DATA ANALYSIS

Use Case: Customer Churn Analysis

Finding out factors that affect the customer churn from a telecommunications service provider, with "churn" being referred to as "the state where the customer does not pay their monthly subscription and their service being terminated.

Datasets and Their Data Dictionary

There are two datasets namely *accounts.csv* and *billing.csv*. Both datasets share the same CustomerID field, which is a unique identifier for each customer in the systems.

Accounts (accounts.csv)

From the Customer Care department, this is the list of all of our customers with the following data fields:

- **Region:** the region the customer is located in
- **Gender:** the customer's gender
- **Partner:** if the customer lives with a partner or is married
- **Dependents:** if the customer has any children/dependents
- **Tenure:** the length of time (in months) this customer has been with the telecom
- **CustomerServiceCalls:** the number of calls a customer placed to Customer Care in the past month

Billing (billing.csv)

From the Credit and Collections department, this is the list of all of our customers' service plans and billing-related information:

- **PhoneService:** if the customer has phone service
- **InternetService:** the kind of Internet service of the customer (if any)
- **OnlineSecurity:** (0,1) if the customer has a security package
- **StreamingTV:** (0,1) if the customer has a streaming TV package
- **LockedIn:** (0,1) if the customer will suffer a penalty fee if they terminate service or if their contract term no longer includes this fee
- **PaperlessBilling:** (0,1) if the customer is enrolled in paperless billing
- **DominantPaymentMethod:** from historical data, the most common kind of payment method the customer uses to pay their bill
- **MonthlyCharges:** the amount (in Pesos) a customer owes every month
- **Churn:** (0,1) if a customer churned in the current month, as prev. defined

Importing Libraries

In [238]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_style("whitegrid")
sns.set_context("poster")

%matplotlib inline
```

Python Libraries Imported:

- Numpy -
- Pandas -
- Seaborn -
- Matplotlib -

In [239]:
```
#Filter Warnings
import warnings
warnings.filterwarnings('ignore')

# Set Options for display
pd.options.display.max_rows = 100
pd.options.display.max_columns = 100
pd.options.display.float_format = '{:.2f}'.format
```

Loading the Datasets

In [240]:
```
df_a = pd.read_csv('accounts.csv')
df_b = pd.read_csv('billing.csv')
```

Exploratory Data Analysis (EDA) and Data Preparation

EDA is done to get a grasp on what the dataset is all about. These also involves the following steps:

- Checking the shape of dataset
- Checking if both datasets share the same CustomerID to be able to merge easily (assuming that it is raw and still needs to be checked)
- Changing dataset into machine-readable

Describing the Data

In this part, we take an initial look on our datasets in order to have an initial insight in the dataset.

Accounts dataset

In [241]:
```
#Viewing a sample of the data to check if it is loaded properly and to get a g
rasp what the data is all about
df_a.head()
```

Out[241]:

	customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls
0	7892-POOKP	National Capital Region	Female	1	0	28	1
1	0280-XJGEX	National Capital Region	Male	0	0	4900	6
2	8779-QRDMV	National Capital Region	Male	0	0	1	5
3	1066-JKSGK	National Capital Region	Male	0	0	1	3
4	8665-UTDHZ	National Capital Region	Male	1	1	1	3

In [242]:
```
#Checking the descriptive statistics of the dataset
df_a.describe()
```

Out[242]:

	Partner	Dependents	Tenure	CustomerServiceCalls
count	7043.00	7043.00	7043.00	7043.00
mean	0.48	0.30	33.06	1.35
std	0.50	0.46	62.99	1.37
min	0.00	0.00	0.00	0.00
25%	0.00	0.00	9.00	1.00
50%	0.00	0.00	29.00	1.00
75%	1.00	1.00	55.00	2.00
max	1.00	1.00	4900.00	7.00

In [243]:
```
#Checking the features and its column names, datatypes, null values, and count
s
df_a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 7 columns):
customerID              7043 non-null object
Region                  7043 non-null object
Gender                  7043 non-null object
Partner                 7043 non-null int64
Dependents              7043 non-null int64
Tenure                  7043 non-null int64
CustomerServiceCalls    7043 non-null int64
dtypes: int64(4), object(3)
memory usage: 385.2+ KB
```

In [244]:
```
#Checking the dimensions of the dataset
df_a.shape
```

Out[244]: (7043, 7)

In [245]:
```
#Sorting the datasets in ascending order based on the CustomerID
df_a.sort_values(["customerID"], axis=0,ascending=True, inplace=True)
df_a.head(20)
```

Out[245]:

	customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls
2919	0002-ORFBO	North Luzon	Female	1	1	9	2
3961	0003-MKNFE	South Luzon	Male	0	0	9	1
512	0004-TLHLJ	South Luzon	Male	0	0	4	0
788	0011-IGKFF	NCR	Male	1	0	13	1
5757	0013-EXCHZ	Mindanao	Female	1	0	3	2
1976	0013-MHZWF	NCR	Female	0	1	9	1
2984	0013-SMEOE	North Luzon	Female	1	0	71	1
3205	0014-BMAQU	North Luzon	Male	1	0	63	1
2898	0015-UOCOJ	North Luzon	Female	0	0	7	1
5018	0016-QLJIS	Visayas	Female	1	1	65	0
2661	0017-DINOC	North Luzon	Male	0	0	54	1
4579	0017-IUDMW	Visayas	Female	1	1	72	1
5071	0018-NYROU	Visayas	Female	1	0	5	1
2621	0019-EFAEP	North Luzon	Female	0	0	72	2
4418	0019-GFNTW	Visayas	Female	0	0	56	1
2955	0020-INWCK	North Luzon	male	1	1	71	0
6149	0020-JDNXP	NCR	Female	1	1	34	1
1227	0021-IKXGC	NCR	Female	0	0	1	0
343	0022-TCJCI	North Luzon	Male	0	0	45	7
585	0023-HGHWL	Visayas	Male	0	0	1	1

In [246]:
```
#Checking for duplicated values
print('Train set duplicate IDs: {}'.format(df_a.duplicated().sum()))
```

```
Train set duplicate IDs: 0
```

In [247]:
```
#Checking the tail or end of the dataset to see the last CustomerID recorded
df_a.tail(10)
```

Out[247]:

	customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls
618	9965-YOKZB	Visayas	Male	0	0	9	5
1270	9967-ATRFS	NCR	Female	0	0	19	2
2041	9968-FFVVH	NCR	Male	0	0	63	1
5658	9970-QBCDA	Mindanao	Female	0	0	6	1
6075	9971-ZWPBF	NCR	Male	1	1	34	2
2234	9972-EWRJS	NCR	Female	1	1	67	1
5093	9972-NKTFD	Visayas	Female	0	0	28	0
6979	9972-VAFJJ	North Luzon	Female	1	0	53	1
4478	9974-JFBHQ	Visayas	Male	0	1	64	1
4611	9975-GPKZU	Visayas	Male	1	1	46	0
5216	9975-SKRNR	Visayas	Male	0	0	1	1
1534	9978-HYCIN	NCR	Male	1	1	47	1
6682	9979-RGMZT	Nor. Luz.	Female	0	0	7	0
5923	9985-MWVIX	Mindanao	Female	0	0	1	0
4079	9986-BONCE	South Luzon	Female	0	0	4	1
4059	9987-LUTYD	South Luzon	Female	0	0	13	1
3416	9992-RRAMN	North Luzon	Male	1	0	22	0
2015	9992-UJOEL	NCR	Male	0	0	2	1
2655	9993-LHIEB	North Luzon	Male	1	1	67	1
2338	9995-HOTOH	NCR	Male	1	1	63	1

In [248]:

```
#Setting our index of the dataset through their CustomerID, as its unique iden
tifier
df_a.set_index('customerID')
```

Out[248]:

customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls
0002-ORFBO	North Luzon	Female	1	1	9	2
0003-MKNFE	South Luzon	Male	0	0	9	1
0004-TLHLJ	South Luzon	Male	0	0	4	0
0011-IGKFF	NCR	Male	1	0	13	1
0013-EXCHZ	Mindanao	Female	1	0	3	2
0013-MHZWF	NCR	Female	0	1	9	1
0013-SMEOE	North Luzon	Female	1	0	71	1
0014-BMAQU	North Luzon	Male	1	0	63	1
0015-UOCOJ	North Luzon	Female	0	0	7	1
0016-QLJIS	Visayas	Female	1	1	65	0
0017-DINOC	North Luzon	Male	0	0	54	1
0017-IUDMW	Visayas	Female	1	1	72	1
0018-NYROU	Visayas	Female	1	0	5	1
0019-EFAEP	North Luzon	Female	0	0	72	2
0019-GFNTW	Visayas	Female	0	0	56	1
0020-INWCK	North Luzon	male	1	1	71	0
0020-JDNXP	NCR	Female	1	1	34	1
0021-IKXGC	NCR	Female	0	0	1	0
0022-TCJCI	North Luzon	Male	0	0	45	7
0023-HGHWL	Visayas	Male	0	0	1	1
0023-UYUPN	North Luzon	Female	1	0	50	1
0023-XUOPT	Nor. Luz.	Female	1	0	13	5
0027-KWYKW	Mindanao	Female	1	1	23	1
0030-FNXPP	NCR	Female	0	0	3	2
0031-PVLZI	NCR	Female	1	1	4	0
0032-PGELS	North Luzon	Female	1	1	1	2
0036-IHMOT	NCR	Female	1	1	55	1
0040-HALCW	NCR	Male	1	1	54	0
0042-JVWOJ	NCR	Male	0	0	26	2
0042-RLHYP	NCR	Female	1	1	69	0
0048-LUMLS	NCR	Male	1	1	37	0
0048-PIHNL	Nor. Luz.	Female	1	0	49	1
0052-DCKON	Visayas	Male	1	0	66	2
0052-YNYOT	North Luzon	Female	0	0	67	0
0056-EPFBG	Visayas	Male	1	1	20	2
0057-QBUQH	NCR	Female	0	1	43	0
0058-EVZWM	North Luzon	Female	1	0	55	1
0060-FUALY	Visayas	Female	1	0	59	0
0064-SUDOG	Nor. Luz.	Female	1	1	12	1
0064-YIJGF	South Luzon	Male	1	1	27	1
0067-DKWBL	NCR	Male	0	0	2	5

customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls
0068-FIGTF	NCR	Female	0	0	27	0
0071-NDAFP	NCR	Male	1	1	25	2
0074-HDKDG	NCR	Male	1	1	25	1
0076-LVEPS	North Luzon	Male	0	1	29	0
0078-XZMHT	NCR	Male	1	0	72	1
0080-EMYVY	NCR	Female	0	0	14	1
0080-OROZO	NCR	Female	0	0	35	2
0082-LDZUE	NCR	Male	0	0	1	0
0082-OQIQY	Visayas	Male	0	0	29	2
...	...	...	...	...	...	...
9924-JPRMC	NCR	Male	0	0	72	0
9926-PJHDQ	NCR	Female	1	1	72	1
9927-DSWDF	Visayas	Male	1	0	22	1
9928-BZVLZ	Nor. Luz.	Female	0	0	12	1
9929-PLVPA	North Luzon	Female	0	1	4	2
9931-DCEZH	South Luzon	Male	0	1	28	1
9931-KGHOA	NCR	Female	1	0	46	0
9932-WBWIK	Mindanao	Male	0	0	11	0
9933-QRGTX	NCR	Female	1	0	60	0
9938-EKRGF	South Luzon	Female	0	0	15	0
9938-PRCVK	NCR	Female	1	1	41	0
9938-TKDGL	Visayas	Male	1	1	68	2
9938-ZREHM	Visayas	Female	1	0	37	1
9940-HPQPG	North Luzon	Female	1	0	9	3
9940-RHLFB	Vis.	Female	0	0	1	2
9943-VSZUV	NCR	Male	0	0	67	0
9944-AEXBM	NCR	Male	0	0	32	0
9944-HKVVB	NCR	Female	0	0	3	2
9945-PSVIP	North Luzon	Female	1	1	25	1
9947-OTFQU	NCR	Male	0	0	15	1
9948-YPTDG	NCR	Male	1	0	38	7
9950-MTGYX	Visayas	Male	1	1	28	1
9953-ZMKSM	North Luzon	Male	0	0	63	1
9955-QOPOY	Visayas	Male	1	0	69	2
9957-YODKZ	NCR	Male	1	0	6	0
9958-MEKUC	Mindanao	Male	1	1	72	1
9959-WOFKT	National Capital Region	Male	0	1	71	2
9961-JBNMK	North Luzon	Male	0	0	21	4
9962-BFPDU	NCR	Female	1	1	1	1
9964-WBQDJ	NCR	Female	1	0	71	0
9965-YOKZB	Visayas	Male	0	0	9	5
9967-ATRFS	NCR	Female	0	0	19	2
9968-FFVVH	NCR	Male	0	0	63	1
9970-QBCDA	Mindanao	Female	0	0	6	1
9971-ZWPBF	NCR	Male	1	1	34	2

customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls
9972-EWRJS	NCR	Female	1	1	67	1
9972-NKTFD	Visayas	Female	0	0	28	0
9972-VAFJJ	North Luzon	Female	1	0	53	1
9974-JFBHQ	Visayas	Male	0	1	64	1
9975-GPKZU	Visayas	Male	1	1	46	0
9975-SKRNR	Visayas	Male	0	0	1	1
9978-HYCIN	NCR	Male	1	1	47	1
9979-RGMZT	Nor. Luz.	Female	0	0	7	0
9985-MWVIX	Mindanao	Female	0	0	1	0
9986-BONCE	South Luzon	Female	0	0	4	1
9987-LUTYD	South Luzon	Female	0	0	13	1
9992-RRAMN	North Luzon	Male	1	0	22	0
9992-UJOEL	NCR	Male	0	0	2	1
9993-LHIEB	North Luzon	Male	1	1	67	1
9995-HOTOH	NCR	Male	1	1	63	1

7043 rows × 6 columns

Billing dataset

In [249]:
```
#Viewing a sample of the data to check if it is loaded properly and to get a g
rasp what the data is all about
df_b.head()
```

Out[249]:

	customerID	PhoneService	InternetService	OnlineSecurity	StreamingTV	LockedIn	PaperlessI
0	8905-IAZPF	Yes	Fiber	0	1	1	
1	8747-UDCOI	Yes	No	0	0	1	
2	5485-ITNPC	Yes	DSL	1	1	1	
3	5666-MBJPT	Yes	No	1	0	1	
4	9938-ZREHM	Yes	DSL	0	0	1	

In [250]:
```
#Checking the descriptive statistics of the dataset
df_b.describe()
```

Out[250]:

	OnlineSecurity	StreamingTV	LockedIn	PaperlessBilling	MonthlyCharges	Churn
count	7043.00	7043.00	7043.00	7043.00	7043.00	7043.00
mean	0.41	0.42	0.45	0.14	1293.93	0.13
std	0.49	0.49	0.50	0.35	603.11	0.34
min	0.00	0.00	0.00	0.00	-1.00	0.00
25%	0.00	0.00	0.00	0.00	710.00	0.00
50%	0.00	0.00	0.00	0.00	1410.00	0.00
75%	1.00	1.00	1.00	0.00	1800.00	0.00
max	1.00	1.00	1.00	1.00	2380.00	1.00

In [251]:
```
#Checking the features,its datatypes, null values, and counts
df_b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 10 columns):
customerID              7043 non-null object
PhoneService            7043 non-null object
InternetService         7043 non-null object
OnlineSecurity          7043 non-null int64
StreamingTV             7043 non-null int64
LockedIn                7043 non-null int64
PaperlessBilling        7043 non-null int64
DominantPaymentMethod   7043 non-null object
MonthlyCharges          7043 non-null int64
Churn                   7043 non-null int64
dtypes: int64(6), object(4)
memory usage: 550.3+ KB
```

In [252]:
```
#Checking the dimensions of the dataset
df_b.shape
```

Out[252]:
```
(7043, 10)
```

In [253]:
```
#Sorting the datasets in ascending order based on the CustomerID
df_b.sort_values(["customerID"], axis=0,ascending=True, inplace=True)
df_b.head(10)
```

Out[253]:

	customerID	PhoneService	InternetService	OnlineSecurity	StreamingTV	LockedIn	Paperle
4267	0002-ORFBO	Yes	DSL	0	1	1	
6213	0003-MKNFE	Yes	DSL	0	0	0	
3355	0004-TLHLJ	Yes	Fiber	0	0	0	
6622	0011-IGKFF	Yes	Fiber	0	1	0	
4941	0013-EXCHZ	Yes	Fiber	0	1	0	
3005	0013-MHZWF	Yes	DSL	0	1	0	
1256	0013-SMEOE	Yes	Fiber	1	1	1	
6325	0014-BMAQU	Yes	Fiber	1	0	1	
5382	0015-UOCOJ	Yes	DSL	1	0	0	
5707	0016-QLJIS	Yes	DSL	1	1	1	
261	0017-DINOC	Yes	DSL	1	1	1	
726	0017-IUDMW	Yes	Fiber	1	1	1	
1318	0018-NYROU	Yes	Fiber	0	0	0	
749	0019-EFAEP	Yes	Fiber	1	1	1	
3707	0019-GFNTW	Yes	DSL	1	0	1	
5032	0020-INWCK	Yes	Fiber	0	0	1	
4720	0020-JDNXP	Yes	DSL	1	1	1	
2111	0021-IKXGC	Yes	Fiber	0	0	0	
842	0022-TCJCI	Yes	DSL	1	0	1	
2395	0023-HGHWL	Yes	DSL	0	0	0	

In [254]:
```
#Checking for duplicated values
print('Train set duplicate IDs: {}'.format(df_b.duplicated().sum()))
```

```
Train set duplicate IDs: 0
```

In [255]:
```
#Checking the tail or end of the dataset to see the last CustomerID recorded
df_b.tail(10)
```

Out[255]:

	customerID	PhoneService	InternetService	OnlineSecurity	StreamingTV	LockedIn	Paperle
6326	9965-YOKZB	Yes	Fiber	0	0	0	
742	9967-ATRFS	Yes	No	0	0	0	
2791	9968-FFVVH	Yes	DSL	1	0	1	
1996	9970-QBCDA	Yes	No	0	0	0	
4844	9971-ZWPBF	Yes	Fiber	1	1	0	
4854	9972-EWRJS	Yes	No	0	0	1	
3681	9972-NKTFD	Yes	DSL	0	0	0	
5391	9972-VAFJJ	Yes	Fiber	1	1	1	
2850	9974-JFBHQ	Yes	Fiber	0	1	0	
6854	9975-GPKZU	Yes	No	0	0	1	
6178	9975-SKRNR	Yes	No	0	0	0	
976	9978-HYCIN	Yes	Fiber	0	1	1	
113	9979-RGMZT	Yes	Fiber	1	1	1	
1673	9985-MWVIX	Yes	Fiber	1	0	0	
324	9986-BONCE	Yes	No	0	0	0	
1020	9987-LUTYD	Yes	DSL	1	0	1	
6970	9992-RRAMN	Yes	Fiber	0	0	0	
6405	9992-UJOEL	Yes	DSL	0	0	0	
2226	9993-LHIEB	Yes	DSL	1	0	1	
2711	9995-HOTOH	Yes	DSL	1	1	1	

In [256]:

```
#Setting CustomerID as the index of the dataset, its unique identifier
df_b.set_index('customerID')
```

Out[256]:

customerID	PhoneService	InternetService	OnlineSecurity	StreamingTV	LockedIn	PaperlessBilli
0002-ORFBO	Yes	DSL	0	1	1	
0003-MKNFE	Yes	DSL	0	0	0	
0004-TLHLJ	Yes	Fiber	0	0	0	
0011-IGKFF	Yes	Fiber	0	1	0	
0013-EXCHZ	Yes	Fiber	0	1	0	
0013-MHZWF	Yes	DSL	0	1	0	
0013-SMEOE	Yes	Fiber	1	1	1	
0014-BMAQU	Yes	Fiber	1	0	1	
0015-UOCOJ	Yes	DSL	1	0	0	
0016-QLJIS	Yes	DSL	1	1	1	
0017-DINOC	Yes	DSL	1	1	1	
0017-IUDMW	Yes	Fiber	1	1	1	
0018-NYROU	Yes	Fiber	0	0	0	
0019-EFAEP	Yes	Fiber	1	1	1	
0019-GFNTW	Yes	DSL	1	0	1	
0020-INWCK	Yes	Fiber	0	0	1	
0020-JDNXP	Yes	DSL	1	1	1	
0021-IKXGC	Yes	Fiber	0	0	0	
0022-TCJCI	Yes	DSL	1	0	1	
0023-HGHWL	Yes	DSL	0	0	0	
0023-UYUPN	Yes	No	0	0	1	
0023-XUOPT	Yes	Fiber	0	1	0	
0027-KWYKW	Yes	Fiber	0	1	0	
0030-FNXPP	Yes	No	0	0	0	
0031-PVLZI	Yes	No	0	0	0	
0032-PGELS	Yes	DSL	1	1	0	
0036-IHMOT	Yes	Fiber	0	1	1	
0040-HALCW	Yes	No	0	0	1	
0042-JVWOJ	Yes	No	1	0	1	
0042-RLHYP	Yes	No	0	0	1	
0048-LUMLS	Yes	Fiber	0	1	1	

	PhoneService	InternetService	OnlineSecurity	StreamingTV	LockedIn	PaperlessBilli
customerID						
0048-PIHNL	Yes	No	1	1	1	
0052-DCKON	Yes	Fiber	1	1	1	
0052-YNYOT	Yes	No	1	1	1	
0056-EPFBG	Yes	DSL	1	0	1	
0057-QBUQH	Yes	No	1	0	1	
0058-EVZWM	Yes	Fiber	1	1	0	
0060-FUALY	Yes	Fiber	1	1	0	
0064-SUDOG	Yes	No	1	1	1	
0064-YIJGF	Yes	Fiber	0	0	0	
0067-DKWBL	Yes	No	1	0	0	
0068-FIGTF	Yes	DSL	1	1	1	
0071-NDAFP	Yes	No	1	0	1	
0074-HDKDG	Yes	DSL	1	0	1	
0076-LVEPS	Yes	DSL	1	1	0	
0078-XZMHT	Yes	DSL	0	1	1	
0080-EMYVY	Yes	DSL	1	0	1	
0080-OROZO	Yes	Fiber	0	1	1	
0082-LDZUE	Yes	DSL	1	0	0	
0082-OQIQY	Yes	Fiber	0	1	0	
...	...	...	...	...	...	
9924-JPRMC	Yes	Fiber	1	1	1	
9926-PJHDQ	Yes	DSL	0	1	1	
9927-DSWDF	Yes	Fiber	1	1	0	
9928-BZVLZ	Yes	DSL	1	1	1	
9929-PLVPA	Yes	No	0	0	0	
9931-DCEZH	Yes	DSL	0	0	1	
9931-KGHOA	Yes	DSL	1	0	0	
9932-WBWIK	Yes	No	0	0	0	
9933-QRGTX	Yes	Fiber	1	1	1	
9938-EKRGF	Yes	DSL	1	1	0	
9938-PRCVK	Yes	No	0	0	1	

customerID	PhoneService	InternetService	OnlineSecurity	StreamingTV	LockedIn	PaperlessBilli
9938-TKDGL	Yes	Fiber	1	1	1	
9938-ZREHM	Yes	DSL	0	0	1	
9940-HPQPG	Yes	Fiber	1	0	0	
9940-RHLFB	Yes	Fiber	0	0	0	
9943-VSZUV	Yes	Fiber	1	0	0	
9944-AEXBM	Yes	Fiber	0	0	0	
9944-HKVVB	Yes	No	0	1	0	
9945-PSVIP	Yes	No	1	1	1	
9947-OTFQU	Yes	Fiber	0	0	0	
9948-YPTDG	Yes	No	1	0	0	
9950-MTGYX	Yes	No	0	0	1	
9953-ZMKSM	Yes	No	0	0	1	
9955-QOPOY	Yes	DSL	0	1	1	
9957-YODKZ	Yes	Fiber	0	0	0	
9958-MEKUC	Yes	Fiber	1	0	1	
9959-WOFKT	Yes	Fiber	1	1	1	
9961-JBNMK	Yes	Fiber	0	1	0	
9962-BFPDU	Yes	No	1	0	0	
9964-WBQDJ	Yes	No	0	0	1	
9965-YOKZB	Yes	Fiber	0	0	0	
9967-ATRFS	Yes	No	0	0	0	
9968-FFVVH	Yes	DSL	1	0	1	
9970-QBCDA	Yes	No	0	0	0	
9971-ZWPBF	Yes	Fiber	1	1	0	
9972-EWRJS	Yes	No	0	0	1	
9972-NKTFD	Yes	DSL	0	0	0	
9972-VAFJJ	Yes	Fiber	1	1	1	
9974-JFBHQ	Yes	Fiber	0	1	0	
9975-GPKZU	Yes	No	0	0	1	
9975-SKRNR	Yes	No	0	0	0	

	PhoneService	InternetService	OnlineSecurity	StreamingTV	LockedIn	PaperlessBilli
customerID						
9978-HYCIN	Yes	Fiber	0	1	1	
9979-RGMZT	Yes	Fiber	1	1	1	
9985-MWVIX	Yes	Fiber	1	0	0	
9986-BONCE	Yes	No	0	0	0	
9987-LUTYD	Yes	DSL	1	0	1	
9992-RRAMN	Yes	Fiber	0	0	0	
9992-UJOEL	Yes	DSL	0	0	0	
9993-LHIEB	Yes	DSL	1	0	1	
9995-HOTOH	Yes	DSL	1	1	1	

7043 rows × 9 columns

Comparing that both datasets share the same number, shape, no duplicate and null values and starts and ends with the same customerIDs, merging and starting the rest of EDA and Data Preparation is the next step.

Merging both datasets

In [257]:
```
#Merging the dataset into one inorder to work on EDA and data Cleaning easily,
and be able to match it to its corresponding CustomerID
merged_dataset = df_a.merge(df_b, left_on='customerID', right_on='customerID')
```

After merging, Checking if the dataset of both sides fit together by getting a sample from the head and tail and see if it is still the same shape.

In [258]:
```
merged_dataset.head(10)
```

Out[258]:

	customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls	PhoneS
0	0002-ORFBO	North Luzon	Female	1	1	9	2	
1	0003-MKNFE	South Luzon	Male	0	0	9	1	
2	0004-TLHLJ	South Luzon	Male	0	0	4	0	
3	0011-IGKFF	NCR	Male	1	0	13	1	
4	0013-EXCHZ	Mindanao	Female	1	0	3	2	
5	0013-MHZWF	NCR	Female	0	1	9	1	
6	0013-SMEOE	North Luzon	Female	1	0	71	1	
7	0014-BMAQU	North Luzon	Male	1	0	63	1	
8	0015-UOCOJ	North Luzon	Female	0	0	7	1	
9	0016-QLJIS	Visayas	Female	1	1	65	0	
10	0017-DINOC	North Luzon	Male	0	0	54	1	
11	0017-IUDMW	Visayas	Female	1	1	72	1	
12	0018-NYROU	Visayas	Female	1	0	5	1	
13	0019-EFAEP	North Luzon	Female	0	0	72	2	
14	0019-GFNTW	Visayas	Female	0	0	56	1	
15	0020-INWCK	North Luzon	male	1	1	71	0	
16	0020-JDNXP	NCR	Female	1	1	34	1	
17	0021-IKXGC	NCR	Female	0	0	1	0	
18	0022-TCJCI	North Luzon	Male	0	0	45	7	
19	0023-HGHWL	Visayas	Male	0	0	1	1	

In [259]:
```
#checking if the dataset matched with the CustomerId and has the same shape
merged_dataset.tail(10)
```

Out[259]:

	customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls	Phon
7013	9965-YOKZB	Visayas	Male	0	0	9	5	
7014	9967-ATRFS	NCR	Female	0	0	19	2	
7015	9968-FFVVH	NCR	Male	0	0	63	1	
7016	9970-QBCDA	Mindanao	Female	0	0	6	1	
7017	9971-ZWPBF	NCR	Male	1	1	34	2	
7018	9972-EWRJS	NCR	Female	1	1	67	1	
7019	9972-NKTFD	Visayas	Female	0	0	28	0	
7020	9972-VAFJJ	North Luzon	Female	1	0	53	1	
7021	9974-JFBHQ	Visayas	Male	0	1	64	1	
7022	9975-GPKZU	Visayas	Male	1	1	46	0	
7023	9975-SKRNR	Visayas	Male	0	0	1	1	
7024	9978-HYCIN	NCR	Male	1	1	47	1	
7025	9979-RGMZT	Nor. Luz.	Female	0	0	7	0	
7026	9985-MWVIX	Mindanao	Female	0	0	1	0	
7027	9986-BONCE	South Luzon	Female	0	0	4	1	
7028	9987-LUTYD	South Luzon	Female	0	0	13	1	
7029	9992-RRAMN	North Luzon	Male	1	0	22	0	
7030	9992-UJOEL	NCR	Male	0	0	2	1	
7031	9993-LHIEB	North Luzon	Male	1	1	67	1	
7032	9995-HOTOH	NCR	Male	1	1	63	1	

In [260]:

```
#Setting customerID as index, to view it more accurately and in descending ord
er
merged_dataset.set_index('customerID')
```

Out[260]:

customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls	PhoneServic
0002-ORFBO	North Luzon	Female	1	1	9	2	Ye
0003-MKNFE	South Luzon	Male	0	0	9	1	Ye
0004-TLHLJ	South Luzon	Male	0	0	4	0	Ye
0011-IGKFF	NCR	Male	1	0	13	1	Ye
0013-EXCHZ	Mindanao	Female	1	0	3	2	Ye
0013-MHZWF	NCR	Female	0	1	9	1	Ye
0013-SMEOE	North Luzon	Female	1	0	71	1	Ye
0014-BMAQU	North Luzon	Male	1	0	63	1	Ye
0015-UOCOJ	North Luzon	Female	0	0	7	1	Ye
0016-QLJIS	Visayas	Female	1	1	65	0	Ye
0017-DINOC	North Luzon	Male	0	0	54	1	Ye
0017-IUDMW	Visayas	Female	1	1	72	1	Ye
0018-NYROU	Visayas	Female	1	0	5	1	Ye
0019-EFAEP	North Luzon	Female	0	0	72	2	Ye
0019-GFNTW	Visayas	Female	0	0	56	1	Ye
0020-INWCK	North Luzon	male	1	1	71	0	Ye
0020-JDNXP	NCR	Female	1	1	34	1	Ye
0021-IKXGC	NCR	Female	0	0	1	0	Ye
0022-TCJCI	North Luzon	Male	0	0	45	7	Ye
0023-HGHWL	Visayas	Male	0	0	1	1	Ye
0023-UYUPN	North Luzon	Female	1	0	50	1	Ye
0023-XUOPT	Nor. Luz.	Female	1	0	13	5	Ye
0027-KWYKW	Mindanao	Female	1	1	23	1	Ye
0030-FNXPP	NCR	Female	0	0	3	2	Ye
0031-PVLZI	NCR	Female	1	1	4	0	Ye
0032-PGELS	North Luzon	Female	1	1	1	2	Ye
0036-IHMOT	NCR	Female	1	1	55	1	Ye
0040-HALCW	NCR	Male	1	1	54	0	Ye
0042-JVWOJ	NCR	Male	0	0	26	2	Ye
0042-RLHYP	NCR	Female	1	1	69	0	Ye

customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls	PhoneServic
0048-LUMLS	NCR	Male	1	1	37	0	Ye
0048-PIHNL	Nor. Luz.	Female	1	0	49	1	Ye
0052-DCKON	Visayas	Male	1	0	66	2	Ye
0052-YNYOT	North Luzon	Female	0	0	67	0	Ye
0056-EPFBG	Visayas	Male	1	1	20	2	Ye
0057-QBUQH	NCR	Female	0	1	43	0	Ye
0058-EVZWM	North Luzon	Female	1	0	55	1	Ye
0060-FUALY	Visayas	Female	1	0	59	0	Ye
0064-SUDOG	Nor. Luz.	Female	1	1	12	1	Ye
0064-YIJGF	South Luzon	Male	1	1	27	1	Ye
0067-DKWBL	NCR	Male	0	0	2	5	Ye
0068-FIGTF	NCR	Female	0	0	27	0	Ye
0071-NDAFP	NCR	Male	1	1	25	2	Ye
0074-HDKDG	NCR	Male	1	1	25	1	Ye
0076-LVEPS	North Luzon	Male	0	1	29	0	Ye
0078-XZMHT	NCR	Male	1	0	72	1	Ye
0080-EMYVY	NCR	Female	0	0	14	1	Ye
0080-OROZO	NCR	Female	0	0	35	2	Ye
0082-LDZUE	NCR	Male	0	0	1	0	Ye
0082-OQIQY	Visayas	Male	0	0	29	2	Ye
...	...	...	...	...	...	...	.
9924-JPRMC	NCR	Male	0	0	72	0	Ye
9926-PJHDQ	NCR	Female	1	1	72	1	Ye
9927-DSWDF	Visayas	Male	1	0	22	1	Ye
9928-BZVLZ	Nor. Luz.	Female	0	0	12	1	Ye
9929-PLVPA	North Luzon	Female	0	1	4	2	Ye
9931-DCEZH	South Luzon	Male	0	1	28	1	Ye
9931-KGHOA	NCR	Female	1	0	46	0	Ye
9932-WBWIK	Mindanao	Male	0	0	11	0	Ye
9933-QRGTX	NCR	Female	1	0	60	0	Ye
9938-EKRGF	South Luzon	Female	0	0	15	0	Ye

customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls	PhoneServic
9938-PRCVK	NCR	Female	1	1	41	0	Ye
9938-TKDGL	Visayas	Male	1	1	68	2	Ye
9938-ZREHM	Visayas	Female	1	0	37	1	Ye
9940-HPQPG	North Luzon	Female	1	0	9	3	Ye
9940-RHLFB	Vis.	Female	0	0	1	2	Ye
9943-VSZUV	NCR	Male	0	0	67	0	Ye
9944-AEXBM	NCR	Male	0	0	32	0	Ye
9944-HKVVB	NCR	Female	0	0	3	2	Ye
9945-PSVIP	North Luzon	Female	1	1	25	1	Ye
9947-OTFQU	NCR	Male	0	0	15	1	Ye
9948-YPTDG	NCR	Male	1	0	38	7	Ye
9950-MTGYX	Visayas	Male	1	1	28	1	Ye
9953-ZMKSM	North Luzon	Male	0	0	63	1	Ye
9955-QOPOY	Visayas	Male	1	0	69	2	Ye
9957-YODKZ	NCR	Male	1	0	6	0	Ye
9958-MEKUC	Mindanao	Male	1	1	72	1	Ye
9959-WOFKT	National Capital Region	Male	0	1	71	2	Ye
9961-JBNMK	North Luzon	Male	0	0	21	4	Ye
9962-BFPDU	NCR	Female	1	1	1	1	Ye
9964-WBQDJ	NCR	Female	1	0	71	0	Ye
9965-YOKZB	Visayas	Male	0	0	9	5	Ye
9967-ATRFS	NCR	Female	0	0	19	2	Ye
9968-FFVVH	NCR	Male	0	0	63	1	Ye
9970-QBCDA	Mindanao	Female	0	0	6	1	Ye
9971-ZWPBF	NCR	Male	1	1	34	2	Ye
9972-EWRJS	NCR	Female	1	1	67	1	Ye
9972-NKTFD	Visayas	Female	0	0	28	0	Ye
9972-VAFJJ	North Luzon	Female	1	0	53	1	Ye
9974-JFBHQ	Visayas	Male	0	1	64	1	Ye
9975-GPKZU	Visayas	Male	1	1	46	0	Ye

customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls	PhoneServic
9975-SKRNR	Visayas	Male	0	0	1	1	Ye
9978-HYCIN	NCR	Male	1	1	47	1	Ye
9979-RGMZT	Nor. Luz.	Female	0	0	7	0	Ye
9985-MWVIX	Mindanao	Female	0	0	1	0	Ye
9986-BONCE	South Luzon	Female	0	0	4	1	Ye
9987-LUTYD	South Luzon	Female	0	0	13	1	Ye
9992-RRAMN	North Luzon	Male	1	0	22	0	Ye
9992-UJOEL	NCR	Male	0	0	2	1	Ye
9993-LHIEB	North Luzon	Male	1	1	67	1	Ye
9995-HOTOH	NCR	Male	1	1	63	1	Ye

7033 rows × 15 columns

In [261]:
```
#Doing another data exploration for the merged dataset in terms of shape
merged_dataset.shape
```

Out[261]: (7033, 16)

In [262]:
```
#Checking the datatype of its feature and shape
merged_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7033 entries, 0 to 7032
Data columns (total 16 columns):
customerID              7033 non-null object
Region                  7033 non-null object
Gender                  7033 non-null object
Partner                 7033 non-null int64
Dependents              7033 non-null int64
Tenure                  7033 non-null int64
CustomerServiceCalls    7033 non-null int64
PhoneService            7033 non-null object
InternetService         7033 non-null object
OnlineSecurity          7033 non-null int64
StreamingTV             7033 non-null int64
LockedIn                7033 non-null int64
PaperlessBilling        7033 non-null int64
DominantPaymentMethod   7033 non-null object
MonthlyCharges          7033 non-null int64
Churn                   7033 non-null int64
dtypes: int64(10), object(6)
memory usage: 934.1+ KB
```

In [263]:
```
#Checking the descriptive statistics of the dataset
merged_dataset.describe(include="all")
```

Out[263]:

	customerID	Region	Gender	Partner	Dependents	Tenure	CustomerServiceCalls	Phoı
count	7033	7033	7033	7033.00	7033.00	7033.00	7033.00	
unique	7033	10	4	nan	nan	nan	nan	
top	4518-FZBSX	NCR	Male	nan	nan	nan	nan	
freq	1	2219	3534	nan	nan	nan	nan	
mean	NaN	NaN	NaN	0.48	0.30	33.09	1.35	
std	NaN	NaN	NaN	0.50	0.46	63.02	1.36	
min	NaN	NaN	NaN	0.00	0.00	0.00	0.00	
25%	NaN	NaN	NaN	0.00	0.00	9.00	1.00	
50%	NaN	NaN	NaN	0.00	0.00	29.00	1.00	
75%	NaN	NaN	NaN	1.00	1.00	55.00	2.00	
max	NaN	NaN	NaN	1.00	1.00	4900.00	7.00	

Renaming Column Names

Renaming columns or features is important in a ways that datasets be properly named and for convenience in accessing those features. Best naming format to be followed has the first letters of the words capitalize and separated using an underscore(_). Features that are improperly named are:

- customerID
- PhoneService
- InternetService
- OnlineSecurity
- StreamingTV
- LockedIn
- PaperlessBilling
- DominantPaymentMethod
- MonthlyCharges

In [264]:
```
#Renaming the columns and checking the head of the dataset if it works.
merged_dataset.rename(columns={'customerID':'Customer_ID',
                               'CustomerServiceCalls':'Customer_Service_Calls',
                               'PhoneService':'Phone_Service',
                               'InternetService':'Internet_Service',
                               'OnlineSecurity':'Online_Security',
                               'StreamingTV':'Streaming_TV',
                               'LockedIn':'Locked_In',
                               'PaperlessBilling':'Paperless_Billing',
                               'DominantPaymentMethod':'Dominant_Payment_Method',
                               'MonthlyCharges':'Monthly_Charges',},
                               inplace=True)
merged_dataset.head()
```

Out[264]:

	Customer_ID	Region	Gender	Partner	Dependents	Tenure	Customer_Service_Calls	Phon
0	0002-ORFBO	North Luzon	Female	1	1	9	2	
1	0003-MKNFE	South Luzon	Male	0	0	9	1	
2	0004-TLHLJ	South Luzon	Male	0	0	4	0	
3	0011-IGKFF	NCR	Male	1	0	13	1	
4	0013-EXCHZ	Mindanao	Female	1	0	3	2	

Splitting Categorical and Numerical Features

To be able to clean the data properly and for easier convenience, one of the steps was to separate the categorical and numerical features.

In [265]:
```
#Categorical Features Only
df_cat = merged_dataset.select_dtypes(include=['object'])

#Numerical Features Only
df_num = merged_dataset.select_dtypes(include=['int64','float64'])
```

Categorical Features

Steps to be done in cleaning the categorical features:

- Correcting misspelled data
- Checking counts of each feature

In [266]:

```
#Setting index to Customer_ID
df_cat.set_index('Customer_ID')
```

Out[266]:

Customer_ID	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Method
0002-ORFBO	North Luzon	Female	Yes	DSL	In-person
0003-MKNFE	South Luzon	Male	Yes	DSL	In-person
0004-TLHLJ	South Luzon	Male	Yes	Fiber	In-person
0011-IGKFF	NCR	Male	Yes	Fiber	In-person
0013-EXCHZ	Mindanao	Female	Yes	Fiber	In-person
0013-MHZWF	NCR	Female	Yes	DSL	Credit card
0013-SMEOE	North Luzon	Female	Yes	Fiber	Bank transfer
0014-BMAQU	North Luzon	Male	Yes	Fiber	Credit card
0015-UOCOJ	North Luzon	Female	Yes	DSL	In-person
0016-QLJIS	Visayas	Female	Yes	DSL	In-person
0017-DINOC	North Luzon	Male	Yes	DSL	Credit card
0017-IUDMW	Visayas	Female	Yes	Fiber	Credit card
0018-NYROU	Visayas	Female	Yes	Fiber	In-person
0019-EFAEP	North Luzon	Female	Yes	Fiber	Bank transfer
0019-GFNTW	Visayas	Female	Yes	DSL	Bank transfer
0020-INWCK	North Luzon	male	Yes	Fiber	Credit card
0020-JDNXP	NCR	Female	Yes	DSL	Credit card
0021-IKXGC	NCR	Female	Yes	Fiber	In-person
0022-TCJCI	North Luzon	Male	Yes	DSL	Credit card
0023-HGHWL	Visayas	Male	Yes	DSL	In-person
0023-UYUPN	North Luzon	Female	Yes	No	In-person
0023-XUOPT	Nor. Luz.	Female	Yes	Fiber	In-person
0027-KWYKW	Mindanao	Female	Yes	Fiber	In-person
0030-FNXPP	NCR	Female	Yes	No	In-person
0031-PVLZI	NCR	Female	Yes	No	In-person
0032-PGELS	North Luzon	Female	Yes	DSL	Bank transfer
0036-IHMOT	NCR	Female	Yes	Fiber	Bank transfer
0040-HALCW	NCR	Male	Yes	No	Credit card
0042-JVWOJ	NCR	Male	Yes	No	Credit card
0042-RLHYP	NCR	Female	Yes	No	Bank transfer
0048-LUMLS	NCR	Male	Yes	Fiber	Credit card
0048-PIHNL	Nor. Luz.	Female	Yes	No	Bank transfer
0052-DCKON	Visayas	Male	Yes	Fiber	Bank transfer
0052-YNYOT	North Luzon	Female	Yes	No	Bank transfer
0056-EPFBG	Visayas	Male	Yes	DSL	Credit card
0057-QBUQH	NCR	Female	Yes	No	Credit card
0058-EVZWM	North Luzon	Female	Yes	Fiber	Bank transfer
0060-FUALY	Visayas	Female	Yes	Fiber	In-person
0064-SUDOG	Nor. Luz.	Female	Yes	No	Bank transfer
0064-YIJGF	South Luzon	Male	Yes	Fiber	Bank transfer

Customer_ID	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Method
0067-DKWBL	NCR	Male	Yes	No	In-person
0068-FIGTF	NCR	Female	Yes	DSL	Credit card
0071-NDAFP	NCR	Male	Yes	No	Credit card
0074-HDKDG	NCR	Male	Yes	DSL	Bank transfer
0076-LVEPS	North Luzon	Male	Yes	DSL	Bank transfer
0078-XZMHT	NCR	Male	Yes	DSL	Bank transfer
0080-EMYVY	NCR	Female	Yes	DSL	Credit card
0080-OROZO	NCR	Female	Yes	Fiber	In-person
0082-LDZUE	NCR	Male	Yes	DSL	Credit card
0082-OQIQY	Visayas	Male	Yes	Fiber	In-person
...	...	...	...	...	...
9924-JPRMC	NCR	Male	Yes	Fiber	Credit card
9926-PJHDQ	NCR	Female	Yes	DSL	Bank transfer
9927-DSWDF	Visayas	Male	Yes	Fiber	In-person
9928-BZVLZ	Nor. Luz.	Female	Yes	DSL	Bank transfer
9929-PLVPA	North Luzon	Female	Yes	No	Credit card
9931-DCEZH	South Luzon	Male	Yes	DSL	Credit card
9931-KGHOA	NCR	Female	Yes	DSL	Bank transfer
9932-WBWIK	Mindanao	Male	Yes	No	In-person
9933-QRGTX	NCR	Female	Yes	Fiber	Credit card
9938-EKRGF	South Luzon	Female	Yes	DSL	In-person
9938-PRCVK	NCR	Female	Yes	No	Bank transfer
9938-TKDGL	Visayas	Male	Yes	Fiber	In-person
9938-ZREHM	Visayas	Female	Yes	DSL	In-person
9940-HPQPG	North Luzon	Female	Yes	Fiber	Bank transfer
9940-RHLFB	Vis.	Female	Yes	Fiber	In-person
9943-VSZUV	NCR	Male	Yes	Fiber	Credit card
9944-AEXBM	NCR	Male	Yes	Fiber	Bank transfer
9944-HKVVB	NCR	Female	Yes	No	In-person
9945-PSVIP	North Luzon	Female	Yes	No	Bank transfer
9947-OTFQU	NCR	Male	Yes	Fiber	In-person
9948-YPTDG	NCR	Male	Yes	No	In-person
9950-MTGYX	Visayas	Male	Yes	No	Credit card
9953-ZMKSM	North Luzon	Male	Yes	No	In-person
9955-QOPOY	Visayas	Male	Yes	DSL	Credit card
9957-YODKZ	NCR	Male	Yes	Fiber	In-person
9958-MEKUC	Mindanao	Male	Yes	Fiber	Credit card.
9959-WOFKT	National Capital Region	Male	Yes	Fiber	Bank transfer
9961-JBNMK	North Luzon	Male	Yes	Fiber	Bank transfer
9962-BFPDU	NCR	Female	Yes	No	Credit card
9964-WBQDJ	NCR	Female	Yes	No	Credit card
9965-YOKZB	Visayas	Male	Yes	Fiber	In-person

Customer_ID	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Method
9967-ATRFS	NCR	Female	Yes	No	In-person
9968-FFVVH	NCR	Male	Yes	DSL	Bank transfer
9970-QBCDA	Mindanao	Female	Yes	No	Credit card
9971-ZWPBF	NCR	Male	Yes	Fiber	Credit card
9972-EWRJS	NCR	Female	Yes	No	Bank transfer
9972-NKTFD	Visayas	Female	Yes	DSL	Bank transfer
9972-VAFJJ	North Luzon	Female	Yes	Fiber	Bank transfer
9974-JFBHQ	Visayas	Male	Yes	Fiber	Credit card
9975-GPKZU	Visayas	Male	Yes	No	Credit card
9975-SKRNR	Visayas	Male	Yes	No	In-person
9978-HYCIN	NCR	Male	Yes	Fiber	Bank transfer
9979-RGMZT	Nor. Luz.	Female	Yes	Fiber	Bank transfer
9985-MWVIX	Mindanao	Female	Yes	Fiber	Credit card
9986-BONCE	South Luzon	Female	Yes	No	Bank transfer
9987-LUTYD	South Luzon	Female	Yes	DSL	In-person
9992-RRAMN	North Luzon	Male	Yes	Fiber	In-person
9992-UJOEL	NCR	Male	Yes	DSL	In-person
9993-LHIEB	North Luzon	Male	Yes	DSL	In-person
9995-HOTOH	NCR	Male	Yes	DSL	In-person

7033 rows × 5 columns

In [267]:
```
#Checking the head of the dataset
df_cat.head()
```

Out[267]:

	Customer_ID	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Method
0	0002-ORFBO	North Luzon	Female	Yes	DSL	In-person
1	0003-MKNFE	South Luzon	Male	Yes	DSL	In-person
2	0004-TLHLJ	South Luzon	Male	Yes	Fiber	In-person
3	0011-IGKFF	NCR	Male	Yes	Fiber	In-person
4	0013-EXCHZ	Mindanao	Female	Yes	Fiber	In-person

In [268]:
```
#Checking the shape
df_cat.shape
```

Out[268]: (7033, 6)

In [269]:
```python
#Checking the datatypes
df_cat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7033 entries, 0 to 7032
Data columns (total 6 columns):
Customer_ID                7033 non-null object
Region                     7033 non-null object
Gender                     7033 non-null object
Phone_Service              7033 non-null object
Internet_Service           7033 non-null object
Dominant_Payment_Method    7033 non-null object
dtypes: object(6)
memory usage: 384.6+ KB
```

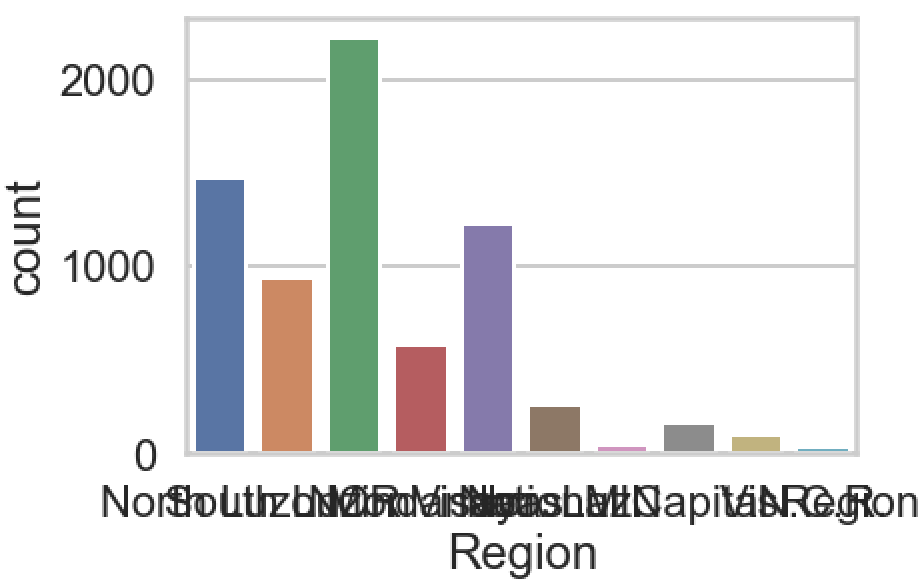
In [270]:
```python
#Checking the descriptive statistics for categorical
df_cat.describe(include='object')
```

Out[270]:

	Customer_ID	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Meth
count	7033	7033	7033	7033	7033	70
unique	7033	10	4	1	3	
top	4518-FZBSX	NCR	Male	Yes	Fiber	In-pers
freq	1	2219	3534	7033	2894	33

In [271]:
```python
#Countplot for Region
sns.countplot(df_cat['Region'])
```

Out[271]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22fcf965e10>
```



In [272]:
```python
#Checking the records in features
df_cat.Region.value_counts()
```

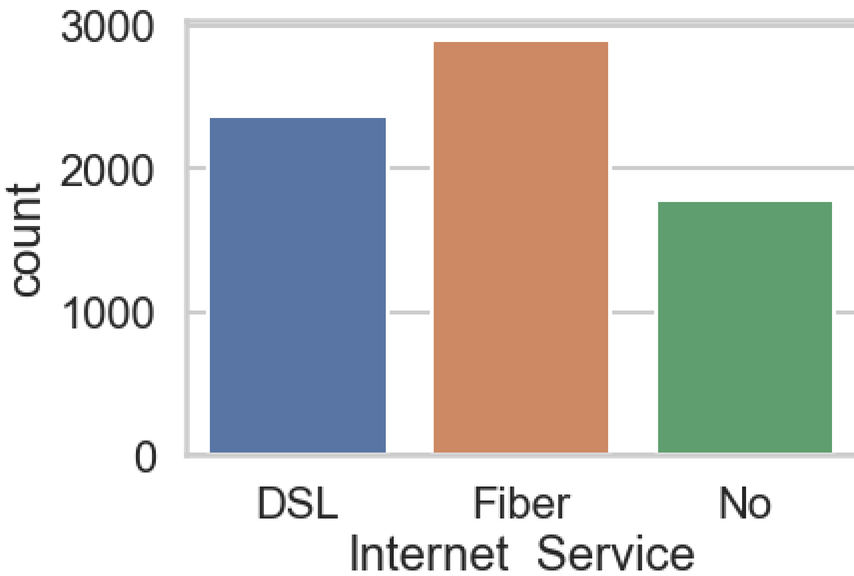
Out[272]:
```
NCR                       2219
North Luzon               1469
Visayas                   1224
South Luzon                931
Mindanao                   585
Nor. Luz.                  264
National Capital Region    160
Vis.                       104
MIN                         48
N.C.R.                      29
Name: Region, dtype: int64
```

It must be taken account that there are redundant values that are incorrectly named such as:

- Nor. Luz.
- MIN
- N.C.R.
- National Capital Region
- Vis.

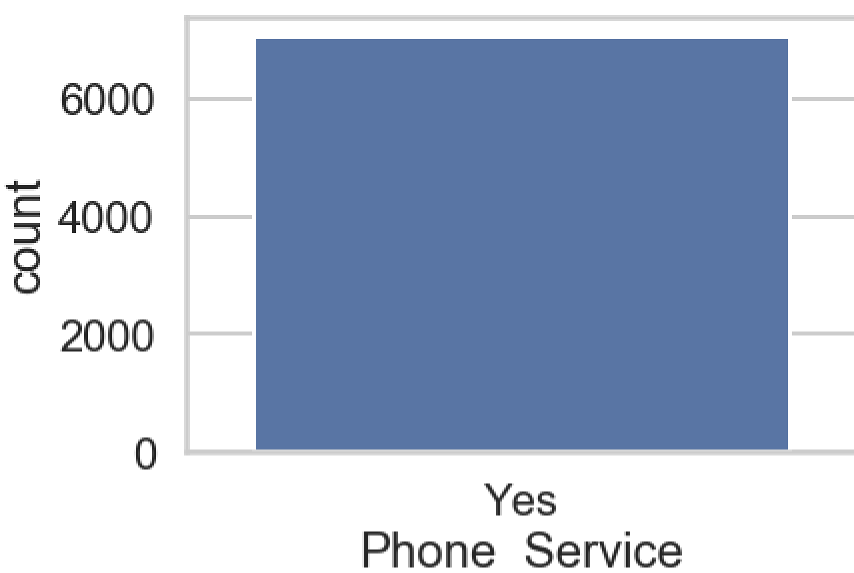
In [273]:
```
sns.countplot(df_cat['Internet_Service'])
```

Out[273]: <matplotlib.axes._subplots.AxesSubplot at 0x22fcf962940>



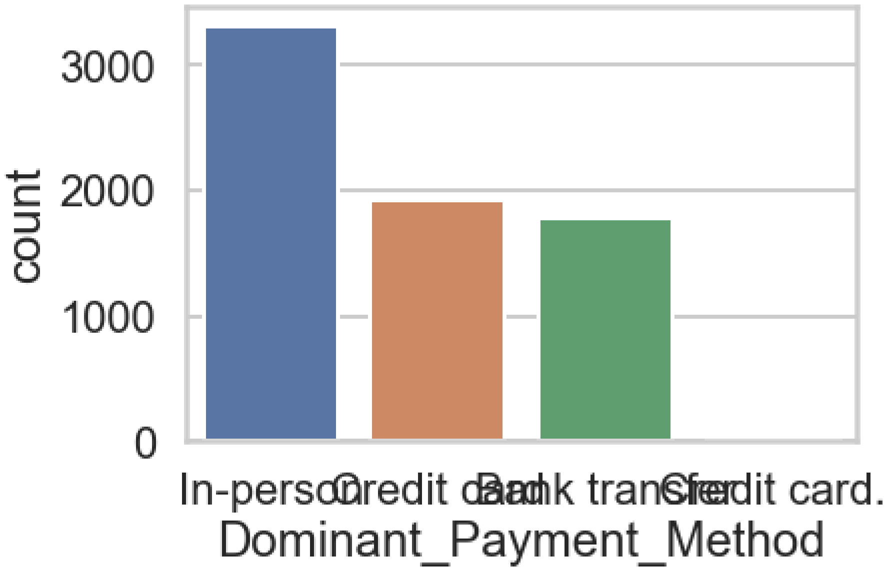
In [274]:
```
sns.countplot(df_cat['Phone_Service'])
#Knowing everyone has phone service, it can be dropped
```

Out[274]: <matplotlib.axes._subplots.AxesSubplot at 0x22fcfa7bcf8>



In [275]:
```
#Countplot for Dominant Payment Method
sns.countplot(df_cat['Dominant_Payment_Method'])
```

Out[275]: <matplotlib.axes._subplots.AxesSubplot at 0x22fcfa56ac8>



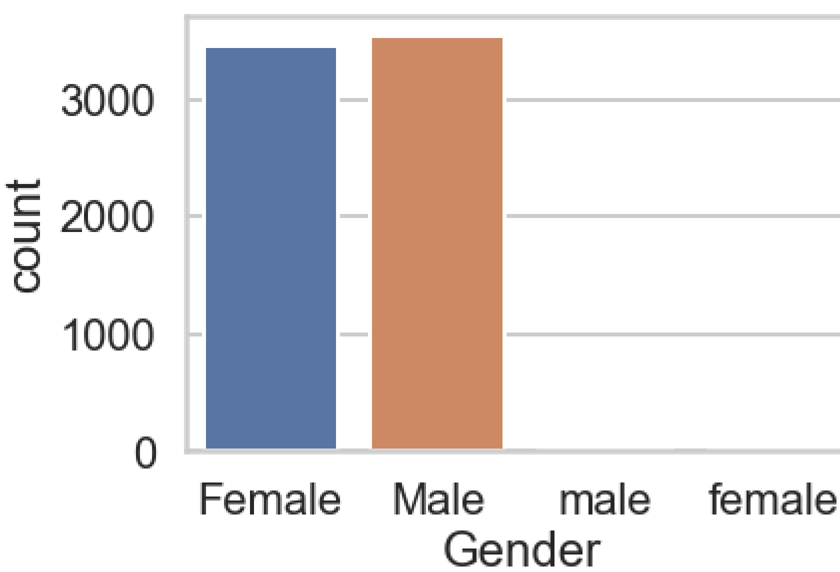
In [276]:
```
#Checking the vlues under the Dominant Payment Method Features
df_cat.Dominant_Payment_Method.value_counts()
```

Out[276]:
```
In-person        3306
Credit card      1915
Bank transfer    1784
Credit card.       28
Name: Dominant_Payment_Method, dtype: int64
```

It must be noted that the Credit card. is redundant here and must be changed

In [277]:
```
sns.countplot(df_cat['Gender'])
```

Out[277]: <matplotlib.axes._subplots.AxesSubplot at 0x22fcf9a83c8>



In [278]:
```
#Checking the vlues under the Gender Features
df_cat.Gender.value_counts()
```

Out[278]:
```
Male      3534
Female    3449
male        39
female      11
Name: Gender, dtype: int64
```

Same goes with the Gender where there are male and female inputs that must be renamed to merge with the proper named values.

In [279]:

```
#Another way to check the values of each columns all at once.
for cat_col in df_cat.columns:
    print (df_cat[cat_col].value_counts())
    print ("\n---------")
```

4518-FZBSX	1
8815-LMFLX	1
4710-FDUIZ	1
2656-FMOKZ	1
9921-QFQUL	1
5781-BKHOP	1
2829-HYVZP	1
4074-SJFFA	1
2535-PBCGC	1
6178-KFNHS	1
7013-PSXHK	1
5288-AHOUP	1
6164-HAQTX	1
8747-UDCOI	1
1080-BWSYE	1
4785-QRJHC	1
0917-EZOLA	1
1456-TWCGB	1
5915-ANOEI	1
3671-SHRSP	1
1764-VUUMT	1
8805-JNRAZ	1
6500-JVEGC	1
3756-VNWDH	1
3585-ISXZP	1
7409-KIUTL	1
0661-KBKPA	1
6179-GJPSO	1
0362-ZBZWJ	1
1309-BXVOQ	1
1202-KKGFU	1
1374-DMZUI	1
9061-TIHDA	1
2929-ERCFZ	1
8328-SKJNO	1
1837-YQUCE	1
8063-RJYNF	1
0666-UXTJO	1
4927-WWOOZ	1
8402-EIVQS	1
5480-XTFFL	1
3511-APPBJ	1
5249-QYHEX	1
7733-UDMTP	1
1905-OEILC	1
0516-UXRMT	1
3629-WEAAM	1
4220-TINQT	1
6198-PNNSZ	1
9274-CNFMO	1
	..
7359-SSBJK	1
5996-DAOQL	1
0853-TWRVK	1
2777-PHDEI	1
3859-CVCET	1
1697-NVVGY	1
2873-ZLIWT	1
6806-YDEUL	1
8327-WKMIE	1
2004-OCQXK	1
9618-LFJRU	1
3727-JEZTU	1
6877-LGWXO	1
3066-RRJIO	1
3255-GRXMG	1
0259-GBZSH	1
7159-FVYPK	1
6583-SZVGP	1
3707-GNWHM	1
5820-PTRYM	1
8590-OHDIW	1
0959-WHOKV	1
3190-FZATL	1

```
3567-PQTSO    1
0939-EREMR    1
8838-GPHZP    1
1293-BSEUN    1
7634-WSWDB    1
0607-DAAHE    1
9838-BFCQT    1
8957-THMOA    1
8015-IHCGW    1
1298-PHBTI    1
5089-IFSDP    1
0302-JOIVN    1
1177-XZBJL    1
4971-PUYQO    1
0744-BIKKF    1
7446-KQISO    1
2589-AYCRP    1
1480-BKXGA    1
2434-EEVDB    1
8085-MSNLK    1
5384-ZTTWP    1
0820-FNRNX    1
6369-MCAKO    1
0795-LAFGP    1
2282-YGNOR    1
0415-MOSGF    1
2961-VNFKL    1
Name: Customer_ID, Length: 7033, dtype: int64

---------
NCR                         2219
North Luzon                 1469
Visayas                     1224
South Luzon                  931
Mindanao                     585
Nor. Luz.                    264
National Capital Region      160
Vis.                         104
MIN                           48
N.C.R.                        29
Name: Region, dtype: int64

---------
Male      3534
Female    3449
male        39
female      11
Name: Gender, dtype: int64

---------
Yes    7033
Name: Phone_Service, dtype: int64

---------
Fiber    2894
DSL      2359
No       1780
Name: Internet_Service, dtype: int64

---------
In-person        3306
Credit card      1915
Bank transfer    1784
Credit card.       28
Name: Dominant_Payment_Method, dtype: int64

---------
```

Need to Take Note for Renaming Values:

- For Region - Nor. Luzon (North Luzon); N.C.R., NCR (National Capital Region); Vis (Visayas); MIN (Mindanao)
- For Gender - male 39 and female 11
- For Dominant_Pay_Method - Credit card 1915 and Credit card. 28

Renaming Values that were spotted together

In [280]:
```
#Renaming values to retain consistency

df_cat['Region'] = df_cat['Region'].replace({'Nor. Luz.': 'North Luzon',
                                             'NCR': 'National Capital Region',
                                             'N.C.R.': 'National Capital Regio
n',
                                             'Vis.': 'Visayas',
                                             'MIN': 'Mindanao'})

df_cat['Gender'] = df_cat['Gender'].replace({'male': 'Male',
                                             'female': 'Female'})

df_cat['Dominant_Payment_Method'] = df_cat['Dominant_Payment_Method'].replace
({'Credit card.': 'Credit Card',

'Credit card': 'Credit Card'})
```

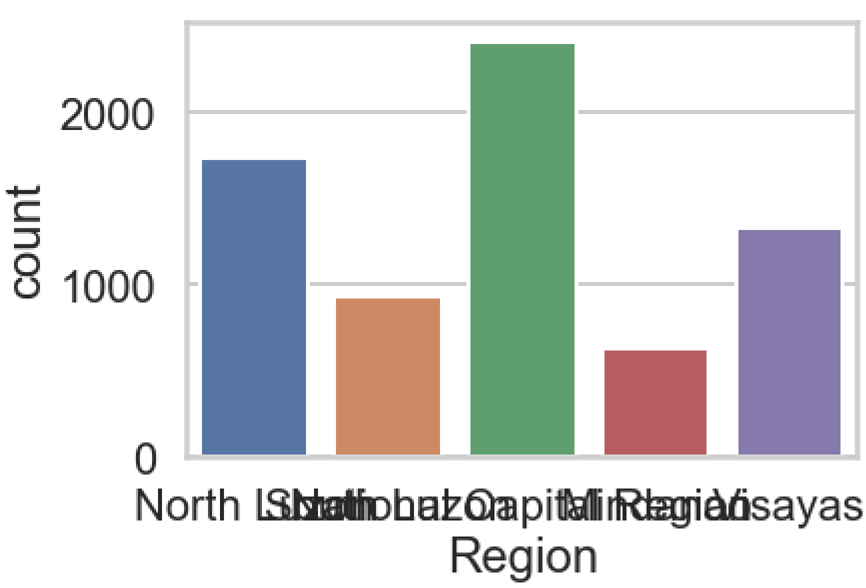
Checking the renamed values

In [281]:
```
df_cat['Region'].value_counts()
```

Out[281]:
```
National Capital Region    2408
North Luzon                1733
Visayas                    1328
South Luzon                 931
Mindanao                    633
Name: Region, dtype: int64
```

In [282]:
```
# For categorical variables, you can use a countplot
sns.countplot(df_cat['Region'])
```

Out[282]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22fcf947898>
```

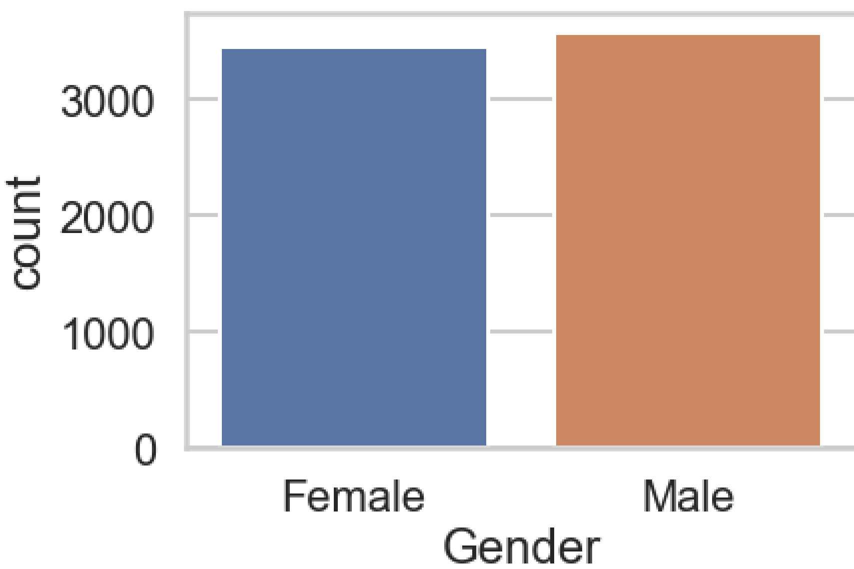


In [283]:
```python
df_cat['Gender'].value_counts()
```

Out[283]:
```
Male      3573
Female    3460
Name: Gender, dtype: int64
```

In [284]:
```python
# For categorical variables, you can use a countplot
sns.countplot(df_cat['Gender'])
```

Out[284]: `<matplotlib.axes._subplots.AxesSubplot at 0x22fcf84de80>`

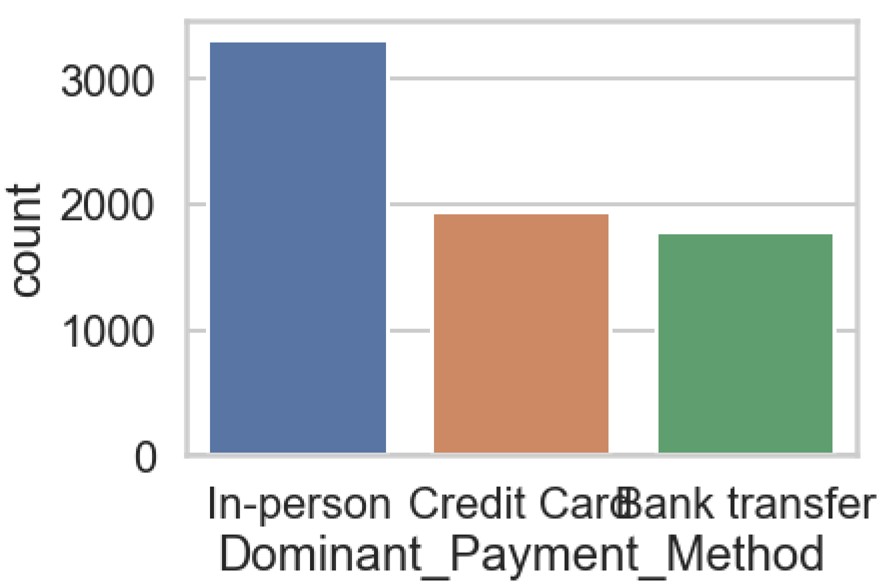


In [285]:
```python
df_cat['Dominant_Payment_Method'].value_counts()
```

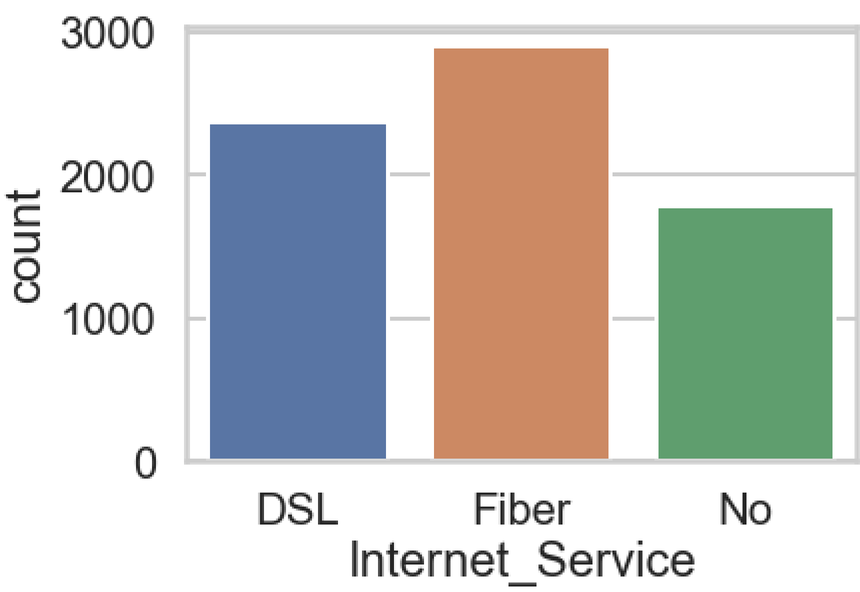
Out[285]:
```
In-person        3306
Credit Card      1943
Bank transfer    1784
Name: Dominant_Payment_Method, dtype: int64
```

In [286]:
```python
# For categorical variables, you can use a countplot
sns.countplot(df_cat['Dominant_Payment_Method'])
```

Out[286]: `<matplotlib.axes._subplots.AxesSubplot at 0x22fcf82ae10>`



In [287]:
```
sns.countplot(df_cat['Internet_Service'])
```
Out[287]: <matplotlib.axes._subplots.AxesSubplot at 0x22fcf35da90>



In [288]:
```
#df_cat.Internet_Service_num.value_counts()
#df_cat.head(10)
#Dropping the categorical features
#df_cat.drop(columns=['Region', 'Gender', 'Phone_Service', 'Internet_Service',
'Dominant_Payment_Method'], axis=1)
#df_cat.set_index('Customer_ID')

#df_cat1 = df_cat.copy()
#df_cat1.head()
```

In [289]:
```
#Getting a dummy copy of the dtaset in case of messing up, and dropping the Cu
stomer_ID since it's just a unique identifier for the person and not relevant
 to the data, but storing it for the future use
df_dum1 = df_cat.copy()
df_dum = df_cat.drop('Customer_ID', axis=1)
x_cust = df_cat.Customer_ID
```

In [290]:
```
df_dum1.head()
```
Out[290]:

	Customer_ID	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Method
0	0002-ORFBO	North Luzon	Female	Yes	DSL	In-person
1	0003-MKNFE	South Luzon	Male	Yes	DSL	In-person
2	0004-TLHLJ	South Luzon	Male	Yes	Fiber	In-person
3	0011-IGKFF	National Capital Region	Male	Yes	Fiber	In-person
4	0013-EXCHZ	Mindanao	Female	Yes	Fiber	In-person

In [291]:
```
df_dum.head()
```
Out[291]:

	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Method
0	North Luzon	Female	Yes	DSL	In-person
1	South Luzon	Male	Yes	DSL	In-person
2	South Luzon	Male	Yes	Fiber	In-person
3	National Capital Region	Male	Yes	Fiber	In-person
4	Mindanao	Female	Yes	Fiber	In-person

Numerical Features

For numerical features, we need to convert these features into one-hut encoding (which means columns have to contain values 1 and 0 to be machine readable).

This part uses one of pandas modules named Get_dummies which reads each festures and breaks it down to certain columns with specific input.

In [292]:
```
#Initial Checking of the Dataset
df_num.head()
```

Out[292]:

	Partner	Dependents	Tenure	Customer_Service_Calls	Online_Security	Streaming_TV	Locked
0	1	1	9	2	0	1	
1	0	0	9	1	0	0	
2	0	0	4	0	0	0	
3	1	0	13	1	0	1	
4	1	0	3	2	0	1	

In [293]:
```
#checking the shap and number of numerical columns
df_num.shape
```

Out[293]: (7033, 10)

In [294]:
```
#Checking the descriptive statistics
df_num.describe()
```

Out[294]:

	Partner	Dependents	Tenure	Customer_Service_Calls	Online_Security	Streaming_TV	L
count	7033.00	7033.00	7033.00	7033.00	7033.00	7033.00	
mean	0.48	0.30	33.09	1.35	0.41	0.42	
std	0.50	0.46	63.02	1.36	0.49	0.49	
min	0.00	0.00	0.00	0.00	0.00	0.00	
25%	0.00	0.00	9.00	1.00	0.00	0.00	
50%	0.00	0.00	29.00	1.00	0.00	0.00	
75%	1.00	1.00	55.00	2.00	1.00	1.00	
max	1.00	1.00	4900.00	7.00	1.00	1.00	

In [295]:
```
df_converted1 = pd.get_dummies(df_dum)
```

In [296]:
```
df_converted1.head()
```

Out[296]:

	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas	Gender_Fe
0	0	0	1	0	0	
1	0	0	0	1	0	
2	0	0	0	1	0	
3	0	1	0	0	0	
4	1	0	0	0	0	

In [297]:
```
#df_cat1 = x_cust.merge(df_converted1, left_on='customerID', right_on='custome
rID')
#df_converted1.insert(0, 'Customer_ID', x_cust)

#df_converted1.head()
#df_converted1.set_index('Customer_ID')
```

Merging Into Final Dataset (Both Categorical and Numerical Features)

In [298]:
```
#df_z = pd.concat([merged_dataset, df_converted1], axis =1, join='inner')
```

In [299]:
```
#Using the concat function to merge the datasets
df = pd.concat([x_cust, df_converted1, df_num], axis =1, join='inner')
```

In [300]:
```
df.head()
```

Out[300]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [301]:
```
df.dtypes
```

Out[301]:
```
Customer_ID                                object
Region_Mindanao                             uint8
Region_National Capital Region              uint8
Region_North Luzon                          uint8
Region_South Luzon                          uint8
Region_Visayas                              uint8
Gender_Female                               uint8
Gender_Male                                 uint8
Phone_Service_Yes                           uint8
Internet_Service_DSL                        uint8
Internet_Service_Fiber                      uint8
Internet_Service_No                         uint8
Dominant_Payment_Method_Bank transfer       uint8
Dominant_Payment_Method_Credit Card         uint8
Dominant_Payment_Method_In-person           uint8
Partner                                     int64
Dependents                                  int64
Tenure                                      int64
Customer_Service_Calls                      int64
Online_Security                             int64
Streaming_TV                                int64
Locked_In                                   int64
Paperless_Billing                           int64
Monthly_Charges                             int64
Churn                                       int64
dtype: object
```

In [302]:
```
#Dropping the categorical features that are not needed anymore
#df_z.drop(columns=['Region', 'Gender', 'Phone_Service', 'Internet_Service','D
ominant_Payment_Method'], inplace = True, axis=0 )
```

In [303]:

```
df.head()
df.set_index('Customer_ID')
```

Out[303]:

Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas
0002-ORFBO	0	0	1	0	0
0003-MKNFE	0	0	0	1	0
0004-TLHLJ	0	0	0	1	0
0011-IGKFF	0	1	0	0	0
0013-EXCHZ	1	0	0	0	0
0013-MHZWF	0	1	0	0	0
0013-SMEOE	0	0	1	0	0
0014-BMAQU	0	0	1	0	0
0015-UOCOJ	0	0	1	0	0
0016-QLJIS	0	0	0	0	1
0017-DINOC	0	0	1	0	0
0017-IUDMW	0	0	0	0	1
0018-NYROU	0	0	0	0	1
0019-EFAEP	0	0	1	0	0
0019-GFNTW	0	0	0	0	1
0020-INWCK	0	0	1	0	0
0020-JDNXP	0	1	0	0	0
0021-IKXGC	0	1	0	0	0
0022-TCJCI	0	0	1	0	0
0023-HGHWL	0	0	0	0	1
0023-UYUPN	0	0	1	0	0
0023-XUOPT	0	0	1	0	0
0027-KWYKW	1	0	0	0	0
0030-FNXPP	0	1	0	0	0
0031-PVLZI	0	1	0	0	0
0032-PGELS	0	0	1	0	0
0036-IHMOT	0	1	0	0	0
0040-HALCW	0	1	0	0	0
0042-JVWOJ	0	1	0	0	0
0042-RLHYP	0	1	0	0	0
0048-LUMLS	0	1	0	0	0
0048-PIHNL	0	0	1	0	0
0052-DCKON	0	0	0	0	1
0052-YNYOT	0	0	1	0	0
0056-EPFBG	0	0	0	0	1
0057-QBUQH	0	1	0	0	0
0058-EVZWM	0	0	1	0	0
0060-FUALY	0	0	0	0	1
0064-SUDOG	0	0	1	0	0
0064-YIJGF	0	0	0	1	0
0067-DKWBL	0	1	0	0	0

Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas
0068-FIGTF	0	1	0	0	0
0071-NDAFP	0	1	0	0	0
0074-HDKDG	0	1	0	0	0
0076-LVEPS	0	0	1	0	0
0078-XZMHT	0	1	0	0	0
0080-EMYVY	0	1	0	0	0
0080-OROZO	0	1	0	0	0
0082-LDZUE	0	1	0	0	0
0082-OQIQY	0	0	0	0	1
...	...	...	...	...	...
9924-JPRMC	0	1	0	0	0
9926-PJHDQ	0	1	0	0	0
9927-DSWDF	0	0	0	0	1
9928-BZVLZ	0	0	1	0	0
9929-PLVPA	0	0	1	0	0
9931-DCEZH	0	0	0	1	0
9931-KGHOA	0	1	0	0	0
9932-WBWIK	1	0	0	0	0
9933-QRGTX	0	1	0	0	0
9938-EKRGF	0	0	0	1	0
9938-PRCVK	0	1	0	0	0
9938-TKDGL	0	0	0	0	1
9938-ZREHM	0	0	0	0	1
9940-HPQPG	0	0	1	0	0
9940-RHLFB	0	0	0	0	1
9943-VSZUV	0	1	0	0	0
9944-AEXBM	0	1	0	0	0
9944-HKVVB	0	1	0	0	0
9945-PSVIP	0	0	1	0	0
9947-OTFQU	0	1	0	0	0
9948-YPTDG	0	1	0	0	0
9950-MTGYX	0	0	0	0	1
9953-ZMKSM	0	0	1	0	0
9955-QOPOY	0	0	0	0	1
9957-YODKZ	0	1	0	0	0
9958-MEKUC	1	0	0	0	0
9959-WOFKT	0	1	0	0	0
9961-JBNMK	0	0	1	0	0
9962-BFPDU	0	1	0	0	0
9964-WBQDJ	0	1	0	0	0
9965-YOKZB	0	0	0	0	1
9967-ATRFS	0	1	0	0	0
9968-FFVVH	0	1	0	0	0
9970-QBCDA	1	0	0	0	0
9971-ZWPBF	0	1	0	0	0

Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas
9972-EWRJS	0	1	0	0	0
9972-NKTFD	0	0	0	0	1
9972-VAFJJ	0	0	1	0	0
9974-JFBHQ	0	0	0	0	1
9975-GPKZU	0	0	0	0	1
9975-SKRNR	0	0	0	0	1
9978-HYCIN	0	1	0	0	0
9979-RGMZT	0	0	1	0	0
9985-MWVIX	1	0	0	0	0
9986-BONCE	0	0	0	1	0
9987-LUTYD	0	0	0	1	0
9992-RRAMN	0	0	1	0	0
9992-UJOEL	0	1	0	0	0
9993-LHIEB	0	0	1	0	0
9995-HOTOH	0	1	0	0	0

7033 rows × 24 columns

In [304]:
```
#Separating Monthly Charges
h = df["Monthly_Charges"]
```

In [305]:
```
df.dtypes
```

Out[305]:
```
Customer_ID                              object
Region_Mindanao                           uint8
Region_National Capital Region            uint8
Region_North Luzon                        uint8
Region_South Luzon                        uint8
Region_Visayas                            uint8
Gender_Female                             uint8
Gender_Male                               uint8
Phone_Service_Yes                         uint8
Internet_Service_DSL                      uint8
Internet_Service_Fiber                    uint8
Internet_Service_No                       uint8
Dominant_Payment_Method_Bank transfer     uint8
Dominant_Payment_Method_Credit Card       uint8
Dominant_Payment_Method_In-person         uint8
Partner                                   int64
Dependents                                int64
Tenure                                    int64
Customer_Service_Calls                    int64
Online_Security                           int64
Streaming_TV                              int64
Locked_In                                 int64
Paperless_Billing                         int64
Monthly_Charges                           int64
Churn                                     int64
dtype: object
```

In [306]:
```
df.Customer_Service_Calls.value_counts()
```

Out[306]:
```
1    3251
2    1617
0    1570
6     124
3     122
5     120
7     118
4     111
Name: Customer_Service_Calls, dtype: int64
```

In [307]:
```
df.Churn.value_counts()
```

Out[307]:
```
0    6115
1     918
Name: Churn, dtype: int64
```

In [308]:

```
df.Tenure.value_counts()
```

```
Out[308]: 1     611
          72    362
          2     236
          3     200
          4     174
          71    170
          5     133
          7     131
          8     123
          70    119
          9     119
          12    117
          10    116
          6     110
          13    108
          68    100
          15     99
          67     98
          11     98
          18     97
          69     95
          24     94
          22     90
          66     89
          35     88
          17     87
          23     85
          16     80
          56     80
          64     80
          52     79
          26     79
          25     79
          60     76
          14     76
          65     76
          61     76
          46     74
          19     72
          63     72
          29     72
          27     72
          30     72
          20     71
          62     70
          41     70
          53     70
          32     69
          51     68
          50     68
          54     68
          47     68
          58     67
          37     65
          42     65
          34     65
          57     65
          43     65
          49     65
          31     65
          33     64
          48     64
          40     64
          55     64
          21     63
          45     61
          59     60
          38     59
          28     57
          39     56
          44     51
          36     50
          0      11
```

```
4900      1
Name: Tenure, dtype: int64
```

In [309]:

```
df.Monthly_Charges.value_counts()
```

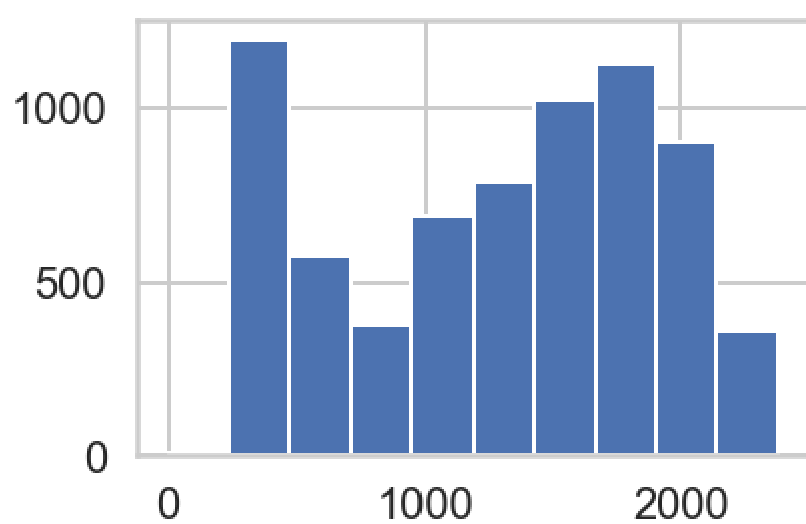
Out[309]:

```
400     424
390     321
410     272
500     131
510      95
1400     93
1490     91
1610     91
490      91
1500     86
1700     82
1600     81
1800     79
420      79
1410     77
380      77
1580     76
1510     76
1710     75
1890     74
1690     74
1900     74
1590     73
1390     70
1810     70
1790     70
1000     65
1620     62
900      62
1090     61
2010     61
2000     60
1100     60
1910     59
1990     58
1780     58
1200     57
1880     57
1010     56
1720     54
1110     53
1380     53
2080     53
910      52
2090     52
2100     51
1480     51
890      50
2110     50
1520     49
       ...
1360     11
870      11
2310     11
930      11
1340     10
1270     10
2330     10
620      10
670      10
2280     10
1550     10
2240     10
680       9
-1        9
470       9
960       9
1250      8
1160      8
1450      8
580       8
1040      7
820       7
1260      7
```

```
1150      6
770       6
1060      6
2340      6
730       6
370       5
2250      5
840       4
2350      4
1050      4
850       4
430       4
2370      4
460       3
570       3
860       2
630       2
540       2
830       2
950       2
2360      2
940       2
2380      1
740       1
660       1
750       1
760       1
Name: Monthly_Charges, Length: 197, dtype: int64
```

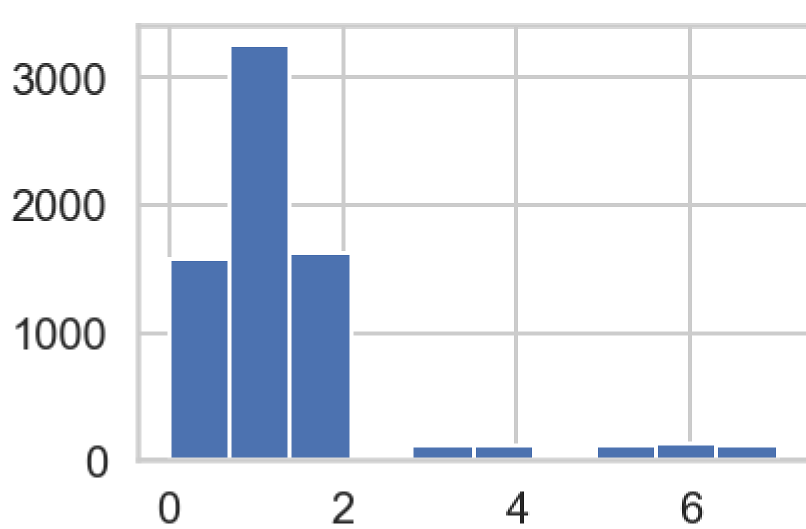
In [310]:
```
df.Monthly_Charges.hist()
```

Out[310]: <matplotlib.axes._subplots.AxesSubplot at 0x22fcfa42b38>



In [311]:
```
df.Customer_Service_Calls.hist()
```

Out[311]: <matplotlib.axes._subplots.AxesSubplot at 0x22fcf801e80>

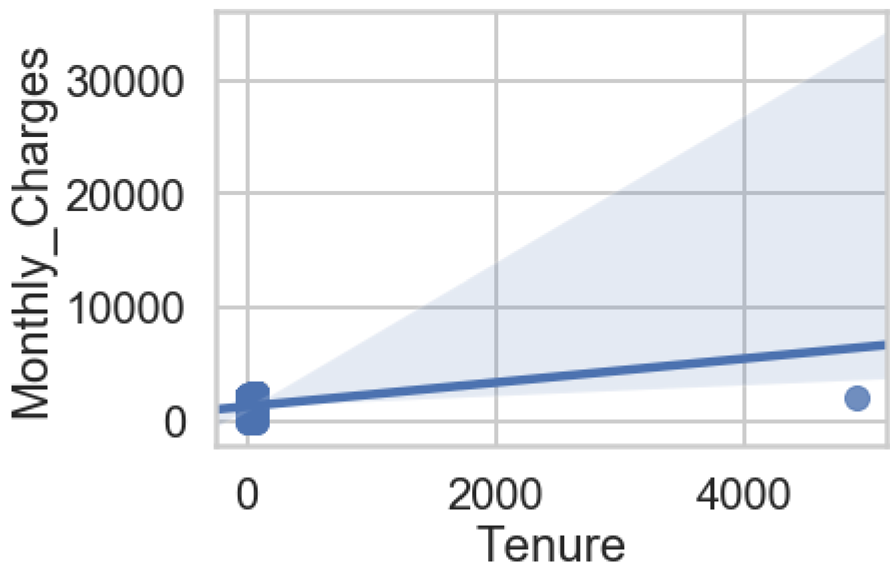


Removing Outliers

Outliers are data points that are out of the usual trend of the data. It may be helpful but one problem is that models have tendency to be sensitive to outliers and greatly affect the result and analysis, so it's advised to be removed.

In [312]:
```
#Visualization of finding the outliers
sns.regplot(data = df, x = 'Tenure', y = 'Monthly_Charges')
```

Out[312]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22fcf7e2ef0>
```



In [313]:
```
#Dropping the outlier
df = df.drop(df[(df.Tenure > 4000) & (df.Monthly_Charges < 5000)].index)
```

In [314]:
```
#Checking if the outlier has been already dropped
df.Tenure.value_counts()
```

Out[314]:

```
1     611
72    362
2     236
3     200
4     174
71    170
5     133
7     131
8     123
70    119
9     119
12    117
10    116
6     110
13    108
68    100
15     99
11     98
67     98
18     97
69     95
24     94
22     90
66     89
35     88
17     87
23     85
64     80
56     80
16     80
52     79
25     79
26     79
14     76
65     76
61     76
60     76
46     74
63     72
29     72
27     72
30     72
19     72
20     71
62     70
41     70
53     70
32     69
47     68
54     68
50     68
51     68
58     67
43     65
42     65
31     65
34     65
37     65
57     65
49     65
55     64
40     64
48     64
33     64
21     63
45     61
59     60
38     59
28     57
39     56
44     51
36     50
0      11
Name: Tenure, dtype: int64
```

In [315]:
```
df.Customer_Service_Calls.value_counts()
```

Out[315]:
```
1    3251
2    1617
0    1570
6     123
3     122
5     120
7     118
4     111
Name: Customer_Service_Calls, dtype: int64
```

In [316]:

```
df.Monthly_Charges.value_counts()
```

```
Out[316]: 400     424
          390     321
          410     272
          500     131
          510      95
          1400     93
          1490     91
          1610     91
          490      91
          1500     86
          1700     82
          1600     81
          1800     79
          420      79
          1410     77
          380      77
          1580     76
          1510     76
          1710     75
          1890     74
          1690     74
          1900     74
          1590     73
          1390     70
          1810     70
          1790     70
          1000     65
          1620     62
          900      62
          1090     61
          2010     61
          2000     60
          1100     60
          1910     59
          1990     58
          1780     58
          1200     57
          1880     57
          1010     56
          1720     54
          1110     53
          1380     53
          2080     53
          910      52
          2090     52
          2100     51
          1480     51
          890      50
          2110     50
          1520     49
                 ...
          1360     11
          870      11
          2310     11
          930      11
          1340     10
          1270     10
          2330     10
          620      10
          670      10
          2280     10
          1550     10
          2240     10
          680       9
         -1         9
          470       9
          960       9
          1250      8
          1160      8
          1450      8
          580       8
          1040      7
          820       7
          1260      7
```

```
1150      6
770       6
1060      6
2340      6
730       6
370       5
2250      5
840       4
2350      4
1050      4
850       4
430       4
2370      4
460       3
570       3
860       2
630       2
540       2
830       2
950       2
2360      2
940       2
2380      1
740       1
660       1
750       1
760       1
Name: Monthly_Charges, Length: 197, dtype: int64
```

In [317]:

```
df.head()
```

Out[317]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [318]:

```
#Checking if it affects the shape
df.shape
```

Out[318]: (7032, 25)

Checking the Correlation between Features through Correlation Map

One of the Exploratory Data Analysis process was to check the correlation of features to our target feature. Correlation refers to the relation between 2 variables (features). This is very helpful in knowing what factors are most relevant or need in a dataset and gives a glimpse on how connected the features in the dataset were.

We check the correlation of the features by calculating the correlation coefficient, which tells the degree of relationship. It ranges from +1.0 to -1.0. The more higher the coefficient, the more positive the correlation between two variables..

In our use case, we have a target feature to be compared to other features in terms of correlation, which is our "Churn" column.

In [319]:

```
#dropping this columns since it is not relevant if we were to check the correl
ation with it.
df.drop(columns=['Customer_ID','Phone_Service_Yes'], axis=1)
```

Out[319]:

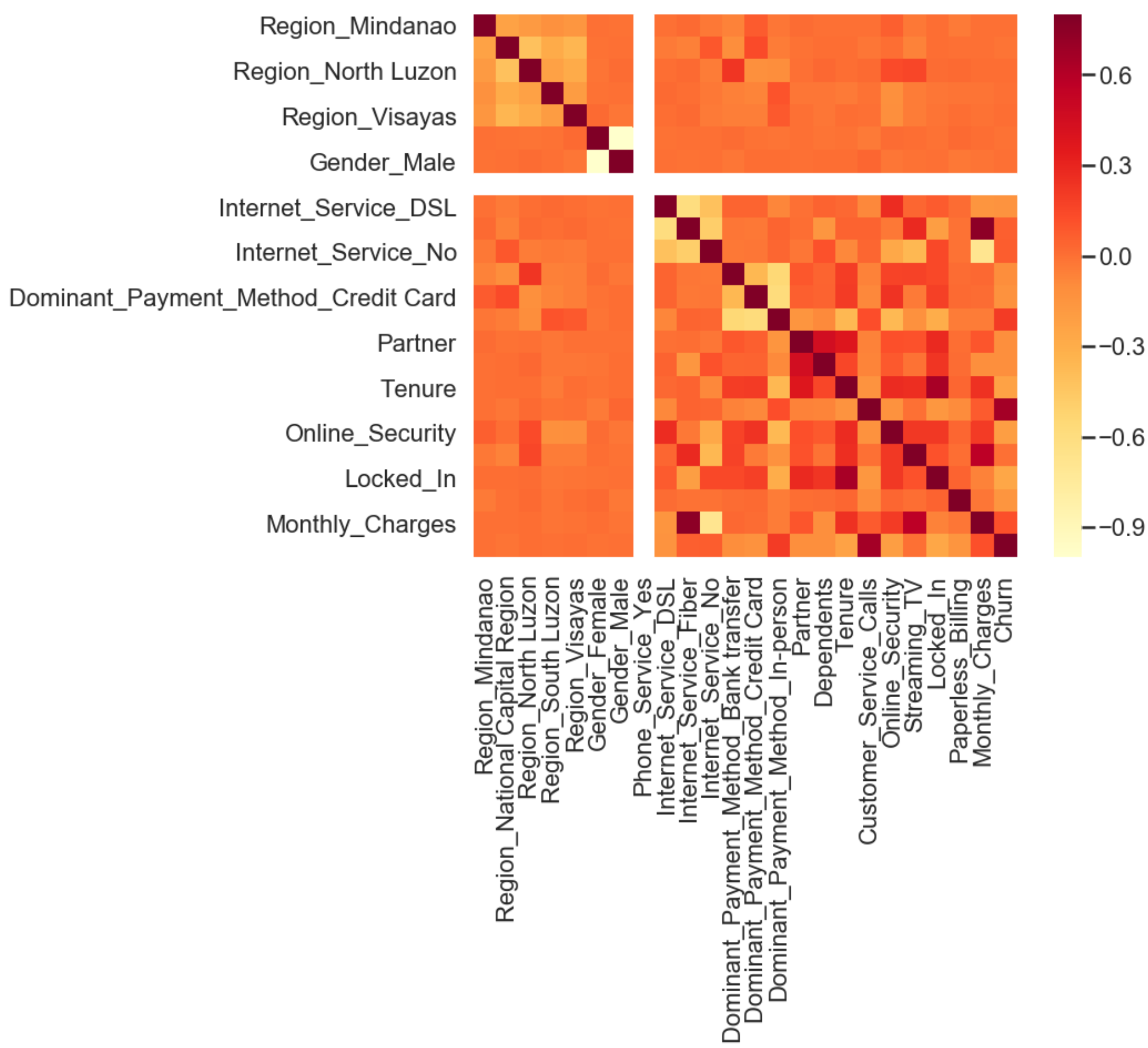
	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas	Gender_
0	0	0	1	0	0	
1	0	0	0	1	0	
2	0	0	0	1	0	
3	0	1	0	0	0	
4	1	0	0	0	0	
5	0	1	0	0	0	
6	0	0	1	0	0	
7	0	0	1	0	0	
8	0	0	1	0	0	
9	0	0	0	0	1	
10	0	0	1	0	0	
11	0	0	0	0	1	
12	0	0	0	0	1	
13	0	0	1	0	0	
14	0	0	0	0	1	
15	0	0	1	0	0	
16	0	1	0	0	0	
17	0	1	0	0	0	
18	0	0	1	0	0	
19	0	0	0	0	1	
20	0	0	1	0	0	
21	0	0	1	0	0	
22	1	0	0	0	0	
23	0	1	0	0	0	
24	0	1	0	0	0	
25	0	0	1	0	0	
26	0	1	0	0	0	
27	0	1	0	0	0	
28	0	1	0	0	0	
29	0	1	0	0	0	
30	0	1	0	0	0	
31	0	0	1	0	0	
32	0	0	0	0	1	
33	0	0	1	0	0	
34	0	0	0	0	1	
35	0	1	0	0	0	
36	0	0	1	0	0	
37	0	0	0	0	1	
38	0	0	1	0	0	
39	0	0	0	1	0	
40	0	1	0	0	0	
41	0	1	0	0	0	
42	0	1	0	0	0	
43	0	1	0	0	0	
44	0	0	1	0	0	
45	0	1	0	0	0	

	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas	Gender_
46	0	1	0	0	0	
47	0	1	0	0	0	
48	0	1	0	0	0	
49	0	0	0	0	1	
...	...	...	...	...	...	
6983	0	1	0	0	0	
6984	0	1	0	0	0	
6985	0	0	0	0	1	
6986	0	0	1	0	0	
6987	0	0	1	0	0	
6988	0	0	0	1	0	
6989	0	1	0	0	0	
6990	1	0	0	0	0	
6991	0	1	0	0	0	
6992	0	0	0	1	0	
6993	0	1	0	0	0	
6994	0	0	0	0	1	
6995	0	0	0	0	1	
6996	0	0	1	0	0	
6997	0	0	0	0	1	
6998	0	1	0	0	0	
6999	0	1	0	0	0	
7000	0	1	0	0	0	
7001	0	0	1	0	0	
7002	0	1	0	0	0	
7003	0	1	0	0	0	
7004	0	0	0	0	1	
7005	0	0	1	0	0	
7006	0	0	0	0	1	
7007	0	1	0	0	0	
7008	1	0	0	0	0	
7009	0	1	0	0	0	
7010	0	0	1	0	0	
7011	0	1	0	0	0	
7012	0	1	0	0	0	
7013	0	0	0	0	1	
7014	0	1	0	0	0	
7015	0	1	0	0	0	
7016	1	0	0	0	0	
7017	0	1	0	0	0	
7018	0	1	0	0	0	
7019	0	0	0	0	1	
7020	0	0	1	0	0	
7021	0	0	0	0	1	
7022	0	0	0	0	1	
7023	0	0	0	0	1	
7024	0	1	0	0	0	

	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas	Gender_
7025	0	0	1	0	0	
7026	1	0	0	0	0	
7027	0	0	0	1	0	
7028	0	0	0	1	0	
7029	0	0	1	0	0	
7030	0	1	0	0	0	
7031	0	0	1	0	0	
7032	0	1	0	0	0	

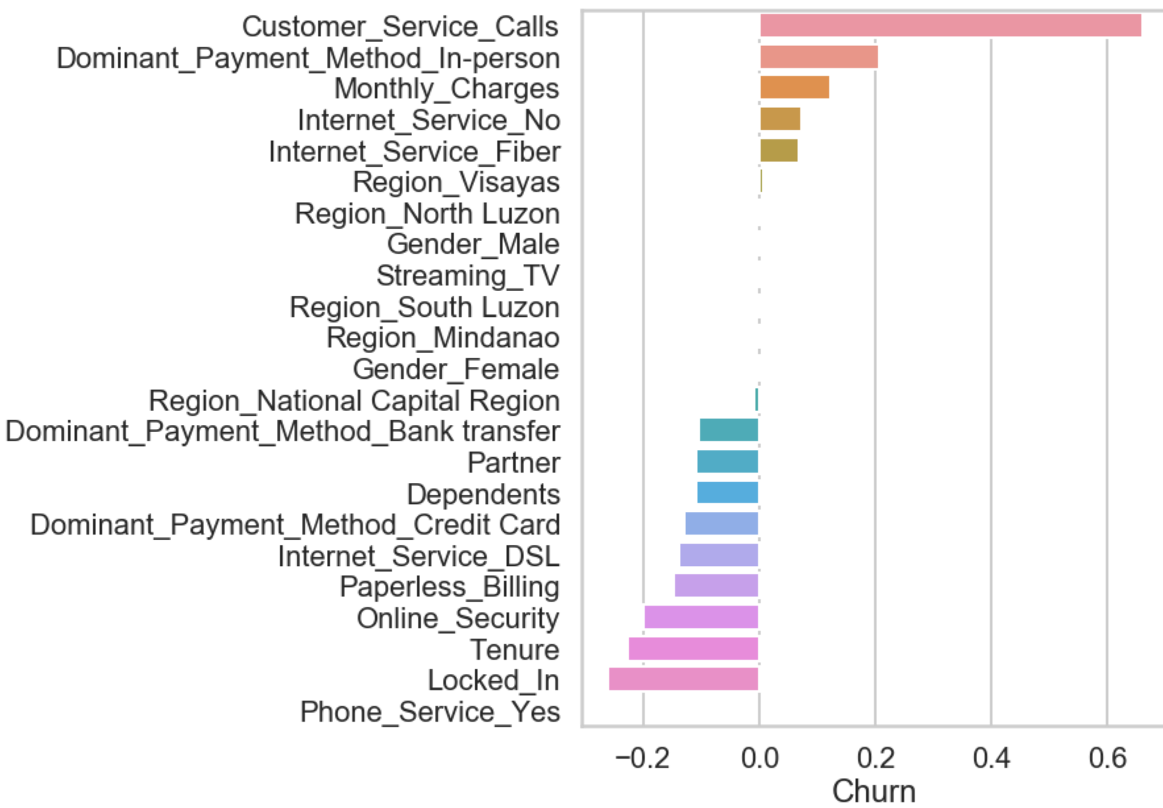
7032 rows × 23 columns

In [320]:
```
#Creating a correlation heatmap
corrmat = df.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True, cmap="YlOrRd");
```



For this heatmap, the darker the color red is, the more likely it is correlated.

In [321]:
```
#Checking the correlation using the barplot and being sorted in descending ord
er
corr = cormat.sort_values('Churn', ascending=False)
plt.figure(figsize=(8,10))
sns.barplot( corr.Churn[1:], corr.index[1:], orient='h')
plt.show()
```



As you can notice, Customer Service Calls is seen to be highly correlated with the Churn variable, followed by paying In-person, which is by a huge margin/gap.

In [322]:
```
#This shows the highly correlated features which is above moderate coefficient
(0.5)
#Correlation with output variable
cor_target = abs(cormat["Churn"])

#Selecting highly correlated features (Anything with a correlation >0.5)
relevant_features = cor_target[cor_target>0.5]
relevant_features.sort_values(ascending=False)
```

Out[322]:
```
Churn                     1.00
Customer_Service_Calls    0.66
Name: Churn, dtype: float64
```

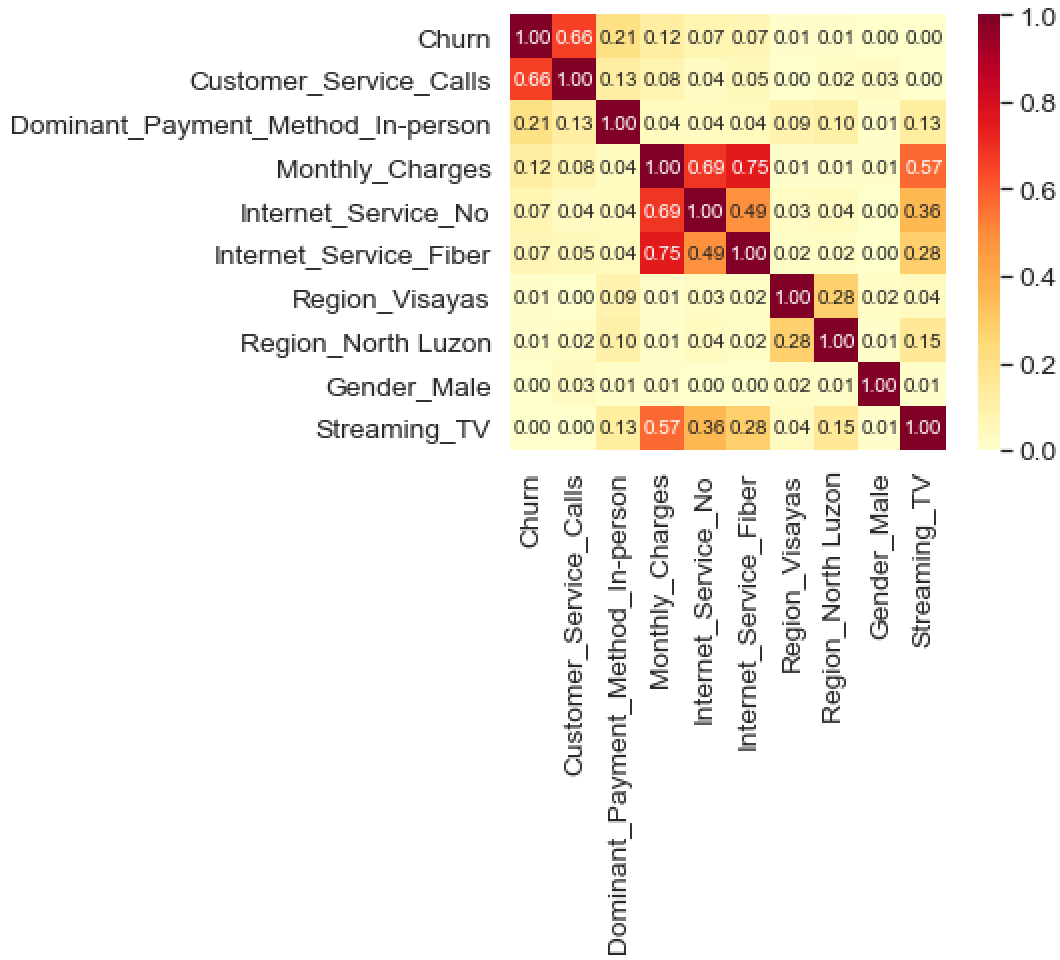
In [323]:
```
#Churn correlation matrix to better understand the correlation heatmap

#number of variables for heatmap
k = 10

cols = corrmat.nlargest(k, 'Churn')['Churn'].index

cm = abs(np.corrcoef(df[cols].values.T))

sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cmap = 'YlOrRd', cbar=True, annot=True, square=True, fmt=
'.2f', annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.value
s)
#annot = True in sns
plt.show()
```



Assumptions and Hypothesis gathered

- *'Customer_Service_Calls'* is strongly correlated with *'Churn'*. (Most who Churn are those who encounter issues with Customer Service Calls)
- Highly correlated pairs: *'Monthly_Charges'* with *'Internet_Service'* (both 'Internet_Service_No' and 'Internet_Service_Fiber') and *'Streaming_TV'*
- 'Interesting Insights: *'Region_Visayas'* and *'Region_North_Luzon'* are regions showing correlation than Mindanao, South Luzon, and NCR in Churn (must be with the competitive market)
- *'Gender_Male'* showing correlation in Customer Service Calls (maybe most males are the ones availing Customer Service)

Feature Engineering

Feature Engineering means creating new features that are derived from the previous dataset created. Such samples are those who are High Paying or not.

In [324]:
```
df.dtypes
```

Out[324]:
```
Customer_ID                                 object
Region_Mindanao                              uint8
Region_National Capital Region               uint8
Region_North Luzon                           uint8
Region_South Luzon                           uint8
Region_Visayas                               uint8
Gender_Female                                uint8
Gender_Male                                  uint8
Phone_Service_Yes                            uint8
Internet_Service_DSL                         uint8
Internet_Service_Fiber                       uint8
Internet_Service_No                          uint8
Dominant_Payment_Method_Bank transfer        uint8
Dominant_Payment_Method_Credit Card          uint8
Dominant_Payment_Method_In-person            uint8
Partner                                      int64
Dependents                                   int64
Tenure                                       int64
Customer_Service_Calls                       int64
Online_Security                              int64
Streaming_TV                                 int64
Locked_In                                    int64
Paperless_Billing                            int64
Monthly_Charges                              int64
Churn                                        int64
dtype: object
```

Features to be added:

- Tenure -

```
     Long-term: Those who have been availing the service for longer Period of t
ime 5 - 6 years (49 - 72 months)
     Steady: Those that are currently in the middle of 3 - 4 years (25 - 48 mon
ths)
     Short-term: Those who are new in availing the services 0- 2 years (0- 24 m
onths)
```

- Customer_Service_Calls

```
     Not_User: Those who never us the Customer Services (0 calls)
     Minimal Users: Those who use Customer Services the least (1 - 3 calls)
     Frequent Users: Those who use the phone calls very frequent (4 - 7 calls)
```

- Monthly_Charges

```
     High-Charged: Those who pay 1700 - 2500
     Middle-Charged: Those who pay 900 - 1600
     Low-Charged: Those who pay pay 400 - 800
```

- Avail_Service

```
     Phone_and_Internet_Service : Those who are subscribed to Phone and Interne
t service
```

In [325]:

```
#Setting index as identifer in the dataset
df.head()
df.set_index('Customer_ID')
```

Out[325]:

Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas
0002-ORFBO	0	0	1	0	0
0003-MKNFE	0	0	0	1	0
0004-TLHLJ	0	0	0	1	0
0011-IGKFF	0	1	0	0	0
0013-EXCHZ	1	0	0	0	0
0013-MHZWF	0	1	0	0	0
0013-SMEOE	0	0	1	0	0
0014-BMAQU	0	0	1	0	0
0015-UOCOJ	0	0	1	0	0
0016-QLJIS	0	0	0	0	1
0017-DINOC	0	0	1	0	0
0017-IUDMW	0	0	0	0	1
0018-NYROU	0	0	0	0	1
0019-EFAEP	0	0	1	0	0
0019-GFNTW	0	0	0	0	1
0020-INWCK	0	0	1	0	0
0020-JDNXP	0	1	0	0	0
0021-IKXGC	0	1	0	0	0
0022-TCJCI	0	0	1	0	0
0023-HGHWL	0	0	0	0	1
0023-UYUPN	0	0	1	0	0
0023-XUOPT	0	0	1	0	0
0027-KWYKW	1	0	0	0	0
0030-FNXPP	0	1	0	0	0
0031-PVLZI	0	1	0	0	0
0032-PGELS	0	0	1	0	0
0036-IHMOT	0	1	0	0	0
0040-HALCW	0	1	0	0	0
0042-JVWOJ	0	1	0	0	0
0042-RLHYP	0	1	0	0	0
0048-LUMLS	0	1	0	0	0
0048-PIHNL	0	0	1	0	0
0052-DCKON	0	0	0	0	1
0052-YNYOT	0	0	1	0	0
0056-EPFBG	0	0	0	0	1
0057-QBUQH	0	1	0	0	0
0058-EVZWM	0	0	1	0	0
0060-FUALY	0	0	0	0	1
0064-SUDOG	0	0	1	0	0
0064-YIJGF	0	0	0	1	0
0067-DKWBL	0	1	0	0	0

Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas
0068-FIGTF	0	1	0	0	0
0071-NDAFP	0	1	0	0	0
0074-HDKDG	0	1	0	0	0
0076-LVEPS	0	0	1	0	0
0078-XZMHT	0	1	0	0	0
0080-EMYVY	0	1	0	0	0
0080-OROZO	0	1	0	0	0
0082-LDZUE	0	1	0	0	0
0082-OQIQY	0	0	0	0	1
...	...	...	...	...	...
9924-JPRMC	0	1	0	0	0
9926-PJHDQ	0	1	0	0	0
9927-DSWDF	0	0	0	0	1
9928-BZVLZ	0	0	1	0	0
9929-PLVPA	0	0	1	0	0
9931-DCEZH	0	0	0	1	0
9931-KGHOA	0	1	0	0	0
9932-WBWIK	1	0	0	0	0
9933-QRGTX	0	1	0	0	0
9938-EKRGF	0	0	0	1	0
9938-PRCVK	0	1	0	0	0
9938-TKDGL	0	0	0	0	1
9938-ZREHM	0	0	0	0	1
9940-HPQPG	0	0	1	0	0
9940-RHLFB	0	0	0	0	1
9943-VSZUV	0	1	0	0	0
9944-AEXBM	0	1	0	0	0
9944-HKVVB	0	1	0	0	0
9945-PSVIP	0	0	1	0	0
9947-OTFQU	0	1	0	0	0
9948-YPTDG	0	1	0	0	0
9950-MTGYX	0	0	0	0	1
9953-ZMKSM	0	0	1	0	0
9955-QOPOY	0	0	0	0	1
9957-YODKZ	0	1	0	0	0
9958-MEKUC	1	0	0	0	0
9959-WOFKT	0	1	0	0	0
9961-JBNMK	0	0	1	0	0
9962-BFPDU	0	1	0	0	0
9964-WBQDJ	0	1	0	0	0
9965-YOKZB	0	0	0	0	1
9967-ATRFS	0	1	0	0	0
9968-FFVVH	0	1	0	0	0
9970-QBCDA	1	0	0	0	0
9971-ZWPBF	0	1	0	0	0

Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas
9972-EWRJS	0	1	0	0	0
9972-NKTFD	0	0	0	0	1
9972-VAFJJ	0	0	1	0	0
9974-JFBHQ	0	0	0	0	1
9975-GPKZU	0	0	0	0	1
9975-SKRNR	0	0	0	0	1
9978-HYCIN	0	1	0	0	0
9979-RGMZT	0	0	1	0	0
9985-MWVIX	1	0	0	0	0
9986-BONCE	0	0	0	1	0
9987-LUTYD	0	0	0	1	0
9992-RRAMN	0	0	1	0	0
9992-UJOEL	0	1	0	0	0
9993-LHIEB	0	0	1	0	0
9995-HOTOH	0	1	0	0	0

7032 rows × 24 columns

Tenure Engineering

- Long-term: Those who have been availing the service for longer Period of time 5 - 6 years (49 - 72 months)
- Steady: Those that are currently in the middle of 3 - 4 years (25 - 48 months)
- Short-term: Those who are new in availing the services 0- 2 years (0- 24 months)

In [326]:
```
#Creating new features in relation to Tenure feature.
df_tenure = df.copy()
a = df_tenure['Tenure']
df_tenure['Long_term'] = np.where(a >= 49, 1, 0)
df_tenure.head()
```

Out[326]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [327]:
```
#Checking the counts of each value
df_tenure['Long_term'].value_counts()
```

Out[327]:
```
0    4795
1    2237
Name: Long_term, dtype: int64
```

In [328]:
```
df_tenure['Steady'] = np.where((a >= 25) & (a <= 48), 1, 0)
```

In [329]:
```
df_tenure.Steady.value_counts()
```
Out[329]:
```
0    5438
1    1594
Name: Steady, dtype: int64
```

In [330]:
```
df_tenure['Short_term'] = np.where((a >= 0) & (a <= 24), 1, 0)
```

In [331]:
```
df_tenure.Short_term.value_counts()
```
Out[331]:
```
0    3831
1    3201
Name: Short_term, dtype: int64
```

In [332]:
```
df_tenure.head()
```
Out[332]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

Customer_Service_Calls Engineering

- Not_User: Those who never use the Customer Services (0 calls)
- Min_User: Those who use Customer Services the least (1 - 3 calls)
- Freq_User: Those who use the phone calls very frequent (4 - 7 calls)

In [333]:
```
df_customer = df_tenure.copy()
b = df_customer['Customer_Service_Calls']
df_customer['Not_User'] = np.where((b == 0), 1, 0)
df_customer.head()
```
Out[333]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [334]:
```
df_customer.Not_User.value_counts()
```
Out[334]:
```
0    5462
1    1570
Name: Not_User, dtype: int64
```

In [335]:
```
df_customer['Min_User'] = np.where((b >= 1) & (b <= 3), 1, 0)
df_customer.head()
```

Out[335]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [336]:
```
df_customer.Min_User.value_counts()
```

Out[336]:
```
1    4990
0    2042
Name: Min_User, dtype: int64
```

In [337]:
```
df_customer['Freq_User'] = np.where((b >= 4) & (b <= 7), 1, 0)
df_customer.head()
```

Out[337]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [338]:
```
df_customer.Freq_User.value_counts()
```

Out[338]:
```
0    6560
1     472
Name: Freq_User, dtype: int64
```

Monthly_Charges Engineering

- High-Charged: Those who pay 1700 - 2500 pesos
- Middle-Charged: Those who pay 900 and below 1600 pesos
- Low-Charged: Those who pay pay 400 - 800 pesos

In [339]:
```
df_monthch = df_customer.copy()
c = df_monthch['Monthly_Charges']
df_monthch['High_Charged'] = np.where((c >= 1600) & (c <= 2500), 1, 0)
df_monthch.head()
```

Out[339]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [340]:
```
df_monthch.High_Charged.value_counts()
```
Out[340]:
```
0    4325
1    2707
Name: High_Charged, dtype: int64
```

In [341]:
```
df_monthch['Middle_Charged'] = np.where((c >= 800) & (c < 1600), 1, 0)
df_monthch.head()
```
Out[341]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [342]:
```
df_monthch.Middle_Charged.value_counts()
```
Out[342]:
```
0    4546
1    2486
Name: Middle_Charged, dtype: int64
```

In [343]:
```
df_monthch['Low_Charged'] = np.where((c >= 0) & (c < 800), 1, 0)
df_monthch.head()
```
Out[343]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [344]:
```
df_monthch.Low_Charged.value_counts()
```
Out[344]:
```
0    5202
1    1830
Name: Low_Charged, dtype: int64
```

In [345]:
```
df_monthch.dtypes
```

Out[345]:
```
Customer_ID                              object
Region_Mindanao                           uint8
Region_National Capital Region            uint8
Region_North Luzon                        uint8
Region_South Luzon                        uint8
Region_Visayas                            uint8
Gender_Female                             uint8
Gender_Male                               uint8
Phone_Service_Yes                         uint8
Internet_Service_DSL                      uint8
Internet_Service_Fiber                    uint8
Internet_Service_No                       uint8
Dominant_Payment_Method_Bank transfer     uint8
Dominant_Payment_Method_Credit Card       uint8
Dominant_Payment_Method_In-person         uint8
Partner                                   int64
Dependents                                int64
Tenure                                    int64
Customer_Service_Calls                    int64
Online_Security                           int64
Streaming_TV                              int64
Locked_In                                 int64
Paperless_Billing                         int64
Monthly_Charges                           int64
Churn                                     int64
Long_term                                 int32
Steady                                    int32
Short_term                                int32
Not_User                                  int32
Min_User                                  int32
Freq_User                                 int32
High_Charged                              int32
Middle_Charged                            int32
Low_Charged                               int32
dtype: object
```

In [346]:
```
df_monthch.Phone_Service_Yes.value_counts()
```

Out[346]:
```
1    7032
Name: Phone_Service_Yes, dtype: int64
```

Package Services Engineering

- Phone_and_Internet_Service : Those who are subscribed to Phone and Internet service

(Phone_Service is already considered as a part of the services so we'll focus on the Internet service package for assessment)

In [347]:
```
df_service = df_monthch.copy()
d = df_service['Internet_Service_DSL']
e = df_service['Internet_Service_Fiber']
f = df_service['Phone_Service_Yes']
df_service['Phone_And_Internet'] = np.where( (f == 1) & (d == 1) | (e == 1), 1
, 0)
df_service.head()
```

Out[347]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [348]:
```
df_service.Phone_And_Internet.value_counts()
```

Out[348]:
```
1    5253
0    1779
Name: Phone_And_Internet, dtype: int64
```

In [349]:
```
df_service.to_csv("dataset_fE.csv")
```

Feature Selection

Feature selection is the process where we choose the features from the pre-existing and ones engineered that are can give more relevance to our dataset when we run our model. This is also done in order to reduce the features to be included in our dataset. The fewer the features, the better the models run.

In [350]:
```
df_s = df_service.copy()
```

In [351]:
```
df_s.head()
```

Out[351]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [352]:
```
df_s.shape
```

Out[352]:
```
(7032, 35)
```

Separating Features from Target (Churn)

In [353]:
```
#Separating the Churn variable, since it's our targets and Customer ID
features = df_s.drop(columns=['Churn','Customer_ID'], axis=1)
target = df_s.Churn
cust = df_s.Customer_ID
h = df_s.Monthly_Charges
```

Low Variance Filter

Variance is a statistical measure of the amount of variation in the given variable or feature. In simple explanation, variance tells of how it can bring change significantly to your data. The higher the variance, the more significant it is to your data. In the Low Variance Filter, it "filters" your features, calculates their variance and tells you the features that actually doesn't give much importance to your data that you can just drop.

In [354]:
```
# Compute the variance and sort
features.var().sort_values()[:60]
```

Out[354]:
```
Phone_Service_Yes                              0.00
Freq_User                                      0.06
Region_Mindanao                                0.08
Region_South Luzon                             0.11
Paperless_Billing                              0.12
Region_Visayas                                 0.15
Not_User                                       0.17
Steady                                         0.18
Region_North Luzon                             0.19
Internet_Service_No                            0.19
Phone_And_Internet                             0.19
Dominant_Payment_Method_Bank transfer          0.19
Low_Charged                                    0.19
Dominant_Payment_Method_Credit Card            0.20
Min_User                                       0.21
Dependents                                     0.21
Long_term                                      0.22
Internet_Service_DSL                           0.22
Region_National Capital Region                 0.23
Middle_Charged                                 0.23
High_Charged                                   0.24
Online_Security                                0.24
Internet_Service_Fiber                         0.24
Streaming_TV                                   0.24
Locked_In                                      0.25
Short_term                                     0.25
Dominant_Payment_Method_In-person              0.25
Partner                                        0.25
Gender_Male                                    0.25
Gender_Female                                  0.25
Customer_Service_Calls                         1.86
Tenure                                       603.10
Monthly_Charges                           363742.62
dtype: float64
```

In [355]:
```
#Import the VarianceThreshold Function
from sklearn.feature_selection import VarianceThreshold

#Instantiate the Function and Set the Threshold
selector = VarianceThreshold(0.05)
```

Variance Threshold is a feature in the sklearn package where it decides those features that doesn't meet the threshold of 0.05. The idea was to keep those zero variance features and drop it.

In [356]:
```
#Apply the Function to filter out the Low Variance Columns/Features
filtered_features = pd.DataFrame(selector.fit_transform(features), index=featu
res.index)
```

In [357]:
```
#Note, the DataFrame Created has no Column Names
filtered_features.head()
```

Out[357]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	0	0	1	0	0	1	0	1	0	0	0	0	1	1	1	9	2	0	1	1	0	1310	0	0	1
1	0	0	0	1	0	0	1	1	0	0	0	0	1	0	0	9	1	0	0	0	0	1200	0	0	1
2	0	0	0	1	0	0	1	0	1	0	0	0	1	0	0	4	0	0	0	0	0	1480	0	0	1
3	0	1	0	0	0	0	1	0	1	0	0	0	1	1	0	13	1	0	1	0	0	1960	0	0	1
4	1	0	0	0	0	1	0	0	1	0	0	0	1	1	0	3	2	0	1	0	0	1680	0	0	1

In [358]:
```
#Use this attribute to get Column Names
selected = selector.get_support()

#Rename the columns
filtered_features.columns = features.columns[selected]
```

In [359]:
```
filtered_features.head()
```

Out[359]:

	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas	Gender_Fei
0	0	0	1	0	0	
1	0	0	0	1	0	
2	0	0	0	1	0	
3	0	1	0	0	0	
4	1	0	0	0	0	

In [360]:
```
filtered_features.var().sort_values().head()
```

Out[360]:
```
Freq_User           0.06
Region_Mindanao     0.08
Region_South Luzon  0.11
Paperless_Billing   0.12
Region_Visayas      0.15
dtype: float64
```

In [361]:
```
#Compare previous vs current number of Features
print("No. of Features (Original): %i" %len(features.columns))
print("No. of Features (Variance Filter): %i" %len(filtered_features.columns))
```

```
No. of Features (Original): 33
No. of Features (Variance Filter): 32
```

High Correlation Filter

High Correlation filter works by removing those highly correlated features that may carry the similar trendsto reduce our model.

In [362]:
```
corr_matrix = filtered_features.corr().abs()
```

In [363]:
```
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.b
ool))
```

In [364]:
```
threshold = 0.80
```

In [365]:
```
to_drop = [column for column in upper.columns if any(upper[column] > threshold
)]
```

In [366]:
```
to_drop
```

Out[366]:
```
['Gender_Male',
 'Long_term',
 'Short_term',
 'Min_User',
 'Freq_User',
 'Low_Charged',
 'Phone_And_Internet']
```

In [367]:
```
filtered_features_2 = filtered_features.drop(to_drop, axis=1)
```

In [368]:
```
#Compare previous vs current number of Features
print("No. of Features (Original): %i" %len(features.columns))
print("No. of Features (Variance Filter): %i" %len(filtered_features.columns))
print("No. of Features (Correlation Filter): %i" %len(filtered_features_2.colu
mns))
```

```
No. of Features (Original): 33
No. of Features (Variance Filter): 32
No. of Features (Correlation Filter): 25
```

In [369]:
```
filtered_features_2.head()
```

Out[369]:

	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visayas	Gender_Fei
0	0	0	1	0	0	
1	0	0	0	1	0	
2	0	0	0	1	0	
3	0	1	0	0	0	
4	1	0	0	0	0	

Custom Function

Theoffers the same step with the other two but for this one, it is created in having its own function of passing our threshold and our data in automaticlly dropping the the filtered features.

In [370]:
```
def correlation_filter(df_s,threshold):
    # Create correlation matrix
    corr_matrix = df_s.corr().abs()

    # Select upper triangle of correlation matrix
    upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(
np.bool))

    # Find index of feature columns with correlation greater than threshold
    to_drop = [column for column in upper.columns if any(upper[column] > thres
hold)]

    return to_drop
```

In [371]:
```
function_test = correlation_filter(filtered_features,threshold)
```

In [372]:
```
len(function_test)
```

Out[372]:
```
7
```

In [373]:
```
function_test
```

Out[373]:
```
['Gender_Male',
 'Long_term',
 'Short_term',
 'Min_User',
 'Freq_User',
 'Low_Charged',
 'Phone_And_Internet']
```

Verification

In [374]:
```
# Print out top correlated features

#Reshape the Matrix
correlated = corr_matrix.unstack()

#Reset Index from Multi-index to single index
correlated = correlated.reset_index(level=0).reset_index()

#Rename Columns
correlated.columns = ["Feature1", "Feature2", "Correlation"]

#Sort by Correlation Value
corr_sorted = correlated.sort_values("Correlation", ascending=False)
```

In [375]:
```
correlated.head()
```

Out[375]:

	Feature1	Feature2	Correlation
0	Region_Mindanao	Region_Mindanao	1.00
1	Region_National Capital Region	Region_Mindanao	0.23
2	Region_North Luzon	Region_Mindanao	0.18
3	Region_South Luzon	Region_Mindanao	0.12
4	Region_Visayas	Region_Mindanao	0.15

In [376]:
```
corr_sorted_pairs = corr_sorted[corr_sorted['Feature1'].values != corr_sorted[
'Feature2'].values]
```

In [377]:
```
corr_sorted_pairs.reset_index(drop=True,inplace=True)
```

In [378]:
```
corr_sorted_final = corr_sorted_pairs.iloc[::2]
```

In [379]:
```
corr_sorted_final.Feature1.nunique()
```

Out[379]: 32

In [380]:
```
corr_sorted_final_ver = corr_sorted_final[corr_sorted_final.Correlation > thre
shold]
```

In [381]:
```
corr_sorted_final_ver
```

Out[381]:

	Feature1	Feature2	Correlation
0	Phone_And_Internet	Internet_Service_No	1.00
2	Gender_Male	Gender_Female	1.00
4	Tenure	Short_term	0.87
6	Tenure	Long_term	0.85
8	Not_User	Min_User	0.84
10	Monthly_Charges	Low_Charged	0.82
12	Customer_Service_Calls	Freq_User	0.82
14	Phone_And_Internet	Low_Charged	0.80
16	Internet_Service_No	Low_Charged	0.80

Low Correlation to Target Filter

In [382]:
```
df_temp = pd.concat([cust,filtered_features_2, target], axis =1, join='inner')
```

In [383]:
```
df_temp.shape
```

Out[383]: (7032, 27)

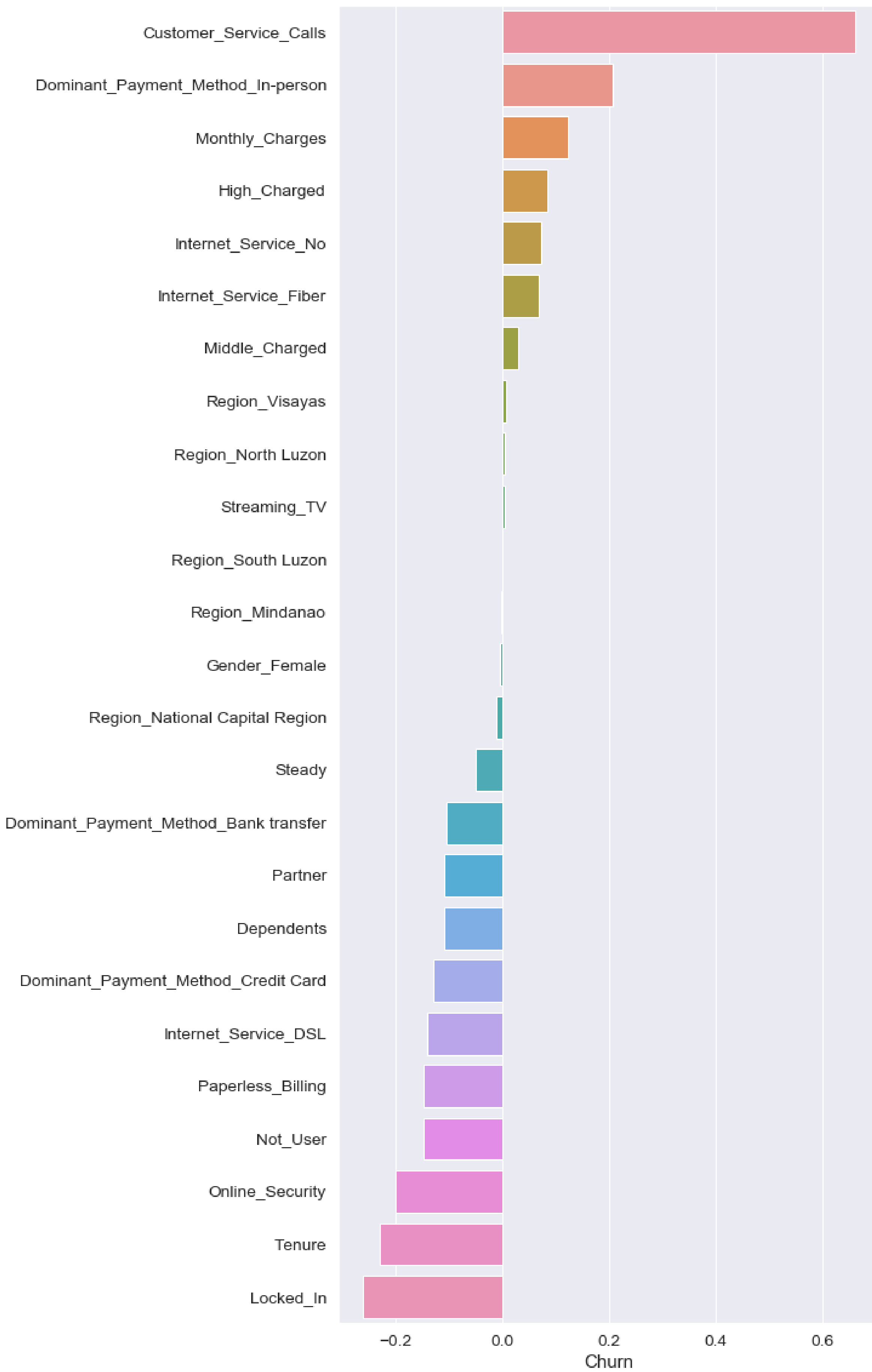
In [384]:
```
df_temp.head()
```

Out[384]:

	Customer_ID	Region_Mindanao	Region_National Capital Region	Region_North Luzon	Region_South Luzon	Region_Visaya
0	0002-ORFBO	0	0	1	0	
1	0003-MKNFE	0	0	0	1	
2	0004-TLHLJ	0	0	0	1	
3	0011-IGKFF	0	1	0	0	
4	0013-EXCHZ	1	0	0	0	

In [385]:
```
#Get the Correlation
corrmat = df_temp.corr()
corr = corrmat.sort_values('Churn', ascending=False)

plt.figure(figsize=(8,20))
sns.barplot( corr.Churn[1:], corr.index[1:], orient='h')
plt.show()
```



In [386]:
```
#Get absolute values and sort by lowest to highest
corr_table =  abs(corr.Churn[1:]).sort_values(ascending = True)
#corr_table[0:20]
```

In [387]:
```
#Option1: Based on Threshold, or the rule of thumb
cor_threshold = 0.20
cols_to_drop = corr_table[corr_table < cor_threshold].index
```

In [388]:
```
len(cols_to_drop)
```
Out[388]:
```
21
```

In [389]:
```
#Option2: lowest N features
cor_lowest_n = 20
cols_to_drop = corr_table[0:cor_lowest_n].index.values
```

In [390]:
```
len(cols_to_drop)
```
Out[390]:
```
20
```

In [391]:
```
cols_to_drop
```
Out[391]:
```
array(['Region_South Luzon', 'Region_Mindanao', 'Streaming_TV',
       'Gender_Female', 'Region_North Luzon', 'Region_Visayas',
       'Region_National Capital Region', 'Middle_Charged', 'Steady',
       'Internet_Service_Fiber', 'Internet_Service_No', 'High_Charged',
       'Dominant_Payment_Method_Bank transfer', 'Partner', 'Dependents',
       'Monthly_Charges', 'Dominant_Payment_Method_Credit Card',
       'Internet_Service_DSL', 'Paperless_Billing', 'Not_User'],
      dtype=object)
```

In [392]:
```
df_final = df_temp.drop(cols_to_drop, axis=1)
```

In [393]:
```
#Compare previous vs current number of Features
print("No. of Features (Original): %i" %len(features.columns))
print("No. of Features (Variance Filter): %i" %len(filtered_features.columns))
print("No. of Features (Correlation Filter): %i" %len(filtered_features_2.colu
mns))
print("No. of Features (Correlation Filter): %i" %(len(df_final.columns)-1))
```
```
No. of Features (Original): 33
No. of Features (Variance Filter): 32
No. of Features (Correlation Filter): 25
No. of Features (Correlation Filter): 6
```

In [394]:
```
df_final.shape
```
Out[394]:
```
(7032, 7)
```

In [395]:
```
df_final.head()
```
Out[395]:

	Customer_ID	Dominant_Payment_Method_In-person	Tenure	Customer_Service_Calls	Online_Security
0	0002-ORFBO	1	9	2	0
1	0003-MKNFE	1	9	1	0
2	0004-TLHLJ	1	4	0	0
3	0011-IGKFF	1	13	1	0
4	0013-EXCHZ	1	3	2	0

In [396]:
```
df_final.to_csv("df_final.csv")
```

In [397]:
```
#abdf = pd.concat([df_dum1, df_final], axis =1, join='inner')
```

In [398]:
```
abdf = pd.merge(df_dum1, df_final, how='inner', on="Customer_ID")
```

In [399]:
```
abdf.head()
```

Out[399]:

	Customer_ID	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Method
0	0002-ORFBO	North Luzon	Female	Yes	DSL	In-person
1	0003-MKNFE	South Luzon	Male	Yes	DSL	In-person
2	0004-TLHLJ	South Luzon	Male	Yes	Fiber	In-person
3	0011-IGKFF	National Capital Region	Male	Yes	Fiber	In-person
4	0013-EXCHZ	Mindanao	Female	Yes	Fiber	In-person

In [400]:
```
abdf = pd.concat([abdf, h], axis=1, join="inner")
```

Data Modelling - Logistic Regression

One of the effective models to be used in costumer churn analysis is Logistic Regression. Logistic regression predicts the values between 0 and 1, based on the sigmoid function. In our usecase, it predicts whether a customer has churn or not. It is training the model in looking for patterns that could help identify in solving our usecase, and how effective it is.

In [401]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, accuracy_
score

from sklearn.preprocessing import MinMaxScaler
```

For this one, we imported some sklearn features that would perform the function.

- LogisticRegression - is one of the models you can play with inside sklearns function.
- train_test_split - this module separates our dataset into 2 parts: our training dataset and testing dataset.
- confusion matrix, classification_report, accuracy score - metrics to determine if our model is working well.
- MinMaxScaler - preprocessing feature where it scales our data before being trained to the model.

In [402]:
```
abdf.head()
```

Out[402]:

	Customer_ID	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Method
0	0002-ORFBO	North Luzon	Female	Yes	DSL	In-person
1	0003-MKNFE	South Luzon	Male	Yes	DSL	In-person
2	0004-TLHLJ	South Luzon	Male	Yes	Fiber	In-person
3	0011-IGKFF	National Capital Region	Male	Yes	Fiber	In-person
4	0013-EXCHZ	Mindanao	Female	Yes	Fiber	In-person

In [403]:
```
df_final.head()
```
Out[403]:

	Customer_ID	Dominant_Payment_Method_In-person	Tenure	Customer_Service_Calls	Online_Security
0	0002-ORFBO	1	9	2	0
1	0003-MKNFE	1	9	1	0
2	0004-TLHLJ	1	4	0	0
3	0011-IGKFF	1	13	1	0
4	0013-EXCHZ	1	3	2	0

In [404]:
```
abdf.describe()
```
Out[404]:

	Dominant_Payment_Method_In-person	Tenure	Customer_Service_Calls	Online_Security	Locked_
count	7031.00	7031.00	7031.00	7031.00	7031.
mean	0.47	32.39	1.35	0.41	0.
std	0.50	24.56	1.36	0.49	0.
min	0.00	0.00	0.00	0.00	0.
25%	0.00	9.00	1.00	0.00	0.
50%	0.00	29.00	1.00	0.00	0.
75%	1.00	55.00	2.00	1.00	1.
max	1.00	72.00	7.00	1.00	1.

In [405]:
```
df_final.describe()
```
Out[405]:

	Dominant_Payment_Method_In-person	Tenure	Customer_Service_Calls	Online_Security	Locked_
count	7032.00	7032.00	7032.00	7032.00	7032.
mean	0.47	32.40	1.35	0.41	0.
std	0.50	24.56	1.36	0.49	0.
min	0.00	0.00	0.00	0.00	0.
25%	0.00	9.00	1.00	0.00	0.
50%	0.00	29.00	1.00	0.00	0.
75%	1.00	55.00	2.00	1.00	1.
max	1.00	72.00	7.00	1.00	1.

In [406]:
```
abdf.shape
```
Out[406]:
```
(7031, 13)
```

In [407]:
```
abdf.dtypes
```

Out[407]:
```
Customer_ID                          object
Region                               object
Gender                               object
Phone_Service                        object
Internet_Service                     object
Dominant_Payment_Method              object
Dominant_Payment_Method_In-person     int64
Tenure                                int64
Customer_Service_Calls                int64
Online_Security                       int64
Locked_In                             int64
Churn                                 int64
Monthly_Charges                       int64
dtype: object
```

In [408]:
```
df_final.shape
```

Out[408]:
```
(7032, 7)
```

In [409]:
```
abdf['Churn'].value_counts()
```

Out[409]:
```
0    6114
1     917
Name: Churn, dtype: int64
```

In [410]:
```
#Identify and check the value counts of the target variable
df_final['Churn'].value_counts()
```

Out[410]:
```
0    6115
1     917
Name: Churn, dtype: int64
```

In [411]:
```
abdf.dtypes
```

Out[411]:
```
Customer_ID                          object
Region                               object
Gender                               object
Phone_Service                        object
Internet_Service                     object
Dominant_Payment_Method              object
Dominant_Payment_Method_In-person     int64
Tenure                                int64
Customer_Service_Calls                int64
Online_Security                       int64
Locked_In                             int64
Churn                                 int64
Monthly_Charges                       int64
dtype: object
```

In [412]:
```
abdf['Churn'].value_counts()
```

Out[412]:
```
0    6114
1     917
Name: Churn, dtype: int64
```

Building the Model

In [413]:
```
#Separate the Features and the Target Variable
X = abdf.drop(["Customer_ID", "Region", "Gender", "Phone_Service", "Internet_Service", "Dominant_Payment_Method","Churn", "Monthly_Charges"], axis=1)
y_1 = abdf["Churn"]
```

In [414]:
```
#Separate the Features and the Target Variable
#X = df_final.drop(["Churn"], axis=1)
#y_1 = df_final["Churn"]
```

In [415]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y_1, test_size=0.25, ra
ndom_state=101)
```

In [416]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[416]:
```
((5273, 5), (1758, 5), (5273,), (1758,))
```

Scaling

MinMaxScaler is used to preserve the shape of the original distribution, keeping it in the range of 0 and 1.

In [417]:
```
#Instantiate the MinMax Scaler
minmax = MinMaxScaler()

#Fit the scaler to the training set
#Because it it is still not used by the system
minmax.fit(X_train)

#Transform the training set
X_train_scaled = minmax.transform(X_train)

#Transform the test set
X_test_scaled = minmax.transform(X_test)
```

In [418]:
```
#View the scaled data
X_train_scaled
```

Out[418]:
```
array([[0.        , 0.68055556, 0.85714286, 0.        , 0.        ],
       [0.        , 0.27777778, 0.14285714, 0.        , 0.        ],
       [1.        , 0.56944444, 0.14285714, 0.        , 1.        ],
       ...,
       [0.        , 0.59722222, 0.42857143, 0.        , 0.        ],
       [1.        , 0.05555556, 0.14285714, 1.        , 0.        ],
       [0.        , 0.52777778, 0.14285714, 0.        , 1.        ]])
```

In [419]:
```
#View the type of the scaled data
type(X_train_scaled)
```

Out[419]:
```
numpy.ndarray
```

In [420]:
```
#Change to Pandas dataframe for easier viewing and manipulation of the data (t
ranformation of the data), Changing into standard dataframe
X_train_sdf = pd.DataFrame(X_train_scaled, index=X_train.index, columns=X_trai
n.columns) #Pass all values, starts with index and whee to get the columns
X_test_sdf = pd.DataFrame(X_test_scaled, index=X_test.index, columns=X_test.co
lumns)
```

In [421]:
```
X_train_sdf.describe()
```

Out[421]:

	Dominant_Payment_Method_In-person	Tenure	Customer_Service_Calls	Online_Security	Locked_
count	5273.00	5273.00	5273.00	5273.00	5273.
mean	0.47	0.45	0.19	0.41	0.
std	0.50	0.34	0.19	0.49	0.
min	0.00	0.00	0.00	0.00	0.
25%	0.00	0.12	0.14	0.00	0.
50%	0.00	0.40	0.14	0.00	0.
75%	1.00	0.76	0.29	1.00	1.
max	1.00	1.00	1.00	1.00	1.

In [422]:
```
X_train_sdf.head()
```

Out[422]:

	Dominant_Payment_Method_In-person	Tenure	Customer_Service_Calls	Online_Security	Locked_I
5290	0.00	0.68	0.86	0.00	0.0
2650	0.00	0.28	0.14	0.00	0.0
3712	1.00	0.57	0.14	0.00	1.0
6327	1.00	0.01	0.14	0.00	0.0
4259	0.00	1.00	0.14	1.00	1.0

Training the Model

In [423]:
```
#Instantiate the Algorithm
#giving more weight to lesser observations,
logreg = LogisticRegression(C=1e9, class_weight="balanced", solver='liblinear'
, random_state=25)

#Train/Fit the model
logreg.fit(X_train_sdf, y_train)
```

Out[423]:
```
LogisticRegression(C=1000000000.0, class_weight='balanced', dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='warn', n_jobs=None, penalty='l2', random_state=25,
          solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

In [424]:
```
#Check the Trained Model Coefficients of the features
print(logreg.coef_)
```

```
[[ 0.67853437 -0.57205198  8.80233723 -0.90943945 -1.52248388]]
```

In [425]:
```
#Create a DataFrame for easy understanding
coef = pd.DataFrame(X_train_sdf.columns, columns=["Features"])
coef['Coef'] = logreg.coef_.reshape(-1,1)
coef.head(10)
```

Out[425]:

	Features	Coef
0	Dominant_Payment_Method_In-person	0.68
1	Tenure	-0.57
2	Customer_Service_Calls	8.80
3	Online_Security	-0.91
4	Locked_In	-1.52

Validating the Model

In [426]:
```
#Make Predictions , validating the model- scaled training dataset, any transfo
rmation must be done on the training set )
y_pred = logreg.predict(X_test_sdf)
```

In [427]:
```
#Get the Confusion Matrix and other metrics to test performance (model precisi
on)
print("Classification report for classifier %s:\n%s\n"
      % (logreg, classification_report(y_test, y_pred)))
```

```
Classification report for classifier LogisticRegression(C=1000000000.0, class
_weight='balanced', dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='warn', n_jobs=None, penalty='l2', random_state=25,
          solver='liblinear', tol=0.0001, verbose=0, warm_start=False):
              precision    recall  f1-score   support

           0       0.97      0.86      0.91      1532
           1       0.46      0.83      0.59       226

   micro avg       0.85      0.85      0.85      1758
   macro avg       0.71      0.84      0.75      1758
weighted avg       0.91      0.85      0.87      1758
```

For our classification reports we have 4 metrics to consider:

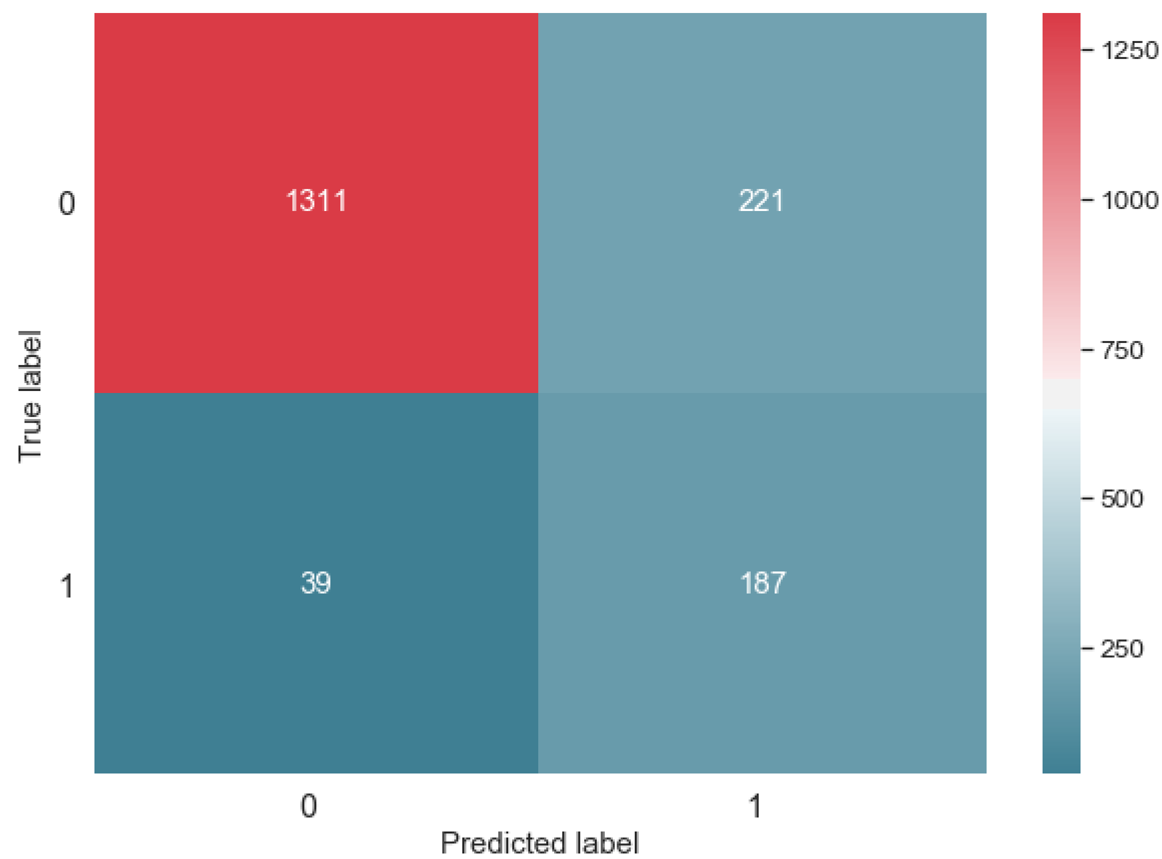
- Precision - Its ability to label a positive whne the sample was a negative.
- Recall - It's ability to find all the positive samples.
- F1-Score - Gives the mean between precision and Recall
- Support - Number of occurances in each class

In [428]:
```
#Plot the confusion matrix for easier viewing
cm = confusion_matrix(y_test, y_pred)

df_cm = pd.DataFrame(cm, index=[0,1], columns=[0,1])

fig = plt.figure(figsize= (10,7))
cmap = sns.diverging_palette(220, 10, as_cmap=True)
heatmap = sns.heatmap(df_cm,annot=True, fmt="d", cmap=cmap)
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=16)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=0, ha='right', fontsize=16)
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

Out[428]: Text(0.5, 37.5, 'Predicted label')



In [429]:
```
#Predict the Probabilities
pred_prob_0 = logreg.predict_proba(X_test_sdf)[:,0]
pred_prob_1 = logreg.predict_proba(X_test_sdf)[:,1]
```

In [430]:
```
#Put all information on a DataFrame for analysis
df_results = X_test.copy()
df_results["Monthly_Charges"] = h
df_results["Predicted_Class"] = y_pred
df_results["Predicted_Prob(0)"] = pred_prob_0
df_results["Predicted_Prob(1)"] = pred_prob_1
```

In [431]:
```
df_results.head()
```

Out[431]:

	Dominant_Payment_Method_In-person	Tenure	Customer_Service_Calls	Online_Security	Locked_In
4247	1	23	2	0	(
2910	1	1	1	0	(
1565	1	4	1	0	(
5219	0	46	1	1	(
4420	1	23	0	1	(

In [432]:
```
df_results = pd.concat([df_dum1, df_results, y_1], axis =1, join='inner')
```

In [433]:
```
df_results.head()
```

Out[433]:

	Customer_ID	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Method
0	0002-ORFBO	North Luzon	Female	Yes	DSL	In-person
6	0013-SMEOE	North Luzon	Female	Yes	Fiber	Bank transfer
7	0014-BMAQU	North Luzon	Male	Yes	Fiber	Credit Card
24	0031-PVLZI	National Capital Region	Female	Yes	No	In-person
30	0048-LUMLS	National Capital Region	Male	Yes	Fiber	Credit Card

Saving Results

In [434]:
```
df_results.to_csv("df_results.csv")
```

Getting Insights from the Predicted Classes and Churn

Storing Insights to Dataframes for Analysis

In [435]:
```
df_Not_Churn = df_results.loc[(df_results['Predicted_Class']== 0 ) | (df_results['Churn']== 0)]
```

In [436]:
```
df_Churn = df_results.loc[(df_results['Predicted_Class']== 1 ) | (df_results['Churn']== 1)]
```

In [437]:
```
df_Predicted_Churn = df_results.loc[(df_results['Predicted_Class']== 1 ) | (df_results['Churn']== 0)]
```

In [438]:
```
df_Predicted_Not_Churn = df_results.loc[(df_results['Predicted_Class']== 0 ) | (df_results['Churn']== 1)]
```

Saving each dataframes into Dataset

In [439]:
```
df_Churn.to_csv("df_Churn")
```

In [440]:
```
df_Predicted_Churn.to_csv("df_Predicted_Churn")
```

In [441]:
```
df_Not_Churn.to_csv("df_Not_Churn")
```

In [442]:
```
df_Predicted_Not_Churn.to_csv("df_Predicted_Not_Churn")
```

Digging to each Result

Churned Customers

In [443]:
```
df_Churn.head()
```

Out[443]:

	Customer_ID	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Method
54	0096-BXERS	Visayas	Female	Yes	DSL	In-person
70	0115-TFERT	National Capital Region	Male	Yes	No	In-person
74	0122-OAHPZ	National Capital Region	Female	Yes	No	In-person
86	0137-UDEUO	Visayas	Female	Yes	No	In-person
92	0151-ONTOV	North Luzon	Female	Yes	Fiber	In-person

In [444]:
```
df_Churn.shape
```

Out[444]: (447, 16)

In [445]:
```
df_Churn.describe()
```

Out[445]:

	Dominant_Payment_Method_In-person	Tenure	Customer_Service_Calls	Online_Security	Locked_
count	447.00	447.00	447.00	447.00	447.(
mean	0.81	15.26	2.69	0.11	0.(
std	0.40	18.10	1.93	0.32	0.2
min	0.00	1.00	0.00	0.00	0.(
25%	1.00	2.00	1.00	0.00	0.(
50%	1.00	7.00	2.00	0.00	0.(
75%	1.00	24.00	4.00	0.00	0.(
max	1.00	72.00	7.00	1.00	1.(

In [446]:
```
df_Churn.Region.value_counts()
```

Out[446]:
```
National Capital Region    154
North Luzon                108
Visayas                     86
South Luzon                 56
Mindanao                    43
Name: Region, dtype: int64
```

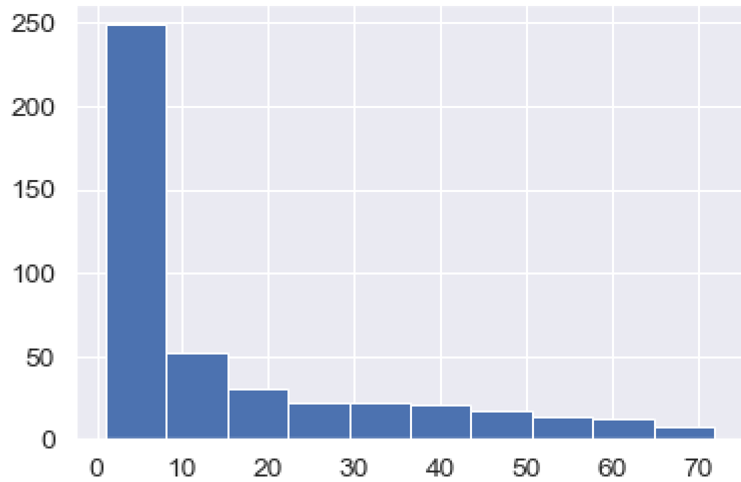
North Luzon and *National Capital Region* are the top regions having Customer Churns, with Luzon Area getting more than half of the total number.

In [447]:
```
df_Churn.Tenure.value_counts()
```

Out[447]:
```
1     98
2     36
3     30
4     28
5     18
7     15
6     13
8     11
14    10
9     10
17     9
11     8
13     8
37     7
10     7
58     7
18     6
24     6
12     5
25     5
34     5
47     5
22     4
15     4
26     4
31     4
33     4
35     4
21     4
55     4
56     4
69     4
42     4
16     3
57     3
32     3
29     3
20     3
43     3
48     3
44     2
19     2
49     2
45     2
23     2
36     2
41     2
39     2
46     2
38     2
50     1
51     1
53     1
54     1
27     1
71     1
40     1
59     1
60     1
62     1
63     1
64     1
68     1
28     1
72     1
Name: Tenure, dtype: int64
```

In [448]:
```
df_Churn.Tenure.hist()
```

Out[448]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22fd0b23668>
```



Most Customers Churn after their first year, with 60% of the customers churn in a little less/after a year (1 year and 2 months at most), and more than 75% of it churning after 8 months. It is also to be noted that the more longer the subscriber is bound to the service, the less likely they will Churn, with even reaching 6 years before they churn.

In [449]:
```
df_Churn.Customer_Service_Calls.value_counts()
```

Out[449]:
```
2    159
1    117
3     37
6     34
7     31
5     26
4     26
0     17
Name: Customer_Service_Calls, dtype: int64
```

Most Customers churns after 2 - 3 calls with Customer Service, with a little more than 61% of them churned after 1-2 calls, considering Filipinos not having too much patience in fixing issues and tend to avail other services.

In [450]:
```
df_Churn.Online_Security.value_counts()
```

Out[450]:
```
0    396
1     51
Name: Online_Security, dtype: int64
```

88.5% of those who churned doesn't avail the security package.

In [451]:
```
df_Churn.Locked_In.value_counts()
```

Out[451]:
```
0    422
1     25
Name: Locked_In, dtype: int64
```

94.5% of the Customers who churned doesn't also avail the Locked In penalty fee or it isn't included in their contract. Let's say that because of not having the Locked In penalty fee, it doesn't have anything that holds them in availing the service longer and therefore free to just churn around. Looking at a Filipino perspective, it would make sense since "walang bayad" naman or "walang fee" so "let's leave it at that".

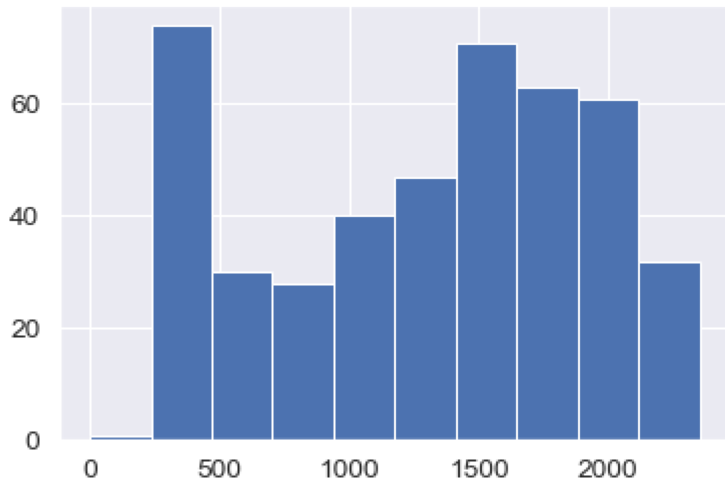
In [452]:
```
df_Churn["Dominant_Payment_Method_In-person"].value_counts()
```

Out[452]:
```
1    360
0     87
Name: Dominant_Payment_Method_In-person, dtype: int64
```

80.5% of the Customers who Churned are paying in-person, which in Filipino behavior, makes sense since you are not worried of your running bill when you either pay through credit card or bank transfer, having the habit of "pawalang-bahala" na lang.

In [453]:
```
df_Churn.Monthly_Charges.hist()
```

Out[453]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22fd16dec88>
```



Customers who churns are charged 400 to less than 2400 pesos a month. With less than 500 pesos monthly charge as the highest count of monthly charges, this amount can be considered as "small amount" and can be considered as something that they could also overlook when they decided to churn. 1,500 - 2,000 monthly charges can be considered also as payment of services packages so others usually churn when something about the service is not favorable to them, or also the monthly charge for them is considered high for them that they tend to delay its payment, leading to churn from it.

In [454]:
```
df_Churn.Gender.value_counts()
```

Out[454]:
```
Male      226
Female    221
Name: Gender, dtype: int64
```

With a very little gap, Males are actually more prone to churn, with 50.8% than the Female's 49.2%. Considering this, those who churn are regardless of gender, and to think of today where everyone has an access to internet thru their mobile surfing plans, they find the latter more practical since it's on prepaid rather than the fixed monthly charges.

In [455]:
```
df_Churn.Internet_Service.value_counts()
```

Out[455]:
```
Fiber    192
DSL      151
No       104
Name: Internet_Service, dtype: int64
```

Those who churn are those with Internet Services. Maybe there's something they found unfavorable with the connection or service that they would churn, regardless of having that internet service monthly charge. They may have found the data allotment and distribution so little, the signal strength being weak in their area and other related problems that they may resort to finding other internet service provider.

Loyal (Not-Churning) Customers

In [456]:
```
df_Not_Churn.head()
```
Out[456]:

	Customer_ID	Region	Gender	Phone_Service	Internet_Service	Dominant_Payment_Method
0	0002-ORFBO	North Luzon	Female	Yes	DSL	In-person
6	0013-SMEOE	North Luzon	Female	Yes	Fiber	Bank transfer
7	0014-BMAQU	North Luzon	Male	Yes	Fiber	Credit Card
24	0031-PVLZI	National Capital Region	Female	Yes	No	In-person
30	0048-LUMLS	National Capital Region	Male	Yes	Fiber	Credit Card

In [457]:
```
df_Not_Churn.shape
```
Out[457]: (1571, 16)

In [458]:
```
df_Not_Churn.describe()
```
Out[458]:

	Dominant_Payment_Method_In-person	Tenure	Customer_Service_Calls	Online_Security	Locked_
count	1571.00	1571.00	1571.00	1571.00	1571.
mean	0.43	35.01	0.98	0.44	0.
std	0.50	24.36	0.71	0.50	0.
min	0.00	0.00	0.00	0.00	0.
25%	0.00	12.00	0.00	0.00	0.
50%	0.00	33.00	1.00	0.00	1.
75%	1.00	58.00	1.00	1.00	1.
max	1.00	72.00	3.00	1.00	1.

In [459]:
```
df_Not_Churn.Region.value_counts()
```
Out[459]:
```
National Capital Region    505
North Luzon                399
Visayas                    320
South Luzon                203
Mindanao                   144
Name: Region, dtype: int64
```

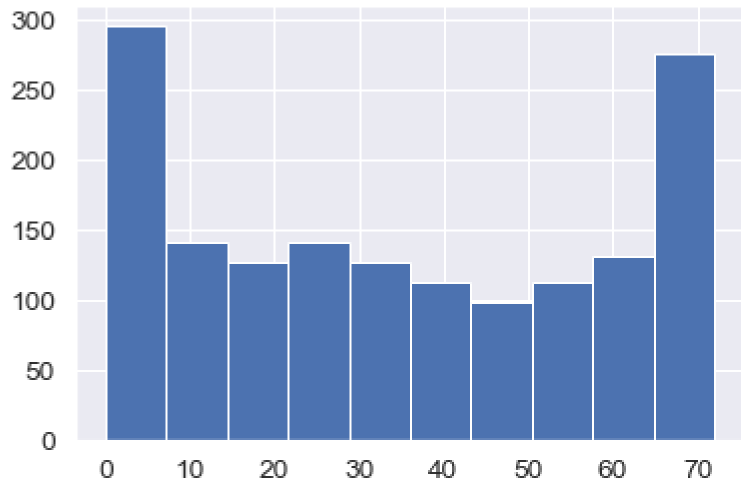
North Luzon and *National Capital Region* are the top regions also not having Customer Churns, with Luzon Area getting more than half of the total number.

In [460]:
```
df_Not_Churn.Tenure.value_counts()
```

```
Out[460]: 1     116
72    100
3      40
2      39
4      36
71     35
70     29
10     28
65     28
64     28
17     26
24     25
69     25
8      24
35     24
11     23
22     23
56     23
6      23
68     22
25     22
7      22
23     21
67     21
14     20
63     20
51     19
47     19
31     19
34     19
60     19
29     19
20     19
43     19
62     18
52     18
9      18
15     18
33     18
16     18
5      18
46     18
26     18
28     17
58     17
66     17
41     17
42     17
50     17
27     16
61     16
37     16
21     16
18     16
32     16
13     15
48     15
19     15
54     15
55     14
12     14
59     14
57     13
30     13
45     13
53     12
40     12
38     12
39     11
36      9
49      9
44      8
0       2
Name: Tenure, dtype: int64
```

In [461]:
```
df_Not_Churn.Tenure.hist()
```

Out[461]: <matplotlib.axes._subplots.AxesSubplot at 0x22fd193ef28>



Most customers doesn't churn from the 1st month,as they would consider it as their experimental month, and possibility of reaching it 6 years and beyond to continue liking the service. It is also to be noted that the more longer the subscriber is bound to the service, the less likely they will Churn.It is also to be take note of that as the month goes longer, it also shows that availing the services flows steadily through the years.

In [462]:
```
df_Not_Churn.Customer_Service_Calls.value_counts()
```

Out[462]:
```
1    776
0    413
2    380
3      2
Name: Customer_Service_Calls, dtype: int64
```

Loyal Customers mostly takes 1-2 calls or no calls at all in Customer Service in order to raise questions, have complaints or ask for help in their service / device. Contrasting to at most 7 calls of the Customers who churned, most likely that these loyal customers may have solve their problems or seem happy with the service that less calls were recorded. We could conclude that the less calls a customer took means less problems regarding with the service and therefore, satisfaction on the service availed.

In [463]:
```
df_Not_Churn.Online_Security.value_counts()
```

Out[463]:
```
0    874
1    697
Name: Online_Security, dtype: int64
```

55.6% of the Loyal Customers doesn't have the Online Security Package but it's also taken note that unlike the Churned Customers, there's no huge gap here between those who have Online Security but those who haven't. It means that the other half was actually loyal customers for having the Online Security package also.

In [464]:
```
df_Not_Churn.Locked_In.value_counts()
```

Out[464]:
```
1    788
0    783
Name: Locked_In, dtype: int64
```

50.2% of the Customers who haven't churned also avail the Locked In penalty fee or it is included in their contract. Same explanation applies where because of the Locked In fee that others continue to avail the services. With this, the availment of other services can be considered a way to reduce the churn and retain the customers. Also, there is no huge gap of percentage between those who does not have a Locked In.

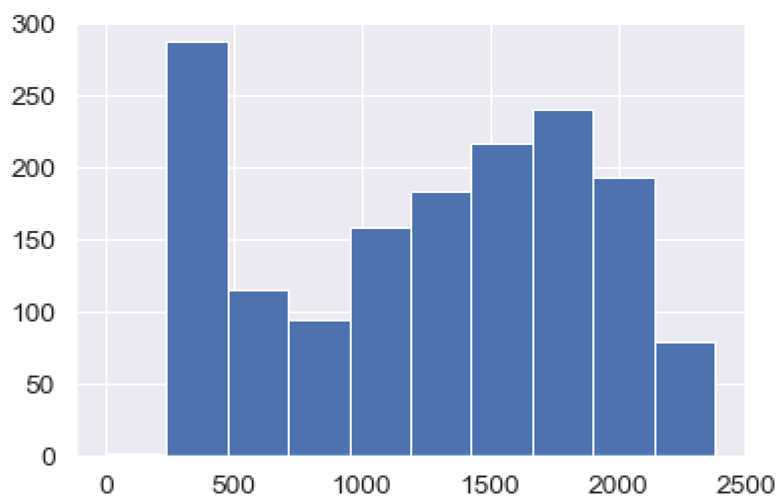
In [465]:
```
df_Not_Churn["Dominant_Payment_Method_In-person"].value_counts()
```

Out[465]:
```
0    890
1    681
Name: Dominant_Payment_Method_In-person, dtype: int64
```

Almost 56.7% of the Customers who continued the services are not paying in-person, which make sense for us Filipinos since we don't have to worry over paying in-person for the account will just be transferred to the bank or just with the credit card. With the ease of convenience nowadays, most would like to just instantly pay bills and therefore making the usage of the services stay.

In [466]:
```
df_Not_Churn.Monthly_Charges.hist()
```

Out[466]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22fd1983588>
```



In [467]:

```
df_Not_Churn.Monthly_Charges.value_counts()
```

```
Out[467]: 400     93
          390     83
          410     63
          1400    27
          500     24
          1410    24
          380     24
          1610    23
          1600    22
          1700    21
          1710    20
          510     18
          1790    18
          1800    17
          1490    17
          1690    16
          1590    16
          900     16
          910     16
          1010    16
          2080    16
          1020    16
          2000    16
          420     16
          1390    15
          1000    14
          490     14
          1100    14
          1890    14
          1820    14
          1900    14
          1620    13
          480     13
          1380    13
          1500    13
          1720    12
          2090    12
          1920    12
          1990    12
          1580    12
          1910    12
          2110    11
          1980    11
          980     11
          1470    11
          2010    11
          1210    11
          1680    11
          1810    11
          520     10
                  ..
          1160     3
          1150     3
          1140     3
          1330     3
          1340     3
          1430     3
          1440     3
          1450     3
          870      3
          770      3
          1870     3
          2350     3
          370      2
          2160     2
          2310     2
          1250     2
         -1        2
          570      2
          840      2
          610      2
          1550     2
          930      2
          580      2
```

```
790     2
820     2
1060    2
1360    2
430     2
1850    2
1260    2
950     1
1950    1
1270    1
830     1
1050    1
2380    1
740     1
730     1
2370    1
670     1
1240    1
2240    1
2250    1
2260    1
2290    1
540     1
460     1
2330    1
2360    1
680     1
Name: Monthly_Charges, Length: 187, dtype: int64
```

Same with the customers who churned where charged 400 to less than 2400 pesos a month. With less than 500 pesos monthly charge as the highest count of monthly charges, this amount can be considered as "small amount" but unlike with the churned customers, it can be considered as something that they could just pay quickly.

In [468]:
```
df_Not_Churn.Gender.value_counts()
```

Out[468]:
```
Male      791
Female    780
Name: Gender, dtype: int64
```

With a very little gap, Almost 50.4% Males are actually also more prone not to churn, with 50.8% than the Female's 49.4%.

In [469]:
```
df_Not_Churn.Internet_Service.value_counts()
```

Out[469]:
```
Fiber    636
DSL      525
No       410
Name: Internet_Service, dtype: int64
```

Most of those who never churned have been bounded by their internet service such as Fiber and DSL. With this, we could also concluded that those who also avail the Internet services also get to retain to avail the services.