In [4]:
```python
# Importing libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline

sns.set(color_codes=True)
```

In [10]:
```python
# Loading data
client_df = pd.read_csv('C:/Users/Jaehyun/Downloads/client_data.csv')
price_df = pd.read_csv('C:/Users/Jaehyun/Downloads/price_data.csv')
```

In [14]:
```python
#Checking the Data
client_df.head()
```

Out[14]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	date_activ	date_end	date_modif_prod	date_renewal	forecast_cons_12m	...	has_gas	imp_cons	margin_gross_pow_ele	margin_net_pow_ele	nb_prod_act	net_margin
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15	2016-06-15	2015-11-01	2015-06-23	0.00	...	t	0.00	25.44	25.44	2	678.99
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0	2009-08-21	2016-08-30	2009-08-21	2015-08-31	189.95	...	f	0.00	16.38	16.38	1	18.89
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0	2010-04-16	2016-04-16	2010-04-16	2015-04-17	47.96	...	f	0.00	28.60	28.60	1	6.60
3	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0	0	2010-03-30	2016-03-30	2010-03-30	2015-03-31	240.04	...	f	0.00	30.22	30.22	1	25.46
4	149d57cf92fc41cf94415803a877cb4b	MISSING	4425	0	526	2010-01-13	2016-03-07	2010-01-13	2015-03-09	445.75	...	f	52.32	44.91	44.91	1	47.98

5 rows × 26 columns

In [13]:
```python
price_df.head()
```

Out[13]:

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

In [15]:
```python
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 [16]:
```python
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 [17]:
```python
#Statistics
client_df.describe()
```

Out[17]:

	cons_12m	cons_gas_12m	cons_last_month	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	imp_cons	margin_gross_pow_ele	margin_net_po
count	1.460600e+04	1.460600e+04	14606.000000	14606.000000	14606.000000	14606.000000	14606.000000	14606.000000	14606.000000	14606.000000	14606.000000	14606.000000	14606.0
mean	1.592203e+05	2.809238e+04	16090.269752	1868.614880	1399.762906	0.966726	63.086871	0.137283	0.050491	43.130056	152.786896	24.565121	24.5
std	5.734653e+05	1.629731e+05	64364.196422	2387.571531	3247.786255	5.108289	66.165783	0.024623	0.049037	4.485988	341.369366	20.231172	20.2
min	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	5.674750e+03	0.000000e+00	0.000000	494.995000	0.000000	0.000000	16.180000	0.116340	0.000000	40.606701	0.000000	14.280000	14.2
50%	1.411550e+04	0.000000e+00	792.500000	1112.875000	314.000000	0.000000	18.795000	0.143166	0.084138	44.311378	37.395000	21.640000	21.6
75%	4.076375e+04	0.000000e+00	3383.000000	2401.790000	1745.750000	0.000000	131.030000	0.146348	0.098837	44.311378	193.980000	29.880000	29.8
max	6.207104e+06	4.154590e+06	771203.000000	82902.830000	175375.000000	30.000000	599.310000	0.273963	0.195975	59.266378	15042.790000	374.640000	374.6

In [18]:
```python
price_df.describe()
```

Out[18]:

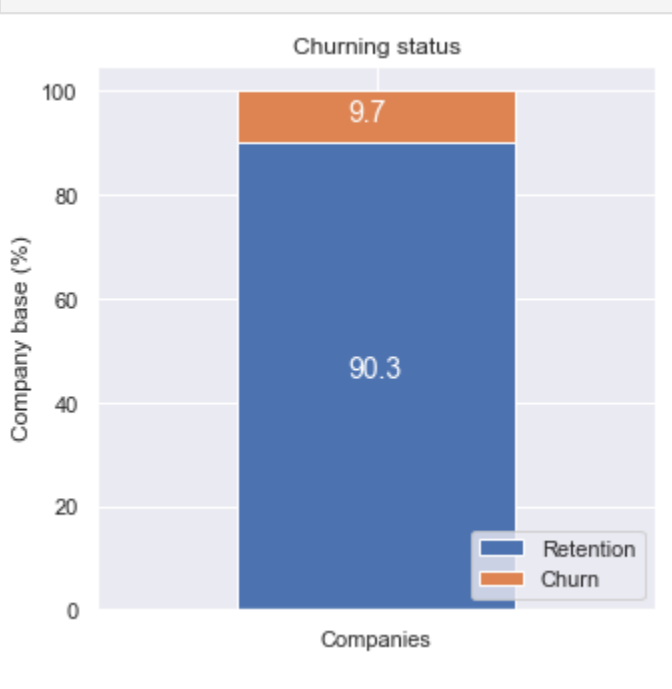
	price_off_peak_var	price_peak_var	price_mid_peak_var	price_off_peak_fix	price_peak_fix	price_mid_peak_fix
count	193002.000000	193002.000000	193002.000000	193002.000000	193002.000000	193002.000000
mean	0.141027	0.054630	0.030496	43.334477	10.622875	6.409984
std	0.025032	0.049924	0.036298	5.410297	12.841895	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

Data Visualization

In [41]:
```python
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_,
        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
        )
```

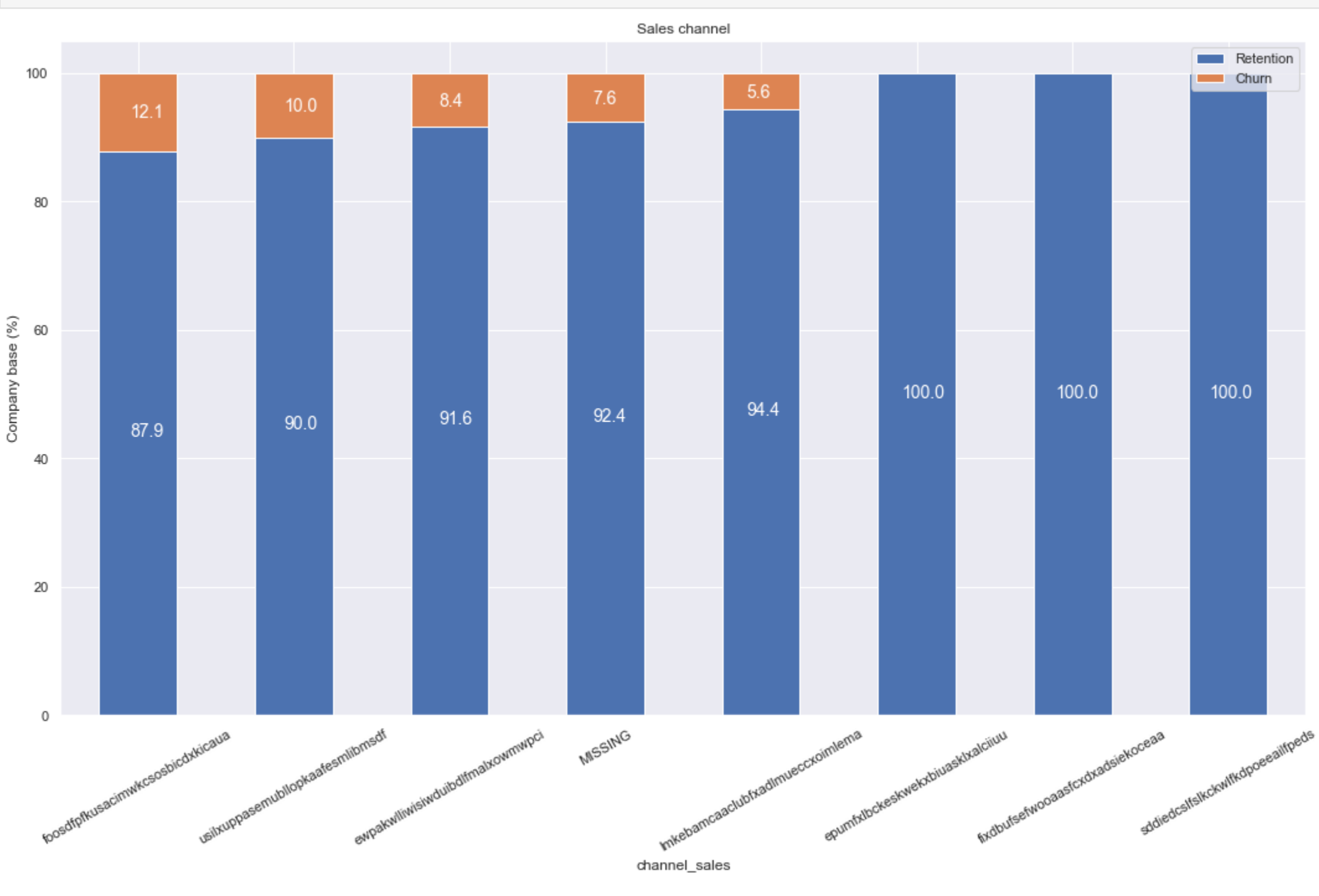
In [42]:
```python
#Churn
churn = client_df[['id','churn']]
churn.columns = ['Companies', 'churn']
churn_total = churn.groupby(churn['churn']).count()
churn_percentage = churn_total / churn_total.sum() * 100
```

In [43]:
```python
plot_stacked_bars(churn_percentage.transpose(), "Churning status", (5,5), legend_="lower right")
```



In [44]:
```python
# Sales channel
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 [45]:
```python
plot_stacked_bars(channel_churn, 'Sales channel', rot_=30)
```



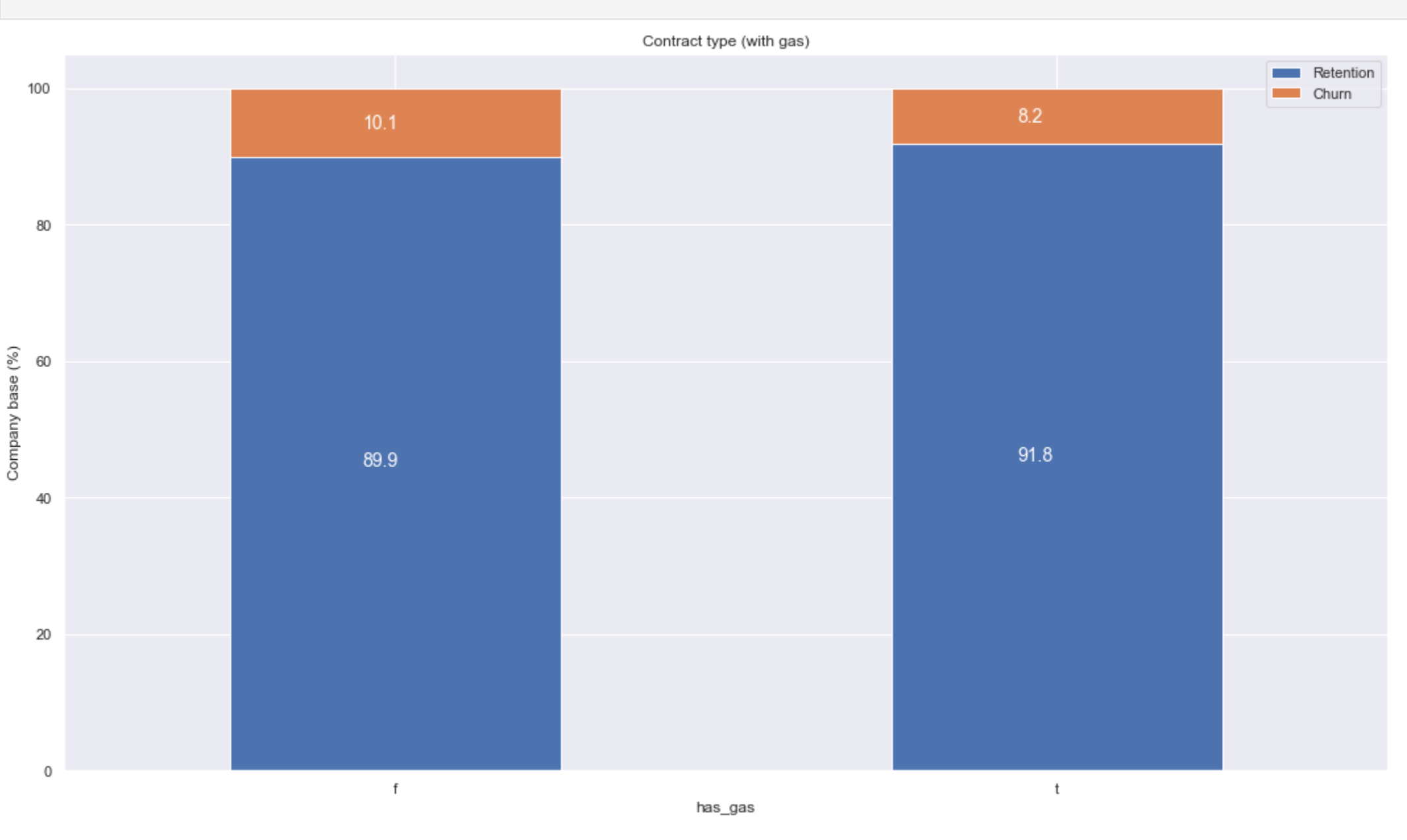
In [47]:
```python
# Consumption
consumption = client_df[['id', 'cons_12m', 'cons_gas_12m', 'cons_last_month', 'imp_cons', 'has_gas', 'churn']]
```

In [54]:
```python
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='plain', axis='x')
```

In [56]:
```python
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])
```

In [57]:
```python
# Contract Type

contract_type = client_df[['id', 'has_gas', 'churn']]
contract = contract_type.groupby([contract_type['churn'], contract_type['has_gas']])['id'].count().unstack(level=0)
contract_percentage = (contract.div(contract.sum(axis=1), axis=0) * 100).sort_values(by=[1], ascending=False)
```

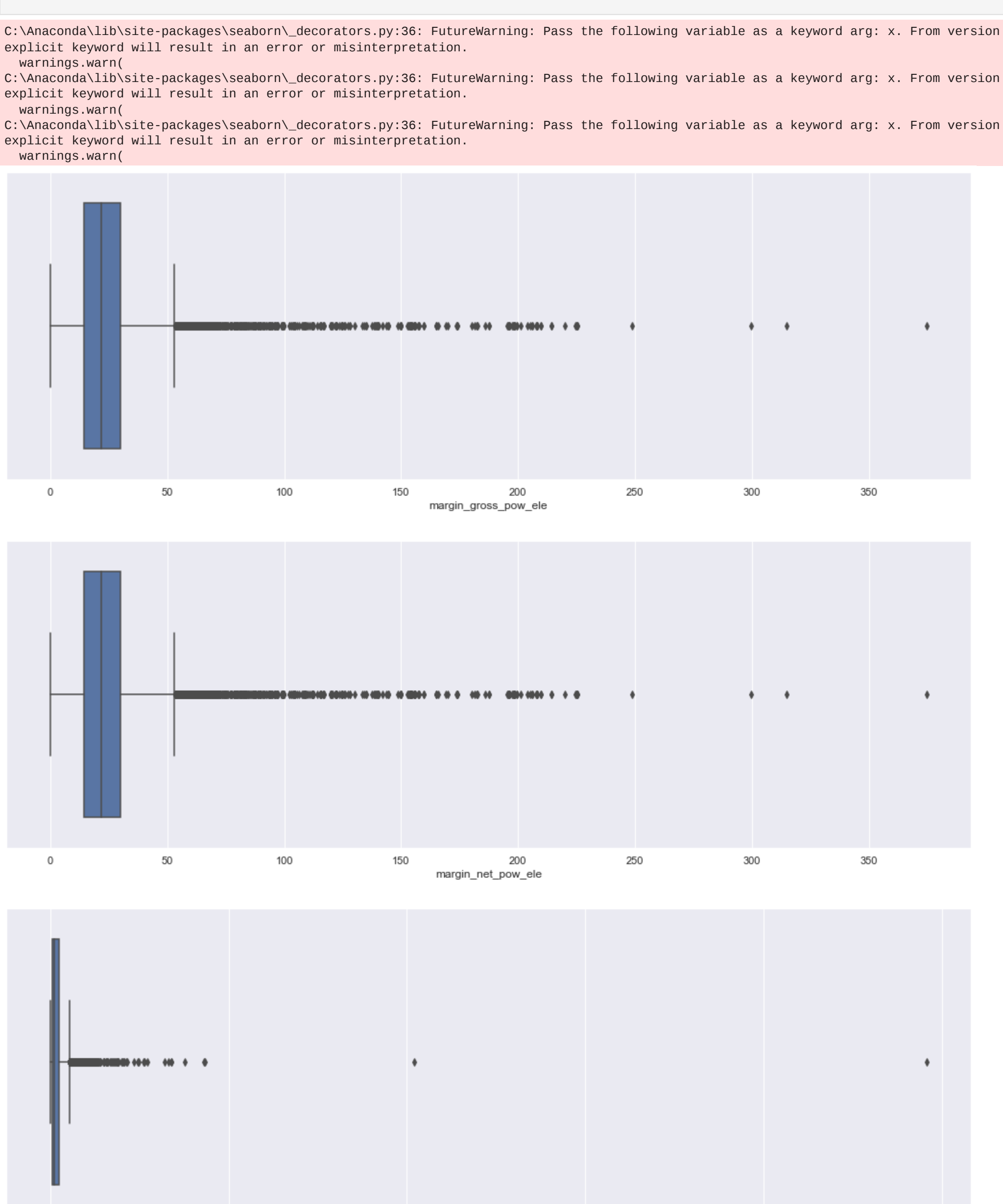
In [58]:
```python
plot_stacked_bars(contract_percentage, 'Contract type (with gas)')
```



In [61]:
```python
# Margin

margin = client_df[['id', 'margin_gross_pow_ele', 'margin_net_pow_ele', 'net_margin']]
```

In [67]:
```python
fig, axs = plt.subplots(nrows=3, figsize=(18,20))
sns.boxplot(margin["margin_gross_pow_ele"], ax=axs[0])
sns.boxplot(margin["margin_net_pow_ele"], ax=axs[1])
sns.boxplot(margin["net_margin"], ax=axs[2])

axs[0].ticklabel_format(style='plain',axis='x')
axs[1].ticklabel_format(style='plain',axis='x')
axs[2].ticklabel_format(style='plain',axis='x')
plt.show()
```

```
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [68]:
```python
# Subscribed Power

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

In [70]:
```python
fig, axs = plt.subplots(nrows=1, figsize=(18,10))
plot_distribution(power,'pow_max',axs)
```