Assignment

Use the "from the expert" (FTE) jupyter notebook as a starter for this assignment, and ask your instructor questions if you need help.

Use the `churn_data.csv` file to carry out a similar data cleaning and preparation as we did in the FTE. Specifically, at least complete these minimum requirements:

- Check for outliers in numeric data, and deal with them if needed
- Check for missing values, and decide how to deal with them if needed
- Convert categorical columns to numeric values
- Create at least one new feature by combining multiple columns. For example, you could calculate the ratio of total charges to tenure. Create at least one plot for your new feature.
- Save the data to a csv (or another filetype of your choice) for use next week.
- Write a short analysis at the end of the notebook describing your findings and what you did.

You can do more data cleaning, preparation, and EDA beyond these basic requirements if you want to learn more and develop your data science skills. For example, you could use a box-cox transformation on the numeric data or try other outlier methods.

DS process status

Here is our data science process, and where we are (#3):

1. Business understanding

Can we use machine learning to predict if a customer will churn before they leave?

2. Data understanding

Done in week 1 (mostly), this is iterative so you might do more of this as we go on.

3. Data preparation

We are here this week.

4. Modeling

Next week

5. Evaluation

Next week

6. Deployment

Next week

In [1]:
```
import pandas as pd
```

```
from pandas_profiling import ProfileReport
import matplotlib.pyplot as plt

%matplotlib inline
import phik
import seaborn as sns
import numpy as np
```

Business Understanding

In [2]:
```
df = pd.read_csv('data\churn_data.csv', index_col='customerID')
df
```

Out[2]:

customerID	tenure	PhoneService	Contract	PaymentMethod	MonthlyCharges	TotalCharges	Churn
7590-VHVEG	1	No	Month-to-month	Electronic check	29.85	29.85	No
5575-GNVDE	34	Yes	One year	Mailed check	56.95	1889.50	No
3668-QPYBK	2	Yes	Month-to-month	Mailed check	53.85	108.15	Yes
7795-CFOCW	45	No	One year	Bank transfer (automatic)	42.30	1840.75	No
9237-HQITU	2	Yes	Month-to-month	Electronic check	70.70	151.65	Yes
...	...	...	...	...	...	...	...
6840-RESVB	24	Yes	One year	Mailed check	84.80	1990.50	No
2234-XADUH	72	Yes	One year	Credit card (automatic)	103.20	7362.90	No
4801-JZAZL	11	No	Month-to-month	Electronic check	29.60	346.45	No
8361-LTMKD	4	Yes	Month-to-month	Mailed check	74.40	306.60	Yes
3186-AJIEK	66	Yes	Two year	Bank transfer (automatic)	105.65	6844.50	No

7043 rows × 7 columns

In [3]:
```
df.describe()
```

Out[3]:

	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7032.000000
mean	32.371149	64.761692	2283.300441
std	24.559481	30.090047	2266.771362
min	0.000000	18.250000	18.800000
25%	9.000000	35.500000	401.450000
50%	29.000000	70.350000	1397.475000
75%	55.000000	89.850000	3794.737500
max	72.000000	118.750000	8684.800000

Data Understanding (Week 2, Phase 1)

EDA

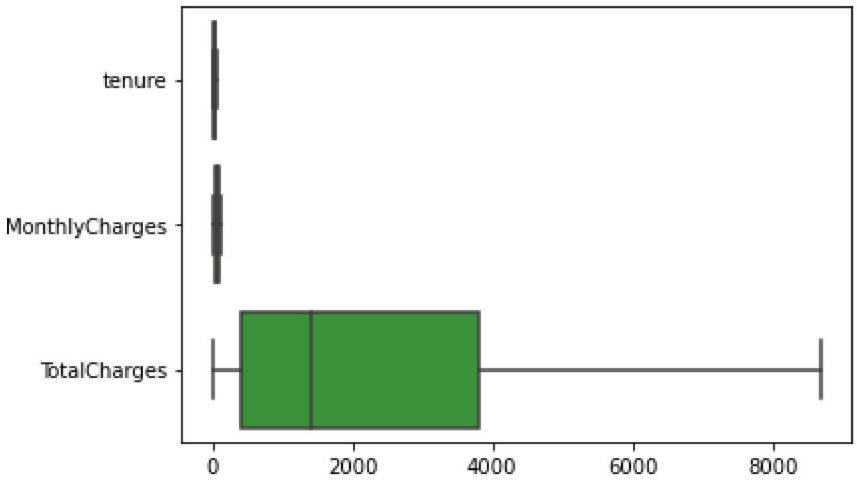
In [4]:
```
report = ProfileReport(df)
report.to_file('churn_eda.html')
```

Data Preparation (Week 2, Phase 1)

Outliers?

In [5]:
```
sns.boxplot(data=df, orient="h",whis=1.5)
```

Out[5]:
```
<AxesSubplot:>
```



Skewed results but the boxplot is not indicating any outliers.

In [6]:
```
df = pd.read_csv('data\churn_data.csv', index_col='customerID')
```

Missing Values?

Dropped the missing values

In [7]:
```
df.isna().sum()
```

Out[7]:
```
tenure             0
PhoneService       0
Contract           0
PaymentMethod      0
MonthlyCharges     0
TotalCharges      11
Churn              0
dtype: int64
```

In [8]:
```
df.dropna(inplace=True)
```

Looking for outliers again (Advanced)

In [9]:
```
numeric_df = df.select_dtypes(exclude=['object'])
```

In [10]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_numeric = scaler.fit_transform(numeric_df)
```

In [11]:
```
from pyod.models.knn import KNN

od = KNN(contamination=0.01)
od.fit(scaled_numeric)
```

Out[11]:
```
KNN(algorithm='auto', contamination=0.01, leaf_size=30, method='largest',
  metric='minkowski', metric_params=None, n_jobs=1, n_neighbors=5, p=2,
  radius=1.0)
```

In [12]:
```
outliers = od.predict(scaled_numeric)
outliers
```

Out[12]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [13]:
```
outliers.sum()
```

Out[13]:
```
49
```

In [14]:
```
df0 = df[outliers.astype('bool')]
```

In [15]:
```
df = pd.concat([df, df0])
```

In [16]:
```
df = df.drop_duplicates(keep=False)
```

I removed the outliers.

Data Understanding (Week 2, Phase 2)

All further data has missing values removed

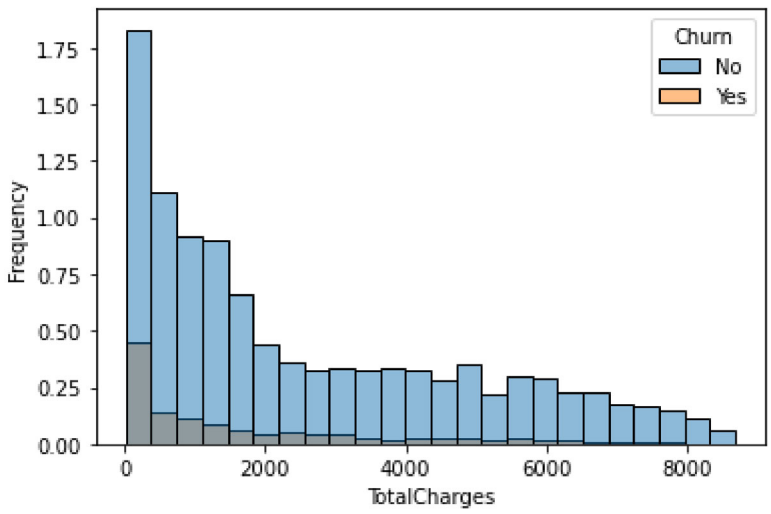
In [17]:
```
df.describe()
```

Out[17]:

	tenure	MonthlyCharges	TotalCharges
count	6793.000000	6793.000000	6793.000000
mean	33.231709	65.309348	2341.866061
std	24.335525	29.944159	2268.883532
min	1.000000	18.250000	18.800000
25%	10.000000	39.100000	454.000000
50%	30.000000	70.800000	1445.200000
75%	56.000000	90.100000	3882.300000
max	72.000000	118.750000	8684.800000

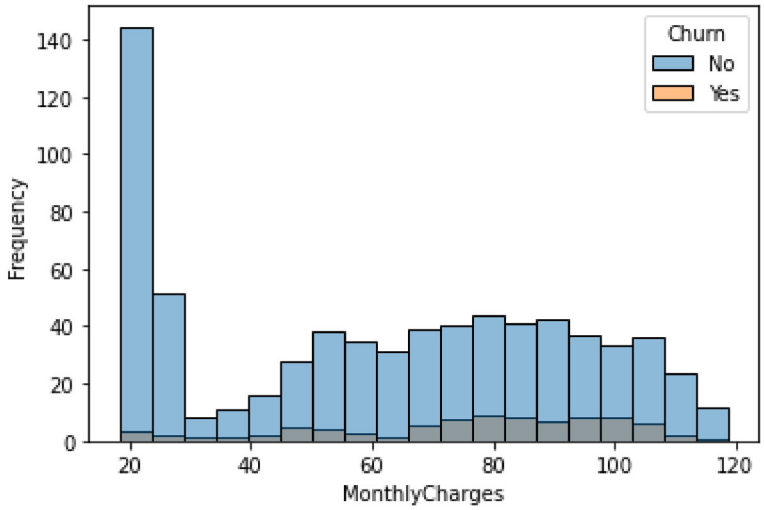
In [18]:
```
sns.histplot(data=df, x='TotalCharges', hue='Churn', stat='frequency')
```

Out[18]:
```
<AxesSubplot:xlabel='TotalCharges', ylabel='Frequency'>
```



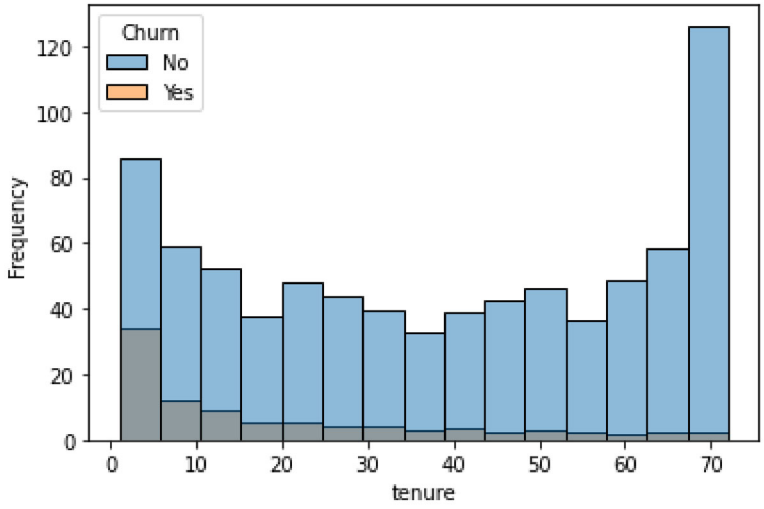
In [19]:
```
sns.histplot(data=df, x='MonthlyCharges', hue='Churn', stat='frequency')
```

Out[19]:
```
<AxesSubplot:xlabel='MonthlyCharges', ylabel='Frequency'>
```



In [20]:
```
sns.histplot(data=df, x='tenure', hue='Churn', stat='frequency')
```

Out[20]:
```
<AxesSubplot:xlabel='tenure', ylabel='Frequency'>
```



Data Preparation (Week 2, Phase 2)

Converting categoral columns to numerical values

In [21]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Churn'] = le.fit_transform(df['Churn'])
```

In [22]:
```
df = pd.concat([df.drop('PhoneService', axis=1), pd.get_dummies(df['PhoneService'],pref
```

In [23]:
```
df = pd.concat([df.drop('Contract', axis=1), pd.get_dummies(df['Contract'])],axis=1)
```

In [24]:
```
df = pd.concat([df.drop('PaymentMethod', axis=1), pd.get_dummies(df['PaymentMethod'])],
```

In [25]:
```
df
```

Out[25]:

customerID	tenure	MonthlyCharges	TotalCharges	Churn	PhoneService_Yes	Month-to-month	One year	Two year	(aut
7590-VHVEG	1	29.85	29.85	0	0	1	0	0	
5575-GNVDE	34	56.95	1889.50	0	1	0	1	0	
3668-QPYBK	2	53.85	108.15	1	1	1	0	0	
7795-CFOCW	45	42.30	1840.75	0	0	0	1	0	
9237-HQITU	2	70.70	151.65	1	1	1	0	0	
...	...	...	...	...	...	...	...	...	
6840-RESVB	24	84.80	1990.50	0	1	0	1	0	
2234-XADUH	72	103.20	7362.90	0	1	0	1	0	
4801-JZAZL	11	29.60	346.45	0	0	1	0	0	
8361-LTMKD	4	74.40	306.60	1	1	1	0	0	
3186-AJIEK	66	105.65	6844.50	0	1	0	0	1	

6793 rows × 12 columns

report = ProfileReport(df) report.to_file('churn_prepped_eda.html')

Assumption 1: tenure and monthly charges are functions of the total charges of a customer

Converting Functions (DV) and IV to Logarithmic Values

In [26]:
```
df_log = df.copy()
df_log['TotalCharges'] = np.log(df_log['TotalCharges'])
df_log['MonthlyCharges'] = np.log(df_log['MonthlyCharges'])
df_log['tenure'] = np.log(df_log['tenure'])
```

Creating new values from ratios

Let's create a ratio of TotalCharges to MonthlyCharges and TotalCharges to Tenure.

In [27]:
```
df['TotalCharges_MonthlyCharges_ratio'] = df['TotalCharges'] / df['MonthlyCharges']
```

```
df['AverageMonthlyCharges'] = df['TotalCharges'] / df['tenure']
df['MonthlyCharges_tenure_Ratio'] = df['MonthlyCharges'] / df['tenure']
```

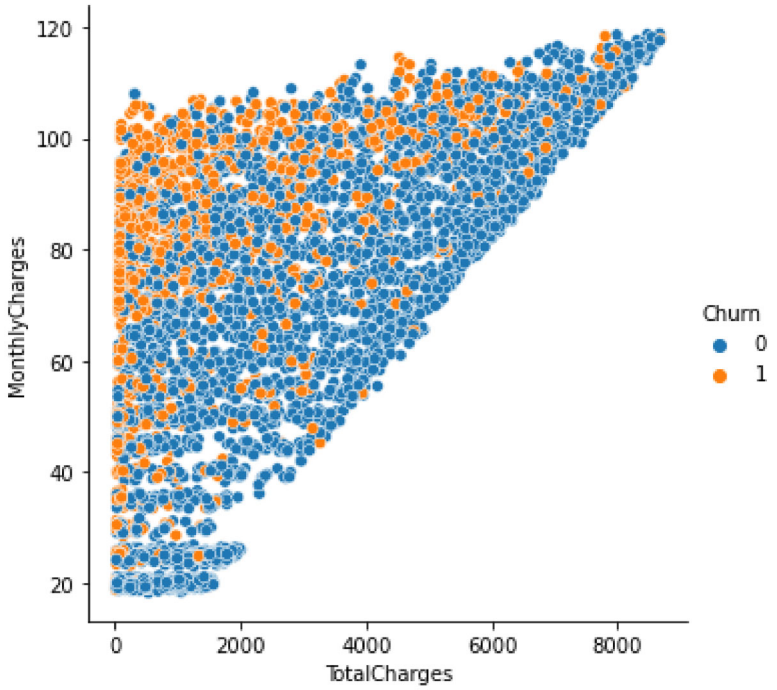
Data Understanding (Week 2, Phase 3)

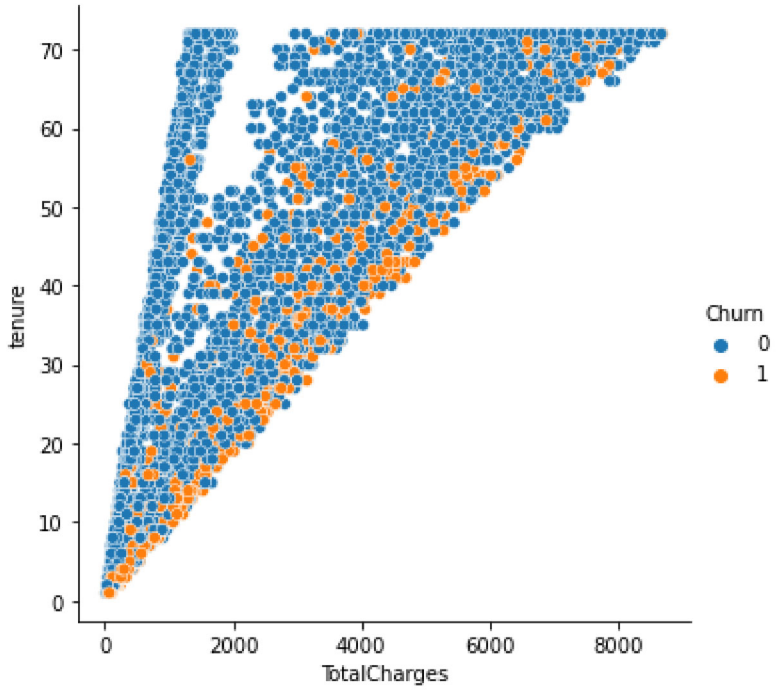
Assumption 1 Confirmed by Correlations to Logaritmic Values

Redistribution

In [28]:
```
sns.relplot(data=df, x='TotalCharges', y='MonthlyCharges', hue='Churn')
sns.relplot(data=df, x='TotalCharges', y='tenure', hue='Churn')
```

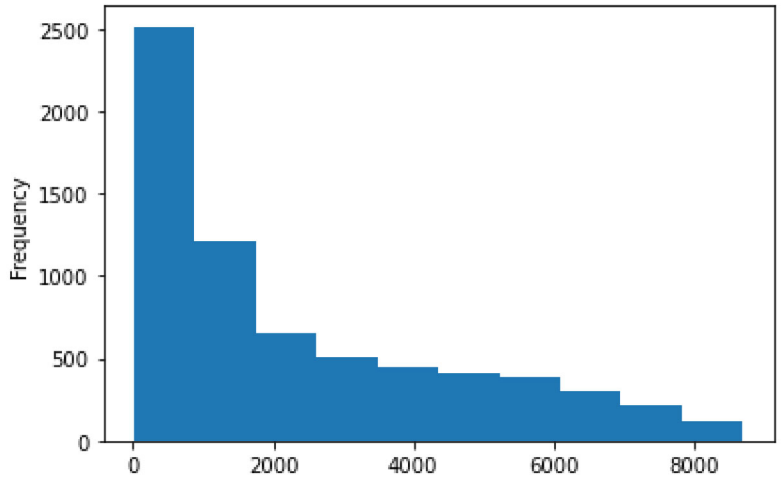
Out[28]: <seaborn.axisgrid.FacetGrid at 0x29020e810d0>





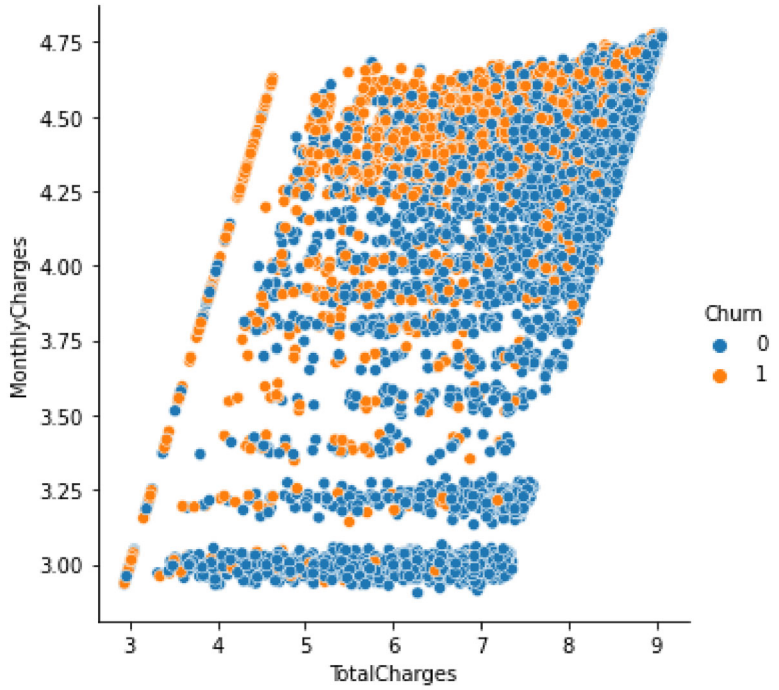
In [29]:
```
df['TotalCharges'].plot.hist()
```

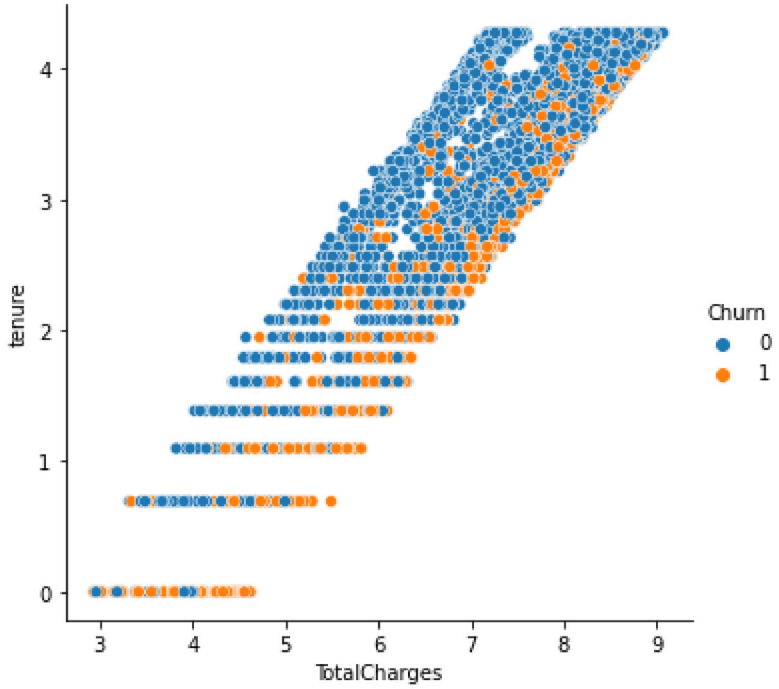
Out[29]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [30]:
```
sns.relplot(data=df_log, x='TotalCharges', y='MonthlyCharges', hue='Churn')
sns.relplot(data=df_log, x='TotalCharges', y='tenure', hue='Churn')
```

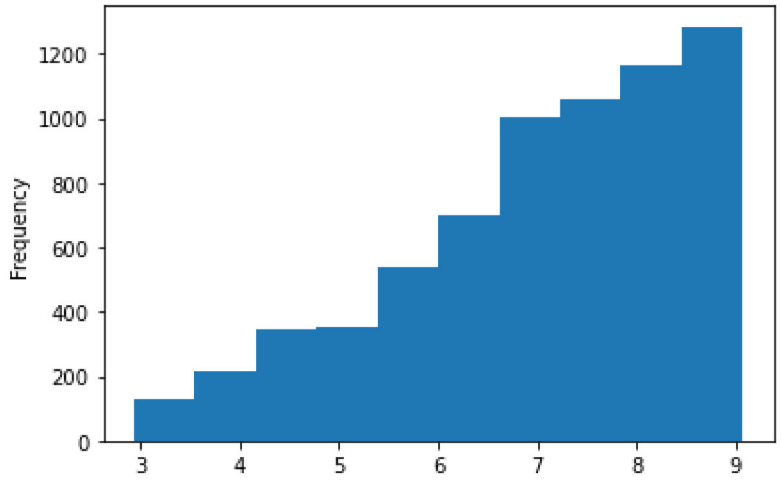
Out[30]:
```
<seaborn.axisgrid.FacetGrid at 0x2901d82e160>
```





In [31]:
```
df_log['TotalCharges'].plot.hist()
```

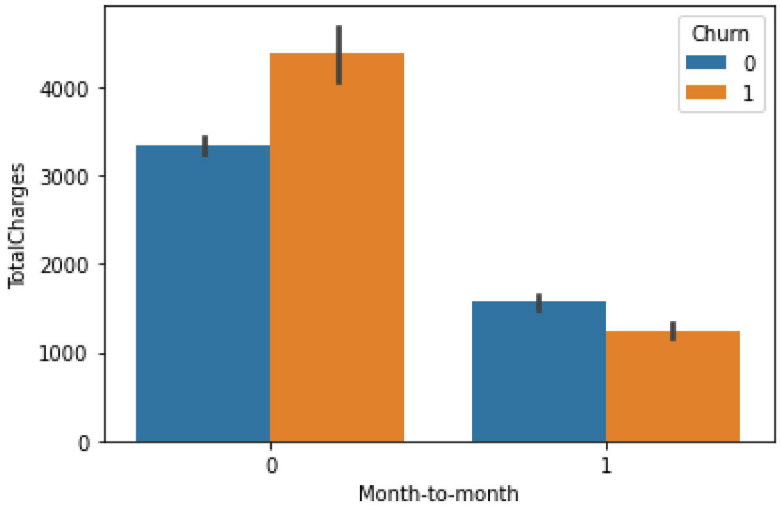
Out[31]:
```
<AxesSubplot:ylabel='Frequency'>
```



Assumption 1a: Customers with long_term contracts with higher total charges tend to leave

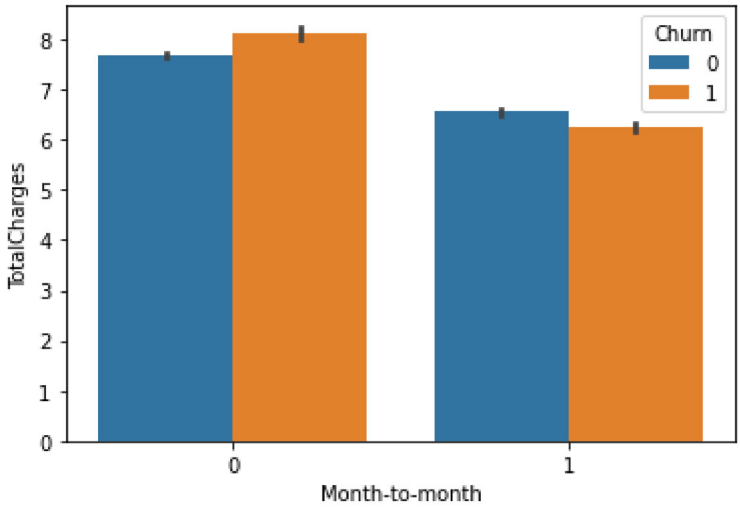
In [32]:
```
sns.barplot(data=df, x='Month-to-month', y='TotalCharges', hue='Churn')
```

Out[32]:
```
<AxesSubplot:xlabel='Month-to-month', ylabel='TotalCharges'>
```



In [33]:
```
sns.barplot(data=df_log, x='Month-to-month', y='TotalCharges', hue='Churn')
```

Out[33]:
```
<AxesSubplot:xlabel='Month-to-month', ylabel='TotalCharges'>
```

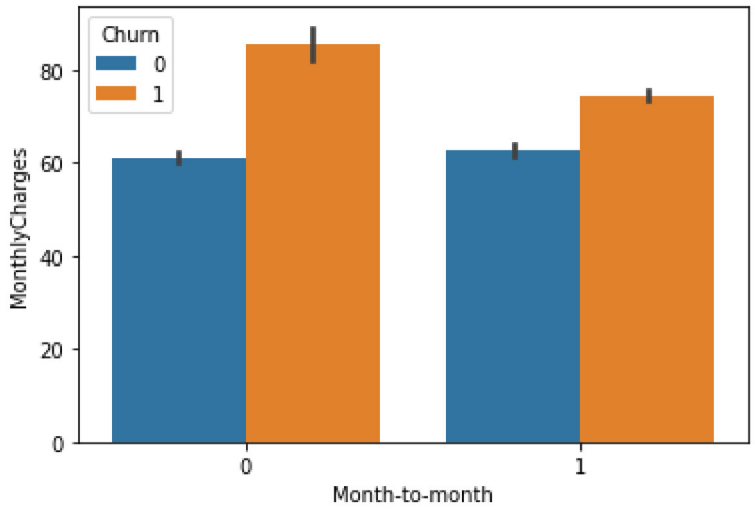


We also found last week that customers with long-term contracts who paid greater than 60 dollars a month tended to stay in contrast to customers in month-to-month contracts who are more likely to churn.

Assumption 1b: Customers with greater monthly charges only keep their accounts because they are in a contract

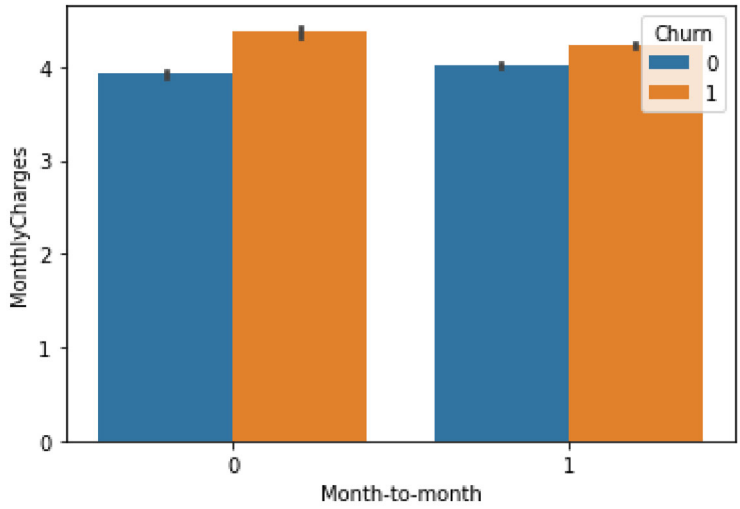
In [34]:
```
sns.barplot(data=df, x='Month-to-month', y='MonthlyCharges', hue='Churn')
```

Out[34]:
```
<AxesSubplot:xlabel='Month-to-month', ylabel='MonthlyCharges'>
```



In [35]:
```
sns.barplot(data=df_log, x='Month-to-month', y='MonthlyCharges', hue='Churn')
```

Out[35]:
```
<AxesSubplot:xlabel='Month-to-month', ylabel='MonthlyCharges'>
```



Assumption 1 Confirmed with Ratios

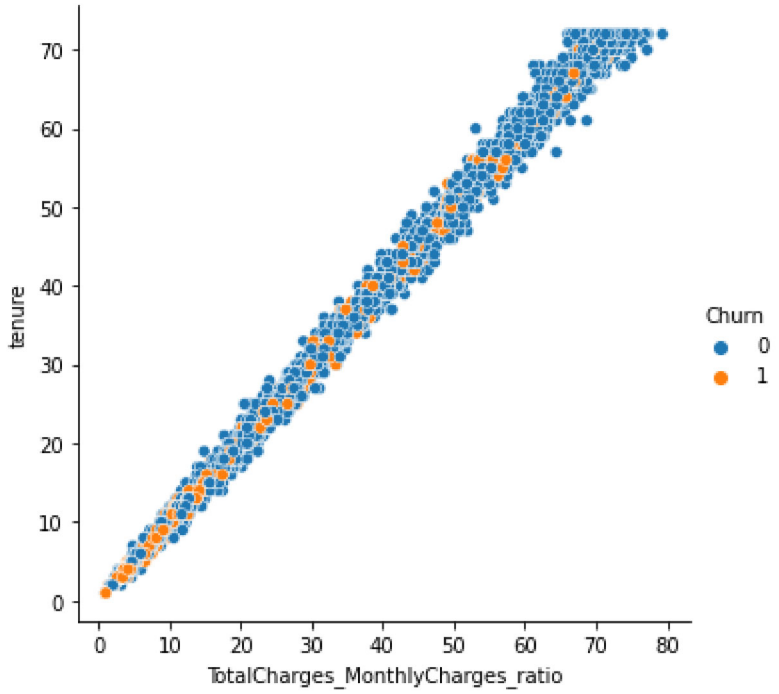
Redistribution

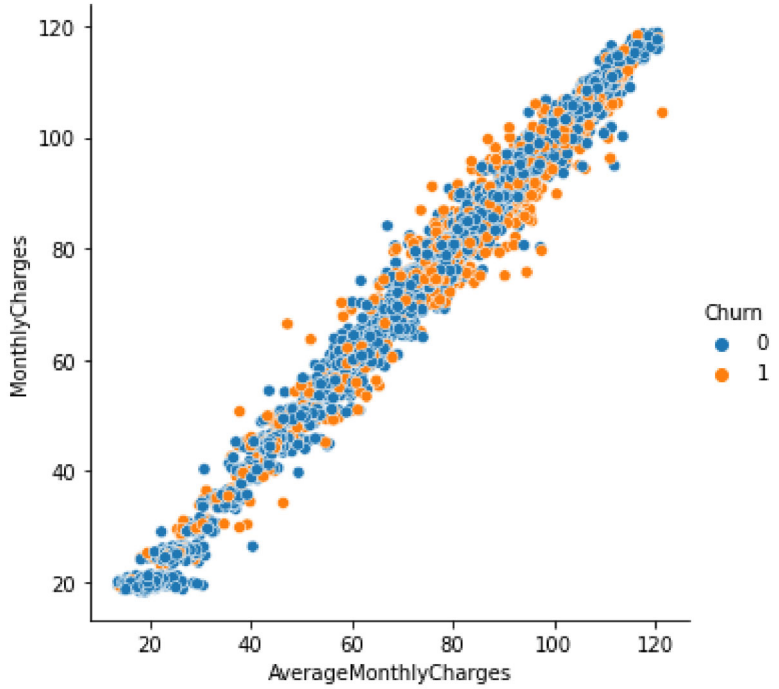
In [36]:
```
report = ProfileReport(df)
report.to_file('churn_ratio_eda.html')
```

As expected TotalCharges divided by tenure would give me the over all average monthly charges and there is a strong positive correlation between the created variable and monthly charges. Also Total charges divided by the monthly charges would give me the amount of time they are with the company, also confirmed by the correlation between TotalCharges_MonthlyCharges_ratio and the tenure.

In [37]:
```
sns.relplot(data=df, x='TotalCharges_MonthlyCharges_ratio', y='tenure', hue='Churn')
sns.relplot(data=df, x='AverageMonthlyCharges', y='MonthlyCharges', hue='Churn')
```

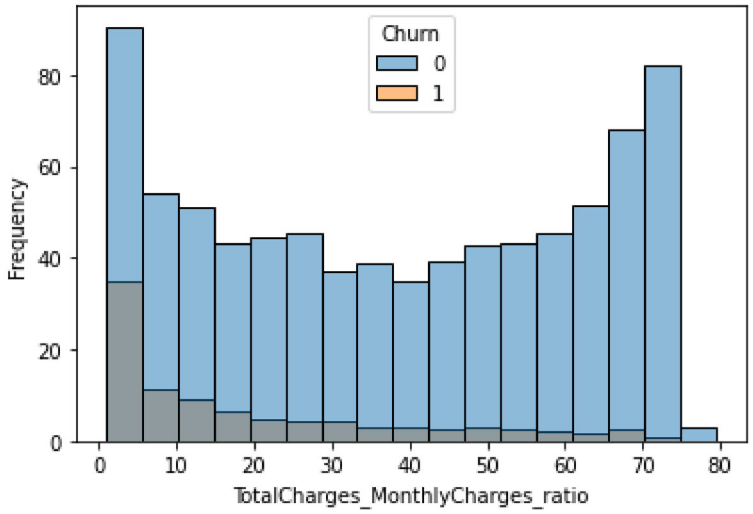
Out[37]:
```
<seaborn.axisgrid.FacetGrid at 0x29026552280>
```





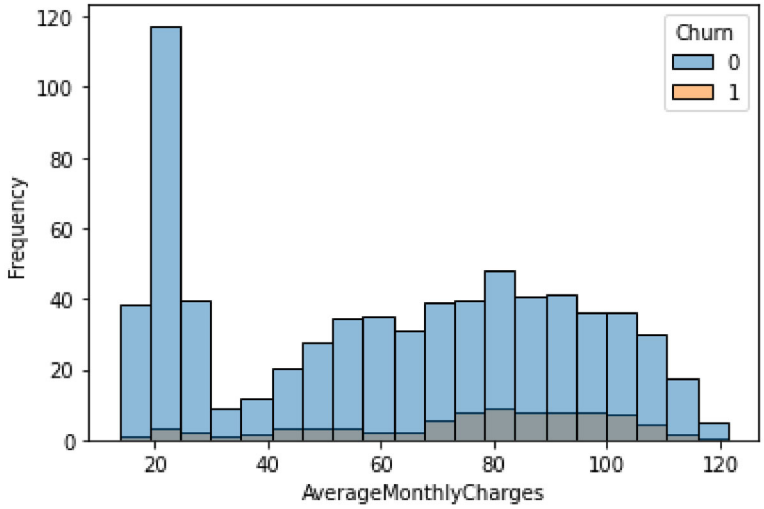
In [38]:
```
sns.histplot(data=df, x='TotalCharges_MonthlyCharges_ratio', hue='Churn', stat='frequen
```

Out[38]:
```
<AxesSubplot:xlabel='TotalCharges_MonthlyCharges_ratio', ylabel='Frequency'>
```



In [39]:
```
sns.histplot(data=df, x='AverageMonthlyCharges', hue='Churn', stat='frequency')
```

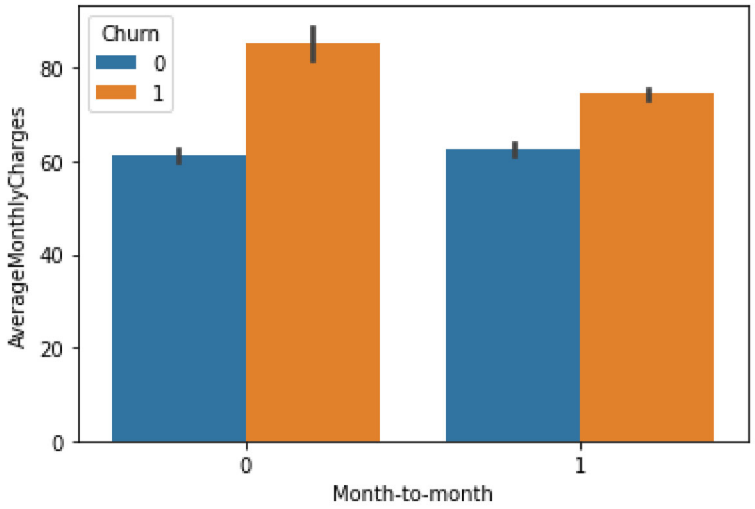
Out[39]:
```
<AxesSubplot:xlabel='AverageMonthlyCharges', ylabel='Frequency'>
```



Assumption 1a: Customers with long_term contracts with higher total charges tend to leave

In [40]:
```
sns.barplot(data=df, x='Month-to-month', y='AverageMonthlyCharges', hue='Churn')
```

Out[40]:
```
<AxesSubplot:xlabel='Month-to-month', ylabel='AverageMonthlyCharges'>
```



This relationship is the comparing the ratio of TotalCharges to Tenure. If the Total Charges reach a ratio of $60:1 month tenure the customer will churn no matter the contract type. Those in long-term contracts are still bound to their contract of one to two years so their ratios tend to reach greater maximums before churning.

Assumption 1b: Customers with greater monthly charges only keep their accounts because they are in a contract

The ratios Contract:tenure is almost equal to the ratio Contract:TotalCharges_MonthlyCharges_ratio showing me that the coeffcients are almost 1 to 1 and also indicate both correlations are slightly positive. Someone with a long-term contract will definitely accrue larger total charges and a longer tenure compared to someone in a month-to-month contract that can churn after every month. There is almost no correlation between contracts and monthly charges. This may be due to the lack of unique values available to the contracts (can only be 0, 1, or 2). The value counts of customers in each contract would give me a more accurate relationship of the two variables.

In [41]:
```
df.corr(method='pearson')
```

Out[41]:

	tenure	MonthlyCharges	TotalCharges	Churn	PhoneService_Ye
tenure	1.000000	0.235241	0.822276	-0.341031	0.02514
MonthlyCharges	0.235241	1.000000	0.651674	0.204150	0.25051
TotalCharges	0.822276	0.651674	1.000000	-0.183648	0.12121
Churn	-0.341031	0.204150	-0.183648	1.000000	0.00362
PhoneService_Yes	0.025148	0.250512	0.121215	0.003624	1.00000
Month-to-month	-0.639801	0.078691	-0.433141	0.397253	-0.01180
One year	0.187280	-0.005469	0.158070	-0.169092	0.00668
Two year	0.561471	-0.085749	0.350384	-0.298432	0.00729
Bank transfer (automatic)	0.233935	0.037041	0.177393	-0.110184	0.01313
Credit card (automatic)	0.221218	0.022070	0.173460	-0.125766	0.00057

	tenure	MonthlyCharges	TotalCharges	Churn	PhoneService_Ye
Electronic check	-0.210922	0.265361	-0.057535	0.295616	-0.00134
Mailed check	-0.217397	-0.361324	-0.287075	-0.099285	-0.01228
TotalCharges_MonthlyCharges_ratio	0.998890	0.234873	0.822561	-0.340693	0.02512
AverageMonthlyCharges	0.234418	0.996110	0.651322	0.203493	0.25025
MonthlyCharges_tenure_Ratio	-0.520729	0.066355	-0.377120	0.355828	0.05209

Save Dataframe

In [42]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6793 entries, 7590-VHVEG to 3186-AJIEK
Data columns (total 15 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   tenure                            6793 non-null   int64  
 1   MonthlyCharges                    6793 non-null   float64
 2   TotalCharges                      6793 non-null   float64
 3   Churn                             6793 non-null   int32  
 4   PhoneService_Yes                  6793 non-null   uint8  
 5   Month-to-month                    6793 non-null   uint8  
 6   One year                          6793 non-null   uint8  
 7   Two year                          6793 non-null   uint8  
 8   Bank transfer (automatic)         6793 non-null   uint8  
 9   Credit card (automatic)           6793 non-null   uint8  
 10  Electronic check                  6793 non-null   uint8  
 11  Mailed check                      6793 non-null   uint8  
 12  TotalCharges_MonthlyCharges_ratio  6793 non-null   float64
 13  AverageMonthlyCharges             6793 non-null   float64
 14  MonthlyCharges_tenure_Ratio       6793 non-null   float64
dtypes: float64(5), int32(1), int64(1), uint8(8)
memory usage: 709.1+ KB
```

In [43]:
```
df.to_csv('prepped_churn_data.csv')
df_log.to_csv('prepped_logchurn_data.csv')
```

Summary

Write your summary of the process and results here.

Process Summary

I began with a reviewing the dataframe to gain insight into the business and how to tackle the question being asked. For a deeper understanding of the data and the relationships I produced an EDA. I checked for any outliers using boxplots and no data points fell outside the whiskers so there were no outliers to deal with. Next, I scanned the data for any missing values and found 11 missing

Total Charges. I contemplated many ways of filling the data including interpolation *which I found most interesting* . Without having further knowledge of how the command interacts with the data I decided to just remove all rows with missing values. I didn't like leaving the Outliers where they were so I used the advanced KNN technique to find some. I removed 49 outliers. I again reviewed the data with the missing values and outliers removed. Next, I converted the categorical columns *(Churn, Contracts, PaymentMethods, PhoneService)* into numerical values. I created an assumption that *tenure* and *monthlycharges* are functions of *totalcharges* and converted all variables into logarithmic data to model in linear regression analyses. I created new ratio variables based on this assumption as well. I displayed the redistribution of the models under each conversion and confirmed whether or not the data was consistent with my assumptions. I saved the prepped dataframe all the ratios and logarithms for later use.

Summary of Results

I did find that there was a positive linear relationship between the totalcharges and tenure as well as the totalcharges and monthlycharges. I found a strong relationship between the Total and Monthly charges and a person's decision to churn. When the total charges of a customer's account reaches the $60 a month average over their tenure they will churn when their contract ends.

NOTE TO SELF!

Holy Mama! I need to look into the phik documentation more.

In [44]:
```
df.significance_matrix()
```

```
interval columns not set, guessing: ['tenure', 'MonthlyCharges', 'TotalCharges', 'Chur
n', 'PhoneService_Yes', 'Month-to-month', 'One year', 'Two year', 'Bank transfer (automa
tic)', 'Credit card (automatic)', 'Electronic check', 'Mailed check', 'TotalCharges_Mont
hlyCharges_ratio', 'AverageMonthlyCharges', 'MonthlyCharges_tenure_Ratio']
```

Out[44]:

	tenure	MonthlyCharges	TotalCharges	Churn	PhoneService_
tenure	171.068848	31.054338	98.905403	28.602258	0.747
MonthlyCharges	31.054338	169.241901	70.471978	23.496673	45.507
TotalCharges	98.905403	70.471978	160.260334	15.742746	14.914
Churn	28.602258	23.496673	15.742746	87.683937	-0.760
PhoneService_Yes	0.747484	45.507271	14.914681	-0.760114	65.292
Month-to-month	55.910926	27.488320	37.669213	34.462934	0.452
One year	25.049109	9.539799	15.159596	14.786262	-0.281
Two year	48.434179	24.941877	31.330225	28.223707	-0.136
Bank transfer (automatic)	18.662357	6.402178	14.778134	9.297893	0.631
Credit card (automatic)	17.613393	3.212492	14.111486	10.711845	-2.777
Electronic check	17.282709	26.209956	5.467159	23.851040	-1.556

	tenure	MonthlyCharges	TotalCharges	Churn	PhoneService_
Mailed check	17.610764	29.359040	25.549308	8.343216	0.447
TotalCharges_MonthlyCharges_ratio	152.607857	31.324091	99.581709	28.627207	0.594
AverageMonthlyCharges	27.334529	146.403649	70.592652	21.265897	41.867
MonthlyCharges_tenure_Ratio	60.614739	32.059021	45.899279	27.557239	6.273

In [45]:
```
cols = ['PhoneService_Yes','MonthlyCharges']
df[cols].hist2d()
```

```
interval columns not set, guessing: ['PhoneService_Yes', 'MonthlyCharges']
```

Out[45]:

MonthlyCharges	1	2	3	4	5	6	7	8	9	10
PhoneService_Yes										
1	76.0	177.0	178.0	150.0	72.0	0.0	0.0	0.0	0.0	0.0
10	1434.0	0.0	158.0	476.0	400.0	834.0	945.0	865.0	756.0	272.0

In [46]:
```
df[cols].outlier_significance_matrix()
```

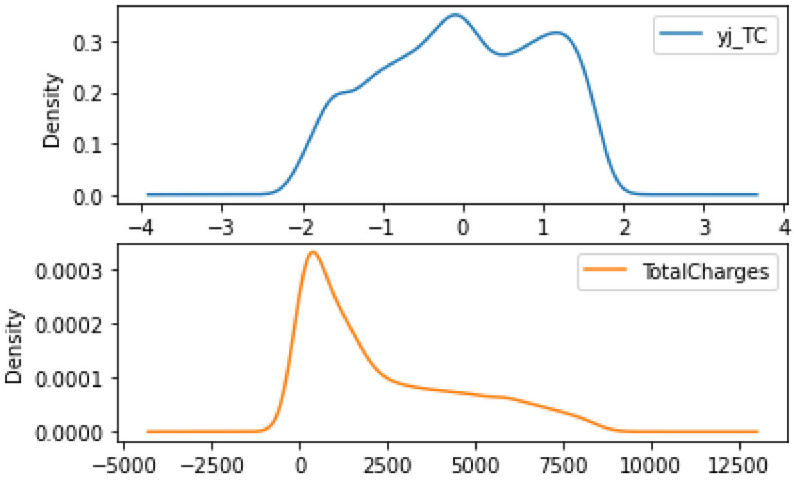
```
interval columns not set, guessing: ['PhoneService_Yes', 'MonthlyCharges']
```

Out[46]:

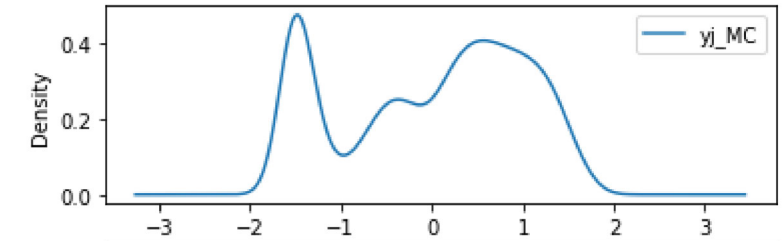
	18.2_28.3	28.3_38.3	38.3_48.4	48.4_58.4	58.4_68.5	68.5_78.5	78.5_88.6	88.6_98.6	98.(
-0.0_0.1	-7.277986	30.233108	21.380044	11.263757	4.033839	-13.208108	-14.144784	-13.474465	-12.
0.9_1.0	7.551288	-27.274342	-8.209536	-8.209536	-4.055075	13.327652	14.400595	13.631199	12.

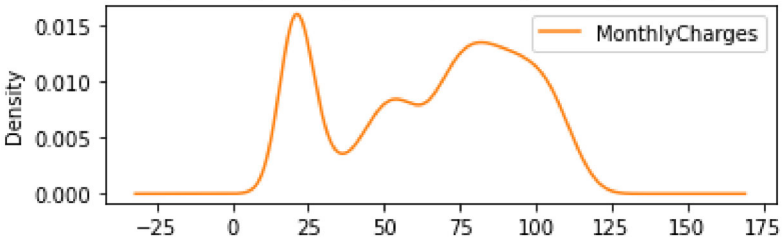
In [47]:
```
from sklearn.preprocessing import PowerTransformer

pt = PowerTransformer()
df['yj_TC'] = pt.fit_transform(df[['TotalCharges']])
df[['yj_TC', 'TotalCharges']].plot.density(subplots=True, sharex=False)
df['yj_MC'] = pt.fit_transform(df[['MonthlyCharges']])
df[['yj_MC', 'MonthlyCharges']].plot.density(subplots=True, sharex=False)
df['yj_tenure'] = pt.fit_transform(df[['tenure']])
df[['yj_tenure', 'tenure']].plot.density(subplots=True, sharex=False)
```

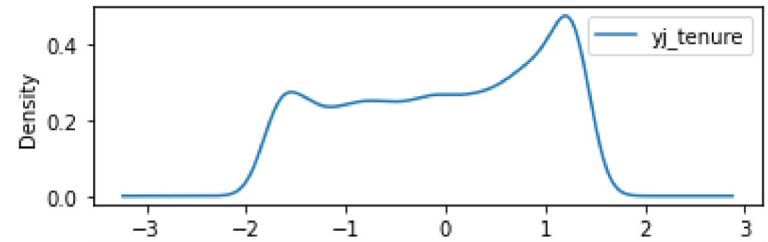
Out[47]:
```
array([<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
      dtype=object)
```

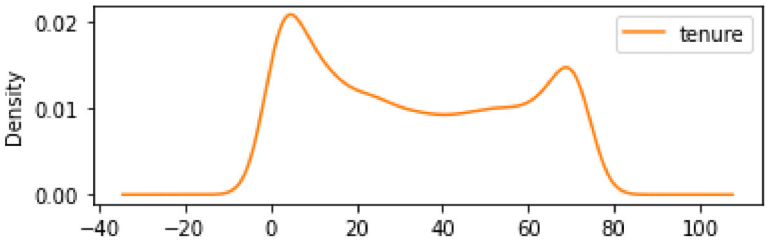

yj_TC
Density
0.3
0.2
0.1
0.0
−4
−3
−2
−1
0
1
2
3
4

TotalCharges
Density
0.0003
0.0002
0.0001
0.0000
−5000
−2500
0
2500
5000
7500
10000
12500


yj_MC
Density
0.4
0.2
0.0
−3
−2
−1
0
1
2
3


MonthlyCharges
Density
0.015
0.010
0.005
0.000
−25
0
25
50
75
100
125
150
175


yj_tenure
Density
0.4
0.2
0.0
−3
−2
−1
0
1
2
3


tenure
Density
0.02
0.01
0.00
−40
−20
0
20
40
60
80
100