Modelo de Machine Learning - Previsão de Vendas de Cancelamento de Operadora TV a Cabo

Um breve resumo do que será feito:

1 - Analise Exploratória das variáveis Categóricas e Numéricas utilizando diversos gráficos

2 - Tratamento de Dados e Engenharia de Atributos

3 - Identificar e Tratar OUTLIERS e Valores Missing

3 - Muitas funções do Pacote Pandas

4 - OneHotEncoding e Padronização de Dados

4 - Criar, Treinar e Avaliar o Algoritmo de Machine Learning

E muito mais...

In [1]:
```
# Importando os pacotes que serão utilizados

# Para manipulação e tratamento dos dados
import numpy as np
import pandas as pd
import time #utilizada para funções de tempo
import matplotlib.pyplot as plt #Utilizada para gráficos
import seaborn as sns #Utilizada para gráficos

# Bibliotecas do Skit Learn
from sklearn.model_selection import train_test_split #Utilizada para separar dados
from sklearn.preprocessing import StandardScaler #Utilizada para fazer a padronizaç
from sklearn.preprocessing import LabelEncoder #Utilizada para fazer o OneHotEncodi
from sklearn.metrics import accuracy_score #Utilizada para avaliar a acurácia do mo
from sklearn.neighbors import KNeighborsClassifier #Nosso Algoritmo para criação do
from imblearn import under_sampling, over_sampling #Utilizada para fazer o balanced
from imblearn.over_sampling import SMOTE #Utilizada para fazer o balanceamento de d


# Para remover avisos de alerta
import warnings #Utilizada para avisos de alertas
warnings.filterwarnings("ignore") #Ignorar avisos de alertas. Obs.: Alertas NÃO são

# Para não limitar a exibição do DataFrame
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
#pd.options.display.float_format = '{:.2f}'.format
```

Coleta de Dados

Utilizaremos o arquivo dados.csv para importar os nosso dataset

In [2]:
```
#df_original = pd.read_csv("dados.csv", sep = ';', encoding="ISO-8859-1")
df_original = pd.read_csv("dados.csv", sep = ';')
```

Analise Exploratória

Vamos analisar as variáveis categoricas e númericas e identificar os dados que precisarão ser tratados

In [3]:
```
# Visualizar as Linhas e Colunas do Arquivo --> Observações e Variáveis
df_original.shape
```

Out[3]:
```
(448447, 24)
```

In [4]:
```
# Visualizar as primeiras linha do arquivo
df_original.head()
```

Out[4]:

	ID_CLIENTE	FORMA_AQUISICAO	IDADE_CLIENTE	SEXO	QT_FILHOS	DT_AQUISICAO	DT_
0	1	Site	23	MASCULINO	0.0	18/06/2021	
1	2	Vendedor	24	FEMININO	0.0	10/04/2018	
2	3	Site	25	MASCULINO	0.0	09/10/2020	
3	4	Vendedor	26	FEMININO	17.0	25/06/2019	
4	5	Vendedor	27	MASCULINO	0.0	19/09/2019	

In [5]:
```
# Verificar os tipos de dados das variáveis
df_original.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 448447 entries, 0 to 448446
Data columns (total 24 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   ID_CLIENTE                    448447 non-null  int64  
 1   FORMA_AQUISICAO               448447 non-null  object 
 2   IDADE_CLIENTE                 448447 non-null  int64  
 3   SEXO                          448447 non-null  object 
 4   QT_FILHOS                     448193 non-null  float64
 5   DT_AQUISICAO                  448447 non-null  object 
 6   DT_CANCELAMENTO               117455 non-null  object 
 7   DIAS_ATIVO                    448447 non-null  int64  
 8   MESES_ATIVO                   448447 non-null  int64  
 9   DURACAO_CONTRATO              448447 non-null  object 
 10  VL_PLANO_ADESAO               448447 non-null  int64  
 11  VL_PLANO_ATUAL                448447 non-null  int64  
 12  NOME_PRODUTO                  448447 non-null  object 
 13  QT_PONTOS_INSTALADOS          448447 non-null  int64  
 14  QT_PC_PAGAS                   448447 non-null  int64  
 15  QT_PC_VENCIDAS                448447 non-null  int64  
 16  QT_PC_PAGA_ATRASO             448447 non-null  int64  
 17  QT_PC_PAGA_EM_DIA             448447 non-null  int64  
 18  QT_ACORDO_PAGAMENTO           448447 non-null  int64  
 19  VL_MENSALIDADE_ATRASO         448447 non-null  int64  
 20  VL_MENSALIDADE_DT_AQUISICAO   448447 non-null  int64  
 21  VL_MENSALIDADE_DT_ATUAL       448447 non-null  int64  
 22  SITUACAO                      448447 non-null  object 
 23  COD_SITUACAO                  448447 non-null  int64  
dtypes: float64(1), int64(16), object(7)
memory usage: 82.1+ MB
```

In [6]:
```
#Avaliar o período dos dados coletados
inicio = pd.to_datetime(df_original['DT_AQUISICAO']).dt.date.min()
fim = pd.to_datetime(df_original['DT_AQUISICAO']).dt.date.max()
print('Período dos dados - De:', inicio, 'Até:',fim)
```

```
Período dos dados - De: 2001-01-01 Até: 2021-12-06
```

In [7]:
```
# Resumo estatistico básico do nosso dataset
df_original.describe()
```

Out[7]:

	ID_CLIENTE	IDADE_CLIENTE	QT_FILHOS	DIAS_ATIVO	MESES_ATIVO	VL_PLANO_AD
count	448447.000000	448447.000000	448193.000000	448447.000000	448447.000000	448447.0
mean	224224.000000	38.891140	1.526385	483.857783	15.772457	303.7
std	129455.642421	6.682351	0.504288	373.649523	12.252344	113.6
min	1.000000	23.000000	0.000000	22.000000	1.000000	230.0
25%	112112.500000	35.000000	1.000000	167.000000	5.000000	230.0
50%	224224.000000	40.000000	2.000000	329.000000	11.000000	230.0
75%	336335.500000	43.000000	2.000000	798.000000	26.000000	350.0
max	448447.000000	55.000000	25.000000	1296.000000	42.000000	600.0

In [8]:
```
# Verificando valores missing
# Variável QT FILHOS deveremos tratar na etapa 2 - Tratamento de Dados
print(df_original.isna().sum())
```

```
ID_CLIENTE                         0
FORMA_AQUISICAO                    0
IDADE_CLIENTE                      0
SEXO                               0
QT_FILHOS                        254
DT_AQUISICAO                       0
DT_CANCELAMENTO               330992
DIAS_ATIVO                         0
MESES_ATIVO                        0
DURACAO_CONTRATO                   0
VL_PLANO_ADESAO                    0
VL_PLANO_ATUAL                     0
NOME_PRODUTO                       0
QT_PONTOS_INSTALADOS               0
QT_PC_PAGAS                        0
QT_PC_VENCIDAS                     0
QT_PC_PAGA_ATRASO                  0
QT_PC_PAGA_EM_DIA                  0
QT_ACORDO_PAGAMENTO                0
VL_MENSALIDADE_ATRASO              0
VL_MENSALIDADE_DT_AQUISICAO        0
VL_MENSALIDADE_DT_ATUAL            0
SITUACAO                           0
COD_SITUACAO                       0
dtype: int64
```

In [9]:
```
# Verificando Valores Únicos
df_original.nunique()
```

Out[9]:
```
ID_CLIENTE                    448447
FORMA_AQUISICAO                    2
IDADE_CLIENTE                     33
SEXO                               2
QT_FILHOS                          7
DT_AQUISICAO                    5888
DT_CANCELAMENTO                 5304
DIAS_ATIVO                      1051
MESES_ATIVO                       42
DURACAO_CONTRATO                   4
VL_PLANO_ADESAO                    6
VL_PLANO_ATUAL                     6
NOME_PRODUTO                       6
QT_PONTOS_INSTALADOS               3
QT_PC_PAGAS                       31
QT_PC_VENCIDAS                    31
QT_PC_PAGA_ATRASO                  9
QT_PC_PAGA_EM_DIA                 30
QT_ACORDO_PAGAMENTO                6
VL_MENSALIDADE_ATRASO             80
VL_MENSALIDADE_DT_AQUISICAO        6
VL_MENSALIDADE_DT_ATUAL            6
SITUACAO                           2
COD_SITUACAO                       2
dtype: int64
```

Analisando as Variáveis Categoricas (FORMA_AQUISICAO, SEXO, DURACAO_CONTRATO, NOME_PRODUTO, SITUACAO)

In [10]:
```
# Agrupar os valores da variável
df_original.groupby(['FORMA_AQUISICAO']).size()
```

Out[10]:
```
FORMA_AQUISICAO
Site        321376
Vendedor    127071
dtype: int64
```

In [11]:
```
# Agrupar os valores da variável
df_original.groupby(['SEXO']).size()
```

Out[11]:
```
SEXO
FEMININO     224223
MASCULINO    224224
dtype: int64
```

In [12]:
```
# Agrupar os valores da variável
df_original.groupby(['DURACAO_CONTRATO']).size()
```

Out[12]:
```
DURACAO_CONTRATO
12 Meses       195
24 Meses       235
36 Meses     31889
48 Meses    416128
dtype: int64
```

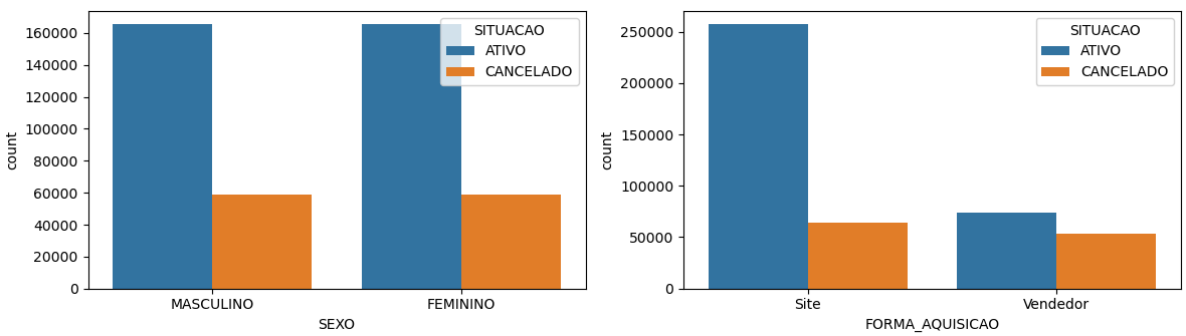
In [13]:
```
# Agrupar os valores da variável
df_original.groupby(['NOME_PRODUTO']).size()
```

Out[13]:
```
NOME_PRODUTO
PLANO BASICO (30 CANAIS HD)          285209
PLANO BASICO PLUS (50 CANAIS HD)       8835
PLANO FAMILIA (100 CANAIS HD)         59716
PLANO MEDIO A (60 CANAIS HD)          62221
PLANO MEDIO TOP (90 CANAIS HD)          295
PLANO PREMIUM TOTAL                   32171
dtype: int64
```

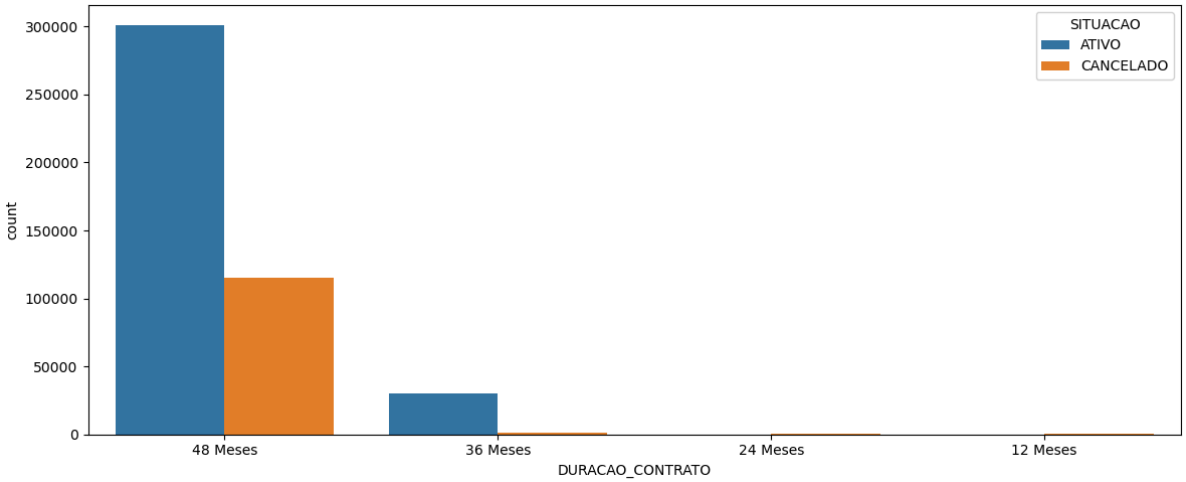
In [14]:
```
# Agrupar os valores da variável
df_original.groupby(['SITUACAO']).size()
```

Out[14]:
```
SITUACAO
ATIVO        330992
CANCELADO    117455
dtype: int64
```

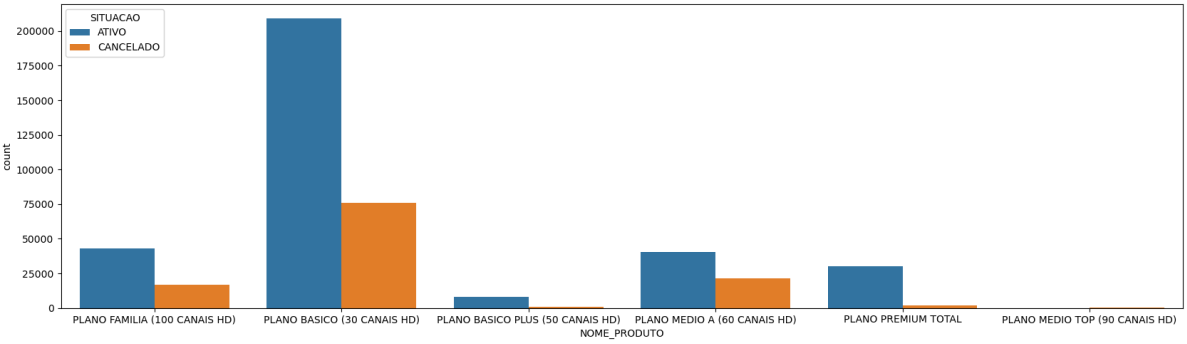
In [15]:
```
# Analisando o gráfico da variavel FORMA_AQUISICAO e SEXO comparadas a variável ALV
#Podemos constatar na analise que não há discrepancias nestas variaveis

plt.rcParams["figure.figsize"] = [12.00, 3.50]
plt.rcParams["figure.autolayout"] = True
f, axes = plt.subplots(1, 2)
sns.countplot(data = df_original, x = "SEXO", hue = "SITUACAO", ax=axes[0])
sns.countplot(data = df_original, x = "FORMA_AQUISICAO", hue = "SITUACAO", ax=axes[
plt.show()
```



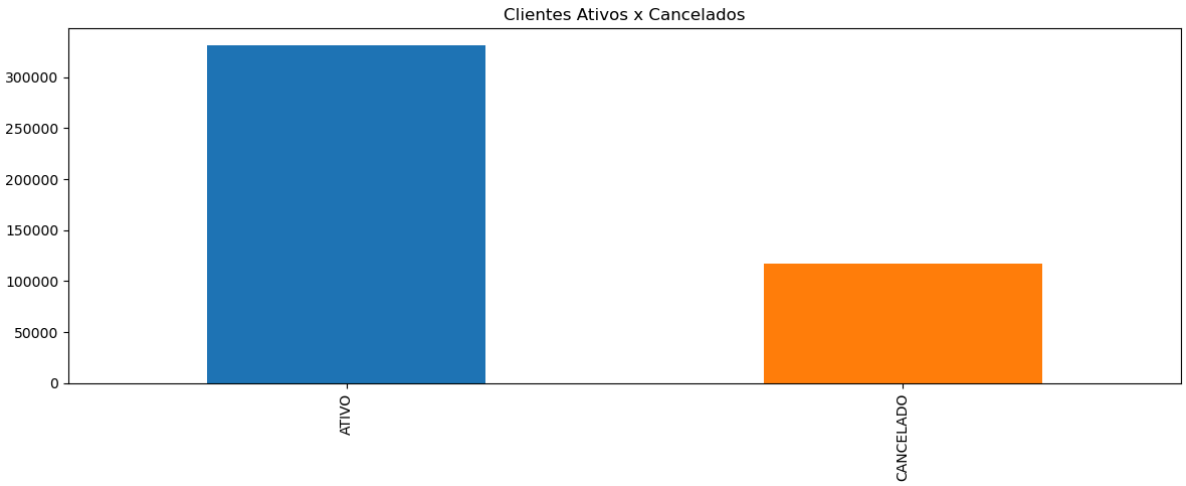
In [16]:
```
# Analisando o gráfico da variavel DURACAO_CONTRATO comparadas a variável ALVO
#Podemos constatar na analise que não há discrepancias nestas variaveis
plt.rcParams["figure.figsize"] = [12.00, 5.00]
plt.rcParams["figure.autolayout"] = True
sns.countplot(data = df_original, x = "DURACAO_CONTRATO", hue = "SITUACAO")
plt.show()
```



In [17]:
```
# Analisando o gráfico da variavel NOME_PRODUTO comparadas a variável ALVO
#Podemos constatar na analise que não há discrepancias nestas variaveis
plt.rcParams["figure.figsize"] = [17.00, 5.00]
plt.rcParams["figure.autolayout"] = True
sns.countplot(data = df_original, x = "NOME_PRODUTO", hue = "SITUACAO")
plt.show()
```



In [18]:
```
#Analisando como a variavel alvo está distribuida.
#Aqui podemos observar que há muito mais CLIENTES ATIVOS do que CLIENTES CANCELADOS
#dessa forma, precisaremos balancear o dataset na etapa 2 - Tratamento de Dados.
plt.rcParams["figure.figsize"] = [12.00, 5.00]
plt.rcParams["figure.autolayout"] = True
df_original.SITUACAO.value_counts().plot(kind='bar', title='Clientes Ativos x Cance
```



Analisando as Variaveis Numéricas

In [19]:
```
# Carregar variaveis para plot
# Pegaremos a partir da variavel 1 porque o ID_CLIENTE não iremos utilizar
variaveis_numericas = []
for i in df_original.columns[1:24].tolist():
        if df_original.dtypes[i] == 'int64' or df_original.dtypes[i] == 'float64':
            print(i, ':' , df_original.dtypes[i])
            variaveis_numericas.append(i)
```

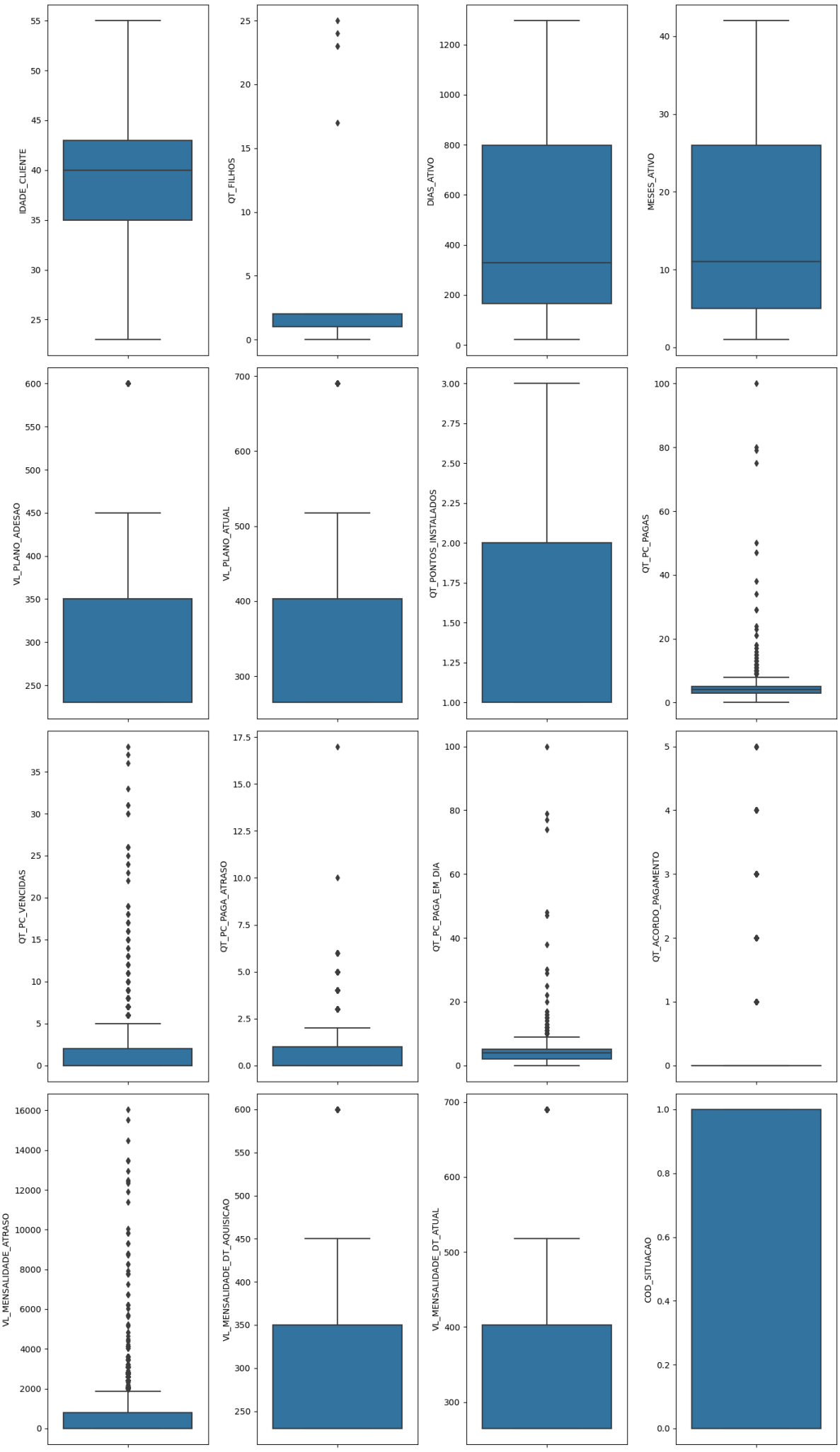
```
IDADE_CLIENTE : int64
QT_FILHOS : float64
DIAS_ATIVO : int64
MESES_ATIVO : int64
VL_PLANO_ADESAO : int64
VL_PLANO_ATUAL : int64
QT_PONTOS_INSTALADOS : int64
QT_PC_PAGAS : int64
QT_PC_VENCIDAS : int64
QT_PC_PAGA_ATRASO : int64
QT_PC_PAGA_EM_DIA : int64
QT_ACORDO_PAGAMENTO : int64
VL_MENSALIDADE_ATRASO : int64
VL_MENSALIDADE_DT_AQUISICAO : int64
VL_MENSALIDADE_DT_ATUAL : int64
COD_SITUACAO : int64
```

In [20]:
```
#Quantidade de variaveis
len(variaveis_numericas)
```

Out[20]:
```
16
```

In [21]:
```
#Podemos observar nos boxplots abaixo que algumas variáveis númericas apresentam um
#Precisamos avaliar cada uma dessas variaveis dentro do contexto dos dados para sab

plt.rcParams["figure.figsize"] = [14.00, 24.00]
plt.rcParams["figure.autolayout"] = True
f, axes = plt.subplots(4, 4) #4 linhas e 4 colunas

linha = 0
coluna = 0
for i in variaveis_numericas:
    sns.boxplot(data = df_original, y=i, ax=axes[linha][coluna])
    coluna += 1
    if coluna == 4:
        linha += 1
```

```
        coluna = 0

plt.show()
```


IDADE_CLIENTE
QT_FILHOS
DIAS_ATIVO
MESES_ATIVO
VL_PLANO_ADESAO
VL_PLANO_ATUAL
QT_PONTOS_INSTALADOS
QT_PC_PAGAS
QT_PC_VENCIDAS
QT_PC_PAGA_ATRASO
QT_PC_PAGA_EM_DIA
QT_ACORDO_PAGAMENTO
VL_MENSALIDADE_ATRASO
VL_MENSALIDADE_DT_AQUISICAO
VL_MENSALIDADE_DT_ATUAL
COD_SITUACAO

In [22]:
```
# A Variavel QT_PC_PAGA_EM_DIA e QT_PC_PAGAS possui um número maior que o prazo máx
# que foi algum erro dos dados gerando este OUTLIER e iremos trata-los considerando

# A Variavel QT_FILHOS também possui alguns OUTLIERS como por exemplo, 17, 23 e 24
# desses dados e verificar como iremos trata-los.
```

In [23]:
```
# Temos apenas 5 registros dentro de um volume de mais de 440 mil registros, dessa
# impacto de perda de dados, pois a quantidade é muito pequena comparada ao total d
df_original.groupby(['QT_FILHOS']).size()
```

Out[23]:
```
QT_FILHOS
0.0         10
1.0     212353
2.0     235825
17.0         1
23.0         2
24.0         1
25.0         1
dtype: int64
```

In [24]:
```
# Aqui podemos ver os registros...
df_original.loc[df_original['QT_FILHOS'] > 2]
```

Out[24]:

	ID_CLIENTE	FORMA_AQUISICAO	IDADE_CLIENTE	SEXO	QT_FILHOS	DT_AQUISICAO	D
3	4	Vendedor	26	FEMININO	17.0	25/06/2019	
91	92	Vendedor	48	FEMININO	23.0	03/08/2018	
164	165	Vendedor	55	MASCULINO	23.0	19/06/2018	
273	274	Vendedor	32	FEMININO	24.0	02/05/2018	
454	455	Vendedor	38	MASCULINO	25.0	04/09/2018	

Tratamento de Dados

Vamos tratar os dados que identificamos na fase de Analise Exploratória

1 - Tratar os OUTILIERS
2 - Tratar valores nulos (Variável QT_FILHOS possui 254 valores nulos)
3 - Tratar as variaveis QT_PC_PAGA_EM_DIA e QT_PC_PAGAS
4 - Engenharia de Atributos (Criar variavel NIVEL PAGAMENTO)
5 - Balancear variavel target
6 - Aplicar o OneHotEncoding

In [ ]:
```
```

In [25]:
```
# Manteremos o DataFrame Priginal e os dados tratados ficarão no DataFrame chamado
df_dados = df_original.loc[df_original['QT_FILHOS'] <= 2]
df_dados.shape
```

Out[25]:
```
(448188, 24)
```

In [26]:
```
df_dados.groupby(['QT_FILHOS']).size()
```

Out[26]:
```
QT_FILHOS
0.0        10
1.0    212353
2.0    235825
dtype: int64
```

In [27]:
```
print('Média de Filhos: ', df_dados['QT_FILHOS'].mean())
print('Mediana de Filhos: ',df_dados['QT_FILHOS'].median())
print('Moda: ', df_dados['QT_FILHOS'].mode())
```

```
Média de Filhos:  1.5261519719403465
Mediana de Filhos:  2.0
Moda:  0    2.0
Name: QT_FILHOS, dtype: float64
```

In [28]:
```
# Preencheremos os valores NULOS com a mediana dos dados
df_dados['QT_FILHOS'] = df_dados['QT_FILHOS'].fillna((df_dados['QT_FILHOS'].median(
df_dados.isnull().sum()
```

Out[28]:
```
ID_CLIENTE                         0
FORMA_AQUISICAO                    0
IDADE_CLIENTE                      0
SEXO                               0
QT_FILHOS                          0
DT_AQUISICAO                       0
DT_CANCELAMENTO               330987
DIAS_ATIVO                         0
MESES_ATIVO                        0
DURACAO_CONTRATO                   0
VL_PLANO_ADESAO                    0
VL_PLANO_ATUAL                     0
NOME_PRODUTO                       0
QT_PONTOS_INSTALADOS               0
QT_PC_PAGAS                        0
QT_PC_VENCIDAS                     0
QT_PC_PAGA_ATRASO                  0
QT_PC_PAGA_EM_DIA                  0
QT_ACORDO_PAGAMENTO                0
VL_MENSALIDADE_ATRASO              0
VL_MENSALIDADE_DT_AQUISICAO        0
VL_MENSALIDADE_DT_ATUAL            0
SITUACAO                           0
COD_SITUACAO                       0
dtype: int64
```

In [29]:
```
# Substituindo os dados da variavel DURACAO_CONTRATO para mantermos somente os núme
df_dados['DURACAO_CONTRATO'] = df_dados['DURACAO_CONTRATO'].replace(['12 Meses'], 1
df_dados['DURACAO_CONTRATO'] = df_dados['DURACAO_CONTRATO'].replace(['24 Meses'], 2
df_dados['DURACAO_CONTRATO'] = df_dados['DURACAO_CONTRATO'].replace(['36 Meses'], 3
df_dados['DURACAO_CONTRATO'] = df_dados['DURACAO_CONTRATO'].replace(['48 Meses'], 4
```

In [30]:
```
# Visualizando as primeiras linhas do dataset
df_dados.head()
```

Out[30]:

	ID_CLIENTE	FORMA_AQUISICAO	IDADE_CLIENTE	SEXO	QT_FILHOS	DT_AQUISICAO	DT_
0	1	Site	23	MASCULINO	0.0	18/06/2021	
1	2	Vendedor	24	FEMININO	0.0	10/04/2018	
2	3	Site	25	MASCULINO	0.0	09/10/2020	
4	5	Vendedor	27	MASCULINO	0.0	19/09/2019	
5	6	Vendedor	28	FEMININO	1.0	23/03/2018	

In [31]:
```
# Visualizando as informações dos tipos de variaveis
df_dados.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 448188 entries, 0 to 448446
Data columns (total 24 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   ID_CLIENTE                     448188 non-null  int64
 1   FORMA_AQUISICAO                448188 non-null  object
 2   IDADE_CLIENTE                  448188 non-null  int64
 3   SEXO                           448188 non-null  object
 4   QT_FILHOS                      448188 non-null  float64
 5   DT_AQUISICAO                   448188 non-null  object
 6   DT_CANCELAMENTO                117201 non-null  object
 7   DIAS_ATIVO                     448188 non-null  int64
 8   MESES_ATIVO                    448188 non-null  int64
 9   DURACAO_CONTRATO               448188 non-null  int64
 10  VL_PLANO_ADESAO                448188 non-null  int64
 11  VL_PLANO_ATUAL                 448188 non-null  int64
 12  NOME_PRODUTO                   448188 non-null  object
 13  QT_PONTOS_INSTALADOS           448188 non-null  int64
 14  QT_PC_PAGAS                    448188 non-null  int64
 15  QT_PC_VENCIDAS                 448188 non-null  int64
 16  QT_PC_PAGA_ATRASO              448188 non-null  int64
 17  QT_PC_PAGA_EM_DIA              448188 non-null  int64
 18  QT_ACORDO_PAGAMENTO            448188 non-null  int64
 19  VL_MENSALIDADE_ATRASO          448188 non-null  int64
 20  VL_MENSALIDADE_DT_AQUISICAO    448188 non-null  int64
 21  VL_MENSALIDADE_DT_ATUAL        448188 non-null  int64
 22  SITUACAO                       448188 non-null  object
 23  COD_SITUACAO                   448188 non-null  int64
dtypes: float64(1), int64(17), object(6)
memory usage: 85.5+ MB
```

In [32]:
```
# Identificando as quantidades máximas para tratar os OUTLIERS
print(df_dados['QT_PC_PAGAS'].max())
print(df_dados['QT_PC_PAGA_EM_DIA'].max())
```

```
100
100
```

In [33]:
```
# Registros que possuirem as quantidades superiores a duração do contrato, iremos a
df_dados.loc[df_dados.QT_PC_PAGAS > df_dados.DURACAO_CONTRATO, 'QT_PC_PAGAS'] = df_
df_dados.loc[df_dados.QT_PC_PAGA_EM_DIA > df_dados.DURACAO_CONTRATO, 'QT_PC_PAGA_EM
```

In [34]:
```
# Verificando se as variáveis foram ajustadas
print(df_dados['QT_PC_PAGAS'].max())
print(df_dados['QT_PC_PAGA_EM_DIA'].max())
```

```
48
48
```

In [ ]:
```
```

In [35]:
```
# Engenharia de Atributos
# Criando uma nova variável de categoria de nivel de pagamento de acordo com a quan
bins = [-100, 3, 6, 12, 48]
labels = ['RUIM', 'MEDIO', 'BOM', 'OTIMO']
df_dados['NIVEL_PAGAMENTO'] = pd.cut(df_dados['QT_PC_PAGAS'], bins=bins, labels=lab
pd.value_counts(df_dados.NIVEL_PAGAMENTO)
```

Out[35]:
```
MEDIO    297750
RUIM     149912
BOM         488
OTIMO        38
Name: NIVEL_PAGAMENTO, dtype: int64
```

In [36]:
```
# Visualizando as primeiras linhas do dataset
df_dados.head()
```

Out[36]:

	ID_CLIENTE	FORMA_AQUISICAO	IDADE_CLIENTE	SEXO	QT_FILHOS	DT_AQUISICAO	DT_
0	1	Site	23	MASCULINO	0.0	18/06/2021	
1	2	Vendedor	24	FEMININO	0.0	10/04/2018	
2	3	Site	25	MASCULINO	0.0	09/10/2020	
4	5	Vendedor	27	MASCULINO	0.0	19/09/2019	
5	6	Vendedor	28	FEMININO	1.0	23/03/2018	

In [37]:
```
# Fazendo uma cópia do DataFrame
df_dados_2 = df_dados.copy()
df_dados_2.head()
```

Out[37]:

	ID_CLIENTE	FORMA_AQUISICAO	IDADE_CLIENTE	SEXO	QT_FILHOS	DT_AQUISICAO	DT_
0	1	Site	23	MASCULINO	0.0	18/06/2021	
1	2	Vendedor	24	FEMININO	0.0	10/04/2018	
2	3	Site	25	MASCULINO	0.0	09/10/2020	
4	5	Vendedor	27	MASCULINO	0.0	19/09/2019	
5	6	Vendedor	28	FEMININO	1.0	23/03/2018	

In [38]:
```
# Cria o encoder
lb = LabelEncoder()

# Aplica o encoder nas variáveis que estão com string
# O encoder irá transformar essas variáveis em números (Lembre-se, os algoritmos de
df_dados_2['SEXO'] = lb.fit_transform(df_dados_2['SEXO'])
df_dados_2['FORMA_AQUISICAO'] = lb.fit_transform(df_dados_2['FORMA_AQUISICAO'])
df_dados_2['NOME_PRODUTO'] = lb.fit_transform(df_dados_2['NOME_PRODUTO'])
df_dados_2['NIVEL_PAGAMENTO'] = lb.fit_transform(df_dados_2['NIVEL_PAGAMENTO'])
```

In [39]:
```
# Visualizando as primerias 20 linhas do Dataset
df_dados_2.head(20)
```

Out[39]:

	ID_CLIENTE	FORMA_AQUISICAO	IDADE_CLIENTE	SEXO	QT_FILHOS	DT_AQUISICAO	DT_CANC
0	1	0	23	1	0.0	18/06/2021	
1	2	1	24	0	0.0	10/04/2018	
2	3	0	25	1	0.0	09/10/2020	
4	5	1	27	1	0.0	19/09/2019	
5	6	1	28	0	1.0	23/03/2018	
6	7	1	29	1	2.0	04/02/2019	
7	8	0	30	0	1.0	11/11/2020	
8	9	1	31	1	2.0	31/07/2018	
9	10	0	32	0	1.0	03/05/2021	
10	11	0	33	1	2.0	16/06/2021	
11	12	0	34	0	1.0	26/03/2021	
12	13	0	35	1	2.0	20/02/2021	
13	14	0	36	0	1.0	18/12/2020	
14	15	0	37	1	2.0	18/12/2020	
15	16	0	38	0	1.0	13/11/2020	
16	17	1	39	1	2.0	07/02/2019	
17	18	1	40	0	1.0	04/02/2019	
18	19	0	41	1	2.0	17/05/2021	
19	20	1	42	0	1.0	17/04/2020	
20	21	0	43	1	2.0	05/11/2020	

In [ ]:

In [40]:
```
#Listando as colunas do nosso Dataset
df_dados_2.columns.tolist()
```

Out[40]:
```
['ID_CLIENTE',
 'FORMA_AQUISICAO',
 'IDADE_CLIENTE',
 'SEXO',
 'QT_FILHOS',
 'DT_AQUISICAO',
 'DT_CANCELAMENTO',
 'DIAS_ATIVO',
 'MESES_ATIVO',
 'DURACAO_CONTRATO',
 'VL_PLANO_ADESAO',
 'VL_PLANO_ATUAL',
 'NOME_PRODUTO',
 'QT_PONTOS_INSTALADOS',
 'QT_PC_PAGAS',
 'QT_PC_VENCIDAS',
 'QT_PC_PAGA_ATRASO',
 'QT_PC_PAGA_EM_DIA',
 'QT_ACORDO_PAGAMENTO',
 'VL_MENSALIDADE_ATRASO',
 'VL_MENSALIDADE_DT_AQUISICAO',
 'VL_MENSALIDADE_DT_ATUAL',
 'SITUACAO',
 'COD_SITUACAO',
 'NIVEL_PAGAMENTO']
```

In [41]:
```
# Vamos filtrar e utilizar somente as colunas necessárias
columns = ['FORMA_AQUISICAO',
 'IDADE_CLIENTE',
 'SEXO',
 'QT_FILHOS',
 'DIAS_ATIVO',
 'MESES_ATIVO',
 'DURACAO_CONTRATO',
 'VL_PLANO_ADESAO',
 'VL_PLANO_ATUAL',
 'NOME_PRODUTO',
 'QT_PONTOS_INSTALADOS',
 'QT_PC_PAGAS',
 'QT_PC_VENCIDAS',
 'QT_PC_PAGA_ATRASO',
 'QT_PC_PAGA_EM_DIA',
 'QT_ACORDO_PAGAMENTO',
 'VL_MENSALIDADE_ATRASO',
 'VL_MENSALIDADE_DT_AQUISICAO',
 'VL_MENSALIDADE_DT_ATUAL',
 'NIVEL_PAGAMENTO',
 'COD_SITUACAO']

df_dados_2 = pd.DataFrame(df_dados_2, columns=columns)
```

In [42]:
```
# Visualizando as primeiras linhas do arquivo novamente
df_dados_2.head()
```

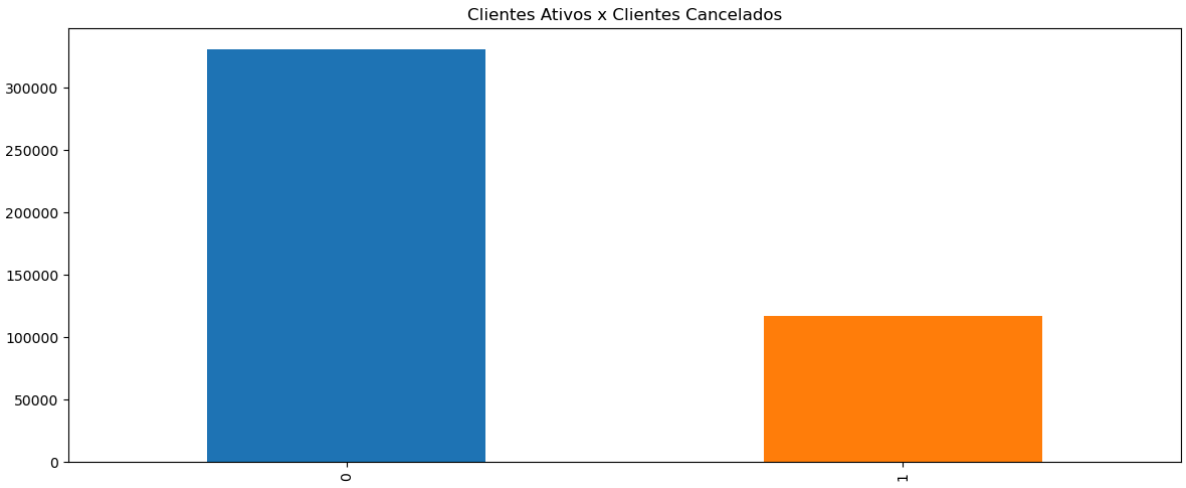
Out[42]:

	FORMA_AQUISICAO	IDADE_CLIENTE	SEXO	QT_FILHOS	DIAS_ATIVO	MESES_ATIVO	DURACAO_
0	0	23	1	0.0	33	1	
1	1	24	0	0.0	1198	39	
2	0	25	1	0.0	285	9	
4	1	27	1	0.0	671	22	
5	1	28	0	1.0	1216	40	

In [43]:
```
# Verificando os tipos de variáveis
df_dados_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 448188 entries, 0 to 448446
Data columns (total 21 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   FORMA_AQUISICAO              448188 non-null  int32
 1   IDADE_CLIENTE                448188 non-null  int64
 2   SEXO                         448188 non-null  int32
 3   QT_FILHOS                    448188 non-null  float64
 4   DIAS_ATIVO                   448188 non-null  int64
 5   MESES_ATIVO                  448188 non-null  int64
 6   DURACAO_CONTRATO             448188 non-null  int64
 7   VL_PLANO_ADESAO              448188 non-null  int64
 8   VL_PLANO_ATUAL               448188 non-null  int64
 9   NOME_PRODUTO                 448188 non-null  int32
 10  QT_PONTOS_INSTALADOS         448188 non-null  int64
 11  QT_PC_PAGAS                  448188 non-null  int64
 12  QT_PC_VENCIDAS               448188 non-null  int64
 13  QT_PC_PAGA_ATRASO            448188 non-null  int64
 14  QT_PC_PAGA_EM_DIA            448188 non-null  int64
 15  QT_ACORDO_PAGAMENTO          448188 non-null  int64
 16  VL_MENSALIDADE_ATRASO        448188 non-null  int64
 17  VL_MENSALIDADE_DT_AQUISICAO  448188 non-null  int64
 18  VL_MENSALIDADE_DT_ATUAL      448188 non-null  int64
 19  NIVEL_PAGAMENTO              448188 non-null  int32
 20  COD_SITUACAO                 448188 non-null  int64
dtypes: float64(1), int32(4), int64(16)
memory usage: 84.5 MB
```

In [44]:
```
#Analisando como a variavel alvo está distribuida.
#Aqui podemos observar que há muito mais CLIENTES ATIVOS do que CLIENTES CANCELADOS
#dessa forma, precisaremos balancear o dataset na etapa 2 - Tratamento de Dados.
plt.rcParams["figure.figsize"] = [12.00, 5.00]
plt.rcParams["figure.autolayout"] = True
df_dados_2.COD_SITUACAO.value_counts().plot(kind='bar', title='Clientes Ativos x Cl
```



In [45]:
```
#Separar variaveis preditoras e target
PREDITORAS = df_dados_2.iloc[:, 0:20]
TARGET = df_dados_2.iloc[:, 20]
```

In [46]:
```
PREDITORAS.head()
```

Out[46]:

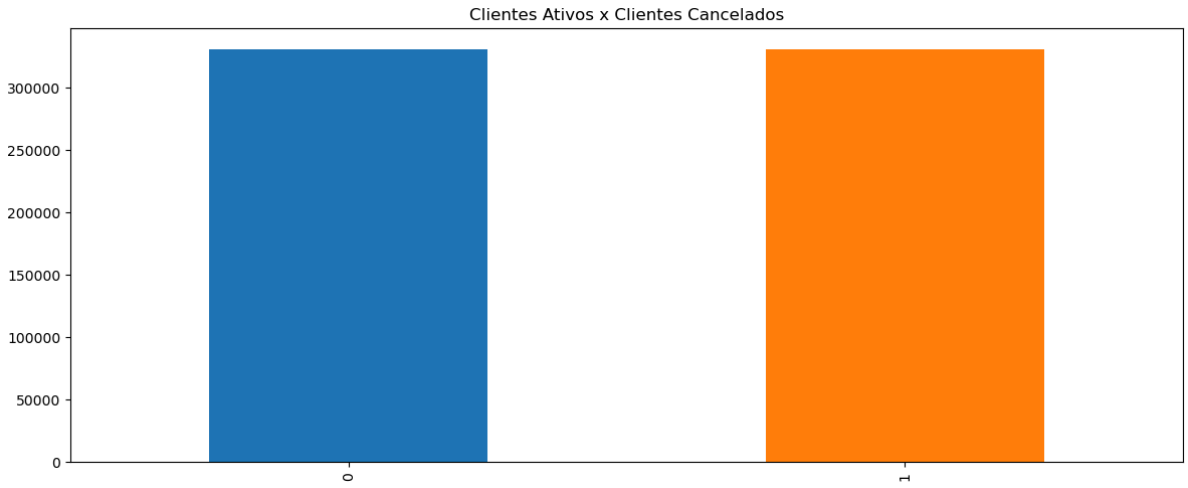
	FORMA_AQUISICAO	IDADE_CLIENTE	SEXO	QT_FILHOS	DIAS_ATIVO	MESES_ATIVO	DURACAO_
0	0	23	1	0.0	33	1	
1	1	24	0	0.0	1198	39	
2	0	25	1	0.0	285	9	
4	1	27	1	0.0	671	22	
5	1	28	0	1.0	1216	40	

In [47]:
```
TARGET.head()
```

Out[47]:
```
0    0
1    0
2    0
4    0
5    0
Name: COD_SITUACAO, dtype: int64
```

In [48]:
```
# Seed para reproduzir o mesmo resultado
seed = 100

# Cria o balanceador SMOTE
balanceador = SMOTE(random_state = seed)

# Aplica o balanceador
PREDITORAS_RES, TARGET_RES = balanceador.fit_resample(PREDITORAS, TARGET)
```

In [49]:
```
# Visualizando o balanceamento da variável TARGET
plt.rcParams["figure.figsize"] = [12.00, 5.00]
plt.rcParams["figure.autolayout"] = True
TARGET_RES.value_counts().plot(kind='bar', title='Clientes Ativos x Clientes Cancel
```



In [50]:
```
# Quantidade de registros antes do balanceamento
PREDITORAS.shape
```

Out[50]:
```
(448188, 20)
```

In [51]:
```
# Quantidade de registros antes do balanceamento
TARGET.shape
```

Out[51]:
```
(448188,)
```

In [52]:
```
# Quantidade de registros após do balanceamento
PREDITORAS_RES.shape
```

Out[52]:
```
(661974, 20)
```

In [53]:
```
# Quantidade de registros após do balanceamento
TARGET_RES.shape
```

Out[53]:
```
(661974,)
```

In [54]:
```
# Divisão em Dados de Treino e Teste.
X_treino, X_teste, Y_treino, Y_teste = train_test_split(PREDITORAS_RES, TARGET_RES,
```

In [55]:
```
# Padronizando as Variáveis - Pré Processamento dos Dados
Padronizador = StandardScaler()
X_treino_padronizados = Padronizador.fit_transform(X_treino)
X_teste_padronizados = Padronizador.transform(X_teste)
```

In [56]:
```
# Visualizando os dados padronizados
X_treino_padronizados
```

Out[56]:
```
array([[-0.69174349,  0.53257787,  1.05970352, ...,  1.41114896,
         1.41223978, -0.94609812],
       [ 1.44562258, -0.02654558, -0.94366017, ...,  1.41114896,
         1.41223978,  1.05540342],
       [-0.69174349, -0.02654558, -0.94366017, ...,  2.81634037,
         2.81422044, -0.94609812],
       ...,
       [-0.69174349, -1.56413505, -0.94366017, ..., -0.64979845,
        -0.64997596, -0.94609812],
       [ 1.44562258,  0.53257787, -0.94366017, ...,  0.47435468,
         0.47486899,  1.05540342],
       [ 1.44562258,  0.53257787, -0.94366017, ...,  1.41114896,
         1.41223978, -0.94609812]])
```

In [57]:
```
# Range de valores de k que iremos testar
kVals = range(3, 10, 2)
```

In [58]:
```
# Lista vazia para receber as acurácias
acuracias = []
```

In [ ]:
```
start = time.time()
for k in kVals:

    # Treinando o modelo KNN com cada valor de k
    modeloKNN = KNeighborsClassifier(n_neighbors = k)
    modeloKNN.fit(X_treino_padronizados, Y_treino)

    # Avaliando o modelo e atualizando a lista de acurácias
    score = modeloKNN.score(X_teste_padronizados, Y_teste)
    print("Com valor de k = %d, a acurácia é = %.2f%%" % (k, score * 100))
    acuracias.append(score)
end = time.time()
print('Tempo de Treinamento do Modelo:', end - start)
```

```
Com valor de k = 3, a acurácia é = 97.73%
Com valor de k = 5, a acurácia é = 97.43%
```

In [ ]:
```
# Obtendo o valor de k que apresentou a maior acurácia
i = np.argmax(acuracias)
print("O valor de k = %d alcançou a mais alta acurácia de %.2f%% nos dados de valic
```

In [ ]:
```
# Criando a versão final do modelo com o maior valor de k
modeloFinal = KNeighborsClassifier(n_neighbors = kVals[i])
```

In [ ]:
```
# Treinamento do modelo
# Observe que neste caso usou a metrica de distancia de minkowski mas isso podemos
modeloFinal.fit(X_treino_padronizados, Y_treino)
```

In [ ]:
```
# Previsões com os dados de teste
previsoes = modeloFinal.predict(X_teste_padronizados)
```

In [ ]:
```
print('Acurácia do modelo: ' , accuracy_score(Y_teste, previsoes))
```

In [ ]:

In [ ]: