-se o total de price para “df”



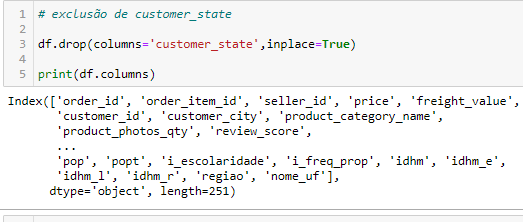
Então dividiu-se o valor de “df\_nan\_total\_price” por “df\_total\_price” e multiplicou por 100 para ser chegar a um numero percentual.



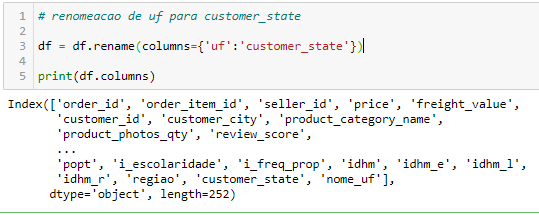
Como o valor foi baixo, decidiu-se por excluir as linhas com NaN.



Agora fez-se a exclusão do campo “customer\_state”. O campo “uf” foi incorporado ao datarame e este contém as siglas dos estados e este dado é mais confiável que “customer\_state”. Após a exclusão, o campo “uf” foi renomado para “customer\_state”.

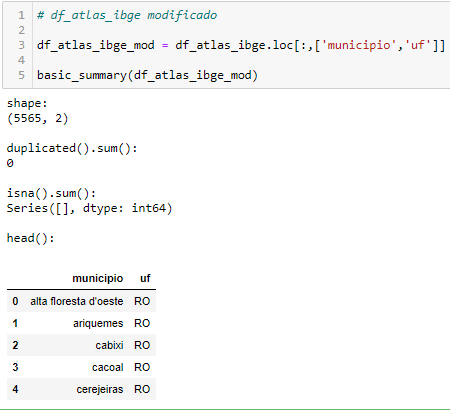


E agora o campo “uf” foi renomado para “customer\_state”.

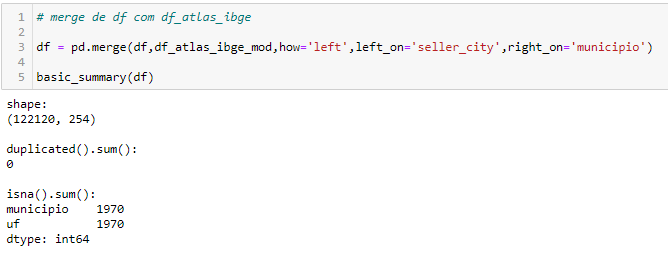


Passou-se para a parte de trabalhar as siglas dos estados para o campo “seller\_state”

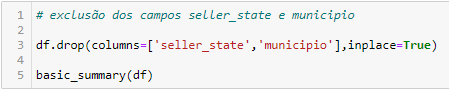
Montagem do dataframe modificado para ser usado no merge.

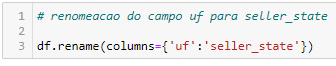


Merge de “df” com “df\_atlas\_ibge\_mod”. A junção será feita pelo campo “seller\_city”

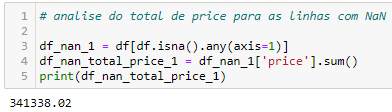


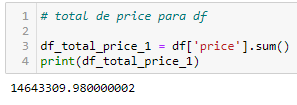
Os campos “seller\_state” e “município”foram excluídos. E o campo “uf” foi renomeado para “seller\_state”.

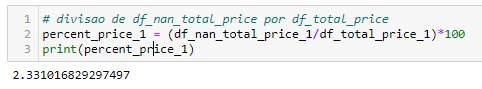




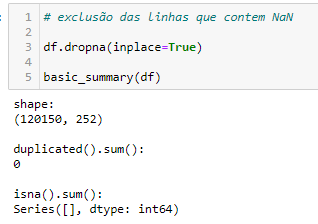
Fazendo os cálculos dos números de NaN temos que:



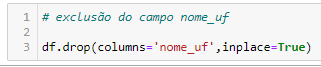




A quantidade de linhas com NaN representam 2,33 do valor total do dataframe. Decidiu-se por excluir estas linhas.

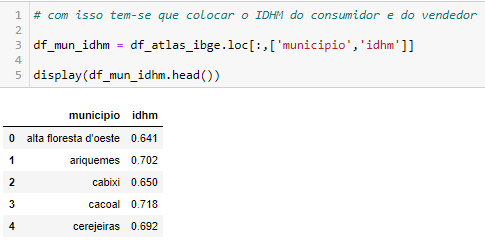


O campo “nome\_uf” foi retirado.

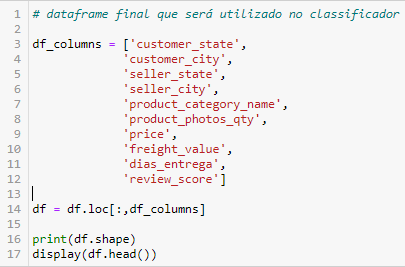


Para o dataframe final era importante colocar o IDHM de cada município referente ao “custumer\_city” e “seller\_city”

Primeiro foi criado um dataframe que é parte de df\_atlas\_ibge contendo os municípios e seu respectivo IDHM.

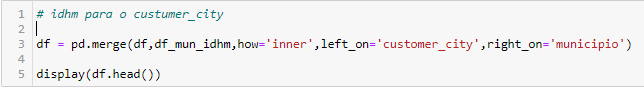


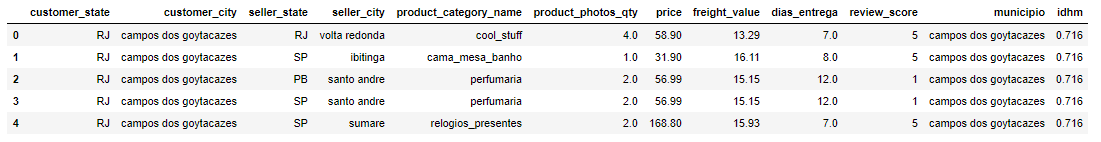
Após isso fez um dataframe já com os campos finais, faltando apenas os de IDHM para cada campo de cidade





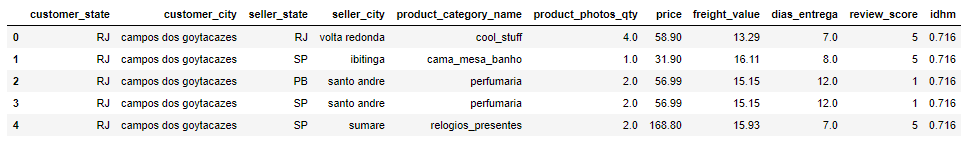
Então fez-se um merge do dataframe acima com o “df\_mun\_idhm”



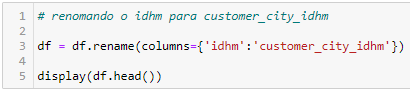


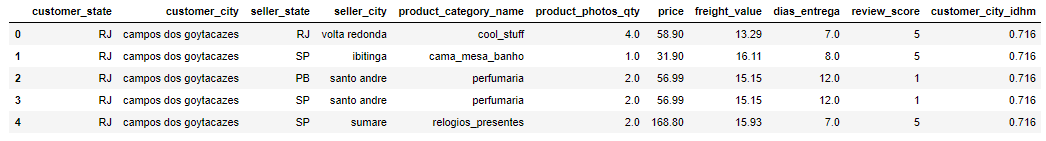
O campo “município” foi removido.porque já se tem os municípios no campo “cutomer\_city” e quer-se apenas o idhm para estes municípios.





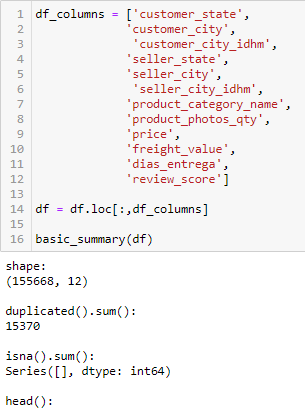
O campo IDHM foi renomeado para “customer\_city\_idhm”.

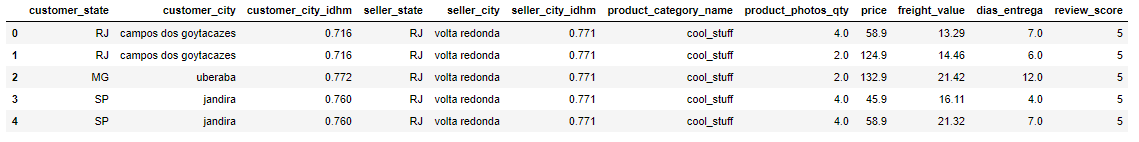




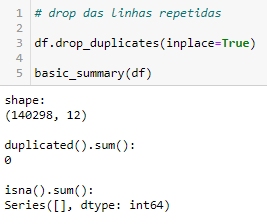
O mesmo procedimento foi efetuado para o “seller\_city” com o IDHM correspondente sendo “seller\_city\_idhm”.

O dataframe final ficou da seguinte maneira:





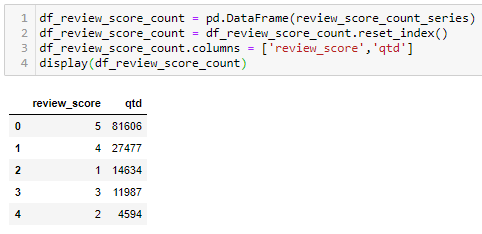
As linhas duplicadas foram eliminadas



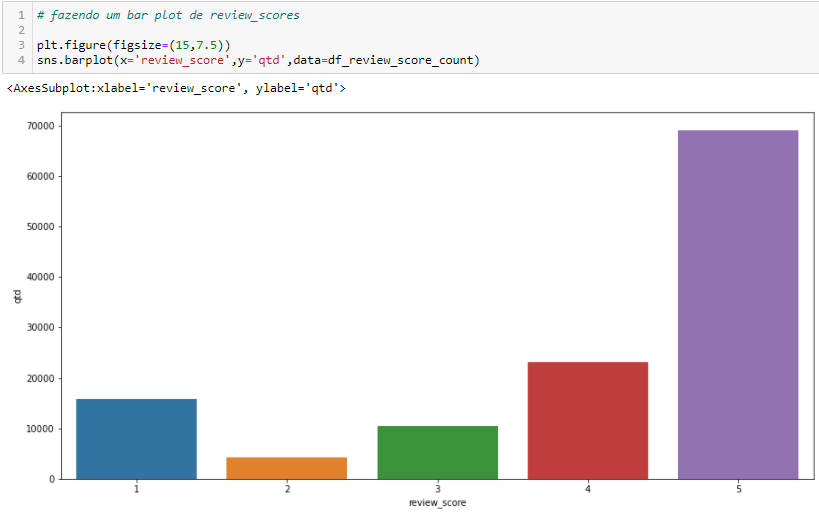
# 4. Análise e Exploração dos Dados

Como o classificador será binário, a primeira variável a ser analisada vai ser a review score

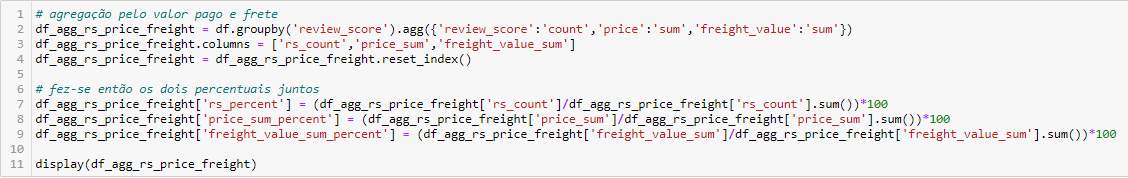
Primeiro fez-se a contagem dos elementos de review\_scores

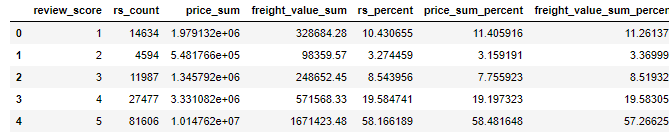


E foi plotado o gráfico de barras deste dataframe.



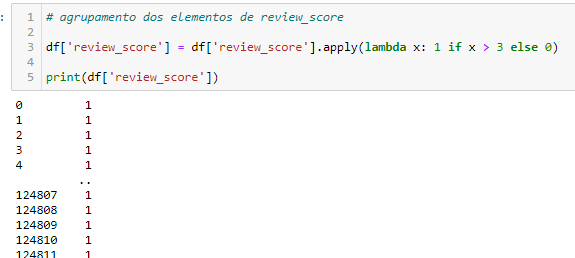
Fez-se um resumo dos somatórios e percentuais para cada valor da variável “review\_socore”.



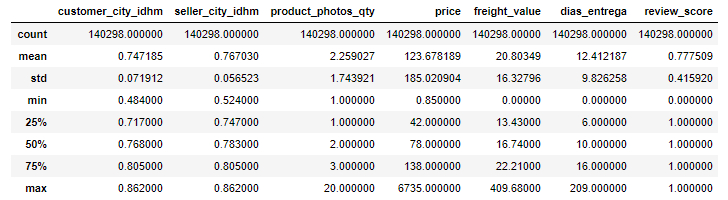


Os valores 4 e 5 representam mais quase 80% da contagem total de elementos da variável “review\_score”.

O campo “review\_score” foi transformado em variável binária de acordo com a seguinte regra: valores 1,2 e 3 foram convertidos para 0 (zero), e os valores 4 e 5 foram convertidos para 1.



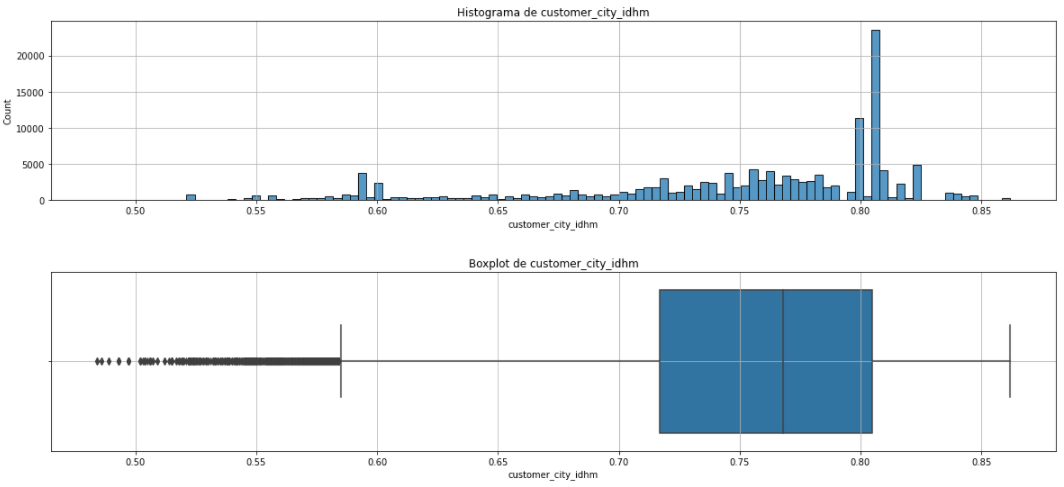
Estatistica descritiva do dataframe:

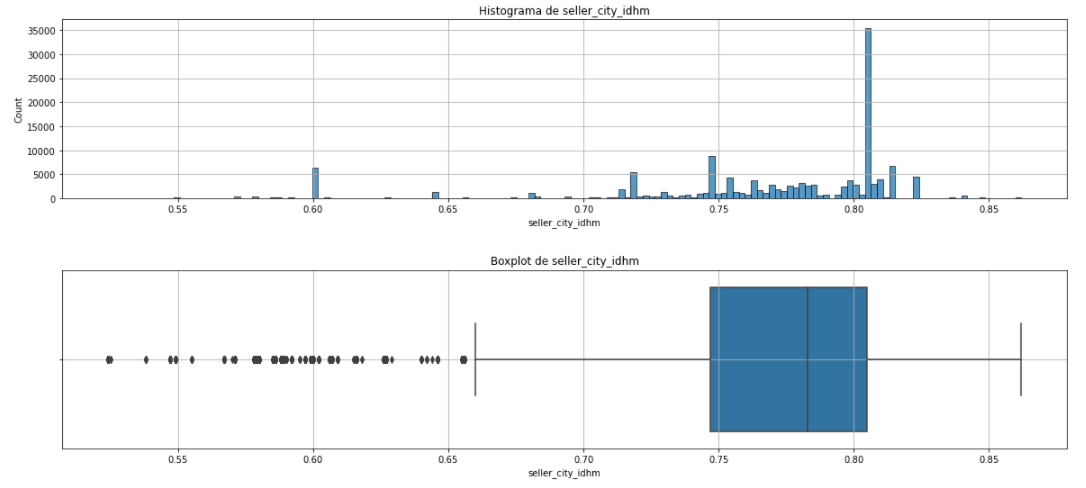


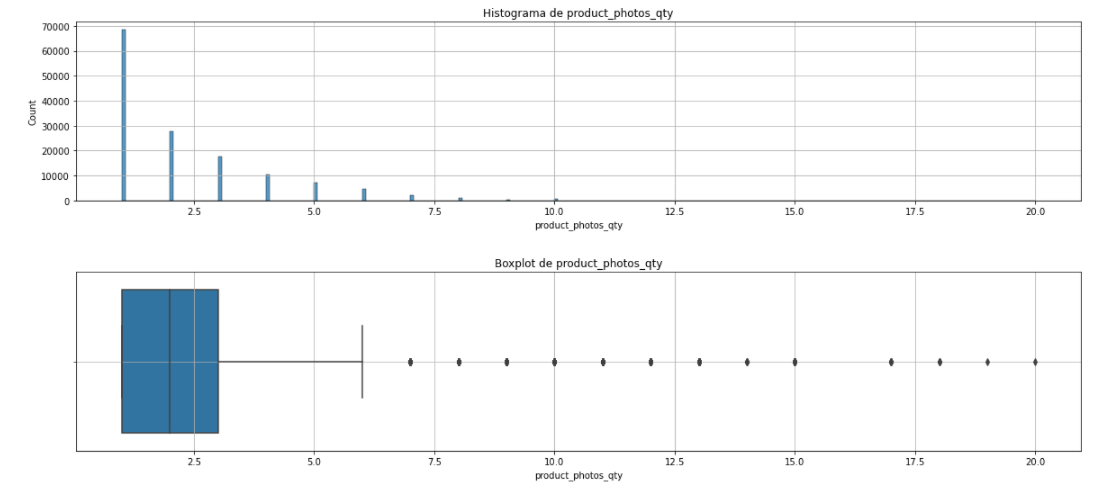
Observa-se que as variáveis “customer\_city\_idhm” e “seller\_city\_idhm” possuem valores próximos.

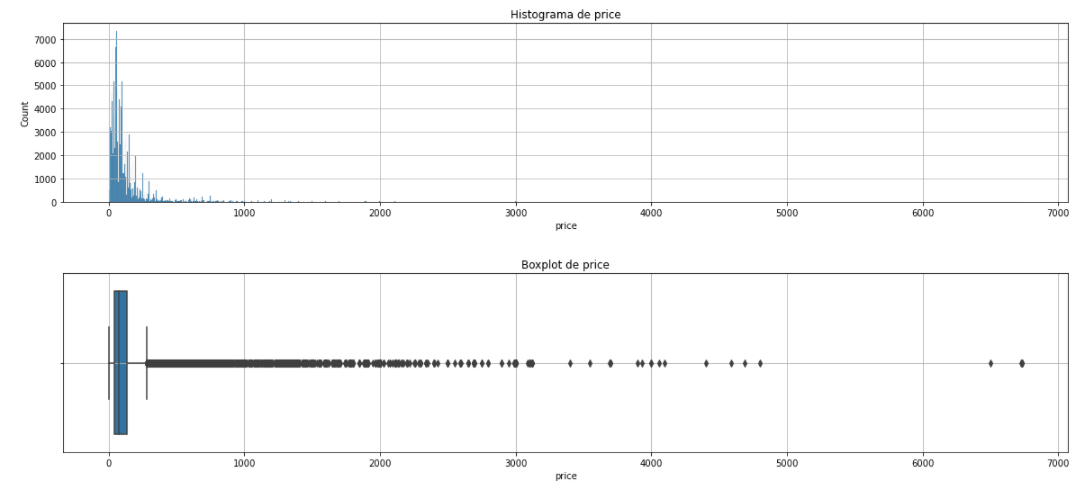
Distribuição das variáveis numéricas:

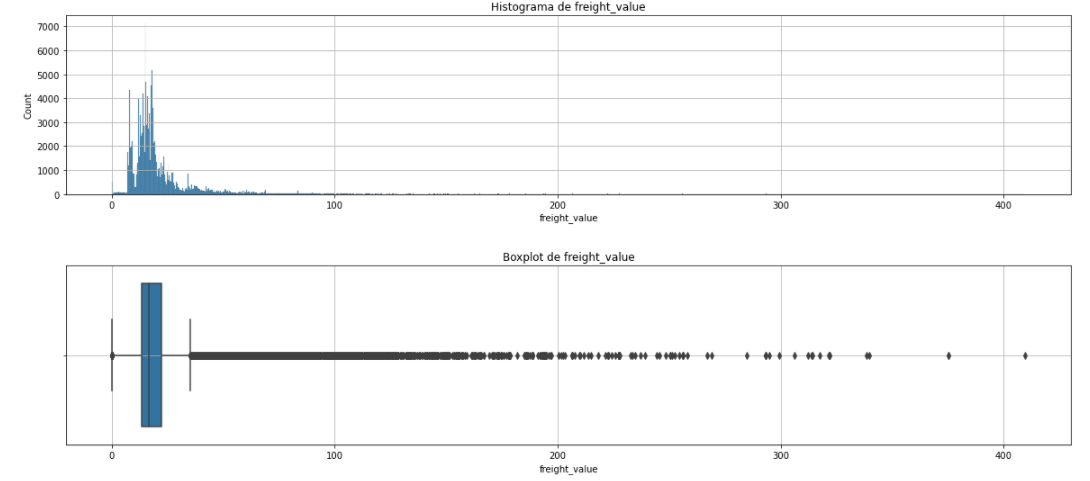


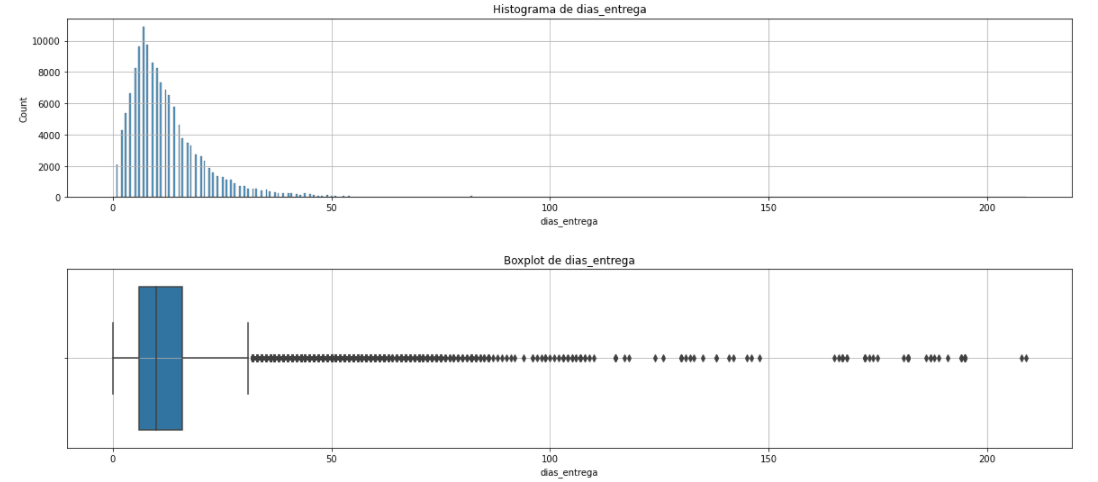






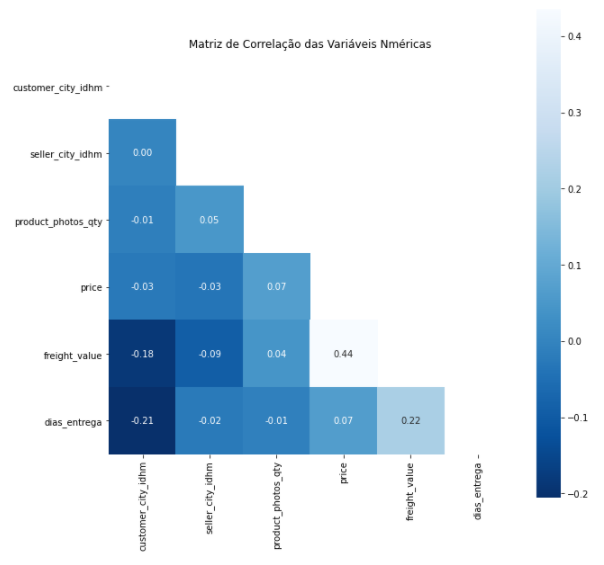






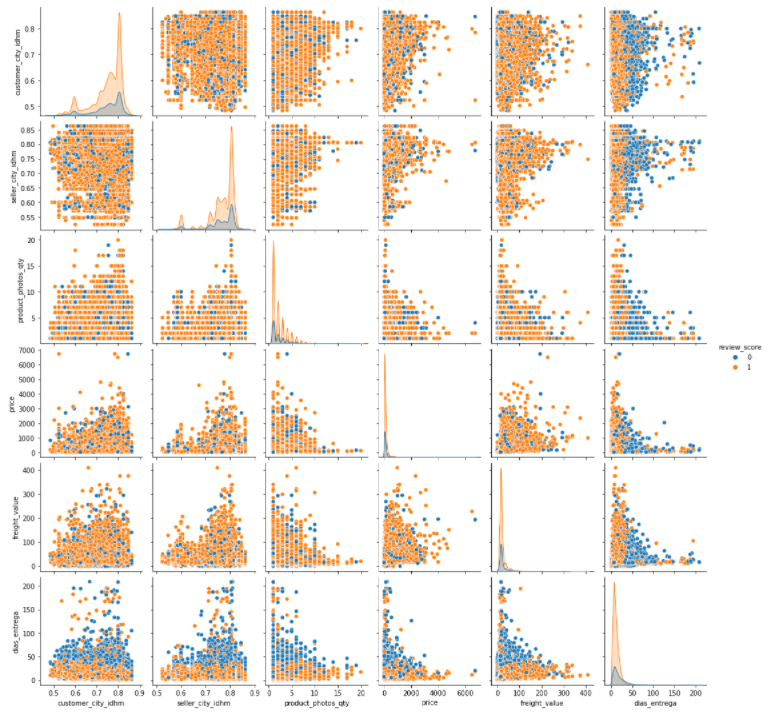
Os outliers não foram removidos porque os algoritimos de classificação utilizados são insensíveis aos outliers. Caso fossem removidos, no caso da arvore de decisão, a remoção pode gerar perda de informação.

Mariz de correlação das variáveis numéricas:



Pode-se obersar que as correlações entre as variáveis são pequenas. O par que apresentou maior correlação foi “price” e freight\_value”.

Matriz de dispersão das variáveis do dataframe. As duas cores representam a associação de cada elemento ao elemento da variável alvo.



Olhando para o gráfico pode-se dizer que a variável “dias\_entrega” possui um boa capacidade de separação na comparação com as outras variáveis.

# 5. Criação de Modelos de Machine Learning

Conforme o documento de instruções para o TCC, essa etapa é obrigatória. Nessa seção você irá descrever as ferramentas e algoritmos utilizados. Se você utilizou ferramentas visuais como Knime e Rapid Miner, coloque aqui um print do seu modelo e a descrição detalhada do workflow do seu modelo. Caso você tenha escrito scripts em Python, por exemplo, coloque aqui o seu script. Explique as *features* utilizadas, faça a comparação entre diferentes algoritmos/modelos, justifique a escolha por determinado modelo, os parâmetros utilizados, etc. Por fim, salienta-se que embora você possa utilizar ferramentas como KNIME e RapidMiner para testar protótipos do seu modelo de ML, encorajamos você a fazer seus modelos em Python ou R.

# 6. Apresentação dos Resultados

Nessa seção você deve apresentar os resultados obtidos. Apresente gráficos, *dashboards*, conte a sua história de forma bastante criativa. Aqui você pode utilizar os modelos de Canvas propostos por Dourard (clique [aqui](https://www.louisdorard.com/machine-learning-canvas)) ou por Vasandani (clique [aqui](https://towardsdatascience.com/a-data-science-workflow-canvas-to-kickstart-your-projects-db62556be4d0)).



# 7. Links

Aqui você deve disponibilizar os links para o vídeo com sua apresentação de 5 minutos e para o repositório contendo os dados utilizados no projeto, scripts criados, etc.

Link para o vídeo: youtube.com/...

Link para o repositório: github.com/...

# REFERÊNCIAS

Um projeto/relatório técnico de Ciência de Dados não requer revisão bibliográfica. Portanto, a inclusão das referências não é obrigatória. No entanto, caso você deseje incluir referências relacionadas às tecnologias ou às metodologias usadas em seu trabalho, relacione-as de acordo com o modelo a seguir.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

# APÊNDICE

**Programação/Scripts**

Cole aqui seus scripts em Python e/ou R.

**Gráficos**

Cole aqui workflows (KNIME, RapidMiner, etc), gráficos e figuras que você tenha gerado e não colocou no texto principal.

**Tabelas**

Cole aqui tabelas de dados que você tenha gerado e não colocou no texto principal.