In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline


sns.set(color_codes=True)
```

Data Preparation and Preprocessing

In [2]:
```
client_df = pd.read_csv('client_data.csv')
price_df = pd.read_csv('price_data.csv')
```

Analysis of data types, data statistics, specific parameters, and variable distributions.

In [3]:
```
client_df.head(5)
```

Out[3]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	date_activ	date_en
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15	2016-06 1
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0	2009-08-21	2016-08 3
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0	2010-04-16	2016-04 1
3	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0	0	2010-03-30	2016-03 3
4	149d57cf92fc41cf94415803a877cb4b	MISSING	4425	0	526	2010-01-13	2016-03 0

5 rows × 26 columns

In [4]:
```
client_df['id'].unique
```

Out[4]:
```
<bound method Series.unique of 0        24011ae4ebbe3035111d65fa7c15bc57
1        d29c2c54acc38ff3c0614d0a653813dd
2        764c75f661154dac3a6c254cd082ea7d
3        bba03439a292a1e166f80264c16191cb
4        149d57cf92fc41cf94415803a877cb4b
                       ...
14601    18463073fb097fc0ac5d3e040f356987
14602    d0a6f71671571ed83b2645d23af6de00
14603    10e6828ddd62cbcf687cb74928c4c2d2
14604    1cf20fd6206d7678d5bcafd28c53b4db
14605    563dde550fd624d7352f3de77c0cdfcd
Name: id, Length: 14606, dtype: object>
```

In [5]:
```
price_df.head(5)
```

Out[5]:

	id	price_date	price_off_peak_var	price_peak_var	price_mid_peak_var	price_off_peak_fix	price_peak_
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	44.266931	
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	44.266931	
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	0.0	44.266931	
3	038af19179925da21a25619c5a24b745	2015-04-01	0.149626	0.0	0.0	44.266931	
4	038af19179925da21a25619c5a24b745	2015-05-01	0.149626	0.0	0.0	44.266931	

In [6]:
```
client_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14606 entries, 0 to 14605
Data columns (total 26 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              14606 non-null  object 
 1   channel_sales                   14606 non-null  object 
 2   cons_12m                        14606 non-null  int64  
 3   cons_gas_12m                    14606 non-null  int64  
 4   cons_last_month                 14606 non-null  int64  
 5   date_activ                      14606 non-null  object 
 6   date_end                        14606 non-null  object 
 7   date_modif_prod                 14606 non-null  object 
 8   date_renewal                    14606 non-null  object 
 9   forecast_cons_12m               14606 non-null  float64
 10  forecast_cons_year              14606 non-null  int64  
 11  forecast_discount_energy        14606 non-null  float64
 12  forecast_meter_rent_12m         14606 non-null  float64
 13  forecast_price_energy_off_peak  14606 non-null  float64
 14  forecast_price_energy_peak      14606 non-null  float64
 15  forecast_price_pow_off_peak     14606 non-null  float64
 16  has_gas                         14606 non-null  object 
 17  imp_cons                        14606 non-null  float64
 18  margin_gross_pow_ele            14606 non-null  float64
 19  margin_net_pow_ele              14606 non-null  float64
 20  nb_prod_act                     14606 non-null  int64  
 21  net_margin                      14606 non-null  float64
 22  num_years_antig                 14606 non-null  int64  
 23  origin_up                       14606 non-null  object 
 24  pow_max                         14606 non-null  float64
 25  churn                           14606 non-null  int64  
dtypes: float64(11), int64(7), object(8)
memory usage: 2.9+ MB
```

In [7]:
```
client_df.describe()
```

Out[7]:

	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_12m	forecast_cons_year	forecast_discount_energy	forecast_meter_r
count	1.460600e+04	1.460600e+04	14606.000000	14606.000000	14606.000000	14606.000000	1460€
mean	1.592203e+05	2.809238e+04	16090.269752	1868.614880	1399.762906	0.966726	6:
std	5.734653e+05	1.629731e+05	64364.196422	2387.571531	3247.786255	5.108289	6(
min	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	(
25%	5.674750e+03	0.000000e+00	0.000000	494.995000	0.000000	0.000000	1(
50%	1.411550e+04	0.000000e+00	792.500000	1112.875000	314.000000	0.000000	1{
75%	4.076375e+04	0.000000e+00	3383.000000	2401.790000	1745.750000	0.000000	13'
max	6.207104e+06	4.154590e+06	771203.000000	82902.830000	175375.000000	30.000000	59!

In [8]:
```
price_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193002 entries, 0 to 193001
Data columns (total 8 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   id                  193002 non-null  object 
 1   price_date          193002 non-null  object 
 2   price_off_peak_var  193002 non-null  float64
 3   price_peak_var      193002 non-null  float64
 4   price_mid_peak_var  193002 non-null  float64
 5   price_off_peak_fix  193002 non-null  float64
 6   price_peak_fix      193002 non-null  float64
 7   price_mid_peak_fix  193002 non-null  float64
dtypes: float64(6), object(2)
memory usage: 11.8+ MB
```

In [9]:
```
price_df.describe()
```

Out[9]:

	price_off_peak_var	price_peak_var	price_mid_peak_var	price_off_peak_fix	price_peak_fix	price_mid_peak_fix
count	193002.000000	193002.000000	193002.000000	193002.000000	193002.000000	193002.000000
mean	0.141027	0.054630	0.030496	43.334477	10.622875	6.409984
std	0.025032	0.049924	0.036298	5.410297	12.841895	7.773592
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.125976	0.000000	0.000000	40.728885	0.000000	0.000000
50%	0.146033	0.085483	0.000000	44.266930	0.000000	0.000000
75%	0.151635	0.101673	0.072558	44.444710	24.339581	16.226389
max	0.280700	0.229788	0.114102	59.444710	36.490692	17.458221

I Merged Data sets for easy Access

In [10]:
```
c=client_df.merge(price_df, how ='inner', on=['id'])
c
```

Out[10]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	date_activ	date_
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15	2016
1	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15	2016
2	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15	2016
3	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15	2016
4	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15	2016
...	...	...	...	...	...	...	
175144	563dde550fd624d7352f3de77c0cdfcd	MISSING	8730	0	0	2009-12-18	2016
175145	563dde550fd624d7352f3de77c0cdfcd	MISSING	8730	0	0	2009-12-18	2016
175146	563dde550fd624d7352f3de77c0cdfcd	MISSING	8730	0	0	2009-12-18	2016
175147	563dde550fd624d7352f3de77c0cdfcd	MISSING	8730	0	0	2009-12-18	2016
175148	563dde550fd624d7352f3de77c0cdfcd	MISSING	8730	0	0	2009-12-18	2016

175149 rows × 33 columns

In [11]:
```
c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175149 entries, 0 to 175148
Data columns (total 33 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   id                              175149 non-null  object 
 1   channel_sales                   175149 non-null  object 
 2   cons_12m                        175149 non-null  int64  
 3   cons_gas_12m                    175149 non-null  int64  
 4   cons_last_month                 175149 non-null  int64  
 5   date_activ                      175149 non-null  object 
 6   date_end                        175149 non-null  object 
 7   date_modif_prod                 175149 non-null  object 
 8   date_renewal                    175149 non-null  object 
 9   forecast_cons_12m               175149 non-null  float64
 10  forecast_cons_year              175149 non-null  int64  
 11  forecast_discount_energy        175149 non-null  float64
 12  forecast_meter_rent_12m         175149 non-null  float64
 13  forecast_price_energy_off_peak  175149 non-null  float64
 14  forecast_price_energy_peak      175149 non-null  float64
 15  forecast_price_pow_off_peak     175149 non-null  float64
 16  has_gas                         175149 non-null  object 
 17  imp_cons                        175149 non-null  float64
 18  margin_gross_pow_ele            175149 non-null  float64
 19  margin_net_pow_ele              175149 non-null  float64
 20  nb_prod_act                     175149 non-null  int64  
 21  net_margin                      175149 non-null  float64
 22  num_years_antig                 175149 non-null  int64  
 23  origin_up                       175149 non-null  object 
 24  pow_max                         175149 non-null  float64
 25  churn                           175149 non-null  int64  
 26  price_date                      175149 non-null  object 
 27  price_off_peak_var              175149 non-null  float64
 28  price_peak_var                  175149 non-null  float64
 29  price_mid_peak_var              175149 non-null  float64
 30  price_off_peak_fix              175149 non-null  float64
 31  price_peak_fix                  175149 non-null  float64
 32  price_mid_peak_fix              175149 non-null  float64
dtypes: float64(17), int64(7), object(9)
memory usage: 44.1+ MB
```

In [12]:
```
c.describe()
```

Out[12]:

	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_12m	forecast_cons_year	forecast_discount_energy	forecast_meter_r
count	1.751490e+05	1.751490e+05	175149.000000	175149.000000	175149.000000	175149.000000	17514
mean	1.592606e+05	2.808072e+04	16095.518404	1868.343884	1399.782380	0.967028	63
std	5.735413e+05	1.629400e+05	64376.741908	2387.560169	3248.331276	5.109025	66
min	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0
25%	5.674000e+03	0.000000e+00	0.000000	494.980000	0.000000	0.000000	16
50%	1.411500e+04	0.000000e+00	792.000000	1112.610000	314.000000	0.000000	18
75%	4.076300e+04	0.000000e+00	3383.000000	2400.350000	1745.000000	0.000000	13
max	6.207104e+06	4.154590e+06	771203.000000	82902.830000	175375.000000	30.000000	59

8 rows × 24 columns

Exploratory Data Analysis

Analysis and Visualisation of overall Churn

In [13]:

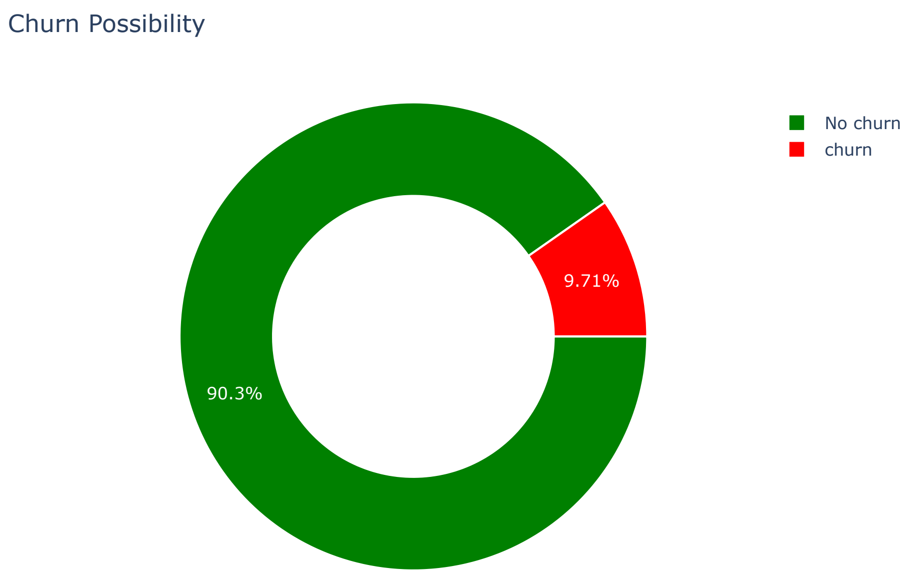
```
churntotal=c['churn'].value_counts()
churntotal
```

Out[13]:

```
churn
0    158146
1     17003
Name: count, dtype: int64
```

In [14]:

```
import plotly.graph_objects as go

plot_data=[
    go.Pie(
        labels=('No churn','churn'),
        values=churntotal,
        marker=dict(colors=["Green","Red"],
                     line=dict(color="white",
                               width=1.5)),

        rotation=90,
        hoverinfo= 'label+value+text',
        hole=.6)
]

# Create layout
plot_layout = go.Layout(dict(title='Churn Possibility'))

fig = go.Figure(data=plot_data, layout=plot_layout)

# Show the figure
fig.show()
```



Churned customers consitute 9.71% of the data

In [15]:
```
def plot_stacked_bars(dataframe, title_, size_=(18, 10), rot_=0, legend_="upper right"):
    """
    Plot stacked bars with annotations
    """
    ax = dataframe.plot(
        kind="bar",
        stacked=True,
        figsize=size_,
        rot=rot_,
        style="Greengrid",
        title=title_
    )

    # Annotate bars
    annotate_stacked_bars(ax, textsize=14)
    # Rename legend
    plt.legend(["Retention", "Churn"], loc=legend_)
    # Labels
    plt.ylabel("Company base (%)")
    plt.show()

def annotate_stacked_bars(ax, pad=0.99, colour="white", textsize=13):
    """
    Add value annotations to the bars
    """

    # Iterate over the plotted rectanges/bars
    for p in ax.patches:

        # Calculate annotation
        value = str(round(p.get_height(),1))
        # If value is 0 do not annotate
        if value == '0.0':
            continue
        ax.annotate(
            value,
            ((p.get_x()+ p.get_width()/2)*pad-0.05, (p.get_y()+p.get_height()/2)*pad),
            color=colour,
            size=textsize
        )

def plot_distribution(dataframe, column, ax, bins_=50):
    """
    Plot variable distirbution in a stacked histogram of churned or retained company
    """
    # Create a temporal dataframe with the data to be plot
    temp = pd.DataFrame({"Retention": dataframe[dataframe["churn"]==0][column],
    "Churn":dataframe[dataframe["churn"]==1][column]})
    # Plot the histogram
    temp[["Retention","Churn"]].plot(kind='hist', bins=bins_, ax=ax, stacked=True)
    # X-axis label
    ax.set_xlabel(column)
    # Change the x-axis to plain style
    ax.ticklabel_format(style='red', axis='x')
```

Analysis, Grouping, Visualization on The diffferent data parameters to identify their churn rate/possiblity

In [16]:
```
channel = client_df[['id', 'channel_sales', 'churn']]
channel = channel.groupby([channel['channel_sales'], channel['churn']])['id'].count().unstack(level=1).fillna(0)
channel_churn = (channel.div(channel.sum(axis=1), axis=0) * 100).sort_values(by=[1], ascending=False)
```

In [17]:
```
channel
```

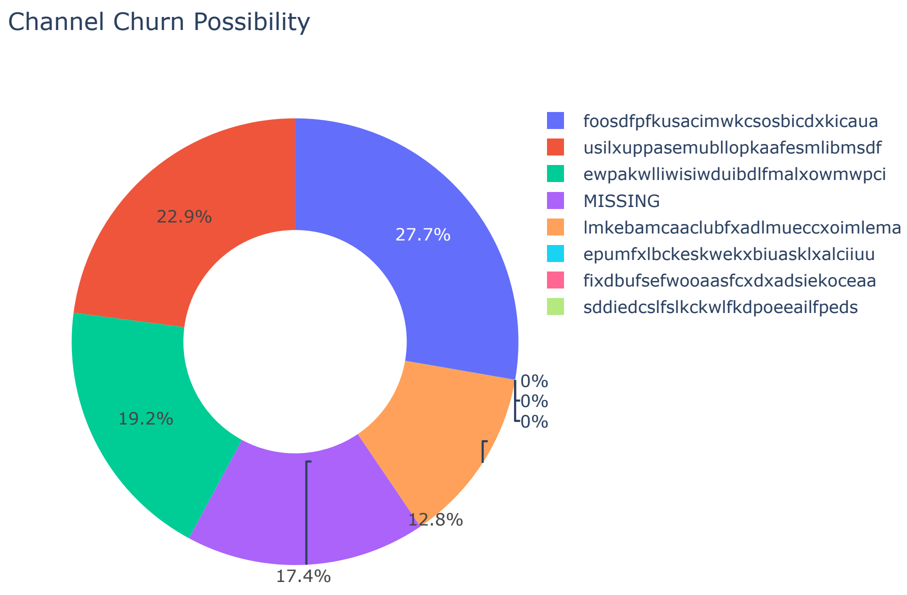
Out[17]:

churn	0	1
channel_sales		
MISSING	3442.0	283.0
epumfxlbckeskwekxbiuasklxalciiuu	3.0	0.0
ewpakwlliwisiwduibdlfmalxowmwpci	818.0	75.0
fixdbufsefwooaasfcxdxadsiekoceaa	2.0	0.0
foosdfpfkusacimwkcsosbicdxkicaua	5934.0	820.0
lmkebamcaaclubfxadlmueccxoimlema	1740.0	103.0
sddiedcslfslkckwlfkdpoeeailfpeds	11.0	0.0
usilxuppasemubllopkaafesmlibmsdf	1237.0	138.0

Churned Customers from the Different sale channels

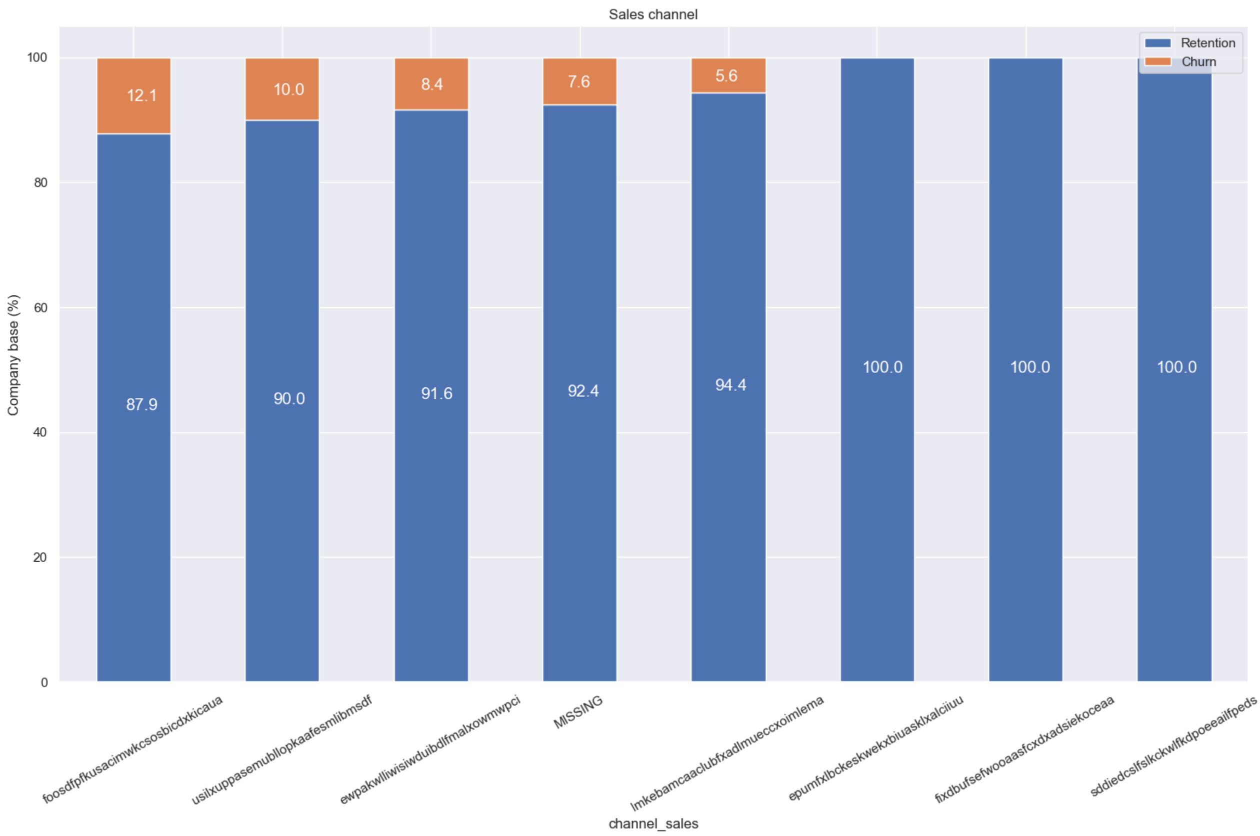
In [18]:
```
import plotly.express as px

# Assuming channel_churn is your DataFrame
```

```
fig = px.pie(channel_churn, names=channel_churn.index, values=1, title='Channel Churn Possibility', hole=0.5)
fig.show()
```



customers with foosdfpfkusacimwkcsosbicdxkicaua have the highest churn

In [19]:
```
plot_stacked_bars(channel_churn, 'Sales channel', rot_=30)
```

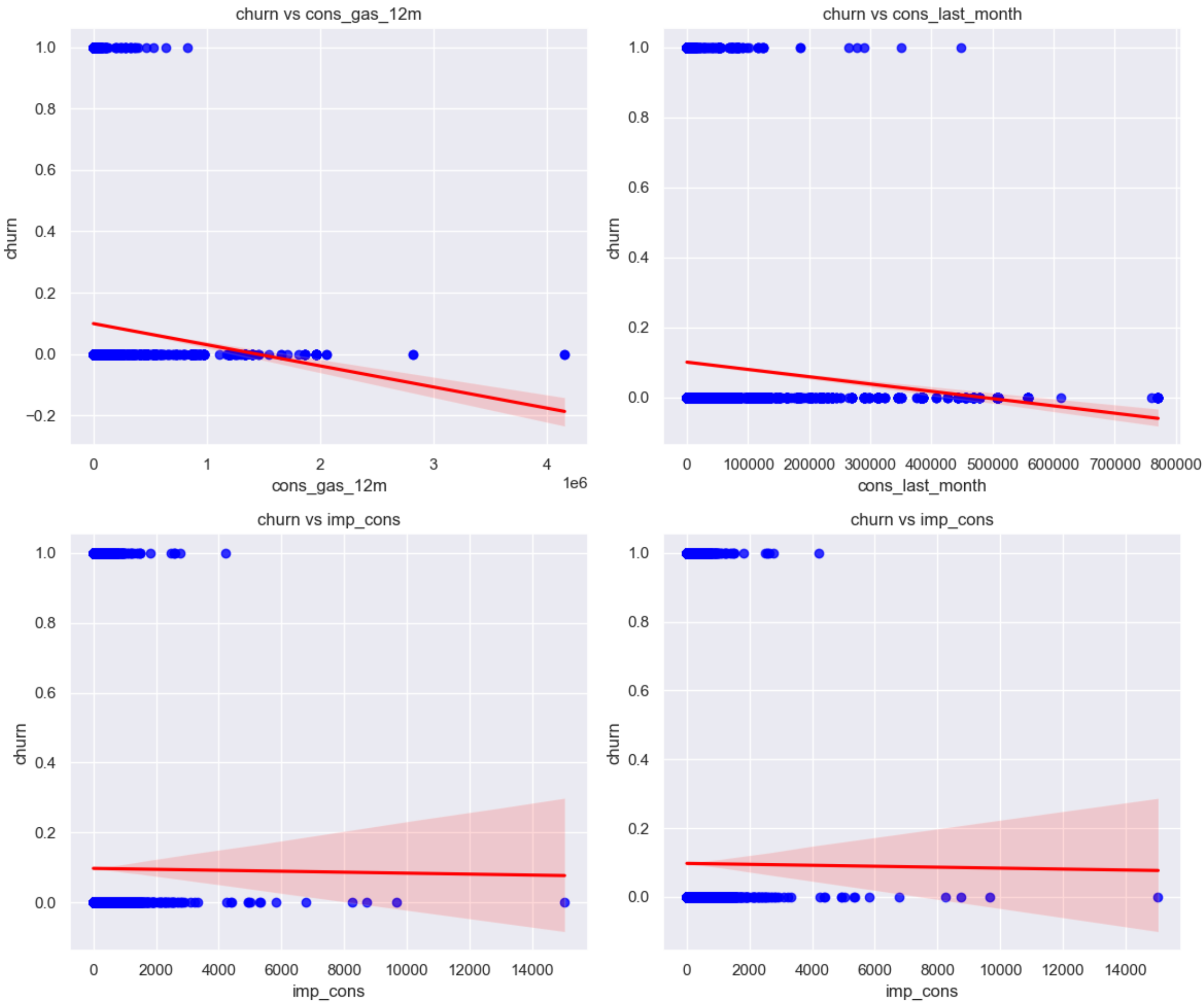


In [20]:
```
consumption = client_df[['id', 'cons_12m', 'cons_gas_12m', 'cons_last_month','imp_cons', 'has_gas', 'churn']]
```

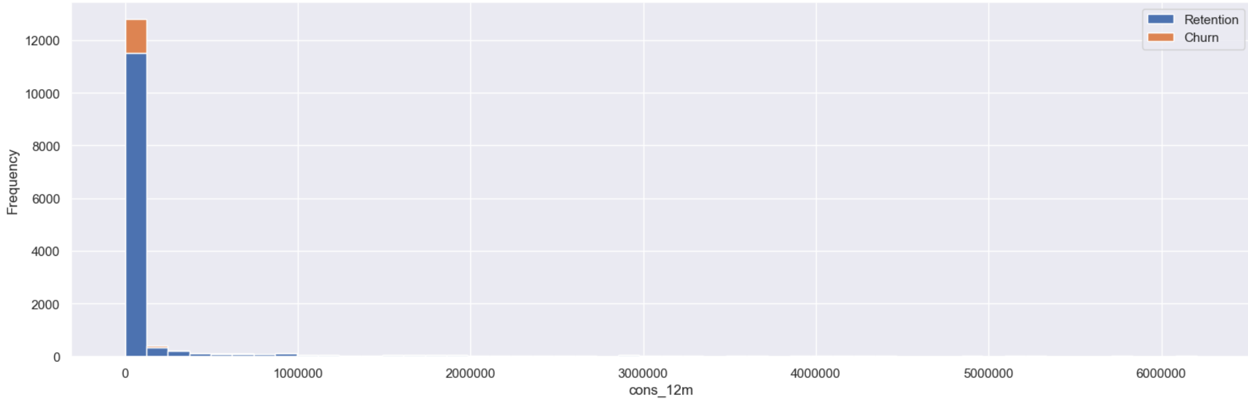
In [21]:
```
def plot_distribution(dataframe, column, ax, bins_=50):

    temp = pd.DataFrame({"Retention": dataframe[dataframe["churn"]==0][column], "Churn":dataframe[dataframe["churn"
    temp[["Retention","Churn"]].plot(kind='hist', bins=bins_, ax=ax,stacked=True)
    ax.set_xlabel(column)
    ax.ticklabel_format(style='plain', axis='x')
```

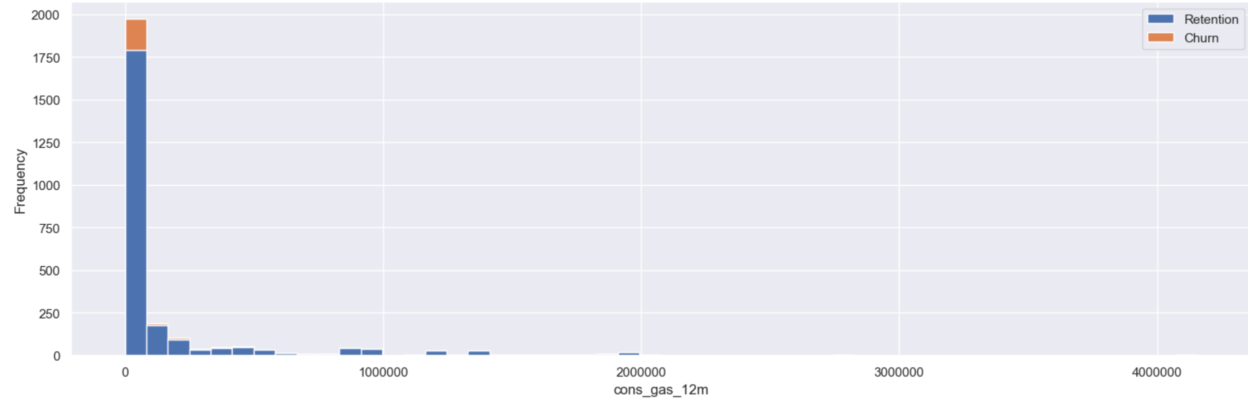
Relation between Consumption and churn

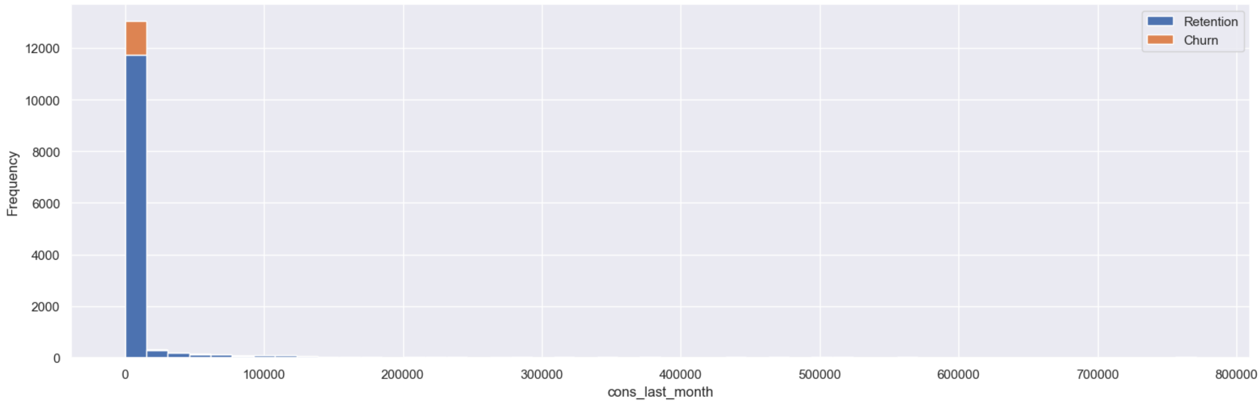
In [22]:
```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))

# Regression plot for 'cons_12m' vs 'cons_gas_12m'
sns.regplot(y='churn', x='cons_gas_12m', data=client_df,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, a
axs[0, 0].set_title('churn vs cons_gas_12m')

# Regression plot for 'cons_12m' vs 'cons_last_month'
sns.regplot(y='churn', x='cons_last_month', data=client_df,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}
axs[0, 1].set_title('churn vs cons_last_month')

# Regression plot for 'cons_12m' vs 'imp_cons'
sns.regplot(y='churn', x='imp_cons', data=client_df,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=ax
axs[1, 0].set_title('churn vs imp_cons')

# Regression plot for 'cons_gas_12m' vs 'imp_cons'
sns.regplot(y='churn', x='imp_cons', data=client_df,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=ax
axs[1, 1].set_title('churn vs imp_cons')


plt.tight_layout()

plt.show()
```

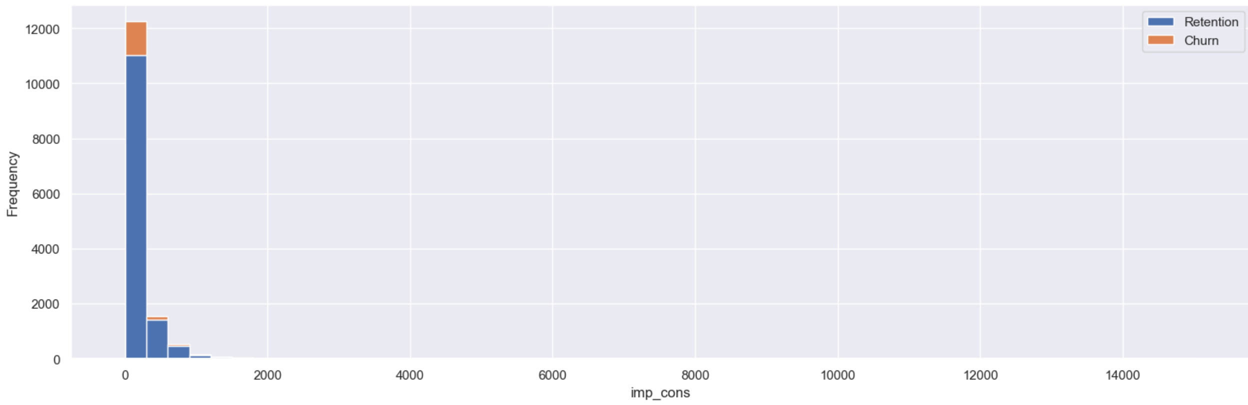


In [23]:
```
fig, axs = plt.subplots(nrows=4, figsize=(18, 25))
plot_distribution(consumption, 'cons_12m', axs[0])
plot_distribution(consumption[consumption['has_gas'] == 't'], 'cons_gas_12m',axs[1])
plot_distribution(consumption, 'cons_last_month', axs[2])
plot_distribution(consumption, 'imp_cons', axs[3])
```



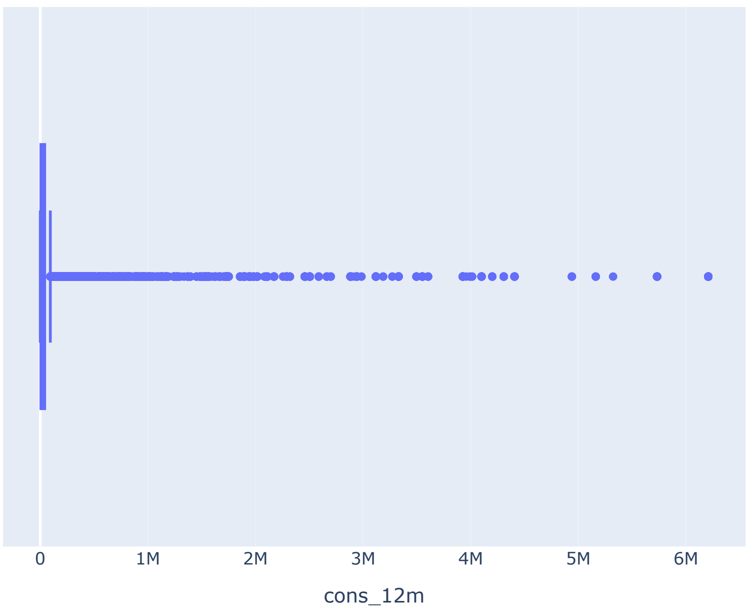


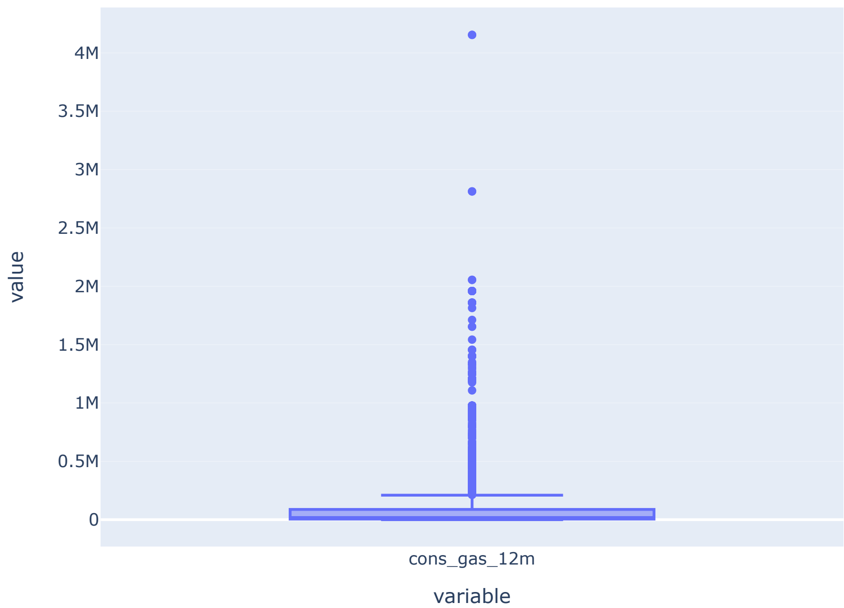


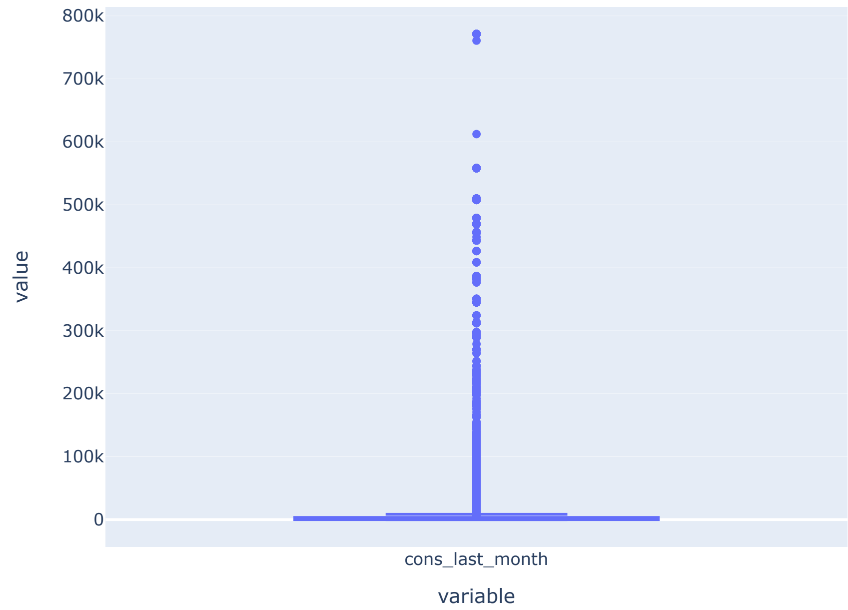


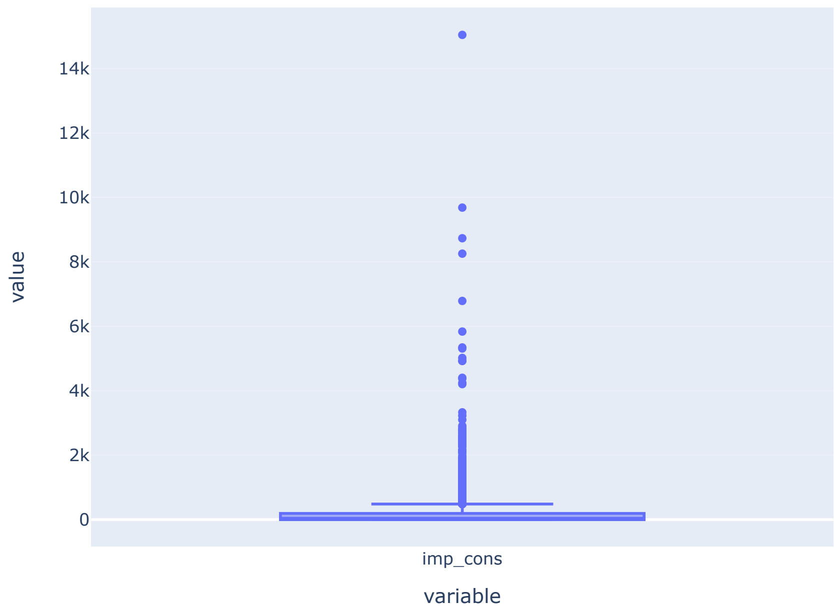
Review of outliers present in consumption data

In [24]:
```
consumption

fig1=px.box(client_df, x='cons_12m')
fig2=px.box(consumption[consumption["has_gas"] == "t"]["cons_gas_12m"])
fig3=px.box(consumption["cons_last_month"])
fig4=px.box(consumption["imp_cons"])

fig1.show()
fig2.show()
fig3.show()
fig4.show()
```


0
1M
2M
3M
4M
5M
6M
cons_12m


4M
3.5M
3M
2.5M
2M
1.5M
1M
0.5M
0
value
cons_gas_12m
variable

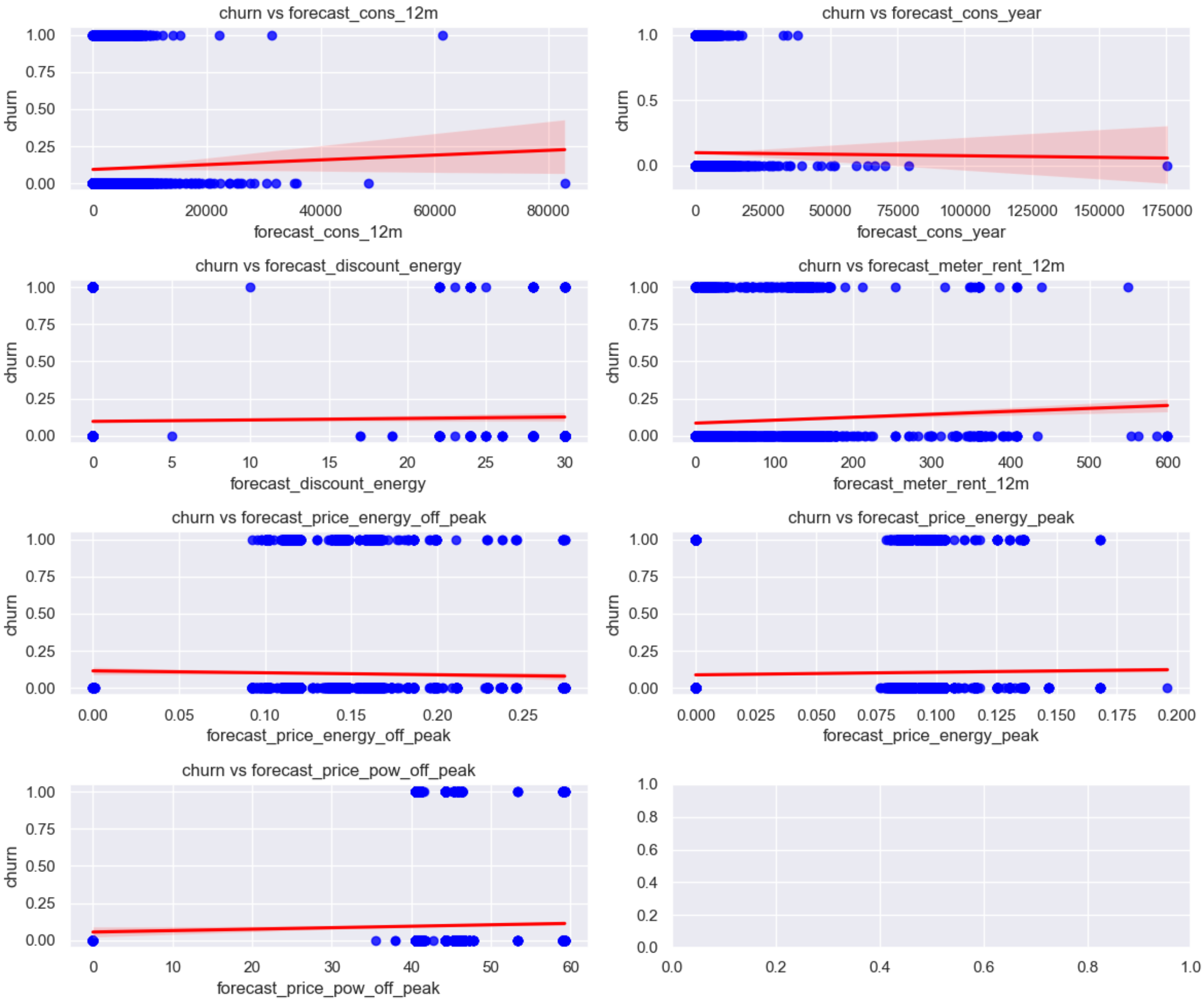




Relation between Forecast and churn

In [25]:
```
fig, axs = plt.subplots(nrows=4, ncols=2, figsize=(12, 10))

# Regression plot for 'churn' vs 'forecast_cons_12m'
sns.regplot(y='churn', x='forecast_cons_12m', data=client_df,scatter_kws={'color': 'blue'}, line_kws={'color': 'red
axs[0, 0].set_title('churn vs forecast_cons_12m')

# Regression plot for 'churn' vs 'forecast_cons_year'
sns.regplot(y='churn', x='forecast_cons_year', data=client_df,scatter_kws={'color': 'blue'}, line_kws={'color': 're
axs[0, 1].set_title('churn vs forecast_cons_year')

# Regression plot for 'churn' vs 'forecast_discount_energy'
sns.regplot(y='churn', x='forecast_discount_energy', data=client_df, scatter_kws={'color': 'blue'}, line_kws={'colo
axs[1, 0].set_title('churn vs forecast_discount_energy')

# Regression plot for 'churn' vs 'forecast_meter_rent_12m'
sns.regplot(y='churn', x='forecast_meter_rent_12m', data=client_df, scatter_kws={'color': 'blue'}, line_kws={'color
axs[1, 1].set_title('churn vs forecast_meter_rent_12m')

# Regression plot for 'churn' vs 'forecast_price_energy_off_peak'
sns.regplot(y='churn', x='forecast_price_energy_off_peak', data=client_df,scatter_kws={'color': 'blue'}, line_kws={
axs[2, 0].set_title('churn vs forecast_price_energy_off_peak')
```

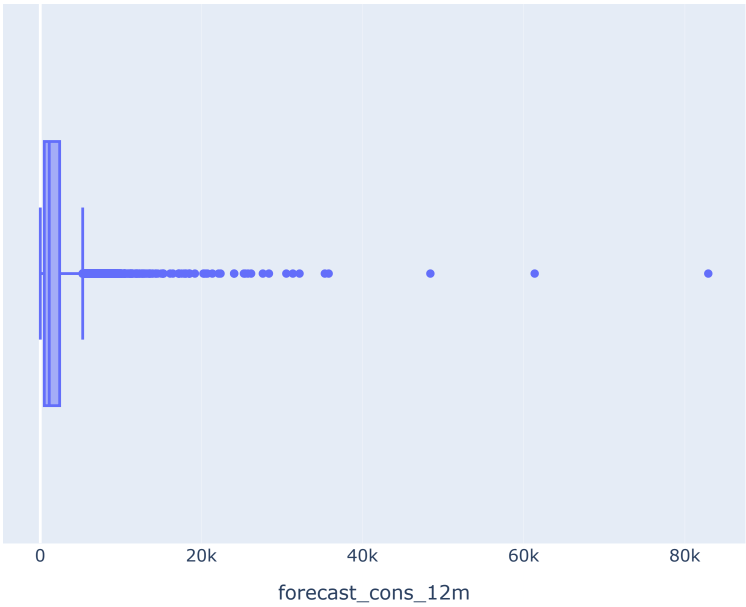
```
# Regression plot for 'churn' vs 'forecast_price_energy_peak'
sns.regplot(y='churn', x='forecast_price_energy_peak', data=client_df,scatter_kws={'color': 'blue'}, line_kws={'col
axs[2, 1].set_title('churn vs forecast_price_energy_peak')

# Regression plot for 'churn' vs 'forecast_price_pow_off_peak'
sns.regplot(y='churn', x='forecast_price_pow_off_peak', data=client_df,scatter_kws={'color': 'blue'}, line_kws={'co
axs[3, 0].set_title('churn vs forecast_price_pow_off_peak')
plt.tight_layout()

plt.show()
```

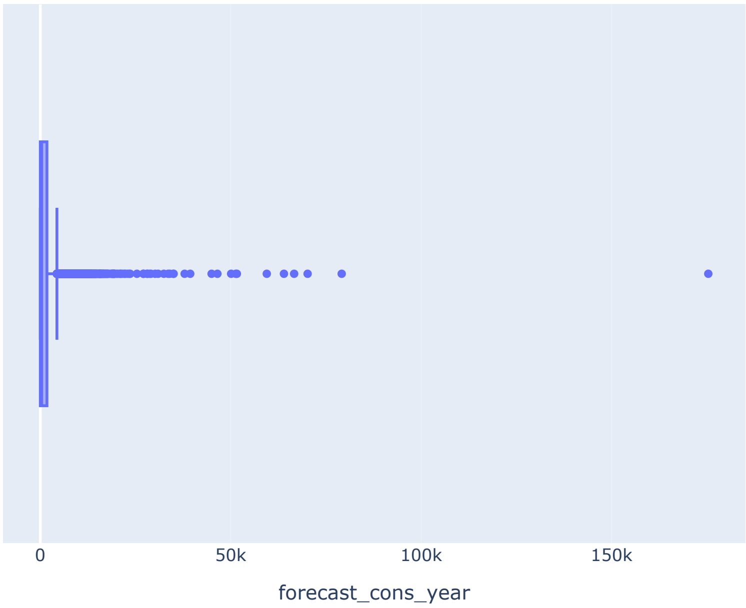


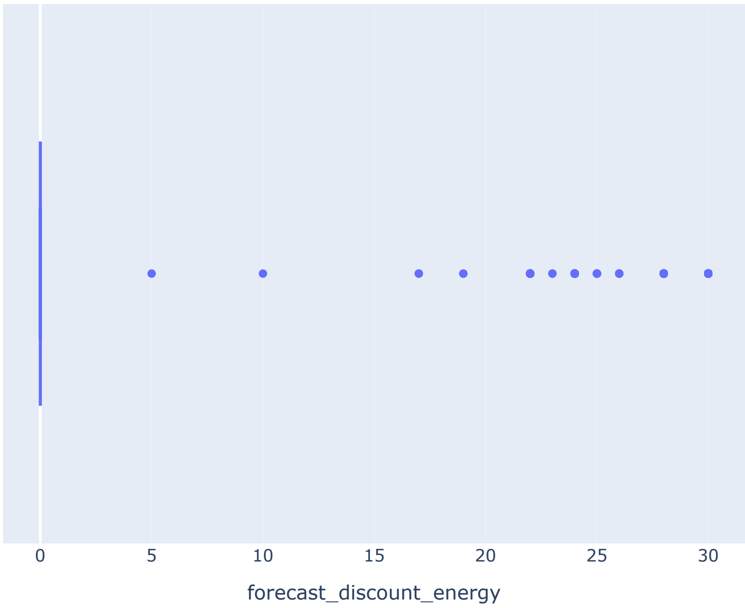
In [26]:
```
forecast = client_df[
    ["id", "forecast_cons_12m",
    "forecast_cons_year","forecast_discount_energy","forecast_meter_rent_12m",
    "forecast_price_energy_off_peak","forecast_price_energy_peak",
    "forecast_price_pow_off_peak","churn"
    ]
]
```

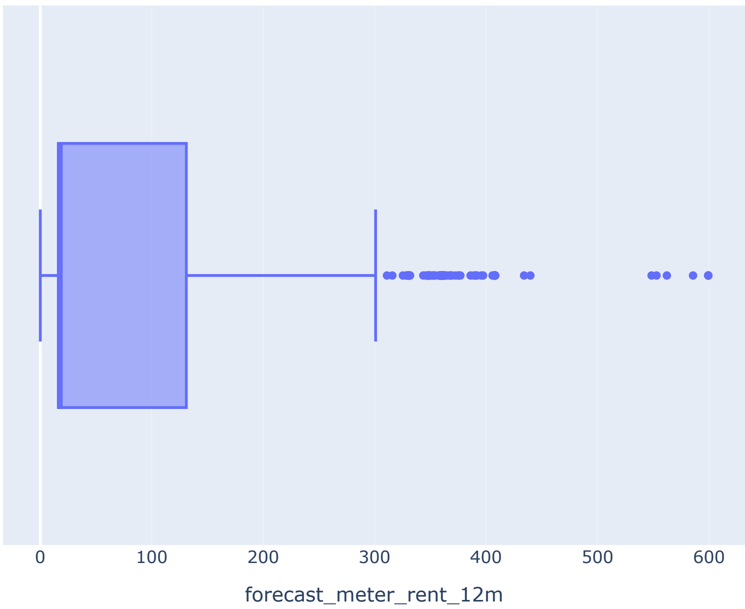
Review Of Outliers in Forcast

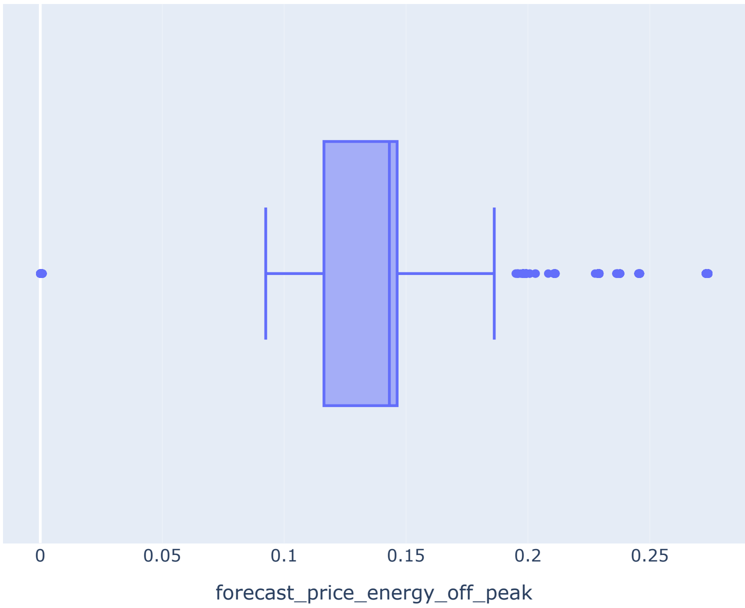
In [27]:
```
forecast

fig1=px.box(client_df, 'forecast_cons_12m')
fig2=px.box(client_df, 'forecast_cons_year')
fig3=px.box(client_df, 'forecast_discount_energy')
fig4=px.box(client_df, 'forecast_meter_rent_12m')
fig5=px.box(client_df, 'forecast_price_energy_off_peak')
fig6=px.box(client_df, 'forecast_price_energy_peak')
fig7=px.box(client_df, 'forecast_price_pow_off_peak')

fig1.show()
fig2.show()
fig3.show()
fig4.show()
fig5.show()
fig6.show()
fig7.show()
```

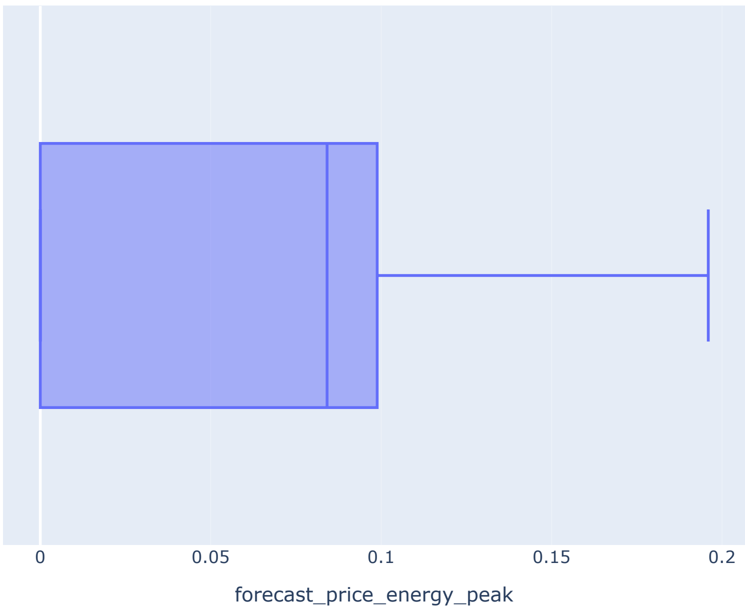

0
20k
40k
60k
80k
forecast_cons_12m

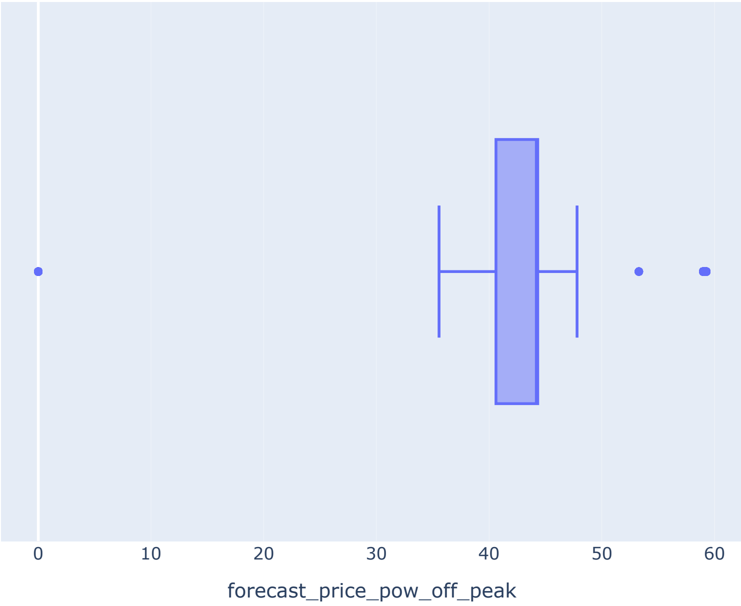

0
50k
100k
150k
forecast_cons_year


0
5
10
15
20
25
30
forecast_discount_energy


0
100
200
300
400
500
600
forecast_meter_rent_12m

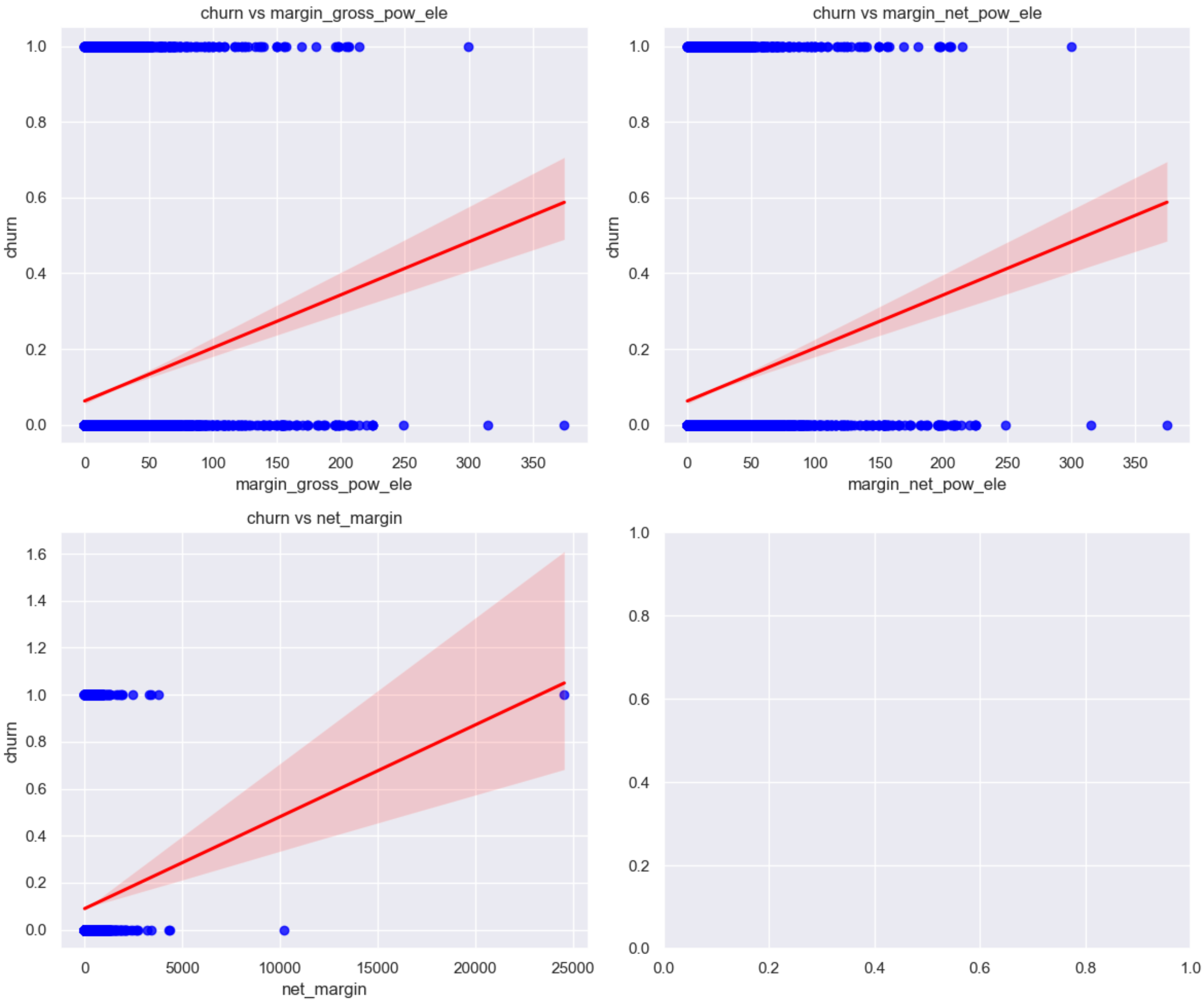

0
0.05
0.1
0.15
0.2
0.25
forecast_price_energy_off_peak


0
0.05
0.1
0.15
0.2
forecast_price_energy_peak



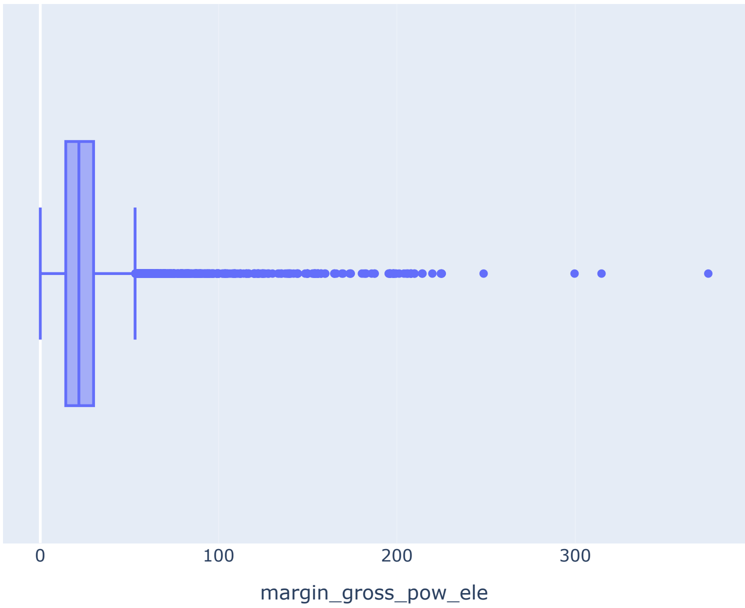
Relation between Margin and churn

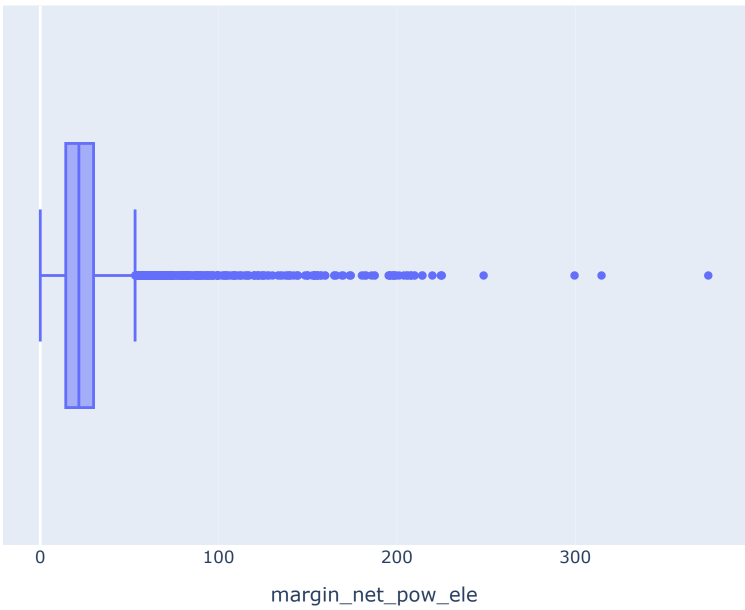
In [28]:
```
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))

# Regression plot for 'churn' vs 'margin_gross_pow_ele'
sns.regplot(y='churn', x='margin_gross_pow_ele', data=client_df,scatter_kws={'color': 'blue'}, line_kws={'color': '
axs[0, 0].set_title('churn vs margin_gross_pow_ele')

# Regression plot for 'churn' vs 'margin_net_pow_el'
sns.regplot(y='churn', x='margin_net_pow_ele', data=client_df,scatter_kws={'color': 'blue'}, line_kws={'color': 're
axs[0, 1].set_title('churn vs margin_net_pow_ele')

# Regression plot for 'churn' vs 'net_margin'
sns.regplot(y='churn', x='net_margin', data=client_df,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=
axs[1, 0].set_title('churn vs net_margin')

plt.tight_layout()

plt.show()
```

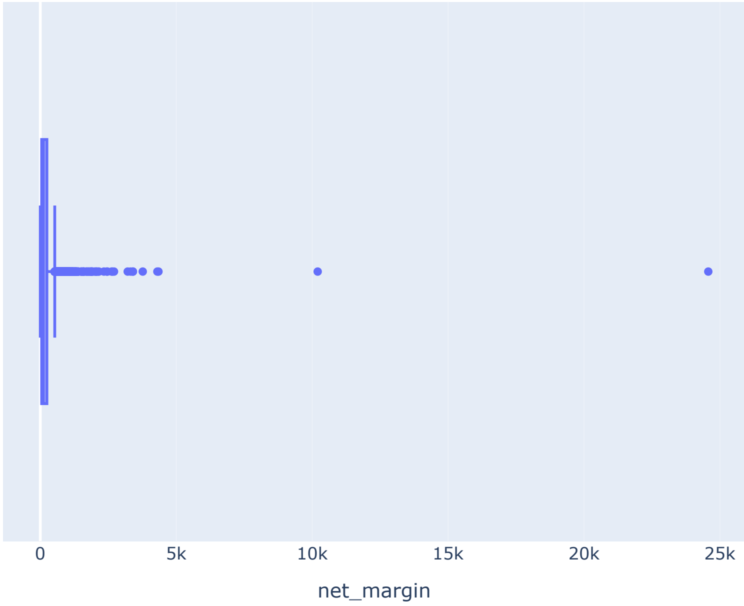


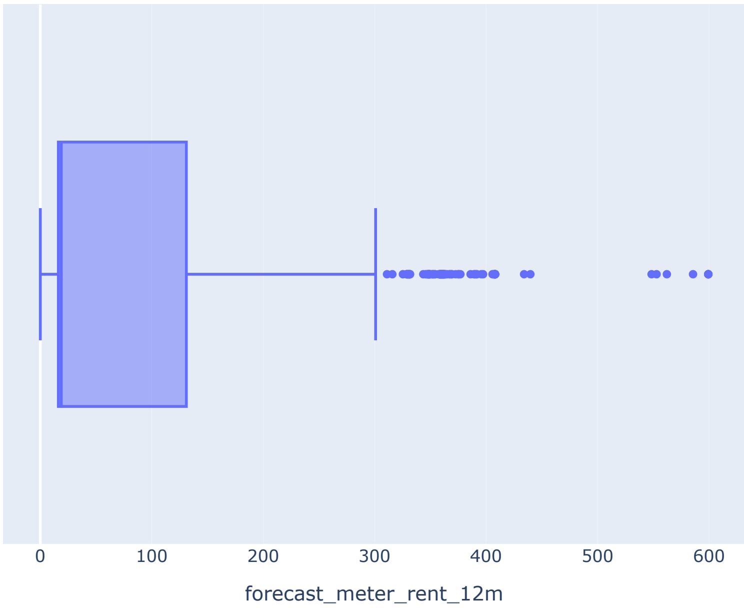
Review of outliers in margin

In [29]:
```
margin = client_df[["id", "margin_gross_pow_ele","margin_net_pow_ele","net_margin", "churn"]]

fig1=px.box(client_df, 'margin_gross_pow_ele')
fig2=px.box(client_df, 'margin_net_pow_ele')
fig3=px.box(client_df, 'net_margin')
fig1.show()
fig2.show()
fig3.show()
fig4.show()
```


0
100
200
300
margin_gross_pow_ele


0
100
200
300
margin_net_pow_ele



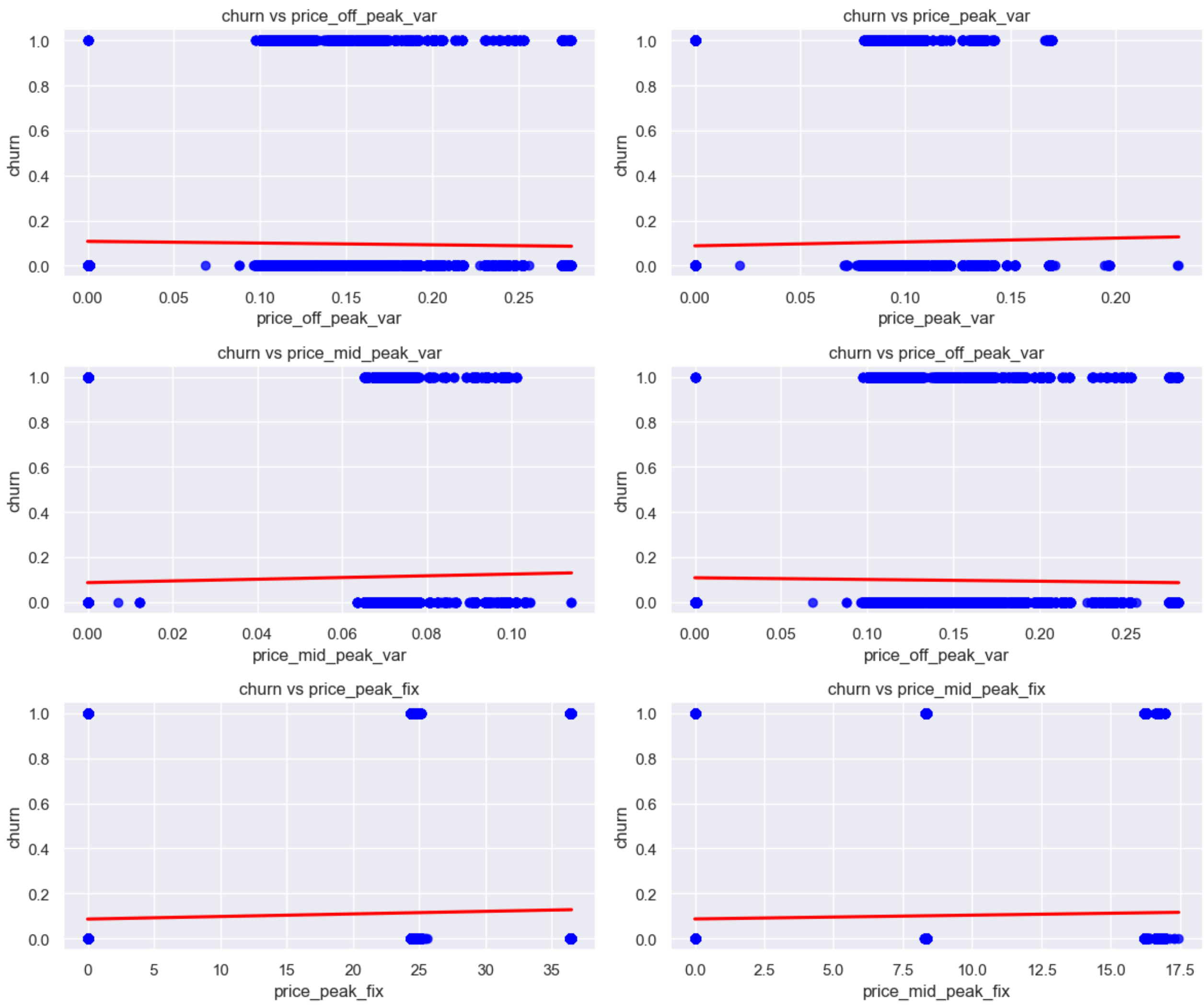


In [30]:
```
client_df["date_activ"] = pd.to_datetime(client_df["date_activ"],format='%Y-%m-%d')
client_df["date_end"] = pd.to_datetime(client_df["date_end"],format='%Y-%m-%d')
client_df["date_modif_prod"] = pd.to_datetime(client_df["date_modif_prod"],format='%Y-%m-%d')
client_df["date_renewal"] = pd.to_datetime(client_df["date_renewal"],format='%Y-%m-%d')
price_df['price_date'] = pd.to_datetime(price_df['price_date'],format='%Y-%m-%d')
```

Relation between Price and churn

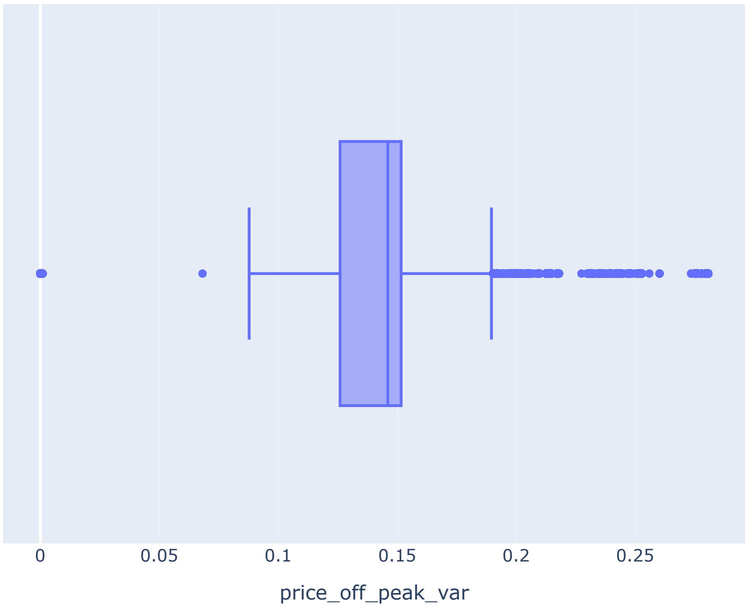
In [31]:
```
fig, axs = plt.subplots(nrows=3, ncols=2, figsize=(12, 10))

# Regression plot for 'churn' vs 'price_off_peak_var'
sns.regplot(y='churn', x='price_off_peak_var', data=c,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=
axs[0, 0].set_title('churn vs price_off_peak_var')

# Regression plot for 'churn' vs 'price_peak_var'
sns.regplot(y='churn', x='price_peak_var', data=c,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=axs[
axs[0, 1].set_title('churn vs price_peak_var')

# Regression plot for 'churn' vs 'price_mid_peak_var'
sns.regplot(y='churn', x='price_mid_peak_var', data=c,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=
axs[1, 0].set_title('churn vs price_mid_peak_var')

# Regression plot for 'churn' vs 'price_off_peak_var'
sns.regplot(y='churn', x='price_off_peak_var', data=c,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=
```

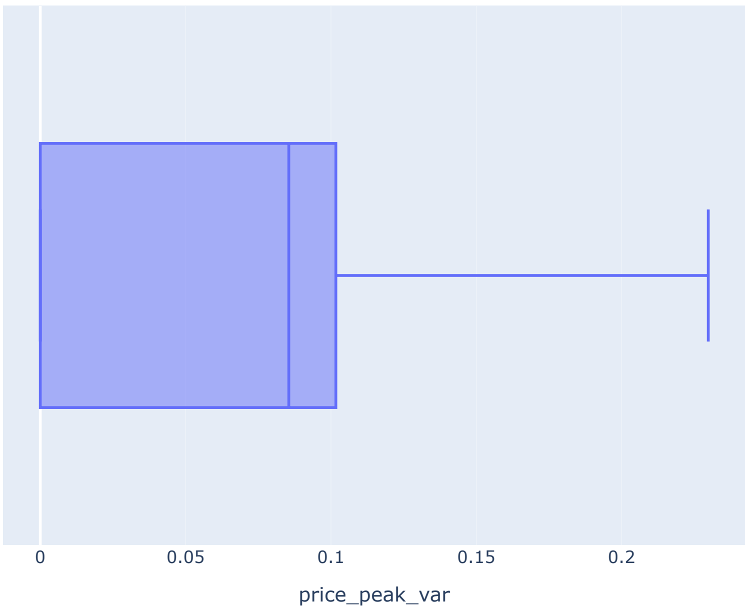
```
axs[1, 1].set_title('churn vs price_off_peak_var')

# Regression plot for 'churn' vs 'price_peak_fix'
sns.regplot(y='churn', x='price_peak_fix', data=c,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=axs[
axs[2, 0].set_title('churn vs price_peak_fix')

# Regression plot for 'churn' vs 'price_mid_peak_fix'
sns.regplot(y='churn', x='price_mid_peak_fix', data=c,scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=
axs[2, 1].set_title('churn vs price_mid_peak_fix')


plt.tight_layout()

plt.show()
```

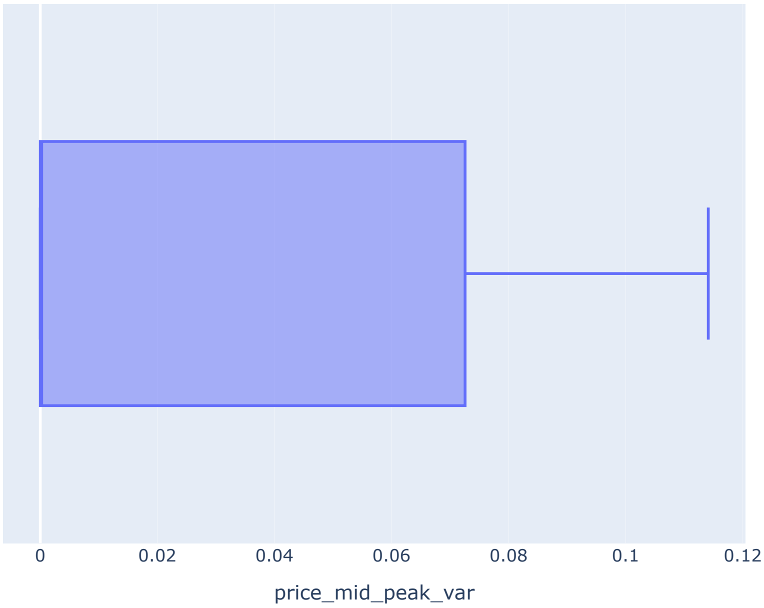


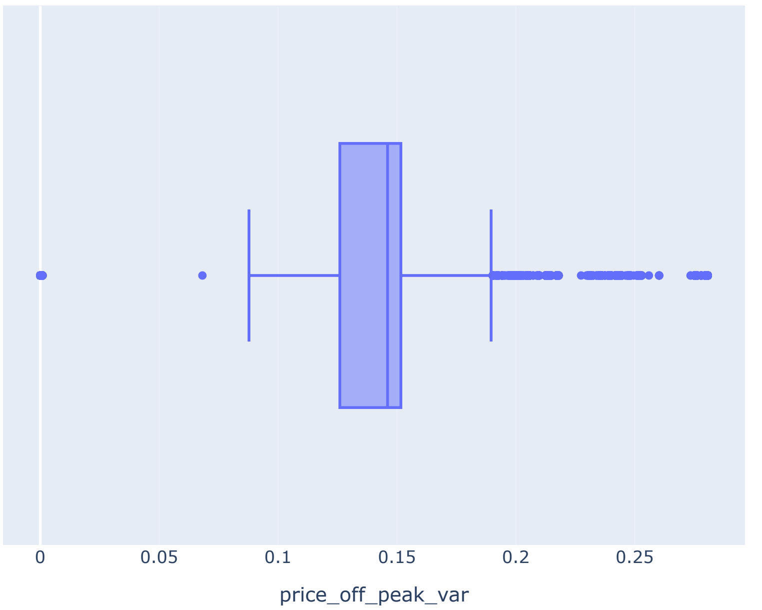
Review of outliers in Prices

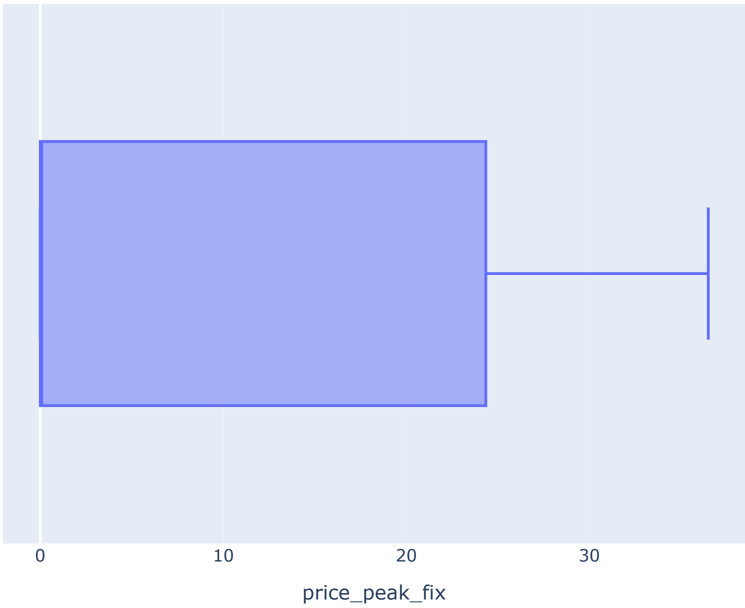
In [32]:
```
price = price_df[["id", "price_off_peak_var","price_peak_var","price_mid_peak_var","price_peak_fix","price_mid_peak

fig1=px.box(price_df, 'price_off_peak_var')
fig2=px.box(price_df, 'price_peak_var')
fig3=px.box(price_df, 'price_mid_peak_var')
fig4=px.box(price_df, 'price_off_peak_var')
fig5=px.box(price_df, 'price_peak_fix')
fig6=px.box(price_df, 'price_mid_peak_fix')

fig1.show()
fig2.show()
fig3.show()
fig4.show()
fig5.show()
fig6.show()
```

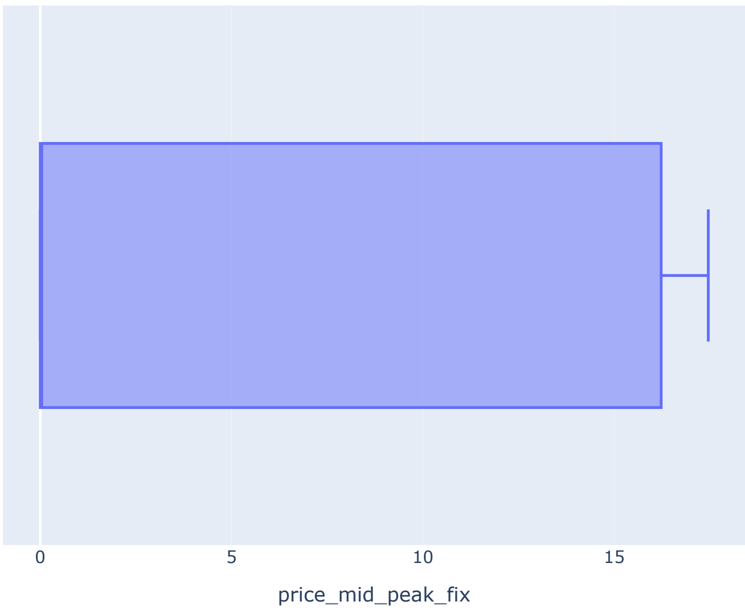

0
0.05
0.1
0.15
0.2
0.25
price_off_peak_var


0
0.05
0.1
0.15
0.2
price_peak_var


0
0.02
0.04
0.06
0.08
0.1
0.12
price_mid_peak_var


0
0.05
0.1
0.15
0.2
0.25
price_off_peak_var

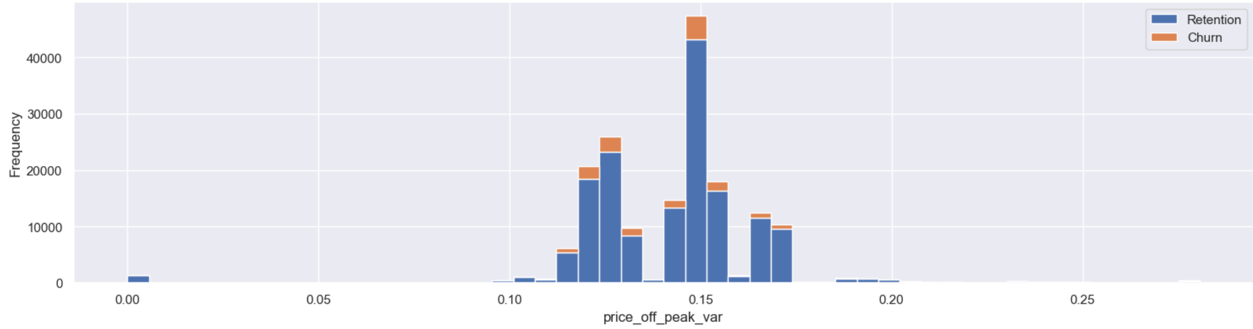


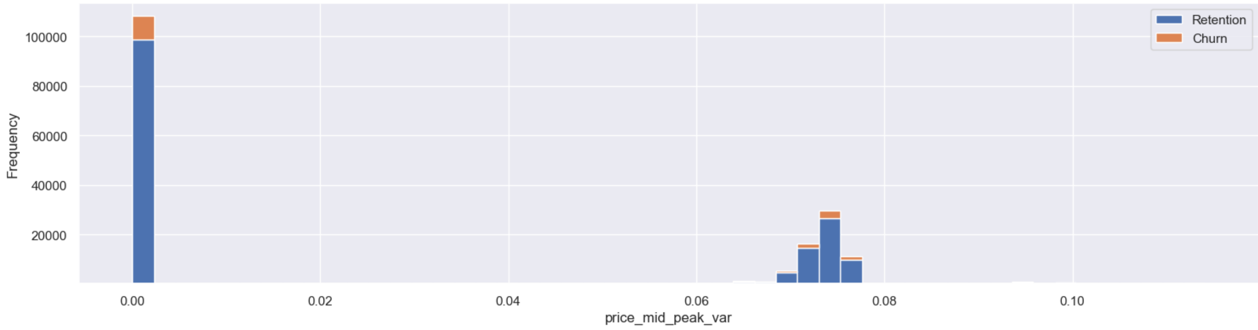


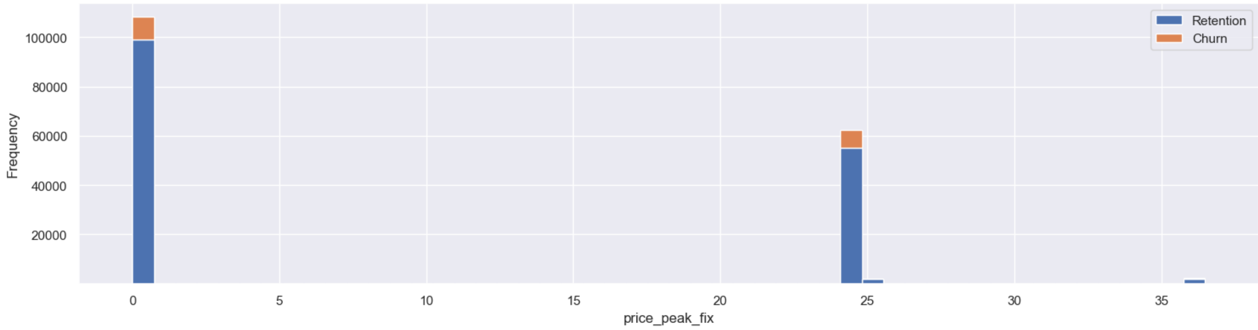
In [ ]:

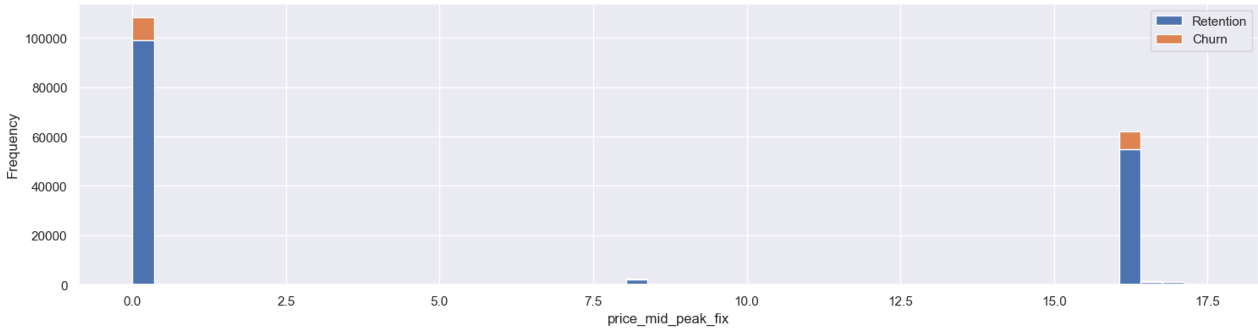
In [33]:

```
prices=c[["id", "price_off_peak_var","price_peak_var","price_mid_peak_var","price_peak_fix","price_mid_peak_fix", "

fig, axs = plt.subplots(nrows=5, figsize=(18, 25))
plot_distribution(prices, "price_off_peak_var", axs[0])
plot_distribution(prices, "price_peak_var",axs[1])
plot_distribution(prices, "price_mid_peak_var", axs[2])
plot_distribution(prices, "price_peak_fix", axs[3])
plot_distribution(prices, "price_mid_peak_fix",axs[4])
```









In [34]:
```
products = client_df[['id', 'nb_prod_act', 'num_years_antig', 'origin_up','churn']]
products = products.groupby([products["nb_prod_act"],products["churn"]])["id"].count().unstack(level=1).fillna(0)
products_percentage = (products.div(products.sum(axis=1), axis=0)*100).sort_values(by=[1], ascending=False)
```

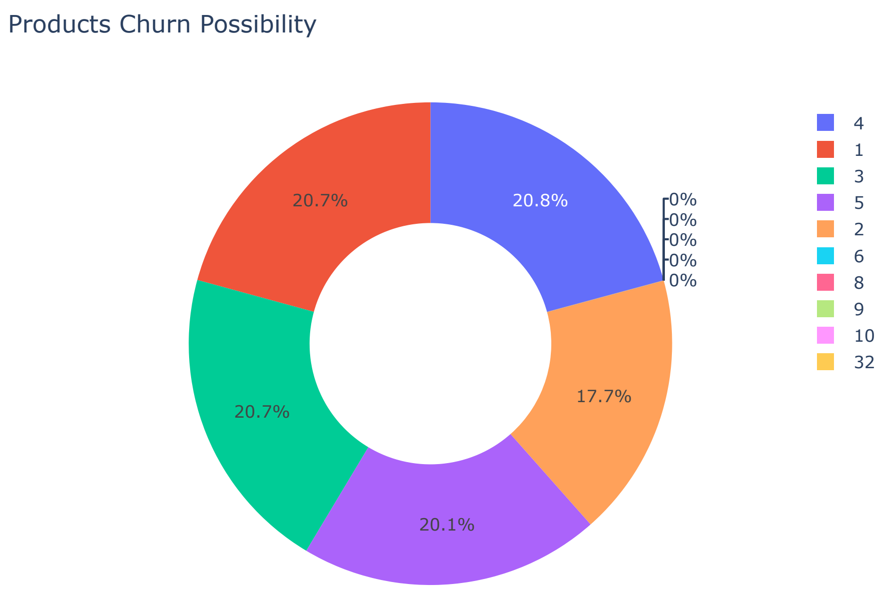
Relation between Number of Active Products and churn

In [35]:
```
products
```

Out[35]:

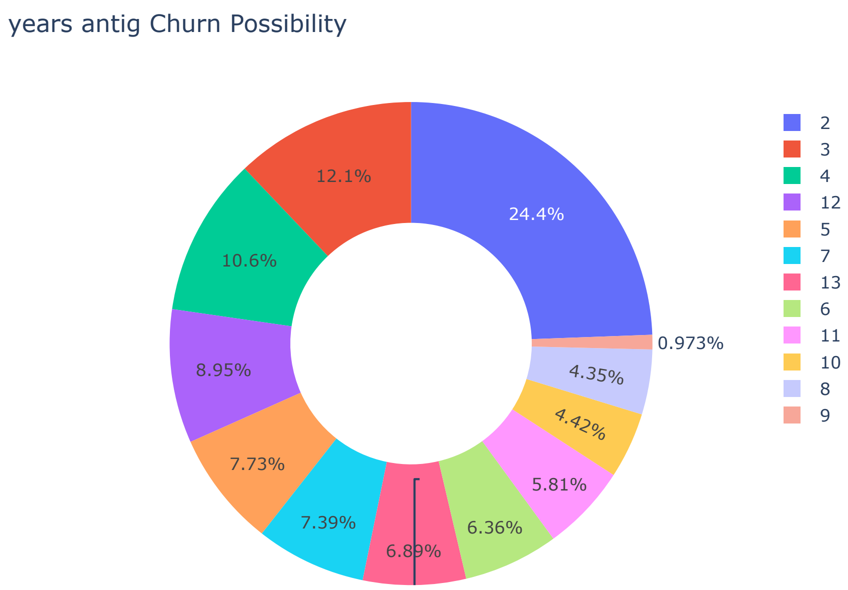
churn	0	1
nb_prod_act		
1	10290.0	1141.0
2	2237.0	208.0
3	471.0	52.0
4	135.0	15.0
5	28.0	3.0
6	8.0	0.0
8	4.0	0.0
9	11.0	0.0
10	2.0	0.0
32	1.0	0.0

In [36]:
```
fig = px.pie(products_percentage, names=products_percentage.index, values=1, title='Products Churn Possibility', ho
fig.show()
```



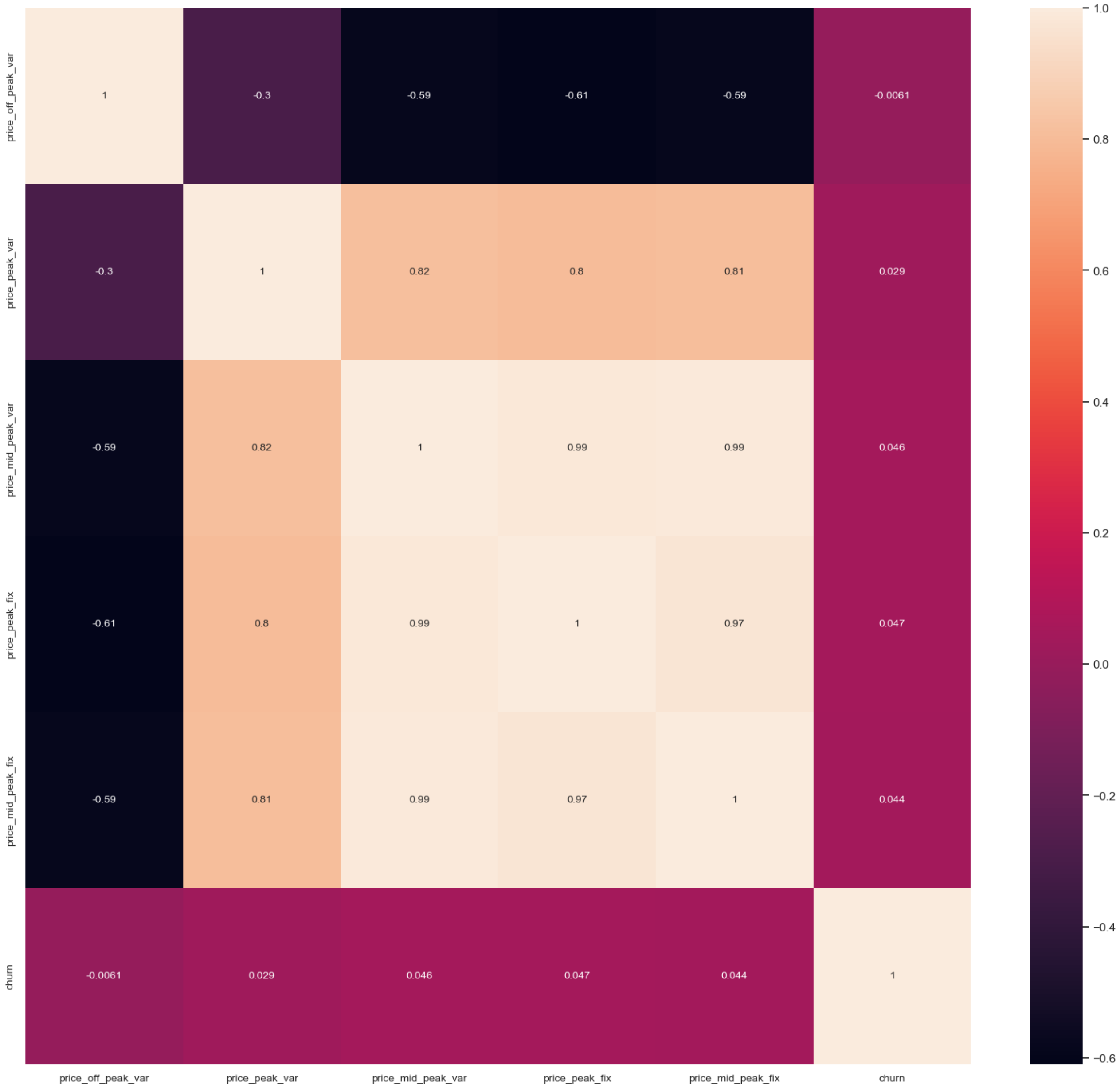
In [37]:
```
years_antig=client_df[['id','num_years_antig','churn']]
years_antig = years_antig.groupby([years_antig["num_years_antig"],years_antig["churn"]])["id"].count().unstack(leve
years_antig_percentage = (years_antig.div(years_antig.sum(axis=1), axis=0)*100)
```

In [38]:
```
fig = px.pie(years_antig_percentage, names=years_antig_percentage.index, values=1, title='years antig Churn Possibi
fig.show()
```



Heatmap showing different price variations and their likelihood to churn

In [39]:
```
corr = prices.drop(columns=['id']).corr()
# Plot correlation
plt.figure(figsize=(20,18))
sns.heatmap(corr, xticklabels=corr.columns.values, yticklabels=corr.columns.values, annot = True, annot_kws={'size'
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```



In [40]:
```
columns_to_drop=['date_modif_prod','date_renewal','forecast_discount_energy','forecast_meter_rent_12m','forecast_pr
BCGXData_Cleaned= c.drop(columns=columns_to_drop, inplace=False)
```

In [41]:
```
BCGXData_Cleaned.to_csv('BCGXData_Cleaned.csv')
```

In [ ]:
```
```