Exploring Brazilian E-commerce Data

In this notebook, we are going to explore a dataset related to e-commerce sales in Brazil.

First, let's import the necessary Python libraries for data processing and visualization.

In [1]:
```
import sys, json
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import datetime as datetime
```

Next, we load a GeoJSON file that contains the map of Brazil's federal states.

In [2]:
```
geojson = json.load(open('geojson/brasil_estados.json'))
```

We now load various CSV files into pandas dataframes. Each file represents different aspects of the orders:

1. `olist_customers_dataset.csv` : Customer data.
2. `olist_order_items_dataset.csv` : Order items data.
3. `olist_order_payments_dataset.csv` : Payment data.
4. `olist_order_reviews_dataset.csv` : Customer reviews data.
5. `olist_orders_dataset.csv` : Orders data.
6. `olist_products_dataset.csv` : Products data.
7. `product_category_name_translation.csv` : Product categories translation data.

In [3]:
```
df0 = pd.read_csv('datasets/olist_customers_dataset.csv')
df2 = pd.read_csv('datasets/olist_order_items_dataset.csv')
df3 = pd.read_csv('datasets/olist_order_payments_dataset.csv')
df4 = pd.read_csv('datasets/olist_order_reviews_dataset.csv')
df5 = pd.read_csv('datasets/olist_orders_dataset.csv')
df6 = pd.read_csv('datasets/olist_products_dataset.csv')
df8 = pd.read_csv('datasets/product_category_name_translation.csv')
```

Now, we calculate and display the number of rows and columns in each dataframe to understand the size of our data.

In [4]:
```
shapes = []
dfs = [df0,df2,df3,df4,df5,df6,df8]
for d in dfs :
    shapes.append(d.shape)

shapes = pd.DataFrame(data = shapes)# , columns = {'Rows','Coloumns'})
shapes
```

Out[4]:

	0	1
0	99441	5
1	112650	7
2	103886	5
3	99224	7
4	99441	8
5	32951	9
6	71	2

In [5]:
```
print(df0.columns)
print(df2.columns)
print(df3.columns)
print(df4.columns)
print(df5.columns)
print(df6.columns)
print(df8.columns)
```

```
Index(['customer_id', 'customer_unique_id', 'customer_zip_code_prefix',
       'customer_city', 'customer_state'],
      dtype='object')
Index(['order_id', 'order_item_id', 'product_id', 'seller_id',
       'shipping_limit_date', 'price', 'freight_value'],
      dtype='object')
Index(['order_id', 'payment_sequential', 'payment_type',
       'payment_installments', 'payment_value'],
      dtype='object')
Index(['review_id', 'order_id', 'review_score', 'review_comment_title',
       'review_comment_message', 'review_creation_date',
       'review_answer_timestamp'],
      dtype='object')
Index(['order_id', 'customer_id', 'order_status', 'order_purchase_timestamp',
       'order_approved_at', 'order_delivered_carrier_date',
       'order_delivered_customer_date', 'order_estimated_delivery_date'],
      dtype='object')
Index(['product_id', 'product_category_name', 'product_name_lenght',
       'product_description_lenght', 'product_photos_qty', 'product_weight_g',
       'product_length_cm', 'product_height_cm', 'product_width_cm'],
      dtype='object')
Index(['product_category_name', 'product_category_name_english'], dtype='object')
```

In [6]:
```
df0 = df0[['customer_unique_id','customer_id','customer_zip_code_prefix','customer_city', 'customer_state'
df2 = df2[['order_id', 'order_item_id', 'product_id', 'price', 'freight_value']]
df3 = df3[['order_id', 'payment_installments', 'payment_value']]
df4 = df4[['order_id','review_score']]
df5 = df5[['order_id', 'customer_id', 'order_purchase_timestamp','order_delivered_customer_date']]
df6 = df6[['product_id', 'product_category_name']]
df8 = df8[['product_category_name', 'product_category_name_english']]
```

In [7]:
```
df = df0
del df0
df_orders = df2.merge(right=df3, on='order_id')
del df2,df3
df_orders = df_orders.merge(right=df4, on='order_id')
del df4
df_orders = df_orders.merge(right=df5, on='order_id')
del df5
df_orders = df_orders.merge(right=df6, on='product_id')
del df6
df_orders = df_orders.merge(right=df8, on='product_category_name')
del df8
df = df.merge(right=df_orders, on='customer_id')
del df_orders
df = df[['customer_unique_id','customer_city','customer_state','order_id','product_id',
         'price','payment_value','payment_installments','review_score','freight_value','order_purchase_tim
df.head()
df.shape
```

Out[7]: (115609, 12)

In [8]:
```
df.isna().mean()
```

Out[8]:
```
customer_unique_id              0.0
customer_city                   0.0
customer_state                  0.0
order_id                        0.0
product_id                      0.0
price                           0.0
payment_value                   0.0
payment_installments            0.0
review_score                    0.0
freight_value                   0.0
order_purchase_timestamp        0.0
product_category_name_english   0.0
dtype: float64
```

Recency, frequency, order amount

Order frequency by customer

In [9]:
```
recence=df[['customer_unique_id','order_purchase_timestamp']]
recence.sort_values(by='order_purchase_timestamp', ascending=False)
recence.head(20)
```

Out[9]:

	customer_unique_id	order_purchase_timestamp
0	861eff4711a542e4b93843c6dd7febb0	2017-05-16 15:05:35
1	290c77bc529b7ac935b93aa66c333dc3	2018-01-12 20:48:24
2	060e732b5b29e8181a18229c7b0b2b5e	2018-05-19 16:07:45
3	259dac757896d24d7702b9acbbff3f3c	2018-03-13 16:06:38
4	345ecd01c38d18a9036ed96c73b8d066	2018-07-29 09:51:30
5	4c93744516667ad3b8f1fb645a3116a4	2017-09-14 18:14:31
6	addec96d2e059c80c30fe6871d30d177	2018-02-19 14:38:35
7	1175e95fb47ddff9de6b2b06188f7e0d	2018-01-18 12:35:44
8	9afe194fb833f79e300e37e580171f22	2018-01-08 11:22:34
9	2a7745e1ed516b289ed9b29c7d0539a5	2017-11-27 17:23:20
10	2a46fb94aef5cbeeb850418118cee090	2018-02-07 11:36:42
11	918dc87cd72cd9f6ed4bd442ed785235	2017-09-09 09:54:57
12	295c05e81917928d76245e842748184d	2018-03-07 15:57:14
13	3151a81801c8386361b62277d7fa5ecf	2018-04-01 18:59:31
14	21f748a16f4e1688a9014eb3ee6fa325	2018-01-29 20:32:08
15	5c2991dbd08bbf3cf410713c4de5a0b5	2018-08-13 23:45:05
16	b6e99561fe6f34a55b0b7da92f8ed775	2018-06-18 13:34:21
17	3e6fd6b2f0d499456a6a6820a40f2d79	2017-11-01 21:54:10
18	e607ede0e63436308660236f5a52da5e	2017-08-13 10:03:36
19	a96d5cfa0d3181817e2b946f921ea021	2017-10-15 11:08:48

In [10]:
```
frequence=df[['customer_unique_id','order_purchase_timestamp']]
```

In [11]:
```
frequence=frequence.groupby('customer_unique_id').count().sort_values('order_purchase_timestamp',ascending
frequence=frequence.reset_index().rename(columns={"order_purchase_timestamp": "frequency"})
```

In [12]:
```
df = df.merge(right=frequence, on='customer_unique_id')
```

Last 20 orders

In [13]:
```
df['order_purchase_timestamp'] = df['order_purchase_timestamp'].astype('datetime64[ns]')
df['order_purchase_timestamp'] = pd.to_datetime(df['order_purchase_timestamp'], format = '%YY%mm%dd')
df['since_last_purchase'] = (datetime.datetime.now() - df['order_purchase_timestamp'])
df['since_last_purchase']=df['since_last_purchase'].astype('timedelta64[D]')
df.sort_values('since_last_purchase').head(20)
```

Out[13]:

	customer_unique_id	customer_city	customer_state	order_id	produc
49668	ff22e30958c13ffe219db7d711e8f564	sao paulo	SP	54282e97f61c23b78330c15b154c867d	b98992ea80b467987a7fbb88e7f2(
35351	fb7e29c65321441231990afc201c1b14	sao paulo	SP	d4fae577806d683110e00e18a5e181be	59536eeb13a52ef81966f81b91fb4
86119	83176537f63ef9c7510572006c85ac50	indaiatuba	SP	cb9628da43a976b479b45230984a05b8	2a34e0af5f72ca6cdeb148377a247
99792	9dfcc502727549f99c9f73dbd28b35e9	sao paulo	SP	e7c290bfc31d7eed478c3d3d2d4d2953	44406b87e5ac6494cdb0c9dccd3b8
70705	0421e7a23f21e5d54efed456aedbc513	salto	SP	d03ca98f59480e7e76c71fa83ecd8fb6	06601c3059e35a3bf65e72f2fd2ac
99964	21dbe8eabd00b34492a939c540e2b1a7	sao paulo	SP	d70442bc5e3cb7438da497cc6a210f80	9a8706b8c060b16e5f0d2925f20bc
35352	fb7e29c65321441231990afc201c1b14	sao paulo	SP	d4fae577806d683110e00e18a5e181be	59536eeb13a52ef81966f81b91fb4
105181	afbcfd0b9c5233e7ccc73428526fbb52	sao jose dos campos	SP	bee12e8653a04e76786e8891cfb6330a	8d4dac6177fb8134f26fb4c5cc6c
58603	0c6d7218d5f3fa14514fd29865269993	sao bernardo do campo	SP	912859fef5a0bd5059b6d48fa79d121a	9865c67a74684715521d1e70226cc
15653	1041688b50cfd8ef6df6086a1746a30c	petropolis	RJ	3064071cf67a2cc381cd53b13055eac5	15de022edf1005363381e66bed514
35353	fb7e29c65321441231990afc201c1b14	sao paulo	SP	d4fae577806d683110e00e18a5e181be	59536eeb13a52ef81966f81b91fb4
35354	fb7e29c65321441231990afc201c1b14	sao paulo	SP	d4fae577806d683110e00e18a5e181be	7001d71d1ad858e07e5a341649412
624	7febafa06d9d8f232a900a2937f04338	paracatu	MG	168626408cb32af0ffaf76711caae1dc	bdcf6a834e8faa30dac3886c7a58e
99791	9dfcc502727549f99c9f73dbd28b35e9	sao paulo	SP	e7c290bfc31d7eed478c3d3d2d4d2953	44406b87e5ac6494cdb0c9dccd3b8
94495	ca36d819c1759cdb6257fef0bb5d362d	sao paulo	SP	0ac69790e2a6e4c075edbd78648a3594	3f0602cdcad0ac24d72b038103a37
110671	f80013faf776e37bcea7634d59c2181e	sao paulo	SP	c84d88553f9878bf2c7ecda2eb211ece	24bc2932a12c983f8e76d828b65cf
105184	afbcfd0b9c5233e7ccc73428526fbb52	sao jose dos campos	SP	bee12e8653a04e76786e8891cfb6330a	8d4dac6177fb8134f26fb4c5cc6c
105183	afbcfd0b9c5233e7ccc73428526fbb52	sao jose dos campos	SP	bee12e8653a04e76786e8891cfb6330a	8d4dac6177fb8134f26fb4c5cc6c
77701	55cfdb1ec3c5bf60d9ccc0d5f276f8a9	sao paulo	SP	986983dd21e5167fac109fadaf7aac16	02b750ee6f6f12a23c311b775ee44
82624	a712a430955027da5bc257a10073a390	osasco	SP	dbb786f88b6d4e52fe3cb5d771b979d6	39a1a3b9314738724fce4b2907ae6

Order volume by customer and by state

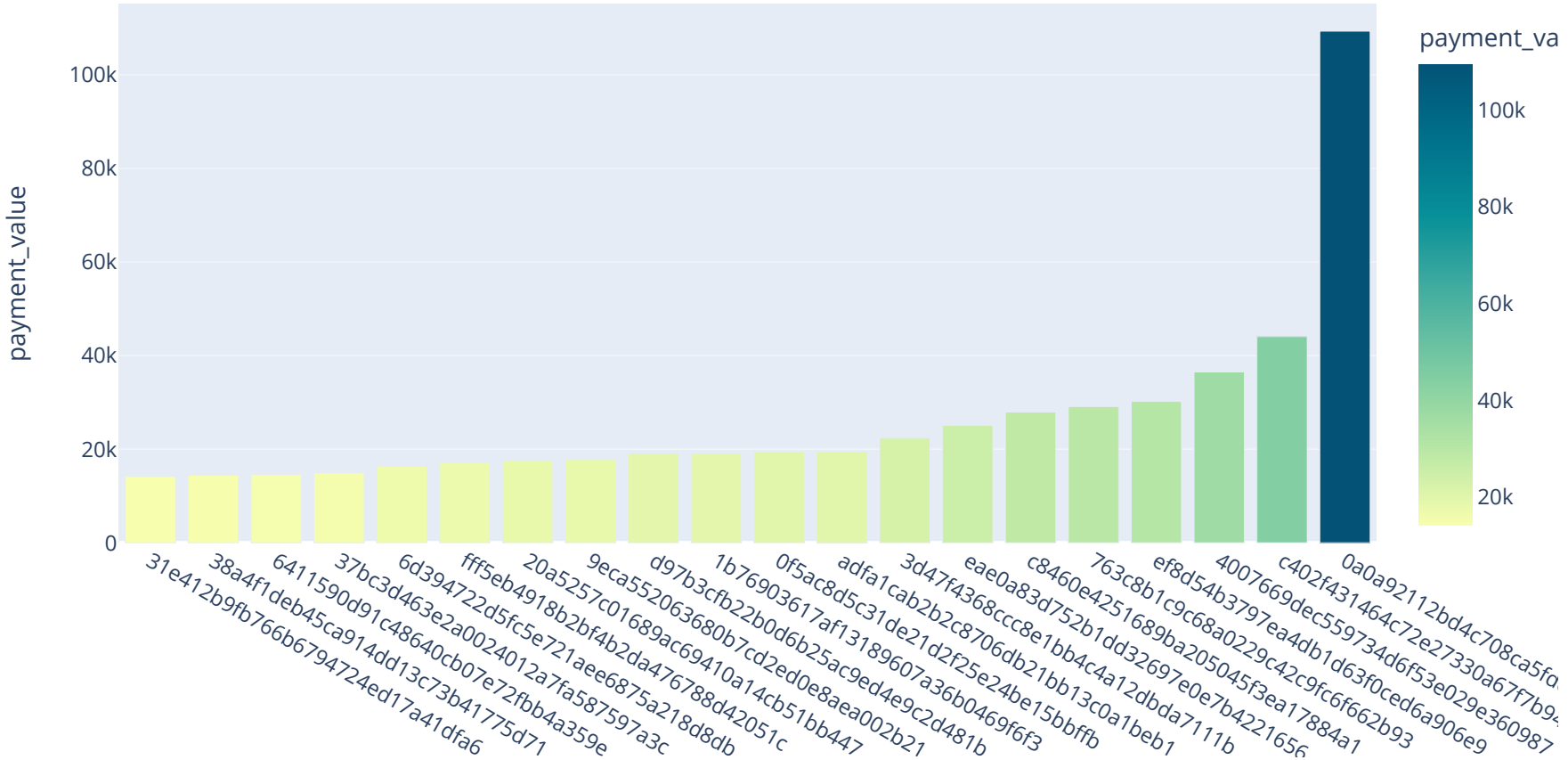
By Customer

In [14]:
```
commandes_client=df[['customer_unique_id', 'payment_value','customer_state','review_score']].groupby(by='c
commandes_client = commandes_client.sort_values('payment_value', ascending = False)
commandes_client.head(20)
```

Out[14]:

	customer_unique_id	payment_value	review_score
3724	0a0a92112bd4c708ca5fde585afaa872	109312.64	1.0
71634	c402f431464c72e27330a67f7b94d4fb	44048.00	1.0
23445	4007669dec559734d6f53e029e360987	36489.24	1.0
87489	ef8d54b3797ea4db1d63f0ced6a906e9	30186.00	5.0
43207	763c8b1c9c68a0229c42c9fc6f662b93	29099.52	1.0
73191	c8460e4251689ba205045f3ea17884a1	27935.46	4.0
85799	eae0a83d752b1dd32697e0e7b4221656	25051.89	3.0
22441	3d47f4368ccc8e1bb4c4a12dbda7111b	22346.60	1.0
63592	adfa1cab2b2c8706db21bb13c0a1beb1	19457.04	3.0
5631	0f5ac8d5c31de21d2f25e24be15bbffb	19342.26	5.0
10099	1b76903617af13189607a36b0469f6f3	19174.38	1.0
79443	d97b3cfb22b0d6b25ac9ed4e9c2d481b	19167.26	1.0
58029	9eca552063680b7cd2ed0e8aea002b21	17786.88	1.0
11984	20a5257c01689ac69410a14cb51bb447	17671.00	1.0
93384	fff5eb4918b2bf4b2da476788d42051c	17069.76	5.0
39970	6d394722d5fc5e721aee6875a218d8db	16313.60	4.0
20425	37bc3d463e2a0024012a7fa587597a3c	14963.64	4.0
36636	6411590d91c48640cb07e72fbb4a359e	14577.57	1.0
20772	38a4f1deb45ca914dd13c73b41775d71	14401.00	1.0
18291	31e412b9fb766b6794724ed17a41dfa6	14196.28	1.0

In [15]:
```
fig_bar = px.bar(commandes_client.head(20), y='payment_value', x='customer_unique_id',text='customer_uniqu
             color= 'payment_value',color_continuous_scale = 'bluyl')

fig_bar.update_traces(texttemplate='%{text:.2s %}', textposition='outside')
fig_bar.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')
fig_bar.update_layout(xaxis={'categoryorder':'total ascending'})

fig_bar.show()
```



By state

In [16]:
```
commandes_etat=df[['customer_state', 'payment_value','review_score']].groupby(by="customer_state").agg({'p
commandes_etat = commandes_etat.sort_values('payment_value', ascending = False)
commandes_etat.head(20)
```

Out[16]:

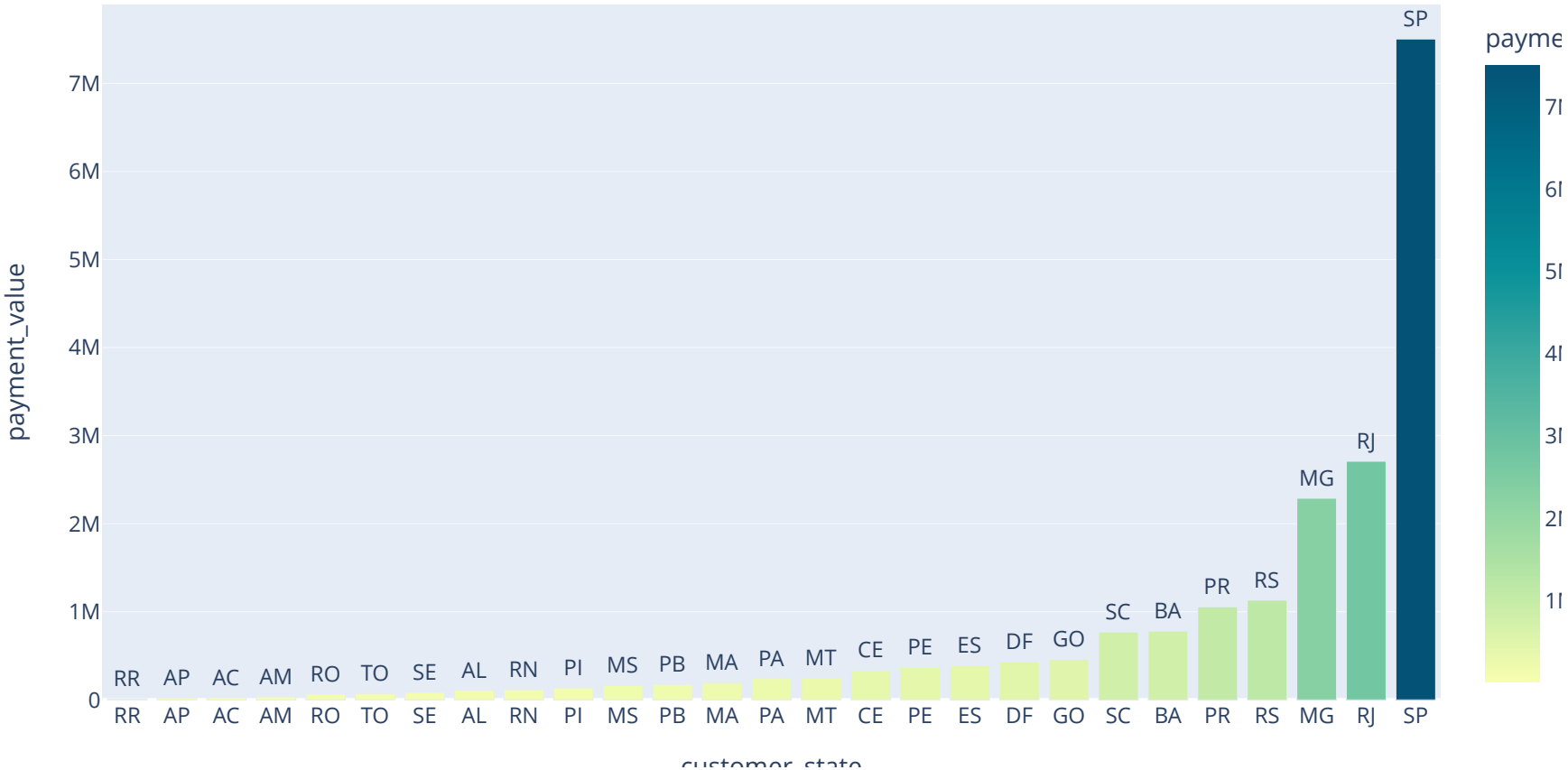
	customer_state	payment_value	review_score
25	SP	7502926.95	4.130910
18	RJ	2708839.33	3.817108
10	MG	2288949.71	4.086231
22	RS	1131899.22	4.044129
17	PR	1055747.81	4.105970
4	BA	780334.54	3.823440
23	SC	769744.94	4.007587
8	GO	459466.06	3.988554
6	DF	430126.51	3.998775
7	ES	390840.58	3.989565
15	PE	368309.23	3.972988
5	CE	337096.92	3.835625
12	MT	253916.03	3.894213
13	PA	241870.17	3.793710
9	MA	195954.74	3.686298
14	PB	175060.52	3.993538
11	MS	165068.30	4.078107
16	PI	135063.84	3.850267
19	RN	114849.98	4.051786
1	AL	109333.74	3.716484

In [17]:
```
commandes_etat.head()
```

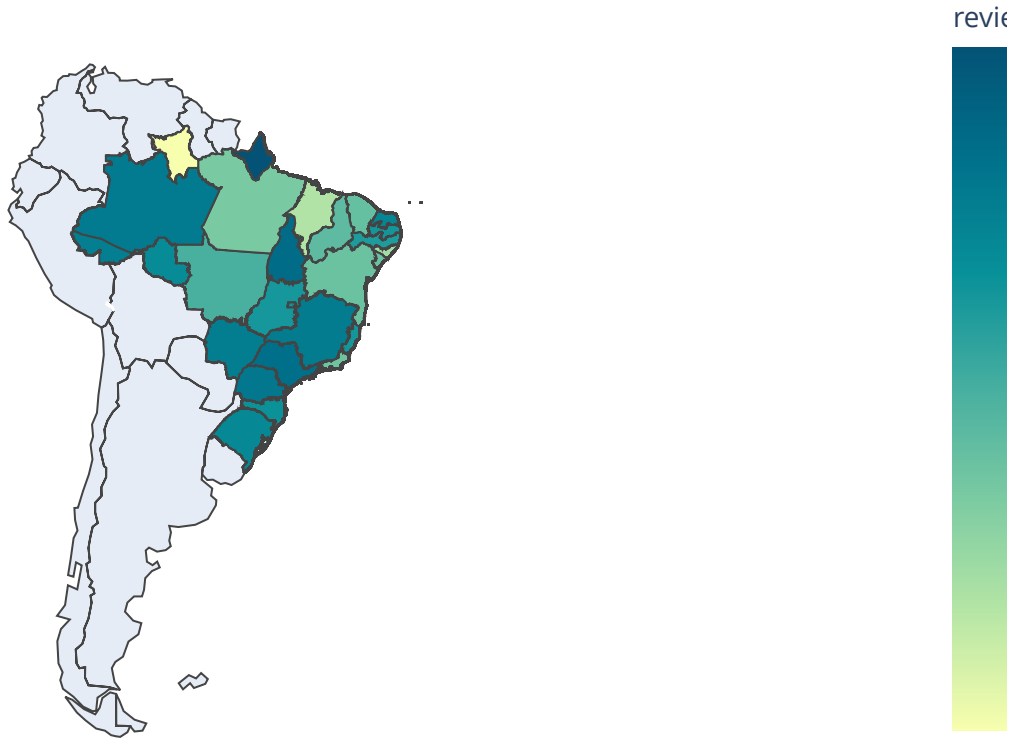
Out[17]:

	customer_state	payment_value	review_score
25	SP	7502926.95	4.130910
18	RJ	2708839.33	3.817108
10	MG	2288949.71	4.086231
22	RS	1131899.22	4.044129
17	PR	1055747.81	4.105970

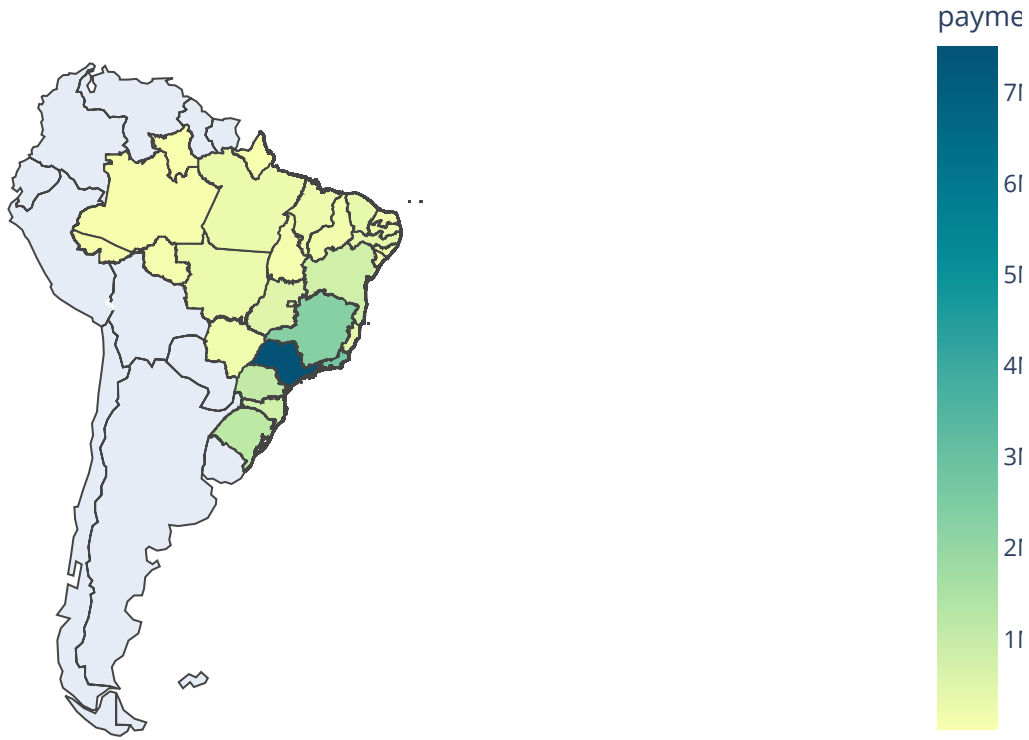
In [18]:
```
fig_bar = px.bar(commandes_etat, y='payment_value', x='customer_state' ,text='customer_state',
             color='payment_value',color_continuous_scale = 'bluyl')

fig_bar.update_traces(texttemplate='%{text:.2s %}', textposition='outside')
fig_bar.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')
fig_bar.update_layout(xaxis={'categoryorder':'total ascending'})

fig_bar.show()
```



In [28]:
```
fig_choropleth = px.choropleth(commandes_etat, geojson=geojson, locations='customer_state', color='review_
                               color_continuous_scale="bluyl",
                               scope='south america'
                              )
fig_choropleth.show()
```



In [20]:
```
fig_choropleth = px.choropleth(commandes_etat, geojson=geojson, locations='customer_state', color='payment
                               color_continuous_scale="bluyl",
                               scope='south america'
                              )
fig_choropleth.show()
```



Stars

He we define "Stars" as a combination. First customers are ordered by payment value, and then by frequency. The last order must be recent (at least january 2018). Thus the most important criterion after the recency is the total amount and then the frequency of purchases.

In [21]:
```
stars=df[['customer_unique_id', 'payment_value','frequency','since_last_purchase']]
#stars=stars[stars['since_last_purchase'] < 1770] #depuis janvier 2018
stars=stars.groupby(by="customer_unique_id").agg({'payment_value':'sum'}).reset_index()
stars = stars.merge(right=frequence, on='customer_unique_id')
stars = stars.sort_values(['payment_value','frequency'], ascending = [False,False])
stars['average_purchase_value']=np.round(stars['payment_value']/stars['frequency'],2)
stars.head(5)
```

Out[21]:

	customer_unique_id	payment_value	frequency	average_purchase_value
3724	0a0a92112bd4c708ca5fde585afaa872	109312.64	8	13664.08
71634	c402f431464c72e27330a67f7b94d4fb	44048.00	20	2202.40
23445	4007669dec559734d6f53e029e360987	36489.24	6	6081.54
87489	ef8d54b3797ea4db1d63f0ced6a906e9	30186.00	10	3018.60
43207	763c8b1c9c68a0229c42c9fc6f662b93	29099.52	4	7274.88

Preparation of the data for clustering

As the clustering algorithm only works with numbers we transform each individual value in the table as a "category code" same values thus have same numerical values thus can be used in the clustering. We do the same for datetimes.

In [22]:
```
df['customer_unique_id'] = pd.Categorical(df['customer_unique_id'])
df['customer_code'] = df['customer_unique_id'].cat.codes

df['customer_city'] = pd.Categorical(df['customer_city'])
df['city_code'] = df['customer_city'].cat.codes

df['customer_state'] = pd.Categorical(df['customer_state'])
df['state_code'] = df['customer_state'].cat.codes

df['order_id'] = pd.Categorical(df['order_id'])
df['order_code'] = df['order_id'].cat.codes

df['product_id'] = pd.Categorical(df['product_id'])
df['product_code'] = df['product_id'].cat.codes

df['product_id'] = pd.Categorical(df['product_id'])
df['product_code'] = df['product_id'].cat.codes

df['order_purchase_timestamp_str'] = df['order_purchase_timestamp']
df['order_purchase_timestamp'] = pd.to_datetime(df['order_purchase_timestamp']).astype(np.int64)

df['product_category_name_english'] = pd.Categorical(df['product_category_name_english'])
df['product_category_code'] = df['product_category_name_english'].cat.codes

df.head()
```

Out[22]:

	customer_unique_id	customer_city	customer_state	order_id	product_id
0	861eff4711a542e4b93843c6dd7febb0	franca	SP	00e7ee1b050b8499577073aeb2a297a1	a9516a079e37a9c9c36b9b78b10169e8
1	290c77bc529b7ac935b93aa66c333dc3	sao bernardo do campo	SP	29150127e6685892b6eab3eec79f59c7	4aa6014eceb682077f9dc4bffebc05b0
2	060e732b5b29e8181a18229c7b0b2b5e	sao paulo	SP	b2059ed67ce144a36e2aa97d2c9e9ad2	bd07b66896d6f1494f5b86251848ced7
3	259dac757896d24d7702b9acbbff3f3c	mogi das cruzes	SP	951670f92359f4fe4a63112aa7306eba	a5647c44af977b148e0a3a4751a09e2e
4	345ecd01c38d18a9036ed96c73b8d066	campinas	SP	6b7d50bd145f6fc7f33cebabd7e49d0f	9391a573abe00141c56e38d84d7d5b3b

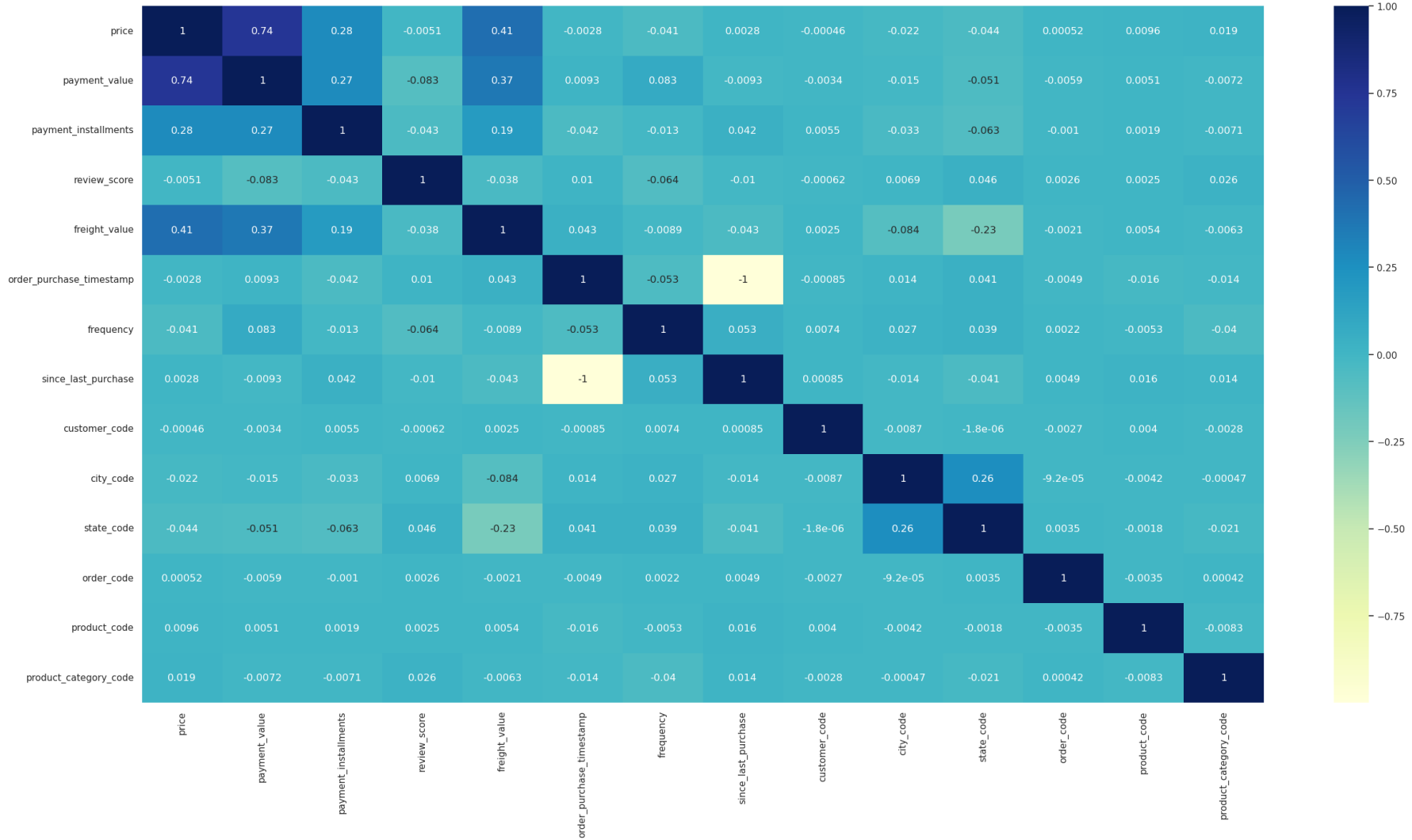
5 rows × 21 columns

We also check for correlations, having features with too much correlation will negatively influence the performance of the clustering.

In [23]:
```
sns.set(rc = {'figure.figsize':(30,15)})
sns.heatmap(df.corr(), annot=True, cmap='YlGnBu')
```

```
/tmp/ipykernel_3809/3736932318.py:2: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default t
o False. Select only valid columns or specify the value of numeric_only to silence this warning.
```

Out[23]:
```
<Axes: >
```



In [24]:
```
df_final=df[['order_purchase_timestamp_str','order_purchase_timestamp','customer_code',
    'price','payment_value','review_score','freight_value','frequency','since_last_purchase']]
```

We now export the data to be processed by the following notebook

In [25]:
```
del df
```

In [26]:
```
df_final.to_csv('datasets/cleaned.csv', index = False, header=True)
```