Data Cleaning and Exploratory Data Analysis

- By - Gautam Sharma

Import all required libraries

In [1]:
```
import pandas as pd
!pip install missingno
import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import zscore as zscore
```

```
Requirement already satisfied: missingno in c:\users\admin\anaconda3\li
b\site-packages (0.5.0)
Requirement already satisfied: numpy in c:\users\admin\anaconda3\lib\si
te-packages (from missingno) (1.18.1)
Requirement already satisfied: scipy in c:\users\admin\anaconda3\lib\si
te-packages (from missingno) (1.4.1)
Requirement already satisfied: matplotlib in c:\users\admin\anaconda3\l
ib\site-packages (from missingno) (3.1.3)
Requirement already satisfied: seaborn in c:\users\admin\anaconda3\lib
\site-packages (from missingno) (0.11.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\admin\anac
onda3\lib\site-packages (from matplotlib->missingno) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1
in c:\users\admin\anaconda3\lib\site-packages (from matplotlib->missing
no) (2.4.6)
Requirement already satisfied: cycler>=0.10 in c:\users\admin\anaconda3
\lib\site-packages (from matplotlib->missingno) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\admin\a
naconda3\lib\site-packages (from matplotlib->missingno) (2.8.1)
Requirement already satisfied: pandas>=0.23 in c:\users\admin\anaconda3
\lib\site-packages (from seaborn->missingno) (1.2.5)
Requirement already satisfied: setuptools in c:\users\admin\anaconda3\l
ib\site-packages (from kiwisolver>=1.0.1->matplotlib->missingno) (45.2.
0.post20200210)
Requirement already satisfied: six in c:\users\admin\anaconda3\lib\site
-packages (from cycler>=0.10->matplotlib->missingno) (1.14.0)
Requirement already satisfied: pytz>=2017.3 in c:\users\admin\anaconda3
\lib\site-packages (from pandas>=0.23->seaborn->missingno) (2021.1)
```

Data importing

In [2]:
```
 data_lst = ['date_activ', 'date_end','date_first_activ','date_modif_prod', 'date_renewal']
```

In [3]:
```
data_main = pd.read_csv('ml_case_training_data.csv', parse_dates=data_lst)
data_hist = pd.read_csv('ml_case_training_hist_data.csv', parse_dates=['price_date'])
data_output = pd.read_csv('ml_case_training_output.csv')
```

In [4]:
```
pd.set_option('display.max_columns', None)
```

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

Out[5]:

	id	activity_new	campaign_disc_ele	
0	48ada52261e7cf58715202705a0451c9	esoiiifxdlbkcsluxmfuacbdckommixw	NaN	lmkeb:
1	24011ae4ebbe3035111d65fa7c15bc57	NaN	NaN	foos
2	d29c2c54acc38ff3c0614d0a653813dd	NaN	NaN	
3	764c75f661154dac3a6c254cd082ea7d	NaN	NaN	foos
4	bba03439a292a1e166f80264c16191cb	NaN	NaN	lmkeb:

I . Data Exploration

The Output Dataset

- From thr output dataset we can derive a quick insights on customer retention.

In [6]:
```
# Replace the churn columns
data_output['churn'] = data_output['churn'].replace({0:'Stayed',1:'Churned'})
```

In [7]:
```
data_output.head(5)
```

Out[7]:

	id	churn
0	48ada52261e7cf58715202705a0451c9	Stayed
1	24011ae4ebbe3035111d65fa7c15bc57	Churned
2	d29c2c54acc38ff3c0614d0a653813dd	Stayed
3	764c75f661154dac3a6c254cd082ea7d	Stayed
4	bba03439a292a1e166f80264c16191cb	Stayed

In [8]:
```
# What number of customers have  churned in the last 3 moths
attrition_count = data_output['churn'].value_counts()
print('Total number of churned customer : \n ', attrition_count)
```

```
Total number of churned customer :
  Stayed     14501
Churned     1595
Name: churn, dtype: int64
```

- Last 3 moths 1595 customer have churned
- currently 14501 actively client

In [9]:
```
# Proportionof customer attrition in the last 3 months
attrition_rate = data_output['churn'].value_counts() / data_output.shape[
0] * 100

print('Attrition rates in last 3 months: \n', attrition_rate)
```

```
Attrition rates in last 3 months:
 Stayed     90.090706
Churned     9.909294
Name: churn, dtype: float64
```

- customer retenction in last 3 months 90.09%
- Customer attrition is 10 % in the last 3 months

The History Dataset

In [10]:
```
data_hist.head(5)
```

Out[10]:

	id	price_date	price_p1_var	price_p2_var	price_p3_var	pric
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	44
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	44
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	0.0	44
3	038af19179925da21a25619c5a24b745	2015-04-01	0.149626	0.0	0.0	44
4	038af19179925da21a25619c5a24b745	2015-05-01	0.149626	0.0	0.0	44

In [11]:
```
data_hist1 = data_hist[['id','price_date']]
```

In [12]:
```
data_hist1.head(5)
```

Out[12]:

	id	price_date
0	038af19179925da21a25619c5a24b745	2015-01-01
1	038af19179925da21a25619c5a24b745	2015-02-01
2	038af19179925da21a25619c5a24b745	2015-03-01
3	038af19179925da21a25619c5a24b745	2015-04-01
4	038af19179925da21a25619c5a24b745	2015-05-01

In [13]:
```
# Examine the hist_data
data_hist1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193002 entries, 0 to 193001
Data columns (total 2 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   id          193002 non-null  object
 1   price_date  193002 non-null  datetime64[ns]
dtypes: datetime64[ns](1), object(1)
memory usage: 2.9+ MB
```

In [14]:
```
data_hist.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  datetime64[ns]
 2   price_p1_var  191643 non-null  float64
 3   price_p2_var  191643 non-null  float64
 4   price_p3_var  191643 non-null  float64
 5   price_p1_fix  191643 non-null  float64
 6   price_p2_fix  191643 non-null  float64
 7   price_p3_fix  191643 non-null  float64
dtypes: datetime64[ns](1), float64(6), object(1)
memory usage: 11.8+ MB
```

In [15]:
```
data_hist.describe()
```

Out[15]:

	price_p1_var	price_p2_var	price_p3_var	price_p1_fix	price_p2_fix	price_p3_f
count	191643.000000	191643.000000	191643.000000	191643.000000	191643.000000	191643.0000(
mean	0.140991	0.054412	0.030712	43.325546	10.698201	6.4554:
std	0.025117	0.050033	0.036335	5.437952	12.856046	7.7822
min	0.000000	0.000000	0.000000	-0.177779	-0.097752	-0.0651
25%	0.125976	0.000000	0.000000	40.728885	0.000000	0.0000(
50%	0.146033	0.085483	0.000000	44.266930	0.000000	0.0000(
75%	0.151635	0.101780	0.072558	44.444710	24.339581	16.2263
max	0.280700	0.229788	0.114102	59.444710	36.490692	17.4582

In [16]:
```
# Identify the nylity of the DataFrame
missing_values_hist = data_hist.isnull().sum()
print('Total missing data in data_hist: \n',missing_values_hist)
```

```
Total missing data in data_hist: 
 id              0
price_date      0
price_p1_var    1359
price_p2_var    1359
price_p3_var    1359
price_p1_fix    1359
price_p2_fix    1359
price_p3_fix    1359
dtype: int64
```

In [17]:
```
# Identify the percentage of nullity in the dataframe for each column
missing_values_hist_per = data_hist.isnull().mean() * 100
print('The Total percentage of missing values: \n', missing_values_hist_p
er)
```

```
The Total percentage of missing values:
 id               0.000000
price_date      0.000000
price_p1_var    0.704138
price_p2_var    0.704138
price_p3_var    0.704138
price_p1_fix    0.704138
price_p2_fix    0.704138
price_p3_fix    0.704138
dtype: float64
```

The main Dataset

In [18]:
```
data_main.head(5)
```

Out[18]:

	id	activity_new	campaign_disc_ele	
0	48ada52261e7cf58715202705a0451c9	esoiiifxdlbkcsluxmfuacbdckommixw	NaN	lmkeba
1	24011ae4ebbe3035111d65fa7c15bc57	NaN	NaN	foos
2	d29c2c54acc38ff3c0614d0a653813dd	NaN	NaN	
3	764c75f661154dac3a6c254cd082ea7d	NaN	NaN	foos
4	bba03439a292a1e166f80264c16191cb	NaN	NaN	lmkeba

- The dataset contain more characteristics about each client'a account and activity.

In [19]:
```
data_main.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16096 entries, 0 to 16095
Data columns (total 32 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   id                        16096 non-null  object
 1   activity_new              6551 non-null   object
 2   campaign_disc_ele         0 non-null      float64
 3   channel_sales             11878 non-null  object
 4   cons_12m                  16096 non-null  int64
 5   cons_gas_12m              16096 non-null  int64
 6   cons_last_month           16096 non-null  int64
 7   date_activ                16096 non-null  datetime64[ns]
 8   date_end                  16094 non-null  datetime64[ns]
 9   date_first_activ          3508 non-null   datetime64[ns]
 10  date_modif_prod           15939 non-null  datetime64[ns]
 11  date_renewal              16056 non-null  datetime64[ns]
 12  forecast_base_bill_ele    3508 non-null   float64
 13  forecast_base_bill_year   3508 non-null   float64
 14  forecast_bill_12m         3508 non-null   float64
 15  forecast_cons             3508 non-null   float64
 16  forecast_cons_12m         16096 non-null  float64
 17  forecast_cons_year        16096 non-null  int64
 18  forecast_discount_energy  15970 non-null  float64
 19  forecast_meter_rent_12m   16096 non-null  float64
 20  forecast_price_energy_p1  15970 non-null  float64
 21  forecast_price_energy_p2  15970 non-null  float64
 22  forecast_price_pow_p1     15970 non-null  float64
 23  has_gas                   16096 non-null  object
 24  imp_cons                  16096 non-null  float64
 25  margin_gross_pow_ele      16083 non-null  float64
 26  margin_net_pow_ele        16083 non-null  float64
 27  nb_prod_act               16096 non-null  int64
 28  net_margin                16081 non-null  float64
 29  num_years_antig           16096 non-null  int64
 30  origin_up                 16009 non-null  object
 31  pow_max                   16093 non-null  float64
dtypes: datetime64[ns](5), float64(16), int64(6), object(5)
memory usage: 3.9+ MB
```

In [20]:
```
# Identify the percentage of nullity in the dataframe for each columns
missing_values_main_par = data_main.isnull().mean() * 100
print('Percentage of Missing values: \n', missing_values_main_par)
```

```
Percentage of Missing values: 
 id                           0.000000
activity_new                59.300447
campaign_disc_ele          100.000000
channel_sales               26.205268
cons_12m                     0.000000
cons_gas_12m                 0.000000
cons_last_month              0.000000
date_activ                   0.000000
date_end                     0.012425
date_first_activ            78.205765
date_modif_prod              0.975398
date_renewal                 0.248509
forecast_base_bill_ele      78.205765
forecast_base_bill_year     78.205765
forecast_bill_12m           78.205765
forecast_cons               78.205765
forecast_cons_12m            0.000000
forecast_cons_year           0.000000
forecast_discount_energy     0.782803
forecast_meter_rent_12m      0.000000
forecast_price_energy_p1     0.782803
forecast_price_energy_p2     0.782803
forecast_price_pow_p1        0.782803
has_gas                      0.000000
imp_cons                     0.000000
margin_gross_pow_ele         0.080765
margin_net_pow_ele           0.080765
nb_prod_act                  0.000000
net_margin                   0.093191
num_years_antig              0.000000
origin_up                    0.540507
pow_max                      0.018638
dtype: float64
```

In [21]:
```
# Examine the statistics of main dataset
data_main.describe()
```

Out[21]:

	campaign_disc_ele	cons_12m	cons_gas_12m	cons_last_month	forecast_base_bill_ele
count	0.0	1.609600e+04	1.609600e+04	1.609600e+04	3508.00000(
mean	NaN	1.948044e+05	3.191164e+04	1.946154e+04	335.843857
std	NaN	6.795151e+05	1.775885e+05	8.235676e+04	649.40600(
min	NaN	-1.252760e+05	-3.037000e+03	-9.138600e+04	-364.94000(
25%	NaN	5.906250e+03	0.000000e+00	0.000000e+00	0.00000(
50%	NaN	1.533250e+04	0.000000e+00	9.010000e+02	162.95500(
75%	NaN	5.022150e+04	0.000000e+00	4.127000e+03	396.18500(
max	NaN	1.609711e+07	4.188440e+06	4.538720e+06	12566.08000(

- The average net margin is $ 217
- The average num_years_antig is 5 years

II. Data Cleaning and Imputation

- Here we dealing with missing dta and workflow for treating missing values

The History Dataset

In [22]:
```
data_hist.head(2)
```

Out[22]:

	id	price_date	price_p1_var	price_p2_var	price_p3_var	pric
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	4
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	4

In [23]:
```
# Indentify the negative columns
negative_col = ['price_p1_fix','price_p2_fix','price_p3_fix']
```

In [24]:
```
data_hist[negative_col] = data_hist[negative_col].apply(abs)
```

In [25]:
```
data_hist.describe()
```

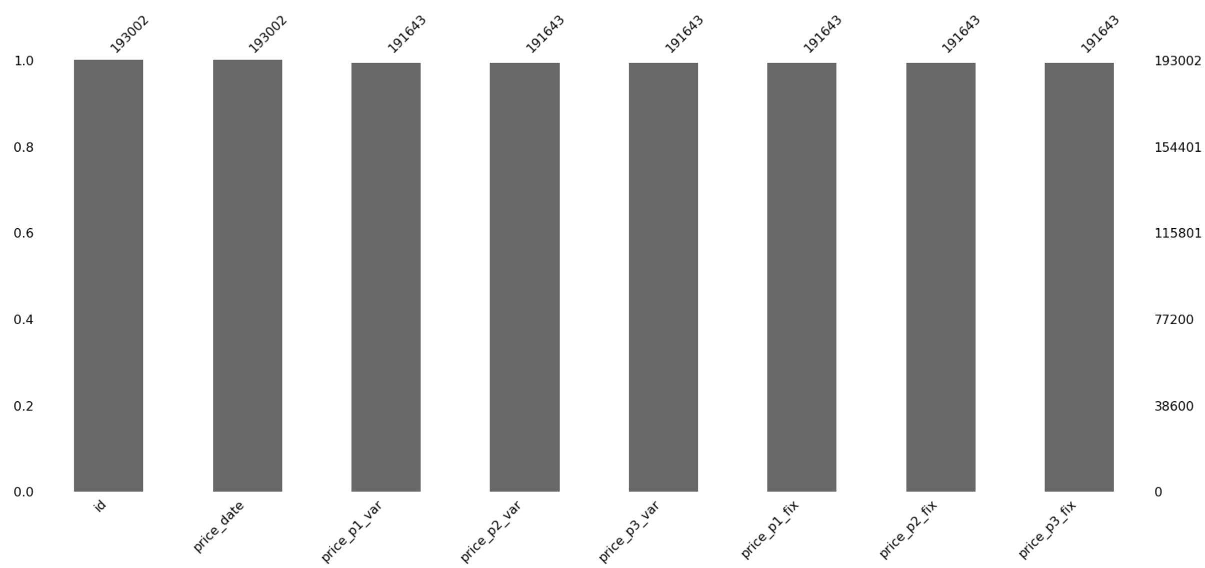
Out[25]:

	price_p1_var	price_p2_var	price_p3_var	price_p1_fix	price_p2_fix	price_p3_f
count	191643.000000	191643.000000	191643.000000	191643.000000	191643.000000	191643.0000(
mean	0.140991	0.054412	0.030712	43.325563	10.698210	6.4554
std	0.025117	0.050033	0.036335	5.437816	12.856039	7.7822
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000(
25%	0.125976	0.000000	0.000000	40.728885	0.000000	0.0000(
50%	0.146033	0.085483	0.000000	44.266930	0.000000	0.0000(
75%	0.151635	0.101780	0.072558	44.444710	24.339581	16.2263
max	0.280700	0.229788	0.114102	59.444710	36.490692	17.4582

Visualizing the amount of missingness

In [26]:
```
# Visualizethe completeness of the data frame
msno.bar(data_hist)
```

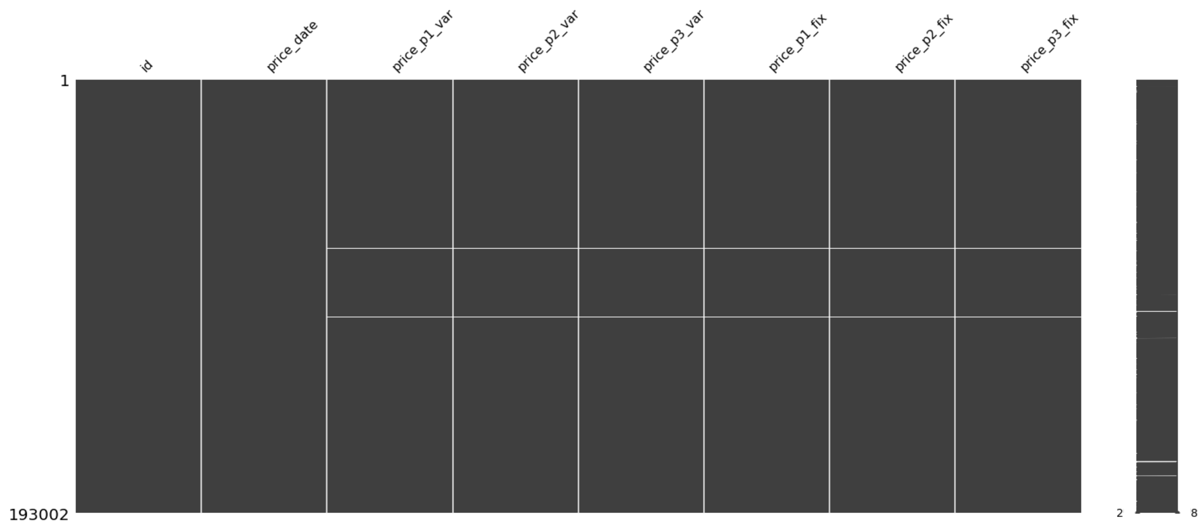
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x219b0354308>



As i see the data we absorb no data is missing but after the the imputation , we estimated that 0.7 % of data is missing

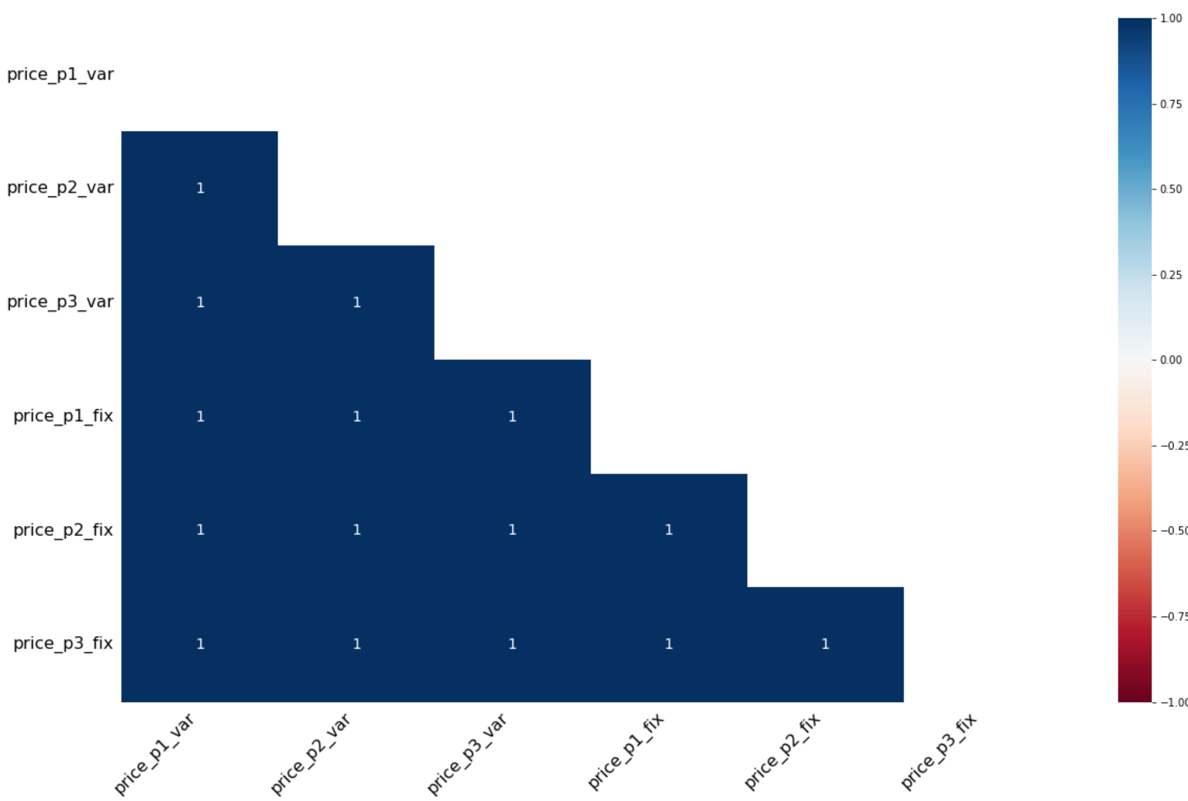
In [27]:
```
# Visua;ize the locations of the missing values of the dataset
sorted = data_hist.sort_values(by = ['id', 'price_date'])
msno.matrix(sorted)
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x219b1560a08>



In [28]:
```
# Visualize the correlation between the numeric variables of the Datafram
e
msno.heatmap(data_hist)
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x219b15cca48>



In [29]:
```
# Identify the index of the IDs containing missing values
hist_NAN_index = data_hist[data_hist.isnull().any(axis=1)].index.values.t
olist()
```

In [30]:
```
# Obtain the Dataframe with missing values
data_hist_missing = data_hist.iloc[hist_NAN_index,:]
```

In [31]:
```
data_hist_missing.head(5)
```

Out[31]:

	id	price_date	price_p1_var	price_p2_var	price_p3_var	pı
75	ef716222bbd97a8bdfcbb831e3575560	2015-04-01	NaN	NaN	NaN	
221	0f5231100b2febab862f8dd8eaab3f43	2015-06-01	NaN	NaN	NaN	
377	2f93639de582fadfbe3e86ce1c8d8f35	2015-06-01	NaN	NaN	NaN	
413	f83c1ab1ca1d1802bb1df4d72820243c	2015-06-01	NaN	NaN	NaN	
461	3076c6d4a060e12a049d1700d9b09cf3	2015-06-01	NaN	NaN	NaN	

In [32]:
```
# Extract the unique dates of missing date
date_lst = data_hist_missing['price_date'].unique()
id_lst = data_hist_missing['id'].unique()
```

In [33]:
```
# Create a time dataframe with the unique dates
time_data = pd.DataFrame(data=date_lst, columns=['price_date'])
```

In [34]:
```
time_data.head(5)
```

Out[34]:

	price_date
0	2015-04-01
1	2015-06-01
2	2015-05-01
3	2015-08-01
4	2015-09-01

- there are 1359 clients who are missing price data at least in 1 months
- There is hogh correlation between the missingness in the numeric and is values, missing or non-missing.

Time series Data

In [35]:
```
# Make a copy of Data_hist dataset
data_hist_ff = data_hist.copy(deep=True)
```

In [36]:
```
# Print prior to imputing missing values
print(data_hist_ff.iloc[hist_NAN_index,3:9].head())
```

```
     price_p2_var  price_p3_var  price_p1_fix  price_p2_fix  price_p3_f
ix
75            NaN           NaN           NaN           NaN           N
aN
221           NaN           NaN           NaN           NaN           N
aN
377           NaN           NaN           NaN           NaN           N
aN
413           NaN           NaN           NaN           NaN           N
aN
461           NaN           NaN           NaN           NaN           N
aN
```

In [37]:
```
# FIll NAN using forward fill
data_hist_ff.fillna(method='ffill', inplace=True)
```

In [38]:
```
print(data_hist_ff.iloc[hist_NAN_index,3:9].head(5))
```

```
     price_p2_var  price_p3_var  price_p1_fix  price_p2_fix  price_p3_f
ix
75       0.000000      0.000000     44.266931      0.000000      0.0000
00
221      0.000000      0.000000     44.266931      0.000000      0.0000
00
377      0.087970      0.000000     44.266931      0.000000      0.0000
00
413      0.102239      0.070381     40.565969     24.339581     16.2263
89
461      0.000000      0.000000     44.266931      0.000000      0.0000
00
```

In [39]:
```
data_hist_ff.describe()
```

Out[39]:

	price_p1_var	price_p2_var	price_p3_var	price_p1_fix	price_p2_fix	price_p3_f
count	193002.000000	193002.000000	193002.000000	193002.000000	193002.000000	193002.0000(
mean	0.141006	0.054376	0.030689	43.326213	10.689406	6.4504(
std	0.025091	0.050040	0.036333	5.431161	12.853850	7.7813:
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000(
25%	0.125976	0.000000	0.000000	40.728885	0.000000	0.0000(
50%	0.146033	0.085450	0.000000	44.266930	0.000000	0.0000(
75%	0.151635	0.101780	0.072558	44.444710	24.339581	16.2263(
max	0.280700	0.229788	0.114102	59.444710	36.490692	17.4582:

In [40]:
```
# Merger output dataset with historical forwARD FILL DATASET
data_hist_ff_merged = data_hist_ff.merge(right=data_output, on=['id'])
```

In [41]:
```
data_hist_ff_merged.head(5)
```

Out[41]:

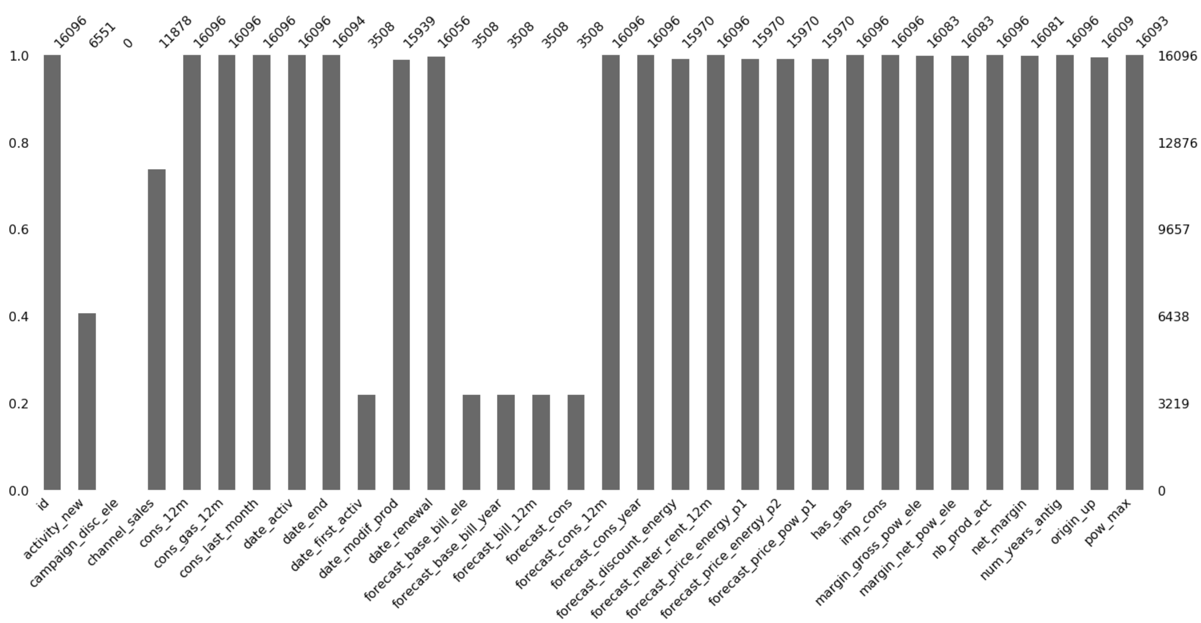
	id	price_date	price_p1_var	price_p2_var	price_p3_var	pric
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	44
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	44
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	0.0	44
3	038af19179925da21a25619c5a24b745	2015-04-01	0.149626	0.0	0.0	44
4	038af19179925da21a25619c5a24b745	2015-05-01	0.149626	0.0	0.0	44

The Main Dataset

Visualizing the amountof missingness

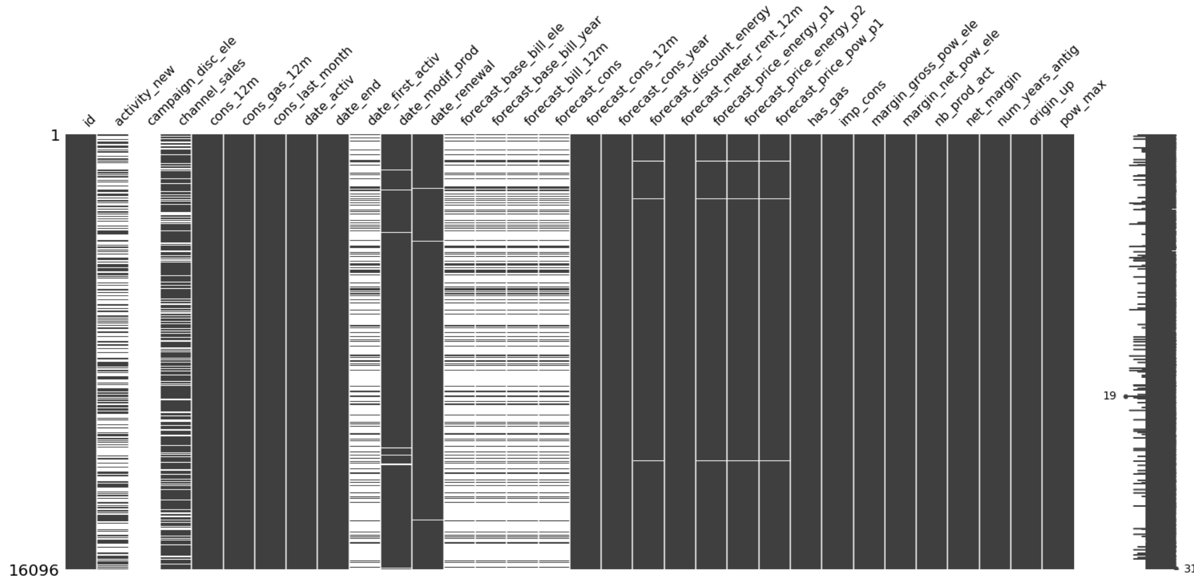
In [42]:
```
# Visualize the completeness of the dataframe
msno.bar(data_main)
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x219b1681a08>



In [43]:
```python
# Visualize the locations of missing data
msno.matrix(data_main)
```

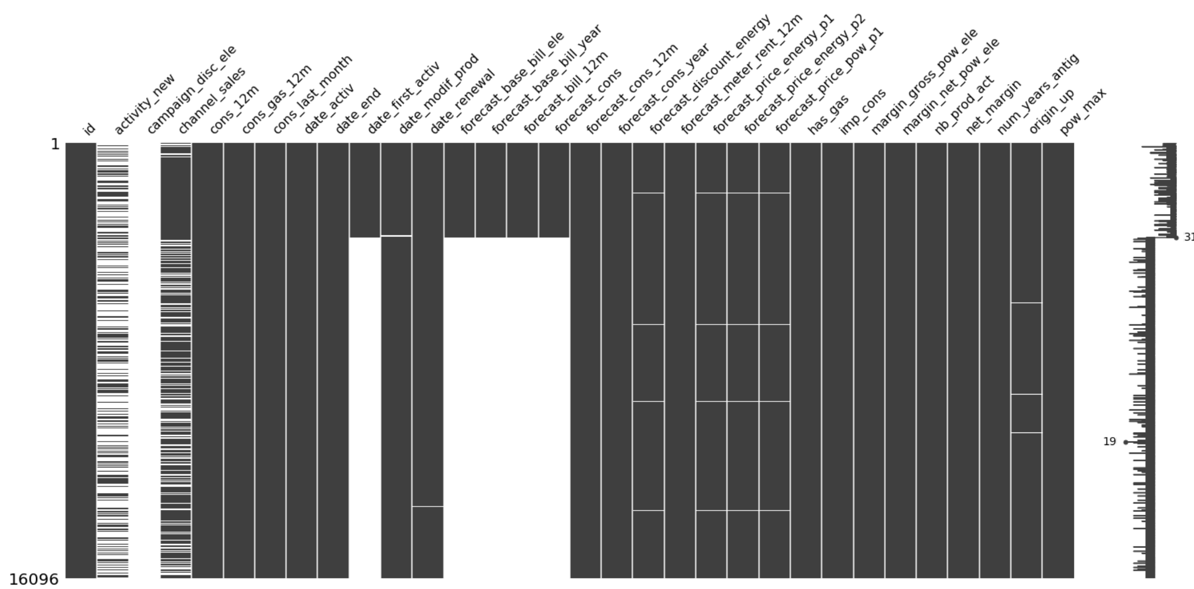
Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x219b3cdcd08>



In [44]:
```python
sorted_main = data_main.sort_values('date_first_activ')
```

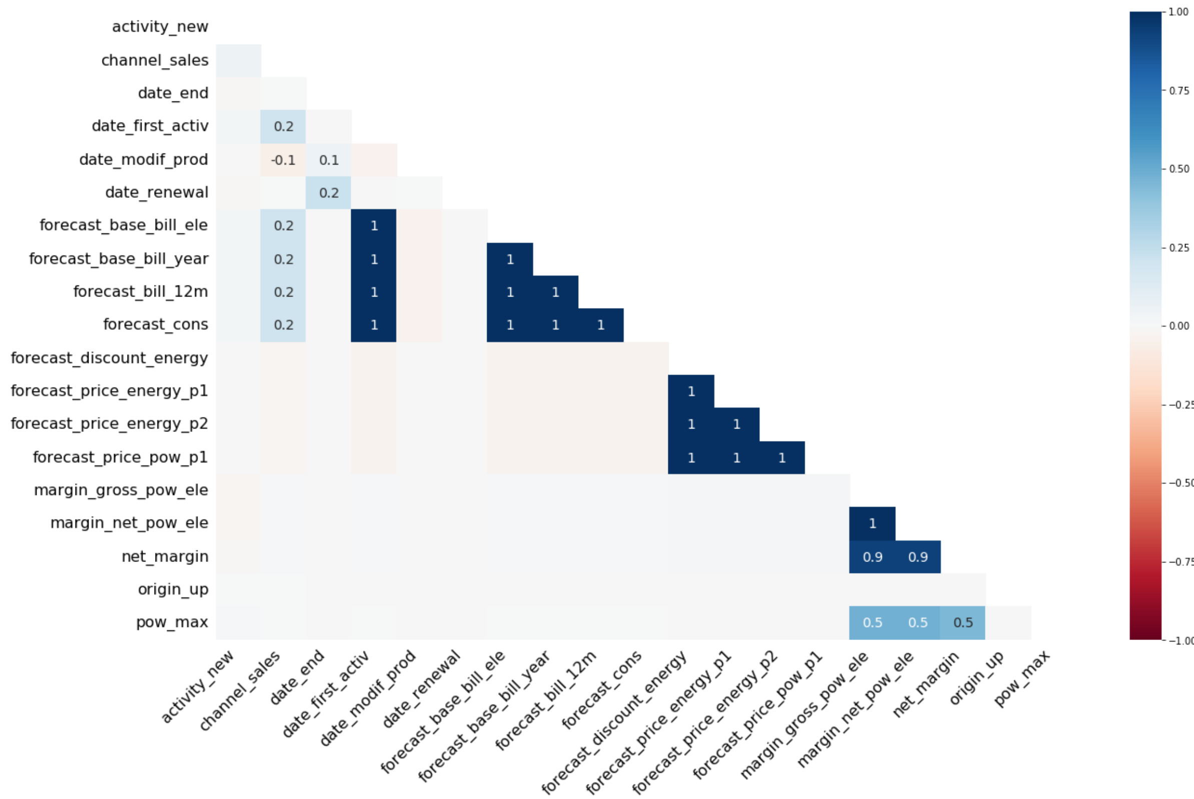
In [45]:
```python
msno.matrix(sorted_main)
```

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x219b3c98408>



In [46]:
```
msno.heatmap(data_main)
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x219b45da188>



In [47]:
```
# Demonstrate why the date_activ column cannot replace completely date_fi
rst_activ
activity = ['date_activ','date_first_activ']
```

In [48]:
```
# Filter the columns of interest
data_activity = data_main[activity]
```

In [49]:
```
# Obtain only the complete interest
data_activity_cc = data_activity.dropna(subset=['date_first_activ'],how=
'any',inplace=False)
```

In [50]:
```
# Test wether two objects contain the same elements
data_activity_cc.date_activ.equals(data_activity_cc.date_first_activ)
```

Out[50]: False

In [51]:
```
# Describe the data
data_activity_cc.describe(datetime_is_numeric=True)
```

Out[51]:

	date_activ	date_first_activ
count	3508	3508
mean	2011-09-03 07:45:05.131128832	2011-06-19 20:20:23.261117440
min	2003-09-23 00:00:00	2001-01-10 00:00:00
25%	2010-10-26 00:00:00	2010-08-04 18:00:00
50%	2012-01-03 00:00:00	2011-10-28 00:00:00
75%	2012-08-08 00:00:00	2012-06-22 06:00:00
max	2014-09-01 00:00:00	2014-09-01 00:00:00

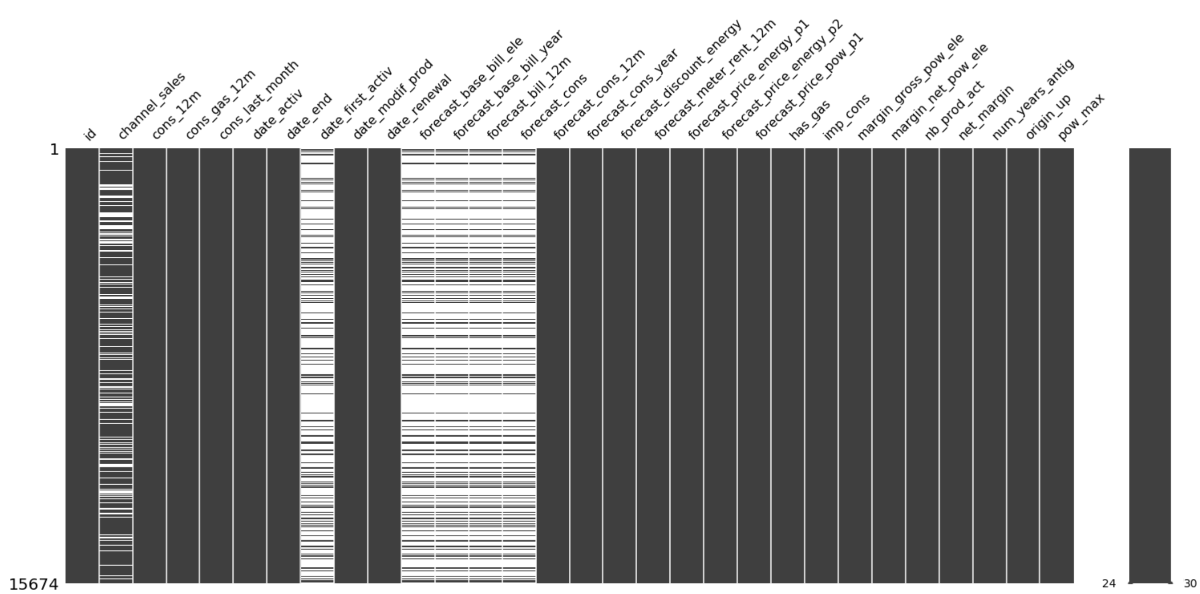
In [52]:
```
# Drop the column activity_new and campaign_disc_elec
data_main_drop = data_main.drop(labels=['activity_new', 'campaign_disc_el
e'],axis=1)
```

In [53]:
```
# Remove date_end date_modif_prod date_renewal origin_up pow_max margin_g
ross_pow_ele margin_net_pow_ele net_margin
brush = ['date_end','date_modif_prod','date_renewal','origin_up','pow_ma
x','margin_gross_pow_ele','margin_net_pow_ele', 'net_margin','forecast_di
scount_energy','forecast_price_energy_p1','forecast_price_energy_p2','for
ecast_price_pow_p1']
```

In [54]:
```
data_main_drop.dropna(subset=brush, how='any', inplace=True)
```

In [55]:
```
msno.matrix(data_main_drop)
```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x219b5eb5b88>



In [56]:
```
# Choose the columns without missing values
incomplete_cols = ['channel_sales','date_first_activ','forecast_base_bill
_ele','forecast_base_bill_year','forecast_bill_12m','forecast_cons']
```

In [57]:
```
complete_cols = [ column_name for column_name in data_main_drop.columns i
f column_name not in incomplete_cols]
```

In [58]:
```
data_main_cc = data_main_drop[complete_cols]
```

In [59]:
```
# Fix negative numeric variables
numeric = [column_name for column_name in data_main_cc.columns
          if data_main_cc[column_name].dtype == 'float64' or data_main_cc
[column_name].dtype == 'int64']
```

In [60]:
```
# Overwrite positive values on negative values
data_main_cc[numeric] = data_main_cc[numeric].apply(abs)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\pandas\core\frame.py:3191: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[k1] = value[k2]
```

In [61]:
```
# Describe
data_main_cc.describe()
```

Out[61]:

	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_12m	forecast_cons_year	f
count	1.567400e+04	1.567400e+04	1.567400e+04	15674.000000	15674.000000	
mean	1.916143e+05	3.132400e+04	1.941588e+04	2359.676441	1911.698354	
std	6.724688e+05	1.716291e+05	8.226881e+04	3979.605687	5224.813531	
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000	
25%	5.893250e+03	0.000000e+00	0.000000e+00	514.045000	0.000000	
50%	1.522000e+04	0.000000e+00	9.090000e+02	1178.970000	382.000000	
75%	4.953825e+04	0.000000e+00	4.131500e+03	2677.220000	1994.750000	
max	1.609711e+07	4.154590e+06	4.538720e+06	103801.930000	175375.000000	

In [62]:
```
# Convert the has_gas column to Yes/No
data_main_cc['has_gas'] = data_main_cc['has_gas'].replace({'t':'Yes', 'f'
:'No'})
```

```
C:\Users\Admin\anaconda3\lib\site-packages\ipykernel_launcher.py:2: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [63]:
```
# Merger the main dataset withthe output dataset
data_main_cc_merged = data_main_cc.merge(right=data_output, on=['id'])
```

In [64]:
```
# Convet the churn column to churned or stayed
data_main_cc_merged['churn'] = data_main_cc_merged['churn'].replace({1:'C
hurned',0:'Stayed'})
```

In [65]:
```
data_main_cc_merged.head(5)
```

Out[65]:

	id	cons_12m	cons_gas_12m	cons_last_month	date_activ	d
0	48ada52261e7cf58715202705a0451c9	309275	0	10025	2012-11-07	:
1	d29c2c54acc38ff3c0614d0a653813dd	4660	0	0	2009-08-21	:
2	764c75f661154dac3a6c254cd082ea7d	544	0	0	2010-04-16	:
3	bba03439a292a1e166f80264c16191cb	1584	0	0	2010-03-30	:
4	568bb38a1afd7c0fc49c77b3789b59a3	121335	0	12400	2010-04-08	:

In [66]:
```
# obtain all the variables except for id
variables = [column_name for column_name in data_main_cc_merged.columns i
f column_name != 'id']
```

In [67]:
```
#Obtain all the categorical variables except for id
categorical = [column_name for column_name in variables if data_main_cc_m
erged[column_name].dtype == 'object']
```

In [68]:
```
# Obtain all the Data Variables
dates = [column_name for column_name in variables if data_main_cc_merged[
column_name].dtype == 'datetime64[ns]']
```

In [69]:
```
# Obtain all the numeric columns
numeric = [column_name for column_name in variables if column_name not in categorical and column_name != 'id' and
          column_name != 'churn'
          and column_name not in dates]
```

Data Visualization

- Let's visualize what we've found

In [70]:
```
# Calculate the zcores of tenure
tenure_zcores = zscore(a=data_main_cc_merged['num_years_antig'])
```

In [71]:
```
# Convert to absolute values
abs_tenure_zscores = np.abs(tenure_zcores)
```

In [72]:
```
# Extract columns of intrest
churn_tenure = data_main_cc_merged[['churn','num_years_antig']]
```

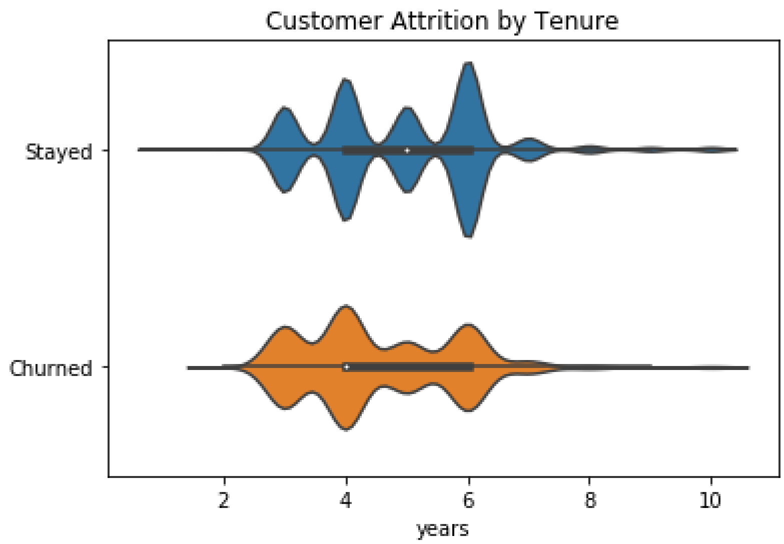
In [73]:
```
# Add z-score column
churn_tenure['z_score'] = list(abs_tenure_zscores)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\ipykernel_launcher.py:2: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [74]:
```
# Remove outliers
churned_tenure_filtered = churn_tenure[churn_tenure['z_score'] < 3]
```

In [75]:
```
# Visualize tenure by retained customer and churner
vio = sns.violinplot( y=churned_tenure_filtered["churn"], x=churned_tenur
e_filtered["num_years_antig"] )
vio.set(xlabel = 'years', ylabel='')
vio.set_title('Customer Attrition by Tenure')
plt.show()
```



- Customer are more likely to churn during the 4th year that the 7th year
- The median age of retained customers is 5years
- The median age of churners is 4 years

The Main Dataset

In [76]:
```
# Most popular electrivcity cmpaign
elec_nm = data_main_cc_merged.loc[(data_main_cc_merged['churn'] >= 'Staye
d')& (data_main_cc_merged['net_margin'] > 0), ['id','origin_up', 'net_mar
gin']]
```

In [77]:
```
elec_nm.value_counts(subset = ['origin_up'])
```

Out[77]:
```
origin_up
lxidpiddsbxsbosboudacockeimpuepw    6584
kamkkxfxxuwbdslkwifmmcsiusiuosws    4188
ldkssxwpmemidmecebumciepifcamkci    3201
usapbepcfoloekilkwsdiboslwaxobdp       2
ewxeelcelemmiwuafmddpobolfuxioce       1
dtype: int64
```

In [78]:
```
# Highest netting electricity subscription campaign
print(elec_nm.groupby('origin_up')['net_margin'].agg('sum').sort_values(a
scending=False))
```

```
origin_up
lxidpiddsbxsbosboudacockeimpuepw    1541159.95
ldkssxwpmemidmecebumciepifcamkci     814230.02
kamkkxfxxuwbdslkwifmmcsiusiuosws     717939.95
usapbepcfoloekilkwsdiboslwaxobdp        250.40
ewxeelcelemmiwuafmddpobolfuxioce         46.22
Name: net_margin, dtype: float64
```

In [79]:
```
# Select current customers with positive net margins
top_customers = data_main_cc_merged.loc[(data_main_cc_merged['churn']>='S
tayed') & (data_main_cc_merged['net_margin']>0),['id','num_years_antig',
'net_margin']]

# Top 10 customers by net margin
top_customers.sort_values(by=['net_margin'],ascending=False).head(10)
```

Out[79]:

	id	num_years_antig	net_margin
11502	d00e8a9951b5551d8f02e45f9ed2b0dd	3	10203.50
6930	78bd1c5c0c67f2be6de89b19df5f8861	3	5625.14
13259	818b8bca0a9d7668252d46b978169325	4	4346.37
8378	a3a739686fbd5ba8b4a21ec835507b6d	4	4305.79
324	89b3406c3ba717f1b788ceeb5af9e8b9	3	4161.74
10100	93435ecb05910c7b87e0ae9dbedb2882	4	4148.99
12028	4519e6a8928a015819466fc9de0fa49e	3	4040.60
6405	933527d7a2f669af49075a2380c10ded	4	3744.72
6850	43580ef6cc40fcfd0a9b76eee17a267a	4	3716.78
13553	ee98a86efa759681cc59c7d4e0d0312f	4	3407.65

These are the most profitable customers for PowerCo in terms of net margin. Beware most of them are within the likely tenure of attrition. Time for a marketing campaign!

In [ ]: