In [1]:

```
import pandas as pd
```

In [4]:

```
import numpy as np
```

In [11]:

```
import matplotlib.pyplot as mpl
```

In [12]:

```
import seaborn as sb
```

In [13]:

```
sb.set(color_codes= True)
```

In [16]:

```
import warnings
```

In [20]:

```
warnings.filterwarnings('ignore')
```

# Importing Datasets

In [23]:

```
client_data = pd.read_csv('client_data.csv')
client_data .head()
```

Out[23]:

|  | id | channel_sales | cons_12m | cons_gas_12m |
|---|---|---|---|---|
| **0** | 24011ae4ebbe3035111d65fa7c15bc57 | foosdfpfkusacimwkcsosbicdxkicaua | 0 | 54946 |
| **1** | d29c2c54acc38ff3c0614d0a653813dd | MISSING | 4660 | 0 |
| **2** | 764c75f661154dac3a6c254cd082ea7d | foosdfpfkusacimwkcsosbicdxkicaua | 544 | 0 |
| **3** | bba03439a292a1e166f80264c16191cb | lmkebamcaaclubfxadlmueccxoimlema | 1584 | 0 |
| **4** | 149d57cf92fc41cf94415803a877cb4b | MISSING | 4425 | 0 |

5 rows × 26 columns

In [24]:

```
price_data = pd.read_csv('price_data.csv')
price_data .head()
```

Out[24]:

|  | id | price_date | price_off_peak_var | price_peak_var | price_mid_pea |
|---|---|---|---|---|---|
| **0** | 038af19179925da21a25619c5a24b745 | 01-01-2015 | 0.151367 | 0.0 |  |
| **1** | 038af19179925da21a25619c5a24b745 | 01-02-2015 | 0.151367 | 0.0 |  |
| **2** | 038af19179925da21a25619c5a24b745 | 01-03-2015 | 0.151367 | 0.0 |  |
| **3** | 038af19179925da21a25619c5a24b745 | 01-04-2015 | 0.149626 | 0.0 |  |
| **4** | 038af19179925da21a25619c5a24b745 | 01-05-2015 | 0.149626 | 0.0 |  |

# 3: Descriptive Statistics of Data

# Data Types

In [28]:

```
print("rows and column are ",client_data.shape)
```

```
rows and column are  (14606, 26)
```

In [29]:

```
print("rows and column are ",price_data.shape)
```

```
rows and column are  (193002, 8)
```

# Basic info of client_data

In [30]:

```
client_data.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  int64  
 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(10), int64(8), object(8)
memory usage: 2.9+ MB
```

# Basic info of price_data

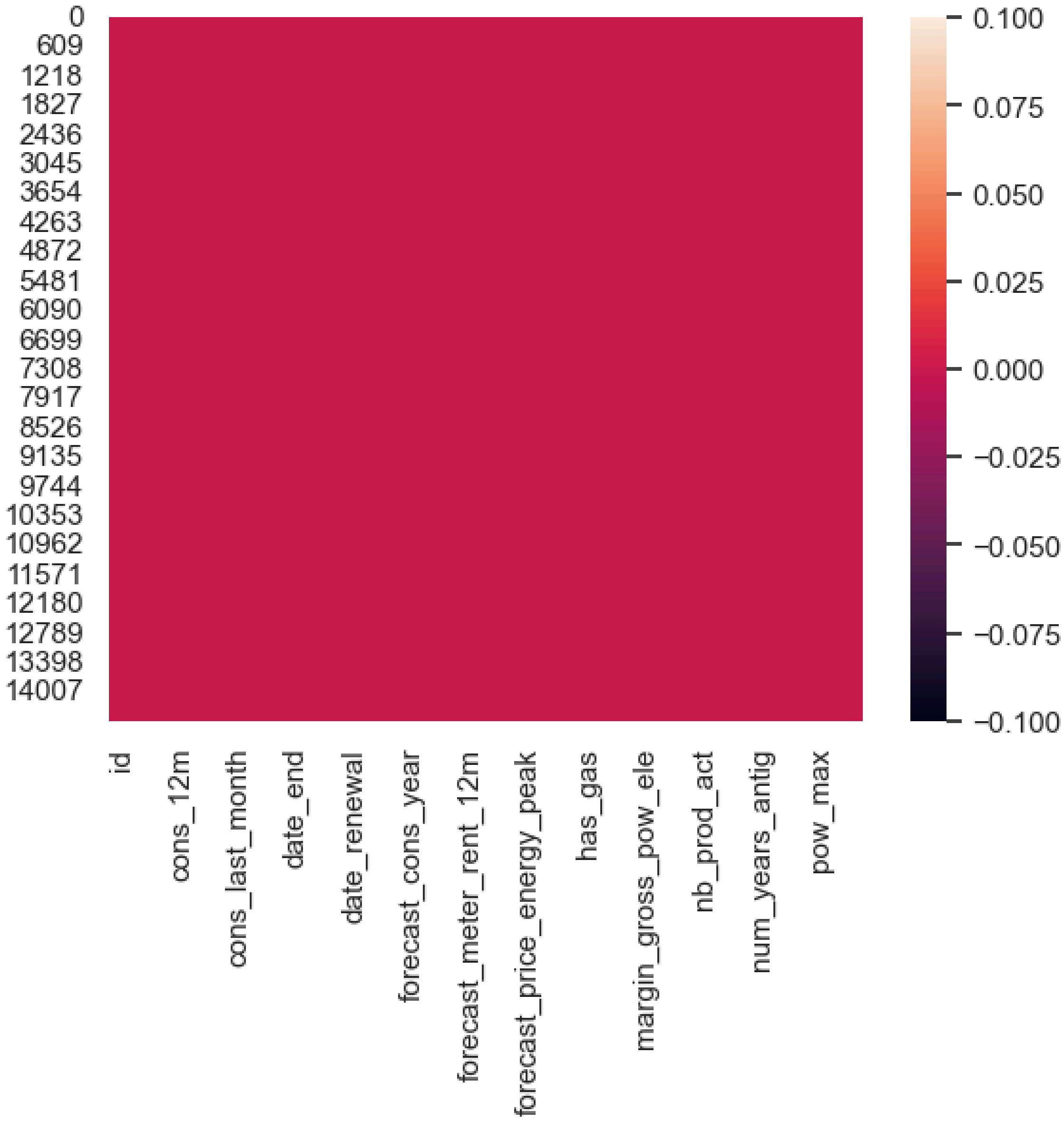
In [32]:

```
price_data.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 [33]:

```
sb.heatmap(client_data.isnull())
mpl.show()
```



In [34]:

```
client_data.isnull().sum()
```

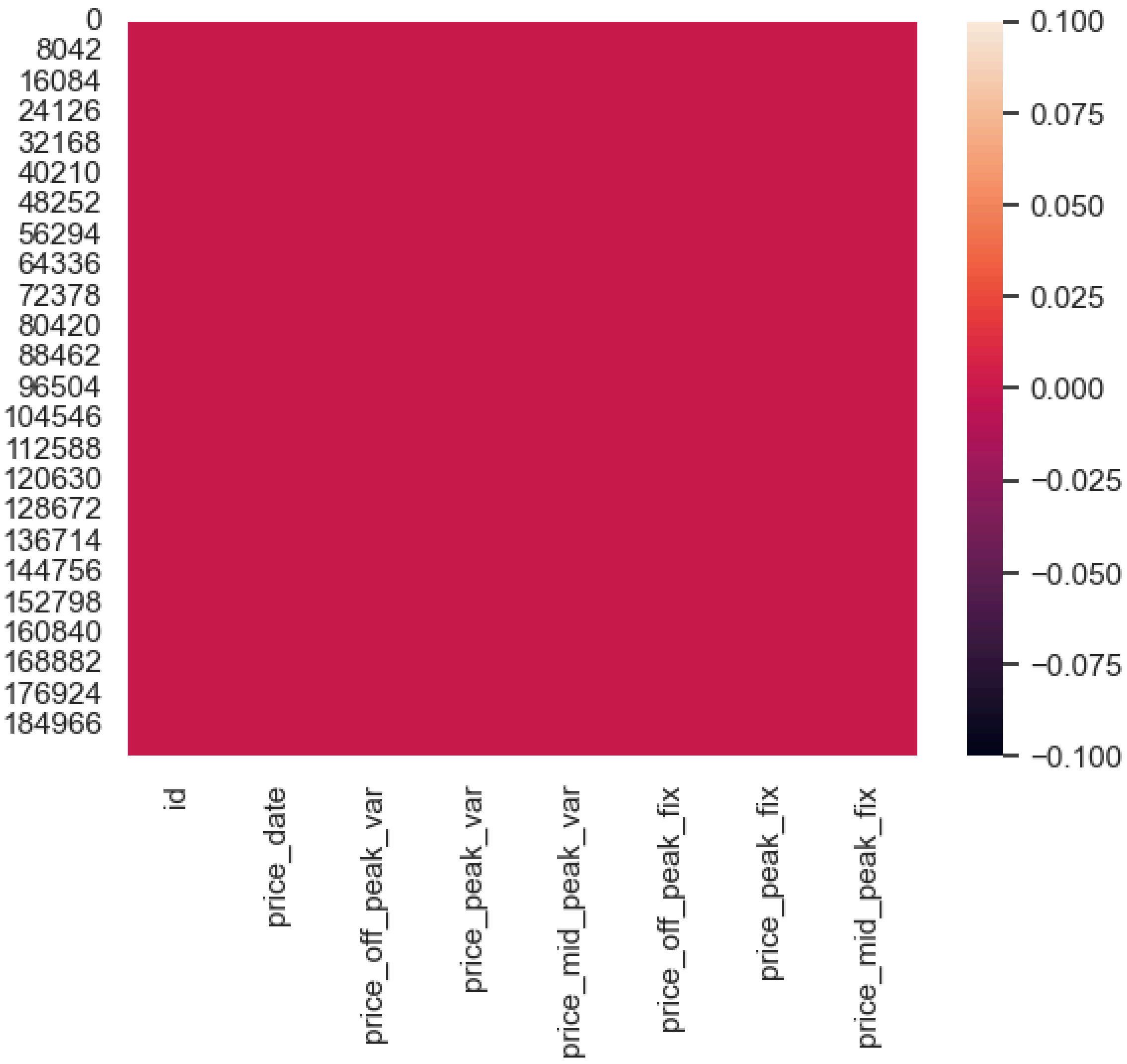
Out[34]:

```
id                                0
channel_sales                     0
cons_12m                          0
cons_gas_12m                      0
cons_last_month                   0
date_activ                        0
date_end                          0
date_modif_prod                   0
date_renewal                      0
forecast_cons_12m                 0
forecast_cons_year                0
forecast_discount_energy          0
forecast_meter_rent_12m           0
forecast_price_energy_off_peak    0
forecast_price_energy_peak        0
forecast_price_pow_off_peak       0
has_gas                           0
imp_cons                          0
margin_gross_pow_ele              0
margin_net_pow_ele                0
nb_prod_act                       0
net_margin                        0
num_years_antig                   0
origin_up                         0
pow_max                           0
churn                             0
dtype: int64
```

# checking missing values of price data

In [35]:

```
sb.heatmap(price_data.isnull())
mpl.show()
```



In [36]:

```
price_data .isnull().sum()
```

Out[36]:

```
id                    0
price_date            0
price_off_peak_var    0
price_peak_var        0
price_mid_peak_var    0
price_off_peak_fix    0
price_peak_fix        0
price_mid_peak_fix    0
dtype: int64
```

As we can see there is no missing value there in both the tables

# Statistics

Now let see some statititics about both the data tables

In [40]:

```
client_data.describe().T
```

Out[40]:

| | count | mean | std | min | 25% | 5 |
|---|---|---|---|---|---|---|
| **cons_12m** | 14606.0 | 159220.286252 | 573465.264198 | 0.0 | 5674.750000 | 14115.500 |
| **cons_gas_12m** | 14606.0 | 28092.375325 | 162973.059057 | 0.0 | 0.000000 | 0.000 |
| **cons_last_month** | 14606.0 | 16090.269752 | 64364.196422 | 0.0 | 0.000000 | 792.500 |
| **forecast_cons_12m** | 14606.0 | 1868.614880 | 2387.571531 | 0.0 | 494.995000 | 1112.875 |
| **forecast_cons_year** | 14606.0 | 1399.762906 | 3247.786255 | 0.0 | 0.000000 | 314.000 |
| **forecast_discount_energy** | 14606.0 | 0.966726 | 5.108289 | 0.0 | 0.000000 | 0.000 |
| **forecast_meter_rent_12m** | 14606.0 | 63.086871 | 66.165783 | 0.0 | 16.180000 | 18.795 |
| **forecast_price_energy_off_peak** | 14606.0 | 0.137283 | 0.024623 | 0.0 | 0.116340 | 0.143 |
| **forecast_price_energy_peak** | 14606.0 | 0.050491 | 0.049037 | 0.0 | 0.000000 | 0.084 |
| **forecast_price_pow_off_peak** | 14606.0 | 43.130056 | 4.485988 | 0.0 | 40.606701 | 44.311 |
| **imp_cons** | 14606.0 | 152.786896 | 341.369366 | 0.0 | 0.000000 | 37.395 |
| **margin_gross_pow_ele** | 14606.0 | 24.565121 | 20.231172 | 0.0 | 14.280000 | 21.640 |
| **margin_net_pow_ele** | 14606.0 | 24.562517 | 20.230280 | 0.0 | 14.280000 | 21.640 |
| **nb_prod_act** | 14606.0 | 1.292346 | 0.709774 | 1.0 | 1.000000 | 1.000 |
| **net_margin** | 14606.0 | 189.264522 | 311.798130 | 0.0 | 50.712500 | 112.530 |
| **num_years_antig** | 14606.0 | 4.997809 | 1.611749 | 1.0 | 4.000000 | 5.000 |
| **pow_max** | 14606.0 | 18.135136 | 13.534743 | 3.3 | 12.500000 | 13.856 |
| **churn** | 14606.0 | 0.097152 | 0.296175 | 0.0 | 0.000000 | 0.000 |

1) client_data.describe() has statistics as rows and features as columns.

2) client_data.describe().T has features as rows and statistics as columns.

In [42]:
```
# statistical summery of price data
price_data.describe().T
```

Out[42]:

| | count | mean | std | min | 25% | 50% | 75% | ma |
|---|---|---|---|---|---|---|---|---|
| **price_off_peak_var** | 193002.0 | 0.141027 | 0.025032 | 0.0 | 0.125976 | 0.146033 | 0.151635 | 0.28070 |
| **price_peak_var** | 193002.0 | 0.054630 | 0.049924 | 0.0 | 0.000000 | 0.085483 | 0.101673 | 0.22978 |
| **price_mid_peak_var** | 193002.0 | 0.030496 | 0.036298 | 0.0 | 0.000000 | 0.000000 | 0.072558 | 0.11410 |
| **price_off_peak_fix** | 193002.0 | 43.334477 | 5.410297 | 0.0 | 40.728885 | 44.266930 | 44.444710 | 59.44471 |
| **price_peak_fix** | 193002.0 | 10.622875 | 12.841895 | 0.0 | 0.000000 | 0.000000 | 24.339581 | 36.49069 |
| **price_mid_peak_fix** | 193002.0 | 6.409984 | 7.773592 | 0.0 | 0.000000 | 0.000000 | 16.226389 | 17.45822 |

# Data visualization

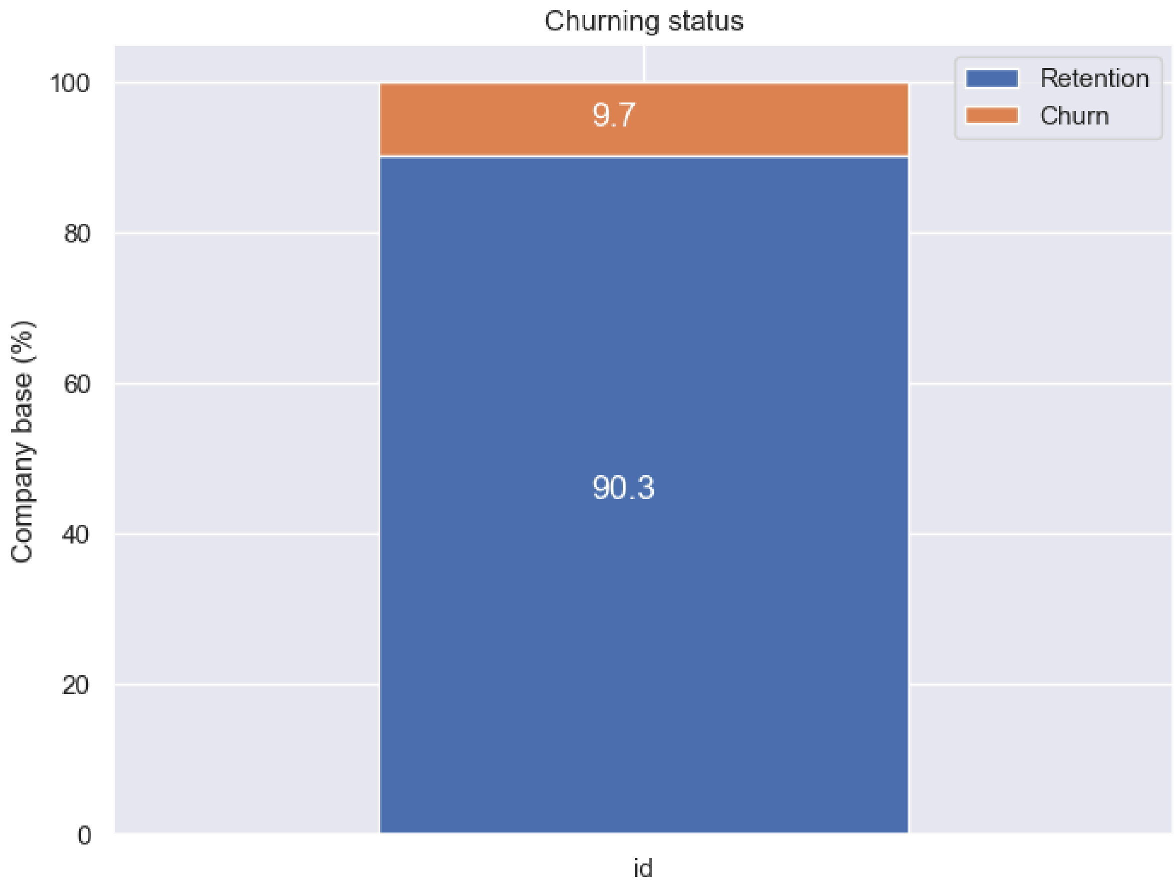
In [49]:
```
def plot_stacked_bars(dataframe, title_, size_=(18, 10), rot_=0, legend_="upper rig
    """
    Plot stacked bars with annotations
    """
    ax = dataframe.plot(
        kind="bar",
        stacked=True,
        figsize=size_,
        rot=rot_,
        title=title_
    )

    # 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)*pa
            color=colour,
            size=textsize
        )
```

In [50]:

```
churn =client_data[['id','churn']]
churn.column =['companies','churn']
churn_total = churn.groupby(churn['churn']).count()
churn_percentage = churn_total/churn_total.sum()*100
```

In [51]:

```
plot_stacked_bars(churn_percentage.transpose(), "Churning status", (8, 6), legend_=

print("\n ----- Value Counts -----\n")
print(client_data['churn'].value_counts())
```

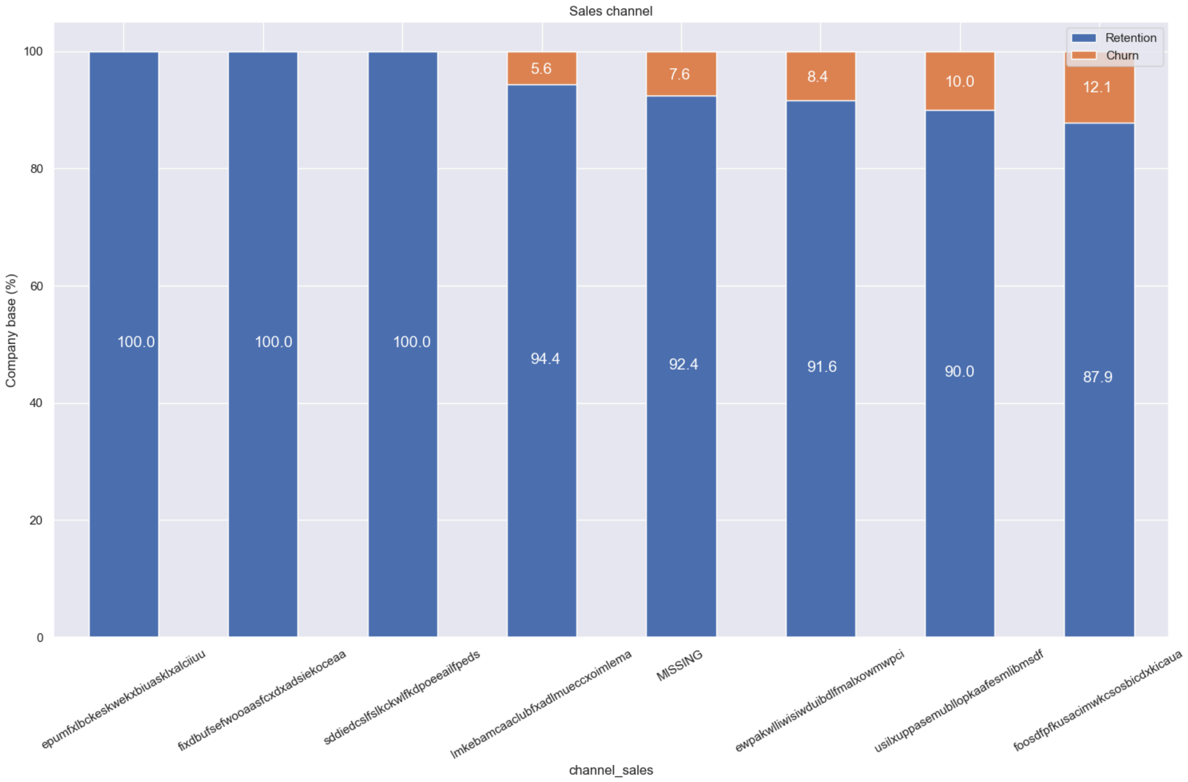


```
 ----- Value Counts -----

0    13187
1     1419
Name: churn, dtype: int64
```

Insight: nearly 10% of users have churned

```
In [59]: channel = client_data[['id', 'channel_sales', 'churn']]
         channel = channel.groupby([channel['channel_sales'], channel['churn']])['id'].count
         channel_churn = (channel.div(channel.sum(axis=1), axis=0) * 100).sort_values(by=[1]
```

```
In [60]: plot_stacked_bars(channel_churn, 'Sales channel', rot_=30)
```



# Consumption

Let's see the distribution of the consumption in the last year and month. Since the consumption data is univariate, let's use histograms to visualize their distribution.

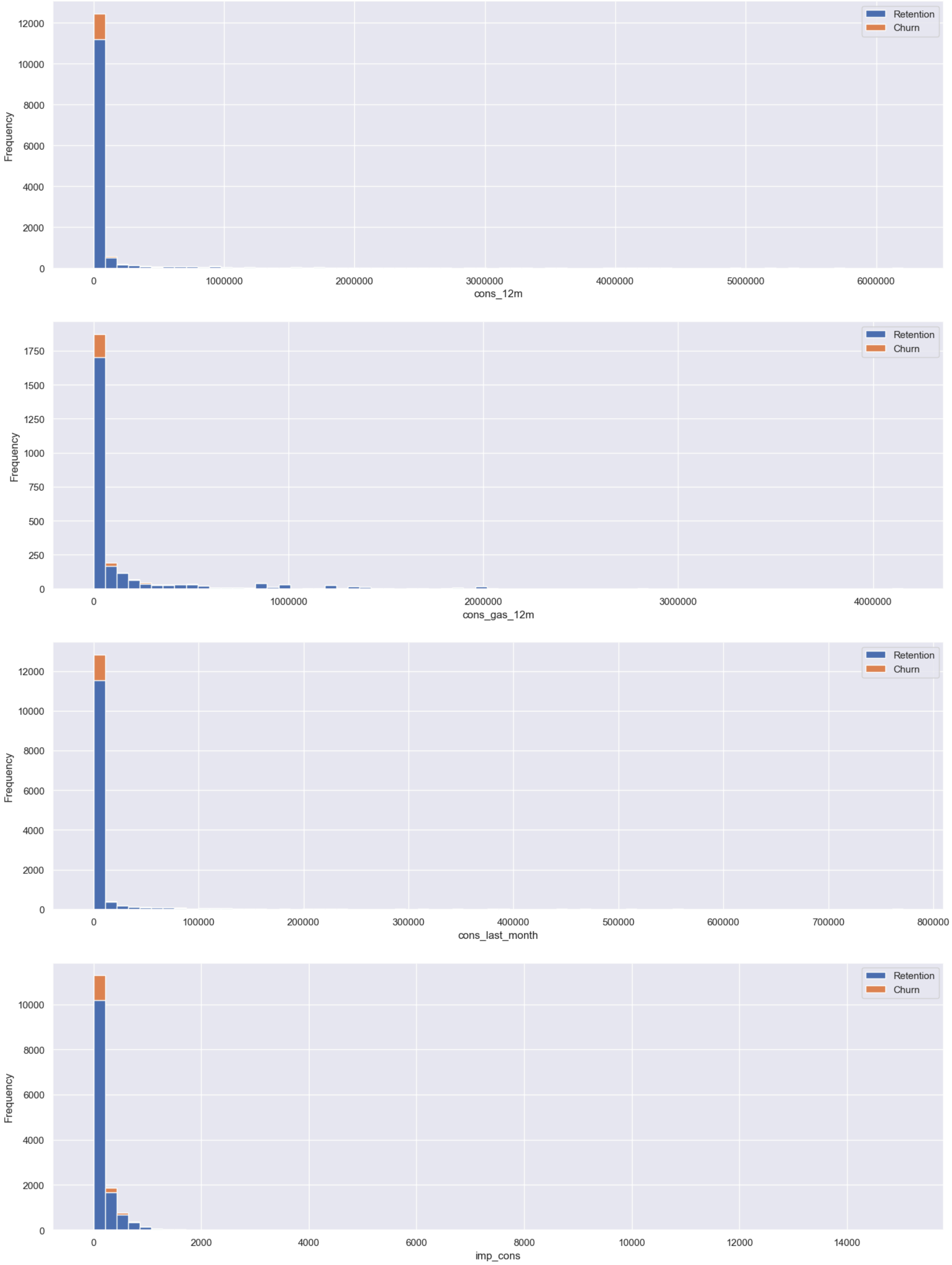
```
In [61]: consumption = client_data[['id', 'cons_12m', 'cons_gas_12m', 'cons_last_month', 'in
         consumption.head()
```

Out[61]:

| | id | cons_12m | cons_gas_12m | cons_last_month | imp_cons | has_g |
|---|---|---|---|---|---|---|
| **0** | 24011ae4ebbe3035111d65fa7c15bc57 | 0 | 54946 | 0 | 0.00 | |
| **1** | d29c2c54acc38ff3c0614d0a653813dd | 4660 | 0 | 0 | 0.00 | |
| **2** | 764c75f661154dac3a6c254cd082ea7d | 544 | 0 | 0 | 0.00 | |
| **3** | bba03439a292a1e166f80264c16191cb | 1584 | 0 | 0 | 0.00 | |
| **4** | 149d57cf92fc41cf94415803a877cb4b | 4425 | 0 | 526 | 52.32 | |

```
In [65]: def plot_distribution(dataframe, column, ax, bins_=70):
             # 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='plain', axis='x')
```

```
In [67]: fig, axs = plt.subplots(nrows=4, figsize=(18, 25))

         plot_distribution(consumption, 'imp_cons', axs[3])
plot_distribution(consumption, 'cons_12m', axs[0])
plot_distribution(consumption, 'cons_last_month', axs[2])
plot_distribution(consumption[consumption['has_gas'] == 't'], 'cons_gas_12m', axs[1

mpl.savefig("consumption Distribution.png", bbox_inches="tight")
```



# Forecast

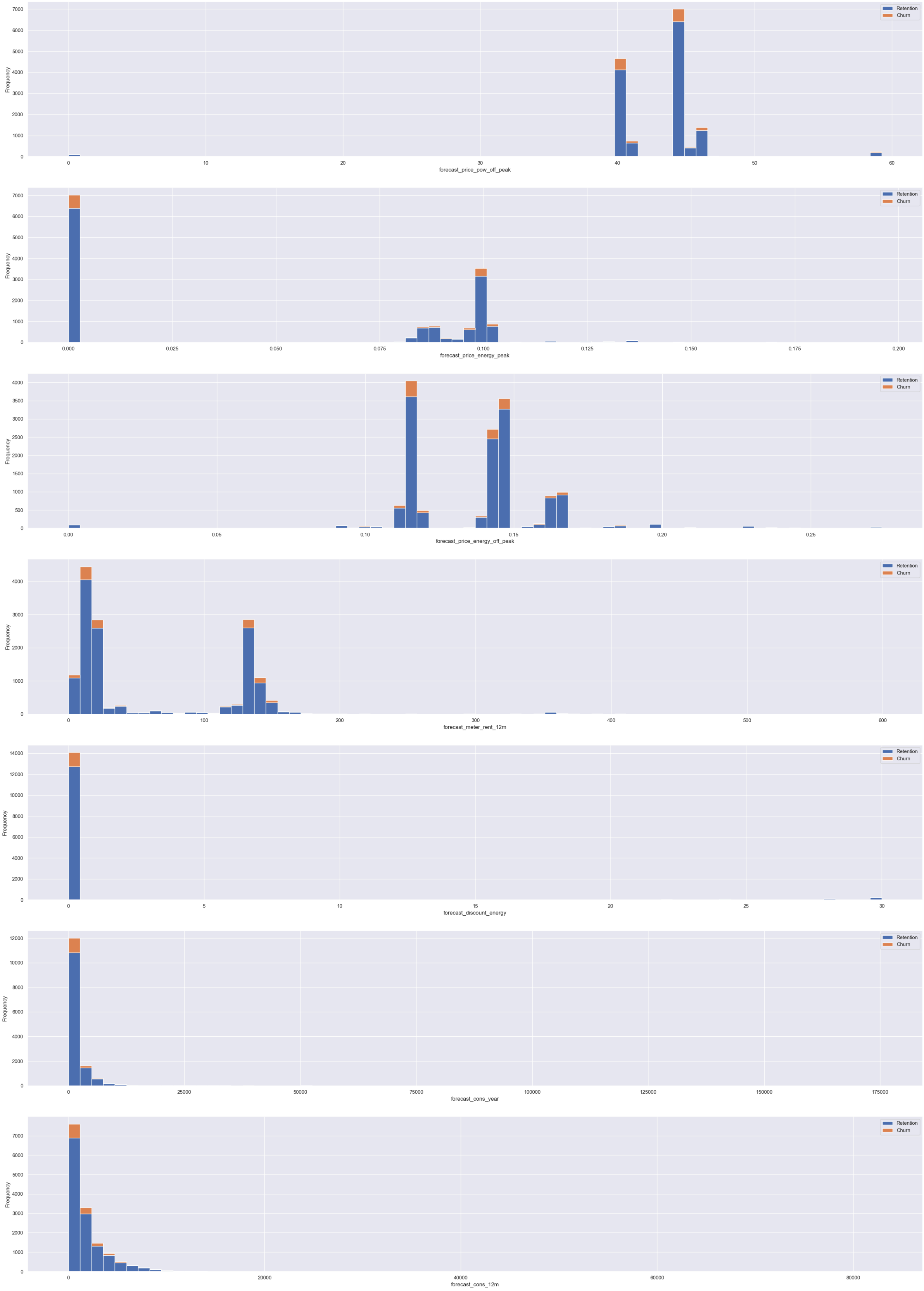
```
In [68]: forecast = client_data[["id", "forecast_cons_12m", "forecast_cons_year","forecast_c
             "forecast_price_energy_off_peak","forecast_price_energy_peak", "forecast_price_

         forecast.head()
```

Out[68]:

|   | id | forecast_cons_12m | forecast_cons_year | forecast_discount_ene |
|---|---|---|---|---|
| **0** | 24011ae4ebbe3035111d65fa7c15bc57 | 0.00 | 0 |  |
| **1** | d29c2c54acc38ff3c0614d0a653813dd | 189.95 | 0 |  |
| **2** | 764c75f661154dac3a6c254cd082ea7d | 47.96 | 0 |  |
| **3** | bba03439a292a1e166f80264c16191cb | 240.04 | 0 |  |
| **4** | 149d57cf92fc41cf94415803a877cb4b | 445.75 | 526 |  |

In [70]:

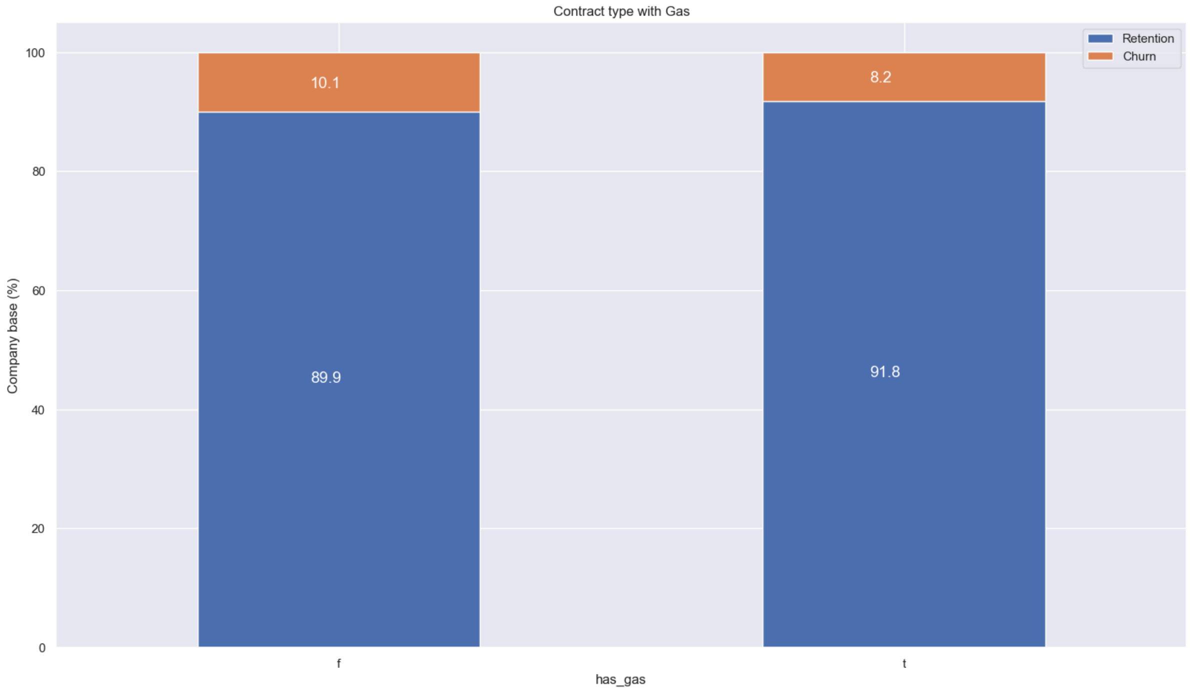
```
fig, axs = plt.subplots(nrows=7, figsize=(35,50))

# Plot histogram
plot_distribution(client_data, "forecast_cons_12m", axs[6])
plot_distribution(client_data, "forecast_cons_year", axs[5])
plot_distribution(client_data, "forecast_discount_energy", axs[4])
plot_distribution(client_data, "forecast_meter_rent_12m", axs[3])
plot_distribution(client_data, "forecast_price_energy_off_peak", axs[2])
plot_distribution(client_data, "forecast_price_energy_peak", axs[1])
plot_distribution(client_data, "forecast_price_pow_off_peak", axs[0])

plt.savefig("Forecast Views .png", bbox_inches="tight")
```



# Contract type

In [71]:
```
contract_type = client_data[['id', 'has_gas', 'churn']]
contract = contract_type.groupby([contract_type['churn'], contract_type['has_gas']]
contract_percentage = (contract.div(contract.sum(axis=1), axis=0) * 100).sort_value
```

In [73]:
```
plot_stacked_bars(contract_percentage, 'Contract type with Gas')
```



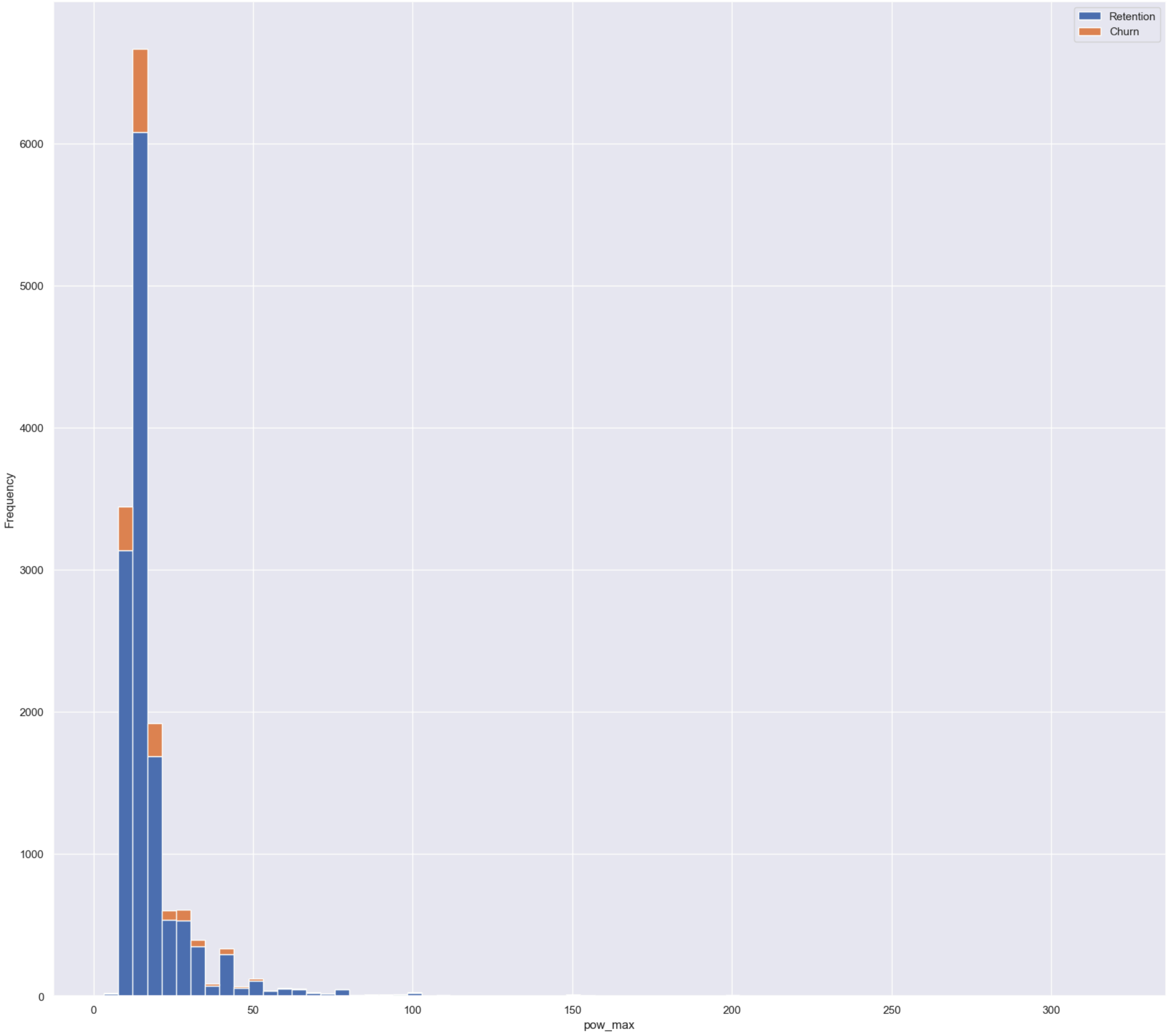
# Subscribed power

In [74]:

```
power = client_data[['id', 'pow_max', 'churn']]
```

In [79]:

```
fig, axs = plt.subplots(nrows=1, figsize=(20, 18))
plot_distribution(power, 'pow_max', axs)

plt.savefig("Subscribed power view .png", bbox_inches="tight")
```


Retention
Churn
Frequency
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250
300
pow_max

```
In [ ]:
```