Reducing Telecom Customer Churn: A Predictive Analytics Approach

Problem Statement:

Customer churn is an enormous challenge for telecom firms as consumers switch to competitors or end their contracts early. The task is to create predictive models capable of identifying and comprehending the aspects contributing to client attrition. This project will address the following questions:

1. What are the primary factors influencing telecom customer churn?

2. Can we create an accurate predictive model to anticipate which customers would churn?

3. How may this information be used to build targeted retention tactics and lower client churn?

Background:

Customer retention is crucial in the telecommunications sector since it is exceptionally competitive and consumer-centric. The rising cost of client acquisition and the possible revenue loss associated with churn have increased the importance of solving this issue. The origins of this problem may be traced back to the rising availability of telecom service providers, which provides consumers with more options, as well as customers' growing expectations for improved service quality, competitive pricing, and tailored experiences.

Significance:

Reduced telecom customer turnover is critical for various reasons, including:

1. Economic Impact: Customer turnover results in considerable revenue loss since gaining new customers is more expensive than maintaining existing ones.

2. Market Competition: The telecom sector is very competitive, with several service providers competing for clients. Reduced turnover can provide you with a competitive edge.

3. Customer happiness: Churn is a reflection of discontent with services, and resolving it leads to increased customer happiness and loyalty.

4. Data-Driven Insights: Using predictive analytics to understand consumer behavior better may help telecom firms customize their services and marketing efforts more successfully.

Contribution:

This initiative has the potential to benefit the telecom industry significantly:

1. Predictive Modeling: By developing accurate predictive models, this project will provide telecom companies with valuable insight into which customers are at risk of churning, allowing for pre-emptive retention efforts.

2. Data-Driven Decision-Making: The initiative will enable telecom firms to make data-driven choices by identifying and prioritizing the most significant churn drivers.

3. Cost Savings: Lowering churn rates can result in significant cost savings since businesses can devote more energy to maintaining existing customers rather than constantly gaining new ones.

4. Improved Customer Experience: Addressing churn reasons improves the customer experience, which can lead to long-term loyalty and advocacy.

Data Source:

This study's dataset was taken from Kaggle, a library of publicly available datasets for different data science and machine learning projects. The dataset was chosen for its relevance to the topic at hand, forecasting client attrition. Here are the data source's specifics:

- **Source:** Kaggle / IBM Sample Data Sets
- **Dataset Name**: Telco Customer Churn
- **Description**: This dataset contains customer information about their churn behavior and numerous factors such as subscribed services, customer account details, and demographic information.

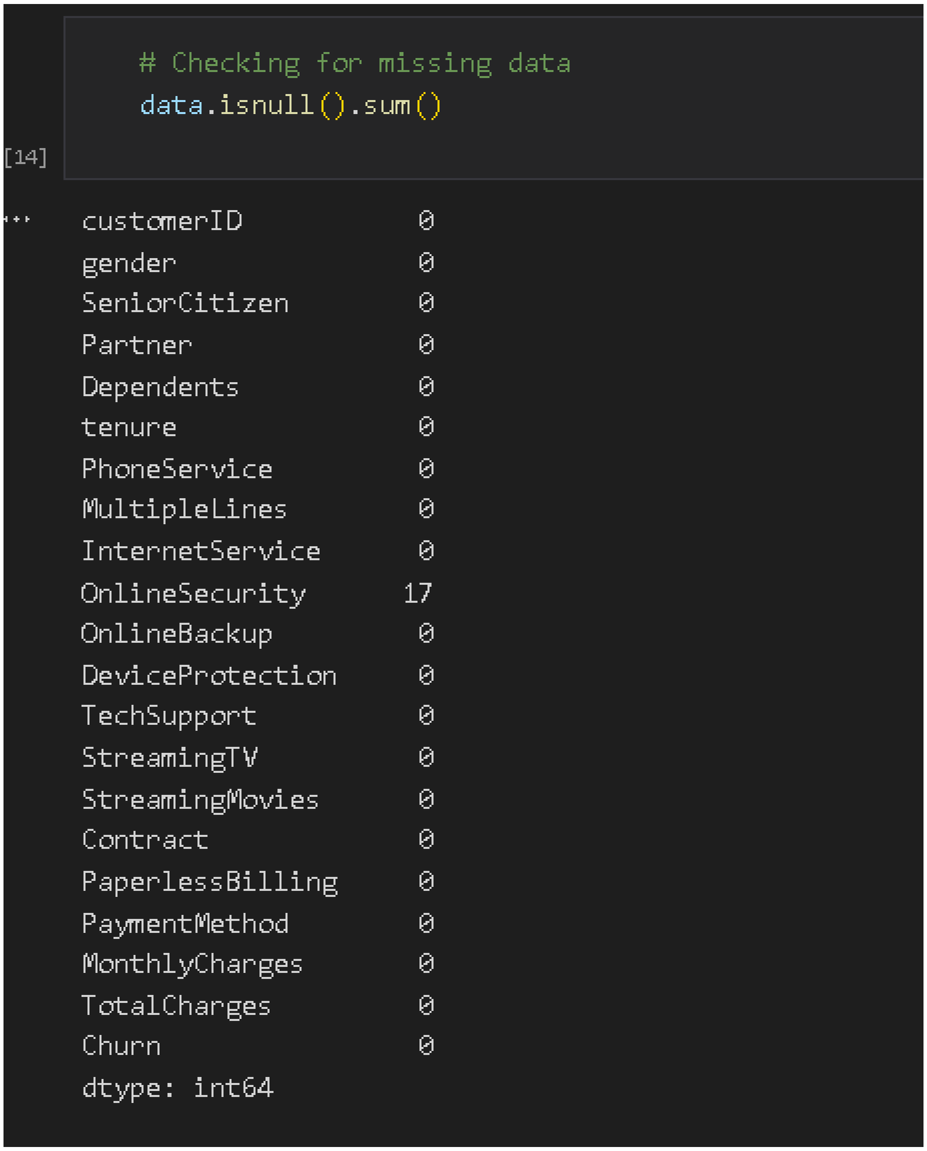
Rationale for Dataset Selection

The necessity for a comprehensive dataset that could be utilized to investigate client retention tactics drove the choice to use this dataset. Customers who departed within the last month are included in the dataset, as are the services they signed up for, account information, and demographic information. It has about 7000 rows and 21 columns, which provides a big enough sample size for significant data analysis and predictive modeling.

In summary, the dataset utilized in this study was obtained from IBM Sample Data Sets and was selected due to its relevance to customer churn analysis. It satisfied the criterion of having sufficient records (over 2000 rows) and the relevant columns to satisfy the Phase 1 objectives.

Data Cleaning

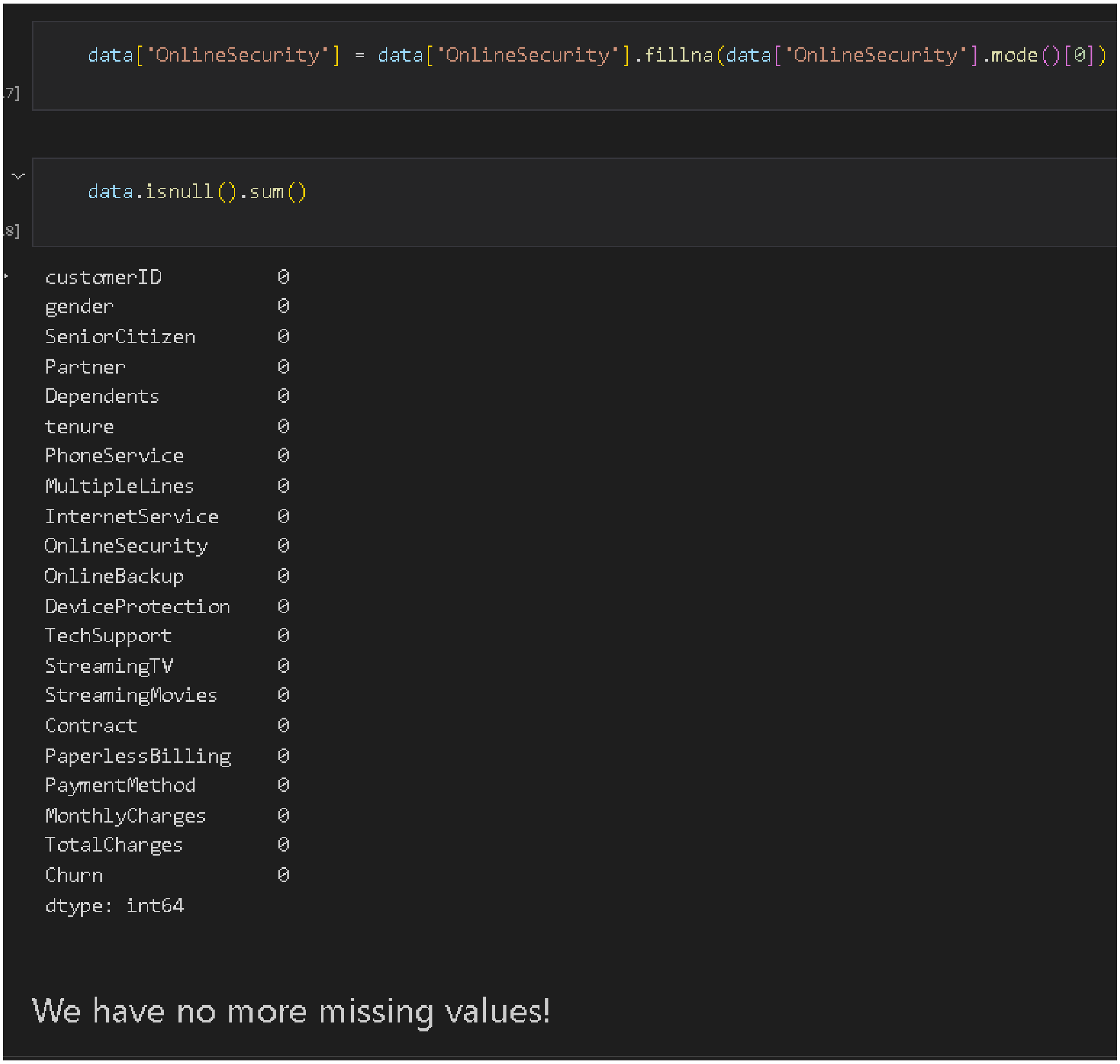
- **Dealing with missing data**
    - We have missing data in the column 'OnlineSecurity.'



```
# Checking for missing data
data.isnull().sum()
```

```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity      17
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges         0
Churn                0
dtype: int64
```

- Column OnlineSecurity is categorical data and 50% of the people have No as the categorical value so replacing missing values with No. Using the mode of the column for data imputation.



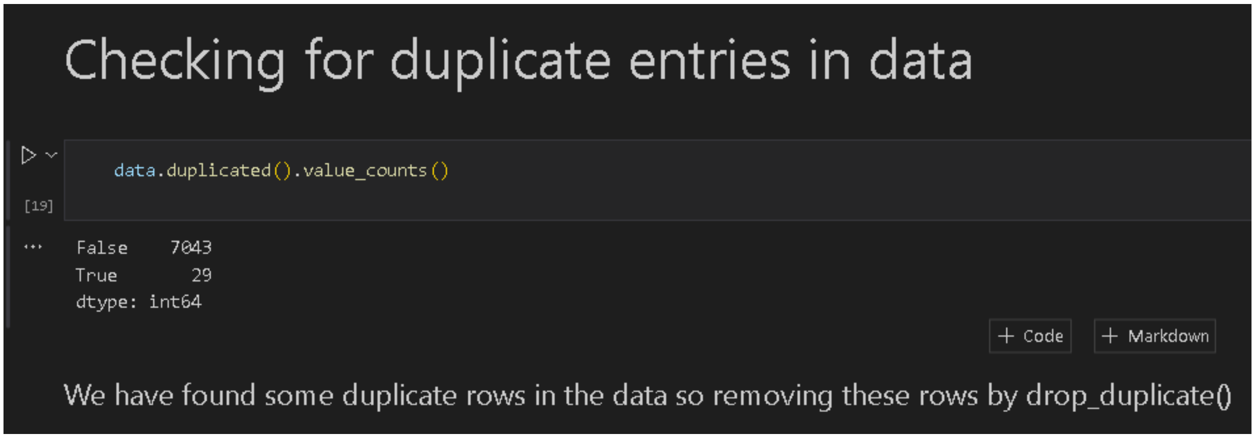
```python
data['OnlineSecurity'] = data['OnlineSecurity'].fillna(data['OnlineSecurity'].mode()[0])
```

```python
data.isnull().sum()
```

```
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
```

We have no more missing values!

- **Removing Duplicate Entries**

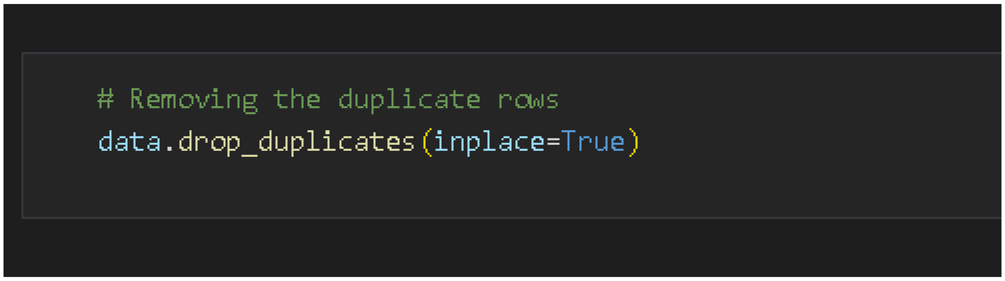


Checking for duplicate entries in data

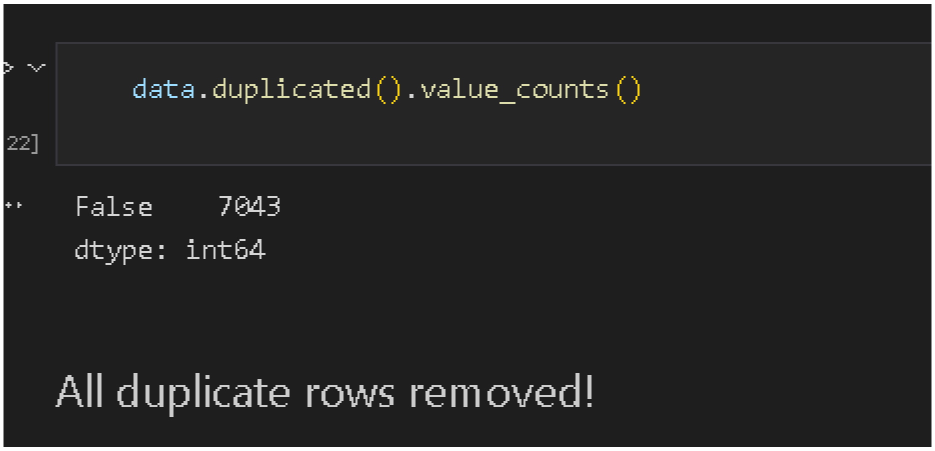
```
data.duplicated().value_counts()
```

```
False    7043
True       29
dtype: int64
```

We have found some duplicate rows in the data so removing these rows by drop_duplicate()



```
# Removing the duplicate rows
data.drop_duplicates(inplace=True)
```

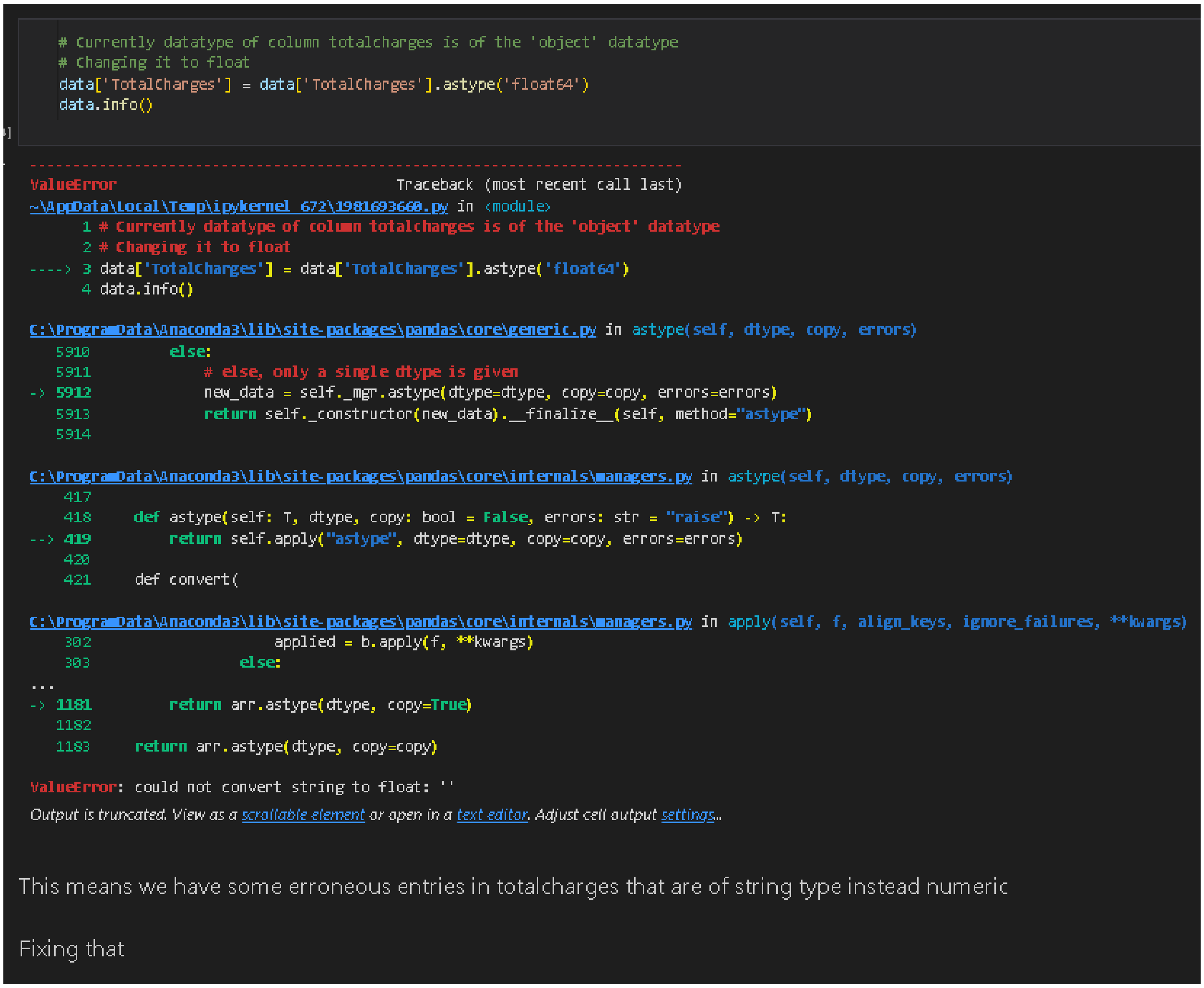


```
data.duplicated().value_counts()
```

```
False    7043
dtype: int64
```

All duplicate rows removed!

➢ Fixing Inconsistencies in column



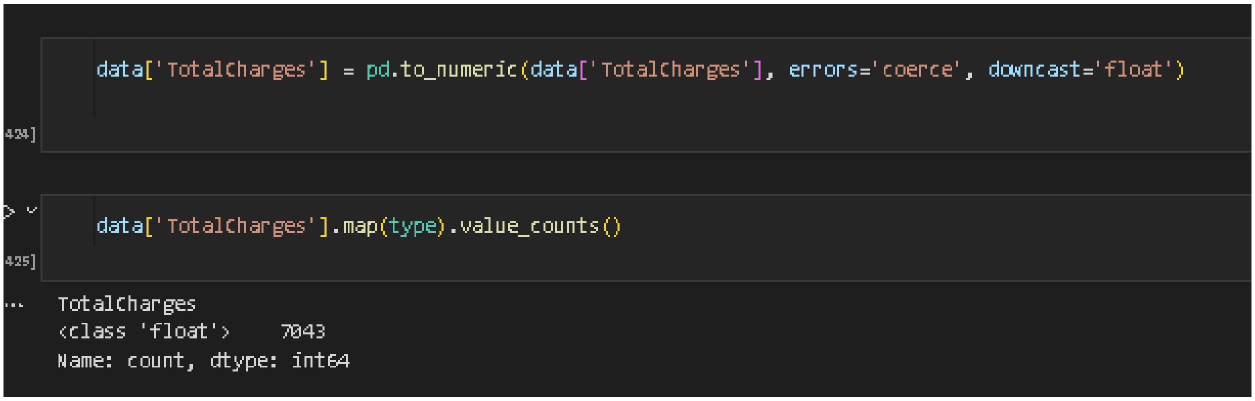
```
# Currently datatype of column totalcharges is of the 'object' datatype
# Changing it to float
data['TotalCharges'] = data['TotalCharges'].astype('float64')
data.info()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_672\1981693660.py in <module>
      1 # Currently datatype of column totalcharges is of the 'object' datatype
      2 # Changing it to float
----> 3 data['TotalCharges'] = data['TotalCharges'].astype('float64')
      4 data.info()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in astype(self, dtype, copy, errors)
   5910         else:
   5911             # else, only a single dtype is given
-> 5912             new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
   5913             return self._constructor(new_data).__finalize__(self, method="astype")
   5914 

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\internals\managers.py in astype(self, dtype, copy, errors)
    417 
    418     def astype(self: T, dtype, copy: bool = False, errors: str = "raise") -> T:
--> 419         return self.apply("astype", dtype=dtype, copy=copy, errors=errors)
    420 
    421     def convert(

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\internals\managers.py in apply(self, f, align_keys, ignore_failures, **kwargs)
    302                     applied = b.apply(f, **kwargs)
    303                 else:
...
-> 1181         return arr.astype(dtype, copy=True)
   1182 
   1183     return arr.astype(dtype, copy=copy)

ValueError: could not convert string to float: ''
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

This means we have some erroneous entries in totalcharges that are of string type instead numeric

Fixing that

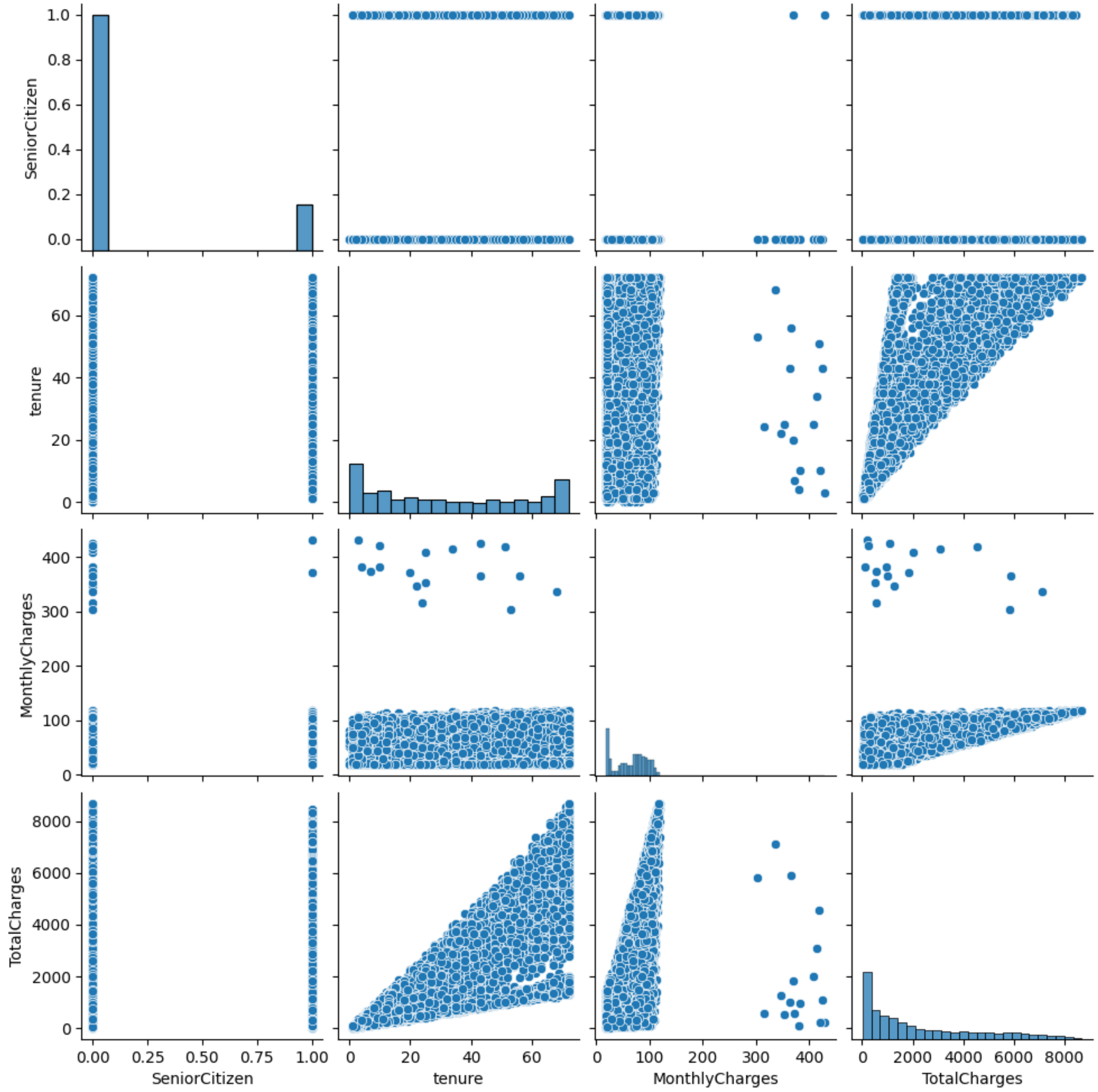


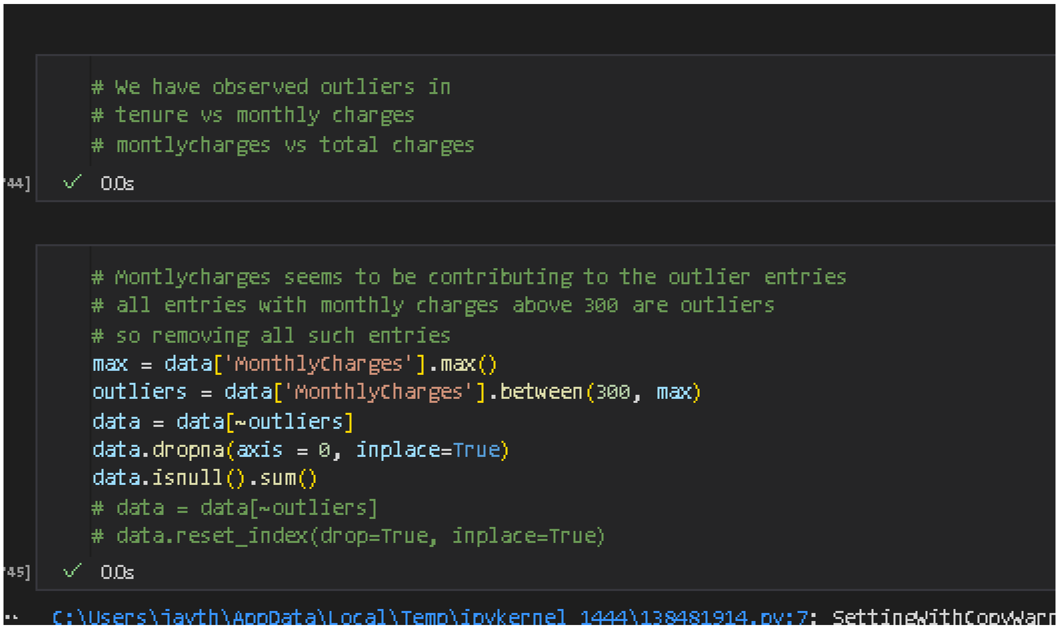
```
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce', downcast='float')
```

```
data['TotalCharges'].map(type).value_counts()
```

```
TotalCharges
<class 'float'>    7043
Name: count, dtype: int64
```

- **Removing Outliers**
  - We have outliers in our data, which are arising due to high-paying customers.
  - This can negatively affect our predictive analysis during the modeling phase, thus removing these outlier customers.





```
# We have observed outliers in
# tenure vs monthly charges
# montlycharges vs total charges
```

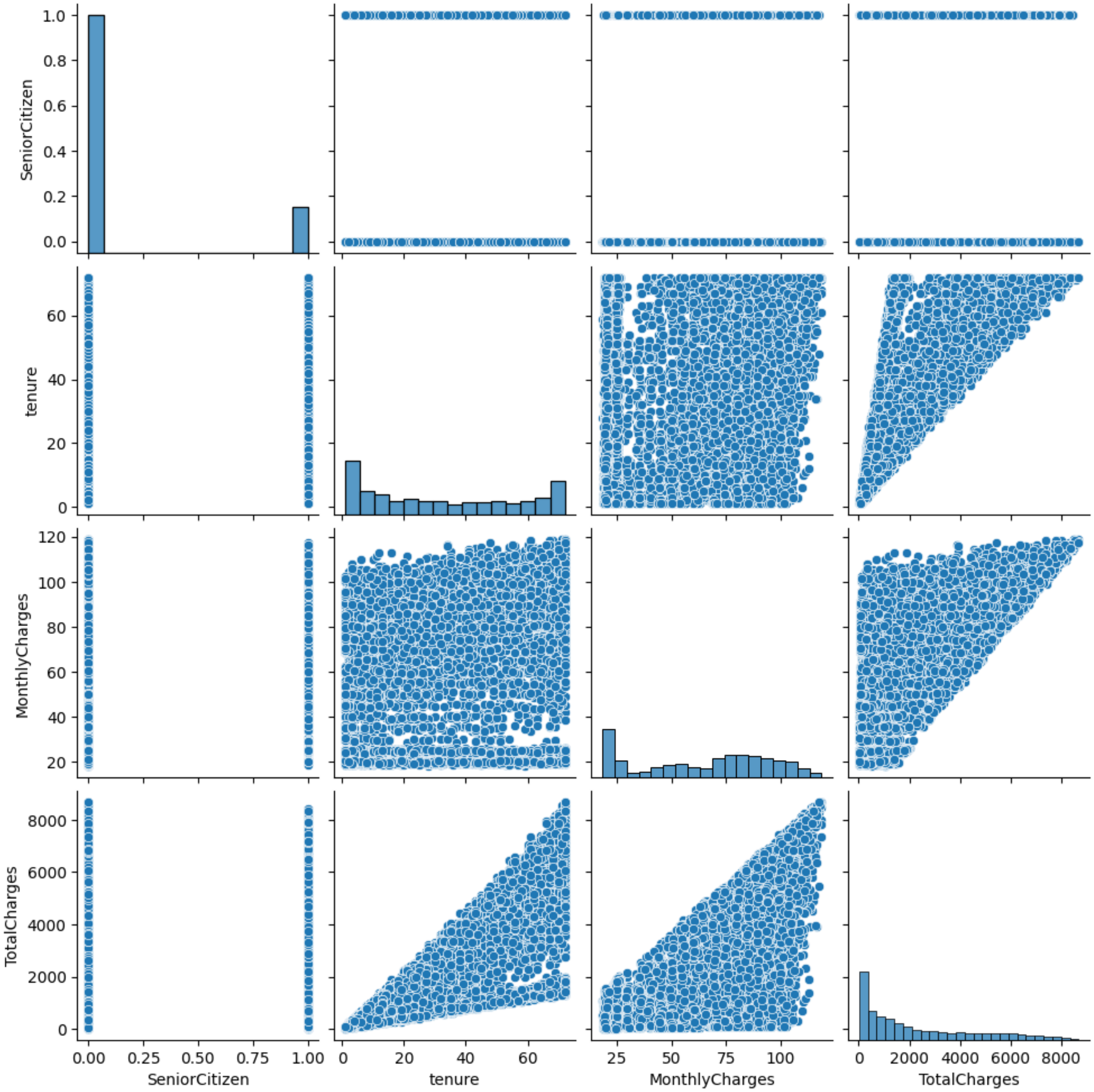
44] ✓ 0.0s

```
# Montlycharges seems to be contributing to the outlier entries
# all entries with monthly charges above 300 are outliers
# so removing all such entries
max = data['MonthlyCharges'].max()
outliers = data['MonthlyCharges'].between(300, max)
data = data[~outliers]
data.dropna(axis = 0, inplace=True)
data.isnull().sum()
# data = data[~outliers]
# data.reset_index(drop=True, inplace=True)
```

45] ✓ 0.0s

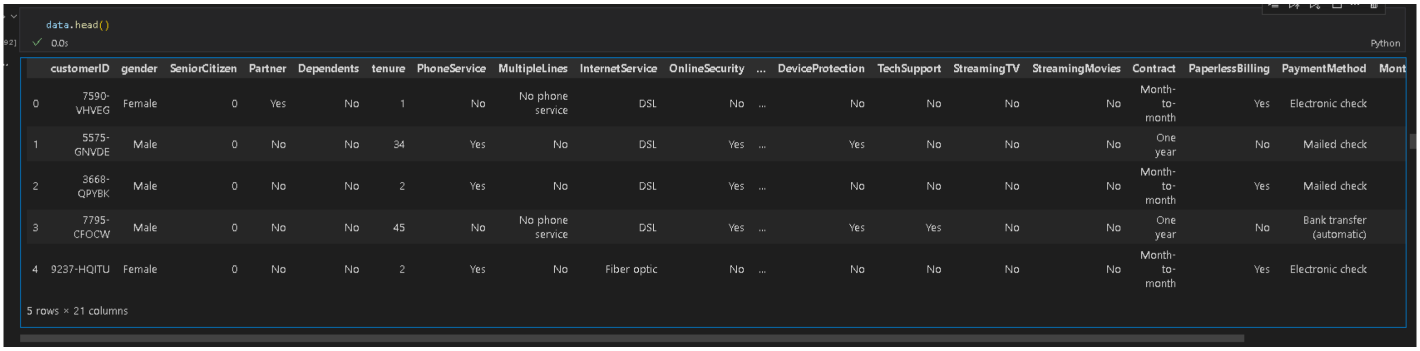
```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\138481914.py:7: SettingWithCopyWarn
```

- After removing outliers, our graph looks like this:



➢ Addressing Inconsistencies in Categorical Data

- Multiple columns have redundant category names that can be merged into one. This will reduce the complexity of the data.
- Before:

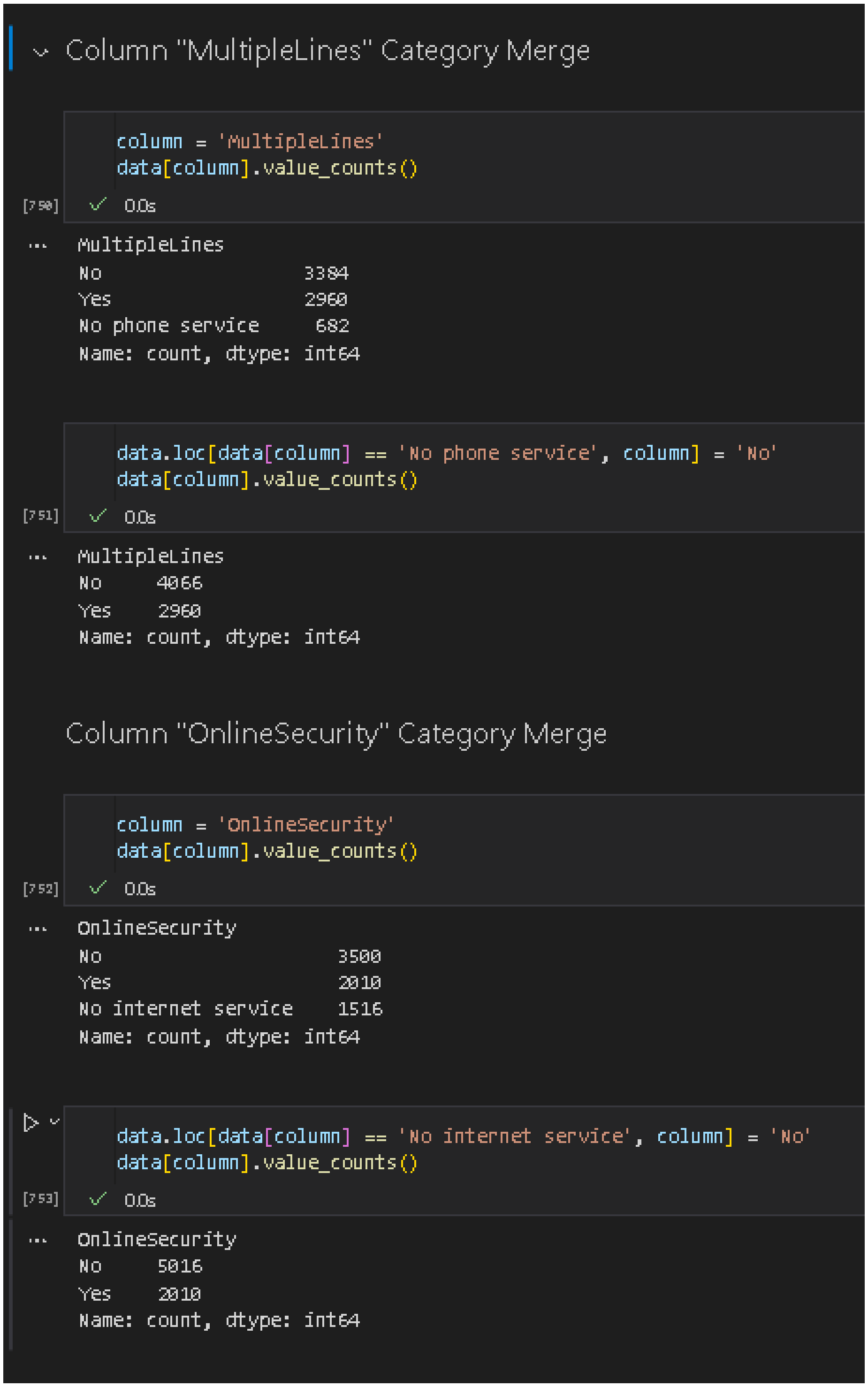


```
data.head()
```

✓ 0.0s

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	Mont
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year	No	Mailed check	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month-to-month	Yes	Mailed check	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month-to-month	Yes	Electronic check	

5 rows × 21 columns



Column "MultipleLines" Category Merge

```
column = 'MultipleLines'
data[column].value_counts()
```

[750] ✓ 0.0s

```
MultipleLines
No                  3384
Yes                 2960
No phone service     682
Name: count, dtype: int64
```

```
data.loc[data[column] == 'No phone service', column] = 'No'
data[column].value_counts()
```

[751] ✓ 0.0s

```
MultipleLines
No     4066
Yes    2960
Name: count, dtype: int64
```

Column "OnlineSecurity" Category Merge

```
column = 'OnlineSecurity'
data[column].value_counts()
```

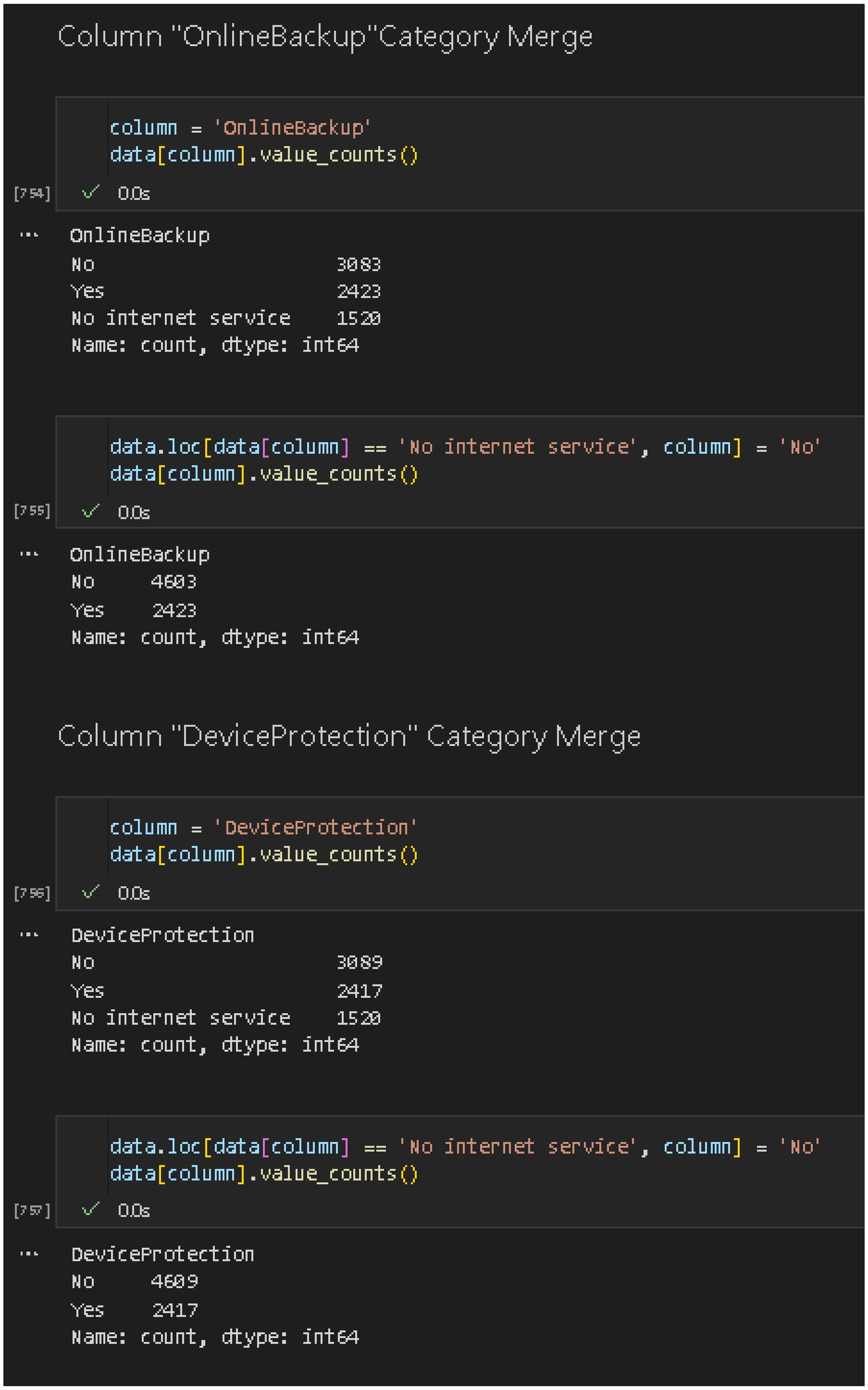
[752] ✓ 0.0s

```
OnlineSecurity
No                     3500
Yes                    2010
No internet service    1516
Name: count, dtype: int64
```

```
data.loc[data[column] == 'No internet service', column] = 'No'
data[column].value_counts()
```

[753] ✓ 0.0s

```
OnlineSecurity
No     5016
Yes    2010
Name: count, dtype: int64
```



Column "OnlineBackup"Category Merge

```
column = 'OnlineBackup'
data[column].value_counts()
```

[754] ✓ 0.0s

```
OnlineBackup
No                     3083
Yes                    2423
No internet service    1520
Name: count, dtype: int64
```

```
data.loc[data[column] == 'No internet service', column] = 'No'
data[column].value_counts()
```

[755] ✓ 0.0s

```
OnlineBackup
No     4603
Yes    2423
Name: count, dtype: int64
```

Column "DeviceProtection" Category Merge

```
column = 'DeviceProtection'
data[column].value_counts()
```

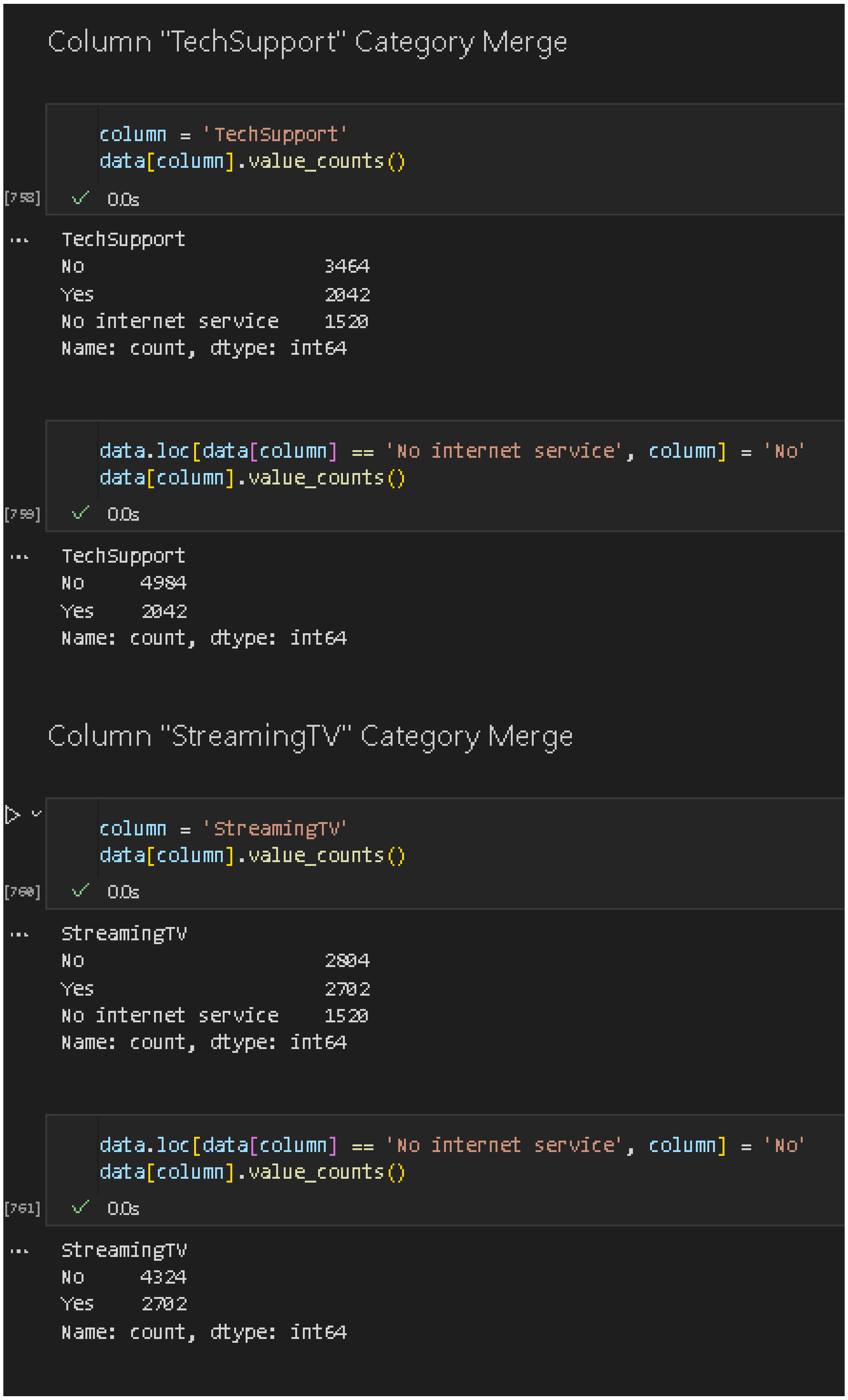
[756] ✓ 0.0s

```
DeviceProtection
No                     3089
Yes                    2417
No internet service    1520
Name: count, dtype: int64
```

```
data.loc[data[column] == 'No internet service', column] = 'No'
data[column].value_counts()
```

[757] ✓ 0.0s

```
DeviceProtection
No     4609
Yes    2417
Name: count, dtype: int64
```



Column "TechSupport" Category Merge

```
column = 'TechSupport'
data[column].value_counts()
```

```
TechSupport
No                     3464
Yes                    2042
No internet service    1520
Name: count, dtype: int64
```

```
data.loc[data[column] == 'No internet service', column] = 'No'
data[column].value_counts()
```

```
TechSupport
No     4984
Yes    2042
Name: count, dtype: int64
```

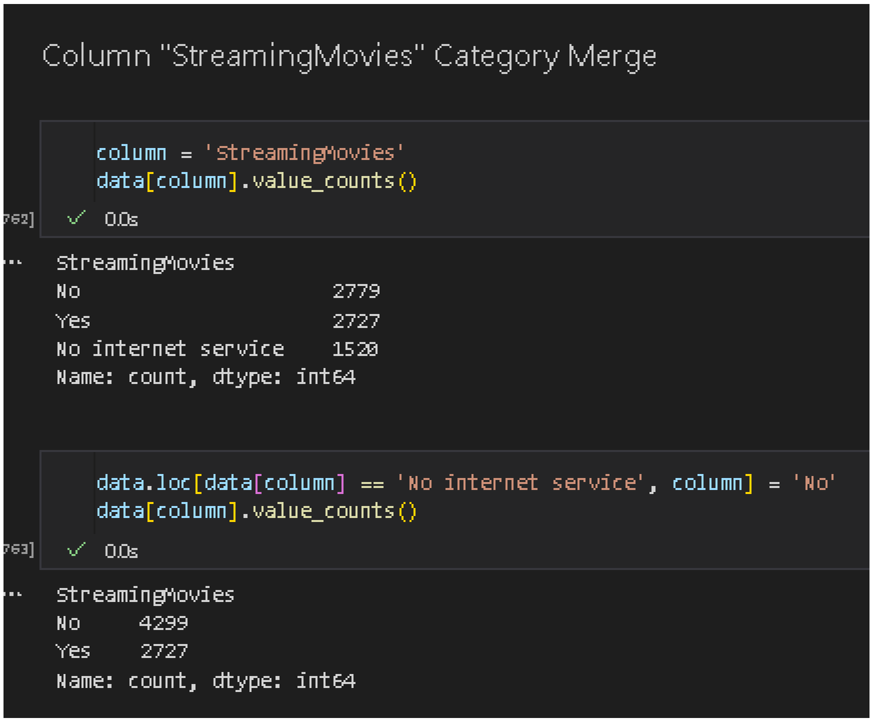
Column "StreamingTV" Category Merge

```
column = 'StreamingTV'
data[column].value_counts()
```

```
StreamingTV
No                     2804
Yes                    2702
No internet service    1520
Name: count, dtype: int64
```

```
data.loc[data[column] == 'No internet service', column] = 'No'
data[column].value_counts()
```

```
StreamingTV
No     4324
Yes    2702
Name: count, dtype: int64
```



Column "StreamingMovies" Category Merge

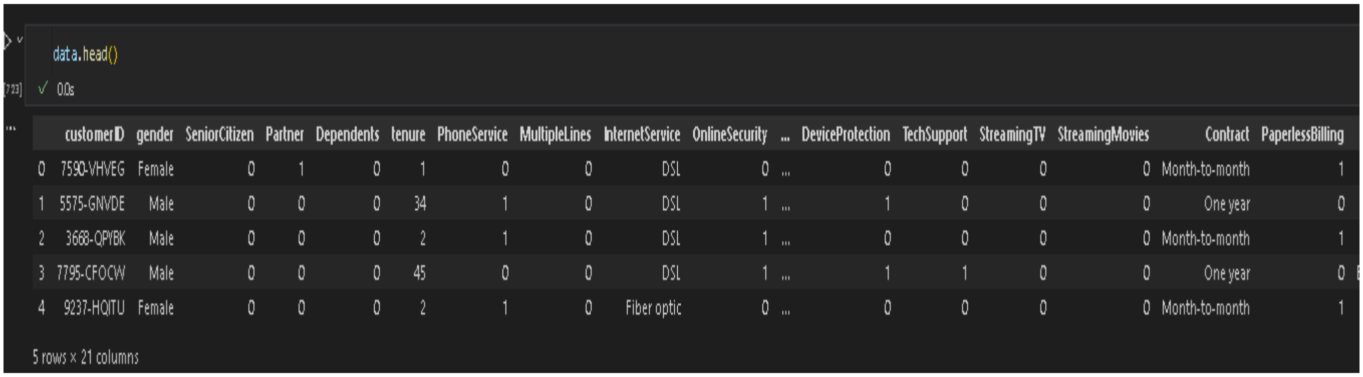
```
column = 'StreamingMovies'
data[column].value_counts()
```

```
StreamingMovies
No                     2779
Yes                    2727
No internet service    1520
Name: count, dtype: int64
```

```
data.loc[data[column] == 'No internet service', column] = 'No'
data[column].value_counts()
```

```
StreamingMovies
No     4299
Yes    2727
Name: count, dtype: int64
```

- **Label Encoding Categorical Columns :**

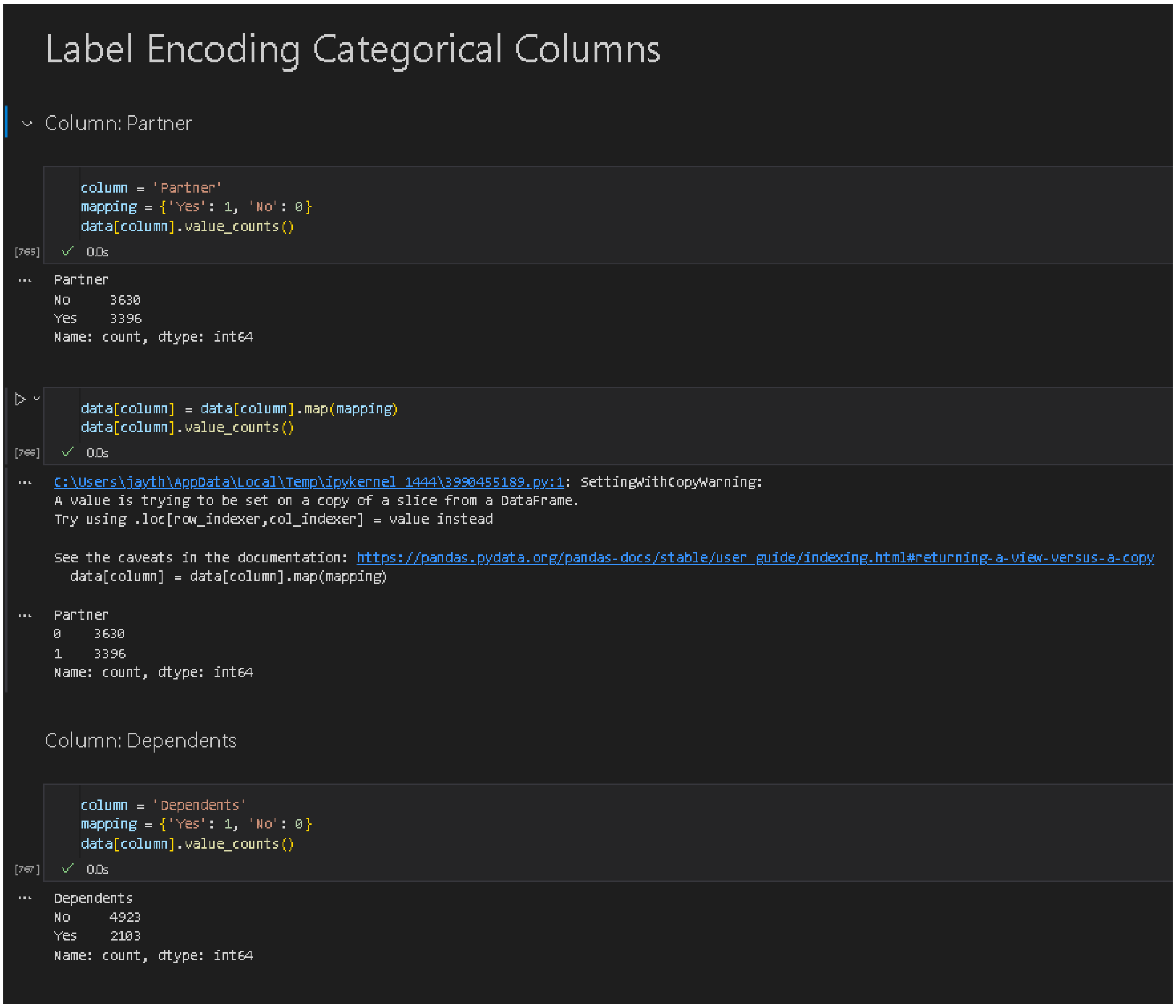


```
data.head()
```

✓ 0.0s

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling
0	7590-VHVEG	Female	0	1	0	1	0	0	DSL	0	...	0	0	0	0	Month-to-month	1
1	5575-GNVDE	Male	0	0	0	34	1	0	DSL	1	...	1	0	0	0	One year	0
2	3668-QPYBK	Male	0	0	0	2	1	0	DSL	1	...	0	0	0	0	Month-to-month	1
3	7795-CFOCW	Male	0	0	0	45	0	0	DSL	1	...	1	1	0	0	One year	0
4	9237-HQITU	Female	0	0	0	2	1	0	Fiber optic	0	...	0	0	0	0	Month-to-month	1

5 rows × 21 columns



Label Encoding Categorical Columns

Column: Partner

```
column = 'Partner'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

✓ 0.0s

```
Partner
No     3630
Yes    3396
Name: count, dtype: int64
```

```
data[column] = data[column].map(mapping)
data[column].value_counts()
```

✓ 0.0s

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\3990455189.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[column] = data[column].map(mapping)
```

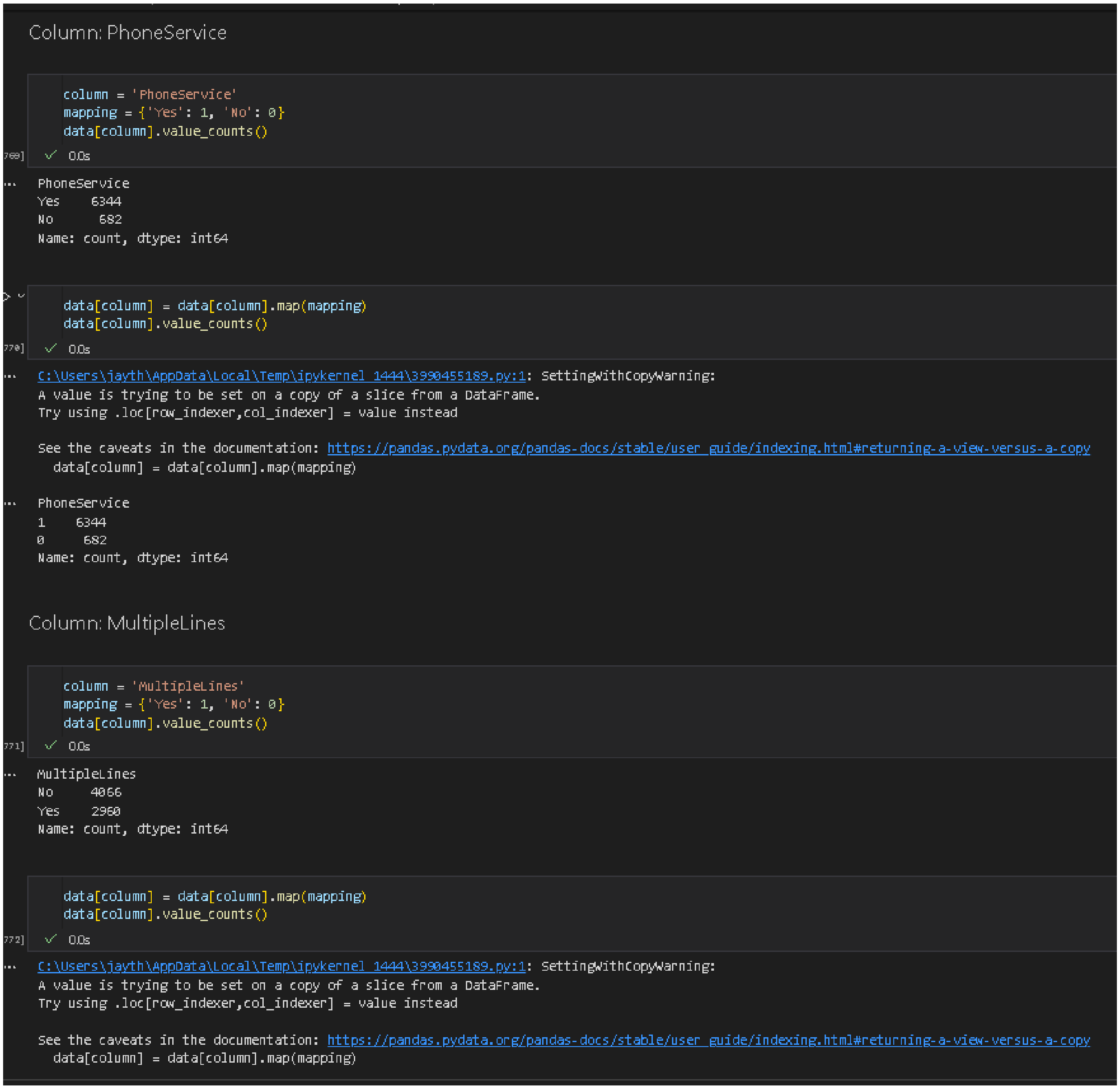
```
Partner
0    3630
1    3396
Name: count, dtype: int64
```

Column: Dependents

```
column = 'Dependents'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

✓ 0.0s

```
Dependents
No     4923
Yes    2103
Name: count, dtype: int64
```



Column: PhoneService

```
column = 'PhoneService'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

769] ✓ 0.0s

```
PhoneService
Yes     6344
No       682
Name: count, dtype: int64
```

```
data[column] = data[column].map(mapping)
data[column].value_counts()
```

770] ✓ 0.0s

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\3990455189.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[column] = data[column].map(mapping)
```

```
PhoneService
1    6344
0     682
Name: count, dtype: int64
```

Column: MultipleLines

```
column = 'MultipleLines'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

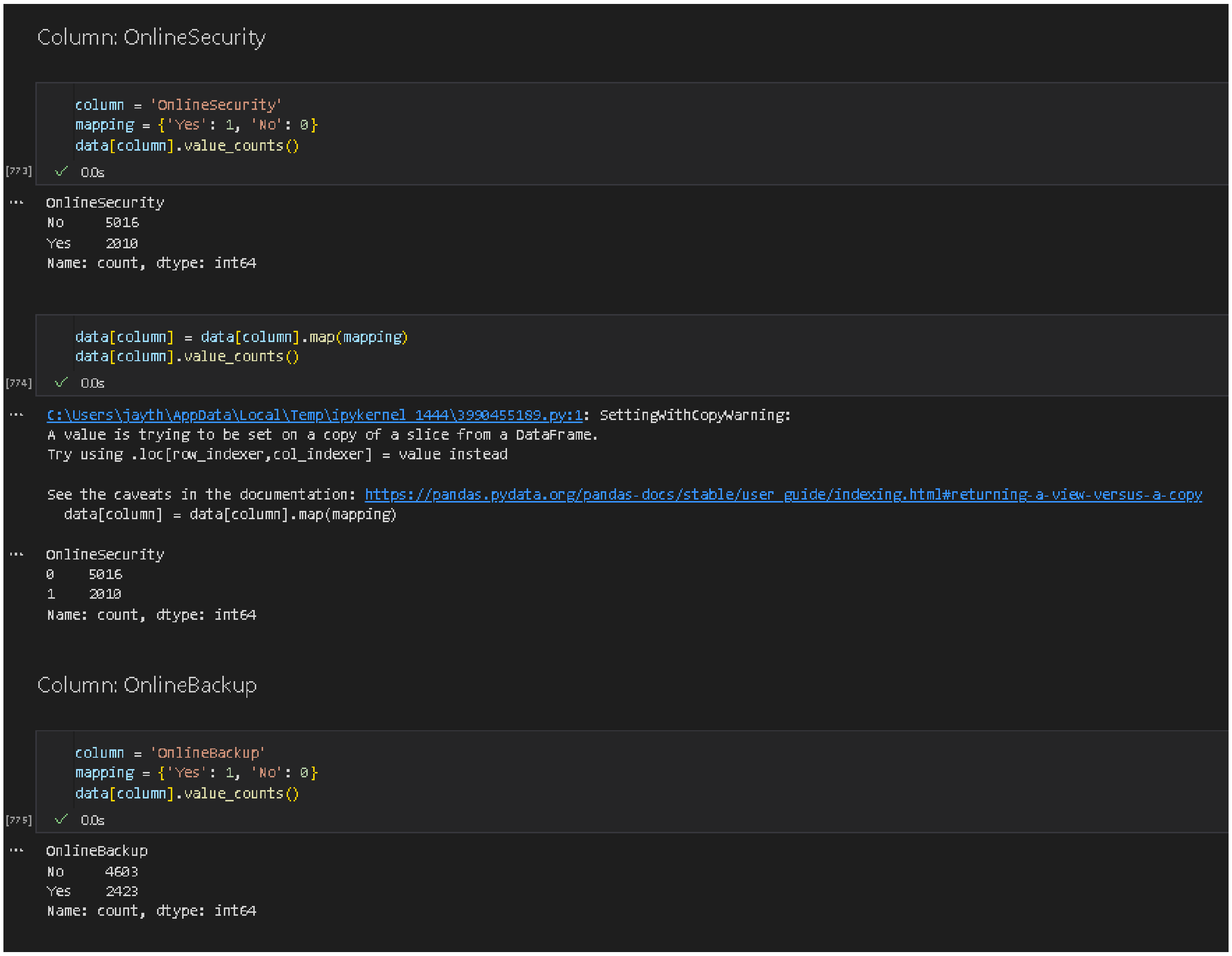
771] ✓ 0.0s

```
MultipleLines
No      4066
Yes     2960
Name: count, dtype: int64
```

```
data[column] = data[column].map(mapping)
data[column].value_counts()
```

772] ✓ 0.0s

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\3990455189.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[column] = data[column].map(mapping)
```



Column: OnlineSecurity

```
column = 'OnlineSecurity'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

[773] ✓ 0.0s

```
OnlineSecurity
No     5016
Yes    2010
Name: count, dtype: int64
```

```
data[column] = data[column].map(mapping)
data[column].value_counts()
```

[774] ✓ 0.0s

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\3990455189.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[column] = data[column].map(mapping)
```

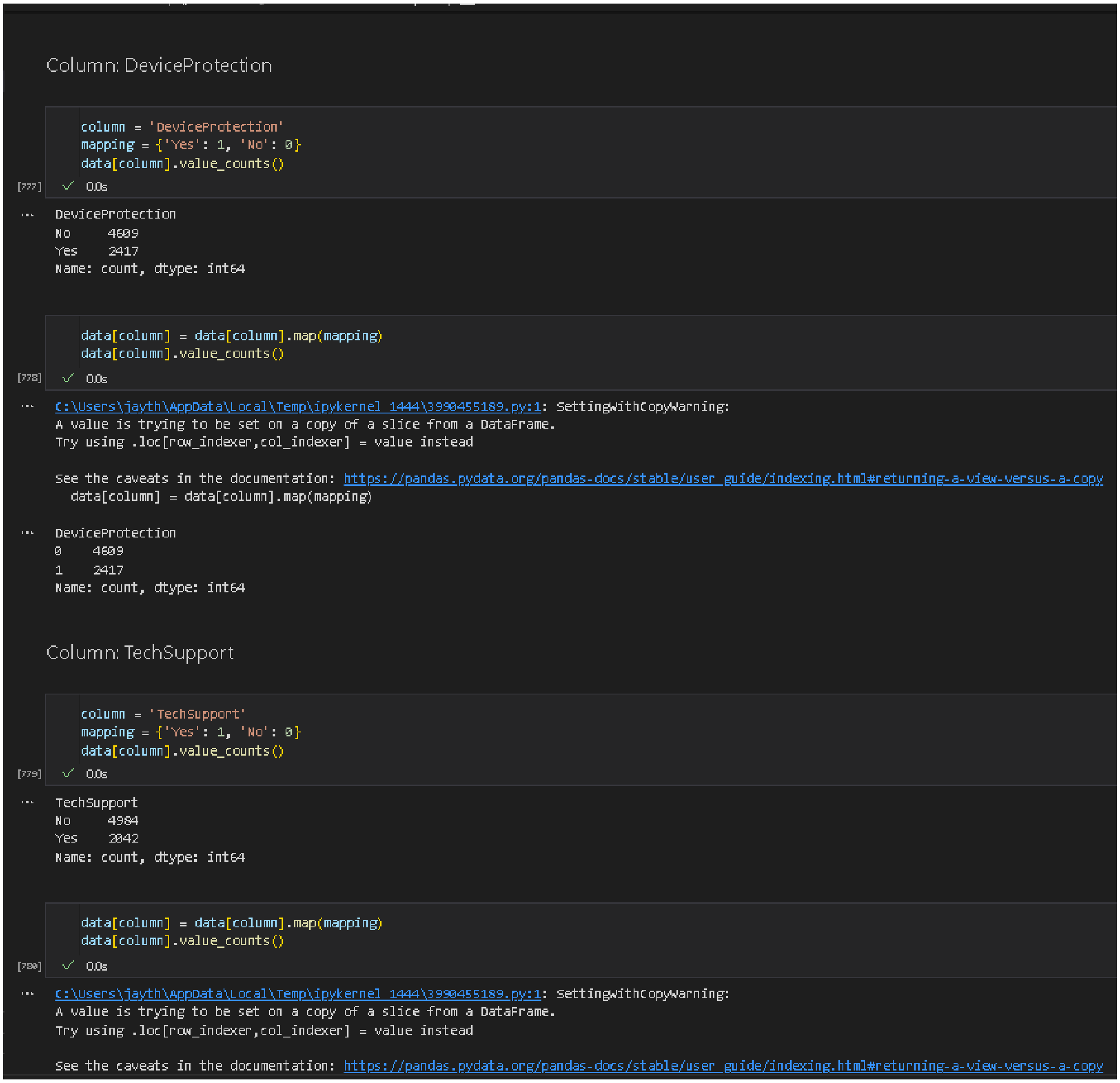
```
OnlineSecurity
0    5016
1    2010
Name: count, dtype: int64
```

Column: OnlineBackup

```
column = 'OnlineBackup'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

[775] ✓ 0.0s

```
OnlineBackup
No     4603
Yes    2423
Name: count, dtype: int64
```



Column: DeviceProtection

```
column = 'DeviceProtection'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

[777] ✓ 0.0s

```
DeviceProtection
No     4609
Yes    2417
Name: count, dtype: int64
```

```
data[column] = data[column].map(mapping)
data[column].value_counts()
```

[778] ✓ 0.0s

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\3990455189.py:1: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[column] = data[column].map(mapping)
```

```
DeviceProtection
0    4609
1    2417
Name: count, dtype: int64
```

Column: TechSupport

```
column = 'TechSupport'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

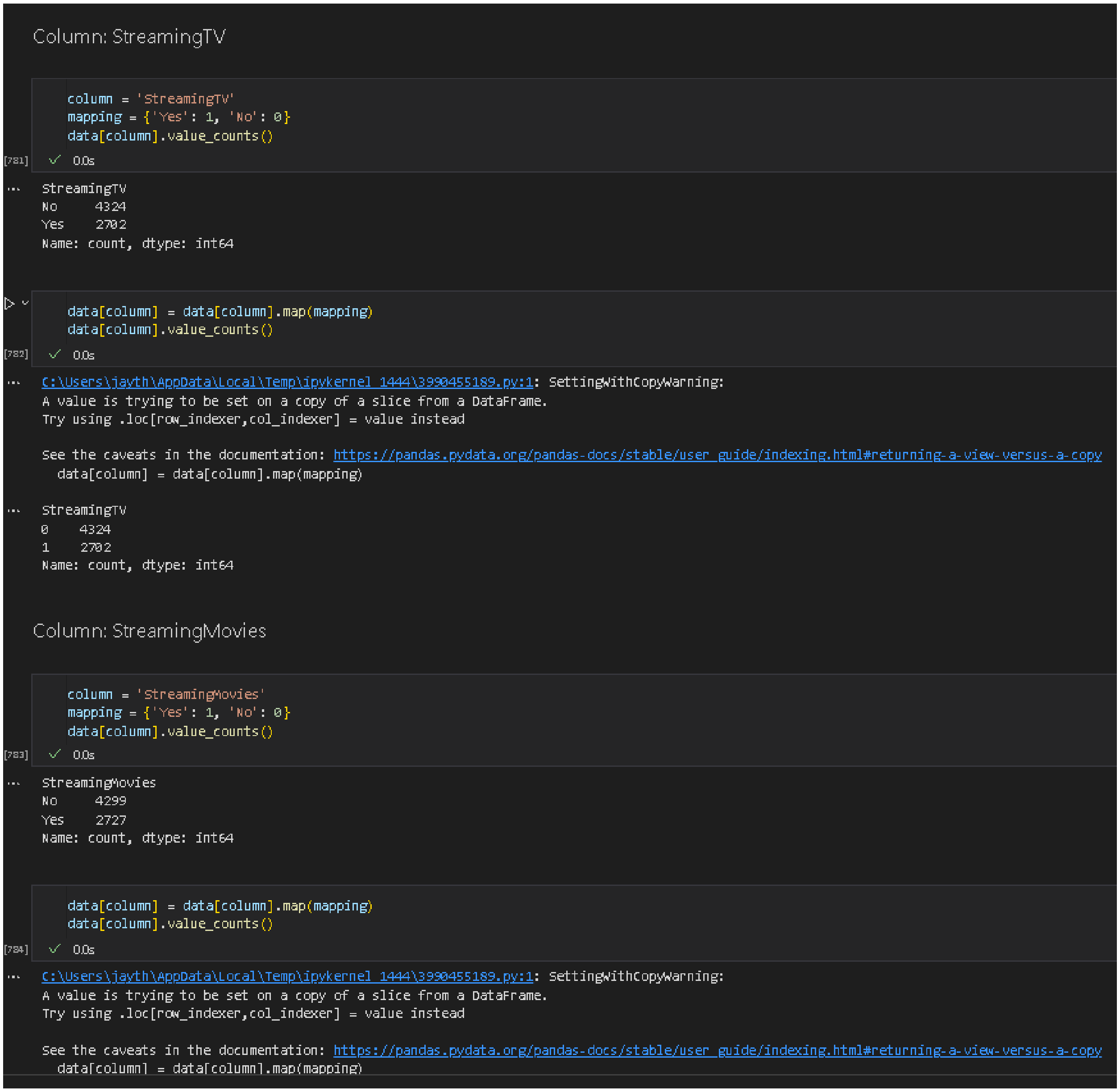
[779] ✓ 0.0s

```
TechSupport
No     4984
Yes    2042
Name: count, dtype: int64
```

```
data[column] = data[column].map(mapping)
data[column].value_counts()
```

[780] ✓ 0.0s

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\3990455189.py:1: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```



Column: StreamingTV

```
column = 'StreamingTV'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

[781] ✓ 0.0s

```
StreamingTV
No     4324
Yes    2702
Name: count, dtype: int64
```

```
data[column] = data[column].map(mapping)
data[column].value_counts()
```

[782] ✓ 0.0s

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\3990455189.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[column] = data[column].map(mapping)
```

```
StreamingTV
0    4324
1    2702
Name: count, dtype: int64
```

Column: StreamingMovies

```
column = 'StreamingMovies'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

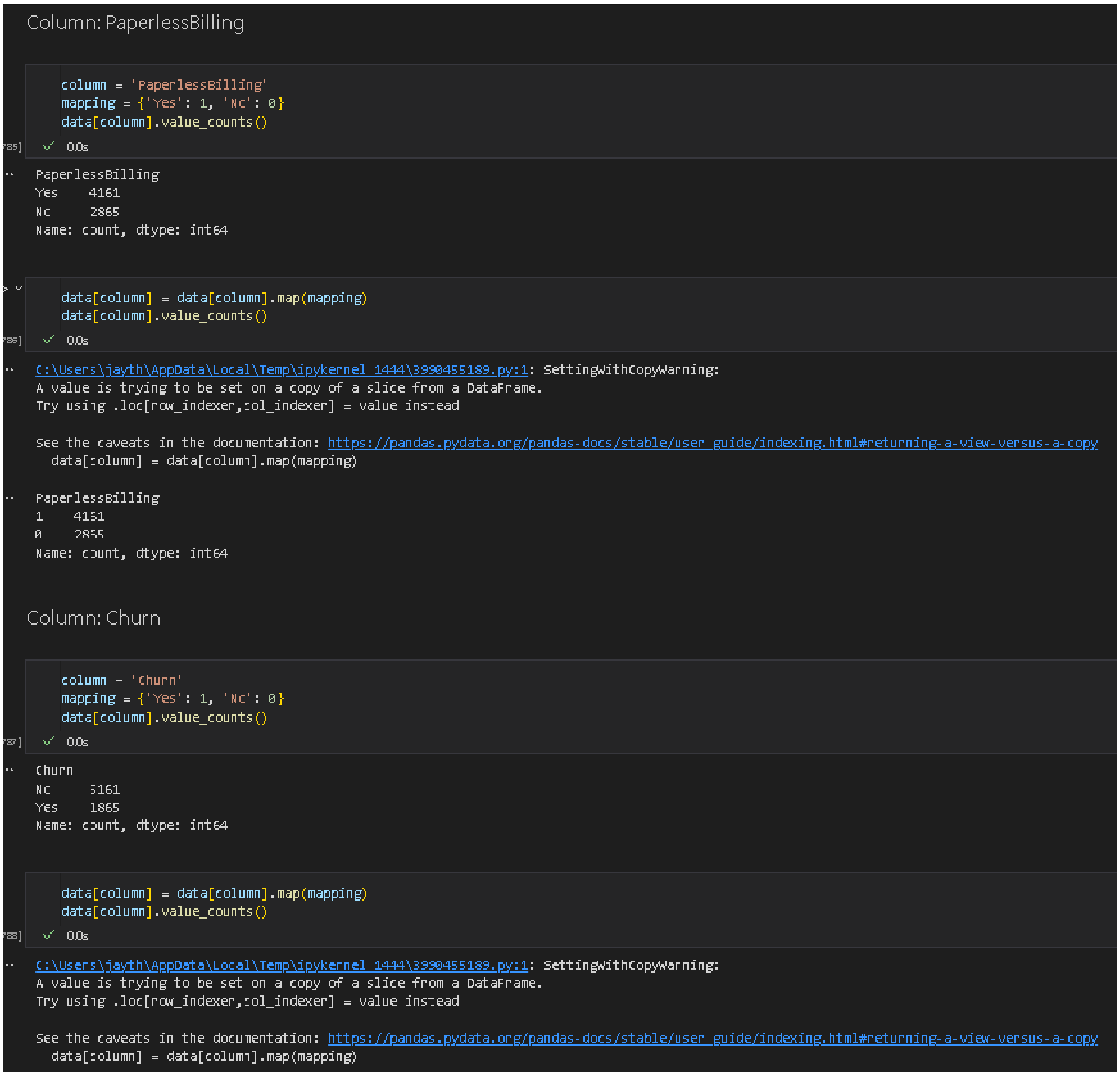
[783] ✓ 0.0s

```
StreamingMovies
No     4299
Yes    2727
Name: count, dtype: int64
```

```
data[column] = data[column].map(mapping)
data[column].value_counts()
```

[784] ✓ 0.0s

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\3990455189.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[column] = data[column].map(mapping)
```



Column: PaperlessBilling

```
column = 'PaperlessBilling'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

```
785]  ✓ 0.0s
```

```
PaperlessBilling
Yes     4161
No      2865
Name: count, dtype: int64
```

```
data[column] = data[column].map(mapping)
data[column].value_counts()
```

```
786]  ✓ 0.0s
```

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\3990455189.py:1: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[column] = data[column].map(mapping)
```

```
PaperlessBilling
1    4161
0    2865
Name: count, dtype: int64
```

Column: Churn

```
column = 'Churn'
mapping = {'Yes': 1, 'No': 0}
data[column].value_counts()
```

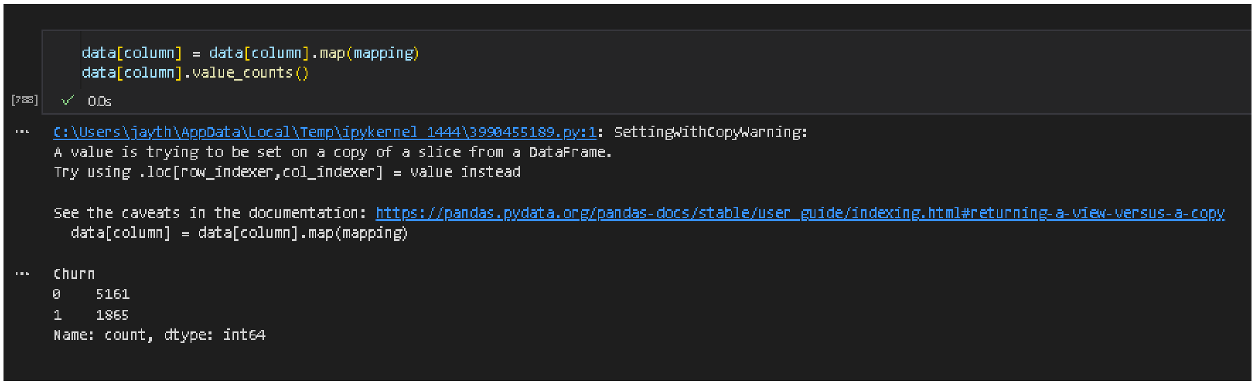
```
787]  ✓ 0.0s
```

```
Churn
No      5161
Yes     1865
Name: count, dtype: int64
```

```
data[column] = data[column].map(mapping)
data[column].value_counts()
```

```
788]  ✓ 0.0s
```

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\3990455189.py:1: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[column] = data[column].map(mapping)
```

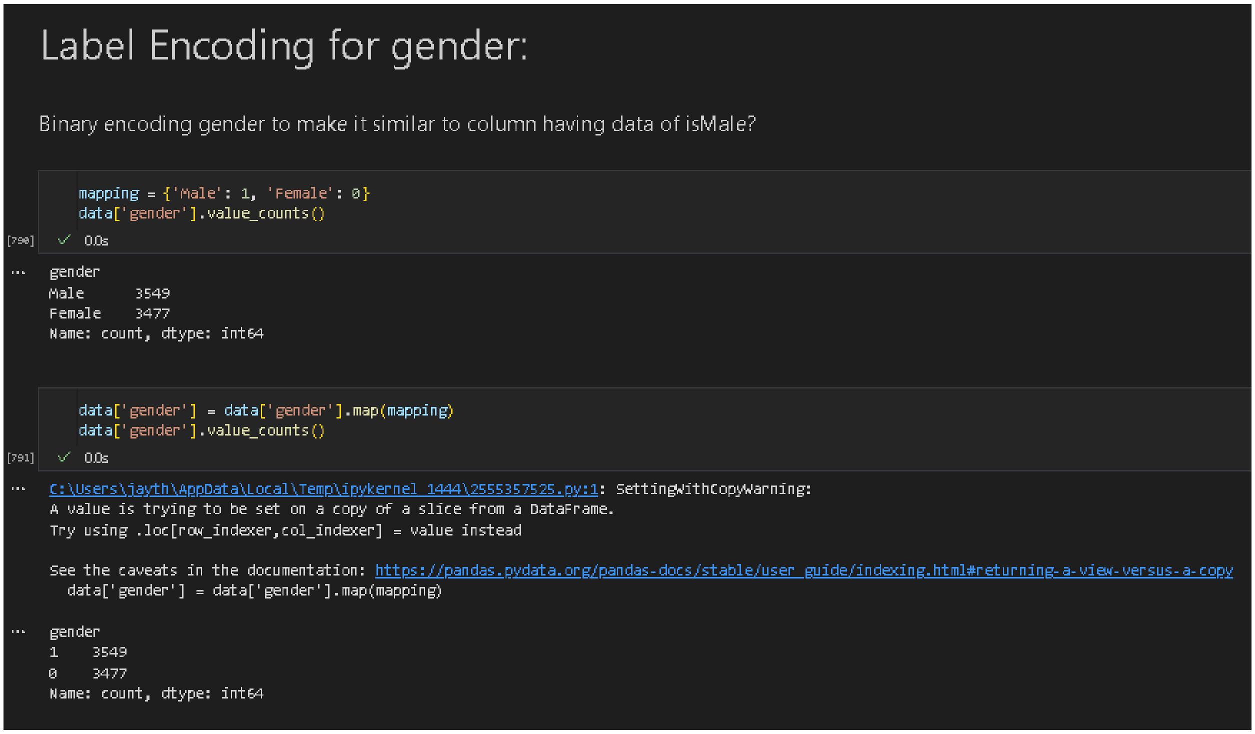


```
data[column] = data[column].map(mapping)
data[column].value_counts()
```

```
[788]  ✓ 0.0s
```

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\3990455189.py:1: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data[column] = data[column].map(mapping)
```

```
Churn
0    5161
1    1865
Name: count, dtype: int64
```



Label Encoding for gender:

Binary encoding gender to make it similar to column having data of isMale?

```
mapping = {'Male': 1, 'Female': 0}
data['gender'].value_counts()
```

[790] ✓ 0.0s

```
gender
Male      3549
Female    3477
Name: count, dtype: int64
```

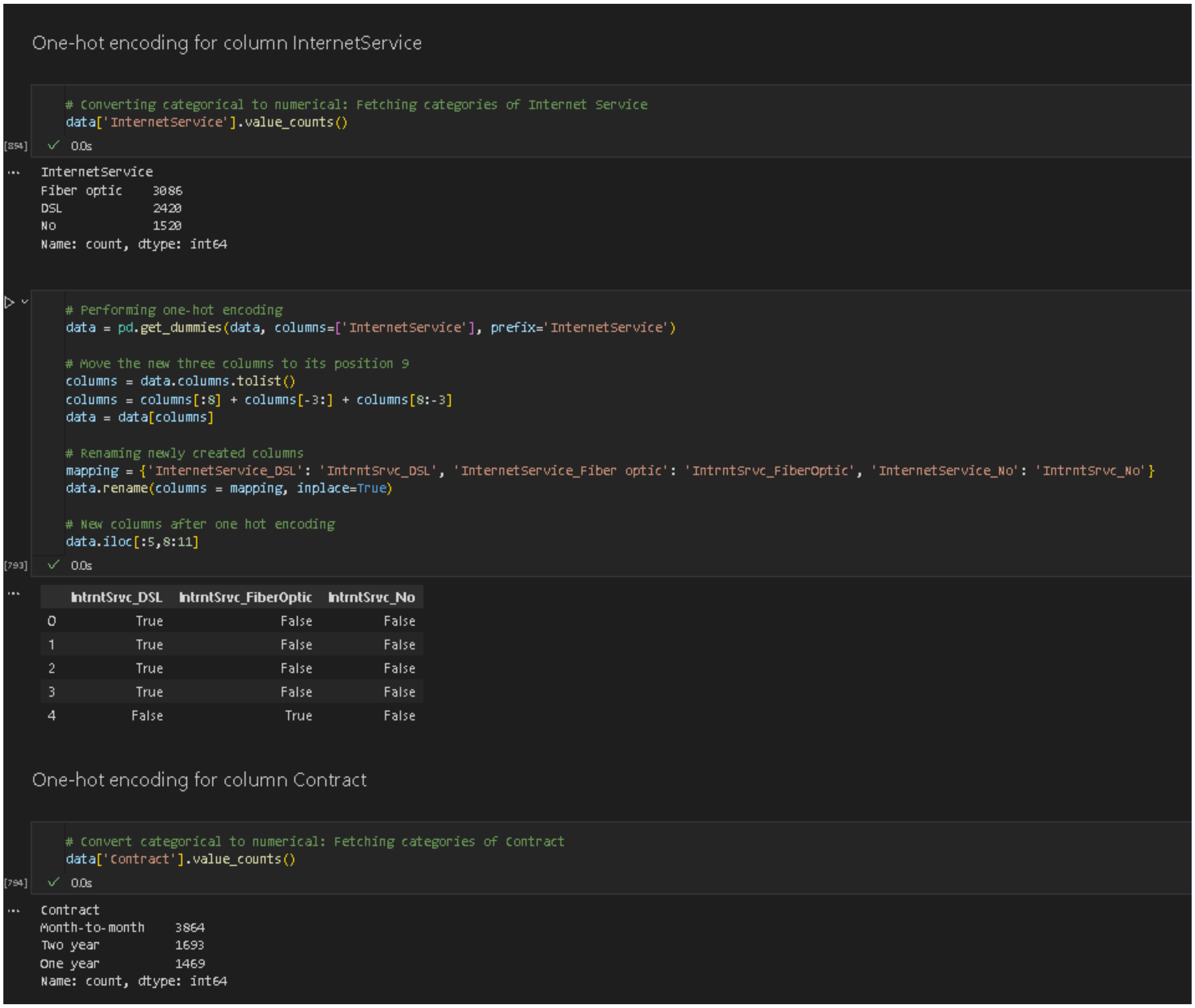
```
data['gender'] = data['gender'].map(mapping)
data['gender'].value_counts()
```

[791] ✓ 0.0s

```
C:\Users\jayth\AppData\Local\Temp\ipykernel_1444\2555357525.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['gender'] = data['gender'].map(mapping)
```

```
gender
1    3549
0    3477
Name: count, dtype: int64
```

- **One - Hot Encoding**
  - One-hot encoding is essential for classification jobs because it converts categorical data into a format machine-learning algorithms can understand. To make predictions in classification tasks, algorithms require numerical input characteristics, which one-hot encoding does by expressing category variables as binary columns. Each category is transformed into a distinct binary column, allowing the algorithm to consider each individually without indicating any ordinal connection. This encoding prevents the algorithm from assigning unwanted meaning or hierarchy to the categories and increases the model's ability to catch patterns and generate accurate predictions on categorical data.



One-hot encoding for column InternetService

```
# Converting categorical to numerical: Fetching categories of Internet Service
data['InternetService'].value_counts()
```

```
InternetService
Fiber optic    3086
DSL            2420
No             1520
Name: count, dtype: int64
```

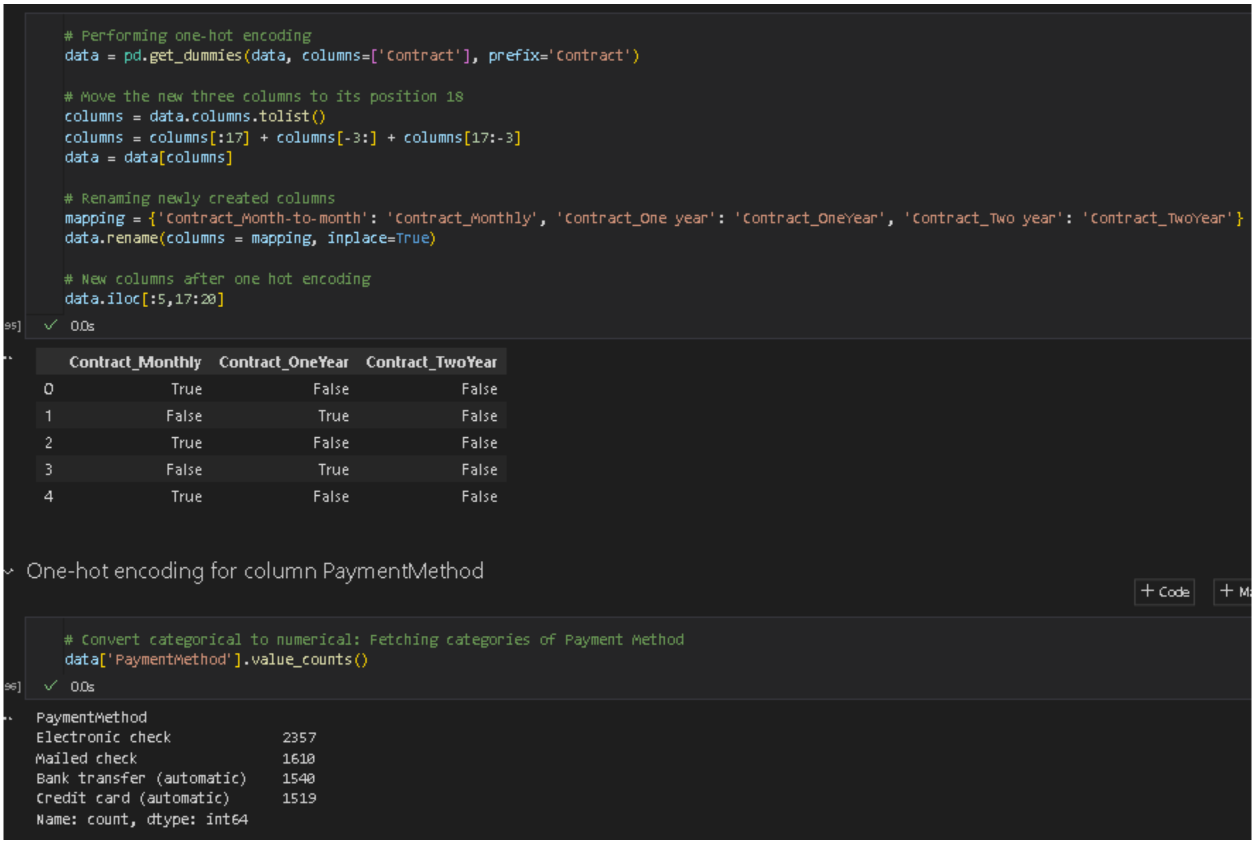
```
# Performing one-hot encoding
data = pd.get_dummies(data, columns=['InternetService'], prefix='InternetService')

# Move the new three columns to its position 9
columns = data.columns.tolist()
columns = columns[:8] + columns[-3:] + columns[8:-3]
data = data[columns]

# Renaming newly created columns
mapping = {'InternetService_DSL': 'IntrntSrvc_DSL', 'InternetService_Fiber optic': 'IntrntSrvc_FiberOptic', 'InternetService_No': 'IntrntSrvc_No'}
data.rename(columns = mapping, inplace=True)

# New columns after one hot encoding
data.iloc[:5,8:11]
```

	IntrntSrvc_DSL	IntrntSrvc_FiberOptic	IntrntSrvc_No
0	True	False	False
1	True	False	False
2	True	False	False
3	True	False	False
4	False	True	False

One-hot encoding for column Contract

```
# Convert categorical to numerical: Fetching categories of Contract
data['Contract'].value_counts()
```

```
Contract
Month-to-month    3864
Two year          1693
One year          1469
Name: count, dtype: int64
```



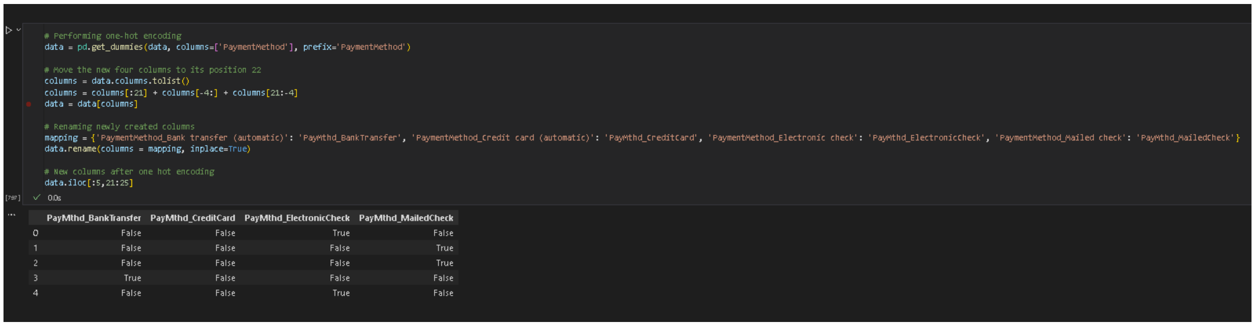
```python
# Performing one-hot encoding
data = pd.get_dummies(data, columns=['Contract'], prefix='Contract')

# Move the new three columns to its position 18
columns = data.columns.tolist()
columns = columns[:17] + columns[-3:] + columns[17:-3]
data = data[columns]

# Renaming newly created columns
mapping = {'Contract_Month-to-month': 'Contract_Monthly', 'Contract_One year': 'Contract_OneYear', 'Contract_Two year': 'Contract_TwoYear'}
data.rename(columns = mapping, inplace=True)

# New columns after one hot encoding
data.iloc[:5,17:20]
```

	Contract_Monthly	Contract_OneYear	Contract_TwoYear
0	True	False	False
1	False	True	False
2	True	False	False
3	False	True	False
4	True	False	False

One-hot encoding for column PaymentMethod

```python
# Convert categorical to numerical: Fetching categories of Payment Method
data['PaymentMethod'].value_counts()
```

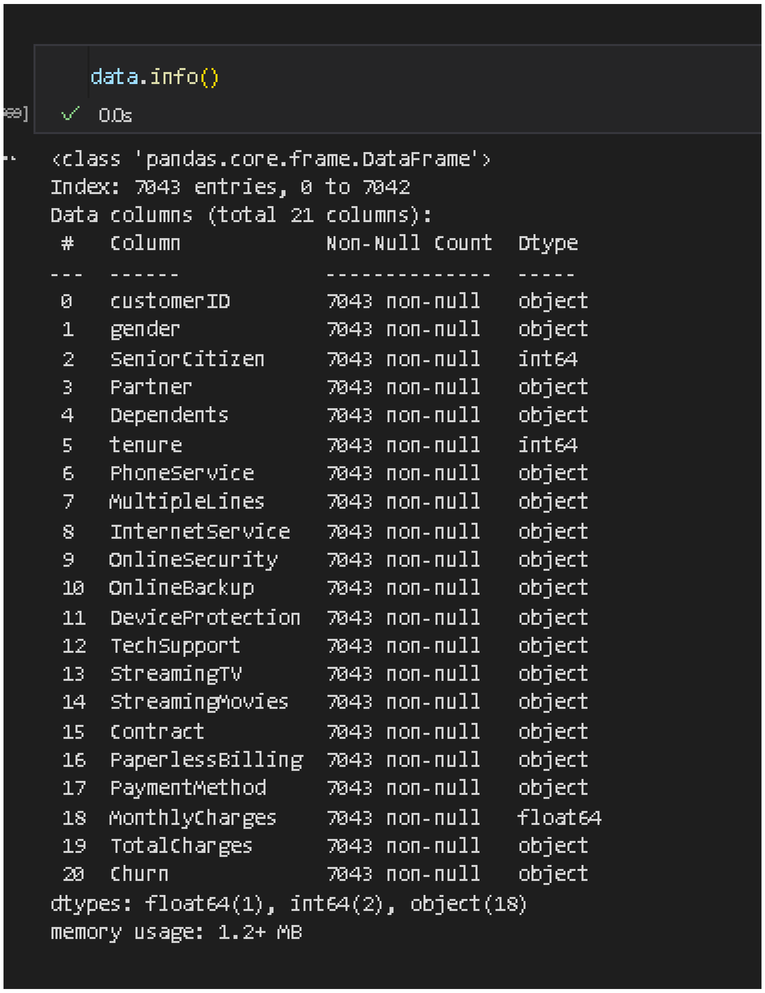
```
PaymentMethod
Electronic check             2357
Mailed check                 1610
Bank transfer (automatic)    1540
Credit card (automatic)      1519
Name: count, dtype: int64
```



```python
# Performing one-hot encoding
data = pd.get_dummies(data, columns=['PaymentMethod'], prefix='PaymentMethod')

# Move the new four columns to its position 22
columns = data.columns.tolist()
columns = columns[:21] + columns[-4:] + columns[21:-4]
data = data[columns]

# Renaming newly created columns
mapping = {'PaymentMethod_Bank transfer (automatic)': 'PayMthd_BankTransfer', 'PaymentMethod_Credit card (automatic)': 'PayMthd_CreditCard', 'PaymentMethod_Electronic check': 'PayMthd_ElectronicCheck', 'PaymentMethod_Mailed check': 'PayMthd_MailedCheck'}
data.rename(columns = mapping, inplace=True)

# New columns after one hot encoding
data.iloc[:5,21:25]
```

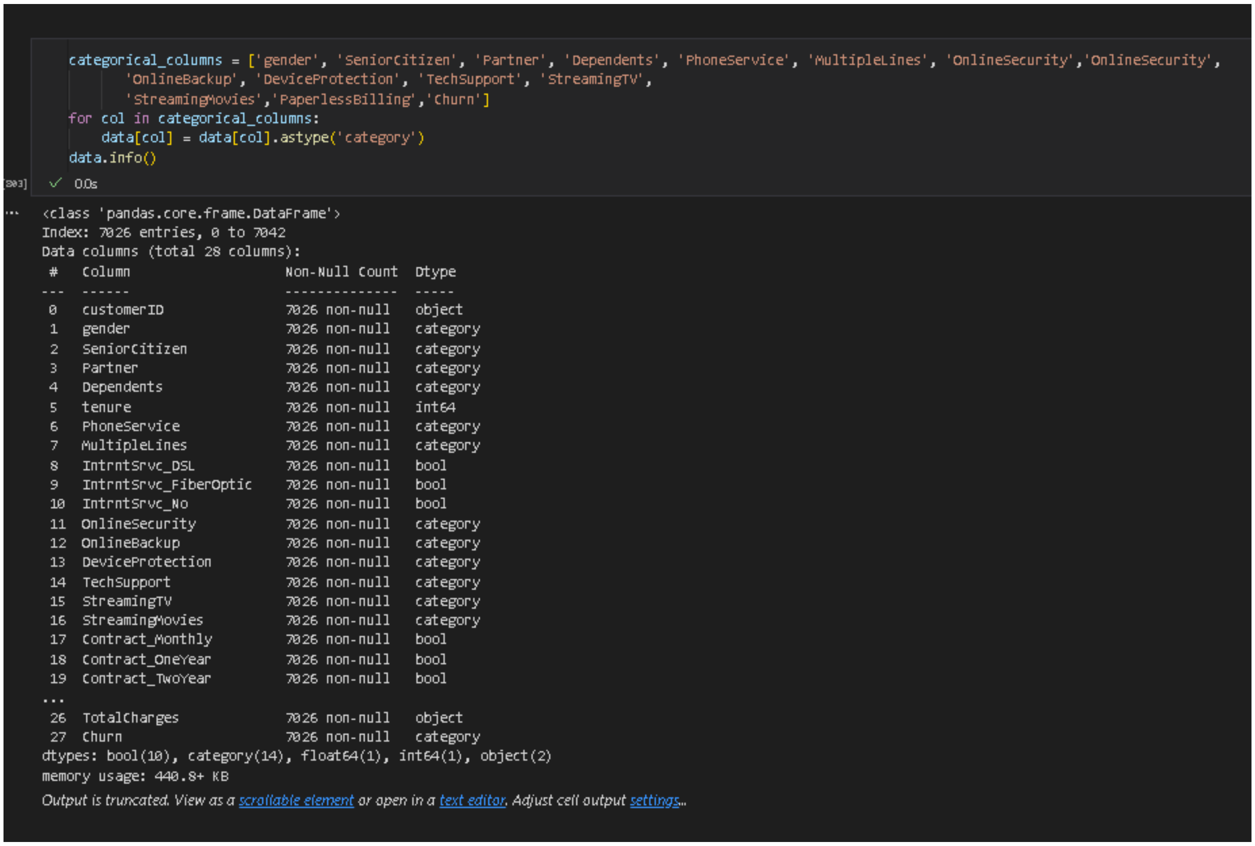
	PayMthd_BankTransfer	PayMthd_CreditCard	PayMthd_ElectronicCheck	PayMthd_MailedCheck
0	False	False	True	False
1	False	False	False	True
2	False	False	False	True
3	True	False	False	False
4	False	False	True	False

- **Typecasting Categorical columns to data type 'category'**
  - Since many of our columns are of the categorical type, storing them as 'object' type is poor practice, thus converting them to the dtype 'category.'
  - Before:



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.2+ MB
```

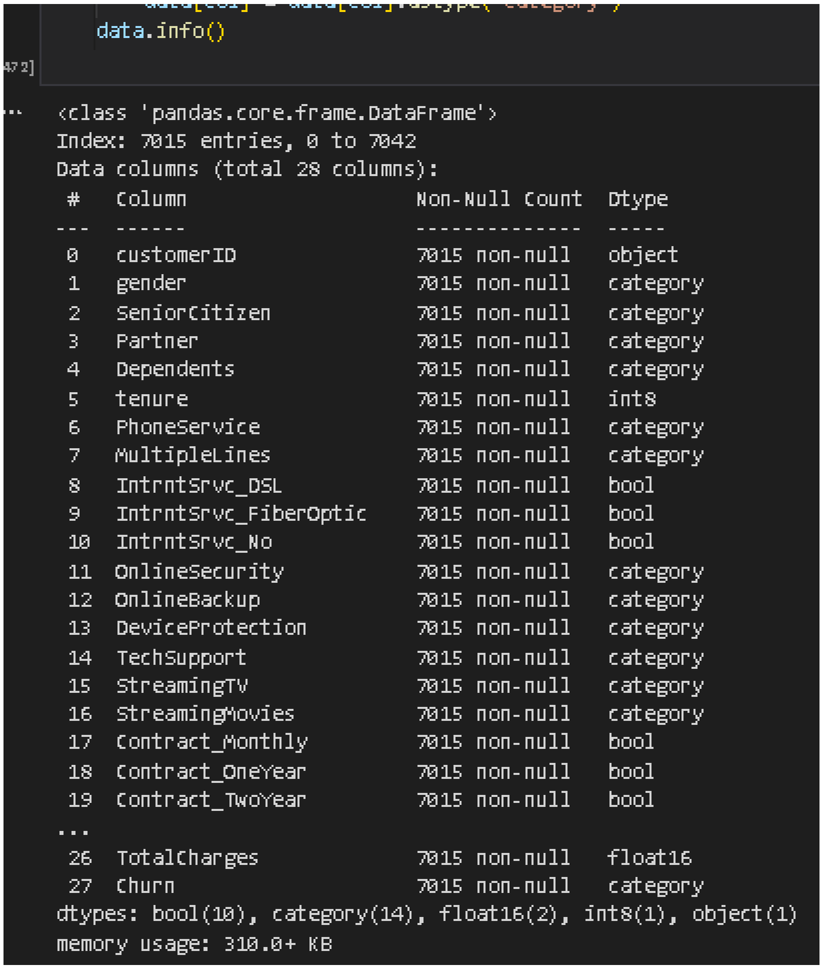


```python
categorical_columns = ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'OnlineSecurity','OnlineSecurity',
        'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
        'StreamingMovies','PaperlessBilling','Churn']
for col in categorical_columns:
    data[col] = data[col].astype('category')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7026 entries, 0 to 7042
Data columns (total 28 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   customerID                7026 non-null   object
 1   gender                    7026 non-null   category
 2   SeniorCitizen             7026 non-null   category
 3   Partner                   7026 non-null   category
 4   Dependents                7026 non-null   category
 5   tenure                    7026 non-null   int64
 6   PhoneService              7026 non-null   category
 7   MultipleLines             7026 non-null   category
 8   IntrntSrvc_DSL            7026 non-null   bool
 9   IntrntSrvc_FiberOptic     7026 non-null   bool
 10  IntrntSrvc_No             7026 non-null   bool
 11  OnlineSecurity            7026 non-null   category
 12  OnlineBackup              7026 non-null   category
 13  DeviceProtection          7026 non-null   category
 14  TechSupport               7026 non-null   category
 15  StreamingTV               7026 non-null   category
 16  StreamingMovies           7026 non-null   category
 17  Contract_Monthly          7026 non-null   bool
 18  Contract_OneYear          7026 non-null   bool
 19  Contract_TwoYear          7026 non-null   bool
...
 26  TotalCharges              7026 non-null   object
 27  Churn                     7026 non-null   category
dtypes: bool(10), category(14), float64(1), int64(1), object(2)
memory usage: 440.8+ KB
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

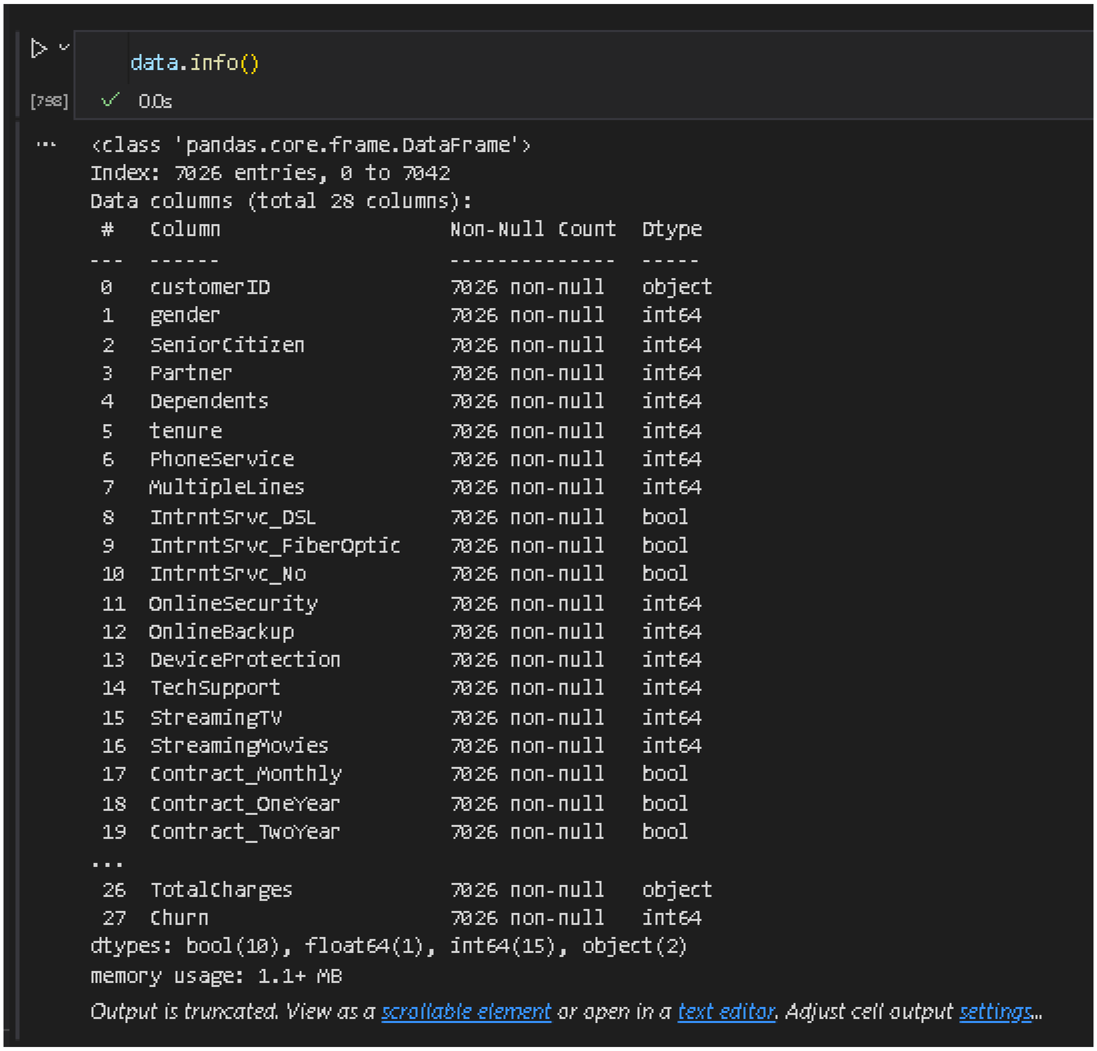
- After:



```
        data[col] = data[col].astype('category')
    data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7015 entries, 0 to 7042
Data columns (total 28 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   customerID             7015 non-null   object
 1   gender                 7015 non-null   category
 2   SeniorCitizen          7015 non-null   category
 3   Partner                7015 non-null   category
 4   Dependents             7015 non-null   category
 5   tenure                 7015 non-null   int8
 6   PhoneService           7015 non-null   category
 7   MultipleLines          7015 non-null   category
 8   IntrntSrvc_DSL         7015 non-null   bool
 9   IntrntSrvc_FiberOptic  7015 non-null   bool
 10  IntrntSrvc_No          7015 non-null   bool
 11  OnlineSecurity         7015 non-null   category
 12  OnlineBackup           7015 non-null   category
 13  DeviceProtection       7015 non-null   category
 14  TechSupport            7015 non-null   category
 15  StreamingTV            7015 non-null   category
 16  StreamingMovies        7015 non-null   category
 17  Contract_Monthly       7015 non-null   bool
 18  Contract_OneYear       7015 non-null   bool
 19  Contract_TwoYear       7015 non-null   bool
...
 26  TotalCharges           7015 non-null   float16
 27  Churn                  7015 non-null   category
dtypes: bool(10), category(14), float16(2), int8(1), object(1)
memory usage: 310.0+ KB
```

- **Datatype downcasting**
  - Storing data such as int64,float64, etc., not only requires more space but also increases processing times.
  - In such scenarios, downcasting will stop the wastage of space and improve data processing times during the training phases of our predictive models.
  - Before:

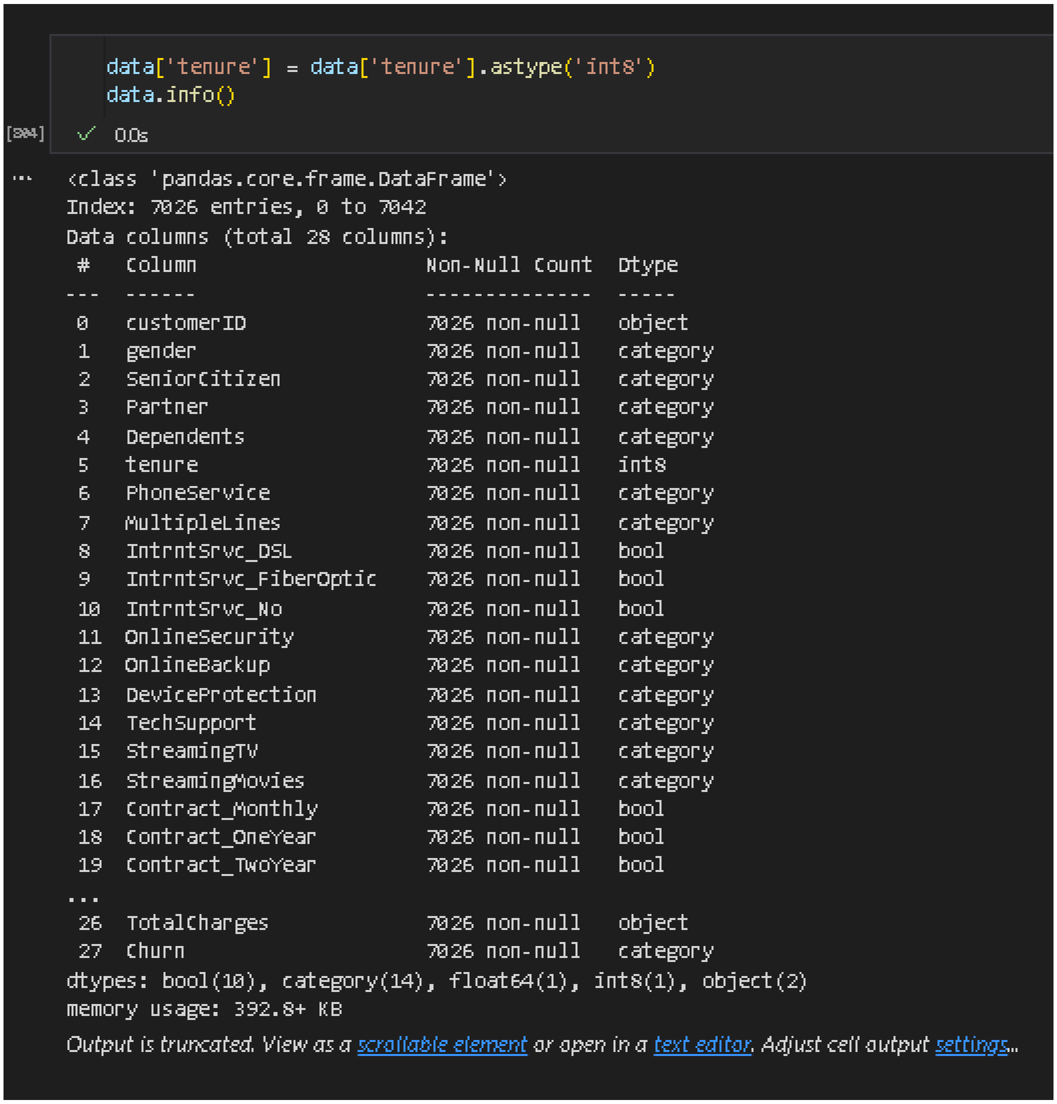


```
data.info()
```

[798] ✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
Index: 7026 entries, 0 to 7042
Data columns (total 28 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   customerID             7026 non-null   object
 1   gender                 7026 non-null   int64
 2   SeniorCitizen          7026 non-null   int64
 3   Partner                7026 non-null   int64
 4   Dependents             7026 non-null   int64
 5   tenure                 7026 non-null   int64
 6   PhoneService           7026 non-null   int64
 7   MultipleLines          7026 non-null   int64
 8   IntrntSrvc_DSL         7026 non-null   bool
 9   IntrntSrvc_FiberOptic  7026 non-null   bool
 10  IntrntSrvc_No          7026 non-null   bool
 11  OnlineSecurity         7026 non-null   int64
 12  OnlineBackup           7026 non-null   int64
 13  DeviceProtection       7026 non-null   int64
 14  TechSupport            7026 non-null   int64
 15  StreamingTV            7026 non-null   int64
 16  StreamingMovies        7026 non-null   int64
 17  Contract_Monthly       7026 non-null   bool
 18  Contract_OneYear       7026 non-null   bool
 19  Contract_TwoYear       7026 non-null   bool
...
 26  TotalCharges           7026 non-null   object
 27  Churn                  7026 non-null   int64
dtypes: bool(10), float64(1), int64(15), object(2)
memory usage: 1.1+ MB
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



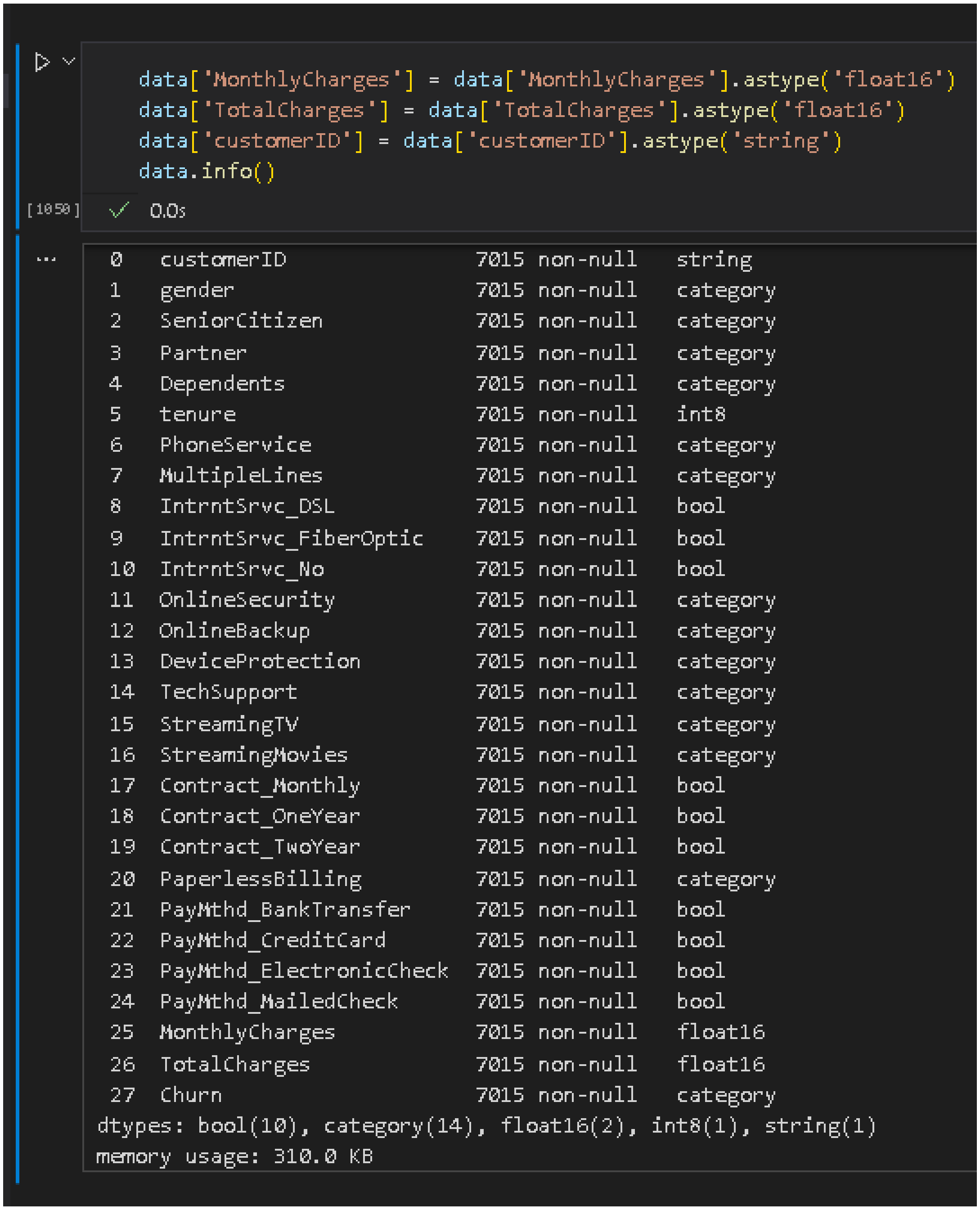
```
data['tenure'] = data['tenure'].astype('int8')
data.info()
```

[304] ✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
Index: 7026 entries, 0 to 7042
Data columns (total 28 columns):
 #   Column                Non-Null Count  Dtype   
---  ------                --------------  -----   
 0   customerID            7026 non-null   object  
 1   gender                7026 non-null   category
 2   SeniorCitizen         7026 non-null   category
 3   Partner               7026 non-null   category
 4   Dependents            7026 non-null   category
 5   tenure                7026 non-null   int8    
 6   PhoneService          7026 non-null   category
 7   MultipleLines         7026 non-null   category
 8   IntrntSrvc_DSL        7026 non-null   bool    
 9   IntrntSrvc_FiberOptic 7026 non-null   bool    
 10  IntrntSrvc_No         7026 non-null   bool    
 11  OnlineSecurity        7026 non-null   category
 12  OnlineBackup          7026 non-null   category
 13  DeviceProtection      7026 non-null   category
 14  TechSupport           7026 non-null   category
 15  StreamingTV           7026 non-null   category
 16  StreamingMovies       7026 non-null   category
 17  Contract_Monthly      7026 non-null   bool    
 18  Contract_OneYear      7026 non-null   bool    
 19  Contract_TwoYear      7026 non-null   bool    
...
 26  TotalCharges          7026 non-null   object  
 27  Churn                 7026 non-null   category
dtypes: bool(10), category(14), float64(1), int8(1), object(2)
memory usage: 392.8+ KB
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

- After:



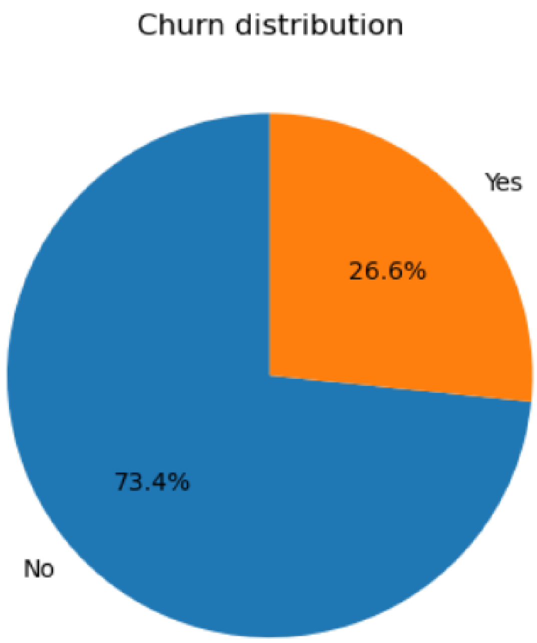
```python
data['MonthlyCharges'] = data['MonthlyCharges'].astype('float16')
data['TotalCharges'] = data['TotalCharges'].astype('float16')
data['customerID'] = data['customerID'].astype('string')
data.info()
```

```
 0   customerID               7015 non-null   string
 1   gender                   7015 non-null   category
 2   SeniorCitizen            7015 non-null   category
 3   Partner                  7015 non-null   category
 4   Dependents               7015 non-null   category
 5   tenure                   7015 non-null   int8
 6   PhoneService             7015 non-null   category
 7   MultipleLines            7015 non-null   category
 8   IntrntSrvc_DSL           7015 non-null   bool
 9   IntrntSrvc_FiberOptic    7015 non-null   bool
 10  IntrntSrvc_No            7015 non-null   bool
 11  OnlineSecurity           7015 non-null   category
 12  OnlineBackup             7015 non-null   category
 13  DeviceProtection         7015 non-null   category
 14  TechSupport              7015 non-null   category
 15  StreamingTV              7015 non-null   category
 16  StreamingMovies          7015 non-null   category
 17  Contract_Monthly         7015 non-null   bool
 18  Contract_OneYear         7015 non-null   bool
 19  Contract_TwoYear         7015 non-null   bool
 20  PaperlessBilling         7015 non-null   category
 21  PayMthd_BankTransfer     7015 non-null   bool
 22  PayMthd_CreditCard       7015 non-null   bool
 23  PayMthd_ElectronicCheck  7015 non-null   bool
 24  PayMthd_MailedCheck      7015 non-null   bool
 25  MonthlyCharges           7015 non-null   float16
 26  TotalCharges             7015 non-null   float16
 27  Churn                    7015 non-null   category
dtypes: bool(10), category(14), float16(2), int8(1), string(1)
memory usage: 310.0 KB
```

EDA

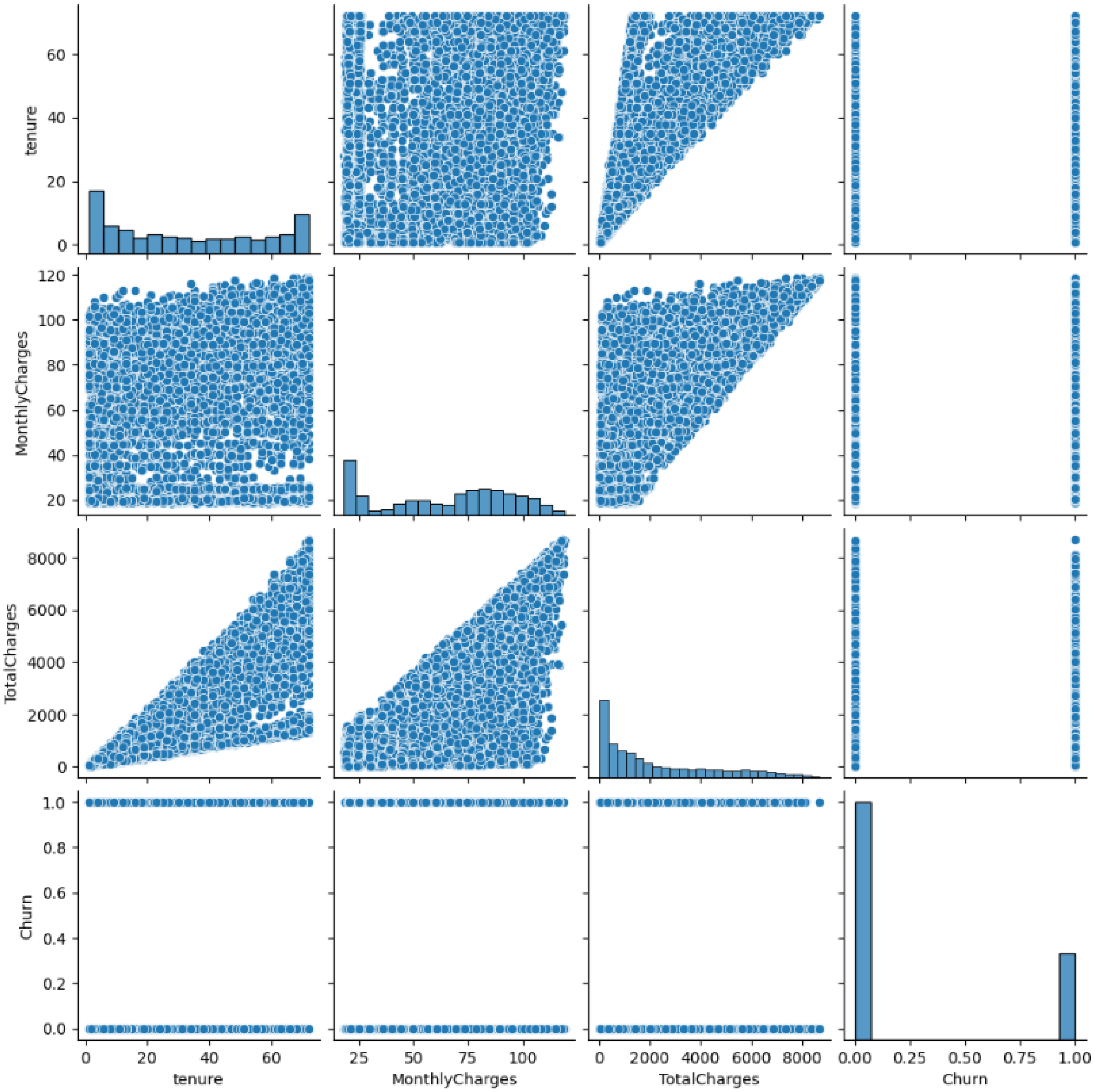
Exploratory Data Analysis (EDA) for the Telecom Customer churn dataset comprises initial data exploration, pattern identification, and understanding of significant factors affecting customer retention, i.e., customer churn. EDA provides insights such as the distribution of churn, correlations between features, and possible predictors for the churn utilizing graphical representations, analysis of data, and various plots, charts, and graphs.

- **Churn Distribution**



We can see from the churn distribution pie chart that 26.6% of customers churned while the rest 73.4% of the customers stayed with the company. It can be induced from this pie chart that more than a quarter of the customers have left in the past month.

- **Pair plots of some crucial features**



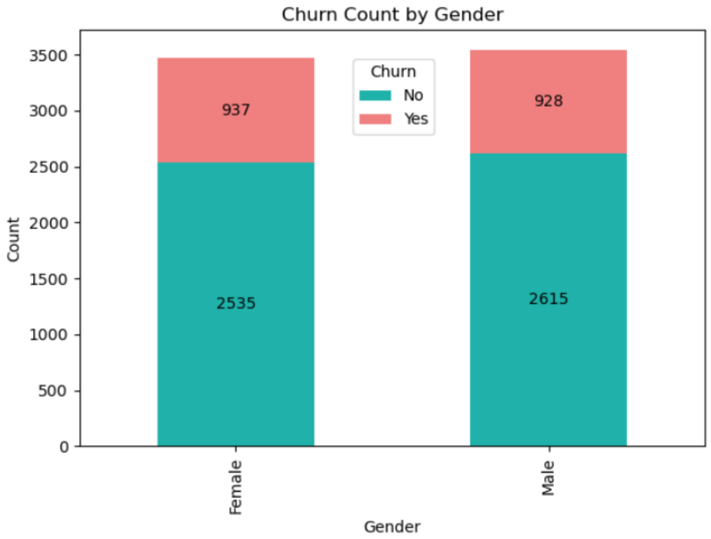
Customer's analysis

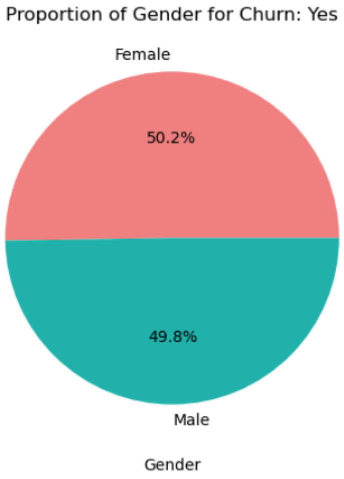
The data has various features of the Customer. These features include their:

- **Gender**: Customer is male or female
- **Dependent**: Does the Customer have dependents or not
- **Partner**: Does the Customer have a partner or not
- **Senior Citizen**: Is the Customer is a senior citizen or not

Analyzing each feature separately:

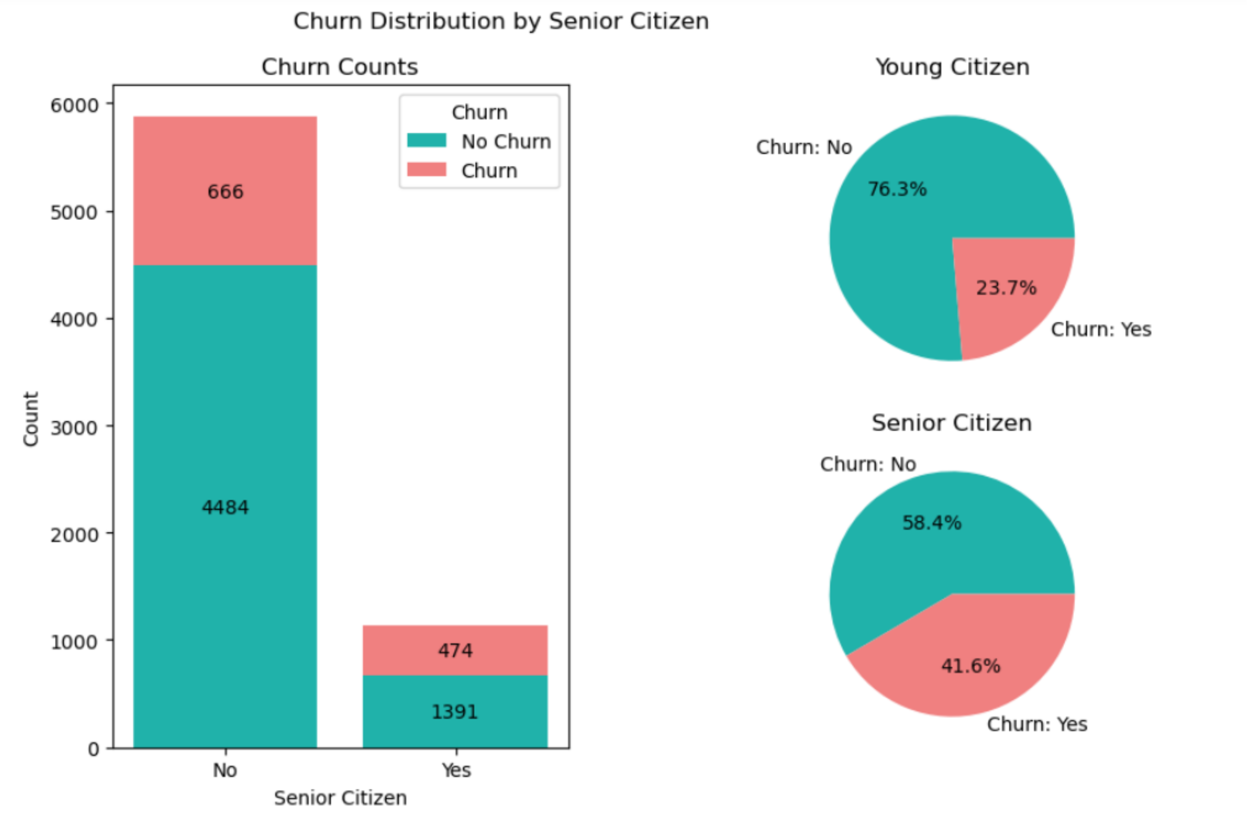
- **Churn distribution by Gender**





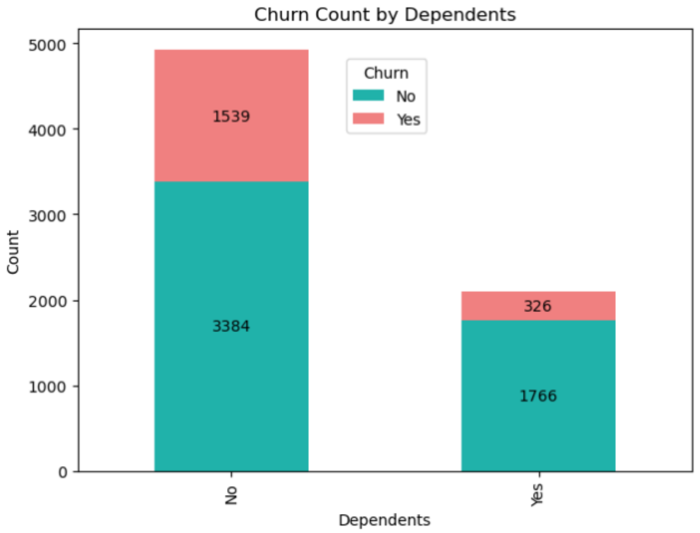
No particular trend was observed based on Gender for churn prediction. Both the genders are equally likely to churn.

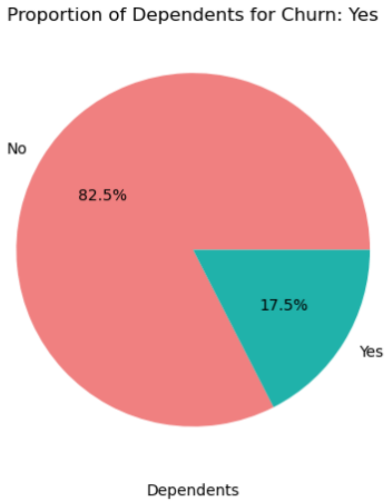
- **Churn distribution by Senior Citizen**



The bar plot unmistakably illustrates that young people, as opposed to senior citizens, have a higher contribution to churn in terms of count. However, the pie charts reveal a distinct pattern when we examine each category individually. Approximately 42% of senior citizens left, nearly double that of young citizens who left (Senior Citizens = No).

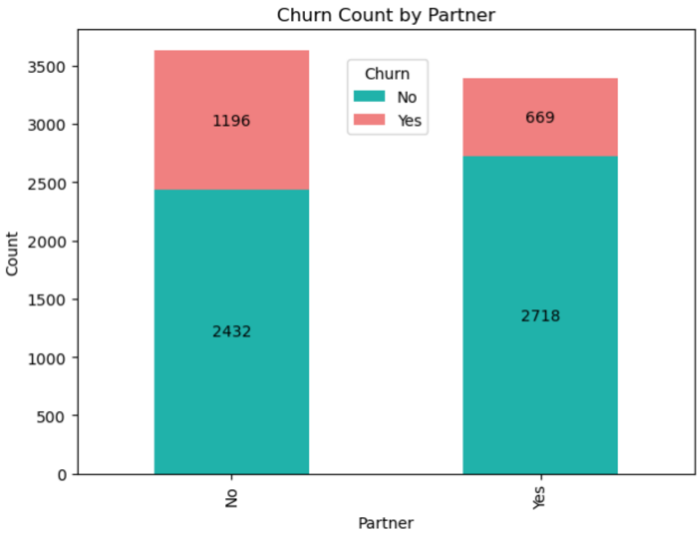
- **Churn distribution by Dependent**

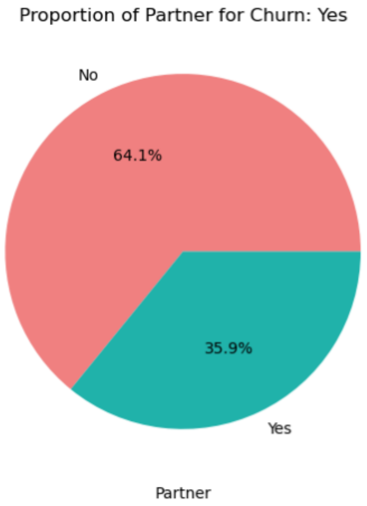




The majority of the customers do not have dependents and customers who do not appear more likely to churn than those who have dependents.

- **Churn distribution by Partner**





Customers who do not have partners (single) appear to contribute more towards the churn.

- ### Insights from Customer's Analysis

  It was discovered that gender and relationship status are pretty evenly distributed within the client base, with approximate percentage values based on the study. While females have a somewhat greater turnover rate, this difference is modest and may not be statistically significant.

  When diving further into the details, though, a noteworthy tendency emerges. Younger consumers, consumers without partners, and consumers without dependents have a greater turnover rate. Based on the data study, these specific categories of the consumer population stand out as being more prone to churn.

  Our findings, in particular, highlight the importance of non-senior citizens without partners or dependents as a separate client niche worthy of consideration when developing customer retention tactics.

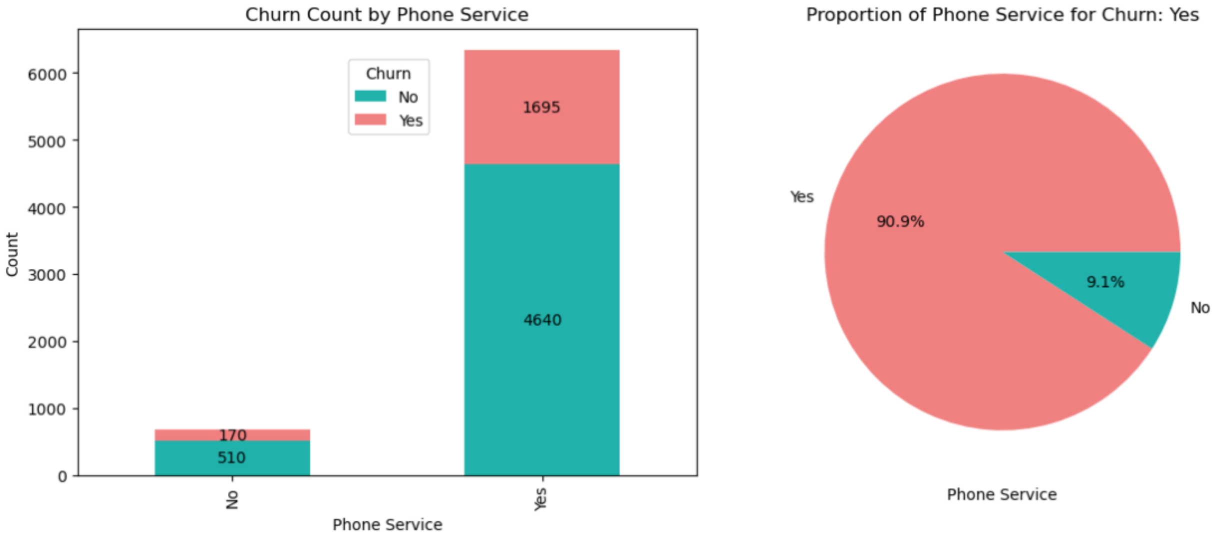
- ## Customer’s Subscription Service Analysis

The dataset has various details of the various services subscribed by the Customer.
These subscription services include various columns as follows:

- **Phone Service**: Does the Customer have phone service or not
- **Multiple Lines**: Does the Customer have multiple lines or not if he has phone service
- **Online Security**: Does the Customer has online security or not if he has internet service
- **Online Backup**: Does the Customer have online backup or not if he has internet service
- **Device Protection**: Does the Customer have device protection or not if he has internet service
- **Tech Support**: Does the Customer have tech support or not if he has internet service
- **Streaming TV**: Does the Customer have streaming TV or not if he has internet service
- **Streaming Movies**: Does the Customer have streaming movies or not if he has internet service

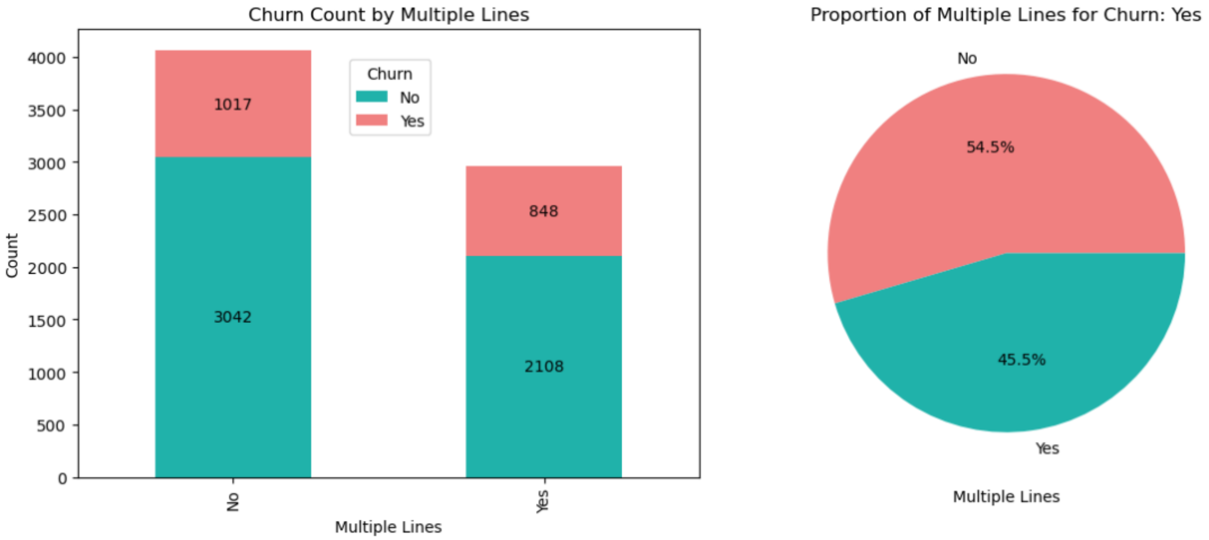
Analyzing each feature separately:

- **Churn distribution by Phone Service**



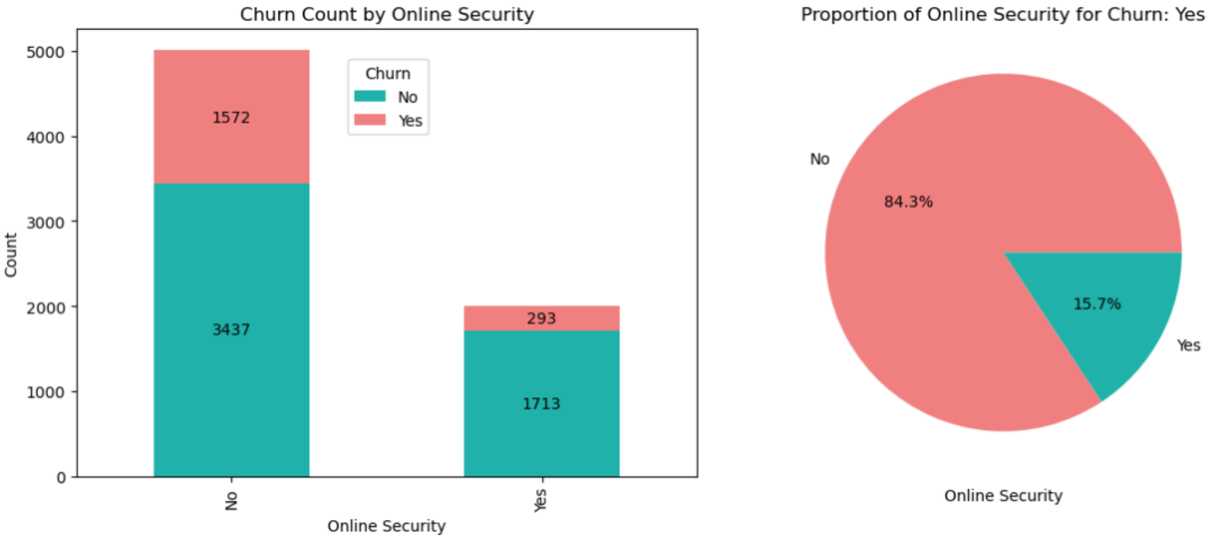
More than 90% of the customers have phone service.

- **Churn distribution by Multiple Lines**



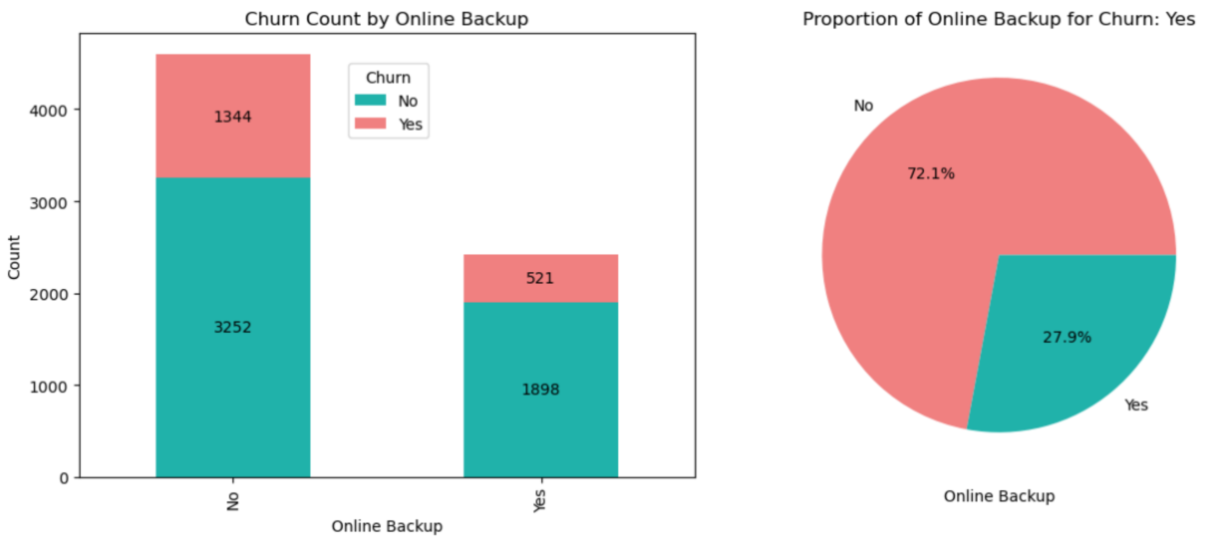
Customers who do not have multiple lines are more likely to be retained than those with multiple lines.

- **Churn distribution by Online Security**



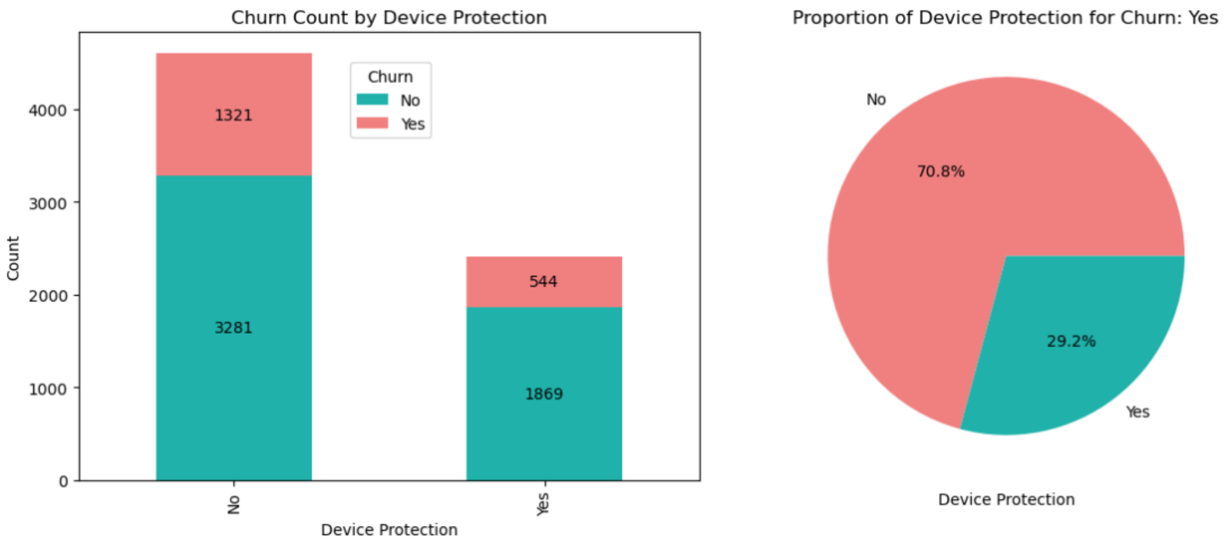
Roughly 5 out of 7 customers need online security, and these customers have higher chances of churning than those with online security.

- **Churn distribution by Online Backup**



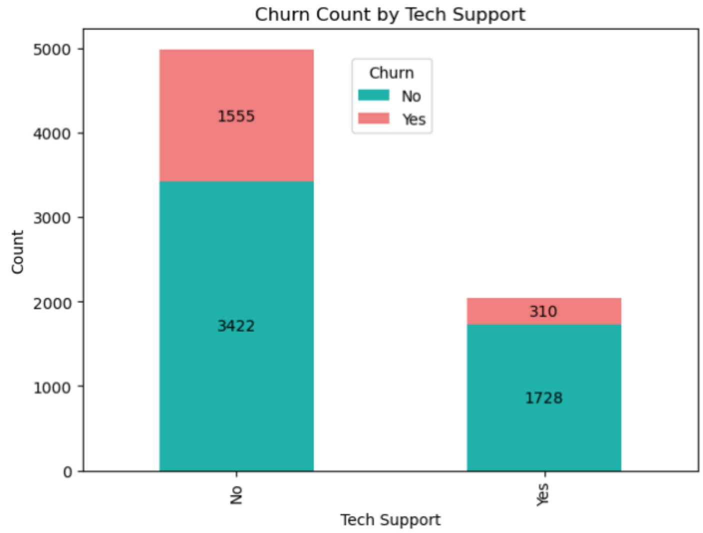
Just like Online security, customers who do not have online backup are churning more.

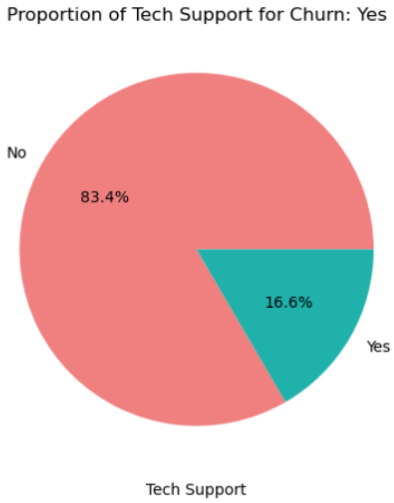
- **Churn distribution by Device Protection**



More than half of the customers do not have device protection, and such customers are more likely to churn.

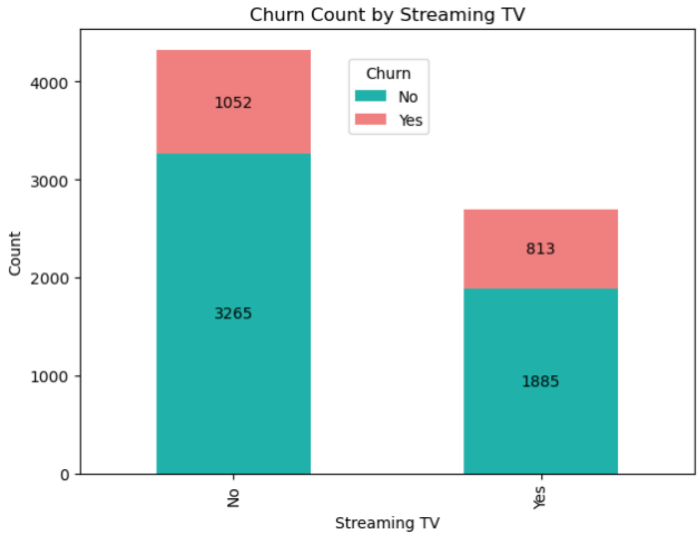
- **Churn distribution by Tech Support**

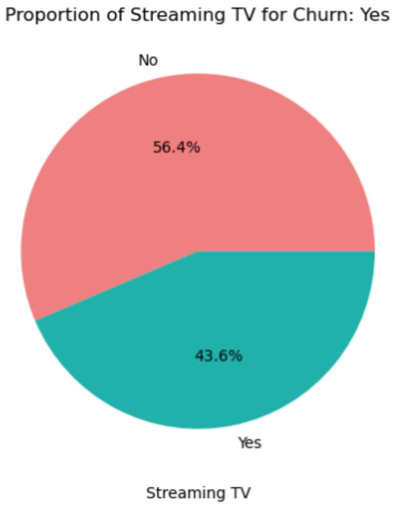




Approximately 5 out of 7 customers opt for something other than tech support. These customers are more likely to churn.

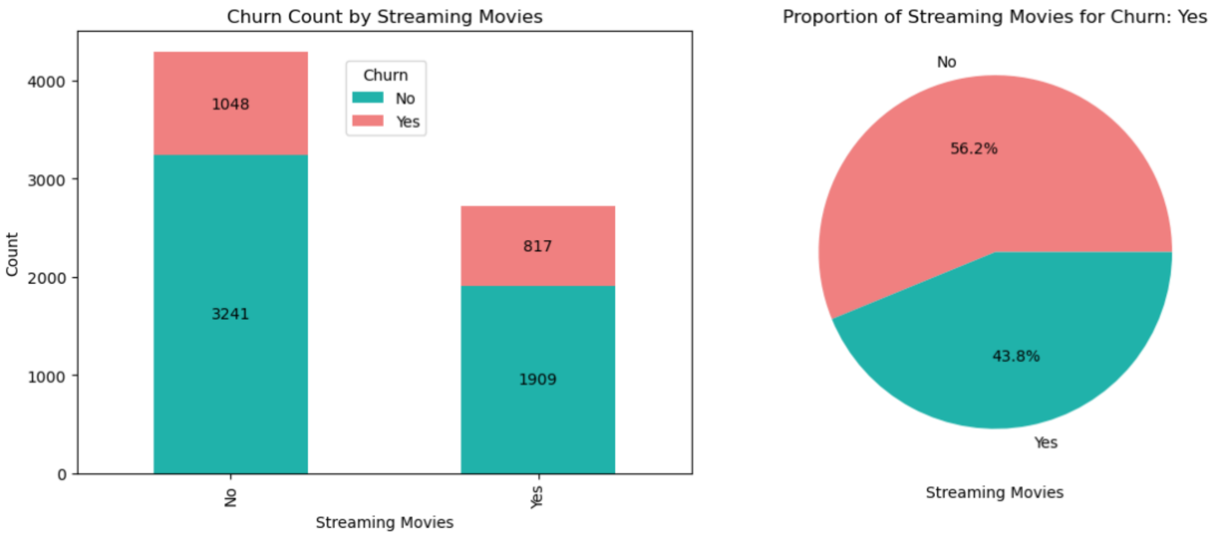
- **Churn distribution by Streaming TV**





More than half of the customers (exactly 61.5%) do not have Streaming TV service, and such customers are slightly more likely to churn than the rest.

- **Churn distribution by Streaming Movies**



For Streaming Movies, the same trend can be observed as streaming TV, where more than half (precisely 61.1%) of customers do not have Streaming Movies and are slightly more likely to churn.

- **Insights from Customer Subscription Service**

Our examination of customer service subscriptions found substantial differences among different service offerings. Notably, the following tendencies may be identified:

- **Dependency on Phone Service**: It should be noted that clients need phone service to have several lines. Phone services are used by about 90.3% of our consumers, and they have a higher turnover rate. This discovery may point to the necessity for more investigation into the causes of this unanticipated trend.

- **Fibre Optic Internet and Churn:** Customers who have chosen fiber optic as their internet service provider are more likely to churn. This can be attributable to various variables, including prospective price increases, greater competition, customer service quality, and other underlying causes. Notably, fiber optic connection is substantially more expensive than DSL, which may contribute to customer turnover.

- **Reduced Turnover Services:** Customers who have subscribed to extra services such as OnlineSecurity, OnlineBackup, DeviceProtection, and TechSupport, however, are less likely to churn. These services are essential to client retention, stressing their importance in customer retention tactics.

- **Neutrality of Streaming Services**: Surprisingly, the availability of streaming service subscribers does not predict attrition. This service is evenly distributed among consumers who select both "yes" and "no" choices, indicating that it is not a significant churn factor.

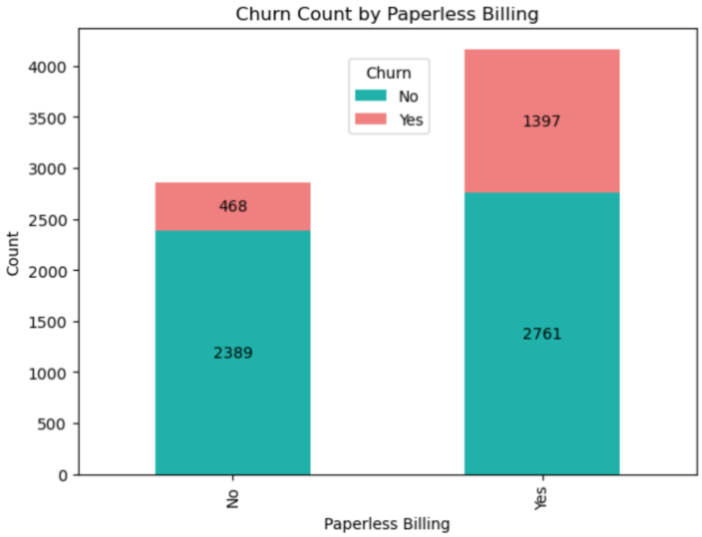
- ## Customer's Contract and Payment Analysis

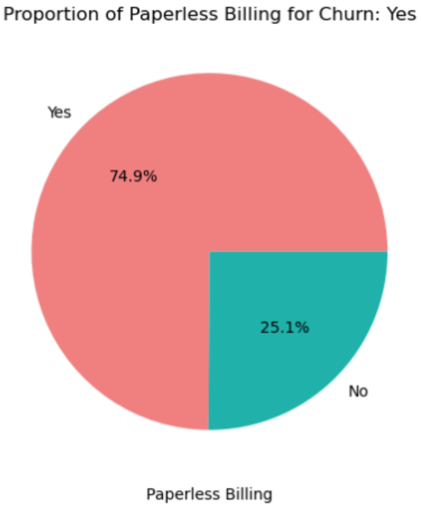
The data includes the Customer's contract duration and payment details.
These features include below columns:

- **Paperless Billing**: Does the Customer have paperless billing or not
- **Internet Services**: Customer's internet services provider (DSL et al., No ISP)
- **Contract**: Customer’s contract term (Month-to-Month, One Year, Two Year)
- **Payment Method**: Customer's Payment method (Electronic et al. (automatic), credit card (automatic))

Analyzing each feature separately:

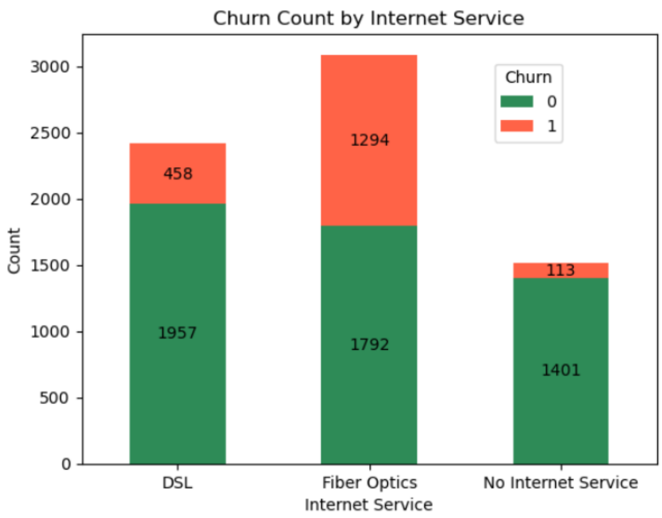
- **Churn distribution by Paperless Billing**





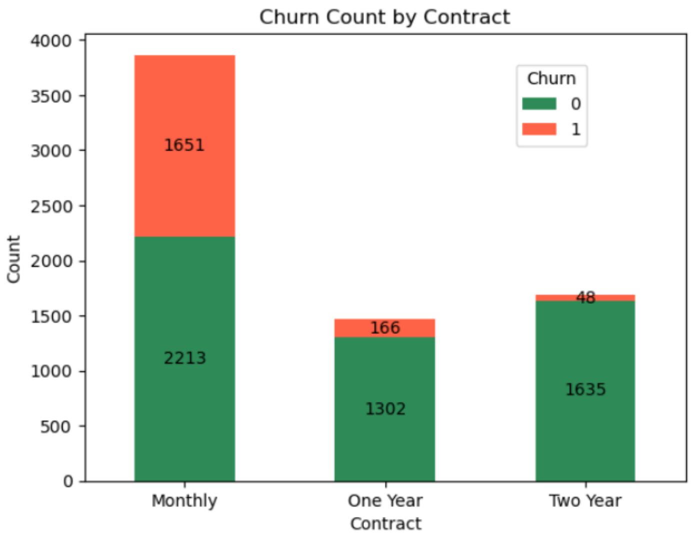
Nearly 6 out of 10 customers have gone for paperless billing. These customers are more likely to churn.

- **Churn distribution by Internet Services**



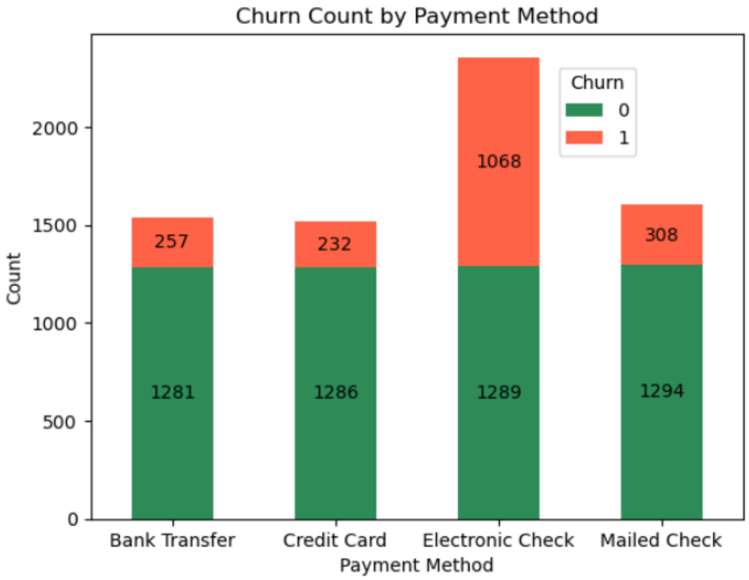
Customers who have Fiber Optics churned the most compared to people with DSL. However, people with no internet service are much less likely to churn.

- **Churn distribution by Contract**



Customers prefer short-term contracts (monthly contracts) to longer-term ones (one-year and two-year contracts). These short-term customers are majorly contributing to the churn. Customers with more extended Contract with the company are significantly less likely to churn.

- **Churn distribution by Payment Method**



Customers who pay through the Electronic check are more likely to churn than the rest of the payment methods.

- **Insights from Customer Contract and Payment**

Our examination of Customer's payment and contracts with the company revealed below trends:

- **Contract Length and Churn:** One intriguing finding is the negative association between contract length and turnover rate. Customers who have shorter contract terms are more likely to leave. On the other hand, those with longer-term obligations face extra obstacles when seeking to cancel early. This research emphasizes the need to develop long-term client connections to lower churn rates since such ties appear more robust.

- **The Impact of Paperless Billing:** It is worth noting that clients who choose paperless billing have a greater turnover rate. Paperless billing has been implemented by about 59.2% of our clients. The reasons for this correlation need further examination since it gives insight into consumer billing preferences and habits.

- **Electronic Checks and Churn**: According to one fascinating result, customers who pay with electronic checks are more likely to churn. This payment type is popular among our clients. Understanding the causes behind this correlation might be critical in devising ways to decrease attrition among electronic check consumers.

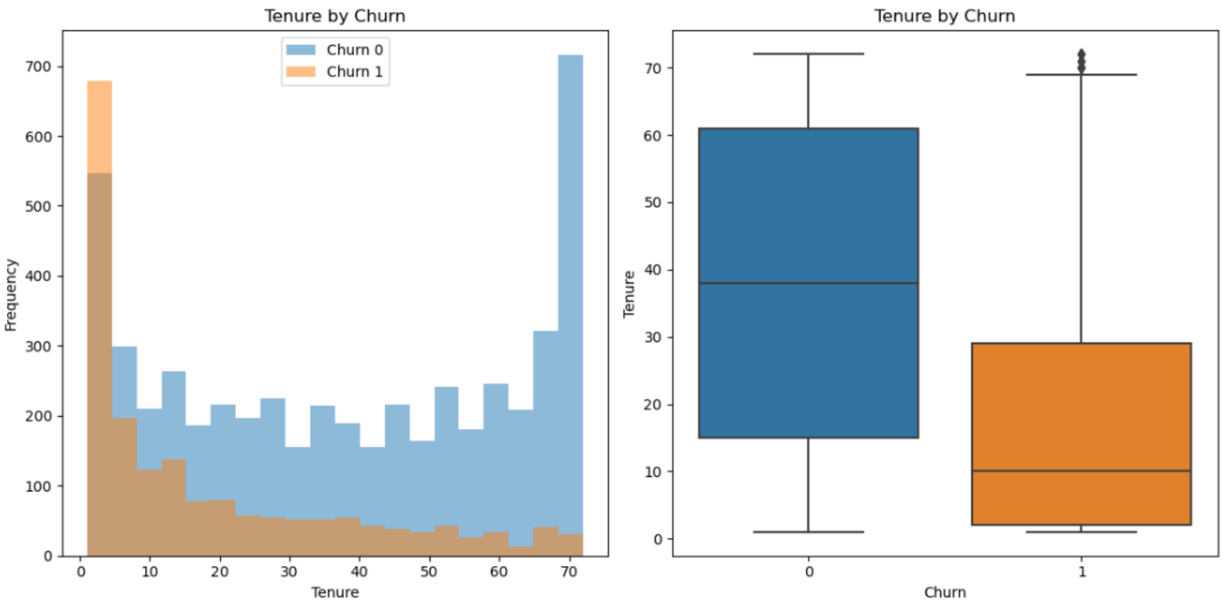
- ## Customer’s account information analysis

The data has features related to the account information of the Customer. These features include their:

- **Tenure**: Number of months the Customer has stayed with the company
- **Monthly Charge**: The amount charged to the Customer monthly
- **Total Charge**: The total amount charged to the Customer

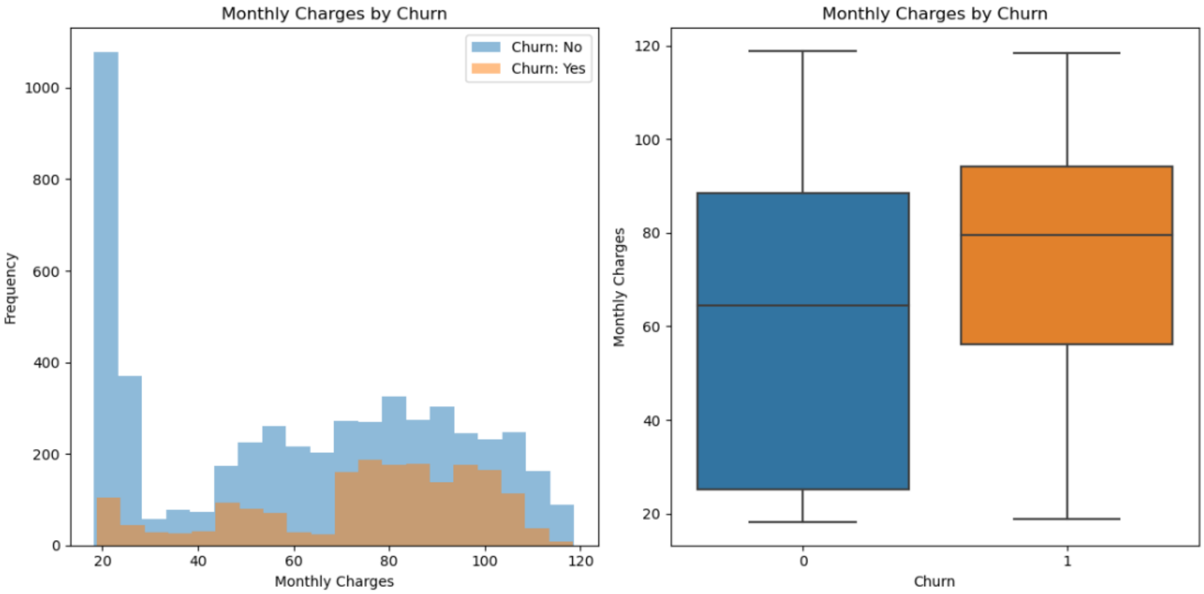
Analyzing each feature separately:

- **Churn distribution by Tenure**



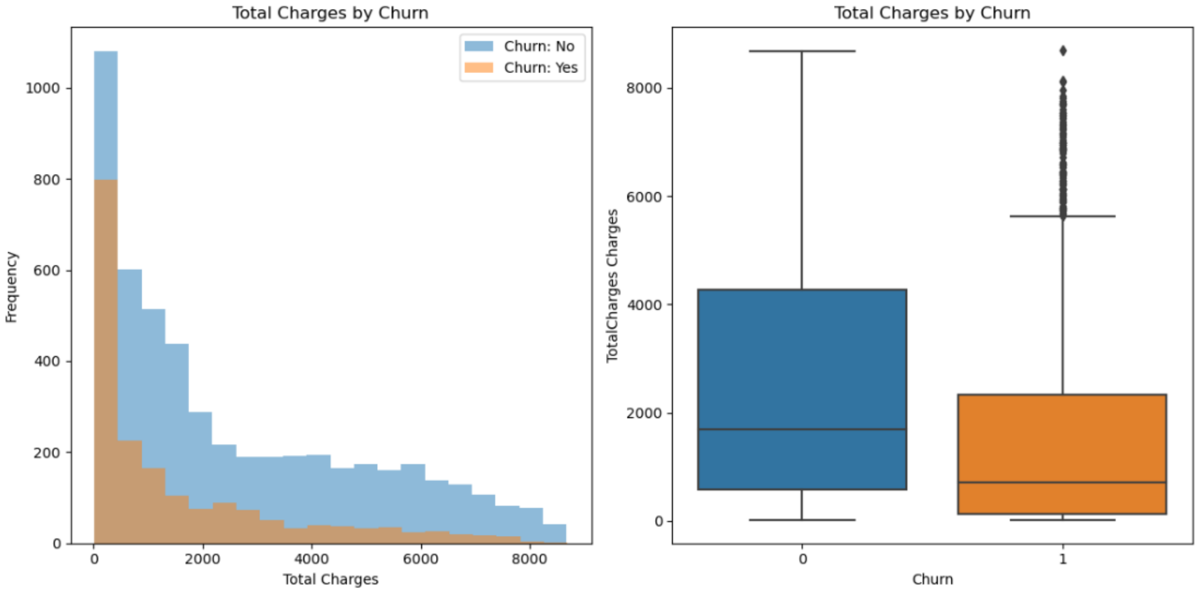
It is evident from the graphs that once people stay more than 20 years, then they are less likely to churn based on the Tenure of the customers compared to their churn rate.

- **Churn distribution by Monthly Charge**



Monthly charges are directly proportional to the churn rate, meaning the lower the monthly charges the Customer pays, the less likely the Customer is to churn.

- **Churn distribution by Total Charge**



The churn rate is inversely proportional to the total charges the Customer pays. That is, the higher the total charges paid by the Customer lower the chances of their churning.

- ### Insights from Customer account information

  Our examination of the Customer's account information has shown the below trends:

  - **Tenure Distribution:** The customer tenure histogram shows a right-skewed distribution, indicating that most consumers have only been with the telecom business for the first few months (0-9 months). This realization emphasizes the significance of efficiently maintaining consumers throughout their first few months of involvement.

  - **Churn Timing**: Surprisingly, the most significant percentage of churn happens within the first few months (0-9 months). This discovery underscores the critical period during which customer retention efforts should be concentrated to minimize churn rates effectively.

  - **Early Churn Concentration**: One significant conclusion is that around 75% of consumers who eventually quit the Telco firm do so during their first 30 months of employment. This statistic emphasizes the importance of early client interaction and satisfaction in developing long-term connections.

  - **Monthly Fees and Churn:** Our examination of the monthly charge histogram indicates an interesting pattern. Customers who pay more significant monthly fees are more likely to leave. This implies that discounts, promotions, or competitive pricing effectively motivate customers to stay loyal. Pricing methods that consider these findings may be beneficial in keeping consumers.