In [34]:
```
#importing the basic libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import missingno as mano
%matplotlib inline
from scipy.stats import norm
from sklearn import preprocessing
```

In [35]:
```
#read the Sales CSV file
churndf = pd.read_csv("churndata.csv")
```

In [36]:
```
churndf.head(10)
```

Out[36]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Inter
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	
4	9237-HQITU	Female	0	No	No	2	Yes	No	
5	9305-CDSKC	Female	0	No	No	8	Yes	Yes	
6	1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	
7	6713-OKOMC	Female	0	No	No	10	No	No phone service	
8	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	
9	6388-TABGU	Male	0	No	Yes	62	Yes	No	

10 rows × 21 columns

In [37]:
```
# Let's see if they are many missing values we have
churndf.isnull().sum()
```

Out[37]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

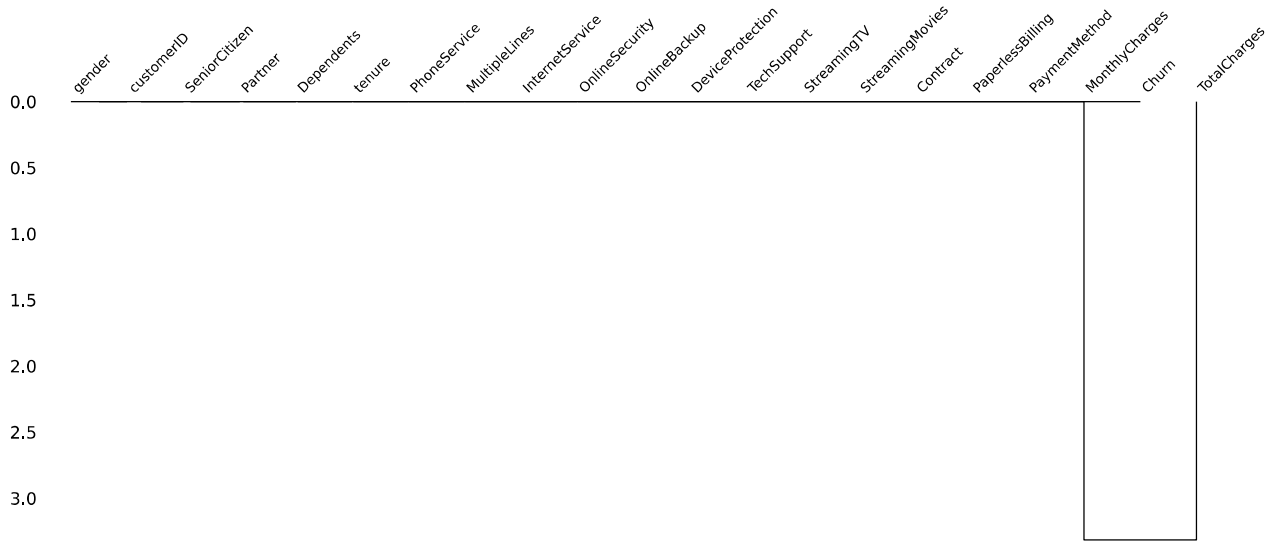
In [38]:
```
#Making a list of missing value types because the above list does not show us a clear v

missing_values = ["n/a", "na", "-", " "]
df = pd.read_csv("churndata.csv", na_values = missing_values)
df.isnull().sum()
```

Out[38]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       11
Churn               0
dtype: int64
```

In [39]:
```
mano.dendrogram(df)
```

Out[39]:
```
<AxesSubplot:>
```



In [40]:
```
#Dropping values of total charges that are null since its less than 1%
df = df[df["TotalCharges"].notnull()]
df = df.reset_index()[df.columns]
```

In [41]:
```
print ("\nUnique values :  \n",churndf.nunique())
```

```
Unique values :
 customerID          7043
gender                 2
SeniorCitizen          2
Partner                2
Dependents             2
tenure                73
PhoneService           2
MultipleLines          3
InternetService        3
OnlineSecurity         3
OnlineBackup           3
DeviceProtection       3
TechSupport            3
StreamingTV            3
StreamingMovies        3
Contract               3
PaperlessBilling       2
PaymentMethod          4
MonthlyCharges      1585
TotalCharges        6531
Churn                  2
dtype: int64
```

In [42]:
```
#To check whether they are any duplicated values in customer_id
bool_custid = not churndf["customerID"].is_unique      # True
bool_custid = churndf['customerID'].duplicated().any() # True
```

In [43]:
```
#We see a lot of values which contain 3 unique values in columns like which contain Yes
#OnlineSecurity         3
#OnlineBackup           3
#DeviceProtection       3
#TechSupport            3
```

```
#StreamingTV            3
#StreamingMovies        3

#Hence for BI analysis NO and no internet service can be categorized as the same
```

In [44]:
```
#replace 'No internet service' to No for the following columns
replace_cols = [ 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
                 'TechSupport','StreamingTV', 'StreamingMovies']
for i in replace_cols :
    df[i]  = df[i].replace({'No internet service' : 'No'})
```

In [45]:
```
#Convert Senior citizen into objects that will allow us a better comparison
df["SeniorCitizen"] = df["SeniorCitizen"].replace({1:"Yes",0:"No"})
```

In [46]:
```
df.head(10)
```

Out[46]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Inter
0	7590-VHVEG	Female	No	Yes	No	1	No	No phone service	
1	5575-GNVDE	Male	No	No	No	34	Yes	No	
2	3668-QPYBK	Male	No	No	No	2	Yes	No	
3	7795-CFOCW	Male	No	No	No	45	No	No phone service	
4	9237-HQITU	Female	No	No	No	2	Yes	No	
5	9305-CDSKC	Female	No	No	No	8	Yes	Yes	
6	1452-KIOVK	Male	No	No	Yes	22	Yes	Yes	
7	6713-OKOMC	Female	No	No	No	10	No	No phone service	
8	7892-POOKP	Female	No	Yes	No	28	Yes	Yes	
9	6388-TABGU	Male	No	No	Yes	62	Yes	No	

10 rows × 21 columns

Statistical Testing

In [47]:
```
# Importing the statistics module
import pandas as pd
from scipy import stats
from statistics import mode
import statsmodels.api as sm
from statsmodels.formula.api import ols
from scipy.stats import chi2_contingency
from scipy.stats import chi2
```

In [48]:
```
df.dtypes
```

Out[48]:
```
customerID           object
gender               object
SeniorCitizen        object
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges        float64
Churn                object
dtype: object
```

In [49]:
```
#Since we already from our data manipulation that all six columns below is directly con

#OnlineSecurity
#OnlineBackup
#DeviceProtection
#TechSupport
#StreamingTV
#StreamingMovies
```

Chi Squared Method

The reason that we have used Chi Squared Method for testing because most of our data is in object form and this method allows to statistically analyze the dependencies

In [50]:
```
data_crosstab = pd.crosstab(df['OnlineSecurity'],df['InternetService'],
margins = False)
print(data_crosstab)


stat, p, dof, expected = chi2_contingency(data_crosstab)
```

```
print('dof=%d' % dof)
print(expected)

# interpret p-value
alpha = 0.05
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
InternetService   DSL  Fiber optic    No
OnlineSecurity
No               1240         2257  1520
Yes              1176          839     0
dof=2
[[1723.70193402 2208.84982935 1084.44823663]
 [ 692.29806598  887.15017065  435.55176337]]
significance=0.050, p=0.000
Dependent (reject H0)
```

In [51]:

```
data_crosstab = pd.crosstab(df['DeviceProtection'],df['InternetService'],
margins = False)
print(data_crosstab)


stat, p, dof, expected = chi2_contingency(data_crosstab)
print('dof=%d' % dof)
print(expected)

# interpret p-value
alpha = 0.05
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
InternetService    DSL  Fiber optic    No
DeviceProtection
No                1355         1739  1520
Yes               1061         1357     0
dof=2
[[1585.24232082 2031.41979522  997.33788396]
 [ 830.75767918 1064.58020478  522.66211604]]
significance=0.050, p=0.000
Dependent (reject H0)
```

In [52]:

```
data_crosstab = pd.crosstab(df['StreamingTV'],df['InternetService'],
margins = False)
print(data_crosstab)


stat, p, dof, expected = chi2_contingency(data_crosstab)
print('dof=%d' % dof)
print(expected)

# interpret p-value
alpha = 0.05
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
```

```
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
InternetService   DSL  Fiber optic    No
StreamingTV
No               1463         1346  1520
Yes               953         1750     0
dof=2
[[1487.32423208 1905.94197952  935.7337884 ]
 [ 928.67576792 1190.05802048  584.2662116 ]]
significance=0.050, p=0.000
Dependent (reject H0)
```

Our preliminary testing confirms our assumption that all online services our directly relate to the internet service

Now lets check that do other services depend on each other or not

In [53]:

```
data_crosstab = pd.crosstab(df['StreamingTV'],df['DeviceProtection'],
margins = False)
print(data_crosstab)


stat, p, dof, expected = chi2_contingency(data_crosstab)
print('dof=%d' % dof)
print(expected)

# interpret p-value
alpha = 0.05
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
DeviceProtection    No   Yes
StreamingTV
No                3474   855
Yes               1140  1563
dof=1
[[2840.44453925 1488.55546075]
 [1773.55546075  929.44453925]]
significance=0.050, p=0.000
Dependent (reject H0)
```

In [54]:

```
data_crosstab = pd.crosstab(df['TechSupport'],df['DeviceProtection'],
margins = False)
print(data_crosstab)


stat, p, dof, expected = chi2_contingency(data_crosstab)
print('dof=%d' % dof)
print(expected)

# interpret p-value
alpha = 0.05
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Dependent (reject H0)')
```

```
else:
    print('Independent (fail to reject H0)')
```

```
DeviceProtection    No   Yes
TechSupport
No                3780  1212
Yes                834  1206
dof=1
[[3275.46757679 1716.53242321]
 [1338.53242321  701.46757679]]
significance=0.050, p=0.000
Dependent (reject H0)
```

In [55]:

```
data_crosstab = pd.crosstab(df['TechSupport'],df['OnlineBackup'],
margins = False)
print(data_crosstab)


stat, p, dof, expected = chi2_contingency(data_crosstab)
print('dof=%d' % dof)
print(expected)

# interpret p-value
alpha = 0.05
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
OnlineBackup    No   Yes
TechSupport
No            3716  1276
Yes            891  1149
dof=1
[[3270.49829352 1721.50170648]
 [1336.50170648  703.49829352]]
significance=0.050, p=0.000
Dependent (reject H0)
```

In [ ]: