Exploratory Data Analysis

---

1. Import packages
2. Loading data with Pandas
3. Descriptive statistics of data
4. Data visualization

---

1. Import packages

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Shows plots in jupyter notebook
%matplotlib inline
```

---

2. Loading data with Pandas

```
client_df = pd.read_csv('./client_data.csv')
```

```
client_df.head(3)
```

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	date_activ	date_end	date_modif_pr
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15	2016-06-15	2015-11
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0	2009-08-21	2016-08-30	2009-08
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0	2010-04-16	2016-04-16	2010-04

3 rows × 26 columns

3. Descriptive statistics of data

```
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
```

Statistics

```
client_df.describe()
```

	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_12m	forecast_cons_year	forecast_discount_energy	forecast_meter_rent_12m	f
count	1.460600e+04	1.460600e+04	14606.000000	14606.000000	14606.000000	14606.000000	14606.000000	
mean	1.592203e+05	2.809238e+04	16090.269752	1868.614880	1399.762906	0.966726	63.086871	
std	5.734653e+05	1.629731e+05	64364.196422	2387.571531	3247.786255	5.108289	66.165783	
min	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	5.674750e+03	0.000000e+00	0.000000	494.995000	0.000000	0.000000	16.180000	
50%	1.411550e+04	0.000000e+00	792.500000	1112.875000	314.000000	0.000000	18.795000	
75%	4.076375e+04	0.000000e+00	3383.000000	2401.790000	1745.750000	0.000000	131.030000	
max	6.207104e+06	4.154590e+06	771203.000000	82902.830000	175375.000000	30.000000	599.310000	

3. Data visualization

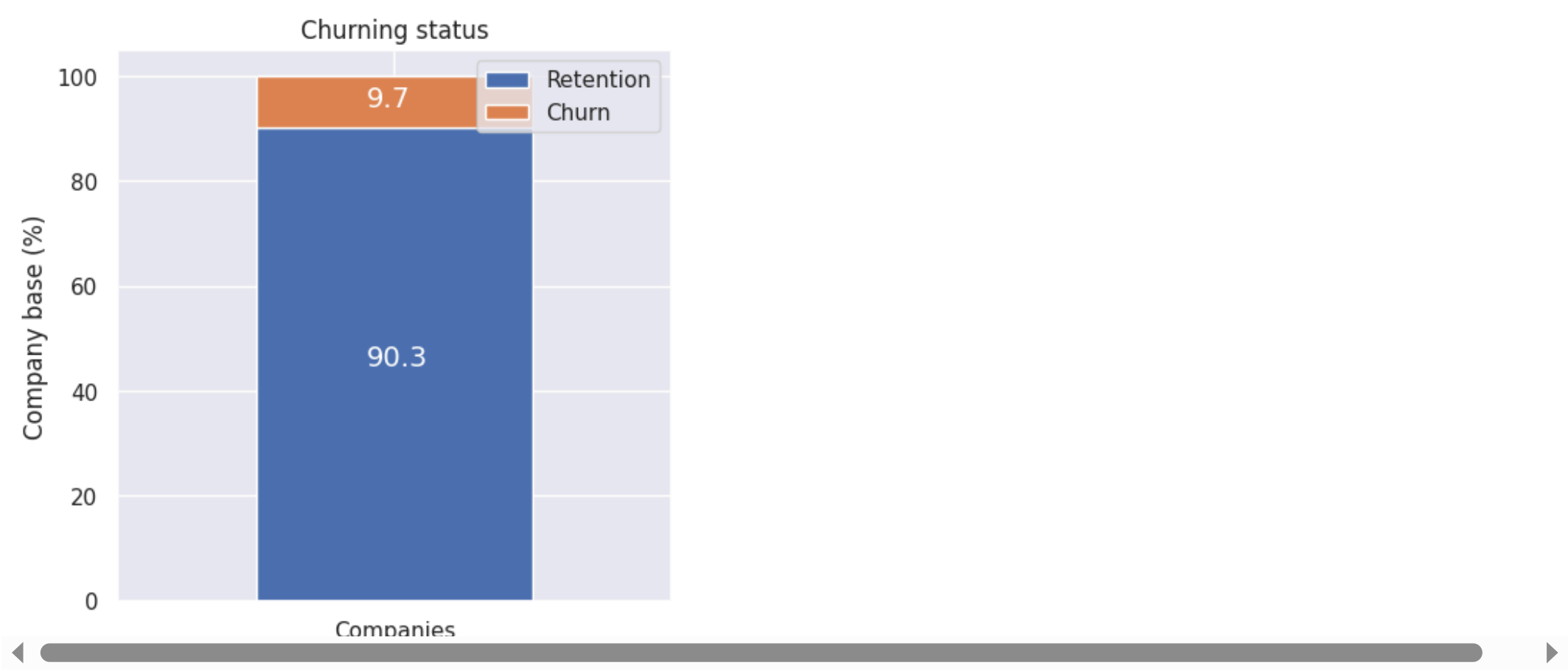
```
def plot_stacked_bars(dataframe, title_, size_=(18, 10), rot_=0, legend_="upper right"):
    """
    Plot stacked bars
    """
    ax = dataframe.plot(
        kind="bar",
```

```
        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
        )
```

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
```

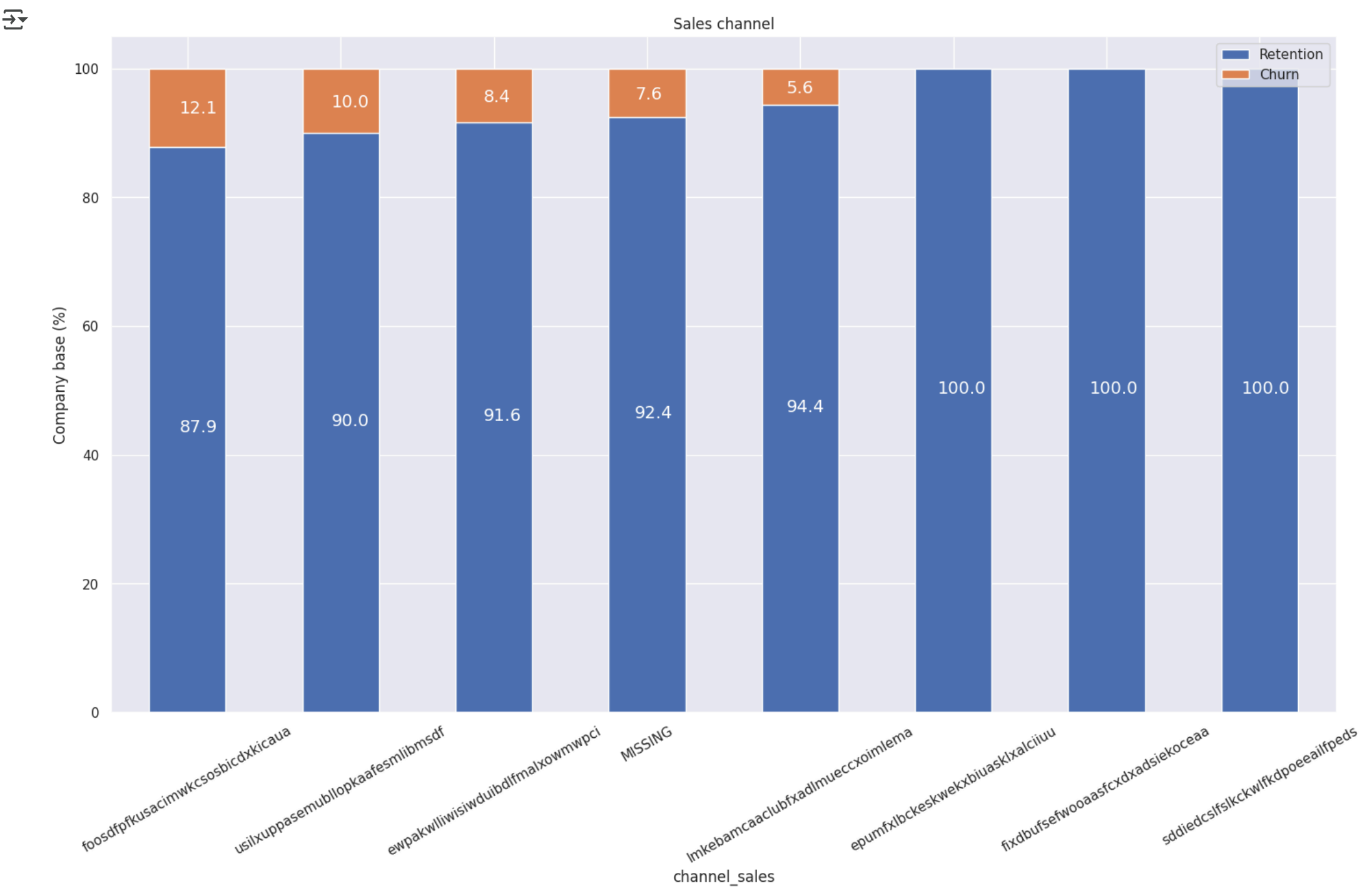
```
plot_stacked_bars(churn_percentage.transpose(), "Churning status", (5, 5))
```



About 10% of the total customers have churned.

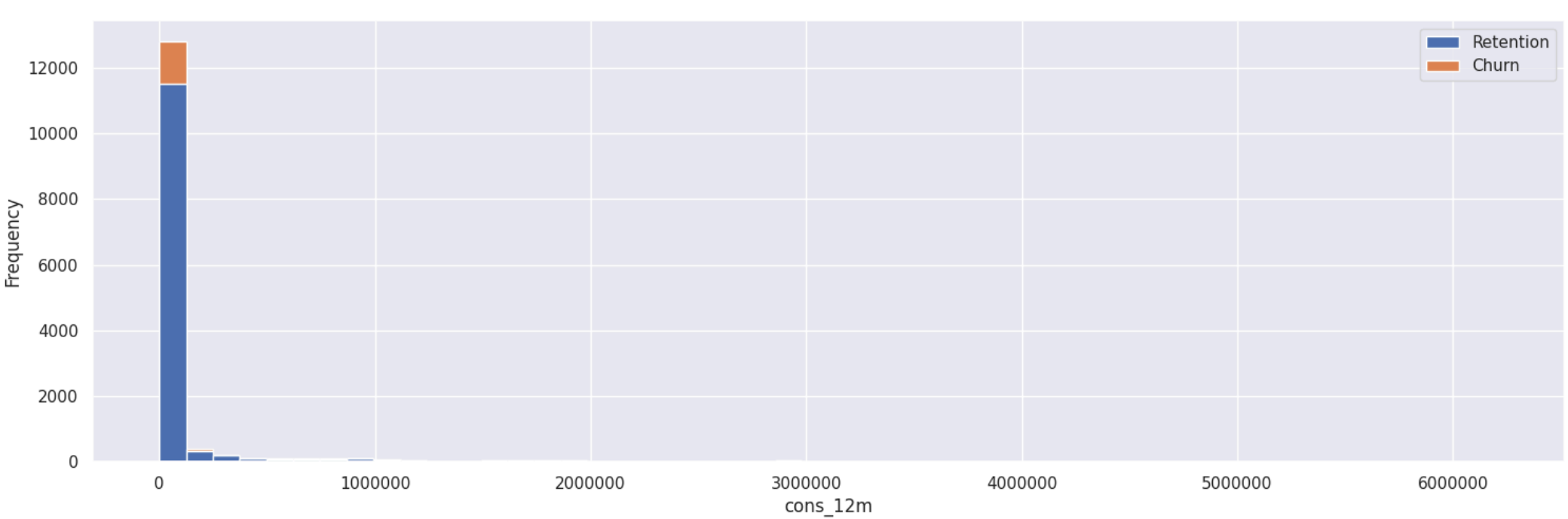
Sales channel

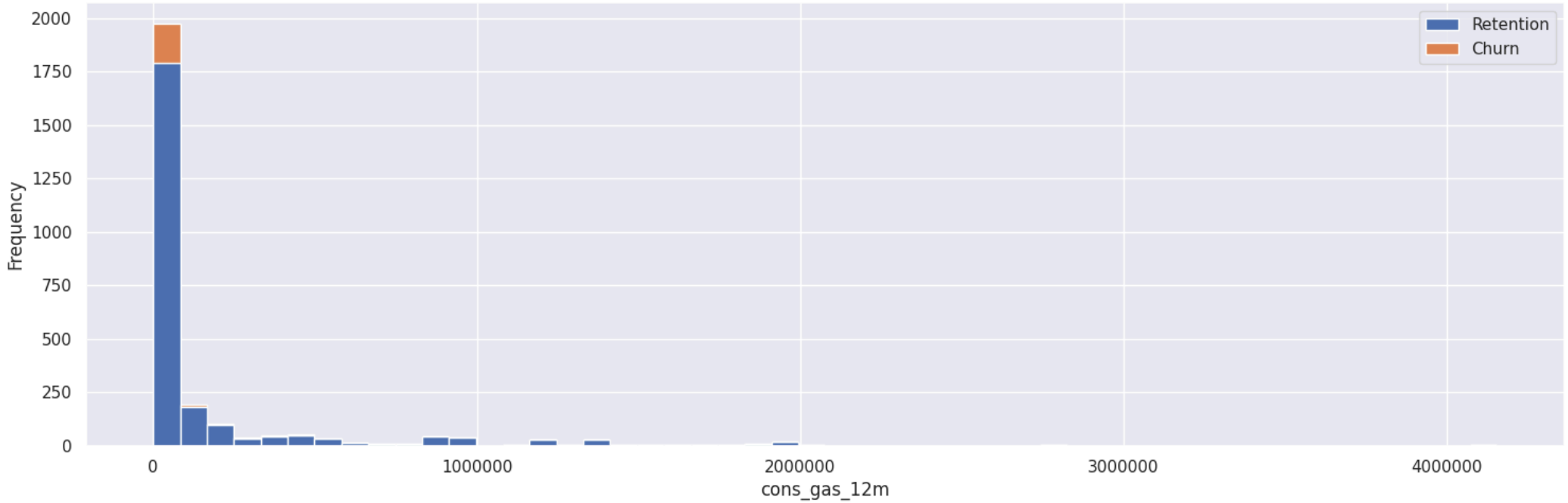
```
channel = client_df[['id', 'channel_sales', 'churn']]
channel = channel.groupby([channel['channel_sales'], channel['churn']])['id'].count().unstack(level=1).fillna(0)
channel_churn = (channel.div(channel.sum(axis=1), axis=0) * 100).sort_values(by=[1], ascending=False)


plot_stacked_bars(channel_churn, 'Sales channel', rot_=30)
```

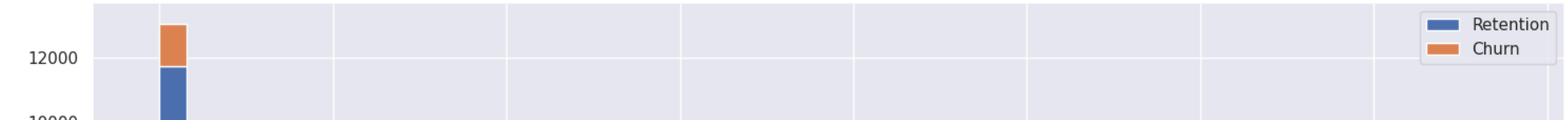

Sales channel
Retention
Churn
Company base (%)
0
20
40
60
80
100
12.1
10.0
8.4
7.6
5.6
87.9
90.0
91.6
92.4
94.4
100.0
100.0
100.0
foosdfpfkusacimwkcsosbicdxkicaua
usilxuppasemubllopkaafesmlibmsdf
ewpakwlliwisiwduibdlfmalxowmwpci
MISSING
lmkebamcaaclubfxadlmueccxoimlema
epumfxlbckeskwekxbiuasklxalciiuu
fixdbufsefwooaasfcxdxadsiekoceaa
sddiedcslfslkckwlfkdpoeeailfpeds
channel_sales

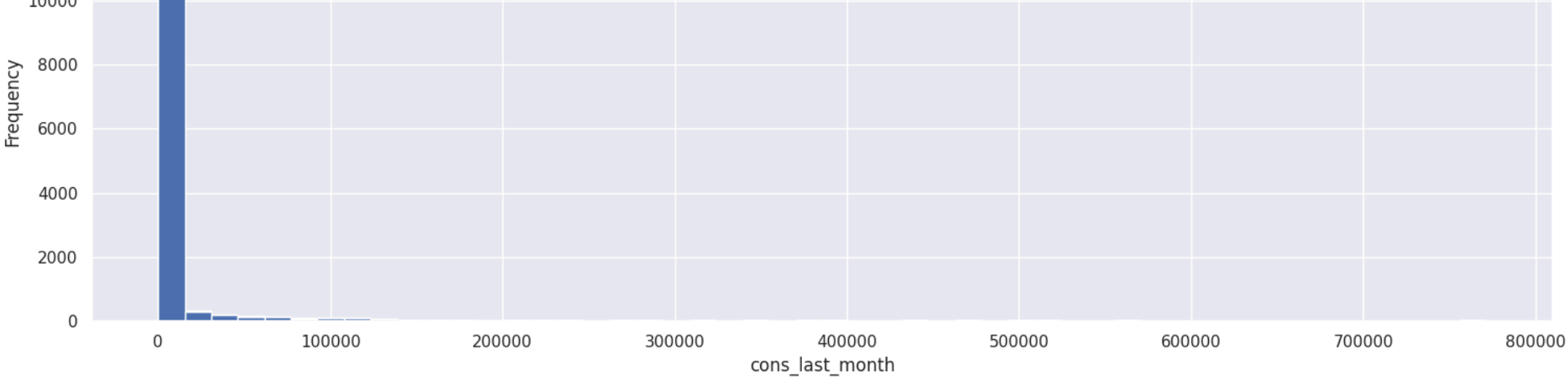
Consumption

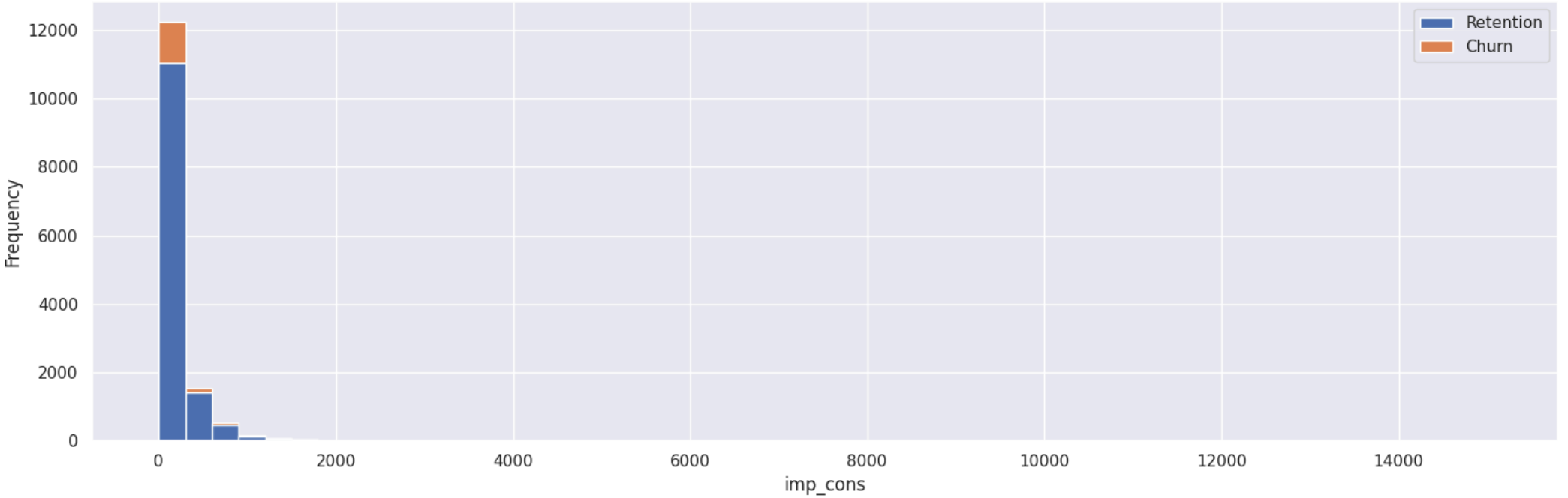
```
consumption = client_df[['id', 'cons_12m', 'cons_gas_12m', 'cons_last_month', 'imp_cons', 'has_gas', 'churn']]


def plot_distribution(dataframe, column, ax, bins_=50):
    """
    Plot variable distribution 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')


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])
```


Retention
Churn
Frequency
12000
10000
8000
6000
4000
2000
0
0
1000000
2000000
3000000
4000000
5000000
6000000
cons_12m


Retention
Churn
Frequency
2000
1750
1500
1250
1000
750
500
250
0
0
1000000
2000000
3000000
4000000
cons_gas_12m

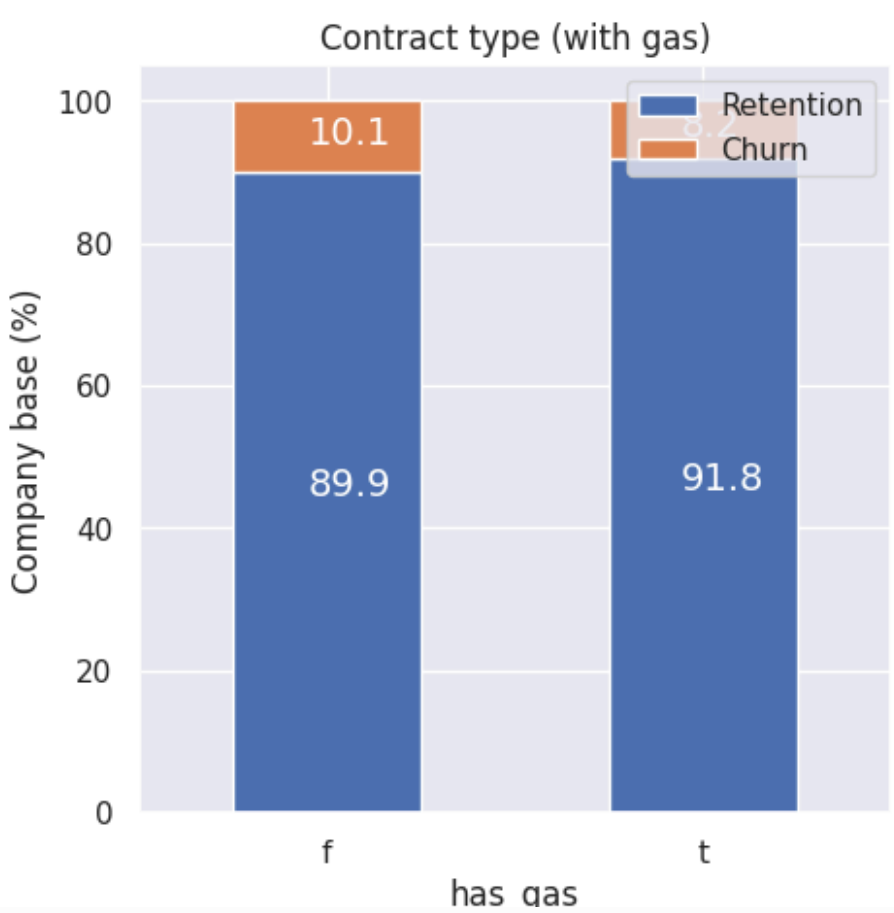

Retention
Churn
12000


Frequency
10000
8000
6000
4000
2000
0
0
100000
200000
300000
400000
500000
600000
700000
800000
cons_last_month


Retention
Churn
Frequency
12000
10000
8000
6000
4000
2000
0
0
2000
4000
6000
8000
10000
12000
14000
imp_cons

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)


plot_stacked_bars(contract_percentage, 'Contract type (with gas)',(5,5))
```



Margins

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


fig, axs = plt.subplots(nrows=1, figsize=(18, 10))
plot_distribution(power, 'pow_max', axs)
```