Exploratory Data Analysis for clients

In this assignment, our objective is to perform a comprehensive analysis of customer data to derive meaningful insights and accurately predict customer churn for an energy company. By employing advanced data analytics techniques, we aim to identify key indicators and patterns that contribute to customer attrition.

In [1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Shows plots in jupyter notebook
%matplotlib inline

# Set plot style
sns.set(color_codes=True)
```

Loading data with Pandas

We need to load `client_data.csv` and `price_data.csv` into individual dataframes so that we can work with them in Python. For this notebook and all further notebooks, it will be assumed that the CSV files will the placed in the same file location as the notebook. If they are not, please adjust the directory within the `read_csv` method accordingly.

In [2]:
```
client_df = pd.read_csv('C:\\Users\\sujoydutta\\Downloads\\client_data.csv')
price_df = pd.read_csv('C:\\Users\\sujoydutta\\Downloads\\price_data.csv')
```

You can view the first 3 rows of a dataframe using the `head` method. Similarly, if you wanted to see the last 3, you can use `tail(3)`

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

Out[3]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0

3 rows × 26 columns

In [4]:
```
price_df.head(3)
```

Out[4]:

	id	price_date	price_off_peak_var	price_peak_var	price_mid_peak_var	price_off
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	4
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	4
2	038af19179925da21a25619c5a24b745	2015-03-	0.151367	0.0	0.0	4

Descriptive statistics of data

Data types

It is useful to first understand the data that you're dealing with along with the data types of each column. The data types may dictate how you transform and engineer features.

To get an overview of the data types within a data frame, use the `info()` method.

In [5]:
```
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  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                       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
```

In [6]:
```
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
```

Statistics

Now let's look at some statistics about the datasets. We can do this by using the `describe()` method.

In [9]:
```
client_df.describe().round()
```

Out[9]:

	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_12m	forecast_cons_year	forecast_discount_energy
count	14606.0	14606.0	14606.0	14606.0	14606.0	14606.0
mean	159220.0	28092.0	16090.0	1869.0	1400.0	1.0
std	573465.0	162973.0	64364.0	2388.0	3248.0	5.0
min	0.0	0.0	0.0	0.0	0.0	0.0
25%	5675.0	0.0	0.0	495.0	0.0	0.0
50%	14116.0	0.0	792.0	1113.0	314.0	0.0
75%	40764.0	0.0	3383.0	2402.0	1746.0	0.0
max	6207104.0	4154590.0	771203.0	82903.0	175375.0	30.0

In [18]:
```
price_df.describe().round(4)
```

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	193002.0000	193002.0000	193002.0000	193002.0000	193002.0000	193002.0000
mean	0.1410	0.0546	0.0305	43.3345	10.6229	6.4100
std	0.0250	0.0499	0.0363	5.4103	12.8419	7.7736
min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25%	0.1260	0.0000	0.0000	40.7289	0.0000	0.0000
50%	0.1460	0.0855	0.0000	44.2669	0.0000	0.0000
75%	0.1516	0.1017	0.0726	44.4447	24.3396	16.2264
max	0.2807	0.2298	0.1141	59.4447	36.4907	17.4582

Data visualization

If you're working in Python, two of the most popular packages for visualization are `matplotlib` and `seaborn`. We highly recommend you use these, or at least be familiar with them because they are ubiquitous!

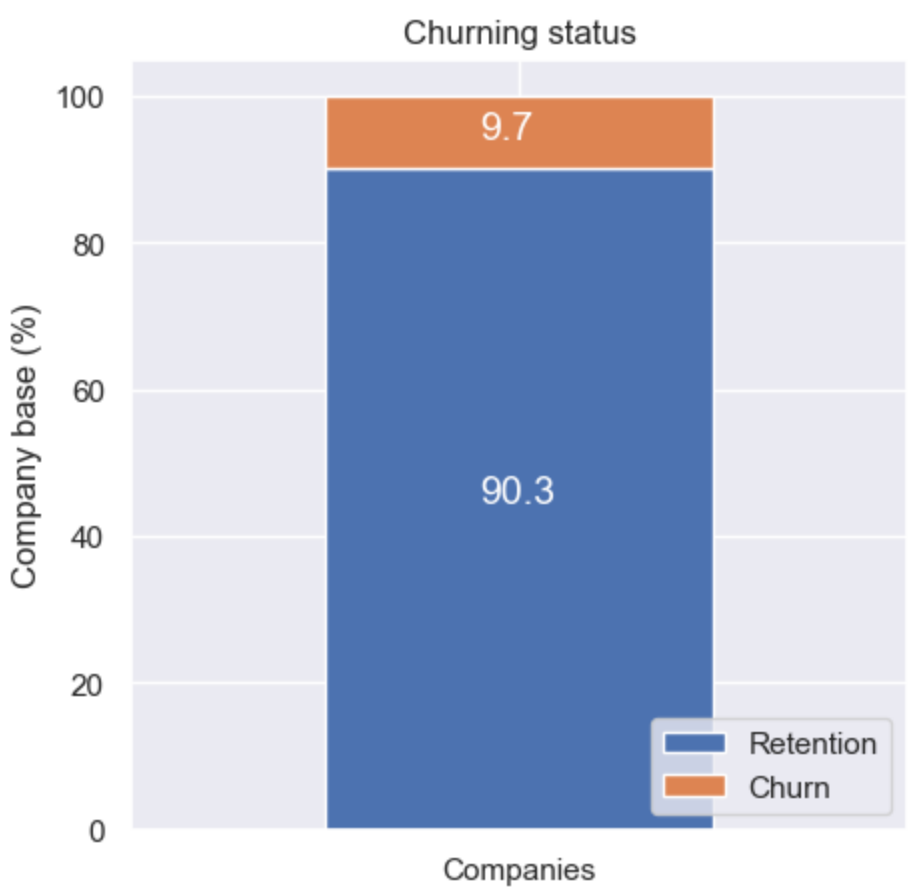
Below are some functions that you can use to get started with visualizations.

In [19]:
```
def plot_stacked_bars(dataframe, title_, size_=(18, 10), rot_=0, legend_="upper right"):
    """
    Plot stacked bars with annotations
```

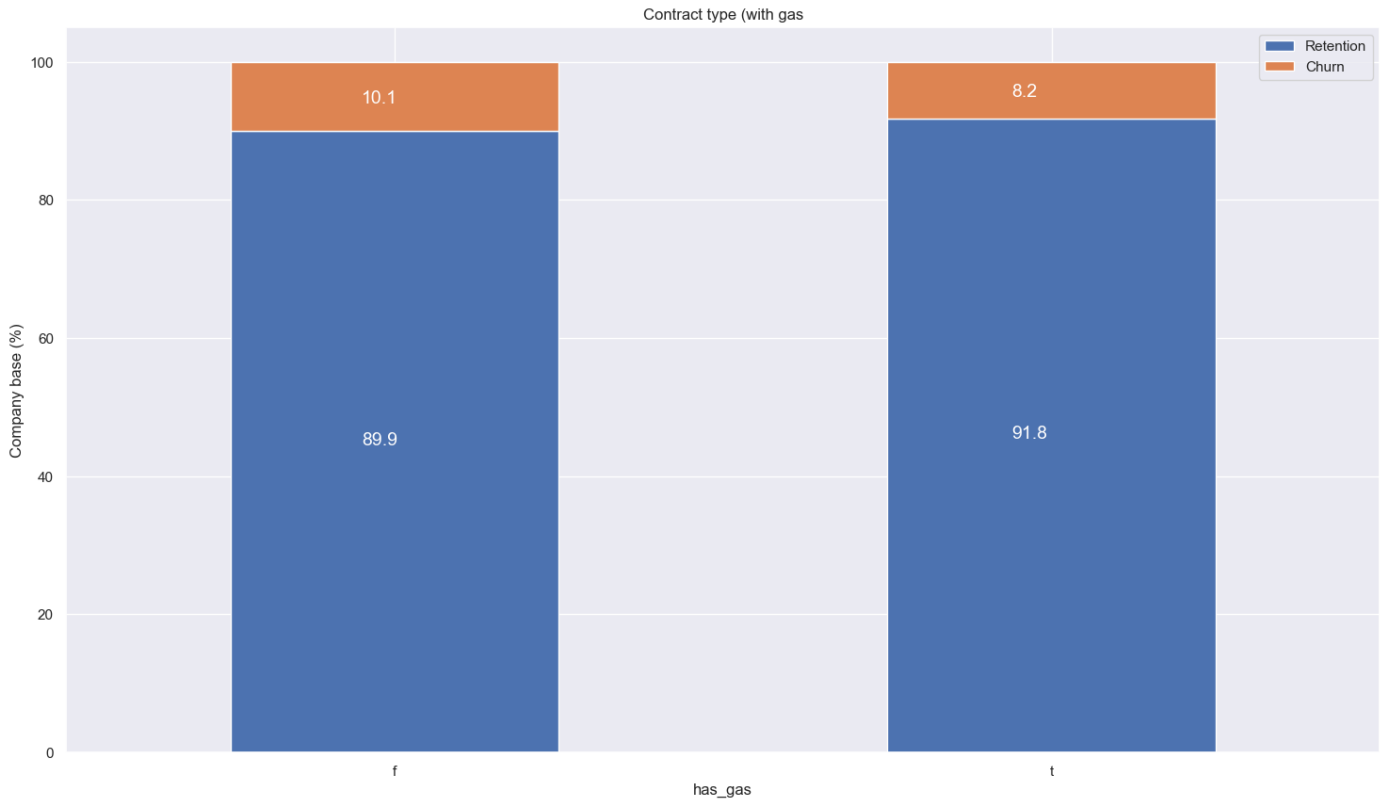
```
    """
    ax = dataframe.plot(
        kind="bar",
        stacked=True,
        figsize=size_,
        rot=rot_,
        title=title_
    )

    # 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
        )

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 [20]:

```
#Churn rate for the company
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), legend_="lowe
```



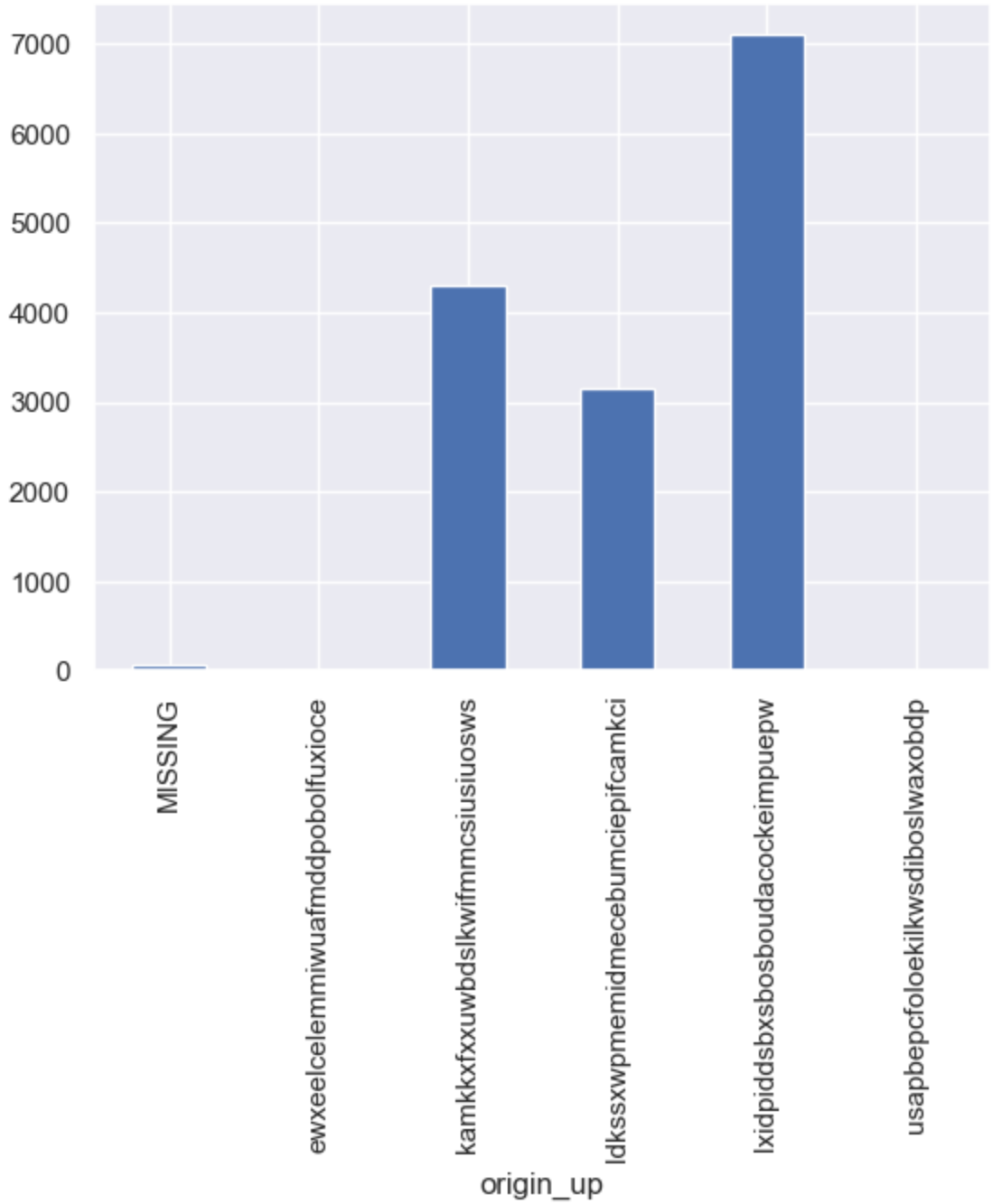
In [24]:
```
#Which contract type has high churn rate
contract_type = client_df[['id', 'has_gas', 'churn']]
contract = contract_type.groupby([contract_type['churn'], contract_type['has_gas']])['id
contract_percentage = (contract.div(contract.sum(axis=1), axis=0) * 100).sort_values(by=

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



In [26]:
```
#which campaign got the most customers?
campaigncount= client_df.groupby('origin_up')['id'].nunique()
campaigncount.plot(kind='bar')
```

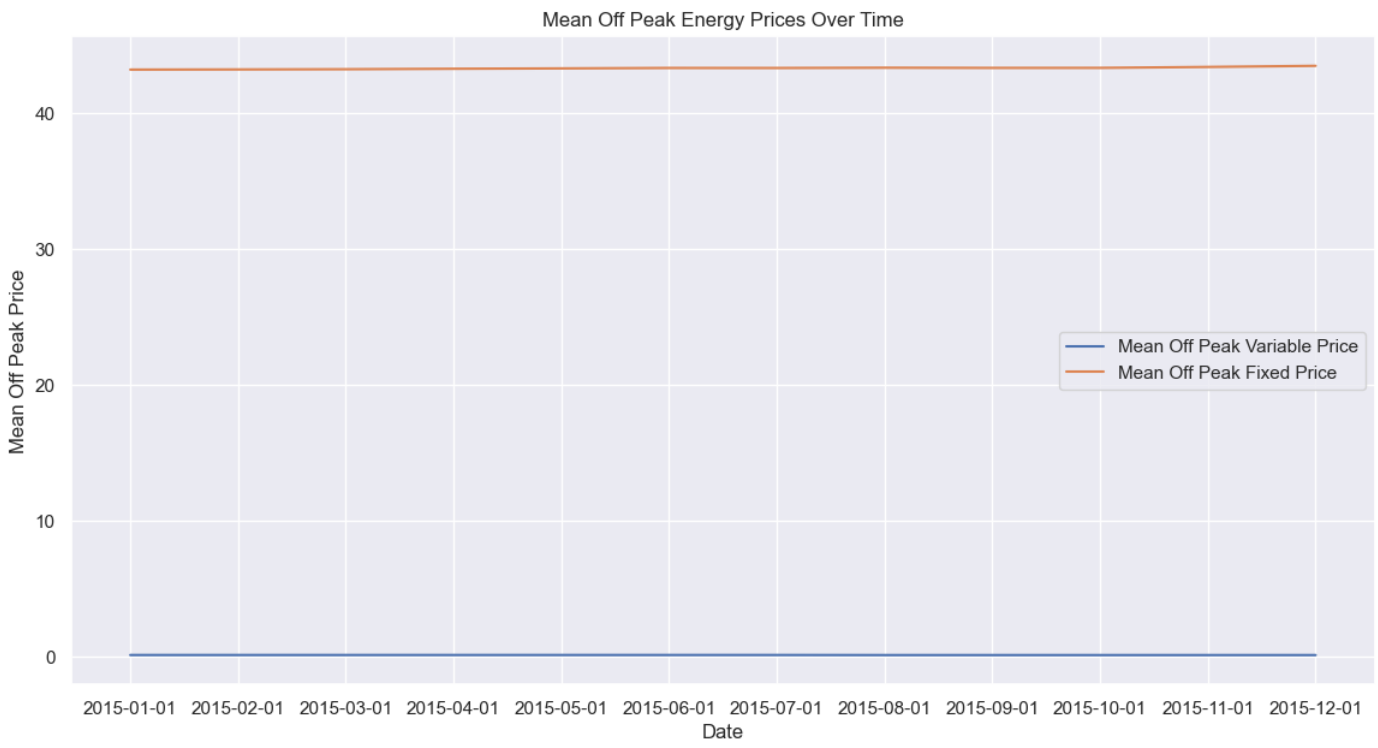
Out[26]:

<Axes: xlabel='origin_up'>

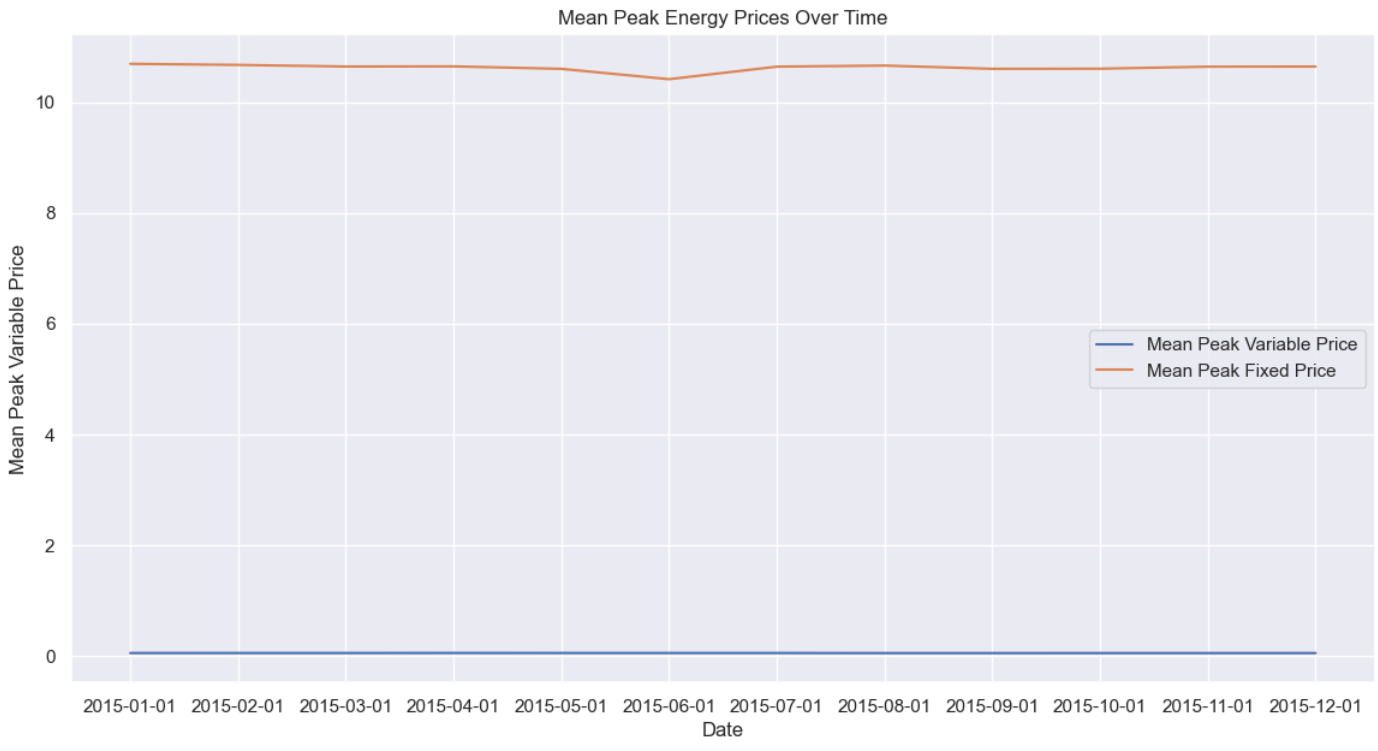


In [39]:

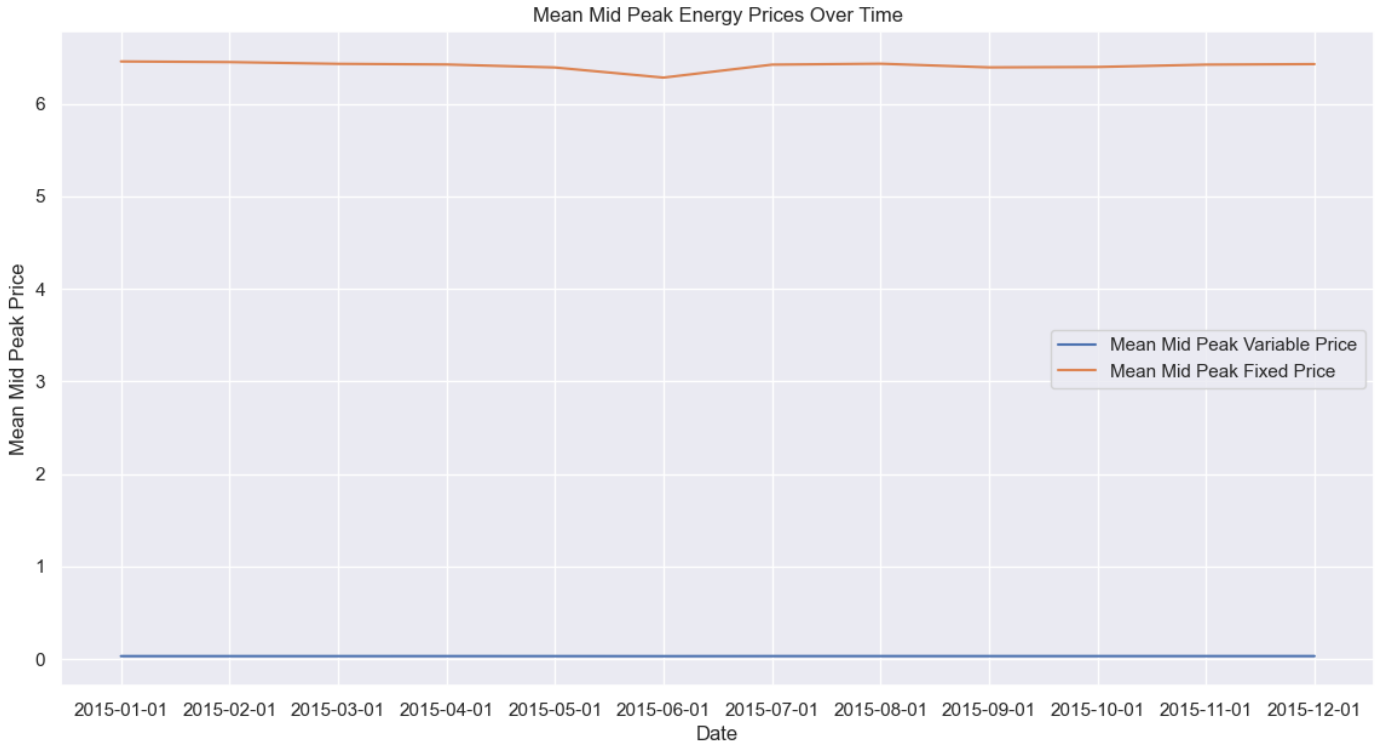
```
# Plotting the Off peak energy prices with respect to time


price_df_offpeakfixed = price_df.groupby('price_date')['price_off_peak_fix'].mean().rese
price_df_offpeakvar = price_df.groupby('price_date')['price_off_peak_var'].mean().reset_

plt.figure(figsize=(14, 7))
plt.plot(price_df_offpeakvar['price_date'], price_df_offpeakvar['price_off_peak_var'], l
plt.plot(price_df_offpeakfixed['price_date'], price_df_offpeakfixed['price_off_peak_fix'
plt.title('Mean Off Peak Energy Prices Over Time')
plt.xlabel('Date')
plt.ylabel('Mean Off Peak Price')
plt.legend()
plt.grid(True)
plt.show()
```



In [38]:
```
# Plotting the  peak energy prices with respect to time


price_df_peakfixed = price_df.groupby('price_date')['price_peak_fix'].mean().reset_index
price_df_peakvar = price_df.groupby('price_date')['price_peak_var'].mean().reset_index()

plt.figure(figsize=(14, 7))
plt.plot(price_df_peakvar['price_date'], price_df_peakvar['price_peak_var'], label='Mean
plt.plot(price_df_peakfixed['price_date'], price_df_peakfixed['price_peak_fix'], label='
plt.title('Mean Peak Energy Prices Over Time')
plt.xlabel('Date')
plt.ylabel('Mean Peak Price')
plt.legend()
plt.grid(True)
plt.show()
```



In [41]:
```
# Plotting the Mid peak energy prices with respect to time


price_df_midpeakfixed = price_df.groupby('price_date')['price_mid_peak_fix'].mean().rese
price_df_midpeakvar = price_df.groupby('price_date')['price_mid_peak_var'].mean().reset_

plt.figure(figsize=(14, 7))
plt.plot(price_df_midpeakvar['price_date'], price_df_midpeakvar['price_mid_peak_var'], l
plt.plot(price_df_midpeakfixed['price_date'], price_df_midpeakfixed['price_mid_peak_fix'
plt.title('Mean Mid Peak Energy Prices Over Time')
plt.xlabel('Date')
plt.ylabel('Mean Mid Peak Price')
plt.legend()
plt.grid(True)
plt.show()
```



In [46]:
```
client_df.head(5)
```

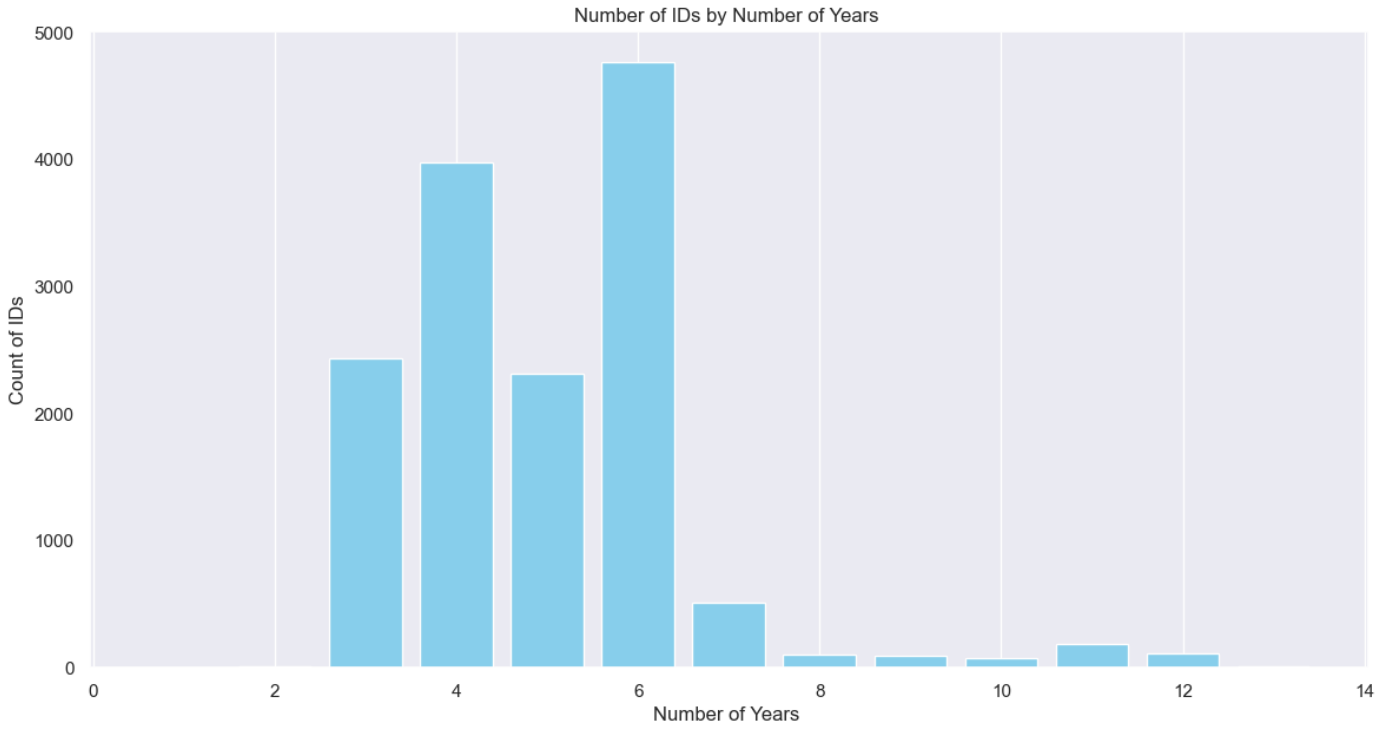
Out[46]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_mon
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

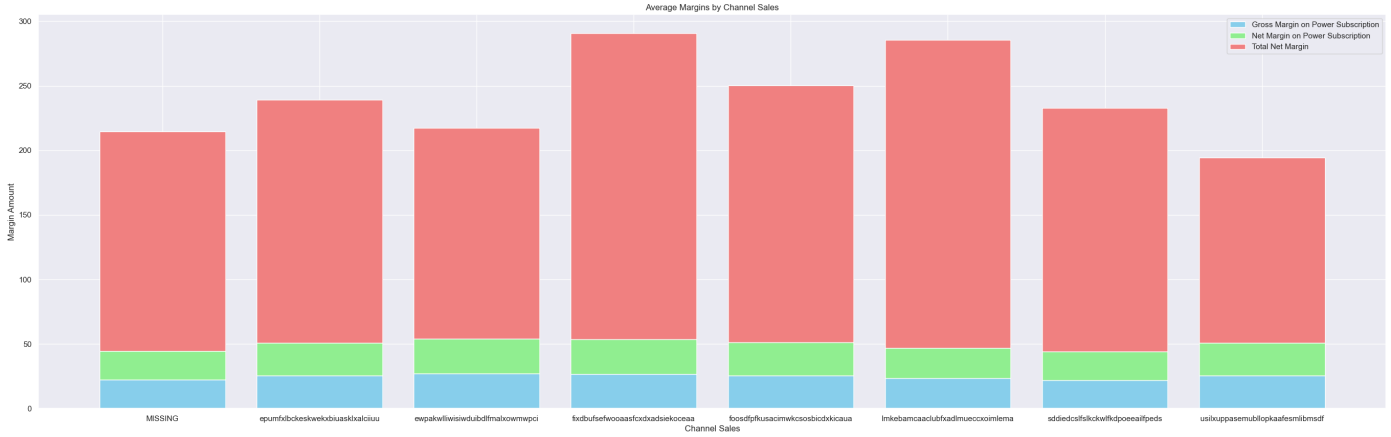
5 rows × 26 columns

In [45]:
```
# Seeing number of customers by tenure
df_grouped = client_df.groupby('num_years')['id'].count().reset_index()
df_grouped.columns = ['num_years', 'count_of_ids']


plt.figure(figsize=(14, 7))
plt.bar(df_grouped['num_years'], df_grouped['count_of_ids'], color='skyblue')
plt.title('Number of IDs by Number of Years ')
plt.xlabel('Number of Years ')
```

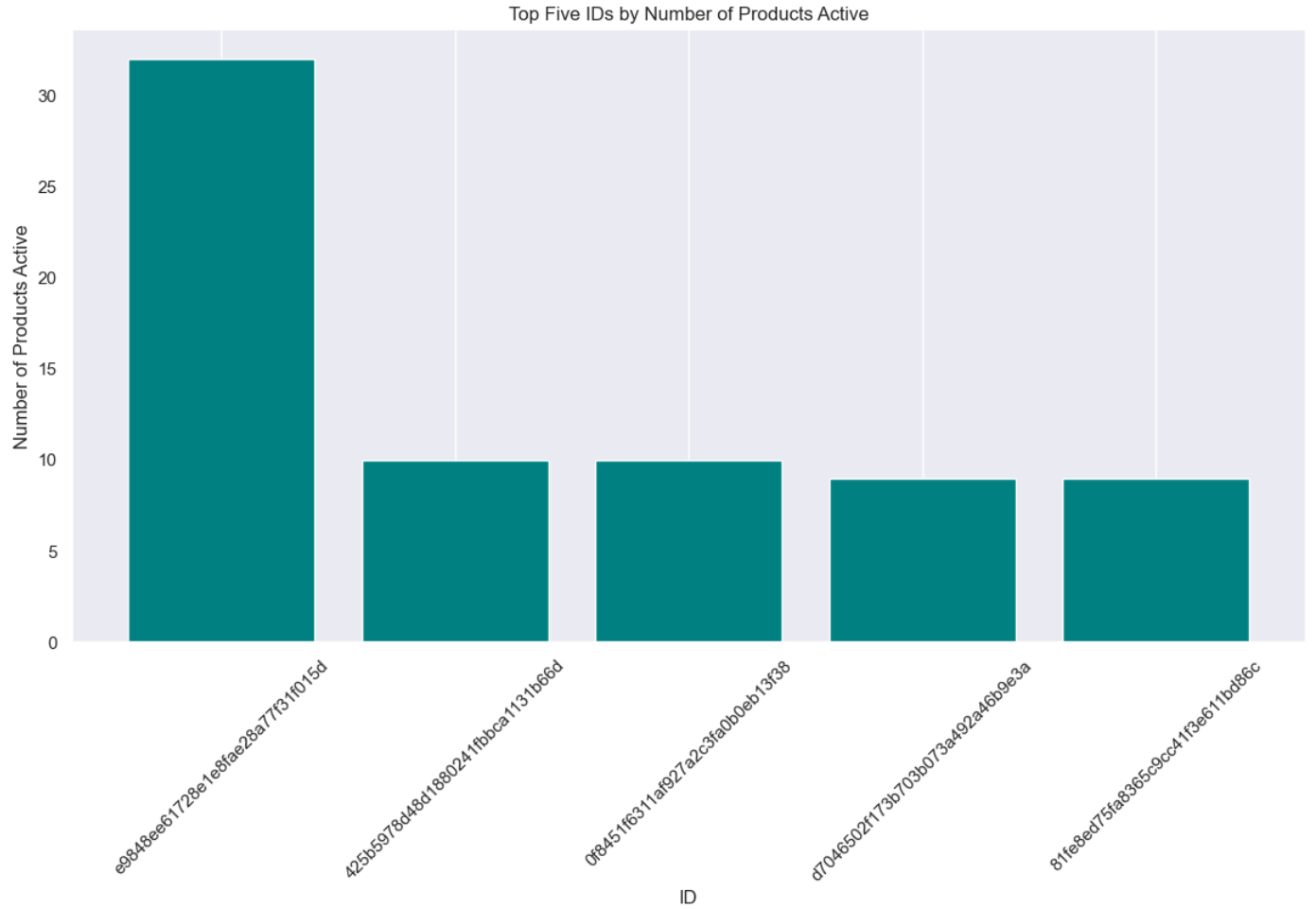
```
plt.ylabel('Count of IDs')
plt.grid(axis='y')
plt.show()
```



In [50]:
```
# Seeing the margin by channel sales
df_grouped = client_df.groupby('channel_sales')[['margin_gross_pow_ele', 'margin_net_pow


plt.figure(figsize=(34, 10))
bar_width = 0.35

plt.bar(df_grouped['channel_sales'], df_grouped['margin_gross_pow_ele'], label='Gross Ma


plt.bar(df_grouped['channel_sales'], df_grouped['margin_net_pow_ele'], bottom=df_grouped


plt.bar(df_grouped['channel_sales'], df_grouped['net_margin'], bottom=df_grouped['margin


plt.title('Average Margins by Channel Sales')
plt.xlabel('Channel Sales')
plt.ylabel('Margin Amount')
plt.legend()
plt.grid(True, axis='y')


plt.show()
```



In [61]:
```
# Plotting the top Five IDs by number of active services
top_5_df = client_df.sort_values(by='nb_prod_act', ascending=False).head(5)


plt.figure(figsize=(14, 7))
plt.bar(top_5_df['id'].astype(str), top_5_df['nb_prod_act'], color='teal')
plt.title('Top Five IDs by Number of Products Active')
plt.xlabel('ID')
plt.ylabel('Number of Products Active')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



In [58]:
```
# Sorting the top 5 consumers by electricity and gas consumption
top_elec_df = client_df.sort_values(by='cons_12m', ascending=False).head(5)
top_gas_df = client_df.sort_values(by='cons_gas_12m', ascending=False).head(5)


fig, ax = plt.subplots(2, 1, figsize=(14, 14))
```

```
ax[0].bar(top_elec_df['id'].astype(str), top_elec_df['cons_12m'], color='green')
ax[0].set_title('Top Five IDs by Electricity Consumption')
ax[0].set_xlabel('ID')
ax[0].set_ylabel('Electricity Consumption (12 months)')
ax[0].grid(axis='y')


ax[1].bar(top_gas_df['id'].astype(str), top_gas_df['cons_gas_12m'], color='maroon')
ax[1].set_title('Top Five IDs by Gas Consumption')
ax[1].set_xlabel('ID')
ax[1].set_ylabel('Gas Consumption (12 months)')
ax[1].grid(axis='y')

plt.tight_layout()
plt.show()
```