Summary of Telecom Industry

Problem Statement:

> Strategies and Solutions" is a detailed exploration of the persistent challenge of customer churn within the telecommunications industry. This resource delves into the factors that drive customer attrition in telecom, including pricing, customer service, competition, and technological advancements, emphasizing the significance of customer retention.
>
> This comprehensive analysis offers a wide array of strategies and innovative solutions for telecom companies to reduce churn rates and enhance customer satisfaction. Whether you are a telecom professional, business owner, or interested in the industry, this publication provides valuable insights and actionable recommendations to address this critical issue. "Understanding Telco Customer Churn" is an indispensable resource for those seeking to improve customer relationships and overall business success in the telecommunications sector.
>
> Customers can choose from multiple service providers in the telecom industry and actively switch from one operator to another. In this highly competitive market, the telecommunications industry experiences an average of 15-25% annual churn rate. Given that it costs 5-10 times more to acquire a new customer than to retain an existing one, customer retention has become even more important than customer acquisition. Retaining high profitable customers is the number one business goal for many incumbent operators.

Overview of Dataset

Source of Dataset - https://www.kaggle.com/datasets/manjit0801/telecom-customer-churn (https://www.kaggle.com/datasets/manjit0801/telecom-customer-churn)

> **The dataset related to customer information is from a telecommunication service provider**

1. customerID: This column contains a unique identifier or code for each customer, allowing you to distinguish between different customers.
2. gender: This column represents the gender of the customer, with values like "Female" or "Male."
3. SeniorCitizen: This is a binary column indicating whether the customer is a senior citizen, with values 0 (usually meaning "No") or 1 (usually meaning "Yes").
4. Partner: This column indicate whether the customer has a partner or spouse, with values like "Yes" or "No."
5. Dependents: This column indicate whether the customer has dependents or family members who rely on the same service, with values like "Yes" or "No."
6. tenure: This column represents the duration of time (in months) that the customer has been with the service provider.
7. PhoneService: This column may indicate whether the customer subscribes to phone service, with values like "Yes" or "No."
8. MultipleLines: If the customer has phone service, this column may indicate whether they have multiple phone lines, with values like "No phone service," "No," or "Yes."
9. InternetService: This column represents the type of internet service the customer has, with options like "DSL" and "Fiber optic."
10. OnlineSecurity: This column indicate whether the customer subscribes to online security services, with values like "No," "Yes," or other variations.
11. OnlineBackup: Similar to the previous column, this indicate whether the customer subscribes to online backup services.
12. DeviceProtection: This column represent whether the customer has device protection services.
13. TechSupport: This column indicate whether the customer subscribes to tech support services.
14. StreamingTV: This column represent whether the customer has streaming TV services.
15. StreamingMovies: Similar to the previous column, this indicate whether the customer has streaming movie services.
16. Contract: This column represents the type of contract the customer has, such as "Month-to-month" or "One year."
17. PaperlessBilling: This indicate whether the customer receives paperless billing, with values like "Yes" or "No."
18. PaymentMethod: This column represents the method of payment used by the customer, such as "Electronic check" or "Bank transfer (automatic)."
19. MonthlyCharges: This is the monthly amount charged to the customer for the service.
20. TotalCharges: This represent the total charges incurred by the customer over the course of their subscription.
21. Churn: This column indicate whether the customer has churned (i.e., canceled their subscription), with values like "Yes" or "No."

Customer Churn Analysis

In [1]:
```
#importing libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

sns.set_theme(color_codes=True)
pd.set_option('display.max_columns', None)
```

In [2]:
```
#importing dataset
df = pd.read_csv("customer_churn.csv")
df
```

Out[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
0	7590-VHVEG	Female	0	Yes	No	1	No	No phor servic
1	5575-GNVDE	Male	0	No	No	34	Yes	N
2	3668-QPYBK	Male	0	No	No	2	Yes	N
3	7795-CFOCW	Male	0	No	No	45	No	No phor servic
4	9237-HQITU	Female	0	No	No	2	Yes	N
...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Ye
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Ye
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phor servic
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Ye
7042	3186-AJIEK	Male	0	No	No	66	Yes	N

7043 rows × 21 columns

Data Cleaning

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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: 1.1+ MB
```

We can clearly see that 'TotalCharges' feature is supposed to be in numeric data-type but it is in object data-type so we will check for any null values present in 'TotalCharges' feature & overall dataset

In [4]:
```python
df.isnull().sum()
```

Out[4]:
```
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
```

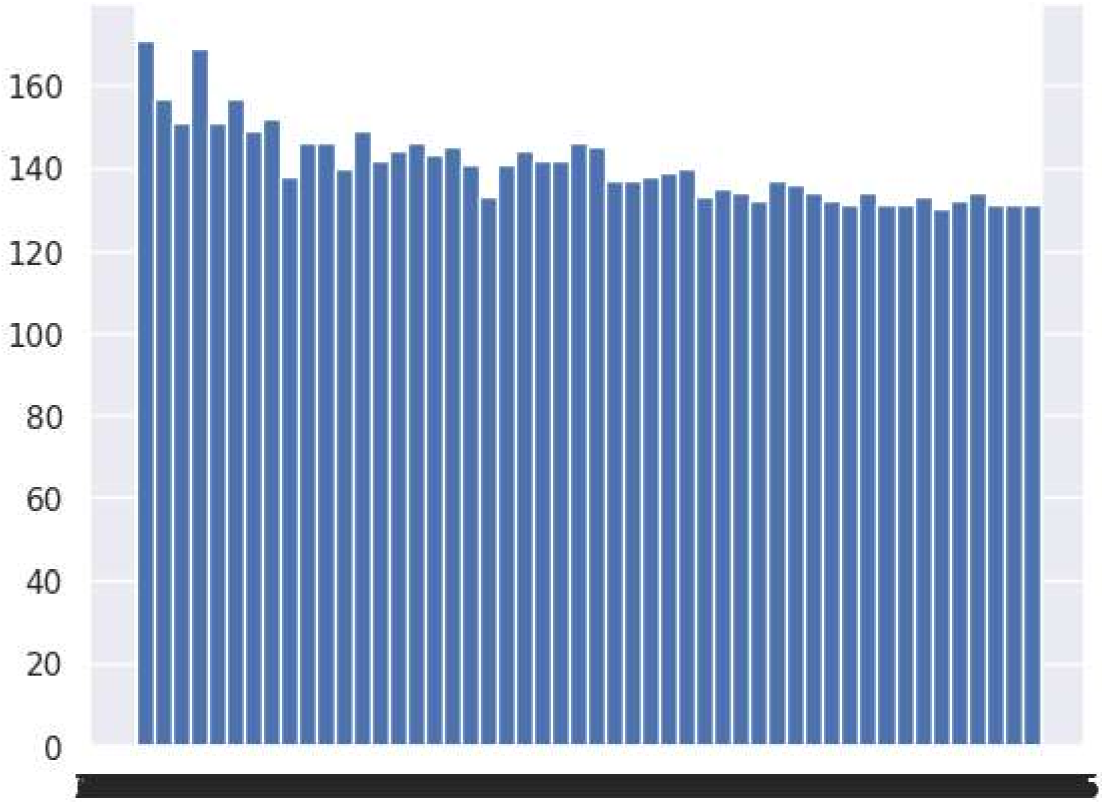
- There are no null values present in our dataset and especially 'TotalCharges' feature.
- Now we will check for empty strings.

In [5]:
```python
df['TotalCharges'].value_counts()
```

Out[5]:
```
           11
20.2       11
19.75       9
20.05       8
19.9        8
           ..
6849.4      1
692.35      1
130.15      1
3211.9      1
6844.5      1
Name: TotalCharges, Length: 6531, dtype: int64
```

Here we can see there are total 11 empty strings present in 'TatalCharges' feature

In [6]:
```
plt.hist(df['TotalCharges'], bins=50);
```



Histogram of 'TotalCharges' is stating that the data is not normally distributed so we will replace it with median

In [7]:
```
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
```

In [8]:
```python
print(df.dtypes)
```

```
customerID           object
gender               object
SeniorCitizen         int64
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 [9]:
```python
df['TotalCharges'] = df['TotalCharges'].fillna(df['TotalCharges'].median())
```

In [10]:
```python
df['TotalCharges'].value_counts()
```

Out[10]:
```
1397.475    11
20.200      11
19.750       9
20.050       8
19.900       8
            ..
6849.400     1
692.350      1
130.150      1
3211.900     1
6844.500     1
Name: TotalCharges, Length: 6531, dtype: int64
```

Exploratory Data Analysis (EDA)

In [11]:
```
#statistics of numeric features
df.describe().drop('SeniorCitizen',axis=1)
```

Out[11]:

	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000
mean	32.371149	64.761692	2281.916928
std	24.559481	30.090047	2265.270398
min	0.000000	18.250000	18.800000
25%	9.000000	35.500000	402.225000
50%	29.000000	70.350000	1397.475000
75%	55.000000	89.850000	3786.600000
max	72.000000	118.750000	8684.800000

If we just focus on the mean of 'tenure', 'MonthlyCharges' & 'TotalCharges' we will be able to see pretty good figures:

- Tenure - 32.37 avg/person i.e. exactly 44.96% of max tenure(72)
- MonthlyCharges - 64.76 avg/person i.e. exactly 54.54% of max MonthlyCharges(118.75)
- TotalCharges - 2281.92 avg/person i.e. exactly 26.28% of max TotalCharges(8684.80)

NOTE: Pecentage calculated using (Part / Whole) * 100

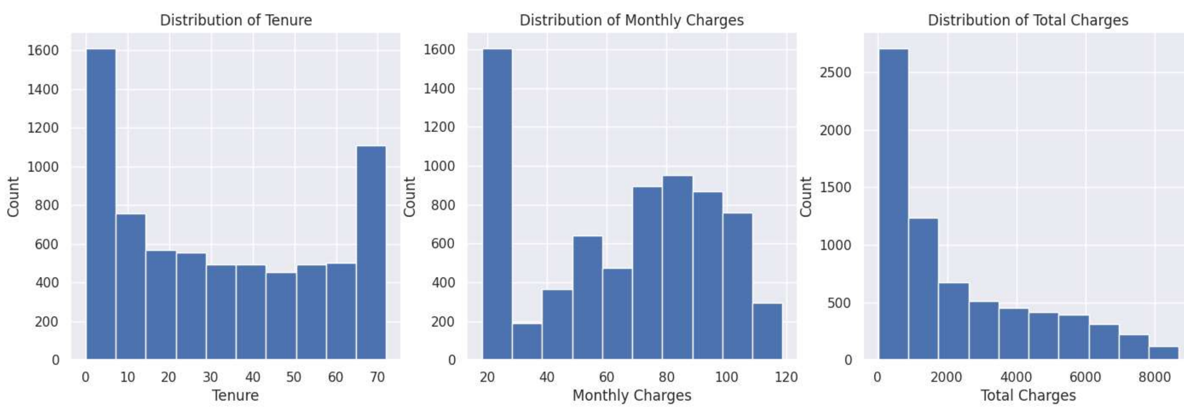
Univariate Analysis

In [12]:
```
plt.figure(figsize=(17,5))

plt.subplot(1,3,1)
plt.hist(df.tenure, bins=10)
plt.title('Distribution of Tenure')
plt.xlabel('Tenure')
plt.ylabel('Count')

plt.subplot(1,3,2)
plt.hist(df.MonthlyCharges, bins=10)
plt.title('Distribution of Monthly Charges')
plt.xlabel('Monthly Charges')
plt.ylabel('Count')

plt.subplot(1,3,3)
plt.hist(df.TotalCharges, bins=10)
plt.title('Distribution of Total Charges')
plt.xlabel('Total Charges')
plt.ylabel('Count');
```

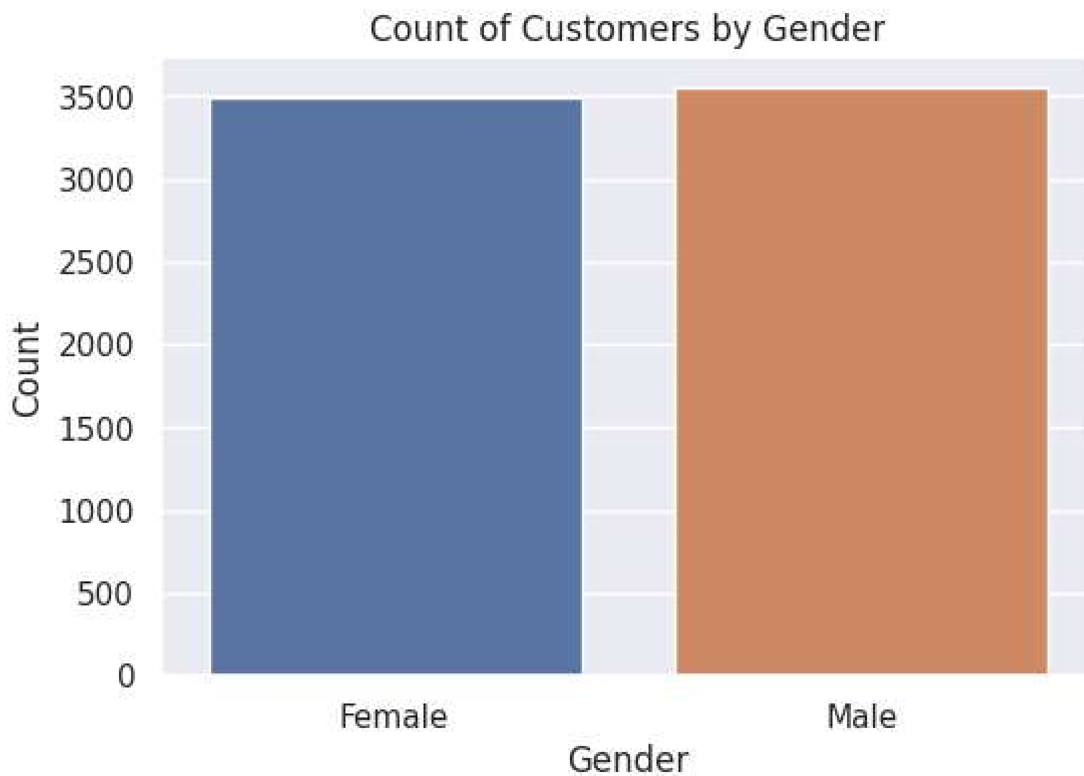


This distribution of different features is helping us to understand:

1. Tenure - The histogram for tenure shows that most customers have been with the company for less than 12 months. There is a smaller peak at around 24 months, and another smaller peak at around 36 months.
2. Monthly charges - The histogram for monthly charges shows a normal distribution, with most customers paying between $100-200 per month. There are a few customers with very high monthly charges, but these are outliers.
3. Total charges - The histogram for total charges shows a right-skewed distribution, with most customers having total charges of less than $2,000. There is a long tail of customers with very high total charges, which is likely due to customers who have been with the company for a long time.

Overall, the bar graph shows that most customers in the telecom churn dataset are relatively new customers with moderate monthly charges and total charges. However, there is a significant minority of customers who have been with the company for a long time and have high total charges.

In [13]:
```
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='gender')
plt.title('Count of Customers by Gender')
plt.xlabel('Gender')
plt.ylabel('Count');
```

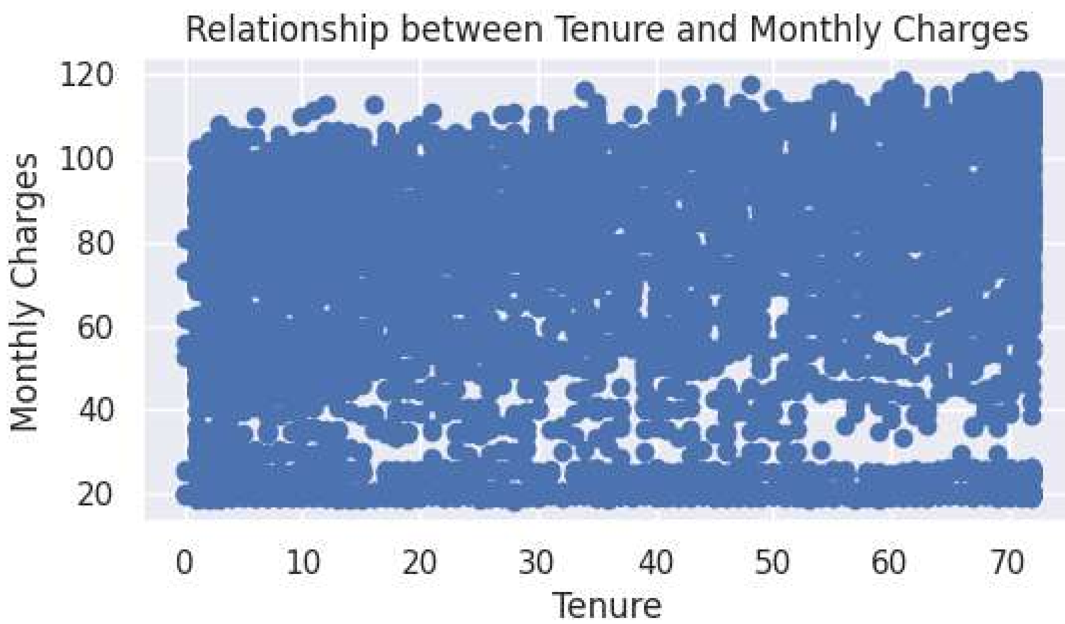


Here are some insights that can be drawn from this count plot:

1. The gender ratio of the customer base is not evenly balanced. There are more male customers than female customers.
2. The gender ratio of the customer base may vary depending on the industry or type of business.

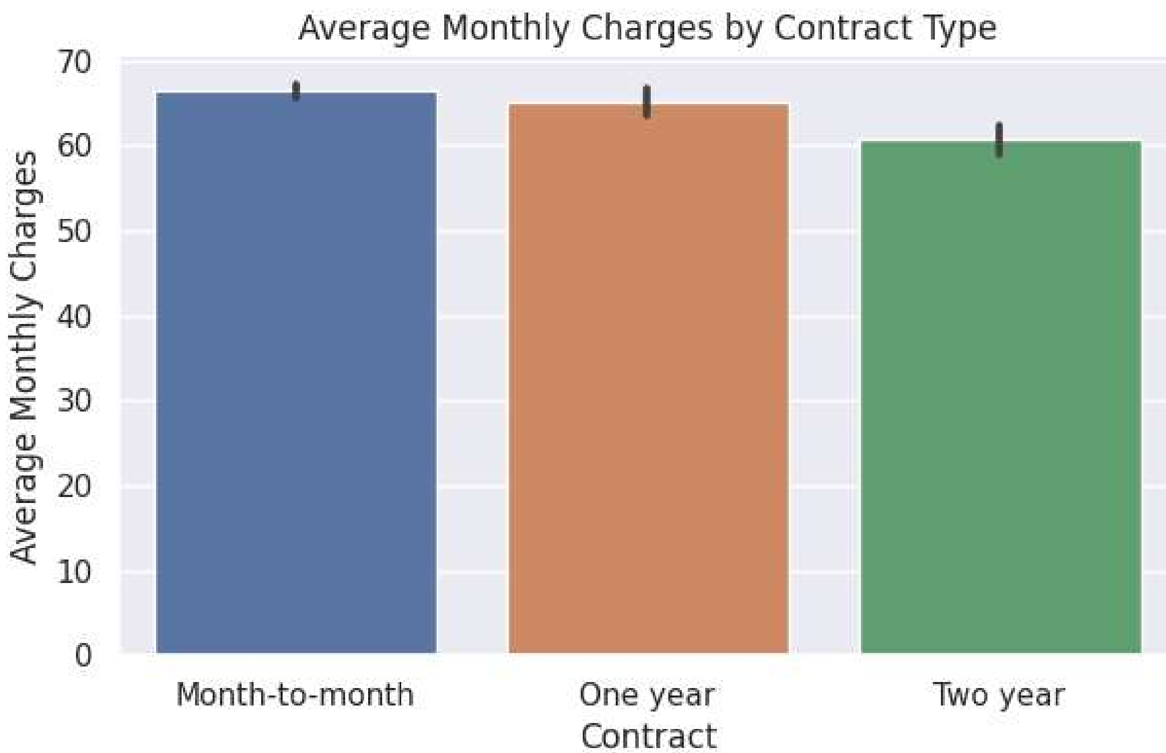
In [14]:

```
plt.figure(figsize=(6, 3))
plt.scatter(df.tenure, df.MonthlyCharges)
plt.title('Relationship between Tenure and Monthly Charges')
plt.xlabel('Tenure')
plt.ylabel('Monthly Charges');
```



Telecom customers with longer tenure tend to have higher monthly charges. This may be because they have had more time to add additional services to their accounts, or because they are on older pricing plans that are more expensive than the current plans.

In [15]:
```
plt.figure(figsize=(7, 4))
sns.barplot(data=df, x='Contract', y='MonthlyCharges')
plt.title('Average Monthly Charges by Contract Type')
plt.xlabel('Contract')
plt.ylabel('Average Monthly Charges');
```

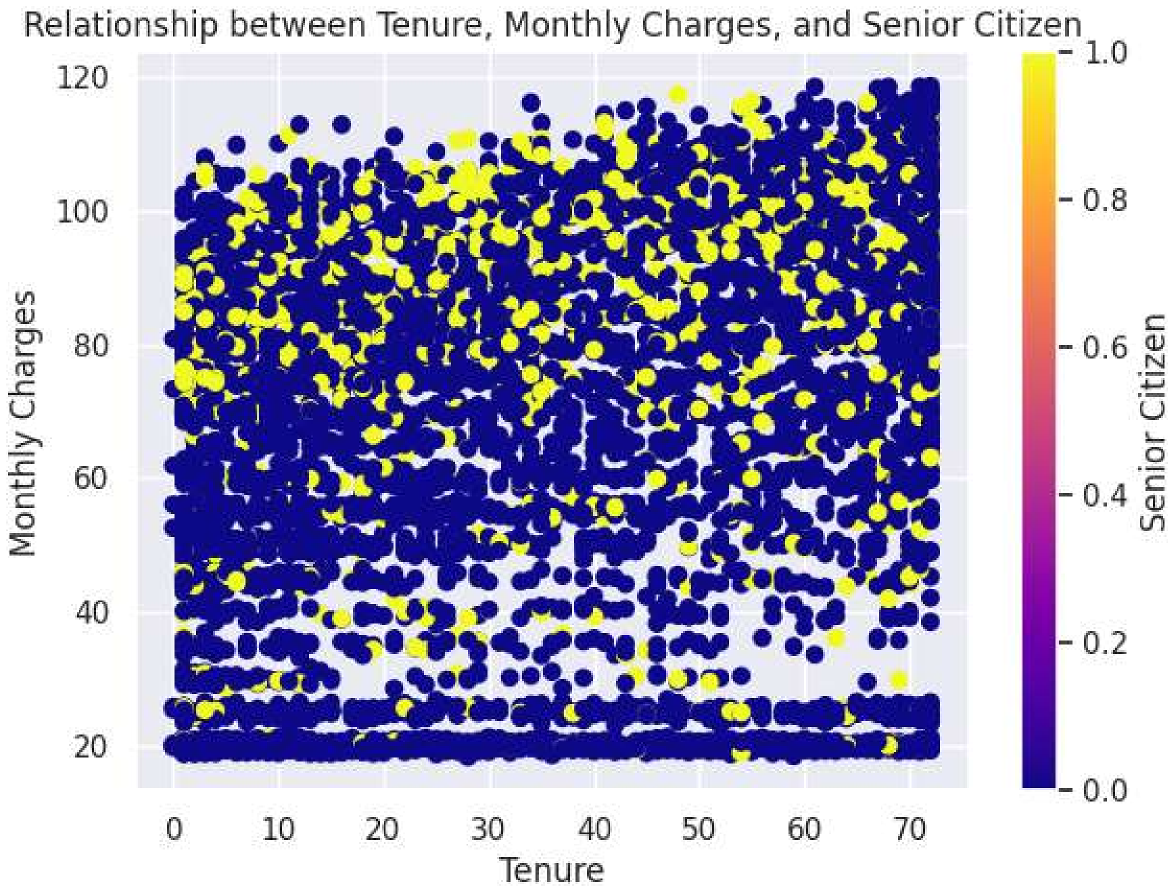


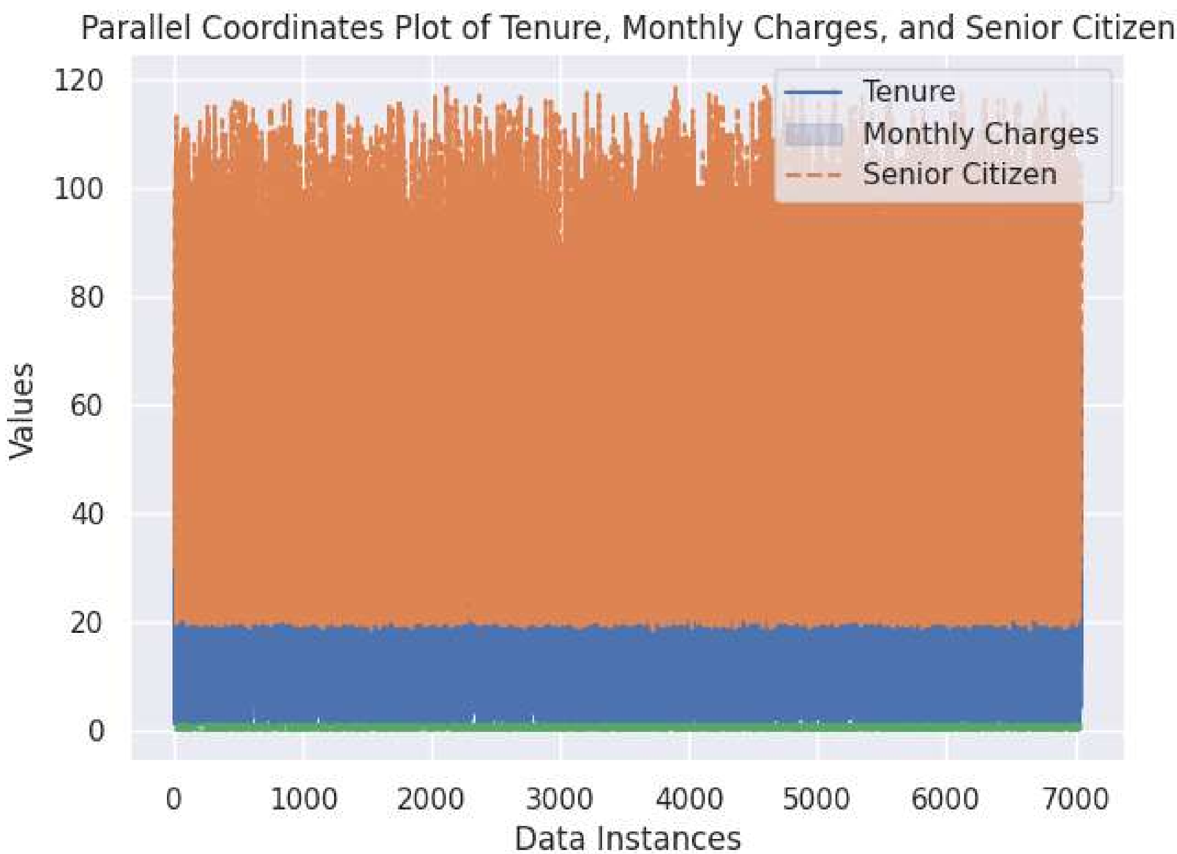
These are some insights that can be drawn from the bar plot:

1. The difference in average monthly charges between customers with two-year contracts and customers with month-to-month contracts is significant. This suggests that customers with two-year contracts are willing to pay a premium for the stability of having a fixed monthly price.
2. The difference in average monthly charges between customers with one-year contracts and customers with month-to-month contracts is smaller. This suggests that customers with one-year contracts are more price-sensitive than customers with two-year contracts.

In [16]:
```
plt.figure(figsize=(7, 5))
plt.scatter(data=df, x='tenure', y='MonthlyCharges', c='SeniorCitizen', cmap
='plasma')
plt.title('Relationship between Tenure, Monthly Charges, and Senior Citizen')
plt.xlabel('Tenure')
plt.ylabel('Monthly Charges')
plt.colorbar(label='Senior Citizen')

plt.figure(figsize=(7, 5))
sns.lineplot(data=df[['tenure', 'MonthlyCharges', 'SeniorCitizen']])
plt.title('Parallel Coordinates Plot of Tenure, Monthly Charges, and Senior Ci
tizen')
plt.xlabel('Data Instances')
plt.ylabel('Values')
plt.legend(['Tenure', 'Monthly Charges', 'Senior Citizen']);
```


Relationship between Tenure, Monthly Charges, and Senior Citizen
Monthly Charges
Tenure
Senior Citizen
120
100
80
60
40
20
0
10
20
30
40
50
60
70
0.0
0.2
0.4
0.6
0.8
1.0


Parallel Coordinates Plot of Tenure, Monthly Charges, and Senior Citizen
Tenure
Monthly Charges
Senior Citizen
Values
Data Instances
120
100
80
60
40
20
0
1000
2000
3000
4000
5000
6000
7000

- There is a positive correlation between tenure and monthly charges for both senior citizens and non-senior citizens.
- However, the correlation is stronger for senior citizens. This suggests that senior citizens are more likely to have higher monthly charges than non-senior citizens with the same tenure.
- There are a few possible explanations for this difference.

> 1. One possibility is that senior citizens are more likely to be on older pricing plans that are more expensive than the current plans.
> 2. Another possibility is that senior citizens are more likely to subscribe to premium services, such as more phone lines, more internet bandwidth, or more premium TV channels.

Following can be actionable insights to reduce churn rate:

1. A telecom company may want to offer special discounts or promotions to customers who have been with the company for a long time or to customers who have high monthly charges.
2. The correlation between tenure and monthly charges is something that telecom companies should be aware of. Telecom companies can use this information to develop pricing plans and marketing strategies that are targeted to customers with different tenure lengths. For example, they may want to offer discounts to new customers or to customers who are willing to switch to a newer pricing plan.
3. Telecom companies may want to offer discounts to customers who are willing to sign up for a two-year contract. They may also want to offer more flexible pricing plans to customers who are more price-sensitive.
4. Telecom companies may want to offer discounts to senior citizens or to senior citizens who are willing to switch to a newer pricing plan.

Here is one interesting insight we found during analysis:

- The gender ratio of the customer base may vary depending on the industry or type of business. Businesses should use this information to better understand their customer demographics and to develop targeted marketing campaigns.
  - For example, a business that sells products that are traditionally considered to be "masculine" may have a higher proportion of male customers or
  - For example, a business that sells products that are more likely to be purchased by women may want to focus their marketing efforts on female customers.

Customer Churn Modeling

Feature Engineering

In [17]:
```
tenure = df['tenure']
monthly_charges = df['MonthlyCharges']

interaction_feature = tenure * monthly_charges

df['TotalMonetaryValue'] = interaction_feature
```

In [18]:
```
numerical_columns = ['tenure', 'MonthlyCharges', 'TotalCharges']

#calculating the IQR for each numerical column
Q1 = df[numerical_columns].quantile(0.25)
Q3 = df[numerical_columns].quantile(0.75)
IQR = Q3 - Q1

#defining the lower and upper bounds for outlier detection
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

#handling outliers by limiting/extending them to a certain range
for column in numerical_columns:
    df[column] = df[column].clip(lower_bound[column], upper_bound[column])

print(df)
```

```
      customerID  gender  SeniorCitizen Partner Dependents  tenure  \
0     7590-VHVEG  Female              0     Yes         No       1   
1     5575-GNVDE    Male              0      No         No      34   
2     3668-QPYBK    Male              0      No         No       2   
3     7795-CFOCW    Male              0      No         No      45   
4     9237-HQITU  Female              0      No         No       2   
...          ...     ...            ...     ...        ...     ...   
7038  6840-RESVB    Male              0     Yes        Yes      24   
7039  2234-XADUH  Female              0     Yes        Yes      72   
7040  4801-JZAZL  Female              0     Yes        Yes      11   
7041  8361-LTMKD    Male              1     Yes         No       4   
7042  3186-AJIEK    Male              0      No         No      66   

     PhoneService     MultipleLines InternetService OnlineSecurity  \
0              No  No phone service             DSL             No   
1             Yes                No             DSL            Yes   
2             Yes                No             DSL            Yes   
3              No  No phone service             DSL            Yes   
4             Yes                No     Fiber optic             No   
...           ...               ...             ...            ...   
7038          Yes               Yes             DSL            Yes   
7039          Yes               Yes     Fiber optic             No   
7040           No  No phone service             DSL            Yes   
7041          Yes               Yes     Fiber optic             No   
7042          Yes                No     Fiber optic            Yes   

     OnlineBackup DeviceProtection TechSupport StreamingTV StreamingMovies  \
0             Yes               No          No          No              No   
1              No              Yes          No          No              No   
2             Yes               No          No          No              No   
3              No              Yes         Yes          No              No   
4              No               No          No          No              No   
...           ...              ...         ...         ...             ...   
7038           No              Yes         Yes         Yes             Yes   
7039          Yes              Yes          No         Yes             Yes   
7040           No               No          No          No              No   
7041           No               No          No          No              No   
7042           No              Yes         Yes         Yes             Yes   

            Contract PaperlessBilling              PaymentMethod  \
0     Month-to-month              Yes           Electronic check   
1           One year               No               Mailed check   
2     Month-to-month              Yes               Mailed check   
3           One year               No  Bank transfer (automatic)   
4     Month-to-month              Yes           Electronic check   
...              ...              ...                        ...   
7038        One year              Yes               Mailed check   
7039        One year              Yes    Credit card (automatic)   
7040  Month-to-month              Yes           Electronic check   
7041  Month-to-month              Yes               Mailed check   
7042        Two year              Yes  Bank transfer (automatic)   

      MonthlyCharges  TotalCharges Churn  TotalMonetaryValue  
0              29.85         29.85    No               29.85  
1              56.95       1889.50    No             1936.30  
2              53.85        108.15   Yes              107.70  
3              42.30       1840.75    No             1903.50  
```

```
4             70.70       151.65   Yes           141.40
...             ...          ...   ...              ...
7038          84.80      1990.50    No          2035.20
7039         103.20      7362.90    No          7430.40
7040          29.60       346.45    No           325.60
7041          74.40       306.60   Yes           297.60
7042         105.65      6844.50    No          6972.90

[7043 rows x 22 columns]
```

Splitting the dataset

In [19]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
# Selecting features for X
features = ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines',
            'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
            'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges']

X = df[features]

# Selecting target variable for y
y = df['Churn']

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [21]:
```
print("Training data shape:")
print(X_train.shape)
print(y_train.shape)

print("\nTesting data shape:")
print(X_test.shape)
print(y_test.shape)
```

```
Training data shape:
(5634, 19)
(5634,)

Testing data shape:
(1409, 19)
(1409,)
```

In [22]:
```
categorical_columns = ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'te
nure',
                       'PhoneService', 'MultipleLines', 'InternetService', 'On
lineSecurity',
                       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'Str
eamingTV',
                       'StreamingMovies', 'Contract', 'PaperlessBilling', 'Pay
mentMethod']

from sklearn.preprocessing import OneHotEncoder

#applying one-hot encoding to the categorical columns
encoder = OneHotEncoder(handle_unknown='ignore')
X_train_encoded = encoder.fit_transform(X_train[categorical_columns])

#fitting and transforming the encoded categorical columns
X_test_encoded = encoder.transform(X_test[categorical_columns])
```

In [23]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifie
r
from sklearn.metrics import accuracy_score

#Initializing and training the models
model1 = LogisticRegression()
model2 = RandomForestClassifier()
model3 = GradientBoostingClassifier()

model1.fit(X_train_encoded, y_train)
model2.fit(X_train_encoded, y_train)
model3.fit(X_train_encoded, y_train)

#making predictions on the testing data
y_pred1 = model1.predict(X_test_encoded)
y_pred2 = model2.predict(X_test_encoded)
y_pred3 = model3.predict(X_test_encoded)

#evaluating the models
accuracy1 = accuracy_score(y_test, y_pred1)
accuracy2 = accuracy_score(y_test, y_pred2)
accuracy3 = accuracy_score(y_test, y_pred3)

print("Model 1 Accuracy:", accuracy1)
print("Model 2 Accuracy:", accuracy2)
print("Model 3 Accuracy:", accuracy3)

#done with modeling
model1 = LogisticRegression(max_iter=1000)
model2 = LogisticRegression(max_iter=1000)
model3 = LogisticRegression(max_iter=1000)

model1 = LogisticRegression(solver='liblinear')
model2 = LogisticRegression(solver='liblinear')
model3 = LogisticRegression(solver='liblinear')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:45
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  n_iter_i = _check_optimize_result(

Model 1 Accuracy: 0.8062455642299503
Model 2 Accuracy: 0.7877927608232789
Model 3 Accuracy: 0.801277501774308
```

In [26]:
```
from sklearn.metrics import classification_report, confusion_matrix

#model 1
print("Model 1 Metrics:")
print(classification_report(y_test, y_pred1))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred1))
print()

#model 2
print("Model 2 Metrics:")
print(classification_report(y_test, y_pred2))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred2))
print()

#model 3
print("Model 3 Metrics:")
print(classification_report(y_test, y_pred3))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred3))
```

```
Model 1 Metrics:
              precision    recall  f1-score   support

          No       0.85      0.90      0.87      1036
         Yes       0.66      0.56      0.60       373

    accuracy                           0.81      1409
   macro avg       0.75      0.73      0.74      1409
weighted avg       0.80      0.81      0.80      1409

Confusion Matrix:
[[928 108]
 [165 208]]

Model 2 Metrics:
              precision    recall  f1-score   support

          No       0.83      0.90      0.86      1036
         Yes       0.63      0.49      0.55       373

    accuracy                           0.79      1409
   macro avg       0.73      0.69      0.71      1409
weighted avg       0.78      0.79      0.78      1409

Confusion Matrix:
[[928 108]
 [191 182]]

Model 3 Metrics:
              precision    recall  f1-score   support

          No       0.84      0.90      0.87      1036
         Yes       0.66      0.51      0.58       373

    accuracy                           0.80      1409
   macro avg       0.75      0.71      0.72      1409
weighted avg       0.79      0.80      0.79      1409

Confusion Matrix:
[[937  99]
 [181 192]]
```

Model Selection

In [25]:
```python
#let's compare the accuracy of the models
accuracies = [accuracy1, accuracy2, accuracy3]

#finding the index of the model with the highest accuracy
best_model_index = accuracies.index(max(accuracies))

#determining the best-performing model based on the index
if best_model_index == 0:
    best_model = model1
    best_model_name = "Model 1"
elif best_model_index == 1:
    best_model = model2
    best_model_name = "Model 2"
else:
    best_model = model3
    best_model_name = "Model 3"

#Printing best performing model and its accuracy
print("Best Model:", best_model_name)
print("Best Model Accuracy:", max(accuracies))
```

```
Best Model: Model 1
Best Model Accuracy: 0.8062455642299503
```

Conclusion

After evaluating the performance of all three models, it has been concluded that the logistic regression model demonstrates superior accuracy compared to the other models. Hence, the logistic regression model is selected as the best-performing model.

NOTE: This notebook is performed on Google Colab